CREACIÓN

DE UN

MICROSERVICIO

Autor: Ildefonso Albares García
Fecha Actualización 29-07-2024

Índice

1. Partes de un Microservicio

Un microservicio lo vamos a definir como una caja a la que le llega una petición de forma de URL, normalmente, y con esa información el microservicio le devuelve la información solicitada.

1.1. Modelo de datos

En esta parte del código se debe definir los datos que serán los representativos dentro de la base de datos (BBDD) a la que acceda el microservicio. Este archivo o archivos no sólo definen el modelo de acceso a la Base de datos, también pueden contener otro tipo de programación como definir un mensaje, lo veremos en un ejemplo más adelante.

1.2. Capa de acceso a datos

La capa de acceso a los datos es el archivo encargado de hacer toda la lógica de forma abstraída para devolvernos los datos solicitados de la BBDD. Esto se consigue con las dependencias JPA y de esta manera evitamos el realizar de forma manual la conexión a la base de datos. Lo rápido es usar la interfaz JpaRepository de Spring Data JPA, en el ejemplo usamos CrudRepository.

La diferencia es que JpaRepository es una extensión específica del Repositorio **JPA (Java Persistence API)** . Contiene la API completa de **CrudRepository** y **PagingAndSortingRepository** . Por lo tanto, contiene API para operaciones CRUD básicas y también API para paginación y clasificación. Dependiendo de lo que necesitemos implementamos una u otra.

1.3. Capa lógica

En esta parte se implementa toda la lógica posible para reducir lo máximo el código en la parte de lógica. Aquí se realizan definiciones de métodos, validaciones o cualquier otro cálculo complejo. Esta parte se implementa con dos archivos, uno es una interfaz y el segundo es un tipo “Class” identificado con @Service. En el primero definimos que métodos tendrá el @Service y en el propio archivo @Service desarrollamos los métodos anteriores, básicamente lo que tienen que hacer.

1.4. El uso de la lógica

Aquí se definen las llamadas de la URL´s y lo que debe devolver en dicha solicitudes. Aquí se integra básicamente la lógica del controlador del @RestController. En este caso hay que definir que hace cada solicitud (GET, PUT, POST y DELETE)

2. Nomenclatura de Microservicio

- @Controller: que registrará el controlador para Spring MVC
- @RequestMapping: anotación que se encarga de relacionar un método con una petición http
- @RequestBody:
- @SpringBootApplication:
- @RestController:
- @Autowired:
- @GetMapping(“/request”):
- @PostMapping(“/request”):
- @PutMapping:

```
@PutMapping("/users/{id}")
public void updateUser(@PathVariable("id") String id, @RequestBody User user) {
        // Update the user details here
}
```

- @DeleteMapping(“/request”):

```java
@DeleteMapping("/users/{id}")
public void deleteUser(@PathVariable("id") String id) {
    // Delete the user in this method with the id.
}
```

- @PathVariable:
- @Entity:
- @Table (name=“marcas”):
- @Id:
- @GeneratedValue(strategy = GenerationType.IDENTITY):
- @Service:
- @Transactional(readOnly = true) :

3. Ejemplo de Microservicio

IProductoServicio.java ProductoServicioImpl.java SprintbootServicioSaludoApplication.java

```java
package com.example.producto.saludo;

import org.springframework.boot.SpringApplication;

@EnableEurekaClient
@SpringBootApplication
public class SprintbootServicioSaludoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SprintbootServicioSaludoApplication.class, args);
    }

}
```

3.1. Modelo de datos

```java
package com.example.producto.saludo.mensaje;

import java.io.Serializable;

@Entity
@Table (name="marcas")
public class ProductoMarca implements Serializable{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nombre;
    private Integer precio;
    private String articulo;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public Integer getPrecio() {
        return precio;
    }
    public void setPrecio(Integer precio) {
        this.precio = precio;
    }
    public String getArticulo() {
        return articulo;
    }
    public void setArticulo(String articulo) {
        this.articulo = articulo;
    }

    private static final long serialVersionUID = 704674473724120359L;

}
```

```java
package com.example.producto.saludo.mensaje;

import java.io.Serializable;

public class ProductoMensaje implements Serializable{

    private String mensaje;

    public String getMensaje() {
        return mensaje;
    }
    public void setMensaje(String mensaje) {
        this.mensaje = mensaje;
    }

    private static final long serialVersionUID = -6387700685089305723L;

}
```

3.2. Capa de acceso a datos

```java
package com.example.producto.saludo.dao;

import org.springframework.data.repository.CrudRepository;

public interface ProductoDao extends CrudRepository<ProductoMarca, String>{

}
```

3.3. Capa lógica

```java
package com.example.producto.saludo.servicio;

import java.util.List;

public interface IProductoServicio {

    public ProductoMensaje mensajeback();
    public List<ProductoMarca> findAll();
}
```

```java
package com.example.producto.saludo.servicio;

import java.util.List;



@Service
public class ProductoServicioImpl implements IProductoServicio{

    @Autowired
    private ProductoDao productodao;

    @Override
    public ProductoMensaje mensajeback() {
        ProductoMensaje mensaje=new ProductoMensaje();
        mensaje.setMensaje("Hola Mundo");

        return mensaje;
    }

    @Override
    @Transactional(readOnly = true)
    public List<ProductoMarca> findAll() {
        return (List<ProductoMarca>) productodao.findAll();
    }

}

```

3.4. El uso de la lógica

```java
package com.example.producto.saludo.controller;

import java.util.List;

@RestController
public class SaludoController {

    @Autowired
    private IProductoServicio productoService;

    @GetMapping("/saludos")
    public ProductoMensaje mostrarmensaje() {
        return productoService.mensajeback();
      }

    @GetMapping("/listar")
    public List<ProductoMarca> seleccion(){
        return productoService.findAll();
    }

}
```