



Certified Tech Developer

The Ultimate Degree

¡Vamos a practicar!

Vamos a poner en práctica lo visto anteriormente. Ahora es tu turno y te proponemos crear un proyecto odontólogo en Spring Boot, siguiendo las instrucciones.

Instrucciones

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

No te olvides de ingresar el nombre.

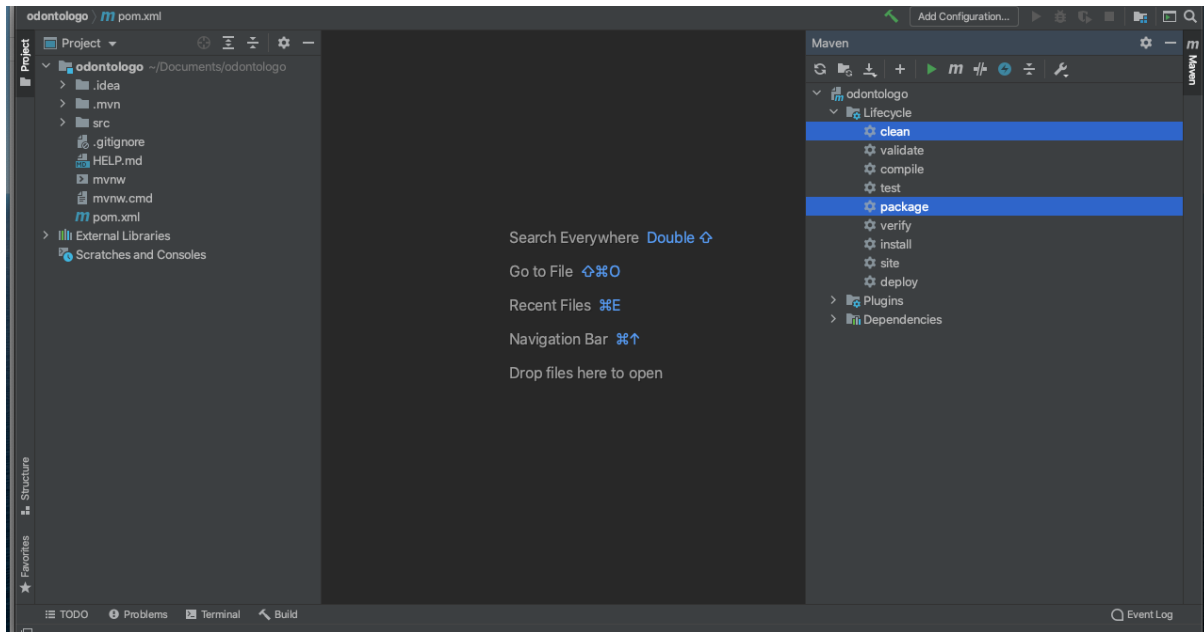
The screenshot shows the Spring Boot Start web application. The browser address bar displays 'start.spring.io'. The page is divided into several sections:

- Project:** Includes radio buttons for 'Maven Project' (selected) and 'Gradle Project'.
- Language:** Includes radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'.
- Spring Boot:** Includes radio buttons for versions: '2.6.0 (SNAPSHOT)', '2.5.4 (SNAPSHOT)', '2.5.3' (selected), '2.4.10 (SNAPSHOT)', and '2.4.9'.
- Project Metadata:** A form with fields for:
 - Group: 'com.example'
 - Artifact: 'odontologo'
 - Name: 'odontologo'
 - Description: 'Demo project for Spring Boot'
 - Package name: 'com.example.odontologo'
- Packaging:** Includes radio buttons for 'Jar' (selected) and 'War'.
- Dependencies:** A section with a 'Spring Web' dependency (marked 'WEB') and a description: 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.' There is an 'ADD DEPENDENCIES...' button.

At the bottom, there are three buttons: 'GENERATE' (with a keyboard shortcut icon), 'EXPLORE' (with 'CTRL + SPACE' shortcut), and 'SHARE...'. On the right side, there are icons for settings and a dark mode toggle.



2. Hacé clic en "generate", que generará que obtengas un zip. Descomprimilo y abrílo en IntelliJ.
3. Allí, elegí new -> existing source. Después debés ir a la solapa de maven y hacer clic en "package".



4. Presionar "play" en la solapa de arriba de Maven.
5. Primero vamos a crear el modelo: para ello, necesitás crear un paquete service dentro del proyecto y agregar la interface OdontologoService y OdontologoServiceImpl. Además, vas a necesitar un objeto odontólogo en el paquete domain.

```
package com.example.odontologo.domain;

public class Odontologo {

    private String nombre;

    public Odontologo(String nombre) {
        this.nombre = nombre;
    }
}
```



```
}

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

```
package com.example.odontologo.service;

import com.example.odontologo.domain.Odontologo;

import java.util.List;

public interface OdontologoService {

    List<Odontologo> listaOdontologos();
}

package com.example.odontologo.service;

import com.example.odontologo.domain.Odontologo;

import java.util.Arrays;
import java.util.List;

@Service
```

```
public class OdontologoServiceImpl implements OdontologoService{  
    @Override  
    public List<Odontologo> listaOdontologos() {  
        return Arrays.asList(new Odontologo("Javier"), new  
Odontologo("Ramon"));  
    }  
}
```

La anotación @Service le dice a Spring que es un servicio. Allí, vemos como listaOdontologos está hardcodeando. Esto sucede agregando de manera manual los datos.

En una aplicación tenés que ir a la capa de DAO para devolverlo desde una base de datos, por ejemplo.

6. Luego, tenés que crear un paquete controller en el proyecto y agregar una clase OdontologoController.

```
import com.example.odontologo.domain.Odontologo;  
import com.example.odontologo.service.OdontologoService;  
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.RestController;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("odontologos")  
public class OdontologoController {
```

```
private final OdontologoService odontologoService;

@Autowired
public OdontologoController(OdontologoService odontologoService) {
    this.odontologoService = odontologoService;
}

@GetMapping
public List<Odontologo> getOdontologos() {
    return odontologoService.listaOdontologos();
}
}
```

Como podés observar, 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.

7. Dentro del Controller debemos agregar @RestController para decirle a Spring que este es nuestro controller y @RequestMapping para agregar nuestra URL, en este caso/odontólogo.

La annotation @Autowired la vamos a ver en los próximos capítulos, pero podemos adelantarte que se trata de la conexión entre el modelo y el controller.

8. Ahora sí, corré tu servidor desde el main de la clase: OdontologoApplication y desde el browser (por ejemplo Chrome) y poné <http://localhost:8080/odontologos>
9. Finalmente, obtendrás tu vista: [{"nombre":"Javier"}, {"nombre":"Ramon"}]

¡Éxitos!