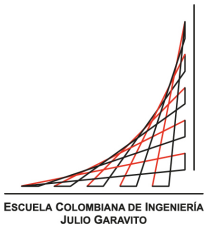

Taller Arquitecturas-IOC-Reflexion

Calderón Ortega Andrés Mateo

Arquitecturas Empresariales
Ingeniería de sistemas, Escuela Colombiana de Ingeniería Julio Garavito, Bogota, Colombia
25 de febrero de 2021

Resumen: Se realizará la explicación del funcionamiento de un framework experimental que intenta simula algunas funciones basicas del framework Spring, el desarrollo guiado a una programación orientada a objetos, su arquitectura básica, la forma en la que se decide leer los datos y archivos, y una de lectura de datos en una variable de entorno haciendo uso de JavaScript.

1. Introducción

El Problema tratado en este proyecto es el de la implementación de un framework web que sea capaz de ofrecer servicios similares al framework Spring, este nuevo 'mini' framework deberá ser capaz de resolver peticiones get por medio de funciones lambda y permitir el acceso a recursos estáticos tales como: Páginas HTML, Imágenes, y archivos de tipo JavaScript y CSS, para probar que el framework funcione de manera correcta, se decide hacer que el usuario ingrese un dato que es su nombre el cual será registrado en una variable temporal (solo durante sesión) y además podrá consultar estos datos en un ambiente intuitivo para él, la razón de este desarrollo de un mi framework es por las desventajas que trae para una organización el uso de framework 'grandes' como Spring; esto debido a que el uso de un framework como Spring, toma mucho más tiempo de aprendizaje respecto a un framework 'mini' creado desde cero, además de que estos pueden tener versiones muy inestables, causan un menor rendimiento debido a la gran cantidad de funcionalidades y extras que traen, y por supuesto quedan librerías y código inutilizable y des aprovechado a diferencia de un framework 'mini' que se implementa exclusivamente para las necesidades de la organización, además de poder tener una cercanía con un ambiente de reflexiones e inyecciones en Java.

Para este proyecto se utilizó la versión 11 de Java; es de vital importancia para el correcto funcionamiento del mismo que esta esté instalada en la máquina donde este sea ejecutado, adicionalmente fue construido haciendo uso del ambiente de desarrollo IntelliJ y para el manejo de dependencias se utilizó Maven.

2. Arquitectura

2.1. Diseño Aplicativo

Para la solución de este problema creamos la clase HttpServer que será la encargada de procesar las peticiones y manejar el puerto por el cual el servicio correrá, para esto se hace uso de la librería 'java.net' la cual nos dará la clase ServerSocket, inicialmente abriremos un puerto haciendo uso de este socket y mantendremos un ciclo infinito para escuchar y atender las peticiones de manera secuencial, una vez tenemos una clase lista para atender las peticiones debemos crear las clases que simularan al framework Spring, para esto creamos las clases 'PicoSpringBoot', 'SpringAppController' y la interfaz 'RequestMapping'.

La clase 'PicoSpringBoot' será la encargada de entender las peticiones get que serán mapeadas a partir de inyecciones; el cual contendrá el path al cual se hace referencia cuando se solicite, esta clase al momento de iniciarse cargara los componentes que son mapeados por la clase 'SpringAppController', y los almacenara para cuando sean solicitados, está una vez llega la petición get pedirá al mapa de processors el método que fue guardado para esta petición e intentara ejecutarlo, dependiendo de si la petición tiene o no parámetros realizara uso de la clase 'ParamsHandler' para entender los mismos y ejecutarlos de manera correcta esta también será la encargada de designar el puerto por el cual se va a correr y adicionalmen-

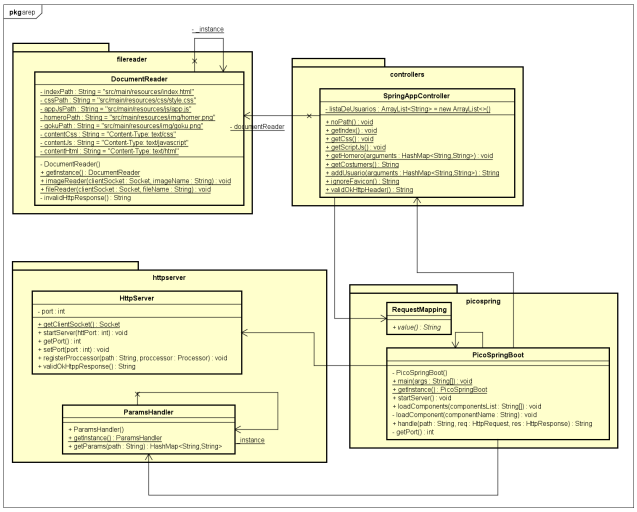

Figura 1: Diagrama De Clases

te es la que iniciara el servidor 'HttpServer', este servidor está programado para atender las peticiones que se hagan sin importar el path que se pida, si este path no esta mapeados responderá con un mensaje que indicara que el path no se cuentra disponible.

Para el manejo de peticiones y la resolución de archivos se implementó una clase que llama 'DocumentReader ' que se encargara de escribir en el socket haciendo uso del OutputStream el archivo solicitado además de colocar el encabezado pertinente para cada tipo de archivo (HTML, JavaScript, CSS y jpeg), adicionalmente esta responderá a la petición del 'favicon.ico' que se haga devolviendo solamente un encabezado correcto para tener un mejor manejo de la plataforma.

3. Despliegue y Construcción

La construcción de la aplicación se da haciendo uso de la JVM número 11 por la facilidad que nos brinda al momento de utilizar el paquete 'java.net.http' y con este las clases 'HttpRequest' y 'HttpResponse', como ambiente de desarrollo utilizamos IntelliJ IDEA en su versión Ultimate que nos permite un mejor manejo de archivos css, js y java, dándonos así un plus a la hora de desarrollar adicionalmente decidimos utilizar el manejador de dependencias Maven para tener una integración completa entre todos los componentes.

Para el despliegue de la aplicación utilizamos Heroku como servidor, debido a su gran integración

Figura 2: Pagina Principal

con git, github y java, para esto debimos modificar el archivo '.propierties' para decirle a Heroku que corra con la máquina 11 de Java.

4. Conclusiones

Se adquirieron conocimientos en el uso de Sockets en Java para la construcción de servidores Http, además se pudo reflejar y conocer el parcial funcionamiento interno de frameworks web tales como Spring, también se pudo ver las ventajas que tiene el construir un framework desde cero debido a la velocidad de procesamiento que este ofrece al momento de implementarse, pero sobre todo se vio el gran trabajo y la dificultad que conlleva la construcción de este tipo de frameworks y la complejidad de los mismos y con esto se pudo ver el porqué hoy en día es más común utilizar frameworks ya construidos como Spring, debido a la agilidad y rapidez que estos ofrecen en el desarrollo, además que estos minimizan errores y de haberlos hay una mayor facilidad para solucionarlos.

Se adquirió un mayor conocimiento en las propiedades reflexivas de Java, además de conocimientos en la construcción de framweorks IoC para POJOS, y se pudo probar el funcionamiento de la inyección de dependencias de manera manual haciendo uso de interfaces.

Referencias

[1] tiThink Technology. 2021. Framework o librerías: ventajas y desventajas. [online] Available at: https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/ [Accessed 13 February 2021].].

[2] Journals.ecs.soton.ac.uk. 2021. Reading from and Writing to a Socket. [online] Available at: http://journals.ecs.soton.ac.uk/java/tutorial/networking/sockets/readingWriting.html [Accessed 17 February 2021].

[3] Usar promesas - JavaScript — MDN. (2021). Retrieved 25 February 2021, from https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Usar_promesas

[4] Daniel Benavides. 2021. Clase de laboratorio. [12 Febrero 2021].