Arquitectura del Web Service Framework (Septiembre 2019)

Carlos Andrés Medina Rivas
carlos.medina-ri@mail.escuelaing.edu.co
Escuela Colombiana de Ingeniería Julio Garavito

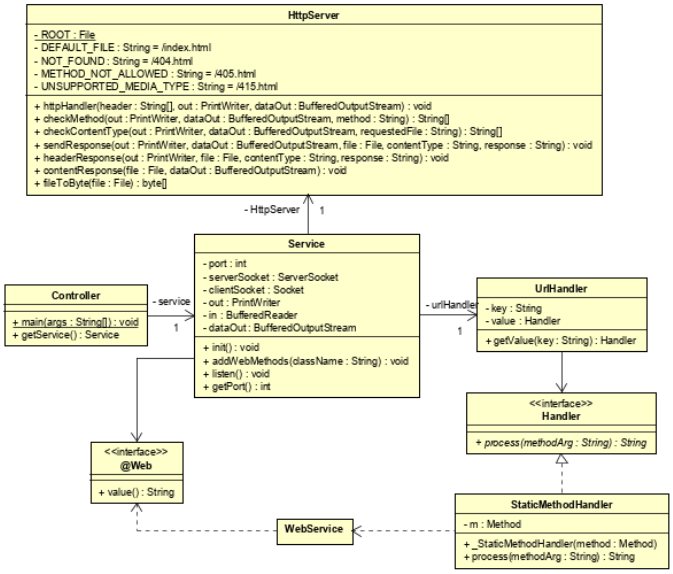
I. Introduction

Este documento tiene como objetivo mostrar la arquitectura, implementación y pruebas del Web Service Framework que permite a un desarrollador crear un servicio web usando anotaciones sin preocuparse del funcionamiento interno del servidor HTTP.
Este web Service Framework permite realizar peticiones a recursos estáticos y también permite al desarrollador crear sus propios métodos que respondan a peticiones usando anotaciones conocido como POJOS (Plain Old Java Objects).

II. Arquitecturas

Los siguientes diagramas de arquitecturas permitirán entender mejor cómo se construyó el Web Service Framework y cómo son las comunicaciones entre los diferentes componentes.

A. *Diagrama de Clases*

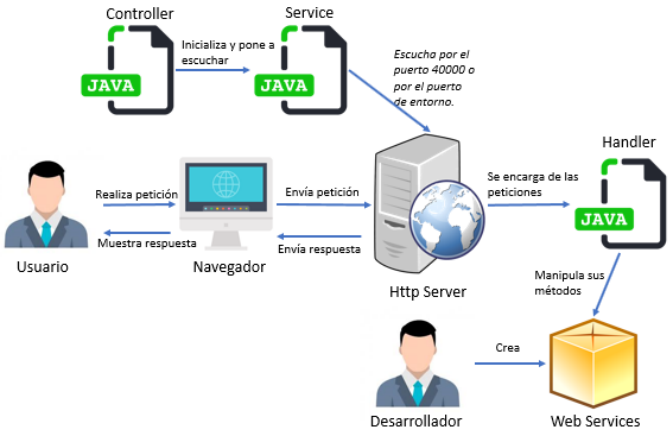


El diagrama de clases muestra la forma en que está organizado el Web Service Framework. La clase principal es el Controller, la cual inicializa el servicio que permite encontrar las clases que el desarrollador haya creado y empieza a escuchar peticiones HTTP por medio del puerto 40000 o en su defecto el puerto que haya sido tomado de la variable de entorno en donde se encuentre el servicio.

La clase Service, como ya se mencionó, es la encargada de buscar las clases que el desarrollador haya creado con métodos que estén usando la anotación @Web (esto en conjunto se denomina POJOS), de manera que se carguen todos los POJOS con los métodos correspondientes y estén listos para responder a las peticiones del usuario. Estas clases creadas por el desarrollador deben estar dentro del paquete “apps”, con el fin que la aplicación los reconozca.

La clase Service maneja una estructura de datos que permite mapear las peticiones que se realicen por el navegador a un Handler, que es una interfaz que provee los servicios de las clases creadas por el desarrollador y que tengan dicha anotación. Cuando llega una petición que se encuentre en la estructura, la respectiva implementación del handler ejecuta el método correspondiente para responder a la petición. Este resultado es pasado al Http Server el cuál mostrará toda la información en el navegador que fue pedida por el usuario. Cabe resaltar que también permite hacer peticiones a recursos estáticos. Más adelante se darán ejemplos al respecto.

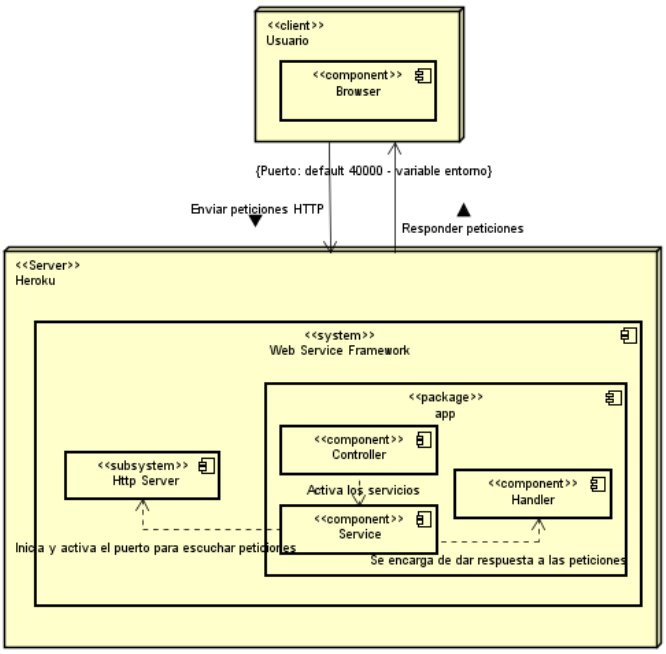
B. *Arquitectura de Software*



El diagrama de la arquitectura de software permite ver todas las interacciones a nivel global de lo que pasa cuando se realiza una petición web de acuerdo con el Web Service Framework creado.

Desde el momento en el que un usuario realiza una petición a un navegador web, este envía al servidor http dicho requerimiento. Este último ya se encontraba inicializado y escuchando por el puerto por defecto o por el tomado de la variable de entorno gracias al controlador. Las peticiones son manejadas por un manejador que usa el servidor http y que contiene todos los servicios web con los métodos que son necesarios para procesar dichas solicitudes, para finalmente enviar una respuesta de vuelta al navegador y que el usuario sea capaz de observar.

C. Arquitectura de Despliegue



Este diagrama es el que representa la arquitectura de despliegue y permite ver dónde se encuentra alojado o desplegado el proyecto, de manera que pueda mantener las peticiones desde el servidor, en este caso Heroku. Aquí se realizó una funcionalidad que permite a la aplicación tomar el puerto de la variable de entorno dentro del servidor que se está usando.

III. Web Service Framework

Este web Service Framework permite realizar peticiones a recursos estáticos como archivos HTML, imágenes PNG, JPG, y hasta archivos que permitan dar un formato visual como CSS y JS. Gracias a esto, es posible tomar una plantilla completa, por ejemplo, de Bootstrap, y usarla para algún proyecto o algo que se quiera realizar. Además, también permite al desarrollador crear sus propios métodos que respondan a peticiones mediante la URL y es posible enviarle parámetros a esas URL, de manera que las peticiones se vuelvan dinámicas.

Algunos ejemplos los podemos ver en los siguientes links:

- https://webserviceframework.herokuapp.com/switch.png
- https://webserviceframework.herokuapp.com/redesSociales.jpg
- https://webserviceframework.herokuapp.com/apps/square?number=100

Entre otras peticiones. Las dos primeras son peticiones estáticas al servidor y la última es una petición que se realiza mediante el servicio de prueba creado por un desarrollador.

IV. Conclusiones

- Se logró desplegar un servidor Http en Heroku, de manera que no sea necesario inicializarlo como localhost.
- Los POJOS son instancias de clases que no se extienden ni implementan de nada en especial y que permiten mediante el uso de anotaciones sobre los métodos que se creen, manejar peticiones dinámicas de acuerdo a la funcionalidad que se haya creado.
- Los diagramas realizados permiten ver de manera clara cómo se comportan los distintos componentes del Web Service Framework, de manera que sea posible entender su construcción, cómo se hablan entre ellos y finalmente el funcionamiento general del framework y del servidor al recibir y responder peticiones de un usuario.