x全文检索

官方文档

> 全文检索就是针对所有内容进行动态匹配搜索的概念
>
> 针对特定的关键词进行**建立索引**并**精确匹配**取出搜索结果，并且达到性能优化的目的

- 为啥要有全文检索

> 最常见的全文检索就是我们在数据库中进行的模糊查询
>
> 但是模糊查询是针对整体内容的一个动态匹配过程，在数据量较大的情况下匹配效率极低
>
> 常规项目中数据量一般都比较多并且内容繁杂，所以正常的项目搜索功能中很少会使用模糊查询进行操作
>
> 如果你开发的项目用户量较少并且项目数据较少，那么此时模糊查询可以是你值得考虑的选项

- `django`使用啥进行全文检索

> `Python`提供了各种模块进行全文检索，最常见的是`haystack`模块
>
> 该模块设计为支持`whoosh`、`solr`、`Xapian`、`Elasticsearch`四种全文检索引擎后端
>
> 使用`haystack`模块，不用更改代码，直接切换引擎，可以极大的减少代码量
>
> `haystack`属于一种**全文检索**的框架

- `whoosh`

> 纯`Python`编写的全文搜索引擎，是目前最快的`python`所编写的检索引擎，虽然性能比不上`solr`、`Xapian`、`Elasticsearch`等；但是无二进制包，程序不会莫名其妙的崩溃，对于小型的站点，`whoosh`已经足够使用

- `solr`

> `Solr`是一个高性能，采用`Java5`开发，基于`Lucene`的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎
>
> `Lucene`：不是一个完整的**全文检索引擎**，是一个全文检索引擎的**架构**，提供了完整的查询引擎和索引引擎，`Lucene`的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能

- `Xapian`

> `Xapian`是一个用`C++`编写的全文检索程序，他的作用类似于`Java`的`lucene`

- `Elasticsearch`

> `ElasticSearch`是一个基于`Lucene`的搜索服务器它提供了一个分布式多用户能力的全文搜索引擎，基于`RESTful web`接口
>
> `Elasticsearch`是用Java开发的，并作为`Apache`许可条款下的开放源码发布，是当前流行的企业级搜索引擎。该引擎常设计用于云计算中；能够达到**实时搜索**，**稳定**，**可靠**，**快速**，**安装**使用方便

中文分词

> `whoosh`作为一个全文搜索引擎模块
>
> 分词功能和检索功能已经非常强大，但是针对中文的处理还是比较欠缺
>
> 可以通过`Jieba`模块重写分词操作，支持`whoosh`对中文的强大操作

- 安装中文分词模块

```
pip install jieba
```

- 除了`jieba`分词，现在还有很多付费的中文分词模块

> 中科院计算所NLPIR
>
> `ansj`分词器
>
> 哈工大的`LTP`
>
> 清华大学`THULAC`
>
> 斯坦福分词器
>
> `Hanlp`分词器
>
> 结巴分词
>
> `KCWS`分词器(字嵌入`+Bi-LSTM+CRF`)
>
> `ZPar`
>
> `IKAnalyzer`

安装

- 首先安装`HayStack`框架以及`whoosh`搜索引擎

```
pip install django-haystack
pip install whoosh
```

settings配置

- 添加`haystack`应用到项目的`settings`文件下的`app`部分

```
INSTALLED_APPS = [
    'django.contrib.admin',
    ...
    'haystack',
]
```

- 添加搜索引擎，这里使用`whoosh`引擎

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

#这里使用django的信号机制，在数据表发生改动时自动更新whoosh的查询索引
HAYSTACK_SIGNAL_PROCESSOR = 'haystack.signals.RealtimeSignalProcessor'
```

> 这里要注意的是，我们使用的引擎为`whoosh_cn_backend`
>
> 本身的`whoosh`引擎名为：`whoosh_backend`
>
> `whoosh_cn_backend`将在接下来我们对安装目录下的引擎文件复制修改得来

- 在项目的路由文件下配置查询的路由映射

```
from django.urls import include,re_path
urlpatterns = [
    path('admin/', admin.site.urls),
    re_path('^search/',include('haystack.urls')),
]
```

> 当查询条件被提交时，会跳转至`search`路由
>
> 并且查询条件会作为`get`请求时的连接参数传入，参数`key`值为`q`

创建索引文件

- 接下来，在需要被搜索的`app`下建立`search_indexes.py`文件，该文件名不许变更

```
#app.models.py
class User(models.Model):
    # 用户表
    name = models.CharField(
        max_length=50,
        verbose_name='昵称'
        )
    account = models.CharField(max_length=50,verbose_name='账号',unique=True)
    passwd = models.CharField(max_length=50,verbose_name='密码')
    def __str__(self):
        return self.name
```

```
#app.search_indexes.py
from haystack import indexes
from . import models

class UserIndex(indexes.SearchIndex, indexes.Indexable):
    text = indexes.CharField(document=True, use_template=True)

    def get_model(self):
        return models.User # 当前模型文件下需要被检索的模型类

    def index_queryset(self, using=None):
        return self.get_model().objects.all()
```

> 该类为索引类，类名为模型类的名称`+Index`：比如模型类为`People`,则这里类名为`PeopleIndex`
>
> `get_model`函数用来获取当前索引类所关联的模型类，这里我们关联上面的`User`类对象
>
> `text=indexes.CharField`语句指定了将模型类中的哪些字段建立索引，而`use_template=True`说明后续我们将通过一个**数据模板**文件来指明需要检索的字段
>
> `document=True`
>
> 为什么要创建索引：索引就像是一本书的目录，可以为读者提供更快速的导航与查找

创建模板数据文件

- 创建数据模板文件

> 数据模板文件路径：`templates/search/indexes/yourapp/note_text.txt`
>
> 放在任何一个你的`Django`能搜索到的模板文件夹`template`下面均可，这个文件主要确定要检索的字段，为他们建立索引
>
> 文件名必须为要索引的类名`_text.txt`，比如这里我们检索的类名是`User`，那么对应的数据模板文件名为`user_text.txt`，文件名小写即可

```
#template.search.indexes.people.user_text.txt
{{ object.name }}
{{ object.account }}
{{ object.online_time }}
```

> 在数据模板文件中使用模板语法，写入需要建立索引的字段，这里我们将模型类中`name`、`account`以及`online_time`字段设置索引，当检索时会对这三个字段去做全文检索

> 接下来创建一个搜索结果展示页面

检索结果模板页面

- 创建检索结果展示页面

> 检索结果展示页面，需要在固定的目录路径下进行模板页面的编写
>
> 路径为：`templates/search/`

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
{% if query %}
    <h3>搜索结果如下：</h3>
    {% for result in page.object_list %}
        {{ result.object.name }}
        <br>
        {{ result.object.account }}
        <br>
        {{ result.object.online_time }}
        <br>
    {% empty %}
        <p>没找到</p>
    {% endfor %}

    {% if page.has_previous or page.has_next %}
        <div>
            {% if page.has_previous %}
                <a href="?q={{ query }}&amp;page={{ page.previous_page_number }}">
                    上一页
                </a>
            {% endif %}

            {% if page.has_next %}
                <a href="?q={{ query }}&amp;page={{ page.next_page_number }}">
                    下一页
                </a>
            {% endif %}
        </div>
    {% endif %}
{% endif %}
</body>
</html>
```

> 这个模板页面中已经自带了分页功能，可以按照需求修改

- 创建检索模板页面内容

> 还需要有一个表单，提交检索信息

```
<form method='get' action="/search/" >
    <input type="text" name="q">
    <input type="submit" value="查询">
</form>
```

> 这部分检索的模板页面内容可以在你的项目中进行添加，查询方式为`get`，并且检索输入的表单框`name`属性必须为`q`

中文分词配置

- 接下来，需要创建有关中文检索的配置文件，这里的配置文件创建为全局

> 进入到`python`的安装目录下，比如我的目录为：`C:\Python37\Lib\site-packages\haystack\backends`
>
> 在该路径下创建名为`ChineseAnalyzer.py`的中文分词文件

```
import jieba
from whoosh.analysis import Tokenizer, Token

class ChineseTokenizer(Tokenizer):
    def __call__(self, value, positions=False, chars=False,
                 keeporiginal=False, removestops=True,start_pos=0, start_char=0, mode='',
**kwargs):
        t = Token(positions, chars, removestops=removestops, mode=mode, **kwargs)
            seglist = jieba.cut(value, cut_all=True)
        for w in seglist:
            t.original = t.text = w
            t.boost = 1.0
            if positions:
                t.pos = start_pos + value.find(w)
            if chars:
                t.startchar = start_char + value.find(w)
                t.endchar = start_char + value.find(w) + len(w)
            yield t

def ChineseAnalyzer():
    return ChineseTokenizer()
```

> 在这个文件中，定义了一个`ChineseAnalyzer`的函数，这个函数将替代搜索引擎配置文件中的分词方式

- 复制引擎文件，修改分词方式为中文

> 同样在该文件夹下`C:\Python37\Lib\site-packages\haystack\backends`，复制`whoosh_backend.py`文件，创建一个新的文件名为`whoosh_cn_backend.py`，这里复制出一份文件也是为了之后如果不需要使用中文分词，可以直接在`settings`配置文件中修改引擎为`'ENGINE':'haystack.backends.whoosh_backend.WhooshEngine',`

> 修改该引擎配置文件中的：`analyzer=StemmingAnalyzer()`变为`analyzer=ChineseAnalyzer()`
>
> 并且要记得在头部引入刚才所编写的中文分词文件

```
#whoosh_cn_backend.py
from .ChineseAnalyzer import ChineseAnalyzer
```

初始化索引

- 最后，初始化索引数据

```
python manage.py rebuild_index
```