# WebServer详细设计

## 1引言

### 1.1目的

根据WebServer的需求分析和概要设计，设计每个模块的实现算法和所需的局部数据结构以及模块输入数据、输出数据及局部数据的全部细节。

### 1.2定义

WebServer一般指网站服务器，是指驻留于互联网上某种类型计算机的程序，可以向浏览器等Web客户端提供文档，也可以放置网站文件，让全世界浏览；可以放置数据文件，让全世界下载。

### 1.3参考资料

1.WebServer需求分析

2.WebServer概要设计

3.[HTTP协议头部与Keep-Alive模式详解](http://blog.csdn.net/ljianhui/article/details/19646087)

4.[服务器与客户端交互原理](http://www.cnblogs.com/zhangzt/archive/2011/09/14/2176171.html)

# 2软件详细设计

### 2.1软件总体结构



### 2.2各模块详细设计

#### 2.2.1请求接收模块

请求接收模块：用于接收客户端发送来的HTTP请求，并解析HTTP请求中的原始数据，然后将解析的结果传送到文件处理模块。

请求接收模块由类Request实现。其基本结构如下：

class Request{

InputStream input;

String uri;

Request(InputStream input);

parse();

String parseUri(String requestString);

String getUri();

}

Input为InputStream对象，其中包含HTTP请求的原始信息，可以调用read()来读取其中的原始信息。

uri为String对象，其中包含了解析之后的URI的字符串。

Request()为Request的构造函数，通过传入一个InputStream来构建Request对象。

parse()用于解析InputStream对象中包含的HTTP请求的原始信息以及调用parseUri()进行进一步解析。

parseUri()在parse()中被调用，用于从原始信息中解析出HTTP请求的具体路径并将其存入到uri中。

getUri()用于获取Request对象中的uri。

#### 2.2.2请求处理模块

请求处理模块：根据请求接收模块的解析结果（如查看文件目录的请求、预览文件的请求、下载文件的请求以及创建连接和断开连接），给出对应的处理办法，然后把处理的结果发送给请求响应模块；同时请求处理模块还需要与服务器进行文件交互以满足用户的请求。

请求处理模块由类RequestHandler实现。

class RequestHandler{

String filePath;

RequestHandler(Request request);

byte[] handler(String filePath);

}

filePath为String对象，其中存储的是Request类中处理后的uri路径。

RequestHandler()为RequestHandler对象的构造函数，通过引入一个Request对象来创建一个RequestHandler对象。

handler()通过引入的路径处理对应的文件，如果filePath指定的为一个目录则返回一个包含该目录下所有文件和子文件夹的信息的数组；如果filePath指定的为一个文件（仅限于文本文件或图片文件），则返回一个包含该文件信息的数组以提供预览功能。

#### 2.2.3请求响应模块

请求响应模块：根据请求处理模块的处理结果，将这些信息包装并发送给客户端。

请求响应模块由类Response实现。

class Response{

byte[] message

OutputStream output;

Response(OutputStream output);

setMessage(byte[] message);

sendMessage();

}

message为byte数组，用于存储RequestHandler中handler()处理的结果。

output为OutputStream对象，其中包含了HTTP响应的信息。

Response()为Response对象的构造函数，通过引入一个OutputStream对象来构建一个Response对象。

setMessage()用于将RequestHandler中handler()处理的结果存储到message中。

sendMessage()用于将要返回给客户端的message写入到output中。

#### 2.2.4 WebServer主模块

class WebServer{

String WEB\_ROOT;

String SHUTDOWN\_COMMAND;

boolean shutdown;

main();

await();

}

WEB\_ROOT为String对象，用于存储web服务器的根目录。

SHUTDOWN\_COMMAND为String对象，用于存储命令字符”-shutdown”。

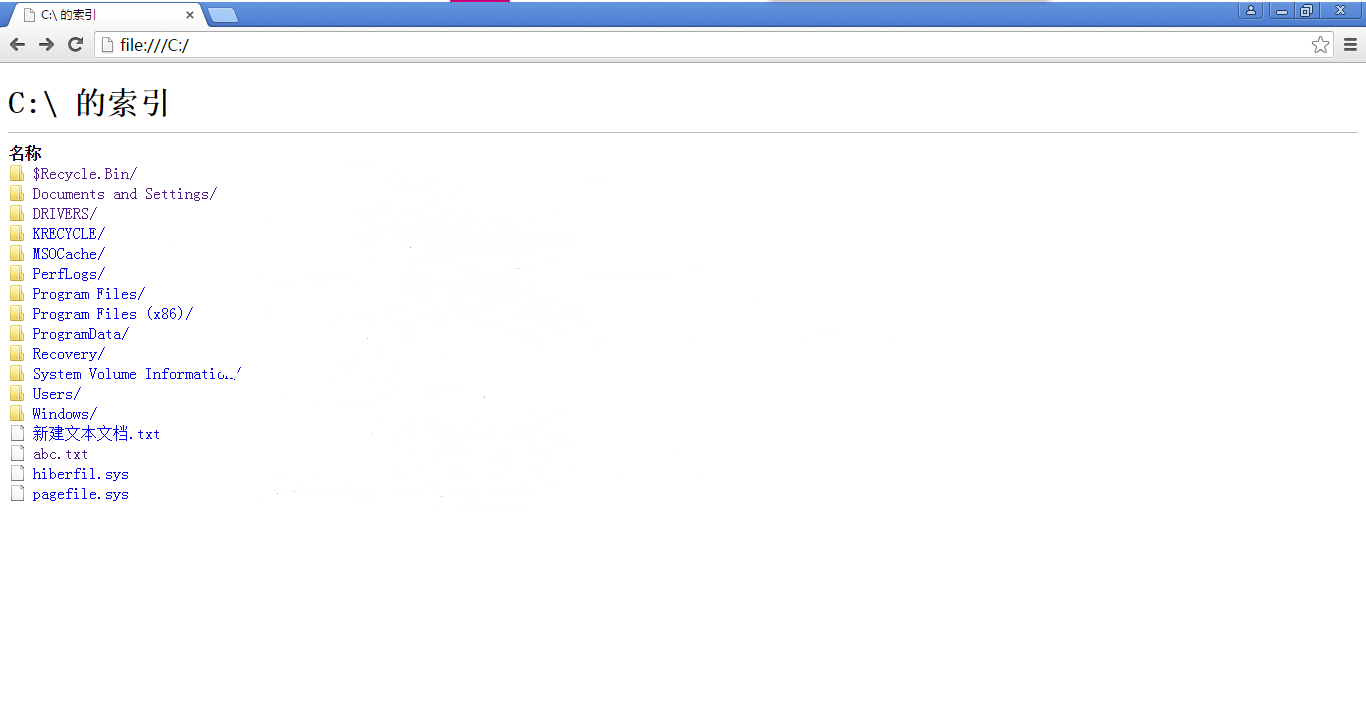
shutdown用于接收管理员在控制台中的输入来控制WebServer的关闭。

main()中创建了一个WebServer对象并调用await();

await()中创建了一个服务端套接字ServerSocket和Socket，并调用ServerSocket的accept()方法来接收和侦听Socket的请求。然后创建了InputStream和OutputStream来存储Socket中的输入流和输出流，然后用InputStream和OutputStream来构建Request、RequestHandler和Response对象来处理用户的请求。

### 2.3用户界面

1.当输入的为一个目录时，就会出现如图1所示的界面，其中显示了一些子目录和其他文件。点击子目录可以进入下级菜单，点击文件可以进行预览（仅限文本文档和图片文件）。

图1

2.当输入的为文本文档时，直接进入到文本文档的预览界面，如图2所示。

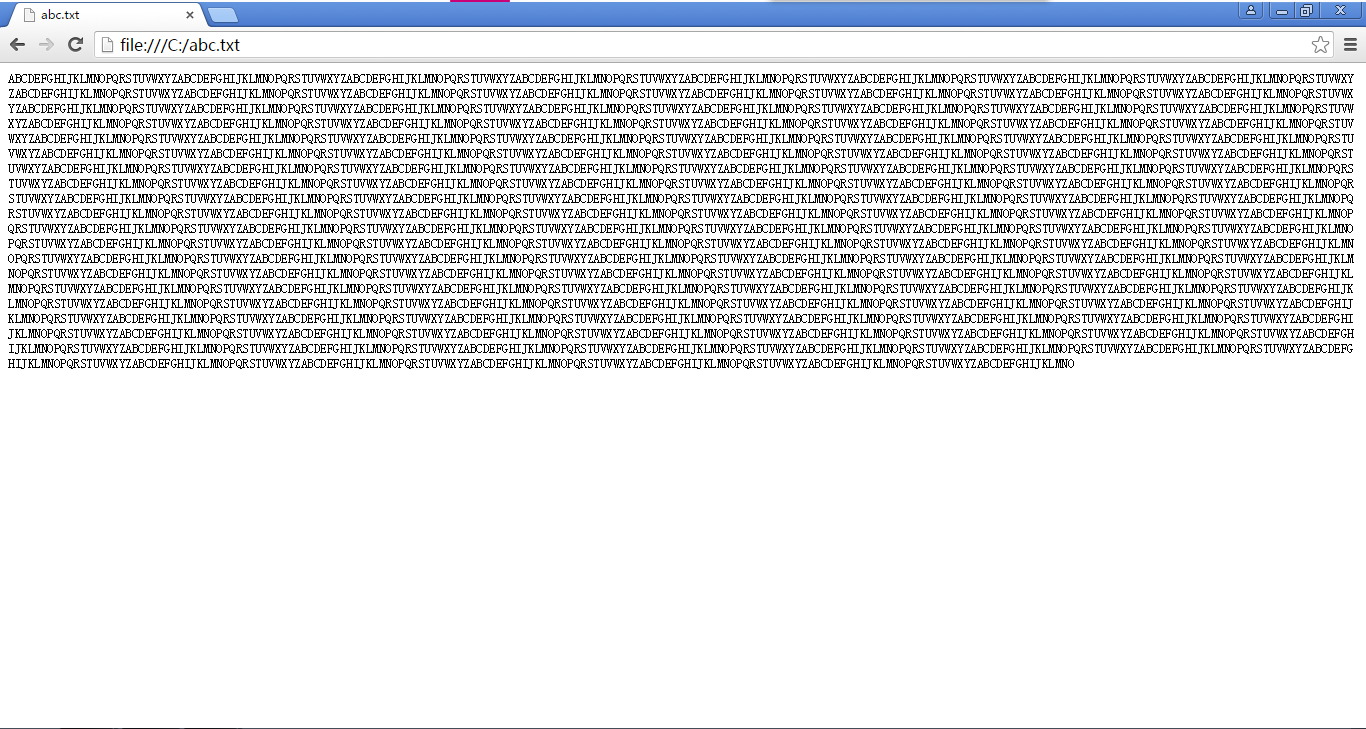


图2

3.当输入的为图片文件时，直接进入到图片的预览界面，如图3所示。

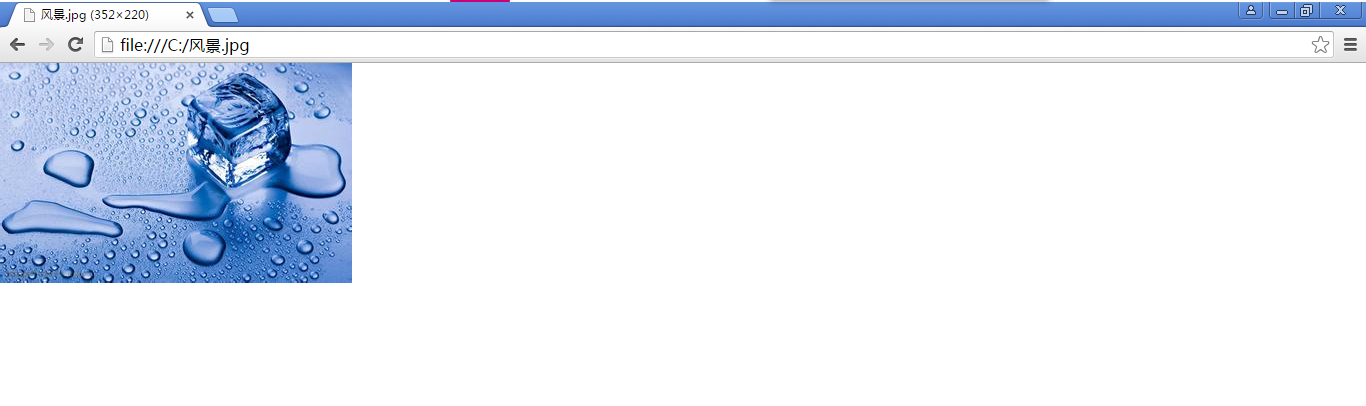


图3

4.当有下载请求时就会进入下载界面，如图4所示。

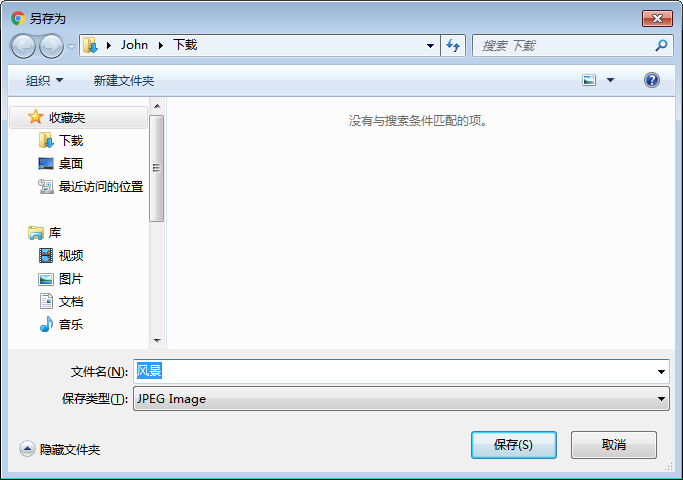


图4

### 2.4软件测试

#### 2.4.1 Request类测试

Request类测试需要先创建一个InputStream对象，然后使用构造函数来构建一个Request对象，在依次调用其成员函数parse()、parseUri()和getUri(),最后对parseUri()和getUri()的结果进行断言，若断言通过则测试通过，反之则不通过。

#### 2.4.2 RequestHandler类测试

RequestHandler类测试需要先创建一个Request对象，然后通过构造函数来创建一个RequestHandler对象，再调用handler()函数，然后对handler的结果进行断言，若断言通过则测试通过，反之则不通过。测试用例需包含对文件、目录、不存在文件的测试。

#### 2.4.3 Response类测试

Response类测试需要先创建一个OutputStream对象，然后通过构造方法创建一个Response对象，在调用setMessage()和sendMessage()，再对变量message进行断言，若断言通过则测试通过，反之则不通过。

#### 2.4.4 WebServer类测试

WebServer类的测试需要人工操作，在前三个类测试通过的基础上，运行WebServer.java，然后在浏览器中输入localhost:8080/以及其子目录，若都能访问者WebServer测试通过，反之则不通过。