

Controlador REST con Spring Boot



Creación del Controlador REST con Spring Boot

En una arquitectura basada en **Spring Boot**, las aplicaciones suelen dividirse en capas para mejorar la organización y la mantenibilidad.

Las **tres capas principales** son:

1. **Capa de Datos (Repositorio)** → Se comunica con la base de datos.
2. **Capa de Servicio** → Contiene la lógica de negocio.
3. **Capa de Presentación (Controlador REST)** → Expone los endpoints HTTP para interactuar con la aplicación.

◆ Paso 1: Crear la Clase Controlador REST

❖ **Objetivo:** Crear un controlador que exponga los datos de productos mediante un endpoint REST.

❖ **Archivo:** ProductoControlador.java

```

package gm.inventarios.controlador;

import gm.inventarios.modelo.Producto;
import gm.inventarios.servicio.ProductoServicio;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;

@RestController
//http://localhost:8080/inventario-app
@RequestMapping("inventario-app")
@CrossOrigin(value = "http://localhost:4200")
public class ProductoControlador {

    private static final Logger logger =
        LoggerFactory.getLogger(ProductoControlador.class);

    @Autowired
    private ProductoServicio productoServicio;

    //http://localhost:8080/inventario-app/productos
    @GetMapping("/productos")
    public List<Producto> obtenerProductos(){
        List<Producto> productos = this.productoServicio.listarProductos();
        logger.info("Productos obtenidos:");
        productos.forEach((producto) -> logger.info(producto.toString()));
        return productos;
    }
}

```

Explicación:

- `@RestController`: Indica que esta clase manejará peticiones HTTP y retornará respuestas en formato JSON.
- `@RequestMapping("inventario-app")`: Define la **ruta base** del controlador.
- `@CrossOrigin("http://localhost:4200")`: Permite que aplicaciones frontend como Angular puedan acceder a este endpoint.
- `@Autowired`: Inyecta el servicio `ProductoServicio` para obtener los datos.
- `@GetMapping("/productos")`: Expone el endpoint `http://localhost:8080/inventario-app/productos`, que devuelve la lista de productos.
- **Logger**: Imprime los productos en la consola para verificar que los datos se están recuperando correctamente.

Tu proyecto debe tener la siguiente estructura:

```

gm.inventarios
|--- InventariosApplication.java  # Punto de entrada de la app
|--- modelo/

```

```

    └── Producto.java           # Entidad JPA
  └── repositorio/
      ├── ProductoRepositorio.java # Capa de datos
  └── servicio/
      ├── IProductoServicio.java   # Interfaz de servicio
      └── ProductoServicio.java     # Implementación del servicio
  └── controlador/
      └── ProductoControlador.java # Controlador REST (Capa presentación)

```

◆ Paso 4: Ejecutar la Aplicación

❖ **Objetivo:** Iniciar el servidor y verificar que el endpoint responde correctamente.

❖ **Archivo:** InventariosApplication.java

```

package gm.inventarios;

import gm.inventarios.modelo.Producto;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

```

Explicación:

- @SpringBootApplication indica que esta es la clase principal de la aplicación.
- SpringApplication.run(InventariosApplication.class, args); inicia el servidor en http://localhost:8080.

◆ Paso 5: Verificar el Resultado en el Navegador

❖ **Objetivo:** Acceder al endpoint http://localhost:8080/inventario-app/productos y ver los datos en formato JSON.

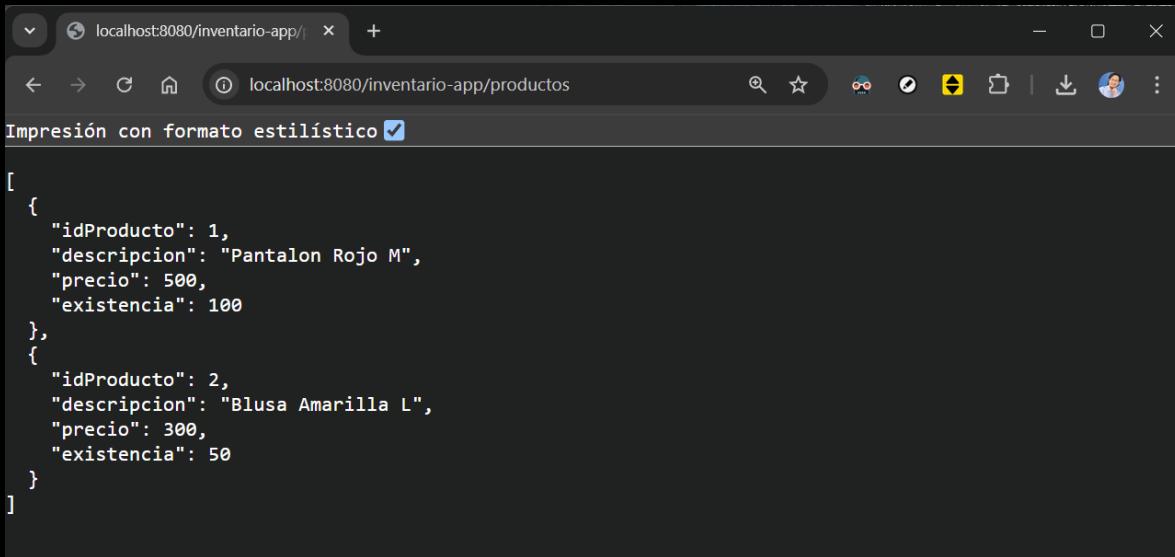
1 Abrir el navegador y acceder a:

http://localhost:8080/inventario-app/productos

2 Ver el resultado JSON, que mostrará algo similar a esto:

[

```
{  
    "idProducto": 1,  
    "descripcion": "Pantalón Rojo M",  
    "precio": 500,  
    "existencia": 100  
},  
{  
    "idProducto": 2,  
    "descripcion": "Blusa Amarilla L",  
    "precio": 300,  
    "existencia": 50  
}  
]
```



3 Revisar la consola, donde el logger imprimirá:

```
Productos obtenidos:  
Producto{idProducto=1, descripcion='Pantalón Rojo M', precio=500,  
existencia=100}  
Producto{idProducto=2, descripcion='Blusa Amarilla L', precio=300,  
existencia=50}
```

📌 Conclusión

Separación de responsabilidades:

- El **Controlador** maneja las solicitudes HTTP.
- El **Servicio** encapsula la lógica de negocio.
- El **Repositorio** accede a la base de datos.

Ventajas de esta estructura:

- Código limpio y organizado.
- Fácil mantenimiento y escalabilidad.
- Reutilización del servicio en otros controladores.

Con esta guía, puedes crear endpoints REST en Spring Boot y conectar cada capa de la aplicación correctamente. 

Saludos!

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](#)