

温州大学瓯江学院

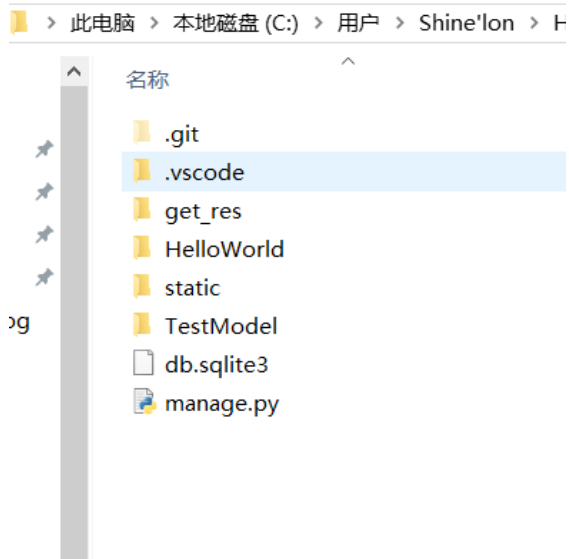
WENZHOU UNIVERSITY OUJIANG COLLEGE



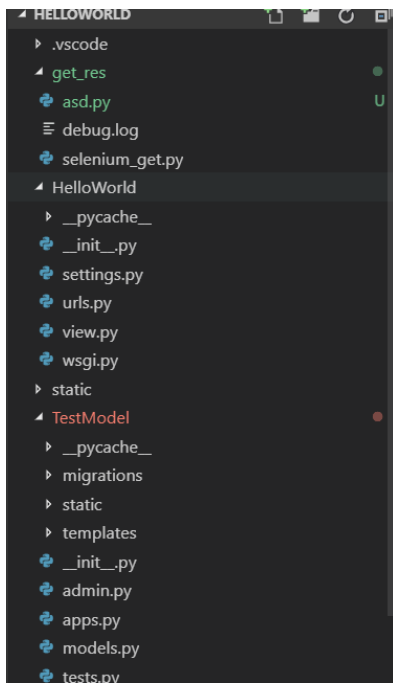
综合实验报告

学 期	大三下学期		
课 程	Python 爬虫期中作业		
专 业	计算机	班 级	三班
学生姓名	梁家欢	学 号	16219111327

1 当前目录结构



其中 `get_res` 里放的是爬虫代码



2 模型及数据库设计

模型设计 models.py

```
# models.py
from django.db import models
from django.contrib.auth.models import User

class Test(models.Model):
    name = models.CharField('分类',max_length=128)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name='博客分类'
        verbose_name_plural=verbose_name

class Tag(models.Model):
    name=models.CharField('标签',max_length=128)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name='博客标签'
        verbose_name_plural=verbose_name

class Entry(models.Model):
    title = models.CharField('文章标题',max_length=128)
    author = models.ForeignKey(User,verbose_name='作者',on_delete=models.CASCADE)
    img = models.ImageField(upload_to='blog_img',null=True,blank=True,verbose_name='博客配图')
    body = models.TextField('正文',)
    abstract = models.TextField('摘要',max_length=256,null=True,blank=True)
    visiting = models.PositiveIntegerField('访问量',default=0)
    category = models.ManyToManyField('Test',verbose_name='博客分类')
    tags = models.ManyToManyField('Tag',verbose_name='标签')
    created_time = models.DateTimeField('创建时间',auto_now_add=True)
    modified_time = models.DateTimeField('修改时间',auto_now=True)

    def __str__(self):
        return self.title

    class Meta:
```

```

        ordering = ['-created_time']
        verbose_name = '博客正文'
        verbose_name_plural = verbose_name

class phone(models.Model):
    img_url=models.CharField(max_length=128)
    link=models.CharField(max_length=128)
    price=models.CharField(max_length=128)
    name=models.CharField(max_length=128)
    comment=models.CharField(max_length=128)

```

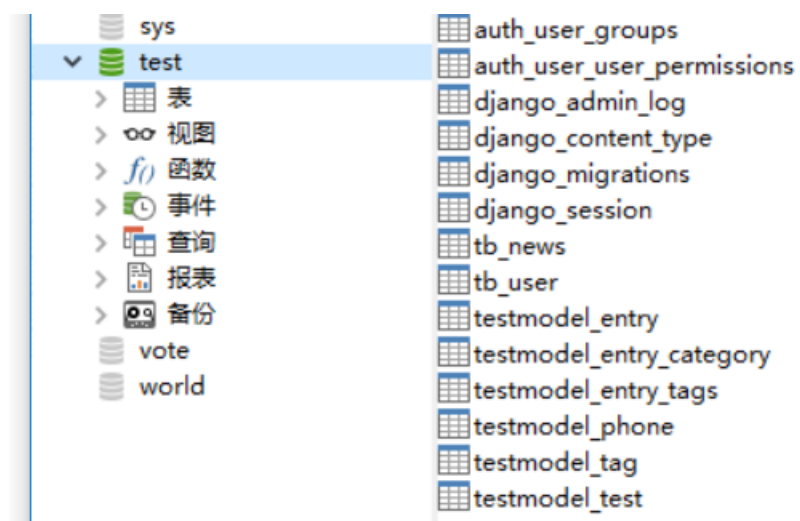
在 settings 中设置连接数据库

```

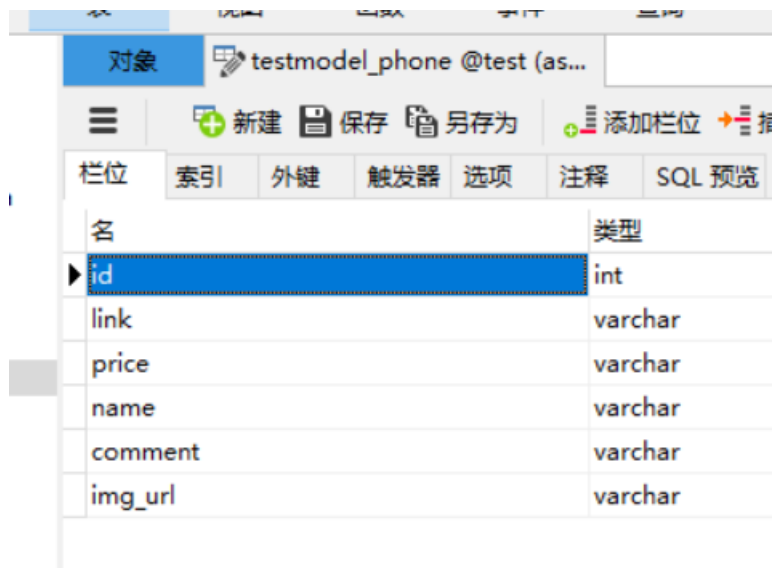
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'test',
        'USER': 'root',
        'PASSWORD': '123456',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}

```

模型生成的数据如下



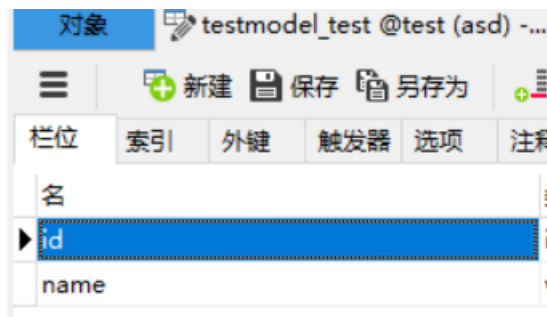
京东手机的表



The screenshot shows a database management interface with a table named 'testmodel_phone' selected. The table structure is displayed in a tabular format with columns for '名' (Name) and '类型' (Type). The 'id' column is highlighted in blue.

名	类型
id	int
link	varchar
price	varchar
name	varchar
comment	varchar
img_url	varchar

电影的表



The screenshot shows a database management interface with a table named 'testmodel_test' selected. The table structure is displayed in a tabular format with columns for '名' (Name) and '类型' (Type). The 'id' column is highlighted in blue.

名	类型
id	int
name	varchar

3 urls 及视图设计

Helloword/urls.py

```
from django.conf.urls import *
from django.contrib import admin

urlpatterns = [
    url(r'^admin/',admin.site.urls),
    url(r'^TestModel/',include('TestModel.urls'),name='TestModel'),
]
```

Testmodel/urls.py

```
from django.conf.urls import url
from . import views
app_name = 'TestModel'

urlpatterns = [
    url(r'^$', views.index,name='blog_index'),
    url(r'^(?P<blog_id>[0-9]+)', views.detail,name='blog_detail'),
    url(r'^show/',views.lists,name='movie_show'),
    url(r'^selemiun_show/',views.phone_msg,name="phone_show")
]
```

TestModel/views.py

```
from django.shortcuts import render
from . import models

def index(request):
    return render(request,'TestModel/index.html',locals())

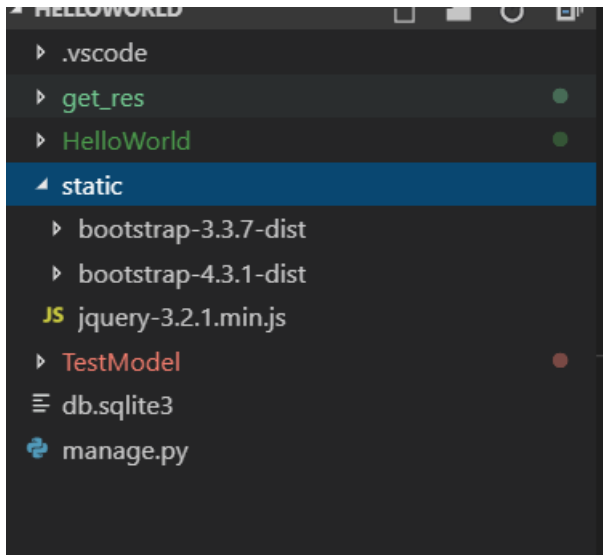
def detail(request,blog_id):

    return render(request,'TestModel/detail.html',locals())

def lists(request):
    movie_list=models.Test.objects.all()
    return render(request,'TestModel/show.html',{'movie_list':movie_list})

def phone_msg(request):
    phone_list=models.phone.objects.all()
    return render(request,'TestModel/selemiun_show.html',{'phone_list':phone_list})
```

Bootstrap 放在了 static 目录下

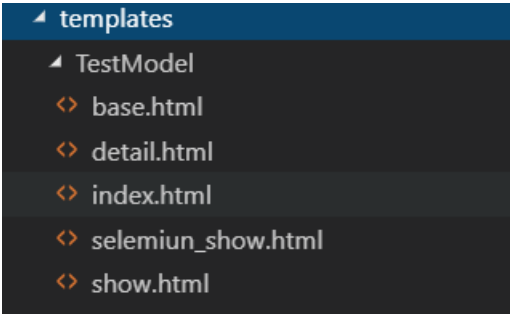


在 settings 中设置路径

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

4 前端页面设计

网站拥有页面如下



其中 base.html 和 index.html 是首页

Show.html 是展示爬到的豆瓣电影的数据

Selemiu_show.html 展示的是动态爬取的京东手机商品的数据

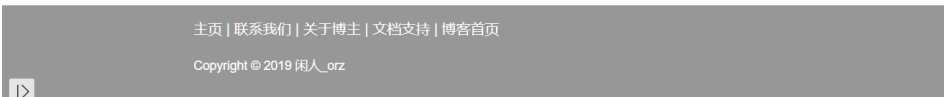
在

首页



爬取豆瓣top50电影

selemiu动态爬取



代码

Index.html

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

{% block title %}博客首页{% endblock %}

{% block content %}
    <a style="font-size:25px" href="{%url 'TestModel:movie_show' %}"><span style="color:
#5e5554">爬取豆瓣 top50 电影</span></a><br/>
</br/>
```



```

    <a style="font-size: 25px" href="{%url 'TestModel:phone_show' %}"><span style="color:
#5e5554">selemiun 动态爬取</span></a>
    <div style="height: 440px; " ></div>
{% endblock %}

```

Base.html

```

{% load staticfiles %}

<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8">
    <title>{% block title %}{% endblock %}</title>

    <link href="{% static 'bootstrap-3.3.7-dist/css/bootstrap.min.css' %}"
rel="stylesheet">
    <link href="{% static 'TestModel/css/TestModel_nav.css' %}" rel="stylesheet">
    {% block css %}{% endblock %}

</head>
<body>

<nav class="navbar navbar-fixed-top">
    <div class="container-fluid">
        <!-- Brand and toggle get grouped for better mobile display -->
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
data-target="#my-nav" aria-expanded="false">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">闲人_orz</a>
        </div>

        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="my-nav">
            <ul class="nav navbar-nav">
                <li class="active"><a href="/blog/">博客</a></li>
                <li><a href="#">学习资源</a></li>
                <li><a href="#">联系我</a></li>
            </ul>

            <form class="navbar-form navbar-left">
                <div class="form-group">

```

```

        <input type="text" class="form-control" placeholder="Search">
    </div>
    <button type="submit" class="btn btn-default">搜索</button>
</form>

    <ul class="nav navbar-nav navbar-right">
        <li><a href="#">登录</a></li>
    </ul>

</div><!-- /.navbar-collapse -->
</div><!-- /.container-fluid -->
</nav>

{% block content %}{% endblock %}

<footer>
    <div class="footer" role="navigation">
        <div class="container">
            <div class="navbar-text">
                <ul class="footer-text">
                    <li><a href="#">主页</a></li>
                    <li><a href="#">联系我们</a></li>
                    <li><a href="#">关于博主</a></li>
                    <li><a href="#">文档支持</a></li>
                    <li><a href="/blog/">博客首页</a></li>
                </ul>
                <p>Copyright © 2019 闲人_orz </p>
            </div>
        </div>
    </div>

<script src="{% static 'jquery-3.2.1.min.js' %}"></script>
<script src="{% static 'bootstrap-3.3.7-dist/js/bootstrap.min.js' %}"></script>

{% block script %}{% endblock %}

</body>
</html>

```

其中头部和尾部是基于 bootstrap 创建的样式

展示页面 1

爬取豆瓣top50电影

排名	电影名
1	肖申克的救赎
2	霸王别姬
3	这个杀手不太冷
4	阿甘正传
5	美丽人生
6	泰坦尼克号
7	千与千寻
8	辛德勒的名单
9	盗梦空间
10	忠犬八公的故事
11	机器人总动员
12	三傻大闹宝莱坞
13	海上钢琴师
14	放牛班的春天
15	楚门的世界
16	大话西游之大圣娶亲

Show.html

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8">
    <title>豆瓣 top25 电影</title>
    <link href="{% static 'bootstrap-3.3.7-dist/css/bootstrap.min.css' %}"
rel="stylesheet">
</head>
<body>
<h1>爬取豆瓣 top50 电影</h1>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>排名</th>
            <th>电影名</th>
        </tr>
    </thead>
    <tbody>
        {% for line in movie_list %}
        <tr>
```

```

        <td>{{line.id}}</td>
        <td>{{line.name}}</td>
    </tr>
    {% endfor %}
</tbody>
</table>

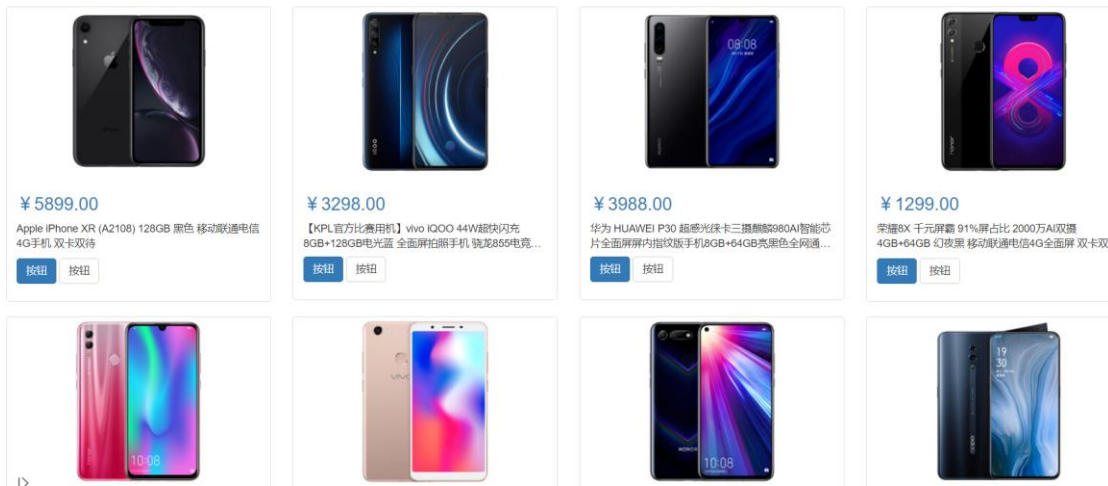
<script src="{% static 'jquery-3.2.1.min.js' %}"></script>
<script src="{% static 'bootstrap-3.3.7-dist/js/bootstrap.min.js' %}"></script>
</body>
</html>

```

其中用了 bootstrap 中的表格的网格样式对数据进行排版

展示页面 2

京东手机商品信息



Selemin.html

```

{% load staticfiles %}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8">
    <title>动态爬取京东手机商品</title>
    <link href="{% static 'bootstrap-3.3.7-dist/css/bootstrap.min.css' %}"
rel="stylesheet">
    <link href="{% static 'TestModel/css/index.css' %}" rel="stylesheet">
</head>
<body>
<h2>京东手机商品信息</h2>
<div class="row">
    {% for line in phone_list %}
    <div class="col-sm-6 col-md-3">

```

```

        <div class="thumbnail">
            <a target="_blank" title="asdfa" href="http://www.baidu.com">
                </a>
                <div class="caption">
                    <h3
style="overflow:hidden;white-space:nowrap;text-overflow:ellipsis;"><a
href="{{line.link}}">¥{{line.price}}</a></h3>
                    <p>{{line.name}}</p>
                    <p>
                        <a href="#" class="btn btn-primary" role="button">
                            按钮
                        </a>
                        <a href="#" class="btn btn-default" role="button">
                            按钮
                        </a>
                    </p>
                </div>
            </div>
        </div>
        {% endfor %}
    </div>
<script src="{% static 'jquery-3.2.1.min.js' %}"></script>
<script src="{% static 'bootstrap-3.3.7-dist/js/bootstrap.min.js' %}"></script>
</body>
</html>

```

用 bootstrap 缩略图的样式将爬到的数据展示出来

其中数据数据是由爬虫所爬得
豆瓣电影爬虫代码为

```

import requests
import lxml
from lxml import etree
import csv
import pymysql

class Spider:
    def __init__(self,version):
        self.version = version
        self.result = []

    def get_page(self,start_num):

```

```

url='https://movie.douban.com/top250?start=%s&filter=' % start_num
res=requests.get(url)

tree=etree.HTML(res.text)
top205=tree.xpath('//span[@class="title"][1]/text()')
print(top205)
return top205
def go(self):
    print("Start..")
    for i in range(0,1):
        top250=self.get_page(i*25)
        self.result +=top250

    return self.result

if __name__=="__main__":
    my_spider=Spider('1.0')
    res=my_spider.go()
    with open('D:/PySy.csv','a+',encoding='UTF-8')as csvfile:
        w=csv.writer(csvfile)
        w.writerow(res)

def getImage():
    my_spider=Spider('1.0')
    res=my_spider.go()
    db=pymysql.connect("localhost","root","123456","mysql")
    cursor=db.cursor()
    sql="INSERT INTO MovieTop(movie) VALUES (%s)"
    for a in res:
        cursor.execute(sql,(a))
        db.commit()
    db.close()

```

动态爬取京东代码为

```

from selenium import webdriver
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import selenium.common.exceptions
import json
import csv
import time
import pymysql

```

```

class JdSpider():
    def open_file(self):
        self.fm = input('请输入文件保存格式 (txt、json、csv): ')
        while self.fm!='txt' and self.fm!='json' and self.fm!='csv':
            self.fm = input('输入错误, 请重新输入文件保存格式 (txt、json、csv): ')
        if self.fm=='txt' :
            self.fd = open('D:/Jd.txt','w',encoding='utf-8')
        elif self.fm=='json' :
            self.fd = open('Jd.json','w',encoding='utf-8')
        elif self.fm=='csv' :
            self.fd = open('Jd.csv','w',encoding='utf-8',newline='')

    def open_browser(self):
        self.browser = webdriver.Chrome()
        self.browser.implicitly_wait(10)
        self.wait = WebDriverWait(self.browser,10)

    def init_variable(self):
        self.data = zip()
        self.isLast = False

    def parse_page(self):
        try:
            skus =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//li[@class="gl-item"]
')))
            skus = [item.get_attribute('data-sku') for item in skus]
            links = ['https://item.jd.com/{sku}.html'.format(sku=item) for item in skus]
            prices =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[3]/strong/i')))
            prices = [item.text for item in prices]
            names =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[4]/a/em')))
            names = [item.text for item in names]
            comments =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[5]/strong')))
            comments = [item.text for item in comments]
            img_urls =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[1]/a/img')))
            img_urls = [item.get_attribute('src') for item in img_urls]
            self.data = zip(links,prices,names,comments,img_urls)
        except selenium.common.exceptions.TimeoutException:

```

```

        print('parse_page: TimeoutException1')
        self.parse_page()
    except selenium.common.exceptions.StaleElementReferenceException:
        print('parse_page: StaleElementReferenceException')
        self.browser.refresh()

def turn_page(self):
    try:

self.wait.until(EC.element_to_be_clickable((By.XPATH, '//a[@class="pn-next"]'))).click
()

        time.sleep(1)

self.browser.execute_script("window.scrollTo(0,document.body.scrollHeight)")
        time.sleep(2)
    except selenium.common.exceptions.NoSuchElementException:
        self.isLast = True
    except selenium.common.exceptions.TimeoutException:
        print('turn_page: TimeoutException2')
        self.turn_page()
    except selenium.common.exceptions.StaleElementReferenceException:
        print('turn_page: StaleElementReferenceException')
        self.browser.refresh()

def write_to_file(self):
    if self.fm == 'txt':
        for item in self.data:
            self.fd.write('-----\n')
            self.fd.write('link: ' + str(item[0]) + '\n')
            self.fd.write('price: ' + str(item[1]) + '\n')
            self.fd.write('name: ' + str(item[2]) + '\n')
            self.fd.write('comment: ' + str(item[3]) + '\n')
    if self.fm == 'json':
        temp = ('link','price','name','comment')
        for item in self.data:
            json.dump(dict(zip(temp,item)),self.fd,ensure_ascii=False)
    if self.fm == 'csv':
        writer = csv.writer(self.fd)
        for item in self.data:
            writer.writerow(item)
def write_to_mysql(self):
    db=pymysql.connect("localhost","root","123456","test")
    cursor=db.cursor()
    sql="INSERT INTO testmodel_phone(link,price,name,comment,img_url) VALUES
(%s,%s,%s,%s,%s)"
    for item in self.data:

```



```

        for item in self.data:
            cursor.execute(sql,(item[0],item[1],item[2],item[3],item[4]))
            db.commit()
        db.close()

def close_file(self):
    self.fd.close()

def close_browser(self):
    self.browser.quit()

def crawl(self):
    #self.open_file()
    self.open_browser()
    self.init_variable()
    db=pymysql.connect("localhost","root","123456","test")
    cursor=db.cursor()
    sql="truncate table testmodel_phone"
    cursor.execute(sql)
    db.commit()
    db.close()
    print('开始爬取')

self.browser.get('https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8')
time.sleep(1)
self.browser.execute_script("window.scrollTo(0,document.body.scrollHeight)")
time.sleep(2)
count = 0
while count!=2:
    count += 1
    print('正在爬取第 ' + str(count) + ' 页.....')
    self.parse_page()
    self.write_to_mysql()
    self.turn_page()
    #self.close_file()
    self.close_browser()
    print('结束爬取')

if __name__ == '__main__':
    spider = JdSpider()
    spider.crawl()

```

5 爬虫代码展示

静态页面爬取

豆瓣 top50

Asd.py

```
import requests
import lxml
from lxml import etree
import csv
import pymysql

class Spider:
    def __init__(self, version):
        self.version = version
        self.result = []

    def get_page(self, start_num):
        url = 'https://movie.douban.com/top250?start=%s&filter=' % start_num
        res = requests.get(url)

        tree = etree.HTML(res.text)
        top205 = tree.xpath('//span[@class="title"][1]/text()')
        print(top205)
        return top205

    def go(self):
        print("Start..")
        for i in range(0, 1):
            top250 = self.get_page(i * 25)
            self.result += top250

        return self.result

if __name__ == "__main__":
    my_spider = Spider('1.0')
    res = my_spider.go()
    with open('D:/PySy.csv', 'a+', encoding='UTF-8') as csvfile:
        w = csv.writer(csvfile)
        w.writerow(res)

def getImage():
    my_spider = Spider('1.0')
    res = my_spider.go()
    db = pymysql.connect("localhost", "root", "123456", "mysql")
    cursor = db.cursor()
    sql = "INSERT INTO MovieTop(movie) VALUES (%s)"
```

```

for a in res:
    cursor.execute(sql,(a))
    db.commit()
db.close()

```

selemium 动态爬取

京东手机商品商品信息

selemium_get.py

```

from selenium import webdriver
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import selenium.common.exceptions
import json
import csv
import time
import pymysql

class JdSpider():
    def open_file(self):
        self.fm = input('请输入文件保存格式 (txt、json、csv): ')
        while self.fm!='txt' and self.fm!='json' and self.fm!='csv':
            self.fm = input('输入错误, 请重新输入文件保存格式 (txt、json、csv): ')
        if self.fm=='txt' :
            self.fd = open('D:/Jd.txt','w',encoding='utf-8')
        elif self.fm=='json' :
            self.fd = open('Jd.json','w',encoding='utf-8')
        elif self.fm=='csv' :
            self.fd = open('Jd.csv','w',encoding='utf-8',newline='')

    def open_browser(self):
        self.browser = webdriver.Chrome()
        self.browser.implicitly_wait(10)
        self.wait = WebDriverWait(self.browser,10)

    def init_variable(self):
        self.data = zip()
        self.isLast = False

    def parse_page(self):
        try:
            skus =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH,'//li[@class="gl-item"]
')))
            skus = [item.get_attribute('data-sku') for item in skus]

```

```

        links = ['https://item.jd.com/{sku}.html'.format(sku=item) for item in skus]
        prices =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[3]/strong/i')))
        prices = [item.text for item in prices]
        names =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[4]/a/em')))
        names = [item.text for item in names]
        comments =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[5]/strong')))
        comments = [item.text for item in comments]
        img_urls =
self.wait.until(EC.presence_of_all_elements_located((By.XPATH, '//div[@class="gl-i-wra
p"]/div[1]/a/img')))
        img_urls = [item.get_attribute('src') for item in img_urls]
        self.data = zip(links,prices,names,comments,img_urls)
    except selenium.common.exceptions.TimeoutException:
        print('parse_page: TimeoutException1')
        self.parse_page()
    except selenium.common.exceptions.StaleElementReferenceException:
        print('parse_page: StaleElementReferenceException')
        self.browser.refresh()

    def turn_page(self):
        try:

self.wait.until(EC.element_to_be_clickable((By.XPATH, '//a[@class="pn-next"]'))).click
()

        time.sleep(1)

self.browser.execute_script("window.scrollTo(0,document.body.scrollHeight)")
        time.sleep(2)
    except selenium.common.exceptions.NoSuchElementException:
        self.isLast = True
    except selenium.common.exceptions.TimeoutException:
        print('turn_page: TimeoutException2')
        self.turn_page()
    except selenium.common.exceptions.StaleElementReferenceException:
        print('turn_page: StaleElementReferenceException')
        self.browser.refresh()

    def write_to_file(self):
        if self.fm == 'txt':
            for item in self.data:

```

```

        self.fd.write('-----\n')
        self.fd.write('link: ' + str(item[0]) + '\n')
        self.fd.write('price: ' + str(item[1]) + '\n')
        self.fd.write('name: ' + str(item[2]) + '\n')
        self.fd.write('comment: ' + str(item[3]) + '\n')
    if self.fm == 'json':
        temp = ('link','price','name','comment')
        for item in self.data:
            json.dump(dict(zip(temp,item)),self.fd,ensure_ascii=False)
    if self.fm == 'csv':
        writer = csv.writer(self.fd)
        for item in self.data:
            writer.writerow(item)
def write_to_mysql(self):
    db=pymysql.connect("localhost","root","123456","test")
    cursor=db.cursor()
    sql="INSERT INTO testmodel_phone(link,price,name,comment,img_url) VALUES
(%s,%s,%s,%s,%s)"
    for item in self.data:
        for item in self.data:
            cursor.execute(sql,(item[0],item[1],item[2],item[3],item[4]))
            db.commit()
    db.close()

def close_file(self):
    self.fd.close()

def close_browser(self):
    self.browser.quit()

def crawl(self):
    #self.open_file()
    self.open_browser()
    self.init_variable()
    db=pymysql.connect("localhost","root","123456","test")
    cursor=db.cursor()
    sql="truncate table testmodel_phone"
    cursor.execute(sql)
    db.commit()
    db.close()
    print('开始爬取')

self.browser.get('https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8')
    time.sleep(1)
    self.browser.execute_script("window.scrollTo(0,document.body.scrollHeight)")
    time.sleep(2)

```

```
count = 0
while count!=2:
    count += 1
    print('正在爬取第 ' + str(count) + ' 页.....')
    self.parse_page()
    self.write_to_mysql()
    self.turn_page()
    #self.close_file()
    self.close_browser()
    print('结束爬取')

if __name__ == '__main__':
    spider = JdSpider()
    spider.crawl()
```