北京邮电大学

Java网络编程结题报告

**项目名称：支持Markdown的静态 Web服务器**

成员: 戴鑫旺 程相睿 杨乐乐

2022年6月13日

|  |  |  |  |
| --- | --- | --- | --- |
| 论文题目 | 支持Markdown的静态WEB服务器 | | |
| 开题日期 | 2022-5-1 | 是否开题题目 | 是 |
| **一、立题依据（包括研究目的、意义、国内外研究现状和发展趋势，需结合科学研究发展趋势来论述科学意义；或结合国民经济和社会发展中迫切需要解决的关键科技问题来论述其应用前景。附主要参考文献目录）**  1.1研究背景与意义  自从Markdown诞生以来，越来越多的写作者被这种轻量简洁却又样式丰富的写作方式所吸引。各类博客网站与讨论社区也都提供了基于Markdown的博文编辑器。此外，现有的绝大多数商业笔记软件，如印象笔记等，也将对Markdown的支持作为自己的产品特色来吸引更多的用户。  本团队受此启发，计划开发一个具备Markdown解析功能的静态WEB服务器，使得用户只需编写简单的Markdown文件即可生成网页进行博文记录或信息发布。在此过程中，我们可以加强对Markdown语法的理解和对语法解析流程的认识，了解网站搭建的原理，以及加深对Java服务器端网络编程特别是HTTP协议的理解。  1.2国内外研究现状  1.2.1研究历史  在PC机诞生初期，软件开始主要运行在桌面上，而数据库这样的软件运行在服务器端，这种Client/Server模式简称CS架构。随着互联网的兴起，人们发现CS架构不适合大多数网络应用程序，最大的原因是网络应用程序的修改和升级非常迅速，而CS架构需要每个客户端逐个升级桌面App，因此，基于Web服务器的Browser/Server模式开始流行，简称BS架构。在BS架构下，客户端只需要浏览器，应用程序的逻辑和数据都存储在服务器端。浏览器只需要请求Web服务器，获取Web页面，并把Web页面展示给用户即可。常用的Web服务器包括Nginx、Apache以及IIS等。  18年前，美国科技博主John Gruber希望“用一种易读易写的纯文本格式写作，并可将其转换为有效的 XHTML (或 HTML)”，于是在 2004 年 3 月 19 日推出初代 Markdown。随后，在Aaron Swartz的帮助下，Markdown的语法不断完善，并逐渐走进大众的视野。随着Web2.0时代的到来，Markdown凭借其在写作和预览上的便利性，受到越来越多程序开发者和博文写作者的欢迎。近年来针对 Markdown 开发的编辑器越来越多，支持 Markdown 也成了 GitHub、简书、CSDN等平台的标配。  1.2.2 应用现状  Github与2008推出了名为GitHub Pages的网页寄存服务，可以用于存放静态网页来作为免费的个人静态Web服务器。Jekyll与Hugo等软件可以用于将文档转换成静态网页，而且该软件提供了将网页上传到GitHub Pages的功能。因此将Jekyll或Hugo与GitHub Pages结合起来就可以得到支持Markdown的静态Web服务器。  **二、研究内容和目标（说明课题的具体研究内容，研究目标和效果，以及拟解决的关键科学问题。此部分为重点阐述内容）**  本课题计划开发一个具备Markdown解析功能的静态WEB服务器，使得用户只需编写简单的Markdown文件即可生成网页进行博文记录或信息发布，同时还需支持图片的上传与展示。  **2.1 研究内容**  具体而言，本课题主要研究内容与研究目标分为两部分：   1. Markdown文件处理与HTML文件生成：    1. 识别Markdown文件与HTML文件，对于HTML文件不做处理；    2. 遍历Markdown文件，并将其解析渲染为原始HTML文件；    3. 提供CSS样式文件与JavaScript脚本文件对原始HTML进行美化；    4. 遍历相应目录中的图片进行处理，生成静态的图片展示HTML页面。 2. 静态服务器：    1. 接受用户请求，并对用户请求按照HTTP请求格式进行解析，得到请求参数；    2. 根据用户的请求参数对HTML文件进行检索；    3. 将检索得到的HTML文件封装成HTTP响应返回给用户；    4. 处理用户的文件上传操作，解析上传请求的报文格式，得到上传文件；    5. 将用户上传的Markdown文件和图片文件进行格式识别并保存到服务器端相应目录；    6. 提供Markdown实时解析能力，将用户上传的Markdown文件交由Markdown文件处理模块实时解析成HTML进行展示；    7. 提供图片实时解析能力，将用户新上传的图片进行处理并渲染出新的图片页进行展示。   **2.2 开发目标与预期效果**  预期的项目运行流程如下：  （1）首先服务器启动，配置解析模块读取本地配置文件，然后通过配置初始化Markdown文件处理模块和用户请求处理模块。  （2）接着，Markdown文件处理模块根据配置读取本地特定目录下的Markdown源文件，并进行解析。在解析的过程中，首先调用Yaml解析模块处理Markdown文件头部的yaml部分，这部分包含博文的元数据，例如该博文的题目、作者、发布时间以及展示优先级。然后调用Markdown解析模块将Markdown转化为HTML格式，附上CSS样式形成HTML文件，并缓存到特定目录。此时，博文页面生成完毕。然后该模块还会根据博文的元数据创建一个index.html页面，该页面是所有博文的目录，也是静态服务器的默认首页。同时该模块还会解析图片目录，并创建一个photos.html页面对用户发布的图片进行展示。  （3）随后用户请求处理模块启动，根据配置创建ServerSocket并监听特定端口。用户通过浏览器发起访问，用户请求处理模块接收用户请求并进行参数解析，得到用户的请求路径。如果请求的地址指向的博文页面存在，则对博文进行展示。否则，返回页面不存在的错误信息。  （4）用户请求处理模块还能处理用户的文件上传请求，解析用户上传的文件格式，并根据格式选择保存在服务器的博文目录还是图片目录。对于用户上传的Markdown文件进行处理并展示，对于用户上传的图片文件进行解析，并更新图片展示页。此外，上传页面要提供文件预览功能，如果用户上传了格式错误的文件则进行错误信息提示。  **2.3 拟解决的关键科学问题**  因为我们的项目没有用开源的Web服务以及Java的Spring或是Servlet等开发框架，所以需要手动解析二进制的HTTP请求，并构造相应的HTTP响应。如果高效对HTTP请求特别是multipart/form-data格式的文件上传请求进行高效解析是我们的需要解决的重要研究问题。我们需要通过网络抓包、查找资料等方式查明各类请求的二进制格式，并进行相应解析。对于文件上传请求会涉及到图片的二进制数据与请求头的文本数据共存的情况，需要额外注意。  **三、研究方案设计与可行性分析（包括：研究方法，技术路线，理论分析、计算、实验方法和步骤及其可行性等）**  3.1 研究方法  学习Markdown的语法与语法解析原理，查阅相关资料，调研现有开源Markdown语法解析器的功能情况。学习计算机网络中网络层与应用层的相关知识，学习并理解HTTP请求响应格式，通过网络抓包的方式深入理解HTTP请求格式。学习Java语言提供的InputStream等IO相关的类以及字节数组、字符串类提供的方法，对HTTP请求进行解析处理。学习总结并应用多线程与非阻塞IO等Java服务器端编程技术。  3.2 技术路线  对于第一个研究内容，我们需要学习Markdown语法解析相关的知识。学习现有Markdown语法解析器的工作流程。我们发现现有的解析库都是首先通过Java的IO库进行本地文件的读取，随后通过状态机的方式进行语法分析，提取Markdown的文本信息与样式信息，最后根据样式信息对原始文本增加特定的HTML标签。同时需要注意的是Markdown文本中可能会使用HTML标签来表达一些Markdown自身语法所不能表达的样式，这些标签需要保留。  对于第二个研究内容，为了深入学习与应用Java网络编程相关的知识，我们计划不采用现成的Web服务器，而是自己从头开始编写一个Web服务器。首先通过ServerSocket建立服务器端端口监听，然后对于每一个用户请求都生成新的线程进行处理，这些新的线程都会通过线程池进行管理。每个线程都需要根据HTTP请求头的格式解析用户请求，然后进行逻辑处理，最后将返回的HTML文件包装成HTTP响应返回给浏览器。  最后我们会将项目部署在本地电脑上，编写一些Markdown文件，并通过Chrome浏览器进行访问，看是否能成功查看。然后还会上传一些图片，看我们的服务器能否自动进行处理和渲染，并展示在相应的页面。后续我们会不断完善扩展相关功能，并将其做我们的博客服务器。  3.3理论分析  对于第一个研究内容，将Markdown文件转化为HTML文件其实是用到了编译原理的思想。需要进行语法提取、词法分析、语法分析、代码生成一系列步骤。可以通过正则表达式匹配的方式进行语法元素抽取，通过状态机的编程思想进行语法解析，最后通过字符串拼接的方式进行HTML代码生成。目前已经有很多成熟的开源工具可以实现，我们可以直接使用或者借鉴他们的做法在现有类库上进行定制或者改写。  对于第二个研究内容，编写一个简单的静态Web服务器需要明确一次网页访问流程中涉及到的计算机网络知识，以及如何通过Java网络编程的方式进行实现。因为Java通过ServerSocket提供了传输层的抽象，所以其重点是在此基础上处理HTTP的请求解析与响应构造。我们组的成员学习了相关的知识，并通过抓包的方式研究了HTTP各类请求的结构，可以通过Java语言进行解析处理。其中网页访问就是简单的GET请求，而用户上传则是content-type为multipart/form-data格式的POST请求。后者相对来说结构更加复杂，但是其结构是有特定的标准的。我们会学习其标准，并进行解析处理。  3.4开发工具  本课题所用到的开发工具有：   1. 项目构建工具：Apache Maven（用于项目构建与jar包管理） 2. 集成开发工具：IntelliJ IDEA（用于Java语言开发） 3. 代码编辑器：VSCode（用于HTML、CSS以及JavaScript开发和Markdown预览） 4. 版本管理工具：Git&GitHub（用于项目备份与团队成员合作） 5. 网络请求模拟工具：Postman（用于模拟网络请求进行功能测试与代码调试） 6. 浏览器：Chrome（浏览器用于测试预览与页面调试）   3.5技术架构  该项目整体上分为两个模块：Markdown文件处理模块与用户请求处理模块。一次完整的项目运行流程是：首先服务器启动，Markdown文件处理模块根据配置读取本地特定目录下的Markdown源文件，进行解析并缓存到特定目录。随后用户通过浏览器发起访问，用户请求处理模块接收用户请求并进行参数解析，得到用户的请求类别。如果是页面访问请求则根据访问参数返回特定的HTML页面，如果是上传请求，则将用户上传的Markdown文件进行本地保存，并调用Markdown文件处理模块进行解析，最后将处理完成后的HTML文件返回给浏览器。  **四、本研究可能的创新之处**   1. 除了支持基础的Markdown语法以外，可以提供一些自定义的语法元素来扩展Markdown的功能。例如可以通过自定义语法设置某篇Markdown文章的展示优先级，通过自定义语法设置文章的作者、标签、所用样式模板等元信息。 2. 利用多线程与线程池提高服务器对并发访问的处理能力。   **五、项目整体开发流程分析**  5.1 项目整体架构  由上图可知，项目整体分为三个大的模块：配置解析模块、文件解析与生成模块以及静态服务器模块。其中文件解析模块内部又分为Markdown文件解析、图片文件检索以及HTML文件生成三个子模块；而静态服务器模块又分为连接管理、请求解析、文件上传请求处理、页面访问请求处理以及相应构造五个子模块。接下来，我们分模块对实现方法进行阐述：  5.1.1 配置解析模块  目前我们通过yaml文件作为配置文件的格式。当前支持的配置项包括：  （1）host：指静态服务器监听的网络地址，默认是localhost；  （2）port：指静态服务器监听的网络端口，默认是80；  （3）web：指静态服务器资源文件的根目录，生成的HTML文件以及CSS文件等在该目录中；  （4）blog：指Markdown文件的保存目录，用户上传的Markdown文件也会放在此处；  （5）theme：指网站博文采用的主题样式，可选样式有：green、dark、blue、article、article2等；  5.1.2 Markdown文件解析模块  将Markdown文件解析成HTML格式有多种方法。第一种方式是将Markdown文件手动解析成抽象语法树，然后用HTML标签渲染。这种方式需要自行处理Markdown中的各种特殊格式字符，还涉及到很多编译原理、自动机相关的知识。由于时间较为紧张，我们最终没有采用这种方式。第二种方式是借助借助Github Markdown API实现，只需通过http接口发送POST请求调用https://api.github.com/markdown，传入所需转换的Markdown文本，返回结果即为转换后的Html文档。这种方式直接调用，简便快捷，但是需要联网，而且不够灵活，完全依赖github第三方实现，无法自定义更多转换设置和细节，如无法在指定标签中加入指定属性。因此，在本项目中，我们采用第三种方式，即通过现有的Markdown文件解析库来实现该过程。经过调研以后，我们选择使用flexmark库进行Markdown的解析。Flexmark库的优势在于解析速度快，灵活性强，而且能够支持自定义抽象语法树，允许对解析过程进行精细控制，内置大量解析器和常用扩展，为解析行为和 HTML 属性样式提供了更多转换设置和选项，如gfm（GitHub Flavored Markdown）扩展用于支持gfm风格样式，Toc扩展用于创建目录和自定义目录层级等，同时也可以自定义扩展来实现业务需求，例如为标签增加自定义属性等。  Flexmark库的使用流程包括：①按需添加自定义扩展等配置；②使用配置初始化Markdown解析器与HTML渲染器；③使用解析器将Markdown文件解析成抽象语法树；④使用渲染器将抽象语法树处理成HTML格式。  **5.1.3 HTML文件生成模块**  HTML文件生成模块会生成四种类型的页面：   1. 遍历Markdown文件并调用上面的Markdown文件解析模块得到Article对象，并生成博文页； 2. 对于名称为about的博文，将其渲染成为“关于页”； 3. 根据得到的Article对象列表，生成网站首页，首页包括所有博文的链接； 4. 调用图片检索模块得到用户上传的照片列表，并据此生成“照片页”，包括所有照片的缩略图以及下载链接。     **博文页生成与关于页生成：**  **首页生成：**  **照片页生成：**  可以看到，在生成上述各类页面的时候，我们还会调用addPostStyle、addAboutStyle、addIndexStyle以及addPhotoStyle等函数为页面增加样式。主要是通过引入预设的CSS样式表与JS脚本来实现。其中博文页生成时，会根据用户配置中的主题参数来选取特定的样式，使得文章页面呈现不同的主题。  5.1.4 静态服务器链接管理模块  在该项目中，我们没有用现有的Web服务器，而是选择自己实现一个简单的静态Web服务器。对于静态服务器中链接管理的实现，我们采用多线程与线程池的方式提高并发处理能力。具体来说服务端通过ServerSocket监听特定端口，对于来自客户端的每个连接都会创建一个名为HttpTask的处理线程来进行处理，防止服务器被单一请求阻塞。  HttpTask的run函数中包含了服务器对于每个来自客户端请求的处理逻辑。具体来说，首先是将用户请求按照HTTP请求格式解析成Request对象，其中包括请求方法、请求URL、请求头字典以及请求参数等信息。然后，根据请求类型选择分别调用页面获取请求处理模块和文件上传请求处理模块。  5.1.5静态服务器请求解析模块  具体而言，HTTP请求的解析过程如下：  解析完毕以后得到的Request对象结构如下所示:  5.1.6 静态服务器网页访问请求处理模块  对于网页访问请求，是从HTTP请求中解析出目标网页，然后在本地找到该网页对应的HTML文件，将其封装成HTTP响应并返回：  5.1.6 静态服务器文件上传请求处理模块  而对于文件上传请求，则需要从中解析出所上传文件的二进制数据，分析出其文件类型，并保存在响应的本地目录中。  文件上传请求中的请求体是multipart/form-data格式的数据。其格式如下所示：  Content-type: multipart/form-data, boundary=AaB03x  --AaB03x  content-disposition: form-data; name="field1"  Joe Blow  --AaB03x  content-disposition: form-data; name="pics"; filename="file1.txt"  Content-Type: text/plain  ... contents of file1.txt ...  --AaB03x--  上面是一个此类请求的格式例子。首先是请求类型，然后是一个 boundary （分割符）。再接下来就是声明上传内容的 form-data 类型，字段名称，如果是文件的话，还包括文件名和文件类型，最后是文件的内容。我们对于该类请求的解析流程如下所示：  5.1.6 静态服务器相应构造模块  处理完文件上传请求以后，该项目又会自动调用HTML文件生成模块把用户上传的Markdown文件转化为响应的HTML文件，同时还会自动生成新的Photos.html来把用户上传的新的图片进行渲染展示。最后把响应封装成HTTP响应格式，我们项目中HTTP响应的结构定义如下：  通过继承该抽象类我们分别实现了MsgResponse，FileResponse以及JsonResponse，它们有着相同的结构与头部，只不过相应内容分别是字符串、文件数据以及Json。其中MsgRespnose用于返回服务器端的错误信息，比如“文件找不到”等。FileResponse用于返回HTML文件数据，而JsonResponse则用于返回文件上传请求的处理结果。  最终将响应返回给客户端，对应到Java里，就是把HTTP响应写入客户端socket的OutputStream中：  **5.2 目录结构与文件**  1. 配置文件：config.yaml  2. 源文件目录：   1. /md/Article.java: 博文类 2. /md/FileTranslator.java: HTML文件生成器类 3. /md/MarkdownParser.java: Markdown文件解析器类 4. /md/YamlParser.java: Yaml文件解析类 5. /server/Mime.java: 文件类型解析类 6. /server/Request.java: HTTP请求类 7. /sever/Response.java: HTTP响应基类 8. /server/MsgResponse.java: 内容为字符串的HTTP响应类 9. /server/FileResponse.java: 内容为文件的HTTP响应类 10. /server/JsonResponse.java: 内容为Json的HTTP响应类 11. /server/HttpTask.java: 客户端连接处理逻辑类 12. /server/HttpServer.java: 静态服务器类 13. /utils/BytesUtil.java: 字节数组工具类 14. /Config.java: 配置类 15. /Main.java: 启动类   **六、部分成果展示**  **（1）网站主页**：  **（2）文章页面**：  **（3）特殊格式渲染举例**：  **（4）其他主题举例**：  **（5）照片页：**  **（6）发布页：**  **（7）发布页预览文章：**  **（8）发布页预览图片：**  **（9）多文件发布：**  **（10）关于页：** | | | |