

Django是一个python语言实现的、MVC架构模式的开源web框架。Django以其优雅和简约使web应用开发变得极其容易。本文作者也是被其设计理念所折服，于是立马书写一篇，为Django的开源推广助一臂之力。在此，本文以如何搭建网站搜索为例，带领大家领略一下其魅力。

了解Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django的主要目的是简便、快速的开发数据库驱动的网站。它强调代码复用，多个组件可以很方便的以“插件”形式服务于整个框架，Django有许多功能强大的第三方插件，你甚至可以很方便的开发出自己的工具包。这使得Django具有很强的可扩展性。它还强调快速开发和DRY(Do Not Repeat Yourself)原则。

Django的主要模块设计：

- 对象关系映射 (ORM,object-relational mapping)：以Python类形式定义你的数据模型，ORM将模型与关系数据库连接起来，你将得到一个非常容易使用的数据库API，同时你也可以在Django中使用原始的SQL语句。
- URL 分派：使用正则表达式匹配URL，你可以设计任意的URL，没有框架的特定限定。像你喜欢的样灵活。
- 模版系统：使用Django强大而可扩展的模板语言，可以分隔设计、内容和Python代码。并且具有可继承性。
- 表单处理：你可以方便的生成各种表单模型，实现表单的有效性

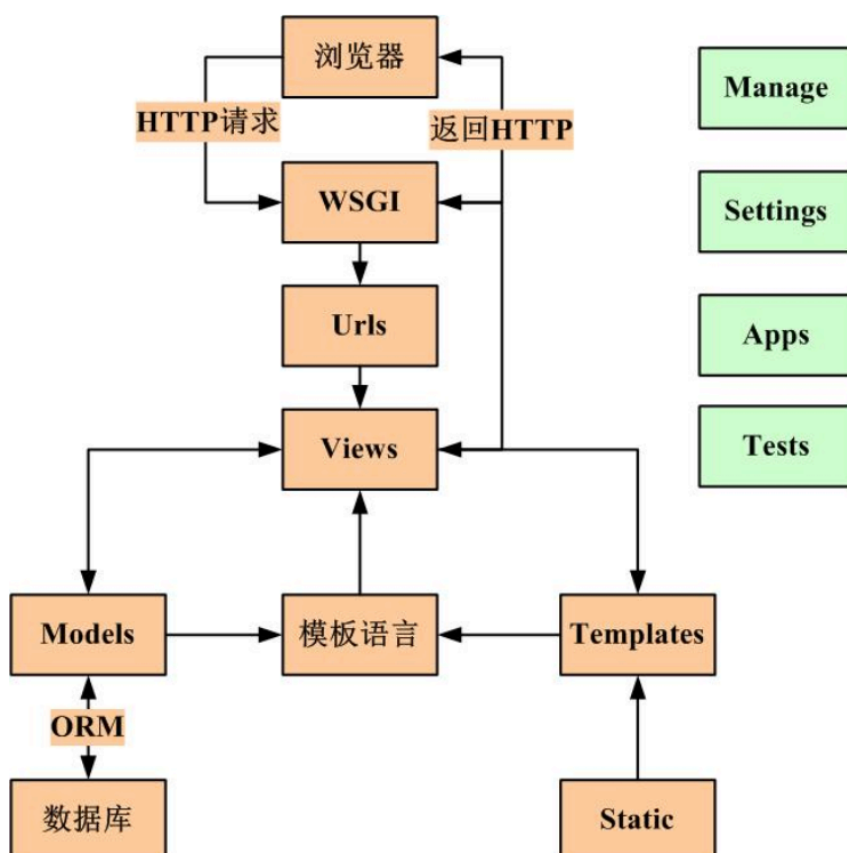
检验。可以方便的从你定义的模型实例生成相应的表单。

- Cache系统：可以挂在内存缓冲或其它的框架实现超级缓冲 —— 实现你所需要的粒度。
- 会话(session)，用户登录与权限检查，快速开发用户会话功能。
- 国际化：内置国际化系统，方便开发出多种语言的网站。
- 自动化的管理界面：不需要你花大量的工作来创建人员管理和更新内容。Django自带一个ADMIN site,类似于内容管理系统

关于Django的介绍，更多信息参见[这里](#)。

Django项目组织

开始动手之前，了解下主要业务流程，通过其项目目录结构设计能够较好理解其处理过程。



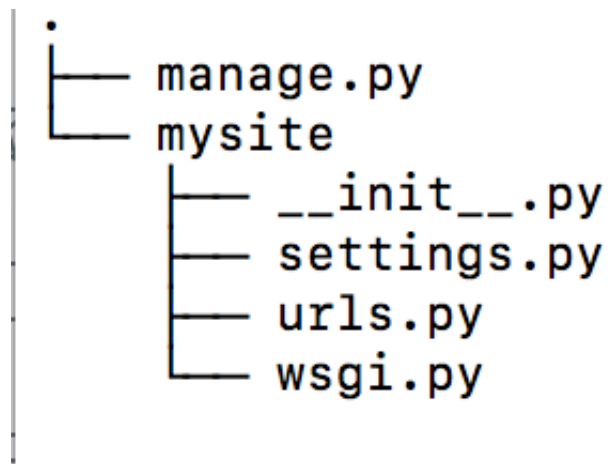
Django提供了完备的控制台命令，诸如用户创建项目和开发调试、测试等。使用起来超级方便，比如使用以下命令即可创建一个带有基本结构的可使用的web框架。

#安装django

```
pip3 install -e django
```

#创建一个project

```
django-admin startproject mysite
```



启动服务器：

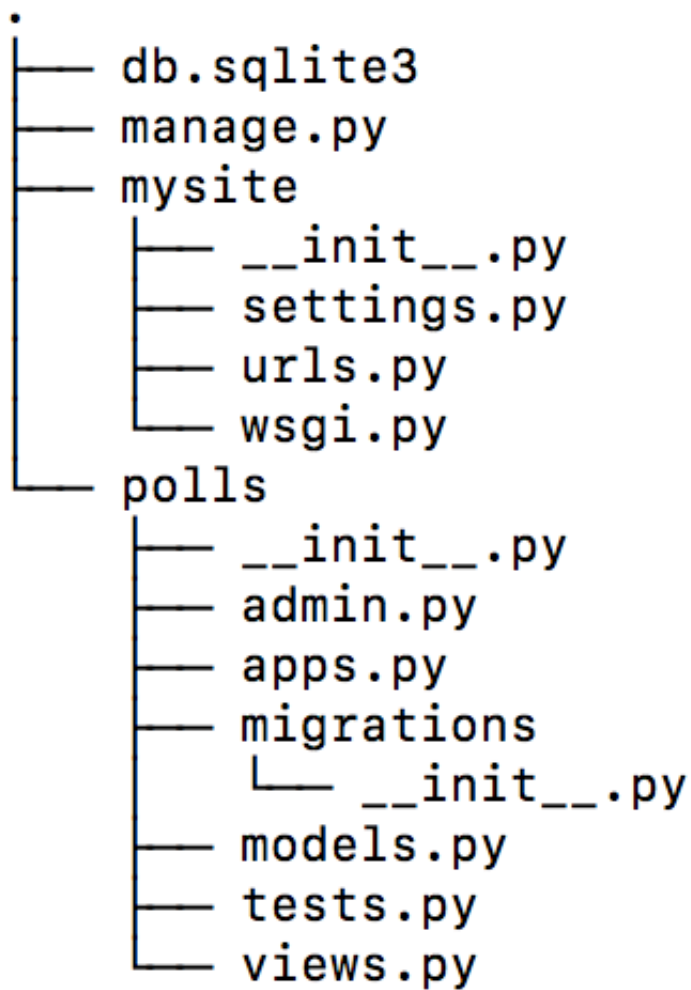
```
python3 manage.py runserver
```

浏览器访问<http://127.0.0.1:8000>

#在项目下增加一个app

```
python3 manage.py startapp polls
```

目录变成如下结构：



在mysite/settings.py中添加

```
INSTALLED_APPS = [  
    'polls',  
]
```

看到没，models.py和views.py就是模型和视图，urls.py就是url映射,settings.py是配置文件！

开始HelloWorld

1. 不用模板

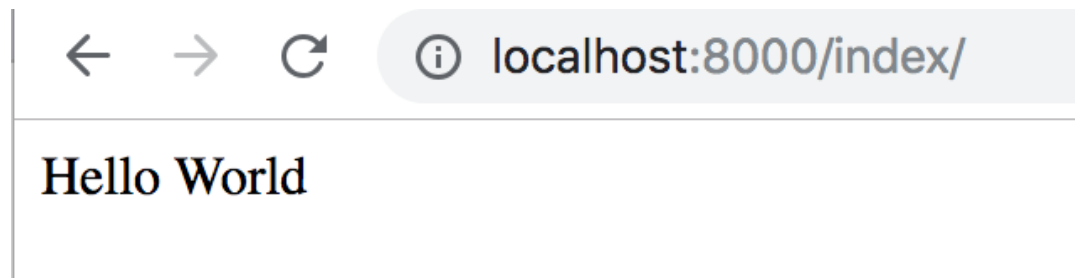
在mysite/views.py加一个函数：

```
def index(request):  
    return HttpResponse(u'Hello World')
```

在mysite/urls.py中添加

```
urlpatterns = [
    path('index/', views.index),
]
```

然后重启应用，浏览器访问http://127.0.0.1:8000/index。



2. 使用模板

在mysite/views.py修改index函数：

```
def index(request):
    context={'hellowho','world'}
    return render(request, 'polls/hello.html',context)
```

在polls目录下创建hello.html，路径为：

polls/templates/polls/hello.html

内容为

```
<h2>hello {{hellowho}}</h2>
```

刷新浏览器即可看到效果，666！

搭建网站搜索

创建全文索引，首先指定需要索引的数据模型，配置相应的分词策略准备索引创建，然后提供外部访问查询接口，并且提供关键字的高亮展示。这个过程在传统网站中实现是个很繁琐的过程，然而基于django实现只需要简单几个步骤，怎么做到的呢？

首先祭出几件大杀器：

- [haystack](#) django的开源搜索框架，该框架支持 [Solr](#), [Elasticsearch](#), [Whoosh](#), [*Xapian*](#) 搜索引擎。

- [Whoosh](#) 一个由纯Python实现的全文搜索引擎。
- [Jieba](#) 中文分词

0. 指定数据模型

在polls/models.py中添加模型：

```
class Product(models.Model):
    title=models.CharField(max_length=100,blank=True,verbose_name='标题')
    mainPhoto = models.ImageField(upload_to='products',verbose_name='上传主图')
    pub_date = models.DateTimeField('发布日期',default = timezone.now)
    content=UEditorField(u'内容',width=600, height=300, toolbars="full",
    imagePath="images/", filePath="", upload_settings=
    {"imageMaxSize":1204000},
    settings={},blank=True)

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

定义了一个Product，主要属性：标题，封面图，发布日期，产品介绍（支持富文本）。

1. 添加Product

进入admin 后台管理，增加几条Product记录

← → ↻ localhost:8000/admin/polls/product/22/change/

修改 产品内容页

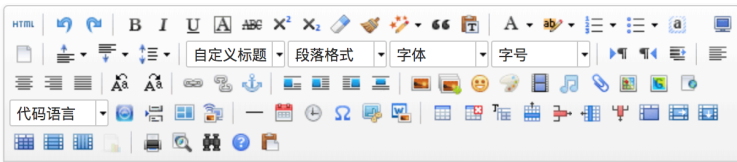
标题: test 标题测试标题中的粉刺效果

选择一级类目: 产品分类

选择二级类目: 电镀设备及周边产品

上传主图: 目前: products/shenjia.jpg
修改: Choose File No file chosen

发布日期: 日期: 2018/11/27 今天
时间: 09:26 现在

内容: 
content 内容测试内容中的中文分词效果

2. 创建索引

- 安装haystack，并添加到mysite/settings.py:

```
INSTALLED_APPS = [  
    'haystack',  
    'polls',  
]
```

- 在项目根目录mysite下面创建索引目录whoosh_index（默认名字），并在mysite/settings.py中配置分词引擎，这里先用自带的英文分词引擎whoose。

```
HAYSTACK_CONNECTIONS = {  
    'default': {  
        'ENGINE': 'haystack.backends.whoosh_backend.WhooshEngine',  
        'PATH': os.path.join(os.path.dirname(__file__), 'whoosh_index'),  
    },  
}
```

- 在polls app下面创建search_indexes.py文件（默认文件名）:

```
import datetime
from haystack import indexes
from .models import Product
```

```
class ProductIndex(indexes.SearchIndex, indexes.Indexable):
    #如果使用一个字段设置了document=True，则一般约定此字段名为text，索引的数据都在text字段里了
    text = indexes.CharField(document=True, use_template=True)
    #包含id属性
    id = indexes.IntegerField(model_attr='id')

    def get_model(self):
        return Product

    def index_queryset(self, using=None):
        """Used when the entire index for model is updated."""
        return self.get_model().objects.all()
```

- 创建数据模板：

在app polls下面创建product_text.txt文件，如果路径目录不存在，创建之。

templates/search/indexes/polls/product_text.txt

内容为：

```
{{ object.title }}
{{ object.content }}
```

这个数据模板的作用是对Product.title, Product.content这两个字段建立索引，当检索的时候会对这些字段做全文检索。

- 创建搜索展现模板

在app polls下面创建templates/search/search.html

```
{% extends 'polls/base.html' %}
{% block content %}
{% load highlight %}

{% if query %}
```



```

<h5>搜索结果</h5>
{% for result in page.object_list %}
    <p>
        <a href="{% url 'polls:product_detail' result.object.id %}">{% highlight
result.object.title with query max_length 300 %}</a>
        <br>
        <span>
            {% highlight result.object.content with query max_length 300 %}
        </span>
    </p>
{% empty %}
    <p>没有找到相关内容.</p>
{% endfor %}

{% if page.has_previous or page.has_next %}
    <div>
        {% if page.has_previous %}<a href="?q={{ query }}&page={{
page.previous_page_number }}">{% endif %}<< 前一页{% if
page.has_previous %}</a>{% endif %}
        {% if page.has_next %}<a href="?q={{ query }}&page={{
page.next_page_number }}">{% endif %}后一页 >>{% if page.has_next %}</a>
{% endif %}
    </div>
{% endif %}
{% else %}
    {# 顶部导航栏重新输入搜索关键词，然后回车搜索 #}
{% endif %}
{% endblock %}

```

- 创建搜索提交表单

```

<form class="navbar-form navbar-right" role="search" method="get"
id="searchform" action="{% url 'haystack_search' %}">
    <input type="search" class="form-control" name="q" placeholder="搜
索" required>

```

</form>

- 配置搜索请求url

最后一步就是在mysite/urls.py中添加

```
urlpatterns = [  
    path('search/', include('haystack.urls')),  
]
```

其中haystack.urls内容为：

```
from django.conf.urls import url  
from haystack.views import SearchView
```

```
urlpatterns = [  
    url(r'^$', SearchView(), name='haystack_search'),  
]
```

重建索引

```
python3 manage.py rebuild_index
```

重启服务器，浏览器访问 <http://127.0.0.1:8000/search>

- 自动更新索引

当数据库内容变化，自动更新索引，在mysite/settings.py配置

```
HAYSTACK_SIGNAL_PROCESSOR =  
'haystack.signals.RealtimeSignalProcessor'
```

- 设置每页显示行数

haystack自带分页功能，可以在mysite/settings.py配置每页显示行数

```
HAYSTACK_SEARCH_RESULTS_PER_PAGE = 5
```

支持中文分词

搜索会发现，搜索英文单词能够搜到，但是中文搜不到，因为默认的whoosh分词对中文支持不够好，需要更换其分词库。

- 实现中文分词引擎whoosh_cn_backend.py

我们把源码whoosh_backend.py cp到polls目录下，改名为

whoosh_cn_backend.py,修改如下：

引入ChineseAnalyzer

```
from jieba.analyse import ChineseAnalyzer
```

将原来的`analyzer=StemmingAnalyzer()` 替换为`analyzer=ChineseAnalyzer()`

该源文件位于本地python 安装包下面，以mac系统为例，其路径为：

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/haystack/backends/whoosh_backend.py
```

- 配置分词引擎

将mysite/settings.py中 ENGINE 修改为

```
'polls.whoosh_cn_backend.WhooshEngine'
```

```
HAYSTACK_CONNECTIONS = {
```

```
    'default': {
```

```
        # 'ENGINE': 'haystack.backends.whoosh_backend.WhooshEngine',
```

```
        'ENGINE': 'polls.whoosh_cn_backend.WhooshEngine',
```

```
        'PATH': os.path.join(os.path.dirname(__file__), 'whoosh_index'),
```

```
    },
```

```
}
```

- 重建索引

```
python3 manage.py rebuild_index
```

- 展示样式调整

haystack提供了很多展示标签，常用的如高亮展示 highlight，设置段落展示最大长度 max_length 等，使用方法如下：

```
{% highlight result.object.content with query max_length 300 %}
```

最后感受下搜索效果



#系统环境

pip3 freeze >requirements.txt看下主要依赖包版本:

#python==3.6.6

Django==2.1.1

django-bootstrap3==11.0.0

django-haystack==2.8.1

DjangoUeditor==1.8.143

jieba==0.39

Pillow==5.2.0

pyquery==1.4.0

python-dateutil==2.7.3

Whoosh==2.7.4

深入了解Django:

[Django官网](#)

Python Web开发最易懂的WSGI协议, 到底包含哪些内容?

<https://django-haystack.readthedocs.io/en/master/tutorial.html>

<https://whoosh.readthedocs.io/en/latest/indexing.html>