

# WordPress开发手册

jabber



# 目 录

简介

主题开发

WordPress许可证

什么是主题

开发环境

主题开发示例

主题基础

模板文件

主样式表(style.css)

文章类型

规划主题文件

模板层级

模板标签

循环

主题函数

连接主题文件和目录

使用CSS和JavaScript

条件标签

类别，标签和自定义分类

模板文件

内容模板文件

页面模板文件

附件模板文件

自定义内容类型

部分和其他模板文件

评论模板

分类模板

404页面

主题功能

核心支持的功能

管理菜单

自定义Headers

自定义Logo

文章格式

置顶文章

Sidebars

Widgets

导航菜单

分页

媒体

Audio

Images

Galleries

Video

精选图片和缩略图

国际化

本地化

辅助功能

主题选项 – 自定义API

定制对象

改进用户体验的工具

定制JavaScript API

JavaScript / Underscore.js渲染的自定义控件

高级用法

主题安全

数据消毒/逃避

数据验证

使用随机数

常见漏洞

高级主题

子主题

UI最佳实践

JavaScript最佳做法

主题单元测试

验证你的主题

Plugin API Hooks

发布你的主题

所需的主题文件

测试

主题评论指南

写文档

提交你的主题到WordPress.org

参考文献

模板标签列表

条件标签列表

编码标准

HTML编码标准

CSS编码标准

JavaScript编码标准

PHP编码标准

插件开发

## 插件开发简介

### 什么是插件

## 插件基础

### 头部要求

### 包括软件许可证

### 启用 / 停用 Hooks

### 卸载方法

### 最佳做法

## 插件安全

### 检查用户功能

### 数据验证

### 保护输入

### 保护输出

### 随机数

## Hooks

### Actions

### Filters

### 自定义Hooks

### 高级主题

## 管理菜单

### 顶级菜单

### 子菜单

## 短代码

### 基本短码

### 封闭短码

### 带参数的短代码

### TinyMCE增强型短码

## 设置

### 设置API

### 使用设置API

### 选项API

### 自定义设置页面

## 元数据

### 管理帖子元数据

### 自定义元数据

### 渲染元数据

## 自定义文章类型

### 注册自定义文章类型

### 使用自定义文章类型

## 分类

### 使用自定义分类

### 在WP 4.2+中使用“split术语”

## 用户

[创建和管理用户](#)

[使用用户元数据](#)

[角色和功能](#)

## HTTP API

### JavaScript

[jQuery](#)

[Ajax](#)

[服务器端PHP和入队](#)

[Heartbeat API](#)

[概要](#)

### 计划任务

[了解WP-Cron计划](#)

[安排WP-Cron 事件](#)

[将WP-Cron挂接到系统任务计划程序中](#)

[WP-Cron简单测试](#)

### 国际化

[本地化](#)

[如何国际化您的插件](#)

[国际化安全](#)

### WordPress.org

[详细插件指南](#)

[规划您的插件](#)

[如何使用Subversion](#)

[插件开发者常见问题](#)

### 开发工具

[Debug Bar 和附加组件](#)

[辅助插件](#)

## REST API手册

### 资源

[文章](#)

[文章修订](#)

[文章类型](#)

[文章状态](#)

[类别](#)

[标签](#)

[页面](#)

[评论](#)

[分类](#)

[媒体](#)

[用户](#)

[设置](#)

## 使用REST API

[全局参数](#)

[分页](#)

[链接和嵌入](#)

[发现](#)

[认证](#)

[经常问的问题](#)

[骨干JavaScript客户端](#)

[客户端库](#)

## 扩展REST API

[添加自定义端点](#)

[自定义内容类型](#)

[修改回应](#)

[模式](#)

[词汇表](#)

[路由和端点](#)

[控制器类](#)

# 简介

---

## 前言

很久以前一直想做WordPress主题的开发，但理由太多没有坚持下去。最近整理了自己手上的工作，决定还是做WordPress的开发。

## 介绍

WordPress是使用PHP语言开发的开源博客系统，用户可以在支持PHP和MySQL数据库的服务器上架设属于自己的网站。随着WordPress使用者的增多，WordPress相关的产品也越来越丰富。

本书主要以WordPress V4.7.2版本为基础，来介绍如何做WordPress二次开发。

在WordPress的后台可在线安装自己喜欢的主题和需要的插件。WordPress主题和插件非常多，我们可以利用已有的插件和主题快速搭建博客网站、企业网站、社区、论坛、在线销售网站等。已有的主题或插件无法满足自己需求的情况下，我们可以自己开发插件和主题。

# 主题开发

---

欢迎来到WordPress主题开发者手册，您的资源，用于学习关于令人兴奋的WordPress主题世界。

主题开发者手册是所有WordPress主题的存储库。无论您是WordPress的新主题，还是您是经验丰富的主题开发人员，您都可以在这里找到许多与主题相关的问题的答案。

- 如果您是开发WordPress主题的新手，请从第1部分开始，您可以在其中找到[主题](#)，了解[WordPress的许可证](#)，[设置开发环境](#)以及构建您的第一个主题。
- 完成介绍后，主题[基础部分](#)将向您介绍WordPress主题的构建块。
- 主题[功能部分](#)将显示您可以在主题中使用的所有不同类型的功能。
- 如果您希望提供灵活性并保护用户，请转到“[定制程序](#)和[安全性](#)”部分
- 如果您必须掌握主题的基础知识，请查看[高级主题](#)，了解高级主题，最佳UI实践，主题测试等。
- 一旦你的主题为世界准备好了，最后一节将介绍如何[发布你的主题](#)，教你一些主题分发的最佳做法，并准备好为WordPress.org主题目录。



# WordPress许可证

## WordPress许可和GPL

要为公众开发WordPress主题（免费或付费），您需要熟悉WordPress使用的GNU通用公共许可证（GPL）。

### GPL开源协议

WordPress社区内部开放和分享的精神蓬勃发展，因为基本原则构成了许可证的核心。考虑GPL的一种方式 是软件的“权利法案”。GPL确定了以下四种自由：

- 为任何目的自由运行程序。
- 自由研究程序如何工作和改变它，所以它执行计算，如你所愿。
- 重新分发副本的自由，所以你可以帮助你的邻居。
- 自由分发您的修改版本的副本，使社区有机会受益于您的更改。

### 什么是“自由”软件

自由软件中的“自由”是指自由而不是价格。自由软件基金会喜欢说“免费在言语中，而不是像啤酒一样”。免费软件是用户可以根据自己的想法使用的软件。它不需要免费，尽管在WordPress.org主题目录中托管的那些。免费软件可以带有价格标签。换句话说，您可以创建一个GPL主题，并以\$ 50的价格销售，而且仍然是免费软件。为什么？因为用户可以自由地运行，修改和分发该软件或该软件的任何修改。

### 保持所有人免费

GPL的自由不仅适用于原始的软件；从GPL许可软件衍生的作品也必须采用相同的许可证，无限制或附加条款。在这个意义上，GPL通过确保从自由软件获得的任何东西在事实之后不能被“锁定”，从而最终保护自由。它必须永远保持未来的实验和探索。

### 我需要根据GPL授权我的主题吗？

如果您没有计划分发您的主题，那么您不需要为您的工作采用GPL许可。GPL仅适用于分布式软件。如果您没有分发软件 - 例如，仅由您自己或本地机器使用的主题 - 您不需要采用GPL。

如果您希望将创作提交到WordPress.org上的免费主题库，则必须符合100%GPL标准，包括CSS和图像文件。由于GPL中阐述的自由是WordPress的核心，我们鼓励开发人员使用100%GPL兼容许可证分发主题。

**注意：**自由是开发WordPress主题的重要组成部分。如果您计划分发您的主题，在GPL下完全授权它是一个好主意，所以其他人可以享受与创建相同的自由。

### 进一步了解

加深对WordPress和GPL的理解：

- [WordPress.org](#)：主题也是GPL
- [Q&A](#)：WordPress的GPL
- [四大自由](#)
- [GNU通用公共许可证，版本2](#)
- [GNU通用公共许可证，版本3](#)

# 什么是主题

## 什么是主题

WordPress主题会改变您的网站的设计，通常包括其布局。更改您的主题会改变您的网站在前端的外观，即访问者浏览您的网站时看到的内容。WordPress.org主题目录中有数以千计的免费WordPress主题，尽管许多WordPress网站都使用自定义主题。

### 主题可以做什么？

主题采用WordPress存储的内容和数据，并将其显示在浏览器中。当您创建WordPress主题时，您可以决定该内容的外观和显示方式。建立主题时，可以使用许多选项。例如：

- 您的主题可以使用不同的布局，例如静态或响应式，使用一列或两列。
- 您的主题可以显示任何想要显示的内容。
- 您的主题可以指定哪些设备或操作使您的内容可见。
- 您的主题可以使用CSS自定义其排版和设计元素。
- 其他设计元素，如图像和视频可以包含在您的主题的任何地方。

WordPress主题是非常强大的。但是，与每个网页设计项目一样，主题不仅仅是颜色和布局。良好的主题除了美丽以外，还可以改善您网站内容的参与度。

### WordPress主题

在最基本的层面上，WordPress主题是不同文件的集合，可共同创建您所看到的内容，以及您的网站的行为。

### 所需文件

WordPress主题中必需要以下两个文件：

- index.php – 主模板文件
- style.css – 主样式文件

虽然不需要，您可能会在主题文件夹中看到其他文件，其中包括：

- PHP文件 – 包括模板文件
- 本地化文件
- CSS文件
- 图片资源
- 脚本文件(JavaScript)
- 文本文件 – 通常是许可证信息，`readme.txt` 指令和 `changelog` 文件

## 主题和插件有什么区别？

在主题和插件中找到的功能之间发现交叉是很常见的。但是，最佳做法是：

- 主题控制内容的呈现; 而插件用于控制您的WordPress网站的行为和功能。
- 您创建的任何主题都不应该添加关键功能。这样做意味着当用户更改主题时，他们将无法访问该功能。例如，假设您使用投资组合功能构建主题。使用您的功能构建投资组合的用户将在更改主题时失去它。
- 通过将关键功能移植到插件中，您可以使您的网站的设计更改，同时功能保持不变。

注意：请记住，有些用户经常转换主题。最佳做法是确保您的网站所需的任何功能，即使设计更改，也是单独的插件。

## WordPress.org上的主题

WordPress主题下载WordPress主题最安全的地方之一是WordPress.org主题目录。所有主题都经过严格审查，必须符合严格的主题审查指南，以确保质量和安全。

## 入门

现在你知道什么是主题是时候开始了。如果还没有这样做，你应该设置你的本地开发环境。然后，您可以查看一些WordPress主题的示例，或者如果您不能再等待开始，请潜入构建您的第一个主题。

# 开发环境

## 为什么要建立开发环境？

在开发主题时，最好在与最终托管WordPress安装的生产服务器相同的环境中执行此操作。您的开发环境可以是本地的或远程的。配置本地环境来处理WordPress主题有以下几个原因：

- 您可以在本地构建您的主题而不依靠远程服务器。这可以加速您的开发过程，并允许您立即在浏览器中查看更改。
- 您不需要Internet连接来构建您的主题。
- 您可以从各种角度测试您的主题。这很重要，特别是如果您打算将主题发布给更多的受众，并希望确保最大的兼容性。

## WordPress本地开发环境

要开发WordPress主题，您需要设置适合于WordPress的开发环境。要开始，您将需要一个本地服务器堆栈和一个文本编辑器。有很多选择，包括：

### 本地服务器堆栈

- 本地服务器堆栈（如LAMP（Linux Apache MySQL PHP）或WAMP（Windows Apache MySQL PHP）是服务器（与您的Web服务器上运行的服务器非常相似），您将在本地机器上进行配置。您可以安装包含所有这些的预捆绑程序，如MAMP（适用于Mac）或XAMPP（Mac或Windows），以快速设置您的环境。

### 虚拟环境

- 使用Vagrant和VirtualBox创建的虚拟化功能允许您创建易于重现的开发环境。[Varyant Vagrant Vagrant（VVV）](#)是流行的Vagrant选项，创建了WordPress开发环境。

### 文本编辑器

- 除了本地服务器环境之外，还需要一个文本编辑器来编写代码。您选择的文本编辑器是个人的，但请记住，一个好的文本编辑器可以加快您的开发过程。您的文本编辑器可以从编写代码到完全集成的开发环境（IDE）的基本工具中进行调试和测试。值得进行研究，有些甚至包括对WordPress开发的支持。热门的选择是Atom，Sublime Text和PhpStorm。

您可以在页面底部找到有关设置开发环境的教程列表。

## 支持较旧版本的WordPress

WordPress主题的标准做法是至少支持两个版本，以确保最小的向后兼容性。例如，如果当前版本的WordPress为4.6，那么您还应该确保您的主题在4.5和4.4版本中运行良好。

您可以参考[WordPress路线图](#)页面访问旧版本的WordPress。然后，您可以下载并安装较旧的WordPress版本，创建多个开发站点，每个版本运行不同的WordPress版本进行测试。

## WP\_DEBUG

配置调试是WordPress主题开发的重要组成部分。WordPress提供了许多常量来支持您的调试工作。这些包括：

### WP\_DEBUG

- `WP_DEBUG` PHP常量用于在您的WordPress安装上触发内置的“调试”模式。这允许您查看主题中的错误。启用它：
  1. 打开您的WordPress安装的wp-config.php文件
  2. 修改:

```
define( 'WP_DEBUG', false );
```

为

```
define( 'WP_DEBUG', true );
```

注意：通常在wp-config.php文件中设置为“false”，默认情况下，即将发布的WP\_DEBUG版本的WordPress-alpha和beta版本的开发副本已设置为“true”。

## WP\_DEBUG\_DISPLAY 和 WP\_DEBUG\_LOG

`WP_DEBUG_LOG` 和 `WP_DEBUG_DISPLAY` 是扩展 `WP_DEBUG` 的其他PHP常量。

`WP_DEBUG_LOG` 与 `WP_DEBUG` 结合使用，将所有错误消息记录到WordPress /wp-content/目录中的debug.log中。要启用此功能，您的 `wp-config.php` 文件中将 `WP_DEBUG_LOG` 设置为 `true`。

```
define( 'WP_DEBUG_LOG', true );
```

`WP_DEBUG_DISPLAY` 用于控制调试消息是否显示在主题页面的HTML中。要在屏幕上显示错误消息，请在 `wp-config.php` 文件中将此设置配置为 `true`。

```
define( 'WP_DEBUG_DISPLAY', true );
```

启用 `WP_DEBUG` 和 `WP_DEBUG_DISPLAY` 后，错误消息将显示在您网站页面的顶部。

注意：错误将显示在您的站点的前端和管理区域。这些调试工具用于本地测试和后台安装，而不是在发布站点。

## 其他WordPress开发工具

除了 `WP_DEBUG`，以下插件和单元测试数据集是开发工具集的重要组成部分，可帮助您开发更好的WordPress主题。

### 测试数据

#### WordPress.org 主题单元测试数据

WordPress.org主题单元测试数据是一个包含虚拟测试数据的XML文件，您可以上传以测试主题如何与不同类型和内容布局进行匹配。

#### WordPress.com 主题单元测试数据

WordPress.com主题单元测试数据是虚拟测试数据，您可以上传到WordPress安装来测试您的主题，包括特定于WordPress.com的功能。

### 插件

#### Debug Bar (WordPress插件)

- Debug Bar为您的WordPress管理员添加一个管理栏，提供调试中心位置。

#### Query Monitor (WordPress插件)

- Query Monitor 允许调试数据库查询，API请求和AJAX调用用于生成主题页和主题功能。

#### Log Deprecated Notices (WordPress插件)

- 日志已弃用通知记录不正确的功能使用和WordPress主题中使用已弃用的文件和功能。

#### Monster Widgets (WordPress插件)

- Monster Widget将核心WordPress小部件整合到一个小部件中，允许您在主题中测试小部件的样式和功能。

#### Developer (WordPress插件)

- 开发人员通过轻松安装工具和插件来帮助您排除故障并确保代码质量，帮助您优化开发环境。

#### Theme-Check (WordPress插件)

- Theme-Check测试您的主题，以符合最新的WordPress标准和做法。

## WordPress主题评论指南

---

除了上述开发工具之外，最好在[WordPress.org主题评审小组](#)的主题提交指南和[WordPress编码标准](#)会议指导上保持最新。这些指南是质量主题开发的“黄金标准”，即使您不打算在WordPress.org上发布主题，也是有用的。

## 其他资源

---

- 使用MAMP本地开发WordPress (Mac, MAMP)
- 如何为Windows设置WordPress开发环境 (Windows, XAMPP)
- WordPress主题评论VVV：测试主题的快速流动设置 (Cross-platform, Vagrant)
- 设置开发环境 ([WordPress.com](#) VIP)
- [wptest.io](#) – 从WordPress的主题单元测试派生的一系列WordPress测试数据



# 主题开发示例

---

了解主题编码标准的最佳方法之一就是找出其他已经用这些标准编写的主题的例子。

## 默认的“Twenty”主题

---

自版本3.0（并在其发布年份之后命名）中的每个版本的WordPress中打包，默认主题是研究如何构建主题的一些最佳方法。这是因为它们被广泛用于设计，并且完全遵守WordPress编码标准。您可以下载并学习他们的主题文件，并将其作为示例参考，同时学习如何开发自己的主题：

- Twenty Seventeen
- Twenty Fifteen
- Twenty Fourteen
- Twenty Thirteen
- Twenty Twelve
- Twenty Eleven
- Twenty Ten

## \_s主题

---

与默认的“Twenties”主题不同，\_s主题针对的是开发人员而不是最终用户。它旨在成为一个起始主题，您可以将其作为加快开发速度的基础。它有一些功能：

- 良好评论的HTML5模板，包括错误模板。
- 在inc/custom-header.php中的示例自定义标头实现。
- 使用inc/template-tags中的自定义模板标签来保持模板的组织并防止代码重复。
- 在js/keyboard-image-navigation.js中找到了许多改进键盘导航的脚本，以及js/navigation.js中的小屏幕导航。
- CSS布局/布局中的五个样本以及启动CSS，用于构建您的设计。
- GPL许可代码。

上面的功能使Underscores成为想要创建自己的主题的开发人员的主题。即使您删除了附加组件，剩下的基础仍然是编写标准的主题的一个很好的例子，这些主题是根据标准和最佳实践开发的。

源码：[https://github.com/Automattic/\\_s](https://github.com/Automattic/_s)

## 其他主题

---

此外，主题目录中发布的所有主题都将在发布之前进行审查。查看目录中的主题是更好地了解主题开发如何工作的好方法，并且是为您自己的主题获得灵感的好方法。

# 主题基础

---

在第1章中，您将看到一个主题的概述。 你也学到了如何开始开发一个主题。

在本章中，您将开始学习如何正确构建主题。 主题及其运动部分的解剖将被分解和解释。 您将首先通过查看主题文件和帖子类型来了解主题的构建块。 然后，您将学习如何将您的文件保存在主题中。

您还将看到The Loop，它负责将内容从WordPress数据库中拉出。

最后，您将通过使用主题功能（包括CSS和JavaScript），利用条件标签仅显示您需要的内容，以及使用默认分类法并创建自己的内容，了解更多关于向主题添加功能的功能。

# 模板文件

---

WordPress主题中使用模板文件，但首先让我们了解术语。

## 模板术语

---

使用WordPress主题时，术语“模板”以不同的方式使用：

- 主题中存在模板文件，并表达您的网站的显示方式。
- 页面模板是仅适用于页面以更改其外观的页面模板。 页面模板可以应用于单个页面，页面部分或一类页面。
- 模板标签是内置的WordPress功能，您可以在模板文件中使用来检索和显示数据（如the\_title()和the\_content()）。
- 模板层次结构是WordPress用于根据所请求的内容决定要使用哪个主题模板文件的逻辑。

## 模板文件

---

WordPress主题由模板文件组成。这些是包含HTML，模板标签和PHP代码混合的PHP文件。

当您构建主题时，您将使用模板文件来影响网站不同部分的布局和设计。例如，您可以使用header.php模板来创建一个页头，或者使用comments.php模板来包含评论。

当有人访问您网站上的页面时，WordPress会根据请求加载模板。由模板文件显示的内容类型由与模板文件相关联的帖子类型确定。模板层次结构描述了WordPress将根据请求的类型以及主题中是否存在模板来加载哪个模板文件。服务器然后解析模板中的PHP，并将HTML返回给访问者。

最关键的模板文件是index.php，如果在模板层次结构中找不到更具体的模板，那么它就是全部的模板。虽然主题只需要一个index.php模板，通常主题包括许多模板来显示不同的内容类型和上下文。

## 模板部分

---

模板部分是作为另一个模板的一部分包括的一个模板，例如站点头。 模板部分可以嵌入多个模板，简化主题创建。 常见的模板部分包括：

- header.php 用于生成站点的头文件
- footer.php 用于生成页脚
- sidebar.php 用于生成侧边栏
- searchform.php 用于生成搜索表单

虽然上述模板文件是WordPress中的特殊情况，仅适用于页面的一部分，您可以创建任意数量的模板部分，并将它们包含在其他模板文件中。

## 常用的WordPress模板文件

---

以下是WordPress识别的一些基本主题模板和文件的列表。

- `index.php` 主模板文件。所有主题都是必需的。
- `style.css` 主要样式表。它在所有主题中都是必需的，并且包含主题的信息标题。
- `rtl.css` 如果网站语言的文本方向是从右到左，则自动包含从右到左的样式表。
- `comments.php` 评论模板。
- `front-page.php` 首页模板始终用作站点首页（如果存在），无论管理员>设置>阅读上的设置如何。
- `home.php` 默认情况下，主页模板是首页模板。如果您没有将WordPress设置为使用静态首页，则此模板用于显示最新的帖子。
- `header.php` 标题模板文件通常包含您的站点的文档类型，元信息，样式表和脚本的链接以及其他数据。
- `singular.php` 单独的模板用于没有找到`single.php`的帖子，或者当没有找到`page.php`的页面时。如果没有找到`singular.php`，则使用`index.php`。
- `single.php` 当访问者请求单个帖子时，使用单个帖子模板。
- `single-{post-type}.php` 访问者从自定义帖子类型请求单个帖子时使用的单个帖子模板。例如，`single-book.php`将用于从定制的帖子类型命名的书中显示单个帖子。如果不存在自定义帖子类型的特定查询模板，则使用`index.php`。
- `archive-{post-type}.php` 当访问者请求自定义帖子类型归档时，将使用归档文件类型模板。例如，`archive-books.php`将用于显示自定义帖子类型命名书籍的帖子存档。如果`archive-{post-type}.php`不存在，则使用`archive.php`模板文件。
- `page.php` 当访问者请求单独的页面（内置模板）时，将使用页面模板。
- `page-{slug}.php` 访问者请求特定页面时使用页面插件模板，例如使用“about” slug（`page-about.php`）的页面插件模板。
- `category.php` 当访问者按类别请求帖子时，将使用类别模板。
- `tag.php` 当访问者通过标签请求帖子时，使用标记模板。
- `taxonomy.php` 当访问者在自定义分类法中请求术语时，将使用分类术语模板。
- `author.php` 访问者加载作者页面时，将使用作者页面模板。
- `date.php` 日期/时间模板在通过日期或时间请求帖子时使用。例如，使用这些子生成的页面：
  - <http://example.com/blog/2014/>
  - <http://example.com/blog/2014/05/>
  - <http://example.com/blog/2014/05/26/>
- `archive.php` 当访问者按类别，作者或日期请求帖子时，使用归档模板。注意：如果存在类似于`category.php`，`author.php`和`date.php`的更多特定模板，则此模板将被覆盖。
- `search.php` 搜索结果模板用于显示访问者的搜索结果。
- `attachment.php` 当查看单个附件（如图像，pdf或其他媒体文件）时，将使用附件模板。

- `image.php` 图像附件模板是attachment.php的更具体的版本，在查看单个图像附件时使用。如果不存在，WordPress将使用attachment.php。
- `404.php` 当WordPress找不到与访问者请求相匹配的帖子，页面或其他内容时，将使用404模板。

## 使用模板文件

---

在WordPress模板中，您可以使用模板标签动态显示信息，包括其他模板文件，或以其他方式自定义您的网站。

例如，在index.php中，您可以在最终生成的页面中包含其他文件：

- 要包括标题，请使用 `get_header()`
- 要包含边栏，请使用 `get_sidebar()`
- 要包括页脚，请使用 `get_footer()`
- 要包含搜索表单，请使用 `get_search_form()`
- 要包括自定义主题文件，请使用 `get_template_part()`

以下是WordPress模板标记的示例，以将特定的模板包含在您的页面中：

```
<?php get_sidebar(); ?>
<?php get_template_part( 'featured-content' ); ?>
<?php get_footer(); ?>
```

# 主样式表(style.css)

style.css是每个WordPress主题所需的样式表（CSS）文件。它控制网页的演示（视觉设计和布局）。

## 位置

为了使WordPress将主题模板文件的集合识别为有效的主题，style.css文件需要位于主题的根本目录中，而不是子目录。

有关如何将style.css文件包含在主题中的更详细说明，请参阅“[启动脚本和样式](#)”的“样式表”部分。

## 基本结构

WordPress使用style.css的标题注释部分在“外观（主题）”仪表板面板中显示有关主题的信息。

## 示例

这是style.css的头部分的一个例子。

```
/*
Theme Name: Twenty Seventeen

Theme URI: https://wordpress.org/themes/twentyseventeen/

Author: the WordPress team

Author URI: https://wordpress.org/

Description: Twenty Seventeen brings your site to life with immersive featured
images and subtle animations. With a focus on business sites, it features multi
ple sections on the front page as well as widgets, navigation and social menus,
a logo, and more. Personalize its asymmetrical grid with a custom color scheme
and showcase your multimedia content with post formats. Our default theme for
2017 works great in many languages, for any abilities, and on any device.

Version: 1.0

License: GNU General Public License v2 or later

License URI: http://www.gnu.org/licenses/gpl-2.0.html

Text Domain: twentyseventeen

Tags: one-column, two-columns, right-sidebar, flexible-header, accessibility-re
ady, custom-colors, custom-header, custom-menu, custom-logo, editor-style, feat
ured-images, footer-widgets, post-formats, rtl-language-support, sticky-post, t
heme-options, threaded-comments, translation-ready
```

```
This theme, like WordPress, is licensed under the GPL.  
Use it to make something cool, have fun, and share what you've learned with others.  
*/
```

使用(\*)表示的项目是WordPress主题库中的主题所必需的。

**注意：**WordPress主题存储库使用此文件中“版本”之后的数字来确定主题是否具有可用的新版本。

- Theme Name (\*): 主题名称。
- Theme URI: 公共网页的URL，用户可以在其中找到有关该主题的更多信息。
- Author (\*): 开发主题的个人或组织的名称。建议使用主题作者的wordpress.org用户名。
- Author URI: 创作个人或组织的网址。
- Description (\*): 简短描述的主题。
- Version (\*): 该版本以X.X或X.X.X格式编写。
- License (\*): 主题的协议。
- License URI (\*): 主题许可证的URL。
- Text Domain (\*): 用于文本域的字符串用于翻译。
- Tags: 允许用户使用标签过滤器查找主题的单词或短语。标签的完整列表在“[主题评论手册](#)”中。

在所需的标题部分之后，style.css可以包含常规CSS文件中的任何内容。

## style.css中的子主题

如果您的主题是“子主题”，则在style.css标题中需要“模板”行。

```
/*  
Theme Name: My Child Theme  
Template: Twenty Seventeen  
*/
```

有关创建子主题的更多信息，请访问“[子主题](#)”页面。

# 文章类型

---

WordPress中有许多不同类型的内容。 这些内容类型通常被描述为Post Types，这可能有点混乱，因为它引用了WordPress中的所有不同类型的内容。 例如，一个帖子是一个特定的帖子类型，页面也是这样。

在内部，所有的Post类型都存储在wp\_posts数据库表中的相同位置 - 但是由名为post\_type的数据库列区分。

除了默认的Post Types之外，您还可以创建自定义帖子类型。

模板文件页面简要说明了不同的模板文件显示不同的Post Types。 由于模板文件的全部目的是以某种方式显示内容，所以邮政类别的目的是分类您正在处理的内容类型。 一般来说，某些帖子类型与某些模板文件相关。

## 默认Post Types

---

有五种默认的Post Types可供用户使用或WordPress安装内部使用：

- 内容 (Post Type: 'post' )
- 页面 (Post Type: 'page' )
- 附件 (Post Type: 'attachment' )
- 修订版 (Post Type: 'revision' )
- 导航菜单 (Post Type: 'nav\_menu\_item' )

上述的内容类型可以通过插件或主题进行修改和删除，但不建议您删除广泛分布的主题或插件的内置功能。

您将作为主题开发人员交互的最常见的帖子类型是帖子，页面，附件和自定义帖子类型。 修改和导航菜单帖子类型不在本手册的范围之内。 但是，请注意，您将与导航菜单进行交互并构建功能，这将在本手册后面详细介绍。

## 内容

---

内容用于博客。 他们是：

- 按照时间顺序顺序显示，最新的帖子
- 有日期和时间戳
- 可能具有应用类别和标签的默认分类
- 用于创建Feed

显示Post，Post Types的模板文件是：

- single.php和single-post.php
- category.php 及其所有迭代
- tag.php 及其所有的迭代
- taxonomy.php 及其所有迭代
- archive.php 及其所有迭代



- author.php 及其所有迭代
- date.php 及其所有迭代
- search.php
- home.php
- index.php

另外，主题开发人员如果愿意，可以在front-page.php中显示Post post类型。

阅读更多关于[内容模板文件](#)。

## 页面

---

页面是一个静态的帖子类型，不在正常的博客Feed中。 他们的特点是：

- 非时间依赖，没有时间戳
- 没有使用类别和/或标签分类法进行组织
- 可以使用页面模板
- 可以以层次结构组织 - 即页面可以是其他页面的父母/子项

显示页面类型的模板文件是：

- page.php及其所有的迭代
- \$custom.php及其所有迭代
- front-page.php
- search.php
- index.php

阅读更多关于[页面模板文件](#)。

## 附件

---

附件通常用于在内容中显示图像或媒体，也可用于链接到相关文件。 他们的特点是：

- 包含关于通过媒体上传系统上传的文件的信息（例如名称或描述）
- 对于图像，这包括存储在wp\_postmeta表中的元数据信息（包括大小，缩略图，位置等）

显示“附件”类型的模板文件包括：

- MIME\_type.php
- attachment.php
- single-attachment.php
- single.php

- [index.php](#)

阅读更多关于[附件模板文件](#)。

## 自定义内容类型

使用自定义帖子类型，您可以创建自己的帖子类型。不建议您将此功能放在主题中。这种类型的功能应该在插件中放置/创建。这确保了用户内容的可移植性，如果主题已更改，存储在自定义帖子类型中的内容将不会消失。

您可以在WordPress插件开发者手册中了解更多有关创建自定义帖子类型的信息。

虽然您通常不会在主题中开发自定义帖子类型，但您可能需要编写显示由插件创建的自定义帖子类型的方法。以下模板可以显示自定义帖子类型：

- `single- {post-type} .php`
- `archive- {post-type} .php`
- `search.php`
- `index.php`

此外，主题开发人员可以在任何模板文件中显示自定义帖子类型，通常通过使用多个循环。

详细了解[自定义内容类型模板](#)。

# 规划主题文件

虽然WordPress主题技术上只需要两个文件（`index.php`和`style.css`），它们通常由许多文件组成。这意味着他们可以很快变得混乱！本节将向您展示如何保持文件的整理。

**注意：**自从WordPress 3.0以来，没有`header.php`和`footer.php`的主题没有可用的，已经被弃用了。您的主题可能还需要包含这些文件。

## 主题文件夹和文件结构

如前所述，默认的二十个主题是良好的主题开发的一些最好的例子。例如，二十七主题如何组织其文件结构：

```
assets (dir)
  - css (dir)
  - images (dir)
  - js (dir)
inc (dir)
template-parts (dir)
  - footer (dir)
  - header (dir)
  - navigation (dir)
  - page (dir)
  - post (dir)
404.php
archive.php
comments.php
footer.php
front-page.php
functions.php
header.php
index.php
page.php
README.txt
rtl.css
screenshot.png
search.php
searchform.php
sidebar.php
single.php
style.css
```

您可以看到主要的主题模板文件位于根目录下，而JavaScript，CSS，图像则放置在素材目录中，模板部分放置在相应子模板部分的子目录下，与核心功能相关的功能的集合被放置 in `inc`目录。

此时WordPress主题中没有必需的文件夹。但是，WordPress默认情况下会识别以下文件夹。

注意：style.css应该位于您的主题的根目录中，不在CSS目录中。

## 语言文件夹

---

最好的做法是将您的主题国际化，以便将其翻译成其他语言。默认主题包括languages文件夹，其中包含用于翻译的.pot文件和任何已翻译的.mo文件。虽然语言是此文件夹的默认名称，您可以更改名称。如果这样做，您必须更新load\_theme\_textdomain()。

# 模板层级

如所讨论的，模板文件是模块化的，可重复使用的文件，用于在WordPress站点上生成网页。一些模板文件（如页眉和页脚模板）用于所有网站的页面，而其他模板文件仅在特定条件下使用。

本文解释WordPress如何确定在单个页面上使用哪个模板文件。如果要自定义现有的WordPress主题，它将帮助您确定需要编辑的模板文件。

**提示：**您还可以使用条件标签来控制在特定页面上加载哪些模板。

## 模板文件层次结构

### 概述

WordPress使用查询字符串来决定哪个模板或一组模板应用于显示页面。查询字符串是指向您网站每个部分的链接中的信息。它来自初始问号，可能包含多个由&符号分隔的参数。

简单地说，WordPress通过模板层次结构搜索，直到找到匹配的模板文件。要确定使用哪个模板文件，WordPress：

- 将每个查询字符串匹配到查询类型以确定正在请求哪个页面（例如，搜索页面，类别页面等）；
- 按照模板层次结构确定的顺序选择模板；
- 在当前主题的目录中查找具有特定名称的模板文件，并使用层次结构指定的第一个匹配的模板文件。
- 除了基本的index.php模板文件外，您可以选择是否要实现特定的模板文件。

如果WordPress找不到具有匹配名称的模板文件，它将跳到层次结构中的下一个文件。如果WordPress找不到任何匹配的模板文件，将使用主题的index.php文件。

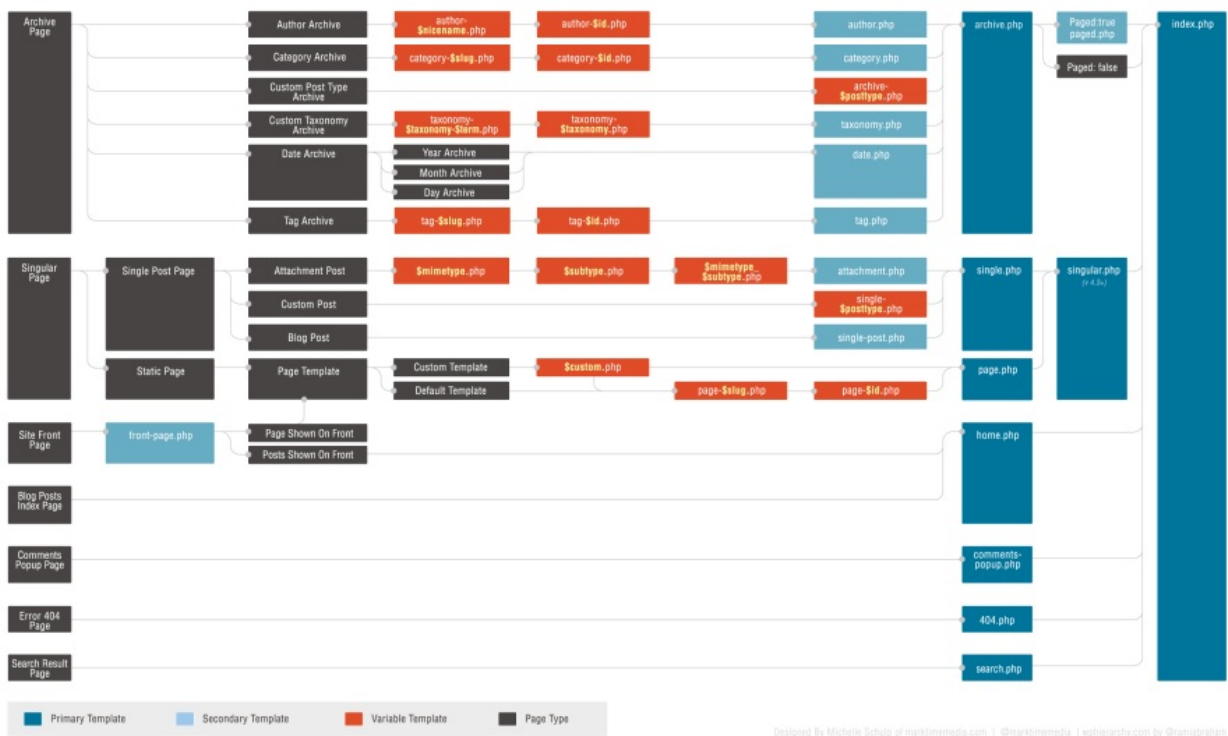
### 示例

如果您的博客位于http://example.com/blog/，访问者点击链接到类别页面（例如http://example.com/blog/category/your-cat/），WordPress会查找模板 在当前主题的目录中匹配类别的ID以生成正确的页面。更具体地说，WordPress遵循以下过程：

- 在当前主题的目录中查找与该类别的插件匹配的模板文件。如果类别slug是“unicorns”，则WordPress会查找名为category-unicorns.php的模板文件。
- 如果category-unicorns.php缺少且类别的ID为4，则WordPress将查找名为category-4.php的模板文件。
- 如果缺少类别4.php，WordPress将寻找一个通用类别模板文件category.php。
- 如果category.php不存在，WordPress将寻找一个通用归档模板archive.php。
- 如果archive.php也丢失，WordPress将回到主题模板文件index.php。

## 视觉概述

下图显示了哪些模板文件被调用以基于WordPress模板层次结构生成WordPress页面。



您也可以与[此图交互](#)。

## 模板层次详细

虽然模板层次结构更容易被理解为图表，但以下部分描述了WordPress为多种查询类型调用模板文件的顺序。

### 主页显示

默认情况下，WordPress设置您的网站的主页以显示最新的博文。这个页面被称为博客帖子索引。您也可以将您的博客帖子设置为在单独的静态页面上显示。模板文件home.php用于呈现博客帖子索引，无论是用作首页还是单独的静态页面。如果home.php不存在，WordPress将使用index.php。

- home.php
- index.php

**注意：**如果front-page.php存在，它将覆盖home.php模板。

## 首页

front-page.php模板文件用于呈现您网站的首页，首页是否显示博客帖子索引（如上所述）或静态页面。首页模

板优先于博客帖子索引 ( home.php ) 模板。 如果front-page.php文件不存在，WordPress将根据设置→阅读中的设置使用home.php或page.php文件。 如果这两个文件都不存在，它将使用index.php文件。

- front-page.php - 用于首页显示设置→阅读部分中的“您的最新帖子”或“静态页面”。
- home.php - 如果WordPress找不到front-page.php和“你的最新帖子”设置在首页显示部分，它将寻找home.php。此外，当在首页显示部分中设置帖子页面时，WordPress将查找此文件。
- page.php - 在首页显示部分设置“首页”时。
- index.php - 在首页显示部分设置“您的最新帖子”，但home.php不存在或者当首页设置但page.php不存在时。

正如你所看到的，WordPress所采用的路径有很多规则。使用上面的图表是确定WordPress将显示的最佳方式。

## 单页内容

---

单页内容模板文件用于呈现单页内容。WordPress使用以下路径：

- single-{post-type} - {slug} .php - （从4.4开始）首先，WordPress寻找特定内容的模板。例如，如果内容类型是产品，并且post slug是dmc-12，则WordPress会查找单个产品dmc-12.php。
- single- {post-type} .php - 如果帖子类型是产品，WordPress将寻找单一product.php。
- single.php - WordPress然后回到single.php。
- singular.php - 然后它回到singular.php。
- index.php - 最后，如上所述，WordPress最终会回到index.php。

## 单页

---

用于呈现静态页面的模板文件（页面后置类型）。请注意，与其他后期类型不同，页面特别适用于WordPress，并使用以下修补程序：

自定义模板文件 - 分配给页面的页面模板。请参阅get\_page\_templates()。

- page- {slug} .php - 如果该页面是最新消息，WordPress将会使用page-recent-news.php。
- page- {id} .php - 如果页面ID为6，WordPress将使用page-6.php。
- page.php
- singular.php
- index.php

## 类别

---

渲染类别归档索引页在WordPress中使用以下路径：

- category- {slug} .php - 如果类别的lug lug是新闻，WordPress将会查找category-news.php。
- category- {id} .php - 如果类别的ID为6，WordPress将寻找类别为6.php。

- category.php
- archive.php
- index.php

## 标签

---

要显示标签归档索引页面，WordPress使用以下路径：

- tag- {slug} .php - 如果标签的插件是某个标签，WordPress将会查找tag-sometag.php。
- tag- {id} .php - 如果标签的ID为6，WordPress将寻找标签6.php。
- tag.php
- archive.php
- index.php

## 自定义分类

---

自定义分类使用稍微不同的模板文件路径：

- taxonomy-{taxonomy}-{term}.php – 如果分类是某种类型，而分类学术语是某种语言，则WordPress将寻找分类法。在发布格式的情况下，分类是“post\_format”，术语是“post-format- {format}”。即链接后期格式的taxonomy-post\_format-post-format-link.php。
- taxonomy-{taxonomy}.php – 如果分类是多数，WordPress会寻找分类法 - sometax.php。
- taxonomy.php
- archive.php
- index.php

## 自定义内容类型

---

自定义帖子类型使用以下路径呈现相应的归档索引页面。

- archive- {post\_type} .php - 如果帖子类型是产品，WordPress将寻找archive-product.php。
- archive.php
- index.php

（要渲染单个帖子类型模板，请参阅上面的单个帖子显示部分。）

## 作者显示

---

基于上述示例，渲染作者存档索引页面是相当明确的：

- author- {nickname} .php - 如果作者的漂亮的名字是哑光，WordPress将寻找author-matt.php。



- author- {id} .php - 如果作者的ID为6，WordPress将寻找author-6.php。
- author.php
- archive.php
- index.php

## 日期

---

基于日期的归档索引页面按照您预期的方式呈现：

- date.php
- archive.php
- index.php

## 搜索结果

---

搜索结果遵循与其他模板类型相同的模式：

- search.php
- index.php

## 404 ( 找不到 )

---

同样，404模板文件按以下顺序调用：

- 404.php
- index.php

## 附件

---

渲染附件页面（附件类型）需要遵循以下路径：

- MIME\_type.php - 它可以是任何MIME类型（例如：image.php，video.php，application.php）。对于text / plain，使用以下路径（顺序）：
- text\_plain.php
- plain.php
- text.php
- attachment.php
- single-attachment.php
- single.php
- index.php

## 嵌入模板

---

嵌入模板文件用于渲染正在嵌入的帖子。自4.5以来，WordPress使用以下路径：

- embed- {post-type} - {post\_format} .php - 首先，WordPress寻找特定帖子的模板。例如，如果其帖子类型是产品，并且具有音频格式，则WordPress将寻找embed-product-audio.php。
- embed- {post-type} .php - 如果帖子类型是汽车，WordPress会寻找embed-car.php。
- embed.php - WordPress然后回到embed.php。
- 最后，WordPress最终还是回到wp-includes/theme-compat/embed.php

## 过滤层次结构

---

WordPress模板系统允许您过滤层次结构。这意味着您可以在层次结构的特定点插入和更改东西。过滤器（位于get\_query\_template（）函数中）使用此过滤器名称：“{\$ type} \_template” 其中\$ type是没有.php扩展名的层次结构中的文件名。

以下是过滤器层次结构中所有模板类型的完整列表：

- index\_template
- 404\_template
- archive\_template
- author\_template
- category\_template
- tag\_template
- taxonomy\_template
- date\_template
- home\_template
- front\_page\_template
- page\_template
- paged\_template
- search\_template
- single\_template
- text\_template, plain\_template, text\_plain\_template (all mime types)
- attachment\_template
- comments\_popup
- embed\_template

## 示例

例如，让我们采用默认的作者层次结构：

- author-{nickname}.php

- author-{id}.php
- author.php

要在author.php之前添加author- {role} .php，我们可以使用'author\_template'模板类型来操作实际的层次结构。这允许对/ author/username的请求，其中username具有编辑器的作用，使用author-editor.php（如果存在于当前主题目录中）显示。

```
function author_role_template( $templates = '' ) {
    $author = get_queried_object();
    $role = $author->roles[0];

    if ( ! is_array( $templates ) && ! empty( $templates ) ) {
        $templates = locate_template( array( "author-$role.php", $templates ),
false );
    } elseif ( empty( $templates ) ) {
        $templates = locate_template( "author-$role.php", false );
    } else {
        $new_template = locate_template( array( "author-$role.php" ) );
        if ( ! empty( $new_template ) ) {
            array_unshift( $templates, $new_template );
        }
    }

    return $templates;
}
add_filter( 'author_template', 'author_role_template' );
```

# 模板标签

---

主题中使用模板标签来从数据库中检索内容。内容可以是从小博客标题到完整侧边栏的任何内容。模板标签是将内容拉入主题的首选方法，因为：

- 他们可以打印动态内容;
- 它们可以用于多个主题文件; 和
- 他们将主题分为更小，更易理解的部分。

## 什么是模板标签？

---

模板标签只是一段代码，可以让WordPress从数据库中获取一些内容。它分为三个组成部分：

- 一个PHP代码标签
- WordPress函数
- 可选参数

您可以使用模板标签来调用另一个主题文件或数据库中的某些信息。

例如，模板标签`get_header()`告诉WordPress获取`header.php`文件并将其包含在当前主题文件中。同样，`get_footer()`告诉WordPress获取`footer.php`文件。

还有其他种类的模板标签：

- `the_title()` - 告诉WordPress从数据库中获取页面或帖子的标题，并将其包含。
- `bloginfo('name')` - 告诉WordPress将博客标题从数据库中取出并将其包含在模板文件中。

如果仔细观察最后一个例子，您还将看到括号之间有一个参数。参数可以让你做两件事情：

要求具体的信息和

以某种方式格式化信息。

下面广泛的介绍了这些参数，但是请注意，您可以发送WordPress特定的说明，了解您希望显示数据的方式。

## 为什么要使用模板标签

---

通过封装特定内容块的所有代码，模板标签使得在主题文件中包含模板的各种部分以及维护主题非常容易。

创建一个`header.php`文件并拥有所有的主题模板，如`single.php`，`page.php`，`front-page.php`等，引用一个使用`get_header()`的主题文件比复制和粘贴代码更容易 进入每个主题文件。它也使维护更容易。每当您在`header.php`文件中进行更改时，更改将自动转移到所有其他主题文件中。

使用模板标签的另一个原因是显示动态数据，即来自数据库的数据。在标题中，您可以手动添加标题标签，如下所示：

```
<title>My Personal Website</title>
```

但是，这样做意味着您随时要更改网站的标题时手动编辑主题。相反，更容易地包含**bloginfo('name')**模板标签，该标签自动从数据库中提取站点标题。现在，您可以在WordPress中更改您的网站的标题，而不必对主题模板进行硬编码。

## 如何使用模板标签

使用模板标签非常简单。在任何模板文件中，您可以通过简单地打印一行php代码来调用模板标签来使用模板标签。打印header.php文件很简单：

```
get_header();
```

## 参数

一些模板标签可以让您传递参数。参数是确定从数据库检索到的内容的额外信息。

例如，**bloginfo()**模板标签允许你给它一个参数，告诉WordPress你想要的具体的信息。要打印博客名称，您只需传递参数“name”，如下所示：

```
bloginfo( 'name' );
```

要打印博客正在运行的WordPress版本，您将传递一个参数“version”：

```
bloginfo( 'version' );
```

对于每个模板标签，参数不同。可以在代码引用的特定模板标签页上找到参数列表以及可以做的事情。

## 在循环中使用模板标签

许多模板标签在WordPress循环中工作。这意味着它们被包含在模板文件中，作为php“循环”的一部分，它根据循环内的指令生成用户看到的页面。

WordPress循环以：

```
if ( have_posts() ) :  
    while ( have_posts() ) :  
        the_post();
```

在循环中工作的模板标签必须位于以下循环结束部分之前的中间区域：

```
        endwhile;  
    else :  
        _e( 'Sorry, no posts matched your criteria.', 'devhub' );  
    endif;
```

需要在循环内部的一些模板标签包括

- `the_content()`
- `the_excerpt()`
- `next_post()`
- `previous_post()`

某些功能需要循环的主要原因是因为它们需要设置全局后置对象。

如果要使用的模板标签不必在循环中

- `wp_list_cats()`
- `wp_list_pages()`

那么你可以把它放在你想要的任何文件中，例如在边栏，页眉或页脚模板文件中。

这些是通常不需要全局后置对象的函数。

## 附件

- [条件标签](#)
- [完整的模板标签列表](#)

# 循环

循环是WordPress用于通过主题的模板文件输出帖子的默认机制。检索的帖子数量取决于阅读设置中定义的每页显示的帖子数量。在循环中，WordPress将检索要显示在当前页面上的每个帖子，并根据您的主题说明进行格式化。

循环从WordPress数据库中提取每个帖子的数据，并插入适当的信息代替每个模板标签。将为每个帖子处理循环中的任何HTML或PHP代码。

简单来说，循环是它的名字：它循环遍历当前页面检索到的每个帖子一次，并执行您的主题中指定的操作。

您可以使用循环来执行多种不同的操作，例如：

- 在您的博客主页上显示帖子标题和摘录;
- 在单个帖子上显示内容和评论;
- 使用模板标签在单个页面上显示内容; 和
- 显示来自Custom Post Types和Custom Fields的数据。
- 您可以在模板文件中自定义循环，以显示和操作不同的内容。

## 循环详情

基本循环是：

```
<?php if ( have_posts() ) : ?>
    <?php while ( have_posts() ) : the_post(); ?>
        ... Display post content
    <?php endwhile; ?>
<?php endif; ?>
```

这个循环说，当有帖子时，循环并显示帖子。详细介绍：

have\_posts()函数检查是否有任何帖子。

如果有帖子，只要括号中的条件在逻辑上为真，则while循环将继续执行。只要have\_posts()继续为true，循环将继续。

## 使用循环

循环应放在index.php中，以及用于显示帖子信息的任何其他模板中。因为你不想一遍又一遍地复制你的头，循环应该总是在调用get\_header()之后放置。例如：

```
<?php get_header(); ?>
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
    ... Display post content
<?php endwhile; endif; ?>
```

在上面的例子中，循环的结尾显示为一个结尾和endif。循环必须始终以相同的if和while语句开始，如上所述，并且必须以相同的结束语句结尾。

您要应用于所有帖子的任何模板标签必须存在于开始和结束语句之间。

**提示：**如果没有符合指定条件的帖子可用，您可以添加自定义的404“未找到”消息。消息必须放在endwhile和endif语句之间，如下面的示例所示。

一个非常简单的index.php文件将如下所示：

```
<?php
    get_header();
    if ( have_posts() ) : while ( have_posts() ) : the_post();
        the_content();
    endwhile;
    else :
        _e( 'Sorry, no posts matched your criteria.', 'textdomain' );
    endif;
    get_sidebar();
    get_footer();
?>
```

## 循环可以显示

循环可以为每个帖子显示多个不同的元素。例如，许多主题中使用的常见模板标签是：

- next\_post\_link() – 在这篇文章后按时间顺序发布的一篇链接
- previous\_post\_link() – 在这篇文章之前根据时间顺序发布了一篇链接
- the\_category() – 与正在查看的帖子或页面相关联的类别或类别
- the\_author() – 作者的帖子或页面
- the\_content() – 页面的主要内容
- the\_excerpt() – 一个帖子的主要内容的前55个字，后跟一个省略号（...）或阅读更多链接到完整的帖子。您还可以使用帖子的“摘录”字段来自定义特定摘录的长度。
- the\_ID() – 帖子或页面的ID
- the\_meta() – 与帖子或页面关联的自定义字段
- the\_shortlink() – 使用网站的网址和帖子或页面的ID链接到页面或帖子
- the\_tags() – 与帖子相关联的标签或标签
- the\_title() – 帖子或页面的标题
- the\_time() – 帖子或页面的时间或日期。这可以使用标准php日期功能格式化来定制。

您还可以使用条件标签，例如：



- `is_home()` – 如果当前页面是主页，则返回true
- `is_admin()` – 如果是管理员，返回true，否则返回false
- `is_single()` – 如果页面当前显示单个帖子，则返回true
- `is_page()` – 如果页面当前显示单个页面，则返回true
- `is_page_template()` – 可用于确定页面是否正在使用特定的模板，例如：`is_page_template ( 'about-page.php' )`
- `is_category()` – 如果页面或帖子具有指定的类别，则返回true，例如：`is_category ( 'news' )`
- `is_tag()` – 如果页面或帖子具有指定的标签，则返回true
- `is_author()` – 如果在作者的归档页面内返回true
- `is_search()` – 如果当前页面是搜索结果页面，则返回true
- `is_404()` – 如果当前页不存在，则返回true
- `has_excerpt()` – 如果帖子或页面有摘录，则返回true

## 示例

让我们来看一下循环中的一些例子：

### 基本例子

#### 博客归档

大多数博客都有一个博客归档页面，可以显示一些内容，包括帖子标题，缩略图和摘录。下面的示例显示了一个简单的循环，检查是否有任何帖子，如果有的话，输出每个帖子的标题，缩略图和摘录。如果没有帖子，它会以括号显示消息。

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

    <h2><?php the_title(); ?></h2>
    <?php the_post_thumbnail(); ?>
    <?php the_excerpt(); ?>
<?php endwhile; else: ?>
    <?php _e( 'Sorry, no posts matched your criteria.', 'textdomain' ); ?>
<?php endif; ?>
```

### 个人帖子

在WordPress中，每个帖子都有自己的页面，显示该帖子的相关信息。模板标签允许您自定义要显示的信息。在下面的例子中，循环输出帖子的标题和内容。您可以在帖子或页面模板文件中使用此示例来显示有关该帖子的最基本信息。您还可以自定义此模板以向帖子添加更多数据，例如类别。

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

<h1><?php the_title(); ?></h1>
    <?php the_content(); ?>
<?php endwhile; else: ?>
    <?php _e( 'Sorry, no pages matched your criteria.', 'textdomain' ); ?>
<?php endif; ?>
```

## 中间例子

### 不同类别的风格帖子

下面的例子有几件事情：

首先，它显示每个帖子的标题，时间，作者，内容和类别，类似于上面的单个帖子示例。

接下来，通过使用in\_category()模板标签，可以使类别ID为“3”的帖子的样式不同。

此示例中的代码注释在循环的每个阶段提供了详细信息：

```
// Start the Loop.
<?php if ( have_posts() ) : while ( have_posts() ) : the_post();
/* * See if the current post is in category 3.
 * If it is, the div is given the CSS class "post-category-three".
 * Otherwise, the div is given the CSS class "post".
 */
if ( in_category( 3 ) ) : ?>

<div class="post-category-three">
    <?php else : ?>
<div class="post">
    <?php endif; ?>

    // Display the post's title.
    <h2><?php the_title() ?></h2>

    // Display a link to other posts by this posts author.
    <small><?php _e( 'Posted by ', 'textdomain' ); the_author_posts_link() ?
></small>

    // Display the post's content in a div.
    <div class="entry">
        <?php the_content() ?>
    </div>

    // Display a comma separated list of the post's categories.
    <?php _e( 'Posted in ', 'textdomain' ); the_category( ' ', ' ' ); ?>

    // closes the first div box with the class of "post" or "post-cat-three"
```

```

</div>

// Stop the Loop, but allow for a "if not posts" situation

<?php endwhile; else :
/*
 * The very first "if" tested to see if there were any posts to
 * display. This "else" part tells what do if there weren't any.
 */
_e( 'Sorry, no posts matched your criteria.', 'textdomain' );
// Completely stop the Loop.
endif;
?>

```

## 删除此部分

注意：此部分需要删除，但存在是因为手册插件中的错误无法正确显示主题框中的下一部分。

### 多重循环

在某些情况下，您可能需要使用多个循环。例如，您可能希望在页面顶部内容列表表中显示帖子的标题，然后在页面上进一步显示内容。由于查询没有被改变，所以我们需要在第二次循环访问这些信息时，回滚循环。为此，我们将使用函数`rewind_posts()`。

#### 使用`rewind_posts()`

您可以再次使用`rewind_posts()`循环遍历相同的查询。如果要在页面上的不同位置显示相同的查询两次，这将非常有用。

以下是正在使用的`rewind_posts()`的示例：

```

// Start the main loop
<?php
    if ( have_posts() ) : while ( have_posts() ) : the_post();
        the_title();
    endwhile;
endif;

// Use rewind_posts() to use the query a second time.
rewind_posts();

// Start a new loop
while ( have_posts() ) : the_post();
    the_content();
endwhile;
?>

```

## 创建辅助查询和循环

使用具有相同查询的两个循环相对容易，但并不总是您需要的。相反，您通常会创建一个辅助查询来在模板上显示不同的内容。例如，您可能希望在同一页面上显示两组帖子，但对每个组执行不同的操作。如下所示，一个常见的例子是显示单个帖子，其中包含单个帖子下方相同类别的帖子列表。

```
<?php
// The main query.
if ( have_posts() ) : while ( have_posts() ) : the_post();
    the_title();

    the_content();

endwhile;

else :

    // When no posts are found, output this text.

    _e( 'Sorry, no posts matched your criteria.' );

endif;

wp_reset_postdata();

/*

 * The secondary query. Note that you can use any category name here. In our
 * example,
 * we use "example-category".

 */
$secondary_query = new WP_Query( 'category_name=example-category' );

// The second loop. if ( $secondary_query->have_posts() )
echo '<ul>';
while ( $secondary_query->have_posts() ) :
    $secondary_query->the_post();
    echo '<li>' . get_the_title() . '</li>';
endwhile;
echo '</ul>';
endif;
wp_reset_postdata();
```

```
?>
```

如上例所示，我们首先显示一个常规循环。然后我们定义一个使用WP\_Query查询特定类别的新变量；在我们的例子中，我们选择了示例类别的插件。

请注意，上述示例中的常规循环有一个区别：它调用wp\_reset\_postdata()来重置帖子数据。在您可以使用第二个循环之前，您需要重置帖子数据。有两种方法可以做到这一点：

- 通过使用rewind\_posts()函数；要么
- 通过创建新的查询对象。

## 重置多个循环

在您重置它们的模板中使用多个循环时，这很重要。不这样做可能会导致意外的结果，因为数据在全局\$post变量中的存储和使用。有三种主要的方法来重置循环，具体取决于它们的调用方式。

- wp\_reset\_postdata()
- wp\_reset\_query()
- rewind\_posts()

### 使用wp\_reset\_postdata()

当您使用WP\_Query运行自定义或多个循环时，请使用wp\_reset\_postdata()。此函数将全局\$post变量恢复到主查询中的当前帖子。如果您遵循最佳做法，这是您将用来重置循环的最常用功能。

要正确使用此功能，请在使用WP\_Query的任何循环后放置以下代码：

```
<?php wp_reset_postdata(); ?>
```

这是一个使用WP\_Query的循环的示例，它使用wp\_reset\_postdata()重置。

```
<?php
// Example argument that defines three posts per page.
$args = array( 'posts_per_page' => 3 );

// Variable to call WP_Query.
$query = new WP_Query( $args );

if ( $query->have_posts() ) :
    // Start the Loop
    while ( $query->have_posts() ) : $query->the_post();
        the_title();
        the_excerpt();
    endwhile;
endwhile;
```

```
// End the Loop
endwhile;
else:
    // If no posts match this query, output this text.
    _e( 'Sorry, no posts matched your criteria.', 'textdomain' );
endif;
wp_reset_postdata();

?>
```

## 使用wp\_reset\_query()

使用wp\_reset\_query()将WP\_Query和全局\$post数据恢复到原始主查询。如果在循环中使用query\_posts()，则必须使用此功能来重置循环。您可以在使用WP\_Query自定义循环后使用它，因为它在运行时实际调用wp\_reset\_postdata()。然而，最好的做法是使用wp\_reset\_postdata()与涉及WP\_Query的任何自定义循环。

请注意：query\_posts()不是最佳实践，如果可能的话应该避免。因此，您不应该对wp\_reset\_query()有很多用处。

要正确使用此功能，请在使用query\_posts()的任何循环后面放置以下代码。

```
<?php wp_reset_query(); ?>
```

# 主题函数

functions.php文件是您向WordPress主题添加独特功能的地方。它可以用于挂接WordPress的核心功能，使您的主题更具模块化，可扩展性和功能性。

## 什么是functions.php？

functions.php文件的行为就像一个WordPress插件，向WordPress网站添加功能和功能。您可以使用它来调用WordPress函数并定义自己的功能。

**注意：**使用插件或functions.php可以生成相同的结果。如果您正在创建应该可用的新功能，无论网站如何，最好将其放入插件。

使用WordPress插件或使用functions.php有优势和折衷。

WordPress插件：

- 需要具体的，唯一的标题文字；
- 存储在wp-content/plugins中，通常在子目录中；
- 激活时仅在页面加载时执行；
- 适用于所有主题；并且应该有一个目的 - 例如，提供搜索引擎优化功能或帮助备份。

同时，一个functions.php文件：

- 不需要唯一的标题文字；
- 存储在wp-content/themes中的主题的子目录中；
- 仅在活动主题的目录中执行；
- 仅适用于该主题（如果主题已更改，则不再使用该功能）；并且可以有許多代码块用于许多不同的目的。
- 每个主题都有自己的函数文件，但只有活动主题的functions.php中的代码才能实际运行。如果你的主题已经有一个功能文件，你可以添加代码。如果没有，您可以创建一个名为functions.php的纯文本文件，以添加到您的主题目录中，如下所述。

子主题可以有自己的functions.php文件。将函数添加到子函数文件是修改父主题的无风险方式。这样，当父主题更新时，您不必担心新添加的功能会消失。

**注意：**虽然子主题的functions.php是在父主题的functions.php之前由WordPress加载的，但它不会覆盖它。子主题的functions.php可用于增加或替换父主题的功能。同样，在加载任何插件文件之后，会加载functions.php。

使用functions.php可以：

- 使用WordPress挂钩。例如，使用excerpt\_length过滤器，您可以更改您的职位摘录长度（默认为55个字）。
- 使用add\_theme\_support()启用WordPress功能。例如，打开帖子缩略图，帖子格式和导航菜单。
- 定义要在多个主题模板文件中重用的功能。

**警告：**如果WordPress插件调用相同的功能或过滤器，就像您在functions.php中所做的那样，结果可能是意外的，甚至导致您的站点被禁用。

## 示例

以下是您可以在functions.php文件中使用的一些示例，以支持各种功能。如果您选择将其提交到WordPress.org主题目录，则这些示例中的每一个都可以在您的主题中使用。

## 主题设置

一些主题功能应该包含在一个“设置”功能中，最初在您的主题被激活时运行。如下图所示，这些功能可以添加到您的functions.php文件中，以激活推荐的WordPress功能。

**注意：**用你的主题名命名你的函数很重要。以下所有示例都使用myfirsttheme\_作为其命名空间，它们应根据您的主题名称进行自定义。

要创建此初始函数，请启动一个名为myfirsttheme\_setup()的新函数，如下所示：

```
if ( ! function_exists( 'myfirsttheme_setup' ) ) :  
/**  
 * Sets up theme defaults and registers support for various WordPress features  
 *  
 * It is important to set up these functions before the init hook so that none  
of these  
 * features are lost.  
 *  
 * @since MyFirstTheme 1.0  
 */  
function myfirsttheme_setup() {
```

**注意：**在上面的例子中，函数myfirsttheme\_setup启动但未关闭。确保关闭您的功能

## automatic-feed-links

默认情况下，自动Feed链接可以发布和评论RSS Feed。这些Feed将自动显示在中。可以使用add\_theme\_support()调用它们。

```
add_theme_support( 'automatic-feed-links' );
```



## 导航菜单

自定义导航菜单允许用户在“菜单”管理面板中编辑和自定义菜单，为用户提供了一个拖放界面来编辑其主题中的各种菜单。

您可以在functions.php中设置多个菜单。可以使用register\_nav\_menus()添加它们，并使用wp\_nav\_menu()插入主题，如本手册后面所述。如果您的主题将允许多个菜单，则应使用数组。虽然某些主题将不具有自定义导航菜单，但建议您允许此功能轻松进行自定义。

```
register_nav_menus( array(  
    'primary'    => __( 'Primary Menu', 'myfirsttheme' ),  
    'secondary' => __( 'Secondary Menu', 'myfirsttheme' )  
) );
```

您可以稍后使用wp\_nav\_menu()并使用分配的名称（即主）将其定义的每个菜单作为theme\_location参数。

## 加载文本域

通过使您的主题中的字符串可用于翻译，主题可以翻译成多种语言。为此，您必须使用load\_theme\_textdomain()。有关使您的主题可用于翻译的更多信息，请阅读[国际化](#)部分。

```
load_theme_textdomain( 'myfirsttheme', get_template_directory() . '/languages' )  
;
```

## 发布缩略图

发布缩略图和精选图片可让您的用户选择一个图片来表示他们的帖子。您的主题可以根据其设计决定如何显示它们。例如，您可以选择在归档视图中显示每个帖子的帖子缩略图。或者，您可能希望在主页上使用大型特色图片。虽然不是每个主题都需要特色图片，但建议您支持发布缩略图和精选图片。

```
add_theme_support( 'post-thumbnails' );
```

## 发布格式

发布格式允许用户以不同的方式格式化其帖子。这对于允许博主根据帖子的内容选择不同的格式和模板非常有用。add\_theme\_support()也用于Post格式。这是推荐的。

```
add_theme_support( 'post-formats', array ( 'aside', 'gallery', 'quote', 'image'  
    , 'video' ) );
```

进一步了解[发布格式](#)。

## 初始设置示例

包括所有上述功能将给你一个如下所示的functions.php文件。 添加了代码注释以便将来的清晰度。

如本示例底部所示，您必须添加所需的add\_action()语句以确保myfirsttheme\_setup函数已加载。

```
if ( ! function_exists( 'myfirsttheme_setup' ) ) :
/**
 * Sets up theme defaults and registers support for various WordPress features.
 *
 * Note that this function is hooked into the after_setup_theme hook, which runs
 * before the init hook. The init hook is too late for some features, such as indicating
 * support post thumbnails.
 */
function myfirsttheme_setup() {

    /**
     * Make theme available for translation.
     * Translations can be placed in the /languages/ directory.
     */
    load_theme_textdomain( 'myfirsttheme', get_template_directory() . '/languages' );

    /**
     * Add default posts and comments RSS feed links to <head>.
     */
    add_theme_support( 'automatic-feed-links' );

    /**
     * Enable support for post thumbnails and featured images.
     */
    add_theme_support( 'post-thumbnails' );

    /**
     * Add support for two custom navigation menus.
     */
    register_nav_menus( array(
        'primary'    => __( 'Primary Menu', 'myfirsttheme' ),
        'secondary' => __( 'Secondary Menu', 'myfirsttheme' )
    ) );

    /**
     * Enable support for the following post formats:
     * aside, gallery, quote, image, and video
     */
    add_theme_support( 'post-formats', array ( 'aside', 'gallery', 'quote', 'image', 'video' ) );
}
```

```
age', 'video' ) );
}
endif; // myfirsttheme_setup
add_action( 'after_setup_theme', 'myfirsttheme_setup' );
```

## 内容宽度

内容宽度添加到您的functions.php文件中，以确保没有内容或资源破坏站点的容器。 内容宽度为添加到您的网站的任何内容（包括已上传的图像）设置允许的最大宽度。 在下面的示例中，内容区域的最大宽度为800像素。没有内容会比这更大。

```
if ( ! isset ( $content_width ) )
    $content_width = 800;
```

## 其他特性

还有其他常见功能可以在functions.php中包含。 下面列出了一些最常见的功能。 点击并了解有关这些功能的更多信息。

（根据新页面展开此部分。）

- [自定义Headers](#)
- [Sidebars \( Widgets \)](#)
- 自定义背景
- 添加编辑器样式
- HTML5
- 标题标签

## 你的functions.php文件

如果您选择包括上面列出的所有功能，这是您的functions.php可能是什么样子。 已经参考上面的评论。

```
/**
 * MyFirstTheme's functions and definitions
 *
 * @package MyFirstTheme
 * @since MyFirstTheme 1.0
 */

/**
 * First, let's set the maximum content width based on the theme's design and s
tylesheet.
 * This will limit the width of all uploaded images and embeds.
 */
```

```

if ( ! isset( $content_width ) )
    $content_width = 800; /* pixels */

if ( ! function_exists( 'myfirsttheme_setup' ) ) :
/**
 * Sets up theme defaults and registers support for various WordPress features.
 *
 * Note that this function is hooked into the after_setup_theme hook, which runs
 * before the init hook. The init hook is too late for some features, such as indicating
 * support post thumbnails.
 */
function myfirsttheme_setup() {

    /**
     * Make theme available for translation.
     * Translations can be placed in the /languages/ directory.
     */
    load_theme_textdomain( 'myfirsttheme', get_template_directory() . '/languages' );

    /**
     * Add default posts and comments RSS feed links to <head>.
     */
    add_theme_support( 'automatic-feed-links' );

    /**
     * Enable support for post thumbnails and featured images.
     */
    add_theme_support( 'post-thumbnails' );

    /**
     * Add support for two custom navigation menus.
     */
    register_nav_menus( array(
        'primary'    => __( 'Primary Menu', 'myfirsttheme' ),
        'secondary' => __( 'Secondary Menu', 'myfirsttheme' )
    ) );

    /**
     * Enable support for the following post formats:
     * aside, gallery, quote, image, and video
     */
    add_theme_support( 'post-formats', array ( 'aside', 'gallery', 'quote', 'image', 'video' ) );
}
endif; // myfirsttheme_setup
add_action( 'after_setup_theme', 'myfirsttheme_setup' );

```

# 连接主题文件和目录

## 链接到核心主题文件

正如你所学到的，WordPress主题是从许多不同的模板文件构建的。至少这通常会包含一个sidebar.php，header.php和footer.php。这些被称为使用模板标签，例如：

- `get_header();`
- `get_footer();`
- `get_sidebar();`

您可以通过命名文件sidebar-{your\_custom\_template}.php，header-{your\_custom\_template}.php和footer-{your\_custom\_template}.php来创建这些文件的自定义版本。然后，您可以使用模板标签与自定义模板名称作为唯一的参数，如下所示：

- `get_header( 'your_custom_template' );`
- `get_footer( 'your_custom_template' );`
- `get_sidebar( 'your_custom_template' );`

WordPress通过组合各种文件创建页面。除了标题，页脚和侧边栏的标准文件外，您可以创建自定义模板文件，并使用`get_template_part()`在页面中的任何位置调用它们。要在主题中创建自定义模板文件，请为文件提供适当的名称，并使用与标题，侧边栏和页脚文件相同的自定义模板系统：

```
slug-template.php
```

例如，如果要创建自定义模板来处理您的帖子内容，您可以创建一个名为content.php的模板文件，然后通过将文件名扩展为content-product.php来为产品内容添加特定的内容布局。然后，您将在主题中加载此模板文件，如下所示：

```
get_template_part( 'content', 'product' );
```

如果您想添加更多的组织到您的模板，您可以将它们放置在您的主题目录中的自己的目录中。例如，假设您为配置文件和位置添加了一些更多的内容模板，并将它们分组在其名为content-templates的目录中。称为my-theme的主题的主题层次结构可能如下所示。style.css和page.php都包含在上下文中。

- themes
  - my-theme
    - content-templates

- content-location.php
- content-product.php
- content-profile.php
- style.css

要包括您的内容模板，请将目录名称添加到slug参数中，如下所示：

```
get_template_part( 'content-templates/content', 'location' );
get_template_part( 'content-templates/content', 'product' );
get_template_part( 'content-templates/content', 'profile' );
```

## 链接到主题目录

要链接到主题的目录，您可以使用以下功能：

```
get_theme_file_uri();
```

If you are not using a child theme, this function will return the full URI to your theme's main folder. You can use this to reference sub-folders and files in your theme like this:

```
echo get_theme_file_uri( 'images/logo.png' );
```

如果您使用的是一个子主题，那么该函数将返回您的子主题中的文件的URI（如果存在）。如果您的子主题中找不到文件，该函数将返回父主题中的文件的URI。在分发主题或其他情况下，儿童主题可能活动或可能不活跃时，这一点特别重要。

要访问主题目录中文件的路径，可以使用以下功能：

```
get_theme_file_path();
```

像get\_theme\_file\_uri()一样，如果子主题存在，它将访问该文件的路径。如果在子主题中找不到文件，该函数将访问父主题中文件的路径。

在子主题中，您可以使用以下功能链接到父主题目录中的文件URI或路径：

- get\_parent\_theme\_file\_uri();
- get\_parent\_theme\_file\_path();

与get\_theme\_file\_uri()一样，您可以参考这样的子文件夹和文件：

```
echo get_parent_theme_file_uri( 'images/logo.png' );
```

```
//or  
echo get_parent_theme_file_path( 'images/logo.png' );
```

引用可能不存在的文件时要小心，因为这些函数将返回URI或文件路径，无论文件是否存在。如果文件丢失，这些功能将返回一个断开的链接。

注意：在WordPress 4.7中引入了函数`get_theme_file_uri()`，`get_theme_file_path()`，`get_parent_theme_file_uri()`，`get_parent_theme_file_path()`。

对于以前的WordPress版本，请使用`get_template_directory_uri()`，`get_template_directory()`，`get_stylesheet_directory_uri()`，`get_stylesheet_directory()`。

请注意，较新的4.7函数作为检查过程的一部分运行较旧的功能，因此在可能时使用较新的功能是有意义的。

## 模板中的动态链接

无论您的永久链接设置如何，您可以通过参考其唯一的数字ID（在管理界面的几个页面中）动态链接到页面或发布页面

```
<a href="<?php echo get_permalink($ID); ?>">This is a link</a>
```

这是创建页面菜单的便捷方式，因为您可以稍后更改页面段，而不会中断链接，因为ID将保持不变。但是，这可能会增加数据库查询。

# 使用CSS和JavaScript

创建主题时，您可能需要创建其他样式表或JavaScript文件。但是，请记住，WordPress网站不会仅仅使您的主题成为活动，它也将使用许多不同的插件。所以一切工作和谐，重要的是主题和插件使用标准的WordPress方法加载脚本和样式表。这将确保网站保持高效，并且没有不兼容的问题。

向WordPress添加脚本和样式是一个相当简单的过程。本质上，您将创建一个将排列所有脚本和样式的函数。当插入脚本或样式表时，WordPress会创建一个句柄和路径来查找您的文件及其可能具有的任何依赖关系（如jQuery），然后您将使用一个将插入脚本和样式表的钩子。

## 启动脚本和样式

为主题添加脚本和样式的正确方法是将它们排入functions.php文件。所有主题都需要style.css文件，但可能需要添加其他文件来扩展主题的功能。

**提示：**WordPress包含许多JavaScript文件作为软件包的一部分，包括常用的库，如jQuery。在添加您自己的JavaScript之前，请检查是否可以使用附带的库。

基本情况是：

使用wp\_enqueue\_script()或wp\_enqueue\_style()来排队脚本或样式

## 样式

您的CSS样式表用于自定义您的主题的演示文稿。样式表也是存储有关您的主题信息的文件。因此，每个主题都需要style.css文件。

而是将样式表加载到header.php文件中，您应该使用wp\_enqueue\_style加载样式表。为了加载主体样式表，可以在functions.php中排队

排入style.css

```
wp_enqueue_style( 'style', get_stylesheet_uri() );
```

这将查找名为“style”的样式表并加载它。

排列风格的基本功能是：

```
wp_enqueue_style( $handle, $src, $deps, $ver, $media );
```

您可以包括以下参数：

- \$handle只是样式表的名称。



- \$src是它所在的位置。 其余的参数是可选的。
- \$deps是指这个样式表是否依赖于另一个样式表。 如果设置了此样式表，则不会加载此样式表，除非首先加载其依赖样式表。
- \$ver设置版本号。
- \$media可以指定加载此样式表的媒体类型，例如 “全部” ， “屏幕” ， “打印” 或 “掌上电脑” 。

因此，如果要在主题根目录中的名为 “CSS” 的文件夹中加载名为 “slider.css” 的样式表，你会使用：

```
wp_enqueue_style( 'slider', get_template_directory_uri() . '/css/slider.css', false, '1.1', 'all');
```

## 脚本

应该使用wp\_enqueue\_script加载主题所需的任何其他JavaScript文件。 这样可确保适当的加载和缓存，并允许使用条件标签来定位特定的页面。 这些是可选的。

wp\_enqueue\_script使用与wp\_enqueue\_style类似的语法。

排列你的脚本：

```
wp_enqueue_script( $handle, $src, $deps, $ver, $in_footer);
```

- \$handle是脚本的名称。
- \$src定义脚本所在的位置。
- \$deps是一个数组，可以处理您的新脚本依赖的任何脚本，如jQuery。
- \$ver可以列出版本号。
- \$in\_footer是一个布尔参数（ true / false ），允许您将脚本放置在HTML文档的页脚中，而不是标题中，以免延迟加载DOM树。

您的排队功能可能如下所示：

```
wp_enqueue_script( 'script', get_template_directory_uri() . '/js/script.js', array ( 'jquery' ), 1.1, true);
```

## 评论回复脚本

WordPress的评论在他们的开箱即用功能，包括线程评论和增强的注释表单。 为了使注释正常工作，它们需要一些JavaScript。 但是，由于需要在此JavaScript中定义某些选项，所以应将注释回复脚本添加到使用注释的每个主题上。

包含评论回复的正确方法是使用条件标签来检查某些条件是否存在，以便脚本不会被不必要地加载。 例如，您只

能使用is\_singular在单个帖子页面上加载脚本，并检查以确保用户选择“启用线程注释”。所以你可以设置一个函数，如：

```
if ( is_singular() && comments_open() && get_option( 'thread_comments' ) ) {
    wp_enqueue_script( 'comment-reply' );
}
```

如果用户启用了注释，并且我们在一个帖子页面上，则会加载评论回复脚本。否则，不会。

## 组合入队函数

最好将所有入队的脚本和样式组合成一个单一的函数，然后使用wp\_enqueue\_scripts操作调用它们。该功能和操作应位于初始设置下方（执行上述）。

```
function add_theme_scripts() {
    wp_enqueue_style( 'style', get_stylesheet_uri() );

    wp_enqueue_style( 'slider', get_template_directory_uri() . '/css/slider.css',
array(), '1.1', 'all' );

    wp_enqueue_script( 'script', get_template_directory_uri() . '/js/script.js',
array ( 'jquery' ), 1.1, true );

    if ( is_singular() && comments_open() && get_option( 'thread_comments' ) ) {

        wp_enqueue_script( 'comment-reply' );
    }
}
add_action( 'wp_enqueue_scripts', 'add_theme_scripts' );
```

## WordPress包含并注册的默认脚本

默认情况下，WordPress包括Web开发人员通常使用的许多流行脚本，以及WordPress本身使用的脚本。其中一些列在下表中：

脚本名称	处理	需要依赖*
Image Cropper	Image cropper (not used in core, see jcrop)	
Jcrop	jcrop	
SWFObject	swfobject	
SWFUpload	swfupload	
SWFUpload Degrade	swfupload-degrade	

SWFUpload Queue	swfupload-queue	
SWFUpload Handlers	swfupload-handlers	
jQuery	jquery	json2 (for AJAX calls)
jQuery Form	jquery-form	jquery
jQuery Color	jquery-color	jquery
jQuery Masonry	jquery-masonry	jquery
jQuery UI Core	jquery-ui-core	jquery
jQuery UI Widget	jquery-ui-widget	jquery
jQuery UI Mouse	jquery-ui-mouse	jquery
jQuery UI Accordion	jquery-ui-accordion	jquery
jQuery UI Autocomplete	jquery-ui-autocomplete	jquery
jQuery UI Slider	jquery-ui-slider	jquery
jQuery UI Progressbar	jquery-ui-progressbar	jquery
jQuery UI Tabs	jquery-ui-tabs	jquery
jQuery UI Sortable	jquery-ui-sortable	jquery
jQuery UI Draggable	jquery-ui-draggable	jquery
jQuery UI Droppable	jquery-ui-droppable	jquery
jQuery UI Selectable	jquery-ui-selectable	jquery
jQuery UI Position	jquery-ui-position	jquery
jQuery UI Datepicker	jquery-ui-datepicker	jquery
jQuery UI Tooltips	jquery-ui-tooltip	jquery
jQuery UI Resizable	jquery-ui-resizable	jquery
jQuery UI Dialog	jquery-ui-dialog	jquery
jQuery UI Button	jquery-ui-button	jquery
jQuery UI Effects	jquery-effects-core	jquery
jQuery UI Effects – Blind	jquery-effects-blind	jquery-effects-core
jQuery UI Effects – Bounce	jquery-effects-bounce	jquery-effects-core
jQuery UI Effects – Clip	jquery-effects-clip	jquery-effects-core
jQuery UI Effects – Drop	jquery-effects-drop	jquery-effects-core
jQuery UI Effects – Explode	jquery-effects-explode	jquery-effects-core

jQuery UI Effects – Fade	jquery-effects-fade	jquery-effects-core
jQuery UI Effects – Fold	jquery-effects-fold	jquery-effects-core
jQuery UI Effects – Highlight	jquery-effects-highlight	jquery-effects-core
jQuery UI Effects – Pulsate	jquery-effects-pulsate	jquery-effects-core
jQuery UI Effects – Scale	jquery-effects-scale	jquery-effects-core
jQuery UI Effects – Shake	jquery-effects-shake	jquery-effects-core
jQuery UI Effects – Slide	jquery-effects-slide	jquery-effects-core
jQuery UI Effects – Transfer	jquery-effects-transfer	jquery-effects-core
MediaElement.js (WP 3.6+)	wp-mediaelement	jquery
jQuery Schedule	schedule	jquery
jQuery Suggest	suggest	jquery
ThickBox	thickbox	
jQuery HoverIntent	hoverIntent	jquery
jQuery Hotkeys	jquery-hotkeys	jquery
Simple AJAX Code-Kit	sack	
QuickTags	quicktags	
Iris (Colour picker)	iris	jquery
Farbtastic (deprecated)	farbtastic	jquery
ColorPicker (deprecated)	colorpicker	jquery
Tiny MCE	tiny_mce	
Autosave	autosave	
WordPress AJAX Response	wp-ajax-response	
List Manipulation	wp-lists	
WP Common	common	
WP Editor	editorremov	

WP Editor Functions	editor-functions	
AJAX Cat	ajaxcat	
Admin Categories	admin-categories	
Admin Tags	admin-tags	
Admin custom fields	admin-custom-fields	
Password Strength Meter	password-strength-meter	
Admin Comments	admin-comments	
Admin Users	admin-users	
Admin Forms	admin-forms	
XFN	xfn	
Upload	upload	
PostBox	postbox	
Slug	slug	
Post	post	
Page	page	
Link	link	
Comment	comment	
Threaded Comments	comment-reply	
Admin Gallery	admin-gallery	
Media Upload	media-upload	
Admin widgets	admin-widgets	
Word Count	word-count	
Theme Preview	theme-preview	
JSON for JS	json2	
Plupload Core	plupload	
Plupload All Runtimes	plupload-all	
Plupload HTML4	plupload-html4	
Plupload HTML5	plupload-html5	
Plupload Flash	plupload-flash	
Plupload Silverlight	plupload-silverlight	
Underscore js	underscore	
Backbone js	backbone	

从核心移除

脚本名称	句柄	已删除版本	替换为
Scriptaculous Root	scriptaculous-root	WP 3.5	Google Version
Scriptaculous Builder	scriptaculous-builder	WP 3.5	Google Version
Scriptaculous Drag & Drop	scriptaculous-dragdrop	WP 3.5	Google Version
Scriptaculous Effects	scriptaculous-effects	WP 3.5	Google Version
Scriptaculous Slider	scriptaculous-slider	WP 3.5	Google Version
Scriptaculous Sound	scriptaculous-sound	WP 3.5	Google Version
Scriptaculous Controls	scriptaculous-controls	WP 3.5	Google Version
Scriptaculous	scriptaculous	WP 3.5	Google Version
Prototype Framework	prototype	WP 3.5	Google Version

名单还远远没有完成。您可以在wp-includes/script-loader.php中找到包含文件的完整列表。

# 条件标签

根据当前页面匹配的条件，可以在模板文件中使用条件标签来更改内容的显示。他们告诉WordPress在特定条件下显示什么代码。条件标签通常使用PHP if / else条件语句。

代码首先检查一个语句是真还是假。如果发现该语句为true，则执行第一组代码。如果它是假的，则跳过第一组代码，而代替执行第二组代码（在else之后）。

```
if ( is_user_logged_in() ):
    echo 'Welcome, registered user!';
else:
    echo 'Welcome, visitor!';
endif;
```

请注意这些标签与WordPress模板层次结构的密切关系。[被添加或删除？]

## 使用条件标签的位置

对于要修改数据的条件标签，信息必须已从数据库检索，即查询必须已经运行。如果在有数据之前使用条件标签，那么就没有什么可以询问if / else语句。

重要的是要注意，在运行查询之前，WordPress加载了functions.php，所以如果您只是在该文件中包含一个条件标签，它将无法正常工作。

两种实现条件标签的方式：

- 将其放在模板文件中
- 在functions.php中创建一个函数，该函数挂钩到稍后触发的action/filter

# 条件

以下列出了以下条件，其中以下每个条件语句证明是真实的。注意可以接受参数的标签。

## 主页

- is\_home()

当显示主博客页面时，此条件返回true，通常以标准的反向时间顺序排列。如果您的主页已经设置为静态页面，那么这只会在您设置为“设置>阅读”中设置为“帖子”页面的页面中证明是真实的。

## 首页

- is\_front\_page()

当站点的首页显示时，无论是否设置为显示帖子或静态页面，此条件都将返回true。

在以下情况下返回true：

正在显示主要博客页面

设置>阅读 - > 首页显示选项设置为您的最新帖子

要么

当设置>阅读 - > 首页显示设置为静态页面和

前页值是当前页面显示。

## 管理页

---

- `is_admin()`

当显示仪表板或管理面板时，此条件返回true。

## 单页面

---

- `is_single()`

当显示任何单个帖子（或附件或自定义帖子类型）时，返回true。如果您在页面上，此条件返回false。

- `is_single( ' 17' )`

`is_single()` 还可以通过ID和其他参数检查某些帖子。当Post 17显示为单个帖子时，上述示例证明是正确的。

- `is_single( 'Irish Stew' )`

参数也包括Post标题。在这种情况下，当标题为“爱尔兰炖肉”的帖子被显示为单个帖子时，这是真实的。

- `is_single( 'beef-stew' )`

当邮政邮局“牛炖肉”被显示为单一邮政时，证明是正确的。

- `is_single( array( 17, 'beef-stew' , 'Irish Stew' ) )`

当显示的单个帖子是帖子ID 17或post\_name是“牛肉炖”，或post\_title是“爱尔兰炖肉”时，返回true。

- `is_single( array( 17, 19, 1, 11 ) )`

当显示的单个帖子是帖子ID = 17，帖子ID = 19，帖子ID = 1或帖子ID = 11时，返回true。

- `is_single( array( 'beef-stew' , 'pea-soup' , 'chilli' ) )`

当显示的单个帖子是post\_name “beef-stew”，post\_name “pea-soup” 或post\_name “chilli” 时，返回true。



- `is_single( array( 'Beef Stew' , 'Pea Soup' , 'Chilli' ) )`

当显示的单个帖子是post\_title是 “Beef Stew” , post\_title是 “Pea Soup” 或post\_title是 “Chilli” 时, 返回true。

注意：此功能不区分帖子ID, 帖子标题或帖子名称。 如果请求17的职位ID, 将显示一个名为 “17” 的帖子。大概同样适用于带有 “the” 的帖子。

## 单个帖子, 页面或附件

- `is_singular()`

对于任何is\_single, is\_page和is\_attachment返回true。 它允许测试post类型。

## 置顶帖子

- `is_sticky()`

如果当前帖子中已经选中了 “置顶帖子” 复选框, 则返回true。 在此示例中, 没有发布帖子ID参数, 因此使用循环帖子的帖子ID。

- `is_sticky( ' 17' )`

Post 17被认为是置顶帖子的帖子时返回true。

## 一个帖子类型

- `get_post_type()`

您可以通过在您的条件中包含get\_post\_type()来测试当前帖子是否为某种类型。 它不是一个条件标签, 但它返回当前帖子的注册的帖子类型。

```
if ( 'book' == get_post_type() ) ...
```

- `post_type_exists()`

如果给定的帖子类型是注册的帖子类型, 则返回true。 这不会测试一个帖子是否是某个post\_type。

注意：此函数在3.0开发中替换了一个简称为is\_post\_type的函数。

## ##帖子类型是分层的

- `is_post_type_hierarchical( $post_type )`

如果\$post\_type在注册时已经设置了分层支持，则返回true。

- is\_post\_type\_hierarchical( 'book' )

如果book类型被注册为具有层次结构的支持，则返回true。

## 帖子类型存档

---

- is\_post\_type\_archive()

在任何帖子类型归档中返回true。

- is\_post\_type\_archive( \$post\_type )

如果在匹配\$post\_type的post类型归档页面（可以是单个帖子类型或Post类型数组），则返回true。

要打开帖子类型的档案，请在注册帖子类型时使用'has\_archive'=> true。

## 评论弹出窗口

---

- is\_comments\_popup()

当在评论弹出窗口。

## 任何页面包含帖子

---

- comments\_open()

当WordPress循环中正在处理当前帖子的注释时。

- pings\_open()

当允许在WordPress循环中处理当前帖子的ping时。

## “PAGE” 页面

---

本节涉及WordPress页面，不是您博客中的任何通用网页，也不是内置的post\_type “页面”。

- is\_page()

何时显示任何页面。

- is\_page( ' 42' )

当显示Page 42 ( ID ) 时。

- is\_page( 'About Me And Joe' )

当显示带有 “About Me And Joe” 的post\_title的页面时。

- is\_page( 'about-me' )

当显示带有 “about-me” 的post\_name ( slug ) 的页面时。

- is\_page( array( 42, 'about-me' , 'About Me And Joe' ) )

当显示的页面是帖子ID = 42或post\_name是 “about-me” 或post\_title是 “About Me And Joe” 时，返回true。

- is\_page( array( 42, 54, 6 ) )

当显示的页面是帖子ID = 42或帖子ID = 54或帖子ID = 6时，返回true。

## 分页的测试

您可以使用此代码来检查您是否位于使用该页面的页面中的第n页 `<!--nextpage-->`

QuickTag 这可能是有用的，例如，如果您希望仅在分为几页的帖子的第一页上显示元数据。

示例 1

```
<?php
    $paged = $wp_query->get( 'page' );
    if ( ! $paged || $paged < 2 ) {
        // This is not a paginated page (or it's simply the first page of a paginat
        ed page/post)
    } else {
        // This is a paginated page.
    } ?>
```

示例 2

```
<?php $paged = get_query_var( 'page' ) ? get_query_var( 'page' ) : false;
    if ( $paged == false ) {
        // This is not a paginated page (or it's simply the first page of a paginat
        ed page/post)
    } else {
        // This is a paginated page.
    }
?>
```

## 子页面测试

没有is\_subpage()函数，但可以用一点代码来测试：

## 片段 1

```
<?php global $post; // if outside the loop
if ( is_page() && $post->post_parent ) {
    // This is a subpage
} else {
    // This is not a subpage
}
?>
```

您可以使用Snippet 2中的代码创建自己的is\_subpage()函数。将其添加到functions.php文件中。它以与Snippet 1相同的方式测试父页面，但如果存在父页面，则返回父页面的ID，如果没有，则返回false。

## 片段 2

```
function is_subpage() {
    global $post; // load details about this page

    if ( is_page() && $post->post_parent ) { // test to see if the page has a
        parent
        return $post->post_parent; // return the ID of the parent p
        ost

    } else { // there is no parent so ...
        return false; // ... the answer to the questio
        n is false
    }
}
```

建议在Snippet 2中使用这样的功能，而不是像Snippet 1那样使用简单的测试，如果您打算经常测试子页面。要测试页面的父级是否是特定页面，例如“关于”（默认为页面ID为2），我们可以在Snippet 3中使用测试。这些测试检查我们是否正在查看有问题的页面，以及我们正在查看任何子页面。这对于设置特定于网站的不同部分的变量，以及不同的横幅图像或不同的标题很有用。

## 片段 3

```
<?php if ( is_page( 'about' ) || '2' == $post->post_parent ) {
    // the page is "About", or the parent of the page is "About"
    $bannerimg = 'about.jpg';
} elseif ( is_page( 'learning' ) || '56' == $post->post_parent ) {
    $bannerimg = 'teaching.jpg';
} elseif ( is_page( 'admissions' ) || '15' == $post->post_parent ) {
    $bannerimg = 'admissions.jpg';
} else {
    $bannerimg = 'home.jpg'; // just in case we are at an unclassified page, pe
    rhaps the home page
```

```
}
?>
```

Snippet 4是一个允许您更容易地进行上述测试的功能。如果我们查看有问题的页面（如“about”）或其一个子页面（因此，具有ID为“2”的父项的页面），此函数将返回true。

片段 4

```
function is_tree( $pid ) {           // $pid = The ID of the page we're looking for
    pages underneath
    global $post;                     // load details about this page

    if ( is_page($pid) )
        return true;                 // we're at the page or at a sub page

    $anc = get_post_ancestors( $post->ID );
    foreach ( $anc as $ancestor ) {
        if( is_page() && $ancestor == $pid ) {
            return true;
        }
    }

    return false; // we aren't at the page, and the page is not an ancestor
}
```

将Snippet 4添加到您的functions.php文件中，并调用is\_tree ( 'id' ) 来查看当前页面是页面还是页面的子页面。在代码段3中，is\_tree ( '2' ) 将替换 “is\_page ( 'about' ) || '2' == \$ post-> post\_parent ”里面的第一个if 标签。

请注意，如果您有多个级别的页面，则父页面是上一页，而不是层次结构顶层的页面。

## 是页面模板

允许您确定是否在页面模板中，或者是否正在使用特定的页面模板。

- is\_page\_template()

是否使用页面模板？

- is\_page\_template( 'about.php' )

是否使用页面模板？ 请注意，与其他条件不同，如果要指定特定的页面模板，则需要使用文件名，如about.php 或my\_page\_template.php。

注意：如果文件位于子目录中，那么您也必须包含此目录。这意味着这应该是与您的主题相关的文件路径以

及文件名，例如 “page-templates/about.php” 。

## 类别页面

---

- `is_category()`

当显示类别归档页面时。

- `is_category( '9' )`

当显示类别9的归档页面时。

- `is_category( 'Stinky Cheeses' )`

当显示名称为 “Stinky Cheeses” 的类别的存档页面时。

- `is_category( 'blue-cheese' )`

当显示具有类别的slug “blue-cheese” 类别的归档页面时。

- `is_category( array( 9, 'blue-cheese' , 'Stinky Cheeses' ) )`

当显示的帖子类别为term\_ID 9或者 “Stinky Cheeses” 或 “blue-cheese” 时，返回true。

- `in_category( '5' )`

如果当前帖子在指定的类别ID中，则返回true。

- `in_category( array( 1, 2, 3 ) )`

如果当前帖子在类别1,2或3中，则返回true。

- `! in_category( array( 4, 5, 6 ) )`

如果当前帖子不在类别4,5或6中，则返回true。注意！一开始

**注意：**测试时请务必检查拼写。 “is” 或 “in” 之间有很大的区别。

另请参见`is_archive()`和类别模板。

## 标签页

---

- `is_tag()`

当任何标签存档页面被显示时。

- `is_tag( 'mild' )`

当显示带有 “mild” 块的标签的存档页面时。

- `is_tag( array( 'sharp' , 'mild' , 'extreme' ) )`

正在显示的标签存档显示为 “sharp” , “mild” 或 “extreme” 时, 返回true。

- `has_tag()`

当前帖子有标签。 必须在循环中使用。

- `has_tag( 'mild' )`

当前帖子标签为 “mild” 时。

- `has_tag( array( 'sharp' , 'mild' , 'extreme' ) )`

当前帖子在数组中有任何标签时。

另请参见`is_archive()`和标签模板。

## 分类页

---

- `is_tax()`

当显示任何分类归档页面时。

- `is_tax( 'flavor' )`

当显示'flavor'分类的分类标准页面时。

- `is_tax( 'flavor' , 'mild' )`

当显示具有 “flavor” 和 “mild” 分类法的归档页面时。

- `is_tax( 'flavor' , array( 'sharp' , 'mild' , 'extreme' ) )`

当显示的'flavor'分类存档显示为 “sharp” , “mild或 “extreme” 时, 返回true。

- `has_term()`

检查当前帖子是否具有任何特定条款。 第一个参数应该是一个空字符串。 它预期分类标语/名称作为第二个参数。

- `has_term( 'green' , 'color' )`

当前帖子从分类 “color” 中有 “green” 一词。

- `has_term( array( 'green' , 'orange' , 'blue' ), 'color' )`

当前职位在阵列中有任何术语。

另请参见is\_archive ( )。

## 注册分类

---

- taxonomy\_exists()

当通过register\_taxonomy()注册一个特定的分类法时。 以前的is\_taxonomy()，在版本3.0中已被弃用

## 作者页

---

- is\_author()

当任何作者页面被显示时。

- is\_author( '4' )

当作者号 ( ID ) 4的存档页面正在显示时。

- is\_author( 'Vivian' )

当昵称为 “Vivian” 的作者的存档页面被显示时。

- is\_author( 'john-jones' )

当显示具有Nickname “john-jones” 的作者的归档页面时。

- is\_author( array( 4, 'john-jones' , 'Vivian' ) )

当作者的存档页面是用户ID 4或user\_nickname “john-jones” 或昵称 “Vivian” 时。

另请参见is\_archive()和作者模板。

## 多作者网站

---

- is\_multi\_author()

当多个作者发布了一个网站的帖子。 版本3.2可用。

## 日期页

---

- is\_date()

当显示任何基于日期的存档页面 ( 即每月，每年，每天或基于时间的存档 ) 时。

- is\_year()



当年度档案被显示时。

- `is_month()`

当显示每月存档时。

- `is_day()`

每日存档显示时。

- `is_time()`

正在显示一个小时，“精细”或“第二”档案。

- `is_new_day()`

如果今天是按照发布日期的新的一天。应该在循环中使用。

## 任何档案页

---

- `is_archive()`

当显示任何类型的存档页面时。类别，标签，作者和日期页面都是档案馆的所有类型。

## 搜索结果页

---

- `is_search()`

当显示搜索结果页面存档时。

## 404找不到页面

---

- `is_404()`

发生“HTTP 404：未找到”错误后显示页面。

## 一个附件

---

- `is_attachment()`

附件文件显示在帖子或页面时。附件是通过帖子编辑器的上传实用程序上传的图像或其他文件。附件可以显示在自己的“页面”或模板上。

## 单页，单张或附件

---

- `is_singular()`

当以下任何一个返回true时：is\_single ( ) , is\_page ( ) 或is\_attachment ( ) 。

- is\_singular( 'book' )

查看 “自定义帖子类型” 图书的帖子时为真。

- is\_singular( array( 'newspaper' , 'book' ) )

当查看自定义帖子类型的报纸或书籍的帖子时是真的。

## 一个联合

---

- is\_feed()

当网站要求是一个联合会。 该标签通常不被用户使用; 它由WordPress内部使用，可用于插件开发人员。

## 回溯

---

- is\_trackback()

当请求的站点是WordPress钩入其Trackback引擎。 该标签通常不被用户使用; 它由WordPress内部使用，可用于插件开发人员。

## 预览

---

- is\_preview()

当在草稿模式下查看显示的单个帖子。

## 有摘录

---

- has\_excerpt()

当前职位有摘录

- has\_excerpt( 42 )

当帖子42 ( ID ) 有摘录。

```
<?php // Get $post if you're inside a function global $post;
if ( empty( $post->post_excerpt ) ) {
    // This post has no excerpt
} else {
    // This post has excerpt
}
?>
```

## 其他用途

当你需要隐藏自动显示的摘录，只显示你的帖子的摘录。

```
<?php
    if ( ! has_excerpt() ) {
        echo '';
    } else {
        the_excerpt();
    }
?>
```

替换文本或代码的自动摘录。

```
<?php if ( ! has_excerpt() ) {
    // you text or code
} ?>
```

## 已分配导航菜单

- `has_nav_menu()`

注册的导航菜单位置是否已分配菜单

返回：`assign ( true )` 或 `not ( false )`

## 内循环

- `in_the_loop()`

检查你是否在“循环内”。对于插件作者很有用，当您在循环中时，此条件返回为真。

## 边栏活跃

- `is_active_sidebar()`

检查给定侧边栏是否处于活动状态（正在使用中）。如果边栏（由name，id或number标识）正在使用，则返回true，否则函数返回false。

## 网络的一部分（多站点）

- `is_multisite()`

检查当前站点是否在WordPress MultiSite安装中。

## 主站（多站点）

- `is_main_site()`

确定站点是否是网络中的主站点。

## 网络管理员（多站点）

---

- `is_super_admin()`

确定用户是否是网络（超级）管理员。

## 一个活动插件

---

- `is_plugin_active()`

检查插件是否被激活。

## 子主题

---

- `is_child_theme()`

检查子主题是否正在使用。

## 主题支持功能

---

- `current_theme_supports()`

检查是否存在各种主题功能。

# 工作实例

以下是演示如何使用这些条件标签的工作示例。

## 单张帖子

---

此示例显示如何使用`is_single()`仅在查看单个帖子时显示特定的内容：

```
if ( is_single() ) {  
  
    echo 'This is just one of many fabulous entries in the ' . single_cat_title()  
    . ' category!';  
  
}
```

循环中如何使用条件标签的另一个例子。选择在`index.php`中显示内容或摘录，这是显示单个帖子或主页时。

```

if ( is_home() || is_single() ) {

    the_content();

}
else {

    the_excerpt();

}

```

当您需要显示代码或元素时，不在主页的地方。

```

<?php if ( ! is_home() ) {

    //Insert your markup ...

}?>

```

## 检查多个条件

您可以使用PHP运算符在单个if语句中评估多个条件。

如果您需要检查条件的组合是否为true或false，那么这是方便的。

```

// Check to see if 2 conditionals are met
if ( is_single() || is_page() ) ) {
    // If it's a single post or a single page, do something special
}

if ( is_archive() && ! is_category( 'nachos' ) ) {
    // If it's an archive page for any category EXCEPT nachos, do something special
}

```

```

// Check to see if 3 conditionals are met
if ( $query->is_main_query() && is_post_type_archive( 'products' ) && ! is_admin() ) {

    // If it's the main query on a custom post type archive for Products
    // And if we're not in the WordPress admin, then do something special

}
if ( is_post_type_archive( 'movies' ) || is_tax( 'genre' ) || is_tax( 'actor' ) ) {
    // If it's a custom post type archive for Movies
}

```

```
// Or it's a taxonomy archive for Genre
// Or it's a taxonomy archive for Actor, do something special

}

<h3>Date Based Differences</h3>

If someone browses our site by date, let's distinguish posts in different years
by using different colors:

<?php // this starts The Loop
if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
<h2 id="post-<?php the_ID(); ?>">
<a title="Permanent Link to <?php the_title(); ?>" href="<?php the_permalink()
?>" rel="bookmark"><?php the_title(); ?></a></h2>

<small><?php the_time('F jS, Y') ?> by <?php the_author() ?></small>

<?php
// are we showing a date-based archive?
if ( is_date() ) {
    if ( date( 'Y' ) != get_the_date( 'Y' ) ) {
        // this post was written in a previous year
        // so let's style the content using the "oldentry" class
        echo '<div class="oldentry">';
    } else {
        echo '<div class="entry">';
    }
} else {
    echo '<div class="entry">';
}

the_content( 'Read the rest of this entry »' );
?></div>
```

## 可变侧栏内容

该示例将根据读者当前正在查看的页面在您的侧栏中显示不同的内容。

```
<div id="sidebar">
<?php // let's generate info appropriate to the page being displayed
if ( is_home() ) {
    // we're on the home page, so let's show a list of all top-level categories
    wp_list_categories( 'optionall=0&sort_column=name&list=1&children=0' );
} elseif ( is_category() ) {
    // we're looking at a single category view, so let's show _all_ the categories
    wp_list_categories( 'optionall=1&sort_column=name&list=1&children=1&hierarc
hical=1' )
}
```

```

} elseif ( is_single() ) {
    // we're looking at a single page, so let's not show anything in the sidebar
} elseif ( is_page() ) {
    // we're looking at a static page. Which one?
    if ( is_page( 'About' ) ) {
        // our about page.
        echo "This is my about page!";
    } elseif ( is_page( 'Colophon' ) ) {
        echo "This is my colophon page, running on WordPress " . bloginfo( 'version' )
        . " ";
    } else {
        // catch-all for other pages
        echo "Vote for Pedro!";
    }
} else {
    // catch-all for everything else (archives, searches, 404s, etc)
    echo "Pedro offers you his protection.";
} // That's all, folks!
?>
</div>

```

## 有用的404页

创建错误404页面文章[NEED Link to this?]有一个在“写入友好的消息”部分中使用PHP条件函数isset()的示例。

## 在主题的footer.php文件中

有时在其他模板（如sidebar.php）中执行的查询可能会损坏某些条件标签。例如，在header.php中，条件标签正常工作，但在主题的footer.php中不起作用。诀窍是在页脚中的条件测试之前放置wp\_reset\_query。例如：

```

<?php wp_reset_query();
if ( is_page( '2' ) ) {
    echo 'This is page 2!';
}
?>

```

## 条件标签索引

### 评论开放

- has\_tag
- has\_term
- in\_category
- is\_404

- is\_admin
- is\_archive
- is\_attachment
- is\_author
- is\_category
- is\_child\_theme
- is\_comments\_popup
- is\_date
- is\_day
- is\_feed
- is\_front\_page
- is\_home
- is\_month
- is\_multi\_author
- is\_multisite
- is\_main\_site
- is\_page
- is\_page\_template
- is\_paged
- is\_preview
- is\_rtl
- is\_search
- is\_single
- is\_singular
- is\_sticky
- is\_super\_admin
- is\_tag
- is\_tax
- is\_time
- is\_trackback
- is\_year
- pings\_open

## 功能参考

---

- Function: comments\_open()



- Function: is\_404()
- Function: is\_admin()
- Function: is\_admin\_bar\_showing()
- Function: is\_archive()
- Function: is\_attachment()
- Function: is\_author()
- Function: is\_category()
- Function: is\_comments\_popup()
- Function: is\_date()
- Function: is\_day()
- Function: is\_feed()
- Function: is\_front\_page()
- Function: is\_home()
- Function: is\_local\_attachment()
- Function: is\_main\_query
- Function: is\_multi\_author
- Function: is\_month()
- Function: is\_new\_day()
- Function: is\_page()
- Function: is\_page\_template()
- Function: is\_paged()
- Function: is\_plugin\_active()
- Function: is\_plugin\_active\_for\_network()
- Function: is\_plugin\_inactive()
- Function: is\_plugin\_page()
- Function: is\_post\_type\_archive()
- Function: is\_preview()
- Function: is\_search()
- Function: is\_single()
- Function: is\_singular()
- Function: is\_sticky()
- Function: is\_tag()
- Function: is\_tax()
- Function: is\_taxonomy\_hierarchical()
- Function: is\_time()

- Function: `is_trackback()`
- Function: `is_year()`
- Function: `in_category()`
- Function: `in_the_loop()`
- Function: `is_active_sidebar()`
- Function: `is_active_widget()`
- Function: `is_blog_installed()`
- Function: `is_rtl()`
- Function: `is_dynamic_sidebar()`
- Function: `is_user_logged_in()`
- Function: `has_excerpt()`
- Function: `has_post_thumbnail()`
- Function: `has_tag()`
- Function: `pings_open()`
- Function: `email_exists()`
- Function: `post_type_exists()`
- Function: `taxonomy_exists()`
- Function: `term_exists()`
- Function: `username_exists()`
- Function: `wp_attachment_is_image()`
- Function: `wp_script_is()`

# 类别，标签和自定义分类

类别，标签和分类法都是相关的，容易混淆。

我们将使用构建食谱网站主题的示例来帮助分类，标签和分类。

在我们的食谱网站，类别是早餐，午餐，晚餐，开胃菜，汤，沙拉，边和甜点。所有食谱都适合这些类别，但用户可能想搜索特定的东西，如巧克力甜点或姜鸡晚餐。

巧克力，姜和鸡都是标签的例子。它们是为用户提供意义的另一个特殊性。

最后有分类。实际上，类别和标签是默认分类法的例子，它们仅仅是组织内容的一种方法。分类法是在WordPress中分类内容和数据的方法。当您使用分类法时，您将类似的东西分组在一起。分类是指这些群体的总和。与Post Types一样，还有许多默认分类法，您也可以创建自己的分类。

食谱通常按类别和标签组织，但还有一些其他有用的方法来打破食谱，以便更加用户友好。例如，食谱网站可能希望通过烹饪时间显示食谱的简单方法。烹饪时间的定制分类为0-30分钟，30分钟至1小时，1至2小时，2+小时将是一个很大的故障。另外，烹饪方法如烤架，烤箱，炉子，冰箱等将成为与该网站相关的定制分类法的另一个例子。也可能有一个习惯分类法，如何辛辣的食谱，然后评级从1-5在spiciness。

## 默认分类

WordPress的默认分类是：

- categories：在帖子类型中组织内容的分级分类
- tags：在分级后类型中组织内容的非分层分类
- post formats：一种为您的帖子创建格式的方法。您可以在“[文章格式](#)”页面上了解更多信息。

## 条款

术语是您的分类学中的项目。所以，例如，如果你有动物分类法，你会有条件，狗，猫和羊。术语可以通过WordPress管理员创建，也可以使用wp\_insert\_term()函数。

## 数据库架构

分类和术语存储在以下数据库表中：

- wp\_terms - 存储所有的术语
- wp\_term\_taxonomy - 将该术语放在分类学中
- wp\_term\_relationships - 将分类法与对象（例如，要发布的类别）相关联

## 模板

WordPress为类别，标签或自定义分类提供了几个不同层次的模板。有关其结构和用法的更多详细信息，请参见“[分类模板](#)”页面。

## 自定义分类

---

可以在WordPress中创建新的分类法。 例如，您可能想在图书评论网站上创建作者分类，或者在电影网站上创建演员分类。 与自定义帖子类型一样，建议您将此功能放在插件中。 这样可以确保在用户更改网站设计时，其内容会保留在插件中。

您可以在“插件开发者手册”中阅读更多有关[创建自定义分类](#)的信息。

# 模板文件

我们之前在手册中介绍了模板文件。下一节将通过他们显示的帖子类型来分解不同的模板文件，并为这些模板文件的目的提供更深入的解释。您还可以了解不同模板文件的一些实际用例。

您将首先介绍处理显示帖子类型的模板文件。然后，您将看到显示页面类型的页面模板文件，并深入到WordPress内置的特定页面模板中。接下来，您将了解显示附件类型的附件模板文件。您还将了解如何在自定义帖子类型模板中显示由插件构建的自定义帖子类型。最后，您将触摸一些部分和各种模板文件，这些模板文件很重要，但不一定显示帖子类型。

**注意：**了解WordPress并不会真正将模板文件分解为特定类型的模板文件（如模板文件或附件模板文件）非常重要。特别是因为许多模板文件，如index.php或search.php可以显示许多不同的post类型。为了帮助您引用特定的模板文件，本手册根据它们可以显示的帖子类型对它们进行分类。

# 内容模板文件

WordPress使用多个模板文件来显示Post post类型。处理博客或其帖子的任何内容都属于Post Post类型。

## Index.php

如果没有其他模板文件，index.php将显示Post post类型。如许多地方所述，每个主题都必须有一个index.php文件才能生效。许多基本的主题可以通过使用index.php来显示他们的Post post类型，但上面给出的用例可以证明创建其他模板文件。

通常，您将需要独特的内容结构或布局，具体取决于显示的内容。您可以使用许多模板来根据站点内的上下文自定义内容结构。两个最引人注目的帖子模板文件是home.php和single.php，分别显示帖子和单个帖子。

## Home.php

当使用静态首页并且站点具有为博客列表定义的页面时，home.php文件用于指定的博客列表页面。鼓励使用此模板创建自定义页面模板，因为自定义页面模板上的博客分页将无法正常工作。如果在主题index.php中没有home.php将被替代使用。

## Single.php

在您的模板结构中尽可能简单地构建是不错的选择，除非您真正需要它们，否则不要制作更多模板。因此，大多数主题开发人员不会创建一个single-post.php文件，因为single.php足够具体。在大多数情况下，所有主题都应该有一个single.php。下面是一个来自主题Twenty Fifteen的single.php文件的例子。

```
<?php
/**
 * The template for displaying all single posts and attachments
 *
 * @package WordPress
 * @subpackage Twenty_Fifteen
 * @since Twenty Fifteen 1.0
 */

get_header(); ?>

<div id="primary" class="content-area">
    <main id="main" class="site-main" role="main">

        <?php
        // Start the loop.
        while ( have_posts() ) : the_post();

            /*
```

```

        * Include the post format-specific template for the content. If yo
u want to
        * use this in a child theme, then include a file called called con
tent-____.php
        * (where ____ is the post format) and that will be used instead.
        */
        get_template_part( 'content', get_post_format() );

        // If comments are open or we have at least one comment, load up th
e comment template.
        if ( comments_open() || get_comments_number() ) :
            comments_template();
        endif;

        // Previous/next post navigation.
        the_post_navigation( array(
            'next_text' => '<span class="meta-nav" aria-hidden="true">' . __
( 'Next', 'twentyfifteen' ) . '</span> ' .
                '<span class="screen-reader-text">' . __( 'Next post:', 'tw
entyfifteen' ) . '</span> ' .
                '<span class="post-title">%title</span>',
            'prev_text' => '<span class="meta-nav" aria-hidden="true">' . __
( 'Previous', 'twentyfifteen' ) . '</span> ' .
                '<span class="screen-reader-text">' . __( 'Previous post:',
'twentyfifteen' ) . '</span> ' .
                '<span class="post-title">%title</span>',
        ) );

        // End the loop.
    endwhile;
    ?>

    </main><!-- .site-main -->
</div><!-- .content-area -->

<?php get_footer(); ?>

```

在上面的代码示例中，您可以看到标题被拉入与get\_header ( ) 然后有两个html标签。接下来循环开始，模板标签get\_template\_part ( 'content', get\_post\_format ( ) ); 通过使用get\_post\_format ( ) 确定帖子类型来拉取适当的内容。接下来，使用模板标签comments\_template ( ) 引入注释。然后有一些分页。最后，内容div被关闭，然后用get\_footer ( ) 拉入页脚。

## Singular.php

WordPress版本4.3添加了singular.php，在层次结构中，single.php为帖子，page.php为页面，以及各自的变体。此模板遵循is\_singular ( ) 的规则，并且用于单个帖子，而不管后置类型。对于这两个文件使用相同代码

的主题（或另一个包含一个）现在可以简化为一个模板。

## Archive.php

除非开发人员在其模板中包含带有永久链接的元数据，否则archive.php将不会被使用。元数据是与帖子相关的信息。例如，发布的日期，作者以及用于帖子的任何类别，标签或分类法都是元数据的示例。当网站的访问者点击元数据时，archive.php将渲染与该元数据相关联的任何帖子。例如，如果访问者点击作者的名称，则archive.php将显示该作者的所有帖子。

通常，archive.php显示的页面的标题将是用户点击的元数据的名称。因此，如果用户点击作者的名字，显示所有其他作者的帖子的页面名称将是作者的名称，并且经常可能会有关于元数据的其他说明。这是一个代码示例，来自Twenty Fifteen，他们的archive.php文件。这个代码片段是使archive.php文件与home.php或index.php文件不同的唯一代码片段。

```
<header class="page-header">
    <?php
        the_archive_title( '

<h1 class="page-title">', ' </h1>

' );
        the_archive_description( '

<div class="taxonomy-description">', ' </div>

' );
    ?>
</header>

<!-- .page-header -->
```

## Author.php 和 Date.php

Author.php和date.php是更具体的存档类型文件。如果您需要一个复习，请查看他们在模板范围内适合的位置。通常，archive.php将满足大多数主题的需求，您不需要创建这些模板。

## Author.php

如果你正在为多个作者设计一个主题，那么构建一个author.php模板可能是有意义的。在author.php模板中，您可以提供有关作者，他们的图形，拉入他们的社交媒体网站以及他们写的所有帖子的更多信息。这将是依赖于archive.php文件的一个步骤。



此外，您可以使用其作者ID或nickname为个别作者构建特定的author.php文件。 例如，说John Doe是有许多客座作者的网站的头号作者。 您可能希望所有访客作者的信息与author.php一起显示，但如果作者ID为3，则可以通过创建author-johndoe.php或author-3.php来构建具有John Doe更多信息的特定作者页面。

## Date.php

同样，如果您正在构建一个专门针对杂志或新闻网站的主题，那么date.php文件可能会有所修改，因为这些网站经常按日期或问题组织文章和帖子。 另外，如果你找到了足够的理由，你可以建立一个day.php，month.php或者year.php。

## Category.php, Tag.php 和 Taxonomy.php

如果您需要重新分析哪些类别，标签和分类法，您可以查看他们的页面。 通常，您不需要构建这些模板文件。 然而，在为食品博客撰写主题的例子中，有一些用例来构建这些特定的模板。 在食品博客网站上，类别可能是美食，美食，民族美食和食谱。

您可能希望大部分的博文都以相同的方式显示，除了任何分类为食谱的博客，因为所有食谱都有成分和分析部分。 因此，您可能需要构建一个category-recipe.php文件，以在网格视图中显示您的食谱博客文章，其中包含有关配方可见的一些重要详细信息。

此外，也许巧克力是您正在建立的主题的一个非常重要的标签。 构建tag-chocolate.php文件可能是有意义的，以便您可以显示巧克力的专门横幅图像。

## Search.php

大多数主题都有一个search.php文件，所以用户清楚他们的查询是通过的。 通常有一些标题查询结果的标题，例如这个二十五分之二的主题的代码段。

```
<header class="page-header">
<h1 class="page-title"><?php printf( __( 'Search Results for: %s', 'twentyfifteen' ), get_search_query() ); ?></h1>
</header>
<!-- .page-header -->
```

此代码段将拉入使用get\_search\_query( ) 搜索的查询。 通常，search.php只会提取摘录而不是完整内容，因为用户尝试确定文章或页面是否适合其搜索。

# 页面模板文件

页面模板是可以应用于特定页面或一组页面的特定类型的模板文件。

**注意：**自WordPress 4.7页面模板支持所有帖子类型。有关如何将页面模板设置为特定帖子类型的详细信息，请参阅下面的示例。

由于页面模板是特定类型的模板文件，因此页面模板的一些区别特征如下：

- 页面模板用于更改页面的外观和感觉。
- 页面模板可以应用于单个页面，页面部分或一类页面。
- 页面模板通常具有高度的特异性，针对单个页面或一组页面。例如，名为page-about.php的页面模板比模板文件page.php或index.php更具体，因为它只会影响具有“about”的页面。
- 如果页面模板具有模板名称，则编辑页面的WordPress用户可以控制将用于渲染页面的模板。

## 用户页面模板

页面模板在网页上显示您网站的动态内容，例如帖子，新闻更新，日历活动，媒体文件等。您可以决定您希望您的主页以特定的方式查看，这与您网站的其他部分完全不同。或者，您可能希望显示链接到页面一部分上的帖子的精选图片，还有其他地方的最新帖子列表，并使用自定义导航。您可以使用页面模板来实现这些功能。本节介绍如何构建可由用户通过其管理屏幕选择的页面模板。

例如，您可以构建页面模板：

- full-width, 一列
- two-column 右边是一个侧边栏
- two-column 左侧有一个侧边栏
- three-column 三列

## 模板层次结构中的页面模板

当某人浏览您的网站时，WordPress将选择用于呈现该页面的模板。正如我们之前在模板层次结构中学到的，WordPress按以下顺序查找模板文件：

- 页面模板 - 如果页面具有分配的自定义模板，则WordPress会查找该文件，如果找到，则使用它。
- page- {slug} .php - 如果没有分配自定义模板，WordPress将查找并使用包含该页面的块的专门模板。
- page- {id} .php - 如果没有找到包含页面小插件的专门模板，WordPress会查找并使用以该页面的ID命名的专用模板。
- page.php - 如果没有找到包含页面ID的专用模板，WordPress会查找并使用主题的默认页面模板。
- singular.php - 如果没有找到page.php，WordPress将查找并使用用于单个帖子的主题模板，不考虑帖子类

型。

- index.php - 如果没有指定或找到特定的页面模板，WordPress将默认使用主题的索引文件来呈现页面。

**警告：**还有一个名为paged.php的WordPress定义的模板。它不用于页面类型，而是用于显示多个存档页面。

## 页面模板用途和用户控制

如果您计划为主题制作自定义页面模板，则应在继续操作之前决定一些事情：

页面模板是否用于特定页面或任何页面；和

您想要为模板提供哪种类型的用户控件。

创建或编辑页面时，用户可以选择具有模板名称的每个页面模板。可以在“页面” > “添加新建” > “属性” > “模板” 中找到可用模板列表。因此，WordPress用户可以选择任何具有模板名称的页面模板，这可能不是您的意图。

例如，如果您想要为“关于”页面设置一个特定的模板，可能不适合将该页面模板命名为“关于模板”，因为它可以在全局范围内对所有页面（即用户可以将其应用于任何页）。相反，只要用户访问“关于”页面，创建单一使用模板，WordPress将使用适当的模板呈现页面。

相反，许多主题都包括选择页面将有多少列的能力。这些选项中的每一个都是全球可用的页面模板。为了让您的WordPress用户使用此全局选项，您将需要为每个选项创建页面模板，并为每个选项提供一个模板名称。判断一个模板是全局使用还是单一使用是通过文件的命名方式来实现的，以及是否有特定的注释。

**注意：**有时候，模板全局可用，即使它似乎是一个单一的用例也是合适的。当您创建发布主题时，可能很难预测用户对其页面的名称。投资组合页面是一个很好的例子，因为并不是每个WordPress用户都将他们的投资组合命名为相同的东西或具有相同的页面ID，但他们可能想要使用该模板。

## 页面模板的文件组织

如主题文件组织中所述WordPress识别子文件夹页面模板。因此，将全局页面模板存储在此文件夹中是一个好主意，以帮助保持组织。

**警报：**专门的页面模板文件（仅一次使用的文件）不能在子文件夹中，也不能在父主题文件夹中使用子主题。

## 创建全局使用的自定义页面模板

有时你会想要一个可以在任何页面或多个页面全局使用的模板。一些开发人员将使用文件名前缀对其模板进行分组，例如page\_two-columns.php

**警示：重要！** 不要使用page-作为前缀，因为WordPress会将文件解释为专门的模板，仅适用于您网站上的

## 一个页面。

有关不能使用的主题文件命名约定和文件名的信息，请参阅保留的[主题文件名](#)。

提示：创建新页面模板的快速，安全的方法是创建一个page.php的副本，并为新文件提供不同的文件名。这样，您可以从其他页面的HTML结构开始，您可以根据需要编辑新文件。

要创建一个全局模板，请在文件顶部写入一个打开的PHP注释，该注释指出模板的名称。

```
<?php /* Template Name: Example Template */ ?>
```

选择一个名称来描述模板在编辑页面时WordPress用户可以看到名称。例如，您可以命名您的模板主页，博客或投资组合。

来自TwentyFourteen主题的此示例创建一个称为“全宽页面”的页面模板：

```
<?php
/**
 * Template Name: Full Width Page
 *
 * @package WordPress
 * @subpackage Twenty_Fourteen
 * @since Twenty_Fourteen 1.0
 */
```

basics-page-templates-03 当您上传文件到主题文件夹（例如页面模板）时，请转到管理信息中心的“页面”>“编辑”屏幕。

在右侧的属性下，您将看到模板。用户可以访问全局页面模板。

提示：选择列表的最大宽度为250px，因此更长的名称可能会被切断。

## 为一个特定页面创建自定义页面模板

如“模板层次”页面中所述，您可以为特定页面创建模板。要创建一个特定页面的模板，请复制您现有的page.php文件，并用您的页面的插槽或ID重命名：

- page-{slug}.php
- page-{ID}.php

例如：您的“关于”页面有“about”和ID为6。如果您的活动主题的文件夹有一个名为page-about.php或page-6.php的文件，则WordPress将自动查找并使用该文件呈现关于页面。

要使用，专业页面模板必须位于主题的文件夹（即/wp-content/themes/my-theme-name/）中。

## 为特定的帖子类型创建页面模板

默认情况下，自定义页面模板将可用于“页面”帖子类型。

要为特定的帖子类型创建页面模板，请在模板名称下面添加一个需要模板支持的帖子类型的一行。

示例:

```
<?php
/*
Template Name: Full-width layout
Template Post Type: post, page, event
*/
// Page code here...
```

**警报：**只有从WordPress 4.7才支持将页面模板添加到“page”以外的帖子类型的功能

当一个帖子类型至少存在一个模板时，“Post Attributes”元框将显示在后端，而不需要为“page-attributes”或其他任何东西添加post类型的支持。“Post Attributes”标签可以在注册一个帖子类型时使用“attributes”标签定制每个帖子类型。

向后兼容性：

假设您想公开发布支持帖子类型模板的主题。4.7之前的WordPress版本将忽略模板帖子类型标题，并在页面模板列表中显示模板，即使它仅适用于常规帖子。为了防止这种情况，您可以挂接到theme\_page\_templates过滤器中以将其从列表中排除。以下是一个例子：

```
/**
 * Hides the custom post template for pages on WordPress 4.6 and older
 *
 * @param array $post_templates Array of page templates. Keys are filenames, values are translated names.
 * @return array Filtered array of page templates.
 */
function makewp_exclude_page_templates( $post_templates ) {
    if ( version_compare( $GLOBALS['wp_version'], '4.7', '<' ) ) {
        unset( $post_templates['templates/my-full-width-post-template.php'] );
    }

    return $post_templates;
}

add_filter( 'theme_page_templates', 'makewp_exclude_page_templates' );
```

这样，您可以在WordPress 4.7及更高版本中支持自定义帖子类型模板，同时保持完全向后兼容性。

请注意，`theme_page_templates`实际上是一个动态主题`{ $ post_type }_templates`过滤器。挂钩名称的动态部分`$post_type`是指模板支持的帖子类型。例如。您可以钩入`theme_product_templates`来过滤产品职位类型的模板列表。

## 在页面模板中使用条件标签

您可以使用主题的`page.php`文件中的条件标签来更小的页面特定更改。例如，下面的示例代码将加载您的首页的文件`header-home.php`，但为“关于”页面加载另一个文件（`header-about.php`），然后对所有其他页面应用默认的`header.php`。

```
if ( is_front_page() ) :
    get_header( 'home' );
elseif ( is_page( 'About' ) ) :
    get_header( 'about' );
else:
    get_header();
endif;
```

您可以在[此处](#)了解有关[条件标签](#)的更多信息。

## 识别页面模板

如果您的模板使用了`body_class()`函数，则WordPress将在文本类标题（页面），页面ID（`page-id- {ID}`）以及所使用的页面模板的`body`标签中打印类。对于默认的`page.php`，生成的类名称为`page-template-default`：

```
<body class="page page-id-6 page-template-default">
```

注意：一个专门的模板（`page- {slug} .php`或`page- {ID} .php`）也获取了`page-template-default`类而不是其自己的`body`类。

当使用自定义页面模板时，将打印类页面模板以及命名特定模板的类。例如，如果您的自定义页面模板文件命名如下：

```
<?php
/* Template Name: My Custom Page */
?>
```

然后渲染HTML生成将如下所示：

```
<body class="page page-id-6 page-template page-template-my-custom-page-php">
```

注意应用于body标签的page-template-my-custom-page-php类。

## 页面模板方法

---

这些内置的WordPress功能和方法可以帮助您使用页面模板：

- `get_page_template()` 返回用于呈现页面的页面模板的路径。
- `wp_get_theme()->get_page_templates()` 返回可用于当前活动主题的所有自定义页面模板（`get_page_templates()` 是WP\_Theme类的一种方法）。
- `is_page_template()` 根据是否使用自定义页面模板来呈现页面，返回true或false。
- `get_page_template_slug()` 返回自定义字段\_wp\_page\_template的值（当该值为空时空或“默认值”）  
如果页面已分配了一个自定义模板，该模板的文件名将被存储为名为 “\_wp\_page\_template” 的自定义字段的值 wp\_postmeta数据库表）。（自定义字段以下划线开头，不会显示在编辑屏幕的自定义字段模块中。）

# 附件模板文件

附件是一种特殊的帖子类型，其中包含通过WordPress媒体上传系统上传的文件的信息，例如其描述和名称，可以通过多个帖子类型 - 附件模板文件显示。

对于图像，作为示例，附件发布类型链接到元数据信息，关于图像的大小，生成的缩略图，图像文件的位置，HTML替代文本以及从嵌入在图像中的EXIF数据获得的甚至信息。

**提示：**利用附件模板获取上传的其他元数据信息，帮助SEO工作。

如模板层次结构所示，您可以通过多个模板文件按照后退顺序显示附件：

- MIME\_type.php and a subtype.php 它可以是任何MIME类型（例如：image.php，video.php，application.php）。对于text/plain，使用以下路径（顺序）：
  - text\_plain.php
  - plain.php
  - text.php
- attachment.php
- single-attachment.php
- single.php
- singular.php
- index.php

## MIME\_type.php

附件由模板文件根据其MIME类型提供。例如，如果您的附件是图像，您可以通过创建一个image.php模板文件来自定义他们的显示方式。具有image\_\*的post\_mime\_type的所有图像将通过您的image.php模板文件呈现。附件还支持使用mime subtype.php文件。要继续使用图像示例，您可以进一步自定义主题，不仅支持image.php文件，还支持jpg.php子类型文件。

## Attachment.php

附件页面（attachment.php）是具有附件类型的单个帖子页面，通过创建一个attachment.php生成。就像一个单独的页面，专门针对您的文章，附件页面为您的主题中的附件提供了专门的页面。

创建附件页面就像创建一个attachment.php文件一样简单。attach.php文件包含与single.php post页面相似的代码。

```
<div class="entry-attachment">
    <?php $image_size = apply_filters( 'wporg_attachment_size', 'large' );
        echo wp_get_attachment_image( get_the_ID(), $image_size ); ?>
```



```
<?php if ( has_excerpt() ) : ?>

<div class="entry-caption">
    <?php the_excerpt(); ?>
</div><!-- .entry-caption -->
<?php endif; ?>
</div><!-- .entry-attachment -->
```

## 参考方法

---

- `get_attachment_template()` : 检索当前或父模板中的附件模板路径。

# 自定义内容类型

WordPress主题系统支持自定义帖子类型的自定义模板。支持属于自定义帖子类型的帖子的单个显示的自定义模板已被支持，因为WordPress 3.0版和对归档显示的自定义模板的支持已在版本3.1中添加。

## 自定义帖子类型 - 模板层次结构

WordPress将通过模板层次结构，并使用它首先遇到的模板文件。所以如果要为您的acme\_product自定义帖子类型创建一个自定义模板，一个好的开始是复制single.php文件，将其保存为single-acme\_product.php并进行编辑。

但是，如果您不想创建自定义模板文件，WordPress将使用您的主题中已存在的文件，这将是archive.php和single.php和index.php文件。

可以使用single.php和archive.php模板文件分别显示单个帖子及其归档，自定义帖子类型的单个帖子将使用single-{post\_type}.php他们的档案将使用archive-{post\_type}.php如果您没有这个帖子类型的存档页面，您可以传递BLOG\_URL ? post\_type = {post\_type}其中{post\_type}是register\_post\_type()函数的\$ post\_type参数。因此，对于上述示例，您可以为单个产品帖子及其归档创建单个acme\_product.php和archive-acme\_product.php模板文件。

或者，您可以在任何模板文件中使用is\_post\_type\_archive()函数来检查查询是否显示给定帖子类型的归档页面以及post\_type\_archive\_title()以显示帖子类型标题。

## 自定义帖子类型模板

- single- {post-type} .php 访问者从自定义帖子类型请求单个帖子时使用的单个帖子模板。例如，single-acme\_product.php将用于显示名为acme\_product的自定义帖子类型中的单个帖子。
- archive- {post-type} .php 当访问者请求自定义帖子类型归档时，将使用归档文件类型模板。例如，archive-acme\_product.php将用于显示名为acme\_product的自定义帖子类型的帖子存档。如果archive-{post-type} .php不存在，则使用archive.php模板文件。
- search.php 搜索结果模板用于显示访问者的搜索结果。要包含您的自定义帖子类型的搜索结果，请参阅下面的代码示例。
- index.php 如果定制帖子类型的特定查询模板 ( single-{post-type} .php , single.php , archive- {post-type} .php , archive.php , search.php ) 为 不存在。

## 搜索自定义帖子类型

默认情况下，搜索结果不包括自定义帖子类型内容到搜索结果中。代码可以附加到functions.php以在搜索结果中包含CPT：

```
// Show posts of 'post', 'page', 'acme_product' and 'movie' post types on home
```

```
page
function search_filter( $query ) {
    if ( !is_admin() && $query->is_main_query() ) {
        if ( $query->is_search ) {
            $query->set( 'post_type', array( 'post', 'page', 'acme_product', 'movie' )
        );
        }
    }
}

add_action( 'pre_get_posts', 'search_filter' );
```

在上面的代码中，函数search\_filter挂钩在pre\_get\_posts上。此功能将限制搜索结果显示帖子，页面和自定义帖子类型movie和acme\_product。

## 参考方法

- register\_post\_type()：注册一个帖子类型。
- is\_post\_type\_archive()：检查是否查询现有的帖子类型归档页面。
- post\_type\_archive\_title()：显示或检索帖子类型归档的标题。

# 部分和其他模板文件

并不是所有的模板文件都会生成将由浏览器呈现的整个内容。一些模板文件被其他模板文件拉入，如 `comments.php`，`header.php`，`footer.php`，`sidebar.php`和`content-{$ slug}.php`。您将通过这些模板文件中的每一个来了解目的以及如何构建它们。

## Header.php

`header.php`文件完全符合您的期望。它包含浏览器将为标题呈现的所有代码。这是一个部分模板文件，因为除非不同的模板文件调用模板标签`get_header()`，浏览器将不会呈现此文件的内容。

通常网站具有相同的标题，无论您所在的页面或帖子如何。但是，根据页面的不同，一些网站会有轻微的变化，例如辅助导航或不同的横幅图像。如果您使用条件标签，您的`header.php`文件可以处理所有这些变体。

几乎所有的主题都有一个`header.php`文件，因为这个文件的功能和可维护性几乎要求它的创建。

下面是一个在 `twenty fifteen`主题中找到的`header.php`的例子。

```
<!DOCTYPE html>
<html <?php language_attributes(); ?> class="no-js">
<head>
    <meta charset="<?php bloginfo( 'charset' ); ?>">
    <meta name="viewport" content="width=device-width">
    <link rel="profile" href="http://gmpg.org/xfn/11">
    <link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">
    <!--[if lt IE 9]>
    <script src="<?php echo esc_url( get_template_directory_uri() ); ?>/js/html
5.js"></script>
    <![endif]-->
    <?php wp_head(); ?>
</head>

<body <?php body_class(); ?>>

<div id="page" class="hfeed site">
    <a class="skip-link screen-reader-text" href="#content"><?php _e( 'Skip to
content', 'twentyfifteen' ); ?></a>

<div id="sidebar" class="sidebar">

<header id="masthead" class="site-header" role="banner">
```

```

<div class="site-branding">
    <?php if ( is_front_page() && is_home() ) : ?>

<h1 class="site-title"><a href="<?php echo esc_url( home_url( '/' ) ); ?>" rel=
"home"><?php bloginfo( 'name' ); ?></a></h1>


    <?php else : ?>

<a href="<?php echo esc_url( home_url( '/' ) ); ?>" rel="home"><?php bloginfo(
'name' ); ?></a>

        <?php endif; $description = get_bloginfo( 'description', 'd
isplay' ); if ( $description || is_customize_preview() ) : ?>

<?php echo $description; ?>

        <?php endif; ?>
        <button class="secondary-toggle"><?php _e( 'Menu and widgets',
'twentyfifteen' ); ?></button>
    </div>

<!-- .site-branding -->
    </header>

<!-- .site-header -->

    <?php get_sidebar(); ?>
</div>

<!-- .sidebar -->

<div id="content" class="site-content">

```

一些代码首先可能看起来有点令人生畏，但如果我们把它打破了，那就变得简单了。开幕后，头部被创建。模板标签wp\_head()拉入我们所有的样式和任何出现在头部而不是我们在我们的functions.php文件中排队的页脚脚本。

接下来，身体被打开，HTML和PHP的混合存在。您可以在网站品牌div中看到一些基于用户所在页面显示的条件标签。然后网站导航被拉入。最后，主站点内容div被打开，最接近footer.php文件。

要注意的一个重要的模板标签是在开头body标签中找到的body\_class()。这是一个超级方便的标签，通过根据模板文件和其他正在使用的设置给予您的身体类别，使您的主题更容易设计。

## Footer.php

很像header.php文件，footer.php是一个非常常见的模板文件，大多数主题都使用。footer.php文件中的代码将不会被渲染，除非另外一个模板文件使用get\_footer()模板标签来拉入footer.php。与标题类似，您可以使用条件标签来创建页脚的变体。

开发人员通常会将脚本区放在页脚中，以便最终用户可以轻松地将不同的内容类型拖放到页脚中。

以下是Twenty Fifteen主题的footer.php文件的示例。

```
</div>

<!-- .site-content -->

<footer id="colophon" class="site-footer" role="contentinfo">

<div class="site-info">
    <?php /** * Fires before the Twenty Fifteen footer text for footer
customization. * * @since Twenty Fifteen 1.0 */ do_action( 'twentyfifteen_credi
ts' ); ?>
    <a href="<?php echo esc_url( __( 'https://wordpress.org/', 'twentyf
ifteen' ) ); ?>"><?php printf( __( 'Proudly powered by %s', 'twentyfifteen' ),
'WordPress' ); ?></a>
</div>

<!-- .site-info -->
</footer>

<!-- .site-footer -->

</div>

<!-- .site -->

<?php wp_footer(); ?>

</body>
</html>
```

## 404.php

当用户尝试访问您网站上不存在的页面时，会将其导向到index.php页面，除非您已经创建了一个404.php模板。这是一个好主意，有一些消息，说明页面丢失或不再可用。创建此模板有助于保持主题的视觉方面一致，并向最终用户提供有用的信息。

这是 twenty fifteen主题的一个404.php模板文件的例子。

```
get_header(); ?>

<div id="primary" class="content-area">
    <main id="main" class="site-main" role="main">

<section class="error-404 not-found">

<header class="page-header">

<h1 class="page-title"><?php _e( 'Oops! That page can&rsquo;t be found.', 'twentyfifteen' ); ?></h1>

    </header>

<!-- .page-header -->

<div class="page-content">

<?php _e( 'It looks like nothing was found at this location. Maybe try a search
?', 'twentyfifteen' ); ?>

        <?php get_search_form(); ?>
    </div>

<!-- .page-content -->
</section>
```

```
<!-- .error-404 -->

    </main><!-- .site-main -->
</div>

<!-- .content-area -->

<?php get_footer(); ?>
```

## Comments.php

comments.php文件处理你所期待的，评论。这是一个部分模板，被拉入其他模板文件以显示用户在页面或帖子上留下的注释。几个不同的页面和帖子显示注释，所以有必要在需要时将一个文件拉入。

在注释模板页面上扩展了创建注释中涉及的代码。

## Sidebar.php

很多主题都使用侧边栏来显示小部件。对于侧边栏工作在主题中，必须注册，然后必须创建侧栏的模板文件。

您将在后面的章节中了解有关注册侧边栏的更多信息。边栏文件通常具有条件语句和其中的

is\_active\_sidebar ( 'sidebar-name' ) 功能，以确保侧边栏中正在使用窗口小部件，以使空HTML不会不必要地添加到页面中。

这是twenty fifteen的侧边栏模板文件的例子。注意底部的侧边栏是使用< ? php dynamic\_sidebar ( 'sidebar-1' ) 拉入的; >。现在无论如何，插入该边栏的小部件将显示在正在使用此模板的页面上。

```
if ( has_nav_menu( 'primary' ) || has_nav_menu( 'social' ) || is_active_sidebar(
    'sidebar-1' ) ) : >

<div id="secondary" class="secondary">

    <?php if ( is_active_sidebar( 'sidebar-1' ) ) : >

<div id="widget-area" class="widget-area" role="complementary">
    <?php dynamic_sidebar( 'sidebar-1' ); >
</div>

    <!-- .widget-area -->
    <?php endif; >
```



```

</div>

<!-- .secondary -->

<?php endif; >

```

## Content-{\$slug}.php

许多主题开发者将他们的模板文件分解成小尺寸的小块。他们经常将包装器和页面架构元素放在诸如page.php，home.php，comments.php等模板文件中，然后将代码显示在另一个模板文件中。输入内容 - {\$ slug} .php：常见的例子是content-page.php，content-post.php，content-portfolio.php，content-none.php。这些不是WordPress识别并将以某种方式解释的文件名，而是显示特定类型内容的常见方法。

例如，通常在博客帖子上，您想要显示作者的姓名，帖子的日期以及可能的帖子类别。你也可能有链接到上一个和下一个帖子。该信息在常规页面上是不合适的。类似地，在投资组合页面上，您可能会有一个特色图片或图库，您可能会以不同于说博客或页面的精选图片的方式显示。

您将要使用get\_template\_part（）模板标签来拉入内容 - {\$ slug} .php文件。要拉入你的content-page.php文件，你可以调用get\_template\_part（'content'，'page'）；

这是一个content-page.php模板文件的twenty fifteen的例子。

```

<article id="post-<?php the_ID(); ?>" <?php post_class(); ?>
    <?php // Post thumbnail. twentyfifteen_post_thumbnail(); ?>

    <header class="entry-header">
        <?php the_title( '

    <h1 class="entry-title">', '</h1>

    ' ); ?>
    </header>

    <!-- .entry-header -->

    <div class="entry-content">
        <?php the_content(); ?>
        <?php wp_link_pages( array( 'before' => '

```

```

<div class="page-links"><span class="page-links-title">' . __( 'Pages:', 'twentyfifteen' ) . '</span>',
    'after' => '</div>'

    'link_before' => '<span>',
    'link_after' => '</span>',
    'pagelink' => '<span class="screen-reader-text">' . __( 'Page', 'twentyfifteen' ) . ' </span>%',
    'separator' => '<span class="screen-reader-text">, </span>',
    ) );
    ?>
</div>

<!-- .entry-content -->

<?php edit_post_link( __( 'Edit', 'twentyfifteen' ), '

<footer class="entry-footer"><span class="edit-link">', '</span></footer>

<!-- .entry-footer -->' ); ?>

</article>

<!-- #post-## -->

```

# 评论模板

WordPress会根据WordPress主题中的comments.php文件中的设置和代码在主题中显示注释。

## 简单的评论循环

```
<?php
//Get only the approved comments
$args = array(
    'status' => 'approve'
);

// The comment Query
$comments_query = new WP_Comment_Query;
$comments = $comments_query->query( $args );

// Comment Loop
if ( $comments ) {
    foreach ( $comments as $comment ) {
        echo '<p>' . $comment->comment_content . '</p>';
    }
} else {
    echo 'No comments found.';
}
?>
```

comments.php模板包含将注释从数据库中拉出并显示在主题中的所有逻辑。

在我们探索模板文件之前，您需要知道如何在相应的页面（如single.php）上拉入部分模板文件。您将包含注释模板标签在条件语句中，所以comments.php只有在有意义的情况下被拉入。

```
// 如果发表评论，或者至少有一个评论，请加载评论模板。
if ( comments_open() || get_comments_number() ) :
    comments_template();
endif;
```

## 另一个Comments.php示例

以下是Twenty Thirteen主题中包含的comments.php模板的示例：

```
<?php
/**
 * 用于显示评论的模板。
 *
 * 包含评论和评论表单的页面区域。
 */
```

```

*
* @package WordPress
* @subpackage Twenty_Thirteen
* @since Twenty Thirteen 1.0
*/

/*
 * If the current post is protected by a password and the visitor has not yet
 * entered the password we will return early without loading the comments.
 */
if ( post_password_required() )
    return;
?>

<div id="comments" class="comments-area">

    <?php if ( have_comments() ) : ?>
        <h2 class="comments-title">
            <?php
                printf( _nx( 'One thought on "%2$s"', '%1$s thoughts on "%2$s"',
get_comments_number(), 'comments title', 'twentythirteen' ),
                    number_format_i18n( get_comments_number() ), '<span>' . get
_the_title() . '</span>' );
            ?>
        </h2>

        <ol class="comment-list">
            <?php
                wp_list_comments( array(
                    'style'      => 'ol',
                    'short_ping' => true,
                    'avatar_size' => 74,
                ) );
            ?>
        </ol><!-- .comment-list -->

        <?php
            // Are there comments to navigate through?
            if ( get_comment_pages_count() > 1 && get_option( 'page_comments' )
) :
                ?>
                <nav class="navigation comment-navigation" role="navigation">
                    <h1 class="screen-reader-text section-heading"><?php _e( 'Comment n
avigation', 'twentythirteen' ); ?></h1>
                    <div class="nav-previous"><?php previous_comments_link( __( '&larr;
Older Comments', 'twentythirteen' ) ); ?></div>
                    <div class="nav-next"><?php next_comments_link( __( 'Newer Comments
&rarr;', 'twentythirteen' ) ); ?></div>
                </nav><!-- .comment-navigation -->
                <?php endif; // Check for comment navigation ?>

```

```

        <?php if ( ! comments_open() && get_comments_number() ) : ?>
        <p class="no-comments"><?php _e( 'Comments are closed.' , 'twentythirte
en' ); ?></p>
        <?php endif; ?>

    <?php endif; // have_comments() ?>

    <?php comment_form(); ?>

</div><!-- #comments -->

```

## 打破comments.php

以上的comment.php可以分解到下面的部分，以便更好的理解。

- 模板头
- 评论标题
- 评论列表
- 评论分页
- 评论是封闭的消息。

## 模板头

用于定义一个模板。

```

<?php
/**
 * The template for displaying Comments.
 *
 * The area of the page that contains comments and the comment form.
 *
 * @package WordPress
 * @subpackage Twenty_Thirteen
 * @since Twenty Thirteen 1.0
 */

```

接下来，有一个测试来看看帖子是否受密码保护，如果是，则停止处理该模板。

```

/*
 * If the current post is protected by a password and the visitor has not yet
 * entered the password we will return early without loading the comments.
 */

```

```
if ( post_password_required() )
    return;
?>
```

最后，有一个测试，看看是否有与此帖子相关联的评论。

```
<div id="comments" class="comments-area">

    <?php if ( have_comments() ) : ?>
```

## 评论标题

打印出在评论上方显示的标题。

**注意：**使用 `_nx()` 翻译方法，因此其他开发人员可以提供替代语言翻译。

```
<h2 class="comments-title">
    <?php
        printf( _nx( 'One thought on "%2$s"', '%1$s thoughts on "%2$s"', get_co
mments_number(), 'comments title', 'twentythirteen' ),
            number_format_i18n( get_comments_number() ), '<span>' . get_the_tit
le() . '</span>' );
    ?>
</h2>
```

## 评论列表

以下代码片段使用 `wp_list_comments()` 函数获取注释的有序列表。

```
<ol class="comment-list">
    <?php
        wp_list_comments( array(
            'style'      => 'ol',
            'short_ping' => true,
            'avatar_size' => 74,
        ) );
    ?>
</ol><!-- .comment-list -->
```

## 评论分页

检查是否有足够的评论，以便添加评论导航，如果是，创建评论导航。

```
<?php
```

```
// Are there comments to navigate through?
if ( get_comment_pages_count() > 1 && get_option( 'page_comments' ) ) :
?>
<nav class="navigation comment-navigation" role="navigation">
  <h1 class="screen-reader-text section-heading"><?php _e( 'Comment navigation', 'twentythirteen' ); ?></h1>
  <div class="nav-previous"><?php previous_comments_link( __( '&larr; Older Comments', 'twentythirteen' ) ); ?></div>
  <div class="nav-next"><?php next_comments_link( __( 'Newer Comments &rarr;', 'twentythirteen' ) ); ?></div>
</nav><!-- .comment-navigation -->
<?php endif; // Check for comment navigation ?>
```

## 评论是封闭的消息。

如果评论未打开，则显示一条表示它们已关闭的行。

```
<?php if ( ! comments_open() && get_comments_number() ) : ?>
<p class="no-comments"><?php _e( 'Comments are closed.', 'twentythirteen' ); ?>
</p>
<?php endif; ?>
```

## 结束

本节结束评论循环，包括注释表单，并关闭评论包装器。

```
<?php endif; // have_comments() ?>

<?php comment_form(); ?>

</div><!-- #comments -->
```

## 评论分页

如果您有很多评论（这使您的页面很长），那么分页您的评论有很多潜在的好处。分页有助于提高页面加载速度，特别是在移动设备上。

启用评论分页是分两步完成的。

在WordPress中启用分页评论，方法是进入“Settings” > “Discussion”，并选中“Break comments into pages”框。您可以输入“每页评论数量”的任意数字。

打开您的comments.php模板文件，并添加以下行，您希望出现评论分页。

```
<div class="pagination">
```

```
<?php paginate_comments_links(); ?>
</div>
```

## 替代评论模板

在某些情况下，您可能希望在主题中显示您的评论。为此，您将构建一个备用文件（例如，short-comments.php），并调用如下：

```
<?php comments_template( '/short-comments.php' ); ?>
```

用于替代注释模板的文件的路径应该是相对于当前的主题根目录，并且包括任何子文件夹。所以如果自定义注释模板在主题中的文件夹中，调用它可能看起来像这样：

```
<?php comments_template( '/custom-templates/alternative-comments.php' ); ?>
```

## 参考方法

- wp\_list\_comments()：根据各种参数（包括在管理区域中设置的参数）显示帖子或页面的所有注释。
- comment\_form()：此标签输出完整的注释表单以在模板中使用。
- comments\_template()：加载在第一个参数中指定的注释模板
- paginate\_comments\_links()：为当前帖子的评论创建分页链接。
- get\_comments()：检索可能使用参数的注释
- get\_approved\_comments()：检索已提供的帖子ID的批准注释。

## 检索评论元素的函数参考

- get\_comment\_link()
- get\_comment\_author()
- get\_comment\_date()
- get\_comment\_time()
- get\_comment\_text()



# 分类模板

访问者点击类别，标签或自定义分类法的超链接时，WordPress会以相反的时间顺序显示一页帖子，按照该分类法过滤。

默认情况下，此页面使用index.php模板文件生成。您可以创建可选的模板文件来覆盖和优化index.php模板文件。本节介绍如何使用和创建此类模板。

## 分类模板层次结构

WordPress按照模板层次结构确定的顺序显示帖子。

category.php，tag.php和taxonomy.php模板允许通过分类过滤的帖子与未过滤的帖子或不同分类过滤的帖子进行对待。（注：帖子是指任何帖子类型 - 帖子，页面，自定义帖子类型等）。这些文件可让您指定特定的分类法或具体的分类术语。例如：

- taxonomy-{taxonomy}-{term}.php
- taxonomy-{taxonomy}.php
- tag-{slug}.php
- tag-{id}.php
- category-{slug}.php
- category-{ID}.php

因此，您可以在名为新闻的动物分类中设置与其他类别中过滤的帖子不同的页面上的所有帖子。

archive.php模板提供最通用的控件形式，为所有存档提供布局; 也就是一个显示帖子列表的页面。

## 类别

对于类别，WordPress寻找category-{slug}.php文件。如果不存在，WordPress则会为下一个层次级别（category-{ID}.php等）查找文件。如果WordPress找不到任何专门的模板或archive.php模板文件，它将使用index.php恢复为默认行为。

类别层次结构如下所示：

- category-{slug}.php: 例如，如果类别slug被命名为“news”，WordPress将寻找名为category-news.php的文件。
- category-{ID}.php: 例如，如果类别ID为“6”，则WordPress将查找名为category-6.php的文件。
- category.php
- archive.php
- index.php

## 标签

对于标签，WordPress寻找tag-{slug}.php文件。 如果不存在，WordPress然后将查找下一个层次级别的文件，tag-{ID}.php等等。 如果WordPress无法找到任何专门的模板或archive.php模板文件，它将恢复为默认行为，使用index.php。

标签层次结构如下所示：

- tag-{slug}.php: 例如，如果标签的命名为“sometag”，WordPress将会找到一个名为tag-sometag.php的文件。
- tag-{id}.php: 例如，如果标签的ID为“6”，则WordPress会查找名为tag-6.php的文件。
- tag.php
- archive.php
- index.php

## 自定义分类

自定义分类层次结构与上述类别和标签层次结构类似。 WordPress寻找taxonomy-{taxonomy}-{term}.php文件。 如果不存在，WordPress则会为下一个层次级别taxonomy-{taxonomy}.php等查找文件。 如果WordPress无法找到任何专门的模板或archive.php模板文件，它将恢复为默认行为，使用index.php。

自定义分类法的层次结构如下所示：

- taxonomy-{taxonomy}-{term}.php: 例如，如果分类法被命名为“sometax”，并且分类术语是“someterm”，则WordPress将寻找一个名为taxonomy-sometax-someterm.php的文件。
- taxonomy-{taxonomy}.php: 例如，如果分类法被命名为“sometax”，WordPress将会查找一个名为taxonomy-sometax.php的文件
- taxonomy.php
- archive.php
- index.php

## 创建分类模板文件

现在您决定要根据分类法为内容创建自定义设计，从哪里开始？

而不是从一个空白文件开始，最好复制层次结构中的下一个文件（如果存在）。 如果您已经创建了archive.php，请创建一个名为category.php的副本，并根据您的设计需求进行修改。 如果您没有archive.php文件，请使用主题的index.php副本作为起点。

如果要创建任何分类模板文件，请按照相同的步骤进行操作。 使用您的archive.php，category.php，tag.php或index.php的副本作为起点。

## 例子

现在，您已经在主题目录中选择了需要修改的模板文件，我们来看一些例子。

## 将文本添加到类别页面

静态文本上面的帖子#Static Text上面的帖子

假设您希望在类别页面上的帖子列表之前显示一些静态文本。“静态”是保持不变的文字，无论下面显示哪些帖子，无论显示哪个类别。

打开您的文件和上面的模板文件的循环部分，插入以下代码：

```
<p>
This is some text that will display at the top of the Category page.
</p>
```

此文本仅显示在显示该类别中的帖子的存档页面上。

## 某些类别页面上的不同文本

如果要根据访问者使用的类别页面显示不同的文本，该怎么办？您可以添加默认文本到主要的category.php文件，并创建特殊的category-{slug}.php文件，每个文件都有自己的版本，但这将在您的主题中创建大量的文件。相反，您可以使用条件标签。

再次，这个代码将在循环之前添加：

```
<?php if (is_category('Category A')) : ?>
    <p>This is the text to describe category A</p>
<?php elseif (is_category('Category B')) : ?>
    <p>This is the text to describe category B</p>
<?php else : ?>
    <p>This is some generic text to describe all other category pages,
    I could be left blank</p>
<?php endif; ?>
```

此代码执行以下操作：

检查访问者是否请求了类别A。如果是，它显示第一个文本。

如果用户没有请求类别A，请检查B类。如果是，则显示第二条文本。

显示默认文本，如果没有请求。

## 仅在存档首页上显示文本

如果您的帖子比您的档案的一页更适合，则该类别会分成多个页面。也许你想显示静态文本，如果用户在结果的第一页。

为此，请使用查询\$paged WordPress变量值的PHP if语句。

将以下内容放在循环中：

```
<?php if ( $paged < 2 ) : ?>
    <p>Text for first page of Category archive.</p>
<?php else : ?>
<?php endif; ?>
```

此代码询问显示的页面是否是归档的第一页。如果是，则显示第一页的文本。否则显示后续页面的文本。

## 修改帖子的显示方式

### 摘录与全职

---

您可以选择是否显示完整的帖子或只是摘录。通过显示摘录，缩短存档页面的长度。

打开你的文件并找到循环。寻找：

- `the_content()` 并替换为：
- `the_excerpt()` 如果您的主题显示摘录，但是要显示完整的内容，请将`the_excerpt`替换为`the_content`。

# 404页面

如果用户绊倒不存在或尚未创建的页面，则404页面对于添加到主题中很重要。同样重要的是，您的404页面为您的访问者提供了到达正确位置的方法。

接下来的几个步骤将帮助您为您的WordPress主题构建一个有用的404页面。

## 创建404.php文件

由于404.php文件用于未找到页面的错误，所以第一步是创建一个名为404.php的文件，并将其添加到您的WordPress主题文件夹中。

**提示：**您的404.php通常是您的index.php的一个很好的起点

## 添加文件头

要使用您的主题的header.php文件，请打开您的404.php文件。在顶部，添加一个描述该文件的注释，并确保包含您的主题标题。

例如：

```
/**
 * The template for displaying 404 pages (Not Found)
 */
get_header();
```

## 将身体添加到您的404页面

要使404页面功能正常，您必须将正文内容添加到模板中：

例：

```
<div id="primary" class="content-area">
    <div id="content" class="site-content" role="main">

        <header class="page-header">
            <h1 class="page-title"><?php _e( 'Not Found', 'twentythirteen' )
; ?></h1>
        </header>

        <div class="page-wrapper">
            <div class="page-content">
                <h2><?php _e( 'This is somewhat embarrassing, isn't it?', '
twentythirteen' ); ?></h2>
                <p><?php _e( 'It looks like nothing was found at this locat
```

```
ion. Maybe try a search?', 'twentythirteen' ); ?></p>
```

```
<?php get_search_form(); ?>
</div><!-- .page-content -->
</div><!-- .page-wrapper -->
```

```
</div><!-- #content -->
</div><!-- #primary -->
```

这个例子来自WordPress的Twenty Thirteen主题。这在404页面中添加了一些文本和搜索表单。

注意：您可以添加自己的文本，使您的404页面看起来更好。

## 添加页脚和侧边栏。

创建404页面的最后一步是添加页脚。如果需要，您还可以添加侧边栏。

例：

```
get_sidebar();
get_footer();
```

现在，您有一个带有搜索表单和一些文本的功能404页面，以防用户绊倒错误的页面。一旦你确定404.php已经上传，你可以测试你的404页面。只需键入不存在的网站的URL地址。您可以尝试像 `http://example.com/fred.php`。这确实会导致404错误，除非你实际上有一个名为fred.php的PHP文件。

# 主题功能

---

现在，您已经熟悉主题开发的基础知识了，现在该开始深入挖掘主题世界了。本节将介绍创建主题时需要考虑的所有基本主题功能：

- 自定义标题 - 学习如何给用户添加自己的标题图像的灵活性
- 侧边栏 - 将主题区域添加到用户可以添加窗口小部件的主题中
- 小部件 - 创建可以添加到侧边栏的可重用的PHP对象
- 导航菜单 - 在主题内注册和显示菜单
- 分类模板 - 为您的分类归档页面创建模板
- 分页 - 使用WordPress的内置分页
- 评论 - 自定义评论模板
- 媒体 - 使用WordPress核心的媒体功能
- 精选图片和发布缩略图 - 使用和自定义帖子的缩略图
- 发布格式 - 创建不同的格式来显示用户的帖子
- 国际化 - 学习如何准备您的主题翻译成不同的语言
- 本地化 - 学习如何翻译WordPress主题
- 辅助功能 - 了解可访问性最佳实践，确保每个人都可以使用您的主题
- 管理菜单 - 将菜单项添加到WordPress管理
- 404页面 - 创建自定义404页面

如果你是新主题，那么值得你通过每个部分的方法，但是如果你正在寻找特定的东西，你也可以跳入一个部分。

# 核心支持的功能

---

此页面将列出并稍微详细介绍所有核心支持的功能。 每个功能都将有自己的页面。



# 管理菜单

---

这不再是推荐使用主题选项的方式。 Customizer API是为用户提供更多控制和灵活性的推荐方法

主题作者可能需要提供设置屏幕，因此用户可以自定义主题的使用或作品。 执行此操作的最佳方法是创建一个管理菜单项，允许用户从所有管理屏幕访问该设置屏幕。

## 参考方法

### 菜单页

---

- `add_menu_page()`
- `add_object_page()`
- `add_utility_page()`
- `remove_menu_page()`

### 子菜单页

---

- `add_submenu_page()`
- `remove_submenu_page()`

### WordPress管理菜单

---

- `add_dashboard_page()`
- `add_posts_page()`
- `add_media_page()`
- `add_links_page()`
- `add_pages_page()`
- `add_comments_page()`
- `add_theme_page()`
- `add_plugins_page()`
- `add_users_page()`
- `add_management_page()`
- `add_options_page()`

### 每个剧情需要一个钩子

---

要添加管理菜单，您需要做三件事：

- 创建一个包含菜单建立代码的函数。

- 如果要为网络添加菜单，请使用admin\_menu操作钩子或network\_admin\_menu注册上述功能。
- 创建屏幕的HTML输出，当单击菜单项时显示。

大多数开发人员忽略了此列表中的第二步。你不能简单地调用菜单代码。你需要把它放在一个函数中，然后注册这个函数。

以下是描述这三个步骤的简单示例。这将在“设置”顶层菜单下添加一个子级菜单项。选择时，该菜单项将显示非常基本的屏幕。

```
<?php
/** Step 2 (from text above). */
add_action( 'admin_menu', 'my_menu' );

/** Step 1. */
function my_menu() {
    add_options_page(
        'My Options',
        'My Menu',
        'manage_options',
        'my-unique-identifier',
        'my_options'
    );
}

/** Step 3. */
function my_options() {
    if ( !current_user_can( 'manage_options' ) ) {
        wp_die( __( 'You do not have sufficient permissions to access this page'
        . ' ) );
    }
    echo 'Here is where I output the HTML for my screen.';
    echo '</div><pre>';
}
?>
```

在此示例中，函数my\_menu()通过add\_options\_page()函数将新项添加到设置管理菜单。

注意：请注意，步骤2中的add\_action()调用在admin\_menu钩子下注册my\_menu()函数。没有这个，add\_action()调用，将会抛出一个未定义函数的PHP错误。最后，add\_options\_page()调用是指当有人点击菜单项时，包含要显示的实际页面（和要处理的PHP代码）的my\_options()函数。

以下部分将对这些步骤进行更详细的描述。请记住将菜单和页面的创建包含在函数中，并使用admin\_menu钩子来在正确的时间开始整个过程。

## 确定新菜单的位置

在创建新菜单之前，首先决定菜单是顶级菜单还是子级菜单项。顶级菜单显示为管理菜单中的新部分，并包含子级菜单项。这意味着子级菜单项是现有顶级菜单的成员。

主题很少需要创建一个新的顶级菜单。如果这个主题向WordPress引入了一个全新的概念，并且需要许多屏幕来做，那么这个主题可能需要一个新的顶级菜单。只有当您真正需要多个相关屏幕才能使WordPress做最初的设计时，才能考虑添加顶级菜单。新的顶级菜单的示例可能包括作业管理或会议管理。请注意，通过本机帖子类型注册，WordPress会自动创建顶级菜单来管理此类功能。

如果不需要创建顶级菜单，则需要决定在什么顶级菜单中放置新的子级菜单项。作为参考，几个主题添加了现有WordPress顶层菜单下的子级菜单项。

使用本指南的WordPress顶级菜单来确定您的子级菜单项的正确位置：

- Dashboard – 信息中心为您的网站，并包括更新选项更新WordPress核心，插件和主题。
- Posts – 显示用于撰写帖子（面向时间的内容）的工具。
- Media – 上传和管理您的图片，视频和音频。
- Links – 管理对其他博客和感兴趣的网站的引用。
- Pages – 显示用于编写称为页面的静态内容的工具。
- Comments – 控制和监管读者对帖子的回复。
- Appearance – 显示用于操纵theme/style文件，侧边栏等的控件。
- Plugins – 显示处理插件管理的控件，而不是插件本身的配置选项。
- Users – 显示用户管理控件。
- Tools – 管理博客数据的导出，导入甚至备份。
- Settings – 显示只有管理员应该查看的插件选项。
- Network Admin – 显示在网络上设置的插件选项。而不是admin\_menu，您应该使用network\_admin\_menu（另请参阅创建网络）

## 顶级菜单

如果您决定了您的主题需要一个全新的顶级菜单，您需要做的第一件事是使用add\_menu\_page()函数创建一个。

**注意：**如果您不需要顶级菜单，请跳到“子级菜单”。

参数值：

- page\_title – 选择菜单时要显示在页面的标题标签中的文本。
- menu\_title – 菜单的屏幕上的名称文本。
- capability – 该菜单显示给用户所需的功能。当使用Settings API处理表单时，您应该在此处使用manage\_options，因为用户将无法保存其中的选项。用户级别已弃用，不能在此处使用。
- menu\_slug – 这个菜单由这个菜单引用（这个菜单应该是唯一的）。在版本3.0之前，这被称为文件（或句柄）参数。如果功能参数被省略，则menu\_slug应该是处理菜单页面内容显示的PHP文件。

- function – 显示菜单页面的页面内容的功能。
- icon\_url – 要用于此菜单的图标的URL。 此参数是可选的。
- position – 菜单中的位置这个菜单应该出现。 默认情况下，如果省略此参数，菜单将显示在菜单结构的底部。 要查看当前的菜单位置，请在菜单加载后使用`print_r ( $GLOBALS ['menu'] )`。
- Sub-Level Menus – 一旦定义了顶级菜单，或者您选择使用现有的WordPress顶级菜单，就可以使用`add_submenu_page()`函数定义一个或多个子级菜单项。

## 子菜单

---

如果您想要将新菜单项设为子菜单项，可以使用`add_submenu_page()`函数创建它。

参数值：

parent\_slug - 父菜单的小数名称，或标准WordPress管理员文件的文件名，提供要插入子菜单的顶级菜单，或者如果此子菜单正在进行，则为插件文件 自定义顶级菜单。 例子：

- Dashboard – `add_submenu_page('index.php', ...)`
- Posts – `add_submenu_page('edit.php', ...)`
- Media – `add_submenu_page('upload.php', ...)`
- Links – `add_submenu_page('link-manager.php', ...)`
- Pages – `add_submenu_page('edit.php?post_type=page', ...)`
- Comments – `add_submenu_page('edit-comments.php', ...)`
- Custom Post Types – `add_submenu_page('edit.php?post_type=your_post_type', ...)`
- Appearance – `add_submenu_page('themes.php', ...)`
- Plugins – `add_submenu_page('plugins.php', ...)`
- Users – `add_submenu_page('users.php', ...)`
- Tools – `add_submenu_page('tools.php', ...)`
- Settings – `add_submenu_page('options-general.php', ...)`
- page\_title – 当子菜单处于活动状态时，将进入页面的HTML页面标题的文本。
- menu\_title – 选择菜单时要显示在页面的标题标签中的文本。
- capability – 该菜单显示给用户所需的功能。 用户级别已弃用，不能在此处使用。
- menu\_slug – 对于现有的WordPress菜单，处理菜单页面内容显示的PHP文件。 对于自定义顶级菜单的子菜单，此子菜单页面的唯一标识符。
- function – 显示菜单页面的页面内容的功能。 在技术上，如在`add_menu_page`函数中，函数参数是可选的，但是如果没有提供，那么WordPress基本上会假设包含PHP文件将生成管理屏幕，而不调用函数。

## 使用包装器功能

---

由于大多数子级菜单属于“设置”，“工具”或“外观”菜单，所以WordPress提供了包装功能，可以使这些顶

级菜单中的子级菜单项更容易。 请注意，功能名称可能与管理UI中看到的名称不一致，因为它们随时间变化：

- Dashboard

```
<?php add_dashboard_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Posts

```
<?php add_posts_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Media

```
<?php add_media_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Links

```
<?php add_links_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Pages

```
<?php add_pages_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Comments

```
<?php add_comments_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Appearance

```
<?php add_theme_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Plugins

```
<?php add_plugins_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

```
ction); ?>
```

- Users

```
<?php add_users_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Tools

```
<?php add_management_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

- Settings

```
<?php add_options_page( $page_title, $menu_title, $capability, $menu_slug, $function); ?>
```

另请参阅主题选项，用于通过Customizer API创建选项的当前推荐方法。

## 示例

以下是一个快速示例，说明如何插入顶层菜单页和子菜单页，子菜单页上的标题与顶级页不同。在此示例中，`register_my_theme_more_settings_menu`是显示第一个子菜单页面的函数的名称：

```
function register_my_theme_settings_menu() {
    add_menu_page(
        "My Theme's Settings",
        "My Theme",
        "manage_options",
        "my-theme-settings-menu"
    );
}

function register_my_theme_more_settings_menu() {
    add_submenu_page(
        "my-themes-settings-menu",
        "More Settings for My Theme",
        "More Settings",
        "manage_options",
        "my-theme-more-settings-menu"
    );
}

add_action( "admin_menu", "register_my_theme_settings_menu");
```

```
add_action( "admin_menu", "register_my_theme_more_settings_menu");
```

以下是在自定义帖子类型菜单块下添加选项页面的示例（另见此处）：

## 插入页面

以下是一个如何将多个菜单插入到各个地方的示例：

```
<?php
// Hook for adding admin menus
add_action('admin_menu', 'mt_add_pages');

// Action function for hook above
function mt_add_pages() {
// Add a new submenu under Settings:
add_options_page(__('Test Settings', 'menu-test'), __('Test Settings', 'menu-test'),
    'manage_options', 'testsettings', 'mt_settings_page');

// Add a new submenu under Tools:
add_management_page( __('Test Tools', 'menu-test'), __('Test Tools', 'menu-test'),
    'manage_options', 'testtools', 'mt_tools_page');

// Add a new top-level menu (ill-advised):
add_menu_page(__('Test Toplevel', 'menu-test'), __('Test Top-level', 'menu-test'),
    'manage_options', 'mt-top-level-handle', 'mt_toplevel_page' );

// Add a submenu to the custom top-level menu:
add_submenu_page('mt-top-level-handle', __('Test Sub-Level', 'menu-test'), __('Test Sub-Level', 'menu-test'), 'manage_options', 'sub-page', 'mt_sublevel_page');

// Add a second submenu to the custom top-level menu:
add_submenu_page('mt-top-level-handle', __('Test Sub-Level 2', 'menu-test'), __('Test Sub-Level 2', 'menu-test'), 'manage_options', 'sub-page2', 'mt_sublevel_page2');
}

// mt_settings_page() displays the page content for the Test settings sub-menu
function mt_settings_page() {
    echo "</pre>";
    <h2>" . __( 'Test Settings', 'menu-test' ) . "</h2>";
    <pre>";
}

// mt_tools_page() displays the page content for the Test Tools sub-menu
function mt_tools_page() {
    echo "</pre>";
    <h2>" . __( 'Test Tools', 'menu-test' ) . "</h2>";
}
```

```

    <pre>
    ";
}

// mt_toplevel_page() displays the page content for the custom Test Top-Level menu
function mt_toplevel_page() {
    echo "</pre>
    <h2>" . __( 'Test Top-Level', 'menu-test' ) . "</h2>
    <pre>
    ";
}

// mt_sublevel_page() displays the page content for the first sub-menu
// of the custom Test Toplevel menu
function mt_sublevel_page() {
    echo "</pre>
    <h2>" . __( 'Test Sub-Level', 'menu-test' ) . "</h2>
    <pre>
    ";
}

// mt_sublevel_page2() displays the page content for the second sub-menu
// of the custom Test Top-Level menu
function mt_sublevel_page2() {
    echo "</pre>
    <h2>" . __( 'Test Sub-Level 2', 'menu-test' ) . "</h2>
    <pre>
    ";
}
?>

```

## 样品菜单页

**注意：**有关创建设置页面的信息，请参阅设置API。

前面的示例包含几个虚拟函数，如mt\_settings\_page()作为实际页面内容的占位符。我们来扩大他们。如果您想创建一个名为mt\_favorite\_color的选项，如果网站所有者可以通过“设置”页面输入最喜欢的颜色，该怎么办？mt\_options\_page ( ) 函数需要在屏幕上输出数据输入表单，并处理输入的数据。

这是一个这样做的功能：

```

// mt_settings_page() displays the page content for the Test settings sub-menu
function mt_settings_page() {
    //must check that the user has the required capability
    if ( !current_user_can( 'manage_options' ) )
    {

```



```

        wp_die( __( 'You do not have sufficient permissions to access this page.'
    ) );
    }

    // Variables for the field and option names
    $opt_name = 'mt_favorite_color';
    $hidden_field_name = 'mt_submit_hidden';
    $data_field_name = 'mt_favorite_color';

    // Read in existing option value from database
    $opt_val = get_option( $opt_name );

    // See if the user has posted us some information
    // If they did, this hidden field will be set to 'Y'
    if( isset($_POST[ $hidden_field_name ]) && $_POST[ $hidden_field_name ] ==
'Y' ) {
        // Read their posted value
        $opt_val = $_POST[ $data_field_name ];

        // Save the posted value in the database
        update_option( $opt_name, $opt_val );

        // Put a "Settings updated" message on the screen
?>
<div class="updated"></div>

<div class="wrap">
<?php
// Header
echo "<h2>" . __( 'Menu Test Settings', 'menu-test' ) . "</h2>";

// Settings form
?>
<form action="" method="post" name="form1"></form>
<?php _e("Favorite Color:", 'menu-test' ); ?>

<hr />
</div>
<?php } ?>

```

几个注释：

- WordPress函数（如add\_menu\_page()和add\_submenu\_page()）具有一个功能，用于确定是否显示顶级或子级菜单。
- 挂接以处理页面输出的功能必须检查用户是否具有所需的功能。
- WordPress管理功能负责验证用户登录，因此您不必担心您的功能。
- 上述功能示例已被国际化 - 有关详细信息，请参阅I18n for WordPress开发人员。

- 在将数据输入表单放在屏幕上之前，该函数将处理任何输入的数据，以便新值将以表单（而不是数据库的值）显示。
- 您不必担心这是第一次工作，因为WordPress `update_option`函数会自动向数据库添加一个选项，如果它不存在。
- 这些管理菜单添加过程将在您每次导航到管理员中的页面时进行解析。因此，如果您正在撰写一个没有选项页面的主题，但稍后添加一个主题，您可以使用上面的说明添加它，并重新上传，并调整，直到您满意为止。换句话说，菜单在激活主题时不会“永久添加”或放入数据库。他们在飞行中解析，所以您可以随意添加或减少菜单项，重新上传，更改将立即反映。

## 页钩子后缀

添加新管理菜单的每个功能 - `add_menu_page()`，`add_submenu_page()`及其专门版本（如 `add_options_page()`）返回一个名为“页面钩子后缀”的特殊值。它可以稍后用作挂钩，只能在该特定页面上调用一个动作。

一个这样的动作钩子是`load-{page_hook}`，其中`{page_hook}`是这些`add_*_page()`函数之一返回的值。当加载特定页面时调用此钩子。在下面的示例中，它用于在所有管理页面上显示“主题未配置”通知，但插件的选项页面除外：

```
<?php
add_action( 'admin_menu', 'my_menu' );

// Here you can check if plugin is configured (e.g. check if some option is set ). If not, add new hook.
// In this example hook is always added.
add_action( 'admin_notices', 'my_admin_notices' );

function my_menu() {
    // Add the new admin menu and page and save the returned hook suffix
    $hook_suffix = add_options_page( 'My Options', 'My Theme', 'manage_options', 'my-unique-identifier', 'my_options' );
    // Use the hook suffix to compose the hook and register an action executed when plugin's options page is loaded
    add_action( 'load-' . $hook_suffix , 'my_load_function' );
}

function my_load_function() {
    // Current admin page is the options page for our plugin, so do not display the notice
    // (remove the action responsible for this)
    remove_action( 'admin_notices', 'my_admin_notices' );
}

function my_admin_notices() {
    echo '<pre>
```

```
<div class="updated fade" id="notice">
My Plugin is not configured yet. Please do it now.</div>
</pre>';
}

function my_options() {
    if (!current_user_can('manage_options')) {
        wp_die( __('You do not have sufficient permissions to access this page.') );
    }

    echo '</pre>';
    <div class="wrap">';
    echo 'Here is where the form would go if I actually had options.';
    echo '</div>';
    <pre>
    ';
}
?>
```

## 资源

关于wp-hackers的顶级菜单讨论

管理员菜单编辑器插件

# 自定义Headers

自定义标头允许网站所有者将自己的“标题”图像上传到其网站，可以将其放置在某些页面的顶部。这些可以由用户通过管理面板的“外观>标题”部分中的可视化编辑器定制和裁剪。您也可以将文本放在标题下方或顶部。为了支持流体布局和响应设计，这些头部也可能是灵活的。标题使用get\_custom\_header()放置在主题中，但必须首先使用add\_theme\_support()将其添加到您的functions.php文件中。自定义标题是可选的。要使用文本设置基本的，灵活的自定义标题，您将包括以下代码：

```
function themename_custom_header_setup() {
    $args = array(
        'default-image'      => get_template_directory_uri() . 'img/default-image.jpg',
        'default-text-color' => '000',
        'width'              => 1000,
        'height'             => 250,
        'flex-width'         => true,
        'flex-height'        => true,
    )
    add_theme_support( 'custom-header', $args );
}
add_action( 'after_setup_theme', 'themename_custom_header_setup' );
```

使用after\_setup\_theme钩子，以便在主题加载后自动标题被注册。

## 什么是自定义标题？

当您在主题中启用自定义标题时，用户可以使用WordPress主题定制程序更改其标题图像。这给用户更多的控制和灵活性，在他们的网站的外观。

## 向您的主题添加自定义标题支持

要在主题中启用自定义标题，请将以下内容添加到您的functions.php文件中：

```
add_theme_support( 'custom-header' );
```

启用自定义头文件时，可以通过将参数传递给add\_theme\_support()函数来配置其他几个选项。您可以使用数组将特定配置选项传递给add\_theme\_support函数：

```
function themename_custom_header_setup() {
    $defaults = array(
        // Default Header Image to display
        'default-image'      => get_template_directory_uri() . '/images/head
```

```

ers/default.jpg',
    // Display the header text along with the image
    'header-text'           => false,
    // Header text color default
    'default-text-color'    => '000',
    // Header image width (in pixels)
    'width'                 => 1000,
    // Header image height (in pixels)
    'height'                => 198,
    // Header image random rotation default
    'random-default'       => false,
    // Enable upload of image file in admin
    'uploads'              => false,
    // function to be called in theme head section
    'wp-head-callback'     => 'wphead_cb',
    // function to be called in preview page head section
    'admin-head-callback'  => 'adminhead_cb',
    // function to produce preview markup in the admin screen
    'admin-preview-callback' => 'adminpreview_cb',
);
}
add_action( 'after_setup_theme', 'themename_custom_header_setup' );

```

## 灵活的头部图像

如果阵列中没有包含flex-height或flex-width，那么height和width将是固定的大小。如果包括柔度高度和弹性宽度，则将使用高度和宽度作为建议的尺寸。

## 标题文字

默认情况下，用户可以选择是否在图像上显示标题文本。没有选项强制用户的标题文本，但是如果完全删除标题文本，可以在参数中将'header-text'设置为'false'。这将删除标题文本和选项来切换它。

# 示例

## 设置自定义标题图像

当用户首次安装您的主题时，您可以在选择自己的标题之前添加默认标题。这样，用户可以更快地设置主题，并使用默认图片，直到他们准备上传自己的主题。

设置默认标题图像980px宽度和60px高度：

```

$header_info = array(
    'width'           => 980,
    'height'          => 60,
    'default-image' => get_template_directory_uri() . '/images/sunset.jpg',
);

```

```

add_theme_support( 'custom-header', $header_info );

$header_images = array(
    'sunset' => array(
        'url'           => get_template_directory_uri() . '/images/sunset.j
pg',
        'thumbnail_url' => get_template_directory_uri() . '/images/sunset_t
humbnail.jpg',
        'description'   => 'Sunset',
    ),
    'flower' => array(
        'url'           => get_template_directory_uri() . '/images/flower.j
pg',
        'thumbnail_url' => get_template_directory_uri() . '/images/flower_t
humbnail.jpg',
        'description'   => 'Flower',
    ),
);
register_default_headers( $header_images );

```

不要忘记调用register\_default\_headers()来注册默认映像。在这个例子中，sunset.jpg是默认的图像，而flower.jpg是Customizer中的另类选择。

从管理屏幕，单击外观>标题以在定制器中显示标题图像菜单。请注意，add\_theme\_support()中指定的宽度和高度按建议大小显示，而flower.jpg显示为可选项。

## 使用灵活的标题

默认情况下，用户必须裁剪上传的图像以适应您指定的宽度和高度。但是，您可以通过将“flex-width”和“flex-height”指定为true，让用户上传任何高度和宽度的图像。这将让用户在上传新照片时跳过裁剪步骤。

设置灵活的标题：

```

$args = array(
    'flex-width'      => true,
    'width'           => 980,
    'flex-height'     => true,
    'height'          => 200,
    'default-image'   => get_template_directory_uri() . '/images/header.jpg',
);
add_theme_support( 'custom-header', $args );

```

将您的header.php文件更新为：

```

width; ?>" height="<?php echo get_custom_header()->height; ?>">

```

## 显示自定义标题

要显示自定义标题，函数`get_header_image()`将返回标题图像。`get_custom_header()`获取自定义头文件数据。

例如。下面显示了如何使用自定义标题图像来显示主题中的标题。以下代码进入`header.php`文件。

```
<?php if ( get_header_image() ) : ?>
    <div id="site-header">
        <a href="<?php echo esc_url( home_url( '/' ) ); ?>" rel="home">
            width; ?>" height="<?php echo get_custom_header()->height; ?>" alt="<?php echo esc_attr( get_bloginfo( 'name', 'display' ) ); ?>" />
        </a>
    </div>
<?php endif; ?>
```

## 向后兼容性

WordPress 3.4及更高版本支持自定义标题。如果您希望您的主题支持早于3.4的WordPress安装，您可以使用以下代码，而不是`add_theme_support( 'custom-header' );`

```
global $wp_version;
if ( version_compare( $wp_version, '3.4', '>=' ) ) :
    add_theme_support( 'custom-header' );
else :
    add_custom_image_header( $wp_head_callback, $admin_head_callback );
endif;
```

## 功能参考

- `header_image()` 显示标题图片网址。
- `get_header_image()` 检索自定义标头的标题图像。
- `get_custom_header()` 获取标题图像数据。
- `get_random_header_image()` 检索自定义标头的标题图像。
- `add_theme_support()` 注册给定功能的主题支持。
- `register_default_headers()` 注册要由Customizer显示的一些默认标题。

# 自定义Logo

## 什么是自定义Logo?

使用自定义Logo可以让网站所有者上传他们网站的图片，这些图片可以放在他们网站的顶部。 它可以从您的管理面板中的外观>标题上传。 应该使用`add_theme_support()`将自定义徽标支持首先添加到您的主题中，然后`the_custom_logo`在主题中调用。 自定义Logo是可选的，但如果主题作者在其主题中包含Logo，则应使用此功能。

## 向您的主题添加自定义Logo支持

要在主题中使用自定义徽标，请将以下内容添加到您的`functions.php`文件中：

```
add_theme_support( 'custom-logo' );
```

启用自定义Logo支持时，可以使用数组将参数传递给`add_theme_support()`函数来配置五个参数：

```
function themename_custom_logo_setup() {  
    $defaults = array(  
        'height'      => 100,  
        'width'       => 400,  
        'flex-height' => true,  
        'flex-width'  => true,  
        'header-text' => array( 'site-title', 'site-description' ),  
    );  
    add_theme_support( 'custom-logo', $defaults );  
}  
add_action( 'after_setup_theme', 'themename_custom_logo_setup' );
```

使用`after_setup_theme`钩子，以便在主题加载后注册自定义Logo支持。

- `height` 预期Logo高度（以像素为单位）自定义Logo还可以使用内置的图像大小，如缩略图，或使用`add_image_size()`注册自定义大小。
- `width` 预期Logo宽度（以像素为单位）自定义Logo还可以使用内置的图像大小，如缩略图，或使用`add_image_size()`注册自定义大小。
- `flex-height` 是否允许灵活的高度。
- `flex-width` 是否允许灵活的宽度。
- `header-text` 要隐藏的元素的类。 它可以在这里传递一个类名称数组，用于构成标题文本的所有元素，可以由Logo代替。

## 在主题中显示自定义Logo



可以使用`the_custom_logo()`函数在主题中显示自定义徽标。但是建议将代码包装在一个`function_exists()`调用中，以保持与旧版本WordPress的向后兼容性，如下所示：

```
if ( function_exists( 'the_custom_logo' ) ) {  
    the_custom_logo();  
}
```

通常，Logo会添加到主题的`header.php`文件中，但它也可以在其他地方。

如果要获取当前的标识URL（或使用自己的标记）而不是默认标记，则可以使用以下代码：

```
$custom_logo_id = get_theme_mod( 'custom_logo' );  
$logo = wp_get_attachment_image_src( $custom_logo_id , 'full' );  
if ( has_custom_logo() ) {  
    echo '';  
} else {  
    echo '<h1>'. esc_attr( get_bloginfo( 'name' ) ) . '</h1>';  
}
```

## 自定义Logo模板标签

要管理在前端显示自定义Logo，可以使用以下三个模板标签：

- `get_custom_logo()` - 返回自定义Logo的标记。
- `the_custom_logo()` - 显示自定义Logo的标记。
- `has_custom_logo()` - 返回一个布尔值`true/false`，是否设置了自定义徽标。

# 文章格式

主题使用文章格式以特定格式和风格展示帖子。邮政格式功能提供了可用于支持该功能的所有主题的标准格式列表。主题可能不支持列表中的每种格式;在这种情况下，使用户知道这是很好的形式。

主题不能引入不在标准化列表中的格式，即使是通过插件。这种标准化确保主题之间的兼容性和外部工具以一致的方式使用该功能的方式。

简而言之，通过支持邮政格式的主题，博客可以通过选择邮政格式来更改邮局的外观。

使用Asides作为示例，过去已创建一个名为Asides的类别，并将帖子分配给该类别，然后根据post\_class()或in\_category ( 'asides' ) 中的样式规则显示不同的内容。

使用Post格式，新方法允许主题添加对Post格式的支持（例如，add\_theme\_support ( 'post-formats' , array ( 'aside' ) ) ），然后可以在Publish meta框中选择Post格式保存帖子。可以使用get\_post\_format ( \$ post-> ID ) 的函数调用来确定格式，并且post\_class() 也将为pure-css样式创建 “format-asides” 类。

## 支持的格式

如果主题支持，则可以使用以下Post格式。

请注意，虽然实际的帖子内容不会更改，但主题可以根据所选格式显示不同的帖子。显示职位的方式完全取决于主题，但以下是对不同Post格式的典型用法的一般指导。

- aside – 通常风格没有标题。类似于Facebook笔记更新。
- gallery – 图像库。帖子可能包含一个图库的短码，并将附有图像附件。
- link – 指向另一个网站的链接。主题可能希望在帖子内容中使用第一个 `<a href="">` 标签作为该帖子的外部链接。一个替代方法可能是，如果帖子只包含一个URL，那么这将是URL，标题 ( post\_title ) 将是附加到锚点的名称。
- image – 单一图像。帖子中的第一个 `<img />` 标签可以被认为是图像。或者，如果帖子只包含一个URL，那将是图像URL，并且该帖子的标题 ( post\_title ) 将是该图像的标题属性。
- quote – 引用。可能会包含一个包含报价内容的blockquote。或者，引用可能只是内容，源/作者是标题。
- status – 一个简短的状态更新，类似于Twitter状态更新。
- video – 一个视频。第一个 `<video />` 标签或对象/嵌入到帖子内容可以被视为视频。或者，如果帖子只包含一个URL，那将是视频URL。如果在博客上启用视频支持（如通过插件），也可以将该视频作为帖子的附件。
- audio – 一个音频文件。可用于播客。
- chat – 聊天记录如下：

```
John: foo
Mary: bar
```

John: foo 2

注意：写入或编辑帖子时，“标准”表示没有指定后期格式。另外如果指定了无效的格式，默认情况下会应用“标准”（无格式）。

## 功能参考

### 主要功能

- `set_post_format()`
- `get_post_format()`
- `has_post_format()`

### 其他功能

- `get_post_format_link()`
- `get_post_format_string()`

### 添加主题支持

主题需要在`functions.php`文件中使用`add_theme_support()`来通过传递如下格式的数组来告诉WordPress支持的格式：

```
function themename_post_formats_setup() {  
    add_theme_support( 'post-formats', array( 'aside', 'gallery' ) );  
}  
add_action( 'after_setup_theme', 'themename_post_formats_setup' );
```

使用`after_setup_theme`钩子，以便在主题加载后注册后期格式支持。

### 添加帖子类型支持

Post Types需要在`functions.php`文件中使用`add_post_type_support()`来告诉WordPress支持哪些格式：

```
function themename_custom_post_formats_setup() {  
    // add post-formats to post_type 'page'  
    add_post_type_support( 'page', 'post-formats' );  
  
    // add post-formats to post_type 'my_custom_post_type'  
    add_post_type_support( 'my_custom_post_type', 'post-formats' );  
}  
add_action( 'init', 'themename_custom_post_formats_setup' );
```

或者在函数register\_post\_type()中，添加'post-formats'，在'supports'参数数组中：

```
$args = array(
    ...
    'supports' => array('title', 'editor', 'author', 'post-formats')
);
register_post_type('book', $args);
```

add\_post\_type\_support应该挂钩到init钩子，因为自定义帖子类型可能没有在后端主题上注册。

## 使用格式

在主题中，使用get\_post\_format()检查帖子的格式，并相应地更改其演示文稿。 请注意，默认格式的帖子将返回值为FALSE。 或者，使用has\_post\_format()条件标签：

```
if ( has_post_format( 'video' ) ) {
    echo 'this is the video format';
}
```

## 建议造型

格式的另一种方法是通过样式规则。 主题应该使用post\_class()函数在包围该文章的包装器代码添加动态样式类。 Post格式会导致使用“format-foo”名称以这种方式添加额外的类。

例如，可以通过将其放在主题的样式表中来隐藏状态格式帖子的帖子：

```
.format-status .post-title {
display:none;
}
```

每种格式都适用于某种类型的“风格”，如现代用法所规定。在应用样式时，请牢记每种格式的预期用法。

例如，旁边，链接和状态格式是简单，简短和次要的。这些将通常显示没有标题或作者信息。旁边可能包含一段或两段，而链接将只是一个连接到其中的URL的句子。链接和旁边可能都有一个链接到单个帖子页面（使用the\_permalink()），因此将允许评论，但状态格式很可能没有这样的链接。

另一方面，图像柱通常只包含单个图像，具有或不具有标题/文本以与它一起。音频/视频文章将是相同的，但添加了音频/视频。这三个中的任何一个可以使用插件或标准嵌入来显示其内容。标题和作者可能不会显示给他们，因为内容可能是不言自明的。

报价格式特别适合发布一个没有额外信息的人的简单报价。如果你把报价单放在帖子内容中，并把引用的人的名字放在帖子的标题中，那么你可以对帖子进行风格化，以便自己显示the\_content()，然后重新设置为blockquote格式，并使用the\_title()显示被引用的人的名称作为旁路。

在许多情况下，特别的聊天可能会趋向于等宽型显示。使用.format-chat上的一些样式，您可以使用等宽字体显

示帖子的内容，也许在灰色背景div或类似内容中，从而将其视为聊天会话。

## 子主题格式

---

子主题继承由父主题定义的帖子格式。在子主题中为post格式调用add\_theme\_support()必须比父主题的优先级更高，并覆盖现有的列表，而不是添加它。

```
add_action( 'after_setup_theme', 'childtheme_formats', 11 );  
function childtheme_formats(){  
    add_theme_support( 'post-formats', array( 'aside', 'gallery', 'link' ) );  
}
```

调用remove\_theme\_support ( 'post-formats' ) 将一起删除它。

# 置顶文章

一个置顶帖子的帖子将被放置在帖子顶部的顶部。 此功能仅适用于内置的帖子类型的帖子，而不适用于自定义的帖子类型。

## 如何置顶帖子

转到管理屏幕 > 帖子 > 添加新的或编辑

在右侧菜单中，单击发布组中的可见性编辑链接

点击粘贴此帖子到首页选项

##显示置顶的帖子

只显示第一个置顶的帖子。 至少有一个帖子必须指定为“置顶帖子”，否则循环将显示所有帖子：

```
$sticky = get_option( 'sticky_posts' );  
$query = new WP_Query( 'p=' . $sticky[0] );
```

显示第一个置顶的帖子，如果没有返回最后发布的帖子：

```
$args = array(  
    'posts_per_page' => 1,  
    'post__in' => get_option( 'sticky_posts' ),  
    'ignore_sticky_posts' => 1  
);  
$query = new WP_Query( $args );
```

只显示第一个置顶帖子，如果没有返回任何内容：

```
$sticky = get_option( 'sticky_posts' );  
$args = array(  
    'posts_per_page' => 1,  
    'post__in' => $sticky,  
    'ignore_sticky_posts' => 1  
);  
$query = new WP_Query( $args );  
if ( isset( $sticky[0] ) ) {  
    // insert here your stuff...  
}
```

## 不显示置顶帖子

从查询中排除所有置顶的帖子：

```
$query = new WP_Query( array( 'post__not_in' => get_option( 'sticky_posts' ) ) )
;
```

排除类别中的置顶帖子。返回类别中的所有帖子，但不要在顶部显示置顶帖子。“置顶”仍将显示在自然的位置（例如按日期）：

```
$query = new WP_Query( 'ignore_sticky_posts=1&posts_per_page=3&cat=6' );
```

排除类别中的置顶帖子。返回类别中的帖子，但完全置顶粘贴帖子，并遵守分页规则：

```
$paged = get_query_var( 'paged' ) ? get_query_var( 'paged' ) : 1;
$sticky = get_option( 'sticky_posts' );
$args = array(
    'cat' => 3,
    'ignore_sticky_posts' => 1,
    'post__not_in' => $sticky,
    'paged' => $paged
);
$query = new WP_Query( $args );
```

**注意：**如果希望此查询在您设置为静态首页的页面模板中工作，请使用`get_query_var( 'page' )`。

```
<?php
/* Get all Sticky Posts */
$sticky = get_option( 'sticky_posts' );

/* Sort Sticky Posts, newest at the top */
rsort( $sticky );

/* Get top 5 Sticky Posts */
$sticky = array_slice( $sticky, 0, 5 );

/* Query Sticky Posts */
$query = new WP_Query( array( 'post__in' => $sticky, 'ignore_sticky_posts' => 1
) );
?>
```

## 风格置顶帖子

为了帮助主题作者执行更简单的样式，`post_class()`函数用于将`class = "..."`添加到DIV，只需添加：

```
<div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
```

`post_class()`输出该div的 `class="whatever"` 片段。这包括几个不同的价值类别：`post`，`hentry`（对于 `hAtom`微格式页面），`category-X`（其中X是帖子的每个类别的块），以及`tag-X`（类似的，但带有标签）。它还为标记为粘滞帖子的帖子添加了“粘性”。

```
.sticky { color:red; }
```

注意：只有在主页的第一页（`is_home()`为true并且`is_paged()`为false）的置顶帖子中才添加“sticky”类。



# Sidebars

## 什么是Sidebars

侧边栏是您的主题的任何小部件区域。小部件区域是您主题中的地方，用户可以添加自己的小部件。您不需要在主题中包含侧边栏，但包括侧栏意味着用户可以通过定制程序或窗口小部件管理面板向窗口小部件区域添加内容。

小部件可以用于各种目的，从列出最近的帖子到进行实时聊天。

提示：名称“sidebars”来自于小部件区域通常在漫长的条带中创建到博客的左侧或右侧的时间。今天，边栏已经超越了原来的名字。它们可以包含在您网站的任何地方。将边栏视为包含小部件的任何区域。

## 注册Sidebar

要使用侧边栏，您必须在functions.php中注册它们。

要开始，register\_sidebar()有几个参数应该始终被定义，不管它们是否被标记为可选。这些包括x，y和z。

- name - 您的侧边栏的名称。这是用户将在“小部件”面板中看到的名称。
- id - 必须是小写。您将使用dynamic\_sidebar函数将其称为主题。
- description - 侧边栏的说明。这也将显示在管理窗口小部件面板中。
- class - 要分配给窗口小部件的HTML的CSS类名称。
- before\_widget - 在每个窗口小部件之前放置的HTML。
- after\_widget - 每个窗口小部件后面放置的HTML。应该用于从before\_widget关闭标签。
- before\_title - 放在每个窗口小部件的标题之前的HTML，例如标题标签。
- after\_title - 每个标题后放置的HTML。应该用于从before\_title关闭标签。

要注册边栏，我们使用register\_sidebar和widgets\_init函数。

```
function themename_widgets_init() {  
    register_sidebar( array(  
        'name'          => __( 'Primary Sidebar', 'theme_name' ),  
        'id'            => 'sidebar-1',  
        'before_widget' => '<aside id="%1$s" class="widget %2$s">',  
        'after_widget'  => '</aside>',  
        'before_title'  => '<h1 class="widget-title">',  
        'after_title'   => '</h1>',  
    ) );  
  
    register_sidebar( array(  
        'name'          => __( 'Secondary Sidebar', 'theme_name' ),  
        'id'            => 'sidebar-2',
```

```

        'before_widget' => '<ul><li id="%1$s" class="widget %2$s">',
        'after_widget'  => '</li></ul>',
        'before_title'  => '<h3 class="widget-title">',
        'after_title'   => '</h3>',
    ) );
}
add_action( 'widgets_init', 'themenname_widgets_init' );

```

注册边栏告诉WordPress，您将在Appearance > Widgets中创建一个新的小部件区域，用户可以将其小部件拖动到其中。注册侧边栏有两种功能：

- register\_sidebar ( )
- register\_sidebars ( )

第一个可以让您注册一个侧边栏，第二个可以让您注册多个侧边栏。

**提示：**建议您单独注册侧边栏，因为它可以为每个侧边栏提供唯一和描述性的名称。

## 示例

对于页眉和页脚中的小部件区域，将其标题为“标题小部件区域”和“页脚小部件区域”，而不是“边栏1”和“边栏2”（这是默认值）。这提供了侧边栏所在位置的有用描述。

以下代码，添加到functions.php注册一个侧边栏：

```

<?php
add_action( 'widgets_init', 'my_register_sidebars' );
function my_register_sidebars() {
    /* Register the 'primary' sidebar. */
    register_sidebar(
        array(
            'id'          => 'primary',
            'name'         => __( 'Primary Sidebar' ),
            'description'  => __( 'A short description of the sidebar.' ),
            'before_widget' => '<div id="%1$s" class="widget %2$s">',
            'after_widget'  => '</div>',
            'before_title'  => '<h3 class="widget-title">',
            'after_title'   => '</h3>',
        )
    );
    /* Repeat register_sidebar() code for additional sidebars. */
}
?>

```

代码执行以下操作：

- `register_sidebar` – 告诉WordPress你正在注册边栏
- `'name' => __( 'Primary Widget Area', 'mytheme' )` , – 是将出现在外观>小部件中的小部件区域的名称
- `'id' => 'sidebar-1'` – 在侧边栏分配一个ID。 WordPress使用 “id” 将小部件分配给特定的侧边栏。
- `before_widget/after_widget` – 分配给侧边栏的小部件的包装元素。 “%1 \$ s” 和 “%2 \$ s” 应该分别保留在id和class中，以便插件可以使用它们。 默认情况下，WordPress将其设置为列表项，但在上述示例中，它们已被更改为div。
- `before_title/after_title` – 小部件标题的包装元素。 默认情况下，WordPress将其设置为h2，但使用h3使其更具语义。

一旦您的侧栏被注册，您可以在主题中显示。

## 在主题中显示侧边栏

现在您的侧边栏已注册，您将要在主题中显示。 为此，有两个步骤：

- 创建sidebar.php模板文件，并使用dynamic\_sidebar函数显示边栏
- 使用get\_sidebar函数加载主题

## 创建侧栏模板文件

侧边栏模板包含您的侧边栏的代码。 WordPress识别文件sidebar.php和名称为sidebar-{name} .php的任何模板文件。 这意味着您可以在自己的模板文件中的每个侧边栏中组织文件。

## 示例:

1. 创建 sidebar-primary.php
2. 添加以下代码：

```
<div id="sidebar-primary" class="sidebar">
    <?php dynamic_sidebar( 'primary' ); ?>
</div>
```

请注意，dynamic\_sidebar采用\$index的单个参数，可以是边栏的名称或ID。

## 加载你的侧栏

要在您的主题中加载您的侧边栏，请使用get\_sidebar函数。 这应该被插入到你想要侧栏显示的模板文件中。 要加载默认的sidebar.php，请使用以下命令：

```
<?php get_sidebar(); ?>
```

要显示主侧边栏，请将\$name参数传递给函数：

```
<?php get_sidebar( 'primary' ); ?>
```

## 自定义您的侧栏

有很多方法可以自定义您的侧边栏。这里有些例子：

### 显示默认侧栏内容

如果用户尚未向边栏添加任何小部件，则可能希望显示内容。为此，您可以使用is\_sidebar\_active()函数检查边栏是否有任何小部件。这接受\$index参数，该参数应该是要检查的边栏的ID。

此代码检查边栏是否处于活动状态，如果不显示某些内容：

```
<div id="sidebar-primary" class="sidebar">
  <?php if ( is_active_sidebar( 'primary' ) ) : ?>
    <?php dynamic_sidebar( 'primary' ); ?>
  <?php else : ?>
    <!-- Time to add some widgets! -->
  <?php endif; ?>
</div>
```

### 显示默认窗口小部件

您可能希望您的侧边栏默认填充一些小部件。例如，显示搜索，存档和元小部件。为此，您将使用：

```
<div id="primary" class="sidebar">
  <?php do_action( 'before_sidebar' ); ?>
  <?php if ( ! dynamic_sidebar( 'sidebar-primary' ) ) : ?>
    <aside id="search" class="widget widget_search">
      <?php get_search_form(); ?>
    </aside>
    <aside id="archives" class="widget">
      <h1 class="widget-title"><?php _e( 'Archives', 'shape' ); ?></h1>
      <ul>
        <?php wp_get_archives( array( 'type' => 'monthly' ) ); ?>
      </ul>
    </aside>
    <aside id="meta" class="widget">
      <h1 class="widget-title"><?php _e( 'Meta', 'shape' ); ?></h1>
      <ul>
        <?php wp_register(); ?>
      </ul>
    </aside>
  </div>
```

```
        <li><?php wp_loginout(); ?></li>
        <?php wp_meta(); ?>
    </ul>
</aside>
<?php endif; ?>
</div>
```

# Widgets

小部件将内容和功能添加到小部件区域（也称为侧边栏）。小部件区域为用户定制其网站提供了一种方式。窗口小部件区域可以显示在多个页面上或仅在一个页面上。您的主题可能只有一个小部件区域或其中许多。可以使用小部件区域的一些示例如下：

- 使用小部件布置首页。这允许网站所有者决定在其主页的每个部分应该出现什么。
- 创建一个页脚，用户可以使用自己的内容进行自定义。
- 向博客添加可自定义的侧边栏。
- 小部件是输出一些HTML的PHP对象。相同类型的小部件可以在同一页面上多次使用（例如，文本小部件）。小部件可以将数据保存在数据库中（在选项表中）。

当您创建一种新的小部件时，它将显示在用户的“管理屏幕”的“外观” > “小部件”中。用户可以将窗口小部件添加到窗口小部件区域，并从WordPress管理员自定义窗口小部件设置。

## 内置和独立的小部件

默认WordPress安装包含一组小部件。除了这些标准的小部件，主题或插件可以包含额外的小部件。内置在主题或插件中的小部件的优点是提供额外的功能并增加小部件的数量。

缺点是，如果主题已更改或插件被停用，插件或主题小部件的功能将会丢失。但是，如果重新激活主题或插件，则小部件的数据和首选项将保存到选项表，并恢复。

如果您的主题包含小部件，则只有在该主题处于活动状态时才能使用。如果用户决定改变他们的主题，他们将无法访问该小部件。但是，如果插件包含插件，则用户可以更改其主题，而不会失去对窗口小部件功能的访问。

## 小部件解剖

视觉上，一个小部件包括两个方面：

- 标题区
- 小部件选项

例如，这里是管理员和前端内置文本小部件的布局：

- 示例可编辑小部件
- 管理区域中的小部件表单。

一个小部件，因为它可以访问一个网站访问者。网站访问者看到的小部件。此小部件的HTML输出如下所示：

```
<div id="text-7" class="widget widget_text"> <!-- Widget Container -->
    <div class="widget-wrap">
```

```

        <h4 class="widgettitle">This is a text widget</h4> <!-- Title -->
        <div class="textwidget"> <!-- Content of Widget -->
            I can put HTML in here.
            <a href="http://google.com">Search me!</a>
        </div>
    </div>
</div>

```

每个小部件都有自己的方式输出与正在显示的数据相关的HTML。小部件的包装器标签由其正在显示的小部件区域定义。

PHP需要创建一个内置的文本小部件，如下所示：

```

class My_Widget extends WP_Widget {

    function __construct() {

        parent::__construct(
            'my-text', // Base ID
            'My Text' // Name
        );

        add_action( 'widgets_init', function() {
            register_widget( 'My_Widget' );
        });

    }

    public $args = array(
        'before_title' => '<h4 class="widgettitle">',
        'after_title'   => '</h4>',
        'before_widget' => '<div class="widget-wrap">',
        'after_widget'  => '</div></div>'
    );

    public function widget( $args, $instance ) {

        echo $args['before_widget'];

        if ( ! empty( $instance['title'] ) ) {
            echo $args['before_title'] . apply_filters( 'widget_title', $instance['title'] ) . $args['after_title'];
        }

        echo '<div class="textwidget">';

        echo esc_html__( $instance['text'], 'text_domain' );

        echo '</div>';
    }
}

```

```

        echo $args['after_widget'];

    }

    public function form( $instance ) {

        $title = ! empty( $instance['title'] ) ? $instance['title'] : esc_html__(
            '', 'text_domain' );
        $text = ! empty( $instance['text'] ) ? $instance['text'] : esc_html__(
            '', 'text_domain' );
        ?>
        <p>
            <label for="<?php echo esc_attr( $this->get_field_id( 'title' ) );
?>"><?php esc_attr_e( 'Title:', 'text_domain' ); ?></label>
            <input class="widefat" id="<?php echo esc_attr( $this->get_field_id
( 'title' ) ); ?>" name="<?php echo esc_attr( $this->get_field_name( 'title' )
); ?>" type="text" value="<?php echo esc_attr( $title ); ?>">
        </p>
        <p>
            <textarea class="widefat" id="<?php echo esc_attr( $this->get_field
_id( 'text' ) ); ?>" name="<?php echo esc_attr( $this->get_field_name( 'text' )
); ?>" type="text" cols="30" rows="10"><?php echo esc_attr( $text ); ?></textare
a>
        </p>
        <?php

    }

    public function update( $new_instance, $old_instance ) {

        $instance = array();

        $instance['title'] = ( !empty( $new_instance['title'] ) ) ? strip_tags(
            $new_instance['title'] ) : '';
        $instance['text'] = ( !empty( $new_instance['text'] ) ) ? $new_instance[
            'text'] : '';

        return $instance;
    }

}

$my_widget = new My_Widget();

```

上面的代码将在后面的文章中详细解释。

## 开发小部件



要创建和显示窗口小部件，您需要执行以下操作：

- 通过扩展标准的WP\_Widget类及其一些功能来创建小部件的类。
- 注册您的小部件，以便它可以在Widgets屏幕中使用。
- 确保您的主题至少有一个小部件区域添加小部件。

## 你的小部件类

WP\_Widget类位于wp-includes / widgets.php中

```
class My_Widget extends WP_Widget {  
  
    public function __construct() {  
        // actual widget processes  
    }  
  
    public function widget( $args, $instance ) {  
        // outputs the content of the widget  
    }  
  
    public function form( $instance ) {  
        // outputs the options form in the admin  
    }  
  
    public function update( $new_instance, $old_instance ) {  
        // processes widget options to be saved  
    }  
  
}
```

每个这些功能的文档可以在窗口小部件类代码中找到：

- `__construct`：在管理员中设置具有说明，名称和显示宽度的小部件。
- `widget`：处理小部件选项并在页面上显示HTML。\$args参数提供了HTML可以用来显示窗口小部件标题类和窗口小部件内容类。
- `form`：显示用于设置窗口小部件选项的窗体。如果您的窗口小部件没有任何选项，则可以跳过此功能（尽管即使空白也是最佳做法）。
- `update`：将小部件选项保存到数据库。如果您的窗口小部件没有任何选项，则可以跳过此功能（尽管即使空白也是最佳做法）。

## 注册小部件

register\_widget()函数用于注册一个小部件。

使用widgets\_init钩子调用此函数：

```
add_action( 'widgets_init', function() { register_widget( 'My_Widget' ); } );
```

当您使用`register_sidebar()`注册窗口小部件时，将指定包含窗口小部件的HTML以及标题和窗口小部件内容的类。

## 示例文本小部件

从本文开头的示例中构建文本小部件。您将首先设置一个扩展`WP_Widget`类的窗口小部件类。

在类构造函数中，您将调用父构造函数并将其传递给您的窗口小部件的基本ID和名称。同样在类构造函数中，您将挂接到`widgets_init`操作以注册您的窗口小部件。

接下来，您将声明创建窗口小部件时使用的参数。你必须定义四个参数，`before_title`，`after_title`，`before_widget`和`after_widget`。这些参数将定义包装小部件标题和小部件本身的代码。

定义参数后，您将定义小部件功能。此函数有两个参数，前面的`$ args`数组和窗口小部件的`$ instance`，它将处理表单中的选项，并在您网站的前端显示窗口小部件的HTML。在上面的例子中，窗口小部件功能简单地输出窗口小部件标题，同时通过`widget_title`过滤器。然后，它放置一个简单的小部件包装器和小部件的文本字段的内容。如示例中所述，您可以从存储在`$ instance`中的小部件访问选项。

接下来你将定义表单函数。此函数使用一个参数`$ instance`，并输出用户在Widgets管理屏幕中用于创建窗口小部件的窗体。在上面的例子中，函数首先定义`$ title`和`$ text`变量，并将它们设置为先前输入的值（如果存在这些值）。然后，它输出一个简单的表单，带有标题的文本字段和文本内容的textarea。

最后你将定义更新功能。此函数具有两个参数`$ new_instance`和`$ old_instance`，并且负责在提交时使用新选项更新您的小部件。在这里，您只需将`$ instance`定义为空数组。然后，您将标题和文本键设置为`$ new_instance`值（如果存在）。然后你返回`$ instance`。

最后，当定义所有上述内容时，您将实例化新的小部件类并测试您的工作。

## 示例小部件

```
/**
 * Adds Foo_Widget widget.
 */
class Foo_Widget extends WP_Widget {

    /**
     * Register widget with WordPress.
     */
    public function __construct() {
        parent::__construct(
            'foo_widget', // Base ID
            'Foo_Widget', // Name
            array( 'description' => __( 'A Foo Widget', 'text_domain' ), ) // A
            args
        );
    }
}
```

```

}

/**
 * Front-end display of widget.
 *
 * @see WP_Widget::widget()
 *
 * @param array $args      Widget arguments.
 * @param array $instance Saved values from database.
 */
public function widget( $args, $instance ) {
    extract( $args );
    $title = apply_filters( 'widget_title', $instance['title'] );

    echo $before_widget;
    if ( ! empty( $title ) ) {
        echo $before_title . $title . $after_title;
    }
    echo __( 'Hello, World!', 'text_domain' );
    echo $after_widget;
}

/**
 * Back-end widget form.
 *
 * @see WP_Widget::form()
 *
 * @param array $instance Previously saved values from database.
 */
public function form( $instance ) {
    if ( isset( $instance[ 'title' ] ) ) {
        $title = $instance[ 'title' ];
    }
    else {
        $title = __( 'New title', 'text_domain' );
    }
    ?>
<p>
<label for="<?php echo $this->get_field_name( 'title' ); ?>"><?php _e(
'Title:' ); ?></label>
<input class="widefat" id="<?php echo $this->get_field_id( 'title' ); ?
>" name="<?php echo $this->get_field_name( 'title' ); ?>" type="text" value="<?
php echo esc_attr( $title ); ?>" />
</p>
<?php
}

/**
 * Sanitize widget form values as they are saved.
 *

```

```

* @see WP_Widget::update()
*
* @param array $new_instance Values just sent to be saved.
* @param array $old_instance Previously saved values from database.
*
* @return array Updated safe values to be saved.
*/
public function update( $new_instance, $old_instance ) {
    $instance = array();
    $instance['title'] = ( !empty( $new_instance['title'] ) ) ? strip_tags(
$new_instance['title'] ) : '';

    return $instance;
}

} // class Foo_Widget

```

然后可以在widgets\_init钩子中注册此示例窗口小部件：

```

// Register Foo_Widget widget
add_action( 'widgets_init', function() { register_widget( 'Foo_Widget' ); } );

```

##使用命名空间的示例

如果您使用PHP 5.3命名空间，则应直接调用构造函数，如下例所示：

```

namespace a\b\c;

class My_Widget_Class extends \WP_Widget {
    function __construct() {
        parent::__construct( 'baseID', 'name' );
    }
    // ... rest of the functions
}

```

并使用以下命令调用注册窗口

```

add_action( 'widgets_init', function() { register_widget( 'a\b\c\My_Widget_Class' ); } );

```

有关详细信息，请参阅堆栈交换中的此答案。

## 特殊考虑

如果要在其他模板文件中使用小部件，而不是侧边栏，则可以the\_widget()以编程方式显示。 该函数接受窗口小

部件类名。 您将小部件类名称传递给这样的函数：

```
<h1><?php the_title(); ?></h1>

<div class="content">
    <?php the_content(); ?>
</div>

<div class="widget-section">
    <?php the_widget( 'My_Widget_Class' ); ?>
</div>
```

如果您需要在页面上的特定区域中使用窗口小部件，则可能需要使用此方法，例如在网站首页的某个部分中显示表单旁边的事件列表，或者显示电子邮件捕获表单 一个超级菜单旁边的导航。

# 导航菜单

导航菜单是您主题中可定制的菜单。它们允许用户在菜单中添加页面，帖子，类别和URL。要创建导航菜单，您需要注册，然后在主题中的适当位置显示菜单。

## 注册菜单

在主题的functions.php中，您需要注册您的菜单。这将设置出现在外观 - > 菜单的名称。

首先，你将使用register\_nav\_menus()来注册菜单。

在此示例中，两个位置添加到“管理位置”选项卡：“标题菜单”和“额外菜单”。

```
function register_my_menus() {
    register_nav_menus(
        array(
            'header-menu' => __( 'Header Menu' ),
            'extra-menu' => __( 'Extra Menu' )
        )
    );
}
add_action( 'init', 'register_my_menus' );
```

## 显示菜单

一旦你注册了菜单，你需要使用wp\_nav\_menu()来告诉你的主题哪里可以显示它们。例如，将以下代码添加到您的header.php文件中，以显示上面注册的标题菜单。

```
wp_nav_menu( array( 'theme_location' => 'header-menu' ) );
```

**注意：**参数的完整列表可以在函数引用的wp\_nav\_menu()页面中找到

对您想要在主题中显示的任何其他菜单重复此过程。或者，您可以添加容器类，允许您使用CSS对菜单进行样式化。

```
wp_nav_menu(
    array(
        'theme_location' => 'extra-menu',
        'container_class' => 'my_extra_menu_class'
    )
);
```

**注意：**可以在函数引用的wp\_nav\_menu()页面中找到CSS类的完整列表。您可以使用这些来设计菜单。

## 显示其他内容

以下是“Twenty Seventeen” 页面社交菜单的简化版本，它在菜单项标签文本之前和之后显示跨度元素。

```
wp_nav_menu(
    array(
        'menu' => 'primary',
        'link_before' => '<span class="screen-reader-text">',
        'link_after' => '</span>',
    )
);
```

输出将显示为...

```
<div class="menu-social-container">
  <ul id="menu-social">
    <li id="menu-item-1">
      <a href="http://twitter.com/"><span class="screen-reader-text">Twitter</s
pan>
    </li>
  </ul>
</div>
```

注意：要在每个菜单项的 `<li>` 和 `<a>` 元素之间显示文本，请在参数之前和之后使用。

## 定义回调

默认情况下，当没有找到指定的菜单或位置时，WordPress显示第一个非空菜单，或者当没有选择自定义菜单时，WordPress将生成一个页面菜单。为了防止这种情况，请使用`theme_location`和`fallback_cb`参数。

```
wp_nav_menu(
    array(
        'menu' => 'primary',
        // do not fall back to first non-empty menu
        'theme_location' => '__no_such_location',
        // do not fall back to wp_page_menu()
        'fallback_cb' => false
    )
);
```

# 分页

分页可让您的用户通过多页内容来回浏览。

WordPress可以使用分页：

- 或者在一个页面上查看更多帖子的帖子列表
- 通过使用以下标签手动打破更长的帖子。

```
<!--nextpage-->
```

## 使用分页浏览帖子列表

在WordPress网站中最常用的分页是将长列表的帖子分解成单独的页面。 无论您是查看博客或网站的类别，归档或默认索引页面，默认情况下，WordPress仅显示每页10个帖子。 用户可以在阅读屏幕上更改每个页面上显示的帖子数量：管理员>设置>阅读。

## 循环与分页

此简化示例显示可以为主循环添加分页功能的位置。 在循环之前或之后添加功能。

```
<?php if ( have_posts() ) : ?>

    <!-- Add the pagination functions here. -->

    <!-- Start of the main loop. -->
    <?php while ( have_posts() ) : the_post(); ?>

    <!-- the rest of your theme's main loop -->

    <?php endwhile; ?>
    <!-- End of the main loop -->

    <!-- Add the pagination functions here. -->

    <div class="nav-previous alignleft"><?php next_posts_link( 'Older posts' ); ?></div>

    <div class="nav-next alignright"><?php previous_posts_link( 'Newer posts' ); ?></div>
```



```
<?php else : ?>

<?php _e('Sorry, no posts matched your criteria.');
```

```
<?php endif; ?>
```

## 显示分页链接的方法

**注意：**在使用分页循环的模板文件外部使用任何这些分页函数时，必须调用全局变量\$wp\_query。

```
function your_themes_pagination(){
    global $wp_query;
    echo paginate_links();
}
```

WordPress具有许多功能，用于显示您循环中其他页面的链接。这些功能中的一些仅在非常具体的上下文中使用。您将在单个页面上使用不同的功能，然后在归档页面上。以下部分介绍归档模板分页功能。之后的部分封面单后分页。

## 简单分页

- posts\_nav\_link

最简单的方法之一是posts\_nav\_link()。在你的循环之后，简单的把你的模板放在你的模板中。这将产生两个链接到下一页的帖子和上一页的帖子（如适用）。此功能适用于具有简单分页要求的主题。

```
posts_nav_link();
```

- next\_posts\_link & prev\_posts\_link

构建主题时，请使用next\_posts\_link()和prev\_posts\_link()。以控制上一页和下一页的页面链接出现在哪里。

```
next_posts_link();
previous_posts_link();
```

如果您需要将分页链接传递给PHP变量，则可以使用get\_next\_posts\_link()和get\_previous\_posts\_link()。

```
$next_posts = get_next_posts_link();
$prev_posts = get_previous_posts_link();
```

## 数字分页

当您有许多页面的内容时，显示页码列表是一个更好的体验，因此用户可以点击任何一个页面链接，而不必重复单击下一个或上一个帖子。WordPress提供了几个自动显示数字分页列表的功能。

对于WordPress 4.1+

如果您想要更强大的分页选项，则可以对WordPress 4.1及更高版本使用`the_posts_pagination()`。这将输出一组页码，其中包含上一页和下一页的链接。

```
the_posts_pagination();
```

对于4.1之前的WordPress

如果您希望分页支持旧版本的WordPress，则必须使用`paginate_links()`。

```
echo paginate_links();
```

## 单页之间的分页

所有以前的功能都应该用在索引和归档页面上。当您查看单个博文时，您必须使用`prev_post_link`和`next_post_link`。在您的`single.php`上的循环下面放置以下功能。

```
previous_post_link();
next_post_link();
```

## 一篇文章中的分页

WordPress为您提供了一个标签，可以放置在帖子内容中以启用该帖子的分页。

```
<!--nextpage-->
```

如果您在内容中使用该标签，则需要确保将`wp_link_pages`函数放在循环中的`single.php`模板中。

```
<?php if ( have_posts() ) : ?>

    <!-- Start of the main loop. -->
    <?php while ( have_posts() ) : the_post(); ?>

        <?php the_content(); ?>

        <?php wp_link_pages(); ?>

    <?php endwhile; ?>

<?php endif; ?>
```



# 媒体

WordPress使主题开发人员能够定制平台核心媒体功能的外观，感觉和功能。

## 常规

在WordPress中，您可以上传，存储和显示各种媒体，如图像，视频和音频文件。媒体可以通过媒体>在管理屏幕添加新的，或添加媒体按钮上传到邮政/页面编辑器。

如果在编辑屏幕中上传了媒体文件，它将被自动附加到正在创建或编辑的当前帖子中。如果是通过媒体的“添加新屏幕”或“媒体库”屏幕进行上传，则将不会附加，但在稍后插入后，可能会附加到帖子。

## 检索附件ID或图像ID

要检索附件ID，请使用get\_posts()或get\_children()函数。此示例通过指定ID来检索当前帖子的所有附件并获取附件的所有元数据。

```
// Insert into the Loop
$args = array(
    'post_parent'    => get_the_ID(),
    'post_type'      => 'attachment',
);
$attachments = get_posts( $args );
if ( $attachments ) {
    foreach ( $attachments as $attachment ) {
        $meta_data = wp_get_attachment_metadata( $attachment->ID, false );
    }
}
```

如果要仅从帖子ID中检索图像，请将post\_mime\_type指定为图像。

```
$args = array(
    'post_parent'    => get_the_ID(),
    'post_type'      => 'attachment',
    'post_mime_type' => 'image',
);
```

## 参考文献

- [get\\_posts\(\)](#)
- [get\\_children\(\)](#)
- [wp\\_get\\_attachment\\_metadata\(\)](#)

## 兼容的媒体格式

---

在媒体库中，您可以上传任何文件（网络管理员的未过滤的上传文件），而不仅仅是图像或视频文件，文件文件，甚至二进制文件。默认情况下，单站点管理员不具备unfiltered\_upload功能，并且需要设置该定义才能启动该功能。音频和视频文件由内部库MediaElement.js处理。

- 支持的音频格式
- 支持视频格式

## 无法检索附件

---

当您无法通过get\_posts()或get\_children()函数获取附加的媒体时，请确认您的媒体真正附加到该帖子。从管理屏幕中，单击媒体>库以打开媒体库，并确认媒体“已上传到”列中的值。

# Audio

## 音频

您可以直接嵌入音频文件，并使用简单的短码[`audio`]播放。 支持的文件类型是mp3，ogg，wma，m4a和wav。

## 音频短码

以下短码显示加载music.mp3文件的音频播放器：

```
[audio src="music.mp3"]
```

要使用模板文件的短代码，请使用do\_shortcode功能。 当music.mp3文件存储在（ theme\_directory ） / sounds目录中时，将以下代码插入到模板文件中：

```
$music_file = get_template_directory_uri() . "/sounds/music.mp3";  
echo do_shortcode('[audio mp3=' . $music_file . ']');
```

短代码创建音频播放器，如下面的屏幕截图所示。

## 循环和自动播放

支持以下基本选项：

### 循环

允许媒体循环。

- off - 不要循环播放媒体。 默认。
- on - 媒体将在完成后循环开始，并自动继续播放。

### 自动播放

导致媒体在媒体文件准备就绪后自动播放。

- 0 - 不要自动播放媒体。 默认。
- 1 - 一旦准备就绪，媒体就会播放。

以下示例在页面加载和循环后立即开始播放音乐。

```
echo do_shortcode('[audio mp3=' . $music_file . ' loop = "on" autoplay = 1]');
```

---

## 造型

---

如果要更改音频播放器的外观，可以通过定位默认类名“wp-audio-shortcode”来实现。如果将以下代码插入到style.css中，则会显示音频播放器的一半宽度。

```
.wp-audio-shortcode {  
    width: 50%;  
}
```

---

## 支持的音频格式

---

- mp3
- ogg
- wma
- m4a
- wav

---

## 参考文献

---


有关更多技术细节，如启用此功能的内部库，请参阅

<https://make.wordpress.org/core/2013/04/08/audio-video-support-in-core/>.

- Audio Shortcode
- Function do\_shortcode()

# Images

## 图片

注意：本节介绍媒体库中图像的处理。如果要显示位于主题目录中的图像文件，只需使用标签指定位置，并使用CSS进行样式化。

```

```

## 获取图片代码

要在媒体库中显示图像，请使用wp\_get\_attachment\_image()函数。

```
echo wp_get_attachment_image( $attachment->ID, 'thumbnail' );
```

您将获得具有所选缩略图大小的以下HTML输出

```

```

您可以为管理屏幕中的“设置” > “媒体”中设置的尺寸或任何一对宽度和高度指定其他尺寸，例如原始图像的“full”或“medium”和“large”。您还可以使用add\_image\_size()设置自定义大小的字符串；

```
echo wp_get_attachment_image( $attachment->ID, Array(640, 480) );
```

## 获取图像的URL

如果要获取图像的URL，请使用wp\_get\_attachment\_image\_src()。如果没有图像可用，它返回一个数组（URL，width，height，is\_intermediate）或false。

```
$image_attributes = wp_get_attachment_image_src( $attachment->ID );  
if ( $image_attributes ) : ?>  
    " height="<?php echo $image_attributes[2]; ?>" />  
<?php endif; ?>
```

## 对齐



在您的站点中添加图像时，可以将图像对齐方式指定为右，左，中心或无。WordPress核心自动添加CSS类来对齐图像：

- alignright
- alignleft
- aligncenter
- alignnone

这是当选择中心对齐时的采样输出

```
<img class="aligncenter size-full wp-image-131" src= ... />
```

为了利用这些CSS类用于对齐和文本换行，您的主题必须包含样式表中的样式，如主样式表文件。您可以使用与官方主题捆绑的style.css，如Twenty Seventeen为参考。

## 标题

如果在媒体库中指定了图像图像，则HTML img元素由短代码 `[caption]` 和 `[ / caption]` 括起来。

```
<div class="mceTemp"><dl id="attachment_133" class="wp-caption aligncenter" style="width: 1210px"><dt class="wp-caption-dt"></dt><dd class="wp-caption-dd">Sun set over the sea</dd></dl></div>
```

而且，它将像HTML代码一样呈现为图形标签：

```
<figure id="attachment_133" style="width: 1200px" class="wp-caption aligncenter">
  
  <figcaption class="wp-caption-text">Sun set over the sea</figcaption>
</figure>
```

与排列类似，您的主题必须包含以下样式。

- wp-caption
- wp-caption-text

## 参考文献

---

`wp_get_attachment_image()`

`wp_get_attachment_image_src()`

# Galleries

图片库是在WordPress网站上展示图片的最佳方式。默认情况下，WordPress会在媒体上传器中包含“创建图库”功能，从而可以创建一个简单的图库。

**注意：**添加图库之前，您必须在媒体库中拥有图像。否则，您需要将图像上传到库中，并可以进行图库创建。

## 图库短码

画廊功能允许您使用简单的短码将一个或多个图像画廊添加到您的帖子和页面。

画廊短码的基本形式是：

```
[gallery]
```

**提示：**如果您使用 `[gallery]` 短代码而不使用您的帖子或页面中的ids参数，则只显示“附加到该帖子或页面”的图像。

如果您需要添加ID的多个图像，请使用以下示例短代码

```
``
```

```
[gallery ids="10, 205, 552, 607"]
```

```
//Note: 10, 205, 552 and 607 are the IDs of respected image.
```

```
``
```

**提示：**注意：找到图库的图像的正确ID。转到媒体库，然后点击尊重的图像，ID将显示在URL上。

要使用模板文件中的短代码，请使用do\_shortcode ( ) 函数。将以下代码插入到您的模板文件中：

```
echo do_shortcode( [gallery] );
```

如果需要带有ID的短代码，请在模板文件中插入以下代码：

```
echo do_shortcode( [gallery ids="10, 205, 552, 607"] );
```

## 用法

有可能使用以下语法指定的选项：

```
[gallery option1="value1" option2="value2"]
```

如果要直接在模板文件上打印图库，请使用 `do_shortcode()` 函数，如下所示：

```
<?php echo do_shortcode('[gallery option1="value1"]'); ?>
```

如果您需要过滤短码，以下示例将为您提供一些提示

```
// 注意：'the_content'过滤器用于在从数据库检索并在打印到屏幕之前过滤帖子的内容
<?php $gallery_shortcode = '[gallery id="' . intval( $post->post_parent ) .
'" ]';
print apply_filters( 'the_content', $gallery_shortcode );
?>
```

## 支持的选项

Gallery Shortcodes支持下列基本选项：

## 排序

‘orderby’ 指定缩略图显示的顺序。默认顺序是 “menu\_order” 。

- menu\_order: 您可以在“添加媒体”弹出窗口的“图库”选项卡中重新排列图像
- title: 按照媒体库中图片的标题排序
- post\_date: 按日期/时间排序
- rand: 随机订购
- ID: 指定帖子ID

指定用于显示缩略图的排序顺序; ASC或DESC。例如，按ID和DESC排序：

```
[gallery order="DESC" orderby="ID"]
```

如果需要在模板文件上打印，请使用`do_shortcode()`函数;

```
<?php echo do_shortcode(' [gallery order="DESC" orderby="ID"]'); ?>
```

## 列

“columns” 选项指定库中的列数。默认值为3。如果要增加列数，请使用以下短代码。

```
[gallery columns="4"]
```

如果您需要在模板文件上打印，请使用do\_shortcode()函数：

```
<?php echo do_shortcode(' [gallery columns="4"] '); ?>
```

## IDs

图库短代码上的ID选项加载具有特定帖子ID的图像。

如果要使用特定的帖子ID显示附加的图像，请按照以下代码示例。

```
//注意：删除括号和“gallery”和括号之间的每个空格，“123”`。  
//这里“123”表示帖子ID。 如果要显示多个ID，请使用逗号（，）`分隔ID。  
[ gallery id="123" ]
```

Use ‘do\_shortcode’ function to print the gallery with IDs on template files like below:

```
// 注意：删除括号和“gallery”和括号之间的每个空格，“123”。  
<?php echo do_shortcode(' [ gallery id="123" ] '); ?>
```

## Size

尺寸决定要用于缩略图显示的图像大小。有效值包括 “thumbnail” ， “medium” ， “large” ， “full” 以及使用add\_image\_size()注册的任何其他附加图像大小。默认值为 “thumbnail” 。 “缩略图” ， “中” 和 “大” 图像的大小可以在 “设置” > “媒体” 中的WordPress管理面板中进行配置。

例如，要显示中等大小的图像库：

```
[gallery size="medium"]
```

Gallery的缩写也有一些高级选项。

### itemtag

HTML标签的名称，用于将每个项目包含在库中。默认值为 “dl” 。

### icontag

HTMLtag的名称用于将每个缩略图图标包围在图库中。默认值为 “dt” 。

### captiontag

用于附加每个标题的HTML标签的名称。默认值为“dd”。

您可以更改默认值。

```
[gallery itemtag="div" icontag="span" captiontag="p"]
```

## Link

---

指定要连接图像的位置。默认值链接到附件固定链接。选项：

- file - 直接链接到图像文件
- none - 无链接

例：

```
[gallery link="file"]
```

## Include

---

包含允许您插入逗号分隔的附件ID的“数组”，以仅显示这些附件的图像。

```
[gallery include="23,39,45"]
```

## Exclude

---

排除cal你插入一个“数组”的逗号分隔的附件ID，不显示这些附件的图像。请注意，包含和排除不能一起使用。

```
[gallery exclude="21,32,43"]
```

## References

---

For more technical details take a reference from below links

Gallery Shortcode

Function do\_shortcode()

# Video

## 视频

WordPress视频功能允许您嵌入视频文件，并使用简单的短码 `[video]` 播放。支持的文件类型是mp4，m4v，webm，ogv，wmv和flv。

## Video短码

以下短码显示加载pepper.mp4文件的视频播放器：

```
[video src="pepper.mp4"]
```

要在模板文件中使用短代码，请使用do\_shortcode()函数。如果视频文件存储在您的主题目录中，请使用get\_template\_directory\_uri()或get\_stylesheet\_uri()直接获取文件url，

```
$video_file = get_template_directory_uri() . "/videos/pepper.mp4"; echo do_shortcode(
```

## 循环和自动播放

短码视频与音频具有相同的选项。有关循环和自动播放选项，请参阅相关章节。

以下示例在页面加载和循环后立即开始播放视频。

```
echo do_shortcode('[video mp4=' . $video_file . ' loop = "on" autoplay = 1]');
```

## 初始图像和样式

支持以下基本选项：

### Poster

定义在媒体播放前显示为占位符的图像。

以下相同的代码将存储在（主题目录）/images文件夹中的album\_cover.jpg作为初始图像：

```
echo do_shortcode('[video mp4=' . $video_file . ' poster = ' . get_template_directory_uri() . '/images/album_cover.jpg]');
```

## Height

定义媒体的高度。文件上传时自动检测到值。当您省略此选项时，将使用媒体文件高度。

## Width

定义媒体的宽度。文件上传时自动检测到值。当您省略此选项时，将使用媒体文件宽度。

**提示：**主题的`content_width`设置最大宽度。

以下示例将加载320像素宽和240像素高的音频播放器：

```
echo do_shortcode('[video mp4=' . $video_file . ' width = 320 height = 240]');
```

## Styling

如果要从样式表中更改视频播放器的外观，可以将类别名称定位为“wp-video-shortcode”。如果要显示320 x 240尺寸的上述音频播放器，请将以下代码插入到样式表中。

```
.wp-video-shortcode {  
    width: 320px;  
    height: 240px;  
}
```

## 支持视频格式

- mp4
- m4v
- webm
- ogv
- wmv
- flv

## 参考文献

有关更多技术细节，如启用此功能的内部库，请参阅

<https://make.wordpress.org/core/2013/04/08/audio-video-support-in-core/>.

- Video Shortcode
- Function `do_shortcode()`



# 精选图片和缩略图

精选图像（有时也称为Post Thumbnails）是表示单个Post，Page或Custom Post Type的图像。当您创建主题时，您可以通过多种不同的方式，在您的存档页面，标题或上面的一个帖子中输出特色图像。

支持特色图像 # 启用特色图像支持

在“精选图像”界面将出现在“编辑”屏幕上之前，主题必须声明对“特色图像”功能的支持。通过在主题的functions.php文件中放置以下内容来声明支持：

```
add_theme_support( 'post-thumbnails' );
```

注意：要仅针对特定的帖子类型启用特色映像，请参阅add\_theme\_support()

## 设置精选图像

一旦添加了对特色图片的支持，“精选图像”元框将在相应的内容项目的“编辑”屏幕上显示。如果用户无法看到它们，则可以在屏幕选项中启用它。

默认情况下，精选图像元框显示在编辑帖子和编辑页面屏幕的侧栏中。

## 功能参考

- add\_image\_size() – 注册新图片大小...
- set\_post\_thumbnail\_size() – 注册文章缩略图的图片大小...
- has\_post\_thumbnail() – 检查帖子是否附加了图片...
- the\_post\_thumbnail() – 显示帖子缩略图...
- get\_the\_post\_thumbnail() – 检索Post Thumbnail ...
- get\_post\_thumbnail\_id() – Retrieve Post Thumbnail ID...

## Image Sizes

WordPress的默认图像大小是“Thumbnail”，“Medium”，“Large”和“Full Size”（您上传的图像的原始大小）。这些图像大小可以在WordPress管理媒体面板的“设置”>“媒体”下配置。您还可以通过传递具有图像尺寸的数组来定义自己的图像大小：

```
the_post_thumbnail(); // 没有参数 - > 缩略图
the_post_thumbnail( 'thumbnail' ); // 缩略图（默认最大150px x 150px）
the_post_thumbnail( 'medium' ); // 中等分辨率（最大300px x 300px）
the_post_thumbnail( 'medium_large' ); // 中大分辨率（默认768px x无高度限制）
the_post_thumbnail( 'large' ); // 大分辨率（最大640x x 640像素）
the_post_thumbnail( 'full' ); // 原始图像分辨率（未修改）
```

```
the_post_thumbnail( array( 100, 100 ) ); // 其他分辨率（高，宽）
```

## 添加自定义特色图像大小

除了单独使用定义图像大小外

```
the_post_thumbnail( array( , ) );
```

您可以在主题的功能文件中创建自定义的特色图像大小，然后可以在主题的模板文件中调用。

```
add_image_size( 'category-thumb', 300, 9999 ); // 300 pixels wide (and unlimited height)
```

以下是在主题的functions.php文件中创建自定义精选图像大小的示例。

```
if ( function_exists( 'add_theme_support' ) ) {  
    add_theme_support( 'post-thumbnails' );  
    set_post_thumbnail_size( 150, 150, true ); // default Featured Image dimensions (cropped)  
  
    // additional image sizes  
    // delete the next line if you do not need additional image sizes  
    add_image_size( 'category-thumb', 300, 9999 ); // 300 pixels wide (and unlimited height)  
}
```

## 设置特色图像输出尺寸

用于当前Theme的functions.php文件。

您可以使用set\_post\_thumbnail\_size()通过按比例调整图像大小来设置默认的精选图像大小（即不扭曲图像）：

```
set_post_thumbnail_size( 50, 50 ); // 50 pixels wide by 50 pixels tall, resize mode
```

通过裁剪图像（从侧面或从顶部和底部）设置默认的精选图像尺寸：

```
set_post_thumbnail_size( 50, 50, true ); // 50 pixels wide by 50 pixels tall, crop mode
```

## 精选图片样式

精选图片被赋予一个类“wp-post-image”。他们还根据显示的缩略图的大小获得一个类。您可以使用这些

CSS选择器对输出进行调整：

```
img.wp-post-image
img.attachment-thumbnail
img.attachment-medium
img.attachment-large
img.attachment-full
```

您还可以使用 `the_post_thumbnail()` 中的 `attribute` 参数给精选图片自己的类。用 “alignleft” 类显示精选图像：

```
the_post_thumbnail( 'thumbnail', array( 'class' => 'alignleft' ) );
```

## 默认使用

```
// 检查帖子或页面是否分配有精选图片。
if ( has_post_thumbnail() ) {
    the_post_thumbnail();
}
```

**注意：**要返回特色图像以用于PHP代码而不是显示它，请使用：`get_the_post_thumbnail()`

```
// 检查特色图像，然后将其分配给一个PHP变量供以后使用
if ( has_post_thumbnail() ) {
    $featured_image = get_the_post_thumbnail();
}
```

## 链接发布固定链接或更大的图像

**警报：**不要在同一主题中一起使用这两个示例。

示例1.要在特定循环中将Post Thumbnails链接到Post固定链接，请在Theme的模板文件中使用以下内容：

```
<?php if ( has_post_thumbnail() ) : ?>
    <a href="<?php the_permalink(); ?>" title="<?php the_title_attribute(); ?>" >

    <?php the_post_thumbnail(); ?>
    </a>
<?php endif; ?>
```

示例2.要将您网站上的所有Post Thumbnails链接到Post的固定链接，请将其放在当前的Theme的

functions.php文件中：

```
add_filter( 'post_thumbnail_html', 'my_post_image_html', 10, 3 );

function my_post_image_html( $html, $post_id, $post_image_id ) {

    $html = '<a href="' . get_permalink( $post_id ) . '" title="' . esc_attr( get_the_title( $post_id ) ) . '">' . $html . '</a>';
    return $html;

}
```

此示例链接到 “large” Post Thumbnail图像大小，必须在 “循环” 中使用。

```
<?php
if ( has_post_thumbnail() ) {
    $large_image_url = wp_get_attachment_image_src( get_post_thumbnail_id(), 'large' );
    echo '<a href="' . $large_image_url[0] . '" title="' . the_title_attribute( 'echo=0' ) . '" >';
    the_post_thumbnail( 'thumbnail' );
    echo '</a>';
}
?>
```

## 源文件

```
wp-includes/post-thumbnail-template.php
```

## 外部资源

- 您需要了解有关WordPress 2.9的帖子图像功能的一切
- WordPress 2.9中的post\_thumbnail的终极指南
- WordPress中的新功能2.9：发布缩略图

# 国际化

## 什么是国际化？

国际化是开发您的主题的过程，因此可以很容易地将其翻译成其他语言。国际化通常缩写为i18n（因为i和n之间有18个字母）。

## 为什么国际化很重要？

WordPress在世界各地使用，这是一个很好的主意，准备WordPress主题，以便他们可以很容易地翻译成其他语言。作为开发人员，您可能没有一个轻松的时间为所有用户提供本地化，但是任何翻译人员都可以在不修改源代码的情况下成功本地化主题。

## 如何使您的主题国际化？

要使您的应用程序中的字符串可以翻译，您必须将原始字符串打包到一组特殊功能的调用中。

## Gettext简介

WordPress使用i18n的gettext库和工具。请注意，如果您在线查看，您将看到 `_()` 函数，它引用了本机PHP gettext兼容的翻译函数，而是使用WordPress，您应该使用 `_()` WordPress定义的PHP函数。如果您想获得更广泛和更深入的gettext视图，我们建议您阅读gettext在线手册。

## 文字域

您需要使用文本域来表示属于该主题的所有文本。文本域是唯一的标识符，可以确保WordPress能够区分所有加载的翻译。这增加了可移植性，并且已经存在的WordPress工具更好。

文本域必须匹配主题的块。如果您的主题名称My Theme在style.css中定义，或者它包含在名为my-theme的文件夹中，则域名应该是my-theme。文本域名必须使用破折号而不是下划线并且小写。

字符串示例：

```
_('text', 'text-domain');
```

```
_('String or text to be internationalized', 'text-domain');
```

将“文本域”更改为主题。

文本域也需要添加到style.css标题中。即使主题被禁用，WordPress也会使用它来国际化您的主题元数据。文本域应与加载文本域时使用的文本域相同。

标题示例：

```
/*
 * Theme Name: My Theme
 * Author: Theme Author
 * Text Domain: my-theme
 */
```

## 域路径

域路径被使用，所以当主题被禁用时，WordPress知道在哪里找到翻译。

仅当翻译位于语言文件夹中被命名为不同于语言的语言文件夹时才有用。例如，如果.mo文件位于languages文件夹中，则Domain Path将是/languages，必须用第一个斜杠写入。默认为主题中的languages文件夹。

标题示例：

```
/*
 * Theme Name: My Theme
 * Author: Theme Author
 * Text Domain: my-theme
 * Domain Path: /languages
 */
```

## 基本字符串

最常用的函数是 `__()`。它返回其参数的翻译：

```
__( 'Blog Options', 'my-theme' );
```

另一个简单的是 `_e()`，它输出其参数的翻译。而不是写：

```
echo __( 'WordPress is the best!', 'my-theme' );
```

你可以使用较短的：

```
_e( 'WordPress is the best!', 'my-theme' );
```

## 变量

如果您使用字符串中的变量，类似于下面的示例，您需要使用占位符。

```
echo 'Your city is $city.'
```

使用printf系列函数。特别有用的是printf和sprintf。例如：

```
printf(
__ ( 'Your city is %s.', 'my-theme' ),
$city
);
```

请注意，用于翻译的字符串是“您的城市是%s”的模板，源和运行时都是相同的。

如果字符串中有多个占位符，建议您使用参数交换。在这种情况下，单引号（'）是必需的：双引号（"）告诉PHP将\$s解释为s变量，这不是我们想要的。

```
printf(
__ ( 'Your city is %1$s, and your zip code is %2$s.', 'my-theme' ),
$city,
$zipcode
);
```

这里的邮政编码显示在城市名称之后。在某些语言中，以相反的顺序显示邮政编码和城市是比较合适的。使用%s前缀，如上例所示，允许这样做。翻译可以写成：

```
printf(
__ ( 'Your zip code is %2$s, and your city is %1$s.', 'my-theme' ),
$city,
$zipcode
);
```

以下示例告诉您什么不做

警告：这是不正确的。

```
// 这不正确不要使用。
__e( 'Your city is $city.', 'my-theme' );
```

用于翻译的字符串从源中提取而不执行与之关联的PHP。例如：变量\$ city可能是温哥华，所以当模板运行时，你的字符串将会显示为“你的城市是温哥华”，但是gettext不会提前知道PHP变量里面是什么。

由于当您的字符串被翻译时变量的值是未知的，所以需要翻译器知道变量\$ country的每个案例。这不是理想的，最好删除动态内容，并允许翻译者专注于静态内容。

## 基本多元化

如果您的项目数量更改时有一个字符串更改。在英语中你有“一个评论”和“两个意见”。在其他语言中，您可以有多个复数形式。要在WordPress中处理这个问题，可以使用 `_n()` 函数。

```
printf(
    _n(
        'One comment',
        '%s comments',
        get_comments_number(),
        'my-theme'
    ),
    number_format_i18n( get_comments_number() )
);
```

`_n()` 接受4个参数：

- singular – 字符串的单数形式
- plural – 字符串的复数形式
- count – 对象的数量，这将决定是否应该返回单数或复数形式（有多种语言，有两种以上的形式）
- text domain – 主题的文本域

功能的返回值是正确的翻译形式，对应于给定的计数。

## 稍后进行多元化

您首先使用 `_n_noop()` 或 `_nx_noop()` 设置多个字符串。

```
$comments_plural = _n_noop(
    'One comment.',
    '%s comments.'
);
```

在稍后的代码中，您可以使用 `translate_nooped_plural` 来加载字符串。

```
printf(
    translate_nooped_plural(
        $comments_plural,
        get_comments_number(),
        'my-theme'
    ),
    number_format_i18n( get_comments_number() )
);
```

## 背景消歧



有时一个术语在多个上下文中使用，并且必须以其他语言单独翻译，即使用英文每个语境使用相同的单词。例如，Post可以用作动词“[点击这里发表你的评论](#)”，作为名词“[编辑这篇文章](#)”。在这种情况下，应该使用 `_x()` 或 `_ex` 函数。它类似于 `__()` 和 `_e()`，但它有一个附加参数 - 上下文：

```
_x( 'Post', 'noun', 'my-theme' );  
_x( 'post', 'verb', 'my-theme' );
```

在这两种情况下使用这种方法，我们得到原始版本的字符串注释。然而，翻译者将看到两个用于翻译的注释字符串，每个字符串在不同的上下文中。

以德文版WordPress为例：Post is Beiträge。德语中相应的动词形式是beitragen。

请注意，类似于 `__()`，`_x()` 有一个回声版本：`_ex()`。前面的例子可以写成：

```
_ex( 'Post', 'noun', 'my-theme' );  
_ex( 'post', 'verb', 'my-theme' );
```

使用你觉得增强易读性和易于编码的。

## 描述

您可以在源代码中添加一个澄清的注释，所以翻译人员知道如何翻译一个字符串 `__( 'g:i:s a' )`。它必须以翻译者的语言开头：并且是gettext调用之前的最后一个PHP注释。这是一个例子：

```
/* translators: draft saved date format, see http://php.net/date */  
$saved_date_format = __( 'g:i:s a' );
```

## 换行字符

Gettext不喜欢 `\r`（ASCII码：13）在可翻译的字符串，所以避免它，并使用 `\n` 代替。

## 空字符串

空字符串保留用于内部Gettext使用，您不得尝试将空字符串国际化。它没有任何意义，因为翻译者不会有上下文。

如果您有一个有效的用例来使一个空字符串国际化，请添加上下文以帮助翻译人员，并与Gettext系统保持一致。

## 处理JavaScript文件

使用 `wp_localize_script()` 将已翻译的字符串或其他服务器端数据添加到先前排入的脚本。

## 逃避字符串

逃避所有的字符串是很好的，防止翻译者运行恶意代码。有几个与国际化功能相结合的逃生功能。

```
<a title="<?php esc_attr_e( 'Skip to content', 'my-theme' ); ?>" class="screen-  
reader-text skip-link" href="#content" >  
    <?php _e( 'Skip to content', 'my-theme' ); ?>  
</a>
```

```
<label for="nav-menu">  
    <?php esc_html_e( 'Select Menu:', 'my-theme' ); ?>  
</label>
```

## 基本功能

---

- `__()`
- `_e()`
- `_x()`
- `_ex()`
- `_n()`
- `_nx()`
- `_n_noop()`
- `_nx_noop()`
- `translate_nooped_plural()`

## 翻译和退出功能

---

必须转义需要翻译并在html标签的属性中使用的字符串。

- `esc_html__()`
- `esc_html_e()`
- `esc_html_x()`
- `esc_attr__()`
- `esc_attr_e()`
- `esc_attr_x()`

## 日期和数字功能

---

- `number_format_i18n()`
- `date_i18n()`

## 写字符串的最佳做法

---

以下是编写字符串的最佳做法

- 使用体面的英式风格 - 尽量减少俚语和缩写。
- 使用整个句子 - 在大多数语言中，单词顺序与英语不同。
- 拆分段落 - 合并相关句子，但不要在一个字符串中包含整个文本页面。
- 不要将前导或尾随的空格留在可翻译的短语中。
- 假设翻译时字符串的长度可以翻倍。
- 避免不正常的标记和不寻常的控制字符 - 不要包含文本周围的标签。
- 不要将不必要的HTML标记放入已翻译的字符串中。
- 不要留下翻译的网址，除非他们可以使用其他语言的版本。
- 将变量作为占位符添加到字符串中，如在某些语言中，占位符更改位置。

```
printf(
__( 'Search results for: %s', 'my-theme' ),
get_search_query()
);
```

使用格式字符串而不是字符串连接 - 翻译短语而不是单词 -

```
printf(
__( 'Your city is %1$s, and your zip code is %2$s.', 'my-theme' ),
$city,
$zipcode
);
```

总是比好

```
__( 'Your city is ', 'my-theme' ) . $city . __( ', and your zip code is ', 'my-
theme' ) . $zipcode;
```

尝试使用相同的单词和符号来防止翻译多个相似的字符串（例如，请勿执行以下操作）

```
__( 'Posts:', 'my-theme' ); and __( 'Posts', 'my-theme' );
```

## 将文本域添加到字符串

您必须将您的Text域作为参数添加到每个 `__()`，`_e()` 和 `_n()` gettext调用中，否则您的翻译将无法正常工作。

例子：

```
__( 'Post' )
```

应该成为

```
__( 'Post', 'my-theme' )
```

```
_e( 'Post' )
```

应该成为

```
_e( 'Post', 'my-theme' )
```

```
_n( 'One post', '%s posts', $count )
```

应该成为

```
_n( 'One post', '%s posts', $count, 'my-theme' )
```

## 加载文本域

您需要使用主题翻译加载MO文件。您可以通过调用load\_theme\_textdomain ( ) 函数加载它们。此调用从您的主题基础目录或{text-domain} - {locale} .mo从WordPress主题语言文件夹中加载{locale} .mo。

语言环境是您在文件wp-config.php中的常量WPLANG中定义的语言代码和/或国家/地区代码。例如，德语的语言环境是de\_DE。所以代码需要wp-config.php将被定义 ( 'WPLANG', 'de\_DE' ); 有关语言和国家/地区代码的更多信息，请参阅您的语言中的WordPress。

小心

将MO文件命名为{locale} .mo ( 例如de\_DE.po&de\_DE.mo ) ，如果将翻译添加到主题文件夹。

将您的MO文件命名为{text-domain} - {locale} .mo ( 例如my-theme-de\_DE.po&my-theme-de\_DE.mo ) ，如果要添加翻译到WordPress主题语言文件夹。

例：

```
function my_theme_load_theme_textdomain() {  
    load_theme_textdomain( 'my-theme', get_template_directory() . '/languages' );  
}  
add_action( 'after_setup_theme', 'my_theme_load_theme_textdomain' );
```

这个功能最好在主题'function.php中运行

注意：WordPress 4.6发布后，翻译现在以[translate.wordpress.org](https://translate.wordpress.org)为优先，因此通过[translate.wordpress.org](https://translate.wordpress.org)翻译的主题不再需要`load_theme_textdomain()`了。但是，没有任何伤害离开这条线。

## 资源

---

- Video: i18n: Preparing Your WordPress Theme for the World
- Video: On Internationalization: Plugins and themes for the whole world Slides
- Video: Big in Japan: A Guide for Themes and Internationalization
- Video: Lost in Translation—i18n and WordPress
- Internationalizing And Localizing Your WordPress Theme
- Internationalization: You’ re probably doing it wrong
- More Internationalization Fun
- Translating custom page template names
- Use `wp_localize_script`, It Is Awesome
- Understanding `_n_noop()`
- Language Packs 101 – Prepwork
- Translating WordPress Plugins and Themes: Don’ t Get Clever
- How to load theme and plugin translations
- Loading WordPress language files the right way

# 本地化

## 什么是本地化？

本地化描述了后续翻译国际化主题的过程。 本地化缩写为l10n（因为l和n之间有10个字母）

## 本地化文件

### POT (Portable Object Template) files

该文件包含主题中的原始字符串（英文）。 这是一个POT文件的例子：

```
#: theme-name.php:123
msgid "Page Title"
msgstr ""
```

第一行是一个注释，通常是字符串的文件和行号。 msgid行是原始字符串，并且msgstr部分是翻译所在的位置。

### PO (Portable Object) files

每个翻译器都使用POT文件，并以自己的语言翻译msgstr部分。 结果是与POT格式相同的PO文件，但具有翻译和语言特定的标题。 每种语言有一个po文件。

### MO (Machine Object) files

从每个翻译的PO文件建立一个MO文件。 这些是gettext函数实际使用的机器可读的二进制文件（它们不关心.POT或.PO文件），是PO文件的“编译”版本。 将po转换为mo文件有几种不同的方法。

## 生成 POT 文件

POT文件是您需要交给翻译人员的文件，以便他们可以做他们的工作。 POT和PO文件可以轻松地重新命名，以便更改文件类型，而不会有任何问题。 提供POT文件以及您的主题是个好主意，因此翻译人员不必专门询问您。 有几种方法可以为主题生成POT文件：

### WordPress i18n tools

您需要从SVN中检出WordPress Trunk目录（请参阅使用Subversion了解SVN）。 您需要检出整个中继线，因为wordpress-i18n工具使用WordPress核心的代码生成POT。 您需要在您的服务器/计算机上安装gettext（GNU国际化实用程序）软件包和PHP才能运行以下命令。

打开命令行并将目录更改为语言文件夹。

```
cd theme-name/languages
```

您可以在命令行中运行makepot.php脚本，如下所示：

```
php path/to/makepot.php wp-theme path/to/your-theme-directory
```

完成之后，您应该在当前目录中看到POT文件。

## Poedit

您也可以在本地使用Poedit进行翻译。这是所有主要操作系统的开源工具。免费的Poedit默认版本支持使用Gettext函数手动扫描所有源代码。它的专业版也具有一键扫描WordPress主题。生成po文件后，您可以将文件重命名为POT。如果生成了一个mo，那么您可以删除该文件，因为它不需要。专业版允许您一键创建po文件。如果您没有专业版本，您可以轻松获取空白POT，并将其用作POT文件的基础。将空白POT放入语言文件夹后，您可以单击Poedit中的“更新”，使用字符串更新POT文件。

## Poedit-Pro

## Grunt Tasks

甚至有一些拙劣的任务可以用来创建POT。grunt-wp-i18n & grunt-pot  
要设置它，您需要安装node.js。这是一个简单的安装。然后，您需要在要使用grunt的目录中安装grunt。这通过命令行完成。可以放在主题根目录中的Grunt.js和package.json的例子。您可以在命令行中使用简单的命令来执行grunt任务。

## WordPress Plugins

您可以在WordPress安装上使用Loco Translate。注意：此插件可能与所有主题不兼容。你的里程会有所不同。

## Translate PO file

翻译PO文件有多种方法。

您可以使用文本编辑器输入翻译。在文本编辑器中，它将如下所示。

```
#: theme-name.php:123  
msgid "Page Title"  
msgstr ""
```

您输入引号之间的翻译。对于德语翻译，它看起来像这样。

```
#: theme-name.php:123  
msgid "Page Title"  
msgstr "Seitentitel"
```

您也可以在翻译时使用Poedit。

第三种选择是使用在线翻译服务。一般的想法是您上传POT文件，然后您可以授权用户或翻译人员翻译您的主题。这允许您跟踪更改，始终具有最新的翻译，并减少翻译两次。

François-Xavier Bérnard正在运行WP-Translations，这是一个“翻译与开发人员会面”的社区。它运行在Transifex上，您可以提交作为开发人员翻译的项目，或翻译用于您的语言的现有插件和主题。这是伟大的，因为那里有现有的翻译。

以下是一些可用于在线翻译PO文件的工具：

- Transifex
- WebTranslateIt
- Poeditor
- Google Translator Toolkit
- GlotPress
- Lingohub

您甚至可以使用WordPress插件进行翻译。

Loco翻译

被翻译的文件将被保存为主题语言文件夹中的{locale} .mo。语言环境是您在文件wp-config.php中的常量WPLANG中定义的语言代码和/或国家/地区代码。例如，德语的语言环境是de\_DE。从上面的代码示例，文本域是'my-theme'，因此德语MO和PO文件应该命名为de\_DE.mo和de\_DE.po。有关语言和国家/地区代码的更多信息，请参阅使用语言安装WordPress。

# Generate MO file

## Command line

程序msgfmt用于创建MO文件。msgfmt是Gettext包的一部分。否则可以使用命令行。典型的msgfmt命令如下所示：

Unix操作系统

```
msgfmt -o filename.mo filename.po
```

Windows操作系统

```
msgfmt -o filename.mo filename.po
```

如果您有很多PO文件一次转换，您可以作为批处理运行它。例如，使用bash命令：

Unix操作系统



```
# Find PO files, process each with msgfmt and rename the result to MO
for file in `find . -name "*.po"` ; do msgfmt -o ${file/.po/.mo} $file ; done
```

Windows操作系统

对于Windows，您需要先安装Cygwin。

创建一个potomo.sh

```
#!/bin/sh
# Find PO files, process each with msgfmt and rename the result to MO
for file in `usr/bin/find . -name '*.po'` ; do /usr/bin/msgfmt -o ${file/.po/.mo} $file ; done
```

您可以在命令行中运行此命令。

```
cd C:/path/to/language/folder/my-theme/languages & C:/cygwin/bin/bash -c /cygdrive/c/path/to/script/directory/potomo.sh
```

## Poedit

msgfmt也集成在Poedit中，允许您使用它来生成MO文件。首选项中有一个设置可以启用或禁用它。

Poedit Prefences MO

### Grunt task

有grunt-po2mo将转换所有的文件。

## 好的翻译提示

### 不要翻译，有机地翻译

双语或多语言你无疑知道你所说的语言有不同的结构，节奏，色调和变化。翻译的消息不需要像英语一样的结构：采取所提出的想法，并提出一种以自然的方式表达目标语言的消息。创建相同消息和等效消息之间的区别是：不要复制，替换。即使在消息中有更多的结构性项目，如果您觉得对目标受众更合乎逻辑或更适合您，您就有创造性的许可来适应和改变。

### 尽量保持相同程度的正式（或非正式）

每个消息具有不同的正式或非正式级别。您的目标语言中每个消息使用的正式或非正式级别都是您自己（或与您的团队）进行比较，但是WordPress消息（特别是信息性消息）往往会有礼貌的非正式 英语口语 尝试在您的文化背景下完成目标语言中的等效项目。

## 不要使用俚语或受众特定的术语

---

在博客中可以预期一些术语，但不要使用只有“在”人群中获得的口语化。如果未经开发的博主以您的语言安装WordPress，他们会知道这个术语是什么意思吗？像pingback，trackback和feed这样的字是这个规则的例外；它们是通常很难翻译的术语，许多翻译选择以英文留下。

## 以您的语言阅读其他软件的本地化

---

如果您遇到困难或需要方向，请尝试阅读其他流行的软件工具的翻译，以了解常用的术语，如何解决方法等等。当然，WordPress有自己的语气和感觉，所以保持 在阅读其他本地化时，请记住，但是请随意挖掘UI术语等来保持与您的语言的其他软件的一致性。

## 使用本地化

---

将本地化文件放在语言文件夹中，主题语言文件夹中或主页语言文件夹中通常在wp-content下的WordPress 3.7。完整的路径将是wp-content / languages / themes / my-theme-fr\_FR.mo。

从WordPress 4.0开始，您可以在“常规设置”中更改语言。如果您没有看到任何选项或要切换到的语言，请执行以下步骤：

将wp-config.php内的WPLANG定义为您选择的语言。例如，如果你想使用法语，你会有：

```
define ( 'WPLANG', 'fr_FR' );
```

- 转到wp-admin / options-general.php或“设置” - > “常规”
- 在“网站语言”下拉列表中选择您的语言
- 转到wp-admin / update-core.php
- 点击“更新翻译”，如果可用
- 核心翻译文件（如有）可以下载

## 资源

---

- [Creating .pot file for your theme or plugin](#)
- [How To Internationalize WordPress Plugins](#)
- [Translating Your Theme](#)
- [Blank WordPress Pot](#)
- [Improved i18n WordPress tools](#)
- [How to update translations quickly](#)
- [Workflow between GitHub/Transifex](#)
- [Gist: Complete Localization Grunt task](#)
- [WordPress.tv](#) tags: i18n, internationalization and translation

# 辅助功能

---

WordPress主题应该生成每个人都可以使用的页面，包括那些看不到或使用鼠标的页面。默认WordPress主题以相当方便的方式生成内容，但作为主题开发人员，您需要在自己的主题中维护这些可访问性标准。虽然网页可访问性可能是一个复杂的问题，但它只归结为四个原则 - 内容必须是：

- 感觉到  
内容必须适用于所有人 - 无论使用什么用户代理或用户感觉不到。
- 可操作  
用户必须能够有效地移动和操作最终的站点，而不管他们是否使用鼠标。
- 可以理解  
内容应以支持理解的方式呈现，包括支持为屏幕阅读器用户构建网站的心理模型。同样地，网站的操作（导航菜单，链接，表单等）应该很容易理解。构建一个包含已知用户行为（如主要内容区域中的下划线链接）的主题有助于此方面。
- 强大的  
在广泛的用户代理中，内容必须同样可用。禁用的用户可以使用一系列硬件和软件解决方案（通常称为“辅助技术”）来允许他们访问网络，包括屏幕阅读和语音识别软件;盲文阅读器和开关（单输入设备）。考虑到这四个原则设计的主题应有助于创建一个可访问的网站。

## 标题

---

这不应该需要说明，但标题不仅仅是大而大胆的文字。它们是将内容分解为逻辑子部分的重要方式，并可能被屏幕阅读器用户依赖。JAWS屏幕阅读器（例如）可以自动创建任何给定页面的标题列表。这允许其用户以类似的方式“扫描”页面内容，使得目标人员可以快速向下滚动页面。

因此，重要的是，标题标签在逻辑上使用，而不是用于任何表示或搜索引擎优化（SEO）效果。包含二十个H1标题的页面可能会产生良好的，理论上的SEO，但它几乎破坏了标题标签的实际使用 - 将复杂的页面分解成子部分。

显然，使用标题标记会因模板而异，但是在构建主题文件时，请尽量保持标题的预期用途。检查您的标题结构的一种方法是使用Firefox的Web Developer工具栏的“信息”菜单下的“查看文档大纲”工具来检查页面。

## 图片

---

如果可能，应使用CSS包含装饰图像。将图像添加到模板标记中的位置，确保它们包含适当的alt属性。主题中的装饰图像可能包括：

- 与标题文字一起使用的横幅或标题图像
- 伴随导航文字链接的图像

主题中的非装饰性图像可能包括：

- 替换标题文字的横幅或标题图片
- 用于代替导航文本的图像
- Alt文字
- 装饰图像 ( null alt )
- 非装饰图像 ( 适当的alt - 至少有1个示例 )

要测试您的模板标记中的图像是信息丰富的还是纯粹的装饰，请使用简单的替代文本决策树来检查图像是否正确使用alt属性。

## 跳过链接

跳过链接提供了一种机制，使用户可以在输入任何给定的页面时直接导航到内容或导航。例如，当生成的页面标记中的内容是最上面的，菜单标记低于页面时，跳过链接可能允许用户“跳到导航”。在主导航菜单在页面标记中最上方的情况下，将使用“跳到内容”链接。

在具有多个菜单和内容区域的设计中，可以使用多个跳过链接 - 例如：

- 跳到主导航栏
- 跳到二级导航
- 跳到页脚

这些链接可能最初使用适当的CSS技术放置在屏幕之外，但应保持屏幕阅读器用户可用，并可在视线键盘导航仪的焦点上可见。

# Links

## ##链接文本

链接文本应描述它链接到的资源 - 即使文本从上下文中读出。一些辅助软件扫描页面以获取链接，并将其作为简单列表呈现给用户。在这些情况下，所有链接都将从上下文中读出。所以链接中使用的文本很重要。不应该将链接用作链接。

避免重复的非上下文文本字符串，如多个“阅读更多”链接。使用像：

```
<?php the_content( the_title(' ', ' ', false) . __( 'Continue reading', 'theme_text_domain' ) ); ?>
```

或者，如果您在the\_content ( ) 之外生成链接，请尝试以下操作：

```
<?php printf( __( '%1$s%2$s%3$s - read more', 'theme_text_domain' ), '<span>', get_the_title(), '</span>' ); ?>
```

请注意，在每种情况下，链接的唯一部分（即帖子标题）正在呈现。这将进一步增强在链接被读取的情况下的可访问性。然而，只要隐藏方法将文本提供给屏幕阅读器（例如使用绝对定位或CSS文本缩进属性）将其移离屏幕，则可以将文章标题隐藏在“阅读更多”链接中。事实上，这种方法可能会对其他用户群体带来积极的好处，通过减少可见的屏幕杂乱。

## 链接突出显示

除Opera之外，大多数现代浏览器中的默认焦点都显得非常无用。这意味着一旦看到的键盘导航器就可以快速地变成“丢失”，因为它们不能很容易地区分哪个链接具有当前焦点。

提供良好：重点和：主动链接突出显示（在导航菜单和其他地方）是一个非常简单的解决方案。在“投资回报率”方面，它极大地提高了视力键盘导航，而您的努力却极少。这也很容易测试。只需将鼠标放在一边，然后尝试挑选主题中的页面。如果你容易迷路，别人也会迷路。尝试复制您当前的：hover造型，看看是否有帮助。

## Link 下划线

一般来说，如果链接在导航菜单之外，应该加下划线。单独使用颜色来区分链接是不够的，因为每个人都不能察觉颜色。下划线链接意味着用户不必“鼠标擦洗”一页或播放“猜测哪个文本是链接”。还要考虑删除下划线作为您的悬停/主动/聚焦造型的一部分，以确保您不依赖于颜色。

## Form 标记

一旦遇到 `<form>` 标签，屏幕阅读器软件就可以自动地从读取模式切换到交互式的表单模式。在这种形式模式中，软件可能不会渲染与表单控件没有明确关联的文本，因此任何使用普通的 `<p> </p>` 标签的文本都可能被忽略。为了提取所有重要信息，屏幕阅读器用户可能需要在表单中进行两次通过，以提取所有视觉信息 - 一次在表单模式中，然后再次以阅读模式。这是一个重要的可及性障碍。

因此，所有形式（包括主题的注释格式）都应确保 `<form> </form>` 标签中的所有内容都通过 `<label>` 标签，其属性和id属性在相关的输入标签内。避免在表单块中使用纯文本（例如inside `<p> </p>` 标签）。到目前为止，在将多个标签与单个表单控件相关联时，没有报告任何问题，因此，如果需要，可以使用“多标签”方法。

## 表格提交

提交后的回复（包括任何错误消息）应始终是可察觉的。如果可能，应在提交后页面的顶部生成错误消息，以使用户立即意识到任何问题。读取上下文时，错误消息也应该是有意义的。

## 单输入表单

仅具有单个输入（例如标准搜索表单）的表单可以将关联的输入标签放置在屏幕之外，以确保标签文本可用于屏幕阅读器用户。

## 标题属性

---

不要依靠title属性来传达信息。 该属性可能不适用于所有用户（例如，语音识别（VR）软件用户）。 如果信息足够重要，可以添加为清晰的文本，以便每个人都可以看到它。

## 可读性

---

有时，您可以做的最简单的事情是创建一个更易读的页面。 从屏幕读取比阅读打印页面要困难得多。 拥挤的文字，大量的图片和太多的信息使得页面很难阅读。 一些通用的设计技巧包括：

- 白色空间是你的朋友。 将其用作减少分心的工具。
- CSS line-height属性可以增加段落文本的可读性。
- CSS letter-spacing属性可以增加标题和较大文本的可读性。
- 将导航元素定位在逻辑位置，并且不允许它们侵入页面的主要内容区域。
- 确保所有字体都足够大，以便可读取。
- 如果您使用自定义字体，请将其嵌入到主题中，这样您就不会依赖第三方网站进行字体传送。 如果您真的需要第三方网站的字体，请检查文本是否仍然可读，如果字体不可用。

## 颜色

---

全频谱色盲模拟大约有10%的互联网用户在看到颜色，特别是那些患有色盲症的患者中有问题。

最常见的色盲形式影响红/绿光谱。 受影响的用户可以将红色，橙色，黄色和绿色感知为单色，其余颜色被感知为蓝色，逐渐变为紫色。 在罕见形式的条件下，蓝色和粉红色可能占主导地位，或者患者可能无法察觉任何颜色 - 所有颜色都减少到灰色阴影。

只要有可能，使用色盲模拟器检查您的设计调色板，避免单独使用颜色来区分重要元素。

## 对比

---

具有视力障碍的游客可能喜欢较高的对比度页面，而阅读困难者可能需要较低的对比度。 尽量达到合理平衡，避免极端。 请记住，当它们包含文本或用于代替文本时，对比度级别也适用于图像。

只要有可能，使用颜色对比工具来查看您的主题中的前景/背景对比度。 您应该达到的最小对比度是4.5:1。 但是，请尽量避免对比度很高，因为这些可能会对使用屏幕放大镜的人造成像素化问题。

## jQuery & JavaScript

---

您仍然可以在可访问的主题中使用jQuery。 只是不要依赖它来做任何主要的功能。 通过浏览主题的输出进行测试，同时在浏览器中禁用JavaScript。 网站是否仍然有效？

在实现jQuery幻灯片等时，请检查它们是否可以由键盘单独导航。

使用ARIA确实在未来的发展方面提供了一些好处，但是再次不应该依赖于当前时间。 还注意到，使用ARIA属性将导致标记验证失败，但使用ARIA比瞄准100%验证更有价值。



在所有与脚本相关的事情中，旨在逐步增强和优雅的退化。

## 验证

验证你的主题的标记和CSS。 验证仍然是确保您的主题页面在整个Web软件范围内显示出最佳优势的最佳方式。大多数人会通过一个网站来描述“作者 - >查看器”的通信，但实际上它实际上是“作者 - >机器（服务器） - >机器（浏览器） - >查看器”。由于其核心是机器机器通信，所以遵循相关规范（尽可能地）最大化有效的通信是有意义的 - 无论使用最终用户代理来查看站点。

也就是说，验证解析器只是那个愚蠢的软件。 例如，ARIA将导致验证失败，但仍然是创建可访问主题的最佳方式。总是用最好的判断力，如有疑问，请问...

## 辅助功能测试工具

这里的一个警告字。 与使用严格二进制“右/错”方法运行的标记和CSS验证器不同，可访问性验证器必须尝试并审核复杂场景。因此，他们的报告可能包括假阳性和/或阴性。再次，不要害怕在这些情况下使用你最好的判断或问问。有很多与辅助功能相关的资源（包括论坛）可以帮助。

## 跨浏览器测试

一切手段，在您首选的浏览器中开发您的主题，但请记住检查其在当前版本的Internet Explorer，Firefox，Chrome，Opera和Safari中的输出。那些使用Mac内置屏幕阅读器VoiceOver的人可能没有选择使用除Safari以外的任何东西。语音识别软件Dragon Naturally Speaking的用户可能会依靠Internet Explorer来有效地浏览网页。由于其众多内置的辅助功能，Opera是残疾人用户中受欢迎的浏览器选择。

在大多数情况下，残疾用户完全有能力优化他们的硬件和软件以满足他们的具体需求。您的工作 - 作为主题开发人员，是确保在一系列现代网络浏览器上查看时，页面内容不会被加扰，隐藏或丢失。显然，并不是所有的浏览器都会以完全相同的方式显示一个给定的页面，但很少有一个可访问性的问题。然而，大多数现代浏览器可以以有效和令人愉快的方式显示有效的页面。不要瞄准像素完美。这样疯狂！相反的目的是在每个测试浏览器中有吸引力，有效，可导航的网站。

## 产生新的Windows或Tabs

产生新的窗口打破浏览器的“后退”按钮，留下一些瞄准的键盘导航员搁浅，没有任何返回原始页面的方法。因此，请避免链接和其他打开新窗口或选项卡的元素。如果真的，真的，必须产生一个新的窗口或选项卡，以明文形式发出警告（最好是链接或控件文本的一部分），以使用户做出明智的选择。

## 自动播放和动画

如果有可能，避免使用动画内容。 1997年，日本电视上的卡通将700多名儿童带到医院，其中包括约500人缉获（引证）。未经用户明确许可，不得播放任何声音。

自动播放适用于：

- 声音 - 为屏幕阅读器用户创造问题，他们可能会发现其软件的输出被网站的音频淹没。
- 动画（Flash或.gif图像） - 可能会在某些情况下触发癫痫发作。
- 幻灯片 - 可能会为屏幕阅读器用户（谁呈现不断变化的内容）和瞄准的键盘导航器（谁可能无法移动通过幻灯片）创建问题。
- 其他特殊效果 - 已知会引起癫痫发作的坠雪等影响。

如果您绝对必须具有（1）自动启动的移动或音频内容，（2）持续超过5秒，并且（3）与其他内容并行呈现，请确保有一个易于操作的机制可暂停，停止，或隐藏它。还要确保视觉动画在任何一秒钟内不会闪烁三次以上。

## Tabindexing

---

在任何情况下，都应避免tabindex属性（在特定情况下除负号tabindex之外）。作为主题开发者，您不是确定任何一个人想要移动到下一个网站的最佳人选。只有用户可以做出决定，所以不要试图劫持他们的浏览器。只要一个页面内的自然标签顺序是合乎逻辑的，并且可以很容易地感知到，大多数用户完全有能力整理自己的导航需求 - 非常感谢。

## 访问键

---

再次 - 理论上的一个好主意 - 但实际上，实施时往往是一个完整的灾难。除了用户没有关于哪些快捷方式在网站上做什么的信息外，使用除数字键之外的任何东西都可能会在用户自己的浏览软件中劫持快捷键。

## 使WordPress可访问

---

使WordPress可访问是WordPress无障碍组的官方博客，致力于改进核心WordPress和相关项目的可访问性。

我们的目标是WordPress核心，主题和插件开发人员提供可访问性建议，反馈和帮助。

任何人都可以加入讨论。您还可以通过电子邮件或通过订阅Feed发表帖子和评论来关注讨论。如果您在更正式的基础上加入我们，我们也会非常高兴。

# 资源

## General

---

- W3C Web Accessibility Initiative ‘How To Meet WCAG2.0’ Quick Reference
- Make WordPress Accessible
- [Accessites.org](#) — general articles on web accessible design.
- Accessify Forum
- Evaluation, Repair, and Transformation Tools for Web Content Accessibility

## 对比度和颜色测试

---

- Vischeck — Online color blindness simulator



- Contrast Analyser for Windows and Mac — free desktop tool with color blindness simulators.
- Sim Daltonism — a color blindness simulator for Mac OS X.
- Alternative Color Contrast Analyzer — provides a warning for high contrasts.

## Toolbars

---

- WAVE — Firefox accessibility evaluation tool.
- Firefox Accessibility Extension — check the use of structural markup in a page.
- Web Developer Toolbar for Firefox — adds various web developer tools to the browser.

# 主题选项 – 自定义API

## 主题选项 - 自定义API

自定义API（定制程序）是用于实时预览任何WordPress更改的框架。它为用户提供统一的界面，可以根据颜色和布局，窗口小部件，菜单等自定义主题及其网站的各个方面。主题和插件都可以为Customizer添加选项。定制程序是为主题添加选项的规范方式。

- 自定义程序出现在WordPress 4.6与Twenty Fifteen 主题。
- 自定义程序出现在WordPress 4.6与Twenty Fifteen 主题。

默认情况下，定制程序选项可以授予具有不同功能的用户，因此，默认情况下，大多数选项仅对管理员可见，其他用户可能会访问某些选项，如果您希望他们能够。定制器的不同部分也可以是上下文，它们是否与用户正在预览的前端上下文相关。例如，核心小部件功能仅显示当前页面上显示的小部件区域;当用户在Customizer预览窗口中导航到包含它们的页面时，将显示其他小部件区域。

本节包含自定义API的概述，包括代码示例和最佳实践的讨论。有关更多详细信息，强烈建议开发人员研究核心定制程序代码（所有包含“自定义”的核心文件）。这被认为是核心代码内的内部文档之外的Customize API的规范官方文档。

# 定制对象

Customize API是面向对象的。 Customizer对象有四种主要类型：面板，部分，设置和控件。 设置将UI元素（控件）与保存在数据库中的设置相关联。 部分是用于控制的UI容器，以改善其组织。 面板是部分的容器，允许将多个部分组合在一起。

每个Customizer对象由PHP类表示，所有对象都由Customize Manager对象WP\_Customize\_Manager管理。 要添加，删除或修改任何Customizer对象，并访问Customizer Manager，请使用customize\_register钩子：

```
function themeslug_customize_register( $wp_customize ) {  
    // Do stuff with $wp_customize, the WP_Customize_Manager object.  
}  
add_action( 'customize_register', 'themeslug_customize_register' );
```

Customizer Manager为每个Customizer对象类型提供add\_，get\_和remove\_方法；每个都有一个id。 get\_方法允许直接修改添加控件时指定的参数。

```
add_action( 'customize_register', 'my_customize_register' );  
function my_customize_register( $wp_customize ) {  
    $wp_customize->add_panel();  
    $wp_customize->get_panel();  
    $wp_customize->remove_panel();  
  
    $wp_customize->add_section();  
    $wp_customize->get_section();  
    $wp_customize->remove_section();  
  
    $wp_customize->add_setting();  
    $wp_customize->get_setting();  
    $wp_customize->remove_setting();  
  
    $wp_customize->add_control();  
    $wp_customize->get_control();  
    $wp_customize->remove_control();  
}
```

**注意：**主题通常不应该使用get方法来修改核心部分和面板，因为主题不应该修改核心的，与主题无关的功能。 鼓励插件在必要时使用这些功能。 主题不应该“重组”主题未添加的自定义程序部分。

## 设置

设置可以处理您的定制器对象的实时预览，保存和清理。 每个设置由控制对象管理。 添加新设置时可以使用几个参数：

```
$wp_customize->add_setting( 'setting_id', array(
    'type' => 'theme_mod', // or 'option'
    'capability' => 'edit_theme_options',
    'theme_supports' => '', // Rarely needed.
    'default' => '',
    'transport' => 'refresh', // or postMessage
    'sanitize_callback' => '',
    'sanitize_js_callback' => '', // Basically to_json.
) );
```

**警报：重要事项：**不要使用看起来像`widget_`，`sidebars_widgets []`，`nav_menu []`或`nav_menu_item []`的设置ID。这些设置ID模式分别保留用于窗口小部件实例，侧边栏，导航菜单和导航菜单项。如果您需要在设置ID中使用“`widget`”，请将其用作后缀而不是前缀，例如“`homepage_widget`”。

有两种主要的设置类型：选项和主题修改。选项直接存储在WordPress数据库的`wp_options`表中，并应用于该站点，而不考虑活动主题。如果添加了选项类型的设置，主题很少。另一方面，主题模式是针对特定主题的。大多数主题选项应该是`theme_mods`。例如，自定义CSS插件可以将自定义主题css设置注册为`theme_mod`，允许每个主题具有独特的一组CSS规则，而不会在切换主题然后切换时丢失CSS。

- `customize-theme-mods-options`
- Theme\_mod与选项设置类型示例。

通常最重要的是设置设置的默认值以及其清理回调，这将确保数据库中不存储不安全的数据。典型主题用法：

```
$wp_customize->add_setting( 'accent_color', array(
    'default' => '#f72525',
    'sanitize_callback' => 'sanitize_hex_color',
) );
```

Typical plugin usage:

```
$wp_customize->add_setting( 'myplugin_options[color]', array(
    'type' => 'option',
    'capability' => 'manage_options',
    'default' => '#ff2525',
    'sanitize_callback' => 'sanitize_hex_color',
) );
```

请注意，定制程序可以使用选项类型处理存储为键控数组的选项以进行设置。这允许将多个设置存储在不是主题模式的单个选项中。要检索并使用您的Customizer选项的值，请使用`get_theme_mod()`和`get_option()`与设置标识：

```
function my_custom_css_output() {
    echo '<style type="text/css" id="custom-theme-css">' .
    get_theme_mod( 'custom_theme_css', '' ) . '</style>';
    echo '<style type="text/css" id="custom-plugin-css">' .
    get_option( 'custom_plugin_css', '' ) . '</style>';
}
add_action( 'wp_head', 'my_custom_css_output');
```

请注意，`get_theme_mod()`和`get_option()`的第二个参数是默认值，它应该与添加设置时设置的默认值相匹配。

## 控制

控件是用于创建UI的主要Customizer对象。具体来说，每个控件必须与设置相关联，并且该设置将用户输入的数据从控件保存到数据库（除了将其显示在实时预览中并将其消毒之外）。控件可以由定制程序管理器添加，并以极少的努力提供一组强大的UI选项：

```
$wp_customize->add_control( 'setting_id', array(
    'type' => 'date',
    'priority' => 10, // Within the section.
    'section' => 'colors', // Required, core or custom.
    'label' => __( 'Date' ),
    'description' => __( 'This is a date control with a red border.' ),
    'input_attrs' => array(
        'class' => 'my-custom-class-for-js',
        'style' => 'border: 1px solid #900',
        'placeholder' => __( 'mm/dd/yyyy' ),
    ),
    'active_callback' => 'is_front_page',
) );
```

添加控件时最重要的参数是它的类型 - 这决定了Customizer将显示哪种类型的UI。Core提供了几种内置控件类型：

`<input>` 任何允许类型的元素（见下文）

- checkbox
- textarea
- radio（将值的数组=>标签传递给选择参数）
- select（将值的数组=>标签传递给choices参数）
- dropdown-pages（使用allow\_addition参数允许用户从控件添加新页面）

对于html input元素支持的任何输入类型，只需在添加控件时将type属性值传递给type参数即可。这允许支持控

件类型，如文本，隐藏，数字，范围，URL，电话，电子邮件，搜索，时间，日期，日期时间和周，等待浏览器支持。

必须将控件添加到部分，然后才能显示（部分必须包含要显示的控件）。这是通过在添加控件时指定section参数来完成的。以下是添加基本textarea控件的示例：

```
$wp_customize->add_control( 'custom_theme_css', array(
    'label' => __( 'Custom Theme CSS' ),
    'type' => 'textarea',
    'section' => 'custom_css',
) );
```

这是一个基本的范围（滑块）控件的例子。请注意，大多数浏览器将不会显示此控件的数值，因为范围输入类型是为确切值不重要的设置而设计的。如果数值很重要，请考虑使用数字类型。input\_attrs参数将映射属性=>值的键控数组到输入元素上的属性，并且可以用于从占位符文本到数据的用途 - 自定义脚本中的JavaScript引用的数据。对于数字和范围控制，它允许我们设置最小值，最大值和步长值。

```
$wp_customize->add_control( 'setting_id', array(
    'type' => 'range',
    'section' => 'title_tagline',
    'label' => __( 'Range' ),
    'description' => __( 'This is the range control description.' ),
    'input_attrs' => array(
        'min' => 0,
        'max' => 10,
        'step' => 2,
    ),
) );
```

## 核心自定义控件

如果没有一个基本的控制类型适合您的需要，您可以轻松地创建和添加自定义控件。自定义控件在本文后面将会更全面地解释，但它们基本上是基于WP\_Customize\_Control对象的子类，允许任何可能需要的html标记和功能。Core提供了几个内置的自定义控件，允许开发人员轻松实现丰富的JavaScript驱动功能。可以添加颜色选择器控件，如下所示：

```
$wp_customize->add_control( new WP_Customize_Color_Control( $wp_customize, 'color_control', array(
    'label' => __( 'Accent Color', 'theme_textdomain' ),
    'section' => 'media',
) ) );
```

MediaPress 4.1和4.2还增加了对任何类型的多媒体内容的支持。媒体控制实现本地WordPress媒体管理器，允

许用户从其库中选择文件或上传新的文件。通过在添加控件时指定mime\_type参数，可以指示媒体库显示为特定类型，如图像或音频：

```
$wp_customize->add_control( new WP_Customize_Media_Control( $wp_customize, 'image_control', array(
    'label' => __( 'Featured Home Page Image', 'theme_textdomain' ),
    'section' => 'media',
    'mime_type' => 'image',
) ) );
```

```
$wp_customize->add_control( new WP_Customize_Media_Control( $wp_customize, 'audio_control', array(
    'label' => __( 'Featured Home Page Recording', 'theme_textdomain' ),
    'section' => 'media',
    'mime_type' => 'audio',
) ) );
```

请注意，与WP\_Customize\_Media\_Control关联的设置保存相关的附件ID，而所有其他媒体相关控件（WP\_Customize\_Upload\_Control的子项）将媒体文件URL保存到该设置。有关Make WordPress Core的更多信息。

此外，WordPress 4.3引入了WP\_Customize\_Cropped\_Image\_Control，它为选择后的图像提供了一个界面。这对于需要特定宽高比的情况很有用。

## 部分

部分是Customizer控件的UI容器。虽然您可以将自定义控件添加到核心部分，但如果您有多个选项，则可能需要添加一个或多个自定义部分。使用WP\_Customize\_Manager对象的add\_section方法添加新的部分：

```
$wp_customize->add_section( 'custom_css', array(
    'title' => __( 'Custom CSS' ),
    'description' => __( 'Add custom CSS here' ),
    'panel' => '', // Not typically needed.
    'priority' => 160,
    'capability' => 'edit_theme_options',
    'theme_supports' => '', // Rarely needed.
) );
```

您只需要包含要覆盖默认值的字段。例如，默认优先级（外观顺序）通常是可接受的，如果您的选项不言自明，大多数部分不应该需要描述性文本。如果您想要更改自定义部分的位置，则核心部分的优先级如下所示：

Title	ID	Priority (Order)

Site Title & Tagline	title_tagline	20
Colors	colors	40
Header Image	header_image	60
Background Image	background_image	80
Menus (Panel)	nav_menus	100
Widgets (Panel)	widgets	110
Static Front Page	static_front_page	120
default		160
Additional CSS	custom_css	200

在大多数情况下，可以添加仅指定一个或两个参数的节。 以下是添加与主题页脚相关的选项部分的示例：

```
// Add a footer/copyright information section.
$swp_customize->add_section( 'footer' , array(
    'title' => __( 'Footer', 'themenname' ),
    'priority' => 105, // Before Widgets.
) );
```

## 面板

Customizer Panels API是在WordPress 4.0中引入的，允许开发人员创建一个超出控件和部分的层次结构。控制面板不仅仅是简单地分组控件，而是设计为定制工具提供不同的上下文，例如自定义窗口小部件，菜单，或者将来可能编辑帖子。部分和面板对象之间存在重要的技术区别。

在大多数情况下，主题不应该注册自己的面板。部分不需要嵌套在面板下，每个部分通常应包含多个控件。控件也应该添加到核心提供的部分，例如为颜色部分添加颜色选项。还要确保您的选择尽可能精简和高效;看到WordPress的哲学。面板设计为整个功能（如小部件，菜单或帖子）的上下文，而不是用于通用部分的包装。如果您绝对必须使用面板，您会发现API与章节几乎完全相同：

```
$swp_customize->add_panel( 'menus', array(
    'title' => __( 'Menus' ),
    'description' => $description, // Include html tags such as <p>.
    'priority' => 160, // Mixed with top-level-section hierarchy.
) );
$swp_customize->add_section( $section_id , array(
    'title' => $menu->name,
    'panel' => 'menus',
) );
```

面板必须包含至少一个必须包含至少一个控件的节，以显示。 如上例所示，Sections可以添加到面板中，类似于将控件添加到Sections。 但是，与控件不同，如果Panel参数在注册Section时空，则它将显示在主要的顶级定



制程序上下文中，因为大多数部分不应包含在面板中。

## 自定义控件，部分和面板

通过对与每个Customizer对象相关联的PHP对象进行子类化，可以轻松创建自定义控件，部分和面板：

WP\_Customize\_Control，WP\_Customize\_Section和WP\_Customize\_Panel（这也可以用于WP\_Customize\_Setting，但自定义设置通常使用自定义设置类型更好地实现，如在下一节概述）。以下是基本自定义控件的示例：

```
class WP_New_Menu_Customize_Control extends WP_Customize_Control {
    public $type = 'new_menu';
    /**
     * Render the control's content.
     */
    public function render_content() {
        ?>
        <button class="button button-primary" id="create-new-menu-submit" tabindex=
"0"><?php _e( 'Create Menu' ); ?></button>
        <?php
    }
}
```

通过对基本控件类进行子类化，您可以使用自定义功能覆盖任何功能，也可以根据需要使用核心功能。要覆盖的最常用的功能是render\_content（），因为它允许您从头开始使用HTML创建自定义UI。应该谨慎使用自定义控件，因为它们可能会引入与周围的核心UI不一致的UI，并导致用户混淆。自定义定制器对象可以类似于添加默认控件，部分和面板的方式添加：

```
$wp_customize->add_control(
    new WP_Customize_Color_Control(
        $wp_customize, // WP_Customize_Manager
        'accent_color', // Setting id
        array( // Args, including any custom ones.
            'label' => __( 'Accent Color' ),
            'section' => 'colors',
        )
    )
);
```

添加控件时传递的参数映射到控件类中的类变量，因此您可以添加和使用自定义对象的特定部分在不同实例之间不同的自定义对象。

在创建自定义控件，部分或面板时，强烈建议引用核心代码，以便完全了解可以覆盖的可用功能。Core还包括每种类型的自定义对象的示例。这可以在wp-includes / class-wp-customize-control.php，wp-includes /

class-wp-customize-section.php和wp-includes / class-wp-customize-panel.php中找到。每个Customizer对象类型还有一个JavaScript API，可以使用自定义对象进行扩展；有关更多详细信息，请参阅Customizer JavaScript API部分。

## 定制界面UI标准

自定义定制程序控件，部分和面板应尽可能匹配核心UI实践。这包括依赖于wp-admin的标准，例如使用.button和.button-primary类。还有一些特定于定制器的标准（从WordPress 4.7开始）：

- 白色背景颜色仅用于指示导航和可操作的项目（如输入）
- 一般的#eee背景颜色提供与白色元素的视觉对比
- 1px #ddd将背景边框和彼此之间的导航元素分开
- 在需要视觉分离的元素之间提供15px的间距
- 导航元素一侧使用4px边框显示悬停或焦点，颜色为 #0073aa
- 定制器文本使用颜色：#555d66，#0073aa用于导航元素上的悬停和焦点状态

## 自定义设置类型

默认情况下，定制程序支持将设置保存为选项或主题修改。但是，可以轻松地覆盖此行为，以手动保存和预览WordPress数据库的wp\_options表外的设置，或应用其他自定义处理。要开始使用，请在添加设置时指定除选项或theme\_mod之外的类型（您几乎可以使用任何字符串）：

```
$wp_customize->add_setting( $nav_menu_setting_id, array(
    'type' => 'nav_menu',
    'default' => $item_ids,
) );
```

当相关控件中的值更改时，该设置将不再保存或预览。现在，您可以使用customize\_update\_ \$ setting-> type和customize\_preview\_ \$ setting->类型操作来实现自定义保存和预览功能。以下是从菜单定制程序项目中保存菜单项的顺序属性的示例（设置的值是菜单ID的有序数组）：

```
function menu_customizer_update_nav_menu( $value, $setting ) {
    $menu_id = str_replace( 'nav_menu_', '', $setting->id );
    // ...
    $i = 0;
    foreach( $value as $item_id ) { // $value is ordered array of item ids.
        menu_customizer_update_menu_item_order( $menu_id, $item_id, $i );
        $i++;
    }
}
add_action( 'customize_update_nav_menu', 'menu_customizer_update_nav_menu', 10,
2 );
```

而且这里是同一个插件实现导航菜单项的预览（请注意，此示例需要PHP 5.3或更高版本）：

```
function menu_customizer_preview_nav_menu( $setting ) {
    $menu_id = str_replace( 'nav_menu_', '', $setting->id );
    add_filter( 'wp_get_nav_menu_items', function( $items, $menu, $args ) use ( $
menu_id, $setting ) {
        $preview_menu_id = $menu->term_id;
        if ( $menu_id == $preview_menu_id ) {
            $new_ids = $setting->post_value();
            foreach ( $new_ids as $item_id ) {
                $item = wp_setup_nav_menu_item( $item );
                $item->menu_order = $i;
                $new_items[] = $item;
                $i++;
            }
            return $new_items;
        } else {
            return $items;
        }
    }, 10, 3 );
}
add_action( 'customize_preview_nav_menu', 'menu_customizer_preview_nav_menu', 10
, 2 );
```

# 改进用户体验的工具

## 上下文控制，部分和面板

WordPress 4.0和4.1还增加了对Customizer UI部分的显示或隐藏的支持，这取决于用户在Customizer预览窗口中预览的部分网站。一个简单的上下文控制示例将是您的主题仅在首页上显示标题图像和站点标语。这是Customizer Manager的get\_方法的完美用例，因为我们可以直接修改这些设置的核心控件，以使其与首页相关：

```
// Hide core sections/controls when they aren't used on the current page.
$wp_customize->get_section( 'header_image' )->active_callback = 'is_front_page';

$wp_customize->get_control( 'blogdescription' )->active_callback = 'is_front_page';
```

在此上下文控制示例中，主题仅在首页上显示站点标语，因此当用户浏览到预览窗口中的其他页面时，定制器中的相应字段将被隐藏。

面板，节和控件的active\_callback参数采用回调函数名称，无论是核心还是自定义。注册对象时，也可以设置此参数。这是二十四主题的例子：

```
$wp_customize->add_section( 'featured_content', array(
    'title'          => __( 'Featured Content', 'twentyfourteen' ),
    'description' => //...
    'priority'       => 130,
    'active_callback' => 'is_front_page',
) );
```

在前面的例子中，直接使用is\_front\_page。但是，对于更复杂的逻辑，例如检查当前视图是否是页面（甚至是特定页面，按id），可以使用自定义函数（有关为什么需要这个信息的详细信息，请参阅# 30251）。如果您不需要支持PHP 5.2，则可以内联：

```
'active_callback' => function () { return is_page(); }
```

PHP 5.2支持与创建命名函数一样简单，并使用active\_callback参数引用它：

```
//...
'active_callback' => 'prefix_return_is_page';
//...
function prefix_return_is_page() {
```

```
return is_page();
}
```

在自定义控件，Sections和面板中，还有一个选项可以直接在自定义Customizer对象类中覆盖active\_callback函数：

```
class WP_Customize_Greeting_Control extends WP_Customize_Control {
    // ...
    function active_callback() {
        return is_front_page();
    }
}
```

最后，有一个过滤器可以用来覆盖所有其他的active\_callback行为：

```
// Hide all controls without a description when previewing single posts.
function title_tagline_control_filter( $active, $control ) {
    if ( '' === $control->description ) {
        $active = is_singular();
    }
    return $active;
}
add_filter( 'customize_control_active', 'title_tagline_control_filter', 10, 2 );
```

请注意，对于所有Customizer对象类型（控件，节和面板），active\_callback API的工作原理相同。作为一个额外的好处，如果其中的所有控件都被上下文隐藏，则部分将自动隐藏，并且面板的功能相同。

## 选择性刷新：快速，准确的更新

在WordPress 4.5中引入，定制程序中的选择性刷新更新“预览”仅刷新相关设置更改的区域。通过仅更新已更改的元素，它比完全iframe刷新更快，更少破坏性。在定制器中选择性刷新中所述的其他一些好处是：

不要重复（DRY）逻辑

准确的预览更新

预览部分与相关设置和控件之间的关联以及WordPress 4.7中的可见编辑快捷方式

纯JavaScript postMessage更新中的逻辑重复。定制程序中的JavaScript必须镜像生成标记的PHP，或者使用快捷方式近似它。但是Selective Refresh是DRY，因为没有JavaScript和PHP的重复。Ajax请求检索预览的新标记。

而且由于这个Ajax调用，刷新是准确的。它使用可以更改标记的过滤器。它显示与前端相同的结果。

另外，选择性刷新部分会提供预览区域与其对应设置之间的关联。定制工具利用这种关系提供可见的编辑快捷方

式，帮助用户找到与其网站特定部分相关联的控件。在将来，部分API可以扩展，以便在预览中直接编辑设置，并且包含一个结构化的JS API，用于预览部分设置。

由于这些原因，强烈建议所有设置利用选择性刷新传输以改善用户体验，并提供额外的基于JavaScript的传输以进一步增强设置预览。

## 注册部分

设置预览需要通过注册必要的部分来选择使用选择性刷新。在这个例子中，很大程度上取自于他们二十六，通过添加相同名称的部分，为博客描述设置添加了选择性刷新。

```
function foo_theme_customize_register( WP_Customize_Manager $wp_customize ) {
    $wp_customize->selective_refresh->add_partial( 'blogdescription', array(
        'selector' => '.site-description',
        'container_inclusive' => false,
        'render_callback' => function() {
            bloginfo( 'description' );
        },
    ) );
}
add_action( 'customize_register', 'foo_theme_customize_register' );
```

如果没有提供设置参数，它默认与部分ID相同，与控件的设置默认为控件ID相同。以下是部分的一些主要参数：

变量	类型	描述
settings	array	设置与部分相关联的ID。
selector	string	指定要更新的页面标记中的元素。
container_inclusive	boolean	如果为真，刷新将替换整个容器。否则，它只会替换容器的孩子。默认为false.
render_callback	function	生成要刷新的标记.
fallback_refresh	bool	是否在文档中找不到部分文件时是否应进行全页刷新.

## 选择性刷新JavaScript事件

这些火wp.customize.selectiveRefresh：

部分内容呈现

展示位置时。如上所述，JavaScript驱动的小部件可以重新构建在此事件上。

## 渲染部分响应

数据返回后，请求部分呈现。 服务器使用'customize\_render\_partials\_response'过滤这些数据。

## 部分内容移动

当窗口小部件在其侧边栏中移动时。 如上所示，JavaScript驱动的小部件可以刷新此事件。

## 小部件更新

当使用其renderContent方法刷新WidgetPartial时。

## 侧边栏更新

当边栏具有刷新或更新的小部件时。 或者当侧边栏的小部件被排序时，使用reflowWidgets（ ）。

## 小部件：选择性选择性刷新

主题和小部件都需要选择使用选择性刷新。 所有核心小部件和主题已经启用。

## 侧栏中的主题支持

为了允许部分刷新主题的边栏中的小部件：

```
add_theme_support( 'customize-selective-refresh-widgets' );
```

## 小部件支持

即使主题支持选择性刷新，小部件也必须选择加入。 所有核心小部件已经启用它。 这是一个添加对Selective Refresh的支持的示例窗口小部件：

```
class Foo_Widget extends WP_Widget {

    public function __construct() {
        parent::__construct(
            'foo',
            __( 'Example', 'bar-plugin' ),
            array(
                'description' => __( 'An example widget', 'bar-plugin' ),
                'customize_selective_refresh' => true,
            )
        );

        if ( is_active_widget( false, false, $this->id_base ) || is_customize_preview() ) {
            add_action( 'wp_enqueue_scripts', array( $this, 'enqueue_scripts' ) );
        }
    }

    ...
}
```



上面的第9行启用了选择性刷新：

```
'customize_selective_refresh' => true,
```

上面的第13行确保小部件的样式表始终显示在Customizer会话中。添加小部件不会导致全页刷新来检索样式：

```
if ( is_active_widget( false, false, $this->id_base ) || is_customize_preview() ) {
    add_action( 'wp_enqueue_scripts', array( $this, 'enqueue_scripts' ) );
}
```



请参阅实现小部件的选择性刷新支持。

## JavaScript驱动的小部件支持

依赖JavaScript进行标记的小部件将需要其他步骤，如实现Widget的选择性刷新支持所示：

按照is\_customize\_preview ( ) 排列任何JavaScript文件，如上图所示。

添加部分内容渲染事件的处理程序，并根据需要刷新窗口小部件：

```
wp.customize.selectiveRefresh.bind( 'partial-content-rendered', function( placement ) {
    // logic to refresh
} );
```

如果小部件包含一个iframe，请添加一个处理程序来刷新部分：

```
wp.customize.selectiveRefresh.bind( 'partial-content-moved', function( placement ) {
    // logic to refresh, perhaps conditionally
}
```

## 使用PostMessage改进设置预览

定制程序会自动处理即时预览所有设置。这是通过静默地重新加载整个预览窗口来完成的，在Ajax调用中PHP被过滤。虽然这样做很好，但它可能非常慢，因为整个前端必须重新加载，以进行每一个设置更改。选择性刷新通过仅刷新已更改的元素来改善此体验，但由于Ajax调用，在预览中看到更改仍然有延迟。

为了进一步改善用户体验，Customizer提供了一个API，用于直接在JavaScript中管理设置更改，从而实现真实的预览。下图显示了使用此技术（称为postMessage）与标准刷新选项的自定义CSS选项的比较：

使用postMessage设置传输的Customizer中的自定义CSS设置。



Customizer中的自定义CSS设置，具有默认的刷新设置传输。

要使用postMessage，请在添加设置时先将传输参数设置为postMessage。许多主题还可以通过修改这些设置的传输属性来修改核心设置，如标题和标语来利用postMessage。

```
$wp_customize->get_setting( 'blogname' )->transport = 'postMessage';
$wp_customize->get_setting( 'blogdescription' )->transport = 'postMessage';
```

一旦设置的传输设置为postMessage，设置将不再触发刷新的预览，当其值更改。要实现JavaScript以在前端预览中更新设置，请首先创建并排入JavaScript文件：

```
function my_preview_js() {
    wp_enqueue_script( 'custom_css_preview', 'path/to/file.js', array( 'customize-preview', 'jquery' ) );
}
add_action( 'customize_preview_init', 'my_preview_js' );
```

您的JavaScript文件应如下所示：

```
( function( $ ) {
    wp.customize( 'setting_id', function( value ) {
        value.bind( function( to ) {
            $( '#custom-theme-css' ).html( to );
        } );
    } );
    wp.customize( 'custom_plugin_css', function( value ) {
        value.bind( function( to ) {
            $( '#custom-plugin-css' ).html( to );
        } );
    } );
} )( jQuery );
```

请注意，您不一定需要使用JavaScript来使用postMessage - 大多数代码是样板。从PostMessage传输中受益最大的设置类型需要简单的JS更改，例如使用jQuery的.html ( ) 或.text ( ) 方法，或者在 <body> 或另一个元素上交换一个类，以触发不同的 CSS规则。做到这一点，或者通过选择性刷新更新完全准确的更改来简化即时预览逻辑，用户体验可以快速，而不会重复JS中的所有PHP逻辑。

## 设置验证

WordPress 4.6包括与Customizer设置值的验证相关的新API。定制程序自介绍以来已经对设置值进行了消毒。消毒涉及强制将某个值安全地保留到数据库中：常见的示例是将值转换为整数或从一些文本输入中剥离标签。因此，消毒是一种有损运作。随着设置验证的增加：

- 所有修改的设置在保存之前先验证。
- 如果任何设置无效，则定制程序保存请求被拒绝：保存因此变为事务性，所有设置为脏，以便再次保存。（定制器交易建议与设置验证密切相关。）
- 验证错误消息显示给用户，提示他们修复错误，然后重试。
- 消毒和验证也分别通过WP\_REST\_Request :: sanitize\_params ( ) 和WP\_REST\_Request :: validate\_params ( ) 作为REST API基础结构的一部分。设定值通过验证后进行清洁。

有关验证行为的更多信息和其他代码示例，请参阅功能公告帖子。

## 验证PHP中的设置

正如您在注册设置时可以提供sanitize\_callback一样，还可以提供validate\_callback arg：

```
$wp_customize->add_setting( 'established_year', array(
    'sanitize_callback' => 'absint',
    'validate_callback' => 'validate_established_year'
) );
function validate_established_year( $validity, $value ) {
    $value = intval( $value );
    if ( empty( $value ) || ! is_numeric( $value ) ) {
        $validity->add( 'required', __( 'You must supply a valid year.' ) );
    } elseif ( $value < 1900 ) {
        $validity->add( 'year_too_small', __( 'Year is too old.' ) );
    } elseif ( $value > gmdate( 'Y' ) ) {
        $validity->add( 'year_too_big', __( 'Year is too new.' ) );
    }
    return $validity;
}
```

正如提供一个sanitize\_callback arg，为customize\_sanitize \_ { \$ setting\_id }添加了一个过滤器，所以提供一个validate\_callback arg也会为customize\_validate \_ { \$ setting\_id }添加一个过滤器。假设WP\_Customize\_Setting实例在其验证方法中对这些实例应用过滤器，如果需要为先前添加的设置添加验证，则可以添加此过滤器。

validate\_callback和任何customize\_validate \_ { \$ setting\_id }过滤器回调采用WP\_Error实例是其第一个参数（最初没有添加任何错误），其次是\$value被清理，最后是正在验证的WP\_Customize\_Setting实例。自定义设置类也可以直接覆盖设置类的validate方法。

## 客户端验证

如果您有一个纯粹通过JavaScript进行预览的设置（并且postMessage传输没有选择性刷新），那么您还应该添加客户端验证。否则，任何验证错误将持续，直到完全刷新发生或尝试保存。客户端验证不能取代服务器端验证，因为如果相应的服务器端验证不到位，恶意用户可能会绕过客户端验证来保存无效值。

在wp.customize.Setting JS类（实际上是wp.customize.Value基类）中有一个可用的验证方法。它的名字有点误导，因为它实际上与WP\_Customize\_Setting :: sanitize ( ) PHP方法类似，但它可以用于在JS中对某个值进行清理和验证。请注意，此JS在“定制器”窗格的上下文中运行而不是预览，因此任何此类JS应具有自定义控件作为依赖关系（而不是自定义预览），并在customize\_controls\_enqueue\_scripts操作期间排入队列。一些例子JS验证：

```
wp.customize( 'established_year', function ( setting ) {
    setting.validate = function ( value ) {
        var code, notification;
        var year = parseInt( value, 10 );

        code = 'required';
        if ( isNaN( year ) ) {
            notification = new wp.customize.Notification( code, {message: myPlugin.l10n.yearRequired} );
            setting.notifications.add( code, notification );
        } else {
            setting.notifications.remove( code );
        }

        code = 'year_too_small';
        if ( year < 1900 ) {
            notification = new wp.customize.Notification( code, {message: myPlugin.l10n.yearTooSmall} );
            setting.notifications.add( code, notification );
        } else {
            setting.notifications.remove( code );
        }

        code = 'year_too_big';
        if ( year > new Date().getFullYear() ) {
            notification = new wp.customize.Notification( code, {message: myPlugin.l10n.yearTooBig} );
            setting.notifications.add( code, notification );
        } else {
            setting.notifications.remove( code );
        }

        return value;
    };
} );
```

## 通知

错误通知提供用户反馈，通常基于控件设置的值。当设置的验证例程返回WP\_Error实例时，会将错误通知添加到设置的通知集合中。添加到PHP WP\_Error实例的每个错误都表示为一个wp.customize.Notification在

JavaScript中：

WP\_Error的代码可以作为JS中的notification.code使用。

WP\_Error的消息在JS中可用作notification.message。请注意，如果在PHP中添加了给定错误代码的多个消息，它们将被连接到JS中的单个消息中。

WP\_Error的数据在JS中可用作notification.data。这将有助于将额外的错误上下文从服务器传递给客户端。

任何时候从服务器上的验证例程返回WP\_Error，将导致一个wp.customize.Notification被创建，其类型属性为“error”。

当前不支持从PHP设置非错误通知（请参阅# 37281），您还可以按照以下方式向JS添加非错误通知：

```
wp.customize( 'blogname', function( setting ) {
    setting.bind( function( value ) {
        var code = 'long_title';
        if ( value.length > 20 ) {
            setting.notifications.add( code, new wp.customize.Notification(
                code,
                {
                    type: 'warning',
                    message: 'This theme prefers title with max 20 chars.'
                }
            ) );
        } else {
            setting.notifications.remove( code );
        }
    } );
} );
```

您也可以提供“信息”作为通知的类型。默认类型为“error”。也可以提供自定义类型，并且可以使用CSS选择器匹配notice.notice-foo来设置通知，其中“foo”是提供的类型。控件也可以覆盖通过覆盖wp.customize.Control.renderNotifications方法来呈现通知的默认行为。

# 定制JavaScript API

在WordPress 4.1中，为所有定制程序对象引入了新扩展的JavaScript API。整个JavaScript API目前位于单个文件wp-admin / js / customize-controls.js中，其中包含所有对象，核心自定义控件等的模型。

## 预览JS和控件JS

定制应用程序当前分为两个不同的区域：自定义程序控件“窗格”和自定义预览。该预览当前在一个iframe中，这意味着所有的JS都可以在控制窗格或预览中运行。postMessage API用于在预览和控件之间进行通信。

大多数主题只能在自定义预览中实现JavaScript，并通过postMessage实现即时预览设置。但是，控件方面的JS可以用于许多事情，例如基于其他设置的值动态显示和隐藏控件，更改预览的URL，重点部分预览等。这里是一个与预览功能交互的控件端JS核心的示例，在这种情况下，当帖子页面更改时更改预览的URL：

```
// Change the previewed URL to the selected page when changing the page_for_posts.
wp.customize( 'page_for_posts', function( setting ) {
    setting.bind( function( pageId ) {
        pageId = parseInt( pageId, 10 );
        if ( pageId > 0 ) {
            api.previewer.previewUrl.set( api.settings.url.home + '?page_id=' +
pageId );
        }
    });
});
```

可以使用相似的逻辑来根据设置的值激活UI对象。二十七主题包括一些有用的例子，用于利用自定义JS API来改善用户体验。请注意，控制面板中有一个名为customize-controls.js的JS文件和一个名为customize-preview.js的自定义预览文件。为了清楚起见，建议所有主题和插件遵循此命名约定，即使只在控件或预览中提供了自定义JS，而不是两者都提供。

本页的其余部分主要用于WordPress 4.1中内置的控件端JS API。

## 控件，部分和面板的模型

和PHP一样，每个Customizer对象类型都有一个JavaScript对应的对象。有wp.customize.Control，wp.customize.Panel和wp.customize.Section模型，以及wp.customize.panel，wp.customize.section和wp.customize.control集合（是的，他们是singular）存储所有控件实例。您可以通过以下方式迭代面板，部分和控件：

```
wp.customize.panel.each( function ( panel ) { /* ... */ } );
wp.customize.section.each( function ( section ) { /* ... */ } );
wp.customize.control.each( function ( control ) { /* ... */ } );
```

## 一起关联控件，部分和面板

在PHP中注册新的控件时，您将传递父节ID：

```
$wp_customize->add_control( 'blogname', array(
    'label' => __( 'Site Title' ),
    'section' => 'title_tagline',
) );
```

在JavaScript API中，可以可预测地获得控件的部分：

```
id = wp.customize.control( 'blogname' ).section(); // returns title_tagline by default
```

要从ID获取节对象，请按照正常的方式查找该部分：wp.customize.section ( id )。

您也可以使用此部分状态将控件移动到另一部分，此处将其移动到“导航”部分：

```
wp.customize.control( 'blogname' ).section( 'nav' );
```

同样，您可以以相同的方式获取部分的面板ID：

```
id = wp.customize.section( 'sidebar-widgets-sidebar-1' ).panel(); // returns widgets by default
```

你可以去另一种方式，让小孩和小孩的孩子：

```
sections = wp.customize.panel( 'widgets' ).sections();
controls = wp.customize.section( 'title_tagline' ).controls();
```

您可以使用这些将所有控件从一个部分移动到另一个部分：

```
_.each( wp.customize.section( 'title_tagline' ).controls(), function ( control )
{
    control.section( 'nav' );
});
```

## 上下文面板，部分和控件

控件，面板和节实例具有活动状态（一个wp.customize.Value实例）。当活动状态发生变化时，面板，部分和

控件实例调用各自的onChangeActive方法，该方法默认情况下分别向上和向下滑动容器元素，如果为false和true。现在还有activate ( ) 和deactivate ( ) 方法用于操作面板，部分和控件的此活动状态。这些状态的主要目的是显示或隐藏对象，而不必将其完全从定制器中删除。

```
wp.customize.section( 'nav' ).deactivate(); // slide up
wp.customize.section( 'nav' ).activate({ duration: 1000 }); // slide down slowly

wp.customize.section( 'colors' ).deactivate({ duration: 0 }); // hide immediately
wp.customize.section( 'nav' ).deactivate({ completeCallback: function () {
    wp.customize.section( 'colors' ).activate(); // show after nav hides completely
} });
```

请注意，只有在预览刷新或加载其他URL之前，手动更改活动状态才会保留。activate ( ) / deactivate ( ) 方法被设计为遵循新扩展状态的模式。

## 聚焦UI对象

基于面板，部分和控件的expand ( ) / collapse ( ) 方法，这些模型还支持一个focus ( ) 方法，它不仅扩展了所有必需的元素，而且还将目标容器滚动到视图中，并将浏览器 专注于容器中的第一个可重点元素。例如，要扩展“静态前端页面”部分，并将重点放在“首页”的选择下拉列表中：

```
wp.customize.control( 'page_on_front' ).focus()
```

焦点功能用于实现自动对焦：深度链接到定制器中的面板，部分和控件。考虑这些URL：

.../wp-admin/customize.php?autofocus[panel]=widgets

.../wp-admin/customize.php?autofocus[section]=colors

.../wp-admin/customize.php?autofocus[control]=blogname

这在WordPress核心中用于在小部件管理页面上添加链接，以直接链接到定制器中的小部件面板，以及将自定义预览中的可见编辑快捷方式与自定义窗格中的关联控件进行连接。

## 重点

在PHP中注册面板，部分或控件时，可以提供优先级参数。该值存储在每个相应的Panel，Section和Control实例的wp.customize.Value实例中。例如，您可以通过以下方式获取小部件面板的优先级：

```
priority = wp.customize.panel( 'widgets' ).priority(); // returns 110 by default
```

然后，您可以动态地更改优先级，并且Customizer UI将自动重新排列以反映新的优先级：

```
wp.customize.panel( 'widgets' ).priority( 1 ); // move Widgets to the top
```

## 自定义控件，面板和部分

---

在JS中使用自定义Customizer对象时，通常最容易检查WordPress核心中的自定义对象来了解代码结构。请参阅wp-admin / js / customize-controls.js，特别是wp.customize.Panel | Section |控制模型。注意核心代码中的几个示例，特别是在媒体控件中，它们通过对象层次结构建立在彼此的功能上。



# JavaScript / Underscore.js渲染的自定义控件

WordPress 4.1还增加了JavaScript的重载和/或高数量控件的支持。这允许更多的动态行为，特别是与动态添加的控件相关。核心的Color和Media控件目前正在利用此API，而所有核心控件最终将在未来使用它。基于PHP的控制API不会消失，但是在将来，大多数控件可能会使用新的API，因为它为用户和开发人员提供了更快的体验。JS模板化部分和面板的类似API被引入WordPress 4.3; 然而，在Word中，JS中动态创建对象的容易性仍然存在一些差距，见 # 30741。

## 注册控制类型

为了引入一个具有相同类型的多个Customizer控件的模板的概念，我们需要引入一种使用Customize Manager注册控件类型的方法。以前，仅当使用WP\_Customize\_Manager::add\_control()添加了自定义控件时才会遇到自定义控件对象。但检测添加的控件类型以每种类型呈现一个模板将不允许动态创建新的控件，如果没有加载该类型的其他实例。WP\_Customize\_Manager::register\_control\_type()解决这个问题：

```
add_action( 'customize_register', 'prefix_customize_register' );
function prefix_customize_register( $wp_customize ) {
    // Define a custom control class, WP_Customize_Custom_Control.
    // Register the class so that its JS template is available in the Customizer.
    $wp_customize->register_control_type( 'WP_Customize_Custom_Control' );
}
```

所有注册的控件类型都将模板打印到定制器中

```
WP_Customize_Manager::print_control_templates();
```

## 发送PHP控制数据到JavaScript

虽然Customizer控件的数据一直被传递给控件JS模型，并且一直能够被扩展，但在使用JS模板时，更有可能需要发送数据。您需要在PHP中的render\_content()中访问的任何内容都需要导出到JavaScript才能在控件模板中访问。WP\_Customize\_Control默认导出以下控件类变量：

- type
- label
- description
- active (boolean state)

您可以通过覆盖自定义控件子类中的WP\_Customize\_Control::to\_json()来添加特定于您的自定义控件的其他参数。在大多数情况下，您还需要调用父类to\_json()方法，以确保所有核心变量也被导出。以下是核心颜

色控制的一个例子：

```
public function to_json() {
    parent::to_json();
    $this->json['statuses'] = $this->statuses;
    $this->json['defaultValue'] = $this->setting->default;
}
```

## JS /下划线模板

将自定义控件类注册为控件类型并导出任何自定义类变量后，可以创建将呈现控件UI的模板。您将覆盖 `WP_Customize_Control::content_template()`（默认为空）作为 `WP_Customize_Control::render_content()` 的替代。渲染内容仍然被调用，所以请确保在子类中使用空的函数覆盖它。

下划线风格的自定义控件模板与PHP非常相似。随着越来越多的WordPress核心成为JavaScript驱动，这些模板越来越普遍。核心中的一些示例模板代码可以在媒体，修订版本，主题浏览器以及Twenty Fifteen主题中找到，其中使用JS模板来同时保存配色方案数据，并立即预览定制程序中的配色方案更改。了解这些模板如何工作的最好方法是研究核心中的类似代码，因此，这里是一个简单的例子：

```
class WP_Customize_Color_Control extends WP_Customize_Control {
    public $type = 'color';
    // ...
    /**
     * Render a JS template for the content of the color picker control.
     */
    public function content_template() {
        ?>
        <# var defaultValue = '';
        if ( data.defaultValue ) {
            if ( '#' !== data.defaultValue.substring( 0, 1 ) ) {
                defaultValue = '#' + data.defaultValue;
            } else {
                defaultValue = data.defaultValue;
            }
            defaultValue = ' data-default-color=' + defaultValue; // Quotes added automatically.
        } #>
        <label>
            <# if ( data.label ) { #>
                <span class="customize-control-title">{{{ data.label }}}</span>
            <# } #>
            <# if ( data.description ) { #>
                <span class="description customize-control-description">{{{ data.description }}}</span>
            <# } #>
            <div class="customize-control-content">
```

```

        <input class="color-picker-hex" type="text" maxlength="7" placeholder="
<?php esc_attr_e( 'Hex Value' ); ?>" {{ defaultValue }} />
    </div>
</label>
<?php
}
}

```

在上述核心自定义颜色控件的模板中，您可以看到在关闭PHP标签后，我们有一个JS模板。用于评估语句的符号 - 在大多数情况下，这用于条件。我们导出到JS的所有控件实例数据存储在 `data` 对象中，我们可以使用 `double`（转义）或三（未转义）括号表示法`{{}}`打印一个变量。正如我之前所说，获得这种写作控制的最佳方式是阅读现有的例子。 `WP_Customize_Upload_Control`最近更新了，以利用此API，并且与媒体管理器的实现方式完美结合，并从最少量的代码中挤出大量功能。如果您想要一些非常好的做法，请尝试转换一些其他核心控件来使用此API，当然也可以将修补程序提交给核心。

## 放在一起

以下是在自定义定制程序控件子类中使用新API所需的内容的摘要：

使你的`render_content()`函数为空（但它需要存在才能覆盖默认值）。

创建一个新的函数`content_template()`，并将`render_content()`的旧内容放在那里。

通过修改`to_json()`函数（参见`WP_Customize_Color_Control`作为示例），将控件导出到浏览器中的JavaScript（JSON数据）所需的任何自定义类变量。

将PHP从`render_content()`转换为JS模板，使用JS语句来评估和`{{}}`关于要打印的变量。PHP类变量在数据对象中可用;例如，可以使用`{{data.label}}`打印标签。

注册自定义控件类/类型。此关键步骤告诉Customizer打印此控件的模板。这与所添加的所有控件的打印模板不同，因为这些想法是可以从一个模板呈现该控件类型的许多实例，并且任何已注册的控件类型将在将来可用于动态控制创建。只要做一些像`$wp_customize->register_control_type( 'WP_Customize_Color_Control' );`。自定义API中仅PHP的部分仍然完全支持，使用非常好。但是，为了使定制工具更加灵活，在定制程序中切换主题而没有页面加载的长期目标，强烈建议您在可行的情况下为所有自定义定制器对象使用JS / Underscore模板。

# 高级用法

自定义API积极开发; 此页面包含其他更高级的主题。 通过在Slack中搜索 `# core-customize` 频道的档案来查找高级主题的更多讨论。

## 允许非管理员访问定制器

定制器访问由自定义元功能（默认映射到 `edit_theme_options`）控制，默认情况下仅分配给管理员。这允许更广泛地使用定制器广泛的功能访问选项，内置在面板，部分和设置中。此外，这样可以允许非管理员使用自定义程序，例如自定义帖子。此更改是将Customizer的范围扩展到主题以外的重要一步。

```
function allow_users_who_can_edit_posts_to_customize( $caps, $cap, $user_id ) {
    $required_cap = 'edit_posts';
    if ( 'customize' === $cap && user_can( $user_id, $required_cap ) ) {
        $caps = array( $required_cap );
    }
    return $caps;
}
add_filter( 'map_meta_cap', 'allow_users_who_can_edit_posts_to_customize', 10, 3 );
```

请注意，如果要向非管理员用户授予自定义元功能，则目前有必要在管理菜单，管理栏或其他位置手动添加定制程序的链接。

# 主题安全

---

## 介绍

---

WordPress开发团队认真对待安全。由于网络依赖于平台的完整性，安全性是必需的。虽然核心开发人员有一个专注于保护核心平台的专门团队，但作为一名主题开发人员，您非常清楚潜在的可能性在核心之外，可能是易受攻击的。因为WordPress提供了如此多的功能和灵活性，插件和主题是弱点的关键点。

## 发展安全心态

---

在开发主题时，在添加功能时，请考虑安全性。在您开始主题开发工作时，请使用以下原则：

不信任任何数据。不要信任用户输入，第三方API或数据库中的数据，无需验证。保护您的WordPress主题始于确保数据进入和离开您的主题是按预期。始终确保在使用或输出之前验证输入和清理（转义）数据。

依靠WordPress API。许多核心WordPress功能提供验证和消毒数据功能的构建。依靠WordPress提供的功能尽可能。

使您的主题保持最新。随着技术的发展，您的主题的新安全漏洞的潜力也越来越大。保持警惕，维护您的代码，并根据需要更新您的主题。

本章提供了一些关于使用数据验证和消毒/转义技术编写安全主题的背景知识，使用随机数生成安全令牌，以及对常见安全攻击的审查以及如何根据他们加强主题。

## 其他资源

---

WordPress主题编写指南 - WordPress.org博客系列

编写安全主题的指南 - 第1部分：简介

编写安全主题的指南 - 第2部分：验证

编写安全主题的指南 - 第3部分：消毒

编写安全主题的指南 - 第4部分：保护帖子元数据

Sucuri安全博客

# 数据消毒/逃避

## 消毒：保障投入

消毒是清理或过滤您的输入数据的过程。无论数据来自用户还是API或Web服务，当您不知道期望或不想严格的数据验证时，您都可以使用清除信息。

消除数据的最简单方法是使用内置的WordPress功能。

消毒系统的帮助函数提供了一种有效的方式来确保您最终获得安全的数据，并且您需要尽可能少的努力：

- `sanitize_email()`
- `sanitize_file_name()`
- `sanitize_html_class()`
- `sanitize_key()`
- `sanitize_meta()`
- `sanitize_mime_type()`
- `sanitize_option()`
- `sanitize_sql_orderby()`
- `sanitize_text_field()`
- `sanitize_title()`
- `sanitize_title_for_query()`
- `sanitize_title_with_dashes()`
- `sanitize_user()`
- `esc_url_raw()`
- `wp_filter_post_kses()`
- `wp_filter_nohtml_kses()`

**提示：**每当您接受潜在的不安全数据时，重要的是验证或清除它。

## 示例 - 简单的输入字段

假设我们有一个名为title的输入字段。

```
<input id="title" type="text" name="title">
```

您可以使用 `sanitize_text_field()` 函数来清理输入数据：

```
$title = sanitize_text_field( $_POST['title'] );
```

```
update_post_meta( $post->ID, 'title', $title );
```

在幕后，`sanitize_text_field()`执行以下操作：

- 检查无效的UTF-8
- 将单个小于字符（<）转换为实体
- 贴上所有标签
- 删除换行符，标签和额外的空格条字节

**提示：**请记住，请依靠WordPress API及其帮助功能来协助确保您的主题。

## 转义：保证输出

无论何时输出数据，请确保正确地将其退出。

Escaping是通过剥离不需要的数据（如格式不正确的HTML或脚本标签）来保护输出的过程，从而防止这些数据被视为代码。

转义有助于在为最终用户呈现数据之前保护您的数据，并防止XSS（跨站点脚本）攻击。

**注意：**跨站点脚本（XSS）是通常在Web应用程序中发现的一种计算机安全漏洞。XSS使攻击者将客户端脚本注入由其他用户查看的网页。攻击者可能会使用跨站点脚本漏洞绕过诸如同源策略的访问控制。

WordPress有一些帮助功能，您可以用于大多数常见的场景。

- `esc_html()` - 当HTML元素包含显示的数据部分时，可以使用此功能。

```
<?php echo esc_html( $title ); ?>
```

- `esc_url()` - 对所有URL使用此功能，包括HTML元素的src和href属性中的URL。

```

```

- `esc_js()` - 使用此功能进行内联JavaScript。

```
<a href="#" onclick="<?php echo esc_js( $custom_js ); ?>">Click me</a>
```

- `esc_attr()` - 将此功能用于打印到HTML元素属性中的所有其他内容。

```
<ul class="<?php echo esc_attr( $stored_class ); ?>"> </ul>
```

- `esc_textarea()` – 对textarea元素内的文本进行编码。

```
<textarea><?php echo esc_textarea( $text ); ?></textarea>
```

提示：输出转义应尽可能晚。

## 随着本地化逃脱

而不是使用echo输出数据，通常使用WordPress本地化函数，如 `_e()` 或 `__()` 。

这些函数只是将定位函数包含在转义函数中：

```
esc_html_e( 'Hello World', 'text_domain' );
// same as
echo esc_html( __( 'Hello World', 'text_domain' ) );
```

这些帮助函数结合本地化和转义：

- `esc_html__()`
- `esc_html_e()`
- `esc_html_x()`
- `esc_attr__()`
- `esc_attr_e()`
- `esc_attr_x()`

## 自定义转义

在需要以特定方式转义输出的情况下，函数 `wp_kses()`（发音为“kisses”）将派上用场。例如，有些情况下，您希望在输出中显示HTML元素或属性。

此功能确保只有指定的HTML元素，属性和属性值才会在输出中出现，并对HTML实体进行规范化。

```
$allowed_html = [
    'a'          => [
        'href'   => [],
        'title'  => [],
    ],
    'br'         => [],
    'em'         => [],
    'strong'     => [],
];
echo wp_kses( $custom_content, $allowed_html );
```



`wp_kses_post()` 是 `wp_kses` 的包装函数，其中 `$allowed_html` 是一组由帖子内容使用的规则。

```
echo wp_kses_post( $post_content );
```

## 数据库转义

---

执行SQL查询之前，SQL查询中的所有数据都必须进行SQL转义，以防止SQL注入攻击。WordPress提供帮助类来协助转义SQL查询\$wpdb。

## 查询数据

---

转义的SQL查询（在此示例中为\$ sql）可以与以下方法一起使用：

- \$wpdb->get\_row(\$sql)
- \$wpdb->get\_var(\$sql)
- \$wpdb->get\_results(\$sql)
- \$wpdb->get\_col(\$sql)
- \$wpdb->query(\$sql)

## 插入和更新数据

---

- \$wpdb->update()
- \$wpdb->insert()

## 像声明

---

- \$wpdb->prepare

# 数据验证

数据验证是根据具有确定结果的预定义模式（或模式）分析数据的过程：有效或无效。

通常这适用于来自外部来源的数据，例如用户输入和通过API调用Web服务。

数据验证的简单示例：

- 检查所需字段未留空
- 检查输入的电话号码只包含数字和标点符号
- 检查输入的邮政编码是否是有效的邮政编码
- 检查数量字段是否大于0
- 数据验证应尽早进行。这意味着在执行任何操作之前验证数据。

**注意：**验证可以通过使用前端的JavaScript和后端使用PHP来执行。

## 验证数据

至少有三种方式：内置PHP函数，核心WordPress函数和您编写的自定义函数。

## 内置PHP功能

使用许多内置的PHP函数进行基本验证，包括：

- `isset()`和`empty()`用于检查变量是否存在且不为空
- `mb_strlen()`或`strlen()`用于检查字符串是否具有预期的字符数
- `preg_match()`，`strpos()`用于检查其他字符串中某些字符串的出现情况
- `count()`用于检查数组中有多少项
- `in_array()`用于检查数组中是否存在某些内容

## 核心WordPress功能

WordPress提供了许多有用的功能，有助于验证不同类型的数据。 以下是几个例子：

- `is_email()`将验证电子邮件地址是否有效。
- `term_exists()`检查是否存在标签，类别或其他分类术语。
- `username_exists()`检查用户名是否存在。
- `validate_file()`将验证输入的文件路径是否为真实路径（但不是文件是否存在）。

检查条件标签的列表以获得更多类似的功能。

搜索具有以下名称的函数：`*_exists()`，`*_validate()`，并且是`_*`。并非所有这些都是验证功能，但许多都是有幫助的。

## 自定义PHP和JavaScript函数

您可以编写自己的PHP和JavaScript函数，并将它们包含在您的插件中。编写验证函数时，您需要将其命名为一个问题（例如：is\_phone，is\_available，is\_us\_zipcode）。

该函数应该返回一个布尔值（true或false），具体取决于数据是否有效。这将允许使用该功能作为条件。

### 示例1

假设您有一个用户提交的美国邮政编码输入字段。

```
<input id="wporg_zip_code" type="text" maxlength="10" name="wporg_zip_code">
```

文本字段允许最多10个字符的输入，对可以使用的字符类型没有限制。用户可以输入一些有效的东西，如1234567890，或者像eval()一样无效（和邪恶）。

我们输入域中的maxlength属性仅由浏览器强制执行，因此您仍然需要验证服务器上输入的长度。如果不这样做，攻击者可以改变maxlength值。

通过使用验证，我们可以确保我们只接受有效的邮政编码。

首先，您需要编写一个功能来验证美国邮政编码：

```
<?php function is_us_zip_code($zip_code) { // scenario 1: empty if (empty($zip_code)) { return false; } // scenario 2: more than 10 characters if (strlen(trim($zip_code)) > 10) {
    return false;
}

// scenario 3: incorrect format
if (!preg_match('/^\d{5}(\-\d{4})?$/ ', $zip_code)) {
    return false;
}

// passed successfully
return true;
}
```

处理表单时，您的代码应检查wporg\_zip\_code字段，并根据结果执行操作：

```
if ( isset( $_POST['wporg_zip_code'] ) && is_us_zip_code( $_POST['wporg_zip_code'] ) ) {
    // your action
}
```

### 例2

假设您要查询数据库中的某些帖子，并希望让用户对查询结果进行排序。

该示例代码通过使用内置的PHP函数`in_array`将允许的排序键的数组进行比较来检查输入的排序键（存储在“`orderby`”输入参数中）的有效性。这样可以防止用户传递恶意数据并潜在地损害网站。

在对数组检查传入排序键之前，将密钥传递到内置的WordPress功能`sanitize_key`。此功能确保了键是小写（`in_array`执行区分大小写的搜索）。

将“`true`”传递给`in_array`的第三个参数可以进行严格的类型检查，这样就可以使功能不仅可以比较值和值类型。这允许代码确定输入的排序键是字符串，而不是其他数据类型。

```
<?php
$allowed_keys = ['author', 'post_author', 'date', 'post_date'];

$orderby = sanitize_key( $_POST['orderby'] );

if ( in_array( $orderby, $allowed_keys, true ) ) {
    // modify the query to sort by the orderby key
}
```

# 使用随机数

WordPress nonces是一次性使用WordPress生成的安全令牌来帮助保护URL和表单免于滥用。

如果您的主题允许用户提交数据; 无论是在管理还是前端; 可以使用nonces来验证用户是否打算执行操作，并且有助于防止跨站点请求伪造（CSRF）。

一个例子是允许授权用户上传视频的WordPress网站。作为授权用户上传视频是有意的行动并被许可。然而，在CSRF中，黑客可以劫持（伪造）使用授权用户并执行欺诈性提交。

由nonce生成的一次性使用哈希通过验证上传请求是由当前登录的用户完成的，防止这种类型的伪造攻击成功。随机数是唯一的唯一的当前用户的会话，所以如果尝试登录或退出页面上的任何随机变量无效。

## 创建一个随机数

- wp\_nonce\_url() - 向URL添加一个随机数。
- wp\_nonce\_field() - 向表单添加一个随机数。
- wp\_create\_nonce() - 以自定义方式使用随机数; 用于处理AJAX请求。

## 验证随机数

- check\_admin\_referer() - 验证在管理屏幕中URL或表单中传递的随机数。
- check\_ajax\_referer() - 检查随机数（但不是引用），如果检查失败，则默认终止脚本执行。
- wp\_verify\_nonce() - 验证在其他上下文中传递的随机数。

## 示例

在这个例子中，我们有一个基本的提交表单。

### 创建随机数

要使用随机数保护窗体，请使用wp\_nonce\_field()函数创建一个隐藏的随机数字段：

```
<form method="post">
  <!-- some inputs here ... -->
  <?php wp_nonce_field( 'name_of_my_action', 'name_of_nonce_field' ); ?>
</form>
```

### 验证随机数

在我们的示例中，我们首先检查是否设置了nonce字段，因为如果表单尚未提交，我们不希望运行任何内容。如果表单已经提交，我们使用nonce字段值函数。如果nonce验证成功，表单将处理。

```
if (
    ! isset( $_POST[ 'name_of_nonce_field' ] )
    || ! wp_verify_nonce( $_POST[ 'name_of_nonce_field' ], 'name_of_my_action' )
)
```

```
) {  
  
    print 'Sorry, your nonce did not verify.';  
    exit;  
  
} else {  
  
    // process form data  
}
```

在这个例子中，基本的随机过程：

- 使用wp\_nonce\_field()函数生成随机数。
- 随机提交表单提交。
- 使用wp\_verify\_nonce()函数验证有效性的随机数。 如果未验证请求退出并显示错误消息。

# 常见漏洞

安全是一个不断变化的环境，漏洞随着时间的推移而演变。 以下是您应该保护的常见漏洞以及保护您的主题免受剥削的技术的讨论。

## 漏洞类型

### SQL注入

这是什么：

当输入的值没有被正确地消毒时，SQL注入会发生，从而允许输入数据中的任何SQL命令潜在地被执行。 为了防止这种情况，WordPress API是广泛的，提供诸如add\_post\_meta ( )；而不是您需要通过SQL ( INSERT INTO wp\_postmeta ... ) 手动添加后期元数据。

exploits\_of\_a\_mom

xkcd一个妈妈的利用

强化您的主题对SQL注入的第一条规则是：当有WordPress功能时，使用它。

但有时您需要执行复杂的查询，这在API中尚未被考虑。 如果是这种情况，请始终使用\$wpdb函数。 这些专门用于保护您的数据库。

执行SQL查询之前，SQL查询中的所有数据都必须进行SQL转义，以防止SQL注入攻击。 用于SQL转义的最佳功能是\$wpdb->prepare ( )，它支持sprintf ( ) 和类似vsprintf ( ) 的语法。

```
$wpdb->get_var( $wpdb->prepare(
    "SELECT something FROM table WHERE foo = %s and status = %d",
    $name, // an unescaped string (function will do the sanitization for you)
    $status // an untrusted integer (function will do the sanitization for you)
) );
```

### 跨站脚本 ( XSS )

跨站脚本 ( XSS ) 发生在一个恶毒的一方向JavaScript页面注入JavaScript时。

通过转义输出来避免XSS漏洞，剥离不需要的数据。 作为主题的主要责任是输出内容，主题应根据内容的类型，使用正确的功能来转义动态内容。

其中一个转义功能的示例是从用户配置文件中转义URL。

```

```

具有HTML实体的内容可以被清理为仅允许指定的HTML元素。

```
$allowed_html = array(
    'a' => array(
        'href' => array(),
        'title' => array()
    ),
    'br' => array(),
    'em' => array(),
    'strong' => array(),
);

echo wp_kses( $custom_content, $allowed_html );
```

## 跨站点请求伪造 ( CSRF )

跨站点请求伪造或CSRF（发音为sea-surf）是当一个恶意的第三方欺骗用户在他们被认证的Web应用程序中执行不需要的操作时。例如，网络钓鱼电子邮件可能包含一个页面的链接，删除WordPress管理员中的用户帐户。如果您的主题包含任何基于HTML或HTTP的表单提交，请使用随机数来保证用户有意执行操作。

```
<form method="post">
    <!-- some inputs here ... -->
    <?php wp_nonce_field( 'name_of_my_action', 'name_of_nonce_field' ); ?>
</form>
```

## 保持现状

保持潜在的安全漏洞很重要。以下资源提供了良好的起点：

- WordPress Security Whitepaper
- WordPress Security Release
- Open Web Application Security Project (OWASP) Top 10



# 高级主题

---

在本节中，我们将介绍一些高级主题。

我们将从使用Child Theme开始修改主题的各个方面。

我们还将介绍用户界面（UI）最佳实践和JavaScript最佳实践。然后使用主题单元测试执行主题检查并验证主题的标记。

最后，我们将看看确保您的主题适用于Plugin API Hooks。

# 子主题

子主题允许您改变网站外观的小部分，但仍保留主题的外观和功能。要理解子主题如何运作，首先要了解父主题与子主题之间的关系。

## 什么是父主题？

父主题是一个完整的主题，其中包含所有必需的WordPress模板文件和主题的资源。所有主题 - 不包括子主题 - 被认为是父主题。

## 什么是子主题？

如概述所示，子主题继承父主题及其所有功能的外观，可用于对主题的任何部分进行修改。以这种方式，自定义与父主题的文件保持分离。使用子主题可以升级父主题，而不影响您对自己的站点的自定义。

子主题：

- 使您的修改便携和可复制;
- 保持定制与父主题功能分离;
- 允许更新父主题，而不会破坏您的修改;
- 让你利用这些努力和测试放在父母主题中;
- 节省开发时间，因为您没有重新创建轮子; 并且是开始学习主题发展的好方法。

**注意：**如果您进行广泛的自定义 - 超出样式和一些主题文件 - 创建父主题可能比一个较小的主题更好的选择。创建父主题可以避免将来使用已弃用的代码的问题。这需要根据具体情况决定。

# 如何创建一个子主题

## 1. 创建一个子主题文件夹

首先，在您的主题目录中创建一个位于wp-content/themes的新文件夹。

该目录需要一个名称。最好的做法是给予主题与父主题一样的名字，但是将-child附加到最后。例如，如果您正在制作一个twentyfifteen的子主题，那么该目录将被命名为twentyfifteen-child。

## 2. 创建一个样式表：style.css

接下来，您将需要创建一个名为style.css的样式表文件，该文件将包含控制主题外观的所有CSS规则和声明。您的样式表必须包含文件顶部的以下必需的标题注释。这告诉WordPress关于主题的基本信息，包括它是一个具有特定父项的子主题的事实。

```
/*
```

```

Theme Name: Twenty Fifteen Child
Theme URI: http://example.com/twenty-fifteen-child/
Description: Twenty Fifteen Child Theme
Author: John Doe
Author URI: http://example.com
Template: twentyfifteen
Version: 1.0.0
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Tags: light, dark, two-columns, right-sidebar, responsive-layout, accessibility-ready
Text Domain: twenty-fifteen-child
*/

```

需要以下信息：

- Theme Name – 需要您的主题独特
- Template – 父主题目录的名称。我们的例子中的父母主题是“twentyfifteen”主题，所以模板将是twentyfifteen。您可能正在使用不同的主题，因此进行相应调整。

根据需要添加剩余信息。唯一必需的子主题文件是style.css，但是functions.php是正确排列样式所必需的（如下）。

### 3. 入队样式表

最后一步是排列父主题和子主题的主题样式表。

**注意：**在过去，常用的方法是使用@import导入父主题样式表；这不再是推荐的做法，因为它增加了加载样式表所需的时间。此外，父样式表可以包含两次。

- 推荐使用父主题样式表的方法是添加一个 `wp_enqueue_scripts` 动作，并在您的子主题的 `functions.php` 中使用 `wp_enqueue_style()`。

因此，您将需要在您的子主题目录中创建一个 `functions.php`。您的子主题的 `functions.php` 的第一行将是一个开放的PHP标签（`<?php`），之后您可以排列父母和子主题样式表。以下示例函数仅在您的父主题仅使用一个主style.css来保存所有css的情况下才起作用。如果你的孩子主题有多个.css文件（例如ie.css，style.css，main.css），那么你必须确保保持所有的父主题依赖关系。

```

<?php
add_action( 'wp_enqueue_scripts', 'my_theme_enqueue_styles' );
function my_theme_enqueue_styles() {
    wp_enqueue_style( 'parent-style', get_template_directory_uri() . '/style.css' );
}

```

```
}
?>
```

如果你的子主题style.css包含实际的CSS代码（通常是这样），你也需要排队它。将“父样式”设置为依赖关系将确保在其后面加载子主题样式表。包括子主题版本号可确保您也可以为子主题突破缓存。完整的（推荐）示例变为：

```
<?php
add_action( 'wp_enqueue_scripts', 'my_theme_enqueue_styles' );
function my_theme_enqueue_styles() {

    $parent_style = 'parent-style'; // This is 'twentyfifteen-style' for the Tw
    enty Fifteen theme.

    wp_enqueue_style( $parent_style, get_template_directory_uri() . '/style.css'
    );
    wp_enqueue_style( 'child-style',
        get_stylesheet_directory_uri() . '/style.css',
        array( $parent_style ),
        wp_get_theme()->get( 'Version' )
    );
}
```

在父主题注册样式表时，父样式与父主题中使用的\$句柄相同。

## 激活子主题

您的子主题现在可以激活了。登录到您的网站的管理屏幕，然后转到管理屏幕>外观>主题。您应该看到您的子主题已列出并准备激活。（如果您的WordPress安装已启用多站点，则可能需要切换到网络管理屏幕以启用主题（在“网络管理主题屏幕”选项卡中），然后可以切换回到特定于站点的WordPress管理屏幕激活你的子主题。）

**注意：**激活子主题后，您可能需要从外观>菜单和主题选项（包括背景和标题图像）重新保存菜单。

## 添加模板文件

除了functions.php文件（如上所述），您添加到您的子主题的任何文件将覆盖父主题中的相同文件。在大多数情况下，最好从父主题创建要更改的模板文件的副本，然后对所复制的文件进行修改，使父文件保持不变。例如，如果要更改父主题的header.php文件的代码，您可以将该文件复制到您的子主题文件夹并在那里进行自定义。

提示：有几个插件允许您检测正在查找的页面上正在使用的具体模板。

- What The File
- What Template File Am I Viewing?
- Debug Bar

您还可以在父主题中未包含的子主题中包含文件。例如，您可能需要创建一个比您的父主题中更具体的模板，例如特定页面或类别归档的模板（例如，页面3.php将为ID为3的页面加载）。

有关WordPress如何确定要使用哪个模板的更多信息，请参阅“模板层次”页面。

## 使用functions.php

与style.css不同，子主题的functions.php不会覆盖其父对象。而是除了父主题的functions.php之外加载它。（具体来说，它是在父文件之前加载的。）

这样，一个子主题的functions.php提供了一个聪明，无故障的修改父主题功能的方法。假设你想添加一个PHP函数到你的主题。最快的方式是打开它的functions.php文件并把功能放在那里。但这不聪明：下次主题更新时，您的功能将会消失。但是有一种替代方法是聪明的方法：您可以创建一个子主题，在其中添加一个functions.php文件，并将该函数添加到该文件中。该功能也将在那里做同样的工作，其优点是不会受到父主题的未来更新的影响。不要将父主题的functions.php的完整内容复制到child主题的functions.php中。

functions.php的结构很简单：在顶部打开PHP标签，下面是PHP的位。在其中，您可以根据需要放置尽可能多的功能。下面的例子显示了一个基本的functions.php文件，它提供了一个简单的事情：向HTML页面的头元素添加一个favicon链接。

```
<?php // Opening PHP tag - nothing should be before this, not even whitespace

// Custom Function to Include
function my_favicon_link() {
    echo '<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico" />'
    . "\n";
}
add_action( 'wp_head', 'my_favicon_link' );
Tip: The fact that a child theme's functions.php is loaded first means that you
can make the user functions of your theme pluggable—that is, replaceable by a
child theme—by declaring them conditionally.

if ( ! function_exists( 'theme_special_nav' ) ) {
    function theme_special_nav() {
        // Do something.
    }
}
```

这样一来，一个子主题就可以通过简单地声明它来代替父代的PHP函数。

有关您的子主题的functions.php文件中包含的内容的详细信息，请阅读“主题功能”页面。

## 参考或包括其他文件

当您需要包含位于子主题目录结构中的文件时，您将需要使用`get_stylesheet_directory()`。由于`style.css`位于子主题子目录的根目录中，所以`get_stylesheet_directory()`指向您的子主题目录（而不是父主题目录）。要引用父主题目录，可以使用`get_template_directory()`。

下面是一个示例，说明如何在引用存储在子主题目录中的文件时使用`get_stylesheet_directory()`：

```
<?php require_once( get_stylesheet_directory(). '/my_included_file.php' ); ?>
```

同时，此示例使用`get_stylesheet_directory_uri()`显示存储在子主题目录中的/ images文件夹中的图像。

```

```

与返回文件路径的`get_stylesheet_directory()`不同，`get_stylesheet_directory_uri()`返回一个URL，这对前端资源很有用。

## 入队样式和脚本

每个脚本和样式都应该使用自己的功能排队，然后应该包含在一个动作中。有关更多信息，请阅读包含CSS和JavaScript的页面。

WordPress不会自动加载您的孩子主题的样式表在前端。以下是使用`wp_enqueue_scripts()`动作钩子调用排序子主题样式表的函数的示例。

```
<?php
add_action( 'wp_enqueue_scripts', 'my_plugin_add_stylesheet' );
function my_plugin_add_stylesheet() {
    wp_enqueue_style( 'my-style', get_stylesheet_directory_uri() . '/style.css',
        false, '1.0', 'all' );
}
```

## 帖子格式

子主题继承父主题定义的帖子格式。但是当创建子主题时，请注意使用`add_theme_support( 'post-formats' )`会覆盖父主题定义的格式，而不添加它。

## RTL支持

要支持RTL语言，请将rtl.css文件添加到您的子主题中，其中包含：

```
/*
Theme Name: Twenty Fifteen Child
Template: twentyfifteen
*/
```

即使父主题没有rtl.css文件，建议将rtl.css文件添加到您的子主题中。只有在is\_rtl()为true时，WordPress才会自动加载rtl.css文件。

## 国际化

子主题可以通过使用WordPress国际化API来准备翻译成其他语言。有关子主题国际化的特殊考虑。

要使子主题国际化，请按照下列步骤操作：

- 1.添加语言目录。  
例如：twentyfifteen/language/
- 2.添加语言文件。  
您的文件名必须是he\_IL.po&he\_IL.mo（取决于您的语言），不同于plugin-he\_IL.xx的插件文件。
- 3.加载文本域  
在after\_setup\_theme动作期间，在functions.php中使用load\_child\_theme\_textdomain()。在load\_child\_theme\_textdomain()中定义的文本域应该用于翻译子主题中的所有字符串。
- 4.使用GetText函数为字符串添加i18n支持。

### 示例：textdomain

```
<?php
/**
 * Set up My Child Theme's textdomain.
 *
 * Declare textdomain for this child theme.
 * Translations can be added to the /languages/ directory.
 */
function twentyfifteenchild_theme_setup() {
    load_child_theme_textdomain( 'twentyfifteenchild', get_stylesheet_directory(
    ) . '/languages' );
}
add_action( 'after_setup_theme', 'twentyfifteenchild_theme_setup' );
?>
```

此时，子主题中的字符串已准备好进行翻译。为了确保它们被正确地翻译成国际化，每个字符串需要有

twentyfifteen文本域。

示例：gettext函数

这是一个回应短语 “Code is Poetry” 的例子：

```
<?php
esc_html_e( 'Code is Poetry', 'twentyfifteenchild' );
?>
```

在load\_child\_theme\_textdomain()中定义的文本域应该用于翻译子主题中的所有字符串。 如果父类中包含一个模板文件，则应将该文本域从父主题中定义的模板文件更改为子主题定义的文本。



# UI最佳实践

## Logo主页链接

每个页面顶部的标志应将用户发送到您网站的主页。假设你的标志在你的主题目录中，这是如何在header.php模板文件中显示它的。

```
<a href="<?php echo home_url(); ?>"></a>
```

## 描述性锚文本

锚文本是超链接的可见文本。良好的链接文本应该给读者一个点击它将发生的动作的想法。

一个坏的例子：

学习WordPress的最好方法是开始使用它。要下载WordPress，请点击[这里](#)。

一个更好的例子：

下载WordPress并开始使用它。这是最好的学习方式。

## 风格链接与下划线

默认情况下，浏览器强调了链接，让用户知道什么是可点击的。一些设计师使用CSS来关闭超链接的下划线。这导致可用性和可访问性问题，因为它更难以识别来自周围文本的超链接。

## 不同的链接颜色

颜色是另一种视觉提示，文本是可点击的。使用与周围文字不同的颜色进行样式化超链接，使其更易于区分。

超链接是拥有状态的少数HTML功能之一。两个最重要的国家被访问和未访问。

为这两个状态使用不同的颜色可以帮助用户识别他们以前访问的页面。将访问链接中的猜测工作的一个好方法是将它们的颜色比未访问链接的颜色深10%-20%。

链接可以有其他3个状态：

- 鼠标悬停在一个元素上时
- 焦点，类似于悬停，但键盘用户
- 当用户点击链接时激活

由于悬停和焦点具有相似的含义，因此赋予它们相同的风格是有用的。

## 颜色对比度

颜色对比度是指两种颜色之间的差异。海军蓝与黑之间的对比度较低。白色和黑色之间的对比度很高。

Jonathan Snook提供了色彩对比计算器，可以帮助您确定网站设计的对比度。WCAG 2.0在正常文本中需要4.5 : 1的比例才符合AA标准。

## 足够的字体大小

使您的文本容易阅读。通过使文本足够大，您可以增加网站的可用性，并使内容更容易理解。14px是最小的文本应该是。

## 将标签与输入相关联

标签通知用户什么是输入字段。您可以使用标签中的for属性将标签连接到输入。这将允许用户单击标签并将焦点放在输入字段上。

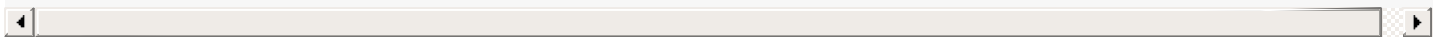
```
<label for="username">Username</label>
<input type="text" id="username" name="login" />
```

标签也可用于单选按钮。由于它使用id字段而不是名称，因此该组的每个输入都将获得自己的标签。

```
<input type="radio" id="user_group_blogger" name="user_group" value="blogger" />
<label for="user_group_blogger">Blogger</label>

<input type="radio" id="user_group_designer" name="user_group" value="designer" />
<label for="user_group_designer">Designer</label>

<input type="radio" id="user_group_developer" name="user_group" value="developer" />
<label for="user_group_developer">Developer</label>
```



## 表单中的占位符文本

占位符文本显示用户要输入的内容的示例。当用户将光标放在字段中时，占位符文本将消失，而标签保留。

```
<label for="name">Name</label>
<input type="text" id="name" name="name" placeholder="John Smith" />
```

## 描述性按钮

网页上填满了不清楚的按钮。请记住，上次您在登录表单上使用“确定”或“提交”？当然你做:)选择更好的单词显示在您的按钮可以使您的网站更容易使用。尝试模式[动词] [名词] - 创建用户，删除文件，更新密码，发送消息。每个描述当用户点击按钮时会发生什么。

# JavaScript最佳做法

许多主题使用JavaScript来提供交互性，动画或其他增强功能。这些最佳做法将有助于确保您的代码有效运行，并且不会导致与您的内容或插件的冲突。

## 使用包含的库

在构建主题时可能需要包含许多有用的JavaScript库。您是否知道WordPress与数十个受欢迎的图书馆捆绑在一起？[查看WordPress附带的默认脚本列表](#)。

开始主题和插件开发人员的一个常见错误是将他们的主题或插件与他们自己的图书馆版本捆绑在一起。这可能会与插入WordPress捆绑的版本的插件冲突。作为最佳做法，使您的主题与WordPress附带的最喜爱的图书版本兼容。

**警告：不要尝试使用自己的版本的已经与WordPress捆绑在一起的JavaScript库。这样做可能会破坏核心功能并与插件冲突。**

如果你觉得你必须用自己的一个WordPress版本替换...好的答案还是：不要这样做。WordPress使用的JavaScript库版本可能包括WordPress需要操作的自定义调整。每当你重写这些库时，都有可能破坏你的WordPress实例。此外，其他作者创建的插件也应该写成与WordPress的这些库的版本兼容。添加您自己的版本可能（并且经常与！）与插件冲突。

## 标准 JavaScript

Javascript正在越来越受到Web开发人员的欢迎，它被用于完成各种任务。以下是编写JavaScript时使用的最佳做法

- 避免使用全局变量
- 保持JavaScript不引人注目
- 使用闭包和模块模式
- 坚持编码风格。使用WordPress Javascript编码标准。
- 验证和删除您的代码 - [JSLint.com](#)
- 确保您的网站仍然无法使用JavaScript，然后添加JavaScript以提供其他功能。这是渐进式增强的一种形式。
- 懒惰的负载资产不是立即需要的。
- 不要使用jQuery，如果你不需要 - 有一个伟大的网站，显示如何做一些常见的任务与纯旧的JavaScript - 你可能不需要jQuery

## #jQuery

## 在您的主题中包含jQuery

Query是一个流行的JavaScript库，可以使浏览器中的JavaScript更轻松，更可靠。如果您使用jQuery，请务必遵循包含JavaScript的手册指南。给你的主题排队的js文件一个array（'jquery'）的依赖数组确保jQuery脚本已经下载并加载到主题的代码之前。

## 使用 \$

因为WordPress中包含的jQuery的副本加载（需要链接）noConflict()模式，请在主题的js文件中使用此包装代码来映射 “\$” 到 “jQuery”：

```
( function( $ ) {  
    // Your code goes here  
} )( jQuery );
```

这个包装器（称为即时调用函数表达式或IIFE）可让您传递一个变量jQuery - 在底线，并在内部给它名称 “\$”。在这个包装器中，您可以使用\$来正常选择元素。

## 选择器

每次使用jQuery选择元素时，它都会通过文档的标记执行查询。这些查询速度非常快，但是它们需要时间，您可以通过重新使用jQuery对象而不是使用新的查询来更快地响应您的站点。所以代替重复选择器：

```
// Anti-pattern  
$('.post img').addClass('theme-image');  
$('.post img').on('click', function() { /* ... */ });
```

建议缓存选择器，以便您可以重新使用返回的元素，而无需重复查找过程：

```
var $postImage = $('.post img');  
// All image tags within posts can now be accessed through $postImage  
$postImage.addClass('theme-image');  
$postImage.on('click', function() { /* ... */ });
```

## 事件处理

当您使用jQuery方法（.bind或.click）时，jQuery会创建一个新的浏览器事件对象来处理请求的事件。创建的每个新事件都占用少量内存，但是所需的内存量会增加绑定的事件数量。如果你有一个页面有一百个锚标签，并且想要在用户点击一个链接时触发一个 “logClick” 事件处理程序，那么编写如下代码是很容易的：

```
// Anti-pattern
```

```
$( 'a' ).click( logClick );
```

这是有效的，但在引擎盖下，您要求jQuery为您网页上的每个链接创建一个新的事件处理程序。

事件委托是一种以更少的开销来实现相同效果的方法。因为jQuery “bubble” 中的事件就是这样，一个点击事件将在一个链接上触发，然后在链接的周围的

标签上，然后在div容器上，依此类推，直到文档对象本身 - 我们可以放一个 单个事件处理程序在页面结构中较高，并且仍然捕获所有这些链接的点击事件：

```
// Bind one handler at the document level, which is triggered  
// whenever there is a "click" event originating from an "a" tag  
$(document).on( 'click', 'a', logClick );
```

# 主题单元测试

---

主题单元测试数据是一个WordPress导入文件将使用足够的stub数据（帖子，媒体，用户）填写WordPress网站来测试主题。

主题单元测试是通过手动测试来测试主题功能，以及主题如何响应内容和设置的边缘。

# 验证你的主题

---

验证您的主题的标记是确保网页符合各种组织定义的网络标准的过程。 这些标准确保不同浏览器，搜索引擎和其他Web客户端以相同的方式解释网页。

符合标准和法规是您可以让您的主题普遍理解的许多方法之一。 确保您的代码和样式全面验证。 这意味着他们必须符合W3C组织规定的标准，并通过各种CSS和XHTML的验证。

并非所有验证器都检查相同的东西。 有些只检查CSS，其他XHTML和其他可访问性。 如果您真诚地向公众提供标准化页面，请用几个验证器进行测试。 万维网联盟设定了标准，并托管了各种网页验证器。

## 验证技巧

---

验证您的WordPress网站意味着不仅仅是检查首页是否有错误。 主题的模板文件以模块化方式加载。 虽然您可能会修复与首页上的index.php和sidebar.php相关联的所有错误，但是在其他模板文件（如single.php，page.php，archives.php或category.php）中可能仍然存在错误。 验证加载每个主题模板文件的页面。

WordPress论坛中的WordPress部分致力于帮助WordPress用户获取有关其网站的反馈。 WordPress志愿者将为您免费提供。 确保并阅读WordPress网站评论指南。

### ##验证清单

为帮助您验证您的WordPress网站，这里是一个快速清单：

- Validate your HTML / CSS / Feeds (see below)
- Validate accessibility – WCAG Guidelines, Section 508 Standards and WAI standards
- Check different browsers
- Re-validate HTML and CSS

## 验证 HTML

---

- The W3C' s HTML Validation Service (URL and upload)
- W3C' s Collection of Validators
- W3C Tidy Online
- Silk Tide Online Validator for Errors and Accessibility
- Windows GUI Interface for TIDY
- Site Report Card Validator
- Valet [Webthing.com](#)
- Watson Addy' s Validator
- Alpine Internet HTML Validator
- AnyBrowser' s HTML Validation
- Cynthia Says Validator

- Doctor-HTML Validator
- [HTMLvalidator.com](#)’ s Validator
- Software QA and Testing Resource Center
- HTML Tag Checker
- [W3.org](#) Tidy Validator
- HTMLHelp’ s File Upload HTML Validator (upload)
- Searchengineworld’ s HTML File Upload Validator (upload)

## 验证 CSS

---

- [W3.org](#)’ s CSS Validator (URL, upload, and direct paste in)
- [Style-Sheets.com](#) Validator (browser specific)
- W3C Style sheet validator
- WDG and [HTMLhelp.com](#)’ s CSS Validator and Checker (URL, upload, and direct paste)

## 验证 Feeds

---

- [Feedvalidator.org](#) – for Atom and RSS feeds.
- Experimental RSS 1.0 Validator
- Redlands RSS 1.0 Validator

## 验证资源和文章

---

- Writing Code in Your Posts
- Mark Freeman’ s Many Links to Validation Resources
- Squarefree’ s Bookmarklets (JavaScript for web page testing)
- [Validator.com](#)
- Understanding How HTML Validators Work
- You Call That Web Site Testing?

## 有关

---

- Accessibility



# Plugin API Hooks

---

一个主题应该适用于WordPress插件。 插件通过使用动作和过滤器来添加功能，这些功能被统称为钩子（有关更多信息，请参阅Plugin API）。

大多数钩子都是由WordPress内部执行的，所以你的主题不需要特殊的标签来使它们工作。 但是，您的主题模板中需要包含几个钩子。 这些钩子被特殊的模板标签：

- `wp_head()` 在主题的header.php模板文件的 `<head>` 元素的末尾。
- `wp_footer()` 在footer.php中，就在关闭 `</body>` 标签之前。
- `wp_meta()` 通常位于主题菜单或侧边栏的 `<li>` 元 `</li>` 部分。
- `comment_form()` 直接在文件的结束标记（ `</div>` ）之前的comments.php中。

看看一个核心主题的模板，了解如何使用这些钩子。

# 发布你的主题

---

本节介绍将主题发布到WordPress主题目录中的要求和提交过程。 如果您已按照本手册中的说明进行操作，则您的主题几乎准备好发布到目录中。

发布主题的第一个要求是确保您至少拥有所需的主题文件，然后再提交主题才能进行审核。

一旦您确认了所需的主题文件，就必须完成对主题（包括内容）的彻底测试。 按照主题审查指导方针，将有助于确保您接受主题。

完成上述之后，提交正确的文档是提交主题以供批准之前的最后一步。 提交主题以供审核后，主题审阅者可能会要求您对主题进行其他更改。

# 所需的主题文件

---

在提交主题进行评审之前，必须确保包含某些文件。这些文件必须遵循主题评审团队设置的模板文件标准。除了下面的文件，还有一些其他标准的模板文件，我们建议您使用，如讨论组织主题文件页。

## 所需的主题文件

---

- style.css

你的主题的主要样式表文件。该文件还将包含您的主题的信息，如作者姓名，版本号和插件URL，在它的头。

- index.php

主模板文件为您的主题。这将是网站主页的模板，除非指定静态页面。如果只包含此模板文件，则必须包含主题的所有功能。但是，您可以在主题中使用尽可能多的相关模板文件。

- comments.php

评论模板，其中包括任何评论是允许的。该文件应提供支持多线程的评论和引用，并应风格不同作者的评论网友评论。查看更多信息的评论页面。

- screenshot.png

WordPress.org主题目录中，screenshot.png作为视觉指示你的主题看起来像什么。在web视图和管理仪表板中都可见。注：这张截图不能大于1200x900px。

虽然这些文件是由验收到WordPress.org主题目录主题审查小组唯一需要的文件，你可以使用其他的模板文件。当然，本手册中提到的任何文件都可以用在你的主题中。

# 测试

---

如果您已经遵循本手册，在将主题提交到WordPress.org主题目录之前，您已经很好地掌握了所需的测试。 如果您没有，这个页面将给你一个快速的复习。

在发布主题之前，测试是非常重要的。 您可能已经构建了最美丽的WordPress主题，但如果有人试图评论或插入图像时，您的主题就不能用于现实世界的使用。

在测试您的主题之前，请确保您已经设置了开发环境。 有许多方法来设置您的环境，其中许多方法在“设置开发环境”页面上有说明。

## 主题单元测试

---

设置开发环境后，您将需要测试内容来测试您的主题。 虽然您可以创建自己的测试内容，但主题审核小组已经创建了主题单元测试，其中包含许多不同类型的内容。 这将有助于确保您的主题在您可能没有预期的情况下工作。 主题单元测试是WordPress导出文件，可以通过使用WordPress导入器导入到任何WordPress安装。 您应该将其导入到本地开发环境中。

## WordPress设置

---

对WordPress安装进行调整和更改是确保您构建主题以处理大量场景的另一个好方法。 使用以下设置来测试您的主题。

- 一般
  - 将站点标题设置得很长，并将Tagline设置为更长。这些设置将测试您的主题如何处理站点标题和标语的边缘案例。
- 读
  - 将“博客页面最多显示” 设置为5.此设置将确保索引/归档分页被触发。
- 讨论
  - 启用“线程评论” 至少3级。此设置将有助于测试您的主题的评论列表样式。
  - 启用“将评论分成页面”，并设置每页5个评论来测试评论的分页和样式。
- 媒体
  - 删除大尺寸媒体的值以测试主题的\$ content\_width设置。
- 固定链接
  - 更改永久链接设置几次，以确保您的主题可以处理各种URL格式。

有关更多设置说明，请查看WordPress法典中的主题单元测试页面。

## WordPress Beta测试仪

---

WordPress发布每年发生三次。测试您的主题是下一个版本的WordPress是一个好主意，所以你可以预料到下一个版本发布时的的问题。这可以通过WordPress Beta Tester插件轻松完成。该插件可以轻松下载最新的每夜版本的WordPress或最新的分支版本（适用于较小的bug修复版本）。这在预测新的主要版本或开发即将到来的功能时尤其有用。

## 测试和调试工具

### Theme Check

---

每个主题都经过一个自动检查，在评论者甚至看到它之前。如果通过自动检查确定主题存在任何直接问题，则主题将被拒绝，并附有关于如何解决问题的说明。主题检查插件在“外观”下添加一个仪表板链接，以便您可以从管理面板运行完全相同的WordPress.org检查。在上传主题之前进行此操作可以让您了解在提交之前需要解决的问题。运行支票将会提供您的主题生成的任何警告的列表，以及WordPress.org主题目录中接受的主题所需的项目以及主题中可能缺少的任何推荐项目。

### Developer

---

开发人员插件实际上只是一个工具，可以自动下载并安装开发主题时所需的一些插件。本手册中讨论的一些内容已经被安装和激活了。其他您可以在激活插件后立即安装。

### Debug Bar

---

调试栏将所有调试消息推送到单独的页面，它们以易于阅读的布局列出，并按消息类型进行组织。还有一些其他插件添加到Debug Bar，扩展其功能或添加更多信息。

### Log Deprecated Notices

---

Log Deprecated Notices显示主题中已弃用的函数通知的列表，以及可以在哪里找到代码。在WordPress的每个主要版本之后，应该至少运行这个应用程序，以便您可以从主题中解析并删除任何已弃用的代码和函数。

### Browser testing

---

当您提交主题到WordPress.org时，预计它可以在现代浏览器中以任何分辨率运行。您应该在提交移动设备和桌面设备之前，先测试一些热门浏览器的主题。许多浏览器具有内置功能，可以方便测试，例如Chrome开发工具，Firefox开发工具和Internet Explorer / Microsoft Edge工具。

### Validation

---

同样，您的主题应该使用有效的HTML5和CSS代码。有各种各样的工具将测试您的网站的有效代码，包括这个HTML5验证器和这个CSS验证器。

# 主题评论指南

---

WordPress主题审查小组提供和维护主题审查指南，作为WordPress的贡献者和开发人员WordPress主题目录任务的一部分。在主题包含在主题目录中之前，需要执行其中一些准则。其他指导方针是建议您开发主题的最佳做法，并尽可能多地向其提供。本手册使用主题审查指南的建议和要求，以确保您开发的主题是所有人都可以访问的。

WordPress.org主题目录中包含的要求可以在这里找到。

## 参与其中

WordPress主题评论团队对任何人都是开放的，是更好地了解主题如何开发的好方法。要成为WordPress主题评测团队的成员，请阅读WordPress主题评论小组的网站。

# 写文档

---

文档对于主题很重要，因为它为用户提供了一种了解主题不支持的方式。 同样，记录您的主题的代码将使其他主题开发人员更容易自定义您的主题，可能与一个小孩主题。

以下是您的主题文档的要求和建议列表。

主题需要提供任何设计限制或非常安装/设置说明的最终用户文档。

主题建议包含一个readme.txt文件，使用插件目录的readme.txt标记格式。 该建议即将

# 提交你的主题到WordPress.org

---

在将主题添加到WordPress主题目录之前，主题审核小组将仔细检查主题，以确保他们遵守基本准则。此审查确保全球的WordPress用户可以下载高品质和安全的主题。

## 需要帮忙？

---

如果您有主题发展问题，请发布在开发与WordPress论坛。来自世界各地的志愿者随时准备协助您开发您的主题。

## 指南

---

在上传主题之前，请确保您查看主题评论指南。如果您对这些指南有任何疑问，可以在Slack的#themereview频道中询问他们。任何拥有WordPress.org帐号的人都可以访问“制作WordPress Slack”。

## 测试样本数据

---

WordPress主题审查小组将使用主题单元测试的示例数据来审查您的主题。在上传您的主题以供审核之前，请使用此示例导出数据进行测试。

## 上传你的主题

---

当您准备好提交主题以供审核时，请在主题>上传上传您的主题ZIP。未来更新将通过同一页面上传。主题审查小组网站上提供了有关主题审查流程的更多信息。



# 参考文献

---

本节包含模板标记和条件标记的列表。

模板标签在模板文件中使用以动态显示信息或自定义网站。

条件标记是一种布尔数据类型，可以根据当前页匹配的条件在模板文件中更改内容的显示。

# 模板标签列表

---

## 完整模板标签列表

---

模板标签文件存储在wp-includes目录中。 这些文件的后缀为 “-template.php” ，以区别于其他WordPress文件。 有9个模板标签文件：

- wp-includes/general-template.php
- wp-includes/author-template.php
- wp-includes/bookmark-template.php
- wp-includes/category-template.php
- wp-includes/comment-template.php
- wp-includes/link-template.php
- wp-includes/post-template.php
- wp-includes/post-thumbnail-template.php
- wp-includes/nav-menu-template.php

# 标签

## 通用标签

---

wp-includes/general-template.php

- get\_header()
- get\_footer()
- get\_sidebar()
- get\_template\_part()
- get\_search\_form()
- wp\_loginout()
- wp\_logout\_url()
- wp\_login\_url()
- wp\_login\_form()
- wp\_lostpassword\_url()
- wp\_register()
- wp\_meta()
- bloginfo()
- get\_bloginfo()

- `get_current_blog_id()`
- `wp_title()`
- `single_post_title()`
- `post_type_archive_title()`
- `single_cat_title()`
- `single_tag_title()`
- `single_term_title()`
- `single_month_title()`
- `get_archives_link()`
- `wp_get_archives()`
- `calendar_week_mod()`
- `get_calendar()`
- `delete_get_calendar_cache()`
- `allowed_tags()`
- `wp_enqueue_script()`

## 作者标签

---

`wp-includes/author-template.php`

- `the_author()`
- `get_the_author()`
- `the_author_link()`
- `get_the_author_link()`
- `the_author_meta()`
- `the_author_posts()`
- `the_author_posts_link()`
- `wp_dropdown_users()`
- `wp_list_authors()`
- `get_author_posts_url()`

## 书签标签

---

`wp-includes/bookmark-template.php` 和 `wp-includes/bookmark.php`

- `wp_list_bookmarks()`
- `get_bookmark()`
- `get_bookmark_field()`

- `get_bookmarks()`

## 类别标签

---

`wp-includes/category-template.php`

- `category_description()`
- `single_cat_title()`
- `the_category()`
- `the_category_rss()`
- `wp_dropdown_categories()`
- `wp_list_categories()`
- `single_tag_title()`
- `tag_description()`
- `the_tags()`
- `wp_generate_tag_cloud()`
- `wp_tag_cloud()`
- `term_description()`
- `single_term_title()`
- `get_the_term_list()`
- `the_terms()`
- `the_taxonomies()`

## 评论标签

---

`wp-includes/comment-template.php`

- `cancel_comment_reply_link()`
- `comment_author()`
- `comment_author_email()`
- `comment_author_email_link()`
- `comment_author_IP()`
- `comment_author_link()`
- `comment_author_rss()`
- `comment_author_url()`
- `comment_author_url_link()`
- `comment_class()`
- `comment_date()`

- `comment_excerpt()`
- `comment_form_title()`
- `comment_form()`
- `comment_ID()`
- `comment_id_fields()`
- `comment_reply_link()`
- `comment_text()`
- `comment_text_rss()`
- `comment_time()`
- `comment_type()`
- `comments_link()`
- `comments_number()`
- `comments_popup_link()`
- `get_avatar()`
- `next_comments_link()`
- `paginate_comments_links()`
- `permalink_comments_rss()`
- `previous_comments_link()`
- `wp_list_comments()`

## 链接标签

---

`wp-includes/link-template.php`

- `the_permalink()`
- `user_trailingslashit()`
- `permalink_anchor()`
- `get_permalink()`
- `get_post_permalink()`
- `get_page_link()`
- `get_attachment_link()`
- `wp_shortlink_header()`
- `wp_shortlink_wp_head()`
- `edit_bookmark_link()`
- `edit_comment_link()`
- `edit_post_link()`

- `get_edit_post_link()`
- `get_delete_post_link()`
- `edit_tag_link()`
- `get_admin_url()`
- `get_home_url()`
- `get_site_url()`
- `home_url()`
- `site_url()`
- `get_search_link()`
- `get_search_query()`
- `the_feed_link()`

## 内容标签

---

wp-includes/post-template.php

- `body_class()`
- `next_image_link()`
- `next_post_link()`
- `next_posts_link()`
- `post_class()`
- `post_password_required()`
- `posts_nav_link()`
- `previous_image_link()`
- `previous_post_link()`
- `previous_posts_link()`
- `single_post_title()`
- `the_category()`
- `the_category_rss()`
- `the_content()`
- `the_excerpt()`
- `the_excerpt_rss()`
- `the_ID()`
- `the_meta()`
- `the_tags()`
- `the_title()`

- `get_the_title()`
- `the_title_attribute()`
- `the_title_rss()`
- `wp_link_pages()`
- `get_attachment_link()`
- `wp_get_attachment_link()`
- `the_attachment_link()`
- `the_search_query()`
- `is_attachment()`
- `wp_attachment_is_image()`
- `wp_get_attachment_image()`
- `wp_get_attachment_image_src()`
- `wp_get_attachment_metadata()`
- `get_the_date()`
- `single_month_title()`
- `the_date()`
- `the_date_xml()`
- `the_modified_author()`
- `the_modified_date()`
- `the_modified_time()`
- `the_time()`
- `the_shortlink()`
- `wp_get_shortlink()`

## 内容缩略图标签

---

`wp-includes/post-thumbnail-template.php`

- `has_post_thumbnail()`
- `get_post_thumbnail_id()`
- `the_post_thumbnail()`
- `get_the_post_thumbnail()`

## 导航菜单标签

---

`wp-includes/nav-menu-template.php`

- `wp_nav_menu()`

- `walk_nav_menu_tree()`

## 参见

---

- [条件标签](#)



# 条件标签列表

条件标签是一种布尔数据类型，可以在您的模板文件中使用，以根据当前页面匹配的条件来更改内容的显示。 他们告诉WordPress在特定条件下显示什么代码。 条件标签通常与PHP一起使用，if/else条件语句，并与WordPress模板层次结构有密切关系。

**警告：**您只能在设置WP\_Query或使用动作挂钩后使用条件查询标签。

## 条件标签的完整列表

- `is_front_page()`
- `is_home()`
- `is_front_page()`
- `is_home()`
- `is_admin()`
- `is_network_admin()`
- `is_admin_bar_showing()`
- `is_single()`
- `is_sticky()`
- `is_post_type_hierarchical( $post_type )`
- `is_post_type_archive()`
- `is_comments_popup()`
- `comments_open()`
- `pings_open()`
- `is_page()`
- `is_page_template()`
- `is_category( $category )`
- `is_tag()`
- `is_tax()`
- `has_term()`
- `term_exists( $term, $taxonomy, $parent )`
- `is_taxonomy_hierarchical( $taxonomy )`
- `taxonomy_exists( $taxonomy )`
- `is_author()`
- `is_date()`
- `is_year()`

- `is_month()`
- `is_day()`
- `is_time()`
- `is_new_day()`
- `is_archive()`
- `is_search()`
- `is_404()`
- `is_paged()`
- `is_attachment()`
- `wp_attachment_is_image( $post_id )`
- `is_local_attachment( $url )`
- `is_singular()`
- `post_type_exists( $post_type )`
- `is_main_query()`
- `is_new_day()`
- `is_feed()`
- `is_trackback()`
- `is_preview()`
- `in_the_loop()`
- `is_dynamic_sidebar()`
- `is_active_sidebar()`
- `is_active_widget( $widget_callback, $widget_id )`
- `is_blog_installed()`
- `is_rtl()`
- `is_multisite()`
- `is_main_site()`
- `is_super_admin()`
- `is_user_logged_in()`
- `email_exists( $email )`
- `username_exists( $username )`
- `is_plugin_active( $path )`
- `is_plugin_inactive( $path )`
- `is_plugin_active_for_network( $path )`
- `is_plugin_page()`
- `is_child_theme()`

- `current_theme_supports()`
- `has_post_thumbnail( $post_id )`
- `wp_script_is( $handle, $list )`

## 条件

---

所有条件标签测试以查看某个条件是否满足，然后返回TRUE或FALSE。 下面列出了各种标签输出TRUE的条件。 可以接受参数的那些标签如此注明。

## 主页

---

- `is_home()`

## 首页

---

- `is_front_page()`

## 博客页

---

- `is_front_page()`
- `is_home()`

## 单页面

---

- `is_single()`

## 内容页

---

- `is_page()`
- `is_page_template()`

## 帖子缩略图

---

- `has_post_thumbnail( $post_id )`

单个页面，单个内容，附件或任何其他自定义帖子类型

- `is_singular()`

## 类别页面

---

- `is_category( $category )`

## 标签页

---

- `is_tag()`
- `has_tag()`

## 分类页（及相关）

---

- `is_tax()`
- `has_term()`
- `term_exists( $term, $taxonomy, $parent )`
- `is_taxonomy_hierarchical( $taxonomy )`
- `taxonomy_exists( $taxonomy )`

## 作者页

---

- `is_author()`

## 日期页

---

- `is_date()`
- `is_year()`
- `is_month()`
- `is_day()`
- `is_time()`
- `is_new_day()`

## 任何档案页

---

- `is_archive()`

## 搜索结果页

---

- `is_search()`

## 404找不到页面

---

- `is_404()`

## 动态SideBar

---

- `is_dynamic_sidebar()`

## SideBar活跃

---

- `is_active_sidebar()`

## Widget活跃

---

- `is_active_widget( $widget_callback, $widget_id )`

## 用户登录

---

- `is_user_logged_in()`

## 电子邮件存在

---

- `email_exists( $email )`

## 用户名存在

---

- `username_exists( $username )`

## 分页

---

- `is_paged()`

## 右到左

---

- `is_rtl()`

## 附件

---

- `is_attachment()`

## 附件是图像

---

- `wp_attachment_is_image( $post_id )`

## 本地附件

---

- `is_local_attachment( $url )`

## 文章类型存在

---

- `post_type_exists( $post_type )`

## 是主要查询

---

- `is_main_query()`

## 新的一天

---

- `is_new_day()`

## 银团贷款

---

- `is_feed()`

## 一个引用

---

- `is_trackback()`

## 预览

---

- `is_preview()`

## 有一个摘录

---

- `has_excerpt()`

## 有一个导航菜单的安排

---

`has_nav_menu()`

## 博客安装

---

- `is_blog_installed()`

## 网络的一部分（多站点）

---

- `is_multisite()`
- `is_main_site()`
- `is_super_admin()`

## 一个活跃的插件

---

- `is_plugin_active( $path )`
- `is_plugin_inactive( $path )`
- `is_plugin_active_for_network( $path )`
- `is_plugin_page()`

## 一个子主题

---

- `is_child_theme()`

## 主题支持功能

---

- `current_theme_supports()`

## 在定制预览

---

- `is_customize_preview()`

# 编码标准

---

WordPress编码标准的目的是在WordPress开源项目和社区的各个方面，从核心代码到主题到插件，创建一个协作和审查的基准。

WordPress社区制定了本手册中包含的标准，这些标准是开发人员和核心贡献者建议遵循的最佳做法的一部分。

## 为什么要编码标准？

---

编码标准有助于避免常见的编码错误，提高代码的可读性，并简化修改。他们确保项目中的文件看起来像是由一个人创建的。

遵循标准意味着任何人都可以理解一段代码，并在需要时对其进行修改，而不考虑何时被编写或由谁编写。

如果您计划为WordPress核心做出贡献，您需要熟悉这些标准，因为您提交的任何代码都需要符合这些标准。

## 语言特定标准

---

- [CSS编码标准](#)
- [HTML编码标准](#)
- [JavaScript编码标准](#)
- [PHP编码标准](#)

## 无障碍标准

---

WordPress致力于满足所有新的和更新的代码在AA级的Web内容无障碍指南（WCAG）。我们提供了在创建修补程序或功能插件时应该注意的常见可访问性问题的快速指南。



# HTML编码标准

## #HTML编码标准

### 验证

所有HTML页面都应该针对W3C验证器进行验证，以确保标记格式正确。这本身并不直接表示良好的代码，但它有助于消除能够通过自动化测试的问题。它不能替代手动代码审查。（对于其他验证者，请参阅食典中的HTML验证。）

### 自闭元素

所有标签必须正确关闭。对于可以包装节点（如文本或其他元素）的标签，终止是一项琐碎的任务。对于自动关闭的标签，正斜杠应该有一个空格：

```
<br />
```

不正确：

```
<br/>
```

W3C指定单个空格应位于自动关闭斜杠（源）之前。

### 属性和标签

所有标签和属性必须用小写字母写。另外，当其中的文本的目的仅由机器解释时，属性值应该是小写的。对于数据需要人为可读的情况，应遵循适当的标题大写。

对于机器：

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

对于人类：

```
<a href="http://example.com/" title="Description Here">Example.com</a>
```

### 属性值

根据XHTML的W3C规范，所有属性必须具有值，并且必须使用双引号或单引号（源）。以下是引号和属性/值对的适当和不正确使用的示例。

正确：

```
<input type="text" name="email" disabled="disabled" />
<input type='text' name='email' disabled='disabled' />
```

不正确:

```
<input type=text name=email disabled>
```

在HTML中，属性并不都必须具有值，并且属性值并不总是被引用。虽然上述所有示例都是有效的HTML，但无法引用属性可能会导致安全漏洞。始终引用属性。

## 缩进

与PHP一样，HTML缩进应始终反映逻辑结构。使用标签，而不是空格。

将PHP和HTML混合在一起时，缩进PHP块以匹配周围的HTML代码。关闭PHP块应该与打开块相同的缩进级别。

正确:

```
<?php if ( ! have_posts() ) : ?>
    <div id="post-1" class="post">
        <h1 class="entry-title">Not Found</h1>
        <div class="entry-content">
            <p>Apologies, but no results were found.</p>
            <?php get_search_form(); ?>
        </div>
    </div>
<?php endif; ?>
```

不正确:

```
<?php if ( ! have_posts() ) : ?>
    <div id="post-0" class="post error404 not-found">
        <h1 class="entry-title">Not Found</h1>
        <div class="entry-content">
            <p>Apologies, but no results were found.</p>
        <?php get_search_form(); ?>
        </div>
    </div>
<?php endif; ?>
```

# CSS编码标准

## #CSS编码标准

像任何编码标准一样，WordPress CSS编码标准的目的是在WordPress开源项目和社区的各个方面，从核心代码到主题到插件，创建一个基准。项目中的文件应该显示为由单个实体创建。最重要的是，创建可读，有意义，一致和美观的代码。

在核心样式表中，通常会发现不一致。我们正在努力解决这些问题，并尽一切努力从这一点开始补丁并提出遵循CSS编码标准。有关上述的更多信息和对UI /前端开发的贡献将在一套单独的指南中提出。

## 结构

有很多不同的方法来构造样式表。以CSS为核心，重要的是要保持高度的可读性。这使得后续的贡献者能够清楚地了解文档的流程。

- 使用制表符，而不是空格来缩进每个属性。
- 在段之间添加两条空白行，并在段中添加一行空白行。
- 每个选择器应该在自己的行上，以逗号或开放的大括号结尾。属性值对应该在自己的行上，一个缩进的选项卡和一个结尾的分号。关闭支架应该向左齐平，使用与打开选择器相同的缩进水平。

正确:

```
#selector-1,
#selector-2,
#selector-3 {
    background: #fff;
    color: #000;
}
```

不正确

```
#selector-1, #selector-2, #selector-3 {
    background: #fff;
    color: #000;
}

#selector-1 { background: #fff; color: #000; }
```

## 选择器

具有特殊性，承担着很大的责任。广泛的选择器使我们能够高效，但如果没有测试可能会产生不利后果。特定

于位置的选择器可以节省时间，但会很快导致杂乱的样式表。做出最好的判断，创建选择器，在DOM的整体风格和布局之间找到适当的平衡。

- 类似于用于文件名的WordPress PHP编码标准，在命名选择器时使用带有连字符的小写和单独的单词。避免使用驼峰命名和下划线。
- 使用人类可读选择器来描述他们的风格。
- 属性选择器应该围绕值使用双引号
- 避免使用组合的选择器，div.container可以简单地表示为.container

正确：

```
#comment-form {  
    margin: 1em 0;  
}  
  
input[type="text"] {  
    line-height: 1.1;  
}
```

不正确

```
#commentForm { /* 避免使用驼峰命名 */  
    margin: 0;  
}  
  
#comment_form { /* 避免下划线。 */  
    margin: 0;  
}  
  
div#comment_form { /* 避免使用组合的选择器。 */  
    margin: 0;  
}  
  
#c1-xr { /* 使用更好的名字。 */  
    margin: 0;  
}  
  
input[type=text] { /* 属性选择器应该围绕值使用双引号 */  
    line-height: 110% /* Also doubly incorrect */  
}
```

## 属性

类似于选择器，太具体的属性将阻碍设计的灵活性。少即是多。确保您不会重复造型或引入固定尺寸（当液体

溶液更可接受时)。

- 属性后面应该有一个冒号和一个空格。
- 除字体名称和供应商特定属性外，所有属性和值都应均为小写。
- 对于颜色使用十六进制代码，如果需要透明度，则使用rgba( )。避免RGB格式和大写字母，并尽可能缩短值：#fff而不是#FFFFFF。
- 尽可能使用缩写（除了覆盖样式）的背景，边框，字体，列表样式，边距和填充值。（有关速记参考，请参阅CSS速记。）

正确：

```
#selector-1 {  
  background: #fff;  
  display: block;  
  margin: 0;  
  margin-left: 20px;  
}
```

不正确:

```
#selector-1 {  
  background: #FFFFFF;  
  display: BLOCK;  
  margin-left: 20PX;  
  margin: 0;  
}
```

## 属性排序

最重要的是，以某种方式选择对您有意义的内容和语义。随机排序是混乱，不是诗歌。在WordPress Core中，我们的选择是逻辑或分组排序，其中属性按意义进行分组，并在这些组内特别排序。组内的属性也有策略地排序，以在节之间创建转换，例如直接在颜色之前的背景。订购的基准是：

- 显示
- 定位
- 盒子模型
- 颜色和排版
- 其他

核心本身尚未使用的内容（例如CSS3动画）可能没有上述规定的位置，但可能以合乎逻辑的方式适合上述之一。正如CSS正在发展，所以我们的标准将会随之发展。

- top/right/bottom/left ( TRBL /故障 ) 应该是任何相关属性 ( 例如边距 ) 的顺序, 就像顺序在值中一样。拐角说明符 ( 例如border-radius - \* - \* ) 应该是左上, 右上, 右下, 左下角。这来自于如何订购速记值。

示例:

```
#overlay {  
  position: absolute;  
  z-index: 1;  
  padding: 10px;  
  background: #fff;  
  color: #777;  
}
```

经常使用的另一种方法, 包括Automattic / WordPress.com主题小组, 是按字母顺序排列属性, 有或没有某些例外。

示例:

```
#overlay {  
  background: #fff;  
  color: #777;  
  padding: 10px;  
  position: absolute;  
  z-index: 1;  
}
```

## 浏览器前缀

我们使用Autoprefixer作为预提交工具来轻松管理必要的浏览器前缀, 从而使本部分的大部分成为可能。对于没有使用Grunt感兴趣的用户, 浏览器前缀应该最长 ( -webkit- ) 到最短 ( 未修改 )。所有其他间距依照其余标准。

```
.sample-output {  
  -webkit-box-shadow: inset 0 0 1px 1px #eee;  
  -moz-box-shadow: inset 0 0 1px 1px #eee;  
  box-shadow: inset 0 0 1px 1px #eee;  
}
```

## 属性的值

有许多方法可以输入属性的值。遵循以下准则, 帮助我们保持高度的一致性。

- 空格之前的值, 冒号后
- 不要用空格填写圆括号

- 总是以分号结尾
- 使用双引号而不是单引号，只有在需要时，例如当字体名称有空格时。
- 应使用数值来定义字体权重（例如，400而不是正常值，而不是粗体）。
- 除非必要，否则0值不应有单位，例如具有过渡持续时间。
- 线高度也应该是无单位的，除非有必要将其定义为特定的像素值。这不仅仅是一个风格的惯例，但值得一提的是这里。更多信息：<http://meyerweb.com/eric/thoughts/2006/02/08/unitless-line-heights/>
- 对于十进制值使用前导零，包括在rgba（）中。
- 一个属性的多个逗号分隔值应由空格或换行符（包括rgba（））分隔开。换行符应用于较长的多部分值，例如速写属性（如box-shadow和text-shadow）。然后，第一个之后的每个后续值应该在新行上，缩进到与选择器相同的级别，然后间隔到左上与先前的值对齐。

正确：

```
.class { /* Correct usage of quotes */
  background-image: url(images/bg.png);
  font-family: "Helvetica Neue", sans-serif;
  font-weight: 700;
}

.class { /* Correct usage of zero values */
  font-family: Georgia, serif;
  text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.5),
               0 1px 0 #fff;
}
```

不正确:

```
.class { /* Avoid missing space and semicolon */
  background:#fff
}

.class { /* Avoid adding a unit on a zero value */
  margin: 0px 0px 20px 0px;
}

.class {
  font-family: Times New Roman, serif; /* Quote font names when required */
  font-weight: bold; /* Avoid named font weights */
}
```

## 媒体查询

媒体查询允许我们优化降低不同屏幕大小的DOM。如果您正在添加任何内容，请务必在您所针对的突破点上方

和下方进行测试。

通常建议将媒体按媒体分组保存在样式表的底部。

对于核心中的wp-admin.css文件是一个例外，因为它非常大，每个部分基本上都代表自己的样式表。因此，介质查询在适用的部分底部添加。

媒体查询的规则集应该缩进一级。

例：

```
@media all and (max-width: 699px) and (min-width: 520px) {  
  
    /* Your selectors */  
}
```

## 注释

注释和评论自由。如果有关于文件大小问题，请使用最小化的文件和SCRIPT\_DEBUG常量。长注释应手动将行长度分为80个字符。

- 应将目录用于更长的样式表，特别是高分段的样式表。使用索引号（1.0,1.1,2.0等）有助于搜索和跳转到某个位置。
- 注释应该像PHPDoc一样格式化。CSSDoc标准不一定被广泛接受或使用，但其一些方面可能会随时间推移。Section / subsection标题应该在前后有换行符。内联注释不应该有空的换行符，将注释与它所涉及的项目分开。

示例:

```
/**  
 * ## Section title  
 *  
 * Description of section, whether or not it has media queries, etc.  
 */  
  
.selector {  
    float: left;  
}
```

行内:

```
/* This is a comment about this selector */  
.another-selector {  
    position: absolute;  
    top: 0 !important; /* I should explain why this is so !important */  
}
```



## 最佳实践

样式表往往长度很长。焦点慢慢失去，而预期的目标开始重复和重叠。从一开始就编写智能代码有助于我们保留概述，同时在变化中保持灵活性。

- 如果您尝试解决问题，请尝试在添加更多代码之前删除代码。
- 魔法数字是不幸的。这些是一次性用作快速修复的数字。示例：`.box {margin-top: 37px}`。
- DOM会随着时间的推移而改变，将目标要使用的元素，而不是通过其父母“找到它”。示例：在元素上使用 `.highlight`，而不是 `.highlight a`（选择器在父项上）
- 知道何时使用 `height` 属性。当您包含外部元素（如图像）时，应使用它。否则使用 `line-height` 可以获得更大的灵活性。
- 不要重新设置默认属性和值组合（例如，显示：块级元素）。

## 编写如一、符合习惯的CSS的原则

以下文档将概述一个合理的CSS开发风格指导。本指导文件强烈鼓励开发者使用已经存在了的、常见的、合力的文风。您应该有选择地吸纳一些内容来创造您自己的风格指南。

### 1. 通用原则

“作为成功的项目的一员，很重要的一点是意识到只为自己写代码是很糟糕的行为。如果将有成千上万人使用你的代码，那么你需要编写最具明确性的代码，而不是以自我的喜好来彰显自己的智商。” - Idan Gazit

- 别想着过早地优化代码。先得保证它们可读又可理解才行。
- 在任何代码库中，无论有多少人参与及贡献，所有代码都应该如同一个人编写的一样。
- 严格执行一致认可的风格。
- 如果有疑议，则使用现有的、通用的模式。

### 2. 空格

在项目的所有代码中，应该只有一个风格。在空格的使用上，必须始终保持一致。使用空格来提高可读性。

- 永远不要在缩进时混用空格和制表符（又称TAB符号）。
- 在软缩进（使用空格）和真正的制表符间选择其一，并始终坚持这一选择。（推荐使用空格）
- 如果使用空格进行缩进，选择每个缩进所使用的空格数。（推荐：4个空格）

提示：将编辑器配置为“显示不可见内容”。这使你可以清除行尾的空格和不需要缩进的空行里的空格，同时可以避免对注释的污染。

提示：确定好一种空格格式后，您可以用一个[EditorConfig](#)文件（或者其他相同的东西）来帮助在代码库之间维持这种基本的空格约定。

### 3. 注释

良好的注释是非常重要的。请留出时间来描述组件（component）的工作方式、局限性和构建它们的方法。不要让你的团队其它成员

来猜测一段不通用或不明显的代码的目的。

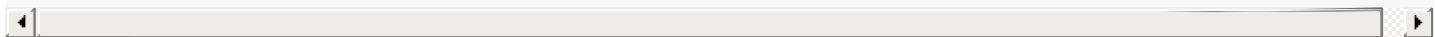
注释的风格应当简洁，并在代码库中保持一致。

- 将注释放在主题上方并独占一行。
- 控制每行长度在合理的范围内，比如80个字符。
- 使用注释从字面上将CSS代码分隔为独立的部分。
- 注释的大小写应当与普通句子相同，并且使用一致的文本缩进。

提示：通过配置编辑器，可以提供快捷键来输出一致认可的注释模式。

CSS示例：

```
/* =====  
  区块代码注释  
===== */  
  
/* 次级区块代码注释  
===== */  
  
/**  
 * “Doxygen式注释格式”的简短介绍  
 *  
 * 较长的说明文本从这里开始，这是这段文字的第一句话，而且这句话还  
 * 没结束，过了好一会儿，我了个去终于结束了，烦不烦啊。  
 *  
 * 当需要进行更细致的解释说明、提供文档文本时，较长的说明文本就很  
 * 有用了。这种长长的说明文本，可以包括示例HTML啦、链接啦等等其  
 * 他你认为重要或者有用的东西。  
 *  
 * @tag 这是一个叫做“tag”的标签。  
 *  
 * TODO：这个“需做”陈述描述了一个接下来要去做的小工作。这种文本，  
 * 如果超长了的话，应该在80个半角字符（如英文）或40个全角字符（  
 * 如中文）后便换行，并（在“ * ”的基础上）再在后面缩进两个空格。  
 */  
  
/* 一般的注释 */
```



## 4. 格式

最终选择的代码风格必须保证：易于阅读，易于清晰地注释，最小化无意中引入错误的可能性，可生成有用的diff和blame。

1. 在多个选择器的规则集中，每个单独的选择器独占一行。
2. 在规则集的左大括号前保留一个空格。
3. 在声明块中，每个声明独占一行。
4. 每个声明前保留一级缩进。
5. 每个声明的冒号后保留一个空格。
6. 对于声明块的最后一个声明，始终保留结束的分号。
7. 规则集的右大括号保持与该规则集的第一个字符在同一列。
8. 每个规则集之间保留一个空行。

```
.selector-1,
.selector-2,
.selector-3[type="text"] {
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
    display: block;
    font-family: helvetica, arial, sans-serif;
    color: #333;
    background: #fff;
    background: linear-gradient(#fff, rgba(0, 0, 0, 0.8));
}

.selector-a,
.selector-b {
    padding: 10px;
}
```

### 声明顺序

样式声明的顺序应当遵守一个单一的原则。我的倾向是将相关的属性组合在一起，并且将对结构来说比较重要的属性（如定位或者盒模型）

放在前面，先于排版、背景及颜色等属性。

小型的开发团体，可能会想要把相关的属性声明（比如说定位和箱模型）摆在一起。

```
.selector {
    /* Positioning */
    position: absolute;
```

```

    z-index: 10;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;

    /* Display & Box Model */
    display: inline-block;
    overflow: hidden;
    box-sizing: border-box;
    width: 100px;
    height: 100px;
    padding: 10px;
    border: 10px solid #333;
    margin: 10px;

    /* Other */
    background: #000;
    color: #fff;
    font-family: sans-serif;
    font-size: 16px;
    text-align: right;
}

```

大型团队，则可能更喜欢按字母排序，因为这样做起来很方便，而且维护起来很容易。

### 例外及细微偏移原则的情况

对于大量仅包含单条声明的声明块，可以使用一种略微不同的单行格式。在这种情况下，在左大括号之后及右大括号之前都应该保留一个空格。

```

.selector-1 { width: 10%; }
.selector-2 { width: 20%; }
.selector-3 { width: 30%; }

```

对于以逗号分隔并且非常长的属性值 -- 例如一堆渐变或者阴影的声明 -- 可以放在多行中，这有助于提高可读性，并易于生成有效的diff。这种情况下有多种格式可以选择，以下展示了一种格式。

```

.selector {
    box-shadow:
        1px 1px 1px #000,
        2px 2px 1px 1px #ccc inset;
    background-image:
        linear-gradient(#fff, #ccc),
        linear-gradient(#f3c, #4ec);
}

```

```
}
```

## 杂项

- 在十六进制值中，使用小写，如 `#aaa`。
- 单引号或双引号的选择保持一致。推荐使用双引号，如 `content: ""`。
- 对于选择器中的属性值也加上引号，如 `input[type="checkbox"]`。
- 如果可以的话，不要给0加上单位，如 `margin: 0`。

## 预处理：格式方面额外的考虑

不同的CSS预处理有着不同的特性、功能以及语法。编码习惯应当根据使用的预处理程序进行扩展，以适应其特有的功能。推荐在使用SCSS时遵守以下指导。

- 将嵌套深度限制在1级。对于超过2级的嵌套，给予重新评估。这可以避免出现过于详实的CSS选择器。
- 避免大量的嵌套规则。当可读性受到影响时，将之打断。推荐避免出现多于20行的嵌套规则出现。
- 始终将 `@extend` 语句放在声明块的第一行。
- 如果可以的话，将 `@include` 语句放在声明块的顶部，紧接着 `@extend` 语句的位置。
- 考虑在自定义函数的名字前加上 `x-` 或其它形式的前缀。这有助于避免将自己的函数和CSS的原生函数混淆，或函数命名与库函数冲突。

```
.selector-1 {
  @extend .other-rule;
  @include clearfix();
  @include box-sizing(border-box);
  width: x-grid-unit(1);
  // other declarations
}
```

## 5. 命名

不要试着把自己当作编译器或者预处理器，因此命名的时候尽量采用清晰的、有意义的名字。另外，对于html文件和css文件中的代码，尽量采用一致的命名规则。

```
/* 没有意义的命名 */
.s-scr {
  overflow: auto;
}
.cb {
  background: #000;
}

/* 比较有意义的命名方式 */
```

```
.is-scrollable {
  overflow: auto;
}
.column-body {
  background: #000;
}
```

## 6. 实例

下面是含有上述约定的示例代码：

```
/* =====
Grid layout
===== */

/**
 * Column layout with horizontal scroll.
 *
 * This creates a single row of full-height, non-wrapping columns that can
 * be browsed horizontally within their parent.
 *
 * Example HTML:
 *
 * <div class="grid">
 *   <div class="cell cell-3"></div>
 *   <div class="cell cell-3"></div>
 *   <div class="cell cell-3"></div>
 * </div>
 */

/**
 * Grid container
 *
 * Must only contain `.cell` children.
 *
 * 1. Remove inter-cell whitespace
 * 2. Prevent inline-block cells wrapping
 */

.grid {
  height: 100%;
  font-size: 0; /* 1 */
  white-space: nowrap; /* 2 */
}

/**
 * Grid cells
 *
```

```

* No explicit width by default. Extend with `.cell-n` classes.
*
* 1. Set the inter-cell spacing
* 2. Reset white-space inherited from `.grid`
* 3. Reset font-size inherited from `.grid`
*/

.cell {
  position: relative;
  display: inline-block;
  overflow: hidden;
  box-sizing: border-box;
  height: 100%;
  padding: 0 10px; /* 1 */
  border: 2px solid #333;
  vertical-align: top;
  white-space: normal; /* 2 */
  font-size: 16px; /* 3 */
}

/* Cell states */

.cell.is-animating {
  background-color: #fffdec;
}

/* Cell dimensions
===== */

.cell-1 { width: 10%; }
.cell-2 { width: 20%; }
.cell-3 { width: 30%; }
.cell-4 { width: 40%; }
.cell-5 { width: 50%; }

/* Cell modifiers
===== */

.cell--detail,
.cell--important {
  border-width: 4px;
}

```

## 7. 代码组织

对于css代码库来说，如何组织代码文件是很重要的，尤其对于那些很大的代码库显得更加重要。这里介绍

几个组织代码的建议：

- 逻辑上对不同的代码进行分离。
- 不同的组件(component)的css尽量用不同的css文件（可以在build阶段用工具合并到一起）
- 如果用到了预处理器（比如less），把一些公共的样式代码抽象成变量，例如颜色，字体等等。

## 8. 构建及部署

---

任何一个项目，都应该有一个build的过程，在这个阶段我们可以通过工具对代码进行检测，测试，压缩等等，还可以为生产环境准备好带有版本号的代码。在这里我推荐一下[grunt](#)这个工具，我感觉它很不错。

## 致谢

---

感谢所有对[idiomatic.js](#)做出贡献的网友。

##许可

Principles of writing consistent, idiomatic CSS 是Nicolas Gallagher发布在[Creative Commons Attribution 3.0 Unported License](#)许可证下的作品。该许可证适用于本代码栈中的所有文档，包括翻译文本。

本作品基于[github.com/necolas/idiomatic-css](#)著就。



# JavaScript编码标准

## JavaScript编码标准

JavaScript已经成为开发基于WordPress的应用程序（主题和插件）以及WordPress核心的关键组件。格式化和造型JavaScript代码需要标准来保持与WordPress标准提供的核心PHP，HTML和CSS代码相同的代码一致性。任何代码库中的所有代码应该看起来像一个人打字，无论有多少人贡献。 - 写作一致，惯用JavaScript的原则

WordPress JavaScript编码标准是从jQuery JavaScript样式指南中进行的。我们的标准与jQuery指南在以下方面不同：

- WordPress使用单引号来进行字符串声明。
- Case语句在切换块中缩进。
- 函数内容始终缩小，包括全文件封闭包装器。
- 一些空格规则不一致，与WordPress PHP编码标准保持一致。
- 鼓励jQuery的100字符硬线限制，但不会严格执行。

下面的很多例子直接从jQuery风格指南中进行了调整;这些差异已经被整合到这个页面上的例子中。以下任何标准和示例应被视为WordPress代码的最佳做法，除非明确指出为反模式。

### 代码重构

“代码重构不应该只是因为我们可以做到。” - Lead Developer Andrew Nacin

JavaScript的WordPress代码结构的许多部分的风格不一致。WordPress正在努力逐步改进这一点，所以代码将一览无余，易于阅读。

虽然编码标准很重要，但重构旧的.js文件只是为了符合标准不是一个紧迫的问题。强烈建议不要使用旧的文件“仅空白”补丁。

所有新的或更新的JavaScript代码将被审查，以确保它符合标准，并通过JSHint。

### 间距

在你的代码中使用空格。 “当有疑问的时候，空间就出来了。

这些规则鼓励自由空间提高开发者的可读性。缩小过程创建一个文件，该文件针对浏览器进行了优化，以进行读取和处理。

- 用标签缩进。
- 在行尾或空白行没有空格。
- 线条通常不应超过80个字符，不应超过100个（将标签计数为4个空格）。这是一个“软”规则，但长行通常

表示不可读或无组织的代码。

- 如果 `/ else / for / while / try` 块应该总是使用大括号，并且始终在多行上。
- 一元的特殊字符运算符（例如，`++`，`-`）不能在它们的操作数旁边有空格。
- 任何和不得有上一个空格。
- 任何 `;` 用作声明终止符必须在行尾。
- 任何：在对象定义中的属性名后面不得有前面的空格。并且：在三元条件下，双方必须有空间。空构造中没有填充空间（例如 `{}`，`[]`，`fn ( )`）。
- 每个文件末尾应该有一行。
- 任何 `!` 否定运算符应具有以下空格。
- 所有功能体都缩进一个选项卡，即使整个文件被封装在一起。
- 空格可以在文档块内或行内对齐代码，但只能在行的开头使用选项卡。

WordPress JavaScript 标准比 jQuery 样式指南更倾向于略宽的空白规则。这些偏差是 WordPress 代码库中 PHP 和 JavaScript 文件之间的一致性。

空格可以轻松地累积在行的末尾 - 避免这一点，因为尾随的空格被捕获为 JSHint 中的错误。捕获空格累积的一种方法是在文本编辑器中启用可见的空白字符。

## 对象

如果对象声明很短，可以在一行上进行对象声明（记住行长度指南）。当一个对象声明太长而不能适合一行时，每行必须有一个属性。如果属性名称是保留字或包含特殊字符，则只需引用属性名称：

```
// Preferred
var map = {
  ready: 9,
  when: 4,
  'you are': 15
};

// Acceptable for small objects
var map = { ready: 9, when: 4, 'you are': 15 };

// Bad
var map = { ready: 9,
  when: 4, 'you are': 15 };
```

## 数组和函数调用

在元素和参数上总是包含多余的空格：

```
array = [ a, b ];

foo( arg );

foo( 'string', object );

foo( options, object[ property ] );

foo( node, 'property', 2 );
```

例外:

```
// For consistency with our PHP standards, do not include a space around
// string literals or integers used as key values in array notation:
prop = object['default'];
firstArrayElement = arr[0];

// Function with a callback, object, or array as the sole argument:
// No space on either side of the argument
foo(function() {
    // Do stuff
});

foo({
    a: 'alpha',
    b: 'beta'
});

foo([
    'alpha',
    'beta'
]);

// Function with a callback, object, or array as the first argument:
// No space before the first argument
foo(function() {
    // Do stuff
}, options );

// Function with a callback, object, or array as the last argument:
// No space after after the last argument
foo( data, function() {
    // Do stuff
});
```

## 良好间距的示例

```
var i;

if ( condition ) {
    doSomething( 'with a string' );
} else if ( otherCondition ) {
    otherThing({
        key: value,
        otherKey: otherValue
    });
} else {
    somethingElse( true );
}

// Unlike jQuery, WordPress prefers a space after the ! negation operator.
// This is also done to conform to our PHP standards.
while ( ! condition ) {
    iterating++;
}

for ( i = 0; i < 100; i++ ) {
    object[ array[ i ] ] = someFn( i );
    $( '.container' ).val( array[ i ] );
}

try {
    // Expressions
} catch ( e ) {
    // Expressions
}
```

## 分号

使用它们。 不要依靠自动分号插入（ASI）。

## 缩进和换行符

缩进和换行符增加了对复杂语句的可读性。

标签应用于缩进。 即使整个文件包含在一个闭包（即一个立即被调用的函数）中，该函数的内容应该被一个tab缩进：

```
(function( $ ) {
    // Expressions indented

    function doSomething() {
        // Expressions indented
    }
})
```

```
})( jQuery );
```

## 块和大括号

如果，其他，for，while和try块应该始终使用大括号，并且始终在多行。开放大括号应与功能定义，条件或循环在同一行。关闭括号应该直接跟随块的最后一个语句。

```
var a, b, c;

if ( myFunction() ) {
    // Expressions
} else if ( ( a && b ) || c ) {
    // Expressions
} else {
    // Expressions
}
```

## 多行语句

当一个语句太长而不能适合一行，换行符必须在操作符之后发生。

```
// Bad
var html = '<p>The sum of ' + a + ' and ' + b + ' plus ' + c
          + ' is ' + ( a + b + c );

// Good
var html = '<p>The sum of ' + a + ' and ' + b + ' plus ' + c +
          ' is ' + ( a + b + c );
```

如果提高可读性，则将行分解为逻辑组，例如将三元运算符的每个表达式分割到其自己的行上，即使两者都适合单行。

```
// Acceptable
var baz = ( true === conditionalStatement() ) ? 'thing 1' : 'thing 2';

// Better
var baz = firstCondition( foo ) && secondCondition( bar ) ? qux( foo, bar ) : foo;
```

当条件太长而不能适合一行时，连续的行必须缩进一个额外的级别以将其与身体区分开来。

```
if ( firstCondition() && secondCondition() &&
    thirdCondition() ) {
    doStuff();
}
```

```
}
```

## 链接方法调用

当一连串的方法调用太长时间不适合一行，每行必须有一个调用，第一个调用与调用方法的对象分开。如果方法更改上下文，则必须使用额外的缩进级别。

```
elements
  .addClass( 'foo' )
  .children()
    .html( 'hello' )
  .end()
  .appendTo( 'body' );
```

## var声明变量和全局变量

### 用var声明变量

每个函数应该以一个逗号分隔的var语句开始，该语句声明任何必需的局部变量。如果函数没有使用var声明变量，则该变量可能会泄漏到外部范围（通常是全局范围，最坏情况），并且可能会无意间引用并修改该数据。var语句中的赋值应该列在单独的行上，而声明可以分组在一行。任何额外的行应该缩进一个额外的选项卡。占用多个行的对象和函数应该被分配到var语句之外，以避免过度缩进。

```
// Good
var k, m, length,
    // Indent subsequent lines by one tab
    value = 'WordPress';

// Bad
var foo = true;
var bar = false;
var a;
var b;
var c;
```

### 全局变量

在过去，WordPress核心使用更大的全局变量。由于内核JavaScript文件有时在插件中使用，所以现有的全局变量不应该被删除。

文件中使用的所有全局变量都应记录在该文件的顶部。多个全局变量可以以逗号分隔。

这个例子会使passwordStrength是该文件中允许的全局变量：

```
/* global passwordStrength:true */
```

passwordStrength之后的“true”表示在此文件中定义了该全局。如果要访问在其他地方定义的全局，请省

略：将全局指定为只读。

## 公共库

Backbone, jQuery, Underscore和全局wp对象都被注册为根jshintrc文件中的允许全局变量。

Backbone和Underscore可以随时直接访问。jQuery应该通过\$通过将jQuery对象传递给匿名函数来访问：

```
(function( $ ) {
    // Expressions
})( jQuery );
```

这将无效调用.noConflict ( ) 或使用另一个变量设置\$。

添加或修改wp对象的文件必须安全地访问全局，以避免覆盖以前设置的属性：

```
// At the top of the file, set "wp" to its existing value (if present)
window.wp = window.wp || {};
```

## 命名约定

变量和函数名称应该是完整的单词，使用小写第一个字母的骆驼壳。这个标准与WordPress PHP编码标准不同。

要用于new的构造函数应该有一个大写的第一个字母 ( UpperCamelCase ) 。

名称应该是描述性的，但不是过分的。允许迭代器使用异常，例如使用i来表示循环中的索引。

## 注释

注释来自他们引用的代码，并且应该始终以空行为前。注释评论的第一个字母，并在写完整句时包括一段时间。

注释标记 ( // ) 和注释文本之间必须有一个空格。

单行注释:

```
// Explanation of something complex on the next line
$( 'p' ).doSomething();
```

多行注释应用于长期注释，另请参阅JavaScript文档标准：

```
/*
 * This is a comment that is long enough to warrant being stretched
 * over the span of multiple lines.
 */
```

使用内联注释作为异常，用于在形式参数列表中注释特殊参数：

```
function foo( types, selector, data, fn, /* INTERNAL */ one ) {
    // Do stuff
}
```

## 比较符

必须使用严格的等式检查 ( === ) 来有利于抽象等价检查 ( == )。唯一的例外是通过null检查undefined和null。

```
// Check for both undefined and null values, for some important reason.
if ( undefOrNull == null ) {
    // Expressions
}
```

## 类型检查

这些是检查对象类型的首选方法：

- String: typeof object === 'string'
- Number: typeof object === 'number'
- Boolean: typeof object === 'boolean'
- Object: typeof object === 'object' or `_isObject( object )`
- Plain Object: `jQuery.isPlainObject( object )`
- Function: `_isFunction( object )` or `jQuery.isFunction( object )`
- Array: `_isArray( object )` or `jQuery.isArray( object )`
- Element: `object.nodeType` or `_isElement( object )`
- null: `object === null`
- null or undefined: `object == null`

undefined:

- Global Variables: `typeof variable === 'undefined'`
- Local Variables: `variable === undefined`
- Properties: `object.prop === undefined`
- Any of the above: `_isUndefined( object )`

随着任何骨干或下划线已经被使用，我们鼓励您使用Underscore.js的类型检查方法来使用jQuery。

## 字符串

对字符串文字使用单引号：



```
var myStr = 'strings should be contained in single quotes';
```

当字符串包含单引号时，需要使用反斜杠（\）进行转义：

```
// Escape single quotes within strings:  
'Note the backslash before the \'single quotes\'';
```

## Switch语句

通常不鼓励使用switch语句，但是在有大量情况下可能会很有用 - 特别是当可以通过相同的块处理多个情况时，或者可以利用掉线逻辑（默认情况）。

使用switch语句时：

- 对于每个案例，除了默认使用中断。当允许语句“通过”时，明确指出。
- 缩进案例语句开关内的一个选项卡。

```
switch ( event.keyCode ) {  
  
    // ENTER and SPACE both trigger x()  
    case $.ui.keyCode.ENTER:  
    case $.ui.keyCode.SPACE:  
        x();  
        break;  
    case $.ui.keyCode.ESCAPE:  
        y();  
        break;  
    default:  
        z();  
}
```

不建议从switch语句中返回值：使用case块来设置值，然后在最后返回这些值。

```
function getKeyCode( keyCode ) {  
    var result;  
  
    switch ( event.keyCode ) {  
        case $.ui.keyCode.ENTER:  
        case $.ui.keyCode.SPACE:  
            result = 'commit';  
            break;  
        case $.ui.keyCode.ESCAPE:  
            result = 'exit';  
            break;  
    }  
    return result;  
}
```

```
        default:
            result = 'default';
    }

    return result;
}
```

## 最佳做法

### 数组

在JavaScript中创建数组应该使用shorthand []构造函数而不是新的Array（）表示法来完成。

```
var myArray = [];
```

您可以在构建期间初始化数组：

```
var myArray = [ 1, 'WordPress', 2, 'Blog' ];
```

在JavaScript中，关联数组被定义为对象。

### 对象

有很多方法可以在JavaScript中创建对象。对象字面符号{}，既是最具性能的，也是最容易阅读的。

```
var myObj = {};
```

应该使用对象字面符号，除非对象需要特定的原型，在这种情况下，应该通过使用new调用构造函数来创建对象。

```
var myObj = new ConstructorMethod();
```

应通过 `obj.prop` 表示法访问对象属性，除非该键是变量，保留字或不是有效标识符的字符串：

```
prop = object.propertyName;
prop = object[ variableKey ];
prop = object['default'];
prop = object['key-with-hyphens'];
```

## 条件

为了与PHP代码标准保持一致，无论何时将对象与字符串，布尔值，整数或其他常量或字面值进行比较，变量应

始终放在右侧，常量或常量放在左侧。

```
if ( true === myCondition ) {
  // Do stuff
}
```

“有点奇怪，就是阅读。习惯了，你会的。”

## 迭代

当使用for循环遍历大型集合时，建议将循环的最大值存储为变量，而不是每次重新计算最大值：

```
// Good & Efficient
var i, max;

// getItemCount() gets called once
for ( i = 0, max = getItemCount(); i < max; i++ ) {
  // Do stuff
}

// Bad & Potentially Inefficient:
// getItemCount() gets called every time
for ( i = 0; i < getItemCount(); i++ ) {
  // Do stuff
}
```

## Underscore.js集合函数

了解和了解Underscore的集合和数组方法。 这些函数（包括`_each`，`.map`和`.reduce`）允许大数据集的高效，可读的变换。

下划线还允许使用常规JavaScript对象进行jQuery样式链接：

```
var obj = {
  first: 'thing 1',
  second: 'thing 2',
  third: 'lox'
};

var arr = _.chain( obj )
  .keys()
  .map(function( key ) {
    return key + ' comes ' + obj[ key ];
  })
  // Exit the chain
  .value();
```

```
// arr === [ 'first comes thing 1', 'second comes thing 2', 'third comes lox' ]
```

## 迭代jQuery集合

jQuery应该用于迭代的唯一时间是迭代jQuery对象的集合：

```
$tabs.each(function( index, element ) {  
    var $element = $( element );  
  
    // Do stuff to $element  
});
```

不要使用jQuery遍历原始数据或者是vanilla JavaScript对象。

## JSHint

JSHint是一种自动化的代码质量工具，旨在捕获JavaScript代码中的错误。JSHint用于WordPress开发，以快速验证补丁没有向前端引入任何逻辑或语法错误。

### 安装和运行JSHint

JSHint使用名为Grunt的工具运行。JSHint和Grunt都是在Node.js中编写的程序。WordPress开发代码随附的配置文件可以轻松安装和配置这些工具。

要安装Node.js，请单击节点网站上的安装链接。您的操作系统的正确安装文件将被下载。按照操作系统的安装步骤安装程序。

一旦安装了Node.js，打开命令行窗口并导航到您签出WordPress SVN存储库副本的目录（使用`cd ~/directoryname`）。您应该在包含package.json文件的根目录中。

接下来，在命令行窗口中输入`npm install`。这将下载并安装WordPress开发中使用的所有Node软件包。

最后，键入`npm install -g grunt-cli`来安装Grunt命令行界面（CLI）包。Grunt CLI是用于在WordPress中实际运行Grunt任务的。

您现在应该能够输入`grunt jshint`以使Grunt检查所有WordPress JavaScript文件的语法和逻辑错误。要检查核心代码，请键入`grunt jshint : core`；只检查单元测试.js文件，键入`grunt jshint : tests`。

### JSHint设置

用于JSHint的配置选项存储在WordPress SVN存储库中的jshintrc文件中。该文件定义JSHint应该在WordPress源代码中找到它们时应该标记哪些错误。

### 定位单个文件

要为JSHint指定单个文件进行检查，请在命令的末尾添加`--file = filename.js`。例如，这只会检查WordPress的核

心JavaScript文件中检查名为“admin-bar.js”的文件：

```
grunt jshint : core --file = admin-bar.js
```

这只会检查单元测试目录中的“password-strength-meter.js”文件：

```
grunt jshint : tests --file = password-strength-meter.js
```

将JSHint限制为单个文件可能会有用，如果您只处理一个或两个特定文件，并且不希望JSHint在每次运行时等待每个文件处理。

## JSHint覆盖：忽略块

在某些情况下，文件的部分应该从JSHint中排除。作为示例，管理栏的脚本文件包含jQuery HoverIntent插件的最小化代码 - 这是不应通过JSHint的第三方代码，尽管它是WordPress核心JavaScript文件的一部分。

要排除JSHint处理的特定文件区域，请将其包含在JSHint指令注释中：

```
/* jshint ignore:start */
if ( typeof jQuery.fn.hoverIntent === 'undefined' ) {
    // hoverIntent r6 - Copy of wp-includes/js/hoverIntent.min.js
    (function(a){a.fn.hoverIntent=.....
})
/* jshint ignore:end */
```

# PHP编码标准

## PHP编码标准

PHP标记的WordPress代码结构的一些部分在其风格上是不一致的。WordPress正在努力通过帮助用户保持一致的风格逐渐改进，以便代码可以一目了然地变得干净，易于阅读。

在为WordPress编写PHP代码时，请注意以下几点：无论是核心编程代码，插件或主题。这些指南在许多方面与Pear标准类似，但在一些关键方面有所不同。

### PHP

#### 单引号和双引号

酌情使用单引号和双引号。如果您不评估字符串中的任何内容，请使用单引号。你应该几乎不必在字符串中避免引号，因为你可以替代你的引用样式，像这样：

```
echo '<a href="/static/link" title="Yeah yeah!">Link name</a>';  
echo "<a href='$link' title='$linktitle'>$linkname</a>";
```

进入属性的文本应该通过 `esc_attr()` 运行，以便单引号或双引号不会结束属性值并使HTML无效并导致安全问题。有关详细信息，请参阅词典中的数据验证。

#### 缩进

您的缩进应始终反映逻辑结构。使用真实的选项卡而不是空格，因为这允许客户端具有最大的灵活性。

异常：如果你有一个代码块，如果事情是对齐的，可以更容易读取，使用空格：

```
[tab]$foo    = 'somevalue';  
[tab]$foo2   = 'somevalue2';  
[tab]$foo34  = 'somevalue3';  
[tab]$foo5   = 'somevalue4';
```

对于关联数组，值应从新行开始。注意最后一个数组项之后的逗号：建议这样做，因为它可以更容易地改变数组的顺序，并在添加新项时使更清晰的差异。

```
$my_array = array(  
[tab]'foo'    => 'somevalue',  
[tab]'foo2'   => 'somevalue2',  
[tab]'foo3'   => 'somevalue3',  
[tab]'foo34'  => 'somevalue3',  
);
```

经验法则：应在起始处使用标签进行缩进，而中间线可用于对齐。

## 大括号风格

大括号应用于所有块中，如下所示：

```
if ( condition ) {
    action1();
    action2();
} elseif ( condition2 && condition3 ) {
    action3();
    action4();
} else {
    defaultaction();
}
```

此外，如果你有一个很长的块，可以考虑它是否可以分成两个或更多个更短的块或者功能。如果你认为这么长的一个块是不可避免的，那么请稍等一下，所以人们可以一眼就知道结束大括号是结束的 - 通常情况下这个逻辑块大约是35行，而不是直观的代码 明显可以注释。

大括号应始终使用，即使不需要使用：

```
if ( condition ) {
    action0();
}

if ( condition ) {
    action1();
} elseif ( condition2 ) {
    action2a();
    action2b();
}

foreach ( $items as $item ) {
    process_item( $item );
}
```

请注意，使用大括号只意味着禁止使用单一语句的内联控制结构。您可以自由地使用控制结构的替代语法（例如，if / endif，while / endwhile） - 特别是在HTML代码嵌入的模板中，例如：

```
<?php if ( have_posts() ) : ?>
    <div class="hfeed">
        <?php while ( have_posts() ) : the_post(); ?>
            <article id="post-<?php the_ID() ?>" class="<?php post_class() ?>">
                <!-- ... -->
```

```

        </article>
    <?php endwhile; ?>
</div>
<?php endif; ?>

```

使用 `elseif` 不是 `else if`

`else if` 不符合 `if | elseif` 块的规范语法。因此，使用 `elseif` 作为条件。

## 正则表达式

应该使用Perl兼容的正则表达式（PCRE，`preg_`函数）优于其POSIX对等体。切勿使用 `/e` 开关，请改用 `preg_replace_callback`。

对于正则表达式，使用单引号的字符串是最方便的，因为与双引号字符串相反，它们只有两个元素：`'`和`\`。

## 简化PHP开始标签

重要提示：切勿使用简化的PHP开始标签。始终使用完整的PHP标签。

正确：

```

<?php ... ?>
<?php echo $var; ?>

```

不正确:

```

<? ... ?>
<?= $var ?>

```

## 删除每行代码末尾的尾随空格

删除每行代码末尾的尾随空格。在文件末尾省略关闭的PHP标签是首选。如果您使用该标签，请确保删除尾随的空格。

## 空间使用

始终将空格放在逗号之后，并在逻辑，比较，字符串和赋值操作符的两边。

```

x == 23
foo && bar
! foo
array( 1, 2, 3 )
$baz . '-5'
$term .= 'X'

```



在if, elseif, foreach, for和switch块的开头和右括号的两侧放置空格。

```
foreach ( $foo as $bar ) { ... }
```

定义一个函数时，可以这样做：

```
function my_function( $param1 = 'foo', $param2 = 'bar' ) { ... }
```

调用函数时，请执行以下操作：

```
my_function( $param1, func_param( $param2 ) );
```

执行逻辑比较时，可以这样做：

```
if ( ! $foo ) { ... }
```

当进行类型转换时，请执行以下操作：

```
foreach ( (array) $foo as $bar ) { ... }
```

```
$foo = (boolean) $bar;
```

当引用数组项时，如果索引是一个变量，则只包括索引周围的空格，例如：

```
$x = $foo['bar']; // correct
$x = $foo[ 'bar' ]; // incorrect
```

```
$x = $foo[0]; // correct
$x = $foo[ 0 ]; // incorrect
```

```
$x = $foo[ $bar ]; // correct
$x = $foo[$bar]; // incorrect
```

## 格式化SQL语句

格式化SQL语句时，如果要保证SQL语句非常复杂，您可以将其分解成多行并缩进。尽管如此，大多数的声明都是一致的。始终将语句的SQL部分大写，如UPDATE或WHERE。

更新数据库的函数应该期望它们的参数在传递时缺少SQL斜杠转义。应尽可能接近查询时间完成转义，最好使用

```
$wpdb->prepare ( )
```

\$wpdb->prepare ( ) 是一种处理SQL查询的转义，引用和int-cast的方法。它使用sprintf ( ) 格式的一个子集。示例：

```
$var = "dangerous"; // raw data that may or may not need to be escaped
$id = some_foo_number(); // data we expect to be an integer, but we're not certain

$wpdb->query( $wpdb->prepare( "UPDATE $wpdb->posts SET post_title = %s WHERE ID = %d", $var, $id ) );
```

%s 用于字符串占位符，%d用于整数占位符。 请注意，它们不是“引用”！\$wpdb->prepare（）将照顾转义和引用我们。这样做的好处是我们不需要记住手动使用esc\_sql（），而且很容易看到有没有被转义，因为查询发生时会发生。

有关详细信息，请参阅词典中的数据验证。

## 数据库查询

避免直接接触数据库。 如果有一个定义的函数可以获取所需的数据，请使用它。 数据库抽象（使用函数而不是查询）有助于保持您的代码向前兼容，并且在结果缓存在内存中的情况下可以快很多倍。

如果您必须触摸数据库，请通过向wp-hackers邮件列表发送消息与某些开发人员联系。 他们可能想考虑为下一个WordPress版本创建一个功能来覆盖您想要的功能。

## 命名约定

在变量，动作和函数名中使用小写字母（绝对不是camelCase）。 通过下划线分隔单词。 不要不必要地缩写变量名称；让代码明确和自我记录。

```
function some_name( $some_variable ) { [...] }
```

类名应使用以下划线分隔的大写字。 任何首字母缩略词都应该是大写的。

```
class Walker_Category extends Walker { [...] }  
class WP_HTTP { [...] }
```

常量应该是大写的，用下划线分隔单词：

```
define( 'DOING_AJAX', true );
```

文件名使用小写。 连字符应该分开单词。

```
my-plugin-name.php
```

类文件名应基于具有类前缀的类名称，类名称中的下划线用连字符替换，例如WP\_Error成为：

```
class-wp-error.php
```

此文件命名标准适用于具有类的所有当前和新文件。 有三个文件包含已被移植到BackPress的代码的一个例外：class.wp-dependencies.php，class.wp-scripts.php，class.wp-styles.php。 这些文件是前面加上类，一个点后面的单词类，而不是一个连字符。

包含wp-includes中的模板标签的文件应该将-template附加到名称的末尾，以使它们显而易见。

```
general-template.php
```

## 函数参数的默认值

在调用函数时，将字符串值设为true和false。

```
// 不正确
function eat( $what, $slowly = true ) {
    ...
}
eat( 'mushrooms' );
eat( 'mushrooms', true ); // what does true mean?
eat( 'dogfood', false ); // what does false mean? The opposite of true?
```

由于PHP不支持命名参数，因此标志的值是无意义的，每次遇到如上例所示的函数调用时，我们必须搜索函数定义。通过使用描述性字符串值而不是布尔值，可以使代码更易读。

```
// Correct
function eat( $what, $speed = 'slowly' ) {
    ...
}
eat( 'mushrooms' );
eat( 'mushrooms', 'slowly' );
eat( 'dogfood', 'quickly' );
```

当需要更多的单词来描述函数参数时，\$args数组可能是更好的模式。

```
// Even Better
function eat( $what, $args ) {
    ...
}
eat ( 'noodles', array( 'speed' => 'moderate' ) );
```

## 命名动态挂钩插值

应使用插值命名动态挂钩，而不是为了可读性和可发现性而连接。

动态挂钩是挂钩，包括其标签名称中的动态值，例如 `{ $new_status } _ { $post-> post_type } ( publish_post )`。钩子标签中使用的变量应该包裹在花括号(和)中，完整的外部标签名称用双引号括起来。这是为了确保PHP可以正确解析插入字符串中给定的变量的类型。

```
do_action( "{$new_status}_{$post->post_type}", $post->ID, $post );
```

在可能的情况下，标签名称中的动态值也应尽可能简洁明了。`$ user_id`比`$ this-> id`更为自我记录。

## 三元运算符

三元运算符是好的，但是如果声明是真的，那么总是让它们进行测试，而不是假的。否则，它只会变得混乱。（一个例外是使用`! empty()`，因为这里的false的测试一般比较直观。）

例如：

```
// (if statement is true) ? (do this) : (else, do this);
$musictype = ( 'jazz' == $music ) ? 'cool' : 'blah';
// (if field is not empty) ? (do this) : (else, do this);
```

## 条件

```
if ( true == $the_force ) {
    $victorious = you_will( $be );
}
```

当进行涉及变量的逻辑比较时，始终将变量放在右侧，并将常量，文字或函数调用放在左侧。如果双方都不是变量，则顺序并不重要。（在计算机科学方面，比较中总是试图把左值的左值和右值放在左边。

在上面的例子中，如果你省略了一个等号（承认它，甚至发生在我们最经验的人），你会得到一个解析错误，因为你不能像一个常量那样分配。如果声明是相反的（`$the_force = true`），则赋值将完全有效，返回1，导致if语句评估为true，您可以追踪该错误一段时间。

有点奇怪，就是阅读。习惯了，你会的。

这适用于`==`，`!=`，`===`和`!==`。对于`<`，`>`，`<=`或`>=`的Yoda条件显然难以阅读，最好避免。

## 聪明的代码

一般来说，可读性比聪明或简洁更重要。

```
isset( $var ) || $var = some_function();
```

虽然上面的这一行很聪明，但如果你不熟悉它需要一些时间。所以，只是写这样：

```
if ( ! isset( $var ) ) {
    $var = some_function();
}
```

### ###错误控制运算符 @

如PHP文档中所述：

PHP支持一个错误控制运算符：at符号（@）。在PHP中添加表达式时，可能会忽略该表达式生成的任何错误消息。

虽然这个操作系统确实存在于Core中，但它通常被懒惰地使用，而不是进行适当的错误检查。它的用法非常不鼓励，即使PHP文档也说明：

警告：目前，“@”错误控制运算符前缀甚至可以关闭将终止脚本执行的关键错误的错误报告。除此之外，这意味着如果您使用“@”来抑制特定功能的错误，或者它不可用或已输入错误，那么脚本就会在那里死亡，没有指示为什么。

`extract()`是一个可怕的函数，使代码更难调试，更难理解。我们应该阻止它的使用和删除我们的所有用途。

# 插件开发

---

欢迎使用WordPress插件开发者手册;你准备好跳到WordPress插件的世界吗？

插件开发者手册是所有WordPress插件的资源。无论您是新的WordPress插件开发，还是您是一名有经验的插件开发人员，您应该可以在这里找到许多与插件有关的问题的答案。

如果你刚开始插件开发，首先阅读介绍，然后继续阅读基础知识。

第3节将介绍插件的安全性。

Hooks是什么让你的插件与WordPress交互。在第4部分中了解它们。

要了解有关您可以在插件中使用的WordPress内置功能的更多信息，请参阅第5 - 11节：管理菜单，短节目，设置，元数据，自定义帖子类型，分类法和用户。

了解如何在第12节中使用HTTP API获取数据。

如果您在插件中使用JavaScript，jQuery或Ajax，则可以在第13部分找到所需的信息。

要使用Cron了解基于时间的WordPress任务，请参阅第14节。

第15-17节将介绍您将插件国际化，准备在WordPress.org上发布，以及您可能会发现的一些开发工具。

WordPress插件开发者手册由WordPress社区为WordPress社区创建。我们一直在寻找更多的贡献者;如果您有兴趣，请通过文档小组博客了解更多有关参与的信息。

# 插件开发简介

---

欢迎来到“插件开发者手册”。无论您是编写第一个插件还是第五十个插件，我们希望此资源可帮助您编写最佳插件。

插件开发者手册涵盖了各种主题 - 从插件头中应该是什么，安全最佳实践，以及可用于构建插件的工具。这也是一个正在进行的工作 - 如果您发现遗失或不完整的内容，请编辑并使其更好。

WordPress有三个主要组件：

- 核心
- 主题
- 插件

这本手册是关于插件和他们如何与WordPress的互动。它将帮助您了解他们的工作原理以及如何创建自己的工作。

## 为什么我们做插件

---

如果在WordPress开发中有一个基本规则，那就是：不要触摸WordPress核心。这意味着您不会编辑核心WordPress文件以向您的网站添加功能。这是因为当WordPress更新到新版本时，它会覆盖所有的核心文件。因此，您要添加的任何功能都应通过使用经过批准的WordPress API的插件添加。

WordPress插件可以像您需要的那样简单或复杂，具体取决于您要做什么。最简单的插件是单个PHP文件。Hello Dolly插件是这样一个插件的例子。插件PHP文件只需要一个插件头，几个PHP函数，以及一些挂钩来附加你的功能。

插件允许您大大扩展WordPress的功能，而不会触及WordPress核心本身。

# 什么是插件

---

插件是扩展WordPress核心功能的代码包。WordPress插件由PHP代码和其他资源（如图像，CSS和JavaScript）组成。

通过制作自己的插件，您正在扩展WordPress，即在WordPress已经提供的功能之上构建附加功能。例如，您可以编写一个插件，显示您网站上十个最新帖子的链接。

或者，使用WordPress的自定义帖子类型，您可以编写一个插件，创建一个全功能的支持票务系统，其中包含电子邮件通知，自定义票据状态和面向客户端的门户。可能性是无止境的！

大多数WordPress插件都是由许多文件组成的，但插件实际上只需要一个主文件，并在标题中使用特定格式的DocBlock。

第一个插件之一，你好Dolly，只有82线长。你好Dolly从WordPress管理员的著名歌曲中显示歌词。PHP文件中使用了一些CSS来控制歌词的风格。

作为WordPress.org插件作者，您有一个惊人的机会来创建一个将被数百万WordPress用户安装，修补和爱的插件。所有你需要做的是把你的好主意变成代码。插件手册在这里帮助您。

# 插件基础

## 入门

最简单的一个WordPress插件是一个带有WordPress插件标题注释的PHP文件。强烈建议您创建一个目录来保存您的插件，以便您的所有插件的文件整齐地组织在一个地方。

要开始创建一个新的插件，请按照以下步骤操作。

- 浏览到您的WordPress安装的wp-content目录。
- 打开plugins目录。
- 创建一个新的目录，并在您的插件（例如plugin-name）之后命名它。
- 打开你的新插件的目录。
- 创建一个新的PHP文件（在您的插件后面命名此文件也很好，例如plugin-name.php）。

以下是Unix命令行上的进程：

```
wordpress$ cd wp-content
wp-content$ cd plugins
plugins$ mkdir plugin-name
plugins$ cd plugin-name
plugin-name$ vi plugin-name.php
```

在上面的例子中，“vi”是文本编辑器的名称。使用适合您的编辑器。

现在您正在编辑您的新插件的PHP文件，您需要添加插件标题注释。这是一个特别格式的PHP块注释，其中包含有关您的插件的元数据，例如其名称和作者。至少，插件标题注释必须包含插件的名称。插件文件夹中只有一个文件应该有标题注释 - 如果您的插件有多个PHP文件，那么这些文件中只有一个应该有注释。

```
<?php
/*
Plugin Name: YOUR PLUGIN NAME
*/
```

保存文件后，您应该能够看到您的WordPress站点中列出的插件。登录您的WordPress站点，然后单击WordPress管理员左侧导航窗格中的插件。此页面显示您的WordPress网站所有插件的列表。您的新插件现在应该在该列表中！

## Hooks: Actions 和 Filters

WordPress钩子允许您在特定点点击WordPress，以更改WordPress的行为，而无需编辑任何核心文件。

WordPress中有两种类型的钩子：动作和过滤器。操作允许您添加或更改WordPress功能，而过滤器允许您更改



内容，因为它被加载并显示给网站用户。

钩子不只是插件开发人员; 钩子广泛用于通过WordPress核心本身提供默认功能。其他挂钩是未使用的占位符，当您需要更改WordPress的工作原理时，您可以直接使用它们。这就是WordPress如此灵活。

## 基本hook

创建插件时需要的3个基本钩子是 `register_activation_hook()` ，  
`register_deactivation_hook()` 和 `register_uninstall_hook()` 。

- 激活插件时，激活钩子就会运行。您将使用它来提供设置插件的功能 - 例如，在选项表中创建一些默认设置。
- 当您停用插件时，会停用停用挂钩。您将使用它来提供一个功能，可以通过插件清除任何临时数据存储。
- 使用WordPress Admin删除插件后，这些卸载方法将被用来清理。您可以使用它来删除插件创建的所有数据，例如添加到选项表中的任何选项。

## 添加 Hooks

您可以添加自己的自定义钩子与`do_action()`，这将允许开发人员通过传递函数通过你的钩子来扩展你的插件。

## 删除 Hooks

您还可以使用`remove_action()`来删除之前定义的函数。例如，如果您的插件是另一个插件的插件，则可以使用与之前插件使用`add_action()`添加的相同函数回调的`remove_action()`。在这些情况下，动作的优先级很重要，因为`remove_action()`将需要在初始`add_action()`之后运行。

从钩子中删除操作时，以及更改优先级时，您应该小心，因为这些更改可能难以看到这些更改将如何影响与同一个钩子的其他交互。我们强烈建议经常进行测试。

您可以在本手册的“钩子”部分中了解更多关于创建钩子和与之互动的信息。

## WordPress APIs

您是否知道WordPress提供了一些应用程序编程接口（API）？这些API可以大大简化您在插件中编写的代码。

你不想重新发明，特别是当这么多人为你做了大量的工作和测试时。

最常见的是选项API，它可以轻松地将数据存储于数据库中，用于插件。如果您正在考虑在插件中使用cURL，那么您可能会感兴趣的是HTTP API。

由于我们在谈论插件，您需要学习插件API。它具有多种功能，可帮助您开发插件。

## WordPress如何加载插件

当WordPress在WordPress管理员的插件页面上加载安装的插件列表时，它将通过plugins文件夹（及其子文件夹）来搜索带有WordPress插件头注释的PHP文件。如果您的整个插件只包含一个单一的PHP文件，如Hello Dolly，该文件可以直接位于plugins文件夹的根目录下。但更常见的是，插件文件将驻留在自己的文件夹中，以

插件命名。

## 共享您的插件

---

有时您创建的插件仅适用于您的网站。但是很多人喜欢与WordPress其他社区分享他们的插件。在分享您的插件之前，您需要做的一件事是选择许可证。这样可以让您的插件的用户知道如何使用您的代码。为了保持与WordPress核心的兼容性，建议您选择使用GNU通用公共许可证（GPLv2 +）的许可证。

# 头部要求

## 头部要求

如“入门”中所述，标题注释是什么告诉WordPress一个文件是一个插件。

至少，标题注释必须包含插件名称，但可以包括几个（通常应该）：

- Plugin Name: (required) 您的插件的名称，将显示在WordPress管理员的插件列表中。
- Plugin URI: 插件的主页，可能在WordPress.org或您自己的网站上。这对您的插件来说必须是唯一的。
- Description: 插件的简短描述，如WordPress管理员的插件部分所示。保持此描述少于140个字符。
- Version: 当前版本号的插件，如1.0或1.0.3。
- Alert: 为您的项目分配版本号时，请记住，WordPress使用PHP `version_compare()` 函数来比较插件版本号。因此，在您发布新版本的插件之前，您应该确保这个PHP函数认为新版本比旧版本“更大”。例如，1.02实际上大于1.1。
- Author: 插件作者的名字。可以使用逗号列出多个作者。
- Author URI: 作者的网站或个人资料在另一个网站，如[WordPress.org](https://WordPress.org)。
- License: 插件的许可证的简称（slug）（例如GPL2）。有关许可证的更多信息可以在WordPress.org指南中找到。
- License URI: 指向许可证全文的链接（例如<https://www.gnu.org/licenses/gpl-2.0.html>）。
- Text Domain: 插件的gettext文本域。更多信息可以在“如何国际化您的插件”页面的“文本域”部分找到。
- Domain Path: 域路径让WordPress知道在哪里找到翻译。有关更多信息，请参见“如何使插件国际化”的“域路径”部分。

具有标题注释的有效PHP文件可能如下所示：

```
<?php
/*
Plugin Name: WordPress.org Plugin
Plugin URI:  https://developer.wordpress.org/plugins/the-basics/
Description: Basic WordPress Plugin Header Comment
Version:     20160911
Author:      WordPress.org
Author URI:  https://developer.wordpress.org/
License:     GPL2
License URI: https://www.gnu.org/licenses/gpl-2.0.html
Text Domain: wporg
Domain Path: /languages
*/
```

# 包括软件许可证

## 包括软件许可证

大多数WordPress插件是在GPL下发布的，这与WordPress本身使用的许可证相同。但是，还有其他选项可用。总是最好清楚地显示您的插件使用的许可证。

在标题要求部分中，我们简要介绍了如何在插件标题注释中指出插件的许可证。另一个常见和鼓励的做法是在主插件文件的顶部附近发布许可证块注释（与插件标题注释相同）。

这个许可证块注释通常看起来像这样：

```
/*
{Plugin Name} is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 2 of the License, or
any later version.

{Plugin Name} is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with {Plugin Name}. If not, see {URI to Plugin License}.
*/
```

当与插件头评论结合时：

```
<?php
/*
Plugin Name: WordPress.org Plugin
Plugin URI:  https://developer.wordpress.org/plugins/the-basics/
Description: Basic WordPress Plugin Header Comment
Version:      20160911
Author:       WordPress.org
Author URI:   https://developer.wordpress.org/
Text Domain: wporg
Domain Path: /languages
License:      GPL2

{Plugin Name} is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 2 of the License, or
any later version.

{Plugin Name} is distributed in the hope that it will be useful,
```

```
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License  
along with {Plugin Name}. If not, see {License URI}.
```

```
*/
```

# 启用 / 停用 Hooks

激活和停用挂钩提供了在插件激活或停用时执行操作的方法。

在激活时，插件可以运行例程来添加重写规则，添加自定义数据库表或设置默认选项值。

在停用时，插件可以运行例程来删除临时数据，如缓存和临时文件和目录。

**警报：**停用挂钩有时与卸载挂钩混淆。卸载钩子最适合永久删除所有数据，例如删除插件选项和自定义表等。

## 启用

要设置激活钩子，请使用`register_activation_hook ( )`函数：

```
register_activation_hook( __FILE__, 'pluginprefix_function_to_run' );
```

## 停用

要设置停用挂钩，请使用`register_deactivation_hook ( )`函数：

```
register_deactivation_hook( __FILE__, 'pluginprefix_function_to_run' );
```

每个这些函数中的第一个参数是指您的主插件文件，这是您已经放置插件标题注释的文件。通常这两个函数将从主插件文件中触发；但是，如果函数放在任何其他文件中，则必须更新第一个参数以正确指向主插件文件。

##示例

激活钩子最常见的用途之一是当插件注册自定义帖子类型时刷新WordPress的固定链接。这摆脱了糟糕的404错误。

我们来看一下如何做到这一点的例子：

```
function pluginprefix_setup_post_types()
{
    // register the "book" custom post type
    register_post_type( 'book', ['public' => 'true'] );
}
add_action( 'init', 'pluginprefix_setup_post_type' );

function pluginprefix_install()
{
    // trigger our function that registers the custom post type
    pluginprefix_setup_post_type();

    // clear the permalinks after the post type has been registered
```

```
flush_rewrite_rules();  
}  
register_activation_hook( __FILE__, 'pluginprefix_install' );
```

如果您不熟悉注册自定义帖子类型，请不要担心 - 稍后将会介绍。这个例子很简单，因为它很常见。使用上面的示例，以下是如何反转此过程并停用插件：

```
function pluginprefix_deactivation()  
{  
    // our post type will be automatically removed, so no need to unregister it  
  
    // clear the permalinks to remove our post type's rules  
    flush_rewrite_rules();  
}  
register_deactivation_hook( __FILE__, 'pluginprefix_deactivation' );
```

有关激活和停用挂钩的更多信息，以下是一些优秀的资源：

- `register_activation_hook()` 在WordPress的功能参考。
- `register_deactivation_hook()` 在WordPress的功能参考。

# 卸载方法

当您从插件卸载时，插件可能需要进行一些清理。

如果用户已停用该插件，然后单击WordPress管理员中的删除链接，则会将插件视为已卸载。

当您的插件被卸载时，您将需要清除插件特定的任何插件选项和/或设置，和/或其他数据库实体（如表）。

经验不足的开发人员有时会使用停用挂钩的错误。

此表说明了停用和卸载之间的区别。

Scenario	Deactivation Hook	Uninstall Hook
Flush Cache/Temp	Yes	No
Flush Permalinks	Yes	No
Remove Options from { \$wpdb->prefix }_options	No	Yes
Remove Tables from wpdb	No	Yes

## 方法1：register\_uninstall\_hook

要设置卸载挂钩，请使用register\_uninstall\_hook()函数：

```
register_uninstall_hook(__FILE__, 'pluginprefix_function_to_run');
```

### ##方法2：uninstall.php

要使用此方法，您需要在插件的根文件夹中创建一个uninstall.php文件。当用户删除插件时，这个魔术文件会自动运行。

例如： /plugin-name/uninstall.php

**警报：**使用uninstall.php时，在执行前，插件应该始终检查常量WP\_UNINSTALL\_PLUGIN以防止直接访问。

该常量将由WordPress在uninstall.php调用期间定义。

当register\_uninstall\_hook ( ) 执行卸载时，常量不被定义。

以下是删除选项条目并删除数据库表的示例：

```
// if uninstall.php is not called by WordPress, die
if (!defined('WP_UNINSTALL_PLUGIN')) {
    die;
}
```



```
$option_name = 'wporg_option';

delete_option($option_name);

// for site options in Multisite
delete_site_option($option_name);

// drop a custom database table
global $wpdb;
$wpdb->query("DROP TABLE IF EXISTS {$wpdb->prefix}mytable");
```

注意：在多站点中，通过所有博客循环删除选项可能非常耗资源。

# 最佳做法

以下是一些最佳做法来帮助组织您的代码，使其与WordPress核心和其他WordPress插件一起工作良好。

## 避免命名冲突

当您的插件与变量，函数或类作为另一个插件使用相同的名称时，会发生命名冲突。

幸运的是，您可以通过使用以下方法避免命名冲突。

## 程序

默认情况下，所有变量，函数和类都在全局命名空间中定义，这意味着插件可以覆盖由另一个插件设置的变量，函数和类，反之亦然。在函数或类中定义的变量不受此影响。

## 前缀一切

所有变量，函数和类应以唯一标识符为前缀。前缀可防止其他插件覆盖变量，并意外调用您的函数和类。它也会阻止你做同样的事情。

## 检查现有的实现

PHP提供了许多函数来验证变量，函数，类和常量的存在。如果实体存在，所有这些将返回true。

- Variables: `isset()` (includes arrays, objects, etc.)
- Functions: `function_exists()`
- Classes: `class_exists()`
- Constants: `defined()`

## 示例

```
<?php
//Create a function called "wporg_init" if it doesn't already exist
if ( !function_exists( 'wporg_init' ) ) {
    function wporg_init() {
        register_setting( 'wporg_settings', 'wporg_option_foo' );
    }
}

//Create a function called "wporg_get_foo" if it doesn't already exist
if ( !function_exists( 'wporg_get_foo' ) ) {
    function wporg_get_foo() {
        return get_option( 'wporg_option_foo' );
    }
}
```

## OOP

解决命名冲突问题的一个更简单的方法是使用一个类来代替你的插件。

您仍然需要注意检查您想要的类的名称是否已经被使用，其余的将由PHP来处理。

## 示例

```
<?php
if ( !class_exists( 'WPOrg_Plugin' ) ) {
    class WPOrg_Plugin
    {
        public static function init() {
            register_setting( 'wporg_settings', 'wporg_option_foo' );
        }

        public static function get_foo() {
            return get_option( 'wporg_option_foo' );
        }
    }

    WPOrg_Plugin::init();
    WPOrg_Plugin::get_foo();
}
```

## 文件组织

您的插件目录的根级别应该包含你的plugin-name.php文件，并且可以选择你的uninstall.php文件。所有其他文件应尽可能地组织成子文件夹。

## 文件夹结构

一个清晰的文件夹结构可以帮助您和其他在您的插件上工作的人保持类似的文件

以下是一个示例文件夹结构供参考：

```
/plugin-name
  plugin-name.php
  uninstall.php
  /languages
  /includes
  /admin
    /js
    /css
    /images
  /public
    /js
```

```
/css
/images
```

## 插件架构

您为插件选择的架构或代码组织可能取决于您的插件的大小。

对于与WordPress核心，主题或其他插件进行有限交互的小型单用途插件，在复杂类工程中几乎没有什么好处；除非你知道这个插件会在以后扩大很多。

对于具有大量代码的大型插件，请注意课程。独立的样式和脚本文件，甚至与构建相关的文件。这将有助于代码组织和长期维护的插件。

## 有条件加载

将管理员代码与公共代码分开是有帮助的。使用条件`is_admin()`。

例如：

```
<?php
if ( is_admin() ) {
    // we are in admin mode
    require_once( dirname( __FILE__ ) . '/admin/plugin-name-admin.php' );
}
```

## 建筑模式

虽然有许多可能的架构模式，但它们可以广泛地分为三个变体：

- 单个插件文件，包含功能
- 单个插件文件，包含类，实例化对象和可选功能
- 主要插件文件，然后一个或多个类文件

## 架构模式解释

上述代码组织的更复杂的具体实现已经写成教程和幻灯片：

- 斜杠 - 单身人士，装载机，动作，屏幕，处理程序
- MVC启发式WordPress插件开发方法
- 在WordPress插件中实现MVC模式

## 锅炉起点

而不是从头开始为每个你写的新插件，你可能想从一个样板开始。使用样板的一个优点是在您自己的插件之间保持一致。如果您使用他们已经熟悉的样板，其他人员也可以轻松地为他们提供代码。

这些也作为不同但可比较的架构的进一步示例。

- WordPress插件Boilerplate：WordPress插件开发的基础，旨在为构建您的插件提供清晰一致的指南。
- WordPress插件引导：使用Grunt，Compass，GIT和SVN开发WordPress插件的基本引导。
- WP骨架插件：骷髅插件，专注于单元测试和使用作曲家进行开发。

一般在GitHub上搜索WordPress插件当然，您可以采取这些和其他方面的不同方面来创建自己的自定义样板。

# 插件安全

---

恭喜您，您的代码正常工作！但它是安全的吗？如果您的网站遭到黑客攻击，该插件将如何保护您的用户？

WordPress.org目录中最好的插件可以保护用户的信息安全。

请记住，您的代码可能会在数百甚至数百万的网站上运行，因此安全性至关重要。

在本章中，我们将介绍如何检查用户功能，验证和消毒输入，清理输出并创建和验证随机数。

## 快速参考

---

请参阅WordPress插件和主题的安全最佳做法的完整示例。

## 外部资源

---

如何修复Jon Cave的故意易受攻击的插件

主题和插件安全演示由马克Jaquith

# 检查用户功能

如果您的插件允许用户在管理员或公共方面提交数据，则应检查用户功能。

## ##用户角色和功能

创建高效安全层最重要的一步就是拥有一个用户权限系统。WordPress以用户角色和功能的形式提供。

每个登录WordPress的用户都会根据用户角色自动分配具体的用户功能。

用户角色只是说明用户所属的组的一种奇特的方式。每个组都有一组特定的预定义功能。

例如，您的网站的主要用户将具有管理员的用户角色，而其他用户可能具有编辑器或作者等角色。您可以将多个用户分配到角色，即网站可能有两个管理员。

用户功能是您分配给每个用户或用户角色的特定权限。

例如，管理员具有“manage\_options”功能，允许他们查看，编辑和保存网站的选项。另一方面，编辑者缺乏这种能力，这将阻止他们与选项进行交互。

然后在管理员的各个不同点检查这些功能。取决于分配给角色的功能;可以添加或删除WordPress体验的菜单，功能和其他方面。

在构建插件时，请确保仅在当前用户具有必要功能时运行代码。

## 层次结构

用户角色越高，用户拥有的功能越多。每个用户角色都会继承层次结构中的以前的角色。

例如，单个站点安装中最高用户角色的“管理员”继承了以下角色及其功能：“订阅者”，“贡献者”，“作者”和“编辑者”。

## 无限制

以下示例创建了一个前端的链接，该链接能够删除帖子。由于此代码不检查用户功能，它允许任何访问者的站点垃圾邮件！

```
<?php
/**
 * generate a Delete link based on the homepage url
 */
function wporg_generate_delete_link($content)
{
    // run only for single post page
    if (is_single() && in_the_loop() && is_main_query()) {
        // add query arguments: action, post
        $url = add_query_arg(
            [
                'action' => 'wporg_frontend_delete',
                'post'    => get_the_ID(),
            ],
            home_url()
        );
    }
}
```

```

    );
    return $content . ' <a href="' . esc_url($url) . '">' . esc_html__( 'Delete Post', 'wporg' ) . '</a>';
}
return null;
}

/**
 * request handler
 */
function wporg_delete_post()
{
    if (isset($_GET['action']) && $_GET['action'] === 'wporg_frontend_delete') {

        // verify we have a post id
        $post_id = (isset($_GET['post'])) ? ($_GET['post']) : (null);

        // verify there is a post with such a number
        $post = get_post((int)$post_id);
        if (empty($post)) {
            return;
        }

        // delete the post
        wp_trash_post($post_id);

        // redirect to admin page
        $redirect = admin_url('edit.php');
        wp_safe_redirect($redirect);

        // we are done
        die;
    }
}

/**
 * add the delete link to the end of the post content
 */
add_filter('the_content', 'wporg_generate_delete_link');

/**
 * register our request handler with the init hook
 */
add_action('init', 'wporg_delete_post');

```

## 限于具体能力



上面的示例允许网站的任何访问者点击“删除”链接并垃圾邮件。但是，我们只希望Editors及以上版本能够点击“删除”链接。

为了实现这一点，我们将检查当前用户是否具有edit\_others\_posts功能，只有Editors或以上版本具有：

```
<?php
/**
 * generate a Delete link based on the homepage url
 */
function wporg_generate_delete_link($content)
{
    // run only for single post page
    if (is_single() && in_the_loop() && is_main_query()) {
        // add query arguments: action, post
        $url = add_query_arg(
            [
                'action' => 'wporg_frontend_delete',
                'post'    => get_the_ID(),
            ],
            home_url()
        );
        return $content . ' <a href="' . esc_url($url) . '"> . esc_html__( 'Delete Post', 'wporg') . '</a>';
    }
    return null;
}

/**
 * request handler
 */
function wporg_delete_post()
{
    if (isset($_GET['action']) && $_GET['action'] === 'wporg_frontend_delete') {

        // verify we have a post id
        $post_id = (isset($_GET['post'])) ? ($_GET['post']) : (null);

        // verify there is a post with such a number
        $post = get_post((int)$post_id);
        if (empty($post)) {
            return;
        }

        // delete the post
        wp_trash_post($post_id);

        // redirect to admin page
        $redirect = admin_url('edit.php');
```

```
        wp_safe_redirect($redirect);

        // we are done
        die;
    }
}

if (current_user_can('edit_others_posts')) {
    /**
     * add the delete link to the end of the post content
     */
    add_filter('the_content', 'wporg_generate_delete_link');

    /**
     * register our request handler with the init hook
     */
    add_action('init', 'wporg_delete_post');
}
```

# 数据验证

数据验证是根据具有确定结果的预定义模式（或模式）分析数据的过程：有效或无效。

通常这适用于来自外部来源的数据，例如用户输入和通过API调用Web服务。

数据验证的简单示例：

- 检查所需字段未留空
- 检查输入的电话号码只包含数字和标点符号
- 检查输入的邮政编码是否是有效的邮政编码
- 检查数量字段是否大于0
- 数据验证应尽早进行。这意味着在执行任何操作之前验证数据。

**注意：**验证可以通过使用前端的JavaScript和后端使用PHP来执行。

## 验证数据

至少有三种方式：内置PHP函数，核心WordPress函数和您编写的自定义函数。

## 内置PHP功能

使用许多内置的PHP函数进行基本验证，包括：

- `isset ( )` 和 `empty ( )` 用于检查变量是否存在且不为空
- `mb_strlen ( )` 或 `strlen ( )` 用于检查字符串是否具有预期的字符数
- `preg_match ( )` , `strpos ( )` 用于检查其他字符串中某些字符串的出现情况
- `count ( )` 用于检查数组中有多少项
- `in_array ( )` 用于检查数组中是否存在某些内容

## 核心WordPress功能

WordPress提供了许多有用的功能，有助于验证不同类型的数据。以下是几个例子：

- `is_email ( )` 将验证电子邮件地址是否有效。
- `term_exists ( )` 检查是否存在标签，类别或其他分类术语。
- `username_exists ( )` 检查用户名是否存在。
- `validate_file ( )` 将验证输入的文件路径是否为真实路径（但不是文件是否存在）。

检查WordPress代码参考以获得更多类似的功能。搜索具有以下名称的函数：`*_exists ( )` , `* validate ( )` , 并且是 `*` ( ) 。并非所有这些都是验证功能，但许多都是有幫助的。

## 定制PHP和JavaScript函数

您可以编写自己的PHP和JavaScript函数，并将它们包含在您的插件中。编写验证函数时，您需要将其命名为一个问题（例如：is\_phone，is\_available，is\_us\_zipcode）。

该函数应该返回一个布尔值（true或false），具体取决于数据是否有效。这将允许使用该功能作为条件。

### 示例1

假设您有一个用户提交的美国邮政编码输入字段。

```
<input id="wporg_zip_code" type="text" maxlength="10" name="wporg_zip_code">
```

文本字段允许最多10个字符的输入，对可以使用的字符类型没有限制。用户可以输入一些有效的东西，如1234567890，或者像eval（）一样无效（和邪恶）。

我们输入域中的maxlength属性仅由浏览器强制执行，因此您仍然需要验证服务器上输入的长度。如果不这样做，攻击者可以改变maxlength值。

通过使用验证，我们可以确保我们只接受有效的邮政编码。

首先，您需要编写一个功能来验证美国邮政编码：

```
<?php
function is_us_zip_code($zip_code)
{
    // scenario 1: empty
    if (empty($zip_code)) {
        return false;
    }

    // scenario 2: more than 10 characters
    if (strlen(trim($zip_code)) > 10) {
        return false;
    }

    // scenario 3: incorrect format
    if (!preg_match('/^\d{5}(\-\d{4})?$/ ', $zip_code)) {
        return false;
    }

    // passed successfully
    return true;
}
```

处理表单时，您的代码应检查wporg\_zip\_code字段，并根据结果执行操作：

```
if (isset($_POST['wporg_zip_code']) && is_us_zip_code($_POST['wporg_zip_code']))  
{  
    // your action  
}
```

## 例2

假设您要查询数据库中的某些帖子，并希望让用户对查询结果进行排序。

该示例代码通过使用内置的PHP函数in\_array将允许的排序键的数组进行比较来检查输入的排序键（存储在“orderby”输入参数中）的有效性。这样可以防止用户传递恶意数据并潜在地损害网站。

在对数组检查传入排序键之前，将密钥传递到内置的WordPress功能sanitize\_key。此功能确保了键是小写（in\_array执行区分大小写的搜索）。

将“true”传递给in\_array的第三个参数可以进行严格的类型检查，这样就可以使功能不仅可以比较值和值类型。这允许代码确定输入的排序键是字符串，而不是其他数据类型。

```
<?php  
$allowed_keys = ['author', 'post_author', 'date', 'post_date'];  
  
$orderby = sanitize_key($_POST['orderby']);  
  
if (in_array($orderby, $allowed_keys, true)) {  
    // modify the query to sort by the orderby key  
}
```

# 保护输入

保护输入是消毒（清理，过滤）输入数据的过程。

当您不知道期望或不想严格的数据验证时，您可以使用清洁。

任何时候您接受潜在的不安全数据，重要的是验证或清除它。

## 消毒数据

消除数据的最简单方法是使用内置的WordPress功能。

消毒的 `_*` ( ) 系列助手功能是非常好的，因为它们确保您的安全数据结束，并且您需要尽可能少的努力：

- `sanitize_email()`
- `sanitize_file_name()`
- `sanitize_html_class()`
- `sanitize_key()`
- `sanitize_meta()`
- `sanitize_mime_type()`
- `sanitize_option()`
- `sanitize_sql_orderby()`
- `sanitize_text_field()`
- `sanitize_title()`
- `sanitize_title_for_query()`
- `sanitize_title_with_dashes()`
- `sanitize_user()`
- `esc_url_raw()`
- `wp_filter_post_kses()`
- `wp_filter_nohtml_kses()`

## 示例

假设我们有一个名为title的输入字段。

```
<input id="title" type="text" name="title">
```

您可以使用`sanitize_text_field` ( ) 函数来清理输入数据：

```
$title = sanitize_text_field($_POST['title']);  
update_post_meta($post->ID, 'title', $title);
```

在幕后，`sanitize_text_field ( )` 执行以下操作：

- 检查无效的UTF-8
- 将单个小于字符 ( < ) 转换为实体
- 贴上所有标签
- 删除换行符，标签和额外的空格条字节

# 保护输出

保护输出是转义输出数据的过程。

转义意味着剥离不必要的数据，如格式不正确的HTML或脚本标签。

无论何时渲染数据，请确保正确地释放数据。 转义输出可以防止XSS（跨站点脚本）攻击。

**注意：**跨站点脚本（XSS）是通常在Web应用程序中发现的一种计算机安全漏洞。XSS使攻击者将客户端脚本注入由其他用户查看的网页。攻击者可能会使用跨站点脚本漏洞绕过诸如同源策略的访问控制。

## 逃避

转义有助于在为最终用户呈现数据之前保护您的数据。WordPress有一些帮助功能，您可以用于大多数常见的场景。

- `esc_html ( )` - 当HTML元素包含显示的数据部分时，可以使用此功能。
- `esc_url ( )` - 对所有URL使用此功能，包括HTML元素的src和href属性中的URL。
- `esc_js ( )` - 将此函数用于内联JavaScript。
- `esc_attr ( )` - 将此函数用于打印到HTML元素属性中的所有内容。

**警报：**大多数WordPress功能正确准备数据进行输出，因此您不需要再次转义数据。例如，您可以安全地调用`the_title ( )`而不转义。

## 随着本地化逃脱

而不是使用echo来输出数据，通常使用WordPress本地化函数，如 `_e ( )` 或 `__ ( )` 。

这些函数只是将定位函数包含在转义函数中：

```
esc_html_e( 'Hello World', 'text_domain' );  
// same as  
echo esc_html( __( 'Hello World', 'text_domain' ) );
```

这些帮助函数结合本地化和转义：

- `esc_html__()`
- `esc_html_e()`
- `esc_html_x()`
- `esc_attr__()`
- `esc_attr_e()`
- `esc_attr_x()`



## 自定义转义

在需要以特定方式转义输出的情况下，函数wp\_kses（）（发音为“kisses”）将派上用场。

此功能确保只有指定的HTML元素，属性和属性值才会在输出中出现，并对HTML实体进行规范化。

```
$allowed_html = [  
    'a'          => [  
        'href'   => [],  
        'title'  => [],  
    ],  
    'br'         => [],  
    'em'         => [],  
    'strong'     => [],  
];  
echo wp_kses( $custom_content, $allowed_html );
```

wp\_kses\_post（）是wp\_kses的包装函数，其中\$ allowed\_html是一组由帖子内容使用的规则。

```
echo wp_kses_post( $post_content );
```

# 随机数

为了安全起见，生成用于验证请求的起源和意图的生成数字。每个随机数只能使用一次。

如果您的插件允许用户提交数据; 无论是在行政部门还是公共方面; 您必须确保用户是他们所说的，他们具有执行操作的必要功能。同时进行这两项意味着数据只会在用户期望更改时发生变化。

## ##使用Nonces

按照我们检查用户功能的示例，用户数据提交安全性的下一步是使用随机数。

功能检查确保只有具有删除帖子权限的用户才能删除帖子。但是如果有人欺骗你点击那个链接呢？你有必要的能力，所以你可以不经意地删除一个职位。

可以使用Nonces来检查当前用户是否实际打算执行该操作。

当您生成删除链接时，您将需要使用`wp_create_nonce()`函数将一个随机数添加到链接中，传递给该函数的参数可确保正在创建的随机数对该特定操作是唯一的。

然后，当您处理删除链接的请求时，可以检查该随机数是您期望的内容。

有关更多信息，马克·贾奇斯 ( James Jaquith ) 在WordPress中的发布是一个很好的资源。

## ##完整例子

使用功能检查，数据验证，安全输入，安全输出和随机数的完整示例：

```
<?php
/**
 * generate a Delete link based on the homepage url
 */
function wporg_generate_delete_link($content)
{
    // run only for single post page
    if (is_single() && in_the_loop() && is_main_query()) {
        // add query arguments: action, post, nonce
        $url = add_query_arg(
            [
                'action' => 'wporg_frontend_delete',
                'post'    => get_the_ID(),
                'nonce'   => wp_create_nonce('wporg_frontend_delete'),
            ],
            home_url()
        );
        return $content . ' <a href="' . esc_url($url) . '"> . esc_html__( 'Delete Post', 'wporg' ) . '</a>';
    }
    return null;
}

/**
 * request handler
```

```
*/  
function wporg_delete_post()  
{  
    if (  
        isset($_GET['action']) &&  
        isset($_GET['nonce']) &&  
        $_GET['action'] === 'wporg_frontend_delete' &&  
        wp_verify_nonce($_GET['nonce'], 'wporg_frontend_delete')  
    ) {  
  
        // verify we have a post id  
        $post_id = (isset($_GET['post'])) ? ($_GET['post']) : (null);  
  
        // verify there is a post with such a number  
        $post = get_post((int)$post_id);  
        if (empty($post)) {  
            return;  
        }  
  
        // delete the post  
        wp_trash_post($post_id);  
  
        // redirect to admin page  
        $redirect = admin_url('edit.php');  
        wp_safe_redirect($redirect);  
  
        // we are done  
        die;  
    }  
}  
  
if (current_user_can('edit_others_posts')) {  
    /**  
     * add the delete link to the end of the post content  
     */  
    add_filter('the_content', 'wporg_generate_delete_link');  
  
    /**  
     * register our request handler with the init hook  
     */  
    add_action('init', 'wporg_delete_post');  
}
```

# Hooks

---

钩子是一段代码来交互/修改另一段代码的方式。它们构成了插件和主题与WordPress Core交互的基础，但它们也被Core本身广泛使用。

钩子有两种类型：动作和过滤器。要使用它们，您需要编写一个称为回调的自定义函数，然后将其注册到特定Action或Filter的WordPress钩子。

操作允许您添加数据或更改WordPress的操作方式。Action的回调函数将在执行WordPress的特定点运行，并且可以执行某种任务，例如向用户回显输出或将数据插入到数据库中。

过滤器可让您在执行WordPress期间更改数据。过滤器的回调函数将接受变量，修改它并返回。它们的意图是以孤立的方式工作，不应该有影响全局变量和产出的副作用。

WordPress提供了许多可以使用的钩子，但您也可以创建自己的钩子，以便其他开发人员可以扩展和修改您的插件或主题。

## 外部资源

---

[过滤器参考](#)

[行动参考](#)

[亚当·布朗的数据库](#)

[操作和过滤器不一样](#)

# Actions

Action是Hooks的两种类型之一。

它们提供了在执行WordPress Core，插件和主题的特定运行功能的方法。 它们是Filter的对应物。

## ##添加Action

添加Action的过程包括两个步骤。

首先，您需要创建一个回调函数，该函数在运行时将被调用。 其次，您需要将Callback函数添加到一个将执行该函数调用的钩子中。

您将使用add\_action ( ) 函数，传递至少两个参数string \$ tag，callable \$ function\_to\_add。

下面的例子将在执行init钩子时运行：

```
<?php
function wporg_custom()
{
    // do something
}
add_action('init', 'wporg_custom');
```

您可以参考Hooks章节了解可用挂钩的列表。

当您获得更多的经验，通过WordPress核心源代码将允许您找到最合适的钩子。

## 附加参数

add\_action ( ) 可以接受两个附加参数，int \$ priority用于给予回调函数的优先级，int \$ accepted\_args表示将传递给回调函数的参数数。

## 优先

优先级决定了与给定钩子相关联的其他回调函数相关的回调函数何时执行。

优先级为11的功能将以优先级为10的功能运行; 并且优先级为9的函数将在优先级为10的函数之前运行。任何正整数是可接受的值，默认值为10。

如果两个回调函数以相同的优先级为同一个钩子注册，那么将按照它们挂钩的顺序运行。

例如，以下回调函数都注册到

init钩子，但具有不同的优先级：

```
<?php
add_action('init', 'run_me_early', 9);
add_action('init', 'run_me_normal'); // default value of 10 is used since a
priority wasn't specified
add_action('init', 'run_me_late', 11);
```

运行的第一个函数是run\_me\_early ( )，其次是run\_me\_normal ( )，最后运行的函数将是run\_me\_late ( )。

## 参数数量

有时候，回调函数需要接收一些与其挂接的函数相关的额外数据。

例如，当WordPress保存一个帖子并运行save\_post钩子时，它会将两个参数传递给回调函数：保存的帖子的ID以及post对象本身：

```
do_action('save_post', $post->ID, $post);
```

所以，当save\_post钩子注册一个回调函数时，它可以指定它要接收这两个参数：

```
add_action('save_post', 'wporg_custom', 10, 2);
```

...然后它可以在函数定义中注册参数：

```
function wporg_custom($post_id, $post)
{
    // do something
}
```

## 示例

如果您想修改在前端的循环中获取搜索结果的查询，则可以钩入pre\_get\_posts钩子。

```
<?php
function wporg_search($query)
{
    if (!is_admin() && $query->is_main_query() && $query->is_search) {
        $query->set('post_type', ['post', 'movie']);
    }
}
add_action('pre_get_posts', 'wporg_search');
```

# Filters

Filter是Hooks的两种类型之一。

它们为功能修改其他功能的数据提供了一种方法。他们是Action的对手。

与Action不同，Filter旨在以孤立的方式工作，并且不应该有副作用，例如影响全局变量和输出。

## 添加 Filter

添加过滤器的过程包括两个步骤。

首先，您需要创建一个回调函数，该函数在运行过滤器时将被调用。其次，您需要将Callback函数添加到一个将执行该函数调用的钩子中。

您将使用add\_filter ( ) 函数，传递至少两个参数string \$ tag , callable \$ function\_to\_add。

以下示例将在执行the\_title过滤器时运行。

```
<?php
function wporg_filter_title($title)
{
    return 'The ' . $title . ' was filtered';
}
add_filter('the_title', 'wporg_filter_title');
```

说我们有一个帖子标题“学习WordPress”，上面的例子将修改为“学习WordPress被过滤”。

您可以参考Hooks章节了解可用挂钩的列表。

当您获得更多的经验，通过WordPress核心源代码将允许您找到最合适的钩子。

## 附加参数

add\_filter ( ) 可以接受两个附加参数，int \$ priority用于给予回调函数的优先级，int \$ accepted\_args表示将传递给回调函数的参数数。

有关这些参数的详细说明，请阅读有关操作的文章。

示例

```
<?php
function wporg_css_body_class($classes)
{
    if (!is_admin()) {
        $classes[] = 'wporg-is-awesome';
    }
    return $classes;
}
add_filter('body_class', 'wporg_css_body_class');
```





# 自定义Hooks

一个重要但经常被忽视的做法是在您的插件中使用自定义钩子，以便其他开发人员可以扩展和修改它。自定义钩子以与WordPress Core钩子相同的方式创建和调用。

## 创建一个钩子

要创建自定义钩子，请对操作使用`do_action()`，对过滤器应用`apply_filters()`。

**注意：**我们建议在输出到浏览器的任何文本上使用`apply_filters()`。特别是在前台。

这样可以根据用户的需要更容易地修改插件。

## 向Hook添加回调

要将一个回调函数添加到自定义钩子中，请对“过滤器”使用“动作”的`add_action()`和`add_filter()`。

## 命名冲突

由于任何插件都可以创建一个自定义钩子，所以重要的是为您的挂钩名称前缀以避免与其他插件的冲突。

例如，名为`email_body`的过滤器将不那么有用，因为另一个开发人员可能会选择相同的名称。如果用户安装这两个插件，可能会导致难以追踪的错误。

命名功能`wporg_email_body`（其中`wporg`是您的插件的唯一前缀）将避免任何冲突。

# 示例

## 可扩展操作：设置表单

如果您的插件向管理面板添加了一个设置表单，您可以使用操作来允许其他插件添加自己的设置。

```
<?php
function wporg_settings_page_html()
{
    ?>
    Foo: <input id="foo" name="foo" type="text">
    Bar: <input id="bar" name="bar" type="text">
    <?php
    do_action('wporg_after_settings_page_html');
}
```

现在另一个插件可以为`wporg_after_settings_page_html`钩子注册一个回调函数，并注入新的设置：

```
<?php
function myprefix_add_settings()
{
    ?>
    New 1: <input id="new_setting" name="new_settings" type="text">
    <?php
}
add_action('wporg_after_settings_page_html', 'myprefix_add_settings');
```

## 可扩展过滤器：自定义文章类型

在此示例中，当注册新的帖子类型时，定义它的参数通过过滤器传递，因此另一个插件可以在创建帖子类型之前更改它们。

```
<?php
function wporg_create_post_type()
{
    $post_type_params = [/* ... */];

    register_post_type(
        'post_type_slug',
        apply_filters('wporg_post_type_params', $post_type_params)
    );
}
```

现在另一个插件可以为wporg\_post\_type\_params钩子注册回调函数，并更改帖子类型参数：

```
<?php
function myprefix_change_post_type_params($post_type_params)
{
    $post_type_params['hierarchical'] = true;
    return $post_type_params;
}
add_filter('wporg_post_type_params', 'myprefix_change_post_type_params');
```

## 外部资源

- Extendable Extensions by Michael Fields
- WordPress Plugins as Frameworks by Josh Harrison
- The Pluggable Plugin by Brandon Dove
- WordPress Plugin Pet Peeves #3: Not Being Extensible by Will Norris

# 高级主题

## 删除操作和过滤器

有时您想要从另一个插件，主题甚至WordPress Core已注册的钩子中删除回调函数。

要从挂钩中删除回调函数，您需要调用`remove_action()`或`remove_filter()`，这取决于回调函数是作为Action还是Filter来添加。

传递给`remove_action()` / `remove_filter()`的参数应与传递给注册它的`add_action()` / `add_filter()`的参数相同。

**警报：**要成功删除回调函数，您必须在注册回调函数后执行删除。 执行顺序很重要。

### 示例

我们希望通过删除不必要的功能来改善大型主题的性能。

我们通过查看`functions.php`来分析主题的代码。

```
<?php
function my_theme_setup_slider()
{
    // ...
}
add_action('template_redirect', 'my_theme_setup_slider', 9);
```

`my_theme_setup_slider`函数正在添加一个我们不需要的滑块，这可能会加载一个巨大的CSS文件，然后是一个JavaScript初始化文件，它使用大小为1MB的自定义书写库。我们可以摆脱这一点。

因为我们希望在注册`my_theme_setup_slider`回调函数（`functions.php`执行）之后挂接到WordPress中，所以最好的机会是`after_setup_theme`钩子。

```
<?php
function wporg_disable_slider()
{
    // make sure all parameters match the add_action() call exactly
    remove_action('template_redirect', 'my_theme_setup_slider', 9);
}
// make sure we call remove_action() after add_action() has been called
add_action('after_setup_theme', 'wporg_disable_slider');
```

### ##删除所有回调

您也可以使用`remove_all_actions()` / `remove_all_filters()`来删除与钩子相关联的所有回调函数。

确定当前挂钩

有时您想要在多个钩子上运行一个Action或Filter，但是根据当前调用它的行为有所不同。

您可以使用`current_action()` / `current_filter()` 来确定当前的Action / Filter。

```
<?php
function wporg_modify_content($content)
{
    switch (current_filter()) {
        case 'the_content':
            // do something
            break;
        case 'the_excerpt':
            // do something
            break;
    }
    return $content;
}
add_filter('the_content', 'wporg_modify_content');
add_filter('the_excerpt', 'wporg_modify_content');
```

## 检查一个钩子有多少次运行

一些钩子在执行过程中被多次调用，但您可能只希望您的回调函数运行一次。

在这种情况下，您可以使用`did_action()` 检查钩子运行的次数。

```
<?php
function wporg_custom()
{
    if (did_action('save_post') !== 1) {
        return;
    }
    // ...
}
add_action('save_post', 'wporg_custom');
```

## 用 “all” 钩子调试

如果你想要一个回调函数在每个钩子上触发，你可以注册到所有的钩子。有时，这在调试情况有助于确定特定事件何时发生或页面崩溃时有用。

```
<?php
function wporg_debug()
{
    echo '<p>' . current_action() . '</p>';
}
add_action('all', 'wporg_debug');
```



# 管理菜单

管理菜单是WordPress管理中显示的界面。它们允许您为插件添加选项页。

**注意：**有关管理导航菜单的信息，请参阅主题开发者手册的导航菜单一章。

## 顶级菜单和子菜单

顶级菜单呈现在WordPress管理的左侧。每个菜单可能包含一组子菜单。

在顶级菜单和子菜单之间决定仔细考虑插件的需求以及最终用户的需求。

**警报：**我们建议开发人员使用单个选项页面将其作为子菜单添加到现有的顶级菜单之一；如设置或工具。

# 顶级菜单

## 添加顶级菜单

要向WordPress管理添加新的顶级菜单，请使用add\_menu\_page ( ) 函数。

```
<?php
add_menu_page(
    string $page_title,
    string $menu_title,
    string $capability,
    string $menu_slug,
    callable $function = '',
    string $icon_url = '',
    int $position = null
);
```

## 示例

假设我们要添加一个名为 “WPOrg” 的顶级菜单。

第一步将创建一个将输出HTML的函数。在此功能中，我们将执行必要的安全检查，并使用Settings API呈现我们注册的选项。

注意：我们建议使用一个包裹类型的 “ ” 包装你的HTML。

```
<?php
function wporg_options_page_html()
{
    // check user capabilities
    if (!current_user_can('manage_options')) {
        return;
    }
    ?>
    <div class="wrap">
        <h1><?= esc_html(get_admin_page_title()); ?></h1>
        <form action="options.php" method="post">
            <?php
                // output security fields for the registered setting "wporg_options"

                settings_fields('wporg_options');
                // output setting sections and their fields
                // (sections are registered for "wporg", each field is registered to
                // a specific section)
                do_settings_sections('wporg');
```

```
// output save settings button
submit_button('Save Settings');
?>
</form>
</div>
<?php
}
```

第二步将注册我们的WPOrg菜单。注册需要在admin\_menu操作钩子中进行。

```
<?php
function wporg_options_page()
{
    add_menu_page(
        'WPOrg',
        'WPOrg Options',
        'manage_options',
        'wporg',
        'wporg_options_page_html',
        plugin_dir_url(__FILE__) . 'images/icon_wporg.png',
        20
    );
}
add_action('admin_menu', 'wporg_options_page');
```

有关参数列表，请参阅引用中的add\_menu\_page（）。

## 使用PHP文件进行HTML

便携式代码的最佳做法是创建一个需要/包含您的PHP文件的回调。

为了完整性和帮助您了解遗留代码，我们将展示另一种方法：将PHP文件路径作为\$ menu\_slug参数传递一个null \$ function参数。

```
<?php
function wporg_options_page()
{
    add_menu_page(
        'WPOrg',
        'WPOrg Options',
        'manage_options',
        plugin_dir_path(__FILE__) . 'admin/view.php',
        null,
        plugin_dir_url(__FILE__) . 'images/icon_wporg.png',
        20
    );
}
```



```
add_action('admin_menu', 'wporg_options_page');
```

## 删除顶级菜单

要从WordPress管理中删除注册的菜单，请使用remove\_menu\_page ( ) 函数。

```
<?php
remove_menu_page(
    string $menu_slug
);
```

**警告：**删除菜单不会阻止用户直接访问它们。这绝对不能用来限制用户的功能。

## 示例

让我们说要从中删除 “工具” 菜单。

```
<?php
function wporg_remove_options_page()
{
    remove_menu_page('tools.php');
}
add_action('admin_menu', 'wporg_remove_options_page', 99);
```

在尝试删除之前，请确保已将菜单注册到admin\_menu钩子，为add\_action ( ) 指定较高优先级号。

# 子菜单

## 添加子菜单

要添加新的子菜单到WordPress管理，请使用add\_submenu\_page ( ) 函数。

```
<?php
add_submenu_page(
    string $parent_slug,
    string $page_title,
    string $menu_title,
    string $capability,
    string $menu_slug,
    callable $function = ''
);
```

## 示例

假设我们要添加子菜单 “WPOrg选项” 到 “工具” 顶级菜单。

第一步将创建一个将输出HTML的函数。在此功能中，我们将执行必要的安全检查，并使用Settings API呈现我们注册的选项。

**注意：**我们建议使用一个包裹类型的 `<div>` 包装你的HTML。

```
<?php
function wporg_options_page_html()
{
    // check user capabilities
    if (!current_user_can('manage_options')) {
        return;
    }
    ?>
    <div class="wrap">
        <h1><?= esc_html(get_admin_page_title()); ?></h1>
        <form action="options.php" method="post">
            <?php
                // output security fields for the registered setting "wporg_options"

                settings_fields('wporg_options');
                // output setting sections and their fields
                // (sections are registered for "wporg", each field is registered t
o a specific section)
                do_settings_sections('wporg');
                // output save settings button
                submit_button('Save Settings');
```

```

        ?>
    </form>
</div>
<?php
}

```

第二步将注册我们的WPOrg选项子菜单。注册需要在admin\_menu操作钩子中进行。

```

<?php
function wporg_options_page()
{
    add_submenu_page(
        'tools.php',
        'WPOrg Options',
        'WPOrg Options',
        'manage_options',
        'wporg',
        'wporg_options_page_html'
    );
}
add_action('admin_menu', 'wporg_options_page');

```

有关参数列表，请参阅引用中的add\_submenu\_page ( )。

## ##自定义子菜单

如果我们有帮助函数来定义WordPress内置顶级菜单的\$ parent\_slug，并保存我们，通过源代码手动搜索，那不是很好吗？

以下是父slug a及其助手功能列表：

- add\_dashboard\_page() – index.php
- add\_posts\_page() – edit.php
- add\_media\_page() – upload.php
- add\_pages\_page() – edit.php?post\_type=page
- add\_comments\_page() – edit-comments.php
- add\_theme\_page() – themes.php
- add\_plugins\_page() – plugins.php
- add\_users\_page() – users.php
- add\_management\_page() – tools.php
- add\_options\_page() – options-general.php
- add\_options\_page() – settings.php
- add\_links\_page() – link-manager.php – requires a plugin since WP 3.5+

## 子菜单

- Custom Post Type – `edit.php?post_type=wporg_post_type`
- Network Admin – `settings.php`

## 删除子菜单

---

删除子菜单的过程与删除顶级菜单完全相同。

# 短代码

---

作为安全防范措施，禁止在WordPress内容中运行PHP;为了允许与内容的动态交互，短消息在WordPress版本2.5中呈现。

短代码是可用于与内容进行动态交互的宏。即从附加到帖子或呈现视频的图像创建图库。

## 为什么是短码？

---

短信是保持内容清洁和语义的一种有价值的方式，同时允许最终用户一些能够以编程方式改变其内容的呈现的能力。

当最终用户使用短代码将照片库添加到他们的帖子时，他们使用尽可能少的数据来指示如何呈现图库。

优点：

- 没有标记添加到帖子内容，这意味着标记和样式可以很容易地在飞行或稍后的状态下被操纵。
- 短码也可以接受参数，允许用户修改一个实例上的短码行为。

## 内置短码

---

默认情况下，WordPress包括以下短节码：

- [caption] – 简短的代码，允许您围绕内容包装字幕
- [gallery] – 短码，可让您显示图像画廊
- [audio] –短码，允许您嵌入和播放音频文件
- [video] – 短码可以让您嵌入和播放视频文件
- [playlist] – 短码可让您显示音频或视频文件的集合
- [embed] – 短代码，允许您封装嵌入的项目

## 短码最佳实践

---

开发短代码的最佳做法包括插件开发最佳实践和下面的列表：

- 总是回来
- 短码本质上是过滤器，因此创建“副作用”将导致意外的错误。
- 前缀您的短代码名称，以避免与其他插件的冲突。
- 消除输入并转义输出。
- 为用户提供有关所有短代码属性的清晰文档。

## 快速参考

---

请参阅使用基本的短代码结构，照顾自我关闭和封闭情景，快捷代码和确保输出的完整示例。

## 外部资源

---

- [WordPress短码生成器](#)

# 基本短码

## ##添加一个短码

可以使用Shortcode API添加自己的短码。该过程包括使用add\_shortcode ( ) 将一个回调\$ func注册到一个shortcode \$标签。

```
<?php
add_shortcode(
    string $tag,
    callable $func
);
```

## 在一个主题

```
<?php
function wporg_shortcode($atts = [], $content = null)
{
    // do something to $content

    // always return
    return $content;
}
add_shortcode('wporg', 'wporg_shortcode');
```

[wporg]是您的新短码。使用短代码将触发wporg\_shortcode回调函数。

## ##在插件中

与主题不同，插件在加载过程的早期阶段运行，因此要求我们推迟添加我们的短代码，直到WordPress初始化为止。

我们建议使用init动作钩子。

```
<?php
function wporg_shortcode_init()
{
    function wporg_shortcode($atts = [], $content = null)
    {
        // do something to $content

        // always return
        return $content;
    }
    add_shortcode('wporg', 'wporg_shortcode');
}
```

```
add_action('init', 'wporg_shortcodes_init');
```

## ##删除一个短码

可以使用Shortcode API来删除短码。该过程涉及使用remove\_shortcode ( ) 删除已注册的\$标签。

```
<?php
remove_shortcode(
    string $tag
);
```

尝试删除之前，请确保已经注册了该短码。为add\_action()指定较高优先级的数字，或者挂钩到稍后运行的动作钩子。

## ##检查短码是否存在

检查一个短码是否已经注册使用shortcode\_exists()。



# 封闭短码

使用短代码的两种情况是：

- 短代码是一个自我关闭的标签，就像我们在“基本短消息”部分中看到的那样。
- 短码是封闭的内容。

## 封闭内容

用短码封装内容可以对封闭的内容进行操作。

```
[wporg]内容来操纵[/ wporg]
```

如上所述，为了封装一部分内容，您需要做的所有操作都添加了一个类似于HTML的开始[\$tag]和结束[/ \$tag]。

## 处理封闭内容

让我们回到我们原来的[wporg]短代码：

```
<?php
function wporg_shortcode($atts = [], $content = null)
{
    // do something to $content

    // always return
    return $content;
}
add_shortcode('wporg', 'wporg_shortcode');
```

看看回调函数，我们看到我们选择接受两个参数\$atts和\$content。 \$content参数将保留我们附带的内容。稍后我们会谈谈\$atts。

\$content的默认值设置为null，因此我们可以通过使用PHP函数is\_null()来区分自我关闭标签和封闭标签。

shortcode [\$tag]，包括其内容和结束[/ \$tag]将被替换为处理函数的返回值。

**警报：**处理函数负责保护输出。

## 短码截图

短码解析器对帖子的内容执行单次传递。

这意味着如果一个短码处理程序的\$content参数包含另一个短码，则不会被解析。

```
[wporg]another [shortcode] is included[/wporg]
```

通过在处理函数的最终返回值上调用do\_shortcode()可以在其他短码内使用短码。

```
<?php
function wporg_shortcode($atts = [], $content = null)
{
    // do something to $content

    // run shortcode parser recursively
    $content = do_shortcode($content);

    // always return
    return $content;
}
add_shortcode('wporg', 'wporg_shortcode');
```

## 限制

短代码解析器无法处理相同[\$tag]的封闭和非封闭形式的混合。

```
[wporg] non-enclosed content [wporg]enclosed content[/wporg]
```

解析器将其视为由文本“非封闭内容”分隔的两个短码，而不是将其视为包围“非封闭内容[wporg]封闭内容”的单个短代码。

# 带参数的短代码

现在我们知道如何创建一个基本的短代码，以及如何使用它作为自我关闭和封闭，我们将使用短代码 `[ $tag ]` 和处理函数中的参数。

短码 `[ $tag ]` 可以接受参数，称为属性：

```
[wporg title="WordPress.org"]  
Having fun with WordPress.org shortcodes.  
[/wporg]
```

Shortcode处理函数可以接受3个参数：

\$atts - array - `[ $tag ]`属性

\$content - string - 发布内容

\$tag - string - `[ $tag ]`的名称（即短码的名称）

```
function wporg_shortcode($atts = [], $content = null, $tag = '') {}
```

## 解析属性

对于用户来说，短信只是在帖子内容中带有方括号的字符串。用户不知道哪些属性可用，幕后发生什么。

对于插件开发人员，无法强制使用属性的策略。用户可以包括一个属性，两个或全部。

为了控制如何使用短码：

- 声明处理函数的默认参数
- 使用array\_change\_key\_case ( ) 对属性数组执行关键案例的归一化
- 使用shortcode\_atts ( ) 提供默认值array和user \$atts来解析属性
- 在返回之前确保输出

## 完整例子

完整的示例使用基本的短代码结构，处理自我关闭和封闭场景，缩短代码并确保输出。

一个`[wporg]`短码，将接受一个标题，并将显示一个框，我们可以用CSS风格。

```
<?php  
function wporg_shortcode($atts = [], $content = null, $tag = '')  
{  
    // normalize attribute keys, lowercase  
    $atts = array_change_key_case((array)$atts, CASE_LOWER);  
  
    // override default attributes with user attributes
```

```
$wporg_atts = shortcode_atts([
    'title' => 'WordPress.org',
], $atts, $tag);

// start output
$o = '';

// start box
$o .= '<div class="wporg-box">';

// title
$o .= '<h2>' . esc_html__( $wporg_atts['title'], 'wporg' ) . '</h2>';

// enclosing tags
if (!is_null($content)) {
    // secure output by executing the_content filter hook on $content
    $o .= apply_filters('the_content', $content);

    // run shortcode parser recursively
    $o .= do_shortcode($content);
}

// end box
$o .= '</div>';

// return output
return $o;
}

function wporg_shortcodes_init()
{
    add_shortcode('wporg', 'wporg_shortcode');
}

add_action('init', 'wporg_shortcodes_init');
```

# TinyMCE增强型短码

---

可以在TinyMCE的可视化编辑器中解析短码，并使它们呈现实际内容，而不是短码本身。

切换到文本选项卡允许您再次查看实际的短码。

以下是使用此功能的内置WordPress短码。

## 音频短码

---

[audio]短码允许您嵌入单个音频文件。

## 字幕短码

---

字幕短码将图像包装在一个div中，并在标题周围放置一个 `<p class="wp-caption-text">` 标签。

## 画廊短码

---

[gallery] shortcode允许您在div中立即嵌入几个图像。

## 播放列表短码

---

[播放列表]短码允许您附加多个媒体文件，并使用html5播放列表进行呈现。

## 视频短码

---

[video]短码非常类似于[audio]短码，它只是渲染视频而不是音频。

# 设置

---

WordPress提供了两个核心API，使管理界面易于构建，安全性和与WordPress管理的设计一致。

Settings API专注于为开发人员提供创建表单和管理表单数据的方法。

Options API专注于使用简单的键/值系统管理数据。

## 快速参考

---

请参阅使用Settings API和Options API构建自定义设置页面的完整示例。

# 设置API

WordPress 2.7中添加的设置API允许包含设置表单的管理页面被半自动管理。它允许您定义设置页面，这些页面中的部分和部分中的字段。

新设置页面可以与其中的部分和字段一起注册。现有设置页面也可以通过在其中注册新的设置部分或字段来添加。

组织领域的注册和验证仍然需要开发人员的一些努力，但是避免了基础选项管理的大量复杂调试。

**警报：**使用设置API时，POST到wp-admin/options.php的表单提供相当严格的功能检查。用户将需要manage\_options功能（并且在多个站点必须是超级管理员）才能提交表单。

## 为什么要使用设置API？

开发人员可以忽略这个API，而不用编写自己的设置页面。这就提出了这个问题，这个API带来了什么好处？以下是一些快速的破坏一些好处。

### 视觉一致性

使用API生成界面元素可以保证您的设置页面看起来像其他的管理内容。你有没有看过一个看起来像是5岁的插件设计页面？你可以打赌开发者没有使用API。所以，一个强有力的论据是你的界面看起来就像它所属的，并且由于有天赋的WordPress设计师团队，它会看起来很棒！

### 鲁棒性（未来证明！）

由于API是WordPress Core的一部分，任何更新都将自动考虑您的插件的设置页面。如果您离开预订并制作自己的界面，WordPress核心更新更有可能破坏您的自定义。还有更广泛的受众测试和维护API代码，所以它往往会更加稳定。

### 减少工作

当然最直接的好处是WordPress API为您提供了大量的工作。以下是“设置”API的一些示例，除了应用一个非常棒的集成设计。

处理表单提交 - 让WordPress处理检索和存储您的\$\_POST提交。

包括安全措施 - 您可以免费获得额外的安全措施，例如过滤器等。

消毒数据 - 您可以访问与WordPress其余部分相同的方法，以确保字符串安全使用。

### 功能参考

- Setting Register/Unregister Add Field/Section
- register\_setting()

- `unregister_setting()` `add_settings_section()`
- `add_settings_field()`
- Options Form Rendering Errors
- `settings_fields()`
- `do_settings_sections()`
- `do_settings_fields()` `add_settings_error()`
- `get_settings_errors()`
- `settings_errors()`



# 使用设置API

## 添加设置

您必须使用`register_setting()`定义新设置，它将在`{ $wpdb-> prefix } _options`表中创建一个条目。

您可以使用`add_settings_section()`在现有页面上添加新的部分。

您可以使用`add_settings_field()`将新字段添加到现有部分。

**警报：**`register_setting()`以及所提到的`add_settings_*`()函数都应该添加到`admin_init`动作钩子中。

## 添加设置

```
register_setting(  
    string $option_group,  
    string $option_name,  
    callable $sanitize_callback = ''  
);
```

关于`register_setting()`的函数参考有关所使用参数的全面说明。

##添加一节

```
add_settings_section(  
    string $id,  
    string $title,  
    callable $callback,  
    string $page  
);
```

部分是您在具有共享标题的WordPress设置页面上看到的设置组。在插件中，您可以向现有设置页面添加新的部分，而不是创建一个全新的页面。这使您的插件更容易维护，并创建更少的新页面供用户学习。

有关使用的参数的完整说明，请参考关于`add_settings_section()`的函数参考。

##添加一个字段

```
add_settings_field(  
    string $id,  
    string $title,  
    callable $callback,  
    string $page,  
    string $section = 'default',  
    array $args = []  
);
```

有关使用的参数的完整说明，请参考关于add\_settings\_field()的函数参考。

示例

```
<?php
function wporg_settings_init()
{
    // register a new setting for "reading" page
    register_setting('reading', 'wporg_setting_name');

    // register a new section in the "reading" page
    add_settings_section(
        'wporg_settings_section',
        'WPOrg Settings Section',
        'wporg_settings_section_cb',
        'reading'
    );

    // register a new field in the "wporg_settings_section" section, inside the
    "reading" page
    add_settings_field(
        'wporg_settings_field',
        'WPOrg Setting',
        'wporg_settings_field_cb',
        'reading',
        'wporg_settings_section'
    );
}

/**
 * register wporg_settings_init to the admin_init action hook
 */
add_action('admin_init', 'wporg_settings_init');

/**
 * callback functions
 */

// section content cb
function wporg_settings_section_cb()
{
    echo '<p>WPOrg Section Introduction.</p>';
}

// field content cb
function wporg_settings_field_cb()
{
    // get the value of the setting we've registered with register_setting()
    $setting = get_option('wporg_setting_name');
    // output the field
```

```
?>
<input type="text" name="wporg_setting_name" value="<?= isset($setting) ? e
sc_attr($setting) : ''; ?>">
<?php
}
```

## 获取设置

---

```
get_option(
    string $option,
    mixed $default = false
);
```

通过get\_option()函数完成设置。

该函数接受两个参数：该选项的名称和该选项的可选默认值。

示例

```
// get the value of the setting we've registered with register_setting()
$setting = get_option('wporg_setting_name');
```

# 选项API

WordPress 1.0中添加的选项API允许创建，阅读，更新和删除WordPress选项。与设置API结合使用，可以控制在设置页面中定义的选项。

## 选项存放在哪里？

选项存储在{\$wpdb->prefix}\_options表中。\$wpdb->前缀由wp-config.php文件中设置的\$table\_prefix变量定义。

## 选项如何存储？

选项可以以两种方式之一存储在WordPress数据库中：作为单个值或值数组。

### 单值

保存为单个值时，选项名称是指单个值。

```
<?php
// add a new option
add_option('wporg_custom_option', 'hello world!');
// get an option
$options = get_option('wporg_custom_option');
```

### 价值数组

当保存为值数组时，选项名称是指一个数组，其本身可以包含键/值对。

```
<?php
// array of options
$data_r = ['title' => 'hello world!', 1, false];
// add a new option
add_option('wporg_custom_option', $data_r);
// get an option
$options_r = get_option('wporg_custom_option');
// output the title
echo esc_html($options_r['title']);
```

如果您正在使用大量相关选项，将它们存储为阵列可能会对整体性能产生积极的影响。

**注意：**作为单独选项访问数据可能导致许多单独的数据库事务，并且通常，数据库事务是昂贵的操作（在时间和服务器资源方面）。当您存储或检索一组选项时，会发生在单个事务中，这是理想的。

## 功能参考

---

- Add Option Get Option Update Option Delete Option
- `add_option()` `get_option()` `update_option()` `delete_option()`
- `add_site_option()` `get_site_option()` `update_site_option()` `delete_site_option()`

# 自定义设置页面

创建自定义设置页面包括以下组合：创建管理菜单，使用Settings API和Options API。

**警报：**请尝试创建自己的设置页面之前阅读这些章节。

以下示例可用于通过以下注释来快速参考这些主题。

## 完整例子

这增加了名为WPOrg一个初始菜单，完整的示例注册名为wporg\_options定制选项，并使用设置API和选项API（包括表示错误/更新消息）执行CRUD（创建，读取，更新，删除）的逻辑。

```
<?php
/**
 * @internal never define functions inside callbacks.
 * these functions could be run multiple times; this would result in a fatal error.
 */

/**
 * custom option and settings
 */
function wporg_settings_init() {
    // register a new setting for "wporg" page
    register_setting( 'wporg', 'wporg_options' );

    // register a new section in the "wporg" page
    add_settings_section(
        'wporg_section_developers',
        __( 'The Matrix has you.', 'wporg' ),
        'wporg_section_developers_cb',
        'wporg'
    );

    // register a new field in the "wporg_section_developers" section, inside the "wporg" page
    add_settings_field(
        'wporg_field_pill', // as of WP 4.6 this value is used only internally
        // use $args' label_for to populate the id inside the callback
        __( 'Pill', 'wporg' ),
        'wporg_field_pill_cb',
        'wporg',
        'wporg_section_developers',
        [
            'label_for' => 'wporg_field_pill',
            'class' => 'wporg_row',
            'wporg_custom_data' => 'custom',
```

```

]
);
}

/**
 * register our wporg_settings_init to the admin_init action hook
 */
add_action( 'admin_init', 'wporg_settings_init' );

/**
 * custom option and settings:
 * callback functions
 */

// developers section cb

// section callbacks can accept an $args parameter, which is an array.
// $args have the following keys defined: title, id, callback.
// the values are defined at the add_settings_section() function.
function wporg_section_developers_cb( $args ) {
    ?>
    <p id="<?php echo esc_attr( $args['id'] ); ?>"><?php esc_html_e( 'Follow the w
hite rabbit.', 'wporg' ); ?></p>
    <?php
}

// pill field cb

// field callbacks can accept an $args parameter, which is an array.
// $args is defined at the add_settings_field() function.
// wordpress has magic interaction with the following keys: label_for, class.
// the "label_for" key value is used for the "for" attribute of the <label>.
// the "class" key value is used for the "class" attribute of the <tr> containi
ng the field.
// you can add custom key value pairs to be used inside your callbacks.
function wporg_field_pill_cb( $args ) {
    // get the value of the setting we've registered with register_setting()
    $options = get_option( 'wporg_options' );
    // output the field
    ?>
    <select id="<?php echo esc_attr( $args['label_for'] ); ?>"
    data-custom="<?php echo esc_attr( $args['wporg_custom_data'] ); ?>"
    name="wporg_options[<?php echo esc_attr( $args['label_for'] ); ?>]"
    >
    <option value="red" <?php echo isset( $options[ $args['label_for'] ] ) ? ( sel
ected( $options[ $args['label_for'] ], 'red', false ) ) : ( ' ' ); ?>>
    <?php esc_html_e( 'red pill', 'wporg' ); ?>
    </option>
    <option value="blue" <?php echo isset( $options[ $args['label_for'] ] ) ? ( se
lected( $options[ $args['label_for'] ], 'blue', false ) ) : ( ' ' ); ?>>

```

```

<?php esc_html_e( 'blue pill', 'wporg' ); ?>
</option>
</select>
<p class="description">
<?php esc_html_e( 'You take the blue pill and the story ends. You wake in your
bed and you believe whatever you want to believe.', 'wporg' ); ?>
</p>
<p class="description">
<?php esc_html_e( 'You take the red pill and you stay in Wonderland and I show
you how deep the rabbit-hole goes.', 'wporg' ); ?>
</p>
<?php
}

/**
 * top level menu
 */
function wporg_options_page() {
    // add top level menu page
    add_menu_page(
        'WPOrg',
        'WPOrg Options',
        'manage_options',
        'wporg',
        'wporg_options_page_html'
    );
}

/**
 * register our wporg_options_page to the admin_menu action hook
 */
add_action( 'admin_menu', 'wporg_options_page' );

/**
 * top level menu:
 * callback functions
 */
function wporg_options_page_html() {
    // check user capabilities
    if ( ! current_user_can( 'manage_options' ) ) {
        return;
    }

    // add error/update messages

    // check if the user have submitted the settings
    // wordpress will add the "settings-updated" $_GET parameter to the url
    if ( isset( $_GET['settings-updated'] ) ) {
        // add settings saved message with the class of "updated"
        add_settings_error( 'wporg_messages', 'wporg_message', __( 'Settings Saved', '

```



```
wporg' ), 'updated' );
}

// show error/update messages
settings_errors( 'wporg_messages' );
?>
<div class="wrap">
<h1><?php echo esc_html( get_admin_page_title() ); ?></h1>
<form action="options.php" method="post">
<?php
// output security fields for the registered setting "wporg"
settings_fields( 'wporg' );
// output setting sections and their fields
// (sections are registered for "wporg", each field is registered to a specific section)
do_settings_sections( 'wporg' );
// output save settings button
submit_button( 'Save Settings' );
?>
</form>
</div>
}
```

# 元数据

---

元数据根据其定义，是有关信息的信息。在WordPress的情况下，它是与帖子，用户，评论和条款相关的信息。

一个例子是一个称为产品的内容类型，其中包含元数据字段。该字段将存储在postmeta表中。

鉴于WordPress中元数据的多对一关系，您的选择是相当无限的。你可以有你想要的多个元选项，你可以存储任何东西在那里。

本章将讨论管理帖子元数据，创建自定义元框以及呈现后置元数据。

# 管理帖子元数据

## 添加元数据

`add_post_meta()`可以很容易地添加元数据。 该函数接受一个`post_id`，一个`meta_key`，一个`meta_value`和一个唯一的标志。

`meta_key`是你的插件如何引用代码中的其他地方的元值。 像`mycrazymetakeyname`这样的东西可以工作，但是与插件或主题相关的前缀跟随关键字的描述会更有用。 `wporg_featured_menu`可能是一个好的。 应该注意的是，可以多次使用相同的`meta_key`来存储元数据的变体（参见下面的唯一标志）。

`meta_value`可以是字符串，整数或数组。 如果它是一个数组，它将被自动序列化，然后被存储在数据库中。

唯一标志允许您声明该键是否应该是唯一的。 一个非唯一的键是一个帖子可以有多个变体，如价格。

如果你只想要一个帖子的价格，你应该标记它是唯一的，并且`meta_key`将只有一个值。

## 更新元数据

如果一个密钥已经存在，并且要更新，请使用`update_post_meta()`。 如果您使用此功能并且该键不存在，那么它将创建它，就像您使用`add_post_meta()`一样。

与`add_post_meta()`类似，该函数接受一个`post_id`，一个`meta_key`，一个`meta_value`和一个唯一的标志。

## 删除元数据

`delete_post_meta()`接受一个`post_id`，一个`meta_key`和可选的`meta_value`。 它正好是名字所暗示的。

## 角色逃避

发送元值被存储时通过`stripslashes()`函数，所以在传递可能包含\转义字符的值（如JSON）时，您需要小心。

考虑JSON值`{ "key" : "value with \" escaped quotes \ \"\" } :`

```
$escaped_json = '{"key":"value with \"escaped quotes\""}';
update_post_meta($id, 'escaped_json', $escaped_json);
$broken = get_post_meta($id, 'escaped_json', true);
/*
$broken, after stripslashes(), ends up unparsable:
{"key":"value with "escaped quotes"}
*/
```

## 解决办法

通过使用函数`wp_slash()`（在WP 3.6中引入）添加一个级别的\escape，您可以补偿对`stripslashes()`的调用：

```
$escaped_json = '{"key":"value with \"escaped quotes\""}';
```

```
update_post_meta($id, 'double_escaped_json', wp_slash($escaped_json));
$fixed = get_post_meta($id, 'double_escaped_json', true);
/*
$fixed, after stripslashes(), ends up as desired:
{"key":"value with \"escaped quotes\""}
*/
```

## 隐藏的自定义字段

如果您是插件或主题开发人员，并且您打算使用自定义字段来存储参数，请务必注意，WordPress不会显示自定义字段，该自定义字段在自定义字段列表中以 “\_”（下划线）开头 后编辑屏幕或使用the\_meta()模板功能时。这可以通过使用add\_meta\_box()函数以异常的方式显示这些自定义字段是有用的。

下面的示例将添加一个唯一的自定义字段与meta\_key名称 “\_color” 和meta\_value “红色” ，但此自定义字段不会显示在后编辑屏幕中：

```
add_post_meta(68, '_color', 'red', true);
```

## 隐藏数组

另外，如果meta\_value是一个数组，它不会显示在页面编辑屏幕上，即使不使用下划线的meta\_key名称前缀。

# 自定义元数据

## What is a Meta Box?

When a user edits a post, the edit screen is composed of several default boxes: Editor, Publish, Categories, Tags, etc. These boxes are meta boxes. Plugins can add custom meta boxes to an edit screen of any post type.

The content of custom meta boxes are usually HTML form elements where the user enters data related to a Plugin's purpose, but the content can be practically any HTML you desire.

## Why Use Meta Boxes?

Meta boxes are handy, flexible, modular edit screen elements that can be used to collect information related to the post being edited. Your custom meta box will be on the same screen as all the other post related information; so a clear relationship is established.

Meta boxes are easily hidden from users that do not need to see them, and displayed to those that do. Meta boxes can be user-arranged on the edit screen. The users are free to arrange the edit screen in a way that suits them, giving users control over their editing environment.

**Alert: All examples on this page are for illustration purposes only. The code is not suitable for production environments.**

Operations such as securing input, user capabilities, nonces, and internationalization have been intentionally omitted. Be sure to always address these important operations.

## Adding Meta Boxes

To create a meta box use the `add_meta_box()` function and plug it's execution to the `add_meta_boxes` action hook.

The following example is adding a meta box to the post edit screen and the `wporg_cpt` edit screen.

```
function wporg_add_custom_box()  
{  
    $screens = ['post', 'wporg_cpt'];  
    foreach ($screens as $screen) {  
        add_meta_box(  
            'wporg_box_id',           // Unique ID  
            'Custom Meta Box Title', // Box title  
            'wporg_custom_box_html', // Content callback, must be of type callable  
            $screen                  // Post type  
        );  
    }  
}
```

```

    );
}
}
add_action('add_meta_boxes', 'wporg_add_custom_box');
```

The `wporg_custom_box_html` function will hold the HTML for the meta box.

The following example is adding form elements, labels, and other HTML elements.

```

function wporg_custom_box_html($post)
{
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
        <option value="something">Something</option>
        <option value="else">Else</option>
    </select>
    <?php
}
```

Note: Note there are no submit buttons in meta boxes. The meta box HTML is included inside the edit screen's form tags, all the post data including meta box values are transferred via POST when the user clicks on the Publish or Update buttons.

The example shown here only contains one form field, a drop down list. You may create as many as needed in any particular meta box. If you have a lot of fields to display, consider using multiple meta boxes, grouping similar fields together in each meta box. This helps keep the page more organized and visually appealing.

## Getting Values

To retrieve saved user data and make use of it, you need to get it from wherever you saved it initially. If it was stored in the `postmeta` table, you may get the data with `get_post_meta()`.

The following example enhances the previous form elements with pre-populated data based on saved meta box values. You will learn how to save meta box values in the next section.

```

function wporg_custom_box_html($post)
{
    $value = get_post_meta($post->ID, '_wporg_meta_key', true);
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
```

```

        <option value="something" <?php selected($value, 'something'); ?>>Something</option>
        <option value="else" <?php selected($value, 'else'); ?>>Else</option>
    </select>
    <?php
}

```

More on the `selected()` function.

## Saving Values

When a post type is saved or updated, several actions fire, any of which might be appropriate to hook into in order to save the entered values. In this example we use the `save_post` action hook but other hooks may be more appropriate for certain situations. Be aware that `save_post` may fire more than once for a single update event. Structure your approach to saving data accordingly.

You may save the entered data anywhere you want, even outside WordPress. Since you are probably dealing with data related to the post, the `postmeta` table is often a good place to store data.

The following example will save the `wporg_field` field value in the `_wporg_meta_key` meta key, which is hidden.

```

function wporg_save_postdata($post_id)
{
    if (array_key_exists('wporg_field', $_POST)) {
        update_post_meta(
            $post_id,
            '_wporg_meta_key',
            $_POST['wporg_field']
        );
    }
}
add_action('save_post', 'wporg_save_postdata');

```

In production code, remember to follow the security measures outlined in the info box!

## Behind the Scenes

You don't normally need to be concerned about what happens behind the scenes. This section was added for completeness.

When a post edit screen wants to display all the meta boxes that were added to it, it calls the `do_meta_boxes()` function. This function loops through all meta boxes and invokes the callback associated with each.

In between each call, intervening markup (such as `divs`, titles, etc.) is added.

## Removing Meta Boxes

To remove an existing meta box from an edit screen use the `remove_meta_box()` function. The passed parameters must exactly match those used to add the meta box with `add_meta_box()`.

To remove default meta boxes check the source code for the parameters used. The default `add_meta_box()` calls are made from `wp-includes/edit-form-advanced.php`.

## Implementation Variants

So far we've been using the procedural technique of implementing meta boxes. Many plugin developers find the need to implement meta boxes using various other techniques.

## OOP

Adding meta boxes using OOP is easy and saves you from having to worry about naming collisions in the global namespace.

To save memory and allow easier implementation, the following example uses an abstract Class with static methods.

```
abstract class WPORG_Meta_Box
{
    public static function add()
    {
        $screens = ['post', 'wporg_cpt'];
        foreach ($screens as $screen) {
            add_meta_box(
                'wporg_box_id',          // Unique ID
                'Custom Meta Box Title', // Box title
                [self::class, 'html'],   // Content callback, must be of type callable
                $screen                  // Post type
            );
        }
    }

    public static function save($post_id)
    {
        if (array_key_exists('wporg_field', $_POST)) {
            update_post_meta(
                $post_id,
                '_wporg_meta_key',
                $_POST['wporg_field']
            );
        }
    }
}
```



```

public static function html($post)
{
    $value = get_post_meta($post->ID, '_wporg_meta_key', true);
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
        <option value="something" <?php selected($value, 'something'); ?>>S
omething</option>
        <option value="else" <?php selected($value, 'else'); ?>>Else</option
    >
    </select>
    <?php
}
}

add_action('add_meta_boxes', ['WPOrg_Meta_Box', 'add']);
add_action('save_post', ['WPOrg_Meta_Box', 'save']);

```

## AJAX

Since the HTML elements of the meta box are inside the form tags of the edit screen, the default behavior is to parse meta box values from the `$_POST` super global after the user have submitted the page.

You can enhance the default experience with AJAX; this allows to perform actions based on user input and behavior; regardless if they've submitted the page.

## Define a Trigger

First, you must define the trigger, this can be a link click, a change of a value or any other JavaScript event.

In the example below we will define change as our trigger for performing an AJAX request.

```

/*jslint browser: true, plusplus: true */
(function ($, window, document) {
    'use strict';
    // execute when the DOM is ready
    $(document).ready(function () {
        // js 'change' event triggered on the wporg_field form field
        $('#wporg_field').on('change', function () {
            // our code
        });
    });
})(jQuery, window, document);

```

## Client Side Code

Next, we need to define what we want the trigger to do, in other words we need to write our client side code.

In the example below we will make a POST request, the response will either be success or failure, this will indicate wither the value of the `wporg_field` is valid.

```

/*jslint browser: true, plusplus: true */
(function ($, window, document) {
    'use strict';
    // execute when the DOM is ready
    $(document).ready(function () {
        // js 'change' event triggered on the wporg_field form field
        $('#wporg_field').on('change', function () {
            // jQuery post method, a shorthand for $.ajax with POST
            $.post(wporg_meta_box_obj.url, // or ajaxurl
                {
                    action: 'wporg_ajax_change', // POST data,
                    wporg_field_value: $('#wporg_field').val() // POST data,
                }, function (data) {
                    // handle response data
                    if (data === 'success') {
                        // perform our success logic
                    } else if (data === 'failure') {
                        // perform our failure logic
                    } else {
                        // do nothing
                    }
                }
            );
        });
    });
})(jQuery, window, document));

```

We took the WordPress AJAX file URL dynamically from the `wporg_meta_box_obj` JavaScript custom object that we will create in the next step.

Note: If your meta box only requires the WordPress AJAX file URL; instead of creating a new custom JavaScript object you could use the `ajaxurl` predefined JavaScript variable.

Available only in the WordPress Administration. Make sure it is not empty before performing any logic.

## Enqueue Client Side Code

Next step is to put our code into a script file and enqueue it on our edit screens.

In the example below we will add the AJAX functionality to the edit screens of the following post types: post, wporg\_cpt.

The script file will reside at /plugin-name/admin/meta-boxes/js/admin.js,

plugin-name being the main plugin folder,

/plugin-name/plugin.php the file calling the function.

```
function wporg_meta_box_scripts()
{
    // get current admin screen, or null
    $screen = get_current_screen();
    // verify admin screen object
    if (is_object($screen)) {
        // enqueue only for specific post types
        if (in_array($screen->post_type, ['post', 'wporg_cpt'])) {
            // enqueue script
            wp_enqueue_script('wporg_meta_box_script', plugin_dir_url(__FILE__)
                . 'admin/meta-boxes/js/admin.js', ['jquery']);
            // localize script, create a custom js object
            wp_localize_script(
                'wporg_meta_box_script',
                'wporg_meta_box_obj',
                [
                    'url' => admin_url('admin-ajax.php'),
                ]
            );
        }
    }
}
add_action('admin_enqueue_scripts', 'wporg_meta_box_scripts');
```

## Server Side Code

The last step is to write our server side code that is going to handle the request.

```
function wporg_meta_box_ajax_handler()
{
    if (isset($_POST['wporg_field_value'])) {
        switch ($_POST['wporg_field_value']) {
            case 'something':
                echo 'success';
                break;
            default:
                // ...
        }
    }
}
```

```
        echo 'failure';
        break;
    }
}
// ajax handlers must die
die;
}
// wp_ajax_ is the prefix, wporg_ajax_change is the action we've used in client
side code
add_action('wp_ajax_wporg_ajax_change', 'wporg_meta_box_ajax_handler');
```

As a final reminder, the code illustrated on this page lacks important operations that take care of security. Be sure your production code includes such operations.

See Handbook' s AJAX Chapter and the Codex for more on AJAX.

## More Information

---

Complex Meta Boxes in WordPress

How To Create Custom Post Meta Boxes In WordPress

WordPress Meta Boxes: a Comprehensive Developer' s Guide

# 渲染元数据

有两个功能可以方便地访问postmeta表中存储的元数据：get\_post\_meta()，get\_post\_custom()。

有关参数详细信息，请参见功能参考。

```
get_post_meta(  
    int $post_id,  
    string $key = '',  
    bool $single = false  
);
```

例:

```
$wporg_meta_value = get_post_meta(get_the_ID(), 'wporg_meta_key');
```

有关参数详细信息，请参见功能参考。

```
get_post_custom(  
    int $post_id  
);
```

例:

```
$meta_array = get_post_custom(get_the_ID());
```

# 自定义文章类型

---

WordPress将Post Types存储在post表中，允许开发人员按照已经存在的方式注册自定义文章类型。本章将向您展示如何注册自定义帖子类型，如何从数据库中检索其内容，以及如何将它们呈现给公众。

# 注册自定义文章类型

WordPress附带五种默认帖子类型：帖子，页面，附件，修订版，菜单。

在开发您的插件时，您可能需要创建自己的特定内容类型：例如，电子商务网站的产品，电子学习网站的作业或评论网站的电影。

使用自定义帖子类型，您可以注册自己的帖子类型。一旦注册了一个帖子类型，它将获得一个新的顶级管理屏幕，可用于管理和创建该类型的帖子。

要注册新的帖子类型，您可以使用register\_post\_type()函数。

**提醒：**我们建议您将自定义帖子类型放入插件而不是主题。这确保即使用户内容更改主题，用户内容也保持便携。

以下示例注册了一个新的职位类型“产品”，它在数据库中标识为wporg\_product。

```
function wporg_custom_post_type()
{
    register_post_type( 'wporg_product',
        [
            'labels'      => [
                'name'          => __( 'Products' ),
                'singular_name' => __( 'Product' ),
            ],
            'public'       => true,
            'has_archive'  => true,
        ]
    );
}
add_action( 'init', 'wporg_custom_post_type' );
```

请参阅register\_post\_type()的参考页面了解参数的描述。

**警告：**您必须在admin\_init之前和after\_setup\_theme操作挂钩后调用register\_post\_type（ ）。一个很好的钩子是init动作钩子。

## 命名最佳实践

您的邮箱类型功能和标识符的前缀与您的插件，主题或网站对应的短前缀是很重要的。

**警告：**为确保前向兼容性，请勿使用wp\_作为您的标识符 - 正在被WordPress核心所使用。

确保您的自定义帖子类型标识符不超过20个字符，因为数据库中的post\_type列当前是该长度的VARCHAR字

段。

如果您的标识符太泛型，例如：产品。它可能与其他插件或主题冲突。

## 网址

自定义帖子类型在网站网址结构中获得自己的s。。

wporg\_product类型的帖子将使用以下URL结构：[http://example.com/wporg\\_product/%product\\_name%](http://example.com/wporg_product/%product_name%)。

wporg\_product是您的自定义帖子类型的细节，%product\_name%是您的特定产品的块。

最终的固定链接是：[http://example.com/wporg\\_product/wporg-is-awesome](http://example.com/wporg_product/wporg-is-awesome)。

您可以在自定义帖子类型的编辑屏幕上看到固定链接，就像默认的帖子类型一样。

## 自定义帖子类型的自定义插件

要为自定义帖子类型的段落设置自定义段，所有您需要做的是向register\_post\_type ( ) arguments数组中的重写键添加一个key =>值对。

例：

```
function wporg_custom_post_type()
{
    register_post_type( 'wporg_product',
        [
            'labels'      => [
                'name'          => __( 'Products' ),
                'singular_name' => __( 'Product' ),
            ],
            'public'       => true,
            'has_archive'  => true,
            'rewrite'      => [ 'slug' => 'products' ], // my custo
m slug
        ]
    );
}
add_action( 'init', 'wporg_custom_post_type' );
```

以上将导致以下URL结构：[http://example.com/products/%product\\_name%](http://example.com/products/%product_name%)

**警告：**使用类似产品的通用插件可能会与其他插件或主题相冲突。

AND

**注意：**与自定义帖子类型标识符不同，可以通过更改其中一个冲突的帖子类型的插件来轻松解决重复的分块问题。



如果插件作者足够聪明地在参数上包含一个`apply_filters()`调用，可以通过覆盖通过`register_post_type()`函数提交的参数以编程方式完成。

解决重复的帖子类型标识符是不可能的，而不会禁用其中一个冲突的帖子类型。

# 使用自定义文章类型

## 自定义帖子类型模板

您可以为自定义帖子类型创建自定义模板。以相同的方式，可以使用single.php和archive.php显示帖子及其存档，您可以创建模板：

- single-{post\_type}.php - 用于定制帖子类型的单个帖子
- archive-{post\_type}.php - 用于存档

其中{post\_type}是register\_post\_type ( ) 函数的\$post\_type参数。

基于我们以前学到的知识，您可以为单个产品帖子和存档创建单一wporg\_product.php和archive-wporg\_product.php模板文件。

或者，您可以在任何模板文件中使用is\_post\_type\_archive ( ) 函数来检查查询是否显示给定帖子类型的归档页面，以及用于显示帖子类型标题的post\_type\_archive\_title ( ) 函数。

## 按帖子类型查询

您可以通过在WP\_Query类构造函数的arguments数组中传递post\_type键来查询特定类型的帖子。

例：

```
$args = [
    'post_type'      => 'product',
    'posts_per_page' => 10,
];
$loop = new WP_Query($args);
while ($loop->have_posts()) {
    $loop->the_post();
    ?>
    <div class="entry-content">
        <?php the_title(); ?>
        <?php the_content(); ?>
    </div>
    <?php
}
```

这循环了最新的十个产品信息，并逐个显示它们的标题和内容。

## 更改主要查询

注册自定义帖子类型并不意味着它会自动添加到主查询中。

如果您希望将自定义帖子类型的帖子显示在标准档案上，或者将其包含在与其他帖子类型混合在一起的主页上，

请使用pre\_get\_posts动作钩子。

下一个示例将显示主页上的帖子，页面和电影帖子类型的帖子：

```
function wporg_add_custom_post_types($query)
{
    if (is_home() && $query->is_main_query()) {
        $query->set('post_type', ['post', 'page', 'movie']);
    }
    return $query;
}
add_action('pre_get_posts', 'wporg_add_custom_post_types');
```

# 分类

---

分类法是分类/分组事物的一个奇特词。 分类可以是等级的（与父母/孩子）或平坦。

WordPress将分类法存储在term\_taxonomy表中，允许开发人员沿着已经存在的分类标准注册自定义分类法。

分类法有条款作为您分类/分组事物的主题。 它们存储在术语表中。 例如。 名为“艺术”的分类将具有多个术语; 他们可以是“现代”和“十八世纪”。

本章将向您展示如何注册自定义分类法，如何从数据库中检索其内容，以及如何将它们呈现给公众。

**注意：**WordPress 3.4及更早版本具有名为“Links”的分类标准，在WordPress 3.5中已被弃用。

# 使用自定义分类

## ##分类学概论

要了解分类标准是什么，他们做什么，请阅读分类标准介绍。

## ##定制分类

WordPress还允许开发人员创建自定义分类。当想要创建不同的命名系统并使它们以可预测的方式在幕后访问时，自定义分类法是有用的。

随着分类系统的发展，“分类”和“标签”的结构并不是非常有限的，所以开发者创建自己的可能是有益的。

## ##为什么要使用自定义分类法？

你可能会问：“为什么要打造自定义分类法，当我可以按类别和标签进行组织时？”

好吧，让我们用一个例子。假设我们有一个厨师的客户，希望你创建一个网站，她将使用原始食谱。

组织网站的一种方法可能是创建一个名为“食谱”的自定义帖子类型来存储她的食谱。然后创建一个分类“课程”，将“开胃菜”与“甜点”分开，最后将分类“成分”从“巧克力”中分离出来。

这些团体可以使用“分类”或“标签”来定义，您可以拥有一个“课程”类别，其中包含“开胃菜”和“甜点”子类别，以及每种成分含有子类别的“成分”类别。

使用自定义分类法的主要优点是您可以独立于类别和标签参考“课程”和“成分”。他们甚至在管理区域获得自己的菜单。

此外，创建自定义分类法允许您构建自定义界面，这将简化客户的使用寿命，并使数据插入到其业务性质的过程中。

现在想象这些自定义分类法和界面是在一个插件中实现的;您刚刚构建了可以在任何WordPress网站上重复使用的自己的Recepies插件。

## 示例：课程分类

以下示例将向您展示如何创建一个将自定义分类“课程”添加到默认帖子类型的插件。

尝试创建自己的插件之前，请务必阅读插件基础章节。

## ##步骤1：开始之前

转到帖子>添加新页面。你会注意到你只有类别和标签。

没有自定义分类标签

## ##步骤2：创建一个新的插件

使用init动作钩子注册分类“课程”的帖子类型“post”。

```
<?php
/*
 * Plugin Name: Course Taxonomy
 * Description: A short example showing how to add a taxonomy called Course.
 * Version: 1.0
 * Author: developer.wordpress.org
```

```

* Author URI: https://wordpress.slack.com/team/aternus
*/

function wporg_register_taxonomy_course()
{
    $labels = [
        'name'           => _x('Courses', 'taxonomy general name'),
        'singular_name'  => _x('Course', 'taxonomy singular name'),
        'search_items'   => __('Search Courses'),
        'all_items'      => __('All Courses'),
        'parent_item'    => __('Parent Course'),
        'parent_item_colon' => __('Parent Course:'),
        'edit_item'      => __('Edit Course'),
        'update_item'    => __('Update Course'),
        'add_new_item'   => __('Add New Course'),
        'new_item_name'  => __('New Course Name'),
        'menu_name'      => __('Course'),
    ];
    $args = [
        'hierarchical'   => true, // make it hierarchical (like categories)
        'labels'         => $labels,
        'show_ui'        => true,
        'show_admin_column' => true,
        'query_var'      => true,
        'rewrite'        => ['slug' => 'course'],
    ];
    register_taxonomy('course', ['post'], $args);
}
add_action('init', 'wporg_register_taxonomy_course');

```

### ##步骤3：查看结果

激活您的插件，然后转到帖子>添加新的。你应该看到一个新的元框为您的“课程”分类。

`courses_taxonomy_post_screen`

### ##代码细分

以下讨论分解了上面使用的代码描述功能和参数。

函数`wporg_register_taxonomy_course`包含注册自定义分类法所需的所有步骤。

`$ labels`数组包含自定义分类标签。

这些标签将用于在管理区域中显示有关分类法的各种信息。

`$ args`数组包含创建自定义分类时将使用的配置选项。

函数`register_taxonomy()`使用`$ args`数组为配置创建一个新的分类标准，其中标题课程为Post Post Type。

函数`add_action()`将`wporg_register_taxonomy_course`函数执行关联到init动作钩子。

## \$ args

`$ args`数组包含自定义分类法的重要配置，它指示WordPress分类法如何工作。

看一下register\_taxonomy ( ) 函数，以获取所接受参数的完整列表，以及它们中的每一个。

### ##总结

通过我们的课程分类示例，WordPress将自动为课程分类法创建归档页面和子页面。

档案页面将使用“课程” ( / course / term-slug / ) 在/ course /中，并在其下生成子页面。

### ##使用你的分类法

WordPress具有许多功能，用于与您的自定义分类法和其中的条款进行交互。

这里有些例子：

- the\_terms: Takes a Taxonomy argument and renders the terms in a list.
- wp\_tag\_cloud: Takes a Taxonomy argument and renders a tag cloud of the terms.
- is\_taxonomy: Allows you to determine if a given taxonomy exists.

# 在WP 4.2+中使用 “split术语”

在WP 4.2之前，具有相同slug（例如，标签和分享slug “a a a a a a a in in in in in

in。。。。。。））））。。。。从WordPress 4.2开始，当这些共享术语中的一个更新时，它将被拆分：更新的术语将被分配一个新的术语ID。

在绝大多数情况下，这种更新将是无缝和平静的。然而，一些插件和主题在选项，后期元数据，用户元数据或其他地方存储术语ID。WP 4.2将包括两种不同的工具来帮助这些插件和主题的作者过渡。

##'split\_shared\_term'动作

当共享术语被分配一个新的术语ID时，会触发一个新的 “split\_shared\_term” 操作。存储术语ID的插件和主题应挂接到此操作以执行必要的迁移。挂钩的文档如下：

```
/**
 * Fires after a previously shared taxonomy term is split into two separate terms.
 *
 * @since 4.2.0
 *
 * @param int $term_id ID of the formerly shared term.
 * @param int $new_term_id ID of the new term created for the $term_taxonomy_id.
 *
 * @param int $term_taxonomy_id ID for the term_taxonomy row affected by the split.
 * @param string $taxonomy Taxonomy for the split term.
 */
```

以下是插件和主题作者如何利用此操作确存储的术语ID更新的几个示例。

##更新存储在选项中的术语ID

假设你的插件存储一个名为 “featured\_tags” 的选项，它包含一个术语ID数组

（update\_option（'featured\_tags'，array（4,6,10）））。在这个例子中，你将钩住'split\_shared\_term'，检查更新后的术语ID是否在数组中，如有必要，进行更新。

```
/**
 * Update featured tags when a term gets split.
 *
 * @param int $term_id ID of the formerly shared term.
 * @param int $new_term_id ID of the new term created for the $term_taxonomy_id.
 *
 * @param int $term_taxonomy_id ID for the term_taxonomy row affected by the split.
 * @param string $taxonomy Taxonomy for the split term.
 */
```



```
function my_featured_tags_split_shared_term( $term_id, $new_term_id, $term_taxonomy_id, $taxonomy ) {
    // We only care about tags, so we'll first verify that the term is a tag.
    if ( 'post_tag' == $taxonomy ) {
        // Get the current featured tags.
        $featured_tags = get_option( 'featured_tags' );

        // If the updated term is in the array, note the array key.
        $found_term = array_search( $term_id, $featured_tags );
        if ( false !== $found_term ) {
            // The updated term is a featured tag! Replace it in the array, and resave.
            $featured_tags[ $found_term ] = $new_term_id;
            update_option( 'featured_tags', $featured_tags );
        }
    }
}
add_action( 'split_shared_term', 'my_featured_tags_split_shared_term', 10, 4 );
```

## ##更新存储在后期元中的术语ID

有时一个插件可能会在post meta中存储术语ids。 在这种情况下，使用get\_posts ( ) 查询来定位具有元键“primary\_category”的帖子以及与分割术语ID匹配的元值。 确定帖子后，请使用update\_post\_meta ( ) 更改存储在数据库中的值。

```
/**
 * Check primary categories when a term gets split to see if any of them
 * need to be updated.
 *
 * @param int $term_id ID of the formerly shared term.
 * @param int $new_term_id ID of the new term created for the $term_taxonomy_id
 *
 * @param int $term_taxonomy_id ID for the term_taxonomy row affected by the split.
 * @param string $taxonomy Taxonomy for the split term.
 */
function my_primary_category_split_shared_term( $term_id, $new_term_id, $term_taxonomy_id, $taxonomy ) {
    // Ignore all updates except those to categories
    if ( 'category' == $taxonomy ) {
        // Find all the posts where the primary category matches the old term ID.
        $post_ids = get_posts( array(
            'fields' => 'ids',
            'meta_key' => 'primary_category',
            'meta_value' => $term_id,
        ) );
```

```
// If we found posts, update the term ID stored in post meta.
if ( $post_ids ) {
    foreach ( $post_ids as $post_id ) {
        update_post_meta( $post_id, 'primary_category', $new_term_id, $
term_id );
    }
}
}
}
add_action( 'split_shared_term', 'my_primary_category_split_shared_term', 10, 4
);
```

## wp\_get\_split\_term ( ) 函数

'split\_shared\_term'是处理术语ID更改的首选方法。但是，可能有些情况 - 例如延迟插件更新 - 哪些术语被拆分，没有插件有机会钩住'split\_shared\_term'动作。WP 4.2存储有关已分裂的分类术语的信息，并提供wp\_get\_split\_term ( ) 实用程序函数，以帮助开发人员检索此信息。

考虑上面的情况，您的插件在名为“featured\_tags”的选项中存储术语ID。您可能需要构建一个验证这些标记ID的功能（可能要在插件更新中运行），以确保没有任何功能标签已被拆分：

```
function my_featured_tags_check_for_split_terms() {
    $featured_tag_ids = get_option( 'featured_tags', array() );

    // Check to see whether any IDs correspond to post_tag terms that have been
    split.
    foreach ( $featured_tag_ids as $index => $featured_tag_id ) {
        $new_term_id = wp_get_split_term( $featured_tag_id, 'post_tag' );

        if ( $new_term_id ) {
            $featured_tag_ids[ $index ] = $new_term_id;
        }
    }

    // Resave.
    update_option( 'featured_tags', $featured_tag_ids );
}
```

请注意，wp\_get\_split\_term ( ) 具有两个参数 - \$ old\_term\_id和\$ taxonomy - 并返回一个整数。如果您需要检索与旧术语ID相关联的所有拆分术语的列表，无论分类如何，请使用wp\_get\_split\_terms ( \$ old\_term\_id ) 。

# 用户

---

## 什么是用户？

每个WordPress用户至少有一个用户名，密码和电子邮件地址。 创建用户帐户后，该用户可以登录到WordPress管理员或以编程方式登录以访问WordPress功能和数据。

用户被分配了角色，每个角色都有一组功能。 您可以使用自己的功能集创建新的角色。 还可以创建自定义功能并将其分配给现有角色或新角色。

## 最低特权原则

最小权限的主体是指向用户帐户提供对用户工作必不可少的权限的做法。

在WordPress中，开发人员可以利用用户角色来限制人员或脚本执行只允许执行的操作，而不会再执行。

这使得管理员能够为新用户以非常基本的（Subscriber）访问级别打开注册，而不用担心这些用户不应该做的事情。

# 创建和管理用户

## 插入用户

要添加用户使用either `wp_insert_user`或`wp_create_user`。 `wp_create_user`只能创建一个用户，不允许管理任何额外的用户字段。 `wp_insert_user`允许填写所有用户字段。

```
<?php wp_insert_user( $userdata ); ?>
```

## 参数

`$userdata`

(混合) (必需) 用户数据数组，stdClass或WP\_User对象。

Default: None

## 返回值

(混合) 如果成功，返回新创建的用户的`user_id`，否则返回一个WP\_Error对象。

## 例子

下面是一个示例，显示如何插入一个新的用户的网站配置文件字段填充。

```
<?php

$website = "http://example.com";
$userdata = array(
    'user_login' => 'login_name',
    'user_url'   => $website,
    'user_pass'  => NULL // When creating an user, `user_pass` is expected.
);

$user_id = wp_insert_user( $userdata );

//On success
if( !is_wp_error($user_id) ) {
    echo "User created : ". $user_id;
} ?>
```

如果成功，返回新创建的用户的`user_id`，否则返回一个WP\_Error对象。

`$userdata`数组可以包含以下字段

|字段名称|描述|关联过滤器|

|---|---|---||

|ID |将用于更新现有用户的整数。 |( 没有 )|

|user\_pass |一个包含用户明文密码的字符串。 |pre\_user\_pass|

|user\_login |包含用户登录用户名的字符串。 |pre\_user\_login|

|user\_nicename |包含用户的URL友好名称的字符串。 默认是用户的用户名。 |pre\_user\_nicename|

|user\_url |包含用户网站的用户URL的字符串。 |pre\_user\_url|

|user\_email |包含用户电子邮件地址的字符串。 |pre\_user\_email|

|display\_name | 将在网站上显示的字符串。 默认为用户的用户名。 很可能您会想通过晦涩（即不使用和删除默认管理员用户）来改变外观和安全性。 |pre\_user\_display\_name|

|nickname |用户的昵称，默认为用户的用户名。 |pre\_user\_nickname|

|first\_name |用户的名字。 |pre\_user\_first\_name|

|last\_name |用户的姓氏。 |pre\_user\_last\_name|

|description |包含用户内容的字符串。 |pre\_user\_description|

|rich\_editing |是否启用富编辑器的字符串。 假如不是空的话。 |(none)|

|user\_registered| 用户注册的日期。 格式为Y-m-d H:i:s。 |(none)|

|role| 用于设置用户角色的字符串。 |(none)|

|jabber|用户的Jabber帐户。 |(none)|

|aim| 用户的AOL IM帐号。 |(none)|

|yim| 用户的Yahoo IM帐号。 |(none)|

## 笔记

---

- 用途: \$wpdb WordPress数据库层。
- 用途: apply\_filters() 调用大部分\$ userdata字段的过滤器，前缀为'pre\_user'。 见上述说明。
- 用途: do\_action() 在更新时拨打 “profile\_update” 钩子，提供用户的ID
- 用途: do\_action() 在创建提供用户ID的新用户时调用'user\_register'钩子

如果没有ID，将创建一个新的用户。 如果您传递ID，则具有该ID的用户将被更新，并且如果在\$ userdata中设置，则这些元字段将被更新，否则它们被设置为null：

- first\_name,
- last\_name,
- nickname,
- description,
- rich\_editing,
- comment\_shortcuts,
- admin\_color,

- use\_ssl,
- show\_admin\_bar\_front

当使用wp\_insert\_user执行更新操作时，user\_pass应为散列密码，而不是纯文本密码。

## 创建用户

```
<?php wp_create_user( $username, $password, $email ); ?>
```

## 描述

wp\_create\_user函数允许您将新用户插入WordPress数据库。它使用\$wpdb类来转义变量值，准备将其插入到数据库中。然后使用PHP compact ( ) 函数创建一个包含这些值的数组。要创建具有附加参数的用户，请使用wp\_insert\_user ( ) 。

## 例

如在wp-admin/upgrade-functions.php中使用的：

```
$user_id = username_exists( $user_name );  
if ( !$user_id and email_exists($user_email) == false ) {  
    $random_password = wp_generate_password( $length=12, $include_standard_  
special_chars=false );  
    $user_id = wp_create_user( $user_name, $random_password, $user_email );  
} else {  
    $random_password = __( 'User already exists. Password inherited.' );  
}
```

## 参数

\$username

(string) (required) 要创建的用户的用户名。

Default: None

\$password

(string) (required) 要创建的用户密码

Default: None

\$email

(string) (optional) 要创建的用户电子邮件地址。

Default: None

## 退货

成功后 - 此函数返回创建的用户的用户ID。 在失败（用户名或电子邮件已存在）的情况下，函数返回一个错误对象，带有这些可能的值和消息：

- empty\_user\_login, 无法创建具有空登录名的用户。
- existing\_user\_login, 此用户名已被注册。
- existing\_user\_email, 该邮箱地址已被注册。

## 更新用户

```
<?php wp_update_user( $userdata ) ?>
```

### 描述

此功能可更新数据库中的单个用户。 此更新可以包含多个用户元数据作为数组。

要更新单个用户元数据，请改用update\_user\_meta()。

要创建新用户，请改用wp\_insert\_user()。

**注意：**如果当前用户的密码正在更新，那么cookie将被清除！

## 参数

\$userdata

(mixed) (required) 用户数据数组，stdClass或WP\_User对象。

Default: None

Return Values (mixed)

如果成功，返回user\_id，否则返回一个WP\_Error对象。

## 例子

以下是一个示例，显示如何更新用户的网站配置文件字段：

```
<?php

$user_id = 1;
$website = 'http://wordpress.org';

$user_id = wp_update_user( array( 'ID' => $user_id, 'user_url' => $website ) );

if ( is_wp_error( $user_id ) ) {
    // There was an error, probably that user doesn't exist.
} else {
    // Success!
}
```

## 删除用户

```
<?php wp_delete_user( $id, $reassign ); ?></code>
```

### 描述

删除用户，并可选地将帖子和链接重新分配给另一个用户。

如果\$重新分配参数未分配给用户ID，则该用户的所有帖子将被删除。 传递被删除的用户ID的操作 “delete\_user” 将在重新分配或删除帖子后运行。 用户元也将被删除用于该用户ID。

### 参数

- \$id  
(integer) (required) 用户ID  
Default: None
- \$reassign  
(integer) (optional) 重新分配帖子和链接到新的用户ID。  
Default: null
- Return Values  
(boolean)  
完成时为真。

### 例子

允许用户终止其用户帐户。

```
wp_delete_user( $user_id );
```

#### ##笔记

如果您希望在插件中使用此功能，则必须在插件函数中包含./wp-admin/includes/user.php文件，否则将抛出“未定义函数调用” 错误

使用全局: (object) \$wpdb

这是一个管理功能。



# 使用用户元数据

WordPress用户表包含一个非常少量的有关用户的信息，并且不能容纳更多。因为这个原因有一个usermeta表。默认情况下，它包含名字，姓氏，昵称等等。任何东西都可以放在那里，它只是通过user\_id相关。

管理用户元数据有两种不同的主要场景。一个是通过用户个人资料页面上的自定义表单字段，另一个是以编程方式具有正确的功能。

## ##创建表单

如果您只想为站点用户提供一个UI来更新特定的元选项，则可以使用下面的代码。

```
add_action( 'show_user_profile', 'my_show_extra_profile_fields' );
add_action( 'edit_user_profile', 'my_show_extra_profile_fields' );

function my_show_extra_profile_fields( $user ) { ?>

    <h3>Extra profile information</h3>

    <table class="form-table">

        <tr>
            <th><label for="twitter">Twitter</label></th>

            <td>
                <input type="text" name="twitter" id="twitter" value="<?php echo esc_attr( get_the_author_meta( 'twitter', $user->ID ) ); ?>" class="regular-text" /><br />
                <span class="description">Please enter your Twitter username.</span>
            </td>
        </tr>

    </table>
<?php }
```

注意有两个动作标签。show\_user\_profile用于显示您自己的个人资料中的表单，edit\_user\_profile用于显示其他人的表单。

上述代码将在配置文件页面上渲染一个html块，如下所示：

- 显示其他字段的HTML块。
- 其他配置文件字段

## ##保存数据

只需使表单不会使数据保存。为此，您应该使用以下代码：

```
add_action( 'personal_options_update', 'my_save_extra_profile_fields' );
add_action( 'edit_user_profile_update', 'my_save_extra_profile_fields' );

function my_save_extra_profile_fields( $user_id ) {

    if ( !current_user_can( 'edit_user', $user_id ) )
        return false;

    /* Copy and paste this line for additional fields. Make sure to change 'twitter' to the field ID. */
    update_usermeta( absint( $user_id ), 'twitter', wp_kses_post( $_POST['twitter'] ) );
}
```

请注意，即使有可能以前不存在该值，也会使用update\_usermeta。没关系，如果不存在就会创建。

还要注意两个add\_action函数。与上述类似，其中一个适用于您自己的个人资料页面，另一个适用于所有人。

上面的大部分文字都是由贾斯汀·塔德洛克（Justin Tadlock）发表的。

##以编程方式添加用户元

以编程方式添加用户元数据有两种方法。 `add_user_meta()` 和 `update_user_meta()` 。

## add\_user\_meta

```
<?php add_user_meta( $user_id, $meta_key, $meta_value, $unique ); ?>
```

## 获取和渲染用户元

获取用户元数据可以一次完成，并且可以返回所有值的数组，也可以一次获得单个值。这是功能：

```
<?php get_user_meta($user_id, $key, $single); ?>
```

如果你只是传递一个user\_id，你会得到一个数组中的所有选项。

如果你传递一个user\_id和key，那么你将得到一个数组中该键的值。

如果你传递全部，并且单个是真的，那么你将获得字符串或int的值，除非存储的值本身是一个数组或对象。

在这一点上，您可以根据需要打印。

# 角色和功能

角色和功能是两个重要功能，允许您使用WordPress扩展用户。角色是您的用户将被调用，功能是您的用户可以看到和做的。

##什么是角色？

一个角色是让您有权力控制用户在其信息中心中看到的内容。在WordPress中，有六个默认角色：超级管理员，管理员，编辑，作者，贡献者和订阅者。但是可以创造更多的角色，并且可以以许多不同的方式运行。它可以让您更好地控制用户看到的内容。您可以在Dashboard-> Users中更改用户的角色。

devhbook-user\_roles

## 创建新角色

除了现有的角色外，您还可以创建新的角色，并为其定义功能。以下是创建新角色的一些代码。

```
function add_simple_role() {
    add_role( 'simple_role', 'Simple Role', array(
        'read' => true,
        'edit_posts' => true,
        'upload_files' => true,
    ) );
}
//Adds the simple role
add_action('init', 'add_simple_role');
```

你去吧！您现在已经在WordPress中创建了第一个角色。您的角色现在将如何改变用户的角色。

## 操纵能力

一个功能告诉用户角色该角色可以做什么（编辑帖子，发布帖子等）。还可以为自定义的帖子类型（edit\_your\_cpt，read\_your\_cpt等）添加功能。

以下是一些代码来为贡献者角色添加功能。

```
function add_theme_caps() {
    // gets the author role
    $role = get_role( 'contributor' );

    // This only works, because it accesses the class instance.
    // would allow the author to edit others' posts for current theme only
    $role->add_cap( 'edit_others_posts' );
}
add_action( 'admin_init', 'add_theme_caps');
```

这使得贡献者角色能够使用WordPress功能来编辑他人的帖子。

还可以将自己的自定义功能添加到任何角色，无论是默认的还是自定义的。它不会有效果，但是您可以在代码中测试。

## 使用功能

---

WordPress有两个主要功能用于检查功能，`current_user_can`和`user_can`。

### current\_user\_can

---

此功能可以检查当前登录的用户的功能。一个插件或主题开发人员将使用它来规划如何允许或限制访问管理区域，甚至前端内容。它不需要`user_id`，因为它总是在当前用户上工作。它接受一个能力，然后一些参数：

```
<?php current_user_can( $capability, $args ); ?>
```

参数可以为功能提供额外的信息。例如，您可以测试编辑帖子的能力，但也可以传入一个`post_id`并测试编辑该帖子的能力。

如果用户具有适当的功能，这是一个在前端制作编辑链接的实际示例：

```
if ( is_user_logged_in() && current_user_can('edit_posts') ) {  
    edit_post_link('Edit', '<p>', '</p>');  
}
```

### user\_can

---

此功能允许您检查给定用户的功能。

```
<?php user_can( $user, $capability ); ?>
```

此功能比`current_user_can`更受限制，因为它不接受功能的参数。除此之外，它们可以以非常相似的方式使用。

## 多站点

---

函数`current_user_can_for_blog()`几乎完全像`current_user_can`，除了它需要一个`blog_id`，不接受`args`。它用于测试当前用户是否具有特定博客上的给定功能。

```
<?php current_user_can_for_blog($blog_id, $capability); ?>
```

# HTTP API

---

## ##介绍

HTTP代表超文本传输协议，是整个互联网的基础通信协议。即使这是您第一次使用HTTP的经历，也许您可能比了解更多。在最基本的层面上，HTTP的工作原理如下：

- “你好，服务器XYZ，请问我有文件abc.html”
- “好，你好，有一点客户，是的，你可能，这里是”

在PHP中发送HTTP请求有许多不同的方法。WordPress HTTP API的目的是支持尽可能多的这些方法，并使用最适合特定请求的方法。

WordPress HTTP API也可用于与Twitter API或Google Maps API等其他API进行通信和交互。

## HTTP方法

---

HTTP有几种描述特定类型操作的方法或动词。虽然还有几个存在，WordPress已经预先建立了三个最常见的功能。每当HTTP请求被发送时，也会传递一个方法来帮助服务器确定客户端请求哪种动作。

## GET

---

GET用于检索数据。这是迄今为止最常用的动词。每次您查看网站或从API中提取数据时，都会看到GET请求的结果。实际上，您的浏览器向您正在阅读的服务器发送GET请求，并请求用于构建此文章的数据。

## POST

---

POST用于向服务器发送数据，以使服务器以某种方式进行操作。例如，联系表。当您在表单域中输入数据并单击提交按钮时，浏览器将收集数据，并将POST请求发送到服务器，并将其输入到表单中。从那里，服务器将处理联系请求。

## HEAD

---

HEAD不如其他两个人所熟知。HEAD与GET请求基本相同，除了它不检索数据，只有关于数据的信息。此数据描述了上次更新数据时，客户端是否应缓存数据，数据类型等。现代浏览器通常会将HEAD请求发送到先前访问过的页面，以确定是否有更新。如果没有，您实际上可能会看到以前下载的页面副本，而不是使用带宽不必要地拉入同一个副本。

所有良好的API客户端在执行GET请求之前可以利用HEAD来潜在地节省带宽。尽管如果HEAD表示有新数据，则需要两个单独的HTTP请求，具有GET请求的数据大小可能非常大。当HEAD表示数据是新的或不应该被缓存时，只使用GET将有助于节省昂贵的带宽和加载时间。

## 自定义方法

---

还有其他HTTP方法，如PUT，DELETE，TRACE和CONNECT。 这些方法将不在本文中，因为没有预先构建的方法来在WordPress中使用它们，也不是API实现它们的常见方法。

根据您的服务器配置方式，您还可以实现您自己的其他HTTP方法。 总是冒险超越标准的方法，并为其他开发者创造客户端以消耗您的网站或API的巨大潜在限制，但是可以利用WordPress中的任何方法。 我们将在本文中简要介绍一下如何做到这一点。

## 响应码

HTTP使用数字和字符串响应代码。 而不是对每个的冗长解释，这里是标准响应代码。 您可以在创建API时定义自己的响应代码，但是除非您需要支持特定类型的响应，否则最好遵守标准代码。 自定义代码通常在1xx范围内。

## 代码类

三位数代码的最左边的数字可以快速看到响应类型。

状态码	说明
2xx	请求成功
3xx	请求被重定向到另一个URL
4xx	由于客户端错误请求失败。 通常无效的认证或丢失的数据
5xx	由于服务器错误而请求失败。 常常丢失或配置错误的配置文件

## 通用代码

这些是您将遇到的最常见的代码。

状态码	说明
200	OK - 请求成功
301	资源永久移动
302	资源暂时移动
403	禁止 - 通常是由于认证无效
404	资源未找到
500	内部服务器错误
503	服务不可用

## 从API获取数据

GitHub提供了一个优秀的API，它不需要许多公共方面的应用程序注册，因此要演示其中的一些方法，示例将针

对GitHub API。

通过wp\_remote\_get()函数在WordPress中获取数据非常简单。此函数具有以下两个参数：

- \$url – 从中检索数据的资源。这必须是标准的HTTP格式
- \$args – 可选 - 您可以在此处传递一组参数来更改行为和标题，例如Cookie，跟随重定向等。

假定以下默认值，尽管可以通过\$args参数更改它们：

- method – GET
- timeout – 5 – 放弃之前等待多久
- redirection – 5 – 多少次遵循重定向。
- httpversion – 1.0
- blocking – true – 页面的其余部分是否等待完成加载，直到此操作完成？
- headers – array()
- body – null
- cookies – array()

让我们使用一个GitHub用户帐户的URL，看看我们可以得到什么样的信息

```
$response = wp_remote_get( 'https://api.github.com/users/lobaugh' );
```

\$response将包含有关我们的请求的所有标题，内容和其他元数据

```
Array
(
    [headers] => Array
        (
            [server] => nginx
            [date] => Fri, 05 Oct 2012 04:43:50 GMT
            [content-type] => application/json; charset=utf-8
            [connection] => close
            [status] => 200 OK
            [vary] => Accept
            [x-ratelimit-remaining] => 4988
            [content-length] => 594
            [last-modified] => Fri, 05 Oct 2012 04:39:58 GMT
            [etag] => "5d5e6f7a09462d6a2b473fb616a26d2a"
            [x-github-media-type] => github.beta
            [cache-control] => public, s-maxage=60, max-age=60
            [x-content-type-options] => nosniff
            [x-ratelimit-limit] => 5000
        )

    [body] => {"type":"User","login":"lobaugh","gravatar_id":"f25f324a47a1efdf"
```

```

7a745e0b2e3c878f", "public_gists":1, "followers":22, "created_at":"2011-05-23T21:38:50Z", "public_repos":31, "email":"ben@lobaugh.net", "hireable":true, "blog":"http://ben.lobaugh.net", "bio":null, "following":30, "name":"Ben Lobaugh", "company":null, "avatar_url":"https://secure.gravatar.com/avatar/f25f324a47a1efdf7a745e0b2e3c878f?d=https://a248.e.akamai.net/assets.github.com%2Fimages%2Fgravatars%2Fgravatar-user-420.png", "id":806179, "html_url":"https://github.com/blobaugh", "location":null, "url":"https://api.github.com/users/blobaugh"}
[response] => Array
(
    [preserved_text 5237511b45884ac6db1ff9d7e407f225 /] => 200
    [message] => OK
)

[cookies] => Array
(
)

[filename] =>
)

```

所有相同的助手功能都可以与前两个功能一样使用。这里的例外是HEAD永远不会返回一个body，因此元素将始终为空。

## 获得你一直想要的身体

只要身体可以使用wp\_remote\_retrieve\_body()检索。此函数只需要一个参数，来自其他检索的其他wp\_remote\_X函数的响应不是下一个值。

```

$response = wp_remote_get( 'https://api.github.com/users/blobaugh' );
$body = wp_remote_retrieve_body( $response );

```

仍然使用前面的例子中的GitHub资源，\$body将会

```

{"type":"User","login":"blobaugh","public_repos":31,"gravatar_id":"f25f324a47a1efdf7a745e0b2e3c878f","followers":22,"avatar_url":"https://secure.gravatar.com/avatar/f25f324a47a1efdf7a745e0b2e3c878f?d=https://a248.e.akamai.net/assets.github.com%2Fimages%2Fgravatars%2Fgravatar-user-420.png","public_gists":1,"created_at":"2011-05-23T21:38:50Z","email":"ben@lobaugh.net","following":30,"name":"Ben Lobaugh","company":null,"hireable":true,"id":806179,"html_url":"https://github.com/blobaugh","blog":"http://ben.lobaugh.net","location":null,"bio":null,"url":"https://api.github.com/users/blobaugh"}

```

如果您没有任何其他操作来执行响应，除了获得身体，您可以将代码减少到一行



```
$body = wp_remote_retrieve_body( wp_remote_get( 'https://api.github.com/users/blobaugh' ) );
```

这些帮助函数中的许多功能也可以在一行上同样使用。

## 获取响应代码

您可能需要检查响应代码，以确保检索成功。这可以通过wp\_remote\_retrieve\_response\_code()函数完成：

```
$response = wp_remote_get( 'https://api.github.com/users/blobaugh' );
$http_code = wp_remote_retrieve_response_code( $response );
```

如果成功\$http\_code将包含200。

## 获取一个特定的头文件

如果你想要检索一个特定的标题，比如说最后一次修改，你可以用wp\_remote\_retrieve\_header ( ) 这样做。此功能需要两个参数

- \$response - 获取调用的响应
- \$header - 要检索的头的名称

检索最后修改的标题

```
$response = wp_remote_get( 'https://api.github.com/users/blobaugh' );
$last_modified = wp_remote_retrieve_header( $response, 'last-modified' );
```

\$ last\_modified 将包含[last-modified] => Fri , 05 Oct 2012 04:39:58 GMT

您还可以使用wp\_remote\_retrieve\_headers ( \$ response ) 检索数组中的所有头。

## GET使用基本认证

受保护的API提供了许多不同类型的身份验证中的一种或多种。 HTTP基本身份验证是一种常见的，但不是非常安全的身份验证方法。 通过将“授权”传递给wp\_remote\_get()函数的第二个参数以及其他HTTP方法函数，可以在WordPress中使用。

```
$args = array(
    'headers' => array(
        'Authorization' => 'Basic ' . base64_encode( YOUR_USERNAME . ':' . YOUR_PASSWORD )
    )
);
```

```
wp_remote_get( $url, $args );
```

## 将数据发布到API

相同的帮助方法（wp\_remote\_retrieve\_body()等）可用于所有HTTP方法调用，并以相同的方式使用。

POST数据使用wp\_remote\_post()函数完成，并且与wp\_remote\_get()完全相同的参数。这里应该注意的是，您需要传递第二个参数的数组中的所有元素。词典提供默认的可接受值。您只需要关心发送的数据，所以其他值将被默认。

要将数据发送到服务器，您将需要构建一个关联的数据数组。该数据将被分配给“body”值。从服务器端的值将会出现在\$\_POST变量中。即如果在服务器\$\_POST['myvar'] = 5上的body => array( 'myvar' => 5 )。

因为GitHub不允许POST到上一个例子中使用的API，这个例子将假装它是这样的。通常，如果要将数据POST到API，您需要联系API的维护者并获取API密钥或某种其他形式的身份验证令牌。这只是证明您的应用程序被允许以与用户对网站登录网站相同的方式操纵API上的数据。

假设我们提交的联系表单包含以下字段：姓名，电子邮件，主题，评论。要设置身体，我们执行以下操作：

```
$body = array(
    'name' => 'Jane Smith',
    'email' => 'some@email.com',
    'subject' => 'Checkout this API stuff',
    'comment' => 'I just read a great tutorial by this Ben Lobaugh. It taught me amazing things about interacting with APIs in WordPress! You gotta check it out!'
);
```

现在我们需要设置将被传递给wp\_remote\_post()的第二个参数的其余值，

```
$args = array(
    'body' => $body,
    'timeout' => '5',
    'redirection' => '5',
    'httpversion' => '1.0',
    'blocking' => true,
    'headers' => array(),
    'cookies' => array()
);
```

那么当然要打电话

```
$response = wp_remote_post( 'http://your-contact-form.com', $args );
```

对于那些不喜欢拼接在一起这里代码的人，整个片段

```

$body = array(
    'name' => 'Jane Smith',
    'email' => 'some@email.com',
    'subject' => 'Checkout this API stuff',
    'comment' => 'I just read a great tutorial by this Ben Lobaugh. It taught m
e amazing things about interacting with APIs in WordPress! You gotta check it o
ut!'
);

$args = array(
    'body' => $body,
    'timeout' => '5',
    'redirection' => '5',
    'httpversion' => '1.0',
    'blocking' => true,
    'headers' => array(),
    'cookies' => array()
);

$response = wp_remote_post( 'http://your-contact-form.com', $args );

```

## HEADING关闭带宽使用

API可能非常重要，有时候需要使用HEAD检查资源状态。在高流量API上，GET通常限于每分钟或小时的请求数。除非HEAD请求显示API上的数据已更新，否则不需要尝试GET请求。

如前所述，HEAD包含有关数据是否已被更新的数据，如果数据应该被缓存，何时到期缓存的副本，以及有时对API的请求的速率限制。

回到GitHub示例，这里有几个标题要注意。大多数这些标头是标准的，但是您应该始终检查API文档，以确保您了解哪些头命名为什么以及其目的。

- x-ratelimit-limit - 一段时间内允许的请求数
- x-ratelimit-remaining - 在时间段内剩余的可用请求数
- content-length - 内容以字节为单位。如果内容相当大，警告用户可能很有用
- last-modified - 资源上次修改时。对缓存工具非常有用
- cache-control - 客户端应如何处理缓存

以下将检查我的GitHub用户帐户的HEAD值：

```

$response = wp_remote_head( 'https://api.github.com/users/blobaugh' );

```

\$response应该看起来类似

```

Array
(
    [headers] => Array
        (
            [server] => nginx
            [date] => Fri, 05 Oct 2012 05:21:26 GMT
            [content-type] => application/json; charset=utf-8
            [connection] => close
            [status] => 200 OK
            [vary] => Accept
            [x-ratelimit-remaining] => 4982
            [content-length] => 594
            [last-modified] => Fri, 05 Oct 2012 04:39:58 GMT
            [etag] => "5d5e6f7a09462d6a2b473fb616a26d2a"
            [x-github-media-type] => github.beta
            [cache-control] => public, s-maxage=60, max-age=60
            [x-content-type-options] => nosniff
            [x-ratelimit-limit] => 5000
        )

    [body] =>
    [response] => Array
        (
            [preserved_text 39a8515bd2dce2aa06ee8a2a6656b1de /] => 200
            [message] => OK
        )

    [cookies] => Array
        (
        )

    [filename] =>
)

```

所有相同的助手功能都可以与前两个功能一样使用。这里的例外是HEAD永远不会返回一个body，因此元素将始终为空。

## 做任何形式的请求

如果您需要使用上述功能不支持的HTTP方法发出请求，请不要惊慌。发展WordPress的伟大人物已经想到了，并且亲切地提供了wp\_remote\_request()。此函数使用与wp\_remote\_get()相同的两个参数，并允许您同时指定HTTP方法。您需要传递哪些数据取决于您的方法。

要发送DELETE方法示例，您可能有以下类似的内容：

```

$args = array(
    'method' => 'DELETE'
)

```

```
);
$response = wp_remote_request( 'http://some-api.com/object/to/delete', $args );
```

## 缓存介绍

缓存是一种习惯，通常需要大量时间构建的对象或对象被保存到快速对象存储中，以便稍后的请求快速检索。这样就可以避免花费时间取回并再次构建对象。缓存是一个广泛的主题，是网站优化的一部分，可以自己完成一系列的文章。以下只是缓存的介绍和一个简单而有效的方法来快速设置缓存的API响应。

为什么要缓存API响应？那么，房间里的大象是因为外部API会减慢你的网站。许多顾问会告诉您，利用外部API将通过减少连接数量和处理性能以及成本高的带宽来提高网站的性能，但有时这根本不是真的。

您的服务器可以发送数据的速度与远程服务器处理请求所需的时间，构建数据并将其发送回来，这是一个很好的平衡。第二个显着的方面是，许多API在一段时间内具有有限数量的请求，并且可能一次应用程序对连接数量的限制。缓存有助于通过在服务器上放置数据副本来解决这些难题，直到需要刷新。

## 什么时候应该缓存？

这个简单的答案是总是，但再次有时你不应该。如果您正在处理实时数据或API专门指出不缓存在标题中，您可能不想缓存，但对于所有其他情况，通常最好缓存从API检索的任何资源。

## WordPress瞬变

WordPress瞬态提供了一种方便的方法来存储和使用缓存的对象。瞬态生存指定的时间，或者直到您需要它们在API更新资源时过期。在WordPress中使用瞬态功能可能是您遇到的最简单的缓存系统。只有三个功能 to do all the heavy lifting for you.

## 缓存一个对象（设置一个瞬态）

使用set\_transient()函数来缓存对象。此功能需要以下三个参数：

- \$transient - 短暂名称供将来参考
- \$value - 瞬态值
- \$expiration - 从暂时保存到到期之前有多少秒钟

从上面缓存GitHub用户信息响应一个小时的例子是

```
$response = wp_remote_get( 'https://api.github.com/users/blobaugh' );

set_transient( 'blobaugh_github_userinfo', $response, 60*60 );
```

## 获取一个缓存的对象（获取一个暂时的）

获取缓存的对象比设置一个临时对象要复杂得多。您需要请求瞬态，但是您还需要检查以确定瞬态是否已过期，如果是这样，获取更新的数据。通常，`set_transient()`调用是在`get_transient()`调用之内进行的。以下是获取GitHub用户配置文件的临时数据的示例：

```
$github_userinfo = get_transient( 'blobaugh_github_userinfo' );

if( false === $github_userinfo ) {
    // Transient expired, refresh the data
    $response = wp_remote_get( 'https://api.github.com/users/blobaugh' );
    set_transient( 'blobaugh_github_userinfo', $response, 60*60 );
}
// Use $github_userinfo as you will
```

## 删除缓存的对象（删除一个瞬态）

删除缓存的对象是所有瞬态函数中最简单的，只需传递一个瞬态名称的参数即可完成。

要删除Github用户信息：

```
delete_transient( 'blobaugh_github_userinfo' );
```

# JavaScript

---

JavaScript是许多WordPress插件中的重要组件。 WordPress附带了与核心捆绑的各种JavaScript库。

WordPress中最常用的库之一是jQuery，因为它是轻量级的，易于使用。 jQuery可以在您的插件中使用来操纵DOM对象或执行Ajax操作。

# jQuery

## 使用jQuery

收到WordPress网页后，您的jQuery脚本会在用户浏览器上运行。一个基本的jQuery语句有两个部分：一个选择器，用于确定代码适用于哪些HTML元素，以及一个动作或事件，用于确定代码的作用或反应。基本事件语句如下所示：

```
jQuery.(selector).event(function);
```

当由选择器选择的HTML元素中发生诸如鼠标点击的事件时，将执行在最后一组括号内定义的函数。

所有以下代码示例均基于此HTML页面内容。假设它出现在插件的管理设置屏幕上，由myplugin\_settings.php文件定义。它是一个简单的表，每个标题旁边都有单选按钮。

```
<pre>
<form id="radioform">
  <table>
    <tbody>
      <tr>
        <td><input class="pref" checked="checked" name="book" type="radio" value="Sycamore Row" />Sycamore Row</td>
        <td>John Grisham</td>
      </tr>
      <tr>
        <td><input class="pref" name="book" type="radio" value="Dark Witch" />Dark Witch</td>
        <td>Nora Roberts</td>
      </tr>
    </tbody>
  </table>
</form>
</pre>
```

输出可能在您的设置页面看起来像这样。

## 样品表

在关于AJAX的文章中，我们将构建一个AJAX交换器，用于在usermeta中保存用户选择，并添加标有选定标题的帖子数量。不是一个非常实用的应用程序，但它说明了所有重要的步骤。jQuery代码可以驻留在外部文件中，也可以输出到 `<script>` 块内的页面。我们将专注于外部文件变体，因为PHP的传递值需要特别注意。如果您似乎更方便，则可以将相同的代码输出到页面。



## 选择器和事件

选择器与CSS选择器的形式相同：“.class”或“#id”。有更多的形式，但这些是您将经常使用的两个。在我们的示例中，我们将使用类“.pref”。还有一大堆可能的事件，一个你可能会使用很多是'点击'。在我们的例子中，我们将使用'change'来捕获单选按钮。请注意，jQuery事件通常被命名为与JavaScript不同的命名。到目前为止，在添加了一个空的匿名函数之后，我们的示例语句如下所示：

```
$(".pref").change(function(){  
    /*do stuff*/  
});
```

当“pref”类的任何元素发生变化时，此代码将“做任何事情”。

注意：此代码片段以及此页面上的所有示例都用于说明AJAX的使用。代码不适合于生产环境，因为有意的省略了相关的操作，如清洁，安全，错误处理和国际化。一定要在生产代码中解决这些重要的操作。

# Ajax

## 什么是Ajax？

Ajax是Asynchronous JavaScript And XML的缩写。XML是数据交换格式，UX是用户体验的软件开发人员简写。Ajax是一种互联网通信技术，允许用户浏览器中显示的网页从服务器请求特定信息，并将该新信息显示在同一页面上，而无需重新加载整个页面。您可以想象如何改善用户体验。

虽然XML是使用的传统数据交换格式，但交换实际上可以是任何方便的格式。当使用PHP代码时，许多开发人员赞成JSON，因为从传输的数据流创建的内部数据结构更容易接口。

要查看Ajax的行动，请转到您的WordPress管理区域并添加一个类别或标签。当您点击添加新按钮时，请注意页面更改，但实际上不会重新加载。不相信检查浏览器的后台历史记录，如果页面已重新加载，您将看到该页面的两个条目。

Ajax甚至不需要用户操作。Google文档每隔几分钟就可以自动保存您的文档，而无需启动保存操作。

## 为什么要使用Ajax？

显然，它改善了用户体验。Ajax不是提供一个无聊的静态页面，而是让您呈现一个动态，响应敏捷，用户友好的体验。用户可以立即获得反馈，表明他们采取的一些行动是对还是错。在发现一个领域有错误之前，不需要提交整个表格。一旦输入数据，就可以验证重要字段。或者可以根据用户类型进行建议。

Ajax可以显着减少数据来回流动的数量。只有相关数据需要交换，而不是所有的页面内容，这是当页面重新加载时会发生什么。

与WordPress插件特别相关，Ajax是独立于WordPress内容的最佳方式。如果你以前编程过PHP，你可能会简单地链接到一个新的PHP页面。链接后的用户启动该过程。问题在于，当您链接到新的外部PHP页面时，您无法访问任何WordPress功能。过去，开发人员通过在其新的PHP页面上包含核心文件wp-load.php来访问WordPress功能。这样做的问题是您可能不知道该文件的正确路径。WordPress架构现在足够灵活，可以将/wp-content/和您的插件文件从其常规位置移动到安装根目录的一个级别。你不知道wp-load.php相对于你的插件文件的位置，你也不知道安装文件夹的绝对路径。

您可以知道的是发送Ajax请求的地方，因为它在全局JavaScript变量中定义。您的PHP Ajax处理程序脚本实际上是一个动作钩子，所以所有WordPress函数都可以自动使用，与外部PHP文件不同。

## 如何使用Ajax？

如果您是新的WordPress，但在其他环境中使用Ajax的经验，您将需要重新学习一些东西。WordPress实现Ajax的方式最有可能与您以前使用的方式不同。如果一切都是新的，没问题。你将在这里学习基础知识。一旦开发了一个基本的Ajax交换机，那么在这个基础上进行扩展就可以开发出一个令人敬畏的用户界面的杀手级应用程序！

WordPress中任何Ajax交换都有两个主要组件。客户端JavaScript或jQuery和服务器端PHP。所有Ajax交易所遵

循以下事件顺序。

某些页面事件会启动JavaScript或jQuery函数。该函数从页面收集一些数据，并通过HTTP请求将其发送到服务器。由于使用JavaScript处理HTTP请求是尴尬的，并且jQuery已经捆绑到WordPress中，我们将仅关注来自这里的jQuery代码。具有直接JavaScript的Ajax是可行的，但在jQuery可用时不值得。

服务器接收请求并对数据执行某些操作。它可以组合相关数据，并以HTTP响应的形式将其发送回客户端浏览器。这不是一个要求，但是由于保持用户了解发生了什么更可取，所以很少发送某种响应。

发送初始Ajax请求的jQuery函数接收服务器响应并执行一些操作。它可以通过某些方式更新页面上的内容和/或向用户呈现消息。

## 使用Ajax与jQuery

现在我们将在jQuery文章中的代码段中定义“do stuff”部分。我们将使用\$.post()方法，它需要3个参数：发送POST请求的URL，要发送的数据，以及一个回调函数来处理服务器响应。在我们这样做之前，我们有一些先进的计划来摆脱困境。我们在回调函数中稍后执行以下作业。回调部分的目的将会更加明显。

```
var this2 = this;
```

## 网址

所有WordPress Ajax请求都必须发送到wp-admin / admin-ajax.php。正确的，完整的URL需要来自PHP，jQuery无法自行确定此值，您无法对jQuery代码中的URL进行硬编码，并希望其他人在其网站上使用您的插件。如果页面来自管理区域，WordPress将在全局JavaScript变量ajaxurl中设置正确的URL。对于公共区域的页面，您需要自己建立正确的URL，并使用wp\_localize\_script()将其传递给jQuery。这将在PHP部分中更详细地介绍。因为现在只需知道可以在前端和后端工作的URL作为您将在PHP段中定义的全局对象的属性。在jQuery中，它被引用为：

```
my_ajax_obj.ajax_url
```

## 数据

需要发送到服务器的所有数据都包含在数据数组中。除了应用程序所需的任何数据外，还必须发送一个动作参数。对于可能导致数据库更改的请求，您需要发送随机数，以便服务器知道请求来自合法来源。我们提供给.post()方法的示例数据数组如下所示：

```
{_ajax_nonce: my_ajax_obj.nonce, //nonce
  action: "my_tag_count",         //action
  title: this.value                //data
}
```

每个组件如下所述。

### ##随机

Nonce是“使用ONCE”的portmanteau。它本质上是分配给任何形式的每个实例的唯一的十六进制序列号。nonce是用PHP脚本建立的，并以与URL相同的方式传递给jQuery，作为全局对象中的属性。在这种情况下，它被引用为my\_ajax\_obj.nonce。

注意：每次使用一个真实的随机数需要刷新，所以下一个Ajax调用有一个新的，未使用的随机数作为验证发送。事实上，WordPress的nonce实现不是一个真正的随机数。在24小时内可以根据需要多次使用相同的随机数。生成具有相同种子短语的随机数将始终产生相同的数字，持续12小时，之后将最终生成新的数字。

如果您的应用程序需要严格的安全性，请实施一个真正的随机系统，其中服务器发送一个新的，新的随机数，以响应Ajax请求，用于验证下一个请求。

如果您将这个随机数值键入\_ajax\_nonce，最简单。如果与验证随机数的PHP代码进行协调，则可以使用其他键，但使用默认值更容易，而不用担心协调。这是键值对的声明出现的方式：

```
_ajax_nonce: my_ajax_obj.nonce
```

## Action

所有WordPress Ajax请求都必须在数据中包含一个动作参数。该值是一个任意字符串，部分用于构建用于挂接Ajax处理程序代码的操作标签。这个值对Ajax调用的目的是一个很简单的描述。毫不奇怪，这个价值的关键是“行动”。在这个例子中，我们将使用“my\_tag\_count”作为动作值。该键值对的声明如下所示：

```
action: "my_tag_count"
```

这个数组中还包括服务器需要完成任务的任何其他数据。如果有很多字段要传输，有两种通用格式将数据字段组合成一个字符串，以便更方便的传输，XML和JSON。使用这些格式是可选的，但无论您做什么，都需要与服务器的PHP脚本进行协调。有关这些格式的更多信息，请参见以下回调部分。以这种格式接收数据比发送数据更常见，但它可以同时工作。

在我们的示例中，服务器只需要一个值，即所选书名的单个字符串，因此我们将使用关键字'title'。在jQuery中，触发事件的对象总是包含在变量this中。因此，所选元素的值为this.value。我们对这个键值对的声明如下：

```
title: this.value
```

## Callback

回调处理程序是在请求完成后响应从服务器返回时执行的功能。再次，我们通常在这里看到一个匿名的功能。该函数传递一个参数，服务器响应。响应可能是从是或否到巨大的XML数据库。JSON格式的数据也是数据的有用格式。回复甚至不需要。如果没有，则不需要指定回调。为了UX的利益，让用户知道任何请求发生了什么，总是一个好主意，因此建议始终回应并提供一些发生的事情。

在我们的示例中，我们将无线电输入之后的当前文本替换为服务器响应，其中包括由书标题标记的帖子数。这是我们的匿名回调函数：

```
function(data) {  
    this2.nextSibling.remove();  
    $(this2).after(data);  
}
```

数据包含整个服务器响应。之前我们分配给this2触发更改事件的对象（引用为这样），使用行var this2 = this;这是因为闭包中的变量范围只能扩展一个层次。通过在事件处理程序（最初刚刚包含“/\* do stuff \*/”的部分）中分配this2，我们可以在回调中使用它，因为这将超出范围。

服务器响应可以采用任何形式。大量的数据应该被编码成数据流，以便于处理。XML和JSON是两种常见的编码方案。

---

## XML

XML是Ajax的经典数据交换格式。这是Ajax毕竟是'X'。它仍然是一种可行的交换格式，即使使用本机PHP函数可能很困难。许多PHP程序员因此而喜欢使用JSON交换格式。如果您使用XML，解析方法取决于正在使用的浏览器。将Microsoft.XMLDOM ActiveX用于Internet Explorer，并使用DOMParser进行其他操作。

---

## JSON

通常，JSON的重量轻且易于使用，因此常常受益。你可以使用eval()来解析JSON，但不要这样做！使用eval()具有重大的安全隐患。相反，使用一个专门的解析器，这也更快。使用解析器对象JSON的全局实例。为了确保它可用，请确保它与页面上的其他脚本一起排队。有关入队的更多信息将在PHP部分稍后介绍。

---

## Other

只要数据格式与PHP处理程序协调一致，它可以是任何您喜欢的格式，例如逗号分隔，制表符分隔或任何适合您的结构。

---

## 客户端摘要

现在我们已将我们的回调函数添加为\$post()函数的最终参数，我们已经完成了我们的jQuery Ajax脚本。所有的部分放在一起看起来像这样：

```
jQuery(document).ready(function($) {                                //wrapper
    $(".pref").change(function() {                                  //event
        var this2 = this;                                          //use in callback
        $.post(my_ajax_obj.ajax_url, {                             //POST request
            _ajax_nonce: my_ajax_obj.nonce,                        //nonce
            action: "my_tag_count",                                //action
            title: this.value                                       //data
        }, function(data) {                                        //callback
            this2.nextSibling.remove();                             //remove current title
            $(this2).after(data);                                   //insert server response
        });
    });
});
```

该脚本可以输出到网页上的一个块中，也可以包含在自己的文件中。该文件可以驻留在互联网上的任何地方，但大多数插件开发人员将其放置在插件主文件夹的 /js /子文件夹中。除非你有理由去做，否则你也可以遵守惯例。在这个例子中，我们将命名我们的文件myjquery.js

# 服务器端PHP和入队

实现AJAX通信所需的服务器端PHP脚本有两个部分。首先，我们需要在网页上排列jQuery脚本，并本地化jQuery脚本需要的任何PHP值。其次是AJAX请求的实际处理。

## 入队脚本

本部分介绍了WordPress中AJAX的两大难点，它将经验丰富的编程人员新增至WordPress。一个是需要排入脚本，以便获得元链接在页面的头部分中正确显示。另一个是所有AJAX请求需要通过wp-admin / admin-ajax.php发送。不要将请求直接发送到您的插件页面。

## 排队

使用函数wp\_enqueue\_script()让WordPress在页面的部分插入一个元链接到您的脚本。不要在标题模板中硬编码这样的链接。作为一个插件开发人员，您没有准备好访问标题模板，但是这个规则不管怎么说。

入队功能有三个参数。第一个是用于在其他功能中引用脚本的任意标签或句柄。第二个是脚本文件的完整URL。为了便于携带，请使用plugins\_url()来构建正确的URL。如果你为插件之外的东西排队脚本，使用一些相关的功能创建一个正确的URL - 从来没有硬编码。第三个参数是脚本依赖的任何脚本标记的数组。由于我们使用jQuery发送一个AJAX请求，所以至少需要在数组中列出'jquery'。始终使用数组，即使它是单个依赖项。我们的示例的入队呼叫如下所示：

```
wp_enqueue_script( 'ajax-script',  
    plugins_url( '/js/myjquery.js', __FILE__ ),  
    array( 'jquery' )  
);
```

加载脚本时，您无法直接从插件代码页排入脚本。脚本必须从几个动作钩子之一排队 - 哪一个取决于脚本需要链接到哪个页面。对于管理页面，请使用admin\_enqueue\_scripts。对于前端页面使用wp\_enqueue\_scripts，除了登录页面，在这种情况下使用login\_enqueue\_scripts。

admin\_enqueue\_scripts钩子将当前页面文件名传递给您的回调。使用此信息仅将脚本排列在需要的页面上。前端版本没有通过任何内容。在这种情况下，使用模板标签（如is\_home()，is\_single()等）来确保您只需将脚本排列在需要的位置。这是我们的示例的完整的入队代码：

```
add_action( 'admin_enqueue_scripts', 'my_enqueue' );  
function my_enqueue( $hook ) {  
    if( 'myplugin_settings.php' != $hook ) return;  
    wp_enqueue_script( 'ajax-script',  
        plugins_url( '/js/myjquery.js', __FILE__ ),  
        array( 'jquery' )  
    );  
};
```



```
}
```

为什么我们在这里使用命名函数，但是使用jQuery的匿名函数？因为闭包最近才被PHP支持。jQuery已经支持了很多时间。由于有些人可能仍在运行旧版本的PHP，因此我们总是使用命名函数来实现最大的兼容性。如果您有最新的PHP版本，并且仅针对您自己的安装开发，请继续使用闭包。

### ##注册与入队

您将在其他教程中看到宗教使用wp\_register\_script()的示例。这很好，但它的使用是可选的。什么是不可选的是wp\_enqueue\_script()。必须调用此函数才能使您的脚本文件在网页上正确链接。那么为什么要注册脚本？它创建一个有用的标签或句柄，您可以根据需要轻松地在代码的各个部分中引用该脚本。如果您只需要加载脚本，而不是在代码中的其他地方引用它，则无需注册。

### ##随机

您需要创建一个随机数，以便jQuery AJAX请求可以被验证为合法请求，而不是来自某个未知的坏行为者的潜在的恶意请求。只有你的PHP脚本和你的jQuery脚本才能知道这个值。收到请求后，您可以验证它是在此创建的相同值。这是为了创建一个我们的例子的随机数

```
$title_nonce = wp_create_nonce( 'title_example' );
```

参数title\_example可以是任意任意的字符串。建议字符串与所使用的内容相关，但它可以是适合您的任何东西。

### ##本地化

如果您从jQuery部分回想起，由jQuery创建的由jQuery创建的数据被传递给一个名为my\_ajax\_obj的全局对象。在我们的示例中，此数据是一个随机数，并且是admin-ajax.php的完整URL。分配对象属性和创建全局jQuery对象的过程称为本地化。这是我们使用wp\_localize\_script()的示例中使用的本地化代码。

```
wp_localize_script( 'ajax-script', 'my_ajax_obj', array(
    'ajax_url' => admin_url( 'admin-ajax.php' ),
    'nonce'    => $title_nonce, // It is common practice to comma after
) );                          // the last array item for easier maintenance
```

注意我们的脚本如何处理ajax脚本被使用，以便将全局对象分配给正确的脚本。对象对我们的脚本是全局的，而不是所有的脚本。也可以从用于排入脚本的同一个钩子调用本地化。创建一个随机数也同样如此，尽管这个特定的功能可以在几乎任何地方被调用。所有这些组合在一起的单个钩子回调看起来像这样：

```
add_action( 'admin_enqueue_scripts', 'my_enqueue' );
function my_enqueue( $hook ) {
    if( 'myplugin-settings.php' != $hook ) return;
    wp_enqueue_script( 'ajax-script',
        plugins_url( '/js/myjquery.js', __FILE__ ),
```



```

        array( 'jquery' )
    );
    $title_nonce = wp_create_nonce( 'title_example' );
    wp_localize_script( 'ajax-script', 'my_ajax_obj', array(
        'ajax_url' => admin_url( 'admin-ajax.php' ),
        'nonce'     => $title_nonce,
    ) );
}

```

## AJAX行动

服务器端PHP代码的另一个主要部分是接收POSTed数据的实际AJAX处理程序，执行一些操作，然后将相应的响应发送回浏览器。这采用WordPress动作钩子的形式。您使用哪个挂钩标签取决于用户是否登录，并且您的jQuery脚本作为action : value传递的值是多少。

**注意:** `$_GET` , `$_POST` and `$_COOKIE` vs `$_REQUEST`

您可能已经使用一个或多个PHP超级全局变量（如`$_GET`或`$_POST`）从表单或Cookie（使用`$_COOKIE`）检索值。也许你更喜欢`$_REQUEST`，或至少看过它使用。这很酷 - 无论请求方法，POST或GET，它将具有表单值。适用于使用这两种方法的页面。除此之外，它还具有cookie值。一站式购物！其中有它的悲剧性缺陷。在名称冲突的情况下，cookie值将覆盖任何表单值。因此，糟糕的演员在他们的浏览器上制作一个假冒的cookie是很容易的，这将覆盖您可能期望的请求的任何表单值。`$_REQUEST`是黑客将任意数据注入到表单值中的简单路径。要特别安全，坚持特定的变量，避免一切都适合。

由于我们的AJAX交换是用于插件的设置页面，所以用户必须登录。如果从jQuery部分回想起来，action : value是“my\_tag\_count”。这意味着我们的动作钩子标签将是wp\_ajax\_my\_tag\_count。如果我们的AJAX交换机被当前未登录的用户使用，则动作挂钩标签将是wp\_ajax\_nopriv\_my\_tag\_count用于挂起动作的基本代码如下所示：

```

add_action( 'wp_ajax_my_tag_count', 'my_ajax_handler' );
function my_ajax_handler() {
    // Handle the ajax request
    wp_die(); // All ajax handlers die when finished
}

```

您的AJAX处理程序应该做的第一件事情是验证jQuery与check\_ajax\_referer ( ) 发送的随机数，该值应与脚本入队时本地化的值相同。

```

check_ajax_referer( 'title_example' );

```

提供的参数必须与之前提供的参数wp\_create\_nonce ( ) 相同。如果随机数不检出，该函数就会死机。如果这

是一个真正的现在，现在使用它，这个价值不再是好的。然后，您将生成一个新的并将其发送到回调脚本，以便它可以用于下一个请求。但是由于WordPress的内容有二十四个小时的好处，所以你不需要做任何事情，但检查它。

### ##数据

随着nonce不可用，我们的处理程序可以处理由\$\_POST ['title']中包含的jQuery脚本发送的数据。我们可以使用update\_user\_meta ( ) 将用户的选择保存在用户元中。

```
update_user_meta( get_current_user_id(), 'title_preference', $_POST['title']);
```

然后，我们构建一个查询以获取所选标题标签的发布数。

```
$args = array(
    'tag' => $_POST['title'],
);
$query = new WP_Query( $args );
```

最后我们可以将响应发送回jQuery脚本。有几种传输数据的方法。在我们处理我们的例子的细节之前，我们来看一些选项。

## XML

对XML的PHP支持留下了一些希望。幸运的是，WordPress提供了WP\_Ajax\_Response类，使任务更容易。WP\_Ajax\_Response类将生成XML格式的响应，为标题设置正确的内容类型，输出响应xml，然后死 - 确保正确的XML响应。

## JSON

这种格式是轻量级和易于使用的，WordPress提供的wp\_send\_json函数来json编码您的响应，打印它，并死 - 有效地替换WP\_Ajax\_Response。WordPress还提供了wp\_send\_json\_success和wp\_send\_json\_error函数，这些函数允许在JS中触发适当的done ( ) 或fail ( ) 回调。

### ##其他

只要发送者和接收者协调一致，就可以传输任何你喜欢的数据。诸如逗号分隔或制表符分隔的文本格式是许多可能性之一。对于少量的数据，发送原始流可能是足够的。这就是我们将使用我们的例子 - 我们将发送实际的替换HTML，没有别的。

```
echo $_POST['title'].' ('.$query->post_count.') ';
```

在现实世界的应用程序中，您必须考虑到由于某种原因可能会失败的可能性 - 例如，数据库服务器可能已关闭。响应应该允许这种偶然性，并且接收响应的jQuery脚本应该相应地行动，或许告诉用户稍后再试。

## die

当处理程序完成所有任务时，它需要死机。如果您使用WP\_Ajax\_Response或wp\_send\_json \*函数，则会自动为您处理。如果没有，只需使用WordPress wp\_die ( ) 函数。

```
wp_die();  
// That's all folks!
```

## AJAX处理者总结

我们的示例的完整AJAX处理程序如下所示：

```
//JSON  
function my_ajax_handler() {  
    check_ajax_referer( 'title_example' );  
    update_user_meta( get_current_user_id(), 'title_preference', $_POST['title'] );  
};  
$args = array(  
    'tag' => $_POST['title'],  
);  
$the_query = new WP_Query( $args );  
wp_send_json( $_POST['title'] . ' (' . $the_query->post_count . ') ' );  
}  
  
//Other  
function my_ajax_handler() {  
    check_ajax_referer( 'title_example' );  
    update_user_meta( get_current_user_id(), 'title_preference', $_POST['title'] );  
};  
$args = array(  
    'tag' => $_POST['title'],  
);  
$the_query = new WP_Query( $args );  
echo $_POST['title'] . ' (' . $the_query->post_count . ') ' ;  
wp_die(); // All ajax handlers should die when finished  
}
```

# Heartbeat API

Heartbeat API是一种内置于WordPress的简单服务器轮询API，可实现近实时前端更新。

## 怎么运行的

当页面加载时，客户端心跳代码设置一个间隔（称为“tick”），每15-60秒运行一次。当它运行时，心跳收集要通过jQuery事件发送的数据，然后将其发送到服务器并等待响应。在服务器上，一个admin-ajax处理程序将传递的数据，准备一个响应，过滤响应，然后以JSON格式返回数据。客户端接收这些数据，并触发一个最终的jQuery事件来指示数据已被接收。

自定义心跳事件的基本过程是：

- 向要发送的数据添加其他字段（JS心跳发送事件）
- 检测PHP中发送的字段，并添加其他响应字段（heartbeat\_received过滤器）
- 处理返回的数据在JS（JS heartbeat-tick）

（您可以选择仅使用一个或两个这些事件，具体取决于您需要的功能。）

## 使用API

使用心跳API需要两个独立的功能：在JavaScript中发送和接收回调，以及服务器端过滤器来处理PHP中传递的数据。

## 发送数据到服务器

当心跳发送数据到服务器时，您可以包括自定义数据。这可以是要发送到服务器的任何数据，也可以是一个简单的真实值，表示您期望数据。

```
jQuery( document ).on( 'heartbeat-send', function ( event, data ) {  
    // Add additional data to Heartbeat data.  
    data.myplugin_customfield = 'some_data';  
});
```

## 在服务器上接收和响应

在服务器端，您可以检测此数据，并向响应中添加其他数据。

```
// Add filter to receive hook, and specify we need 2 parameters.  
add_filter( 'heartbeat_received', 'myplugin_receive_heartbeat', 10, 2 );  
  
/**  
 * Receive Heartbeat data and respond.
```

```

*
* Processes data received via a Heartbeat request, and returns additional data
to pass back to the front end.
*
* @param array $response Heartbeat response data to pass back to front end.
* @param array $data Data received from the front end (unslashed).
*/
function myplugin_receive_heartbeat( $response, $data ) {
    // If we didn't receive our data, don't send any back.
    if ( empty( $data['myplugin_customfield'] ) ) {
        return $response;
    }

    // Calculate our data and pass it back. For this example, we'll hash it.
    $received_data = $data['myplugin_customfield'];

    $response['myplugin_customfield_hashed'] = sha1( $received_data );
    return $response;
}

```

## 处理响应

返回到前端，然后可以处理接收到的数据。

```

jQuery( document ).on( 'heartbeat-tick', function ( event, data ) {
    // Check for our data, and use it.
    if ( ! data.myplugin_customfield_hashed ) {
        return;
    }

    alert( 'The hash is ' + data.myplugin_customfield_hashed );
});

```

并不是每个功能都需要这三个步骤。例如，如果您不需要将任何数据发送到服务器，则可以仅使用后两个步骤。

# 概要

以下是上述讨论中的所有示例代码段，汇编成两个完整的代码页：一个用于jQuery，另一个用于PHP。

## PHP

此代码驻留在您的一个插件页面上。

```
<?php add_action('admin_enqueue_scripts', 'my_enqueue');
function my_enqueue($hook) {
    if( 'myplugin_settings.php' != $hook) return;
    wp_enqueue_script( 'ajax-script',
        plugins_url( '/js/myjquery.js', __FILE__ ),
        array('jquery')
    );
    $title_nonce = wp_create_nonce('title_example');
    wp_localize_script('ajax-script', 'my_ajax_obj', array(
        'ajax_url' => admin_url( 'admin-ajax.php' ),
        'nonce'    => $title_nonce,
    ));
}

add_action('wp_ajax_my_tag_count', 'my_ajax_handler');
function my_ajax_handler() {
    check_ajax_referer('title_example');
    update_user_meta( get_current_user_id(), 'title_preference', $_POST['title']
);
    $args = array(
        'tag' => $_POST['title'],
    );
    $the_query = new WP_Query( $args );
    echo $_POST['title'].' ('.$the_query->post_count.') ';
    wp_die(); // all ajax handlers should die when finished
}
```

## ##查询

此代码位于您的插件文件夹下面的文件js/myjquery.js中。

```
jQuery(document).ready(function($) { //wrapper
    $(".pref").change(function() { //event
        var this2 = this; //use in callback
        $.post(my_ajax_obj.ajax_url, { //POST request
            _ajax_nonce: my_ajax_obj.nonce, //nonce
            action: "my_tag_count", //action
            title: this.value //data
```

```
    }, function(data) { //callback
        this2.nextSibling.remove(); //remove the current title
        $(this2).after(data); //insert server response
    });
});
});
```

在存储首选项之后，将所得的帖子计数添加到所选标题。

## 更多信息

---

- [How To Use AJAX In WordPress](#)
- [AJAX for WordPress](#)

# 计划任务

---

## ##什么是WP-Cron

Cron是在UNIX系统上可用的基于时间的任务调度系统。 WP-Cron是WordPress如何处理在WordPress中调度基于时间的任务。使用WP-Cron的几个WordPress核心功能，如检查更新和发布预定的帖子。

WP-Cron的工作原理是：在每个页面加载时，都会检查一个计划任务的列表，以查看需要运行的内容。计划运行的任何任务将在该页面加载期间运行。 WP-Cron不像系统cron那样持续运行;它只是在页面加载时触发。如果您将任务计划在下午2:00，并且在5:00 PM之前没有页面加载，则可能会发生计划错误。

## ##为什么要用WP-Cron

为什么要用WP-Cron？许多托管服务是共享的，不提供对系统cron的访问，但WordPress核心和许多插件确实需要一个cron系统来执行基于时间的任务。 Cron是一个有用的工具，因此是WP-Cron的开始。虽然在特定时间可能无法运行，WP-Cron将及时完成您的任务。使用WordPress API是一种更简单的方法来设置cron任务而不是在WordPress之外。

使用系统cron，如果时间过去，任务没有运行，它将丢失，永远不会运行。 WP-Cron将运行任务，无论它们多大。任务将坐在队列中，直到加载页面来触发它们，因此任务不会丢失。



# 了解WP-Cron计划

## 了解WP-Cron计划

与传统的系统cron不同，即将特定时间（即每小时5分钟以上的小时）调度任务，WP-Cron使用间隔来模拟系统cron。WP-Cron被赋予了第一个任务的时间和一个间隔，以秒为单位的时间，之后重复任务。例如，如果您计划任务从下午2:00开始，间隔时间为300秒（5分钟），则任务将首先在下午2:00运行，然后在下午2:05和之后每5分钟运行一次。

为了简化调度任务，WordPress提供了三个默认间隔，也是添加自定义间隔的简单方法。

WordPress提供的默认间隔是：

- hourly
- twicedaily
- daily

要添加自定义间隔，您可以创建一个过滤器，例如：

```
add_filter( 'cron_schedules', 'example_add_cron_interval' );

function example_add_cron_interval( $schedules ) {
    $schedules['five_seconds'] = array(
        'interval' => 5,
        'display'  => esc_html__( 'Every Five Seconds' ),
    );

    return $schedules;
}
```

此过滤器函数创建一个新的间隔，这将允许我们每五秒运行一次cron任务。

**注意：**所有时间间隔为秒。

# 安排WP-Cron 事件

## 计划重复的任务

为了让您的任务执行，您必须创建自己的自定义钩子并给该钩子执行一个函数的名称。这是非常重要的一步。忘记它，你的任务永远不会运行。

以下将创建钩子。第一个参数是挂钩的名称，第二个参数是要调用的函数的名称。

```
add_action( 'bl_cron_hook', 'bl_cron_exec' );
```

现在对实际的任务调度。另一个重要的注意事项是WP-Cron在调度任务时是天真的。任务由为任务提供的钩子驱动，但是如果您多次调用wp\_schedule\_event()，即使使用相同的钩子名称，事件将被多次调度。如果您的代码在每个页面加载任务，这可能导致任务安排几千次。可能不是一个好主意 WordPress提供了一个名为wp\_next\_scheduled()的方便函数来检查是否已经安排了一个特定的钩子。

wp\_next\_scheduled()取一个参数，即挂钩的名称。它将返回一个包含下一个执行时间戳或false的字符串，表示该任务未被安排。它像这样使用：

```
wp_next_scheduled( 'bl_cron_hook' )
```

使用wp\_schedule\_event()完成计划循环任务。此函数需要三个必需参数，另外一个附加参数是可以传递给执行wp-cron任务的函数的数组。我们将重点关注前三个参数。参数如下：

- \$timestamp - 此任务应该首次执行的UNIX时间戳
- \$recurrence - 任务将以秒为单位重复的间隔的名称
- \$hook - 我们要调用的自定义钩子的名称

我们将使用之前创建的5秒间隔和钩子，如：

```
wp_schedule_event( time(), '5seconds', 'bl_cron_hook' );
```

记住，我们需要首先确保任务尚未安排，其完整代码如下：

```
if ( ! wp_next_scheduled( 'bl_cron_hook' ) ) {  
    wp_schedule_event( time(), '5seconds', 'bl_cron_hook' );  
}
```

## 计划外任务

当您不再需要安排任务时，可以使用`wp_unschedule_event()`取消任务。此功能需要以下两个参数：

- `$timestamp` - 任务下一次出现的时间戳
- `$hook` - 要调用的自定义钩子的名称

此功能不仅会取消对时间戳指示的任务进行调度，还会取消调度任务的所有将来的发生。因为你可能不知道下一个任务的时间戳是有功能的，它将为你找到它的`wp_next_schedule()`。`wp_next_scheduled()`需要一个参数（我们关心的）：

- `$hook` - 被调用来执行任务的钩子的名称

把它放在一起，代码如下：

```
$timestamp = wp_next_scheduled( 'bl_cron_hook' );  
wp_unschedule_event( $timestamp, 'bl_cron_hook' );
```

当您不再需要它们时，卸载任务非常重要，因为WordPress将继续尝试执行任务，即使它们不再使用。记住卸载任务的一个重要的地方是插件停用。不幸的是，WordPress.org插件目录中有很多插件在自己之后不会被清理。如果您发现其中一个插件，请让作者知道更新代码。WordPress提供了一个名为`register_deactivation_hook()`的函数，允许开发人员在插件被停用时运行一个函数。这是非常简单的设置和看起来像：

```
register_deactivation_hook( __FILE__, 'bl_deactivate' );  
  
function bl_deactivate() {  
    $timestamp = wp_next_scheduled( 'bl_cron_hook' );  
    wp_unschedule_event( $timestamp, 'bl_cron_hook' );  
}
```

# 将WP-Cron挂接到系统任务计划程序中

如前所述，WP-Cron不会连续运行，如果有关键任务必须按时运行，这可能是一个问题。有一个简单的解决方案。只需设置系统的任务调度程序，就可以按需要的间隔（或者在特定的时间）运行。最简单的解决方案是使用工具向wp-cron.php文件发出Web请求。

在您的系统上安排任务后，还有一个步骤要完成。WordPress将在每次加载页面上继续运行WP-Cron。这不再需要，并将有助于您的服务器上额外的资源使用。可以在wp-config.php文件中禁用WP-Cron。打开wp-config.php文件进行编辑，并添加以下行：

```
define( 'DISABLE_WP_CRON', true );
```

## Windows

Windows调用其基于时间的调度系统任务计划程序。可以通过控制面板中的“管理工具”进行访问。

如何设置任务因服务器设置而异。一种方法是使用PowerShell和基本任务。创建基本任务后，以下命令可用于调用WordPress Cron脚本。

```
powershell "Invoke-WebRequest http://YOUR_SITE_URL/wp-cron.php"
```

## Mac OS X and Linux

Mac OS X和Linux都使用cron作为其基于时间的调度系统。通常从终端接入

```
crontab -e
```

命令。应该注意的是，任务将作为常规用户运行，也可以以root身份运行，具体取决于运行该命令的系统用户。

Cron具有需要遵循的特定语法，并包含以下部分：

- Minute
- Hour
- Day of month
- Month
- Day of week
- Command to execute
- plugin-wp-cron-cron-scheduling

如果一个命令应该运行，而不管其中一个时间段应该使用星号（\*）。例如，如果您希望每15分钟运行一次命

将WP-Cron挂接到系统任务计划程序中

令，无论时间，日期或月份如何，它将如下所示：

```
15 * * * * * command
```

许多服务器都有

```
wget
```

这是一个简单的工具来调用WordPress Cron脚本。

```
wget http://YOUR_SITE_URL/wp-cron.php
```

每天晚上在午夜触发您网站WordPress Cron的每日电话可能类似于

```
0 0 * * * wget http://YOUR_SITE_URL/wp-cron.php
```

# WP-Cron简单测试

在本教程中，我们将创建一个插件，每5秒钟运行一个任务并显示一条消息。为了测试这个，我们将直接在浏览器中加载wp-cron.php文件并将数据输出到显示器，否则我们必须执行一些其他操作，也许在数据库中，因为输出通常不会显示在现场。所以让我们来看看初步的步骤来快速得到这个设置。

## ##创建插件文件

在wp-content/plugins文件夹中创建文件夹'my-wp-cron-test'和文件'my-wp-cron-test.php'。很明显，你可以命名任何你想要的。这个名字只是描述我们的预期用途。

打开PHP文件进行编辑，并插入以下几行

```
<?php
/*
Plugin Name: My WP-Cron Test
*/
```

此文本将在您的wp-admin插件菜单中设置显示和激活的插件。确保启用该插件。

## 测试代码

打开浏览器并将其指向YOUR\_SITE\_URL/wp-cron.php

## 查看所有当前安排的任务

WordPress有一个未记录的函数\_get\_cron\_array，它返回所有当前调度任务的数组。我们将使用粗略但有效的方法使用var\_dump转储所有任务。为了方便使用，在插件中放置以下代码：

```
echo '<pre>'; print_r( _get_cron_array() ); echo '</pre>';
```

进入一个简单的调用功能像：

```
function bl_print_tasks() {
    echo '<pre>'; print_r( _get_cron_array() ); echo '</pre>';
}
```

# 国际化

---

## ##什么是国际化？

国际化是开发您的插件的过程，因此可以轻松地将其翻译成其他语言。国际化通常缩写为i18n（因为i和n之间有18个字母）。

## ##为什么国际化重要？

WordPress用于世界各地，在英语不是主要语言的国家。WordPress插件中的字符串需要以特殊的方式编码，以便可以轻松翻译成其他语言。作为开发人员，您可能没有一个容易的时间为所有用户提供本地化;然而，任何翻译器都可以成功本地化插件，而无需修改源代码本身。

阅读更多关于“如何国际化您的插件”。

## ##资源

视频：i18n：准备您的WordPress主题为世界

视频：国际化：全球插件和主题幻灯片

视频：大日本：主题和国际化指南

视频：丢失在翻译i18n和WordPress

国际化和本地化您的WordPress主题

国际化：你可能做错了

更多国际化乐趣

使用wp\_localize\_script，它是真棒

了解\_n\_noop（）

语言包101 - 准备工作

翻译WordPress插件和主题：不要聪明

如何加载主题和插件翻译

正确的方式加载WordPress语言文件

# 本地化

## 什么是本地化？

本地化描述了翻译国际化插件的后续过程。 本地化缩写为l10n（因为l和n之间有10个字母）

## 本地化文件

POT（便携式对象模板）文件#POT（便携式对象模板）文件

该文件包含插件中的原始字符串（英文）。 这是一个POT文件的例子：

```
#: plugin-name.php:123
msgid "Page Title"
msgstr ""
```

## PO（便携式对象）文件

每个翻译器都会使用POT文件，并以自己的语言翻译msgstr部分。 结果是与POT具有相同格式的PO文件，但具有翻译和一些特定标题。 每个语言有一个PO文件。

## MO（机器对象）文件

从每个翻译的PO文件建立一个MO文件。 这些是gettext函数实际使用的机器可读的二进制文件（它们不关心.POT或.PO文件），是PO文件的“编译”版本。 转换使用msgfmt工具完成。 通常，应用程序可以相应地使用多个大的逻辑可翻译模块和不同的MO文件。 文本域是每个模块的句柄，它具有不同的MO文件。

## 生成POT文件

POT文件是您需要交给翻译人员的文件，以便他们可以做他们的工作。 POT和PO文件可以轻松地重新命名，以便更改文件类型，而不会有任何问题。 提供POT文件和插件是一个好主意，因此翻译人员不必专门询问您。 有几种方式为您的插件生成POT文件：

## 插件目录管理工具

如果您的插件托管在WordPress.org插件目录中，请转到您的“管理”页面，然后单击“生成POT文件”部分的“继续”。

- 插件管理区域

然后点击获取POT下载POT文件。

- 生成POT



## WordPress i18n工具

如果您的插件不在目录中，您可以从SVN中检出WordPress Trunk目录（请参阅使用Subversion了解SVN）。您需要检出整个中继线，因为wordpress-i18n工具使用WordPress核心的代码生成POT。您需要在您的服务器/计算机上安装gettext（GNU国际化实用程序）软件包和PHP才能运行以下命令。

打开命令行并将目录更改为语言文件夹。

```
cd theme-name/languages
```

您可以在命令行中运行makepot.php脚本，如下所示：

```
php path/to/makepot.php wp-plugin path/to/your-plugin-directory
```

完成之后，您应该在当前目录中看到POT文件。

## Poedit

您也可以在本地使用Poedit进行翻译。这是所有主要操作系统的开源工具。免费的Poedit默认版本支持使用Gettext函数手动扫描所有源代码。它的专业版也具有一键扫描WordPress插件。生成po文件后，您可以将文件重命名为POT。如果生成了一个mo，那么您可以删除该文件，因为它不需要。如果您没有专业版本，您可以轻松获取空白POT，并将其用作POT文件的基础。将空白POT放入语言文件夹后，您可以单击Poedit中的“更新”，使用字符串更新POT文件。

国际化 - 本地化03

## Grunt 的任务

甚至有一些拙劣的任务可以用来创建POT。grunt-wp-i18n&grunt-pot

要设置它，您需要安装node.js。这是一个简单的安装。然后，您需要在要使用grunt的目录中安装grunt。这通过命令行完成。可以放置在插件根目录中的Grunt.js和package.json的例子。您可以在命令行中使用简单的命令来执行grunt任务。

## 翻译PO文件

翻译PO文件有多种方法。

您可以使用文本编辑器输入翻译。在文本编辑器中，它将如下所示。

```
#: plugin-name.php:123
msgid "Page Title"
msgstr ""
```

您输入引号之间的翻译。 对于德语翻译，它看起来像这样。

```
#: plugin-name.php:123
msgid "Page Title"
msgstr "Seitentitel"
```

您也可以在翻译时使用Poedit。这是所有主要操作系统的开源工具。免费的Poedit默认版本支持使用Gettext函数手动扫描所有源代码。它的专业版也具有一键扫描WordPress插件和主题。

第三种选择是使用在线翻译服务。一般的想法是您上传POT文件，然后您可以授权用户或翻译人员翻译您的插件。这允许您跟踪更改，始终具有最新的翻译，并减少翻译两次。

François-Xavier Bérnard正在运行WP-Translations，这是一个“翻译与开发人员会面”的社区。它运行在Transifex上，您可以提交作为开发人员翻译的项目，或翻译用于您的语言的现有插件和主题。这是伟大的，因为那里有现有的翻译。

以下是一些可用于在线翻译PO文件的工具：

- Transifex
- WebTranslateIt
- Poeditor
- Google Translator Toolkit
- GlotPress
- You can even use WordPress plugins to do the translation.

## Codestyling本地化

翻译后的文件将被保存为my-plugin- {locale} .mo。语言环境是您在文件wp-config.php中的常量WPLANG中定义的语言代码和/或国家/地区代码。例如，德语的语言环境是de\_DE。从上面的代码示例，文本域是'my-plugin'，因此德语MO和PO文件应该命名为my-plugin-de\_DE.mo和my-plugin-de\_DE.po。有关语言和国家/地区代码的更多信息，请参阅使用语言安装WordPress。

#生成MO文件

## 命令行

程序msgfmt用于创建MO文件。msgfmt是Gettext包的一部分。否则可以使用命令行。典型的msgfmt命令如下所示：

Unix操作系统

```
msgfmt -o filename.mo filename.po
```

Windows操作系统

```
msgfmt -o filename.mo filename.po
```

如果您有很多PO文件一次转换，您可以作为批处理运行它。 例如，使用bash命令：

Unix操作系统

```
# Find PO files, process each with msgfmt and rename the result to MO
for file in `find . -name "*.po"` ; do msgfmt -o ${file/.po/.mo} $file ; done
```

Windows操作系统

对于Windows，您需要先安装Cygwin。

[创建一个potomo.sh](#)

```
#!/bin/sh
# Find PO files, process each with msgfmt and rename the result to MO
for file in `usr/bin/find . -name '*.po'` ; do /usr/bin/msgfmt -o ${file/.po/.mo} $file ; done
```

您可以在命令行中运行此命令。

```
cd C:/path/to/language/folder/my-plugin/languages & C:/cygwin/bin/bash -c /cygdrive/c/path/to/script/directory/potomo.sh
```

## Poedit

msgfmt也集成在Poedit中，允许您使用它来生成MO文件。 首选项中有一个设置可以启用或禁用它。

国际化 - 本地化04

## Grunt task

有grunt-po2mo将转换所有的文件。

#好翻译提示

## 不要翻译字面，有机地翻译

双语或多语言你无疑知道你所说的语言有不同的结构，节奏，色调和变化。翻译的消息不需要像英语一样的结构：采取所提出的想法，并提出一种以自然的方式表达目标语言的消息。创建相同消息和等效消息之间的区别是：不要复制，替换。即使在消息中有更多的结构性项目，如果您觉得对目标受众更合乎逻辑或更适合您，您就有创造性的许可来适应和改变。

尝试保持相同水平的手续（或非正式）

每个消息具有不同的正式或非正式级别。您的目标语言中每个消息使用的正式或非正式级别都是您自己（或与您

的团队) 进行比较, 但是WordPress消息 (特别是信息性消息) 往往会有礼貌的非正式英语口语尝试在您的文化背景下完成目标语言中的等效项目。

### ##不要使用俚语或受众特定的术语

在博客中可以预期一些术语, 但不要使用只有“在”人群中获得的口语化。如果未经开发的博主以您的语言安装WordPress, 他们会知道这个术语是什么意思吗? 像pingback, trackback和feed这样的字是这个规则的例外; 它们是通常很难翻译的术语, 许多翻译选择以英文留下。

### 以其语言阅读其他软件的本地化

如果您遇到困难或需要方向, 请尝试阅读其他流行的软件工具的翻译, 以了解常用的术语, 如何解决方法等等。当然, WordPress有自己的语气和感觉, 所以保持在阅读其他本地化时, 请记住, 但是请随意挖掘UI术语等来保持与您的语言的其他软件的一致性。

### ##使用本地化

将本地化文件放在语言文件夹中, 无论是插件语言文件夹还是通常在wp-content下的插件语言文件夹中的WordPress 3.7。完整的路径将是wp-content / languages / plugins / my-plugin-fr\_FR.mo。

从WordPress 4.0开始, 您可以在“常规设置”中更改语言。如果您没有看到任何选项或要切换到语言, 请执行以下步骤:

将wp-config.php内的WPLANG定义为您选择的语言。例如, 如果你想使用法语, 你会有:

```
define ( 'WPLANG', 'fr_FR' );
```

转到wp-admin / options-general.php或“设置” - > “常规”

在“网站语言”下拉列表中选择您的语言

转到wp-admin / update-core.php

点击“更新翻译”, 如果可用

核心翻译文件 (如有) 可以下载

### ##资源

- 为您的主题或插件创建.pot文件
- 如何国际化WordPress插件
- 翻译你的主题
- 空白WordPress POT
- 改进了i18n WordPress工具
- 如何快速更新翻译
- GitHub / Transifex之间的工作流程
- Gist : 完成本地化Grunt任务
- WordPress.tv标签 : i18n , 国际化和翻译

# 如何国际化您的插件

为了使您的应用程序中的字符串可以翻译，您必须将原始字符串打包到一组特殊功能的调用中。

## Gettext简介

WordPress使用i18n的gettext库和工具。如果你在网上看，你会看到 `_()` 函数，它引用了本地PHP gettext兼容的翻译功能。使用WordPress，您应该使用 `__()` WordPress定义的PHP函数。如果您想获得更广泛和更深入的gettext视图，我们建议您阅读gettext在线手册。

## 文字域

使用文本域来表示属于该插件的所有文本很重要。文本域是唯一的标识符，可以确保WordPress能够区分所有加载的翻译。这增加了可移植性，并通过已经存在的WordPress工具更好地发挥作用。文本域必须与插件的插槽相匹配。如果您的插件是一个名为my-plugin.php的文件，或者它包含在一个名为my-plugin的文件夹中，则域名应为my-plugin。如果您的插件托管在wordpress.org上，那么它必须是插件URL ([wordpress.org/plugins/](https://wordpress.org/plugins/)) 中的一小部分。

文本域名必须使用破折号而不是下划线。

字符串示例：

```
__( 'String (text to be internationalized)', 'text-domain' );
```

将“文本域”更改为插件的插件。

**警告：**不要在gettext函数的文本域部分使用变量名。不要做这个快捷方式：`__( 'Translate me.', $text_domain );`

文本域也需要添加到插件头。即使插件被禁用，WordPress也使用它来国际化你的插件元数据。文本域应与加载文本域时使用的文本域相同。

标题示例：

```
/*
 * Plugin Name: My Plugin
 * Author: Plugin Author
 * Text Domain: my-plugin
 */
```

## 域路径

使用域路径，以便当插件被禁用时，WordPress知道在哪里找到翻译。 仅当翻译位于单独的语言文件夹中时才有用。 例如，如果.mo文件位于languages文件夹中，则Domain Path将是“/languages”，必须用第一个斜杠写入。 默认为插件的languages文件夹：

标题示例：

```
/*
 * Plugin Name: My Plugin
 * Author: Plugin Author
 * Text Domain: my-plugin
 * Domain Path: /languages
 */
```

## 基本字符串

最常用的是 `__( )`。 它只是返回其参数的翻译：

```
__( 'Blog Options', 'my-plugin' );
```

另一个简单的是 `_e()`，它输出其参数的翻译。 而不是写：

```
echo __( 'WordPress is the best!', 'my-plugin' );
```

你可以使用较短的：

```
_e( 'WordPress is the best!', 'my-plugin' );
```

## 变量

如果您在字符串中使用变量，如下面的示例，您将使用占位符。

```
echo 'Your city is $city.'
```

解决方案是使用printf系列函数。 特别有用的是printf和sprintf。 以下是正确的解决方案：

```
printf(
    /* translators: %s: Name of a city */
    __( 'Your city is %s.', 'my-plugin' ),
    $city
);
```

请注意，这里的翻译字符串只是模板“你的城市是%s”，这在源代码和运行时都是一样的。

如果字符串中有多个占位符，建议您使用参数交换。在这种情况下，单引号（'）是强制性的：双引号（"）将告诉php将\$ s解释为s变量，这不是我们想要的。

```
printf(
    /* translators: 1: Name of a city 2: ZIP code */
    __( 'Your city is %1$s, and your zip code is %2$s.', 'my-plugin' ),
    $city,
    $zipcode
);
```

这里的邮政编码正在城市名后显示。在某些语言中，以相反的顺序显示邮政编码和城市更为合适。在上面的例子中使用%s的前缀，就允许这样的情况。因此，翻译可以写成：

```
printf(
    /* translators: 1: Name of a city 2: ZIP code */
    __( 'Your zip code is %2$s, and your city is %1$s.', 'my-plugin' ),
    $city,
    $zipcode
);
```

重要！此代码不正确。

```
// This is incorrect do not use.
_e( "Your city is $city.", 'my-plugin' );
```

翻译的字符串是从源中提取出来的，所以翻译人员会得到这个短语：“你的城市是\$ city”。

但是在应用程序中，将会被称为“你的城市是伦敦”。gettext将不会找到适合的翻译，并返回其论点：“你的城市是伦敦”。不幸的是，它没有正确翻译。

## 多个

### ##基本多元化

如果在项目数量更改时有变化的字符串。在英语中你有“一个评论”和“两个意见”。在其他语言中，您可以有多个复数形式。要在WordPress中处理这个问题，可以使用\_n（）函数。

```
printf(
    _n(
        '%s comment',
        '%s comments',
```

```
        get_comments_number(),
        'my-plugin'
    ),
    number_format_i18n( get_comments_number() )
);
```

`_n()` 接受4个参数：

- singular – 字符串的单数形式（注意，它可以用于某些语言中的一个以外的数字，因此应使用“%s项”而不是“一个项”）
- plural – 字符串的复数形式
- count – 对象的数量，这将决定是否应该返回单数或复数形式（有多种语言，有两种以上的形式）
- text domain – 插件文本域

功能的返回值是正确的翻译形式，对应于给定的计数。

##稍后完成

您首先使用 `_n_noop()` 或 `_nx_noop()` 设置多个字符串。

```
$comments_plural = _n_noop(
    '%s comment.',
    '%s comments.'
);
```

然后在稍后的代码中，您可以使用 `translate_nooped_plural()` 来加载字符串。

```
printf(
    translate_nooped_plural(
        $comments_plural,
        get_comments_number(),
        'my-plugin'
    ),
    number_format_i18n( get_comments_number() )
);
```

## 背景消歧

有时一个术语在多个语境中被使用，虽然它是英文中同一个词，但是在其他语言中它必须被不同地翻译。例如，Post可以用作动词“点击这里发表你的评论”，作为名词“编辑这篇文章”。在这种情况下，应该使用 `_x()` 或 `_ex()` 函数。它类似于 `_()` 和 `_e()`，但它有一个附加的参数 - 上下文：

```
_x( 'Post', 'noun', 'my-plugin' );
_x( 'Post', 'verb', 'my-plugin' );
```



在这两种情况下使用此方法，我们将获得原始版本的字符串注释，但翻译器将看到两个用于翻译的注释字符串，每个字符串在不同的上下文中。

请注意，与 `__()` 类似，`_x()` 具有回调版本：`_ex()`。前面的例子可以写成：

```
_ex( 'Post', 'noun', 'my-plugin' );
_ex( 'Post', 'verb', 'my-plugin' );
```

使用您觉得增强易读性和易于编码的任何一个。

### ##说明

所以翻译人员知道如何翻译一个字符串`__( 'g:i:s a' )`，您可以在源代码中添加一个澄清的注释。它必须从翻译人员开始：在`gettext`调用之前成为最后一个PHP注释。这是一个例子：

```
/* translators: draft saved date format, see http://php.net/date */
$saved_date_format = __( 'g:i:s a' );
```

它还用于解释字符串中的占位符，如`_n_noop( <strong>版本%1 $ s </ strong> 已解决%2 $ s错误, <strong>版本%1 $ s </ strong> “解决了%2 $ s的错误” )`。

```
/* translators: 1: WordPress version number, 2: plural number of bugs. */
_n_noop( '<strong>Version %1$s</strong> addressed %2$s bug.',
        '<strong>Version %1$s</strong> addressed %2$s bugs.' );
```

## 换行字符

Gettext不喜欢`\r`（ASCII码：13）在可翻译的字符串，所以请避免它，并使用`\n`代替。

### ##空字符串

空字符串保留用于内部Gettext使用，您不得尝试将空字符串国际化。它没有任何意义，因为翻译者不会看到任何上下文。

如果您有一个有效的用例来使一个空字符串国际化，请添加上下文以帮助翻译人员，并与Gettext系统保持一致。

## 处理JavaScript文件

使用`wp_localize_script()`将已翻译的字符串或其他服务器端数据添加到先前排入的脚本。

### ##转义字符串

逃避所有的字符串是好的，这样翻译者就不能运行恶意代码。有几个与国际化功能相结合的逃生功能。

### ##本地化功能

### ##基本功能

- `__()`
- `_e()`
- `_x()`
- `_ex()`
- `_n()`
- `_nx()`
- `_n_noop()`
- `_nx_noop()`
- `translate_nooped_plural()`

## ##翻译和退出功能

必须转义需要翻译并在html标签的属性中使用的字符串。

- `esc_html__()`
- `esc_html_e()`
- `esc_html_x()`
- `esc_attr__()`
- `esc_attr_e()`
- `esc_attr_x()`

## 日期和数字功能

---

- `number_format_i18n()`
- `date_i18n()`

## 写字符串的最佳做法

---

以下是编写字符串的最佳做法

- 使用体面的英式风格 - 尽量减少俚语和缩写。
- 使用整个句子 - 在大多数语言中，单词顺序与英语不同。
- 拆分段落 - 合并相关句子，但不要在一个字符串中包含整个文本页面。
- 不要将前导或尾随空格留在可翻译的短语中。
- 假设翻译时字符串的长度可以翻倍
- 避免不正常的标记和不寻常的控制字符 - 不要包含文本周围的标签
- 不要将不必要的HTML标记放入已翻译的字符串中
- 不要留下翻译的网址，除非他们可以使用其他语言的版本。
- 将变量作为占位符添加到字符串中，如在某些语言中，占位符更改位置。

```
printf(
    __( 'Search results for: %s', 'my-plugin' ),
    get_search_query()
);
```

使用格式字符串而不是字符串连接 - 翻译短语而不是单词 -

```
printf(
    __( 'Your city is %1$s, and your zip code is %2$s.', 'my-plugin' ),
    $city,
    $zipcode
);
```

总是比：

```
__( 'Your city is ', 'my-plugin' ) . $city . __( ', and your zip code is ', 'my-plugin' ) . $zipcode;
```

尝试使用相同的单词和相同的符号，因此不需要翻译多个字符串，例如 `__( 'Posts: ', 'my-plugin' );` 和 `__( 'Posts', 'my-plugin' );`；

##将文本域添加到字符串

您必须将您的文本域作为参数添加到每个`__( )`，`_e( )`和`_n( )`gettext调用中，否则您的翻译将无法正常工作。

例子：

```
__( 'Post' )
```

应该成为

```
__( 'Post', 'my-theme' )
```

```
_e( 'Post' )
```

应该成为

```
_e( 'Post', 'my-theme' )
```

```
_n( 'One post', '%s posts', $count )
```

应该成为

```
_n( 'One post', '%s posts', $count, 'my-theme' )
```

如果您的插件中也有用于WordPress核心的字符串（例如“设置”），那么您还应该添加自己的文本域，否则如果核心字符串发生变化（如果发生），那么它们将变为非翻译。

手动添加文本域可能是一个负担，如果不是在编写代码时连续完成，这就是为什么你可以自动执行：

如果您的插件位于WordPress.org插件目录中，请转到您的“管理”页面，然后滚动到“添加域到Gettext呼叫”。

上传要添加文本域的文件。

点击“获取域文件”。

[WordPress.org Plugin Admin区域](#)

[WordPress.org Plugin Admin区域](#)

除此以外：

将add-textdomain.php脚本下载到要添加文本域的文件夹

在命令行中移动到文件所在的目录

运行此命令创建一个添加了文本域的新文件

```
php add-textdomain.php my-plugin my-plugin.php > new-my-plugin.php
```

如果您希望将add-textdomain.php放在不同的文件夹中，那么您只需要在命令中定义位置。

```
php \path\to\add-textdomain.php my-plugin my-plugin.php > new-my-plugin.php
```

如果您不想输出新文件，请使用此命令。

```
php add-textdomain.php -i my-plugin my-plugin.php
```

如果要更改目录中的多个文件，还可以将目录传递给脚本。

```
php add-textdomain.php -i my-plugin my-plugin-directory
```

完成后，文本域将被添加到文件中所有gettext调用的末尾。如果存在现有的文本域，则不会被替换。

## 加载文本域

**注意：**WordPress 4.6出来之后，翻译现在以[translate.wordpress.org](https://translate.wordpress.org)为优先，因此通过

[translate.wordpress.org](https://translate.wordpress.org) 翻译的插件不再需要 `load_plugin_textdomain()` 了。

如果您不想在您的插件中添加一个 `load_plugin_textdomain()` 调用，则必须将 `readme.txt` 中的至少要求至少为：

您需要使用插件的翻译加载 MO 文件。您可以通过调用函数 `load_plugin_textdomain()` 来加载它们。此调用从插件的基本目录载入 `{text-domain} - {locale}.mo`。语言环境是常规设置下的站点语言设置的语言代码和/或国家/地区代码。有关语言和国家/地区代码的更多信息，请参阅您的语言中的 WordPress。

从上面的代码示例，文本域是 `my-plugin`，因此德语 MO 和 PO 文件应该命名为 `my-plugin-de_DE.mo` 和 `my-plugin-de_DE.po`。

例：

```
function my_plugin_load_plugin_textdomain() {  
    load_plugin_textdomain( 'my-plugin', FALSE, basename( dirname( __FILE__ ) )  
        . '/languages/' );  
}  
add_action( 'plugins_loaded', 'my_plugin_load_plugin_textdomain' );
```

## 语言包

如果您对语言包感兴趣，以及如何进行 [translate.wordpress.org](https://translate.wordpress.org) 的导入，请阅读关于翻译的 Meta 手册页面。

# 国际化安全

在谈论国际化时，安全往往被忽视，但有一些重要的事情要记住。

## ##检查垃圾邮件和其他恶意串

当翻译人员向您提交本地化时，请务必检查，以确保他们的翻译不包括垃圾邮件或其他恶意单词。 您可以使用 Google 翻译将其翻译翻译成母语，以便您可以轻松比较原始和翻译的字符串。

## Escape国际化字符串

你不能相信翻译者只会在本地化中增加良性文本; 如果他们想要，他们可以添加恶意JavaScript或其他代码。 为了保护这一点，重要的是要像国际化的字符串一样处理任何其他不可信任的输入。

如果你输出的字符串，那么应该被转义。

不安全

```
<?php _e( 'The REST API content endpoints were added in WordPress 4.7.', 'your-text-domain' ); ?>
```

安全：

```
<?php esc_html_e( 'The REST API content endpoints were added in WordPress 4.7.', 'your-text-domain' ); ?>
```

或者，有些人选择依赖翻译验证机制，而不是将转义添加到他们的代码中。 验证机制的一个例子是WordPress Polyglots团队用于translate.wordpress.org的编辑器角色。 这可以确保不受信任的贡献者提交的任何翻译都已经被受信任的编辑器验证，然后被接受。

## ##将占位符用于URL

不要在国际化字符串中包含URL，因为恶意翻译可以将其更改为指向不同的URL。 而是使用printf()或sprintf()的占位符。

不安全

```
<?php _e(
    'Please <a href="https://wordpress.org/support/register.php">
    register for a WordPress.org account</a>.',
    'your-text-domain'
); ?>
```

安全：

```
<?php printf(
    _ (
        'Please <a href="%s">register for a WordPress.org account</a>.',
        'your-text-domain'
    ),
    'https://wordpress.org/support/register.php'
); ?>
```

## 编译自己的.mo二进制文件

翻译者通常会发送编译的.mo文件以及纯文本.po文件，但是您应该放弃他们的.mo文件并自己编译，因为您无法知道是否从相应的.po文件中编译，或不同的。如果是根据不同的编译，那么可能会包含垃圾邮件和其他恶意的字符串。

使用PoEdit生成二进制文件将覆盖.po文件中的标题，因此最好从命令行编译它：

```
msgfmt -cv -o /path/to/output.mo /path/to/input.po
```

# WordPress.org

---

如果您希望我们在WordPress.org上托管您的插件，只需要我们为您托管该插件。

如果您刚刚开始，请先考虑如何计划您的插件。

## 特征

---

WordPress.org上托管的所有插件均可访问：

监控下载

获取版本统计信息

收到用户的反馈和评论

通过免费论坛提供支持

我们还有一个[WordPress.org Plugin API](#)，可用于监视插件的统计信息。

## 要求

---

详细的插件指南列表有详细的例子和说明。以下是对最重要方面的简要概述：

您的插件必须与GNU通用公共许可证V2或更高版本兼容。我们强烈建议您使用与WordPress相同的许可证 - “GPLv2或更高版本”。

如果您没有指定兼容的许可证，您检查的内容将被视为GPLv2或更高版本。

插件和开发人员不得做任何非法，不诚实或道德上的冒犯。

所提供的Subversion存储库只能用于功能性WordPress插件。目录是托管网站，而不是列表网站。

该插件不得在公共网站上嵌入外部链接（如“powered by”链接或广告），而不需要明确的用户权限。

## 获取插件托管

---

要将您的插件托管在WordPress.org上，请按照下列步骤操作：

在WordPress.org上注册一个有效的，定期检查的电子邮件地址。如果您是代表公司提交插件，请使用官方公司电子邮件进行验证。

白名单plugins@wordpress.org在您的电子邮件客户端，以确保您接收电子邮件通信。

提交你的插件，简要介绍一下它的作用，以及一个链接到一个完整的，准备去插件的zip。

在14个工作日内，您的插件将由WordPress插件审核小组的成员进行手动审核。您可能会通过电子邮件发送并要求提供更多信息或正确的问题。一旦批准，您将收到一封电子邮件，其中包含有关如何访问Subversion存储库的详细信息，您将在其中存储插件。

通过SVN将您的插件（和自述文件）上传到该存储库后，它将显示在插件目录中。

## 附加阅读

---



如何使用SVN

readme.txt如何工作

插件资源（标题图像和图标）如何工作

特殊用户角色和功能

开发者常见问题

插件安全性

接管一个现有的插件

# 详细插件指南

最后更新：2017年1月30日

WordPress插件目录的目标是为所有WordPress用户（从非技术人员）提供安全的位置，以下载与WordPress项目目标一致的插件。

为此，我们希望为开发人员提供一个简单透明的过程，为目录提供插件。作为我们不断努力使插件目录包含过程更加透明的一部分，我们创建了开发人员指南列表。我们力争为所有开发人员创造一个公平的竞争环境。

如果您有改进文档的建议或有关的问题，请发送电子邮件至[plugins@wordpress.org](mailto:plugins@wordpress.org)并通知我们。

## ##插件提交

为了提交一个插件，有三个步骤：

在WordPress.org上注册一个有效的，定期检查的电子邮件地址。如果您是代表公司提交插件，请使用官方公司电子邮件进行验证。

白名单[plugins@wordpress.org](mailto:plugins@wordpress.org)在您的电子邮件客户端，以确保您收到电子邮件通信。

提交你的插件，简要介绍一下它的作用，以及一个链接到一个完整的，准备去插件的zip。

一旦插件排队等待审核，我们将检查插件的任何问题。大多数问题可以通过遵循以下准则来避免。如果我们发现问题，我们将联系开发人员，并致力于解决问题。

## ##开发者期望

开发人员，具有提交访问权限的所有用户以及正式支持插件的所有用户都应遵守目录指南。违规可能会导致从目录中删除插件或插件数据（以前批准的插件），直到问题解决。插件数据（如用户评论）可能无法恢复，具体取决于违规的性质和同行评审结果。重复违规可能导致所有作者的插件被删除，并且开发人员被禁止在WordPress.org上托管插件。插件开发人员有责任确保他们在WordPress.org上的联系信息是最新和准确的，以便他们收到插件团队的所有通知。

目录中的所有代码应尽可能安全。安全性是插件开发人员的最终责任，但是插件目录最大限度地强化了这一点。如果一个插件被发现有安全问题，它将被关闭，直到情况解决。在极端情况下，WordPress安全团队可能会更新该插件，并为了公众的安全而进行传播。

尽管我们试图解释尽可能多的指导原则的相关解释，但期望明确涵盖每一种情况都是不合理的。如果您不确定插件是否可能违反指南，请通过[plugins@wordpress.org](mailto:plugins@wordpress.org)与我们联系并询问。

## ##指南

- 插件必须与WordPress.org上托管的GNU通用公共许可证v2或任何更高版本兼容。

虽然任何与GPL兼容的许可证都是可以接受的，但建议使用与WordPress相同的许可证 - “GPLv2或更高版本”。所有代码，数据和图像 - 存储在目录中的任何内容必须符合GPL（任何版本，两个或更高版本），无论其创建者如何。包括的第三方库也必须兼容。有关兼容许可证的具体列表，请阅读[gnu.org](http://gnu.org)上的GPL兼容许可证列表。

- 插件开发人员负责他们上传的文件和他们使用的服务。

插件开发人员有责任确保其插件内的所有文件符合指南。预计在上传到SVN之前，将所有包含的文件的许可证从原始源代码确认到图像和图书馆。此外，他们必须遵守其插件使用的所有第三方服务和API的使用条款。如果无法验证图书馆的许可或API的条款，则不能包括它们。

- 您的插件的稳定版本必须可以从其WordPress插件目录页面。

WordPress.org分发的插件的唯一版本是目录中的插件。虽然您可能会在其他地方开发代码，但请记住，用户将从目录下载，而不是您的开发环境。

- 保持你的代码（主要是）人的可读性。

通过使用与p, a, c, k, e, r的混淆特征相似的技术或系统隐藏代码来有意识地隐藏代码，uglify的mangle或不明确的命名约定（如\$ z12sdf813d）在目录中是不允许的。不幸的是，许多人使用这种方法来尝试隐藏恶意代码，如后门或跟踪。此外，WordPress代码旨在让任何人能够学习，编辑和调整。使代码非人类可读迫使未来的开发人员面临不必要的障碍。可以使用最小化的代码，但是尽可能地包括未最终版本。我们建议遵循WordPress核心编码标准。

- 目录中不允许使用试用版。

尝试在其他产品和功能上加强用户的限制是可以接受的。

Upsell通知不应过于突出或令人讨厌。

插件可能不包含被瘫痪或锁定的功能，只能通过付款或升级来解锁。付费功能必须是外部托管服务或单独的插件的一部分，而不是托管在wordpress.org上。

试用期或配额后，插件可能不会禁用包含的功能。

- 目录中允许软件即服务。

作为一些外部第三方服务（例如，视频托管网站）的界面的插件，即使是付费服务也被允许。服务本身必须提供实质的功能，并在与插件一起提交的自述文件中明确记录，最好是链接到服务使用条款。

不允许的服务和功能包括：

存在仅用于验证许可证或密钥的服务，而插件的所有功能方面都包含在本地的服务是不允许的。

通过将任意代码移出插件来创建服务，以防止服务可能错误地显示为提供补充的功能。

不是服务的店面。仅作为从外部系统购买的产品的前端的插件将不被接受。

- 插件可能不会在没有他们明确的，明确的，选择的同意的情况下“手机回家”或跟踪用户。

为了保护用户隐私，插件可能不需要用户通过要求在设置中注册服务或复选框的明确同意来联系外部服务器。这种方法被称为“选择加入”。关于如何收集和使用任何用户数据的文档应该包含在插件的自述文件中，最好用明确的隐私政策。

此限制包括以下内容：

未经授权收集用户数据。可能会要求用户提交信息，但不能自动记录用户的明确确认。

禁止有意误导用户提交信息作为使用插件本身的要求。

图像和脚本应尽可能在本地加载，作为插件的一部分。如果需要外部数据（如块列表），则必须向用户明确其包含的内容。

默认情况下，插件内使用的任何第三方广告机制都必须禁用所有跟踪功能。禁止不具有禁止用户跟踪功能的广告机制。

这个政策的唯一例外是软件即服务，如Twitter，Amazon CDN插件或Akismet。通过安装，激活，注册和配置利用这些服务的插件，可以同意这些系统。

- 插件可能不会通过第三方系统发送可执行代码。

允许从文件化服务中外部加载代码，但所有通信必须尽可能安全。不允许在插件中执行外部代码，例如：

从WordPress.org以外的其他服务器更新或安装插件，主题或附件

安装相同插件的高级版本

由于字体包含以外的原因，呼叫第三方CDN;所有非服务相关的JavaScript和CSS必须包含在本地

使用第三方服务来管理定期更新的数据列表，当服务的使用条款未明确允许时

使用iframe连接管理页面;应该使用API来最小化安全风险

- 插件及其开发人员不得做任何非法，不诚实或道德上的冒犯。

虽然这是主观和相当广泛的，但其目的是防止插件，开发人员和公司滥用最终用户以及其他插件开发人员的自由和权利。

这包括（但不限于）以下示例：

- 通过关键字填充，黑帽SEO或其他方式人工操纵搜索结果
- 提供更多的流量到使用该插件的网站
- 补偿，误导，压力，勒索或黑客用户进行评论
- 应用用户必须支付解锁包含的功能
- 创建帐户以生成假评论或支持票据（即，sockpuppeting）
- 伪造个人信息，故意伪装身份，避免对以前违规的制裁
- 考虑其他开发人员的插件，并将其作为原创作品
- 利用用户的服务器或资源作为僵尸网络的一部分
- 有意试图利用指南中的漏洞
- 违反WordCamp行为准则
- 插件不得在公共网站上嵌入外部链接或信用额度，而不会明确要求用户的许可。

所有“Powered By”或插件代码中包含的信用显示和链接必须是可选的，默认值不会显示在用户的前端网站

上。用户必须选择通过明确且可理解的选择来显示任何和所有信用和链接，不被隐瞒在使用条款或文档中。插件可能不需要信用或链接显示才能运行。允许服务按照合适的方式对其输出进行品牌标记，只要代码在服务中处理而不是插件处理。

- 插件不应该劫持管理仪表板。

用户喜欢和期望插件感觉像是WordPress的一部分。恒定的n ags。 ming the the the the the the the。 . . . . . . . . . .

升级提示，通知和警报的范围应受到限制，谨慎使用或仅在插件的设置页面上使用。任何网站广泛的通知或嵌入的仪表板小部件必须被忽略。错误消息和警报应包括有关如何解决情况的信息，并在完成时删除。

WordPress仪表板上的广告应受到限制。当成员被允许宣传自己的产品和服务时，用户历史上变得无效;理想情况下，用户很少访问这些屏幕。请记住：不允许通过这些广告追踪转介（见准则7），大多数第三方系统不允许后端广告（特别是Google）。滥用广告系统的指导方针将导致此类行为被报告。

开发人员是受欢迎的，并鼓励包括链接到自己的网站或社交网络，以及本地（在插件内），包括图像，以增强体验。

- WordPress.org上的面向公众的页面（readmes）可能不是垃圾邮件。

公开的网页（包括readmes和translation文件）可能不会被用于垃圾邮件。垃圾邮件行为包括（但不限于）购买的会员链接，标签到竞争对手插件，总共使用超过12个标签，黑色SEO和关键字填充。

允许直接要求的产品的链接，例如主题或插件使用所需的其他插件。类似的相关产品可能用于标签，但不能用于竞争对手。如果一个插件是一个WooCommerce扩展，它可以使用标签'Wuocommerce'。'如果插件是Akismet的替代品，它可能不会将该术语用作标签。重复使用标签或特定术语被认为是关键字填充，不允许。为人写的你的readmes，而不是机器人。

在所有情况下，联盟链接必须进行，并且必须直接链接到联盟服务，而不是重定向或隐藏的URL。

- 插件应该使用WordPress的默认库。

WordPress包含许多有用的库，如jQuery，Atom Lib，SimplePie，PHPMailer，PHPass等。为了安全和稳定的原因，插件可能不会在自己的代码中包含这些库，而是必须使用WordPress打包的那些库的版本。

有关WordPress中包含的所有JavaScript库的列表，请查看WordPress中包含并注册的默认脚本。

- 频繁提交插件应该是违反。

SVN存储库是一个版本库，而不是开发版本。所有提交将触发与插件相关联的zip文件的再生，因此只有准备好部署的代码（即稳定版本，beta或RC）才能被推送到SVN。强烈建议在每个提交中包含一个描述性和翔实的信息。频繁的“垃圾”提交消息，如“更新”或“清理”，使其他人难以追踪更改。承诺仅调整插件的小部分（包括自述文件）会对系统造成不必要的压力，并可被视为游戏最近更新的列表。

- 插件版本号必须在每次发布新版本时增加。

当代码版本在SVN中递增时，用户将只会被提醒。开发人员可以通过增加中继分支中readme.txt中的插件版本号或通过创建一个带有readme.txt的新标签分支来部署这些更新，该分支具有与分支目录名匹配的增量插件版本。如果开发人员采用标签目录方式分发其最新版本的插件，则可以不间断更新中继文件夹，而无需更改版本号。标签目录通常不会通过初始标记更新，除非需要更新readme.txt以支持发布新版本的WordPress。

有关标记的更多信息，请阅读我们的SVN标签说明以及readme.txt的工作原理。

- 在将插件请求提交到目录时，必须有一个完整的插件。

所有插件在批准之前都会进行检查，这就是为什么需要链接到zip的原因。名称不能“保留”供将来使用。在合理的时间内未使用的已批准插件的目录名称可能会提供给其他开发人员。

- 尊重商标和项目。

除非证明合法所有权/代表证明，否则禁止使用商标或其他项目作为插件的唯一或初始条款。例如，WordPress基金会将术语“WordPress”注册，并且在域名中使用“wordpress”是违规行为。这个政策扩展到插件插件。

另一个例子，只有Facebook的员工才能使用“Facebook”或其品牌在“Facebook Dancing Sloths”这样的背景下使用。非员工应该使用像“Dancing Sloths for Facebook”这样的格式，以避免潜在的误导用户认为该插件是由Facebook开发的。同样，如果您不代表“Chart.js”项目，将其用作插件的名称是不合适的。

推荐原创品牌，因为它不仅有助于避免混淆，而且对用户更加难忘。

- 我们保留随时更改插件指南的权利，恕不另行通知。

我们保留在任何时候根据需要更新这些指南的权利。

我们保留以任何理由任意禁用或删除任何插件的权利，即使由于这些准则未明确涵盖的原因。我们的目的是以尽可能公平的方式执行这些指南，以确保插件的质量和用户的安全。



# 规划您的插件

您已经写下了下一个Hello Dolly，并希望世界使用它。你该怎么办？

## ##测试一次并再次测试

任何运气，您的插件将被很多人在许多不同的情况和托管环境中使用。您需要确保您已经测试过您的插件，以确保它在任何情况下都可以工作，并且不会让您的用户失望。

## 选择一个好名字

插件名称应该反映出您和您的工作的独特性。当您选择姓名时，请确保您没有违反商标或踩踏他人的产品名称。如果你不在Facebook上工作，那么你应该不要命名你的插件“Facebook的跳舞松鼠”。一个更好的名字将是“为Facebook跳舞松鼠”。很难拿出一个好名字，所以花时间。您的插件网址提交后无法更改，但显示名称可以更改一千次。

## ##写出很好的文档

README.txt文件是开始的最佳位置，因为它是所有插件的标准参考点。你会想确保你包括：简要说明您的插件实际上是做什么的。如果它做的很多，可能会更好的两个插件。安装说明，特别是如果有特殊配置要完成。如果用户需要注册您的服务，请确保您链接到它。关于如何获得支持的指导，以及您所做的和不支持的内容。

## ##推出第一个版本到WordPress.org

WordPress.org plugins目录是潜在用户下载和安装插件的最简单方式。WordPress与插件目录的集成意味着用户可以通过几次点击来更新插件。

当您准备发布第一个版本时，您需要注册。审核过程成功完成后，您将获得一个代码的Subversion存储库。

WordPress.org网站提供了良好的文档，用于制作您的第一个Subversion提交和整个过程。

## 拥抱开源

开源是我们时代最强大的想法之一，因为它有助于跨界合作。通过鼓励捐款，您允许其他人尽可能多地爱上您的代码。有几个选项来打开你的代码：

Github可以让其他人参与您的项目变得简单。其他开发人员和用户可以轻松地提交错误修复或报告，功能要求或全新的贡献。如果您以前从未使用过Git，Github有一个很好的文档门户，甚至是一个交互式演示。

Bitbucket是具有类似功能的Github的替代品。

WordPress.org插件目录提供并要求您使用Subversion版本库。

## ##听你的用户

您经常会发现，您的用户将您的代码放置在比您想象的更多测试用例之外。这可以是非常有价值的反馈。

通过WordPress.org发布您的代码意味着您的插件自动有一个支持论坛。用它！您可以订阅通过电子邮件接收新的帖子，并及时回复您的用户。他们只想尽可能多地爱上你的插件。

Automattic的幸福工程师Andrew Spittle在提供支持方面有一些好的帖子：“避免容易”和“支持速度”。

Jetpack还有一篇文章，您可以指出编写大量错误报告。

## 定期推新版本

最好的插件是随着时间的推移不断重复的插件，推动了小的变化。等待太久才能更新，不要让你的辛勤工作过时。请记住，不断升级可能导致“更新疲劳”，用户将停止升级。保持太少的更新和太多更新之间的平衡很重要。

## ##冲洗并重复

像生活的其他部分一样，最好的事情是耐心和努力工作。



# 如何使用Subversion

我们将在这里介绍一些关于使用subversion的基础知识：启动，更改事物和标记版本。

本文档不是使用SVN的完整和强大的解释，而是更多的快速入门，以开始使用WordPress.org上的插件。有关更全面的文档，请参阅SVN Book。

有关其他信息，请参阅以下文档：

- [readme.txt如何工作](#)
- [插件资源（标题图像和图标）的工作原理](#)

## 术语

如果你刚刚颠覆，你需要知道几个字是什么意思，所以让我们来看一下颠覆行为来介绍一些术语。

您的所有文件将集中存储在我们服务器上的svn存储库中。从该存储库中，任何人都可以将您的插件文件的副本检出到本地机器上，但作为插件作者，只有您有权登记，这意味着您可以更改文件，添加新文件和删除本地机器上的文件，并将这些更改上传到中央服务器。这是检查存储库中的文件以及WordPress.org插件目录中显示信息的过程。

Subversion会跟踪所有这些更改，以便您可以随时返回并查看旧版本或更新版本，如果您需要的话。除了记住每个修订版本之外，您还可以通过subversion来标记存储库的某些修订版本，以供参考。标签对于标记不同版本的插件非常棒。

## SVN文件夹

SVN存储库有四个文件夹：

```
/assets/  
/branches/  
/tags/  
/trunk/
```

accsets用于屏幕截图，插件标题和插件图标。

所有开发工作属于trunk。

发行进入tags。

代码分支变化branchhs。

即使您在其他地方进行开发工作（如git仓库），我们建议您使用代码保持trunk文件夹的最新版本，方便SVN比较。

## 如何 ...

以下部分将介绍SVN的一些基础知识

开始你的全新插件

所有您需要做的是将您已经拥有的插件文件添加到新的SVN信息库。

为此，您需要首先在机器上创建一个本地目录以容纳SVN存储库的副本：

```
$ mkdir my-local-dir
```

接下来，查看预建的存储库

```
$ svn co https://plugins.svn.wordpress.org/your-plugin-name my-local-dir
> A my-local-dir/trunk
> A my-local-dir/branches
> A my-local-dir/tags
> Checked out revision 11325.
```

正如你所看到的，subversion已经将所有从中央SVN存储库的目录添加到“add”（“A”）到本地副本。现在，您可以使用复制/粘贴命令通过命令行或拖放将文件添加到本地存储库副本的目录中。无论你喜欢什么一旦您的文件位于trunk文件夹中，您必须让subversion知道您要将这些新文件添加到中央存储库中。

```
my-local-dir/$ svn add trunk/*
> A trunk/my-plugin.php
> A trunk/readme.txt
```

**警告：**请勿将主插件文件放在trunk的子文件夹中，如/trunk/my-plugin/my-plugin.php，因为这会破坏下载。您可以使用子文件夹包含的文件。

添加所有文件后，您将检查更改回到中央存储库。

```
my-local-dir/$ svn ci -m 'Adding first version of my plugin'
> Adding trunk/my-plugin.php
> Adding trunk/readme.txt
> Transmitting file data .
> Committed revision 11326.
```

需要为所有签入包含提交消息。

如果由于“禁止访问”而导致提交失败，并且您知道您有提交访问权限，请将您的用户名和密码添加到check-in命令中。

```
my-local-dir/$ svn ci -m 'Adding first version of my plugin' --username your_username --password your_password
```

## 编辑已在存储库中的文件

我们假设你已经有你的插件库填满了所需的文件。现在假设你需要编辑一个文件来改进插件。

首先进入您的本地存储库副本，并确保它是最新的

```
$ cd my-local-dir/
my-local-dir/$ svn up
> At revision 11326.
```

在上面的例子中，我们都是最新的。如果中央存储库中有更改，则它们将被下载并合并到本地副本中。

现在，您可以使用任何您喜欢的编辑器编辑需要更改的文件。

如果您没有使用SVN GUI工具（例如SubVersion或Coda），则在进行更改后，您仍然可以检查本地副本和中央存储库之间的区别。首先我们检查本地副本的状态：

```
my-local-dir/$ svn stat
> M trunk/my-plugin.php
```

这告诉我们，我们的本地trunk / my-plugin.php与我们从中央存储库下载的副本不同（“修改”的“M”）。

我们来看看这个文件究竟有什么变化，所以我们可以检查一下，确保事情看起来正确。

```
my-local-dir/$ svn diff
> * What comes out is essentially the result of a
  * standard `diff -u` between your local copy and the
  * original copy you downloaded.
```

如果一切都看起来不错，那么现在是时候检查中央存储库的这些更改。

```
my-local-dir/$ svn ci -m "fancy new feature: now you can foo *and* bar at the same time"
> Sending trunk/my-plugin.php
> Transmitting file data .
> Committed revision 11327.
All done!
```

如果由于“禁止访问”而导致提交失败，并且您知道您有提交访问权限，请将您的用户名和密码添加到check-in命令中。

```
my-local-dir/$ svn ci -m 'Adding first version of my plugin' --username your_username --password your_password
"Tag" a new version
```

每次你正式发布你的插件，你应该标记该版本的代码的副本。这可以让您的用户轻松获取最新版本（或更旧版本），让您更轻松地跟踪更改，并让WordPress.org插件目录知道您的插件版本应该告诉人们下载。

为了做到这一点，你需要记住在trunk/readme.txt中更新Stable Tag字段！

首先将代码复制到tags/目录中的子目录中。为了WordPress.org插件浏览器的目的，新的子目录应该总是看起来像一个版本号。2.0.1.3好 酷热标签不好

我们想使用svn cp而不是常规cp，以利用SVN的功能。

```
my-local-dir/$ svn cp trunk tags/2.0
> A tags/2.0
```

现在，一如以往，请检查更改。

```
my-local-dir/$ svn ci -m "tagging version 2.0"
> Adding          tags/2.0
> Adding          tags/2.0/my-plugin.php
> Adding          tags/2.0/readme.txt
> Committed revision 11328.
```

或者，您可以使用http URL复制，并自己保存带宽：

```
my-local-dir/$ svn cp https://plugins.svn.wordpress.org/your-plugin-name/trunk
https://plugins.svn.wordpress.org/your-plugin-name/tags/2.0
```

这样做将远程执行副本，而不是在本地复制一切并上传。如果您的插件较大，这可能是有益的。

恭喜！你已经更新了你的代码！

## 也可以看看

- [readme.txt如何工作](#)
- [插件资源（标题图像和图标）的工作原理](#)
- [SVN书](#)

# 插件开发者常见问题

---

## # 提交和批准

### ## 获得插件批准需要多长时间？

没有官方的平均值，因为没有两个插件是一样的。如果你的插件很小，所有的代码都是正确的，应该在七天之内批准。如果您的插件有任何代码问题，您需要花费更长时间来纠正问题。无论哪种方式，您将收到来自 [plugins@wordpress.org](mailto:plugins@wordpress.org) 的电子邮件地址，因此请将其添加到您的电子邮件白名单中，耐心等待我们的回复。

### ## 如果我的插件有问题，我需要解决多长时间？

从我们与您联系的时间起，我们允许七天的时间，当我们希望您完成更正。这是我们必须保持排队的唯一方法之一。如果您需要超过七天的时间，我们会拒绝您的请求，您可以在完成后重新提交。

你有什么特别的东西我应该避免吗？

我们寻找一些非常明显的东西，所有这些都列在我们的指南中。大多数人可以总结为“不要成为垃圾邮件发送者”，而是触及人们最多的工作：

作为服务时不包括一个readme.txt文件。

readme.txt文件是插件目录中插件显示信息的主要来源。它应该是现在的，应该解释如何使用插件和什么插件，即使你觉得很明显。

### 不用WP\_DEBUG测试插件

我们用WP\_DEBUG测试你的插件。你也应该在你提交之前。如果插件不起作用，我们推回。

包括打包的JavaScript库的自定义版本

WordPress附带了许多默认脚本（包括jQuery），我们要求您不要在您的插件中。用我们的调用外部文件

所有插件应尽可能独立。显然，如果您的插件依赖于您自己的服务器（如Twitter）上的服务，那么远程调用JS就可以了。但是，如果您尝试将图像和JS卸载到CDN而不提供服务，那么我们不允许这样做。

### “Powered By” 链接

网站前端方面的所有链接都必须是“选择加入”，这意味着您的插件可能不会默认链接到访问者看到的网站或其他任何网站。

### 打电话回家

这不仅仅是跟踪用户，而且还不包括将任何数据发送回服务器的文件（如iframe），除非这是插件的工作原理。即使这样，我们也可能会问。

### ## 你有不接受的插件吗？

一个插件应该是直接有益于用户的东西。它不应该要求用户编辑文件，它应该是开箱即用的。

### ## 我可以更改我的插件的名称吗？

是与否您可以更改显示名称，但是插件 - 您的插件网址的那部分 - 在批准插件后无法更改。所以，如果您提交“我的酷工具”作为名称，您的URL将是<https://wordpress.org/plugins/my-cool-tool/>，也将是每个人的网

站上的文件夹名称。

要更改显示名称，请编辑您的主插件文件，并将“插件名称：”的值更改为新名称。您可能还想在readme.txt中编辑标题

你有不允许的名字吗？

我们不允许在插件名称中使用“WordPress”，因为它是多余的，有些显而易见的是你是一个WordPress插件。

同样的原因我们也一般不允许使用“插件”。只有英文字母和阿拉伯数字才能在s一。。

所以。。。。。。我们不允许插件名称中的版本号。

我们还禁止在Web服务，工具或库之后完全命名插件，除非由正式表示Web服务，工具或库的人员提交。特别是如果这个插件涉及到这个东西。同样的，你不能用别人的商标用语开始插件。

有创意，想出你自己独特的名字。

## ##我已经有有了一个插件，但是我想重做它！我只是再次提交，对吧？

我们宁愿你实际上只是重写现有的插件。使其成为主要版本。我们不能重命名插件，所以一个新的不会覆盖任何现有的用户，评论，支持主题，评级，下载，收藏夹等。

## ##我犯了一个错误，提交了一个错误名称的插件。我该怎么解决？

电邮[plugins@wordpress.org](mailto:plugins@wordpress.org)并解释情况。我们可以关闭插件，让您重新提交，也可以更改显示名称。在我们批准任何事情之前，我们试图在名字中打错字，但是我们也犯错误。

## # 使用SVN存储库

## ##在什么目录我应该把我的文件？

将您的代码文件直接放在您的存储库的trunk /目录中。每当您发布新版本时，通过将当前中继版本复制到tags /目录的新子目录来标记该版本。

确保更新trunk / readme.txt以反映新的stable标签。

自述文件的图像（如屏幕截图，插件标题和插件图标）属于您的SVN结帐根目录中的assets /目录（您可能需要创建）。例如，这将与tags /和trunk /相同。

## ##不能把我的文件放在trunk /的子目录中？

你可以，但不要。 [WordPress.org Plugin Directory](#)创建的.zip文件将自动将所有文件包装在目录中，因此无需将文件放入子目录中。

如果您有很多文件的复杂插件，您当然可以将它们组织到子目录中，但是readme.txt文件和根插件文件应该直接进入trunk /。

## ##我应该如何命名我的标签（a.k.a.版本）？

您的Subversion标签应该看起来像版本号。具体来说，它们应该只包含数字和句点。2.8.4是一个很好的查找标签，我的内容是一个不好看的标签。不灵活？对。易于处理和消毒？你打赌！我们建议您使用语义版本控制来跟踪版本。

请注意，我们正在谈论Subversion标签，而不是readme.txt标签。那些显然可以是喜欢的任何单词。

## ##我的更改日志应该是什么？

更改日志是对插件进行的所有或所有显著更改的日志或记录，包括更改的记录，如错误修复，新功能等。如果需

要帮助格式化更改日志，我们建议使用保留更改日志作为所使用的格式由许多产品在那里。

##我应该在我的更新日志中保留多少个版本？

始终保持更改日志中的当前主要版本。例如，如果您当前的版本是3.9.1，那么您将需要更改日期和3.9。旧版本应该被删除并迁移到changelog.txt文件。这将允许用户访问它们，同时保持您的自述文件更短和更相关。最多可以在自述文件的更新日志中保留最新版本的插件和一个主要版本。您的changelog.txt将不会显示在WordPress.org插件目录中，但没关系。大多数用户只想知道新功能。

##我可以指定我的插件版本的WordPress.org插件目录应该使用什么？

是的，通过在中继线目录的readme.txt文件中指定“稳定标签”字段。

## WordPress.org插件目录在哪里可以获取其数据？

---

从您在插件文件和readme.txt文件中以及从Subversion存储库本身指定的信息。

阅读关于readme.txt如何工作的更多信息。

##插件如何在其插件描述页面上包含视频？

对于YouTube和Vimeo视频，只需在描述中将视频链接自行粘贴到一行。请注意，视频必须设置为允许嵌入才能使嵌入进程工作。

对于由WordPress.com VideoPress服务托管的视频，请使用wpvideo短代码。短信也可以用于YouTube和Vimeo，如果需要，就像在WordPress中一样。

##我对SVN信息库做了一些修改。WordPress.org插件目录需要多长时间来反映这些更改？

WordPress.org插件目录每隔几分钟更新一次。但是，根据更新队列的大小，可能需要更长时间才能显示更改。请至少6个小时，然后再与我们联系。

我可以在我的插件中包含SVN外部程序吗？

不，谢谢。您可以将svn externals添加到您的存储库，但不会添加到可下载的zip文件中。

##如何为我的插件页面制作酷炫的横幅之一？

您可以通过将正确命名的文件上传到assets文件夹中来创建自己的插件头。

阅读关于插件标题的更多信息。

##如何制作插件图标？

您可以通过将正确命名的文件上传到资产文件夹中来创建自己的插件图标。

阅读有关插件图标的更多信息。

##支持论坛

##如何获得论坛帖子的通知？

转到<https://wordpress.org/support/plugin/YOURPLUGIN>并向下滚动到帖子列表的底部。在那里，您将看到RSS链接的选项，以及注册电子邮件。

电子邮件/ rss的注册链接

点击订阅链接的电子邮件，或使用您最喜爱的阅读器中的RSS链接。

##如何获得所有我的插件的通知？



如果您正在跟踪WordPress论坛，请<https://wordpress.org/support/view/plugin-committer/YOURID>将列出您提交访问的任何插件的所有支持请求和评论。

不是一个comitter，只是有人列为作者？使用<https://wordpress.org/support/view/plugin-contributor/YOURID>

那些只是RSS。如果您需要电子邮件，请访问<https://profiles.wordpress.org/YOURID/profile/notifications/>并输入您要发送电子邮件的条款。

# 封闭插件

##如何关闭我的插件？

如果你要求你的插件被删除，你不会得到它，除非你可以证明你的情况。通过请求关闭插件是永久性的。

从提交访问和链接到您的插件的帐户电子邮件plugins@wordpress.org。如果您的电子邮件与提交访问插件的人不匹配，您将被要求从其他电子邮件发送。

##当插件关闭时会发生什么？

当插件关闭时，插件的前端URL重定向到主插件目录，并且不再生成拉链。没有人能够通过网站下载插件，也不能通过WordPress管理员安装它。根据目录的原则，SVN存储库将保持可访问，以允许其他人下载并分支代码（如果需要）。

##为什么我的插件关闭？

这是违反指南，您在.org系统上的行为或安全问题。在所有情况下，您应该已经收到plugins@wordpress.org的电子邮件，解释为什么。

##为什么别人的插件关闭？

我们不宣传为什么我们关闭插件到插件开发人员和WordPress核心开发人员。不要打扰问。这是为了安全起见。如果我们宣布为什么我们关闭一个插件，每个人都会知道漏洞。如果我们选择不说“这是为了安全”，那么每当我们说“我们不能告诉你”，世界就会知道这是为了安全。基本上，这样做会使事情变得更安全。

##可以让别人的插件关闭吗？

如果您在plugins@wordpress.org的插件中报告安全问题或违规指南，我们将审核该报告并采取适当的措施。大多数情况下，这涉及到关闭一个插件。

有人张贴了我的插件的副本！我该怎么办？

电子邮件plugins@wordpress.org与被盗插件的链接，以及链接到我们可以下载或附加zip的链接。我们将比较两个文件，以及我们拥有的所有编码历史，以确定插件是否确实是盗窃，或者仅仅是一个未认证的分支。请记住，如果您将插件授权为GPLv2或更高版本，则只要版权保持原样并被记录，则完全可以将您的工作分叉。

##如何发送安全报告？

电子邮件plugins@wordpress.org一个清晰简明的说明的问题。确保解释你如何验证这是一个漏洞（链接到插件列表上的网站，如secunia.com是完美的）。如果您提供的报告链接，请勿删除！我们将直接传递给插件的开发人员。

##我可以在一个插件中找到一个bug吗？

我们与任何错误奖励计划没有关系，所以我们不会向您提交报告等。[我们工作的唯一一个是](#)



[hackerone.com/automattic](https://hackerone.com/automattic)，这是与Automattic属性相关的错误。一切都是你自己的，不要求我们提交事情。

你是否帮助文件或提供CVE？

不，我们目前没有能力协助。

##我的插件关闭了，可以重新打开吗？

也许。如果由于安全原因而关闭，请解决问题，回复电子邮件，大部分时间我们将重新打开该插件。如果违反准则的行为接近，则取决于违规行为的严重程度和性质。例如，重复违规者不太可能重新打开插件，而不是第一次使用。

如果你要求插件关闭，你会被期望解释为什么心脏变化。插件打算在开发者要求时保持关闭，并且一个月后不再重新打开。

# 插件所有权

##如何让别人访问我的插件？

如果要用户添加为提交者，那么可以访问更新代码，您需要访问

<https://wordpress.org/plugins/YOURPLUGIN/admin>并将其用户名添加为提交者。

如果您希望他们作为作者出现，您将需要将其用户名添加到readme.txt文件。

##如何从我的插件中删除某人的访问权？

任何有提交访问权限的人都可以做到转到<https://wordpress.org/plugins/YOURPLUGIN/admin>并将其悬停在其ID上。将显示删除链接。点击它。

##如何接管一个废弃的插件？

我们允许用户采用目前不再开发的现有插件。

我们要求您尝试先与原始开发人员联系，以便他们可以添加您。在某些情况下，这是不可能的，你应该从修复插件开始。确保它符合编码标准，安全，并更新版权信息以包括自己。然后，您可以联系我们了解插件采用。

我们不保证将被给予任何人的插件。

##如果插件开发人员死了会怎么样？

当开发人员决定死亡时，他们将从他们自己的插件中删除，以防止不道德的访问和损害用户。如果他们是唯一的开发人员，则该插件是关闭的。所有的尝试都是找到他们的朋友和同事，为他们提供一个首先采用代码的机会，但是如果没有人可靠或愿意可以找到该插件被关闭。

# 开发工具

---

有各种工具可用于帮助插件开发。 其中一些运行在您的开发环境（ xdebug ， PHPCS 等 ）中，但也有一些优秀的工具可以在WordPress内部运行，以帮助您正确构建并诊断问题。 本章将介绍浏览器中的工具。

# Debug Bar 和附加组件

## Debug Bar

调试栏处于活动状态时，会向管理栏添加一个调试菜单，该菜单显示查询，缓存和其他有用的调试信息。

当WP\_DEBUG启用时，它还会跟踪“PHP警告和通知”，使其更容易找到。

启用SAVEQUERIES时，会跟踪并显示mysql查询。

访问调试栏

## Debug Bar Console

这个插件提供了一个可以运行任意PHP的大型文本区域。这对于测试变量的内容非常出色

访问调试栏控制台

## Debug Bar Shortcodes

调试条快捷键为调试栏添加一个新面板，显示当前请求的注册短码。

另外它会显示你：

短代码调用哪个函数/方法。

是否在当前的帖子/页面/帖子类型上使用短代码以及如何（仅在单数时）。

关于短码的任何附加信息，如描述，它所需要的参数，无论是否自动关闭。

查找使用短码的所有页面/帖子/ etc。

访问调试条短消息

## Debug Bar Constants

调试条常量向调试栏添加三个新面板，其中显示可用于当前请求的开发人员的定义常量：

WP常数

WP类常量

PHP常量

访问调试条常量

## Debug Bar Post Types

调试栏帖子类型在调试栏中添加一个新面板，其中显示有关您网站的注册的帖子类型的详细信息。

访问调试栏帖子类型

## Debug Bar Cron

Debug Bar Cron将有关WP预定事件的信息添加到Debug栏中的新面板。此插件是Debug Bar的扩展，因此依

依赖于Debug Bar正在安装它。

一旦安装，您将可以访问以下信息：

预定事件数

如果cron正在运行

下一个事件的时间

当前时间

自定义预定事件列表

核心预定活动清单

时间表列表

访问Debug Bar Cron

## Debug Bar Actions and Filters Addon

---

此插件在调试栏中添加了两个选项卡，以显示附加到当前请求的钩子（动作和过滤器）。“操作”选项卡显示与当前请求挂钩的操作。“过滤器”选项卡显示过滤器标签及其附带的功能，并分别优先。

访问调试栏操作和过滤器插件

## Debug Bar Transients

---

调试条瞬态将关于WordPress瞬变的信息添加到调试栏中的新面板。此插件是Debug Bar的扩展，因此依赖于Debug Bar正在安装它。

一旦安装，您将可以访问以下信息：

- 现有瞬变数
- 自定义瞬变列表
- 核心瞬变列表
- 自定义站点瞬变列表
- 核心站点瞬变列表
- 删除瞬态的选项

访问调试条瞬态

## Debug Bar List Script & Style Dependencies

---

列出加载的脚本和样式，以及它们的加载顺序以及依赖关系的存在。

访问调试条列表脚本和样式依赖关系

## Debug Bar Remote Requests

---

这将记录和配置通过HTTP API创建的远程请求。

此插件将向调试栏添加一个“远程请求”面板，它将显示：

- 请求方法（GET，POST等）
- 网址
- 每次请求的时间
- 所有请求的总时间
- 请求总数

或者，您可以在您的URL中添加？dbrr\_full = 1以获取其他信息，包括所有请求参数以及使用标题的响应的完整转储。

访问调试栏远程请求

# 辅助插件

---

## Query Monitor

查询监视器是用于使用WordPress开发的任何人的调试插件。 您可以查看关于数据库查询，钩子，条件，HTTP请求，重定向等的调试和性能信息。 它具有一些其他调试插件不可用的高级功能，包括自动AJAX调试，以及通过插件或主题缩小内容的能力。

## Visit Query Monitor

# REST API手册

## REST API 手册

WordPress REST API为WordPress数据类型提供API端点，允许开发人员通过发送和接收JSON（JavaScript对象符号）对象远程与站点进行交互。JSON是一种开放标准的数据格式，它是轻量级和可读的，看起来像是在JavaScript中的对象；由此得名。当您向API发送内容或发出请求时，将以JSON返回响应。这使开发人员能够从客户端JavaScript或外部应用程序创建，阅读和更新WordPress内容，甚至是用PHP语言编写的内容。

寻找WordPress中可用的其他API的列表？您可以在[这里](#)找到文档。

## 如何使用WordPress REST API

WordPress REST API使以前更容易使用WordPress新的和令人兴奋的方式，如在WordPress之上创建单页应用程序。您可以创建一个插件，为WordPress提供全新的管理体验，或创建全新的互动前端体验。

您甚至不必在PHP中编写应用程序：可以使HTTP请求和解释JSON的任何编程语言都可以通过REST API从Node.js到Java以及其他方面与WordPress进行交互。

WordPress REST API也可以作为核心中的admin-ajax API的强大替代品。通过使用REST API，您可以更轻松地构建要将数据导入和移出WordPress的方式。通过使用REST API可以大大简化AJAX调用，使您能够花更少的时间访问所需的数据，更多的时间创建更好的用户体验。

我们的想象力是WordPress REST API可以做到的唯一限制。底线是，如果您希望使用结构化，可扩展和简单的方式通过HTTP获取和传出WordPress中的数据，则可能需要使用REST API。对于所有的简单性，REST API首先可以感觉到相当复杂，我们将尝试将其分解成较小的组件，以便我们可以轻松地整理拼图。

### 关键概念

要开始使用WordPress REST API，我们将分解与API相关联的一些关键概念和术语：

- Routes/Endpoints
- Requests
- Responses
- Schema
- Controller Classes

这些概念中的每一个在使用和理解WordPress REST API方面起关键作用。让我们简单地将它们分解下来，以便我们能够更深入地探索每一个。

### Routes & Endpoints

在WordPress REST API的上下文中，路由是可以映射到不同HTTP方法的URI。单个HTTP方法映射到路由被称为“端点”。要澄清：如果我们向http://oursite.com/wp-json/发出GET请求，我们将获得一个JSON响应，显示我们可用的路由，并在每个路由中提供哪些端点。 / wp-json /是一个路由本身，当GET请求被创建时，它与显示所谓的WordPress REST API的索引的端点相匹配。我们将在以下部分中学习如何注册自己的路由和端点。

## Requests

---

WordPress REST API基础架构中的主要类之一是WP\_REST\_Request。此类用于存储和检索当前请求的信息;请求可以通过HTTP远程提交，但也可以通过WordPress从PHP内部进行。当您向注册的路由发出HTTP请求时，WP\_REST\_Request对象将自动生成。请求中指定的数据将确定您从API中返回的响应。使用请求类可以做很多整齐的事情。请求部分将进一步详细介绍。

## Responses

---

响应是从API获取的数据。 WP\_REST\_Response提供了一种与端点返回的响应数据交互的方法。响应可以返回所需的数据，也可以用于返回错误。

## 模式

---

每个端点都要求并提供稍微不同的数据结构，这些结构在API模式中定义。 模式结构API数据，并提供了API可以返回的所有属性以及可以接受的输入参数的综合列表。 模式还为API提供安全性好处，因为它使我们能够验证对API所做的请求。 模式部分进一步探讨了这个大题目。

### ##控制器类

您可以看到，WordPress REST API有很多移动部分，所有这些都需要一起工作。 控制器类将所有这些元素放在一起。 使用控制器类，您可以管理路由和端点的注册，处理请求，使用模式和生成API响应。



# 资源

## 参考

WordPress REST API围绕REST进行组织，旨在具有可预测的面向资源的URL，并使用HTTP响应代码来指示API错误。API使用内置的HTTP功能，如HTTP身份验证和HTTP动词，可以由现成的HTTP客户端来理解，并支持跨原始资源共享，以便您可以从客户端安全地与API进行交互Web应用程序。

REST API使用JSON作为请求和响应格式，包括错误响应。虽然REST API不完全符合HAL标准，但它确实实现了HAL的.\_links和.\_embedded属性，用于链接API资源，并且通过响应中的超链接可以完全发现。

REST API提供匿名访问的任何客户端的公共数据，以及仅在身份验证后可用的专用数据。通过身份验证，REST API支持大多数内容管理操作，允许您为站点构建替代仪表板，通过更灵敏的管理工具增强插件，或构建复杂的单页应用程序。

此API参考提供了有关通过API可用的特定端点的信息，其参数及其响应数据格式。

## REST API开发人员端点参考

Resource	Base Route
Posts	/wp/v2/posts
Post Revisions	/wp/v2/revisions
Categories	/wp/v2/categories
Tags	/wp/v2/tags
Pages	/wp/v2/pages
Comments	/wp/v2/comments
Taxonomies	/wp/v2/taxonomies
Media	/wp/v2/media
Users	/wp/v2/users
Post Types	/wp/v2/types
Post Statuses	/wp/v2/statuses
Settings	/wp/v2/settings

## 分布式API

与许多其他REST API不同，WordPress REST API可以在支持它的每个站点上单独分发和提供。这意味着没有单一的API根或基础来联系；相反，我们有一个发现过程，允许在没有事先联系的情况下与网站进行交互。API还会在索引端点或通过OPTIONS请求向任何端点公开自我文档，从而允许人机或机器发现端点功能。

用于测试目的的API的演示安装可在<https://demo.wp-api.org/wp-json/>获取；该站点支持自动发现，并提供只

资源

读演示数据。

# 文章

---

## Schema

The schema defines all the fields that exist for a post object.

date

string, datetime (ISO8601) The date the object was published, in the site' s timezone.

Context: view, edit, embed

date\_gmt

string, datetime (ISO8601) The date the object was published, as GMT.

Context: view, edit

guid

object The globally unique identifier for the object.

Read only

Context: view, edit

id

integer Unique identifier for the object.

Read only

Context: view, edit, embed

link

string, uri URL to the object.

Read only

Context: view, edit, embed

modified

string, datetime (ISO8601) The date the object was last modified, in the site' s timezone.

Read only

Context: view, edit

modified\_gmt

string, datetime (ISO8601) The date the object was last modified, as GMT.

Read only

Context: view, edit

slug

string An alphanumeric identifier for the object unique to its type.

Context: view, edit, embed

status

## 文章

string A named status for the object.

Context: edit

One of: publish, future, draft, pending, private  
type

string Type of Post for the object.

Read only

Context: view, edit, embed

password

string A password to protect access to the content and excerpt.

Context: edit

title

object The title for the object.

Context: view, edit, embed

content

object The content for the object.

Context: view, edit

author

integer The ID for the author of the object.

Context: view, edit, embed

excerpt

object The excerpt for the object.

Context: view, edit, embed

featured\_media

integer The ID of the featured media for the object.

Context: view, edit

comment\_status

string Whether or not comments are open on the object.

Context: view, edit

One of: open, closed

ping\_status

string Whether or not the object can be pinged.

Context: view, edit

One of: open, closed

format

string The format for the object.

文章

Context: view, edit

One of: standard

meta

object Meta fields.

Context: view, edit

sticky

boolean Whether or not the object should be treated as sticky.

Context: view, edit

template

string The theme file to use to display the object.

Context: view, edit

One of:

categories

array The terms assigned to the object in the category taxonomy.

Context: view, edit

tags

array The terms assigned to the object in the post\_tag taxonomy.

Context: view, edit

liveblog\_likes

integer The number of Liveblog Likes the post has.

Context: view, edit, embed

## Example Request

---

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json/wp/v2/posts
```

Top ↑

List Posts #List Posts

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page Current page of the collection.

Default: 1

per\_page Maximum number of items to be returned in result set.

Default: 10

search Limit results to those matching a string.

after Limit response to posts published after a given ISO8601 compliant date.

author Limit result set to posts assigned to specific authors.

Default:

author\_exclude Ensure result set excludes posts assigned to specific authors.

Default:

before Limit response to posts published before a given ISO8601 compliant date.

exclude Ensure result set excludes specific IDs.

Default:

include Limit result set to specific IDs.

Default:

offset Offset the result set by a specific number of items.

order Order sort attribute ascending or descending.

Default: desc

One of: asc, desc

orderby Sort collection by object attribute.

Default: date

One of: date, relevance, id, include, title, slug

slug Limit result set to posts with one or more specific slugs.

status Limit result set to posts assigned one or more statuses.

Default: publish

categories Limit result set to all items that have the specified term assigned in the categories taxonomy.

Default:

categories\_exclude Limit result set to all items except those that have the specified term assigned in the categories taxonomy.

Default:

tags Limit result set to all items that have the specified term assigned in the tags taxonomy.

Default:

tags\_exclude Limit result set to all items except those that have the specified term assigned in the tags taxonomy.

Default:

sticky Limit result set to items that are sticky.

## Definition

---

GET /wp/v2/posts

## Example Request

---

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/posts>

# Retrieve a Post

## Arguments

---

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

password The password for the post if it is password protected.

Definition #Definition

GET /wp/v2/posts/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/posts/>

Create a Post #Create a Post

Arguments #Arguments

date The date the object was published, in the site's timezone.

date\_gmt The date the object was published, as GMT.

slug An alphanumeric identifier for the object unique to its type.

status A named status for the object.

One of: publish, future, draft, pending, private

password A password to protect access to the content and excerpt.

title The title for the object.

content The content for the object.

author The ID for the author of the object.

excerpt The excerpt for the object.

featured\_media The ID of the featured media for the object.

comment\_status Whether or not comments are open on the object.

One of: open, closed

ping\_status Whether or not the object can be pinged.

One of: open, closed

format The format for the object.

One of: standard

meta Meta fields.

sticky Whether or not the object should be treated as sticky.

template The theme file to use to display the object.

One of:

categories The terms assigned to the object in the category taxonomy.

tags The terms assigned to the object in the post\_tag taxonomy.

liveblog\_likes The number of Liveblog Likes the post has.

Definition #Definition

POST /wp/v2/posts

Update a Post #Update a Post

Arguments #Arguments

date The date the object was published, in the site' s timezone.

date\_gmt The date the object was published, as GMT.

slug An alphanumeric identifier for the object unique to its type.

status A named status for the object.

One of: publish, future, draft, pending, private

password A password to protect access to the content and excerpt.

title The title for the object.

content The content for the object.

author The ID for the author of the object.

excerpt The excerpt for the object.

featured\_media The ID of the featured media for the object.

comment\_status Whether or not comments are open on the object.

One of: open, closed

ping\_status Whether or not the object can be pinged.

One of: open, closed

format The format for the object.

One of: standard

meta Meta fields.

sticky Whether or not the object should be treated as sticky.

template The theme file to use to display the object.

One of:

categories The terms assigned to the object in the category taxonomy.

tags The terms assigned to the object in the post\_tag taxonomy.

liveblog\_likes The number of Liveblog Likes the post has.

## Definition

---



```
POST /wp/v2/posts/<id>
```

## ##示例请求

```
$ curl -X POST http://demo.wp-api.org/wp-json -d '{"title":"My New Title"}'
```

## #Delete a Post

## Arguments

---

force Whether to bypass trash and force deletion.

## Definition

---

DELETE /wp/v2/posts/

## Example Request

---

```
$ curl -X DELETE http://demo.wp-api.org/wp-json/wp/v2/posts/<id>
```

# 文章修订

---

Schema #Schema

The schema defines all the fields that exist for a posts-revision object.

author

integer

The id for the author of the object.

Context: view

date

string,

datetime (ISO8601) The date the object was published.

Context: view

date\_gmt

string,

datetime (ISO8601) The date the object was published, as GMT.

Context: view

guid

string

GUID for the object, as it exists in the database.

Context: view

id

integer

Unique identifier for the object.

Context: view

modified

string,

datetime (ISO8601) The date the object was last modified.

Context: view

modified\_gmt

string,

datetime (ISO8601) The date the object was last modified, as GMT.

Context: view

parent

integer

The id for the parent of the object.

Context: view

slug

string

An alphanumeric identifier for the object unique to its type.

Context: view

title

string

Title for the object, as it exists in the database.

Context: view

content

string

Content for the object, as it exists in the database.

Context: view

excerpt

string

Excerpt for the object, as it exists in the database.

Context: view

Example Request #Example Request

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json
```

Top ↑

List Post Revisions #List Post Revisions

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view

Top ↑

Definition #Definition

GET /wp/v2/posts/<parent\_id>/revisions

Example Request #Example Request

```
$ curl http://demo.wp-api.org/wp-json/wp/v2/posts/<parent\_id>/revisions
```

Top ↑

Retrieve a Post Revision #Retrieve a Post Revision

Top ↑

## Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view

Top ↑

## Definition #Definition

GET /wp/v2/posts/<parent\_id>/revisions/

## Example Request #Example Request

\$ curl [http://demo.wp-api.org/wp-json/wp/v2/posts/<parent\\_id>/revisions/](http://demo.wp-api.org/wp-json/wp/v2/posts/<parent_id>/revisions/)

Top ↑

## Delete a Post Revision #Delete a Post Revision

There are no arguments for this endpoint.

Top ↑

## Definition #Definition

DELETE /wp/v2/posts/<parent\_id>/revisions/

## Example Request #Example Request

\$ curl -X DELETE [http://demo.wp-api.org/wp-json/wp/v2/posts/<parent\\_id>/revisions/](http://demo.wp-api.org/wp-json/wp/v2/posts/<parent_id>/revisions/)

# 文章类型

## Schema

The schema defines all the fields that exist for a type object.

capabilities

array

All capabilities used by the resource.

Read only

Context: edit

description

string

A human-readable description of the resource.

Read only

Context: view, edit

hierarchical

boolean

Whether or not the resource should have children.

Read only

Context: view, edit

labels

object

Human-readable labels for the resource for various contexts.

Read only

Context: edit

name

string

The title for the resource.

Read only

Context: view, edit, embed

slug

string

An alphanumeric identifier for the resource.

Read only

Context: view, edit, embed

Example Request #Example Request

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json/wp/v2/types
```

Top ↑

List Types #List Types

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

## Definition

```
GET /wp/v2/types
```

## Example Request

```
$ curl http://demo.wp-api.org/wp-json/wp/v2/types
```

# Retrieve a Type

## Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

## Definition

```
GET /wp/v2/types/<type>
```

## Example Request

```
$ curl http://demo.wp-api.org/wp-json/wp/v2/types/<type>
```

# 文章状态

---

Schema #Schema

The schema defines all the fields that exist for a status object.

name

string

The title for the resource.

Read only

Context: embed, view, edit

private

boolean

Whether posts with this resource should be private.

Read only

Context: edit

protected

boolean

Whether posts with this resource should be protected.

Read only

Context: edit

public

boolean

Whether posts of this resource should be shown in the front end of the site.

Read only

Context: view, edit

queryable

boolean

Whether posts with this resource should be publicly-queryable.

Read only

Context: view, edit

show\_in\_list

boolean

Whether to include posts in the edit listing for their post type.

Read only

Context: edit

slug

string

An alphanumeric identifier for the resource.

Read only

Context: embed, view, edit

Example Request #Example Request

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json/wp/v2/statuses
```

Top ↑

List Status #List Status

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

Definition #Definition

GET /wp/v2/statuses

Example Request #Example Request

```
$ curl http://demo.wp-api.org/wp-json/wp/v2/statuses
```

Top ↑

Retrieve a Status #Retrieve a Status

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

Definition #Definition

GET /wp/v2/statuses/

Example Request #Example Request

```
$ curl http://demo.wp-api.org/wp-json/wp/v2/statuses/
```



# 类别

---

Schema #Schema

The schema defines all the fields that exist for a category object.

id

integer Unique identifier for the term.

Read only

Context: view, embed, edit

count

integer Number of published posts for the term.

Read only

Context: view, edit

description

string HTML description of the term.

Context: view, edit

link

string, uri URL of the term.

Read only

Context: view, embed, edit

name

string HTML title for the term.

Context: view, embed, edit

slug

string An alphanumeric identifier for the term unique to its type.

Context: view, embed, edit

taxonomy

string Type attribution for the term.

Read only

Context: view, embed, edit

One of: category, post\_tag, nav\_menu, link\_category, post\_format

parent

integer The parent term ID.

Context: view, edit

meta

类别

object Meta fields.

Context: view, edit

Example Request #Example Request

\$ curl -X OPTIONS -i <http://demo.wp-api.org/wp-json/wp/v2/categories>

Top ↑

List Categorys #List Categorys

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page Current page of the collection.

Default: 1

per\_page Maximum number of items to be returned in result set.

Default: 10

search Limit results to those matching a string.

exclude Ensure result set excludes specific IDs.

Default:

include Limit result set to specific IDs.

Default:

order Order sort attribute ascending or descending.

Default: asc

One of: asc, desc

orderby Sort collection by term attribute.

Default: name

One of: id, include, name, slug, term\_group, description, count

hide\_empty Whether to hide terms not assigned to any posts.

parent Limit result set to terms assigned to a specific parent.

post Limit result set to terms assigned to a specific post.

slug Limit result set to terms with a specific slug.

Top ↑

Definition #Definition

GET /wp/v2/categories

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/categories>

类别

Top ↑

Retrieve a Category #Retrieve a Category

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

Definition #Definition

GET /wp/v2/categories/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/categories/>

Top ↑

Create a Category #Create a Category

Top ↑

Arguments #Arguments

description HTML description of the term.

name HTML title for the term.

Required: true

slug An alphanumeric identifier for the term unique to its type.

parent The parent term ID.

meta Meta fields.

Top ↑

Definition #Definition

POST /wp/v2/categories

Top ↑

Update a Category #Update a Category

Top ↑

Arguments #Arguments

description HTML description of the term.

name HTML title for the term.

slug An alphanumeric identifier for the term unique to its type.

parent The parent term ID.

meta Meta fields.

Top ↑

类别

Definition #Definition

POST /wp/v2/categories/

Example Request #Example Request

Top ↑

Delete a Category #Delete a Category

Top ↑

Arguments #Arguments

force Required to be true, as terms do not support trashing.

Top ↑

Definition #Definition

DELETE /wp/v2/categories/

Example Request #Example Request

\$ curl -X DELETE <http://demo.wp-api.org/wp-json/wp/v2/categories/>

# 标签

---

Schema #Schema

The schema defines all the fields that exist for a tag object.

id

integer Unique identifier for the term.

Read only

Context: view, embed, edit

count

integer Number of published posts for the term.

Read only

Context: view, edit

description

string HTML description of the term.

Context: view, edit

link

string, uri URL of the term.

Read only

Context: view, embed, edit

name

string HTML title for the term.

Context: view, embed, edit

slug

string An alphanumeric identifier for the term unique to its type.

Context: view, embed, edit

taxonomy

string Type attribution for the term.

Read only

Context: view, embed, edit

One of: category, post\_tag, nav\_menu, link\_category, post\_format

meta

object Meta fields.

Context: view, edit

Example Request #Example Request

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json/wp/v2/tags
```

Top ↑

List Tags #List Tags

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page Current page of the collection.

Default: 1

per\_page Maximum number of items to be returned in result set.

Default: 10

search Limit results to those matching a string.

exclude Ensure result set excludes specific IDs.

Default:

include Limit result set to specific IDs.

Default:

offset Offset the result set by a specific number of items.

order Order sort attribute ascending or descending.

Default: asc

One of: asc, desc

orderby Sort collection by term attribute.

Default: name

One of: id, include, name, slug, term\_group, description, count

hide\_empty Whether to hide terms not assigned to any posts.

post Limit result set to terms assigned to a specific post.

slug Limit result set to terms with a specific slug.

Top ↑

Definition #Definition

GET /wp/v2/tags

Example Request #Example Request

```
$ curl http://demo.wp-api.org/wp-json/wp/v2/tags
```

Top ↑

Retrieve a Tag #Retrieve a Tag

Top ↑

## Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

## Definition #Definition

GET /wp/v2/tags/

## Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/tags/>

Top ↑

## Create a Tag #Create a Tag

Top ↑

## Arguments #Arguments

description HTML description of the term.

name HTML title for the term.

Required: true

slug An alphanumeric identifier for the term unique to its type.

meta Meta fields.

Top ↑

## Definition #Definition

POST /wp/v2/tags

Top ↑

## Update a Tag #Update a Tag

Top ↑

## Arguments #Arguments

description HTML description of the term.

name HTML title for the term.

slug An alphanumeric identifier for the term unique to its type.

meta Meta fields.

Top ↑

## Definition #Definition

POST /wp/v2/tags/

## Example Request #Example Request

Top ↑

## Delete a Tag #Delete a Tag

标签

Top ↑

Arguments #Arguments

force Required to be true, as terms do not support trashing.

Top ↑

Definition #Definition

DELETE /wp/v2/tags/

Example Request #Example Request

\$ curl -X DELETE <http://demo.wp-api.org/wp-json/wp/v2/tags/>



# 页面

---

## Schema #Schema

The schema defines all the fields that exist for a page object.

date

string,

datetime (ISO8601)

The date the object was published, in the site' s timezone.

Context: view, edit, embed

date\_gmt

string,

datetime (ISO8601)

The date the object was published, as GMT.

Context: view, edit

guid

object

The globally unique identifier for the object.

Read only

Context: view, edit

id

integer

Unique identifier for the object.

Read only

Context: view, edit, embed

link

string,

uri

URL to the object.

Read only

Context: view, edit, embed

modified

string,

datetime (ISO8601)

The date the object was last modified, in the site' s timezone.

页面

Read only

Context: view, edit

modified\_gmt

string,

datetime (ISO8601)

The date the object was last modified, as GMT.

Read only

Context: view, edit

slug

string

An alphanumeric identifier for the object unique to its type.

Context: view, edit, embed

status

string

A named status for the object.

Context: edit

One of: publish, future, draft, pending, private

type

string

Type of Post for the object.

Read only

Context: view, edit, embed

parent

integer

The id for the parent of the object.

Context: view, edit

title

object

The title for the object.

Context: view, edit, embed

content

object

The content for the object.

Context: view, edit

author

页面

integer

The id for the author of the object.

Context: view, edit, embed

excerpt

object

The excerpt for the object.

Context: view, edit, embed

featured\_media

integer

The id of the featured media for the object.

Context: view, edit

comment\_status

string

Whether or not comments are open on the object.

Context: view, edit

One of: open, closed

ping\_status

string

Whether or not the object can be pinged.

Context: view, edit

One of: open, closed

menu\_order

integer

The order of the object in relation to other object of its type.

Context: view, edit

meta

object

Meta fields.

Context: view, edit

template

string

The theme file to use to display the object.

Context: view, edit

One of:

Example Request #Example Request

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json/wp/v2/pages
```

Top ↑

List Pages #List Pages

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page

Current page of the collection.

Default: 1

per\_page

Maximum number of items to be returned in result set.

Default: 10

search

Limit results to those matching a string.

after

Limit response to resources published after a given ISO8601 compliant date.

author

Limit result set to posts assigned to specific authors.

Default:

author\_exclude

Ensure result set excludes posts assigned to specific authors.

Default:

before

Limit response to resources published before a given ISO8601 compliant date.

exclude

Ensure result set excludes specific ids.

Default:

include

Limit result set to specific ids.

Default:

menu\_order

Limit result set to resources with a specific menu\_order value.

页面

offset

Offset the result set by a specific number of items.

order

Order sort attribute ascending or descending.

Default: desc

One of: asc, desc

orderby

Sort collection by object attribute.

Default: date

One of: date, relevance, id, include, title, slug, menu\_order

parent

Limit result set to those of particular parent ids.

Default:

parent\_exclude

Limit result set to all items except those of a particular parent id.

Default:

slug

Limit result set to posts with a specific slug.

status

Limit result set to posts assigned a specific status; can be comma-delimited list of status types.

Default: publish

One of: publish, future, draft, pending, private, trash, auto-draft, inherit, any

filter

Use WP Query arguments to modify the response; private query vars require appropriate authorization.

Top ↑

Definition #Definition

GET /wp/v2/pages

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/pages>

Top ↑

Retrieve a Page #Retrieve a Page

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

页面

Default: view

One of: view, embed, edit

password

The password for the post if it is password protected.

Top ↑

Definition #Definition

GET /wp/v2/pages/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/pages/>

Top ↑

Create a Page #Create a Page

Top ↑

Arguments #Arguments

date

The date the object was published, in the site' s timezone.

date\_gmt

The date the object was published, as GMT.

slug

An alphanumeric identifier for the object unique to its type.

status

A named status for the object.

One of: publish, future, draft, pending, private

parent

The id for the parent of the object.

title

The title for the object.

content

The content for the object.

author

The id for the author of the object.

excerpt

The excerpt for the object.

featured\_media

The id of the featured media for the object.

comment\_status

Whether or not comments are open on the object.

One of: open, closed

ping\_status

Whether or not the object can be pinged.

One of: open, closed

menu\_order

The order of the object in relation to other object of its type.

meta

Meta fields.

template

The theme file to use to display the object.

One of:

Top ↑

Definition #Definition

POST /wp/v2/pages

Top ↑

Update a Page #Update a Page

Top ↑

Arguments #Arguments

date

The date the object was published, in the site' s timezone.

date\_gmt

The date the object was published, as GMT.

slug

An alphanumeric identifier for the object unique to its type.

status

A named status for the object.

One of: publish, future, draft, pending, private

parent

The id for the parent of the object.

title

The title for the object.

content

The content for the object.

author

The id for the author of the object.

excerpt

The excerpt for the object.

featured\_media

The id of the featured media for the object.

comment\_status

Whether or not comments are open on the object.

One of: open, closed

ping\_status

Whether or not the object can be pinged.

One of: open, closed

menu\_order

The order of the object in relation to other object of its type.

meta

Meta fields.

template

The theme file to use to display the object.

One of:

Top ↑

Definition #Definition

POST /wp/v2/pages/

Example Request #Example Request

Top ↑

Delete a Page #Delete a Page

Top ↑

Arguments #Arguments

force

Whether to bypass trash and force deletion.

Top ↑

Definition #Definition

DELETE /wp/v2/pages/

Example Request #Example Request

\$ curl -X DELETE <http://demo.wp-api.org/wp-json/wp/v2/pages/>



# 评论

---

Schema #Schema

The schema defines all the fields that exist for a comment object.

id

integer

Unique identifier for the object.

Read only

Context: view, edit, embed

author

integer

The id of the user object, if author was a user.

Context: view, edit, embed

author\_email

string,

email

Email address for the object author.

Context: edit

author\_ip

string,

ipv4

IP address for the object author.

Context: edit

author\_name

string

Display name for the object author.

Context: view, edit, embed

author\_url

string,

uri

URL for the object author.

Context: view, edit, embed

author\_user\_agent

string

评论

User agent for the object author.

Read only

Context: edit

content

object

The content for the object.

Context: view, edit, embed

date

string,

datetime (ISO8601)

The date the object was published.

Context: view, edit, embed

date\_gmt

string,

datetime (ISO8601)

The date the object was published as GMT.

Context: view, edit

karma

integer

Karma for the object.

Context: edit

link

string,

uri

URL to the object.

Read only

Context: view, edit, embed

parent

integer

The id for the parent of the object.

Context: view, edit, embed

post

integer

The id of the associated post object.

Context: view, edit

评论

status

string

State of the object.

Context: view, edit

type

string

Type of Comment for the object.

Context: view, edit, embed

author\_avatar\_urls

object

Avatar URLs for the object author.

Read only

Context: view, edit, embed

meta

object

Meta fields.

Context: view, edit

Example Request #Example Request

\$ curl -X OPTIONS -i <http://demo.wp-api.org/wp-json/wp/v2/comments>

Top ↑

List Comments #List Comments

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page

Current page of the collection.

Default: 1

per\_page

Maximum number of items to be returned in result set.

Default: 10

search

Limit results to those matching a string.

评论

after

Limit response to resources published after a given ISO8601 compliant date.

author

Limit result set to comments assigned to specific user ids. Requires authorization.

author\_exclude

Ensure result set excludes comments assigned to specific user ids. Requires authorization.

author\_email

Limit result set to that from a specific author email. Requires authorization.

before

Limit response to resources published before a given ISO8601 compliant date.

exclude

Ensure result set excludes specific ids.

Default:

include

Limit result set to specific ids.

Default:

karma

Limit result set to that of a particular comment karma. Requires authorization.

offset

Offset the result set by a specific number of comments.

order

Order sort attribute ascending or descending.

Default: desc

One of: asc, desc

orderby

Sort collection by object attribute.

Default: date\_gmt

One of: date, date\_gmt, id, include, post, parent, type

parent

Limit result set to resources of specific parent ids.

Default:

parent\_exclude

Ensure result set excludes specific parent ids.

Default:

post

Limit result set to resources assigned to specific post ids.

Default:

status

Limit result set to comments assigned a specific status. Requires authorization.

Default: approve

type

Limit result set to comments assigned a specific type. Requires authorization.

Default: comment

Top ↑

Definition #Definition

GET /wp/v2/comments

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/comments>

Top ↑

Retrieve a Comment #Retrieve a Comment

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

Definition #Definition

GET /wp/v2/comments/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/comments/>

Top ↑

Create a Comment #Create a Comment

Top ↑

Arguments #Arguments

author

The id of the user object, if author was a user.

author\_email

Email address for the object author.

author\_ip

## 评论

IP address for the object author.

Default: 127.0.0.1

author\_name

Display name for the object author.

author\_url

URL for the object author.

content

The content for the object.

date

The date the object was published.

date\_gmt

The date the object was published as GMT.

karma

Karma for the object.

parent

The id for the parent of the object.

Default: 0

post

The id of the associated post object.

Default: 0

status

State of the object.

type

Type of Comment for the object.

Default: comment

meta

Meta fields.

Top ↑

Definition #Definition

POST /wp/v2/comments

Top ↑

Update a Comment #Update a Comment

Top ↑

Arguments #Arguments

author

## 评论

The id of the user object, if author was a user.

author\_email

Email address for the object author.

author\_ip

IP address for the object author.

author\_name

Display name for the object author.

author\_url

URL for the object author.

content

The content for the object.

date

The date the object was published.

date\_gmt

The date the object was published as GMT.

karma

Karma for the object.

parent

The id for the parent of the object.

post

The id of the associated post object.

status

State of the object.

type

Type of Comment for the object.

meta

Meta fields.

Top ↑

Definition #Definition

POST /wp/v2/comments/

Example Request #Example Request

Top ↑

Delete a Comment #Delete a Comment

Top ↑

Arguments #Arguments

评论

force

Whether to bypass trash and force deletion.

Top ↑

Definition #Definition

DELETE /wp/v2/comments/

Example Request #Example Request

\$ curl -X DELETE <http://demo.wp-api.org/wp-json/wp/v2/comments/>



# 分类

---

Schema #Schema

The schema defines all the fields that exist for a taxonomy object.

capabilities

object All capabilities used by the taxonomy.

Read only

Context: edit

description

string A human-readable description of the taxonomy.

Read only

Context: view, edit

hierarchical

boolean Whether or not the taxonomy should have children.

Read only

Context: view, edit

labels

object Human-readable labels for the taxonomy for various contexts.

Read only

Context: edit

name

string The title for the taxonomy.

Read only

Context: view, edit, embed

slug

string An alphanumeric identifier for the taxonomy.

Read only

Context: view, edit, embed

show\_cloud

boolean Whether or not the term cloud should be displayed.

Read only

Context: edit

types

array Types associated with the taxonomy.

分类

Read only

Context: view, edit

rest\_base

string REST base route for the taxonomy.

Read only

Context: view, edit, embed

Example Request #Example Request

\$ curl -X OPTIONS -i <http://demo.wp-api.org/wp-json/wp/v2/taxonomies>

Top ↑

List Taxonomys #List Taxonomys

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

type Limit results to taxonomies associated with a specific post type.

Top ↑

Definition #Definition

GET /wp/v2/taxonomies

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/taxonomies>

Top ↑

Retrieve a Taxonomy #Retrieve a Taxonomy

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

Definition #Definition

GET /wp/v2/taxonomies/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/taxonomies/>

# 媒体

---

## Schema #Schema

The schema defines all the fields that exist for a attachment object.

date

string,

datetime (ISO8601)

The date the object was published, in the site' s timezone.

Context: view, edit, embed

date\_gmt

string,

datetime (ISO8601)

The date the object was published, as GMT.

Context: view, edit

guid

object

The globally unique identifier for the object.

Read only

Context: view, edit

id

integer

Unique identifier for the object.

Read only

Context: view, edit, embed

link

string,

uri

URL to the object.

Read only

Context: view, edit, embed

modified

string,

datetime (ISO8601)

The date the object was last modified, in the site' s timezone.

媒体

Read only

Context: view, edit

modified\_gmt

string,

datetime (ISO8601)

The date the object was last modified, as GMT.

Read only

Context: view, edit

slug

string

An alphanumeric identifier for the object unique to its type.

Context: view, edit, embed

status

string

A named status for the object.

Context: edit

One of: publish, future, draft, pending, private

type

string

Type of Post for the object.

Read only

Context: view, edit, embed

title

object

The title for the object.

Context: view, edit, embed

author

integer

The id for the author of the object.

Context: view, edit, embed

comment\_status

string

Whether or not comments are open on the object.

Context: view, edit

One of: open, closed

媒体

ping\_status

string

Whether or not the object can be pinged.

Context: view, edit

One of: open, closed

meta

object

Meta fields.

Context: view, edit

alt\_text

string

Alternative text to display when resource is not displayed.

Context: view, edit, embed

caption

string

The caption for the resource.

Context: view, edit

description

string

The description for the resource.

Context: view, edit

media\_type

string

Type of resource.

Read only

Context: view, edit, embed

One of: image, file

mime\_type

string

MIME type of resource.

Read only

Context: view, edit, embed

media\_details

object

Details about the resource file, specific to its type.

媒体

Read only

Context: view, edit, embed

post

integer

The id for the associated post of the resource.

Context: view, edit

source\_url

string,

uri

URL to the original resource file.

Read only

Context: view, edit, embed

Example Request #Example Request

\$ curl -X OPTIONS -i <http://demo.wp-api.org/wp-json/wp/v2/media>

Top ↑

List Media #List Media

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page

Current page of the collection.

Default: 1

per\_page

Maximum number of items to be returned in result set.

Default: 10

search

Limit results to those matching a string.

after

Limit response to resources published after a given ISO8601 compliant date.

author

Limit result set to posts assigned to specific authors.

Default:

author\_exclude

Ensure result set excludes posts assigned to specific authors.

Default:

before

Limit response to resources published before a given ISO8601 compliant date.

exclude

Ensure result set excludes specific ids.

Default:

include

Limit result set to specific ids.

Default:

offset

Offset the result set by a specific number of items.

order

Order sort attribute ascending or descending.

Default: desc

One of: asc, desc

orderby

Sort collection by object attribute.

Default: date

One of: date, relevance, id, include, title, slug

parent

Limit result set to those of particular parent ids.

Default:

parent\_exclude

Limit result set to all items except those of a particular parent id.

Default:

slug

Limit result set to posts with a specific slug.

status

Limit result set to posts assigned a specific status; can be comma-delimited list of status types.

Default: inherit

One of: inherit, private, trash

filter

Use WP Query arguments to modify the response; private query vars require appropriate authorization.

media\_type

Limit result set to attachments of a particular media type.

One of: image, video, audio, application

mime\_type

Limit result set to attachments of a particular MIME type.

Top ↑

Definition #Definition

GET /wp/v2/media

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/media>

Top ↑

Retrieve a Media Item #Retrieve a Media Item

Top ↑

Arguments #Arguments

context

Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

password

The password for the post if it is password protected.

Top ↑

Definition #Definition

GET /wp/v2/media/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/media/>

Top ↑

Create Media #Create Media

Top ↑

Arguments #Arguments

date

The date the object was published, in the site' s timezone.

date\_gmt

The date the object was published, as GMT.

slug

An alphanumeric identifier for the object unique to its type.



媒体

status

A named status for the object.

One of: publish, future, draft, pending, private

title

The title for the object.

author

The id for the author of the object.

comment\_status

Whether or not comments are open on the object.

One of: open, closed

ping\_status

Whether or not the object can be pinged.

One of: open, closed

meta

Meta fields.

alt\_text

Alternative text to display when resource is not displayed.

caption

The caption for the resource.

description

The description for the resource.

post

The id for the associated post of the resource.

Top ↑

Definition #Definition

POST /wp/v2/media

Top ↑

Update Media #Update Media

Top ↑

Arguments #Arguments

date

The date the object was published, in the site' s timezone.

date\_gmt

The date the object was published, as GMT.

slug

## 媒体

An alphanumeric identifier for the object unique to its type.

### status

A named status for the object.

One of: publish, future, draft, pending, private

### title

The title for the object.

### author

The id for the author of the object.

### comment\_status

Whether or not comments are open on the object.

One of: open, closed

### ping\_status

Whether or not the object can be pinged.

One of: open, closed

### meta

Meta fields.

### alt\_text

Alternative text to display when resource is not displayed.

### caption

The caption for the resource.

### description

The description for the resource.

### post

The id for the associated post of the resource.

### Top ↑

## Definition #Definition

POST /wp/v2/media/

## Example Request #Example Request

### Top ↑

## Delete Media #Delete Media

### Top ↑

## Arguments #Arguments

### force

Whether to bypass trash and force deletion.

### Top ↑

媒体

Definition #Definition

DELETE /wp/v2/media/

Example Request #Example Request

\$ curl -X DELETE <http://demo.wp-api.org/wp-json/wp/v2/media/>

# 用户

---

Schema #Schema

The schema defines all the fields that exist for a user object.

id

integer Unique identifier for the user.

Read only

Context: embed, view, edit

username

string Login name for the user.

Context: edit

name

string Display name for the user.

Context: embed, view, edit

first\_name

string First name for the user.

Context: edit

last\_name

string Last name for the user.

Context: edit

email

string, email The email address for the user.

Context: edit

url

string, uri URL of the user.

Context: embed, view, edit

description

string Description of the user.

Context: embed, view, edit

link

string, uri Author URL of the user.

Read only

Context: embed, view, edit

locale

用户

string Locale for the user.

Context: edit

One of: , en\_US

nickname

string The nickname for the user.

Context: edit

slug

string An alphanumeric identifier for the user.

Context: embed, view, edit

registered\_date

string, datetime (ISO8601) Registration date for the user.

Read only

Context: edit

roles

array Roles assigned to the user.

Context: edit

password

string Password for the user (never included).

Context:

capabilities

object All capabilities assigned to the user.

Read only

Context: edit

extra\_capabilities

object Any extra capabilities assigned to the user.

Read only

Context: edit

avatar\_urls

object Avatar URLs for the user.

Read only

Context: embed, view, edit

meta

object Meta fields.

Context: view, edit

Example Request #Example Request

用户

\$ curl -X OPTIONS -i <http://demo.wp-api.org/wp-json/wp/v2/users>

Top ↑

List Users #List Users

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

page Current page of the collection.

Default: 1

per\_page Maximum number of items to be returned in result set.

Default: 10

search Limit results to those matching a string.

exclude Ensure result set excludes specific IDs.

Default:

include Limit result set to specific IDs.

Default:

offset Offset the result set by a specific number of items.

order Order sort attribute ascending or descending.

Default: asc

One of: asc, desc

orderby Sort collection by object attribute.

Default: name

One of: id, include, name, registered\_date, slug, email, url

slug Limit result set to users with a specific slug.

roles Limit result set to users matching at least one specific role provided. Accepts csv list or single role.

Top ↑

Definition #Definition

GET /wp/v2/users

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/users>

Top ↑

Retrieve a User #Retrieve a User

Top ↑

Arguments #Arguments

context Scope under which the request is made; determines fields present in response.

Default: view

One of: view, embed, edit

Top ↑

Definition #Definition

GET /wp/v2/users/

Example Request #Example Request

\$ curl <http://demo.wp-api.org/wp-json/wp/v2/users/>

Top ↑

Create a User #Create a User

Top ↑

Arguments #Arguments

username Login name for the user.

Required: true

name Display name for the user.

first\_name First name for the user.

last\_name Last name for the user.

email The email address for the user.

Required: true

url URL of the user.

description Description of the user.

locale Locale for the user.

One of: , en\_US

nickname The nickname for the user.

slug An alphanumeric identifier for the user.

roles Roles assigned to the user.

password Password for the user (never included).

Required: true

meta Meta fields.

Top ↑

Definition #Definition

POST /wp/v2/users

Top ↑

Update a User #Update a User

Top ↑

## Arguments #Arguments

username Login name for the user.

name Display name for the user.

first\_name First name for the user.

last\_name Last name for the user.

email The email address for the user.

url URL of the user.

description Description of the user.

locale Locale for the user.

One of: , en\_US

nickname The nickname for the user.

slug An alphanumeric identifier for the user.

roles Roles assigned to the user.

password Password for the user (never included).

meta Meta fields.

Top ↑

## Definition #Definition

POST /wp/v2/users/

Example Request #Example Request

Top ↑

## Delete a User #Delete a User

Top ↑

## Arguments #Arguments

force Required to be true, as users do not support trashing.

reassign Reassign the deleted user' s posts and links to this user ID.

Top ↑

## Definition #Definition

DELETE /wp/v2/users/

Example Request #Example Request

\$ curl -X DELETE <http://demo.wp-api.org/wp-json/wp/v2/users/>



# 设置

## Schema

模式定义了设置对象存在的所有字段。

title

string

Site title.

Context:

description

string

Site description.

Context:

url

string,

uri

Site URL.

Context:

email

string,

email

This address is used for admin purposes. If you change this we will send you an email at your new address to confirm it. The new address will not become active until confirmed.

Context:

timezone

string

A city in the same timezone as you.

Context:

date\_format

string

A date format for all date strings.

Context:

time\_format

string

A time format for all time strings.

## 设置

Context:

start\_of\_week

number

A day number of the week that the week should start on.

Context:

language

string

WordPress locale code.

Context:

use\_smilies

boolean

Convert emoticons like :-) and :-P to graphics on display.

Context:

default\_category

number

Default category.

Context:

default\_post\_format

string

Default post format.

Context:

posts\_per\_page

number

Blog pages show at most.

Context:

## Example Request

---

```
$ curl -X OPTIONS -i http://demo.wp-api.org/wp-json/wp/v2/settings
```

Top ↑

## Update a Setting

---

##Arguments

title

Site title.

description

## 设置

Site description.

url

Site URL.

email

This address is used for admin purposes. If you change this we will send you an email at your new address to confirm it. The new address will not become active until confirmed.

timezone

A city in the same timezone as you.

date\_format

A date format for all date strings.

time\_format

A time format for all time strings.

start\_of\_week

A day number of the week that the week should start on.

language

WordPress locale code.

use\_smilies

Convert emoticons like :-) and :-P to graphics on display.

default\_category

Default category.

default\_post\_format

Default post format.

posts\_per\_page

Blog pages show at most.

# 使用REST API

---

## ##使用REST API

这些文章探讨了WordPress REST API的基本结构。

全局参数：了解适用于每个端点的全局REST API查询参数

分页：使用大量资源并控制从REST API接收的记录数量

链接和嵌入：了解如何读取和修改不同对象之间的连接，并将相关资源（如作者和媒体数据）嵌入到REST API的响应中

发现：确定站点支持的REST API资源以及它们位于何处

身份验证：授权您的REST API请求，以便您可以创建，更新和删除您的数据

常见问题：请参阅有关REST API的一些最常见的查询，并了解如何解决常见问题

## ##资源和实用程序

Backbone.js客户端：使用Backbone模型和集合与WP-Admin内的API进行交互

客户端库：用于简化与API交互的各种编程语言的实用程序

# 全局参数

API包括一些控制API如何处理请求/响应处理的全局参数（也称为“元参数”）。它们在实际资源本身以上的层次上运行，并且在所有资源上可用。

## \_jsonp

API原生支持JSONP响应，以允许传统浏览器和客户端的跨域请求。此参数需要一个JavaScript回调函数，它将被添加到数据中。然后可以通过 `<script>` 标签加载该URL。

回调函数可以包含任何字母数字，\_（下划线）或。（期）字。包含无效字符的回调将收到HTTP 400错误响应，不会调用回调。

注意：现代浏览器可以使用跨域请求的跨原始资源共享（CORS）预检要求，但可以使用JSONP来确保所有浏览器的支持。

浏览器支持

关于CORS的MDN文章

例如：

```
<script>
function receiveData( data ) {
  // Do something with the data here.
  // For demonstration purposes, we'll simply log it.
  console.log( data );
}
</script>
<script src="https://demo.wp-api.org/wp-json/?_jsonp=receiveData"></script>
```

## \_method（或X-HTTP-Method-Override标题）

某些服务器和客户端无法正确处理API使用的一些HTTP方法。例如，对资源的所有删除请求都使用DELETE方法，但有些客户端不提供发送此方法的功能。

为了确保与这些服务器和客户端的兼容性，API支持方法覆盖。这可以通过\_method参数或X-HTTP-Method-Override标头传递，其值设置为要使用的HTTP方法。

警报：客户端应该只发送具有POST请求的方法覆盖参数或头。使用GET请求的方法覆盖可能会导致请求被错误地缓存。

一个POST到/wp-json/wp/v2/posts/42?\_method=DELETE将被转换为DELETE到wp/v2/posts/42路由。

类似地，以下POST请求将成为DELETE：

```
POST /wp-json/wp/v2/posts/42 HTTP/1.1
```

```
Host: example.com
X-HTTP-Method-Override: DELETE
```

## \_envelope

与\_method类似，一些服务器，客户端和代理不支持访问完整的响应数据。API支持传递一个\_envelope参数，该参数发送正文中的所有响应数据，包括头文件和状态代码。

如果在查询字符串（GET参数）中传递了\_envelope参数，则启用信封模式。该参数不需要一个值（即？\_envelope有效），但如果客户端库需要，可以将其作为值“1”传递。

注意：为了将来的兼容性，不应该传递其他值。

包含的回应包括一个“假的”HTTP 200响应代码，没有额外的标头（Content-Type除外），应该确保响应正确地通过中介。

例如，给予对wp / v2 / users / me的GET的以下响应：

```
HTTP/1.1 302 Found
Location: http://example.com/wp-json/wp/v2/users/42

{
  "id": 42,
  ...
}
```

等效的包络响应（使用GET到wp / v2 / users / me ? \_envelope）将是：

```
HTTP/1.1 200 OK

{
  "status": 302,
  "headers": {
    "Location": "http://example.com/wp-json/wp/v2/users/42"
  },
  "body": {
    "id": 42
  }
}
```

## \_embed

大多数资源包括相关资源的链接。例如，一个帖子可以链接到父帖子，或者发表评论。为了减少所需的HTTP请求数量，客户端可能希望获取资源以及链接的资源。\_embed参数向服务器指示响应应包括这些嵌入式资源。

如果在查询字符串（GET参数）中传递了\_embed参数，则嵌入模式被启用。该参数不需要一个值（即\_embed是有效的），但是如果客户端库需要，则可以将“1”作为值传递。

**注意：**为了将来的兼容性，不应该传递其他值。

嵌入模式中的资源将在包含链接资源的\_links键旁边包含一个附加的\_embedded键。将嵌入嵌入式参数设置为true的连接。

有关链接和嵌入的更多信息，请参阅链接和嵌入页面。

# 分页

WordPress网站可以拥有大量内容 - 远远超过您希望在单个请求中下拉。 API端点默认为每个请求提供有限数量的项目，与在归档视图中WordPress站点默认为每页10个帖子相同。

## 分页参数

任何包含多个资源的API响应都支持几个常见的查询参数，以通过响应数据来处理分页：

? page = : 指定要返回的结果的页面。

例如/ wp / v2 / posts ? page = 2是帖子结果的第二页

通过检索/ wp / v2 / posts，然后/ wp / v2 / posts ? page = 2等等，您可以通过API访问每个可用的帖子，一次一页。

? per\_page = : 指定在一个请求中返回的记录数，指定为1到100之间的整数。

例如，/ wp / v2 / posts ? per\_page = 1将只返回集合中的第一个帖子

? offset = : 指定开始检索帖子的任意偏移量

例如，/ wp / v2 / posts ? offset = 6将使用每页的默认帖子数，但从集合中的第6个帖子开始

? per\_page = 5 & page = 4相当于 ? per\_page = 5 & offset = 15

提示：大型查询可能会损害网站性能，因此per\_page的上限为100条记录。如果您希望检索超过100条记录，例如构建所有可用类别的客户端列表，您可以创建多个API请求并将结果合并到应用程序中。

要确定有多少页面的数据可用，API返回两个标题字段与每个分页响应：

- X-WP-Total：集合中的记录总数
- X-WP-TotalPages：包含所有可用记录的总页数

通过检查这些标题字段，您可以确定API中有多少数据可用。

## 订购结果

除了上面详细分析的查询参数之外，其他几个参数控制了返回结果的顺序：

? order = : 控制结果是按升序还是降序返回

有效值为 ? order = asc ( 升序 ) 和 ? order = desc ( 降序 )。

默认情况下，所有本地集合都将按降序返回。

? orderby = : 控制集合被排序的字段

orderby的有效值将根据查询的资源而有所不同; 对于/ wp / v2 / posts集合，有效值

为 “date” ， “relevance” ， “id” ， “include” ， “title” 和 “slug”

有关其他集合支持的值，请参阅REST API参考

所有带有日期资源的集合都默认为orderby = date



# 链接和嵌入

WP REST API在整个API中集成了超链接，以允许可发现性和可浏览性，并在一个响应中嵌入相关资源。虽然REST API不完全符合整个HAL标准，但是它将按照下面的描述从该标准中实现`_links`和`_embedded`属性。

## ##链接

响应对象的`_links`属性包含指向其他API资源的链接映射，按“关系”分组。该关系指定链接资源与主资源的关联。（例子包括描述资源及其作者之间的关系的“作者”，或描述一个帖子与其标签或类别之间的关系“`wp:term`”）。关系是标准化关系，URI关系`https://api.w.org/term`）或紧凑的URI关系（如`wp:term`）。（紧凑型URI关系可以归一化为完全的URI关系，以确保完全兼容性，如果需要）。这与HTML

`<link>` 标签或 `<a rel="">` 链接类似。

链接是包含`href`属性的对象，其中包含资源的绝对URL，以及其他可选属性。这些包括内容类型，消歧信息以及可以通过链接采取的操作的数据。

对于收集响应（返回对象列表而不是顶级对象的响应），每个项目都包含链接，顶层响应包含通过链接头的链接。

**注意：**如果您的客户端库不允许访问标题，可以使用`_envelope`参数将标题作为正文数据。

## 响应示例

典型的单一帖子请求（`/wp/v2/posts/42`）：

```
{
  "id": 42,
  "_links": {
    "collection": [
      {
        "href": "https://demo.wp-api.org/wp-json/wp/v2/posts"
      }
    ],
    "author": [
      {
        "href": "https://demo.wp-api.org/wp-json/wp/v2/users/1",
        "embeddable": true
      }
    ]
  }
}
```

## 嵌入

可选地，一些链接的资源可以包括在响应中以减少所需的HTTP请求的数量。这些资源被“嵌入”为主要响应。

通过在请求上设置\_embed查询参数来触发嵌入。然后，这将包括与\_links键相邻的\_embedded键下的嵌入资源。此对象的布局反映\_links对象，但包含嵌入式资源代替链接属性。

只有嵌入标志设置为true的链接才能被嵌入，并且\_embed将导致嵌入所有嵌入式链接。只有包含嵌入式响应的关系才包含在\_embedded中，然而与混合嵌入式和不可嵌入链接的关系将包含用于不可嵌入链接的虚拟响应，以确保数字索引与\_links中的匹配。

## 响应示例

```
{
  "id": 42,
  "_links": {
    "collection": [
      {
        "href": "https://demo.wp-api.org/wp-json/wp/v2/posts"
      }
    ],
    "author": [
      {
        "href": "https://demo.wp-api.org/wp-json/wp/v2/users/1",
        "embeddable": true
      }
    ]
  },
  "_embedded": {
    "author": {
      "id": 1,
      "name": "admin",
      "description": "Site administrator"
    }
  }
}
```

# 发现

当您的客户与未知网站交谈时，您需要了解该网站的能力以及该网站的配置方式。这有几个步骤，这取决于您需要发现什么。

## ##发现API

连接到网站的第一步是找出网站是否启用了API。通常，您将使用用户输入的URL，因此您访问的网站可能是任何东西。发现步骤允许您验证API是否可用，以及如何访问它。

## ##链接头

处理发现的首选方法是向所提供的地址发送HEAD请求。REST API会自动将Link标头添加到所有前端页面，如下所示：

链接：<http://example.com/wp-json/>; rel = "https://api.w.org/" >

该URL指向API的根路径（/），然后将其用于进一步的发现步骤。

对于没有启用“漂亮的永久链接”的网站，/wp-json/不会被WordPress自动处理。这意味着将使用正常/默认的WordPress固定链接。这些标题看起来更像这样：

链接：[http://example.com/?rest\\_route=/](http://example.com/?rest_route=/); rel = "https://api.w.org/"

客户应牢记这种变化，并确保两条路线能够无缝地处理。

此自动发现可以应用于WordPress安装服务的任何URL，因此不需要添加用户输入的预处理。由于这是HEAD请求，所以请求应该安全地盲目发送到服务器，而不用担心会产生副作用。

## ##元素

对于具有HTML解析器或在浏览器中运行的客户端，通过 `<link>` 元素，前端页面的中包含了Link标题的等价物：

```
<link rel='https://api.w.org/' href='http://example.com/wp-json/' />
```

In-browser Javascript can access this via the DOM:

```
// jQuery method
var $link = jQuery( 'link[rel="https://api.w.org/"]' );
var api_root = $link.attr( 'href' );

// Native method
var links = document.getElementsByTagName( 'link' );
var link = Array.prototype.filter.call( links, function ( item ) {
    return ( item.rel === 'https://api.w.org/' );
} );
var api_root = link[0].href;
```

对于浏览器中的客户端，或客户端无法访问HTTP标头，这可能是更有用的发现API的方式。这与Atom / RSS feed发现类似，因此也可以自动为此目的使用现有的代码

改为代替。

## RSD ( 真正简单发现 )

对于支持XML-RPC发现的客户端，RSD方法可能更适用。这是一种通常用于XML-RPC的基于XML的发现格式。

RSD有两个步骤。第一步是找到RSD端点，以 `<link>` 元素的形式提供：

```
<link rel="EditURI" type="application/rsd+xml" title="RSD" href="http://example.com/xmlrpc.php?rsd" />
```

第二步是获取RSD文档并解析可用的端点。这涉及在如下文档中使用XML解析器：

```
<?xml version="1.0" encoding="utf-8"?>
<rsd version="1.0" xmlns="http://archipelago.phrasewise.com/rsd">
  <service>
    <engineName>WordPress</engineName>
    <engineLink>https://wordpress.org/</engineLink>
    <homePageLink>http://example.com/</homePageLink>
    <apis>
      <api name="WordPress" blogID="1" preferred="true" apiLink="http://example.com/xmlrpc.php" />
      <!-- ... -->
      <api name="WP-API" blogID="1" preferred="false" apiLink="http://example.com/wp-json/" />
    </apis>
  </service>
</rsd>
```

REST API始终具有名称属性，其值等于WP-API。

RSD是自动发现的最不喜欢的方法，有几个原因。基于RSD的发现的第一步涉及解析HTML以首先找到RSD文档本身，这相当于元素自动发现。然后，它需要另一个步骤来解析RSD文档，这需要一个完整的XML解析器。在可能的情况下，由于复杂性，我们建议避免基于RSD的发现，但如果现有的XML-RPC客户端已经启用了RSD解析器，则可能更喜欢使用此方法。对于希望使用REST API作为逐步增强代码库的XML-RPC客户机，这避免了需要支持不同形式的发现。

### ##认证发现

发现也可用于通过API可用的身份验证方法。API根的响应是描述API的对象，其中包含一个验证密钥：

```
{
  "name": "Example WordPress Site",
  "description": "YOLO",
  "routes": { ... },
  "authentication": {
    "oauth1": {
```

```

        "request": "http://example.com/oauth/request",
        "authorize": "http://example.com/oauth/authorize",
        "access": "http://example.com/oauth/access",
        "version": "0.1"
    }
}
}

```

认证值是认证方法ID与认证选项的映射（关联数组）。这里提供的选项特定于身份验证方法本身。有关特定身份验证方法的选项，请参阅身份验证文档。

### ##扩展发现

一旦您发现了API，下一步就是检查API支持的内容。API的索引公开了命名空间项，其中包含支持的API的扩展名。

对于使用版本4.4到4.6的WordPress网站，只有基本API基础架构可用，而不是具有端点的完整API。这也包括oEmbed端点：

```

{
    "name": "Example WordPress Site",
    "namespaces": [
        "oembed/1.0/"
    ]
}

```

具有完整API（即安装了WordPress 4.7+或安装了REST API插件）的网站也将在命名空间中使用wp / v2项目：

```

{
    "name": "Example WordPress Site",
    "namespaces": [
        "wp/v2",
        "oembed/1.0/"
    ]
}

```

在尝试使用任何核心端点之前，您应该确保通过检查wp / v2支持来检查API是否受支持。WordPress 4.4启用所有站点的API基础架构，但不包括wp / v2下的核心端点。核心终端在WordPress 4.7中添加。

这种相同的机制可用于检测支持REST API的任何插件的支持。例如，拿一个注册以下路由的插件：

```

<?php
register_rest_route( 'testplugin/v1', '/testroute', array( /* ... */ ) );

```

This would add the testplugin/v1 namespace to the index:

```
{  
  "name": "Example WordPress Site",  
  "namespaces": [  
    "wp/v2",  
    "oembed/1.0/",  
    "testplugin/v1"  
  ]  
}
```

# 认证

## Cookie验证

Cookie认证是包含的标准认证方法WordPress。当您登录到仪表板时，这将正确设置Cookie对于您来说，插件和主题开发者只需要登录用户。

然而，REST API包括一种名为nonces的技术来避免CSRF问题。这样可以防止其他站点强制您未明确地执行操作打算这样做这需要稍微特别处理API。

对于使用内置Javascript API的开发人员，这是自动处理的为您。这是推荐使用API插件和主题的方法。自定义数据模型可以扩展wp.api.models.Base以确保发送正确地任何自定义请求。

对于开发手工Ajax请求的开发人员，需要传递该随机数与每个请求。API使用的是将动作设置为wp\_rest的随机数。这些然后可以通过\_wpnonce数据参数（POST）传递给API数据或查询GET请求），或通过X-WP-Nonce头。

**注意：**直到最近，大多数软件对DELETE请求都有很多支持。对于例如，PHP不会将DELETE请求的请求体转换为超级全球。因此，将随机数作为标题提供是最可靠的方法。

重要的是要记住，这种认证方法依赖于WordPress饼干。因此，此方法仅在内部使用REST API时适用的WordPress和当前用户登录。此外，当前用户必须具有执行正在执行的动作的适当能力。

例如，这是内置的JavaScript客户端如何创建的随机数：

```
<?php
wp_localize_script( 'wp-api', 'wpApiSettings', array(
    'root' => esc_url_raw( rest_url() ),
    'nonce' => wp_create_nonce( 'wp_rest' )
) );
```

然后在基本模型中使用它：

```
options.beforeSend = function(xhr) {
    xhr.setRequestHeader( 'X-WP-Nonce', wpApiSettings.nonce);

    if (beforeSend) {
        return beforeSend.apply(this, arguments);
    }
};
```

以下是使用jQuery AJAX编辑帖子标题的示例：

```
$.ajax( {  
  url: wpApiSettings.root + 'wp/v2/posts/1',  
  method: 'POST',  
  beforeSend: function ( xhr ) {  
    xhr.setRequestHeader( 'X-WP-Nonce', wpApiSettings.nonce );  
  },  
  data:{  
    'title' : 'Hello Moon'  
  }  
} ).done( function ( response ) {  
  console.log( response );  
} );
```

## 认证插件

虽然Cookie身份验证是WordPress本身可用的唯一身份验证机制，但可以添加插件以支持将从远程应用程序工作的替代认证模式。一些示例插件是OAuth 1.0a服务器，应用程序密码和JSON Web令牌。



# 经常问的问题

此页面提供了解使用API时可能出现的常见问题和问题的解决方案。如果您的问题在这里没有解释，可能已经在WordPress支持论坛中得到回答。

## 我可以禁用REST API吗？

您不应该禁用REST API，因为这样做会破坏将取决于API处于活动状态的将来WordPress管理功能。但是，您可以使用过滤器来要求API消费者进行身份验证，从而有效防止匿名外部访问。有关详细信息，请参阅下文。

## 需要所有Requests的身份验证

您可以通过向rest\_authentication\_errors过滤器添加is\_user\_logged\_in检查来要求对所有REST API请求进行身份验证：

```
add_filter( 'rest_authentication_errors', function( $result ) {
    if ( ! empty( $result ) ) {
        return $result;
    }
    if ( ! is_user_logged_in() ) {
        return new WP_Error( 'rest_not_logged_in', 'You are not currently logged in.', array( 'status' => 401 ) );
    }
    return $result;
});
```

## 我可以在插件中从PHP发出API请求吗？

是的你可以！在其他WordPress代码内部使用rest\_do\_request来制作API请求：

```
$request = new WP_REST_Request( 'GET', '/wp/v2/posts' );
// Set one or more request query parameters
$request->set_param( 'per_page', 20 );
$response = rest_do_request( $request );
```

## ? filter = 查询参数发生了什么？

当REST API合并到WordPress核心中时，过滤器查询参数被删除，以防止将来的兼容性和维护问题。在REST API项目的起源上，使用? filter查询参数传递任意WP\_Query参数到API的功能是必要的，但大多数API响应过滤功能已经被更强大的查询参数所取代，如? categories =，? slug =和? per\_page =。

应尽可能使用第一方查询参数。但是，如果需要，其余的过滤器插件可以恢复在API请求中传递任意的过滤器值的功能。

## 查询参数不起作用

---

如果您发现诸如 ?page = 2 或 ?\_embed 的查询参数没有任何影响，您的服务器可能没有被正确配置来检测它们。如果您使用Nginx来为您的网站提供服务，请在您的网站配置中查找一个try\_files行。如果看起来像这样：

```
try_files $uri $uri/ /index.php$args;
```

改为：

```
try_files $uri $uri/ /index.php$is_args$args;
```

添加\$is\_args（如果找到查询参数，将打印一个字符）将允许WordPress正确接收和解释查询参数。

## 为什么验证不起作用？

---

如果您发现您正在发送身份验证标头，但请求未被接受，并且您在CGI环境中使用Apache，则Apache可能会剥离头文件。尝试将其添加到配置文件或.htaccess中：

```
SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
```

# 骨干JavaScript客户端

REST API包含一个JavaScript / Backbone客户端库。

该库为WP REST API提供了一个界面，为所有展现API模式的端点提供了骨干模型和集合。

##使用

激活WP-API插件。 直接排队脚本：

```
wp_enqueue_script( 'wp-api' );
```

或作为脚本的依赖：

```
wp_enqueue_script( 'my_script', 'path/to/my/script', array( 'wp-api' ) );
```

库解析根端点（“Schema”）并创建匹配的Backbone模型和集合。 你现在可以有两个根对象：

`wp.api.models` 和 `wp.api.collections` 。

模型和收藏包括：

模型:

- Category
- Comment
- Media
- Page
- PageMeta
- PageRevision
- Post
- PostMeta
- PostRevision
- Schema
- Status
- Tag
- Taxonomy
- Type
- User

Collections:

- Categories

- Comments
- Media
- PageMeta
- PageRevisions
- Pages
- Posts
- Statuses
- Tags
- Taxonomies
- Types
- Users

您可以使用这些端点来使用标准的Backbone方法来读取，更新，创建和删除项目（获取，同步，保存和销毁模型，同步收集）。您还可以扩展这些对象，使其成为您自己的，并在其上构建您的视图。

## Default values

---

每个模型和集合都包含对其默认值的引用，例如：

`wp.api.models.Post.prototype.args`

- author: null
- comment\_status: null
- content: null
- date: null
- date\_gmt: null
- excerpt: null
- featured\_image: null
- format: null
- modified: null
- modified\_gmt: null
- password: null
- ping\_status: null
- slug: null
- status: null
- sticky: null
- title: null

## 可用的方法

每个模型和集合都包含相应端点支持的方法列表。例如，从 `wp.api.models.Post` 创建的模型有一个方法数组：

```
["GET", "POST", "PUT", "PATCH", "DELETE"]
```

## 接受的选项

每个模型和集合包含相应端点接受的选项列表（注意，在创建模型或集合时，选项作为第二个参数传递），例如：

```
wp.api.collections.Posts.prototype.options
* author
* context
* filter
* order
* orderby
* page
* per_page
* search
* status
```

## 本地化API模式

客户端将接受并使用本地化模式作为“wpApiSettings”对象的一部分。默认情况下，模式目前不会传递；而是客户端向API加载ajax请求以加载模式，然后将其缓存到浏览器的会话存储中（如果可用）。启用了“SCRIPT\_DEBUG”的client-js插件使用本地化的模式。检查[client-js示例]（<https://github.com/WP-API/client-js/blob/master/client-js.php>）或者这个分支，它试图只将每个客户端的模式本地化一次。（<https://github.com/WP-API/client-js/compare/features/only-localize-schema-once?expand=1>）。

## 等待客户端加载

客户端启动是异步的。如果api模式是本地化的，客户端可以立即启动；如果客户端不是ajax请求来加载模式。客户端公开了一个负担承诺，提供可靠的等待，等待客户端准备好：

```
wp.api.loadPromise.done( function() {
//... use the client here
} )
```

## 模型示例：

要创建一个帖子并修改其类别，请确保您已登录，然后：

```
// Create a new post
var post = new wp.api.models.Post( { title: 'This is a test post' } );
post.save();

// Load an existing post
var post = new wp.api.models.Post( { id: 1 } );
post.fetch();

// Get a collection of the post's categories (returns a promise)
// Uses _embedded data if available, in which case promise resolves immediately.

post.getCategories().done( function( postCategories ) {
// ... do something with the categories.
// The new post has an single Category: Uncategorized
console.log( postCategories[0].name );
// response -> "Uncategorized"
} );

// Get a posts author User model.
post.getAuthorUser().done( function( user ){
// ... do something with user
console.log( user.get( 'name' ) );
} );

// Get a posts featured image Media model.
post.getFeaturedImage().done( function( image ){
// ... do something with image
console.log( image );
} );

// Set the post categories.
post.setCategories( [ 'apples', 'oranges' ] );

// Get all the categories
var allCategories = new wp.api.collections.Categories()
allCategories.fetch();

var appleCategory = allCategories.findWhere( { slug: 'apples' } );

// Add the category to the postCategories collection we previously fetched.
appleCategory.set( 'parent_post', post.get( 'id' ) );

// Use the POST method so Backbone will not PUT it even though it has an id.
postCategories.create( appleCategory.toJSON(), { type: 'POST' } );

// Remove the Uncategorized category
postCategories.at( 0 ).destroy();

// Check the results - re-fetch
```

```
postCategories = post.getCategories();

postCategories.at( 0 ).get( 'name' );
// response -> "apples"
```

## 集合示例：

得到最后10个帖子：

```
var postsCollection = new wp.api.collections.Posts();
postsCollection.fetch();
```

获取最后25个帖子：

```
postsCollection.fetch( { data: { per_page: 25 } } );
```

使用过滤器更改订单和orderby选项：

```
postsCollection.fetch( { data: { 'filter': { 'orderby': 'title', 'order': 'ASC' } } } );
```

所有收藏都自动支持分页，您可以使用 `more` 获得下一页结果：

```
postsCollection.more();
```

获取一个集合的第5页：

```
posts.fetch( { data: { page: 5 } } );
```

检查收集是否有更多的帖子：

```
posts.hasMore();
```

## 使用修订

您可以使用PostRevisions或PageRevisions集合或通过Post或页面集合访问帖子或页面修订。

例如，要获取所有版本的帖子ID 1的集合：

```
var revisions = new wp.api.collections.PostRevisions({}, { parent: 1 });
```

修订集合也可以通过父母的收藏进行访问。这个例子使得2个HTTP请求而不是一个HTTP请求，但现在可以使用原始帖子及其修订版本：

```
var post = new wp.api.models.Post( { id: 1 } );
post.fetch();
post.getRevisions().done( function( revisions ){
  console.log( revisions );
});
```

如果您将自定义端点添加到api，它们也将作为模型/集合使用。例如，当您[为您的自定义帖子类型添加REST API支持] ( <http://v2.wp-api.org/extending/custom-content-types/> ) 时，您将获得新的模型和集合。注意：由于模式存储在用户的会话缓存中以避免重新获取，因此您可能需要打开一个新的选项卡才能对Schema进行新的读取。



# 客户端库

---

API可以通过发送基本的HTTP请求从任何应用程序使用;然而, 客户端库简化了查询或创建特定资源的过程。这些库可以轻松地使用各种编程语言将外部应用程序连接到WordPress REST API。

**警报:** 除了Backbone.js客户端, 这些库不是WordPress核心的一部分, 并不一定由WordPress REST API团队维护或认可。

AND

**注意:** 要从WordPress管理员, 主题或插件外部执行身份验证请求, 需要单独的身份验证插件。

## JavaScript

---

Backbone.js客户端内置于WordPress内核, 并提供Backbone.js模型和集合以使用REST API资源。

node-wpapi是一个同构的JavaScript客户端库, 用于使用直观的链接语法查询或写入REST API。它可以与Node.js或客户端JavaScript应用程序一起使用。

ember-wordpress提供了Ember Data和REST API之间的连接

## ruby

---

wp-api-client: 用Ruby编写的只读REST API客户端。

# 扩展REST API

---

## ##指导

添加端点：为插件或应用程序创建自定义REST API端点

使用自定义内容类型：了解如何通过REST API与自定义帖子类型和自定义分类法进行交互

修改响应：使用register\_rest\_field将字段添加到REST API响应对象

## ##资源

定义您的API模式：定义REST API资源及其参数的模式

词汇表：通过我们的文档中使用的短语加快速度

路由和端点：深入了解REST API路由及其提供的端点的细微差别

控制器类：发现如何构造和扩展REST API端点控制器类

# 添加自定义端点

WordPress REST API不仅仅是一组默认路由。它也是创建自定义路由和端点的工具。WordPress前端提供了一组默认的URL映射，但是用于创建它们的工具（例如Rewrites API以及查询类：WP\_Query，WP\_User等）也可用于创建自己的URL映射，或自定义查询。

本文档详细介绍了如何使用自己的端点创建完全自定义的路由。我们将首先通过一个简短的例子，然后将其扩展到内部使用的完全控制器模式。

## 基础不错

那么你要添加自定义端点到API？太棒了！让我们开始一个简单的例子。

我们从一个简单的函数开始，看起来像这样：

```
<?php
/**
 * Grab latest post title by an author!
 *
 * @param array $data Options for the function.
 * @return string|null Post title for the latest,
 * or null if none.
 */
function my_awesome_func( $data ) {
    $posts = get_posts( array(
        'author' => $data['id'],
    ) );

    if ( empty( $posts ) ) {
        return null;
    }

    return $posts[0]->post_title;
}
```

要通过API提供此功能，我们需要注册一个路由。这告诉API使用我们的函数来响应给定的请求。我们通过一个名为register\_rest\_route的函数来执行此操作，该函数应该在rest\_api\_init的回调中调用，以避免在API未加载时执行额外的工作。

我们需要将三件事传递给register\_rest\_route：命名空间，我们想要的路由和选项。我们稍后会回到命名空间，但现在我们来选择myplugin / v1。我们将路由与/ author / {id}匹配，其中{id}是一个整数。

```
<?php
add_action( 'rest_api_init', function () {
    register_rest_route( 'myplugin/v1', '/author/(?P<id>\d+)', array(
```

```
'methods' => 'GET',
'callback' => 'my_awesome_func',
) );
} );
```

现在，我们只注册路由的一个端点。（“Route”是URL，而“endpoint”是与之相对应的方法和URL），如果您的站点域是[example.com](http://example.com)，并且您保留了API路径的wp-json，那么完整的URL将是[http://example.com/wp-json/myplugin/v1/author/\(?P\d+\)](http://example.com/wp-json/myplugin/v1/author/(?P\d+))。每个路由都可以有任意数量的端点，对于每个端点，您可以定义允许的HTTP方法，用于响应该请求的回调函数和用于创建自定义权限的权限回调。此外，您可以在请求中定义允许的字段，并且每个字段指定默认值，清理回调，验证回调以及该字段是否必需。

## 命名空间

命名空间是端点的URL的第一部分。它们应该用作供应商/包前缀，以防止自定义路由之间的冲突。命名空间允许两个插件添加相同名称的路由，具有不同的功能。

命名空间一般应遵循供应商/v1的模式，供应商通常是您的插件或主题，而v1代表API的第一个版本。如果您需要破坏与新端点的兼容性，那么您可以将其限制到v2。

上述情况，两个具有相同名称的路由，来自两个不同的插件，要求所有供应商使用唯一的命名空间。如果没有这样做的话，就不会在主题或插件中使用供应商功能前缀，类前缀和/或类名称空间，这是非常\_doing\_it\_wrong。使用命名空间的另一个好处是客户端可以检测到对您的自定义API的支持。API索引列出了一个站点上可用的命名空间：

```
{
  "name": "WordPress Site",
  "description": "Just another WordPress site",
  "url": "http://example.com/",
  "namespaces": [
    "wp/v2",
    "vendor/v1",
    "myplugin/v1",
    "myplugin/v2",
  ]
}
```

如果客户端想要检查您的API是否存在于站点上，则可以检查该列表。（有关详细信息，请参阅“发现指南”。）

## 参数

默认情况下，路由接收从请求传入的所有参数。这些被合并到一组参数中，然后被添加到Request对象中，该对象作为第一个参数传递给您的端点：

```
<?php
function my_awesome_func( WP_REST_Request $request ) {
    // You can access parameters via direct array access on the object:
    $param = $request['some_param'];

    // Or via the helper method:
    $param = $request->get_param( 'some_param' );

    // You can get the combined, merged set of parameters:
    $parameters = $request->get_params();

    // The individual sets of parameters are also available, if needed:
    $parameters = $request->get_url_params();
    $parameters = $request->get_query_params();
    $parameters = $request->get_body_params();
    $parameters = $request->get_json_params();
    $parameters = $request->get_default_params();

    // Uploads aren't merged in, but can be accessed separately:
    $parameters = $request->get_file_params();
}
```

（要准确了解参数的合并方式，请检查WP\_REST\_Request :: get\_parameter\_order（）的源代码；基本顺序是正文，查询，URL，然后默认）。

通常，您将获得所有参数不变。但是，您可以在注册路由时注册自己的参数，从而可以对其进行清理和验证。

如果请求具有Content-type：application / json标头集合和正文中的有效JSON，则get\_json\_params（）将返回已解析的JSON主体作为关联数组。

参数被定义为每个端点的键参数中的映射（在回调选项旁边）。此映射使用该键的参数的名称，其值为该参数的选项的映射。该数组可以包含默认的key，必需的，sanitize\_callback和validate\_callback。

- default：用作参数的默认值，如果没有提供参数。
- required：如果定义为true，并且没有为该参数传递值，则会返回错误。如果设置默认值，则不起作用，因为参数始终具有值。
- validate\_callback：用于传递将传递参数值的函数。如果值有效，该函数应返回true，否则返回false。
- sanitize\_callback：用于传递一个函数，用于在将参数值传递给主回调之前对其进行消毒。

使用sanitize\_callback和validate\_callback允许主回调仅用于处理请求，并使用WP\_REST\_Response类准备要返回的数据。通过使用这两个回调，您将能够安全地假设您的输入在处理时是有效和安全的。

以我们前面的例子，我们可以确保传入的参数总是一个数字：

```
<?php
add_action( 'rest_api_init', function () {
    register_rest_route( 'myplugin/v1', '/author/(?P<id>\d+)', array(
```

```

    'methods' => 'GET',
    'callback' => 'my_awesome_func',
    'args' => array(
        'id' => array(
            'validate_callback' => function($param, $request, $key) {
                return is_numeric( $param );
            }
        ),
    ),
) );
} );

```

您还可以将函数名称传递给`validate_callback`，但是直接传递诸如`is_numeric`之类的某些函数将不仅会传递有关传递额外参数的警告，还会返回`NULL`，从而使用无效数据调用回调函数。我们希望最终在WordPress核心解决这个问题。

我们也可以使用像`'sanitize_callback'=>'absint'`这样的东西，但验证会抛出一个错误，让客户了解他们做错了什么。当您宁愿更改正在输入的数据而不是抛出错误（例如无效的HTML）时，消毒是有用的。

## 返回值

调用回调后，返回值将被转换为JSON，并返回给客户端。这允许您基本返回任何形式的数据。在我们上面的例子中，我们返回一个字符串或者`null`，它们被API自动处理并转换成JSON。

像任何其他WordPress功能一样，您也可以返回`WP_Error`实例。该错误信息将传递给客户端，以及500个内部服务错误状态代码。您可以通过将`WP_Error`实例数据中的状态选项设置为代码来进一步自定义错误，例如对于错误的输入数据为400。

以前的例子，我们现在可以返回一个错误实例：

```

<?php
/**
 * Grab latest post title by an author!
 *
 * @param array $data Options for the function.
 * @return string|null Post title for the latest,
 * or null if none.
 */
function my_awesome_func( $data ) {
    $posts = get_posts( array(
        'author' => $data['id'],
    ) );

    if ( empty( $posts ) ) {
        return new WP_Error( 'awesome_no_author', 'Invalid author', array( 'status'
=> 404 ) );
    }
}

```

```
return $posts[0]->post_title;
}
```

当作者没有任何属于他们的帖子时，这将向客户端返回404 Not Found错误：

HTTP/1.1 404 Not Found

```
[{
  "code": "no_author",
  "message": "Invalid author",
  "data": { "status": 404 }
}]
```

要获得更高级的使用，您可以返回一个WP\_REST\_Response对象。此对象“包裹”正常的身体数据，但允许您返回自定义状态代码或自定义标题。您还可以添加链接到您的回复。最快的方法是通过构造函数：

```
<?php
$data = array( 'some', 'response', 'data' );

// Create the response object
$response = new WP_REST_Response( $data );

// Add a custom status code
$response->set_status( 201 );

// Add a custom header
$response->header( 'Location', 'http://example.com/' );
```

当包装现有的回调时，应该始终在返回值上使用rest\_ensure\_response（）。这将从端点返回原始数据，并自动将其转换为WP\_REST\_Response。（请注意，WP\_Error不会转换为WP\_REST\_Response以允许正确的错误处理。）

## 权限回调

您还可以注册端点的权限回调。这是一个功能，它检查用户是否可以在调用真正的回调之前执行动作（读取，更新等）。这样就可以让API告知客户他们可以在给定的URL上执行什么操作，而无需首先尝试该请求。

此回调可以注册为permission\_callback，再次在您的回调选项旁边的端点选项中。此回调应返回一个布尔值或WP\_Error实例。如果此函数返回true，则响应将被处理。如果返回false，将返回默认错误消息，并且请求不会继续处理。如果它返回一个WP\_Error，该错误将返回给客户端。

权限回调在远程身份验证之后运行，这将设置当前用户。这意味着您可以使用current\_user\_can来检查已验证的用户是否具有适当的操作能力，或者基于当前用户ID的任何其他检查。在可能的情况下，应始终使用current\_user\_can;而不是检查用户是否登录（验证），检查它们是否可以执行操作（授权）。

继续我们以前的例子，我们可以使它只有编辑或以上才能查看作者的数据。我们可以在这里检查一些不同的功能，但最好的是`edit_others_posts`，这是编辑器的核心。为此，我们只需要一个回调：

```
<?php
add_action( 'rest_api_init', function () {
    register_rest_route( 'myplugin/v1', '/author/(?P<id>\d+)', array(
        'methods' => 'GET',
        'callback' => 'my_awesome_func',
        'args' => array(
            'id' => array(
                'validate_callback' => 'is_numeric'
            ),
        ),
        'permission_callback' => function () {
            return current_user_can( 'edit_others_posts' );
        }
    ) );
} );
```

请注意，权限回调也接收到`Request`对象作为第一个参数，因此如果需要，您可以根据请求参数进行检查。

## ##控制器模式

控制器模式是使用API处理复杂端点的最佳做法。

在阅读本节之前，建议您阅读“扩展内部类”。这样做会使您熟悉默认路由使用的模式，这是最佳做法。虽然不需要您用于处理请求的类扩展了`WP_REST_Controller`类或扩展它的类，但这样做允许您继承在这些类中完成的工作。此外，您可以放心，您是根据您使用的控制器方法遵循最佳做法。

在他们的核心，控制器只不过是一组通用命名的方法来配合REST约定，还有一些方便的帮手。控制器以`register_routes`方法注册其路由，使用`get_items`，`get_item`，`create_item`，`update_item`和`delete_item`响应请求，并具有类似的命名权限检查方法。遵循此模式将确保您不会错过端点中的任何步骤或功能。

要使用控制器，您首先需要对基本控制器进行子类化。这为您提供了一套基本的方法，可以为您添加自己的行为。

一旦我们对控制器进行了子类化，我们需要实例化该类以使其工作。这应该在挂接到`rest_api_init`的回调内部完成，这确保我们只需要在我们需要时实例化类。正常控制器模式是在此回调内调用`$ controller->register_routes ( )`，然后该类可以注册其端点。

## 例子

以下是“起始”定制路线：

```
<?php
```



```

class Slug_Custom_Route extends WP_REST_Controller {

    /**
     * Register the routes for the objects of the controller.
     */
    public function register_routes() {
        $version = '1';
        $namespace = 'vendor/v' . $version;
        $base = 'route';
        register_rest_route( $namespace, '/' . $base, array(
            array(
                'methods'             => WP_REST_Server::READABLE,
                'callback'            => array( $this, 'get_items' ),
                'permission_callback' => array( $this, 'get_items_permissions_check' ),
                'args'                => array(

            ),
        ),
        array(
            'methods'             => WP_REST_Server::CREATABLE,
            'callback'            => array( $this, 'create_item' ),
            'permission_callback' => array( $this, 'create_item_permissions_check' )
        ,
            'args'                => $this->get_endpoint_args_for_item_schema( true ),
        ),
    );
    register_rest_route( $namespace, '/' . $base . '/(?P<id>[\d]+)', array(
        array(
            'methods'             => WP_REST_Server::READABLE,
            'callback'            => array( $this, 'get_item' ),
            'permission_callback' => array( $this, 'get_item_permissions_check' ),
            'args'                => array(
                'context'         => array(
                    'default'      => 'view',
                ),
            ),
        ),
        array(
            'methods'             => WP_REST_Server::EDITABLE,
            'callback'            => array( $this, 'update_item' ),
            'permission_callback' => array( $this, 'update_item_permissions_check' )
        ,
            'args'                => $this->get_endpoint_args_for_item_schema( false ),
        ),
        array(
            'methods'             => WP_REST_Server::DELETABLE,
            'callback'            => array( $this, 'delete_item' ),
            'permission_callback' => array( $this, 'delete_item_permissions_check' )
        ,
            'args'                => array(

```

```

        'force'      => array(
            'default'  => false,
        ),
    ),
),
);
register_rest_route( $namespace, '/' . $base . '/schema', array(
    'methods'      => WP_REST_Server::READABLE,
    'callback'     => array( $this, 'get_public_item_schema' ),
) );
}

/**
 * Get a collection of items
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|WP_REST_Response
 */
public function get_items( $request ) {
    $items = array(); //do a query, call another class, etc
    $data = array();
    foreach( $items as $item ) {
        $itemdata = $this->prepare_item_for_response( $item, $request );
        $data[] = $this->prepare_response_for_collection( $itemdata );
    }

    return new WP_REST_Response( $data, 200 );
}

/**
 * Get one item from the collection
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|WP_REST_Response
 */
public function get_item( $request ) {
    //get parameters from request
    $params = $request->get_params();
    $item = array();//do a query, call another class, etc
    $data = $this->prepare_item_for_response( $item, $request );

    //return a response or error based on some conditional
    if ( 1 == 1 ) {
        return new WP_REST_Response( $data, 200 );
    }else{
        return new WP_Error( 'code', __( 'message', 'text-domain' ) );
    }
}

/**

```

```

* Create one item from the collection
*
* @param WP_REST_Request $request Full data about the request.
* @return WP_Error|WP_REST_Request
*/
public function create_item( $request ) {

    $item = $this->prepare_item_for_database( $request );

    if ( function_exists( 'slug_some_function_to_create_item' ) ) {
        $data = slug_some_function_to_create_item( $item );
        if ( is_array( $data ) ) {
            return new WP_REST_Response( $data, 200 );
        }
    }

    return new WP_Error( 'cant-create', __( 'message', 'text-domain' ), array( 'status' => 500 ) );
}

/**
* Update one item from the collection
*
* @param WP_REST_Request $request Full data about the request.
* @return WP_Error|WP_REST_Request
*/
public function update_item( $request ) {
    $item = $this->prepare_item_for_database( $request );

    if ( function_exists( 'slug_some_function_to_update_item' ) ) {
        $data = slug_some_function_to_update_item( $item );
        if ( is_array( $data ) ) {
            return new WP_REST_Response( $data, 200 );
        }
    }

    return new WP_Error( 'cant-update', __( 'message', 'text-domain' ), array( 'status' => 500 ) );
}

/**
* Delete one item from the collection
*
* @param WP_REST_Request $request Full data about the request.
* @return WP_Error|WP_REST_Request
*/
public function delete_item( $request ) {
    $item = $this->prepare_item_for_database( $request );

```

```

    if ( function_exists( 'slug_some_function_to_delete_item' ) ) {
        $deleted = slug_some_function_to_delete_item( $item );
        if ( $deleted ) {
            return new WP_REST_Response( true, 200 );
        }
    }

    return new WP_Error( 'cant-delete', __( 'message', 'text-domain' ), array( '
status' => 500 ) );
}

/**
 * Check if a given request has access to get items
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|bool
 */
public function get_items_permissions_check( $request ) {
    //return true; <--use to make readable by all
    return current_user_can( 'edit_something' );
}

/**
 * Check if a given request has access to get a specific item
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|bool
 */
public function get_item_permissions_check( $request ) {
    return $this->get_items_permissions_check( $request );
}

/**
 * Check if a given request has access to create items
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|bool
 */
public function create_item_permissions_check( $request ) {
    return current_user_can( 'edit_something' );
}

/**
 * Check if a given request has access to update a specific item
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|bool
 */
public function update_item_permissions_check( $request ) {
    return $this->create_item_permissions_check( $request );
}

```

```

}

/**
 * Check if a given request has access to delete a specific item
 *
 * @param WP_REST_Request $request Full data about the request.
 * @return WP_Error|bool
 */
public function delete_item_permissions_check( $request ) {
    return $this->create_item_permissions_check( $request );
}

/**
 * Prepare the item for create or update operation
 *
 * @param WP_REST_Request $request Request object
 * @return WP_Error|object $prepared_item
 */
protected function prepare_item_for_database( $request ) {
    return array();
}

/**
 * Prepare the item for the REST response
 *
 * @param mixed $item WordPress representation of the item.
 * @param WP_REST_Request $request Request object.
 * @return mixed
 */
public function prepare_item_for_response( $item, $request ) {
    return array();
}

/**
 * Get the query params for collections
 *
 * @return array
 */
public function get_collection_params() {
    return array(
        'page' => array(
            'description' => 'Current page of the collection.',
            'type' => 'integer',
            'default' => 1,
            'sanitize_callback' => 'absint',
        ),
        'per_page' => array(
            'description' => 'Maximum number of items to be returned in result set.',
            'type' => 'integer',

```

```
        'default'           => 10,  
        'sanitize_callback' => 'absint',  
    ),  
    'search' => array(  
        'description' => 'Limit results to those matching a string.',  
        'type'        => 'string',  
        'sanitize_callback' => 'sanitize_text_field',  
    ),  
);  
}
```

# 自定义内容类型

REST API可以使用与默认帖子类型或分类术语控制器相同的控制器为wp / v2命名空间中的自定义帖子类型和自定义分类法创建路线。或者，您可以使用自己的控制器和命名空间。本文将介绍如何使用自定义内容类型的API路由的默认控制器。这是最简单的方法，并确保与第三方兼容的最高机会。

##使用REST API支持注册自定义帖子类型

注册自定义帖子类型时，如果希望通过REST API可用，您应该在传递给register\_post\_type的参数中设置'show\_in\_rest'=> true。将此参数设置为true将在wp / v2命名空间中添加路由。

```
/**
 * Register a book post type, with REST API support
 *
 * Based on example at: https://codex.wordpress.org/Function_Reference/register_post_type
 */
add_action( 'init', 'my_book_cpt' );
function my_book_cpt() {
    $args = array(
        'public'           => true,
        'show_in_rest'     => true,
        'label'            => 'Books'
    );
    register_post_type( 'book', $args );
}
```

您可以选择设置rest\_base参数来更改基本URL，否则默认为该类型的名称。在下面的示例中，“books”用作rest\_base的值。这将使路由的URL为wp-json / v2 / books，而不是wp-json / v2 / book /，这将是默认的。另外，您可以传递rest\_controller\_class的参数。这个类必须是WP\_REST\_Controller的子类。默认情况下，WP\_REST\_Posts\_Controller用作控制器。如果您使用的是自定义控制器，则可能不在wp / v2命名空间内。下面是一个注册一个post类型，一个完整的标签，对REST API的支持，一个定制的rest\_base以及默认控制器的显式注册表的例子：

```
/**
 * Register a book post type, with REST API support
 *
 * Based on example at: https://codex.wordpress.org/Function_Reference/register_post_type
 */
add_action( 'init', 'my_book_cpt' );
function my_book_cpt() {
    $labels = array(
        'name'                => _x( 'Books', 'post type general name', 'your-plugin
```

```

-textdomain' ),
    'singular_name'      => _x( 'Book', 'post type singular name', 'your-plugin-
-textdomain' ),
    'menu_name'          => _x( 'Books', 'admin menu', 'your-plugin-textdomain'
),
    'name_admin_bar'     => _x( 'Book', 'add new on admin bar', 'your-plugin-te
xtdomain' ),
    'add_new'            => _x( 'Add New', 'book', 'your-plugin-textdomain' ),
    'add_new_item'       => __( 'Add New Book', 'your-plugin-textdomain' ),
    'new_item'           => __( 'New Book', 'your-plugin-textdomain' ),
    'edit_item'          => __( 'Edit Book', 'your-plugin-textdomain' ),
    'view_item'          => __( 'View Book', 'your-plugin-textdomain' ),
    'all_items'           => __( 'All Books', 'your-plugin-textdomain' ),
    'search_items'       => __( 'Search Books', 'your-plugin-textdomain' ),
    'parent_item_colon'  => __( 'Parent Books:', 'your-plugin-textdomain' ),
    'not_found'          => __( 'No books found.', 'your-plugin-textdomain' ),
    'not_found_in_trash' => __( 'No books found in Trash.', 'your-plugin-textdo
main' )
    );

$args = array(
    'labels'              => $labels,
    'description'         => __( 'Description.', 'your-plugin-textdomain' ),
    'public'              => true,
    'publicly_queryable'  => true,
    'show_ui'             => true,
    'show_in_menu'        => true,
    'query_var'           => true,
    'rewrite'             => array( 'slug' => 'book' ),
    'capability_type'     => 'post',
    'has_archive'         => true,
    'hierarchical'        => false,
    'menu_position'       => null,
    'show_in_rest'        => true,
    'rest_base'           => 'books',
    'rest_controller_class' => 'WP_REST_Posts_Controller',
    'supports'            => array( 'title', 'editor', 'author', 'thumbnail', 'e
xcerpt', 'comments' )
    );

register_post_type( 'book', $args );
}

```

## 使用REST API支持注册自定义分类法

使用REST API支持注册自定义分类法与注册自定义帖子类型非常相似：在传递给register\_taxonomy的参数中传递'show\_in\_rest'=> true。您可以选择通过rest\_base更改分类法路由的基本URL。



分类法的默认控制器是WP\_REST\_Terms\_Controller。如果您选择使用自定义控制器，则可以使用rest\_controller\_class进行修改。

以下是使用REST API支持注册自定义分类法的示例：

```
/**
 * Register a genre post type, with REST API support
 *
 * Based on example at: https://codex.wordpress.org/Function_Reference/register_taxonomy
 */
add_action( 'init', 'my_book_taxonomy', 30 );
function my_book_taxonomy() {

    $labels = array(
        'name'          => _x( 'Genres', 'taxonomy general name' ),
        'singular_name'  => _x( 'Genre', 'taxonomy singular name' ),
        'search_items'   => __( 'Search Genres' ),
        'all_items'      => __( 'All Genres' ),
        'parent_item'    => __( 'Parent Genre' ),
        'parent_item_colon' => __( 'Parent Genre:' ),
        'edit_item'      => __( 'Edit Genre' ),
        'update_item'    => __( 'Update Genre' ),
        'add_new_item'   => __( 'Add New Genre' ),
        'new_item_name'  => __( 'New Genre Name' ),
        'menu_name'      => __( 'Genre' ),
    );

    $args = array(
        'hierarchical'   => true,
        'labels'          => $labels,
        'show_ui'        => true,
        'show_admin_column' => true,
        'query_var'      => true,
        'rewrite'        => array( 'slug' => 'genre' ),
        'show_in_rest'   => true,
        'rest_base'      => 'genre',
        'rest_controller_class' => 'WP_REST_Terms_Controller',
    );

    register_taxonomy( 'genre', array( 'book' ), $args );
}
```

## 向现有内容类型添加REST API支持

当您不能控制的代码（例如您正在使用的主题或插件）添加了自定义帖子类型或自定义分类法时，可能需要在已注册alredy之后添加REST API支持。参数与前面的例子相同，但需要添加到全局\$ wp\_post\_types和\$

wp\_taxonomies数组中。

以下是向现有自定义帖子类型添加REST API支持的示例：

```
/**
 * Add REST API support to an already registered post type.
 */
add_action( 'init', 'my_custom_post_type_rest_support', 25 );
function my_custom_post_type_rest_support() {
    global $wp_post_types;

    //be sure to set this to the name of your post type!
    $post_type_name = 'planet';
    if( isset( $wp_post_types[ $post_type_name ] ) ) {
        $wp_post_types[$post_type_name]->show_in_rest = true;
        // Optionally customize the rest_base or controller class
        $wp_post_types[$post_type_name]->rest_base = $post_type_name;
        $wp_post_types[$post_type_name]->rest_controller_class = 'WP_REST_Posts_Controller';
    }
}
```

以下是如何向已注册的自定义分类法添加REST API支持的示例。

```
/**
 * Add REST API support to an already registered taxonomy.
 */
add_action( 'init', 'my_custom_taxonomy_rest_support', 25 );
function my_custom_taxonomy_rest_support() {
    global $wp_taxonomies;

    //be sure to set this to the name of your taxonomy!
    $taxonomy_name = 'planet_class';

    if ( isset( $wp_taxonomies[ $taxonomy_name ] ) ) {
        $wp_taxonomies[ $taxonomy_name ]->show_in_rest = true;

        // Optionally customize the rest_base or controller class
        $wp_taxonomies[ $taxonomy_name ]->rest_base = $taxonomy_name;
        $wp_taxonomies[ $taxonomy_name ]->rest_controller_class = 'WP_REST_Terms_Controller';
    }
}
```

如果您在执行这些示例中遇到问题，请确保您正在添加具有足够高优先级的这些钩子。如果回调函数在发布类型或分类法被注册之前运行，则isset检查将阻止错误，但不会添加支持。

##自定义链接关系

分类和自定义帖子类型在WordPress内部有一个内置关联，但是如果要在两个自定义帖子类型之间建立链接呢？这在WordPress本身不正式支持，但是我们可以使用\_link关系在任意内容类型之间创建我们自己的连接

# 修改回应

WordPress REST API的默认端点设计为默认返回数据，为大多数站点和用例提供数据，但是往往需要扩展各种对象类型的响应。

为了适应这些需求，REST API被设计为高度可扩展的，就像其余的WordPress一样。本文档详细介绍了如何使用register\_rest\_field和register\_meta函数向默认端点的响应添加附加数据。您可以使用这些函数将字段添加到REST API支持的任何对象类型中。这些自定义字段可以支持get和update操作。

## ##关于改变回应的重要注意事项

API提供了许多关于API响应的领域，包括您可能不需要的内容，或者可能不适合您的网站的工作原理。虽然从响应中修改或删除字段很诱人，但这将导致API客户端出现预期标准响应的问题。这包括移动客户端或第三方工具，以帮助您管理您的网站，最终是wp-admin本身。

您可能只需要少量数据，但请务必记住，API是将界面暴露给所有客户端，而不仅仅是您正在开发的功能。变化的反应是危险的。

添加字段不是危险的，所以如果你需要修改数据，最好是用修改的数据来复制字段。不鼓励删除字段;如果您需要回收较小的数据子集，请使用上下文，并考虑使用自己的上下文。

请注意，API不能阻止您更改响应，但代码的结构强烈阻止此。在内部，现场注册由过滤器供电，如果绝对没有其他选择，则可以使用这些注册。

## 什么register\_rest\_field

在响应的基础架构中，全局变量\$ wp\_rest\_additional\_fields用于保存要添加到每个对象类型的响应的字段。

REST API提供了register\_rest\_field作为添加到这个全局变量的效用函数。应避免直接添加到全局变量，以确保最大化前向兼容性。

对于每个对象类型 - 帖子或用户，术语，注释等，\$ wp\_rest\_additional\_fields包含一个字段数组，每个字段可以具有用于检索该值的回调值，并使用添加该字段的任何端点更新该值可用于更新。

## 如何使用register\_rest\_field

函数register\_rest\_field字段接受三个参数：

\$ object\_type：对象的名称，作为字符串，或该字段正在注册的对象名称的数组。当添加到帖子类型端点时，应使用“post”。或者可以使用“条款”，“元”，“用户”或“注释”。

\$ attribute：字段的名称。该名称将用于定义响应对象中的键。

\$ args：具有键的数组，用于定义用于检索字段值的回调函数，以更新字段的值并定义其模式。每个这些键都是可选的，但如果不使用，则不会添加该功能。

这意味着如果您指定一个回调函数来读取该值，而不是一个用于更新的回调函数，那么它将是可读的，但不可更新。这在许多情况下可能是需要的。

应在rest\_api\_init操作中注册字段。使用此操作而不是init将阻止在不使用REST API的WordPress请求期间发生字段注册。

## 在评论回应中读写一个额外的字段

这个例子的TODO：

将匿名函数更改为命名函数（讨论）

权限检查如何工作？ 错误处理？（在什么条件下更新代码被调用，还有什么其他检查需要在现实世界插件中完成？）

开发手册中的代码块的更短的线条和/或样式修复

```
<?php
add_action( 'rest_api_init', function() {
    register_rest_field( 'comment', 'karma', array(
        'get_callback' => function( $comment_arr ) {
            $comment_obj = get_comment( $comment_arr['id'] );
            return (int) $comment_obj->comment_karma;
        },
        'update_callback' => function( $karma, $comment_obj ) {
            $ret = wp_update_comment( array(
                'comment_ID'      => $comment_obj->comment_ID,
                'comment_karma' => $karma
            ) );
            if ( false === $ret ) {
                return new WP_Error( 'rest_comment_karma_failed', __( 'Failed to update comment karma.' ), array( 'status' => 500 ) );
            }
            return true;
        },
        'schema' => array(
            'description' => __( 'Comment karma.' ),
            'type'        => 'integer'
        ),
    ) );
} );
```

此示例说明如何将一个称为业务的字段添加到帖子的响应中。它的作用是因为comment\_karma字段存在，但未被核心使用。请注意，实际执行评论业务需要使用单独的端点。

使用Meta&register\_meta

在帖子回复中读写一个后期元字段

```
<?php
// The object type. For custom post types, this is 'post';
// for custom comment types, this is 'comment'. For user meta,
```

```
// this is 'user'.
$object_type = 'post';
$args1 = array( // Validate and sanitize the meta value.
    // Note: currently (4.7) one of 'string', 'boolean', 'integer',
    // 'number' must be used as 'type'. The default is 'string'.
    'type' => 'string',
    // Shown in the schema for the meta key.
    'description' => 'A meta key associated with a string meta value.',
    // Return a single value of the type.
    'single' => true,
    // Show in the WP REST API response. Default: false.
    'show_in_rest' => true,
);
register_meta( $object_type, 'my_meta_key', $args1 );
```

此示例显示如何允许读取和写入后期元字段。这将允许通过POST请求将宇宙飞船字段更新为wp-json / wp / v2 / posts / ，或者通过POST请求发送到wp-json / wp / v2 / posts / 。

请注意，对于在自定义帖子类型上注册的元字段，帖子类型必须支持自定义字段。否则，元字段不会出现在REST API中。

# 模式

一个模式是元数据，告诉我们我们的数据结构如何。大多数数据库实现某种形式的模式，使我们能够以更结构化的方式推理我们的数据。WordPress REST API使用JSON模式来处理其数据的结构化。您可以在不使用架构的情况下实现端点，但是您会错过很多事情。你最好决定什么适合你。

## JSON Schema

首先让我们来谈谈JSON。JSON是类似于JavaScript对象的人类可读数据格式。JSON代表JavaScript对象表示法。JSON越来越受欢迎，似乎正在冒着数据结构的世界。WordPress REST API使用JSON的特殊规范称为JSON模式。要了解有关JSON模式的更多信息，请查看JSON Schema网站，这更容易理解JSON Schema的介绍。模式为我们带来了许多好处：改进的测试，可发现性和整体更好的结构。我们来看一下JSON的数据块。

```
{
  "shouldBeArray": 'LOL definitely not an array',
  "shouldBeInteger": ['lolz', 'you', 'need', 'schema'],
  "shouldBeString": 123456789
}
```

一个JSON解析器会经历这些数据没有问题，不会抱怨任何事情，因为它是有效的JSON。客户端和服务端什么也不知道数据，期望他们只看到JSON。通过实现模式，我们实际上可以简化我们的代码库。模式将有助于更好地构建我们的数据，因此我们的应用程序可以更容易地理解我们与WordPress REST API的交互。WordPress REST API不会强制您使用模式，但它是鼓励的。模式数据包含在API中有两种方式：我们注册参数的资源和模式的模式。

## 资源架构

资源的模式指示特定对象存在哪些字段。当我们注册我们的路由时，我们也可以指定路由的资源模式。我们来看看一个简单的注释模式在JSON模式的PHP表示中可能是什么样的。

```
// Register our routes.
function prefix_register_my_comment_route() {
    register_rest_route( 'my-namespace/v1', '/comments', array(
        // Notice how we are registering multiple endpoints the 'schema' equate
        // s to an OPTIONS request.
        array(
            'methods' => 'GET',
            'callback' => 'prefix_get_comment_sample',
        ),
        // Register our schema callback.
        'schema' => 'prefix_get_comment_schema',
    )
}
```

```

    ) );
}

add_action( 'rest_api_init', 'prefix_register_my_comment_route' );

/**
 * Grabs the five most recent comments and outputs them as a rest response.
 *
 * @param WP_REST_Request $request Current request.
 */
function prefix_get_comment_sample( $request ) {
    $args = array(
        'post_per_page' => 5,
    );
    $comments = get_comments( $args );

    $data = array();

    if ( empty( $comments ) ) {
        return rest_ensure_response( $data );
    }

    foreach ( $comments as $comment ) {
        $response = prefix_rest_prepare_comment( $comment, $request );
        $data[] = prefix_prepare_for_collection( $response );
    }

    // Return all of our comment response data.
    return rest_ensure_response( $data );
}

/**
 * Matches the comment data to the schema we want.
 *
 * @param WP_Comment $comment The comment object whose response is being prepared.
 */
function prefix_rest_prepare_comment( $comment, $request ) {
    $comment_data = array();

    $schema = prefix_get_comment_schema();

    // We are also renaming the fields to more understandable names.
    if ( isset( $schema['properties']['id'] ) ) {
        $comment_data['id'] = (int) $comment->comment_id;
    }

    if ( isset( $schema['properties']['author'] ) ) {
        $comment_data['author'] = (int) $comment->user_id;
    }
}

```



```

        if ( isset( $schema['properties']['content'] ) ) {
            $comment_data['content'] = apply_filters( 'comment_text', $comment->comment_content, $comment );
        }

        return rest_ensure_response( $comment_data );
    }

/**
 * Prepare a response for inserting into a collection of responses.
 *
 * This is copied from WP_REST_Controller class in the WP REST API v2 plugin.
 *
 * @param WP_REST_Response $response Response object.
 * @return array Response data, ready for insertion into collection data.
 */
function prefix_prepare_for_collection( $response ) {
    if ( ! ( $response instanceof WP_REST_Response ) ) {
        return $response;
    }

    $data = (array) $response->get_data();
    $server = rest_get_server();

    if ( method_exists( $server, 'get_compact_response_links' ) ) {
        $links = call_user_func( array( $server, 'get_compact_response_links' ), $response );
    } else {
        $links = call_user_func( array( $server, 'get_response_links' ), $response );
    }

    if ( ! empty( $links ) ) {
        $data['_links'] = $links;
    }

    return $data;
}

/**
 * Get our sample schema for comments.
 */
function prefix_get_comment_schema() {
    $schema = array(
        // This tells the spec of JSON Schema we are using which is draft 4.
        '$schema' => 'http://json-schema.org/draft-04/schema#',
        // The title property marks the identity of the resource.
        'title' => 'comment',
        'type' => 'object',
    );
}

```

```
// In JSON Schema you can specify object properties in the properties attribute.
    'properties' => array(
        'id' => array(
            'description' => esc_html__( 'Unique identifier for the object.', 'my-textdomain' ),
            'type' => 'integer',
            'context' => array( 'view', 'edit', 'embed' ),
            'readonly' => true,
        ),
        'author' => array(
            'description' => esc_html__( 'The id of the user object, if author was a user.', 'my-textdomain' ),
            'type' => 'integer',
        ),
        'content' => array(
            'description' => esc_html__( 'The content for the object.', 'my-textdomain' ),
            'type' => 'string',
        ),
    ),
);

return $schema;
}
```

如果您注意到，每个注释资源现在都符合我们指定的模式。我们将此开关置于 `prefix_rest_prepare_comment()` 中。通过为我们的资源创建模式，我们现在可以通过进行 `OPTIONS` 请求查看此模式。为什么这很有用？如果我们想要其他语言（例如 JavaScript）来解释我们的数据并验证来自我们的端点的数据，JavaScript 将需要知道我们的数据结构。当我们提供模式时，我们为其他作者和我们自己打开门，以一致的方式建立在我们的端点之上。

模式提供机器可读数据，所以潜在的任何可以读取 JSON 的内容都可以了解它正在查看什么样的数据。当我们通过向 `https://ourawesomesite.com/wp-json/` 发出 GET 请求来查看 API 索引时，我们返回了 API 的模式，使其他人能够编写客户端库来解释我们的数据。读取模式数据的这个过程被称为发现。当我们为资源提供架构时，我们可以通过 `OPTIONS` 请求发现该资源到该路由。公开资源模式只是我们模式难题的一部分。我们也想为我们注册的参数使用模式。

## 参数模式

当我们注册一个端点的请求参数时，我们还可以使用 JSON Schema 来提供有关参数应该是什么的数据。这使我们能够编写可以在我们的端点扩展时重用的验证库。模式是更多的工作，但是如果你要编写一个可以增长的生产应用程序，你应该考虑使用模式。我们来看一个使用参数模式和验证的例子。

```
// Register our routes.
function prefix_register_my_arg_route() {
    register_rest_route( 'my-namespace/v1', '/schema-arg', array(
        // Here we register our endpoint.
        array(
            'methods' => 'GET',
            'callback' => 'prefix_get_item',
            'args' => prefix_get_endpoint_args(),
        ),
    ) );
}

// Hook registration into 'rest_api_init' hook.
add_action( 'rest_api_init', 'prefix_register_my_arg_route' );

/**
 * Returns the request argument `my-arg` as a rest response.
 *
 * @param WP_REST_Request $request Current request.
 */
function prefix_get_item( $request ) {
    // If we didn't use required in the schema this would throw an error when m
    y arg is not set.
    return rest_ensure_response( $request['my-arg'] );
}

/**
 * Get the argument schema for this example endpoint.
 */
function prefix_get_endpoint_args() {
    $args = array();

    // Here we add our PHP representation of JSON Schema.
    $args['my-arg'] = array(
        'description' => esc_html__( 'This is the argument our endpoint r
        eturns.', 'my-textdomain' ),
        'type' => 'string',
        'validate_callback' => 'prefix_validate_my_arg',
        'sanitize_callback' => 'prefix_sanitize_my_arg',
        'required' => true,
    );

    return $args;
}

/**
 * Our validation callback for `my-arg` parameter.
 *
 * @param mixed $value Value of the my-arg parameter.

```

```

* @param WP_REST_Request $request Current request object.
* @param string          $param    The name of the parameter in this case, 'my-arg'.
*/
function prefix_validate_my_arg( $value, $request, $param ) {
    $attributes = $request->get_attributes();

    if ( isset( $attributes['args'][ $param ] ) ) {
        $argument = $attributes['args'][ $param ];
        // Check to make sure our argument is a string.
        if ( 'string' === $argument['type'] && ! is_string( $value ) ) {
            return new WP_Error( 'rest_invalid_param', sprintf( esc_html__( '%1$s is not of type %2$s', 'my-textdomain' ), $param, 'string' ), array( 'status' => 400 ) );
        }
    } else {
        // This code won't execute because we have specified this argument as required.
        // If we reused this validation callback and did not have required args then this would fire.
        return new WP_Error( 'rest_invalid_param', sprintf( esc_html__( '%s was not registered as a request argument.', 'my-textdomain' ), $param ), array( 'status' => 400 ) );
    }

    // If we got this far then the data is valid.
    return true;
}

/**
 * Our sanitization callback for `my-arg` parameter.
 *
 * @param mixed          $value    Value of the my-arg parameter.
 * @param WP_REST_Request $request Current request object.
 * @param string          $param    The name of the parameter in this case, 'my-arg'.
 */
function prefix_sanitize_my_arg( $value, $request, $param ) {
    $attributes = $request->get_attributes();

    if ( isset( $attributes['args'][ $param ] ) ) {
        $argument = $attributes['args'][ $param ];
        // Check to make sure our argument is a string.
        if ( 'string' === $argument['type'] ) {
            return sanitize_text_field( $value );
        }
    } else {
        // This code won't execute because we have specified this argument as required.
        // If we reused this validation callback and did not have required args

```

```
then this would fire.  
    return new WP_Error( 'rest_invalid_param', sprintf( esc_html__( '%s was  
not registered as a request argument.', 'my-textdomain' ), $param ), array( 's  
tatus' => 400 ) );  
}  
  
// If we got this far then something went wrong don't use user input.  
return new WP_Error( 'rest_api_sad', esc_html__( 'Something went terribly w  
rong.', 'my-textdomain' ), array( 'status' => 500 ) );  
}
```

在上面的例子中，我们已经使用'my-arg'名称进行了抽象。我们可以为任何其他参数使用这些验证和清理函数，这些参数应该是我们指定的模式的字符串。随着您的代码库和端点的增长，架构将有助于保持代码轻量级和可维护性。没有模式，你可以验证和消毒，但是更难以跟踪哪些功能应该验证什么。通过向请求参数添加模式，我们还可以将我们的参数模式公开给客户端，因此验证库可以被构建为客户端，可以通过防止无效请求被发送到API来帮助执行性能。

注意：如果您对使用模式感到不舒服，仍然可以对每个参数进行验证/消毒回调，在某些情况下，做出自定义验证将是最有意义的。

## 概述

模式在点上可能看起来很愚蠢，可能是不必要的工作，但是如果您想要可维护，可发现和易于扩展的端点，则必须使用模式。模式还有助于自我记录人类和计算机的端点！

# 词汇表

---

最新的REST API ? 加快我们文档中使用的短语速度。

## 控制器

---

模型 - 视图 - 控制器是软件开发中的标准模式。如果你还不熟悉，你应该做一些阅读来加快速度。在WP-API中，我们采用了控制器概念，为代表资源端点的类提供了标准模式。所有资源端点扩展WP\_REST\_Controller以确保它们实现常用方法。

## HEAD，GET，POST，PUT和DELETE请求

---

这些“HTTP动词”表示HTTP客户端可能对资源执行的操作的类型。例如，GET请求用于获取帖子的数据，而DELETE请求用于删除帖子。它们被统称为“HTTP动词”，因为它们在网络上标准化的。

如果您熟悉WordPress函数，则GET请求与wp\_remote\_get（）相当，POST请求与wp\_remote\_post（）相同。

### HTTP客户端

短语“HTTP Client”是指用于与WP-API进行交互的工具。您可以使用Postman（Chrome）或REST Easy（Firefox）来测试浏览器中的请求，或者使用httpie在命令行中测试请求。

WordPress本身在WP\_HTTP类和相关函数（例如wp\_remote\_get（））中提供一个HTTP客户端。这可以用于从另一个访问一个WordPress站点。

## Resource

---

“资源”是WordPress中的离散实体。您可以将这些资源作为帖子，页面，评论，用户，条款等知道。WP-API允许HTTP客户端对资源执行CRUD操作（CRUD表示创建，读取，更新和删除）。

务实地，以下是通常与WP-API资源进行交互的方式：

- GET /wp-json/wp/v2/posts 收集帖子。这大致相当于使用WP\_Query。
- GET /wp-json/wp/v2/posts/123 获得ID为123的单个帖子。这大致相当于使用get\_post（）。
- POST /wp-json/wp/v2/posts 创建一个新的帖子。这大致相当于使用wp\_insert\_post（）。
- DELETE /wp-json/wp/v2/posts/123 删除具有ID 123的帖子。这大致相当于wp\_delete\_post（）。

## Routes / Endpoints

---

端点是通过API可用的功能。这可以是检索API索引，更新帖子或删除注释。端点执行一个特定的功能，采取一些参数和返回数据到客户端。

路由是您用于访问端点的“名称”，用于URL中。一个路由可以有多个端点与它相关联，哪个使用取决于HTTP动词。

例如，使用URL <http://example.com/wp-json/wp/v2/posts/123>：

“route” 是wp / v2 / posts / 123 - 路由不包括wp-json，因为wp-json是API本身的基本路径。

该路由有3个端点：

- GET 触发get\_item方法，将后期数据返回给客户端。
- PUT 触发update\_item方法，将数据更新，并返回更新的帖子数据。
- DELETE 触发delete\_item方法，将现在删除的帖子数据返回给客户端。

注意：在没有很多固定链接的站点上，路由替代作为rest\_route参数添加到URL中。对于上述示例，完整的网址将是[http://example.com/?rest\\_route=wp/v2/posts/123](http://example.com/?rest_route=wp/v2/posts/123)

## Schema

“模式” 是WP-API的响应数据的格式的表示。例如，Post架构通信请求获取Post将返回id，title，content，author和其他字段。我们的模式还指出每个字段的类型，提供人类可读的描述，并显示该字段将返回哪些上下文。

# 路由和端点

REST API为我们提供了一种将URI匹配到我们的WordPress安装中的各种资源的方法。默认情况下，如果您启用了漂亮的永久链接，则WordPress REST API “生活在 / wp-json /”。在我们的WordPress网站 <https://ourawesomesite.com> 上，我们可以通过向 <https://ourawesomesite.com/wp-json/> 发出GET请求来访问REST API的索引。该索引提供有关可用于特定WordPress安装的路由的信息，以及支持哪些HTTP方法以及注册了哪些端点。

## 路线与端点

端点是通过API可用的功能。这可以是检索API索引，更新帖子或删除注释。端点执行一个特定的功能，采取一些参数和返回数据到客户端。

路由是您用于访问端点的“名称”，用于URL中。一个路由可以有多个端点与它相关联，哪个使用取决于HTTP动词。

例如，使用网址 <http://example.com/wp-json/wp/v2/posts/123>：

“route” 是 wp / v2 / posts / 123 - 路由不包括 wp-json，因为 wp-json 是 API 本身的基本路径。

该路由有3个端点：

- GET触发一个get\_item方法，将post数据返回给客户端。
- PUT触发update\_item方法，使数据更新，并返回更新后的数据。
- DELETE触发一个delete\_item方法，将现在删除的帖子数据返回给客户端。

**警报：**在没有很好的固定链接的站点上，路由替代地添加到URL作为rest\_route参数。对于上述示例，完整的网址将是 [http://example.com/?rest\\_route=/wp/v2/posts/123](http://example.com/?rest_route=/wp/v2/posts/123)

## 创建端点

如果我们想要创建一个返回短语“Hello World”的端点，当它接收到一个GET请求时，我们首先需要注册该端点的路由。要注册路由，您应该使用register\_rest\_route（）函数。它需要在rest\_api\_init操作钩子上调用。register\_rest\_route（）处理到端点的路由的所有映射。我们尝试创建一个“Hello World，这是WordPress REST API”路由。

```
/**
 * This is our callback function that embeds our phrase in a WP_REST_Response
 */
function prefix_get_endpoint_phrase() {
    // rest_ensure_response() wraps the data we want to return into a WP_REST_Response, and ensures it will be properly returned.
    return rest_ensure_response( 'Hello World, this is the WordPress REST API' );
};
```



```

}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_example_routes() {
    // register_rest_route() handles more arguments but we are going to stick to the basics for now.
    register_rest_route( 'hello-world/v1', '/phrase', array(
        // By using this constant we ensure that when the WP_REST_Server changes our readable endpoints will work as intended.
        'methods' => WP_REST_Server::READABLE,
        // Here we register our callback. The callback is fired when this endpoint is matched by the WP_REST_Server class.
        'callback' => 'prefix_get_endpoint_phrase',
    ) );
}

add_action( 'rest_api_init', 'prefix_register_example_routes' );

```

传入`register_rest_route()`的第一个参数是命名空间，它为我们提供了一种分组路由的方法。传入的第二个参数是资源路径或资源库。对于我们的示例，我们检索的资源是“Hello World，这是WordPress REST API”短语。第三个参数是一组选项。我们指定端点可以使用什么方法以及当端点匹配时应该发生什么回调（可以做更多的事情，但这些都是基本原理）。

第三个参数还允许我们提供权限回调，这可以限制端点对特定用户的访问。第三个参数还提供了一种注册端点参数的方法，以便请求可以修改端点的响应。我们将在本指南的端点部分中介绍这些概念。

当我们转到`https://ourawesomesite.com/wp-json/hello-world/v1/phrase`时，我们现在可以看到我们的REST API，我们亲切地问候了。让我们来看一下更多的路线。

## ##路线

REST API中的路由由URI表示。路由本身就是在`https://ourawesomesite.com/wp-json`的末尾。API的索引路径是`/`，这就是为什么`https://ourawesomesite.com/wp-json`返回API的所有可用信息的原因。所有路线都应该建在这条路线上，`wp-json`部分可以改变，但是一般来说，建议保持一致。

我们想确保我们的路线是独一无二的。例如，我们可以有一个这样的书籍路线：`/books`。我们的书籍路线现在将存在于`https://ourawesomesite.com/wp-json/books`。然而，这不是一个好的做法，因为我们最终会污染API的潜在路线。如果另一个插件我们也想注册一本书籍路线怎么办？在这种情况下，我们会遇到很大的麻烦，因为这两条路线会相互冲突，只能使用一条路线。`register_rest_field()`的第四个参数是一个布尔值，用于路由是否应该覆盖现有路由。

覆盖参数也不能真正解决我们的问题，因为两个路由都可以覆盖，或者我们想要使用两个路由来做不同的事情。这是我们的路线使用命名空间的地方。

## ##命名空间

添加命名空间到您的路由是非常重要的。“核心”端点使用wp / v2命名空间。

**警告：**不要将任何东西放入wp命名空间中，除非您正在将端点合并成核心。

在核心端点命名空间中有一些关键的事情要注意。命名空间的第一部分是wp，它代表供应商名称; WordPress。对于我们的插件，我们将想要提供我们称为命名空间的供应商部分的唯一名称。在上面的例子中，我们使用了hello-world。

跟随供应商部分是命名空间的版本部分。“核心”端点使用v2来表示WordPress REST API的版本2。如果您正在编写一个插件，您可以通过简单地创建新的端点并增加您提供的版本号来保持REST API端点的向后兼容性。这样可以访问原始v1和v2端点。

命名空间后面的路由部分是资源路径。

## 资源路径

资源路径应该表示端点与哪个资源相关联。在上面我们使用的例子中，我们使用单词来表示我们正在交互的资源是一个短语。为了避免任何冲突，我们注册的每个资源路径在命名空间中也应该是唯一的。资源路径应用于定义给定命名空间内的不同资源路由。

假设我们有一个插件来处理一些基本的电子商务功能。我们将有两种主要的资源类型订单和产品。订单是对产品的请求，但它们不是产品本身。同样的概念适用于产品。虽然这些资源是相关的，但它们并不一样，每个资源都应该存在于单独的资源路径中。我们的路线最终会看到我们的电子商务插件：/ my-shop / v1 / orders和/ my-shop / v1 / products。

使用这样的路线，我们希望每个都返回一个订单或产品的集合。如果我们想通过ID获取特定的产品，我们需要在路由中使用路径变量。

## 路径变量

路径变量使我们能够添加动态路由。为了扩展我们的电子商务路线，我们可以注册一条路线来抢购个别产品。

```
/**
 * This is our callback function to return our products.
 *
 * @param WP_REST_Request $request This function accepts a rest request to process data.
 */
function prefix_get_products( $request ) {
    // In practice this function would fetch the desired data. Here we are just making stuff up.
    $products = array(
        '1' => 'I am product 1',
        '2' => 'I am product 2',
        '3' => 'I am product 3',
    );
}
```

```

    return rest_ensure_response( $products );
}

/**
 * This is our callback function to return a single product.
 *
 * @param WP_REST_Request $request This function accepts a rest request to process data.
 */
function prefix_get_product( $request ) {
    // In practice this function would fetch the desired data. Here we are just making stuff up.
    $products = array(
        '1' => 'I am product 1',
        '2' => 'I am product 2',
        '3' => 'I am product 3',
    );

    // Here we are grabbing the 'id' path variable from the $request object. WP_REST_Request implements ArrayAccess, which allows us to grab properties as though it is an array.
    $id = (string) $request['id'];

    if ( isset( $products[ $id ] ) ) {
        // Grab the product.
        $product = $products[ $id ];

        // Return the product as a response.
        return rest_ensure_response( $product );
    } else {
        // Return a WP_Error because the request product was not found. In this case we return a 404 because the main resource was not found.
        return new WP_Error( 'rest_product_invalid', esc_html__( 'The product does not exist.', 'my-text-domain' ), array( 'status' => 404 ) );
    }

    // If the code somehow executes to here something bad happened return a 500.

    return new WP_Error( 'rest_api_sad', esc_html__( 'Something went horribly wrong.', 'my-text-domain' ), array( 'status' => 500 ) );
}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_product_routes() {
    // Here we are registering our route for a collection of products.
    register_rest_route( 'my-shop/v1', '/products', array(
        // By using this constant we ensure that when the WP_REST_Server change

```

```

s our readable endpoints will work as intended.
    'methods' => WP_REST_Server::READABLE,
    // Here we register our callback. The callback is fired when this endpo
int is matched by the WP_REST_Server class.
    'callback' => 'prefix_get_products',
) );
// Here we are registering our route for single products. The (?P<id>[\d]+)
>[\d]+) is our path variable for the ID, which, in this example, can only be so
me form of positive number.
    register_rest_route( 'my-shop/v1', '/products/(?P<id>[\d]+)', array(
        // By using this constant we ensure that when the WP_REST_Server change
s our readable endpoints will work as intended.
        'methods' => WP_REST_Server::READABLE,
        // Here we register our callback. The callback is fired when this endpo
int is matched by the WP_REST_Server class.
        'callback' => 'prefix_get_product',
    ) );
}

add_action( 'rest_api_init', 'prefix_register_product_routes' );

```

上面的例子很多。要注意的重要的一点是，在第二条路线中，我们注册，我们添加一个路径变量/( ? P \ d ] + ) 到我们的资源路径/产品。路径变量是一个正则表达式。在这种情况下，它使用[\d]+表示应该是任何数字字符至少一次。如果您使用资源的数字ID，那么这是一个很好的例子，说明如何使用路径变量。当使用路径变量时，我们现在必须小心，因为用户输入可以匹配什么。

幸运的是，正则表达式将过滤出不是数字的任何东西。但是，如果请求的ID的产品不存在，该怎么办？我们需要做错误处理。您可以在上面的代码示例中看到我们处理错误的基本方法。当您在终端回调中返回WP\_Error时，API服务器将自动处理向客户端提供错误。

虽然本节是关于路线，但是我们已经对端点进行了很多的介绍。端点和路由是相互关联的，但它们绝对有区别。

## ##端点

端点是路由需要映射到的目的地址。对于任何给定的路由，您可以为其注册多个不同的端点。我们将扩展我们的虚拟电子商务插件，以更好地显示路由和端点之间的区别。我们将在/wp-json/my-shop/v1/products/route中创建两个端点。一个端点使用HTTP动词GET获取产品，另一个端点使用HTTP动词POST创建新产品。

```

/**
 * This is our callback function to return our products.
 *
 * @param WP_REST_Request $request This function accepts a rest request to proc
ess data.
 */
function prefix_get_products( $request ) {
    // In practice this function would fetch the desired data. Here we are just
making stuff up.

```

```

    $products = array(
        '1' => 'I am product 1',
        '2' => 'I am product 2',
        '3' => 'I am product 3',
    );

    return rest_ensure_response( $products );
}

/**
 * This is our callback function to return a single product.
 *
 * @param WP_REST_Request $request This function accepts a rest request to process data.
 */
function prefix_create_product( $request ) {
    // In practice this function would create a product. Here we are just making stuff up.
    return rest_ensure_response( 'Product has been created' );
}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_product_routes() {
    // Here we are registering our route for a collection of products and creation of products.
    register_rest_route( 'my-shop/v1', '/products', array(
        array(
            // By using this constant we ensure that when the WP_REST_Server changes, our readable endpoints will work as intended.
            'methods' => WP_REST_Server::READABLE,
            // Here we register our callback. The callback is fired when this endpoint is matched by the WP_REST_Server class.
            'callback' => 'prefix_get_products',
        ),
        array(
            // By using this constant we ensure that when the WP_REST_Server changes, our create endpoints will work as intended.
            'methods' => WP_REST_Server::CREATABLE,
            // Here we register our callback. The callback is fired when this endpoint is matched by the WP_REST_Server class.
            'callback' => 'prefix_create_product',
        ),
    ) );
}

add_action( 'rest_api_init', 'prefix_register_product_routes' );

```

根据我们用于路由 `/wp-json/my-shop/v1/products` 的 HTTP 方法，我们与不同的端点匹配，并且触发不同的回调。当我们使用 POST 时，我们触发 `prefix_create_product()` 回调，当我们使用 GET 时，我们触发 `prefix_get_products()` 回调。

有许多不同的 HTTP 方法，REST API 可以使用它们中的任何一种。

## HTTP 方法

HTTP 方法有时被称为 HTTP 动词。它们只是通过 HTTP 进行通信的不同方式。WordPress REST API 使用的主要功能有：

GET 应用于从 API 检索数据。

POST 应用于创建新资源（即用户，帖子，分类）。

PUT 应用于更新资源。

DELETE 应用于删除资源。

应该使用选项来提供有关我们资源的上下文。

重要的是要注意，每个客户端都不支持这些方法，因为它们在 HTTP 1.1 中引入。幸运的是，API 为这些不幸的情况提供了解决方法。如果要删除资源但无法发送 DELETE 请求，则可以在请求中使用 `_method` 参数或 `X-HTTP-Method-Override` 标头。您如何发送 POST 请求到 `https://ourawesomesite.com/wp-json/my-shop/v1/products/1?_method=DELETE`。现在您将删除产品编号 1，即使您的客户端无法在请求中发送适当的 HTTP 方法，也可能存在阻止 DELETE 请求的防火墙。

HTTP 方法与路由和回调相结合，构成端点的核心。

## 回调

REST API 支持的端点目前只有两种类型的回调方式：回调和 `permission_callback`。主要回调应该处理与资源的交互。权限回调应该处理用户对端点的访问权限。您可以通过在注册端点时添加其他信息来添加额外的回调。然后，您可以钩入 `rest_pre_dispatch`，`rest_dispatch_request` 或 `rest_post_dispatch` 钩子来触发新的自定义回调。

## 端点回调

删除端点的主要回调只能删除资源并在响应中返回它的副本。创建端点的主要回调只能创建资源并返回与新创建的数据匹配的响应。更新回调应该仅修改实际存在的资源。读取回调应该只检索已经存在的数据。考虑到幂等概念是很重要的。

在 REST API 的上下文中，Idempotence 意味着如果向端点发出相同的请求，服务器将以相同的方式处理请求。想像一下，如果我们读取的端点不是幂等的。每当我们向它提出请求时，我们的服务器状态将被请求修改，即使我们只是试图获取数据。这可能是灾难性的。任何时候有人从您的服务器中获取数据，内部会发生变化。重要的是确保读取，更新和删除端点不具有令人讨厌的副作用，并坚持要做的事情。

在 REST API 中，幂等式的概念与 HTTP 方法而不是端点回调相关联。使用 GET，HEAD，TRACE，OPTIONS，PUT 或 DELETE 的任何回调不应产生任何副作用。POST 请求不是幂等的，通常用于创建资源。如果你创建了一个

幂等的创建方法，那么您只会创建一个资源，因为当您做同样的请求时，服务器不会有更多的副作用。对于创建，如果您通过服务器发出相同的请求，则每次都应生成新的资源。

为了限制端点的使用，我们需要注册一个权限回调。

## 权限回调

权限回调对于使用WordPress REST API的安全性至关重要。如果您有任何不应公开显示的私人数据，那么您需要为您的端点注册权限回调。以下是如何注册权限回调的示例。

```
/**
 * This is our callback function that embeds our resource in a WP_REST_Response
 */
function prefix_get_private_data() {
    // rest_ensure_response() wraps the data we want to return into a WP_REST_Response,
    // and ensures it will be properly returned.
    return rest_ensure_response( 'This is private data.' );
}

/**
 * This is our callback function that embeds our resource in a WP_REST_Response
 */
function prefix_get_private_data_permissions_check() {
    // Restrict endpoint to only users who have the edit_posts capability.
    if ( ! current_user_can( 'edit_posts' ) ) {
        return new WP_Error( 'rest_forbidden', esc_html__( 'OMG you can not view private data.', 'my-text-domain' ), array( 'status' => 401 ) );
    }

    // This is a black-listing approach. You could alternatively do this via white-listing,
    // by returning false here and changing the permissions check.
    return true;
}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_example_routes() {
    // register_rest_route() handles more arguments but we are going to stick to the basics for now.
    register_rest_route( 'my-plugin/v1', '/private-data', array(
        // By using this constant we ensure that when the WP_REST_Server changes our readable endpoints will work as intended.
        'methods' => WP_REST_Server::READABLE,
        // Here we register our callback. The callback is fired when this endpoint is matched by the WP_REST_Server class.
        'callback' => 'prefix_get_private_data',
        // Here we register our permissions callback. The callback is fired before the main callback to check if the current user can access the endpoint.
    ) );
}
```



```

        'permission_callback' => 'prefix_get_private_data_permissions_check',
    ) );
}

add_action( 'rest_api_init', 'prefix_register_example_routes' );

```

如果您尝试此端点，而不启用任何身份验证，那么您也将返回错误响应，从而阻止您看到数据。身份验证是一个很大的课题，最终将会创建本章的一部分，向您展示如何创建自己的身份验证过程。

## 参数

当向端点发出请求时，您可能需要指定额外的参数来更改响应。可以在注册端点时添加这些额外的参数。我们来看一个如何使用一个端点参数的例子。

```

/**
 * This is our callback function that embeds our resource in a WP_REST_Response
 */
function prefix_get_colors( $request ) {
    // In practice this function would fetch the desired data. Here we are just
    making stuff up.
    $colors = array(
        'blue',
        'blue',
        'red',
        'red',
        'green',
        'green',
    );

    if ( isset( $request['filter'] ) ) {
        $filtered_colors = array();
        foreach ( $colors as $color ) {
            if ( $request['filter'] === $color ) {
                $filtered_colors[] = $color;
            }
        }
        return rest_ensure_response( $filtered_colors );
    }
    return rest_ensure_response( $colors );
}

/**
 * We can use this function to contain our arguments for the example product endpoint.
 */
function prefix_get_color_arguments() {
    $args = array();
    // Here we are registering the schema for the filter argument.

```



```

    $args['filter'] = array(
        // description should be a human readable description of the argument.
        'description' => esc_html__( 'The filter parameter is used to filter the collection of colors', 'my-text-domain' ),
        // type specifies the type of data that the argument should be.
        'type' => 'string',
        // enum specified what values filter can take on.
        'enum' => array( 'red', 'green', 'blue' ),
    );
    return $args;
}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_example_routes() {
    // register_rest_route() handles more arguments but we are going to stick to the basics for now.
    register_rest_route( 'my-colors/v1', '/colors', array(
        // By using this constant we ensure that when the WP_REST_Server changes our readable endpoints will work as intended.
        'methods' => WP_REST_Server::READABLE,
        // Here we register our callback. The callback is fired when this endpoint is matched by the WP_REST_Server class.
        'callback' => 'prefix_get_colors',
        // Here we register our permissions callback. The callback is fired before the main callback to check if the current user can access the endpoint.
        'args' => prefix_get_color_arguments(),
    ) );
}

add_action( 'rest_api_init', 'prefix_register_example_routes' );

```

我们现在已经为此示例指定了一个过滤器参数。当我们请求端点时，我们可以将参数指定为查询参数。如果我们向 `https://ourawesomesitem.com/my-colors/v1/colors?filter=blue` 发出 GET 请求，我们将只收回我们的集合中的蓝色。您也可以将它们作为身体参数传递到请求正文中，而不是在查询字符串中。要了解查询参数和体参数之间的区别，您应该阅读有关 HTTP 规范的内容。查询参数实际上位于查询字符串中的 URL 和 body 参数直接嵌入到 HTTP 请求的正文中。

我们已经为我们的端点创建了一个参数，但是我们如何验证参数是一个字符串，并判断它是否匹配红色，绿色或蓝色的值。为此，我们需要为我们的参数指定一个验证回调。

## 验证

验证和消毒对 API 的安全性至关重要。验证回调（在 WP 4.6+ 中），在消毒回调之前触发。您应该使用 `validate_callback` 作为参数来验证您正在接收的输入是否有效。在参数由主回调处理之前，应该使用

`sanitize_callback`来转换参数输入或清除参数中的不需要的部分。

在上面的示例中，我们需要验证过滤器参数是一个字符串，并且它与红色，绿色或蓝色的值匹配。让我们看一下在`validate_callback`中添加代码之后的代码。

```
/**
 * This is our callback function that embeds our resource in a WP_REST_Response
 */
function prefix_get_colors( $request ) {
    // In practice this function would fetch more practical data. Here we are j
    ust making stuff up.
    $colors = array(
        'blue',
        'blue',
        'red',
        'red',
        'green',
        'green',
    );

    if ( isset( $request['filter'] ) ) {
        $filtered_colors = array();
        foreach ( $colors as $color ) {
            if ( $request['filter'] === $color ) {
                $filtered_colors[] = $color;
            }
        }
        return rest_ensure_response( $filtered_colors );
    }
    return rest_ensure_response( $colors );
}

/**
 * Validate a request argument based on details registered to the route.
 *
 * @param mixed      $value      Value of the 'filter' argument.
 * @param WP_REST_Request $request The current request object.
 * @param string      $param      Key of the parameter. In this case it is '
filter'.
 * @return WP_Error|boolean
 */
function prefix_filter_arg_validate_callback( $value, $request, $param ) {
    // If the 'filter' argument is not a string return an error.
    if ( ! is_string( $value ) ) {
        return new WP_Error( 'rest_invalid_param', esc_html__( 'The filter argu
ment must be a string.', 'my-text-domain' ), array( 'status' => 400 ) );
    }

    // Get the registered attributes for this endpoint request.
    $attributes = $request->get_attributes();
}
```

```

// Grab the filter param schema.
$args = $attributes['args'][$param];

// If the filter param is not a value in our enum then we should return an
error as well.
if ( ! in_array( $value, $args['enum'], true ) ) {
    return new WP_Error( 'rest_invalid_param', sprintf( __( '%s is not one
of %s' ), $param, implode( ', ', $args['enum'] ) ), array( 'status' => 400 ) );
}
}

/**
 * We can use this function to contain our arguments for the example product en
dpoint.
 */
function prefix_get_color_arguments() {
    $args = array();
    // Here we are registering the schema for the filter argument.
    $args['filter'] = array(
        // description should be a human readable description of the argument.
        'description' => esc_html__( 'The filter parameter is used to filter th
e collection of colors', 'my-text-domain' ),
        // type specifies the type of data that the argument should be.
        'type' => 'string',
        // enum specified what values filter can take on.
        'enum' => array( 'red', 'green', 'blue' ),
        // Here we register the validation callback for the filter argument.
        'validate_callback' => 'prefix_filter_arg_validate_callback',
    );
    return $args;
}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_example_routes() {
    // register_rest_route() handles more arguments but we are going to stick t
o the basics for now.
    register_rest_route( 'my-colors/v1', '/colors', array(
        // By using this constant we ensure that when the WP_REST_Server change
s our readable endpoints will work as intended.
        'methods' => WP_REST_Server::READABLE,
        // Here we register our callback. The callback is fired when this endpo
int is matched by the WP_REST_Server class.
        'callback' => 'prefix_get_colors',
        // Here we register our permissions callback. The callback is fired bef
ore the main callback to check if the current user can access the endpoint.
        'args' => prefix_get_color_arguments(),
    ) );
}

```

```
}

add_action( 'rest_api_init', 'prefix_register_example_routes' );
```

## 消毒

在上面的例子中，我们不需要使用`sanitize_callback`，因为我们将输入限制在我们枚举中的值。如果我们没有严格的验证并接受任何字符串作为参数，我们一定需要注册一个`sanitize_callback`。如果我们要更新内容字段，用户输入的内容如“alert”（“ZOMG Hacking you”），该怎么办？字段值可能是可执行脚本。要删除不需要的数据或将数据转换成所需的格式，我们需要为我们的参数注册一个`sanitize_callback`。以下是使用WordPress的`sanitize_text_field()`进行消毒回调的示例：

```
/**
 * This is our callback function that embeds our resource in a WP_REST_Response
 *
 * The parameter is already sanitized by this point so we can use it without any worries.
 */
function prefix_get_item( $request ) {
    if ( isset( $request['data'] ) ) {
        return rest_ensure_response( $request['data'] );
    }

    return new WP_Error( 'rest_invalid', esc_html__( 'The data parameter is required.', 'my-text-domain' ), array( 'status' => 400 ) );
}

/**
 * Validate a request argument based on details registered to the route.
 *
 * @param mixed $value Value of the 'filter' argument.
 * @param WP_REST_Request $request The current request object.
 * @param string $param Key of the parameter. In this case it is 'filter'.
 * @return WP_Error|boolean
 */
function prefix_data_arg_validate_callback( $value, $request, $param ) {
    // If the 'data' argument is not a string return an error.
    if ( ! is_string( $value ) ) {
        return new WP_Error( 'rest_invalid_param', esc_html__( 'The filter argument must be a string.', 'my-text-domain' ), array( 'status' => 400 ) );
    }
}

/**
 * Sanitize a request argument based on details registered to the route.
```

```

*
* @param mixed      $value    Value of the 'filter' argument.
* @param WP_REST_Request $request The current request object.
* @param string      $param    Key of the parameter. In this case it is '
filter'.
* @return WP_Error|boolean
*/
function prefix_data_arg_sanitize_callback( $value, $request, $param ) {
    // It is as simple as returning the sanitized value.
    return sanitize_text_field( $value );
}

/**
 * We can use this function to contain our arguments for the example product en
dpoint.
 */
function prefix_get_data_arguments() {
    $args = array();
    // Here we are registering the schema for the filter argument.
    $args['data'] = array(
        // description should be a human readable description of the argument.
        'description' => esc_html__( 'The data parameter is used to be sanitize
d and returned in the response.', 'my-text-domain' ),
        // type specifies the type of data that the argument should be.
        'type'        => 'string',
        // Set the argument to be required for the endpoint.
        'required'    => true,
        // We are registering a basic validation callback for the data argument.

        'validate_callback' => 'prefix_data_arg_validate_callback',
        // Here we register the validation callback for the filter argument.
        'sanitize_callback' => 'prefix_data_arg_sanitize_callback',
    );
    return $args;
}

/**
 * This function is where we register our routes for our example endpoint.
 */
function prefix_register_example_routes() {
    // register_rest_route() handles more arguments but we are going to stick t
o the basics for now.
    register_rest_route( 'my-plugin/v1', '/sanitized-data', array(
        // By using this constant we ensure that when the WP_REST_Server change
s our readable endpoints will work as intended.
        'methods' => WP_REST_Server::READABLE,
        // Here we register our callback. The callback is fired when this endpo
int is matched by the WP_REST_Server class.
        'callback' => 'prefix_get_item',
        // Here we register our permissions callback. The callback is fired bef

```

```
ore the main callback to check if the current user can access the endpoint.  
    'args' => prefix_get_data_arguments(),  
  ) );  
}  
  
add_action( 'rest_api_init', 'prefix_register_example_routes' );
```

## 概要

我们已经介绍了为WordPress REST API注册端点的基础知识。路由是我们的端点所在的URI。端点是回调，方法，参数和其他选项的集合。当使用`register_rest_route()`时，每个端点映射到路由。默认端点可以支持各种HTTP方法，主要回调，权限回调和注册参数。我们可以注册端点来覆盖与WordPress交互的任何用例。端点作为与REST API的核心交互点，但还有许多其他要探索和理解的主题，以充分利用这个强大的API。

# 控制器类

编写端点时，使用控制器类来处理端点的功能是有帮助的。控制器类将提供与API进行交互的标准方式，也是与API进行交互的更可维护的方式。WordPress目前的最低PHP版本为5.2，如果您正在开发将由WordPress生态系统使用的端点，您应该考虑支持WordPress的最低要求。

PHP 5.2没有内置命名空间。这意味着您声明的每个函数都将在全局范围内。如果您决定使用`get_items()`等端点的常用函数名称，另一个插件也会注册该函数，则PHP将失败并导致致命错误。这是因为函数`get_items()`被声明为两次。通过封装我们的端点，我们可以避免这些命名冲突，并具有与API进行交互的一致方式。

## ##控制器

控制器通常做一件事情：他们接收输入并产生输出。对于WordPress REST API，我们的控制器将处理作为`WP_REST_Request`对象的请求输入，并将响应输出生成为`WP_REST_Response`对象。我们来看一个例子的控制器类：

```
class My_REST_Posts_Controller {

    // Here initialize our namespace and resource name.
    public function __construct() {
        $this->namespace      = '/my-namespace/v1';
        $this->resource_name = 'posts';
    }

    // Register our routes.
    public function register_routes() {
        register_rest_route( $this->namespace, '/' . $this->resource_name, array(
            // Here we register the readable endpoint for collections.
            array(
                'methods'   => 'GET',
                'callback'  => array( $this, 'get_items' ),
                'permission_callback' => array( $this, 'get_items_permissions_check' ),
            ),
            // Register our schema callback.
            array(
                'schema' => array( $this, 'get_item_schema' ),
            )
        ));
        register_rest_route( $this->namespace, '/' . $this->resource_name . '/(?P<id>[\d]+)', array(
            // Notice how we are registering multiple endpoints the 'schema' equates to an OPTIONS request.
            array(
                'methods'   => 'GET',
                'callback'  => array( $this, 'get_item' ),
                'permission_callback' => array( $this, 'get_item_permissions_check' ),
            ),
        ));
    }
}
```

```

        ),
        // Register our schema callback.
        'schema' => array( $this, 'get_item_schema' ),
    ) );
}

/**
 * Check permissions for the posts.
 *
 * @param WP_REST_Request $request Current request.
 */
public function get_items_permissions_check( $request ) {
    if ( ! current_user_can( 'read' ) ) {
        return new WP_Error( 'rest_forbidden', esc_html__( 'You cannot view the post resource.' ), array( 'status' => $this->authorization_status_code() ) );
    }
    return true;
}

/**
 * Grabs the five most recent posts and outputs them as a rest response.
 *
 * @param WP_REST_Request $request Current request.
 */
public function get_items( $request ) {
    $args = array(
        'post_per_page' => 5,
    );
    $posts = get_posts( $args );

    $data = array();

    if ( empty( $posts ) ) {
        return rest_ensure_response( $data );
    }

    foreach ( $posts as $post ) {
        $response = $this->prepare_item_for_response( $post, $request );
        $data[] = $this->prepare_response_for_collection( $response );
    }

    // Return all of our comment response data.
    return rest_ensure_response( $data );
}

/**
 * Check permissions for the posts.
 *
 * @param WP_REST_Request $request Current request.

```



```

    */
    public function get_item_permissions_check( $request ) {
        if ( ! current_user_can( 'read' ) ) {
            return new WP_Error( 'rest_forbidden', esc_html__( 'You cannot view the post resource.' ), array( 'status' => $this->authorization_status_code() ) );
        }
        return true;
    }

    /**
     * Grabs the five most recent posts and outputs them as a rest response.
     *
     * @param WP_REST_Request $request Current request.
     */
    public function get_item( $request ) {
        $id = (int) $request['id'];
        $post = get_post( $id );

        if ( empty( $post ) ) {
            return rest_ensure_response( array() );
        }

        $response = prepare_item_for_response( $post );

        // Return all of our post response data.
        return $response;
    }

    /**
     * Matches the post data to the schema we want.
     *
     * @param WP_Post $post The comment object whose response is being prepared
     */
    public function prepare_item_for_response( $post, $request ) {
        $post_data = array();

        $schema = $this->get_item_schema( $request );

        // We are also renaming the fields to more understandable names.
        if ( isset( $schema['properties']['id'] ) ) {
            $post_data['id'] = (int) $post->ID;
        }

        if ( isset( $schema['properties']['content'] ) ) {
            $post_data['content'] = apply_filters( 'the_content', $post->post_content, $post );
        }
    }

```

```

        return rest_ensure_response( $post_data );
    }

    /**
     * Prepare a response for inserting into a collection of responses.
     *
     * This is copied from WP_REST_Controller class in the WP REST API v2 plugin.
     *
     * @param WP_REST_Response $response Response object.
     * @return array Response data, ready for insertion into collection data.
     */
    public function prepare_response_for_collection( $response ) {
        if ( ! ( $response instanceof WP_REST_Response ) ) {
            return $response;
        }

        $data = (array) $response->get_data();
        $server = rest_get_server();

        if ( method_exists( $server, 'get_compact_response_links' ) ) {
            $links = call_user_func( array( $server, 'get_compact_response_links' ), $response );
        } else {
            $links = call_user_func( array( $server, 'get_response_links' ), $response );
        }

        if ( ! empty( $links ) ) {
            $data['_links'] = $links;
        }

        return $data;
    }

    /**
     * Get our sample schema for a post.
     *
     * @param WP_REST_Request $request Current request.
     */
    public function get_item_schema( $request ) {
        $schema = array(
            // This tells the spec of JSON Schema we are using which is draft 4.

            '$schema' => 'http://json-schema.org/draft-04/schema#',

            // The title property marks the identity of the resource.
            'title' => 'post',
            'type' => 'object',
            // In JSON Schema you can specify object properties in the properti

```

```

es attribute.
    'properties'          => array(
        'id' => array(
            'description' => esc_html__( 'Unique identifier for the ob
ject.', 'my-textdomain' ),
            'type'        => 'integer',
            'context'     => array( 'view', 'edit', 'embed' ),
            'readonly'    => true,
        ),
        'content' => array(
            'description' => esc_html__( 'The content for the object.',
'my-textdomain' ),
            'type'        => 'string',
        ),
    ),
);

return $schema;
}

// Sets up the proper HTTP status code for authorization.
public function authorization_status_code() {

    $status = 401;

    if ( is_user_logged_in() ) {
        $status = 403;
    }

    return $status;
}
}

// Function to register our new routes from the controller.
function prefix_register_my_rest_routes() {
    $controller = new My_REST_Posts_Controller();
    $controller->register_routes();
}

add_action( 'rest_api_init', 'prefix_register_my_rest_routes' );

```

## 概述与未来

控制器类在开发端点时为我们解决两个大问题:缺乏命名空间和一致的结构。重要的是要注意,你不应滥用你的端点的继承。例如:如果您为一个帖子端点编写了一个控制器类,就像上面的例子,并且也希望支持自定义的帖子类型,你不应该这样扩展My\_REST\_Posts\_Controller: `class My_CPT_REST_Controller extends My_REST_Posts_Controller`。

相反，您应该创建一个完全独立的控制器类，或者使My\_REST\_Posts\_Controller处理所有可用的post类型。当您开始继承遗产的路径时，重要的是要明白，如果父类有必要在任何时候改变，而你的子类依赖于它们，那么您将会很头痛。在大多数情况下，您将要创建基控制器类作为接口或抽象类，每个端点控制器都可以实现或扩展。抽象类方法在核心WordPress REST API端点内使用。

## 扩展内部类

WordPress REST API遵循其内部类的故意设计模式，可以将其分类为基础架构或端点类。

基础架构类支持端点类。他们处理WordPress REST API的逻辑，而不执行任何数据转换。另一方面，端点类封装了对WordPress资源执行CRUD操作所必需的功能逻辑。更具体地说，我们的基础设施类包括WP\_REST\_Server和WP\_REST\_Request，我们的端点类包括WP\_REST\_Posts\_Controller和WP\_REST\_Users\_Controller。

让我们深入了解每个基础设施类的功能：

- WP\_REST\_Server：WordPress REST API的主控制器。路由被注册到WordPress中的服务器。当调用WP\_REST\_Server服务请求时，它确定要调用哪个路由，并将路由回调传递给WP\_REST\_Request对象。WP\_REST\_Server还处理身份验证，并可执行请求验证和权限检查。
- WP\_REST\_Request：表示请求性质的对象。该对象包括请求详细信息，如请求头，参数和方法以及路由。它还可以执行请求验证和消毒。
- WP\_REST\_Response：表示响应性质的对象。此类扩展了WP\_HTTP\_Response，其中包括头文件，正文和状态，并提供了一些辅助方法，如添加链接媒体的add\_link（），以及用于获取查询导航标题的query\_navigation\_headers（）。

所有端点类都扩展WP\_REST\_Controller。此类旨在表示操纵WordPress资源的一致模式。

WP\_REST\_Controller实现这些方法：

- register\_routes（）：在第一次实例化类之后，调用register\_routes（）将资源的路由注册到服务器。
- get\_items（）：获取现有实体的集合。
- get\_item（）：获取现有实体。如果实体不存在，则应返回HTTP错误代码404。如果请求者没有访问实体的权限，则应返回HTTP错误代码403。
- create\_item（）：创建一个新的实体，给定一个有效的WP\_REST\_Request。如果创建成功，则应返回WP\_REST\_Response，HTTP status = 201，并将位置头添加到新资源。如果以某种形式出现创建错误，则应返回适当的HTTP错误代码和消息。
- update\_item（）：给一个有效的WP\_REST\_Request更新现有实体。
- delete\_item（）：删除现有实体，给定一个有效的WP\_REST\_Request。如果以某种方式删除错误，则应返回适当的HTTP错误代码。
- get\_items\_permissions\_check（）：在调用回调之前，检查给定的请求是否具有资源集合的权限。

- `get_item_permissions_check ( )` : 在调用回调之前, 检查给定的请求是否具有获取单个资源的权限。
- `create_item_permissions_check ( )` : 在调用回调之前, 检查给定的请求是否具有创建单个资源的权限。
- `update_item_permissions_check ( )` : 在调用回调之前, 检查给定的请求是否具有更新单个资源的权限。
- `delete_item_permissions_check ( )` : 在调用回调之前, 检查给定的请求是否具有删除单个资源的权限。
- `prepare_item_for_response ( )` : 格式化一个资源以匹配它应该在响应中显示。
- `prepare_response_for_collection ( )` : 当使用`prepare_item_for_response ( )`时, 返回一个 `WP_REST_Response`。这个帮助函数将所有这些响应包装到一个集合中。
- `filter_response_by_context ( )` : 根据提供的上下文参数过滤响应形状。
- `get_item_schema ( )` : 获取资源的schema JSON Schema对象。

当与实现`WP_REST_Controller`的端点进行交互时, HTTP客户端可以期望每个端点以类似的方式运行。