**基于SH的小说网站**

# 准备工作

开发环境：JDK8、python3.8、Oracle数据库、SpringToolSuite4

运行环境：Tomcat9.0

数据准备：python爬虫爬虫小说娃网站的部分小说分别存入classification(分类表)、novels(小说表)、chapters(章节表)

Python爬虫代码：

#导入urllib.request爬取网站信息

import urllib.request

#导入re正则表达式提取信息

import re

from bs4 import BeautifulSoup

#导入连接Oracle库的库

import cx\_Oracle

import time

#创建连接Oracle数据库类

class Oracle( object ):

#连接URL和账号密码

conn=cx\_Oracle.connect('scott/xyh991007@localhost:1521/XYH')

print("正在连接数据库...")

cursor=conn.cursor()

print("连接数据库连接成功！")

#创建classification、novel、chapters表

print("正在创建classification表...")

sql\_create\_table ='''create table classification (

id number not null primary key,

typeName varchar2(50) not null

)'''

cursor.execute(sql\_create\_table)

print("classification表创建成功！")

print("正在创建novel表...")

sql\_create\_table ='''create table novels (

id number primary key,

novelName varchar2(2000),

author varchar2(500),

state varchar2(200) not null,

num varchar2(200) not null,

sutou CLOB,

img BLOB,

type\_id number not null,

foreign key(type\_id) references classification(id)

)'''

cursor.execute(sql\_create\_table)

print("novel表创建成功！")

print("正在创建chapters表...")

sql\_create\_table ='''create table chapters (

id number not null primary key,

chapterName varchar2(500),

contents CLOB,

novel\_id number,

foreign key(novel\_id) references novel(id)

)'''

cursor.execute(sql\_create\_table)

print("chapters表创建成功！")

#创建分类、文章、章节序列

#seq\_type\_id = "create sequence seq\_type\_id start with 1 increment by 1 maxvalue 999999999 noCycle"

#cursor.execute(seq\_type\_id)

#seq\_novel\_id = "create sequence seq\_novel\_id start with 1 increment by 1 maxvalue 999999999 noCycle"

#cursor.execute(seq\_novel\_id)

seq\_content\_id = "create sequence seq\_content\_id start with 1 increment by 1 maxvalue 999999999 noCycle"

cursor.execute(seq\_content\_id)

#提交

conn.commit()

cursor.close()

#往classification表中添加数据

def addClassification(self,ID,typeName):

cursor = self.conn.cursor()

sql = "INSERT INTO classification VALUES (:1,:2)"

args = (ID,typeName)

cursor.execute(sql,args)

#提交

self.conn.commit()

cursor.close()

#往novels表中添加数据

def addNovel(self,Id,novelName,author,state,num,sutou,img,type\_id):

cursor = self.conn.cursor()

sql = "INSERT INTO novels VALUES (:1,:2,:3,:4,:5,:6,:7,:8)"

args = (Id,novelName,author,state,num,sutou,img,type\_id)

cursor.execute(sql,args)

#提交

self.conn.commit()

cursor.close()

#往chapters表中添加数据

def addChapters(self,chapterName,contents,novel\_id):

cursor = self.conn.cursor()

sql = "INSERT INTO chapters VALUES (seq\_content\_id.nextval,:1,:2,:3)"

args = (chapterName,contents,novel\_id)

cursor.execute(sql,args)

#提交

self.conn.commit()

cursor.close()

#设置头信息

headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'}

#定义域名

domain = 'https://m.xiaoshuowawa.com'

#获取所有分类

def getTypeList():

#实例将要请求的对象

req = urllib.request.Request('https://m.xiaoshuowawa.com/sort/')

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码，将uff8解码成控制台编码

html = res.read().decode('UTF-8')

#正则表达式筛选信息

reg = r'<a href="(.\*?)">(.\*?)</a>'

#增加匹配效率，返回值类型为list

reg = re.compile(reg)

return re.findall(reg,html)

#获取分类的一页小说列表

def getNovelList(url):

#实例将要请求的对象

req = urllib.request.Request(domain + url)

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码，将uff8解码成控制台编码

html = res.read().decode('UTF-8')

#正则表达式筛选信息

reg = r'<div class="tt"><div class="left"><a href="(.\*?)"><img src="(.\*?)" width="78" height="103" /></a></div><div class="right"><p class="p1"><a href="(.\*?)">(.\*?)</a></p><div class="label"><p class="p2">(.\*?)</p><p class="p3">(.\*?)</p><p class="p4"><a href="(.\*?)">(.\*?)</a></p></div><div class="clear"></div><p class="p5">(.\*?)</p></div></div>'

#增加匹配效率，返回值类型为list

reg = re.compile(reg)

return re.findall(reg,html)

#获取小说简介信息

def getNovelSutou(url):

#实例将要请求的对象

req = urllib.request.Request(domain + url)

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码，将uff8解码成控制台编码

html = res.read().decode('UTF-8')

#创建一个BeautifulSoup解析对象

html = BeautifulSoup(html,'html.parser')

#查找class=".middle"下的select标签下的option标签

sutou\_list = html.select('.jj > p')

return sutou\_list

#获取章节页面链接

def getPageListUrl(url):

#实例将要请求的对象

req = urllib.request.Request(domain + url)

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码，将uff8解码成控制台编码

html = res.read().decode('UTF-8')

#创建一个BeautifulSoup解析对象

html = BeautifulSoup(html,'html.parser')

#查找class=".middle"下的select标签下的option标签

page\_list = html.select('.middle > select > option')

'''for pages in page\_list:

#输出value属性的值

print(pages['value'])

#输出标签内的值

print(pages.标签名.string)'''

return page\_list

#获取某一页章节

def getChapters(url):

#实例将要请求的对象

req = urllib.request.Request(domain + url)

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码，将uff8解码成控制台编码

html = res.read().decode('UTF-8')

#正则表达式筛选信息

reg = r'<li><a href="(.\*?)">(.\*?)</a></li>'

#增加匹配效率，返回值类型为list

reg = re.compile(reg)

return re.findall(reg,html)

#获取章节内容

def getContents(url):

#实例将要请求的对象

req = urllib.request.Request(domain + url)

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码，将uff8解码成控制台编码

html = res.read().decode('UTF-8')

#正则表达式筛选信息

reg = r'<div id="novelcontent" class="novelcontent">(.\*?)</div>'

#增加匹配效率，返回值类型为list

reg = re.compile(reg)

return re.findall(reg,html)

#将图片保存到数据库的转换方法

def imgToBLOB(url):

req = urllib.request.Request(url)

#替换所有头信息

req.headers = headers

#开始请求

res = urllib.request.urlopen(req)

#解码

imgdata = res.read()

return imgdata

#创建Oracle数据库实例

oracle = Oracle()

if \_\_name\_\_ == '\_\_main\_\_':

#分类起始id

typeID=-9

#小说起始id

novelId=0

#只爬取3页60章小说

chapterPageLength=3

for url,typeName in getTypeList():

typeID+=1

#分类从第10条到第18条才是我们所需要的

if typeID>=1 and typeID<9:

print('正在爬取--------{}'.format(typeName))

oracle.addClassification(typeID,typeName)

for url,imgUrl,url\_title,novelName,state,num,url\_author,author,smallsutou in getNovelList(url):

novelId+=1

img=imgToBLOB(imgUrl)

for sutou in getNovelSutou(url):

pass

print('正在爬取-----{}-----{}'.format(typeName,novelName))

oracle.addNovel(novelId,novelName,author,state,num,sutou.string,img,typeID)

print('正在存储-----{}-----{}'.format(typeName,novelName))

i=0

for pagesUrl in getPageListUrl(url):

i+=1

if i<=chapterPageLength:

for url,chapterName in getChapters(pagesUrl['value']):

for contents in getContents(url):

print('正在爬取---{}---{}---{}'.format(typeName,novelName,chapterName))

oracle.addChapters(chapterName,contents,novelId)

print('正在存储---{}---{}---{}'.format(typeName,novelName,chapterName))

else:

break

time.sleep(3)

#chapterPageLength+=0.5

time.sleep(3)

print('小说爬取成功！！！')

# 开始创建其他表及其序列

创建身份表，用于权限管理：

create table identify(

id number primary key,

identifyName varchar2(30) not null

);

创建用户表：

create table users(

id number primary key,

userName varchar2(30) not null,

passwd varchar2(30) not null,

avatars BLOB,

identify\_id number,

foreign key(identify\_id) references identify(id)

);

创建身份序列：

create sequence seq\_identify\_id start with 1 increment by 1 maxvalue 999999999 noCycle;

创建用户序列：

create sequence seq\_users\_id start with 1 increment by 1 maxvalue 999999999 noCycle;

创建分类序列：

create sequence seq\_classification\_id start with 7 increment by 1 maxvalue 999999999 noCycle;

创建小说序列：

create sequence seq\_novel\_id start with 330 increment by 1 maxvalue 999999999 noCycle;

创建章节序列：

create sequence seq\_chapter\_id start with 41005 increment by 1 maxvalue 999999999 noCycle;

# 手段往部分表中添加记录

往身份表中添加记录：

insert into identify values(seq\_identify\_id.nextval,'系统管理员');

insert into identify values(seq\_identify\_id.nextval,'普通用户');

往用户表中添加记录：

insert into users(id,username,passwd,identify\_id) values(seq\_users\_id.nextval,'test','test',3);

insert into users(id,username,passwd,identify\_id) values(seq\_users\_id.nextval,'admin','admin',2);

# 开始着手准备项目

SpringToolSuite4创建一个动态web项目

在src下创建hibernate.cfg.xml文件

hibernate.cfg.xml文件代码如下：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration SYSTEM "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd" PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN">

<!-- Generated by MyEclipse Hibernate Tools. -->

[<hibernate-configuration><session-factory>](D:/spring-tool-suite-4-4.8.1.RELEASE_workspace/Novel/src/hibernate.cfg.xml)<property name="**connection.username**">scott</property>

<!--用户名-->

<property name="**connection.password**">xyh991007</property>

<!--密码-->

<property name="**connection.driver\_class**"> oracle.jdbc.driver.OracleDriver </property>

<!--数据库驱动程序名-->

<property name="**connection.url**"> jdbc:oracle:thin:@localhost:1521:XYH </property>

<!--数据库实例url-->

<property name="**dialect**"> org.hibernate.dialect.OracleDialect </property>

<!--数据库方言 -->

<property name="**show\_sql**">false</property>

<!--是否将数据库反馈的SQL输出到日志-->

<property name="**format\_sql**">true</property>

<!--是否格式化SQL-->

<property name="**current\_session\_context\_class**">thread</property>

<!--session由当前执行的线程管理-->

<property name="**javax.persistence.validation.mode**"> none </property>

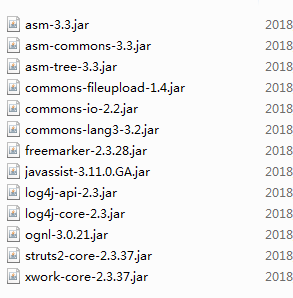
<!--在Web项目中使用Hibernate时需添加此属性-->

<mapping class="**po.Identify**"/><mapping class="**po.Users**"/><mapping class="**po.Classification**"/><mapping class="**po.Chapters**"/><mapping class="**po.Classification**"/><mapping class="**po.Novels**"/></session-factory></hibernate-configuration>

MyEclipse10反向过程生成po类和DAO类

# Struts2配置

用到的jar包：



Struts.xml基本代码：

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<constant name=*"struts.ui.theme"* value=*"simple"*/>

<package name=*"default"* namespace=*"/"* extends=*"struts-default"*>

</package>

</struts>



现在就可以愉快的写项目了

# 开始项目

项目目录结构：

