Reference

Reference

Gentics Portal.Node 4.7.2
Build 20150528-121047 20150528-121047

Published 2014

Table of Contents

List of Tables

List of Examples

Chapter 1. Introduction

This document should support people implementing, maintaining, working or developing with PN4 (Gentics Portal.Node). It should be the main and first reference to look at if you have specific questions or problems with a portal implementation or don't know how to do something specific.

Please consult the *SDK Guide introduction chapter* [http://www.gentics.com/help/topic/com.gentics.portalnode.sdk.eclipseplugin/misc/doc/sdkguide/xhtml/sdkguide-intro.html] for a list of further documentation sources.

Chapter 2. Installation

2.1. Introduction

Easy installation, stand-alone or deployable into nearly every enterprise application server. Apache Tomcat is most frequently used, but every Servlet compliant application server is supported.

2.2. Requirements

2.2.1. Platform

2.2.1.1. Standard stand-alone recommendation

These is a common platform recommendation for a standard stand-alone installation. Any other components fitting the requirements will do as well:

Table 2.1. Platform recommendations

Package	URL
Debian Linux 5.0	http://www.debian.org/
Sun J2SE 5.0 JDK	http://java.sun.com/j2se/1.5.0/
Apache Tomcat 5.5	http://jakarta.apache.org/tomcat/
MySQL 5.0	http://dev.mysql.com/downloads/mysql/

Please consult the documentation on the vendors homepages for installation instructions. For failsafe or high volume installations please contact a Gentics consultant.

2.2.1.2. Requirements

- Runtime Environment:

  J2SE 5.0 compliant, J2EE 5.0 support, Servlet 2.4 compliant application server platform (Apache Tomcat 5.5, BEA WebLogic, IBM Webshpere, Jetty (on JBoss), JBoss...)

- Database:

  SQL-92 compliant, JDBC compliant database (Oracle 10g, IBM DB2, Microsoft SQL Server, MySQL 5.0, ...). For a more complete list of supported and tested databases, and restrictions see the Gentics Infoportal [http://www.gentics.com/Content.Node/infoportal/knowledgebase/faq/faq_new.php?faq_link=/Content.Node/infoportal/knowledgebase/faq/welche-datenquellen-sind-kompatibel.php]

- Operation System:

  100% OS independent (Microsoft Windows, Linux, Sun Solaris, ...)

- Hardware:

  100% hardware platform independent (Intel, Sparc, PowerPC, ...)

- Libraries:

  Gentics Portal.Node is shipped with a zip-file containing shared libraries. Please do not replace these

libraries (e.g. with other versions) since we can only guarantee compatibility and correct operation only with the shipped libraries.

2.3. Quick Installation

This is short list of the major steps to do when installing Gentics Portal.Node. Refer to the next chapter for detailed instructions.

1. Extract `gentics-portalnode-3-sharedlibs.zip` to `${catalina.home}/shared/lib/`.
2. The libraries found in the directory `${catalina.home}/shared/lib/endorsed/` define APIs which possibly also exist in the JVM in other versions. Therefore those libraries need to override the JVM APIs to avoid version conflicts. This can be achieved in one of the two possible ways:
   - Either copy those libraries to the configured JVM endorsed directory (which would normally be <java-home>/lib/endorsed)
   - Or define the system environment variable *`$JAVA_ENDORSED_DIR`* to point to the directory `${catalina.home}/shared/lib/endorsed/`.
3. Copy `gentics/` to `${catalina.home}/conf/`.
4. Finish application server configuration:
   - manager user in tomcat and portalconfiguration
   - emptySessionPath="true" in connector config (server.xml) for Apache Tomcat 6 or sessionCookiePath="/" in global context configuration (context.xml) for Apache Tomcat 7+
   - JVM memory in startup scripts
5. Deploy `Portal.Node.war`.
6. (Optionally) deploy `GenticsImageStore.war`.
7. Open `http://[hostname]/Portal.Node/` in your browser.
8. Follow the instructions to activate the portal by downloading `license.key` file.

2.4. Installation

1. Copy the shared libraries into the classpath of the application server. For Apache Tomcat, copy the libraries into the folder `${catalina.home}/shared/lib/`. When using Tomcat 5.0 you have to copy the database driver classes (eg. JDBC Driver) to `common/lib`.

2. The libraries found in the directory `${catalina.home}/shared/lib/endorsed/` define APIs which possibly also exist in the JVM in other versions. Therefore those libraries need to override the JVM APIs to avoid version conflicts. This can be achieved in one of the two possible ways:

   - Either copy those libraries to the configured JVM endorsed directory (which would normally be <java-home>/lib/endorsed)

   - Or define the system environment variable *`$JAVA_ENDORSED_DIR`* to point to the directory `${catalina.home}/shared/lib/endorsed/`.

3. Copy the content of the Gentics configuration directory into `${com.gentics.portalnode.confpath}`. See Section 3.2, “System properties” for details on the configuration path.

4. Modify your application server's configuration:

   - Make sure the application server does not include the context paths into the generated session cookies.

     - For Apache Tomcat 6: Set *`emptySessionPath="true"`* for all *`<Connector>`* -nodes in the file `tomcat/conf/server.xml`.

     - For Apache Tomcat 7+: Configure *`sessionCookiePath="/"`* in the global context.xml `tomcat/conf/context.xml`.

   - For Apache Tomcat create a user account with access to the manager application and add the login data together with the access data for the manager into your portal configuration file. For details see Section 3.8.8, “administration-section”

5. Deploy the Portal.Node.war on your application server.

   a. In Tomcat you can use /manager/html to upload and deploy the webapp. Take care you stop and undeploy any already existing instances in before. It's also suggested to restart Tomcat after redeploying an application, see http://jakarta.apache.org/tomcat/faq/deployment.html for details.

   b. Your application server should be configured to provide the maximum amount of heap memory possible (*`-Xmx`* see **man java** for details). The minimum is 512MB (-Xmx512M). The maximum on a 32bit platform is about 1,5GB (-Xmx1536M). On dedicated 64bit platforms you should provide nearly all you have.

6. (Optionally) deploy the GenticsImageStore.war on your application server (see above).

7. Open the url `http://[hostname]/Portal.Node` on your browser (substitute [hostname] with the servers hostname or ip address).

8. When not already done before, follow the shown instructions to activate your copy of Portal.Node. You need a valid License Key (e.g. A234-B678...) to activate your installation, it can be obtained from `<sales@gentics.com>` .

Note

This activation process will only work if you access Gentics Portal.Node directly. If a reverse proxy is configured in front of Gentics Portal.Node which might replace the 'Host' header from the request, please use the following manual activation process.

If you get a HTTP 503 (service unavailable) error instead of the activation instructions, you do not have activated the activation dialog. In this case, you can either turn on the activation dialog (see Section 3.8.2.3, “Portal server activation” for details) or find the activation instructions in your portal server's log file (for Tomcat typically `${CATALINA_HOME}/logs/catalina.out`).

Warning

To finish the installation Process you will have to activate the product with your license key. Unless activated properly the portal will keep showing the activation instructions or - when the activation dialog is turned off in the portal configuration - responding with a HTTP 503 (service unavailable) error and write an error message in the logfile.

Chapter 3. Administration and Configuration

3.1. Introduction

Easy and central fileconfiguration of infrastructure and low level settings.

3.2. System properties

Some configuration parameters allow usage of system properties, for example to refer to special paths.

Table 3.1. System properties

Name	Description
com.gentics.portalnode.home	Absolute path to the root of the main Portal.Node webapplication. Some parts of the default configuration shipped with Portal.Node are located there.
com.gentics.portalnode.confpath	Absolute path of the default configuration directory. For installations using Jakarta Tomcat 5, this will usually be `${catalina.home}/conf/gentics`
com.gentics.portalnode.apps.{applicationcontext}.path	Absolute path to the root of the deployed portlet application {applicationcontext}
com.gentics.portalnode.security.accesscontrol.class	Classname of an implementation of the interface com.gentics.api.portalnode.security.AccessControl for access control when Gentics Portal.Node runs in a secure environment.

3.3. Authentication

3.3.1. Introduction

Authentication Managers are used to integrate use repositories (identity management) and signon methods (access management).

When more than one authentication manager is configured, one of them has to be made the ***`default`*** and will serve as primary authentication system, whereas all others are secondary authentication systems. The primary authentication system is responsible for the access management and any secondary authentication systems define extending user repositories for storing additional data. See Portal Property Paths (Section 4.8.1, “Portal”) to learn how to access each Authentication Manager's properties separately.

3.3.2. Configuration

The configuration consists of two parts

1. *Authentication Descriptors* that defines how users are authenticated and how parameters from the user repositories are mapped onto user parameters in the portal.

2. *Anonymous Parameters* to defined the user profile for users that are not logged in.

See Section 3.8.3, “authentication-section” for details on configuration of the authentication managers.

3.3.3. Gentics Content.Node Portal Authentication Using CAS SSO

3.3.3.1. Overview

This overview will describe how frontend editing will work with CAS authentication. An authenticated portal user with the appropriate permissions will be able to request resources from the Gentics Content.Node Server. Those requests will be dispatched using the “GCNProxyServlet” (Section 6.2, p.226). This servlet enables the client to request data via ajax from the Gentics Content.Node server. Otherwise client browsers would block those requests due to cross site ajax limitations. Additionally the “GCNProxyServlet” (Section 6.2, p.226) modifies the request so that the response data can be used within the context of your portal application.

CAS will also be used to authenticate the portal user against Content.Node. CAS Proxy authentication will be used in order to overcome the limitations of the separation between backend and frontend services. The “GCNCasProxyAuthenticationServletFilter” (Section 7.3, p.227) will add the needed Content.Node authentication information to the passed request. The filter itself will dispatch a CAS proxy request to Gentics Content.Node in order to collect the needed authentication information. Those information will be added to the filtered request.

You'll have to configure your portal and Gentics Content.Node server in order to enable Gentics Content.Node authentication for your portal implementation.

Components that have to be configured:

- CAS - CAS Server
- Portal - CAS Filter
- Portal - “GCNCasProxyAuthenticationServletFilter” (Section 7.3, p.227)
- Portal - “GCNProxyServlet” (Section 6.2, p.226)
- CMS - “Example Gentics Content.Node Configuration” (Section 3.3.3.3, p.22)

3.3.3.2. Brief Step By Step Guide

3.3.3.2.1. JASIG CAS Server

The CAS server must be configured in order to work with cas proxy authentication. Please refer to the cas documentation. (e.g: A HttpBasedServiceCredentialsToPrincipalResolver credential resolver bean is mandatory). The CAS server must be able to communicate with the Portal server over HTTPS. You may need to import the used Portal SSL certificate within the keystore that is used for the CAS server.

3.3.3.2.2. Gentics Content.Node

The Gentics Content.Node server must be able to access the CAS server via HTTPS. Importing the CAS ssl certificate in the JDK keystore is needed in order to avoid problems with ssl certificates authentication. The CAS server must be able to communicate with the Gentics Content.Node backend server. This is needed because the CAS filters will try to validate incoming proxy tickets against the CAS server. It's also mandatory to configure the CAS SSO acording to provided guides. This includes filter configuration on the Gentics Content.Node backend server. Please refer to the provided Gentics Content.Node documentation.

3.3.3.2.3. Gentics Portal.Node

The Portal server must be able to access the Gentics Content.Node server via HTTP and it must be able

to communicate with the CAS Server over HTTPS. You may have to install the CAS server SSL certificate in the keystore of the JDK as well.

Note: You may have to add the Portal SSL certificate when you deploy the CAS Webapp on the same system and both share one certificate.

The CAS filters and the GCN authentication filters have to be configured. A example configuration can be found here: “Example configuration” (Section 7.3.4, p.228)

Please note that the “GCNProxyServlet” (Section 6.2, p.226) is now a part of Portal.Node. The package path was changed to com.gentics.portalnode.proxy.GCNProxyServlet

The configuration of the “GCNCasProxyAuthenticationServletFilter” (Section 7.3, p.227) can be found here.

Note: You'll have to change the DispatchFilter ignore pattern when you want to place the GCNProxyServlet within the Gentics Portal.Node Webapp configuration.

3.3.3.3. Example Gentics Content.Node Configuration

Please configure your Gentics Content.Node Server according to the Gentics Content.Node documentation. An example configuration witch is CAS proxy enabled could look like this.

Example 3.1. Syntax Example Gentics Content.Node Configuration

```
<!-- =============== CAS SSO Filter Start =============== -->
<!-- Filter parameter shared by some filters -->
<context-param>
        <param-name>casServerLoginUrl</param-name>
        <param-value>https://mycas/cas-server/login</param-value>
</context-param>
<context-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://mycas/cas-server</param-value>
</context-param>
<context-param>
        <param-name>serverName</param-name>
        <param-value>http://cms.my-gcn-server</param-value>
</context-param>

<!-- The authentication filter will check, whether CAS Authentication is
        necessary and will redirect to the CAS Server -->
<filter>
        <filter-name>CAS Authentication Filter</filter-name>
        <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter<
        <init-param>
                <param-name>gateway</param-name>
                <param-value>false</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CAS Authentication Filter</filter-name>
        <url-pattern>/rest/auth/login</url-pattern>
</filter-mapping>

<!-- The Ticket Validation Filter will validate the Ticket (found in the
        query as additional parameter) against the CAS Server and will do a red
        to the service without the ticket -->
<filter>
        <filter-name>CAS Ticket Validation Filter</filter-name>
        <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicket
        <init-param>
                <param-name>acceptAnyProxy</param-name>
                <param-value>true</param-value>
```

```
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CAS Ticket Validation Filter</filter-name>
        <url-pattern>/rest/auth/login</url-pattern>
</filter-mapping>

<!-- The Assertion Thread Local Filter makes the assertion available as
        ThreadLocal (the CASIntegrationFilter depends on it) -->
<filter>
        <filter-name>CAS Assertion Thread Local Filter</filter-name>
        <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</fil
</filter>
<filter-mapping>
        <filter-name>CAS Assertion Thread Local Filter</filter-name>
        <url-pattern>/rest/auth/login</url-pattern>
</filter-mapping>

<!-- The CASIntegrationFilter reads the assertion, matches it against the
        User-DB (or creates a new user). And performs the login. -->
<filter>
        <filter-name>CAS GCN Integration Filter</filter-name>
        <filter-class>com.gentics.contentnode.auth.filter.CASIntegrationFilter<
        <init-param>
                <param-name>initGroups</param-name>
                <param-value>[4, 6]</param-value> <!-- expression that returns
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CAS GCN Integration Filter</filter-name>
        <url-pattern>/rest/auth/login</url-pattern>
</filter-mapping>
<!-- =============== CAS SSO Filter End =============== -->
```

Table 3.2. Settings Example Gentics Content.Node Configuration

Name	Type	Default	Description
No Settings.			

For more information about cas proxy authentication please take a look at the following walkthrough: https://wiki.jasig.org/display/CAS/Proxy+CAS+Walkthrough

3.4. Datasources

3.4.1. Introduction

Datasources are an abstraction layer to handle different types of data storages in a type-independent way. Query by generalized rules, versioning, futurechanges, fallback, multiple servers are some feature examples. See Section 4.13, “Datasource” to learn about implementation of datasources.

3.4.2. Configuration

Datasources are configured in two steps:

- `Datasource handles` define the connections to specific data repositories (e.g. ldap, sql databases, ...)

- `Datasources` use one or multiple handles and define the abstract storage mechanism that portlets can use.

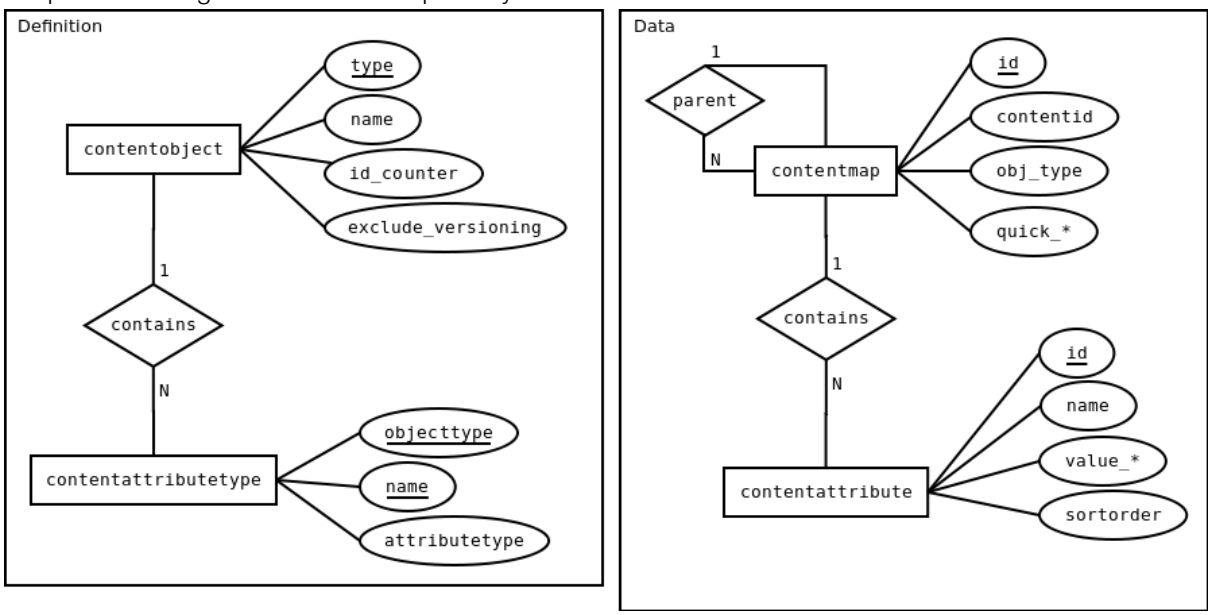
For details on the configuration see Section 3.8.6, “datasource-section”

3.4.3. ContentRepository

3.4.3.1. Introduction

A simple database scheme, used for storing content published by Gentics Content.Node and all standard modules provided by Gentics. The ContentRepository may be versioning. You don’t need to access the repository by hand. Use Datasource for reading, the AdministrationPortlet for managing the data structure and Views for visualisation and data management.

Simplified ER diagram of a contentrepository:



A detailed SQL table layout can be found below.

3.4.3.2. Tables

3.4.3.2.1. contentobject

Stores definitions of object types

Table 3.3. Columns of table `contentobject`

Column	Description
type	Unique type of this object. Gentics range is from 0 to 59999. Customertypes should have ranges of 1000 beginning from 100000. Integer, unique.
name	Name of the object, preferable an i18n key, otherwise readable and displayed as-is. String, unique.
id_counter	counter for object-ids (for generating unique object ids), managed by the Datasource of type `contentrepository`
exclude_versioning	Flag to mark objecttypes that are excluded from versioning.

3.4.3.2.2. contentattributetype

Stores definitions of attributes for object types

Table 3.4. Columns of table `contentattributetype`

Column	Description
name	Internal name of an attribute. String, unique per object.
attributetype	Type of this attribute. All attributes with same name must have same type, independent of their objecttype. See “Attribute Types” (Table 4.169, p.199) for a description of the possible attribute types.
optimized	If this column is optimized for faster read access in the contentmap table. 1 or 0
quickname	The name of the column in the contentmap table, where this column is stored. Prefixed with "quick_". Column must be added and indexed manually if managed manually. Gentics Content.Node; Content.Repositories do this automaticaly. 1 or 0
multivalue	If this attribute is multivalue. Multivalue attributes must not be optimized. Multivalue attributes are represented by multi contentattribute rows with an optional sortorder. 1 or 0
objecttype	Which object this attribute belongs to. Integer, Relation to contentobject.type
linkedobjecttype	Which object this attribute links to (attributetype 2 & 7). Integer
foreignlinkattribute	In which attribute of the foreign object, the link to the current object is stored in (attributetype 7). Name of attribute. Text
foreignlinkattributerule	Optional rule to restrict the objects related with this attribute (attributetype 7). You may use the object.attribute syntax to refer to the foreign object, but currently there is no way to access your source object. Text
exclude_versioning	Flag to mark attributetypes that are excluded from versioning.

3.4.3.2.3. contentmap

Stores objects (one object per row)

Table 3.5. Columns of table `contentmap`

Column	Description
id	Unique row id (required for versioning)
contentid	Contentid of the object (unique)
obj_id	Id of the object (unique per obj_type)
obj_type	Type of the object (refers to types in table contentobject)
motherid	contentid of the mother object
mother_obj_id	object id of the mother object
mother_obj_type	Type of the mother object
updatetimestamp	Timestamp of the last change in this row
quick_...	optional quick columns (for every optimized attribute, see Section 3.4.3.2.2, “contentattributetype”

3.4.3.2.4. contentattribute

Stores attributes of objects (each row for a value of an attribute).

Table 3.6. Columns of table `contentattribute`

Column	Description
id	unique row id (required for versioning)
contentid	contentid of the object this attribute belongs to
name	Name of the attribute, refers to the name in table contentattributetype
value_text	Column for storing attribute values (types 1, 2 see Section 3.4.3.2.2, “contentattributetype”)
value_clob	Column for storing attribute values (type 5 see Section 3.4.3.2.2, “contentattributetype”)
value_blob	Column for storing attribute values (type 6 see Section 3.4.3.2.2, “contentattributetype”)
value_int	Column for storing attribute values (type 3 see Section 3.4.3.2.2, “contentattributetype”)
value_long	Column for storing attribute values (type 8 see Section 3.4.3.2.2, “contentattributetype”)
value_double	Column for storing attribute values (type 9 see Section 3.4.3.2.2, “contentattributetype”)
value_date	Column for storing attribute values (type 10 see Section 3.4.3.2.2, “contentattributetype”)
Sortorder	Sortorder for multivalue attributes

3.4.3.2.5. contentmap_nodeversion

Versioning table for contentmap (required when datasource is versioning).

Table 3.7. Columns of table `contentmap_nodeversion`

Column	Description
... columns of content-map	All columns of contentmap
nodeversiontimestamp	Timestamp of the versioned record
nodeversion_user	User reference (who created the version)
nodeversionlatest	Flag for latest (current) versions of records
nodeversionremoved	Timestamp of the removal of the record
nodeversion_autoupdate	Flag for future record versions that still have to ba updated automatically (when nodeversiontimestamp is reached)

3.4.3.2.6. contentattribute_nodeversion

Versioning table for contentattribute (required when datasource is versioning).

Table 3.8. Columns of table `contentattribute_nodeversion`

Column	Description
... columns of content-map	All columns of contentattribute
nodeversiontimestamp	Timestamp of the versioned record
nodeversion_user	User reference (who created the version)

Column	Description
nodeversionlatest	Flag for latest (current) versions of records
nodeversionremoved	Timestamp of the removal of the record
nodeversion_autoupdate	Flag for future record versions that still have to be updated automatically (when nodeversiontimestamp is reached)

3.4.3.2.7. contentstatus

This table may contain meta information about the contentrepository. Currently it is used to store the timestamp (lastupdate) when Gentics Content.Node; last synced content into the contentrepository. This is needed for datasource caching.

Table 3.9. Columns of table `contentstatus`

Column	Description
name	Metaproproperty name
intvalue	Value of the metaproperty

3.4.3.3. Attribute Prefetching

To reduce the number of SQL Statements needed to filter objects from a content repository and access attributes of the objects, the Gentics .Node PortalConnector provides a feature called "Attribute Prefetching".

To successfully use attribute prefetching, the attributes which will be used lateron an shall be prefetched have to be provided.

- When using the Gentics .Node PortalConnector in Java Code:

```
// this is the datasource we are using
Datasource ds = ...;

// generate an expression to filter all pages (objects of type "10007")
// from the content repository
Expression expr =
        ExpressionParser.getInstance().parse("object.obj_type == 10007");

// now create a (reusable) datasource filter for our datasource
DatasourceFilter filter = ds.createDatasourceFilter(expr);

// now comes the magic, get the objects and have the names prefetched
Collection objects = ds.getResult(filter, new String[] {"name"});
```

- When using the DatasourceQueryImp in a template:

```
// set the filter rule to get all pages (objects of type "10007")
#set($portal.imps.query.rule = "object.obj_type == 10007")

// now comes the magic, have the names prefetched
#set($portal.imps.query.prefillAttributes = ["name"])

// now get the objects
#set($pages = $portal.imps.query.result)
```

- When using the datasourcetreecomponent in a view:

```
<datasourcetreecomponent>
   ...
   // set the root rule to get all pages (objects of type "10007")
   <rootrule>object.obj_type == 10007</rootrule>

   // now comes the magic, have the names prefilled
   <prefillattributes>name</prefillattributes>
</datasourcetreecomponent>
```

How does the attribute prefetching actually work? What about the caches? The steps to filter objects and prefetch attributes are

1. Get the objects without any attributes. When the filtering result is found in the cache, this will not do execute any SQL statements.

2. Calculate the maximum number of attributes that eventually need to be prefetched: `[number of objects] x [number of attributes]`

3. Check whether the maximum number of attributes to prefetch exceeds the configured *`prefetchAttribute.threshold`* (which defaults to `1000`). If that is the case, prefetch all attributes with a single (or at least few) SQL statements (algorithm finishes here).

4. When the total number of attributes is less than the *`prefetchAttribute.threshold`* , start fetching the attributes from the cache.

5. Calculate a threshold for cache misses (attributes not found in the cache) by taking the lesser of *`prefetchAttribute.cacheMissThreshold`* (defaults to `100`) and the percentage given in *`prefetchAttribute.cacheMissThresholdPerc`* (defaults to `20`) of the total number of attributes to prefetch.

   Example: 20 objects are filtered, and 2 attributes prefetched per object.

   Table 3.10.

Total number of attributes:	20 x 2 = 40
absolute cachemiss threshold	configured to 100
relative cachemiss threshold (20%)	40 x 20 / 100 = 8
effective cachemiss threshold	min(100, 8) = 8

6. When more than the calculated effective cachemiss threshold attributes were not found in the cache, prefetch ALL attributes (like above). Otherwise, the attributes not found in the cache will be fetched in single SQL statements, when accessed.

3.5. Imps

Information about their usage can be found in the Section 4.1, “Imps”

Imps are configured in the Section 3.8.4, “formatter-section”.

3.6. Languages

Gentics Portal.Node supports multilingualism. Languages are configured in the Section 3.8.5, “language-section”. Information on how the corresponding dictionaries have to be configured can be found at Section 3.8.5.3, “Dictionaries”

3.7. Templates

For detailed information on templates, see Section 4.11, “Templates”.

General portal-wide configuration of templates is described in Section 3.8.7, “template-section”.

3.8. Portal Configuration File

3.8.1. Introduction

The Portal Configuration File is located at `${com.gentics.portalnode.confpath}/default.portal.xml` and contains the basic configuration of the portal server.

3.8.2. general-section

3.8.2.1. Introduction

The `general-section` is used to define general parameters that are needed to run Gentics Portal.Node.

Example 3.2. Structure of the general-section

```
<general-section>
  <parameters>
    <parameter id="[paramname]">[paramvalue]</parameter>
        ...
  </parameters>
  <properties>
    <property id="[propertyname]">[propertyvalue]</property>
    ...
  </properties>
  <bookmarks>
    <bookmark pathinfo="[pathinfo]" destination="[destinationpath]">
      <parameters>
        <parameter id="[parameter]">[staticvalue]</parameter>
        <parameter id="[parameter]" mapped="[requestparameter]"/>
        ...
      </parameters>
    </bookmark>
    ...
  </bookmarks>
</general-section>
```

3.8.2.2. Portal server response settings

Table 3.11. Portal parameters for response settings

Name	Description	Default
portal.serversignature	True if the poral shall add a signature header	true

Name	Description	Default
	(X-Server) to every response, false if not server signature header shall be sent.	

3.8.2.3. Portal server activation

Table 3.12. Portal parameters for activation feature

Name	Description	Default
portal.activation.auto	True or false, whether the portal server shall automatically try to (re-)activate, when no valid license.key found (e.g. the hardware checksum changed). For SDK installations, this parameter is ignored and activations are always done interactively.	false
portal.activation.dialog	True or false, whether the activation shall be done with an activation dialog or must be done manually. For SDK installations, this parameter is ignored and activations are always done interactively.	false

Warning

For portal server installations, where the portal parameter *`portal.activation.dialog`* is set to `true` and the license becomes invalid (e.g. due to changes in the systems configuration), any user accessing the portal with a browser would see the activation dialog. To prevent this happening in production environments, it is STRONGLY recommended to turn this feature off, once the installation is successfully activated!

3.8.2.4. Updates

Table 3.13. Portal parameters for update checking

Name	Description	Default
portal.update.check	True or false, whether to check for updates on each startup. Writes upate status information to console when enabled.	true

3.8.2.5. Template Engine 2

Table 3.14. Portal parameters for the Template Engine 2

Name	Description	Default
portal.templateengine2.compatibilitymode	True or false, whether the Template Engine 2 is in compatibility mode or not. When the Template Engine 2 is in compatibility mode, all available imps will also be available directly under their respective id (e.g. $date for the DateFormatter). The compatibility mode is DEPRECATED and should not be used. Address the imps like $portal.imps.[impid] instead.	false

3.8.2.6. Portlet Template Finder Cache

The Portlet Template Finder Cache may provide you with a performance gain if a whole lot of component templates are used.

Table 3.15. Portlet Template Finder Cache

Name	Description	Default
portal.templatefinder.cache	Enable templatefinder caching, where best matching component templates are cached.	false

3.8.2.7. Portlet Cache

The Portlet Cache can be used to cache the render output of Portlets supporting Expiration Cache as defined by the Portlet Specification (JSR 168). See “Portlet Cache” (Section 4.4.5, p.169) for more information.

Table 3.16. Portal parameters for Portlet Cache

Name	Description	Default
portal.portlet.cache	Enables or Disables caching of Portlets.	false

3.8.2.8. Portal Template Configuration

The design, the used portlets and the adjustment of the used portlets of a portal are defined by the Gentics Portal.Node Portal Template. For details on the structure and content of the Portal Template see Section 4.11.1, “Portaltemplate” .

Note

When using Portal Pages, the portal template and with it all configuration parameters in this section are not used.

See “pages-section” (Section 3.8.9, p.60) and “Portal Pages Implementation” (Section 4.14, p.201) for details on Portal Pages.

Table 3.17. General portaltemplate parameters

Name	Description	Default
portal.template.templateengine2	Whether the portaltemplate itself shall be rendered using the Template Engine 2. See Section 4.11.4, “TemplateEngine2” for details on the Template Engine 2. Enabling this parameter is quite performance expensive.	FALSE
portal.template.hiderequesttime	Whether to disable and hide the HTML comment line at the of the template, containing the time taken by the portal to process this request, excluding network response time. A logger can be used instead.	FALSE

There are two possible ways to provide the portal template to the portal:

1. The portal template is loaded from a Gentics Content.Node; 3.6 content repository.

Table 3.18. Portal parameters for content repository-based portaltemplate

Name	Description	Default
portal.template.datasource	Id of the datasource where the template shall be loaded from. The datasource has to be configured in the Section 3.8.6, "datasource-section" and must be of type `contentrepository` .	NULL
portal.template.contentid	Contentid of the portaltemplate.	NULL

2. The portal template is provided as a file in the servers filesystem. For this, the absolute path of the file must be configured.

Table 3.19. Portal parameters for file-based portaltemplate

Name	Description	Default
portal.template.file	Absolute path to the portaltemplate file. This path may contain Section 3.2, "System properties" and it is recommended to locate the file relative to `com.gentics.portalnode.confpath` .	NULL

3.8.2.9. PLink Processor

PLinks are links to documents that are stored in the Gentics .Node ContentRepository and are displayed with a portlet of type GenticsContentModule. In order to have PLinks that are placed outside of GenticsContentModules correctly working, the id of a GenticsContentModule has to be configured here as well.

<plink>s have two different types of locations:

1. Inside a content that is shown in a GenticsContentModule. In this case, the default behaviour of the link is to trigger the `onSelect` event of this portlet. This event can be caught in a <reaction> and the business logic implemented accordingly. Note that this behaviour can be modified by setting the portal parameter `portal.plinkprocessor.forceall` (see below).

2. Outside of GenticsContentModules (in content rendered by other portlets or even in the portal template itself). These <plink>s will trigger the `onSelect` event of the configured `plinkprocessor` .

For details on the portlet see Section 5.4, "GenticsContentModule". Details on configuring and placing portlets and implementing reactions can be found in Section 4.11.1.3, "PNodes"

Table 3.20. Portal parameters for plink processor

Name	Description	Default
portal.plinkprocessor	Id of the "GenticsContentModule" (Section 5.4, p.?) that will process all <plink>s that are not placed inside content rendered in GenticsContentModules. When the parameter `portal.plinkprocessor.forceall` is set to `true`, the plinkprocessor will process all <plink>s.	NULL
portal.plinkprocessor.forceall	If set to `true` all >plink>s will be rendered using the provided `portal.plinkprocessor` as their	false

Name	Description	Default
	target. Enable to ensure backwards compatibility to Gentics Portal.Node versions prior to 3.2.	

3.8.2.10. Fileupload Configuration

Fileupload parameters are mainly used to limit the default maximum fileupload size.

Table 3.21. Portal parameters for fileuploads

Name	Description	Default
portal.upload.maxFilesize	Maximum allowed fileupload size in bytes.	1048576 (1mb)
portal.upload.sizeThreshold	Uploads larger than the given bytesize will temporarily be stored in the filesystem (repositoryPath. Smaller uploads will be kept in memory.	4096 (4kb)
portal.upload.repositoryPath	The path where upload files will temporarily be stored when their size exceed sizeThreshold.	${java.io.tmpdir}

3.8.2.11. ViewPlugin Configuration

ViewPlugin configuration parameters are used to modify the general behaviour of the viewplugin.

Table 3.22. Portal parameters for the viewplugin.

Name	Description	Default
portal.viewplugin.checkviewchanges	Whether changes in the view files shall be checked periodically or not. This helps in development environments since changes in view files are automatically recognized and the files are reloaded. It is strongly recommended that in production environments this is set to `false` for performance reasons!	true
portal.viewplugin.doubleclickprotection	This parameter is the portal-wide setting of the double-click protection of views. When the double-click protection is `true` for a specific view, the portal will check whether an action request directed to the view contains the renderkey which was generated the last time the view was rendered. This prevents unwanted effects when users use the browsers back button and submit a form from a portalpage that was fetched from the browsers history. The portal-wide setting may be overwritten per ViewPortlet or event per view (see “ViewPortlet” (Section 5.3, p.209) and “Views” (Section 4.2.2.6, p.?) for details).	true
portal.viewplugin.disableviewreferences	With this parameter, the support for the deprecated view references can be disabled. Doing so will significantly reduce the memory consumption per user session. It is therefore strongly recommended, that view references are disabled by setting this parameter to `true` , unless they are used in the implementation.	false

3.8.2.12. Datasources Configuration

Datasources configuration parameters are used to modify the general behaviour of datasources.

Table 3.23. Portal parameters for datasources

Name	Description	Default
portal.datasources.autorepair	Flag to switch off the auto-repair functionality of Gentics Portal.Node. When auto-repair is turned on, all datasources of type `contentrepository` are checked and the server tries to repair the datasources by adding missing columns. Modifications in the data structure may fail due to database permission settings or may be long running operations (for example when the table is very large). Therefore an administrator may opt to turn the auto-repair off by setting this parameter to `false` and make eventually necessary modifications manually during an update process. This global setting can be overwritten for specific datasources. See “Datasources of type `contentrepository`” (Section 3.8.6.3.1, p.53) for details.	true

3.8.2.13. ExpressionParser Configuration

These parameters can be used to use and configure the new “ExpressionParser” (Section 4.10, p.182) in favor of the old RuleParser.

Table 3.24. Portal parameters for the ExpressionParser

Name	Description	Default
portal.expressionparser	When this parameter is set to `false` , the ExpressionParser is not used and Gentics Portal.Node runs in compatibility mode (all rules, filters and reactions are processed in a way that is compatible with older versions of Gentics Portal.Node) . When set to `true` (default value) , the ExpressionParser is used. Note that some rules, filters or assignment commands might be incompatible with the new ExpressionParser (or might be interpreted in a different way). To find possible problems, it is recommended to use the `test` mode and the log-level for the logger `com.gentics.lib.expressionparser` to `DEBUG` while upgrading an existing Gentics Portal.Node implementation to the ExpressionParser. With this setting, the ExpressionParser will be used, but generate a lot more log messages for possibly incompatible expressions. See Section 3.9, “Logging” for details on setting the log level and Section 4.10, “ExpressionParser” for details on the ExpressionParser.	true
portal.expressionparser.treatEmptyStringAsNull	When *`portal.expressionparser`* is set to	false

Name	Description	Default
	`true` or `test` (the ExpressionParser is used, default), this parameter modifies the behaviour of comparisons `[object.attribute] == ""` . When *`portal.expressionparser.treatEmptyStringAsNull`* is set to `false` , the comparison is only true for empty strings (not for null values, meaning that the attribute is not set). When set to `true` , the comparison is also true for null values.	

3.8.2.14. Dictionaries Configuration

Parameters for configuration of general dictionary handling. See “Dictionaries” (Section 3.8.5.3, p.47) for details.

Table 3.25. Portal parameters for Dictionaries

Name	Description	Default
portal.dictionaries.modificationCheckInterval	Modification check interval for all dictionary files in seconds. When set to -1, no modification check is done and dictionary changes will become available only on restart of Gentics Portal.Node (recommended for production systems). A setting of 0 will check on every access to a dictionary.	-1 (no modification check)

3.8.2.15. Portal pages Configuration

Parameter for switching on/off the usage of portal pages. See “pages-section” (Section 3.8.9, p.60) and “Portal Pages Implementation” (Section 4.14, p.201) for details on portal pages.

When portal pages are used, the “Portal Template Configuration” (Section 3.8.2.8, p.31) is completely replaced by the “Portlet Entities Configuration” (Section 3.13, p.78) .

Table 3.26. Portal parameters for Portal pages

Name	Description	Default
portal.pages	With this parameter the multiple portal pages can be activated. When set to `false` (default), the portal template will be rendered.	false
portal.portlet.entities.file	Path to the file configuring the portlet entities. This path must be set, when portal pages are used. See “Portlet Entities Configuration” (Section 3.13, p.78) for details on portlet entities.	NULL

3.8.2.16. URL Mapping Configuration (Beautiful URLs)

Parameter for switching on/off the usage of “urlmapping-section” (Section 3.8.10, p.63)

Table 3.27. Portal parameters for Portal pages

Name	Description	Default
portal.urlmappings	Enables usage of the urlmapping section.	false

3.8.2.17. Portlet Reloading

Global configuration for portlet reloading (see “Portlet Reloading” (Section 4.6.3.3, p.171)). These global settings can be overridden by PNode parameters (see “Parameters of all Gentics Portlets” (Section 5.2.3, p.208)).

Table 3.28. Portal parameters for Portlet Reloading

Name	Description	Default
portal.portletreload	Enables portlet reloading for all portlets.	false
portal.portletreload.allclasses	Replace all elements, regardless of CSS class (see *portal.portletreload.replaceclasses*).	true
portal.portletreload.replaceclasses	Enables portlet reloading only for elements (links, forms) which have at least one of the specified CSS classes. List all CSS classes as a comma-separated list.	NULL
portal.portletreload.replaceform	Enables portlet reloading for forms (replace the default submit action).	true
portal.portletreload.replacelink	Enables portlet reloading for links (replace the default onclick action).	true

3.8.2.18. Namespace Compliance

Parameter for switching on/off the JSR 168 compliant generation of namespaces.

For historic reasons, the namespaces generated for render or resource responses contained special characters like "." (dots), which is not compliant to the portlet specification. With this portal parameter, this behaviour can be changed.

Table 3.29. Portal parameters for Namespace compliance

Name	Description	Default
portal.namespace.compliance	Enables generation of JSR 168 compliant namespaces.	false

3.8.2.19. Login Behaviour

Parameter to influence the login behaviour: When switched off, logging in will create a new portal session, which is the original behaviour. When switched on, logging in will keep the portal session together with all current settings. Logging out will always create a new portal session.

The new login behaviour will also reduce the memory consumption per session.

Table 3.30. Portal parameters for Login Behaviour

Name	Description	Default
portal.keepsessiononlogin	Enables the behaviour to keep the portal session when logging in.	false

3.8.2.20. Parameters for configuring the markup aggregation

With the following parameters, the aggregation of markup provided by portlets can be configured.

Table 3.31. Portal parameters for markup aggregation

Name	Description	Default
portal.headeratend	When this parameter is set to `true` , any markup provided by portlets that should be added to the head section sent to the client will be appended to the end of the head section. When set to `false` (which is the default), that markup will be prepended to any existing `<head>` content.	false

3.8.2.21. Portal Properties

Portal Properties are used to store portal-wide settings for use in views or velocity templates. These settings can be accessed and set via portal.properties.[propertyname], where propertyname is the id of the property you defined. So if you would like to define your company name for use in the portal it should look as stated below.

Example 3.3. Portal Property configuration

```
<general-section>
  ...
  <properties>
    <property id="companyname">MyCompany Inc.</property>
  </properties>
</general-section>
```

Now, if you'd like to use the property you just have defined in an velocity template, you can simply use `$portal.properties.companyname`. Please keep in mind that the portal has to be restarted for the new properties set in the portal configuration file to take effect. Portal Properties can also be used in views as the following example depicts.

Example 3.4. Usage of Portal Property inside a view

```
...
<buttoncomponent>
  <label>Click Me!</label>
  <actions>
    <action class="EchoAction">
```

```
      <parameters>
        <parameter id="CompanyName" mapped="portal.properties.companyname"/>
      </parameters>
    </action>
  </actions>
</buttoncomponent>
...
```

This example creates a buttoncomponent inside an existing view. On click of the button the company name ist echoed into the logfile.

3.8.2.22. Bookmarkable URLs

The section for bookmarkable URLs defines the pathinfo and parameters for bookmarkable URLs and how they are mapped to normal portal request URLs.

Bookmarkable URLs are constructed like `http://[hostname]:[portname]/Portal.Node/go/[path-info]?[query-string]` , where *`path-info`* identifies the URL definition in this section and *`query-string`* may contain request parameters that can be mapped to the destination URL.

The destination URL is `[destinationpath]?[query-string]` , where *`destinationpath`* is defined for the bookmarkable URL and *`query-string`* is constructed from the defined *`parameters`* . Every parameter may either have a static value, or may be *`mapped`* from an incoming request parameter.

Example 3.5. Bookmarkable URL

```
<bookmarks>
  <bookmark pathinfo="/minimize" destination="/Portal.Node/portal">
    <parameters>
      <parameter id="gentics.rm" mapped="portlet"/>
      <parameter id="gentics.pb" mapped="portlet"/>
      <parameter id="gentics.ws">minimized</parameter>
    </parameters>
  </bookmark>
</bookmarks>
```

Will forward the bookmarkable URL `http://hostname/Portal.Node/go/minimize?portlet=AdministrationPortlet` to `http://hostname/Portal.Node/portal?gentics.rm=AdministrationPortlet&gentics.pb=AdministrationPortlet&gentics.ws=minimized` (which minimizes the portlet `AdministrationPortlet` in the pbox `AdministrationPortlet`).

3.8.3. authentication-section

3.8.3.1. Introduction

This section contains the configuration of the used Authentication Managers and the anonymous user profile.

Example 3.6. Structure of the authentication-section

```
<authentication-section>
```

```
  <authentication-descriptor default="...">
    <authentication id="...">
      <Description>...</Description>
      <class-name>...</class-name>
      <parameters>
        <parameter id="...">...</parameter>
      </parameters>
      <mapping>
        <map from="..." to="..."/>
        ...
      </mapping>
    </authentication>
    <authentication id="...">
    </authentication>
    ...
  </authentication-descriptor>
  <anonymous-user>
    <domain name="..." authentication="...">
      <anonymous id="...">
        <anonymous-parameter id="...">...</anoymous-parameter>
      </anonymous>
    </domain>
  </anonymous-user>
</authentication-section>
```

3.8.3.2. Configuration Details

Table 3.32. Configuration of Authentication Managers

Name	Type	Description
authentication-descriptor	Structure	Base node for the definition of authentication managers
authentication-descriptor.default	String	Id of the default (primary) authentication manager that is used for access management.
authentication	Structure	Node that defines a single authentication manager.
authentication.id	String	Id of this authentication manager. must be unique within all authentication manager ids.
Description	String	Description of this authentication manager.
class-name	String	Full qualified class-name of the java class that implements the authentication manager. The class must implement the class `com.gentics.portalnode.auth.AuthenticationSystem`
parameters	Structure	Definition of the parameters for this authentication manager.
parameter	Node	Definition of a single parameter for this authentication manager, the value is given as node value.
parameter.id	String	Id of the parameter to set.
mapping	Structure	Definition of parameter mappings from the user repository to the portal user parameters.
map	Node	Definition of a single parameter mapping.
map.from	String	Original parameter name in the user repository.
map.to	String	Name of the parameter for the portal user.
anonymous-user	Structure	Definition for the user profile for the anonymous user.
domain	Structure	User profile for an anonymous user for a given domain pattern. The domain pattern must match the hostname under which the user reaches the portal.

Name	Type	Description
domain.name	String	Pattern of the hostname to match. May contain * as wildcard for arbitrary characters.
domain.authentication	String	Id of the authentication manager that is the default for this domain.
anonymous	Structure	Definition of anonymous user parameter for a given authentication manager.
anonymous.id	String	Id of the authentication manager.
anonymous-parameter	Node	Definition of a single anonymous parameter. The value is given as node value.
anonymous-parameter.id	String	Id of the anonymous parameter.

Table 3.33. Parameters for all AuthenticationManagers

Name	Default	Description
secondaryrule	null	Rule for checking, whether an authentication manager shall be used as secondary authentication manager (if it is not the primary auth manager). The rule will allow access to the current request (as request.*) and the user, with all data from the primary and the other secondary authentication managers, that already added their data (as user.*).

3.8.3.3. PNAuthenticationManager

This authentication manager is implemented by the class `com.gentics.portalnode.auth.manager.PNAuthenticationManager` and provides authentication based on a datasource of type `contentrepository`. For details on configuration of datasources, see Section 3.8.6, “datasource-section” .

Note

Please note that request parameters have to be prefixed with "p." when a user should be logged in using an external HTTP request. e.g. "_PNlogin" will become "p._PNlogin" since Gentics Portal.Node will automatically prefix userrequestparameter, passwordrequestparameter and logoutrequestparameter internally.

Table 3.34. Parameters for the PNAuthenticationManager

Name	Default	Description
userrequestparameter	_PNlogin	Name of the request parameter that holds the login name of the user requesting authentication.
passwordrequestparameter	_PNpassword	Name of the request parameter that holds the (unencrypted) password.
logoutrequestparameter	_PNlogouturl	Name of the request parameter that identifies a logout request.
logoutrequestvalue	true	Value of the request parameter that identifies a logout request.
datasourceid	{required}	Id of the datasource that holds the user profiles. Must be a datasource of type `contentrepository` .
objecttype	{required for secondary authentica-	Objecttype of the user objects in the datasource.

Name	Default	Description
	tion managers}	
userattribute	login	User attribute that holds the login name and therefore identifies the user profile.
passwordencryption	MD5	The used encrytion method for the password. May be one of `plaintext` (no encryption), `MD5` or `SHA-1`.
loginrule	{required for primary authentication managers}	The filterrule that will find the user profile matching the given login data, which are provided as *`data.login`* and *`data.password`* (encrypted). It is strongly recommended that *`data.login`* is matched against the object attribute configured in the parameter *`userattribute`*, but the rule may as well contain other relevant filter criteria.

3.8.3.3.1. PNAuthenticationManager as primary authentication system

When this authentication manager is used as primary authentication system, it will authenticate requests that contain login data as values of the *`userrequestparameter`* and *`passwordrequestparameter`*. The configured *`loginrule`* is used as filter for a search in the datasource and if it returns objects, the user is supposed to be authenticated. The returned object should be unique and should be the user profile of the authenticated user. When no objects are returned in the search, the authentication fails and the user remains anonymous.

3.8.3.3.2. PNAuthenticationManager as secondary authentication system

When this authentication manager is used as secondary authentication system, it will not use any *`loginrule`* but will just store any additional set user attributes in the datasource. The datasource object holding the additional user profile is defined by

- object.obj_type == *`objecttype`*
- object.*`userattribute`* == [userid from the primary authentication system]

3.8.3.4. LDAPAuthenticationManager

This authentication manager is implemented by the class `com.gentics.portalnode.auth.manager.LDAPAuthenticationManager` and provides authentication based on a datasource of type `ldap`. For details on configuration of datasources, see Section 3.8.6, “datasource-section” .

Table 3.35. Parameters for the LDAPAuthenticationManager

Name	Default	Description
ldapDatasourceName	(required)	Id of the datasource that holds the user profiles. Must be a datasource of type `ldap` .
ldapVersion	(optional)	LDAP Version the server is using. Defaults to 3 if no value provided.
ldapPort	(optional)	Port of the ldap server. Defaults to 389 if no value provided.
ldapHost	(required)	Hostname (or ip address) of the ldap server.
userDNAttribute	(required)	the user's attribute which forms the complete DN when concatenated with *`baseDN`*, eg: mail or cn. Do NOT add equals or comma here!
filterRule	(optional)	allows you to specify a custom filter rule. The filter rule used by default is: `object.USERDNATTRIBUTE == login.name`. USERDNATTRIBUTE is the userDNAttribute you specified, while *`login.name`* is the login name the user

Name	Default	Description
		entered in the authentication form. When specifying a custom filter rule you can use the variables *login.name* and *login.pass* which will be resolved to the user's login name or password respectively. A custom rule could look as follows: `object.uid == login.name && object.class = "Person"`. Note that the deprecated placeholders $loginName and $loginPass still can be used, but it is recommended to use login.name and login.pass instead.

3.8.3.4.1. Mode of Operation

The LDAPAuthenticationManager determines a user's DN by executing an LDAP search via the LDAP datasource specified, filtered by *filterRule*. If an object is returned the LDAP Authentication Manager will try to bind with the password provided by the login form. On successful bind the user is authenticated and provided with his properties which where retrieved during LDAP search.

Another vital configuration part is the mapping section, since only attributes which are configured in this section will be available as user properties when the user is logged in. So eg. if you forget to map a user's mail attribute you won't be able to read his mail address.

3.8.3.4.2. Example

Here is an example configuration which includes authentication and datasource section from default.portal.xml.

```
...
<authentication-section>
  <authentication-descriptor default="LDAP">
    <authentication id="LDAP">
      <Description>LDAP Authentication Module</Description>
      <class-name>com.gentics.portalnode.auth.manager.LDAPAuthenticationManager
        </class-name>
      <parameters>
        <parameter id="ldapDatasourceName">ldap</parameter>
        <parameter id="userDNAttribute">uid</parameter>
        <parameter id="userPasswordAttribute">userPassword</parameter>
        <parameter id="ldapPort">389</parameter>
        <parameter id="ldapHost">localhost</parameter>
        <parameter id="filterRule"><![CDATA[object.uid == "$loginName"]]&lt;
          </parameter>
      </parameters>
      <mapping>
        <map from="cn" to="cn" />
        <map from="givenName" to="givenName" />
        <map from="mail" to="mail" />
      </mapping>
    </authentication>
  </authentication-descriptor>
  <anonymous-user>
    <domain name="*" authentication="LDAP">
      <anonymous id="LDAP">
        <anonymous-parameter id="userid">max.mustermann@gentics.com
          </anonymous-parameter>
        <anonymous-parameter id="firstname">Max</anonymous-parameter>
        <anonymous-parameter id="lastname">Mustermann</anonymous-parameter>
        <anonymous-parameter id="cn">Max Mustermann</anonymous-parameter>
      </anonymous>
    </domain>
  </anonymous-user>
</authentication-section>
<datasource-section>
  <datasource-handles>
    <datasource-handle typeid="ldap" id="ldap_handle">
```

```
      <parameter name="host">localhost</parameter>
      <parameter name="port">389</parameter>
      <parameter name="basedn">dc=abc,o=def</parameter>
      <parameter name="binddn"/>
      <parameter name="password"/>
    </datasource-handle>
  </datasource-handles>
  <datasources>
    <datasource id="ldap" typeid="ldap">
      <parameter name="binddn"/>
      <handles>
        <handle>ldap_handle</handle>
      </handles>
    </datasource>
  </datasources>
</datasource-section>
...
```

3.8.3.4.3. Limitations

Currently the LDAP Authentication Manager cannot handle multivalue attributes.

3.8.3.5. SiteMinderAuthenticationManager

This authentication manager is implemented by the class `com.gentics.portalnode.auth.manager.SiteMinderAuthenticationManager` and provides authentication based on CA eTrust® SiteMinder® module for the Apache WebServer. It currently supports singon and read operations.

Table 3.36. Parameters for the SiteMinderAuthenticationManager

Name	Default	Description

collections	NULL	List of SiteMinder attributes that contain collections of attributes. Separated by ';'
		
		The values of the attributes listed here should follow this structure: `attribute1=key1:value1|key2:value2^attribute2=value3^...` . This would set user parameters as follows:
		
		• *attribute1* = Map
		  • *key1* = `value1`
		  • *key2* = `value2`
		• *attribute2=* = `value3`

authentication_key	SM_UNIVERSALID	Specifies the header attribute which will be checked to verify if a user is logged in.
decodeUTF8	false	This parameter can be set to `true` in case of incorrectly encoded Siteminder headers. Normally, request headers should be encoded according to the request encoding, but in special situations Siteminder does not respect the request encoding, but adds additional headers in ISO-8859-1 encoding. When the request encoding is set to 'UTF-8' (which is normally true

Name	Default	Description
		for usual server configurations), setting this property will let the SiteminderAuthenticationManager decode the headers, which were incorrectly interpreted by the application server as being 'UTF-8' into correct 'UTF-8'.

3.8.3.6. RemoteUserAuthenticationManager

This authentication manager is implemented by the class `com.gentics.portalnode.auth.manager.RemoteUserAuthenticationManager` and provides authentication based on the RemoteUser information provided by the HttpServletRequest instance (return value of the method `javax.servlet.http.HttpServletRequest.getRemoteUser()`).

This allows coupling the portal authentication to authentication mechanisms provided by the underlying application server, or by independent implementations e.g. in ServletFilters.

Since passing additional attributes of the authenticated individual to the Web Application is not in the scope of the Servlet Specification, this authentication manager provides the possibility to pass user attributes to the portal by storing them as a `java.util.Map` either in the session or as request attribute (depending on the setting of the parameter ***`attributescope`*** described below) under the key `com.gentics.portalnode.remoteuserdata` .

Table 3.37. Parameters for the RemoteUserAuthenticationManager

Name	Default	Description
attributescope	session	When this parameter is set to `session` , the authentication manager expects the Map of user attributes to be stored in the session, when set to `request` it expectes them to be stored as request attribute.

3.8.3.7. GCNAuthenticationManager

This authentication manager is implemented by the class `com.gentics.portalnode.auth.manager.GCNAuthenticationManager` and provides authentication by login to a Gentics Content.Node 5 backend system.

When the GCNAuthenticationManager is configured as primary Authentication Manager and a User tries to login (e.g. with the GenticsLoginModule), the Portal Server will POST the login credentials to the configured backendb url. This means that the given URL must be reachable by the Portal Server, NOT the client.

The GCNAuthenticationManager can also be configured as secondary Authentication Manager behind another one (e.g. SiteMinderAuthenticationManager). In that case, either a mapping for username and password or for the headers for the authentication request (or both) need to be configured.

Table 3.38. Parameters for the GCNAuthenticationManager

Name	Default	Description
url	NULL	URL to the backend system which will be used to authenticate the portal user, e.g. ***`http://localhost/.Node/`*** . This parameter is mandatory.
header	NULL	Configuration of the request headers which must be added to the authentication request, when the backend is configured to

Name	Default	Description
		authenticate by request headers (e.g. for use with SiteMinder). The headers need to be configured as comma separated list of key/value pairs, which are separated by the character '='. Keys are the names of the request headers, values are expressions, which resolve to the header values, based on the portal user attributes which might be provided by other authentication managers. Example: `UID=sm.UID,GIVENNAME=sm.GIVENNAME,SN=sm.SN,MAIL=sm.MAIL,CN_GRUPPE=sm.CN_GRUPPE` In this example, the header values are mapped from the primary Authentication Manager (with id `sm`).
username	NULL	Expression which resolves to the value to be used as username when authenticating as secondary Authentication Manager. The expression is evaluated based on the current user object. (See description of *`header`* parameter).
password	NULL	Expression which resolves to the value to be used as password when authenticating as secondary Authentication Manager. The expression is evaluated based on the current user object. (See description of *`header`* parameter).
cookie	NULL	Comma separated list of cookie names, that will be forwarded to the authentication request.
timeout	5000	Timeout in ms before getting the connection fails.
sockettimeout	5000	Timeout in ms before connecting or reading from the socket fails.
connectionretry	3	Number of retries for establishing a connection to the authentication URL.
refreshinterval	600	Interval in seconds for keeping the backend session alive. The default is 600s (10 minutes).
backendLoginRetryInterval	0	This parameter specifies the number of seconds to wait to retry a failed login request to the GCN backend. The value 0 will retry on every request. Logins to the GCN backend are unsuccessful if the user authenticated with the primary authentication system doesn't have a GCN account (e.g. read-only privileges for the content in the portal, but no editing privileges in the GCN backend). Another reason may be that the GCN backend may be temporarily unavailable. Retrying unsuccessful logins may cause performance problems if there are many users that can be authenticated with a primary authentication system but can't be authenticated with the GCNAuthentication manager as a secondary authentication system - each request by the user (browser) would cause an additional internal authentication request.

The following table lists the properties which are readable for authenticated users.

Table 3.39. User Properties from GCNAuthenticationManager

Name	Description
firstname	First name of the user
lastname	Last name of the user

Name	Description
fullname	firstname lastname
email	eMail Address of the user
description	Description of the user
login	Login name of the user
userid	Alias for login
groups	List of group ids the user is member of.
sid	Gentics Content.Node 5 session id
sessionSecret	Gentics Content.Node 5 session secret
sessionToken	Gentics Content.Node 5 session token

3.8.4. formatter-section

3.8.4.1. Introduction

The formatter section contains the configuration of the Gentics Portal.Node Imps.

Imps are small helper classes that provide useful functionality, mostly used in templates - e.g. to output formatted data. In prior versions of Gentics Portal.Node they used to be called Formatters.

Example 3.7. Syntax of the formatter-section

```
<formatter-section>
  <formatter id="...">
    <class-name>...</class-name>
    <parameters>
      <parameter id="...">...</parameter>
        ...
    </parameters>
  </formatter>
  ...
</formatter-section>
```

3.8.4.2. Configuration

Table 3.40. Configuration of imps

Name	Type	Description
formatter	Structure	Definition of an imp.
formatter.id	String	Id of the imp. Must be unique within all imp-ids.
class-name	String	Full qualified java class-name of the class implementing the imp's features.
parameters	Node	Configuration parameters for the imp.
parameter	Node	Value of one configuration parameter.
parameter.id	String	Id of the configuration parameter.

3.8.5. language-section

3.8.5.1. Introduction

The available languages in the portal. This is the basic configuration needed for internationalization. The currently selected language of the user can be read and set with the portal property path *portal.language.id*. See Section 4.8, “Portal Property Paths” for Details on property paths.

Example 3.8. Structure of the language-section

```
<language-section>
  <languages default="...">
    <language id="..." locale="..."/>
    ...
  </languages>
</language-section>
```

3.8.5.2. Configuration

Table 3.41. Configuration of Languages

Name	Type	Description
languages	Structure	Base node for the language definitions.
languages.default	String	Id of the default language.
language	Node	A single language definition.
language.id	String	Id of the language definition.
language.locale	String	Locale of the language definition. The locale may consist of up to three parts, separated by '_' (underscore): 1. lowercase two-letter ISO-639 code of the language 2. uppercase two-letter ISO-3166 code of the country 3. vendor and browser specific code for the variant Examples: use de for German, de_AT for Austria, de_DE for Germany, en for English, en_GB for UK, en_US for US, ... See http://java.sun.com/j2se/1.4.2/docs/api/java/util/Locale.html for details on Locales.

3.8.5.3. Dictionaries

For implementation details, see Section 4.12.2, “Language Handling”.

3.8.5.3.1. Syntax

The basic Syntax of dictionary files is *key*=value. Note that if the key contains spaces, they have to be escaped with \. Also if the value contains newlines, they have to be escaped in the same way.

Example 3.9. Example dictionary

```
welcome\ text=Welcome,\
this is Gentics Portal.Node brought to you by $name.
custom.diclaimer2=All my rights reserved.
```

The values may contain variables in the syntax `$[a-zA-Z0-9]+`, which might be filled with dynamic data.

3.8.5.3.2. Package Dictionaries

The package dictionaries are located at `${com.gentics.portalnode.home}/META-INF/dictionaries/portal.[languageid].property` , are maintened by Gentics and should not be changed. Changes will be lost after the next product update. Changes should be done by creating a custom dictionary file in your configuration path.

3.8.5.3.3. Custom Dictionaries

Custom dictionaries are located at `${com.gentics.portalnode.confpath}/dictionaries/portal.[languageid].property` .

If you want to customize the package dictionary, create an empty file in this directory and add the items you want to customize from the Package dictionary by copy pasting the corresponding lines. It's not suggested to copy the whole file, since future changes in the package dictionary of uncustomized items will not be visible any more.

If you want to add new items, it's suggested to do this in a custom section, with a custom prefix, to prevent unintentional overwriting of already existing words.

3.8.5.3.4. Portlet application Dictionaries

Portletapplication dictionaries are located at `/WEB-INF/dictionaries/portal.[languageid].property` in the respective portlet application's directory.

Portlet application dictionaries are the means to translate portlet application specific words and package the dictionaries within the portlet application. These dictionaries have higher priority than the package or custom portal dictionaries.

3.8.6. datasource-section

3.8.6.1. Introduction

The datasource-section is used to define different datasources that are available to the portal. The datasources are used by the portal modules.

Every datasource has an id, a type and can have one or more handles.

This section is split up into a datasource section and a datasource handle section.

Example 3.10. Structure of the datasource-section

```
<datasource-section>

   <datasource-handles>
        <!-- the datasource-handles -->
   </datasource-handles>
```

```
   <datasources>
        <!-- the datasources  -->
   </datasources>

</datasource-section>
```

3.8.6.2. Defining datasource-handles

Each datasource handle is defined by an unique id, and typeid and can have several parameters.

3.8.6.2.1. Handle type `sql`

This handle type defines datasource handles that provide access to sql-databases. The database connection can be created in two ways: Directly by Portal.Node using `jdbc` database drivers, or created by the underlying application server and provided as `jndi` ressources.

Warning

When using a handle of type "sql" together with Jakarta Tomcat 5.0 *be sure to place the jdbc driver package at {catalina.base}/common/lib/*. Otherwise you will not be able to connect to the database.

Table 3.42. Parameters for jdbc connections

Name	Description	Required	Default value
type	must be set to `jdbc` for poolable connections created by Portal.Node. This is the preferred method of creating database connections.	x	must be set to jdbc
driverClass	Full qualified class name of the driver class to use for the connection. The jdbc driver package must be located somewhere in the classpath of Portal.Node. Important note for Jakarta Tomcat 5.0: The jdbc driver package must be located in the directory {catalina.base}/common/lib/.	x	
url	Connection url to use for connecting to the database. The url may contain system properties	x	
username	Username for connecting to the database. Note: when at least one ***`driver.`*** parameter is set, the username must be given as ***`driver.user`*** and this parameter will be ignored.		
passwd	Password for connecting to the database. Note: when at least one ***`driver.`*** parameter is set, the password must be given as ***`driver.password`*** and this parameter will be ignored.		
driver.*	Any parameter that is prefixed with ***`driver.`*** will be passed as custom parameter to the JDBC driver (without the prefix ***`driver.`***).		
pooling.maxActive	Maximum number of active connections in		8

Name	Description	Required	Default value
	the connection pool.		
pooling.minIdle	Minimum number of idle connections in the connection pool (initial size of the pool).		0
pooling.maxIdle	Maximum number of idle connections in the connection pool. If more than maxIdle unused connections exist in the pool, some are closed.		8
pooling.whenExhaustedAction	Action to be taken when the pool is exhausted. Must be one of • `block` the client will block and wait for a free connection. • `fail` the request for a connection will fail immediately. • `grow` the pool will grow (a new connection is created).		block
pooling.maxWait	The maximum amount of time (in millis) the client will wait for a free commection when the pool is exhausted and whenExhaustedAction is block. -1 will wait forever.		-1
pooling.testOnBorrow	Whether a connection should be checked before given to the client.		false
pooling.testOnReturn	Whether a connection should be checked when returned to the pool.		false
pooling.testWhileIdle	Whether idle connections in the pool should be checked regularly by the idle object evictor thread.		false
pooling.timeBetweenEvictionRunsMillis	The number of milliseconds to sleep between runs of the idle object evictor thread. When non-positive, no idle object evictor thread will be run..		-1
pooling.numTestsPerEvictionRun	The number of objects to examine during each run of the idle object evictor thread (if any).		3
pooling.minEvictableIdleTimeMillis	The minimum amount of time in milliseconds an object may sit idle in the pool before it is eligable for eviction by the idle object evictor (if any).		3600000 (30 minutes)
pooling.validationQuery	The query to be used to validate connections. Should at least return one row.		null
shutDownCommand	The command (query) to be sent to the database when the handle is closed during the shutdown of Gentics Portal.Node . This has to be set dependent on the database type.		null
cachedbmetadata	With this option, the handle can be configured to cache database metadata, in cases where the database structure does not change in runtime, and multiple calls to the method `getMetaData()` of the JDBC connector leads to performance problems.		false
dbschema	This specifies the dbschema name to be		null

Name	Description	Required	Default value
	used when fetching the database metadata. If left empty, no schema name will be used to get the database metadata (default).		

Note

All pooling parameters may also be used without the prefix ***pooling.*** , but the new syntax with prefix is encouraged.

Warning

When at least one ***driver.*** parameter is set, the parameters ***username*** and ***passwd*** will be ignored and the login data must be given as ***driver.user*** and ***driver.password*** .

Example 3.11. Example of a jdbc datasource-handle

```
<datasource-handle typeid="sql" id="sample">
  <parameter name="type">jdbc</parameter>
  <parameter name="url">
    jdbc:hsqldb:${com.gentics.portalnode.home}/WEB-INF/config/db/sample
  </parameter>
  <parameter name="driverClass">org.hsqldb.jdbcDriver</parameter>
  <parameter name="driver.user">sa</parameter>
  <parameter name="shutDownCommand">SHUTDOWN</parameter>
</datasource-handle>
```

Table 3.43. Parameters for jndi connections

Name	Description	Required	Default value
type	must be set to `jndi` for jdbc connections provided by the underlying J2EE application server	x	must be set to jndi
name	Name under which the DataSource is bound to the JNDI context (excluding the prefix).	x	
context	Context under which the DataSource is bound.		java:comp/env
shut-Down-Command	The command (query) to be sent to the database when the handle is closed during the shutdown of Gentics Portal.Node . This has to be set dependent on the database type.		null

Example 3.12. Example of a jndi datasource-handle

```
<datasource-handle typeid="sql" id="sample">
        <parameter name="type">jndi</parameter>
        <parameter name="name">jdbc/sample</parameter>
</datasource-handle>
```

3.8.6.2.2. Handle type `ldap`

This Handle type provides access to an LDAP server.

Table 3.44. Parameters for ldap Handles

Name	Description	Required	Default value
host	Host of the LDAP server.		localhost
port	Port number of the LDAP server.		389
binddn	DN to use for binding to the LDAP server for authentication.	x	
password	Password to use for binding to the LDAP server for authentication.	x	
scope	Scope for searches in the LDAP. Must be one of base|one|sub.		one
poolsize	Size of the connection pool.		20
timeout	Timeout in ms before the access fails.		5000
sockettimeout	Timeout in ms before connecting or reading from the socket fails.		5000

3.8.6.3. Defining datasources

A datasource is defined by an ID, a datasource type and one or more datasource handles. Readonly datasources support the usage of multiple datasource handles. Additionally for some datasource types, parameters may be defined.

Note

The datasource ID may not be set to `false`.

Example 3.13. Datasource examples

```
<datasources>
  <datasource id="sample" typeid="contentrepository">
        <parameter name="versioning">false</parameter>
        <parameter name="versioning.autoupdate.interval">60</parameter>
        <handles>
          <handle>sample</handle>
        </handles>
  </datasource>
  <datasource id="ldap" typeid="ldap">
    <parameter name="binddn">ou=MyApplications,dc=mycompany,dc=com</parameter>
    <handles>
      <handle>ldap_connection1</handle>
      <handle>ldap_connection2</handle>
      <handle>ldap_connection3</handle>
    </handles>
  </datasource>
<datasources>
```

Table 3.45. General Parameters for datasources

Name	Description	Default
backgroundvalidation	Whether the connectivity of handles shall be checked periodically in the background. This option is only available when more than one handle is configured for the datasource. When the background validation is activated and more than one handle is configured for the datasource, all handles are checked every `backgroundvalidation.interval` ms (default: 10 minutes). When a invalid handle is detected (e.g. the server cannot be reached) the handle is taken out of service, until it is checked again after the configured interval.	true
backgroundvalidation.interval	Interval in milliseconds for the background checking of handles.	600000 (10 Minutes)

3.8.6.3.1. Datasources of type `contentrepository`

Datasources of type contentrepository provide access to sql-based databases that must follow a strict database-scheme. See Section 3.4.3, “ContentRepository” for more details. These datasources can only use handles of type `sql` and allow

- Runtime manipulation of objects and attributes stored in the repository
- Multivalue attributes
- Storing of binary data

Datasources of type contentrepository support cache warming, which pre-loads configured objects and attributes into the cache. This is in contrast to standard caching, which does not load and cache these items until they are used for the first time.

Additionally, datasources of type contentrepository also allow filesystem attributes, which moves stored data from the databank to the file system. This allows for streaming of large files and can lead to better performance. In order to activate this option, the attribute.path parameter must be set, which is the absolute path to the base folder where the attributes will be stored.

Table 3.46. Parameters for datasources of type contentrepository

Name	Description	Default
versioning	Whether versioning shall be used for this datasource.	false
versioning.autoupdate.interval	The interval in seconds for the autoupdate job which performs future changes (when `versioning` is set to `true`).	60
cache	With this parameter, the datasource cache can be switched on. Datasource cache will increase performance when accessing objects coming from this datasource.	false
cache.syncchecking	When enabling the datasource cache for data-	false

Name	Description	Default
	sources that are filled by Gentics Content.Node;, you should set this to `true` . When syncchecking is enabled, Gentics Portal.Node checks frequently for an updated contentrepository and will clear all relevant caches.	
cache.syncchecking.interval	Interval in seconds for background checks whether the updatetimestamp of a Content Repository has changed (when cache.syncchecking is enabled).	10
cache.syncchecking.differential	When the datasource parameters *`cache`* and *`cache.syncchecking`* are enabled for this datasource, setting this parameter to `true` will modify the behaviour of clearing the caches when datasource modifications are detected: By default, all datasource caches (queries, objects, attributes) are cleared, but with differential syncchecking, only the modified objects (and their attributes) are removed from the cache (together with all query results).	true
cache.attribute.[attribute name]	When caching for this datasource is activated, all attributes would be cached. With this setting, the caching for the attribute *`attributename`* can be deactivated.	true
cache.attribute.[attribute name].region	With this parameter, the cache region for attribute *`attributename`* can be defined. This can be used to have better control of how specific attributes will be cached.	gentics-portal-contentrepository-atts
cache.foreignlinkattributes	Setting this parameter to `true` will enable the caching also for foreign link attributes (when the cache is enabled at all). This should not be done for writeable datasources, because the cache for foreign linked attributes can not be dirted correctly and this will most likely lead to inconsistent cache data. If you use caching for a datasource filled from Gentics Content.Node; and have *`cache.syncchecking`* set to `true`, you should also set *`cache.foreignlinkattributes`* to `true`.	false
cache.warming.onInit	Enable `true` or disable `false` cache warming on startup.	true
cache.warming.filter	Filter used to determine which objects should be cached. If a syntactically incorrect filter is configured, cache warming will be disabled.	true
cache.warming.attributes	Comma-separated list of attributes to be cached. If no attributes are specified, cache warming will be disabled. Furthermore, if a syntactically incorrect list is entered, cache warming will also be disabled.	empty
attribute.path	The absolute path to the folder that should be used to store file system attributes (additional subfolders will be created here).	empty
sanitycheck	Enable/disable the sanitycheck of the datasource handles on startup. When the sanitycheck is enabled (default), Gentics Portal.Node checks the database structure (tables and columns) of all datasource handles upon startup of the server.	true
sanitycheck2	Enable/disable the extended sanitycheck of the datasource handles on startup. When the sanitycheck2 is enabled (default is disabled), Gen-	false

Name	Description	Default
	tics Portal.Node performs extended checks on the database structure. When an incompatibility is found, the datasource will not be available (and error messages will be found in the server logs).	
autorepair	Flag to overwrite the global autorepair setting. See "Datasources Configuration" (Section 3.8.2.12, p.34) for details on auto repair.	Do not overwrite the global setting
compatibility.integerasstring	Compatibility setting for treating attributes of type 3 (Integer) as Strings. This should only be turned on, when this old and deprecated behaviour is required by the implementation.	false
setUpdatetimestampOnWrite	When this flag is set to `true`, every insert/update/delete on the datasource will also update the timestamp *`lastupdate`* in the table *`contentstatus`*. This setting in conjunction with the parameters *`cache`* and *`cache.syncchecking`* can be used to synchronize caches between multiple Gentics Portal.Node instances that write into the same datasource. Note: This flag should not be used in the CRSync process (because the CRSync updates the *`lastupdate`* timestamp once the synchronization is done, regardless of this flag). Note: For performance reasons it is recommended to set the flag *`cachedbmetadata`* for the datasource handle (see "Handle type `sql` " (Section 3.8.6.2.1, p.49) for details).	false
prefetchAttribute.threshold	This parameter modifies the behaviour for attribute prefetching when the datasource uses caches. See "Attribute Prefetching" (Section 3.4.3.3, p.27) for an explanation of how the attribute prefetching works and can be configured.	1000
prefetchAttribute.cacheMissThreshold	This parameter modifies the behaviour for attribute prefetching when the datasource uses caches. See "Attribute Prefetching" (Section 3.4.3.3, p.27) for an explanation of how the attribute prefetching works and can be configured.	100
prefetchAttribute.cacheMissThresholdPerc	This parameter modifies the behaviour for attribute prefetching when the datasource uses caches. See "Attribute Prefetching" (Section 3.4.3.3, p.27) for an explanation of how the attribute prefetching works and can be configured.	20
autoprefetch	Enabling this feature will have some or all (depending on the setting of *`autoprefetch.attributes`*) optimized attributes be prefetched with a single sql statement.	false
autoprefetch.attributes	Comma separated list of optimized attributes that will be autoprefetched, when *`autoprefetch`* is set to `true` . Leaving this empty will autoprefetch ALL optimized attributes.	[empty] (prefetch all optimized attributes)

3.8.6.3.2. Datasources of type `ldap`

The LDAP datasource currently provides only read access to the ldap attributes. Writing is currently not

supported.

Note

When displaying results from an LDAP datasource, you need to make sure a Rule is set. - To show everything you could use

```
(objectClass=*)
```

to select all.

Table 3.47. Parameters for datasources of type ldap

Name	Description	Default
searchDN	SearchBaseDN for searches in the datasource.	NULL
scope	Scope for searches in the datasource. Must be one of base|one|sub.	take setting from handle.
maxResults	The maximum number of records this LDAP datasource will ever return. (0 for unlimited)	1000
cache	With this parameter, the datasource cache can be switched on. Datasource cache will increase performance when accessing objects coming from this datasource	false
dnattribute	Name of the attribute that will hold the DNs of the fetched LDAP entries. The DN is the unique identifier of an entry in LDAP. To successfully obtain the DNs of LDAP entries, this parameter must be set to an attribute name that does not yet exist as attribute in LDAP itself.	name
binaryattributes	Comma separated list of LDAP attributes that contain binary data. If an attribute with binary data is fetched from LDAP and not listed here, the output is non deterministic.	[empty]

3.8.7. template-section

3.8.7.1. Introduction

The template-section contains configuration for the Gentics Template Engine 2. This configuration consists of four parts:

1. General parameters
2. Implementation specific parameters (velocity)
3. Template loaders
4. Portal-wide template definitions

Example 3.14. Syntax of the template-section

```
<template-section>
  <velocity-parameters>
    <parameter id="...">...</parameter>
    ...
  </velocity-parameters>
  <parameters>
    <parameter id="...">...</parameter>
    ...
  </parameters>
  <template-loaders>
    <template-loader id="...">
      <class-name>...</class-name>
      <parameters>
        <parameter id="...">...</parameter>
        ...
      </parameters>
    </template-loader>
    ...
  </template-loaders>
  <templates>
    <template loader="...">
      <classes>
        <class id="...">...</class>
        ...
      </classes>
      <parameters>
        <parameter id="...">...</parameter>
        ...
      </parameters>
      <source>...</source>
    </template>
    ...
  </templates>
</template-section>
```

3.8.7.2. General Parameters

The overall behaviour of the TemplateEngine2 can be configured with the general parameters.

Table 3.48. General Parameters of the TemplateEngine2

Name	Description	Default
portletapp.loader.modificationCheckInterval	Modification check interval in seconds for all portletapplication template loaders (these are the template loaders which are addressed by `gentics.porletapp`). Set to -1 for no modification check (recommended for production systems).	-1
velocity.allow_caching	Parameter to enable caching of parsed velocity templates.  **Note** This option requires the StringResourceLoader and it will use the cache region: `gentics-velocity-template-processor`. Please make sure to have the follow-	false

Name	Description	Default

	ing velocity-parameters defined:
	
	`<!-- Remove line wraps ! -->`
	`<parameter`
	`  id="string.loader.description">`
	`  String Resource Loader`
	`</parameter>`
	`<parameter`
	`  id="string.resource.loader.class">`
	`org.apache.velocity.runtime.resource`
	`  .loader.StringResourceLoader`
	`</parameter>`
	`<parameter id="resource.loader">`
	`  string`
	`</parameter>`
	`<parameter id="input.encoding">`
	`  UTF-8`
	`</parameter>`	

3.8.7.3. Template Loaders

Template loaders define how templates can be loaded.

3.8.7.3.1. Portletapplication Template Loader

For every deployed portlet application, there will automatically exist a portlet application template loader that loads templates from files in the directory `/WEB-INF/templates` from within the portlet application's context path. This portlet application template loaders are addressed with the *`loaderid`* = `gentics.portletapp` and can only be used in template definitions in the respective portlet application.

Note

Since the *`loaderid`* = `gentics.portletapp` is reserved for the portlet application template loaders, it is not possible to define a custom template loader with this id.

3.8.7.3.2. FileLoader

The FileLoader is implemented by the class `com.gentics.portalnode.templateengine.loader.FileLoader` and loads templates from files, relative to a base directory.

Table 3.49. Parameters of the FileLoader

Name	Default	Description
path	(required)	Base path of the FileLoader. The pathname might (and should) contain system properties. See Section 3.2, “System properties” for details on the available system properties.
modificationCheckInterval	-1	Interval (in seconds) between checks for file changes. Set to -1 for no check changes (recommended for production environments), 0 for checks on every access to the templates.

Table 3.50. Template parameters for Templates loaded from the FileLoader

Name	Default	Description
filename	(required)	Path of the template file, relative to the *path* configured with the FileLoader.

3.8.7.4. Template definitions

The configuration of templates consists of two parts:

1. The Classification to defined what the template is used for (which component will use this template).
2. The loader and parameters to locate the template or the source of the template itself.

Example 3.15. Syntax of a template definition

```
<template loader="...">
  <classes>
        <class id="...">...</class>
        ...
  </classes>
  <parameters>
        <parameter id="...">...</parameter>
        ...
  </parameters>
  <source>...</source>
</template>
```

Table 3.51. Configuration parameters for a template definition

Name	Type	Description
loader	String	Id of the template loader to locate the template source.
classes	Structure	XML Structure of the template classification.
class	Node	One part of the classification. The value is the node value.
class.id	String	Key of the classification part.
parameters	Structure	XML Structure defining the template parameters needed to locate the template with the template loader. See Section 3.8.7.3, “Template Loaders” for additional information.
parameter	Node	One parameter for the template loader.
parameter.id	String	Key of the parameter. Available keys depend on the used template loader and are described there.
source	String	Source of the template, when it could not be located using the template loader, or no loader was given at all.

See Example 4.6, “ViewPlugin Template Customisation Example” for an example.

3.8.8. administration-section

3.8.8.1. Introduction

The administration section holds necessary configuration for administrative purposes such as the deployer used by the AdministrationPortlet to deploy portletapplications.

Example 3.16. Syntax of the administration-section

```
<administration-section>
  <deployer>
    <class-name>...</class-name>
    <parameters>
      <parameter id="...">...</parameter>
      ...
    </parameters>
  </deployer>
</administration-section>
```

3.8.8.2. Deployer

A deployer is a java class the implements the functionality needed to deploy/undeploy portletapplications on the underlying Web Application Server. The configuration consists of the *`class-name`* and specific *`parameters`*.

3.8.8.2.1. TomcatDeployer

The TomcatDeployer is the deployer used for Jakarta Tomcat. It is implemented by the class `com.gentics.portalnode.genericmodules.admin.TomcatDeployer`.

Table 3.52. Parameters for TomcatDeployer

Name	Type	Description
managerUsername	String	Username of a Tomcat user role `manager`. For details on creation of Tomcat users see http://tomcat.apache.org/tomcat-5.5-doc/index.html .
managerPassword	String	Password of the Tomcat user
managerUrlString	URL	URL to access the Tomcat manager application. Typically of the form `http://[hostname]:[port]/manager` (NOT `http://[hostname]:[port]/manager/html` !)
appBase	String	Path to the base directory of webapplications. May contain system properties. Typically `${catalina.base}/webapps` .

3.8.9. pages-section

3.8.9.1. Introduction

The pages-section contains the configuration of the portal pages.

Example 3.17. Syntax of the pages-section

```
<pages-section>
  <parameters>
    <parameter id="...">...</parameter>
    ...
  </parameters>
  <pages>
    <page id="...">
      <rule>...</rule>
      <properties>
        <property id="...">...</property>
        ...
      </properties>
      <positions>
        <position id="..." defaultwindowstate="..." allowedwindowstates="...">
          <portlets>
            <portlet id="..." windowstate="..."/>
            ...
          </portlets>
        </position>
        ...
      </positions>
    </page>
    ...
  </pages>
</pages-section>
```

Table 3.53. Configuration parameters for the pages definition

Name	Type	Description
parameters	Node	Definition of general portal pages parameters.
parameters.parameter	String	The value of a single parameter.
parameters.parameter.id	String	Id of the portal page parameter (must be unique within all portal page parameters).
pages	Node	Configuration of the portal pages.
pages.page	Node	Configuration of a single portal page.
pages.page.id	String	Id of the portal page (must be unique within all portal pages).
pages.page.rule	String	Optional rule to determine accessibility of the portal page. When a portal page has a rule configured, the portal page will only be visible and accessible when the rule matches.
pages.page.properties	Node	Optional set of properties of the portal page.
pages.page.properties.property	String	Value of a single portal page property.
pages.page.properties.property.id	String	Id of the portal page property.
pages.page.positions	Node	Configuration of the positions within the portal page.
pages.page.positions.position	Node	Configuration of a single position within the portal page.
pages.page.positions.po	String	Id of the position. Must be unique within the posi-

Name	Type	Description
sition.id		tions of this portal page.
pages.page.positions.position.defaultwindowstate	String	The default windowstate for portlets in this position. If not specified, the default windowstate is `normal`
pages.page.positions.position.allowedwindowstates	List of Strings (separated by spaces)	List of allowed windowstates for portlets in this position. When not specified, all windowstates are allowed in this position. Any attempt to change the windowstate of a portlet to a restricted value will not modify the windowstate at all. The `portal.events.portlet.onBeforeWindowStateChange` event will be triggered, but not the `portal.events.portlet.onWindowStateChange` event. See "Portlet events" (Section 4.3.5, p.164) for details about the events.
pages.page.positions.position.portlets	Node	Configuration of the portlets contained in the position.
pages.page.positions.position.portlets.portlet	Node	Configuration of a single portlet contained in the position.
pages.page.positions.position.portlets.portlet.id	String	Id of the portlet as defined in a <pnode> in the portal template. See "PNodes" (Section 4.11.1.3, p.193) for details on definition of portlets.
pages.page.positions.position.portlets.portlet.windowstate	String	Windowstate of the portlet in the position.

3.8.9.2. General Parameters for portal pages

Table 3.54.

Name	Description	Default
store	When this parameter is set, the portal server will try to store the possibly customized portlet positions on portal pages, when the user session is invalidated (for example when the user logs out). The portlet positions are serialized into a xml representation and then assigned to the object defined by this parameter. Typically, this parameter will be set to something like `portal.user.customizedsettings` , to store the customized settings into the user content repository, when using a `PNAuthenticationManager` . When a new session is created (for example, when the user logs in), this value is read and merged with the configuration of the portlet positions. It is also possible to store the current portlet position prior to logout by reading the property path `portal.pagesetting` and setting this into e.g. `portal.user.customizedsettings` (or whatever property is defined as store). See "Portal	null (no storing of portlet positions)

Name	Description	Default
	Property Paths" (Section 4.8, p.175) for details about portal property paths. See "Persisting and restoring customized portlet positions" (Section 4.14.4, p.206) for details on how persistence of portlet positions works.	
store.rule	Expression which defines the rule to determine, whether storing/restoring of portlet positions shall be performed. A typical implementation would set this to `portal.user.isloggedin` to store portlet positions only for logged in users.	true

3.8.9.3. Properties for portal pages

Table 3.55.

Name	Description	Default
suppress_gentics_pp	By setting this parameter to `true`, the output of the parameter `gentics.pp` for URLs rendered in this portal page will be suppressed, even if the portal page was rendered using the `gentics.pp` parameter (see "URL Parameters" (Section 3.8.10.5, p.69) for details).	false

Note
Implementation note: Every position in a page may contain each portlet only once.

3.8.10. urlmapping-section

3.8.10.1. Introduction

The urlmapping-section allows configuration of "Beautiful URLs" which are URLs that look like those used for serving static files. For example

```
http://www.gentics.com/Portal.Node/portal/?gentics.rm=LoginModule
```

can be transformed to:

```
http://www.gentics.com/Portal.Node/portal/LoginModule
```

This is achieved by mapping the parameters to entries in the URL. This allows "Beautiful URLs" on the portal layer transparent to the actual portlet. This feature needs to be activated with a portal parameter (see Section 3.8.2.16, "URL Mapping Configuration (Beautiful URLs)").

An URL mapping describes the transformation of parameters to entries and vice versa. It defines which parameter is mapped to a certain position in the generated URL. Multiple different mappings can be defined. When a URL is generated or parsed the appropriate mapping is chosen. This decision is based on the parameters and on the order of the mapping entries. The most precise mapping should be definded in the first entry. If that mapping is not applicable the next one will tried and so on.

Note

Naming: given the URL http://www.gentics.com/Portal.Node/portal/content/pub/directory/file.html

1. `http://www.gentics.com/`: The protocol and host name for the URL. These are currently completely ignored for beautiful URLs.
2. `/Portal.Node`: The Context Path under which the web application is registered in the application server. (Also not used for url mappings.)
3. `/portal`: The Servlet Path, which is used to route the request to the Portal.Node PortalServlet. Other possible values would be `public` or `secure`. (Also not used for url mappings.)
4. `content/pub/directory/file.html`: Pathinfo which consists of multiple entries that are separated by slashes.

When a URL is rendered, the parameters are analyzed to determine whether an appropriate URL mapping can be applied to transform the parameters to entries in the path.

Beautiful URLs try to stay as simple as possible. It basically takes the whole "pathinfo" (Everything in the URL after the context path but before the query string) and separates it into single "segments". (A segment is everything between two slashes (/) within a pathinfo.) Afterwards it matches these segments against the urlmapping configuration from top to bottom. Once a matching urlmapping is found all segments are mapped to the corresponding request parameters and stored in the servlet request which will then be transparently read by Gentics Portal.Node and all portlets.

The main difference between Beautiful URLs and “Bookmarkable URLs” (Section 3.8.2.22, p.38) is that Beautiful URLs work in both directions. Every Portlet URL which is generated will be matched against the urlmappings to create Beautiful URLs.

Example 3.18. Syntax of the urlmapping-section

```
<urlmapping-section>
        <parameters>
                <parameter id="...">...</parameter>
        </parameters>
        <mapping>
                <pathtransformation>
                        <segment>static</segment>
                        <segment urlparameter="..." greedy="..." />
                        ...
                        <hidden urlparameter="..">...</hidden>
                        ...
                </pathtransformation>
                <parameters>
                        <parameter id="...">...</parameter>
                        <parameter id="..."><expression>...</expression></parameter>
                </parameters>
        </mapping>
</urlmapping-section>
```

Table 3.56. Configuration section for the urlmapping definition

Name	Type	Description
parameters	Node	Parameters for all url mappings. See Table 3.57, “Parameters for url mappings”
mapping	Node	Represents one url configuration.
mapping.pathtransformation	Node	Defines, how the transformation from the original url to the path of the beautiful url will be done. Of course this implicitly also defines how incoming beautiful urls will be parsed and re-transformed into the original urls.
mapping.pathtransformation.segment	Node	A segment entry represents one part of the url between two slashes. It can be either defined using a static content, or dynamically mapped to a request parameter.
mapping.pathtransformation.segment.urlparameter	String	Name of the request parameter from the original url that shall be mapped to this segment in the beautiful url.
mapping.pathtransformation.segment.greedy	Boolean	Indicates that this segment can contain one or more slashes (e.g. a directory path). This attribute is optional and there can be at most one greedy segment for each mapping. Furthermore only segments with greedy set to true will match empty URL parameters. (Make sure to check out “Restrictions” (Section 3.8.10.3, p.68)).
mapping.pathtransformation.hidden	Node	This defines, which request parameters will be hidden from the beautiful url. The definition includes the required value of the request parameter, which means that this mapping will only match, when all urlparameters, that will be hidden have the defined values.
mapping.pathtransformation.hidden.urlparameter	String	Name of the request parameter, which must have the defined value and will be hidden, when transforming to a beautiful url
mapping.parameters	Node	Additional configuration parameters for this mapping. See Table 3.57, “Parameters for url mappings”
mapping.parameters.parameter	String or <expression>	Value of the parameter, which can either be static (a String) or an expression.
mapping.parameters.parameter.expression	String	Expression String

Table 3.57. Parameters for url mappings

Name	Description
render.prefix	Affects the prefix for rendered urls. It can be configured either separately for every mapping or globally for the whole urlmapping-section. The value can also be an expression (see “ExpressionParser” (Section 4.10, p.182) and “Generation of URLs that do not start with /Portal.Node/portal” (Section 3.8.10.4, p.68)).

Warning

Once urlmapping is activated (Section 3.8.2.16, “URL Mapping Configuration (Beautiful

URLs)") using the portal parameter "Beautiful URLs" are used for all rendered portlet URLs. Because of the way they work this means that all URLs in your portal will be *absolute* URLs. This means that if your current setup relies on an proxy or anything which transforms incoming URLs you have to be careful and probably use the configuration parameter 'render.prefix' (see "Generation of URLs that do not start with /Portal.Node/portal" (Section 3.8.10.4, p.68) for details). Additionally it has to be noted, that hardcoded relative links to the portal will no longer work as expected and should therefore be made absolute as well.

Example 3.19. URL Config Example

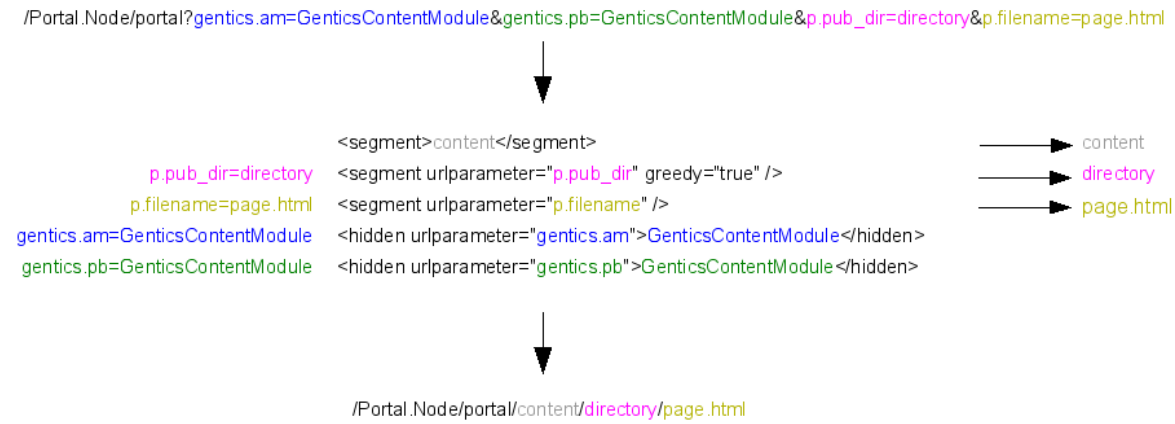
```
<urlmapping-section>
        <mapping>
                <pathtransformation>
                        <segment>content</segment>
                        <segment urlparameter="p.pub_dir" greedy="true" />
                        <segment urlparameter="p.filename" />
                        <hidden urlparameter="gentics.am">GenticsContentModule</hidden>
                        <hidden urlparameter="gentics.pb">GenticsContentModule</hidden>
                </pathtransformation>
        </mapping>
</urlmapping-section>
```

With the above URL mapping configuration the following two URLs are equal:

```
http://localhost:42880/Portal.Node/portal/content/directory/page.html
```

```
http://localhost:42880/Portal.Node/portal
  ?gentics.am=GenticsContentModule
  &gentics.pb=GenticsContentModule
  &p.pub_dir=directory
  &p.filename=page.html
```

The transformation works as following:


/Portal.Node/portal?gentics.am=GenticsContentModule&gentics.pb=GenticsContentModule&p.pub_dir=directory&p.filename=page.html
content
p.pub_dir=directory

p.filename=page.html

gentics.am=GenticsContentModule
<hidden urlparameter="gentics.am">GenticsContentModule</hidden>
gentics.pb=GenticsContentModule
<hidden urlparameter="gentics.pb">GenticsContentModule</hidden>
content
directory
page.html
/Portal.Node/portal/content/directory/page.html

1. The first segment is a static value.
2. The next segment is mapped to the value of the urlparameter ***`p.pubdir=directory`***
3. And the last segment is mapped to the value of the urlparameter ***`p.filename=page.html`***

4. The urlparameter ***gentics.am=GenticsContentModule*** is hidden from the generated URL because this mapping defines a fixed value for this urlparameter and both the urlparameter name and value matched. The same applies for the urlparameter ***gentics.pb***

Because the segment for p.pub_dir is greedy this works even if the value of p.pubdir contains slashes:

```
http://localhost:42880/Portal.Node/portal
  ?gentics.am=GenticsContentModule
  &gentics.pb=GenticsContentModule
  &p.pub_dir=documents/public/pages
  &p.filename=page.html
```

will be transformed to:

```
http://localhost:42880/Portal.Node/portal/content/documents/public/pages/page.html
```

Note

By default the GenticsContentModule does not generate URLs using pub_dir and filename, but by contentid. Please see Section 5.4, “GenticsContentModule” for the configuration settings to modify this behaviour.

Your portal configuration can use multiple different URL mappings. Before a URL is rendered (and eventually transformed) a correct mapping must be selected. This is primarily done using the hidden urlparameters, e.g. if the URL didn't have the urlparameter ***gentics.pb=GenticsContentModule*** this mapping would not be applicable. Thus the URL will be rendered as usual with all the urlparameters.

3.8.10.2. Conflicting configurations

The mapping between the original URL and the beautiful URL must be unique and reversable. This ensures that generated beautiful URLs can be transformed back into the correct original URL. The portal server ensures this reversability during generation of beautiful URLs and will not transform a URL if the result would be ambiguous. The following example shows a configuration with conflicting mappings which would lead to ambiguous URLs:

Example 3.20.

Given the following configuration:

```
<mapping>
        <pathtransformation>
                <segment urlparameter="p.pub_dir"/>
                <segment urlparameter="p.filename"/>
                <hidden urlparameter="gentics.rm">GenticsContentModule</hidden>
        </pathtransformation>
</mapping>
<mapping>
        <pathtransformation>
                <segment urlparameter="gentics.pb"/>
                <segment urlparameter="gentics.ws"/>
                <hidden urlparameter="gentics.rm">SimpleViewExample</hidden>
        </pathtransformation>
</mapping>
```

the URL

```
/Portal.Node/portal?gentics.rm=SimpleViewExample&gentics.pb=main&gentics.ws=maximized
```

would be transformed using the second mapping into:

```
/Portal.Node/portal/main/maximized
```

However the URL

```
/Portal.Node/portal?gentics.rm=GenticsContentModule&p.pub_dir=main&p.filename=maximized
```

would also be transformed to

```
/Portal.Node/portal/main/maximized
```

As you can see the transformation is not reversible and the generated beautiful URL is ambiguous! It is not possible to restore the original parameters correctly. In case of such conflict the original (not beautiful) URL will be rendered as fallback.

3.8.10.3. Restrictions

There are a few restrictions regarding the parameters. In case of violation of any of these rules no mapping will be applied and the classic URL will be rendered.

- Non-greedy segments must not contain any slashes.
- Greedy segments must not start or end with a slash.
- Greedy segments must not contain double-slashes.
- A mapping can have at most one greedy segment.
- The value of a non-greedy segment must not be empty.
- If the value contains special characters it will be encoded according to W3C specifications. The only exception are slashes in greedy segments which will be preserved.

Tip

During development you might want to enable the logger `com.gentics.portalnode.urlmapping.URLMappingConfiguration` at level WARN or even DEBUG.

3.8.10.4. Generation of URLs that do not start with /Portal.Node/portal

In cases, where the generated URLs shall not begin with `/Portal.Node/portal` , the configuration parameter *`render.prefix`* can be used to modify the default behaviour. It can be used in two different ways:

1. Set the configuration parameter *`render.prefix`* to `/Portal.Node` to generate URLs that do not contain the part `/portal` (for example `/Portal.Node/content/pub/dir/filename.html`). Note that it is compulsive that the entry right after `/Portal.Node` is none of
   - /portal
   - /public
   - /secure
   - /go

- /ws
- /profiler
- /static

. The best way to achieve this, is to configure static segments in the first place with different values.

2. Set the configuration parameter to a completely different value (e.g. `/content`). When doing this, it is also necessary to route those URLs to the portal, for example by using a web server in front of the application server, that rewrites the incoming URLs to start with `/Portal.Node/portal` .

3.8.10.4.1. Restrictions of the render.prefix

- The *`render.prefix`* must begin with `/`
- The *`render.prefix`* must not end with `/`

3.8.10.5. URL Parameters

To create optimal url mapping configurations it is useful to know the meaning of the URL parameters used by Gentics Portal.Node.

Table 3.58. Gentics Portal.Node URL parameters

Name	Meaning
gentics.am	Used for action URLs and contains the id of the portlet.
gentics.rm	Used for render URLs and contains the id of the portlet.
gentics.pb	Contains the id of the pbox of the URLs target, when no portal pages are used, otherwise the parameter will contain the position id of the portlet.
gentics.pp	The Portal Page which should be rendered. By adding this parameter, the given Portal Page is rendered, even if it is not the current Portal Page. The current Portal Page will not be changed. When a Portal Page is rendered with this parameter, all portlet or portal URLs in this page will also contain this parameter, unless the Portal Page property `suppress_gentics_pp` (see "Properties for portal pages" (Section 3.8.9.3, p.63) for details) is set to `true`.
gentics.ws	The window state the portlet should be switched to.
gentics.pm	The portlet mode in which the portlet should be rendered.
gentics.ts	The current timestamp used in a few URLs to prevent caching (Should never appear in an url mapping.)
gentics.rl	`true` for portlet reload requests. The response is not a fully rendered portal page but a JSON object. (See "Portlet Reloading" (Section 4.6.3.3, p.171))
p.*	All portlet parameters are prefixed with p. to distin-

Name	Meaning
	guish them from Gentics Portal.Node specific parameters.

3.9. Logging

3.9.1. Introduction

This section describes the customizable logging of portal messages. Logging is useful for administrators and developers mainly for qualitity ensurance of installation, configuration and portlet development. Logging is done with apaches log4j. For details see http://logging.apache.org/log4j/docs/index.html.

3.9.2. Customizing logging

For customization of log levels during runtime of the Portal.Node server, administrators can create/modify the file `${com.gentics.portalnode.confpath}/nodelog.properties` . Changes will be recognized automatically and the logging reconfigured accordingly. By default, only `FATAL` and `ERROR` logs are written. For performance reasons it is discouraged to use log levels other than `FATAL` and `ERROR` in production systems.

Table 3.59. Standardized LogLevels

Name	Description
`FATAL`	log very severe error events that will presumably lead the application to abort.
`ERROR`	log error events that might still allow the application to continue running.
`WARN`	log potentially harmful situations.
`INFO`	log informational messages that highlight the progress of the application at coarse-grained level.
`DEBUG`	log fine-grained informational events that are most useful to debug an application.

3.10. Cache

3.10.1. Introduction

The PortalCache functionality currently relies on the JCS caching system. The default JCS configuration file is located in `tomcat/webapps/Portal.Node/WEB-INF/defaultconfig/cache.ccf`. If you want to customize the cache configuration you can copy this file to `${com.gentics.portalnode.confpath}/cache.ccf` and modify it.

Further information on how to configure the cache is available at the official JCS Website [http://jakarta.apache.org/jcs/]. If your heart is set on another caching system you'll be happy to hear that our PortalCache API is soon to come.

The list of available cache regions including description and reasonable default values can be found in the default configuration file cache.ccf

3.11. Gentics .Node PortalConnector WebService

3.11.1. Introduction

This section is the documentation for the Gentics .Node PortalConnector WebService configuration. The Gentics .Node PortalConnector WebService provides access to datasources defined in Gentics Portal.Node using web services.

3.11.2. Configuration

3.11.2.1. Web Services Configuration File

The web services configuration file for Gentics .Node PortalConnector WebService can be found at `${com.gentics.portalnode.confpath}/webservices/configuration.properties` and is in the Java properties format. When this file does not exist at the startup of Gentics .Node PortalConnector WebService , a new one is created with the default settings (no access allowed).

3.11.2.1.1. General properties

With the general properties, the authentication type and default access can be configured.

3.11.2.1.1.1. defaultProtected

Defines whether access is allowed or forbidden to web services unless defined otherwise. Set to `true` (default) to deny and to `false` to allow access.

3.11.2.1.2. WebService configuration

Configuration of the access properties for web services. Web services are accessed using URLs. Each URL is constructed as `ws-path/servicename[/additional-pathinfo]` (additional-pathinfo is optional). For Gentics .Node PortalConnector WebService , the servicename is `datasource` and the additional-pathinfo is the datasource id.

3.11.2.1.2.1. webservice.protected

Overwrite the general value for *`defaultAccess`* to set the protection for access to the listing of all existing web services.

3.11.2.1.2.2. webservice.role

Set the required userrole to access the listing of all existing web services.

3.11.2.1.2.3. webservice.datasource.protected

Overwrite the general value for *`defaultAccess`* to set the protection for general access to the datasource web service.

3.11.2.1.2.4. webservice.datasource.role

Set the required userrole for general access to the datasource web service. This protects access to the `*.wsdl` file defining the datasource. Access to specific datasources can be configured differently.

3.11.2.1.2.5. webservice.datasource.[datasourceid].protected

Overwrite the general value for *`defaultAccess`* . If set to `true` , the datasource is protected and only users with appropriate roles may access the datasource. If set to `false` , the datasource is not protected and everybody may access it.

3.11.2.1.2.6. webservice.datasource.[datasourceid].role

Define the roles that grant access to the datasource [datasourceid]. Users must have at least one of the given roles to access this datasource over the webservice, when the datasource is protected.

3.11.2.2. User configuration

The users are configured in the file `${com.gentics.portalnode.confpath}/users.properties`.

3.11.2.2.1. user.[login].password

Defines a user and sets a password (which may be empty).

3.11.2.2.2. user.[login].role

Comma separated list of roles the user incorporates. The user will have access to web services that are assigned at least one of the user's roles.

3.11.2.3. Configuration Example

The following example defines the user `adminuser` with password `password` and roles `admin` and `manager` . The user will have access to all three configured datasources, since two have appropriate roles and the third is unprotected.

Example 3.21. Example for ${com.gentics.portalnode.confpath}/users.properties

```
# user configuration
user.adminuser.password=password
user.adminuser.rule=admin,manager
```

Example 3.22. Example for ${com.gentics.portalnode.confpath}/webservices/configuration.properties

```
# per default everything is protected
defaultProtected=true

# webservice configuration

# everybody may list the existing webservices
webservice.protected=false

# access to the .wsdl for the datasource webservice only for admins
webservice.datasource.role=admin

# set the protection for specific datasources
webservice.datasource.protecteddatasource.role=admin
webservice.datasource.unprotecteddatasource.protected=false
webservice.datasource.managerdatasource.role=manager,user
```

3.12. Profiler

3.12.1. Introduction

The Gentics Portal.Node Profiler allows users to inspect the performance of their portal. Various perform-

ance aspects can be easily verified. The main questions answered by the profiler include:

- What is the bottleneck of my portal? (templates, SQL statements, 3rd party portlets, pluggable actions, ...)
- Is the expiration cache applied correctly to all portlets?
- Which portlet requires most time to render or process its actions?
- Should I optimize a Gentics .Node ContentRepository attribute? Is the datasource caching configuration effective?
- For implementations based on the “ViewPlugin” (Section 4.2.2, p.97):
  - How fast are my actions processed? How often are they actually called?
  - Are the velocity templates eating performance?

3.12.2. Accessing the profiler

For security reasons the Gentics Portal.Node Profiler is protected by a security constraint which only allows users with the role `manager` or `gentics` to access the profiler. This requires a corresponding user configuration for your application server.

Example 3.23. Tomcat User Configuration

When using Apache Tomcat, the user configuration is usually handled by the file `tomcat/conf/tocmat-users.xml`. The following is a very simple example for a user with the username: `tomcat` and password: `tomcat` who has access to the Gentics Portal.Node Profiler.

```
<tomcat-users>
  <role rolename="tomcat" />
  <role rolename="manager" />
  <role rolename="gentics" />
  <user name="tomcat" password="tomcat" roles="manager"/>
</tomcat-users>
```

After successfully configuring your application server you can access the Gentics Portal.Node Profiler by opening the url `/Portal.Node/profiler` in your browser. For example for Gentics Portal.Node SDK Users: http://localhost:42880/Portal.Node/profiler.

3.12.3. Commands

Below the list of sections is a list of commands which can be executed:

- Start: Start recording of profiler marks.
- Stop: Stop recording.
- Reset: Discard all recorded profiler marks.
- Refresh: Reload the current page showing up to date information.

- Call GC: Invokes the Garbage Collector of the Java Virtual Machine.

3.12.4. Performance Impact

Before the Gentics Portal.Node Profiler can display detailed performance statistics it has to record invocations of specific profiler marks. This means that starting the profiler can have an impact on two performance aspects of your portal:

- Memory: By default every invocation will be stored in memory for immediate analysis. This can mean up to 200 bytes per invocation. You can see the number of invocations recorded so far in the “Status View” (Section 3.12.5, p.74).
- CPU Performance: Every recorded invocation will have a minor cpu performance impact. Usually this has almost no noticeable effect but it should be considered before activating the profiler in a production environment.

3.12.5. Status View

When first accessing the Gentics Portal.Node Profiler the "Status" view is opened. It gives a very brief overview of the status of the profiler itself, and the Java Virtual Machine in general.

3.12.5.1. Statistic Table

The table to the left shows general information about your installation. The most important data are:

- Profiler Status: Shows if the profiler is currently recording invocations.
- Recorded Invocations: If this is non-zero there are already requests which were recorded and you can switch into the Basic View for analyzing the performance of your portal.
- Memory: Gives you insight in how much memory the Java Virtual Machine currently uses. This is also displayed in the memory graph on the right.
- Portal Server: Displays the version, codename and build number of the Gentics Portal.Node instance currently running on the server.

3.12.5.2. Memory

In a production environment the memory graph to the right should give a good insight if your memory settings are in effect and if the full capacity is used by the portal.

- Free Memory: Memory within the java heap space which is not in use.
- Used Memory: Memory that was allocated by the portal server including garbage memory which will be collected and freed by the JVM automatically or when manually manually invoking the 'Call GC' (Call Garbage Collector) link. (Free Memory + Used Memory = Total Memory currently in use by the JVM process)
- Unallocated Memory: Memory that will be allocated by the JVM before running out of memory. (This maximum memory is usually defined by a JVM specific setting like '-Xmx' for the Sun JVM.)

3.12.6. Profiler Basic

The basic profiler view is the main part of the profiler, giving an overview of the overall performance of the

portal and each portlet.

3.12.6.1. Summary

On the top of the basic view is a short summary which gives you a reference for all performance measurements by providing information on how much requests were recorded and how long they have taken in total.

This can be compared to the duration of every measured point to get an idea how much impact it has on the overall performance.

3.12.6.2. Overview Tab

The overview pie chart gives a good overview of what section is responsible for most of the time of a request. The total 100% of this chart represent the total request duration with every section having its own tab (described below) showing more detailed information. All sections in the pie chart are "exclusive times" only counting times which they are directly responsible for, without anything that is already counted for another section. (e.g. Datasource Queries which are executed within a velocity template are only counted once for 'SQL Queries' and *not* for rendering templates.)

3.12.6.3. Portlets Tab

The Portlets Tab shows the total time every portlet required for its action and render phase. Probably the most important are the number of invocations per portlet because it can be a good indicator if the portlet cache is working properly.

Note

Times in this tab will be much higher from what is shown in the “Overview Tab” (Section 3.12.6.2, p.75). This is because the overview excludes times which are already measured in other sections (like database access, template rendering or pluggable actions).

3.12.6.4. SQL Queries Tab

This tab shows the slowest SQL queries produced by Gentics Portal.Node APIs (within views or by directly using Gentics Portal.Node datasources). (These are usually generated from expression strings from the “ExpressionParser” (Section 4.10, p.182).)

- Invocations: A very high number of invocations can indicate that there is a problem with the datasource cache - for example by including a variable in an expression which changes for every request like a timestamp or counter.
- Time: If there are few very slow queries it might be possible to improve performance by creating optimized attributes for Gentics .Node ContentRepository attributes or by adding additional database indices.

3.12.6.5. Pluggable Actions Tab

Shows the total runtime of all pluggable actions used in views.

3.12.6.6. Templates Tab

This graph shows the slowest velocity templates. It includes the time required by velocity to render the template as well as subsequent method calls not recorded by other invocations. (E.g. generating portlet URLs will be included in these time statistics, but rendering of child components or executing sql statements are not included.)

3.12.7. Profiler Advanced

In the advanced section of the Gentics Portal.Node Profiler every single profiler mark can be seen in a flat overview as well as a hierarchical caller tree.

Warning

Because of the vast amounts of profiler marks and the complexity required to show detailed performance measurements, the advanced profiler section is still considered *beta* and is provided AS IS without any warranty or support.

3.12.7.1. Flat

The flat view lists all recorded profiler marks with their number of invocations and performance statistics. By clicking on a profiler mark you can switch into the tree mode to see a hierarchical invocation tree.

Note

Profiler marks are listed once for every 'instance key' (for example for portlets the name of the portlet) and once summed up without an instance key.

3.12.7.2. Tree

The tree has two modes: The default mode which is opened when clicking on any profiler mark in the flat view shows a tree with the root node being the outer most profiler mark which invokes all others.

The second mode is the 'Caller Tree' which can be opened by right clicking on a profiler mark and selecting 'Caller Tree'. This shows the inner most profiler mark as root node and below it are profiler marks from where it was called.

3.12.7.3. Inclusive / Exclusive Times

In the flat and tree view there are two radio boxes in the header which allow switching between an 'Inclusive' and an 'Exclusive' mode.

The inclusive mode shows times from the start of the profiler mark until the end.

The exclusive mode only shows the time required by the current profiler mark - any time which is already counted for another profiler mark is subtracted.

Note

Exclusive times are calculated no matter how the profiler settings are configured. Usually when adding up all exclusive 'Total' times in the flat view (without view limit) it would result in the same time displayed for the 'Gentics' profiler mark in inclusive mode, or the 'Total recorded request duration' in the Basic Profiler. But when profiler marks are not recorded the sum of all exclusive profiler marks will be much lower.

3.12.7.4. View Limit

The view limit, which can be customized with the combo box in the 'Percent' column, is used to hide all profiler marks under a certain percent threshold.

3.12.7.5. Time Units / 90% Line

By default all time values are displayed in milliseconds (ms). But it can be customized to display either seconds or nano seconds (in a Java 1.5 environment). To do so click on the '(ms)' link in any column

header which displays time values.

The same way it is possible to switch the 90% line to any other xx% of your choosing. (The 90% line shows the average time per invocation for the fastest 90%.)

3.12.7.6. Snapshots

It is possible to store a profiler run in a 'Profiler Snapshot' by simply clicking on the 'Save Snapshot' link. It can then be loaded into any other Gentics Portal.Node (or Gentics Portal.Node SDK) installation.

The 'Load Snapshot' functionality can be used for either snapshots saved through the 'Save Snapshot' link, or for files created by “Recording to file” (Section 3.12.8.1, p.78).

3.12.7.7. Velocity Profiler Mark Directive

To make it possible to benchmark custom templates in greater detail, a velocity directive is provided to allow implementors to add custom profiler marks to their velocity templates.

The profiler mark used has the syntax: #profilermark("profiler mark" "instance key (optional)"). (The profiler mark has to be a string, while the optional instance key can be any serializable object.)

Example 3.24. Profiling component template

The following example profiles the rendering of a simple ButtonComponent.

```
#profilermark( "custom/velocitymark/button" $button.name )
   <input id="$button.name" value="$button.label" name="$button.name" type="submit" />
#end
```

The custom profiler marks can then be seen in the advanced profiler section as shown below. (Make sure that all marks are recorded in the “Recording Settings” (Section 3.12.8.2, p.78) as well as that there is no viewlimit in place which might prevent these marks from showing up.)

element custom null	total (ms)	%	avg (ms)	min (ms)	max (ms)
custom/velocitymark/button	15	9.6%	2	0	13
custom/velocitymark/button -- p.AdministrationPortlet.AdministrationPortlet.teaserview.c.2.btn0.v	0	0.2%	0	0	0
custom/velocitymark/button -- p.AdministrationPortlet.AdministrationPortlet.teaserview.c.3.btn0.v	13	8.5%	13	13	13
custom/velocitymark/button -- p.LanguageSelector.LanguageSelector.portletaccess.c.3.1.btn0.v	0	0.3%	0	0	0
custom/velocitymark/button -- p.LanguageSelector.LanguageSelector.portletaccess.c.3.2.btn0.v	0	0.2%	0	0	0
custom/velocitymark/button -- p.main.ComponentUsageExample.view.c.41.inputcomponentstab.21.3.btn0.v	0	0.2%	0	0	0
custom/velocitymark/button -- p.main.ComponentUsageExample.view.c.41.inputcomponentstab.21.9.calendarButton.btn0.v	0	0.2%	0	0	0

Warning

Adding customized profiler marks can distort the overview of the basic view, because these profiler marks will be excluded from the render times of templates.

3.12.8. Profiler Settings

This section allows users to customize the way the profiler records invocations.

3.12.8.1. Recording to file

Instead of holding all data in memory for analyzing this option allows using the profiler just to record data and imediately write them out into the filesystem without holding them in memory. This is especially useful to allow recording of profiler marks under decent load (e.g. in a minor load test, or even a production environment).

Warning

The profiler will still decrease overall system performance. It is not recommended to be activated on any system which is already under heavy load.

When recording to a file no statistics can be viewed. After finishing a recording session download the file and load it into the Gentics Portal.Node Profiler in the “Profiler Advanced” (Section 3.12.7, p.76) (see Section 3.12.7.6, “Snapshots”).

3.12.8.2. Recording Settings

This setting allows customization on what profiler marks are recorded and therefore limit memory and CPU overhead produced by the Gentics Portal.Node Profiler. The default marks which are activated are the `Minimal Recording Settings` which define a minimum of profiler marks required for the “Profiler Basic” (Section 3.12.6, p.74) . If you are exploring the “Profiler Advanced” (Section 3.12.7, p.76) you can try activating the `Normal Recording Settings` for more runtime information. If even more information is required, for example when using the “Velocity Profiler Mark Directive” (Section 3.12.7.7, p.77) the `Verbose Recording Settings` can be used.

3.12.8.3. System Properties

It is possible to configure the runtime profiler configuration basepath by setting the system property `profiler.config.rootpath`. When configured the profiler will search for the `profiler.properties` file in the given location.

3.12.9. Cache (JCS)

The JCS Cache section shows a brief overview of all cache regions used by Gentics Portal.Node and their total contained items with the corresponding cache hits and cache misses. Please see Section 3.10, “Cache” for more information on the cache and cache regions.

3.13. Portlet Entities Configuration

This section describes the configuration of the portlet entities, when portal pages are used.

Portlet entities define "instances" of portlets and their defining parameters. Every portlet entity is identified by its unique *`id`* and refers to a portlet (definition) via the *`type`* . Note that it is possible to define multiple - differently configured - entities of the same portlet. Portlet entities are also called "modules".

Example 3.25. Structure of the portlet entities configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-entities xmlns="http://www.gentics.com/xml/ns/portal/portletentities">
  <reactions>
    <reaction event="...">
     ...
    </reaction>
```

```
    ...
  </reactions>
  <defaultactions>
    <action name="..." [deprecated alternative: type="..."]>
      <properties>
        <property id="...">...</property>
      </properties>
    </action>
  </defaultactions>
  <pnodes>
    <pnode id="..." type="...">
      <reactions>
        <reaction event="...">
          ...
        </reaction>
        ...
      </reactions>
      <actions>
        <action name="...">
          <visible>
            true|false|[<prule>...</prule>]|[<pobject>...</pbobejct>]
          </visible>
          <properties>
            <property id="...">...</property>
          </properties>
        </action>
      </actions>
      <style>...</style>
      <windowstate>...</windowstate>
      <isvisible>
        true|false|[<prule>...</prule>]|[<pobject>...</pbobejct>]
      </isvisible>
      <title>
        [cdata]|[<prule>...</prule>]|[<pobject>...</pbobejct>]
      </title>
      <titleaction>...</titleaction>
      <parameters>
        <parameter name="...">...</parameter>
        ...
      </parameters>
      <templates>
        <template id="..." loader="...">
          <classes>
            <class id="...">...</class>
            ...
          </classes>
          <source>...</source>
          <parameters>
            <parameter id="...">...</parameter>
          </parameters>
        </template>
        ...
      </templates>
    </pnode>
    ...
  </pnodes>
</portlet-entities>
```

Table 3.60. Portlet Entities Configuration Attributes

Name	Type	Default	Description
portlet-entities	Container	NULL	Container node for all defined portlet entities (pnodes) and general event reactions.

Name	Type	Default	Description
reactions	Container	NULL	Container node for the general event reactions.
reaction	Object	NULL	A single (general) event reaction. The content of this node defines the command that has to be invoked as reaction to the specified event. The command has to be an assignment, formatted as described in “Assignments” (Section 4.10.2.2, p.183) . Note that multiple assignments can be grouped with the functions `do()` and `if()` (see “Functions” (Section 4.10.2.11, p.186)).
reaction.event	String	[required]	The event, that has to be triggered to invoke this reaction
defaultactions	Container	NULL	Allows configuration of default settings (properties) for actions.
defaultactions.action	Object	NULL	Definition of an action.
defaultactions.action.name	String	[required]	Defines the name of the action under which it will be referenced.
defaultactions.action.type (DEPRECATED)	String	NULL	Attribute which is equal to 'name'. Only available for backward compatibility. Do NOT use this attribute.
defaultactions.action.icon (DEPRECATED)	String	NULL	Element available for backward compatibility only (Use properties instead!). Can be accessed in templates through $action.icon
defaultactions.action.alt (DEPRECATED)	String	NULL	Element available for backward compatibility only (Use properties instead!). Can be accessed in templates through $action.alt
defaultactions.action.properties	Collection	NULL	Allows configuration of properties which are available in templates as $action.properties.[property id]
pnodes	Container	NULL	Container node for all pnodes. Every pnode defines a single portlet entity.
pnode	Object	NULL	Definition of a single portlet entity.
pnode.id	String	[required]	Unique identifier of the portlet entity. Often also referred to as ***`moduleid`*** .
pnode.type	String	[required]	Type of the portlet entity. Generally, the type must be in the form `[application-context-path] / [portlet-id]` .
pnode.reactions	Container	NULL	Container node for the entity specific event reactions. The definition of entity specific event reactions is identical to the definition of global event reactions (see above).
pnode.actions	Container	NULL	Container node for definition of portlet actions for this portlet entity. Normally, the portlet actions are rendered in the portlet frame. The default portlet actions are: one for each portlet mode supported by the portlet and one for each of

Name	Type	Default	Description
			the following windowstates: `minimized` , `normal` , `maximized` . With this container, the default portlet actions can be modified: actions can be made invisible or new custom actions can be added.
action	Object	NULL	Definition of a single portlet action
action.name	String	[required]	Name of the action. May be one of the default actions or any other name for a custom action. Custom actions will trigger the event `portal.events.actions.on[Actionname]` when activated by the user (the first character of the action name is made uppercase). See “Custom action events” (Section 4.3.7, p.166) for details on the events.
action.visible	Boolean, prule or pobject	true	Defines whether the action is visible in the portlet frame or not.
action.properties	Collection	NULL	Allows overwriting of properties defined in the defaultactions node. Available in templates as $action.properties.[property id]
style	String	NULL	Style definition of the portlet entity. Style definitions can be used to group portlet entities together in terms of layout. The style definition can be used to classify portlet frame templates, such that all portlets with the same style will be rendered using the same portlet frame templates. See “Portlet Frame Templates” (Section 4.14.2, p.202) for details.
windowstate	String	normal	Default windowstate for the portlet entity. This setting can be overwritten with the windowstate setting when placing the portlet entity in a portal page position.
isvisible	Boolean, prule or pobject	TRUE	Setting for the "visibility" of portlets (e.g. depending in user permission settings)
title	String or pobject	[title rendered by the portlet]	Title of the portlet entity
titleaction	String	NULL	Name of the action that shall be linked to the rendered portlet entity title.
parameters	Container	NULL	Container node for the definition parameters of the portlet entity. The definition parameters (also called pnode parameters) define the general behaviour of the portlet entity. The set of possible parameter names is defined by the portlet implementation.
parameter	Object	NULL	Definition of a single parameter. The parameter value can either be given as content (String) or as embedded `<pobject>` .

Name	Type	Default	Description
parameter.name	String	[required]	Name of the parameter.
templates	Container	NULL	Container node for portlet entity templates (or pnode templates). Templates defined for a specific portlet entity always take precedence over any other (matching) template. See "Template definitions" (Section 3.8.7.4, p.59) for details on template definitions.

Note

The namespace of the portletentities configuration elements must be `http://www.gentics.com/xml/ns/portal/portletentities`, otherwise the configuration cannot be parsed correctly.

When other xml structures are included (e.g. as value of a pnode parameter), it is important to set the correct namespace of the embedded xml structure. For example: The GenticsContentModule has a pnode parameter `ruleactions` that may contain an `actions` node. This node must have an empty namespace, which can be accomplished like:

```
<pnode type="GenticsContentModule" id="..">
  <parameters>
    <parameter name="ruleactions">
      <actions xmlns="">
        ...
      </actions>
    </parameter>
  </parameters>
</pnode>
```

3.13.1. Frame Actions

Frame Actions represent user interaction elements which can be used to change the state of the current portlet window.

The default actions which are displayed for all portlets are (Portlet mode actions will only be displayed if the portlet supports the given portlet mode):

- portletmode.View
- portletmode.Edit
- portletmode.Help
- windowstate.minimized
- windowstate.normal
- windowstate.maximized

By default these actions can be iterated in the portlet frame templates (see Section 4.14.2, "Portlet

Frame Templates" for details). To give template writers more flexibility it is possible to attach properties to actions. These can be configured in the portlet entities configuration file, either within <defaultactions> or overloaded in the given pnode.

Note

Only the actions listed above will be visible for all portlets. If you configure a new (custom) action it has to be explicitly set to visible for every pnode where it should be displayed.

3.13.1.1. Titleactions

Title actions are usually displayed as links of the portlet title. They can be configured using the <titleaction> tag within pnodes. Titleactions can either be one of the predefined actions or a custom action. They will always be available within the template, no matter if the action referred by the titleaction is set to visible or not.

3.14. Customizing Portal.Node web application

Sometimes it might be necessary to customize settings in the Portal.Node web application directory. Examples for this are modifications to the web.xml to customize security roles, install new servlet filters, etc.

Warning

Customizing the Portal.Node.war is generally discouraged and should only be used as a last resort. For example in tomcat it is possible to define global servlet filter in the global `tomcat/conf/web.xml` file.

It is important to understand that any customizations could lead to a much harder time when updating Gentics Portal.Node

If you still need to do modifications to the Portal.Node webapplication it is recommended to write scripts which make those modifications to the Portal.Node.war right before deployment.

Please see the SDK Guide for information on customizing the Portal.Node web application in the Gentics Portal.Node SDK

3.15. GenticsImageStore

The Portal package also contains the GenticsImageStore web application, which brings the GenticsImageStore functionality from Gentics Content.Node into Gentics Portal.Node. This web application needs no further configuration, but just needs to be deployed next to Portal.Node.war web application. For details about the GenticsImageStore see the Gentics Content.Node documentation in the Gentics Infoportal.

With setting the servlet init parameter ***`secret`*** the "Image Resize Validation" (Section 3.16, p.83) is switched on.

The servlet init parameter ***`requestDecorator`*** can be set to the fully qualified class of an implementation of the interface ***`com.gentics.api.imagestore.RequestDecorator`***. The class will be instantiated and will be called for every request that is made to fetch the original image to be resized. In this class, it is possible to modify the `imageUri`, `queryString`, `Cookies` and `headers` for the request to load the image. Consult the API Documentation about details of the interface ***`com.gentics.api.imagestore.RequestDecorator`***.

3.16. Image Resize Validation

Resizing Images in the Gentics Portal.Node Server consumes processing time. Therefore, unsecured requests to resize images may open the Server for DoS attacks. With the Image Resize Validation feature,

the “GenticsImageStore” (Section 3.15, p.83), “GenticsContentPortlet” (Section 5.5, p.216) and “GenticsContentModule” (Section 5.4, p.210) will no longer accept requests to resize images, unless they contain a certain checksum as ***validation*** request parameter.

The Feature is activated by setting the parameter ***secret*** (as pnode parameter for the portlets or init-param for the servlet) to a non-empty String value. Once activated, all requests to resized images must be validated by adding the request parameter ***validation***. The parameter must be constructed as `SHA256([secret][resizepath])` encoded with two-digit hexcode (leading zeros). Where `[resizepath]` is the path containing the resize properties:

Example 3.26. Validating Image Resize Requests

```
Request to /GenticsImageStore/100/auto/prop/images/image.png
must be validated with
SHA256([secret]/100/auto/prop)
e.g. (with "secret" as the secret):
/GenticsImageStore/100/auto/prop/images/image.png?validation=55364166F44A14AC9AE70E86C6
```

```
Request to /GenticsImageStore/auto/200/cropandresize/smart/10/10/100/100/images/image.p
must be validated with
SHA256([secret]/auto/200/cropandresize/smart/10/10/100/100)
e.g. (with "secret" as the secret):
/GenticsImageStore/auto/200/cropandresize/smart/10/10/100/100/images/image.png?validati
```

```
Request to /Portal.Node/portal?gentics.rs=content&gentics.rsid=10008.1&maxwidth=100
must be validated with
SHA256([secret]/100/auto/prop)
e.g. (with "secret" as the secret):
/Portal.Node/portal?gentics.rs=content&gentics.rsid=10008.1&maxwidth=100&validation=553
```

Unvalidated or incorrectly validated requests will get a response with status `403 (Forbidden)`.

Note

The GenticsContentPortlet an GenticsContentModule will always use `prop` as resize mode. If either ***maxwidth*** or ***maxheight*** is not given, the value `auto` is assumed.

Chapter 4. Implementation

4.1. Imps

4.1.1. Introduction

Imps are small helpers, providing different functionality an implementer can use in portlets, supporting TemplateEngine2.

4.1.2. Configuration

Imps must be configured in Portalconfiguration file, see Section 3.8.4, “formatter-section” for detailed information. They provide basic functionality that shall be available in an easy and convenient way in every template. Generally, all configured imps are available through

```
$portal.imps.[id]
```

4.1.3. Date Formatter

4.1.3.1. Introduction

Format a date in a standard or custom format, centrally managed and localizable. The class used is `Class com.gentics.portalnode.formatter.GenticsDateFormatter` .

By default, this imp will support the formats `short` , `date` and `time` (and any custom format passed to the methods directly). With the parameter ***`configuration`*** you can configure the path to a custom configuration file.

4.1.3.2. Configuration

Date formats have to be defined in `META-INF/config/formatter/dateformatter.xml` for use with the dateformatter's methods. See http://java.sun.com/j2se/1.4.2/docs/api/java/text/DateFormat.html and http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html for detailed information on formatting styles. Here is an example configuration:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- date formatter information -->
<date-formats default="STANDARD">
    <date-format id="STANDARD" defaultdate="MEDIUM" defaulttime="MEDIUM"/>
          <date-format id="short" defaultdate="SHORT" defaulttime="SHORT"/>
    <date-format id="time" defaulttime="SHORT"/>
    <date-format id="date" defaultdate="MEDIUM"/>
    <date-format id="custom" defaultdate="dd.MM.yyyy" defaulttime="HH:mm">
      <date language="de">dd.MM.yyyy</date>
      <date language="en">dd/MM/yyyy</date>
      <time language="de">HH:mm</time>
      <time language="en">hh:mm a</time>
    </date-format>
</date-formats>
```

Instead of the Formats `SHORT`, `MEDIUM`, `STANDARD`, `LONG` and `FULL` of the `DateFormat` class, it is also possible to use the Date and Time Patterns of the `SimpleDateFormat` class - this also applys for all Imp methods which have a parameter 'format'.

The subnodes `<date>` and `<time>` can be used for language specific date/time formats. Note that it is generally NOT necessary to define language specific formats for just generating formats that print e.g. month names. Month names will automatically be output in the current or given language. The language specific date/time formats are necessary, when the FORMAT itself shall be different, depending on the language. E.g. the separation characters between day, month and year shall be `.` for german and `/` for english.

4.1.3.3. Methods

All formats have to be predefined in file `dateformatter.xml`.

Table 4.1. Methods

Template Syntax	Description
$portal.imps.date.format()	retrieve current date formatted as defined in dateformatter.xml
$portal.imps.date.format(date)	format provided "date" as defined in file dateformatter.xml
$portal.imps.date.format(date, format)	format provided "date" with "format" defined in file dateformatter.xml
$portal.imps.date.format(date, format, languagecode)	format provided "date" in the given language with "format" defined in file dateformatter.xml
$portal.imps.date.format(format)	get current date formatted with "format" which is defined in file dateformatter.xml
$portal.imps.date.format(format, languagecode)	get current date formatted in the given language with "format" which is defined in file dateformatter.xml
$portal.imps.date.parse(formatteddate)	parse string "formatteddate" into a date object
$portal.imps.date.parse(formatteddate, format)	parse string "formatteddate" which accords to "format" into a date object
$portal.imps.date.parse(formatteddate, format, languagecode)	parse string "formatteddate" which accords to "format" in the given language into a date object
$portal.imps.date.fromTimestamp(timestamp)	parse a unix timestamp into a date object
$portal.imps.date.formatDate(*)	deprecated. use format(*) instead.
$portal.imps.date.parseDate(*)	deprecated. use parse(*) instead.
$portal.imps.date.getRfc3339Timezone()	Get current timezone, formatted according to http://www.ietf.org/rfc/rfc3339.txt
$portal.imps.date.getRfc3339Timezone(date)	Get timezone of given date, formatted according to http://www.ietf.org/rfc/rfc3339.txt
$portal.imps.date.socialTime(date)	Print the given time in a "social time" format, e.g. like "10 seconds ago". The i18n Keys for localization are (`$time` is the placeholder for the time in numbers): • `social.time.now` (right now) • `social.time.seconds` ($time seconds ago) • `social.time.minute` ($time minute ago) • `social.time.minutes` ($time minutes ago)

Template Syntax	Description
	• `social.time.hour` ($time hour ago) • `social.time.hours` ($time hours ago) • `social.time.day` ($time day ago) • `social.time.days` ($time days ago) • `social.time.week` ($time week ago) • `social.time.weeks` ($time weeks ago) • `social.time.month` ($time month ago) • `social.time.months` ($time months ago) • `social.time.year` ($time year ago) • `social.time.years` ($time years ago)

4.1.3.4. Examples

The GenticsDateFormatter may be used to format dates in the way that it was defined in the dateformatter.xml file. Syntax:

```
$portal.imps.date.format()       ## Mar 6, 2006 1:31:46 PM
$portal.imps.date.format("date") ## Mar 6, 2006
$portal.imps.date.format("time") ## 1:31 PM
$portal.imps.date.parse("Mar 6, 2006 1:31:46 PM",
    "STANDARD").hours            ## 13
```

4.1.4. Number Formatter

4.1.4.1. Introduction

The Number Formatter Imp provides you with automated formatting for decimal numbers, currencies and filesizes. Locales are also taken into account. Furthermore it provides you with rounding facilities. The class used is `com.gentics.portalnode.formatter.GenticsNumberFormatter`. There are no configuration parameters.

4.1.4.2. Methods

These are the methods provided by the Number Formatter. Please keep in mind that decimal numbers have to be provided as strings since velocity is incapable of handling floating point notation directly. All floating point numbers are formatted according to the portal's current locale.

Table 4.2. Methods

Template Syntax	Description
$portal.imps.number.floor(string)	top the given value down
$portal.imps.number.ceil(string)	round the value up
$portal.imps.number.round(string, [int])	will round a floating point value for you. if you want to round to a specific precision use the optional

Template Syntax	Description
	parameter to set the number of digits to display.
$portal.imps.number.currency(string, [string])	this function ist used to format a floating point value according to locale settings. A desired locale (eg. en_US, de_AT, ...) may be provided via the second (optional) parameter. If there is no locale provided the function will stick to the portal's locale which is currently selectet and therefore will adapt output if the user switches between languages.
$portal.imps.number.filesize(string, [int])	automatically convert a filesize provided in bytes to the highest unit possible. Currently the largest unit supported is Yottabyte (YB, equals 2^80) which should suit well for all tasks. Decimal prcision can be defined using the second (optional) parameter. If left out a precision of 2 is applied by default.
$portal.imps.number.bytesTo(string, string, [precision])	convert a filesize from bytes to a desired target unit. The filesize is filled in as the first parameter, the desired unit es the second. Units supported are: "B", "KB", "MB", "GB", "TB", "PB", "EB", "ZB", "YB". The third (optional) parameter describes the decimal precision which is applied to the output and defaults to 2.
$portal.imps.number.format(string)	format a number according to current locale

4.1.4.3. Examples

Here are some basic use cases of the Number Formatter.

```
$portal.imps.number.ceil("17.25")              ## Output: 18
$portal.imps.number.floor("17.25")             ## Output: 17
$portal.imps.number.round("17.25")             ## Output: 17
$portal.imps.number.round("17.25",1)           ## Output: 17.3
$portal.imps.number.currency("17.25")          ## Output: €17,25
$portal.imps.number.currency("17.25","en_US") ## Output: $17.25
$portal.imps.number.filesize("1048576")        ## Output: 1 MB
$portal.imps.number.filesize("150000", 2)      ## Output: 146.48 KB
$portal.imps.number.bytesTo("1048576", "KB")  ## Output: 1024.00 KB
$portal.imps.number.bytesTo("1048576", "MB")  ## Output: 1.00 MB
$portal.imps.number.format("1048576.123")      ## Output; 1,048,576.123
```

4.1.5. String Formatter

4.1.5.1. Introduction

Format or work with strings for usage in teasers, html, javascript, etc. The class used is `com.gentics.portalnode.formatter.GenticsStringFormatter` . There are no configuration parameters.

4.1.5.2. Methods

Note

All methods requiring a regular expression use the syntax as described in the Java API documentation for the class `java.util.regex.Pattern`

Table 4.3. Methods

Template Syntax	Description
$portal.imps.string.escapeHTML(string)	escapes special characters (like '<', '>', '&') with entities
$portal.imps.string.escapeJS(string)	escapes special javascript characters (like '\', ''', '"') with \ to make them save for output in javascript strings
$portal.imps.string.stripML(string)	Removes the embedded HTML tags from user input string to prevent potential problems (in fact it removes anything beginning with < and ending with >).
$portal.imps.string.trim(string, length)	trims the given string to be no longer than length characters
$portal.imps.string.regexp(string, pattern, replacement)	replaces in string all occurances of pattern with replacement
$portal.imps.string.testRegex(string, regex)	tests whether string matches regex
$portal.imps.string.trimWords(string, length)	trims all words in string to be no longer than length. longer words are trimmed using ... as ellipsis.
$portal.imps.string.trimWords(string, length, ellipsis)	trims all words in string to be no longer than length. longer words are trimmed using ellipsis.
$portal.imps.string.trimWords(string, length, ellipsis, template)	trims all words in string to be no longer than length. longer words are replaced by template, where $trimmedword will be replaced by the trimmed word and $word with the original one
$portal.imps.string.encodeURL(string[, string])	URL - encodes the specified String and returns it. The encoding used to get the bytes for unsafe characters defaults to "utf-8" but may be given as second parameter (e.g. "iso-8859-1").
$portal.imps.string.implode(Array/Collection, seperator)	Converts the given array or collection into a string seperating the items with the given string seperator.
$portal.imps.string.implode(Array/Collection, separator, prefix, postfix)	See above. In addition the items will be pre and postfixed.
$portal.imps.string.toUpper(string)	Converts the given string to all uppercase.
$portal.imps.string.md5(string)	Compute the md5 hash of the given String.

4.1.5.3. Examples

Example 4.1. implode

Convert an array of numbers into a special string.

```
Array: [1,2,3]
```

Example 4.2. escapeHTML

Fill a HTML textbox inside a velocity template with an escaped value.

```
<textarea name="text">
$portal.imps.string.escapeHTML($value)
</textarea>
```

Example 4.3. escapeJS

```
<script language="JavaScript">
        var message = "$portal.imps.string.escapeJS($value)";
        alert(message);
</script>
```

4.1.6. RuleMatcherImp

4.1.6.1. Introduction

The RuleMatcherImp is used to evaluate rules in templates. The corresponding class is `com.gentics.portalnode.formatter.GenticsRuleMatcherImp` . There are no configuration parameters for the RuleMatcherImp.

4.1.6.2. Methods

Table 4.4. Methods

Template Syntax	Description
$portal.imps.rule.matches(rulestring)	Evaluates the given rule and returns true when the rule matches (false if not)

4.1.6.3. Examples

```
#if($portal.imps.rule.matches("portal.user.systemrole CONTAINSONEOF admin")
        Welcome admin!
#else
        Welcome user!
#end
```

4.1.7. DatasourceQueryImp

4.1.7.1. Introduction

Query any data accessible by supported datasources by rules, loop through results. Support for nestable

queries. The class used is `com.gentics.portalnode.formatter.DatasourceQueryImp` . There are no configuration parameters.

4.1.7.2. Methods

Table 4.5. Methods

Template Syntax	Description
$portal.imps.query.rule = [rulestring]	set the primary rule (for fetching objects)
$portal.imps.query.sortBy = [sortBy]	Sort results by a certain attribute; may also contain a comma separated list of attributes. (optional)
$portal.imps.query.sortOrder = [sortOrder]	set the sortorder to be used. Possible values are: • ASC or ASCENDING for ascending sortorder (default) • DESC or DESCENDING for descending sortorder • NONE for no specific sort order
$portal.imps.query.data.[id] = [data]	set the data parameter [id] to the value [data]. Data parameters can be evaluated in rules and subrules
$portal.imps.query.subrules.[id] = [subrulestring]	set a named subrule. Subrules can be used in primary rules and other subrules
$portal.imps.query.result	Get the list of objects filtered by the primary rule
$portal.imps.query.count	Get the number of objects filtered by the primary rule
$portal.imps.query.reset()	Reset all previously set rules, subrules and data
$portal.imps.query.versionDate	Property `versionDate` (date to be used for versioned queries if the datasource supports versioning)
$portal.imps.query.resetVersionDate()	Reset property `versionDate` (query on current data)
$portal.imps.query.versioning	True when the datasource supports versioning false if not
$portal.imps.query.getDifference(List list1, List list2)	Get the difference of list1 and list2, that means a list of objects contained in list1 but not in list2
$portal.imps.query.getDifferenceCount(List list1, List list2)	Get the difference count of list1 and list2, that means the number of objects contained in list1 but not in list2
$portal.imps.query.getIntersection($list1, $list2)	Get the intersection of list1 and list2, the result is a list of pairs, every element holds in $pair.left the object from $list1 and in $pair.right the object from $list2
$portal.imps.query.prefillAttributes = [attributeslist]	Set a list of attributes (including linked objects and their attributes) that shall be prefilled in the fetched objects (increased performance)
$portal.imps.query.channel = [channelId]	When using a multichannelling contentrepository datasource (MCCR), this will set the current channel. Note that this property is not readable (see $portal.imps.query.channels).
$portal.imps.query.channels	When using a multichannelling contentrepository

Template Syntax	Description
	datasource (MCCR), this will get the list of currently selected channels (one for each node/channel structure). Note that this property is not writable (see $portal.imps.query.channel). This property will return a list of objects, each having an id and a name.

4.1.7.3. DatasourceQueryImp for MCCR datasources

When using the DatasourceQueryImp for multichannelling content repository datasources, the current channels can be read and modified as shown in the following examples:

```
## use the channel 17
#set($portal.imps.query.channel = 17)

## get all currently selected channels
#foreach($channel in $portal.imps.query.channels)
  $channel.name ($channel.id)
#end
```

4.1.7.4. Examples

```
## get all companies
#set ($portal.imps.query.rule = "object.obj_type == 50001")
#set ($companies = $portal.imps.query.result)

## for each company, get all users
#set ($portal.imps.query.rule = "object.obj_type == 50000 &&
    object.organisation == data.organisation")
#set ($portal.imps.query.prefillAttributes = ['firstname','lastname'])
#foreach ($company in $companies)
#set ($portal.imps.query.data.organisation = $company.contentid)
#set ($portal.imps.query.sortOrder = "ASC")
#set ($portal.imps.query.sortBy = "lastname")
#set ($users = $portal.imps.query.result)
<table>
#foreach($user in $users)
  <tr>
    <td>$user.firstname</td>
    <td>$user.lastname</td>
  </tr>
#end
</table>
#end
```

4.1.8. CalculatorImp

4.1.8.1. Introduction

Can be used for calculations in templates. The calculator imp supports arbitrary calculator variables that can be used independently. The class used is `com.gentics.portalnode.formatter.CalculatorImp`. There are no configuration parameters.

4.1.8.2. Methods

Table 4.6. Methods

Template Syntax	Description
$portal.imps.calc.[id].reset()	reset the calculator variable [id] to 0
$portal.imps.calc.[id].add([number])	add a value to the calculator variable [id]
$portal.imps.calc.[id].sub([number])	subtract a value from the calculator variable [id]
$portal.imps.calc.[id].mult([number])	multiply the calculator variable [id] with a value
$portal.imps.calc.[id].div([number])	divide the calculator variable [id] by a value
$portal.imps.calc.[id].precision = [int]	set precision to be used for calculations
$portal.imps.calc.get([id])	See $portal.imps.calc.[id]
$portal.imps.calc.[id]	get the current value of the calculator variable [id]
$portal.imps.calc.[id].format	get the current value of the calculator variable [id] formatted according to current locale

4.1.8.3. Examples

```
## reset the variables mycalc1 and mycalc2
$portal.imps.calc.mycalc1.reset()
$portal.imps.calc.mycalc2.reset()

## change the variables
$portal.imps.calc.mycalc1.add("1.5")
$portal.imps.calc.mycalc2.sub("7000")
$portal.imps.calc.mycalc2.add($portal.imps.calc.mycalc1)

## mycalc2 should now be -6998.5
Result: mycalc2 = <strong>$portal.imps.calc.mycalc2</strong>
```

4.1.9. GenticsPLinkImp

4.1.9.1. Introduction

Can be used to generate Plinks (links to content). The class used is `com.gentics.portalnode.formatter.GenticsPLinkImp`. There are no configuration parameters.

4.1.9.2. Methods

Table 4.7. Methods

Template Syntax	Description
$portal.imps.link.to([contentid])	Generate the URL to the given contentid. The URL will not be XML escaped. When this method is used inside a GenticsContentModule, the link will automatically be directed to that portlet, otherwise the portal-wide configured plinkprocessor will be used.
$portal.imps.link.to([contentid], [escapeXML])	Generate the URL to the given contentid. When *`escapeXML`* is set to `true` , the generated URL will be XML escaped. When this method is used inside a GenticsContentModule, the link will automat-

Template Syntax	Description
	ically be directed to that portlet, otherwise the portal-wide configured plinkprocessor will be used.
$portal.imps.link.to([contentid], [portletid])	Generate the URL to the given contentid. The URL will not be XML escaped. The URL will be directed to the portlet [portletid].
$portal.imps.link.to([contentid], [portletid], [escapeXML])	Generate the URL to the given contentid. When *escapeXML* is set to `true` , the generated URL will be XML escaped. The URL will be directed to the portlet [portletid].

4.1.9.3. Examples

```
Find more <a href="$portal.imps.link.to("10007.1234")">information</a>
```

4.1.10. VelocityToolsImp

The VelocityToolsImp is a wrapper for the Velocity Tools. The syntax of each Tool is described on the Velocity Tools Website [http://velocity.apache.org/tools/devel/generic/].

4.1.10.1. Configuration

Optionally it is possible to configure the location of a xml file (toolbox.xml) as used by XMLToolboxManager
[http://velocity.apache.org/tools/devel/javadoc/org/apache/velocity/tools/view/XMLToolboxManager.html
]. For this define the parameter `configuration` with the file path to the .xml file. (System properties ${myprop} are resolved)

4.1.10.2. Default VelocityTools

The following default VelocityTools are available without changing configuration:

- `$portal.imps.velocityTools.date` - DateTool
- `$portal.imps.velocityTools.math` - MathTool
- `$portal.imps.velocityTools.number` - NumberTool
- `$portal.imps.velocityTools.render` - RenderTool
- `$portal.imps.velocityTools.esc` - EscapeTool
- `$portal.imps.velocityTools.alternator` - AlternatorTool
- `$portal.imps.velocityTools.parser` - ParserTool
- `$portal.imps.velocityTools.list` - ListTool
- `$portal.imps.velocityTools.sort` - SortTool
- `$portal.imps.velocityTools.iterator` - IteratorTool

Example 4.4. Simple Example of the usage of EscapeTool and RenderTool

```
## We don't want to type the whole path each time...
#set( $tools = $portal.imps.velocityTools )
#set( $torender =
  "$tools.esc.getHash()set( $tools.esc.getD()harhar = 'World' ) \
   Hello $tools.esc.getD()harhar !!!" )
Rendering: $torender
<pre>$tools.render.eval( $ctx, $torender )</pre>
```

4.1.11. I18nImp

4.1.11.1. Introduction

Can be used to generate internationalized output in templates. The class used is `com.gentics.portalnode.formatter.I18nImp`. There are no configuration parameters.

4.1.11.2. Methods

Table 4.8. Methods

Template Syntax	Description
$portal.imps.i18n.language	The currently set language, or null if the Imp shall use the current portal language. Note that setting the language of an I18nImp does not affect the portal language.
$portal.imps.i18n.resetLanguage()	Method to reset the currently used language.
$portal.imps.i18n.translate(String key)	Translate the given key into the current language (either set by `$portal.imps.i18n.language` or the portal language).
$portal.imps.i18n.translate(String key, String languageId)	Translate the given key into the language with id *`languageId`*.

4.1.11.3. Examples

```
## set the language to English
#set($portal.imps.i18n.language = "en")
## translate this to the current language (English)
$portal.imps.i18n.translate("welcome")
## and this to German
$portal.imps.i18n.translate("welcome", "de")
```

4.1.12. SortImp

4.1.12.1. Introduction

Can be used to sort Resolvables by arbitrary properties. The class used is `com.gentics.portalnode.formatter.SortImp`. There are no configuration parameters.

Note

Implementation note: The given sort properties might also be paths to properties of resolved

subobjects. The general syntax for sort properties is *[path.to.property][:asc|:desc]* where the postfix determines the sortorder. If neither `:asc` nor `:desc` is given, the sorting will be done ascending for this property.

4.1.12.2. Methods

Table 4.9. Methods

Template Syntax	Description
$portal.imps.sorter.sort(Collection collection, String[] properties[, boolean caseSensitive][, String languageCode])	Sort the given collection (might also be an array or a map) by the given list of properties. The sortorder can be independently set for each property in the list. When sorting by only one property, one can also pass a single String instead of an array of Strings as value for the parameter *properties* . The function parameter *caseSensitive* is optional. The function parameter *languageCode* is optional, and - if given - must be a lower-case, two-letter code as defined by ISO-639-1. See http://www.loc.gov/standards/iso639-2/php/code_list.php for details.

4.1.13. SortImp

4.1.13.1. Introduction

This Imp can be used to get content from a given URL and include it into the template. The class used is `com.gentics.portalnode.formatter.URLIncludeImp`. There are no configuration parameters.

4.1.13.2. Methods

Table 4.10. Methods

Template Syntax	Description
$portal.imps.url.include(String url[, int cacheLifeTime[, int timeout[, String defaultContent]]])	Will include the content fetched from the given *url*. The content will be cached for *cacheLifeTime* seconds (defaults to 300 seconds). The parameter *timeout* defines a timeout for trying to contact the given URL in milliseconds (defaults to 2000 ms). If the parameter *defaultContent* is set to a String (including the empty String), this String will be output in case of a connection error. When set to null, this will cause a render error.

4.2. Plugins

4.2.1. Introduction

Plugins are reusable implementations of standard functionality. Plugins can be used in Portlets that need to provide the plugin's functionality. The Gentics portlet has to register every plugin under a unique name (unique for the specific portlet). This pluginname is used as prefix for plugin parameters when they are defined in the portlet's pnode.

4.2.2. ViewPlugin

4.2.2.1. Introduction

The ViewPlugin is the core of the portal's visualisation engine. It is capable of rendering userinterfaces based on a XML userinterface definition, and small templates. Userinteraction is handled by an action layer.

4.2.2.2. Basic elements of a View

View - is the definition of a dialog of an application. Technically it's a XML file based on a View-Schema, containing Components and PluggableActions.

Component - A Component is a visualisation element, either used for user input, presentation or interaction (e.g. a textfield, a list or a button). Technically Components are Java objects and nodes of the View XML Schema.

PluggableAction - A PluggableAction is used to execute any businesslogic in the userinterface and dynamicly interact with the user. Technically PluggableActions are Java objects and nodes of the View XML Schema.

4.2.2.3. Property Paths

The following Properties are provided by the ViewPlugin - usually accessible by portal.modules.<module id>.plugins.viewplugin. See “Portal Property Paths” (Section 4.8, p.175) for more information.

Table 4.11. ViewPlugin Properties

Name	Type	Description
activeview	Object	The currently active view / can also be used to change the currently active view.

4.2.2.4. Parameters

ViewPlugin parameters are used to configure the source for the XML View.

Table 4.12. ViewPlugin Parameters

Parameter	Type	Description
{pluginname}.views	Node	structure defining the views
{pluginname}.viewbasedir	String	relative path to the base directory, where the views are defined in files - If it couldn't be found relative it is search absolute. - Can contain system properties. (See

Parameter	Type	Description
		Section 3.2, “System properties”)

The ViewPlugin supports two different modes of view configuration: inline (in the pnode) or external (every view is defined in a separate file). When the parameter {pluginname}.viewbasedir is given, views are defined externally and inline views are ignored, otherwise the views have to be defined with the parameter {pluginname}.views.

When a {pluginname}.viewbasedir is given, the directory is searched for view definitions. The given path is interpreted relative to the web-application’s home directory (e.g. …/webapps/Portal.Node/). Every .xml file in the directory is supposed to contain the definition of a single view. Additional parameters can be given in a single file views.properties.

The views.properties file has the following properties:

```
defaultview=myviewname
```

Example 4.5. View Parameters

This example shows how the viewplugin is configured for a portlet in the Section 4.4.4, “Gentics Portletdescriptor” .

```
<portlet>
  <portlet-name>DatasourceListExample</portlet-name>
  <parameter-description>
    <parameter-definition type="string" name="viewplugin.viewbasedir">
                                path of the view definitions
        </parameter-definition>
  </parameter-description>
  <start-parameters>
    <parameter name="viewplugin.viewbasedir">WEB-INF/views/list</parameter>
  </start-parameters>
        ...
</portlet>
```

4.2.2.5. Templates

4.2.2.5.1. Introduction

The ViewPlugin Templates are used to render the Views and their Components.

The plugin comes with a set of example templates. They come with the Portal.Node webapp and are updated during productupdates. The templatefiles can be found in the folders `${com.gentics.portalnode.home}/WEB-INF/templates/FormPlugin2` and `${com.gentics.portalnode.home}/WEB-INF/templates/ViewPlugin` respectively. They are referenced in the Gentics Portlet Descriptor in `${com.gentics.portalnode.home}/WEB-INF/gentics-portlet.xml`.

If you want to customize any of those templates, transfer the template file to your portletapplication or portalserver configurationpath, add a templateloader in your Portal Configuration pointing to your template location. In case you want to customize the template for your whole portal, add a reference to your template in your Portal Configuration's template-section. If you want to customize the template only for one portlet, add it to the templates section of your Gentics Portlet Descriptor. See Section 3.8, “Portal Configuration File” for details on the Portal Configuration and Section 4.4.4, “Gentics Portletdescriptor” for details on the Gentics Portlet Descriptor. Please note, that it is currently not possible to define custom templates Portletapplication-wide.

The variables provided in every template are provided by the portal. See the Section Section 4.8, “Portal Property Paths” for details.

In addition component provides specific variables, see the Section Section 4.2.2.7, “Components” for further details.

Example 4.6. ViewPlugin Template Customisation Example

```
<portlet>
  ...
  <templates>
    <template loader="portalloader">
      <classes>
        <class id="plugin">formplugin2</class>
        <class id="component">TextComponent</class>
      </classes>
      <parameters>
        <parameter id="filename">plugins/FormPlugin2/TextComponent.vm</parameter>
      </parameters>
    </template>
    ...
  </templates>
</portlet>
```

4.2.2.5.2. View Templates

View Templates are identifed via the classes section.

Example 4.7. Syntax of View Template Classes

```
<classes>
  <class id="plugin">ViewPlugin</class>
  <class id="windowstate">...</class>
</classes>
```

Table 4.13. View Template Classes

Class	Required	Description
plugin	x	Must be set to "ViewPlugin".
windowstate	x	Windowstate for which the template is used. Must be one of `normal`, `maximized`.

Customizing templates for specific windowstates can for example be used to show one specific view in the windowstate `normal` and other views in the windowstate `maximized`.

4.2.2.5.3. Form Templates

The form template is classified by the following classes section and renders the <form> tag.

Example 4.8. Syntax of Form Template Classes

```
<classes>
  <class id="plugin">formplugin2</class>
  <class id="form">...</class>
  <class id="class">...</class>
</classes>
```

Table 4.14. Form Template Classes

Class	Required	Description
plugin	x	Must be set to "formplugin2".
form		Id of the rendered form (view). Templates with a form set will never be used for forms that have other ids.
class		Class of the form (view). Templates with a `class` set will only be used for forms that have the same `class` setting.

When a form is rendered, the "best matching" template is used for rendering. A template is matching for a specific form, when it's classes definition is exactly one of the following combinations (order defines priority):

- plugin + form + class

  for all forms with the given id and class.

- plugin + form

  for all forms with the given id.

- plugin + class

  for all forms with the given class.

- plugin

  for all forms. this is the most general form template

Table 4.15. Template Variables for Form Templates

Name	Type	Description
$portal	Object	Main portal object. See Section 4.8, "Portal Property Paths" .
$form	Object	The form object.
$form.name	String	The current form name, as used in the output.
$form.enctype	String	The encoding type of the form (should be used for <form enctype="">).
$form.error	Boolean	True when one of the form components contains an input error, false if not.
$form.components	Collection of objects	Collection of all components in the form.

Name	Type	Description
$form.properties	Map of objects	Map of all custom view properties.
$component in $form.components	Objects	Information object for a component in the form. Note that this object cannot be used to get input data from the component.
$component.id	String	The id of the component.
$component.type	String	The type like "tabcollection".
$component.optional	Boolean	Whether this component must contain a value.
$component.visible	Boolean	Whether this component is visible. Invisible components are automaticaly not rendered.
$component.enabled	Boolean	Whether this component is enabled, and may be interacted with.
$component.label	String	Readable label.
$component.help	String	Help content for the current component.
$component.error	Boolean	True if the current component has invalid content.
$component.errortext	String	Errortext for the current component if error.
$component.focusedfield	String	Name of the focused component field or null if the component is not focused.
$component.focused	Boolean	True when the component is focused, false if not. Only one component per form can be focused.
$component.focus	Boolean	Alias for $component.focused
$component.properties	Map	List of custom component properties.
$content	String	Rendered content of the form.
$actionurl	String	Action url of the form. Posting data to this url will submit the form.

4.2.2.5.4. Component Templates

Component Templates are identified via the classes section. Each component template may be specialized by adding specific class restrictions. Following classes are available for all templates, some of them are required within some special templates.

Example 4.9. Syntax of Component Template Classes

```
<classes>
  <class id="plugin">formplugin2</class>
  <class id="component">...</class>
  <class id="form">...</class>
  <class id="class">...</class>
  <class id="id">...</class>
</classes>
```

Note

Please note that "plugin" is always set to "formplugin2", even though templates are defined for the ViewPlugin since the ViewPlugin internally utilizes the FormPlugin2 to render components.

Table 4.16. Component Template Classes

Class	Required	Description
plugin	x	Must be set to "formplugin2".
component	x	Must be set to the type of the component. Identifies the template to belong to a specific component type (e.g. "TextComponent", ...).
form		Id of the form (view) this component shall be used. Templates with a form set will never be used for forms that have other ids.
class		Components class. Template with a `class` set will only be used for components that have the same `class` setting.
id		Id of the component. Templates with a component id set will only be used for components (typically only one) with that id.

When a component is rendered, the "best matching" template is used for rendering. A template is matching for a specific component, when it's classes definition is exactly one of the following combinations (order defines priority):

- plugin + component + form + id

  for the one component of the given type in the form and the given id. This is the most special template for a specific component.

- plugin + component + form + class

  for all components of the given type and class in the given form.

- plugin + component + form

  for all components of the given type in the form.

- plugin + component + id

  for all components of the given type and the id (in all forms).

- plugin + component + class

  for all components of the given type and class (in all forms).

- plugin + component

  for all components of the given type. This is the most general template.

4.2.2.5.5. Custom Action URLs / Events

It is possible to create an action URL within every Plugin template which triggers an event which can be used together with a reaction. (See Section 4.11.1, “Portaltemplate”)

Example 4.10. Custom Action URL in Template

```
#set( $url = $javax.portlet.response.createActionURL() )
$url.setParameter( "action", "myEvent" )
$url.setParameter( "myParameter", "some value" )
```

```
<a href="$url">Some URL</a><br/>
```

If this code is in a component template, e.g. of a TextComponent in a pnode with 'TestPortlet' the path for the reaction would be: portal.modules.TestPortlet.plugins.formplugin2.onMyEvent. Custom parameters may also be accessed via event.properties.*

4.2.2.6. Views

The View XML file is based on the XML Schema pn4doc_views.xsd and can currently be found in the Gentics .Node® Infoportal.

When defining a View XML it already has an embedded 'ListComponent' which is used to define the initial layout of the view. All components within a view are layed out using the template for a ListComponent.

By default the view files are checked for modifications on each access in Gentics Portal.Node SDK installations. To improve performance you can disable this behaviour with the portal.viewplugin.checkviewchanges parameter. See “ViewPlugin Configuration” (Section 3.8.2.11, p.33) for more information.

Example 4.11. Syntax Views

```
<view id="" class="" label="" doubleclickprotection="" xmlns="http://www.gentics.com/xm
  <onload>
    <actions>
      ...
    </actions>
  </onload>
  <onview>
    <actions>
      ...
    </actions>
  </onview>
  <callableactions>
    <callableaction id="">
        <actions>
          ...
        </actions>
    </callableaction>
  </callableactions>
  <properties>
    <property id="..">...</property>
    ...
  </properties>
  ...
</view>
```

Table 4.17. Settings Views

Name	Type	Default	Description
id	String	(required)	Identifier for the view, must be unique in this viewbasedir.
class	String	NULL	Identifier for class used for the embedded ListComponent (This allows you to use different templates depending on the views class).

Name	Type	Default	Description
label	String	NULL	Label of the view
doubleclickprotection	Boolean	NULL	With this flag, the portal-wide or ViewPortlet-wide setting for the double-click protection can be overwritten for a specific view. The default is to not overwrite the setting (which would default to `true` if not set differently). See "ViewPortlet" (Section 5.3, p.209) and "ViewPlugin Configuration" (Section 3.8.2.11, p.33) for details.
xmlns	String	NULL	can be set to `http://www.gentics.com/xml/ns/portal/view` to enable autocompletion and syntax checking when the view is edited in Gentics Portal.Node SDK
onload	Node	NULL	Container for Pluggable Actions that shall be done once for every user session. The actions will be performed when the view is rendered the first time. For details on Pluggable Actions see Section 4.2.2.8, "Pluggable Actions"
onview	Node	NULL	Container for Pluggable Actions that shall be done whenever the view is rendered. When a view is shown more than once in the same portal page, the actions are only performed once. For details on Pluggable Actions see Section 4.2.2.8, "Pluggable Actions"
callableactions	Node	NULL	Container for Callable Actions that can be executed using AJAX requests by the browser.
callableaction	Node	NULL	A Callable Action which can be called by the browser using an AJAX request.
callableaction.id	String	(required)	The identifier of the Callable Action.
callableaction.actions	Node	NULL	Container for Pluggable Actions that are executed whenever the Callable Action is called. For details on Pluggable Actions see Section 4.2.2.8, "Pluggable Actions". Special Pluggable Actions are used to define the return value that is sent to the client, see Section 4.2.2.8.19, "PlainCallableActionResponseAction", Section 4.2.2.8.16, "JsonCallableActionResponseAction" and Section 4.2.2.8.3, "BinaryCallableActionResponseAction"
properties	Node	NULL	Container for all custom view properties to be initially set to this view.

Table 4.18. Template Variables for Views

Name	Type	Description
$views	List	All defined views. See Section 4.8.2, "Views"

Name	Type	Description
$activeview	Object	Currently active view object. See Section 4.8.2, “Views”
$form	String	The rendered content of the currently active view.

4.2.2.7. Components

4.2.2.7.1. All Components

All components have some basic properties to define their templates and settings. Id and class are available for every component, and are not listed within each component description.

Example 4.12. Syntax All Components

```
<{component} id="" class="">
        <label></label>
        <optional>[<prule></prule>]</optional>
        <visible>[<prule></prule>]</visible>
        <enabled>[<prule></prule>]</enabled>
        <help></help>
        <properties>
                <property id="[key]">[value]</property>
                <property id="[key]">[value]</property>
                ...
        </properties>
</{component}>
```

Table 4.19. Settings All Components

Name	Type	Default	Description
id	String	NULL	Id of this component. Can be used in template properties to define a template for a specific component.
class	String	NULL	Class of this component. The class can be used to group components of the same type to use the same template.
label	String	{required}	Readable label. The label should be the i18n key of the real labels in the dictionaries for multilanguage support.
optional	Boolean or prule	TRUE	Whether this component is optional or not. Components that are not optional must be filled in order to successfully submit the whole form and will produce an error if not filled.
visible	Boolean or prule	TRUE	Whether this component is visible or not.
enabled	Boolean or prule	TRUE	Whether this component is enabled or disabled. Disabled components show their values but cannot be modified by the user.
help	String	NULL	Help content for the current component (i18n key).
clearErrors	String	NULL	Set to "true" to clear previous errors.

Name	Type	Default	Description
properties	List	NULL	List of custom properties of the component. The properties can be read in the components template.

Optional, visible and enabled may contain <prule>s, that can resolve the base objects "portal.*", "view.*", "views.*" and "module.*". See Section 4.8, “Portal Property Paths” for details on property paths.

Table 4.20. Template Variables for All Components

Name	Type	Description
$portal	Object	Main portal object. See Section 4.8, “Portal Property Paths” .
$module	Object	The current portlet in which the component is rendered. See Section 4.8.3, “Modules”.
$form	Object	Main form object.
$form.name	String	The current form name, as used in the output.
$component	Object	The current component.
$component.id	String	The id of the component.
$component.type	String	The type like "tabcollection".
$component.optional	Boolean	Whether this component must contain a value.
$component.visible	Boolean	Whether this component is visible. Invisible components are automaticaly not rendered.
$component.enabled	Boolean	Whether this component is enabled, and may be interacted with.
$component.label	String	Readable label.
$component.help	String	Help content for the current component.
$component.error	Boolean	True if the current component has invalid content.
$component.errortext	String	Errortext for the current component if error.
$component.focusedfield	String	Name of the focused component field or null if the component is not focused.
$component.focused	Boolean	True when the component is focused, false if not. Only one component per form can be focused.
$component.focus	Boolean	Alias for $component.focused
$component.properties	Map	List of custom component properties.

4.2.2.7.2. Input Components

These components are used to enter data of any kind (e.g. text, files, options).

4.2.2.7.2.1. ButtonComponent

Buttons used to handle forms or fields.

Example 4.13. Syntax ButtonComponent

```
<buttoncomponent>
        <errorcheck component=""></errorcheck>
        <actions>...</actions>
</buttoncomponent>
```

Table 4.21. Settings ButtonComponent

Name	Type	Default	Description
errorcheck	Boolean	false	Whether the button shall check components for errors upon submit.
errorcheck.component	String	root component of the form	Id of the component were errorchecking shall be started
actions	Node	NULL	Container for pluggable actions that shall be triggered by the buttoncomponent. See Section 4.2.2.8, “Pluggable Actions”

A ButtonComponent renders a button. For the button is stored in a list, but only one button exists.

Note

When errorchecking is used for a ButtonComponent, there are some things the implementor has to care about:

- When no *`errorcheck.component`* is configured, errorchecking will be done for the main Form, even if the ButtonComponent is nested inside a NestedFormComponent.
- When a form is checked for errors, eventually nested forms (inside a NestedFormComponent) are NOT checked, since they are considered to be different forms. Checking multiple forms (e.g. the main form an a nested form) can be accomplished by usage of the “CheckErrorsAction” (Section 4.2.2.8.5, p.141) .

Table 4.22. Template Variables for ButtonComponent

Name	Type	Description
$component.buttons	List	List containing the button.
$button IN $component.buttons	Object	The button.
$button.label	String	Readable label of the button
$button.enabled	Boolean	If button is enabled and clickable or not.
$button.name	String	Unique identifier of this button.
$button	String	Placeholder for rendering the button (deprecated).

ButtonComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/ButtonComponent.xml`

4.2.2.7.2.2. CaptchaComponent

Component which can be used to verify that the current user is no robot. (Currently displays a simple text captcha.)

Example 4.14. Syntax CaptchaComponent

```
<captchacomponent>
</captchacomponent>
```

Table 4.23. Settings CaptchaComponent

Name	Type	Default	Description
No Settings.			

Table 4.24. Template Variables for CaptchaComponent

Name	Type	Description
$component.captchaurl	Resource URL	The resource URL under which the captcha can be retrieved.
$component.captchawidth	Integer	Width of the captcha image in px.
$component.captchaheight	Integer	Height of the captcha image in px.
$component.name	String	Internal name of the component, used as html property.

Table 4.25. Error Keys for CaptchaComponent

Key	Description
fill out image captcha	Key of the error message when the captcha was not filled
Invalid captcha	Key of the error message when the captcha was incorrectly filled

4.2.2.7.2.3. CheckboxComponent

Component used to display a checkbox.

Example 4.15. Syntax CheckboxComponent

```
<checkboxcomponent>
        <truevalue></truevalue>
        <falsevalue></falsevalue>
</checkboxcomponent>
```

Table 4.26. Settings CheckboxComponent

Name	Type	Default	Description
truevalue	String	true	Value that will be assigned if checkbox is

Name	Type	Default	Description
			checked
falsevalue	String	false	Value that will be assigned if checkbox is unchecked

Table 4.27. Template Variables for CheckboxComponent

Name	Type	Description
$component.selected	Boolean	True if checkbox is selected.

CheckboxComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/CheckboxComponent.xml`

4.2.2.7.2.4. DatasourceSelectComponent

A single or multi selectcomponent with Resolvables matching the given rule.

Note

Altough this component can be used with every datasource, you should make sure that all requirements for the used datasource are provided.

For example “Datasources of type `ldap`” (Section 3.8.6.3.2, p.55) require that a rule is given. Otherwise it will not display any results.

Example 4.16. Syntax DatasourceSelectComponent

```
<datasourceselectcomponent>
        <datasource></datasource>
        <rule></rule>
        <nameattribute></nameattribute>
        <valueattribute></valueattribute>
        <multivalue></multivalue>
        <sortby></sortby>
        <sortorder></sortorder>
</datasourceselectcomponent>
```

Table 4.28. Settings DatasourceSelectComponent

Name	Type	Default	Description
datasource	String	default datasource of the portlet	Datasource to use for the objects
rule	String	NULL	Rule to filter the shown objects. The current object can be accessed via the "object" keyword when assembling rules.
nameattribute	String	(required)	Attribute to be shown in the select box
valueattribute	String	NULL	Attribute to be used as values of the selected objects. When this setting is used,

Name	Type	Default	Description
			the component will return the referenced attribute values of the selected objects. Otherwise the component will return the selected objects themselves.
multivalue	Boolean	false	Whether multiple values can be selected.
sortby	String	NULL	Attributes by which the objects are sorted
sortorder	String	ASC	sortorder ASC for ascending, DESC for descending

Table 4.29. Template Variables for DatasourceSelectComponent

Name	Type	Description
$component.multivalue	Boolean	Whether the component allows selection of multiple values
$component.name	String	Internal name of the component, used as html property.
$component.items	List	All options.
$item IN $component.items	Object	Item object.
$item.object	Object	Resolvable object of this item.
$item.object(.*)	Object	Properties of the Resolvable object.
$item.selected	Boolean	True if current item is selected.
$item.name	String	Displayname of the item. Note: when the name is found in a dictionary, the translated String is rendered here. If this behaviour is not desired, implementations may use $item.nameKey instead.
$item.nameKey	String	
$item.value	String	Value of the select item.

Table 4.30. Error Keys for DatasourceSelectComponent

Key	Description
Error required field must not be empty	Key of the error message when a mandatory field was not filled

DatasourceSelectComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/DatasourceSelectComponent.xml`

4.2.2.7.2.5. DateComponent

Component used to input Date/Time/Datetime.

Example 4.17. Syntax DateComponent

```
<datecomponent>
        <mindate></mindate>
        <maxdate></maxdate>
        <dateformatterid></dateformatterid>
        <dateformat></dateformat>
        <format-description></format-description>
        <help></help>
</datecomponent>
```

Table 4.31. Settings DateComponent

Name	Type	Default	Description
mindate	Date	NULL	Earliest allowed date. Must be given in the form yyyy-MM-dd
maxdate	Date	NULL	Latest allowed date. Must be given in the form yyyy-MM-dd
dateformatterid	String	date	Id of the dateformatter imp.
dateformat	String	NULL	Format of the date. Must be a format definition that can be interpreted by the dateformatter imp.
format-description	String	NULL	Error message shown when the input could not be interpreted as date. Supports variable $date to show a correctly formatted date. If mindate and/or maxdate are set, additional variables $mindate and/or $maxdate are available.
help	String	NULL	Help message. Supports variable $date to show a correctly formatted date. If mindate and/or maxdate are set, additional variables $mindate and/or $maxdate are available.

Table 4.32. Template Variables for DateComponent

Name	Type	Description
$component.calendarbuttoncomponent	Object	Buttoncomponent to open and close the calendar. The Calendari s initially closed. $textcomponent is readonly while $calendar.open is true.
$component.calendar	Object	Calendar Object.
$component.calendar.open	Boolean	True if calendar is open, false if closed. Is set, after $calendarbuttoncomponent click is processed.
$component.textcomponent	Object	Textcomponent for display and text input of date.
$component.monthselectcomponent	Object	Selectbox for months.
$component.yearselectcomponent	Object	Selectbox for months.
$component.applybuttoncomponent	Object	Apply button for the year and month selectbox, used for users who have no javascript.
$component.weekdays	List	List of Weekdays (in correct order, startin with the first weekday in a week.)

Name	Type	Description
$weekday IN $component.weekdays	Object	Day of week.
$weekday.date	Date	Date object of weekday (the first fitting weekday of this month, 0:00)
$component.weeks	List	List of weeks to be shown in the calendar.
$week IN $weeks	Object	A single week of the calendar.
$week.date	Date	First day in the week. (Can be used to show the # of the week.)
$week.days	List	List of days of the week.
$day IN $week.days	Object	Day object.
$day.date	Date	Date object of day (0:00)
$day.selected	Boolean	True if this is the currently selected day.
$day.today	Boolean	True if this is the current day. (today)
$day.url	String	URL to select the day and close the calendar again.
$day.selectable	Boolean	Wether the day can be selected, because of min/max date.
$component.calendar.today	Object	Today object.
$component.calendar.today.url	String	URL to store the day and close the calendar again.
$component.calendar.today.date	Date	Date object of today (0:00)
$component.calendar.mindate	Date	Earliest day allowed.
$component.calendar.maxdate	Date	Latest date allowed.
$component.calendar.dateformat	String	Dateformat configured for the datecomponent.

Table 4.33. Error Keys for DateComponent

Key	Description
Error no empty text allowed	Key of the error message when a mandatory field was not filled

Table 4.34. Error Keys for DateComponent

Key	Description
Invalid date syntax	Key of the error message when the date component was filled with an invalid format.

DateComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/DateComponent.xml`

4.2.2.7.2.6. FileUploadComponent

Component used to handle binary data (upload and download files). When the components data is stored into a Resolvable with a Form2ObjectAction, the binary data is stored in the attribute [id] and additional data (contenttype, filesize, uploaddate, filename) is stored in the configured attributes.

Example 4.18. Syntax FileUploadComponent

```
<fileuploadcomponent>
        <contenttypeattribute></contenttypeattribute>
        <filesizeattribute></filesizeattribute>
        <uploaddateattribute></uploaddateattribute>
        <filenameattribute></filenameattribute>
        <keepcontent></keepcontent>
        <downloadactions>...</downloadactions>
</fileuploadcomponent>
```

Table 4.35. Settings FileUploadComponent

Name	Type	Default	Description
contenttypeattribute	String	(required)	name of the attribute where the contenttype is stored
filesizeattribute	String	NULL	name of the attribute where the filesize is stored
uploaddateattribute	String	NULL	name of the attribute where the uploaddate is stored
filenameattribute	String	(required)	name of the attribute where the filename is stored
keepcontent	Boolean	true	Whether to keep the uploaded file in the session after the upload.
downloadactions	Node	false	Container for pluggable actions that shall be triggered when the binary data is downloaded. See Section 4.2.2.8, “Pluggable Actions”

Note

Some Browsers (e.g. Microsoft® Internet Explorer or Opera) are sending the full path to a file, instead of just the filename. This is circumvented by stripping everything from the beginning to the last / or \ from the filename.

The full path can still be retrieved by the component property "*originalFilename*" , or just the path with "*filepath*" right after the upload.

Table 4.36. Component Properties for FileUploadComponent

Name	Type	Description
data	InputStream	Alias for filestream.
filestream	InputStream	The content of the uploaded file.
filename	String	The filename.
filesize	Long	The filesize in bytes.
contenttype	String	The delivered contenttype of the file.

Name	Type	Description
filepath	String	The locale path the file was uploaded from, if available.
viewurl	String	The viewurl that was created during the last rendering process. This url can be used to download the current binary data with Content-Disposition: inline.
downloadurl	String	The downloadurl that was created during the last rendering process. This url can be used to download the current binary data with Content-Disposition: attachment.

Table 4.37. Template Variables for FileUploadComponent

Name	Type	Description
$component.filesize	Integer	Size of the uploaded file (if available) in bytes.
$component.contenttype	String	Contenttype of the uploaded file (if available).
$component.filename	String	name of the uploaded file (if available).
$component.viewurl	String	URL to view the uploaded file in a new window (disposition:inline).
$component.downloadurl	String	URL to download the uploaded file (disposition:attachment).

Table 4.38. Error Keys for FileUploadComponent

Key	Description
file doesn't match current pattern	The uploaded file did not match the given filename pattern
file too big	The uploaded file exceeded the file upload limit
No file uploaded	No file was uploaded for a mandatory component
file upload error	Another file upload error occurred.

FileUploadComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/DownloadUploadComponents.xml`

4.2.2.7.2.7. NumberComponent

Component used to input integer or floating point (double) numbers.

Example 4.19. Syntax NumberComponent

```
<numbercomponent>
  <min></min>
  <max></max>
  <type></type>
  <fractiondigits failifmore=""></fractiondigits>
</numbercomponent>
```

Table 4.39. Settings NumberComponent

Name	Type	Default	Description
min	Number	NULL	Minimum number allowed to be entered. Must be entered in the "internal" format for numbers, like: `-1003.14159`. When not entered, there is no lower limit for the entered number.
max	Number	NULL	Maximum number allowed to be entered. Must be entered in the "internal" format for numbers, like: `-1003.14159`. When not entered, there is no upper limit for the entered number.
type	"integer"|"double"	double	Type of the entered number.
fractiondigits	Integer	0	Number of allowed fraction digits, when the type is set to `Double`. When more fraction digits are entered, the number will be rounded or an error thrown, depending on the setting of *`failifmore`*.
fractiondigits.failifmore	Boolean	FALSE	Behaviour when more than *`fractiondigits`* are entered for `Double` numbers. When set to `true`, an error will be thrown, when set to `false`, the number will be rounded.

Table 4.40. Template Variables for NumberComponent

Name	Type	Description
$component.value	String	Current value.
$component.name	String	Internal name of the component, used as html property.

Table 4.41. Error Keys for NumberComponent

Key	Description
Error no number entered	No number entered for a mandatory component.
Error Not a valid number	The entered text was not a number
Error Number not in range	The entered number was not in the valid range
Error Too many fraction digits given	The entered number had too many fraction digits.

NumberComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/NumberComponent.xml`

4.2.2.7.2.8. PasswordComponent

Singleline textfield with error checking by regexes. The component can render a second input field for verification of the input.

Example 4.20. Syntax PasswordComponent

```
<passwordcomponent>
        <encrypted></encrypted>
        <labelverify></labelverify>
        <syntax></syntax>
        <syntax-description></syntax-description>
</passwordcomponent>
```

Table 4.42. Settings PasswordComponent

Name	Type	Default	Description
encrypted	Boolean	true	Whether to provide the password cleartext or encrypted.
labelverify	String	Verify	Label for the verify field.
syntax	String	NULL	Regular expression the value is matched against. Creates an error in case of mismatch.
syntax-description	String	Invalid Syntax	Readable description of the syntax. Will be displayed as error message.

Table 4.43. Component Properties for PasswordComponent

Name	Type	Description
data	String	The given password, either cleartext or encrypted depending on the component's settings
encrypted	String	The given password, encrypted. The algorithm used is MD5 in a 32 characters (0-9A-F) representation independent of the component's settings.
cleartext	String	The given password, cleartext. If encryption is enabled this will be NULL as long as the user has not entered a new password.

Table 4.44. Template Variables for PasswordComponent

Name	Type	Description
$component.namepassword	String	Internal name of the component, used as html property.
$component.namepasswordverify	String	Internal name of the verify component, used as html property.
$component.labelverify	String	Label for the verify-password input field.

Table 4.45. Error Keys for PasswordComponent

Key	Description
Error: Please enter a password!	No password was entered for a mandatory component.

Key	Description
Error: Passwords do not match!	The password was not entered identically in both fields.

PasswordComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/PasswordComponent.xml`

4.2.2.7.2.9. SelectComponent

Component used to select static options.

Example 4.21. Syntax SelectComponent

```
<selectcomponent>
        <multivalue></multivalue>
        <options>
                <option id=""></option>
                ...
        </options>
        <optionsproperty type="" sortorder=""></optionsproperty>
        <nameattribute></nameattribute>
        <valueattribute></valueattribute>
</selectcomponent>
```

Table 4.46. Settings SelectComponent

Name	Type	Default	Description
multivalue	Boolean	false	Whether more than one value can be selected
options	Node	NULL	list of static options
options.option	String	NULL	The shown option value
options.option.id	String	NULL	The stored option id
optionsproperty	String	NULL	Path to a portal property holding an encoded map of selectable values
optionsproperty.type	String	keyfirst	Either `keyfirst` or `valuefirst`. Whether the key comes first in the encoded map or the value
optionsproperty.sortorder	String (asc|desc)	NULL	Sortorder for the options defined in the optionsproperty.
nameattribute	String	NULL	If the options are set as a collection, use this attribute to read the name.
valueattribute	String	NULL	If the options are set as a collection, use this attribute to read the value.

Table 4.47. Template Variables for SelectComponent

Name	Type	Description
$component.items	List	All options.

Name	Type	Description
$item IN $component.items	Object	Item object.
$item.name	String	Display name of option.
$item.value	String	Internal value of option. May be null (see $component.hasvalue)
$item.selected	String	Displayname of item.
$item.object	Object	Object representation of the item.
$component.hasvalues	Boolean	True if $component.items have values and names, false if $component.items have only names.
$component.name	String	Internal name of the component, used as html property.
$component.multiple	Boolean	True if multiple selections are allowed.
$component.hassize	Boolean	True if a specific size has been set for this component.
$component.size	Integer	Size of the selection. May be null (see $component.hassize)

Table 4.48. Component Properties for SelectComponent

Name	Type	Description
singlevalue	String	Selected value as a single String (not as collection)
options	Node	list of static options
nameattribute	String	If the options are set as a collection, use this attribute to read the name.
valueattribute	String	If the options are set as a collection, use this attribute to read the value.

Table 4.49. Error Keys for SelectComponent

Key	Description
Error required field must not be empty	No value selected in a mandatory select component.

SelectComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/SelectComponent.xml`

4.2.2.7.2.10. TextAreaComponent

Multiline textfield with error checking by regexes.

Example 4.22. Syntax TextAreaComponent

```
<textareacomponent>
        <syntax></syntax>
        <syntax-description></syntax-description>
```

```
</textareacomponent>
```

Table 4.50. Settings TextAreaComponent

Name	Type	Default	Description
syntax	String	NULL	Regular expression the value is matched against. Creates an error in case of mismatch.
syntax-description	String	NULL	Readable description of the syntax. Will be displayed as error message.

Table 4.51. Template Variables for TextAreaComponent

Name	Type	Description
$component.value	Text	Current value.
$component.name	String	Internal name of the component, used as html property.
$component.length	Integer	Length of the textfield.

Table 4.52. Error Keys for TextAreaComponent

Key	Description
Error no empty text allowed	No text entered into a mandatory text field.

TextAreaComponent Example
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/TextAreaComponent.xml`

4.2.2.7.2.11. TextComponent

Singleline textfield with error checking by regexes.

Example 4.23. Syntax TextComponent

```
<textcomponent>
        <syntax></syntax>
        <syntax-description></syntax-description>
</textcomponent>
```

Table 4.53. Settings TextComponent

Name	Type	Default	Description
syntax	String	NULL	Regular expression the value is matched against. Creates an error in case of mis-

Name	Type	Default	Description
			match. This can also be the key of entries in the dictionaries, to support language specific syntax checking.
syntax-description	String	NULL	Readable description of the syntax. Will be displayed as error message.

Table 4.54. Template Variables for TextComponent

Name	Type	Description
$component.value	String	Current value.
$component.name	String	Internal name of the component, used as html property.

Table 4.55. Error Keys for TextComponent

Key	Description
Error no empty text allowed	No text entered into a mandatory text field.

TextComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/TextComponent.xml`

4.2.2.7.3. Display Components

4.2.2.7.3.1. DatasourceListComponent

Lists Resolvables Objects in ViewPlugin according to Objectstructure, with sorting, formatting and quicksearch.

Note

Altough this component can be used with every datasource, you should make sure that all requirements for the used datasource are provided.

For example “Datasources of type `ldap`” (Section 3.8.6.3.2, p.55) require that a rule is given. Otherwise it will not display any results.

Example 4.24. Syntax DatasourceListComponent

```
<datasourcelistcomponent>
  <columns>
    <column id="">
      <visible>[<prule></prule>]</visible>
      <label></label>
      <property></property>
      <foreignproperty></foreignproperty>
      <sortable></sortable>
      <actions>...</actions>
      <properties>
```

```
        <property id="...">...</property>
        ...
      </properties>
    </column>
    ...
  </columns>
  <sortorder></sortorder>
  <sortby></sortby>
  <sortable></sortable>
  <quicksearch></quicksearch>
  <rule></rule>
  <filterrule></filterrule>
  <filterlabel></filterlabel>
  <paging>
    <size></size>
    <current></current>
    <customsizes></customsizes>
  </paging>
  <datasource></datasource>
  <idattribute></idattribute>
  <stickyselection>[<prule></prule>]</stickyselection>
</datasourcelistcomponent>
```

Table 4.56. Settings DatasourceListComponent

Name	Type	Default	Description
columns	List	NULL	All columns to be listed
column in columns	Object	NULL	A single column and its properties
column.id	String	(required)	Id of the column. For columns showing object attributes this must be the attribute's id.
column.visible	Boolean or prule	TRUE	Whether this column is visible or not.
column.label	String	(required)	Label of the column
column.property	String	id	Object property to show in this column
column.foreignproperty	String	NULL	When the show property is another Resolvable object, the foreignproperty is the attribute of the Resolvable to be shown in the column (otherwise, the string representation of the Resolvable would be shown). This may also be a property path, if the foreignproperty of the linked object is again an object link (thus it is e.g. possible to show the name of the manager of the company of a listed person).
column.actions	Node	NULL	Configured pluggable actions for the column. The actions will resolve the clicked object (row) as `data.object.*` in their contexts.Section 4.2.2.8, "Pluggable Actions"
column.properties	Node	NULL	Custom properties set for this column. Can be used to set properties that are interpreted in the template of the DatasourceListComponent.
property	Node	NULL	Single property setting.
property.id	String	(required)	Id of the property set.
column.sortable	Boolean	NULL	Whether this column is sortable or not.

Name	Type	Default	Description
			(Sortable means that the user may sort the list by this column). This setting overwrites the general setting ***sortable*** .
sortorder	String	ASC	sort order, ASC for ascending or DESC for descending
sortby	String	NULL	defines the name of the attribute by which the elements in the list are sorted
sortable	Boolean	false	Whether the list shall be sortable or not. This setting applies to all columns but can be overwritten with ***column.sortable*** .
quicksearch	Node	NULL	Node containing the quicksearch settings
quick-search.filterlabel	String	(required)	Filterlabel to be set when the list is filtered by a quicksearch
rule	String	NULL	Base rule that filters the objects shown in the list. (The current item is accessible through 'object.' in the rule)
filterrule	String	NULL	Additional filterrule that further limits the objects shown in the list (combined with AND to an eventually existing rule). The filterrule is changed by a quicksearch or by other search mechanisms (e.g. RuleSearchAction). (The current item is accessible through 'object.' in the rule)
filterlabel	String	NULL	Filterlabel shown when a filterrule is applied
paging	Node	NULL	Paging settings. This column is optional. If no paging settings are provided no paging is applied.
paging.size	Integer	(required)	Number of objects shown on a single page. When this is set to 0, no paging is used.
paging.current	Integer	1	Initial value for the currently shown page.
paging.customsizes	List of Integers	NULL	Space delimited list of Custom page sizes the user may select for the list. 0 would be shown as "all" in the pagesize selectbox.
datasource	String	NULL	Id of the datasource to be used. When left empty, the portlet's default datasource is used (when configured). Set the datasource to `false` when the component shall not fetch the objects itself, but shall be filled by modification of the property ***items*** or by filling objects into a view using the Object2FormAction.
idattribute	String	contentid	Attribute of the listed objects that identifies the objects uniquely. It is necessary to set this correctly so that list items can be identified.
stickyselection	Boolean or Prule	FALSE	Whether the selection of rows in the list will remain longer than until the next click. With ***stickyselection*** set to

Name	Type	Default	Description
			FALSE, a selection made by the user will automatically be removed when the user clicks on a button or link. This is the default behaviour. If you need the selection for more than the next click, set this to TRUE. The selection can be reset by setting the component property *view.components.[datasourcelistcomponent].selection* = "".

Note

When a column is defined to represent binary data (e.g. the contents of a file), the rendered data is not the binary data itself, but is a link that can be used to download the binary data. Additionally, the template of the DatasourceListComponent has to take care about this fact and generate the html link for such columns!

Another solution for providing download links in a DatasourceListComponent would be to use a DownloadAction in combination with a DownloadComponent. See Section 4.2.2.7.3.3, “DownloadComponent” and Section 4.2.2.8.10, “DownloadAction” for details.

Table 4.57. Template Variables for DatasourceListComponent

Name	Type	Description
$component.quicksearch	String	The quicksearch content.
$component.quicksearch.enabled	Boolean	True when the list is filtered by a quicksearch.
$component.headers	List	All column headers
$header IN $component.headers	Object	A single column header
$header.label	String	Label of this column header
$header.sortlink	String	The URL to sort this list by this property ASC or, if already ASC, URL to order DESC
$header.sortlinkAsc	String	Returns an URL to sort this list by this property ascending.
$header.sortlinkDesc	String	Returns an URL to sort this list by this property descending.
$header.sortorder	String	Current sortorder of this property. Either "ASC" or "DESC"
$header.properties	Map	Map of the set custom properties for the column.
$component.filterobject	Object	Object for the currently used filter. May be used to display properties of the filter together with the filterlabel.
$component.filterlabel	String	Label of the sued filter.
$component.items	List	All items (rows) of the current page.
$item in $component.items	Object	Item object.
$item.props	List	All listed properties of the item (columns)
$item.content	Object	The resolvable object of this item. This variable is DEPRECATED, use $item.object instead.

Name	Type	Description
$item.content{.*}	Object	Properties of the object.
$item.object	Object	The resolvable object of this item.
$item.object{.*}	Object	Properties of the object.
$prop IN $item.props	Object	Property (column) of an item.
$prop.columnid	String	Id of the column.
$prop.value	String	Value of the itemproperty if the column shows a property of the item.
$prop.link	String	Link to trigger the action defined for the column. Is empty if no action was defined or if all defined actions are inactive due to their rules.
$prop.properties	Map	Map of the set custom properties for the column.
$prop.properties.[propertyid]	String	Value of the property `[propertyid]`.
$component.paging.size	Integer	Maximum number of items to be shown on one page.
$component.paging.current	Integer	Current page number, starting with 1.
$component.paging.pages	List	List of all available pages.
$page IN $component.paging.pages	Object	Page object.
$page.link	String	Link to activate this page.
$page.number	Integer	Page number.
$component.count	Integer	Total number of items within all pages.
$component.advancedsearchlink	String	Link to the advanced search (or null if no advancedsearch was configured)
$component.toggleselection	ButtonComponent	Button Container for Checkbox toggle actions, or null if not available.
$component.toggleselection.buttons.none	Button	Button to deselect all items.
$component.toggleselection.buttons.all	Button	Button to select all items.
$component.toggleselection.buttons.toggle	Button	Button to deselect all items if any is selected or to select all if there is currently no item selected.
$component.pagingsizeselect	SelectComponent	Component to select paging size, or null if not available.

Table 4.58. Component Properties for DatasourceListComponent

Name	Type	Description
items	Collection	Objects shown in the DatasourceListComponent. The returned items are dependent on set rule or filterrule and also on the currently set sorting, but not on the paging or the attributes shown in the list (the returned objects will not have any attributes prefilled, see below on more useful information about using this property). When the DatasourceListComponent has a datasource configured, this property is not changeable (since the DatasourceListCom-

Name	Type	Description
		ponent fetches the items itself). When the DatasourceListComponent is used to display data stored in a multivalue attribute, this property can be modified.
selection	Collection	Currently selected objects. Remove the selection by setting this property to `""`
filterrule	String	Rule that filters the shown items in the list. Can be set/unset to implement search functionality in the list.
pagingsize	Integer	Allows setting and retrieving of the paging size (Items displayed per page).
currentpage	Integer	Will switch paging to the page number provided. Settings lower than 1 will result in displaying the first page, while settings which are higher than the actual number of pages will lead to the last page.
sortby	String	Name of the sorted column.
sortorder	String	Sort order (`asc` for ascending, `desc` for descending).
count	Integer	Total number of entries in the list (independent of paging). This property is readonly.

Note

When items are fetched from a DatasourceListComponent by reading the property "*`items`*" , it has to be taken into consideration that really all objects are fetched from the configured datasource that match the currently set rule and/or filterrule regardless of their total number (paging of the component is not used here). For very large datasets this will consume both cpu and memory ressources on the server and could even lead to OutOfMemoryExceptions. Also it has to be noted that the fetched objects will have no attributes prefilled. Accessing attributes for many (or all) of the objects without having them prefilled first could possibly create a high server load by accessing the underlying storage (database) for every single attribute.

DatasourceListComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/DatasourceListComponent.xml`

4.2.2.7.3.2. DatasourceTreeComponent

Component used to display object relations in a tree. The objects are selected by a rootrule (for filtering the objects shown in the first hierarchy level) and noderelations (attributes that link to other objects).

Example 4.25. Syntax DatasourceTreeComponent

```
<datasourcetreecomponent>
  <datasource></datasource>
  <idattribute></idattribute>
  <rootrule></rootrule>
  <noderule></noderule>
  <disableflapping></disableflapping>
  <hideclosednodes></hideclosednodes>
  <maxlevel></maxlevel>
```

```
  <nodeactions defaultaction="">
    <nodeaction id="">
      <label></label>
      <visible></visible>
      <enabled></enabled>
      <actions>
        ...
      </actions>
      <properties>
        <property id="...">...</property>
          ...
      </properties>
    </nodeaction>
    ...
  </nodeactions>
  <sortby></sortby>
  <sortorder></sortorder>
  <paging>
    <size></size>
    <current></current>
    <customsizes></customsizes>
  </paging>
  <relations>
    <relation attribute="" sortby="" sortorder="" reverseattribute="">
      <rule></rule>
    </relation>
    ...
  </relations>
  <idattribute></idattribute>
  <maxlevel></maxlevel>
  <hideclosednodes>[<prule></prule>]</hideclosednodes>
  <prefillattributes></prefillattributes>
</datasourcetreecomponent>
```

Table 4.59. Settings DatasourceTreeComponent

Name	Type	Default	Description
datasource	String	default portlet datasource	Id of the datasource to use. When not configured, the default datasource of the portlet is used.
idattribute	String	contentid	Name of the attribute of the resolvables loaded from the datasource, that uniquely identify the objects. This needs to be set correctly, otherwise the tree will not work like expected.
rootrule	String	true	Rule to restrict the objects shown in the first hierarchy level of the tree. Resolves portal., views., view. and form. See Property Path.
noderule	String	true	Rule to further restrict all objects shown in the tree. Resolves portal., views., view. and form. See Property Path.
disableflapping	Boolean or prule	false	Whether flapping is enabled or disabled. If true then $child.flappable will be false and $child.flappedOpen will be true for all children.
nodeactions	Node	NULL	List of nodeactions defined for the tree.
nodeactions.defaultaction	String	NULL	The id of the default action (will be supported by a direct link)

Name	Type	Default	Description
nodeaction.id	String	(required)	Id of the nodeaction. Must be unique in the scope of the component.
nodeaction.label	String	NULL	Label of the nodeaction.
nodeaction.visible	Boolean or prule	true	Whether the action is visible. The rule may contain "object" as base object to match the currently shown treeobject.
nodeaction.enabled	Boolean or prule	true	Whether the action is enabled. The rule may contain "object" as base object to match the currently shown treeobject.
nodeaction.actions	Node	NULL	Actions container holding pluggable action definitions. The treeobject will be available in the actions' contexts as "data.object".
nodeaction.properties	Node	NULL	Custom properties of the nodeaction.
property	Node	NULL	A single custom property.
property.id	String	(required)	Id of the custom property.
sortby	String	NULL	Name of the attribute by which the root objects are sorted.
sortorder	String	ASC	One of ASC or DESC. Sort order for the root objects.
paging	Node	NULL	Paging settings
paging.size	Integer	(required)	Number of objects shown on a single page. When this is set to 0, no paging is used.
paging.current	Integer	1	Initial value for the currently shown page.
paging.customsizes	List of Integers	NULL	Space delimited list of Custom page sizes the user may select for the list. 0 would be shown as "all" in the pagesize selectbox.
relations	Node	NULL	list of relations to build the tree.
relation.attribute	String	(required)	Name of the attribute defining the relation. The attribute should link to other objects.
relation.sortby	String	NULL	name of the attribute by which objects linked to by this relation are sorted. NULL means no sorting.
relation.sortorder	String	ASC	One of ASC or DESC. Sort order (ascending or descending)
relation.rule	String	true	Determines whether the children of an object, related with this relation, should be shown. The mother object is matched against the rule.
relation.reverseattribute	String	NULL	Name of the attribute that is the reverse of the relation ***`attribute`***. This attribute must lead from any node to its parent node. When this property is set and a node is opened via setting the property ***`openpath`***, all parent nodes of the opened node are also opened to ensure that the opened node is really visible.
idattribute	String	contentid	Name of the attribute that uniquely identify the Resolvables shown in the

Name	Type	Default	Description
			tree. Note that is is necessary to set this correctly, otherwise the component will fail to work. The default `contentid` works with datasources of type `contentrepository` .
maxlevel	Integer	-1	Maximum number of levels to be shown in the tree. Set this to `0` to show only the root nodes or `-1` for no level limit.
hideclosednodes	Boolean or prule	false	Whether closed nodes shall be hidden from view. When this is true, only the open nodes (including all parent nodes) are rendered. Otherwise also closed nodes will be rendered.
prefillattributes	String	NULL	A comma-separated list of attributes that should be prefilled when loading the objects from the database (increases performance).

Table 4.60. Template Variables for DatasourceTreeComponent

Name	Type	Description
$component.tree	Object	Contains the root node of the Tree.
$component.tree.children	List	Children of the root node.
$child IN $tree.children	Object	Contains one tree node.
$child.children	List	Contains the list of children of the tree node
$child.nodeCount	Integer	Calculates number of all direct children (non-recursive)
$child.allNodeCount	Integer	Calculates number of all subnodes (recursive)
$child.getAllNodeCount("rule")	Integer	Calculates number of all subnodes applying to the defined rule (recursive) - e.g.: "object.obj_type == 2"
$child.visible	Boolean	Whether the child is visible or not.
$child.flappable	Boolean	Whether the node is flappable (has children).
$child.flappedOpen	Boolean	Whether the node is flapped open.
$child.flapLinkOpen	String	URL to open this node.
$child.flapLinkClose	String	URL to close this node.
$child.flapLinkAllOpen	String	URL to open this node and all nodes below.
$child.flapLinkClose	String	URL to close this node and all nodes below.
$child.userObject	Object	The Data Object from the datasource. This variable is DEPRECATED, use $child.object instead.
$child.userObject(.*)	Object	Properties of the object.
$child.level	Integer	Hierarchy level of the current node. Will be `0` for root nodes.
$child.object	Object	The resolvable object of this item.
$child.object{.*}	Object	Properties of the object.
$child.defaultActionLink	String	URL to activate the default action on this node, when such an action exists and is enabled.
$child.first	Boolean	True when the child is the first child of its parent

Name	Type	Description
$child.last	Boolean	True when the child is the last child of its parent.
$child.number	Integer	Number of the child within all children of its parent, starting with 1.
$action in $child.nodeActions	Object	A configured nodeAction.
$action.id	String	The ID of the action if it was specified.
$action.enabled	Boolean	Whether the action is enabled.
$action.visible	Boolean	Whether the action is visible.
$action.label	String	Label of the action.
$action.url	String	URL to activate this action.
$action.properties	Map	Custom properties of the nodeaction.
$action.properties.[propertyid]	String	Value of the custom property `[propertyid]`.

Table 4.61. Component Properties for DatasourceTreeComponent

Name	Type	Description
open	Collection of Strings	Collection of the id's of the currently opened nodes. This property can also be modified to open/close nodes.
openpath	Collection of Strings	This "property" is not readable but can be set to open a specific node (or some specific nodes) with all their respective parent nodes. Setting this property will not change other open nodes. For this property to work correctly, it is necessary to set *`relation.reverseattribute`* .
reload	Boolean	This "property" is not readable, you can use it to force the component to reload it's data. (E.g. when something referenced in the rule was modified which is unrelated to the component itself.)
sortby	String	Allows to set the 'sortby' setting of this component. Expects the name of the attribute.
sortorder	String	Allows to set the 'sortorder' setting of this component. This is either "asc" or "desc"

DatasourceTreeComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/DatasourceTreeComponent.xml`

4.2.2.7.3.3. DownloadComponent

The download component can be used to provide binary data (files) for downloading to the user client. When placed inside a view, this component normally does not render any output (and therefore does not need a template). When a DownloadAction is processed that refers to this component, it will provide the binary data as download to the client. See Section 4.2.2.8.10, “DownloadAction” for details.

Example 4.26. Syntax DownloadComponent

```
<downloadcomponent>
</downloadcomponent>
```

Table 4.62. Settings DownloadComponent

Name	Type	Default	Description
No Settings.			

Table 4.63. Component Properties for DownloadComponent

Name	Type	Description
content	various	The content the component shall provide as download. May be binary data, a string or stream
filename	String	Filename of the download.
disposition	[attachment]|[inline]	"Type" of the download. "Attachment" should be used for normal downloads (browser will ask whether to save or open the file). When set to "inline", the browser should try to open the file automatically.
contenttype	String	Mimetype of the content to download.

DownloadComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/DownloadUploadComponents.xml`

4.2.2.7.3.4. FeedbackComponent

Component used to display one-time messages (message will automatically be reset when rendered once). The messages are defined or set by actions.

Note

When using a FeedbackComponent to display one-time messages, you should avoid to render the same view more than once on the same portal page, because the feedback would only be visible in one of the view occurances (the one which is rendered first) and this behaviour might be unexpected.

Example 4.27. Syntax FeedbackComponent

```
<feedbackcomponent>
</feedbackcomponent>
```

Table 4.64. Settings FeedbackComponent

Name	Type	Default	Description
No Settings.			

Table 4.65. Template Variables for FeedbackComponent

Name	Type	Description
$component.text	String	Feedback message.

FeedbackComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/FeedbackComponent.xml`

4.2.2.7.3.5. InformationComponent

Component used to display data coming from other components in the same form.

Example 4.28. Syntax InformationComponent

```
<informationcomponent>
        <reference></reference>
</informationcomponent>
```

Table 4.66. Settings InformationComponent

Name	Type	Default	Description
reference	String	(required)	Id of the referenced component. When the value of the component is an object, the reference could also contain a specific attribute of this object.

Table 4.67. Template Variables for InformationComponent

Name	Type	Description
$component.value	Object	Value of the referenced component.

4.2.2.7.3.6. LabelComponent

Example 4.29. Syntax LabelComponent

```
<labelcomponent>
        <text></text>
</labelcomponent>
```

Table 4.68. Settings LabelComponent

Name	Type	Default	Description
text	String	NULL	Text to be shown in the label component. This should be an i18n key to an entry in the dictionaries.

Table 4.69. Template Variables for LabelComponent

Name	Type	Description
$component.text	String	The text to output.

Table 4.70. Component Properties for LabelComponent

Name	Type	Description
text	String	The text to output.

LabelComponent Example
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/LabelComponent.xml`

4.2.2.7.3.7. ListComponent

A container component for other components. Used to list components.

Example 4.30. Syntax ListComponent

```
<listcomponent>
        <content>...</content>
</listcomponent>
```

Table 4.71. Settings ListComponent

Name	Type	Default	Description
content	Node	NULL	Node holding the subcomponents.

Table 4.72. Template Variables for ListComponent

Name	Type	Description
$component.items	List	All list items.
$item IN $component.items	Collection	One of the items in $component.items
$item	String	The content of the subitem itself.

Name	Type	Description
$item.*	Object	Main object of subcomponent for component properties.

ListComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/ListComponent.xml`

4.2.2.7.3.8. NestedFormComponent

Component used to implement a nested form inside another form. A NestedFormComponent can hold any combination of components as children, even other NestedFormComponents (so it is possible to generate a form in form in form in ... All inner components are prefixed with the id of the nestedformcomponent (prefix and component id separated by underscore _). Nested forms can be treated analogously to views, in the sense that the Form2ObjectAction, Object2FormAction and GeneralViewAction may refer to NestedFormComponents instead of a whole view.

See Section 4.2.2.8.13, “Form2ObjectAction” , Section 4.2.2.8.17, “Object2FormAction” and Section 4.2.2.8.14, “GeneralViewAction” for details.

Example 4.31. Syntax NestedFormComponent

```
<nestedformcomponent>
        <content>...</content>
        <exclusive></exclusive>
        <visibility></visibility>
        <openbuttonlabel></openbuttonlabel>
        <closebuttonlabel></closebuttonlabel>
        <closeon></closeon>
</nestedformcomponent>
```

Table 4.73. Settings NestedFormComponent

Name	Type	Default	Description
content	Node	NULL	Container for inner components of the nested form
exclusive	Boolean or Rule	false	whether the inner form shall be exclusive (when the visibility mode is "ondemand") when an exclusive nested form is opened, all surrounding components are disabled.
visibility	String	always	visibility mode of the form in form: "always" to have the inner form always open, "ondemand" to have the inner form being opened and closed using default buttons (which will be autogenerated), "custom" when the opening and closing of the inner form is done through setting the *open* property or via PluggableActions.
openbuttonlabel	String	Open	label of the open button (when the visibility mode is "ondemand")
closebuttonlabel	String	Close	label of the close button (when the visib-

Name	Type	Default	Description
			ility mode is "ondemand")
closeon	List of Strings	NULL	space delimited list of buttoncomponent id's; defines the inner buttoncomponents, which will automatically close the form, when all actions defined for the button were successfully processed.

Table 4.74. Template Variables for NestedFormComponent

Name	Type	Description
$component.items	List	List of inner components
$component.buttons	Component	Button component holding the default buttons (for opening/closing the inner form when the visibility mode is "ondemand")

Table 4.75. Component Properties for NestedFormComponent

Name	Type	Description
open	Boolean	Whether the form is open or closed. Can be used to open/close the inner form, when ***visibility*** is set to `custom`.

NestedFormComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/NestedFormComponent.xml`

4.2.2.7.3.9. TabComponent

Example 4.32. Syntax TabComponent

```
<tabcomponent>
        <content>
                <tab>
                        <label></label>
                        <visible></visible>
                        <enabled></enabled>
                        <optional></optional>
                        <properties></properties>
                        <content>...</content>
                </tab>
                ...
        </content>
        <activetab></activetab>
</tabcomponent>
```

Table 4.76. Settings TabComponent

Name	Type	Default	Description
content	Node	NULL	Node holding the tabs
content.tab	Node	NULL	Node holding a tab. Many tabs may be defined.
tab.label	String	required	Readable label. The label should be the i18n key of the real labels in the dictionaries for multilanguage support.
tab.visible	Boolean or prule	true	Whether this tab is visible or not.
tab.enabled	Boolean or prule	true	Whether this tab is enabled or not. Disabled tabs can not be activated or clicked.
activetab	String	first tab defined	Id the the initial active tab.

Table 4.77. Template Variables for TabComponent

Name	Type	Description
$component.items	List	All tabs
$item in $component.items	Object	One of the tabs in $component.items
$item.visible	Boolean	Visibility of a tab.
$item.enabled	Boolean	False, if the tab is not available to the user.
$item.component	Object	General variables of the component of the tab. Cannot render the tab-component
$item.active	Boolean	True for the currently selected tab by the user.
$item.id	String	The id of a tab.
$item.name	String	The name of a tab.
$item.taburl	String	The url to activate a tab.
$component.activetab	String	Id of the active tab.
$component.content	String	The content of the active tab.

TabComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/TabComponent.xml`

4.2.2.7.3.10. VersionInformationComponent

Component used to display information about the version that is currently displayed or edited in a view. This component should be used together with the VersioningComponent (see Section 4.2.2.7.3.11, “VersioningComponent”).

Example 4.33. Syntax VersionInformationComponent

```
<versioninformationcomponent>
</versioninformationcomponent>
```

Table 4.78. Settings VersionInformationComponent

Name	Type	Default	Description
No Settings.			

Table 4.79. Template Variables for VersionInformationComponent

Name	Type	Description
$component.versiontimestamp	Integer	Timestamp of the currently displayed or edited version. Will be set to -1 for the current version.
$component.current	Boolean	Whether the version is the current version.
$component.past	Boolean	Whether the version is the past version.
$component.future	Boolean	Whether the version is the future version.

Table 4.80. Component Properties for VersionInformationComponent

Name	Type	Description
versiontimestamp	Integer	Timestamp of the currently displayed or edited version. Will be set to -1 for the current version.
current	Boolean	Whether the version is the current version.
past	Boolean	Whether the version is the past version.
future	Boolean	Whether the version is the future version.

VersionInformationComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/VersionComponents.xml`

4.2.2.7.3.11. VersioningComponent

Component used to display and manage versions of the contentobject edited in the current form. This component can only be used when the datasource supports versioning. The VersioningComponent shows a list of all existing versions of an object. Since this list may be long and space consuming, it is recommended to place this component in an own tab of the TabComponent. To ensure that a user is aware of which version of an object is currently shown in a view, it is recommended to use a VersionInformationComponent (not as part of the TabComponent, but always visible). See Section 4.2.2.7.3.11, “VersioningComponent” and Section 4.2.2.7.3.9, “TabComponent” for detailed information.

Example 4.34. Syntax VersioningComponent

```
<versioningcomponent>
        <currentversionlabel></currentversionlabel>
        <versiondatelabel></versiondatelabel>
        <createversionlabel></createversionlabel>
        <viewpastobjects></viewpastobjects>
        <viewfutureobjects></viewfutureobjects>
        <editpastobjects></editpastobjects>
        <editfutureobjects></editfutureobjects>
</versioningcomponent>
```

Table 4.81. Settings VersioningComponent

Name	Type	Default	Description
currentversionlabel	String	current version	Label for the button to switch to the current version of an object
versiondatelabel	String	new version date	Label for the DateComponent to enter the date for a new version
createversionlabel	String	create version	Label for the button to create a new version at the given date
viewpastobjects	Boolean	false	Whether viewing of past versions is allowed or not
viewfutureobjects	Boolean	false	Whether viewing of future versions is allowed or not
editpastobjects	Boolean	false	Whether editing of past versions is allowed or not
editfutureobjects	Boolean	false	Whether editing of future versions is allowed or not

Table 4.82. Template Variables for VersioningComponent

Name	Type	Description
$component.versions	List	List of versions of the edited object.
$version IN $component.versions	Object	One version of the edited object.
$version.date	Date	Date of the version (when version was created.)
$version.diffcount	Integer	# of changed records in this version.
$version.restoreurl	String	URL to restore the version (may be null if restoring is not allowed.)
$version.displayurl	String	URL to display the version.
$version.new	Boolean	True if this version is new.
$version.shownVersion	Boolean	True if this version is currently shown.
$comopnent.date	DateComponent	Date component for creation of new versions. (can be null)
$component.create	ButtonComponent	Button component used to create new versions. (can be null)
$component.current	ButtonComponent	Button component used to switch to the current version.

VersioningComponent Example:
The example illustrates the usage of this component. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ComponentExamples/VersionComponents.xml`

4.2.2.8. Pluggable Actions

4.2.2.8.1. Introduction

Pluggable Actions are small pieces of Java-Code that can be plugged into views. Pluggable actions can be triggered by button components (the code is executed when the button is clicked by the user) or

some other components.

Some pluggable actions can be invoked "manually" by using the PluggableActionInvoker. See the CsvImportExample in the SDK Guide for an example.

4.2.2.8.2. Implementation

Example 4.35. Structure of pluggable action definitions.

```
<actions>
  <action id="" class="" feedback="">
    <prule>...</prule>
      <parameters>
      <parameter id="" mapped="" required="">[static value]</parameter>
        ...
      </parameters>
      <onsuccess message="..." proceedsequence="...">
        <actions/>
      </onsuccess>
      <onfailure message="..." proceedsequence="...">
        <actions/>
      </onfailure>
    </action>
    ...
</actions>
```

Table 4.83. General Action Attributes

Name	Type	Default	Description
actions	Container	NULL	Container of defined actions. May hold one or more actions (an action sequence), which are invoked in the given order. Normally all actions in the sequence are processed, but this can be controlled by setting proceedsequence to FALSE in any of the actions.
action	Object	(required)	Definition of one particular action
action.id	String	NULL	Id of the action. The id of actions is used for referencing the generated response objects of particular actions (e.g. in subsequent pluggable actions). Note that the id of the action does not necessarily need to be unique within the actions context. When multiple actions with the same id are invoked, the responses will be merged. When multiple actions with the same id generate response values with the same id, only the response value from the last action will be available.
action.class	String	(required)	Java-class of the implementation of the action. For Pluggable Actions that are part of Portal.Node, the package name com.gentics.portalnode.genericmodules.object.actions may be omitted. Custom pluggable actions have to be defined

Name	Type	Default	Description
			with the full class name.
action.feedback	String	NULL	Id of the feedback component, the action may write messages to. When configured, there must be a feedback component in the view that holds this action.
action.prule	String	NULL	Rule to decide whether the pluggable action has to be invoked or not. When no rule is configured the action is always processed. If this rule does not match onsuccess/onfailure and the *.proceedsequence parameters are ignored and the next action in the sequence will always be executed.
action.parameters	Container	NULL	Container for action parameters. Every specific implementation of a pluggable action will required or support a specific set of parameters that can or have to be defined.
action.parameters.parameter	Object	NULL	Particular parameter setting for the pluggable action. Parameters may be mapped from portal or view properties or may be defined static.
action.parameters.parameter.id	String	(required)	Id (key) of the action-parameter
action.parameters.parameter.mapped	String	NULL	Path to the value of the defined parameter. The path must be resolvable in the context of this pluggable action. This may also contain results of previous actions (via .actions.[actionid].*). This may also be an expression with functions, operations, literals and resolvables. See “ExpressionParser” (Section 4.10, p.182) for further details.
action.parameters.parameter.required	Boolean	FALSE	Whether the parameter is required for the action (value may not be empty)
action.onsuccess	Object	NULL	Definition of what is to be done in case the action succeeds
action.onsuccess.message	String	NULL	Message to be writted into the feedback component when the action succeeds
action.onsuccess.proceedsequence	Boolean	TRUE	Whether actions in this sequence shall be proceeded when this action succeeds. When set to FALSE and this action succeeds, no other actions of this sequence are processed. This does NOT affect an eventually configured container in this onsuccess part.
action.onsuccess.actions	Container	NULL	Action Container holding an action sequence that shall be processed when this action succeeds.
action.onfailure	Object	NULL	Definition of what is to be done in case the action fails.
action.onfailure.message	String	NULL	Message to be writted into the feedback component when the action fails.
action.onfailure.	Boolean	TRUE	Whether actions in this sequence shall

Name	Type	Default	Description
proceedsequence			be proceeded when this action fails. When set to FALSE an this action fails, no other actions of this sequence are processed.
action.onfailure. actions	Container	NULL	Action Container holding an action sequence that shall be processed when this action fails. This does NOT affect an eventually configured container in this onfailure part.

4.2.2.8.2.1. ActionContext

Each action has an ActionContext which is used in the following elements, and some specific actions:

- action.prule
- action.parameters.parameter.mapped

The ActionContext provides the following Portal Property Paths, see Section 4.8, “Portal Property Paths” for details.

- views.*
- view.*
- portal.*
- actions.*
- data.* base object of context specific additional data, mostly provided by the enclosing component.
- javax.request.*

4.2.2.8.3. BinaryCallableActionResponseAction

Returns binary data from a Callable Action.

Table 4.84. Parameters: BinaryCallableActionResponseAction

Name	Type	Default	Description
data	ByteArray or InputStream		The data to return in the response
mimetype	String	binary/octet-stream	Mimetype of the response.

Table 4.85. Response Objects: BinaryCallableActionResponseAction

Name	Type	Description
response	Binary	The binary data that will be returned to the client.

Table 4.86. Action: BinaryCallableActionResponseAction - Return Values

TRUE	When no error occurred.
FALSE	If an error occured.

4.2.2.8.4. BinaryToTextAction

This action transforms a binary input stream, like the filestream from a fileuploadcomponent, into a reader, using a given character encoding.

If the input data is not set, the output content parameter will also not be set. If the input is already text or a reader, the content is not modified. If an error occurs, an error message is set as feedback message.

Table 4.87. Parameters: BinaryToTextAction

Name	Type	Default	Description
data	ByteArray or InputStream		The data to transform
encoding	String	ISO-8859-1	The encoding used to transform the binary content into text. The is equal to the java character encoding names, like UTF-8 or UTF-16.

Table 4.88. Response Objects: BinaryToTextAction

Name	Type	Description
content	Reader	The converted data as text reader, or not set if the input data is not set.

Table 4.89. Action: BinaryToTextAction - Return Values

TRUE	on success.
FALSE	on error.

4.2.2.8.5. CheckErrorsAction

Allows you to start errorchecking on a form starting from the component provided via the component-parameter. As errorchecking won't involve nested forms by default you can use the CheckErrorsAction to accomplish this task. The action will return true if everything was fine or false if errors where encountered.

Table 4.90. Parameters: CheckErrorsAction

Name	Type	Default	Description
component	String	(required)	This is the id of the component where errorchecking will be started. All subsequent components will be checked for errors. Note that ids of components which reside inside of nested form components have to be prefixed with the nested form component's id and an underscore. It is possible to use multiple parameters with id `component` to do

Name	Type	Default	Description
			errorchecking starting with multiple components.
errors2feedback	Boolean	false	When this is set to `true`, eventually found errors will not only be set for the components, but also set as feedback into the feedback component of this action (if any set).
errors2feedbackseparator	String	(emptry String)	This parameter defines the separator between multiple component errors, when the errors are set as feedback (if the parameter *`errors2feedback`* is set to `true`).

No Response Objects.

Table 4.91. Action: CheckErrorsAction - Return Values

TRUE	No errors encountered while checking components
FALSE	Component errors encountered or configuration error

4.2.2.8.6. CollectObjectsByRelationAction

Given an object this action follows the relationattribute of this object until it reaches all the leaves. If it did so it will create a collection of all the objects and return this collection.

Table 4.92. Parameters: CollectObjectsByRelationAction

Name	Type	Default	Description
baseobject	Object	(required)	Starting Object.
relationattribute	String	(required)	The attribute that describes the relation. This parameter may be given multiple times if more than one relation has to be followed.
includebase	Boolean	True	If base object needs to be included in the collection as well.
failifempty	Boolean	False	Wheter the action fails when the collection is empty.
rule	String	NULL	An additional rule to restrict the returned objects.

Table 4.93. Response Objects: CollectObjectsByRelationAction

Name	Type	Description
objects	List of Objects	Collected objects as Collection (may be an empty Collection, but never null)
lastobject	Object	The last object that was collected.

Table 4.94. Action: CollectObjectsByRelationAction - Return Values

TRUE	on success.
FALSE	on error.

4.2.2.8.7. CreateResolvablesAction

The CreateResolvablesAction can be used to create Resolvable objects from a given XML source. This XML source has to conform to a specific XML schema. If a WriteableDatasource is given it can be used to create and load objects from this datasource. (E.g. With a CNDatasource given an attribute 'contentid' would load this resolvable from the datasource.)

This can be very handy in combination with a “XSLTRenderAction” (Section 4.2.2.8.31, p.162) so you can transform e.g. a RSS feed into resolvables which are then displayed in a “DatasourceListComponent” (Section 4.2.2.7.3.1, p.120).

Example 4.36. Example XML input.

```
<?xml version="1.0" encoding="UTF-8"?>
<objects xmlns="http://www.gentics.com/create-resolvables">
  <object>
    <attribute name="title">My Title</attribute>
    <attribute name="description">Some description</attribute>
  </object>
  <object>
    ...
  </object>
</objects>
```

Table 4.95. Parameters: CreateResolvablesAction

Name	Type	Default	Description
xml	String	(required)	XML source describing the Resolvable objects which should be created.
datasource	String	NULL	This has to be the name for a WriteableDatasource which is used to create the resolvable objects.

Table 4.96. Response Objects: CreateResolvablesAction

Name	Type	Description
result	Collection	The created Resolvables within a Collection. (Resolvables are either created by the given datasource, or MapResolvable's)

Table 4.97. Action: CreateResolvablesAction - Return Values

TRUE	If resolvables could be created successfully
FALSE	Otherwise

4.2.2.8.8. CSVDataImportAction

This action reads data from a CSV (comma separated values) content and imports it into a contentrepository. A status message is returned as feedback message.

The first non-skipped line in the csv content is interpreted as header, which defines into which attributes the fields are imported. A new import always creates new objects and does not check for duplicate entries.

The import creates a log during the import. It also generates a list of all contentids which are created during the import, including newly created referenced objects. If a line contains invalid data, an error is created and the line is ignored. Even if an error occurs, successfully imported lines are NOT deleted.

4.2.2.8.8.1. CSV header format

Each column which should be imported must have a header in the following form:

```
<attributename>[:<foreignObjectNr>][.<referenceAttribute>][|<options>]
```

Each header starts with the attributename of the attribute of the object matching the field.

There are three types of attributes:

- Simple attributes like text or numbers do not require additional attributes.

  For binary attributes the option ***binary*** can be used. The field value is then interpreted as the filename of the file to import into the attribute. If the input parameter ***fileprefix*** is set, each filename is prefixed with the fileprefix.

  If the attribute is a date, the field must contain a timestamp in seconds since January 1, 1970 00:00:00 GMT.

- References require an additional, dot-separated reference attribute name. This is the attribute name of the referenced object type, by which the referenced object is searched. If the option ***create*** is given, a new referenced object is created, if it is not found.

  For example, using the following header

  ```
  company.name|create
  ```

  the import action will search for companies by name using the values of the column fields and store the reference in the attribute ***company***. If the company is not found, a new company by the given name is created.

- If the attribute is a foreign link, a reference attribute name and a foreign object number must be given. The object number can be any number and is used to group columns.

  For each distinct object number, a new foreign object is created, if any of the columns in a row contains a value. Each attribute of the foreign object can itself be a simple attribute or a reference.

  For example, the following header

  ```
  job:1.name; job:1.category.name; job:2.name; job:2.category.name
  ```

  will create up to two foreign objects per row, each with an attribute name and category, which is itself a reference to a category object which is searched by the field ***name***.

4.2.2.8.8.2. Skip line ranges

One or more ranges of lines can be specified which should be skipped when reading the content. There are three possible ranges:

- A single line number <nr>
- A range of lines with lower and upper bound <min>-<max>
- An upper limit, <max>+

Ranges can be separated by semicolons.

For example, the following line

```
1;3-5;50+
```

skips the first line, lines 3 to 5, and all lines after the 50th line, including line 50. Therefore, the second line is interpreted as header, and lines 6 to 49 are imported.

Table 4.98. Parameters: CSVDataImportAction

Name	Type	Default	Description
csv	String or Reader	(required)	The CSV content to be imported.
objecttype	Integer	(required)	The object type id of the main objects to import.
delimiter	String, 1 character	"	The text delimiter character.
separator	String, 1 character	;	The field separator character.
skiplines	String		A list of ranges of lines to skip.
fileprefix	String		A path which is used to prefix binary file names.
loglevel	String	info	The maximum log message level to include in the response. Possible values are 'error', 'warn' and 'info'.
datasource	String/ WriteableDatasource		The datasource to use to store objects, either the datasource itself or the id of the datasource. The datasource must be a WriteableDatasource on the contentrepository. If the parameter is not set, the default datasource for the current module is used.

Table 4.99. Response Objects: CSVDataImportAction

Name	Type	Description
createdids	Array of String	An array of all contentids as Strings which were created during the import, including reference objects.
logmessages	String	The detailed logmessages of the import as String.

Table 4.100. Action: CSVDataImportAction - Return Values

TRUE	On Success
FALSE	If any input parameter contains invalid data, or an error occured during the import. Modifications done by the import (like the creation of objects) are not reverted if an error occurs.

Example 4.37. Example import data

An example import csv file as well as a view using the pluggable action and a standalone commandline import example can be found in the sdk. For details see the CsvImportExample in the sdk guide.

4.2.2.8.9. DatasourceAction

The Datasource Action allows storage of objects to a WriteableDatasource and loading from a Datasource. It performs either "insert", "update", "save", "delete", "load" or "create". Where save is a method that will figure out if to insert or update itself, if possible.

Table 4.101. Parameters: DatasourceAction

Name	Type	Default	Description
operation	String	(required)	Operation to be performed. One of: • update Used to update the values of an existing Object. (If object does not exist, it will result in an error) • insert Inserts a new object and saves it. • save Saves the object. - If it does not yet exist, an object will be created, otherwise the existing one will be updated. • delete Deletes the given object. • load Loads objects. You have to supply either a rule or the primary key as an additional parameter to successfully load an object. See Section 4.13, “Datasource” for details. • create Creates a new object but does not immediately save it. • clearcache

Name	Type	Default	Description
			Clear caches for the given objects in the datasource. Objects can either be given in parameters "object" or in "contentid" for contentrepository datasources.
datasource	String	datasource of the portlet	Id of the datasource to use for the action.
object	Object	NULL	Object which has to be modified. Can also be a collection of objects. Required for operation "update", "insert", "save" and "delete".
rule	String	NULL	Rule to load objects, may be empty to load objects by primary key (`contentid` for datasources of type `ContentRepository`).
sortby	String	NULL	Name of the sorted attribute when loading objects by rule. When not set, the objects are not sorted.
sortorder	`asc` or `desc`	asc	Sortorder, when objects are loaded by rule and *`sortby`* is set as well.
prefill	String	NULL	Name of an attribute that will be prefilled for objects loaded by rule. Prefilling attributes increases the performance when accessing them later. This parameter can be used multiple times for prefilling more than one attribute. Please note that prefilling attributes will only increase performance when handling large amounts of data. For example: when using prefill to retrieve 10 of the latest news items, you may experience performance drawbacks. On the other hand fetching the whole load of news items from the last two years will be a lot faster.
channel	Integer (or List of Integers)	NULL	When the datasource is a Multichannelling ContentRepository (MCCR), this parameter will set the channel(s) to be used in the action. Using this parameter will not change the user session setting for the selected channels, which can be modified with the portal property path *`portal.datasources.[id].channel`*.
*	*	NULL	Any other request parameters are set to created objects as attributes.

Table 4.102. Response Objects: DatasourceAction

Name	Type	Description
objects	Collection	loaded/created objects always as collection (even if no or only one object is returned). Empty collection when no object is returned.

Name	Type	Description
object	Object	First (or only) loaded/created object as single Resolvable, NULL when no object is returned

Table 4.103. Action: DatasourceAction - Return Values

TRUE	if the action has been performed without errors.
FALSE	if an exception was thrown.

4.2.2.8.10. DownloadAction

The download action can be used to initiate a file download via a DownloadComponent. See Section 4.2.2.7.3.3, “DownloadComponent” for details.

Table 4.104. Parameters: DownloadAction

Name	Type	Default	Description
content	String, binary data or Stream	{required}	The content to download.
contenttype	String	NULL	The contenttype (mimetype) of the content to download.
component	String	{required}	Id of the DownloadComponent to use
disposition	"attachment"|"inline"	attachment	Type of the download.
filename	String	NULL	Filename to use for the download.

No Response Objects.

No Return values.

4.2.2.8.11. EchoAction

The EchoAction simply prints all its parameters into the logfiles. This can be used during development of pluggable actions for debugging purposes. The log messages are generated with the log-level `INFO`, but there is no need to reconfigure the logging since the log-level is temporarily set to `INFO` in the EchoAction which will enforce the output, no matter what loglevel is configured otherwise. The output consists of the name of the parameter, the String representation of its value and the java classname.

Table 4.105. Parameters: EchoAction

Name	Type	Default	Description
*	*	NULL	All parameters are just printed into the logfiles.

No Response Objects.

Table 4.106. Action: EchoAction - Return Values

TRUE	This Action always returns true
FALSE	This Action never returns false

4.2.2.8.12. Form2CNObjectAction

Note

This action is deprecated. The more general Form2ObjecAction (Section 4.2.2.8.13, “Form2ObjectAction”) should be used instead. Remember to correct the parameter from objecttype to obj_type when you upgrade to the new action.

This Action maps the content of a given form to a contentrepository object. the id's of the components must equal the objects attributenames. The form must have a contentid component (usually hidden) to recognize an existing contentobject. Otherwise a new ContentObject will be created.

Table 4.107. Parameters: Form2CNObjectAction

Name	Type	Default	Description
objecttype	Integer	(required)	the contentrepository objecttype used for creating the object
datasource	String	portlet's default datasource	the datasource where the given object-type is located

Table 4.108. Response Objects: Form2CNObjectAction

Name	Type	Description
object	Object	a new contentrepository object created based on the form content.

Table 4.109. Action: Form2CNObjectAction - Return Values

TRUE	on succesfull creation of ContentObject
FALSE	on failure (e.g. datasource unavailable, unknown objecttype)

4.2.2.8.13. Form2ObjectAction

Action to transform the content of a view into an object for further processing like saving.

It takes a form and maps it to an object according to the given set of instructions. The created object will either be instanciated from the given `objectclass` or will be created from the given `datasource`, which must be a WriteableDatasource. One of them is required.

Depending on the type of created object, some additional parameters might be needed. When the object shall be created for a datasource of type `contentrepository`, the `obj_type` must be given to specify the object type, otherwise the creation would fail. Additionally if the object is created for an existing object in the repository, the `contentid` has to be provided, which is typically be done with a hidden component in the form.

Table 4.110. Parameters: Form2ObjectAction

Name	Type	Default	Description
objectclass	String	NULL	Class the new object is instanciated from. The class has to implement Changeable and has to have a default

Name	Type	Default	Description
			Constructor.
datasource	String	Datasource of the portlet.	Datasource for which the object shall be created (if no *objectclass* is given).
view	String	the current view	Id of the view which shall be filled into the object. This may also be the id of a NestedFormComponent in the form *[viewid].[nestedformcomponent id]*
*	*	NULL	Any other request object is set to the created object as attribute.
obj_type	String	required (ContentRepository objects only)	The TypeId of the object to create.

Table 4.111. Response Objects: Form2ObjectAction

Name	Type	Description
object	Object	The created object filled with the additional action parameters and the content from the view components.

Table 4.112. Action: Form2ObjectAction - Return Values

TRUE	If the object could be created and filled.
FALSE	If an Exception occurred during creation or filling.

4.2.2.8.14. GeneralViewAction

The GeneralViewAction can be used to perform several tasks on views or on the portal itself: It can clear a view, switch to another view and set view or portal properties.

Table 4.113. Parameters: GeneralViewAction

Name	Type	Default	Description
clearview (multiple)	String	NULL	Id of the view to be cleared. This may also be the id of a NestedFormComponent in the form. The full path notation provides the parameter module to specify a ViewPortlet. This is useful if you want to clear Views in other portlets. Wildcards are also possible. The calling ViewPortlet is that ViewPortlet that contains the View with the GeneralViewAction that invokes clearview. The short notation will only invoke clearview on the Views of the calling ViewPortlet Syntax for the notation. The round brackets include an optional extension of the notation: *(portal.modules.[moduleid].pl*

Name	Type	Default	Description
			ugins.viewplugin.views.)[viewid][*].[nestedformcomponentid][*] .
showview	String	NULL	Id of the view to be shown next. This may also be the id of a NestedFormComponent in the form *[viewid].[nestedformcomponentid]* .
hideview (multiple)	String	NULL	Id of the NestedFormComponent to hide. Must be of the form *[viewid].[nestedformcomponentid]* .
set (multiple)	String	NULL	Set command for modification of a property. See Section 4.7, “Property Setters” for details. If you want to modify eg. a text component's contents you can use it's data property to add your custom text: *view.components.[componentid].data = "my custom text"* It is possible to resolve variables before the set is actually executed by using the syntax ${resolvable.path}. See “GeneralViewAction Usage Example” (Example 4.38, p.151). Using this syntax it is possible to resolve all properties from within the “ActionContext” (Section 4.2.2.8.2.1, p.140).

No Response Objects.

No Return values.

Note

Please pay special attention to the fact that action parameters are not bound to be processed as listed. In fact if you use two "set" parameters where the second depends on the first you have to use two General View Actions to maintain correct execution sequence.

Example 4.38. GeneralViewAction Usage Example

```
<action class="GeneralViewAction">
  <parameters>
    <parameter id="clearview">editview</parameter>
    <parameter id="showview">editview</parameter>
    <!-- Setting a static value -->
    <parameter id="set">views.editview.components.textfield.data =
          "Some static value"</parameter>

    <!-- Setting a variable value -->
    <parameter id="set">views.editview.components.textfield.data =
          portal.vars.now</parameter>

    <!-- Use some crazy way to set portal.vars.now
                                  (containing the current date) -->
```

```
    <parameter id="set">views.editview.components.textfield.data =
          "now"</parameter>
    <!-- the textfield now contains the string 'now' so use
                                  this to resolve portal.vars.now -->
    <parameter id="set">views.editview.components.textfield.data =
          portal.vars.${views.editview.components.textfield.data}
    </parameter>

    <!--
         The above example is of course only for demonstration,
         in reality it is
         1.) completely useless and
         2.) not guaranteed to work because the order in which
             'set' commands are executed is undefined.
    -->

  </parameters>
</action>
```

4.2.2.8.15. ImpEncapsulateAction

This action can be used to call Imp methods as action. See Section 4.1, “Imps” for information about IMPs.

Table 4.114. Parameters: ImpEncapsulateAction

Name	Type	Default	Description
imp	String	(required)	Id of the IMP to be used. - Can also be a direct reference to an Imp you've obtained through #getImp or #returnImp
method	String	(required)	Name of the method to call from the Imp. See the Documentation of the Imp you are calling for a description of methods. There are two special methods #getImp and #returnImp - If you want to use an Imp which is not stateless and you need to call several methods on the same Imp instance you can use #getImp to get an imp (returned in the response object as 'imp') and #returnImp to return it (you always need to call #returnImp when you are done using an imp you obtained with #getImp).
param1 (param2,param3,..)	Object (depending on imp method)	NULL	Depending on the imp method you want to invoke the number of param1 (param2,param3) and the type may differ.
paramCount	Integer	(optional)	If you want to pass null values as parameters, you can set this to the number of parameters the method you want to call has. Otherwise all param1,2,3,.. values will be taken until the first null value.

Table 4.115. Response Objects: ImpEncapsulateAction

Name	Type	Description
result	Object	The result returned by the invoked method.

Table 4.116. Action: ImpEncapsulateAction - Return Values

TRUE	If the method was invoked successfully
FALSE	If an error occured (e.g. an exception when invoking the method, or if the parameters couldn't be converted.

Example 4.39. Example Usage of ImpEncapsulateAction

```
<action class="ImpEncapsulateAction" id="regexpexample">
  <parameters>
    <parameter id="imp">string</parameter>
    <parameter id="method">regexp</parameter>
    <parameter id="param1">Hello Regexp</parameter>
    <parameter id="param2">Reg.*</parameter>
    <parameter id="param3">World</parameter>
  </parameters>
</action>
```

The above example uses the Section 4.1.5, “String Formatter” to rewrite the input 'Hello Regexp'. You can afterwards access the value of the action through actions.regexpexample.result within following action.

4.2.2.8.16. JsonCallableActionResponseAction

Returns JSON code from a Callable Action.

Table 4.117. Parameters: JsonCallableActionResponseAction

Name	Type	Default	Description
*	Object	NULL	The content that should be returned to the client. The data is encoded as JSON.

Table 4.118. Response Objects: JsonCallableActionResponseAction

Name	Type	Description
response	String	The JSON code that will be returned to the client.

Table 4.119. Action: JsonCallableActionResponseAction - Return Values

TRUE	When no error occurred.
FALSE	If an error occured.

4.2.2.8.17. Object2FormAction

This Action will take a Resolvable Object and transfer it to the fiels of a form according to given set of instructions.

Table 4.120. Parameters: Object2FormAction

Name	Type	Default	Description
object	Object	(required)	Object that will be displayed in the Form.
view	String	(required)	Id of the view that will be filled with data of the object. This may also be the id of a NestedFormComponent in the form `[viewid].[nestedformcomponent id]` .
activateview	Boolean	false	If set to true the view will be activated after values have been successfully set.
clearview	Boolean	false	If set to true the view will be cleared before values are filled.

No Response Objects.

Table 4.121. Action: Object2FormAction - Return Values

TRUE	If transfer worked.
FALSE	If an exception was thrown.

4.2.2.8.18. PDF2TextAction

Extracts the text content from PDF documents.

Table 4.122. Parameters: PDF2TextAction

Name	Type	Default	Description
pdf	InputStream or String	(required)	PDF document to be converted into a String.
maxpdffilesize	Integer	10 MB	Maximum allowed filesize of the PDF document in bytes. Note that this limit is not effective if it is larger than the overall upload limit for the portal.

Table 4.123. Response Objects: PDF2TextAction

Name	Type	Description
content	String	Text that was extracted from the given PDF document.

Table 4.124. Action: PDF2TextAction - Return Values

TRUE	When no error occurred.
FALSE	When the pdf could not be converted to text.

4.2.2.8.19. PlainCallableActionResponseAction

Returns plain text from a Callable Action.

Table 4.125. Parameters: PlainCallableActionResponseAction

Name	Type	Default	Description
*	String	NULL	The content that should be returned to the client. If more that one parameter is specified, the contents of all parameters are simply concatenated.

Table 4.126. Response Objects: PlainCallableActionResponseAction

Name	Type	Description
response	String	Text that will be returned to the client.

Table 4.127. Action: PlainCallableActionResponseAction - Return Values

TRUE	When no error occurred.
FALSE	If an error occured.

4.2.2.8.20. RenderTemplateAction

Renders the given template source. Returns the render result and all objects in the template context after the template was rendered. This provides the possibility to generate objects in the template that can be used later. You can also use the RenderTemplateAction for scripting which may come in handy.

Table 4.128. Parameters: RenderTemplateAction

Name	Type	Default	Description
template	String	(required)	Source of the template to be rendered. See Section 4.11.4, “TemplateEngine2” .
scriptEngine	String	(optional)	Scriptengine to use. Options are "Velocity" or "JavaScript". Defaults to Velocity [http://velocity.apache.org/] when undefined. The JSR-233 compliant JavaScript Engine [http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/index.html] requires Sun JDK 1.6 or higher. There is a Programmer's Guide [http://java.sun.com/javase/6/docs/technotes/guides/scripting/programmer_guide/index.html] as well.
*	*	NULL	Any other request object is provided in the template/script context under its name.

Table 4.129. Response Objects: RenderTemplateAction

Name	Type	Description
template	String	Result of the rendered Template.
*	*	All objects from the template context. This also includes objects that were created during processing of the template.

Table 4.130. Action: RenderTemplateAction - Return Values

TRUE	When no error occurred.
FALSE	If there was an error rendering the template.

4.2.2.8.21. Resolvable2MapAction

Resolves all specified paths for the given object into a map. Returns the generated map as `result`.

Table 4.131. Parameters: Resolvable2MapAction

Name	Type	Default	Description
object	Object	(required)	The object from which the attributes shall be resolved.
attribute	String		The name of the attribute that should be resolved from the object.

Table 4.132. Response Objects: Resolvable2MapAction

Name	Type	Description
result	Map	The map containing all resolved attributes.

Table 4.133. Action: Resolvable2MapAction - Return Values

TRUE	When no error occurred.
FALSE	If an error occured.

4.2.2.8.22. RuleSearchAction

The RuleSearchAction restricts the objects shown in a DatasourceListComponent by setting the configured rule. When called the action will set the given searchrule for the DatasourceListComponent of the searchview. The searchrule will resolve the baseobject "data." to all additional request parameters given to the action. The values can be included directly (data.[name]) or converted into a search pattern for comparisons with LIKE (data.[name]_pattern). Finally the searchview is activated.

Table 4.134. Parameters: RuleSearchAction

Name	Type	Default	Description
searchview	String	(required)	Id of the view holding a datasourcelist-component
component	String	NULL	Id of the datasourcelistcomponent in the searchview that will show the search results.
searchrule	String	(required)	Rule that will be set as filterrule for the datasourcelistcomponent of the searchview.
*	*	NULL	any other request object can be used inside the searchrule as `data.[parameterid]` or `data.[parameterid]_pattern`

No Response Objects.

Table 4.135. Action: RuleSearchAction - Return Values

TRUE	on success.
FALSE	on error.

4.2.2.8.23. ScriptingAction

This Action is an alias for the Section 4.2.2.8.20, “RenderTemplateAction” . It can be used to execute scripting code and return created variables as well.

4.2.2.8.24. SendMailAction

The SendMailAction is used to send emails.

Note

All e-Mail adresses have to be formatted following the specification RFC 822 (see http://www.ietf.org/rfc/rfc822.txt).

Table 4.136. Parameters: SendMailAction

Name	Type	Default	Description
mailhost	String	(required)	Mail Host to be used to send mails.
to	String, Object or List of Objects	(required)	Recipients of the mail.
toemail	String	NULL	Name of the attribute holding the email address, when `to` is given as Object(s).
cc	String	NULL	CC of the mail.
bcc	String	NULL	BBC of the mail.
from	String	(required)	Sender of the mail
subject	String	(required)	Subject of the mail. This may be a template and may contain all given action parameters under their respective names. See Section 4.11.4, “TemplateEngine2”
body	String	(required)	Mail body. This may be a template and may contain all given action parameters

Name	Type	Default	Description
			under their respective names. See Section 4.11.4, “TemplateEngine2”
*	*	NULL	Any additional parameters are provided to the template contexts for *`subject`* and *`body`*.

No Response Objects.

Table 4.137. Action: SendMailAction - Return Values

TRUE	on success
FALSE	on error

Tip

You can use Portal Properties (see Section 3.8.2.21, “Portal Properties”) to configure properties like `mailhost` which are unlikely to differ among various SendMailActions.

4.2.2.8.25. SendRedirectAction

Sends a HTTP Redirect response to the client. (Can only be used for absolute URLs). Make sure you execute this action before the rendering starts, e.g. putting this action in the onview-section of a view will cause an error.

Table 4.138. Parameters: SendRedirectAction

Name	Type	Default	Description
url	String	(required)	Absolute URL to redirect to.

No Response Objects.

Table 4.139. Action: SendRedirectAction - Return Values

TRUE	If redirect was set successfully.
FALSE	On Error.

4.2.2.8.26. StoreFileAction

Stores binary or String data into a file.

Table 4.140. Parameters: StoreFileAction

Name	Type	Default	Description
content	Stream, byte array or String	(required)	Content to store into the file.
filename	String	(required)	Name of the created file.
temporary	Boolean	false	Whether the file shall be a temporary file or not. Temporary files will internally get new filenames and will never overwrite

Name	Type	Default	Description
			existing files.
directory	String	Session temporary path (or ${java.io.tmpdir} if not called within a portal session).	Absolute path of the directory where to store the file.

Table 4.141. Response Objects: StoreFileAction

Name	Type	Description
filepath	String	Absolute path of the stored file.
fileinformation	Object	FileInformation object holding additional data of the stored file. This object can be passed to a "FileUploadComponent" (Section 4.2.2.7.2.6, p.112) to provide download links for the stored file. The FileInformation object has the following properties • *fileSize*: size of the file • *contentType*: content type (mime-type). e.g. image/gif, ... • *fileName*: name of the file (without path information) • *filePath*: absolute path of the file • *inputStream*: stream to load the data from the file

Table 4.142. Action: StoreFileAction - Return Values

TRUE	When the file was successfully stored
FALSE	On error

4.2.2.8.27. TextDiffAction

Generates a word-based diff between two strings. All templates for rendering modifications support the following template variables:

- *$add*: added content
- *$addsource*: source code of the added content (special characters are html encoded)
- *$remove*: removed content
- *$removesource*: source code of the removed content (special characters are html encoded)
- *$before*: content before the modification

- *$after*: content after the modification

Table 4.143. Parameters: TextDiffAction

Name	Type	Default	Description
text1	String	(required)	Original version of the string.
text2	String	(required)	Modified version of the string.
addtemplate	String	<ins class="diff">$add</ins>	Template for rendering add text parts.
removetemplate	String	<del class="diff">$remove</del>	Template for rendering removed text parts.
changetemplate	String	<del class="diff modified">$remove</del><ins class="diff modified">$add</ins>	Template for rendering modified text parts.
showhtml	Boolean	FALSE	Whether to show html tags in the diff. When set to TRUE, the rendered output will show the html source code.
ignoreregex	String	NULL	Regular expression of text parts that shall be ignored when diffing (e.g. timestamps, ...).
wordsbefore	Integer	10	Number of words before a modified text part to be contained in the template variable *$before*.
wordsafter	Integer	10	Number of words after a modified text part to be contained in the template variable *$before*.

Table 4.144. Response Objects: TextDiffAction

Name	Type	Description
diff	String	Rendered diff between the given contents.

Table 4.145. Action: TextDiffAction - Return Values

TRUE	When the diff was successfully generated
FALSE	On error

4.2.2.8.28. TriggerEventAction

The TriggerEventAction triggers an event that can be caught in the module using the views. All configured parameters are set as event properties and can be used in the module.

Table 4.146. Parameters: TriggerEventAction

Name	Type	Default	Description
event	String	(required)	command of the event to be triggered. This will trigger the event `portal.modules.[pnodeid].plugins.viewplugin.[event]`, where *`[pnodeid]`* is the id of the portlet's pnode and *`[event]`* is the event's name configured here.
*	*	NULL	Any additional parameters are set as event properties.

No Response Objects.

No Return values.

4.2.2.8.29. TriggerPortalEventAction

The TriggerPortalEventAction is an alternative of the TriggerEventAction, which allows full control of the event path.

Table 4.147. Parameters: TriggerPortalEventAction

Name	Type	Default	Description
path	String	(required)	Full path of the triggered event.
*	*	NULL	Any additional parameters are set as event properties.

No Response Objects.

No Return values.

4.2.2.8.30. URLLoaderAction

The URLLoaderAction loads a file from the given url. URL can be a file:///, http://, https:// or anything else supported by java. See the javadoc for the class URLConnection [http://java.sun.com/j2se/1.4.2/docs/api/java/net/URLConnection.html] for more information.

Table 4.148. Parameters: URLLoaderAction

Name	Type	Default	Description
url	String	(required)	The URL to be used to download the file.
method	String	GET	(Only for HTTP URLs) Method to be used in the request (GET,POST,HEADER,etc.)
timeout	Integer	10	After how many seconds the action should give up trying to connect to the given URL.
readTimeout	Integer	10	After how many seconds the action should give up on downloading from the given URL.
postparameters	String	(not set)	(Only for HTTP URLs) The POST parameters sent in the body of an HTTP re-

Name	Type	Default	Description
			quest. (e.g.: key1=value1&key2=value2)
encoding	String	Default System Encoding or Content-Type when loading from HTTP	Can be either any encoding supported by Java (e.g. UTF-8) or "binary" - if it is "binary" the response object "content" will contain a byte array instead of a String
headers	Collection (String) or Map (String, String)	(not set)	Additional headers that will be sent with the request. Unless a Map (String, String) is specified, each entry must contain the name and value of the header, delimited by a colon (e.g. "x-device-type: Blackberry 9000").

Table 4.149. Response Objects: URLLoaderAction

Name	Type	Description
content	String or byte[]	The content fetched from the URL - depending on 'encoding' either String or byte[] (for binary)
size	Integer	Size of the content in characters
headers	Map (String, Collection (String))	(Only for HTTP URLs) The header properties of the response.
filename	String	Name of the loaded file (either from the response header Content-Disposition or guessed from the URL).
contenttype	String	Content type (mime-type) of the loader file (either from the response or guessed from the file name).

Table 4.150. Action: URLLoaderAction - Return Values

TRUE	on success
FALSE	on error (invalid URL syntax, timeout, etc.)

4.2.2.8.31. XSLTRenderAction

Transforms a given XML source string using a given XSLT.

Table 4.151. Parameters: XSLTRenderAction

Name	Type	Default	Description
xmlsource	String	(required)	The XML source to be transformed using XSLT.
xslt	String	(required)	XSLT used to transform the XML source string.

Table 4.152. Response Objects: XSLTRenderAction

Name	Type	Description
size	Integer	Size of the transformed content string.

Name	Type	Description
content	String	Transformed content.

Table 4.153. Action: XSLTRenderAction - Return Values

TRUE	On Success
FALSE	On Error (e.g. invalid XML/XSL syntax)

4.2.2.9. Callable Actions

4.2.2.9.1. Introduction

Callable Actions are actions that can be triggered by client AJAX requests. Each Callable Action can contain multiple Pluggable Actions (see Section 4.2.2.8, “Pluggable Actions”), which are used to perform the desired actions. By using specialized Pluggable Actions, namely the `PlainCallableActionResponseAction` (see Section 4.2.2.8.19, “PlainCallableActionResponseAction”) and the `JsonCallableActionResponseAction` (see Section 4.2.2.8.16, “JsonCallableActionResponseAction”), the response to the client is constructed.

```
<callableactions>
        <callableaction id="">
                <actions>
                        ...
                </actions>
        </callableaction>
</callableactions>
```

4.2.2.9.2. Template Variables

Callable Actions are accessible in view and component templates using `$form.callableactions.[actionid]`. The URL string of the Callable Action is provided as `$form.callableactions.[actionid].url`. You may prefer using a ResourceURL as object, which can be generated by using `$form.callableactions.[actionid].resourceUrl`. With the resourceurl object, it is possible to add additional parameters to the URL.

4.2.2.9.3. Input Parameters

Any URL parameters passed to the Callable Action are available as `javax.portlet.request.parameters.*` to the defined Pluggable Actions. Please note that since a URL parameter can have multiple values, the values have to be treated as arrays.

4.3. Portal implementation

4.3.1. Introduction

The portal provides some basic events, which can be used for reactions.

4.3.2. Portal events

- portal.events.onLogin - successfull user login

- portal.events.onSessionCreate - new usersession
- portal.events.onTemplateChange - a page impression where the output has changed somehow
- portal.events.onPrepareRender - the lifecycle moment before rendering

4.3.3. PBox events

Events triggered, when pboxes change their state. These events are never triggered, when the implementation uses portal pages. See “Portal Page events” (Section 4.3.6, p.165) for details about the portal page events.

- portal.pboxes.[pboxid].windowstate.onMinimized - The windowstate of the pbox was changed to "minimized"
- portal.pboxes.[pboxid].windowstate.onNormal - The windowstate of the pbox was changed to "normal"
- portal.pboxes.[pboxid].windowstate.onMaxmimized - The windowstate of the pbox was changed to "maximized"
- portal.pboxes.[pboxid].portletmode.onView - The portletmode of the pbox was changed to "view"
- portal.pboxes.[pboxid].portletmode.onEdit - The portletmode of the pbox was changed to "edit"
- portal.pboxes.[pboxid].portletmode.onHelp - The portletmode of the pbox was changed to "help"
- portal.pboxes.[pboxid].onModuleIdChange - The module id displayed in the pbox has changed. Event contains the properties: *`pbox`* and *`oldModuleId`*

4.3.4. ViewPlugin events

- portal.modules.[moduleid].plugins.viewplugin.onViewChange - The active view has changed.
  - *`portlet`* The modified portlet.
  - *`oldview`* The id of the old view.
  - *`newview`* The id if the new view.
- portal.events.onViewChange - The active view has changed (alias for portal.modules.[moduleid].plugins.viewplugin.onViewChange)

4.3.5. Portlet events

This section describes some general portlet events. The events are only triggered, when portal pages are used. See “Portal pages Configuration” (Section 3.8.2.15, p.35) for details about how to activate portal pages.

- portal.events.portlet.onBeforeWindowStateChange - The windowstate of a portlet is about to be changed. The event contains the following properties:
  - *`portlet`* The modified portlet

- *position* The position of the modified portlet
- *portalpage* Portalpage of the modified portlet
- *oldwindowstate* Name of the current windowstate (lowercase).
- *newwindowstate* Name of the new windowstate (lowercase).
  Note that this event will be triggered, even if the portlet cannot change in the requested windowstate, because the position does not allow it.

- portal.events.portlet.onWindowStateChange - The windowstate of a portlet was changed. The event contains the following properties:
  - *portlet* The modified portlet
  - *position* The position of the modified portlet
  - *portalpage* Portalpage of the modified portlet
  - *windowstate* Name of the new windowstate (lowercase).
- portal.events.portlet.onPortletModeChange - The portletmode of a portlet was changed. The event contains the following properties:
  - *portlet* The modified portlet
  - *position* The position of the modified portlet
  - *portalpage* Portalpage of the modified portlet
  - *portletmode* Name of the new portletmode (lowercase).
- portal.events.portlet.onChange - Is triggered when portlet positions are changed. The event contains the following properties:
  - *portlet* The modified portlet
  - *position* The position of the modified portlet
  - *portalpage* Portalpage of the modified portlet
  - *index* The index of the portlet in the current position

4.3.6. Portal Page events

Events triggered by portal pages. These events are never triggered, when the implementation does not use portal pages. See “PBox events” (Section 4.3.3, p.164) for details about pbox events.

- portal.events.onBeforePageChange - the user is about to change the active portalpage (implicitly or explicitly). This event is triggered, before the page is actually changed and contains the following properties:
  - *oldpage* Id of the old portalpage.
  - *newpage* Id of the new portalpage.
- portal.events.onPageChange - the user has changed the active portalpage (implicitly or explicitly). This event is triggered, after the page is changed and contains the following properties:
  - *oldpage* Id of the old portalpage.

  - *newpage* Id of the new portalpage.

- portal.pages.[pageid].positions.[positionid].onChange - Portlets in the given position have changed (This can be because a portlet was added, removed or sorted differently. The event contains the following properties:
  - *position* The modified position.
  - *portalpage* The portal page of the position.
  - *oldPortlets* Collection as it were before the current change.

4.3.7. Custom action events

portal.events.actions.on[Actionname]

Events triggered, when custom actions are clicked by the user for a specific portlet entity. These events will only be triggered, when portal pages are used.

The event contains the following properties:

- *portlet* The portlet entity for which the action was triggered
- *position* The position of the portlet
- *portalpage* Portalpage of the portlet

4.3.8. Reactions

Reactions can be defined in <pnode> Tags in the portal template or the portletentities file. See “PNodes” (Section 4.11.1.3, p.193) or “Portlet Entities Configuration” (Section 3.13, p.78) for details.

Reaction implementations can be made in two different ways:

1. (deprecated) implement a list of assignments, each one in a new line. This implementation has some major limitations (no functions, may not contain quotes, ...) and is therefore deprecated.
2. Implement a single expression (which can consume more than one line). If e.g. many assignments have to be done in the same reaction, this can be accomplished by using the function `do()`. For details on expressions see “ExpressionParser” (Section 4.10, p.182) .

Either way, one can use property paths starting with three base words in reactions:

- `portal` for resolving portal properties (see “Portal” (Section 4.8.1, p.175) for a description of possible property paths)
- `event.properties` for resolving event properties. See the event descriptions for details on the available properties.
- `portlet` for resolving properties of the portlet-entity (pnode) holding the reaction. Paths based on `portlet` will resolve to `null` when the reaction is defined in the general reactions node of a portletentities definition.

4.4. Portlet implementation

4.4.1. Introduction

This section covers information needed during implementation of portlets. The portlets themselfes are described in the section Chapter 5, *Portlets*.

Further information about portlets, portlet applications and the portlet descriptor can be found in the JSR 168 Portlet Specification [http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html].

4.4.2. Portlet Application

A portlet application bundles several portlets together with other resources, such as templates, view definition files, images, dictionaries, etc.

Portlet applications can be developed in the Gentics Portal.Node SDK and then deployed onto a production system using the AdministrationPortlet. For details on the deployment, see Section 5.7, “AdministrationPortlet”.

Every portlet application provides a portlet application template loader with the *`loaderid`* `gentics.portletapp` . This loader loads the templates from the directory `/WEB-INF/templates` within the portlet application's context path. When templates are bundled with a portlet application, it is strongly recommended to locate them in files under the directory `/WEB-INF/templates` and to use this portlet application loader.

The portlet application's dictionaries must be located at `/WEB-INF/dictionaries/portal.[languageid].property` . See “Dictionaries” (Section 3.8.5.3, p.47) for details.

4.4.3. Portletdescriptor

The Portletdescriptor is used to define the portlets, that are part of a portlet application. It follows strictly the standard defined in the JSR 168 Portlet Specification [http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html].

4.4.4. Gentics Portletdescriptor

4.4.4.1. Introduction

The Gentics Portletdescriptor is used to configure specific extended settings of portlets and plugins. It consists of the file `WEB-INF/gentics-portlet.xml`.

Note

It is not necessary to define the default plugins in the Gentics Portletdescriptor of every portlet application. The default plugins are automatically available to all deployed portlet applications. Only when default settings (e.g. the templates) shall be overwritten for the plugins in a specific portlet application, you need to redefine this plugin in the portlet application.

Example 4.40. Syntax of the Gentics Portletdescriptor

```
<portlet-app xmlns="http://www.gentics.com/gentics-portlet-app">
  <portlet>
    <portlet-name>...</portlet-name>
    <parameter-description>
      <parameter-definition type="..." name="...">...</parameter-definition>
    </parameter-description>
    <start-parameters>
      <parameter name="...">...</parameter>
```

```
      ...
        </start-parameters>
    <templates>
      ...
    </templates>
    <portlet-preferences>
      <preferences-modifier></preferences-modifier>
    </portlet-preferences>
  </portlet>
  ...
  <plugin>
    <plugin-name>...</plugin-name>
    <plugin-class>...</plugin-class>
    <parameter-description>
      <parameter-definition type="..." name="..."/>
    </parameter-description>
    <templates>
      ...
    </templates>
  </plugin>
  ...
</portlet-app>
```

4.4.4.2. Configuration

Table 4.154. Configuration parameters in the Gentics Portletdescriptor

Name	Type	Description
portlet-app	Structure	Base node of the file. Note that the namespace declaration is needed.
portlet	Structure	Defines additional settings for a portlet. Not every portlet defined in the Portletdescriptor needs to have extensions defined here.
portlet-name	String	Name of the portlet like defined in the Portletdescriptor.
parameter-description	Structure	Description of parameters a portlet uses.
parameter-definition	Node	Definition of a single portlet parameter.
parameter-definition.name	String	Name of the portlet parameter.
parameter-definition.type	String	Type of the portlet parameter. Must be one of `string`, `integer`, `boolean`, `rule`, `node`, `listOfObjects` .
start-parameters	Structure	Definition of the default values for parameters.
parameter	Node	Default value for a parameter. The exact format and interpretation of the value depends on the parameter type.
parameter.name	String	Name of the parameter.
templates	Structure	Definition of portlet templates. See Section 3.8.7.4, “Template definitions” on details about definition of templates. You might also want to take a look at Section 3.8.7.3, “Template Loaders” for detailed information on Template Loaders.
portlet-preferences	Structure	Additional configuration settings regarding the portlet preferences.
portlet-preferen-	String	Class name of a preferences modifier. A preferences modifier is a class that may modify portlet preferences before a portlet

Name	Type	Description
ces.preferences-modifier		handles an action or renders its output. This can for example be used to modify the behaviour of a third party portlet, that relies on portlet preferences.
plugin	Structure	Definition of a plugin.
plugin-name	String	Name of the plugin.
plugin-class	String	Full qualified classname of the Java Class that implements the functionality of the plugin. The class must implement the interface `com.gentics.api.portalnode.plugin.GenticsPlugin` and is recommended to extend `com.gentics.api.portalnode.plugin.AbstractGenticsPlugin` .
parameter-description	Structure	See above.
templates	Structure	See above.

4.4.5. Portlet Cache

The Portlet Cache can be used to let Gentics Portal.Node cache the render output of a Portlet. This can lead to drastic performance improvements.

To activate the Portlet Cache for a specific Portlet you have to modify the Portlet Descriptor `portlet.xml` and add a <expiration-cache> setting (after <portlet-class>) - This defines the maximum number of seconds the output should be cached. You can disable caching by specifying 0 or use -1 if you don't want the cache to expire.

Example 4.41. Portlet Cache Configuration Example

This example would enable the Portal to cache the render output of this portlet for 10 seconds.

```
<portlet>
  <portlet-name>HelloCachedWorld</portlet-name>
  <portlet-class>com.gentics.HelloWorld</portlet-class>
  <expiration-cache>10</expiration-cache>
  ...
</portlet>
```

Note

To use the Portlet Cache it also has to be activated in the Portal Configuration. See “Portlet Cache” (Section 3.8.2.7, p.31) for more information.

4.4.5.1. Portlet Cache Behaviour

Before making use of the Portlet Cache you should have a basic understanding on how it works and when the cache is cleared.

According to the Portlet Specification (JSR 168) the render output of a Portlet is cached (and therefore it's render() method not called) as long as the Expiration Cache has not expired and no action or render request was targeted at the specific Portlet.

In addition to this definition Gentics Portal.Node will invalidate the cache of a Portlet if any property set-

ters are invoked. This way you can e.g. set properties of any portlet in a Reaction of “PNodes” (Section 4.11.1.3, p.193) or within a “GeneralViewAction” (Section 4.2.2.8.14, p.150) without having to worry about caching.

Warning

When using Portlet Caching you need to make sure that your portlet has no outside dependencies.

This means you should not reference anywhere within your portlet (e.g. in your views or visible rule of the pnode) to properties outside of your own portlet.

Instead, always use “Property Setters” (Section 4.7, p.175) if the state of your portlet should change.

4.5. Inter Portlet Communication

4.5.1. Java Property Getting and Setting

When implementing a GenticsPortlet inter portlet communication may be established by reading information from other portlets using the getPortletContext().resolvePortalProperty("portal.[propertypath]") method, or writing data using getPortletContext().setPortalProperty("portal.[propertypath]") respectively. You may want to head over to the “Portal” (Section 4.8.1, p.175) chapter for details on Portal Property Paths.

4.5.2. Reactions

A portlet entity's reactions may also be used for inter portlet communication. Just read and set parameters to exchange data.

Example 4.42. Setting a portlet's parameter

```
<reactions>
        <reaction event="portal.events.portlet.onWindowStateChange">
                portal.modules.mymodule.parameters.myparameter = "test"
        </reaction>
</reactions>
```

4.6. Ajax

4.6.1. Ajax using Callable Actions

Portlets can be Ajax-enabled by using Callable Actions (see “Callable Actions” (Section 4.2.2.9, p.163)). Each Callable Action can contain multiple Pluggable Actions which are used to perform the desired actions. By using specialized Pluggable Actions, namely the `PlainCallableActionResponseAction` (see Section 4.2.2.8.19, “PlainCallableActionResponseAction”) and the `JsonCallableActionResponseAction` (see Section 4.2.2.8.16, “JsonCallableActionResponseAction”), the response to the client is constructed.

Using a client-side JavaScript framework such as jQuery, existing View portlets can be easily extended to use Ajax functionality. For examples using Callable Actions to provide features like autocompletion, see the SDK portal.

4.6.2. Ajax using JSR 286

Custom portlets can be Ajax-enabled by using the new `serveResource()` method defined in the JSR 286 standard. Using resource URLs, portlets can return arbitrary data to the client, which can be used by client-side JavaScript frameworks. For an example on how resource serving may be used see the ServeResourceExample from the advanced examples page in Gentics Portal.Node SDK.

4.6.3. AJAX enabled requests to the Server

Besides specially implemented portlets that use AJAX functionality, it is also possible to enhance the overall user experience of the portal implementation by using AJAX enabled requests to the portal server. The main idea is to avoid full page reloads for every single click into a portlet (or into the portlet's frame), but to issue AJAX requests to the server instead and let the AJAX framework just replace the new rendered portlets in the portal page. This feature is called "Portlet Reloading" and will be described later in detail. Another example of AJAX enabled requests is "Portlet Drag & Drop", which even goes a step further: Here, part of the implementation (reordering the portlets on the portal page) is done on the client side, and only the result sent to the server as AJAX request (which again might lead to some portlets be re-rendered).

Any portal URL can be made AJAX enabled, by simply adding the parameter *`gentics.rl`* = `true` . The only difference to a non-AJAX enabled request is the format of the response. For a "normal" request, the response will always be the fully rendered portal page. The response format of an AJAx enabled request is described in the next section.

4.6.3.1. Response Format of AJAX enabled requests API

The response of an AJAX enabled request is a JSON object of the following structure:

Table 4.155. Portlet Reload Response Structure

Name	Type	Description
*	Array	An array of JSON objects.
element in *	JSON Object	A single result object.
element.functionName	String	The name of the JavaScript function which should be called. See “Replace rendered portlets API” (Section 4.6.3.3.3.3, p.174) for the function names, which are used for portlet reloading.
element.parameters	JSON Object	The parameters for the function.

4.6.3.2. Response of AJAX enabled requests with an invalid session

When an AJAX enabled request is done with an invalid session cookie (e.g. when the session timed out), the response to this request cannot be what the user will expect. In such a case, the response will only contain the single function call to the client side function named `GENTICS.invalidSession` which should be implemented appropriately.

A basic implementation of this function would for example simply call `location.reload();` to reload the whole page, such that the user can continue browsing the portal with the new session.

Additionally, EVERY request to the Portal to a new Session (i.e. with invalid or no session cookie) will result in a response that contains the header "X-Gentics: NEWSESSION".

4.6.3.3. Portlet Reloading

Portlet reloading enables the use of AJAX requests to selectively reload portlets without the need of a

full-page reload. It does so by replacing the default link and submit actions by JavaScript methods, which submit the data to the portal server and interpret the response. Please note that portlet reloading only works together with portal pages (see “Portal Pages Implementation” (Section 4.14, p.201)). Portlet reloading is fully backwards-compatible and will fall back to the default behaviour (full-page reload) if JavaScript is disabled or unavailable on the client.

The portal server will try to detect, which portlets actually have changed their state upon an AJAX request and will render them. The algorithm is very similar to the automatic clearing of portlet caches in a "normal" (non AJAX) render-, action- or resource-request.

Note

It is important to keep in mind, that the change detection of portlets fail, when portlets have dependencies to other portlets e.g. by reading properties of them. The change detection only works, when properties of portlets are actively changed.

Note

Portlet reloading should not be enabled for portlets using the downloadcomponent in views (like the AdministrationPortlet). The downloadcomponent will not work properly when used in conjunction with portlet reloading. See Section 4.2.2.7.3.3, “DownloadComponent” for details on the downloadcomponent.

4.6.3.3.1. Parameters

Portlet reloading can be enabled portal-wide using portal parameters (see “Portlet Reloading” (Section 3.8.2.17, p.36)) or selectively for specific portlets using pnode parameters (see “Parameters of all Gentics Portlets” (Section 5.2.3, p.208)). For portal parameters use key "portal", for pnode parameters use "gentics".

Setting *[gentics|portal].portletreload* to `true` enables portlet reloading. To have the client-side script replace the standard links, set *[gentics|portal].portletreload.replacelink* to `true`. To also handle form submits by AJAX, set *[gentics|portal].portletreload.replaceform* to `true`. Which links and forms are replaced is defined by the *[gentics|portal].portletreload.allclasses* and the *[gentics|portal].portletreload.replaceclasses* parameters. If *[gentics|portal].portletreload.allclasses* is set to `true`, all elements are replaced. *[gentics|portal].portletreload.replaceclasses* can contain a comma-separated list of CSS classes, which define the elements to replace (i.e. the element must have at least one of the specified CSS classes to be replaced).

4.6.3.3.2. Templates

To use portlet reloading, several templates must be adapted.

4.6.3.3.2.1. Portal Page Template

The portal page template (see Section 4.14.1, “Portal Page Templates”) must include the necessary JavaScript files (see the SDK Portal for an example). The HTML element surrounding a portlet position must have a CSS class in the form `gentics-portletreload-position-[positionid]`.

Example 4.43. Portal Page Template Example

```
...
<ul class="gentics-portletreload-position-left">
        #foreach($portlet in $portal.page.positions.left.portlets)
                $portlet
        #end
</ul>
...
```

4.6.3.3.2.2. Portlet Frame Template

In the portlet frame template (see "Portlet Frame Templates" (Section 4.14.2, p.202)), the outermost HTML element must have specific CSS classes, which have to be included using `$portlet.portletreload` (see the portlet frame template in the SDK portal for an example).

Example 4.44. Portlet Frame Template Example

```
<li class="$portlet.portletreload">
...
</li>
```

Additionally, it is necessary to define a portlet frame template for invisible portlets, that renders an invisible placeholder with the css classes.

Example 4.45. Portlet Frame Template Example for invisible portlets

```
<li class="$portlet.portletreload" style="display:none">
</li>
```

4.6.3.3.3. Client-side scripting

This section describes, how the Gentics Portal.Node APIs for the JSON response to AJAX enables calls can be used to implement Javascript methods to dynamically replace portlets in the browsers DOM tree with new versions. Based on this description you can either use the example implementation of the Gentics Portal.Node SDK Demo Portal, or implement it on your own.

Client-side scripting consists of three parts:

1. Automatic modification of URLs to be AJAX enabled and using them in AJAX calls
2. Generic handling of the response and calling the specified javascript methods
3. Implementation of the portlet-reloading specific javascript functions (substitution of an already rendered portlet with a new version or adding a portlet to a position)

4.6.3.3.3.1. Automatic modification of URLs to be AJAX enabled

For all rendered portlets that have the *`portletreload`* feature activated, the AJAX framework must bind callback functions that will automatically make AJAX requests whenever a link is clicked or a form is submitted from this portlet. The Gentics Portal.Node SDK Demo Portal contains an example implementation of this in the file `gentics-portletreload.js` (method `GENTICS.PortletReload.replaceAjaxReloadUrls`). When javascript is not available in the browser, the links and forms will not be AJAX enabled and will still work as expected.

4.6.3.3.3.2. Generic handling of the response

The JSON response of an AJAX enabled request must be interpreted and all specified javascript functions must be called. The Gentics Portal.Node SDK Demo Portal contains an example implementation for this in the file `gentics-ajax.js` (method `GENTICS.processResult`).

4.6.3.3.3.3. Replace rendered portlets API

To render portlets in a specific position, the javascript function `GENTICS.PortletReload.renderPortlet` must be implemented. It will get the following parameters (as javascript object):

- *`portletId`* id of the portlet
- *`portletPosition`* id of the position in which the portlet needs to be rendered
- *`portletPositionIndex`* index of the portlet in the position (starting with 0). Note that also invisible portlets will consume an index number, when contained in the position.
- *`portletContent`* sourcecode of the rendered portlet (including the portlet frame).

The Gentics Portal.Node SDK Demo Portal contains an example implementation for this method in the file `gentics-portletreload.js` .

4.6.3.3.4. Restrictions and common pitfalls

- Portlet reloading should not be activated for a specific portlet (or link/form) when triggering this link/submitting this form will do one of the following:
  1. Login or logout the user
  2. Change the current portal page
  3. Change the visibility of portlets
- Portlet reloading will not render any part of a portal page template again, but only single portlets (with their frames). So if the portal page template itself contains dynamic code depending on the current status of portlets, it is necessary to have URLs that might possibly change the state of portlets that influence the rendering of the portal page to be "normal" URLs (not AJAX enabled). The portal server has no means to automatically detect such situations, so this must be handled in the implementation appropriately!
- Disabling portlet reloading for a single portlet does not prevent this portlet to be reloaded (e.g. by URLs to other portlets), but only prevents the URLs of this portlet to be AJAX enabled.

  Example: `portlet A` has portlet reloading activated and `portlet B` has portlet reloading not activated. Clicking an URL in `portlet A` might somehow modify the current status of `portlet B` as well, so both portlets would need to be re-rendered.

  This especially means that ALL portlets (no matter, whether they have portlet reloading activated or not) should be rendered using portlet frame templates with the necessary css classes (see “Portlet Frame Template” (Section 4.6.3.3.2.2, p.173)), and ALL positions should also have the necessary css classes (see “Portal Page Template” (Section 4.6.3.3.2.1, p.172)).

4.6.3.3.5. Disabling it for single button

Sometimes it is necessary to disable portlet reloading for only a single button. For example if it switches portlet pages or similar. To achive this, the following javascript code has to be executed prior to the form submit:

```
GENTICS.PortletReload.disabled = true
```

The default ButtonComponent template will add an onclick event to the button which executes the above statement if the component property "portal.portletreload" is set to "false".

4.6.3.4. Portlet Drag & Drop

Using a JavaScript framework, drag&drop for portlets can be implemented. Using a special Portal Action URL (available in the portal page template as `$portal.portletpositionchange`), the new positions of the portlets are transfered to the server (see Gentics Portal.Node SDK Demo Portal for an example and “Customizing portlet positions by URL” (Section 4.14.3.2, p.205) for details).

4.7. Property Setters

Property Setters are used to manipulate portal properties. A setter consists of two operands and an operator. The left operand is the path to a portal property and the right operand may be a static value or also a property path. Property Setters are mainly used in two places:

- GeneralViewAction see Section 4.2.2.8.14, “GeneralViewAction”
- pnode reactions see Section 4.11.1, “Portaltemplate”

See also Section 4.8, “Portal Property Paths” for details on property paths.

A Property Setter may be one of the following:

- *`path.to.object.property`* = *`path.to.source.property`*
- *`path.to.object.property`* = `"staticvalue"`
- *`path.to.object.property`* += *`path.to.source.property`*
- *`path.to.object.property`* += `"staticvalue"`
- *`path.to.object.property`* -= `path.to.source.property`
- *`path.to.object.property`* -= `"staticvalue"`

Note

The operators `+=` and `-=` can only be used on multivalue properties and will add/remove the referenced values.

Note

When *`path.to.object`* resolves to more than one object, ALL referenced objects that have to property *`property`* are modified in the same way.

4.8. Portal Property Paths

Portal Property Paths can be used to access properties in areas like View definitions or Velocity template files.

4.8.1. Portal

The portal property is the root for all other properties. All other property root paths are only shortcuts which refer to an element within the portal.* tree.

Table 4.156. Portal Property Path

Name	Type	Description
portal.imps.*	List	List of all imps provided by the portal. See Chapter 3, *Administration and Configuration* and Section 4.1, “Imps”
portal.i18n(String)	Method	used to display internationalized texts. syntax: $portal.i18n("welcome")
portal.i18n(String, String)	Method	used to display internationalized texts in a given language. syntax: $portal.i18n("welcome", "de"). Only operable under templateengine2 but not in expressions.
portal.user.*	Properties	The userproperties, as provided by the authmanager used.
portal.user.userid	String	UserID of the current user.
portal.user.isloggedin	Boolean	Whether the user is currently logged in.
portal.user.[authenticationid].*	Properties	Properties of a primary or secondary AuthenticationManager may be accessed directly by specifying the according <authentication>-id. This mechanism enables you to use a non-writeable authentication system as your primary AuthenticationManager whilst having a writable secondary AuthenticationManager to store additional data.
portal.modules.*	List	All portlet instances defined in the Portal Template, identified by their pnode id.
portal.portlets.*	List	Alias for portal.modules.*
portal.pboxes.*	List	All pboxes defined in the Portal Template, identified by their id.
portal.pboxes.*.moduleid	String	The portlet instance currently loaded in this pbox, identified by it's pnode id.
portal.vars.time	String	Current time as String. e.g. 19:20
portal.vars.date	String	Current date as String. e.g. 25.12.2006
portal.vars.now	Date	Current date as 'Date' object.
portal.vars.timestamp	Integer	Current date as timestamp. (Seconds since midnight, January 1, 1970 UTC.)
portal.datasources.[datasource-name].channel	Integer	This property can be used to set or retrieve the selected channel for the given datasource.
portal.language	String	The current language, and a synonym for portal.language.name
portal.language.name	String	The display name, including the defined locales, of the language.

Name	Type	Description
portal.language.id	String	The id of the language
portal.language.language	String	The language of the languages' locale.
portal.language.country	String	The country of the languages' locale.
portal.language.variant	String	The variant of the language's locale.
portal.languages	List	A list of all configured languages. See portal.language for their properties.
portal.languages.[id]	Object	Language with the specified id. See portal.language for its properties.
portal.defaultLanguage	Object	The global default language as configured in the portal configuration.
portal.request.path	String	Any extra path information associated with the URL the client sent when it made this request.
portal.request.port	Integer	The port number on which this request was received
portal.request.queryString	String	The query string that is contained in the request URL after the path.
portal.request.host	String	The host name of the server that received the request.
portal.request.parameters.*	Object(s)	Parameters of the request.
portal.request.remoteAddress	String	The Internet Protocol (IP) address of the client that sent the request.
portal.request.remoteHost	String	The fully qualified name of the client that sent the request.
portal.request.servletPath	String	Returns the part of this request's URL that calls the servlet. This includes either the servlet name or a path to the servlet, but does not include any extra path information or a query string. Typically, this will return either "/portal" or "/secure".
portal.request.contextPath	String	Returns the portion of the request URI that indicates the context of the request. The context path always comes first in a request URI. The path starts with a "/" character but does not end with a "/" character. Normally this will return "/Portal.Node".
portal.request.protocol	String	Returns the name and version of the protocol the request uses in the form protocol/majorVersion.minorVersion, for example, HTTP/1.1.
portal.request.requestURI	String	Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request. Typically the

Name	Type	Description
		requestURI is something like "/Portal.Node/portal".
portal.request.requestURL	String	Reconstructs the URL the client used to make the request. The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters.
portal.request.scheme	String	Returns the name of the scheme used to make this request, for example, http, https.
portal.request.secure	Boolean	Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.
portal.request.method	String	Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
portal.request.*	Object	Returns the return value of the corresponding bean getter method from the request object. See the Servlet API for available properties. (e.g. portal.request.localAddr)
portal.session.tmpdir	String	Path of the directory for temporary (session) files. Files stored in this path will only be available to this session and will be automatically removed when the session is invalidated or during startup and shutdown of Gentics Portal.Node
portal.session.id	String	Id of the current session.
portal.properties.*	Object(s)	Access (readable and writeable) to the portal properties.
portal.pagesetting	String	String containing the current page setting of the user, if portal pages are used (for storing user defined setting). See “General Parameters for portal pages” (Section 3.8.9.2, p.62) for details about storing the user settings.
portal.pages	Map	Map of portal pages, contains only the portal pages that have no rule defined or have a rule that currently matches.
portal.pages.[pageid]	Object	Portal page with id `pageid`.
portal.pages.[pageid].positions	Map	Map of positions in the portal page of the given id.
portal.pages.[pageid].positions.[positionid]	Object	Position with id `positionid` in the portal page with id `pageid` .
portal.pages.[pageid].positions.[positionid].portlets	List	List of portlet ids from portlets currently placed in the referenced position. This can be modified to change the positioning of portlets

Name	Type	Description
		on the portal page.
portal.pages.[pageid].reset	N/A	Set this attribute to any value (e.g. true) to reset the current user settings of a portal page.
portal.page	Object	The currently set portal page. It is possible to switch the current portal page by setting this property to the id of the desired portal page. When no portal page exists with the given id, or the requested portal page is invisible due to the given portal page rule, the current portal page is not changed. This object also provides access to information of the current portal page.
portal.request.host	String	Hostname included in the http request to the portal.
portal.request.port	Integer	Port to which the http request was directed.
portal.request.path	String	Path information included to the request to the portal. This does not contain the path of the portal (usually `/Portal.Node`) and the path to the servlet (usually `/portal` , `/public` or `/secure`).
portal.request.queryString	String	Query String contained in the http request.
portal.request.parameters	Map	Map of parameters included in the http request.
portal.response.redirect	String	This property can be used in reactions (ie. BEFORE the render phase) to force a HTTP redirect to another URL.
portal.createPortalURL()	Portal URL	Returns a portal URL which you can use (for example) to create a custom event. Can only be used in Velocity code.

```
## Create a triggerEvent URL
#set( $eventurl = $portal.createPortalURL("triggerEvent") )
## Set the event name
$eventurl.setParameter("event", "simpletest")
## add some various arguments
$eventurl.setParameter("someargument", "somevalue")
## output the event url
<a href="$eventurl">My Event URL</a>

## this example will trigger an event called: "portal.events.simpletest"
## it will also contain 'somevalue' at event.properties.someargument
```

4.8.2. Views

This PortalPropertyPaths are provided by the ViewPlugin in all View Rules and Component Templates.

Table 4.157. View PortalPropertyPaths

Name	Type	Description
views	List	List of all views. A single view can be accessed by either resolving it by its id (see below), or by iterating over the views List.
views.[id]	Object	A view object, see view
view	Object	The current view
view.label	String	Label of the view.
view.link	String	URL that activates this view.
view.components	List	All components of the current view, visible and invisible.
view.components.[id]	Object	A component. See Section 4.2.2.7.1, “All Components”.
view.components.[id].[property]	Object	A component property.
view.components.[id].[property].data	Object	The value of a component property.
view.properties	Map	All custom view properties, which can be read and/or set with this path.
view.properties.[id]	Object	Value of the custom view property [id] or null if the property is not set.
data	Object	Data provided by the Pluggable Action container.
actions	List	List of actions
actions.[id]	Object	Action object
actions.[id].[property]	Object	Action response properties.
module.*	Object	The portlet that contains this view. See Section 4.8.3, “Modules” for more details.

4.8.3. Modules

Generic Properties provided by AbstractGenticsPortlet. (Usually available as portal.modules.<module id>)

Table 4.158. AbstractGenticsPortlet - Property Paths

Name	Type	Description
parameters	Resolvable	Portlet Parameters
properties	Resolvable	Portlet Properties
plugins	Resolvable	Resolves all used plugins. (plugins.<plugin id> would resolve the plugin with the given id)
session	Resolvable	Resolves information about the portlet's session.

Name	Type	Description
session.id	String	Current session id.
session.portletscope.*	Object(s)	Access (readable and writeable) to the session attributes in portlet scope (attributes are only visible to this portlet).
session.appscope.*	Object(s)	Access (readable and writeable) to the session attributes in application scope (attributes are visible to all portlets in the same portlet application).

Note

Attribute data stored in the application scope of a portlet's session is not visible to portlets of another portlet application. If you need to store session specific data portal wide, you can use the portal properties. See Section 4.8.1, “Portal” for details on the portal properties.

4.8.4. Positions (in Portal Page)

The property path starting with "position" is only available in velocity templates (starting with the portlet-frame template) and only if portal pages are used.

Table 4.159. Positions - Property Paths

Name	Type	Description
position.id	String	Id of the currently rendered position
position.index	String	Index of the currently rendered portlet in its position (starting with 0)

4.8.5. javax.portlet

Details on the objects can be found in the API documentation.

Table 4.160. Portlet PortalPropertyPaths

Name	Type	Description
javax.portlet.config	Object	Portlet configuration
javax.portlet.context	Object	Portlet context
javax.portlet.request	Object	Portlet request
javax.portlet.request.preferences.*	Object	Portlet Preferences
javax.portlet.request.parameters.*	Object	Parameters
javax.portlet.request.portletMode	String	The current portlet mode in which this portlet is rendered.
javax.portlet.request.windowState	String	The current window state in which this portlet is rendered.
javax.portlet.response	Object	Portlet response

4.9. Rules

Note

Rules are now deprecated and replaced by the new “ExpressionParser” (Section 4.10, p.182) and should not be used anymore where possible.

4.9.1. Introduction

Rules are readable logical rulestrings to control portal and modules functionality, as well as filter data from datasource queries, independent of the datasource type.

Example 4.46. Rule examples

```
Rule within a <visible> of a component: (The component will invisible if
windowstate is not normal, but e.g. maximized)
<visible><prule>javax.portlet.request.windowstate != "normal"</prule></visible>

More complex rule for a DatasourceListComponent:
<rule><![CDATA[( view.components.fromdate.unixtimestamp == ""
       || object.edittimestamp >= view.components.fromdate.unixtimestamp )
    && ( view.components.untildate.unixtimestamp == ""
       || object.edittimestamp <= view.components.untildate.unixtimestamp )
]]></rule>
(This example assumes you have two DateComponents, fromdate and untildate
which are used to filter the objects within a DatasourceListComponent)
```

4.9.2. Variables

Objects matched against a rule are usually provided through the variable named "object".

4.9.3. Operators

CONTAINSONEOF, CONTAINSNONE, <, <=, >, >=, ==

To test if a specified value is null or empty you can use: value == "" (Things that will NOT work: value == null, !value)

4.10. ExpressionParser

4.10.1. Introduction

The ExpressionParser is the part of Gentics Portal.Node that parses and interprets expressions, like rules (prules), datasource filters and assignment commands in reactions and the GeneralViewAction. See Section 3.8.2.13, “ExpressionParser Configuration” for detailed information about configuring the ExpressionParser and compatibility issues with older implementations.

4.10.2. Syntax

4.10.2.1. Introduction

This section describes the valid syntax for expressions. In general, there are two different types of expressions: assignments and rules.

4.10.2.2. Assignments

Assignments are of the form *`[Name] [Assignment Operator] [Expression]`*.

Examples for valid assigments are:

Example 4.47. Examples for valid assignments

```
view.components.firstname.data = "Franz"
portal.properties.values += ["one", "two", "three"]
portal.properties.values -= data.object.unwanted
data.object.number = 18 + (23 - portal.vars.timestamp * 42)
```

4.10.2.3. Assignment Operators

Assignment operators are used in Assignments and define how the lefthand-side property is to be modified.

Table 4.161. Assignment Operators

Operator	Description
=	Assign the right hand side value(s) to the left hand side property.
+=	Add the right hand side value(s) to the multivalue left hand side property.
-=	Remove the right hand side value(s) from the multivalue left hand side property.

4.10.2.4. Expressions

An Expression is any legal combination of Operations, Constants, Literals, Names and Functions. Expressions may NOT contain Assignment Operators.

4.10.2.5. Rules

Rules are basically boolean Expressions (Expressions that evaluate to `true` or `false`) and can be used as rules for boolean settings or datasource filters.

4.10.2.6. Binary Operations

Binary Operations define how operators (left hand side [lhs] operator and righ hand side [rhs] operator) shall be combined to calculate other values. The supported values of the operators and the generated value depend on the Operator and are listed below.

Table 4.162. Binary Operations

Operator	Supported operator types	Result type	Description
OR	boolean	boolean	Logical or: is `true` when at least one of the operands is `true`. Can also be written as `||`.
AND	boolean	boolean	Logical and: is `true` when both operands are `true`. Can also be written as *`&&`*.

Operator	Supported operator types	Result type	Description
==	any	boolean	Equality comparison: is `true` when the operand values are equal.
!=	any	boolean	Unequality comparison: is `true` when the operand values are not equal.
>	numbers	boolean	"Greater" comparison: is `true` when the numerical value of the lhs operator is greater than the numerical value of the rhs operator.
>=	numbers	boolean	"Greater or equal" comparison: is `true` when the numerical value of the lhs operator is greater or equal than the numerical value of the rhs operator.
<	numbers	boolean	"Smaller" comparison: is `true` when the numerical value of the lhs operator is smaller than the numerical value of the rhs operator.
<=	numbers	boolean	"Smaller or equal" comparison: is `true` when the numerical value of the lhs operator is smaller or equal than the numerical value of the rhs operator.
LIKE	strings	boolean	"Like" operation: the result is `true`, when the rhs value matches the pattern of the lhs value. The lhs value might contain the character `%` as wildcard for any character sequence.
CONTAINSONEOF	arrays	boolean	"Contains one of": the result is `true`, when the lhs values contain at least one of the values of the rhs array.
CONTAINSNONE	arrays	boolean	"Contains none of": the result is `true`, when the lhs values contain none of the values of the rhs array.
CONTAINSALL	arrays	boolean	"Contains all": the result is `true`, when the lhs values contain all of the values of the rhs array. Note: currently, this operator can only be used with static values (i.e. it is not possible to filter a datasource with a rule like "object.attribute CONTAINSALL ['a', 'b', 'c']")
+	numbers	number	"Summation" operation: the result is the sum of the operands values.
-	numbers	number	"Subtraction" operation: the result is the difference of the operands values.
*	numbers	number	"Multiplication" operation: the result is the product of the operands values.
/	numbers	number	"Division" operation: the result is the quotient of the operands values. This will fail when the rhs value is 0.
%	numbers	number	"Modulus" operation: the result is the modulus of the integer division of the operands. This will fail when the rhs value is 0.

4.10.2.7. Unary Operators

Unary Operations modify values of a single operand.

Table 4.163. Unary Operations

Operator	Supported operator type	Result type	Description
+	number	number	This operation does not modify the value (is just for the sake of completeness).
-	number	number	Modifies the sign of the numerival value.
!	boolean	boolean	Logical "Not" operation: Switches `true` and `false`.

4.10.2.8. Constants

There are three constants: `true`, `false` and `null` (empty value).

4.10.2.9. Literals

There exist four types of literals: integer numbers, floating point numbers, strings and arrays.

Table 4.164. Literals

Type	Description
Integer	Integer literals can be notated in decimal form (starting with a number, but not with 0), in hexadecimal notation (starting with `0x`) or in octal notation (starting with `0`).
Floating Point Number	Floating point numbers can be given in the technical notation, including signs and exponents (see examples below).
String	Strings have to be enclosed by `"` (double quotes) or `'` (single quotes). The special characters enclosing character (double or single quote) and backslash (`/`) have to be escaped by backslash. Newline is written as `\n` and tabulator as `\t`.
Array	Arrays are notated as `[value1, value2, ...]`, where the values may be literals (inluding nested arrays) or constants.

Example 4.48. Examples of literals in expressions

```
42 (decimal integer)
-99 (signed decimal integer)
0xff (hexadecimal integer, decimal value: 255)
010 (octal integer, decimal value: 8)
-18.98e+1 (floating point integer, value: -189.8)
31.415926e-1 (floating point integer, value: 3.1415926)
"Franz" (string literal)
'Sepp' (string literal)
"\"'\n\t" (string literal, containing escaped characters)
["Franz", 42, true] (array)
[['Sepp', 1.95], ['Franz', 1.84]] (nested arrays)
```

Warning

There is an incompatibility between the old RuleParser and the ExpressionParser: The ExpressionParser is more strict in the notation of string literals, they always have to be enclosed by double or single quotes, whereas the RuleParser also interprets names that did not contain a `.` (dot) as strings.

Example 4.49. Expressions that are interpreted differently by RuleParser and ExpressionParser

```
portal.user.roles CONTAINSONEOF admin
```

was a legal rule for the old RuleParser, but would be interpreted differently in the ExpressionParser (`admin` will be interpreted as name and will most likely resolve to `null`). This rule should be rewritten into

```
portal.user.roles CONTAINSONEOF "admin"
```

or - more correctly -

```
portal.user.roles CONTAINSONEOF ["admin"]
```

4.10.2.10. Names and Variables

All character sequences that are not operators, literals, constants or functions (see below) and for which the rules for Java identifiers apply, are interpreted as names. Sequences of names that are only separated by `.` (dots) are interpreted as name-paths. When expression containing names are evaluated, name-paths are resolved into their current values (which might be null). Special name-paths starting with `object.` are interpreted as variables. When the expression is used as filter for datasource queries, the variables are placeholders for attributes of the filtered objects and their properties. For details on Java identifiers, see the Javadoc for Character.isJavaIdentifierPart() [http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Character.html#isJavaIdentifierPart(char)] .

Example 4.50. Datasource filter with variable

The datasource filter

```
object.name LIKE "N*"
```

would filter all objects, that have an attribute *`name`* with a value starting with `N` .

4.10.2.11. Functions

Functions are Names followed by parenthesis `()` . Functions might have function parameters that can be Literals, Constants, Name-paths, Functions or even Expressions.

Table 4.165. ExpressionParser functions

Name	Parameters	Result value	Description
concat(string, string, ...)	2 or more	string	Concatenates the given strings.
isempty(object)	1	boolean	Returns `true` when the object's value is null, an empty string or an empty Collection or Array.
subrule(attribute, subrule)	2	array	This function can only be used in datasource filters for datasources of type `contentrepository`. It generates a subquery and re-

Name	Parameters	Result value	Description
			turns the given attribute from all filtered objects. Note that for compatibility reasons, the subrule must contain variables as `subobject` instead of `object` .
if(expression1, expression2[, expression3])	2 or 3	any	Evaluates *`expression2`* when *`expression1`* is `true` or *`expression3`* when *`expression1`* is `false` and 3 parameters where given.
do(assignment1[, assignment2[, assignment3]...])	1 or more	assignment	Performs all given assignments in the given order. This function can be nested within the if() function to perform a sequence of assignments, when the given expression evaluates to `true` .
fromArray(array, index)	2	any	Fetches the object with index *`index`* from the given *`array`* .

matches(object, expression)	2	boolean	The matches() function implements a special kind of permission rule (for Gentics .Node ContentRepository datasource filters and evaluation). The function is used like:
			
			`matches(portal.user.permission, object.location CONTAINSONEOF this.location && object.region CONTAINSONEOF this.region)`
			
			Meaning: filter all objects that matches at least one of the given user permissions (portal.user.permission).
			`portal.user.permission` is supposed to be a collection of objects with attributes `location` and `region` each. An object matches the user's permissions if it matches the given rule agains one of those permission objects that are referred to as `this` in the rule. This is different from the rule
			
			`object.location CONTAINSONEOF portal.user.permission.location && object.region CONTAINSONEOF portal.user.permission.region`
			
			because in this rule, it is not garantueed that the filtered object matches location and region for the SAME permissions object.

eval(expression)	1	any	The eval() function takes the given expression, evaluates it and then interprets the result as part of the outer expression. This can for example be used for expressions that are configured in properties: The expression `eval(portal.properties.rule)` would first resolve the property *`portal.properties.rule`* (e.g. to `portal.user.isloggedin`) and then evaluate this as rule.
insert(array, newob-	2 - 4	array	The insert() function can be used to insert an

Name	Parameters	Result value	Description
ject[, index[, unique]])			object to a given array (or collection) at the specified position. The optional parameter *`index`* defines the position where to insert the *`newobject`* into the given *`array`* , where `0` would be the start and `-1` the end (which is the default). When the parameter *`unique`* is set to `true` (default is `false`), the inserted object is removed from the array first (if contained), thus ensuring the uniqueness of the object in the array. It is important to notice, that this function does not modify the *`array`* itself, but merely returns a copy with the requested modifications done.
get(object, attributeName)	2	any	The get method allows you to get a given attributeName from the object. This can be used, when the attributeName itself is not static, but is itself resolved or constructed by using functions.
set(object, attributeName, value)	3	none	The set method allows you to set a given attributeName to a defined value for the given object.
foreach(Map|Collection|array, iteratorName, loopBody)	3	list	Allows you to iterate over a Map, Collection or array which is stored in a property named as 'iteratorName' within the 'loopBody'. (loopBody can be any other function call, e.g. do()). This function returns a list of all loopBody results. `foreach(portal.pages, "page", page.reset = true)`
echo(Object)	1	None	Similar to the EchoAction the echo function is only suitable for debugging - it will force a info log output containing string representation of the passed in value.
i18n(string [,string])	1-2	string	Translate the first parameter using the dictionary. The second parameter is optional and specifies the language id. If not specified the current language will be used.
docallableaction("module Id", "view Id", "callable action id")	3	Boolean	Invokes the callable action with the given id which is defined in the given module id and view id. Returns true if action sequence was invoked successfully. See also Section 4.2.2.9, “Callable Actions”
getContentID(datasource, pub_dir, filename, [node_id])	3-4	String	Tries to convert the given input data (pub_dir, filename, optionally node_id) into the content-id by looking in the configured datasource. See also Section 5.4, “GenticsContentModule” for details on referencing content by pub_dir, filename and node_id.
filter(rule, "postprocessorclass"[, data])	2-3	Rule	The `filter()` function can only be used in datasource filters. When used it will create an instance of the class *`postprocessorclass`* and call the `process()` method with the objects that were returned by the *`rule`*.

Tip

The *`if()`* function can be used for conditional assignments, since *`expression2`* and *`expression3`* may also be assignments!

Example 4.51. Example for conditional assignments

```
if(portal.user.isloggedin, data.value = "yes", data.value = "no")
  or
data.value = if(portal.user.isloggedin, "yes", "no")
```

Warning

The *`if()`* function can not be used in datasource filters.

Warning

The *`eval()`* function can only be used in a limited way in datasource filters: currently the result of the inner expression will be evaluated first and then inserted into the outer expression. Therefore the result of the *`eval()`* function must not contain something like `object.` .

Tip

The *`filter()`* function can be used to replace filtering rules, which would result in complex SQL statements (like e.g. filtering for multivalue attributes) with a Java implementation.

Example 4.52. Example for using the filter() function

Let's assume the following rule to get pages from a folder that match the user's permissions groups:

```
object.obj_type == 10007 AND object.folder_id = data.folder_id
  AND object.permgroups_view CONTAINSONEOF portal.user.groups
```

Since the attribute *`permgroups_view`* is a multivalue attribute (and therefore not optimizable), the resulting SQL statement will contain at least one join.

By moving the logic of filtering by permission groups into an instance of PostProcessor and using the *`filter()`* function, the complex SQL statement can be avoided.

Example implementation of the PostProcessor

```
package com.example;
import java.util.Collection;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
```

```
import com.gentics.api.lib.etc.ObjectTransformer;
import com.gentics.api.lib.expressionparser.EvaluationException;
import com.gentics.api.lib.expressionparser.ExpressionEvaluator;
import com.gentics.api.lib.expressionparser.filtergenerator.PostProcessor;
import com.gentics.api.lib.resolving.Resolvable;

/**
 * Permission filter
 */
public class PermissionFilter implements PostProcessor {
  /**
   * Process the objects
   * @param resolvables collection of resolvables
   * @param data data object (containing the user permissions)
   */
  public void process(List<Resolvable> resolvables, Object data)
      throws EvaluationException {
    // get the user permissions, which are provided to the filter() function
    Collection userPerms = ObjectTransformer.getCollection(data, Collections.EMPTY_LIST

    // iterate over all resolvables
    for (Iterator<Resolvable> i = resolvables.iterator(); i.hasNext();) {
      Resolvable res = i.next();
      // get the object permissions
      Collection objPerms = ObjectTransformer.getCollection(
        res.get("permgroups_view"), Collections.EMPTY_LIST);

      // filter out the objects with non-matching permissions
      if (!ExpressionEvaluator.containsOneOf(objPerms, userPerms)) {
        i.remove();
      }
    }
  }
}
```

Example usage

```
filter(object.obj_type == 10007 AND object.folder_id = data.folder_id,
 'com.example.PermissionFilter', portal.user.groups)
```

4.10.2.12. Evaluation priority

The evaluation priority of different expression parts can be seen in the following table (operations with highest priority listed on top):

1. Constants, Literals, Names and Functions
2. Unary operations
3. Multiplicative calculation operations
4. Additive calculation operations
5. Comparison operations
6. Boolean operations
7. Assignments

Sequences of operations with the same priority level are evaluated from right to left. It is legal to use par-

enthesis within expressions to modify the evaluation order.

4.11. Templates

4.11.1. Portaltemplate

4.11.1.1. Introduction

The portaltemplate defines the basic layout of the portal page and the portlets that are rendered in the portal page. In most cases, the portaltemplate is a HTML page that is extended by special Gentics Portal.Node3 XML tags. The portaltemplate will be loaded at the start of the portal's webapplication. If changes have been made in the portaltemplate the portal's webapplication has to be restarted.

4.11.1.2. pportal Tag

The pportal tag contains the styledefintion of the frames that encapsulate the content of the different portlets.

Example 4.53. Syntax of the pportal tag

```
<pportal>
  <modulestyles default="...">
    <modulestyle id="..." actionstyle="...">
      <framestyle>
        <windowstate>...</windowstate>
        ...
        <frame>...</frame>
      </framestyle>
      ...
    </modulestyle>
    ...
  </modulestyles>
  <actionstyles>
    <actionstyle name="...">
      <actiontemplate>...</actiontemplate>
      <actionseperator>...</actionseperator>

      <action type="...">
        <icon>...</icon>
        <alt>...</alt>
      </action>
      ...
    </actionstyle>
   </actionstyles>
</pportal>
```

Table 4.166. Parameters of the pportal-tag

Name	Type	Description
modulestyles	Structure	Definition of the modulestyles.
module-styles.default	String	Id of the default modulestyle. If not set, the first defined modulestyle will be the default.
modulestyle	Structure	Definition of a single modulestyle
modulestyle.id	String	Id of the modulestyle.
module-	String	Id of the actionstyle to be used with this modulestyle.

Name	Type	Description
style.actionstyle		
framestyle	Structure	Definition of how a portlet frame shall be rendered for specific windowstates. Each modulestyle may contain multiple framestyles (for different windowstates).
framestyle.windowstate	String	Windowstate for which this framestyle shall be used. May be one of `maximized`, `normal` or `minimized`. The same framestyle may be used for multiple windowstates (more than one `<windowstate>`-Node in the framestyle definition).
framestyle.frame	String	Template for the portlet frame. May contain the following placeholders: • `<box>` content of the portlet window that is rendered by the portlet itself. In windowstate `minimized`, no portlet content will be rendered at all. • `<boxtitle>` title of the portlet, eventually rendered as >a href=""< when a titleaction is defined for the portlet. • `<boxaction>` all available boxactions. These include icons for changing the portletmode and windowstate, when they are not rendered with specific placeholders described below. • `<windowstates>` boxactions for switching the windowstates. When this placeholder is not given, the icons for switching the windowstates are rendered as part of <boxaction>. • `<portletmodes>` boxactions for switching the portletmodes. When this placeholder is not given, the icons for switching the portletmodes are rendered as part of <boxaction>. • `<moduleid>` id of the portlet.
actionstyles	Structure	Definition of how icons for window actions are rendered. The window actions include icons for switching windowstates and portletmodes.
actionstyle	Structure	Definition of a single actionstyle.
actionstyle.name	String	Id of the actionstyle.
actiontemplate	String	Template for an icon for a window action. May contain the following placeholders: • `<icon>` name of the action icon as defined below. • `<url>` url to activate the window action. • `<alt>` alt-text (description) of the window action as defined below. • `<moduleid>` id of the portlet.
actionseperator	String	Template for the separator to be rendered between action icons.
action	Structure	Definition for a specific portlet window action.
action.type	String	Definition of the action for which this style shall be used.

Name	Type	Description
		This may be `portletmode.[mode]` for the portletmode `[mode]`, `windowstate.[state]` for the windowstate `[state]` or the name of a specific defined portlet action (without the prefix `on`).
action.icon	String	Definition of the action *`icon`* that will be rendered into `<icon>` from the actiontemplate.
action.alt	String	Descriptive alt-text of the action.

4.11.1.2.1. Defining Custom Actions

To define a custom action which can be used for a pnode/pbox and be catched in a reaction you need to follow the following three simple steps:

1. Create a new action in the actionstyles / actionstyle tag:

```
<action type="CustomAction">
  <icon>edit</icon> <!-- Select a custom icon here -->
  <alt>New Custom Action</alt>
</action>
```

2. Add the action to your pnode/pbox:

```
<action name="CustomAction">
  <visible>true</visible>
</action>
```

3. Use a reaction in your pnode/pbox to catch this action:

```
<reaction event="portal.pboxes.SimpleViewTest.onCustomAction">
  <!-- do whatever you want here ... -->
</reaction>
```

4.11.1.3. PNodes

The portaltemplate may contain <pnode> tags, each one defining an instance of a portlet with optional parameters and customized templates. Each <pnode> implicitly defines also a `pbox` with the same id. For details on pboxes see Section 4.11.1.4, “PBoxes” .

Example 4.54. Syntax of a pnode tag

```
<pnode type="..." id="...">
  <style>...</style>
  <isvisible>[<prule>...</prule>]</isvisible>
  <windowstate>...</windowstate>
  <title>...</title>
  <titleaction>...</titleaction>
  <actions>
    <action name="...">
```

```
      <visible>[<prule>...</prule>]</visible>
    </action>
    ...
  </actions>
  <reactions>
    <reaction event="...">
      ...
    </reaction>
    ...
  </reactions>
  <parameters>
    <parameter name="...">[<pobject>...</pobject>]</parameter>
    ...
  </parameters>
  <templates>
  </templates>
</pnode>
```

Table 4.167. Parameters of the pnode tag

Name	Type	Default	Description
pnode	Structure		Definition of a portlet instance.
pnode.type	String	(required)	*`portlet-name`* of the portlet to be instantiated.
pnode.id	String	(required)	Id of the portlet instance. Must be unique within all <pnode>s and <pbox>es in this portalpage.
style	String	default modulestyle. See Section 4.11.1.2, “pportal Tag” .	Modulestyle of the portletinstance.
isvisible	Boolean or PRule	true	Whether the implicit pbox shall be visible or not.
windowstate	String	normal	Initial windowstate in which the pbox shall be rendered.
title	String	*`title`* of the portlet defined in the portlet descriptor.	Title of the portlet.
titleaction	String	NULL	Action to be activated upon link on the portlets title.
actions	Structure	NULL	Definition of the portlet window actions.
action	Structure	NULL	Definition of a portlet window action. This may also override the default actions for switching portletmodes and windowstates.
action.name	String	(required)	Name of the portlet window action. See Section 4.11.1.2, “pportal Tag” for details on action names for the standard actions. Custom actions will trigger an event called `portal.pboxes.[pboxid].on[ActionName]` when activated.
action.visible	Boolean or PRule	true	Whether the action icon shall be visible or not.
reactions	Structure	NULL	Definitions of reactions.
reaction	Structure	NULL	Definition of a reaction on a specific event. The node value may consist of one or more lines, each one setting a portal property. See Section 4.7, “Property Setters” for de-

Name	Type	Default	Description
			tails - reaction property setter have the limitation that you must not use quotes (").
reaction.event	String	(required)	Full path of the event this reaction shall listen to. The path begins with • `portal.modules.[moduleid]` for portlet events. `portal.pboxes.[pboxid]` for pbox events.
parameters	Structure	NULL	Definition of parameters for the portlet instance.
parameter	various	NULL	Definition of a specific parameter. The allowed values depend on the parameter definition in the Gentics portlet descriptor. You can use <pobject> to access portal properties, see Section 4.8, “Portal Property Paths” for more details.
parameter.name	String	(required)	Name of the parameter to set.
templates	Structure	NULL	Definition of specific templates for this portlet instance. See Section 3.8.7.4, “Template definitions” for details on definition of templates.

4.11.1.4. PBoxes

The portaltemplate may contain <pbox> tags, each one defining a place where a portlet may be rendered. The difference between a <pnode> tag and a <pbox> tag is that with a <pbox> does not create a new instance of a portlet.

Example 4.55. Syntax of a pbox tag

```
<pbox id="...">
  <moduleid>...</moduleid>
  <windowstate>...</windowstate>
  <isvisible>[<prule>...</prule>]</isvisible>
</pbox>
```

Table 4.168. Parameters of the pbox tag

Name	Type	Default	Description
pbox	Structure		Definition of a place in the portal page where a portlet may be rendered.
pbox.id	String	(required)	Id of the pbox. Must be unique within all pnodes and pboxes of the portal page.
moduleid	String	NULL	Id of a portlet that shall be initially loaded in-

Name	Type	Default	Description
			to this pbox.
isvisible	Boolean or PRule	true	Whether the pbox shall be visible or not.
windowstate	String	normal	Initial windowstate in which the pbox shall be rendered.

4.11.2. Portlet Templates

Portlet templates are specified in the Gentics Portlet Descriptor in the <templates> section of a specific portlet or may be overridden in the <pnode> tag in the Portal Template. The loading order is as follows. The first matching is taken.

- PNode Template. See Section 4.11.1.3, “PNodes”
- Portlet Template. See Section 4.4.4, “Gentics Portletdescriptor”
- Portal Template. See Section 3.8.7, “template-section”

4.11.3. Plugin Templates

Plugin templates are specified in the Gentics Portlet Descriptor in the <templates> section of a specific plugin or may be overridden in the <pnode> tag in the Portal Template. The loading order is as follows. The first matching is taken.

- PNode Template. See Section 4.11.1.3, “PNodes”
- Portlet Template. See Section 4.4.4, “Gentics Portletdescriptor”
- Plugin Template. See Section 4.4.4, “Gentics Portletdescriptor”
- Portal Template. See Section 3.8.7, “template-section”

4.11.4. TemplateEngine2

4.11.4.1. Introduction

Velocity based template engine, with helper objects embedded.

4.11.4.2. Velocity

Velocity is a simple but powerful template language which is part of the Apache Jakarta Project. For full documentation see: http://velocity.apache.org/.

VTL (Velocity Template Language) has a few predefined language constructs - the most important are listed here:

- Comments

  Single line comments using two hashes: ## This is a comment

  Multiline comments using #* this is a comment *#

- References - Indicated using a $ - e.g. $myvar

There are three types of References in the VTL: variables, properties and methods. Variables are objects which are in the Velocity Context, e.g. using #set($myvar = "test") would set the Variable $myvar

Properties look like: $myobj.myproperty - Velocity will first try to call the java method .get("myproperty") on the specified object, if it is not found it will try to call .getMyproperty().

Methods can simply be called using $myobj.myMethod() which would call the method specified and returns the object of the method.

- Directives and Velocimacros

  To call predefined VTL Directives simply use a hash followed by it's name. e.g.: #set($var = "value"), #if($var == "$value") #else #end, #foreach($var IN $collection) #end, etc.

  Velocimacros are userdefined directives which look very similar in it's syntax but are defined in a VTL file using: #macro(macroName $argName1 $argName2) #end - they are then called using #macroName($argValue1 $argValue2)

Note

The Velocity Template Language is very powerful and most is far beyond the scope of this short Introduction. So if you have never used Velocity before you should head over to the Velocity User Guide [http://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html] to get more in-depth documentation.

4.11.4.3. Context

The templateengine's context contains usefull objects, which can be addressed through Section 4.8, “Portal Property Paths” .

$portal - the portal object, including imps and i18n.

$javax.portlet - portlet information, including context, config, request and response.

$portlet - the portlet rendering the template (may be null if the template is not rendered by a portlet)

$plugin - the plugin rendering the template (may be null if the template is not rendered by a plugin)

$position - the current position of the portlet (may be null if the template is not rendered by a portlet)

$ctx - the velocity context

4.12. Language

4.12.1. Introduction

Support for multilanguage portals, with a customizable dictionary. Languages can be freely defined and associated to locales for e.g. language specific date formatting.

4.12.2. Language Handling

This chapter will help you to understand how to handle Gentics Portal.Node's language features. See Section 3.8.5, “language-section” to learn how multilanguage support is configured. In views the current language can be read or set using `portal.language.id`.

Example 4.56. Setting the current language

Here's a simple example which depicts how to set the portal's language to German, assuming that the id of "German" is configured to be "de". Also see the Gentics Portal.Node SDK's SamplePortletapplication/LanguageSelector portlet for a working real-life example.

```
<view>
  ...
    <action class="GeneralViewAction">
      <parameters>
        <parameter id="set">portal.language.id="de"</parameter>
      </parameters>
    </action>
  ...
</view>
```

In templates, you can either use `$portal.i18n("key")` or the “I18nImp” (Section 4.1.11, p.95). For setting parameters in the translation, you can use the `setParameter()` method:

```
## welcome is translated into "Welcome $name!"
#set($message = $portal.i18n("welcome"))
$message.setParameter("name", $portal.user.lastname)
## this will output something like "Welcome Mustermann!"
$message
```

Keep in mind that you will need to maintain a dictonary for every language configured, as described in Section 3.8.5.3, “Dictionaries”

4.13. Datasource

This chapter covers implementation-oriented topics on datasources. See Section 3.4, “Datasources” to learn how to configure your datasources.

Datasources provide an easy and abstracted way to query and manage data, independent of the specific backend system using a common query language (expressions). Currently 2 types of datasource implementation exist: Gentics .Node ContentRepository datasources are often called enhanced relational databases, are stored in SQL92 compliant database systems in a fixed database layout, and bring several advantages against common relational SQL layouts. LDAP datasources are stored in LDAPv3 compliant directories.

See the Gentics Infoportal for a detailed list of database products which were successfully tested for compatibility.

Please also consult the Gentics Infoportal for a list of limitations in functionality for different datasources and/or datastore products.

4.13.1. ContentRepository

4.13.1.1. typeid

The typeid of a datasource of type ContentRepository is `contentrepository`

4.13.1.2. Primary Key

The primary key for Gentics .Node ContentRepository datasources is "contentid".

4.13.1.3. Advantages

- Versioning (store, read) datasets
- Multivalue attributes
- Runtime datastructure changes
- Simplified relational queries

4.13.1.4. Database scheme

A generic database scheme, used for storing content published by Gentics Content.Node;, all standard modules provided by Gentics and the Gentics .Node PortalConnector. The scheme is independent of your data structure, and must not be adapted for new attributes. See “ContentRepository” (Section 3.4.3, p.24) for more information.

The following datatypes are supported:

Table 4.169. Attribute Types

Type Id	Type	Java Representation	Type Description
1	Text Short	String	A short text (maximum length 255 characters)
2	Object Link	Resolvable	Link to another object (the 1:n relation, the id to the linkedobject is stored in this attribute)
3	Integer	String	A normal-sized Integer (signed 32 bit integer). Will be returned as String for backward compatibility reasons.
5	Text Long	String	For example used for Page content in Gentics Content.Node; (Replaced the deprecated type 4)
6	Binary Content	byte array	Binary data (e.g. files)
7	Foreign Link	Collection of Resolvables	Reverse attribute to type 2. the n:1 relation. this attribute contains no data , it just points to the linkattribute of the foreignobject.
8	Long Integer	Long	A large integer (signed 64 bit integer).
9	Double	Double	A normal-size (double-precision) floating-point number.
10	Date	Timestamp	An attributetype which can hold a date and time.

Note

If you retrieve a Multivalue attribute the result will be a Collection containing all values as described above (Except Object Links which will be retruend as `Collection` of `Resolvables`).

4.13.2. Multichannelling ContentRepository

4.13.2.1. typeid

The typeid of a datasource of type ContentRepository is `mccr`

4.13.2.2. Primary Key

The Primary Key for MCCR datasources is "contentid".

4.13.2.3. Channel selection

The main difference between a Multichannelling ContentRepository and a ContentRepository is that when using a Multichannelling ContentRepository, the channel (or channels) to be used have to be selected.

Selecting channels can be done for each separate channel structure, that is published into the Multichannelling ContentRepository. A channel structure is a master node together with all its channels.

Initially, for each channel structure, the master node is selected.

The channel can be set using the mccr specific portal property path *`portal.datasources.[id].channel = [channelid]`*. (See “Portal” (Section 4.8.1, p.175) portal property paths). Selecting a channel using the portal property path will store this setting in the user session. Every datasource instance fetched by a portlet will now have the modified set of channels selected.

Alternatively, the channels may be set on a datasource instance, which will only affect this instance, e.g. like documented in “DatasourceQueryImp for MCCR datasources” (Section 4.1.7.3, p.92).

4.13.2.4. Meta Attributes

Meta Attributes are always present for all objects and can be used in filter rules.

Table 4.170. Multichannelling ContentRepository Meta Attributes

Name	Type	Description	
obj_type	Integer	Type of the object: 10002 for folders, 10007 for pages and 10008 for files	
obj_id	Integer	Object ID	
contentid	String	Composition of "[obj_type].[obj_id]"	
channel_id	Integer	Channel ID of the object	
channelset_id	Integer	Channelset ID of the object. The channelset_id is a unique identifier of an object in a channel. Objects in different channels with the same channelset_id are channel-specific variants of the same object.	
updatetimestamp	Integer	Unix Timestamp of the last update to the object	

4.13.3. LDAP

4.13.3.1. typeid

The typeid of a datasource of type ContentRepository is `ldap`

4.13.3.2. Primary Key

The primary key for LDAP datasources is "dn".

4.13.3.3. Limitations

Gentics .Node ContentRepository specific datatypes like LinkedObject are not supported.

Writing is currently not supported.

Limited filter functionality in queries.

4.14. Portal Pages Implementation

The use of Portal Pages provide an alternative to the single portal template file for implementing portals. Every request to the portal will render exactly one portal page which aggregates a set of portlets that are placed in so-called "positions" on that page.

Positions allow grouping portlets for layout purposes. Typically, every position on a portal page will consume a specific part of the page (e.g. the left column, the right column and the focus in the center of the page).

Portal Pages can be used for several purposes:

- Provide multiple different variants of presentation. For example one could define two pages, one with two columns, one with three.
- Thematically distribute the portlets among several pages (e.g. one page for content presentation, one for administration tasks, etc.)
- Serve different virtual portals using the same infrastructure.

From the administrative point of view, portal pages allow a strict separation between portlet definition (see “Portlet Entities Configuration” (Section 3.13, p.78)), portlet aggregation (see “pages-section” (Section 3.8.9, p.60)) and the layout, which is defined in templates for the Template Engine2.

Last but not least, the complete rendering of a portal page is now based on the Template Engine2, which significantly improves the rendering performance.

4.14.1. Portal Page Templates

The portal page templates define the layout of portal pages and portlet frames. By classification of the templates, it is possible to have portlets be rendered differently, depending on the page, position, windowstate, portletmode and some other parameters.

Example 4.57. Portal Page Template Classification

```
<classes>
  <class id="type">portalpage</class>
  <class id="portalpage">...</class>
</classes>
```

Table 4.171. Portal Page Template Classes

Class	Required	Description
type	x	Must be set to "portalpage".
portalpage		The portal page id.

Table 4.172. Template Variables for Portal Page Templates

Name	Type	Description
$portal.pages	List	List of all visible portal pages
$page in $portal.pages	Object	A specific portal page
$page.id	String	Id of the portal page
$page.positions	List	Portlet positions of the portal page.
$page.positions.[positionid]	Object	Portlet position with the id `[positionid]`.
$page.positions.[positionid].portlets	List	Ordered list of portlets currently in the position.
$portlet in $page.positions.[positionid].portlets	String	Renders the portlet with frame.
$portlet.*	(various)	Provides access to the properties as defined in the context for the portlet frame, see “Portlet Frame Templates” (Section 4.14.2, p.202) .
$page.properties	List	Properties of the page as defined in the portal pages configuration.
$page.switchUrl	String	URL to switch to this portal page (makes it the current portal page). Will fire a portalevent, see “Portal Page events” (Section 4.3.6, p.165) for details.
$page.renderUrl	String	URL to render this portal page, without making it the current portal page. This URL can be used e.g. for rendering different portal pages in popups.
$portal.page	Object	Current portal page (which must not necessarily be the currently rendered portal page), provides access to further attribute of the current portal page (see above).
$page	Object	Currently rendered portal page, provides access to further attributes like $page.positions, ... (see above).

4.14.2. Portlet Frame Templates

Portlet frame templates define, how the frame around portlet content will look like. Typically the portlet frame templates will contain markup to render the portlet title, portlet action icons and everything else surrounding the portlet content.

In some special situations, it is also necessary to render markup for portlets, which are not visible (but nevertheless contained in a position). For those cases, it is possible to define portlet frames to be used for invisible portlets.

Example 4.58. Portlet Frame Template Classification for visible portlets

```
<classes>
  <class id="type">portletframe</class>
  <class id="portlet">...</class>
  <class id="portalpage">...</class>
  <class id="windowstate">...</class>
  <class id="style">...</class>
  <class id="position">...</class>
</classes>
```

Example 4.59. Portlet Frame Template Classification for invisible portlets

```
<classes>
  <class id="type">invisibleportletframe</class>
  <class id="portlet">...</class>
  <class id="portalpage">...</class>
  <class id="windowstate">...</class>
  <class id="style">...</class>
  <class id="position">...</class>
</classes>
```

Table 4.173. Portlet Frame Template Classes

Class	Required	Description
type	x	Must be set to "portletframe" (or "invisibleportletframe").
portlet		Id of the portlet.
portalpage		Id of the portalpage.
windowstate		Windowstate of the rendered portlet.
style		Style of the portlet (configured in the pnode in the portaltemplate).
position		Id of the position.

When a portlet is rendered in a position of a portal page, the portlet frame is rendered with the best matching template. The optional classification items are considered with the following priority:

1. portlet
2. portalpage
3. windowstate
4. style
5. position

A template matching a specific item will always be preferred to other templates matching any number of items with lower priority.

Table 4.174. Template Variables for Portlet Frame Templates

Name	Type	Description
$portlet	Object	The portlet object. Provides access to portlet data.
$portlet.id	String	Id of the portlet.
$portlet.title	String	Title of the portlet as defined in the portlet descriptor. The title will not be internationalized automatically. If you need i18n support for the portlet title, use `$portal.i18n($portlet.title)` and add the translated portlet titles to your dictionaries.
$portlet.content	String	The rendered portlet content, or empty, when the portlet is not visible.
$portlet.titleaction	Object	The titleaction, if one was defined.
$portlet.actions	Collection	Collection of portletactions
$action in $portlet.actions	Object	A portletaction object
$action.id	String	Id of the action. Is either `windowstate.*` (for windowstate actions) or `portletmode.*` (for portletmode actions) or the defined id.
$action.portletmode	String	The portletmode for portletmode switching actions, `null` for other actions.
$action.windowstate	String	The windowstate for windowstate switching actions, `null` for other actions.
$action.url	String	URL to activate the action.
$action.properties.[property id]	String	Property as it was configured either in the pnode or in the defaultactions.
$action.alt, $action.icon	String	Deprecated elements configured using <icon> tag - This should be replaced by properties.
$portlet.parameters	Map	Portlet parameters.
$portlet.parameters.[name]	(various)	Value of the portlet parameter `name` or `null` if the parameter is not defined or has no value.
$portlet.properties	Map	Portlet properties
$portlet.properties.[name]	(various)	Value of the portlet property `name` or `null` if the property is not defined or has no value.
$position.id	String	Id ofthe currently rendered portletposition.
$position.index	Integer	Index of the portlet in the position, starts with 0.

4.14.3. Customizing portlet positions

This section describes, how portal implementations can allow users to customize the positioning of portlets on portal pages.

4.14.3.1. Customizing portlet positions by property setting

It is possible to modify the portlet positions on a portal page by setting various properties. The property path ***`portal.page`*** provides access to the current portal page and via ***`portal.pages.[pageid]`*** it is possible to access and modify properties of any page by its ***`pageid`***. For a page object, with the property ***`positions.[positionid]`***, one can access the position ***`positionid`*** (if that position exists on the page), and with ***`positions.[positionid].portlets`*** one will get the list of portlets cur-

rently contained in the referenced position.

Modification of the portlet positions can be done in one of the following ways (the following code applies for the current portal page, but can easily be changed to modify an arbitrary portal page):

- `portal.page.positions.[positionid].portlets = insert(portal.page.positions.[positionid].portlets, [portletid], -1, true)` will insert the portlet [portletid] as last item in the position [positionid]. When the position already contains this portlet is is moved to the end. By using another number instead of `-1` as third parameter to the function `insert()` it is possible to insert the portlet at an arbitrary position (or moved it there).

  Note that it is necessary to assign the result of the `insert()` function to the position, since the function would not change the list, but return a modified copy of the list of portlets.

- `portal.page.positions.[positionid].portlets -= [portletid]` will remove the portlet `portletid` from the position.

- `portal.page.reset = true` will reset the page to the configured portlet positions.

  `foreach(portal.pages, "page", page.reset = "true")` will reset all pages to the configured portlet positions.

See “Portal Property Paths” (Section 4.8, p.175) for a full list of all available portal property paths.

4.14.3.2. Customizing portlet positions by URL

It is possible to customize the portlets currently contained in a position of the current portal page by sending a special URL to the portal server. The base URL is provided as `$portal.portletpositionchange` in every template. This URL can be used to change portlet positions by AJAX Drag'n'Drop. See “Portlet Drag & Drop” (Section 4.6.3.4, p.175) for details and the SDK Demo Portal for an example.

4.14.3.2.1. Request

To store the new portlet positions, call the `$portal.portletpositionchange` URL and supply the following parameters:

Table 4.175. Portlet Position Customization URL Parameters

Name	Type	Description
gentics.rl	Boolean	True to return a JSON response as defined in Section 4.6.3.1, “Response Format of AJAX enabled requests API”, false to render the whole portal page as response.
positions	List	A list of portlet position IDs which will be updated by this request.
[portlet position ID]	List	The list of portlet IDs in the order in which they appear in this portlet position.

Example 4.60. Portlet Position Customization URL Example

```
/Portal.Node/portal?gentics.pa=portletpositionchange&gentics.pb=
```

```
&gentics.rl=true&positions=left&positions=right&left=pageselector
&left=helloworld&right=calculator&right=history
```

4.14.3.2.2. Response

The response to the portlet position customization URL depends on whether the AJAX parameter *gentics.rl* was set to `true` or `false`. When set to `false` (which is the default value), the current portal page is simply rendered again. When set to `true`, the response has the same format as the response to a portlet reload request (see Section 4.6.3.1, "Response Format of AJAX enabled requests API") and can contain portlet reload data, if portlets need to be reloaded as a result to their new position.

4.14.4. Persisting and restoring customized portlet positions

By setting the general portal page parameters *store* and *store.rule* (see Section 3.8.9.2, "General Parameters for portal pages"), the feature for persisting and restoring portlet positions can be activated.

When a user session is invalidated (for example, when the user logs out), the current portlet positions of all pages the user has visited (and possibly customized) will be persisted into the configured object.

When a user session is created, and persisted portlet position data is found in the configured object, the persisted data is merged with the portlet position setting configured in the portal configuration.

- Persisted data for portal pages, which are no longer present in the configuration are removed (removing of portal pages from the configuration is immediately effective for all users).
- Persisted data for positions in portal pages, that no longer have them configured are also removed (removing of positions from portal pages by configuration is immediately effective for all users).
- Persisted data for positions in portal pages that still exist, override the configured setting (which means that configuration changes will not be affective until the user resets his/her customized settings).

4.15. Portlet Specification 2.0 Support (JSR 286)

Although Gentics Portal.Node 4 is compiled against the API of the Portlet Specification 2.0 (JSR 286), it is important to notice that Gentics Portal.Node 4 is not yet fully JSR 286 compliant.

Currently, only the following features of the Portlet Specification 2.0 are supported in Gentics Portal.Node 4:

- Resource Serving
- Portlet Events
- Public Render Parameters

Please visit http://jcp.org/en/jsr/detail?id=286 for details about the Portlet Specification 2.0.

Note

The JSR 286 API does not break binary compatibility with JSR 168 compliant portlets. This means that Gentics Portal.Node 4 still fully supports JSR 168 compliant portlets and portlet applications with all features of the Portlet Specification 1.0.

Chapter 5. Portlets

5.1. Introduction

Gentics Portal.Node provides a set of portlets with default parameters and templates, ready to use by simply adding them to the Portal Template.

In prior versions of Gentics Portal.Node, Gentics portlets used to be called "modules" and some parts of this documentation or some configuration parameters may still refer to "modules" instead of "portlets".

5.2. All GenticsPortlets

5.2.1. Introduction

All GenticsPortlets support extended parameters, events and properties.

5.2.2. Events of all Gentics Portlets

Table 5.1. Events of the All Gentics Portlets

Name	Description	Event properties
onChanged	This event is triggered every time, a portletparameter changes.	**Name** | **Description** No event properties.
onParameter-Change		**Name** | **Description** No event properties.
onProper-tyChange		**Name** | **Description** No event properties.

5.2.3. Parameters of all Gentics Portlets

Table 5.2. Portlet parameters for All Gentics Portlets

Name	Type	Default	Description
gen-tics.portletreloa d	Boolean	NULL	Enable portlet reloading (see “Portlet Reloading” (Section 4.6.3.3, p.171)) for this portlet. This set-ting overwrites the global setting *portal.portletreload* .
gen-tics.portletreloa d.allclasses	Boolean	NULL	Enable portlet reloading (see “Portlet Reloading” (Section 4.6.3.3, p.171)) for all elements, regard-less of CSS class (see *gen-tics.portletreload.replaceclasses*). This setting overwrites the global setting *portal.portletreload.allclasses* .
gen-tics.portletreloa d.replaceclasse s	String	NULL	Enable portlet reloading (see “Portlet Reloading” (Section 4.6.3.3, p.171)) only for elements (links, forms) which have at least one of the specified CSS

Name	Type	Default	Description
			classes. List all CSS classes as a comma-separated list. This setting overwrites the global setting *portal.portletreload.replaceclasses* .
gentics.portletreload.replacelink	Boolean	NULL	Enable portlet reloading (see "Portlet Reloading" (Section 4.6.3.3, p.171)) for links (replace the default onclick action). This setting overwrites the global setting *portal.portletreload.replacelink* .
gentics.portletreload.replaceform	Boolean	NULL	Enable portlet reloading (see "Portlet Reloading" (Section 4.6.3.3, p.171)) for forms (replace the default submit action). This setting overwrites the global setting *portal.portletreload.replaceform* .
gentics.actionurls.appendtimestamp	Boolean	true	With this parameter set to `true` (which is the default), all action URLs generated by this portlet will automatically have a timestamp parameter appended, to prevent the browser from caching the response - and thus not performing the action request again - which might lead to unexpected behaviour.

5.3. ViewPortlet

5.3.1. Introduction

This portlet is most frequently used, although the portlet itself is pretty simple. It's used for any application based on the "ViewPlugin" (Section 4.2.2, p.97).

5.3.2. Parameters

Table 5.3. Portlet parameters for ViewPortlet

Name	Type	Default	Description
viewplugin.viewbasedir	String	NULL	Path to the directory that holds the views definition files for the portlet. The path may contain "System properties" (Section 3.2, p.20). It is strongly recommended to place the views in a directory somewhere under `WEB-INF/` of your portletapplication. See Example 5.1, "Example of views configuration for the ViewPortlet" and Section 4.2.2.4, "Parameters" for more details on the parameters of the viewplugin.
viewplugin.doubleclickprotection	Boolean	NULL	Parameter to overwrite the portal-wide setting for the double-click protection. The default is to not overwrite the portal-wide setting (which itself defaults to `true`). See "ViewPlugin Configuration" (Section 3.8.2.11, p.33) for details on the global setting.
datasource	String	NULL	Id of the default datasource to be used in this port-

Name	Type	Default	Description
			let. The datasource configured here will be used as default in every component that uses datasources.

Example 5.1. Example of views configuration for the ViewPortlet

- Place the views files in the directory `WEB-INF/[portletid]/views` of your portletapplication.
- Set the portlet parameter *`viewplugin.viewbasedir`* to `${com.gentics.portalnode.apps.[portletapplication].path}/WEB-INF/[portletid]/views` , where [portletapplication] is the context path of your portletapplication and [portletid] the id of your portlet.

5.3.3. Events of the ViewPortlet

The ViewPortlet does not trigger any events on its own, but the ViewPlugin (which is used in the ViewPortlet) does. See "ViewPlugin events" (Section 4.3.4, p.164) for details.

5.4. GenticsContentModule

5.4.1. Introduction

Display content, stored in the GenticsContentRepository, published by Gentics Content.Node. Supports dynamic lists and some user interaction.

5.4.2. Details

Contents based on templates of type CSS, JS or XML are rendered fullscreen automatically.

5.4.3. Parameters

Table 5.4. Portlet parameters for GenticsContentModule

Name	Type	Default	Description
contentid	String	(either contentid or filename required)	Id of the content to initially load.
filename	String	(either contentid or filename required)	Filename of the content to load initially (if no contentid is set)
pub_dir	String	/	Publish directory of the content to load initially (if a filename, but no contentid is set)
node_id	Integer	NULL	Id of the node, from which the initial content shall be taken (if a filename, but no contentid is set). The default value NULL means that the node_id is not used to find the content.
datasource	String	(required)	Id of the datasource to use, must be of type Gentics .Node ContentRepository.
rule	String	true	Rule to determine whether the content can be

Name	Type	Default	Description
			viewed or not. Is also derived to plist items.
maxwidthname	String	maxwidth	Name of the parameter passed to the module to spezify the width of the resized image(
maxheightname	String	maxheight	Name of the parameter passed to the module to spezify the height of the resized image
imagetypename	String	auto	Name of the parameter passed to the module to spezify the image type.
maxallowed-width	Integer	16384	Maximal allowed with in pixel for resizing images. Maximal allowed width.
maxallowed-height	Integer	16384	Maximal allowed height in pixel for resizing images. Maximal allowed height.
templateEngine	Boolean	false	whether to use TemplateEngine2
editPage	String		The ML code to use for developer's editmode.
tem-plate_noperm	String		The ML code to use if a requested page was forbidden by rule.
prefixstaticurls	String		By setting this parameter all static URLs made from PLinks to files or images contained in the contentrepository will be prefixed with the string provided.
ruleactions	Node		<actions> node to define pluggable actions that determine whether the user has permission to render the current content. There must exist at least one action with id ***permission*** that returns TRUE or FALSE in a return variable named ***permission*** . The content object will be available as `data.object` in the pluggable actions. For further details on pluggable actions, see Section 4.2.2.8, “Pluggable Actions” .
languageman-agement	Boolean	false	With this setting, the language management for the GenticsContentModule can be switched on. See “Language Management” (Section 5.4.10, p.216) for details.
language	String		Current content language, when ***languagemanagement*** is used.
languagefall-back	List of Strings		Space separated list of language codes to be used for language fallback, when ***languagemanagement*** is used, but the current content is not available in the desired language.
pathlinks	Boolean	false	When this parameter is set to `true` , the generated “PLinks” (Section 5.4.6, p.213) will contain pathinformation (pub_dir and filename and eventually node_id) instead of the contentid. This can be used to create bookmarkable urls for the GenticsContentModule. For this feature to work correctly, it is necessary, that the pathinformation (pub_dir, filename, node_id) is really published to the content repository.
nodeidinpath-links	Boolean	true	This parameter is only interpreted, when the parameter ***pathlinks*** is set to `true` . By setting this to `false` , inclusion of the ***node_id*** into the gen-

Name	Type	Default	Description
			erated “PLinks” (Section 5.4.6, p.213) can be disabled. Note that when the *node_id* is not used to identify the linked content object, the link may not unique, when content objects from multiple nodes are all contained in the same content repository.
contentChangeChecking	Boolean	false	Flag to switch on/off automatic checking for content changes. When switched on, the shown content is frequently checked for changes and refreshed, if necessary.
contentChangeCheckingInterval	Integer	60	Interval in seconds, in which the shown content is checked for modifications, when the parameter *contentChangeChecking* is `true`.
secret	String		By setting a secret, the Image Resize Validation is switched on. See “Image Resize Validation” (Section 3.16, p.83) for details.

5.4.4. Properties

Table 5.5. Portlet Properties for GenticsContentModule

Variable	Type	Description
forbidden	Boolean	Whether the content was forbidden by the rule parameter.
renderedcontent	String	The currently rendered content.
contentid	String	The currently rendered contentid
*	String	All other content properties in the ContentRepository, as defined in Gentics Content.Node's tagmap.

5.4.5. Events of the GenticsContentModule

Table 5.6. Events of the GenticsContentModule

Name	Description	Event properties	
onSelect	This event is triggered every time a contentlink is requested.	**Name**	**Description**
		contentid	The contentid if the linktarget, which was requested. May also be accessed using the portal property event.properties.contentid
onContentNotFound	This event is triggered in one of the following cases: • The content with the initially configured contentid does not exist. • The content with the initially configured node_id, pub_dir and filename does not exist.	**Name**	**Description**
		contentid	The contentid of the non-existent content (if content was requested/configured by contentid).
		node_id	The node_id which was requested/configured (if content was requested/configured by node_id/pub_dir/filename).
		pub_dir	The pub_dir which was requested/configured (if content was requested/configured by

<table>
<tr><th>Name</th><th>Description</th><th>Event properties</th></tr>
<tr><td></td><td>

A non-existent content was requested (via an action url) by contentid or node_id/pub_dir/filename.
A non-existent content should be set (e.g. in a reaction) by setting the parameter "contentid" to the portlet.

</td><td>
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td></td><td>node_id/pub_dir/filename).</td></tr>
<tr><td>filename</td><td>The filename which was requested/configured (if content was requested/configured by node_id/pub_dir/filename).</td></tr>
</table>
</td></tr>
</table>

5.4.6. PLinks

This portlet renders <plink>s either because it is configured as “PLink Processor” (Section 3.8.2.9, p.32) or automatically for <plink>s contained in the rendered content. When a user clicks the <plink>, an `onSelect` event is thrown for the plink processing portlet. Any further activity (e.g. rendering the target content) has to be implemented as reaction to this `onSelect` event. For details on reactions, see Section 4.11.1.3, “PNodes”.

The <plink>s rendered in the content shown in the GenticsContentModule will automatically be XML escaped, when the *`mimetype`* of the content is either `text/html` or `text/xml` . For all other *`mimetype`* s (e.g. `text/js` or `text/css`), the <plink>s will NOT be XML escaped.

XML escaped URLs with have the characters `&` , `<` , `>` , `'` and `"` be replaced by their entities `&` , `<` , `>` , `&#039;` and `&#034;` respectively.

Note

Note that just by placing <plink>s and configuring a “PLink Processor” (Section 3.8.2.9, p.32), this processor will NOT automatically display the pages linked to by clicked <plink>s. You will always need a reaction that forces the clicked page to be displayed.

5.4.7. Image resizing

Images can also be dislplayed by this portlet. Whenever an image is to be processed it is possible to resize to image to a custom size as followd (if parameternames are default). If one of the parameters of height or width is not spezified the image is processed proportional.

```
<plink id="10008.12345">&p.maxwidth=500&p.maxheight=200&p.imagetype=png
```

Resized images are chached using the portal's internal cache (see Section 3.10, “Cache”).

5.4.8. Custom actions

This module offers custom actions to enable interaction between the user and the portal through links. Similar to plinks, paction placeholders in the content are transformed to dynamic portallinks:

```
<a href="<paction [customactionvalue]>">Click this action!</a>
```

The customaction is stored into the portal after the click and can be used in the whole portal through the moduleproperty "customaction" until it is changed or the portalsession is ended. A customaction can be also generated manually for POSTs:

```
<form method="POST">
<input type=hidden name="am" value="PCP">
<input type=hidden name="p.action" value="custom">
```

```
<select name="p.customaction">
<option value="[customactionvalue]">
</select>
<input type="submit">
</form>
```

5.4.9. Plists

Plists are used to display dynmic content lists.

Example 5.2. Syntax Plists

```
<plist id="...">
      <pagelinkunselected></pagelinkunselected>
      <pagelinkselected></pagelinkselected>
      <pagelinkseperator></pagelinkseperator>
      <listtemplate></listtemplate>
      <pagesize></pagesize>
      <start></start>
      <limit></limit>
      <pagebarleftsize></pagebarleftsize>
      <pagebarrightsize></pagebarrightsize>
      <pagelinkellipsis></pagelinkellipsis>
      <showfirstpage> </showfirstpage>
      <showlastpage> </showlastpage>
      <sortby> </sortby>
      <sortorder> </sortorder>
      <emptyresult></emptyresult>
      <filterrule></filterrule>
      <template></template>
</plist>
```

Table 5.7. Settings Plists

Name	Type	Default	Description
id	String		Id of the plist, must be unique in the scope of the content, required for usage of paging
pagelinkunselected	String		Template of the page link for unselected pages. Supports variables of the pagelink template (see below)
pagelinkselected	String		Template of the page link for the selected page. Supports variables of the pagelink template (see below)
pagelinkseperator	String		Template for the separator between pagelinks to neighbouring pages
listtemplate	String		Template for the list, when paging is active. Supports variables <insert plist> and <insert pagebar> (see below)
pagesize	Integer		number of elements shown on each page, optional (defaults to "no paging")
start	Integer		Number of the first shown element, optional (defaults to 0)
limit	Integer		Maximum number of elements shown in

Name	Type	Default	Description
			the plist
pagebarleftsize	Integer		maximum number of pagelinks shown to the left of the selected page (excluding the first page). Optional, defaults to -1 (all pages shown)
pagebarrightsize	Integer		maximum number of pagelinks shown to the right of the selected page (excluding the last page). Optional, defaults to -1 (all pages shown)
pagelinkellipsis	String		String to be shown when pagelinks are omitted due to pagebar limitations. Optional, defaults to " … "
showfirstpage	Boolean		Flag, whether the link to the first page is always shown. Optional, defaults to true.
showlastpage	Boolean		Flag, whether the link to the last page is always shown. Optional, defaults to true.
sortby	String		Attribute name for sorting of elements
sortorder	String		Sortorder (asc or desc)
emptyresult	String		String to be shown when no elements found. Optional, defaults to empty string
filterrule	String		Rule for filtering the elements shown in the plist
template	String		Template for display of one element (row) in the plist. Supports <insertlist>s shown below. All attributes of the shown contentobject can be accessed by <insert>

Table 5.8. Template Variables for Rowtemplate

Name	Type	Description
nr	Integer	Number of the element on the page, starting with 1

The Rowtemplate variables are provided through <insertlist *> tags, in the template parameter.

Table 5.9. Template Variables for Listtemplate

Name	Type	Description
plist	String	filled with the plist (exclusive paging)
pagebar	String	filled with the pagebar

The Listtemplate variables are provided through <insert *> tags, in the listtemplate parameter.

Table 5.10. Template Variables for Pagelink

Name	Type	Description
start	Integer	Number of the first element shown on this page
end	Integer	Number of the last element shown on this page
page	Integer	Number of thuis page

Name	Type	Description
pageurl	String	url to switch onto this page

The Pagelink variables are provided through <insert *> tags, in the pagelinkunselected and pagelinkselected parameter.

5.4.10. Language Management

The optional language management can be used to automatically display the selected content in the clients language, when available. It also supports fallback to other languages in a specified order.

The language management can be switched on by setting the pnode parameter *languagemanagement* . When it is switched on and the `GenticsContentModule` should render a page, it tries to find a better suitable one by following this algorithm:

1. Determine the desired content language (by reading the pnode parameter *language*).
2. Get the attribute *contentid_[languagecode]* (where [languagecode] is the desired language) from the content object that shall be rendered. When the attribute links to the content object in the desired language, render that object and finish.
3. When the content object was not found in the desired language, do the step above with the languages optionally defined in the pnode parameter *languagefallback* and render the content in the first language found.
4. When the content was not found in any of the specified languages, simply render the given content object, no matter what language it is in.

The language of the GenticsContentModule can be set to the current portal language by configuring the pnode-parameter like this:

```
<pnode ...>
  <parameters>
    <parameter name="language"><pobject>portal.language.id</pobject></parameter>
    ...
  </parameters>
</pnode>
```

5.4.11. Limitations

The GenticsContentModule is not able to handle data which is stored inside Multichanneling Content.Repositories. Please use the GenticsContentPortlet instead.

5.5. GenticsContentPortlet

5.5.1. Introduction

The GenticsContentPortlet basically is a variation of the “GenticsContentModule” (Section 5.4, p.210) with some deprecated features removed and some new features added. The following sections will just describe the differences, for additional information see the documentation of the GenticsContentModule.

5.5.2. Removed Features

The following list of features has been removed from the GenticsContentPortlet

- `<plist>` in the content (can be replaced by Velocity)
- `<pxnl>` in the content (can be replaced by Velocity)
- `<insert>` in the content (can be replaced by Velocity)
- `<pobject>` in the content (can be replaced by Velocity)

5.5.3. Parameters

Additionally to the Parameters of the GenticsContentModule, the GenticsContentPortlet supports the following pnode parameters.

Table 5.11. Portlet parameters for GenticsContentPortlet

Name	Type	Default	Description
backendurl	String	NULL	URL to the Gentics Content.Node 5 Backend for support of rendering content which was fetched from the backend. This is also needed for the support of frontend editing in conjunction with Gentics Content.Node 5. The backend URL must be reachable from the Portal Server, NOT the client.
backendcookie	String	NULL	Comma separated list of names of cookies which need to be forwarded to the request to the backend to fetch the content.
backendmode	Boolean	false	This setting switches the source for fetching the rendered content: When set to `false` (which is the default), the content is fetched from the configured Content Repository. When set to `true` , the content is fetched from the linked backend (Gentics Content.Node 5) via the configured *`backendurl`* .
proxyprefix	String	/ GCN5_Portal/GCN/	Prefix for requests which are directed to the backend. The prefix must direct the request to a Proxy Servlet, that proxies the request to the backend (since the backend itself normally is not available via the same hostname).
nodeidinpath-links	Rule	true	This parameter is only interpreted, when the parameter *`pathlinks`* is set to a rule (can also be just `true`/`false`). The rule will have `portal`, `portlet` and `object` (current link target) available. Per default this is always `true` . By setting this to a rule evaluating to `false` , inclusion of the *`node_id`* into the generated “PLinks” (Section 5.4.6, p.213) can be disabled. Note that when the *`node_id`* is not used to identify the linked content object, the link may not unique, when content objects from multiple nodes are all contained in the same content repository.
max-age	Expression	5184000	This parameter controls the value of the response header `Cache-Control:max-age=[value]` when returning binary data. The value must be an Expression, that evaluates to an Integer (in Seconds). In the expression, all attributes of the content object are available.

Name	Type	Default	Description
			Examples: 1. `86400` for statically caching one day. 2. `if(mimetype LIKE 'image/%', 3600, 86400)` for caching images for an hour, everything else for a day.

Since fetching content from Gentics Content.Node 5 requires a valid Gentics Content.Node 5 Session, this will only work, when the Portal User has logged in and was authenticated via a "GCNAuthenticationManager" (Section 3.8.3.7, p.44) .

5.5.4. Additional Events of the GenticsContentPortlet

Additionally to the Events described for the GenticsContentModule, the GenticsContentPortlet will throw the following events:

Table 5.12. Events of the GenticsContentPortlet

Name	Description	Event properties	
onContentLoad-Error	This event is triggered when a content cannot be loaded (either from the datasource or from the backend) due to an error in the underlying system.	**Name**	**Description**
		contentid	The contentid if the content which could not be loaded

5.6. GenticsLoginModule

5.6.1. Introduction

This portlet can be used to let users log in. It consists of two TextComponents and one ButtonComponent. You may customize these templates by adding a template loader to your configuration. The class 'PasswordClass' is assigned to the TextComponent that is used for password input.

5.6.2. Parameters

Table 5.13. Portlet parameters for GenticsLoginModule

Name	Type	Default	Description
loginregex	String	NULL	Regex to check for allowed login names.
logindescription	String	NULL	Message to be shown when the entered login name does not match the given *`loginregex`*.
passwordregex	String	NULL	Regex to check for allowed passwords.
passwordde-scription	String	NULL	Message to be shown when the entered password does not match the given *`passwordregex`*.

Warning

The parameters `loginregex` and `passwordregex` are only used to validate user input if the *user login fails*. If the regular expressions do not match used passwords the users will still be able to log in.

5.6.3. Properties

Table 5.14. Portlet Properties for GenticsLoginModule

Variable	Type	Description
logouturl	String	URL to log out.

5.6.4. Events of the GenticsLoginModule

Table 5.15. Events of the GenticsLoginModule

Name	Description	Event properties	
onUserLogout	This event is triggered when a user logs out.	**Name**	**Description**
		No event properties.	

When the user logs in (either using the GenticsLoginModule or through other means) a `portal.events.onLogin` is triggered. See “Portal events” (Section 4.3.2, p.163) for details.

5.7. AdministrationPortlet

5.7.1. Introduction

The AdministrationPortlet is used to deploy and undeploy portlet applications, and to manage object definitions for datasources of type `contentrepository`. For details, see Section 4.4, “Portlet implementation” and Section 3.8.6.3.1, “Datasources of type `contentrepository`” .

5.7.2. Deployment

Deployment is the process of installing portlet applications on a Gentics Portal.Node Server so that it can render and use the portlets contained in the portlet application. A valid .war file, generated by the SDK's build script (MyPortletapplication/tools/build.xml) is required, and needs to be uploaded with your browser using the AdministrationPortlet.

During deployment of an portletapplication the affected portlets are unavailable for a short time, which means that their boxes will be invisible for the portal visitor. The applicationserver or portal requires no restart.

Note

It is necessary to correctly configure a Tomcat user with role `manager` and enter the login data for this user into the Portal Configuration File in the `<administration-section>`. Otherwise the deployment of portletapplications will not work! See Section 3.8.8.2, “Deployer” for details.

Note

The AdministrationPortlet's deployment feature is currently only supported on Tomcat. For deployment of portletapplications on other servers, please get in contact with us.

5.7.3. Objectmanagement

In the objectmanagement, you can define the object definitions for datasources of type `contentrepository`.

Note

When defining equally named attributetypes for different objecttypes, it is important to use identical definitions. The AdministrationPortlet will warn you about violations.

5.7.3.1. ViewWizard

5.7.3.1.1. Basic Functionality

The ViewWizard is part of the ObjectManagement Module and will assist you in generating views for your objects. Furthermore a simple navigation view wich lists your generated views will also be created. Once your objects and their attributes have been defined using the ObjectManagement Module you can create views for your objects.

Select all objects you want to generate views for in the ObjectManagement Module and click the "Generate Views" button. This will lead you to the ViewWizard where a list of selected objects and their ForeignLinkAttributes is presented. For those attributes there are two options available:

1. No Edit: this option will just list the linked objects, but provide no facility to edit them. If you want to edit the linked object you have to generate a seperate view.

2. Edit Nested: when selecting this option you will be able to create and edit linked objects via a nested form component. Of course you can generate seperate views to edit those objects as well.

After setting the options for your objects you have to enter a pnode id. Please keep in mind that the corresponding pnode must have a proper viewBaseDir, otherwise creation of views will fail with an appropriate error message. You will be asked to confirm overwrite if previously generated views have been found inside the portlet's viewBaseDir. If you agree to overwrite all your changes to those views will be lost. The navigation view will be generated each time you create views without overwrite confirmation so be sure you don't edit it in any case.

After clicking the "Generate Views"-button and possibly confirming the overwrite message your views are written to disk and ready to use. Now you can start editing your objects right away!

5.7.3.1.2. Integration of the Navigation View

To integrate the autogenerated navigation view in your portal template you just have to create a pnode, which will assign it's id to the main pnode when the onViewChange event occurs. Furthermore you'll need to define a template which loads the navigation view when the pnode's windowstate is set to "normal". Have a look at the Gentics Portal.Node SDK Portal for detailed information on how to implement this.

5.7.3.2. Data import

The 'data import' link next to each objecttype in the objecttype list can be used to import new data into the contentrepository for the corresponding objecttype. For a detailed description of the csv file header format and the parameters, see Section 4.2.2.8.8, “CSVDataImportAction”.

The file encoding selection must match the file format, else the imported text will be corrupted or the import may fail. The file must be selected directly before the import is started, because the file is not stored on the server.

A filename for a logfile can be specified, where all log output of the import will be stored. The logfile is created, after the import is finished. Both the fileprefix and the logfile specify files and paths on the server. The files to import must be placed on the portal server.

After the import is finished, the imported objects can be deleted, using the 'Dismiss import' button. The import can be dismissed, until a new import is started. The loglevel selection allows to select how much information is displayed in the logmessages display. This does not affect the logfile.

5.8. GenticsContentSearchModule2

This portlet can be used to display results of a 3rd party search engine.

The standard configuration is best suited for a mnoGoSearch™ but it can be used for any search which is invoked by a HTTP URL (with GET parameters) and returns a XML document.

To use this module you have to create a new portlet in the portlet descriptor (portlet.xml and gentics-portlet.xml) of your Portletapplication (see below for a sample configuration)

Example 5.3. GenticsContentSearchModule2 - Minimal portlet.xml portlet descriptor

```
<portlet>
  <portlet-name>GenticsContentSearchModule2</portlet-name>
    <portlet-class>
      com.gentics.portalnode.genericmodules.ViewPortlet
    </portlet-class>
    <supports>
      <mime-type>text/html</mime-type>
    </supports>
    <portlet-info>
      <title>GenticsContentSearchModule2</title>
    </portlet-info>
  <portlet-preferences>
    <preference>
      <name>datasource</name>
      <value>content</value>
    </preference>
    <preference>
      <name>searchurl</name>
      <value>
http://localhost/cgi-bin/mnogosearch.cgi?ps=800&amp;q=${query}
      </value>
    </preference>
    <preference>
      <name>xslt</name> <!-- Newlines added for readability -->
      <value>           <!-- needs to be removed            -->
file:///${com.gentics.portalnode.home}/WEB-INF/views/
GenticsContentSearchModule2/searchconvert_mnogosearch.xslt
      </value>
    </preference>
    <preference>
      <name>contentidfilter</name>
      <value>http.*?index.php\?C=([0-9\.]+)</value>
    </preference>
  </portlet-preferences>
</portlet>
```

Example 5.4. GenticsContentSearchModule2 - Minimal gentics-portlet.xml portlet configuration

```
<portlet>
  <portlet-name>GenticsContentSearchModule2</portlet-name>
  <parameter-description>
    <parameter-definition type="string" name="viewplugin.viewbasedir">
      path of the view definitions</parameter-definition>
    <parameter-definition type="string" name="searchmodule.searchurl">
      search url</parameter-definition>
  </parameter-description>
  <start-parameters>
    <parameter name="viewplugin.viewbasedir">
      ${com.gentics.portalnode.home}/WEB-INF/views/GenticsContentSearchModule2
    </parameter>
  </start-parameters>
  <templates>
    <template loader="plugins">
      <classes>
        <class id="plugin">formplugin2</class>
        <class id="component">DatasourceListComponent</class>
        <class id="class">search</class>
      </classes>
      <parameters>
        <parameter id="filename">
          modules/GenticsContentSearchModule2/listcomponent_search.vm
        </parameter>
      </parameters>
    </template>
  </templates>
</portlet>
```

Table 5.16. Events of the GenticsContentSearchModule2

Name	Description	Event properties	
onSearch	This event is triggered when a user requests a new search..	**Name**	**Description**
		searchquery	Searchquery entered by the user.
openResult	This event is triggered when a user clicks on a search result (The only exception is if the search result is a file, which will be immediately downloaded - if it is a content object).	**Name**	**Description**
		contentid	The contentid of the result object.
		object	The complete result object which was clicked.

Note

The above events are triggered within a View - therefore the full path for a reaction for onSearch would look like: portal.modules.GenticsContentSearchModule2.plugins.viewplugin.onSearch

Table 5.17. Portlet Preferences for GenticsContentSearchModule2

Variable	Description
searchurl	Search URL used for the search. ${query} will be replaced by the user input.
datasource	Datasource from which to load the objects.
defaultpagesize	Page size used for the DatasourceListComponent
xslt	URL of the XSLT which should be used to convert search results to the input of a "CreateResolvablesAction" (Section 4.2.2.8.7, p.143).
contentidfilter	A regular expression which should replace a URL returned in the results of the search so that only the contentid is left over. E.g. if the search returns URLs like: http://localhost/cgi-bin/index.php?C=10007.123 you can use the following regular expression: http.*?index.php\?C=([0-9\.]+) which would replace the whole URL and leave: 10007.123 (the first reg-exp group has to match the contentid)
teaserview_showresults	true/false - Wether the result list should be shown if the window state is 'normal'
hideadvanced	true/false - Wether to hide the 'Advanced Search' option.

The GenticsContentSearchModule2 will try to display 3 attributes of the resolvables: name, edit-timestamp and description (if they exist).

5.8.1. Further Customization

Further customization of the GenticsContentSearchModule2 can be made by providing Resolvable objects. These are parsed by the "CreateResolvablesAction" (Section 4.2.2.8.7, p.143).

These have 2 objects. The first object are general preferences for a select component, while the second will hold the values for the SelectComponent.

5.8.1.1. Pagesize

First of all you can present the user a list of Pagesizes which are available.

Example 5.5. Pagesize configuration

```
<preference>
  <name>pagesize</name>
    <value><![CDATA[
<objects xmlns="http://www.gentics.com/create-resolvables">
        <object>
                <attribute name="label">Page Size</attribute>
                <attribute name="default">10</attribute>
        </object>
        <object>
                <attribute name="5">5</attribute>
                <attribute name="10">10</attribute>
                <attribute name="15">15</attribute>
                <attribute name="20">20</attribute>
        </object>
</objects>
]]></value>
</preference>
```

As mentioned above the first object will be used to set up general SelectComponent options. 'label' will be the label of the SelectComponent, and 'default' will be the default value if the user does not change the value.

5.8.1.2. outputformat

Another predefined option is 'outputformat' where the user can switch between a full list including all details and a short list which will only display the name column.

This SelectComponent can only have two valid options: 'full' and 'short' all others will be ignored.

5.8.1.3. Custom SelectComponents

In addition the GenticsContentSearchModule2 supports up to 4 custom SelectComponents which can be freely configured. The user selected value of these SelectComponents can afterwards be used within the searchurl.

These have the name: selectbox1,selectbox2,selectbox3 and selectbox4 and will be replaced in the searchurl as ${selectvalue1} through ${selectvalue4}

Example 5.6. Custom SelectComponent

```
<preference>
  <name>selectbox1</name>
  <value><![CDATA[
<objects xmlns="http://www.gentics.com/create-resolvables">
        <object>
                <attribute name="label">Word Match</attribute>
        </object>
        <object>
                <attribute name="wrd">Whole Word</attribute>
                <attribute name="beg">Word beginning</attribute>
                <attribute name="end">Word ending</attribute>
                <attribute name="sub">Word substring</attribute>
        </object>
</objects>
]]></value>
</preference>
```

You could then adapt your searchurl e.g. for mnogosearch: http://localhost/cgi-bin/search_mnogo.cgi?ps=800&q=${query}&wm=${selectvalue1}

5.8.1.4. Customizing the Template

The GenticsContentSearchModule2 uses by default a very simple template for the DatasourceListComponent. You can overload this template to use your own and customize the look. For this to work, all you need to do is overloading the template for the DatasourceListComponent of the class 'search'

5.8.1.5. Customizing the XSLT

A XSLT is used to convert the XML document returned by the 3rd party search engine to a predefined xml document used to create resolvables. (Using Section 4.2.2.8.7, “CreateResolvablesAction”). The default xslt stylesheet contained in Gentics Portal.Node only works for one XML format (see below). If you want to support another format, or require additional attributes you need to customize the default XSLT stylesheet.

Example 5.7. Sample XML document returned by mnoGoSearch™

```
<search>
  <res>
    <url>http://mydomain.com/index.php?C=10007.1151</url>
  </res>
  <res>
    <url>http://mydomain.com/index.php?C=10007.1153</url>
  </res>
  <res>
    <url>http://mydomain.com/index.php?C=10007.1160</url>
  </res>
</search>
```

Example 5.8. Default XSLT Stylesheet

Using XSLT it is very easy to convert e.g. the above example XML returned by a search engine to the predefiend format required by Section 4.2.2.8.7, “CreateResolvablesAction”. This example shows the default XSLT which could be adopted to support other search engines.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.gentics.com/create-resolvables">

  <xsl:template match="/">
    <objects>
      <xsl:for-each select="search/res">
        <object>
          <attribute name="contentid"><xsl:value-of select="url" /></attribute>
        </object>
      </xsl:for-each>
    </objects>
  </xsl:template>
</xsl:stylesheet>
```

Chapter 6. Servlets

6.1. Introduction

Gentics Portal.Node provides a set of servlets and servlet filters for a variety of applications.

6.2. GCNProxyServlet

6.2.1. Introduction

The GCNProxyServlet will relay incoming requests to the Gentics Content.Node backend Server.

The package path is com.gentics.portalnode.proxy.GCNProxyServlet.

6.2.2. Details

The servlet will accept incoming GET and POST request and just relay them to the Gentics Content.Node backend server. The filter will also add a proxyprefix attribute the the dispatched request. This attribute it used within Gentics Content.Node to prepare the URLs so that they are usable within the Portal. The servlet will also forward cookies and it will try to follow all redirects.

6.2.3. Parameters

Table 6.1. Portlet parameters for GCNCasProxyAuthenticationServletFilter

Name	Type	Default	Description
gcnBackend-BasePath	String	-	Gentics Content.Node backend path. Example 'http://cms.my-gcn-server/

Chapter 7. ServletFilters

7.1. Introduction

Gentics Portal.Node provides a set of servlets and servlet filters for a variety of applications.

7.2. CASParameterWorkaroundServletFilter

7.2.1. Introduction

The CASParameterWorkaroundServletFilter class is a filter that will workaround a common issue that affects the behavior of the ServletRequest.getInputStream method. Any filter or servlet that invokes the ServletRequest.getParameter method may cause a following call to getInputStream to fail. This filter works around that issue.

7.2.2. Details

Any call to ServletRequest.getParameter might result in the invocation of a ServletRequest background method. This method will open the InputStream of the request and read from it. The inputstream is not resettable. This means that any following code that tries to read from the ServletRequest.getInputStream will get no data at all. The stream can only be read once. Normally this is not a problem since eg. POST variables are also accessible through the ServletRequest.getParameter method. This issue will cause problems when handling post request with GCNProxyServlet since the servlet itself just relays post data to the target.

The filter itself will parse the query string and extract the needed parameters from this data source instead of calling the getParameter() method of the wrapped request object.

7.2.3. Parameters

Table 7.1. Portlet parameters for CASParameterWorkaroundServletFilter

Name	Type	Default	Description
handledParameterNames	String	-	Defines set of parameter names which should be handled by this servlerfilter. Example: 'sid,ticket'.

7.3. GCNCasProxyAuthenticationServletFilter

7.3.1. Introduction

The GCNCasProxyAuthenticationServletFilter class is a filter that will examine incomming requests and decide whether an authentication request should be send to Gentics Content.Node.

7.3.2. Details

The filter will modify the request and response headers in case that the request headers do not contain Gentics Content.Node authentication information. A correct sid and session secret will be added to the request and a set-cookie parameter will be set to the response headers. The Gentics Content.Node session information will be stored in the HttpSession of the portal user. The session id is stored with key 'content.node.session.id' and the session secret is stored with the secret 'content.node.session.secret'. A reauthentication will be performed when the filter detects that the request was not sucuessful.

This filter will perform a cas proxy request. Therefore it is mandatory to have a working cas proxy setup.

7.3.3. Parameters

Table 7.2. Portlet parameters for GCNCasProxyAuthenticationServletFilter

Name	Type	Default	Description
filterURLAcceptRegex	String	-	Defines a regex which can be used to tell the filter which urls it should process. Example: '^(/GCN/.*)$'
gcnRestLoginPath	String	/CNPortletapp/rest/auth/login	Path to the gcn rest api login action (Optional)
gcnBackendBasePath	String	-	Gentics Content.Node backend path. Example: http://cms.my-gcn-server/
casServerUrlPrefix	String	-	Prefix for the cas server. Example: https://mycas/cas-server

7.3.4. Example configuration

Configuration for cas proxy enabled portal server that uses the proxy servlet.

Example 7.1. Syntax Example configuration

```
<context-param>
        <param-name>contextPath</param-name>
        <param-value>/GCN5_Portal</param-value>
</context-param>
<context-param>
        <param-name>gcnBackendBasePath</param-name>
        <param-value>http://cms.my-gcn-server/</param-value>
</context-param>
<context-param>
        <param-name>casServerLoginUrl</param-name>
        <param-value>https://mycas/cas-server/login</param-value>
</context-param>
<context-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://mycas/cas-server</param-value>
</context-param>
<context-param>
        <param-name>serverName</param-name>
        <param-value>https://myfrontend</param-value>
</context-param>

<servlet>
        <servlet-name>ProxyServlet</servlet-name>
        <servlet-class>com.gentics.portalnode.proxy.GCNProxyServlet</servlet-cl
</servlet>
<servlet-mapping>
        <servlet-name>ProxyServlet</servlet-name>
        <url-pattern>/GCN/*</url-pattern>
</servlet-mapping>

<filter>
        <filter-name>CAS Workaround Ticket Parameter Fix</filter-name>
        <filter-class>com.gentics.portalnode.auth.gcn.cas.CASParameterWorkaroun
        <init-param>
```

```
                <param-name>handledParameterNames</param-name>
                <param-value>sid,ticket</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CAS Workaround Ticket Parameter Fix</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- SSO listener -->
<!-- 1. Validation filter to validate cas tickets -->
<filter>
        <filter-name>CAS Ticket Validation Filter</filter-name>
        <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicket
        <init-param>
                <param-name>acceptAnyProxy</param-name>
                <param-value>true</param-value>
        </init-param>
        <!-- The service behind this url will watch out for cas calls -->
        <init-param>
                <param-name>proxyReceptorUrl</param-name>
                <param-value>/GCN5_Portal/receptor</param-value>
        </init-param>
        <!-- pgtUrl for proxyValidate -->
        <init-param>
                <param-name>proxyCallbackUrl</param-name>
                <param-value>https://myfrontend/GCN5_Portal/receptor</param-val
        </init-param>

</filter>
<filter-mapping>
        <filter-name>CAS Ticket Validation Filter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>


<!-- 2. AuthenticationFilter -->
<filter>
        <filter-name>CAS Authentication Filter</filter-name>
        <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter<
        <init-param>
                <param-name>gateway</param-name>
                <param-value>false</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CAS Authentication Filter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>

<filter>
        <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
        <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter
</filter>
<filter-mapping>
        <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
<filter>
        <filter-name>CAS Assertion Thread Local Filter</filter-name>
        <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</fil
</filter>
<filter-mapping>
        <filter-name>CAS Assertion Thread Local Filter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- The GCN Authenticationfilter -->
<filter>
```

```
        <filter-name>CAS - GCN Authentication Filter</filter-name>
        <filter-class>com.gentics.portalnode.auth.gcn.cas.GCNCasProxyAuthentica
        <init-param>
                <param-name>filterURLAcceptRegex</param-name>
                <param-value>^(/GCN/.*)$</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CAS - GCN Authentication Filter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
<!-- end SSO Filters -->
```

Table 7.3. Settings Example configuration

Name	Type	Default	Description
No Settings.			

Chapter 8. API

8.1. Gentics .Node PortalConnector WebServic e API

The Gentics .Node PortalConnector WebService API can be used to read data from the Gentics .Node ContentRepository using WebServices. This section describes the accessible service methods and the properties of used objects.

8.1.1. Service URLs

The URL to get the `*.wsdl` is `http://[hostname]:[port]/Portal.Node/ws/services/datasource?wsdl` . Where *`hostname`* is the server hostname, *`port`* the server port. To access a specific datasource, add the datasourceid to construct an URL like `http://[hostname]:[port]/Portal.Node/ws/services/datasource/[datasourceid]` .

8.1.2. Service Methods

8.1.2.1. getObjects(rule,attributeNames,start,count,sortColumns)

Get all objects from the datasource that match the given rule. Returns an array of SimpleWSObjects.

Table 8.1. Parameters for method getObjects()

Name	Type	Description
rule	String	Filter rule to filter returned objects
attributeNames	Array of Strings	Array of attribute names of the prefetched attributes
start	Integer	Index of the first returned object (starting with 0)
count	Integer	Maximum number of returned objects (-1 for all objects)
sortColumns	Array of Sorting	Optional sorting, may be null or empty for no sorting

8.1.2.2. getCount(rule)

Get the number of objects that match the given rule. Returns an Integer.

Table 8.2. Parameters for method getCount()

Name	Type	Description
rule	String	Filter rule to filter counted objects

8.1.2.3. getObjectsByID(ids,attributeNames,start,count,sortColumn s)

Get all objects from the datasource with given ids. Returns an array of SimpleWSObjects.

Table 8.3. Parameters for method getObjectsByID()

Name	Type	Description
ids	Array of Strings	Ids of the objects to fetch
attributeNames	Array of Strings	Array of attribute names of the prefetched attributes
start	Integer	Index of the first returned object (starting with 0)
count	Integer	Maximum number of returned objects (-1 for all objects)
sortColumns	Array of Sorting	Optional sorting, may be null or empty for no sorting

8.1.2.4. getObject(id,attributeNames)

Get a single object with the given id. Returns a SimpleWSObject.

Table 8.4. Parameters for method getObject()

Name	Type	Description
id	String	Id of the object to fetch
attributeNames	Array of Strings	Array of attribute names of the prefetched attributes

8.1.2.5. getAttribute(objectId,attributeName)

Get an attribute for the given object. Returns a SimpleWSAttribute.

Table 8.5. Parameters for method getAttribute()

Name	Type	Description
objectId	String	Id of the object
attributeName	String	Name of the attribute

8.1.3. Objects

8.1.3.1. SimpleWSObject

The SimpleWSObject represents a single object.

Table 8.6. Properties of SimpleWSObject

Name	Type	Description
id	String	Id of the object
attributes	Array of SimpleWSAttribute	Array of all attributes that are prefetched for this object. Additional attributes can be fetched using “getAttribute(objectId,attributeName)” (Section 8.1.2.5, p.232) .

8.1.3.2. SimpleWSAttribute

The SimpleWSAttribute represents an attribute of a SimpleWSObject.

Table 8.7. Properties of SimpleWSAttribute

Name	Type	Description
name	String	Name of the attribute
type	String	Datatype of the attribute. Is one of (`stringValue` , `integerValue` , `longValue` , `doubleValue` , `dateValue` , `objectValue` , `binaryValue` , `multiStringValue` , `multiIntegerValue` , `multiLongValue` , `multiDoubleValue` , `multiDateValue` , `multiObjectValue` , `multiBinaryValue`). Access the property with this name to get the attribute value.
(Value of property *`type`*)	(Various)	Value of the attribute.

8.1.3.3. Sorting

The Sorting represents sorting information.

Table 8.8. Properties of Sorting

Name	Type	Description
columnName	String	Name of the sorted column (attribute)
sortOrder	Integer	Sort order. Use `1` for ascending and `2` for descending.

8.2. Gentics Portal.Node SDK Java API

This API is included with Gentics Portal.Node SDK and used by developers for their own portlets or extensions, or developers integrating the portal into a custom environment.

See the Gentics Portal.Node SDK Guide and Gentics Portal.Node SDK API documentation for further details and examples.

8.3. Gentics .Node PortalConnector Java API

The Gentics .Node PortalConnector Java API can be used to read and write data from the Gentics .Node ContentRepository using a JDBC or JNDI connection. It is included with Gentics Portal.Node SDK and available standalone as well.

See the Gentics .Node PortalConnector API documentation or Gentics Portal.Node SDK API documentation for details and examples.

The static class `PortalConnectorFactory` allows access to datasources that are managed by the Gentics Portal.Node server, like shown in the following example.

```
// get readable datasource
Datasource ccrDs = PortalConnectorFactory.createDatasource(Datasource.class, "ccr");

// get writable datasource
WriteableDatasource pcrDs = PortalConnectorFactory.createDatasource(WriteableDatasource

// get multichannelling datasource
MultichannellingDatasource mccrDs = PortalConnectorFactory.createDatasource(Multichanne
```

8.4. Gentics Portal.Node AJAX request API

When Gentics Portal.Node receives an AJAX enabled request, the response will be a JSON object with a specified layout, see “Response Format of AJAX enabled requests API” (Section 4.6.3.1, p.171) for the detailed documentation of this API.

This response may include the request to replace a portlet in the browsers DOM tree with a new content. This request (client-side function call) is specified in “Replace rendered portlets API” (Section 4.6.3.3.3.3, p.174) .

Index

A

B

C

D

E

F

G

I

S

T

U

V

W

X