Spring framework 5 - Swagger + Cache

Configuración

Para configurar swagger agregaremos las siguientes dependencias a nuestra aplicación:

Brinda soporte para utilizar swagger en spring:

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
```

Interfaz de usuario para visualizar la documentación de swagger:

```
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger-ui</artifactId>
   <version>2.9.2</version>
</dependency>
```

Configuración de beans

Para configurar swagger se debe configurar un Docket como se muestra a continuación:

```
@Configuration
@EnableSwagger2
publicclassSwaggerConfig {
 @Bean
 publicDocket api() {
  return newDocket(DocumentationType.SWAGGER_2)
    .select().apis(RequestHandlerSelectors.any())
    .paths(PathSelectors.any()).build();
 }
}
```

Visualización de la documentación

Para consultar la documentación debes acceder a la siguiente url:

http://localhost:8080/swagger-ui.html

Si definiste tu propio context path la url sería:

http://localhost:8080/context-path/swagger-ui.html

Selectors

Es posible seleccionar los recursos que deseas exponer a través de 2 selectores:

- RequestHandlerSelectors
- PathSelectors

A continuación dos ejemplos:

```
newDocket(DocumentationType.SWAGGER_2).select()
   .apis(RequestHandlerSelectors.
    basePackage("com.devs4j.users.controllers"))
   .paths(PathSelectors.ant("/*"))
   .build();
```

Descripciones propias

Puedes definir una descripción y un tipo de dato de retorno a través de las anotaciones:

```
@ApiOperation(value ="View a list of users", response = Page.class)

@ApiResponses(value = {@ApiResponse(code = 200, message ="Successfully retrieved list") })
```

Configuración de cache

Para configurar Spring cache debes agregar la siguiente dependencia:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-cache</artifactId>
 </dependency>
```

Configuración de los beans

Una vez agregada la dependencia se debe configurar del siguiente modo:

```
@EnableCaching
@Configuration
publicclassCacheConfig {

 @Bean
 publicCacheManager getManager() {
      return new ConcurrentMapCacheManager("codes");
 }
}
```

Uso de cache

Una vez configurado se deben agregar los objetos al cache, para esto utilizaremos la siguiente anotación como se muestra a continuación:

```
@Cacheable("codes")
 publicList<String> getCp() {
      ./Code
}
```

Remover valores con @CacheEvict

Es posible remover valores de un cache a través de la anotación @CacheEvict:

```
@CacheEvict("users")
 publicUser getUserById(IntegeruserId){
  .....
}
```

Cache en memoria

Cuando se utiza **ConcurrentMapCacheManager** el cache se realiza en la máquina que está ejecutando la aplicación.

Si se colocarán en cache muchos datos esto puede causar problemas en la aplicación.

Uso de un sistema de cache externo

Para el caso de que se deseen tener muchos datos en cache, es posible almacenarlos en un sistema de cache externo, en este ejemplo se utilizará redis, puedes descargarlo en la siguiente dirección:

https://redis.io/download

Iniciar redis

Para iniciar redis debes seguir los siguientes pasos:

- Descarga redis
- Descomprimelo
- Ejecuta el comando make
- Ejecuta el comando src/redis-server

< DEVS4J >
Hecho por Devs para Devs

Configuración de redison

Para configurar la integración con redis agregar la siguiente dependencia:

```
<dependency>
 <groupId>org.redisson</groupId>
 <artifactId>redisson</artifactId>
 <version>3.11.5</version>
</dependency>
```

Agregar el bean de redison al contexto

Agregaremos al contexto de spirng :

```
 @Bean(destroyMethod ="shutdown")
publicRedissonClient redisson() {
      Configconfig=newConfig();
      config.useSingleServer().
      setAddress("redis://127.0.0.1:6379");
 returnRedisson.create(config);
}
```

Modificaremos el Cache Manager

El siguiente paso es indicar a spring que utilizará redis en lugar del ConcurrentMapCache, como se muestra a acontinuación:

```
@Bean
 publicCacheManager
cacheManager(RedissonClientredissonClient)
{
  Map<String,
CacheConfig>config=newHashMap<>();
  config.put("testMap",newCacheConfig());
  returnnewRedissonSpringCacheManager
(redissonClient);
 }
```

Nota importante

Los objetos que se escriban en el cache deben implementar la interfaz Serializable

Revisar información en Redis

Para consultar la información almacenada en redis puedes utilizar el cliente que provee, ejecutando el comando en la carpeta raíz de redis:

-> src/redis-cli

Comandos útiles:

DEL key - Borra una llave

EXISTS key - Determina si la llave existe

HGETALL key - Obtiene el valor de un map