



软件开发技术课程设计 报告

学 号 : 20171002608

姓 名 : 徐鸿飞

中国地质大学信息工程学院软件工程系

2019 年 6 月

目录

1. 系统概述.....	2
1.1 系统背景.....	2
1.2 系统目标.....	2
1.3 开发环境与工具.....	3
2. 需求分析.....	3
2.1 系统需求概述.....	3
2.2 功能性需求.....	3
2.3 非功能性需求.....	3
3. 概要设计.....	4
3.1 系统总体设计.....	4
3.2 系统功能设计.....	5
4. 详细设计.....	7
4.1 模块接口设计.....	7
5. 系统成果展示.....	10
5.1 支持静态任务.....	10
5.2 支持目录.....	11
5.3 支持图片.....	12
5.4 图形化界面输出日志.....	13
5.5 支持 Servlet.....	14
5.6 支持简单的 Jsp.....	14
5.7 Web-MVC.....	14
5.8 提高的 Jsp 支持.....	15
5.9 Filter 行为支持.....	15
5.10 Session 共享.....	16
6. 实习体会.....	16
6.1 实习过程.....	16
6.2 收获.....	17
6.3 建议.....	17

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]

《Java&.net》是一门实践性较强的软件基础课程，为了学好这门课程，必须在掌握理论知识的同时，加强上机实践。在基本的学习了 Java 与 .net 开发这门课程之后，老师给出了这次的实习内容。

本次实习的核心内容是实现一个简化版的 Tomcat, Tomcat 是由 Apache 软件基金会属下 Jakarta 项目开发的 Servlet 容器，按照 Sun Microsystems 提供的技术规范，实现了对 Servlet 和 Java Server Page (JSP) 的支持，并提供了作为 Web 服务器的一些特有功能,如 Tomcat 管理和控制平台、安全局管理和 Tomcat 阀等。由于 Tomcat 本身也内含了 HTTP 服务器，因此也可以视作单独的 Web 服务器。但是，不能将 Tomcat 和 Apache HTTP 服务器混淆，Apache HTTP 服务器是用 C 语言实现的 HTTP Web 服务器；这两个 HTTP web server 不是捆绑在一起的。Apache Tomcat 包含了配置管理工具，也可以通过编辑 XML 格式的配置文件来进行配置。我们只需要实现其中部分功能。

1.2 系统目标

[概述系统要完成的内容或建设目标，可采用条目描述或表格形式给出]

序号	目标内容
1	静态任务支持
2	图形化界面
3	Servlet 支持
4	简单的 JSP 支持
5	拓展功能实现

1.3 开发环境与工具

序号	类别	具体需求	备注
1	硬件	电脑	
2	软件	IntelliJ IDEA 2019.1.3	

2. 需求分析

2.1 系统需求概述

[对待实现系统做一个概要的高层描述]

实现一个简单的 Servlet 容器，提供对静态任务，Servlet 和简单的 JSP 的支持，并提供图形化界面。

2.2 功能性需求

[列出待实现系统的功能性需求]

- 1) 对静态任务的支持；
- 2) 对 Servlet 的支持；
- 3) 简单的 JSP 支持；
- 4) 图形化界面（包含启动按钮、主目录设置与日志输出）；
- 5) 提供 Filter 行为，Session 的共享。

2.3 非功能性需求

[此处可根据待实现系统的自身特点，对下列条目进行裁剪或补充]

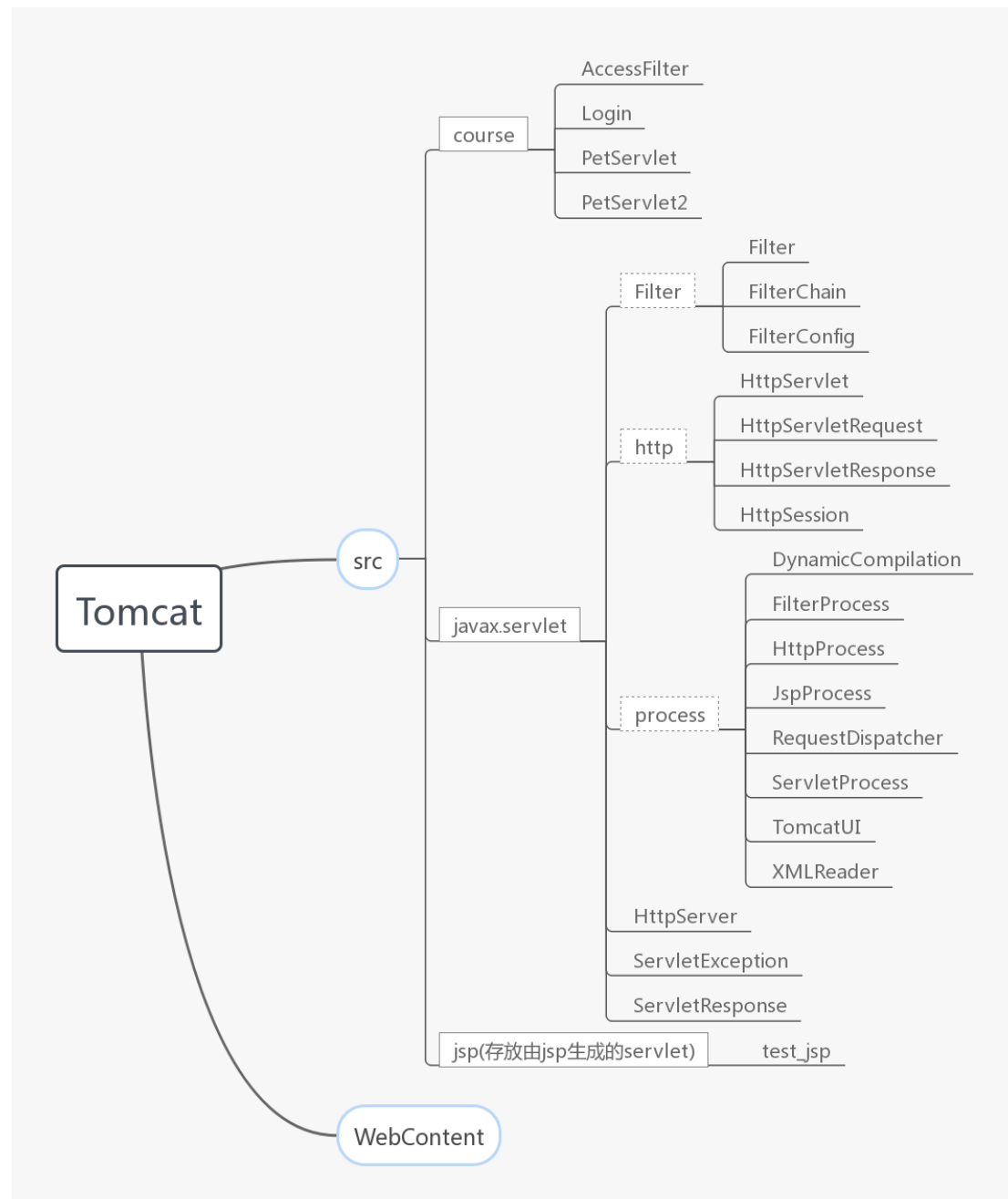
- 1) 安全性；
- 2) 可维护性；
- 3) 易用性；

3. 概要设计

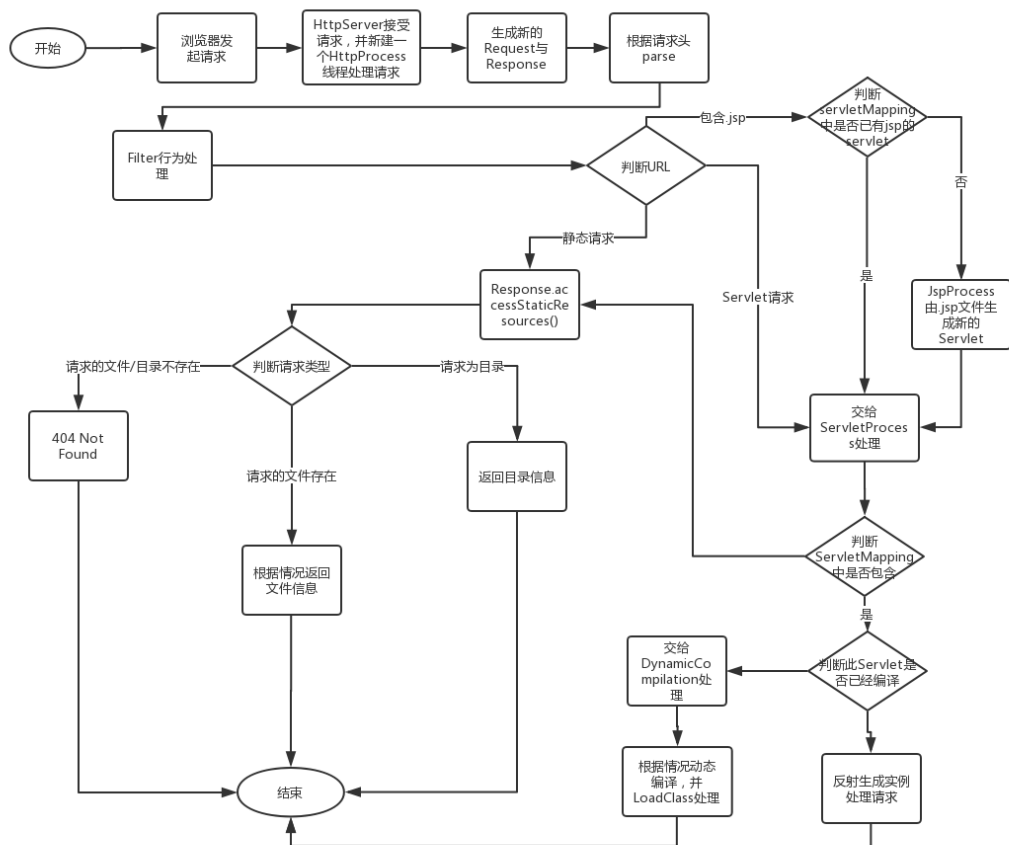
3.1 系统总体设计

[说明待实现系统的总体框架、系统逻辑结构和软件结构架构等。]

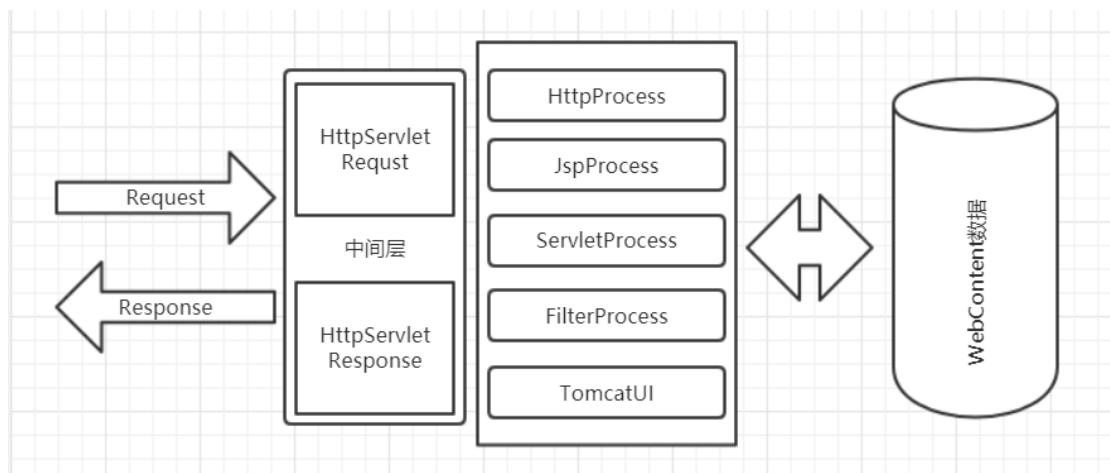
总体框架：



处理流程：

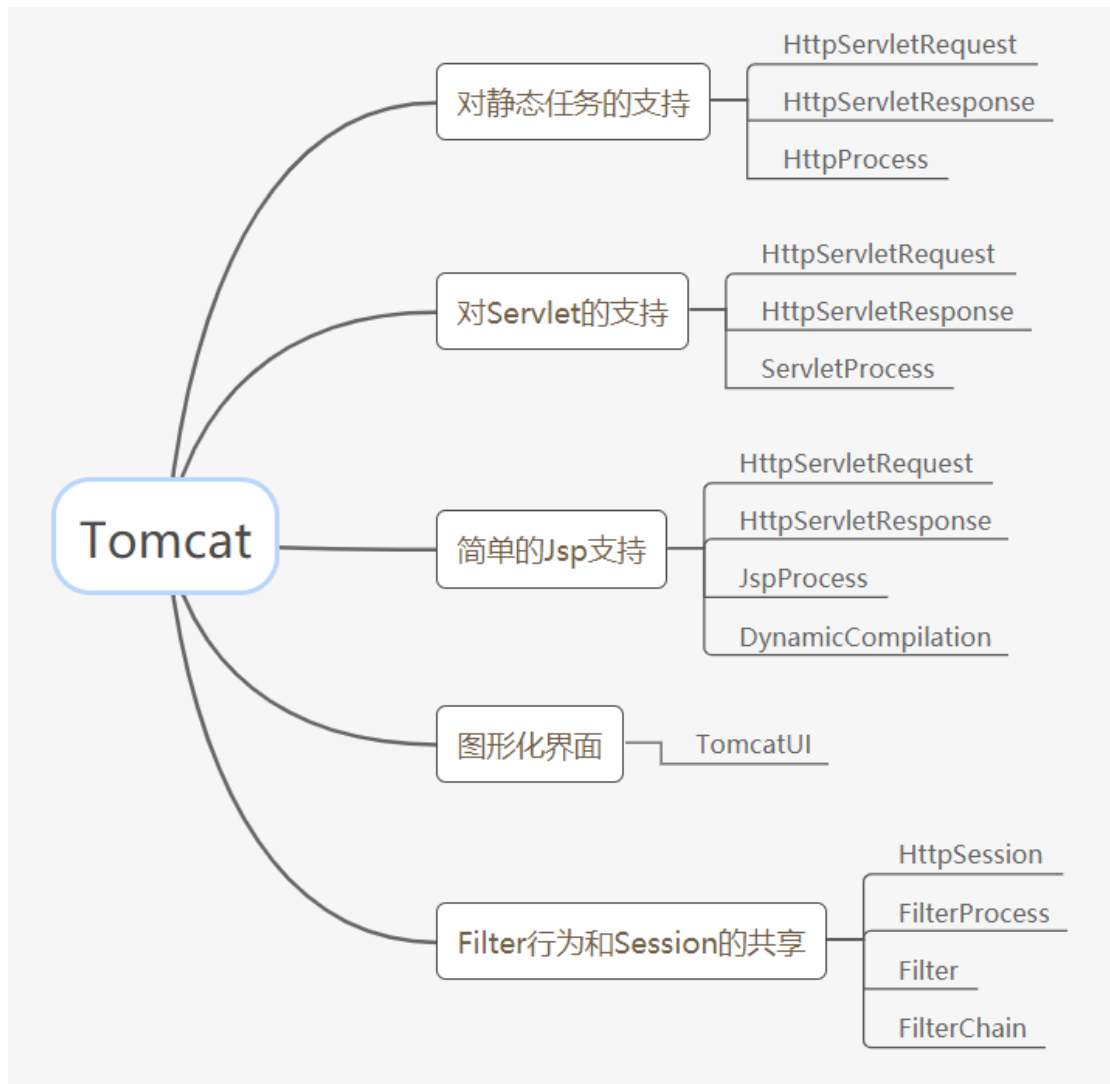


架构设计：

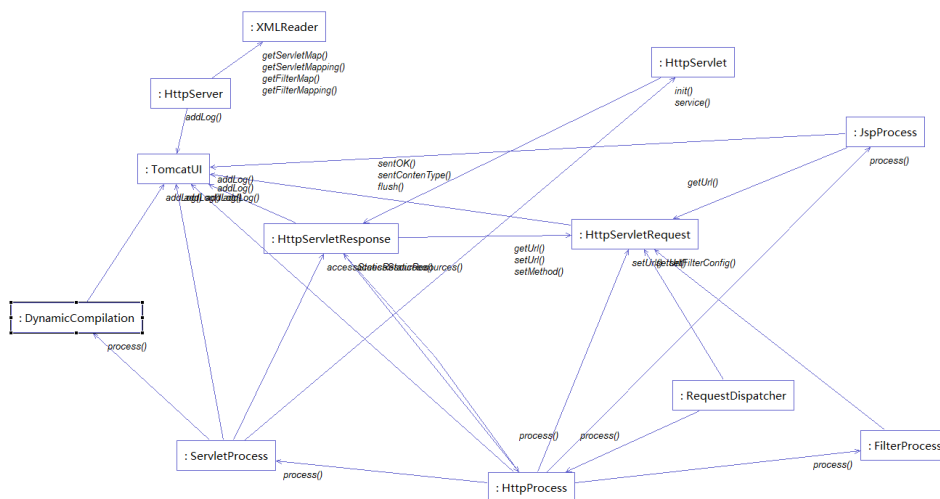


3.2 系统功能设计

[说明系统按照功能划分的总体结构。可用结构图来描述系统的子系统划分情况；如果待实现系统比较简单，则可以直接描述系统中模块间关系的层次。结构图的基本组成部分是模块，模块用来标识一个功能，在结构图中表示了系统的层次关系和调用关系]



调用关系:



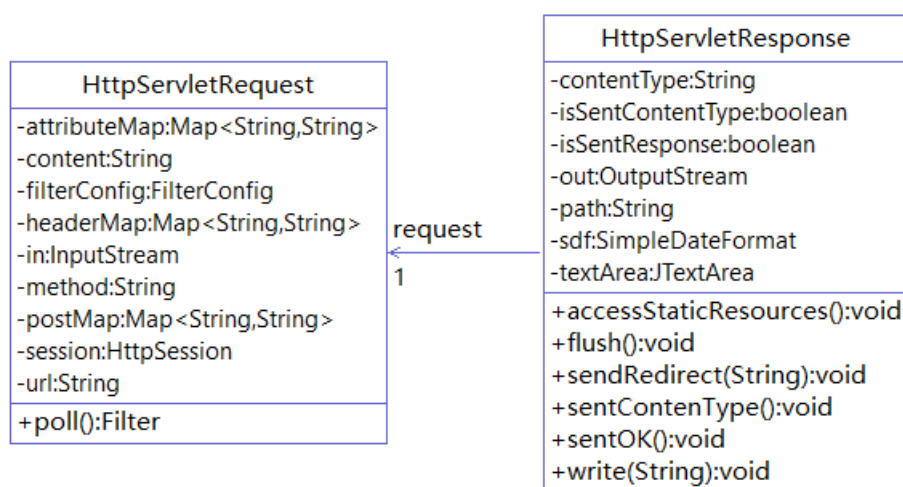
4. 详细设计

4.1 模块接口设计

[对照概要设计中的系统功能设计,依次给出各个子系统中各个模块的接口设计详细说明。可以按照如下章节进行组织,也可以采用 UML 类图加上程序逻辑描述的形式给出。]

[若待实现系统功能较为简单,那么只需给出其中若干关键模块的详细设计说明即可。]

HttpServletRequest 和 HttpServletResponse:



方法概要:

方法概要		
所有方法	实例方法	具体方法
修饰符和类型	方法	说明
java.lang.String	getAttribute(java.lang.String s)	获取属性
java.lang.String	getHeader(java.lang.String s)	返回指定的HTTP头标
java.util.Enumeration<java.lang.String>	getHeaderNames()	返回请求给出的所有HTTP头标名称的枚举值
java.lang.String	getMethod()	
java.lang.String	getParameter(java.lang.String s)	获取参数
RequestDispatcher	getRequestDispatcher(java.lang.String s)	获取RequestDispatcher对象进行请求转发
HttpSession	getSession()	获取HttpSession对象
java.lang.String	getURL()	
boolean	isEmpty()	
Filter	poll()	
void	setAttribute(java.lang.String key, java.lang.String value)	设置属性
void	setFilterConfig(FilterConfig filterConfig)	设置需要的Filter
void	setMethod(java.lang.String method)	设置方法
void	setURL(java.lang.String url)	设置url

修饰符和类型	方法	说明
void	<code>accessStaticResources()</code>	处理静态请求
void	<code>flush()</code>	将缓冲区的内容输出
java.lang.String	<code>getContentType()</code>	
java.io.PrintWriter	<code>getWriter()</code>	获取一个输出流的PrintWriter对象
void	<code>sendRedirect(java.lang.String s)</code>	重定向到指定url
void	<code>sentContentType()</code>	输出文件类型
void	<code>sentOK()</code>	输出状态码
void	<code>setContentType(java.lang.String s)</code>	设置文件类型
void	<code>write(java.lang.String s)</code>	输出一个字符串

HttpProcess:

修饰符和类型	方法	说明
static void	<code>process(HttpServletRequest request, HttpServletResponse response)</code>	对不同类型链接分别进行处理
void	<code>run()</code>	

ServletProcess:

修饰符和类型	方法	说明
void	<code>process(HttpServletRequest request, HttpServletResponse response)</code>	处理链接为Servlet的请求

JspProcess:

修饰符和类型	方法	说明
private void	<code>end()</code>	处理尾部
private void	<code>head()</code>	处理头部
static void	<code>main(java.lang.String[] args)</code>	
private void	<code>parser()</code>	解析jsp文件
void	<code>process(HttpServletRequest request, HttpServletResponse response)</code>	处理请求为jsp的文件
private void	<code>readFile(java.io.File jspFile)</code>	读jsp文件
private void	<code>removePage()</code>	移除page
void	<code>setMethod(java.lang.String s)</code>	设置方法
private void	<code>updateInfo()</code>	修改相关属性
private void	<code>writeJsp(java.lang.String s)</code>	写入动态内容
private void	<code>writeStatic(java.lang.String s)</code>	写静态html内容

DynamicCompilation:

构造器		说明
DynamicCompilation(java.lang.String path)		动态编译指定的文件,若已经便宜,则不再编译
修饰符和类型	方法	说明
void	process(HttpServletRequest request, HttpServletResponse response)	ClassLoader 反射动态编译的类

TomcatUI:提供界面化

TomcatUI
-fileChooser:JFileChooser
-httpServer:HttpServer
-port:int
-run:boolean
-runButton:JButton
-stopBtn:JButton

Filter 行为: FilterChain 采用责任链设计模式达到对不同过滤器的执行首先 FilterChain 会调用它重写 FilterChain 的 doFilter 方法, 然后 doFilter 里面会调用 doFilter (request, response) 方法; 该方法使过滤器容器拿到每个过滤器, 然后调用它们重写 Filter 接口里面的 dofilter 方法。

Filter
«Interface»
+destroy():void
+doFilter(ServletRequest, ServletResponse, FilterChain):void
+init(FilterConfig):void

FilterChain
+doFilter(ServletRequest, ServletResponse):void

FilterConfig
~filter:Queue<Filter>
+poll():Filter

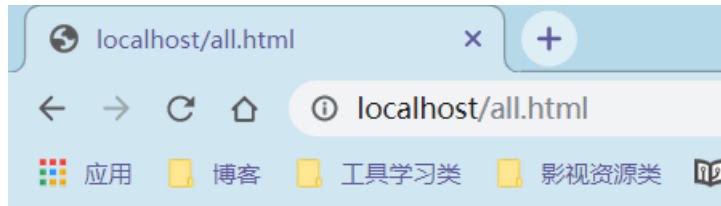
XMLReader 读取 xml 配置: 使用了 org.w3c.dom 进行解析。

java.util.Map<java.lang.String, java.lang.String>	getFilterMap()	获取filterMap
java.util.Map<java.lang.String, java.lang.String>	getFilterMapping()	获取filterMapping
java.util.Map<java.lang.String, java.lang.String>	getServletMap()	获取ServletMap
java.util.Map<java.lang.String, java.lang.String>	getServletMapping()	获取ServletMapping
static void	main (java.lang.String[] args)	
private void	readXMLFile (java.io.File file)	读XML文件进行解析

5. 系统成果展示

[对照概要设计中的系统功能设计，依次给出各个子系统中各个模块的运行结果和解释说明]

5.1 支持静态任务



Test Cases

[Test for files list](#)

[Test for Servlet](#)

[Test for Simple Jsp](#)

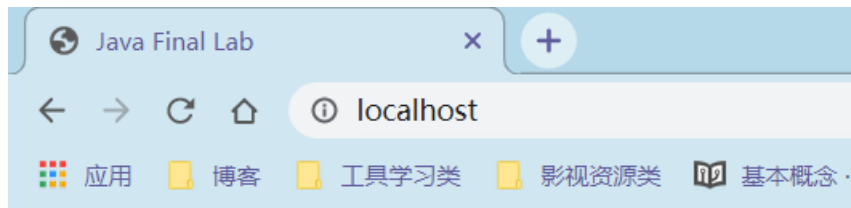
[Test for Servlet MVC](#)

[Test for advanced Jsp](#)

[Test for Filter](#)

[Test for Session](#)

5.2 支持目录



/的索引

名称	大小	修改日期
404.html	239B	2019/06/23 16:04:28
all.html	511B	2019/06/23 16:04:28
META-INF	文件夹	2019/06/24 01:01:16
show3.jsp	432B	2019/06/23 16:04:28
show5.jsp	297B	2019/06/23 16:04:28
test1.html	522B	2019/06/23 16:04:28
test2.jsp	101B	2019/06/23 16:04:28
test3.html	547B	2019/06/23 16:04:28
test4.jsp	462B	2019/06/23 16:04:28
test6.jsp	416B	2019/06/23 16:04:28
web	文件夹	2019/06/24 01:01:16

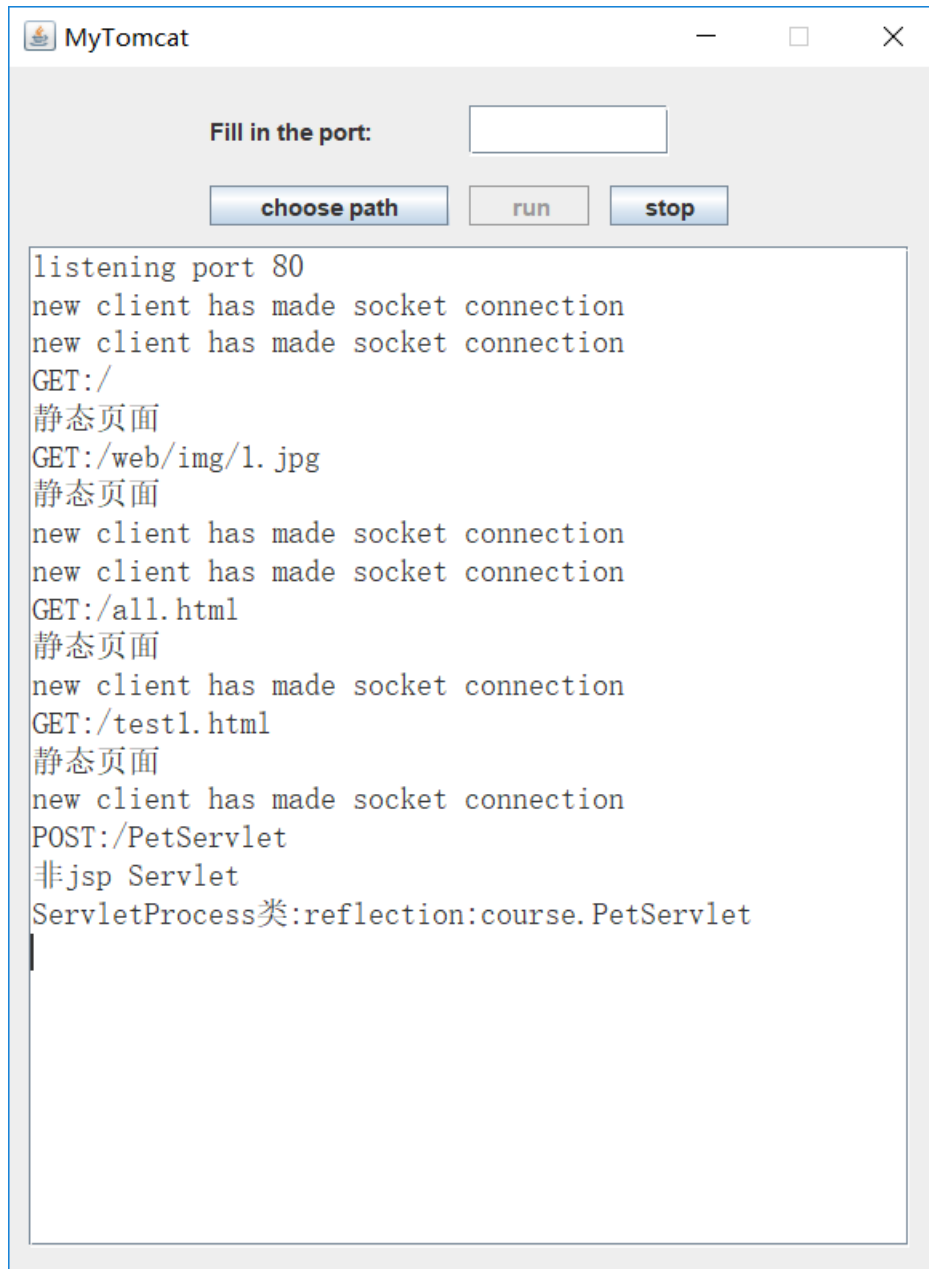
5.3 支持图片



Final Java Lab :a



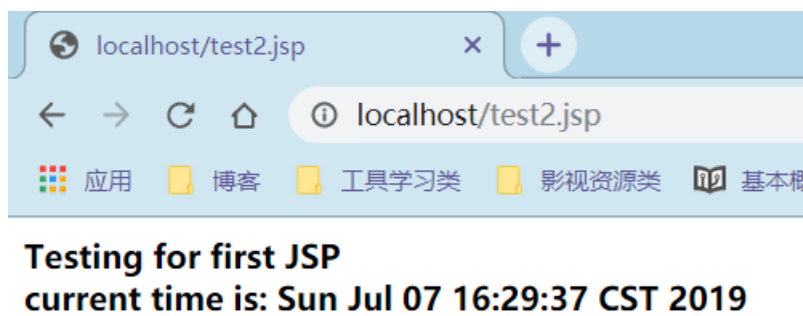
5.4 图形化界面输出日志



5.5 支持 Servlet



5.6 支持简单的 Jsp



5.7 Web-MVC



5.8 提高的 Jsp 支持

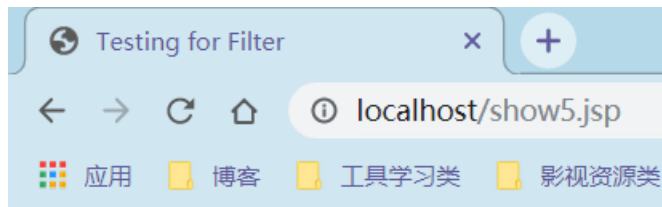


此工作属于提高要求

The Echo JSP - Testing for Jsp tasks

header: Cookie value: Webstorm-bbf7ba1a=5e3f036d-b282-4950-a2ac-406aeeb41500
header: Accept value: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
header: Upgrade-Insecure-Requests value: 1
header: Connection value: keep-alive
header: User-Agent value: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
header: Referer value: http
header: Host value: localhost
header: Accept-Encoding value: gzip, deflate, br
header: Accept-Language value: zh-CN,zh;q=0.9

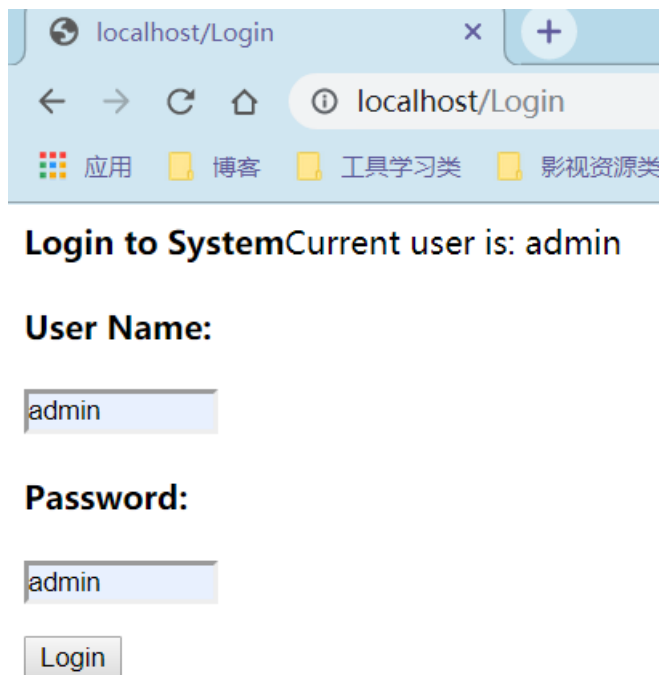
5.9 Filter 行为支持



Testing for Filter

The site have been visited 14 times.

5.10 Session 共享



localhost/Login

localhost/Login

应用 博客 工具学习类 影视资源类

Login to SystemCurrent user is: admin

User Name:

admin

Password:

admin

Login

6. 实习体会

[给出综合实习过程中实习体会的汇总]

主要讲遇到的困难，如何解决的，收获、建议等

6.1 实习过程

写这次实习的大概的一个过程是：刚开始不知道怎么做，看了老师给的文档有了思路：

先是对静态页面的一个支持，在这之前我们需要先封装 `request` 和 `response`，`request` 接受请求并解析，`response` 处理请求。

对于静态页面，我们只需要读取静态页面中的内容然后加上头发给浏览器就可以了，比较简单，刚开始遇到了图片乱码问题，然后发现是 `ContentType` 的原因，得以解决。

对于目录的支持：我选择的是打开这个目录，然后遍历目录下的文件与文件夹，然后动态生成一个包含目录的 `html` 页面。

之后我就开始写了界面，其中给 `TextArea` 加上滚动条花了一番功夫。

对于 `Servlet` 的实现，就是把 `request` 和 `response` 给 `Servlet`，让 `Servlet` 根据配置去回应请求。然后这其中又涉及到 `XML` 文件的解析，刚开始想用正则表达式解析，但是特别慢，然后中间上课又讲到有可以解析 `xml` 文件的库，最后参

考一篇 blog 用 w3c.dom 库解析 xml。

对于 jsp 的支持，jsp 本质上也是一个 Servlet，所以主要的工作就是怎么把一个 jsp 文件转化成一个 Servlet 类。要解析 jsp 中的内容，首先是 jsp 中的 html 静态内容，直接 `out.println("\" + s + "\");`；但是要注意一个问题，页面中有可能会有引号，我们需要加入转义符，刚开始生成的 Servlet 类总是有莫名的换行，推测可能是编码的问题，后面改用 `BufferedReader` 读取 jsp 又没有问题了。然后是 Jsp 中的标签内容，首先了解到 jsp 中有一些内置对象，我们用到的有 `request`，`response`，`page`，`out`；处理表达式：等效于 jsp 内置对象 `out` 的 `print()` 方法，其中 jsp 的内置对象本应是 `jspWriter` 类型，我这里就直接用了 `PrinterWriter`，区别不大；对于脚本元素：直接写入即可。

然后在 jsp 中还涉及到一个动态编译的过程，因为在运行过程中生成的 Servlet 没有被编译生成字节码，无法利用反射处理，所以我这里使用的是 Java 的 `ToolProvider` 提供的 `javax.tools.JavaCompiler` 用于动态编译 Servlet，然后用 `URLClassLoader` 加载。其中遇到了 `NoClassDefFoundError` 的错误，解决方法是不要在 url 里加包名而是在类名上加包名（但是我在另一个测试文件里是 url 加包名，类名不加包名正确，反之 `NoClassDefFoundError`，玄学？？）。

对与 Filter 行为的支持，我这里的处理与 Servlet 类似，先找出符合请求的 Filter，反射得到对象，然后加入 `FilterConfig` 最后通过过滤链处理。对于 Session 的共享中，只进行了简单的实现。

6.2 收获

本次实习基本了解了 Tomcat 的框架与实现，知道了如何去实现一个服务器，同时也增强了对 Java Socket 编程与多线程的实现的能力，了解到更多的 Java 库的强大（`ToolProvider`、`JavaCompiler`、`org.w3c.dom`……）；同时也知道了规范编码的重要性。

6.3 建议

实习涉及内容广泛，但是还是只这一个题，希望可以是像计网一样多个题目可供选择。