Django-Admin-Cookbook-CN

Release 1

Jan 31, 2020

Contents

CHAPTER 1

介绍

1.1 译者的话

身为 PythonWeb 开发热爱者

希望能对开源社区做出一些贡献，尽量抽出时间，尽早完成翻译。

想一起的小伙伴可以联系我。QQ: 189219902

本人能力有限，难免有疏漏或者表意不当的地方。如果译文中有什么错漏的地方请大家见谅，也欢迎大家随时指正。

英文原文 Django Admin Cookbook

1.2 Django Admin Cookbook 介绍

Django Admin Cookbook 是一本关于 Django Admin 模块的书

它面向 Django 的对 DjangoAdmin 有一些使用经验但是想要扩大对 DjangoAdmin 的认知甚至想要精通 DjangoAdmin 的中级 Django 开发者.

它采用问答的形式来讨论关于你可能会使用 DjangoAdmin 完成的常见任务。

所有章节均基于一组通用模型:

1.2.1 App entities

```
class Category(models.Model):
    name = models.CharField(max_length=100)

    class Meta:
        verbose_name_plural = "Categories"

    def __str__(self):
        return self.name


class Origin(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name


class Entity(models.Model):
    GENDER_MALE = "Male"
    GENDER_FEMALE = "Female"
    GENDER_OTHERS = "Others/Unknown"

    name = models.CharField(max_length=100)
    alternative_name = models.CharField(
        max_length=100, null=True, blank=True
    )


    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    origin = models.ForeignKey(Origin, on_delete=models.CASCADE)
    gender = models.CharField(
        max_length=100,
        choices=(
            (GENDER_MALE, GENDER_MALE),
            (GENDER_FEMALE, GENDER_FEMALE),
            (GENDER_OTHERS, GENDER_OTHERS),
        )
    )
    description = models.TextField()
```

(continues on next page)

(continued from previous page)

```
    def __str__(self):
        return self.name

    class Meta:
        abstract = True


class Hero(Entity):

    class Meta:
        verbose_name_plural = "Heroes"

    is_immortal = models.BooleanField(default=True)

    benevolence_factor = models.PositiveSmallIntegerField(
        help_text="How benevolent this hero is?"
    )
    arbitrariness_factor = models.PositiveSmallIntegerField(
        help_text="How arbitrary this hero is?"
    )
    # relationships
    father = models.ForeignKey(
        "self", related_name="+", null=True, blank=True, on_delete=models.SET_NULL
    )
    mother = models.ForeignKey(
        "self", related_name="+", null=True, blank=True, on_delete=models.SET_NULL
    )
    spouse = models.ForeignKey(
        "self", related_name="+", null=True, blank=True, on_delete=models.SET_NULL
    )


class Villain(Entity):
    is_immortal = models.BooleanField(default=False)

    malevolence_factor = models.PositiveSmallIntegerField(
        help_text="How malevolent this villain is?"
    )
    power_factor = models.PositiveSmallIntegerField(
```

(continues on next page)

(continued from previous page)

```
        help_text="How powerful this villain is?"
    )
    is_unique = models.BooleanField(default=True)
    count = models.PositiveSmallIntegerField(default=1)###
```

1.2.2 App events

```
    name = models.CharField(max_length=255)
    participating_heroes = models.ManyToManyField(Hero)
    participating_villains = models.ManyToManyField(Villain)


class Event(models.Model):
    epic = models.ForeignKey(Epic, on_delete=models.CASCADE)
    details = models.TextField()
    years_ago = models.PositiveIntegerField()


class EventHero(models.Model):
    event = models.ForeignKey(Event, on_delete=models.CASCADE)
    hero = models.ForeignKey(Hero, on_delete=models.CASCADE)
    is_primary = models.BooleanField()


class EventVillain(models.Model):
    event = models.ForeignKey(Event, on_delete=models.CASCADE)
    hero = models.ForeignKey(Villain, on_delete=models.CASCADE)
    is_primary = models.BooleanField()
```

1.3 如何使用这本书

你可以把这本书从头读到尾，也可以选择你需要的章节读，每一个章节是独立的，特定的任务

无论如何，你都应该先读 `entities/models.py` 和 `events/models.py`

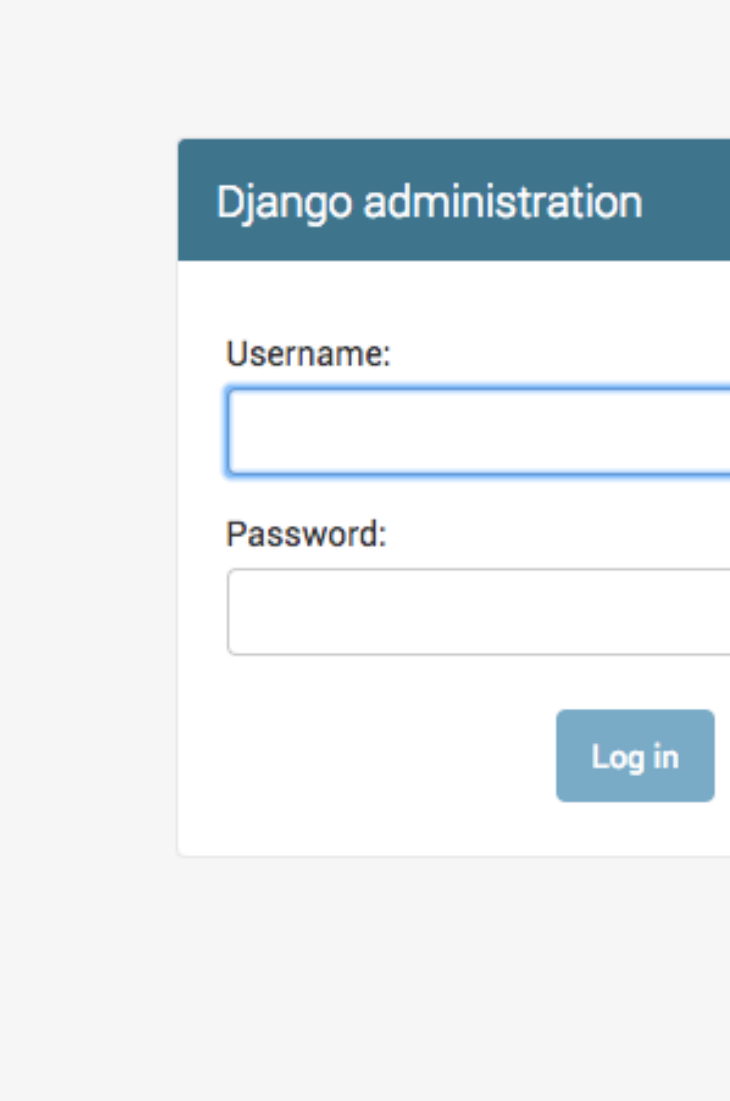
CHAPTER 2

1. 怎么去修改‘Django administration’文字?

Django admin 默认就是显示‘Django administration’. 你被要求使用‘UMSRA Administration’代替文字在这些页面上:

- 登录页面
- 列表显示页面
- HTML 的标题

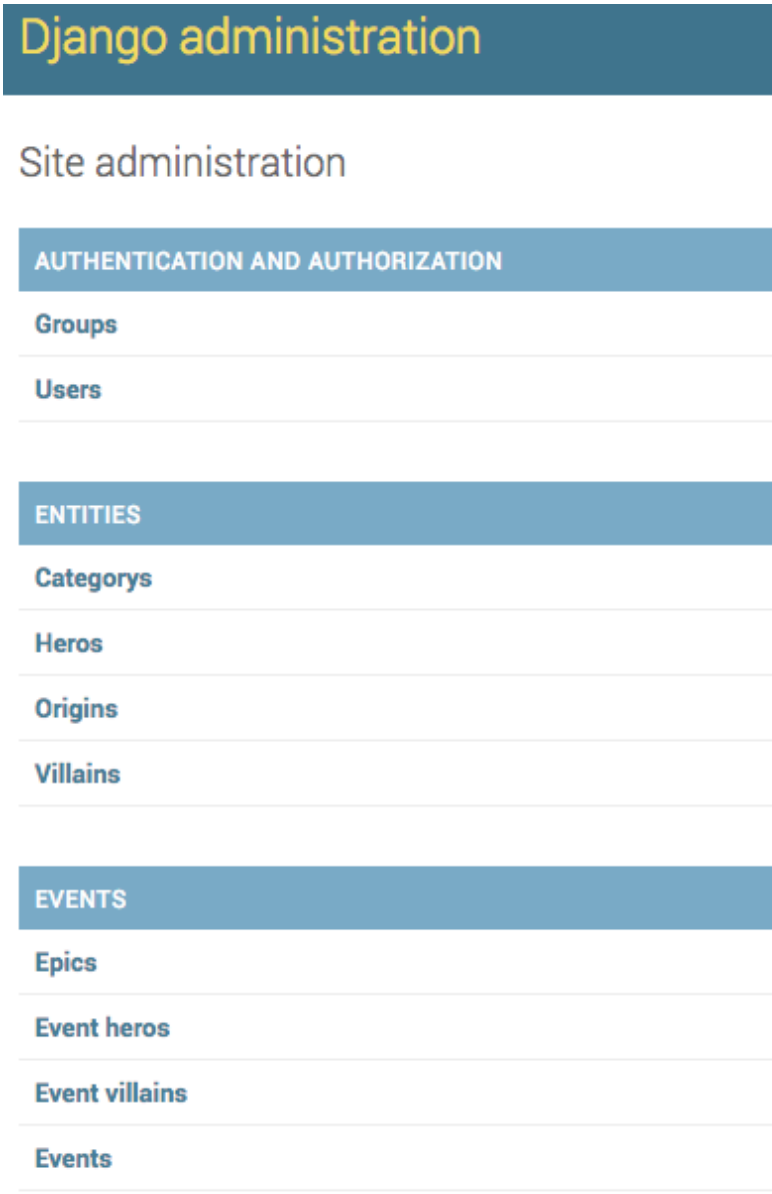
2.1 1.1. 登录，列表，变更视图



默认情况下,它会像这样,并且设置成‘Django administration’

`site_header` 可以设置并修改它

2.2 1.2. 列表视图



默认情况下,它会像这样,并且设置成‘Site administration’

`index_title` 可以设置并修改它

2.3 1.3.HTML 标题标签

默认情况下，它会像这样，并且设置成‘Django site admin’

site_title 可以设置并修改它

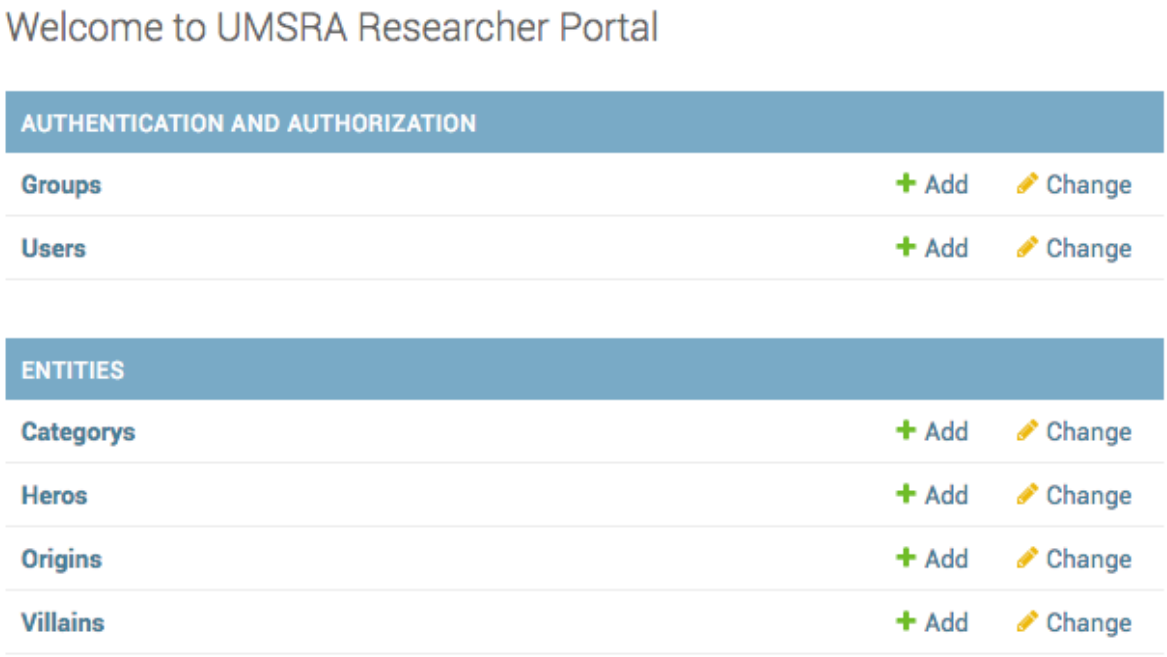
我们可以在 url.py 做出这 3 处修改：

```
admin.site.site_header = "UMSRA Admin"
admin.site.site_title = "UMSRA Admin Portal"
admin.site.index_title = "Welcome to UMSRA Researcher Portal"
```

CHAPTER 3

2. 怎么为一个模型设置复数文字

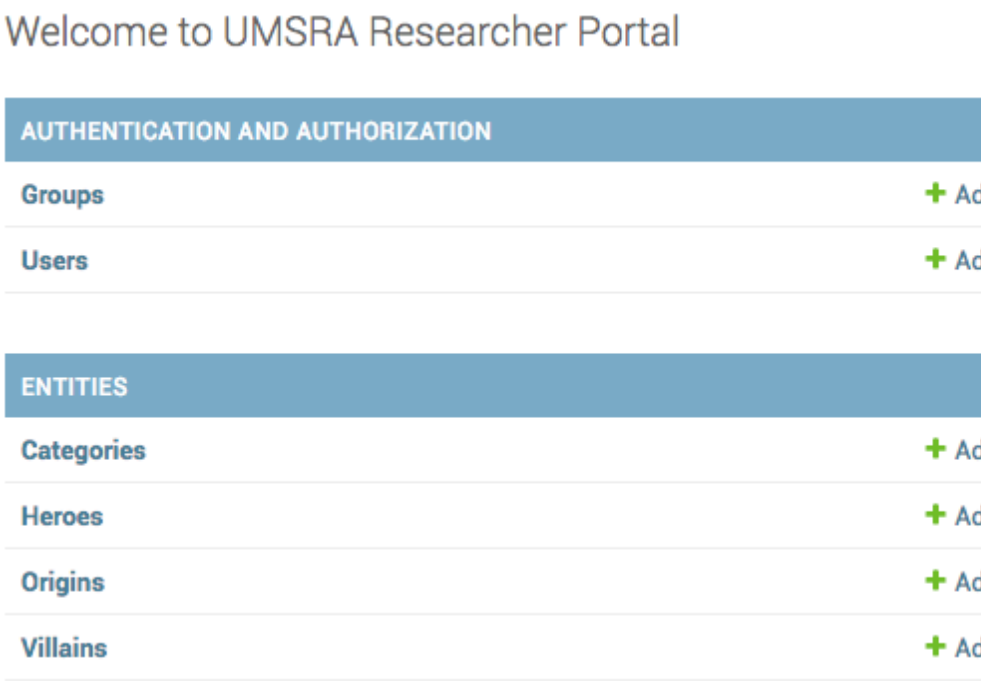
默认情况下，admin 给你的模型末尾加上‘s’，又称做你模型的复数形式，像这个样子：



你被要求设置正常的复数形式，*Categories and Heroes*

你可以通过在模型里面设置 verbose_name_plural 来达到这个目标，在你的模型中修改：

```
class Category(models.Model):
    ...

    class Meta:
        verbose_name_plural = "Categories"


class Hero(Entity):
    ...

    class Meta:
        verbose_name_plural = "Heroes"
```



修改之后,你的 admin 就会成为这样子:

CHAPTER 4

3. 怎么去创建 2 个独立的 admin 站点

创建 admin 页面的通常方法是把所有的模型放到单一的 admin 文件中，然而，在单一的 Django 应中，我们可以有多个 admin 站点现在我们的 `entity` 和 `event` 模型在同一个位置，UMSRA 有 2 个不同的组去研究 *Events* 和 *Entities*，所以希望将对这个 2 个模型的管理分开

我们将保持对 *entities* 的默认的 admin 保持不变，并且为 *events* 创建新的 `AdminSite` 在我们的 events/admin.py:

```
from django.contrib.admin import AdminSite
class EventAdminSite(AdminSite):
    site_header = "UMSRA Events Admin"
    site_title = "UMSRA Events Admin Portal"
    index_title = "Welcome to UMSRA Researcher Events Portal"

event_admin_site = EventAdminSite(name='event_admin')



event_admin_site.register(Epic)
event_admin_site.register(Event)
event_admin_site.register(EventHero)
event_admin_site.register(EventVillain)
```

并改变 url.py：

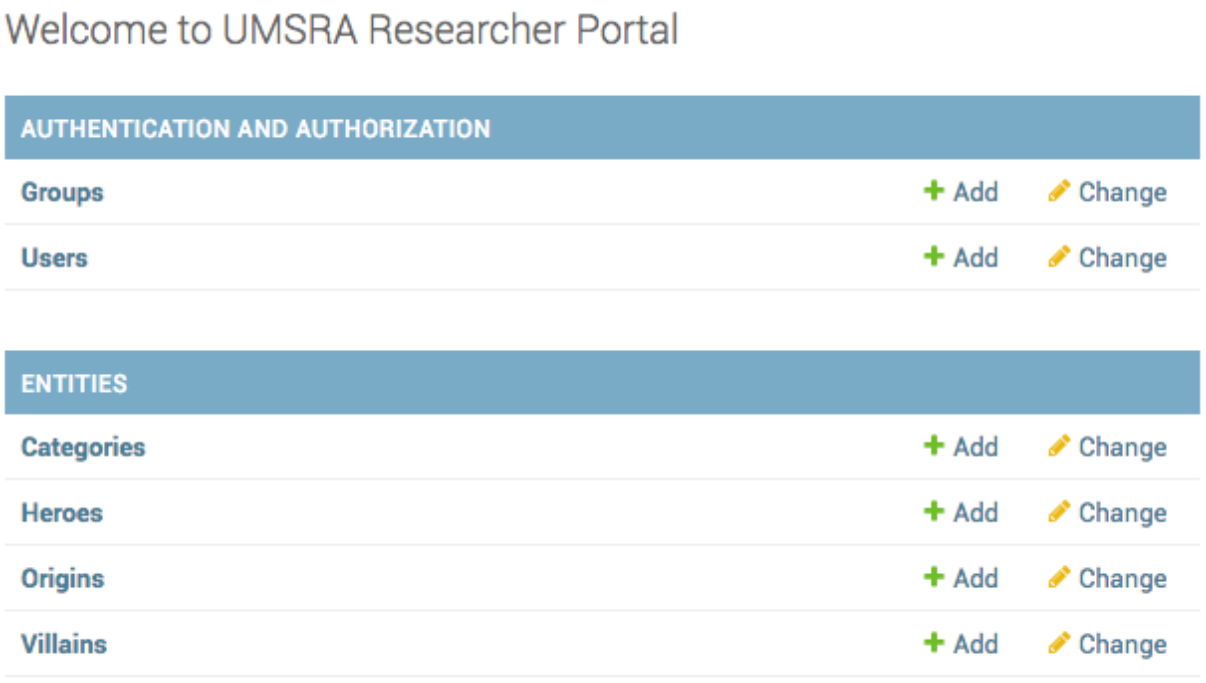
```
from events.admin import event_admin_site

urlpatterns = [
    path('entity-admin/', admin.site.urls),
    path('event-admin/', event_admin_site.urls),
]
```

这样就实现了分隔 admin,2 个管理都在各自的 url 下可以使用 `/entity-admin/` 和 `event-admin/`

CHAPTER 5

4. 怎么把默认的 apps 从 Django admin 移除

Django 会把 `django.contrib.auth` 添加到 `INSTALLED_APPS`, 这意味着, *User* 和 *Groups* 模型会被自动添加



到 admin 里面

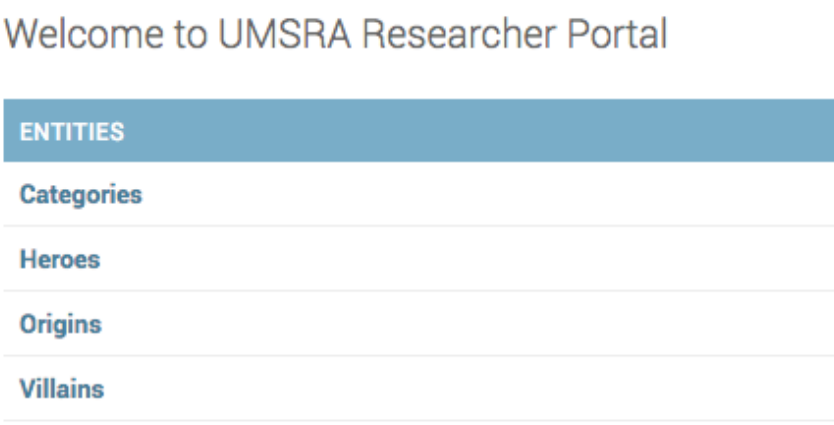
如果你想移除它的话，你需要去取消注册掉他们

```
from django.contrib.auth.models import User, Group
```

(continues on next page)

(continued from previous page)

```
admin.site.unregister(User)
admin.site.unregister(Group)
```



当你做完这些改变之后,你的 admin 就像下面这样:

CHAPTER 6

5. 怎么给 Django-admin 添加图标

UMSRA 的领导现喜欢你到目前创建的 admin，但是市场营销人员希望你在所有的 admin 页面添加 UMSRA logo 图标

你需要在 django settings 重写 django 提供的默认 templates django 设置 *TEMPLATES* 的代码像这样：

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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',
            ],
        },
    },
]
```

这意味着 Django 将在每个应用程序内的名为 `template` 的目录中查找模板，但是你可以通过重写 TEMPLATES.DIRS 的值，我可以把`'DIRS': []` 转换成`'DIRS': [os.path.join(BASE_DIR, 'templates/')]`，并且创建一个 `templaes` 文件夹。如果你的 `STATICFILE_DIRS` 是空的话，就设置：

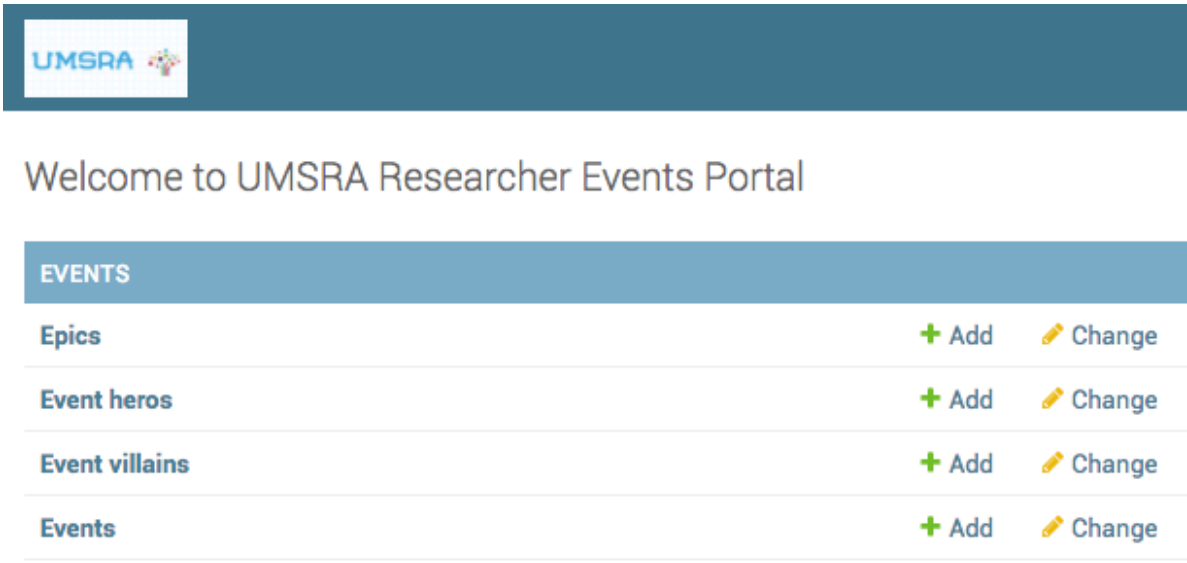
```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
```

现在从 admin app 里面拷贝 `base_site.html` 到你刚刚创建的 `templates\admin` 文件夹下，替换默认的品牌标记块

```
<h1 id="site-name">
    <a href="{% url 'admin:index' %}">
        <img src="{% static 'umsra_logo.png' %}" height="40px" />
    </a>
</h1>
```

改变你的 base_site.html 之后，将会像这样：

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

{% load staticfiles %}

{% block title %}{{ title }} | {{ site_title|default:_('Django site admin') }}{%␣
↪endblock %}

{% block branding %}
<h1 id="site-name">
    <a href="{% url 'admin:index' %}">
        <img src="{% static 'umsra_logo.png' %}" height="40px" />
    </a>
</h1>
{% endblock %}

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



你的 admin 会像这样：

CHAPTER 7

6. 如何重写 Django 管理模板？

https://docs.djangoproject.com/en/dev/ref/contrib/admin/#overriding-admin-templates

CHAPTER 8

1. 如何在列表页展示计算字段

你有一个 Origin 模型的 admin，向下面这样：

```
@admin.register(Origin)
class OriginAdmin(admin.ModelAdmin):
    list_display = ("name",)
```

除了名称，我们还希望显示每个 Origin 的 Hero 英雄数量和 Villain 反派数量，这些不是 Origin 上的 DB 字段。您可以通过两种方式执行此操作。

8.1 1.1 在模型上添加方法

你可以为你的 Origin 添加 2 个方法，像下面这样：

```
def hero_count(self, obj):
    return obj.hero_set.count()

def villain_count(self, obj):
    return obj.villain_set.count()
```

list_display 改为 list_display = ("name", "hero_count", "villain_count")

8.2 1.2 为 ModelAdmin 添加方法

如果你不想再 model 上添加方法，你可以在 ModelAdmin 上添加

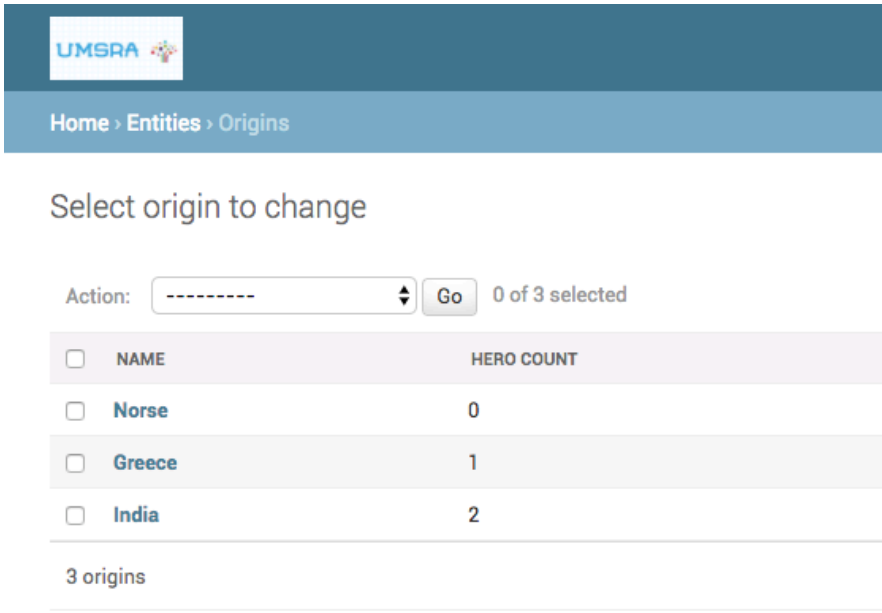
```
def hero_count(self, obj):
    return obj.hero_set.count()

def villain_count(self, obj):
    return obj.villain_set.count()
```

list_display 改为 list_display = ("name", "hero_count", "villain_count")

8.3 1.3 计算字段，性能的考虑

上面 2 个方法，你将会进行为每个对象进行二次查询，你可以查看如何在 Django Admin 优化查询

进行上述更改之后，你的 admin 就会像下面这样：



CHAPTER 9

2. 如何在 Django admin 优化查询

如果在你的 admin 有大量的计算字段，你可能会对每个对象进行多个查询，导致你的 admin 页面变得非常慢，要修复这个问题，你可以在 ModelAdmin 上注释这些计算字段。让我们以下面这个 ModelAdmin 举例子，

```
@admin.register(Origin)
class OriginAdmin(admin.ModelAdmin):
    list_display = ("name", "hero_count", "villain_count")

    def hero_count(self, obj):
        return obj.hero_set.count()

    def villain_count(self, obj):
        return obj.villain_set.count()
```

这将为 listview 页面中的每一行添加两个额外的查询。要解决这个问题，你可以覆盖 `get_queryset` 来注释计算字段，然后在 ModelAdmin 方法中使用带注释的字段。

修改之后，你的 ModelAdmin 如下：

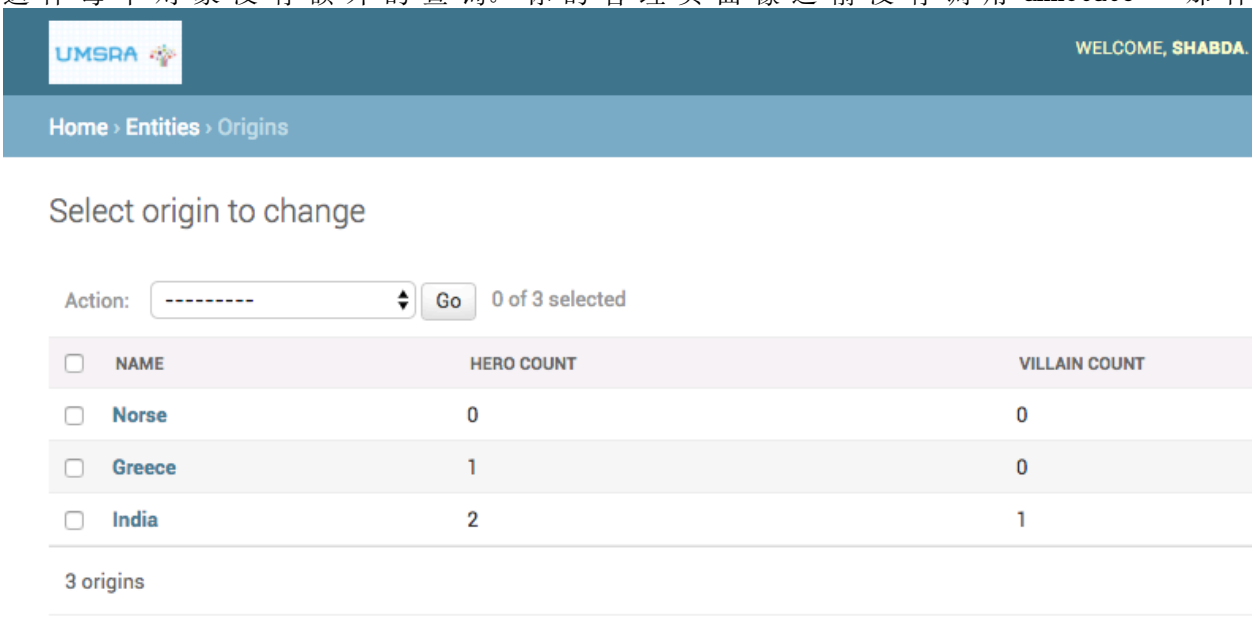
```
@admin.register(Origin)
class OriginAdmin(admin.ModelAdmin):
    list_display = ("name", "hero_count", "villain_count")

    def get_queryset(self, request):
```

(continues on next page)

(continued from previous page)

```
        queryset = super().get_queryset(request)
        queryset = queryset.annotate(
            _hero_count=Count("hero", distinct=True),
            _villain_count=Count("villain", distinct=True),
        )
        return queryset

    def hero_count(self, obj):
        return obj._hero_count

    def villain_count(self, obj):
        return obj._villain_count
```

这样每个对象没有额外的查询。你的管理页面像之前没有调用 annotate 那样



CHAPTER 10

3. 如何让计算字段可以排序

Django 在模型属性的字段上添加了排序功能。当您添加计算字段时，Django 不知道如何执行 `order_by`，因此不会在该字段上添加排序功能。如果你想让计算字段也可以排序，你需要告诉 Django 如何去 `order_by` 你可以在计算字段方法上设置 `admin_order_filed` 属性。

从之前的章节如何在 django admin 中优化查询的页面中，添加如下：

```
hero_count.admin_order_field = '_hero_count'
villain_count.admin_order_field = '_villain_count'
```

修改了这些之后，你的 admin 就变成了这样：

```
@admin.register(Origin)
class OriginAdmin(admin.ModelAdmin):
    list_display = ("name", "hero_count", "villain_count")

    def get_queryset(self, request):
        queryset = super().get_queryset(request)
        queryset = queryset.annotate(
            _hero_count=Count("hero", distinct=True),
            _villain_count=Count("villain", distinct=True),
        )
        return queryset

    def hero_count(self, obj):
```

(continues on next page)

(continued from previous page)

```
        return obj._hero_count

    def villain_count(self, obj):
        return obj._villain_count

    hero_count.admin_order_field = '_hero_count'
    villain_count.admin_order_field = '_villain_count'
```

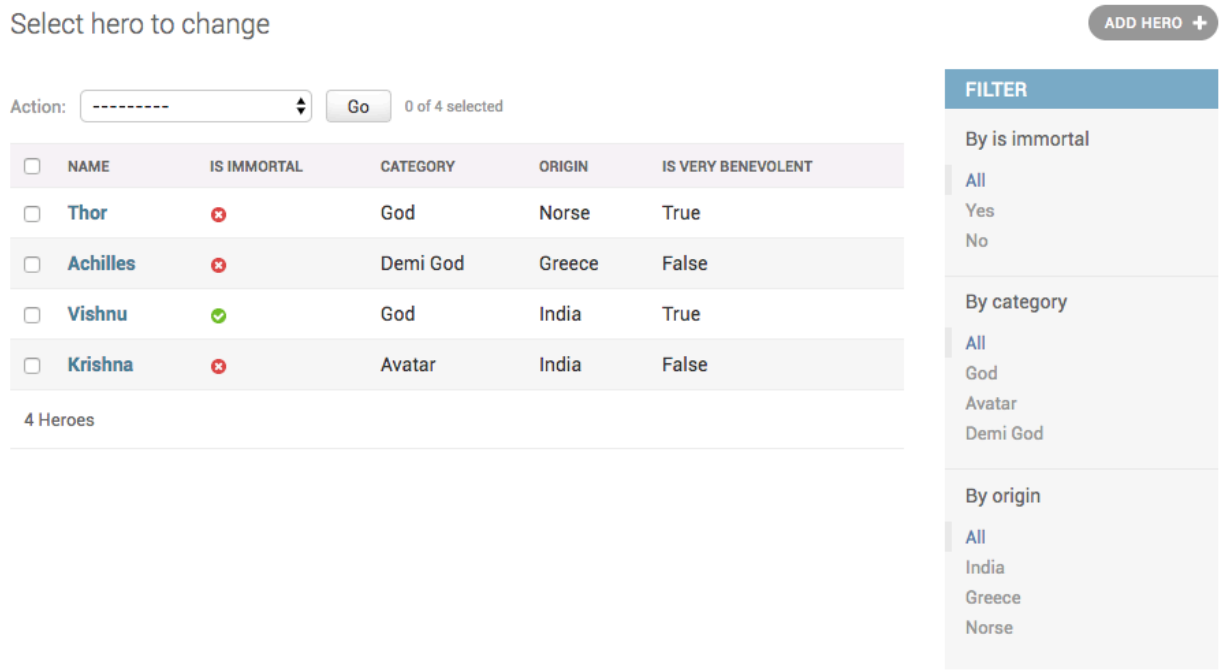
CHAPTER 11

4. 如何让计算字段可以过滤

你有 Heroadmin，像下面这样：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin):
    list_display = ("name", "is_immortal", "category", "origin", "is_very_benevolent")
    list_filter = ("is_immortal", "category", "origin",)

    def is_very_benevolent(self, obj):
        return obj.benevolence_factor > 75
```

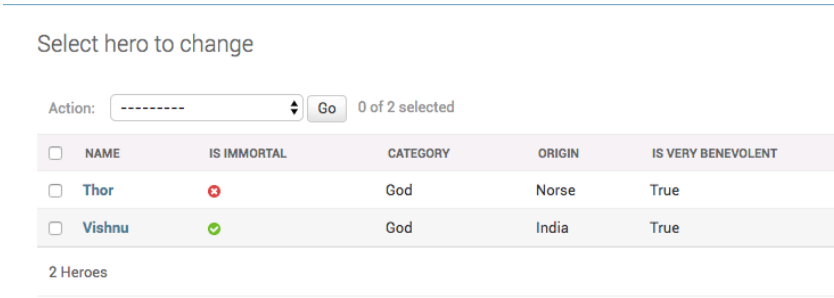
如果有一个 is_very_benevolent 计算字段，你的 admin 页面会像这样一样：



你已经在模型字段中添加了过滤，但是你还想计算字段添加过滤，为了做到这个，你需要这样子类化 SimpleListFilter

```
class IsVeryBenevolentFilter(admin.SimpleListFilter):
    title = 'is_very_benevolent'
    parameter_name = 'is_very_benevolent'

    def lookups(self, request, model_admin):
        return (
            ('Yes', 'Yes'),
            ('No', 'No'),
        )

    def queryset(self, request, queryset):
        value = self.value()
        if value == 'Yes':
            return queryset.filter(benevolence_factor__gt=75)
        elif value == 'No':
            return queryset.exclude(benevolence_factor__gt=75)
        return queryset
```

并且将你的 list_filter 改为: `list_filter = ("is_immortal", "category", "origin", IsVeryBenevolentFilter)`



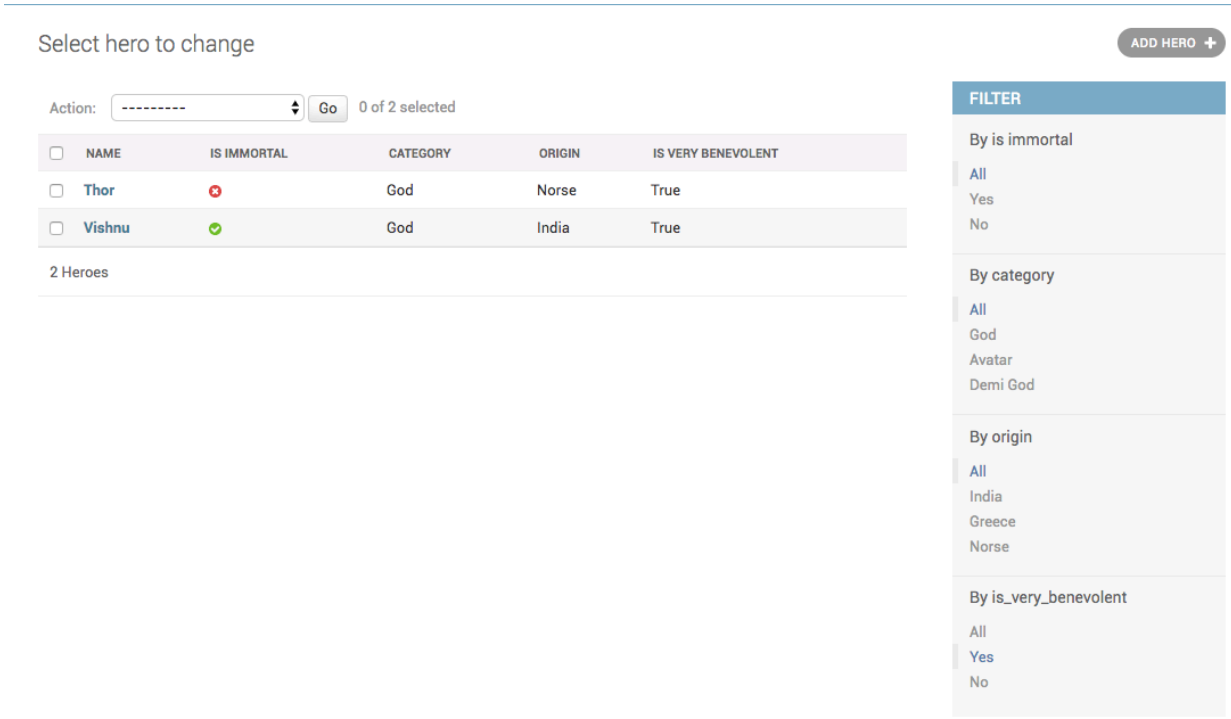
这样你可以过滤你的计算字段,admin 就变成了这样:

CHAPTER 12

5. 布尔的计算字段显示‘on’或者‘off’图标

在之前的章节如何让计算字段可过滤中，你添加了一个布尔的字段

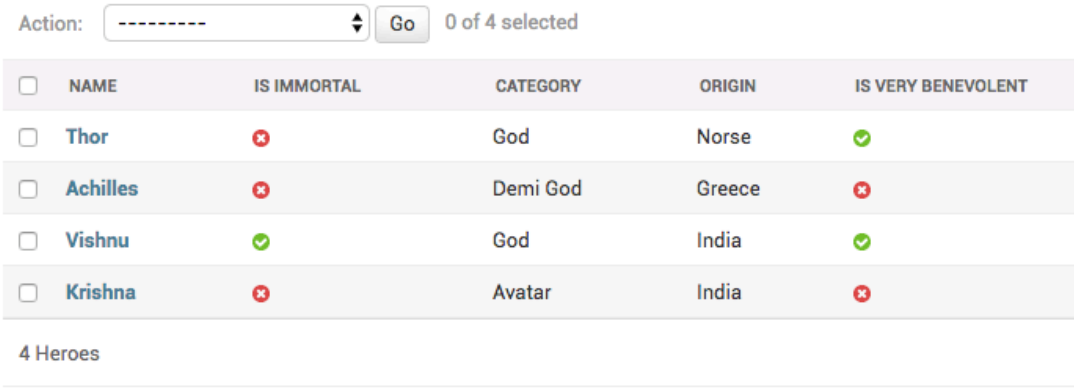
```
def is_very_benevolent(self, obj):
    return obj.benevolence_factor > 75
```



就像这样:

这里的 `is_very_benevolent` 字段展示了 True 或者 False，不像内置的布尔字段展示 on 或者 off 标识符，为了解决这个问题你可以在你的方法中添加一个 boolean 属性，你最后的 modeladmin 像这样：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin):
    list_display = ("name", "is_immortal", "category", "origin", "is_very_benevolent")
    list_filter = ("is_immortal", "category", "origin", IsVeryBenevolentFilter)

    def is_very_benevolent(self, obj):
        return obj.benevolence_factor > 75

    is_very_benevolent.boolean = True
```



你的 admin 管理页面像这样：

CHAPTER 13

1. 如何在 Django admin 添加额外的 actions

Django admin 可以允许你添加额外的 actions，这些 actions 可以允许你进行批量操作，你现在被要求添加一个将多个 `Hero` 标记为不朽的 action 你可以通过 ModelAdmin 加上该方法，并将该方法作为字符串添加到 `actions` 中

```
actions = ["mark_immortal"]

def mark_immortal(self, request, queryset):
    queryset.update(is_immortal=True)
```

CHAPTER 14

2. 如何从 djagno admin 导出 CSV

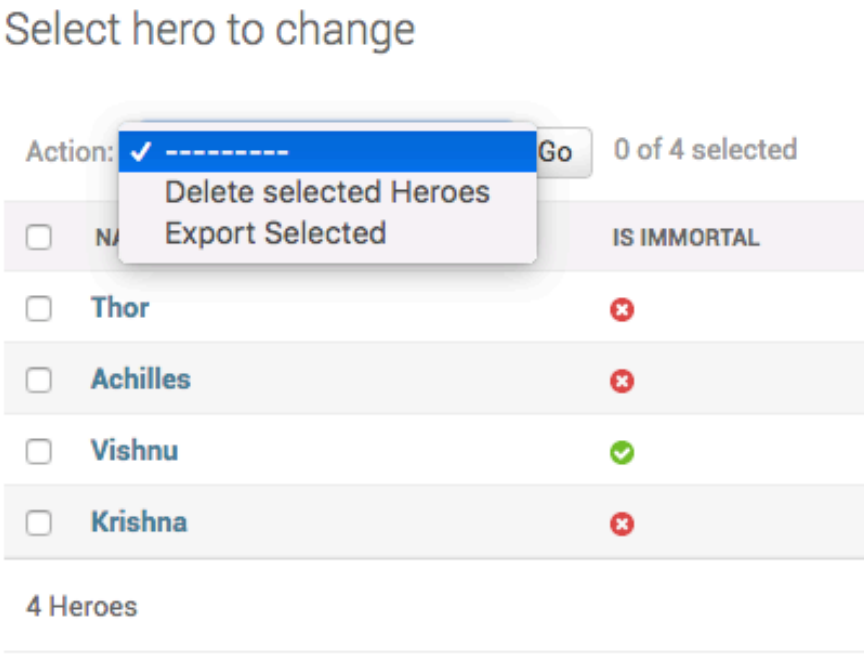
你被要求可以从 admin 导出 `Hero` 和 `VillanAdmin`，有很多第三方模块可以做到。但是不依赖其他项也非常容易。你将添加 admin action 到 `HeroAdmin` 和 `VillanAdmin`。

一个 admin action 可以总是有这样的函数签名（声明）`def admin_action(modeladmin, request, queryset)`: 或者你可以直接在 ModelAdmin 中作为一个方法。

```
class SomeModelAdmin(admin.ModelAdmin):

    def admin_action(self, request, queryset):
```

为了 `HeroAdmin` 添加 csv 导出，你可以像下面这样做:

```
actions = ["export_as_csv"]

def export_as_csv(self, request, queryset):
    pass

export_as_csv.short_description = "Export Selected"
```



添加了叫 export selected 的 action, 就像这样:
你将修改 `export_as_csv` 如下:

```
import csv
from django.http import HttpResponse
...

def export_as_csv(self, request, queryset):

    meta = self.model._meta
    field_names = [field.name for field in meta.fields]

    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename={}.csv'.format(meta)
    writer = csv.writer(response)

    writer.writerow(field_names)
    for obj in queryset:
        row = writer.writerow([getattr(obj, field) for field in field_names])

    return response
```

这将导出所有选中行，如果你注意到，`export_as_csv` 没有 `Hero` 的内容，你可以将方法提取成一个 mixin

类。

进行这些修改之后，你的代码会变成这样：

```
class ExportCsvMixin:
    def export_as_csv(self, request, queryset):

        meta = self.model._meta
        field_names = [field.name for field in meta.fields]

        response = HttpResponse(content_type='text/csv')
        response['Content-Disposition'] = 'attachment; filename={}.csv'.format(meta)
        writer = csv.writer(response)

        writer.writerow(field_names)
        for obj in queryset:
            row = writer.writerow([getattr(obj, field) for field in field_names])

        return response

    export_as_csv.short_description = "Export Selected"


@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    list_display = ("name", "is_immortal", "category", "origin", "is_very_benevolent")
    list_filter = ("is_immortal", "category", "origin", IsVeryBenevolentFilter)
    actions = ["export_as_csv"]

...


@admin.register(Villain)
class VillainAdmin(admin.ModelAdmin, ExportCsvMixin):
    list_display = ("name", "category", "origin")
    actions = ["export_as_csv"]
```

继承 ExportCsvMixin 类其他的 models 同样可以导出 csv

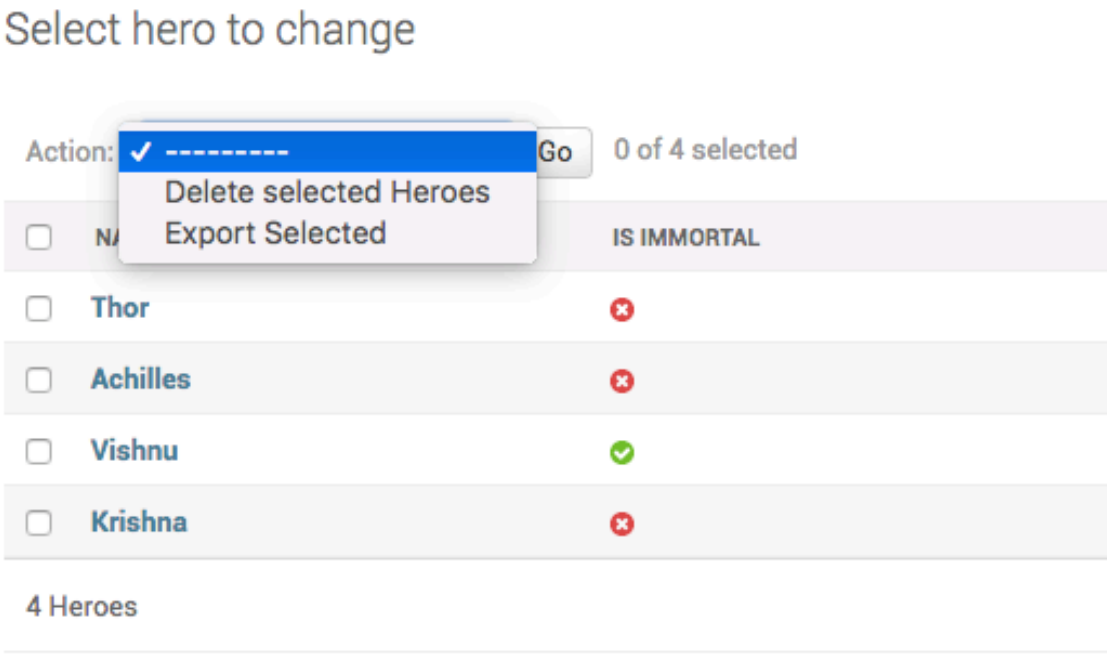
CHAPTER 15

3.Django admin 怎么移除删除 action

Django Admin 默认是有 *Delete Selected* action, 你被要求在 `Hero`admin 页面移除这个 action

`ModelAdmin.get_actions` 返回展示操作，通过重写方法移除 `delete_selected` 。你的代码将会改成这个样子:

```
def get_actions(self, request):
    actions = super().get_actions(request)
    if 'delete_selected' in actions:
        del actions['delete_selected']
    return actions
```



你的 admin 页面像这样：

你应该也读一下如何为一个 model 移除‘添加’/‘删除’按钮

CHAPTER 16

4. 如何在 Django Admin 列表页添加自定义操作按钮（不是 actions）

UMSRA 决定只要有足够的 kryptonite，所有英雄都是不朽的，然而，你想要改变他们的想法并说所有的英雄都是不朽的。

你被要求添加 2 个按钮，一个按钮可以使得所有英雄变得平凡，另一个按钮可以使得所有英雄变得不朽。因为它影响所有的英雄而与选择无关，所以需要一个单独的按钮，而不是 action 下拉框。

实现，我们需要修改 `HeroAdmin` 的模板以至于我们可以添加 2 个按钮

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    change_list_template = "entities/heroes_changelist.html"
```

然后我们重写 `get_urls`，并且在这个 model admin 添加 `set_immortal` 和 `set_mortal` 方法，他们将作为 2 个视图方法

```
def get_urls(self):
    urls = super().get_urls()
    my_urls = [
        path('immortal/', self.set_immortal),
        path('mortal/', self.set_mortal),
    ]
    return my_urls + urls

def set_immortal(self, request):
```

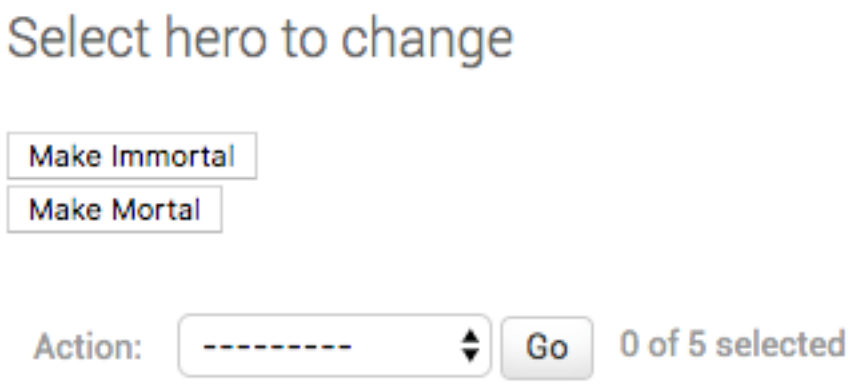
(continues on next page)

(continued from previous page)

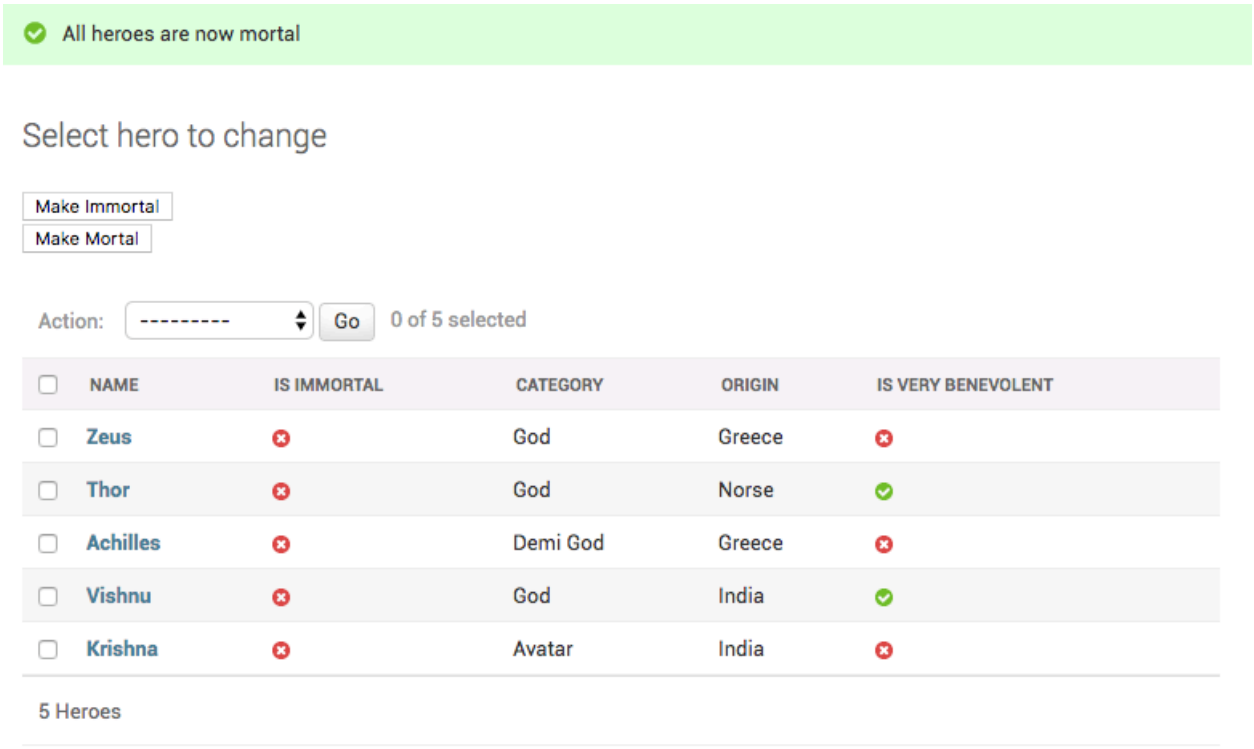
```
    self.model.objects.all().update(is_immortal=True)
    self.message_user(request, "All heroes are now immortal")
    return HttpResponseRedirect("../")

def set_mortal(self, request):
    self.model.objects.all().update(is_immortal=False)
    self.message_user(request, "All heroes are now mortal")
    return HttpResponseRedirect("../")
```

最后，我们创建继承 admin/change_list.html 的 entities/heroes_changelist.html 模板。

```
{% extends 'admin/change_list.html' %}

{% block object-tools %}
    <div>
        <form action="immortal/" method="POST">
            {% csrf_token %}
                <button type="submit">Make Immortal</button>
        </form>
        <form action="mortal/" method="POST">
            {% csrf_token %}
                <button type="submit">Make Mortal</button>
        </form>
    </div>
    <br />
    {{ block.super }}
{% endblock %}
```



当使用 *make_mortal* action 之后，所有的英雄将会变得平凡，如你看到这个消息


All heroes are now mortal
Select hero to change
Make Immortal
Make Mortal
Action:

Go
0 of 5 selected
NAME
IS IMMORTAL
CATEGORY
ORIGIN
IS VERY BENEVOLENT
Zeus
God
Greece
Thor
God
Norse
Achilles
Demi God
Greece
Vishnu
God
India
Krishna
Avatar
India
5 Heroes

CHAPTER 17

5. 如何使用 Django admin 上传 CSV

你被要求在 Hero admin 页面可以上传 csv，你需要添加一个链接到 Hero 更改列表页面，该页面将转到带有上传表单的页面。你将为 POST 操作编写一个处理程序，以从创建 csv 对象：

```
class CsvImportForm(forms.Form):
    csv_file = forms.FileField()

@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    change_list_template = "entities/heroes_changelist.html"

    def get_urls(self):
        urls = super().get_urls()
        my_urls = [
            ...
            path('import-csv/', self.import_csv),
        ]
        return my_urls + urls

    def import_csv(self, request):
        if request.method == "POST":
            csv_file = request.FILES["csv_file"]
            reader = csv.reader(csv_file)
```

(continues on next page)

(continued from previous page)

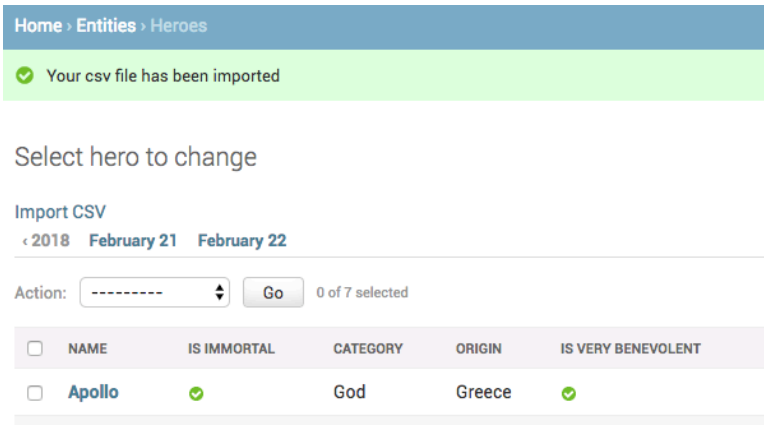
```
            # Create Hero objects from passed in data
            # ...
            self.message_user(request, "Your csv file has been imported")
            return redirect("..")
        form = CsvImportForm()
        payload = {"form": form}
        return render(
            request, "admin/csv_form.html", payload
        )
```

然后通过重写 admin/change_list.html 模板创建 entites/heroes_changelist.html 模块:

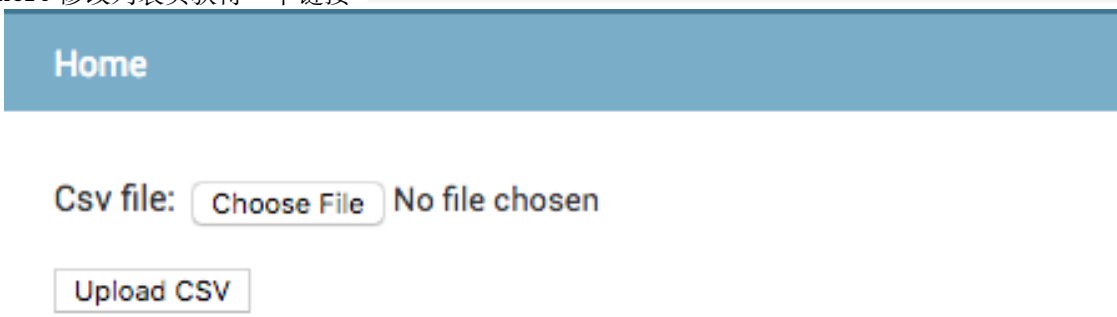
```
{% extends 'admin/change_list.html' %}

{% block object-tools %}
    <a href="import-csv/">Import CSV</a>
    <br />
    {{ block.super }}
{% endblock %}
```

最后创建 csv_form.html，像这样:

```
{% extends 'admin/base.html' %}

{% block content %}
    <div>
        <form action="." method="POST" enctype="multipart/form-data">
            {{ form.as_p }}
            {% csrf_token %}

                <button type="submit">Upload CSV</button>
        </form>
    </div>
    <br />

{% endblock %}
```



通过这些更改之后,你会在 `Hero` 修改列表页获得一个链接



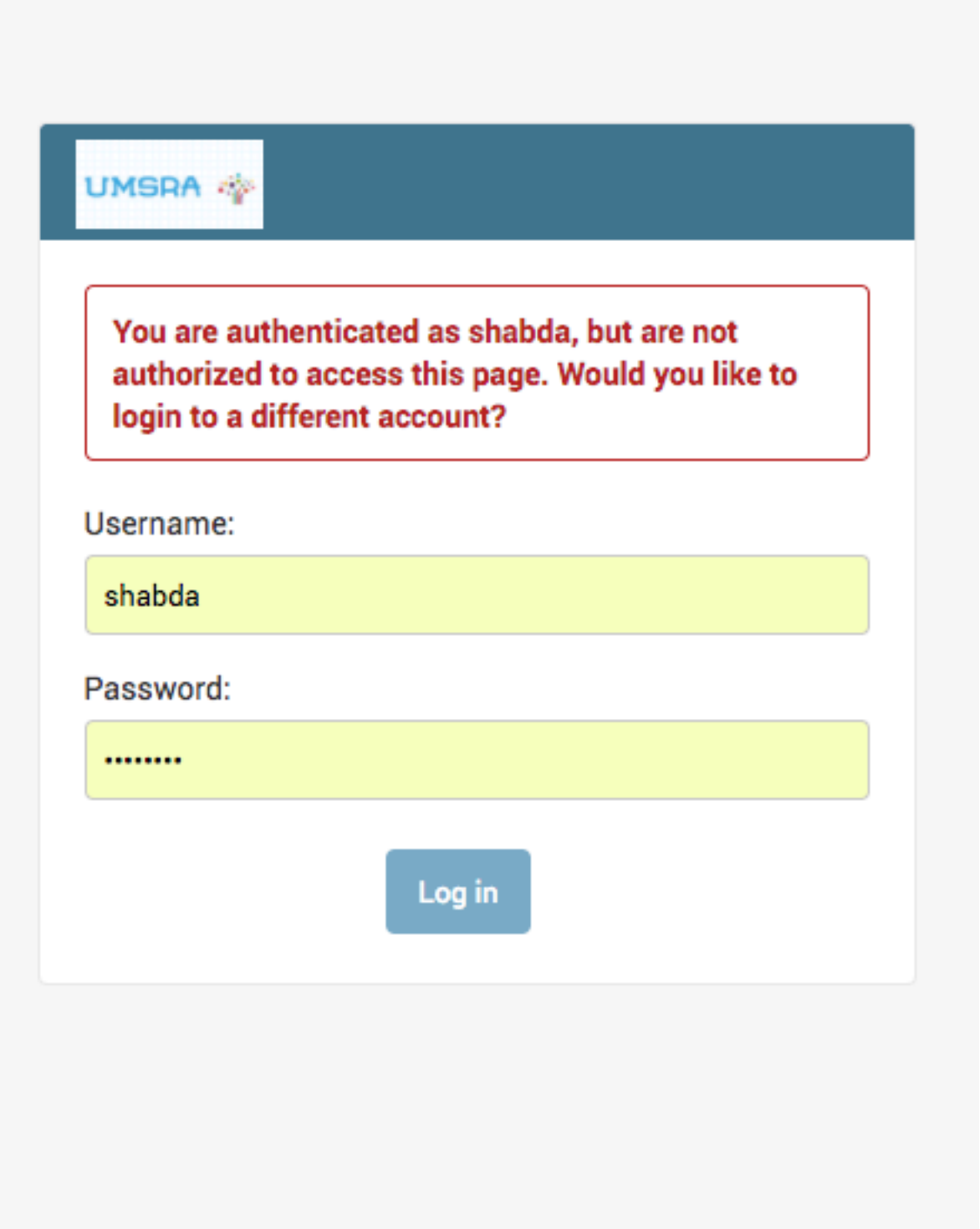
这个是上传 csv 页面像这样:

CHAPTER 18

1. 如何限定为特定用户的使用 Django admin

Django admin 允许标志为 `is_staff=True` 的用户访问，如果想让一个用户无法访问 Django admin ，你应该设置 `is_staff=False`

甚至这个用户是超级用户也是这样，non-staff 的用户访问 admin, 将会看到这个:

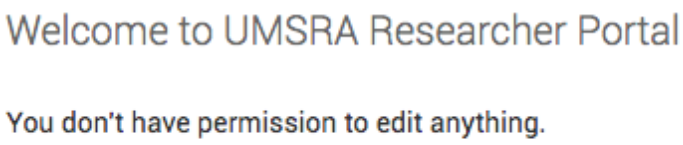

UMSRA
You are authenticated as shabda, but are not authorized to access this page. Would you like to login to a different account?
Username:
shabda
Password:
Log in

CHAPTER 19

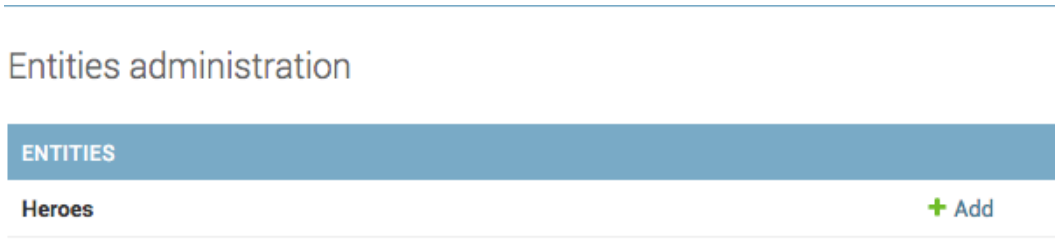
2. 如何限定部分 Django admin 权限

你可以使用权限系统启用和限制对 Django admin 特定部分的访问。添加模型时，默认情况下，Django 创建三个权限。添加，更改和删除

管理员使用这些权限来决定用户的访问权限。对于具有 is_superuser = False 且没有权限的用户，管理员看起



来像这样：
如果你添加一个权限 `user.user_permissions.add(Permission.objects.get(codename="add_hero"))`



这 admin 页面会展示成为这样子：
你可以通过更改以下方法来添加更复杂的逻辑来限制访问权限：

```
def has_add_permission(self, request):
    ...

def has_change_permission(self, request, obj=None):
    ...

def has_delete_permission(self, request, obj=None):
    ...

def has_module_permission(self, request):
    ...
```

CHAPTER 20

3.admin 中如何只能创建一个对象

UMSRA 管理员要求你将类别的数量限制为一个。他们希望每个实体都属于同一类别。

你可以通过下面这个做：

```
MAX_OBJECTS = 1

def has_add_permission(self, request):
    if self.model.objects.count() >= MAX_OBJECTS:
        return False
    return super().has_add_permission(request)
```

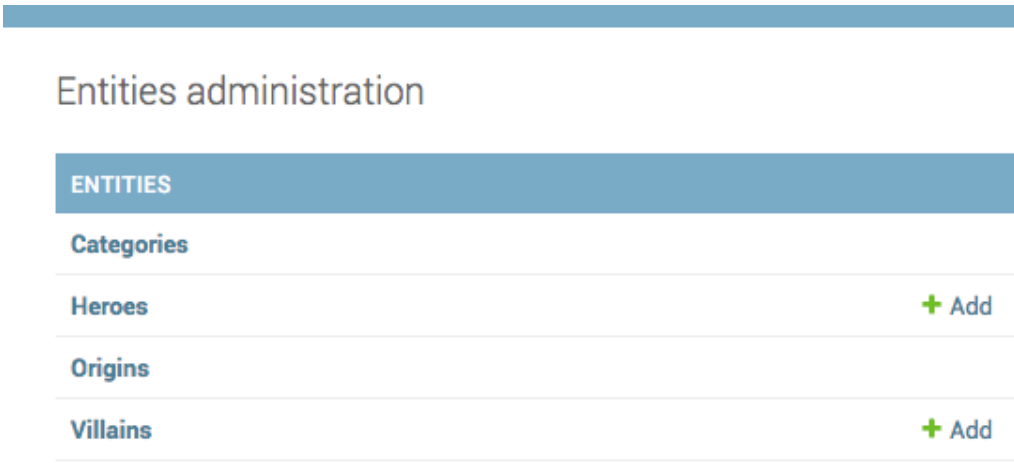
一旦创建一个对象，就会隐藏添加按钮，你可以设置 MAX_OBJECTS 设置为任何值，以确保最多可以容纳超过 MAX_OBJECTS 个对象。

CHAPTER 21

4. 如何使一个 model 移除’添加’/’删除’按钮

UMSRA 管理层已添加所有 `Category` 和 `Origin` 对象，并希望禁用任何进一步的添加和删除。他们要求你禁用“添加”和“删除”按钮。您可以通过在 Django 管理员中重写 `has_add_permission` 和 `has_delete_permission` 来做到这一点：

```
def has_add_permission(self, request):
    return False

def has_delete_permission(self, request, obj=None):
    return False
```



通过这些修改,你的 admin 就像这样:
注意移除 *Add* 按钮，这个添加和删除按钮也会从详情页面中删除，你还可以阅读如何从 Django Admin 中移除删除 action

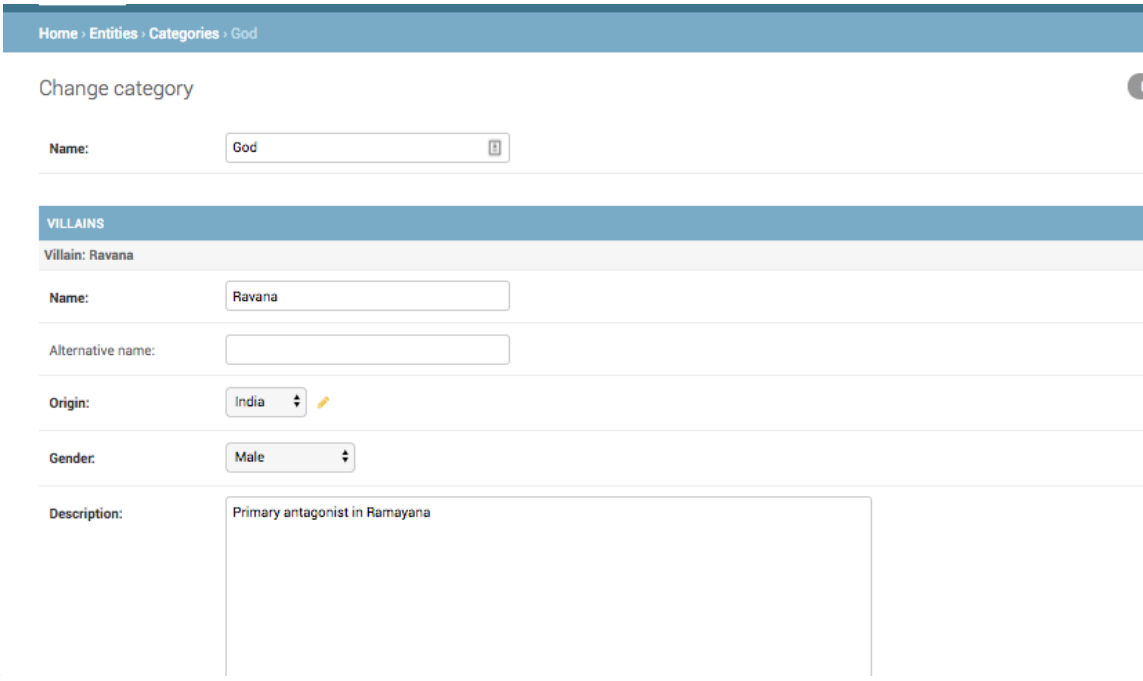
CHAPTER 22

1. 如何在一个 Django admin 页面编辑多个模型

为了达到这个目标，你需要使用内联你有 `Category` 模型，你需要在 Category 管理页面添加、编辑 `Villain` 模型

```
class VillainInline(admin.StackedInline):
    model = Villain

@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    ...

    inlines = [VillainInline]
```

你可以看到在 `Category` 页面有添加/编辑 `Villain`，如果内联的模型有很多字段，使用 `StackedInline`，也



可以使用 `TabularInile`。

CHAPTER 23

2. 如何添加一对一关系作为 admin 内联

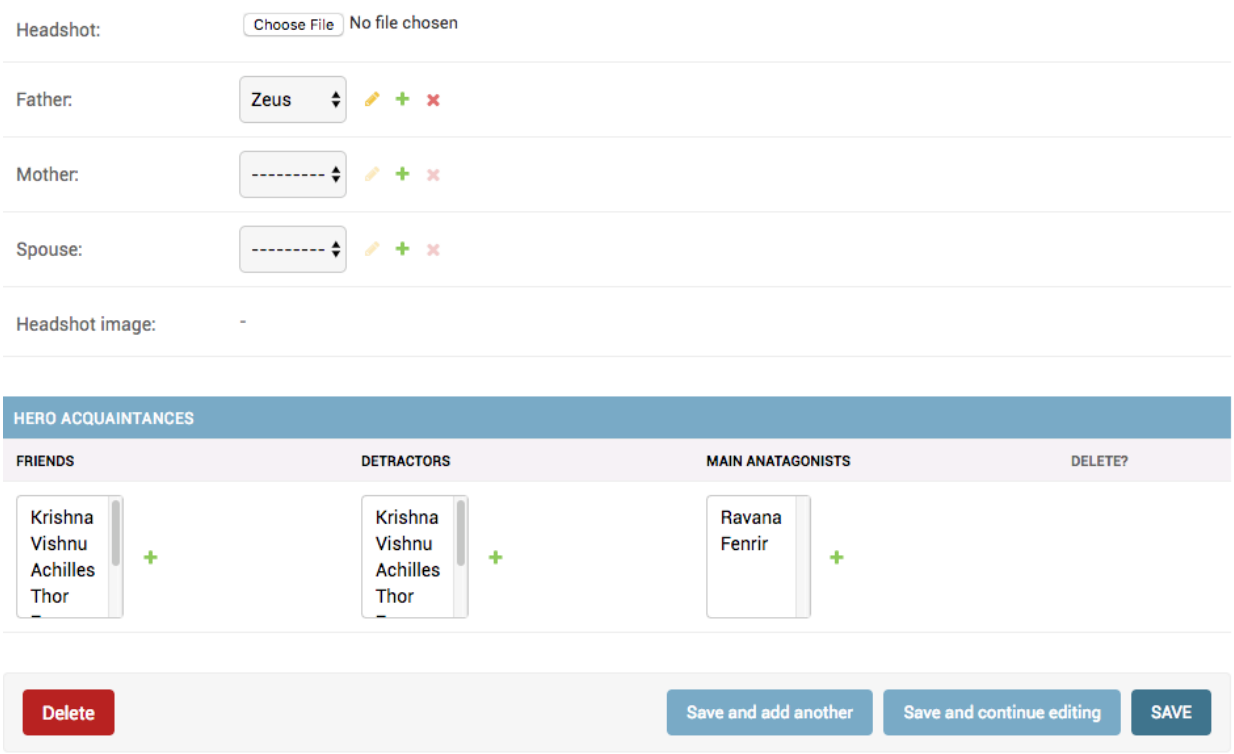
OneToOneFields 可以像外键一样设置为内联，但是，只能将 OneToOneFields 的一侧设置为 inline 模型

你有 `HeroAcquaintance` 模型，这个和 hero 有一对一关系，像这样：

```
class HeroAcquaintance(models.Model):
    "Non family contacts of a Hero"
    hero = models.OneToOneField(Hero, on_delete=models.CASCADE)
    ....
```

你可以添加作为内联，像这样：

```
class HeroAcquaintanceInline(admin.TabularInline):
    model = HeroAcquaintance


@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    inlines = [HeroAcquaintanceInline]
```


Headshot:
Choose File
No file chosen
Father:
Zeus
Mother:

Spouse:

Headshot image:
-
HERO ACQUAINTANCES
FRIENDS
DETRACTORS
MAIN ANATAGONISTS
DELETE?
Krishna
Vishnu
Achilles
Thor
Krishna
Vishnu
Achilles
Thor
Ravana
Fenrir
Delete
Save and add another
Save and continue editing
SAVE

CHAPTER 24

3. 如何在 Django admin 中添加嵌套内联

你定义了下面这样的模型:

```
class Category(models.Model):
    ...

class Hero(models.Model):
    category = models.ForeignKey(Catgeory)
    ...

class HeroAcquaintance(models.Model):
    hero = models.OneToOneField(Hero, on_delete=models.CASCADE)
    ...
```

你想要一个 admin 页面创建 `Category`，`Hero` 和 `HeroAcquaintance` 对象。但是，Django 不支持嵌套在多个级别上的外键内联或一对一关系。你有几个选择。你可以修改 `HeroAcquaintance` 模型。使其具有直接的 FK 到 `Category`，如下所示:

```
class HeroAcquaintance(models.Model):
    hero = models.OneToOneField(Hero, on_delete=models.CASCADE)
    category = models.ForeignKey(Category)

    def save(self, *args, **kwargs):
        self.category = self.hero.category
```

(continues on next page)

(continued from previous page)

```
        super().save(*args, **kwargs)
```

然后，你可以将 HeroAcquaintanceInline 附加到 CategoryAdmin，并获得一种嵌套内联。

另外，也有一些第三方 Django 应用允许嵌套内联。Github 或 DjangoPackages 搜索将找到适合你需求和品味的应用。

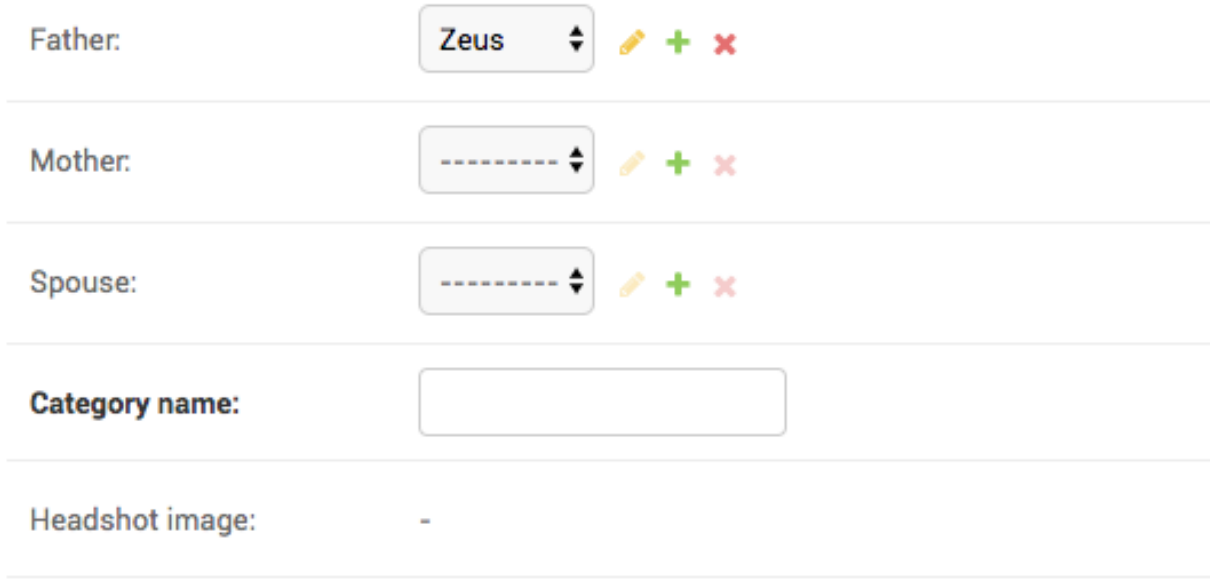
CHAPTER 25

4. 如何从 2 个不同的模型创建单个 Django admin

Hero 有外键 Category，所以你可以在 Hero 的管理页面选择 category, 如果你还想从 Hero 管理页面创建 Category 对象，你可以修改 Hero admin 的表单，并自定义 save_model 行为：

```
class HeroForm(forms.ModelForm):
    category_name = forms.CharField()

    class Meta:
        model = Hero
        exclude = ["category"]


@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    form = HeroForm
    ....

    def save_model(self, request, obj, form, change):
        category_name = form.cleaned_data["category_name"]
        category, _ = Category.objects.get_or_create(name=category_name)
        obj.category = category
        super().save_model(request, obj, form, change)
```

进行修改后，你的管理页面所示，允许从 Hero 管理页面创建或更新 Category。


Father:
Zeus
Mother:

Spouse:

Category name:
Headshot image:
-

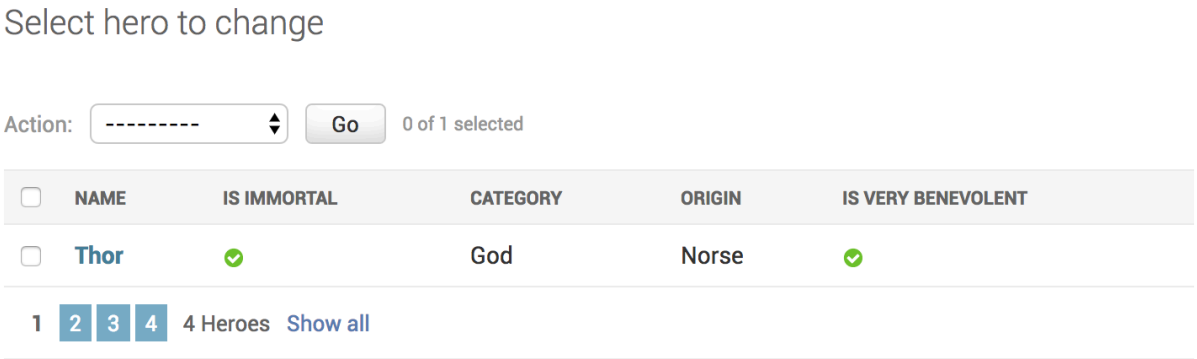
CHAPTER 26

1./2. 如何列表视图上显示大量行/不分页

你被要求在单个页面上看到英雄增加到 250（默认 100），你可以这样做：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    list_per_page = 250
```

你也可以将其设置为较小的值，如果将它设置为 1 `list_per_page=1` ，admin 页面就像这样：



如何使得不分页呢？

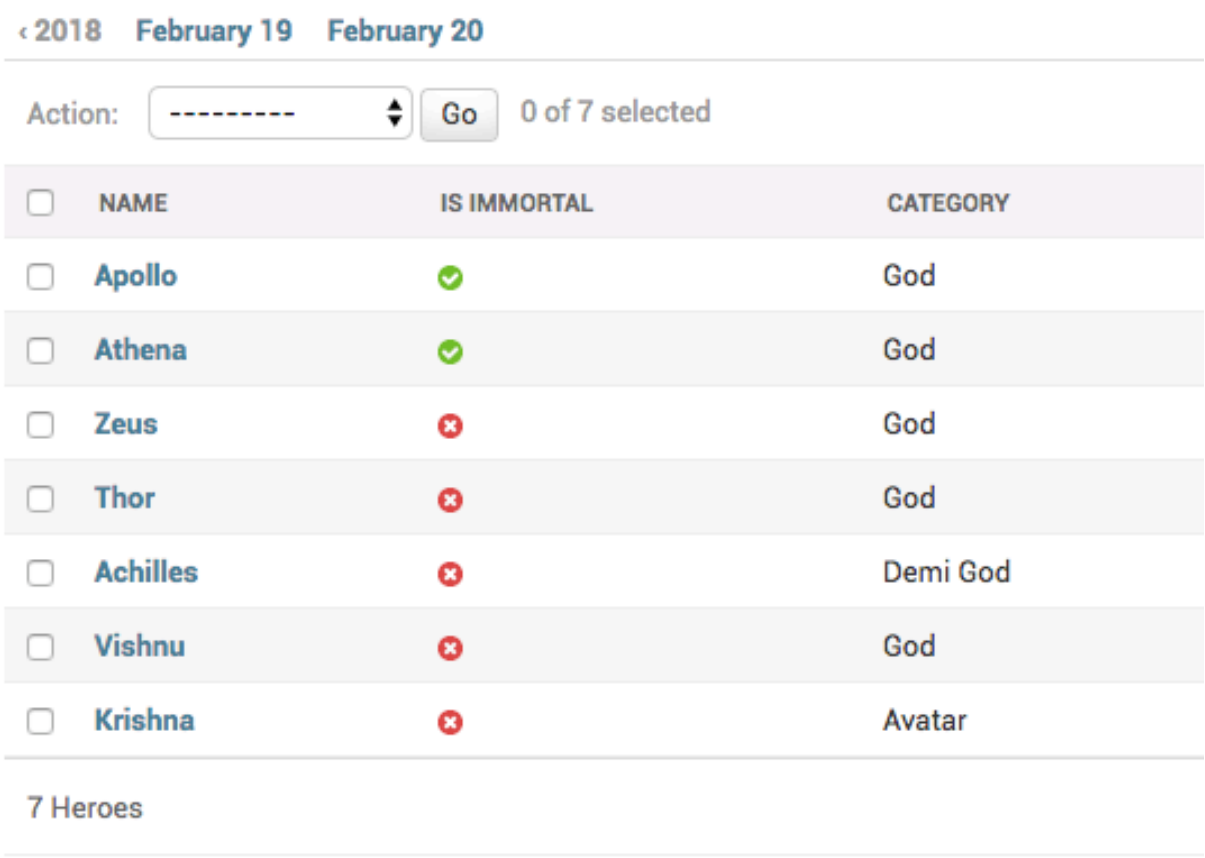
```
import sys
...

@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...

    list_per_page = sys.maxsize
```

CHAPTER 27

3.Django admin 如何添加基于日期的过滤

你可以通过设置 `date_hierarchy` 在任何日期字段添加基于日期的过滤：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    date_hierarchy = 'added_on'
```



看起来像这样：

对于很多对象，这会非常消耗性能，另外，你可以继承 SimpleListFilter 的子类，并仅允许过滤年份或月份。

CHAPTER 28

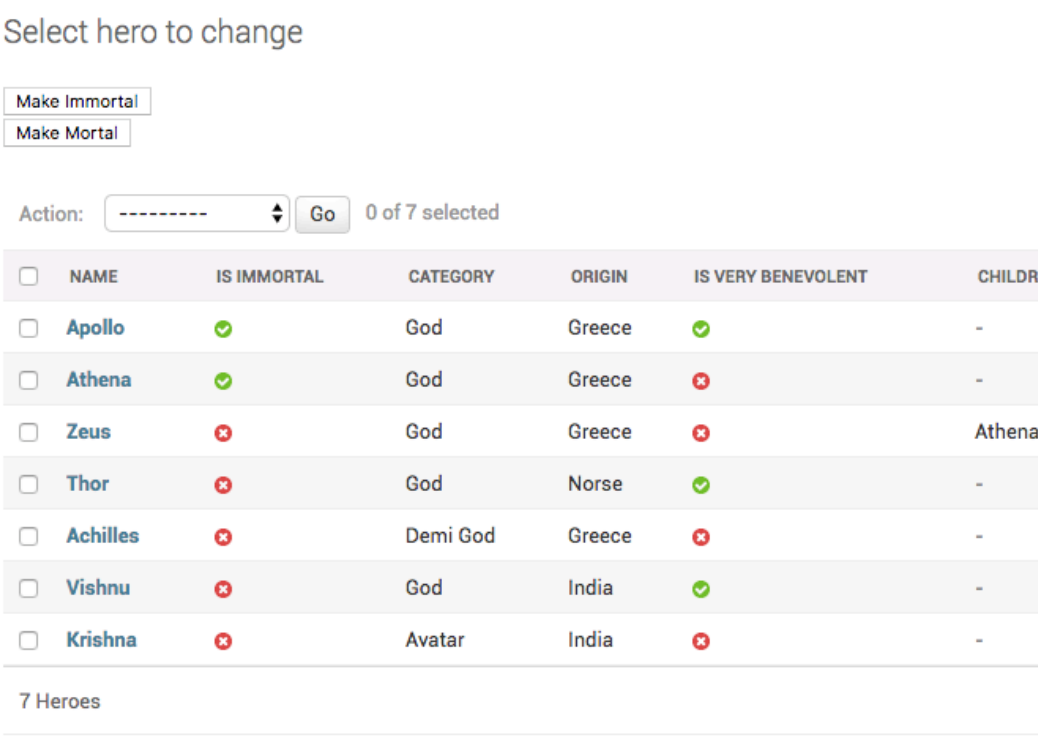
4. 如何在列表视图页面显示多对多或者反转 FK 字段

对于英雄，你可以使用以下字段跟踪到它的父亲

```
father = models.ForeignKey(
    "self", related_name="children", null=True, blank=True, on_delete=models.SET_NULL
)
```

你被要求在列表页面上显示每个英雄的孩子，Hero 对象有 `children` 反转 FK 属性，但是你不能将它添加到 `list_display` 里，你需要向 ModelAdmin 添加一个属性，并在 list_display 中使用它，你可以这样做：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...

    def children_display(self, obj):
        return ", ".join([
            child.name for child in obj.children.all()
        ])
    children_display.short_description = "Children"
```



你可以看到 children 列,想这样:

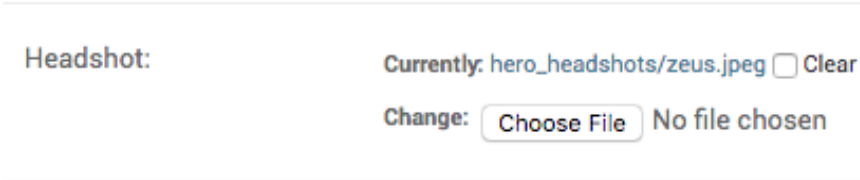
对于多对多关系，你可以使用相同的方法。你应该也阅读一下如何获取特定对象的 admin url

CHAPTER 29

1. 如何在 Django admin 显示 Imagefield 的图片

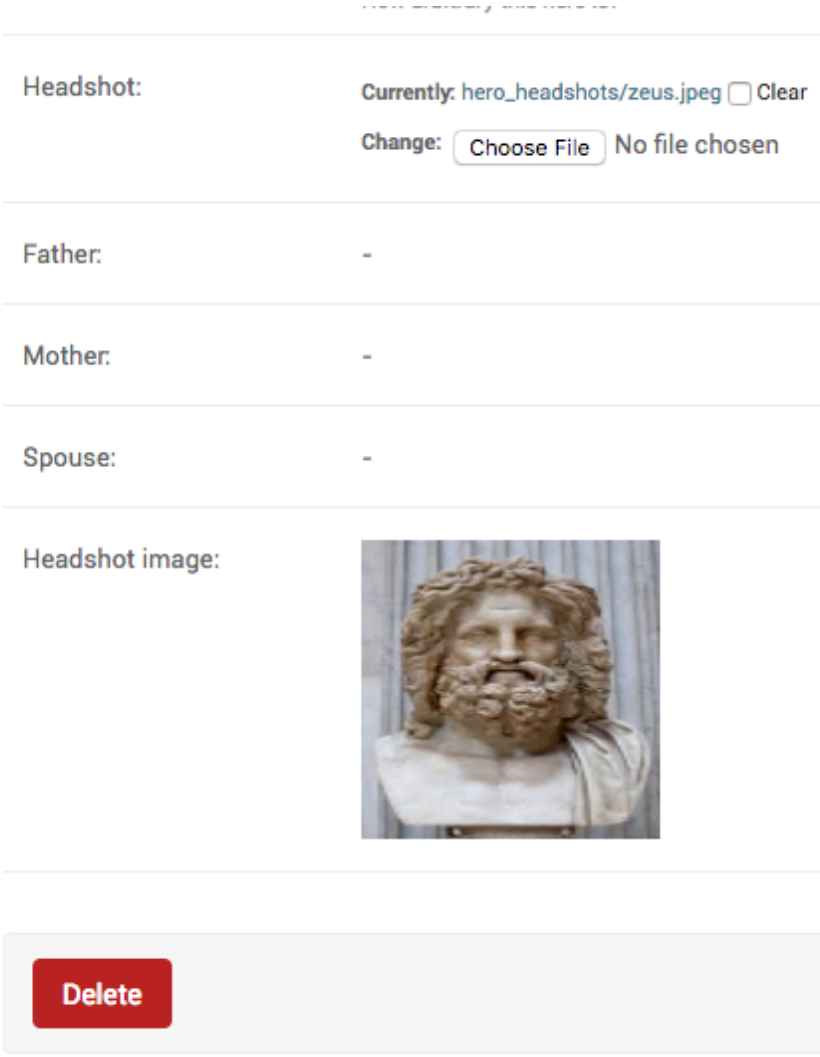
在你的 Hero 模型，你有一个图片字段

```
headshot = models.ImageField(null=True, blank=True, upload_to="hero_headshots/")
```



默认是这样显示的：
你被要求将其更改的实际图片也显示在页面上，你可以这样做：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):

    readonly_fields = [..., "headshot_image"]

    def headshot_image(self, obj):
        return mark_safe('<img src="{url}" width="{width}" height={height} />'.format(
            url = obj.headshot.url,
            width=obj.headshot.width,
            height=obj.headshot.height,
            )
    )
```



通过这些修改之后,你的 imagefiled 像这个样子:

CHAPTER 30

2. 保存时，如何将当前的用户和模型关联起来

Hero 模型有以下在字段

```
added_by = models.ForeignKey(settings.AUTH_USER_MODEL,
        null=True, blank=True, on_delete=models.SET_NULL)
```

当对象通过 admin 创建时，你需要 added_by 字段自动设置为当前用户。你可以这样做：

```
def save_model(self, request, obj, form, change):
    if not obj.pk:
        # Only set added_by during the first save.
        obj.added_by = request.user
    super().save_model(request, obj, form, change)
```

如果你想始终保存当前用户，就可以这样做:

```
def save_model(self, request, obj, form, change):
    obj.added_by = request.user
    super().save_model(request, obj, form, change)
```

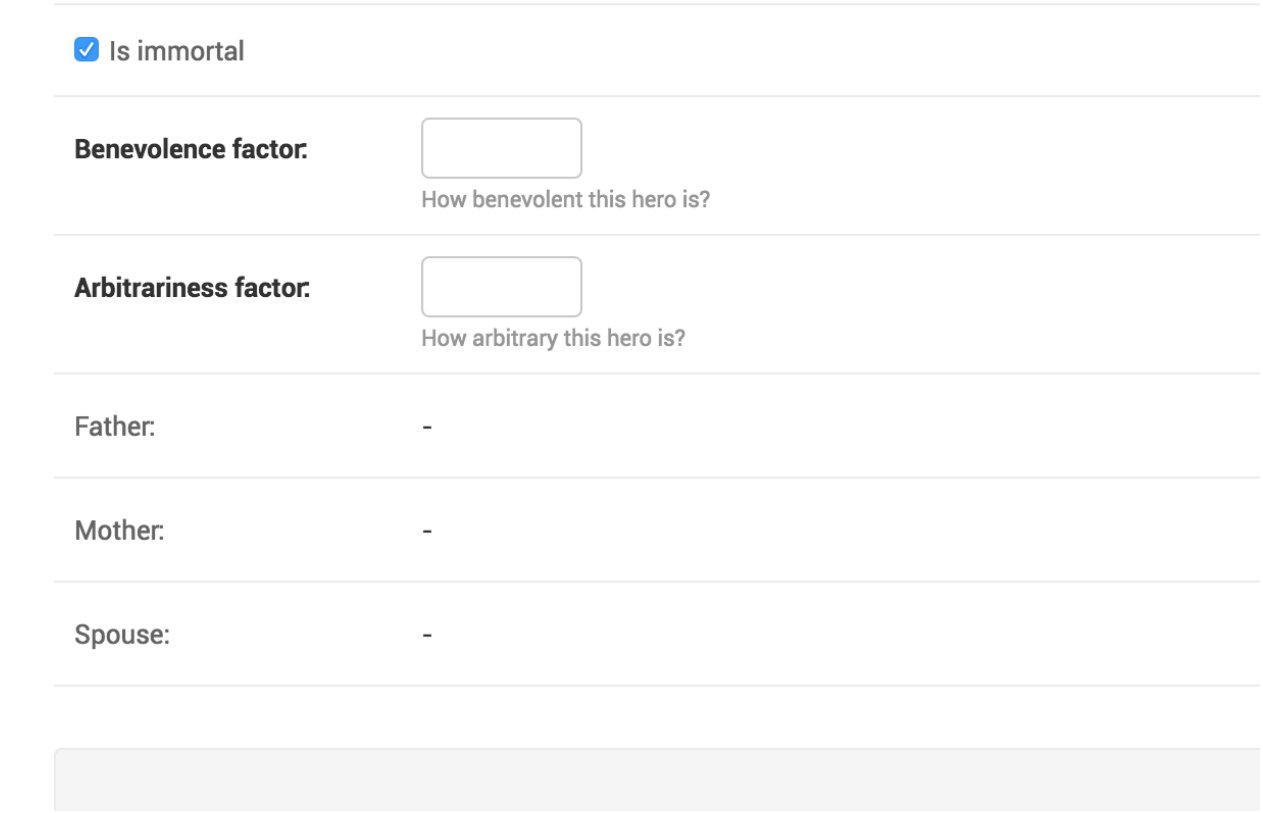
如果你还想隐藏 add_by 字段，让它不显示在更改表单上，可以这样做：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
```

(continues on next page)

(continued from previous page)

```
    ...
    exclude = ['added_by',]
```



CHAPTER 31

3. 如何标记一个字段只读？

UMSRA 临时决定停止追踪神话人物的家谱。你被要求将 `father`，`mother` 和 `spouse` 字段设为只读。你可以这样做：

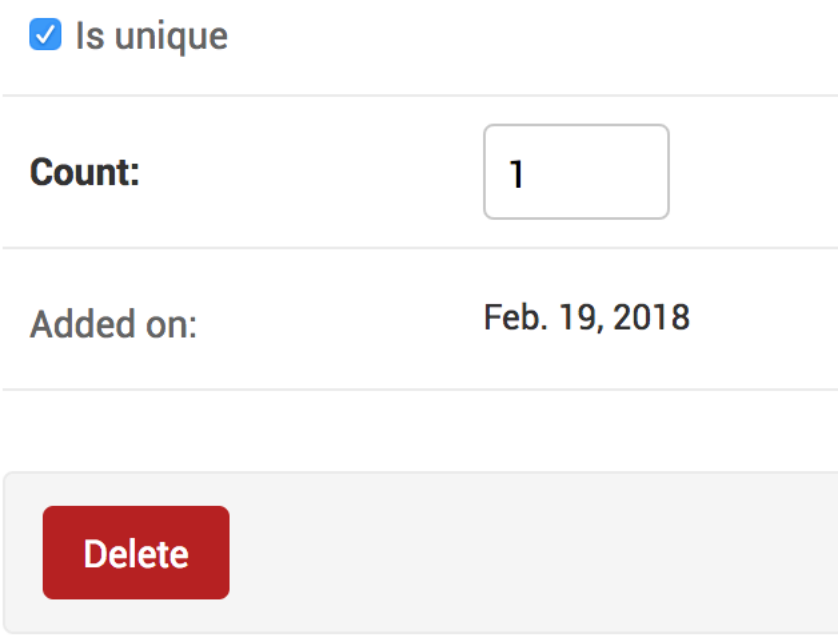
```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    readonly_fields = ["father", "mother", "spouse"]
```

CHAPTER 32

4. 如何显示不可编辑的字段

如果你的模型有一个 editable=False 字段，默认情况下，这个字段会隐藏在更改页面中，对应标记为 auto_now 或 auto_now_add 的任何字段也是一样的。因为这些字段上会设置 editable=False 如果你希望这些字段显示在更改页面上，可以将他们添加到只读字段。

```
@admin.register(Villain)
class VillainAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    readonly_fields = ["added_on"]
```



进行这些修改，Villain admin 页面看起来会像这样：

CHAPTER 33

5. 如何在创建时，字段可编辑，但是现有只读？

创建 Hero 对象后，你需要让 name 和 category 设置为只读，但是第一次写入的时候，这些字段需要可编辑

你可以重写 get_readonly_fields 方法，像这样：

```
def get_readonly_fields(self, request, obj=None):
    if obj:
        return ["name", "category"]
    else:
        return []
```

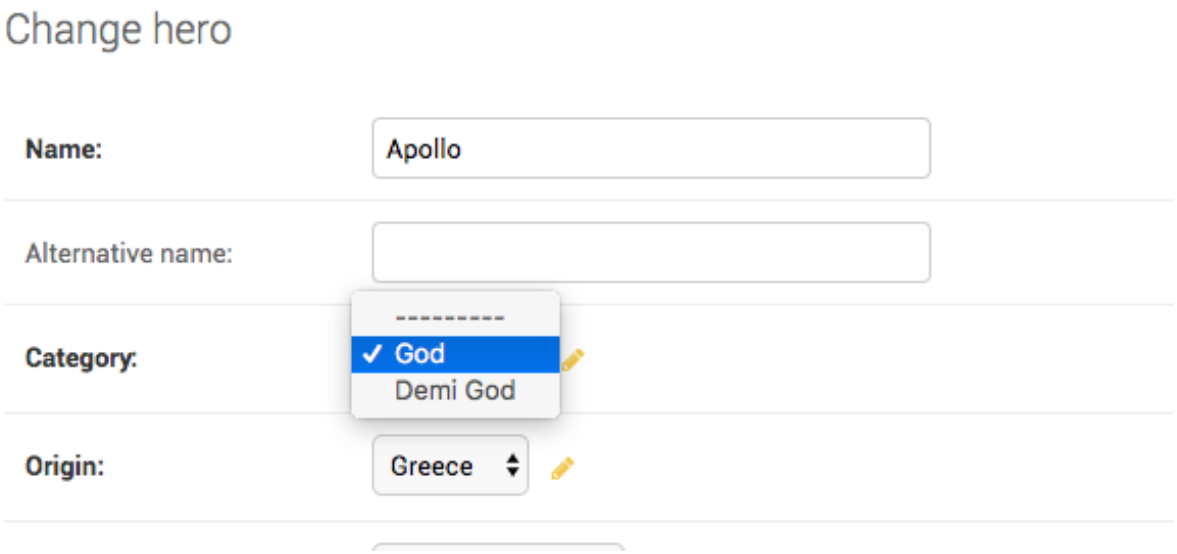
obj 在对象创建期间是 None，但是编辑期间，编辑对象期间是可编辑的。

CHAPTER 34

6. 如何在 Django-admin 过滤外键下拉

Hero 模型类有 Category 外键。所以，所有的 category 对象将出现在管理页面下拉列表中。另外，你如果只想查看一个子集，则 Django 允许你通过覆盖 formfield_for_foreignkey 对其进行自定义：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    def formfield_for_foreignkey(self, db_field, request, **kwargs):
        if db_field.name == "category":
            kwargs["queryset"] = Category.objects.filter(name__in=['God', 'Demi God'])
        return super().formfield_for_foreignkey(db_field, request, **kwargs)
```


Change hero
Name:
Apollo
Alternative name:
Category:

God
Demi God
Origin:
Greece

CHAPTER 35

7. 如何管理一个有很多外键对象的模型？

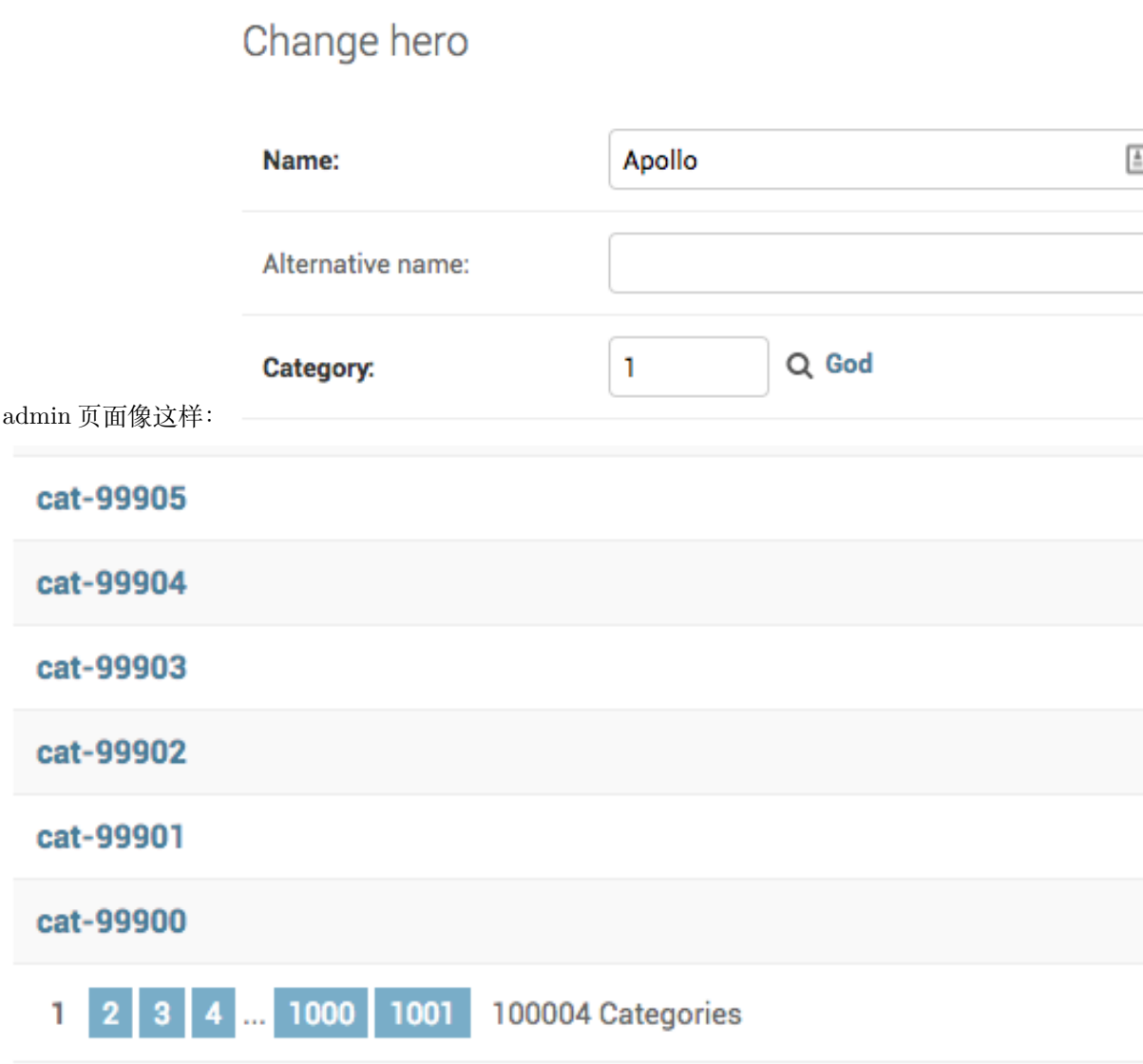
你可以创建大量的 category, 像这样：

```
categories = [Category(**{"name": "cat-{}".format(i)}) for i in range(100000)]
Category.objects.bulk_create(categories)
```

现在 `Category` 超过了 100000 个对象，当你在 `Hero`admin 页面的时候，你将会有一个超过 100000 选择的下拉框。这会让页面变得很慢，并且下拉框难于使用。你可以通过设置 `raw_id_fields` 来改变 admin 的管理方式。

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...
    raw_id_fields = ["category"]
```

修改了之后,Hero 的 admin 页面像这样:



弹出窗口像这样:

CHAPTER 36

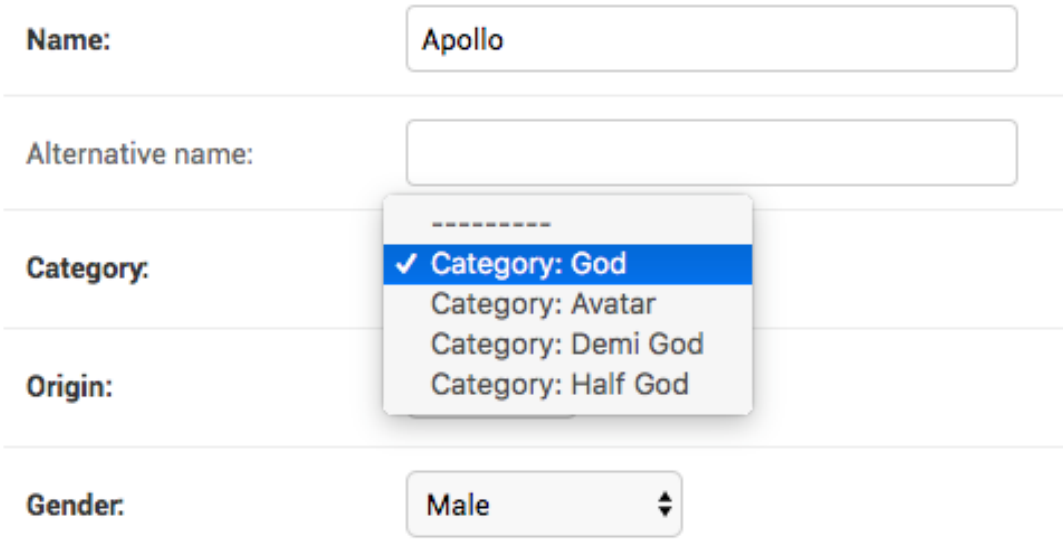
8. 如何在下拉菜单中更改外键显示的文本

Hero 有 Category 外键，在下拉框里，显示的只是名字，但是你想显示”Category:name”格式。你可以在 Category 修改 __str__ 方法，但是你只希望在 admin 中进行更改，你可以通过创建一个 forms.ModelChoiceField 的子类，并且自定义 label_from_instance 方法。

```
class CategoryChoiceField(forms.ModelChoiceField):
     def label_from_instance(self, obj):
         return "Category: {}".format(obj.name)
```

之后我们可以重写 category 的以用到这个字段 formfield_for_foreignkey

```
def formfield_for_foreignkey(self, db_field, request, **kwargs):
    if db_field.name == 'category':
        return CategoryChoiceField(queryset=Category.objects.all())
    return super().formfield_for_foreignkey(db_field, request, **kwargs)
```



你的 admin 看起来会想这样：

CHAPTER 37

9. 如何在修改页面添加自定义的按钮

Villain 有一个叫 is_unique 的字段

```
class Villain(Entity):
    ...
    is_unique = models.BooleanField(default=True)
```

你想在 Villain 页面添加一个叫 “Make Unique” 按钮让这个 Villian 唯一，其他的同名的 villian 都应该被删除。

你先扩展 change_form 添加一个按钮

```
{% extends 'admin/change_form.html' %}

{% block submit_buttons_bottom %}
    {{ block.super }}
    <div class="submit-row">
            <input type="submit" value="Make Unique" name="_make-unique">
    </div>
{% endblock %}
```

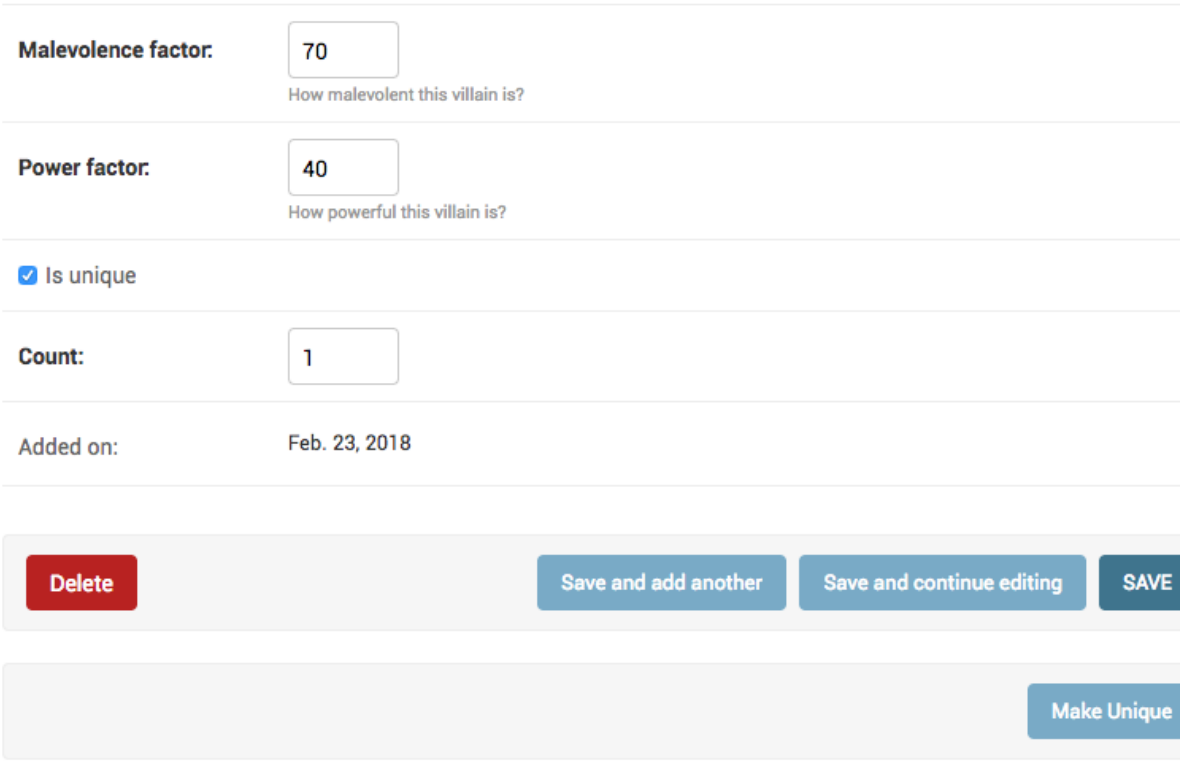
然后你可以覆盖 response_change 并将模板连接到 VillainAdmin

```
@admin.register(Villain)
class VillainAdmin(admin.ModelAdmin, ExportCsvMixin):
```

(continues on next page)

(continued from previous page)

```
    ...
    change_form_template = "entities/villain_changeform.html"


    def response_change(self, request, obj):
        if "_make-unique" in request.POST:
            matching_names_except_this = self.get_queryset(request).filter(name=obj.
↪name).exclude(pk=obj.id)
            matching_names_except_this.delete()
            obj.is_unique = True
            obj.save()
            self.message_user(request, "This villain is now unique")
            return HttpResponseRedirect(".")
        return super().response_change(request, obj)
```



你现在的 admin 页面：

CHAPTER 38

1. 如何获取特殊对象的 django admin url

你有显示每个英雄孩子的孩子列，你被要求将每个孩子的链接放到更改页面，你可以这样做：

```
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    ...

    def children_display(self, obj):
        display_text = ", ".join([
            "<a href={}>{}</a>".format(
                    reverse('admin:{}_{}_change'.format(obj._meta.app_label, obj._meta.
↪model_name),
                    args=(child.pk,)),
                child.name)
             for child in obj.children.all()
        ])
        if display_text:
            return mark_safe(display_text)
        return "-"
```

`reverse('admin:{}_{}_change'.format(obj._meta.app_label, obj._meta.model_name), args=(child.pk,))` 会给出对象的 url

其他选项：删除：

`reverse('admin:{}_{}_delete'.format(obj._meta.app_label, obj._meta.model_name),`

args=(child.pk,))

历史记录: reverse('admin:{}_{}_history'.format(obj._meta.app_label, obj._meta.model_name), args=(child.pk,))

CHAPTER 39

2. 如何两次添加模型到 Django admin 中

你需要将 `Hero` 模型添加两次到 Django admin 中，一个作为常规管理区域，另外一个是只读区域。如果你重复注册相同的模型两次：

```
admin.site.register(Hero)
admin.site.register(Hero)
```

你将会获取以下错误：

```
raise AlreadyRegistered('The model %s is already registered' % model.__name__)
```

解决的方案是子类化 `Hero` 模型，作为 ProxyModel:

```
# In models.py
class HeroProxy(Hero):

    class Meta:
        proxy = True

...
# In admin.py
@admin.register(Hero)
class HeroAdmin(admin.ModelAdmin, ExportCsvMixin):
    list_display = ("name", "is_immortal", "category", "origin", "is_very_benevolent")
```

(continues on next page)

(continued from previous page)

```
    ....


@admin.register(HeroProxy)
class HeroProxyAdmin(admin.ModelAdmin):
    readonly_fields = ("name", "is_immortal", "category", "origin",
        ...)
```

CHAPTER 40

3. 如何重写 Django admin 的保存行为

ModelAdmin 有一个 save_model 方法，这个方法用于创建和更新模型对象，你可以自定义 admin 保存行为。

Hero 模型有下面的字段：

```
added_by = models.ForeignKey(settings.AUTH_USER_MODEL,
        null=True, blank=True, on_delete=models.SET_NULL)
```

如果你想 Hero 更新的时候，总是保存当前用户，你可以这样做：

```
def save_model(self, request, obj, form, change):
    obj.added_by = request.user
    super().save_model(request, obj, form, change)
```

CHAPTER 41

4. 如何在 Django-admin 添加数据库视图

你有这样创建的数据库视图：

```
create view entities_entity as
    select id, name from entities_hero
    union
    select 10000+id as id, name from entities_villain
```

它具有 `Hero` 和 `Villain` 所有的名称，Villain 的 id 设置为 10000+id, 因为我们不打算横过 10000 名 Heros

```
sqlite> select * from entities_entity;
1|Krishna
2|Vishnu
3|Achilles
4|Thor
5|Zeus
6|Athena
7|Apollo
10001|Ravana
10002|Fenrir
```

然后添加一个 `managed=False` 模型:

```
class AllEntity(models.Model):
```

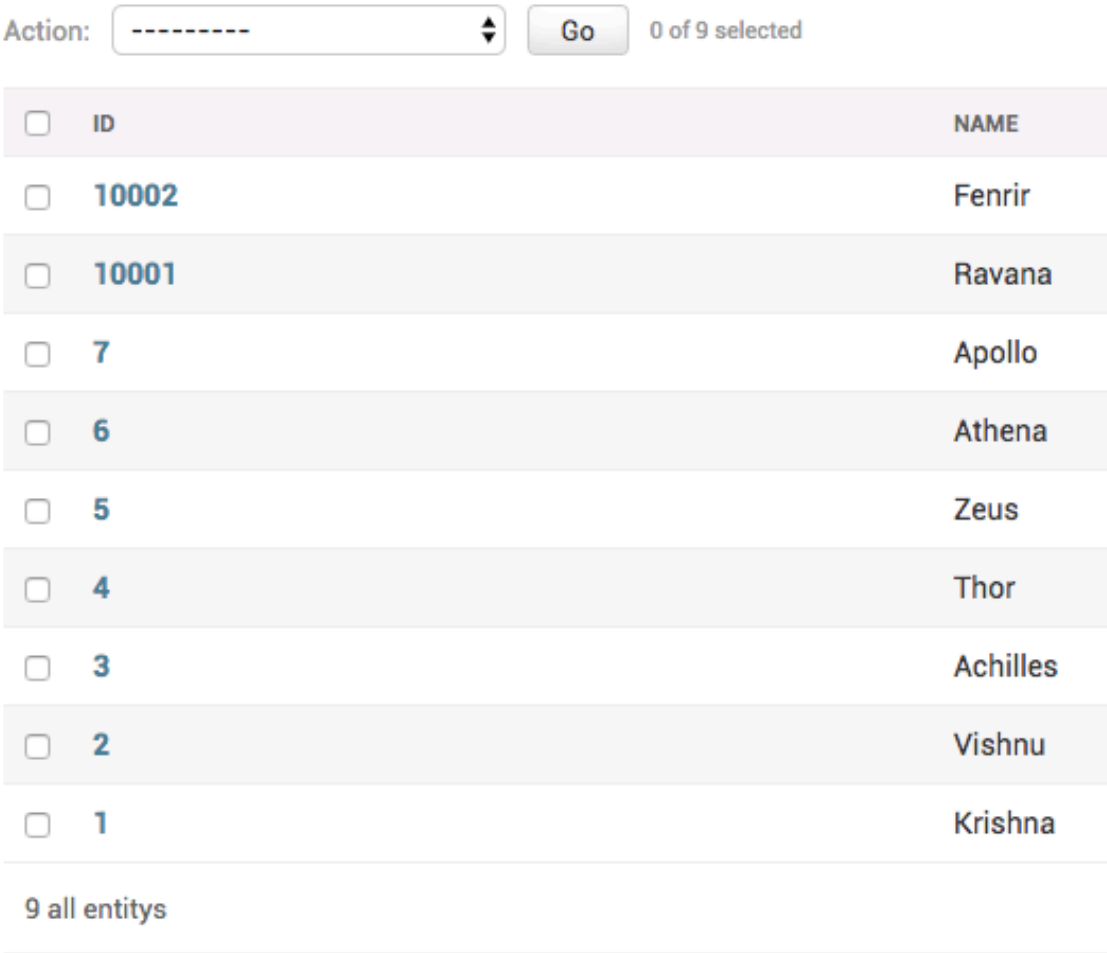
(continues on next page)

(continued from previous page)

```
    name = models.CharField(max_length=100)

    class Meta:
        managed = False
        db_table = "entities_entity"
```

并添加到 admin:

```
@admin.register(AllEntity)
class AllEntiryAdmin(admin.ModelAdmin):
    list_display = ("id", "name")
```



你的 admin 看起来像这样:

CHAPTER 42

Indices and tables

- genindex
- modindex
- search