# JBoss ESB Tools User Guide

## Version: 1.3.1.GA

# Introduction

## 1.1. What is ESB and JBoss ESB Tools?

ESB (Enterprise Service Bus) is an abstraction layer on top of the implementation of an enterprise messaging system that provides the features that Service Oriented Architectures may be implemented with.

If you want to develop applications using ESB technology JBoss ESB also meets your needs. Read more about JBoss ESB at *http://www.jboss.org/jbossesb*.

JBoss ESB Tools provides an ESB editor and wizard for creating an ESB file.

This guide will provide you with the information you need to install, configure and deploy JBoss ESB Tools. Information will also be provided on the use of the ESB Editor, which allows you to quickly develop an ESB file with fewer errors.

## 1.2. Key Features of ESB Tools

The table below lists the main features of the ESB plugin:

### Table 1.1. Key Functionality of ESB Tools

| Feature | Benefit | Chapter |
|---|---|---|
| JBoss Tools Project Examples Wizard | Quickly create example projects with a predefined structure. | *Section 2.2, “Using ESB Project Examples”* |
| JBoss Enterprise SOA Platform | The SOA Platform integrates specific versions of JBoss ESB, jBPM, Drools and the JBoss Enterprise Application Platform that are certified to work together in a single supported enterprise distribution. | *Section 2.7, “Using and Configuring SOA Platform”* |
| ESB Editor | JBoss ESB tooling includes a powerful editor with features including syntax validation, support for XML Schema and more. | |

# Tasks

In this section we will focus on all concepts that JBoss Tools integrate for working with JBoss ESB.

## 2.1. Creating a ESB Project

In this chapter we suggest a step-by-step walk-through of creating a new ESB project. Let's try to create a new JBoss ESB project.

We will show you how to use the ESB Project Creation wizard for creating a new ESB project and setting basic ESB classpath.

Select **File** → **New** → **Project...** in the main menu bar or context menu for selected project and then **ESB** → **ESB Project** in the dialog opened:

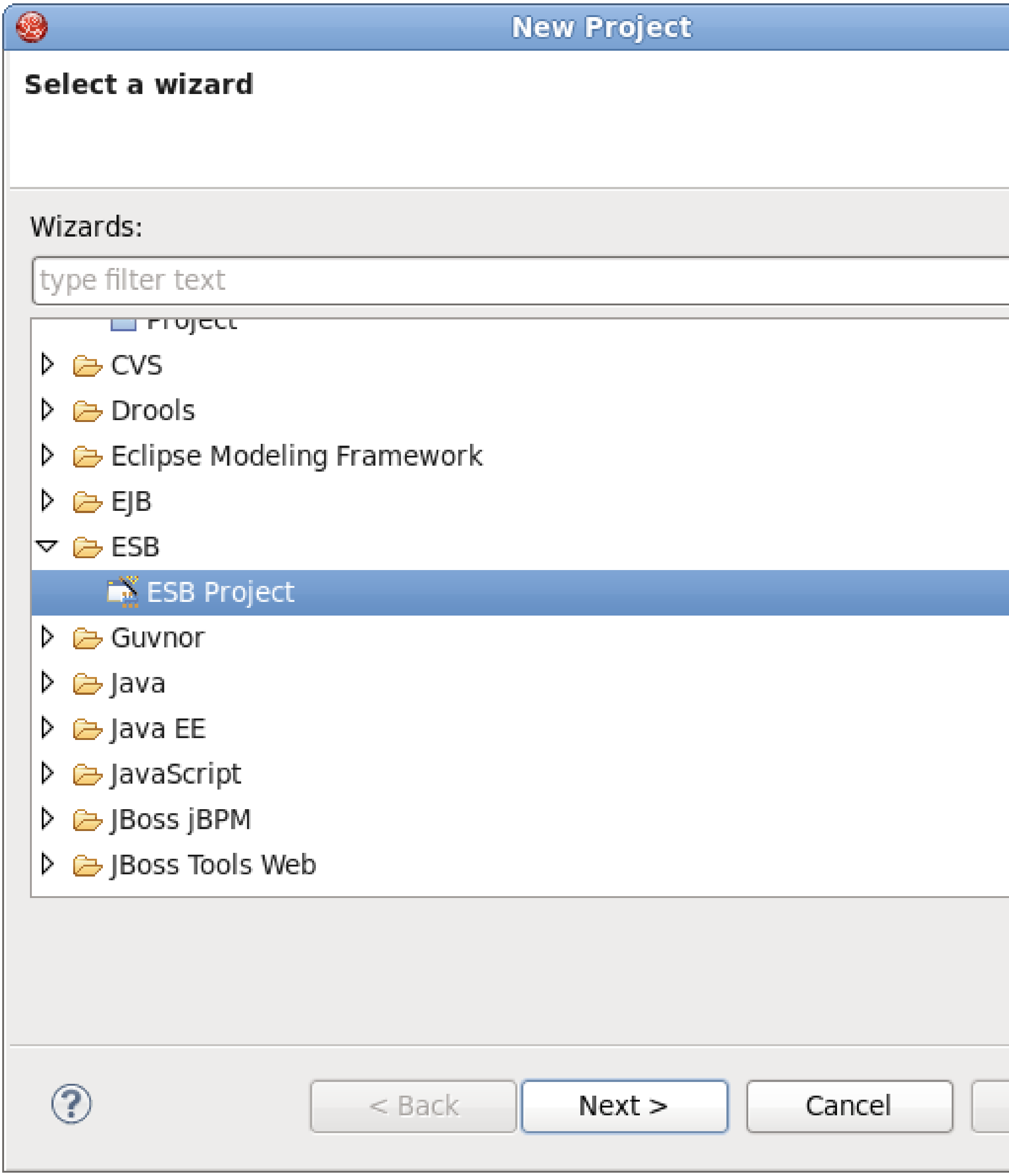
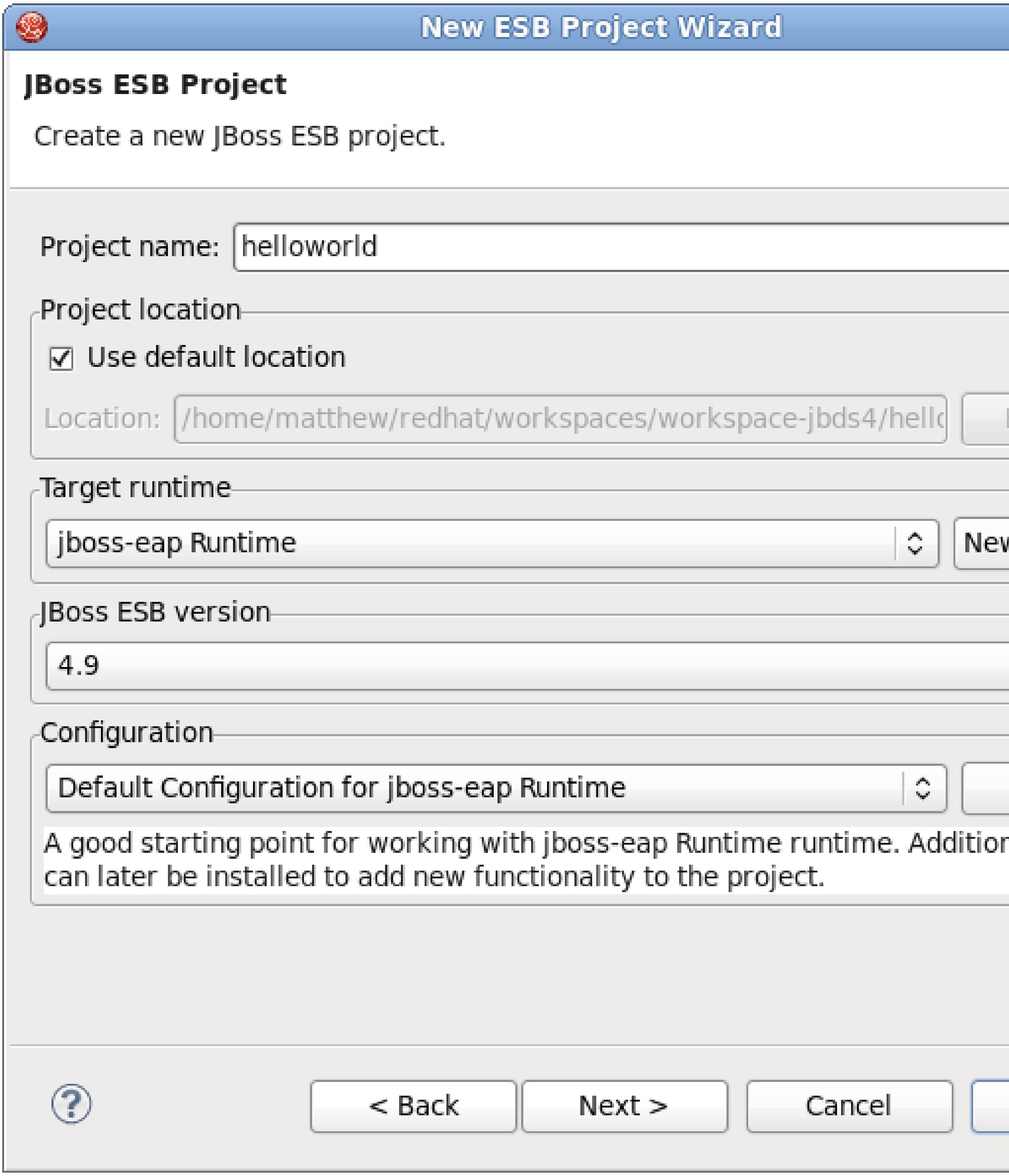


**Figure 2.1. Select a Wizard dialog**

Clicking *Next* brings you to the JBoss ESB Project wizard page where a project name, ESB version and target JBoss Runtime are to be specified. Specify, for example, *helloworld* as a Project name and accept the default ESB version.



New ESB Project Wizard

JBoss ESB Project

Create a new JBoss ESB project.

Project name: helloworld

Project location

☑ Use default location

Location: /home/matthew/redhat/workspaces/workspace-jbds4/hell(

Target runtime

jboss-eap Runtime

JBoss ESB version

4.9

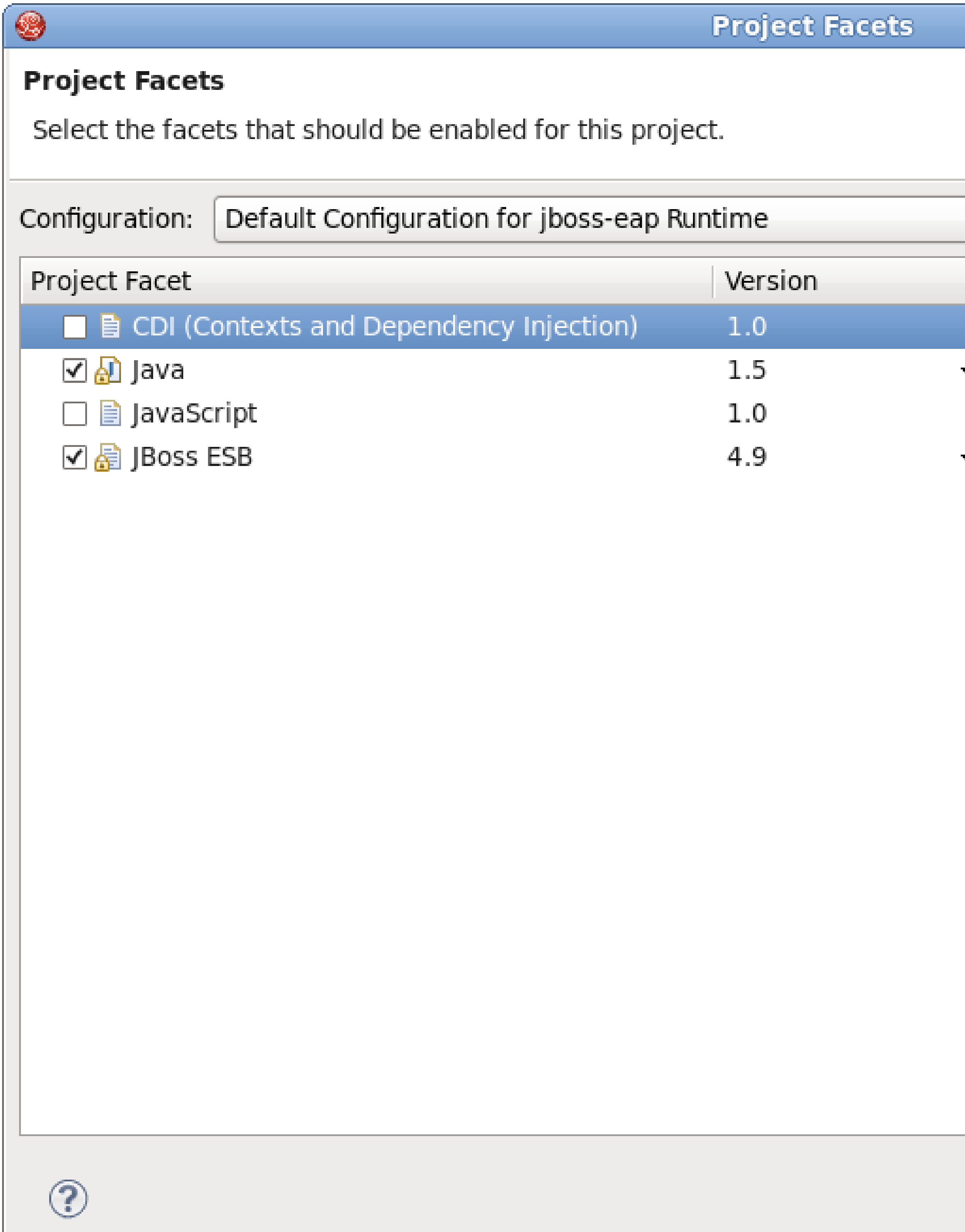
Configuration

Default Configuration for jboss-eap Runtime

A good starting point for working with jboss-eap Runtime runtime. Additior
can later be installed to add new functionality to the project.

Figure 2.2. JBoss ESB Project wizard

By clicking the **Modify** button you can open *Project Facets Wizard* page, where you can select facets that should be enabled for this project. On the *Project Facets Wizard* page you can also configure runtime for the application



**Project Facets**

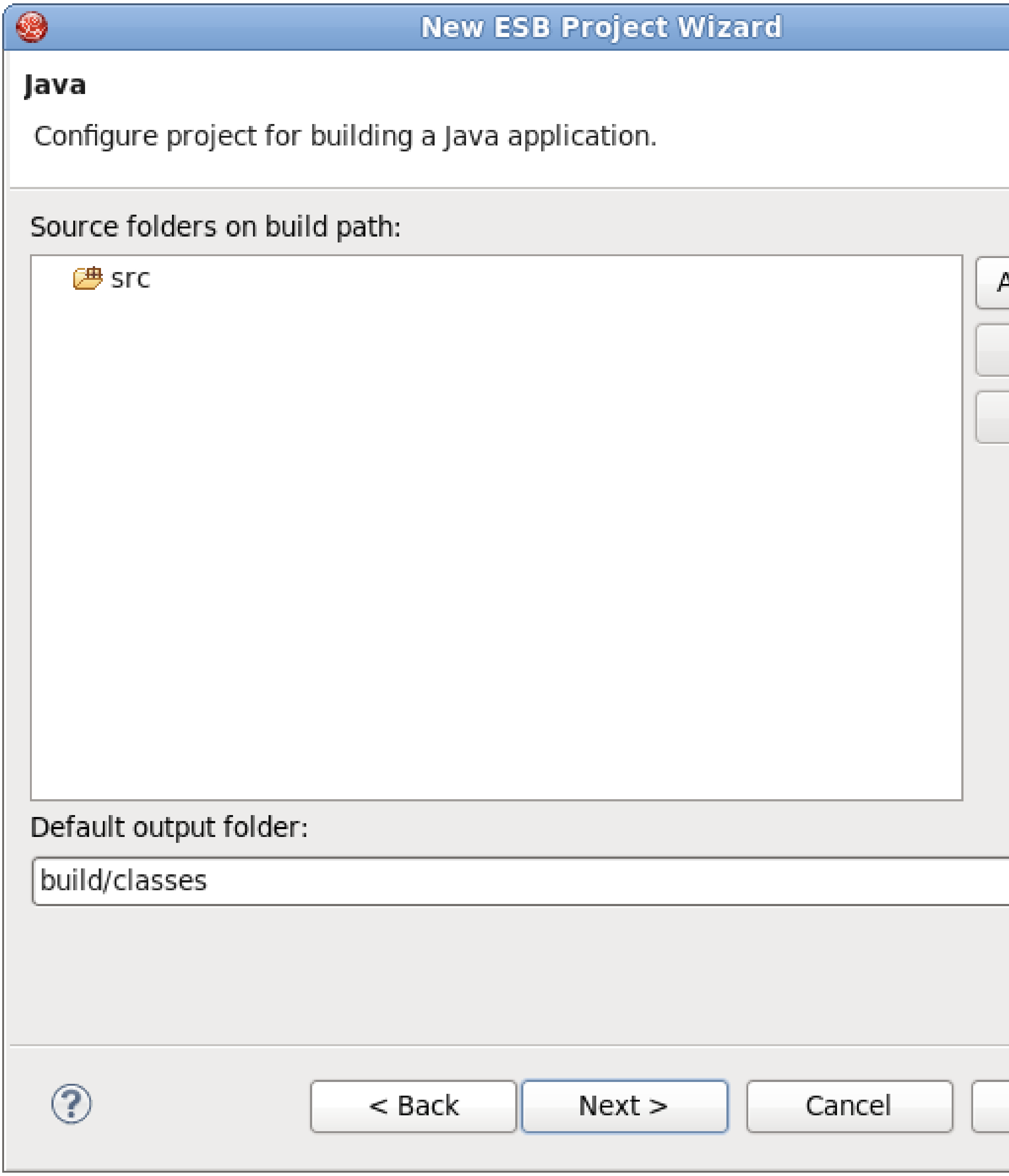
Select the facets that should be enabled for this project.

Configuration: Default Configuration for jboss-eap Runtime

| Project Facet | Version |
|---|---|
| □ CDI (Contexts and Dependency Injection) | 1.0 |
| ☑ Java | 1.5 |
| □ JavaScript | 1.0 |
| ☑ JBoss ESB | 4.9 |

Figure 2.3. Project Facets Wizard

Click the **Next** button to move to the next step in the wizard, where you can select the source and output folders.



Java

Configure project for building a Java application.

Source folders on build path:

src

Default output folder:

build/classes

**Figure 2.4. Project Facets Wizard**

Next step provides you an opportunity to configure your project for building a java application

Clicking the **Next** button on this form brings you to the ESB facet installation page where you can specify the ESB Content Directory. ESB Content Directory is a folder that contains the most of artifacts that an ESB archive needs. You also can configure ESB libraries to the project by selecting a ESB runtime using one of the options:

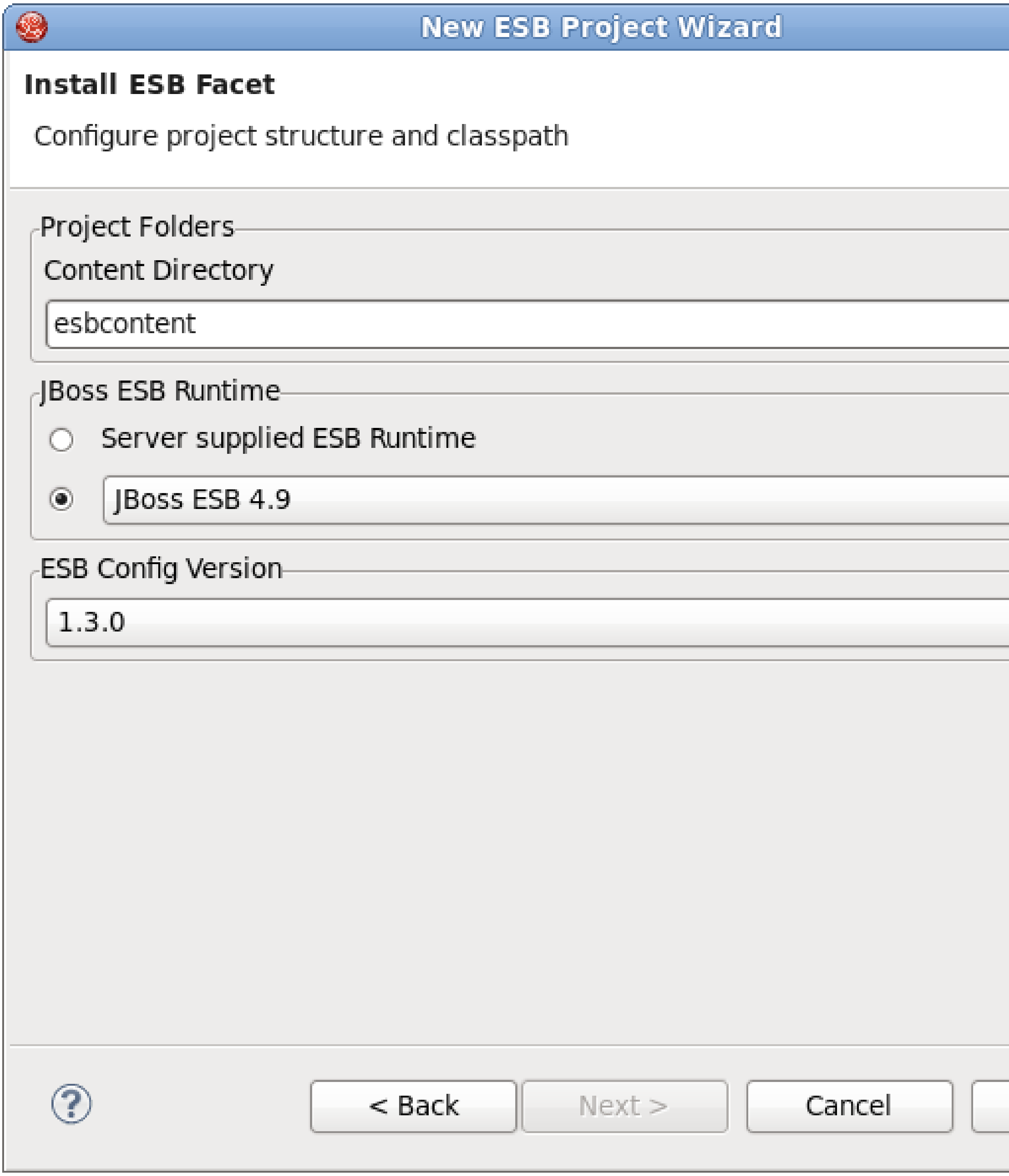
1. Use **Server Supplied ESB Runtime**

2. Select a ESB runtime from the JBoss ESB runtime list predefined in the preferences. If you choose the first option, make sure that the project has the Target JBoss Runtime set and this runtime has a ESB runtime installed.

3. Choose ESB Config Version. From the version 3.1.0 JBoss ESB Tools supports three different jboss-esb.xsd versions: jbossesb-1.0.1.xsd, jbossesb-1.1.0.xsd and jbossesb-1.2.0.xsd.

### Note:

If you use ESB 4.7 you should select jbossesb-1.2.0.xsd.



New ESB Project Wizard

Install ESB Facet

Configure project structure and classpath

Project Folders

Content Directory

esbcontent

JBoss ESB Runtime

○ Server supplied ESB Runtime

● JBoss ESB 4.9

ESB Config Version

1.3.0

**Figure 2.5. Install ESB facet step**

Click the **Finish** button and a ESB project with the default `jboss-esb.xml` file will be created.

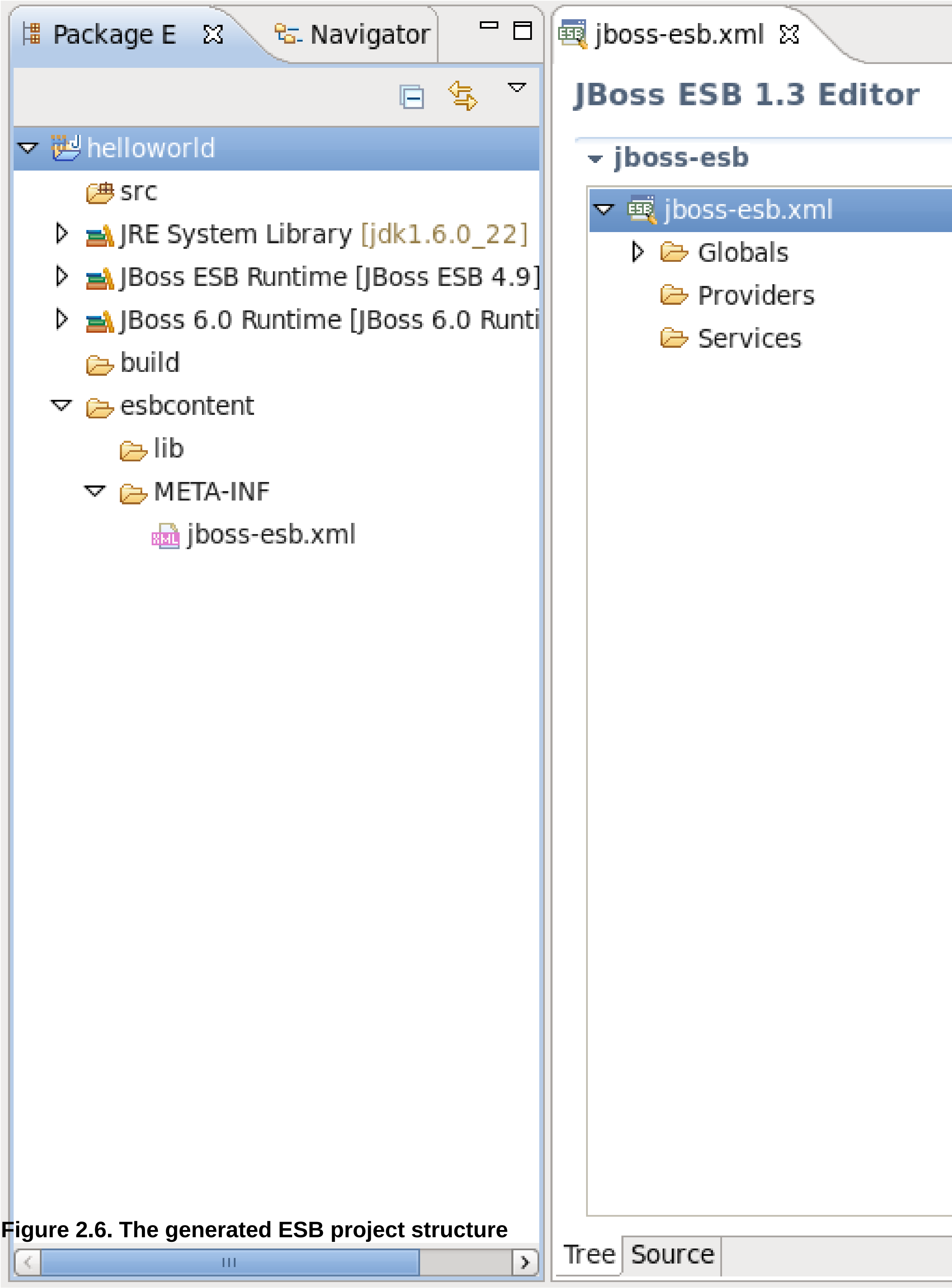


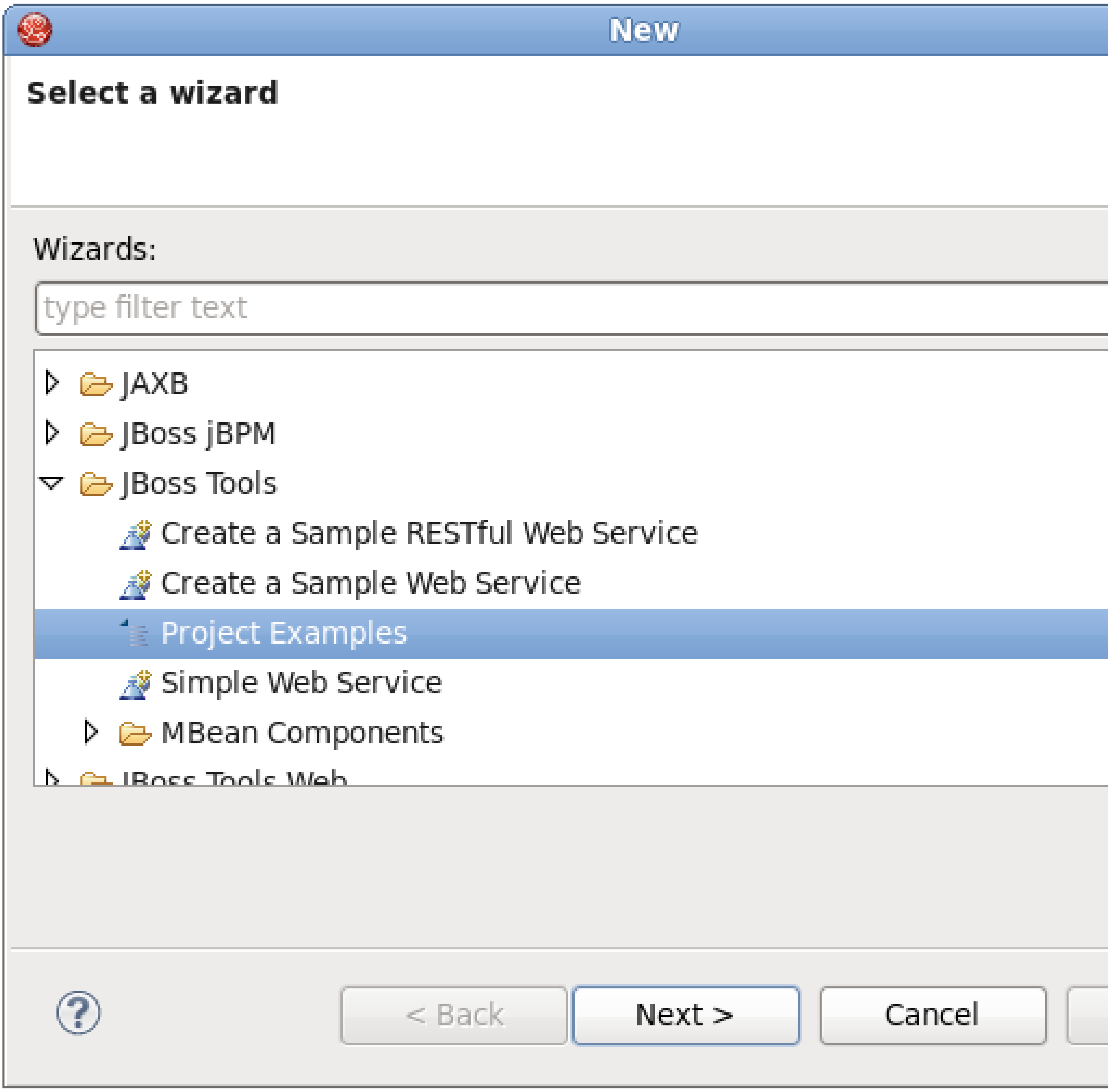
Figure 2.6. The generated ESB project structure

## 2.2. Using ESB Project Examples

JBoss Tools provides a Project Example wizard that is an easy way for users to create some kinds of projects to be used as examples with some predefined structure. Let's start creating a ESB project using this wizard.

Before creating a ESB project example create JBoss Runtime with name *JBoss 4.2 Runtime*, it will be used by your ESB project example.

Select **File** → **New** → **Other...** in the main menu bar or context menu for selected project and then **JBoss Tools** → **Project Examples** in the New dialog:



New

Select a wizard

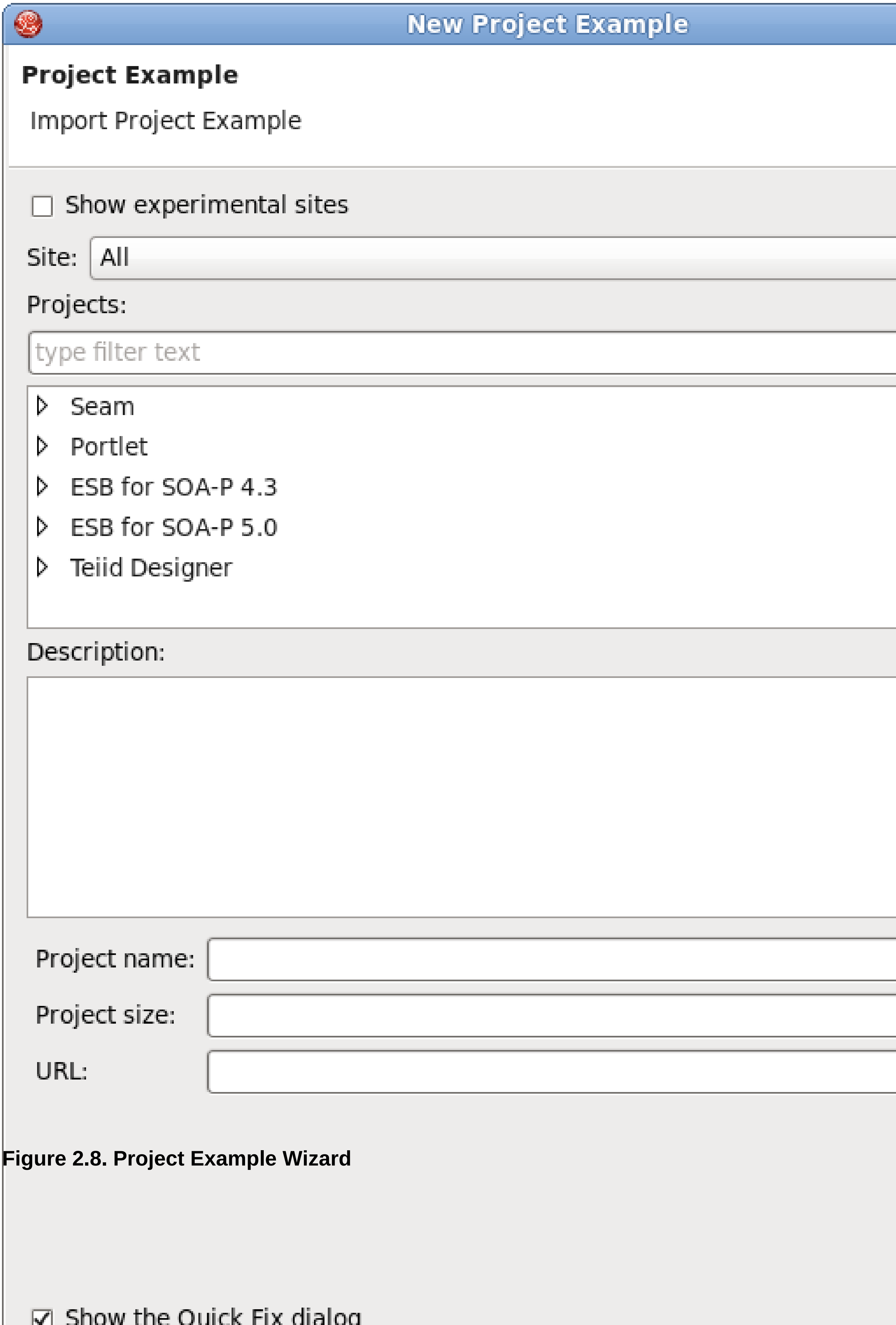
Wizards:

type filter text

- JAXB
- JBoss jBPM
- JBoss Tools
  - Create a Sample RESTful Web Service
  - Create a Sample Web Service
  - Project Examples
  - Simple Web Service
  - MBean Components
- JBoss Tools Web

Figure 2.7. Select a wizard - Project Examples

Clicking the **Next** button brings you to the wizard page where you can select a ESB project example from the example list.



New Project Example

**Project Example**

Import Project Example

□ Show experimental sites

Site: All

Projects:

type filter text

- Seam
- Portlet
- ESB for SOA-P 4.3
- ESB for SOA-P 5.0
- Teiid Designer

Description:

Project name:

Project size:

URL:

☑ Show the Quick Fix dialog

**Figure 2.8. Project Example Wizard**

> **Note:**
>
> Under the Projects section you can find two categories related to ESB:
>
> - ESB for SOA-P 4.3
> - ESB for SOA-P 5.0
>
> It means that if you use a runtime pointed to a SOA-P 5.0, you should download the examples from the *ESB for SOA-P 5.0* category in order to avoid the incompatibility errors.

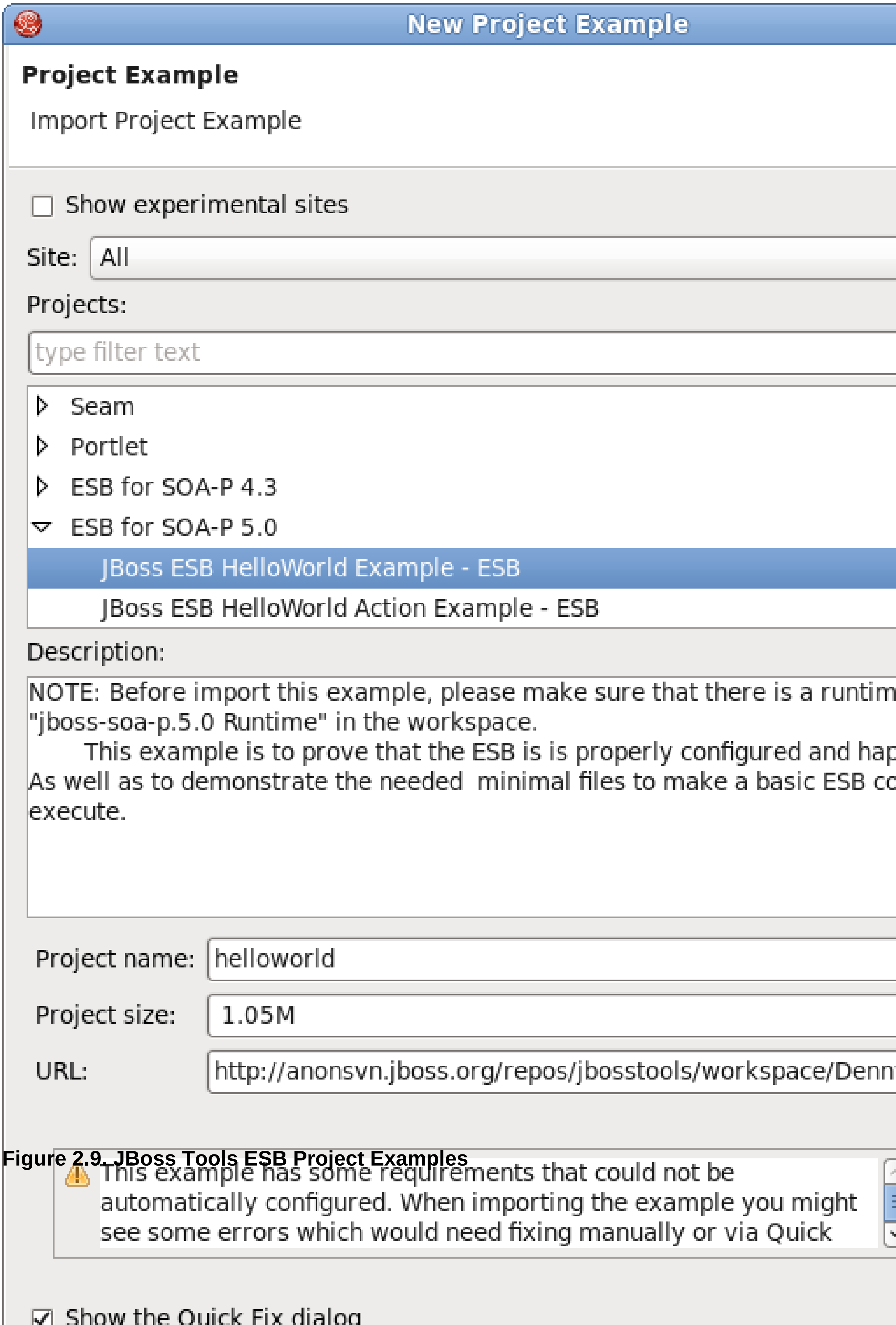
Every ESB example has two projects, one is a ESB project and another is a Java project used to test the ESB project.

Here is a list of ready examples available in both categories (ESB for SOA-P 4.3 and ESB for SOA-P 5.0):

- **JBoss ESB HelloWorld Example** - demonstrates the minimal files necessary to make a basic ESB component execute as well as to prove that the ESB is properly configured.
- **JBoss ESB HelloWorld Action Example** - demonstrates the use of multiple action invocations from a single configuration. You can use a single Action class and make multiple method calls or use multiple Action classes.
- **JBoss ESB HelloWorld Email Example** - demonstrates how to configure a mail gateway using a JCA mail adapter and a Camel mail endpoint. Both gateways route the messages through a JMS queue and finally to your action class for processing.
- **JBoss ESB HelloWorld File Action Example** - demonstrates using the File gateway feature of the JBoss ESB. Files that are found in a particular directory with a particular extension are sent to a JMS queue with actions for processing.
- **JBoss ESB Web Service consumer1 Example** - demonstrates how to consume a 181 Web Service in an ESB action.
- **JBoss ESB Web Service producer Example** - demonstrates how to deploy a JSR181 Webservice endpoint on JBossESB using the SOAPProcessor action.
- **JBoss ESB Smooks CSV -> XML Example** - demonstrates how to transform a comma separated value (CSV) file to an XML.
- **JBoss ESB Smooks XML -> POJO Example** - demonstrates the use of Smooks performing a simple transformation by converting an XML file into Java POJOs.
- **JBoss ESB Smooks XML -> XML date-manipulation Example** - demonstrates how to manually define and apply a Message Transformation within JBoss ESB.

- **JBoss ESB Smooks XML -> XML Example** - a very basic example of how to manually define and apply a Message Transformation within JBoss ESB. It applies a very simple XSLT to a SampleOrder.xml message and prints the before and after XML to the console.

We will take as our example **JBoss ESB HelloWorld Example** ESB project:



New Project Example

**Project Example**

Import Project Example

□ Show experimental sites

Site: All

Projects:

type filter text

- Seam
- Portlet
- ESB for SOA-P 4.3
- ESB for SOA-P 5.0
  - JBoss ESB HelloWorld Example - ESB
  - JBoss ESB HelloWorld Action Example - ESB

Description:

NOTE: Before import this example, please make sure that there is a runtim
"jboss-soa-p.5.0 Runtime" in the workspace.
This example is to prove that the ESB is is properly configured and hap
As well as to demonstrate the needed minimal files to make a basic ESB co
execute.

Project name: helloworld

Project size: 1.05M

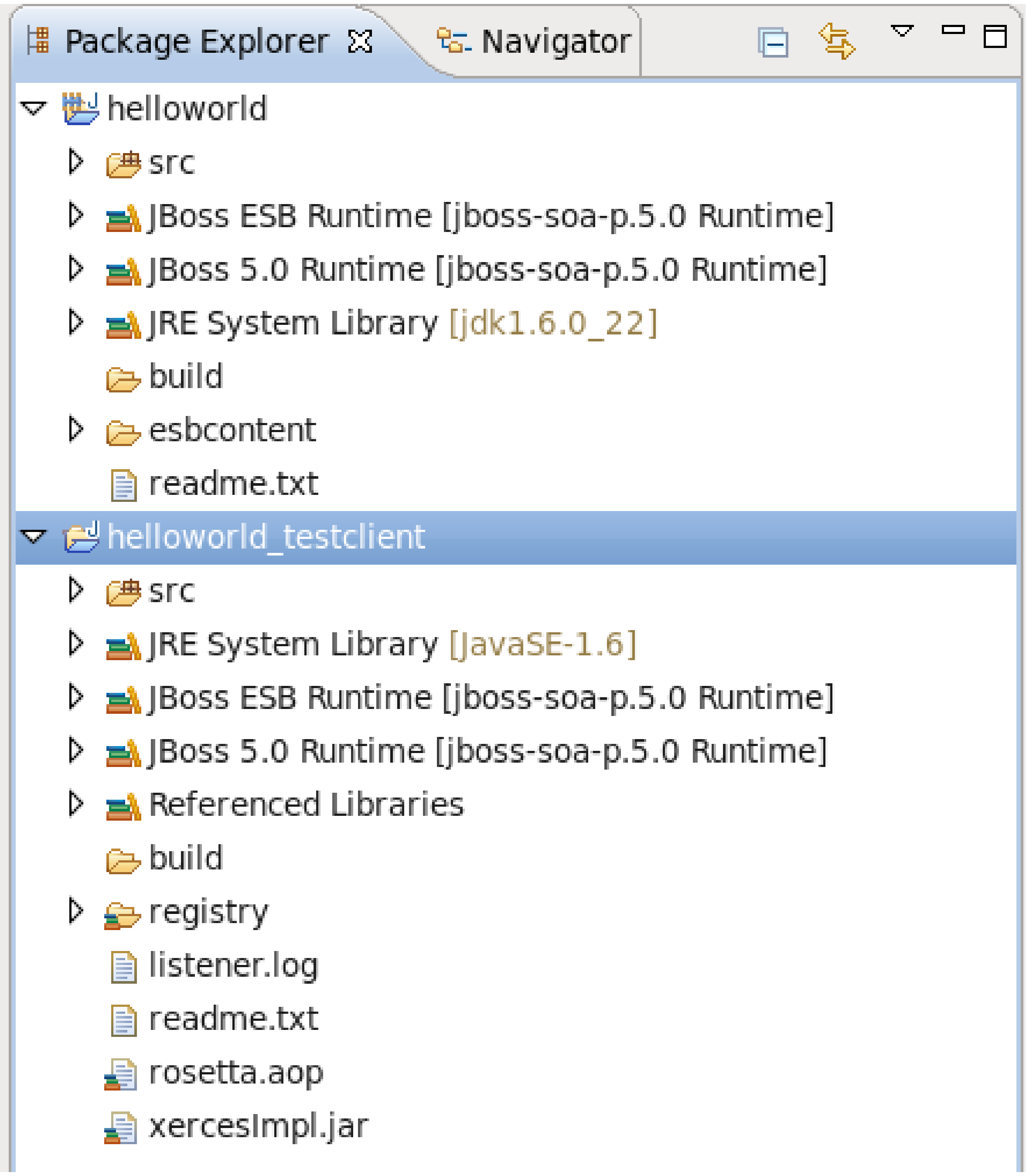
URL: http://anonsvn.jboss.org/repos/jbosstools/workspace/Denn

⚠ This example has some requirements that could not be automatically configured. When importing the example you might see some errors which would need fixing manually or via Quick

☑ Show the Quick Fix dialog

**Figure 2.9. JBoss Tools ESB Project Examples**

Select the **JBoss ESB HelloWorld Example** ESB project and then click the **Finish** button. As a result you will get two projects created:



```
helloworld
    src
    JBoss ESB Runtime [jboss-soa-p.5.0 Runtime]
    JBoss 5.0 Runtime [jboss-soa-p.5.0 Runtime]
    JRE System Library [jdk1.6.0_22]
    build
    esbcontent
    readme.txt
helloworld_testclient
    src
    JRE System Library [JavaSE-1.6]
    JBoss ESB Runtime [jboss-soa-p.5.0 Runtime]
    JBoss 5.0 Runtime [jboss-soa-p.5.0 Runtime]
    Referenced Libraries
    build
    registry
    listener.log
    readme.txt
    rosetta.aop
    xercesImpl.jar
```

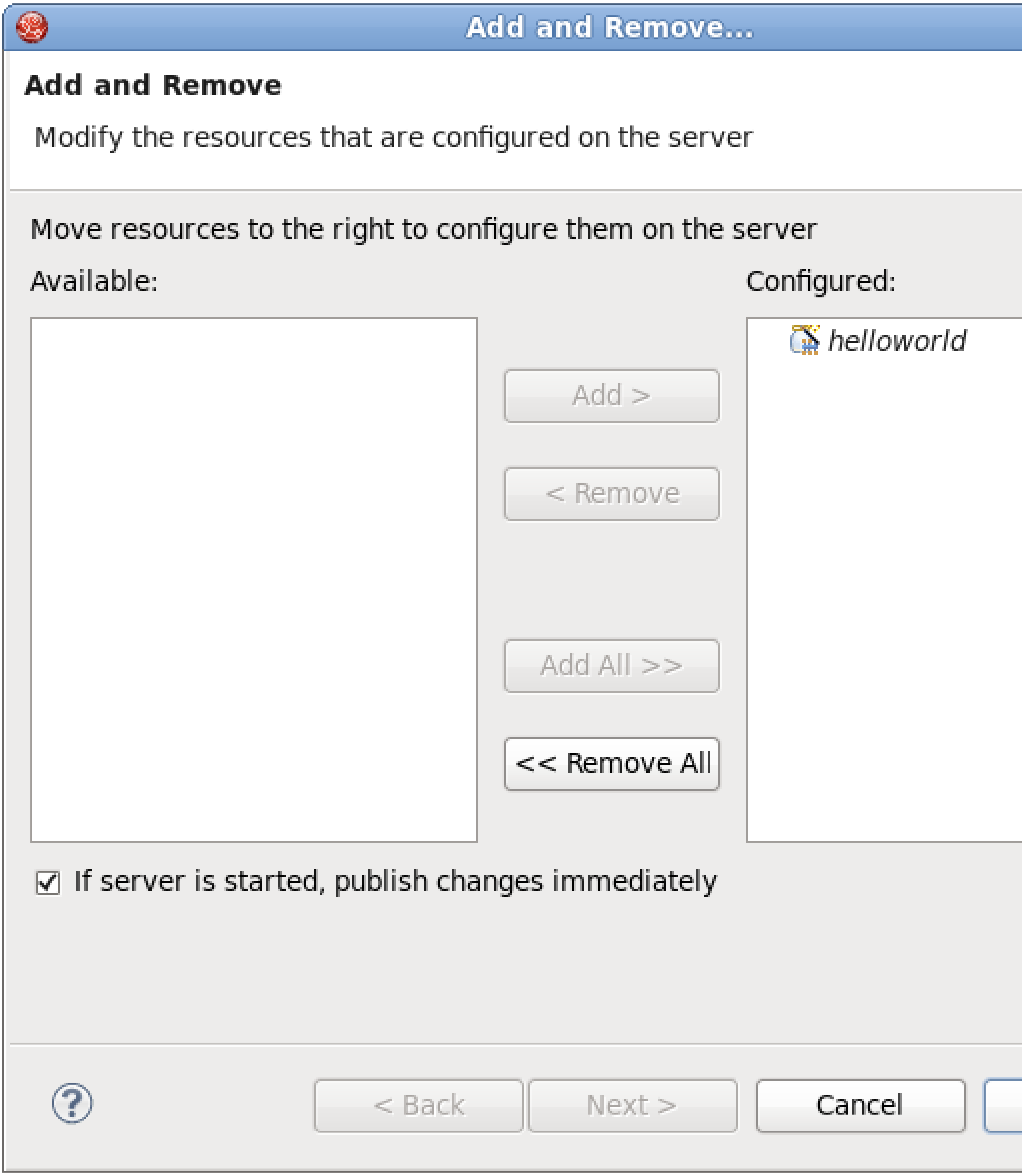
**Figure 2.10. JBoss ESB Project Examples: helloworld and helloworld_testclient**

Deploy the HelloWorld ESB project and run a test class in the client Java project to see the test result in the Console view.

## 2.3. Deploying a ESB Project

In this chapter you will see how to deploy a ESB project using the WTP deployment framework.

Before deploying the project, open the Servers View by selecting **Window** → **Show View** → **Other** → **Server** → **Servers**, create a JBoss Server in the Server view and start it, and then right click the created JBoss server, select **Add and Remove Projects**, and add the ESB projects you want to deploy from the left side to the right side in the opened dialog.



Add and Remove...

Add and Remove

Modify the resources that are configured on the server

Move resources to the right to configure them on the server

Available:

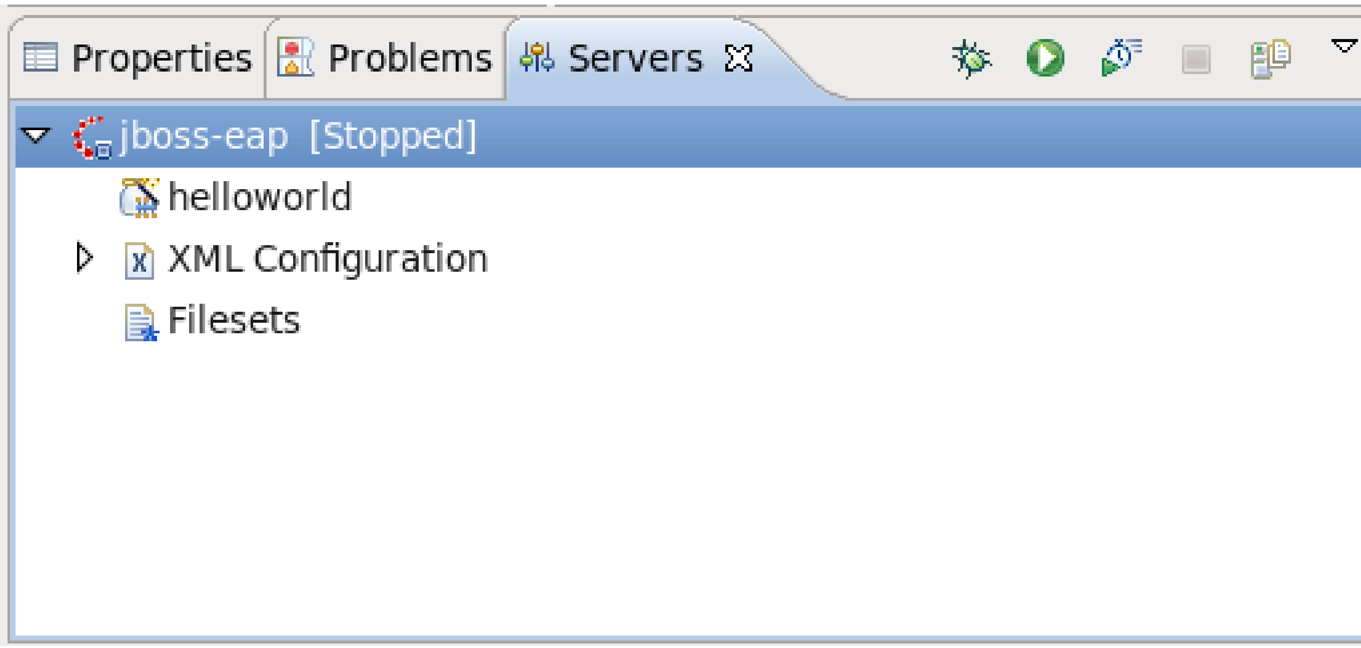
Configured:

helloworld

☑ If server is started, publish changes immediately

Figure 2.11. Add and Remove Projects

Click the **Finish** button to add the project to the server. You also can drag the ESB project from the Project View to the server.

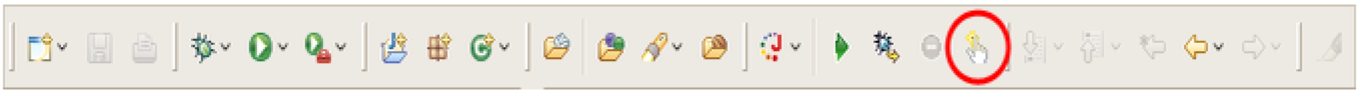


### Figure 2.12. Servers View

Thus, you have just added the ESB project to the JBoss server module list. Right click the JBoss Server and select **Publish** to publish the project on the server. You can check the deploying result in the Console view.

The **Run** and **Debug** options work on ESB projects causing a (re)deploy for a user designated server.

You can also use the "Finger touch" for a quick restart of the project without restarting the server:



### Figure 2.13. Finger Touch button

The "Finger" touches descriptors dependent on project (i.e. `web.xml` for WAR, `application.xml` for EAR) and now it is also available for `jboss-esb.xml` in ESB projects.

You can also deploy your ESB project as an .esb archive. Right-click on the project, choose **Export**:

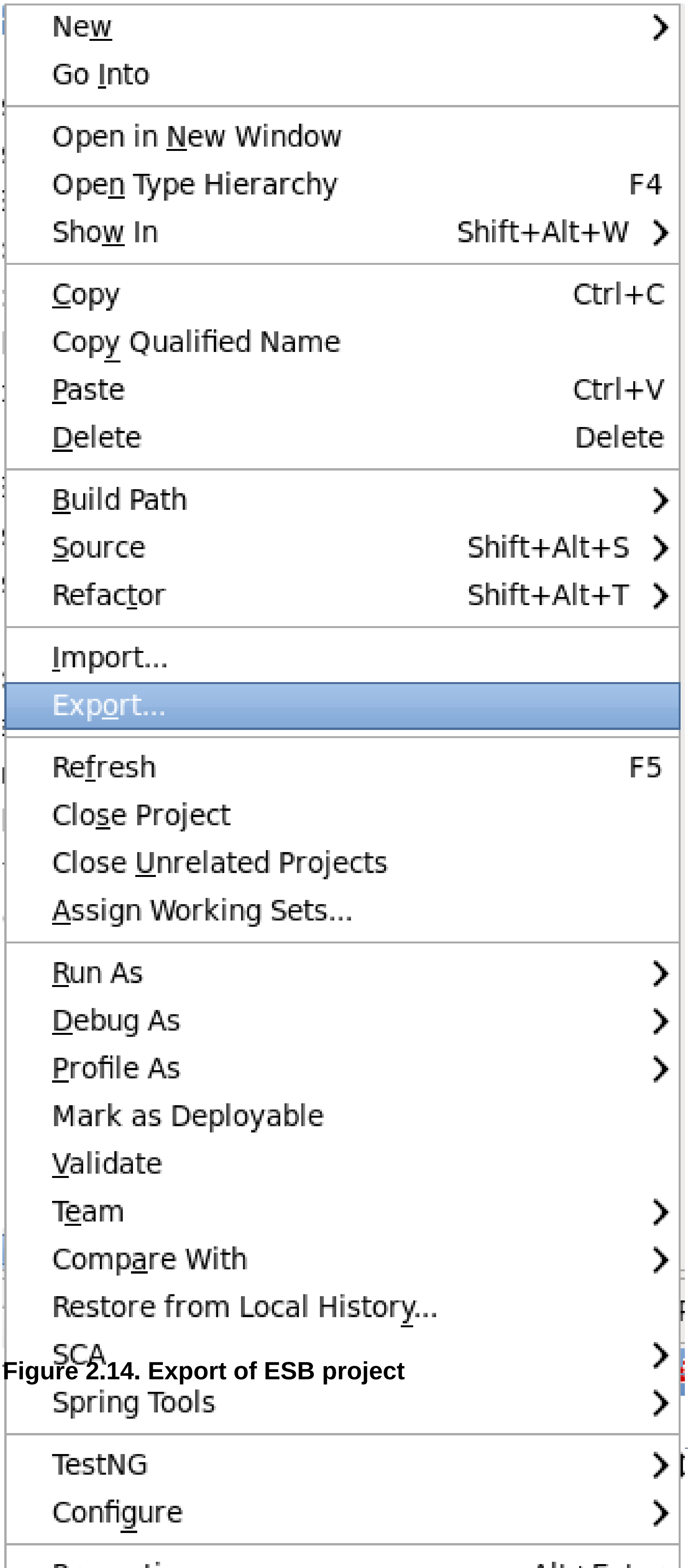


**Figure 2.14. Export of ESB project**

Choose **ESB** → **ESB File** and click **Next**:

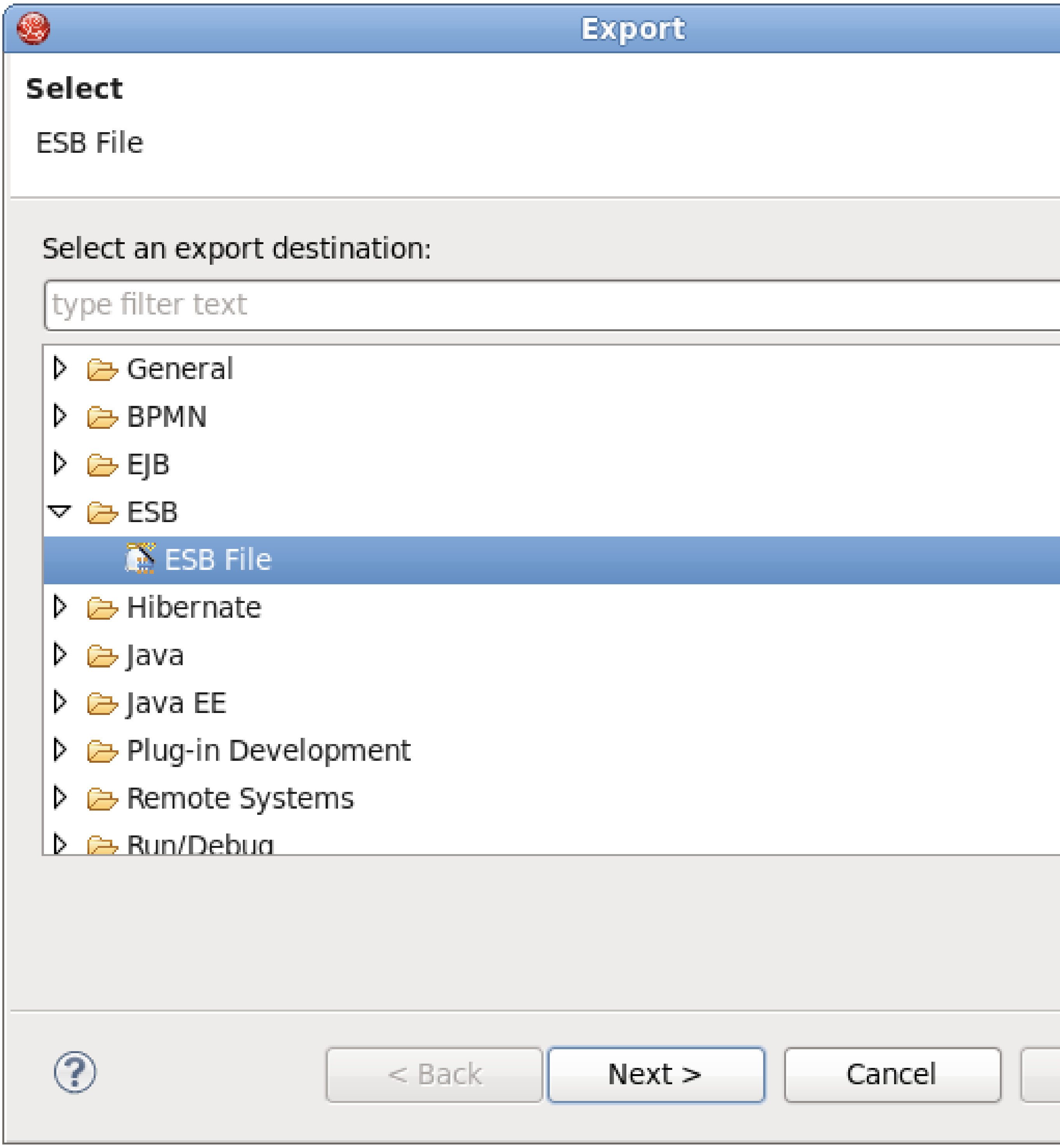
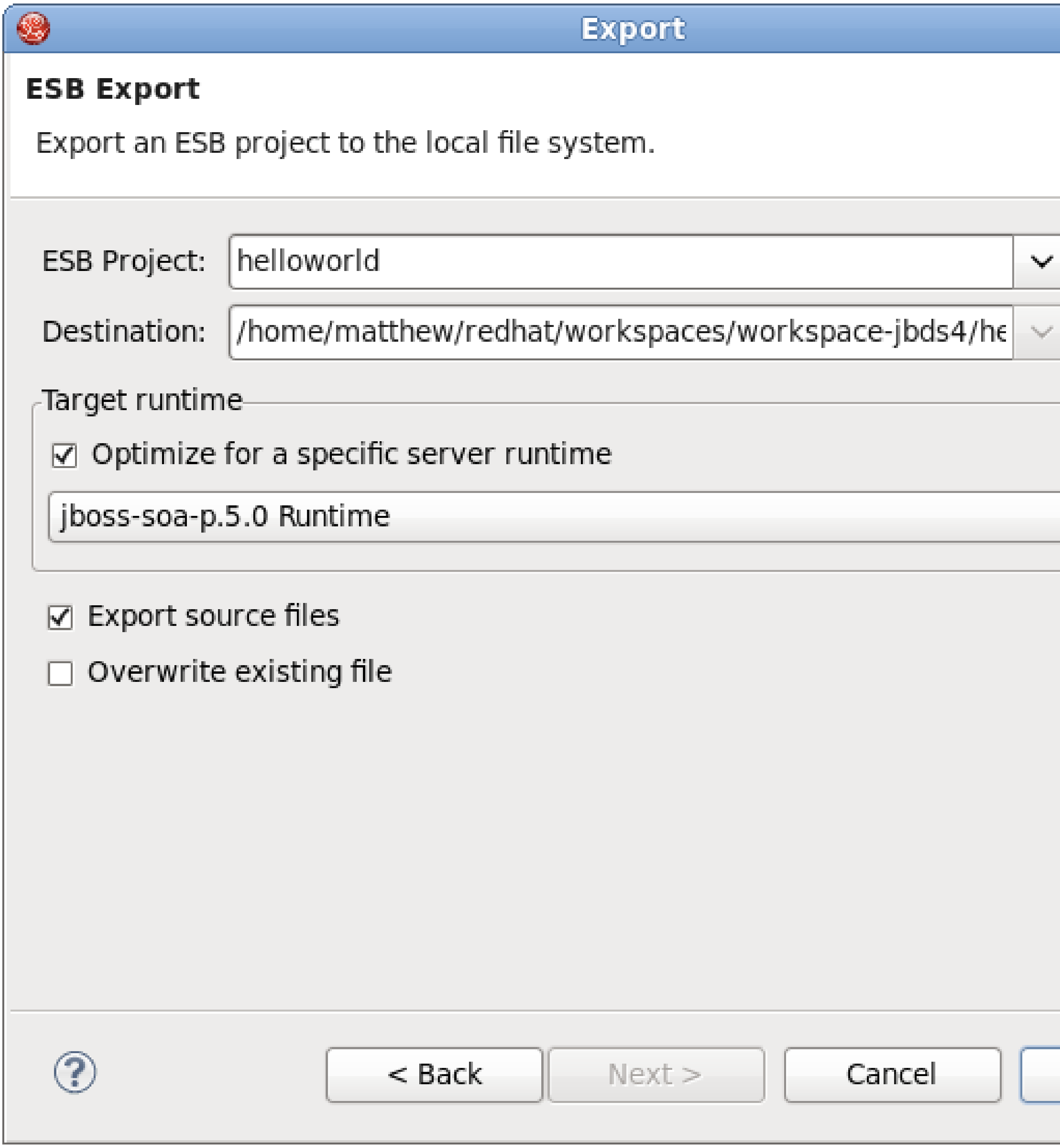


**Figure 2.15. Choosing ESB File**

And finally export the ESB project to the file system: choose the destination, choose the target runtime if need a specific one and make the appropriate settings for the archive. Then click **Finish**.



ESB Export

Export an ESB project to the local file system.

ESB Project: helloworld

Destination: /home/matthew/redhat/workspaces/workspace-jbds4/he

Target runtime

☑ Optimize for a specific server runtime

jboss-soa-p.5.0 Runtime

☑ Export source files

□ Overwrite existing file

**Figure 2.16. ESB Export**

Your project is deployed as an .esb archive.

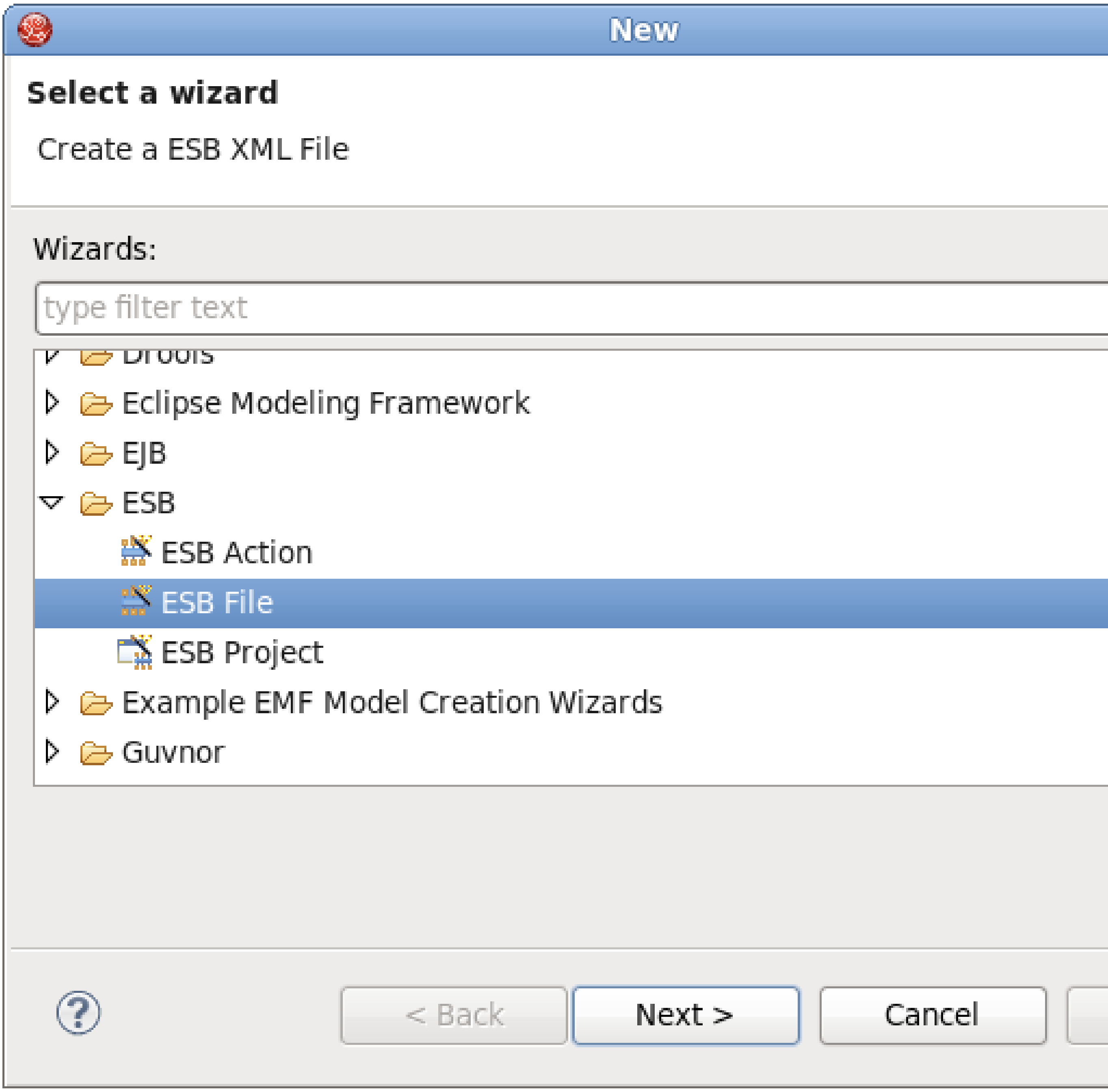
An ESB archive can be created for ESB projects only. It is also possible to deploy an .esb archive to a JBoss AS based server with JBoss ESB installed.

## 2.4. Creating a ESB File

In this chapter we suggest a step-by-step walk-through of creating your own simple file. Let's try to organize a new ESB file.

We will show you how to use the Creation wizard for creating a new ESB file.

At first you should open a project that has the ESB facet enabled. Select **File** → **New** → **Other...** in the main menu bar or context menu for selected project and then **ESB** → **ESB File** in the New dialog:



New

Select a wizard

Create a ESB XML File

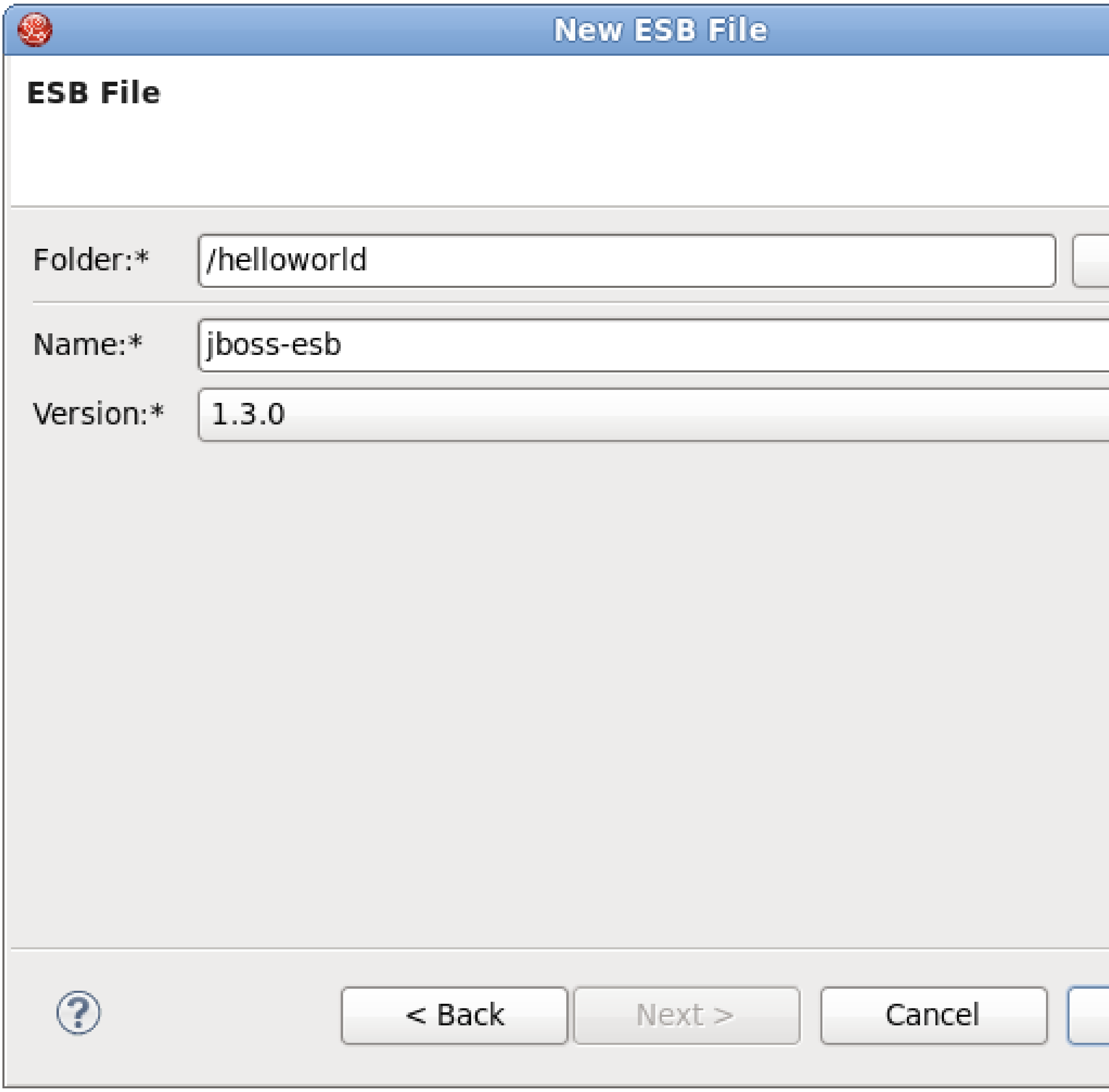
Wizards:

**Figure 2.17. Select a wizard - ESB File**

Clicking *Next* brings you to the wizard page where a folder, a name and a version for the file should be specified. Choose, for example, *jboss-esb.xml* as the name and accept the selected projects folder and the default version.

**Note:**

From the version 3.1.0 JBoss ESB Tools supports three different jboss-esb.xsd versions: jbossesb-1.0.1.xsd, jbossesb-1.1.0.xsd and jbossesb-1.2.0.xsd. If you use ESB 4.7 you should select jbossesb-1.2.0.xsd.



Folder:* /helloworld

Name:* jboss-esb

Version:* 1.3.0

Figure 2.18. Folder, Name and Version for ESB file

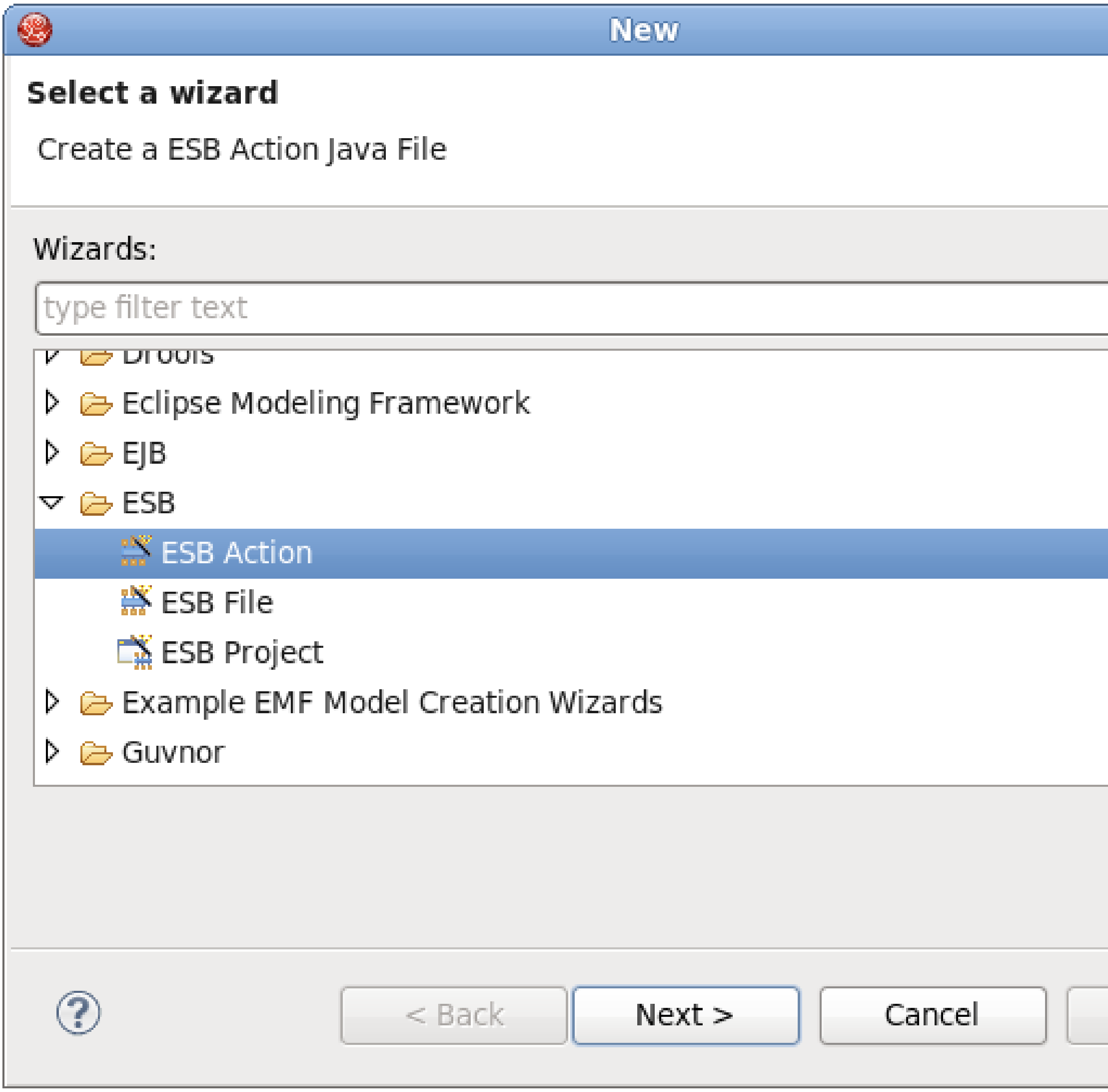
Thus, your file will be created in the selected projects folder by default. If you want to change the folder for your future file click *Browse...* button to set needed folder or simply type it.

Clicking on *Finish* results in the file being generated. The wizard creates one xml file.

## 2.5. Creating a ESB Action

From this chapter you will find out how to create a *ESB Action Java File* .

At first you need to open a ESB or simple Java project. Then you should select **File** → **New** → **Other** in the main menu or from the context project menu. Then click **ESB** → **ESB Action** in the *New* dialog.



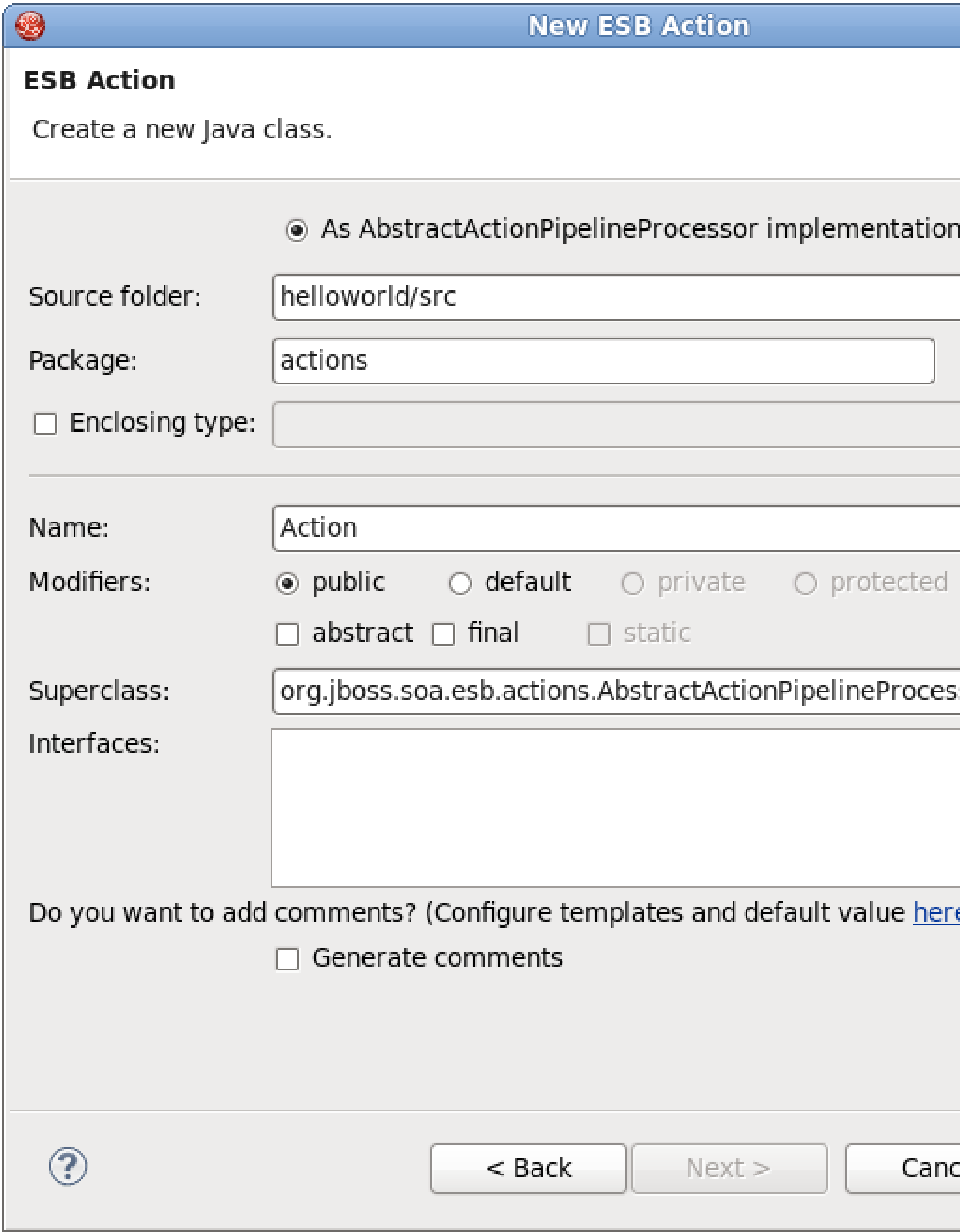
New

Select a wizard

Create a ESB Action Java File

**Figure 2.19. Select a wizard - ESB Action**

After that click *Next* and you will be brought to the *New ESB Action* wizard. In this wizard the class name should be specified, also you can set a package or add a interface as for any Java class.



New ESB Action

ESB Action

Create a new Java class.

● As AbstractActionPipelineProcessor implementation

Source folder: helloworld/src

Package: actions

□ Enclosing type:

Name: Action

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: org.jboss.soa.esb.actions.AbstractActionPipelineProces

Interfaces:

Do you want to add comments? (Configure templates and default value here

□ Generate comments

**Figure 2.20. New ESB Action wizard**

As a result, the ESB Action Java File will be created in the selected package and it will have org.jboss.soa.esb.actions.AbstractActionPipelineProcessor as superclass.

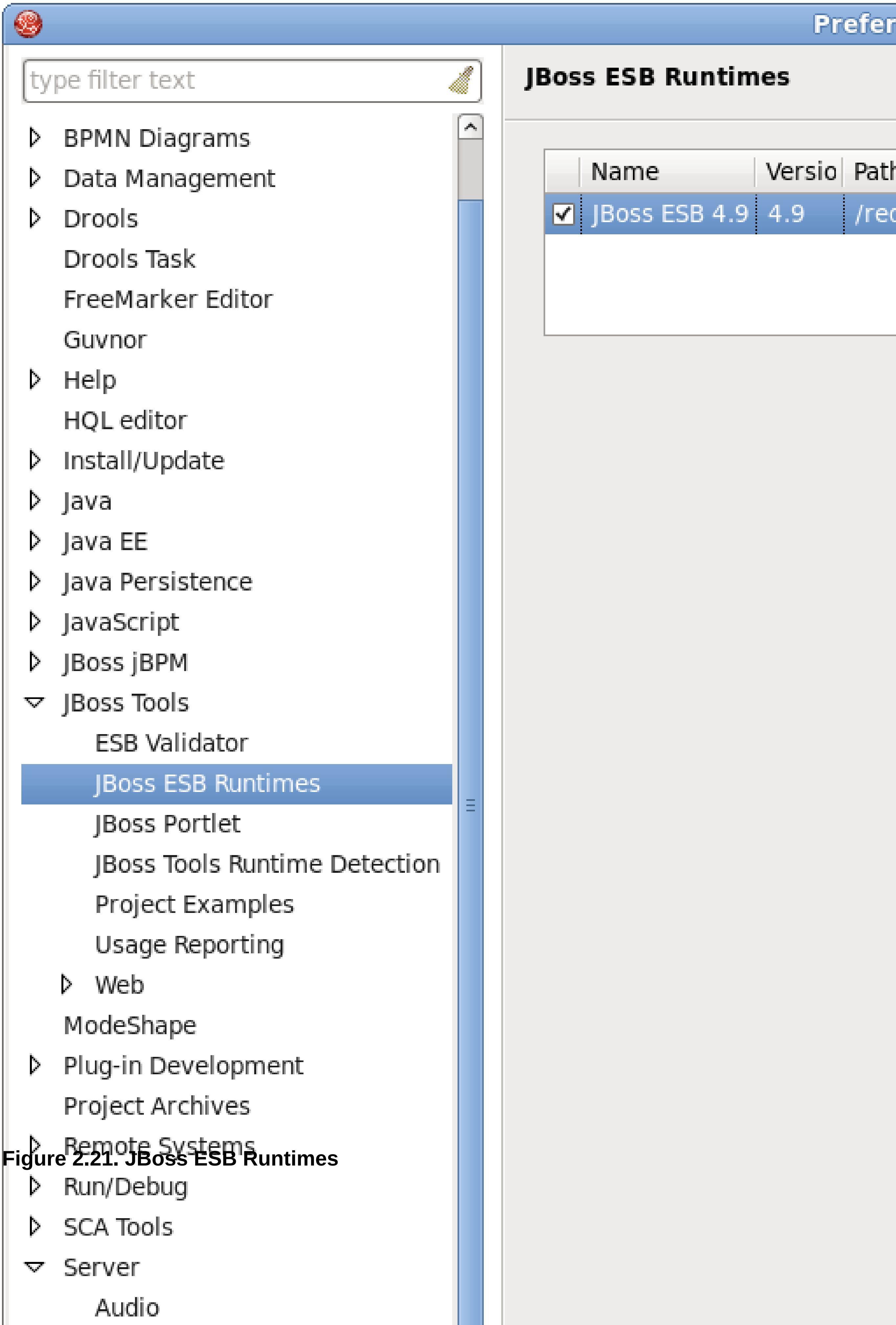
Clicking on *Finish* will generate the ESB Action class. Also this class will become available in ESB Editor wizards.

## 2.6. Configuring ESB Runtime in Preferences

In this chapter you will learn how to predefine a JBoss ESB runtime on the Preferences page.

You may already know, there are two ways to set JBoss ESB runtime when creating a ESB project, one is to use the project target JBoss runtime, and another is to select a JBoss ESB runtime predefined in JBoss Tools preferences. Let's configure it.

Select **Window** → **Preferences** → **JBoss Tools** → **JBoss ESB Runtimes** to open the JBoss ESB Runtime Preferences page where you can add, remove and Edit a JBoss ESB runtime.

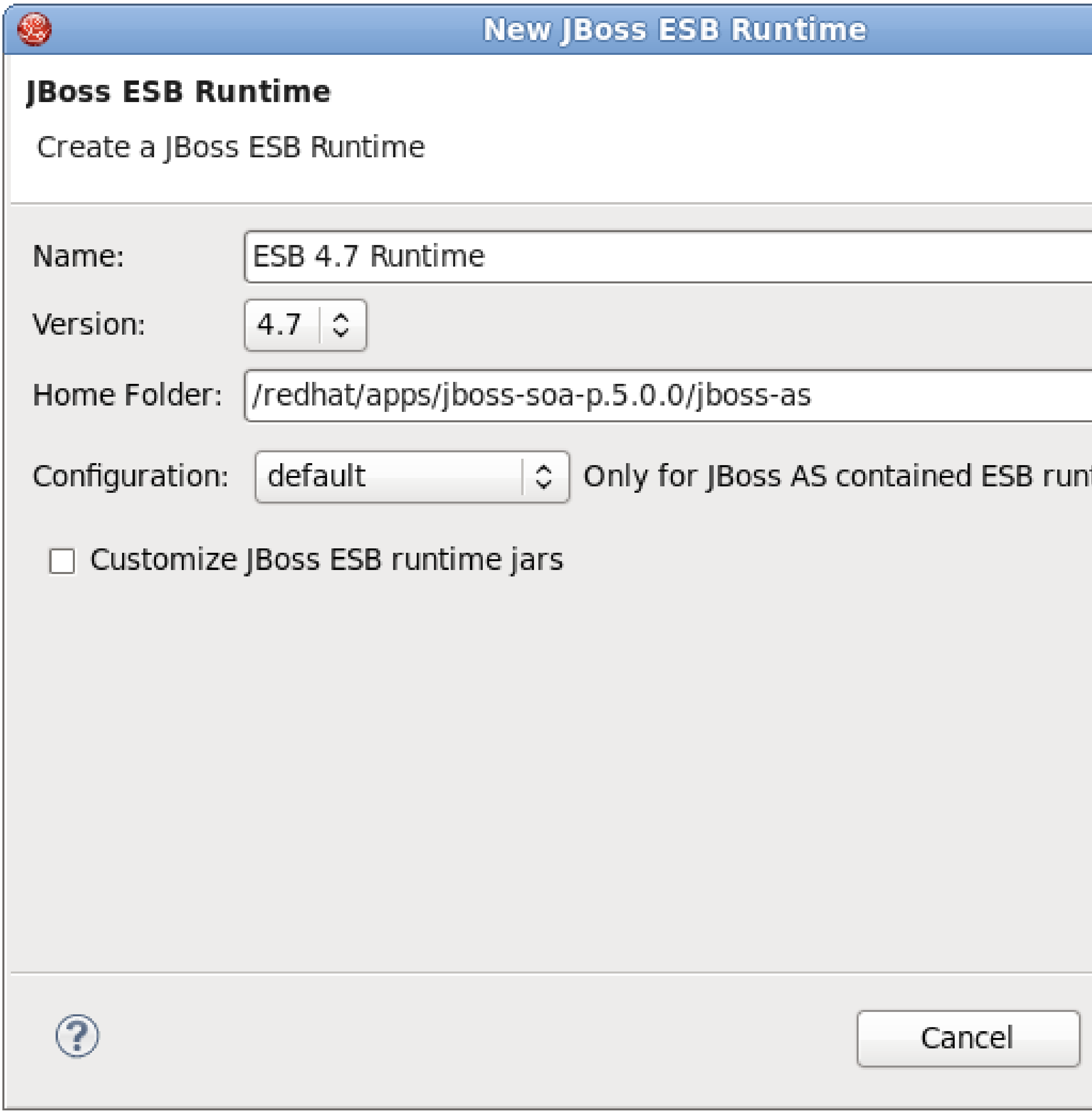


JBoss ESB Runtimes

| | Name | Versio | Path |
|---|---|---|---|
| ☑ | JBoss ESB 4.9 | 4.9 | /rec |

Figure 2.21. JBoss ESB Runtimes

Select *Add* to open a dialog where you can specify the JBoss ESB runtime location, name and version number. It's also possible to define configuration if you point the home location to a Jboss AS or SOA-p, in case you select a standalone ESB runtime location, the configuration combo will be empty and should be ignored. You can also customize the libraries of the runtime by checking the *Customize JBoss ESB Runtime jars* checkbox.



JBoss ESB Runtime

Create a JBoss ESB Runtime

Name: ESB 4.7 Runtime

Version: 4.7

Home Folder: /redhat/apps/jboss-soa-p.5.0.0/jboss-as

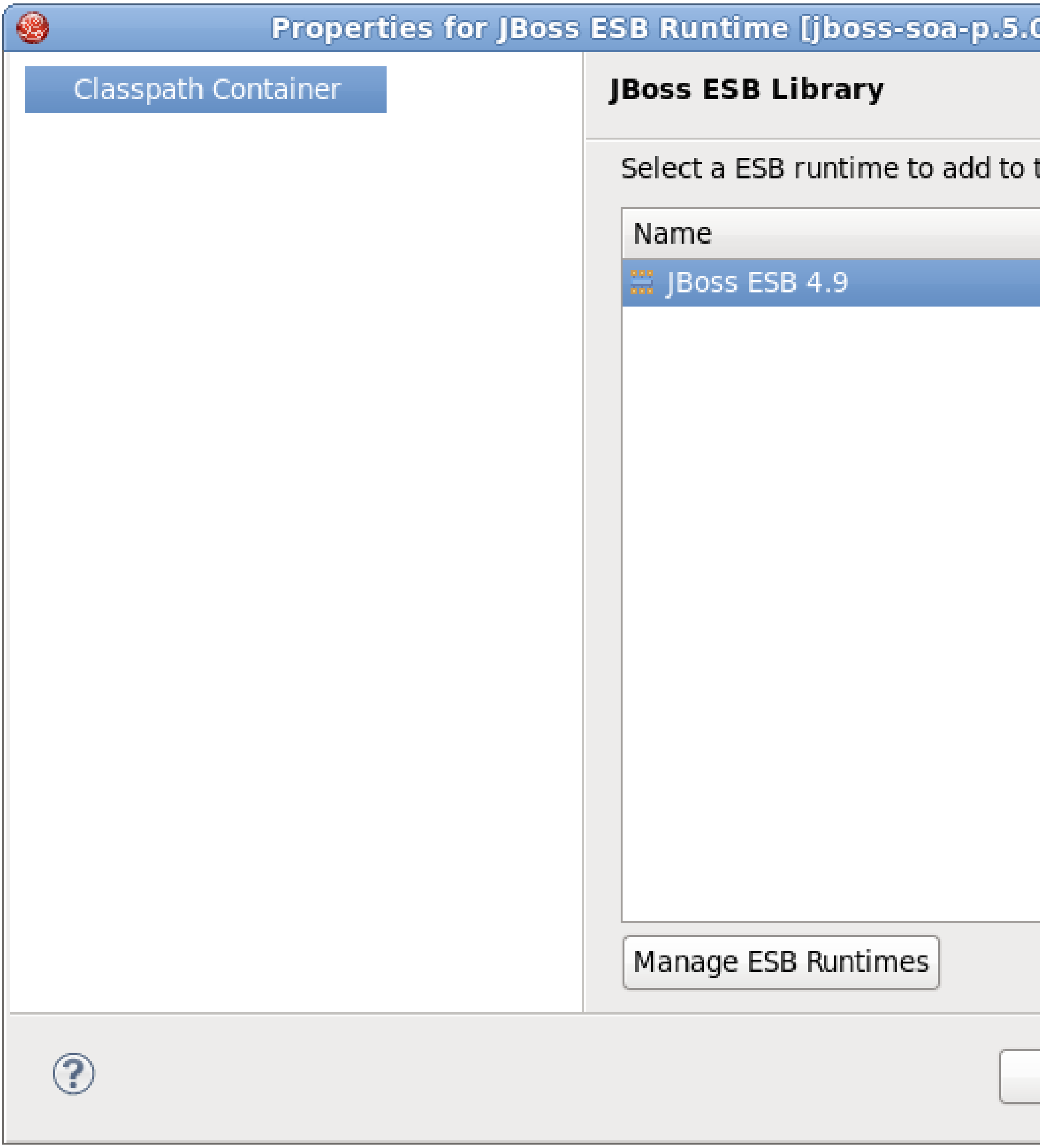
Configuration: default Only for JBoss AS contained ESB run

□ Customize JBoss ESB runtime jars

**Figure 2.22. Configure new JBoss ESB Runtime**

The new JBoss ESB Runtime will be configured. Click **Finish** to finish and save the preferences. You can use the configuration when creating a JBoss ESB project.

When a ESB runtime is configured for your ESB project you are able to change it to any other using the classpath container page for ESB runtime. To do that, turn to the Package Explorer view and right-click the "JBoss ESB Runtime" library. Select *Properties* and a table listing all available JBoss ESB runtimes will appear:



Properties for JBoss ESB Runtime [jboss-soa-p.5.0

JBoss ESB Library

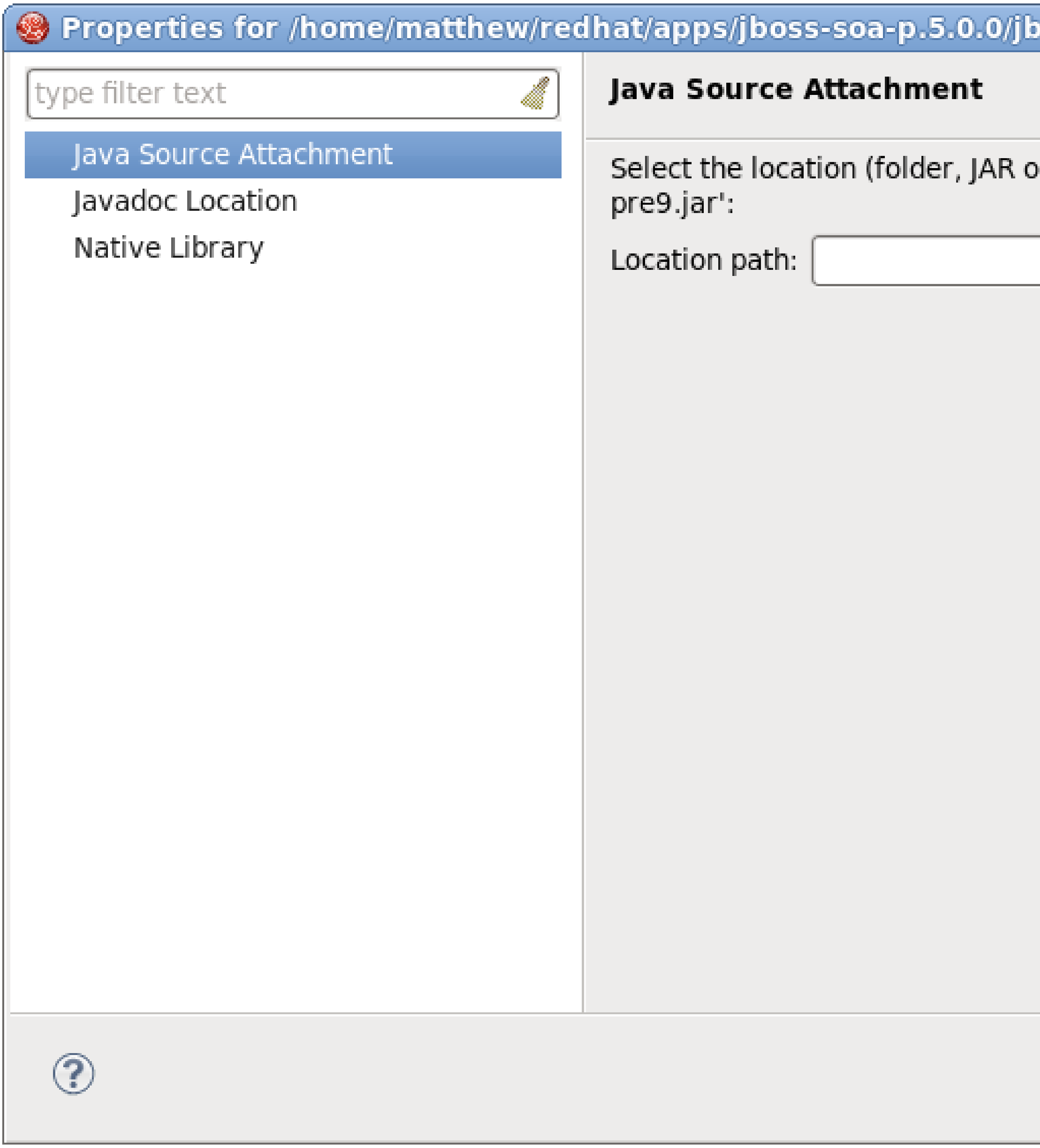
Select a ESB runtime to add to t

| Name |
|---|
| JBoss ESB 4.9 |

**Figure 2.23. Classpath Container Page to change ESB runtime**

Choose one of them to set to the ESB project and click *Ok* .

ESB container allows Source and JavaDoc locations to be set via the Properties dialog on each contained .jar: right-click on any .jar file, select *Properties* . Choose *Java Source Attachment* and select location (folder, JAR or zip) containing new source for the chosen .jar using one of the suggested options (workspace, external folder or file) or enter the path manually:



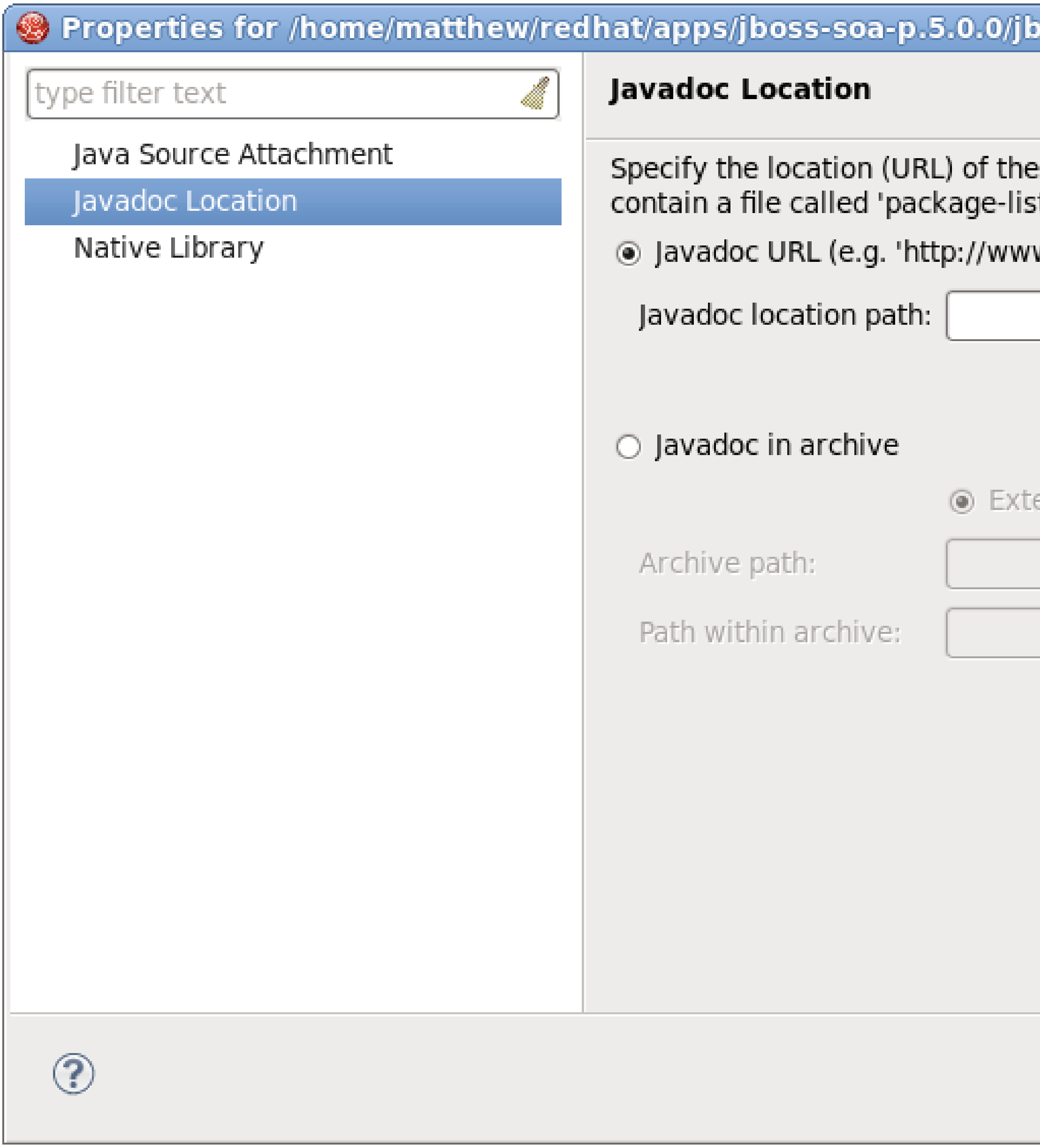
Select the location (folder, JAR o
pre9.jar':

Location path:

**Figure 2.24. Classpath Container: Java Source Attachment**

Click on *Apply* and then on *Ok* .

To change Javadoc Location choose *Javadoc Location* and specify URL to the documentation generated by Javadoc. The Javadoc location will contain a file called *package-list* :



Properties for /home/matthew/redhat/apps/jboss-soa-p.5.0.0/jb

Javadoc Location

Specify the location (URL) of the
contain a file called 'package-lis

● Javadoc URL (e.g. 'http://www

Javadoc location path:

○ Javadoc in archive

● Exte

Archive path:

Path within archive:

**Figure 2.25. Classpath Container: Javadoc Location**

Click on *Apply* and then on *Ok* .

## 2.7. Using and Configuring SOA Platform

In this chapter you will learn what the JBoss Enterprise SOA Platform is and how you can configure it to use for your ESB projects.

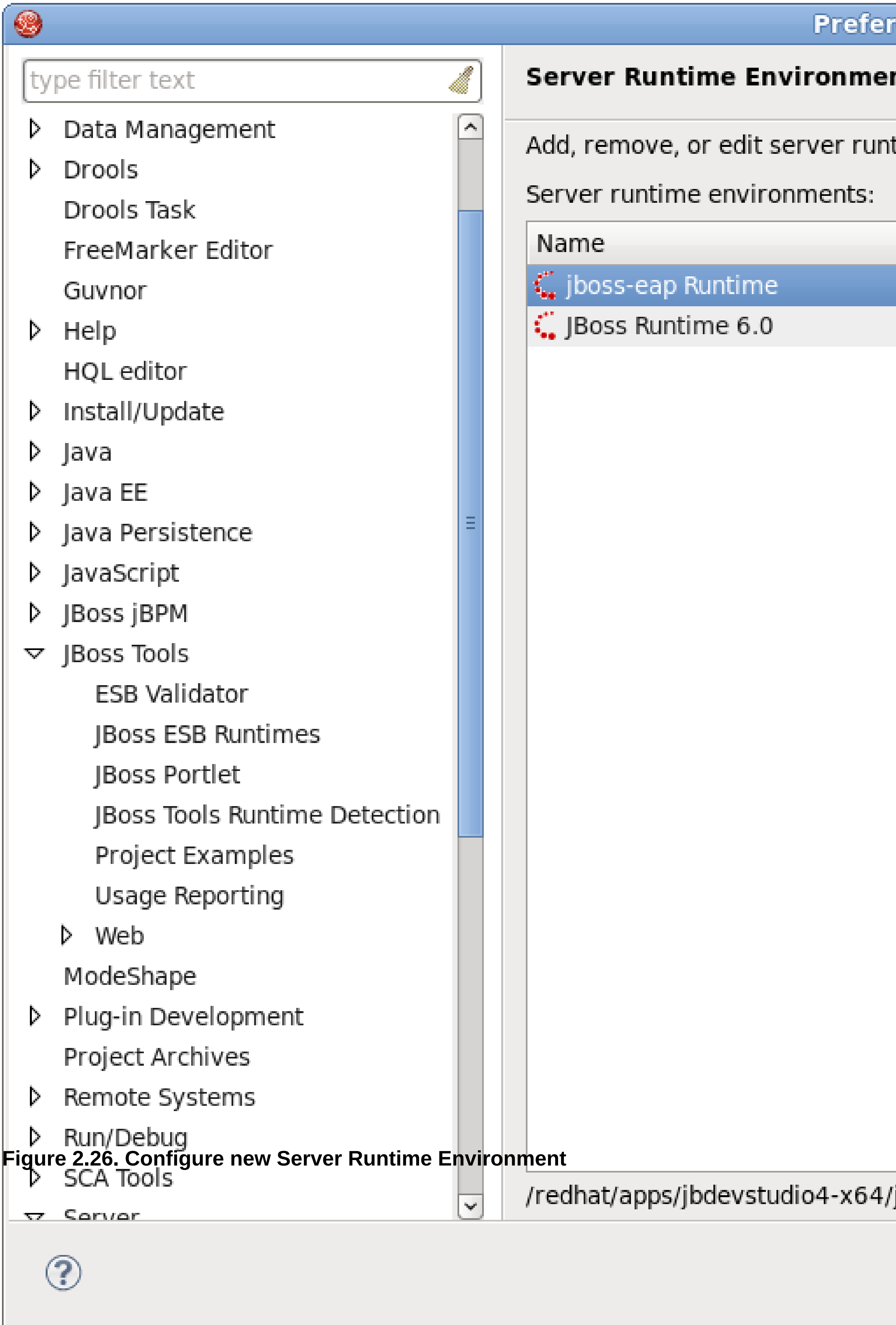
JBoss Enterprise SOA Platform delivers a flexible, standards-based platform to integrate applications, SOA services, business events and automate business processes. The SOA Platform integrates specific versions of JBoss ESB, jBPM, Drools and the JBoss Enterprise Application Platform that are certified to work together in a single supported enterprise distribution.

Having configured JBoss Enterprise SOA Platform for your ESB project you don't need to install and configure ESB server and runtime as they are already included.

Check here to find more details on the platform: *JBoss Enterprise SOA Platform* [http://www.jboss.com/products/platforms/soa] and *JBoss Enterprise SOA Platform Component Details* [http://www.jboss.com/products/platforms/soa/components].

You can find out what is SOA here: *Basics of SOA* [http://www.jboss.org/jbossesb/resources/SOABasics.html] and *SOA and EOA* [http://www.jboss.org/jbossesb/resources/SOAEOA.html].

To configure the JBoss Enterprise SOA platform select **Window** → **Preferences** → **Server** → **Runtime Environments**, which will open the Server Runtime Environments Preferences page where you can add, remove and edit a Server Runtime Environment.



Server Runtime Environmen

Add, remove, or edit server runt

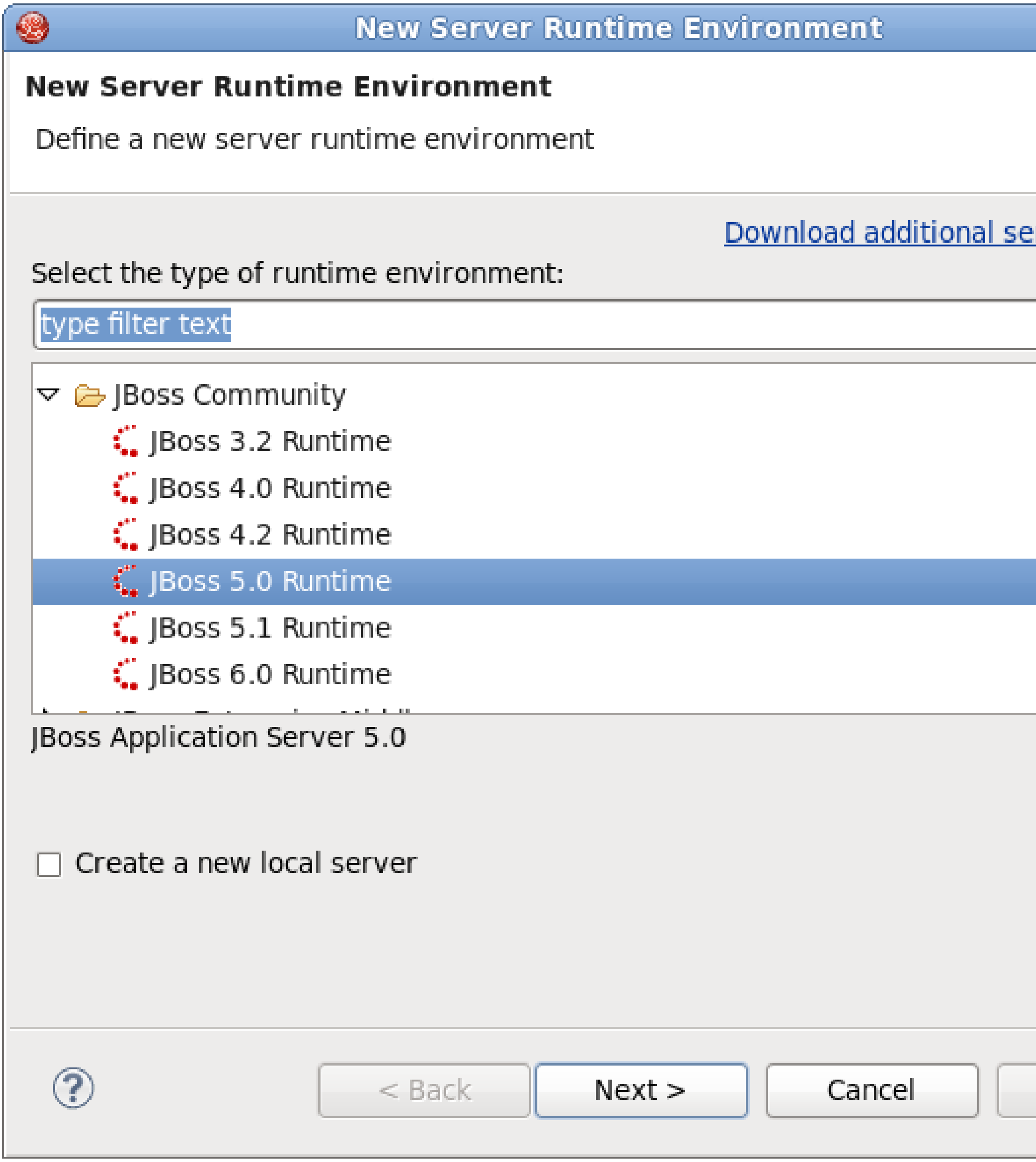
Server runtime environments:

| Name |
|---|
| jboss-eap Runtime |
| JBoss Runtime 6.0 |

/redhat/apps/jbdevstudio4-x64/j

Figure 2.26. Configure new Server Runtime Environment

Select *Add* , choose *JBoss 5.0 Runtime* as a type of runtime environment, check the *Create a new local server* checkbox and click *Next* :



New Server Runtime Environment

Define a new server runtime environment

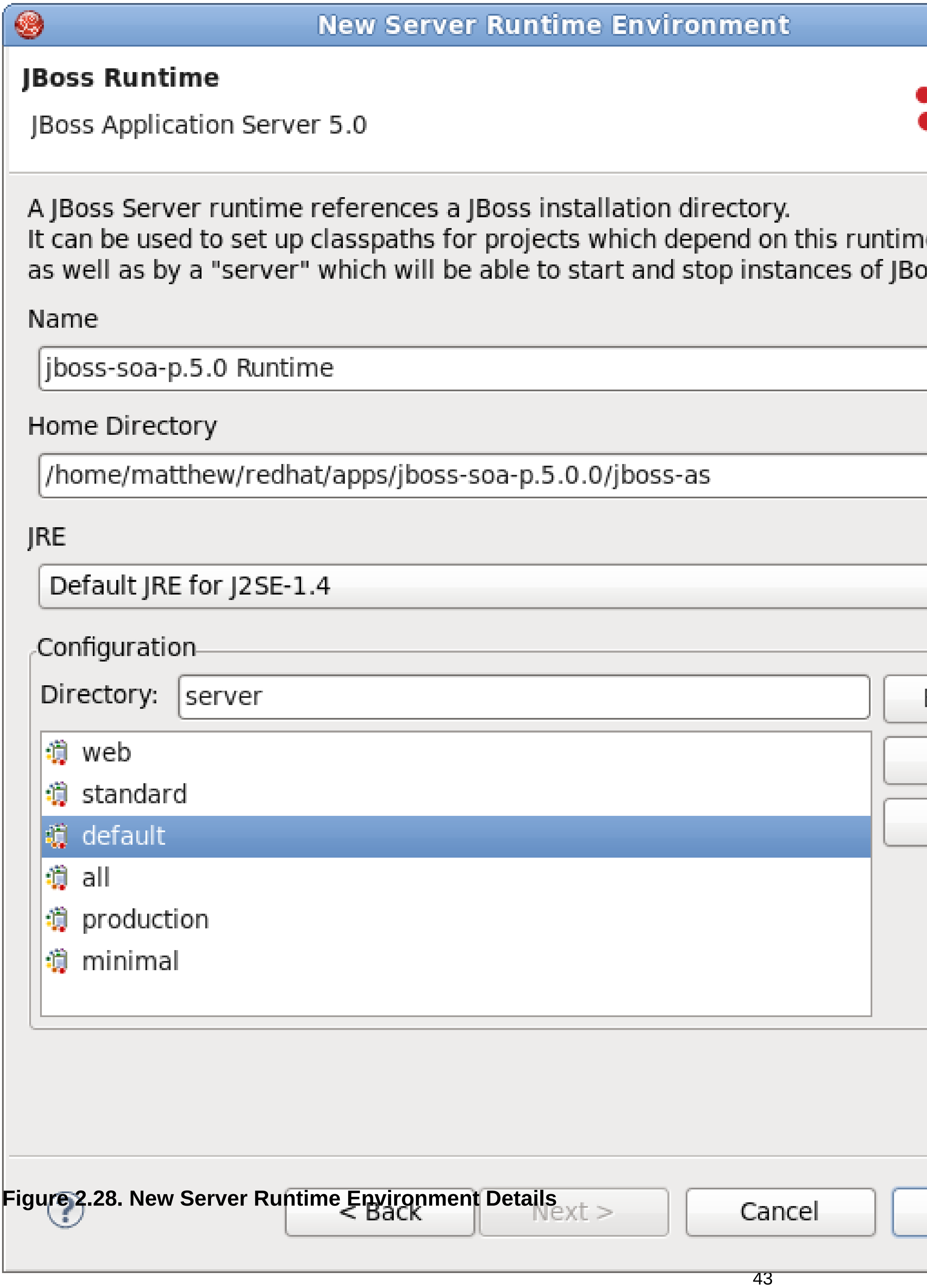
Select the type of runtime environment:

JBoss Application Server 5.0

☐ Create a new local server

**Figure 2.27. Type of Server Runtime Environment**

On the next step you can specify a name of the server runtime environment and browse to its location. Click *Finish* to add the server runtime environment.



**JBoss Runtime**

JBoss Application Server 5.0

A JBoss Server runtime references a JBoss installation directory.
It can be used to set up classpaths for projects which depend on this runtim
as well as by a "server" which will be able to start and stop instances of JBo

Name: jboss-soa-p.5.0 Runtime

Home Directory: /home/matthew/redhat/apps/jboss-soa-p.5.0.0/jboss-as

JRE: Default JRE for J2SE-1.4

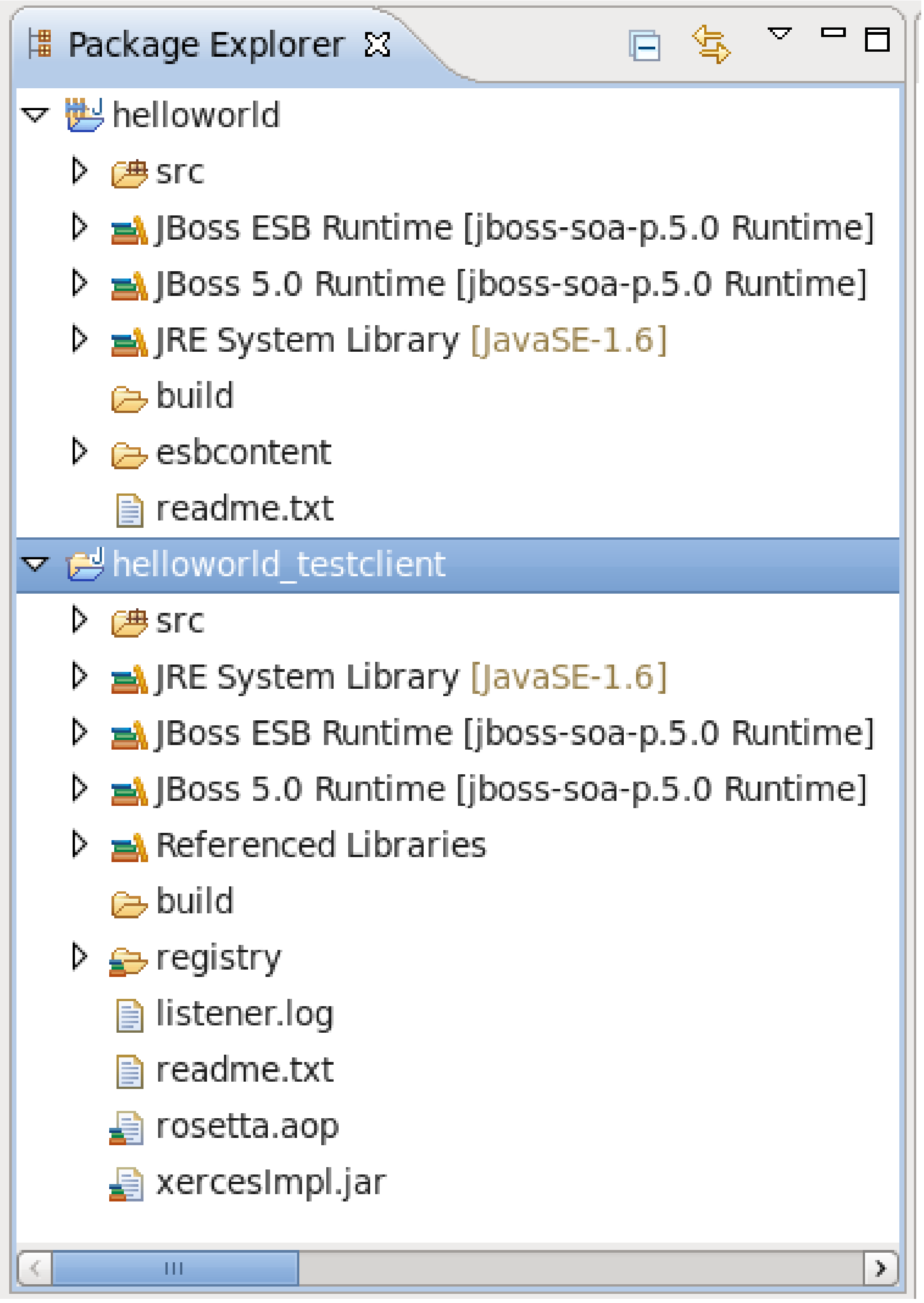
Configuration

Directory: server

- web
- standard
- default
- all
- production
- minimal

Figure 2.28. New Server Runtime Environment Details

Now you have your SOA platform configured. To check the configuration create a ESB Project using instructions described *Section 2.2, “Using ESB Project Examples”*. As a result you will have two projects created:



```
helloworld
    src
    JBoss ESB Runtime [jboss-soa-p.5.0 Runtime]
    JBoss 5.0 Runtime [jboss-soa-p.5.0 Runtime]
    JRE System Library [JavaSE-1.6]
    build
    esbcontent
    readme.txt
helloworld_testclient
    src
    JRE System Library [JavaSE-1.6]
    JBoss ESB Runtime [jboss-soa-p.5.0 Runtime]
    JBoss 5.0 Runtime [jboss-soa-p.5.0 Runtime]
    Referenced Libraries
    build
    registry
    listener.log
    readme.txt
    rosetta.aop
    xercesImpl.jar
```

### Figure 2.29. Helloworld Projects Created

Then you will need to add JBoss ESB libraries to your projects to configure the SOA server runtime exactly for your projects. Right-click on your project, select **Build Path** → **Add Libraries**:

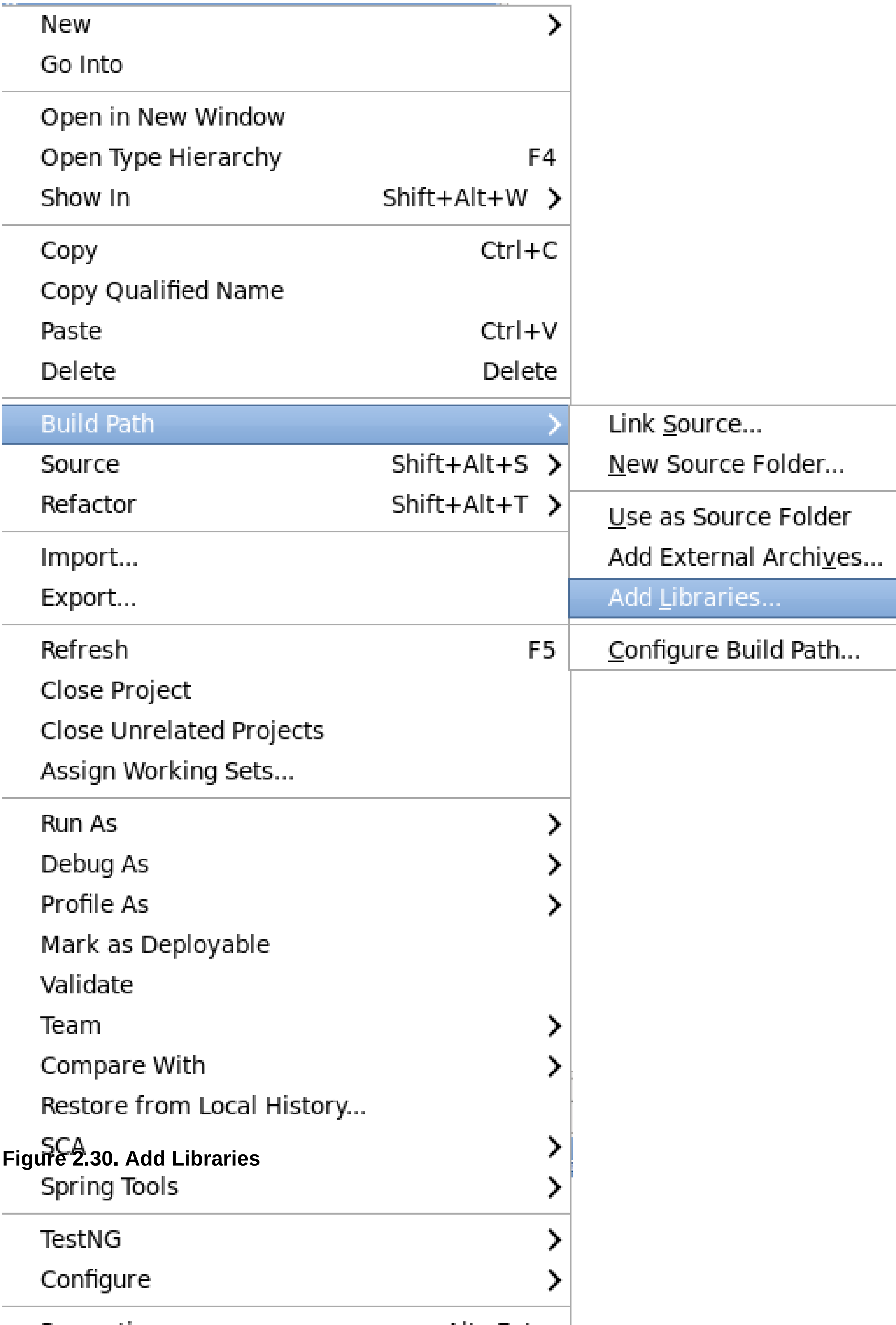


**Figure 2.30. Add Libraries**

Choose *JBoss ESB Libraries* and click *Next* :

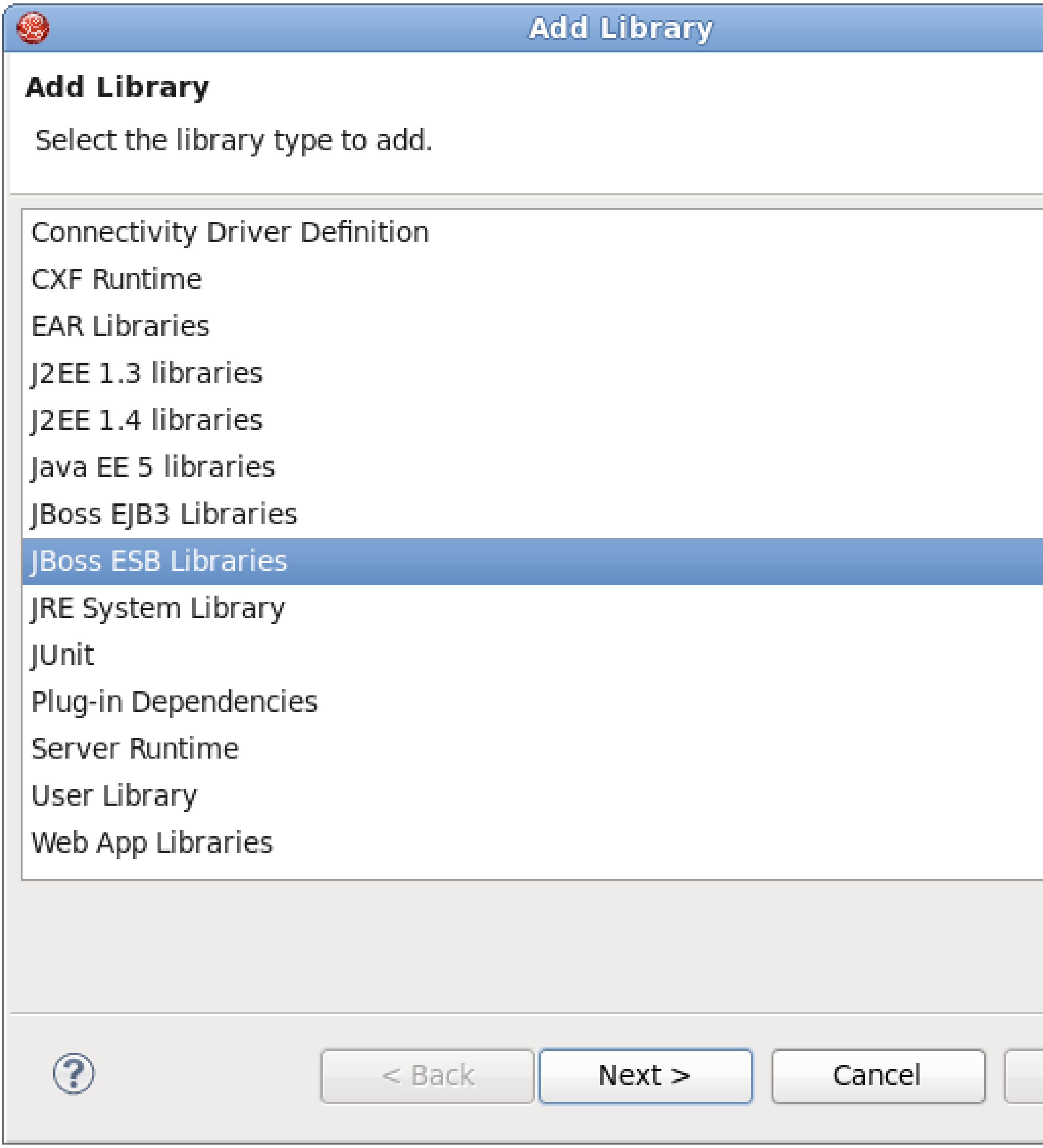
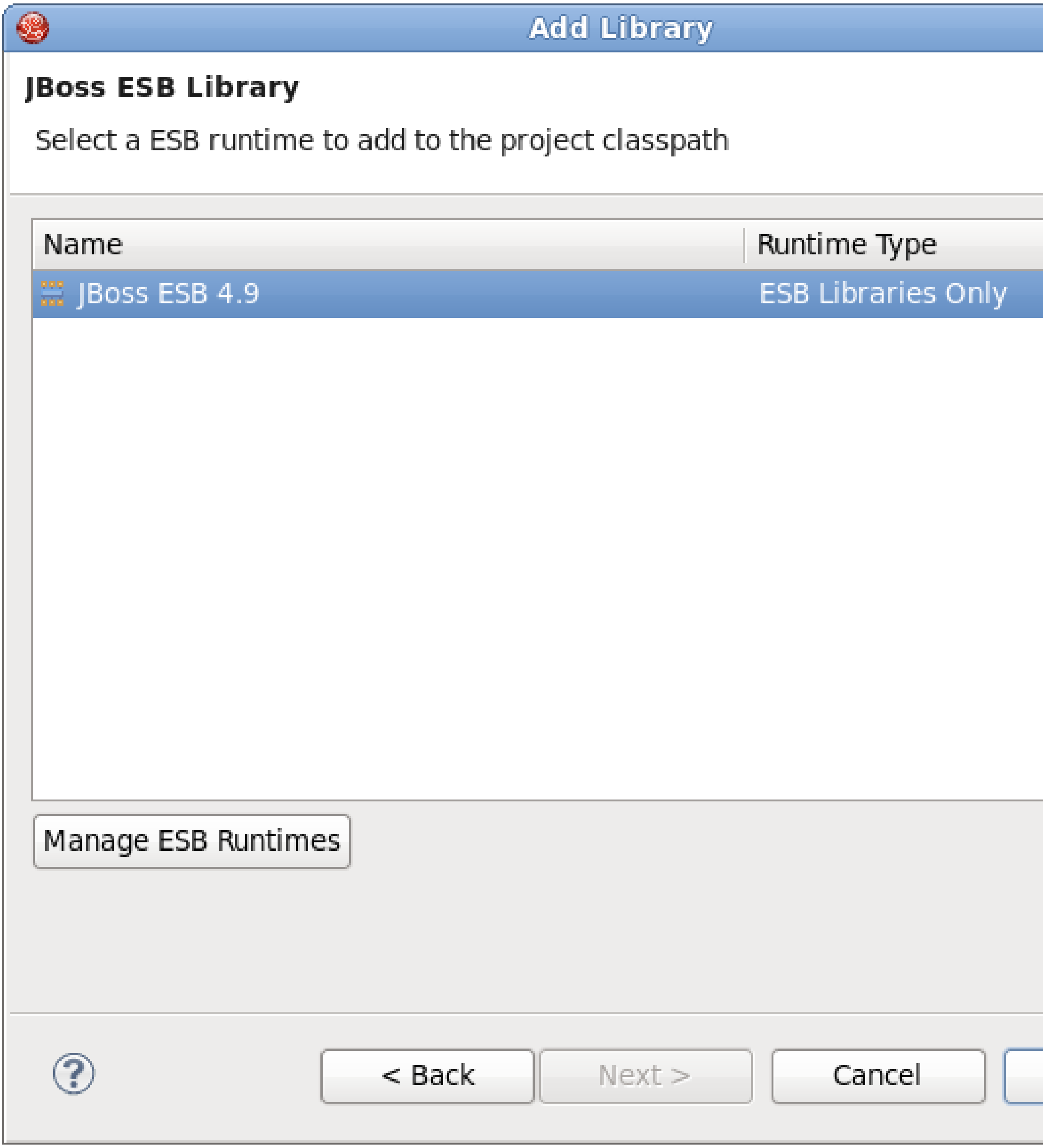


**Figure 2.31. ESB Libraries**

Select the necessary runtime to add to the project classpath:



JBoss ESB Library

Select a ESB runtime to add to the project classpath

| Name | Runtime Type |
|---|---|
| JBoss ESB 4.9 | ESB Libraries Only |

**Figure 2.32. Select a ESB runtime**

Click *Finish* .

Now you can deploy your Helloworld project to the server and run a test class in the client Java project to see the test result in the Console view.

# Reference

This chapter includes detailed reference information about JBoss ESB Tools.

## 3.1. JBoss ESB Editor

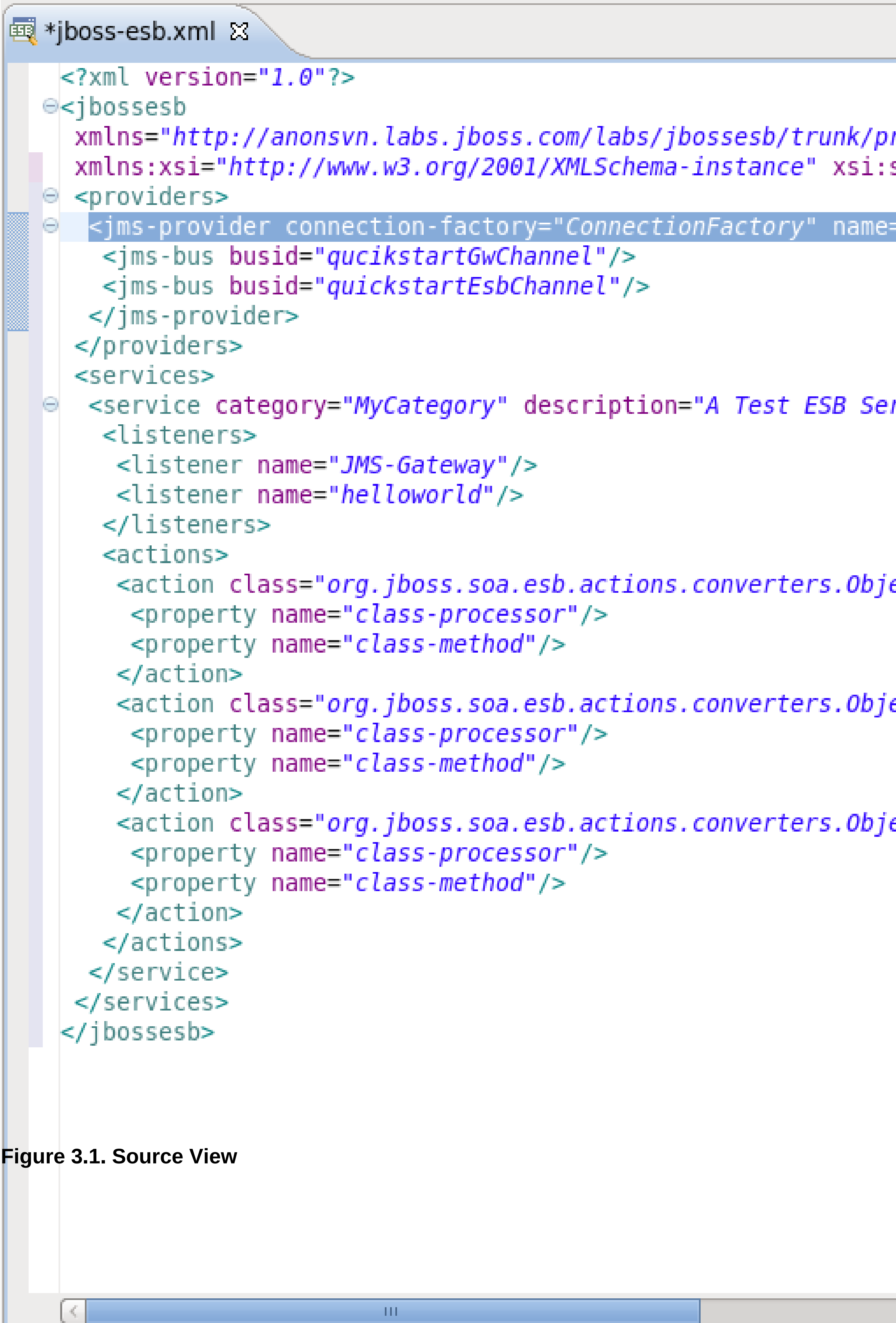
ESB editor has lots of useful features, they are described in details in this chapter.

ESB File Editor is a powerful and customizable tool which allows developing an application using ESB technology.

ESB file editor has two tabs: Tree and Source.

### 3.1.1. Source View

You can easily switch from Tree to Source by selecting the Source tab at the bottom of the editor and work in Source view.

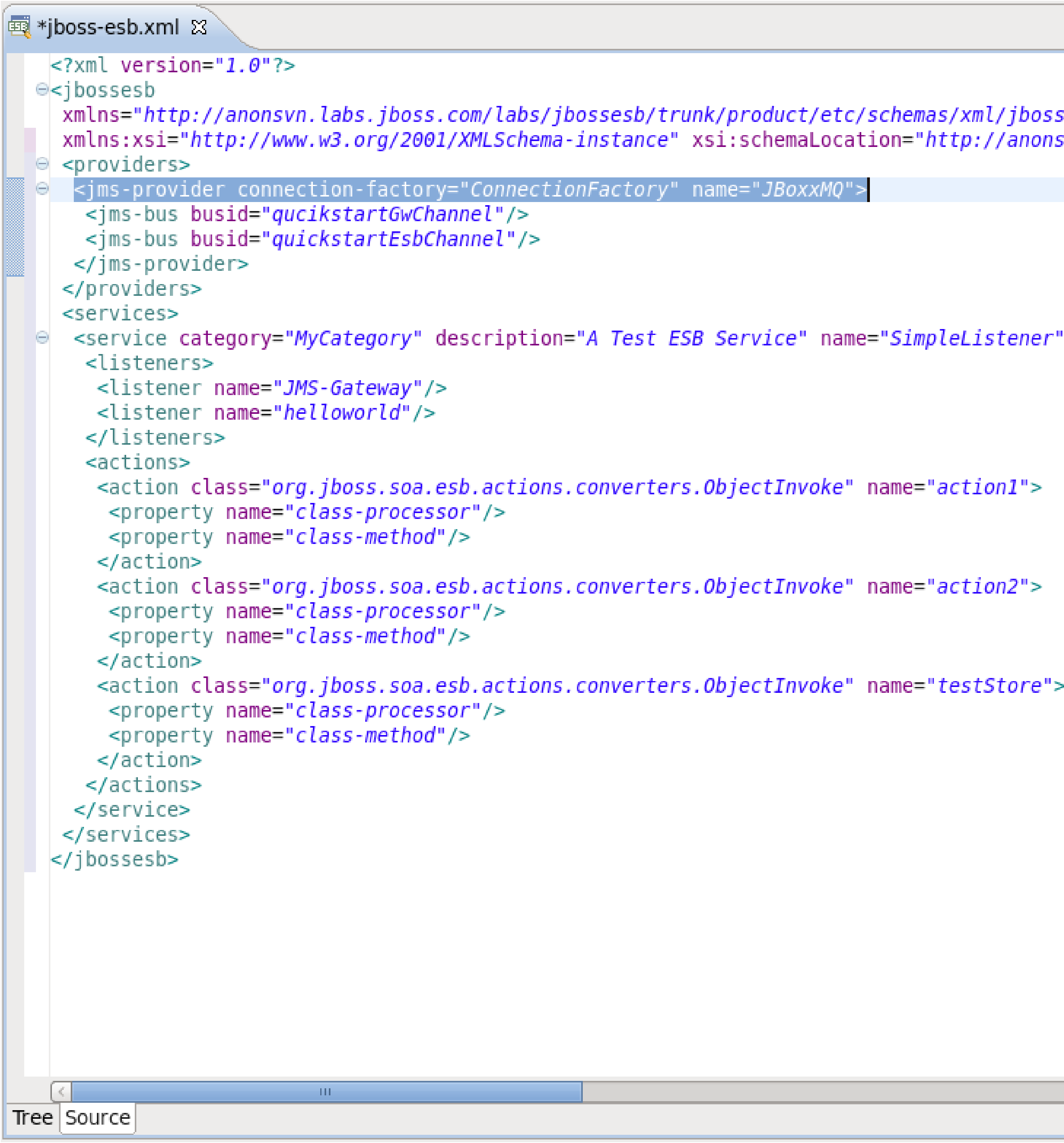


```
<?xml version="1.0"?>
<jbossesb
 xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/pr
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:s
 <providers>
  <jms-provider connection-factory="ConnectionFactory" name=
   <jms-bus busid="quckstartGwChannel"/>
   <jms-bus busid="quickstartEsbChannel"/>
  </jms-provider>
 </providers>
 <services>
  <service category="MyCategory" description="A Test ESB Ser
   <listeners>
    <listener name="JMS-Gateway"/>
    <listener name="helloworld"/>
   </listeners>
   <actions>
    <action class="org.jboss.soa.esb.actions.converters.Obje
     <property name="class-processor"/>
     <property name="class-method"/>
    </action>
    <action class="org.jboss.soa.esb.actions.converters.Obje
     <property name="class-processor"/>
     <property name="class-method"/>
    </action>
    <action class="org.jboss.soa.esb.actions.converters.Obje
     <property name="class-processor"/>
     <property name="class-method"/>
    </action>
   </actions>
  </service>
 </services>
</jbossesb>
```

**Figure 3.1. Source View**

The Source view for the editor displays a text content of the ESB file. It is always synchronized with Tree view, so any changes made in one of the views will immediately appear in the other.

No matter what view you select, you get full integration with Outline view. For example, you can work in the Source view with the help of the Outline view. The Outline view shows a tree structure of the ESB file. Simply select any element in the Outline view and it will jump to the same place in the Source editor, so you can navigate through the source code with Outline view.

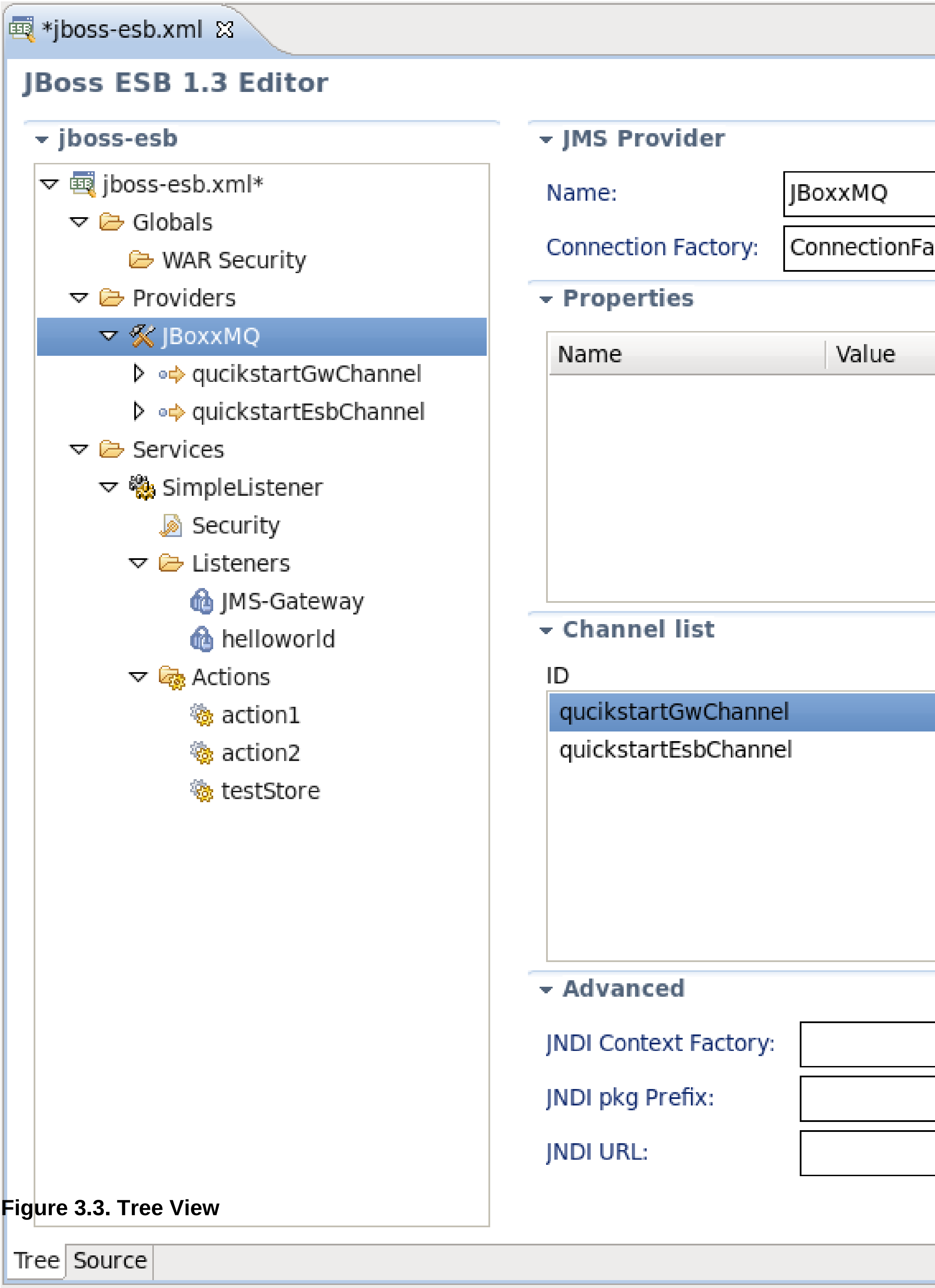


```
<?xml version="1.0"?>
<jbossesb
 xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jboss
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://anons
 <providers>
  <jms-provider connection-factory="ConnectionFactory" name="JBoxxMQ">
   <jms-bus busid="qucikstartGwChannel"/>
   <jms-bus busid="quickstartEsbChannel"/>
  </jms-provider>
 </providers>
 <services>
  <service category="MyCategory" description="A Test ESB Service" name="SimpleListener"
   <listeners>
    <listener name="JMS-Gateway"/>
    <listener name="helloworld"/>
   </listeners>
   <actions>
    <action class="org.jboss.soa.esb.actions.converters.ObjectInvoke" name="action1">
     <property name="class-processor"/>
     <property name="class-method"/>
    </action>
    <action class="org.jboss.soa.esb.actions.converters.ObjectInvoke" name="action2">
     <property name="class-processor"/>
     <property name="class-method"/>
    </action>
    <action class="org.jboss.soa.esb.actions.converters.ObjectInvoke" name="testStore">
     <property name="class-processor"/>
     <property name="class-method"/>
    </action>
   </actions>
  </service>
 </services>
</jbossesb>
```

**Figure 3.2. Outline View**

## 3.1.2. Tree View

You can switch to Tree. The Tree view for the editor displays all ESB artifacts in a tree format. By selecting any node you can see and edit its properties which will appear in the right-hand area. For example, a Provider:



*jboss-esb.xml

JBoss ESB 1.3 Editor

jboss-esb

- jboss-esb.xml*
  - Globals
    - WAR Security
  - Providers
    - JBoxxMQ
      - qucikstartGwChannel
      - quickstartEsbChannel
  - Services
    - SimpleListener
      - Security
      - Listeners
        - JMS-Gateway
        - helloworld
      - Actions
        - action1
        - action2
        - testStore

JMS Provider

Name: JBoxxMQ

Connection Factory: ConnectionFa

Properties

| Name | Value |
|---|---|

Channel list

ID

- qucikstartGwChannel
- quickstartEsbChannel

Advanced

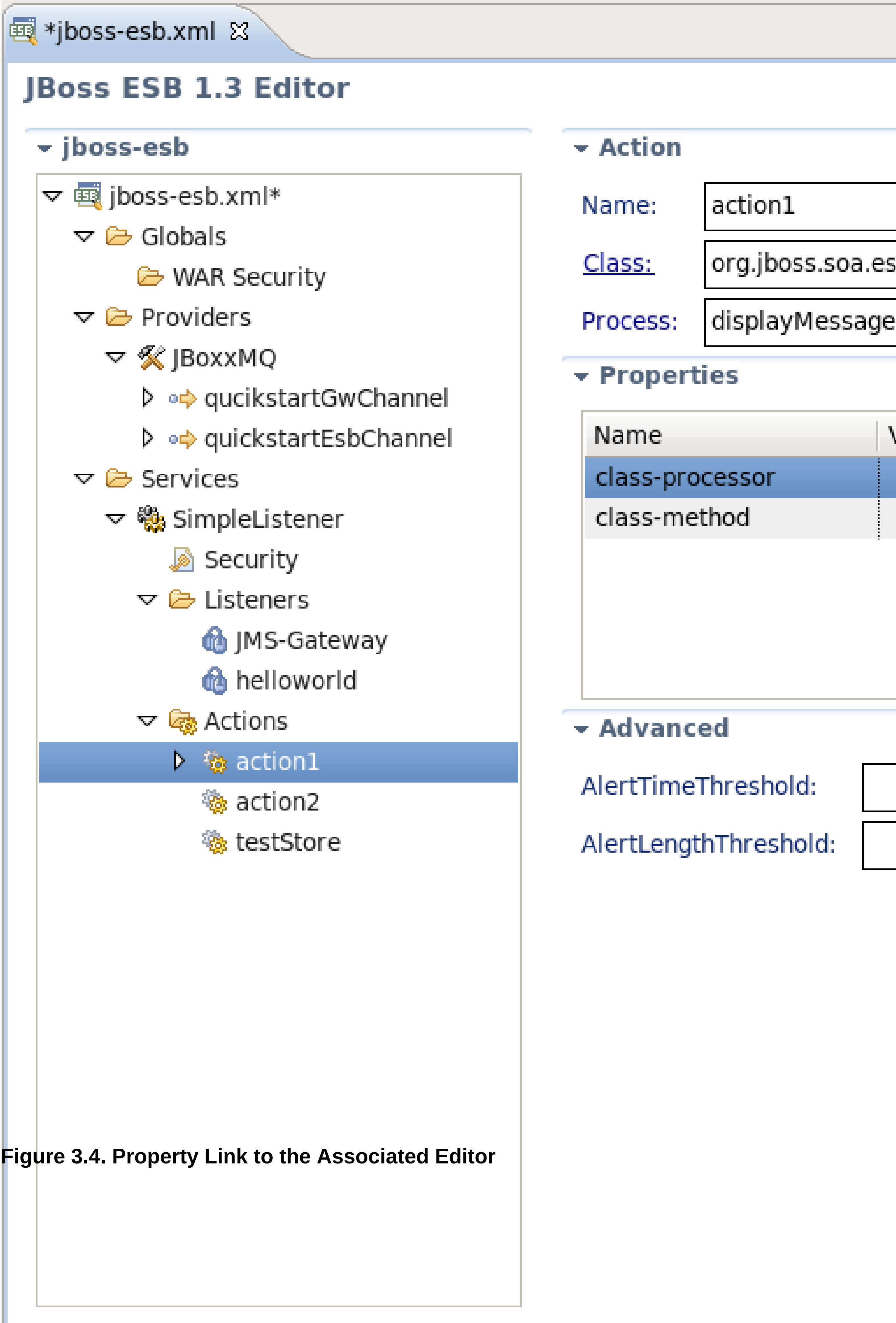
JNDI Context Factory:

JNDI pkg Prefix:

JNDI URL:

Figure 3.3. Tree View

Some properties are represented as links to the associated editors.



*jboss-esb.xml

JBoss ESB 1.3 Editor

jboss-esb

- jboss-esb.xml*
  - Globals
    - WAR Security
  - Providers
    - JBoxxMQ
      - qucikstartGwChannel
      - quickstartEsbChannel
  - Services
    - SimpleListener
      - Security
      - Listeners
        - JMS-Gateway
        - helloworld
      - Actions
        - action1
        - action2
        - testStore

Action

Name: action1

Class: org.jboss.soa.es

Process: displayMessage

Properties

| Name |
|---|
| class-processor |
| class-method |

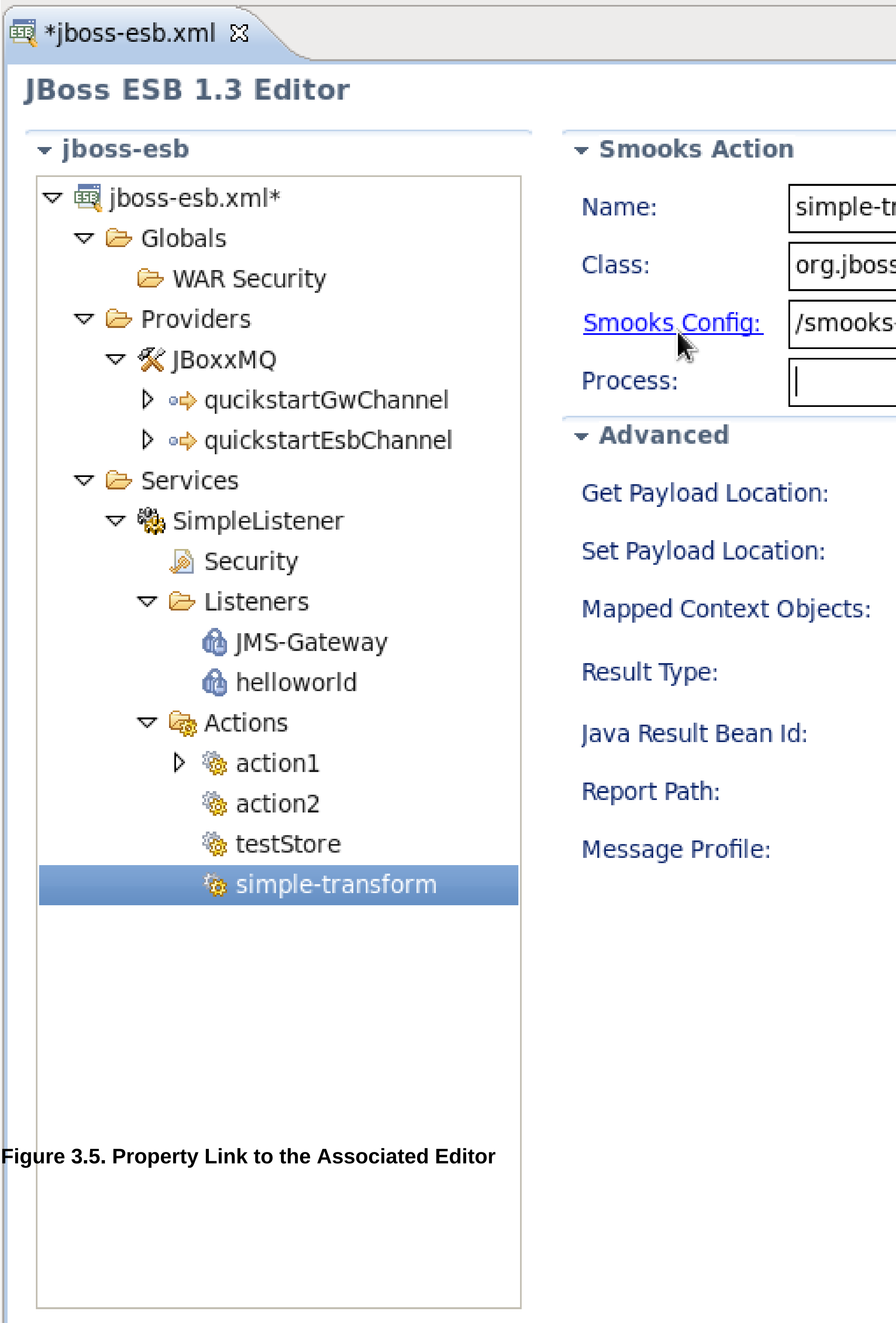
Advanced

AlertTimeThreshold:

AlertLengthThreshold:

Figure 3.4. Property Link to the Associated Editor

Now when editing ESB actions which refer to other files (Drools, Groovy, Smooks, etc.), the label for the field turns into a link to launch the editor associated with that type of file.



*jboss-esb.xml

JBoss ESB 1.3 Editor

jboss-esb

- jboss-esb.xml*
  - Globals
    - WAR Security
  - Providers
    - JBoxxMQ
      - quckstartGwChannel
      - quickstartEsbChannel
  - Services
    - SimpleListener
      - Security
      - Listeners
        - JMS-Gateway
        - helloworld
      - Actions
        - action1
        - action2
        - testStore
        - simple-transform

Smooks Action

Name: simple-tr

Class: org.jboss

Smooks Config: /smooks-

Process:

Advanced

Get Payload Location:

Set Payload Location:

Mapped Context Objects:

Result Type:

Java Result Bean Id:

Report Path:

Message Profile:

Figure 3.5. Property Link to the Associated Editor

Adding, editing or deleting of some artifacts operations are available right in the Tree view. Right-click any node and select one of the available actions in the context menu. For example, you can easily add a new Provider:

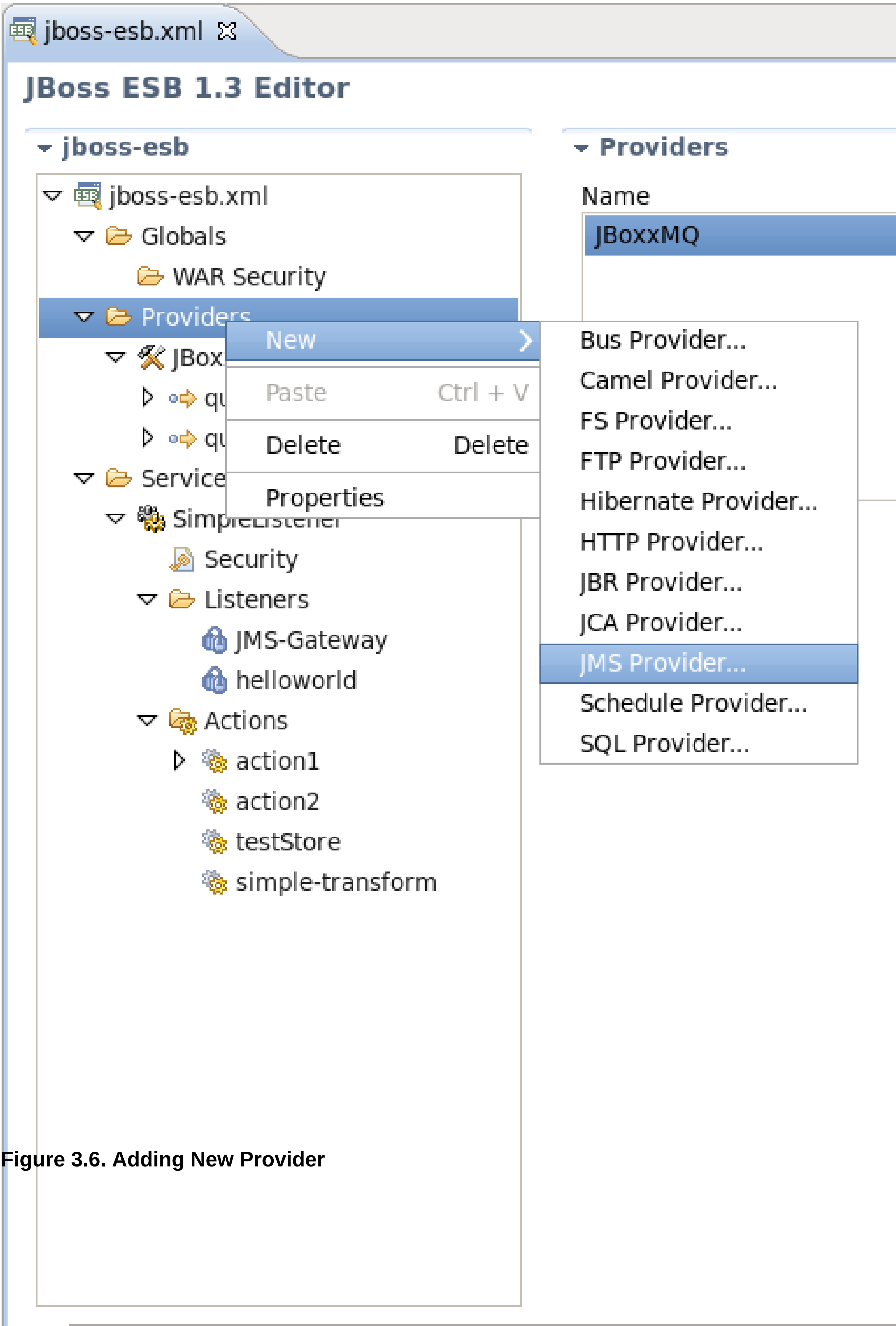


Figure 3.6. Adding New Provider

Then you can add Channels and Properties for the Providers the same way or using the forms with Add, Edit and Remove buttons to the right.

You can easily add a new Service too:

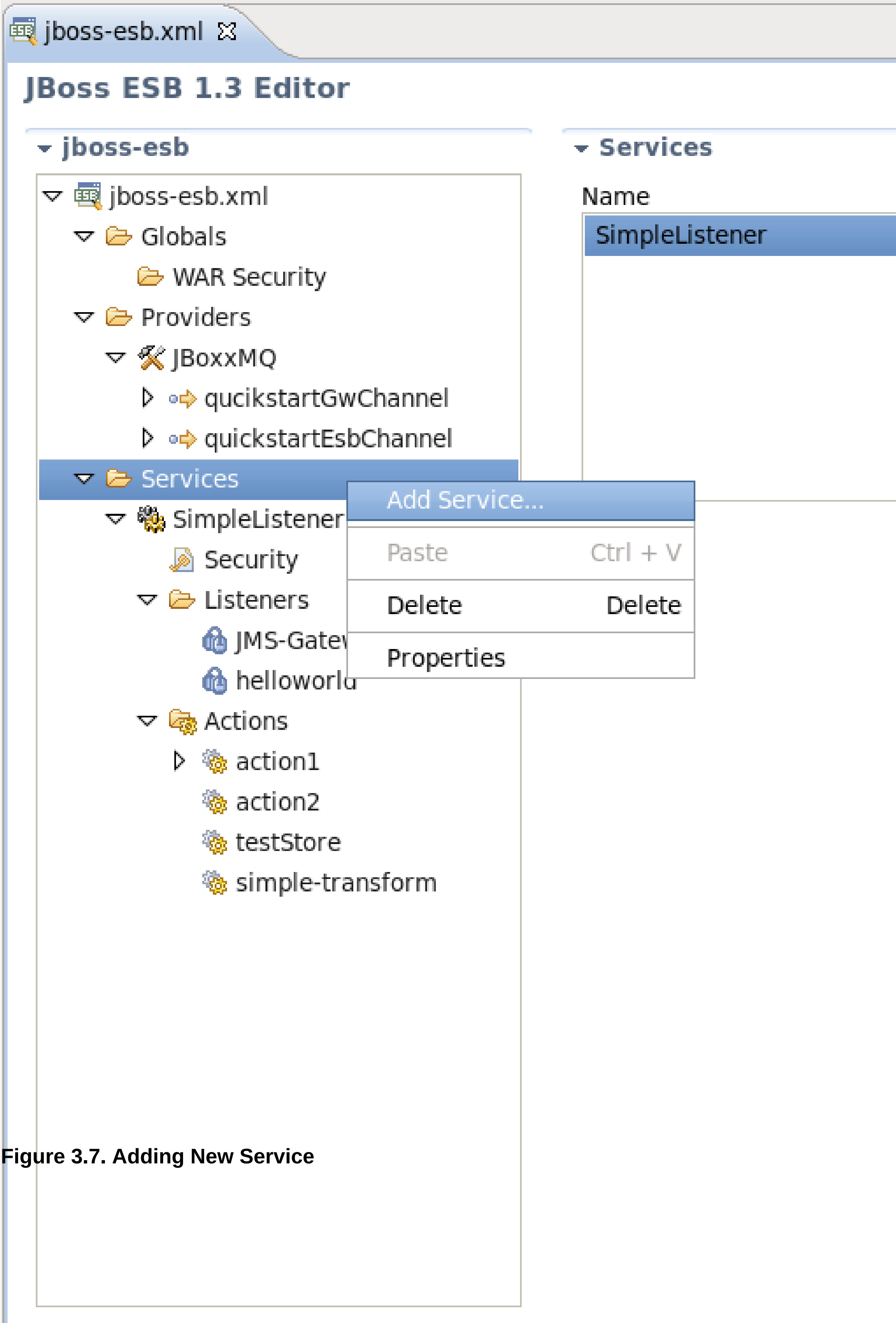


Figure 3.7. Adding New Service

The same way you can create a listener for service and other elements of ESB:

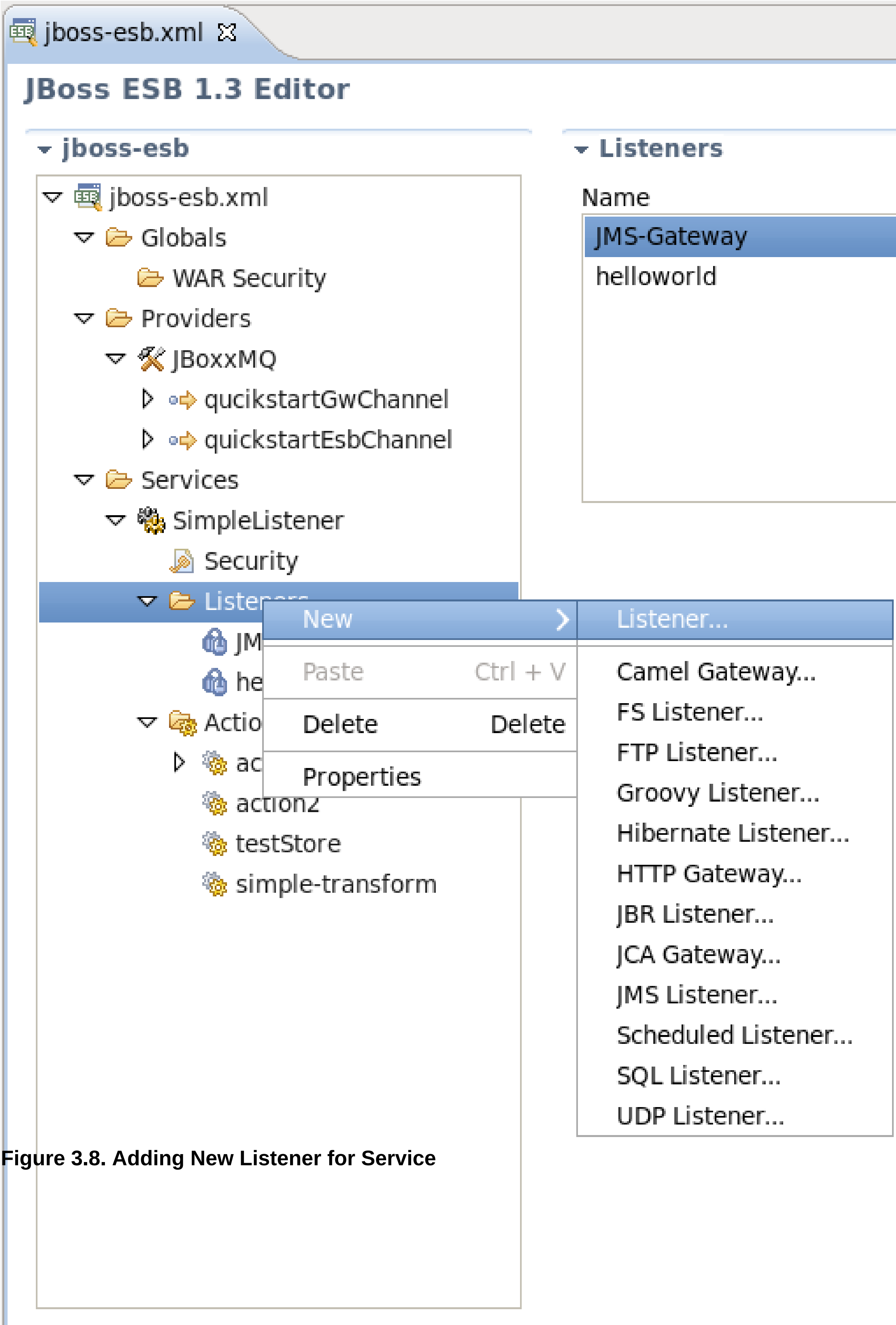


Figure 3.8. Adding New Listener for Service

The same actions can be done in the right part of Tree view tab (Form editor) using Add, Edit and Remove buttons.

Filter can be also edited this way

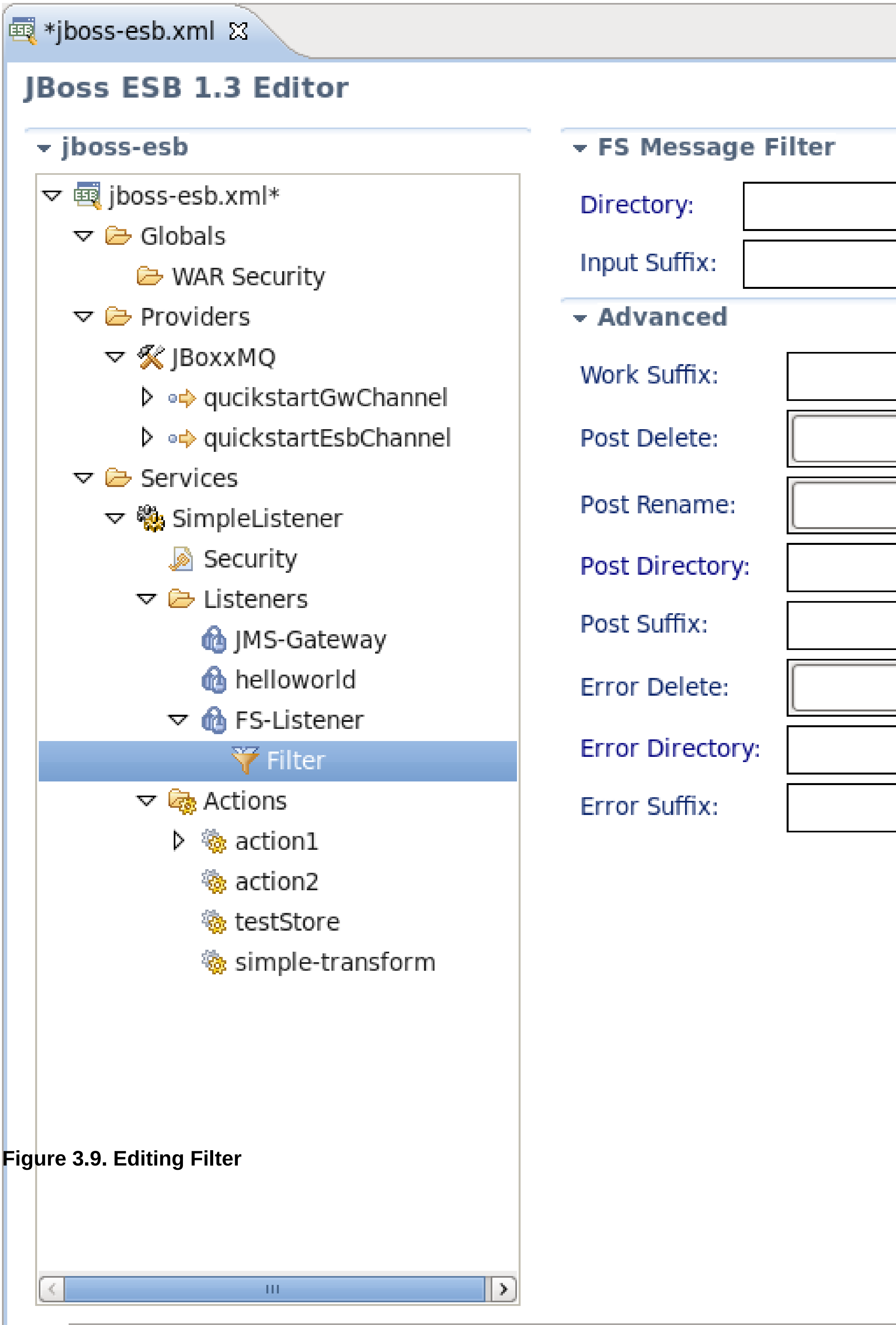


Figure 3.9. Editing Filter

In order to add a new custom Action to your ESB XML file you should select the Actions node under the Services, then right-click and choose **New** → **Custom Action**.

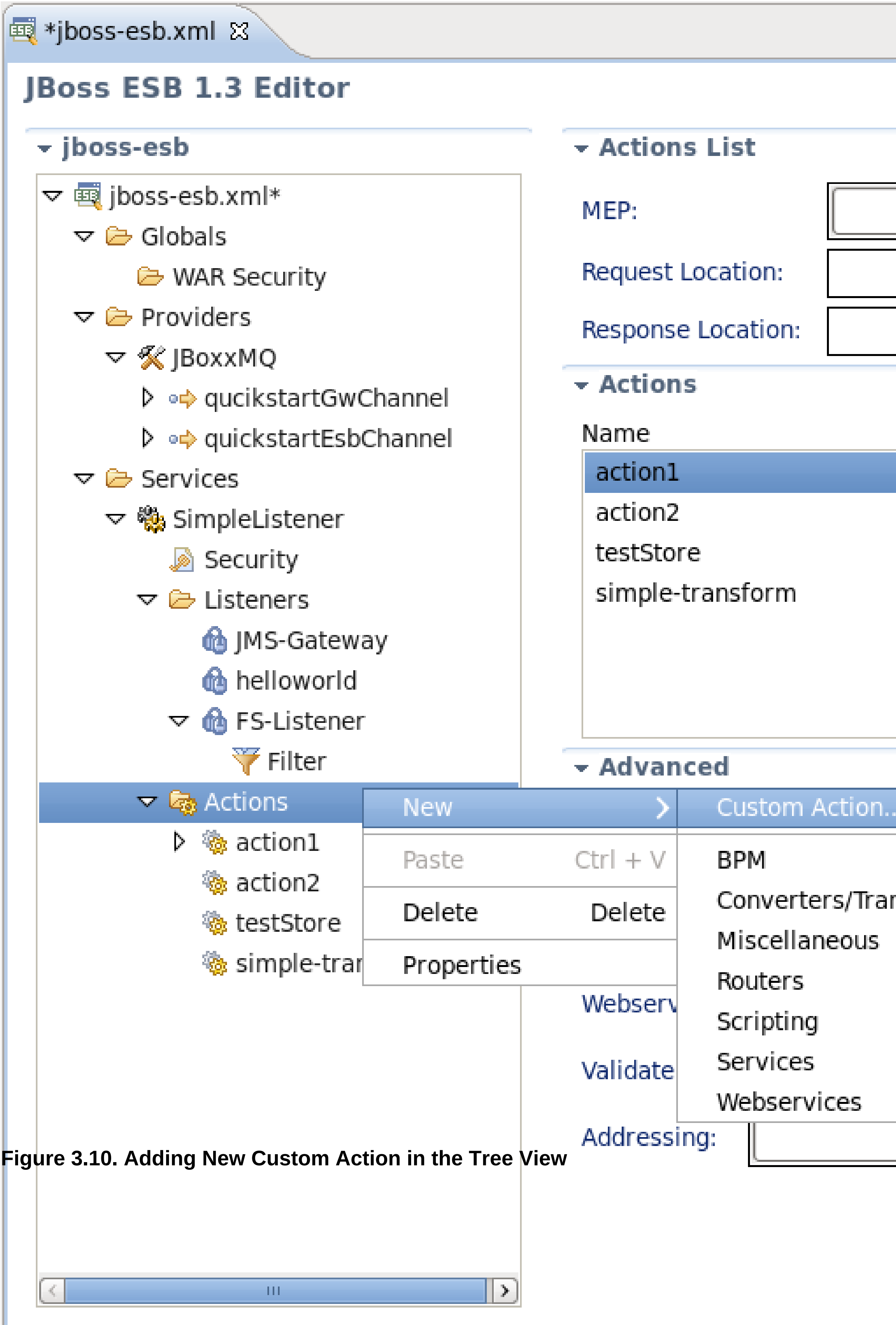


Figure 3.10. Adding New Custom Action in the Tree View

Or instead make use of *Add...* button in the Form editor on the left.

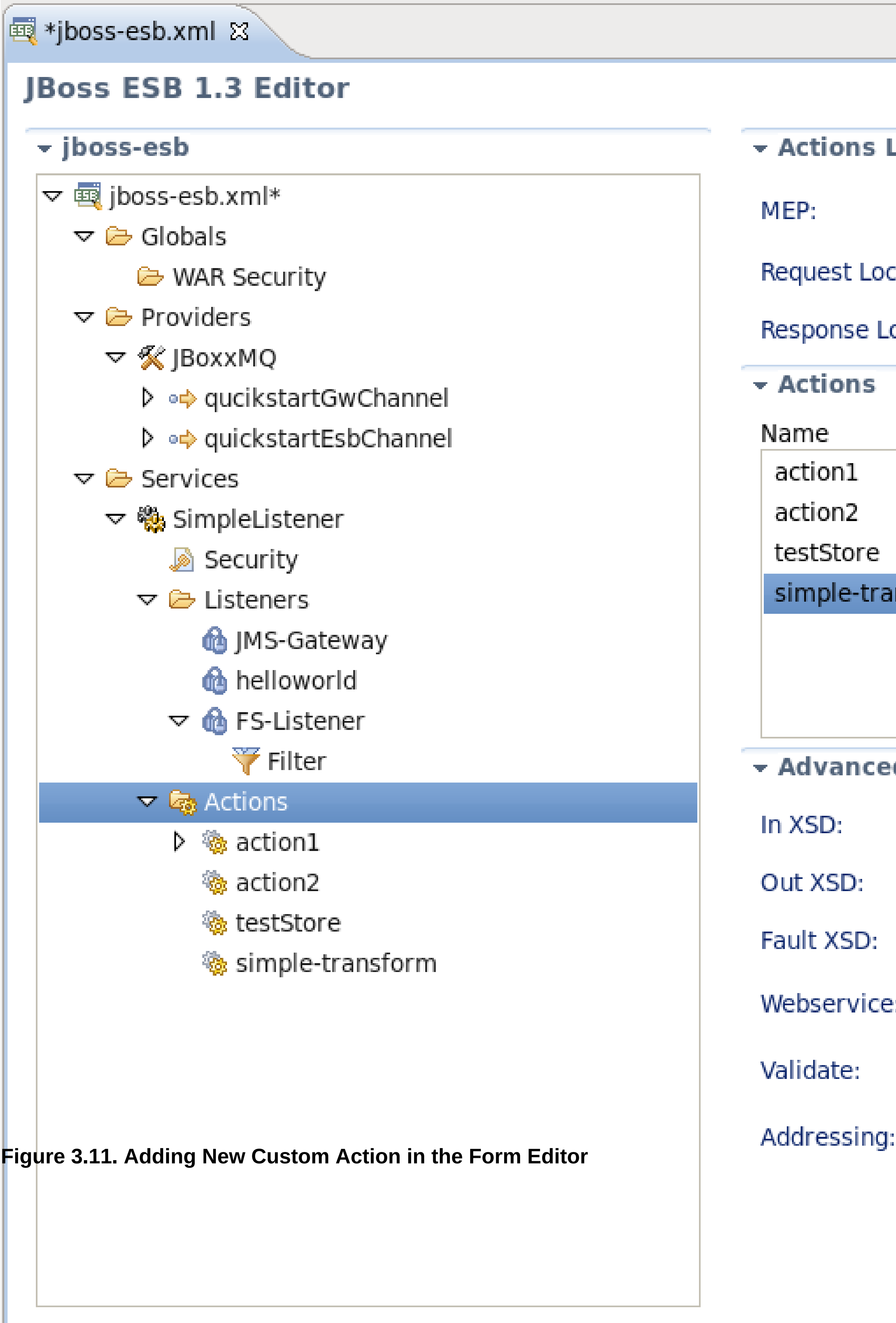
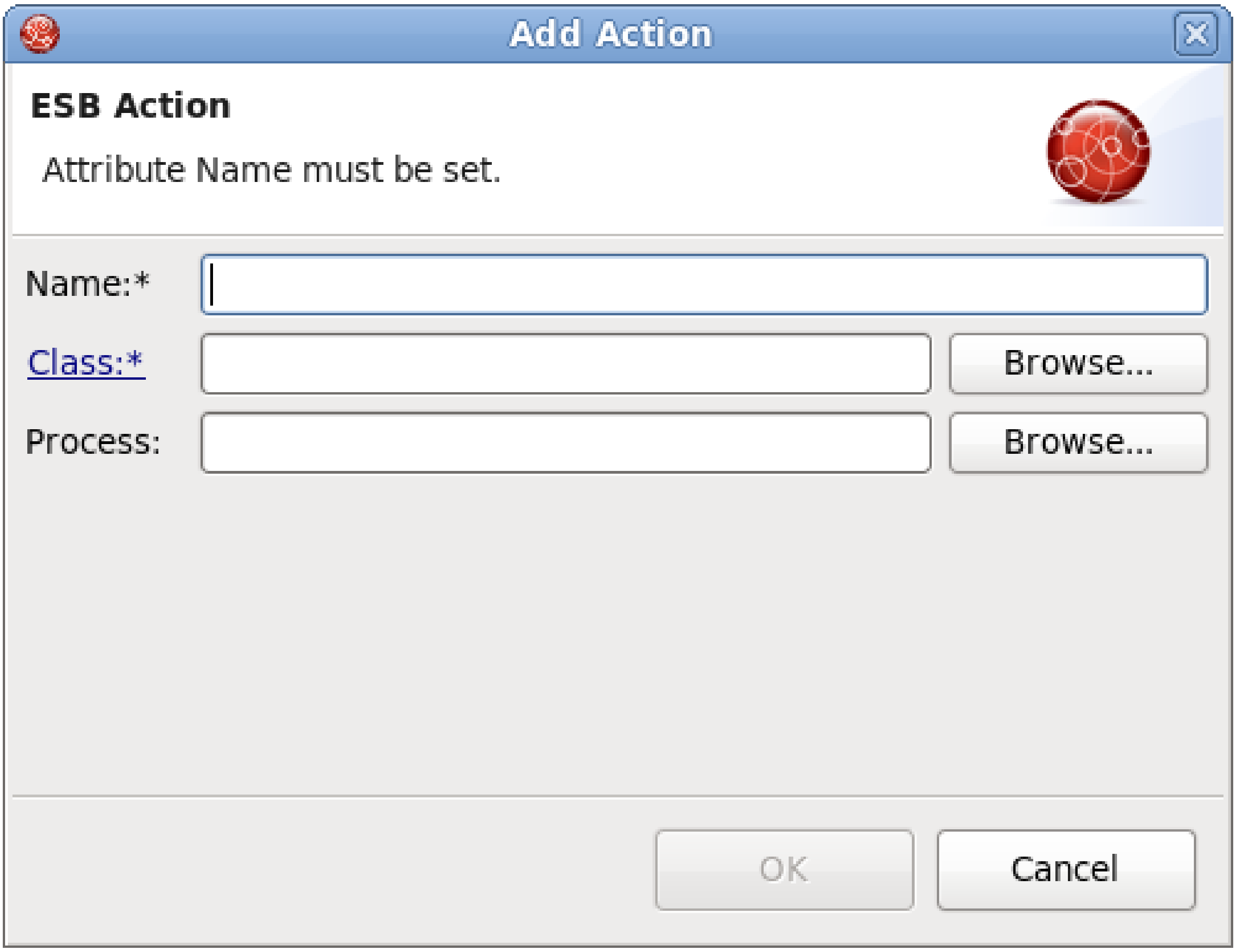


Figure 3.11. Adding New Custom Action in the Form Editor

**Note:**

Some new components are available to support ESB 4.7, such as: new actions (XsltAction, PersistAction, BpmProcessor, ScriptingAction), new processors (EJBProcessor), new routers (HttpRouter, JMSRouter, EmailRouter).

Then you will see Add Action wizard. There is a need to specify Action name and Action Java class.



Add Action

ESB Action

Attribute Name must be set.

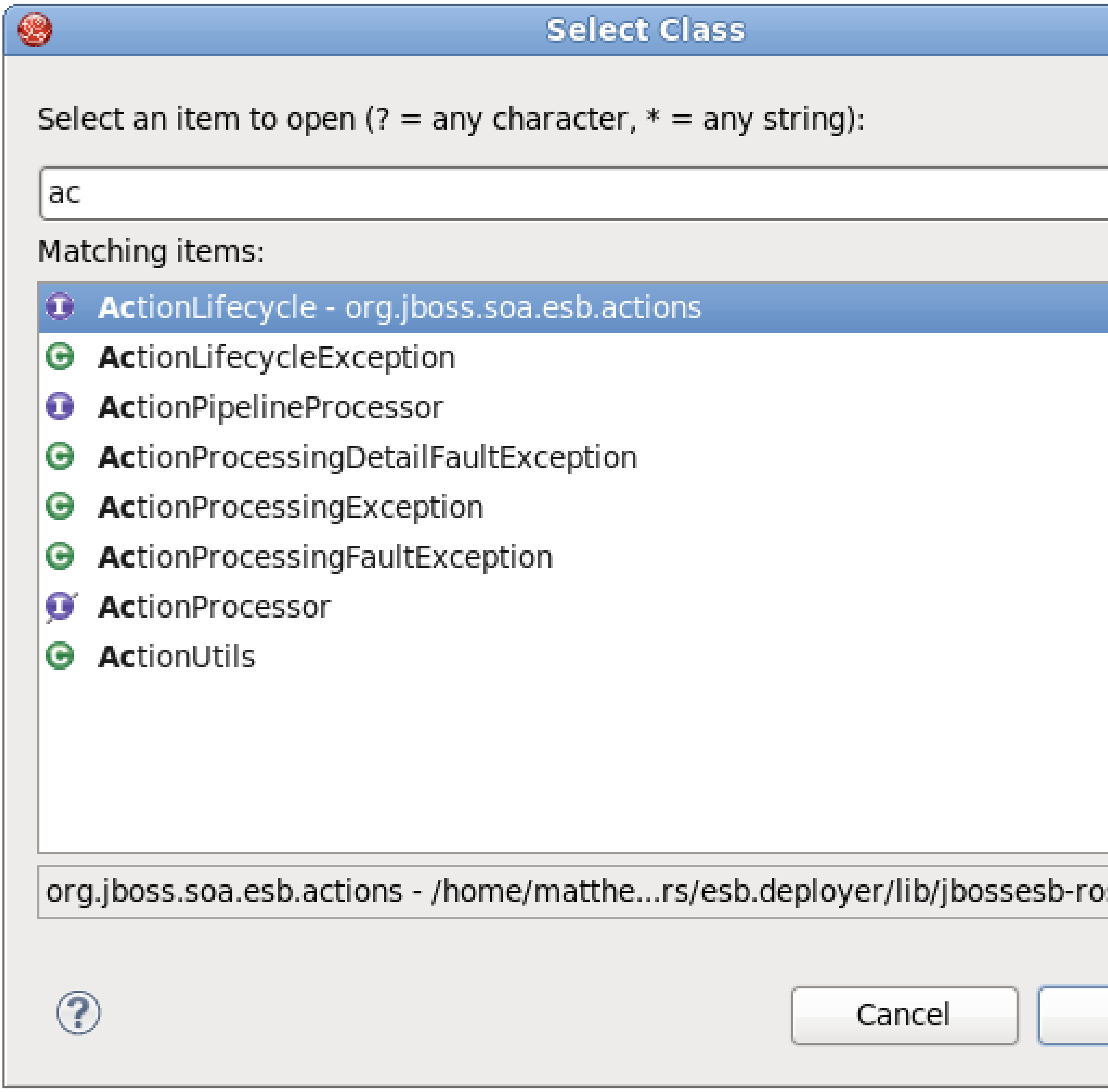
Name:*

Class:*

Process:

**Figure 3.12. Add Action Wizard**

To get a help with finding a proper class you can select *Browse* to open Select class dialog.



Select an item to open (? = any character, * = any string):

ac

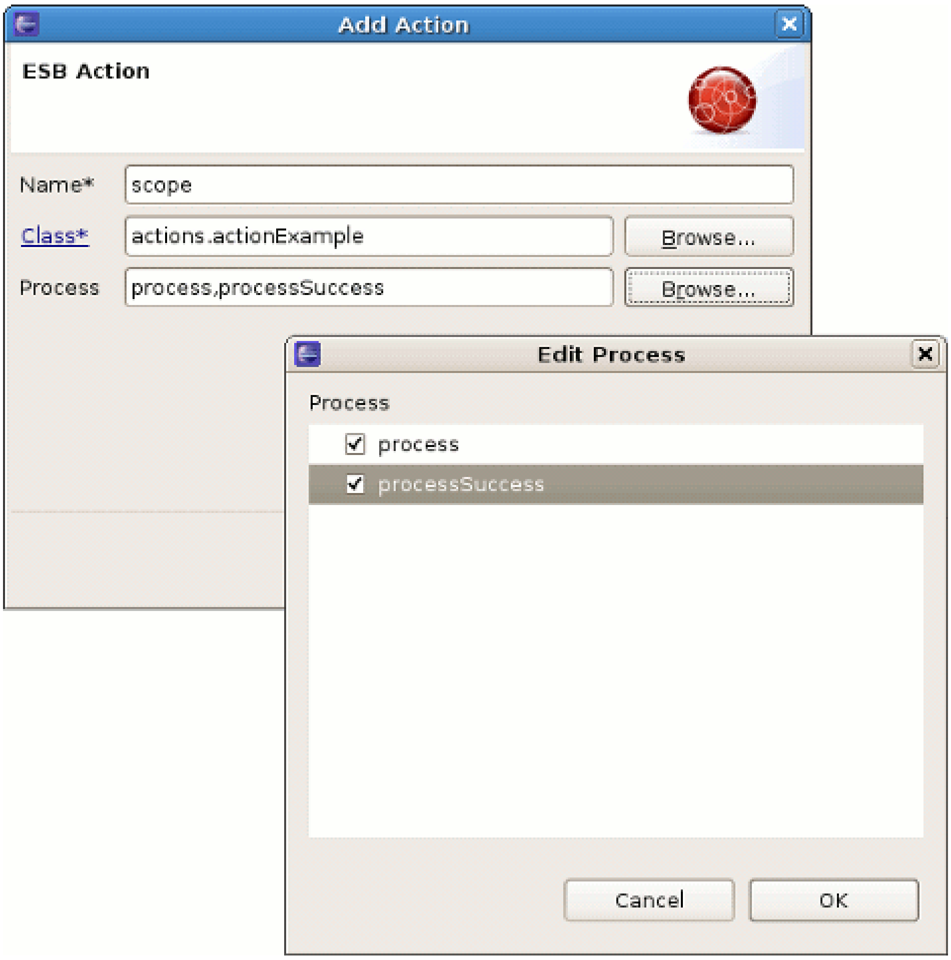
Matching items:

ActionLifecycle - org.jboss.soa.esb.actions
ActionLifecycleException
ActionPipelineProcessor
ActionProcessingDetailFaultException
ActionProcessingException
ActionProcessingFaultException
ActionProcessor
ActionUtils

org.jboss.soa.esb.actions - /home/matthe...rs/esb.deployer/lib/jbossesb-ro

**Figure 3.13. Select class dialog**

Moreover it's possible to type a Process name or select it with Edit Process dialog which is called out by clicking *Browse*.



Name* scope

Class* actions.actionExample

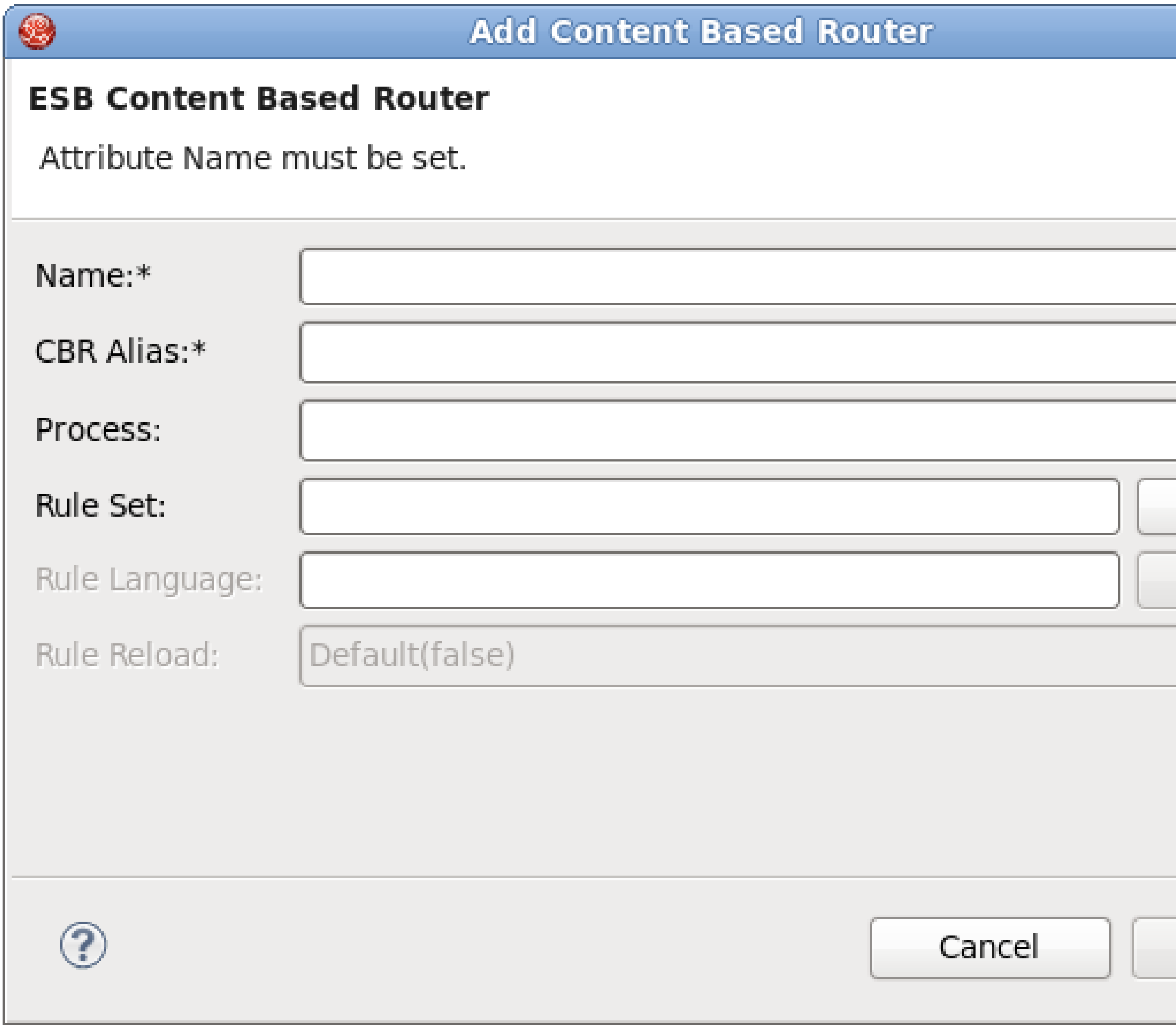
Process process,processSuccess

☑ process

☑ processSuccess

## Figure 3.14. Edit Process dialog

As you can see on the both figures above, the context menu will also prompt you to insert one of the Actions that are supplied out-of-the-box with JBoss ESB. After choosing one an appeared New Pre-Packed Action wizard will ask you to fill out a name field and other fields specific for each Action property. For example, for *Content Based Router* Action the wizard looks as follows:



Add Content Based Router

ESB Content Based Router

Attribute Name must be set.

Name:*

CBR Alias:*

Process:

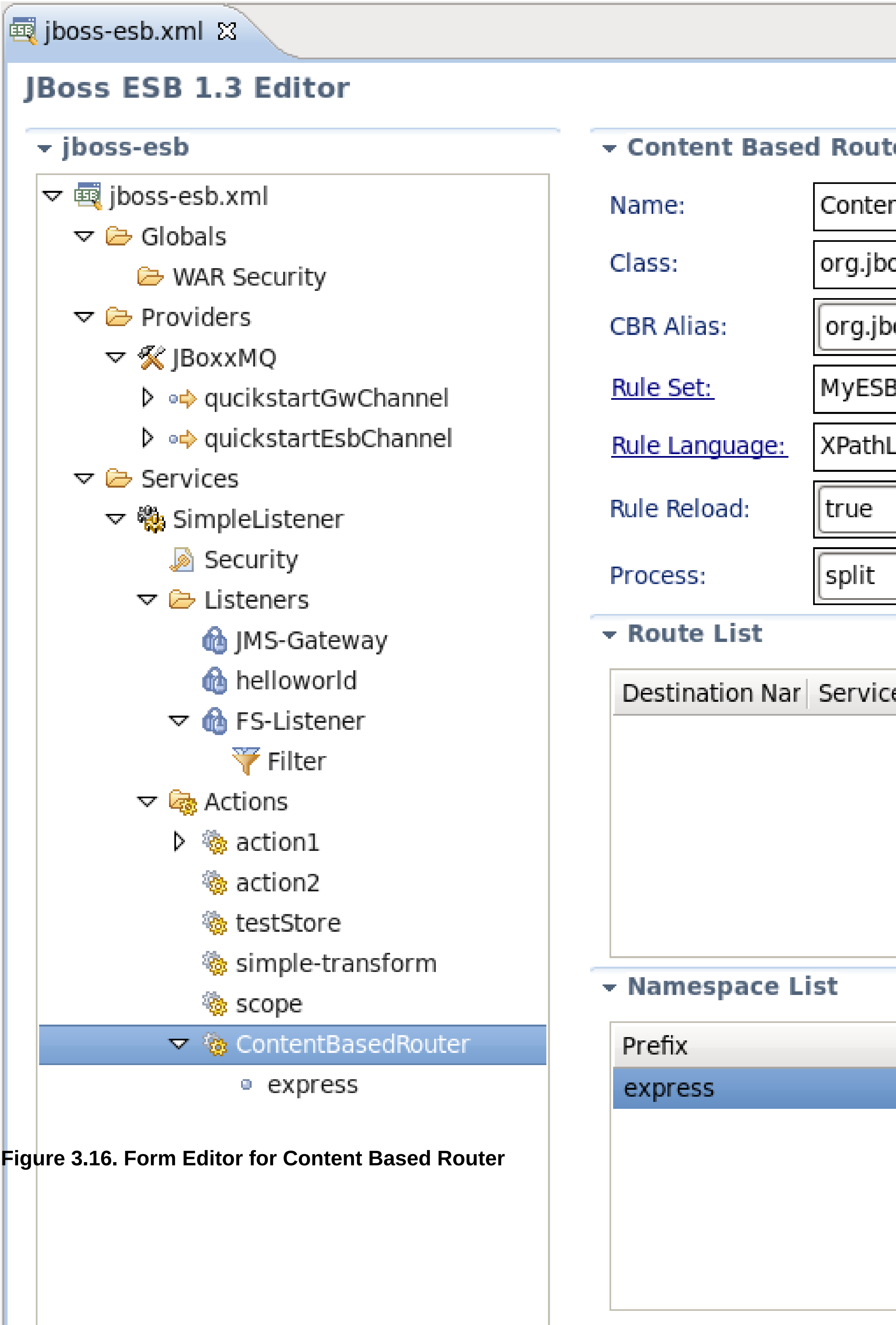
Rule Set:

Rule Language:

Rule Reload: Default(false)

### Figure 3.15. Add Pre-Packed Action Wizard

After confirming creating the Action you can see it in the Tree under the *Actions* node and preview as well as edit its settings in the Form editor on the left.



jboss-esb.xml

JBoss ESB 1.3 Editor

jboss-esb

- jboss-esb.xml
  - Globals
    - WAR Security
  - Providers
    - JBoxxMQ
      - qucikstartGwChannel
      - quickstartEsbChannel
  - Services
    - SimpleListener
      - Security
      - Listeners
        - JMS-Gateway
        - helloworld
        - FS-Listener
          - Filter
      - Actions
        - action1
        - action2
        - testStore
        - simple-transform
        - scope
        - ContentBasedRouter
          - express

Content Based Rout

Name: Conter

Class: org.jbo

CBR Alias: org.jb

Rule Set: MyESB

Rule Language: XPathL

Rule Reload: true

Process: split

Route List

| Destination Nar | Service |
|---|---|

Namespace List

| Prefix |
|---|
| express |

Figure 3.16. Form Editor for Content Based Router

ESB editor can recognize some specific objects. On the figure you can see *org.jboss.soa.esb.actions.ContentBasedRouter* in the *Class* section.

## 3.2. ESB Editor Features

JBoss ESB tooling has powerful editor features that help you easily make use of content and code assist.

This last chapter covers capabilities on how you can use ESB editor.

### 3.2.1. ESB Syntax Validation

When working in JBoss ESB editor you are constantly provided with feedback and contextual error checking as you type. In the Source viewer, if at any point a tag is incorrect or incomplete, an error will be indicated next to the line and also in the Problems view below.

### 3.2.2. Support for XML Schema

JBoss ESB Framework fully supports XML files based on schemas as well as DTDs (see *Figure 3.1, “Source View”*).

The schema checks the child elements of any kind of provider element; the ESB generates errors on startup if you attempt to define an incorrect combination (e.g.: a jms-bus inside an ftp-provider).

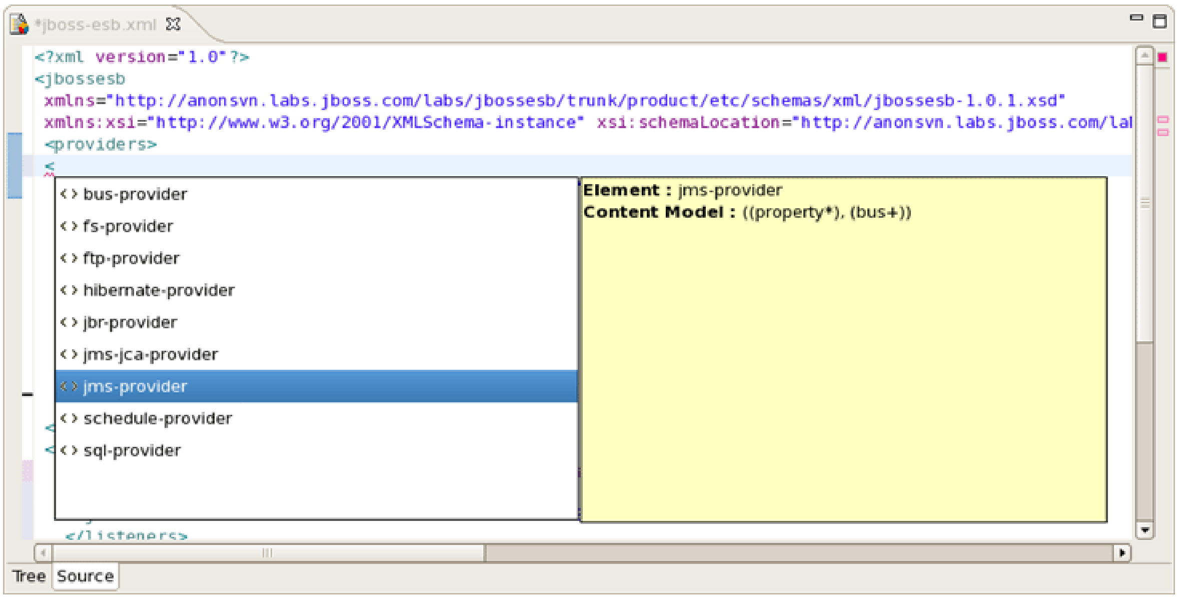
**Note:**

The schema used behind ESB editor now uses the latest version available (from SOA-P 4.3). This removes the errors/warnings some users have reported seeing when using SOA-P specific esb.xml files.

### 3.2.3. Content Assist for ESB XML File

When you work with any ESB XML file Content Assist is available to help you. It provides pop-up tip to help you complete your code statements. It allows you to write your code faster and with more accuracy. Content assist is always available in the Source mode. Simply type *Ctrl-Space* to see what is available.

Content Assist for ESB XML file:

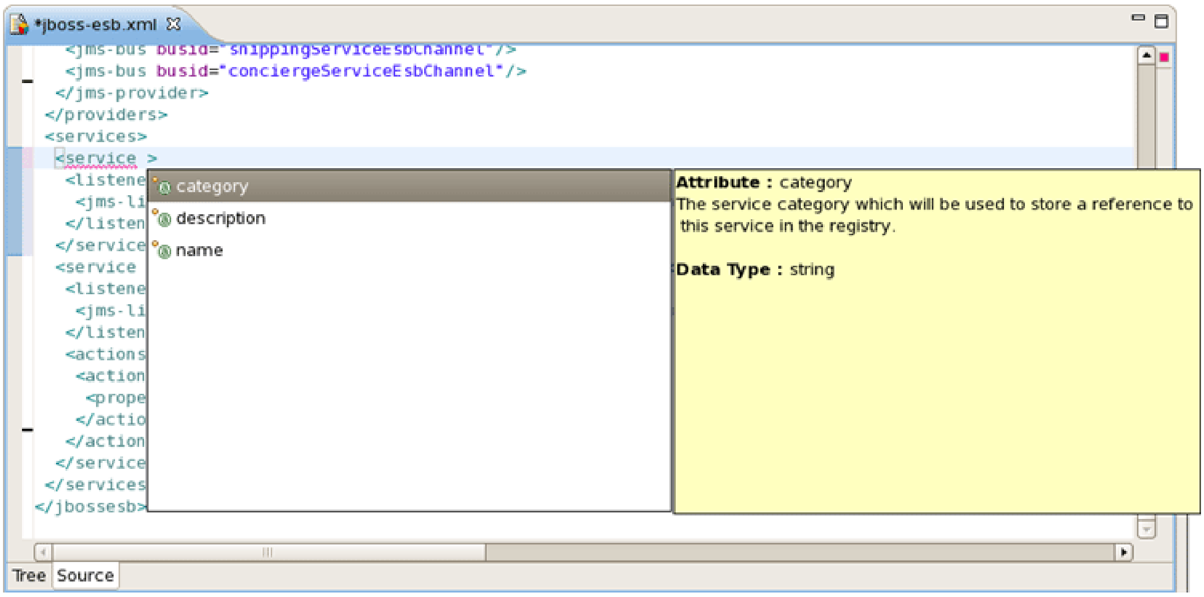


```
<?xml version="1.0"?>
<jbossesb
 xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.0.1.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://anonsvn.labs.jboss.com/lal
 <providers>
 <
   </listeners>
```

**Element :** jms-provider
**Content Model :** ((property*), (bus+))

### Figure 3.17. Content Assist for ESB XML file

Content Assist for attributes:



```
    <jms-bus busid="shippingServiceEsbChannel"/>
    <jms-bus busid="conciergeServiceEsbChannel"/>
   </jms-provider>
  </providers>
  <services>
   <service >
    <listene
     <jms-li
    </listen
   </service
   <service
    <listene
     <jms-li
    </listen
    <actions
     <action
      <prope
     </actio
    </action
   </service
  </services
</jbossesb>
```

**Attribute :** category
The service category which will be used to store a reference to this service in the registry.

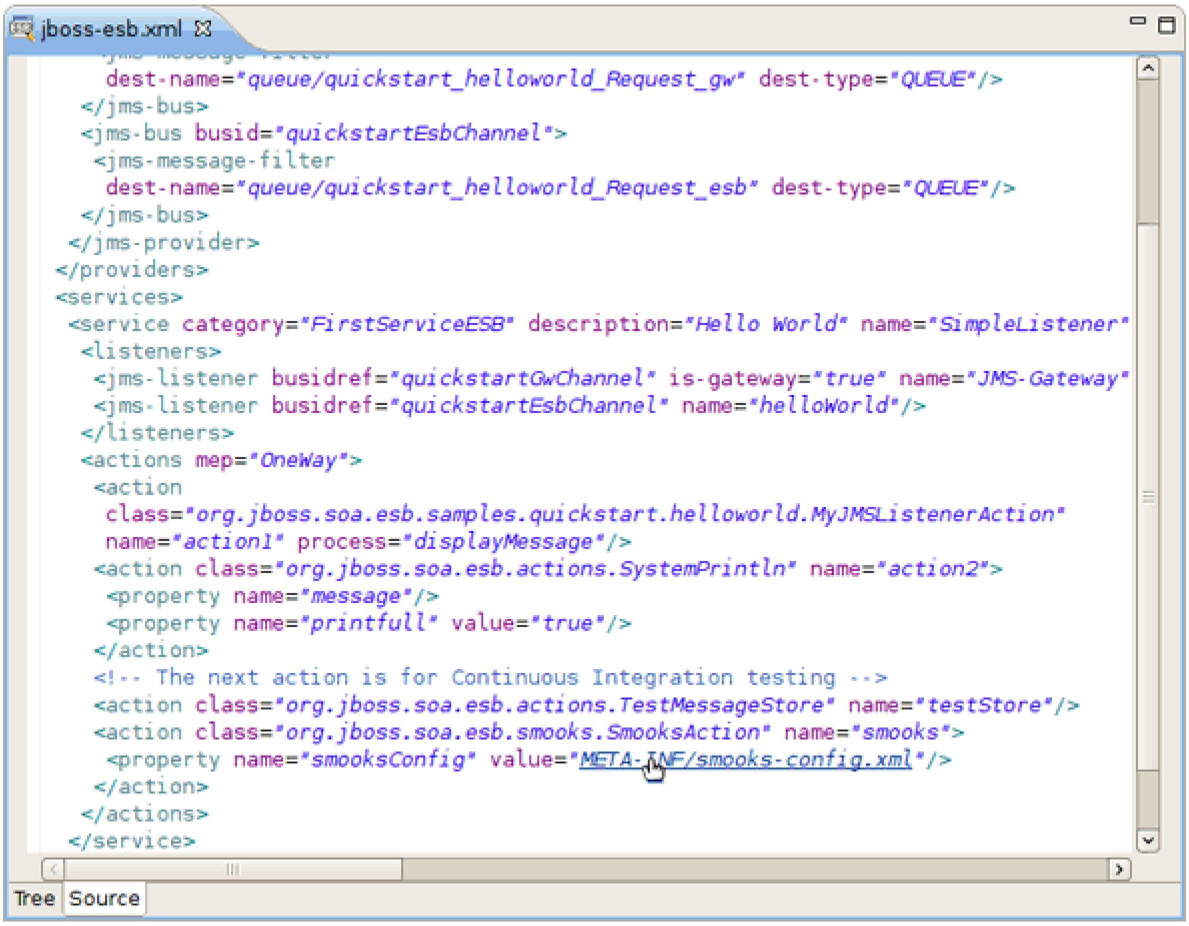
**Data Type :** string

### Figure 3.18. Content Assist for attributes

## 3.2.4. OpenOn for ESB XML File

ESB file comes with the OpenOn feature that allows to make use of multiple file references in the file just with a click and the *Ctrl* key hold down.

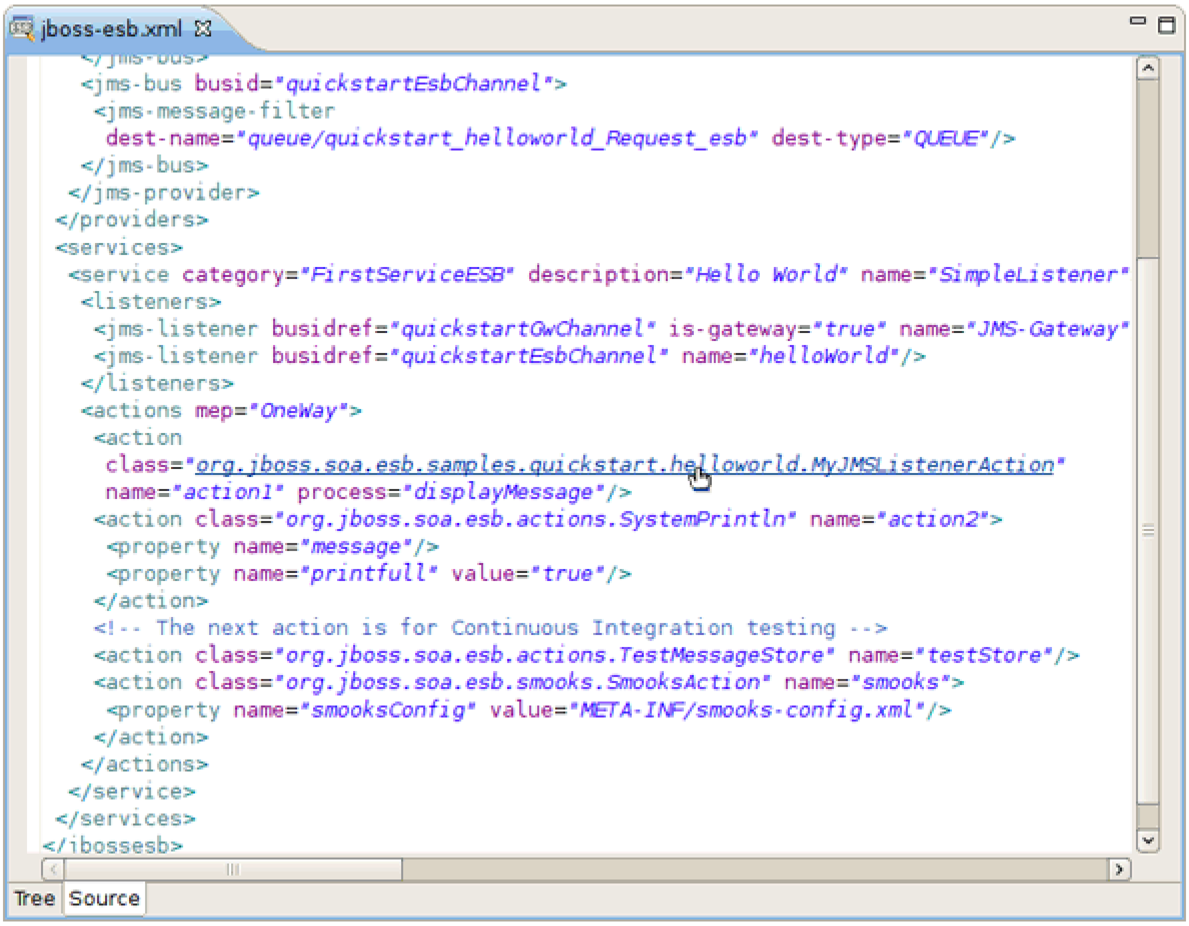
The OpenOn is implemented for different types of files/pages inside the `<action>` tag: .xsd, .xml, etc.



```
      dest-name="queue/quickstart_helloworld_Request_gw" dest-type="QUEUE"/>
    </jms-bus>
    <jms-bus busid="quickstartEsbChannel">
     <jms-message-filter
      dest-name="queue/quickstart_helloworld_Request_esb" dest-type="QUEUE"/>
    </jms-bus>
   </jms-provider>
  </providers>
  <services>
   <service category="FirstServiceESB" description="Hello World" name="SimpleListener"
    <listeners>
     <jms-listener busidref="quickstartGwChannel" is-gateway="true" name="JMS-Gateway"
     <jms-listener busidref="quickstartEsbChannel" name="helloWorld"/>
    </listeners>
    <actions mep="OneWay">
     <action
      class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
      name="action1" process="displayMessage"/>
     <action class="org.jboss.soa.esb.actions.SystemPrintln" name="action2">
      <property name="message"/>
      <property name="printfull" value="true"/>
     </action>
     <!-- The next action is for Continuous Integration testing -->
     <action class="org.jboss.soa.esb.actions.TestMessageStore" name="testStore"/>
     <action class="org.jboss.soa.esb.smooks.SmooksAction" name="smooks">
      <property name="smooksConfig" value="META-INF/smooks-config.xml"/>
     </action>
    </actions>
   </service>
```

## Figure 3.19. OpenOn for smooks configuration file

It is also available for classes:



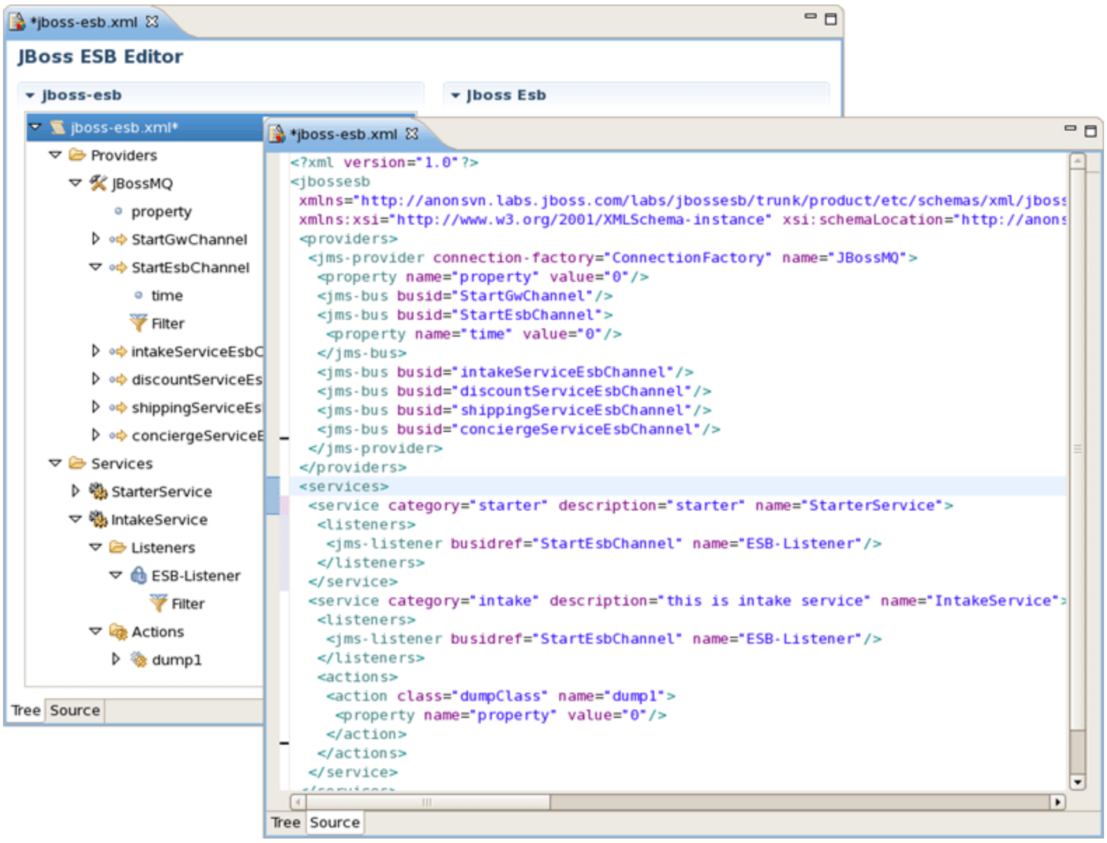
```
      </jms-bus>
      <jms-bus busid="quickstartEsbChannel">
       <jms-message-filter
        dest-name="queue/quickstart_helloworld_Request_esb" dest-type="QUEUE"/>
      </jms-bus>
     </jms-provider>
    </providers>
    <services>
     <service category="FirstServiceESB" description="Hello World" name="SimpleListener"
      <listeners>
       <jms-listener busidref="quickstartGwChannel" is-gateway="true" name="JMS-Gateway"
       <jms-listener busidref="quickstartEsbChannel" name="helloWorld"/>
      </listeners>
      <actions mep="OneWay">
       <action
        class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
        name="action1" process="displayMessage"/>
       <action class="org.jboss.soa.esb.actions.SystemPrintln" name="action2">
        <property name="message"/>
        <property name="printfull" value="true"/>
       </action>
       <!-- The next action is for Continuous Integration testing -->
       <action class="org.jboss.soa.esb.actions.TestMessageStore" name="testStore"/>
       <action class="org.jboss.soa.esb.smooks.SmooksAction" name="smooks">
        <property name="smooksConfig" value="META-INF/smooks-config.xml"/>
       </action>
      </actions>
     </service>
    </services>
   </jbossesb>
```

Figure 3.20. OpenOn for classes

## 3.2.5. Synchronized Source and Visual Editing

ESB file can be edited in either source or extra visual modes at the same time.

JBoss Tools provide you two different editors to speed your development: a graphical view (Tree) and source (Source). At the same time, you always have full control over esb source file. Any changes you make in the source view will immediately appear in the tree view. Both views are synchronized, you can edit the file in any view.



```
<?xml version="1.0"?>
<jbossesb
 xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jboss
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://anons
 <providers>
  <jms-provider connection-factory="ConnectionFactory" name="JBossMQ">
   <property name="property" value="0"/>
   <jms-bus busid="StartGwChannel"/>
   <jms-bus busid="StartEsbChannel">
    <property name="time" value="0"/>
   </jms-bus>
   <jms-bus busid="intakeServiceEsbChannel"/>
   <jms-bus busid="discountServiceEsbChannel"/>
   <jms-bus busid="shippingServiceEsbChannel"/>
   <jms-bus busid="conciergeServiceEsbChannel"/>
  </jms-provider>
 </providers>
 <services>
  <service category="starter" description="starter" name="StarterService">
   <listeners>
    <jms-listener busidref="StartEsbChannel" name="ESB-Listener"/>
   </listeners>
  </service>
  <service category="intake" description="this is intake service" name="IntakeService">
   <listeners>
    <jms-listener busidref="StartEsbChannel" name="ESB-Listener"/>
   </listeners>
   <actions>
    <action class="dumpClass" name="dump1">
     <property name="property" value="0"/>
    </action>
   </actions>
  </service>
```

## Figure 3.21. Two Views are Synchronized

# Summary

On the whole, this reference supplies you with all necessary information on the functionality that JBoss ESB plugin provides for work with JBoss ESB.

We hope, this guide helped you to get started with the JBoss ESB Tools. For additional information you are welcome on *JBoss forum* [http://www.jboss.com/index.html?module=bb&op=viewforum&f=201].

## 4.1. Other Relevant Resources on the Topic

You can find a set of benefits and other extra information on:

- *JBoss ESB* [http://www.jboss.org/jbossesb]
- *JBoss Wiki* [http://community.jboss.org/en/wiki]
- *JBoss ESB Documentation Library* [http://www.jboss.org/jbossesb/docs/index.html]

The latest JBoss Tools/JBoss Developer Studio documentation builds are available on the *JBoss Tools nightly documentation page* [http://download.jboss.org/jbosstools/nightly-docs/].