# Minecraft 1.20.4 Fabric模组开发技术研究报告 (修订版)

**作者**: MiniMax Agent

**日期**: 2025-06-22

## 1. 简介

本报告旨在为开发一个名为“里程表模组”(Milometer Mod)的Minecraft 1.20.4 Fabric模组提供全面的技术指导。报告详细介绍了开发环境的配置、项目结构、核心功能的实现方法以及相关的最佳实践。本修订版报告新增了游戏内配置界面和通过指令调整UI位置的功能。

## 2. 技术规范

* **Minecraft 版本**: 1.20.4
* **Fabric Loader 版本**: 0.15.11 或更高
* **Fabric API 版本**: 0.97.3+1.20.4 或更高
* **Java 开发工具包 (JDK)**: 17 或更高
* **构建工具**: Gradle (通过 Fabric Loom 插件)
* **配置库**: Cloth Config API

## 3. 项目结构

一个标准的 Fabric 模组项目具有以下结构:

.  
├── build.gradle  
├── gradle.properties  
├── gradlew  
├── gradlew.bat  
└── src  
 └── main  
 ├── java  
 │ └── com  
 │ └── tonyv2  
 │ └── milometermod  
 │ ├── MilometerMod.java  
 │ ├── MilometerModClient.java  
 │ ├── client  
 │ │ └── MilometerHud.java  
 │ ├── command  
 │ │ └── ResetDistanceCommand.java  
 │ ├── compat  
 │ │ └── ModMenuIntegration.java  
 │ ├── config  
 │ │ └── MilometerConfig.java  
 │ └── event  
 │ └── PlayerMovementTracker.java  
 └── resources  
 └── fabric.mod.json

### 3.1. build.gradle (无变化)

### 3.2. gradle.properties (无变化)

### 3.3. fabric.mod.json (精简后)

{  
 "schemaVersion": 1,  
 "id": "milometermod",  
 "version": "${version}",  
  
 "name": "Milometer Mod",  
 "description": "Displays player speed and total distance traveled.",  
 "authors": [  
 "MiniMax Agent"  
 ],  
 "license": "CC0-1.0",  
 "icon": "assets/milometermod/icon.png",  
  
 "environment": "client",  
 "entrypoints": {  
 "client": [  
 "com.tonyv2.milometermod.MilometerModClient"  
 ],  
 "main": [  
 "com.tonyv2.milometermod.MilometerMod"  
 ],  
 "modmenu": [  
 "com.tonyv2.milometermod.compat.ModMenuIntegration"  
 ]  
 },  
  
 "depends": {  
 "fabricloader": ">=0.15.11",  
 "fabric-api": "\*",  
 "minecraft": "~1.20.4",  
 "java": ">=17"  
 }  
}

## 4. 核心功能实现 (更新部分)

### 4.5. 游戏内配置界面

通过集成 Cloth Config 和 Mod Menu，我们可以为模组提供一个易于使用的游戏内配置界面。

**ModMenuIntegration.java**

package com.tonyv2.milometermod.compat;  
  
import com.terraformersmc.modmenu.api.ConfigScreenFactory;  
import com.terraformersmc.modmenu.api.ModMenuApi;  
import com.tonyv2.milometermod.config.MilometerConfig;  
import me.shedaniel.clothconfig2.api.ConfigBuilder;  
import me.shedaniel.clothconfig2.api.ConfigCategory;  
import me.shedaniel.clothconfig2.api.ConfigEntryBuilder;  
import net.minecraft.client.gui.screen.Screen;  
import net.minecraft.text.Text;  
  
public class ModMenuIntegration implements ModMenuApi {  
  
 @Override  
 public ConfigScreenFactory<?> getModConfigScreenFactory() {  
 return parent -> {  
 ConfigBuilder builder = ConfigBuilder.create()  
 .setParentScreen(parent)  
 .setTitle(Text.literal("Milometer Mod Config"));  
  
 ConfigCategory general = builder.getOrCreateCategory(Text.literal("General"));  
  
 ConfigEntryBuilder entryBuilder = builder.entryBuilder();  
  
 general.addEntry(entryBuilder.startIntField(Text.literal("HUD X"), MilometerConfig.INSTANCE.hudX)  
 .setDefaultValue(10)  
 .setSaveConsumer(newValue -> MilometerConfig.INSTANCE.hudX = newValue)  
 .build());  
  
 general.addEntry(entryBuilder.startIntField(Text.literal("HUD Y"), MilometerConfig.INSTANCE.hudY)  
 .setDefaultValue(10)  
 .setSaveConsumer(newValue -> MilometerConfig.INSTANCE.hudY = newValue)  
 .build());  
  
 builder.setSavingRunnable(() -> {  
 MilometerConfig.INSTANCE.save();  
 });  
  
 return builder.build();  
 };  
 }  
}

### 4.6. 配置指令

为了提供更快捷的配置方式，我们添加了一个指令来设置 HUD 的位置。

**ResetDistanceCommand.java (更新后)**

package com.tonyv2.milometermod.command;  
  
import com.mojang.brigadier.CommandDispatcher;  
import com.mojang.brigadier.arguments.IntegerArgumentType;  
import com.tonyv2.milometermod.config.MilometerConfig;  
import com.tonyv2.milometermod.MilometerModClient;  
import net.fabricmc.fabric.api.client.command.v2.ClientCommandManager;  
import net.fabricmc.fabric.api.client.command.v2.FabricClientCommandSource;  
import net.minecraft.command.CommandRegistryAccess;  
import net.minecraft.text.Text;  
  
public class ResetDistanceCommand {  
 public static void register(CommandDispatcher<FabricClientCommandSource> dispatcher, CommandRegistryAccess registryAccess) {  
 dispatcher.register(ClientCommandManager.literal("milometer")  
 .then(ClientCommandManager.literal("reset")  
 .executes(context -> {  
 MilometerModClient.movementTracker.resetDistance();  
 context.getSource().sendFeedback(Text.literal("Milometer distance reset."));  
 return 1;  
 }))  
 .then(ClientCommandManager.literal("setpos")  
 .then(ClientCommandManager.argument("x", IntegerArgumentType.integer())  
 .then(ClientCommandManager.argument("y", IntegerArgumentType.integer())  
 .executes(context -> {  
 int x = IntegerArgumentType.getInteger(context, "x");  
 int y = IntegerArgumentType.getInteger(context, "y");  
 MilometerConfig.INSTANCE.hudX = x;  
 MilometerConfig.INSTANCE.hudY = y;  
 MilometerConfig.INSTANCE.save();  
 context.getSource().sendFeedback(Text.literal("HUD position set to (" + x + ", " + y + ")"));  
 return 1;  
 })))));  
 }  
}

## 5. 总结

本报告提供了一个完整的 Fabric 1.20.4 模组开发框架，涵盖了从项目设置到核心功能实现的各个方面。开发者可以以此为基础，进一步扩展和定制模组功能。