



Certified Tech Developer

The Ultimate Degree

Vamos praticar!

Vamos colocar em prática o que aprendemos. Agora é a sua vez! Sugerimos que crie um projeto chamado de “Dentista” no Spring Boot, seguindo as instruções.

Instruções

1. Criar um projeto a partir do site <https://start.spring.io>

Lembre-se de informar os campos “Artefact” e “Name”.

The screenshot shows the Spring Boot start.spring.io website. The interface is divided into several sections:

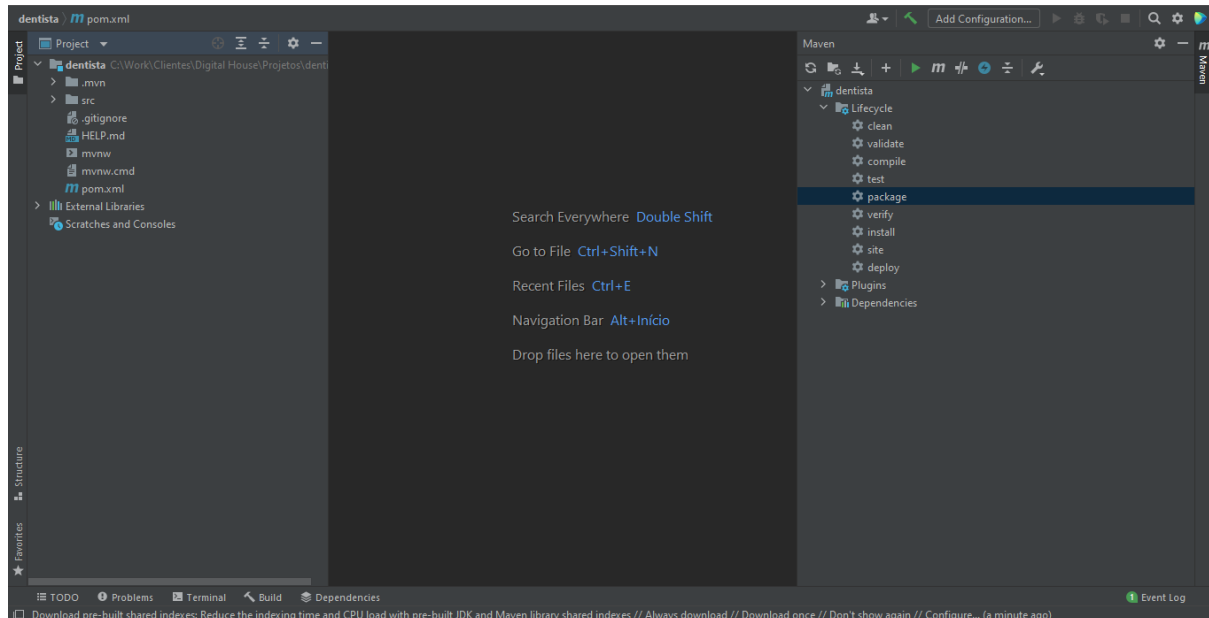
- Project:** Includes radio buttons for **Maven Project** (selected) and **Gradle Project**.
- Language:** Includes radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Includes radio buttons for versions: **2.6.0 (SNAPSHOT)**, **2.5.4 (SNAPSHOT)**, **2.5.3** (selected), **2.4.10 (SNAPSHOT)**, and **2.4.9**.
- Project Metadata:** Includes input 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**.
- Java:** Includes radio buttons for versions: **16**, **11** (selected), and **8**.
- Dependencies:** Includes a button **ADD DEPENDENCIES... ⌘ + B** and a section for **Spring Web** (selected) with a description: "Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container."

At the bottom, there are three buttons: **GENERATE ⌘ + ↵**, **EXPLORE CTRL + SPACE**, and **SHARE...**

2. Clique em “generate”, para gerar o projeto e realizar o download de um arquivo .zip.

3. Em sequência, descompactar o arquivo a abrir o projeto no IntelliJ.

No IntelliJ, acesse o menu “File” > “New” > “Project from Existing Sources”, depois acessar a aba Maven e clicar em “package”.



4. Pressione “play” na guia superior do Maven.
5. Primeiro vamos criar a model: para isso, você precisa criar um pacote de serviços dentro do projeto e adicionar a interface DentistaService e DentistaServiceImpl. Além disso, você precisará de um objeto Dentista no pacote domain.

```
package com.example.dentista.domain;

public class Dentista{

    private String nome;

    public Dentista(String nome) {
        this.nome = nome;
    }
}
```

```

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}

```

```

package com.example.dentista.service;
import com.example.dentista.domain.Dentista;
import java.util.List;

public interface DentistaService {
    List<Dentista> listaDentistas();
}

package com.example.dentista.service;
import com.example.dentista.domain.Dentista;
import java.util.Arrays;
import java.util.List;

@Service
public class DentistaServiceImpl implements DentistaService{
    @Override
    public List<Dentista> listaDentistas() {
        return Arrays.asList(new Dentista("Maria"), new
Dentista("João"));
    }
}

```

A anotação `@Service` sinaliza ao Spring que é um serviço. Também vemos como a lista de dentistas está codificada. Isso acontece adicionando manualmente os dados.

Em uma aplicação, você deve ir para a camada do DAO para obter os dados de um banco, por exemplo.

6. Então, você deve criar um pacote controller no projeto e adicionar uma classe `DentistaController`.

```
import com.example.dentista.domain.Dentista;
import com.example.dentista.service.DentistaService;
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("dentistas")
public class DentistaController {

    private final DentistaService dentistaService;

    @Autowired
    public DentistaController(DentistaService dentistaService) {
        this.dentistaService = dentistaService;
    }

    @GetMapping
    public List<Denstista> getDentistas() {
        return dentistaService.listaDentistas();
    }
}
```

```
}
```

Como você pode ver, a classe do Controller faz referência ao serviço (Model) e então automaticamente o transforma em JSON, que seria a nossa View. Isso acontece dentro da anotação @GetMapping.

7. Dentro do Controller, devemos adicionar @RestController para informar ao Spring que este é o nosso controlador e @RequestMapping para adicionar nossa URL, neste caso "/dentistas".

Veremos a anotação @Autowired nas próximas aulas, mas podemos mencionar que trata-se da conexão entre a model e o controller.

8. Agora sim, execute seu servidor a partir da main da classe: DentistaApplication e no navegador (por exemplo, o Chrome) acessar o endereço <http://localhost:8080/dentistas>
9. Finalmente, visualizar a view: [{"nome":"Maria"}, {"nome":"João"}]

Sucesso!