



Platform Extensibility

Eclipse

Overview

Package Explorer

- Example Extension
 - src
 - JRE System Library [JavaSE-1.8]
 - Referenced Libraries
 - configfiles
 - lib
 - build-extension.xml

ExampleThing.java metadata.xml build-extension.xml

```
18
19 public class ExampleThing extends Thing {
20     private static final long serialVersionUID = -734037441678274101L;
21
22     protected Logger logger = LogUtilities.getInstance().getApplicationLogger(this.getClass());
23
24     @Override
25     protected void initializeThing() throws Exception {
26         super.initializeThing();
27         // Custom initialization
28     }
29
30     @ThingworxServiceDefinition(
31         name = "GetEntityCounts",
32         description = "Returns an InfoTable of entity counts sorted in descending order..")
33     @ThingworxServiceResult(
34         name = "result",
35         description = "An InfoTable (EntityCount) of entity counts.",
36         baseType = "INFOTABLE",
37         aspects = { "dataShape:EntityCount" })
38     public InfoTable GetEntityCounts() throws Exception {
39         InfoTable result = new InfoTable();
40
41         String[] entityTypes = {
42             "Things", "ThingTemplates", "ThingShapes", "DataShapes", "Networks",
43             "ModelTags", "Mashups", "Dashboards", "Menus", "MediaEntities",
44             "StyleDefinitions", "StateDefinitions", "DataTags", "PersistenceProviders",
45             "Users", "Groups", "Organizations", "ApplicationKeys", "DirectoryServices",
46             "LocalizationTables", "Resources", "Subsystems", "Logs"
47         };
48
49         try {
50             InfoTable entityCounts = InfoTableInstanceFactory.createInfoTableFromDataShape("name", "EntityCount"); // name and count
51
52             PlatformSubsystem platformSubsystem = (PlatformSubsystem) SubsystemManager.getInstance().getEntity("PlatformSubsystem");
53
54             // Get entity counts
55             for (int i = 0; i <= entityTypes.length; i++) {
56                 InfoTable entityCount = platformSubsystem.GetEntityCount(entityTypes[i], null);
57
58                 ValueCollection value = new ValueCollection();
59                 value.put("name", new StringPrimitive(entityCount.getRow(0).getValue("name").toString()));
60                 value.put("count", new IntegerPrimitive(Integer.parseInt(entityCount.getRow(0).getValue("count").toString()));
61                 entityCounts.addRow(value);
62             }
63
64             // Sort entityCounts in descending order
65             InfoTableFunctions infoTableFunctions = (InfoTableFunctions) ResourceManager.getInstance().getEntity("InfoTableFunctions");
66             result = infoTableFunctions.Sort(entityCounts, "count", false);
67         } catch (Exception e) {
68             logger.error("null");
69         }
70
71         return result;
72     }
73 }
```

Outline

- com.example.things
 - ExampleThing
 - serialVersionUID : long
 - logger : Logger
 - initializeThing() : void
 - GetEntityCounts() : InfoTable

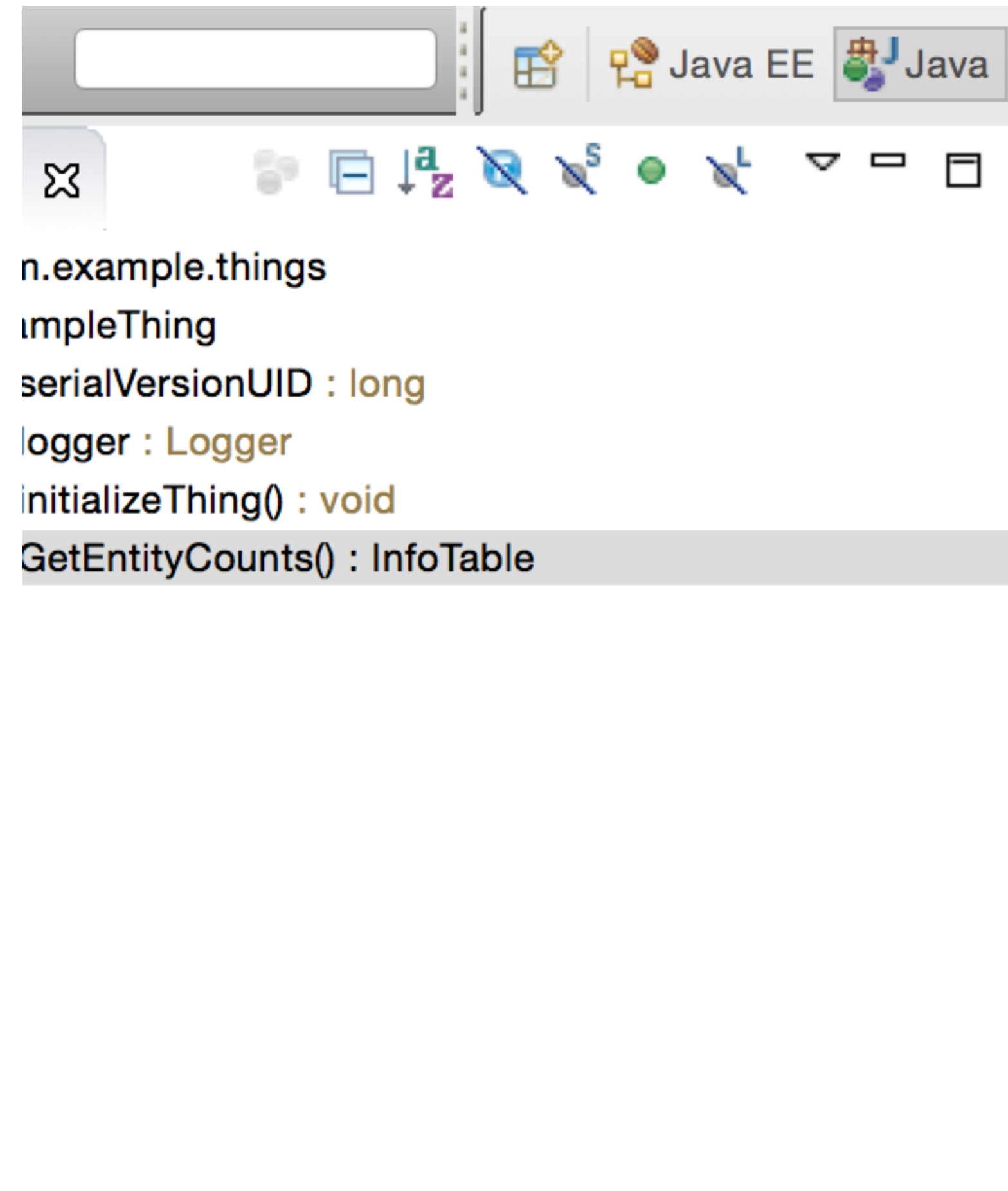
Problems Javadoc Declaration Console

No consoles to display at this time.

Perspectives

A Java perspective is a collection of views and an editor.

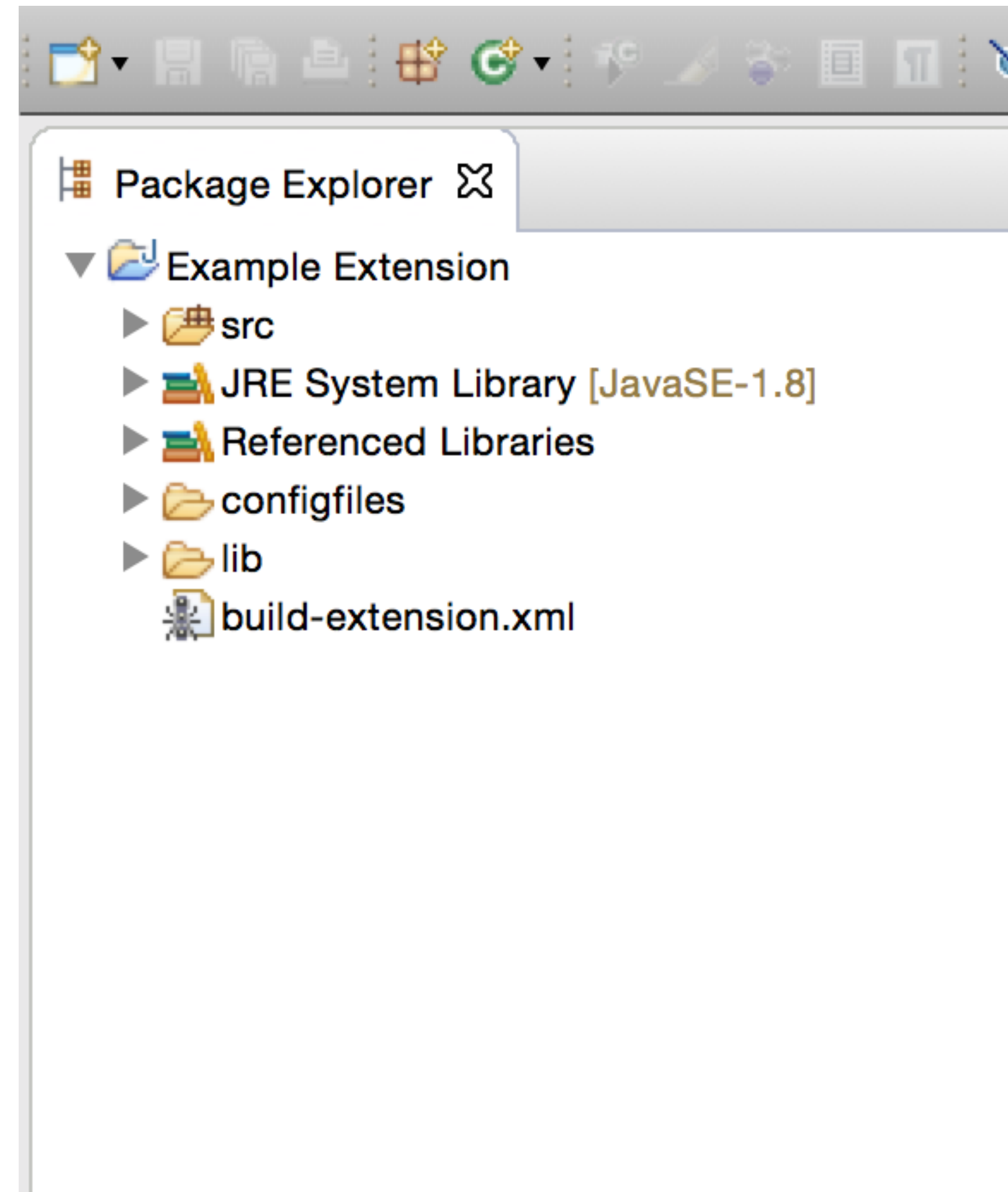
The Java perspective is designed for working with Java projects; the Debug perspective is designed for debugging Java programs.

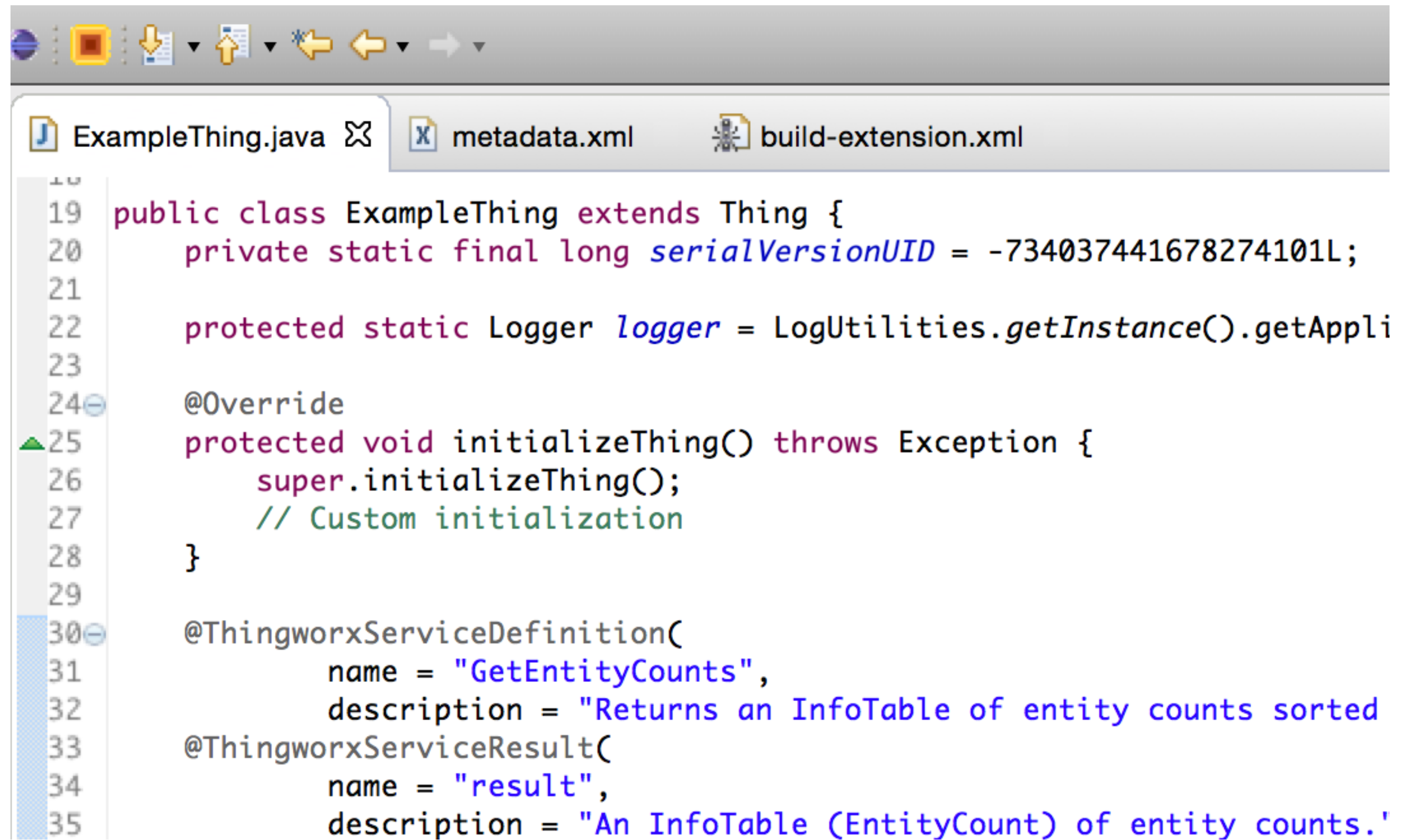


Package Explorer

The Package Explorer view shows the Java element hierarchy of the Java project.

It provides a Java-specific view of the resources shown in the Navigator.





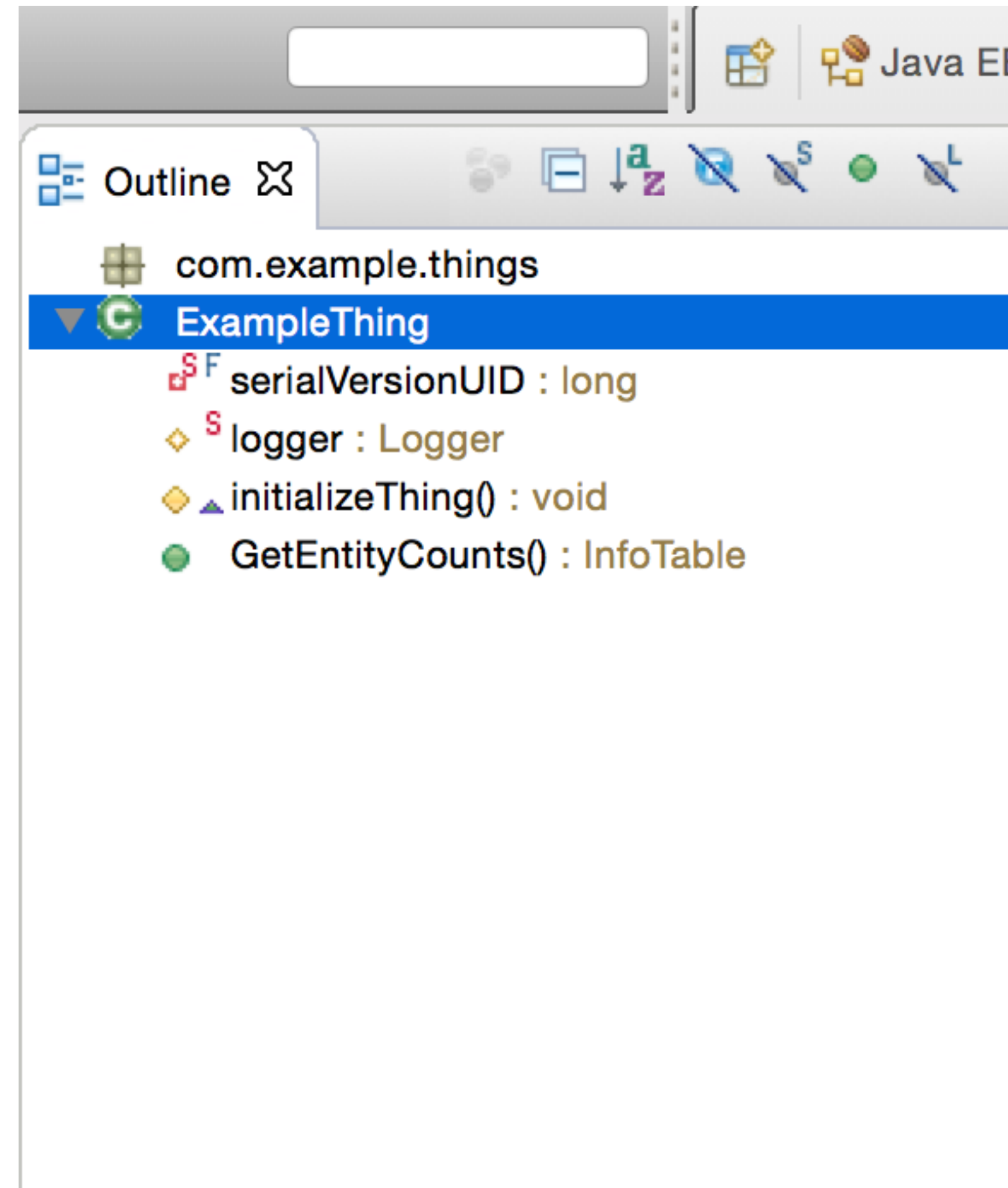
```
19 public class ExampleThing extends Thing {
20     private static final long serialVersionUID = -734037441678274101L;
21
22     protected static Logger logger = LogUtilities.getInstance().getAppli
23
24     @Override
25     protected void initializeThing() throws Exception {
26         super.initializeThing();
27         // Custom initialization
28     }
29
30     @ThingworxServiceDefinition(
31         name = "GetEntityCounts",
32         description = "Returns an InfoTable of entity counts sorted
33     @ThingworxServiceResult(
34         name = "result",
35         description = "An InfoTable (EntityCount) of entity counts.'
```

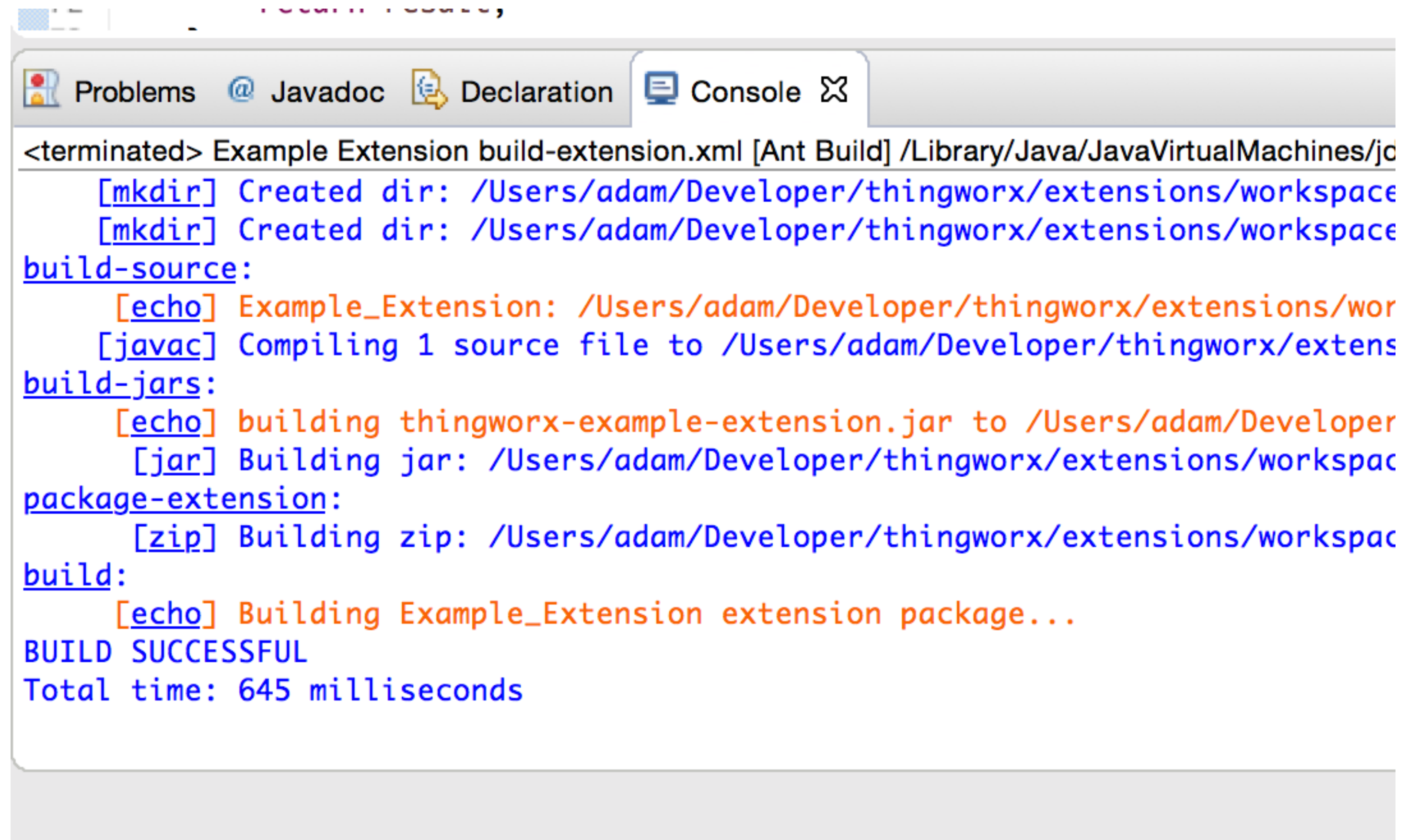
Editor

The Java editor provides features for editing Java code.

Outline

The Outline view displays an outline of the structure of the currently-active Java file in the editor.





```
<terminated> Example Extension build-extension.xml [Ant Build] /Library/Java/JavaVirtualMachines/jc
  [mkdir] Created dir: /Users/adam/Developer/thingworx/extensions/workspace
  [mkdir] Created dir: /Users/adam/Developer/thingworx/extensions/workspace
build-source:
  [echo] Example_Extension: /Users/adam/Developer/thingworx/extensions/wor
  [javac] Compiling 1 source file to /Users/adam/Developer/thingworx/extens
build-jars:
  [echo] building thingworx-example-extension.jar to /Users/adam/Developer
  [jar] Building jar: /Users/adam/Developer/thingworx/extensions/workspac
package-extension:
  [zip] Building zip: /Users/adam/Developer/thingworx/extensions/workspac
build:
  [echo] Building Example_Extension extension package...
BUILD SUCCESSFUL
Total time: 645 milliseconds
```

Console

The Console view displays a variety of console types.

Eclipse

Extension Development

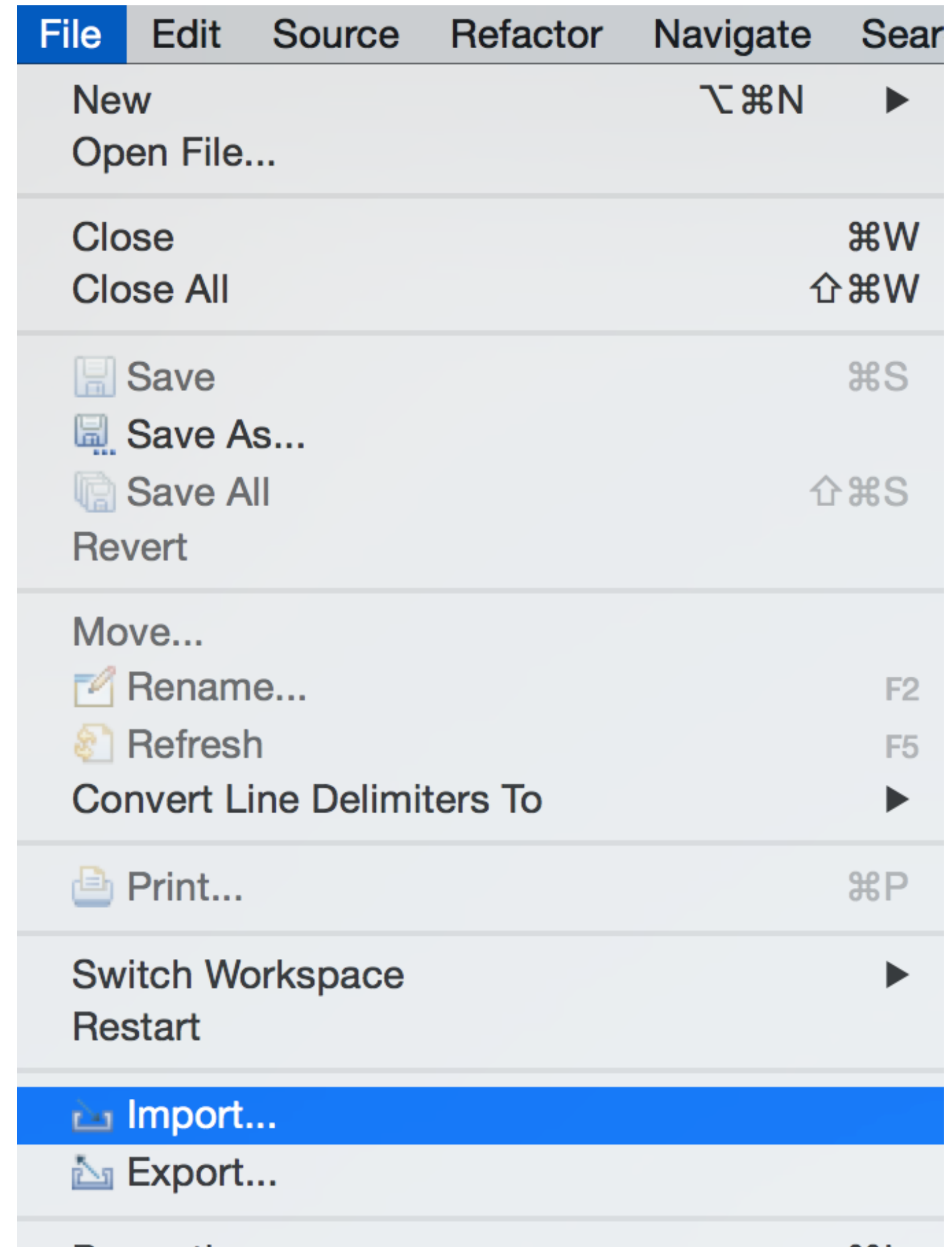
Projects

A Java project contains source code and related files for building a Java program.

It maintains a model of its contents including information about the type hierarchy, references and declarations of Java elements.

Importing Projects

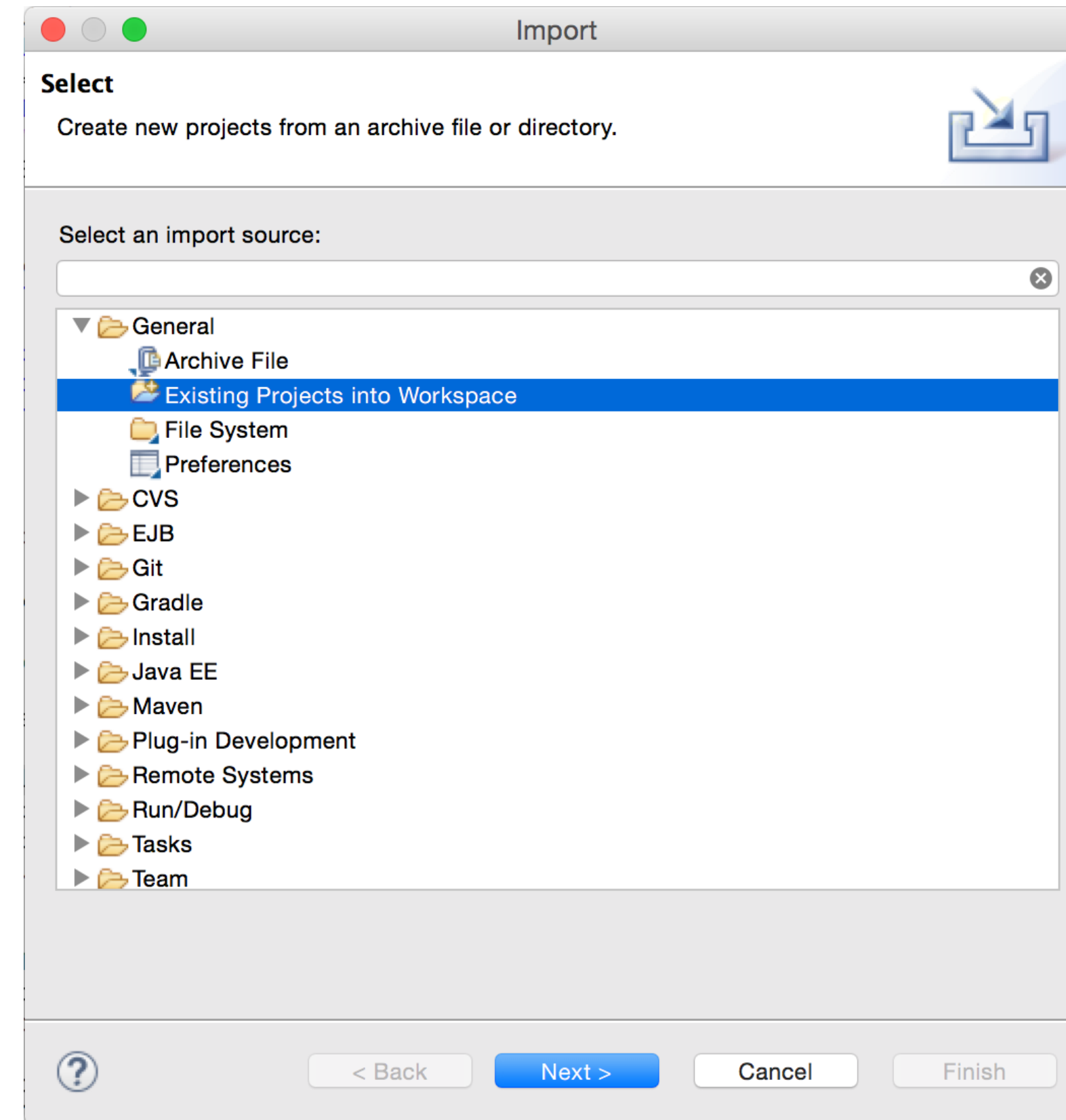
Choose **File** > **Import....**



Importing Projects

Select **Existing Projects into Workspace**.

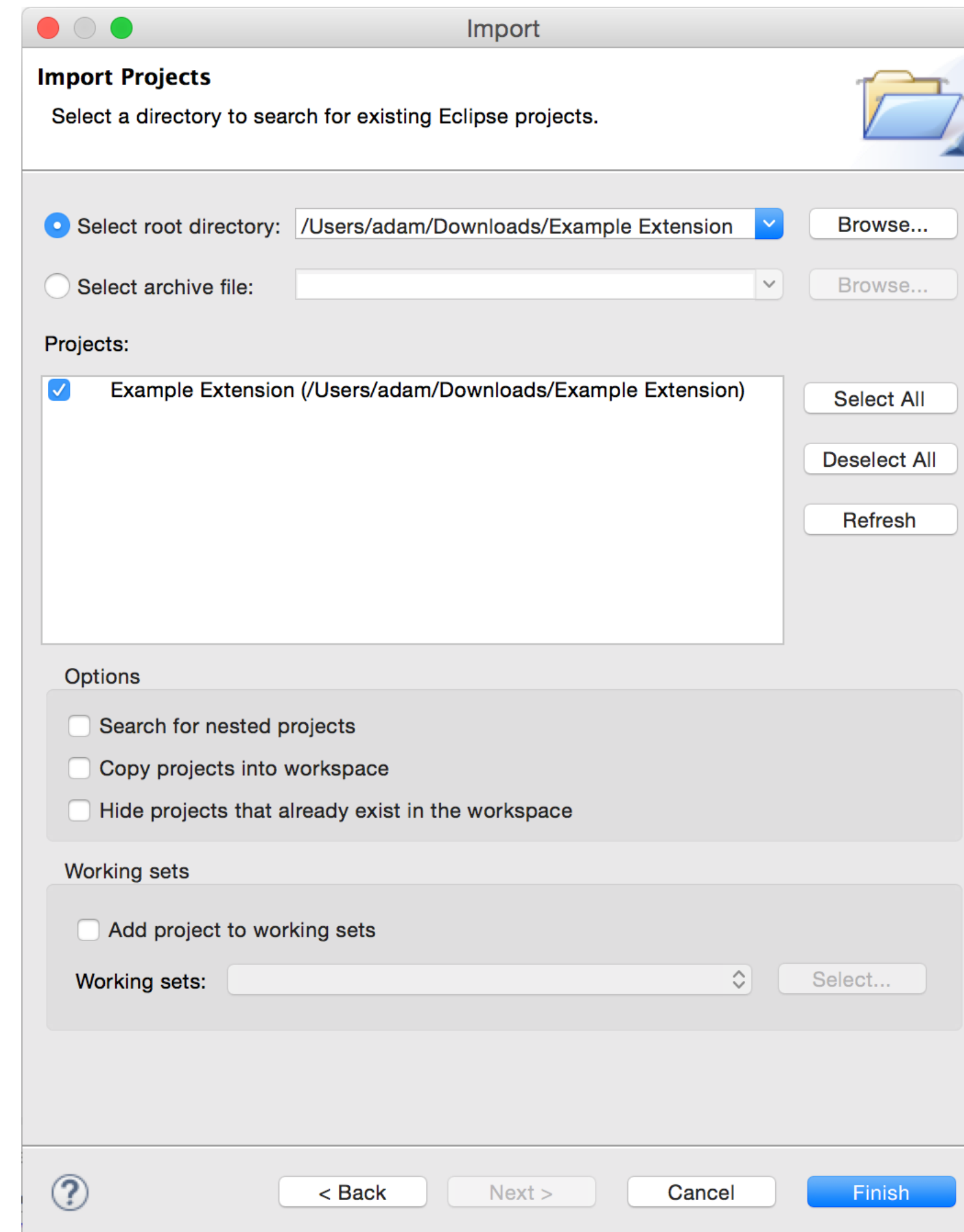
Click **Next >**.



Importing Projects

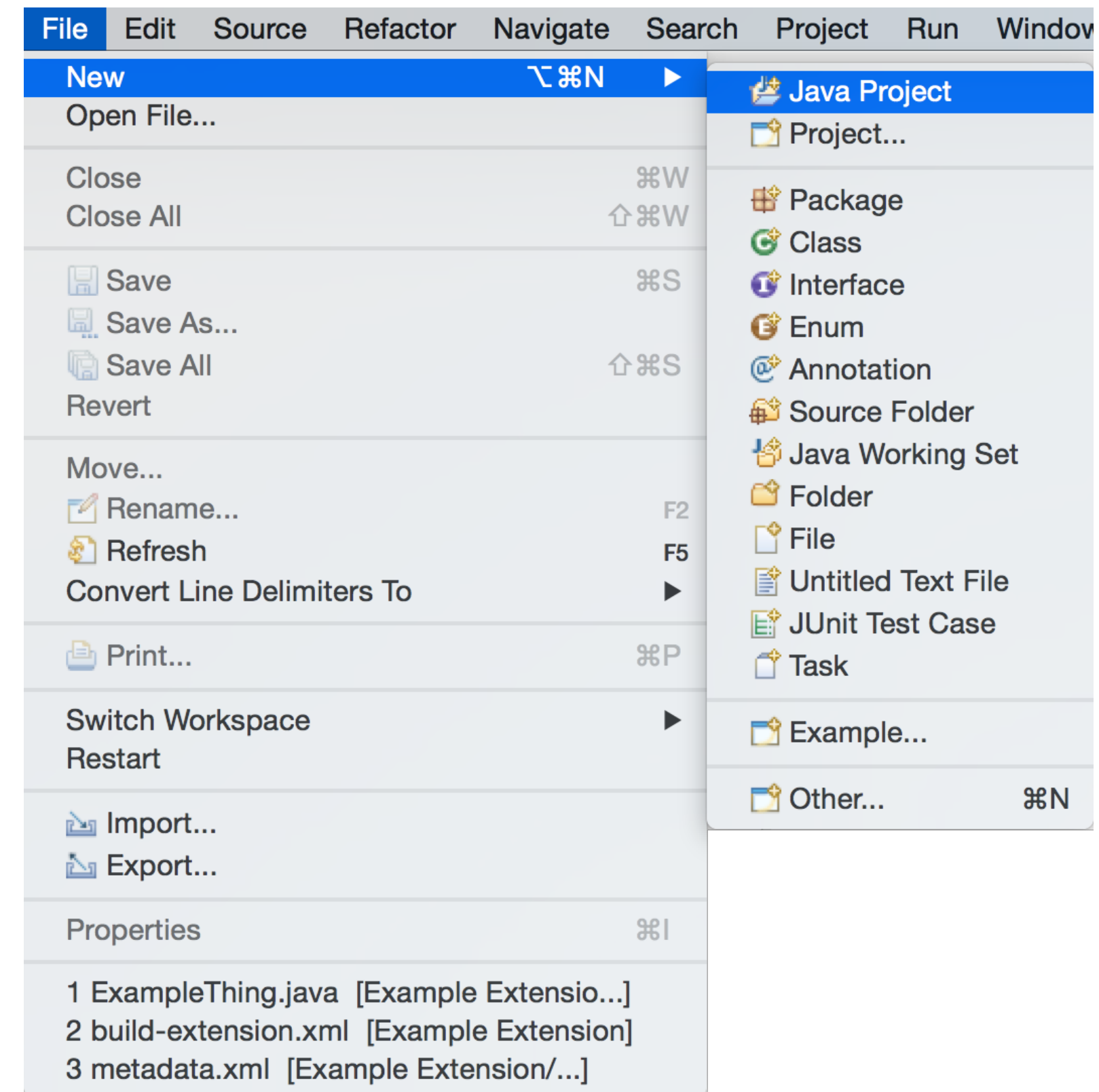
Choose **Select root directory** or **Select archive file** and click the associated **Browse** to locate the directory or file containing the project.

Click **Finish**.



Creating Projects

Choose **File** > **New** > **Java Project**.



Creating Projects

Click **Finish**.

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

Use default location

Location:

JRE

Use an execution environment JRE:

Use a project specific JRE:

Use default JRE (currently 'Java SE 7 [1.7.0_79]') [Configure JREs...](#)

Project layout

Use project folder as root for sources and class files

Create separate folders for sources and class files [Configure default...](#)

Working sets

Add project to working sets

Working sets:

i The default compiler compliance level for the current workspace is 1.8. The new project will use a project specific compiler compliance level of 1.7.

Project Structure

configfiles/metadata.xml

ThingWorx entities

lib/

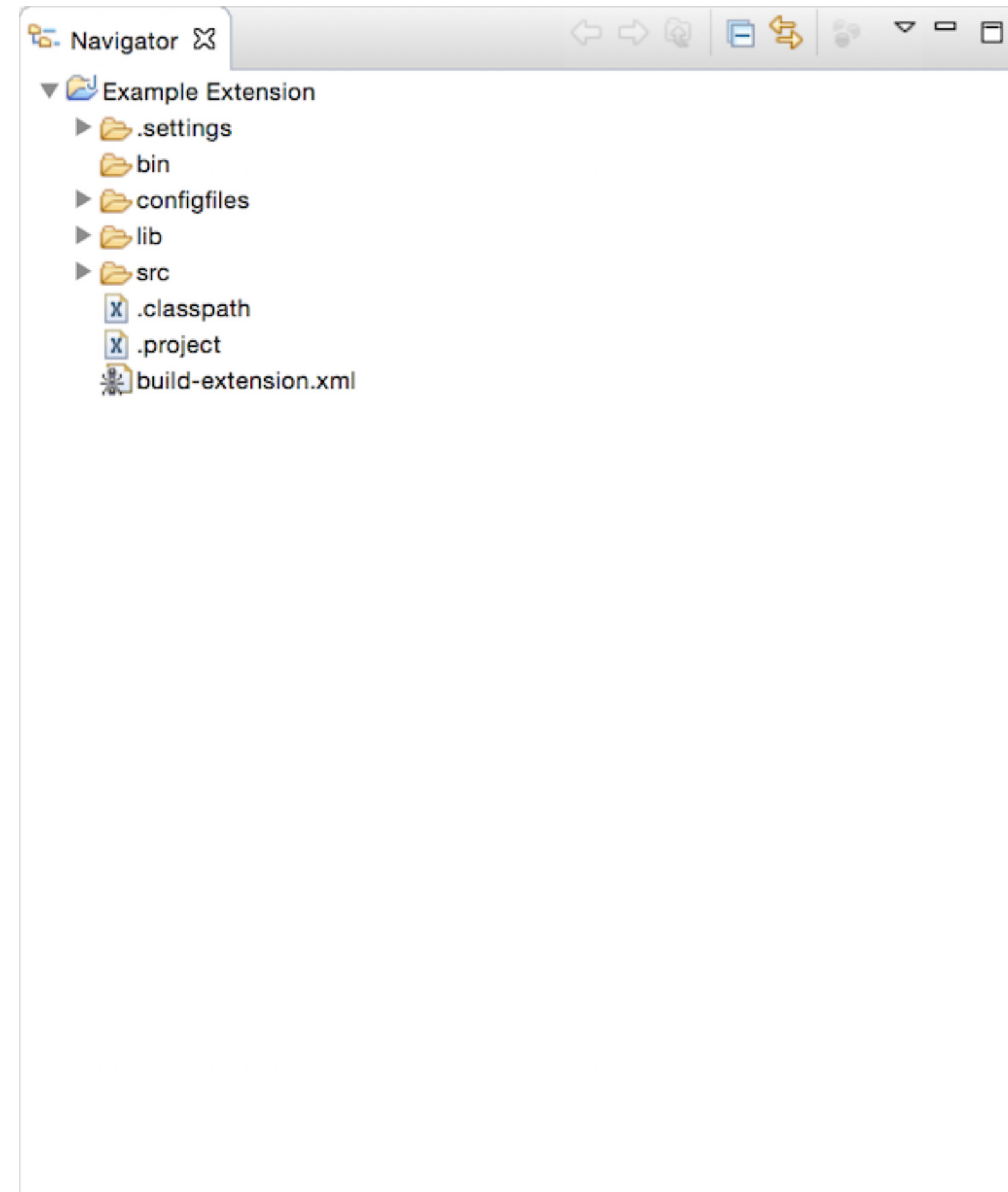
JAR libraries

src/

Java source code

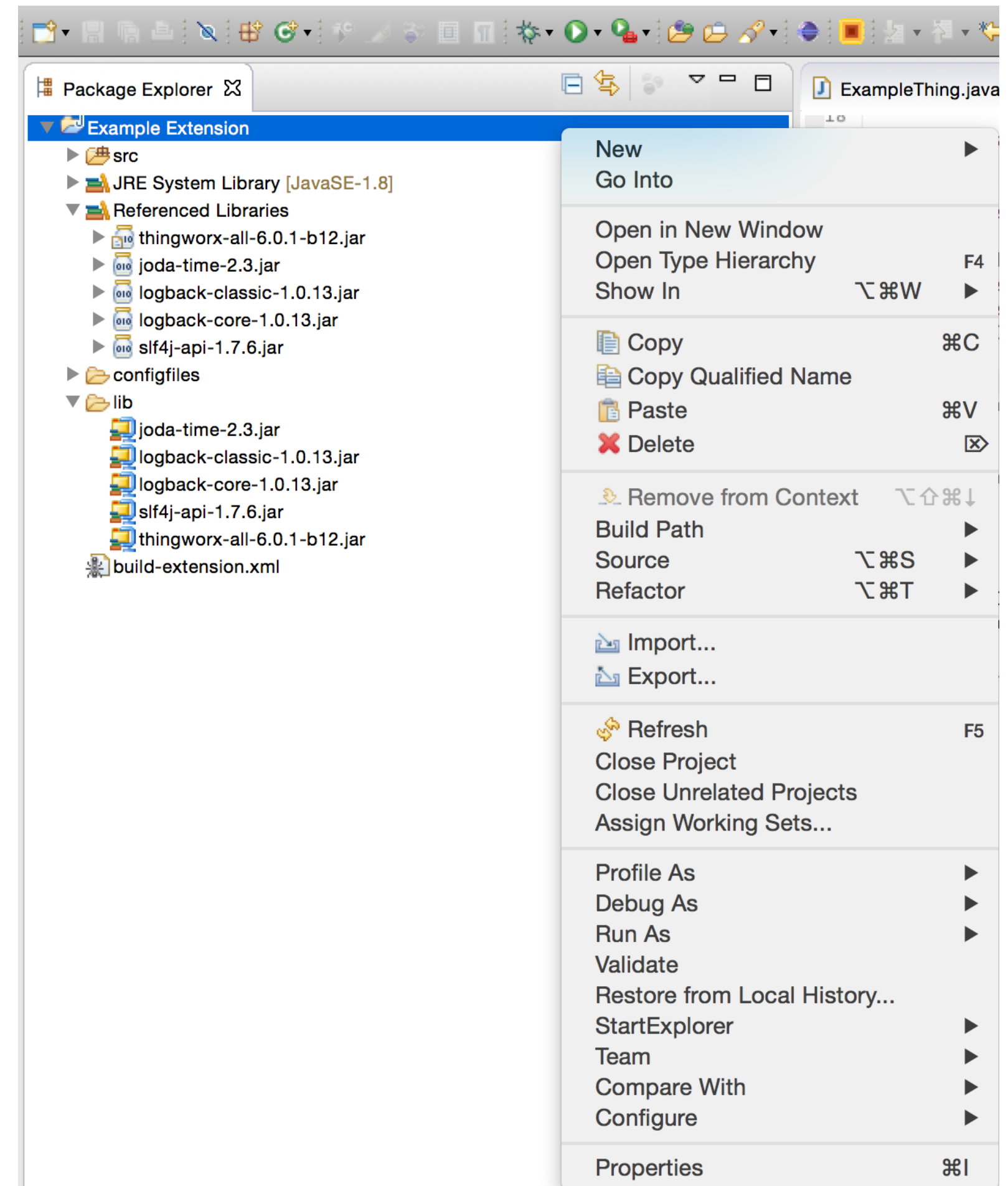
build-extension.xml

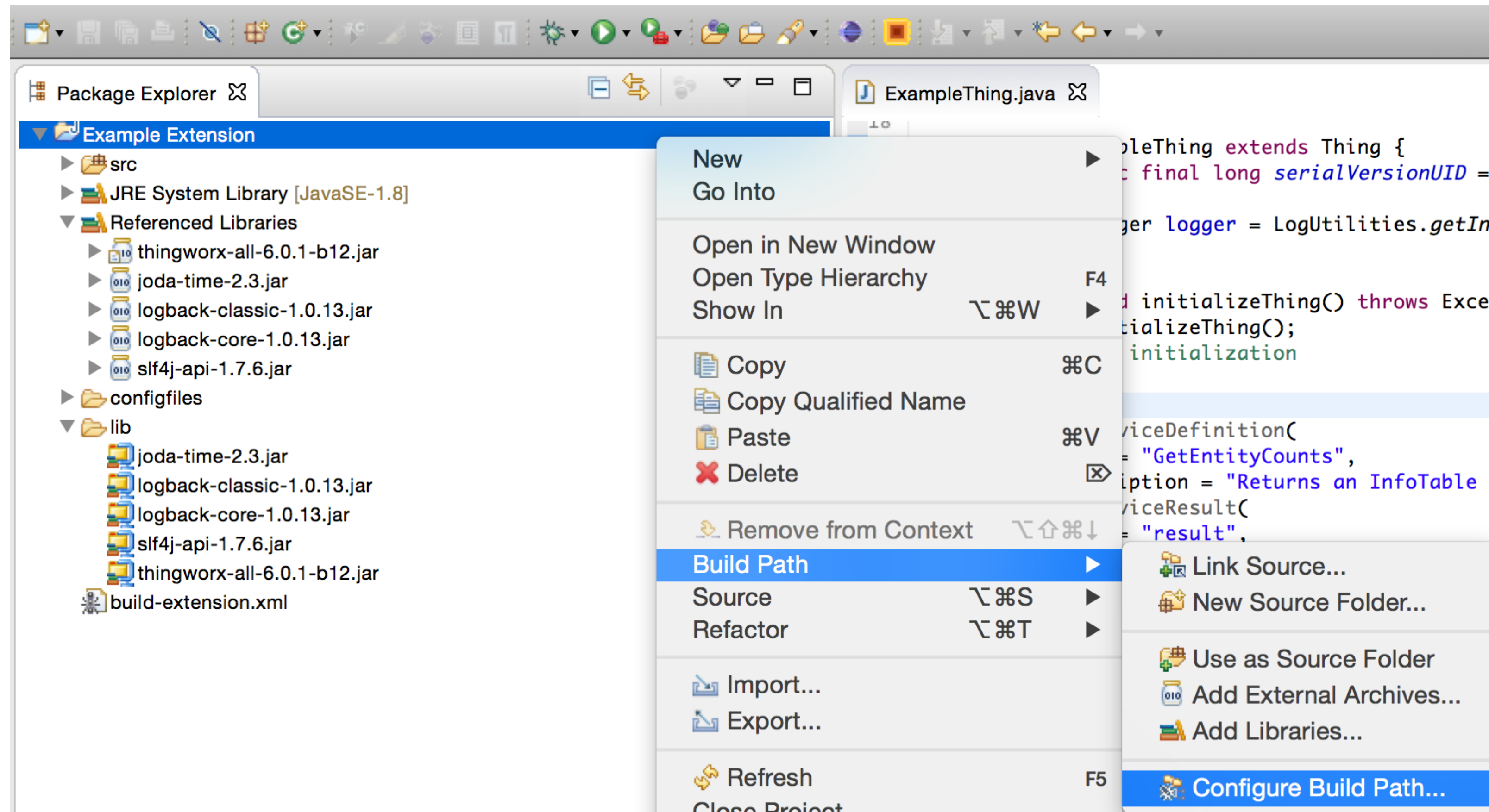
Ant build file



Configuring the Build Path

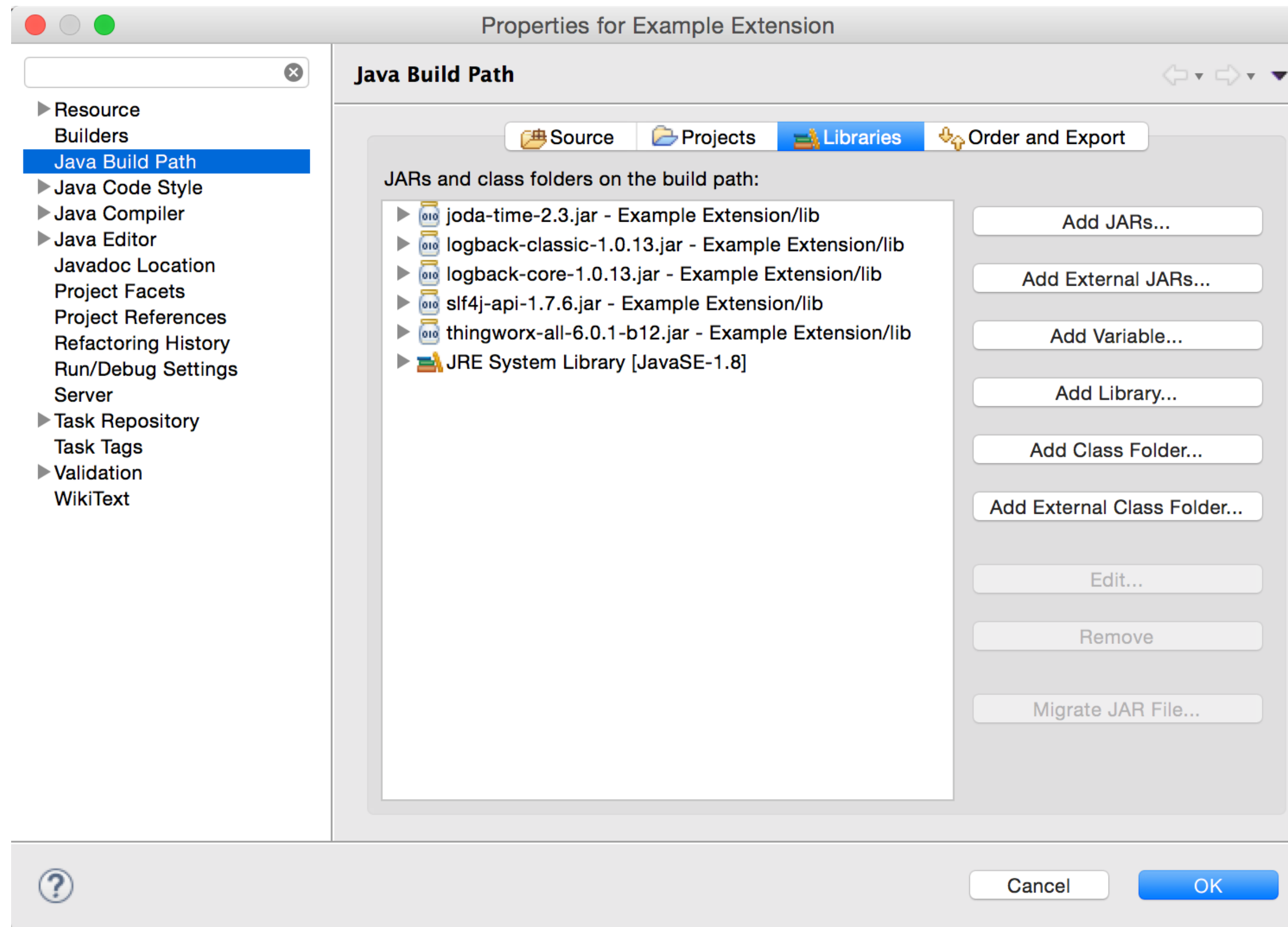
Right-click the project in the Package Explorer.





Configuring the Build Path

Choose **Build Path** > **Configure Build Path....**



Configuring the Build Path

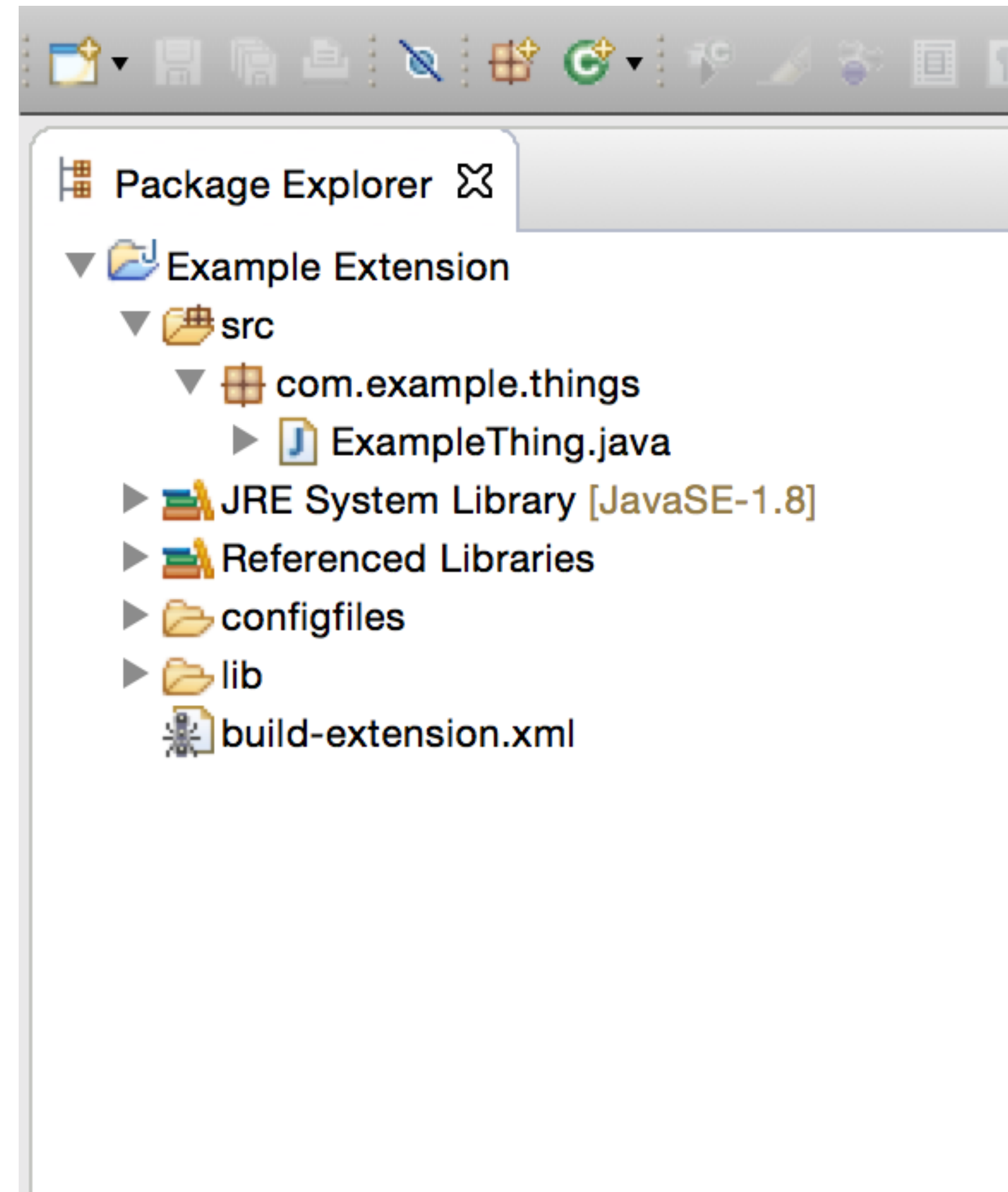
Click **Add JARS...** to add JARs to the build path.

Packages

A Java package is a mechanism for organizing Java classes into namespaces.

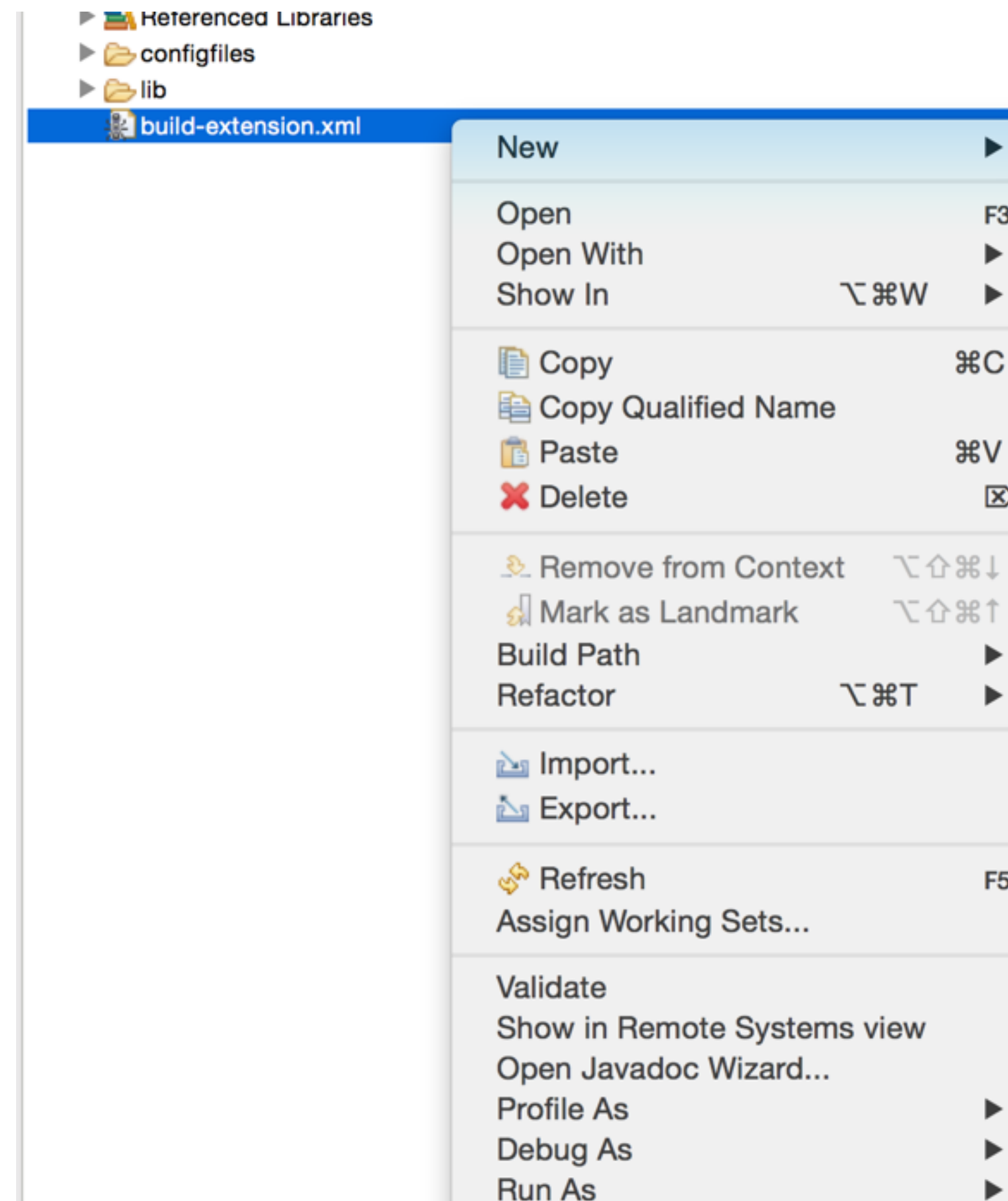
Packages correspond to the location of Java source files in the filesystem.

In a Java source file, the package of the class is specified using the **package** keyword.



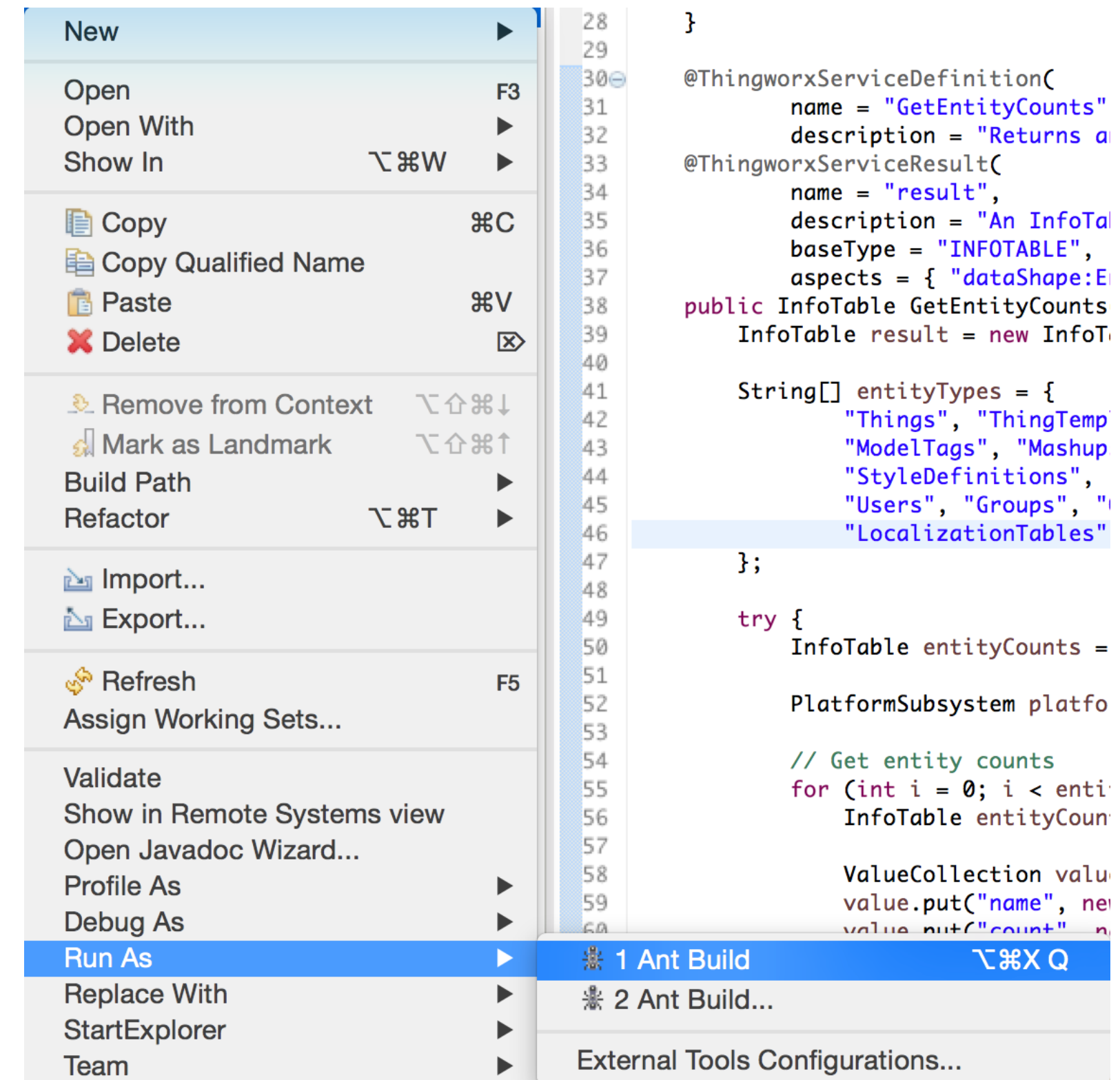
Building Projects

Right-click *build-extension.xml*.



Building Projects

Choose **Run As > 1 Ant Build.**



The screenshot shows an IDE interface. On the left, a context menu is open for the 'Run As' option. The menu items are: 'New', 'Open', 'Open With', 'Show In', 'Copy', 'Copy Qualified Name', 'Paste', 'Delete', 'Remove from Context', 'Mark as Landmark', 'Build Path', 'Refactor', 'Import...', 'Export...', 'Refresh', 'Assign Working Sets...', 'Validate', 'Show in Remote Systems view', 'Open Javadoc Wizard...', 'Profile As', 'Debug As', 'Run As', 'Replace With', 'Start Explorer', and 'Team'. The 'Run As' option is selected, and its sub-menu is visible, containing '1 Ant Build', '2 Ant Build...', and 'External Tools Configurations...'. The '1 Ant Build' option is highlighted. On the right, a snippet of Java code is visible, showing a class with annotations and a method:

```
28 }
29
30 @ThingworxServiceDefinition(
31     name = "GetEntityCounts"
32     description = "Returns a
33 @ThingworxServiceResult(
34     name = "result",
35     description = "An InfoTable
36     baseType = "INFOTABLE",
37     aspects = { "dataShape:Entity
38 public InfoTable GetEntityCounts
39     InfoTable result = new InfoTable
40
41     String[] entityTypees = {
42         "Things", "ThingTemplate
43         "ModelTags", "Mashup
44         "StyleDefinitions",
45         "Users", "Groups", "
46         "LocalizationTables"
47     };
48
49     try {
50         InfoTable entityCounts =
51         PlatformSubsystem platfo
52
53         // Get entity counts
54         for (int i = 0; i < enti
55         InfoTable entityCount
56
57         ValueCollection valu
58         value.put("name", ne
59         value.put("count", n
```

Exercise 1

1. Using the provided resources, create a Java project and structure it properly.
2. Build the extension.
3. Import the extension.

Remote Debugging

Configuring Tomcat

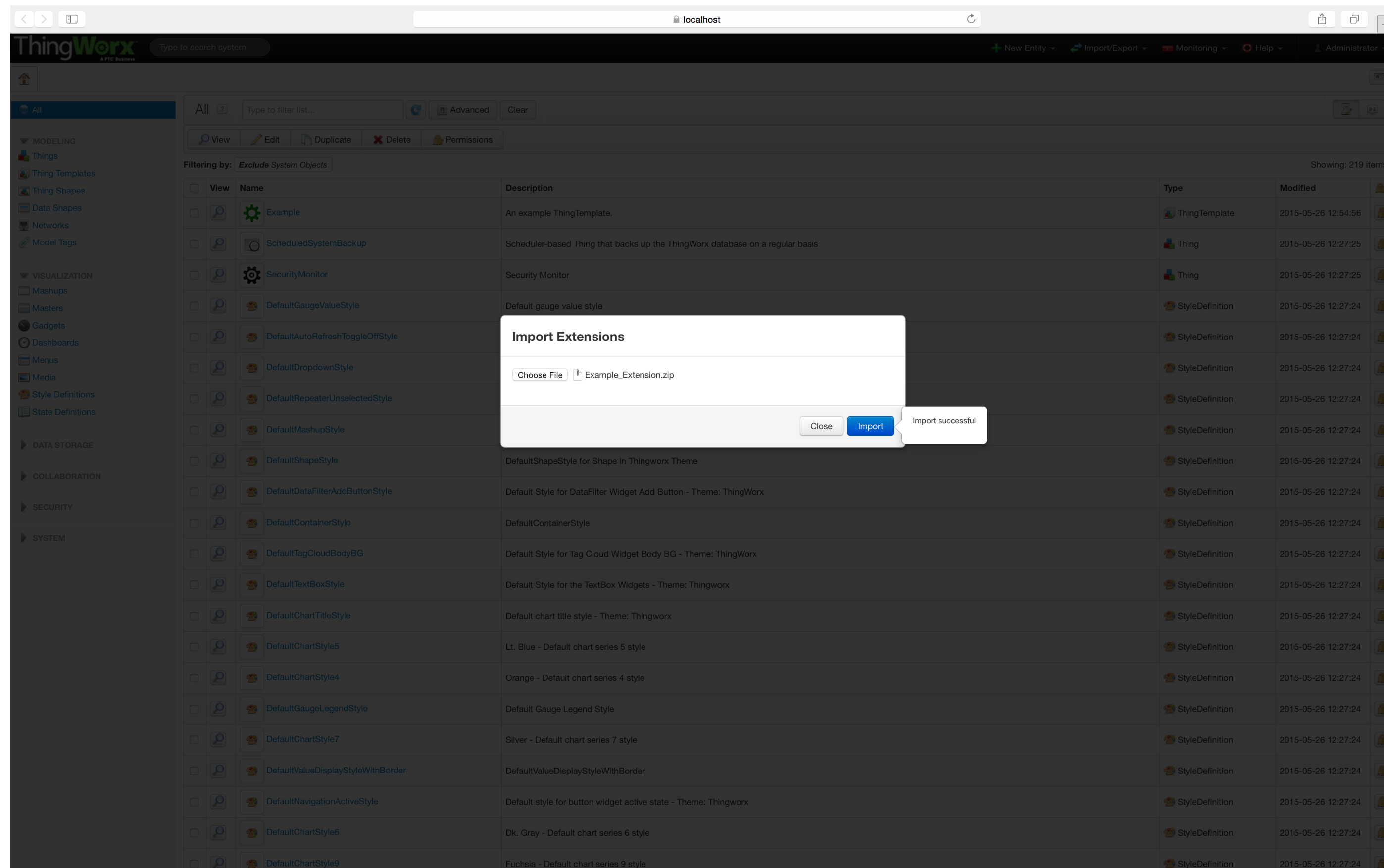
Add the following options:

```
-Xdebug -Xrunjdp:transport=dt_socket,address=8000,server=y,suspend=n
```

or

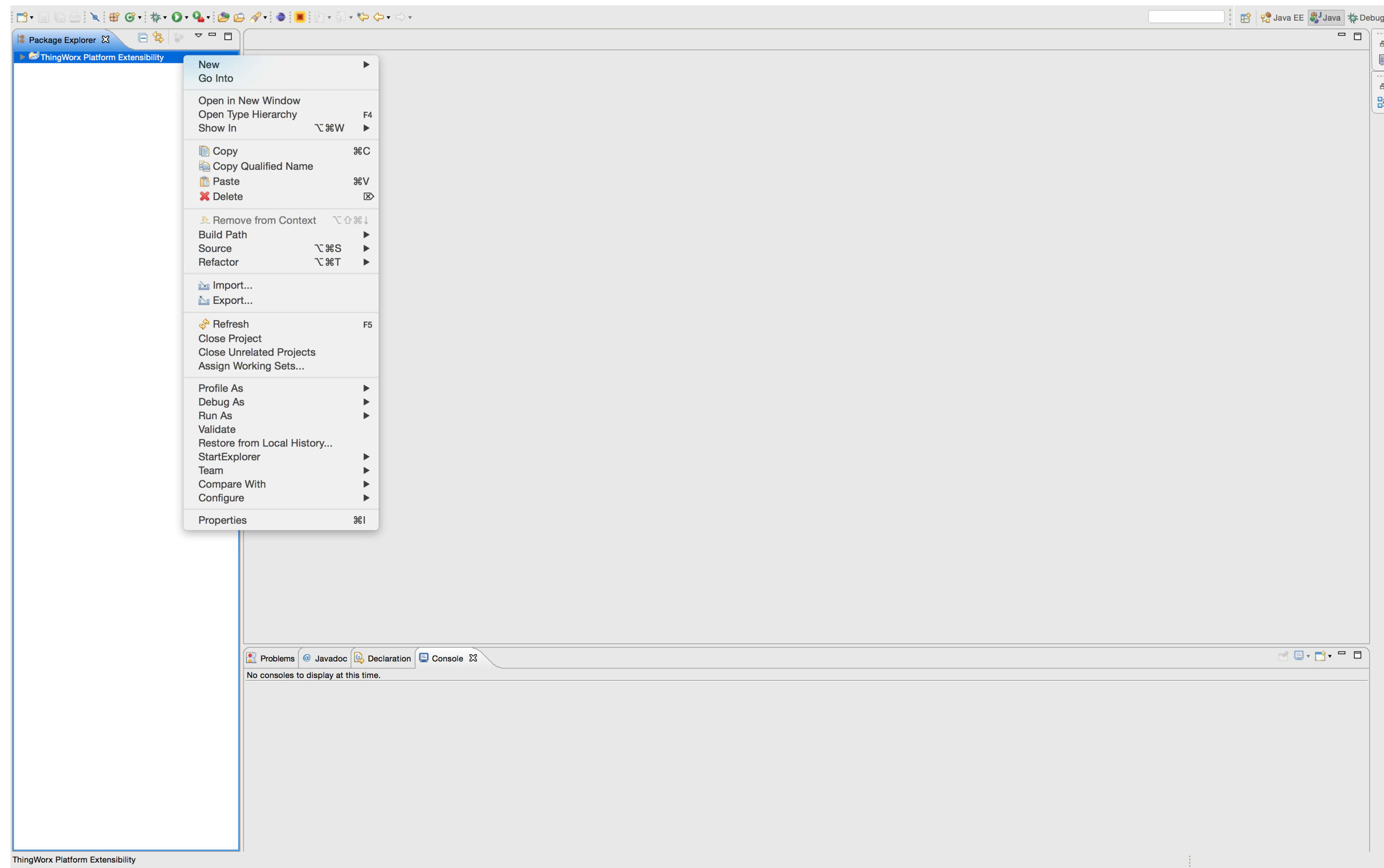
Start Tomcat with the following command:

```
catalina jpda start
```



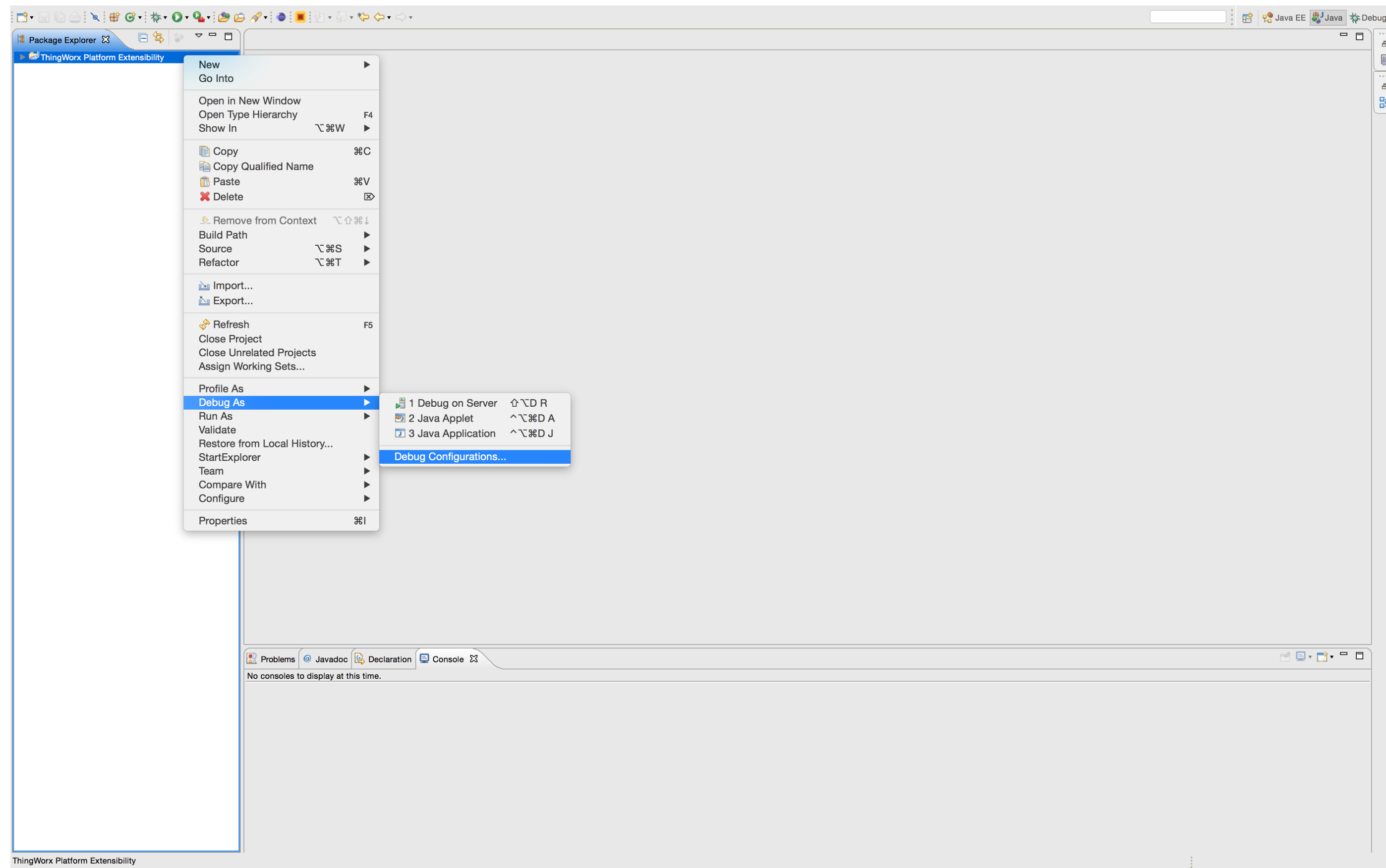
Configuring ThingWorx

Build and import the extension.



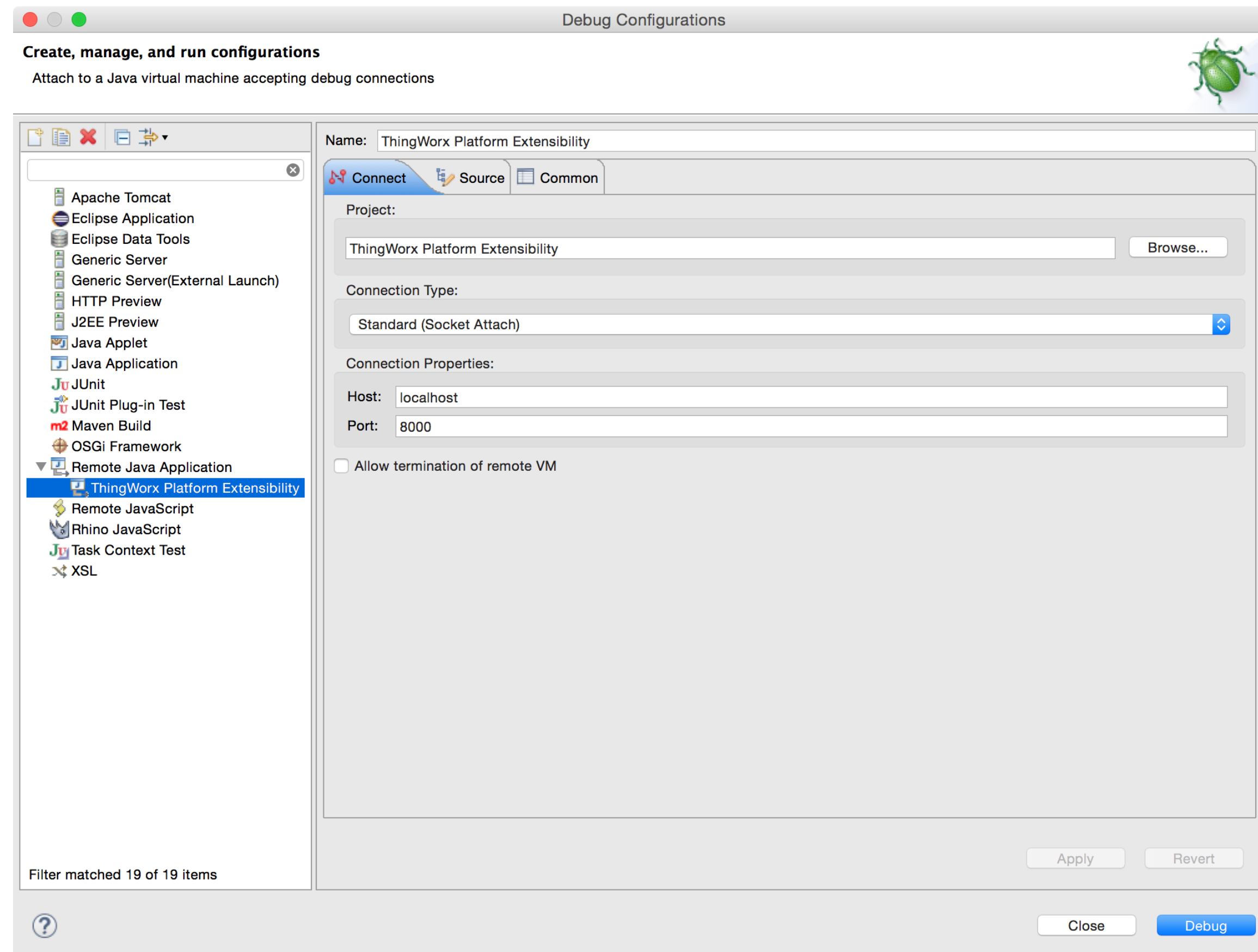
Configuring Eclipse

Right-click the project in the Package Explorer.



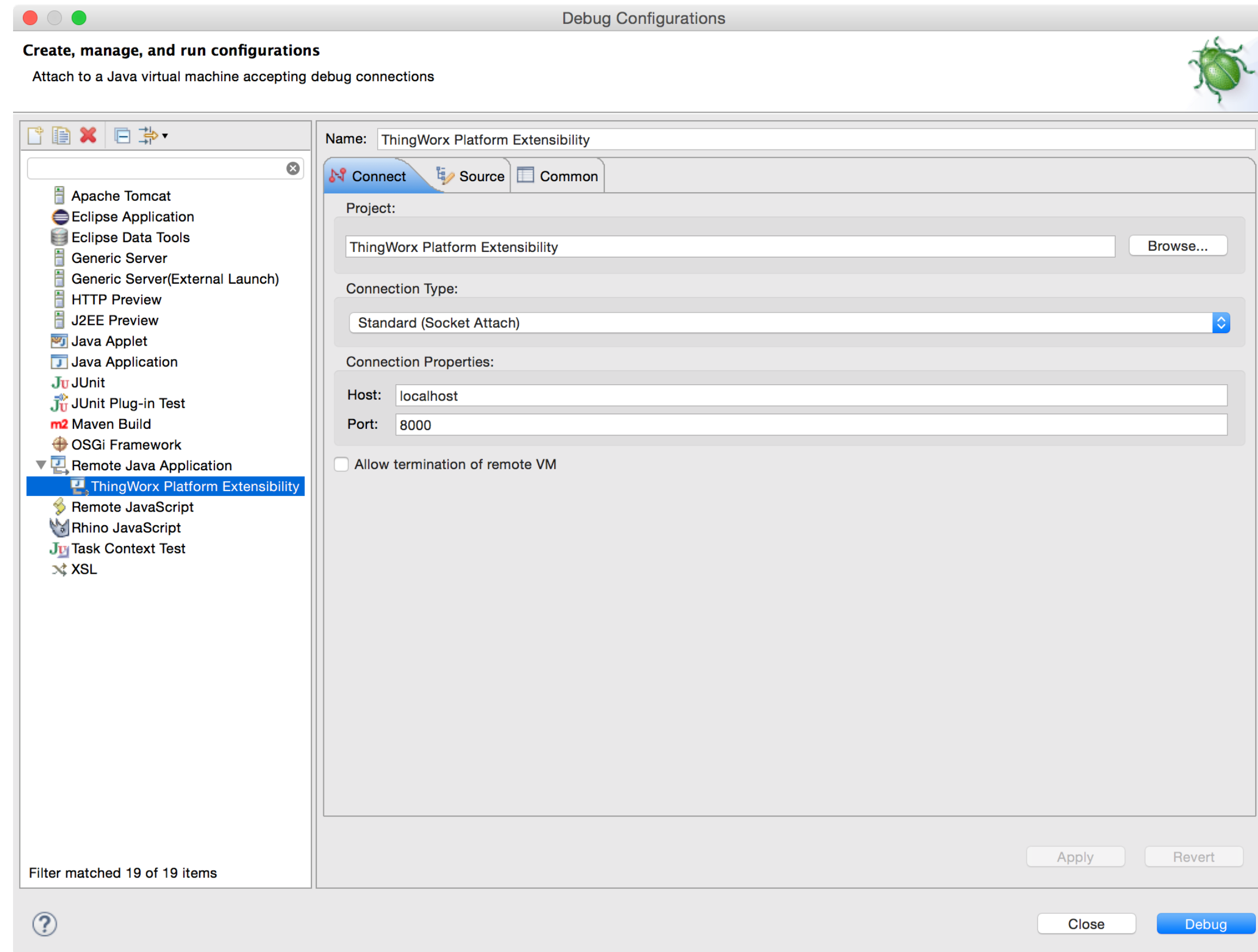
Configuring Eclipse

Choose Debug As > Debug Configurations....



Configuring Eclipse

Double-click Remote Java Application.



Configuring Eclipse

Click Debug.

Package Explorer

- Example Extension
 - src
 - JRE System Library [JavaSE-1.8]
 - Referenced Libraries
 - configfiles
 - lib
 - build-extension.xml

```

18
19 public class ExampleThing extends Thing {
20     private static final long serialVersionUID = -734037441678274101L;
21
22     protected Logger logger = LogUtilities.getInstance().getApplicationLogger(this.getClass());
23
24     @Override
25     protected void initializeThing() throws Exception {
26         super.initializeThing();
27         // Custom initialization
28     }
29
30     @ThingworxServiceDefinition(
31         name = "GetEntityCounts",
32         description = "Returns an InfoTable of entity counts sorted in descending order..")
33     @ThingworxServiceResult(
34         name = "result",
35         description = "An InfoTable (EntityCount) of entity counts.",
36         baseType = "INFOTABLE",
37         aspects = { "dataShape:EntityCount" })
38     public InfoTable GetEntityCounts() throws Exception {
39         InfoTable result = new InfoTable();
40
41         String[] entityTypees = {
42             "Things", "ThingTemplates", "ThingShapes", "DataShapes", "Networks",
43             "ModelTags", "Mashups", "Dashboards", "Menus", "MediaEntities",
44             "StyleDef",
45             "Users",
46             "Localiza
47         };
48
49         try {
50             InfoTable ent
51
52             PlatformSubsy
53
54             // Get entity
55             for (int i =
56                 InfoTable
57
58             ValueColl
59             value.put("name", new StringPrimitive(entityCount.getRow(0).getValue("name").toString()));
60             value.put("count", new IntegerPrimitive(Integer.parseInt(entityCount.getRow(0).getValue("count").toString())));
61             entityCounts.addRow(value);
62         }
63
64         // Sort entityCounts in descending order
65         InfoTableFunctions infoTableFunctions = (InfoTableFunctions) ResourceManager.getInstance().getEntity("InfoTableFunctions");
66         result = infoTableFunctions.Sort(entityCounts, "count", false);
67     } catch (Exception e) {
68         logger.error("null");
69     }
70
71     return result;
72

```

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

Remember my decision

No Yes

Outline

- com.example.things
 - ExampleThing
 - serialVersionUID : long
 - logger : Logger
 - initializeThing() : void
 - GetEntityCounts() : InfoTable

Problems Javadoc Declaration Console

No consoles to display at this time.



Debug

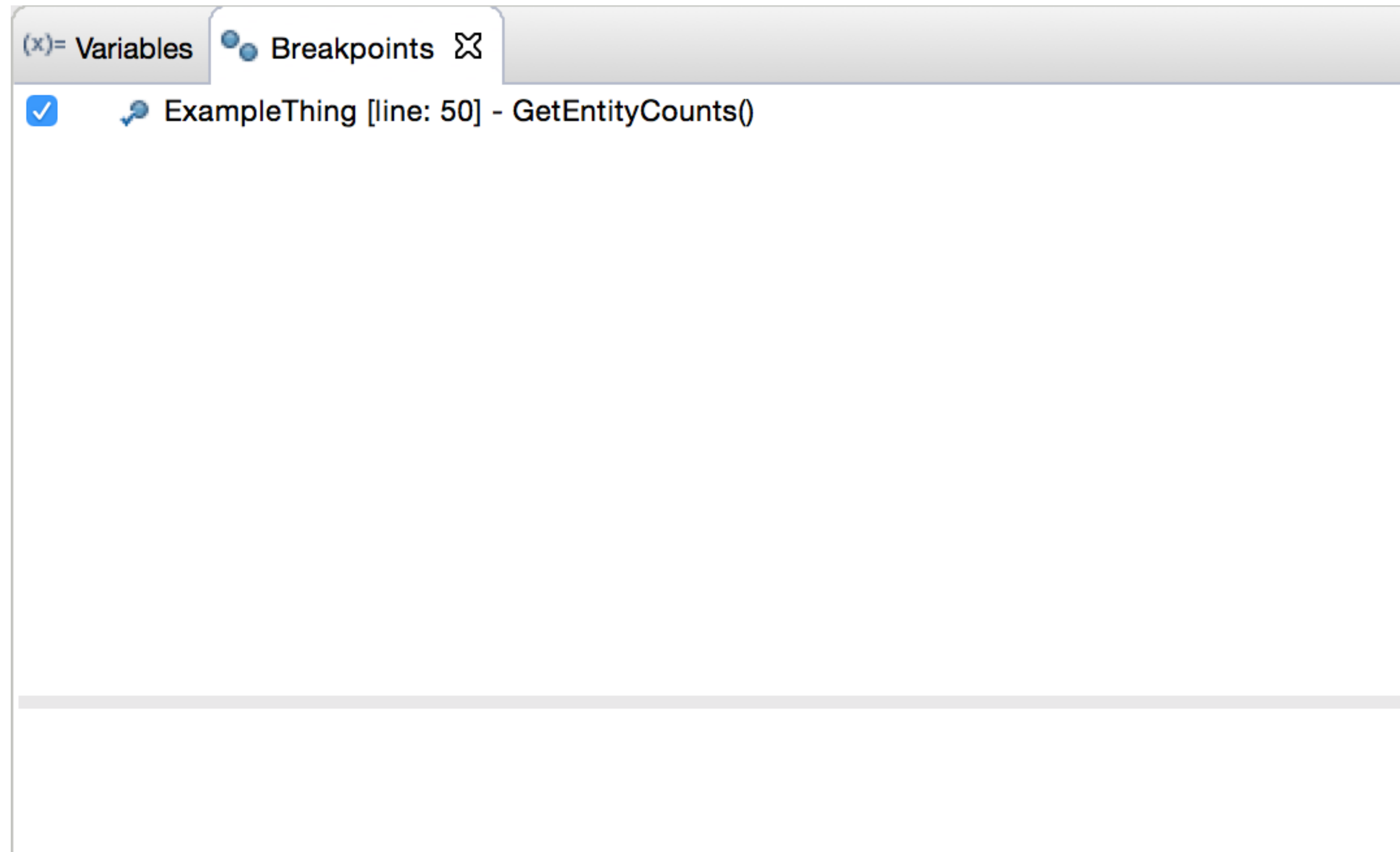
The Debug view displays the stack frame for the suspended threads for each target you are debugging.

(x) Variables ✕ Breakpoints

| Name | Value |
|-----------------------|---------------|
| ▶ ● this | ExampleThi |
| ▶ ⓘ result | InfoTable (ic |
| ▶ ⓘ entityType | String[23] (i |
| ▶ ⓘ entityCounts | InfoTable (ic |
| ▶ ⓘ platformSubsystem | PlatformSub |
| ▶ ⓘ i | 1 |
| ▶ ⓘ entityCount | InfoTable (ic |
| ▶ ⓘ value | ValueCollec |
| | |
| | |
| | |
| | |

Variables

The Variables view displays information about the variables associated with the stack frame selected in the Debug view.



Breakpoints

The Breakpoints view lists all of the breakpoints set in the workspace.

Step Controls

Step over

The currently-selected line to be executed is invoked and suspends on the next line.

Step into

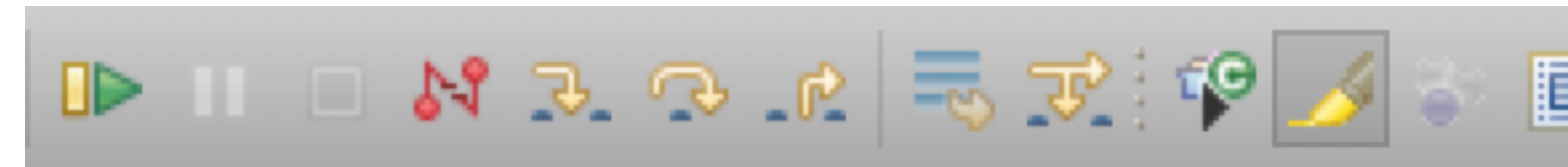
The next expression on the currently-selected line to be executed is invoked and execution suspends on the next line.

Step return

Resume execution until the next return statement in the current method and execution suspends on the next line.

Resume

Resume the execution of suspended threads.



```
rs
hread [http-nio-8443-exec-8] (Suspended (breakpoint at
ecureNioChannel (id=95)
eThing.GetEntityCounts() line: 50
methodAccessorImpl.invoke0(Method, Object, Object[]) li
methodAccessorImpl.invoke(Object, Object[]) line: 62
ingMethodAccessorImpl.invoke(Object, Object[]) line: 43
.invoke(Object, Object...) line: 497
onProcessor.processService(Object, ValueCollection) lin
onServiceHandler.processService(IServiceProvider, Value
eThing(Thing).processServiceRequestDirect(ServiceDefir
eThing(Thing).processAPIServiceRequest(String, ValueC
BaseService).handleInvoke(HttpExecutionContext) line: 2
BaseService).service(HttpServletRequest, HttpServletRe
HttpServletRequest).service(ServletRequest, ServletResponse) li
tionFilterChain.internalDoFilter(ServletRequest, ServletR
tionFilterChain.doFilter(ServletRequest, ServletResponse
doFilter(ServletRequest, ServletResponse, FilterChain)
```

Exercise 2

1. A logic error exists in ExampleThing.java. Step through the code to find it.
2. After you've located the bug, fix it.
3. Re-build the project and import it.
4. Verify the extension works properly.