

Servlet Example

SE - Introduction	1
SE - Java API Workspace preparation	1
SE - Import of the example	1
SE - Hints to develop your own plug-in	1
SE - Deployment to Installed Polaron	4
SE - Execution from Workspace	4
SE - Configuration	4

See also main SDK page.

SE - Introduction

This example allows you to create an extension for Wiki pages in form of creating a custom servlet to inform users, e.g. about statistics at the Home or Dashboard. The result will be your own servlet with a title and body represented by '.jsp' page (written by you) embedded into a Wiki page.

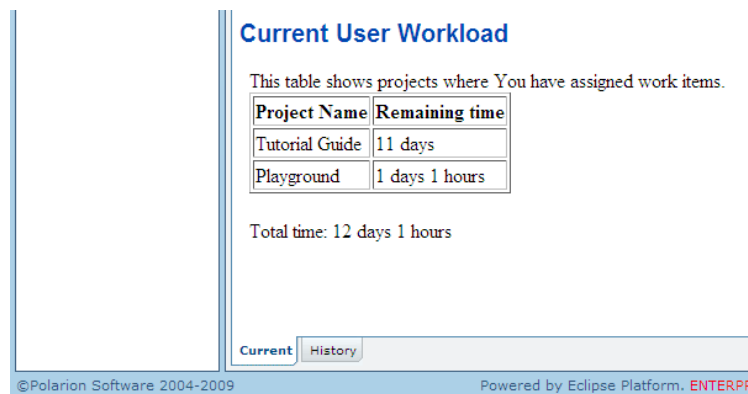


Figure SE-1: Expected result

SE - Java API Workspace preparation

See section *Workspace preparation*

SE - Import of the example

Info: You must ensure that your plugin is compiled against your Polaron version. This example contains precompiled jar plugin. You can remove it before you start developing your plugin based on this example. The Eclipse ensure that new jar plugin will be created against your source code and Polaron version.

To import workflow project example to workspace, do the following steps:

1. Select **File > Import...**
2. In the dialog that appears, select **Existing Project into Workspace** in **General** section and press **Next** button.
3. By pressing **Browse..** button, select the directory of examples (mostly in `C:\Polarion\polarion\SDK\examples\`). Submit it.
4. Select `com.polarion.example.servlet` and press **Finish**.

SE - Hints to develop your own plug-in

1. First, we need to create new eclipse plugin: Select **File > New.. > Project**.
2. In the dialog that appears, select **Plug-in Project** and press **Next** button.

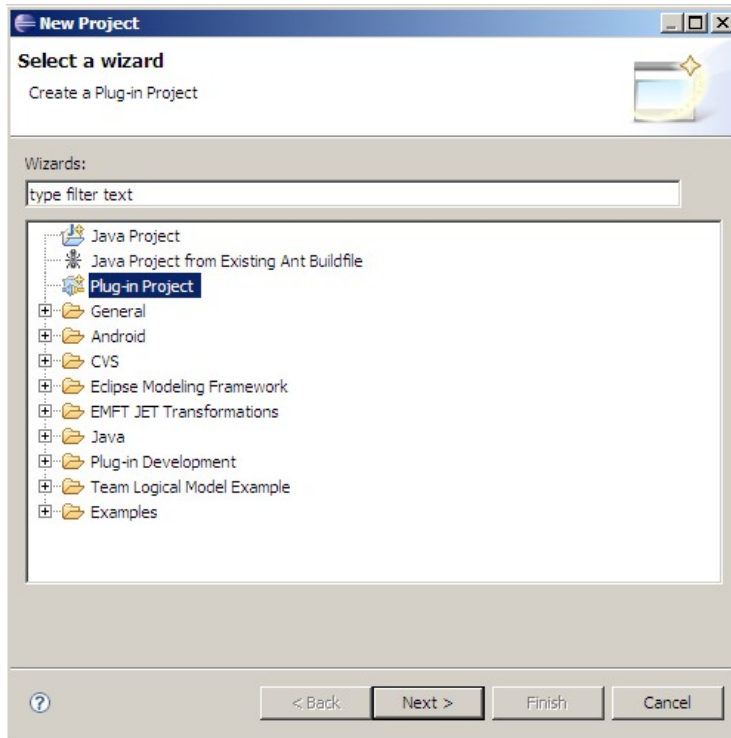


Figure SE-2: New Plug-in project

3. Set project name to e.g. `com.polarion.example.servlet`. Press **Next**.
4. Unselect **Generate an activator...** Press **Next**.

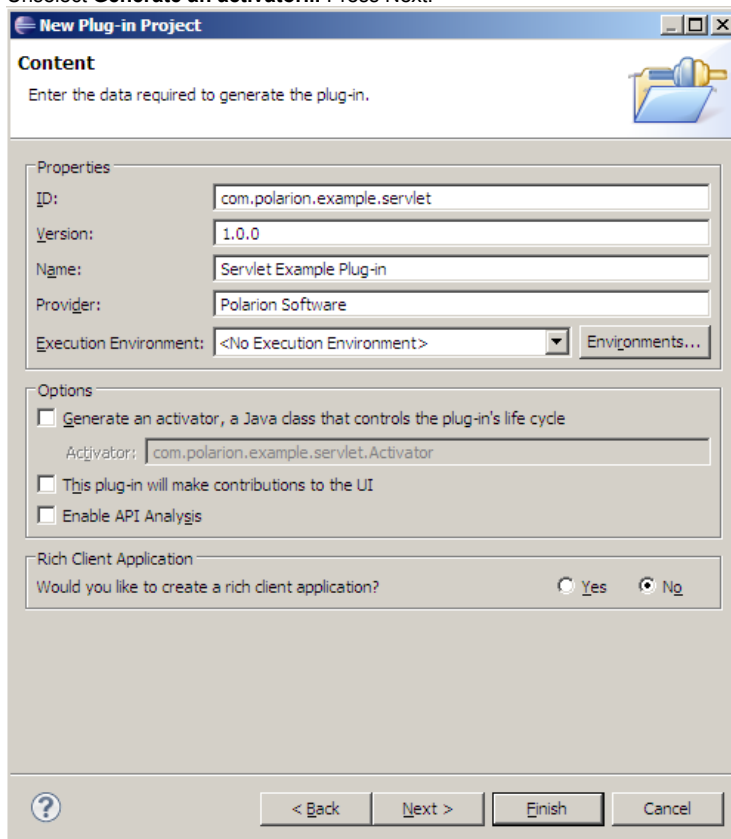


Figure SE-3: Third wizard's page

5. Press **Finish**.
6. Open `MANIFEST.MF` from the directory with the same name.
7. Click on the second page - **Dependencies** and click **Add..** button.
8. Type `com.polarion.portal.tomcat` and submit it.
9. Repeat previous two steps, but type `com.polarion.alm.tracker` and submit it.

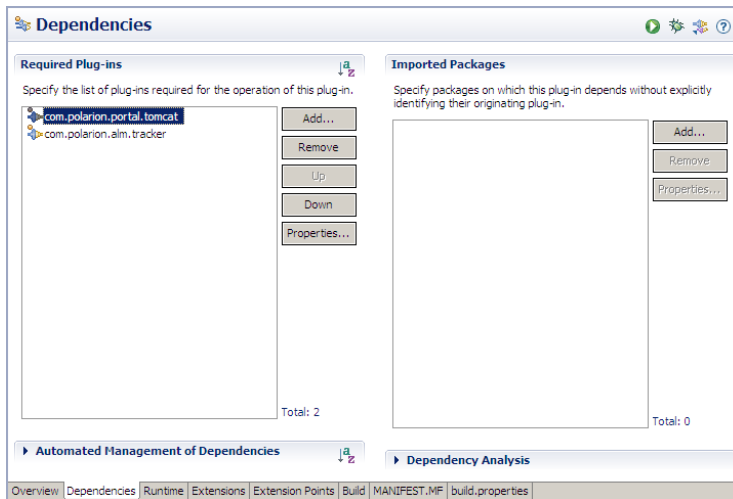


Figure SE-4: Dependencies page

10. On the next page click **New..** in the 'Classpath' corner, type `servlet.jar` and submit it.

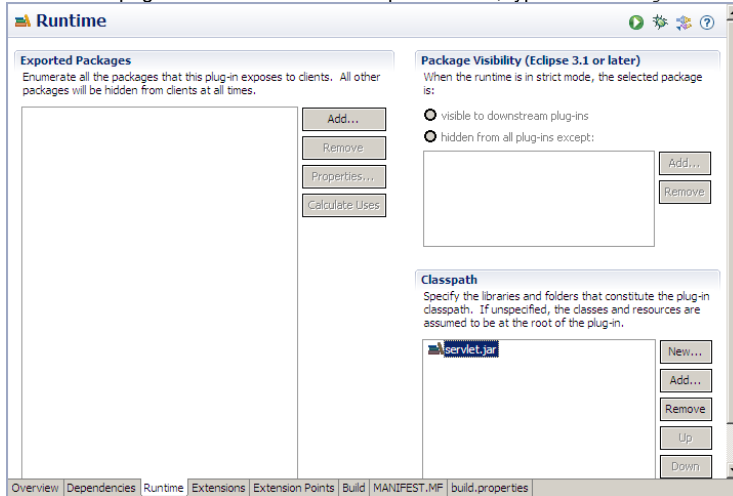


Figure SE-5: Runtime page

11. On the next page click **Add..** and choose `com.polarion.portal.tomcat.webapps` extension point and submit it.
12. In the Extension Element Details set name for new application, e.g. `example` and `contextRoot` set to `webapp`.

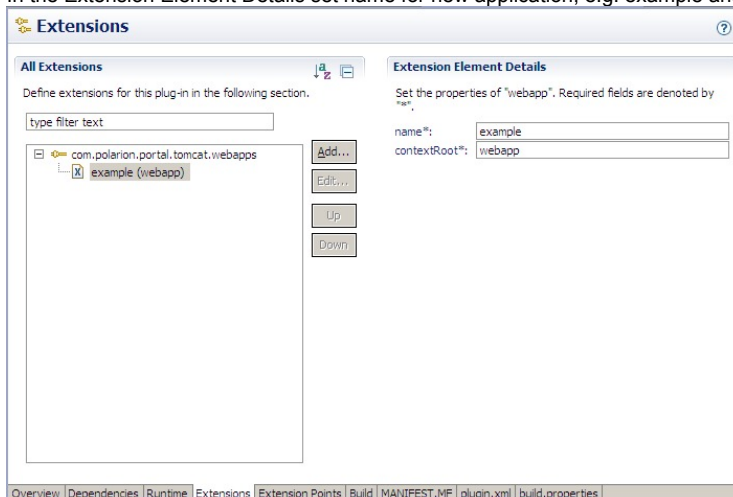


Figure SE-6: Extensions page

13. Select folders and files in **Binary Build** corner according to Figure SE-7. (If some folders are missing, You will select them later).

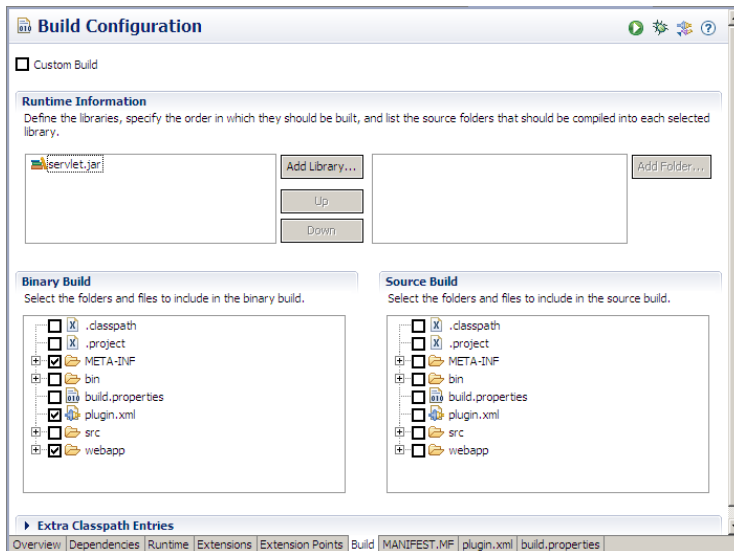


Figure SE-7: Build page

14. To check previous steps, you can compare the newly created files with example files.
15. Click on the src directory and select File.. > New > Package and as the name of package You can paste `com.polarion.example.servlet`.
16. Click on the package You have just created and select File.. > New > Class and as the name set the name of your servlet class, e.g. `CurrentUserWorkloadServlet`.
17. Click Browse in Superclass row and type here `HttpServlet`. Select OK and press Finish.
18. Create files and directories according Figure SE-8.

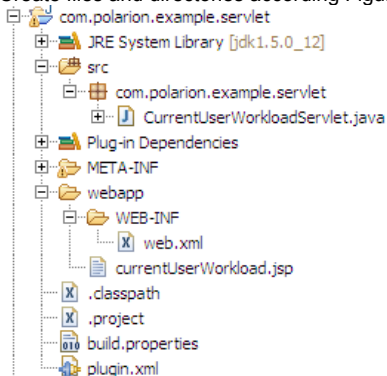


Figure SE-8: Projects files and directories

SE - Deployment to Installed Polarion

See section *Deployment to Installed Polarion*

SE - Execution from Workspace

See section *Execution from Workspace*

SE - Configuration

After successful deployment of plug-in into Polarion, you have to include servlet on Dashboard.

- Go to the Dashboard topic on Repository level.
- Edit Dashboard Wiki page by adding these lines:

```
1.1.1 Current User Workload
<iframe width="100%" height="200" src="/polarion/example/" frameborder="0"></iframe>
```

- After save you should see your new servlet at the bottom of the page.

Requirements

Development Environments

- [Eclipse IDE for Java EE Developers](#) or any other Eclipse IDE with The Eclipse Plug-in Development Environment (Go to Help > Install New Software... > Install Eclipse Plug-in Development Environment > Restart Eclipse)
- [Java Platform, Standard Edition 8 Development Kit](#) or Java bundled with your Polarion installation

Polarion Server

- Polaron Server with version 3.1.0 and higher

Workspace Preparation

To start developing a Polaron Java API plug-in, you first need to perform following steps:

1. Start Eclipse, then select **Window > Preferences...**
2. In the dialog that appears, select **Plug-In Development > Target Platform**.
3. Click the **Add** button on the right.
4. Keep the **Nothing: Start with an empty target definition** option selected and click **Next**.

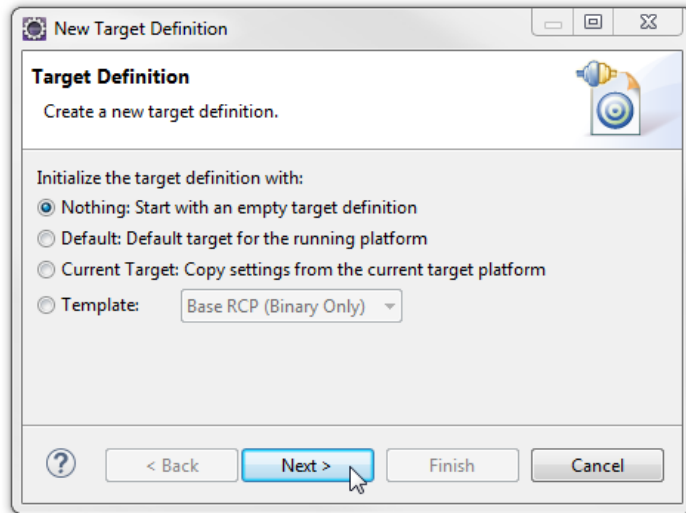



Figure WP-1: Starting with an Empty Target Definition

5. Enter a **Name** and click **Add**.
6. Select  **Directory** and click **Next**.
7. Click **Browse**, select the C:\Polarion\polarion folder and click **Next**.

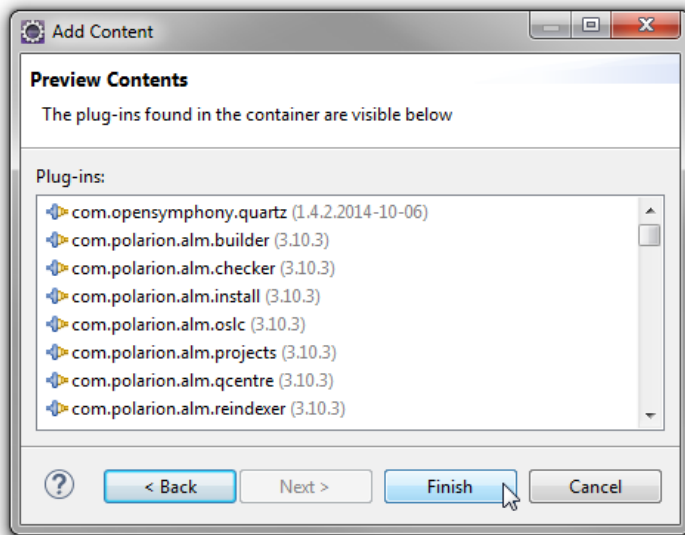


Figure WP-2: Currently Installed Polaron Plug-ins

8. A list of currently installed Polaron plug-ins appears. Click **Finish**.

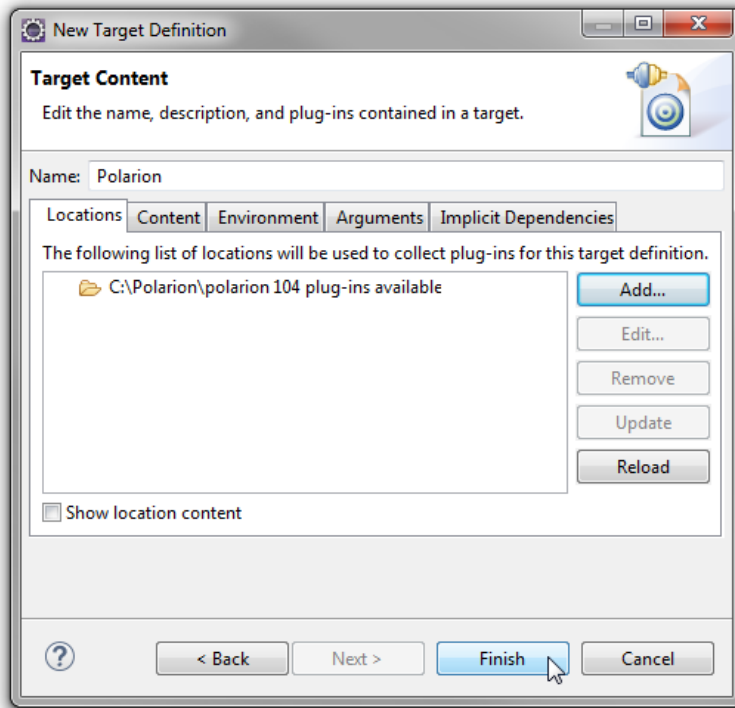


Figure WP-2: Confirm the Selected Path

9. The selected path and the number of discovered plug-ins available appear. Confirm that the path is correct and click **Finish**.

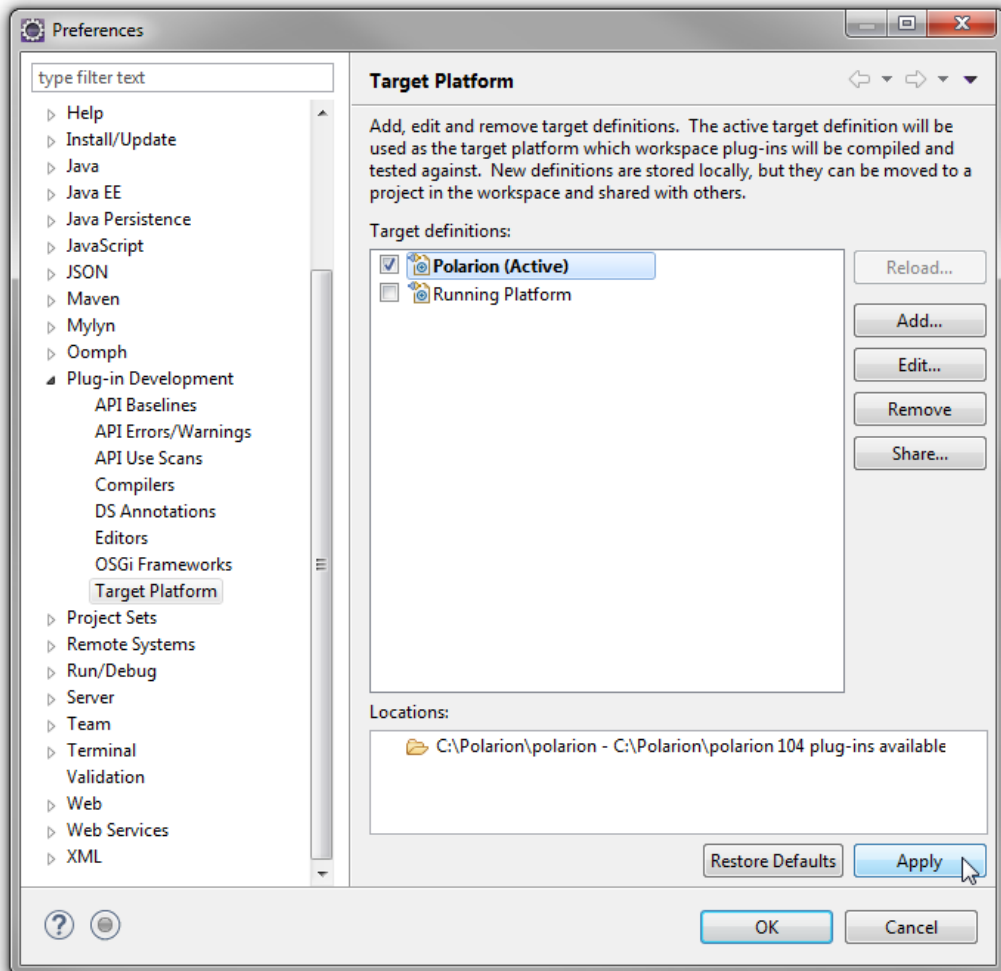


Figure WP-2: Select the Target Platform

10. Check the box beside the newly added path and click **Apply**.

Deployment to Installed Polarion

You can deploy a plugin to Polarion in two ways. First you can export a project as **Deployable Plugins and Fragments**. The second

way is described in the following section *Execution from Workspace*. To export the plug-in, perform these steps:

1. Select **File > Export...**
2. In the dialog that appears, select **Deployable Plugins and Fragments** in **Plug-in Development** section and click the **Next** button.
3. Mark your project (e.g. for **Servlet** example it will be `com.polarion.example.servlet`), and as the destination directory specify the `polarion` folder of your Polarion installation directory (usually in `C:\Polarion\polarion`)
4. At the **Options** card be sure, that **Package plug-ins as individual JAR archives** is unchecked. Click *Finish*.
5. Because this is a new polarion plug-in extension, you have to restart your Polarion server.

Execution from Workspace

The second way to deploy the plug-in to Polarion is to launch Polarion directly from your Eclipse workspace. This method has the added advantage of debugging the code directly in Eclipse.

1. Select **Run > Open Debug Configurations..**
2. Create a new Eclipse application (double click on *Eclipse Application*)
3. You should set:

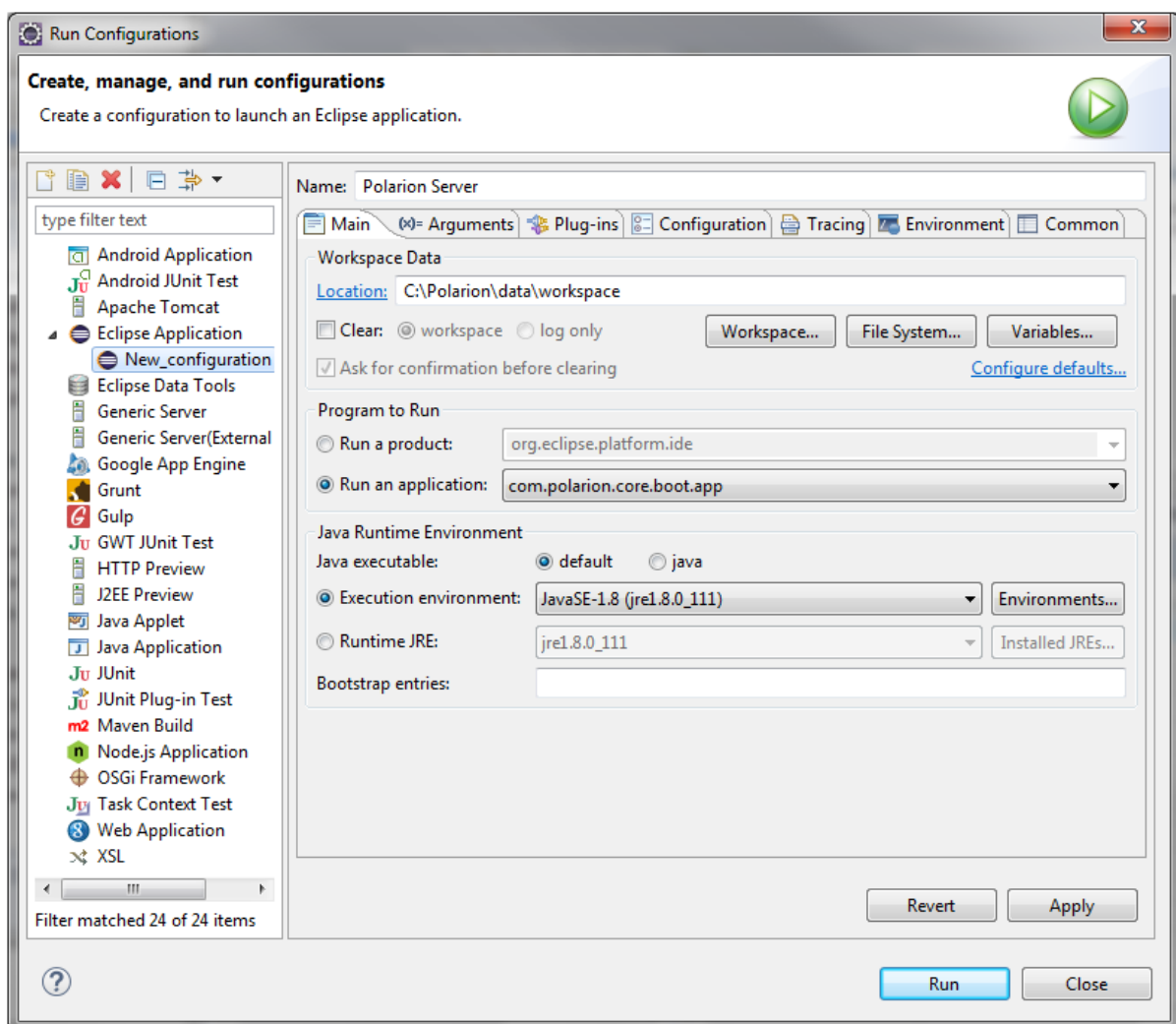


Figure WP-2: Debug - Main page

- **Name** to Polarion Server
 - **Workspace Data Location** to `C:\Polarion\data\workspace` (assuming that your Polarion is installed in `C:\Polarion\`)
 - **Run an application** to `com.polarion.core.boot.app` in the **Program to Run** section
 - Finally set your Runtime JRE
4. On the second, "**Program Arguments**" tab, set the following arguments:
 - to **Program Arguments** section:

```
-os win32 -ws win32 -arch x86 -appId polarion.server
```

1.

- to **VM Arguments** section:

```
-Xms256m -Xmx640m -XX:MaxPermSize=128m -XX:PermSize=128m  
-Xbootclasspath/a:C:\Polarion\bundled\java\lib\tools.jar  
-Dcom.polarion.home=C:\Polarion\polarion
```

Of course, you have to change parameters to Polarion server according your installation.

Note that, when using Java 8 the arguments `XX:MaxPermSize` and `XX:PermSize` are obsolete and must not be used anymore.

1.

- You can check the settings with the following screenshot:

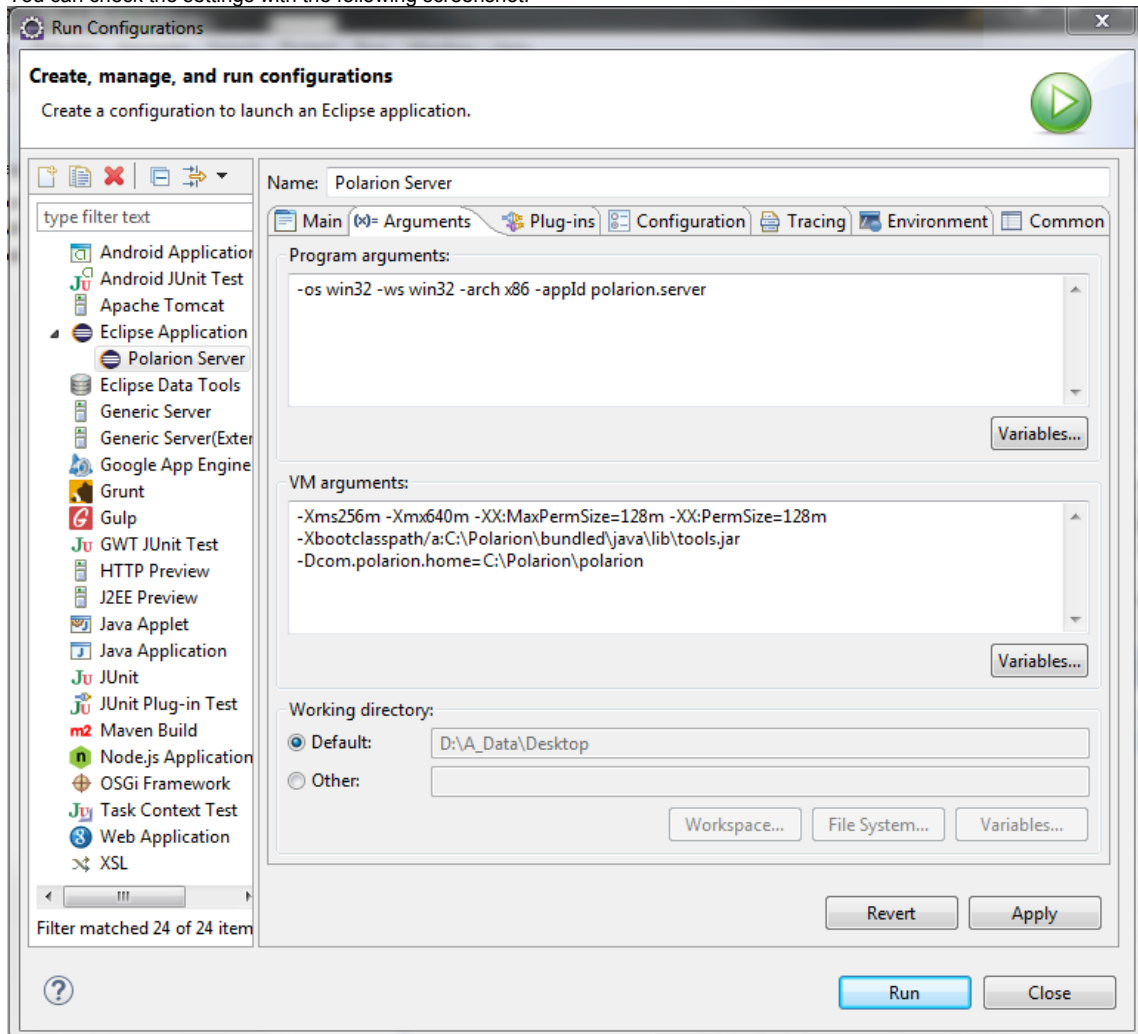


Figure WP-3: Debug - Arguments page

2. On the third "**Plugins**" tab, make sure, you have also selected "**Target Platform**" plugins.
3. Select all, and then click the **Validate Plug-ins** button. If there are some problems, uncheck the plugins which are in conflict.

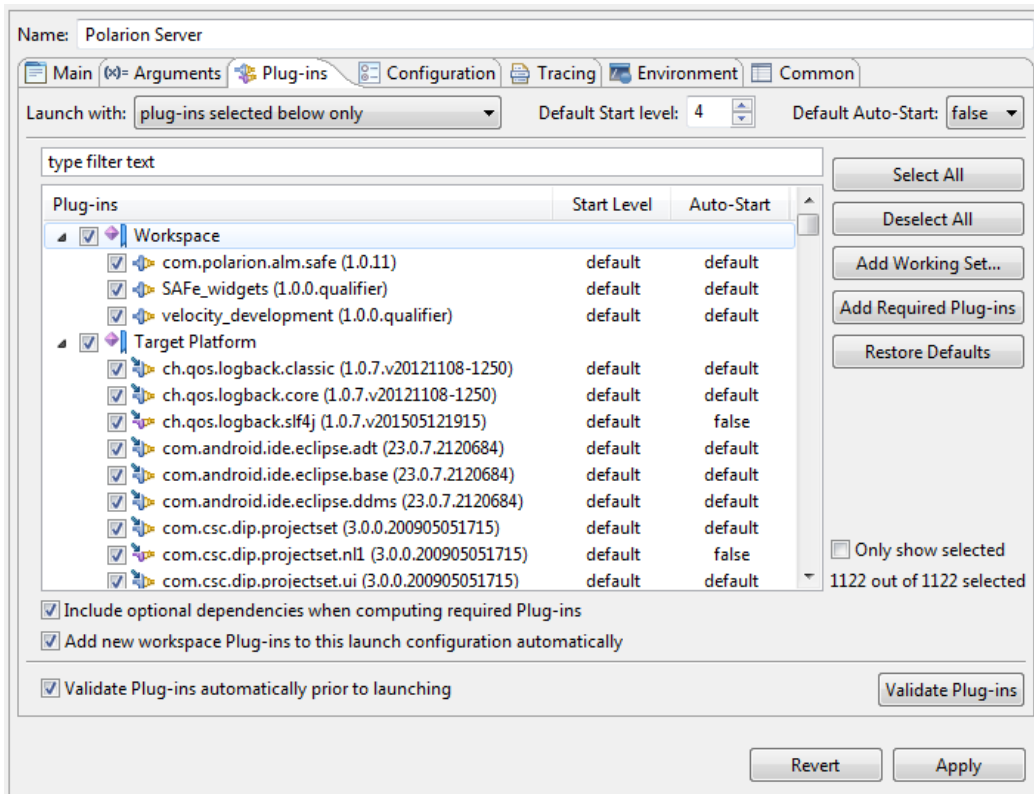


Figure WP-4: Debug - Plug-ins page

- Other pages shouldn't be changed. Just click the **Debug** button, and go on with your new Polarion Server application.