

# Table of Contents

前言	1.1
教程	
📖 入门指南	2.1
🔗❤️🧠 用标签制作好感度系统	2.2
🕒 日常任务	2.3
⌚ 自定义	2.4
💬 对话	2.5
文档（未完成）	
文档	3.1
📂 类型	3.2
✍️ 目标	3.2.1
📄 条件	3.2.2
🔗 事件	3.2.3
🔧 触发器	3.2.4
★ 变量	3.2.5
💖 兼容	3.3
🔗 BetonQuest	3.3.1
🔒 权限	3.4
📄 占位符	3.5
🔄 从v3及以下版本升级	3.6
💬 对话系统	3.7
❓ 常见问题	3.8
😊 API	3.9
📖 API使用方法	3.9.1
📖 API示例项目教程	3.9.2

## 前言

此Wiki为NotQuests的中文Wiki，包含原文档中的教程部分和文档部分。

翻译：Aikini

原帖：<https://www.notquests.com/>

MCBBS搬运帖：<https://www.mcbbs.net/thread-1356502-1-1.html>

powered by GitbookFile Modify: 2023-09-20 17:58:15

## 👉 入门指南

[!WARNING]style:flat|label:写在前面] 本教程是根据插件5.17.1及以上版本、Paper核心1.20.1版本进行撰写的。

如果你运行的是旧版本或Spigot服务器，可能会遇到功能缺失、命令与教程不符的情况。本教程不提供相应的指导，请自行研究。

本WIKI中插入的图片均有翻译，点击即会显示。

[!NOTE]style:flat|label:文档不完整] 我知道这个文档并不完善，其内容仅为NotQuests实际功能的10%。这不怪我，这份文档可是开源的，所以要怪就怪你，或者说你们这个玩家社区，怪懒的！可耻！

要是你对NotQuests有了更多的了解，你就可以在这里[参与文档的撰写](#)，成为贡献者。（需要Github账号，文档可以直接在这里进行编辑）- 欢迎你的参与！

让我们赶紧来探索NotQuests这个插件吧！有两种入门方式供君选择：

- 要么读完这篇教程
- 要么看我们的[视频](#)（油管链接，别急，我还没做）。注意，这个视频教程里的NotQuests是v3版本，有些命令可能不一样。

## 结构

在NotQuests里，一项任务是由各种不同的的属性构成的。例如：

- **displayName**（显示名称）：玩家所能看见的任务名称。
- **description**（描述）：任务描述。
- **limits**（限制）：该任务玩家能接取、完成或失利的最大次数。
- 等等.....

除此之外，你还可以将下面这些附加进任务中：

- **Objectives**（目标）：设定一些“玩家需要去做的事情”作为目标。一旦这些事情全部做完，任务就会完成。
- **Requirements**（前置）：判定玩家是否可以接取任务。如果玩家未满足任务所需的前置条件，就无法接取该任务。
- **Rewards**（奖励）：奖励一词应该不难理解吧！奖励是在玩家完成任务后会对玩家“执行”的一系列操作。
- **Triggers**（触发器）：这项属性叫做触发器，理解起来可能有点困难。说白了，就是在某些事情发生时“执行”一个事件。

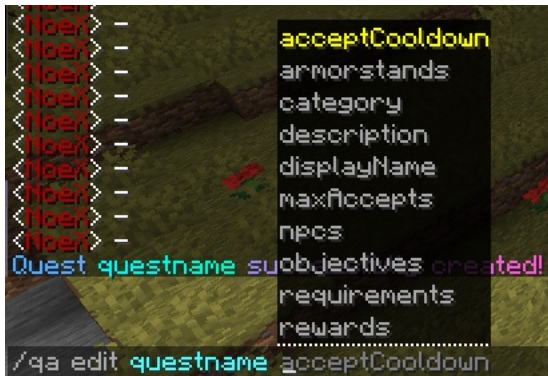
## 命令

插件一共有2种类型的命令：

- `/q` 或 `/notquests`：这个类型的命令是玩家命令。所有玩家都可以使用该命令。
  - `/qa` 或 `/notquestsadmin`：这个类型的命令是管理员命令。我们需要用这个类型的命令来创建和编辑任务。
- 在NotQuests中，无论创建什么任务，只需在游戏中执行各种命令即可完成，而不需要去查改配置文件。

## 完善命令

在你输入命令后，需要用空格进行分割，而分割后输入的这部分内容我们称之为参数。用下图作为举例，**edit**和**questname**即为参数，这条命令中，共有2个参数：



如果你在最后一个参数后再按一下空格键，我们的智能命令系统会自动为你显示“后续”可用（可补全）的参数。如果你输入的命令无法执行，不妨试试在后面加一个空格，看看后面是否还有需要填写的参数，然后，再看一遍完善命令这部分内容！

[!TIP]style:flat|label:仍有问题?] 如果无法自动填充参数，后面没有显示后续可用参数，只需按下回车执行现有的命令就好！之后你就会看到一个帮助菜单。这个菜单会为你提供帮助，它会告诉你可用参数及其用途。

## 创建我们的第一个任务

想要学会一件事，我觉得最好最简单的方式就是：边学边做！所以，别在这无聊的文档上浪费时间了，我们来创建我们的第一个任务吧：

`/qa create TheVirus`

执行该命令后会出现下面一则消息：

**Quest TheVirus successfully created!**

## 1.任务描述&显示名称

现在，我们成功创建了一个名叫TheVirus的任务，TheVirus即是这个任务的任务名称，任务名称中不能有任何空格，它只是作为一个任务的识别符。不过，我们可用给它设定一个显示名称，显示名称顾名思义，就是玩家实际在游戏中能够看到的名称，这个名称中可用使用空格。在这里，我们为的任务设定一个显示名称**A Deadly Virus**（一个死亡病毒）。

```
/qa edit TheVirus displayName set A Deadly Virus
```

接下来，我们想要为该任务添加一个描述，这个描述会在很多地方显示出来，例如，玩家正打算尝试预览任务或是接受任务时显示。在这里，我们为的任务添加一个描述：一个死亡病毒感染了临冬城的居民。你必须物理净化掉这些被感染的村民，从而阻止病毒的继续传播。

```
/qa edit TheVirus description set A deadly virus has infected the people of Winterfell. You have to murder the infected villagers to prevent the virus from spreading further.
```

至此，你的玩家接取任务后就会看到这样一个漂亮的描述和显示名称：



## 2.前置

在没有设定任何前置的情况下，每个玩家都可以接受你的任务。不过，这个任务相当棘手！所以我们得要求玩家至少获得10点任务点数，才能接受这个任务：

```
/qa edit TheVirus requirements add QuestPoints moreOrEqualThan 10
```

任务点数可以通过完成任务获得，或者手动发放。

如果玩家目前不满足前置条件，接取任务时会看到这个提示。

Quest TheVirus successfully created!

为了进一步测试我们的任务，先给自己发放10点任务点数吧：

```
/qa questpoints putyourminecraftnamehere add 10
```


[!TIP!label:感谢你的关注] 如果你喜欢NotQuests插件，请在Modrinth上关注NotQuests☺这样能激励我为您带来更多更新和功能！

## 3.目标

目标是每个任务的核心组成部分。那我们来添加第一个目标吧！

首先，玩家需要清理被僵尸堵塞的道路：

```
/qa edit TheVirus objectives add BreakBlocks dirt 64
```

当完成破坏64个泥土这一目标时即该目标  如果你想，你还可以在这里指定多种方块。

比如，如果你想将玩家破坏的方块指定为泥土或石头，就需要输入 dirt,stone。

现在，我们给我们的目标添加一条描述：被感染的僵尸在街上拉屎了！你需要通过破坏64个泥土才能清理干净！

```
/qa edit TheVirus objectives edit 1 description set The infected Zombies shat on the street. Clean it up by breaking 64 dirt blocks!
```

再设定目标的显示名称：臭街

```
/qa edit TheVirus objectives edit 1 displayName set Stinky Street
```

然后就可以用命令 /q take TheVirus 接取我们的任务了！



这些都很简单对吧？

### 第二目标

接下来就是我们的下一个任务目标！街道已经清理干净了！该物理净化一下这些被感染的村民了：

```
/qa edit TheVirus objectives add KillMobs zombie_villager 15
```

在击杀15个僵尸村民后，该目标就会完成！那现在我们来给这个目标加上描述吧：如你所见，你的面前都是被感染的村民！为了阻止病毒的传播，干掉他们！

```
/qa edit TheVirus objectives edit 2 description set You can see the infected villagers in front of you! Murder them all to stop the virus from spreading!
```

然后还有显示名称：僵尸突围！

```
/qa edit TheVirus objectives edit 2 displayName set Zombies ahead!
```

### 目标依赖

现在，在接受这个任务后，你将看到以下内容：

```
Objectives:
1. Stinky Street:
  L_ Description: The infected Zombies shat on the street.
  Clean it up by breaking 64 dirt blocks
  L_ Break Blocks: Dirt
  L_ Progress: 0 / 64
2. Zombies ahead!:
  L_ Description: You can see the infected villagers in front
  of you! Murder them all to stop the virus from spreading!
  L_ Kill mobs: zombie villager
  L_ Progress: 0 / 15
Quest description: A deadly virus has infected the people of
Winterfell. You have to murder the infected villagers to
prevent the virus from spreading further.

[Quest Accepted]
A Deadly Virus
```

如你所见，当前这个任务的两个目标都是可见的。可以按照任意顺序完成任务目标。

不过，我们想让第二目标“僵尸突围！”在完成第一目标后才变得可见、可完成，换言之，就是预置目标进程的完成顺序。

为了实现这一想法，我们需要给第二目标添加一个条件，让玩家先去完成第一目标才能解锁第二目标。在NotQuests中要想实现的话有两种方法（任选其一）：

- 简单一点的方法： `/qa edit TheVirus objectives predefinedProgressOrder set firstToLast`。根据目标添加的前后顺序自动为任务设定目标顺序，后续添加的任务也会继续这样排序。
- 难一点的方法： `/qa edit TheVirus objectives edit 2 conditions unlock add CompletedObjective 1`。设置解锁目标2的条件为完成目标1。如果有多个目标，你就需要挨个给它们设置。不过，这种方法有很高的灵活性。

完成！现在要是你刚接受这个任务，你就会看见第二目标是隐藏着的。

```
Objectives:
1. Stinky Street:
  L_ Description: The infected Zombies shat on the street.
  Clean it up by breaking 64 dirt blocks
  L_ Break Blocks: Dirt
  L_ Progress: 0 / 64
2. [HIDDEN]
Quest description: A deadly virus has infected the people of
Winterfell. You have to murder the infected villagers to
prevent the virus from spreading further.

[Quest Accepted]
A Deadly Virus
```

第二目标会在你完成第一目标后解锁。☺

## 4. 触发器

通过触发器，我们可以给任务添加一些事件。

当玩家做到第二个目标时，他需要去击杀一定数量的僵尸村民，然后他就会发现一件事，那就是他没有僵尸村民可杀。那咋办？

为此，我们就让NotQuests在玩家完成第一目标后为他生成一些僵尸村民以供击杀。

首先，我们需要创建生成僵尸村民的事件。事件在创建后，可以在该任务外的其它任意任务中重复使用。我们来创建一个SpawnMob的事件，将事件命名为 Spawn15ZombieVillagers：

```
/qa actions add Spawn15ZombieVillagers SpawnMob zombie_villager 15 PlayerLocation
```

这个事件会在接取任务的玩家当前所在的位置生成15个僵尸村民☺

现在，我们把这个事件添加进我们任务的触发器里：

```
/qa edit TheVirus triggers add Spawn15ZombieVillagers BEGIN --applyon O2 --world_name ALL
```

一旦我们开始（BEGIN）在所有世界（不限定世界）里进行第二目标（O2，O是Objective的缩写）时，就会运行“Spawn15ZombieVillagers”事件。请随意测试 - 会起效的。🔧



## 5.奖励

要是你的玩家完成这么艰难的任务后你都不给他一点奖励，那他就既浪费了时间又没有收获，他一定会恨你的。所以，我们来加一点奖励吧：

- +2任务点数： `/qa edit TheVirus rewards add QuestPoints add 2`
- 2把剑： `/qa edit TheVirus rewards add GiveItem hand 2`
- - 上面这个命令在输入时，由于你想设定奖励为剑，所以你手上必须手持剑。除此之外，你也可以用这个命令： `/qa edit TheVirus rewards add GiveItem wooden_sword 2`
- +300金钱（该功能必须安装Vault）： `/qa edit TheVirus rewards add Money add 300`
- 想通过命令给予玩家来自一些其它插件的奖励吗？你可以用{PLAYER}占位符表示玩家。举例： `/qa edit TheVirus rewards add ConsoleCommand cr give to {PLAYER} DailyCrate 2`

### 奖励显示名称

玩家在完成任务后，会获得奖励，然而，这并不会通知玩家。这是因为在默认情况下奖励是隐藏着的（可以在config中修改），除非你给它们添加一个显示名称！所以，我们来添加吧：

1. `/qa edit TheVirus rewards edit 1 displayName set +2 Quest Points`
2. `/qa edit TheVirus rewards edit 2 displayName set +2 handcrafted Wooden Swords`
3. `/qa edit TheVirus rewards edit 3 displayName set +300 Coins`

完成！这下玩家在完成任务后，就会看见任务的奖励：



## 6.更多任务设定

我们不想让玩家在完成任务后一遍又一遍地接受该任务。我们就可以对其进行限制。

首先，我们可以设置，玩家在完成任务后需要等待20个小时后才能再次接受该任务：

`/qa edit TheVirus acceptCooldown complete set 20h`

20h=20小时，现在，我们给任务再添加一个最多只能完成10次的硬性限制。在第10次完成任务后，玩家无论等待多久都不能再次接受该任务：

`/qa edit TheVirus limits completions 10`

要是你想限制玩家的失败次数，当玩家失败或放弃了3次以上任务的话，就不给他接受任务的话，简单：

`/qa edit TheVirus limits fails 3`

除此之外，你还可以在他第一次试图发起任务时阻止他：

`in the first place`

最后，我们把takeEnabled设定为false：

`/qa edit TheVirus takeEnabled false`

这样会使玩家无法通过 `/q take TheVirus` 命令接受任务。因此，玩家们必须通过右击任务派发NPC或者任务派发盔甲架上。详细内容会在下一节讲述：

## 7.NPC交互式接取任务

使用 `/q take TheVirus` 命令可以接受任务。不过你还可以使用 `/qa edit TheVirus npcs add [NPC ID]` 或者 `/qa edit TheVirus armorstands add` 命令将接受任务的操作绑定在某个CitizensNPC或者盔甲架上。

任务文件将会储存在 `plugins/NotQuests/default/quests.yml` 以及 `plugins/NotQuests/default/actions.yml` 文件中。



## 高级概念

### 类别

你可以根据类别将任务进行分组。类别只是一种将任务划分到一起的方式。我们来创建一个名叫“Virus Quests”（病毒任务）的类别吧：

```
/qa categories create VirusQuests
```

现在将我们的任务移动到这个类别（默认情况下，新建任务都会被归类至“default”类别）。

```
/qa edit TheVirus category set VirusQuests
```

完成 - 就是这么简单！一个任务只能属于一个类别。值得一提的是，类别这个功能决定着NotQuests的文件夹结构 - 甚至还决定着GUI中的布局！

### 子类别

每个类别都可以拥有子类别，并且子类别没有限制，你可以根据自己的需要随意套娃。下面这个是为“VirusQuests”（病毒任务）类别创建一个名为“Zombies”的子类别的示例：

```
/qa categories create VirusQuests.Zombies
```

### 类别显示名称

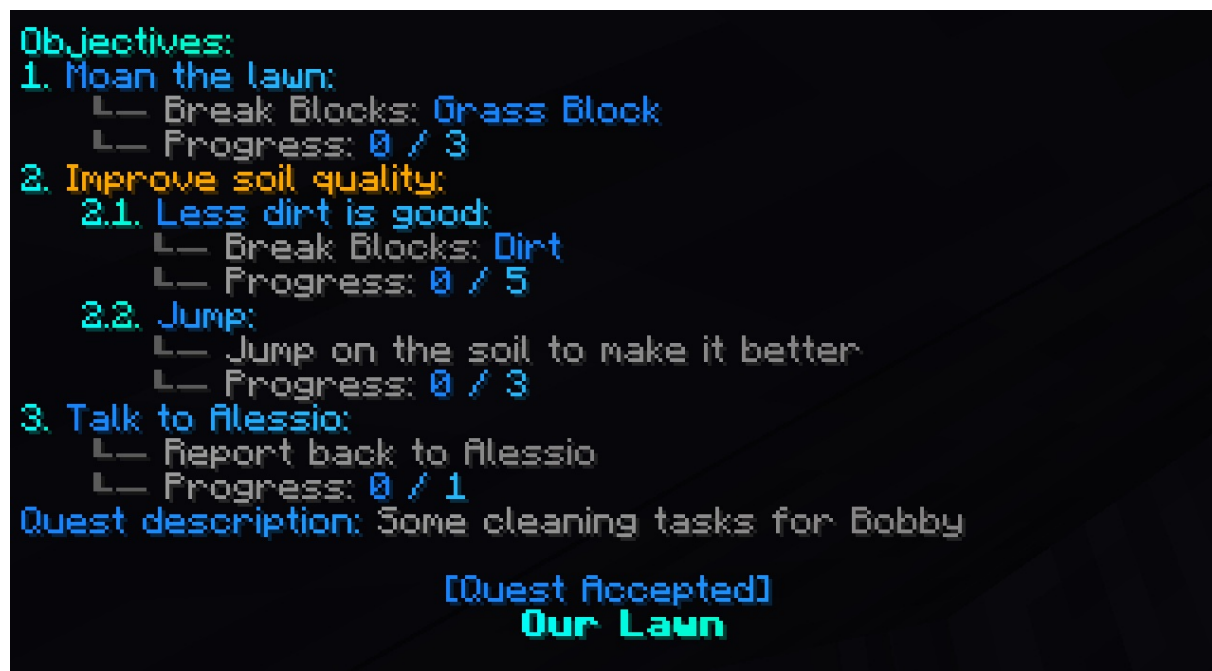
我上面创建时所输入的，只是作为类别的一种标识符。你可以添加一个显示名称（玩家实际在游戏中会看到的）。显示名称可以包含空格，甚至颜色代码，就跟前面所说的任务名称一样！下面是一个示例：

```
/qa categories edit VirusQuests displayName set <dark_green>Virus Quests <main>(dangerous)
```

### 预置类别的进程顺序

（待实现，但我仍要写这部分内容。这个功能类似于前文任务>目标依赖中所说的预置进程完成顺序，不过这项功能是应用于类别>任务。非常地有用，可以极大地简化你的工作，提升你的工作速度。）

### 子目标



NotQuests支持子目标，每个目标都可以有无数个子目标。而每个子目标也可以拥有无数个子子目标.....以此类推！

要创建子目标的话，你必须要先创建一个“Objective”目标作为它的父目标。在上面那个截图中，我们可以看到，子目标2.1和2.2归属于“Objective”目标2。

如果你为“Objective”目标添加了子目标，那么当所有子目标完成后，该“Objective”目标就会被标记为完成。但如果你给其它类型的目标添加了子目标的话，那么你需要先完成子目标，才能继续进行父目标（即，子目标完成后，父目标不会被标记为完成，你需要在完成子目标后再去完成父目标的目标要求）。

举例：

1. /qa edit quest objectives add Objective "gold:Improve soil quality"
2. /qa edit quest objectives edit 1 objectives add BreakBlocks dirt 5
3. /qa edit quest objectives edit 1 objectives edit 1 displayName set Less dirt is good
4. /qa edit quest objectives edit 1 objectives add Jump 3.0 --taskDescription "Jump on the soil to make it better"

### 目标条件类型

在前面的教程部分，我们讲述了如何通过条件来设定目标依赖。命令是： /qa edit TheVirus objectives edit 2 conditions unlock add CompletedObjective 。

不过，你可以给目标添加不同类型的条件。这里说到的是解锁条件（unlock），用于设定目标是否隐藏，隐藏的情况下，不会计入进度，也无法完成该目标。

此外，你还可以添加目标条件用来检查你是否能够推进目标进度或完成目标，这一点也很有用！

举例：

- `/qa edit 任务名称 objectives edit 1 conditions unlock add Flying equals false` - 解锁条件。如果你处于飞行状态，目标将会保持锁定/“隐藏”状态。
- `/qa edit 任务名称 objectives edit 1 conditions progress add Flying equals false` - 目标将一直处于显示状态，但当你处于飞行状态时，你无法推进目标进度。
- `/qa edit 任务名称 objectives edit 1 conditions complete add Flying equals false` - 目标将一直处于显示状态，你也可以推进目标进度，但是如果你不满足条件（根据命令可知，条件是不处于飞行状态），即便达成进度，目标也不会完成。

## 目标变量

**NotQuests**还会提供一些变量，让你可以把一些意想不到的目标添加进任务里！举个例子，如果你想添加一个“游玩100分钟”（**PlaytimeMinutes**）的目标：

```
/qa edit 任务名称 objectives add NumberVariable PlaytimeMinutes moreOrEqualThan 100
```

那要是玩家已经玩了100分钟了该怎么办呢？那就这样设定，让玩家再玩100分钟 - 也就是说，从接受任务/解锁目标的那一刻才开始计时：

```
/qa edit 任务名称 objectives add NumberVariable PlaytimeMinutes moreOrEqualThan PlaytimeMinutes+100
```

此处 `PlaytimeMinutes+100` 的意思是要求玩家再游玩100分钟。在**NotQuests**中这是可以实现的，当然，你还能在很多地方用上这样的表达式（如数学、其它变量等）。

发挥一下你的想象力，这么想象一下，我们是否可以根据玩家已完成的任务量/玩家的实力/玩家的NPC好感度/玩家所持有的任务点数/玩家赚取的金钱数，去设置一个动态的目标？这并不是什么难事，在**NotQuests**中这样的事情轻而易举。

## 高级方块/材料选择器

在为**BreakBlocks**目标设定相应目标方块时，或是在为事件/奖励设定**GiveItem**的相应方块/材料时，你会用到这项功能。我们为此设计了一套方块选择工具，你可以随意指定你所需的方块/材料。并且，在**NotQuests**中，你不仅可以指定一种方块/材料，你还可以指定多种方块/材料！这里举个例子：

```
/qa edit 任务名称 objectives add BreakBlocks diamond_ore,deepslate_diamond_ore 20
```

没错！这样设定的话，无论是破坏钻石矿石，还是破坏深层钻石矿石，都会计入进度。

再来个例子：

```
/qa edit 任务名称 objectives add PlaceBlocks hand,acacia_log,spruce_log,birch_log,dark_oak_log 15
```

你会发现这项功能有用，对吧？

这项功能甚至可以用在**GiveItem**事件中，这样一来，你就可以一次给玩家发放多个物品！

## NotQuests表达式非常强大！

如果你想，**NotQuests**甚至还有能力处理更为复杂的条件。现在你还可以在表达式中使用布尔比较。注意：条件可以用于任务需求/目标条件，它们是一样的。

举几个实际的例子：

- `/qa conditions add cname True equals (Money>10)&Flying` 这个命令会检查你是否在飞行，同时检查你的当前财产是否大于10。
- - 还有其它方法同样能做到：`/qa conditions add cname True equals Condition(Conditions:Flying&IsRich)`
- - 又或者：`/qa conditions add cname Condition equals Flying&IsRich`

在表达式内，你还可以在条件里使用 `|`（或者）操作符或者 `!`（否定）。

再举个例子：

```
/qa conditions add moneyCondition Money moreThan 10+TagInteger(TagName:reputation)*TagInteger(TagName:level)
```

这个条件要求你拥有一定数量的金钱，金钱数=10+“好感度（**Reputation**）”标签值\*“等级（**Level**）”标签值。

这还有一个更为复杂的：

```
/qa actions add pp3 Money add ((TagInteger(TagName:points)>=4)*(10+30))+(!TagInteger(TagName:points)>=4)*5)
```

如果你的“点数（**Points**）”标签（整数标签）大于等于4，这个事件将会被激活，然后给你40块钱。如果没有达到这个要求，将会给你5块钱。如果你想了解标签系统的更多内容，请查阅[标签系统指南](#)。

[操作符大全](#)。

诸如此类的**NotQuests**表达式可以在很多地方使用 - 非常强大！

## 个人资料

在**NotQuests**中，每个拥有“**notquests.user.profiles**”权限节点的玩家都会拥有一份个人资料！💎 每份个人资料都会记录他们的任务点数、标签状况以及完成的任务等信息。所以，如果玩家需要的话，这项功能可以让他们重头再来，以选择不同的路径，从而尝试竞速玩法，亦或者仅仅是为了重新体验RPG游戏的乐趣 - 再或者为了其它目的！

显示你当前的个人资料：`/notquests profiles show`

创建一份新的个人资料：`/notquests profiles create 个人资料名称`

修改你的个人资料：`/notquests profiles change 个人资料名称`

## 接下来？

至此，你已经学会了如何创建任务了！开始上手吧！你可以在文档部分获取到关于**NotQuests**的更多信息。如果你需要任何帮助，欢迎你加入我们的[Discord](#)。

powered by [Gitbook](#)File Modify: 2023-09-27 16:00:32



## 👤❤️👤 用标签制作好感度系统



本篇教程将教会你如何使用标签系统制作一个好感度系统 - 用权力的游戏作例子 ☺  
让我们来创建我们的好感度系统吧

### 构思

在你完成任务的时候，该NPC对你的好感度会提升 - 不仅如此，该NPC所属的“家族”对你的好感度也会随之提升。在你获得一定好感度时，你将会解锁更多来自他们家族的任务，获得更好的奖励。家族里的NPC还会邀请你进入他们的家，届时，你将会获得额外秘密地点的访问权限。

### 如何实现

这个构思大体上都可以通过NotQuests插件中强大的标签功能实现，我们可以很巧妙的运用起事件、条件和对话系统，将它们结合起来就可以实现这个构思。

### 标签系统

NotQuests可以保存每个玩家的标签系统数据，就是这样，听起来很简单，而且非常有用，可以保存的玩家数据包括整数（=数字）、单精度浮点数/双精度浮点数（=带小数的数字）、字符串（=文本）和布尔值（=true或false）。

你可以通过事件功能，修改玩家的标签，你也可以通过条件功能，检查玩家的标签。甚至，你还可以通过数学计算这些标签，或者将这些标签与其它变量和标签比较。

[!TIP|label:标签类型] 由于下面标签类型需要输入英文，所以这里给出几个标签类型的英文： 整数（Integer）、单精度浮点数（Float）、双精度浮点数（Double）、字符串（String）和布尔值（Boolean）。

### 创建我们的第一个标签



我们这有几个现成的NPC：Jon Snow（ID=0）、Arya Stark（ID=1）和Cersei Lannister（ID=2），Jon Snow和Arya Stark都归属于家族Stark。Cersei Lannister归属于家族Lannister。

首先，我们来创建一个叫做JonSnowReputation。这个标签将会记录Jon Snow这位NPC对玩家的好感度。用于记录好感度应该使用整数，这个标签应该属于整数（Integer）标签：

```
/qa tags create Integer JonSnowReputation
```

然后再照着这个逻辑给其它NPC，以及它们的家族创建标签

```
/qa tags create Integer AryaStarkReputation
/qa tags create Integer CerseiLannisterReputation
/qa tags create Integer HouseStarkReputation
/qa tags create Integer HouseLannisterReputation
```



搞定！

## 创建事件用于修改好感度标签

好了，那我们来创建一些事件，先创建提升10点家族好感度的事件，再创建降低10点好感度的事件（更多详情请看下方命令）：

```
/qa actions add HouseStarkReputationAdd10 TagInteger housestarkreputation add 10
/qa actions add HouseStarkReputationDeduct1 TagInteger housestarkreputation deduct 1
/qa actions add HouseLannisterReputationAdd10 TagInteger houselannisterreputation add 10
/qa actions add HouseLannisterReputationDeduct1 TagInteger houselannisterreputation deduct 1
```

现在，我们再来创建一个真实事件，当玩家为一个家族完成任务时，就会运行该事件。要是想让这两个家族敌对，那我们就设定，为Stark家族完成任务时，该家族对玩家的好感度+10，同时Lannister家族对玩家的好感度-1。

这个命令将会调用刚才创建的事件 HouseStarkReputationAdd10（Stark家族好感度+10）和 HouseLannisterReputationDeduct1（Lannister家族好感度-1）。

```
/qa actions add HouseStarkQuestCompleted Action HouseStarkReputationAdd10,HouseLannisterReputationDeduct1 1
```

当这个“实际”的事件触发时，我们所创建的另外两个事件也会随之执行。然后，我们同样也给Lannister家族也整个事件：

```
/qa actions add HouseLannisterQuestCompleted Action HouseLannisterReputationAdd10,HouseStarkReputationDeduct1 1
```

## 把真实事件添加到NPC：更多事件嵌套

不过，照这样想，我们还可以设定在玩家完成任务时增加NPC个人对玩家的好感度，对吧？那我们来创建最后的几个事件吧！首先是好感度事件：

```
/qa actions add JonSnowReputationAdd5 TagInteger jonsnowreputation add 5
/qa actions add AryaStarkReputationAdd5 TagInteger aryastarkreputation add 5
/qa actions add CerseiLannisterReputationAdd5 TagInteger cerseilannisterreputation add 5
```

然后，创建最后的真实事件：

```
/qa actions add JonSnowQuestCompleted Action HouseStarkQuestCompleted,JonSnowReputationAdd5 1
/qa actions add AryaStarkQuestCompleted Action HouseStarkQuestCompleted,AryaStarkReputationAdd5 1
/qa actions add CerseiLannisterQuestCompleted Action HouseLannisterQuestCompleted,CerseiLannisterReputationAdd5 1
```

## 把事件添加到我们的任务

好耶！现在我们只需将最后的两个事件当作任务的奖励添加进做好的任务里就好了。我们在之前已经为此创建了 `HelpJonSnow`、`HelpAryaStark` 和 `HelpCerseiLannister` 这三个任务。（偷摸建的，自己建去，你应该知道该怎么创建任务了，如果不知道，去再读一遍[入门教程](#)）。

```
/qa edit HelpJonSnow rewards add Action JonSnowQuestCompleted 1
/qa edit HelpAryaStark rewards add Action AryaStarkQuestCompleted 1
/qa edit HelpCerseiLannister rewards add Action CerseiLannisterQuestCompleted 1
```

噢我的老天爷，我们可算做完了！快去完成一下任务，看看你的好感度有没有变化。

## 查看你的好感度（标签）

这里是一个查看**Stark**家族好感度的例子：

```
/qa variables check TagInteger housestarkreputation
```

## 让我们为我们的好感度系统增加一些特权和奖励：条件

这里举一个例子，这是一条检查**Stark**家族对你是否有至少**50**点的好感度的条件 - 可以用于任何地方，比如任务前置、目标或条件：

```
/qa conditions add HouseStarkReputation50 TagInteger housestarkreputation moreOrEqualThan 50
```

未完待续.....等我有时间再来摸这部分内容。随时欢迎您对这篇教程或者其它教程做出贡献：[为文档作贡献](#)

powered by GitbookFile Modify: 2023-09-27 14:44:31

## 🕒 日常任务

很多人会来问如何创建日常任务 - 这个问题说难也难，说简单也简单，这得根据你的想法来。

### 选择1：每天可接受或完成一个任务的次数不得超过1次

如果你所说的“日常”任务是让玩家每天只能完成或接受某个任务一次，那么做起来非常简单！只需要简单地将接受任务冷却事件设定为1天即可。

```
/qa edit 任务名称 acceptCooldown complete set 1d
```

### 选择2：从预设任务池里抽选日常任务

如果你想创建一个由多个任务组成的任务池，然后玩家每天都可以在这个任务池里抽选一个任务做，这种想法实现起来比较难，但还是能做到的！值得一提的是，最近我们正在为此开发一个系统，用于简化这个过程。

NotQuests这款插件的功能非常强大，虽说是能做到的，但这个过程非常复杂。以下是设计构思：

加入你创建了一个日常任务池，里面有2个任务，一个是A，一个是B。（除此之外，你还可以为每周和每月都设计一下周常任务和月常任务）

1. 给2个任务都先设置1天接受任务冷却时间： `/qa edit A acceptCooldown complete set 1d` 、 `/qa edit B acceptCooldown complete set 1d` 。
2. 创建2个事件（giveQuestA、giveQuestB），该事件执行时会给予玩家任务： `/qa actions add giveQuestA GiveQuest A` 、 `/qa actions add giveQuestB GiveQuest B` 。
3. 创建一个事件，事件命名为Action，该事件执行时将随机执行giveQuestA事件或giveQuestB事件： `/qa actions add giveDailyQuest Action giveQuestA,giveQuestB 1 --minRandom 1 --maxRandom 1` 。这里使用了--minRandom 1和--maxRandom 1两个Flag，用于限制执行的事件数量，最小是1，最大也是1。如果没有这两个Flag，该事件执行时会执行2个事件，给予玩家2个任务，加上这两个Flag后，就只会在这2个事件中随机选择1个事件，即为玩家随机提供1个任务。
4. 前面我们给2个任务都添加了接受任务冷却时间，那这里我们为Action事件添加2个条件，使其在任务A和任务B都不在冷却时间内时才能执行： `/qa actions edit giveDailyQuest conditions add QuestOnCooldown A equals false` 、 `/qa actions edit giveDailyQuest conditions add QuestOnCooldown B equals false` 。
5. 像这样执行事件，这下就能够实现先前构思出的任务池功能： `/qa actions edit giveDailyQuest execute` 。

powered by GitbookFile Modify: 2023-09-30 18:22:11

## 自定义

### GUI物品

你可以自定义在GUI界面中类别（默认：箱子）和任务（默认：书）的物品显示。任务的话，需要使用命令 `/qa edit 任务名称 guiItem ...`，类别的话，需要使用 `/qa categories edit 类别名称 guiItem ...`。

### 高级物品（例如：自定义模型数据）

NotQuests支持任意物品 - 无论是多复杂的物品，比如其它插件的那些特殊物品！NotQuests使用了Bukkit物品序列化API，因此，只要那些插件的物品支持该API，NotQuests就支持这些插件。

如果你有符合这个条件的复杂物品（不仅是材料），要想将它们加进GUI里，只需将该物品拿在手中，然后使用"hand"替代其名称。例如，如果你想将GUI中的类别物品槽的材质换成你那炫酷吊炸天的自定义模型数据物品，只需手拿该物品，然后输入 `/qa categories edit 类别名称 guiItem hand`。就是这么简单！

### 语言

在NotQuests中，你可以修改大部分字符串和翻译（目前还不是全部，但都是最重要的部分）。你可以在NotQuests的"languages"文件夹里找到对应语言进行修改。

### 自定义整个GUI

GUI中的所有物品的材料及其位置都是可以在"languages"文件夹的翻译文件中进行某种程度上的自定义。

### 颜色代码

NotQuests几乎所有地方都支持颜色代码 - 不过可能和你所熟知的颜色代码有所差异。像&6或&c之类的颜色代码不会给你带来太大帮助 - 毕竟它们早已被Mojang遗弃，现如今是由Spigot进行维护 - 而由于NotQuests致力于现代化和创新，所以也遗弃了这些被Mojang遗弃的颜色代码。

与此相对的，我们现如今使用的是MiniMessages，相比之前的颜色代码，它能做一些更高级的东西，而且更简单一些。比如RGB颜色、渐变、甚至还能做到点击命令和悬停消息！你可以在[这里](#)查看MiniMessages文档。

现在，我们来给我们的任务添加一个带颜色的显示名称：

```
/qa edit TheVirus displayName set <red>A <bold>Deadly</bold> <#eb34a1>Virus
```



或者，甚至还可以这样：

```
/qa edit TheVirus displayName set <rainbow>A <bold>Deadly</bold></rainbow> <gradient:#eb34a1:#ffffff>Virus</gradient>
```



超级酷，对吧？你甚至可以在很多命令上用自动补全功能补全MiniMessage颜色标签。同时，NotQuests还内置了一些颜色，让你能使用它们来实现一些统一的外观。例如 `<highlight>`、`<highlight2>` 或 `<main>`。这些都可以在config里自定义，以调整NotQuests整体的颜色主题。

举例：在任务的第一目标描述中添加可点击的链接：

```
/qa edit 你的任务名称 objectives edit 1 description set <click:OPEN_URL:此处放入url链接>Click here</click>
```

powered by GitbookFile Modify: 2023-09-27 20:24:03

## 💬 对话

NotQuests有一个特别棒的对话系统，你可以创建无数个对话，每个对话中，说话人、选项、路径和结果都是无限的。这也是整个插件中最难以理解的功能，因为对话系统是通过创建YAML文件来完成的。

在这之前，请先了解一下YAML（yaml），先对它有一个基本的认识，[这里](#)有一个教程可供你学习。确保在继续往下阅读之前，先理解yaml的层级关系（例如，在下面的这段演示用的对话代码中，你要一眼就看得出来，“greeting”是说话人“Atlas”的一部分，而“Atlas”又是“Lines”的一部分。在YAML中，通常使用缩进来表示层级关系，某个位置的缩进多了或少了，都会破坏整个层级关系，进而破坏整个句法）。

接下来，安装一个合适且支持YAML的编辑器。这里我推荐使用[Visual Studio Code](#)。如果没有一个合适的编辑器，YAML的句法会很容易被破坏（特别是空格、标签和缩进），这会导致你的对话系统无法工作。况且Visual Studio Code会在你打错句法的时候提示你。

## 创建一个演示对话

如果你还不知道这么创建对话，那么跟着我们先用命令 `/qa conversations create test --demo` 创建一个演示对话。在命令的句尾输入`--demoFlag`可以创建一个填充了演示数据的对话，而不是空白文件。命令中的`test`是这个对话的名称。创建完名叫`test`的对话后，将会生成一个对话文件，你可以在 `plugin/notquests/default/conversations/test.yml` 找到这个对话的文件 - 去找这个文件，然后用Visual Studio Code打开！这个文件应该是这个样子的：

```
start: Atlas.specialgreeting,Atlas.greeting1
lines:
  Atlas:
    color: "<BLUE>"
    delay: 200
    greeting1:
      text: "Hello traveler! I am atlas, the keeper of time!"
      next: Player.greeting1,Player.greeting2
      shout: true
    specialgreeting:
      text: "I don't wanna talk to you while you fulfill this condition. Bye!"
      conditions:
        - condition replaceThisWithTheNameOfYourCondition
    answer1:
      text: "That's a secret, but without me, you all wouldn't exist."
      next: Atlas.answer3
    notime:
      text: "Time is a rare good. Au revoir!"
    answer3:
      text: "Anyways though, what are you doing here?"
      next: Player.three,Player.four,Player.five
    answer4:
      text: "Oooh I see! I'm sure you are in need for some time, then. Should I lend you some?"
      next: Player.lend,Player.nolend
    answer5:
      text: "Here it is!"
      next: Atlas.answer6
    answer6:
      text: "[Hands time]"
      next: Player.bye
    nicebye:
      text: "You're very welcome. Good luck on your ventures!"
      actions:
        - "action replaceThisWithTheNameOfYourAction"
        - "action anotherAction"
  Player:
    delay: 200
    greeting1:
      text: "Nice to meet you! Why do you need to keep time?"
      next: Atlas.answer1
    greeting2:
      text: "I have no time for you."
      next: Atlas.notime
    three:
      text: "I'm just exploring the area!"
      next: Atlas.answer4
    four:
      text: "I'm on a mission."
      next: Atlas.answer4
    five:
      text: "I'm here to meet the king."
      next: Atlas.answer4
    lend:
      text: "Yes, please! I could use some time"
      next: Atlas.answer5
    nolend:
      text: "No, sorry, I have enough time. Thank you for the offer, though!"
      next: Atlas.notime
    bye:
      text: "Thank you a lot, time keeper. See you around!"
      next: Atlas.nicebye
```

用你刚学到的YAML知识，尝试着去理解一下这个文件中的句法，以及它是如何运作的。动动你的小脑瓜，从头到尾看一遍。尝试改动一点东西，看看会发生什么。每当你修改完文件并保存后，你可以在游戏内使用 `/qa reload conversations` 命令去加载你的改动。还请留意你的控制台，看看是否有警告和错误，如果有，看一下哪里出错了。然后，你可以使用 `/qa conversations start 你的对话名称` 命令开始这段对话，本篇内容创建的对话名称叫做`test`，所以我们这里应该输入 `/qa conversations start test` 命令。

## 一些解释

我没有太多时间，所以我暂时做不出100%详尽的教程 - 不过，如果后续我有时间，我会尝试去实验并理解这些功能，随后再慢慢优化这篇教程，那我们先来讲一些解释：



## 专业属于

这份文件是一个对话（**test**）。一个对话可以包含多个说话人（本篇举例Atlas和玩家）。每个说话人都有多个对话行，格式是"**<说话人>**,**<对话行名称>**"。举例：`Atlas.greeting1`。

## start

在第一行，你应该可以看见 `start: Atlas.specialgreeting,Atlas.greeting1` 这一行。这行设定了对话会从何处开始，并指定了2个对话行：`Atlas.specialgreeting`、`Atlas.greeting1`。

这行句法的运行原理是先运行Atlas.specialgreeting对话行，那我们先来看这个对话行：

```
specialgreeting:
  text: "I don't wanna talk to you while you fulfill this condition. Bye!"
  conditions:
    - condition replaceThisWithTheNameOfYourCondition
```

正如你所见，这个对话行包含着文本（**text**）和条件（**conditions**）。文本的内容就是会发给玩家的内容。而条件则是要求玩家必须满足这个条件，才能播放对话行的文本。你可以在游戏中使用 `/qa conditions create 你的条件名称 ...` 命令创建条件 - 演示对话中的条件显然不可能存在（演示对话中的条件一栏翻译过来是“在这里替换成你的条件”），所以你可以替换掉它。

那么，如果不满足这个条件，会发生什么？当然，由于这是对话的开头部分，而且在开头部分的 `start: Atlas.specialgreeting,Atlas.greeting1` 指定了2个对话行，第一个不满足条件，那么对话系统就会尝试播放逗号后的下一个对话行 - 也就是Atlas.greeting1：

```
answer1:
  text: "That's a secret, but without me, you all wouldn't exist."
  next: Atlas.answer3
```

所以说，开头部分的设定是按照前后顺序依次进行的，基本上只会播放第一个对话行（示例中是Atlas.specialgreeting），并且忽略掉第二个对话行（Atlas.greeting1），但是，如果玩家不满足第一个对话行的条件，那么就会按顺序进行下一个对话行。这就是将对话分支成多个路径的方法！

## next

每个对话行都有一个 `next`：对象。我们用Atlas.answer1来做个例子：

```
answer1:
  text: "That's a secret, but without me, you all wouldn't exist."
  next: Atlas.answer3
```

**next**属性的对象决定了该对话行之后播放哪个对话。在这个示例中，接下来会播放的是 `Atlas.answer3`。这点应该一眼就看得出来吧。类似于前面所说的**start**，你也可以在此处用逗号分隔的方式指定多个对话行，从而根据所指定的对话行的条件进行分支。

要是是一个对话行里没有**next**属性的话，会发生什么呢？

很简单！

对话将会结束。没有下一步，对话会直接结束。

## 时间和条件

每个对话行都可以添加**action**事件属性和**conditions**条件属性。当播放到此对话行时，就会执行该对话行下所指定的所有事件。只有对话行中指定的所有条件都满足时，才会播放该对话行。

看看上面的演示对话 - 上面有关于事件属性和条件属性的示例。

条件属性的句法通常是"**condition** 你的条件名称"。以"**condition**"作为开头，可以让它查询你在游戏中事先创建的条件，后面再写上指定的条件名称。事件的运作原理也相似。

## 否定条件

你在条件前面加上"**!**"用以否定条件。示例：

- "**!condition** 你的条件名称" 确保使用引号（""）将这个条件全部包裹起来。

## 排列式事件/条件

根据上面所说的，在前面加上"**condition**"可以让它查询你在游戏中事先创建的条件。你还可以这样：

``` conditions:

- Money moreThan 100

或者：

actions:

- StartConversation 其它对话名称

而不能：

actions:

- **action** 你在游戏中实现创建的用于触发另一个对话行的事件名称 我非常不推荐使用第三个句法，因为这部分内容没有相关文档，并且没有任何东西会检查你做的是否正确，不像在游戏里那样可以准确地告知你是否有错误（`/qa conditions create` 和 `/qa actions create`）。

## 常见问题

**Q:** 对话系统延迟功能无法正常运转

**A:** 这个功能处于一个“半损坏”的状态，预计在未来的版本中修复。

**Q:** 一些东西发生了错误

**A:** 请阅读下一部分（“一些东西无法正常运转”）。

**Q:** 我先前的对话正在消失，消息也正在消失，出现了一些奇怪的问题，非常奇怪

**A:** 你所遇到的问题可能是插件的对话神奇封包/对话修整功能导致的，这个功能可以简化你的对话历史记录！有人喜欢有人爱，但也有人觉得这很奇怪，这是可以理解的，你可以在NotQuests的config文件里禁用这个功能！

**Q:** 为什么我在Discord里无法获得任何帮助？

**A:** 对话系统复杂的要命，我可能需要很长事件才能搞懂问题的所在。并且，我是真的真的没啥时间。不过别担心！99%的问题最后都会被证明是用户自己的问题，而不是NotQuests插件本身的问题，所以说，问题的关键不在插件，而在你本身，这是可以解决的！对话系统非常容易出错，这也能引出另一个问题：

**Q:** 既然对话系统那么难，那能搞个GUI界面吗？

**A:** 不会的，绝对不会。要真用上这玩意儿的话情况不会变好，反而会更加糟糕。相反，我正计划着弄一个漂亮的Web UI。

**Q:** feature XY何时可以实现？

**A:** 或许需要一周时间，或许永远也不能。我给不出一个大体的时间，也无法预测。因为我不知道我何时才能摸一摸NotQuests，也不知道这一摸能摸出多少进度。

**Q:** 这篇教程何时更新更多内容？

**A:** 上面一些答案或许已经说明了，很显然，我没有太多时间去更新这篇内容。这个插件是开源的，但可悲的是，掌握这个插件使用方法的人并不愿意与这个社区的其它玩家进行分享，或许屏幕前的你也会做出这样的选择。这里的文档和教程都是开源的，每个人都可以参与到其中，但没有人这样做，所以不要指责我！很多人都有足够的知识编写这样的教程，但我没有看见有人这样做。文档和教程都是公益性质的，我和玩家们（包括你）一样，无法从中获得任何报酬。因此，如果你可以，请为他人考虑，来贡献出自己的一份力吧！♥

## 一些东西无法运作

如果你每次使用 `/qa reload conversations` 时，如果出现任何错误，都会在服务器控制台上弹出警告。所以请看看控制台的内容，看看这些该死的警告，它们会告诉你到底出了什么错误。

所以，每次重载对话系统时，要是出现了什么错误，多看看服务器控制台会是一个很好的习惯。如果控制台说出现了一个大错误，并且字体颜色不是黄色的，那么错误就肯定涉及到了YAML句法，你绝对对哪个句法搞错了。正如我在本篇内容的第二段和第三段所说的，你得使用一个合适的编辑器，它会在你出现句法错误时给你提示（比如Visual Studio Code），然后再去看看YAML教程吧。TAML对语法要求非常严苛 - 如果你的空格多了或是少了，都会引发错误。

再加一个温馨提示：引号还是多用点比较好。

powered by [GitbookFile](#) Modify: 2023-09-30 18:22:25

## 文档

欢迎来到我们的新文档！你可以在右边栏里选择你想了解的内容。还请务必看一看子分类的内容，例如“类型”。另外，还请务必查看前面的教程部分！目前那可比文档内容有用得多，其中还包含了许多有价值且有帮助的内容。

powered by GitbookFile Modify: 2023-09-29 22:41:58

## 目标

玩家必须推进并达成目标才能完成任务。有一些目标来源于其它插件，你必须安装这些插件作为前置才能使用这部分目标。  
首先，我们会先介绍我们插件所自带的默认目标：

## 默认目标

### 破坏方块|BreakBlocks

[!NOTE|label:描述] 达成需求：玩家破坏指定数量的方块。

命令参数：

- **<材料名称>** - 用于指定需要玩家破坏的方块。可以用“hand”来指定你当前手持的物品。可以用“any”将其设定为任何物品都可以计入数量。你还可以指定在NotQuests物品系统中制作的物品。
- **<数量>** - 用于指定需要玩家破坏的方块的数量。
- **(flags)** - 可选Flag
  - **--doNotDeductIfBlockIsPlaced** - 默认情况下，如果你放置了破坏方块目标中指定的方块，进度将会被相应地扣除。否则，玩家可以通过放置后再破坏的方式刷进度。若设置了此Flag，将会禁用该安全机制。

示例： `/qa edit 任务名称 objectives add BreakBlocks grass_block 4` （设定某个任务的目标为破坏4个草方块）

### 繁殖|BreedMobs

[!NOTE|label:描述]

达成需求：玩家繁衍出指定生物。

命令参与：

- **<实体名称>** - 用于指定需要玩家繁殖的生物类型。可以用“any”将其设定为任何生物都可以计入数量。
- **<数量>** - 用于指定需要玩家繁殖出的生物数量。

示例： `/qa edit 任务名称 objectives add BreedMobs cow 4` （设定某个任务的目标为繁衍4只牛）

### 条件|Condition

[!NOTE|label:描述]

达成需求：玩家需要满足特定条件（需要先使用 `/qa conditions` 创建出条件）。插件每秒都会检测玩家是否满足特定条件，或是检测玩家是否执行了该条件所指定的事件（例如：在条件中指定了某个金钱事件）。毫无疑问这是在以另一种方式增加目标。例如，你甚至可以使用PlaceholderAPI来创建目标。

命令参数：

- **<条件名称>** - 用于指定玩家需要达成的条件名称。这里输入的是需要让插件来检测的条件名称，条件需要用 `/qa conditions` 命令事先创建。
- **(flags)** - 可选Flag
  - **--checkOnlyWhenCorrespondingVariableValueChanged** - 若设置了此Flag，插件就只会在玩家执行条件所指定的事件时才会进行检测。插件将会停止定时检测功能。正因如此，这样做的话会更加高效一些。

示例： `/qa edit 任务名称 objectives add Condition myCondition` （设定某个任务的目标为达成名叫“myCondition”的条件）

### 消耗物品|ConsumeItems

[!NOTE|label:描述] 达成需求：玩家需要消耗指定物品。通常来说，吃喝某物才算是消耗物品 - 比如说吃苹果，就是消耗了苹果。

命令参数：

- **<材料名称>** - 用于指定玩家需要消耗的物品名称。可以用“hand”来指定你当前手持的物品。你还可以指定在NotQuests物品系统中制作的物品。
- **<数量>** - 用于指定玩家需要消耗的物品数量。

示例： `/qa edit 任务名称 objectives add ConsumeItems apple 5` （设定某个任务的目标为消耗5个苹果）

### 合成物品|CraftItems

[!NOTE|label:描述] 达成需求：玩家需要合成指定物品。

命令参数：

**<材料名称>** - 用于指定玩家需要合成的物品名称。可以用“hand”来指定你当前手持的物品。你还可以指定在NotQuests物品系统中制作的物品。

- **<数量>** - 用于指定玩家需要合成的物品数量。

示例： `/qa edit 任务名称 objectives add CraftItems enchanting_table 1` （设定某个任务的目标为合成1个工作台）

### 交付物品|DeliverItems

[!NOTE|label:描述] 达成需求：玩家需要交付指定物品给NPC（Citizen NPC和盔甲架都可以）。交付的意思是从玩家物品栏那里移除一定数量的指定物品。

命令参数：

- **<材料名称>** - 用于指定玩家需要交付的物品名称。可以用“hand”来指定你当前手持的物品。你还可以指定在NotQuests物品系统中制作的物品。
- **<数量>** - 用于指定玩家需要交付的物品数量。
- **<NPC ID 或者 Armorstand>** - 要么输入Citizen的NPC ID，要么输入Armorstand获取一个盔甲架，直接放置盔甲架即可将该目标的交付对象指定为此盔甲架。

示例： `/qa edit 任务名称 objectives add DeliverItems coal 20 armorstand` （设定某个任务的目标为交付20个煤炭给指定盔甲架）

## 🔮 附魔|Enchant

[!NOTE|label:描述]

达成需求：玩家需要进行附魔 命令参数：

- **<魔咒名称>** - 用于指定玩家需要附魔出的魔咒。
- **<材料名称>** - 用于指定玩家需要附魔的物品名称。可以用“any”将其设定为任何物品都可以计入数量。
- **<数量>** - 用于指定玩家需要附魔的物品数量。
- **(flags)** - 可选Flag
  - **--min <最小等级>** - 用于指定附魔的最小等级。
  - **--max <最大等级>** - 用于指定附魔的最大等级。 示例： `/qa edit 任务名称 objectives add Enchant minecraft:efficiency diamond_pickaxe 3.0 --min 2.0` （设定某个任务的目标为给钻石镐附魔出不低于2级的效率魔咒3次）

## 🎣 垂钓|FishItems

[!NOTE|label:描述] 达成需求：玩家需要钓上指定物品。

命令参数：

- **<材料名称>** - 用于指定玩家需要钓上的物品。可以用“hand”来指定你当前手持的物品。可以用“any”将其设定为任何物品都可以计入数量。你还可以指定在NotQuests物品系统中制作的物品。
- **<数量>** - 用于指定玩家需要钓上的物品数量。

示例： `/qa edit 任务名称 objectives add FishItems cod 3` （设定某个任务的目标为钓上3条鲤鱼）

## 👉 交互|Interact

[!NOTE|label:描述] 达成需求：玩家需要按下鼠标左键或右键，或者二者一起按，以此与指定位置进行交互。

命令参数：

- **<数量>** - 用于指定玩家需要交互的次数。
- **<世界名称>** - 玩家需要进行交互的指定位置所在的世界名称。
- **<x>** - 玩家需要进行交互的指定位置的x轴坐标。
- **<y>** - 玩家需要进行交互的指定位置的y轴坐标。
- **<z>** - 玩家需要进行交互的指定位置的z轴坐标。
- **(flags)** - 可选Flag
  - **--leftClick** - 若设置了此Flag，玩家需要与指定位置进行交互的交互操作将会被设定为鼠标左击。
  - **--rightClick** - 若设置了此Flag，玩家需要与指定位置进行交互的交互操作将会被设定为鼠标右击。
  - **--cancelInteractions** - 若设置了此Flag，玩家与指定位置交互时产生的交互事件将不会被执行。例如，如果位置指定的是一个箱子，玩家需要右击箱子，在未设置此Flag的情况下，箱子会被打开，目标也会达成，但如果设置了此Flag，那么“打开箱子”这一交互事件将不会被执行，箱子不会被打开，目标也会达成。
  - **--taskDescription <目标描述>** - 若设置了此Flag，输入描述后，玩家就可以在目标位置看到你所输入的描述了，以便玩家了解该目标具体要做什么。
  - **--maxDistance <半径（单位：格）>** - 若你想指定的交互位置不是一个点，而是一个范围区域，那么可以设置此Flag，输入半径后，需要进行交互的位置就会限定在指定位置的某个半径范围内。

示例： `/qa edit 任务名称 objectives add Interact 2 world 1453 71 -2451 --rightClick --maxDistance 2 --cancelInteraction --taskDescription "Find Trents fishing rod"` （设定某个任务的目标为需要在名叫world的世界里，鼠标右击以x1453,y71,z-2451为中心周围2格半径范围内的方块2次，右击时像开箱子这样的交互事件会被取消，目标描述是“Find Trents fishing rod”）

## 🦘 跳跃|Jump

[!NOTE|label:描述] 达成需求：玩家需要进行指定次数的跳跃。

命令参数：

- **<数量>** - 用于指定玩家需要跳跃的次数。

示例： `/qa edit 任务名称 objectives add Jump 10` （设定某个任务的目标为跳跃10次）

## 🩸 击杀生物|KillMobs

[!NOTE|label:描述] 达成需求：玩家需要击杀指定生物。

命令参数：

- **<实体名称>** - 用于指定需要玩家击杀的生物类型。可以用“any”将其设定为任何生物都可以计入数量。可以用“player”指定生物类型为玩家。如果你安装了MythicMobs或是EcoBosses插件，你也可以使用这些插件（英文原文写的是mod，应该是口误？）创造出的生物，你甚至还可以直接用 `mmfaction:你的阵营名称` 指定MythicMobs的某个阵营。
- **<数量>** - 用于指定玩家需要击杀生物的次数。
- **(flags)** - 可选Flag
  - `--nametag_containsany <字符串>` - 若设置了此Flag，插件将会检测生物名称是否包含该字符串。
  - `--nametag_equals <字符串>` - 若设置了此Flag，插件将会检测生物名称是否吻合该字符串。
  - ProjectKorra联动：
    - `--withProjectKorraAbility <技能>` - 若设置了此Flag，玩家需要使用ProjectKorra的技能击杀才能计入进度。  
示例： `/qa edit 任务名称 objectives add KillMobs zombie 10` （设定某个任务的目标为击杀10只僵尸）

## 13.4 数字变量|NumberVariable

[!NOTE|label:描述] 达成需求：玩家需要满足变量中的要求。这样做可以相对地在目标中使用变量。 命令参数：

- **<待补充>** 示例： `/qa edit 任务名称 objectives add NumberVariable PlaytimeMinutes moreOrEqualThan PlaytimeMinutes+2` （设定某个任务的目标为需要再游玩2分钟）

## 13.4 目标|Objective

[!NOTE|label:描述] 达成需求：玩家需要完成下属于目标，所有目标完成后，该目标才会被标记为完成。这是一个子目标的容器，也就是父目标。 命令参数：

- **<目标显示名称>** - 用于指定该目标的显示名称。  
示例： `/qa edit 任务名称 objectives add Objective "Crafting Objectives"` （设定某个任务为父目标，显示名称为“Crafting Objectives”）

## 13.4 打开埋藏的宝藏|OpenBuriedTreasure

[!NOTE|label:描述] 达成需求：玩家需要打开指定数量的埋藏的宝藏（埋藏的宝藏是Minecraft原版特性）。 命令参数：

- **<数量>** - 用于指定玩家需要打开埋藏的宝藏的个数。  
示例： `/qa edit 任务名称 objectives add OpenBuriedTreasure 3` （设定某个任务的目标为打开3个埋藏的宝藏）

## 13.4 其它任务|OtherQuest

[!NOTE|label:描述] 达成需求：玩家需要完成其它任务。

命令参数：

- **<其它任务名称>** - 用于指定玩家需要完成的其它任务的名称。
- **<数量>** - 用于指定玩家需要完成的其它任务的次数。
- **(flags)** - 可选Flag
  - `--countPreviouslyCompletedQuests` - 若设置了此Flag，玩家在解锁该目标前完成的所有指定任务的次数都将计入在内。否则，只有在解锁后完成的指定任务才会被计算在内。  
示例： `/qa edit 任务名称 objectives add OtherQuest 其它任务名称 2 --countPreviouslyCompletedQuests` （设定某个任务的目标为完成名叫“其它任务名称”的任务2次，该目标解锁前的完成次数也会被计入。）

## 13.4 拾取物品|PickupItems

[!NOTE|label:描述] 达成需求：玩家需要从地上拾取指定物品到物品栏里。

命令参数：

- **<材料名称>** - 用于指定需要玩家拾取的物品。可以用“hand”来指定你当前手持的物品。可以用“any”将其设定为任何物品都可以计入数量。你还可以指定在NotQuests物品系统中制作的物品。
- **<数量>** - 用于指定玩家需要拾取的物品数量。
- **(flags)** - 可选Flag
  - `--doNotDeductIfItemIsDropped` - 默认情况下，如果你丢弃了拾取物品目标中指定的物品，进度将会被相应地扣除。否则，玩家可以通过丢弃后再拾取的方式刷进度。若设置了此Flag，将会禁用该安全机制。
  - `--doNotDeductIfItemIsPlaced` - 默认情况下，如果你放置了拾取物品目标中指定的物品，进度将会被相应地扣除。否则，玩家可以通过放置后再破坏并拾取的方式刷进度。若设置了此Flag，将会禁用该安全机制。



- `--doNotDeductIfItemIsRemovedFromInventory` - 默认情况下，如果你从物品栏里移除拾取物品目标中指定的物品（比如：放入箱子），进度将会被相应地扣除。否则，玩家可以通过将物品放入箱子后再破坏并拾取的方式刷进度。若设置了此Flag，将会禁用该安全机制。示例：`/qa edit 任务名称 objectives add PickupItems dirt 12`（设定某个任务的目标为拾取12个泥土）

## 🏠 放置方块|PlaceBlocks

[!NOTE|label:描述] 达成需求：玩家需要放置指定方块。

命令参数：

- `<材料名称>` - 用于指定需要玩家放置的方块。可以用“hand”来指定你当前手持的物品。可以用“any”将其设定为任何物品都可以计入数量。你还可以指定在NotQuests物品系统中制作的物品。
- `<数量>` - 用于指定玩家需要放置的方块数量。
- `(flags)` - 可选Flag
- `--doNotDeductIfBlockIsBroken` - 默认情况下，如果你破坏了放置方块目标中指定的方块，进度将会被相应地扣除。否则，玩家可以通过破坏后再放置的方式刷进度。若设置了此Flag，将会禁用该安全机制。  
示例：`/qa edit 任务名称 objectives add PlaceBlocks acacia_planks 20`（设定某个任务的目标为放置20个金合欢木板）

## 📍 抵达位置|ReachLocation

[!NOTE|label:描述] 达成需求：玩家需要抵达指定位置。

命令参数：

- `<选择器类型>` - 目前，只有`worldeditselection`一种选择器类型。你需要用WorldEdit中的选择器选择一个区域设定为玩家需要抵达的位置。正因如此，要创建此目标，你需要安装一个WorldEdit。请在运行此命令前，先创建一个WorldEdit选择器（使用`//wand`或者`//pos1`和`//pos2`命令）。
- `<位置名称>` - 用于指定玩家需要抵达的位置的名称。这个名称将会显示在目标描述中。  
示例：`/qa edit 任务名称 objectives add ReachLocation worldeditselection Luthers Church`（设定某个任务的目标为地抵达一个名叫Lythers Church的位置，需要使用worldeditselection选择器事先选择出该区域）

## 👤 🖥️ 运行命令|RunCommand

[!NOTE|label:描述] 达成需求：玩家需要运行指定命令。命令参数：

- `<数量>` - 用于指定玩家需要运行命令的次数。
- `<命令>` - 用于指定玩家需要运行的命令，如果需要运行的命令中包含着空格，请将该命令放在引号（" "）中。指定的命令无需在开头加斜杠（/）
- `(flags)` - 可选Flag
- `--ignoreCase` - 若设置了此Flag，将会无视指定的命令的大小写。也就是说，运行的命令无论是`/warp spawn`还是`/wArP sPaWn`都可以。
- `--cancelCommand` - 若设置了此Flag，当目标处于解锁状态时，运行命令的事件不会执行，而是会被取消。所以说，如果玩家运行指定的命令，除了增加该目标的进度以外，什么都不会发生。  
示例：`/qa edit 任务名称 objectives add RunCommand 2 "warp spawn" --ignoreCase`（设定某个任务的目标为运行`/warp spawn`命令2次，无视大小写）

## 👤 🐑 剪羊毛|ShearSheep

[!NOTE|label:描述] 达成需求：玩家需要剪指定数量的羊毛。命令参数：

- `<数量>` - 用于指定玩家需要剪的的羊毛数量。
- `(flags)` - 可选Flag
- `--cancelShearing` - 若设置了此Flag，当目标处于解锁状态时，剪羊毛的事件不会执行，而是会被取消。所以说，如果玩家拿剪刀右击羊，除了增加该目标的进度以外，什么都不会发生。  
示例：`/qa edit 任务名称 objectives add ShearSheep 4 --cancelShearing`（设定某个任务的目标为剪4次羊毛，取消剪羊毛事件）

## 🔥 烧炼物品|SmeltItems

[!NOTE|label:描述] 达成需求：玩家需要从熔炉中取出特定数量的烧炼好的物品。没错，玩家必须将物品烧炼后，再将其去除，该目标计入的是输出的物品数量，而不是放入熔炉的物品数量。

命令参数：

- `<材料名称>`：用于指定需要玩家通过烧炼获得的物品名称。可以用“hand”来指定你当前手持的物品。可以用“any”将其设定为任何物品都可以计入数量。你还可以指定在NotQuests物品系统中制作的物品。
- `<数量>` - 用于指定玩家需要烧炼出的物品数量。  
示例：`/qa edit 任务名称 objectives add SmeltItems coal 4`（设定某个任务的目标为通过烧炼得到4个煤炭）

## 👤 🦏 潜行|Sneak

[!NOTE|label:描述] 达成需求：玩家需要进行潜行。命令参数：

- `<数量>` - 用于指定玩家需要进行潜行的次数。

示例: `/qa edit 任务名称 objectives add Sneak 4` (设定某个任务的目标为潜行4次)

## 与NPC对话|TalkToNPC

[!NOTE|label:描述] 达成需求: 玩家需要与NPC (Citizen NPC或者盔甲架) 进行对话 (右击)。

命令参数:

- **<NPC ID 或者 Armorstand>** - 要么输入Citizen的NPC ID, 要么输入Armorstand获取一个盔甲架, 直接放置盔甲架即可将该目标的对话对象指定为此盔甲架。

示例: `/qa edit 任务名称 objectives add TalkToNPC armorstand` (设定某个任务的目标为与指定盔甲架对话)

## ! 触发器命令|TriggerCommand

[!NOTE|label:描述] 达成需求: 需要有足够权限的人 (可以是命令方块或是有OP权限的玩家。通常由控制台运行) 才能运行的特殊命令。这可以使得NotQuests能够与许多不同的插件进行联动。

### 使用案例

设定一个“为我们投票”的目标, 一旦玩家进行投票后, 目标即达成。至于投票插件, 你得在奖励部分添加一个触发器命令。

然后, 一旦插件检测投票插件奖励部分的触发器命令被从控制台运行了 (即玩家投票后), 该玩家的触发器命令目标就会达成。

如果你想用上像“Interactions”这样的插件制作出特别的任务, 这项功能非常有用, 如果你想制作一个更细节、能与NPC开展脚本化对话的话可以用那个插件。不过请注意: 那个插件的上手门槛很高。同时, 请注意: 较新版本的NotQuests插件已经拥有了完整的对话系统。

示例: 如何给任务名称为“1”的示例任务添加触发器命令

```
/nquestadmin edit 1 objectives add TriggerCommand [Trigger-Name] [Amount To Trigger]
```

```
/qa edit 1 objectives add TriggerCommand playervoted 1
```

该示例的对应触发器命令应该是 `/qa triggerObjective playervoted NoeX` - 此处NoeX是我的游戏ID。

在我们的投票示例中, 如果你将该命令在你的投票插件中设定为投票奖励, 玩家一旦进行投票, 目标就会达成。

```
You have successfully completed the objective  
TriggerCommand for quest 1!
```

```
/qa triggerObjective playervoted NoeX_
```

命令参数:

- **<触发器名称>** - 用于指定触发器的名称。这里触发器名称需要应用到触发器命令中 (`/qa triggerObjective <触发器名称>`)。
- **<数量>** - 用于指定完成目标所需的触发器/命令被触发/运行的次数。

示例: `/qa edit 任务名称 objectives add TriggerCommand trigger1 1` (设定某个任务的目标为执行名为“trigger1”的触发器命令1次)

触发“触发器”/达成目标的命令: `/qa triggerObjective trigger1 {玩家ID}` 玩家ID需要替换为玩家游戏ID。

## Citizens联动目标

### KillEliteMobs

## Jobs Reborn联动目标

### JobsRebornReachJobLevel

## Project Korra联动目标

### ProjectKorraUseAbility

## Slimefun联动目标

### SlimefunResearch

## Towny联动目标

 **TownyNationReachTownCount**

 **TownyReachResidentCount**

## BetonQuest联动目标

 **BetonQuestObjectiveStateChange**

## UltimateJobs联动目标

 **UltimateJobsReachJobLevel**

powered by GitbookFile Modify: 2023-10-04 18:40:59

## 🧐 条件

原始条件可以使用 `/qa conditions` 命令进行创建。这些都将保存在 `plugins/notquests/categoryname/conditions.yml`。条件还可以作为任务前置、目标条件和事件条件。你可以通过 `/qa conditions edit conditionname check <玩家ID(可选)>` 命令“测试”你通过 `/qa conditions` 命令创建的条件。

所有条件通用命令参数：

- `(flags)` - 可选Flag
- - `--negate` - 否定条件，如果条件是True，那么就会转变为False，如果条件是False，那么就会转变为True。
- - `--category <类别名称>` - 如果是一个原始条件，储存于conditions.yml里的条件，你可以在这里为其设定类别。

## 默认变量条件

这些条件将会充当变量的“容器”，这是啥意思呢？

好吧.....举个例子，**Number**条件就是这样的变量条件。在游戏中，它会将**Number**变量像很多条件的那样，转化为相应的条件，所以条件的数量实际上比你所认为的要得多。

更多请参阅变量部分。

## ? 布尔|Boolean

[!NOTE|label:描述] 满足需求：The Boolean variable equals the expression。Boolean的取值只有2种：True和False。换言之就是表达Yes和No。

命令参数：

`<变量类型>` - 用于指定Boolean变量的类型。  
`<变量参数>` - 有些变量类型可能存在附加参数，也可能不存在。  
`<运算符>` - 当然，Boolean的运算符只有一种：`equals`（等于）`<表达式>` - 用于指定变量的输出结果。常见值为 `true` 和 `false`，不过你也可以在这里将它与其它Boolean变量进行比较。

示例：

- `/qa conditions add 条件名称 Flying equals false`（设定某条件为正在飞行时输出false）- 此条件中Flying为变量类型。
- `/qa conditions add 条件名称 True equals (Money>10)&Flying`（设置某条件为金钱大于10且正在飞行时输出true）- 此条件中“True”为变量类型。
- - 同样的条件还可以这样写：`/qa conditions add 条件名称 True equals Condition(Conditions:Flying&IsRich)`（设定某条件为同时满足条件Flying和IsRich时输出true）- 你需要事先创建 Flying 和 IsRich 条件。
- - 再或者：`/qa conditions add 条件名称 Condition equals Flying&IsRich`（设定某条件为同时满足条件Flying和IsRich时输出true）- 你需要事先创建 Flying 和 IsRich 条件。

## 📋 列表|List

[!NOTE|label:描述] 满足需求：The List variable equals the expression。命令参数：

`<变量类型>` - 用于指定List变量的类型。  
`<变量参数>` - 有些变量类型可能存在附加参数，也可能不存在。  
`<运算符>` - 运算符有这些选择：`contains`（包含）、`containsIgnoreCase`（包含，不区分大小写）、`equals`（等于）和 `equalsIgnoreCase`（等于，不区分大小写）。  
`<表达式>` - 用于指定变量的输出结果。这取决于变量类型。

示例：`/qa conditions add 条件名称 ActiveQuests contains 任务名称`（设定某条件为名叫“任务名称”的任务处于激活状态）- 检测如果该玩家名叫“任务名称”的任务当前处于激活状态，条件即满足。该条件中ActiveQuests为List变量。

## 📖 ItemStackList

[!NOTE|label:描述] 满足需求：The ItemStackList variable equals the expression。命令参数：

`<变量类型>` - 用于指定ItemStackList变量的类型。  
`<变量参数>` - 有些变量类型可能存在附加参数，也可能不存在。  
`<运算符>` - 运算符有这些选择：`contains`（包含）和 `equals`（等于）。  
`<表达式>` - 用于指定变量的输出结果。这取决于变量类型，但必须是物品。

示例：`/qa conditions add 条件名称 Inventory contains diamond 32`（设定某条件为检测玩家物品栏是否有至少32个钻石）- 该条件中Inventory为ItemStackList变量。

## 100 数字|Number

[!NOTE|label:描述] 满足需求：The Number variable equals the expression。命令参数：

`<变量类型>` - 用于指定Number变量的类型。  
`<变量参数>` - 有些变量类型可能存在附加参数，也可能不存在。  
`<运算符>` - 运算符有这些选择：`equals`（等于）、`lessOrEqualThan`（小于或等于）、`lessThan`（小于）、`moreOrEqualThan`（大于或等于）和 `moreThan`（大于）。  
`<表达式>` - 用于指定变量的输出结果。此处的表达式很强大，因为你可以在这里输入任何类型的数学表达式，甚至可以包括其它Number变量。

示例：`/qa conditions add 条件名称 Health moreOrEqualThan 9.0`（设定某条件为检测玩家生命值是否大于或等于9）- 该条件中生命值Health为Number变量。再举个包含

了数学表达式和其他变量的例子： `/qa conditions add 条件名称 Money moreThan QuestPoints*Money+500-30/2`（设定某条件为检测玩家金钱是否大于任务点数×金钱+500-30÷2这个表达式的运算结果）- 该条件中Money为Number变量。再举个带有非常高级的表达式的例子： `/qa conditions add 条件名称 Money moreThan 10+TagInteger(TagName=reputation)*TagInteger(TagName=level)`（设定某条件为检测玩家的金钱是否大于10+整数标签“好感度”数值×整数标签“等级”数值）- 该条件中Money为Number变量。

## AB 字符串|String

[!NOTE|label:描述] 满足需求：The String variable equals the expression。 命令参数：

<变量类型> - 用于指定String变量的类型。

<变量参数> - 有些变量类型可能存在附加参数，也可能不存在。 <运算符> - 运算符有这些选择： `contains`（包含）、 `endsWith`（字符串尾部字符是……）、 `equals`（等于）、 `equalsIgnoreCase`（等于，不区分大小写）、 `isEmpty`（字符串是否为空）和 `startsWith`（字符串开头字符是……）。

<表达式> - 用于指定变量的输出结果。String的意思是字符串，所以表达式的取值范围取决于变量类型。

示例： `/qa conditions add 条件名称 Name equals Tom`（设定某条件为检测玩家ID是否为Tom）- 该条件中Name为String变量。

## 默认条件

这部分讲述的是默认条件，这些条件是“独立”存在的，不依附于变量。

## 🧠 条件|Condition

[!WARM] 该条件已于4.13.0版本被移除。取而代之的是Boolean变量 `Condition`，可以提供更多的功能。

>[!NOTE|label:描述]

满足需求：其它条件均满足。没错，这个条件仅仅会检查其下属在 `conditions.yml` 里的另外一些条件是否满足。

命令参数：

+ <其它条件名称> - 用于指定玩家需要满足的其它条件。

示例： `/qa conditions add 条件名称 Condition 其它条件名称`（设定某条件为满足名叫“其它条件名称”的条件）

## 🌐 世界事件|WorldTime

[!NOTE|label:描述] 满足需求：玩家所处的世界时间处于某个时间段内。

命令参数：

- <最小时间> - 用于指定时间段的最小时间（24小时制）。
- <最大时间> - 用于指定时间段的最大时间（24小时制）。 示例： `/qa conditions add 条件名称 WorldTime 11 20`（设定某条件为玩家当前所处世界的时间为上午11点至下午8点之间）

## 📅 日期|Date

[!NOTE|label:描述] 满足需求：当前日期处于指定日期，可以通过操作符指定在日期之前，还是之后。

命令参数：

- <日期操作符> - 可以通过使用比较操作符将当前日期和指定日期进行比较。操作符有： `before`（之前）和 `after`（之后）。比如：想要指定在2023年之后，那么就应当使用 `after`。
- (flags) - 可选Flag
  - `--year <年>` - 用于指定具体哪年。
  - `--month <月>` - 用于指定具体哪月。
  - `--day <日>` - 用于指定具体哪日。
  - `--hours <时>` - 用于指定具体是当天几时。
  - `--minutes <分>` - 用于指定具体几分（需要指定几时才能指定几分）。
  - `--seconds <秒>` - 用于指定具体几秒（需要指定几分才能指定几秒）。
  - `--timeZone <时区名称>` - 用于指定该指定日期的时区名称。 示例：
    - `/qa conditions add 条件名称 Date after --year 2022 --timeZone Europe/Berlin`（设定某条件为当前处于2022年之后，时区为欧洲/柏林时间）- 设定一个条件，需要当前处于2022年后才满足，时区为欧洲/柏林时间
    - `/qa conditions add 条件名称 Date after --month 11`（设定某条件为当前处于11月份之后）- 季节性条件！每年12月才能满足
    - `/qa conditions add 条件名称 Date before --month 11 --year 2022`（设定某条件为当前处于2022年11月份之前）- 设定截至日期，直至2022年10月底，在那之后将不再满足条件

## 特殊默认条件

### 🏆 完成目标|CompletedObjective

[!NOTE|label:描述] 这个条件在使用时需要附上一个目标。和其它条件可能不一样，这个条件会直接绑定在现有任务上。  
满足需求：玩家需要完成当前任务指定的目标。

<目标ID> - 用于指定需要完成的目标ID。想要查询目标ID，需要使用 `/qa edit 任务名称 objectives list` 命令。

示例： `/qa edit 任务名称 objectives edit 1 conditions add CompletedObjective 2` （设定某任务的目标2解锁条件为完成目标1）

powered by GitbookFile Modify: 2023-10-04 17:26:18



## ✎ 事件

可以使用 `/qa actions` 命令创建原始事件。这些事件都会保存在 `plugins/notquests/categoryname/actions.yml`。事件也可以作为任务奖励或者目标奖励。事件（**Action**），顾名思义，就是会“发生”的一些事情。

你可以通过 `/qa actions edit actionname execute <玩家ID(可选)>` 命令“测试”你通过 `/qa actions` 命令创建的条件。如果你想无视这个事件下的条件限制，你可以使用一个可选 Flag: `--ignoreConditions`

所有事件通用命令参数：

- **(flags)** - 可选Flag
- - `--category <类别名称>` - 如果是一个原始事件，储存于 `actions.yml` 里的事件，你可以在这里为其设定类别。
  - `--delay <delay>` - 如果是一个原始事件，储存于 `actions.yml` 里的事件，你可以在这里为其设定执行延迟。比如：如果输入 **1s**，那么事件将会在1秒后执行，而不是立即执行。

## 默认变量事件

这些事件将会充当变量的“容器”，这是啥意思呢？

好吧.....举个例子，**Number**事件就是这样的变量事件。在游戏中，它会将**Number**变量像很多事件的那样，转化为相应的事件，所以事件的数量实际上比你所认为的要得多。然而，有些变量适用于条件，而不适用于事件。例如，**DayOfWeek**变量就只适用于条件，而不适用于事件。

有道理吧？在现实生活中，你不能单纯靠一个Minecraft的插件就做到穿越时空。不过，“**Money**”这个变量既适用于条件，也适用于事件。因为你可以查看玩家的金钱，也可以改变他们的金钱。当然，我说的只是游戏中的金钱☹

更多请参阅变量部分。

## ？ 布尔|Boolean

[!NOTE|label:描述] 发生事件：Boolean的取值只有2种：True和False。换言之就是表达Yes和No。

命令参数：

- <变量类型>** - 用于指定Boolean变量的类型。
- <变量参数>** - 有些变量类型可能存在附加参数，也可能不存在。
- <运算符>** - 可用运算符：`equals`（等于） **<表达式>** - 用于指定变量的输出结果。常见值为 `true` 和 `false`，但你也可以在这里设定一个不同的Boolean变量。

示例：

- `/qa actions add 事件名称 Flying set true`（设定某事件为开启飞行）- 此条件中Flying为变量类型。
- `/qa actions add 事件名称 Money add ((TagInteger(TagName:points)>=4)*(10+30))+(!TagInteger(TagName:points)>=4)*5`（设定某事件为给予玩家金钱，如果玩家的整数标签“points”数值大于等于4，那么该事件将给予玩家40的金钱。否则，就给予5）

## 📄 列表|List

[!NOTE|label:描述] 发生事件：The value of this List variable is changed。命令参数：

- <变量类型>** - 用于指定List变量的类型。
- <变量参数>** - 有些变量类型可能存在附加参数，也可能不存在。
- <运算符>** - 可选操作符：`add`（增加）、`clear`（清空）、`remove`（删除）和 `set`（设置）。
- <表达式>** - 用于指定变量的输出结果。这取决于变量类型。

示例：`/qa actions add 事件名称 ActiveQuests add 任务名称`（设定某事件为强制给予玩家名叫“任务名称”的任务）- 该条件中ActiveQuests为List变量。

## 📄 ItemStackList

[!NOTE|label:描述] 发生事件：The value of this ItemStackList variable is changed。命令参数：

- <变量类型>** - 用于指定ItemStackList变量的类型。
- <变量参数>** - 有些变量类型可能存在附加参数，也可能不存在。
- <运算符>** - 可选操作符：`add`（增加）、`clear`（清空）、`remove`（删除）和 `set`（设置）。
- <表达式>** - 用于指定变量的输出结果。这取决于变量类型，但必须是物品。

示例：`/qa actions add 事件名称 Inventory remove hand 3` - 该条件中Inventory为ItemStackList变量。

## 100 数字|Number

[!NOTE|label:描述] 发生事件：The value of this Number variable is changed。命令参数：

- <变量类型>** - 用于指定Number变量的类型。
- <变量参数>** - 有些变量类型可能存在附加参数，也可能不存在。
- <运算符>** - 可选操作符：`add`（增加）、`deduct`（减去）、`divide`（除以）、`multiply`（乘以）和 `set`（设置）。
- <表达式>** - 用于指定变量的输出结果。此处的表达式很强大，因为你可以在这里输入任何类型的数学表达式，甚至可以包括其它Number变量。

示例：`/qa actions add 事件名称 Money multiply 2`（设定某事件为玩家金钱乘以2）- 该事件中Money为Number变量。再举个包含了数学表达式和其它变量的例子：`/qa actions add 事件名称 Money set QuestPoints*Money+500-30/2`（设定某事件为设置玩家金钱=任务点数×金钱+500-30÷2这个表达式的运算结果）- 该条件中Money

为Number变量。再举个带有非常高级的表达式的例子：`/qa actions add 条件名称 Money set 10+TagInteger(TagName:reputation)*TagInteger(TagName:level)`（设定某事件为设置玩家的金钱为10+整数标签“好感度”数值×整数标签“等级”数值）- 该条件中Money为Number变量。

## AB 字符串|String

[!NOTE|label:描述] 发生事件：The value of this String variable is changed。命令参数：

<变量类型> - 用于指定String变量的类型。

<变量参数> - 有些变量类型可能存在附加参数，也可能不存在。<运算符> - 可选操作符：`set`（设置）和 `append`（句末加字符串）。

<表达式> - 用于指定变量的输出结果。这取决于变量类型，但必须是字符串（即文本）。

示例：`/qa actions add 事件名称 CurrentWorld set world`（设定某事件为将玩家传送到世界world）- 该条件中CurrentWorld为String变量。

## 默认事件

这部分讲述的是默认事件，这些事件是“独立”存在的，不依附于变量。

## ☹️ 事件|Action

[!NOTE|label:描述] 发生事件：执行其它一个或多个事件。没错，这个事件会执行其下属在actions.yml里的其它事件。

命令参数：

- <其它事件名称> - 用于指定其它会一同执行的事件。
- <数量> - （可选）用于指定该事件的执行次数。
- (flags) - 可选Flag
- - `--ignoreConditions` - 若设置了此Flag，那么被附加到其下属的其它事件执行时所需的条件都会被无视，无论其下属的事件需要什么样的条件，都会强制执行事件。
  - 
  - `--minRandom <数量>` - 若设置了此Flag，那么被附加到其下属的其它事件将会被随机执行，此Flag用于指定随机执行的最少数量。
  - 
  - `--maxRandom <数量>` - 若设置了此Flag，那么被附加到其下属的其它事件将会被随机执行，此Flag用于指定随机执行的最多数量。
  - 
  - `--onlyCountForRandomIfConditionsFulfilled` - 若设置了此Flag，那么只有满足条件的事件会被随机执行所选中。
  - 
  - `--executedActionDelay` - 若设置了此Flag，那么其下属的事件均会出现执行延迟。这可能会覆盖现有的延迟。

示例：

- `/qa actions add 事件名称 Action a1 2 --ignoreConditions`（设定某事件为执行名叫“a1”的事件2次，无视事件a1的条件）
- `/qa actions add 事件名称 Action a1,a2,money10 1`（设定某事件为执行名叫“a1”、“a2”、“money10”的事件1次）带随机事件Flag的示例：我事先创建了事件sm1、sm2、sm3、.....、sm10，他们都是不同的SendMessage事件。
- `/qa actions add sendAllMessages Action sm1,sm2,sm3,sm4,sm5,sm6,sm7,sm8,sm9,sm10 1`（设定sendAllMessages事件为依照顺序依次执行事件sm1、sm2、.....sm10）
- `/qa actions add sendAllMessagesInRandomOrder Action sm1,sm2,sm3,sm4,sm5,sm6,sm7,sm8,sm9,sm10 1 --minRandom 10 --maxRandom 10`（设定sendAllMessagesInRandomOrder事件为从10个事件中随机抽选最少10个、最多10个事件执行）
- `/qa actions add sendRandomMessage Action sm1,sm2,sm3,sm4,sm5,sm6,sm7,sm8,sm9,sm10 1 --minRandom 1 --maxRandom 1`（设定sendRandomMessage事件为从10个事件中随机抽选最少1个、最多1个事件执行）
- `/qa actions add sendGuaranteedRandomMessage Action sm1,sm2,sm3,sm4,sm5,sm6,sm7,sm8,sm9,sm10 1 --minRandom 1 --maxRandom 1 --onlyCountForRandomIfConditionsFulfilled`（设定sendGuaranteedRandomMessage事件为从10个事件中随机抽选最少1个、最多1个事件执行，只有满足条件的事件会被抽选，若玩家并不满足某个事件的条件，那么该事件就不会被纳入抽选范围内）
- `/qa actions add sendOneOrTwoRandomMessage Action sm1,sm2,sm3,sm4,sm5,sm6,sm7,sm8,sm9,sm10 1 --minRandom 1 --maxRandom 2`（设定sendOneOrTwoRandomMessage事件为从10个事件中随机抽选最少1个、最多2个事件执行）

## i Beam

## i Boolean

## i BroadcastMessage

## i Chat

## i CompleteQuest

## i ConsoleCommand

## i FailQuest

 **GiveItem**

 **GiveQuest**

 **PlayerCommand**

 **PlaySound**

 **SendMessage**

 **SpawnMob**

 **StartConversation**

 **TriggerCommand**

**BetonQuest**联动事件

 **BetonQuestFireEvent**

 **BetonQuestFireInlineEvent**

powered by GitbookFile Modify: 2023-10-04 18:41:11

## 📁 触发器

触发器可以附加在任务上。触发器基本上是在某些事情发生时触发（执行）的事件。

例如，如果在一个任务中创建一个 `DEATH` 触发器，那么玩家一旦死亡，该触发器就会被激活，随后，该触发器所绑定的事件将会执行。

## 如果触发器被激活会发生什么？

如果触发器被激活（在上面这个例子中，玩家死亡即激活），将会执行一个事件。你可以通过 `/qa actions` 命令事先创建事件。

## 触发器的高级选项

触发器还有更高级的选项。你不仅可以设定在玩家死亡时触发（`DEATH` 触发器），还可以设定在玩家死亡时目标3处于已解锁状态才能触发触发器。再或者，设定再玩家在特定世界中死亡时才触发。

## 事件

事件可以通过 `/qa actions` 命令进行创建，事件创建后，可以重复使用。因此，这些事件实际上可以用于多个触发器上。

## 实践示例

让我们先创建一个任务失败的事件`failQuest`：`/qa actions add failQuest FailQuest test`。

然后，我们再在一个示例任务`test`上创建一个触发器，并将其绑定在我们的事件上：`/qa edit test triggers add failQuest DEATH Quest ALL 1`。

至此，目前的情况是，当你激活了这个任务后，一旦死亡，任务就会失败。这里的 `failQuest` 就是我们的事件名称，而 `DEATH` 就是触发器类型（`TriggerType`）。

## 详细解释一下示例中的添加触发器命令

后半部分参数`"Quest"`用于指定触发器激活时机。`"Quest"`表示任务需要处于激活状态触发器才可以被激活。如果设定为`"O1"`（目标1）而不是`"Quest"`的话，那么只有在目标1处于激活状态时，触发器才可以被触发。在我们给出的示例中，并无大碍，因为目标1始终处于激活状态。

但是如果任务有着两个目标的话，您可以为其设置依赖关系，让目标2依赖于目标1。如此一来，只有在达成目标1之后，目标2才会显示。在这个任务中，如果将这个触发器设定为`"O2"`（目标2），则达成目标2后触发器才会触发（当然还得先达成目标1）。相当灵活，对吧？

下一个参数`"ALL"`，是在指定适用的世界。如果设定为`"world_nether"`，则需要在下界死亡才能触发触发器。最后一个参数用于指定触发器会在多少次死亡后激活。

powered by GitbookFile Modify: 2023-10-04 16:54:47

## ★ 变量

变量还可以用于快速创建高级事件和条件！正因如此，下面的很多变量都可以运用于条件当中，并且大多数情况也可以运用于事件当中。不仅如此，变量还可以在一些所谓的表达式中运用。因此，你甚至可以对这样变量进行计算！

## 默认变量

### ? ActiveQuests

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串、列表 条件示例：

- `/qa conditions add isWoodCutterQuestActive ActiveQuests contains woodCutter` （设定条件isWoodCutterQuestActive为名叫“woodCutter”的任务处于激活状态）
- `/qa conditions add isOnlyWoodCutterQuestActive ActiveQuests equals woodCutter` （设定条件isOnlyWoodCutterQuestActive为名叫“woodCutter”的任务处于激活状态，且除此之外没有其它任务）
- `/qa conditions add hasAllWoodQuestsActive ActiveQuests contains woodCutter,saveTheWoods` （设定条件hasAllWoodQuestsActive为名叫“woodCutter”和“saveTheWoods”的任务同时处于激活状态） 事件示例：
- `/qa actions add giveQuestWoodCutter ActiveQuests add woodCutter` （设定事件giveQuestWoodCutter为强制给予玩家名叫“woodCutter”的任务）
- `/qa actions add setQuestWoodCutter ActiveQuests set woodCutter` （设定事件setQuestWoodCutter为强制给予玩家名叫“woodCutter”的任务，同时放弃玩家其它所有任务）
- `/qa actions add giveForestQuests ActiveQuests add woodCutter,saveTheWoods` （设定事件giveForestQuests为强制给予玩家名叫“woodCutter”和“saveTheWoods”的任务）

### ? Advancement

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：布尔值

### ? Block

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串

### ? Chance

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：布尔值

### ? Climbing

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：布尔值

### ? CompletedObjectiveIdsOfQuest

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串、列表

### ? CompletedQuests

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串、列表

### ? Condition

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：布尔值

### ? ContainerInventory

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串、列表

### ? CurrentBiome

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串

### ? ContainerInventory

[!NOTE|label:描述] 条件：☒ 事件：☒ 类型：字符串、列表

## ? CurrentPositionX

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? CurrentPositionY

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串、列表

## ? CurrentPositionZ

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串、列表

## ? CurrentWorld

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

## ? DayOfWeek

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

## ? EnderChest

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: ItemStack、列表

## ? Experience

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## ? ExperienceLevel

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## ? False

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Flying

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 单精度浮点数

## ? FlySpeed

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串、列表

## ? GameMode

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

## ? Glowing

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Health

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? InLava

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? InWater



[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Inventory

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: ItemStack、列表

## ? ItemInInventoryEnchantments

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串、列表

## ? MaxHealth

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? Money

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? Name

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

## ? Op

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Permission

[!NOTE|label:描述] 条件: ☒ 事件: ☒ (需要LuckPerms前置) 类型: 布尔值

## ? Ping

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## ? PlaytimeTicks

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## ? PlaytimeMinutes

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? PlaytimeHours

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? QuestAbleToAccept

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? QuestOnCooldown

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? QuestPoints

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 长整型数

## ? QuestReachedMaxAccepts

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? QuestReachedMaxCompletions

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? QuestReachedMaxFails

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? RandomNumberBetweenRange

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## ? ReflectionStaticBoolean

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? ReflectionStaticDouble

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

## ? ReflectionStaticFloat

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 单精度浮点数

## ? ReflectionStaticInteger

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## ? ReflectionStaticString

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

## ? Sleeping

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Sneaking

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Sprinting

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? Swimming

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? True

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## ? WalkSpeed

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 单精度浮点数

## BetonQuest联动变量

## ? BetonQuestCondition

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

## PlaceholderAPI联动变量

### ? PlaceholderAPINumber

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 双精度浮点数

### ? PlaceholderAPIString

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

## Project Korra联动变量

### ? ProjectKorraElements

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串、列表

### ? ProjectKorralsBender

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

### ? ProjectKorraSubElements

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串、列表

## Towny联动变量

### ? TownyNationName

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 字符串

### ? TownyNationTownCount

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

### ? TownyTownPlotCount

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

### ? TownyTownResidentCountVariable

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## UltimateClans联动变量

### ? UltimateClansClanLevel

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 整数

## Floodgate联动变量

### ? FloodgateIsFloodgatePlayer

[!NOTE|label:描述] 条件: ☒ 事件: ☒ 类型: 布尔值

powered by GitbookFile Modify: 2023-10-04 19:06:11



## BetonQuest

你可以在BetonQuest里使用NotQuests的所有条件和事件。同时，你也可以在NotQuests里使用BetonQuest的所有事件、条件和目标。仅支持BetonQuest 2.x及以上版本

### ♥ 在BetonQuest里使用NotQuests对话拦截器

效果展现: <https://youtu.be/uKFuSV1CLFo>

使用方法: 在BetonQuest config里使用 `default_interceptor: notquests`。同时，还需要确保你的 `default_conversation_IO`: 设定的是 `menu` 或者 `simple`。

### ♥ 在NotQuests里使用BetonQuest事件

示例: `/qa actions add 事件名称 BetonQuestFireEvent default tag_wood_done`

或

示例: `/qa actions add 事件名称 BetonQuestFireInlineEvent tag add wood_done`

### ♥ 在NotQuests里使用BetonQuest条件

示例: `/qa conditions add 条件名称 BetonQuestCondition default wood_done equals true`

### ♥ 你也可以在BetonQuest事件里使用NotQuests事件

- `nq_action` <事件类型> <该NotQuests事件的内联参数> - 内联参数后续将在相应的类型页面中进行文档化。
- - 示例1: `nq_action Beam test world -15140 85 2234`
  - 示例2: `nq_action Beam test -15140;85;2234;world`
- `nq_triggerobjective` <触发器名称>
- `nq_startquest` <任务名称> (可选flag: `-force -silent -notriggers`)
- `nq_failquest` <任务名称>
- `nq_abortquest` <任务名称> - 如果该任务处于激活状态，这只是移除玩家的任务，不算做失败。
- `nq_questpoints` <set/add/remove> <数量> (可选flag: `-silent`)

### ♥ 你也可以在BetonQuest条件里使用NotQuests条件

- `nq_condition` <条件类型> <该NotQuests条件的内联参数>

powered by GitbookFile Modify: 2023-10-04 19:46:14

## 权限

### 玩家权限

- **notquests.use** - 这是默认用户命令权限，玩家需要使用它才能玩你制作的任務。該权限默认启用。如果你不想让玩家玩你做的任务，请移除此权限。
- **notquests.user.profiles** - 如果你希望玩家自行创建和使用不同的个人资料（`/nq profiles`），此权限默认为禁用。

### 管理员权限

- **notquests.admin.armorstandeditingitems** - 对盔甲架使用任务编辑物品的权限，该权限应仅赋予管理员使用。
- **notquests.admin** - 管理员命令，通常管理员都需要使用 `/qa` 命令来执行管理员操作，例如创建、编辑任务。该权限自动包含上述所有权限，你可以使用此权限进行所有插件操作。

powered by GitbookFile Modify: 2023-10-04 19:58:32

## 📄 占位符

### 玩家占位符

这些是你可以使用的所有玩家占位符（需要安装[PlaceholderAPI](#)）：

- %notquests\_player\_has\_completed\_quest\_QUESTNAME%
- %notquests\_player\_has\_current\_active\_quest\_QUESTNAME%
- %notquests\_player\_is\_objective\_unlocked\_and\_active\_OBJECTIVEID\_from\_active\_quest\_QUESTNAME%
- %notquests\_player\_is\_objective\_unlocked\_OBJECTIVEID\_from\_active\_quest\_QUESTNAME%
- %notquests\_player\_is\_objective\_completed\_OBJECTIVEID\_from\_active\_quest\_QUESTNAME%
- %notquests\_player\_questpoints%
- %notquests\_player\_active\_quests\_list\_horizontal%
- %notquests\_player\_active\_quests\_list\_vertical%
- %notquests\_player\_completed\_quests\_amount%
- %notquests\_player\_active\_quests\_amount%
- %notquests\_player\_tag\_TAGNAME%
- %notquests\_player\_variable\_VARIABLENAME%
- %notquests\_player\_expression\_EXPRESSION%
- %notquests\_player\_rounded\_expression\_EXPRESSION%
- %notquests\_player\_quest\_cooldown\_left\_formatted\_QUESTNAME%
- %notquests\_player\_objective\_progress\_OBJECTIVEID\_from\_active\_quest\_QUESTNAME%
- %notquests\_player\_objective\_progress\_percentage\_OBJECTIVEID\_from\_active\_quest\_QUESTNAME% 应用时，需要将占位符中的 QUESTNAME 换作任务名称，将 OBJECTIVEID 换作目标的ID（可在 /qa edit questname objectives list 中查看），因此，占位符中的大写字母部分都需要进行替换，诸如此类，还有 TAGNAME 要换作标签名称， VARIABLENAME 要换作变量名称， EXPRESSION 要换作表达式。

powered by GitbookFile Modify: 2023-10-04 20:09:56

## 从v3及以下版本升级

在NotQuests的4.x版本中，我对条件（包括前置）、事件（包括奖励）和整个文件夹结构（类别的原因）进行了诸多更改。为此编写一个自动转换器什么的既困难又耗时。这就是为什么早期版本的NotQuests与4.x不完全兼容。

### 如何升级

#### 最简单的方法：清除安装

最简单的方法当然就是将服务器关闭，然后删除 `plugins/NotQuests` 文件夹，下载新版本的NotQuests插件，然后重新启动服务器。这样一来，你会丢失所有的进度，不过这样更新非常地干净，不会出现什么问题。尽管如此，如果你像保留之前的对话的话，你可以看看下方较难的步骤3，这非常简单，只需要移动文件夹即可。

#### 较难的方法：尝试尽可能地保留一些数据

1. 关闭服务器并删除 `plugins/NotQuests/translations` 文件夹。
2. 升级插件.jar文件，启动服务器，然后再次关闭。
3. 移动 `plugins/NotQuests/conversations` 文件夹到 `plugins/NotQuests/default/conversations` 。这样一来，就可以保留先前的对话。
4. 移动 `plugins/NotQuests/quests.yml` 文件到 `plugins/NotQuests/default/quests.yml` ，然后打开`quests.yml`文件，清除文件中所有奖励/前置（事件/条件）内容。如果觉得太难了，可以直接删除`quests.yml`，然后再启动再关闭服务器。
5. 启动服务器，重新创建所有事件和条件。大多数事件和条件在此更新中已被破坏，我不觉得这值得我去尝试修复。 `plugins/NotQuests/actions.yml` 和 `plugins/NotQuests/conditions.yml` 可以删除，因为插件只会读取 `plugins/NotQuests/default` 文件夹中的文件。
6. 服务器启用时查看控制台是否有警告或错误。如果有，请尝试阅读并解决他们。执行到第4步的时候困难会发生这种情况。

powered by GitbookFile Modify: 2023-10-04 20:35:13



## 💬 对话系统

原文称该教程已严重过时，故不再翻译，若感兴趣可以去看原文：<https://www.notquests.com/docs/documentation/conversation-system>

powered by GitbookFile Modify: 2023-10-04 23:30:10

## ? 常见问题

HELP! 我的所有奖励都显示[已隐藏]

默认情况下，奖励都会处于隐藏状态，除非你为它们添加一个显示名称，需要使用命令 `/qa edit [任务名称] rewards edit [奖励ID] 显示名称`，显示名称将会替换掉[已隐藏]。  
这样一来你就可以更好地指定奖励是些什么

我需要弄一个MySQL数据库吗？

不！普通的SQLite数据库（将会在你的 `plugins/NotQuests` 文件夹里创建）就可以支撑插件的正常工作吗。不过，MySQL更快，我也更推荐。数据库查询的设计也是适配MySQL的。

如何将玩家数据从SQLite迁移到MySQL，或者二者相互迁移？

我建议你通过DBEaver或类似的工具手工操作。这并不难，因为不管是SQLite还是MySQL，NotQuests都使用着相同的SQL查询。还有一种目前处于实验性阶段的自动化迁移方式。我并不能保证其是否生效，因此请在那之前做好数据备份：

1. 在general.yml设定load-playerdata-on-join和save-playerdata-on-quit为false
2. 使用旧数据库启动服务器
3. 添加新数据库到general.yml
4. 输入/qa reload命令（可能不需要）
5. 输入/qa debug loadDataManagerUnsafe命令
6. 关闭服务器
7. 撤销你在第一步所做的操作

powered by GitbookFile Modify: 2023-10-04 23:57:23

## API使用方法

### 添加API到你的项目中

#### Gradle Kotlin DSL（推荐）

添加这个到你的build.gradle:

```
repositories{
    maven {
        url = "https://maven.pkg.github.com/alessiogr/NotQuests"
        content{
            includeGroup("rocks.gravili.notquests")
        }
        credentials {
            username = System.getenv("GITHUB_PACKAGES_USERID") ?: "alessiogr"
            password = System.getenv("GITHUB_PACKAGES_IMPORT_TOKEN") ?: "ghp_o40cknVScvIXSl3jeKRrF0Rw4Kaagf4C72F4"
        }
    }
}

dependencies {
    compileOnly 'rocks.gravili.notquests:paper:5.17.1'
}
```

注意: 请确保你使用的是最新版本的API。

注意: 上述所写的Github Packages令牌将不再起作用。因为每次我重新搞一个令牌Github都会撤销掉它。你可以自己在Github上生成自己的令牌并使用它, 或是直接将NotQuests jar添加到你的项目中。

完成! 如果你想用自己的Github用户ID或是令牌, 请随意。

## Paper和Spigot模块

安装完API后, 你会发现, 我们有两个几乎相同的模块: Paper和Spigot。请使用Paper模块, 而不是Spigot。这里的Spigot模块仅仅是为了遵守spigot.org资源指南而存在, 一旦spigot.org噶了就会被删除, 且由Hangar替代。

我将不再会Spigot模块进行任何维护和更新, 请使用Paper模块。

为什么?

因为Spigot API很古老, 并且还使用着很多早就被遗弃的上古时期的特性。NotQuests可是一个想要跟上时代的插件, 使用着现代化的API特性, Spigot不仅没有还拒绝添加。(更重要的一点是连Kyori components, 甚至是Bukkit.getTPS())这样简单的东西都没有)

## 使用API

首先, 将 NotQuests 在你的plugin.yml文件中添加为 depend: 或是 softdepend: 。

然后, 使用 NotQuests.getInstance() 访问实例 (使用Paper模块的NotQuests), 而后根据需要进行操作。☺

请确保在NotQuests.getInstance()为空时禁用你的NotQuests integration。由于你希望使用的是Paper模块, 它在Spigot服务器上会返回空值。

## 注册你自己的目标

你可以轻松地将目标添加到NotQuests中 (直接添加或通过API), 示例:

```
NotQuests.getInstance().getObjectiveManager().registerObjective("jumpobjective", JumpObjective.class);
```

使用JumpObjective继承"Objective", 与其它目标类相似。只需复制结构即可。

powered by GitbookFile Modify: 2023-10-05 22:08:03

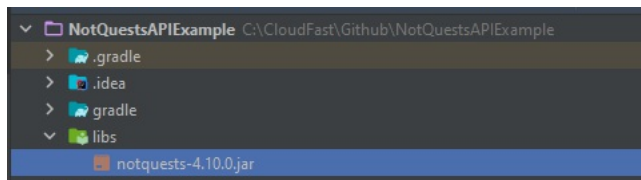
## API示例项目教程

[!WARN][label: 阅前须知] 此教程基于插件版本5.17.1，Paper版本1.20.1编写的。

让我们用NotQuests API创建我们的第一个项目吧！

### 步骤1：添加NotQuests API到你的项目

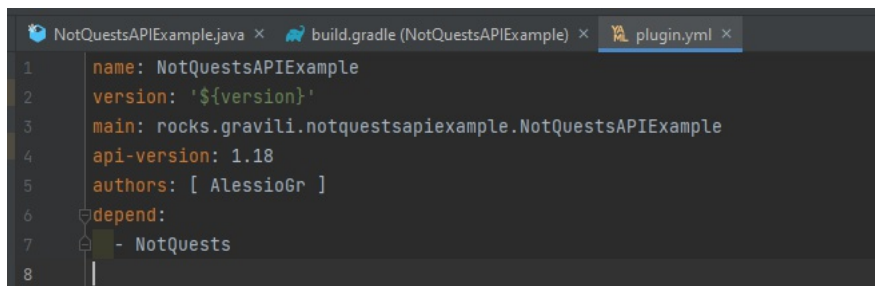
首先，在你的项目中先创建一个libs文件夹，并将NotQuests jar文件放入其中。你还可以使用我们的[GitHub packages](#)仓库，不过它目前还无法很好地运作，因为你需要生成你自己的[Github](#)密钥且无法公开。至于[JitPack](#)，它们不支持[Java 17](#)。



如果你使用gradle作为构建工具（你就该这么做），那么请打开您的 `build.gradle` 文件，在你的dependencies部分添加这些内容：

```
dependencies {
    compileOnly 'io.papermc.paper:paper-api:1.20.1-R0.1-SNAPSHOT'
    compileOnly files('libs/notquests-5.17.1.jar')
}
```

接下来，打开你的plugin.yml文件，并将NotQuests添加为depend或softdepend。在示例中，我们将其添加到了depend中：

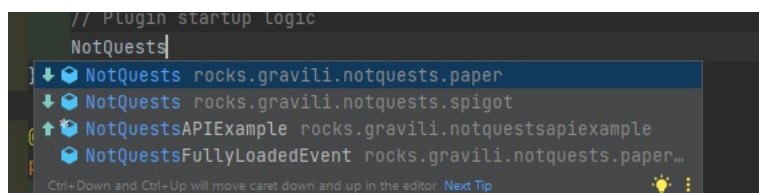


### 步骤2：获取NotQuests实例

NotQuests有一个适用于Spigot的模块和一个适用于Paper的模块。二者大相径庭，Spigot模块非常古老。在本教程中，我们将只使用Paper模块来创建我们的目标、条件和事件。只有在使用Paper核心的服务器上，它们才能正常运行，而在Spigot上，Paper模块的NotQuests.getInstance()会返回空值，请在注册任何内容前都检查一下。

无论如何你都不应该使用Spigot模块，不过不用担心，你的插件依旧可以在Spigot服务器上加载。

请确保只使用来自Paper模块的类：



好了，然后我们在主类里创建一个NotQuests实例：

```
public final class NotQuestsAPIExample extends JavaPlugin {

    private NotQuests notQuestsInstance;

    @Override
    public void onEnable() {
        // Plugin startup logic
        notQuestsInstance = NotQuests.getInstance();
    }

    @Override
    public void onDisable() {
        // Plugin shutdown logic
    }

}
```

NotQuests几乎没有使用任何静态内容，所以我们将会在几乎所有地方使用实例。

## 通过变量创建一个事件和条件（=前置）

在NotQuests中，你可以分别创建条件（Conditions）和事件（Actions）。不过，你也可以使用变量（Variable）来同时创建二者！

能用到变量的话尽可能地使用变量。只有在使用变量无法做出你想要的功能时，再考虑直接注册条件和事件。

我们来给玩家的饥饿值（Food Level）创建一个变量。创建一个名叫“FoodLevelVariable”新类，并使其继承Variable。饥饿值的取值范围是整数，而Variable类使用泛型（Generics）。

然后，如果你的IDE没有出现问题，可以使所有必要的机制都生效的话，应该是这个样子：

```
public class FoodLevelVariable extends Variable<Integer> {
    private final NotQuests main;

    public FoodLevelVariable(NotQuests main) {
        super(main);
        this.main = main;
    }

    @Override
    public Integer getValue(QuestPlayer questPlayer, Object... objects) {
        return null;
    }

    @Override
    public boolean setValueInternally(Integer newValue, QuestPlayer questPlayer, Object... objects) {
        return false;
    }

    @Override
    public List<String> getPossibleValues(QuestPlayer questPlayer, Object... objects) {
        return null;
    }

    @Override
    public String getPlural() {
        return null;
    }

    @Override
    public String getSingular() {
        return null;
    }
}
```

All we need to do is fill out the each and every method. Let's start with getValue. This method will be used for the Condition which is generated from this variable. Here, we need to return the player's current food level. Pretty simple:

```
@Override
public Integer getValue(QuestPlayer questPlayer, Object... objects) {
    return questPlayer.getPlayer().getFoodLevel();
}
```

Next, in setValueInternally, that's what's used internally for the action. By default, a variable only needs the getValue method filled out. Only SOME variables can also change the value. It works here, so let's fill it out:

```
@Override
public boolean setValueInternally(Integer newValue, QuestPlayer questPlayer, Object... objects) {
    questPlayer.getPlayer().setFoodLevel(newValue);
    return true;
}
```

Always return true there if the setting-of-the-value is successful. Now we need to enable setting the value in the constructor. Otherwise, the corresponding action will not be generated:

```
public FoodLevelVariable(NotQuests main) {
    super(main);
    this.main = main;
    setCanSetValue(true);
}
```

As for getPossibleValues, let's leave it at null. NotQuests will just use the default integer auto-completion there. For getSingular and getPlural, use this:

```
@Override
public String getPlural() {
    return "Food level";
}

@Override
public String getSingular() {
    return "Food levels";
}
```

And we're done! Now just register this variable in the onEnable method in your Main:

```
@Override
public void onEnable() {
    // Plugin startup logic
    notQuestsInstance = NotQuests.getInstance();
    if(notQuestsInstance != null){ //For Spigot compatibility
        notQuestsInstance.getVariablesManager().registerVariable("FoodLevel", FoodLevelVariable.class);
    }
}
```

Done! Let's see how it looks in-game. During startup, you should be able to see this "Registering Variable" line in the console:

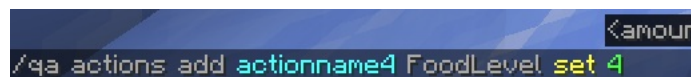
```
[NotQuestsAPIExample] Disabling NotQuestsAPIExample v1.0
[NotQuestsAPIExample] Enabling NotQuestsAPIExample v1.0
[NotQuests]: Registering Variable FoodLevel
[NotQuests]: Re-registering condition Number due to variable changes...
[NotQuests]: Registering number condition: FoodLevel
[NotQuests]: Registering number condition: FoodLevel
[NotQuests]: Registering number condition: FoodLevel
[NotQuests]: Registering number condition: FoodLevel
[NotQuests]: Re-registering condition ItemStackList due to variable changes...
[NotQuests]: Re-registering condition List due to variable changes...
[NotQuests]: Re-registering condition String due to variable changes...
[NotQuests]: Re-registering condition Boolean due to variable changes...
```

这是创建条件的方法:



```
/qa conditions add conditionname FoodLevel moreThan 100
```

还有事件:



```
/qa actions add actionname4 FoodLevel set 4
```

你还可以通过 /qa actions edit actionname4 execute 命令执行:



For many simple values you can use the variable system. Not only is it easier, it also gives you access to the advanced comparison operators (like being able to use Math and other variables in the expression for Integer variables).

## 创建一个目标

Create the class called `TakeDamageObjective` and make it extend `Objective`. Then implement everything. It should look like this:

```
public class TakeDamageObjective extends Objective {
    private final NotQuests main;

    public TakeDamageObjective(NotQuests main) {
        super(main);
        this.main = main;
    }

    @Override
    public String getObjectiveTaskDescription(QuestPlayer questPlayer) {
        return null;
    }

    @Override
    public void save(FileConfiguration fileConfiguration, String initialPath) {
    }

    @Override
    public void load(FileConfiguration fileConfiguration, String initialPath) {
    }

    @Override
    public void onObjectiveUnlock(ActiveObjective activeObjective, boolean unlockedDuringPluginStartupQuestLoadingProcess) {
    }

    @Override
    public void onObjectiveCompleteOrLock(ActiveObjective activeObjective, boolean lockedOrCompletedDuringPluginStartupQuestLoadingProcess, boolean completed) {
    }
}
```

Let's already register it in our `onEnable` in our Main:

```
@Override
public void onEnable() {
    // Plugin startup logic
    notQuestsInstance = NotQuests.getInstance();
    if(notQuestsInstance != null){ //For Spigot compatibility
        notQuestsInstance.getVariablesManager().registerVariable("FoodLevel", FoodLevelVariable.class);
        notQuestsInstance.getObjectiveManager().registerObjective("TakeDamage", TakeDamageObjective.class);
    }
}
```

Now back to our `TakeDamageObjective` , just fill out each method. You can see how other Objectives do it [here](#).

Then, you'll need to register and handle your own Bukkit events to add Progress (and eventually complete) your objective. For the internal objectives, I'm doing that [here](#). Feel free to copy the boilerplate code.

I'll add a more explanatory tutorial on Objective Creation later, feel free to ask for help on our Discord. You can find the API example project [on GitHub](#). Not that it might not have been updated to the latest NotQuests API yet.

powered by GitbookFile Modify: 2023-10-05 23:18:46