

第31课

插件出生

用 Gradle 和 Paper 打开服务器传送门



玩家踏进世界，欢迎插件才真正开口。沿着源码、构建、JAR 与服务器加载的路线，理解谁创建插件、谁调用事件，再用首次和再次加入验证问候。

前置背包：类、继承、接口和方法。建议分两次课完成：先搭好工作台，再进入游戏验收。今日作品是服务器插件。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

环境：JDK 21、普通 Paper 1.21.11、同版本客户端。服务器插件由 Paper 加载；只有明确标注的规则测试使用 main。

开始前：先认清运行位置

前30课打好了普通 Java 程序的基础。这一阶段把代码接到 Paper 服务器：Minecraft 客户端负责进入世界，服务器加载插件，事件和命令把玩家动作交给你的方法。

本书固定使用 JDK 21 与 Paper 1.21.11，客户端版本也保持一致。下载页面上的最新版本可能已经变了，按版本号选择。本书采用普通 plugin.yml 与 Bukkit API；本阶段没有声称支持 Folia。

图框中的短例是局部片段，先读本节的位置说明，不要各自保存成插件。完整文件会明确给出文件名；同一文件分成多段时，按编号连接，包名、import 与外层类只出现一次。

- 预测：先写成功条件和预期消息。
- 运行：构建成功后，在专用本机测试服安装并重启。
- 核对：普通玩家、管理身份与控制台分别走对应路径。

场景插画帮助理解故事；图解解释真实规则，不作为 IDE 或游戏实测截图。做 Java 修改后重新构建、停服替换新 JAR，再启动；不要把旧 JAR 的表现当作新代码结果。

01 代码这次住在服务器里

阿方把欢迎语写好，盯着 main 等玩家上线。服务器已经接待三个人，他的程序还在旁边喝茶。插件的出场方式不同：服务器会在合适的时刻调用约定的方法。

先把三个角色分清：客户端让玩家进入世界，Paper 服务器管理世界，插件响应服务器通知。Java 源码要经构建变成 JAR，plugin.yml 再告诉服务器去哪里找插件入口。



读图小任务：用手指画出“玩家连接 → 加入事件 → 插件欢迎语”的路线。谁调用 onJoin？答案是服务器。

核对表

对象	负责什么	启动方式
控制台程序	自己调用 main	java 类名
Paper	管理世界与插件	java -jar paper.jar
本课插件	响应玩家加入	由 Paper 加载

02 先给版本贴好标签

本书统一使用下表这套练习环境。下载页面会更新，选版本时要看清数字；最新按钮不能替你完成兼容性检查。

先给版本贴好标签

Java 21

IDE、构建与服务器保持一致

Paper 1.21.11

编译依赖和服务端对应

客户端1.21.11

进入同版本测试服

三个地方：IDE项目SDK、Gradle使用的JVM、启动服务器的Java都选21。Paper 26.1及之后改用Java25；本课不随下载页自动升级。

检查 Java 与编译器 • 位置见本节说明

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

运行核对

两条命令都应显示21系列版本

核对表

工具	练习选择	检查位置
JDK	21	java -version
Paper 与客户端	1.21.11	启动日志与游戏启动器
构建工具	Gradle 8.14.2	gradlew --version

03 项目目录是插件的背包

在 IntelliJ IDEA 新建 Java 项目，构建系统选 Gradle，语言脚本选 Kotlin DSL，JDK选21。把项目命名为 BlockAcademyL31。Kotlin DSL 只负责构建配置，本课业务代码仍是 Java。

项目目录是插件的背包

java

包目录与源文件

resources

plugin.yml等资源

build/libs

构建后的JAR

位置很重要：plugin.yml 放在 resources 根目录。包名 de.teacherma.lesson31 对应 java 下的三层文件夹；不要把点号写进一个文件夹名。

目录清单 • 位置见本节说明

```
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
```

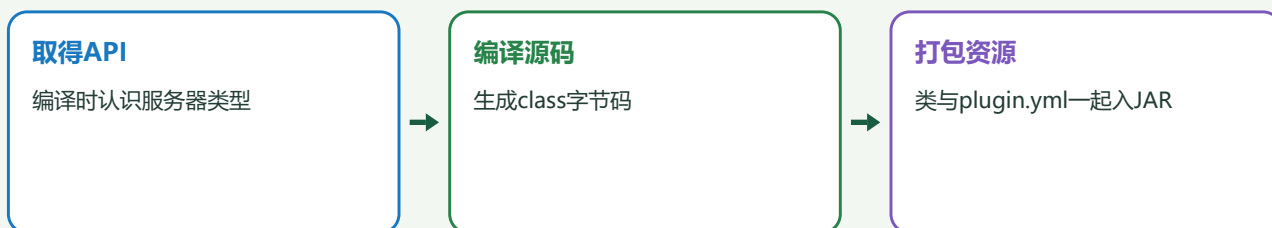
settings 文件内容 • 位置见本节说明

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

04 Gradle 把零件装成成品

Gradle 会下载编译所需的 API、调用编译器，再把类和资源打成 JAR。把下面的内容放进 `build.gradle.kts`，然后在 IDE 中重新加载 Gradle 项目。首次下载需要联网。

Gradle 把零件装成成品



读懂两行：`compileOnly` 表示编译时认识 Paper API，运行时由服务器提供它。`toolchain` 选择JDK21。
1.21.11的依赖使用SNAPSHOT格式；练习保留已验证的构建产物。

完整构建脚本 • 完整文件

```
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)
}
```

05 给插件一张身份证

服务器打开 JAR 后会寻找 plugin.yml。name 是插件显示名；main 必须等于 Java 文件的包名加类名。少一个字母，欢迎台就会变成失物招领处。

给插件一张身份证

name

插件显示名

main

包名加主类名

api-version

声明目标API版本

YAML 小规矩：用空格缩进，不用Tab；冒号后留空格。api-version 用引号包住。本课程使用普通 plugin.yml。

完整 plugin.yml • 完整文件

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

主类对应关系 • 位置见本节说明

```
package de.teacherma.lesson31;
// 主类名称 WelcomePlugin
// main: de.teacherma.lesson31.WelcomePlugin
```

06 服务器会叫醒哪些方法

JavaPlugin 是 Paper 能识别的插件基类。服务器创建插件对象，启用时调用 onEnable，停用时调用 onDisable。这里不用自己编写 main，也不要 new 一个插件主类。



观众不同：getLogger 输出给服务器管理者；player.sendMessage 输出给玩家。日志里热闹，不代表游戏里已经收到欢迎语。

打开和关闭欢迎台 · 位置见本节说明

```
@Override
public void onEnable() {
    getLogger().info("欢迎台开门");
}
@Override
public void onDisable() {
    getLogger().info("欢迎台关门");
}
```

核对表

位置	适合做什么
onEnable	注册监听器和命令
事件方法	处理本次玩家动作
onDisable	取消任务和清理资源

07 监听器要登记才会收到消息

事件像学校广播。实现 Listener 表示你准备接收广播；@EventHandler 标记接收方法；registerEvents 把接收者登记到服务器。三步缺一不可，广播就可能与你擦肩而过。

监听器要登记才会收到消息

Listener

接收者身份

EventHandler

标记接收方法

registerEvents

把接收者登记到服务器

两个 this：第一个 this 是监听器对象，第二个 this 是拥有这个监听器的插件。当前类同时扮演两种角色，所以写成一样。

登记当前对象 • 位置见本节说明

```
// 主类声明: extends JavaPlugin implements Listener
// 放进 onEnable
getServer().getPluginManager().registerEvents(this, this);
```

接收加入事件 • 位置见本节说明

```
@EventHandler
public void onJoin(PlayerJoinEvent event) {
    var player = event.getPlayer();
    player.sendMessage(Component.text("欢迎来到方块学院"));
}
```

08 欢迎语也能认识老朋友

hasPlayedBefore 查询服务器是否已经保存过这位玩家的游玩记录。第一次来到这个服务器和再次进入，可以收到不同问候；这不是在判断玩家是否第一次玩 Minecraft。

欢迎语也能认识老朋友

首次来到本服

尚无已保存的游玩记录

再次来到本服

hasPlayedBefore为true

在线人数

本次加入时的服务器人数

补充 import: Component 来自 net.kyori.adventure.text; NamedTextColor 来自 net.kyori.adventure.text.format。颜色例子是可选改造，完整程序用普通文字。

判断新老玩家 • 位置见本节说明

```
String greeting = player.hasPlayedBefore()
    ? "欢迎回来, " : "初次见面, ";
player.sendMessage(Component.text(greeting + player.getName()));
```

显示当前在线人数 • 位置见本节说明

```
int count = getServer().getOnlinePlayers().size();
player.sendMessage(Component.text("当前在线: " + count));
```

给文字染色 • 位置见本节说明

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

09 欢迎插件完整程序

把这一页完整保存为 src/main/java/de/teacherma/lesson31/WelcomePlugin.java。先做原版，看到两条欢迎消息后，再改成自己的口吻。

欢迎插件完整程序

入口正确

主类继承JavaPlugin

登记一次

onEnable注册监听器

每次加入

向当前玩家发送两条消息

预测结果：首次加入收到“初次见面”和在线人数；退出再进收到“欢迎回来”。停服信息出现在服务器日志里。

WelcomePlugin 源码 · 位置见本节说明

```
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("方块学院欢迎台已开门");
    }

    @EventHandler
    public void onJoin(PlayerJoinEvent event) {
        var player = event.getPlayer();
        String greeting = player.hasPlayedBefore()
            ? "欢迎回来, " : "初次见面, ";
        player.sendMessage(Component.text(greeting + player.getName() + "! "));
        int online = getServer().getOnlinePlayers().size();
        player.sendMessage(Component.text("当前在线: " + online + " 人"));
    }

    @Override
    public void onDisable() {
        getLogger().info("欢迎台下班了, 火把留给下一次冒险");
    }
}
```

10 打包并检查 JAR

IDE 创建的 Gradle 项目应包含 Wrapper 文件。Windows 在项目根目录运行下面命令；如果缺少 Wrapper，先用 IDE 的 Gradle 工具窗口执行 wrapper 任务，或用已安装的 Gradle 执行第二个例子。

打包并检查 JAR



构建和运行：BUILD SUCCESSFUL 表示编译和打包成功。欢迎语是否在正确时机出现，还需要进入服务器验证。Mac/Linux 使用 ./gradlew。

Windows 构建命令 • 位置见本节说明

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

运行核对

成功时末尾出现 BUILD SUCCESSFUL

生成 Wrapper 的备用方法 • 位置见本节说明

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

检查打包内容 • 位置见本节说明

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

运行核对

清单应包含 plugin.yml 与 WelcomePlugin.class

11 在本机世界验收欢迎台

建立专用练习服务器文件夹，从 Paper 官方下载1.21.11服务器JAR并命名为 paper.jar。由老师或家长阅读下载与使用条款；首次启动生成 eula.txt 后，只有同意条款才把其中的 false 改为 true。

停下来想一想：加入方法和注解都写好了，玩家上线却没有消息。除拼写外，还要检查哪一步？

在本机世界验收欢迎台

停服安装

把新JAR放进plugins



重新启动

观察插件启用日志



客户端加入

测试首次与再次问候

想通这一步：检查 onEnable 是否登记监听器，再检查安装的 JAR 是否是新构建版本。写了接收方法不等于已经订阅事件。

在服务器文件夹启动 · 位置见本节说明

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

- 服务器停止时，把本课JAR复制到 plugins 文件夹；没有该文件夹就创建它。
- 重新启动，确认日志中出现欢迎台开门。内存可由老师按电脑情况调整。
- 使用1.21.11客户端的多人游戏连接 localhost；第一次进入和第二次进入都测试。
- 更新插件时在服务器控制台输入 stop，等待退出，再替换JAR并启动。

12 欢迎台不说话时怎么查

一口气修改六个文件，相当于让六只末影人同时搬走证据。按现象从前往后查，每次只改一个疑点，然后重新构建、重启和加入。

欢迎台不说话时怎么查

编译故障

依赖、import与构建输出

加载故障

JAR位置和入口清单

响应故障

登记、注解与实际安装版本

练习 挑战 验收：向同桌展示：完整项目树、成功构建日志、首次与再次加入的两组消息。截图时遮住不需要公开的账号信息。

给一次修改做标记 · 位置见本节说明

```
player.sendMessage(Component.text("欢迎台 v2 已上线"));
```

核对表

现象	先检查	验收
import 变红	Gradle同步和依赖版本	build成功
找不到主类	main包名与类名	启用日志出现
插件不在列表	JAR位置与plugin.yml	服务器识别插件
有开门日志没欢迎语	监听器登记与注解	重新加入收到消息
修改没有变化	是否替换了新JAR	欢迎语带测试版本号

13 拓展挑战与自己的验收记录

留下源代码、实际运行结果和一条问题记录。先用自己的话解释，再运行验证；结果与预测不同的地方，往往最值得讨论。

拓展任务与通关复盘

木镐

改写欢迎语

铁镐

解释在线人数与两种输出

钻石镐

验证首次及再次加入

下一站：第32课让玩家用 /kit 主动领取工具包。
官方延伸阅读：docs.papermc.io/paper/dev/project-setup/；docs.papermc.io/paper/dev/plugin-yml/

- 木镐：把欢迎语改成自己的方块学院风格，保留玩家名字。
- 铁镐：增加在线人数，并解释为什么日志与聊天栏是不同输出。
- 钻石镐：区分首次加入与再次加入，设计两组验收步骤。
- 验收要求：分别记录首次加入、再次加入和停服日志。记下JDK、客户端、Paper和插件版本，便于复现问题。

核对表

我能独立完成	已掌握	再练一次
能解释谁调用插件方法	☐	☐
能对应main与包名	☐	☐
能注册并测试加入事件	☐	☐

把证据留给下一次冒险

把“能编译”“数字规则通过”和“游戏实际表现正确”分成三栏。没有运行的场景写待测；发现 Bug 时记录触发条件、预期、实际、最小修复与复测结果。

你的验证记录

类别	你选择的操作	留下的证据
正常	本课的一次成功路径	版本与实际输出
边界	结束前一刻、最大值或满背包	与合同逐项比较
拒绝	错误参数、无权限或不相关事件	反馈清楚，状态未误改

- 解释本课每条箭头为什么从这里指向那里；并列卡片没有先后含义。
- 先改一个规则，再同时修改说明和预期，用同一输入复测。

技术核对：本课以 Paper 1.21.11 API 和 Java SE 21 为准。参考文档可能切换到较新版本，查阅时注意版本号。

<https://jd.papermc.io/paper/1.21.11/>
<https://docs.papermc.io/paper/dev/>
<https://docs.oracle.com/en/java/javase/21/docs/api/>