

Django Admin 管理工具

Django 提供了基于 web 的管理工具。

Django 自动管理工具是 django.contrib 的一部分。你可以在项目的 settings.py 中的 INSTALLED_APPS 看到它：

/HelloWorld/HelloWorld/settings.py 文件代码：

```
INSTALLED_APPS = (  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
)
```

django.contrib是一套庞大的功能集，它是Django基本代码的组成部分。

激活管理工具

通常我们在生成项目时会在 urls.py 中自动设置好，我们只需去掉注释即可。

配置项如下所示：

/HelloWorld/HelloWorld/urls.py 文件代码：

```
# urls.py  
from django.conf.urls import url  
from django.contrib import admin  
urlpatterns = [  
    url(r'^admin/', admin.site.urls),  
]
```

当这一切都配置好后，Django 管理工具就可以运行了。

使用管理工具

启动开发服务器，然后在浏览器中访问 <http://127.0.0.1:8000/admin/>，得到如下界面：

Django administration

Username:

Password:

Log in

你可以通过命令 `python manage.py createsuperuser` 来创建超级用户，如下所示：

```
# python manage.py createsuperuser
Username (leave blank to use 'root'): admin
Email address: admin@runoob.com
Password:
Password (again):
Superuser created successfully.
[root@solar HelloWorld]#
```

之后输入用户名密码登录，界面如下：

Site administration | Django sit x

菜鸟教程

127.0.0.1:8000/admin/

🔍 ☆ ⋮

Django administration

WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

🔧 Change

Users

+ Add

🔧 Change

Recent actions

My actions

None available

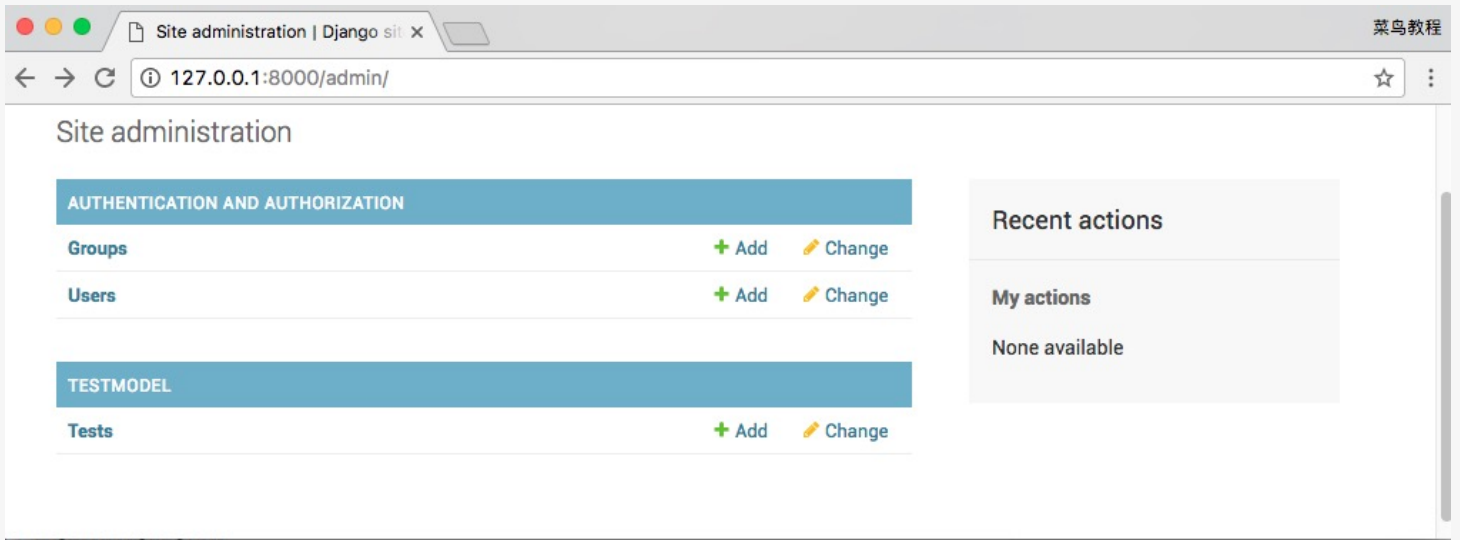
为了让 admin 界面管理某个数据模型，我们需要先注册该数据模型到 admin。比如，我们之前在 TestModel 中已经创建了模型 Test。修改 TestModel/admin.py:

HelloWorld/TestModel/admin.py: 文件代码：

```
from django.contrib import admin
from TestModel.models import Test
```

```
# Register your models here.
admin.site.register(Test)
```

刷新后即可看到 Testmodel 数据表:



复杂模型

管理页面的功能强大，完全有能力处理更加复杂的数据模型。

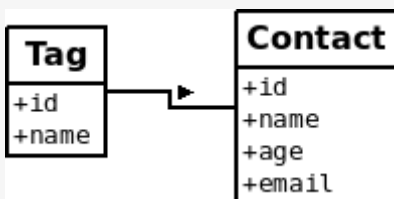
先在 TestModel/models.py 中增加一个更复杂的数据模型：

HelloWorld/TestModel/models.py: 文件代码：

```
from django.db import models
# Create your models here.
class Test(models.Model):
    name = models.CharField(max_length=20)
class Contact(models.Model):
    name = models.CharField(max_length=200)
    age = models.IntegerField(default=0)
    email = models.EmailField()
    def __unicode__(self):
        return self.name
class Tag(models.Model):
    contact = models.ForeignKey(Contact)
    name = models.CharField(max_length=50)
    def __unicode__(self):
        return self.name
```

这里有两个表。Tag 以 Contact 为外部键。一个 Contact 可以对应多个 Tag。

我们还可以看到许多在之前没有见过的属性类型，比如 IntegerField 用于存储整数。

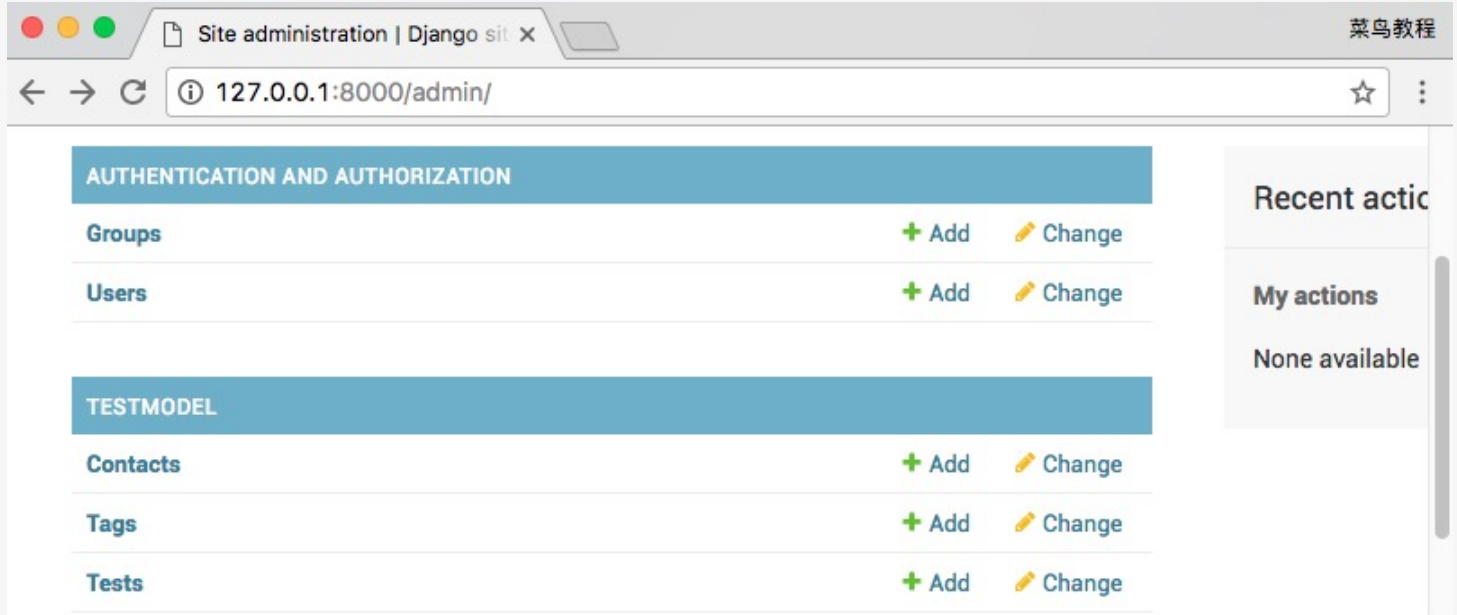


在 TestModel/admin.py 注册多个模型并显示：

HelloWorld/TestModel/admin.py: 文件代码：

```
from django.contrib import admin
from TestModel.models import Test, Contact, Tag
# Register your models here.
admin.site.register([Test, Contact, Tag])
```

刷新管理页面，显示结果如下：



在以上管理工具我们就能进行复杂模型操作。

如果你之前还未创建表结构，可使用以下命令创建：

```
$ python manage.py makemigrations TestModel # 让 Django 知道我们在我们的模型有一些变更
$ python manage.py migrate TestModel # 创建表结构
```

自定义表单

我们可以自定义管理页面，来取代默认的面。比如上面的 "add" 页面。我们想只显示 name 和 email 部分。修改 TestModel/admin.py:

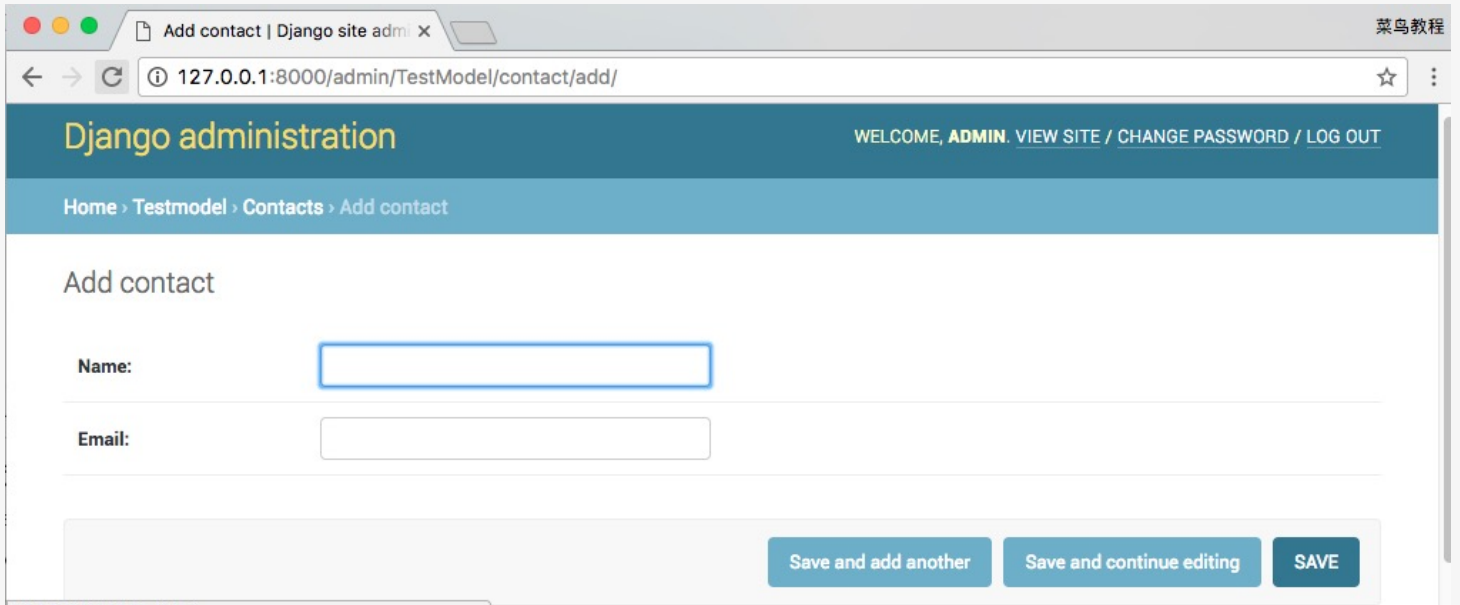
HelloWorld/TestModel/admin.py: 文件代码：

```
from django.contrib import admin
from TestModel.models import Test, Contact, Tag
# Register your models here.
class ContactAdmin(admin.ModelAdmin):
    fields = ('name', 'email')
admin.site.register(Contact, ContactAdmin)
admin.site.register([Test, Tag])
```

以上代码定义了一个 ContactAdmin 类，用以说明管理页面的显示格式。

里面的 `fields` 属性定义要显示的字段。

由于该类对应的是 `Contact` 数据模型，我们在注册的时候，需要将它们一起注册。显示效果如下：



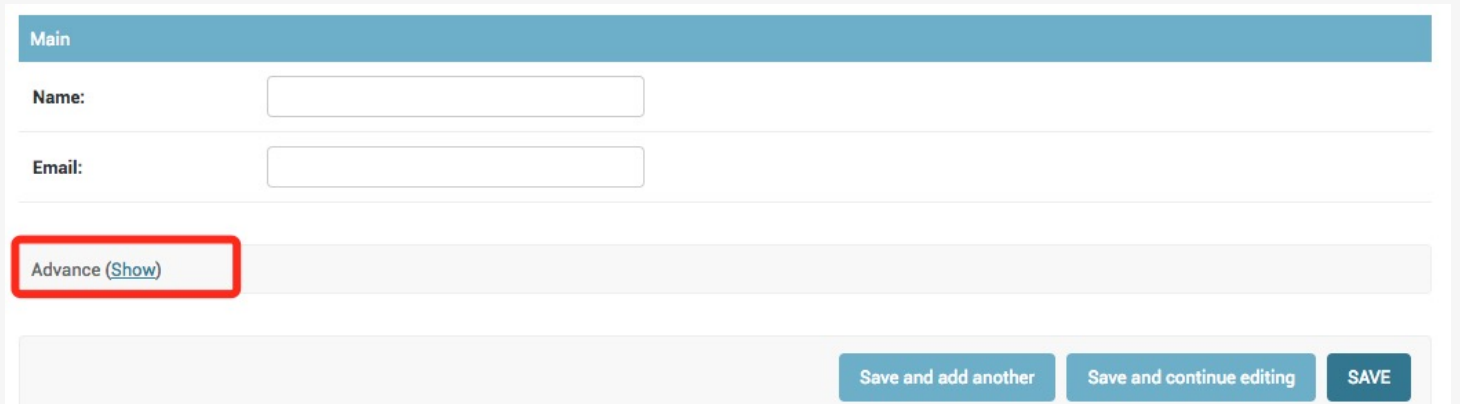
The screenshot shows the Django Admin interface for adding a new contact. The browser address bar shows the URL `127.0.0.1:8000/admin/TestModel/contact/add/`. The page title is "Django administration" and the user is logged in as "ADMIN". The breadcrumb trail is "Home > Testmodel > Contacts > Add contact". The form has two input fields: "Name:" and "Email:". At the bottom, there are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

我们还可以将输入栏分块，每个栏也可以定义自己的格式。修改 `TestModel/admin.py` 为：

HelloWorld/TestModel/admin.py: 文件代码：

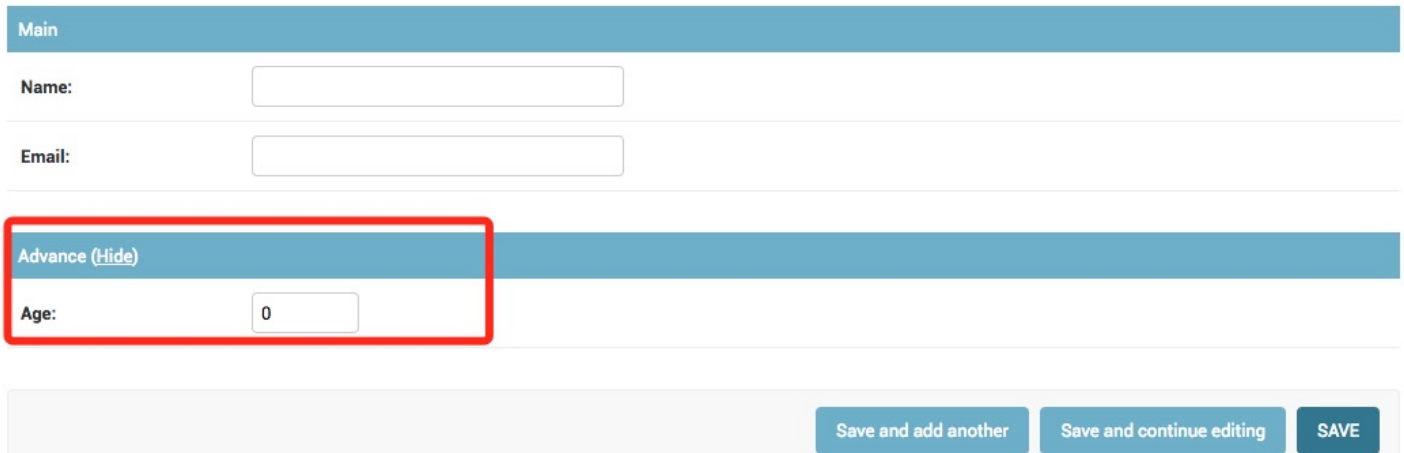
```
from django.contrib import admin
from TestModel.models import Test, Contact, Tag
# Register your models here.
class ContactAdmin(admin.ModelAdmin):
    fieldsets = (
        ['Main', {
            'fields': ('name', 'email'),
        }],
        ['Advance', {
            'classes': ('collapse',), # CSS
            'fields': ('age',),
        }]
    )
admin.site.register(Contact, ContactAdmin)
admin.site.register([Test, Tag])
```

上面的栏目分为了 `Main` 和 `Advance` 两部分。`classes` 说明它所在的部分的 CSS 格式。这里让 `Advance` 部分隐藏：



The screenshot shows the Django Admin interface for adding a new contact. The 'Main' section is expanded, showing the 'Name' and 'Email' fields. The 'Advance' section is collapsed, and a red box highlights the 'Advance (Show)' button. At the bottom, there are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

`Advance` 部分旁边有一个 `Show` 按钮，用于展开，展开后可点击 `Hide` 将其隐藏，如下图所示：



Main

Name:

Email:

Advance (Hide)

Age:

Save and add another Save and continue editing SAVE

内联(Inline)显示

上面的 Contact 是 Tag 的外部键，所以有外部参考的关系。

而在默认的页面显示中，将两者分离开来，无法体现出两者的从属关系。我们可以使用内联显示，让 Tag 附加在 Contact 的编辑页面上显示。

修改TestModel/admin.py：

HelloWorld/TestModel/admin.py: 文件代码：

```
from django.contrib import admin
from TestModel.models import Test, Contact, Tag
# Register your models here.
class TagInline(admin.TabularInline):
    model = Tag
class ContactAdmin(admin.ModelAdmin):
    inlines = [TagInline] # Inline
    fieldsets = (
        ['Main', {
            'fields': ('name', 'email'),
        }],
        ['Advance', {
            'classes': ('collapse',),
            'fields': ('age',),
        }]
    )
admin.site.register(Contact, ContactAdmin)
admin.site.register([Test])
```

显示效果如下：

127.0.0.1:8000/admin/TestModel/contact/add/

Main

Name:

Email:

Advance (Show)

TAGS

NAME	DELETE?
<input type="text"/>	
<input type="text"/>	
<input type="text"/>	

+ Add another Tag

列表页的显示

在 Contact 输入数条记录后，Contact 的列表页看起来如下：

✓ The contact "alibaba" was added successfully.

Select contact to change

ADD CONTACT +

Action: Go 0 of 3 selected

<input type="checkbox"/>	CONTACT
<input type="checkbox"/>	alibaba
<input type="checkbox"/>	google
<input type="checkbox"/>	runoob

我们也可以自定义该页面的显示，比如在列表中显示更多的栏目，只需要在 ContactAdmin 中增加 list_display 属性：

HelloWorld/TestModel/admin.py: 文件代码：

```
from django.contrib import admin
from TestModel.models import Test, Contact, Tag
# Register your models here.
class TagInline(admin.TabularInline):
    model = Tag
class ContactAdmin(admin.ModelAdmin):
    list_display = ('name', 'age', 'email') # list
    inlines = [TagInline] # Inline
```

```

fieldsets = (
    ['Main',{
        'fields':('name','email'),
    }],
    ['Advance',{
        'classes': ('collapse',),
        'fields': ('age',),
    }]
)
admin.site.register(Contact, ContactAdmin)
admin.site.register([Test])

```

刷新页面显示效果如下：

Action: 0 of 3 selected

<input type="checkbox"/>	NAME	AGE	EMAIL
<input type="checkbox"/>	alibaba	0	admin@alibaba.com
<input type="checkbox"/>	google	0	admin@google.com
<input type="checkbox"/>	runoob	3	admin@runoob.com

3 contacts

搜索功能在管理大量记录时非常有，我们可以使用 `search_fields` 为该列表页增加搜索栏：

HelloWorld/TestModel/admin.py: 文件代码：

```

from django.contrib import admin
from TestModel.models import Test,Contact,Tag
# Register your models here.
class TagInline(admin.TabularInline):
    model = Tag
class ContactAdmin(admin.ModelAdmin):
    list_display = ('name','age','email') # list
    search_fields = ('name',)
    inlines = [TagInline] # Inline
    fieldsets = (
        ['Main',{
            'fields':('name','email'),
        }],
        ['Advance',{
            'classes': ('collapse',),
            'fields': ('age',),
        }]
    )
admin.site.register(Contact, ContactAdmin)
admin.site.register([Test])

```

在本实例中我们搜索了 name 为 runoob 的记录，显示结果如下：

Select contact to change

1 result (3 total)

Action: ----- 0 of 1 selected

<input type="checkbox"/>	NAME	AGE	EMAIL
<input type="checkbox"/>	runoob	3	admin@runoob.com

Django Admin 管理工具还有非常多实用的功能，感兴趣的同学可以深入研究下。

[← Django 表单](#)[Django Nginx+uwsgi 安装配置 →](#)

2 篇笔记



写笔记



一个 Contact 类可以有多个 Tag:

关联 contact 外键时会报错：**TypeError: __init__() missing 1 required positional argument: 'on_delete'**

解决办法：

```
contact = models.ForeignKey(Contact, on_delete=models.CASCADE)
```

游客 9个月前 (06-26)



Django 在根据 models 生成数据库表时报 __init__() missing 1 required positional argument: 'on_delete' 错误

原因：

在 django2.0 后，定义外键和一对一关系的时候需要加 on_delete 选项，此参数为了避免两个表里的数据不一致问题，不然会报错：**TypeError: __init__() missing 1 required positional argument: 'on_delete'**。

举例说明：

```
user=models.OneToOneField(User)owner=models.ForeignKey(UserProfile)
```

需要改成：

```
user=models.OneToOneField(User,on_delete=models.CASCADE) --在老版本这个参数（models.CASCADE）是默认值
owner=models.ForeignKey(UserProfile,on_delete=models.CASCADE) --在老版本这个参数（models.CASCADE）是默认值参数
```

说明：on_delete 有 CASCADE、PROTECT、SET_NULL、SET_DEFAULT、SET() 五个可选择的值。

- **CASCADE**：此值设置，是级联删除。
- **PROTECT**：此值设置，是会报完整性错误。

- ● **SET_NULL**：此值设置，会把外键设置为 null，前提是允许为 null。
- ● **SET_DEFAULT**：此值设置，会把设置为外键的默认值。
- ● **SET()**：此值设置，会调用外面的值，可以是一个函数。一般情况下使用 CASCADE 就可以了。

乐斗之星 2个月前 [01-28]