EasyItem

简介

EasyItem是一个基于TabooLib，使用Kotlin编写的固定物品库插件

交流

QQ群: 648142579

安装

需求

1.12.2-1.19.3 bukkit服务端

已测试：

- paper1.12.2-1.19.3
- arclight1.16.5
- spigot1.12.2
- catserver1.12.2

安装EasyItem

EasyItem-自动构建

1. 点击上方链接，通过GitHub下载EasyItem插件
2. 将文件丢入plugins文件夹
3. 重启服务器

注意

> ! 不要尝试通过Plugman等热重载工具加载EasyItem

默认配置

config.yml

```yaml
Main:
  MMItemsPath: MMItems.yml
  Debug: false
Messages:
  invalidPlayer: §e[EI] §6玩家不在线或不存在
  successInfo: §e[EI] §6成功给予 §f{player} §a{amount} §6个 §f{name}
  givenInfo: §e[EI] §6你得到了 §a{amount} §6个 §f{name}
  dropSuccessInfo: §e[EI] §6成功在 §a{world} §6的 §a{x},{y},{z} §6掉落了 §a{
  unknownItem: §e[EI] §6找不到ID为 §a{itemID} §6的物品
  existedKey: §e[EI] §6已存在ID为 §a{itemID} §6的物品
  onlyPlayer: §e[EI] §6该指令仅可玩家使用
  successSaveInfo: §e[EI] §6成功将 §f{name} §6以ID §a{itemID} §6保存至 §a{pa
  mmImportSuccessInfo: §e[EI] §6成功将所有MM物品保存至 §a{path}
  airItem: §e[EI] §6请不要试图保存空气, 谢谢合作
  invalidAmount: §e[EI] §6无效数字
  invalidWorld: §e[EI] §6无效世界
  invalidLocation: §e[EI] §6无效坐标
  insufficientPermissions: §e[EI] §6权限不足
  reloadedMessage: §e[EI] §6重载完毕
  invalidNBT: §6[EI] §cNBT加载失败, 请勿在列表型NBT中混用键值对, 数字及字符串
  invalidItem: '§6[EI] §c物品加载失败, 物品可能缺损数据, 物品ID: §6{itemID}'
  failureInfo: '§e[EI] §6物品给予失败, 可能原因: 物品未配置材质/玩家已下线'
  invalidMaterial: '§e[EI] §6物品 {itemID} 使用了未知的材质 {material}'
  clickGiveMessage: §e点击获取该物品
Help:
  prefix: |-
    §6====================§eEasyItems§6====================
    §6==================[]为必填, ()为选填==================
  suffix: §6=================<< §e{prev} §f{current}§e/§f{total} §e{next} §
  amount: 10
  format: "{command} §7> {description}"
  prev: 上一页
  next: 下一页
  commands:
    list:
      command: §e/ei §flist (页码)
      description: 查看所有EI物品
```

```
    get:
      command: §e/ei §fget [物品ID] (数量) (是否反复随机) (指向数据)
      description: 根据ID获取EI物品
    give:
      command: §e/ei §fgive [玩家ID] [物品ID] (数量) (是否反复随机) (指向数据)
      description: 根据ID给予EI物品
    giveAll:
      command: §e/ei §fgiveAll [物品ID] (数量) (是否反复随机) (指向数据)
      description: 根据ID给予所有人EI物品
    drop:
      command: §e/ei §fdrop [物品ID] [数量] [世界名] [X坐标] [Y坐标] [Z坐标] [
      description: 于指定位置掉落EI物品
    save:
      command: §e/ei §fsave [物品ID] (保存路径)
      description: 将手中物品以对应ID保存至对应路径
    cover:
      command: §e/ei §fcover [物品ID] (保存路径)
      description: 将手中物品以对应ID覆盖至对应路径
    mm load:
      command: §e/ei §fmm load [物品ID] (保存路径)
      description: 将对应ID的MM物品保存为EI物品
    mm cover:
      command: §e/ei §fmm cover [物品ID] (保存路径)
      description: 将对应ID的MM物品覆盖为EI物品
    mm loadAll:
      command: §e/ei §fmm loadAll (保存路径)
      description: 将全部MM物品转化为EI物品
    reload:
      command: §e/ei §freload
      description: 重新加载EI物品
    help:
      command: §e/ei §fhelp
      description: 查看帮助信息
ItemList:
  Prefix: §6===========§eEasyItems§6===========
  Suffix: §6======<< §e{prev} §f{current}§e/§f{total} §e{next} §6>>======
  ItemAmount: 10
  ItemFormat: §6{index}. §a{ID} §6- §f{name}
  Prev: 上一页
  Next: 下一页
```

Items/ExampleItem.yml

```yaml
ExampleItem:
  # 物品材质
  material: LEATHER_HELMET
  # 物品CustomModelData(适用于1.14+)
  custommodeldata: 1
  # 物品损伤值
  damage: 1
  # 物品名
  name: §6一件皮革甲
  # 物品Lore
  lore:
  - '简单节点测试: <easyTest>'
  - '快速计算测试: <calcTest>'
  - '文本中小于号请添加反斜杠, 防止错误识别'
  - '形如: \<\<\<\>\>\>'
  - "换行符测试\n换行符测试"
  # 物品附魔
  enchantments:
    ARROW_DAMAGE: 1
    ARROW_KNOCKBACK: 1
  # 物品隐藏标识
  hideflags:
  - HIDE_ATTRIBUTES
  - HIDE_DESTROYS
  # 物品颜色(适用于药水/皮革装备)
  color: 65535
  # 物品NBT
  nbt:
    # 可以在NBT中编辑物品的原版属性
    AttributeModifiers:
    - Amount: 10
      AttributeName: minecraft:generic.max_health
      Operation: 0
      UUID:
      - 0
      - 31453
      - 0
      - 59664
      Name: generic.maxHealth
  # 物品私有节点
```

```yaml
  sections:
    easyTest: 简单节点测试
    calcTest:
      type: fastcalc
      formula: 1+2+3
      min: 1
      max: 100
      fixed: 3

GradientTest:
  material: STONE
  lore:
    - <test1>
    - <test2>
  sections:
    test1: <gradient::000000_FFFFFF_1_---------->
    test2:
      type: gradient
      colorStart: 000000
      colorEnd: FFFFFF
      step: 1
      text: ----------

# 一个测试模板
template1:
  material: IRON_SWORD
  lore:
  - "&e攻击伤害: &f<damage>"
  nbt:
    MMOITEMS_ATTACK_DAMAGE: (Double) <damage>
# 一个测试模板
template2:
  material: DIAMOND_SWORD

# 一个全局继承测试, 它继承了"template1"的所有内容
TemplateItem1:
  inherit: template1
  name: §f物品继承测试
  sections:
    damage: 100
# 一个部分继承测试, 它继承了"template1"的lore, 以及"template2"的material
TemplateItem2:
  inherit:
    lore: template1
```

```
    material: template2
  name: §f物品继承测试
  sections:
    damage: 100
# 一个顺序继承测试, 它将按顺序进行节点继承. 先继承"template1"的所有内容, 再继承"temp
TemplateItem3:
  inherit:
  - template1
  - template2
  name: §f物品继承测试
  sections:
    damage: 100

# join节点测试
JoinTest1:
  material: STONE
  lore:
    # 结果: 1, 2, 3, 4, 5
    - 'join节点: <test>'
  sections:
    test:
      type: join
      # 待操作的列表
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
JoinTest2:
  material: STONE
  lore:
    # 结果: 1-2-3-4-5
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
      # 分隔符(默认为", )
```

```
      separator: "-"
JoinTest3:
  material: STONE
  lore:
    # 结果: <1, 2, 3, 4, 5>
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
      # 前缀
      prefix: "<"
      # 后缀
      postfix: ">"
JoinTest4:
  material: STONE
  lore:
    # 结果: 1, 2, 3
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
      # 限制长度
      limit: 3
JoinTest5:
  material: STONE
  lore:
    # 结果: 1, 2, 3, ...
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
```

```
        - 1
        - 2
        - 3
        - 4
        - 5
      limit: 3
      # 超过长度的部分用该符号代替
      truncated: "..."
JoinTest6:
  material: STONE
  lore:
    # 等同于:
    # - 第一行
    # - 第二行
    # - 第三行
    #
    # 这个节点应该单独占据一行
    # 不要在这行写其他文本(比如'join节点: <test>')
    # 具体请自行测试
    - '<test>'
  sections:
    test:
      type: join
      list:
        - 第一行
        - 第二行
        - 第三行
      # 像下面这样写分隔符、前缀和后缀
      # 即可达到调用多行lore的效果
      separator: "\\n"
      prefix: '"'
      postfix: '"'

RepeatTest1:
  material: STONE
  lore:
    # 等同于:
    # - 文本
    # - 文本
    # - 文本
    # 这个节点应该单独占据一行
    # 不要在这行写其他文本(比如'repeat节点: <test>')
    # 具体请自行测试
    - '<test>'
```

```yaml
  sections:
    test:
      type: repeat
      content: 文本
      repeat: 3
      # 像下面这样写分隔符、前缀和后缀
      # 即可达到调用多行lore的效果
      separator: "\\n"
      prefix: '"'
      postfix: '"'
RepeatTest2:
  material: STONE
  lore:
    # 4行"&4&l<红宝石槽>"
    - '<repeat>'
  sections:
    repeat:
      type: repeat
      content: '&4&l<红宝石槽>'
      repeat: 4
      # 像下面这样写分隔符、前缀和后缀
      # 即可达到调用多行lore的效果
      separator: "\\n"
      prefix: '"'
      postfix: '"'
RepeatTest3:
  material: STONE
  lore:
    # "§4§l<★>-§4§l<★>-§4§l<★>"
    - '<repeat>'
  sections:
    repeat:
      type: repeat
      content: '§4§l<★>'
      repeat: 3
      separator: "-"
RepeatTest4:
  material: STONE
  lore:
    # 形似&4||||||||||&f||||||||||
    - 'repeat节点: &4<repeat1>&f<repeat2>'
  sections:
    repeat1:
      type: repeat
```

```
      content: "|"
      repeat: 10
    repeat2:
      type: repeat
      content: "|"
      repeat: 10
```

Items/RPGExample.yml

```
RPGSwordTemplate:
  material: <材质>
  name: <前缀>大剑
  lore:
  - §e----属性----
  - <属性>
  - §e----宝石----
  - <宝石>
  unbreakable: true
  hideflags:
  - HIDE_ATTRIBUTES
  - HIDE_UNBREAKABLE
  sections:
    材质: IRON_SWORD
    宝石:
      type: repeat
      content: §7可镶嵌 <宝石类型>
      repeat: <宝石数量>
      separator: "\\n"
      prefix: '"'
      postfix: '"'
    宝石类型: §4§l红宝石
    宝石数量: 4

RPGSword1:
  inherit: RPGSwordTemplate
  sections:
    前缀: §a§l霍格沃茨
    属性:
      type: join
      list:
```

```
      - '§f攻击力: 100'
      - '§f暴击率: 10%'
      - '§f暴击伤害: 100%'
      separator: "\\n"
      prefix: '"'
      postfix: '"'
    宝石类型: §b§l蓝宝石
    宝石数量: 2

RPGSword2:
  inherit: RPGSwordTemplate
  sections:
    前缀: §b§l莱因哈特
    属性:
      type: join
      list:
      - '§f攻击力: 200'
      - '§f暴击率: 20%'
      - '§f暴击伤害: 200%'
      separator: "\\n"
      prefix: '"'
      postfix: '"'
    宝石类型: §a§l绿宝石
    宝石数量: 3
```

物品列表

> 全部命令需要OP权限/后台执行, []为必填, ()为选填

list

> /ei list (页码) > 查看所有EI物品

- (页码) 打开对应页的物品列表(默认为1)

物品获取

> 全部命令需要OP权限/后台执行, []为必填, ()为选填

get

> /ei get [物品ID] (数量) > 根据ID获取EI物品

- [物品ID] NI物品ID
- (数量) 获取的数量 (默认为1)

give

> /ei give [玩家ID] [物品ID] (数量) > 根据ID给予EI物品

- [玩家ID] 待给予玩家的ID
- [物品ID] NI物品ID
- (数量) 获取的数量 (默认为1)

giveAll

> /ei giveAll [物品ID] (数量) > 根据ID给予所有人EI物品

- [物品ID] NI物品ID
- (数量) 获取的数量 (默认为1)

物品掉落

> 全部命令需要OP权限/后台执行, []为必填, ()为选填

drop

> /ei drop [物品ID] [数量] [世界名] [X坐标] [Y坐标] [Z坐标] [是否反复随机] [物品解析对象] (指向数据) > 于指定位置掉落EI物品

- [物品ID] NI物品ID
- [数量] 获取的数量，默认为1
- [世界名] 物品掉落世界的名称

- [X坐标] 物品掉落世界的X轴坐标
- [Y坐标] 物品掉落世界的Y轴坐标
- [Z坐标] 物品掉落世界的Z轴坐标

如果你想让MM怪物被玩家击杀后掉落EI物品，你可以直接查看：NI物品掉落(NI物品掉落同时支持NI、MM、EI物品)

物品保存

> 全部命令需要OP权限/后台执行, []为必填, ()为选填

save

> /ei save [物品ID] (保存路径) > 将手中物品以对应ID保存至对应路径

- [物品ID] 保存后的EI物品ID
- (保存路径) 物品存储的文件路径, 默认为 物品ID.yml

  形如 `test.yml` ，将存储于 `plugins/EasyItem/Items/test.yml`

! 如果物品ID重复(已存在对应ID的EI物品)，将保存失败并收到提示。

cover

> /ei cover [物品ID] (保存路径) > 将手中物品以对应ID覆盖至对应路径

- [物品ID] 保存后的EI物品ID
- (保存路径) 物品存储的文件路径, 默认为 物品ID.yml

  形如 `test.yml` ，将存储于 `plugins/EasyItem/Items/test.yml`

! 如果物品ID重复(已存在对应ID的EI物品)，将直接覆盖原物品，强行保存。

mm load

> /ei mm load [物品ID] (保存路径) > 将对应ID的MM物品保存为NI物品

- [物品ID] 待转换的MM物品ID
- (保存路径) 物品存储的文件路径，默认为配置文件中的Main.MMItemsPath

  形如 `test.yml` ，将存储于 `plugins/EasyItem/Items/test.yml`

! 如果物品ID重复(已存在对应ID的EI物品)，将保存失败并收到提示。

mm cover

> /ei mm cover [物品ID] (保存路径) > 将对应ID的MM物品覆盖为NI物品

- [物品ID] 待转换的MM物品ID
- (保存路径) 物品存储的文件路径，默认为配置文件中的Main.MMItemsPath

  形如 `test.yml` ，将存储于 `plugins/EasyItem/Items/test.yml`

! 如果物品ID重复(已存在对应ID的EI物品)，将直接覆盖原物品，强行保存。

mm loadAll

> /ei mm loadAll (保存路径) > 将全部MM物品转化为NI物品

- (保存路径) 物品存储的文件路径，默认为配置文件中的Main.MMItemsPath 形如 `test.yml` ，将存储于 `plugins/EasyItem/Items/test.yml`

! 如果物品ID重复(已存在对应ID的EI物品)，将保存失败并收到提示。

杂项

> 全部命令需要OP权限/后台执行,[]为必填,()为选填

help

> /ei help > 查看帮助信息

reload

> /ei reload > 重新加载EI物品

物品配置

路径

所有物品配置文件应存放于 `plugins/EasyItem/Items` 文件夹

重复 ID 的物品仍然会被加载，但可能互相覆盖

最后哪个物品活下来。。。随缘了属于是

配置

详见默认配置

编写你的物品

/ei save是万物起源

遇事不决，/ei save。如果不行，就/ei cover。这是最简单最便捷的快速生成物品配置的方法

物品保存指令

物品覆盖指令

ID

所有物品都应该有一个ID，如下格式：

```
物品ID:
  # 具体的配置项，以物品材质为例
  material: STONE
```

材质

即，物品是石头还是木头还是钻石剑

```
物品1:
  # 这个物品是石头
  material: STONE
物品2:
  # 这个物品是钻石
  material: DIAMOND
```

ID都有哪些，见下方链接

https://hub.spigotmc.org/javadocs/spigot/org/bukkit/Material.html

如果你看着 ID 不知道它对应什么物品。。。

一般来讲，你可以在游戏中同时按下 F3+H，启用高级显示框，这样物品下方就会出现对应的ID。

如上图所示， `minecraft:stone` 对应 `STONE`

对于 mod 物品，前缀不能省略。

比如一个名称为 `mod:test` 的物品，对应的 ID 应为 `MOD_TEST`

但是啊但是，你有没有看上面啊？

/ei save是万物起源。别搁这儿看ID了，保存一下什么都有了，看个锤子看。

物品名

具体配置如下

```
有名字的铁剑:
  material: IRON_SWORD
  name: 我有名字
```

非法物品名

该配置项在生成物品时将覆盖name项，具体配置如下

```
有名字的铁剑:
  material: IRON_SWORD
  illegalName: '{"italic":false,"color":"white","text":"玉米饼"}'
```

众所周知，高版本的name和lore在nbt里其实是以一段json的形式存在的。
本来这没什么，他这么做，你接受就好了。
但他妈的总有一些傻逼插件能给你整出点活儿。
比如上面那个就是某MMOItems干的，上面那一段非法物品名的合法形式应该是：
{"extra":
[{"bold":false,"italic":false,"underlined":false,"strikethrough":false,"obfuscated":false,"color":"white","text":"玉米饼"}],"text":""}
这导致EasyItem无法以更可读的形式无损保存物品，只能将它原版的傻逼德行记录下来。
但是这无伤大雅，如果你准备修改它，那你也就不需要让他与原先的物品完全一致了。
直接使用name配置项吧。

物品Lore

具体配置如下

```
有Lore的铁剑:
  material: IRON_SWORD
  lore:
  - 我有lore
  - 我真有lore
  - 信我
```

你可以通过换行符 `\n` 换行, 在一行中书写多行lore

值得一提的是, 在yaml语法中, 双引号包裹的 `"\n"` 才代表换行符

单引号包裹的 `'\n'` 只代表一段形似 `\n` 的字符

例:

```
有Lore的铁剑:
  material: IRON_SWORD
  lore:
  - "我有lore\n我真有lore\n信我"
```

非法Lore

该配置项在生成物品时将覆盖lore项，具体配置如下

```
有Lore的铁剑:
  material: IRON_SWORD
  illegalLore:
  - '{"italic":false,"extra":[{"strikethrough":true,"color":"dark_gray","t
    "},{"bold":true,"color":"aqua","text":"通用 "},{"color":"gray","text":"
```

众所周知，高版本的name和lore在nbt里其实是以一段json的形式存在的。
本来这没什么，他这么做，你接受就好了。
但他妈的总有一些傻逼插件能给你整出点活儿。
比如上面那个就是某MMOItems干的，上面那一段非法Lore的合法形式应该是：
{"extra":
[{"bold":false,"italic":false,"underlined":false,"strikethrough":true,"obfuscated":false,"color":"dark_gray","text":"-
-----"},{"italic":false,"strikethrough":false,"color":"gray","text":"[[20:56:46 INFO]: "},
{"bold":true,"italic":false,"color":"aqua","text":"通用 "},
{"bold":false,"italic":false,"color":"gray","text":"]"},
{"italic":false,"strikethrough":true,"color":"dark_gray","text":"------"}],"text":""}
这导致EasyItem无法以更可读的形式无损保存物品，只能将它原版的傻逼德行记录下来。
但是这无伤大雅，如果你准备修改它，那你也就不需要让他与原先的物品完全一致了。
直接使用lore配置项吧。

子ID/损伤值

在 1.12.2 及以下的版本中，某些物品存在“子ID”。

比如 WOOL 是白色羊毛，而子ID为 1 的 WOOL 是橙色羊毛。

对应配置方法如下

```
白色羊毛:
  material: WOOL
橙色羊毛:
  material: WOOL
  # 子ID为1
  damage: 1
```

而对于有耐久的物品，damage对应损伤值，即，物品消耗了几点耐久。

```
铁剑:
  material: IRON_SWORD
用了一下的铁剑:
  material: IRON_SWORD
  # 消耗了1点耐久
  damage: 1
```

CustomModelData

对于 1.14+ 的服务器，物品有了一个新的属性，CustomModelData。

一般人们用它搭配材质包制作自定义材质物品。

对应配置方法如下

```
铁剑:
  material: IRON_SWORD
  # CustomModelData 为 1
  custommodeldata: 1
```

附魔

附魔名称列表，应前往以下链接查看

https://hub.spigotmc.org/javadocs/spigot/org/bukkit/enchantments/Enchantment.html

具体配置方法如下

```
有附魔的铁剑:
  material: IRON_SWORD
  enchantments:
    # 锋利5
    DAMAGE_ALL: 5
```

啥？你说全是英文你根本看不懂哪个对哪个？

/ei save干什么用的

无法破坏

具体配置如下

```
无法破坏的铁剑:
  material: IRON_SWORD
  unbreakable: true
```

隐藏属性

有的物品明明无法破坏，物品信息里却看不到。

有的物品明明有附魔，物品信息里却看不到。

具体配置方法如下

```
啥都看不到的铁剑:
  material: IRON_SWORD
  hideflags:
  # 隐藏物品属性
```

```
  - HIDE_ATTRIBUTES
  # 隐藏物品可破坏方块
  - HIDE_DESTROYS
  # 隐藏物品染料颜色
  - HIDE_DYE
  # 隐藏物品附魔
  - HIDE_ENCHANTS
  # 隐藏物品可放置方块
  - HIDE_PLACED_ON
  # 隐藏物品药水效果
  - HIDE_POTION_EFFECTS
  # 隐藏物品无法破坏
  - HIDE_UNBREAKABLE
```

物品颜色

药水和皮革护甲可以拥有自定义颜色，具体配置方法如下

```
有颜色的皮革头盔1:
  material: LEATHER_HELMET
  color: 'ABCDEF'
有颜色的皮革头盔2:
  material: LEATHER_HELMET
  color: 666666
```

如上所示，你可以用十进制和十六进制两种方式配置物品颜色。

如果你想要以十进制表示颜色，那么color必须配置一个数字（不被引号包裹）

如果你想要以十六进制表示颜色，那么color必须是一个字符串（被引号包裹）

比如， `color: '666666'` 表示的是十六进制，等价于 `color: 6710886`

自定义NBT

许多插件会向物品中插入一些自定义NBT，用来记录某些信息。

NeigeItems也允许你这样做。

你可以通过插入自定义NBT，兼容一些基于NBT的插件，比如

```
超猛镐子:
  material: IRON_PICKAXE
  nbt:
    MMOITEMS_ATTACK_DAMAGE: (Double) 1000000
```

如果你装了MMOItems，那这个镐子现在应该有100万攻击力了。

你可能注意到，1000000前面有一个 `(Double)` 。

这个前缀代表，生成这条NBT的时候，会以 Double 类型生成（写的时候不要忘记括号后面的空格）。

如果你不写的话，生成时这条NBT很有可能就变成了Int类型或者Long类型。

这种用于转换类型的前缀应该应用于数值类型的NBT

具体有以下类型可以选择

```
# Byte 类型的 1
(Byte) 1
# Short 类型的 1
(Short) 1
# Int 类型的 1
(Int) 1
# Long 类型的 1
(Long) 1
# Float 类型的 1
(Float) 1
# Double 类型的 1
(Double) 1
```

使用类型转换前缀，一定要加空格

但是啊但是，别搁这儿看了，你直接/ei save一下，自动就都出来了。

模板继承

你可以让一个配置继承其他配置的部分或全部内容

具体内容请查看模板继承

节点

私有节点应直接配置与物品下方，比如

```
测试铁剑:
  material: IRON_SWORD
  name: <test>
  sections:
    test: 测试内容
```

有关私有节点的各个类型，具体请查看私有节点

模板继承

配置

以默认指令配置为例

```
# 一个测试模板
template1:
  material: IRON_SWORD
  lore:
  - "&e攻击伤害: &f<damage>"
  nbt:
    MMOITEMS_ATTACK_DAMAGE: (Double) <damage>
# 一个测试模板
template2:
  material: DIAMOND_SWORD

# 一个全局继承测试, 它继承了"template1"的所有内容
templateItem1:
  inherit: template1
  name: §f物品继承测试
  sections:
    damage: 100
# 一个部分继承测试, 它继承了"template1"的lore, 以及"template2"的material
templateItem2:
  inherit:
    lore: template1
```

```
    material: template2
  name: §f物品继承测试
  sections:
    damage: 100
# 一个顺序继承测试，它将按顺序进行节点继承. 先继承"template1"的所有内容，再继承"temp
templateItem3:
  inherit:
  - template1
  - template2
  name: §f物品继承测试
  sections:
    damage: 100
```

可以看到，我们可以通过在物品配置中添加"inherit"来继承其他物品的配置。

```
inherit: template1
```

代表这个物品将继承"template1"的全部内容

```
inherit:
  lore: template1
  material: template2
```

代表这个物品将继承"template1"的"lore"配置项，以及"template2"的"material"配置项

```
inherit:
- template1
- template2
```

代表这个物品将先继承"template1"的所有配置项，再继承"template2"的所有配置项。

因此对于重复的项，后者会对前者进行覆盖。

私有节点

> 节点配置内全面支持节点调用

私有节点配置

查看：私有节点配置，形如

```
测试铁剑:
  material: IRON_SWORD
  name: <test>
  sections:
    test: 测试内容
```

快速计算节点

```
节点ID:
  type: fastcalc
  formula: 1+2+3
  min: 1
  max: 100
  fixed: 3
```

- `formula` 待计算公式，支持代入节点
- `min` 结果的最小值
- `max` 结果的最大值
- `fixed` 小数保留位数

Join节点

```
节点ID:
  type: join
  list:
    - 第一行
    - 第二行
    - 第三行
    - 第四行
  separator: "-"
  prefix: '<'
```

```
  postfix: '>'
  limit: 3
  truncated: "..."
```

简介: 将list中的多段文本连接成一段文本

- `list` 待操作的列表
- `separator` 分隔符 (默认为", ")
- `prefix` 前缀 (默认无前缀)
- `postfix` 后缀 (默认无后缀)
- `limit` 限制列表长度
- `truncated` 超过长度的部分用该符号代替 (默认直接吞掉超过长度的部分)

示例中的节点将返回:

```
<第一行-第二行-第三行-...>
```

由于该节点功能较其他节点更加复杂, 因此我为它编写了多个示例配置帮助理解, 如下:

```
# 帮助理解list
JoinTest1:
  material: STONE
  lore:
    # 结果: 1, 2, 3, 4, 5
    - 'join节点: <test>'
  sections:
    test:
      type: join
      # 待操作的列表
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
# 帮助理解separator
JoinTest2:
  material: STONE
  lore:
    # 结果: 1-2-3-4-5
```

```yaml
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
      # 分隔符(默认为", )
      separator: "-"
# 帮助理解prefix及postfix
JoinTest3:
  material: STONE
  lore:
    # 结果: <1, 2, 3, 4, 5>
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
      # 前缀
      prefix: "<"
      # 后缀
      postfix: ">"
# 帮助理解limit
JoinTest4:
  material: STONE
  lore:
    # 结果: 1, 2, 3
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
```

```
        - 4
        - 5
      # 限制长度
      limit: 3
# 帮助理解truncated
JoinTest5:
  material: STONE
  lore:
    # 结果: 1, 2, 3, ...
    - 'join节点: <test>'
  sections:
    test:
      type: join
      list:
        - 1
        - 2
        - 3
        - 4
        - 5
      limit: 3
      # 超过长度的部分用该符号代替
      truncated: "..."
# 利用join节点插入多行lore
JoinTest6:
  material: STONE
  lore:
    # 等同于:
    # - 第一行
    # - 第二行
    # - 第三行
    #
    # 这个节点应该单独占据一行
    # 不要在这行写其他文本(比如'join节点: <test>')
    # 具体请自行测试
    - '<test>'
  sections:
    test:
      type: join
      list:
        - 第一行
        - 第二行
        - 第三行
      # 像下面这样写分隔符、前缀和后缀
      # 即可达到调用多行lore的效果
```

```
    separator: "\\n"
    prefix: '"'
    postfix: '"'
```

Repeat节点

```
节点ID:
  type: repeat
  content: '待重复文本'
  separator: "-"
  prefix: '<'
  postfix: '>'
  repeat: 3
```

简介: 将content的文本重复多次, 生成一整段文本

- `content` 待重复文本
- `separator` 分隔符 (默认无分隔符)
- `prefix` 前缀 (默认无前缀)
- `postfix` 后缀 (默认无后缀)
- `repeat` 重复次数

示例中的节点将返回:

```
<待重复文本-待重复文本-待重复文本>
```

由于该节点功能较其他节点更加复杂, 因此我为它编写了多个示例配置帮助理解, 如下:

```
RepeatTest1:
  material: STONE
  lore:
    # 等同于:
    # - 文本
    # - 文本
    # - 文本
    # 这个节点应该单独占据一行
```

```
    # 不要在这行写其他文本(比如'repeat节点: <test>')
    # 具体请自行测试
    - '<test>'
  sections:
    test:
      type: repeat
      content: 文本
      repeat: 3
      # 像下面这样写分隔符、前缀和后缀
      # 即可达到调用多行lore的效果
      separator: "\\n"
      prefix: '"'
      postfix: '"'
RepeatTest2:
  material: STONE
  lore:
  # 4行"&4&l<红宝石槽>"
    - '<repeat>'
  sections:
    repeat:
      type: repeat
      content: '&4&l<红宝石槽>'
      repeat: 4
      # 像下面这样写分隔符、前缀和后缀
      # 即可达到调用多行lore的效果
      separator: "\\n"
      prefix: '"'
      postfix: '"'
RepeatTest3:
  material: STONE
  lore:
  # "§4§l<★>-§4§l<★>-§4§l<★>"
    - '<repeat>'
  sections:
    repeat:
      type: repeat
      content: '§4§l<★>'
      repeat: 3
      separator: "-"
RepeatTest4:
  material: STONE
  lore:
    # 形似&4||||||||||&f||||||||||
    - 'repeat节点: &4<repeat1>&f<repeat2>'
```

```
  sections:
    repeat1:
      type: repeat
      content: "|"
      repeat: 10
    repeat2:
      type: repeat
      content: "|"
      repeat: 10
```

渐变色节点

```
节点ID:
  type: gradient
  colorStart: "000000"
  colorEnd: "FFFFFF"
  step: 1
  text: 哈哈哈哈哈哈哈哈哈哈哈哈哈哈
```

- `colorStart` 起始颜色
- `colorEnd` 结尾颜色
- `step` 每几个字符变一次颜色(默认为1, 可省略)
- `text` 文本内容

简单节点

```
节点ID: 值
```

如上所示，你直接添加节点的值。

比如：

```
test: test
```

调用 `<test>` 将返回 `test`

即时声明节点

> 节点配置内全面支持节点调用

格式

`<节点类型::参数1_参数2_参数3...>`

即时声明节点无法指定节点ID, 如有需求，请配置私有/全局节点
即时声明节点中的 `_` 请用 `\_` 代替, 避免被当做参数分隔符

快速计算节点

```
<fastcalc::1+1+3_2_5_100>
```

- `参数1` 计算公式
- `参数2` 保留小数位数
- `参数3` 公式结果最小值
- `参数4` 公式结果最大值

渐变色节点

```
<gradient::000000_FFFFFF_1_哈哈哈哈哈哈哈哈哈哈哈哈哈哈>
```

- `参数1` 起始颜色
- `参数2` 结尾颜色
- `参数3` 每几个字符变一次颜色
- `参数4` 文本内容

16进制颜色

<#FFFFFF>

如上所示

节点调用

节点可以在任意位置通过<节点ID>的形式调用

> ! 物品配置中出现的起装饰作用的<和>应替换为\<和\>，避免错误识别

高级应用

直接展示例子:

```
test:
  A: test1
  B: test2
```

如上配置节点后

调用 `<test.A>` 将返回 `test1`

调用 `<test.B>` 将返回 `test2`

MythicMobs

EI所有MM适配均基于NI, 请查看**NeigeItems-Wiki**

EI物品掉落 **EI物品穿戴** **EI穿戴物品掉落**