Laboratorio #02

HTTP RFC 2616

Ciencias de la Computación VIII

Resumen:

Desarrollar una aplicación de servidor web capaz de manejar los Request HTTP entrantes y enviar los Response HTTP adecuadas. Esto incluye el manejo de conexiones TCP, procesamiento de solicitudes HTTP mediante las clases Request y Response, y asegurar que el servidor pueda manejar múltiples solicitudes concurrentes mediante el uso de un ThreadPool, todo lo mencionado anteriormente se le entrega funcionando al alumno, por lo que se busca implementar las especificaciones del protocolo HTTP, sino que también enseña a manejar correctamente los Request y Response de HTTP, el procesamiento de diferentes tipos de contenido y la gestión de errores.

Componentes a Implementar

Clase Request

La clase Request debe ser capaz de:

- Procesar las solicitudes HTTP que llegan al servidor.
- Leer tanto la cabecera como el cuerpo de la solicitud.
- Clasificar los métodos de solicitud (GET, POST, HEAD).
- Almacenar las cabeceras en una estructura adecuada (por ejemplo, HashMap o ArrayList).
- Extraer correctamente el cuerpo de la solicitud y manejar diferentes tipos de contenido (multipart/form-data, application/json, etc.).
- Analizar los encabezados recibidos para pasar la información solicitada a la clase Response.
- Manejar errores de manera adecuada para devolver respuestas de error según sea necesario.

Clase Response

La clase Response debe ser capaz de:

- Generar respuestas HTTP basadas en la solicitud recibida.
- Manejar diferentes tipos de respuestas según el tipo de contenido solicitado (HTML, JSON, JPEG, CSS, etc.).
- Asegurar que el Content-Length en la respuesta sea correcto y que el cuerpo de la respuesta se envíe adecuadamente.
- Agregar los encabezados necesarios para que el navegador no considere la respuesta como un error.
- Preprocesar archivos .cc8, que son archivos HTML con tags especiales {keyValue} que deben ser reemplazados por los valores correspondientes provenientes de los formularios en los métodos GET y POST.

Pruebas y Validaciones

Se proporcionan pruebas dentro de la carpeta ./www/ que incluyen varios escenarios:

- index.html --> Index de Contenido
- test01/
  - index.html--> CSS, Imágenes y Font
- test02/
  - index.html--> Formularios POST y GET
- test03/
  - index.html--> Página Web Completa Funcional

Procesamiento QUERY del GET y Body del método POST

Para los formularios usted debe de tokenizar según el formato del método que el cliente está enviando por medio del Request.

Ejemplo POST:

Request Header POST : application/x-www-form-urlencoded

```
POST /Test.cc8 HTTP/1.1
Host: www.galileo.edu
Connection: keep-alive
Content-Length: 68
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0

class=CC8&year=2020&email=yorch%40galileo.edu&text=Texto+Prueba+POST
```

En en **test02** y **test03** existen archivos con extensión **cc8** esto es solo por mera nomenclatura de nuestro servidor de CCVIII ya que estos son archivos HTML con TAGS, así como lo es un servidor APACHE que trabaja con archivos PHP y su salida es un archivo HTML preprocesado. Cual es la idea detrás de estos archivos, en su interior van a tener una estructura HTML y su **Http Response** va a ser **content-type text/html** dentro de ellos existen **TAGs** de la forma **{keyValue}** donde estas hacen referencia a los **<input name="keyValue">** del formulario que se envían, Query en los método GET y Body Request en los métodos Post, usted debe de preprocesar estos valores según el formato y reemplazar el TAG con la variable correspondiente.

Notaciones de código entregado:

Se entrega un Servidor Multithreading implementado con Threadpool, este servidor ya realiza la lectura del **Request Header HTTP y su Body** si existe a través Socket TCP, se establece una estructura de ejemplo de almacenamiento, también se escribe sobre el mismo Socket un **Response Header HTTP** mínimo y estático, que sirve como ejemplo de estructura para enviar un archivo leído desde la carpeta ./www/ entregada junto al laboratorio. No puede modificar los archivos **Server.java**, **ThreadServer.java** y **FormatterWebServer.java**, usted puede crear nuevos archivo y clases además de modificar **Request.java** y **Response.java** sin modificar la firma de la función.

Entrega

- Suba un archivo ZIP al GES con todos los archivos fuente.
- La calificación se realizará de manera presencial, en dado caso que no pueda presentarse enviar un video Explicando su Código, mostrando la compilación y ejecución, hacer varios ejemplos de su Funcionalidad leyendo los LOGs.
- Use Java 21 o 22 para su implementación.
- El laboratorio puede tener una calificación de cero si:
    - Si no compila con el Makefile (en java 21 o 22)
    - Si los LOGS no tienen el formato establecido.
    - Si la consola del Cliente arroja un error.
    - Si no siguió las instrucciones del funcionamiento.
    - Si se modifica la estructura dada.

Referencias:

- Wikipedia - Hypertext Transfer Protocol v1.1
- RFC 2616 - HTTP
- RFC 1867 - Form-based File Upload in HTML
- RFC 7578 - multipart/form-data
- RFC 4627 - Application/json Media Type for JSON
- RFC 2045 and 2046 - application/octet-stream