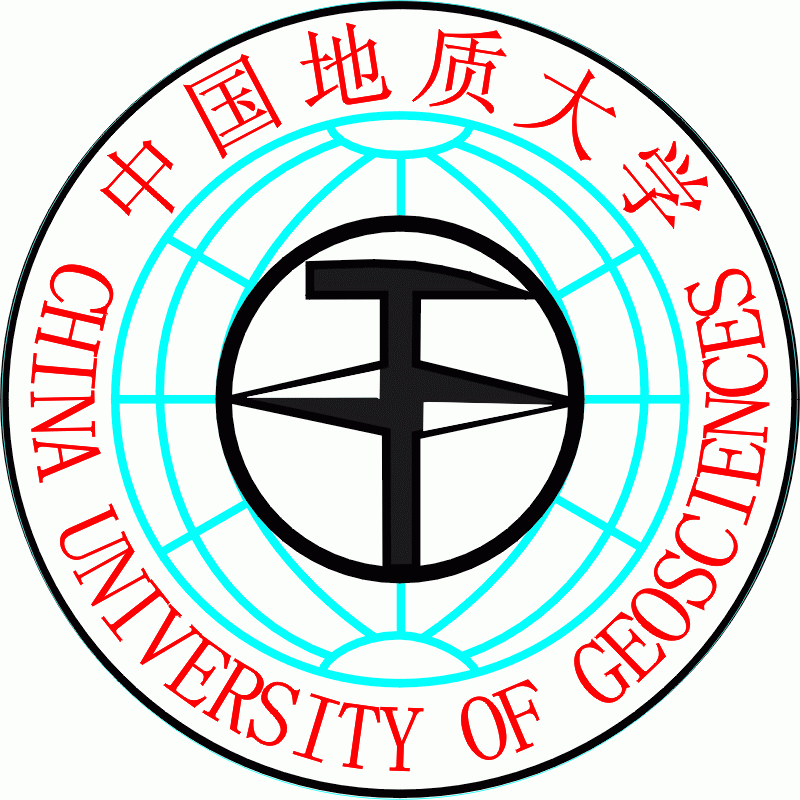
****

**软件开发技术课程设计**

**报告**

实习任务名称： java Web服务器模拟

小组成员： 20171001561 周麟

20181003540 王晨

**中国地质大学地理信息工程学院**

**软件工程系**

**2020年 6 月**

目录

[1.系统概述 2](#_Toc41476785)

[1.1 系统背景 2](#_Toc41476786)

[1.2 系统目标 2](#_Toc41476787)

[1.2 人员分工 2](#_Toc41476788)

[1.3 开发环境与工具 2](#_Toc41476789)

[2.需求分析 3](#_Toc41476790)

[2.1系统需求概述 3](#_Toc41476791)

[2.2 功能性需求 3](#_Toc41476792)

[3.概要设计 3](#_Toc41476794)

[3.1 系统总体设计 3](#_Toc41476795)

[3.2 系统功能设计 3](#_Toc41476796)

[3.3 系统功能细节 3](#_Toc41476796)

[4.详细设计 4](#_Toc41476797)

[4.1 模块接口设计 4](#_Toc41476798)

[5.系统成果展示 4](#_Toc41476799)

[5.1 模块1 4](#_Toc41476800)

[5.2模块2 4](#_Toc41476801)

[6.实习体会 5](#_Toc41476802)

# 1.系统概述

## 1.1 系统背景

[实习选题内容进行说明，可给出对当前研究问题或现有系统的现状分析]

1.要实现一个web服务器，实质上就是要实现对于BS通信，则浏览器与服务端对于资源文件共享的行为，那么必须要建立起这两者的联系。

2.共享资源文件，则就分为静态文件和动态文件。对于静态文件如图片，html等等只需要判断是否存在返回相应的状态码，存在则显示到请求页面。对于动态资源文件，如jsp文件（html代码和java代码的结合）则才服务器上运行这段程序，然后再将结果显示到请求页面中。

3.既然将资源显示到页面上，那么Java代码输出格式则要按照html语法格式。

4.实现一个简单的浏览器客户端，模拟浏览器发送请求。

## 1.2 系统目标

|  |  |
| --- | --- |
| 序号 | 目标内容 |
| 1 | 响应静态资源请求 |
| 2 | 根据下xml文件进行对servlet的配置 |
| 3 | 对不同的servlet进行响应 |
| 4 | 对简单的jsp格式文件进行响应 |
| 5 | 可以响应GET,POST（未实现）请求 |

|  |  |  |
| --- | --- | --- |
| 序号 | 姓名 | 承担角色和职责 |
| 1 | 周麟 | 负责目标内容123 |
| 2 | 王晨 | 负责目标内容45 |

## 1.2 人员分工

## 1.3 开发环境与工具

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 类别 | 具体需求 | 备注 |
| 1 | 硬件 |  |  |
|  |  |  |  |
| 2 | 软件 | 浏览器 |  |
|  |  | Tomcat |  |

# 

# 2.需求分析

## 2.1系统需求概述

本系统基于tomcat对于资源的请求响应原理进行模拟，利用servlet对于不同资源不用请求操作进行相应，并通过html语法或者jsp在请求页面中相应请求的资源信息，如果是jsp则将运行结果显示到请求页面。

客户端模拟器，通过对Http的header信息的模拟，来实现GET功能，POST未能实现。

## 2.2 功能性需求

[列出待实现系统的功能性需求]

1. 首先要跟浏览器进行通信连接。
2. 读取xml配置好servlet每个servletName对应的class。
3. 每次接收到请求，都提取出相应的信息，判断资源是否存在，返回相应的响应码。若存在，判断是GET或者POST，进行相应的操作。
4. 如果是jsp后缀则除了<%= 表达式 %>中提取出表达式，其他都是直接显示在网页中。

# 3.概要设计

## 3.1 系统总体设计

**系统总体框架：**

系统分为下列几层：

**第一层网络连接层**：利用socket对相应的ip和端口进行监听，并且对于监听分别赋予MyRequest和MyResponse输入流和输出流，MyRequest解析相应的请求信息，而MyResponse保存着相应的输出流和对于静态资源请求的页面显示操作。

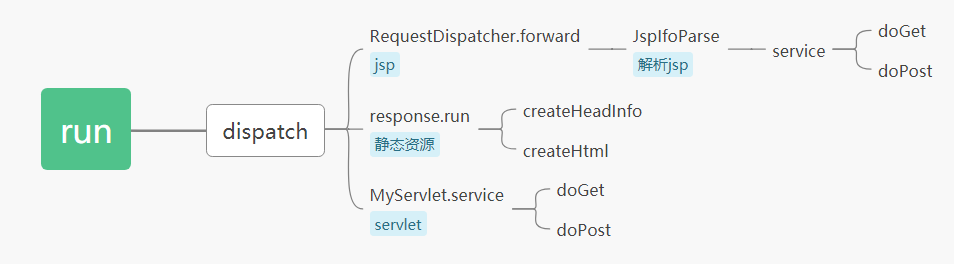
**第二层资源识别层**：

1. Myservlet模拟servlet作为一个抽象类，里边有抽象函数doGet和doPost，还有识别GET或者POST操作的service方法。所有需要进行servlet操作识别的java文件都要继承MyServlet类，重写doGet和doPost方法。
2. WebHandle读取xml文件对于servletName，className,mapping的读取，并用相应的容器进行保存。
3. WebContext则将WebHandle容器中存取的servletName，className,mapping转化成servletName，className和className,mapping的一对一的关系，并提供一个得到mapping相应className的方法。
4. JspIfoParse对于给定相应的jsp路径名和方法，解析成相应的java格式代码，方便在接收到jsp资源请求时进行显示。
5. Dispatch对于不同的请求调用不同的方法进行不同的操作。

**第三层数据传递层：**

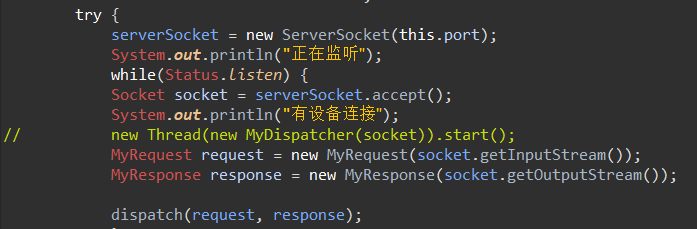
1. createHeadInfo实现了对于不同的状态码而形成相应的响应头，响应头格式按照html格式生成。
2. createHtml实现了对于不同静态资源类型，进行相应的资源响应，利用htlm语法在页面显示。

## 3.2 系统功能设计



## 3.3 系统功能细节

1.run方法实现socket监听，和向MyRequest和MyResponse传递参数。

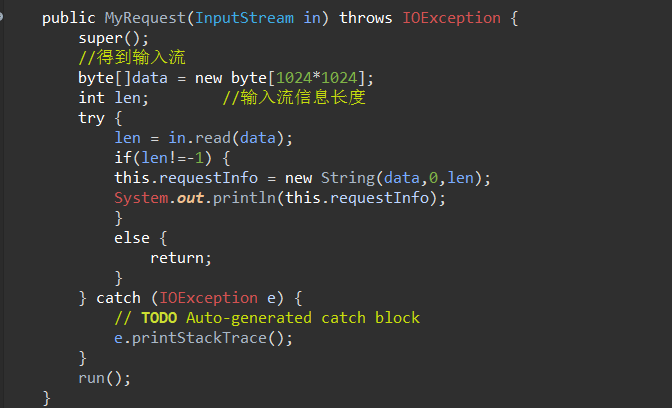


MyRequest和MyResponse处理请求信息，然后产生相应的输入输出流。

MyRequest解析出相应的请求操作，还有连接的ip地址和端口。

解析的相应操作是：

先读入输入流携带的http请求信息，保存作为成员变量。

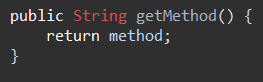


然后开始解析：

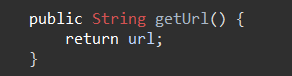
通过获取http和host下标获取相应的请求类型和ip端口。



并且设置方法来使得外界能够得到相应的请求操作类型。即：



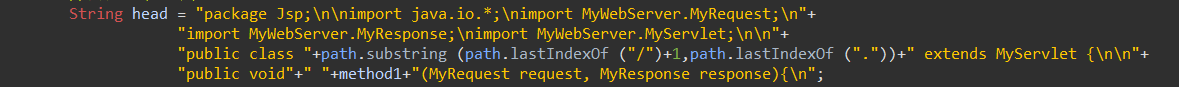
还有相应的url



2.在监听到连接之后，调用dispatch方法，进行请求资源类型识别，使用MyRequest得到getUrl方法得到文件路径，提取出文件后缀。

3.如果是jsp文件，调用RequestDispatcher类的forward方法，在此方法中，调用JspIfoParse类将jsp文件转化成java文件。转化概述：利用给定的路径得出包名和类名，然后模拟Java类文件格式利用filewriter写入新的Java文件，main函数保留除了<%= 表达式 %>格式的式子，然后将<%= 表达式 %>提取出表达式，写入Java文件完成后，利用反射到响应的类中，再调用servlet的service处理请求。

模拟java语言格式写入文件中：



因为要在页面中显示信息，所以要相应的响应头：



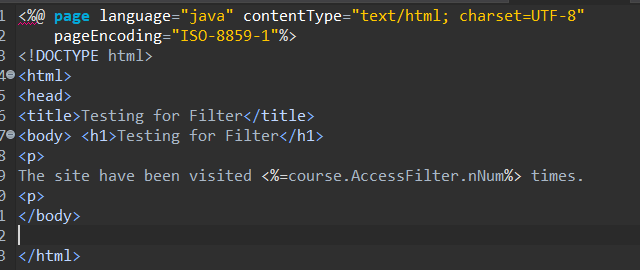
然后开始解析jsp文件：

解析jsp主要的操作就是消除html语言元素，然后将java代码提取出来。

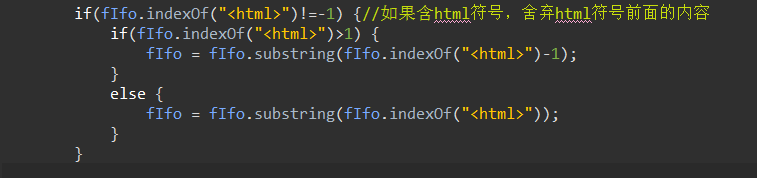
在解析之前要记录写入的.java文件的包和类名。

然后开始解析jsp文件。

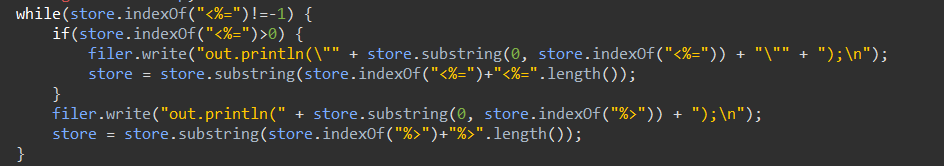
比如，show5.jsp文件：



<head>之前的代码都不需要，从后面开始出现java代码，所以舍去前面的代码



还有一个重点就是遇到了<%= 表达式 %>格式的提取：



在我们对于jsp生成的String数组中，如果判断出含有子字符串<%= 那么就利用String的indexof获得的下标将<%= 和 %>去掉。然后回到RequestDispatcher，通过得到的包+类名，通过反射，指向相应的类对象：



然后利用servlet的service进行操作的判断（下面servlet部分会介绍如何判断操作类型。

4.如果是静态文件，按照http响应信息报文信息得，首先先写入相应行，根据Myrequest得到的url响应的资源存在与否，给出响应的状态码和响应的表述。接下来是对于不同文件类型响应的响应头，然后不要忘记还要一个空白行，最后写入请求静态文件的数据，通过输出数据流输出到请求页面中。

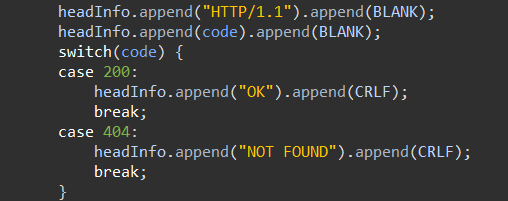
显示静态资源最重要的就是响应信息的部分，如果缺少了一些信息，就会导致浏览器无法识别，而响应不出信息。

Response除了保存输出流之外还提供了响应静态资源的方法。

包括createHeadInfo和createHtml。

createHeadInfo是通过响应码和文件类型来产生相应的响应的响应头

如果是200，说明资源存在，如果是400说明资源不存在:



然后根据文件种类：



这里我做了一点处理，因为图片需要提供文件大小，所以图片资源请求多加了一段：



然后进行请求体的创建：

也是根据传入的文件的种类，我这里设定如果请求的资源为空，也就是当前请求的是127.0.0.1：端口号，那么文件的种类就是first,其他都是后缀。

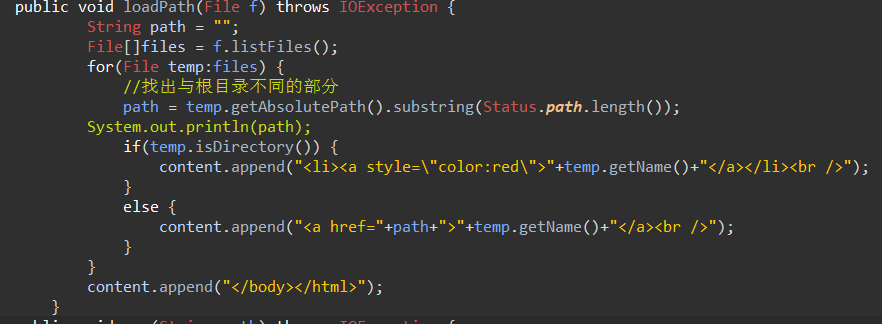
如果是first,那么就要调用loadPath方法:

loadPath的实现：

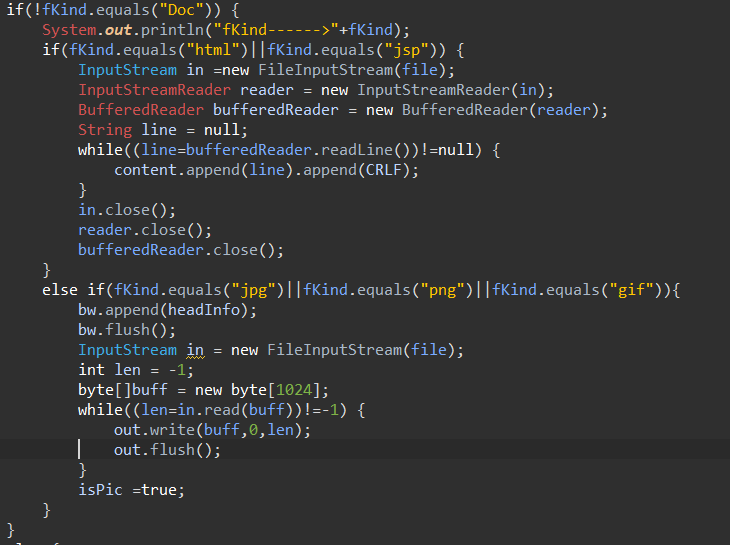
之前尝试将文件和文件夹都设置超链接，但是会出错，所以干脆取消了文件夹的超链接，这样能正常的处理请求。

显示的原理，如果for循环遍历当前目录，如果是文件，那么设置超链接，而响应的地址就是资源所在的地址+文件名。如果是目录，则设为红色且没有超链接。



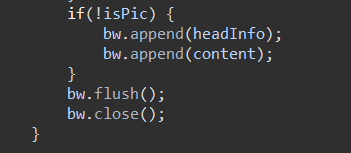


然后就是对静态资源的请求：



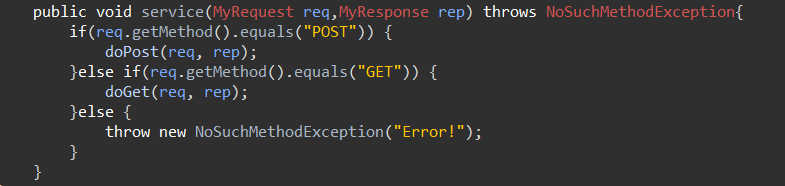
Jsp或者html文件则可以直接通过字节输出流输出，而图片类文件则需要单独的用字节流输出，所以如果当前是图片文件，就单独用字节流输出，之后就不用再被字符流输出。

即加了一层判定：



5.否则则需要进行servlet操作，根据Myrequest给出的url通过反射对应到响应的类中，然后通过service进行GET或者POST操作的判断。

Servlet的实现，只要是写了一个抽象类，里面的抽象方法doGet和doPost子类都要实现，还有service方法，只要是实现了判断是get或者post：

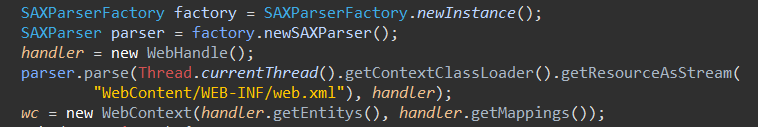


调用的是MyRequest的getMethod,获得请求方法来判断操作类型。

生成servlet对象时也是利用的反射，而反射对应的包名和类名则是通过xml文件的解析来完成的。

下一点介绍如何解析xml文件。

6．Xml文件的解析。首先必须读取xml文件，这里利用的是Java自带的库SAX。



首先产生相应的SAX工厂，再产生响应的解析对象。

SAX的操作如果要重写，那么就要继承DefaultHandler。

然后这里实现了一个类WebHandle继承了DefaultHandler。

重写startElement，这里的qName保存的是标签，而characters的ch数组保存的则是属性。

然后通过不同的容器将servlet-name和servlet-class，url-pattern和servlet-name，filter-name和 filter-class，filter-name和rl-pattern对应起来。

类WebHandle还提供响应的方法来让对象调用这些容器的方法。

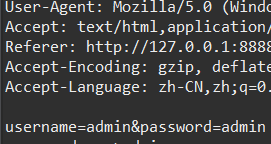
虽然在WebHandle实现了servlet-name和servlet-class，url-pattern和servlet-name，filter-name和 filter-class，filter-name和rl-pattern对应，但是servlet-class，url-pattern，filter-name和rl-pattern并不是一一对应的，所以另外写了一个类WebContext，将他们一一对应起来，并且也提供了对象通过parttern得到相应的类名。

这样，得到类名之后，就可以通过反射进行调用相应的类了，然后产生相应的servlet对象，调用service方法判断操作类型。

7，浏览器爬虫的设计，因为httpclient读取我们服务器的jsp时有问题，所以写了一个基于socket的爬虫，首先模拟Http头设置请求信息（包括http协议url资源，HOST，Connection，User-Agent和cookie几个信息），将这些请求发给服务器后，通过socket接收回应并保存下来。

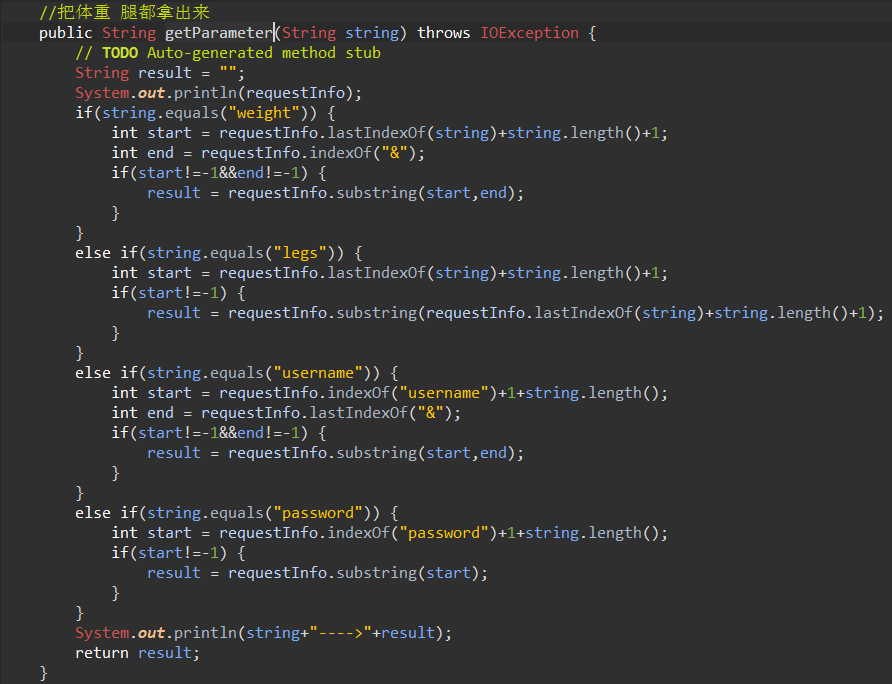
8.Login.java的实现，首先，让这个类继承我们写的MyServlet类，那样还需要实现的就是getParameter方法和sendRedirect方法。

getParameter方法主要实现的就是根据提供的字符串，返回相应的属性值，操作主要是解析出请求体中的字符串的内容，如：



我们需要解析出username=后的内容和password=后面的内容。

也是用indexOf得到下标提取。

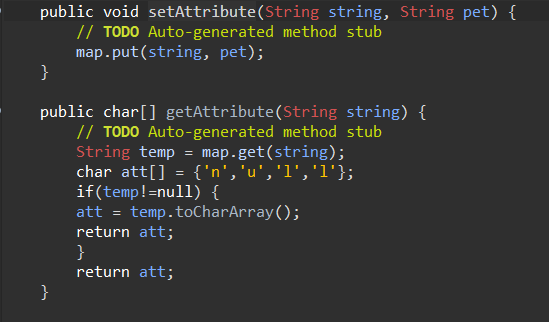
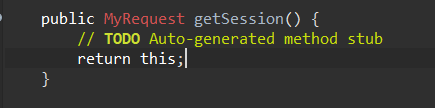


提取完之后还需要执行setAttribute方法，则利用一个hashmap对应username和属性值，password和属性值。

然后还需要判断是否是admin。

那么还需要实现getSession，getAttribute。

getSession则指向当前的MyRequest对象，getAttribute通过给定字符串名返回相应的属性值。就是利用setAttribute保存的haspmap中的键值对来返回相应的属性值。



然后是sendRedirect方法，主要实现的是通过给出的.jsp文件，产生相应的对象，来调用我们jsp解析产生的.java文件。



9. PetServlet只需要实现getParameter，然后用htmp语法在浏览器中显示即可。

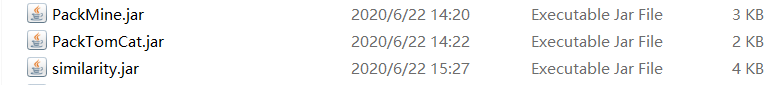
10.PetServlet2则需要通过jsp文件来生成相应的java文件在产生相应的java对象，然后通过类名反射指向相应的类，再用servlet，原理和jsp文件解析相同。

11，相似度比较，我们的服务器用的connect来保存消息，tomcat用Test01来保存消息，都存在了D盘，命名标准见代码，CodeFile内部代码不可修改。（如果要修改，请见两个GET里面的命名标准）。

12.所有功能大致完成之后，引入批处理文件，将启动服务器的指令写入，还将爬取请求资源信息写入文件的指令和计算相似度的指令写入一个bat文件中。先启动服务器才能进行爬取资源计算相似度。Bat批处理文件是一串模拟控制台指令的一个字符集。

具体bat文件的内容为：

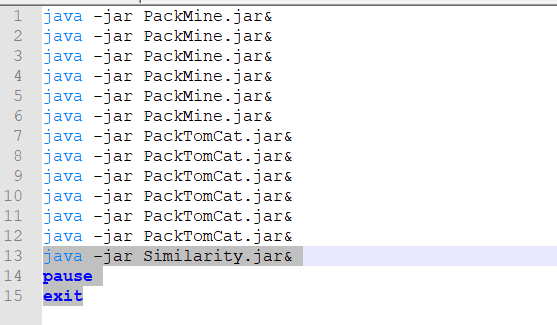
爬取和计算相似度的jar包



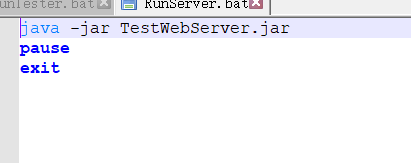
服务器的jar包



爬取和计算相似度的bat文件

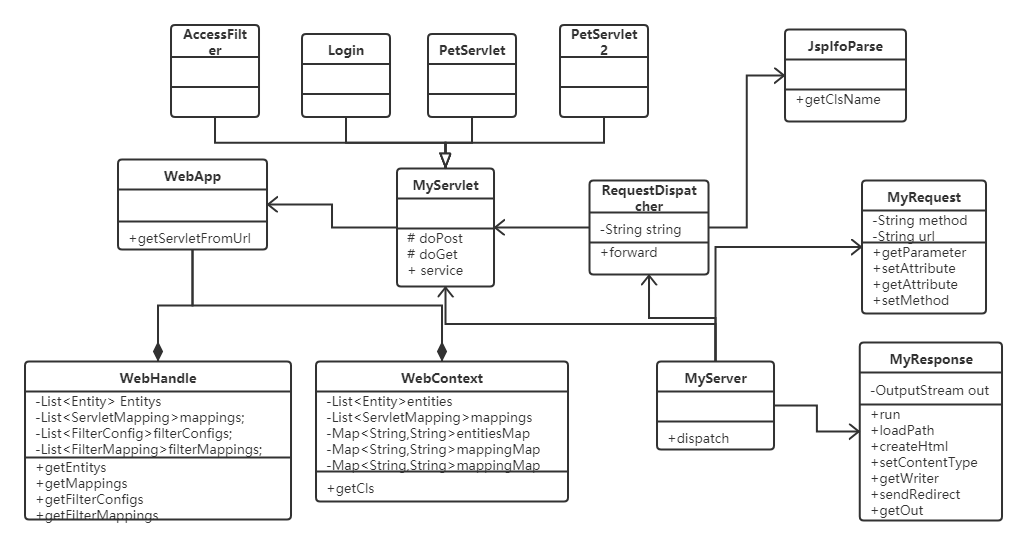


启动服务器的bat文件



# 4.详细设计

## 4.1 模块接口设计



# 5.系统成果展示

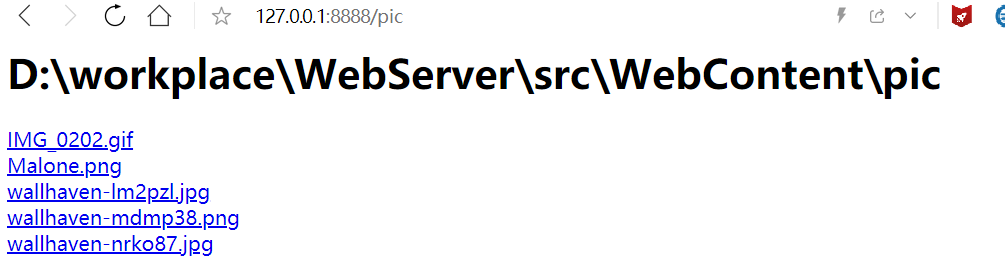
## 5.1 目录浏览（未实现可点击目录）



虽然没有实现点击目录，但可以直接在网址中输入路径

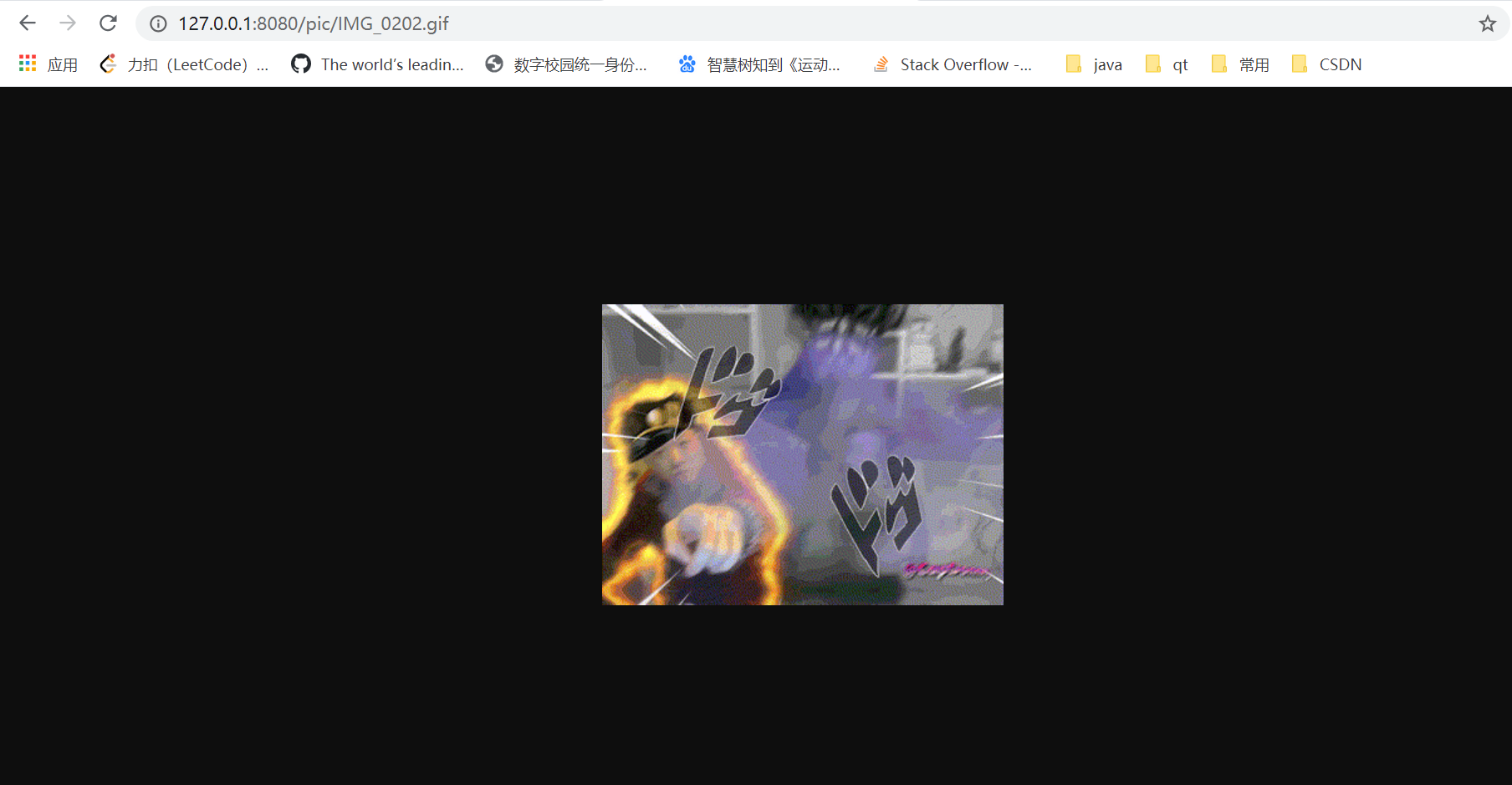
如果要访问pic文件夹，则应输入127.0.0.1：（端口号）/pic

如图

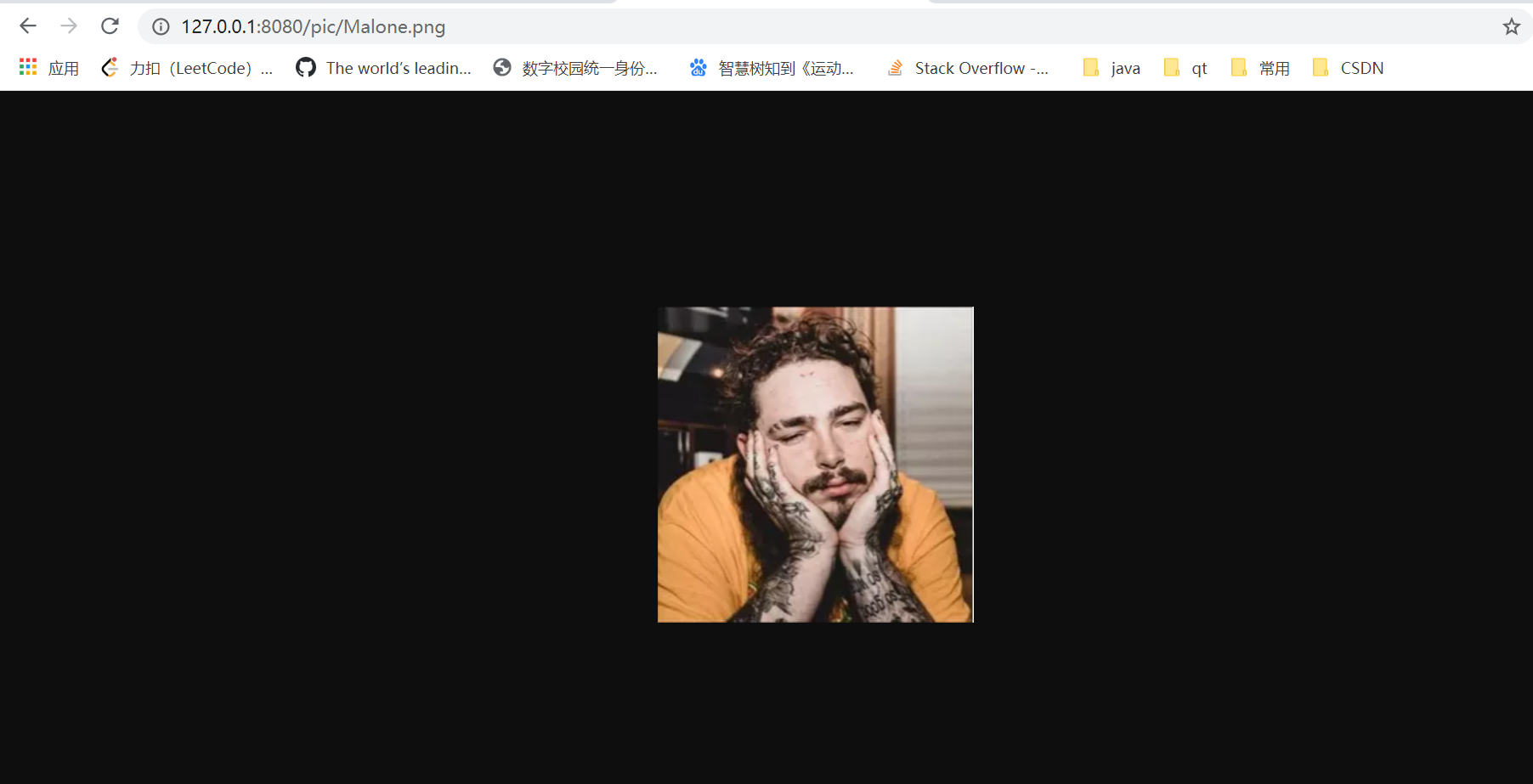


## 5.2 图片浏览

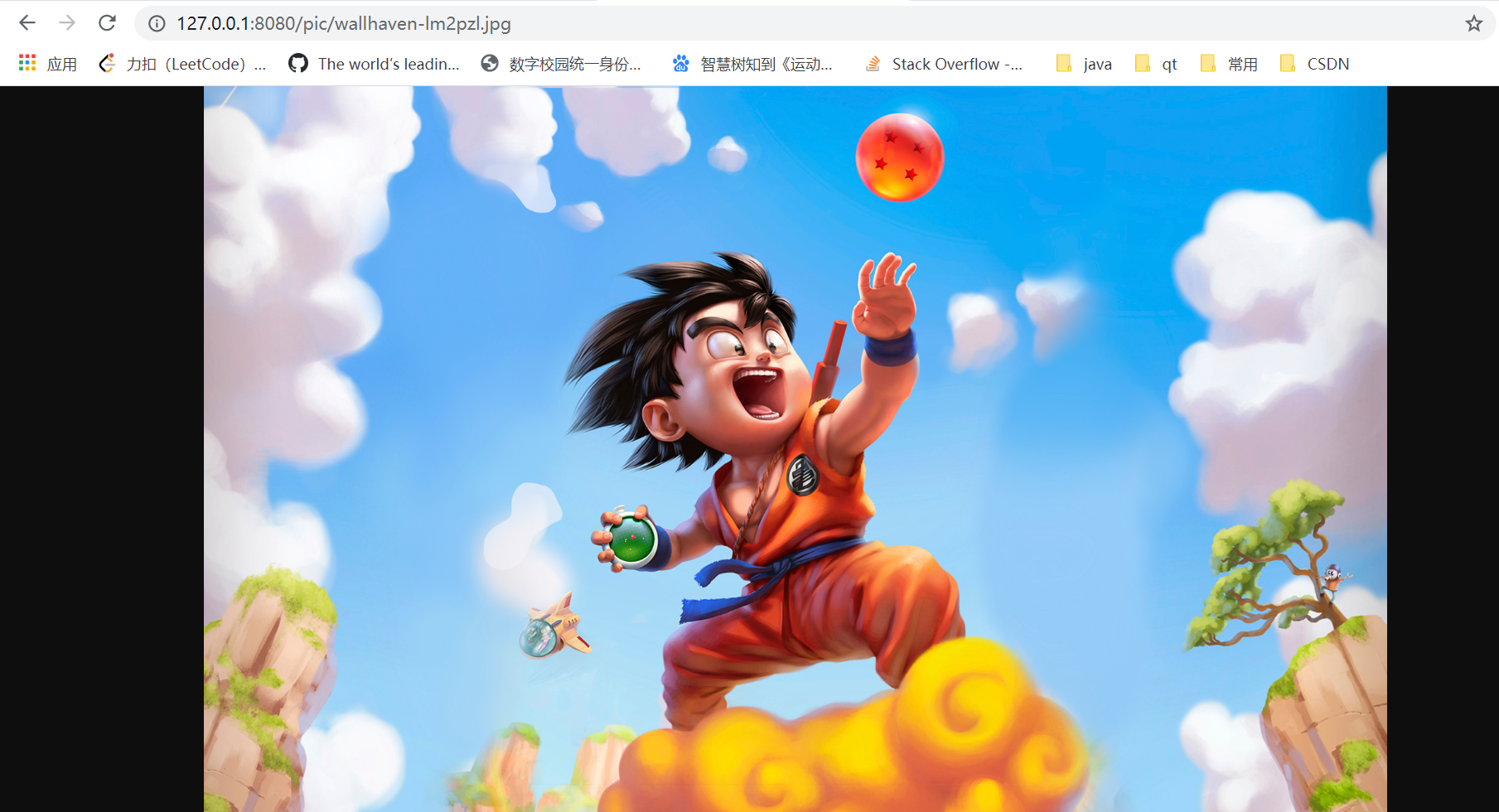
（gif）



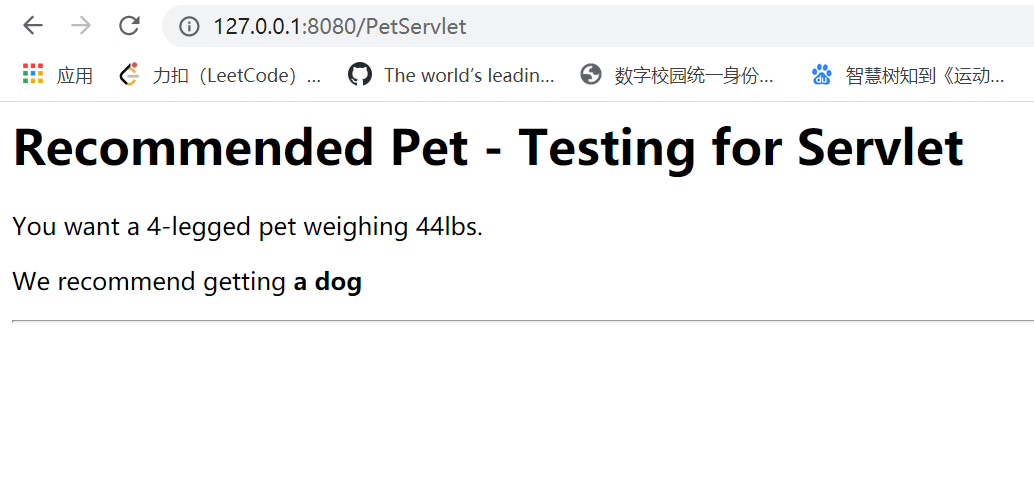
(png)



(jpg)



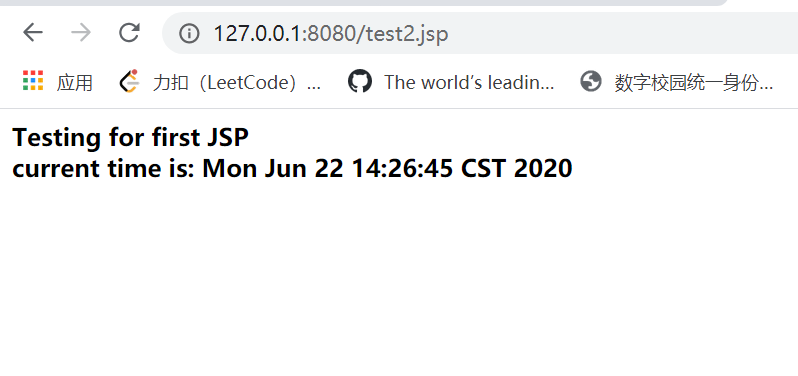
## 5.3 模块3 Servlet and Login





## 

## 5.4 jsp



(因为filter没有实现，所以advanced jsp未能实现)

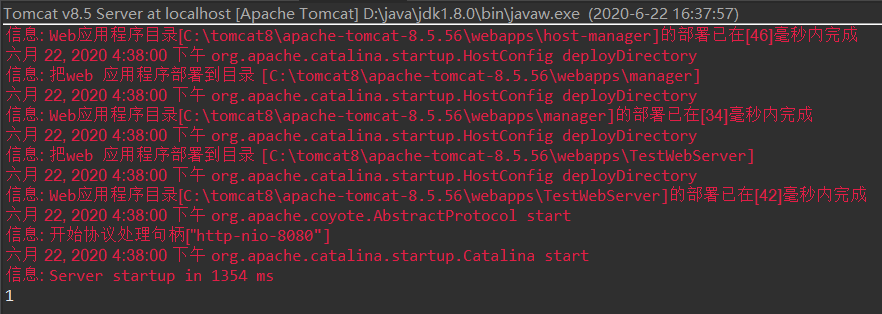
## 5.5 tomcat部署

我们这次试验所要完成的所有功能都是模拟的tomcat服务器，所以我们进行部署的时候，不需要提供任何request或者response或者servlet的代码，因为tomcat自带jar包，我们只提供响应的xml文件需要的对应的类文件，还有相应的静态资源文件。类文件保存在src中

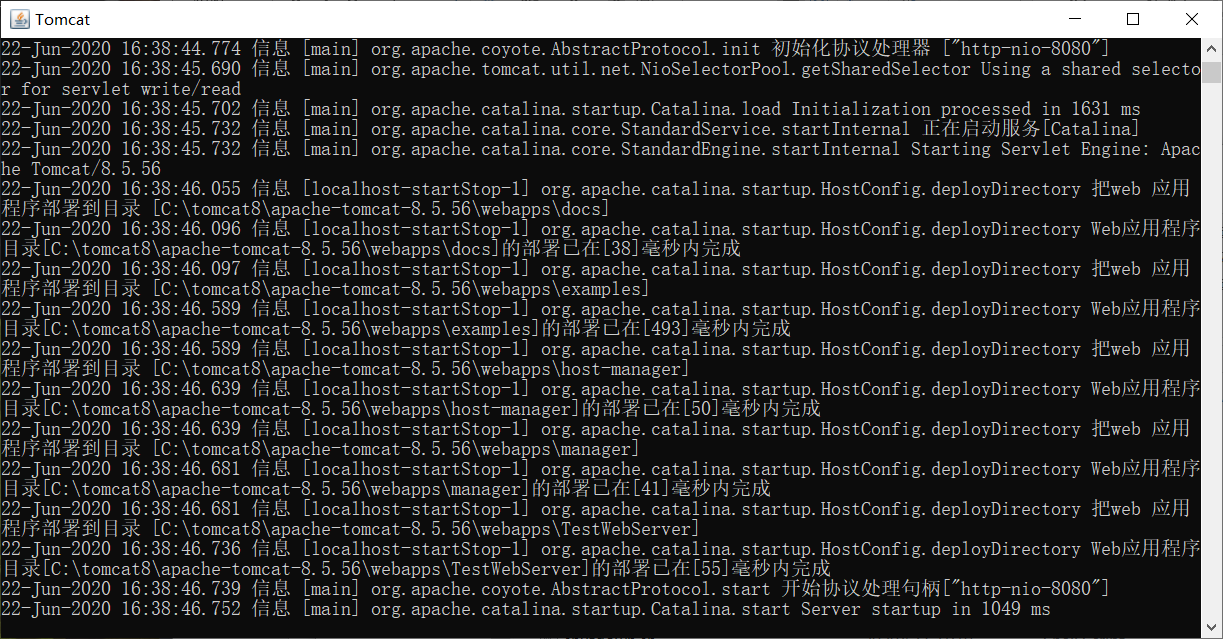
而静态资源则放在META-INF和WEB-INF之外，不然永远请求不到这些资源。WEB-INF中的Web.xml存放servlet的配置文件。

接下来启动tomcat服务器可以用eclipse或者直接通过bat文件打开。

用eclipse打开



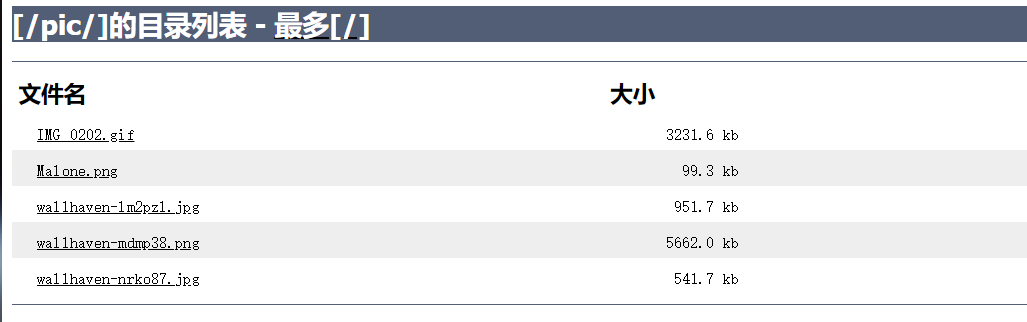
用批处理文件打开

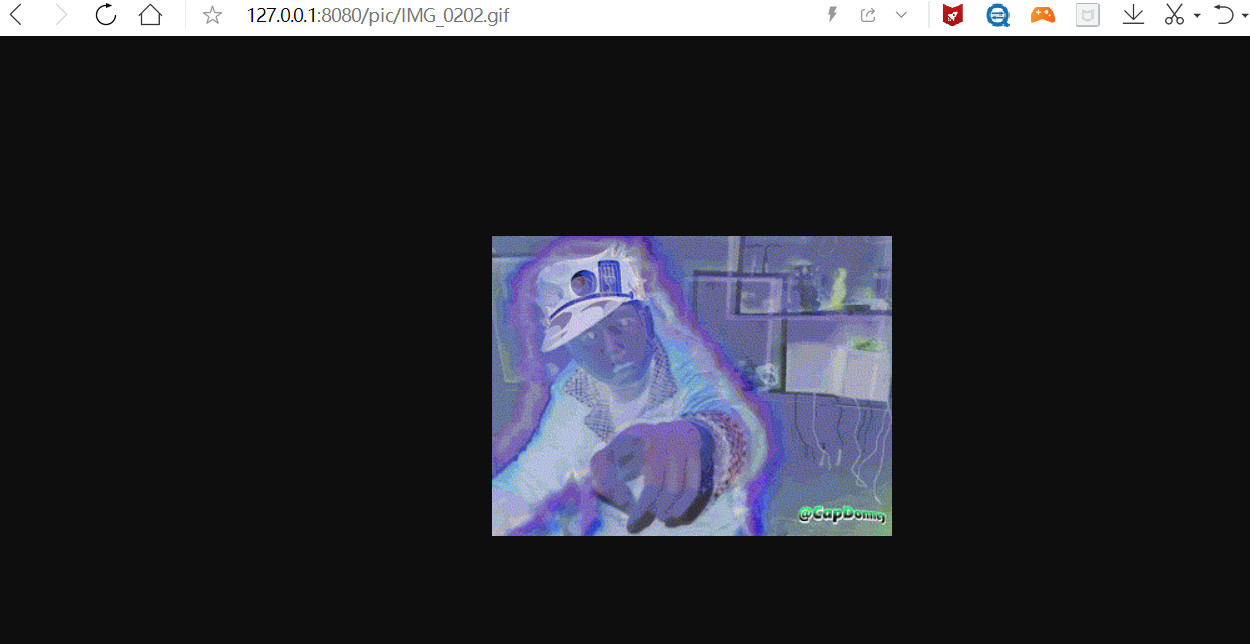


登录页面

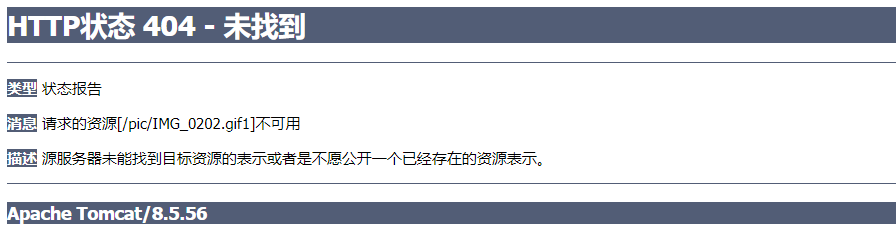


请求资源



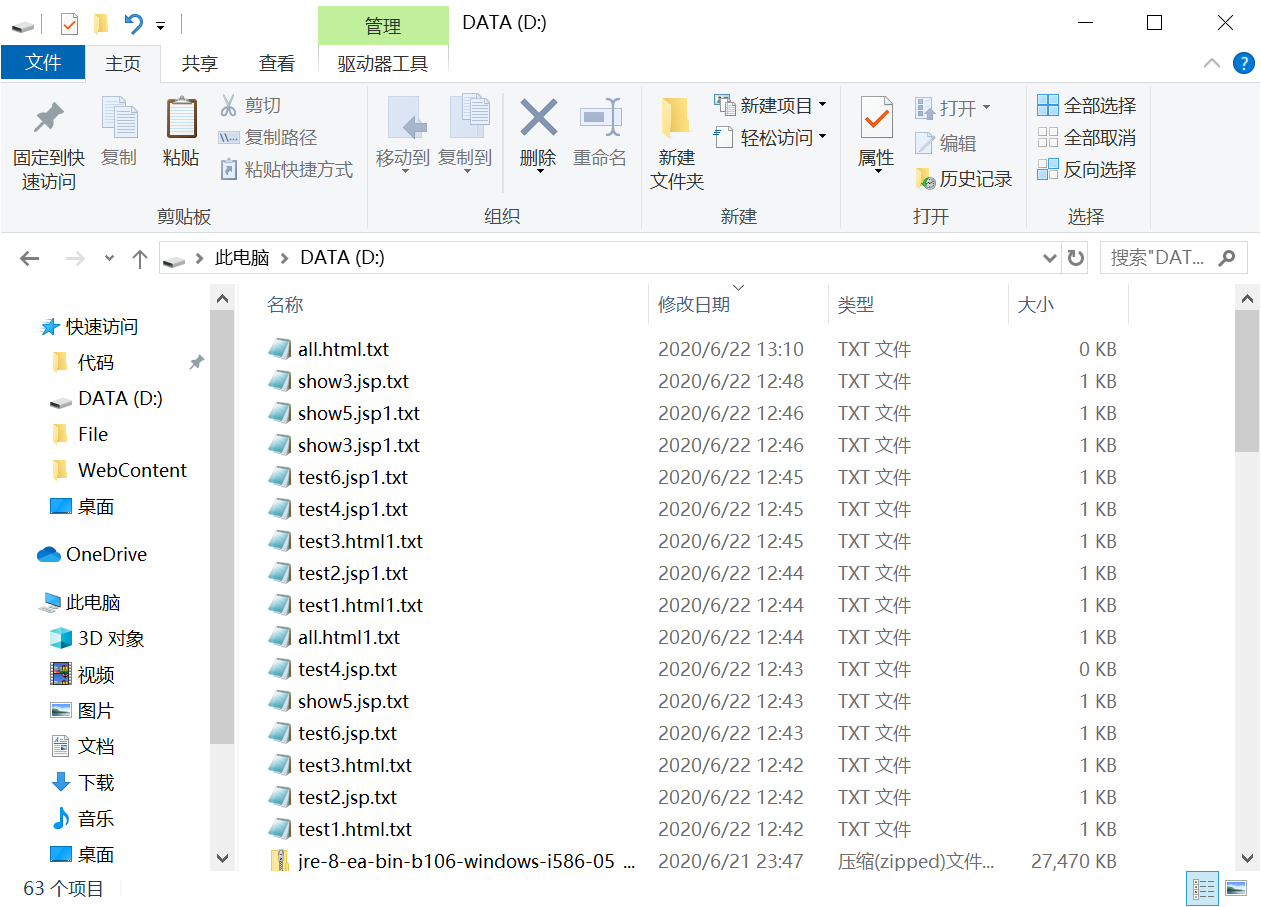


不存在的资源

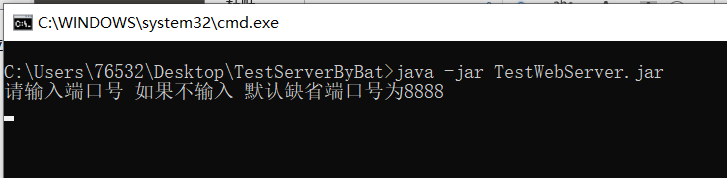


## 5.6 浏览器

手动输入url和端口后爬出来的文件。



## 5.7 bat批处理文件运行



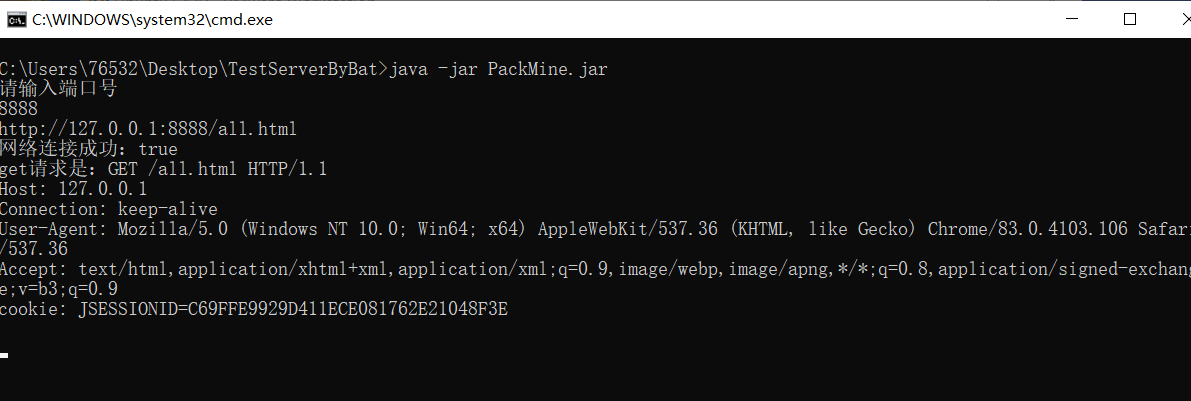
启动服务器。

选择输入端口号或者不输入。



不输入开始监听，默认端口号8888。

启动爬信息和计算相似度bat文件。



返回信息。



服务器也返回信息。

并且在系统磁盘中也保存了爬下来的网页信息。

由于没有能够实现一些功能，所以有些资源信息不能爬下来。

目前能爬下来的文件只有：

all.html

show5.jsp

test1.html

test2.jsp

test3.html

test6.jsp

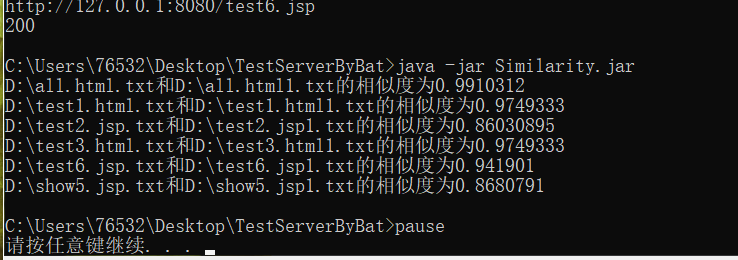
等服务器都爬完这些资源信息之后

开始运行tomcat服务器

然后继续输入端口和网址

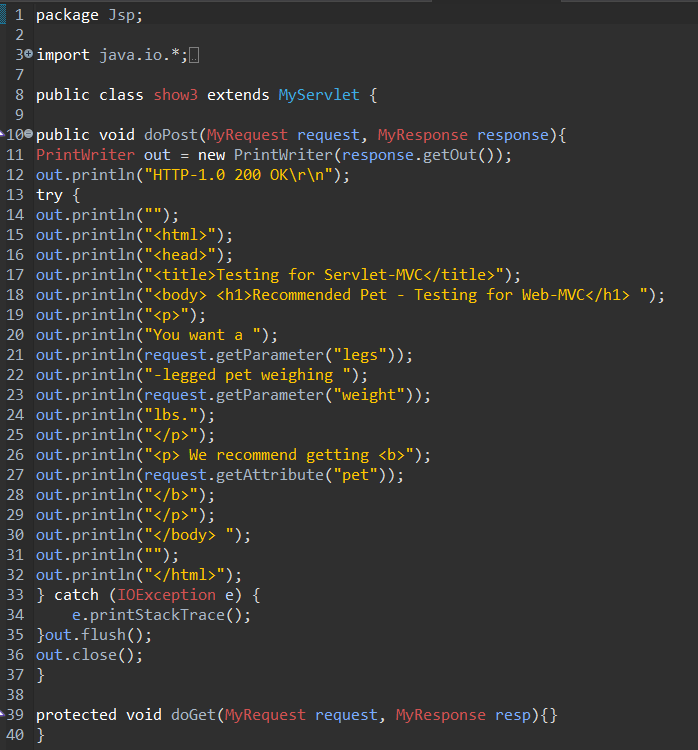
等待输入完成之后，执行相似度jar包。

结果：

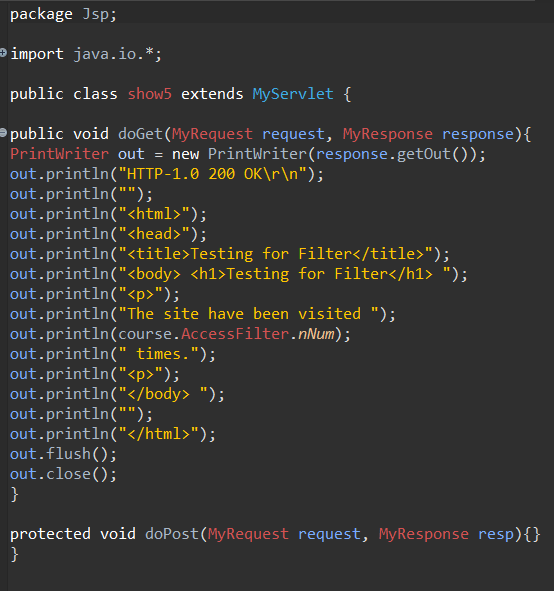


## 5.7 jsp解析产生的相应java代码

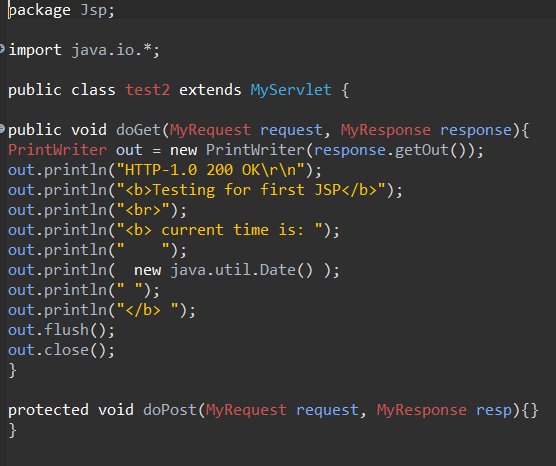
Show3



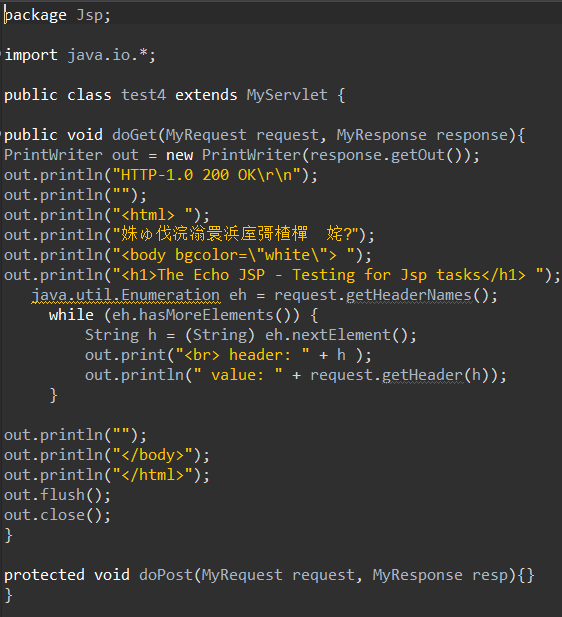
Show5



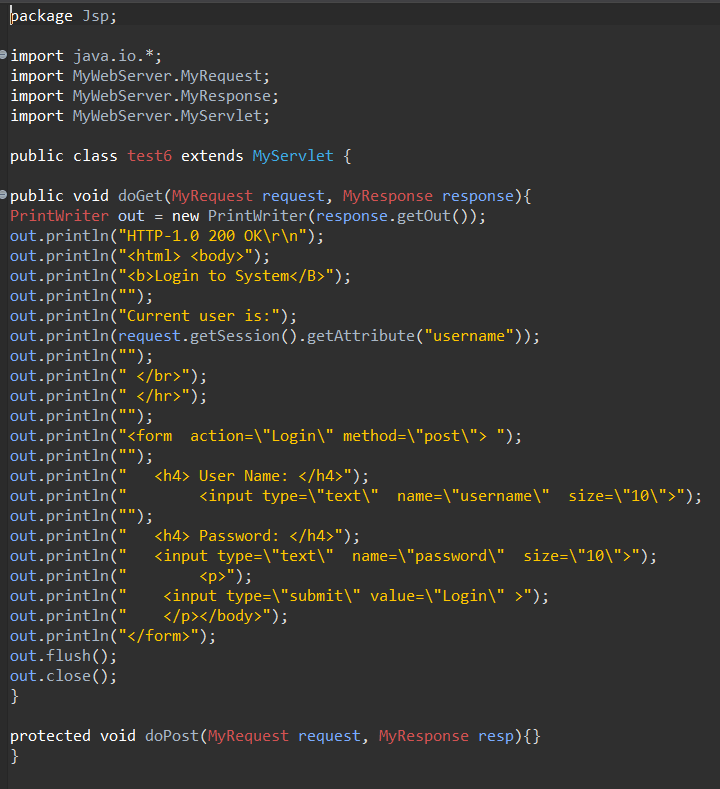
Test2



Test4



Test6



# 实习体会

周麟：这次进行整个服务器的搭建，我深刻明白了每个类之间关系，一个类会对另一个类而造成的影响。比如如果没有xml文件对于servlet的配置，就不会解析出相应的每个请求对应的类，如果没有响应的类，无法利用反射去调用相应的类。如果没有servlet的service的功能，就无法识别出get或者是post的操作。

我还了解到了每次的请求和相应的信息都具有一定的格式，如果格式不对，那么浏览器并不能识别我们所发送的请求，再者我们也不能识别浏览器给我们发送过来的请求。因为当时不了解结构的原因，我在一个地方卡了很久，那就是解析图片的部分，所有的操作都检查了，但是忘记了向服务器发送图片的大小，以至于一直都无法在浏览器中显示图片。

我还了解到了反射，这对于进行开发来说，是一个非常方便的操作。

可以说这次的课程实习对于我来来说，虽然很困难，但是却了解到了很多的之前没有接触过的知识，充实了我对于学习编程的了解。

王晨：1.jsp模块，主要就是了解jsp的语法，解析jsp文件和通过.jsp文件构造.java文件，这里参考了部分网上的代码，只写了一个函数，它所支持的jsp文件的语法有<% %>和<%= %>以及jsp的头。逻辑是，若读取到的一行中不包含任何java语法就直接将它写入.java文件中(当然在此之前要写入一些Java类的前缀，比如开头的包名，类名，类的成员变量和构造函数等)，如读到的这行包含“<%= ”就找到所有对应的<%和%>的位置，处理后写入，除此之外如果包含” <% ”就继续读下面的行，直到读取到的行包含“%>”将两者之间的内容当成Java代码写到.java文件同时创建对应的MyServlet派生类，构造此类的对象，将需要的内容写到response传回客户端。在Servlet MVC中，经过上述操作后还要调用对应Servlet，把数据存入request中，由jsp的类输出对应的值传给客户端。此部分因为对写出java文件的类调试较久，都写在主函数里了，创建对象并提供url和method,直接写出文件检查是哪里的问题，调试起来简单一些，对具体的功能也没有拆分。

2．客户端爬虫，根据我们的服务器的输出，了解了http请求的格式，包括请求的资源、协议、HOST包括后面的浏览器信息和cookie，因为基于httpclient的爬虫无法爬取我们服务器的jsp页面，所以我写了一个基于Socket的爬虫，自己写Http请求头再发给服务器爬取，神奇的是这个socket的爬虫又无法爬取tomcat的页面，所以就写了两个类来分别爬取我们的服务器和tomcat。关于POST，我尝试了用HttpClient来实现，在请求末尾加上封装好的参数，但是不知道是因为header的信息不包含cookie还是因为需要先GET再POST的原因或者是包的版本原因，无论是我们的服务器还是tomcat都无法实现POST请求，后期我又看了我们服务器关于POST请求的输出，尝试了用socket来模拟，但是依旧无法得到动态回馈，我一直用的Login界面测试，id总是返回null，最后未能实现POST。

3.实习心得。这是我第一次通过小组作业来完成一个项目，让我明白了前期的工作分配真的很重要，如果没有给小组成员清晰地分配工作任务规定完成期限，对后面工作的进度有很大的影响。其次就是后期的项目移交和需求沟通，在项目移交的时候要告诉队友项目实现了哪些功能，有哪些地方有问题，测试的时候应该如何输入，为了能让队友明白自己代码的架构，应该尽可能多的写一些注释，对于变量的命名要尽可能的规范。