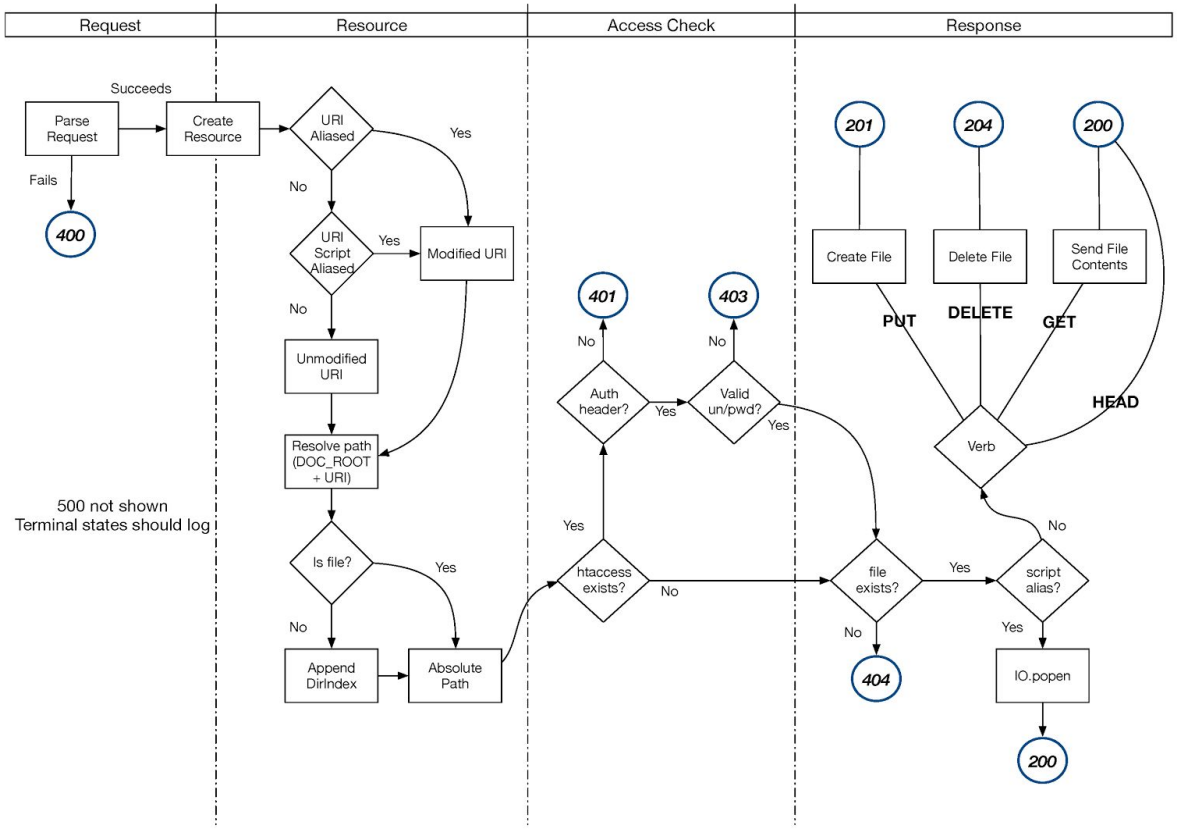
C S C 6 6 7
W E B S E R V E R
P R O J E C T

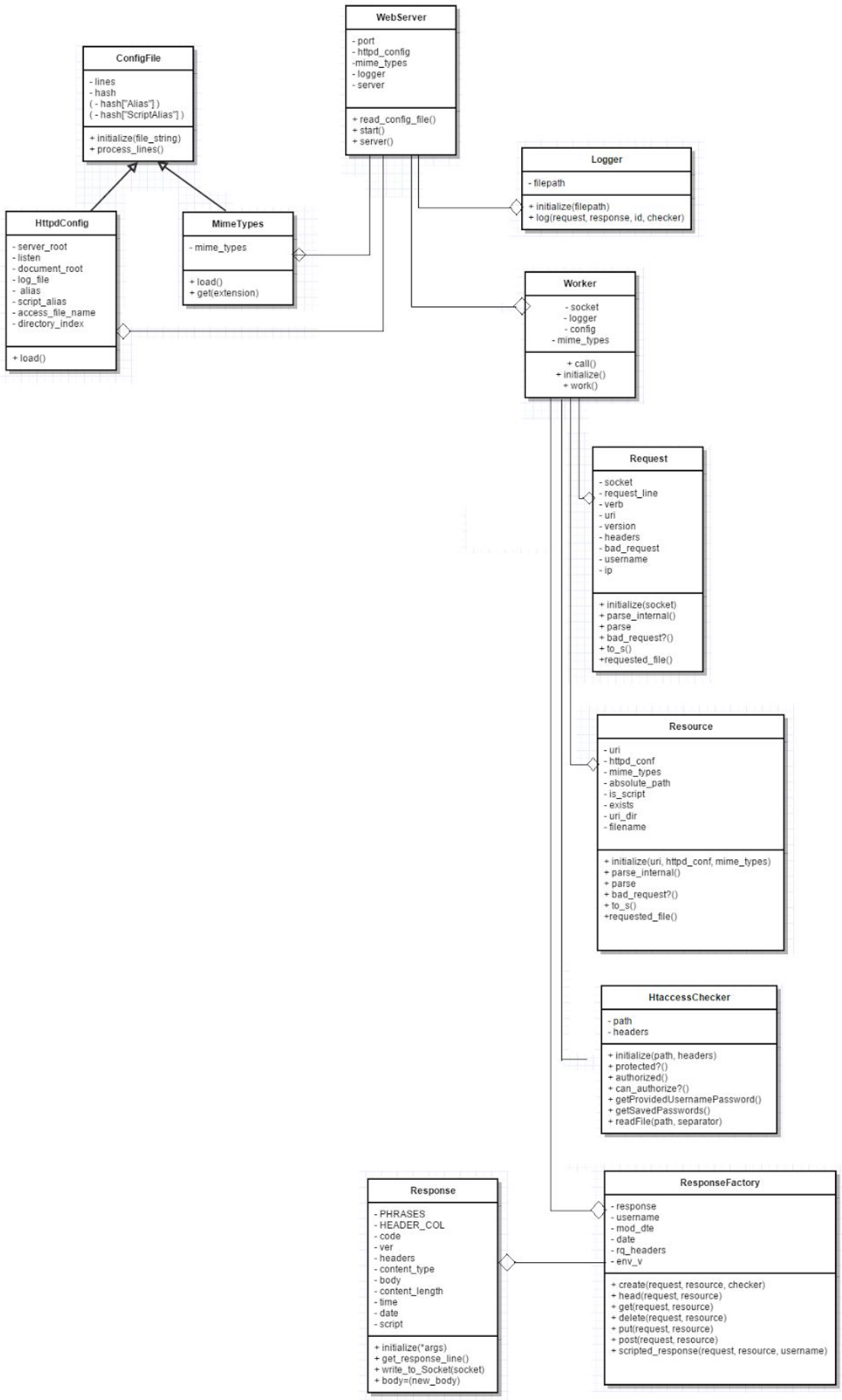
D O C U M E N T A T I O N

GITHUB-REPO: SERVER-9

A N O U R A G N A N D A
E V G E N Y S T U K A L O V
M A X I M I L I A N F I S C H E R

Architecture




WebServer
- port
- httpd_config
-mime_types
- logger
- server
+ read_config_file()
+ start()
+ server()
ConfigFile
- lines
- hash
(- hash["Alias"])
(- hash["ScriptAlias"])
+ initialize(file_string)
+ process_lines()
Logger
- filepath
+ initialize(filepath)
+ log(request, response, id, checker)
HttpdConfig
- server_root
- listen
- document_root
- log_file
- alias
- script_alias
- access_file_name
- directory_index
+ load()
MimeTypes
- mime_types
+ load()
+ get(extension)
Worker
- socket
- logger
- config
- mime_types
+ call()
+ initialize()
+ work()
Request
- socket
- request_line
- verb
- uri
- version
- headers
- bad_request
- username
- ip
+ initialize(socket)
+ parse_internal()
+ parse
+ bad_request?()
+ to_s()
+requested_file()
Resource
- uri
- httpd_conf
- mime_types
- absolute_path
- is_script
- exists
- uri_dir
- filename
+ initialize(uri, httpd_conf, mime_types)
+ parse_internal()
+ parse
+ bad_request?()
+ to_s()
+requested_file()
HtaccessChecker
- path
- headers
+ initialize(path, headers)
+ protected?()
+ authorized()
+ can_authorize?()
+ getProvidedUsernamePassword()
+ getSavedPasswords()
+ readFile(path, separator)
Response
- PHRASES
- HEADER_COL
- code
- ver
- headers
- content_type
- body
- content_length
- time
- date
- script
+ initialize(*args)
+ get_response_line()
+ write_to_Socket(socket)
+ body=(new_body)
ResponseFactory
- response
- username
- mod_dte
- date
- rq_headers
- env_v
+ create(request, resource, checker)
+ head(request, resource)
+ get(request, resource)
+ delete(request, resource)
+ put(request, resource)
+ post(request, resource)
+ scripted_response(request, resource, username)

Test Plan

Unit/Integration tests
Before integrating all the code, we used driver programs to test various classes or groups of classes in isolation, for example:

```
mime_types = MimeTypes.new( lines_from_mime_types )
mime_types.load
assert_equal(mime_types.get("html"), "text/html")
```

System tests
Done using curl and ruby's Net::HTTP class.

`GET`
Verify that code 404 is returned as a result of of a request for a missing file.
Verify that code 200 and the file body is returned in response to a valid request.

Root directory:
```
curl localhost:2345/index.html
curl localhost:2345
```

Subdirectory:
```
curl localhost:2345/subdir/index.html
curl localhost:2345/subdir/
```

Alias:
```
curl localhost:2345/aliastest/index.html
curl localhost:2345/aliastest/
```

ScriptAlias:
script alias directory is correctly resolved, the specified script is executed, and the script output is returned in the response.
```
curl localhost:2345/cgi-bin/script.pl
```

Htaccess:

```
curl localhost:2345/protected/index.html
curl localhost:2345/protected/
```

Request for a protected resource without auth header should result
in status code 401.

Request for a protected resource with incorrect username/password in auth header should result in status code 403.

Request for a protected resource with valid username/password in auth header should be granted.

PUT
Verify that this request creates or replaces the file with the given URI and content and response status is 201.

POST
Verify that code 404 is returned as a result of of a request for a missing file.
Verify that code 200 and the file body is returned in response to a valid request.

HEAD
Verify that code 404 is returned as a result of of a request for a missing file.
Verify that code 200 in response to a valid request.
Response has no body, but the file size of the file in request URI is specified in Content-Length header.

Status code 400
Try the following (at least Google web servers return status 400 as a result of this request sent to google.com:80 from Netcat):
Use the netcat utility to send a garbled HTTP request, and verify a response with status 400:

```
$nc localhost 2345
```

Paste the following garbled request to console (this worked on Mac):

```
/ HTTP/1.1
User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0
OpenSSL/0.9.8x zlib/1.2.5
Host: www.google.com
Accept: */*
```

Status code 500
We can try something like the following:

In httpd.conf, specify an alias without a path.
Implement alias handling so that this causes a compile error (like "undefined method for `nil:NilClass`"). Implement exception handling in WebServer or Worker so that a 500 status response is returned if this error happens. From client, make a request for the alias without a path. Verify status 500.

Respond to multiple requests simultaneously
Start a request for a large file over a slow connection: (Request 1)
`curl -i --limit-rate 20 localhost:2345/large_file.zip`
While "Request 1" is being processed, start a second request for a small file (Request 2)
`curl -i localhost:2345/index.html`
Processing Request 2 should be completed while Request 1 is still being processed.