AREP Taller 03 - Socket en heroku

Jeisson Geovanny Sanchez Ramos

Agosto 2020

1 Introduccion

En el presente documento no adentraremos en la construcción de un servidor http, vamos a analizar un poco el contenido de las peticiones y vamos a construir un mecanismo para poder dar respuesta de una manera óptima.

También tendremos en cuenta que con motivos de prueba se nos solicitó construir un servicio, del cual miraremos su construcción desde esta perspectiva.

2 Requerimientos

Para este trabajo se nos pidió construir un servidor que tuviera la capacidad de responder a archivos de texto, imágenes y funciones de tal manera que sea una aproximación a la libreria SparkJava ya existen, para este trabajo únicamente construiremos el método GET.

3 Especificación del servidor

Primero que todo quiero analizar la vista mas general de las operaciones que realiza un servidor web con las especificaciones dadas.

- Lectura y escritura sobre el socket.
- Lectura de contenido estático.
- Procesamiento de Request.
- determinar el reponse

4 Lectura y escritura sobre sockets

Para esta tarea se decidió crear una clase independiente ya que esto facilitaría la construcción de la aplicación, ya que al momento de ejecutar las solicitudes sería tan fácil como invocar un método y esperar por la respuesta, para la

construcción de estos métodos se decidió que recibieran y/o respondieran con strings correspondientes a la respuesta a escribir y/o recibir.

Ahora quiero comentar que en esta fase contábamos con la posibilidad de construir un request pero con el fin de no complicar esta lectura se decidió asignar esta responsabilidad a un tercero que miraremos más adelante.

Estas operaciones las podemos encontrar implementadas dentro de la clase ReadWriter del código.

5 Lectura de contenido estático

Como pudimos ver en la sección anterior una buena idea es separar responsabilidades y como ya sabemos esto puede ser beneficioso a nivel de mantener el código y extenderlo, así que la lectura no es la excepción por lo tanto dentro del código de nuestro servidor nos encontraremos con una clase llamada FileSolve, esta clase se encarga de determinar la existencia de un archivo dentro del directorio resources de nuestro proyecto java, este lee los archivos y nos entrega el buffer de la imagen para su posterior procedimiento dentro de la fase de dar un response..

6 Procesamiento de Request

Construir una respuesta nueva es valioso ya que mediante este podemos acceder a alguna información que nos puede ayudar a construir nuestros programas como lo son los parámetros que se pueden pasar mediante la url, y es precisamente por esta razón que decidimos construir un objeto request.

Ahora surge una pregunta importante a considerar a quien le deberíamos asignar la responsabilidad de construir un objeto request, para ello se decidio asignarle al mismo request la tarea de tomar este texto que es la solicitud y nos devolviera un objeto con la información que nos es necesaria para desarrollar una solución.

Como podrán notar si examinaron el código, en la clase request existen otros atributos como el body que no llegan a ser utilizados y estos fueron dejados pensando en futuras versiones para las implementación de POST, PUT, DELETE las cuales cuentan con un payload.

7 Determinar el response

Esta tarea no la delegamos si no que permitimos que la misma clase servidor se encargara dado un request darle la debida solicitud de lectura y escritura a los archivos, o determinara si es una función a quien debería responder un path especifico.

8 ¿Y el manejo de funciones?

Para administrar las funciones que podemos tener asociadas a una petición y un path construimos un diccionario cuyo valor es precisamente la combinación de método Http y path.

Esta tarea si se la asignamos específicamente a nuestro servidor ya que no implica mayor costo o complejidad.

Para esta fase es donde encontramos el valor real de limpiar los datos del request ya que, si no obtenemos el path real, no sería posible responder a algunas peticiónes.

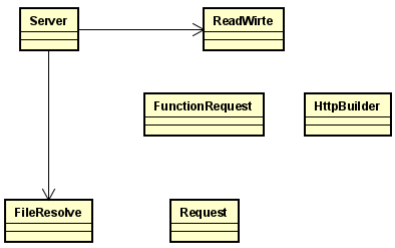
9 Http cabeceras

Para la respuesta se decidió construir una clase externa que se encargara de administrar los encabezados.

10 Puesta en marcha del servidor

En este punto necesitamos construir una función que nos ejecute las fases anteriores de construir un request y responder, para ello se utilizó una función llamada loadProcess.

Pero también hay otro detalle, necesitamos añadir funciones a la aplicación a la vez que el servidor se inicializa y responde, para ello se decidió lo más conveniente era construir un hilo de tal manera que este se encargara de ejecutar continuamente la operación loadProcess para leer peticiones simultáneamente.



Estructura del servidor.

11 Dando una fachada al servidor

Como el objetivo de este trabajo es construir una librería tipo Spark, nos es necesario poder ofrecer operaciones estáticas, para ello se decidió como comúnmente

la tarea principal debería ser dar el puerto de ejecución, esta trabajaria como el inicio del servidor.

Para el servidor, como puede generar errores al haber múltiples instancias de este corriendo por el mismo puerto, se decidió implementara el patrón singleton, ya que este nos restringe a que exista una única instancia del servidor el cual pueda responder.

Algunas otras operaciones que construimos estáticas fueron poder cambiar el directorio para servir contenido estático y el método get que nos añade un procedimiento para dar un response.

12 Probando el servidor

Con motivos de prueba del servidor implementar 2 funciones, la primera nos va a recolectar desde la base de datos un objeto de tipo persona, para traerlos se utilizó la librería nativa de java.sql, pero por motivos de facilidad los datos fueron mapeados a objetos (Esto implica cálculos extra), pero es beneficioso ya que nos facilita el manejo al igual que para generar un response se decidió crear una clase que nos devolviera un string que representara un objeto json.

Para la segunda función se decidió registrar una persona, pero como todavía no disponemos del método post por la definición del programa se decidió pasar los datos como parámetros en la url.

En la vista principal que es un html consumimos el api gracias a jquery, además estamos trayendo los archivos javascript y css.

El servidor también tiene dentro de sus archivos estáticos una imagen llamada porter2.PNG.

13 Concluciones

Dentro del presente documento analizamos las implicaciones que tiene construir un servidor web, pensar en la lectura y la escritura y como modularizar el programa de tal manera que al final sea tan simple como llamar 2 o 3 funciones, y eso es lo que podemos ver en el método loadProcess de la clase server, una abstracción simple del programa que aunque en tamaño es extenso se presta para modificaciones y sumar nuevas tareas minimizando el riesgo de tener que realizar grandes modificaciones para por ejemplo implementar los métodos PUT, DELETE o POST.

La parte más compleja de la construcción de este programa se encuentre en resolver la petición de la imagen, ya que como esta no es texto y no es tan simple como tan solo unir el encabezado al texto, si no que por el contrario la cabecera dependía de la cantidad de bytes que representaba la imagen.