PaperTime检测报告简明打印版

相似度：42.74%
编　号：0UCFCKRD87BDACJ9
标　题：基于Java的Web服务器设计与开发
作　者：金明涛
长　度：21868字符
时　间：2018-05-14 01:20:45
比对库：本地库（学术期刊、学位论文、会议论文）；PaperTime云论文库；互联网

本地库相似资源（学术期刊、学位论文、会议论文）
1. 相似度：1.84%　篇名：《JAVA网络编程的探讨》
来源：《煤炭技术》　年份：2012　作者：钱娟
2. 相似度：1.04%　篇名：《Web服务器的架设与应用》
来源：《科技创新导报》　年份：2013　作者：张苏颖
3. 相似度：0.67%　篇名：《消息导语结构探研》
来源：《采写编》　年份：2016　作者：胡勇
4. 相似度：0.62%　篇名：《Web服务器的架设与应用》
来源：《科技创新导报》　年份：2013　作者：张苏颖
5. 相似度：0.48%　篇名：《面向TCP/IP网络编程实验平台的设计》
来源：《通信技术》　年份：2015　作者：张爱科
6. 相似度：0.40%　篇名：《web服务器的架设与应用》
来源：《科技与生活》　年份：2017　作者：马杰
7. 相似度：0.27%　篇名：《Web技术教学改革探讨》
来源：《钦州学院学报》　年份：2014　作者：李丹
8. 相似度：0.21%　篇名：《访问Web服务器》
来源：《网管员世界》　年份：2011　作者：黄东
9. 相似度：0.21%　篇名：《利用VPN远程访问图书馆内网数字资源途径初探》
来源：《科技情报开发与经济》　年份：2013　作者：宿大东
10. 相似度：0.15%　篇名：《浅析STP协议》
来源：《中国科技纵横》　年份：2015　作者：陈建伟
11. 相似度：0.14%　篇名：《基于物联网技术的智能变电站在线监测系统的设计》
来源：《仪表技术与传感器》　年份：2015　作者：何方
12. 相似度：0.13%　篇名：《移动互联网创新业务模式分析》
来源：《科技创新导报》　年份：2014　作者：金晶
13. 相似度：0.13%　篇名：《WEB服务器安全防范》
来源：《现代企业文化》　年份：2011　作者：陈涛
14. 相似度：0.12%　篇名：《入侵Tomcat服务器》
来源：《黑客防线》　年份：2015　作者：彭文波
15. 相似度：0.11%　篇名：《基于WinSock的局域网聊天室设计与实现》
来源：《数字技术与应用》　年份：2013　作者：陈健苇
16. 相似度：0.11%　篇名：《新媒体环境中基于Java技术的信息系统初探》
来源：《市场周刊》　年份：2014　作者：金兴
17. 相似度：0.11%　篇名：《Tomcat结构分析》
来源：《教学与科技》　年份：2013　作者：蒋波
18. 相似度：0.10%　篇名：《分布式负载均衡的java实现》
来源：《北京邮电大学硕士论文》　年份：2017　作者：邹方曼
19. 相似度：0.10%　篇名：《《网络基础知识》教学案例》
来源：《中小学电教：上》　年份：2011　作者：梁阳
20. 相似度：0.09%　篇名：《基于Web的温室远程监控的设计》
来源：《微型机与应用》　年份：2013　作者：李星沛
21. 相似度：0.08%　篇名：《用Hessian协议实现3G终端接入图书馆系统的研究》

来源：《图书馆理论与实践》 年份：2013 作者：孙欣
22. 相似度：0.08% 篇名：《面向接口编程的魅力》
来源：《程序员》 年份：2015 作者：杜玄

PaperTime云论文库(知网，万方，维普，百度文库等镜像）

1. 相似度：1.15% 标题：《轻量级Web服务器的实现与应用_图文_百度文库》
来源：http://wenku.baidu.com/view/5177fd7ba26925c52cc5bfcb.html
2. 相似度：1.04% 标题：《web服务器和应用服务器_百度文库》
来源：http://wenku.baidu.com/view/e882863ecdbff121dd36a32d7375a417866fc16e.html
3. 相似度：0.97% 标题：《HTTP协议入门_教育指南_百度教育攻略》
来源：http://wenku.baidu.com/view/39572641793e0912a21614791711cc7931b778f2.html
4. 相似度：0.73% 标题：《基于Java的高效兼容HTTP服务器的研究及实现.doc_淘豆网》
来源：https://www.taodocs.com/p-96983674.html
5. 相似度：0.39% 标题：《A Performance Study of Monitoring and Information Services for ...》
来源：http://wenku.baidu.com/view/8c0d9b61caaedd3383c4d38b.html
6. 相似度：0.35% 标题：《3.3连接器_图文_百度文库》
来源：http://wenku.baidu.com/view/ff093cbe08a1284ac950435d.html
7. 相似度：0.25% 标题：《JAVA语言项目11 JAVA网络编程 - 道客巴巴》
来源：http://www.doc88.com/p-591324279913.html
8. 相似度：0.21% 标题：《how-tomcat-works_图文_百度文库》
来源：http://wenku.baidu.com/view/6ca866150b4e767f5acfceec.html
9. 相似度：0.21% 标题：《http头部信息_百度文库》
来源：http://wenku.baidu.com/view/e92eb066f5335a8102d22004.html?from=rec&pos=3
10. 相似度：0.19% 标题：《C Socket_百度文库》
来源：http://wenku.baidu.com/view/737131244b35eefdc8d333a4.html
11. 相似度：0.19% 标题：《实验四 客户服务器通信_百度文库》
来源：http://wenku.baidu.com/view/4f53a378bd64783e08122b47.html
12. 相似度：0.11% 标题：《web服务器_图文_百度文库》
来源：http://wenku.baidu.com/view/b3c5771efc4ffe473368abb5.html

互联网相似资源(博客，百科，论坛，新闻等）

1. 相似度：9.90% 标题：《HTTP协议发展介绍 - 飞鱼湾 - 博客园》
来源：https://www.cnblogs.com/fishbay/p/7209696.html
2. 相似度：1.84% 标题：《java socket编程- linzheng - 博客园》
来源：http://www.cnblogs.com/linzheng/archive/2011/01/23/1942328.html
3. 相似度：1.82% 标题：《http协议简介 - CSDN博客》
来源：https://blog.csdn.net/viewcode/article/details/44779049
4. 相似度：1.61% 标题：《45、java网络编程 - CSDN博客》
来源：https://blog.csdn.net/kaoa000/article/details/8507403
5. 相似度：1.44% 标题：《HTTP协议小结 - CSDN博客》
来源：https://blog.csdn.net/xsf50717/article/details/47277347
6. 相似度：1.34% 标题：《web服务器和应用服务器的区别_百度知道》
来源：https://zhidao.baidu.com/question/2057633545419583747.html
7. 相似度：1.33% 标题：《HTTP协议 - foodoir - 博客园》
来源：http://www.cnblogs.com/foodoir/p/5905946.html
8. 相似度：1.13% 标题：《HTTP 消息结构 | 菜鸟教程》
来源：http://www.runoob.com/http/http-messages.html
9. 相似度：1.07% 标题：《header标头说明- ThinkPHP框架》
来源：http://www.thinkphp.cn/topic/40412.html
10. 相似度：1.07% 标题：《HTTP 协议简介》
来源：http://www.360doc.cn/article/17799864_400329626.html
11. 相似度：0.97% 标题：《HTTP基本认识 - 简书》
来源：https://www.jianshu.com/p/5c21f75d2120
12. 相似度：0.81% 标题：《http header 包括哪些参数_百度知道》

来源：https://zhidao.baidu.com/question/1609682123595808667
13. 相似度：0.75% 标题：《http协议是什么协议_百度知道》
来源：https://zhidao.baidu.com/question/1881940426315746908.html
14. 相似度：0.68% 标题：《HTTP请求模型、头信息参考、aardio开发实例 -》
来源：http://bbs.aardio.com/forum.php?mod=viewthread&tid=6214
15. 相似度：0.67% 标题：《HTTP 十分钟教程 - CSDN博客》
来源：https://blog.csdn.net/hunter_wyh/article/details/51525292
16. 相似度：0.67% 标题：《HTTP协议学习总结 - CSDN博客》
来源：https://blog.csdn.net/hytgxljh/article/details/52445286
17. 相似度：0.55% 标题：《HTTP协议学习总结 - 越努力,越幸运! - CSDN博客》
来源：https://blog.csdn.net/liujie19901217/article/details/52385865?locationNum=8
18. 相似度：0.55% 标题：《毕业论文(设计)基于Java的高效兼容HTTP服务器的研究及实现.doc》
来源：https://max.book118.com/html/2016/0516/43083161.shtm
19. 相似度：0.52% 标题：《HTTP请求头大全 - 常用参考表对照表 - 脚本之家在线工具》
来源：http://tools.jb51.net/table/http_header
20. 相似度：0.51% 标题：《Http请求中Content》
来源：http://www.360doc.com/content/17/0711/16/29229598_670574320.shtml
21. 相似度：0.46% 标题：《...Socket】TCP Socket通信中由read返回值造成的的死锁问题（含代码...》
来源：http://www.importnew.com/20151.html
22. 相似度：0.44% 标题：《如何用eclipse创建你的第一个servlet小程序- CSDN博客》
来源：https://blog.csdn.net/shiwy_ab/article/details/77040061
23. 相似度：0.39% 标题：《...popularapplicationofInternettechnology,hasbeen..._百度作业帮》
来源：https://www.zybang.com/question/0c0f0d986b8e210125c08d354dd486d7.html
24. 相似度：0.37% 标题：《HTTP超文本传输协议》
来源：http://www.360doc.com/content/07/0805/17/36481_654998.shtml
25. 相似度：0.37% 标题：《HTTP协议详解 - jaky666 - 博客园》
来源：http://www.cnblogs.com/jaky666/articles/network_1.html
26. 相似度：0.34% 标题：《关于HTTP协议,一篇就够了 - 简书》
来源：https://www.jianshu.com/p/80e25cb1d81a
27. 相似度：0.32% 标题：《HTTP Header 详解_突发妄想_新浪博客》
来源：http://blog.sina.com.cn/s/blog_15b95c3150102w5av.html
28. 相似度：0.30% 标题：《Web服务器与应用服务器,Apache & Tomcat & JBoss & Weblo..._博客园》
来源：https://www.cnblogs.com/JCSU/articles/2391979.html
29. 相似度：0.29% 标题：《HTTP的Accept Encoding》
来源：https://www.aliyun.com/jiaocheng/439181.html
30. 相似度：0.28% 标题：《管道是如何处理HTTP请求的？ - HackerVirus - 博客园》
来源：http://www.cnblogs.com/Leo_wl/p/5544342.html
31. 相似度：0.28% 标题：《127.0.0.1和0.0.0.0地址的区别 - CSDN博客》
来源：https://blog.csdn.net/ythunder/article/details/61931080
32. 相似度：0.27% 标题：《Jetty构架 - Java - IT610.com》
来源：http://www.it610.com/article/2257312.htm
33. 相似度：0.27% 标题：《...Integrated Systems - Fire, Security, Communications ...》
来源：http://www.riverside-is.com/
34. 相似度：0.25% 标题：《web 容器】 - CSDN博客》
来源：https://blog.csdn.net/zhangsunxiaobai/article/details/53019519
35. 相似度：0.25% 标题：《HTTP响应报文与工作原理详解_网络_比特网》
来源：http://network.chinabyte.com/401/13238901.shtml
36. 相似度：0.23% 标题：《http协议详解 - CSDN博客》
来源：https://blog.csdn.net/anlina_1984/article/details/5620456
37. 相似度：0.21% 标题：《Socket如何保证长连接 - CSDN博客》
来源：https://blog.csdn.net/liaomengge/article/details/50760834
38. 相似度：0.21% 标题：《技术人生-RESTful中GET,POST,PUT,DELETE这么多的请求方式,到底该...》
来源：http://blog.360chwl.net/detail/8a2390184d76d30e014d7a2926cd0004.html
39. 相似度：0.21% 标题：《TCP连接、HTTP连接与Socket连接的区别- CSDN博客》
来源：https://blog.csdn.net/min996358312/article/details/68969519

40. 相似度：0.21% 标题：《OutputStream的flush()方法?什么调用?-CSDN论坛》
来源：https://bbs.csdn.net/topics/370245823
41. 相似度：0.21% 标题：《网络基础知识 - HTTP协议 - 战争热诚 - 博客园》
来源：http://www.cnblogs.com/wj-1314/p/8366776.html
42. 相似度：0.20% 标题：《默认连接器 - beggar_1982 - 博客园》
来源：https://www.cnblogs.com/laoxia/p/8097923.html
43. 相似度：0.20% 标题：《深度剖析tomcat简介 - CSDN博客》
来源：https://blog.csdn.net/fengzijinliang/article/details/51728134
44. 相似度：0.19% 标题：《了解http - A_bet_of_three_years的博客 - CSDN博客》
来源：https://blog.csdn.net/a_bet_of_three_years/article/details/78644835
45. 相似度：0.18% 标题：《使用VS Code 开发.NET Core 应用程序 部署到Linux 跨平台..._博客园》
来源：https://www.cnblogs.com/linezero/p/5460759.html
46. 相似度：0.17% 标题：《Socket简介及客户端服务器连接实例 - CSDN博客》
来源：https://blog.csdn.net/qq_32319583/article/details/53523023
47. 相似度：0.17% 标题：《深入剖析tomcat之servlet容器 - CSDN博客》
来源：https://blog.csdn.net/chenleixing/article/details/48153235
48. 相似度：0.17% 标题：《HTTP响应头和请求头信息对照 - CSDN博客》
来源：https://blog.csdn.net/xiongchun11/article/details/53035497
49. 相似度：0.17% 标题：《javaweb学习总结(五)——Servlet开发(一) - 孤傲苍狼 - 博客园》
来源：https://www.cnblogs.com/xdp-gacl/p/3760336.html
50. 相似度：0.16% 标题：《Servlet规范 - 谢多 - 博客园》
来源：http://www.cnblogs.com/xieduo/articles/822091.html
51. 相似度：0.16% 标题：《Android:这是一份很详细的Socket使用攻略 - CSDN博客》
来源：https://blog.csdn.net/qq_38729449/article/details/79416457
52. 相似度：0.16% 标题：《深入理解 Tomcat(八)源码剖析之连接器 - CSDN博客》
来源：https://blog.csdn.net/qq_38182963/article/details/78660790
53. 相似度：0.16% 标题：《web开发中 web 容器的作用(如tomcat) - 简书》
来源：https://www.jianshu.com/p/99f34a91aefe
54. 相似度：0.16% 标题：《一张图说明访问网站的流程 - CSDN博客》
来源：https://blog.csdn.net/sunmc1204953974/article/details/49963311
55. 相似度：0.15% 标题：《三次握手协议_百度百科》
来源：
https://baike.baidu.com/item/%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B%E5%8D%8F%E8%
56. 相似度：0.15% 标题：《Servlet、Servlet容器等内容讲解 - 池的巧克力 - 博客园》
来源：https://www.cnblogs.com/widget90/p/5640359.html
57. 相似度：0.14% 标题：《Servlet规范简介 - 源哥L 》
来源：https://my.oschina.net/sarsuki2016/blog/1606264
58. 相似度：0.14% 标题：《HTTP协议请求响应过程和HTTPS工作原理 - 邢越 - 博客园》
来源：https://www.cnblogs.com/binyue/p/4500578.html
59. 相似度：0.14% 标题：《Http协议与TCP协议简单理解 - 皎陽 - 博客园》
来源：https://www.cnblogs.com/dingjiaoyang/p/5326544.html
60. 相似度：0.13% 标题：《Tomcat（三）：tomcat处理连接的详细过程 - 骏马金龙 - 博客园》
来源：https://www.cnblogs.com/f-ck-need-u/archive/2018/02/03/8408670.html
61. 相似度：0.13% 标题：《OA系统哪家好!希望真实、客观的评价?》
来源：https://www.zhihu.com/question/19819782
62. 相似度：0.13% 标题：《loadrunner_http长连接设置 - 测试天堂 - 51Testing软件测试网 51...》
来源：http://www.51testing.com/html/48/202848-241574.html
63. 相似度：0.12% 标题：《使用socket连接实现客户端向服务器端的单向通信(socket通信第二弹...》
来源：https://blog.csdn.net/changhe6669/article/details/71800148
64. 相似度：0.12% 标题：《Tomcat剖析(五):Tomcat 容器 - 多啦A - 博客园》
来源：https://www.cnblogs.com/lzb1096101803/p/5134390.html
65. 相似度：0.11% 标题：《第一章:Symfony2和HTTP基本原理 - 为了这有限..._博客园》
来源：https://www.cnblogs.com/suihui/p/4506317.html
66. 相似度：0.11% 标题：《java Socket 短连接和长连接的区别 - CSDN博客》
来源：https://blog.csdn.net/xwq2324/article/details/50962079

67. 相似度：0.11% 标题：《HTTP协议详解 - TopCoderMyDream - 博客园》
来源：https://www.cnblogs.com/wqh17/p/6401525.html
68. 相似度：0.10% 标题：《用ServerSocket 实现服务端与客户端之间信息传递 - CSDN博客》
来源：https://blog.csdn.net/PeicongHe/article/details/54978873
69. 相似度：0.10% 标题：《浏览器输入URL与Web服务器交互过程 - 静夏 - 博客园》
来源：https://www.cnblogs.com/gnss523-webrtc/articles/4834900.html
70. 相似度：0.10% 标题：《自己写一个Web服务器》
来源：http://www.360doc.com/content/16/0504/12/31913486_556154952.shtml
71. 相似度：0.09% 标题：《HTTP请求报文和HTTP响应报文 - jiu~ - 博客园》
来源：https://www.cnblogs.com/jiu0821/p/5641600.html
72. 相似度：0.08% 标题：《Android系列之网络（一）----使用HttpClient发送HTTP请求（通过get方...》
来源：http://www.cnblogs.com/smyhvae/archive/2014/10/03/4004983.html
73. 相似度：0.08% 标题：《tomcat 系统架构与设计模式 第一部分 系统架构工作原理 转-卡卡。冰-...》
来源：http://www.cnblogs.com/kaka-bing/p/3240729.html
74. 相似度：0.08% 标题：《对Django框架架构和Request/Response处理流程的分析 – 疯狂的蚂蚁》
来源：http://www.crazyant.net/1001.html
75. 相似度：0.07% 标题：《简单servlet第三篇---使用servlet读取配置文件的内容 - CSDN博客》
来源：https://blog.csdn.net/liuying1802028915/article/details/78356314
76. 相似度：0.07% 标题：《TCP连接的状态与关闭方式,及其对Server与Client的影响 - CSDN博客》
来源：https://blog.csdn.net/zhaofuguang/article/details/12853915

全文简明报告

基于JAVA的Web服务器设计与开发

摘 要

{ 63% : 随着互联网技术的发展，Web得到了很好的发展， }因为其具有平台无关性，通用性，有相当多的应用系统都从c/s模式转变为了b/s模式，{ 65% : 以及开发者们不满足在互联网上获取静态资源， }{ 67% : 网络的发展方向开始朝着向服务器动态获取资源前进； }{ 56% : 同时为了满足高负载和支撑事务，安全，集成，通信等， }{ 69% : Web服务器为互联网上的应用提供了一个可以实时交互的平台， }{ 57% : 即可以处理静态资源又可以处理动态的Servlet。 }{ 59% : 本文简要的描述了web服务器的背景，现状， }以及发展趋势，并介绍Web服务器所使用到的技术。

关键词：web服务器；Java；http协议；Servlet

JAVA-based Web Server Design and Develop

ABSTRACT

{96% : With the development of Internet technology, } the network has developed very well because of its platform independence and versatility. A considerable number of application systems have changed from C / S mode to B / S mode, and the developers are not satisfied. Obtain static resources on the Internet, the development direction of the network began to move towards the server to obtain resources dynamically; at the same time in order to meet the high load and support services,{ 66% : security, integration, communications, etc. }, the website server provides a real-time interaction for applications on the Internet.{ 57% : The platform can handle static resources and can handle dynamic servlets. } This article briefly describes the background, status quo, and development trends of web servers, and introduces the technologies used by web servers.

KEY WORDS ：Web server; Java; http protocol; Servlet

目 录

摘 要 I
ABSTRACT II
引言（前言） 4

1. 系统背景介绍 5
1.1 Web服务器的发展现状 5
1.2 本文的主要工作 5
2 Web容器的研究 7
2.1 HTTP协议分析 7
2.1.1 HTTP协议概述 7
2.1.2 HTTP协议的发展 7
2.1.3 HTTP协议的传输过程 8
2.1.4 HTTP协议的主要内容 10
2.2 Socket编程 14
2.2.1 Socket概述 14
2.2.2 Java Socket编程 15
2.3 本章小结 16
3 Web服务器的设计 18
3.1 总体模块划分 18
3.2 业务流程 19
4 Web服务器的实现 22
4.1 概述 22
4.2 connector模块 23
4.2.1 connector模块概述 23
4.2.2 监听socket请求的实现 23
4.2.3 解析request请求的实现 25
4.2.4 编码response响应的实现 27
4.3 container模块 28
4.3.1 container模块概述 28
4.3.2 功能链 28
4.3.3 URL映射 29
4.4 parse模块 30
结论 33
参考文献 34
致谢 35

引言（前言）

随着web应用的兴起和移动设备的快速增长，互联网迎来了爆炸式的发展，这也让web系统承载了巨大的压力，但这也推动了web的发展，{ 61% : 使得web技术得到了很好的发展， }如今web已经成为了Internet最主要的信息传送媒介，而web的发展同时也带动了web服务器的发展以适应日益增多的访问量以及各种需求。

Web服务器主要解决的是各种平台的计算机之间的信息交互。它主要采用的是传输层的TCP/IP协议来交换报文，{ 63% : http协议是位于传输层之上的应用层， }http协议的内容就存放在TCP/IP协议中的报文中，本课题要研究的主要问题就是解析HTTP协议，正确加载静态与动态内容。

1. 系统背景介绍

Web服务器就是可以为我们提供Web服务的主机，常用的web服务是http，{ 63% : 我们在浏览器中键入URL地址， }{ 69% : 相对应的web服务器接收到一个HTTP请求，解析请求，并返回一个HTTP响应， }返回

的内容经过JavaScript的处理，以人类可读的方式显示在浏览器页面中。本课题做的就是做一个基于HTTP协议的实现Servlet规范的轻量级Web服务器。

{80%：项目采用Java语言进行开发。 }{89%：Java语言从一开始就支持网络编程， }{87%：并提供了大量类和接口可以用来处理各种网络协议，共享信息，并行计算， }{ 76%：并且Java提供了完全意义的多线程支持， }{ 61%：这些特点使得使用Java成为开发Web服务器的首选计算机语言， }目前基于Java语言开发商用的Web服务器已经有一些了，{ 63%：如WebLogic，Tomcat等。 }

1.1 Web服务器的发展现状

随着互联网的快速发展，Web服务器也随之一起在数量和种类上得到了飞速发展，现如今，{ 57%：Web服务器主要分为WWW服务器， }FTP服务器，邮件服务器，数据库服务器，代理服务器，域名服务器等等.....在本文当中，{ 58%：我们主要介绍的是基于Java的Web容器， }既能实现对静态资源的访问，当然主要是实现servlet规范从而可以实现响应动态内容。虽然Web容器可以单独运行，{ 56%：但是一般都是将其与apache等HTTP服务器一起搭配使用， }因为apache等HTTP服务器加载静态内容的速度更加的快 ，由apache服务器来处理静态资源，而将动态响应交给Web容器来处理。

1.2 本文的主要工作

从前文得知，当前有许多种类型的Web容器，Web服务器，{ 56%：它们适合运行在不同的操作系统， }不同的Web服务器以及不同的Web容器上，如C＃开发的服务器端后台一般都是部署在Windows Server操作系统的IIS Web服务器上，当然现在微软出了.{ 55%：NET Core后，C＃开发的服务器端程序也可以部署在Linux上面了， }而Java开发的服务器端程序主要部署在Linux上的apache服务器与tomcat容器中。

本文主要是设计一个基于Java的Web服务器，实现这个服务器是为了给上层的应用系统提供一个合适便利的开发平台。因为涉及到在此平台上进行上层的应用开发，所以将其称之为上层应用开发。因此，{ 57%：本Web服务器需要解决四个问题： }{ 55%：①将TCP协议的报文解析成HTTP协议的内容， }上层应用不需要再面对解析HTTP协议②上层应用不需要考虑对网络输入输出进行编程，只需要对业务进行编程即好③线程调度问题，上层应用不需要管理多线程，交由此Web服务器来管理多线程④URL映射问题，由web容器来决定此URL交由上层应用的哪个方法运行。

2 Web容器的研究

Web服务器，其实就是能够给人们提供HTTP服务的主机，当用户通过在浏览器上键入该Web服务器的主机地址或者ip地址来访问该Web服务器上的各种所需资源。那么究竟什么是Web呢？{97%：WEB你可以这样简单理解为你所看到的HTML页面就是WEB的数据元素,处理这些数据元素的应用软件就叫WEB服务器,如IIS、apache。 }{83%：WEB服务器与客户端打交道，它主要处理的信息有：session、request、response、HTML、JavaScript、CSS等。 }而Web容器如tomcat，{100%：处理的是非常规性WEB页面（JSP文件），他动态生成WEB页面， }生成的WEB页面再发送给客户端。{87%：Web服务器一般是普遍通用的，而Web容器一般是专用的，如Tomcat只能处理JAVA应用程序而不能处理ASPX或PHP。 }{93%：而Apache是一个Web服务器（HTTP服务器），后来连接Tomcat应用服务器来支持java。 }

2.1 HTTP协议分析

2.1.1 HTTP协议概述

{85%：超文本传输协议（Hypertext Transfer Protocol，简称HTTP）是应用层协议，它是基于TCP/IP通信协议来传递数据的，自 1990 年起，HTTP 就已经被应用于 WWW 全球信息服务系统。 }{100%：HTTP 是一种请求/响应式的协议。 }{92%：一个客户机与服务器成功建立连接后，发送一个请求给服务器；然后等服务器接到请求后，给予相应的响应信息。 }

2.1.2 HTTP协议的发展

{95%：HTTP 的第一版本 HTTP/0.9是一种比较简单的用于网络间原始数据传输的协议, }{97%：功能确实极其简单，不涉及数据包的传输，默认使用80端口，只有一个GET请求方法，且服务器只能响应HTML格式的字符串，服务器响应后即关闭连接。 }

{100%：HTTP/1.0由 RFC 1945 定义 ，在原 HTTP/0.9 的基础上， }{85%：引入了POST和HEAD命令，大大增强了网络交互功能，可以发送任何格式的内容，为互联网的大发展而奠定了基础；同时，除去数据部分，每次通信还要求包含头信息（HTTP header），来描述一些meta数据。 }{97%：新增功能还包括：状态码（status code）、多字符集支持、多部分发送（multi-part type）、权限（authorization）、缓存（cache）、内容编码（content encoding）等。 }{93%：HTTP/1.0版的主要缺点有，由于TCP的每次连接都需要客户端和服务端进行3次握手，但是连接成功后却只能发送一次请求，而后连接就断开了，假如需要多次请求，这样效率就很低。 }{90%：所以，为了解决多次请求效

率低下的问题，有一个非标准的connection字段算是暂时解决了这个问题。 }{98% : Connection: keep-alive。 }{89% : 这样我们可以复用TCP连接，直到客户端或者服务端主动关闭了连接。 }{95% : 不过这不是标志字段，不同的实现可能行为不一致，所以是一种暂时的解决方案。 }

{ 75% : HTTP/1.1进一步完善了HTTP协议，一直到现在还在使用，是最受欢迎的版本。 }{97% : HTTP/1.1新增了许多特性。 }

1 ） 持久连接

{96% : HTTP/1.1默认TCP连接不关闭，能够被多个请求复用，不用声明Connection: keep-alive。 }{93% : 在最后一个请求时，客户端可以主动发送Connection: close，明确要求服务器关闭TCP连接，或者选择不发送，那么客户端和服务器发现对方一段时间没有活动，就会主动关闭连接。 }{94% : 目前，针对同一个域名，大多数浏览器允许同时建立6个持久连接。 }

2 ） 管道机制

{94% : HTTP/1.1引入了管道机制（ pipelining ），即在同一个TCP连接里面，客户端能够同时发送多个请求，这样只不过改进了客户端HTTP协议请求的效率，服务器依然按照请求的先后顺序来响应。 }

3 ） Content-Length

{96% : Content-Length字段显示本次响应的数据长度，如果数据被压缩，则是显示压缩后的长度。 }{98% : 在Connection: keep-alive条件下，Content-Length是必须要有的；反之，和HTTP/1.0一样，Content-Length不是必须的。 }

4 ） 分块传输编码

{87% : 使用Content-Lenght字段有一个前提条件，就是在服务器发送响应之前，必须要知道响应的数据长度。 }{85% : 但是，对于一些耗时的动态操作来说，等到所有操作完成了，服务器才能发送数据，效率是不高的。 }{95% : 因此，HTTP采用了“流模式（ stream ）”，即“分块传输编码”（ chunked transfer encoding ）方式，表明响应的数据长度不确定，这样就可以每产生一块数据，就发送一块数据，从而提高服务器的响应效率。 }

5 ） Transfer-Encoding: chunked

{92% : 这样，只要请求或响应的头部信息里有Transfer-Encoding字段，就表明响应是由数量未定的数据来组成。 }{91% : 每个非空的数据块前面，都会有一个16进制的数值，来表示这个块的长度；最后一个是大小为0的块，表示本次响应数据已发送完了。 }

2.1.3 HTTP协议的传输过程

{ 71% : HTTP协议是位于OSI网络模型中的应用层， }{ 64% : 它是基于TCP/IP协议作为底层传输的。 }客户端的浏览器先是发起socket套接字的请求与服务器建立连接，等TCP三次握手成功后，客户端与服务器就连接建立，客户端将HTTP协议的内容封装在TCP协议中的报文，由TCP协议来发送给HTTP请求消息，这时客户端也从自己的TCP中的报文中接受到HTTP响应。等到客户端或者服务器将HTTP协议的内容发送给TCP之后，HTTP协议就不去管这个内容是怎样去传输的，{ 69% : TCP提供了一个可靠的端到端的数据传输服务， }这就说明了每一个由客户端发出去的HTTP协议的内容都能以相同的顺序到达服务器端，且是无损的到达。这就是OSI网络模型中的解耦合，HTTP协议只需要密切关注本协议是如何运行的，{ 59% : 无须担心数据会丢失，也不必担心TCP如何将数据从错误的顺序，以及错误的数据中恢复出来， }这些已经是位于应用层以下的协议要去处理的任务了。

下面分析具体的HTTP消息传输过程。

图 2 1 HTTP消息传输过程

1 ） 建立TCP连接

TCP连接在计算机中是通过socket来封装的，{ 60% : 建立socket连接要提供一个ip地址和目的主机的端口号， }{ 71% : 直到通过三次握手成功建立连接后， }{ 77% : 就可以开始在客户端和服务器端进行数据传输了。 }

2 ） HTTP请求消息

{ 59% : 客户端将HTTP协议的内容封装交给socket来发送， }其本质是将HTTP协议的内容封装到TCP协议报文中然后发送。

3 ） HTTP响应消息

{ 56% : 服务器端从socket中接受到HTTP协议的内容，经过解析后， }按照请求内容发送相对应的响应消息。

4 ） 释放TCP连接

{ 57% : 客户端和服务器端双方都可以通过关闭socket来释放TCP连接，在HTTP/1.0中，当服务器端发送完客户端请求的文件后会主动的关闭socket连接， }{ 67% : 而在HTTP/1.1中，一般是保持长连接的， }并不关闭socket连接。

2.1.4 HTTP协议的主要内容

HTTP协议的主要内容是关于HTTP请求和响应的文本，请求的文本是客户端要求服务器端提供的服务的信息，{ 71% : 响应的文本则是服务器端接收到请求后返回给客户端的消息。 }

1 ） 客户端请求格式

a ） 请求的消息格式

{98% : 客户端发送一个HTTP请求到服务器的请求消息包括以下格式：请求行（request line）请求头部（header）、空行和请求数据四个部分组成。 }

图 2 2 请求消息格式

{ 57% : Get请求例子，使用Postman code模式看到的request请求的文本： }

表 2 1 Get请求例子

GET http://download.microtool.de:80/somedata.exe

Host: download.microtool.de

Accept:*/*

Pragma: no-cache

Cache-Control: no-cache

Referer: http://download.microtool.de/

User-Agent:Mozilla/4.04[en]（Win95;I;Nav）

Range:bytes=554554-

Postman-Token: 26d138d4-6123-479e-0bb9-15d37a936dcd

b ） 请求方法

方法描述了对于资源所要做的操作。{ 68% : 在HTTP/1.1中的请求的方法有:OPTIONS,GET,HEAD,PUT,POST,TRACE,DELETE， }{ 61% : 其中最常用的是GET,POST,PUT,DELETE，在RESTFUL规范中， }RESTFUL是以资源为导向设计API的，{ 63% : GET表示获取资源，POST表示添加资源，PUT表示修改资源，DELETE表示删除资源。 }

图 2 3 HTTP请求方法

c ） 头部字段

{ 63% : 头部字段分为通用请求头，请求头和响应头， }{ 75% : 通用请求头就是既能用于请求，也能用于响应，是作为一个整体而不是与事务相关联的特定资源， }并且请求头和响应头就是只有在请求和响应时可以带上。{90% : 请求头允许客户端传递关于自身的信息和所希望的响应形式， }{83% : 响应头用来表示服务器和于传递自身信息的响应。 }

常用的请求头如下表所示：

表 2 2 常用的请求头

Header 解释 示例

Accept 指定客户端能够接收的内容类型 Accept: text/plain, text/html

{ 76% : Accept-Charset 浏览器可以接受的字符编码集。 }{92% : Accept-Charset: iso-8859-5 }

{81% : Accept-Encoding 指定浏览器可以支持的web服务器返回内容压缩编码类型。 }{98% : Accept-Encoding: compress, gzip }

Accept-Language 浏览器可接受的语言 Accept-Language: en,zh

Cookie HTTP请求发送时，会把保存在该请求域名下的所有cookie值一起发送给web服务器。 Cookie: $Version=1; Skin=new;

Content-Type 请求的与实体对应的MIME信息 Content-Type: application/x-www-form-urlencoded

Host 指定请求的服务器的域名和端口号 Host: www.zcmhi.com

1 ） 服务器端响应

a ） 响应的消息格式

图 2 4 响应消息格式

服务器端响应例子：

表 2 3 服务器端响应例子

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache01

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

ETag: “34aa387-d-1568eb00 “

Accept-Ranges: bytes

Content-Length: 51

Vary: Accept-Encoding

Content-Type: text/plain

{100% : Hello World! My payload includes a trailing CRLF. }

b ） 响应头

常用的响应头如下表所示：

表 2 4 常用的响应头

Header 解释 示例

{100% : Content-Encoding web服务器支持的返回内容压缩编码类型。 }{100% : Content-Encoding: gzip }

Content-Language 响应体的语言 Content-Language: en,zh

Content-Length 响应体的长度 Content-Length: 348

{91% : Content-Location 请求资源可替代的备用的另一地址 Content-Location: /index.htm }

Content-Type 返回内容的MIME类型 Content-Type: text/html; charset=utf-8

Last-Modified 请求资源的最后修改时间 Last-Modified: Tue, 15 Nov 2010 12:45:26 GMT

Server web服务器软件名称 Server: Apache/1.3.27 （ Unix ） （ Red-Hat/Linux ）

Set-Cookie 设置Http Cookie Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1

2.2 Socket编程

2.2.1 Socket概述

{ 69% : 要了解socket，首先需要知道socket是对TCP/IP的封装， }那么什么是TCP哪。{100% : TCP是Tranfer Control Protocol的 简称，是一种面向连接的保证可靠传输的协议。 }{100% : 通过TCP协议传输，得到的是一个顺序的无差错的数据流。 }{100% : 发送方和接收方的成对的两个socket之间必须建立连接，以便在TCP协议的基础上进行通信，当一个socket（通常都是server socket）等待建立连接时，另一个socket可以要求进行连接，一旦这两个socket连接起来，它们就可以进行双向数据传输，双方都可以进行发送 或接收操作。 }

2.2.2 Java Socket编程

Socket编程分为服务器端socket编程和客户端socket编程。{ 68% : 下面分别介绍两种socket编程的步骤。

}

1 ） 服务器端socket编程

a ） 服务器端的socket首先需要绑定在本地的某个IP和端口上。在Java中默认为0.0.0.0，{ 69% : 在服务器中0.0.0.0表示本机上所有的IPv4地址，因为服务器可能会有多个IP地址， }比如绑定在127.0.0.1就表示这个Socket只能接受局域网内的socket连接。具体代码如下,表示绑定在本机IP为0.0.0.0的80端口上，80也是HTTP的默认端口号。

表 2 5 服务器端绑定端口

ServerSocket serverSocket=new ServerSocket (80)

{ 63% : b ） 将socket设置到监听模式等待客户端连接。 }这个accept ( ) 方法是阻塞的，当有连接建立时，将返回一个socket用来和客户端进行交互。代码如下：

表 2 6 等待连接

socket=serverSocket.accept ( )

c ） 使用连接建立后返回的socket与客户端进行通信。{ 56% : TCP是可以进行全双工通信的， }所以通过socket变量，{ 56% : 我们可以拿到输入流和输出流， }输入流就是客户端发送给服务器端的信息，{ 65% : 输出流就是服务器端想要返回给客户端的信息。 }代码如下:

表 2 7 socket读写

InputStream is=socket.getInputStream ( ) ;

BufferedInputstream bis=new BufferedInputstream (is) ;

String content=bis.readLine ( ) ;

OutputStream os=socket.getOutpuStream ( ) ;

BufferedOutputStream bos=new BufferedOutputStream (os) ;

String outputContent=” hello world” ;

bos.write (bos) ;

d ） 关闭连接。{ 64% : 当客户端和服务器端的通信结束之后， }需要释放连接。因为我们将socket的输入流和输出流用缓冲流包装了，{ 57% : 所以关闭包装后的输入流和输出流也就是关闭socket自身的输入流和输出流， }{ 76% : 最后再关闭socket自身。 }代码如下:

表 2 8 关闭socket

bis.close ( ) ;

bos.close ( ) ;

socket.close ( )

2 ） 客户端socket编程

{ 65% : a ） 客户端首先需要新建Socket， }{ 72% : 并输入需要连接的服务器的IP地址和需要访问的端口号， }下图表示与IP为1.1.1.1的主机在80端口号上建立连接。

表 2 9 新建socket连接

Socket socket = new Socket (‘1.1.1.1’ ,80)

b ） 使用连接建立后的socket与服务器端进行通信。{ 63% : Java的socket是自动进行连接的， }不需要函数调用。

表 2 10 socket读写

InputStream is=socket.getInputStream ( ) ;

BufferedInputstream bis=new BufferedInputstream (is) ;

String content=bis.readLine ( ) ;

OutputStream os=socket.getOutpuStream ( ) ;

BufferedOutputStream bos=new BufferedOutputStream (os) ;

String outputContent=” hello world”；

bos.write（bos）；

c）关闭连接。{ 64%：当客户端和服务器端的通信结束之后， }需要释放连接。因为我们将socket的输入流和输出流用缓冲流包装了，{ 57%：所以关闭包装后的输入流和输出流也就是关闭socket自身的输入流和输出流， }{ 76%：最后再关闭socket自身。 }代码如下:

表 2 11 关闭socket

bis.close（ ）;

bos.close（ ）;

socket.close（ ）

2.3 本章小结

Web使用的主要是HTTP协议，在这一章中对HTTP进行了详尽的介绍，{ 59%：包括HTTP的请求格式，响应格式， }请求方法，请求头以及传输过程等。

本章之后介绍了Java下的Socket编程，服务器socket编程和客户端socket编程并没有太多区别，唯一的区别是服务器端最初需要一个ServerSocket来监听在绑定的IP和端口下有没有新新连接，当获得返回的socket后与客户端的编程就并无区别了。

3 Web服务器的设计

3.1 总体模块划分

{ 60%：将整个服务器分为几个模块来分别编写，各个模块负责不同的功能， }模块通过接口来调用其他模块，{ 56%：实现了解耦合。总体模块结构如图所示。 }

图 3 1 总体模块划分

{ 77%：下面将分别简单的介绍各个模块的功能。 }

1）Socket连接调度模块：Socket连接调度模块是提供对Socket的接受连接以及将socket封装在一个任务中将其交给线程池来处理。

2）HTTP解析模块：{ 59%：HTTP解析模块中可以分为两个子模块， }分别是对HTTP请求解码模块以及对HTTP响应编码模块，如下图所示。

图 3 2 HTTP解析模块划分

3）请求响应模块：请求响应模块主要处理的是socket的输入输出，以及数据的缓存.

4）Servlet调度模块：当获得解析的URL时候，由Servlet调度模块来决定应该使用哪个Servlet进行响应.

5）Filter调度模块：当获得解析的URL时候，由Filter调度模块来决定使用哪些个Filter来进行过滤

6）配置解析加载模块：Web服务器是采用外置xml文件来设置参数，如项目需要绑定的端口，{ 61%：Servlet需要映射的URL， }打印日志的级别，项目的名称等。

3.2 业务流程

下面我将详细的介绍Web服务器的运行流程。

1）当Web服务器启动的时候首先加载xml配置文件进行配置

2）创建服务器套接字，并绑定端口，然后监听这个端口

{ 64%：3）Connector连接器负责接受接受客户端连接， }初始化一个任务交给线程池，任务中包含解析HTTP请求，并构造请求和响应对象以供后续对象调用

4）将请求和响应对象交给container来调用，{ 65%：container检索请求对象的URL是否是servlet，若是， }则加载servlet到内存中，{ 58%：执行该servlet相关方法， }若不是，则加载相对应静态文件，若没有相对应的静态文件，则加载一个错误页面内容到响应对象中

5）将响应对象中的内容写入socket中

6）清除HTTP请求和响应对象

7）断开socket连接，一次请求结束

{ 67%：这就是Web服务器运行的详细流程， }{ 55%：同时我也画了具体的流程图如下所示。 }

图 3 3 Web服务器流程

4 Web服务器的实现

4.1 概述

{ 60% : 本Web服务器是面向接口编程的， }这样当有一个类B需要用到类A的时候，就不用需要A的实现类，只需要持有A的接口即可，不必关心实现类具体是哪个，达到了类之间的解耦合。为这个Web服务器，我设计了如下16个接口。

下面我将分别介绍这16个接口的用途。

1 ） Chain,Valve,ValveSupport接口：这三个接口放在一起讲，是因为它们用来配合完成同一件事情。设计这三个接口中使用了责任链设计模式。Valve接口是用来定义一个在每次调用servlet方法前的方法，Chain接口是功能链，通过Chain将所有的Valve串联在一起，ValveSupport是一个支撑类，用来给Chain提供帮助。

{81% : 2 ） Connector接口： }connector接口主要完成的接受socket请求，然后转交给container进行处理。

{ 76% : 3 ） Contained接口： }contained接口用来指示实现它的接口持有哪个container容器的变量。

4 ） Container接口：container接口的实现类有ContextContainer和ServletContainer分别表示Web容器上下文以及单个servlet的容器。

5 ） ContextContainer接口：{ 57% : ContextContainer接口继承了Container接口， }表示的是此Web容器的上下文。

{ 60% : 6 ） HttpRequest，HttpResponse接口：这两个接口分别是HTTP请求和响应的接口， }同时也继承了servlet规范的request,response接口

7 ） LifeCycleListener,LifeCycleSubject接口：这两个接口的设计使用了观察者模式，主要作用是实现模块间的关联启动。

8 ） Loader接口：Loader接口主要是实现安全加载servlet到容器中。

9 ） Mapper接口：Mapper接口的主要功能是实现URL到servletClass的映射。

10 ） Request，Response接口：这两个接口继承了servlet规范中的request,response接口，是这个Web容器中请求和响应的基类接口。

4.2 connector模块

4.2.1 connector模块概述

{ 70% : 此模块中的HttpRequestImpl, }{ 70% : HttpResponseImpl都是请求和响应接口的具体实现， }RequestStream,ResponseStream都是对socket的输入和输出流的缓冲流，同时加上一些功能。下面看这个模块下的所有的类的组成。

图 4 1 connector模块中的类

4.2.2 监听socket请求的实现

JerryConnector这个类是这个包的主类，由它来负责调用其他类。下面请看这个类的主方法，如下表所示

表 4 1 监听socket请求的实现

```
public void run ( )
{
while ( start )
{
Socket socket=null;
try {
socket=serverSocket.accept ( ) ;
if ( timeOut > 0 )
socket.setSoTimeout ( timeOut ) ;
```

```
socket.setTcpNoDelay ( true ) ;
HttpProcess process=new HttpProcess ( socket,container ) ;
executorService.submit ( process ) ;
}
catch （IOException e ）{
e.printStackTrace （ ）;
}
finally {
if ( socket!=null )
{
try {
socket.close （ ）;
}
catch （IOException e ）{
e.printStackTrace （ ）;
}
}
}
}
if ( serverSocket!=null )
{
try {
serverSocket.close （ ）;
}
catch （IOException e ）{
e.printStackTrace （ ）;
}
}
}
```

这个方法主要完成的是等待客户端socket的连接，当有连接之后，为这个连接设置一个超时自动关闭连接的时间，将这个socket包装进一个HttpProcess类，然后交给线程池来处理这个HttpProcess类，这样，这个服务器就可以以非阻塞的方式继续运行下去，当同时有多个连接进来时，也不会阻塞。同时，线程池也是一种高效的方式来利用线程，降低了每次创建新线程初始化线程上下文的时间，使得可以同时有更多的连接进来。

4.2.3 解析request请求的实现

解析HTTP协议的内容放在HttpProcess中进行的。我们知道，{ 71%：HTTP的请求中，第一行是请求行，}第二行开始一直到空白行之前都是请求头，下面总览一下解析request请求代码。

表 4 2 解析request请求的总体设计

```
void parseRequest （ ） throws IOException
{
parseRequestLine （ ）;
parseHeader （ ）;
```

```
setHeaderAttribute ( ) ;
}
```

首先先是解析请求行，然后是解析请求头，最后是将请求行，请求头里解析出来的相关数据赋值到request请求中，以便后续使用，这里注意到我抛出了一个异常，假如请求解析失败则返回一个错误，记录到日志中，解析失败有可能是因为网络问题以及有人发送了错误的HTTP协议的文本请求到服务器。接下来，我们详细看下解析请求行和请求头的代码。

我们来看解析请求行的代码

表 4 3 解析请求行

```
void parseRequestLine ( ) throws IOException
{
String str=readLine ( ) ;
Pattern pattern=patternFactory.getRequestLinePattern ( ) ;
Matcher matcher= pattern.matcher ( str ) ;
int condition=0;
while ( matcher.find ( ) )
{
String arg= matcher.group ( ) ;
switch ( condition )
{
case 0:
httpRequest.setMethod ( arg.toUpperCase ( ) ) ;
break;
case 1:
if ( arg.contains ( "? " ) )
{
String queryString=arg.substring ( arg.indexOf ( '? ' ) +1 ) ;
queryString=
URLDecoder.decode ( queryString,httpRequest.getCharacterEncoding ( ) ) ;
arg=arg.substring ( 0,arg.indexOf ( '? ' ) ) ;
httpRequest.setQueryString ( queryString ) ;
Pattern parametersPattern=patternFactory.getParameterPattern ( ) ;
Matcher parametersMatch= parametersPattern.matcher ( queryString ) ;
while ( parametersMatch.find ( ) )
{
String key=parametersMatch.group ( 1 ) ;
String value=parametersMatch.group ( 2 ) ;
httpRequest.setParameter ( key,value ) ;
}
}
httpRequest.setRequestURI ( arg ) ;
break;
```

```
case 2:
httpRequest.setProtocol ( arg ) ;
break;
}
condition++;
}
}
```

在解析请求行中，因为请求行是一种规范的文本，所以我使用了正则表达式来帮助解析，之后是解析请求头

表 4 4 解析请求头

```
void parseHeader ( ) throws IOException
{
String str= " ";
while ( ( str=readLine ( ) ) != " " )
{
//blank line
if ( str== "-1 " )
break;
Pattern headerPattern=patternFactory.getHeaderPattern ( ) ;
Matcher headerMatcher=headerPattern.matcher ( str ) ;
String header,value;
if ( headerMatcher.find ( ) )
{
header=headerMatcher.group ( 1 ) ;
value=headerMatcher.group ( 2 ) ;
httpRequest.setHeader ( header,value ) ;
if ( header.equals ( "Cookie " ) )
{
Pattern cookiePattern=patternFactory.getCookiePattern ( ) ;
Matcher cookieMatcher=cookiePattern.matcher ( value ) ;
String cookieKey,cookieValue;
while ( cookieMatcher.find ( ) )
{
cookieKey=cookieMatcher.group ( "name " ) ;
cookieValue=cookieMatcher.group ( "value " ) ;
httpRequest.setCookie ( cookieKey,cookieValue ) ;
}
}
}
}
}
```

4.2.4 编码response响应的实现

在用户编写的servlet代码里，是不需要关心返回的响应中的那些HTTP状态码，响应头这些的，{ 59%：这些都是由Web服务器来完成的， }用户编写的servlet响应的文本是写入到Web服务器的缓存中的，{ 64%：也即ResponseStream中， }因为servlet获取到的outputStream早已不是socket的outputStream，{ 56%：而是被我们替换成ResponseStream了， }当servlet写入完成调用outputStrean的close方法啊，{ 71%：或者第一次调用outputStream的flush方法的时候， }都会触发response的编码，{ 60%：具体的代码是在ResponseStream里面， }如下所示

表 4 5 编码response响应

```
private void writeHeaders ( )
{
HttpResponse httpResponse= （HttpResponse） response;
StringBuilder sb=new StringBuilder ( “ “ ) ;
int s=httpResponse.getStatus ( ) ;
HttpStatus status=HttpStatus.getHttpStatus ( s ) ;
sb.append ( “HTTP/1.1 “+status.getStatus ( ) + “ “+status.getName ( ) + “\r\n “ ) ;
for （String headerName : httpResponse.getHeaderNames ( ) ) {
String value=httpResponse.getHeader ( headerName ) ;
sb.append ( headerName+ “: “+value+ “\r\n “ ) ;
}
sb.append ( “\r\n “ ) ;
try {
socketOutputStream.write ( sb.toString ( ) .getBytes ( response.getCharacterEncoding ( ) ) ) ;
}
catch （IOException e ) {
e.printStackTrace ( ) ;
}
}
```

4.3 container模块

4.3.1 container模块概述

如图是container模块中的具体代码。

图 4 2 container模块中的类

container模块是Web容器中相当重要的一个模块，{ 61%：connector模块接受socket后， }就是直接转交给container模块进行处理的。container模块主要包含四个类，其中ContextContainerImpl是Web容器的上下文，ServletContainerImpl是单个servlet的容器，通过调用它就能对servlet进行操作。ChainImpl是在执行servlet方法前调用的一系列可横向扩展的类，MapperImpl类主要功能是根据配置文件中的URL映射，{ 68%：依据当前请求的URL来找到进行响应的servlet。 }

4.3.2 功能链

这是我对ChainImpl的中文名称的翻译。{ 62%：这个类的执行顺序是在执行servlet前， }对于这个类的设计采用了责任链的设计模式，使得横向添加Valve类不需要改变代码。

表 4 6 Valve集合

```
private Valve basicValve;
private ArrayList < Valve > valves=new ArrayList < > ( ) ;
```

这个类中有一个Valve的集合，Valve接口只有一个invoke方法，如下所示

表 4 7 Valve接口方法

void invoke (HttpRequest httpRequest,HttpResponse httpResponse,ValveSupport valveSupport) ;

通过这个接口我们就可以对请求和响应接口进行一些修改，或者是在响应前打日志，当然最主要的还是Servlet规范中的Filter接口的执行，Servlet规范中Filter是在Servlet的方法执行前调用的，有了ChainImpl这个类后我们就可以将Filter接口也放在这里执行，同时使用这个Web服务器的人也可以通过自定义Valve类来丰富Web服务器的功能。

4.3.3 URL映射

URL映射的过程是这样子的，在配置文件中的写法参考了tomcat的xml配置文件的样式。

表 4 8 XML配置文件范例

```
< servlet >
< servlet-name > FirstServlet < /servlet-name >
< servlet-class > Two < /servlet-class >
< /servlet >
< servlet-mapping >
< servlet-name > FirstServlet < /servlet-name >
< url-pattern > /* < /url-pattern >
< /servlet-mapping >
```

{ 56% : Servlet-class中的参数是实际servlet文件的名称， }我们在Web容器中为它指定另外一个名称，{ 63% : 这个名称就是servlet-name标签中的名称。 }{ 56% : servlet-mapping标签中包含了两个子标签，其中servlet-name就是容器中指代某个servlet的名称，url-pattern标签的内容表示某个URL. }{ 60% : 那么当我们的服务器在接受到一个请求后， }它查找servlet的流程是这样的

1 ） 根据URL查找对应的servlet-name

2 ） 若找到相对应的servlet-name则找到相对应的servlet-class

我们从xml解析出来的映射关系存储在ContextContainerImpl中

表 4 9 contextContainerImpl中的映射关系

private HashMap < String, String > patternMap = new HashMap < > ( ) ;

具体查找映射关系是通过Mapper接口的实现类MapperImpl，下面看下具体代码:

表 4 10 查找映射请求关系

```
public Container map ( HttpRequest request )
{
if ( request.getServletContainer ( ) != null ) return request.getServletContainer ( ) ;
String contextPath = request.getContextPath ( ) ;
String relativeURI = request.getRequestURI ( ) .substring ( contextPath.length ( ) +1 ) ;
ServletContainer servletContainer = null;
String name = null;
if ( servletContainer == null ) {
name = container.findServletContainerMapping ( relativeURI ) ;
if ( name != null ) {
servletContainer = ( ServletContainer ) container.getChild ( name ) ;
}
}
if ( servletContainer == null ) {
```

```
while （true）{
name = container.findServletContainerMapping（relativeURI）;
if （name != null）{
servletContainer = （ServletContainer） container.getChild（name）;
}
if （servletContainer != null）{
break;
}
int index = relativeURI.lastIndexOf（ ‘/ ‘）;
if （index < 0） break;
relativeURI = relativeURI.substring（0, index）;
}
}
if （servletContainer == null）{
relativeURI = request.getRequestURI（ ）.substring（contextPath.length（ ）+1）;
List < String > urlList=URLUtil.splitUrl（relativeURI）;
urlList.add（ “/ “）;
for（int i=0;i < urlList.size（ ）;i++）
{
name = container.findServletContainerMapping（urlList.get（i））;
if （name != null）{
servletContainer = （ServletContainer） container.getChild（name）;
}
if （servletContainer != null）{
break;
}
}
}
return servletContainer;
}
```

4.4 parse模块

Parse模块的主要工作是解析xml配置文件，并赋值给相关类，配置文件的格式如下

表 4 11 xml配置文件范例

```
< ?xml version= “1.0 “ encoding= “UTF-8 “? >
< web-app >
< context name= “jerry “ port= “80 “ debug= “info “ >
< webroot > /home/jmt/IdeaProjects/Jerry/webroot/ < /webroot >
< servlet >
< servlet-name > FirstServlet < /servlet-name >
< servlet-class > Two < /servlet-class >
```

```
</servlet>
<servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<url-pattern>/*</url-pattern>
</servlet-mapping>
</context>
</web-app>
```

具体的主要解析代码如下

表 4 12 具体解析代码

```
private static ContextBean parseContext ( Node contextNode )
{
ContextBean contextBean = new ContextBean ( ) ;
NamedNodeMap attributeMap = contextNode.getAttributes ( ) ;
Node name = attributeMap.getNamedItem ( "name" ) ;
contextBean.setName ( name.getNodeValue ( ) ) ;
Node port = attributeMap.getNamedItem ( "port" ) ;
contextBean.setPort ( Integer.parseInt ( port.getNodeValue ( ) ) ) ;
Node debug = attributeMap.getNamedItem ( "debug" ) ;
contextBean.setDebug ( debug.getNodeValue ( ) ) ;
NodeList childList = contextNode.getChildNodes ( ) ;
for ( int i = 0; i < childList.getLength ( ) ; i++ ) {
Node node = childList.item ( i ) ;
if ( node.getNodeName ( ) == "webroot" ) {
contextBean.setWebroot ( node.getTextContent ( ) ) ;
}
else if ( node.getNodeName ( ) == "servlet" ) {
NodeList servletNodeList = node.getChildNodes ( ) ;
String servletName = servletNodeList.item ( 0 ) .getTextContent ( ) ;
String servletClass = servletNodeList.item ( 1 ) .getTextContent ( ) ;
contextBean.setServletNameClass ( servletName, servletClass ) ;
}
else if ( node.getNodeName ( ) == "servlet-mapping" ) {
NodeList servletNodeList = node.getChildNodes ( ) ;
String servletName = servletNodeList.item ( 0 ) .getTextContent ( ) ;
String servletUrl = servletNodeList.item ( 1 ) .getTextContent ( ) ;
contextBean.setServletNameUrl ( servletName, servletUrl ) ;
}
}
return contextBean;
}
```

结论

{ 61% : 使用者可以将servlet代码部署在这个Web服务器上， }服务器可以对静态资源和动态资源进行很好的响应。这个系统最突出的地方就是使用了大量的设计模式使得系统有良好的解耦性和可扩展性，{ 71% : connector和container是解耦合的， }我们可以通过更换connector或者container来实现一个更符合业务的Web服务器。也可以自定义Valve来增强这个Web服务器的功能。不足的地方在于这个Web服务器的连接是同步非阻塞的，势必会遇到C10k问题，希望以后可以改成异步非阻塞，{ 59% : 采用Java NIO来重写connector。 }

参考文献

[1]马毅. 轻量级Web服务器的实现与应用[D]. 西北大学, 2008

[2]宋立昊. 基于线程池的WEB服务器实现和监测[D]. 吉林大学, 2011

[3]刘穿时. 轻量级web服务器的设计与实现[D]. 华中科技大学, 2016

[4]柴快长. 轻量级web容器的设计[J]. 科学与财富, 2013（1）:132-132

[5]Budi Kurniawan, Paul Deck, 曹旭东. 深入剖析Tomcat[M]. 机械工业出版社, 2012

[6]陈涛, 任海兰. 基于Linux的多线程池并发Web服务器设计[J]. 电子设计工程, 2015（11）:167-169

[7]兰红, 柳显涛, 李文琼. 基于Linux的预线程化并发Web服务器设计[J]. 江西理工大学学报, 2012, 33（1）:63-67

[8]上野宣. 图解 HTTP. 于均良译. 北京: 人民邮电出版社, 2014: 2-3

[9]沙洛韦. 设计模式精解[M]. 清华大学出版社, 2004

[10]VenkatSubramaniam, 苏帕拉马尼亚姆, 薛笛. Java虚拟机并发编程[M]. 机械工业出版社, 2013

[11]盖茨, 童云兰. Java并发编程实战[M]. 机械工业出版社, 2012

[12]杨小娇. 轻量级高并发Web服务器的研究与实现[D]. 南京邮电大学, 2014

[13]秦浏杰. 高并发Web系统的异步化研究[D]. 华北电力大学, 2015

[14]web服务器分类及发展趋势的研究[A].2010

“http://www.itxx.com.cn/Server/webserver/Server_44.html”

[15]Bielecki M, Hidders J, Paredaens J, et al. The Navigational Power of Web Browsers. Theory of Computing Systems, 2012, 50（2）: 213-240

[16]Snehi J, Dhir R. Web Client and Web Server approaches to Prevent XSS Attacks[J]. International Journal of Computers & Technology, 2013, 4（2）:345-352

致谢

这次毕业论文的顺利完成，要感谢林春梅老师对我的精心指导。这使我能较好的完成这次的项目。如果没有老师的指导，我可能已经完全不知道如何去完成。感谢林春梅老师给我这么一个学习的机会。