|  |
| --- |
|  |
| WEB PROXY SERVER |
| Network Architecture-I |
|  |
| **Team:**  L V Sandeep Bathina(16174348)  Ritesh kumar reddy Kuchukulla(16173317)  Sundeep kumar Alampally(16171651) |

**Team Members and Roles**:

* LV SANDEEP BATHINA (16174348) – Developed Web Proxy server
* RITESH KUMAR REDDY KUCHUKULLA (16173317) – Developed Cache for storing the files
* SUNDEEP KUMAR ALAMPALLY (16171651) – Developed Client module with applets embedded in it

**Tools used:**

Platform: - Java, Java AWT

Tool: - Eclipse

**OBJECTIVE:-**

The primary goal of this project is to illustrate the working of a web proxy server. To embark on this goal we use few more modules such as: Client, Server and Cache. Here we accomplish the project in such a way that the client sends the request and server responds to it through the proxy server and those files are saved in the cache. And our objective is that whenever we get identical response from the server for the same URL the request is not forwarded to the server as we get the response from the web proxy. The front end part is implemented by Java applet while the back end is implemented through java.

**Architecture overview:-**

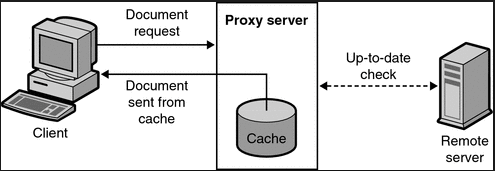
**Client**: - Part of the network edge. It is the one which send requests to the server in the form of URL.

**Server**: - Part of the network edge. It is the one which responds to the requests from the client.

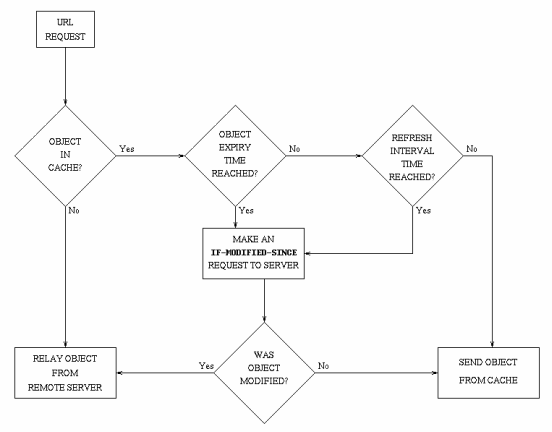
**Web** **proxy**:-Present in between the server and the client, in short it acts as an interface between client and server. Requests send to the server should pass through the web proxy.

**Cache**: - Memory which stores the specific fields such as:- header information, content of the URL for requests send by the client.

**Architecture diagram:-**



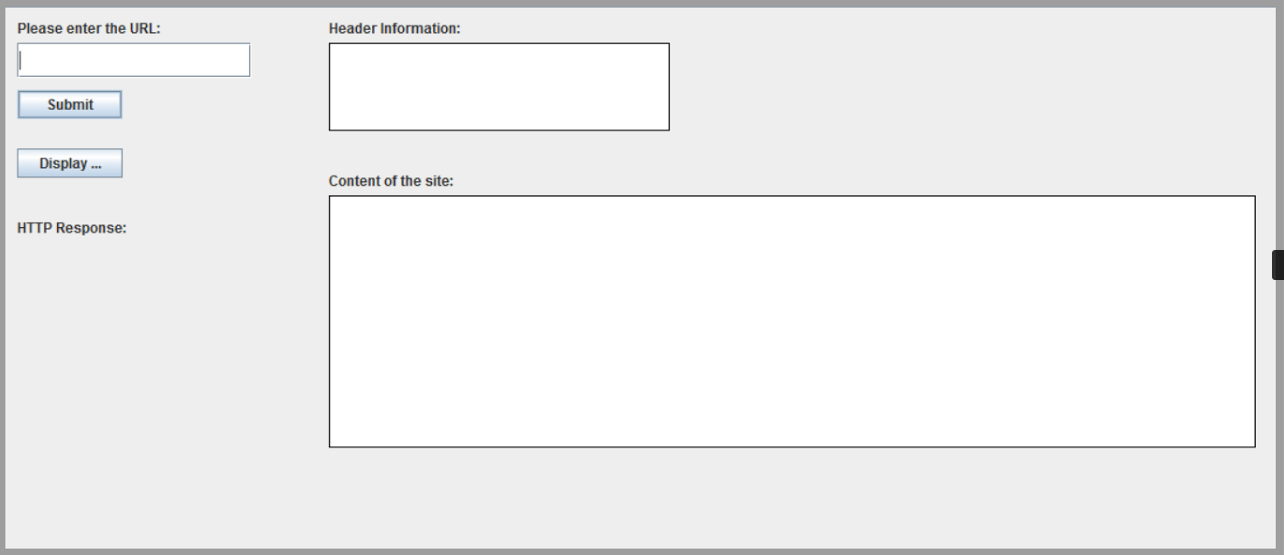
**Procedure:-**



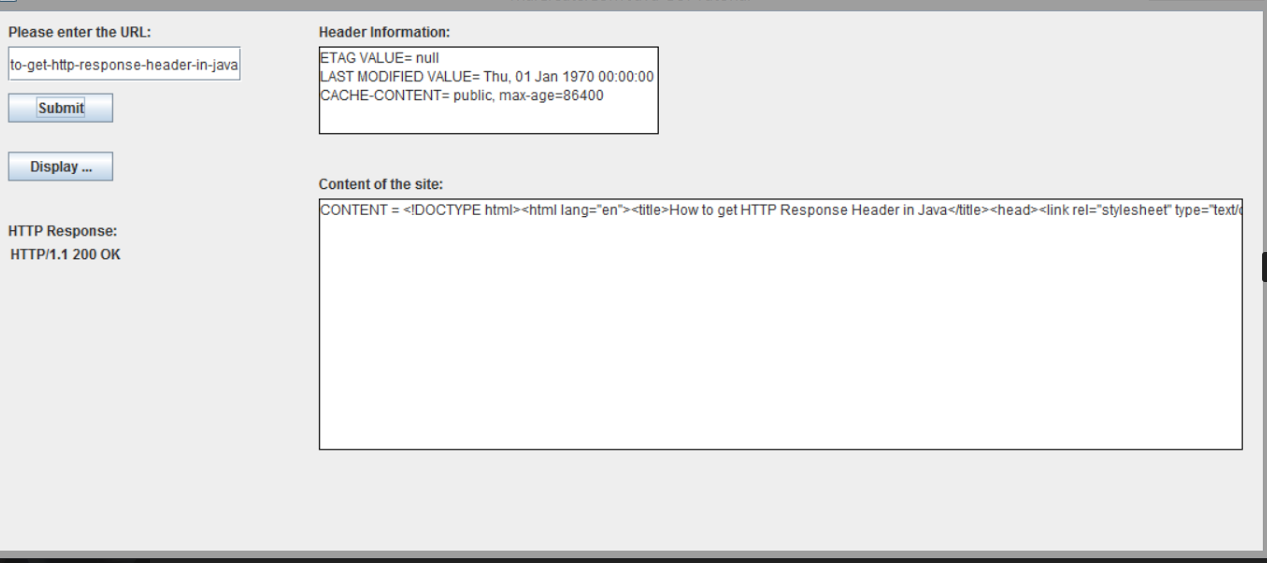
* Request from client.
* Request Acceptance by Web Proxy.
* Get header information of the URL from the server.
* Checks the Etag and last modified information from the server information with the cache content.
* If the content matches the HTTP response will be ‘304 NOT MODIFIED’ and the content will be displayed to the client from the cache file.
* If the content doesn’t matches the request is forwarded to the server and the content will be displayed to the client from the server via web proxy with HTTP response as ‘200 OK’. In parallel the last modified content will be copied into the cache.

**Project Screenshots:-**

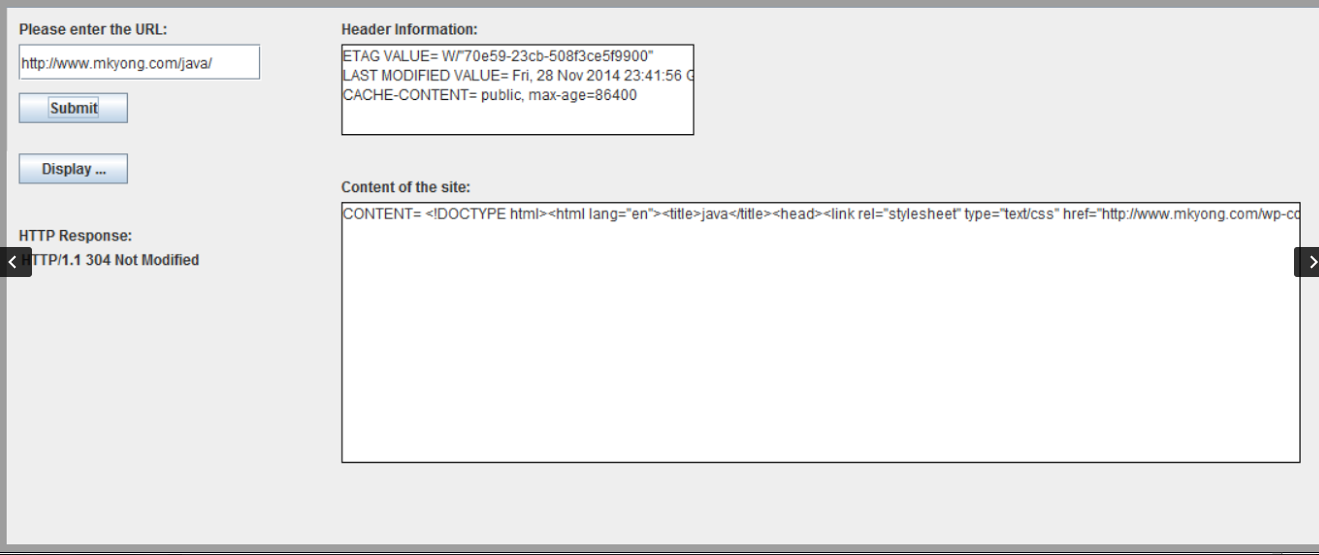
USER INTERFACE APPLET:-

****

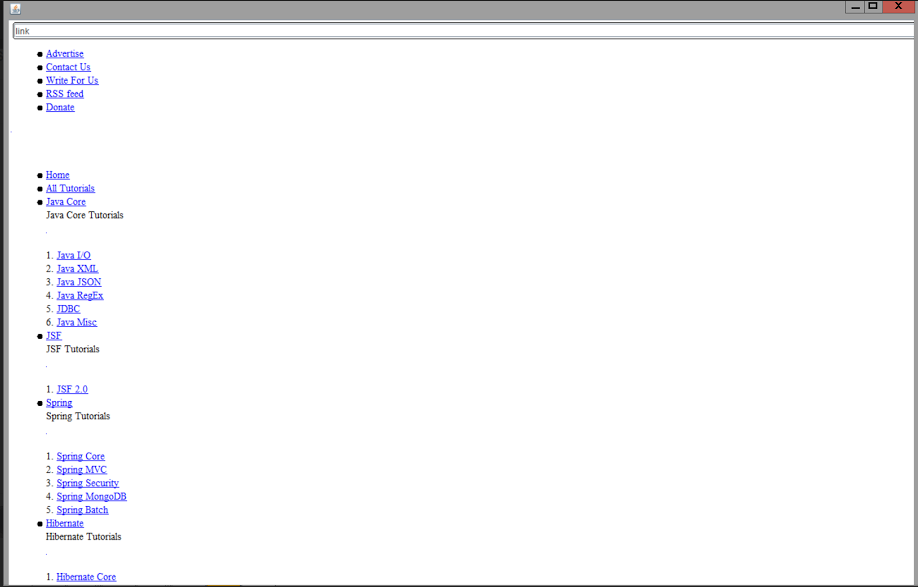
URL OF 200 OK RESPONSE:-

****

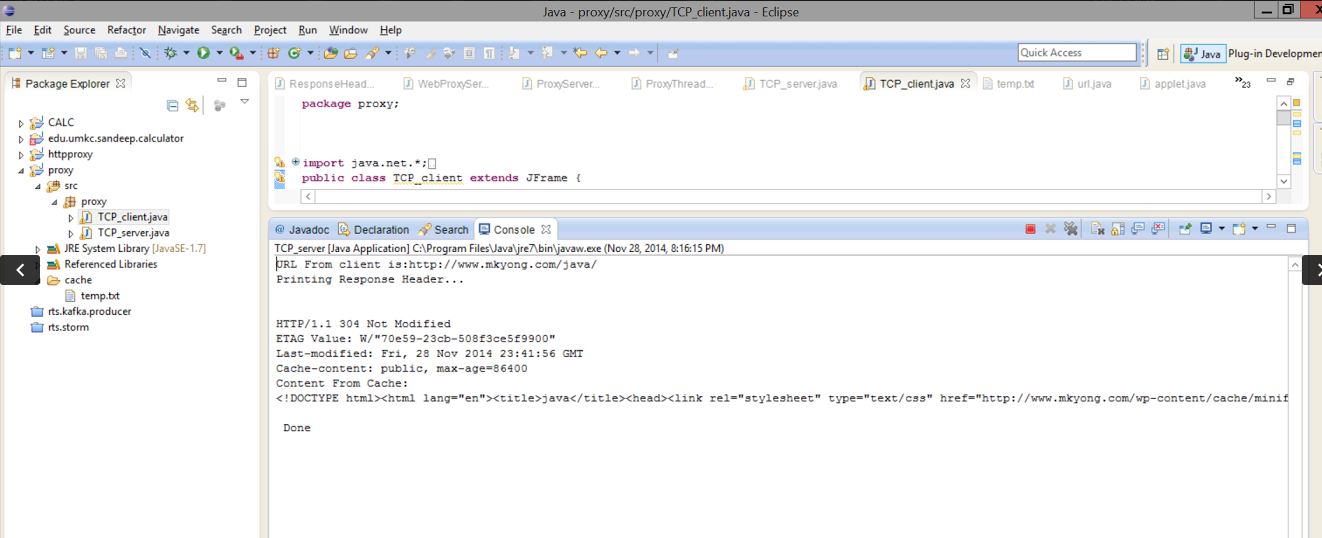
URL OF 304 NOT MODIFIED RESPONSE:-

****

DISPLAY CONTENT IN HTML PAGE:-

****

RESPONSE MESSAGE OF PROXY SERVER:-

****

**Problems Faced:-**

* Storing the information in the cache.
* Comparing the ‘LAST MODIFIED’ and ‘ETAG’ values.

**Solutions to overcome:-**

* Initially, we thought of going with database but due its complexity we made way for file systems where we stored specific header fields and content.
* Using the logical operations we could make a decision.

**Distinctive features:-**

* Java applets, a dramatic feature which takes the URL and displays the response with header information and content.
* Content obtained is displayed in the form of HTML page.

**References:-**

[**http://www.tutorialspoint.com/java/java\_applet\_basics.htm**](http://www.tutorialspoint.com/java/java_applet_basics.htm)

[**http://www.javaworld.com/article/2077322/core-java/sockets-programming-in-java-a-tutorial.html**](http://www.javaworld.com/article/2077322/core-java/sockets-programming-in-java-a-tutorial.html)

[**http://www.javacreed.com/caching-made-easy-with-spring/**](http://www.javacreed.com/caching-made-easy-with-spring/)

[**https://devcenter.heroku.com/articles/jax-rs-http-caching**](https://devcenter.heroku.com/articles/jax-rs-http-caching)