

minigameslib.de

Spigot Eclipse-Plugin

Eclipse-Plugin and JUnit support

mepeisen
27.10.2016

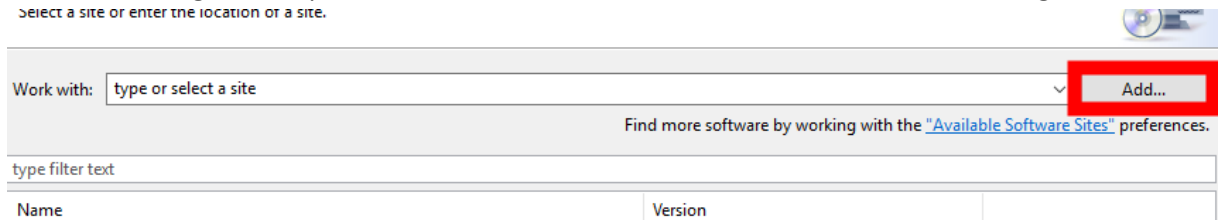
Table of contents

Eclipse setup.....	2
Setup spigot servers.....	3
Installing third party plugins.....	6
First plugin project.....	6
Hot code replacement.	9
Junit support.....	11

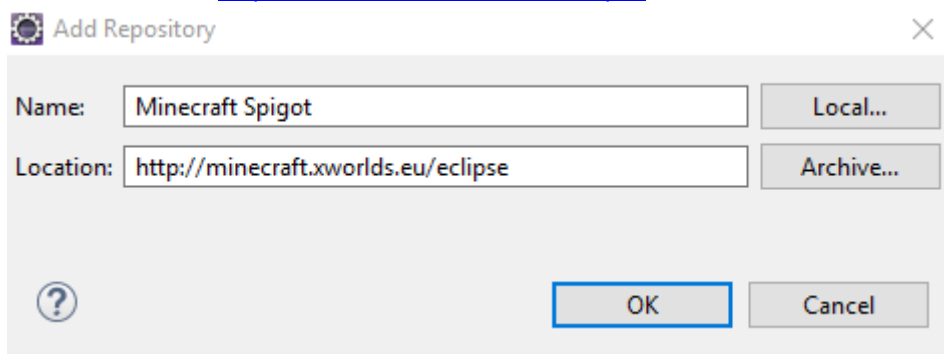
Eclipse setup

The spigot plugins were tested on Eclipse neon.

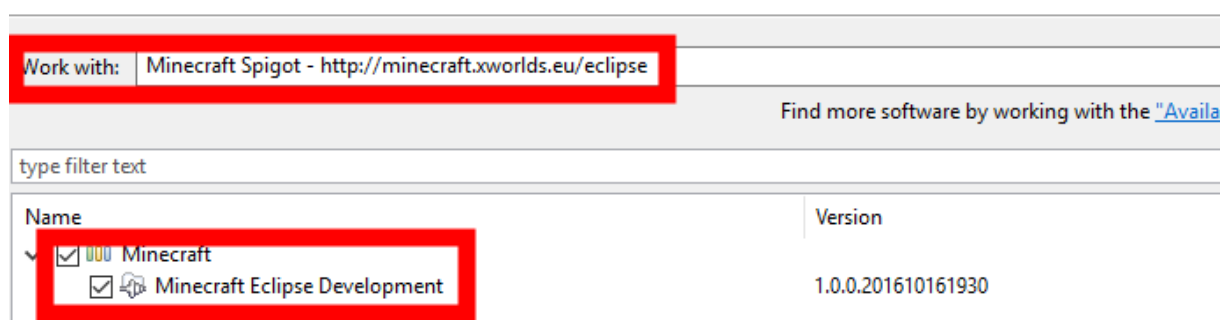
- 1) You can download eclipse neon for java EE development at <http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/neon1a>
- 2) After first start go to “help” > “Install new software”. Select the “Add” button on the right.



- 3) You can now enter a new update site with following values:
 Name: Minecraft Spigot
 Location: <http://minecraft.xworlds.eu/eclipse>



- 4) Select the site in drop down and select the minecraft spigot feature to install.
 Check the items that you wish to install.



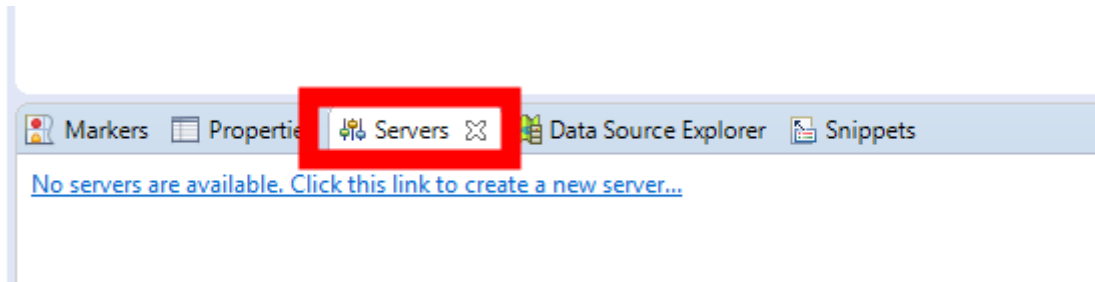
- 5) Follow the wizard steps and let eclipse install the plugins.

Setup spigot servers

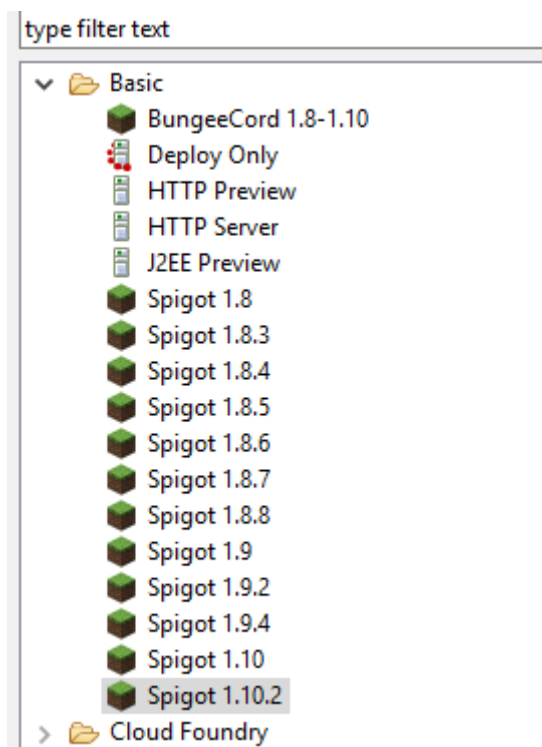
Eclipse webtools divides servers and runtimes. A runtime is a location of your spigot server on your local hdd. A server holds all configurations, plugins and everything you need to start spigot.

If you are a developer of two plugins you can use the same runtime within two servers. It is one server for each plugin. Or you can use only one server for both.

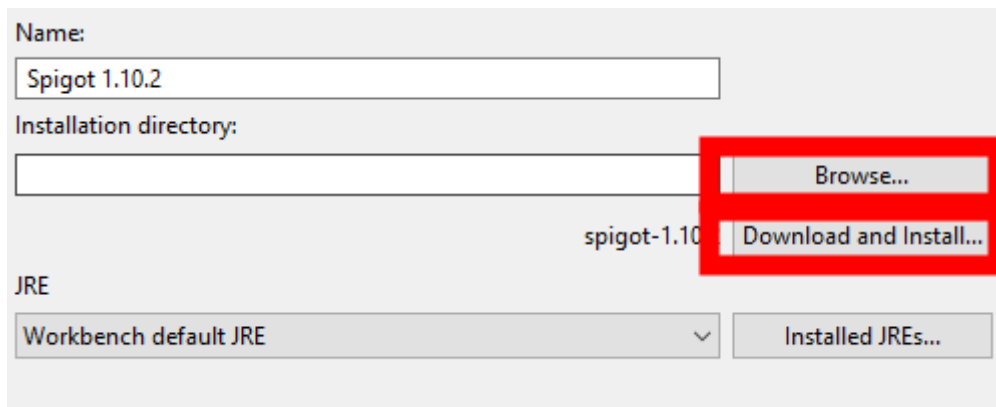
- 1) To create a server, go to the server view on the bottom of the java ee perspective:



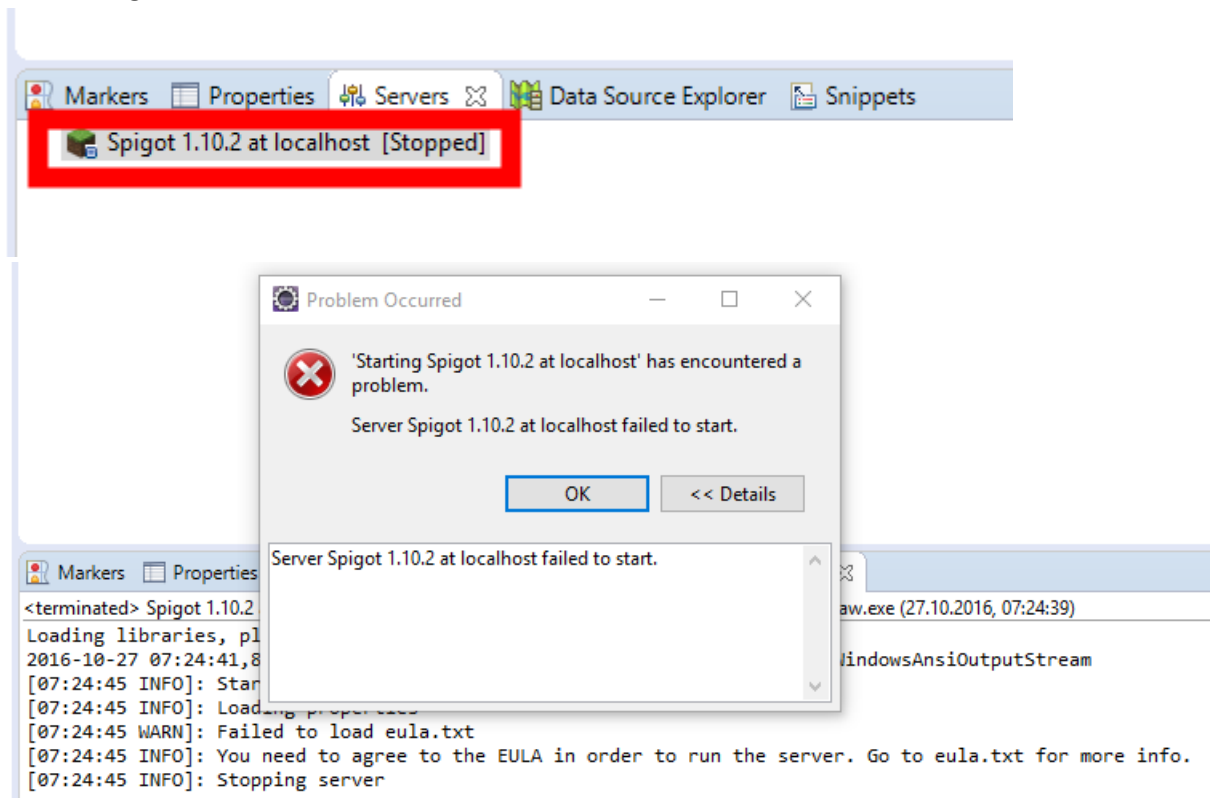
- 2) Click on the hyperlink. In the “new Server” wizard select the spigot server under category “Basic”.



- 3) You will be forced to create a new runtime. Select either your existing installation or download it from our repository.

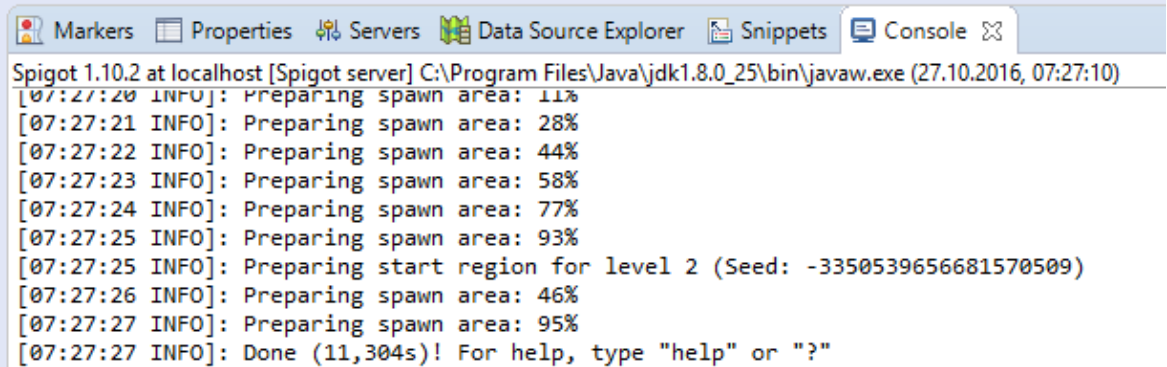


- 4) On the next wizard page it will ask you which resources to add. We do not yet have any spigot plugin in our workspace. Simply ignore and click on “finish”.
- 5) Now you have a working spigot server. Right click or use the buttons on the right to start/debug it.



- 6) As you see it will fail. However as an experienced minecraft/ spigot user you will know what to do. Have a look at your workspace and open the “Servers” projects. After doing a refresh

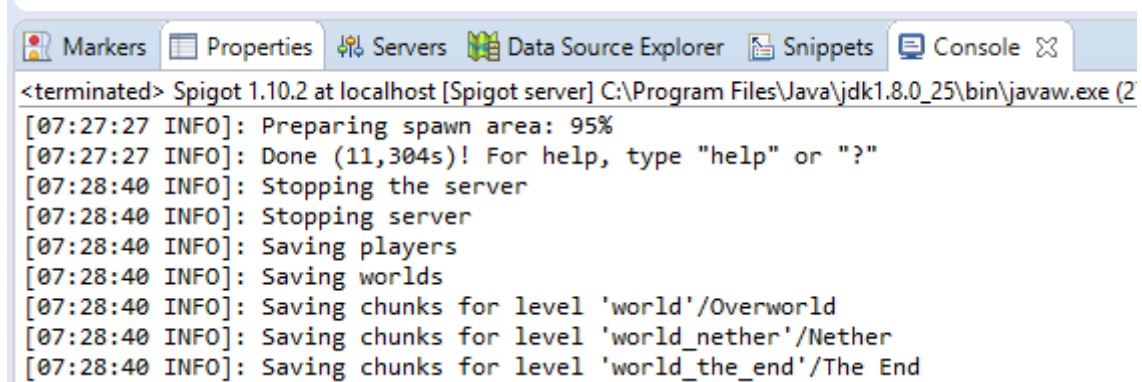
(F5) you will see the eula.txt. Edit it, accept the eula and restart your server again.



```

Spigot 1.10.2 at localhost [Spigot server] C:\Program Files\Java\jdk1.8.0_25\bin\javaw.exe (27.10.2016, 07:27:10)
[07:27:20 INFO]: Preparing spawn area: 11%
[07:27:21 INFO]: Preparing spawn area: 28%
[07:27:22 INFO]: Preparing spawn area: 44%
[07:27:23 INFO]: Preparing spawn area: 58%
[07:27:24 INFO]: Preparing spawn area: 77%
[07:27:25 INFO]: Preparing spawn area: 93%
[07:27:25 INFO]: Preparing start region for level 2 (Seed: -3350539656681570509)
[07:27:26 INFO]: Preparing spawn area: 46%
[07:27:27 INFO]: Preparing spawn area: 95%
[07:27:27 INFO]: Done (11,304s)! For help, type "help" or "?"
  
```

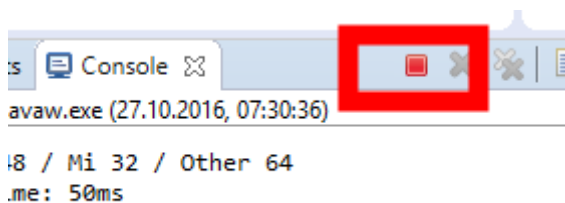
- 7) To shut down the server simply right click in the servers view or type “stop” in the console. You may use your minecraft to connect to the server and type “/stop” in the chat as well.

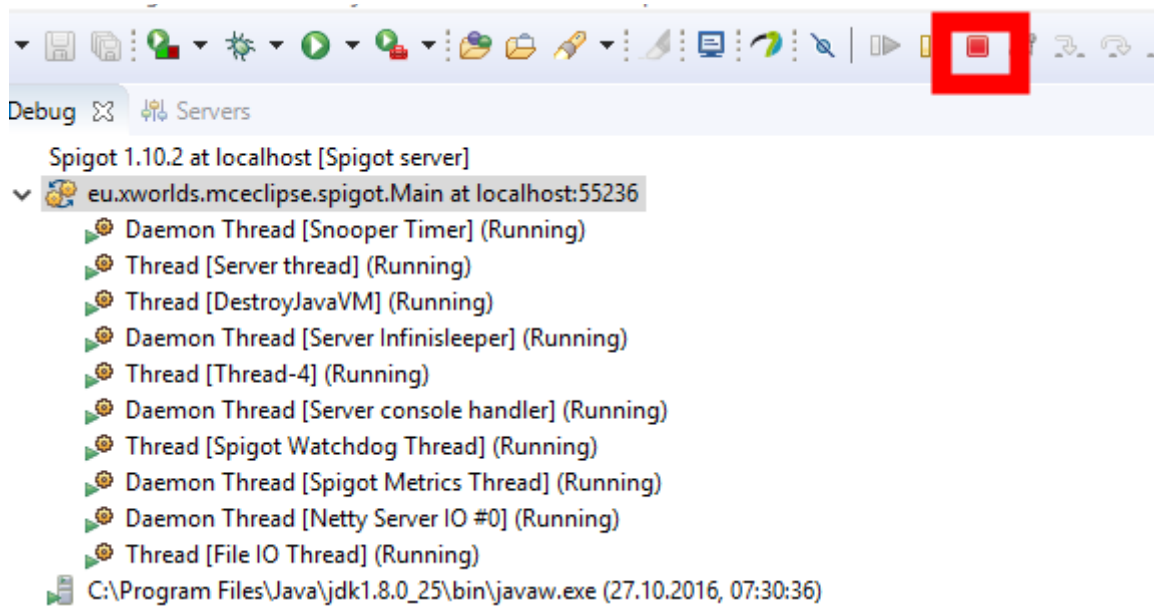


```

<terminated> Spigot 1.10.2 at localhost [Spigot server] C:\Program Files\Java\jdk1.8.0_25\bin\javaw.exe (2
[07:27:27 INFO]: Preparing spawn area: 95%
[07:27:27 INFO]: Done (11,304s)! For help, type "help" or "?"
[07:28:40 INFO]: Stopping the server
[07:28:40 INFO]: Stopping server
[07:28:40 INFO]: Saving players
[07:28:40 INFO]: Saving worlds
[07:28:40 INFO]: Saving chunks for level 'world'/Overworld
[07:28:40 INFO]: Saving chunks for level 'world_nether'/Nether
[07:28:40 INFO]: Saving chunks for level 'world_the_end'/The End
  
```

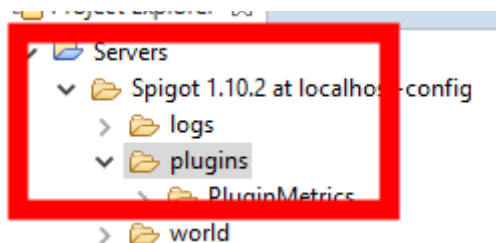
NOTICE: In the console view there is a red terminate button. Do not use it. It will shut down the java VM but does not save world data etc. You may use it if the spigot server crashed and does not respond. You should always prefer stopping the server by “stop” command or via “Servers” view. The second non-preferred terminate button is inside debug view.





Installing third party plugins

It is fairly simple. Stop the server. Download the plugin jar files. Copy them. Open the Servers project and put them into the plugins folder.



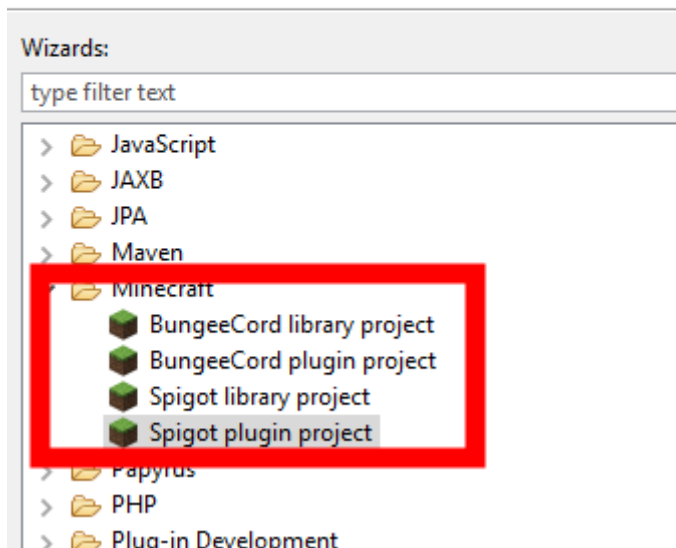
First plugin project

- 1) Right click in the “project explorer” view and select “New” > “Project”. You will find a “Minecraft” category and the “Spigot plugin project” type.

 New Project

Select a wizard

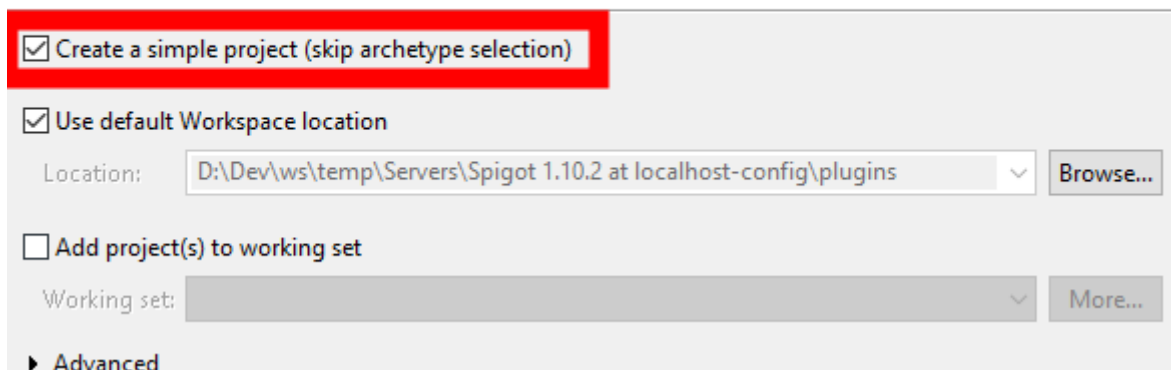
Create a new spigot plugin



- 2) Select the “Create a simple project” checkbox.

New Maven project

Select project name and location



- 3) Type a maven group id (typically a reversed internet domain you own). Type a maven artifact id and select "Finish".

New Maven project

Configure project

Artifact

Group Id:

Artifact Id:

Version: ▼

Packaging: ▼

Name:

Description:

Parent Project

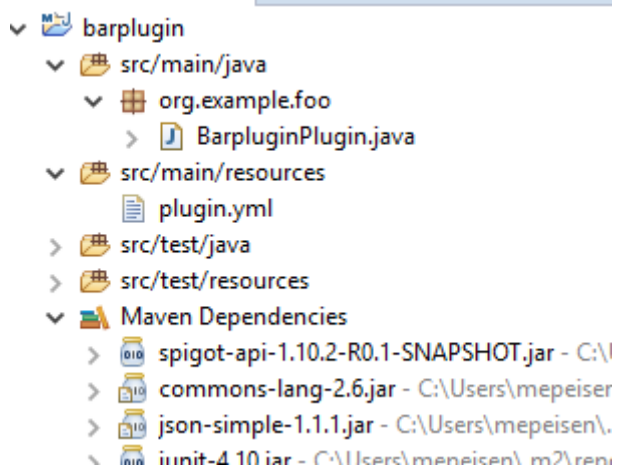
Group Id:

Artifact Id:

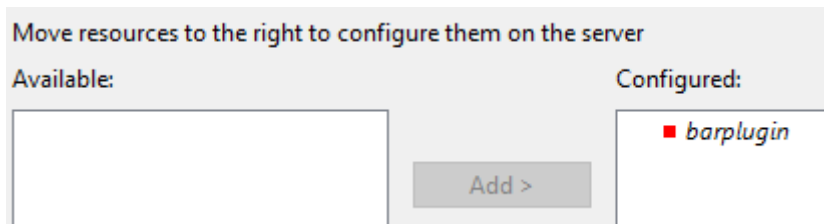
Version: ▼

► Advanced

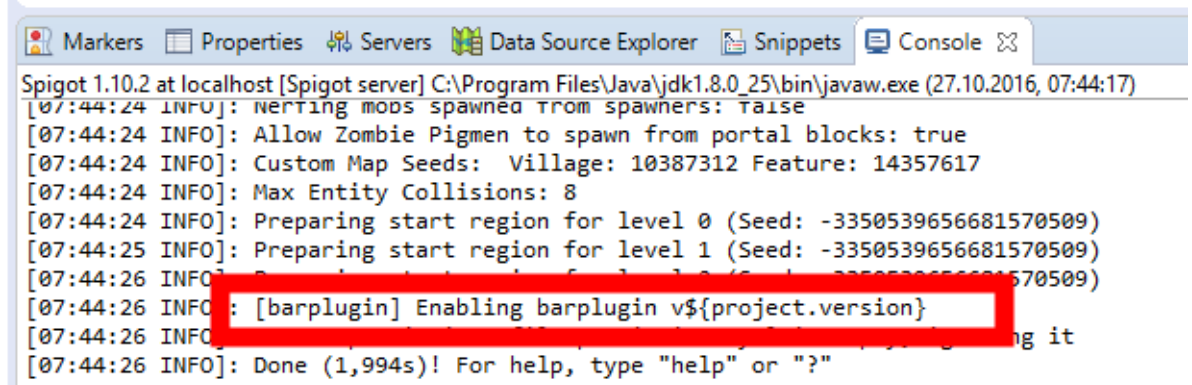
- 4) You now have a new spigot plugin within your workspace.



- 5) Go back to your server and right click on it. Select the “Add and remove” command. You will now see the barplugin being available for your spigot server. Move it to the right.

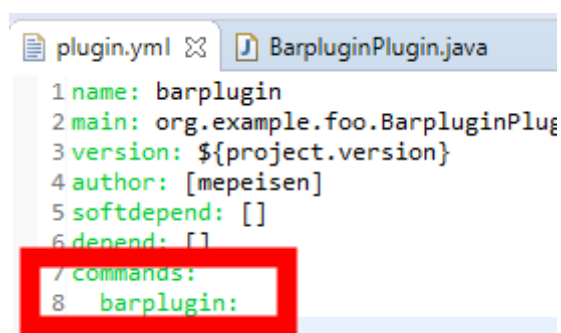


- 6) Restart the server and watch the console.



Hot code replacement.

Eclipse supports a feature called hot code replacement. That’s useful during development. It allows you to change code without restarting the whole server. We will create a small useful example. Add a command by editing the plugin.yml and the BarpluginPlugin class.

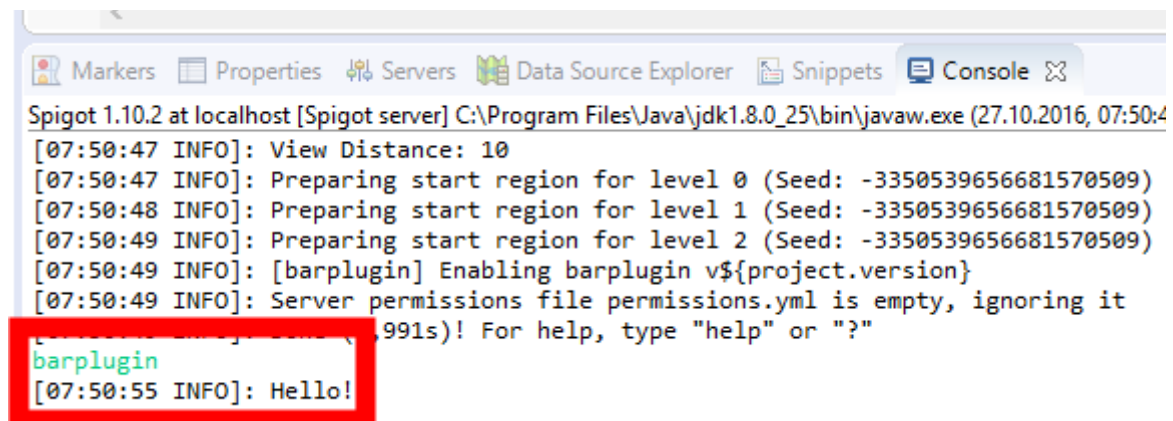


```

6
7 public class BarpluginPlugin extends JavaPlugin
8 {
9
10 public BarpluginPlugin()
11 {
12     // TODO Put in some initialization code.
13 }
14
15 @Override
16 public void onEnable()
17 {
18     // TODO Put in your activation code.
19 }
20
21 @Override
22 public void onDisable()
23 {
24     // TODO Put in your deactivation code.
25 }
26
27 @Override
28 public boolean onCommand(CommandSender sender, Command command, String label, String[] args)
29 {
30     switch (command.getName())
31     {
32     case "barplugin":
33         sender.sendMessage("Hello!");
34         break;
35     }
36     return false;
37 }
38
39 }

```

Now start the server and check your code.



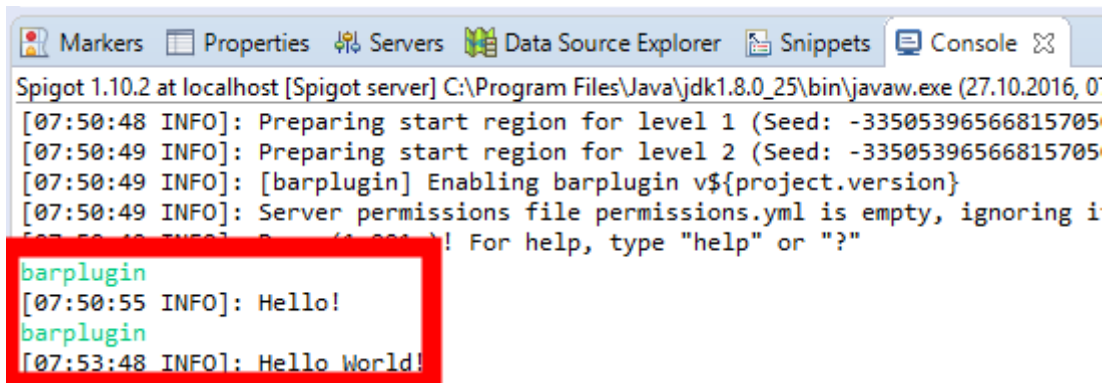
The screenshot shows the Spigot console output. The top part shows the server startup logs for Spigot 1.10.2 at localhost. The bottom part shows the command 'barplugin' being executed, resulting in the message 'Hello!' being sent to the console. The command and its output are highlighted with a red box.

```

Spigot 1.10.2 at localhost [Spigot server] C:\Program Files\Java\jdk1.8.0_25\bin\javaw.exe (27.10.2016, 07:50:4
[07:50:47 INFO]: View Distance: 10
[07:50:47 INFO]: Preparing start region for level 0 (Seed: -3350539656681570509)
[07:50:48 INFO]: Preparing start region for level 1 (Seed: -3350539656681570509)
[07:50:49 INFO]: Preparing start region for level 2 (Seed: -3350539656681570509)
[07:50:49 INFO]: [barplugin] Enabling barplugin v${project.version}
[07:50:49 INFO]: Server permissions file permissions.yml is empty, ignoring it
[07:50:49 INFO]: (1991s)! For help, type "help" or "?"
barplugin
[07:50:55 INFO]: Hello!

```

It is working. But let us now use the hot code replacement. WIHTOUT restarting the server change the message in your code to "Hello World!". Save the java file and check the command again.



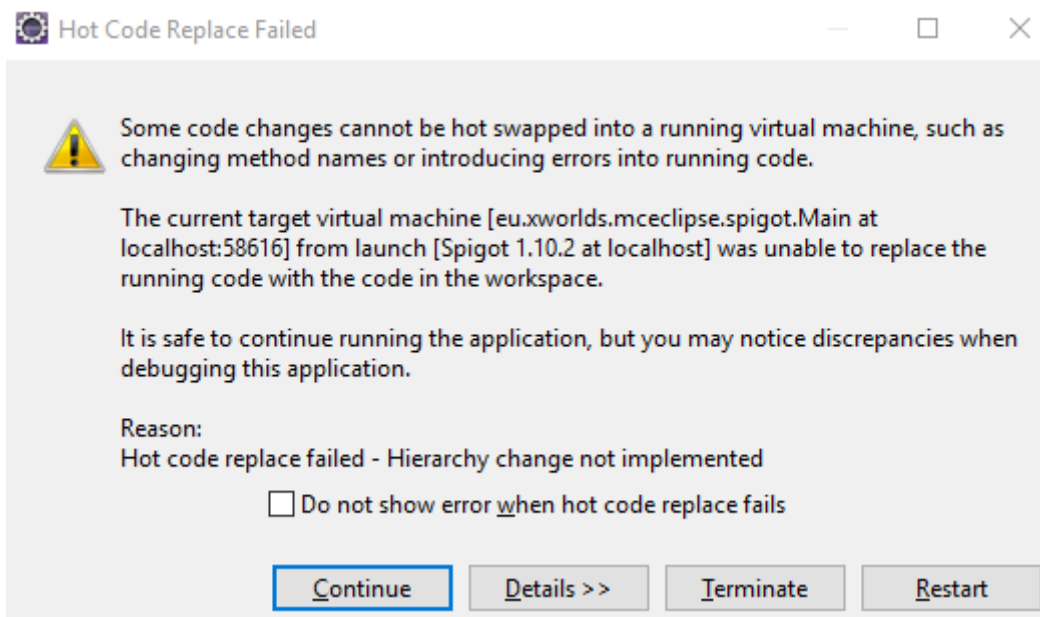
```

Spigot 1.10.2 at localhost [Spigot server] C:\Program Files\Java\jdk1.8.0_25\bin\javaw.exe (27.10.2016, 0'
[07:50:48 INFO]: Preparing start region for level 1 (Seed: -33505396566815705)
[07:50:49 INFO]: Preparing start region for level 2 (Seed: -33505396566815705)
[07:50:49 INFO]: [barplugin] Enabling barplugin v${project.version}
[07:50:49 INFO]: Server permissions file permissions.yml is empty, ignoring it
[07:50:49 INFO]: Done! For help, type "help" or "?"
barplugin
[07:50:55 INFO]: Hello!
barplugin
[07:53:48 INFO]: Hello World!

```

The message changed. Without building jar files, without redeploying to server and without restarting the server.

The hot code replacement is limited. Changing class structures often result in warnings to restart the server. For example let your plugin implement the bukkit listener interface. You will get the following warning.



Your spigot server is now out of sync. Restart it to get test the new code.

JUnit support

- 1) Edit your pom.xml and add the following elements:

```
<dependency>
  <groupId>de.minigameslib.mclib</groupId>
  <artifactId>spigot-testsupport</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <scope>test</scope>
</dependency>

</dependencies>
<repositories>
  <repository>
    <id>mce-repo</id>
    <url>http://nexus.xworlds.eu/nexus/content/groups/mce</url>
  </repository>
```

- 2) Right click on “src/test/java” and create a JUnit class for testing (“New “ > “Other” > “Java” > “JUnit” > “JUnit Test Case”).

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.



☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

Package:

Name:

Superclass:

Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()
☐ setUp() ☐ tearDown()
☐ constructor

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

☒ Generate comments

Class under test:



< Back

Next >

Finish

Cancel

- 3) Add the annotations “RunWith” and “SpigotTest” at class level.

```

    */
    @RunWith(SpigotJUnit4Runner.class)
    @SpigotTest(versions = {SpigotVersion.V1_10_2})
    public class BarpluginTest {

```

- 4) To access our spigot server during junit tests we need to add a variable annotated with “SpigotInject”.

```

    @SpigotInject
    private SpigotServer server;

```

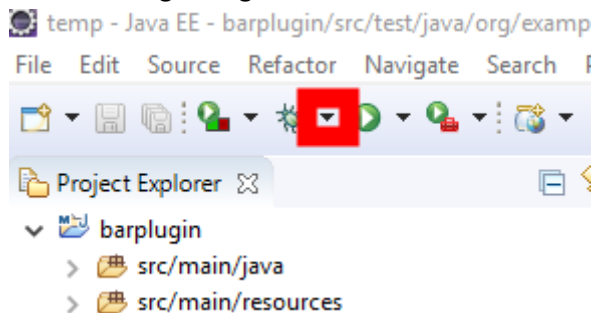
- 5) We are now ready to perform our test. As we already have a command we will simply test it:

```

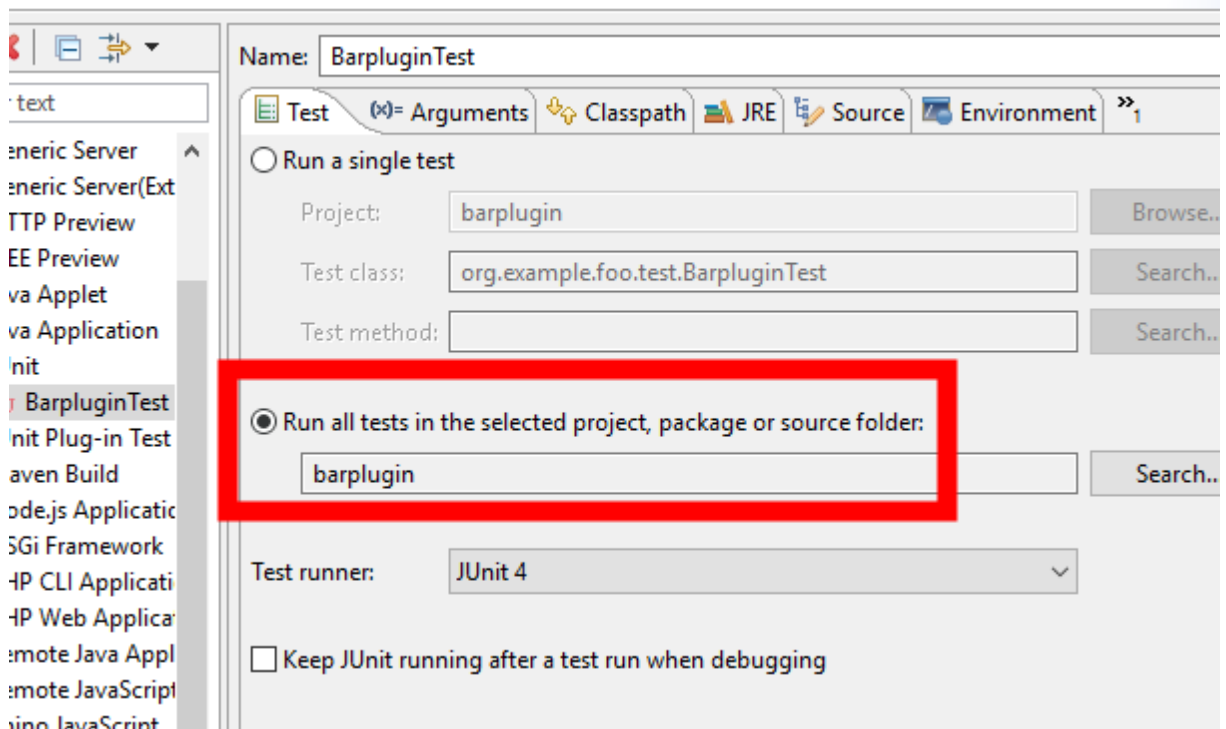
    @Test
    public void testBarpluginCommand() throws IOException
    {
        this.server.clearConsole();
        this.server.sendCommand("barplugin");
        assertTrue(this.server.waitForConsole(".*Hello.*", 5000));
    }

```

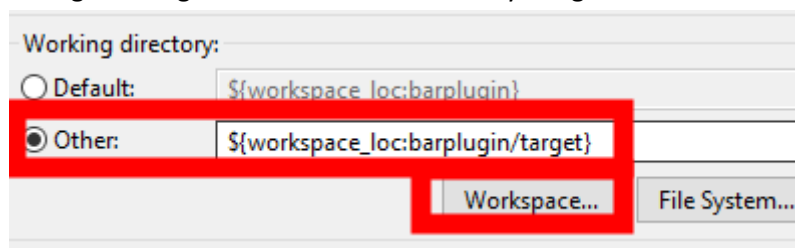
- 6) We now create a new debug profile via toolbar. Click the small arrow right from the bug and select “debug configurations...”.



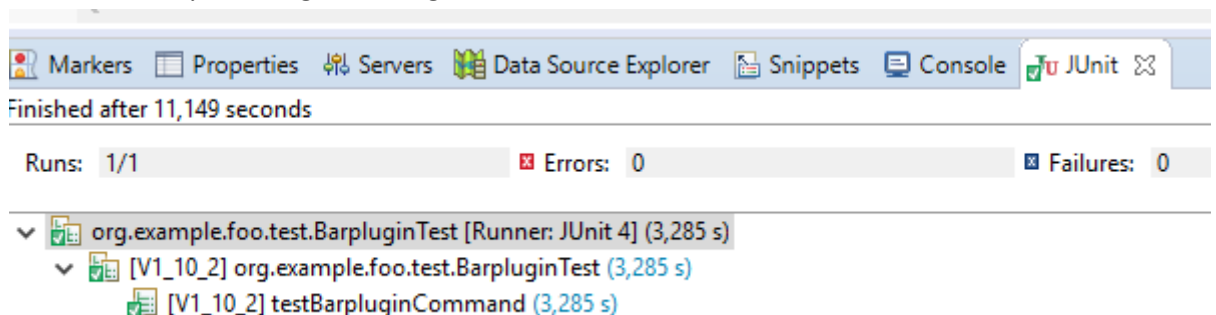
- 7) Right click on “JUnit” and select “New”. Change the properties to run all tests from your project.



- 8) Change the arguments to start in directory “target”.



- 9) Start the test by selecting the debug button.



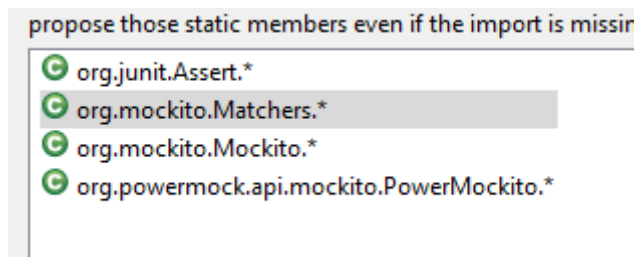
Hint: We do not really need our own debug configuration for the whole project and with folder “target”.

You can simply run your tests by right clicking the class and select “Run as JUnit test”.

However eclipse always chooses to use the projects root directory. You may be messed up with log files, configuration files etc.

Typically in scm (SVN or GIT) you will never commit the target directory. Choosing the target directory as test working directory will automatically hide all the test files from scm.

Hint: We use a static import “assertTrue” in our sample code. Eclipse is able to automatically import statics but the feature is somehow hard to find. Go the “Window” > “Preferences” > “Java” > “Editor” > “Content Assist” > “Favorites”. Add the following by selecting the “New Type” button:



Now you can type “assertTrue” in the java code editor and press “Ctrl+Space” for content assist and creating a static import.

Hint: As you see the SpigotTest annotation controls the servers you want to test. You can add more servers or even simply set “all” to true. See what happens if we test all the versions:

