# **[返回主页](index.html)**

# **语言文件使用教程**

## **介绍**

  你好！很高兴您愿意抽出一点时间来阅读此教程，以更好地模组开发。

  众所周知：

语言(英文：Language，法文：Langue )是人类进行沟通交流的表达方式。它是人与人之间的一种交流方式，人们彼此的交往离不开语言。尽管通过[图片](https://baike.baidu.com/item/%E5%9B%BE%E7%89%87)、[动作](https://baike.baidu.com/item/%E5%8A%A8%E4%BD%9C/33802)、[表情](https://baike.baidu.com/item/%E8%A1%A8%E6%83%85)等可以传递人们的思想，但是语言是其中最重要的，也是最方便的[媒介](https://baike.baidu.com/item/%E5%AA%92%E4%BB%8B/5378050)。然而世界各地的人们所用的语言各不相同，彼此间直接交谈是困难的，甚至是不可能的。即使是同一种语言，还有不同的方言，其差别程度也不相同。有的方言可以基本上相互理解，有的差别极大。

  我们早期的模组开发者早已意识到多语言的重要性，@SweelLong为主要语言文件编写者，并且随着tModLoader的不断更迭，我们不同时期的使用不同方式以本地化模组：纯代码，自制语言格式读取，json文件读取……

  好在，TML 1.4的发布，TML-Team加入了新的本地化方案{\*.hjson}，我们开发组，与时俱进，目前采纳了此方案。

## **文件类型**

JSON 是我们平时使用非常频繁的数据格式，尤其在网络传输中非常受欢迎。大部分编程语言都支持这种格式。但是作为配置文件还是有些许不足，最主要的一点是缺乏注释（其他比如低信噪比其实还可以接受），所以在 JSON 的基础上衍生出了更方便的作为配置文件的格式 HJSON 。

HJSON的写法还是毕竟多的，甚至可以直接模仿JSON的写法

而我们开发组 采用 类JSON写法

Xxx:

{

 Xx: “x”

 Xxx:

 {

  ...

}

}

我们称：

Xxx和xx为键名

x为键值

  调用HJSON值的方式类似调用命名空间：

HJSON文件如下：

Mods:

{

 InterspaceMod:

 {

   Test1: “测试1”

   Test2:

   {

     Test1: “测试1”

     Test2:

     {

     // 注释：禁止套娃

}

}

}

}

调用方式（键名的完全路径）：

输出InterspaceMod下的Test1：

>>> “Mods.InterspaceMod.Test1”

输出Test2下的Test1：

>>> “Mods.InterspaceMod.Test2.Test1”

......

## **TML本地化方案：**

# You may find this .hjson file approach to translations easier for your mod.

// 你会发现.hjson文件的方法让模组翻译更容易。

# .hjson files contain translations for the language specified in the filename. (Make sure this file is UTF-8 encoded only.)

// .hjson的文件夹名是指定的（确保文件编码只能为UTF-8）

# The possible languages are: English ("en-US"), German ("de-DE"), Italian ("it-IT"), French ("fr-FR"), Spanish ("es-ES"), Russian ("ru-RU"), Chinese ("zh-Hans"), Portuguese ("pt-BR"), and Polish ("pl-PL")

// 可能的语言是（引号内为指定的文件名）：英语("en-US"), 德语("de-DE"), 意大利语("it-IT"), 法语("fr-FR"), 西班牙语("es-ES"), 俄语("ru-RU"), 汉语("zh-Hans"), 葡萄牙语("pt-BR"), 波兰语("pl-PL")

# We have organized these files in a folder called Localization. A single file per language makes it extremely easy to to integrate and maintain translations.

// 我们将这些文件组织在一个名为本地化的文件夹中。每种语言只有一个文件，因此非常容易整理和维护翻译。

# Each translation entry in .hjson files contains a key and a value. The key is determined following the rules of JSON [https://www.w3schools.com/js/js\_json\_syntax.asp]

// 每个翻译条目.hjson文件都包含一个键名和一个键值。键是根据JSON的规则确定的[https://www.w3schools.com/js/js\_json\_syntax.asp]

# The Key automatically inherits "Mods.ModNameHere.", which is useful to know if you want to use substitutions (as seen in the Paper Airplane example) or use Language.GetTextValue or Network text.

// 键自动继承“Mods.模组名”，如果您想使用代码设置语言或使用Language.GetTextValu或网络文本是非常有用的。

# The following are the currently autogenerated keys: ItemName, ItemTooltip, BuffName, BuffDescription, ProjectileName, NPCName, MapObject, and Prefix

// 这些键是自动继承的（开发组要求不使用自动键，可忽略）：ItemName, ItemTooltip, BuffName, BuffDescription, ProjectileName, NPCName, MapObject, and Prefix

## **Localizer.cs**

  根据简洁码字的原则，我们引进一些方法可以省略”Mods.InterspaceMod.”，让我们调用的步骤代码，更美观，减少重复。

  这里储存着大量的调用模板，例如：

/// <summary>

/// Items/Weapons文件夹专用键

/// </summary>

/// <param name="keyName"></param>

/// <returns></returns>

internal static string WeaponsText(string keyName)

{

return Language.GetTextValue("Mods.InterspaceMod.Weapons.{0}", keyName);

}

根据第二行，我们发现这个方法允许Weapons文件夹下的类调用语言文本：

查阅语言文件（如zh-Hans.hjson）：如果要调用

 Weapons:

{

InterspaceSword: "星际天河之刃"

}

直接调用：Language.GetTextValue("Mods.InterspaceMod.Weapons.InterspaceSword");

方法调用：WeaponsText("InterspaceSword");

对于Armors键里有多层，则可以这样写：

ArmorsText("InterspaceArmors.Head");

Language.GetTextValue("Mods.InterspaceMod.Armors.InterspaceArmors.Head");