

TFC (79) Add-on Development Environment Setup

This document was created to provide a consistent way of setting up the TFC add-on development environment. It was developed for use under Microsoft Windows and the following software applications are used:

1. Java Development Kit (jdk 1.7.0_80).
2. Eclipse IDE for Java Developers (Luna Service Release 1a (4.4.1))
3. GitHub for Windows.

A basic template structure was created as a starting point for coding the add-ons. The template code requires some customisation changes for your specific add-on and they will be listed when we get to that section.

This document does not teach the use of Java, Eclipse, GitHub, Minecraft or TFC. This document assumes you have an understanding on each of them. This document is solely for the purpose of setting up an environment so you can create an add-on for TFC.

This document assumes that each of the software has been installed and this is the first time the TFC development environment has been setup.

Note: At the time of writing this document the current version of TFC was 0.79.19 for Minecraft 1.7.10. As Minecraft and TFC develop this document may become outdated.

Disclaimer – This may not be the best way to setup a TFC environment, but it is the one I use and it seems to work well.

TFC (79) Add-on Development

Environment Setup

Contents

Download TFC Code	3
Process	3
Setup Eclipse Workspace	4
Process	4
Java Runtime Setup.....	5
Process	5
TerraFirmaCraft (TFC)	7
TFC Setup	7
Process	7
Importing TFC into Eclipse	8
Process	8
Creating Debug and Run Configurations.....	12
Process	12
Testing the TFC Project	18
Process	18
TFC Add-on.....	19
TFC Add-on Setup.....	19
Process	19
Importing Your Add-on into Eclipse	20
Process	20
Configure Add-on Build Settings	22
Process	22
Creating Debug and Run Configurations.....	25
Process	25
Testing the Add-on Project	31
Process	31
Add-on Customisation	32
Namespace Change.....	32
Proxy Files	33
Add-on Details.....	33
Building Your Add-on for Distribution	34
Building TFC.....	34
Process	34
Building Your Add-on	34
Process – Windows Explorer.....	34
Process – Eclipse	34

TFC (79) Add-on Development Environment Setup

Download TFC Code

In order to develop an add-on for TFC, you will require a deobfuscate version of the jar file. TFC do not provide this to download, so the easiest way to get one is to download the code and set it up in your development environment. This has the added benefit of providing you access to the TFC code to better understand it and also ensure your add-on compliments it rather than hinders it.

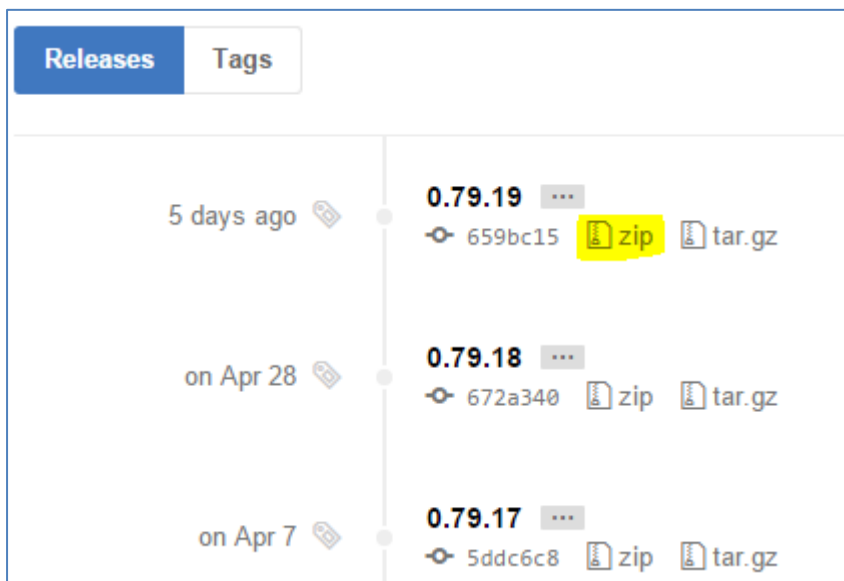
Each release of TFC, I download a copy of the source and import into my environment. This allows me to test my add-on against multiple TFC versions.

Process

1. Open a browser window and navigate to the TFC GitHub site ([TFC GitHub](#)).
2. Click the releases link



3. Locate the current release (should be the first one as they are listed in reverse order) and click the zip link.



4. When prompted for a download location, save to your desktop. We will need this file later on.
5. Close your browser.

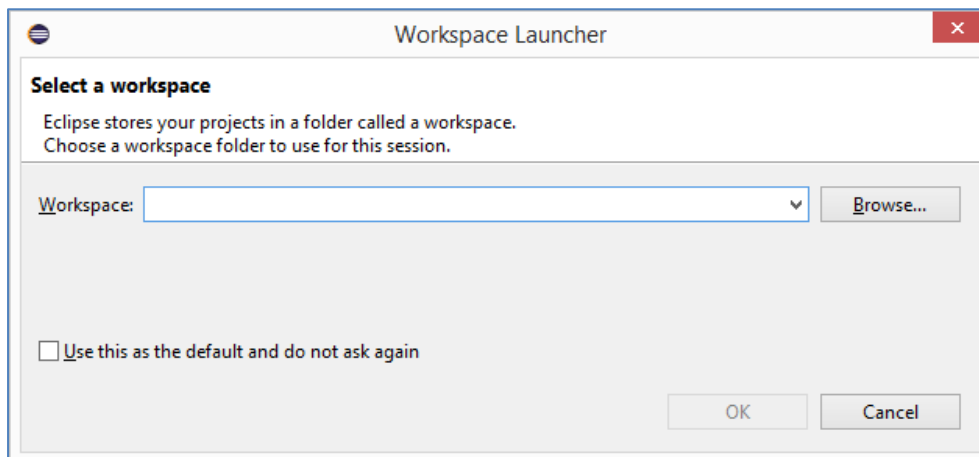
TFC (79) Add-on Development Environment Setup

Setup Eclipse Workspace

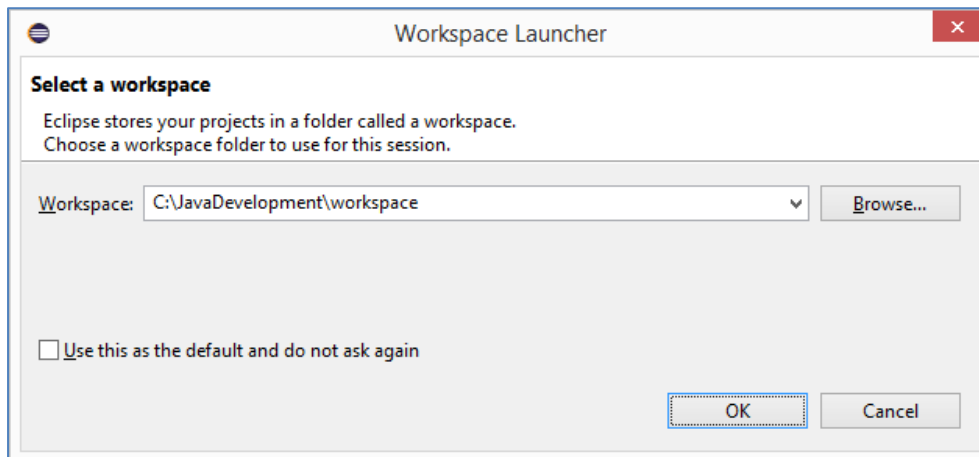
There is no reason behind the use of Eclipse over any other Java IDE. It was the first one I was introduced to and it is the one I still use. Eclipse uses workspaces to store java projects and settings and TFC is no different. The first step is to create a new workspace for the TFC project and your add-on project.

Process

1. Start Eclipse.
2. You will be prompted for a workspace.



3. Click the browse button and navigate to a drive/folder where you want your workspace to exist. (e.g. C:\JavaDevelopment\workspace)



4. DO NOT check the 'Use this as the default and do not ask again' option.
5. Click OK to accept your choice and open Eclipse.

If you want to change your workspace location, just shutdown eclipse, start it again and you should be prompted for the workspace again. It will default to the last opened workspace, but following the steps above, you can create a new workspace location.

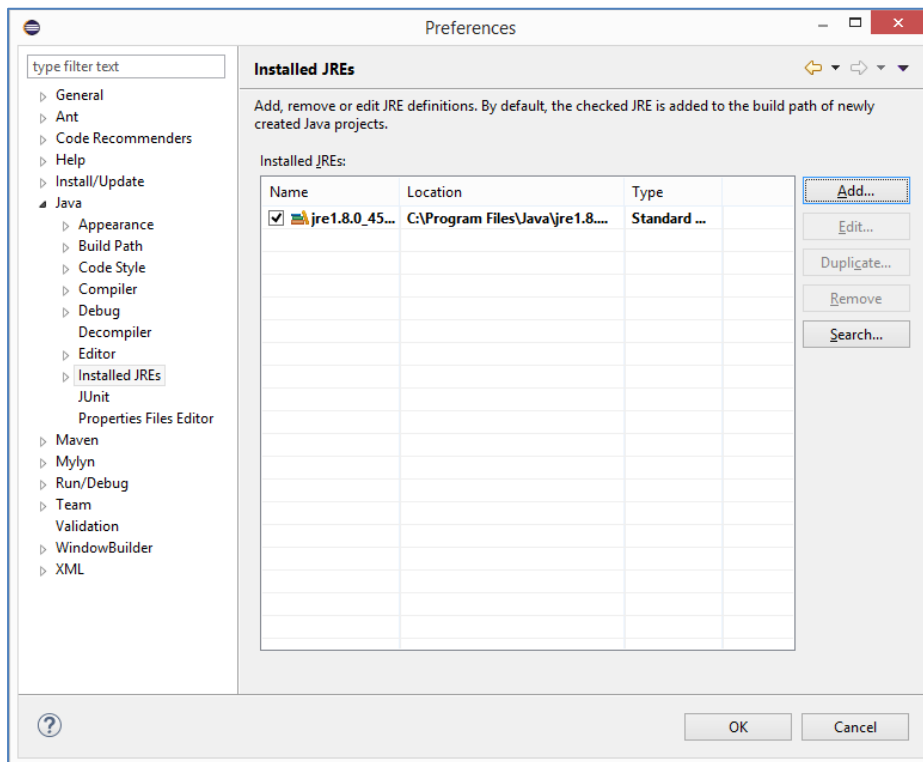
TFC (79) Add-on Development Environment Setup

Java Runtime Setup

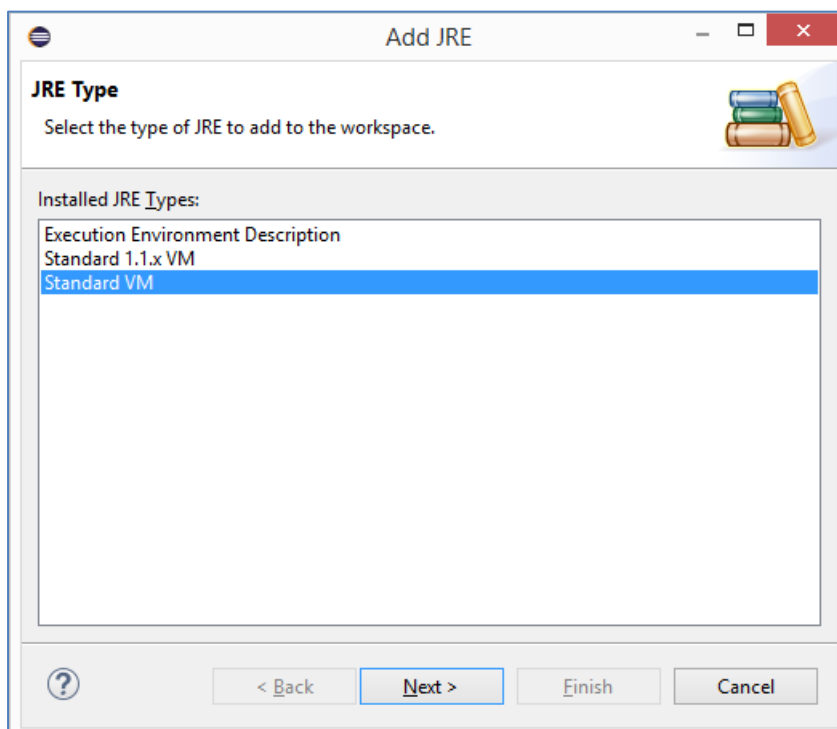
TFC Uses Java Runtime 7 and the workspace must be setup to use that by default. If you have other JDK's installed, the workspace may default to another version.

Process

1. Open the preferences window, expand the Java node and select Installed JREs. As shown, the default for me instance was JRE 8. If your workspace already has JRE 7 as default, you can skip this section.

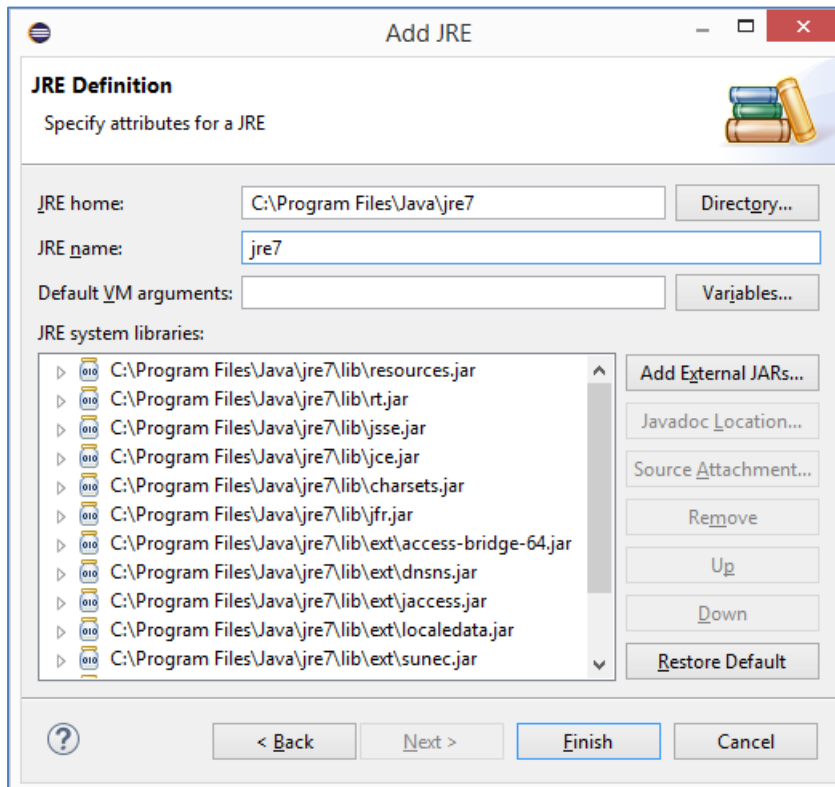


2. Click the Add button, select Standard VM and click the Next button.

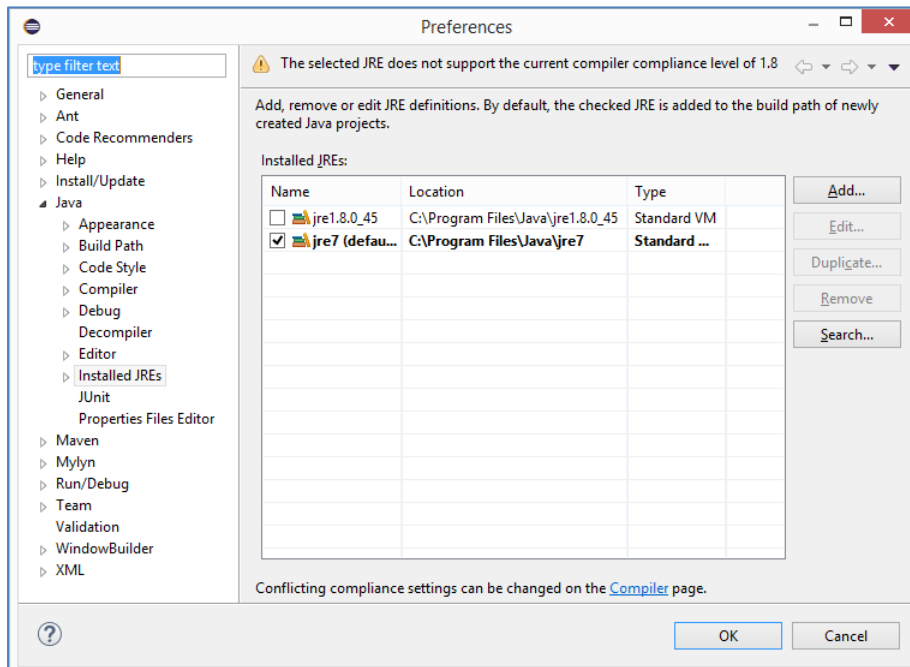


TFC (79) Add-on Development Environment Setup

3. Enter the JRE Home folder and click finish.



4. Make sure to check the JRE 7 option to make it the default. Click OK to finish.



TFC (79) Add-on Development Environment Setup

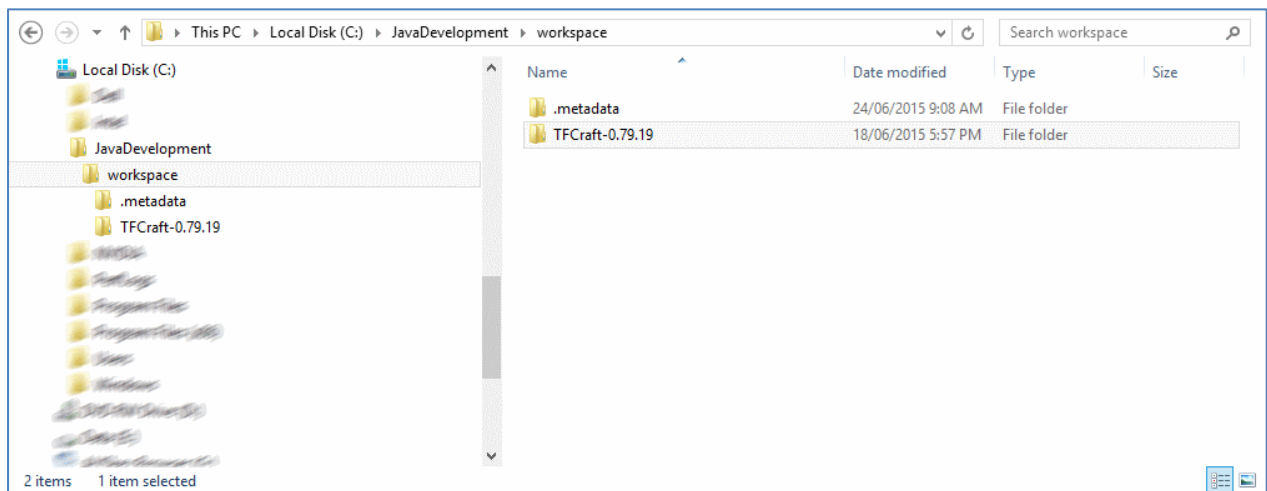
TerraFirmaCraft (TFC)

TFC Setup

In order to import the TFC code into the eclipse workspace, we must first setup the TFC project. TFC uses gradle which makes the whole process easy.

Process

1. Locate the downloaded TFC code zip file on your desktop. This is the file you downloaded earlier.
2. Unzip the file into your eclipse workspace folder that you created in the previous section.
(e.g. C:\JavaDevelopment\workspace).



3. Open the new TFCraft-0.79.x folder.
4. Locate the setup.bat file and double-click it. This will start gradle and begin setting up your project, ready to be imported into eclipse. This step can take quite a while as it will be downloading libraries from internet repositories.

Note: On occasions I have found that some errors have occurred during this step. If this happens to you, try running setup.bat again. If the problem persists, then delete the TFCraft-0.79.x folder and start this section again.

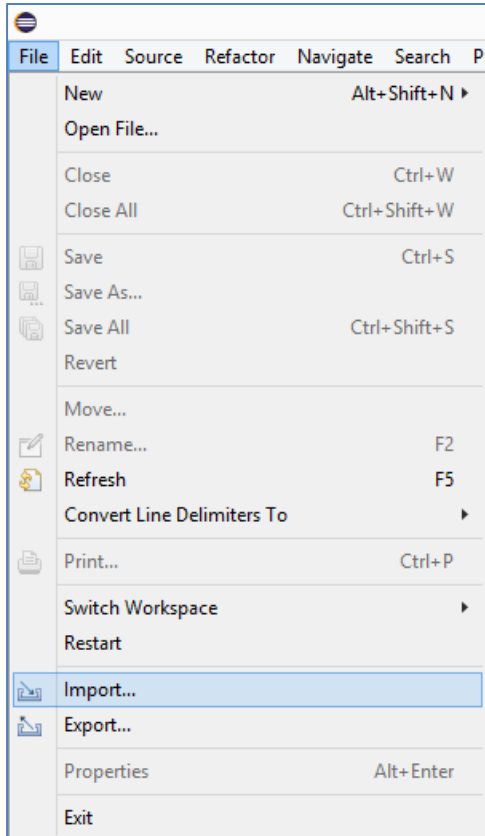
TFC (79) Add-on Development Environment Setup

Importing TFC into Eclipse

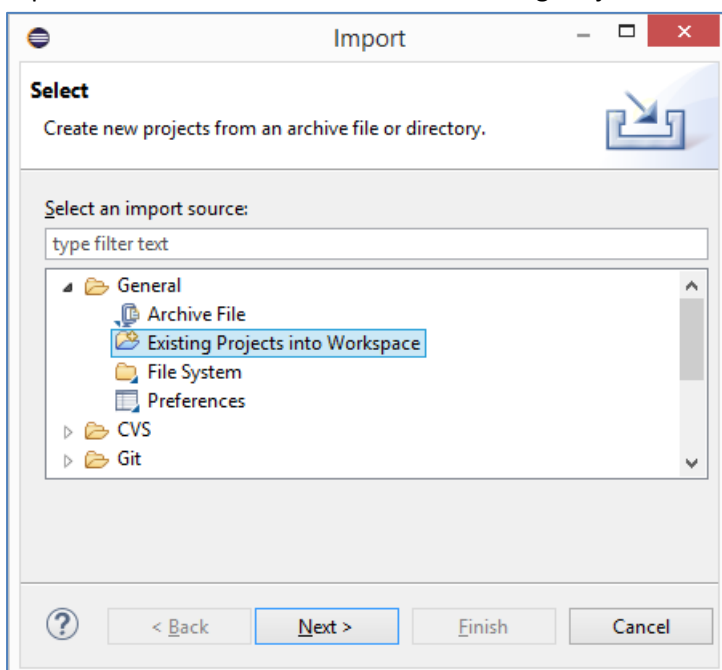
Now that the setup.bat file has completed, the project needs to be imported into eclipse.

Process

1. Start Eclipse and select your new workspace.
2. Click the File Menu and select Import...

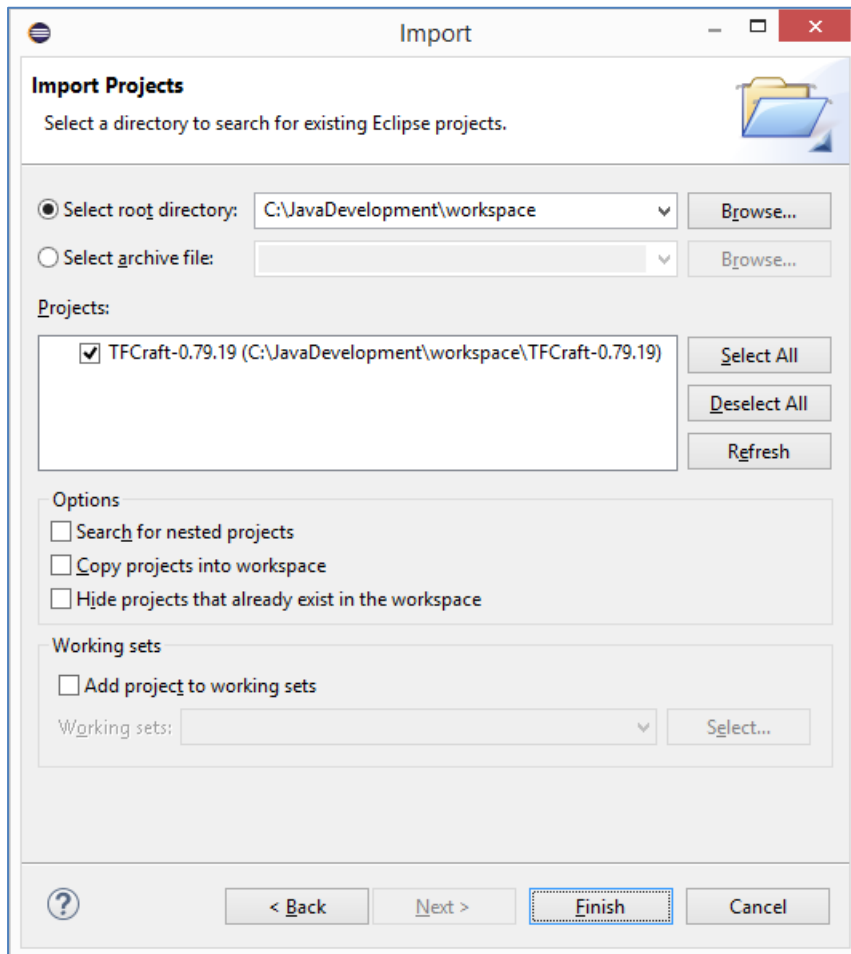


3. Expand the General folder and select 'Existing Projects into Workspace', then click next.

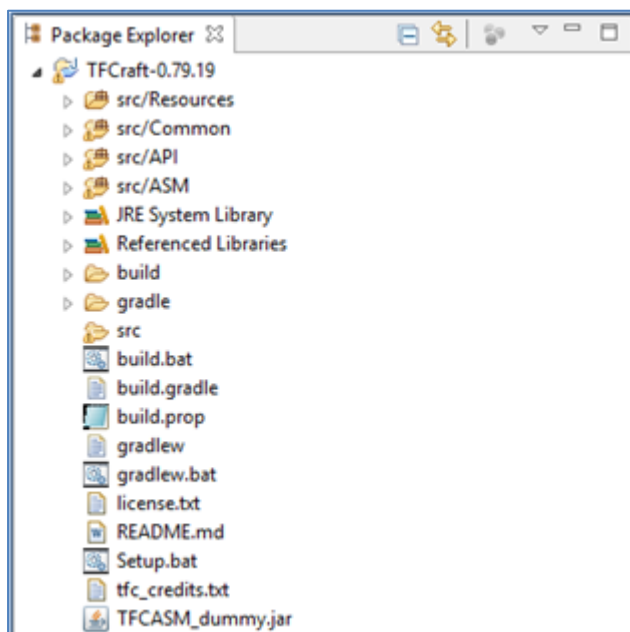


TFC (79) Add-on Development Environment Setup

- Click the Browse button and navigate to your workspace. The projects will be listed, make sure the TFC project is checked and click the finish button.

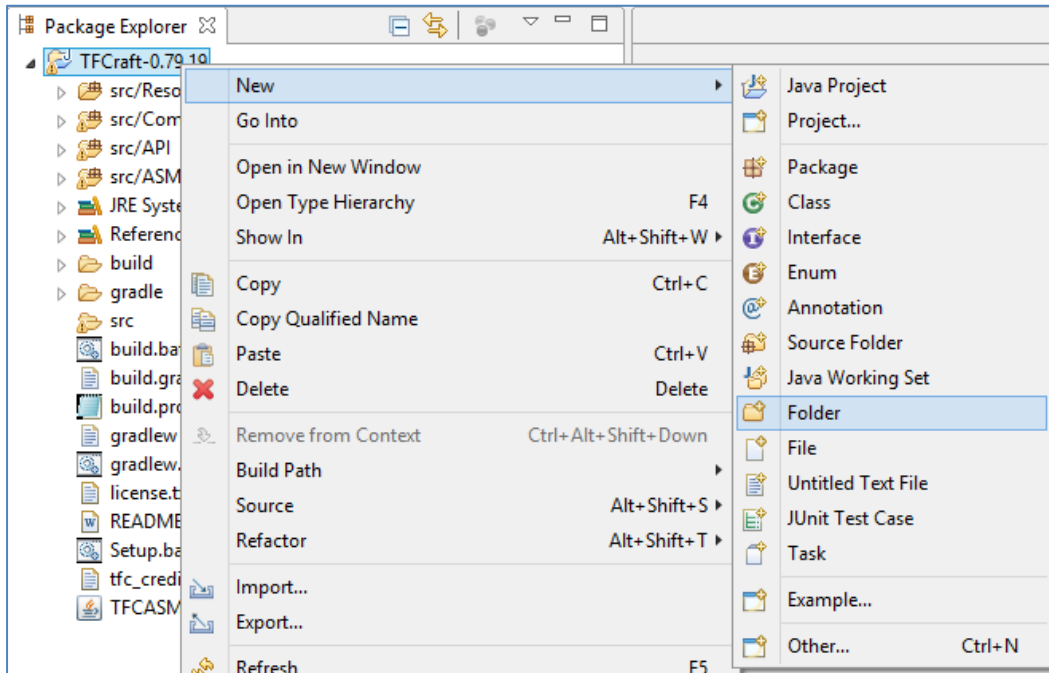


- The TFC project should now be imported into your workspace.

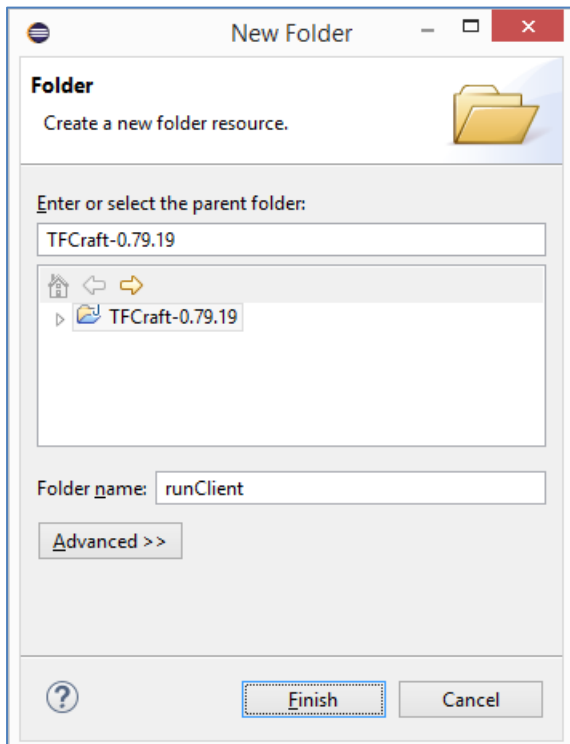


TFC (79) Add-on Development Environment Setup

6. Now we need to create 2 new folders, runClient and runServer. Right click on your project (TFCraft-0.79.19), go to the new menu, select folder.

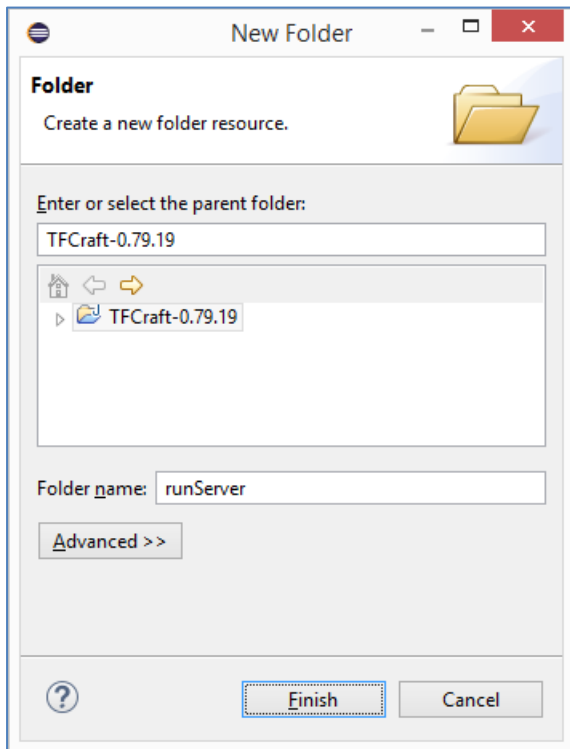


7. Select the parent folder and enter the folder name (runClient). Click the finish button.

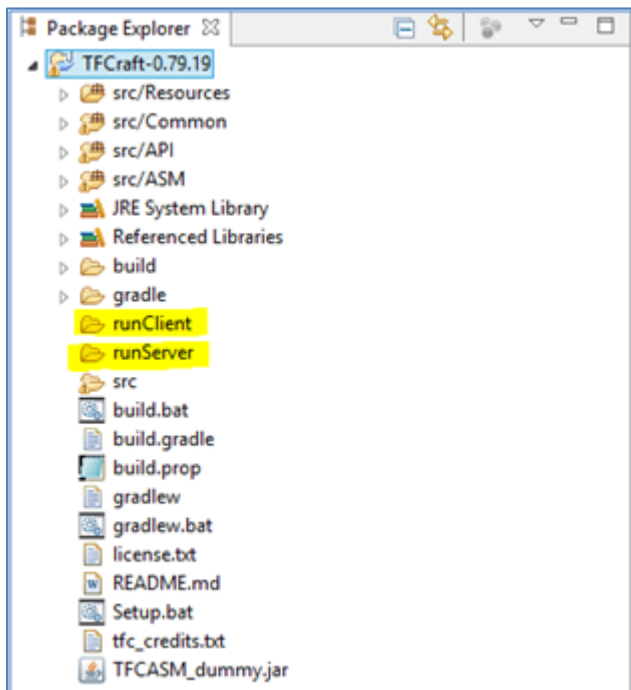


TFC (79) Add-on Development Environment Setup

8. Select the parent folder and enter the folder name (runServer). Click the finish button.



9. The two new folders should now be created.



TFC (79) Add-on Development

Environment Setup

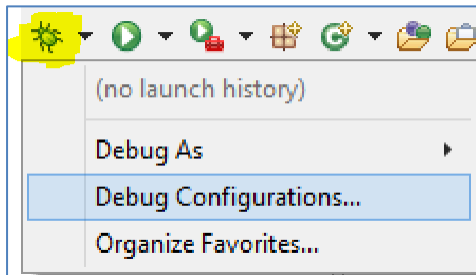
Creating Debug and Run Configurations

In order to debug or run TFC, configurations must first be setup. The configurations provide parameters and arguments to the Minecraft instance.

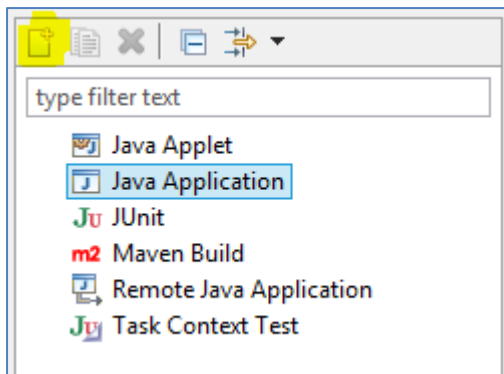
Process

Creating a client configuration

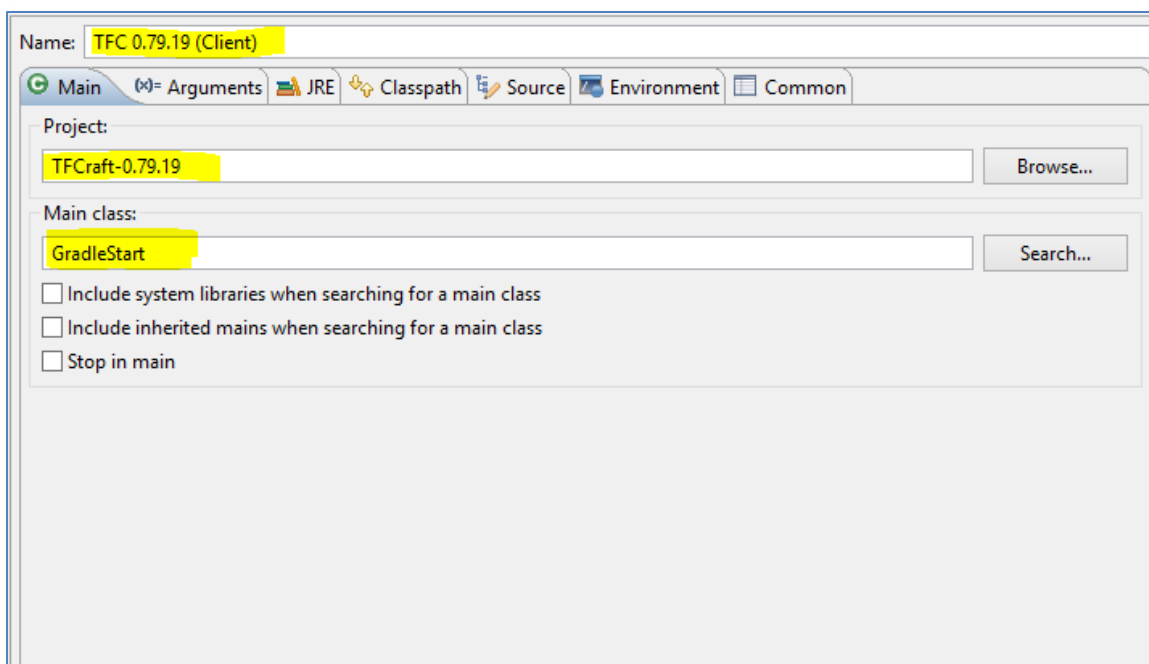
1. Open the configurations screen. Click the arrow next to the bug icon on the main toolbar. Select Debug Configurations...



2. Highlight the Java Application item in the list and click the new button.



3. Select the Main tab and enter the Name, select the Project (browse button) and select the Main class (browse button).



TFC (79) Add-on Development Environment Setup

4. Select the Arguments tab. Enter the working directory (adding /runClient to the end).
To use your own Minecraft username and password (instead of guest player), enter the following into the Program arguments and changing the values.
--username <your email address> --password <your password>

The screenshot shows the 'Arguments' tab of the 'TFC 0.79.19 (Client)' configuration window. The 'Program arguments' field contains '--username [redacted] --password [redacted]'. The 'VM arguments' field is empty. The 'Working directory' section has the 'Other' radio button selected, with the text field containing '\${workspace_loc:TFCraft-0.79.19/runClient}'. The 'Default' radio button is unselected. At the bottom right, there are buttons for 'Workspace...', 'File System...', and 'Variables...'.

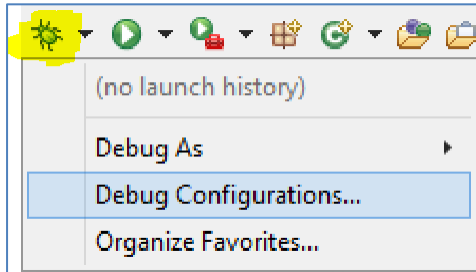
5. Click the apply button.

The client configuration has now been created.

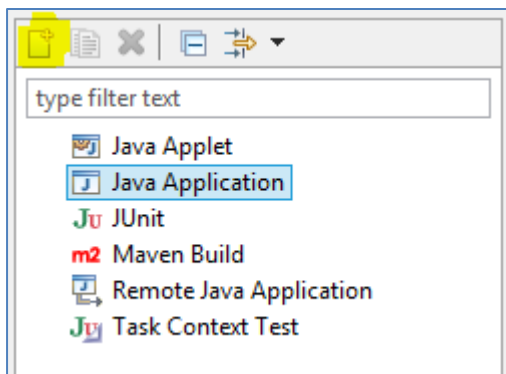
TFC (79) Add-on Development Environment Setup

Creating a server configuration

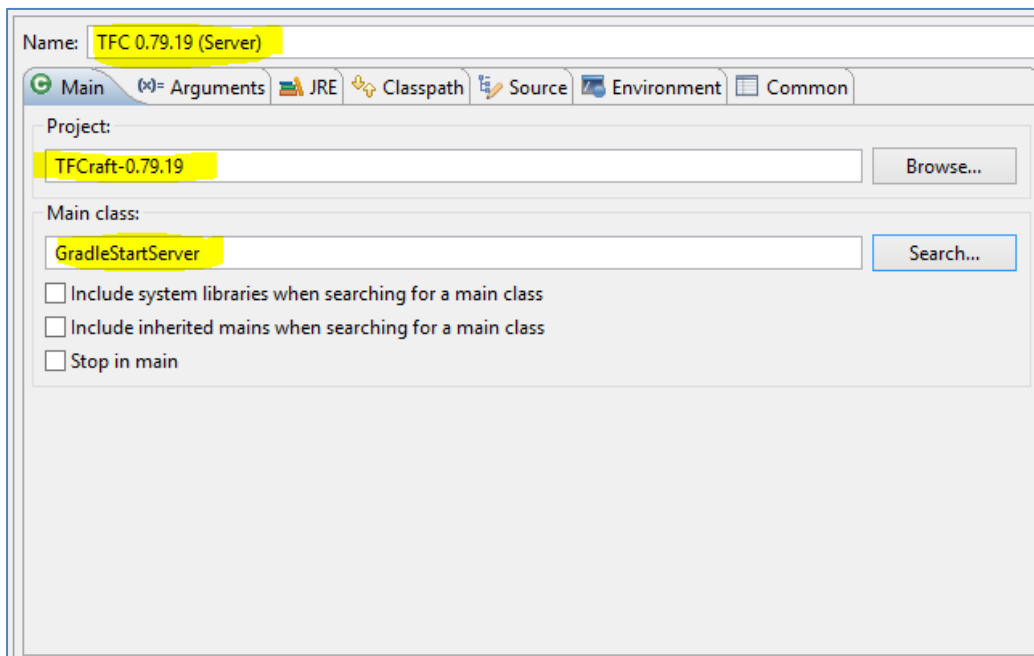
1. Open the configurations screen. Click the arrow next to the bug icon on the main toolbar. Select Debug Configurations...



2. Highlight the Java Application item in the list and click the new button.

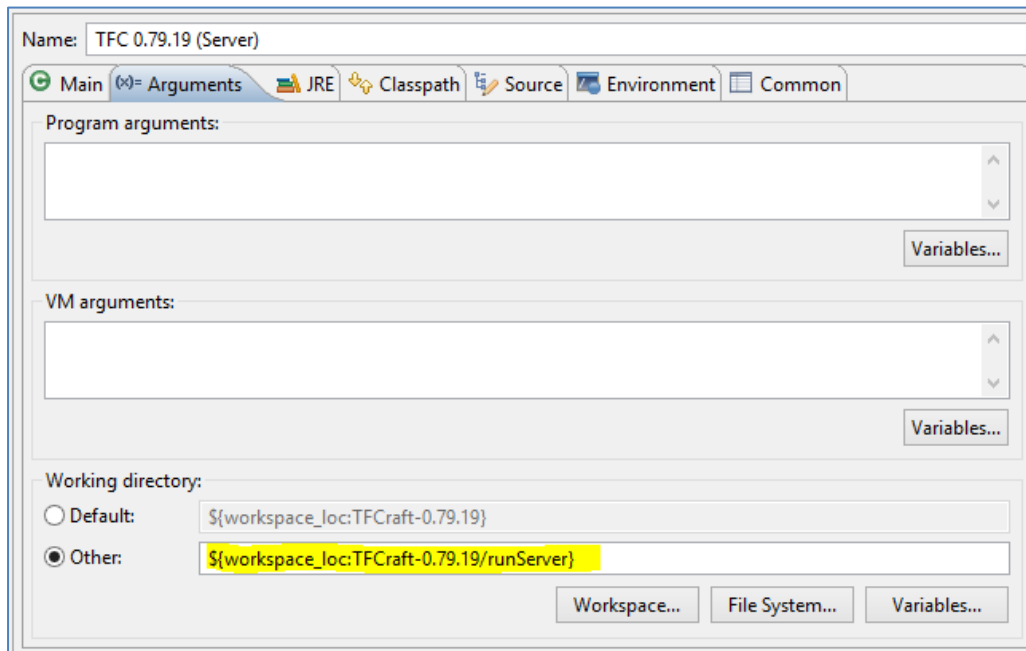


3. Select the Main tab and enter the Name, select the Project (browse button) and select the Main class (browse button).



TFC (79) Add-on Development Environment Setup

4. Select the Arguments tab. Enter the working directory (adding /runServer to the end).



5. Click the apply button.

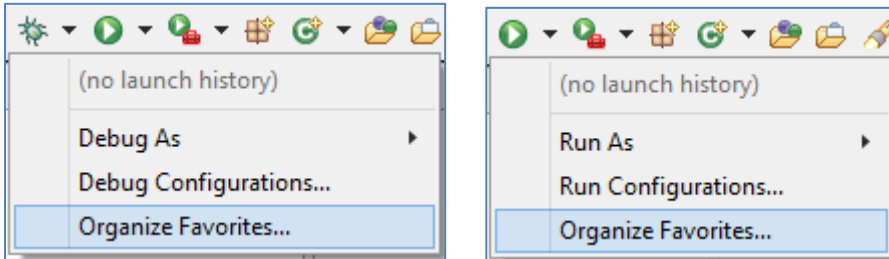
The server configuration has now been created.

TFC (79) Add-on Development

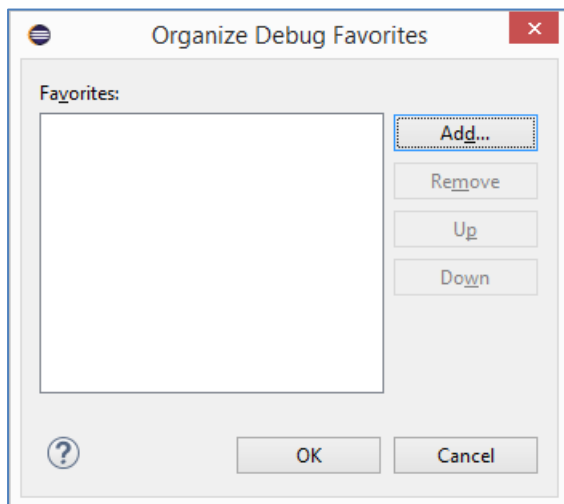
Environment Setup

Adding Configurations to Favourites

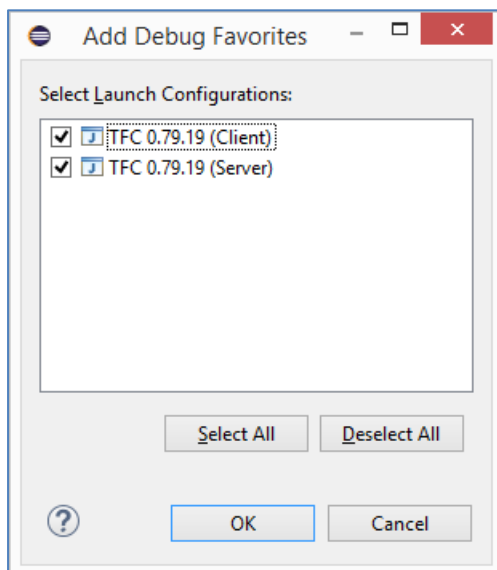
In order to access your configurations easily, add them to the Debug and Run favourites.



1. Open the favourites screen. Click the arrow next to the bug icon (debug) or the arrow icon (run) on the main toolbar. Select Organize Favourites...
2. Click the Add... button

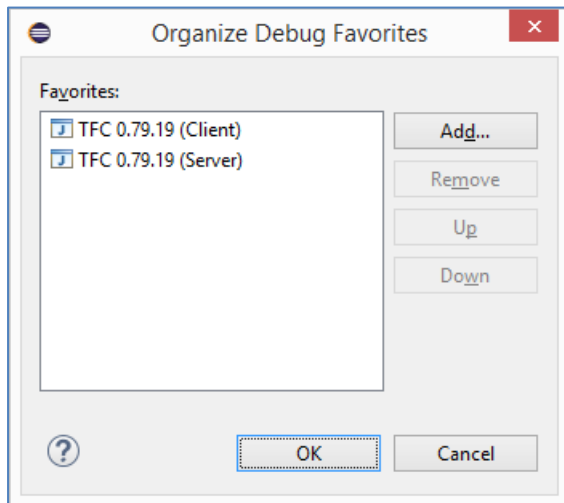


3. Check the configurations you want to add as favourites and click the OK button.

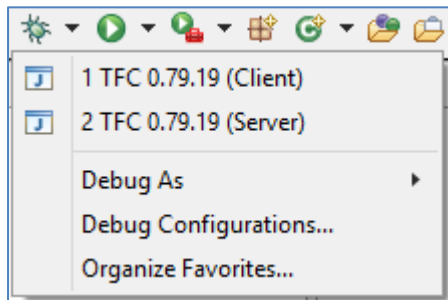


TFC (79) Add-on Development Environment Setup

- Click the OK button to accept your new favourite configurations.



- You should now see the configurations in the drop menu.



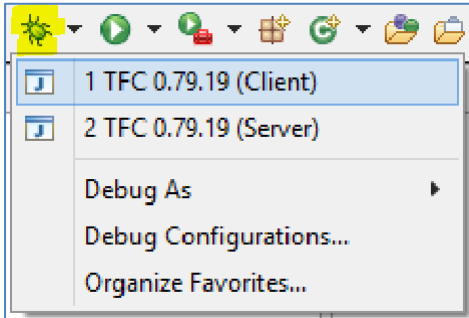
TFC (79) Add-on Development Environment Setup

Testing the TFC Project

Now to test if the TFC project will compile and run.

Process

1. Click the arrow next to the bug icon on the main toolbar. Select your Client configuration.



2. If all goes well, then Minecraft should have started up and you will be sitting on the main screen.



If you are unable to get Minecraft started at this stage, then you will need to go back to the start and check everything. It may even be necessary to start the entire process again.

TFC (79) Add-on Development

Environment Setup

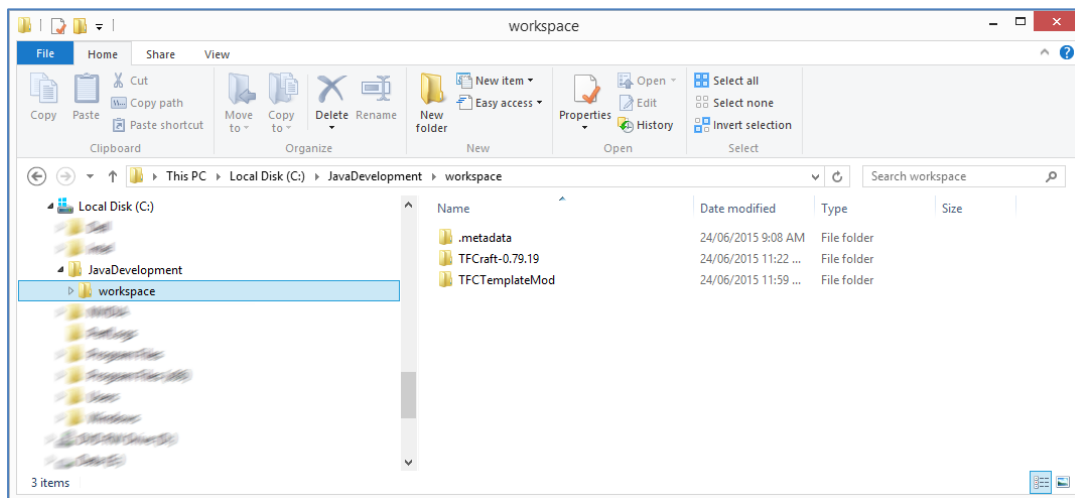
TFC Add-on

TFC Add-on Setup

Now we need to import and setup the template project. There will be some renaming to be done to customise for your own add-on.

Process

1. Locate the template zip file.
2. Unzip the file into your eclipse workspace folder that you created.
(e.g. C:\JavaDevelopment\workspace).



3. Rename the TFCTemplateMod to the name of your new add-on.
4. Navigate to the add-on folder.
5. Locate the setup.bat file and double-click it. This will start gradle and begin setting up your project, ready to be imported into eclipse. This step can take quite a while as it will be downloading libraries from internet repositories.

Note: On occasions I have found that some errors have occurred during this step. If this happens to you, try running setup.bat again. If the problem persists, then delete the new add-on folder and start this section again.

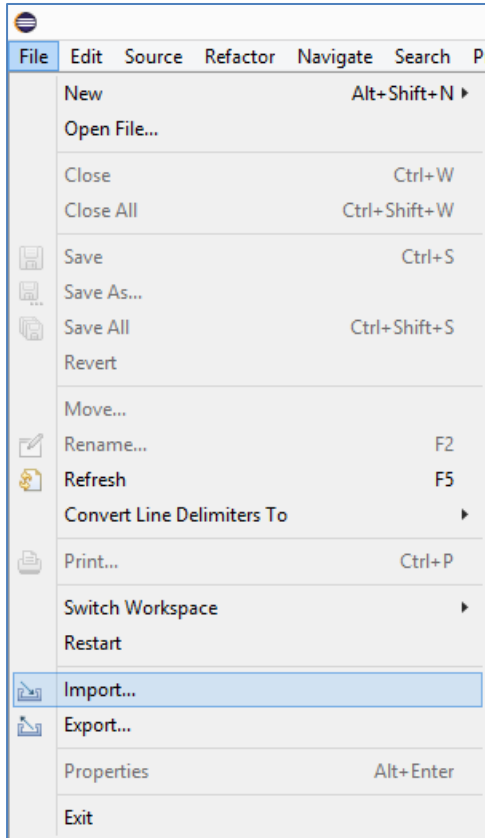
TFC (79) Add-on Development Environment Setup

Importing Your Add-on into Eclipse

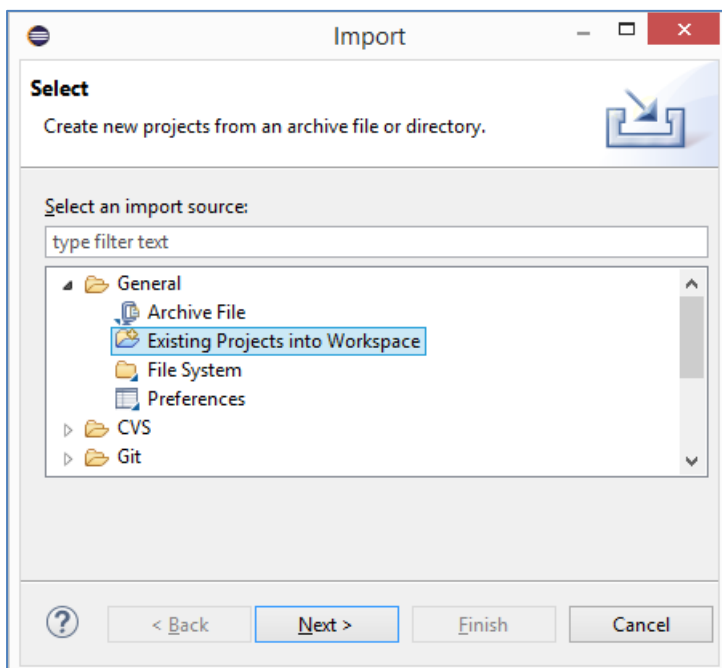
Now that the setup.bat file has completed, the project needs to be imported into eclipse.

Process

1. Start Eclipse and select your new workspace.
2. Click the File Menu and select Import...

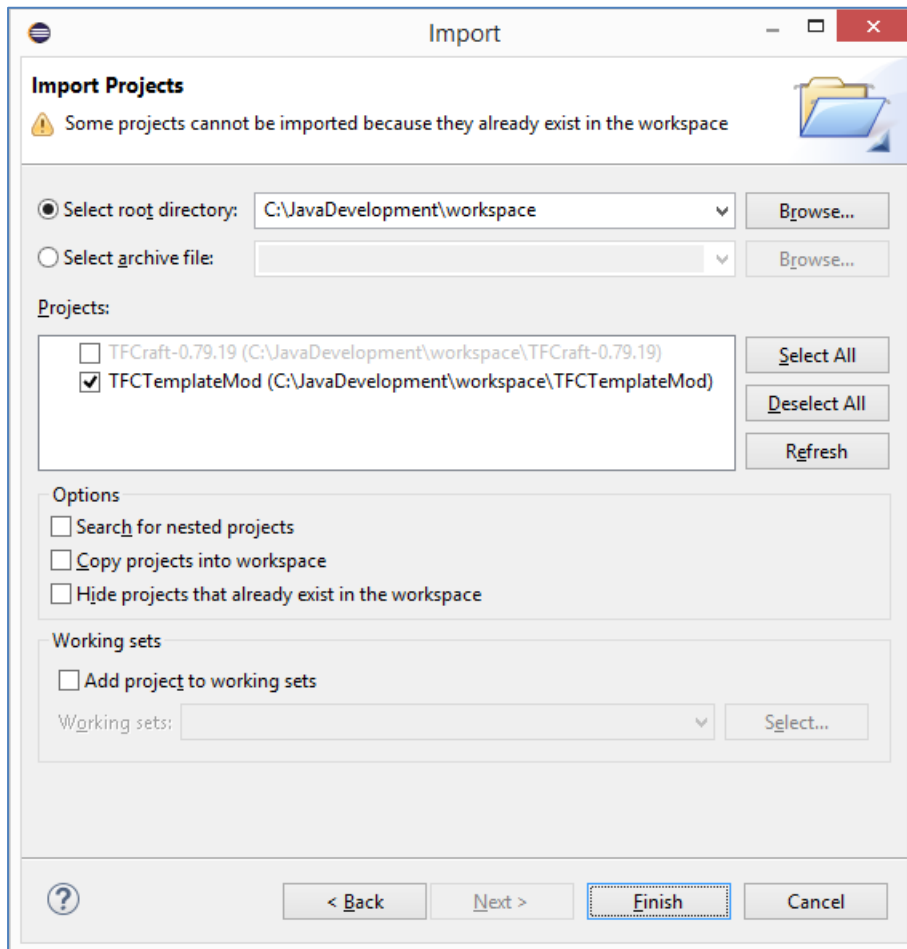


3. Expand the General folder and select 'Existing Projects into Workspace', then click next.



TFC (79) Add-on Development Environment Setup

- Click the Browse button and navigate to your workspace. The projects will be listed, make sure the TFC project is checked and click the finish button.



- The TFC project should now be imported into your workspace.

TFC (79) Add-on Development

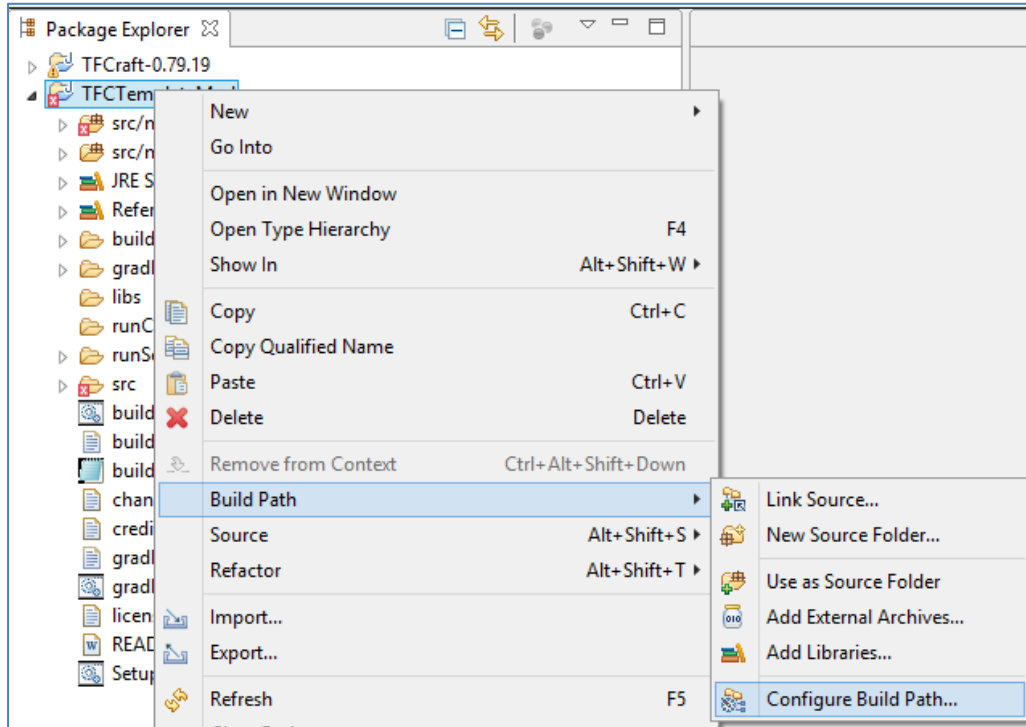
Environment Setup

Configure Add-on Build Settings

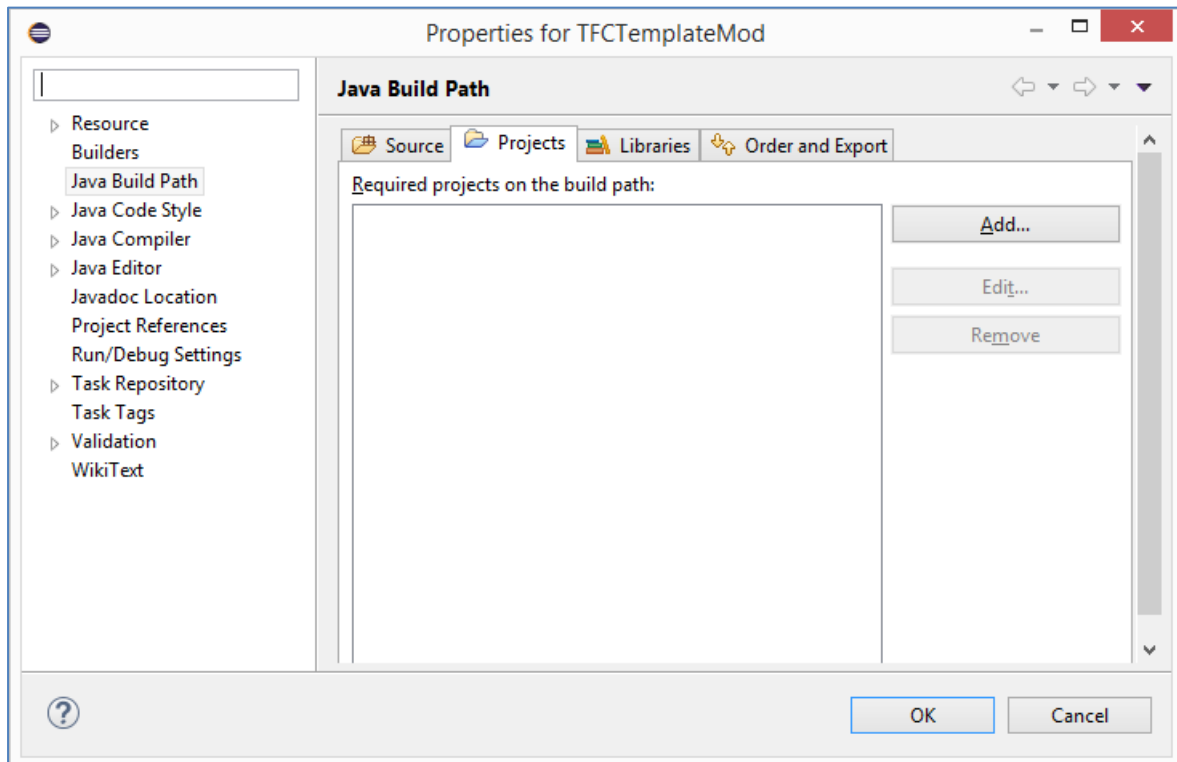
In order for your add-on to work with TFC in Debug and Run modes, TFC must be linked into the project.

Process

1. Right click on your add-on project, go to the Build Path menu, and select Configure Build Path.

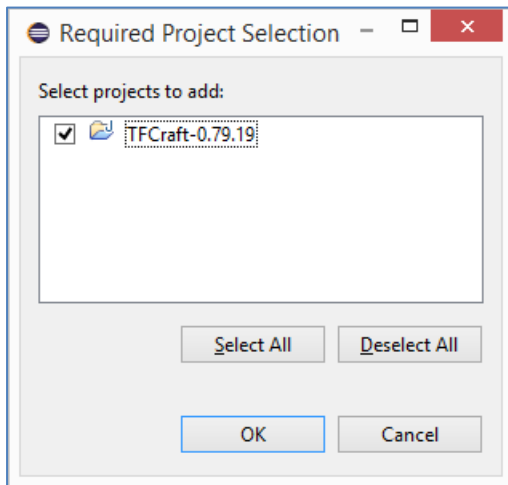


2. Change to the Projects tab and click the Add... button.

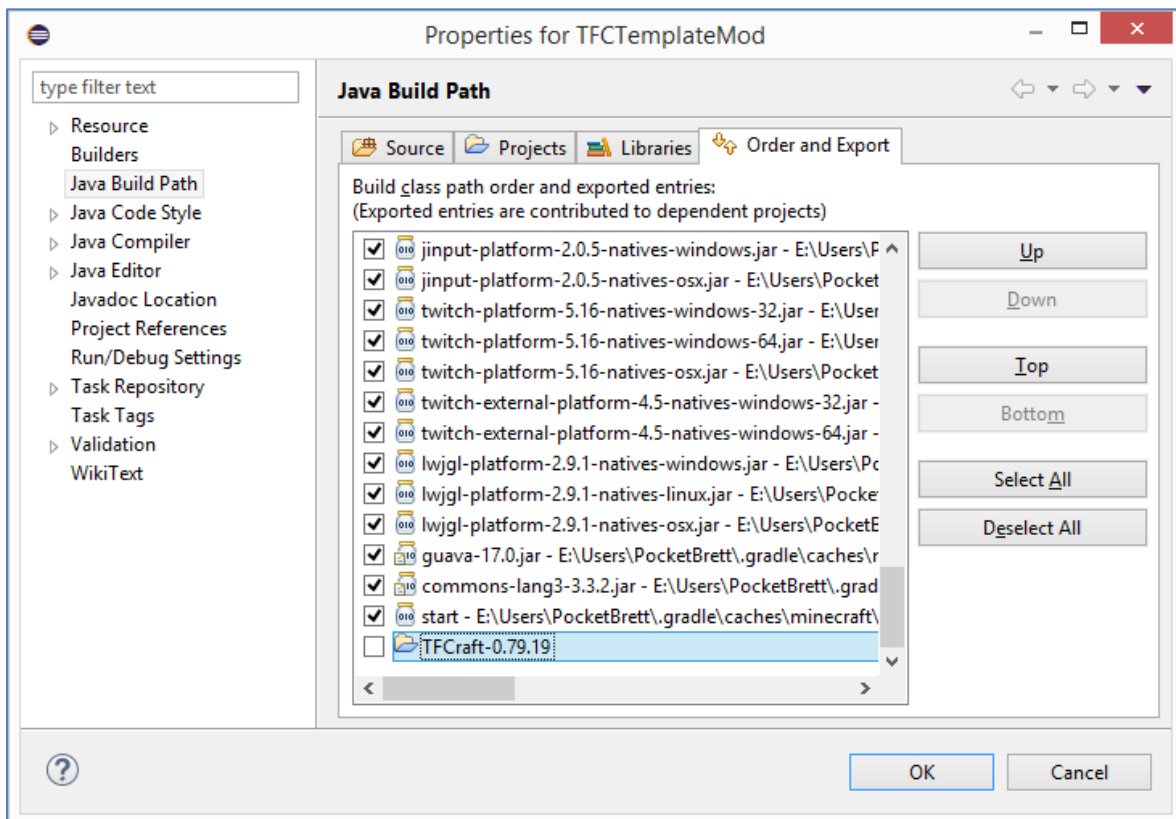


TFC (79) Add-on Development Environment Setup

3. Check the TFC project and click the OK button.



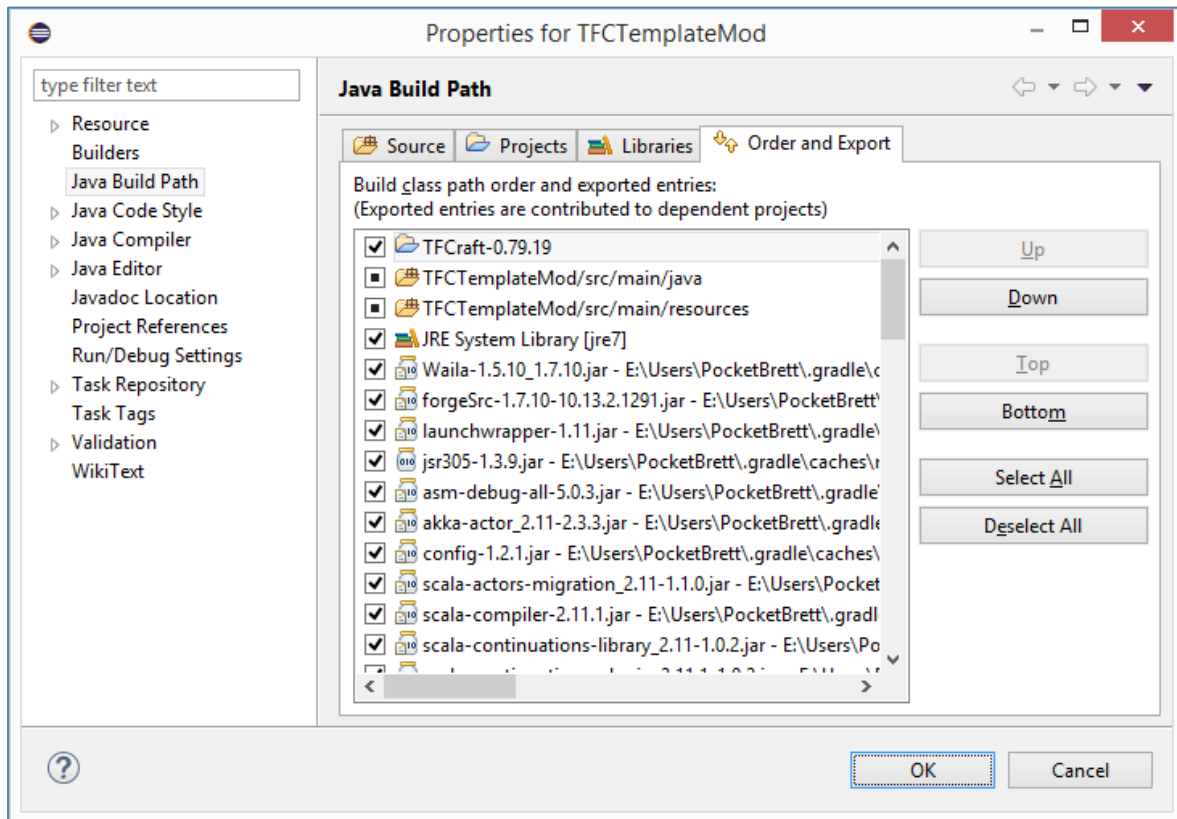
4. Change to the Order and Export tab, scroll to the bottom and locate the TFC project item.



5. Click the Top button and then scroll to the top of the list.

TFC (79) Add-on Development Environment Setup

6. Check the TFC project item and click the OK button.



TFC (79) Add-on Development

Environment Setup

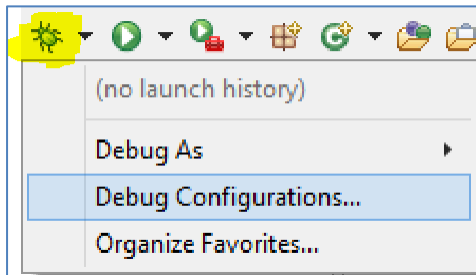
Creating Debug and Run Configurations

In order to debug or run your add-on, configurations must first be setup. The configurations provide parameters and arguments to the Minecraft instance.

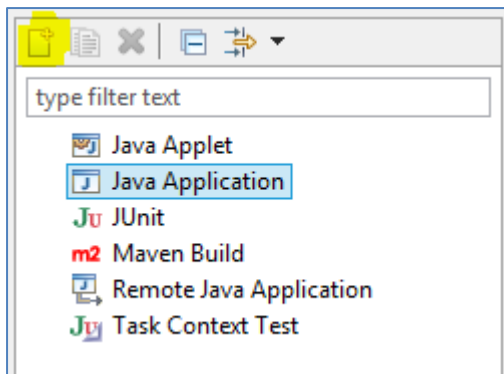
Process

Creating a client configuration

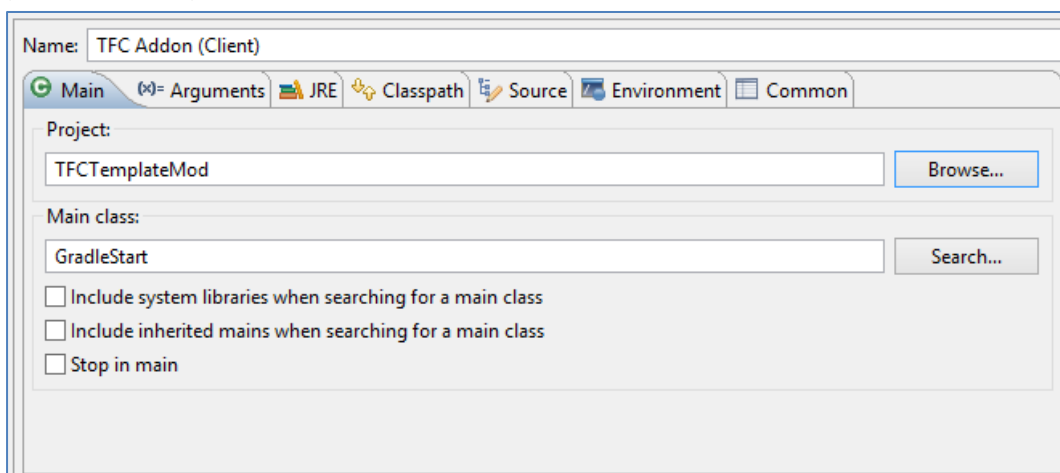
6. Open the configurations screen. Click the arrow next to the bug icon on the main toolbar. Select Debug Configurations...



7. Highlight the Java Application item in the list and click the new button.



8. Select the Main tab and enter the Name, select the Project (browse button) and select the Main class (browse button).

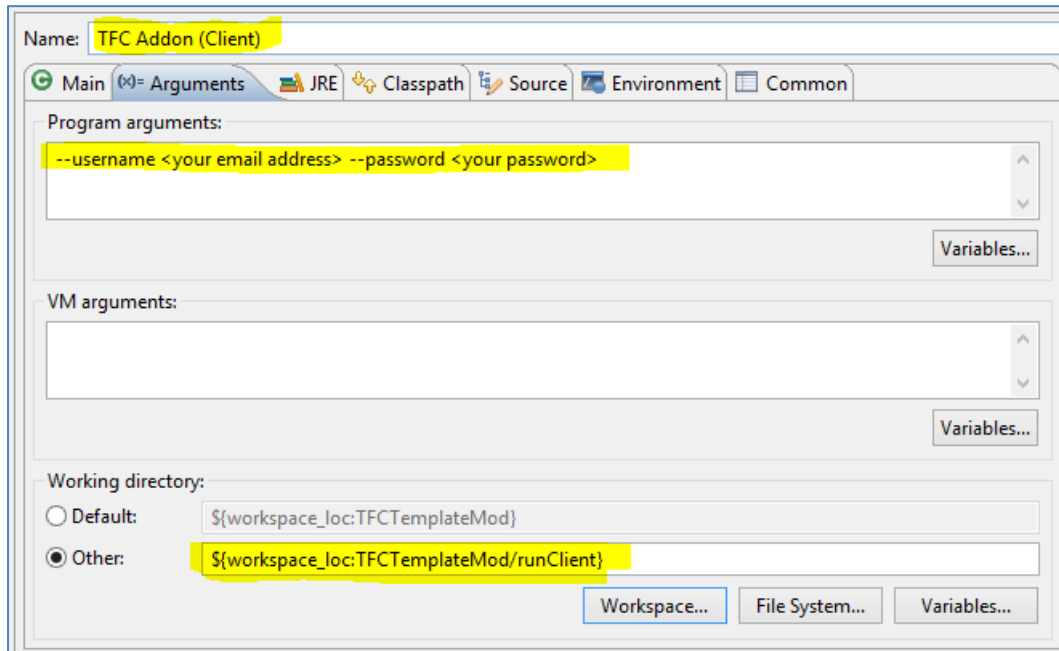


TFC (79) Add-on Development Environment Setup

9. Select the Arguments tab. Enter the working directory (adding /runClient to the end).

To use your own Minecraft username and password (instead of guest player), enter the following into the Program arguments and changing the values.

--username <your email address> --password <your password>



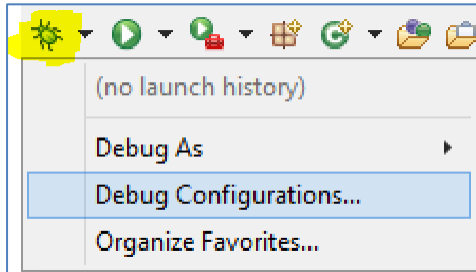
10. Click the apply button.

The client configuration has now been created.

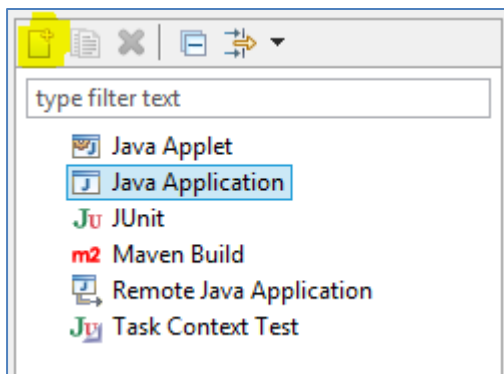
TFC (79) Add-on Development Environment Setup

Creating a server configuration

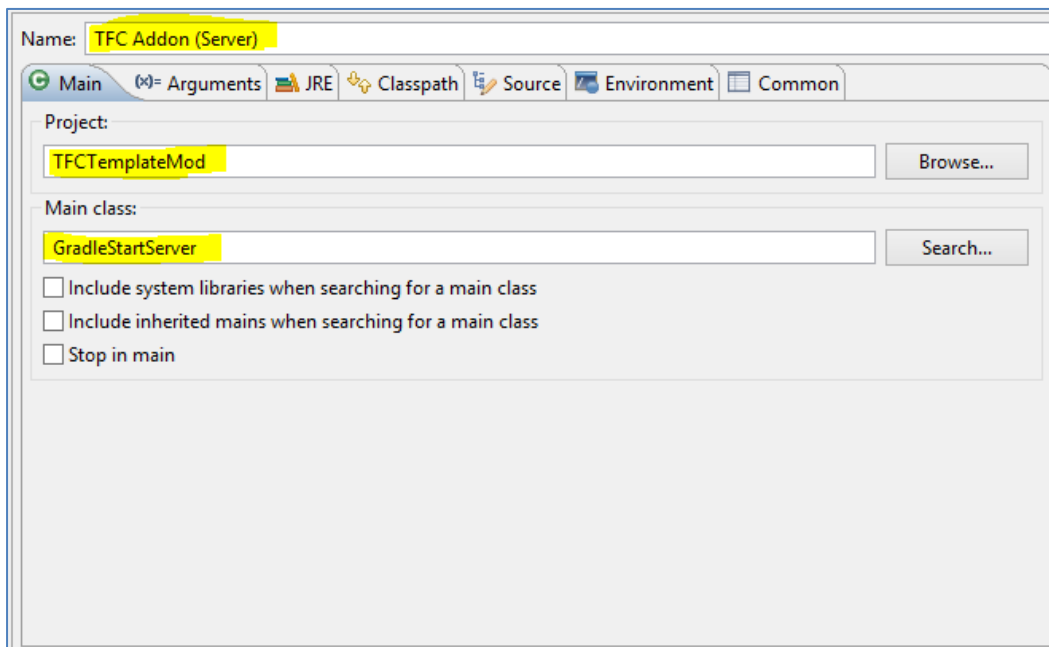
6. Open the configurations screen. Click the arrow next to the bug icon on the main toolbar. Select Debug Configurations...



7. Highlight the Java Application item in the list and click the new button.



8. Select the Main tab and enter the Name, select the Project (browse button) and select the Main class (browse button).



TFC (79) Add-on Development Environment Setup

9. Select the Arguments tab. Enter the working directory (adding /runServer to the end).

The screenshot shows the 'Arguments' tab of the 'TFC Addon (Server)' configuration window. The 'Name' field is 'TFC Addon (Server)'. The 'Program arguments' and 'VM arguments' fields are empty, each with a 'Variables...' button to its right. The 'Working directory' section has two radio buttons: 'Default' (selected) and 'Other'. The 'Other' option is selected, and its text field contains the path `${workspace_loc:TFCTemplateMod/runServer}`, which is highlighted in yellow. At the bottom right are three buttons: 'Workspace...', 'File System...', and 'Variables...'.

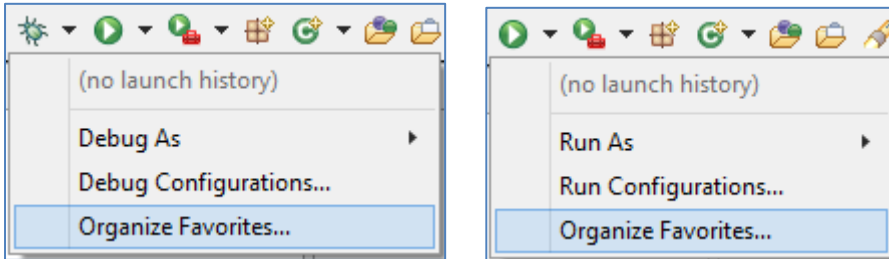
10. Click the apply button.

The server configuration has now been created.

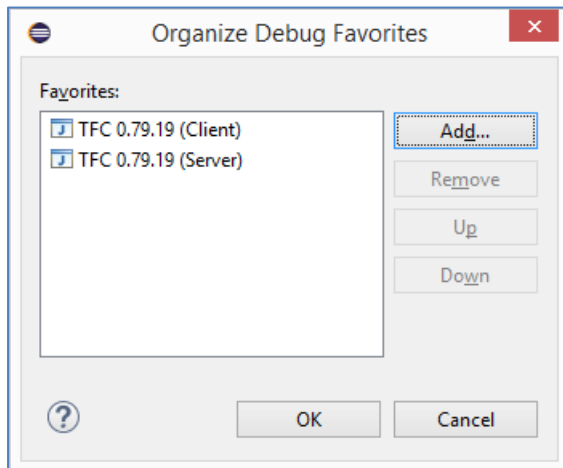
TFC (79) Add-on Development Environment Setup

Adding Configurations to Favourites

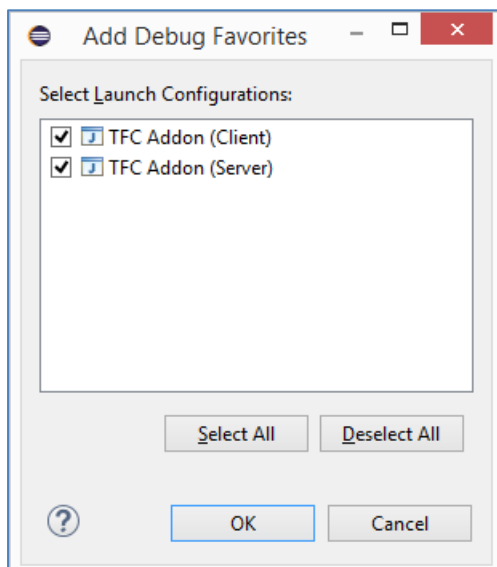
In order to access your configurations easily, add them to the Debug and Run favourites.



6. Open the favourites screen. Click the arrow next to the bug icon (debug) or the arrow icon (run) on the main toolbar. Select **Organize Favourites...**
7. Click the **Add...** button



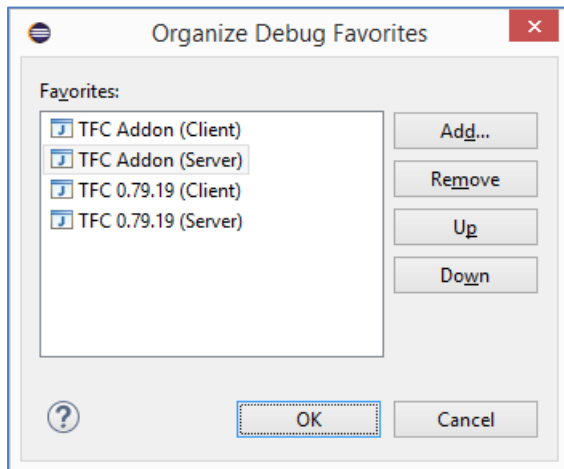
8. Check the configurations you want to add as favourites and click the **OK** button.



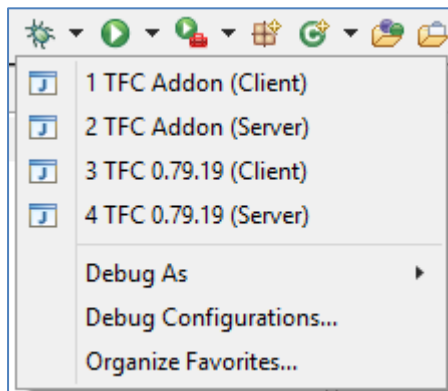
TFC (79) Add-on Development

Environment Setup

9. Click the OK button to accept your new favourite configurations.



10. You should now see the configurations in the drop menu.



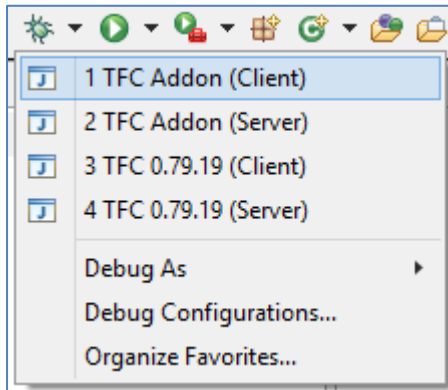
TFC (79) Add-on Development Environment Setup

Testing the Add-on Project

Now to test if the TFC project will compile and run.

Process

1. Click the arrow next to the bug icon on the main toolbar. Select your Client configuration.



2. If all goes well, then Minecraft should have started up and you will be sitting on the main screen.



If you are unable to get Minecraft started at this stage, then you will need to go back to the start and check everything. It may even be necessary to start the entire process again.

TFC (79) Add-on Development

Environment Setup

Add-on Customisation

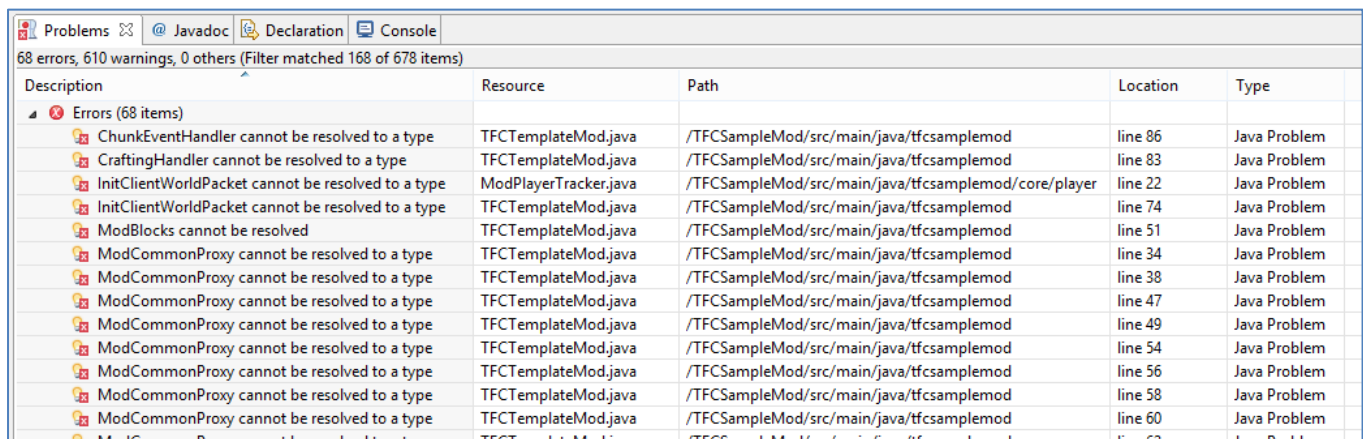
Namespace Change

As a template add-on project was used, all the project namespaces are tfctemplatemod. You must now change them all to reflect the name of the add-on.

There are 2 folders and 1 file that must be renamed.

1. src/main/java/tfctemplatemod
2. src/main/resources/assets/tfctemplatemod
3. src/main/java/tfcsamplemod/TFCTemplateMod.java

Both folders must be renamed to the name of your add-on. After this is done, most, if not all, of the java classes will have problems. The best way to fix these problems within Eclipse is using the Problems Form. The problems tab is located along the bottom set of tabs.



Description	Resource	Path	Location	Type
Errors (68 items)				
ChunkEventHandler cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 86	Java Problem
CraftingHandler cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 83	Java Problem
InitClientWorldPacket cannot be resolved to a type	ModPlayerTracker.java	/TFCSampleMod/src/main/java/tfcsamplemod/core/player	line 22	Java Problem
InitClientWorldPacket cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 74	Java Problem
ModBlocks cannot be resolved	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 51	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 34	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 38	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 47	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 49	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 54	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 56	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 58	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 60	Java Problem
ModCommonProxy cannot be resolved to a type	TFCTemplateMod.java	/TFCSampleMod/src/main/java/tfcsamplemod	line 62	Java Problem

Double click on each of the problems and you should be able to fix them in the code. Most of the issues will be because of the namespace. You can also open each java file and change the namespace at the top of each file.

TFC (79) Add-on Development

Environment Setup

Proxy Files

There are two proxy files that are referenced by the main java class (TFCTemplateMod.java). The proxy files are referenced using string values and do not cause a problem to appear in the Problems window, but you will get a runtime error:

cpw.mods.fml.common.LoaderException: java.lang.ClassNotFoundException: tfctemplatemod.core.ModClientProxy

To fix these errors open the ModDetails.java (located in the core folder) and change these two lines.

```
public static final String SERVER_PROXY_CLASS = "tfctemplatemod.core.ModCommonProxy";  
public static final String CLIENT_PROXY_CLASS = "tfctemplatemod.core.ModClientProxy";
```

Just change the `tfctemplatemod` to match the name of your mod.

Add-on Details

There are multiple files that contain the details for the add-on.

1. build.prop
2. src/main/java/ tfctemplatemod /core/ModDetails.java
3. src/main/resources/mcmod.info

Each of these files needs to be changed to reflect the name of your add-on. You may also want to take a little time trying to understand these files and the information they contain.

TFC (79) Add-on Development

Environment Setup

Building Your Add-on for Distribution

Finally you have arrived at the place where all the hard work is about to begin, writing your add-on. Using the folder structure within your new add-on project should help you organise your classes.

But what happens once you are finished? How do I compile my JAR file so I can distribute it to the community?

Building TFC

First we must build TFC. As your add-on references it, we need to build it and copy the deobfuscated version into your add-ons project structure.

Process

1. Open windows explorer and navigate to your workspace (e.g. C:\JavaDevelopment\workspace).
2. Open the TFCraft-0.79.x folder.
3. Locate the build.bat file and double-click it. This will start gradle and begin building the TFC project. This step can take quite a while.
4. Once the build is complete, navigate to the build/libs folder under the TFCraft-0.79.x folder.
5. Locate the .jar file - [1.7.10]TerraFirmaCraft-deobf-0.79.19.jar. **MUST BE the deobf file.**
6. Copy the file to your add-on project's libs folder (e.g. C:\JavaDevelopment\workspace\TFCTemplateMod\libs).

Building Your Add-on

Now that TFC has been built and the DEOBF file has been copied into your project folder, you can build your add-on. This can be done in either Windows Explorer (like TFC) or in Eclipse. I have modified the Build.bat file so that it can be run within Eclipse.

Process – Windows Explorer

1. Open windows explorer and navigate to your workspace (e.g. C:\JavaDevelopment\workspace).
2. Open your add-on folder.
3. Locate the build.bat file and double-click it. This will start gradle and begin building the add-on project. This step can take quite a while.
4. Once the build is complete, navigate to the build/libs folder.
5. Locate the .jar file - [1.7.10]<your add-on name>-0.01.01.jar. This is the file you can distribute.
DO NOT use the deobf or src file, that is only for development purposes.

Process – Eclipse

This build is done within Eclipse. If the folder does not exist, just right-click your project node and select Refresh.

1. Locate the Build.bat file in the project structure.
2. Double-click the file. This will start gradle and begin building the add-on project. This step can take quite a while.
3. Once the build is complete, navigate to the build/libs folder in the project structure.
4. Locate the .jar file - [1.7.10]<your add-on name>-0.01.01.jar. This is the file you can distribute.
DO NOT use the deobf or src file, that is only for development purposes.