这篇教程承接 [教程第 6 部分](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial06/) 结束的地方。我们继续修改在线投票应用，这次我们专注于自定义我们在 [教程第 2 部分](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial02/) 初见过的 Django 自动生成后台的过程。

自定义后台表单[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#customize-the-admin-form)

通过 **admin.site.register(Question)** 注册 **Question** 模型，Django 能够构建一个默认的表单用于展示。通常来说，你期望能自定义表单的外观和工作方式。你可以在注册模型时将这些设置告诉 Django。

让我们通过重排列表单上的字段来看看它是怎么工作的。用以下内容替换 **admin.site.register(Question)**：

polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id1)

**from** **django.contrib** **import** admin

**from** **.models** **import** Question

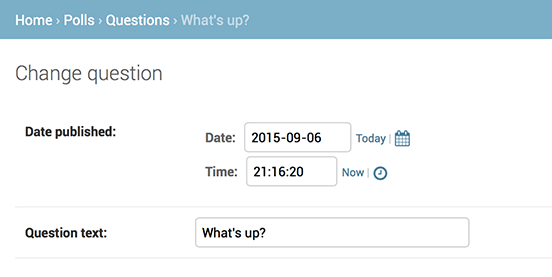
**class** **QuestionAdmin**(admin.ModelAdmin):

fields = ['pub\_date', 'question\_text']

admin.site.register(Question, QuestionAdmin)

你需要遵循以下流程——创建一个模型后台类，接着将其作为第二个参数传给 **admin.site.register()** ——在你需要修改模型的后台管理选项时这么做。

以上修改使得 "Publication date" 字段显示在 "Question" 字段之前：



这在只有两个字段时显得没啥卵用，但对于拥有数十个字段的表单来说，为表单选择一个直观的排序方法就显得你的针很细了。

说到拥有数十个字段的表单，你可能更期望将表单分为几个字段集：

polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id2)

**from** **django.contrib** **import** admin

**from** **.models** **import** Question

**class** **QuestionAdmin**(admin.ModelAdmin):

fieldsets = [

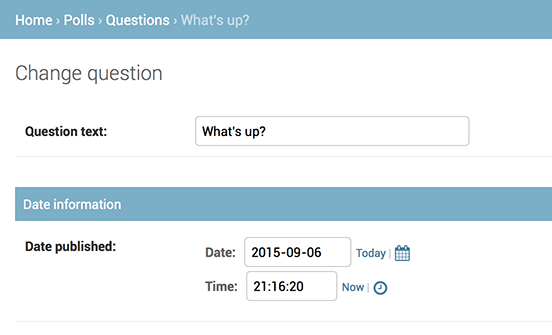
(None, {'fields': ['question\_text']}),

('Date information', {'fields': ['pub\_date']}),

]

admin.site.register(Question, QuestionAdmin)

[**fieldsets**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.fieldsets) 元组中的第一个元素是字段集的标题。以下是我们的表单现在的样子：



添加关联的对象[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#adding-related-objects)

好了，现在我们有了投票的后台页。不过，一个 **Question** 有多个 **Choice**，但后台页却没有显示多个选项。

好了。

有两个方法可以解决这个问题。第一个就是仿照我们向后台注册 **Question** 一样注册 **Choice** 。这很简单：

polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id3)

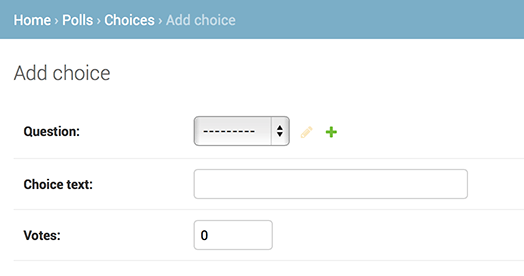
**from** **django.contrib** **import** admin

**from** **.models** **import** Choice, Question //这里也要改

*# ...*

admin.site.register(Choice)

现在 "Choices" 在 Django 后台页中是一个可用的选项了。“添加选项”的表单看起来像这样：



在这个表单中，"Question" 字段是一个包含数据库中所有投票的选择框。Django 知道要将 [**ForeignKey**](https://docs.djangoproject.com/zh-hans/2.1/ref/models/fields/#django.db.models.ForeignKey)在后台中以选择框 **<select>** 的形式展示。此时，我们只有一个投票。

同时也注意下 "Question" 旁边的“添加”按钮。每个使用 **ForeignKey** 关联到另一个对象的对象会自动获得这个功能。当你点击“添加”按钮时，你会见到一个包含“添加投票”的表单。如果你在这个弹出框中添加了一个投票，并点击了“保存”，Django 会将其保存至数据库，并动态地在你正在查看的“添加选项”表单中选中它。

不过，这是一种很低效地添加“选项”的方法。更好的办法是在你创建“投票”对象时直接添加好几个选项。让我们实现它。

移除调用 **register()** 注册 **Choice** 模型的代码。随后，像这样修改 **Question** 的注册代码：

polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id4)

**from** **django.contrib** **import** admin

**from** **.models** **import** Choice, Question

**class** **ChoiceInline**(admin.StackedInline):

model = Choice

extra = 3

**class** **QuestionAdmin**(admin.ModelAdmin):

fieldsets = [

(None, {'fields': ['question\_text']}),

('Date information', {'fields': ['pub\_date'], 'classes': ['collapse']}),

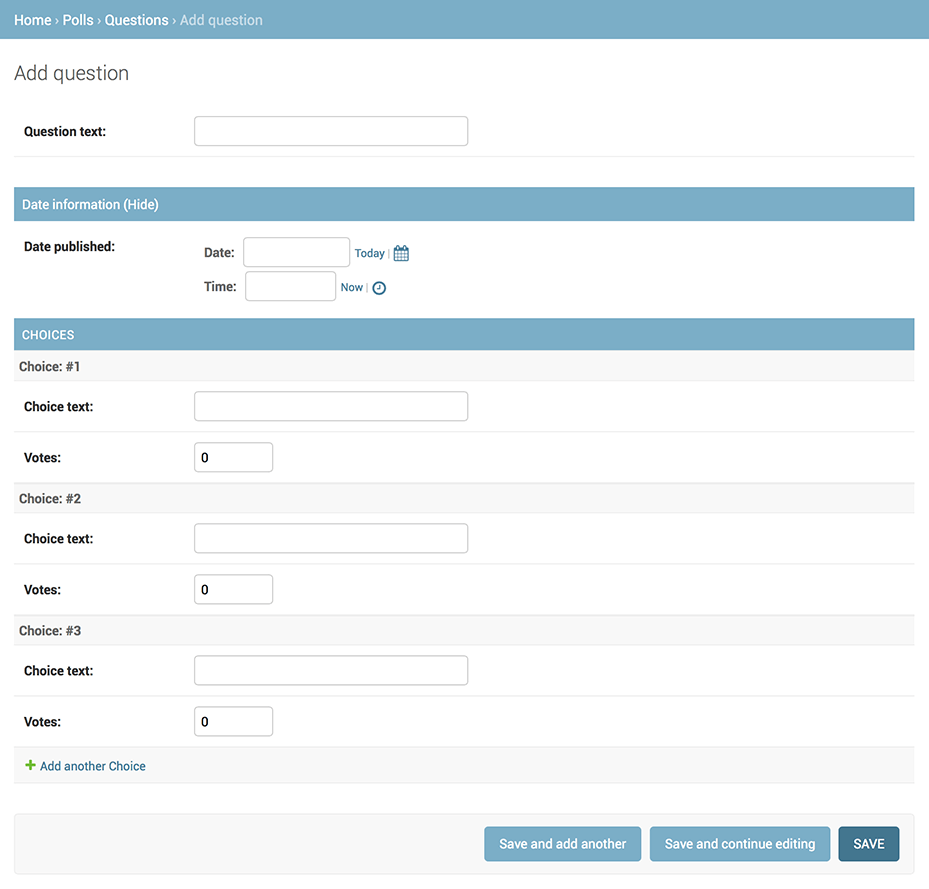
]

inlines = [ChoiceInline]

admin.site.register(Question, QuestionAdmin)

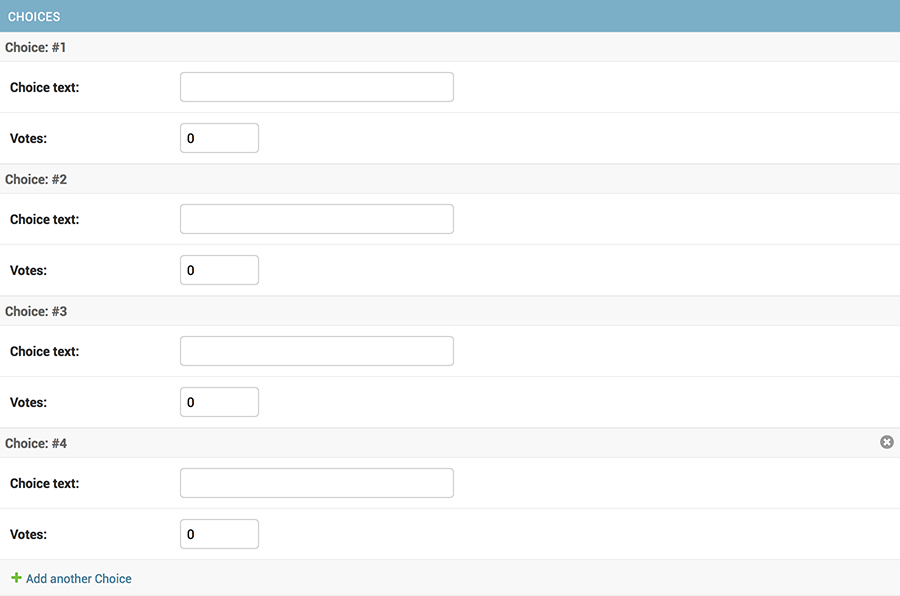
这会告诉 Django：“**Choice** 对象要在 **Question** 后台页面编辑。默认提供 3 个足够的选项字段。”

加载“添加投票”页面来看看它长啥样：



它看起来像这样：有三个关联的选项插槽——由 **extra** 定义，且每次你返回任意已创建的对象的“修改”页面时，你会见到三个新的插槽。

在三个插槽的末端，你会看到一个“添加新选项”的按钮。如果你单击它，一个新的插槽会被添加。如果你想移除已有的插槽，可以点击插槽右上角的X。注意，你不能移除原始的 3 个插槽。以下图片展示了一个已添加的插槽：



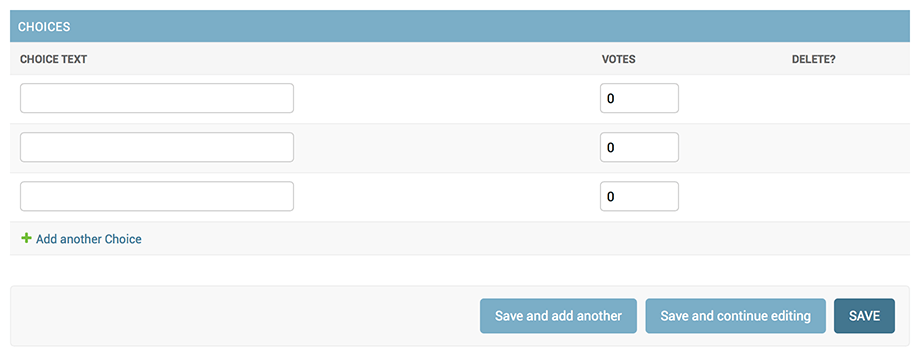
不过，仍然有点小问题。它占据了大量的屏幕区域来显示所有关联的 **Choice** 对象的字段。对于这个问题，Django 提供了一种表格式的单行显示关联对象的方法。你只需按如下形式修改 **ChoiceInline** 申明：

polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id5)

**class** **ChoiceInline**(admin.TabularInline):

*#...*

通过 **TabularInline``（替代 ``StackedInline** ），关联对象以一种表格式的方式展示，显得更加紧凑：

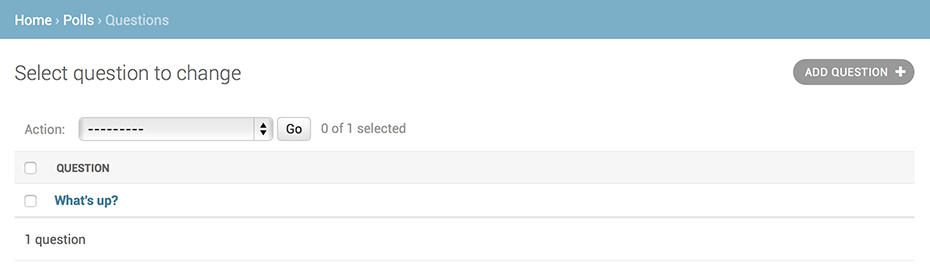


注意这里有一个额外的“删除？”列，这允许移除通过“添加新选项”按钮添加的，或是已被保存的行。

自定义后台更改列表[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#customize-the-admin-change-list)

现在投票的后台页看起来很不错，让我们对“更改列表”页面进行一些调整——改成一个能展示系统中所有投票的页面。

以下是它此时的外观：



默认情况下，Django 显示每个对象的 **str()** 返回的值。但有时如果我们能够显示单个字段，它会更有帮助。为此，使用 [**list\_display**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_display) 后台选项，它是一个包含要显示的字段名的元组，在更改列表页中以列的形式展示这个对象：

polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id6)

**class** **QuestionAdmin**(admin.ModelAdmin):

*# ...*

list\_display = ('question\_text', 'pub\_date')

为了更好用，让我们也包含 [教程第 2 部分](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial02/) 中的 **was\_published\_recently()** 方法：

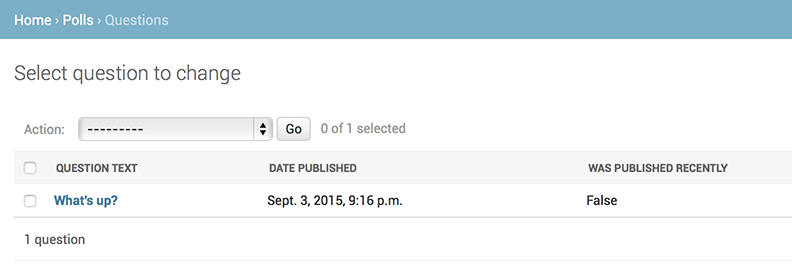
polls/admin.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id7)

**class** **QuestionAdmin**(admin.ModelAdmin):

*# ...*

list\_display = ('question\_text', 'pub\_date', 'was\_published\_recently')

现在修改投票的列表页看起来像这样：



你可以点击列标题来对这些行进行排序——除了 **was\_published\_recently** 这个列，因为没有实现排序方法。顺便看下这个列的标题 **was\_published\_recently**，默认就是方法名（用空格替换下划线），该列的每行都以字符串形式展示出处。

你可以通过给这个方法（在 **polls/models.py** 中）一些属性来达到优化的目的，像这样：

polls/models.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id8)

**class** **Question**(models.Model):

*# ...*

**def** was\_published\_recently(self):

now = timezone.now()

**return** now - datetime.timedelta(days=1) <= self.pub\_date <= now

was\_published\_recently.admin\_order\_field = 'pub\_date'

was\_published\_recently.boolean = True

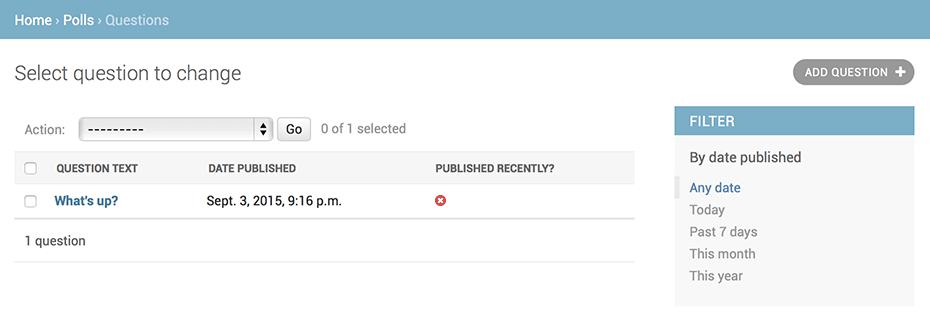
was\_published\_recently.short\_description = 'Published recently?'

更多关于这些方法属性的信息，参见 [**list\_display**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_display)。

再次编辑文件 **polls/admin.py**，优化 **Question** 变更页：过滤器，使用 [**list\_filter**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_filter)。将以下代码添加至 **QuestionAdmin**：

list\_filter = ['pub\_date']

这样做添加了一个“过滤器”侧边栏，允许人们以 **pub\_date** 字段来过滤列表：



展示的过滤器类型取决你你要过滤的字段的类型。因为 **pub\_date** 是类 [**DateTimeField**](https://docs.djangoproject.com/zh-hans/2.1/ref/models/fields/#django.db.models.DateTimeField)，Django 知道要提供哪个过滤器：“任意时间”，“今天”，“过去7天”，“这个月”和“今年”。

这已经弄的很好了。让我们再扩充些功能:

search\_fields = ['question\_text']

在列表的顶部增加一个搜索框。当输入待搜项时，Django 将搜索 **question\_text** 字段。你可以使用任意多的字段——由于后台使用 **LIKE** 来查询数据，将待搜索的字段数限制为一个不会出问题大小，会便于数据库进行查询操作。

现在是给你的修改列表页增加分页功能的好时机。默认每页显示 100 项。[**变更页分页**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_per_page), [**搜索框**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.search_fields), [**过滤器**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_filter), [**日期层次结构**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.date_hierarchy), 和 [**列标题排序**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_display) 均以你期望的方式合作运行。

## 自定义后台界面和风格[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#customize-the-admin-look-and-feel)

在每个后台页顶部显示“Django 管理员”显得很滑稽。这只是一串占位文本。

不过，这可以通过 Django 的模板系统很方便的修改。Django 的后台由自己驱动，且它的交互接口采用 Django 自己的模板系统。

### 自定义你的 工程的 模板[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#customizing-your-project-s-templates)

在你的工程目录（指包含 **manage.py** 的那个文件夹）内创建一个名为 **templates** 的目录。模板可放在你系统中任何 Django 能找到的位置。（谁启动了 Django，Django 就以他的用户身份运行。）不过，把你的模板放在工程内会带来很大便利，推荐你这样做。

打开你的设置文件（**mysite/settings.py**，牢记），在 [**TEMPLATES**](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-TEMPLATES) 设置中添加 [**DIRS**](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-TEMPLATES-DIRS) 选项：

mysite/settings.py[**¶**](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id9)

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE\_DIR, 'templates')],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

[**DIRS**](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-TEMPLATES-DIRS) 是一个包含多个系统目录的文件列表，用于在载入 Django 模板时使用，是一个待搜索路径。

**组织模板**

就像静态文件一样，我们 可以 把所有的模板文件放在一个大模板目录内，这样它也能工作的很好。但是，属于特定应用的模板文件最好放在应用所属的模板目录（例如 **polls/templates**），而不是工程的模板目录（**templates**）。我们会在 [创建可复用的应用教程](https://docs.djangoproject.com/zh-hans/2.1/intro/reusable-apps/) 中讨论 为什么 我们要这样做。

现在，在 **templates** 目录内创建名为 **admin** 的目录，随后，将存放 Django 默认模板的目录（**django/contrib/admin/templates**）内的模板文件 **admin/base\_site.html** 复制到这个目录内。

**Django 的源文件在哪里？**

如果你不知道 Django 源码在你系统的哪个位置，运行以下命令：

/ 

**$** python -c "import django; print(django.\_\_path\_\_)"

接着，用你站点的名字替换文件内的 [``](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#id1){{ site\_header|default:\_('Django administration') }}``（包含大括号）。完成后，你应该看到如下代码：

***{%*** **extends** "admin/base.html" ***%}***

***{%*** **block** title ***%}{{*** title ***}}*** | ***{{*** site\_title**|default:\_(**'Django site admin'**)** ***}}{%*** **endblock** ***%}***

***{%*** **block** branding ***%}***

<h1 id="site-name"><a href="***{%*** **url** 'admin:index' ***%}***">投票站点管理界面</a></h1>

***{%*** **endblock** ***%}***

***{%*** **block** nav-global ***%}{%*** **endblock** ***%}***

我们会用这个方法来教你复写模板。在一个实际工程中，你可能更期望使用 [**django.contrib.admin.AdminSite.site\_header**](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#django.contrib.admin.AdminSite.site_header) 来进行简单的定制。

这个模板文件包含很多类似 **{% block branding %}** 和 **{{ title }}** 的文本。 **{%** 和 **{{** 标签是 Django 模板语言的一部分。当 Django 渲染 **admin/base\_site.html** 时，这个模板语言会被求值，生成最终的网页，就像我们在 [教程第 3 部分](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial03/) 所学的一样。

注意，所有的 Django 默认后台模板均可被复写。若要复写模板，像你修改 **base\_site.html** 一样修改其它文件——先将其从默认目录中拷贝到你的自定义目录，再做修改。

### 自定义你 应用的 模板[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#customizing-your-application-s-templates)

机智的同学可能会问： [**DIRS**](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-TEMPLATES-DIRS) 默认是空的，Django 是怎么找到默认的后台模板的？因为 [**APP\_DIRS**](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-TEMPLATES-APP_DIRS) 被置为 **True**，Django 会自动在每个应用包内递归查找 **templates/** 子目录（不要忘了 **django.contrib.admin** 也是一个应用）。

我们的投票应用不是非常复杂，所以无需自定义后台模板。不过，如果它变的更加复杂，需要修改 Django 的标准后台模板功能时，修改 应用 的模板会比 工程 的更加明智。这样，在其它工程包含这个投票应用时，可以确保它总是能找到需要的自定义模板文件。

更多关于 Django 如何查找模板的文档，参见 [加载模板文档](https://docs.djangoproject.com/zh-hans/2.1/topics/templates/#template-loading)。

## 自定义后台主页[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#customize-the-admin-index-page)

在类似的说明中，你可能想要自定义 Django 后台索引页的外观。

默认情况下，它展示了所有配置在 [**INSTALLED\_APPS**](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-INSTALLED_APPS) 中，已通过后台应用注册，按拼音排序的应用。你可能想对这个页面的布局做重大的修改。毕竟，索引页是后台的重要页面，它应该便于使用。

需要自定义的模板是 **admin/index.html**。（像上一节修改 **admin/base\_site.html** 那样修改此文件——从默认目录中拷贝此文件至自定义模板目录）。打开此文件，你将看到它使用了一个叫做 **app\_list** 的模板变量。这个变量包含了每个安装的 Django 应用。你可以用任何你期望的硬编码链接（链接至特定对象的管理页）替代使用这个变量。

## 接下来要做什么？[¶](https://docs.djangoproject.com/zh-hans/2.1/intro/tutorial07/#what-s-next)

初学者教程到这就结束了。随后，你可能想阅读 [下一步看什么](https://docs.djangoproject.com/zh-hans/2.1/intro/whatsnext/)，看看下一步能做什么。

如果你很熟悉 Python 打包，且对学习如何把投票应用改成“可复用应用”感兴趣，查看 [进阶教程：如何创建可复用应用](https://docs.djangoproject.com/zh-hans/2.1/intro/reusable-apps/)。