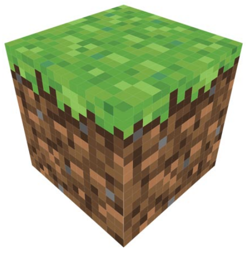
B0 - HUB

B-INN-000

Minecraft Server & API

Setting up your first minecraft server



1.1

Minecraft Server & AP

binary name:	no need here
repository name:	no need here
language:	bash, java
compilation:	no need here
build tool:	no need here

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Spigot is an open-source Java project that lets users run their own Minecraft server and add plugins to extend the possibilities of their server. There are over 100,000 Spigot servers in existence today. This makes Spigot one of the most stable and diverse Minecraft servers available.

The goal of this project is the following, to set up a Minecraft server in 1.9.4 and then to make a plugin that will:

- Change the connection message in the chat
- Change the disconnection message in the chat
- Teleport the player when connecting to a defined location
- Set up a command for this teleporter to the formerly defined location
- To go further ... Make a command to store the location and then be able to use the command to teleport there
- And more ...

Server Part

Part 1 - Setup the Minecraft Server

Be careful, during the first launch, a file: **eula.txt** will be generated. You need to open it and pass the eula value to *true*. After that you can restart the server normally

```
Terminal
~/B-INN-000> java -jar server.jar
Loading libraries, please wait...
Starting minecraft server version 1.9.4
Loading properties
...
```

Once the server is up and running, your terminal will become the server console, you can enter commands that will affect the server directly.
Here is a small list of the most important commands for this workshop :

- reload : This command reloads the server (plugins, server properties ...)
- stop : This command stops the server
- help : This command shows you all commands who are available

Part 2 - Config the Minecraft Server

If you don't have an official version of Minecraft, open server.properties and change the value of **online-mode** to *false*

The basic server configuration will be used for this activity, if you want to change the settings shutdown the server with the *stop* command if it is not already the case and modify the file **server.properties**.

Part 3 - Try to connect to your Minecraft Server

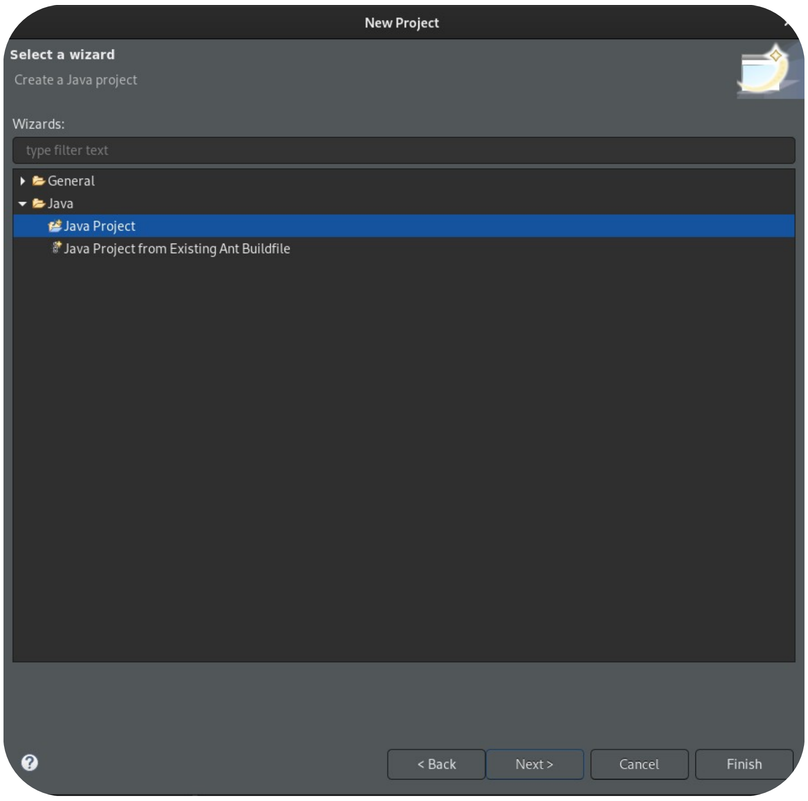
Once the *java -jar server.jar* command has been launched, the server is ready to welcome you, by default the server uses port *25565* and your *local IP*.
Open your game and go to the multiplayer part, add a new server, register a server name and for the IP just type "**localhost**".

Dev Part

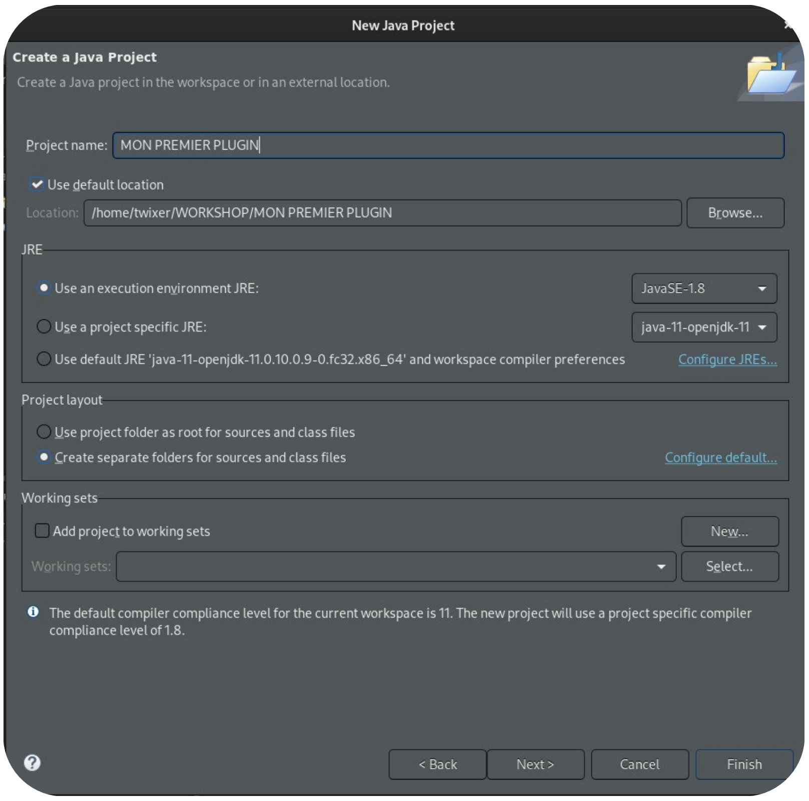
Part 1 - Open Eclipse IDE

A workspace is the place where your files will be generated, remember, the rendering of the project will be in this file !

Open Eclipse and choose your workspace, press Launch to open it.
Close the home page and in the Project explorer tab, press create a project, choose **Java Project** from the list.



Give a *name* to your project
Check that the **Use default location** box *is checked*.
In the **JRE** section select Use an execution environment JRE: **javaSE-1.8**



New Java Project

Create a Java Project

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

Project name: MON PREMIER PLUGIN

[✓] Use default location

Location: /home/twixer/WORKSHOP/MON PREMIER PLUGIN

JRE

[●] Use an execution environment JRE: JavaSE-1.8

[○] Use a project specific JRE: java-11-openjdk-11

[○] Use default JRE 'java-11-openjdk-11.0.10.0.9-0.fc32.x86_64' and workspace compiler preferences

Project layout

[○] Use project folder as root for sources and class files

[●] Create separate folders for sources and class files

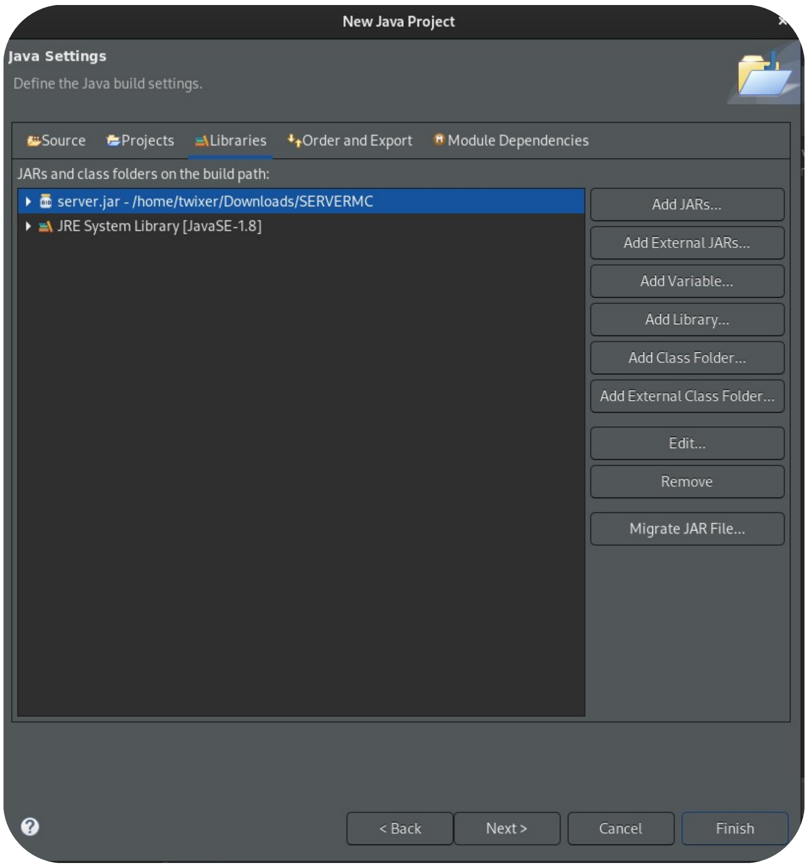
Working sets

[ ] Add project to working sets

Working sets:

The default compiler compliance level for the current workspace is 11. The new project will use a project specific compiler compliance level of 1.8.

Press next.
In Libraries added an external jars, and select jar of the minecraft server used previously.
press finish


New Java Project

Java Settings

Define the Java build settings.

JARs and class folders on the build path:

server.jar - /home/twixer/Downloads/SERVERMC
JRE System Library [JavaSE-1.8]

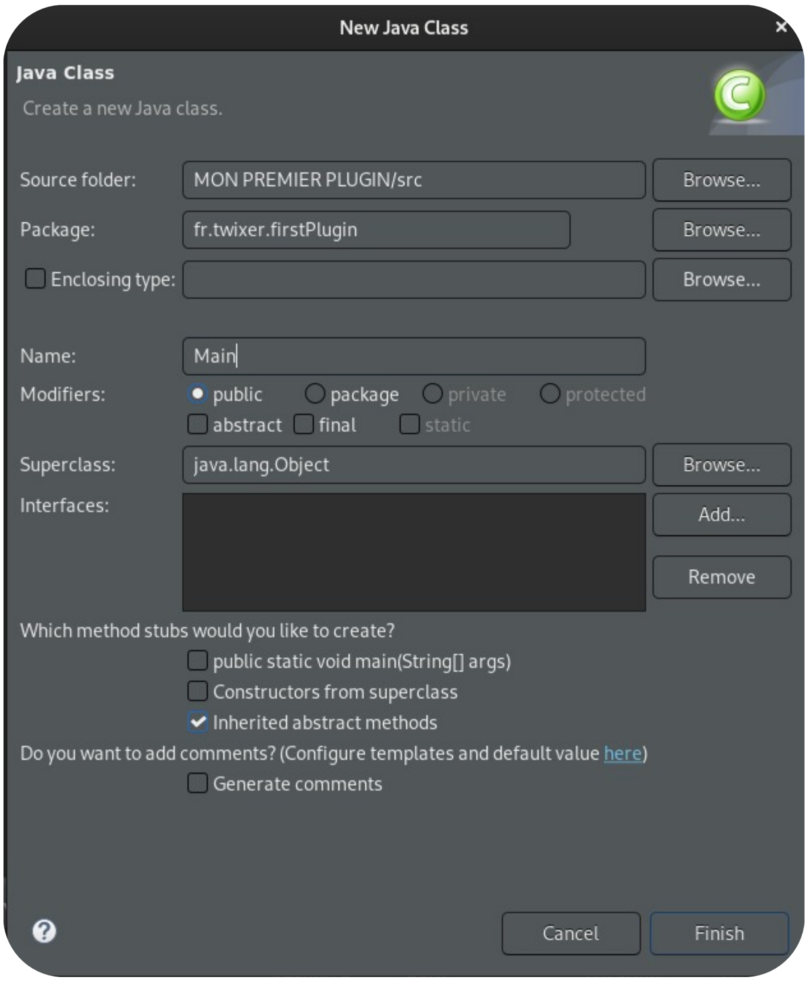
PART 2 - CREATE THE FIRST PACKAGE, CLASS AND SPIGOT.YML

Right click on your project located in Package explorer, then: new -> Package
Name it as you wish but in general we name it like this:

My syntax : **[nationality_code].[your_name].[project_name]**

Example : **fr.twixer.firstPlugin**

Right click on the package you just created and do : **new** -> **Class** Create the class **Main**



New Java Class

Java Class

Create a new Java class.

Source folder: MON PREMIER PLUGIN/src

Package: fr.twixer.firstPlugin

[ ] Enclosing type:

Name: Main

Modifiers: (●) public ( ) package ( ) private ( ) protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

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

Now we must indicate to java where your main is, and provide some information on the plugins being created.
Right click on your project located in the Package Explorer -> New -> File and set the name to **plugin.yml**
You will then have to fill in the following information:

```
name: My Magical Minecraft Plugin
version: 1.0.2.3.42
description: This plugin is very fun :D.
author: Twixer
main: [nationality_code].[your_name].[project_name].[Main_Class]
```

Part 3 - First line of your code

Now we are going to implement the Spigot API in the Main class we just created

```
package fr.twixer.firstPlugin;

import org.bukkit.plugin.java.JavaPlugin;

public class Main extends JavaPlugin {

}
```

Now we are going to implement the two functions which will establish the base of the plugin. The first will run when the server is started, and the next when the server is stopped.

Now we are going to implement the two functions which will establish the base of the plugin. The first will run when the server is started, and the next when the server is stopped.

```
package fr.twixer.firstPlugin;

import org.bukkit.plugin.java.JavaPlugin;

public class Main extends JavaPlugin {

    @Override
    public void onEnable() {

    }

    @Override
    public void onDisable() {

    }

}
```

Now you can put a message in both functions to see if the plugin works :

```
package fr.twixer.firstPlugin;

import org.bukkit.plugin.java.JavaPlugin;

public class Main extends JavaPlugin {

    @Override
    public void onEnable() {
        System.out.println("Le plugin a bien demmare ! ");
    }

    @Override
    public void onDisable() {
        System.out.println("Le plugin c'est bien etain ! ");
    }

}
```

Part 4 - Compile your project

We will now compile your project, to do this, right click on your project -> Export -> Java -> JAR File -> Next
In “Select the export location” go to the root of the server in the plugin folder.
And press Finish.

Part 5 - Reload the serveur

The NullPointerException (NPE) occurs when you declare a variable but did not create an object and assign it to the variable before trying to use the contents of the variable (called dereferencing). So you are pointing to something that does not actually exist.

You can **reload** or **stop / start** the server.
Then look in the console if the message put in the plugin is displayed.
If this is the case you can skip the next step otherwise you will have the error which will appear instead of your message.
Try to understand this error and fix it yourself, otherwise contact the person who runs the workshop.

Let's add functionality to the server

This is a list of all event you can use.

Part 1 - Change the connection message

In the server start function, we will add this line :

```
getServer().getPluginManager().registerEvents(new ConnectionEvent(), this);
```

Mouse over **ConnectionEvent** and do: **Create Class: ConnectionEvent**
We are going to structure the code a bit, modify the **Package** to :

```
[nationality_code].[Your_name].[Project_name].events
```

And paste this method in it:

```
    @EventHandler
    public void onJoin(PlayerJoinEvent e) {
        e.setJoinMessage("Welcome, " + e.getPlayer().getName());
    }
```

Here the **onJoin** is customizable, you put what you want to define the method and be able to find yourself later.
PlayerJoinEvent is the name of the event in Spigot's API, so we call it here to do an action when a player joins the server and the variable **e** allows us to interact with this event.

Part 2 - Change the leave message

Using the link given at the top, find the event allowing you to interact when disconnecting the player and modify the message

Part 3 - Teleport the player when connecting

Use **new Location ()** to define a location in the game

By using the link given above make sure that when the player connects, its automatically teleports to the location created

Part 4 - Create an command that teleports you to the location

In the spigot.yml file you must therefore add this:

```
commands:
 flagrate:
```

In the function that starts when starting the added server:

```
getCommand("spawn").setExecutor(new CommandSpawn());
```

Mouse over **CommandSpawn** and do: **Create Class: CommandSpawn**
We are going to structure the code a bit, modify the **Package** to :

```
[nationality_code].[Your_name].[Project_name].commands
```

Change :
* *arg0* to *sender*
* *arg1* to *command*
* *arg2* to *label*
* *arg3* to *args*
We will only use **sender** and **args**

Use the *instaceof* function to verify that sender is indeed a *Player* example :
sender instaceof Player

To be able to fully benefit from the Player object you must cast sender in Player

```
Player p = (Player) sender;
```

Now make sure that if there is no additional argument with the command, and teleports the player to the defined location.

Part 5 - Create an order that saves the location so that you can then teleport there

Use your knowledge to try to make a command that saves coordinates in a variable. Then make the command / spawn teleport you there.

Part 6 - Chat Formating

By your own means make sure that the message in the chat is in a format that you define

Part 7 - Block prevention

Prevent players from breaking all blocks

Part 8 - Block prevention

Prevent players from breaking all blocks of dirt