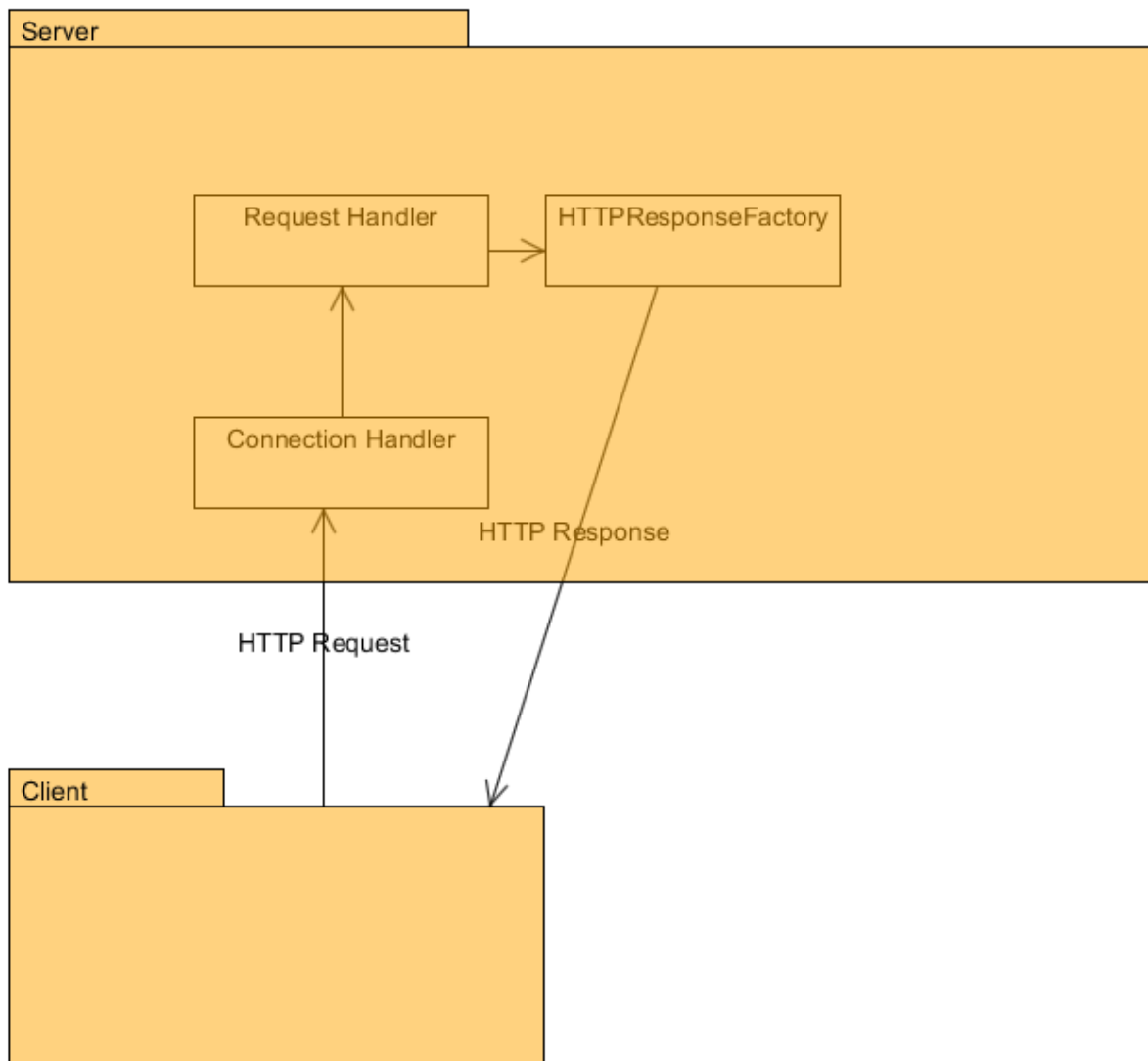


Architecture Diagram



The architecture is quite simple. The client using the HTTP protocol makes an HTTP request of the server. The server creates a connection for them, handles the request, and returns an HTTP Response.

The UML for our server follows.

Design Patterns

Factory Method – We make use of factory methods in our code getting the required handler for requests. This simplifies the code and decouples the handlers from the connection and the requests.

Thread Per Connection – We make use of this concurrency pattern to make the system be able to have more efficient multi-user support. It gives each user a dedicated thread for their requests.

Builder – We use a modification of the builder pattern when we generate the responses from the requests. There is a smaller number of parameters than is usual for the builder pattern, but it allows us to hide the complexity of creating and initializing a response from the connection handler and instead encapsulate it within the response factory.

Design Improvements

As suggested by Chandan, we should do some work with a command pattern perhaps with the generation of the responses. The way that the code currently is, all the work is done within the factory and the system is not very extensible with the responses.

Also, we could encapsulate the handling the HTTP messages into a more plugin like architecture. This would make it so that we could also handle different types of requests beyond HTTP requests without extensive modification of the system, only through adding a new plugin which respects the design we have put in place.

Testing

All testing is being done using both the Google Chrome browser (Version 42.0.2311.90 m) and the Postman REST client (v1) extension for Google Chrome, or by using the HTTP test utility supplied by Chandan Rupakheti for this project.

Get Request Successful

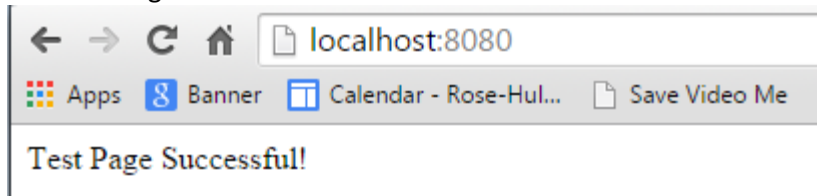
Test 1: Tested Get request on existing resource using browser

Resource: /index.html

Expected Behavior: Screen renders the test page.

Actual Behavior: Screen renders the test page.

Image:



Test 2: Tested Get request on existing resource using Postman REST client

Resource: /index.html

Expected Behavior: HTML page is returned as xml and rendered in REST client with 200 OK

Actual Behavior: HTML page is returned as xml and rendered in REST client with 200 OK

Image:

Preview Limitations

GET HTTP/1.1

Host: localhost:8080

Cache-Control: no-cache

Send

Build

Add to collection

Body

Headers (7)

STATUS

200 OK

TIME

44 ms

Pretty

Raw

Preview

JSON

XML

```
1 <html>
2   <head>
3     <title>Test Page</title>
4   </head>
5   <body>
6     <p>Test Page Successful!</p>
7   </body>
8 </html>
```

Get Request Missing Resource

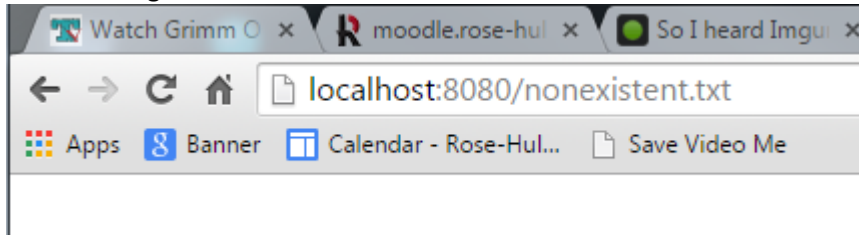
Test One: Missing resource using browser

Resource: /nonexistent.txt

Expected Result: Nothing is rendered.

Actual Result: Nothing is rendered.

Image:



Test Two: Missing resource using Postman

Resource: /nonexistent.txt

Expected Result: 404 Not Found is returned

Actual Result: 404 Not Found is returned

Image:

Preview [Limitations](#)

GET /nonexistent.txt HTTP/1.1

Host: localhost:8080

Cache-Control: no-cache

Send

Build

Add to collection

Body

Headers (4)

STATUS

404 Not Found

TIME

29 ms

Pretty

Raw

Preview

■

≡

JSON

XML

1

Malformed HTTP Request

Test One: Malformed HTTP request made with HTTP Test Utility

Resource: N/A

Expected Result: 400 Bad Request Returned

Actual Result: 400 Bad Request Returned

The screenshot shows the 'HTTP 1.1 Test Client' window. In the 'Connection Settings' section, 'Server Name' is 'localhost', 'Port Number' is '8080', 'Sleep Time (ms)' is '100', and 'Requests' is '10'. The 'Connection Request' section shows a GET request for '/index.html' with various headers. The 'Connection Command' section has buttons for 'Generate Persistent Request', 'Generate Cache Request', 'Start DOS Attack', 'Send', 'Generate Bad Request', 'Generate Version Request', 'Stop DOS Attack', and 'Clear'. The 'Connection Response' section shows the following output:

```
Connection Established!
Request Sent. Waiting for response ...
----- Header -----
HTTP/1.1 400 Bad Request
date : Sun Apr 26 14:49:21 EDT 2015
server : SimpleWebServer(SWS)/1.0.0 (Windows 7/6.1/amd64)
provider : Chandan R. Rupakheti
connection : Close
----- Body -----
Socket Disconnected!
```

The malformed piece of this packet is that we did not specify protocol.

Unsupported Method

Test One: Test unsupported method using Postman

Method: OPTIONS

Expected Behavior: 405 Method Not Allowed Returned

Actual Behavior: 405 Method Not Allowed Returned

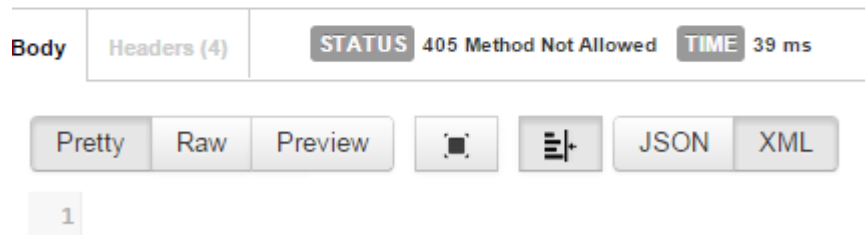
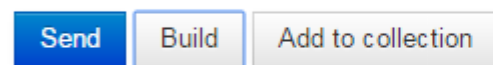
Image:

Preview [Limitations](#)

OPTIONS /test.txt HTTP/1.1

Host: localhost:8080

Cache-Control: no-cache



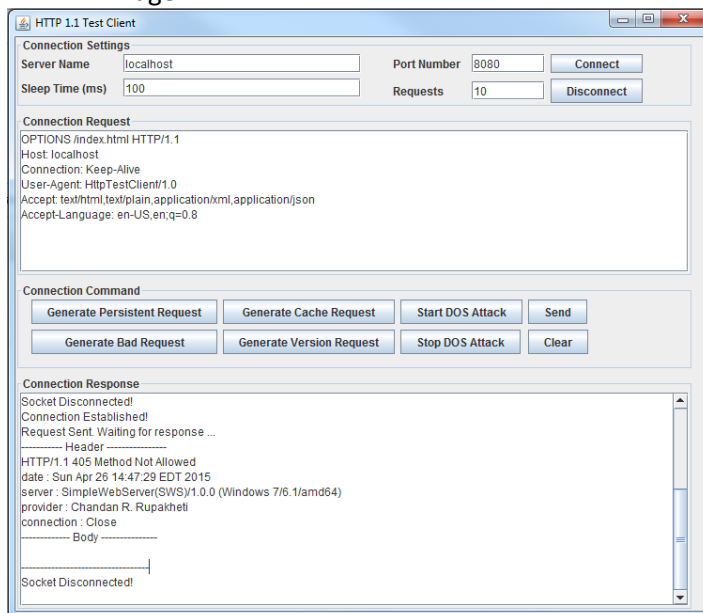
Test Two: Test Unsupported Method using HTTP Test Utility

Method: OPTIONS

Expected Behavior: 405 Method not Allowed Returned

Actual Behavior: 405 Method not Allowed Returned

Image:



HTTP Version Not Supported

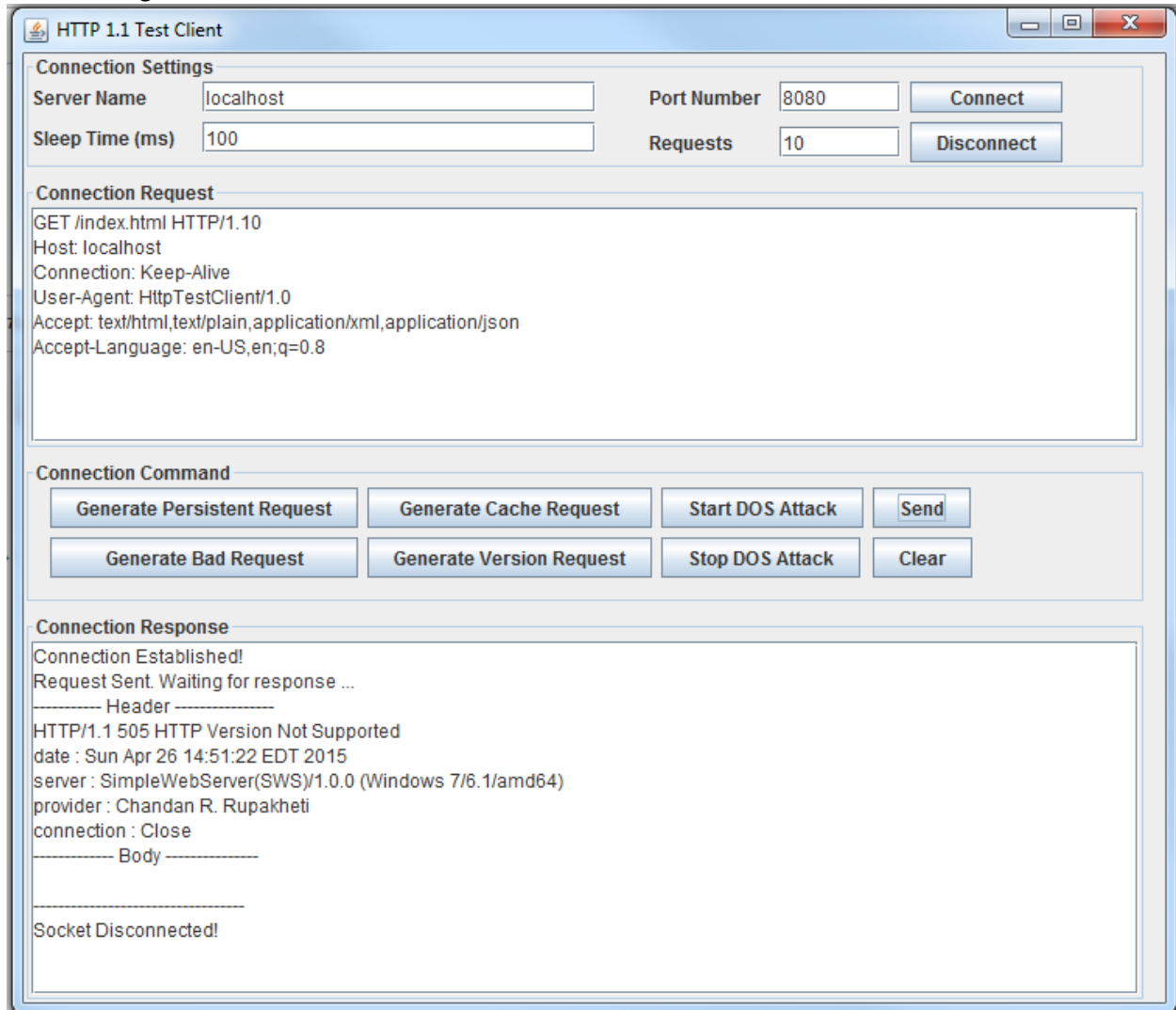
Test 1: Test version not supported using HTTP Test Client

HTTP Version: 1.0

Expected Result: 505 Version not supported returned

Actual Result: 505 Version not supported returned

Image:



The server that we have built explicitly only supports HTTP 1.1. It rejects outdated clients that are using 1.0.

Post Request

Our post request implementation writes the body of the http message to disk. This means that when form encoded data is sent, it looks strange on disk.

Test 1: Post request made from browser:

HTML Image:

```
1 <html>
2 <body>
3   <form action="http://localhost:8080/uploaded.txt" method="post" enctype="multipart/form-data">
4     <p>File: <input type="file" name="file1">
5     <p><button type="submit">Submit</button>
6   </form>
7 </body>
8 </html>
9
```

This means that it will upload said file to a file called uploaded.txt on the server.

Expected behavior: File is uploaded. When requested via GET it will be shown.

Actual Behavior: File is uploaded. Sequence below.

Image 1: Post

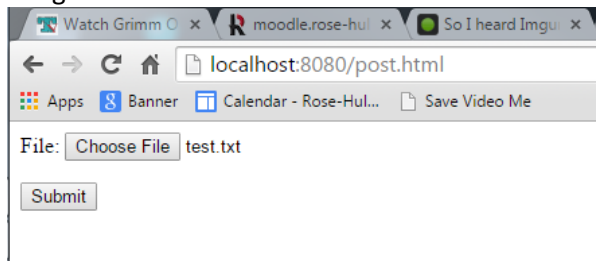
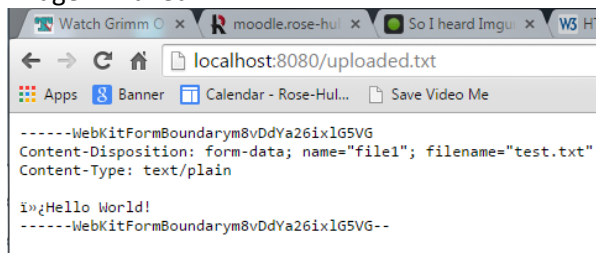


Image 2: Pulled



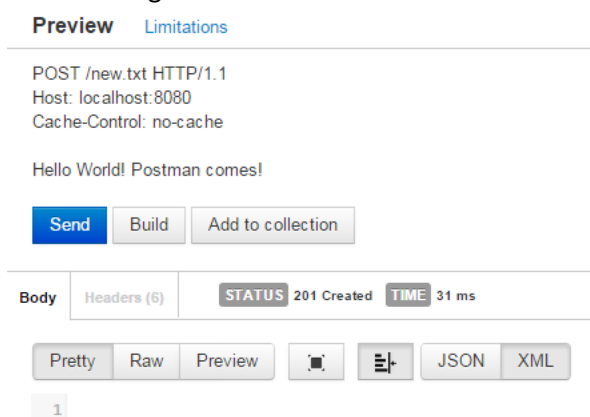
Test 2: POST made with Postman

File: new.txt

Expected Behavior: 201 Created returned and file exists.

Actual Behavior: 201 Created and file exists.

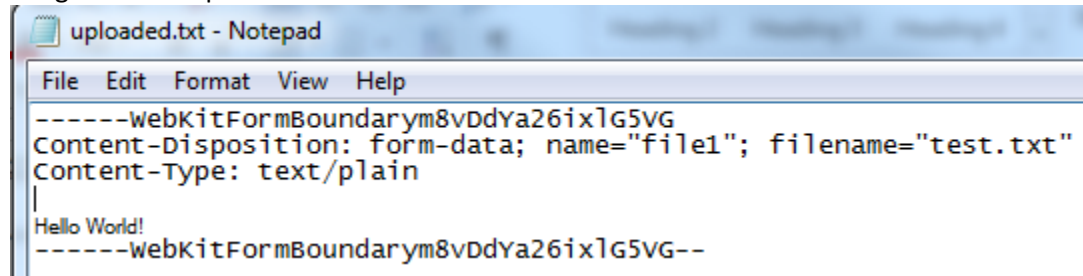
Image:



Test 3: Post made with Postman replacing file

File: uploaded.txt

Image Prior to Upload:



Expected Behavior: File Contents are replaced.

Actual Behavior: File Contents are replaced.

Image of Request:

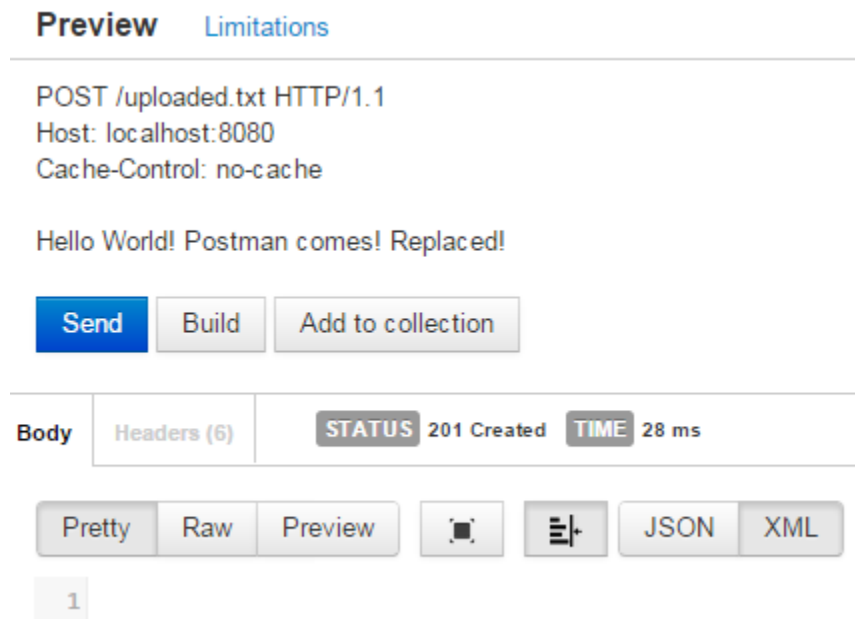
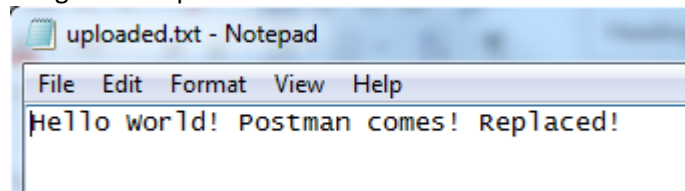


Image After Upload:



Put Request New File

Test 1: Postman with Put request and new file

File: new_put.txt

Expected Behavior: File is created and 201 Created is returned.

Actual Behavior: File is created and 201 Created is returned.

Request Image:

Preview [Limitations](#)

PUT /new_put.txt HTTP/1.1

Host: localhost:8080

Cache-Control: no-cache

Put! Postman!

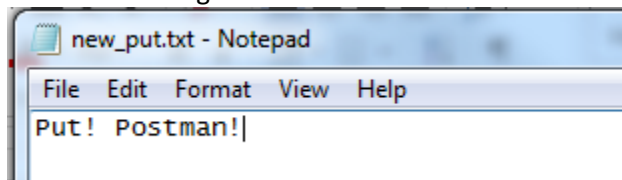
Send Build Add to collection

Body Headers (6) **STATUS** 201 Created **TIME** 56 ms

Pretty Raw Preview   JSON XML

1

File Image:



Put Request with Existing File

Test 1: Postman with existing file.

File: new_put.txt

Expected Behavior: File is appended with contents of request and 202 Accepted is returned.

Actual Behavior: File is appended with contents of request and 202 Accepted is returned.

Request Image:

Preview [Limitations](#)

PUT /new_put.txt HTTP/1.1

Host: localhost:8080

Cache-Control: no-cache

New Line!

Send

Build

Add to collection

Body

Headers (4)

STATUS 202 Accepted

TIME 45 ms

Pretty

Raw

Preview

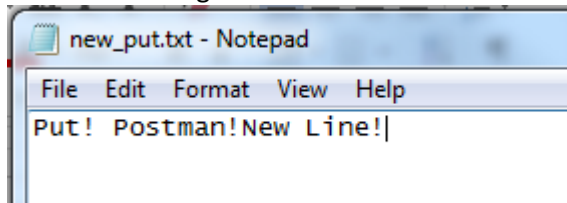


JSON

XML

1

File Image:



Delete Request on existing file

Test 1: Delete the created file new_put.txt

Expected Behavior: File is no longer in the web directory and 410 Gone is returned.

Actual Behavior: File is no longer in directory and 410 Gone is returned.

Request Image:

Preview [Limitations](#)

DELETE /new_put.txt HTTP/1.1
Host: localhost:8080
Cache-Control: no-cache

Send

Build

Add to collection

Body


Headers (4)


STATUS 404 Not Found **TIME** 28 ms

Pretty

Raw

Preview











JSON

XML

Directory Image:

Documents library		Arrange by: Folder ▼
web		
Name	Date modified	
 uploaded.txt	4/26/2015 3:50 PM	
 new.txt	4/26/2015 3:47 PM	
 post.html	4/26/2015 3:06 PM	
 rmiclient.jar	3/25/2015 11:25 PM	
 rmiserver.jar	3/25/2015 11:20 PM	
 index.html	11/5/2013 10:00 PM	

DELETE request with missing file

Test 1: Postman missing file

Expected Behavior: 404 Not Found is returned

Actual Behavior: 404 Not Found is returned

Image:

Preview [Limitations](#)

DELETE /new_put.txt HTTP/1.1

Host: localhost:8080

Cache-Control: no-cache

Send

Build

Add to collection

body

Headers (4)

STATUS

404 Not Found

TIME

33 ms

Pretty

Raw

Preview



JSON

XML

1