

第36课

让服主改配置

让配置经过检查再成为新规则



配置不是把任意文字塞进数字格。先检查类型、范围和世界，把合格的候选配置一次换成新规则；读取失败时，上一份有效配置继续工作。

前置背包：第31课的Gradle项目、第32课的命令和第28课的异常处理。继续使用Java21、普通Paper1.21.11。新项目名BlockAcademyL36。

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 面包多一块不用重造烤箱

服主只想把奖励面包从3块改成4块，阿方已经准备重编整个插件。配置文件像烤箱的旋钮：规则允许调整时，转旋钮就能换设置，先别拆房子。

配置模板提供默认值，磁盘文件保存实际设置，内存保存当前生效的规则。重载时先完整读取并校验候选配置，全部合格才替换内存中的旧规则。

面包多一块不用重造烤箱

模板默认值

首次安装的起点

磁盘配置

服主编辑的文件

内存规则

当前真正生效的设置

这一课的作品：完成/`rules show`、/`rules snack`、/`rules reload`。`snack`是管理员授权的可重复测试命令，不是每人只能领取一次的新手礼包。

放入src/main/resources/config.yml • 位置见本节说明

```
welcome-message: "欢迎来到方块学院！"
starter-bread: 3
ore-rush:
  duration-seconds: 60
  world: "world"
```

- `starter-bread`控制测试奖励数量；0表示关闭。
- `ore-rush.world`指定允许测试面包的世界。
- `duration-seconds`先供查看，后面用同样思路控制比赛时长。

02 模板文件和正在使用的文件

JAR里的config.yml像空白配方本；plugins文件夹里的那份才是服主正在填写的本子。saveDefaultConfig只在目标文件不存在时复制默认文件，不会每天开机把老师的改动擦掉。

模板文件和正在使用的文件

resources/config.yml

随JAR发布的默认模板

插件数据目录

当前服务器配置

saveConfig

写回，不是重新读取

别按错保存键：saveConfig把内存配置写回磁盘，不是“重新读取”。本课手动编辑文件后读取候选配置，不在reload里调用saveConfig。

启用时准备默认文件 • 位置见本节说明

```
saveDefaultConfig();
```

两个不同的位置 • 位置见本节说明

```
src/main/resources/config.yml
plugins/BlockAcademyL36/config.yml
```

核对表

文件	职责	改后怎样生效
plugin.yml	插件身份与命令	重新构建并重启
JAR内config.yml	首次安装的默认值	重新构建影响新安装
数据目录config.yml	当前服务器的规则	本课用/rules reload

03 先检查类型 再检查大小

数字3、文字"3"和小数3.5长得像亲戚，却不能全住进整数格子。只把负数夹到0，仍然没有解释“好多块面包”这种输入。我们先检查类型，再检查约定的范围。

先检查类型 再检查大小

类型

整数不等于文字数字

范围

面包0到64

世界

名称对应已加载世界

冒号后留空格：YAML层级用空格缩进，不用Tab。上面三个starter-bread是三个独立试验，不要同时复制为重复键。

看起来相似的三个值 • 位置见本节说明

```
starter-bread: 3      # 整数 合格
starter-bread: "3"    # 文字 拒绝
starter-bread: 3.5    # 小数 拒绝
```

读取嵌套键 • 位置见本节说明

```
Object value = config.get("ore-rush.duration-seconds");
```

核对表

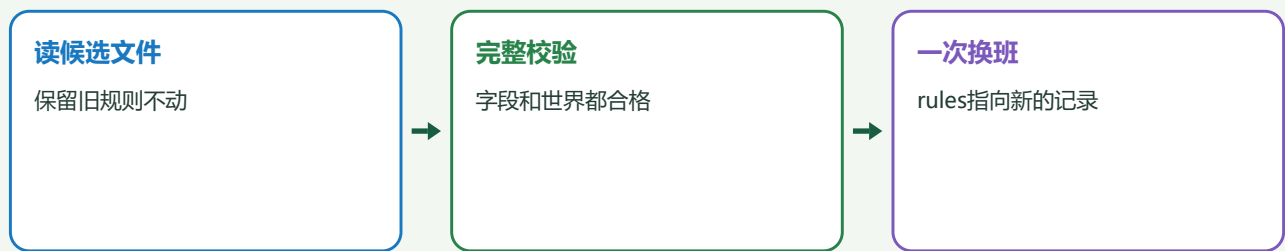
配置项	本课约定
welcome-message	非空文字，最多80个Java字符
starter-bread	整数0到64
ore-rush.duration-seconds	整数1到3600，单位秒
ore-rush.world	非空名称，并且世界已加载

04 新规则全部合格才换班

如果面包先改成64，随后才发现世界名拼错，服务器就处在半新半旧的规则里。我们先读一份候选配置，检查全部字段和世界，最后只用一次赋值替换正在使用的规则。

停下来想一想：奖励数量合法，但活动秒数写成文字。能先应用奖励，再报告秒数错误吗？

新规则全部合格才换班



沿箭头进入下一步前，必须满足前一步条件；被拒绝的分支提前返回。

想通这一步：应整份拒绝，继续使用旧规则，避免出现半套新规则半套旧规则。日志指出具体配置键和期望类型。

检查世界后再应用 · 位置见本节说明

```

RuleSettings next = RuleSettings.read(candidate);
if (getServer().getWorld(next.world()) == null)
    throw new IllegalArgumentException("世界未加载");
rules = next;
  
```

故障实验：下面故意保留错误，仅在练习副本中观察。不要把它合并进可用插件。

不要先改一半 · 故意错误，先预测再修复

```

// 错误思路：先更新奖励，再检查世界
// 如果第二步失败，第一步已经生效。
  
```

05 把检查集中在规则类里

新建同包的RuleSettings.java。record是Java的数据记录类型：括号列出数据，Java自动提供welcome()、bread()等读取方法。下面的方法返回一份已通过字段检查的记录；世界是否存在由插件检查。

把检查集中在规则类里

只读记录

String与int保存规则

text校验

非空文字与长度

integer校验

整数类型与范围

先认识再使用：本课只把record当只读数据盒。字段是String或int，创建后不再改字段值；应用新配置时换成新的盒子。

完整RuleSettings.java • 完整文件

```
package de.teacherma.lesson36;

import org.bukkit.configuration.file.FileConfiguration;

public record RuleSettings(String welcome, int bread,
    int seconds, String world) {
    public static RuleSettings read(FileConfiguration c) {
        String welcome = text(c, "welcome-message");
        int bread = integer(c, "starter-bread", 0, 64);
        int seconds = integer(c, "ore-rush.duration-seconds", 1, 3600);
        String world = text(c, "ore-rush.world");
        return new RuleSettings(welcome, bread, seconds, world);
    }
    private static String text(FileConfiguration c, String key) {
        Object value = c.get(key);
        if (!(value instanceof String s) || s.isBlank() || s.length() > 80)
            throw new IllegalArgumentException(key + " 需要1到80字的非空文字");
        return s;
    }
    private static int integer(FileConfiguration c, String key, int min, int max) {
        Object value = c.get(key);
        if (!(value instanceof Integer n) || n < min || n > max)
            throw new IllegalArgumentException(key + " 需要整数 " + min + ".." + max);
        return n;
    }
}
```

06 配置台的项目清单

项目沿用第31课的build.gradle.kts和Java21设置，settings.gradle.kts的rootProject.name改成BlockAcademyL36。两个Java文件放在lesson36包；两个yaml文件放在resources目录。

配置台的项目清单

两个Java文件

规则类和主类

plugin.yml

身份、命令与权限

config.yml

首次安装的默认设置

先确定管理权限：权限节点默认op。普通玩家不能自行领取无限面包，也不能改动配置；测试时请老师在本地练习服操作。

完整plugin.yml • 完整文件

```
name: BlockAcademyL36
version: 1.0.0
main: de.teacherma.lesson36.RuleDeskPlugin
api-version: '1.21.11'
author: TeacherMa
website: https://www.teacher-ma.de
commands:
  rules:
    description: 查看和测试课堂配置
    usage: /rules <show|reload|snack>
permissions:
  blockacademy.rules.admin:
    default: op
```

Java文件目录 • 位置见本节说明

```
src/main/java/de/teacherma/lesson36/
  RuleSettings.java
  RuleDeskPlugin.java
```

07 配置台主类上半部

新建RuleDeskPlugin.java，先放本页，再紧接下一页。candidate.load会把读取或YAML格式问题交给catch处理；不要把失败后的空配置误当成有效新规则。

配置台主类上半部

准备配置

不存在才复制

加载候选

捕获读写或解析失败

初始化命令

启动配置合格才继续

主类第1部分 · 本段1/2，顺序连接

```
package de.teacherma.lesson36;

import java.io.File;
import java.io.IOException;
import java.util.Objects;
import java.util.logging.Level;
import org.bukkit.Material;
import org.bukkit.command.Command;
import org.bukkit.command.CommandSender;
import org.bukkit.configuration.InvalidConfigurationException;
import org.bukkit.configuration.file.YamlConfiguration;
import org.bukkit.entity.Player;
import org.bukkit.inventory.ItemStack;
import org.bukkit.plugin.java.JavaPlugin;

public final class RuleDeskPlugin extends JavaPlugin {
    private RuleSettings rules;
    @Override
    public void onEnable() {
        saveDefaultConfig();
        if (!loadRules()) {
            getServer().getPluginManager().disablePlugin(this);
            return;
        }
        Objects.requireNonNull(getCommand("rules")).setExecutor(this);
    }
}
```

07 配置台主类上半部（续）

主类第1部分 • 本段2/2, 顺序连接

```
}
private boolean loadRules() {
    try {
        var candidate = new YamlConfiguration();
        candidate.load(new File(getDataFolder(), "config.yml"));
        RuleSettings next = RuleSettings.read(candidate);
        if (getServer().getWorld(next.world()) == null)
            throw new IllegalArgumentException("世界未加载: " + next.world());
        rules = next;
        return true;
    } catch (IOException | InvalidConfigurationException
            | IllegalArgumentException ex) {
        getLogger().log(Level.WARNING, "配置未应用, 请检查config.yml", ex);
        return false;
    }
}
```

08 配置台主类下半部

继续上一页，保留所有大括号。show展示当前规则；reload尝试换规则；snack测试实际奖励。未知参数只给说明，不能顺手发一车面包。

配置台主类下半部

show

看当前生效规则

reload

合格才换新规则

snack

按配置演示奖励

主类第2部分 · 本段1/2，顺序连接

```

    }
    @Override
    public boolean onCommand(CommandSender sender, Command command,
                            String label, String[] args) {
        if (!sender.hasPermission("blockacademy.rules.admin")) {
            sender.sendMessage("这项课堂测试需要老师授权");
            return true;
        }
        if (args.length != 1) {
            sender.sendMessage("用法: /rules show 或 reload 或 snack");
            return true;
        }
        switch (args[0]) {
            case "reload" -> sender.sendMessage(loadRules()
                ? "新配置已生效" : "配置失败，继续使用上次有效规则");
            case "show" -> {
                sender.sendMessage(rules.welcome());
                sender.sendMessage("面包=" + rules.bread() + ", 时长=" + rules.seconds()
                    + "秒, 世界=" + rules.world());
            }
            case "snack" -> {
                if (!(sender instanceof Player p)) {
                    sender.sendMessage("请在游戏中测试面包");
                    return true;
                }
            }
        }
    }

```

08 配置台主类下半部（续）

主类第2部分 • 本段2/2, 顺序连接

```
        if (!p.getWorld().getName().equals(rules.world())) {
            p.sendMessage("请先到配置指定的世界");
            return true;
        }
        if (rules.bread() == 0) { p.sendMessage("面包奖励已关闭"); return true; }
        var left = p.getInventory().addItem(new ItemStack(Material.BREAD, rules.bread()));
        left.values().forEach(item -> p.getWorld().dropItemNaturally(p.getLocation(), item));
        p.sendMessage(left.isEmpty() ? "面包已放入背包" : "剩余面包在脚边");
    }
    default -> sender.sendMessage("参数只能是 show、reload、snack");
}
return true;
}
```

09 零奖励和满背包也算正常情况

0块面包表示老师关闭了奖励，不需要创建一个数量为0的物品堆。背包满了也不是程序世界末日：addItem会返回装不下的部分。练习示例把剩余物品放在玩家脚边，并如实提示。

零奖励和满背包也算正常情况

数量0

奖励关闭，不造0数量物品

背包可容纳

正常入包

背包装不下

只处理剩余部分

测试场地：在平坦空地测试满背包。地上的奖励可能被别人捡走或消失；正式玩法应另定领取、冷却和溢出规则。snack只演示读取配置。

先处理关闭状态 • 位置见本节说明

```
if (rules.bread() == 0) {
    p.sendMessage("面包奖励已关闭");
    return true;
}
```

只把剩余部分放到地上 • 位置见本节说明

```
left.values().forEach(item ->
    p.getWorld().dropItemNaturally(p.getLocation(), item));
```

10 亲手把三改成四

先构建并安装，进入游戏运行/rules show，确认面包为3。只修改数据目录的starter-bread为4，保存文件，再执行/rules reload。最后用snack验证实际多了一块，而不是只相信屏幕上的成功提示。

亲手把三改成四



记录证据：保存“修改前、执行reload后、实际领取”三组结果。版本旧JAR、编辑错文件，也能造成“我明明改了却没变”。

构建和练习命令 • 位置见本节说明

```
.\gradlew.bat clean build
/rules show
/rules reload
/rules snack
```

核对表

修改	预期
3改为4	新配置生效，奖励4块
设为0	关闭提示，不发物品
设为65或负数	重读失败，旧规则不变
改成"4"	拒绝文字型数字
世界名写错	拒绝，旧世界规则仍有效

11 让错误指出要修哪一行

报错说“配置失败”还不够。规则检查把键名写进异常信息；主类记录异常及堆栈，聊天栏只给简短反馈。先找到具体配置键，再由有管理权限的人决定何时重读。

让错误指出要修哪一行

格式错误

检查缩进冒号引号

规则错误

异常指出具体键

加载错误

保留日志与旧规则

保存旧文件：编辑前复制一份小配置做备份。不要为了消除红字而直接吞掉异常；清空日志不等于问题消失。

保留异常线索 · 位置见本节说明

```
getLogger().log(Level.WARNING, "配置未应用，请检查config.yml", ex);
```

核对表

现象	先检查
YAML解析错误	缩进、冒号、引号
需要整数0..64	字段是否为整数且在范围内
世界未加载	世界名称与服务器实际世界
初次启动后命令不可用	启用日志是否显示配置失败
reload后数值仍旧	是否出现失败提示，是否改错文件

12 把秒数交给下一场活动

第35课把60写死在代码里。现在开始活动时，把已经校验过的秒数复制到本场变量。本场开始后再改配置，应该影响下一场，还是立刻改本场？先把规则写清楚，再写代码。

把秒数交给下一场活动

本场快照

开场复制已验证秒数

途中改配置

本场仍按原合同

下一场

采用后来生效的新值

规模边界：练习小文件的同步重读便于理解，但I/O会占用主线程。大型配置应异步解析独立数据，再回主线程验证世界并替换，不能在异步线程直接访问世界。

开场取一次快照 · 位置见本节说明

```
int thisRoundSeconds = rules.seconds();  
// 用这个值初始化本场倒计时  
// 不要在每个tick里重新读取文件。
```

两场比赛的预期 · 位置见本节说明

第一场开始时：60秒
运行中修改为：90秒
第一场保持60秒，下一场使用90秒

- 本课rules插件只展示时长，不会自动开启比赛。
- 第38课会制作真正使用时长配置的比赛插件。
- 读取小配置只发生在启动和管理员明确重读时；不要每次挖矿都读磁盘。

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

改欢迎语

铁镐

证明坏配置不覆盖旧规则

钻石镐

设计新字段及边界

下一站：第37课给矿工罗盘加上不靠显示名判断的身份标签。
官方延伸阅读：docs.papermc.io/paper/dev/plugin-configurations/

- 木镐：为show配置自己的欢迎语，解释模板与生效文件的区别。
- 铁镐：完成0、64、65、负数、文字数字和错误世界名测试，失败后旧规则保持不变。
- 钻石镐：设计三档面包奖励配置，先写字段与范围说明，再增加读取检查。
- 验收要求：测试缺失键、错误类型、负数和边界值；坏配置重载后，证明上一份有效设置仍然生效。

核对表

我能独立完成	已掌握	再练一次
能找到正在使用的config.yml	☐	☐
会校验类型范围和世界	☐	☐
能解释候选规则何时替换	☐	☐



马老师教编程
www.teacher-ma.de

17 / 18

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/>