jBPM Tools Reference Guide

Version: 3.1.7.GA

Introduction

All developers and process analysts who are beginning to use JBoss jBPM should read this Getting Started guide. It will give them a jumpstart showing how to create a process definition.

1.1. Preface

This document introduces the use of the JBoss jBPM Graphical Process Designer (GPD) to create workflow processes. It will help first time users with the following tasks :

- Install the JBoss jBPM GPD Eclipse plugin available from the JBoss jBPM download area
- Set up a Java project in Eclipse and prepare it to do test driven process development
- Using the creation wizard to create an empty process definition
- Use the designer palette to draw the first processdefinition
- Show how the xml processdefinition can be inspected as an xml file
- Set up a Java project in Eclipse and prepare it to do test driven process development
- Write an example process test case

If you have questions, please feel free to contact *Koen Aers* [mailto:koen.aers@jboss.com] or *Tom Baeyens* [mailto:tom.baeyens@jboss.com] for more information.

1.2. Feature list

JBoss jBPM is a workflow that enables creating and automatization business processes. Look at the list of features below to understand its main functionality.

Table 1.1. Key Functionality for JBoss jBPM

Feature	Benefit
jBDL support	Enables managing workflow processes as well as human tasks and interactions between them. jBDL combines the best both Java and declarative process techniques.
Support of Graphical Process Designer (GPD)	Is used for simplifying declarative process development and visualizations of all actions.
Project Creation wizard	Allows to create a new jBPM template project that already includes all advanced artifacts and core jBPM libraries.

Feature	Benefit
Rich palette of pre-build process nodes	Provides process-building functionality and gives opportunity even non-programmers to develop processes.
Support of XML code view	Shows the corresponding XML that's generated automatically in the Source view of the process definition editor when developing the process.
Properties view	Facilitates configuring and editing of all nodes properties.
Interaction with all of the J2EE based integration technologies including Web Services, Java Messaging, J2EE Connectors, JBDC, EJBs.	Enables implementation, provides better functionality and flexibility.
Integration with jBoss Seam	Allows to write applications with complex workflows and provides easier interactions between them.

1.3. Other relevant resources on the topic

All JBoss Developer Studio/JBoss Tools release documentation you can find at http://docs.jboss.org/tools in the corresponding release directory.

The latest documentation builds are available at *http://download.jboss.org/jbosstools/nightly-docs* [http://download.jboss.org/jbosstools/nightly-docs/].

JBoss jBPM Runtime Installation

The main purpose of this chapter is to let you know how to launch the JBoss jBPM (business process management).

The jBPM plugin (jBPM Designer) is already included in the JBoss Tools. To make it work, you should only download the jBPM runtime (jbpm-jpdl-3.2.3 currently) and specify the directory where you extracted the runtime either when you create a jBPM project or by using the jBPM preference pages.

Note:

Try to avoid using spaces in the names of installation folders. It can provoke problems in some situations with Sun-based VMs.

Navigate to *Window > Preferences > JBoss jBPM > Runtime Locations*. Here you can add, edit and remove JBoss jBPM installation locations. Click *Add* button. In the dialog that appeared enter a name for a newly added jBPM runtime and point to the correct location of this package on your harddrive. Click *OK* then click *OK* again.

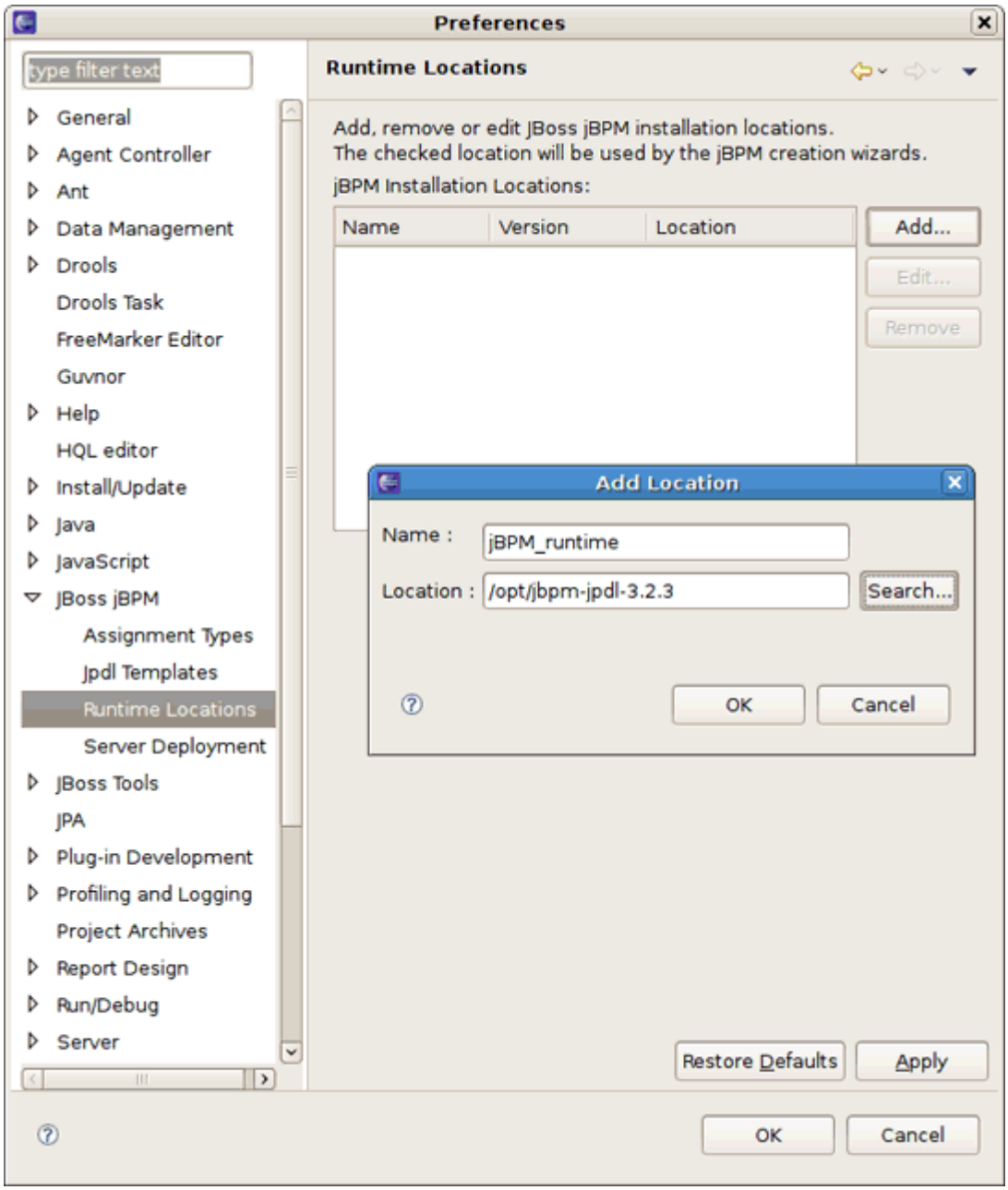

Figure 2.1. Adding jBPM Location

Now, when you have a runtime installed, we are going to demonstrate some powerful features of the jBPM.

The views

Here, it will be explained how to work with views and editors provided by JBDS.

The views are used for representation and navigation the resources you are working on at the moment. One of the advantages of all the views is that all modifications made in the current active file are immediately displayed in them. Let's get acquainted more closely with those that the jPDL perspective provides.

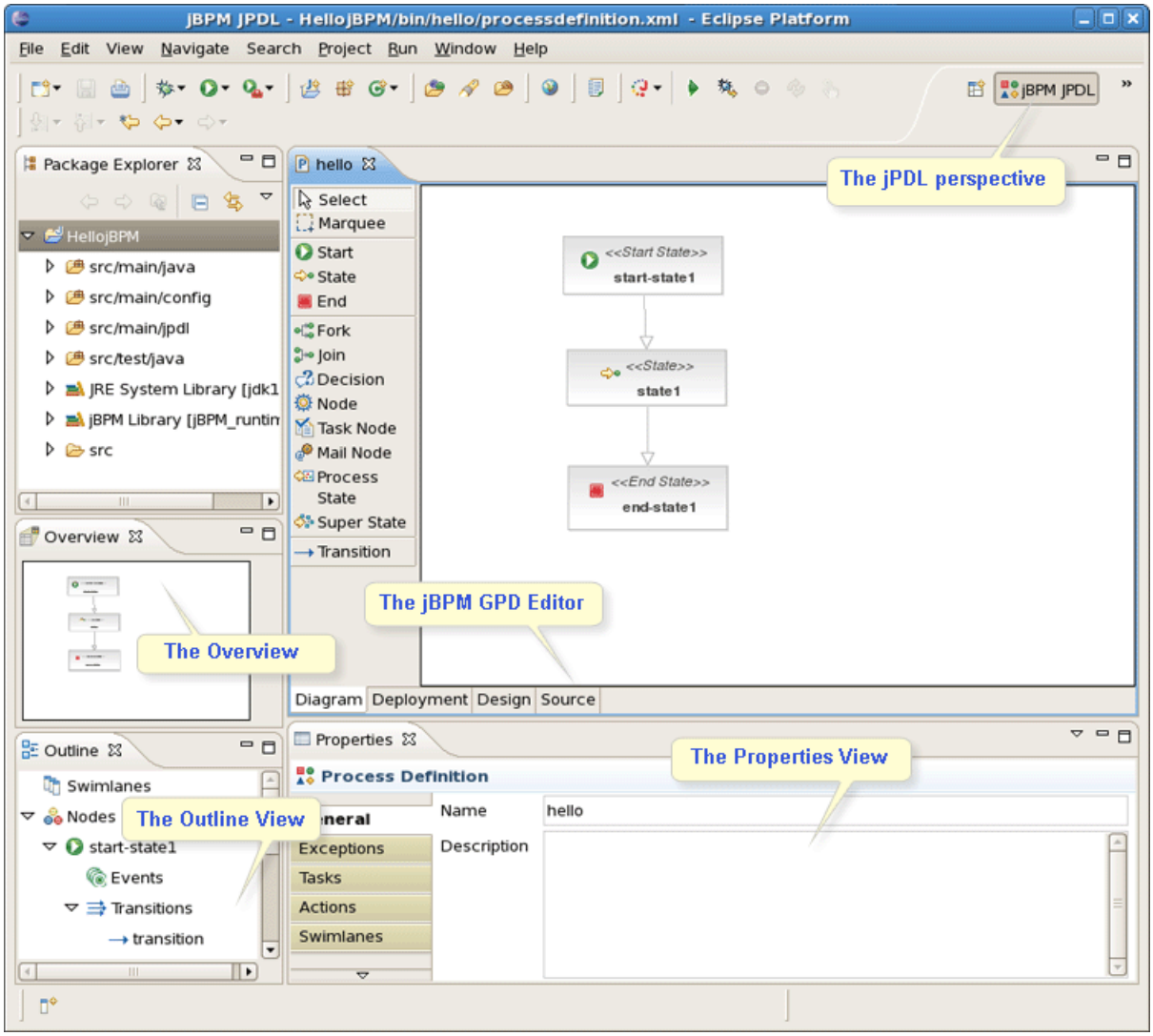

Figure 3.1. The jPDL Perspective Views and Editors

As you can see in the picture above, the jPDL perspective contains a complete set of functionality that's necessary for working on the jBPM project.

3.1. The Outline View

To have a way to quickly see an outline of the process use the *Outline view* that is presented as the classical tree. If it is not visible select *Window > Show view > Outline*.

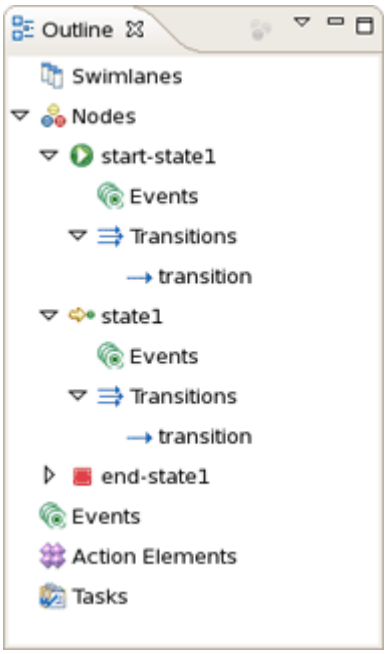

Figure 3.2. The Overview View

3.2. The Overview

The main advantage of this view is that it gives visual representation of the whole current developing process. Besides, the *Overview* comes as a scrollable thumbnail which enables a better navigation of the process structure if it's too large.

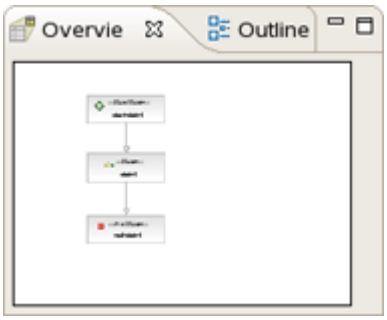

Figure 3.3. The Overview

3.3. The Properties View

Here, we dwell on the JBDS Properties view.

Notice if it's not visible you can access it by navigating *Window > Show view > Properties*.

The view shows the relevant properties of the selected item in the tabbed form. Every item has its own set of properties, which can be directly editable in the Properties view or by brining up the context menu.

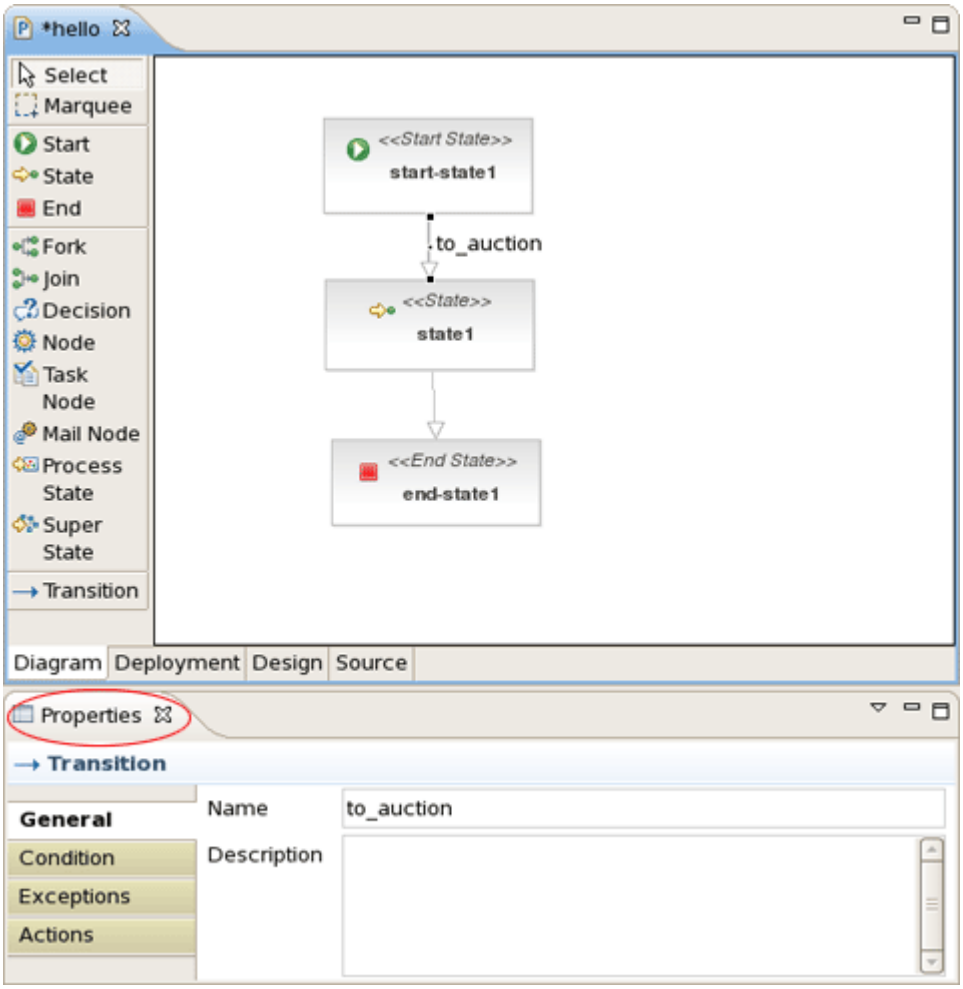

Figure 3.4. The Properties View of selected Transition

For example, on the picture above the Properties view displays all the properties for a selected transition. Its name has been changed to *to_auction*. We've done it directly in active General tab of the view. The same way let's change the name for the second transition to *to_end*.

If no one item is selected, the view represents the properties of the whole process definition.

Figure 3.5. The Properties View of Process Definition

In this case, it contains six tabs. The first one is the *General*. It allows to specify a process name and add necessary description. To illustrate let's change the process definition name to *jbay*.

3.4. The jBPM Graphical Process Designer editor.

The jBPM GPD editor includes four modes: Diagram, Deployment, Design and Source, which are available as switchable tabs at the bottom of the editor. Let's dwell on each of them.

3.4.1. The Diagram mode

In this mode we define the process in the form of a diagram by means of tools provided on the left-hand side of the jBPM GPD.

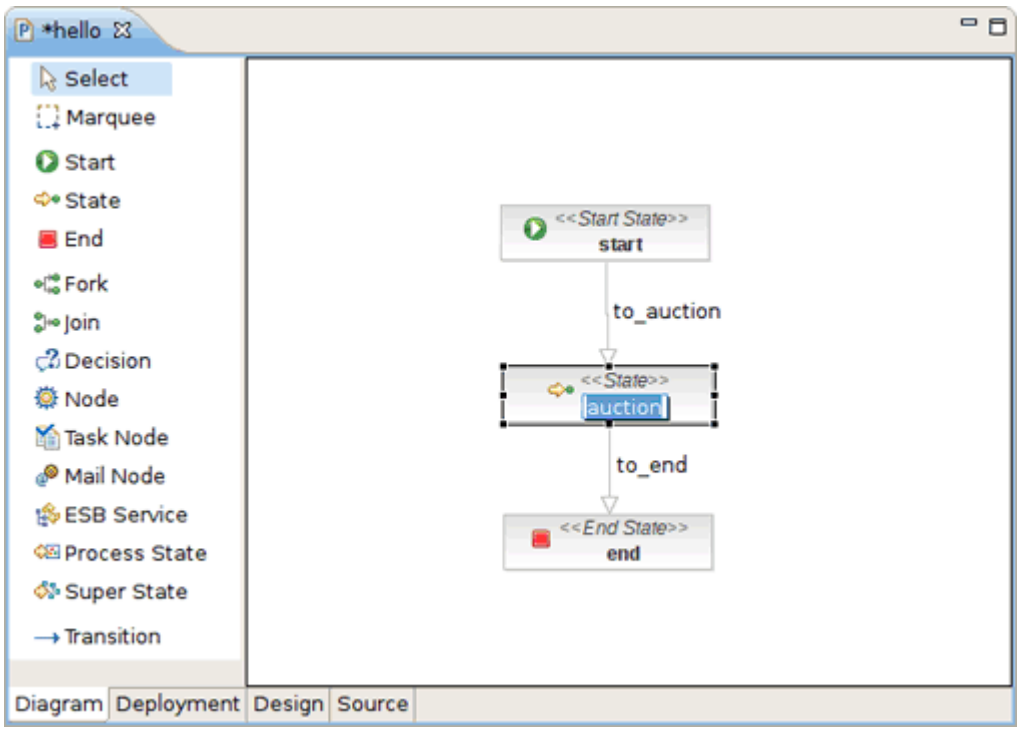

Figure 3.6. The Diagram mode

Besides, some properties can be directly edited in the Diagram mode of the graphical editor. One example of this is the *name* property of nodes. You can edit this directly by selecting the node of which you want to change the name and then click once inside this node. This enables an editor in the node. We change the name of the node to *auction*.

3.4.2. The Source Mode

Now, that we have defined a simple process definition, we can have a look at the XML that is being generated under the covers. To see this XML click on the Source tab of the graphical process designer editor.

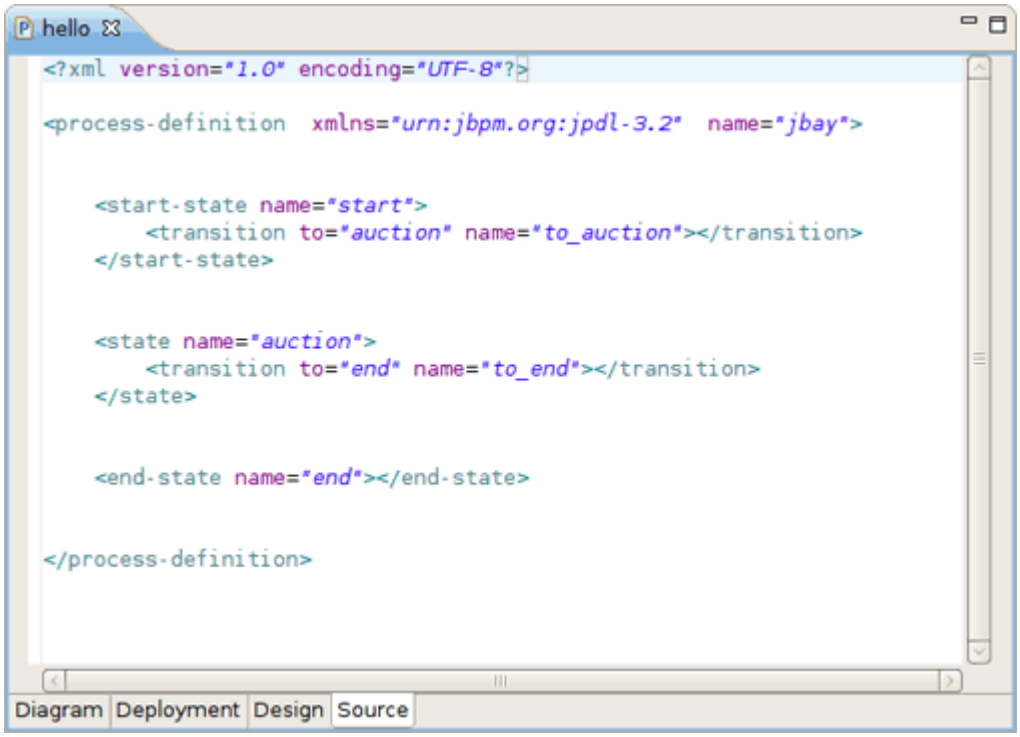

Figure 3.7. The Source Mode

The Source mode enables to easily manipulate our XML. That is manually inserting and editing necessary elements or attributes. In addition, here you can take advantage of content assist.

3.4.3. The Design Mode

One more way to edit your file is to use Design mode. You can see it in the next picture:

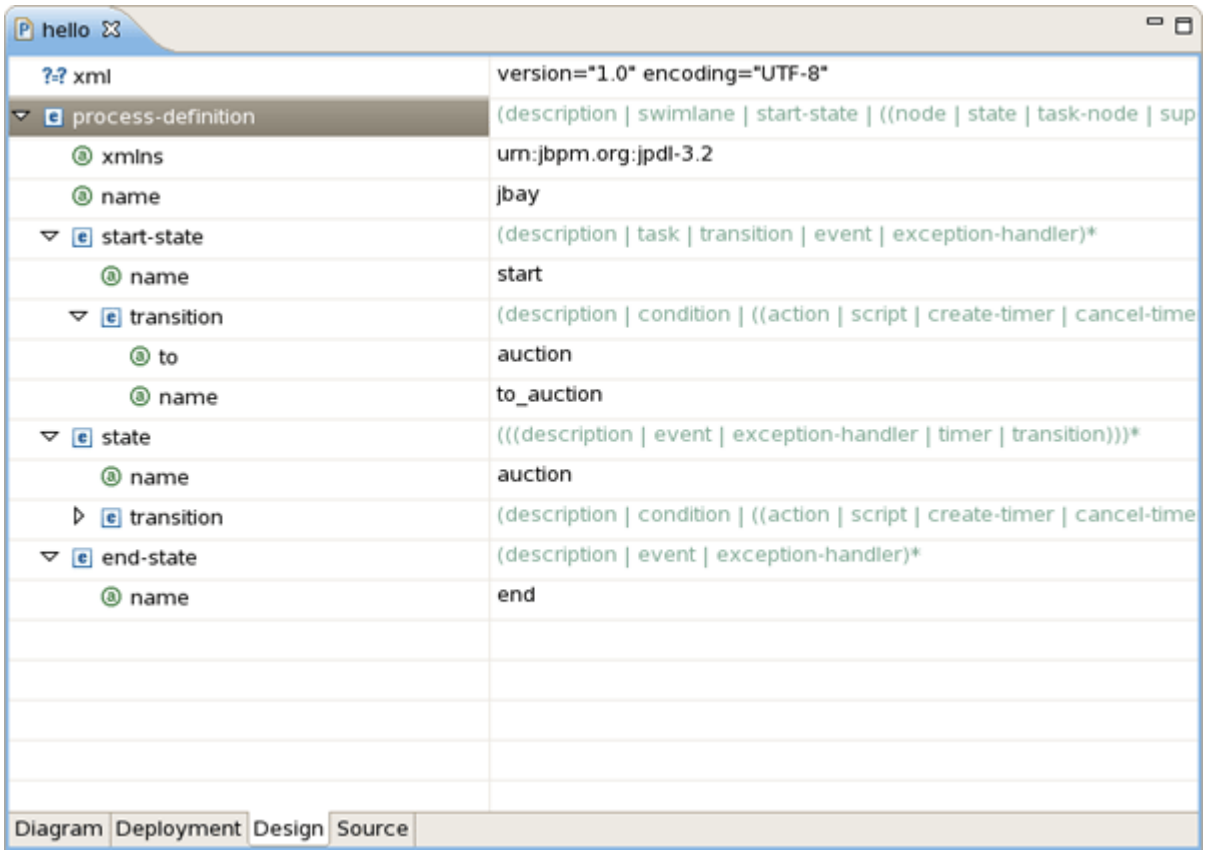

Figure 3.8. The Design Mode

As you can see above, this mode looks like a table in the first column of which the process structure is performed. Here, you can also insert, remove and edit elements or attributes, moreover add comments and instructions. Their values can be directly edited in the second column of the Design mode table.

For instance, let's add a comment on the second transition. For that, you should bring up the context menu for it and choose *Add Before > Comment*.

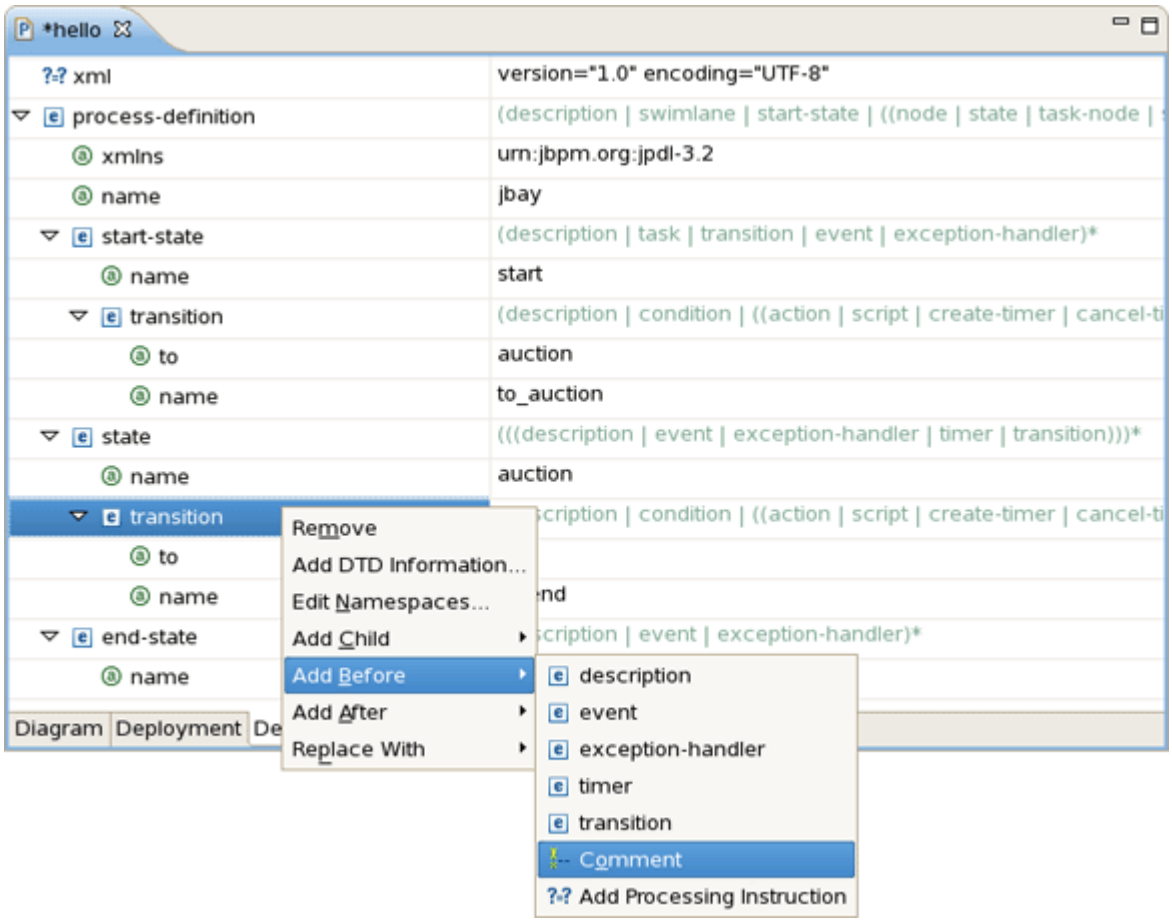

Figure 3.9. Adding a Comment

Then, we can put the text *This transition leads to the end state* in the right column as its value.

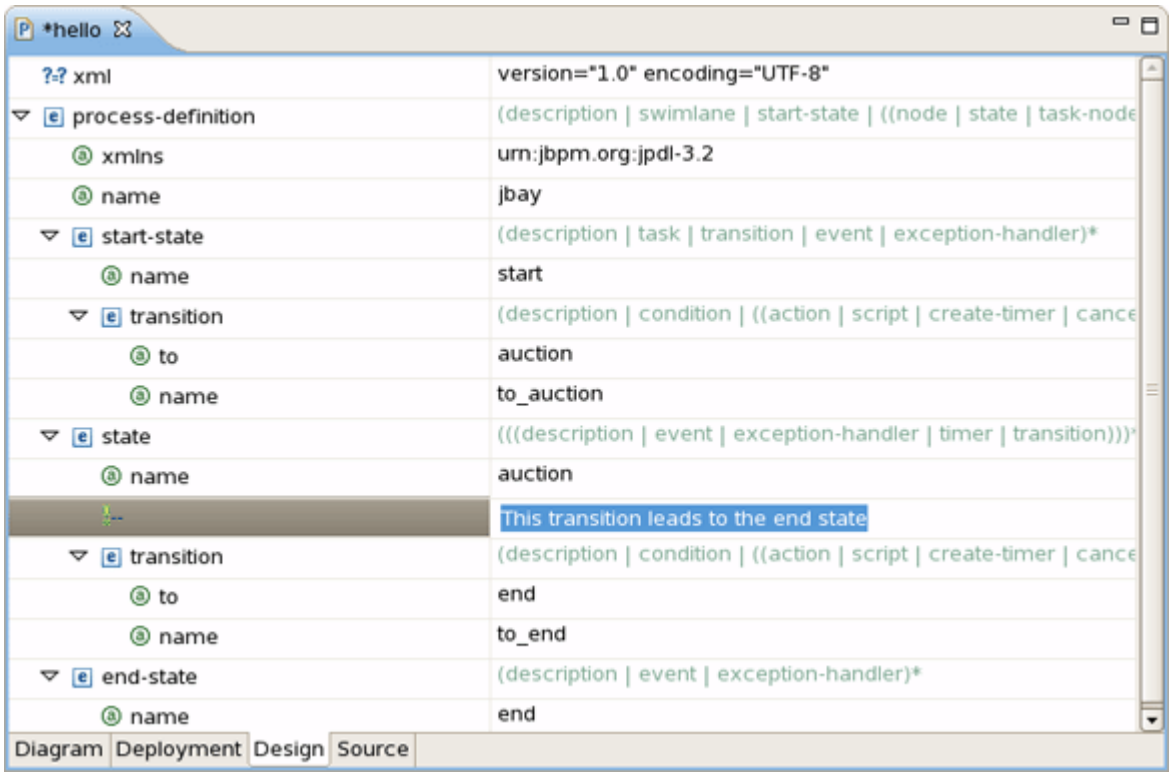

Figure 3.10. Comment is added

3.4.4. The Deployment Mode

Finally, to adjust the deployment settings of the project you should switch on to the tab that opens the Deployment mode. On the picture below the Deployment mode is performed with default settings. Here, you can easily modify them or, if the settings won't match your needs, to reset defaults.

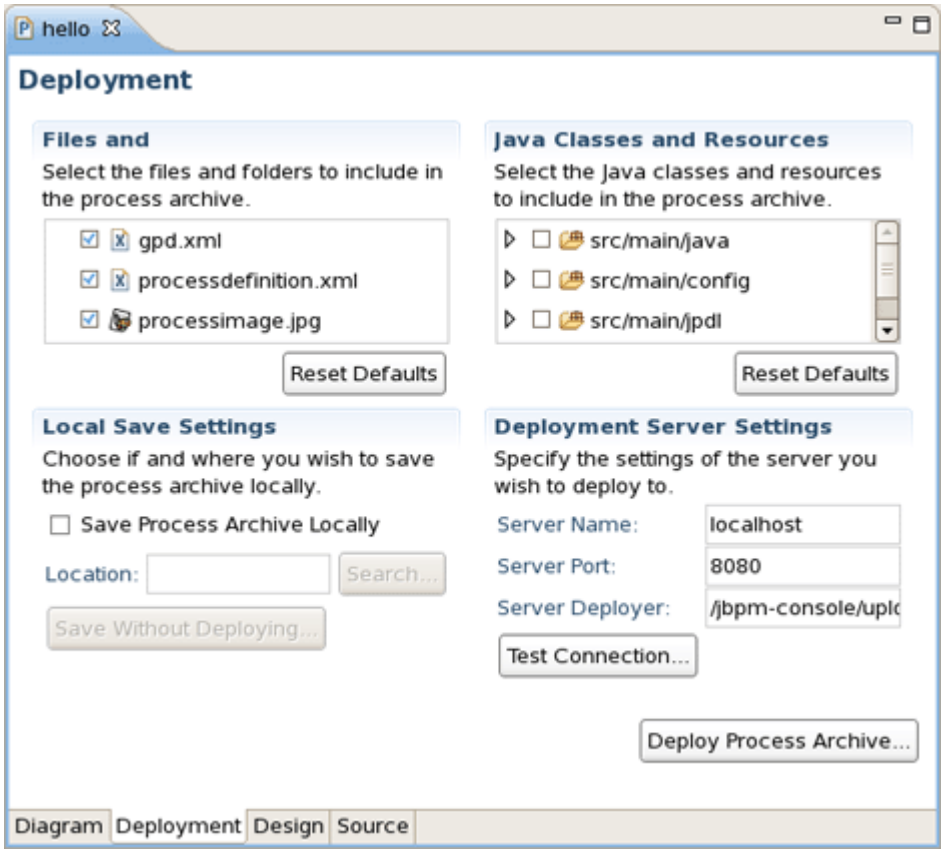

Figure 3.11. The Deployment Mode

The button *Test Connections* is necessary to make sure whether all your settings are valid before deploying the process.

Now that we've seen how to work with jPDL perspective, let's pass on to the project testing.

Test Driven Process Development

One of the most important advantages of JBoss jBPM's lightweight approach to BPM and workflow management is that developers can easily leverage their usual programming skills and techniques. One of these well-known techniques is Unit Testing and Test Driven Development.

In this chapter we will show how developers, making use of the JBoss jBPM GPD, can use a technique we have baptized Test Driven Process Development to create process definitions and test their correctness.

When creating the *HellojBPM* project the Project Creation wizard has already put in place all the library requirements we need to start writing the jBPM unit tests. They are contained in the jBPM Library container and the most important of them is the *.jar* file containing the core jBPM classes. While working on the project you could find them all in the Package Explorer.

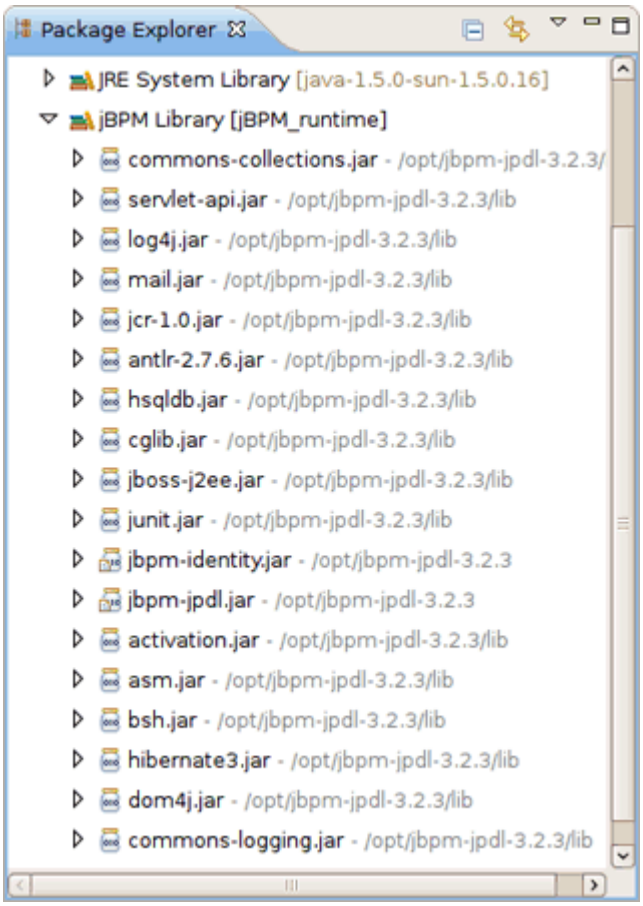

Figure 4.1. The jBPM Libraries

It must be noted that it is possible to change the location of the core jBPM installation by changing the preference settings. More on this *see later* in this book.

With that extra knowledge on the project settings, you can create your first test. To do this, we create the *com.jbay* package in the *test/java* source folder. Then we bring up the context menu on this package and select *New > Other...*

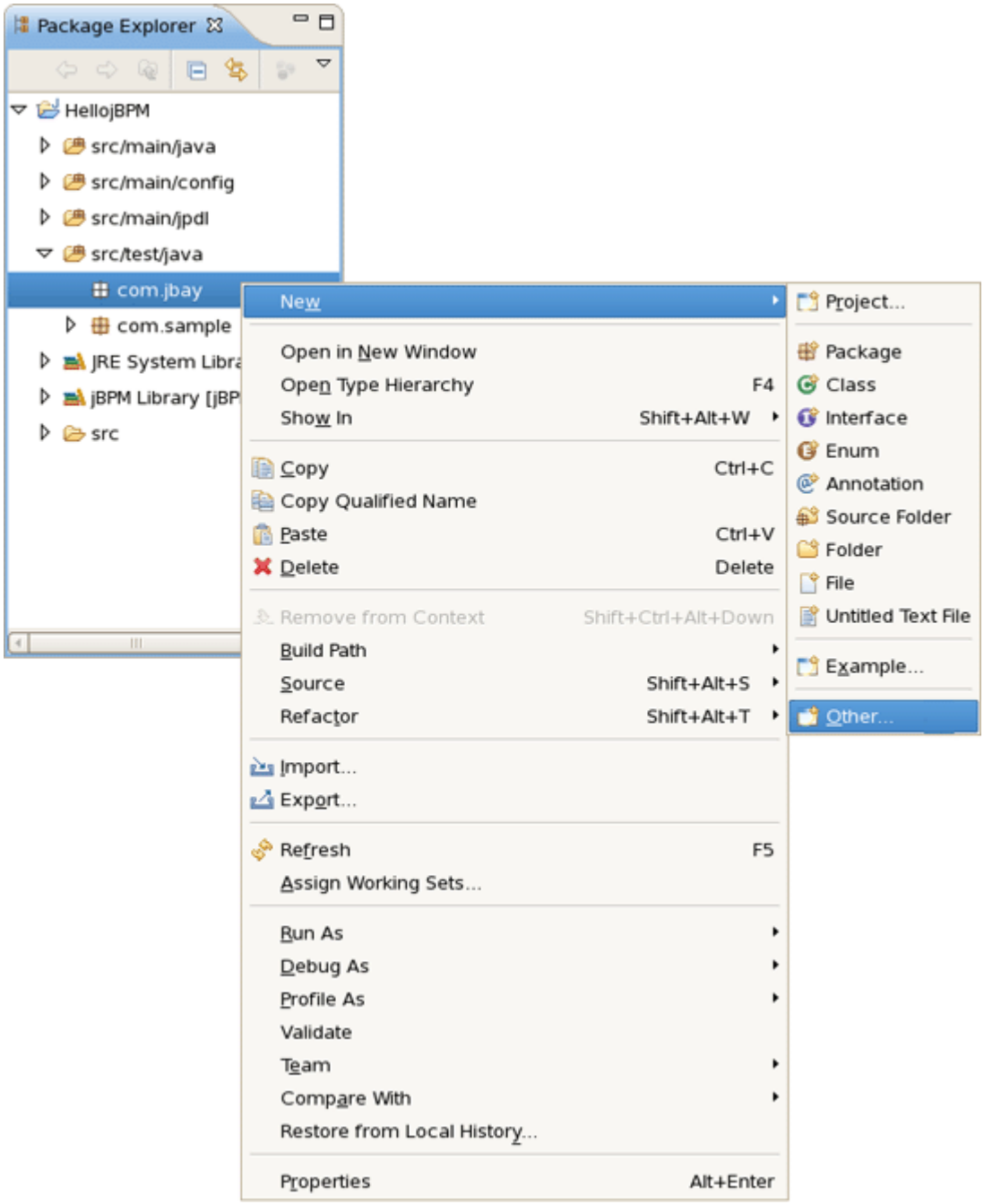

Figure 4.2. Call the JUnit Test Case Creation wizard

And then *Java > JUnit > JUnit Test Case* to call the specialized JUnite Test case creation wizard.

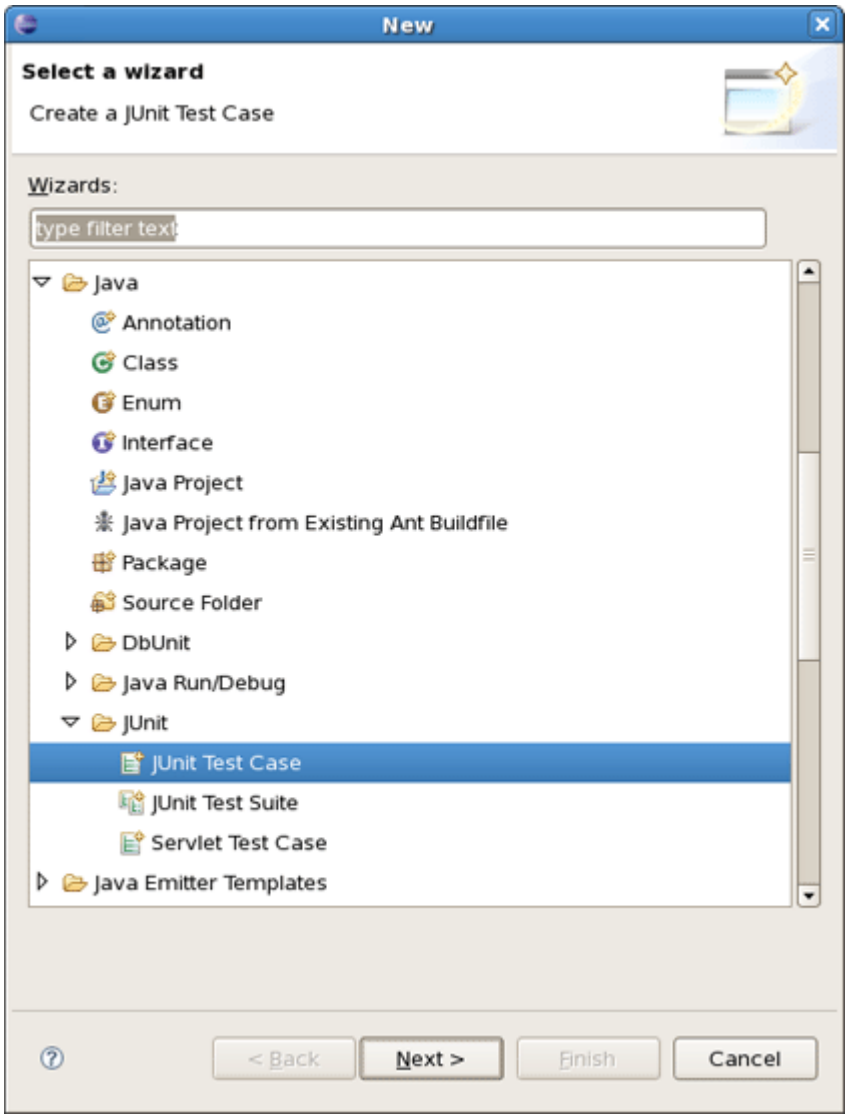

Figure 4.3. Call the JUnit Test Case Creation wizard

The wizard looks as follows:

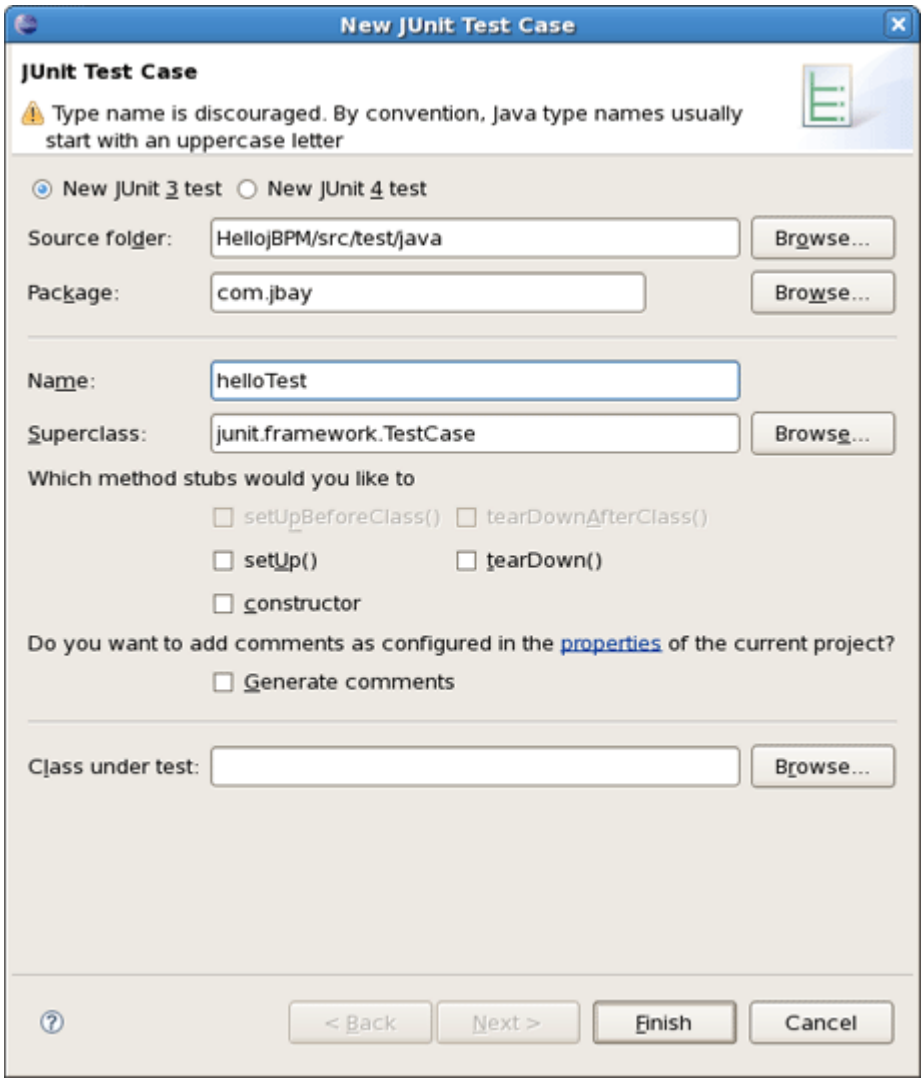

Figure 4.4. Create Test Dialog

By default JUnite 3 version of testing framework is selected. Of course, you can choose new advanced JUnit 4 version. In this case you'll be prompted to add new JUnit Library to your build path. To add it automatically just click on the appropriate link. In the *Class under test* section you can specify the class to test.

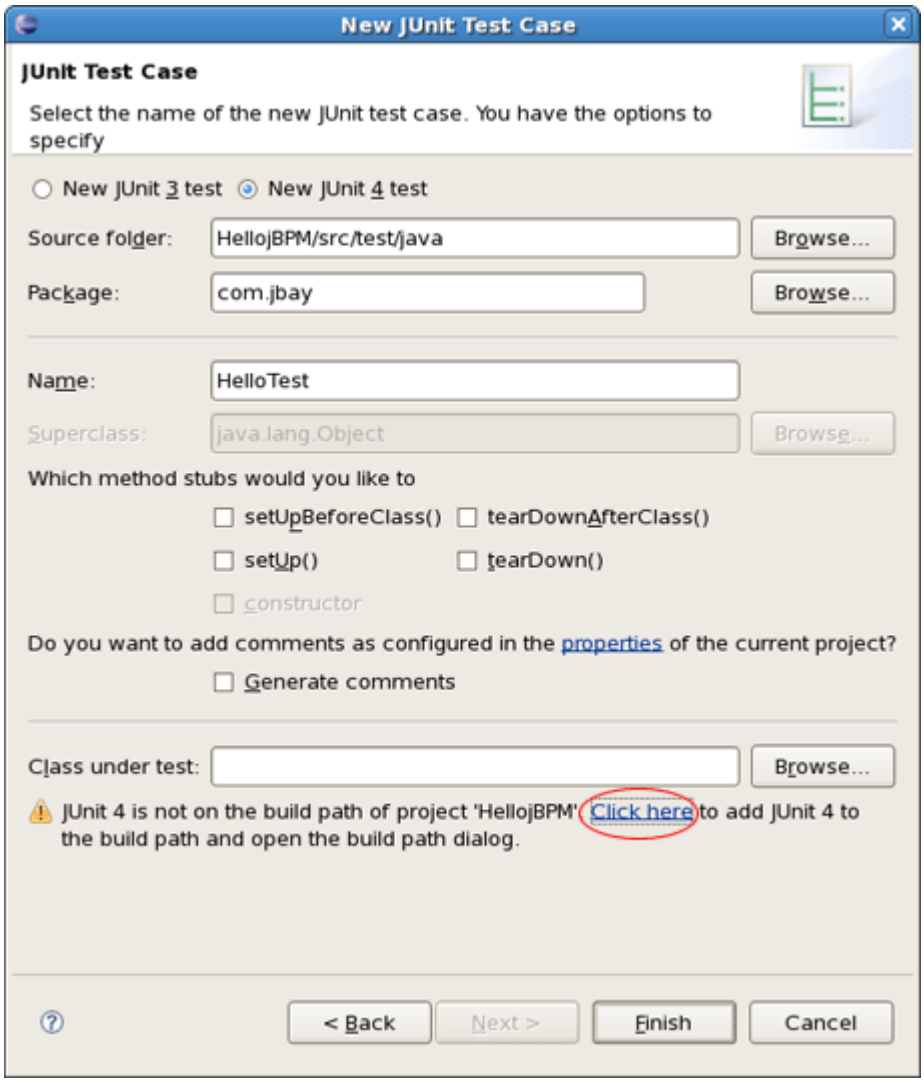

Figure 4.5. A First Test Scenario

Then, we call the test class *HelloTest* and press *Finish* button to complete.

Next, we should write a simple test scenario as shown on the next figure. Let's study the code of this test case.

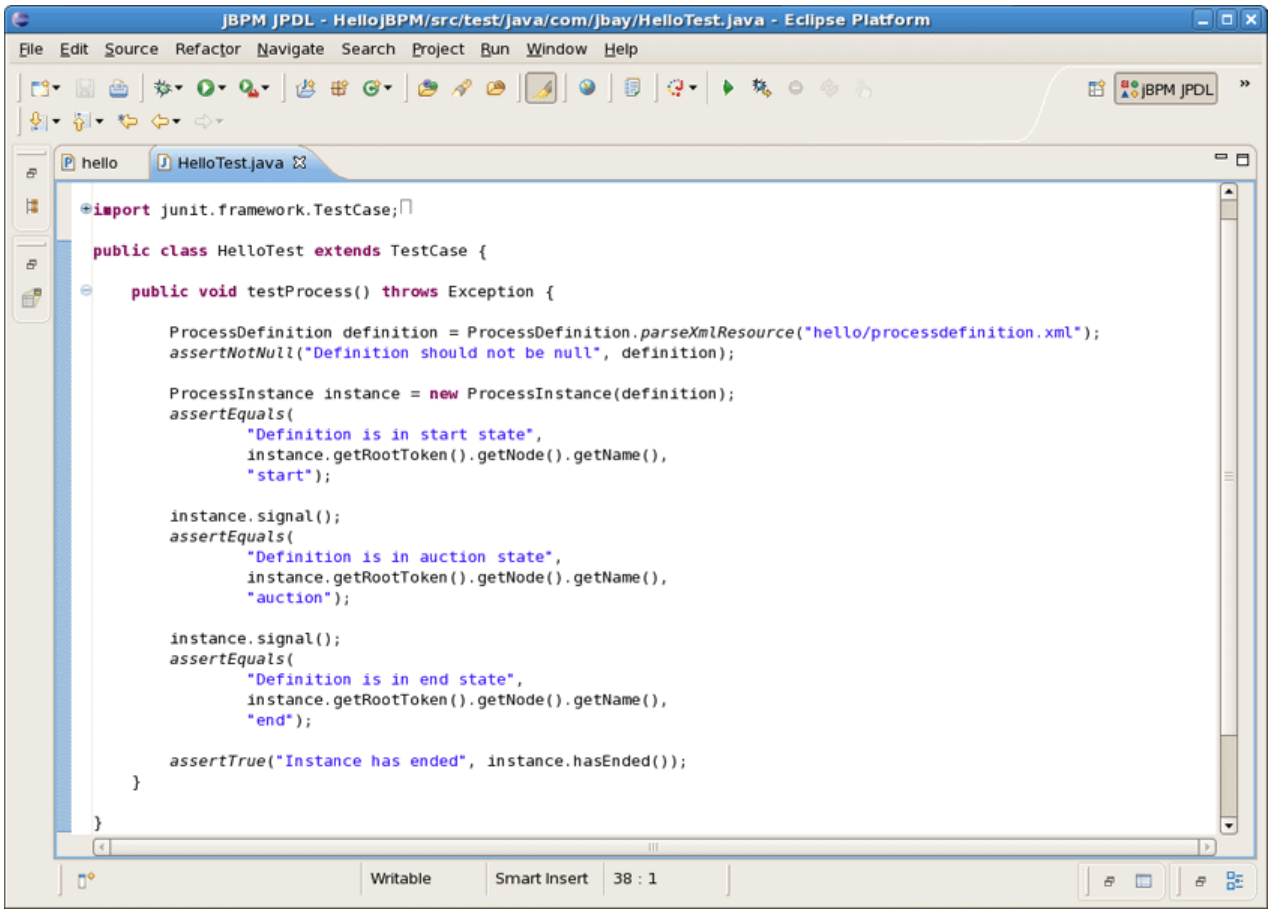

Figure 4.6. A First Test Scenario

In the first line of the method, a jBPM process archive object is created. We use a constructor accepting the filename of the archive. In our case it is the *hello* file we created earlier and which lives in the *src/main/jpdl* folder of our project. After asserting that this object is really created, we extract a process definition object from it. This object is fed to the constructor of a process instance object. We have a process instance object, but this process is not yet started, so we can safely assert that its root token still resides in the start node. After signalling the token will move to the next state and the process will be in the *auction* state. Finally another signal will end the process.

After writing this test we can check whether it works as expected by running it .

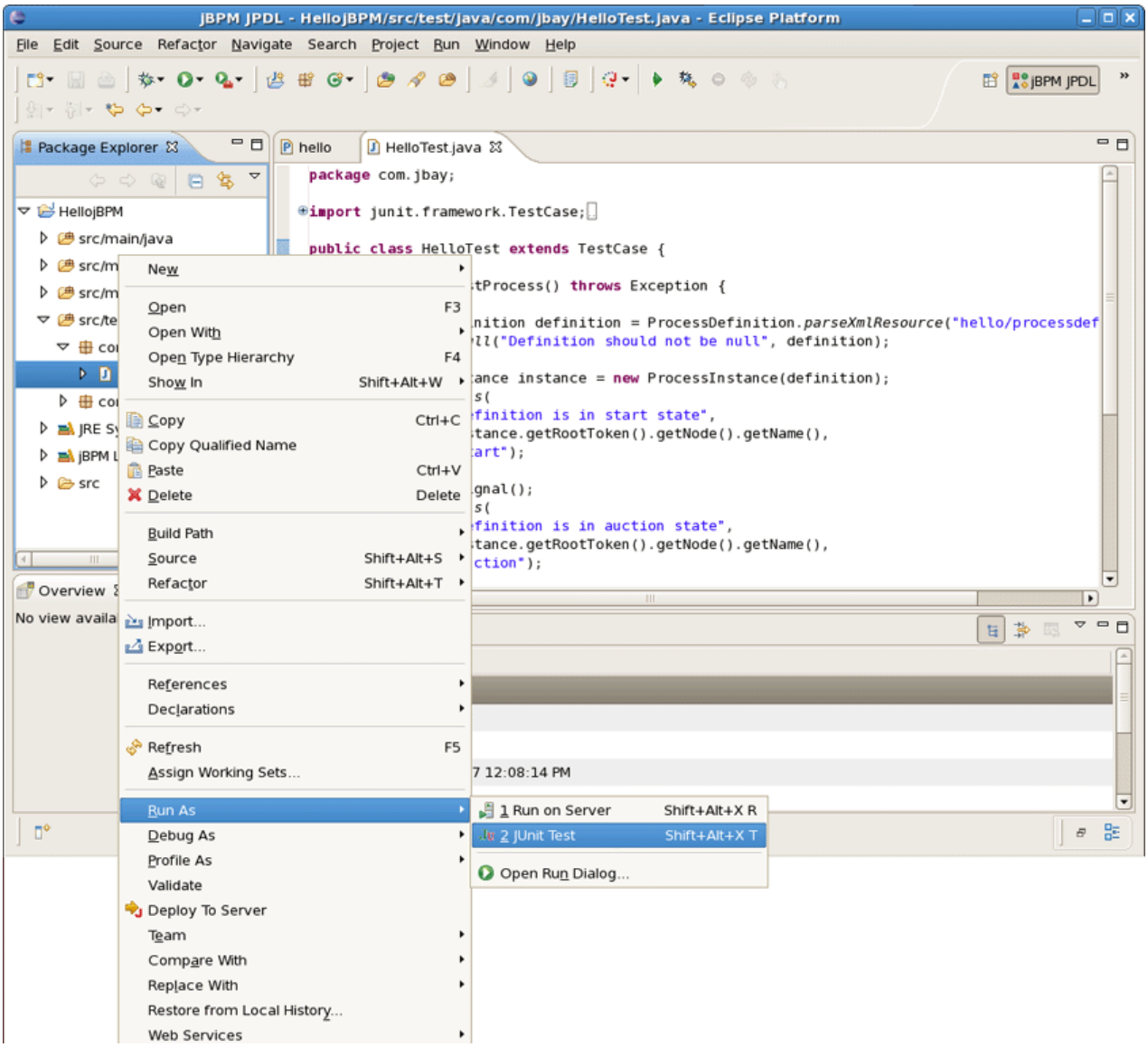

Figure 4.7. Running the Process Test

All went well as we have a green light:

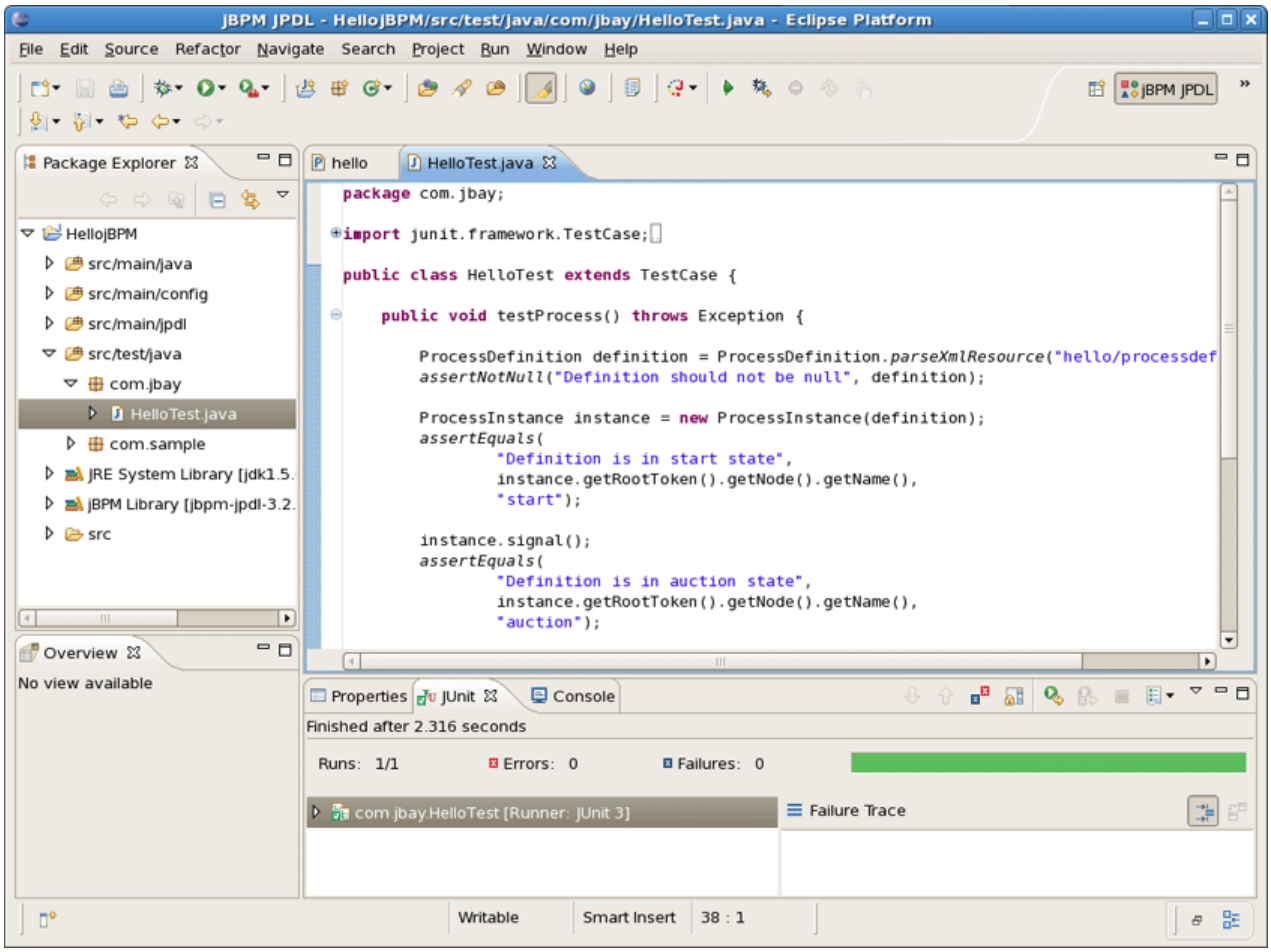

Figure 4.8. Successful Test Run

Of course, this simple scenario was not very interesting, but the purpose of it was to show how you can reuse your development skills in a very straightforward way when doing process development. To see how more interesting processes and process test scenario's can be developed, we suggest you to read the *JBoss jBPM User Guide* [http://docs.jboss.com/jbpm/v3/userguide/] and to study the API reference. You can find it in the jBPM download folder. (To get started we downloaded jbpm-jpdl-3.2.2 in *the second chapter*. You should just remember where you extracted it.) All we've mentioned are in the 'javadoc- *' subfolders of the 'doc' folder. Moreover, some more examples will be given later in this book.

Actions : The JBoss jBPM Integration Mechanism

In this chapter we will show how to do software integration with JBoss jBPM. The standard mechanism to implement this is to wrap the functionality you want to integrate in a class that implements the *ActionHandler* interface. In order to demonstrate it let's specify Hello World action for our process.

5.1. Creating a Hello World Action

Each Hello World process should integrate one or more Hello World actions, so this is what we will be doing. We can integrate custom code at different points in the process definition. To do this we have to specify an action handler, represented by an implementation of the *ActionHandler* interface, and attach this piece of code to a particular event. These events are amongst others, going over a transition, leaving or entering nodes, after and before signalling.

To make things a little bit more concrete, let's create a new class called *HelloActionHandler*. For that firstly we'll create a new package *com.jbay.action* in the *src/java/main* folder of our project. Then, we should call New Class Creation wizard as usual by right-clicking and navigating *New > Class*.

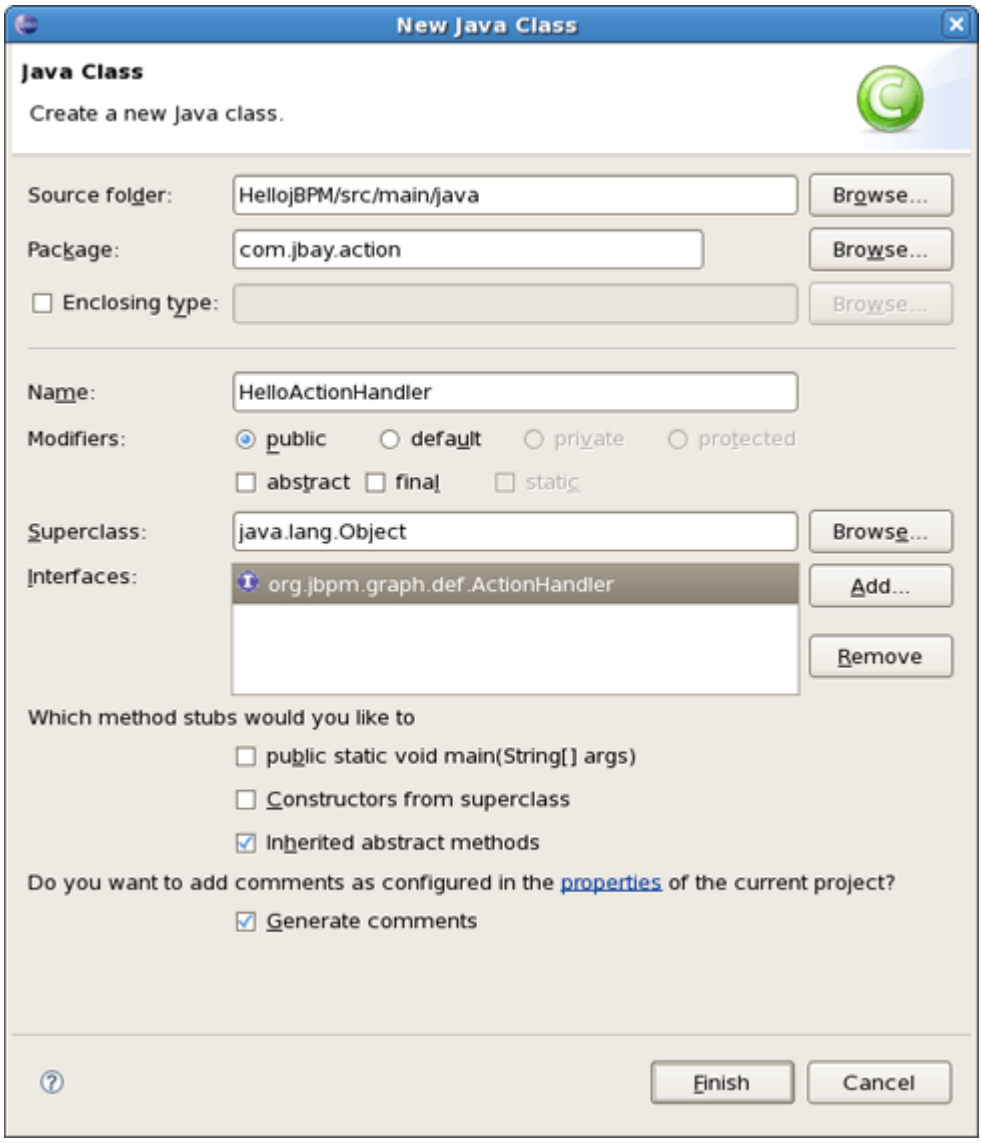

Figure 5.1. Creating HelloActionHendler Class

Notice that two first gaps have been filled automatically. Here, instead of *Package* option *Enclose type* option can be selected where a type in which to enclose a new class should be specified.

In our case, we leave everything as it is, just type *HelloActionHandler* as a name of new class and add *org.jbpm.graph.ActionHendler* interface as it's shown in the picture above.

Thus, our *HelloActionHandler* implements the *ActionHandler* interface including the *execute* method as shown in the next figure. Here, we add a variable named *greeting* to the collection of process variables and put a message in it : *"Hello from ActionHandler"*.

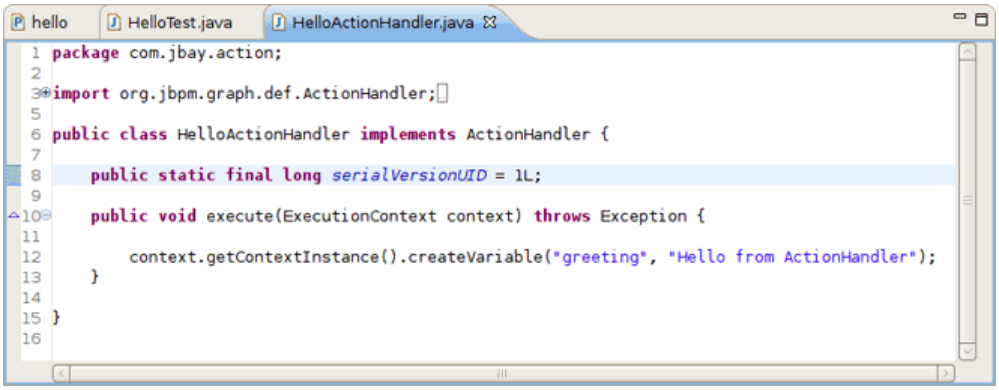

Figure 5.2. A Simple Hello Action

Now, as we have *HelloActionHandler* class defined, let's explore how we can handle it.

5.2. Integrating the Hello World Action

The main purpose of this chapter is to provide you with the steps associating our Hello World action with a particular event and test the correctness of our actions as well.

As good Testcity citizens we will first create a Unit Test that proves the behaviour we want to achieve by adding the *ActionHandler* to the process. So we implement another test.

At first, let's return to the code we already saw *in the previous chapter* and add new test method *testActionHendler* to it.

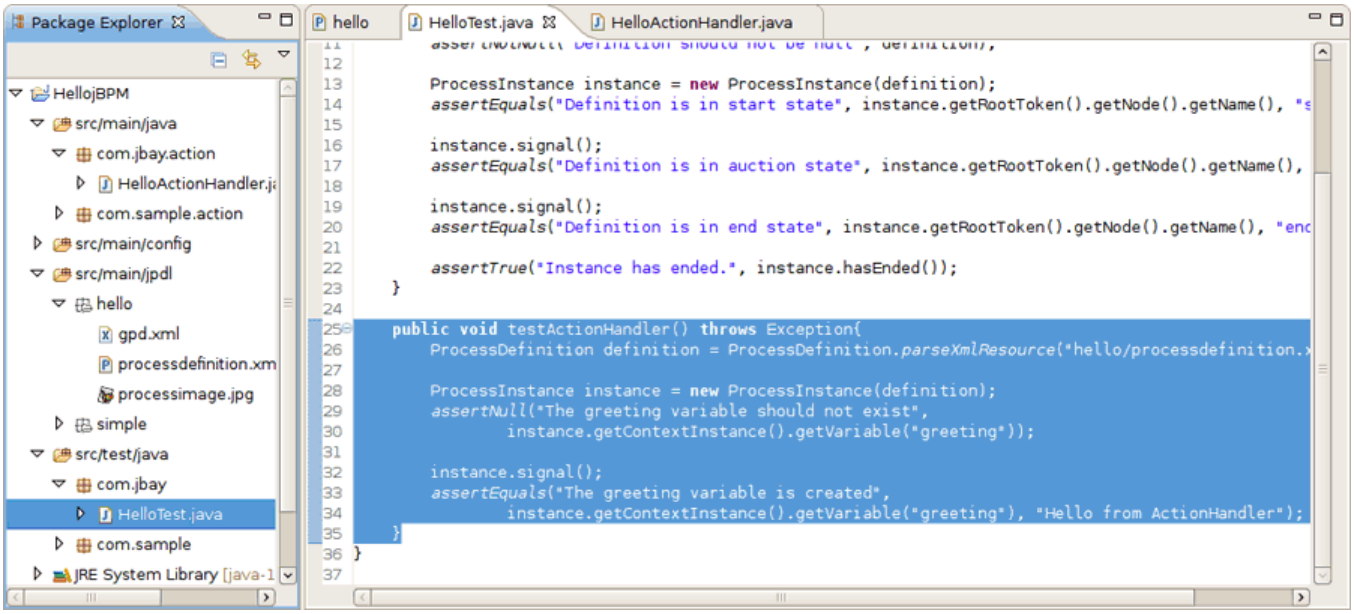

Figure 5.3. Create the Hello Action Test

We assert that no variable called *greeting* exist. Then we give the process a signal to move it to the auction state. We want to associate the execution of the action with the event of going over the transition from the start state to the auction state. So after the signal, the process should

be in the auction state as in the previous scenario. But moreover, the *greeting* variable should exist and contain the string *"Hello from ActionHandler"*. That's what we assert in the last lines of the test method.

Running the tests now results in a failure. The point is that we did not associate the action with any particular event in the process definition, so the process variable did not get set.

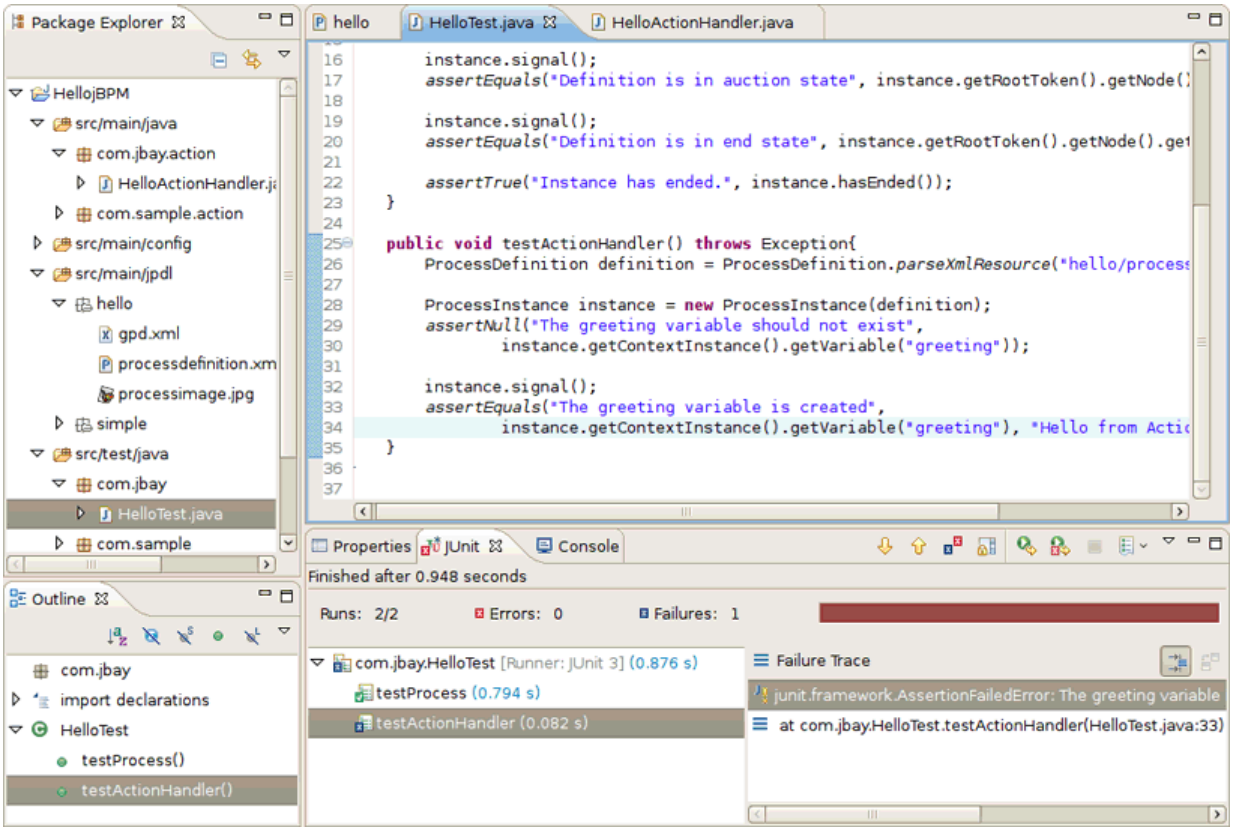

Figure 5.4. Test Results Before Integration

Let's do something about it and add an action to the first transition of our sample process. To do this you can use the Actions tab in the Properties Editor that is under the graphical canvas. Bring up the popup menu of the action element container and chose New Action as it's shown on the figure below. The other way to add an action to the transition is simply to use the dropdown menu that is available under the action icon in the right upper corner of the Properties View.

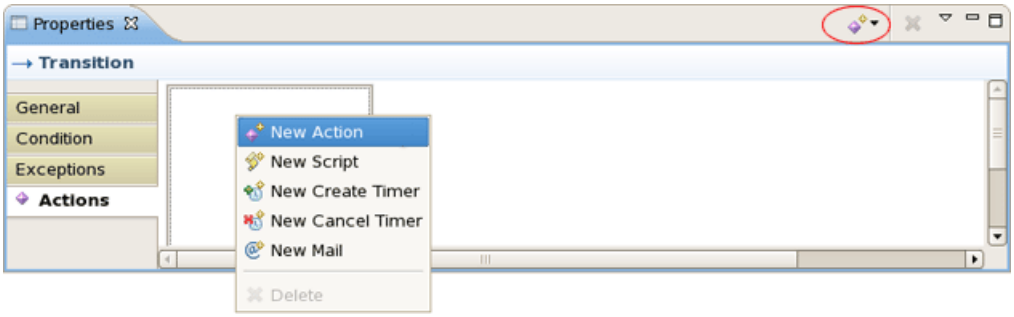

Figure 5.5. Adding an Action to a Transition

After adding the action a tabbed view with three pages will appear.

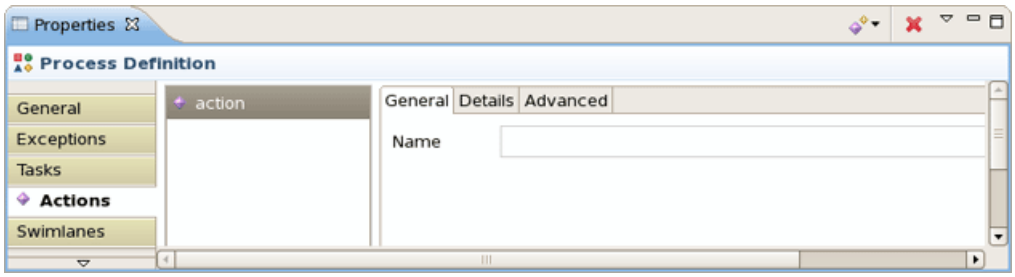

Figure 5.6. Configuration Dialog for an Action

The first of these three pages allows you to give the Action a name. The last page contains some advanced attributes such as whether the Action is asynchronous. The Details page is the most important. It allows to choose and configure the actual action handler implementation.

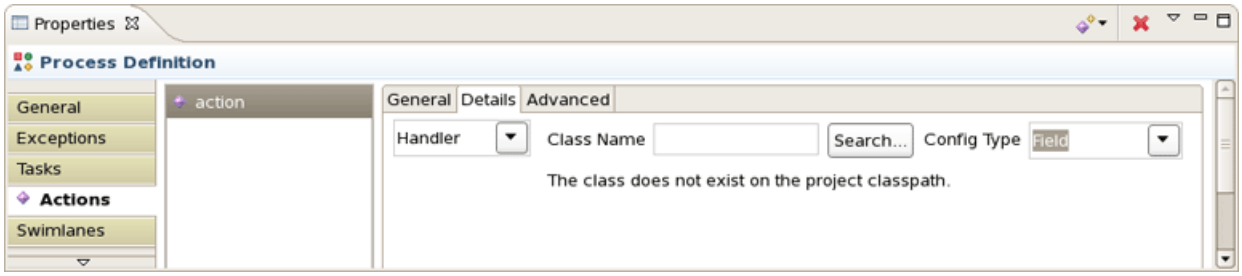

Figure 5.7. The Details page of an Action Configuration Dialog

Clicking on the *Search...* button brings us to a Choose Class dialog.

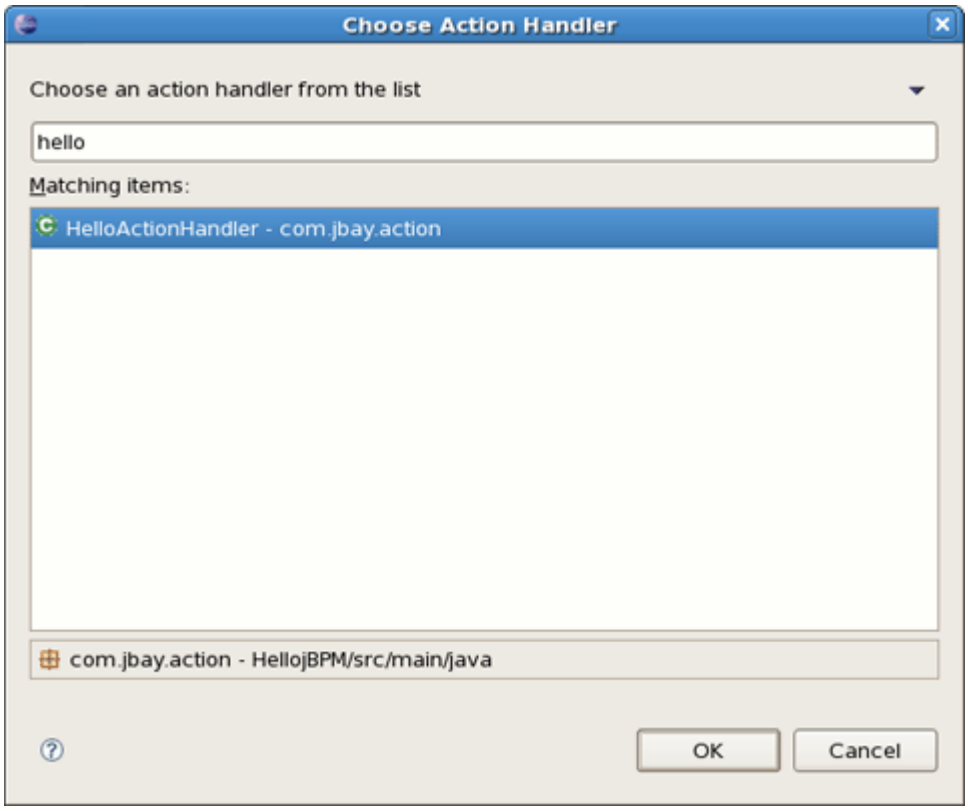

Figure 5.8. The Choose Action Handler Dialog

We choose our previously created 'HelloActionHandler' class and push the OK button. After the selection of the action handler for the action, we can run the test and observe it gives us a green light.

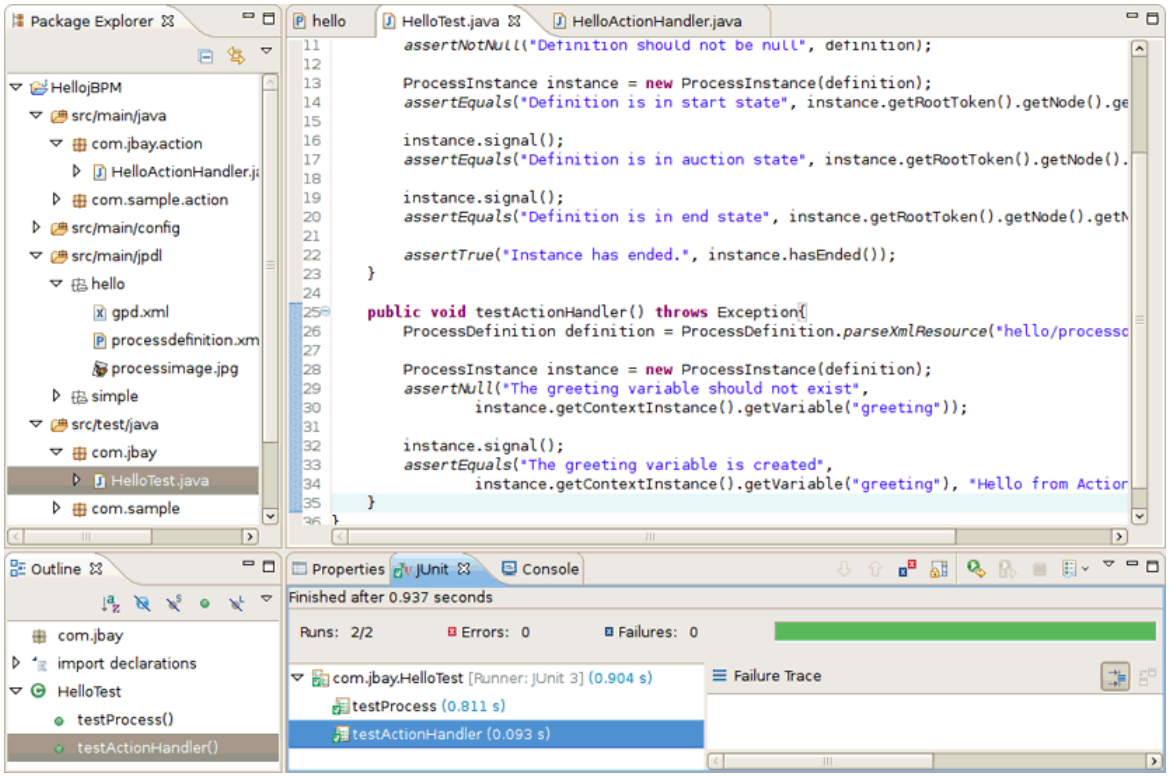

Figure 5.9. Test Results

There we are. The above objective has been achieved.

5.3. Integration Points

The different integration points in a process definition are thoroughly documented in the *JBoss jBPM User Guide* [http://docs.jboss.com/jbpm/v3/userguide/]. Instance nodes can contain many action elements. Each of these will appear in the Action element list of the Actions tab. But each Action also has a properties view of itself. You can navigate to this view by selecting the added Action in the outline view.

Quick Howto Guide

This chapter contains additional information related to the JBoss jBPM.

6.1. Change the Default Core jBPM Installation

You can change the default jBPM installation by means of the Eclipse preference mechanism. Open the Preferences dialog by selecting *Window > Preferences* and select the *JBoss jBPM > Runtime Location* category. Using this page you can add multiple jBPM installation locations and change the default one. The default installation is used for the classpath settings when creating a new Process Project. Changing the preferences has no influence on already created projects. Getting rid of a jBPM installation that's being referenced by a project however will cause the classpath to contain errors.

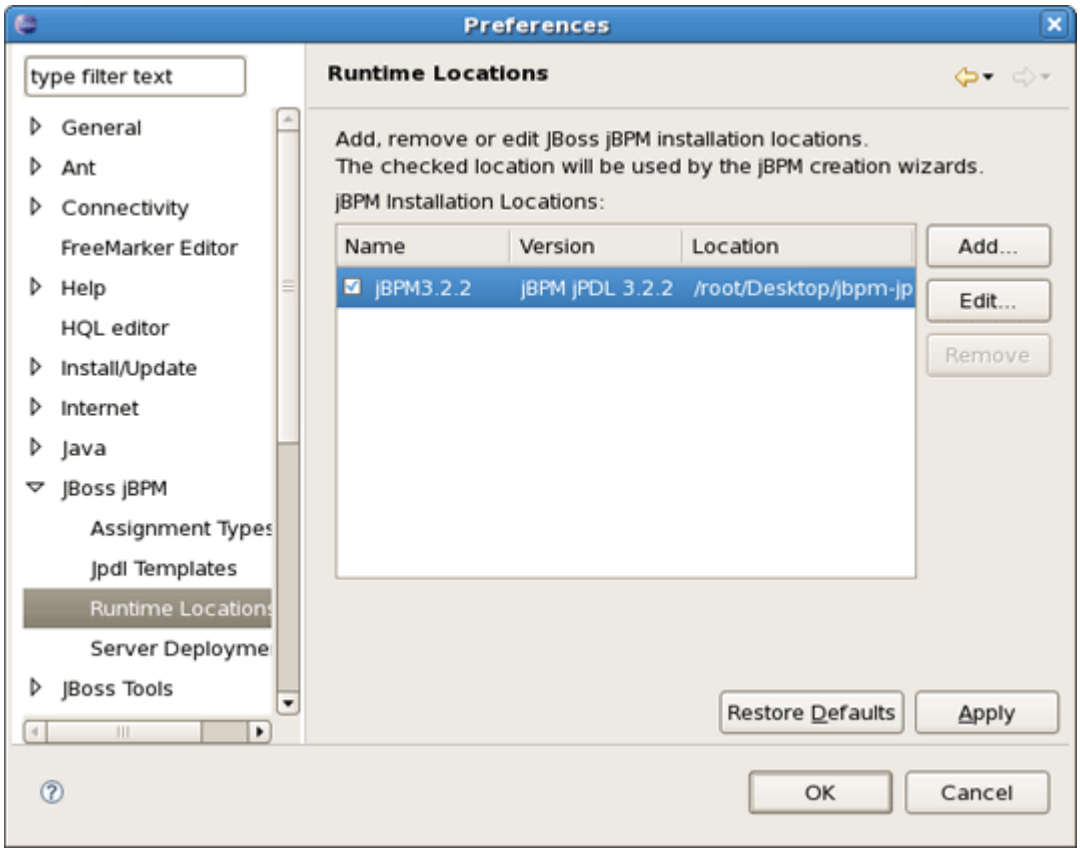

Figure 6.1. The jBPM Preferences Page

6.2. Configuring Task Nodes

Here, we'll examine how you can configure the Task nodes in jBPM jPDL GPD.

You can add Tasks to Task nodes and then configure them in a similar manner as the Action configuration mechanism. Let's consider the process definition similar to the previous one that

contains three nodes: Start state, Task node and End state. The Properties view for selected Task node includes several tabs.

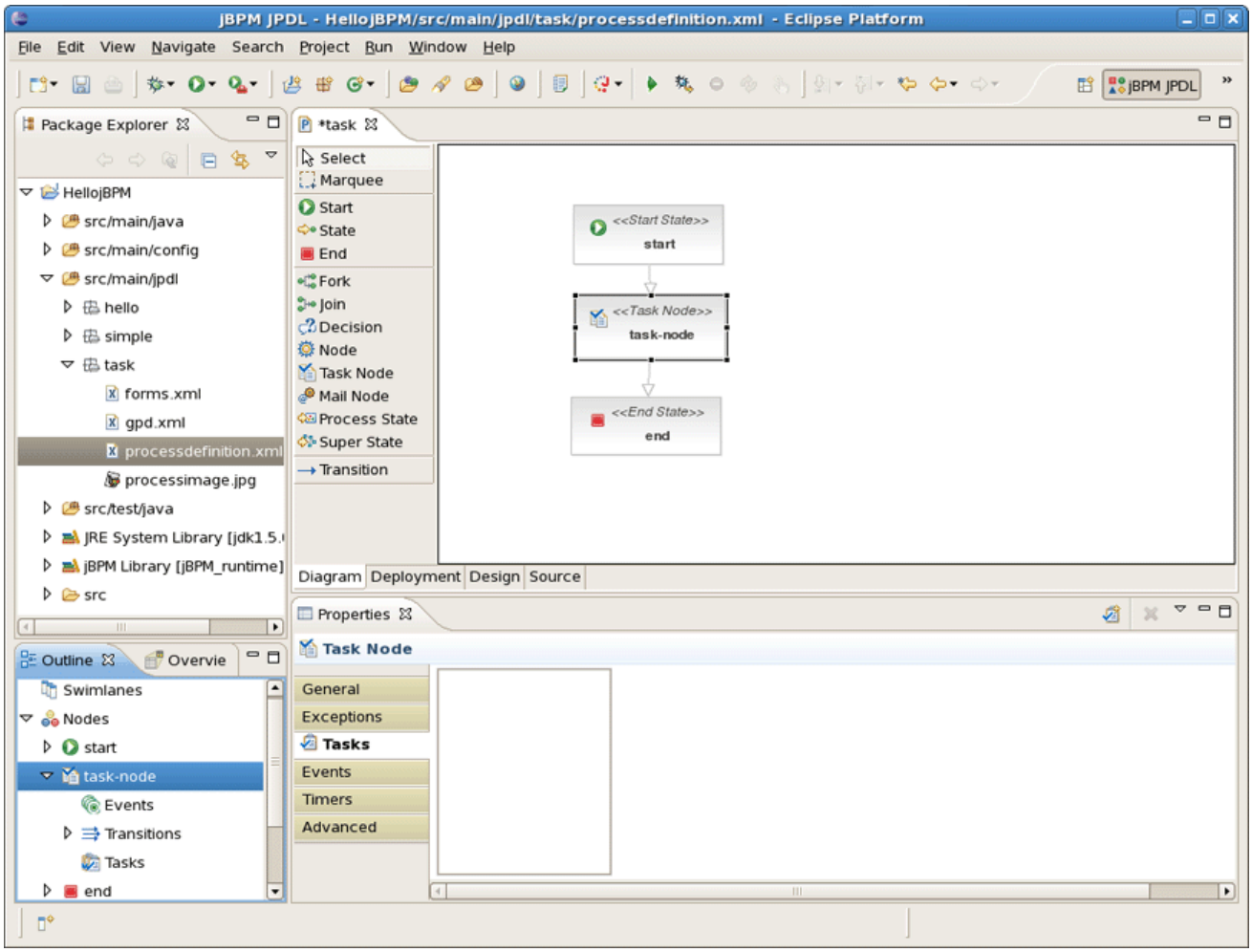

Figure 6.2. The Properties View of the selected Task Node

We should choose the Task tab and then bring up the context menu or click the button in the top right corner of the view to add a Task to our Task node.

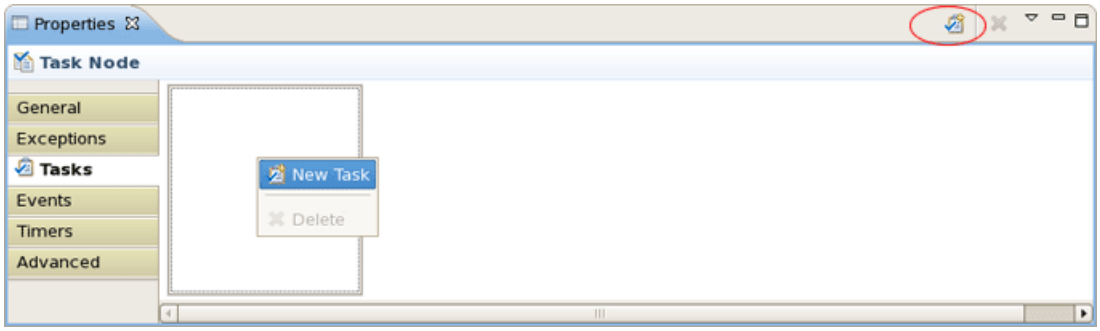

Figure 6.3. Adding a Task to the Task Node

Every added Task has its own configuration possibilities. You can access them through the Properties view as well.

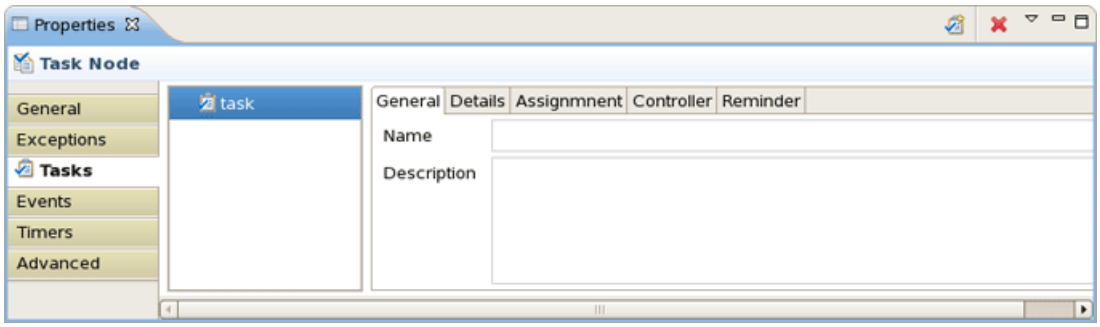

Figure 6.4. The Task properties

The General page is a place where you can specify the name of a Task and its description. For instance, let it be *approve oder* with appropriate description that you can see in the figure below.

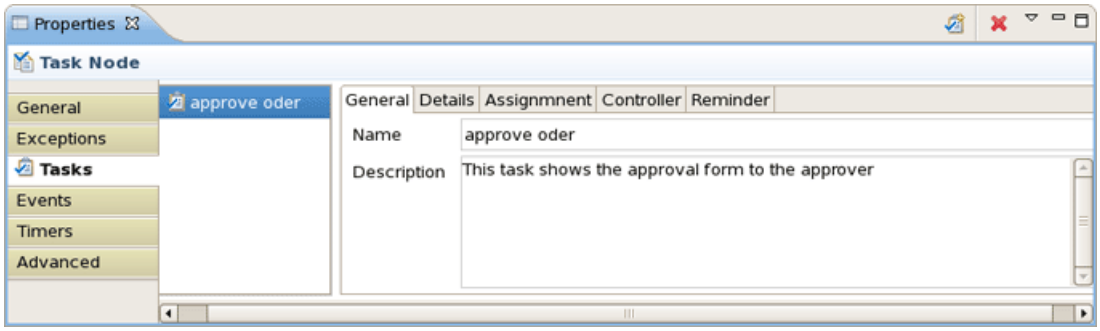

Figure 6.5. The Task General Page

Now, look at Details page. First, you should specify the due date that is a mandatory property for the Task. The due date is the date on which the task should be accomplished. Here you can also set a Task priority as well as signalling, notifying or blocking. The *Blocking* attribute indicates that the process will not be able to continue if this task is still unaccomplished. The *Generate Form...* button is for creating a simple task form that can be rendered by the jBPM console.

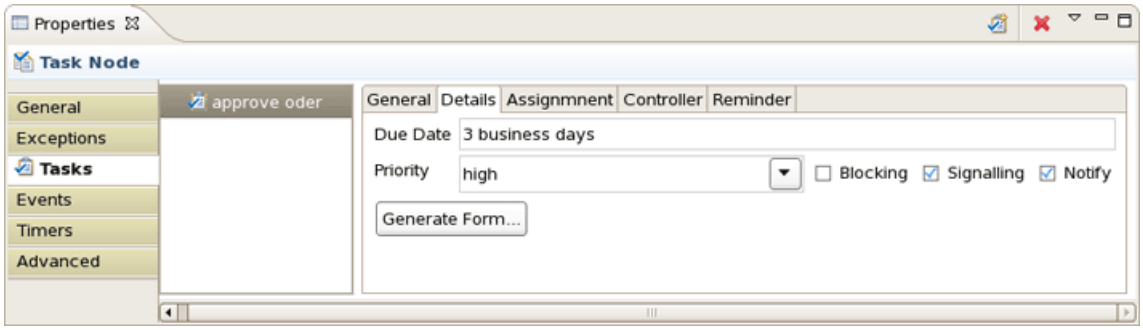

Figure 6.6. The Task Details Page

For our example, we specify the due date as 2 business days, choose the high priority and also check the *Signalling* and *Notify* attributes. It means that the Task should be accomplished in 2 business days and the assignee will be notified by email when the task is assigned. To specify how the Task should be assigned switch on to the Assignment page.

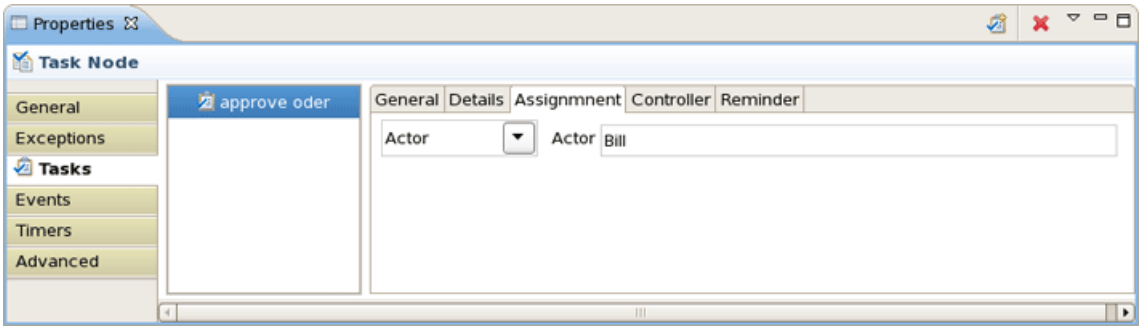

Figure 6.7. The Task Assignment Page

On the Reminder page you can specify whether the assignee will be reminded of the task that awaits him.

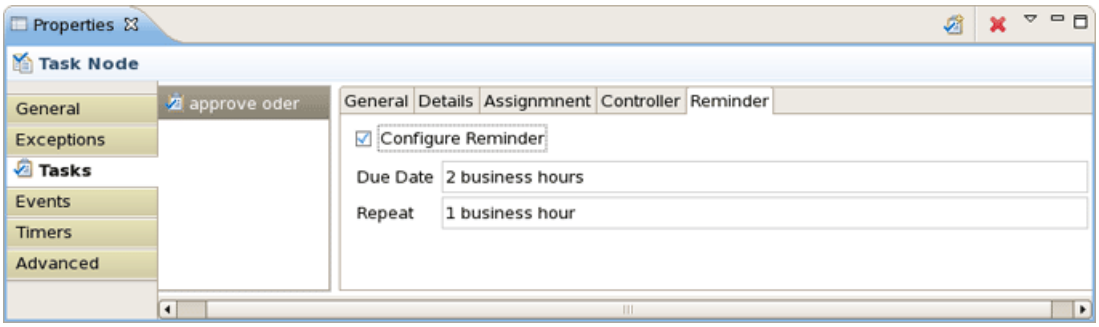

Figure 6.8. The Task Reminder Page

In our case, the assignee will be reminded by email after two business hours and continue to get reminding every business hour after that.

In the next figure you can see our configuring generated into XML.

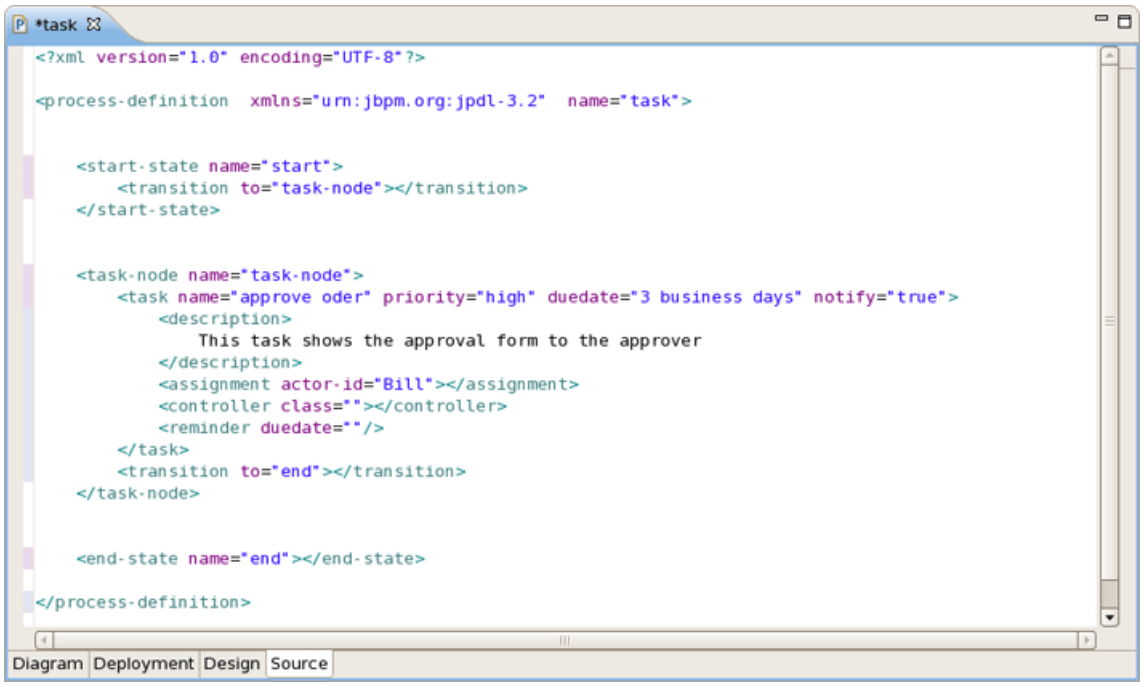

Figure 6.9. The Task Reminder Page

We hope, our guide will help you to get started with the jPDL process language and jBPM workflow on the whole. Besides, for additional information you are welcome on JBoss forum
.