

OSGi services in DSI v8.8

Creating OSGi bundles

The first step of integrating an OSGi service is to create an OSGi bundle in Eclipse. You then create a Java™ interface and a Java class implementation, and you define the dependencies within the service.

1. Click **File > New > Other > OSGi > OSGi Bundle Project**, and then click **Next**.
2. In the wizard, specify the name of your project.
3. Clear the **Add bundle to application** option, and click **Next** twice.

New OSGi Bundle Project

OSGi Bundle Project

Create a standalone OSGi bundle project or add it to a new or existing application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Configuration

☐ Add Web support

☐ Add persistence support

☐ Add EJB support

☐ Custom

☒ Generate blueprint file

Application membership

☐ Add bundle to application

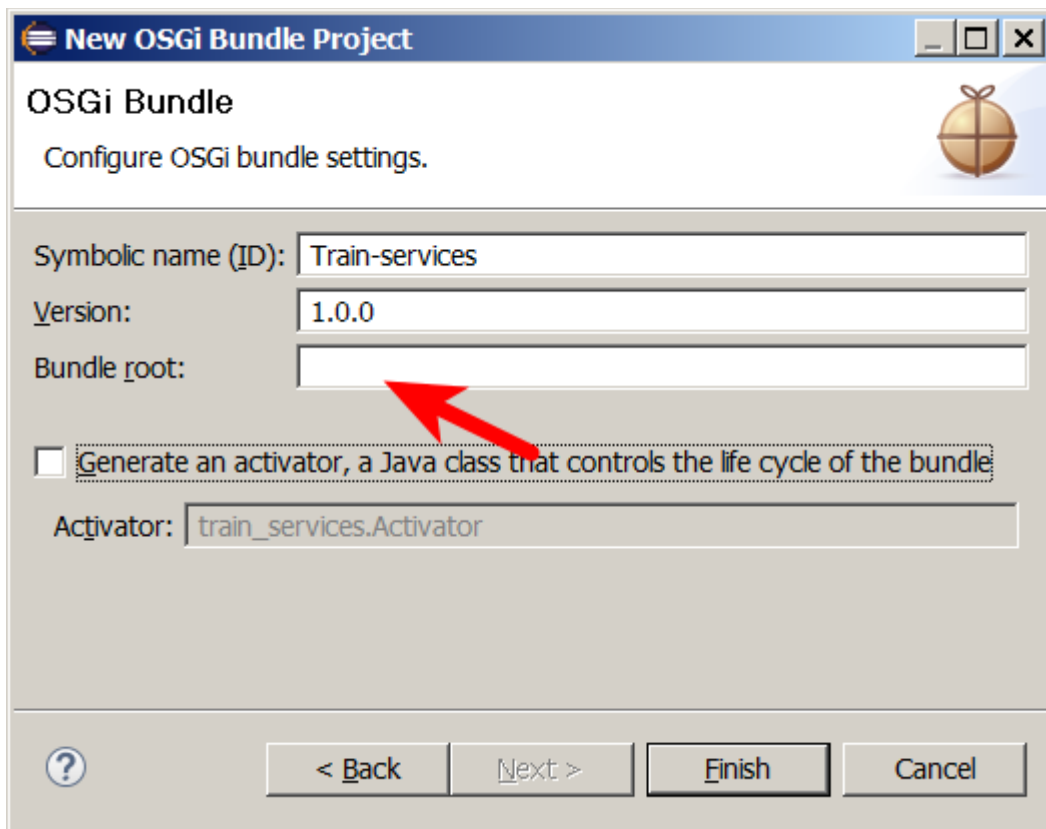
Application project:

Working sets

☐ Add project to working sets

Working sets:

4. To build and deploy the service outside of the Plug-in Development Environment build, **remove** .qualifier from the version name, and **check** the Generate an activator option.
5. Clear **Bundle root**



6. Click **Finish**.
7. Open blueprint.xml
8. Click Add, select Service then OK
9. Click **New** for Service Interface and follow the wizard to define a new interface for the service

New Java Interface

Java Interface
Create a new Java interface.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected

Extended interfaces:

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

10. Click **New** for Bean Reference, then again **New** for Bean Class. Define the implementation class and add the previously defined interface.

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

11. Click **OK**, then **OK** to complete the definition. **Close and save** the blueprint.xml
12. Open the MANIFEST.MF file
13. In Dependencies tab, **add packages** needed in your service. If you need reference the Solution Business Object Model add here (e.g. train)
14. In Runtime tab, in the export section click Add.. to add the packages that contains the service interface. (e.g. services.api)
15. Add the Java logic.

Prepare the OSGi build

Since this OSGi service is linked to the solution model, it should be embedded to the Solution build.

In order to ensure that the project is correctly built, you need to change the build properties:

1. Open **build.properties** file and change the content with the following text:

```
source.. = src/  
output.. = bin/  
bin.includes = META-INF/, \  
               ., \  
               OSGI-INF/
```

Pay attention to the “.”!

2. In the **solution project**, add a reference to the OSGi service project:
 1. Right-click the solution project, and click Properties.
 2. Click Project References, and select the OSGi service project. Click **OK**.

Consuming the OSGi services

Create a rule project

1. Click **File > New > Other > Insight Designer > Rule Project**.
2. Follow the wizard to create a standard rule project. (Suggestion: use a naming convention that help you to identify it as the BOM of the OSGi Service project. Eg. If the services project is named Train-services, use Train-services-BOM for this project)
3. On the Rule Project References page, add the .
4. On the **Rule Project XOM Settings** page, select the OSGi service project that contains your service, and click Finish. This setting is used at export time to check the validity of the OSGi service API.
5. In the Solution Explorer, right-click the new rule project and click Properties.
6. Click **Rule Engine**, select Decision Engine, and click OK. (The Classic Rule Engine is not supported in Decision Server Insights.)
7. Click **Project References**, add the reference to the solution BOM project.

Create a BOM entry

1. Right-click the new rule project, and click New > Other > Insight Designer > BOM Entry, and then click Next.
2. Enter a name for the BOM entry, select **Create a BOM entry from a XOM**
3. Browse XOM and select the services project.
4. Select the interface then **Next**.
5. Select all methods and remove the term verbalization

New BOM Entry

BOM Verbalization

Select the business elements to verbalize and define the verbalization.

Select the elements to verbalize.

Verbalize: ☒ Getters ☒ Setters ☒ All Methods ☐ Static References

- ☒ services
 - ☒ api
 - ☒ Calculate
 - ☐ the calculate, a calculate, the calculates...
 - ☒ count(Coach)

Select All Deselect All

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

6. Click **Finish** and Open the BOM Entry to perform some adjustment:

1. Open the method make it **static** and choose the preferred verbalization.

services_model

Member count (class: services.api.Calculate)

General Information

Name:

Type: [Browse...](#)

Class: [Browse...](#)

☒ Static ☐ Final

☐ Deprecated ☐ Update object state

Member Verbalization

✗ [Remove](#) the verbalization.

+ [Create](#) a navigation phrase.

▼ **Navigation : "count a coach"** ✗

Template: [?](#)

2. In the **Class** tab, expand the Custom Properties section and click Add. You might have to

scroll down in the BOM editor to view the Custom properties section.

3. Enter **OSGi.service** as the name of the property, and the fully qualified name of the Java type for your OSGi service as the value. You can also add an **OSGi.service.version** property to refer to a specific version of the service. If you don't specify a version, the most recent version of the service available on the server is used.

▼ Custom Properties
Define custom properties for this class.

Name	Value	
OSGi.service	service.api.Calculate	

Add
Remove

7. Save the changes.
8. If you later, you have to add other method to the service, you need to add the other class manually. Otherwise you can repeat the automatic generation from the XOM, but you'll loose the adjustment performed at previous step.
9. In the rule agent project, add a reference to the rule project that contains the service BOM:
 1. Right-click the agent project, and click Properties.
 2. Click Project References, and select the rule project where you created the service BOM. Click OK.