LogicBlaze FUSE Tooling

07/06 Version 1.0

LogicBlaze FUSE Tooling Guide

This document is a developer's guide explaining the Eclipse-based tooling for LogicBlaze FUSE.

Contents

1. Introduction To LogicBlaze FUSE Tooling

The easiest way to work with the FUSE platform is through the use of the Eclipse-based tooling. The premise behind the Eclipse-based tooling is to make FUSE a first-class citizen of the Eclipse integrated developer environment (IDE), providing access to functionality from the FUSE platform through the standard Eclipse mechanisms. One of the most important parts of this tooling is the integration with the Web Tooling Platform (WTP) - this allows you to work with the FUSE Server in much the same way are you would use a J2EE server within Eclipse, providing seamless integration with the environment and giving you a clean and simple way to work with the FUSE platform.

Much of the tooling is designed to work with Apache Maven (http://maven.apache.org), this provides a clean way to perform not only the construction of the various package types that FUSE accepts, but also leverages plugins for the construction of many of the descriptors. This all provides a smoother and more structured development environment, and if you are working with JBI components, then the inclusion of the Apache ServiceMix Maven Plugin provides an excellent basis for both packaging and deployment.

For more information on how the Maven JBI plugin works and the functionality it offers see the ServiceMix Web site (http://www.servicemix.org/site/maven-jbi-plugin.html).

2. Prerequisites

The following software is required:

- Callisto - Eclipse 3.2 with the Web Tools Platform (WTP)
- LogicBlaze FUSE 1.2

3. Defining Servers

One of the first features implemented was integrating the FUSE platform with the Eclipse Web Tools Platform. This was done by providing a server definition for the FUSE server that allows you to reference an installed version of FUSE and start/stop the server. This feature is also the first step toward being able to use the platform for projects.

In order to configure a server instance for use you can simply open the Servers View in Eclipse; to do this go to the “Window>Show View >Other...” and select “Servers”.

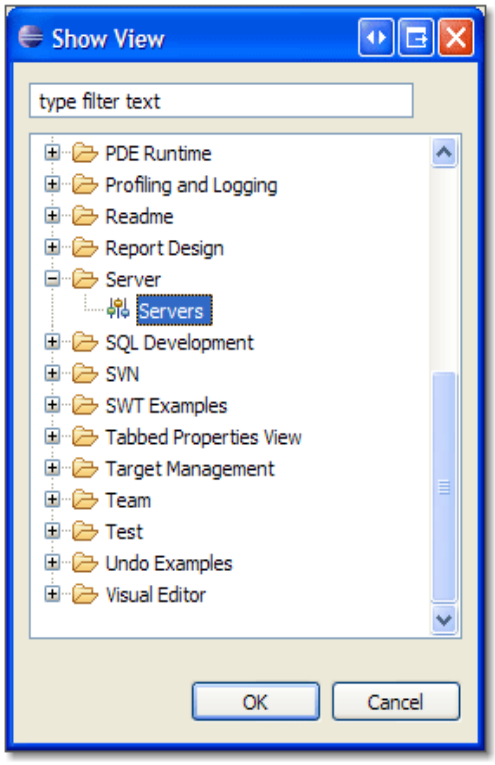
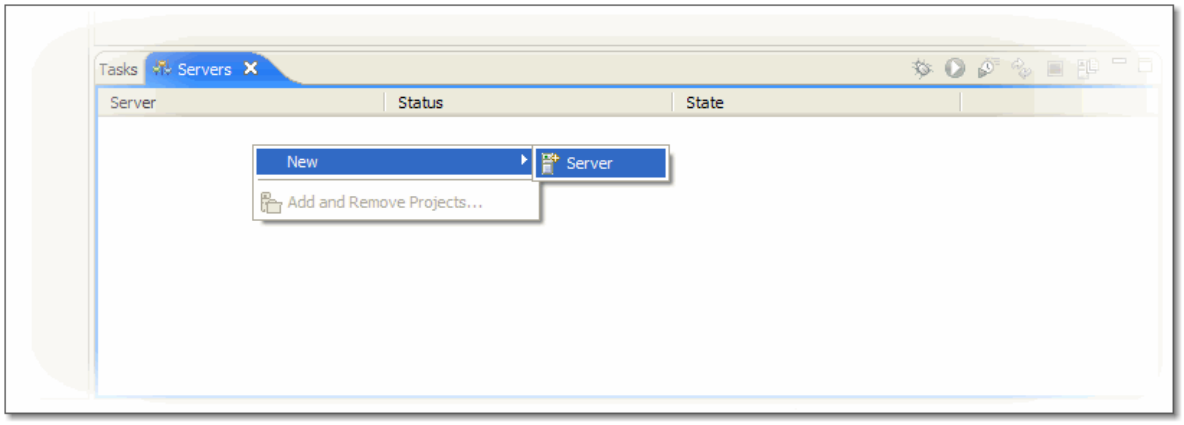


Figure 1: Servers View

Once you have the view open you should see it at the bottom part of the Eclipse workspace. You can right click on the empty pane and click “New>Server”.



This will bring you to the standard screen for adding JEE servers for management in the platform. You can simply expand the LogicBlaze selection and choose the FUSE server version you have installed.

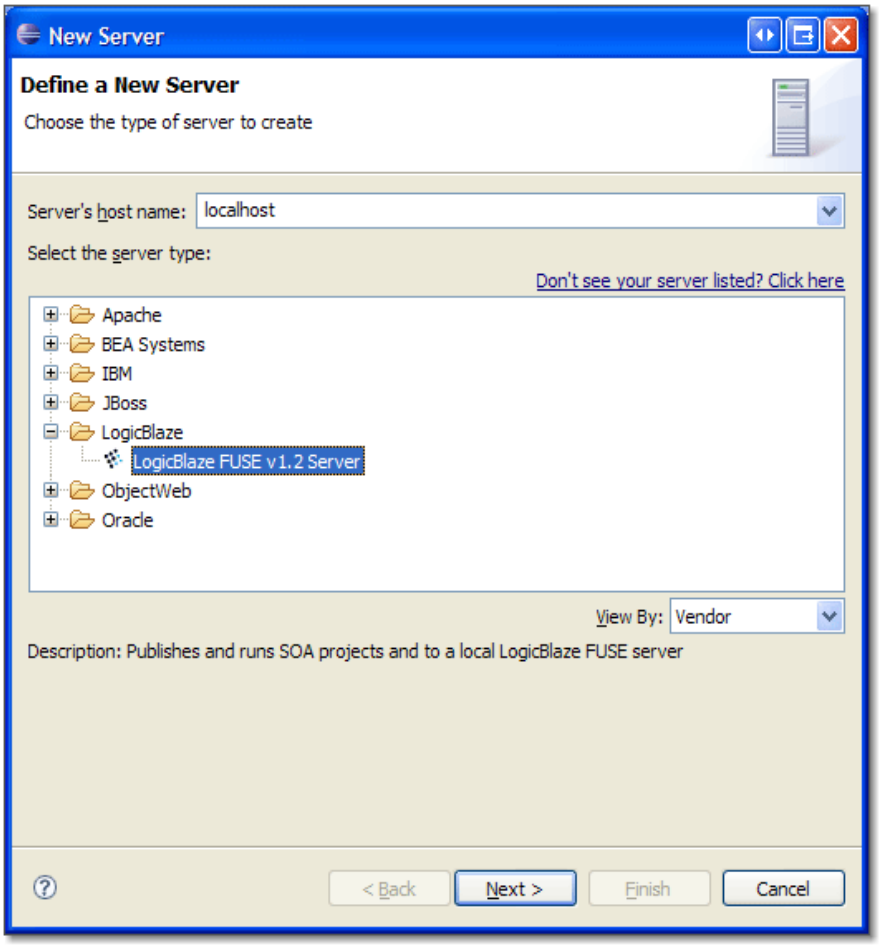


Figure 3: Define a New Server

Note that this requires that you have a local installation of FUSE available

When you select "Next" you are asked to provide the path to your FUSE installation.

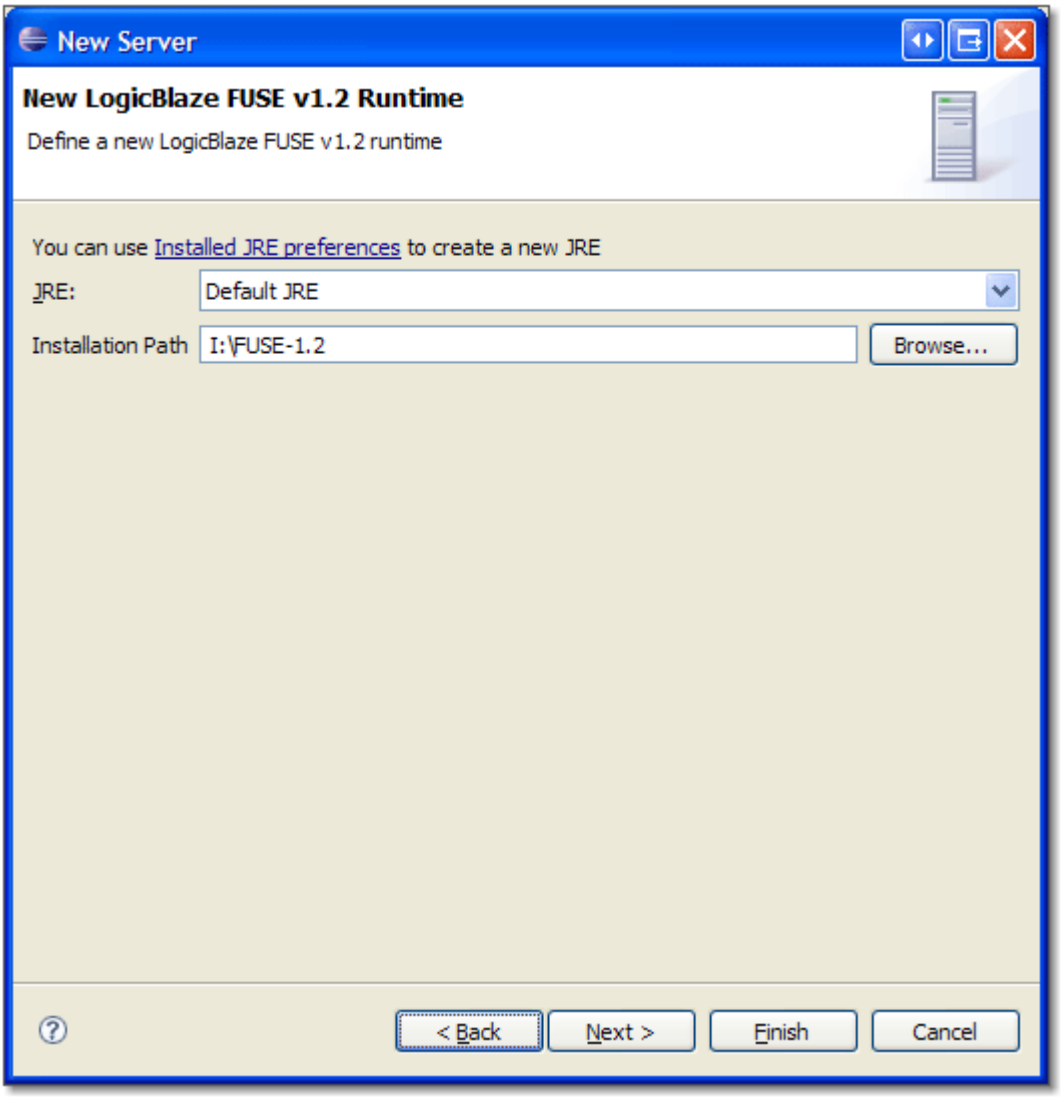


Figure 4: Select FUSE Installation Path

Pressing "Next" will allow you to set the settings for your Server. The defaults are in line with a standard FUSE installation, however if you have changed your FUSE installation (for example, changed the port number) then you will need to change the settings to reflect that. This ensures that Eclipse can communicate with the server.

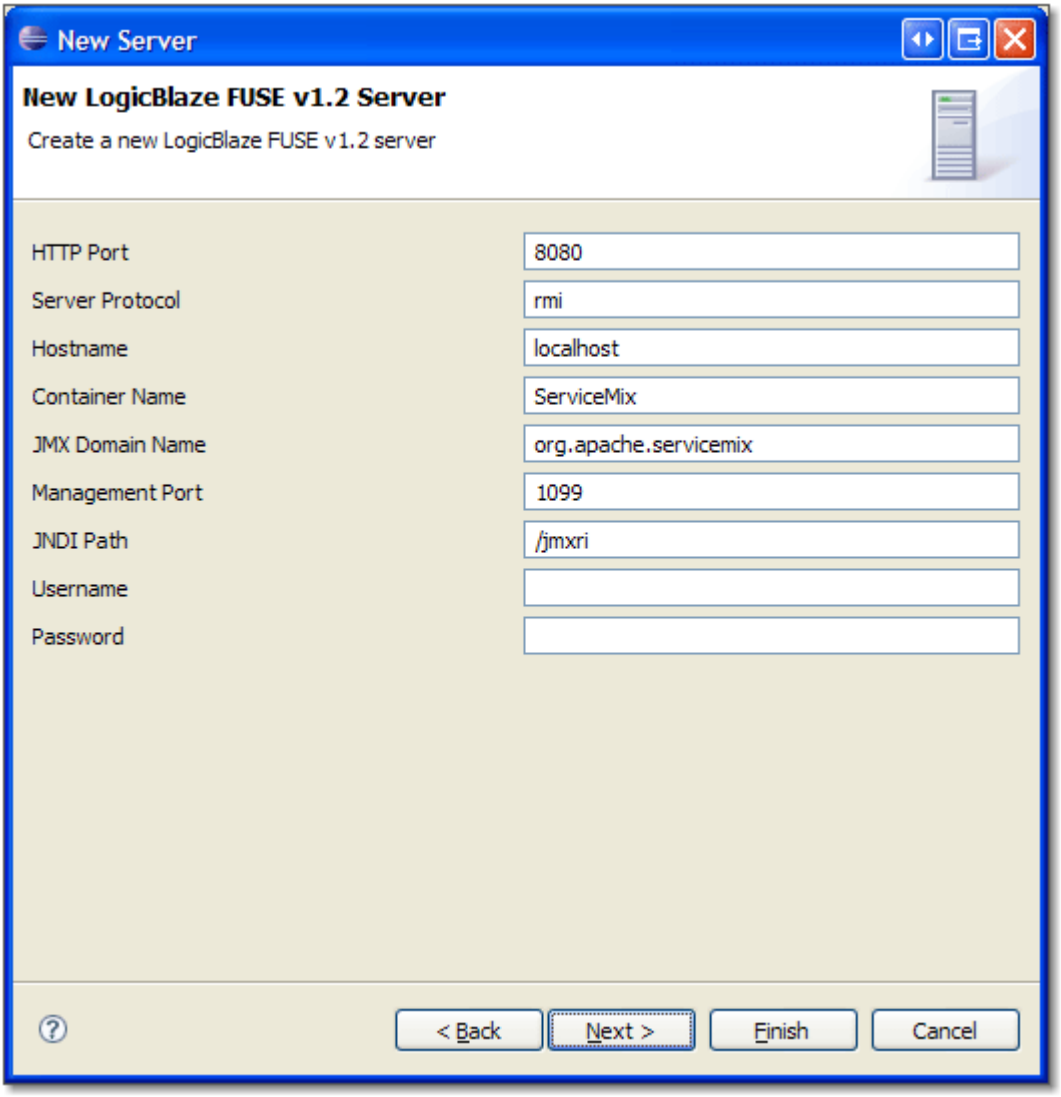


Figure 5: Configuring settings for the FUSE server

Once you have informed Eclipse of your server configuration you can click "Finish" to create the server. You should then see the server showing in the server view. Note when the status is "stopped" that means it is not currently running under Eclipse. The state shows whether projects in the workspace need to be published to it, because they have changed.

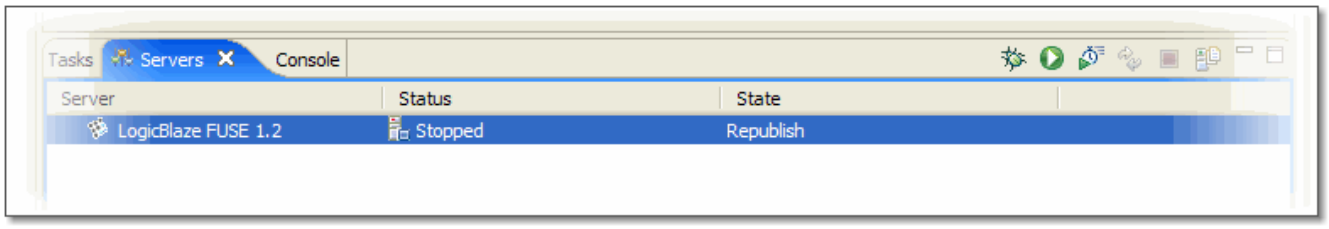


Figure 6: Server status

You can simply prove that the server is working properly by right clicking and choosing Start, this will boot up an instance of FUSE using the runtime you provided under Eclipse. The output from the running version of FUSE will show in the Eclipse console and can be used to determine if it is starting properly. Once fully started you should see the status of the Server switch from "Starting" to "Started". You may have to click the Servers tab to see this. This completes defining a server under Eclipse.

4. Creating New Projects

Getting started with any type of project can be difficult, and one of the great features of any tooling is providing a set of wizards to allow you to quickly choose the type of project you wish to create and then create it.

The FUSE tooling provides a set of wizards which are linked to archetypes that can be used to create projects which are fully configured to work with Eclipse tooling. One of the important features of the FUSE tooling is that it works with the Apache Maven tooling, this means that when you create a new project type the dependencies you might need, such as third-party libraries, are automatically downloaded for you making it quick and easy to get started.

In order to select a project click "New>Project" from the File menu in Eclipse.

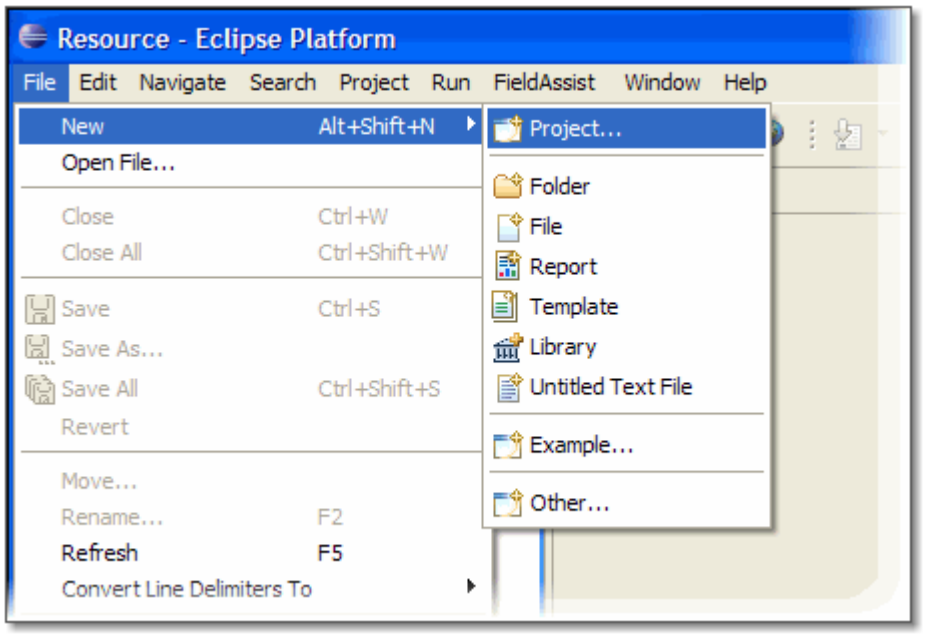


Figure 7: New Project

This will take you to the New Project dialog where you can see all the available project types in Eclipse. The actual projects you will have available are based on the plugins you have in your Eclipse environment, though the FUSE tooling creates two categories for the projects. The two project categories are Java Business Integration and Java Business Integration Service Units.

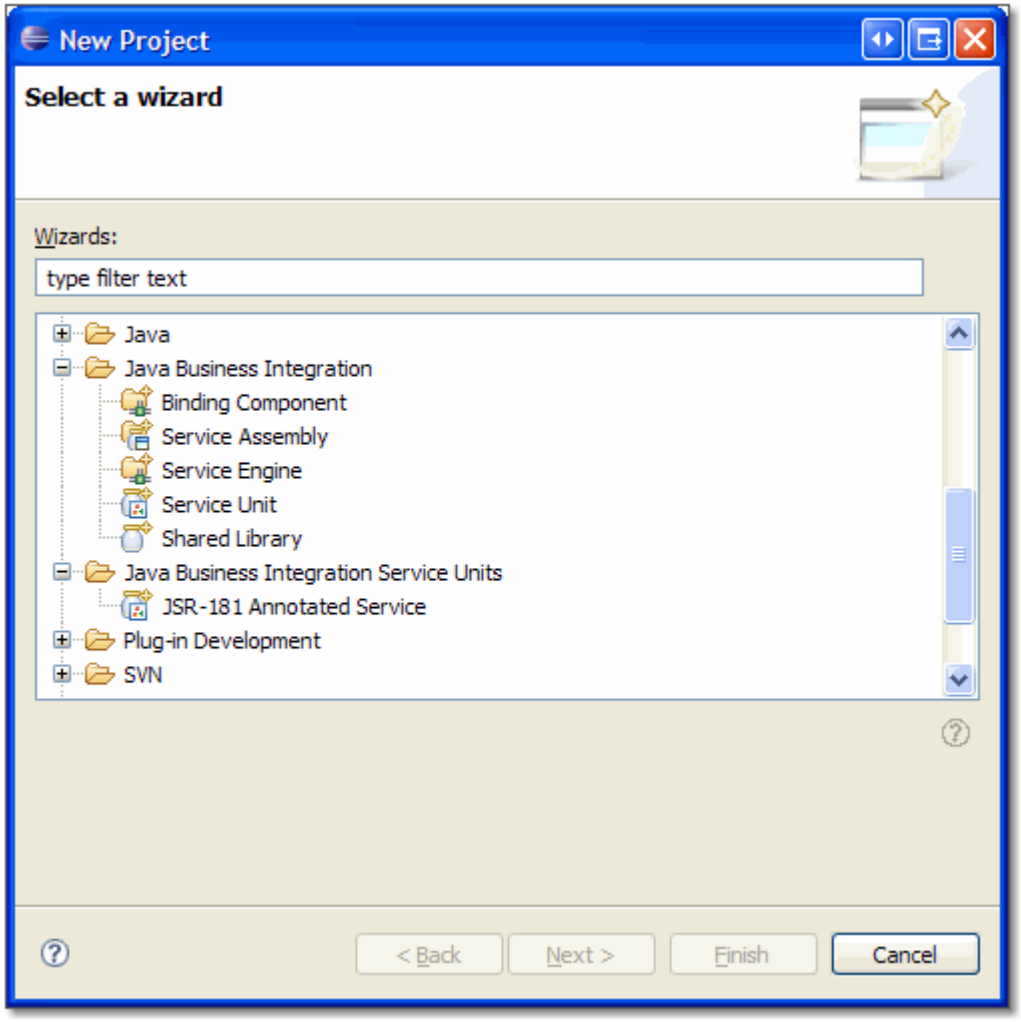


Figure 8: New Project dialog

The two categories provide access to two different aspects of development on FUSE.

Java Business Integration

The Java Business Integration project wizards are in place to allow you to create your own shared libraries, components, service engines and service assemblies. Though they don't provide out-of-the-box functionality, they do provide you with the basic project and code structures for each of these component types, and should be used as the basis for creating your own services.

Java Business Integration Service Units

These JBI service unit definitions can be used when you wish to leverage an existing component or service engine, such as the JSR-181 annotated service engine. They provide the ability to quickly define a service unit for a component based on a template. The service unit will already reference its component or service engine and since the projects are based on Apache Maven functionality these components will be automatically downloaded for you.

For this example we will go through the creation of a Service Engine, therefore, you simply select the Service Engine option from under the Java Business Integration category and click "Next".

This will display the Service Engine wizard page; on this page we can enter the name of our new project.

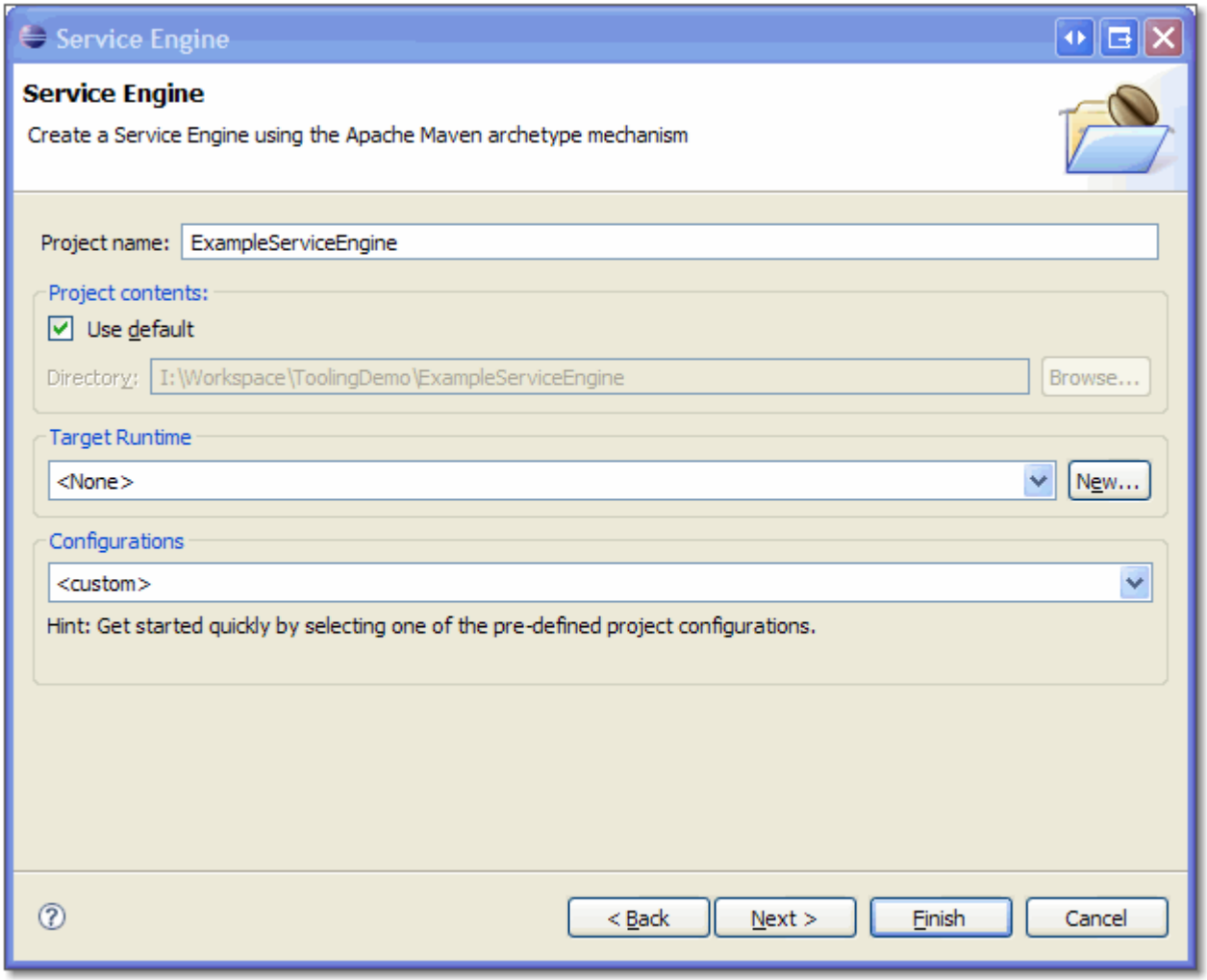


Figure 9: Service Engine wizard

Note that we can also select a runtime for the engine (though this isn't necessary) and also use any custom configurations, if there are any available, for the project type are you creating.

Simply pressing "Next" will bring you to the Facet view. Facet's are a way of associating Eclipse functionality with a project. The FUSE tooling has been written to take advantage of this functionality. For most projects we see that the facets are fixed and can not be changed.

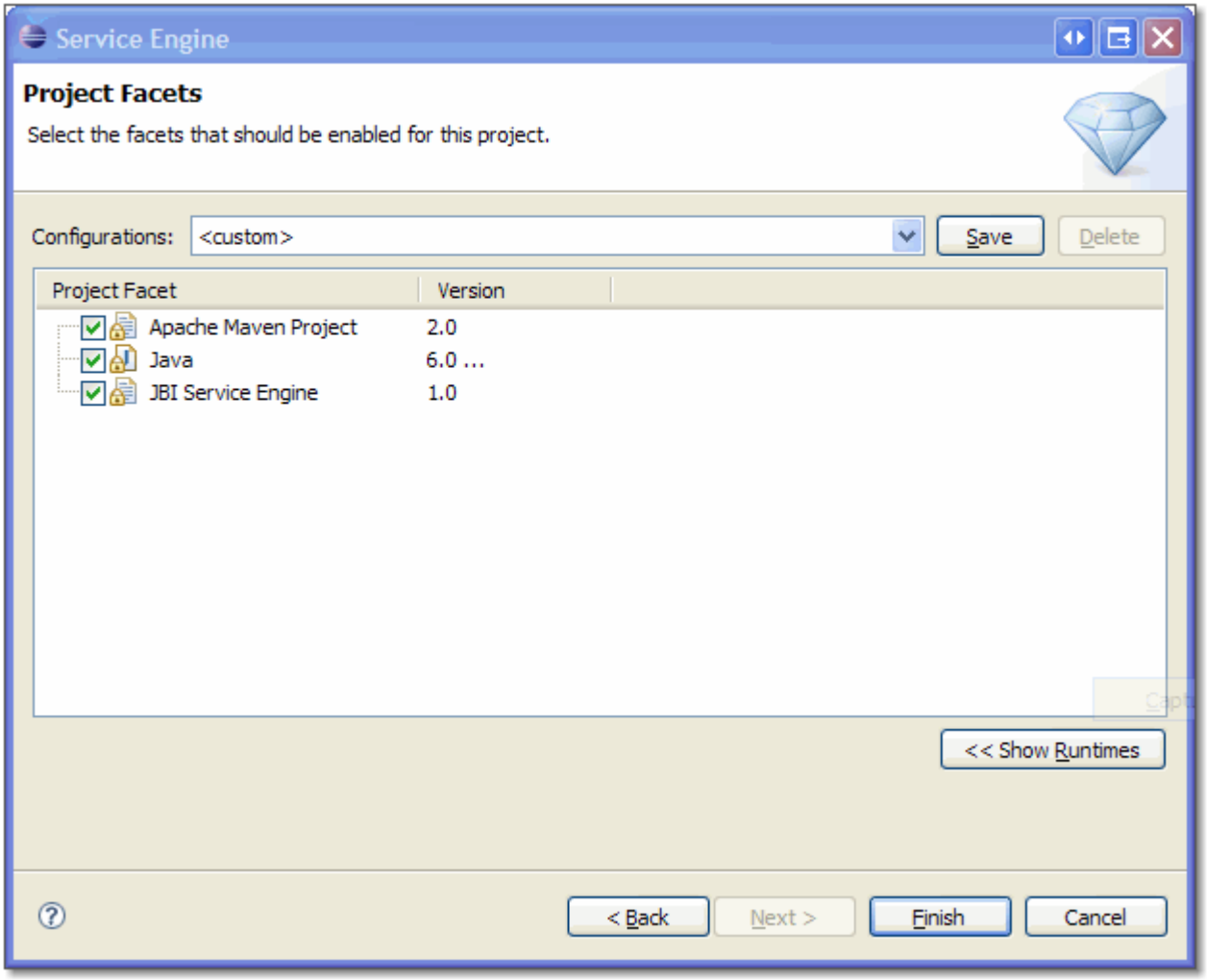


Figure 10: Project Facet view

You can simply click "Finish" to complete the wizard and you should then find that a project is created in your project with the given name. The use of Apache Maven will mean that all dependencies are downloaded for you and therefore once the project has been built you can immediately deploy it to a server.

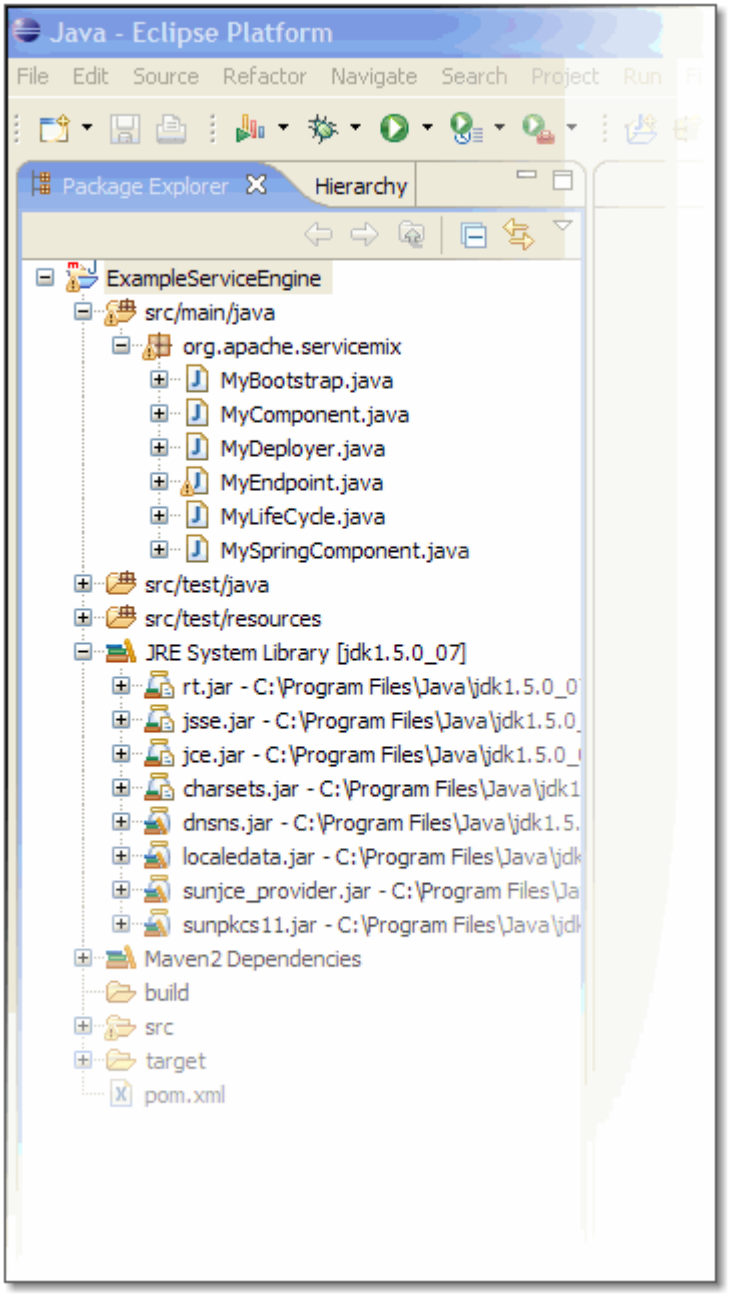


Figure 11: Eclipse Package Explorer showing the example service engine

Understanding a little about Facets

Facets are a powerful feature of Eclipse and worth a little more information. Basically the FUSE tooling provides five new facets that a project can have, these are:

- JBI Binding Component
- JBI Service Engine
- JBI Service Assembly
- JBI Service Unit
- JBI Shared Library

Each represents one of the packaging types within the JBI specification, and it is the presence of these facets on a project that determine whether is can be deployed to a JBI compliant server such as FUSE or ServiceMix. Also note that we use facets such as those from Apache Maven to enable Maven functionality on the projects.

5. Working With Servers

One of the powerful features of the FUSE tooling is the ability to deploy projects to running Servers, which is built around the Eclipse Web Tools Platform concept of Servers and allows you to see FUSE and ServiceMix as Server that can be associated with a project in much the same way as a J2EE server (such as IBM WebSphere).

In order to do this you first need to have defined a server and also created a JBI project that can be deployed. The deployable JBI projects are:

- JBI Service Engines
- JBI Binding Components
- JBI Shared Libraries
- JBI Service Assemblies

If you have a project you can associate it with a defined server, to do this you right click on the Server from the Server View and click “Add/Remove Projects”.

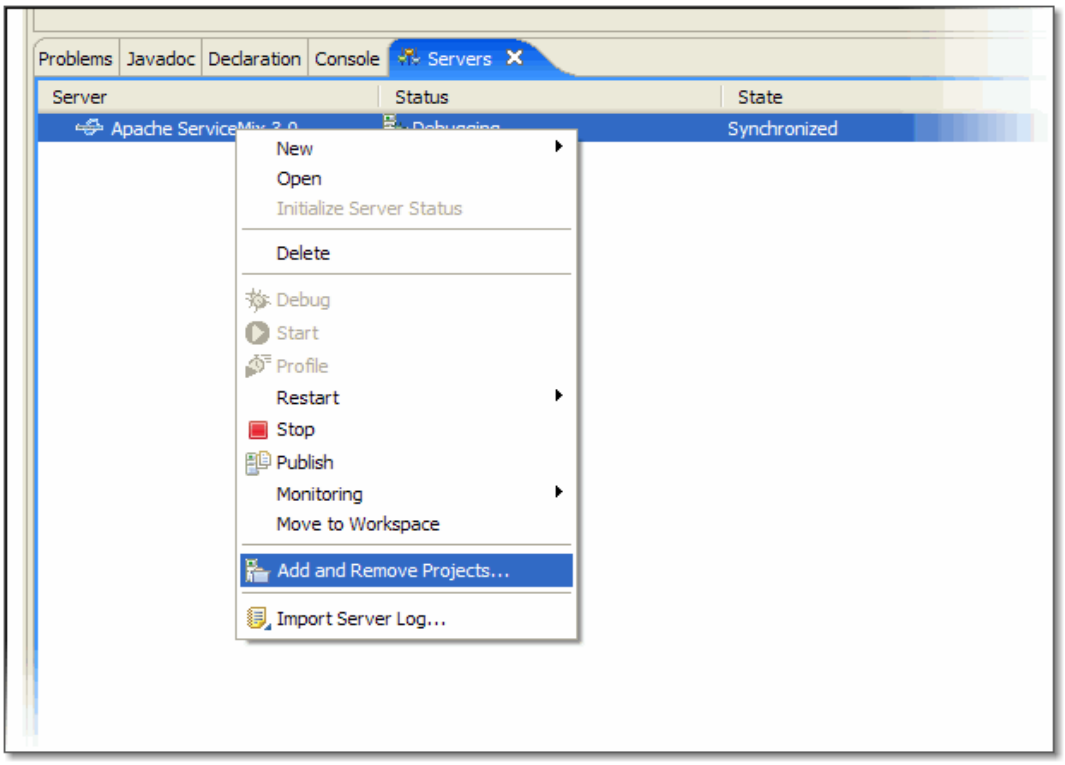


Figure 12: Adding and Removing projects

Once you select “Add and Remove Projects” you can select the available projects from your workspace that you wish to deploy. The projects listed here are those that are eligible for deployment on the Server - for FUSE this means a project with one of the JBI deployable facets.

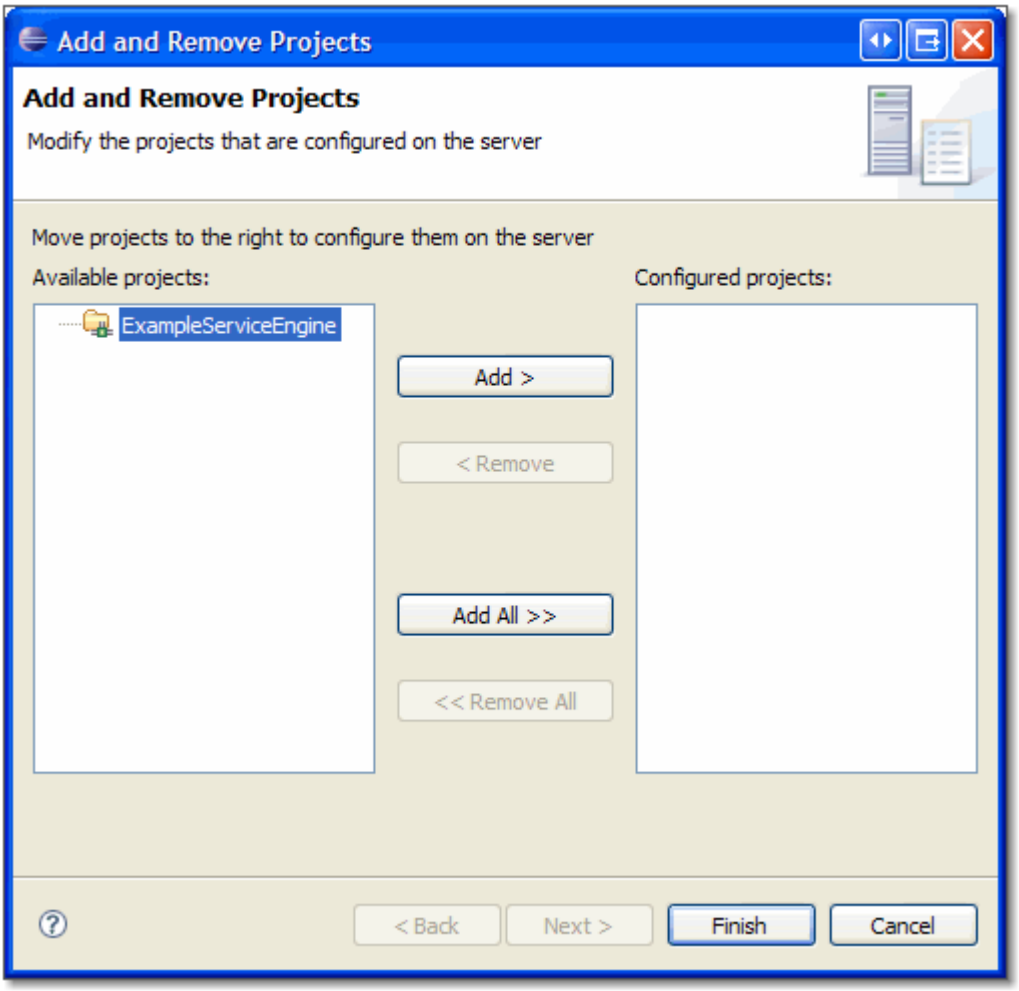


Figure 13: Add and Remove Projects view

Use the “Add/Remove” buttons to decide which projects you wish to run on the server. Once you have added your project you can simply click "Finish". You will then see the project shown under the Server on the Server View.

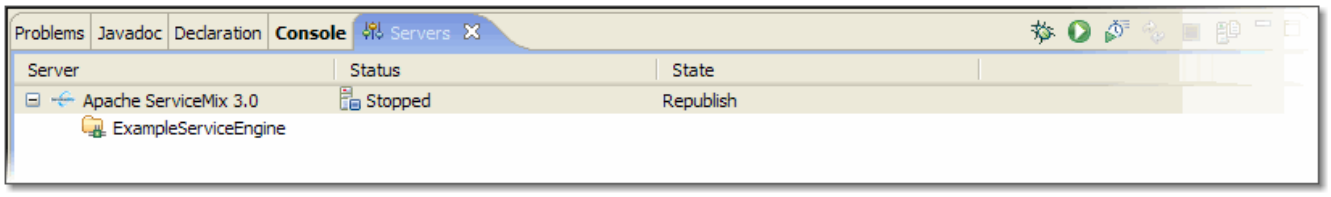


Figure 14: Example project under the Server

You can now start your Server. When you start the Server the project will be automatically published to the server. It is important to note that by default the publishing mechanism determines whether your project is dependent on other components or shared libraries, if so these will be automatically deployed with it. This is provided to make the process of deploying components to the server quick and easy for developers, taking away the problems of determining and deploying dependencies outside of Eclipse.

If you modify your project in any way Eclipse will recognize the changes and the state of the project will switch to "Republish". When you want to do this, right click on the Server in the Server View and click “Publish”. This will stop the project on the server (and its dependencies) and redeploy each of them again.

It is also important to note that you can start the Server in Debug Mode, this provides a great way of debugging your code while running against a FUSE server without any special configuration. The debugger will function just as it does normally in Eclipse.

6. Additional Resources

To read the Java Business Integration specification (JSR 208) please see:
http://www.jcp.org/en/jsr/detail?id=208.