DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Back End I

Ejercitación Patrón MVC (API) - Práctica con Docente

Objetivo

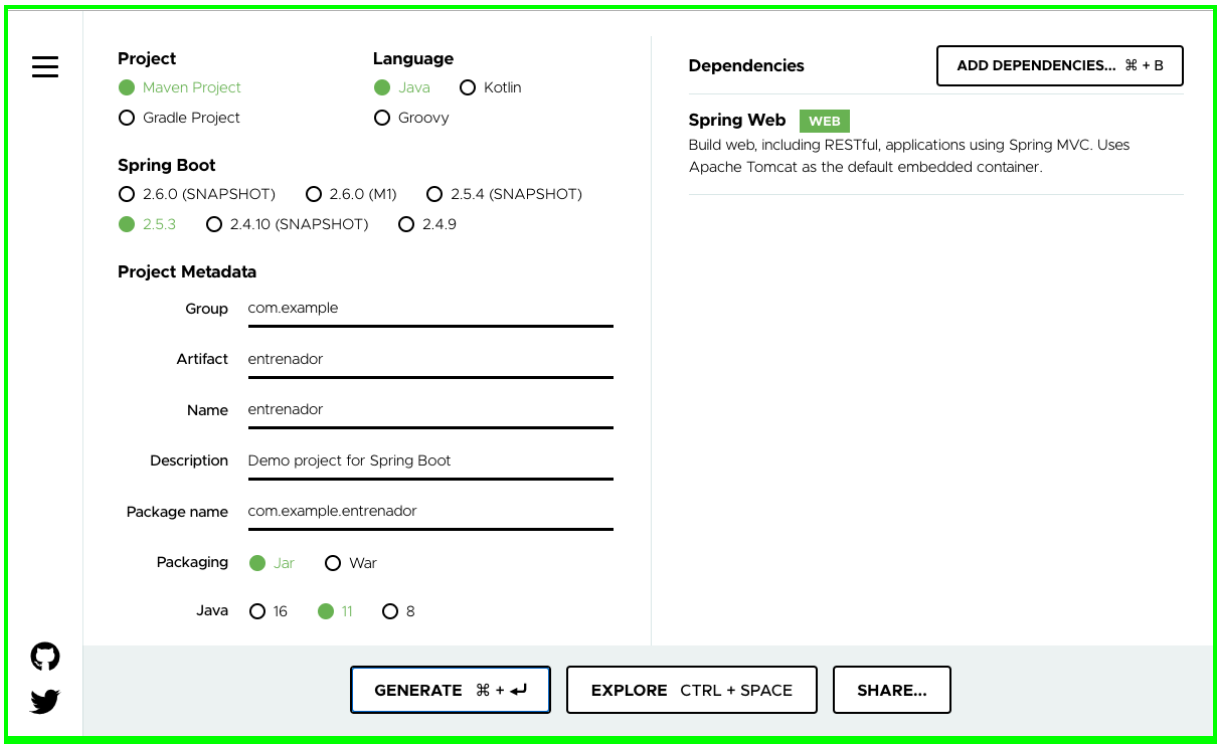
Continuando con las ejercitaciones realizadas en la clase asincrónica, vamos a crear un proyecto llamado Entrenador en Spring MVC siguiendo las instrucciones.

- Ejercicio individual
- Complejidad: baja 🔥

Instrucciones

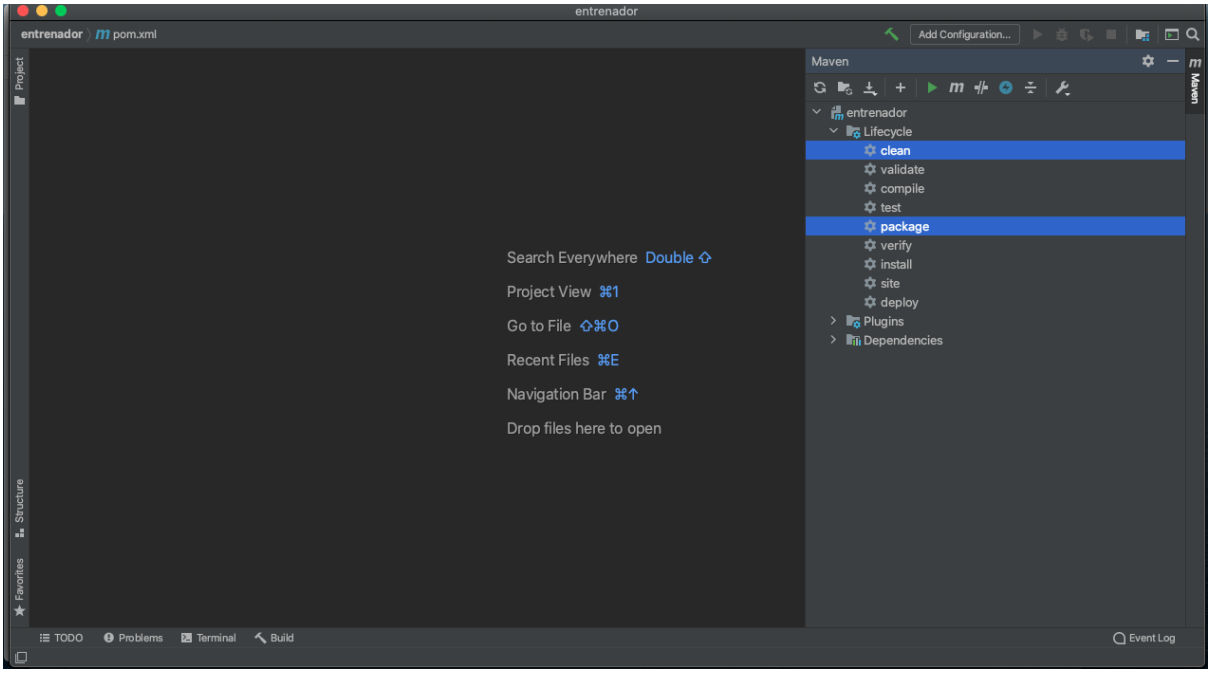
1- Crear un proyecto desde https://start.spring.io.

> ✅ Recordá poner el nombre del proyecto y agregar en dependencias Spring Web.



2- Generar el proyecto y descomprimir el archivo .zip para abrirlo en Intellij IDEA.
En Intellij IDEA elegimos “New -> existing source”.

3- En la solapa de Maven hacer clean package y presionar “Play” en la solapa de arriba de Maven.



4- Crear un **modelo**, es decir, una clase de negocio Entrenador.

```
package com.example.entrenador.domain;

public class Entrenador {

    private String nombre;

    public Entrenador(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
```

```
        this.nombre = nombre;
    }
}
```

5- Crear un paquete service dentro del proyecto. Agregar la interface **EntrenadorService** y su clase que la implementa: **EntrenadorServiceImpl**.

```
package com.example.entrenador.service;

import com.example.entrenador.domain.Odontologo;

import java.util.List;

public interface EntrenadorService {

    List<Entrenador> listaEntrenador();
}
```

La annotation **@Service** le dice a Spring que es un servicio.
Vemos cómo en listaEntrenador estamos agregando de manera manual los datos. En una aplicación debemos ir a nuestra capa de DAO para devolverlo desde una base de datos. A continuación un caso cuando no tenemos base de datos::

```
package com.example.entrenador.service;

import com.example.entrenador.domain.Entrenador;

import java.util.Arrays;
import java.util.List;
@Service
```

```
public class EntrenadorServiceImpl implements EntrenadorService{
    @Override
    public List<Entrenador> listaEntrenador() {
        return Arrays.asList(new Entrenador("Pibe"), new New
Entrenador("Roman"));
    }
}
```

6- Crear un controller en el paquete Controller.

```
import com.example.entrenador.domain.Entrenador;
import com.example.entrenador.service.EntrenadorService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.Controller;

import java.util.List;

@RestController
@RequestMapping("/entrenador")
public class EntrenadorController {

    private final EntrenadorService entrenadorService;

    @Autowired
    public EntrenadorController(EntrenadorService entrenadorService) {
        this.entrenadorService = entrenadorService;
    }
```

```
    @GetMapping
    public List<Entrenador> getEntrenador() {
        return entrenadorService.listaEntrenador();
    }
}
```

Como vemos, la clase Controller hace referencia a service (el modelo) y después automáticamente lo transforma en JSON, que sería nuestra vista. Esto pasa dentro de **@GetMapping** annotation. Dentro del Controller debemos agregar **@Controller** para decirle a Spring que este es nuestro controller y **@RequestMapping** para agregar nuestra URL, en este caso /entrenador.

La annotation **@Autowired** la vamos a ver en las próximas clases, pero podemos adelantar que es la conexión entre el modelo y el controller.

Ahora, si corremos nuestro servidor desde el main de la clase EntrenadorApplication y vamos al navegador (por ejemplo Chrome) y ponemos **http://localhost:8080/entrenador**, obtenemos nuestra vista que en este caso es la respuesta a una API Rest: **[{"nombre":"Pibe"},{"nombre":"Roman"}]**.