Minecraft 1.12.2 Forge Mod 开发教程-Sphinx

发布 *1.0*

2020 年 04 月 15 日

Contents

首先声明一下，编写这篇教程是为了让更多人能够真正做出一个 Mod 来。

如果你看过其他人的教程，你会发现这些教程要么杂碎，要么作者弃坑。而这篇教程将会帮助你在自己的 Mod 中添加各种新特性。

仓库地址：https://github.com/squid233/MC1.12.2ModTutorial

如果你对源码感兴趣的话可以看一下。

如果还有问题可以在 MCBBS 私聊我，用户名：squid233

说实话在我重写时再看一遍自己写的文章完全看不懂……所以我就重写了一遍。

Forge 官方中文文档

CHAPTER 1

目录:

1.1 构建开发环境

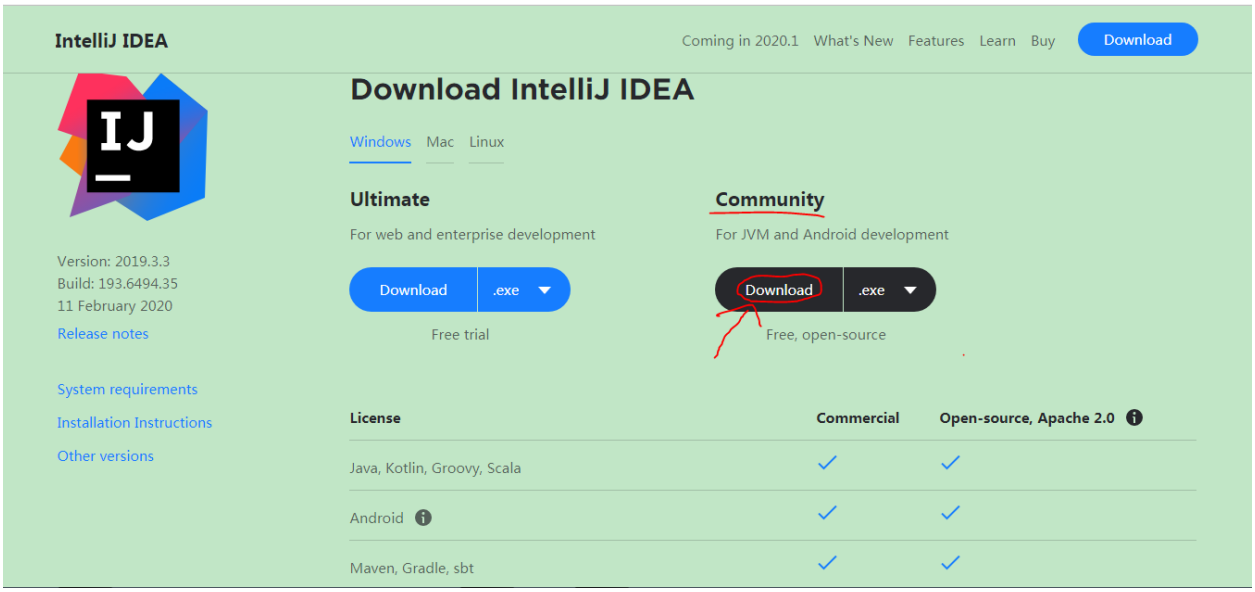
这一章介绍了如何构建开发环境。

1.1.1 JDK 下载

若想制作 Mod，必须先构建好开发环境。你需要下载的版本是 JDK1.8.0

1.1.2 IDE 下载

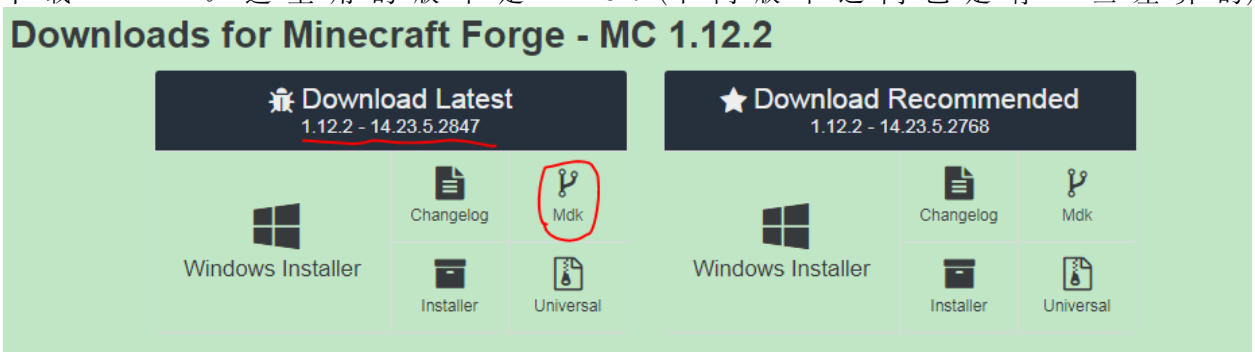
Java 运行时环境已经安装完成，接下来就是选择 IDE 了。这里推荐使用 IntelliJ IDEA。



选择右边免费的社区版下载并安装。

1.1.3 Forge MDK 包下载

IDEA 安装好后，该下载 Forge MDK 包了。点击左侧的 1.12.2，点击下载 MDK。这里用的版本是 2847（不同版本之间也是有一些差异的）



包下载好后，将其解压到**全英文路径**（如果不是全英文路径你会遇到一些奇怪的问题）

1.1.4 正式开始构建开发环境

按下 Shift ＋右键，选择在此处打开命令窗口或 PowerShell

输入：

```
gradlew setupDecompWorkspace
```

这个过程非常耗时间，建议大家搜索 Lss233’s.Mirror();

如果出现 BUILD FAILED 请重新输入该指令

```
gradlew genIntellijRuns
```

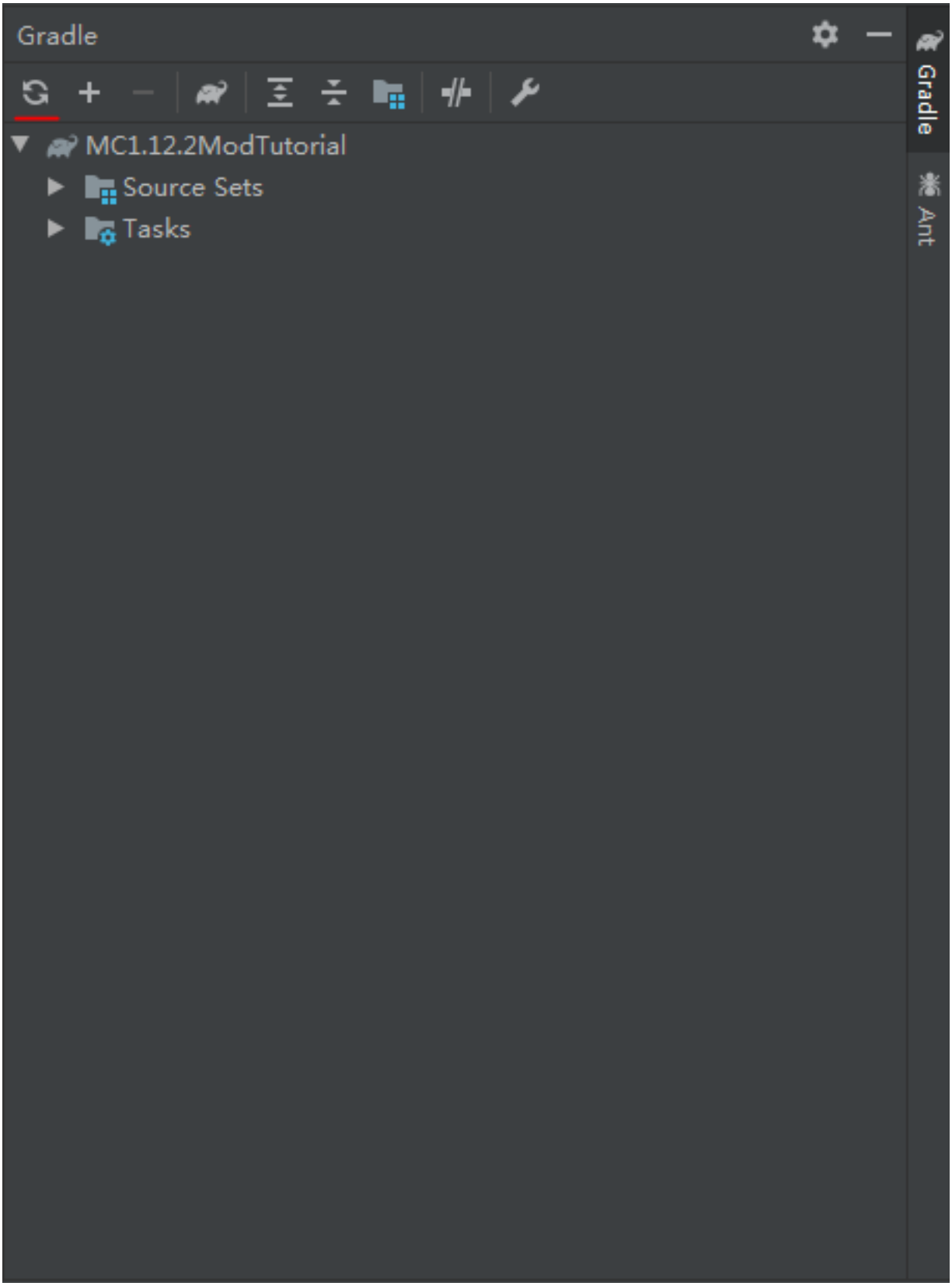
如果出现 BUILD FAILED 请重新输入该指令

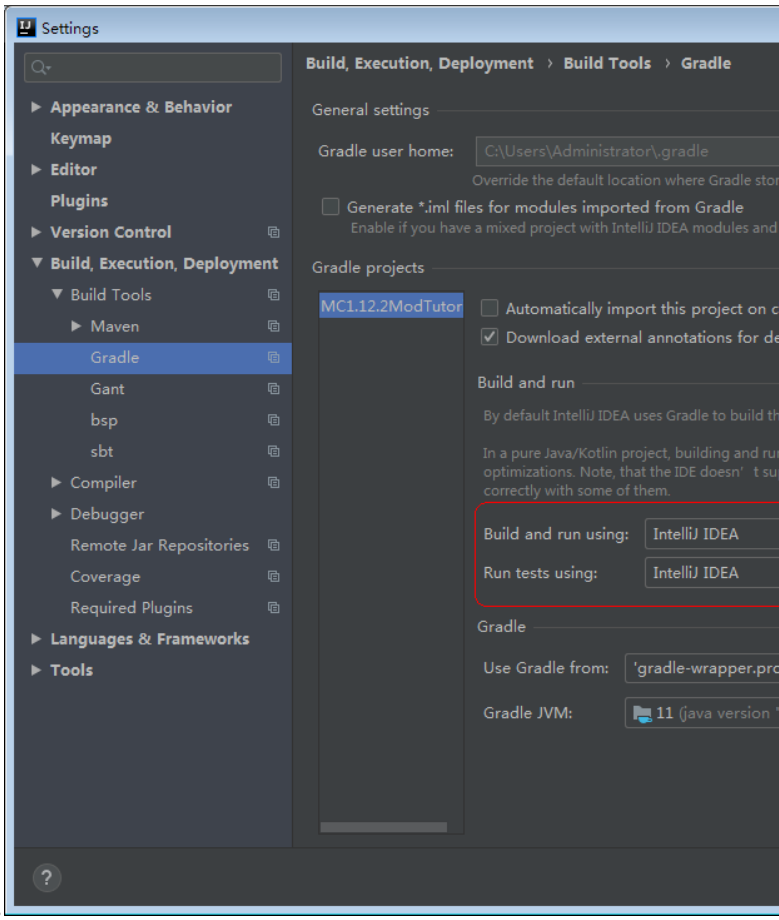
当出现 BUILD SUCCESSFUL 的时候，差不多就大功告成了。

注：BUILD SUCCESSFUL 长这个样子：

```
BUILD SUCCESSFUL in 3m 19s
10 actionable tasks: 9 executed, 1 up-to-date
```

之后将其导入 IDEA，并点击 Gradle 面板中的刷新按钮，以确保你的项目能正常工作。


Gradle
MC1.12.2ModTutorial
Source Sets
Tasks
Gradle
Ant



还需要设置一下:按下 Ctrl+Alt+S,然后按图片中的设置。

按下 Ctrl+Alt+Shift+S，将 `Project Compiler Output` 设定为 `~\out`，其中 `~` 是你的 Mod 目录。最后，指定 Project SDK 为 1.8 版本。

到此为止，工作环境已经配置好了，下一期我们将配置主类和 Mod 信息

1.2 主类、Mod 信息与代理

这一章介绍了如何配置主类、Mod 信息与代理。

1.2.1 mcmod.info

在 resources 文件夹中，有一个 mcmod.info 文件，它储存了 Mod 的信息，我们需要去编辑它。

modid

将 modid 改为你的 Mod 的命名空间，例如 examplemod

正确的示例：- examplemod - example*mod* - *example*mod_1

错误的示例：- ExampleMod

modid 应以 a-z 0-9 _ 命名，并且不允许大写。

name

name 可以是 Mod 的名称。

description

用一两段话来介绍你的 Mod。

version

Mod 的版本。具体命名规则会在以后讲到。

mcversion

Mod 支持的 Minecraft 的版本。

authorlist

Mod 作者的列表。

credits

可以说出任何你想感谢的东西。

将这些东西改好以后，就可以进入下一步了。

1.2.2 主类

每个 Mod 都会有一个主类，每一个主类都会有一个 `@Mod` 注解。下面这个例子展示出了 `@Mod` 注解如何使用（注：以后所有的例子都会以 ExampleMod 为准，请根据自己的需求更改）：

```
@Mod(modid = ExampleMod.MODID, name = ExampleMod.NAME, version = ExampleMod.VERSION)
public class ExampleMod
{
    public static final String MODID = "examplemod";
    public static final String NAME = "Example Mod";
    public static final String VERSION = "1.0";
}
```

MODID, NAME, VERSION 就不必解释了。接下来我们在 `public static final String VERSION = "1.0";` 这段代码的后面添加上下面的代码：

```
@Mod.Instance(ExampleMod.MODID)
    private static ExampleMod instance;
```

这段代码用于保存 Mod 的实例。

然后创建两个新类，分别是 `com.example.examplemod.proxy.ClientProxy` 和 `com.example.examplemod.proxy.CommonProxy`。

在 `CommonProxy` 中添加如下代码：

```
public void preInit(FMLPreInitializationEvent event) {

    }

public void init(FMLInitializationEvent event) {

    }
```

然后在 ClientProxy 中添加如下代码：

```
@Override
public void preInit(FMLPreInitializationEvent event) {
    super.preInit(event);
}

@Override
public void init(FMLInitializationEvent event) {
    super.init(event);
}
```

注意：`ClientProxy` 继承于 `CommonProxy`。

现在回到 `ExampleMod` 类，在 `private static ExampleMod instance;` 的下面加上如下代码：

```
@SidedProxy
        (clientSide = "com.example.examplemod.proxy.ClientProxy",
         serverSide = "com.example.examplemod.proxy.CommonProxy"
        )
private static CommonProxy proxy;
```

接下来在主类的 `preInit` 方法中加上 `proxy.preInit(event);`，在 `init` 方法中加上 `proxy.init(event);`。

1.2.3 示例代码

ExampleMod.java

```
package com.example.examplemod;

import com.example.examplemod.proxy.CommonProxy;
import net.minecraft.init.Blocks;
import net.minecraftforge.fml.common.Mod;
import net.minecraftforge.fml.common.Mod.EventHandler;
import net.minecraftforge.fml.common.SidedProxy;
import net.minecraftforge.fml.common.event.FMLInitializationEvent;
import net.minecraftforge.fml.common.event.FMLPreInitializationEvent;
import org.apache.logging.log4j.Logger;

@Mod(modid = ExampleMod.MODID, name = ExampleMod.NAME, version = ExampleMod.VERSION)
public class ExampleMod
{
    public static final String MODID = "examplemod";
    public static final String NAME = "Example Mod";
    public static final String VERSION = "1.0";

    @Mod.Instance(ExampleMod.MODID)
    private static ExampleMod instance;

    @SidedProxy
            (clientSide = "com.example.examplemod.proxy.ClientProxy",
            serverSide = "com.example.examplemod.proxy.CommonProxy"
            )
    private static CommonProxy proxy;

    private static Logger logger;

    @EventHandler
    public void preInit(FMLPreInitializationEvent event)
    {
        logger = event.getModLog();
        logger.info("preInit");
        proxy.preInit(event);
    }
```

(下页继续)

(续上页)

```
    @EventHandler
    public void init(FMLInitializationEvent event)
    {
        // some example code
        logger.info("DIRT BLOCK >> {}", Blocks.DIRT.getRegistryName());
        proxy.init(event);
    }
}
```

CommonProxy.java

```
package com.example.examplemod.proxy;

import com.example.examplemod.register.*;
import net.minecraftforge.fml.common.event.FMLInitializationEvent;
import net.minecraftforge.fml.common.event.FMLPreInitializationEvent;

public class CommonProxy {
    public void preInit(FMLPreInitializationEvent event) {

    }

    public void init(FMLInitializationEvent event) {

    }

}
```

ClientProxy.java

```
package com.example.examplemod.proxy;

import net.minecraftforge.fml.common.event.FMLInitializationEvent;
import net.minecraftforge.fml.common.event.FMLPreInitializationEvent;

public class ClientProxy extends CommonProxy{
    @Override
    public void preInit(FMLPreInitializationEvent event) {
```

(下页继续)

(续上页)

```
        super.preInit(event);
    }

    @Override
    public void init(FMLInitializationEvent event) {
        super.init(event);
    }
}
```

你的 Mod 最基本的已经配置完成，很快我们将做出第一个物品。

1.3 物品

这一章介绍了如何添加物品、注册物品以及给物品添加更多功能。

1.3.1 创建物品

如果你想偷懒，直接 `Item item = new Item();` 就可以了。但是这只适用于那些无功能的物品，如钻石。我们要做的是一个有功能的物品。

新建物品类

就拿 `Bubble` 当例子吧。

由于大部分的物品都继承于 `Item` 类，所以我们也要写一个类来**继承**它。

首先，新建 `com.example.examplemod.common.item.ItemBubble` 类。

然后，粘贴:

```
private static String name = "bubble";
public ItemBubble() {
    this.setRegistryName(name);
    this.setUnlocalizedName(ExampleMod.MODID+"."+name);
    this.setCreativeTab(CreativeTabs.MISC);
}
```

`setRegistryName`: 设置物品的注册名称

`setUnlocalizedName`: 设置物品的未本地化名称

setCreativeTab: 设置物品的创造模式物品栏位置

现在我们需要注册这个物品。

注册物品

新建 com.example.examplemod.register.ItemsRegister 类。

首先在 public class ItemsRegister 上面添加 @Mod.EventBusSubscriber。

然后添加 public static final Item BUBBLE = new ItemBubble();。创建一个构造方法。

```
public ItemsRegister() {
    MinecraftForge.EVENT_BUS.register(this);
}
```

这里我们不使用古老的 GameRegistry 方法来注册物品，而是使用 Forge 推荐的注册方法。

```
@SubscribeEvent
    public static void registerItems(RegistryEvent.Register<Item> event) {
        event.getRegistry().registerAll(
                BUBBLE
    );
}
```

Forge 提供了大量的注册表，其中包括 Item、Block、Biome 等等。这些我们会在后面讲到。

你还可以在 BUBBLE 的下面添加更多的物品，只要确保你用 public static final Item xxx = new ItemXXX(); 声明了一个物品并创建了对应的类，以及每个物品之间用了逗号隔开。

最后我们只需要在 CommonProxy 类中的的 preInit 方法中 new ItemsRegister() 即可。

现在物品已经成功注册了，运行游戏看一下效果：

已经可以在创造模式物品栏的杂项中找到了。

1.3.2 本地化与国际化

什么是本地化与国际化

用官方的话说就是

> 国际化，简称为 i18n，是一种可以适用于各种语言的代码设计思路。 本地化是指显示适用于用户语言的文字的过程 。
>
> I18n 用 _ 翻译密钥 (translation keys)_ 实现。 翻译密钥是一个字符串，用于标识一段没有特定语言的可显示文本。 例如，翻译密钥 tile.dirt.name 指的是泥土方块的名字。 这样，可以引用可显示的文本而不关心特定语言。 加入新语言时，代码不需要修改。
>
> 本地化将在游戏的本地设置中发生。 在 Minecraft 客户端中，位置由语言设置指定。 在专用服务器上，唯一受支持的语言环境是 en_US。 可以在 Minecraft Wiki 上找到可用语言环境的列表。

因为 Minecraft 是一个面向全世界的游戏，我们不能忽视 Minecraft 的玩家来自世界各地，他们使用不同的语言。为了让更多人可以游玩这个游戏，所以 Minecraft 添加了引入了国际化与本地化支持。本地化大部分时间是在翻译外国作品，而这些翻译必须要考虑到地区差异。正如同酒石酸菌说的一样，本地化 MOD 不要使用机翻，这会使得翻译不准确或无法理解，所以我们在翻译是应该考虑本地人民的语言习惯以及在游戏中实质的作用。

了解一个语言文件

我说了那么多，但是我们是来翻译 MOD 的，我们要知道怎么给我们的 MOD 创建出来一个语言文件以方便其他人对你的 MOD 进行翻译。

如果你按照之前的步骤注册出了一个物品，那么你会发现那个物品的名字不对。它的名字叫 `item.exeamplemod.bubble.name`

这个名字是在创建物品时 `this.setUnlocalizedName(ExampleMod.MODID+"."+name)` 这个语句进行定义的。由此我们可以知道它的名字叫 `MODID.` *物品名* `.name` 再加上 Minecraft 的物品前缀 `item`，就是他的的本地化名称。如果你实在不知道它的本地化名称，你也可以进入游戏查看它的名称得知它的本地化命名。

我们在 `assets/[modid]/lang` 创建一个文件 `en_us.lang`。这里需要注意不是 `en_US`, 因为 1.11 开始，资源包格式版本号更新到了 3。自此，资源包中所有文件名强制使用 *小写字母*（*正则表达式* `[a-z0-9_]+`)。所以你应该使用的是 `en_us.lang`。

资源包格式版本	文件名	Minecraft 版本
1	en_US.lang	1.6
2	en_US.lang	1.8
3	en_US.lang	1.11
4	en_us.lang	1.13
5	en_us.lang	1.15

到这里你可能会疑惑，我们是中国为什么我们要创建 `en_us.lang` 而不是 `zh_cn.lang`。在这里我们只是以 `en_us.lang` 为示例，`zh_cn.lang` 同理, 在此我推荐你们创建两个语言文件 `en_us` 和 `zh_cn`。

制作一个语言文件

1. 在 `assets/[modid]/lang` 创建两个文件 `en_us.lang` 和 `zh_cn.lang`。在这里为 `assets/exeamplemod/lang`。
2. 在 `en_us` 写入 `item.exeamplemod.bubble.name=Bubble`

   在 `zh_cn` 写入 `item.exeamplemod.bubble.name=` *泡泡*

   并保存文件
3. 重启游戏，进入游戏你就可以看到你的翻译了。

如果你有闲功夫，可以考虑去酒石酸菌的 1.12.2 的通用汉化项目看看，如果有兴趣请点击 这里 。

本文大部分内容都由 **吃货一枚**提供

1.3.3 添加物品贴图

物品已经翻译好了，接下来就该添加物品材质了。

首先创建文件夹 `assets/[modid]/textures/items`，将 `bubble.png` 放进去：

之后还需要创建文件 `assets/[modid]/models/item/bubble.json`。文件的内容如下：

```
{
    "parent": "minecraft:item/generated",
    "textures": {
        "layer0": "examplemod:items/bubble"
    }
}
```

- `item/generated`：使用默认的物品渲染方式
- `layer0`：使用第 0 层图层
- `examplemod:items/bubble`：寻找 assets 文件夹下的 examplemod 下的 textures 下的 items 下的 bubble.png（好长

现在物品模型渲染文件也做好了，接下来，到注册材质的时候了。

注册物品材质

要注册材质，我们首先在 register 文件夹下新建 `ModelsRegister` 类。
然后我们需要输入

```
public ModelsRegister() {
    MinecraftForge.EVENT_BUS.register(this);
}
```

注：这次不需要在开头加上注解了。

接着我们需要

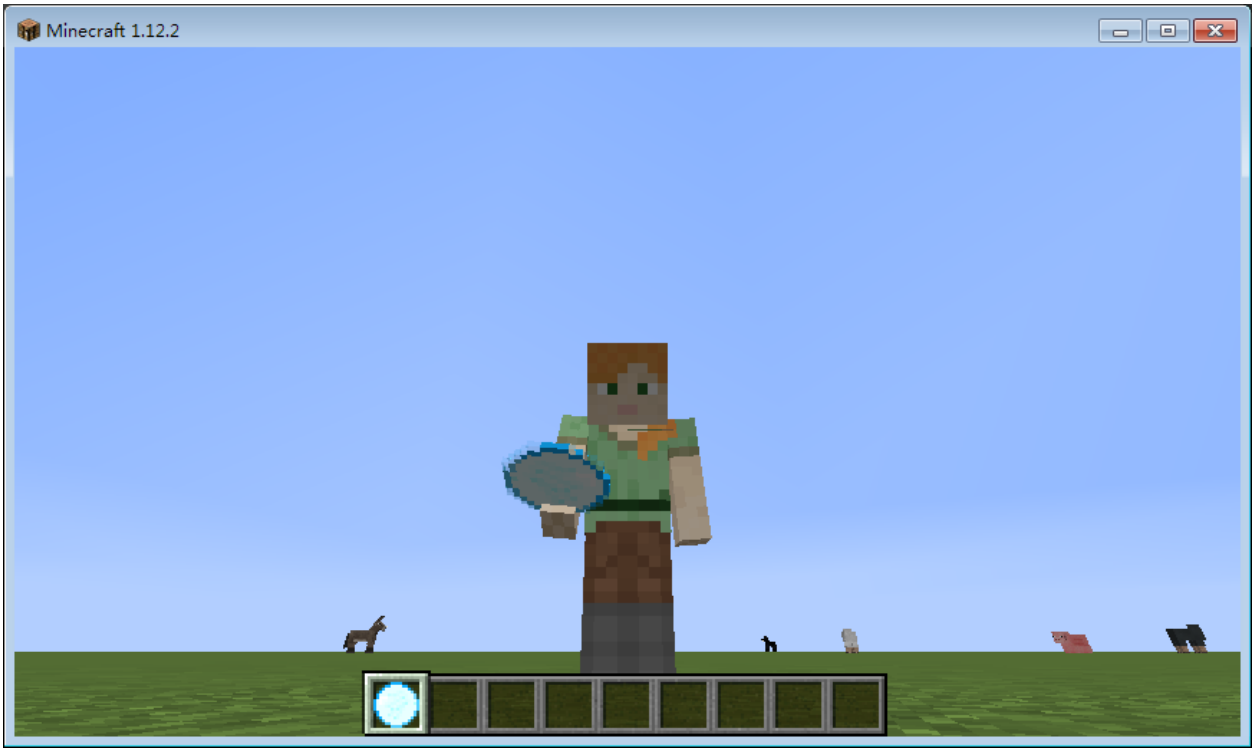
```
@SubscribeEvent
public void registerModels(ModelRegistryEvent event) {
    registerModel(ItemsRegister.BUBBLE);

}

private void registerModel(Item item) {
    ModelLoader.setCustomModelResourceLocation(item, 0, new ModelResourceLocation(item.
↪getRegistryName(), "inventory"));
}
```

- `registerModel(ItemsRegister.BUBBLE);`：调用 `registerModel` 方法注册物品材质，BUBBLE 可以换成其他的已经注册的物品
- `ModelLoader.setCustomModelResourceLocation(item, 0, new ModelResourceLocation(item.getRegistryName(), "inventory"));`：注册物品模型。0 是 Metadata。其中”inventory”一般不需要更改。

不要把方法名搞混了！

物品模型已经注册完成，你只需要在 **ClientProxy** 类中的 `preInit` 方法中 `new ModelsRegister();` 即可。

让我们进游戏看一下吧：



1.3.4 添加食物

我们来添加一个食物。

并不是所有的物品都继承于 Item 类，例如工具继承于 ItemTool 类，盔甲继承于 ItemArmor 类，而食物继承于 ItemFood 类。

ItemFood 类的构造函数有这些：

```
public ItemFood(int amount, float saturation, boolean isWolfFood)
{
    this.itemUseDuration = 32;
    this.healAmount = amount;
    this.isWolfsFavoriteMeat = isWolfFood;
    this.saturationModifier = saturation;
    this.setCreativeTab(CreativeTabs.FOOD);
}

public ItemFood(int amount, boolean isWolfFood)
{
    this(amount, 0.6F, isWolfFood);
}
```

amount：回复的饥饿值。int 整数类型。

saturation：回复的饱食度。饱食度越高回血速度就越快。float 浮点数类型。

isWolfFood：狼是否可食用。true 为可以，false 为不能。

其中，从上看出 saturation 是可选的。默认值为 0.6f。

首先创建一个类，名字叫做 `ItemFoodChestnut`。之后，在类里添加如下代码：

```
public ItemFoodChestnut() {
    super(1, 0.6f, false);
    String name = "chestnut";
    this.setRegistryName(name).setUnlocalizedName(ExampleMod.MODID+"."+name).
↪setCreativeTab(CreativeTabs.FOOD);
}
```

只需要注册这个食物以及添加物品贴图，再进行国际化与本地化就可以了。

我们希望这个食物能在饱的时候也能吃，因此，再添加这一段代码：`setAlwaysEdible();`

我们还希望可以在吃掉后玩家会获得效果，因此，参考原版金苹果的代码，我们需要重写 onFoodEaten 方法。

```
@ParametersAreNonnullByDefault
@Override
protected void onFoodEaten(ItemStack stack, World worldIn, EntityPlayer player) {
    if (!worldIn.isRemote) {
        player.addPotionEffect(new PotionEffect(Potion potionIn, int durationIn, int␣
↪amplifierIn));
    }
    super.onFoodEaten(stack, worldIn, player);
}
```

potionIn：药水效果。例如 MobEffects.SPEED 就是速度。

durationIn：效果时长。理论上最长是 2147483647，但原版用命令时最高为 1000000。

amplifierIn：效果等级。最高为 255。

将这些替换成你想要的数据，就可以了。

注意：一定要将食物类继承 ItemFood 类！

饮料同理。

示例

```
package com.example.examplemod.common.item;

import com.example.examplemod.ExampleMod;
import net.minecraft.creativetab.CreativeTabs;
import net.minecraft.entity.player.EntityPlayer;
import net.minecraft.init.MobEffects;
import net.minecraft.item.ItemFood;
import net.minecraft.item.ItemStack;
import net.minecraft.potion.PotionEffect;
import net.minecraft.world.World;

import javax.annotation.ParametersAreNonnullByDefault;

public class ItemFoodChestnut extends ItemFood {
    public ItemFoodChestnut() {
        super(1, 0.6f, false);
        String name = "chestnut";
```

(下页继续)

(续上页)

```
        this.setRegistryName(name).setUnlocalizedName(ExampleMod.MODID+"."+name).
↪setCreativeTab(CreativeTabs.FOOD);
        this.setAlwaysEdible();
    }

    @ParametersAreNonnullByDefault
    @Override
    protected void onFoodEaten(ItemStack stack, World worldIn, EntityPlayer player) {
        if (!worldIn.isRemote) {
            player.addPotionEffect(new PotionEffect(MobEffects.SPEED, 1000000, 255));
        }
        super.onFoodEaten(stack, worldIn, player);
    }
}
```

1.3.5 合成

简单的配方文件

一个简单的有序合成配方应该如下：

```
{
    "type": "minecraft:crafting_shaped",
    "pattern": [
            "xxx",
            "xax",
            "xxx"
    ],
    "key": {
            "x": {
                    "item": "minecraft:water_bucket"
            },
            "a": {
                    "type": "forge:ore_dict",
                    "ore": "gemDiamond"
            }
    },
    "result": {
            "item": "examplemod:bubble",
```

(下页继续)

(续上页)

```
            "count": 9
    }
}
```

这个配方允许你用 8 个水桶来合成 9 个泡泡。

合成类型

你可以把它当作定义用哪一种合成布局，例如 minecraft:crafting_shaped(有序合成) 或 minecraft:crafting_shapeless(无序合成)。

你也可以定义你自己的合成类型，只需要使用 _factories.json 文件。

有序合成

有序合成需要 pattern 和 key 两个关键字。

pattern 定义物品的排序，它必须是占位符的形式。每一个物品你都可以选择任意的一个字符作为占位符。

key 定义了占位符对应的物品。可以添加附加属性 forge:ore_dict，它可以定义物品是矿物词典的一部分。例如，无论什么铜矿都可以生成铜锭。要这样的效果的话，要用 ore 标签定义物品，而不是用 item 定义那个物品。

默认有很多这样的类型，你也可以自己定义。

data 是一个可选标签，用于定义方块或物品的 metadata。

无序合成

无序合成不需要 pattern 和 key 关键字。

要定义一个无序合成,你要使用 ingredients 列表。它定义了合成中要用的物品,也可以用 forge:ore_dict 它函数式的声明在上方。默认有很多这样的类型，你也可以自己定义。它甚至可以一个对象定义多个实例，意味着合成时必须要在合成台里放多个这样的物品。

提示 你的配方里有多少 ingredients 没有限制，原版合成台没有只允许一个合成里放 9 个物品。

```
{
    "type": "crafting_shapeless",
    "ingredients": [{
        "item": "examplemod:bubble"
    },
    {
        "item": "examplemod:bubble"
```

(下页继续)

(续上页)

```
    }
    ],
    "result": {
        "item": "minecraft:diamond",

    }
    }
```

这个配方允许你用 2 个泡泡来合成 1 颗钻石。

加载配方

只需要将配方文件放到./assets/<modid>/recipes/ 目录下即可。配方文件必须为标准的 JSON 格式。Be JSON 是一个很好的在线检查 JSON 格式是否正确的网站。

熔炼

要定义一个熔炼配方，我们只需要用 1 行代码即可。新建一个 SmeltingRecipes 类，并在里面写上：

```
public SmeltingRecipes() {
        GameRegistry.addSmelting(ItemsRegister.BUBBLE, new ItemStack(ItemsRegister.
↪BUBBLE, 2), 0.9f);
    }
```

这样当你烧完泡泡的时候，就会出现 2 个泡泡，并获得 0.9 点经验。

最后我们只需要在 CommonProxy 的 init 方法中 new SmeltingRecipes(); 即可。

1.3.6 ItemStack

Minecraft 1.12.2 原版只有 4096 个物品 ID（某些 Mod 会将其提升至 int 上限），当我们想写大一些的 Mod 的话，可不能随便乱用 Item 声明那个物品。因此，我们要用 ItemStack 来声明那些物品。

ItemStack 可以让多个物品使用同一个 ID，例如一些 mod 的矿石，虽然是多个物品，但是只用了 1 个 ID：ore。

ItemStack 使用 metadata 来区分物品。

声明一个 ItemStack

我们来试着添加一个 ItemStack。该 ItemStack 含有 low_coal, medium_coal, high_coal, super_coal。

```
Item coals = new Item();
public ItemStack low_coal = new ItemStack(coals);
public ItemStack medium_coal = new ItemStack(coals, 1, 1);
public ItemStack high_coal = new ItemStack(coals, 1, 2);
public ItemStack super_coal = new ItemStack(coals, 1, 3);
```

coals: 表示该物品属于 coals。

数量 1: 通常情况下为 1。

metadata1: 表示该物品的 metadata 值为 1。

实战

首先，新建一个类，名叫 ItemCoals。并让其继承 Item 类。然后添加上面说的代码。

接着，声明一个构造函数，里面写上

```
coals.setHasSubtypes(true);
coals.setRegistryName("coals");
low_coal.getItem().setUnlocalizedName(ExampleMod.MODID+"."+"low_coal").
↪setCreativeTab(CreativeTabs.MISC);
medium_coal.getItem().setUnlocalizedName(ExampleMod.MODID+"."+"medium_coal").
↪setCreativeTab(CreativeTabs.MISC);
high_coal.getItem().setUnlocalizedName(ExampleMod.MODID+"."+"high_coal").
↪setCreativeTab(CreativeTabs.MISC);
super_coal.getItem().setUnlocalizedName(ExampleMod.MODID+"."+"super_coal").
↪setCreativeTab(CreativeTabs.MISC);
```

使用 getItem(); 可以把 ItemStack 转为 Item。

再注册物品和物品贴图就好了。

这样，当我们输入/give @p examplemod:coals 1 1 时就会获得 medium_coal。

此页待完善。

1.3.7 创造模式物品栏

创建 CreativeTab 类，在里面写上：

```
public static final CreativeTabs EXAMPLE_CREATIVE_TAB = new CreativeTabs("example") {
    @SideOnly(Side.CLIENT)
```

(下页继续)

(续上页)

```
    @Override
    public ItemStack getTabIconItem() {
        return new ItemStack(ItemsRegister.BUBBLE);
    }
};
```

其中 ItemsRegister.BUBBLE 是创造模式物品栏的图标。

完成后把物品改一下：

```
this.setCreativeTab(CreativeTab.EXAMPLE_CREATIVE_TAB);
```

再启动游戏就能看到物品了。

1.4 方块

这一章介绍了如何添加方块。

1.4.1 添加方块

是时候添加方块了。新建 common.block 包，并新建 ExampleBlock 类，然后让 ExampleBlock 类继承 Block 类。在里面写上如下代码：

```
public ExampleBlock() {
    super(Material.GROUND, MapColor.DIRT);
    String name = "example_block";
    this.setRegistryName(name).setUnlocalizedName(ExampleMod.MODID+"."+name).
↪setCreativeTab(CreativeTabs.MISC).setHardness(0.5f).setResistance(10);
}
```

setHardness: 设置方块硬度。

setResistance: 设置方块爆炸抗性。

注册方块

我们在 register 包下新建 BlocksRegister 类，在里面写上如下代码：

```
@Mod.EventBusSubscriber
public class BlocksRegister {
    public static final Block EXAMPLE_BLOCK = new ExampleBlock();

    public BlocksRegister() {
        MinecraftForge.EVENT_BUS.register(this);
    }

    private static Block[] blocks = {
            EXAMPLE_BLOCK
    };

    @SubscribeEvent
    public static void registerBlocks(RegistryEvent.Register<Block> event) {
        for (Block block : blocks) {
            ModelLoader.setCustomModelResourceLocation(Item.getItemFromBlock(block), 0,␣
↪new ModelResourceLocation(block.getRegistryName(), "inventory"));
            event.getRegistry().register(block);
        }
    }

    @SubscribeEvent
    public static void registerItemBlocks(RegistryEvent.Register<Item> event) {
        for (Block block : blocks) {
            Item itemBlock = new ItemBlock(block).setRegistryName(block.
↪getRegistryName());
            ModelLoader.setCustomModelResourceLocation(itemBlock, 0, new␣
↪ModelResourceLocation(block.getRegistryName(), "inventory"));
            event.getRegistry().register(itemBlock);
        }
    }
}
```

方块就注册完成了。

I18n

我们在 en_us.lang 中写上

```
tile.examplemod.example_block.name=Example Block
```

这样就能看到方块的名字了。

添加方块模型

```
Blockstate: src/main/resources/assets/examplemod/blockstates/example_block.json
Block Model: src/main/resources/assets/examplemod/models/block/example_block.json
Item Model: src/main/resources/assets/examplemod/models/item/example_block.json
Block Texture: src/main/resources/assets/examplemod/textures/block/example_block.png
```

blockstates/example_block.json

```
{
"variants": {
    "": { "model": "examplemod:block/example_block" }
    }
}
```

models/block/example_block.json

```
{
"parent": "block/cube_all",
"textures": {
    "all": "examplemod:block/example_block"
    }
}
```

models/item/example_block.json

```
{
    "parent": "examplemod:block/example_block"
}
```

textures/block/example_block.png

掉落物

在 ExampleBlock 中添加：

```
@Override
public Item getItemDropped(IBlockState state, Random rand, int fortune) {
    return Item.getItemFromBlock(this);
}
```