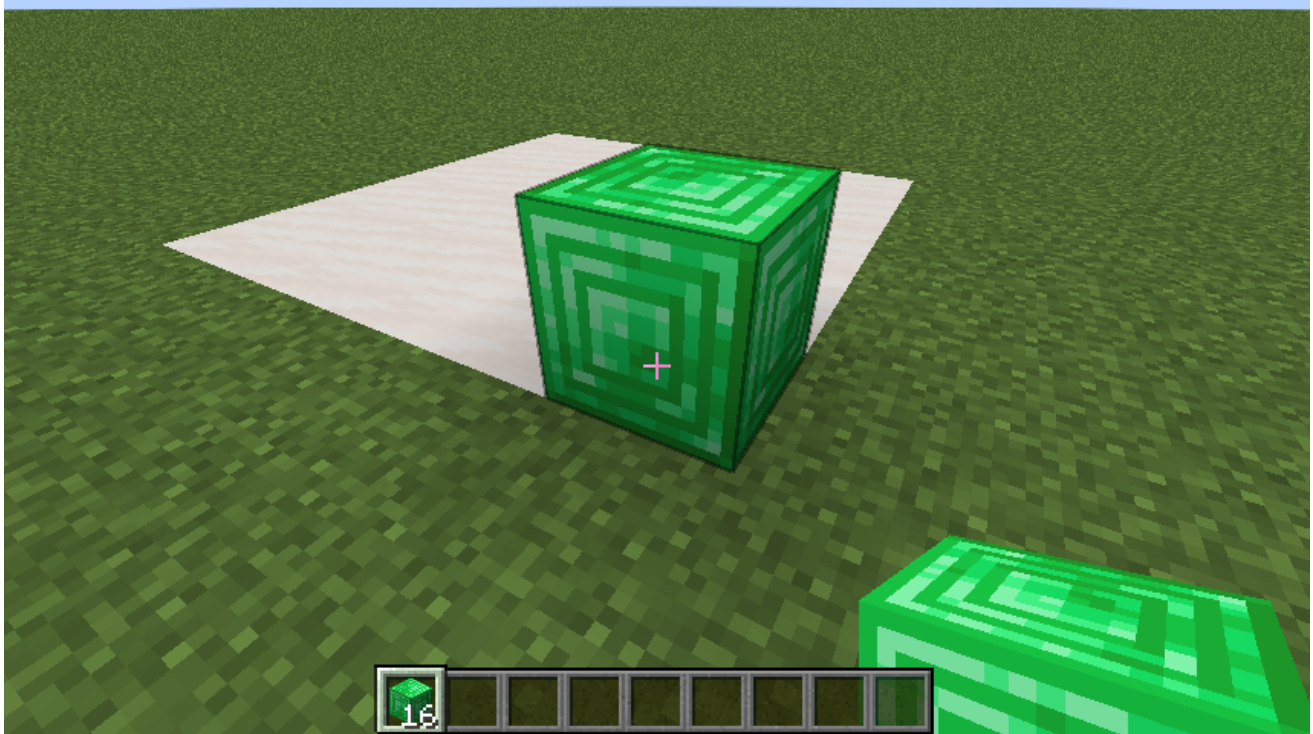


FABRIC 1.20 / JDK 21

第 2 课

# 注册第一个自定义方块

从模板出发，做一个独立的入门方块



原教材中的真实 Fabric 客户端截图。

## 本课作品

做出有独立 ID、正常图标和显示名称、可以放置保存的入门方块。

100–120 分钟 | [simon-minecraft-template-1.20](#)

保留原模板，从一个普通立方体开始，完整走过注册与资源连接。

前置知识：先完成第 1 课，确认模板可以进入游戏。

[teacher-ma.de/minecraft-mod-lesson.php?lesson=2&lang=zh](https://teacher-ma.de/minecraft-mod-lesson.php?lesson=2&lang=zh)

01 / 11

# 一件作品，两种身份

先分清世界中的方块与背包里的物品。

本课新建 `simon-minecraft:starter_block`，先借用原版绿宝石方块贴图。它是独立注册的普通方块；后续课程再加入吊灯模型、真实属性与掉落。

|           |                                                                |
|-----------|----------------------------------------------------------------|
| Block     | 描述方块类型和行为；世界中具体位置保存的是 BlockState。                              |
| BlockItem | 是背包里的物品，使用时把对应方块放进世界。注册 Block 不会自动注册 BlockItem。                |
| 创造栏       | 控制物品在哪个分类中出现，独立于 ITEM 注册。能用 <code>/give</code> 获取，不代表它已经加入创造栏。 |

### 先预测，再运行

如果只注册 BLOCK，`/setblock` 可能成功，而 `/give` 会提示未知物品。完成双注册后再比较两个命令。

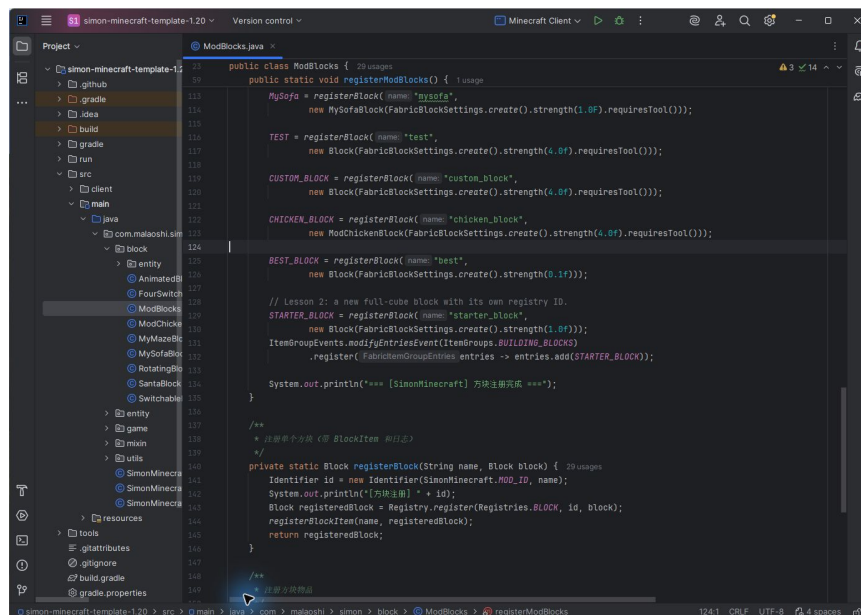
1. 读模板 / 2. 新建 ID / 3. 完成双注册 / 4. 连接资源 / 5. 进入游戏

时间建议：概念 15 分钟，Java 25 分钟，资源 25 分钟，游戏验证 20 分钟，挑战 15–35 分钟。

02 / 11

## 先读懂模板里的 best

沿用模板结构，不重复创建注册类。



真实 IntelliJ 工程：在 block/ModBlocks.java 中查找字段与注册方法。

打开 C:\TeacherMa\Mod\simon-minecraft-template-1.20，在 ModBlocks.java 中找到 BEST\_BLOCK、best 和现有的 registerModBlocks、registerBlock、registerBlockItem 三个方法。

### 保留模板方块

保留 BEST\_BLOCK 和 best。新方块使用 STARTER\_BLOCK 与 starter\_block，不要把已有 ID 再注册一遍，也不要新建第二个 ModBlocks 类。

### 版本锁定

本课固定使用 Minecraft 1.20、Yarn 1.20+build.1、Fabric API 0.83.0+1.20。指定模板使用 JDK 21 运行，Java 编译目标为 17；不要混用新版教程的方法签名。

03 / 11

## 声明并注册入门方块

字段在类中，赋值在已有注册方法中。

ModBlocks.java · 类字段

```
public static Block STARTER_BLOCK;
```

把字段放在类内部、所有方法外部，可以紧挨 BEST\_BLOCK。随后在现有 registerModBlocks() 方法中加入下面的赋值。

ModBlocks.java · registerModBlocks()

```
STARTER_BLOCK = registerBlock(  
    "starter_block",  
    new Block(FabricBlockSettings.create()  
        .strength(1.0f))  
);
```

**starter\_block** 小写注册名，与 MOD\_ID 合成完整 ID。Java 变量名 STARTER\_BLOCK 不等于资源 ID。

**new Block(...)** 普通静态方块直接使用 Block；本课不需要新建子类。

**strength(1.0f)** 设置硬度；第 4 课再对比硬度与其他材料属性。

已有 import · 核对即可

```
import net.minecraft.block.Block;  
import net.fabricmc.fabric.api.object.builder.v1.block.FabricBlockSettings;
```

04 / 11

## 读懂双注册辅助方法

同一个 ID 可以分别进入方块表和物品表。

### 阅读已有代码

下面省略日志，只展示模板辅助方法的职责。若模板已经有这些方法，不要再次粘贴。

ModBlocks.java · registerBlock

```
private static Block registerBlock(
    String name, Block block) {
    Identifier id = new Identifier(
        SimonMinecraft.MOD_ID, name);
    Block registeredBlock = Registry.register(
        Registries.BLOCK, id, block);
    registerBlockItem(name, registeredBlock);
    return registeredBlock;
}
```

ModBlocks.java · registerBlockItem

```
private static void registerBlockItem(
    String name, Block block) {
    Identifier id = new Identifier(
        SimonMinecraft.MOD_ID, name);
    Item blockItem = new BlockItem(
        block, new Item.Settings());
    Registry.register(Registries.ITEM, id, blockItem);
}
```

第一个方法把 Block 放入 Registries.BLOCK，再调用第二个方法注册对应的 BlockItem，最后把返回值赋给 STARTER\_BLOCK。simon-minecraft:starter\_block 在两个不同注册表中各出现一次是正确的；在同一注册表里重复注册则不正确。

05 / 11

## 连接入口与创造栏

先注册，再把物品加入分类。

```
SimonMinecraft.java · 已有入口调用  
  
ModBlocks.registerModBlocks();
```

在已有 onInitialize() 中确认上述调用只执行一次，不要新建第二个入口方法。

```
ModBlocks.java · STARTER_BLOCK 赋值之后  
  
ItemGroupEvents.modifyEntriesEvent(  
    ItemGroups.BUILDING_BLOCKS)  
    .register(entries -> entries.add(STARTER_BLOCK));
```

```
创造栏 import  
  
import net.fabricmc.fabric.api.itemgroup.v1.ItemGroupEvents;  
import net.minecraft.item.ItemGroups;
```

把创造栏代码放入现有 registerModBlocks(), 位于 STARTER\_BLOCK 赋值之后。BUILDING\_BLOCKS 是建筑方块分类; 改成 FUNCTIONAL 只改变分类, 不改变 ID、模型或 /give 的结果。

06 / 11

## 建立四份资源文件

所有位置都相对于 `src/main/resources/assets/simon-minecraft/`。

|                                       |               |
|---------------------------------------|---------------|
| <b>blockstates/starter_block.json</b> | 世界中方块状态的模型入口。 |
|---------------------------------------|---------------|

|                                        |               |
|----------------------------------------|---------------|
| <b>models/block/starter_block.json</b> | 定义世界模型，并引用贴图。 |
|----------------------------------------|---------------|

|                                       |               |
|---------------------------------------|---------------|
| <b>models/item/starter_block.json</b> | 决定背包和手中物品的外观。 |
|---------------------------------------|---------------|

|                        |                                           |
|------------------------|-------------------------------------------|
| <b>lang/zh_cn.json</b> | 提供中文显示名称；英文名称放在 <code>en_us.json</code> 。 |
|------------------------|-------------------------------------------|

### 入口名与引用路径

`blockstates` 与 `models/item` 的入口文件名对应注册名 `starter_block`。被引用的模型或贴图可以另取文件名，只要所有引用一致。资源 ID 中省略 `.json`、`.png` 和 `textures/` 前缀。

07 / 11

## 连接状态、方块和物品模型

三个代码块分别保存为三个文件。

blockstates/starter\_block.json

```
{
  "variants": {
    "": { "model": "simon-minecraft:block/starter_block" }
  }
}
```

models/block/starter\_block.json

```
{
  "parent": "minecraft:block/cube_all",
  "textures": {
    "all": "minecraft:block/emerald_block"
  }
}
```

models/item/starter\_block.json

```
{
  "parent": "simon-minecraft:block/starter_block"
}
```

空字符串 variant 用于本课没有额外属性的静态方块。cube\_all 让六个面共用一张贴图；minecraft:block/emerald\_block 复用游戏已有资源，本课不必新增 PNG。物品模型通过 parent 继承方块模型。

08 / 11

## 添加名称并正确重载

合并语言键，不覆盖已有内容。

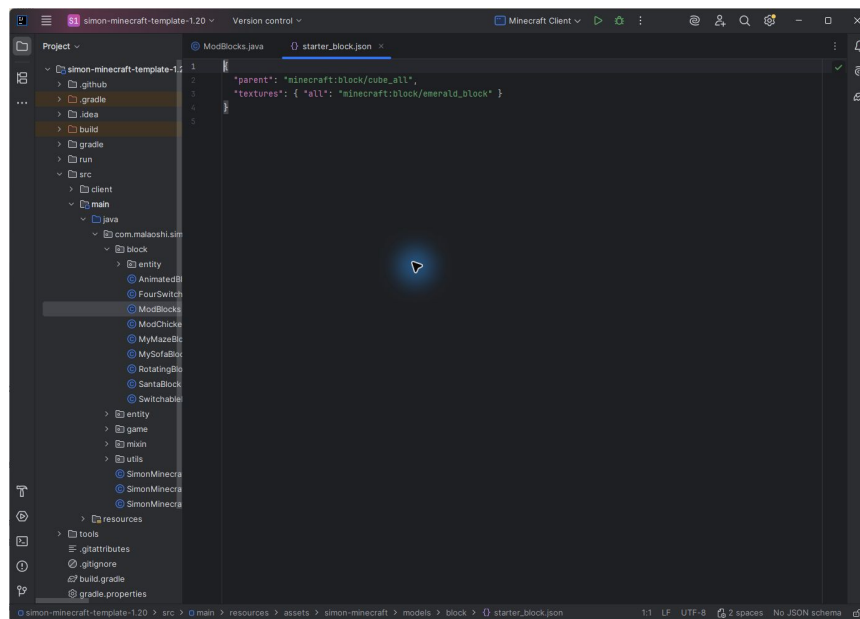
lang/zh\_cn.json · 合并示例

```
{
  "block.simon-minecraft.light": "\u540a\u706f",
  "block.simon-minecraft.starter_block": "\u5165\u95e8\u65b9\u5757"
}
```

lang/en\_us.json · 对应英文

```
{
  "block.simon-minecraft.light": "Light",
  "block.simon-minecraft.starter_block": "Starter Block"
}
```

中文示例中的 Unicode 转义分别显示为“吊灯”和“入门方块”。保留文件中的其他键，键值之间加逗号，最后一项不要尾逗号。游戏选择的语言决定读取 zh\_cn 还是 en\_us，网站语言不会改变游戏设置。



真实 IntelliJ 截图：核对模型路径、贴图引用与语言文件。

### Java 与资源的刷新不同

修改 Java 后停止并重新运行客户端。只改模型、贴图或语言时，可在构建输出已经更新的前提下按 F3+T；若没有变化，先同步资源并重启，不要把旧结果当作新代码的效果。

09 / 11

## 在背包中找到新方块

先验证注册与图标，再验证放置。

游戏聊天框

```
/give @s simon-minecraft:starter_block 16
```



真实游戏截图：创造模式背包中的入门方块。

1. 按 E 打开创造模式背包，搜索“入门方块”；英文游戏搜索 Starter Block。
2. 悬停核对名称和正常图标，不能出现紫黑缺失纹理。
3. 按 F3+H 显示高级提示，核对完整 ID 为 simon-minecraft:starter\_block。

外观像绿宝石方块不代表继承了原版绿宝石的全部行为。贴图只决定外观，行为由我们注册的 Block 和 Settings 决定。

10 / 11

## 放进世界并保存验证

六个面、碰撞与存档都要检查。



真实游戏截图：把独立注册的方块放入测试世界。

1. 手持新方块，右键地面放置，从不同角度检查六个面和亮处的纹理。
2. 靠近方块检查碰撞与选中轮廓，确认是预期的完整立方体。
3. 保存退出并重新进入世界，确认方块仍在，名称与贴图没有变化。

### 本课验收边界

本课验证获取、显示、放置和保存。创造模式破坏通常没有掉落；生存模式的掉落还需要 loot table，第 5 课会补齐。

11 / 11

# 排错与通关挑战

一次检查一个连接点，再用可观察结果验证。

|           |                                                     |
|-----------|-----------------------------------------------------|
| 未知物品      | 检查 BlockItem 注册、入口调用和完整 ID；改 Java 后重启客户端。           |
| 创造栏找不到    | 先用 /give 判断是否已注册，再检查分类回调和 STARTER_BLOCK 赋值顺序。       |
| 世界正常，图标紫黑 | 检查 models/item/starter_block.json 的 parent。         |
| 世界与图标都紫黑  | 沿 blockstates → block model → texture 检查，并查看资源加载日志。 |
| 显示翻译键     | 核对语言 JSON、键名与游戏语言；英文游戏需要 en_us.json。                |
| 重复注册 ID   | 查找同一 ID 的重复注册，以及入口是否调用了两次。                          |

- 1. 挑战 A：把 emerald\_block 改成 gold\_block。先预测世界和背包是否都改变，再验证；ID 与名称应保持不变。
- 2. 挑战 B：新建 copper\_starter，独立完成 Java 注册、三份模型资源和语言名称。
- 3. 用自己的话解释 Block、BlockItem、创造栏的三个职责。下一课把 Blockbench 吊灯接入这条流程。

## 官方参考

教程按指定的 Fabric 1.20 模板编写；查阅文档时请保持版本一致。

Yarn 1.20: Block

Fabric API 0.83.0: ItemGroupEvents