Getting Started Guide

Getting Started Guide

Gentics Portal.Node 4.5.0 SDK
Build 20140310-170102 20140310-170102

Published 2013

Table of Contents

List of Figures

List of Tables

List of Examples

Chapter 1. Introduction

This document will guide Java developers who are new to Gentics Portal.Node on their first steps in the exciting world of Portlet development. We will focus on setting up the SDK including Eclipse and Tomcat, which establish the most common Java Development Environment. Developers who are used to other IDEs should also be able to apply the given instructions. We suggest sticking to the provided sequence of chapters since examples will become more and more challenging throughout the guide.

Table 1.1. Overview of other documents and information sources

Gentics Portal.Node SDK Guide	Step by step guide for installation and learning development and implementation of Gentics Portal.Node portals and portlets. In this interactive guide, you'll find examples to nearly every feature. • eclipse: Help - Help Contents • *web* [http://www.gentics.com/help/topic/com.gentics.portalnode.sdk.eclipseplugin/misc/doc/sdkguide/xhtml/index.html] • *pdf* [http://www.gentics.com/Content.Node/infoportal/documentation/SDK-Documentation.php]
Gentics Portal.Node SDK Welcome View	Launch and experience the two sample portals packaged with the SDK, launch the most important documentation in eclipse, launch the eclipse cheat sheets. • eclipse: Help - Help Contents
Gentics Portal.Node SDK Cheat Sheets	Use cheat sheets for guidence through common tasks, like starting a portal instance, or creating a portlet. • eclipse: Help - Cheat Sheets
Gentics Portal.Node Reference	Describes how the configuration and the features of the portalserver work. • eclipse: Help - Help Contents • *web* [http://www.gentics.com/help/topic/com.gentics.portalnode.sdk.eclipseplugin/misc/doc/reference/xhtml/index.html] • *pdf* [http://www.gentics.com/Content.Node/infoportal/documentation/SDK-Documentation.php]
Gentics Portal.Node API	Java API for developing portlets, and portal exten-

	sions. • *zipped html, pdf* [http://www.gentics.com/Content.Node/infoportal/documentation/SDK-Documentation.php]
Gentics Infoportal	Read articles and further information, download addons and more. • *web* [http://www.gentics.com/infoportal/]
Support	Are you stuck and need help, contact our professional support team. • *web* [http://www.gentics.com/support/] • *mail* [mailto:support@gentics.com]

Your fingers are itching to start some coding? First things first! The next chapter will help you to get your copy of Gentics Portal.Node SDK up and running.

Chapter 2. Installation

2.1. Introduction

The Gentics Portal.Node SDK is usually delivered on CD. You can either use the automated installation which will take about five minutes and takes care of installing all required components for you, or choose manual installation. The latter will take about ten minutes in preparation and another five minutes for installation, and suites best if you have any custom needs, or some components (eg. Eclipse or Tomcat) are already installed.

Table 2.1. Platform requirements

Operating System*:	Linux, Microsoft Windows XP, Microsoft Windows Vista, Microsoft Windows 7 RC
Disk space:	1 GB
Memory space:	1024 MB
JVM**:	Sun JVM 1.5 (32 Bit)

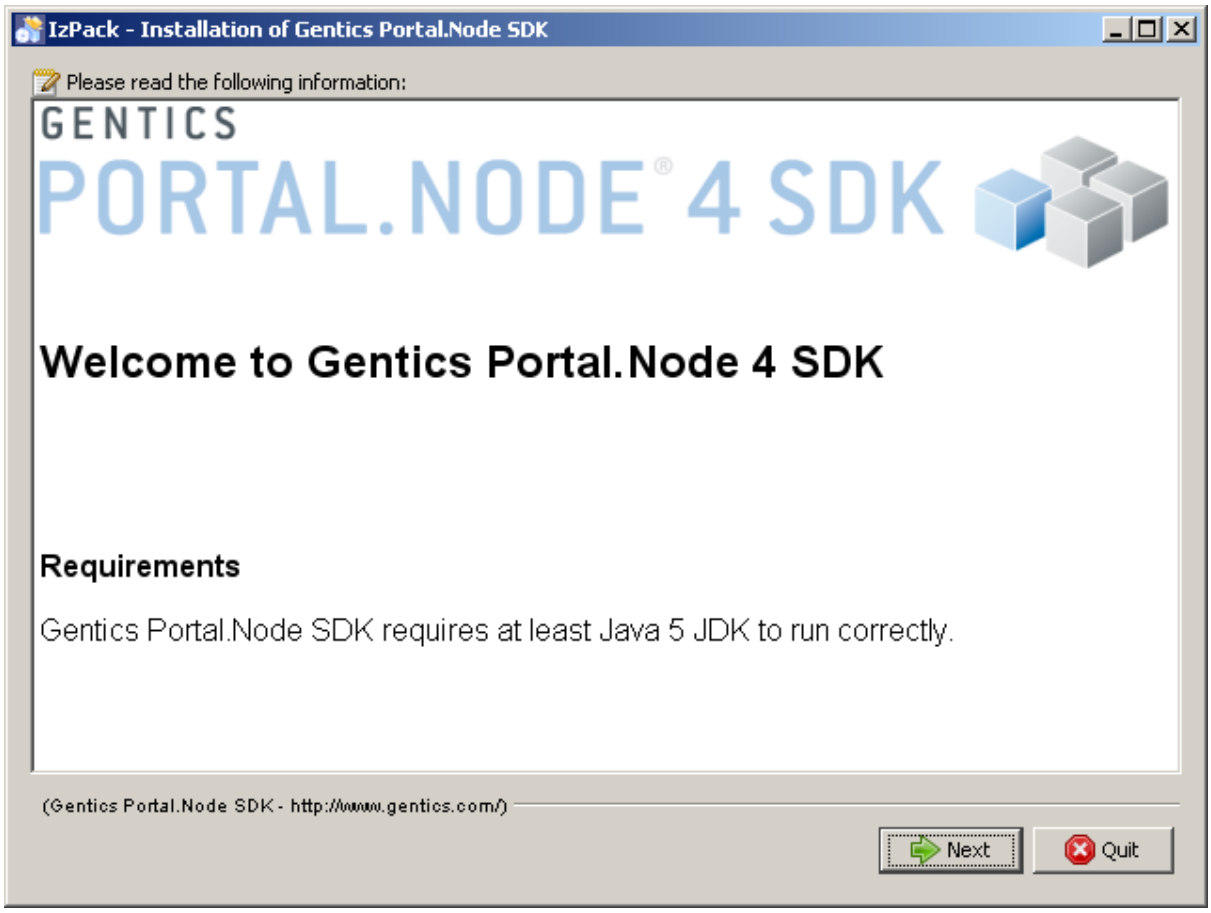
* Installation under Microsoft Windows require administrator rights.

** Please note that 64 Bit JVMs are currently not supported as runtime for the Portal.Node 4 SDK Eclipse environment. This limitation does not affect the JVM that is used to run the application server that contains a Portal.Node 4 web application.

2.2. Automated Installation

Doubleclick setup.exe to launch the installation wizard and follow the instructions. You can safely ignore eventual signature warnings, if you obtained the installation disc in a trustworthy way.

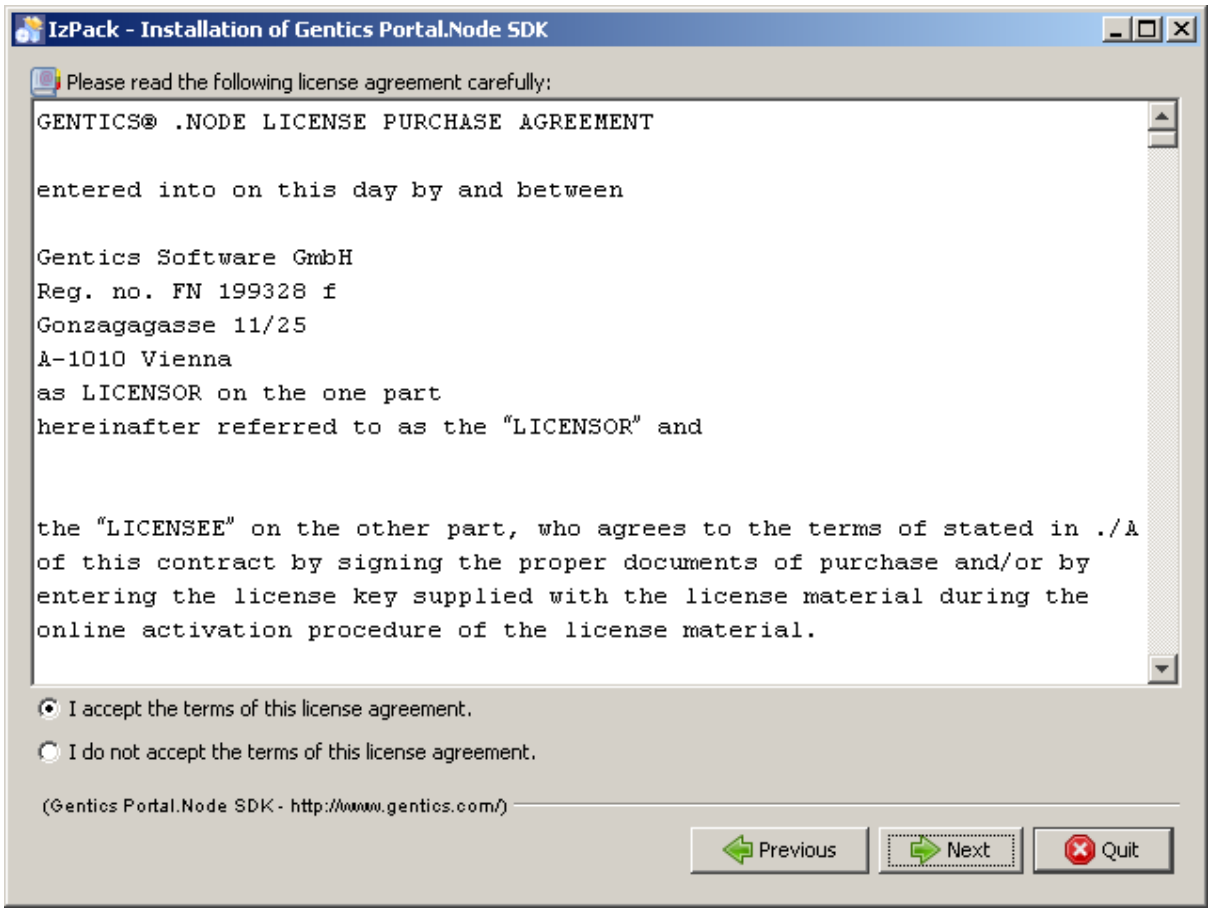
Figure 2.1. Automated Installation



2.2.1. License Agreement

Read and accept the License Terms

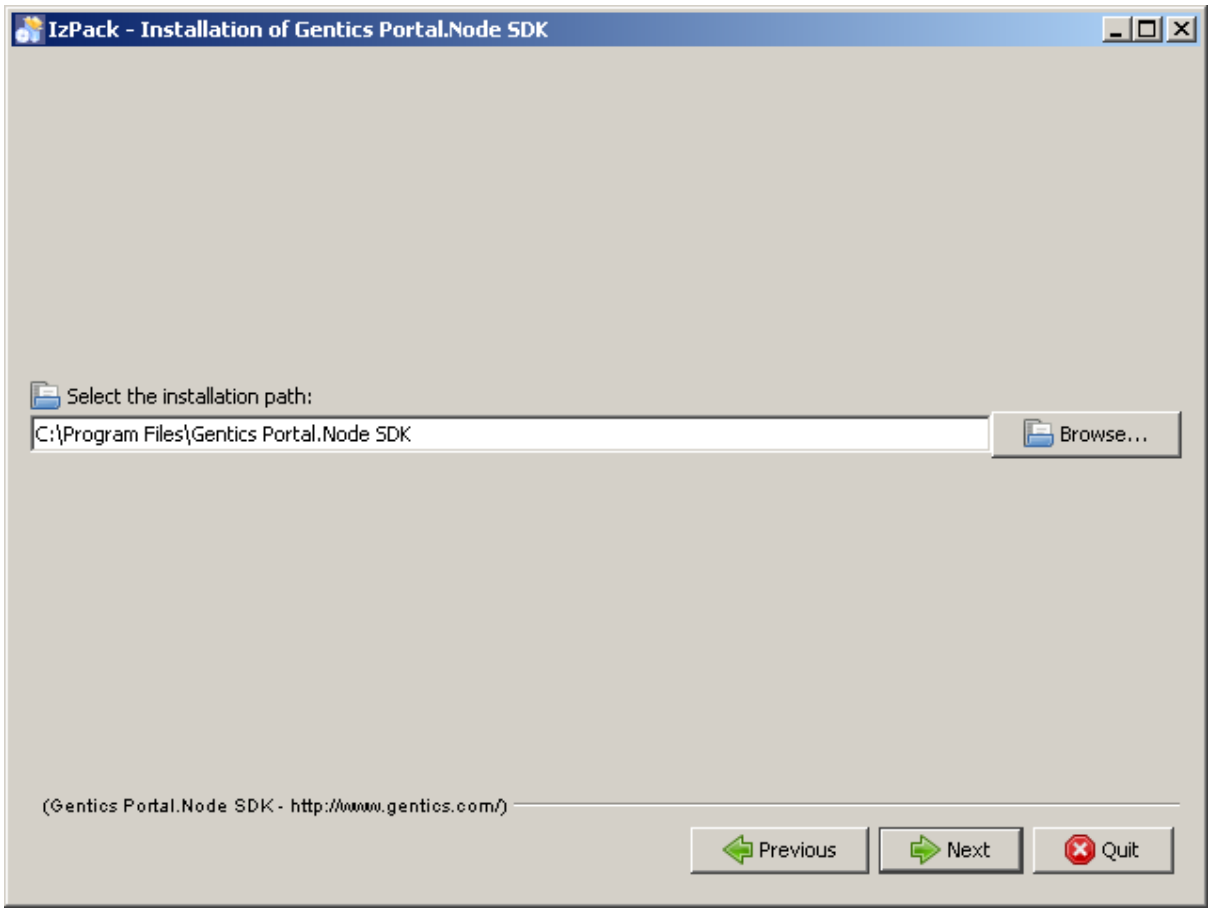
Figure 2.2. License Terms



2.2.2. Choose Directories

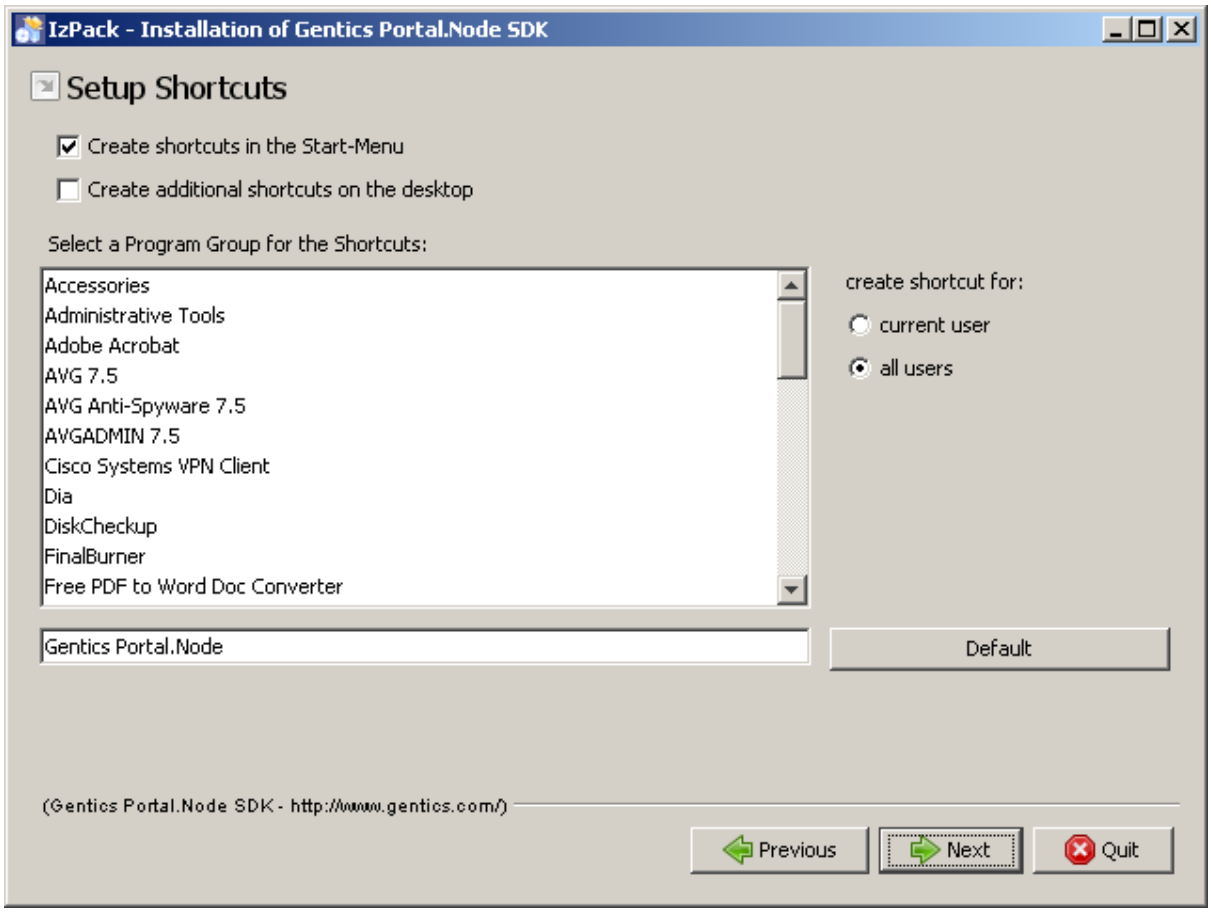
Choose an installation directory. This Directory will contain an IDE configured and ready to be used to develop for Gentics Portal.Node.

Figure 2.3. Choose Installation Directory



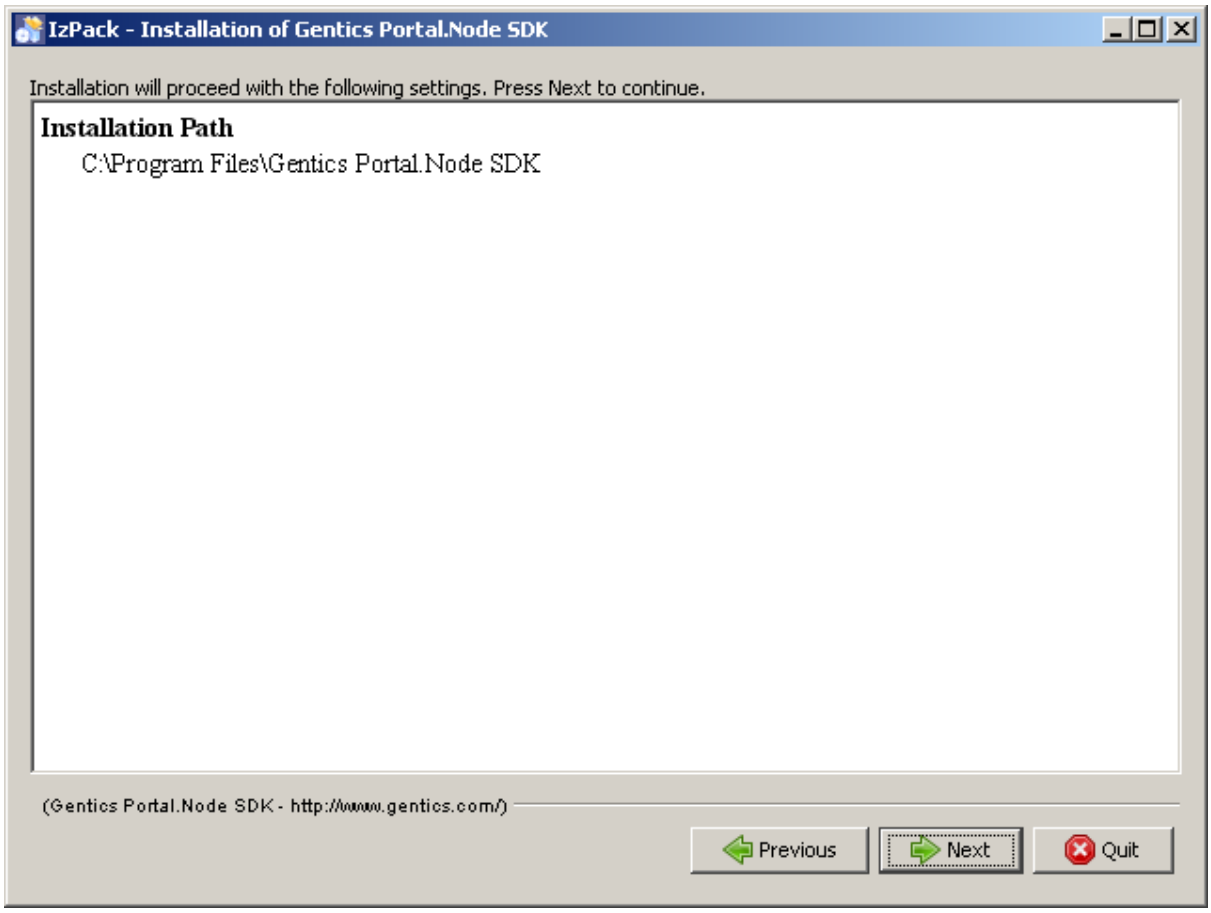
Now choose an Eclipse workspace directory, which will contain your Gentics Portal.Node SDK project files.

Figure 2.4. Choose Workspace Directory



2.2.3. Verify Installation Options

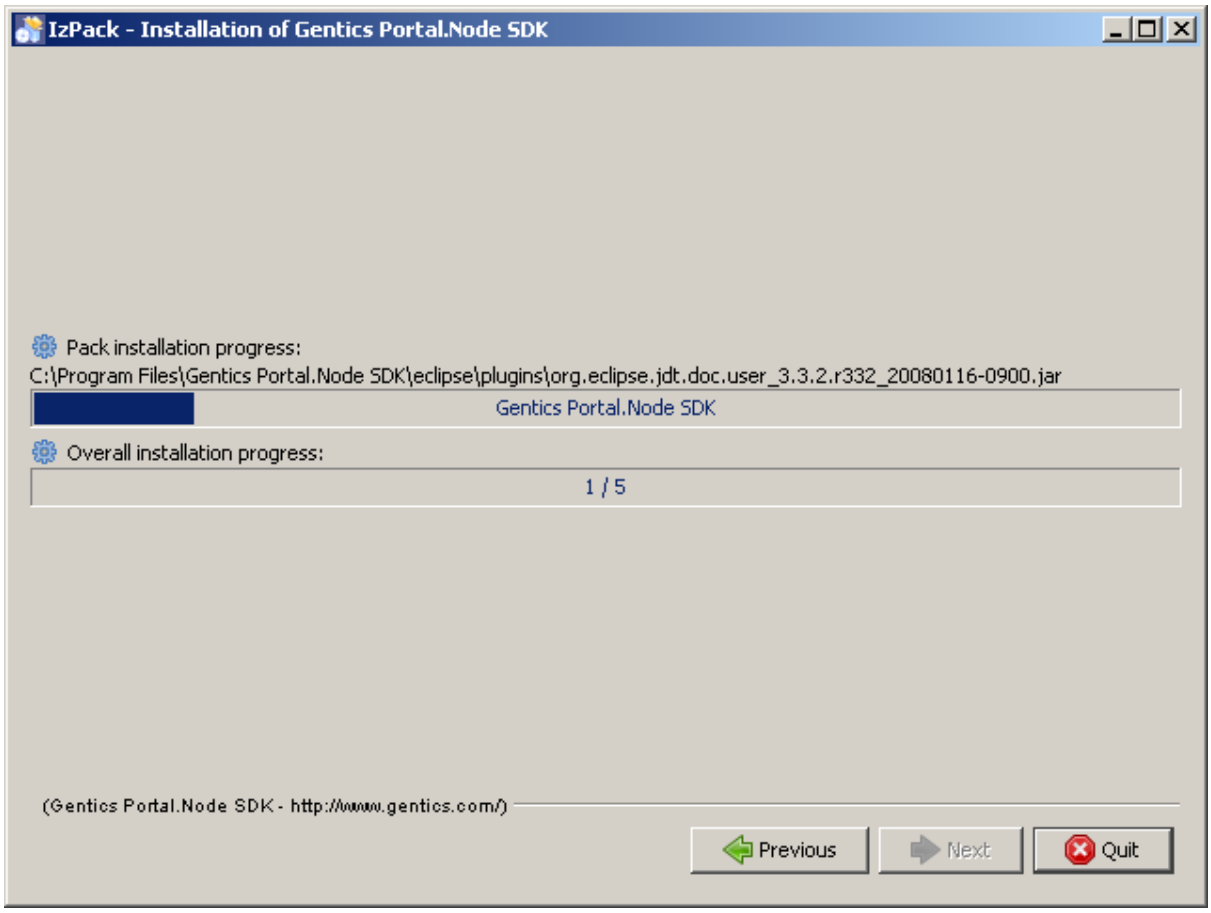
Figure 2.5. Verify Installation Options



2.2.4. Watch Installation

This step may take several minutes to complete.

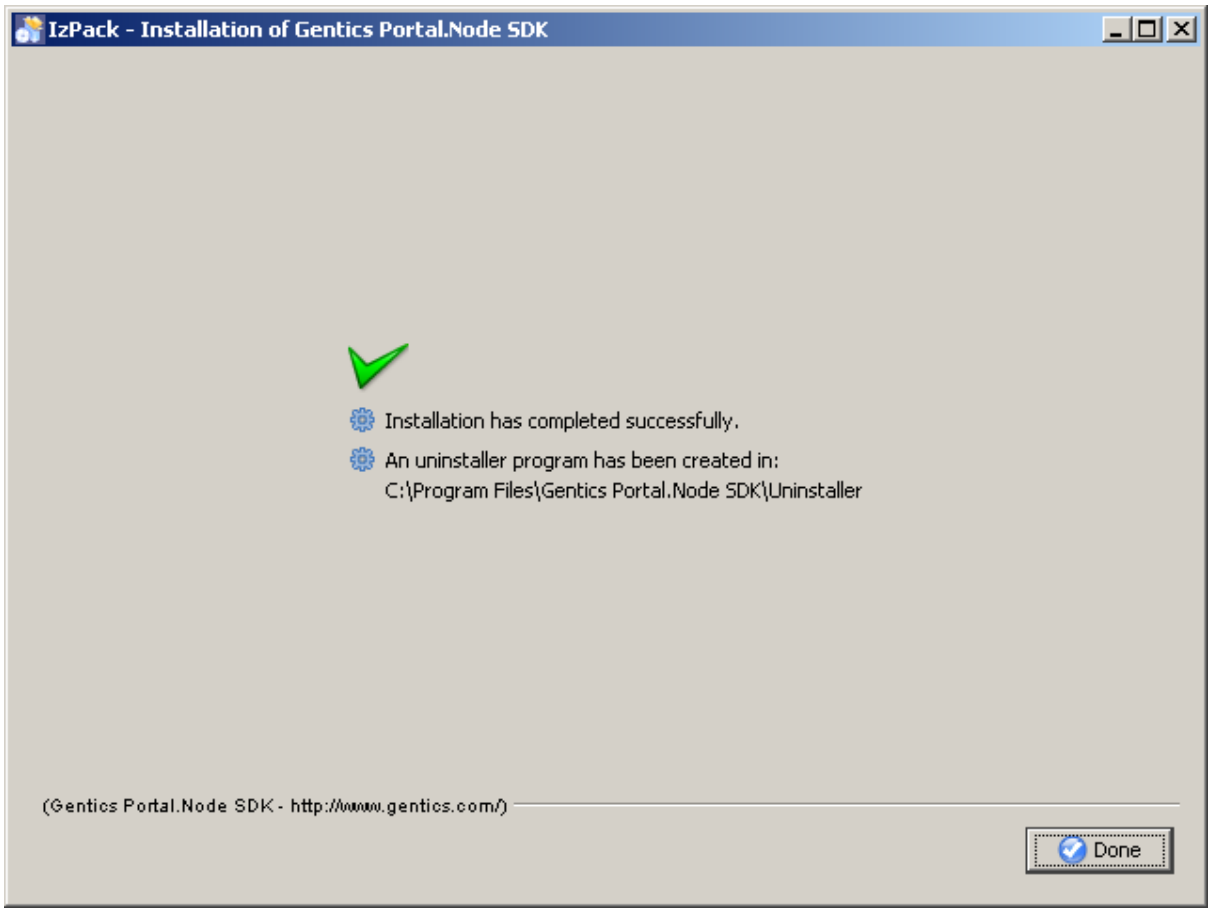
Figure 2.6. Installation Progress



2.2.5. Start Gentics Portal.Node SDK

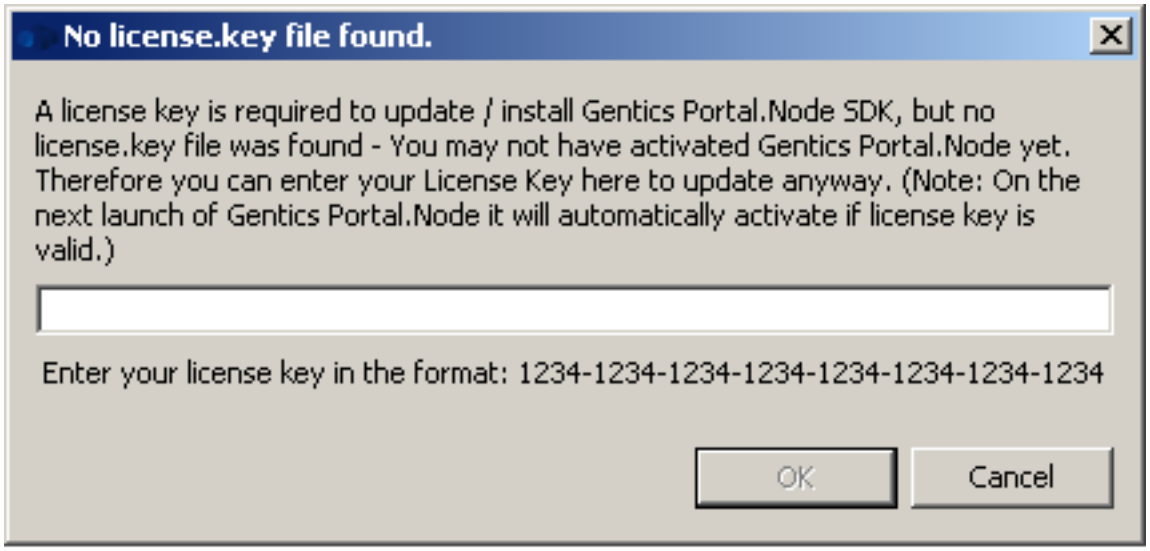
To verify if your installation was successful you can now start the Gentics Portal.Node SDK and launch the provided HelloWorld Portal:

Figure 2.7. Finished Installation



After Pressing Done on the last setup screen your Eclipse installation will launch using a default Java perspective.

A few seconds after the first launch you should see the following dialog where you can enter your license key:



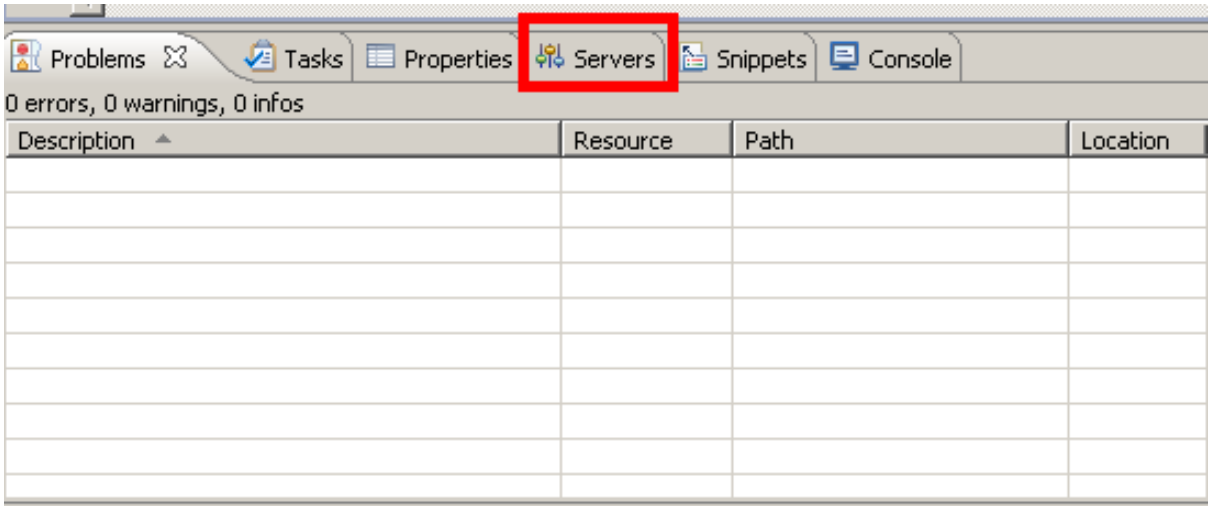
Note

This dialog is part of the update process. After you have entered the license key it will check for new updates. Please note also that you have to configure your proxy server manually via Window -> Preferences -> General -> Network Connections in case you are using a proxy

server for your internet connection. After you configured your proxy server you can activate your SDK installation by using the proxy server connection. Please select Help -> Gentics Portal.Node SDK -> Activate Installation.

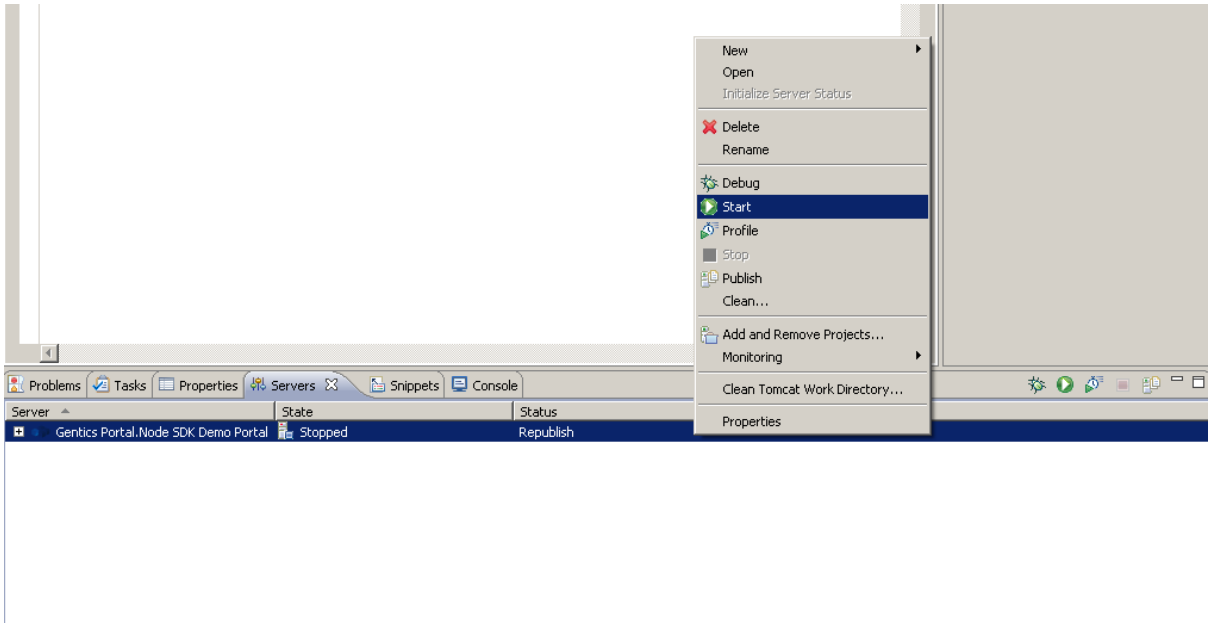
Now you can start Tomcat to see the SDK-Portal in action by clicking the Servers-Button located in the bottom tab-list:

Figure 2.8. Launching Tomcat



After you have to select the desired server instance and click on the green button in the left corner of the tab list or right click on the server instance and select Start

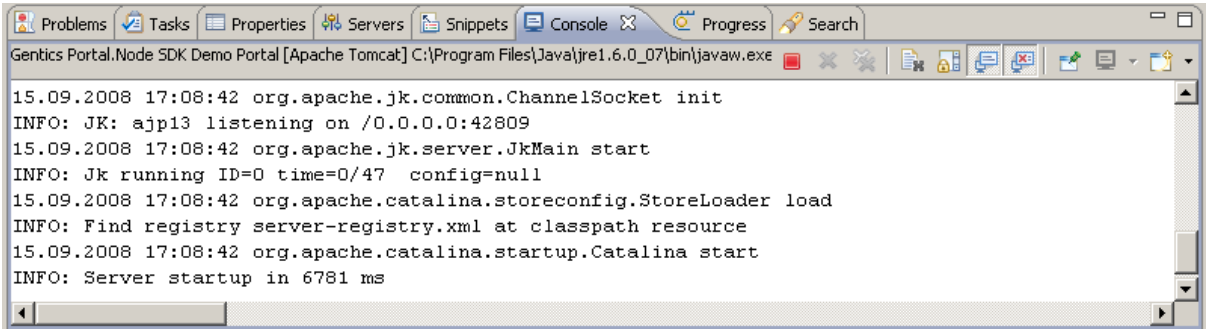
Figure 2.9. Launching Tomcat



Watch the Eclipse console, to see Gentics Portal.Node starting. If your Firewall notifies you of outgoing connections by Java(TM) 2 Platform Standard Edition binary, you can safely allow the connection. The

sun java virtual machine as well as Gentics Portal.Node SDK will try to find updates online.

Figure 2.10. Tomcat Launched Successfully



Now point your browser to *http://localhost:42880/Portal.Node/* to see your SDK-Portal.

Figure 2.11. SDK Portal

Note

If you have already entered your valid license key after starting eclipse, you will be automatically redirected to the 'Successful Activation' Page.

Otherwise you will have to enter your license key to activate Gentics Portal.Node

2.2.6. Unattended (Silent) Installation

The Gentics Portal.Node SDK installer (on all platforms) supports unattended installation by providing a XML file with instructions.

You can use the following example XML to tweak for your needs. The example given is for a windows installation - save it as unattended.xml and start the installer using:
`setup.exe` [*`unattended.xml`*]
On other platforms you can use
`java -jar` [*`Gentics Portal.Node SDK - installer.jar`*] [*`unattended.xml`*]

```
<AutomatedInstallation langpack="eng">
```

```
        <com.izforge.izpack.panels.HTMLInfoPanel/>
        <com.izforge.izpack.panels.TargetPanel>
        <installpath>C:\Program Files\Gentics Portal.Node SDK</installpath>
        </com.izforge.izpack.panels.TargetPanel>
        <com.izforge.izpack.panels.ShortcutPanel>
        <programGroup name="Gentics Portal.Node SDK"/>
        <shortcut mimetype="" createForAll="false" encoding="" type="2" url="" group="fa
        <shortcut mimetype="" createForAll="false" encoding="" type="1" url="" group="t
        </com.izforge.izpack.panels.ShortcutPanel>
        <com.izforge.izpack.panels.SummaryPanel/>
        <com.izforge.izpack.panels.InstallPanel/>
        <com.izforge.izpack.panels.FinishPanel/>
</AutomatedInstallation>
```

2.3. Manual Installation

This chapter will explain the contents of the Gentics Portal.Node SDK CD. You can then use the installation guide in the Reference Documentation for the server installation to get your development environment up and running manually. See Chapter 1, *Introduction* for an overview over all documentation resources.

Table 2.2. Installation CD - Directory structure

documentation/	All documentation to suit your developer needs. Includes the Gentics Portal.Node SDK Guide itself,the reference docs and the API.
licenses/	3rd party licenses distributed with Gentics Portal.Node SDK.
src/	3rd party sources distributed with Gentics Portal.Node SDK.
sdk/	Files you need for manual installation are placed here. • `Portal.Node.war`: The Gentics Portal.Node SDK Portal webapp. • `gentics-portalnode-3-sharedlibs.zip`: libraries for your applicationserver. • `SamplePortletapplication.jar`: a sample portletapplication with sources. Simply import into your IDE as web application and deploy it. • `MyPortletapplication.jar`: put your own new developments here. • `gentics-portalconnector-3-example.zip`: Simple Gentics .Node PortalConnector Examples. (See README.txt in this zip file for more information) • `gentics/`: configuration files for the Gentics Portal.Node SDK Portal (should be put into the path represented by com.gentics.portalnode.confpath - usually in `tomcat/conf/gentics`.
setup.exe	Windows Installer, which leaves you with a full-blown IDE for Gentics Portal.Node consisting of Java, Eclipse, Tomcat and the SDK sample files - ready to use!
LICENSE.txt, .rtf	The Gentics license disclaimer.
README.txt	The first thing to read when installing Gentics Portal.Node SDK. It refers you to this document.

2.4. Activation

After startup, your installation needs to be activated. This process is pretty easy. You are prompted to enter your License Key, which is printed on your Gentics Portal.Node SDK disc, or can be optained by your local Administrator. During activation basic information about your hardware and software configuration are used to validate your license key. For further information please visit http://www.gentics.com

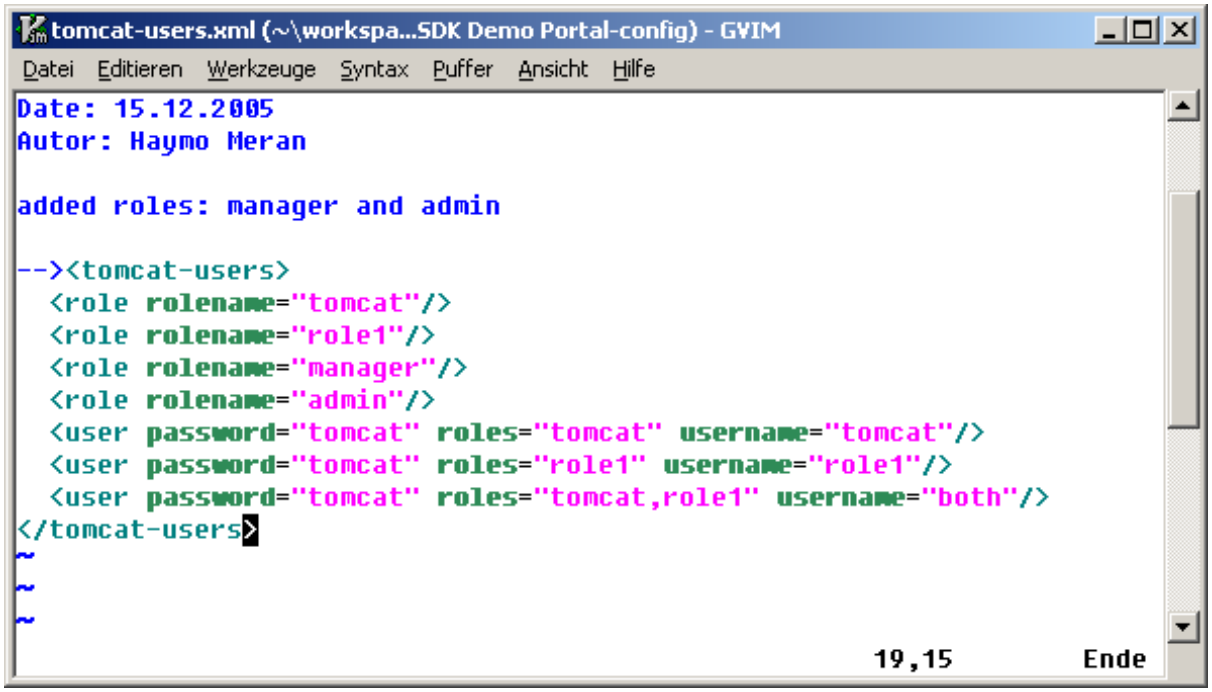
2.5. Manager Configuration

In order to gain access to the Tomcat manager application, you need to create a tomcat-user account with role `manager`. The Tomcat manager application is needed to

- Deploy and manage web-applications
- Deploy portletapplications with the AdministrationPortlet

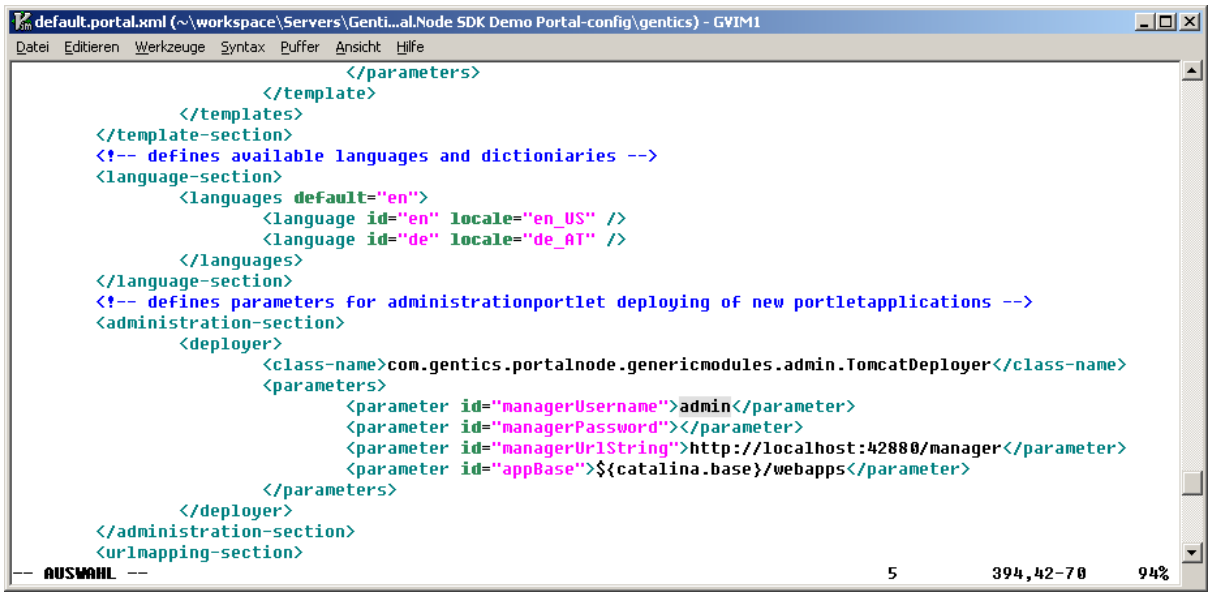
Open the file `Servers/Gentics Portal.Node SDK Demo Portal-config/tomcat-users.xml` in an editor and add a new user with the role `manager`.

Figure 2.12. Create Tomcat User



Also enter the name and password of the user into the file `Servers/Gentics Portal.Node SDK Demo Portal-config/gentics/default.portal.xml` in the `<administration-section>`.

Figure 2.13. Enter Tomcat User in Administration Section



2.6. Optional Configuration

After a basic installation your Gentics Portal.Node SDK is ready to be used out of the box. This chapter deals with further optional configuration, like using other database systems than HSQLDB.

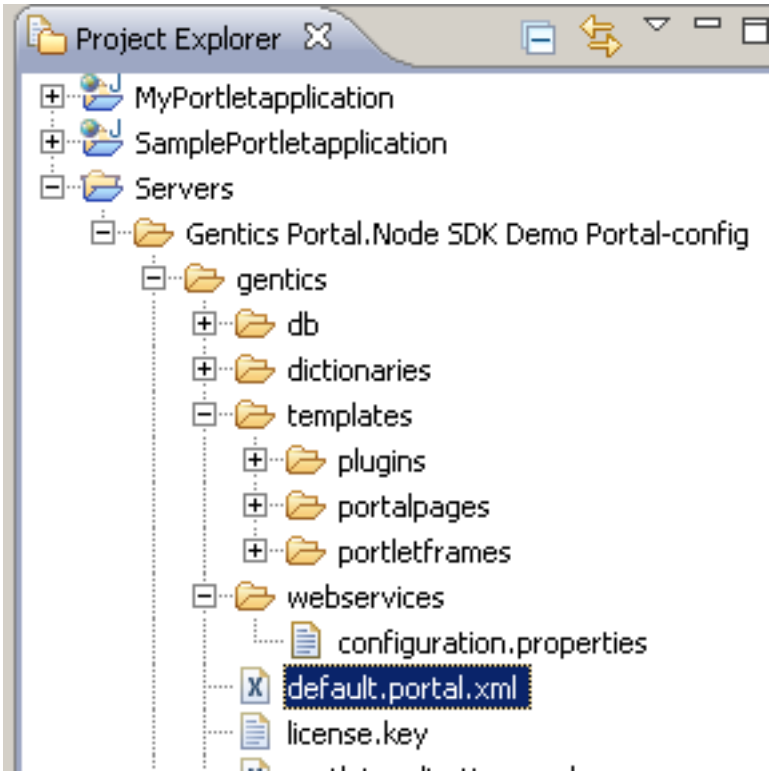
2.6.1. Database Configuration

To use an arbitrary database other than HSQL with Gentics Portal.Node SDK you need to:

1. Configure your Database and import the Gentics .Node ContentRepository database dump specific to your Database as well as the Gentics Portal.Node SDK data dump - Both can be found on the Gentics Portal.Node SDK CD in `documentation/database`.

2. Download the JDBC Driver for your Database and place it into your `tomcat/shared/lib` directory.

3. Modify the datasource settings to adjust the JDBC Connection URL and Driver Class. This can be done in the `<datasource-handles>` section of the Portal Configuration (`default.portal.xml`).



The syntax of the JDBC Connection URL as well as the driver class can be found in the documentation of the JDBC Driver library.

Chapter 3. Getting Started

3.1. Experience the Portals

Gentics Portal.Node SDK comes with two portals ready to be launched off the box. Right from the welcome screen you're able to launch both of them with a single click, or by using the Servers view respectively. To get to know how you reach the Servers view and launch the portals you might want to check out one of our Cheat Sheets (Section 3.2.2, “Cheat Sheets”) called "Welcome to Gentics Portal.Node SDK", wich can be accessed via "Help > Cheat sheets... > Gentics Portal.Node cheat sheets" from the menu.

3.1.1. Gentics Portal.Node SDK Demo Portal

The SDK Portal will start you off with a whole load of examples, ranging from very basic "Hello World" portlets up to far more sophisticated Ajax implementations. Thus we hope to have you glimpse on all the cool things you can achive with Gentics Portal.Node.

The Portal itself consists of two Portletapplications named MyPortletapplication and SamplePortletapplicaion which can be accessed from the Project Explorer on the left. In conjunction with a proper server configuration these elements make up the Gentics Portal.Node SDK Demo Portal Server found in the Servers view. MyPortletapplication is intended for you to have a crack at Gentics Portal.Node whilst SamplePortletapplicaion demoes various features and the ways to implement them. You may add your own portlet application at any time by drag'n'dropping it right on a desired server. You might also want to hop over to Chapter 4, *Development* to get a more in-depth introduction.

3.1.2. Gentics Enterprise Portal

Together with our Gentics Portal.Node SDK we deliver the Gentics Enterprise Portal, which demoes some features and twists that can be achieved with Gentics Portal.Node, that may act as a hint for your own implementations. Amongst many some of the features we're really proud of are:

- Drag'n'drop-style configureable sidebar which holds your applications
- smooth integration of applications as portlets or via iframes and clipping
- multilanguage content portal

All you have to do ist launch the Gentics Enterprise Portal Server from your Gentics Portal.Node SDK Server View! In-depth coverage on how the Gentics Enterprise Portal works can be found at http://www.gentics.com/infoportal .

3.2. Wizards, Cheatsheets, Help & Examples

To assist you in your quest for fame, glory and smooth portal implementations we provide you with a solid set of tools, which will get you started in no time!

3.2.1. Wizards

Wizards, despite helping dwarfish, furry-feeted, sissy boys to find plain pieces of jewelry, will also be kind to assist you at some tasks which they organize in a step-by-step fashion. Access them by right clicking on a folder from the Project Explorer selecting "New > Other... > Gentics Portal.Node". You'll find a list of Wizards available:

- Gentics Portal.Node View

- View Portlet
- Portlet

3.2.2. Cheat Sheets

Cheat Sheets are step-by-step guides, which are capable of handling some of the steps explained auto-magically for you, executed on click. Use them to learn on how to accomplish various standard tasks within Gentics Portal.Node SDK in the most efficient way. Cheat Sheets are available on the following topics:

- Customizing templates for view components
- exporting and deploying Portal Applications
- adding a new custom Datasource
- adding a new Generic Portlet
- adding a Portal Project
- adding a Portlet Application
- adding a ViewPortlet
- Gentics Portal.Node SDK Welcome Guide

3.2.3. Eclipse Help

The Eclipse Help system is your starting point to learn how to work with the Eclipse IDE. A good point to start from is Eclipse's "Getting Started" guide, wich can be found at "Help > Help Contents > Workbench User Guide > Getting Started". But there's much more to be found, since the whole Gentics Portal.Node Manual is compiled and integrated within the Eclipse Help system at "Help > Help Contents > Gentics Portal.Node Manual. So be sure if you're stuck: help just a click away at your fingertips.

3.2.4. Examples

Furthermore than in-depth explanations on how Gentics Portal.Node and it's components are being used, the reference documentation provides active links, which will drop you right into the action, and launch a split view which demonstrates the current topic, ready for you to have a stab at it. You can edit the example in the upper viewport, and experience the outcome of your changes by reloading the browser in the lower one. Don't hesitate to fool around - each and every one of these examples can be reset, as Gentics Portal.Node SDK will automatically detect manipulation of those examples and inform you in case, or by resetting the whole Gentics Portal.Node SDK.

Chapter 4. Development

4.1. Introduction

You are done with the installation? It will help, understanding how the SDK Portal is structured, if you read the Structure section first. Otherwise continue:

In the following section a selection of sample portlets are explained. The sourcecode of theses examples can be found within your Gentics Portal.Node SDK installation. You may use the example selector of your SDK Portal to navigate through the examples. Of course you can also use the interactive links which are usable within your Gentics Portal.Node SDK Help.

Note

Please note that all examples may be used at your own risk. Instead you should consider them as additional help to the reference documentation when implementing your projects.

We will start with simple “Standard Portlet Examples” (Section 4.2, p.27) , including the well-known Hello-World -Portlet, head on to “GenticsPortlet and ViewPortlet Examples” (Section 4.3, p.29) and finally reach some “Portal Feature Examples” (Section 4.4, p.36) like AJAX portlets. After reading the explanation of each portlet in this document, we suggest to take a look at the portlet itself and its inline Javadocs. Fooling around, changing things or reimplementing the example by heart will be a great help getting used to the Gentics Portal.Node SDK and learning how to work with it.

4.2. Standard Portlet Examples

This section shows some examples of portlets, that strictly follow the portlet standards JSR 168: Portlet Specification [http://jcp.org/en/jsr/detail?id=168] and JSR 286: Portlet Specification 2.0 [http://jcp.org/en/jsr/detail?id=286] .

4.2.1. Hello World

This chapter covers a simple HelloWorld portlet example. Look for the `HelloWorld.java` class in the example sources of your SamplePortletapplication. All other examples are stored in your SamplePortletapplication. It simply writes text into the response. If you want to create your own new HelloWorld Portlet, follow these steps or use the appropriate eclipse cheat sheet. You can launch an action within your Gentics Portal.Node SDK help at this place. You can also find an existing HelloWorld portlet within the Gentics Portal.Node SDK Portal.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/HelloWorld.java`

1. Create a new class `MyHelloWorld` next to the existing `HelloWorld.java` which will extend `GenericPortlet` .
2. Implement the method `doView()` . This is the method that will be called by Gentics Portal.Node in order to let the portlet render its content in the default windowstate `view`.
3. Instantiate a writer using the method `getWriter()` from the renderResponse object and print any line you want: `renderResponse.getWriter().println("my first portlet!");`
4. Register the module in the Portlet-Descriptor file `portlet.xml`

Example 4.1. Hello World XML

```
<portlet>
  <portlet-name>myHelloWorld</portlet-name>
  <portlet-info>HelloWorld</portlet-info>
  <portlet-class>com.gentics.portalnode.sdk.myHelloWorld</portlet-class>
  <supports>
    <mime-type>text/html</mime-type>
  </supports>
</portlet>
```

5. Place the portlet somewhere in the portlet entities file `portletentities.xml`, eg. above the existing HelloWorld:

Example 4.2. HelloWorld pnode

```
<pnode type="myHelloWorld" id="myHello">
<style>default</style>
</pnode>
```

6. Place the portlet entity in at least one portal page in the portal configuration file `default.portal.xml` in the `pages-section` .
7. Restart sdk portal server and take a look at the new portal.

4.2.2. Resource Serving Example

This simple example demonstrates how JSR 286 Resource Serving can be used.

The ServeResourceExample Portlet, once placed inside a Portal Page will output textual content with special css styling applied. Thus the browser would render white text upon white background which cannot be read by the user, followed by a red warning text. Incorporating the portlet's serveResource() method (q.e.d.) a simple stylesheet is applied, which turns the white text to readable black, and hides the red warning text. So if you are able to read the black text when displaying the portlet, everything went smooth, and the resource (the stylesheet file in our case) was served properly.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/ServeResourceExample.java`

4.2.3. Events Example

This is a simple example, demonstrating how the event handling works according to the JSR 286 specification.

The example consists of two portlets:

- `EventSenderPortlet` which sends the event (every time the action URL is triggered).

  You can find this file in your Gentics Portal.Node SDK installation.
  Project: `SDK SamplePortletapplication`
  Path: `WEB-INF/examples/com/gentics/portalnode/sdk/events/EventSenderPortlet.java`

- `EventReceiverPortlet` which counts, how many times the event was caught.

  You can find this file in your Gentics Portal.Node SDK installation.
  Project: `SDK SamplePortletapplication`
  Path: `WEB-INF/examples/com/gentics/portalnode/sdk/events/EventReceiverPortlet.java`

In the deployment descriptor `WEB-INF/portlet.xml` , the event has to be defined as `<event-definition>` along with the definition, which portlets support sending (`<supported-publishing-event>`) and which support receiving (`<supported-processing-event>`) the event.

When you maximize the EventReceiverPortlet, such that the portlet is rendered twice in the page, you will notice that for every click on the event sender action URL, the counter in the EventReceiverPortlet is incremented by two. This is because the portlet receives the event for every visible instance (on every page).

4.2.4. 3rd Party Portlet

Covers integration of existing 3rd party portlets.

Copy the class files of the 3rd party portlets into `WEB-INF/classes` and eventually needed libraries into `WEB-INF/lib` . Typically, portlets will be delivered with additional web files (like images, jsp files, html files, etc.). Copy all those files into the root folder of the project which serves also as web folder of the application. Insert the content of the portlets' Portlet-Descriptor into the general Portlet-Descriptor file. Place the module into the portaltemplate sample.portaltemplate.xml Restart SDK Portal Server.

There are two 3rd party portlets already integrated with the SDK, the Bookmarks portlet and the Notepad portlet. Both are selfexplaining, just view them in portletmode help to get a short description.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `bookmarkportlet/content.jsp`

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `notepad/content.jsp`

4.3. GenticsPortlet and ViewPortlet Examples

4.3.1. SimpleTemplateExample

The SimpleTemplateExample is implemented as an AbstractGenticsPortlet, and renders a very basic Velocity template. The template processor is provided with the template's id to be loaded, and renders the template whilst taking the portal's dictionaries into account. Take a peek into SimpleTemplateExample.java to get to know how the template processor is used! You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/SimpleTemplateExample.java`

4.3.2. SimpleViewExample

This chapter will lead you through the process of creating a simple ViewPortlet using two views.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/SimpleViewExample/edit.xml`

The cheatsheet New ViewPortlet guides you through all steps to create a new ViewPortlet. These include

1. Opening the wizard New View Portlet
2. Choosing the Portlet Application Project, for which the new portlet shall be created.
3. Choosing the name of the portlet and of the default view.
4. Creating an entity of the new portlet in your portal configuration.
5. Position the new portlet entity in one of your portal pages.

You can launch an action within your Gentics Portal.Node SDK help at this place.

The xml code of the first view `edit.xml` is listed below:

Example 4.3. SimpleViewExample - edit.xml

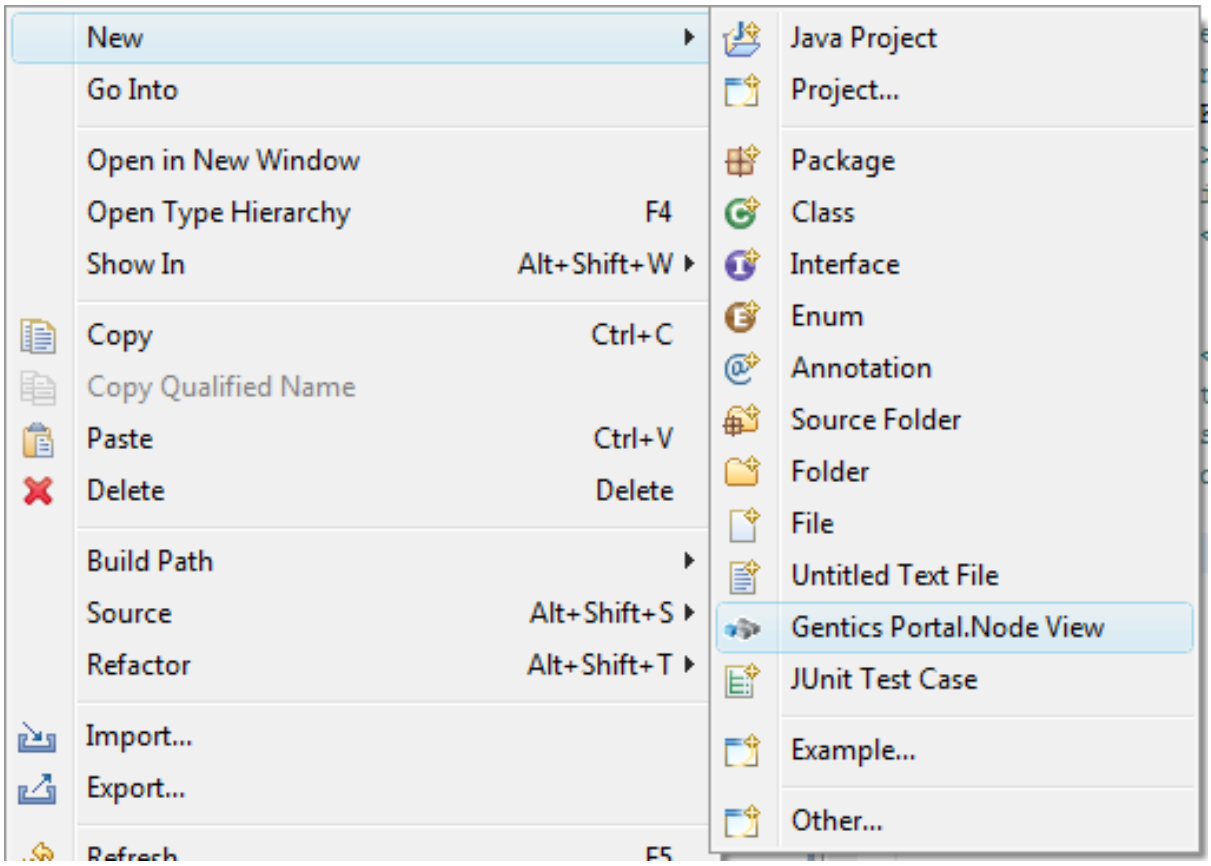
```
<view id="edit">
  <textcomponent id="name">
    <label>Name</label>


  </textcomponent>
  <buttoncomponent>
    <label>OK</label>
    <actions>
      <action class="GeneralViewAction">
        <parameters>
          <parameter id="set">views.show.components.info.data=
                view.components.name.data</parameter>

          <parameter id="showview">show</parameter>
        </parameters>
      </action>
    </actions>
  </buttoncomponent>
</view>
```

Defines a simple `TextComponent` which prompts the user for his name.
This sets the value of the `textcomponent` of id `info` in the view with the id `show` to the specified value. (In this case the value of the component `name` of the current view.)

This will create a simple view with only two components. An input field for your name and a button to confirm it.

With the help of the wizard Gentics Portal.Node View, you can then create the second view.

Figure 4.1. Select "New|Gentics Portal.Node View"



Create the view `show.xml` with the following contents:

Example 4.4. SimpleViewExample - show.xml

```
<view id="show">
  <informationcomponent id="infolbl">
    <label>Your Name</label>
    <reference>info</reference>


  </informationcomponent>
  <textcomponent id="info">
    <label>name</label>
    <visible>false</visible>


  </textcomponent>
  <buttoncomponent>
    <label>Back</label>
    <actions>
      <action class="GeneralViewAction">
        <parameters>
          <parameter id="showview">edit</parameter>


        </parameters>
      </action>
    </actions>
  </buttoncomponent>
</view>
```

Displays the value of the referenced `TextComponent` .
Holds the value displayed by the informationcomponent.
A simple button which will get the user back to the input form to enter his name.

4.3.3. RenderTemplateActionExample

This Viewportlet profits of the RenderTemplateAction. It also uses the scriptEngine JavaScript. You have two Textcomponents that allow processing of input text by using the scriptEngien feature. Every single letter of the input text is going to be checked if there is a 1337 Counterpart and if necessary changed to 1337-Speech. The most common 1337 letters are used here. 1 stands for 'i', 3 for 'e', 4 for 'a', 7 for 't' and 0 for 'o'.

4.3.4. Component Examples

The Component Example covers all display and input components available in Gentics Portal.Node (the VersioningComponent won't be functional because of missing versioning capability of the default datasource). Each component is implemented within a single view. It will help you to understand how to use them in your views. More detailed information about each component can be found within the SDK Reference: (Reference -> Implementation -> Plugins -> View Plugin -> Components)

If you're curious to know what a certain parameter does this is the place to give it a try!

Overview of all Component Examples:

ButtonComponent Example

CheckboxComponent Example

DatasourceSelectComponent Example

DateComponent Example

FileUploadComponent Example

NumberComponent Example

PasswordComponent Example

SelectComponent Example

TextAreaComponent Example

TextComponent Example

DatasourceListComponent Example

DatasourceTreeComponent Example

DownloadComponent Example

FeedbackComponent Example

LabelComponent Example

ListComponent Example

NestedFormComponent Example

TabComponent Example

VersionInformationComponent Example

VersioningComponent Example

4.3.5. DatasourceListExample

This Example contains a simple View which utilises a DatasourceListComponent. It demonstrates how to "overload" a component template, the usage of a datasource (listing the contents of an objecttype) as well as how to define a custom view if the pnode is rendered in "normal" windowstate ("teaserview").

A DatasourceListComponent is used to display a list of objects which are retrieved from a datasource. If you open your sdk sample portal and click "Object Management" -> Datasource: "sample" -> "Choose Datasource", you will see the object type "datasourcetest" which is used in this example. (It is referenced in the View by its type id 1001).

In the Gentics Portlet-Descriptor of the SamplePortletapplication you can see how this example renders a custom view for a specific window state as well as overloads DatasourceListComponent.vm to provide a proper formatting of the date column.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/DatasourceListExample/list.xml`

Example 4.5. DatasourceListExample - Define custom view for a specific window state

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


</classes>
<source><![CDATA[ $views.teaserviewlist
                                          ]]></source>
```

Specifies a custom template to be rendered by ViewPlugin on windowstate 'normal'
Renders the view with the id 'teaserview'.

4.3.6. CalculatorActionExample

This example covers a simple View, with a calculator PluggableAction. It demonstrates how the business logic of portlets can be implemented by programming own PluggableActions, like the SimpleCalculate-Action in this example.

The form is rendered using the ViewPortlet. The calculation is done by a calculator PluggableAction. The View contains two input fields and a text-field which is used to display the result. The Buttons trigger the SimpleCalculateAction and a setter action which writes the result back into the output field.

The input values of the action are read from the view's fieldvalues by using parameter mapping. The operation is given as a parameter. The SimpleCalculateAction calculates the result and stores it into the parameter 'value', which is then written back into the view using the GeneralViewAction.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/calculator/SimpleCalculateAction.java`

4.3.7. CalculatorModule

This example uses a calculator module which does the actual calculations in a plugin-event handler. The form is generated again with a view like in the CalculatorActionExample, but instead of using a specific PluggableAction, the generic TriggerEventAction is used to trigger the onPluginEvent handler in GenticsCalculatorModule.

The eventhandler routine loads the input values directly from its viewplugin. It expects the view to be named 'calc', the input fields must be named 'input1' and 'input2' and the output is stored into the field 'output' by the module. The operation which is calculated is given as a parameter to the event.

The values are read from and set to the view using a resolver and a setter. If a number conversion error occurs, an error message is set to the view.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/calculator/GenticsCalculatorModule.java`

4.3.8. HistoryModule

Covers events and communication between modules through session and parameters.

The GenticsHistoryModule displays a history of all strings which has been send to the module as parameter or as session.

The CalculatorActionExample sends each expression to the history using the GenericViewAction. After the result has been set to the view's output field, the expression provided by the calculator action is written to the 'data' parameter of the 'history'-instance of the GenticsHistoryModule. The history module is notified every time the parameter is set.

The CalculatorModule uses a session attribute to send its expressions to the history module. It uses a list to store the expression strings into the session to prevent overwriting old expression strings which has not yet been fetched by the history module. If the list does not exist in the session, the calculator module creates a new one. The list is fetched by the history module every time the history module is rendered.

The history module stores its history list in the session. If a parameter is changed, the method onParameterChanged is called, where the module checks, if the data parameter has been set and adds the data into its history.

When the module is rendered, it loads the data list from the session and adds its values to the history, if it has any.

The output of the module is rendered using a template processor. An action link is generated to allow the user to clear the history.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/GenticsHistoryModule.java`

4.3.9. ImpsUsageExample

This Example demonstrates a few easy usages of Imps.

Imps are small helpers, providing different functionality an implementer can use in portlets, supporting TemplateEngine2. For more information see the Gentics Portal.Node Reference Documentation.

Since Imps are only used in templates, the interesting parts of this example are in the template files located in `SamplePortletapplication/WEB-INF/templates/modules/ImpsUsageExample/` . Each of these files are template files which render a LabelComponent.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`

Path: `WEB-INF/views/ImpsUsageExample/main.xml`

The following examples are given:

1. Date - Shows how to format the current date.
2. Number - Shows how to convert a number into a human readable filesize string.
3. String - Shows how to trim a long string.
4. RuleMatcher - Shows how to match a given Rule and output if it matches or doesn't.
5. DatasourceQuery - Shows how to retrieve data from a Datasource using a QueryImp
6. Calculator - Shows how to perform easy calculations.

To develop custom imps, take a look at the sample imp L33tImp, which is included in the source folder of the SamplePortletapplication. Since imps must be loaded by the applicationserver's classloader, you need to move imps into a seperate "Simple Java Project" and compile them into `shared/classes` directory of your portal servers configuration path. Refer to the L33tImp's inline and Gentics Infoportal [http://www.gentics.com/Content.Node/infoportal/portal/3.3/eigene_imps_im_gentics_portalnode_sdk_entwickeln.php] documentation for further information.

4.3.10. ResolvableDatasourceExample

This Example demonstrates the usage of the ResolvableDatasource and custom Javabean classes for integration of custom data sources (databases) into Gentics Portal.Node views.

Implemented as an AbstractGenticsPortlet a new ResolvableDatasource object is instantiated and prefilled with two UserBeans. After prefilling it the ResolvableDatasource is put inside the portlet session. Rendering is handed over to Gentics Portal.Node's ViewPlugin, which renders a very basic list view, and refers to the ResolvabeDatasource created before from the portlet session. So all the listed and editable content displayed in the ResolvableDatasourceExample was generated on the fly, without any database engines being used.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/resolvable/ResolvableDatasourceExample.java`

4.3.11. AdvancedViewExample

Covers an advanced view.

See how it all fits together with our advanced AdvancedViewExample, which totally focuses on how components would be arranged in an real-life example by demoing an in-depth human resource management usecase. There is no whatsoever functionality included so far, but this complex view should be able to give you a feeling of what can be accomplished using those powerful tools Gentics Portal.Node provides you with.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/AdvancedViewExample.java`

4.3.12. GoogleSearchExample

Covers integration of an abitrary web service, using Google search for this example.

This example uses Google's API to implement a search portlet. When viewing the portlet a simple search view is rendered via the ViewPlugin, which just displays an input field four your search term along with a submit button. Hitting this button will finally trigger a PluginEvent keyed "search" which is handled by the portlet's onPluginEvent method, and finally queries Google for the search result. Query results are displayed in a maximized view of the portlet, rendered via Google's autogenerated classes.

After filling in all the information asked for in the form, the AdvancedViewExample renders it's content using a basic velocity template, to give you an overview of the data provided. This is the point where you come into play, turning our example into an all decked-out web application.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/GoogleSearchExample/GoogleSearch.xml`

4.3.13. RSSDisplay

Demonstrates how to use a simple XSLT in combination with URLLoaderActions, XSLTRenderAction and CreateResolvablesAction to show a RSS feed in a default DatasourceListComponent.

To display an information from an RSS stream, the contents obviously have to be loaded from the web, and transformed into a kind of object, which Gentics Portal.Node is able to handle, thus making Resolvables our choice. Here is how it works:

First a RSS stream is loaded from an URL using the URLLoaderAction along with a XSLT stylesheet. In the next step those contents are transformed using a XSLTRenderAction incorporating both the RSS file and the XSLT stylesheet, resulting in XML output which is handed over to the CreateResolvablesAction. In the last step the items from the RSS feed are displayed by setting them to a DatasourceListComponent using a GeneralViewAction. As these steps may sound confusing in textual form be sure to check out the according view file "rsslist.xml", which should give you an idea on how things work.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/RSSDisplay/rsslist.xml`

4.3.14. ViewWizard

Covers an example on how views generated from the sample datasource objects look like. If you want to try the ViewWizard yourself head over to the AdministrationPortlet's Object Management dialouge, select some objects and hit "Generate View". Enter "ViewWizard" in the "Portlet ID" field and look how the ViewWizard's navigation view updates right after your overwrite confirmation. Clicking one of the navigation buttons will take you to the list view, where you can create, delete or browse objects.

For detailed information on how to use the ViewWizard read the corresponding section from the Gentics Portal.Node Reference.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/ViewWizard/navigation.xml`

4.4. Portal Feature Examples

4.4.1. New portal page

This example explains how to create an new portal page with its own template and frame templates. We assume that your portal configuration already uses portal pages. If this is not the case please read the migration guide for portal pages.

You can find this file in your Gentics Portal.Node SDK installation.
Server: `Gentics Portal.Node SDK Demo Portal`
Path: `gentics/default.portal.xml`

1. You can define new pages in the portal configuration file default.portal.xml of your server in the <pages-section>. Your portal page can use one or more positions that will contain the portlets.

2. In order to load a custom template you must define a template loader. This is done in the <template-section>. The default configuration has already a loader for portal page templates with the id "portalpage". It loads the template files from ${com.gentics.portalnode.confpath}/templates/portalpages (in your SDK this means Servers/<your-server>/gentics/templates/portalpages).

3. We want our page "mypage" to use template file "mypagetemplate.vm". This is also configured in the template section:

4. Given the configuration above and the template loader "portalpage" you have to save your template to file "mypagetemplate.vm" in the folder ${com.gentics.portalnode.confpath}/templates/portalpages.

5. The list of variables which can be used in the portal page template can be found in reference documentation of Gentics Portal.Node. You can also have a look at default template "demo.vm" in you SDK portal.

6. Furthermore you can customize the portlet frames. Again you will need a template loader. The default configuration defines a template loader "portletframe" which loads the template files from ${com.gentics.portalnode.confpath}/templates/portletframes.

7. The following code will cause the the portal to use the template "myframe.vm" for all portlets in "mypage" in windowstate normal: You can also define a custom frame template for a specific portlet, position or style. See reference documentation for more information about how the matching algorithm for template selection works.

4.4.2. GenticsLoginModule

This example covers usage of the Gentics Login Module.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/templates/modules/URLMappingExample/main.vm`

Though there may not be that much eyecandy on this example it demonstrates well how the <authentication-section> within the portal configuration file is used together with the GenticsLoginModule. User profiles stored in our sample HSQL database are able to authenticate using their firstname along with a default dummy password which is currently set to "password". Not that tricky. So try logging in using "Max" as login name. The PNAuthenticationManager will handle the rest for you, and authenticate according to the given loginrule.

4.4.3. URL Mapping Example

This example demonstrates the usage of the URL mappings.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `Gentics Portal.Node SDK Demo Portal-config/gentics/default.portal.xml`

The first thing to do when you want to make URLs beautiful, is to enable this feature in your portal configuration file by setting the portal parameter ***`portal.urlmappings`*** to `true` (the default is `false`).

Normally most links in Gentics Portal.Node will have many GET parameters. This is neither beautiful nor very practical since such URLs tend to be quite long and unreadable. To avoid this you can configure your portal to use URL mapping. All relevant configuration options are stored in the portal configuration file:

In general the mappings affect all URLs rendered by Gentics Portal.Node including PLinks and URLs in GenticsContentModule. To keep things simple in this example we use the URL mappings to change the URLs for maximizing and minimizing the portlet.

Without the URL mappings the maximizing URL would look like http://localhost:42880/Portal.Node/portal?gentics.rm=URLMappingExample&gentics.ws=maximized&gentics.pb=right

- The parameter `gentics.rm=URLMappingExample` specifies the portlet.
- `gentics.ws=maximized` specifies the new window state
- `gentics.pb=right` specifies the position

In the urlmapping-section of the portal configuration we have defined a mapping to transform the URL. In our example there is only one configured mapping however more different mappings can be used.

1. When a URL is rendered Gentics Portal.Node must decide whether a mapping should be applied. This is done using the hidden urlparameters. In our case we have defined, that the urlparameter `gentics.rm` must have the value `URLMappingExample` and will be hidden. This means that only URLs with the urlparameter `gentics.rm=URLMappingExample` will be transformed using this mapping (e.g. our URL for maximizing above).
2. The pathtransformation defines how the transformed URL will look like. It consists of several segments.
   a. The first segment has a static value. Therefore the transformed result will always start /URLExample/
   b. The second segment is a dynamic one and uses the value of the urlparameter `gentics.ws` . In our case this would result in /maximized/
   c. The last segment is also dynamic and uses the value of `gentics.pb` , thus resulting in /right/

   The resulting transformed URL is created by simply putting the entries together - in our case http://localhost:42880/Portal.Node/portal/URLExample/maximized/right
3. When you click on this URL then Gentics Portal.Node has to reconstruct the original URL with its urlparameters. So how can the original urlparameters be recognized? How does Gentics Portal.Node know that "maximized" is the value for the parameter `gentics.ws` The key is to recognize the mapping that was used to transform the URL. The static segment "URLExample" can be seen as a hint to the portal. It helps to determine which mapping was used to render the URL. Therefore it is recommended to embed a static segment in the transformed URL if your portal is configured to use multiple different mappings.

Note

The portlet itself and its implementation is the same as the “SimpleTemplateExample” (Section 4.3.1, p.29) but it uses a different template. The implementation of the portlet is not important. The only relevant part is the configuration of the URL mappings.

4.4.4. Page Selector Example

This example demonstrates how pages can be selected by a simple view.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/PageSelector/pageselect.xml`

4.4.5. AjaxExamples

This section walks you through all Ajax examples provided with the Gentics Portal.Node SDK.

4.4.5.1. Hello World

A very basic "Hello World" example incorporating Ajax functionality, which can be accessed via the Ajax portlet from the SDK portal. Believe it or not - the text which pops up after clicking the button is requested and delivered directly from the server, and not generated by the client, as one might expect. So at this time there's more under the hood than can be seen at first glance. When checking the helloworld.xml view file, you come to know that a CallableAction is in charge for delivering the highly anticipated "Hello World!"-string to you.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/AjaxExample/helloworld.xml`

4.4.5.2. Autocomplete

The Autocomplete example features a typical Ajax usecase found in most Web 2.0 applications. When typing the portal will automagically be queried for matching contents from within our sample database, which are displayed in a box. Implementation is quite straightforward, which uses jQuery to invoke a CallableAction from the autocomplete view. Take a look at the custom template for the autocomplete input field, which takes care of querying the server and response handling all by itself, whilst keeping things easy with only very few lines of JavaScript.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/AjaxExample/autocomplete.xml`

4.4.5.3. Load & Store

The Load & Store example demonstrates a classical usecase for Ajax applications. Items can be loaded from a datasource by specifying their contentid. After changing some of the properties available, they might be stored back to the database, without reloading the page even just a single time.

Tip
Try using contentid "1001.5" or "1001.6" to find database entries for your testing needs!

As one might expect this example relies heavily on usage of callable actions for both load and store operations. The form displayed itself is rendered using a custom template (loadandstore.vm) for both edit and query tasks. The load operation is implemented quite straightforward, and will also handle error cases like objects which could not be found or datasource errors from within the callable action itself. The store operation is not complicated either and works just as you'd implement a non-Ajax store operation. There is only one difference, as the object to be stored has to be built from request variables.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/AjaxExample/loadandstore.xml`

4.4.5.4. SimpleAjax Custom Form Example

This example demoes a very basic Ajax implementation, which does not rely on Gentics Portal.Node views. Instead the form is generated entirely from the SimpleAjax class, which delivers outputs on call of it's doView() method, and query result via serveResource().

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/SimpleAjax.java`

4.4.5.5. AjaxAddRemovePortlet

This portlet allows you to add and remove portlets to your portal page. It is a ViewPortlet with three views. In order to add the portlets via AJAX without reloading the whole page you have to activate portlet reloading and replacing of links for this portlet. The SDK example portal has portlet reloading activated globally in the `default.portal.xml`. It can also be configured individually for each portlet in the `portletentities.xml`.

The addportlet-View allows you to add new portlets. It consists of two simple select components where the user can choose the portlet and the position where the portlet will be placed. The [Add]-button reads the values of the select components and adds the portlet to the selected position of the current portal page by simply using the "+=" operator. Thanks to the portlet reloading your portal will take care of everything and no JavaScript programming is necessary in this example. You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/views/AjaxAddRemovePortlet/addportlet.xml`

The removing of portlets is also very simple. To show the close button on portlets you have to define a new action. This is done in the portlet entities configuration file. The reaction on the close event is also configured in the portlet entities file. It resolves the the portlet that caused the event, its position and the portal page and removes it from the list of portlets in that position using the "-=" operator.

4.4.5.6. Portlet Reloading

Portlet reloading means that all links and forms are replaced with ajax calls to Gentics Portal.Node which will only render and replace the affected portlets, not the whole portal page. This results in a much faster response time.

Procedure 4.1. Setting up Portlet Reloading

1. **Adding Javascript libraries to portlet application**

   In the SDK SamplePortletapplication, you find our example implementations of the client-side scripting necessary for portletreloading. The Javascript files are all located in the path `WebContent/ajax`. To use those Javascript libraries in your own implementation, you can simply copy the following files into your own portlet application:

   - `jquery-1.2.6.js` contains the 3rd party jQuery library
   - `jquery.form.js` contains the form plugin for jQuery (needed to manipulate forms)
   - `gentics-ajax.js` contains the generic code to parse and interpret responses from the Gentics Portal.Node server to AJAX requests
   - `gentics-portletreload.js` contains the portletreloading specific code to substitute the portlets in the DOM tree with new rendered versions.

2. **Include Javascript libraries into the portal page template**

   Of course, the javascript libraries need to be included in the html output for your pages.

3. **Initialize the client-side javascript for portlet reloading**

   The simple javascript code `GENTICS.PortletReload.create();` will enable portlet reloading.

4. **Add $portlet.portletreload to portlet frame templates**

   The client-side javascript code will need special css classes in the outermost HTML tags of rendered portlets to identify the DOM parts, which have to be replaced by new rendered portlets. It is therefore necessary to add `$portlet.portletreload` to the *`class`* of the outermost HTML tag in all

of your portlet frame templates.

5. **Add css class to positions in portal page template**

   Also the positions need to be indentified in the DOM tree by css classes, which need to be added manually. The css classes need to be named `gentics-portletreload-position-[positionid]`.

6. **Enable portlet reloading in portal configuration**

   Portlet Reloading must be activated by setting the parameter *`portal.portletreload`* to `true` in the portal configuration file `default.portal.xml` .

It can then be deactivated for single portles by using the *`gentics.portletreload`* = `false` parameter as seen for example in the pnode of the `LanguageSwitcher` portlet (`portletentities.xml`).

4.4.5.7. Drag'n'Drop

Our demo Drag'n'Drop implementation shows you what can be achieved with our new Ajax features by means of a pretty cool usecase. On every one of our example portal pages you are able to drag and drop each and every portlet from left and right portlet positions. Right after the drop occurs positions are stored back to the server, so the portal will remember your positioning, and apply it even after reloading the page. Furthermore your favoured positioning is put to the user object and stored back to the database. Therefore on your next login portlet positions will be restored.

You can find this file in your Gentics Portal.Node SDK installation.
Server: `Gentics Portal.Node SDK Demo Portal`
Path: `gentics/templates/portalpages/demo.vm`

4.4.5.7.1. JavaScript libraries needed for Drag'n'Drop

The following list contains all JavaScript libraries necessary for Drag'n'Drop. It consists of 3rd party libraries (jQuery) and demo libraries provided by Gentics.

- `jquery-1.2.6.min.js` jQuery library. jQuery is an AJAX framework, which we use for all our AJAX examples. See http://jquery.com/ [http://jquery.com/] for details on jQuery.
- `jquery-ui-personalized-1.5.2.js` jQuery UI plugin library. See http://ui.jquery.com/ [http://ui.jquery.com/] for details about that. The personalized download contains the following Components:
  - UI Core
  - Draggable
  - Droppable
  - Resizable
  - Selectable
  - Sortable
- `gentics-ajax.js` small library for decoding and interpreting server responses to AJAX calls. See the reference documentation (index term AJAX) for more details.
- `gentics-portletdragdrop.js` encapsulation of the `sortable` features of jQuery to support Drag'n'Drop of portlets.

- `gentics-portletreload.js` another small demo library to support portlet reloading. See “Portlet Reloading” (Section 4.4.5.6, p.40) for details about portlet reloading.

4.4.5.7.2. Enabling Drag'n'Drop for portlets

1. For each position that should support Drag'n'Drop (in our case: `left` and `right`), we embed the rendered portlets into a container tag (in this example: <ul>), which is identified by the class `gentics-portletreload-position-[position_id]` and the id `col_[position_id]`.
2. All portlet frames must have their outermost html tag have the class `sortable` (which marks the DOM element as being draggable) and the id `${portlet.id}` (which will render the portlet's id).
3. Calling the JavaScript method `GENTICS.DragDrop.create()` in the portal pages template binds the necessary jQuery functions to all sortable items inside the specified Drag'n'Drop containers. The method accepts an array of JavaScript objects to specify the sortable positions and the url for the server call to store back the new portlet positions. The sortable positions can have the following attributes:
   - *`position`* : id of the position (e.g. `left`)
   - *`id`* : id of the html container tag of the position (e.g. `col_left`)
   - *`accept`* : jQuery selector to identify all sortable elements (here: `.sortable` , which finds all elements with class `sortable`)
   - *`connectWith`* : array of jQuery selectors of other containers (e.g. `#col_right` to connect the position left with the position right, which means that portlets can be moved from left to right)
   - *`helperclass`* : Name of the class which the drop target "helper" will get (such that the user can see, where he/she is about to drop a portlet)
   - *`handle`* : jQuery selector to identify the "drag-handle" (the part of a portlet, which allows dragging the portlet, e.g. `div.head` to make the portlets draggable when clicked onto the headline)

   The url for the server call is accessible as `$portal.portletpositionchange`.

4.4.5.7.3. How does the portlet Drag'n'Drop actually work

- The source code of the JavaScript function can be found in the library file `gentics-portletdragdrop.js` . It enables sorting for all specified positions and binds the function `GENTICS.DragDrop.updatePositions()` (which is defined in the same file) as callback function when portlets have been dragged to different positions (or reordered in one position).
- The method `GENTICS.DragDrop.updatePositions()` constructs the parameter list for the portal server request to store back the new portlet positioning and finally issues the AJAX request.
- In the callback method of the AJAX call, the method `GENTICS.processResult()` (which can be found in the file `gentics-ajax.js`) is called. This method decodes and interprets the server response. As a result, all portlets that need to be re-rendered due to the portlet position changes will be replaced on the page by means of jQuery DOM manipulation methods.

4.4.6. Gentics .Node PortalConnector Examples

The update file and installation CD contains a file named 'gentics-portalconnector-3-example.zip' which contains simple Gentics .Node PortalConnector examples (Read/Write as well as a JSP usage example).

4.4.7. PCWSClientExample

This example demonstrates the usage of the Gentics .Node PortalConnector WebService to access datasources from Java client programs. The example is a very simple commandline program that uses Apache Axis [http://ws.apache.org/axis/] .

4.4.7.1. Prerequisites

In order to access a WebService you may examine the `*.wsdl` file that defines the services, methods and objects. The `*.wsdl` for Gentics .Node PortalConnector WebService can be obtained via web access using the URL http://localhost:42880/Portal.Node/ws/services/datasource?wsdl .

Note

Please note that you need a valid Gentics Portal Connector Webservice License to use this feature. In the default configuration of your SDK installation, the web service access is disabled for security reasons. Configure a user and passwort in `${com.gentics.portalnode.confpath}/users.properties` . You also need to enable the user within the webservice configuration: `${com.gentics.portalnode.confpath}/webservices/configuration.properties` Take a look at the reference documentation for further information.

With axis, you could auto-generate client classes with the command
java -cp mail-api.jar:activation.jar:saaj.jar:wsdl4j-1.5.1.jar:jaxrpc-1.2.1.jar:commons-discovery-0.2.jar:commons-logging-1.0.4.jar:axis.jar org.apache.axis.wsdl.WSDL2Java -p com.gentics.portalnode.pcws.example -U admin -P secret http://localhost:42880/Portal.Node/ws/services/datasource?wsdl

These classes have already been generated and packaged in the archive `SamplePortletapplication/WEB-INF/lib/pcws-clientexample.jar` .

Note

Although the access to a specific datasource is done through an URL that contains the datasource id, the client classes have been generated without datasource id and can be used to access all datasources that are exposed via web service.

4.4.7.2. Running the Example

The example consists of only the main class `com.gentics.portalnode.sdk.PCWSClientExample` and the runtime libraries

- pcws-clientexample.jar (containing the auto-generated client classes)
- axis.jar
- commons-logging-1.0.4.jar
- commons-discovery-0.2.jar
- jaxrpc.jar
- wsdl4j-1.5.1.jar
- saaj.jar

To run the example, start Tomcat (if not already done before) and then simply right click on the `*.java` file in the package explorer and choose "Run as..." and then "Java Application". Whatch the Console view in your SDK for the program's output.

4.4.8. CsvImportExample

This example covers a standalone commandline program, which uses the CsvDataImportAction to import a csv file into a datasource, and a simple view using the same pluggable action. The example csv file can be found within your SDK installation. Please use `WebContent/WEB-INF/examples/com/gentics/portalnode/sdk/cmdline/csvdata.csv` in your `SDK SamplePortletapplication Project`.

For a detailed description of the csv file header format and the action parameters, see the CsvDataImportAction documentation in the Gentics Portal.Node reference.

4.4.8.1. Commandline example

The datasource to a contentrepository is created using the connection settings specified in the jdbchandle.properties file. The cache.ccf file is used by the contentrepository cache. Before the commandline program can be used, the jdbc connection must be set to an existing contentrepository, which contains the objecttypes from the sdk sample database.

The implementation uses the PortalConnectorFactory to create a datasource, using the settings from the properties file. Then, a new instance of the pluggable action is created, as well as a request object, which holds all request parameters, and a response object, which stores the feedback message the response from the action.

After the action has been executed, the response messages and the ids of all new objects are printed. If the import fails (i.e. any error occurs), all created objects are deleted while the ids are printed.

The example can be run by selecting 'Run As' 'Java Application' or 'Debug As' 'Java Application' from the context menu.

4.4.8.2. View example

The view example can be accessed by the ExampleSelector-portlet. The view contains some simple input fields for the input parameter of the action, a button which triggers the action and a feedback- and a labelcomponent, to display the results of the import.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/cmdline/CsvImportExample.java`

The uploadcomponent provides the uploaded content as data stream without encoding. Therefore, the BinaryToTextAction is used to convert it into text, which uses the given encoding setting to handle special characters. The encoding of the sample csv file is ASCII. If the specified encoding does not match the format of the file, the resulting text may be corrupted.

After the text is successfully transformed, the action imports the content using the given settings. The result messages are then displayed using a labelcomponent.

To import the sample csv file, the object type must be set to 1001, the file encoding to ASCII and the file path prefix must be set to the directory containing the binary files. All other fields can be left empty.

4.4.9. PostProcessor Example

This example shows how the implementation of the interface `com.gentics.api.lib.expressionparser.filtergenerator.PostProcessor` instances can be used in conjunction with the `filter()` function to move contentrepository filter logic from the database into Java code.

The implementation of the portlet `com.gentics.portalnode.sdk.filter.PostProcessorExample` fetches data from a ContentRepository datasource with four different filter rules:

- `object.obj_type == 1001` is a normal filter to get all objects with a given type.

- `filter(object.obj_type == 1001, 'com.gentics.portalnode.sdk.filter.StartsWith', ['firstname', 'B'])` also fetches all objects, but then removes all objects with firstnames not starting with 'B' by letting an instance of `com.gentics.portalnode.sdk.filter.StartsWith` process the intermediary result.

- `filter(object.obj_type == 1001, 'com.gentics.portalnode.sdk.filter.Sorter', 'surname')` fetches all objects and in uses an instance of `com.gentics.portalnode.sdk.filter.Sorter` to sort them by surname.

- `filter(filter(object.obj_type == 1001, 'com.gentics.portalnode.sdk.filter.StartsWith', ['firstname', 'B']), 'com.gentics.portalnode.sdk.filter.Sorter', 'surname')` finally combines the two filters from the previous examples to get a filtered and sorted result.

You can find this file in your Gentics Portal.Node SDK installation.
Project: `SDK SamplePortletapplication`
Path: `WEB-INF/examples/com/gentics/portalnode/sdk/filter/PostProcessorExample.java`

Chapter 5. Packaging

With Gentics Portal.Node SDK you are able to easily package your portletapplications, which can be deployed into the application server using the AdministrationPortlet.

The easiest way to accomplish this is by using the provided ant script located in `MyPortletapplication/tools/packaging`. To package your portletapplication follow these steps in your Gentics Portal.Node SDK Eclipse installation:

Procedure 5.1. Packaging MyPortletapplication

1. Open the directory `MyPortletapplication/tools/packaging` in the Package Explorer. (By default located to the right in the Java Perspective.
2. Rightclick on `build.xml`.
3. Select Run As+Ant Build (**Alt-Shift-X-Q**).
4. Now Refresh (**F5**) the `MyPortletapplication/tools` directory.
5. `MyPortletapplication/tools/out/MyPortletapplication.war` is ready to be deployed in any Gentics Portal.Node installation. Just upload the .war-file using the AdministrationPortlet and you'll be fine. See the AdministrationPortlet section of the reference guide for further information.

Chapter 6. Documentation

Gentics Portal.Node SDK comes with a small documentation framework you can use to document your work.

It is located in `MyPortletapplication/tools/documentation` and uses DocBook. DocBook is an XML dialect used to easily write technical documentation which can be converted to various output formats. Gentics Portal.Node SDK comes with a ant build script which converts your documentation into PDF as well as chunked XHTML.

For a complete documentation about DocBook you may want to refer to the (free) Book DocBook: The Definitive Guide [http://www.docbook.org/tdg/en/html/docbook.html] for further information about how to write documentations using DocBook.

To build your documentation follow these steps:

Procedure 6.1. Building MyPortletapplication Documentation

1. Open the directory `MyPortletapplication/tools/documentation` in the Package Explorer. (By default located to the right in the Java Perspective.
2. Rightclick on `build.xml`.
3. Select Run As+Ant Build (**Alt-Shift-X-Q**).
4. Now Refresh (**F5**) the `MyPortletapplication/tools` directory.
5. You can view your documentation in `MyPortletapplication/tools/out/manual` either `manual.pdf` for the PDF version or `xhtml/index.html` for the XHTML version.

Chapter 7. Miscellaneous

7.1. Uninstalling the SDK

You should backup all your work, since all installationdirectories and subdirectories will be removed completely. Then select the Uninstall icon from your SDK startmenu folder.

7.2. Updating the SDK

By default there are two update checks enabled in Gentics Portal.Node SDK:

- Gentics Portal.Node Update Check: Each time you start Gentics Portal.Node within eclipse it will check for updates by default (see PortalConfiguration).

  If new updates are available it will notify you with a console message. You can then either use the Eclipse Update to update manually or visit the Gentics .Node Infoportal [http://www.gentics.com/infoportal].

- Eclipse Update Check: Gentics Portal.Node SDK uses a custom Plugin within eclipse which will also check for updates each time you launch eclipse. If a new update is available it will let you know with a small dialog box where you can immediately download the new update.

  This update check can be configured in the eclipse preferences: Window → Preferences+Install/Update.

  If you want to manually check for updates or download new updates use: Help → Software Updates → Find and Install...+Search for updates of the currently installed features..

  After a update through the Eclipse Update Manager you can verify the successful installation by using the Gentics Portal.Node update check by watching the console output to make sure the 'New Updates are available' message has disappeared.

7.2.1. Proxy Settings

All update operations work through HTTP, so even if you have no direct Internet access you can configure the Eclipse Update Manager to use an HTTP Proxy.

This can be done in the eclipse preferences: Window → Preferences+Install/Update Select Enable HTTP proxy connection and enter your HTTP proxy host address and HTTP proxy host port.

Note

The update check of Gentics Portal.Node will not use the proxy you configure in the eclipse preferences.

7.3. Resetting your SDK Installation

To Reset your Gentics Portal.Node SDK Installation you can use Help → Gentics Portal.Node SDK → Reset Installation from the menubar in your Gentics Portal.Node SDK Eclipse.

This option will start the 'Custom Update' process. By selecting all options you will be able to reset all workspace projects, reinstall portal webapplication and shared libraries as well as all Gentics Portal.Node specific configuration.

Warning

'Reset Installation' will not remove tomcat specific settings (e.g. `server.xml`) - So if you have manually removed SamplePortletapplication / MyPortletapplication they will not be re-added to your tomcat configuration.

7.4. Gentics Portal.Node Runtime

The Gentics Portal.Node Runtime is one of the core elements of the Gentics Portal.Node SDK. You'll be using it when creating a new server instance for your personal portal project built from the ground up. What it does for you is adding all Gentics Portal.Node libraries you need just by selecting it, so no further hands-on is required.

When updating Gentics Portal.Node SDK you can choose to preserve the old version as separate Runtime, so that you can even have different projects use different versions of Gentics Portal.Node.

7.5. Gentics Portal.Node Server

A Gentics Portal.Node Server represents a specific portal implementation project. You can have multiple Gentics Portal.Node Server instances, each with different configurations, a different set of deployed Gentics Portal.Node Applications and even using different Gentics Portal.Node Runtimes.

The cheatsheet New Portal Project helps you in setting up a new Gentics Portal.Node Server instance. You can launch an action within your Gentics Portal.Node SDK help at this place.

7.6. Gentics Portal.Node Application

A Gentics Portal.Node Application is a specialized Dynamic Web Project, that adds a lot of helpful functionality to use Gentics features:

- Compile with all needed libraries
- Generate a fixed folder structure with skeleton files for the Deployment Descriptors
- Automatic Deployment into a Gentics Portal.Node Server instance by drag & drop
- Wizards to easily create Gentics ViewPortlets, views, etc.

The cheatsheet New Portlet Application helps you in creating a new Gentics Portal.Node Application. You can launch an action within your Gentics Portal.Node SDK help at this place.

7.7. Autocompletion

To benefit from autocompletion you just have to add the correct XML Namespace to the various xml files like views, portletentities configuration file or portal configuration. Use the following URLs:

- Views: `http://www.gentics.com/xml/ns/portal/view`
- Portlet Entities Configuration: `http://www.gentics.com/xml/ns/portal/portletentities`
- Gentics-Portlet Descriptor: `http://www.gentics.com/gentics-portlet-app`

7.8. Customizing Portal.Node webapp in the SDK

The Gentics Portal.Node SDK usually hides the Portal.Node web application from the user in the form of a "Gentics Portal.Node Runtime". It is still possible to customize the Portal.Node.war before deployment by creating a folder named `customwebapp` in the server's configuration folder (Usually within the 'Servers' project). All files in that folder will overwrite there equivalents in the `Portal.Node.war` . For example to customize the web.xml to introduce new security constraints you need to create a file called `customwebapp/WEB-INF/web.xml`.

You can find the original Portal.Node.war in the installation directory of Gentics Portal.Node SDK `eclipse/pn_updates/builds/<buildnumber>/sdk/Portal.Node.war`.

Please see the Gentics Portal.Node Reference for details on how to customize the Portal.Node web application in a server environment. (Indexterm 'Customizing Portal.Node.war').

7.9. Accessing Gentics Portal.Node libraries

If you need access to the Gentics Portal.Node libraries for a non-web module java project or in an ant script you can use special variables to access the Gentics Portal.Node .jar files from the latest build.

- Java Classpath Variable: When modifying the Java Build Path of a java project click on 'Add Variable' and select 'GPN_LIBS' from the popup and click 'Extend..'

  If you add `gentics-portalnode-api.jar` to your classpath you can associate it with the javadoc API by right clicking on the file -> Properties -> Javadoc Location -> and using the URL: http://www.gentics.com/help/topic/com.gentics.portalnode.sdk.doc/misc/doc/apijavadoc/

- Ant Property: The directory of the latest build file can be accessed using ${com.gentics.portalnode.sdk.sharedlibs}