

JAVA / LESSON 31 OF 40

Your First Plugin

Use Gradle and Paper to open the server portal

Environment: JDK 21, ordinary Paper 1.21.11, client of the same version. Server plugins are loaded by Paper; only explicitly labeled rule tests use main. | 150-180 minutes



WelcomePlugin to welcome players

When the player steps into the world, the welcome plug-in really opens its mouth. Follow the path of source, build, JAR, and server loading to understand who created the plugin and who called the events, greetings with first-time and re-join validation.

Prerequisites: classes, inheritance, interfaces and methods. It is recommended to complete it in two sessions: first set up the workbench, and then enter the game acceptance. Today's work is a server plug-in.

Before you start: It is recommended to complete Lessons 17-30 first; prepare for JDK 21

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=31&lang=en>

Long code lines may wrap at the > margin marker. The marker is not part of the code.

In this lesson

01. Before starting: first recognize the running position
02. The code lives in the server this time
03. Label the version first
04. The project directory is the backpack of the plug-in
05. Gradle assembles parts into finished products
06. Give the plug-in an ID card
07. What methods will the server wake up?
08. The listener must register to receive the message
09. Welcome words can also meet old friends
10. Welcome plug-in complete program
11. Pack and inspect JAR
12. In the local world acceptance welcome desk
13. How to check when the welcome desk is silent
14. Expansion challenges and own acceptance records
15. Leave the evidence for the next adventure

01 / 15

Before starting: first recognize the running position

The first 30 lessons lay the foundation for ordinary Java programs. At this stage, the code is connected to the Paper server: the Minecraft client is responsible for entering the world, the server loads the plug-in, and the events and commands hand over player actions to your methods.

This book uses JDK 21 and Paper 1.21.11, and the client version is also consistent. The latest version on the download page may have changed, select it by version number. This book uses normal plugin.yml and the Bukkit API; no claims are made to support Folia at this stage.

The short examples in the picture frame are partial fragments. Read the location description of this section first, and do not save each as a plug-in. The file name will be clearly given for the complete file; when the same file is divided into multiple segments, they will be connected by number, and the package name, import and outer class will only appear once.

- Prediction: Write success conditions and expected messages first.
- Run: After successful construction, install and restart on the dedicated local test server.
- Verification: Ordinary players, management identities and consoles take corresponding paths respectively.

The scene illustrations help to understand the story; the illustrations explain the real rules and are not used as IDE or game actual measurement screenshots. After making Java modifications, rebuild, stop the server, replace it with the new JAR, and start it again; do not treat the performance of the old JAR as the result of the new code.

02 / 15

The code lives in the server this time

A Fang wrote the welcome message and stared at main waiting for the players to come online. The server has already received three people, and his program is still drinking tea nearby. Plug-ins appear differently: the server will call the agreed method at the appropriate moment.

First distinguish the three roles: the client allows players to enter the world, the Paper server manages the world, and the plug-in responds to server notifications. The Java source code needs to be built into a JAR, and plugin.yml tells the server where to find the plug-in entry.

The code lives in the server this time

Step	Explanation
1. Player connection	The client enters the designated server
2. Join event	Paper notification has registered listener
3. Welcome message	The plug-in sends text to this player

Picture reading small task: Use your fingers to draw the route of "Player connection → Join event → Plug-in welcome message". Who calls onJoin? The answer is the server.

Checklist

Object	In charge of what?	Startup mode
Console program	Call main yourself	java class name
Paper	Managing the world and plug-ins	java -jar paper.jar
Plug-in for this lesson	In response to player joining	Loaded by Paper

03 / 15

Label the version first

This book uses the following set of practice environments uniformly. The download page will be updated. Please read the number clearly when selecting the version; the latest button cannot complete the compatibility check for you.

First label the version

Element	Explanation
Java 21	IDE, build and server are consistent
Paper 1.21.11	Compilation dependencies and server correspondence
Client 1.21.11	Enter the test server of the same version

Select 21 for three places: IDE project SDK, JVM used by Gradle, and Java that starts the server. Paper 26.1 and later use Java25; this course will not be automatically upgraded with the download page.

Check Java and compiler · See the instructions in this section for the location

```
java -version
javac -version
```

Output

Both commands should be displayed 21 Series version

Checklist

Tools	Exercise selection	Check position
JDK	21	java -version
Paper and client	1.21.11	Startup log and game launcher
Build tools	Gradle 8.14.2	gradlew --version

04 / 15

The project directory is the backpack of the plug-in

Create a new Java project in IntelliJ IDEA, select Gradle for the build system, Kotlin DSL for the language script, and 21 for the JDK. Name the project BlockAcademyL31. Kotlin DSL is only responsible for building configuration, and the business code in this lesson is still Java.

The project directory is the backpack of the plug-in

Element	Explanation
java	Package directory and source files
resources	plugin.yml and other resources
build/libs	The constructed JAR

The location is very important: plugin.yml is placed in the resources root directory. The package name de.teacherma.lesson31 corresponds to the three-layer folder under java; do not write dots into a folder name.

Directory list · See the description in this section for the location

```
BlockAcademyL31/  
  build.gradle.kts  
  settings.gradle.kts  
  gradlew gradlew.bat gradle/wrapper/  
  src/main/java/de/teacherma/lesson31/WelcomePlugin.java  
  src/main/resources/plugin.yml
```

Settings file content and location are explained in this section

```
rootProject.name = "BlockAcademyL31"
```

05 / 15

Gradle assembles parts into finished products

Gradle will download the API required for compilation, call the compiler, and then type the classes and resources into JAR. Put the following content into build.gradle.kts and reload the Gradle project in the IDE. The first download requires an Internet connection.

Gradle assembles parts into finished products

Step	Explanation
1. Get API	Compile time understanding server type
2. Compile source code	Generate class bytecode
3. Packaging resources	Class and plugin.yml are put into JAR together

Understand two lines: compileOnly means that the Paper API is known at compile time and provided by the server at runtime. Select JDK21 for toolchain. Dependencies for 1.21.11 use SNAPSHOT format; practice retaining verified build products.

Complete build script · Complete file

```
plugins { java }
group = "de.teacherma"
version = "1.0.0"
repositories {
    mavenCentral()
    maven("https://repo.papermc.io/repository/maven-public/")
}
dependencies {
    compileOnly("io.papermc.paper:paper-api:1.21.11-R0.1-SNAPSHOT")
}
java {
    toolchain.languageVersion.set(JavaLanguageVersion.of(21))
}
tasks.withType<JavaCompile>().configureEach {
    options.encoding = "UTF-8"
    options.release.set(21)
}
```

06 / 15

Give the plug-in an ID card

The server will look for plugin.yml after opening the JAR. name is the plug-in display name; main must be equal to the package name of the Java file plus the class name. One less letter and the welcome desk becomes a lost and found office.

Give the plug-in an ID card

Element	Explanation
name	Plug-in display name
main	Package name plus main class name
api-version	Declare target API version

YAML small rules: use spaces for indentation, no tabs; leave a space after the colon. Enclose api-version in quotes. This course uses the normal plugin.yml.

Complete plugin.yml · Complete file

```
name: BlockAcademyL31
version: 1.0.0
main: de.teacherma.lesson31.WelcomePlugin
api-version: '1.21.11'
author: TeacherMa
website: https://www.teacher-ma.de
```

Correspondence between main classes · See the description in this section for the location

```
package de.teacherma.lesson31;
// Main class name WelcomePlugin
// main: de.teacherma.lesson31.WelcomePlugin
```

07 / 15

What methods will the server wake up?

JavaPlugin is a plug-in base class recognized by Paper. The server creates the plug-in object and calls `onEnable` when enabled and `onDisable` when disabled. There is no need to write main yourself here, nor to create a new plug-in main class.

What methods will the server wake up?

Step	Explanation
1. Enable	The server calls <code>onEnable</code>
2. Response	Process player events after registration
3. Disable	<code>onDisable</code> cleans up resources

The audience is different: `getLogger` outputs to the server manager; `player.sendMessage` outputs to the player. The excitement in the log does not mean that the welcome message has been received in the game.

Opening and closing the welcome desk · See the instructions in this section for the location

```
@Override
public void onEnable() {
    getLogger().info("Welcome desk opens");
}
@Override
public void onDisable() {
    getLogger().info("The welcome desk is closed");
}
```

Checklist

Location	Suitable for what to do
onEnable	Register listeners and commands
Event method	Process this player action
onDisable	Cancel tasks and clean up resources

08 / 15

The listener must register to receive the message

Events are like school broadcasts. Implementing Listener indicates that you are ready to receive broadcasts; `@EventHandler` marks the receiving method; `registerEvents` registers the receiver to the server. If you miss one of the three steps, the broadcast may pass you by.

The listener must register to receive the message

Element	Explanation
Listener	Recipient identity
EventHandler	Mark receiving method
registerEvents	Register the recipient to the server

Two this: the first this is the listener object, and the second this is the plug-in that owns the listener. The current class plays two roles at the same time, so it is written the same.

Register the current object · See the description in this section for the location

```
// Main class declaration: extends JavaPlugin implements Listener
// Put in onEnable
getServer().getPluginManager().registerEvents(this, this);
```

Receive join event · See the description in this section for the location

```
@EventHandler
public void onJoin(PlayerJoinEvent event) {
    var player = event.getPlayer();
    player.sendMessage(Component.text("Welcome to Block Academy"));
}
```

09 / 15

Welcome words can also meet old friends

hasPlayedBefore Query whether the server has saved this player's play record. The first time you come to this server and the second time you enter, you can receive different greetings; this is not to judge whether the player is playing Minecraft for the first time.

Welcome words can also meet old friends

Element	Explanation
First time coming to this server	There are no saved play records yet
Come to this server again	hasPlayedBefore is true
Number of people online	The number of people in the server when joining this time

Added import: Component comes from net.kyori.adventure.text; NamedTextColor comes from net.kyori.adventure.text.format. The color examples are optional modifications, the full program is in plain text.

Determine new and old players · See the description in this section for their positions

```
String greeting = player.hasPlayedBefore()
    ? "Welcome back, " : "First meeting, ";
player.sendMessage(Component.text(greeting + player.getName()));
```

Display the number of people currently online. See the description in this section for the location.

```
int count = getServer().getOnlinePlayers().size();
player.sendMessage(Component.text("Currently online: " + count));
```

Color the text · See the instructions in this section for the position

```
player.sendMessage(Component.text("Welcome!")
    .color(NamedTextColor.GREEN));
```

10 / 15

Welcome plug-in complete program

Save this page completely as `src/main/java/de/teacherma/lesson31/WelcomePlugin.java`. Make the original version first, and then change it to your own tone after seeing the two welcome messages.

Welcome plug-in complete program

Element	Explanation
The entrance is correct	The main class inherits JavaPlugin
Register once	onEnable registration listener
Each time you add	Send two messages to the current player

Prediction results: first time joining will receive "first meeting" and the number of people online; exiting and then entering will receive "welcome back". Outage information appears in the server log.

WelcomePlugin source code · See the description in this section for the location

```
package de.teacherma.lesson31;

import net.kyori.adventure.text.Component;
import org.bukkit.event.EventHandler;
import org.bukkit.event.Listener;
import org.bukkit.event.player.PlayerJoinEvent;
import org.bukkit.plugin.java.JavaPlugin;

public final class WelcomePlugin extends JavaPlugin implements Listener {
    @Override
    public void onEnable() {
        getServer().getPluginManager().registerEvents(this, this);
        getLogger().info("The Welcome Desk of Cube Academy is open");
    }

    @EventHandler
    public void onJoin(PlayerJoinEvent event) {
        var player = event.getPlayer();
        String greeting = player.hasPlayedBefore()
            ? "Welcome back, " : "First meeting, ";
        player.sendMessage(Component.text(greeting + player.getName() + "!"));
        int online = getServer().getOnlinePlayers().size();
        player.sendMessage(Component.text("Currently online: " + online + " People"));
    }

    @Override
    public void onDisable() {
        getLogger().info("The welcome desk is off duty, leaving the torch for the next adventure");
    }
}
```

11 / 15

Pack and inspect JAR

Gradle projects created by the IDE should contain Wrapper files. Windows Run the following command in the project root directory; if Wrapper is missing, first use the Gradle tool window of the IDE to execute the wrapper task, or use the installed Gradle to execute the second example.

Pack and inspect the JAR

Step	Explanation
1. Source code and resources	Check project path
2. clean build	Recompile and package
3. Check JAR	Confirm that both the class and plugin.yml are present

Build and run: BUILD SUCCESSFUL indicates successful compilation and packaging. Whether the welcome message appears at the correct time needs to be verified by entering the server. Mac/Linux uses ./gradlew.

Windows build command · See the description in this section for the location

```
.\gradlew.bat clean build
```

Output

Appears at the end when successful BUILD SUCCESSFUL

Alternate method of generating Wrapper · See the description in this section for the location

```
gradle wrapper --gradle-version 8.14.2
```

Check the packaged content. See the description in this section for the location.

```
jar tf build/libs/BlockAcademyL31-1.0.0.jar
```

Output

The list should contain plugin.yml with WelcomePlugin.class

12 / 15

In the local world acceptance welcome desk

Create a dedicated practice server folder, download the 1.21.11 server JAR from Paper official website and name it paper.jar. Teachers or parents should read the download and usage terms; after starting eula.txt for the first time, change false to true only if you agree to the terms.

Stop and think about it: the joining method and annotations have been written, but there is no news when the player comes online. Besides spelling, what step should I check?

In the local world acceptance welcome desk

Step	Explanation
1. Stop service and install	Put the new JAR into plugins
2. Restart	Observe plug-in activation log
3. Client joins	Testing First and Again Greetings

Figure this out: Check onEnable to see if the listener is registered, and then check if the installed JAR is a new build. Writing a receiving method does not mean that you have subscribed to the event.

Start in the server folder · See the instructions in this section for the location

```
java -Xms1G -Xmx2G -jar paper.jar --nogui
```

- When the server stops, copy the JAR of this lesson to the plugins folder; create it if there is no such folder.
- Restart and confirm that the welcome desk opens the door appears in the log. The memory can be adjusted by the teacher according to the computer conditions.
- Use the multiplayer game of the 1.21.11 client to connect to localhost; test both the first entry and the second entry.
- When updating the plug-in, enter stop on the server console, wait for exit, then replace the JAR and start it.

13 / 15

How to check when the welcome desk is silent

Modifying six files in one go is equivalent to having six Endermen remove evidence at the same time. Check from front to back according to the phenomenon, only correct one doubtful point at a time, and then rebuild, restart and join.

How to check when the welcome desk is silent

Element	Explanation
Compilation failure	Dependencies, import and build output
Loading failure	JAR location and entry list
Response failure	Registration, annotation and actual installation version

Exercise Challenge Acceptance: Show the complete project tree, successful build log, and two sets of messages for the first time and the second time to the tablemate. When taking screenshots, cover account information that does not need to be disclosed.

Mark a modification · See the description in this section for the location

```
player.sendMessage(Component.text("Welcome desk v2 Already online"));
```

Checklist

Phenomenon	Check first	Acceptance
import turns red	Gradle synchronization and dependent versions	Build successful
Main class not found	main package name and class name	Enable log occurrence
Plug-in is not in the list	JAR location and plugin.yml	Server identification plug-in
There is a door opening log but no welcome message	Listener registration and annotation	Rejoin received message
Modification has no change	Whether the new JAR has been replaced	Welcome text with test version number

14 / 15

Expansion challenges and own acceptance records

Leave the source code, actual running results and a problem record. Explain it in your own words first, and then run the verification; where the results are different from the predictions, they are often the most worthy of discussion.

Expansion tasks and clearance review

Element	Explanation
Wooden Pickaxe	Rewrite the welcome message
Iron Pickaxe	Explain the number of people online and two outputs
Diamond Pickaxe	Verify first time and rejoin

Next step: Lesson 32 allows players to use /kit to actively receive tool kits.

Official extended reading: docs.papermc.io/paper/dev/project-setup/; docs.papermc.io/paper/dev/plugin-yml/

- Wooden Pickaxe: Change the welcome message to your own block academy style and keep the player name.
- Iron Pick: Increase the number of people online and explain why the log and chat bar have different outputs.
- Diamond Pickaxe: Distinguish between joining for the first time and joining again, and design two sets of acceptance steps.

- Acceptance requirements: record the first time joining, rejoining and service stop logs respectively. Note down the JDK, client, Paper and plug-in versions to facilitate reproducing the problem.

Checklist

I can complete it independently	Mastered	Practice again
Can explain who calls the plug-in method	☐	☐
Can correspond to main and package names	☐	☐
Ability to register and test join events	☐	☐

15 / 15

Leave the evidence for the next adventure

Divide "can compile", "number rules pass" and "actual performance of the game is correct" into three columns. Scenarios that are not running are written to be tested; when bugs are discovered, trigger conditions, expected, actual, minimum repairs, and retest results are recorded.

Your verification record

Category	The operation you selected	Evidence left behind
Normal	A successful path for this lesson	Version and actual output
Border	The moment before the end, the maximum value or the full backpack	Compare item by item with contract
Reject	Wrong parameters, no permissions or irrelevant events	The feedback is clear and the status has not been changed by mistake.

- Explain why each arrow in this lesson points from here to there; the parallel cards have no sequential meaning.
- First change a rule, then modify the description and expectations at the same time, and retest with the same input.

Technical check: This lesson is based on Paper 1.21.11 API and Java SE 21. The reference document may be switched to a newer version, so pay attention to the version number when checking it.

<https://jd.papermc.io/paper/1.21.11/>

<https://docs.papermc.io/paper/dev/>

<https://docs.oracle.com/en/java/javase/21/docs/api/>