

Programação WEB I

Prof. Marcelo Estruc

C:\workspaceSprigBootProg\ProjetoAula

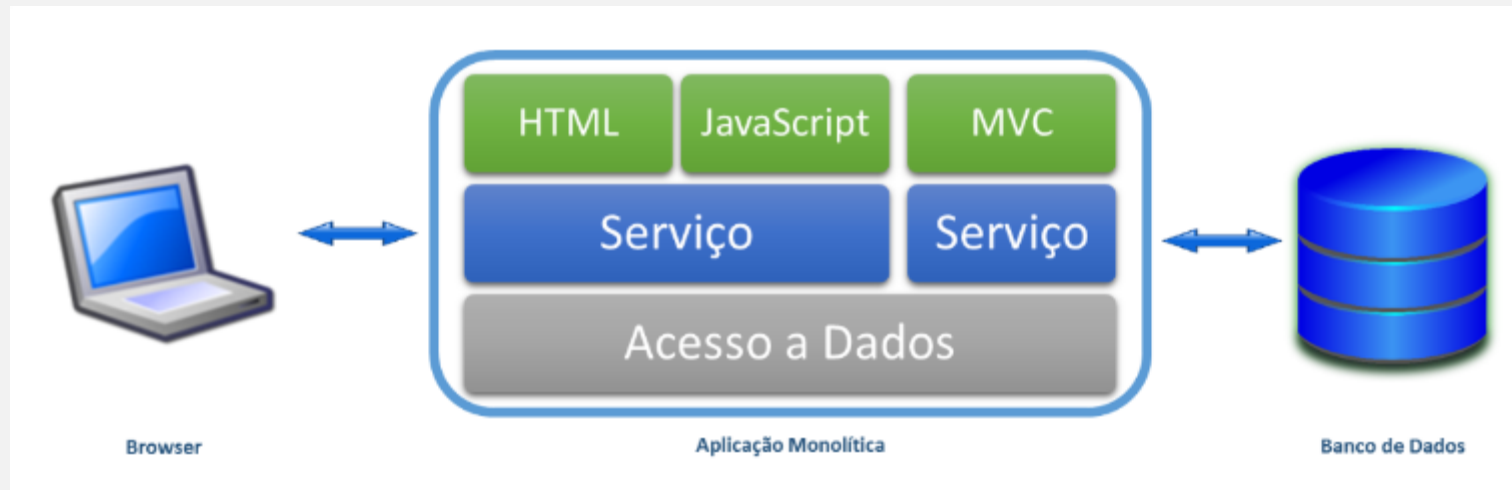
Spring Boot	Conhecer o Spring Boot como framework de desenvolvimento de aplicações web.
Instalação e configuração	Executar a instalação e configuração do framework Spring Boot.
Maven	Compreenda o uso do pom.xml nas aplicações
Banco de Dados em Memória	Como utilizar um BD para teste e desenvolvimento
O modelo MVC	Conhecer o conceito de MVC e preparar ambiente de trabalho para o desenvolvimento de um projeto MVC com Spring Boot.
Exemplo de projeto	Desenvolver um projeto exemplo com o Spring Boot.
Camadas da aplicação	Compreendendo as camadas utilizadas em uma aplicação spring boot
Repository e Entity	Entender o conceito de repository na arquitetura MVC e implementá-lo em um projeto Spring Boot.
Service	Entender o conceito de service na arquitetura MVC e implementá-lo em um projeto Spring Boot.
Controller	Entender o conceito de controller na arquitetura MVC e implementá-lo em um projeto Spring Boot.

Arquitetura de aplicações

Prof. Marcelo Estruc

Aplicações monolíticas

- O são projetadas para a criação de um único executável monolítico, no qual toda a modularização utilizada é executada em uma mesma máquina. Assim, os módulos compartilham recursos de processamento, memória, bancos de dados e arquivos.
- Uma arquitetura monolítica típica de um sistema complexo pode ser representada pela figura abaixo, na qual todas as funções do negócio estão implementadas em um único processo.



Desafios da arquitetura monolítica

- Ao longo do tempo o sistema vai crescendo, se tornando mais complexo e consumindo cada vez mais recursos, o que acaba gerando também alguns desafios substanciais para a manutenção desse tipo de arquitetura. São eles:
 - **Aumento da complexidade e tamanho ao longo do tempo**
 - Quando algum dev trabalha nesse sistema pode ter muita dificuldade de fazer simples configuração.
 - **Alta dependência de componentes de código**
 - Alterar uma funcionalidade pode impactar todo o sistema
 - **Escalabilidade do sistema é limitada**
 - Acrescentar algo que demanda processamento pode impactar todo o sistema
 - **Falta de flexibilidade**
 - Amarração de tecnologias que o sistema foi desenvolvido e não pode ser alterada
 - **Dificuldade para colocar alterações em produção**
 - Quando é feita uma alteração todo o sistema deve parar para ser colocado em produção

Vantagens arquitetura monolítica

- Sistemas monólito podem ser muito bem vistos quando :
 - **Custo**
 - Em uma única máquina podemos ter nosso sistema
 - **Escopo bem definido**
 - Sistemas com escopo bem definido e sem perspectiva de crescimento
 - **Tecnologia**
 - Definição da tecnologia que não será alterada e atende o escopo do projeto

Servlet

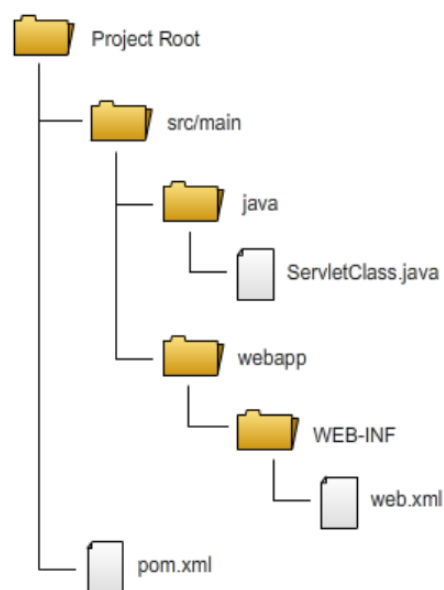
Prof. Marcelo Estruc

O que são servlets

Os servlets são os blocos de construção de quase todos os aplicativos da Web criados em Java. Eles fornecem a funcionalidade principal para aceitar solicitações HTTP e retornar respostas HTTP ao usuário. Mesmo se você usar JSP para construir suas páginas da web, os arquivos JSP são eventualmente compilados em Servlets pelo servidor web, como Glassfish ou Tomcat.

Servlets são essenciais para o seu web container fazer seu trabalho, atendendo requisições GET, POST, HEAD, PUT, DELETE, OPTIONS e TRACE e retornando uma resposta para web clients. Você provavelmente está familiarizado com solicitações GET e POST

Estrutura de um projeto



Exemplo de um servlet

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SimpleServlet extends HttpServlet {

    private static final long serialVersionUID = -4751096228274971485L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.getWriter().println("Hello World!");
    }

    @Override
    public void init() throws ServletException {
        System.out.println("Servlet " + this.getServletName() + " has started");
    }

    @Override
    public void destroy() {
        System.out.println("Servlet " + this.getServletName() + " has stopped");
    }
}
```

web.xml Deployment Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd"
  version="3.1">

  <display-name>Simple Servlet Application</display-name>

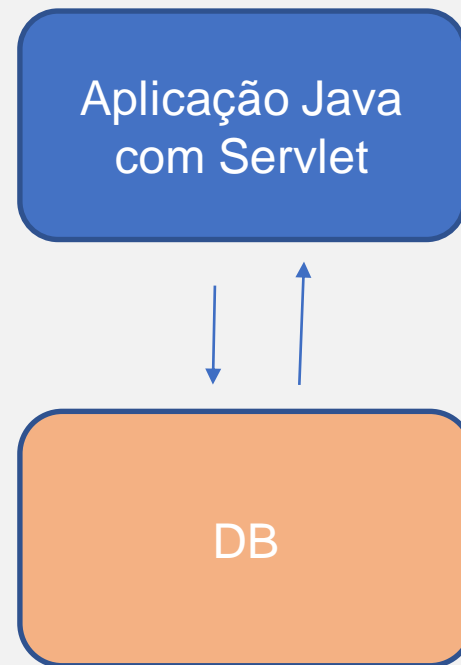
  <servlet>
    <servlet-name>simpleServlet</servlet-name>
    <servlet-class>net.javatutorial.tutorials.SimpleServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>simpleServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>

</web-app>
```

- Deve ser gerado um arquivo war
- Realizar o deploy em um servidor de aplicação
- Chamar sua aplicação.

<http://localhost:8080/SimpleServlet/hello>



Spring Boot

Prof. Marcelo Estruc

Spring Boot

- O Spring Boot é um **framework Java *open source*** que tem como objetivo facilitar o processo de desenvolvimento de aplicações Java. Ele traz mais agilidade para o processo, uma vez que desenvolvedores conseguem **reduzir o tempo** gasto com as configurações iniciais.
- Com o Spring Boot conseguimos abstrair e facilitar a configuração de **servidores, gerenciamento de dependências, configurações de bibliotecas, métricas e *health checks***.

Spring Boot

- O Spring Boot utiliza um conceito conhecido como ***Convention over Configuration*** (convenção ao invés de configuração).
- Isso significa que é uma ferramenta que decide para você a melhor forma de se fazer algo. É o que chamamos de ferramenta opinativa, ela toma as decisões no nosso lugar baseado em convenções, aplicando configurações padrões e facilitando o trabalho.
- O Spring Boot é composto por vários módulos que ajudam em nossos projetos, dentre os quais destacamos:

Spring Boot Starters

- Os **starters** são dependências que agrupam outras dependências com um propósito em comum. Dessa forma, somente uma configuração é realizada no seu gerenciador de dependências.
- Alguns exemplos de starters disponíveis são:

Spring Boot Starters

- **Spring Boot Starter Web:** auxilia na construção de aplicações web trazendo já disponíveis para uso **Spring MVC**, **Rest** e o **Tomcat** como servidor.
- **Spring Boot Starter Test:** Contém a maioria das dependências necessárias para realizar testes da sua aplicação: **Junit**, **AssertJ**, **Hamcrest**, **Mockito**, entre outros.
- **Spring Boot Starter Data JPA:** Facilita a construção da nossa camada de persistência, ajudando na abstração do nosso banco de dados provendo uma série de facilidades para criação de repositories, escrita de queries, entre outros.

Instalação e configuração

Prof. Marcelo Estruc

Instalação e configuração

Instalando o STS















O próximo passo será a instalação de uma IDE Java para desenvolvermos nossas aplicações. Utilizaremos o STS, que pode ser baixado e instalado no link a seguir:

<https://spring.io/tools>

Instalação e configuração de uma aplicação

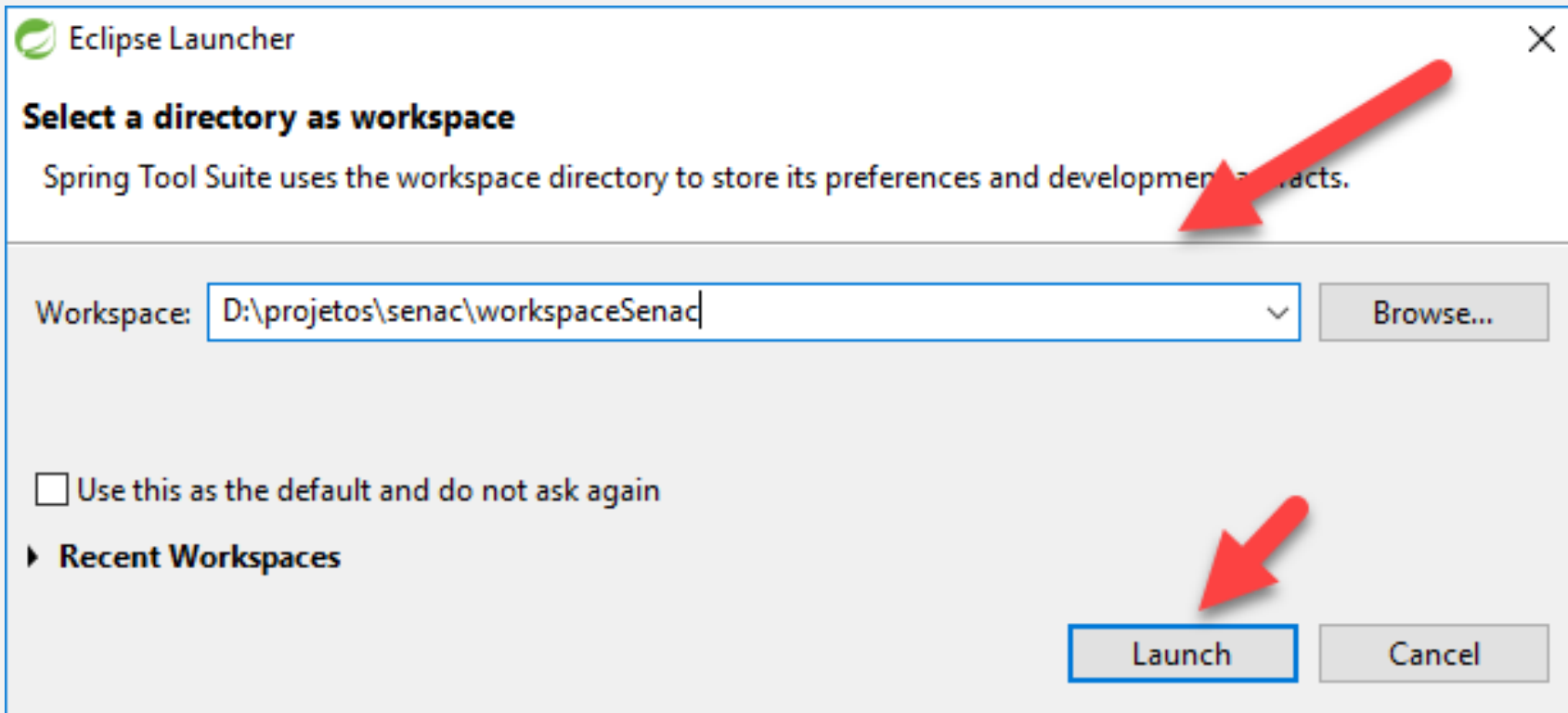
Mão na massa

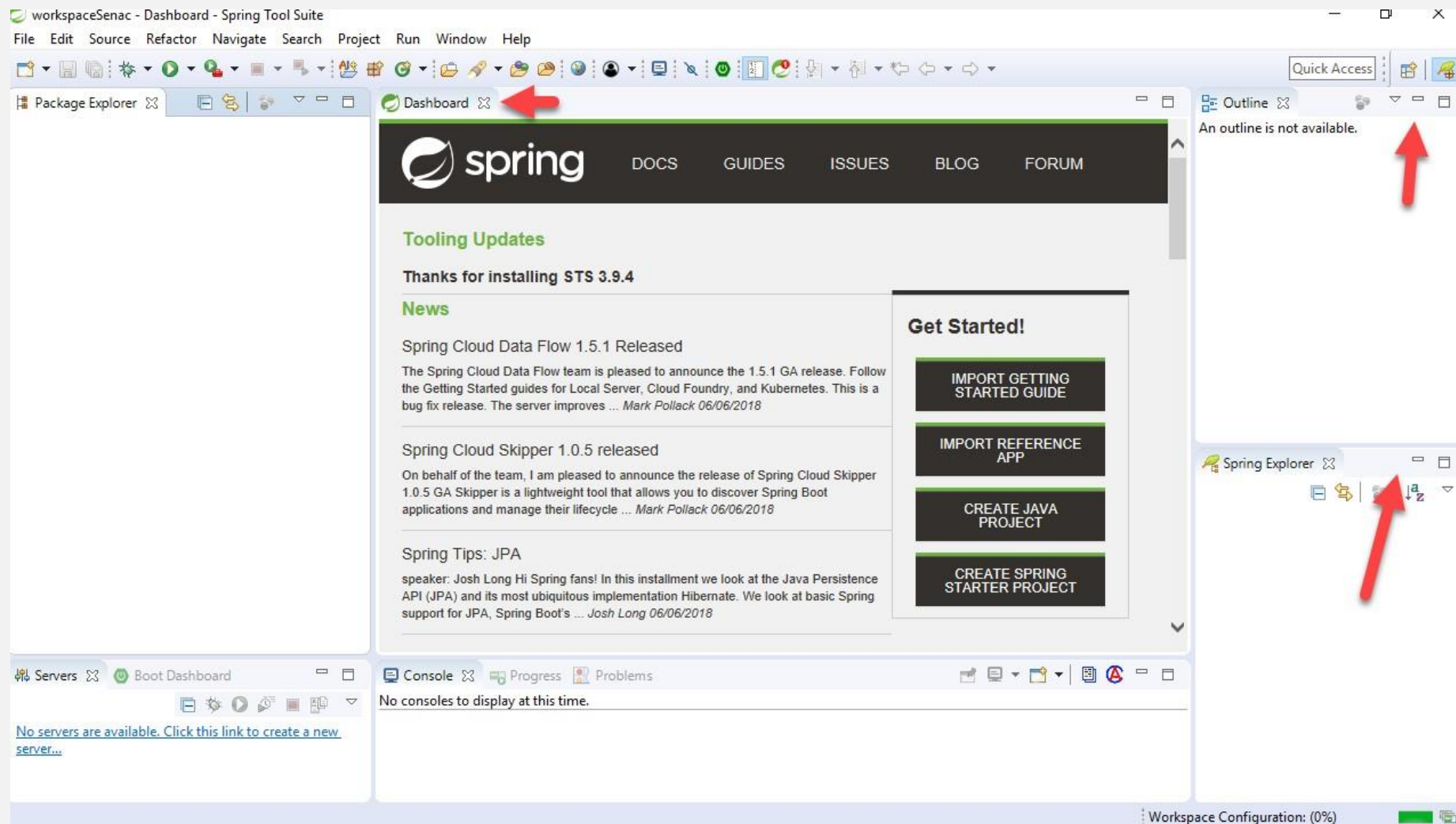


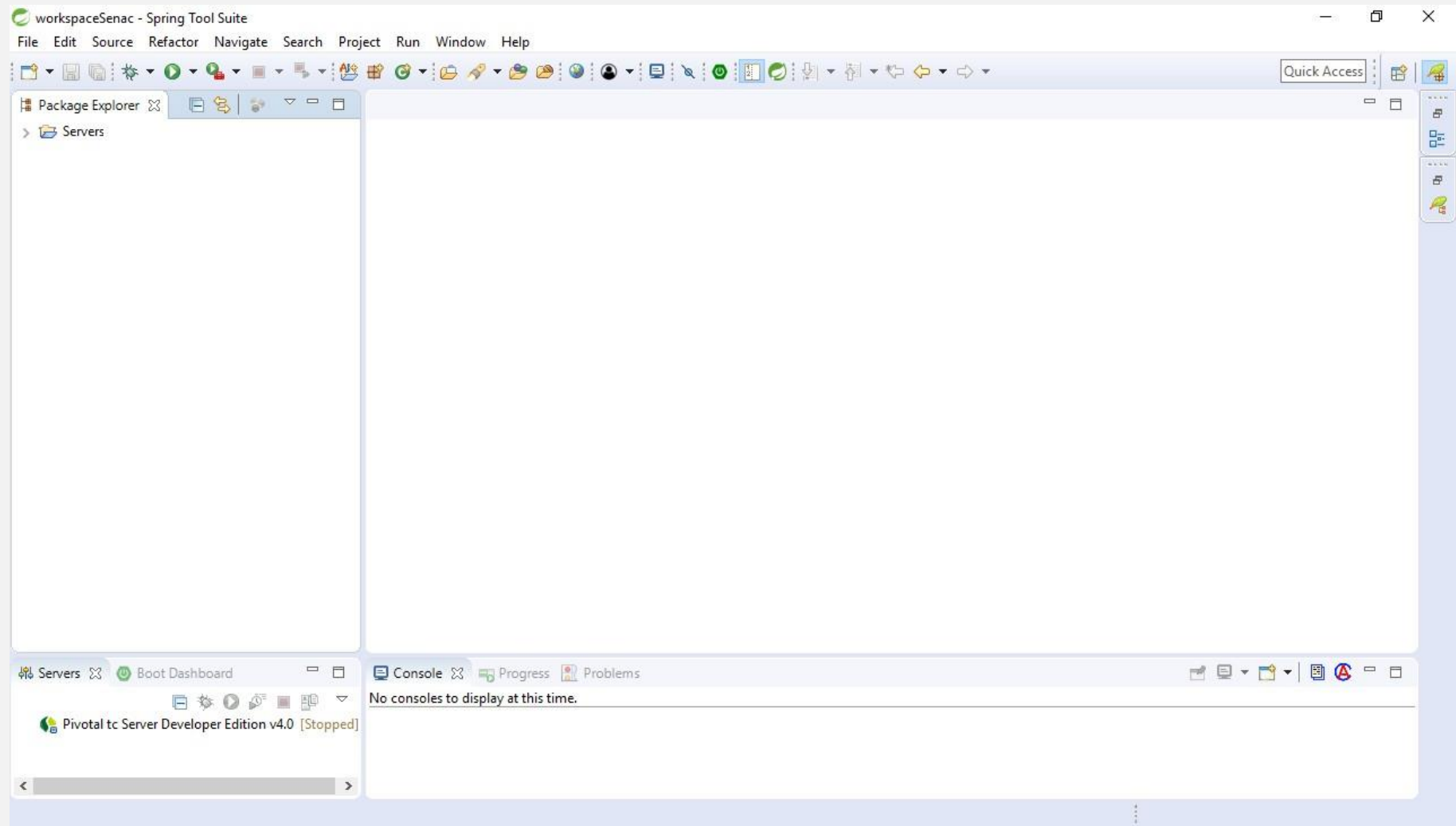
<input type="checkbox"/>	Nome	Data de modificaç...	Tipo	Tamanho
	configuration	07/06/2018 15:32	Pasta de arquivos	
	dropins	12/04/2018 02:24	Pasta de arquivos	
	features	12/04/2018 02:24	Pasta de arquivos	
	META-INF	12/04/2018 02:24	Pasta de arquivos	
	p2	07/06/2018 15:46	Pasta de arquivos	
	plugins	12/04/2018 02:24	Pasta de arquivos	
	readme	12/04/2018 02:24	Pasta de arquivos	
	.eclipseproduct	12/04/2018 02:25	Arquivo ECLIPSEP...	1 KB
	artifacts.xml	12/04/2018 02:24	Documento XML	279 KB
	eclipsesec.exe	12/04/2018 02:21	Aplicativo	18 KB
	license.txt	12/04/2018 02:16	Arquivo TXT	12 KB
	open_source_licenses.txt	12/04/2018 02:16	Arquivo TXT	2.095 KB
<input checked="" type="checkbox"/>	 STS.exe	12/04/2018 02:21	Aplicativo	306 KB
	STS.ini	12/04/2018 02:24	Parâmetros de co...	1 KB

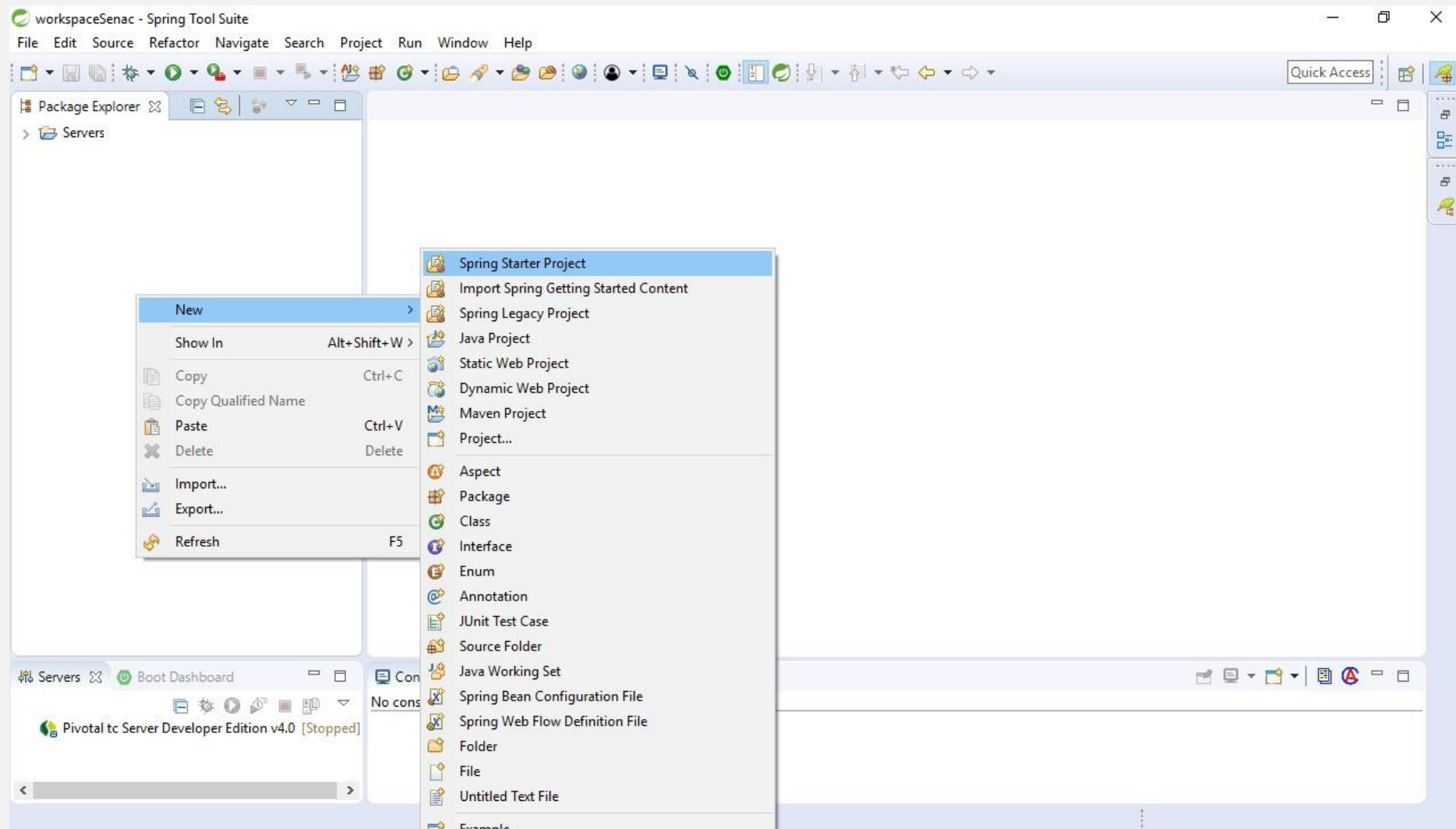
Configuration	07/06/2018 10:17	Pasta de arquivos
dropins	12/04/2018 02:24	Pasta de arquivos
features	12/04/2018 02:24	Pasta de arquivos
META-INF	12/04/2018 02:24	Pasta de arquivos
p2	07/06/2018 15:46	Pasta de arquivos
plugins	12/04/2018 02:24	Pasta de arquivos
readme		
.eclipseproduct		
artifacts.xml		
eclipsec.exe		
license.txt		
open_source_licenses.txt		
STS.exe		
STS.ini		











New Spring Starter Project



Service URL

Name

☒ Use default location

Location

Type: Packaging:

Java Version: Language:

Group

Artifact

Version

Description

Package

Working sets

☐ Add project to working sets

Working sets:

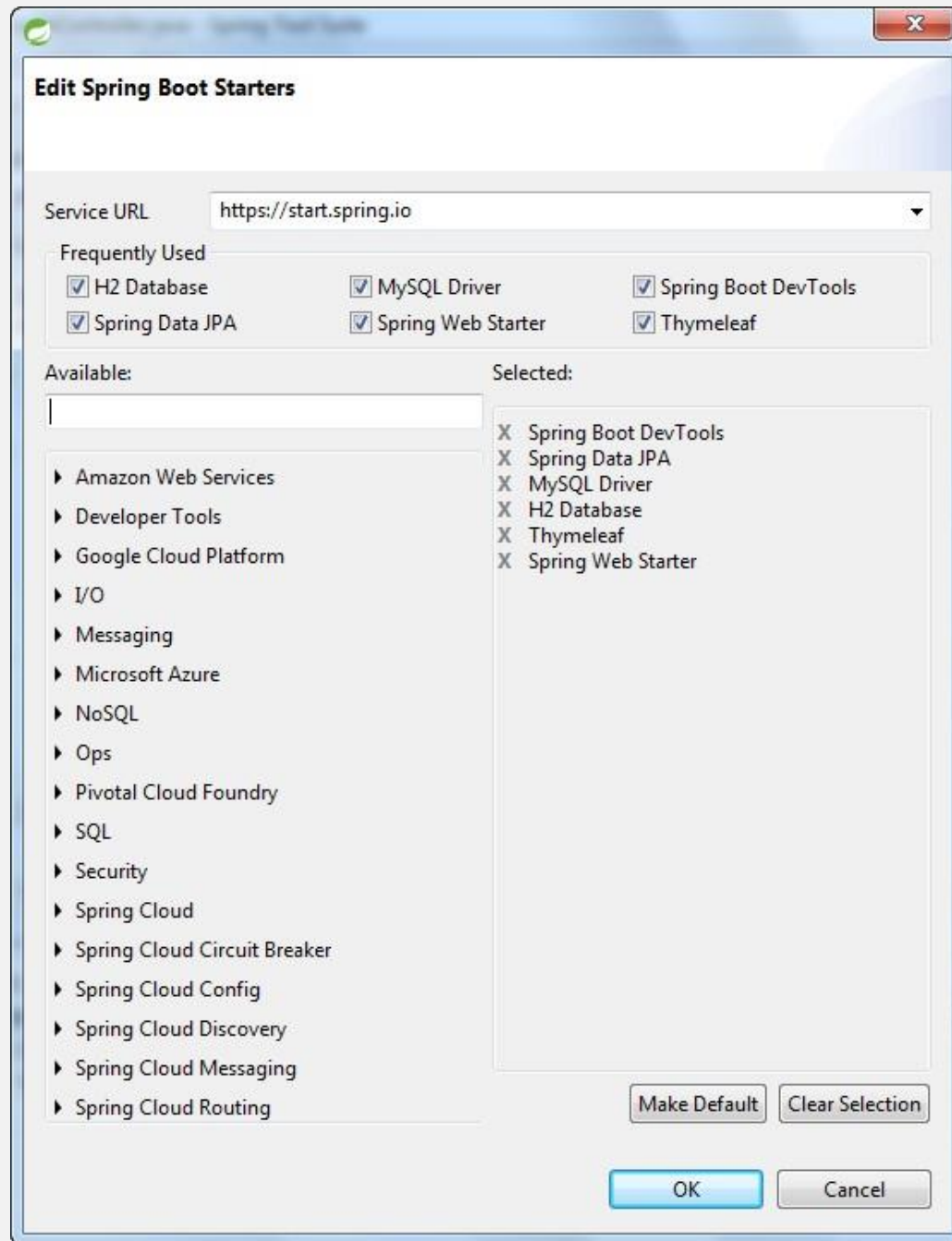


< Back

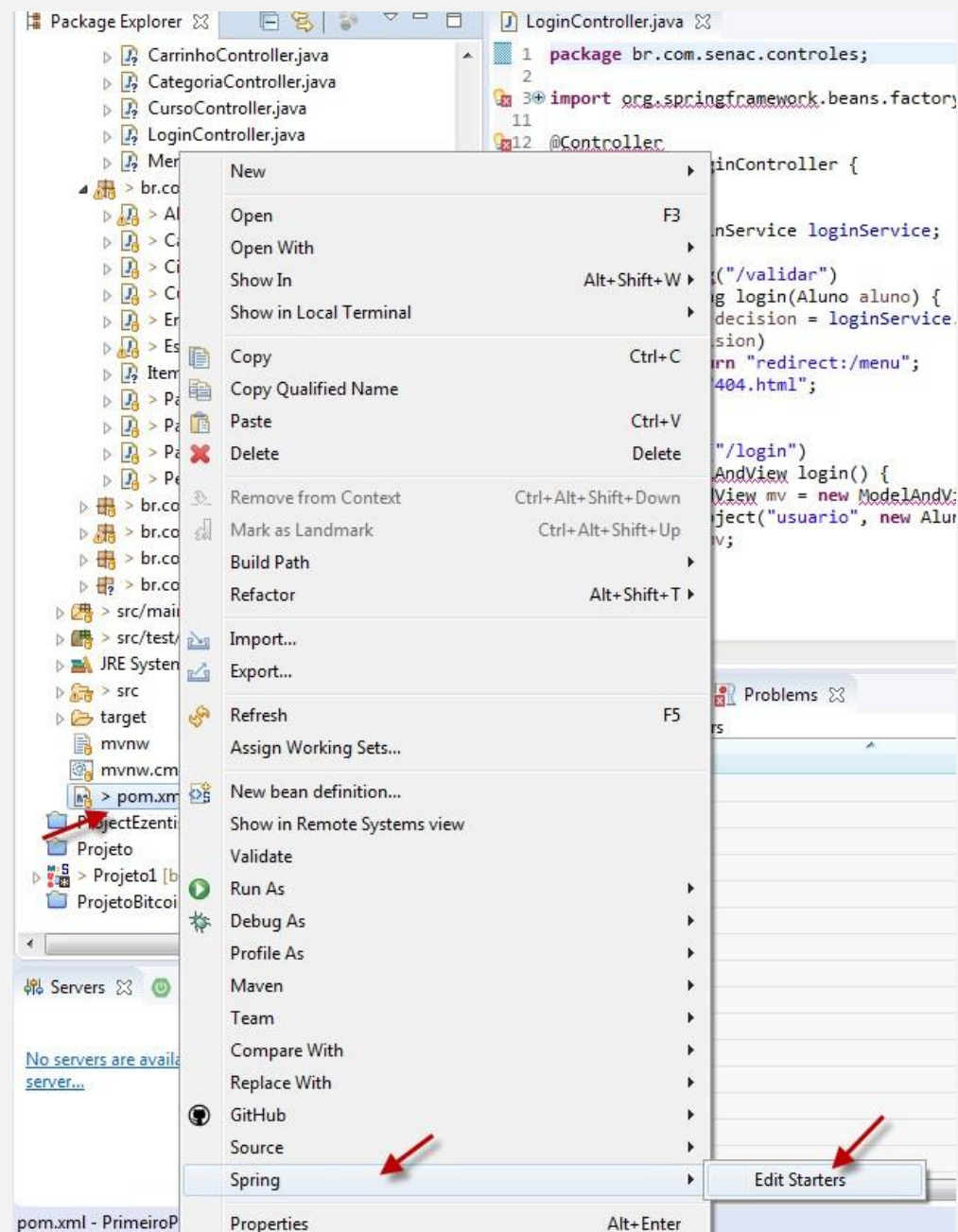
Next >

Finish

Cancel



Se quiser acessar quando sair.





New Spring Starter Project



Site Info

Base Url

<https://start.spring.io/starter.zip>

Full Url

[https://start.spring.io/starter.zip?
name=PrimeiroProjeto&groupId=Senac&artifactId=PrimeiroProjeto&v
ersion=0.0.1-SNAPSHOT&description=Primeiro+Projeto+com
+Spring+Boot&packageName=br.com.senac&type=maven-
project&packaging=war&javaVersion=1.8&language=java&bootVersio](https://start.spring.io/starter.zip?name=PrimeiroProjeto&groupId=Senac&artifactId=PrimeiroProjeto&version=0.0.1-SNAPSHOT&description=Primeiro+Projeto+com+Spring+Boot&packageName=br.com.senac&type=maven-project&packaging=war&javaVersion=1.8&language=java&bootVersion=)

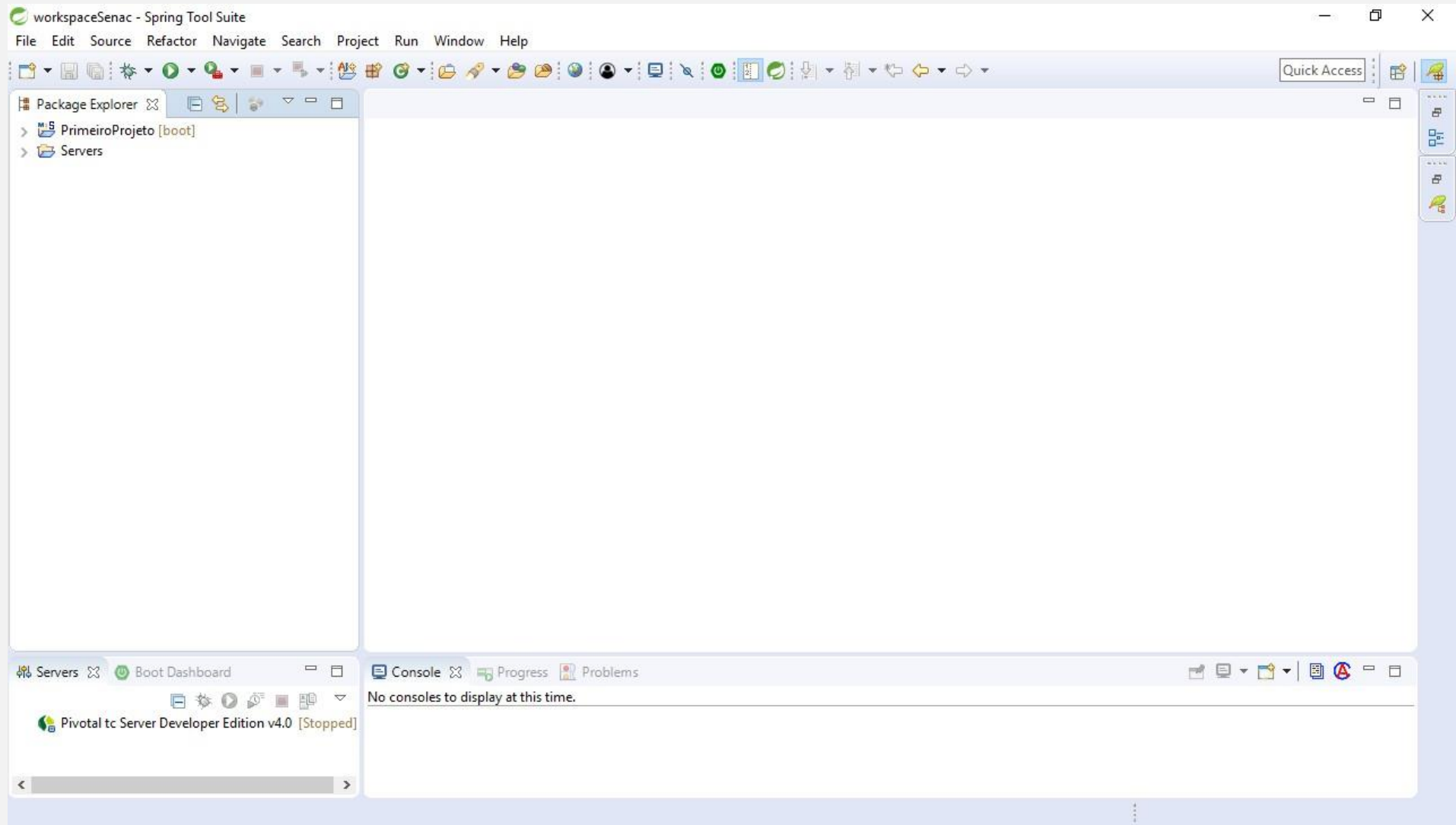




















< Back

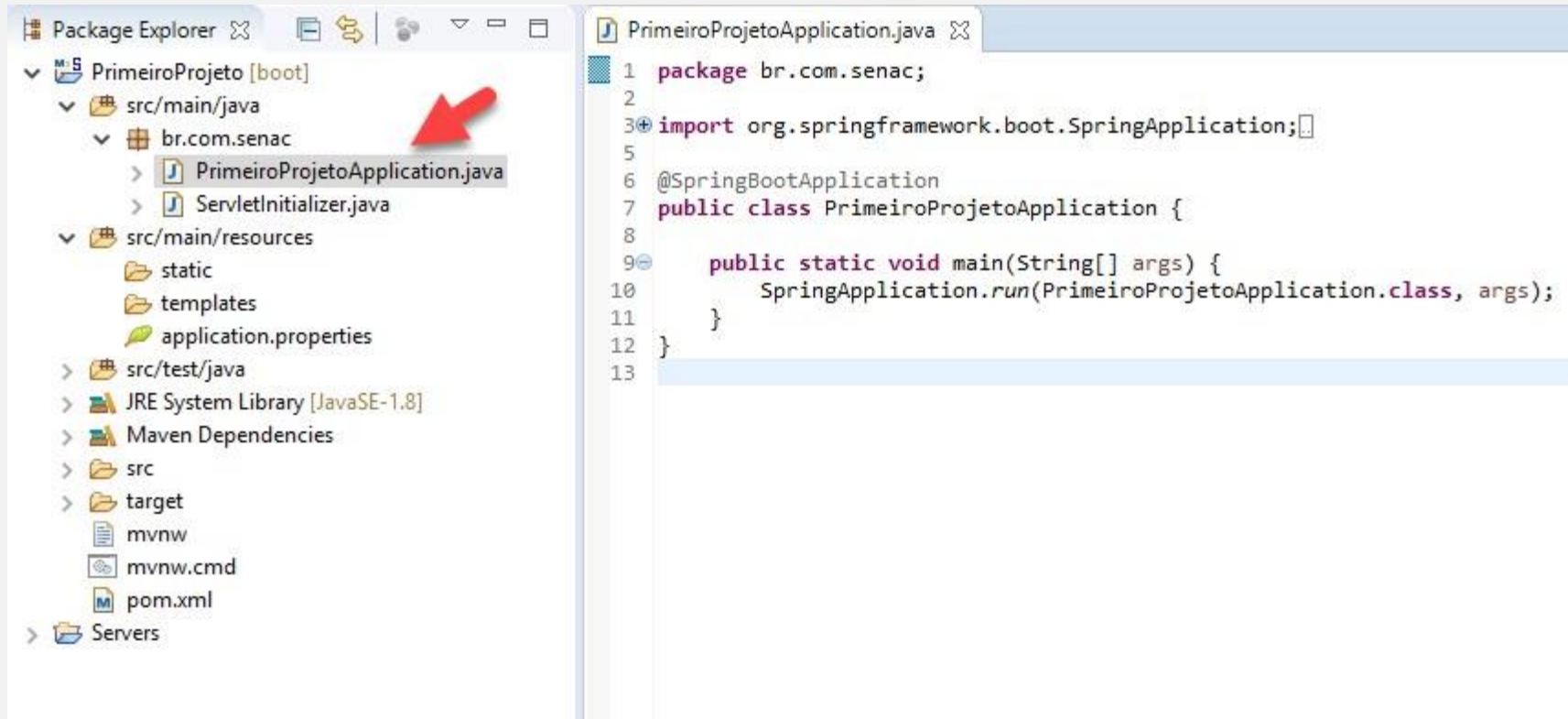
Next >

Finish

Cancel



- ▼  PrimeiroProjeto [boot]
 - ▼  src/main/java
 - ▼  br.com.senac
 - >  PrimeiroProjetoApplication.java
 - >  ServletInitializer.java
 - ▼  src/main/resources
 -  static
 -  templates
 -  application.properties
 - >  src/test/java
 - >  JRE System Library [JavaSE-1.8]
 - >  Maven Dependencies
 - >  src
 - >  target
 -  mvnw
 -  mvnw.cmd
 -  pom.xml
- >  Servers



m/senac/PrimeiroProjetoApplication.java - Spring Tool S

ct Run Window Help

PrimeiroProjetoApplication.java

```
1 package br.com.senac;
2
3 import org.springframework.boot.Sprin
4
5 @SpringBootApplication
6
7 public class PrimeiroProjetoApplicati
8
9 public static void main(String[]
10     SpringApplication.run(Primeir
11 }
12
13
```

- 1 Run on Server Alt+Shift+X, R
- 2 Java Application Alt+Shift+X, J
- 3 Spring Boot App Alt+Shift+X, B
- Run Configurations...

- Undo Ctrl+Z
- Revert File
- Save Ctrl+S
- Open Declaration F3
- Open Type Hierarchy F4
- Open Call Hierarchy Ctrl+Alt+H
- Show in Breadcrumb Alt+Shift+B
- Quick Outline Ctrl+O
- Quick Type Hierarchy Ctrl+T
- Open With >
- Show In Alt+Shift+W >
- Cut Ctrl+X
- Copy Ctrl+C
- Copy Qualified Name
- Paste Ctrl+V
- Quick Fix Ctrl+1
- Source Alt+Shift+S >
- Refactor Alt+Shift+T >
- Local History >
- References >
- Declarations >
- Add to Snippets...
- AspectJ Refactoring >
- Run As >
- Debug As >
- Profile As >
- Validate
- GitHub >
- Team >





localhost:8080



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Jun 07 16:43:48 BRT 2018

There was an unexpected error (type=Not Found, status=404).

No message available

maven

Prof. Marcelo Estruc

POM.xml

Project Object Model ou POM, é a peça fundamental de um projeto do Maven. Um POM possui as informações básicas de um projeto, bem como as diretivas de como o artefato final deste projeto deve ser construído.

O arquivo **pom.xml** é considerado o coração de um projeto **Maven**. Com a configuração de poucos descritores é possível gerenciar dependências, centralizar documentação sobre o projeto e principalmente compilar e distribuir uma aplicação.

Configurando o banco de dados H2. Vá ate o site <https://mvnrepository.com/> e busque por H2

Package Explorer

- PrimeiroProjeto [boot]
 - src/main/java
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - mvnw
 - mvnw.cmd
 - pom.xml**
- Servers

PrimeiroProjeto/pom.xml

```
28<dependency>
29  <groupId>org.springframework.boot</groupId>
30  <artifactId>spring-boot-starter-web</artifactId>
31</dependency>
32
33<dependency>
34  <groupId>org.springframework.boot</groupId>
35  <artifactId>spring-boot-starter-tomcat</artifactId>
36  <scope>provided</scope>
37</dependency>
38<dependency>
39  <groupId>org.springframework.boot</groupId>
40  <artifactId>spring-boot-starter-test</artifactId>
41  <scope>test</scope>
42</dependency>
43<dependency>
44  <groupId>org.springframework.boot</groupId>
45  <artifactId>spring-boot-starter-data-jpa</artifactId>
46</dependency>
47<dependency>
48  <groupId>com.h2database</groupId>
49  <artifactId>h2</artifactId>
50  <scope>runtime</scope>
51</dependency>
52<dependency>
53  <groupId>org.springframework.boot</groupId>
54  <artifactId>spring-boot-devtools</artifactId>
55</dependency>
56</dependencies>
```


Configurando o banco de dados H2. Vá ate o site <https://mvnrepository.com/> e busque por H2

The screenshot shows the Maven Repository website with a search for 'h2'. The search bar at the top contains 'h2' and a red arrow points to it. Below the search bar, the text 'Found 29 results' is displayed. The results are sorted by 'relevance'. The first result is '1. H2 Database Engine' by 'com.h2database', which has 5,072 usages and is licensed under MPL and EPL. A red arrow points to this result. The second result is '2. WSO2 Carbon Orbit H2 Osgi Console Service' by 'com.h2database.wso2', with 124 usages and MPL/EPL licenses. The third result is '3. WSO2 Carbon Orbit H2 OSGi Console Service' by 'org.wso2.orbit.com.h2database', with 26 usages. The fourth result is partially visible at the bottom: '4. H2 Database Engine' with 10 usages. On the left side, there are filters for 'Repository' (Central, Sonatype, etc.) and 'Group' (com.h2database, org.eclipse, etc.).

MVNREPOSITORY

h2 | Search

Repository

- Central 19
- Sonatype 7
- Boundless 2
- Spring Lib M 2
- WSO2 Public 2
- CamundaBPM 1
- EBI 1
- Jabylon 1

Group

- com.h2database 3
- org.eclipse 3
- org.orbisgis 2
- com.goldmansachs 1
- com.walterjwhite 1
- io.r2dbc 1
- io.thorntail 1
- it.rebase 1

Found 29 results

Sort: **relevance** | popular | newest

1. H2 Database Engine 5,072 usages
com.h2database » h2
H2 Database Engine
Last Release on Mar 13, 2019
MPL EPL

2. WSO2 Carbon Orbit H2 Osgi Console Service 124 usages
com.h2database.wso2 » h2-database-engine
WSO2 Carbon Orbit H2 Osgi Console Service
Last Release on Aug 8, 2012
MPL EPL

3. WSO2 Carbon Orbit H2 OSGi Console Service 26 usages
org.wso2.orbit.com.h2database » h2
WSO2 Carbon Orbit H2 OSGi Console Service
Last Release on Jul 8, 2019

4. H2 Database Engine 10 usages



H2 Database Engine

H2 Database Engine

License	EPL 1.0 MPL 2.0
Categories	Embedded SQL Databases
Tags	database embedded sql
Used By	5,072 artifacts

Central (127)

WSO2 Dist (2)

Redhat GA (8)

ICM (1)

Nuiton (1)

Version	Repository	Usages	Date
1.4.199	Central	335	Mar, 2019
1.4.198	Central	69	Feb, 2019
1.4.197	Central	918	Mar, 2018
1.4.196	Central	777	Jun, 2017
1.4.195	Central	184	Apr, 2017
1.4.194	Central	187	Mar, 2017

Home » com.h2database » h2 » 1.4.199



H2 Database Engine » 1.4.199

H2 Database Engine

License	EPL 1.0 MPL 2.0
Categories	Embedded SQL Databases
HomePage	http://www.h2database.com
Date	(Mar 13, 2019)
Files	jar (2.1 MB) View All
Repositories	Central
Used By	5,072 artifacts

[Maven](#)

[Gradle](#)

[SBT](#)

[Ivy](#)

[Grape](#)

[Leiningen](#)

[Buildr](#)

```
<!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <version>1.4.199</version>
  <scope>test</scope>
</dependency>
```



Copiar o texto e tirar version e
scope colocar runtime

Banco de Dados em memória

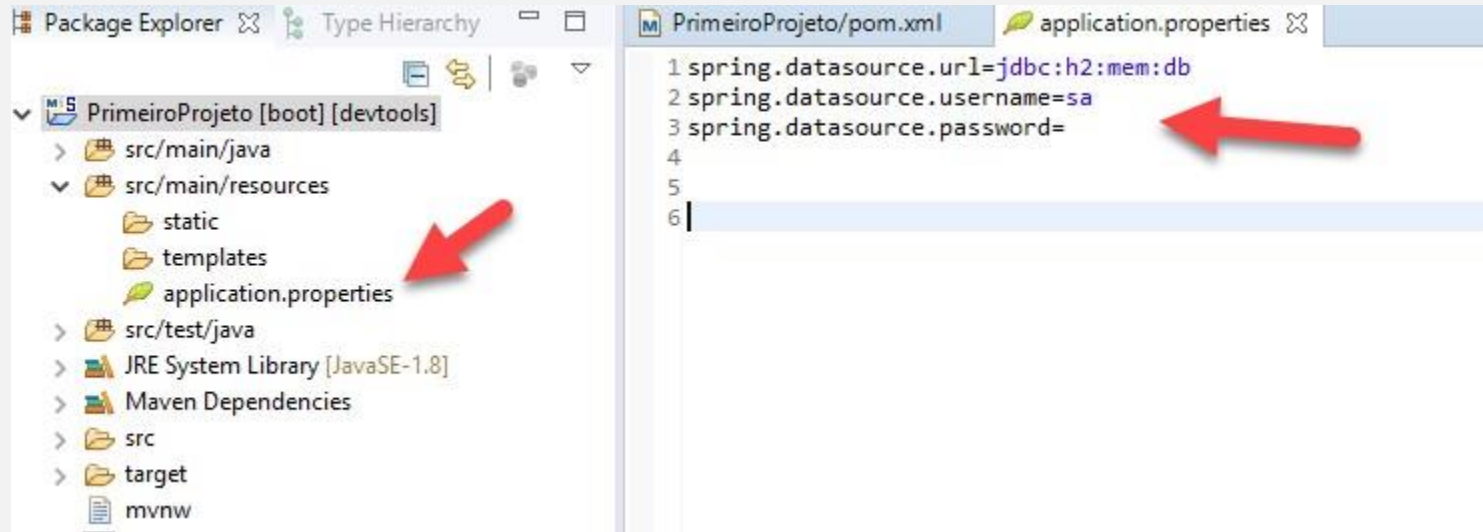
Prof. Marcelo Estruc

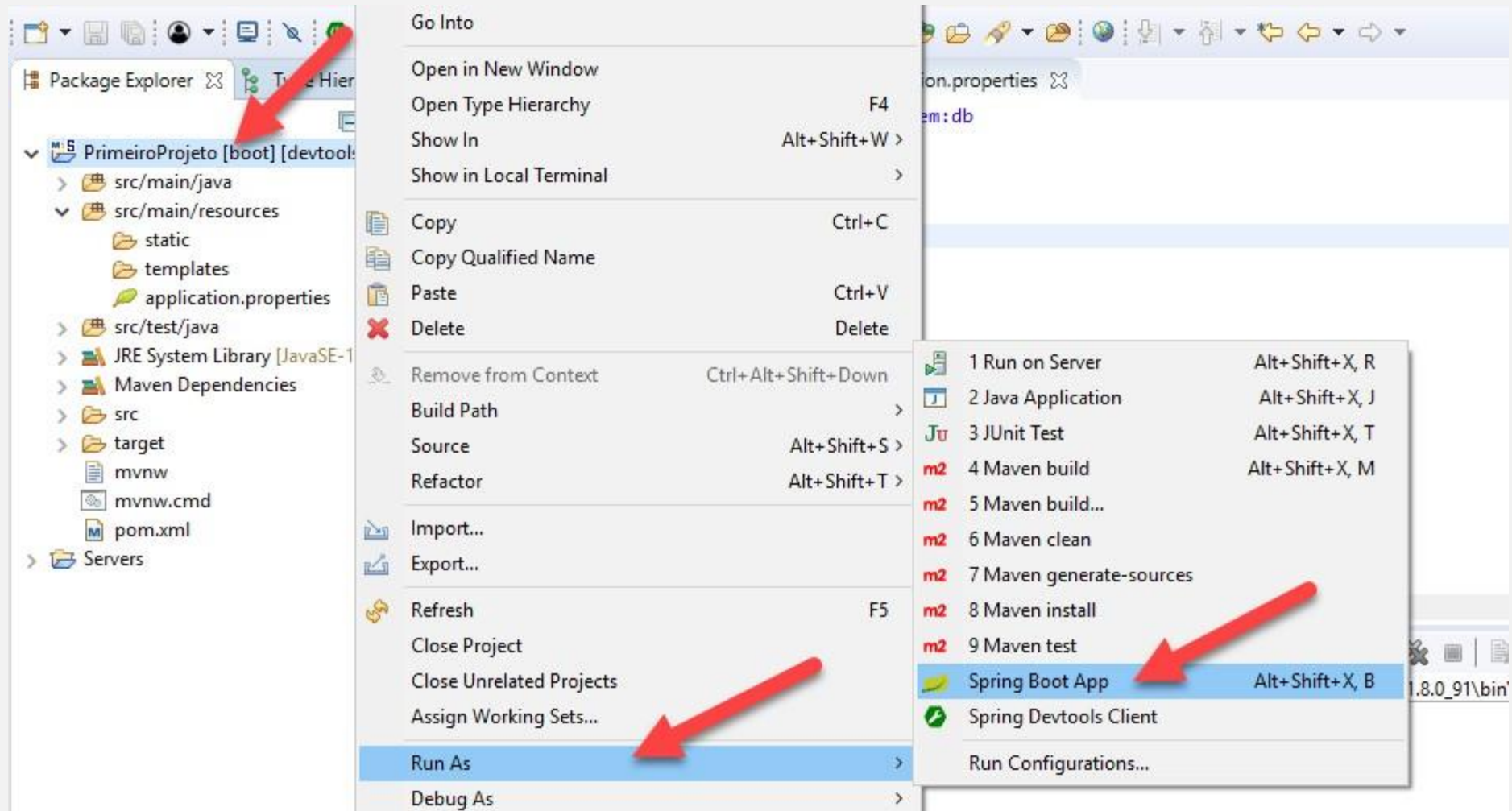
H2

H2 é um banco de dados Java leve de código aberto. Ele pode ser incorporado em aplicativos Java ou executado no modo cliente-servidor. Principalmente, o banco de dados H2 pode ser configurado para ser executado como banco de dados na memória, o que significa que os dados não persistirão no disco. Por causa do banco de dados embutido, ele não é usado para desenvolvimento de produção, mas usado principalmente para desenvolvimento e teste.

Mão na massa







explorer Type Hierarchy

PrimeiroProjeto [boot] [devtools]

- c/main/java
- c/main/resources
- static
- templates
- application.properties
- c/test/java
- System Library [JavaSE-1.8]
- aven Dependencies
- c
- rgset
- vnw
- vnw.cmd
- om.xml
- rs

Boot Dashboard

al tc Server Developer Edition v4.0 [Stopped]

PrimeiroProjeto/pom.xml application.properties

```
1 spring.datasource.url=jdbc:h2:mem:db
2 spring.datasource.username=sa
3 spring.datasource.password=
4
5
```

Console Progress Problems

PrimeiroProjeto - PrimeiroProjetoApplication [Spring Boot App] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (15 de jun de 2018 19:26:06)

```
2018-06-15 19:26:17.036 INFO 6132 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(
2018-06-15 19:26:17.136 INFO 6132 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2018-06-15 19:26:17.136 INFO 6132 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache
2018-06-15 19:26:17.152 INFO 6132 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener : The APR based Apache Tomcat Native
2018-06-15 19:26:17.483 INFO 6132 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[/] : Initializing Spring embedded web applic
2018-06-15 19:26:17.483 INFO 6132 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initializ
2018-06-15 19:26:17.769 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Servlet dispatcherServlet mapped
2018-06-15 19:26:17.769 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Servlet webServlet mapped to /
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncoding
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethod
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormCon
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContext
2018-06-15 19:26:18.554 INFO 6132 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2018-06-15 19:26:18.755 INFO 6132 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed
2018-06-15 19:26:18.955 INFO 6132 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Building JPA container EntityMan
2018-06-15 19:26:19.040 INFO 6132 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing Persistence
    name: default
    ...]
2018-06-15 19:26:19.625 INFO 6132 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate Core {5.2.10.Final}
2018-06-15 19:26:19.630 INFO 6132 --- [ restartedMain] org.hibernate.cfg.Environment : HHH000206: hibernate.properties not f
2018-06-15 19:26:20.110 INFO 6132 --- [ restartedMain] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Common
2018-06-15 19:26:21.611 INFO 6132 --- [ restartedMain] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate
2018-06-15 19:26:22.991 INFO 6132 --- [ restartedMain] o.h.t.schema.internal.SchemaCreatorImpl : HHH000476: Executing import s
2018-06-15 19:26:23.008 INFO 6132 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerF
2018-06-15 19:26:23.222 INFO 6132 --- [ restartedMain] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.
2018-06-15 19:26:23.906 INFO 6132 --- [ restartedMain] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
2018-06-15 19:26:23.959 WARN 6132 --- [ restartedMain] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is ena
2018-06-15 19:26:24.119 INFO 6132 --- [ restartedMain] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto publi
2018-06-15 19:26:24.122 INFO 6132 --- [ restartedMain] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[t
```

← → ↻ 🏠

localhost:8080/h2-console/

⚙️ Mais visitados 🌐 Primeiros passos 📁 Advogados 📁 CTPA 📁 Ponnta 📁 v

English ▾ [Preferences](#) [Tools](#) [Help](#)

Login

Saved Settings: Generic H2 (Embedded) ▾

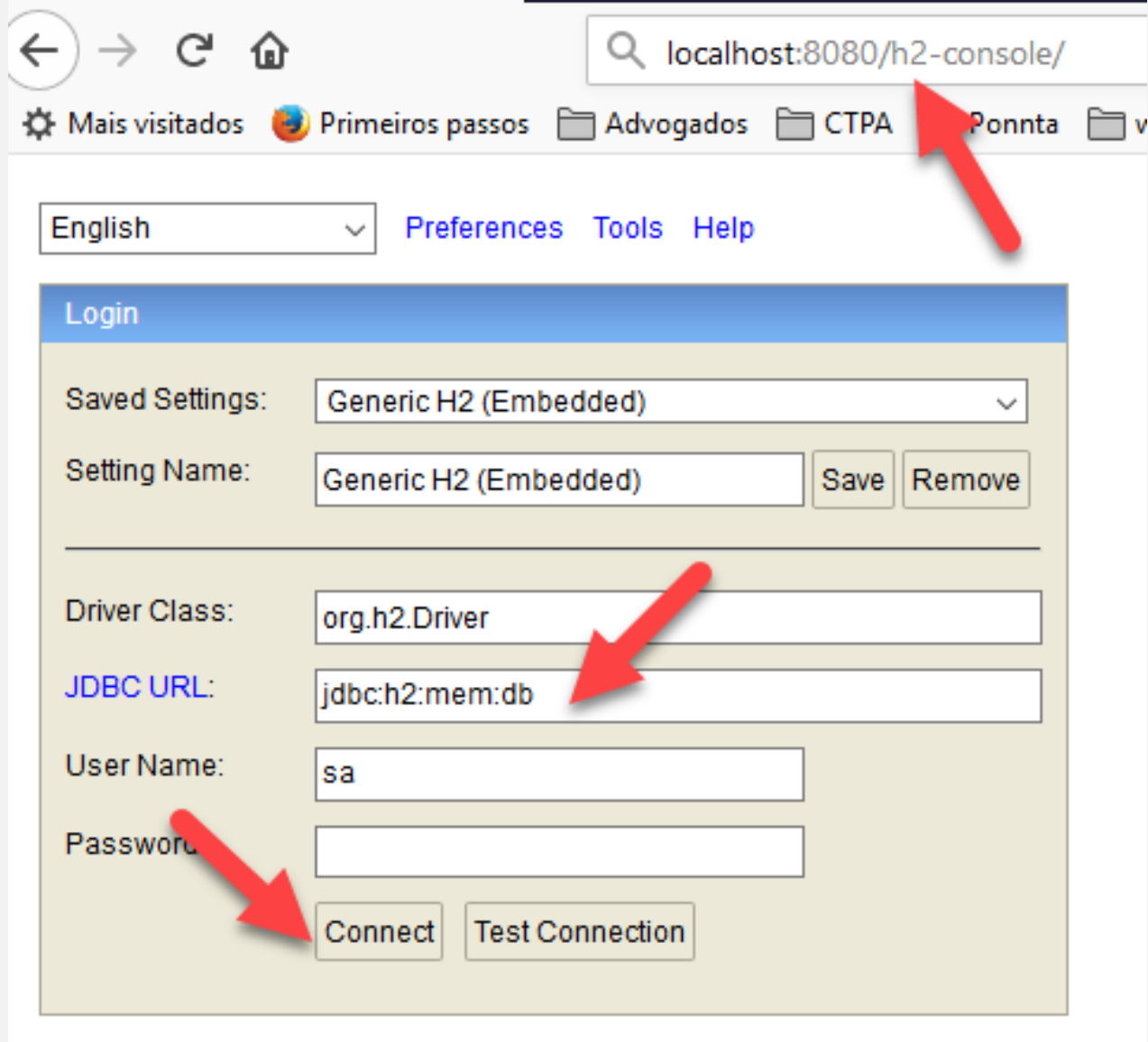
Setting Name: Generic H2 (Embedded)

Driver Class: org.h2.Driver

[JDBC URL:](#) jdbc:h2:mem:db

User Name: sa

Password:

The image is a screenshot of a web browser displaying the H2 console login page. The browser's address bar shows 'localhost:8080/h2-console/'. Below the address bar, there are several tabs: 'Mais visitados', 'Primeiros passos', 'Advogados', 'CTPA', 'Ponnta', and 'v'. The main content area has a header with 'English' (a dropdown menu), 'Preferences', 'Tools', and 'Help'. The 'Login' section contains a 'Saved Settings' dropdown set to 'Generic H2 (Embedded)', a 'Setting Name' field with the same text, and 'Save' and 'Remove' buttons. Below this is a horizontal line. The 'Driver Class' field contains 'org.h2.Driver'. The 'JDBC URL' field, with a blue link icon to its left, contains 'jdbc:h2:mem:db'. The 'User Name' field contains 'sa'. The 'Password' field is empty. At the bottom are 'Connect' and 'Test Connection' buttons. Three red arrows are overlaid on the image: one points to the address bar, another points to the 'JDBC URL' field, and a third points to the 'Connect' button.

jdbc:h2:mem:db
+ INFORMATION_SCHEMA
+ Users
i H2 1.4.197 (2018-03-18)

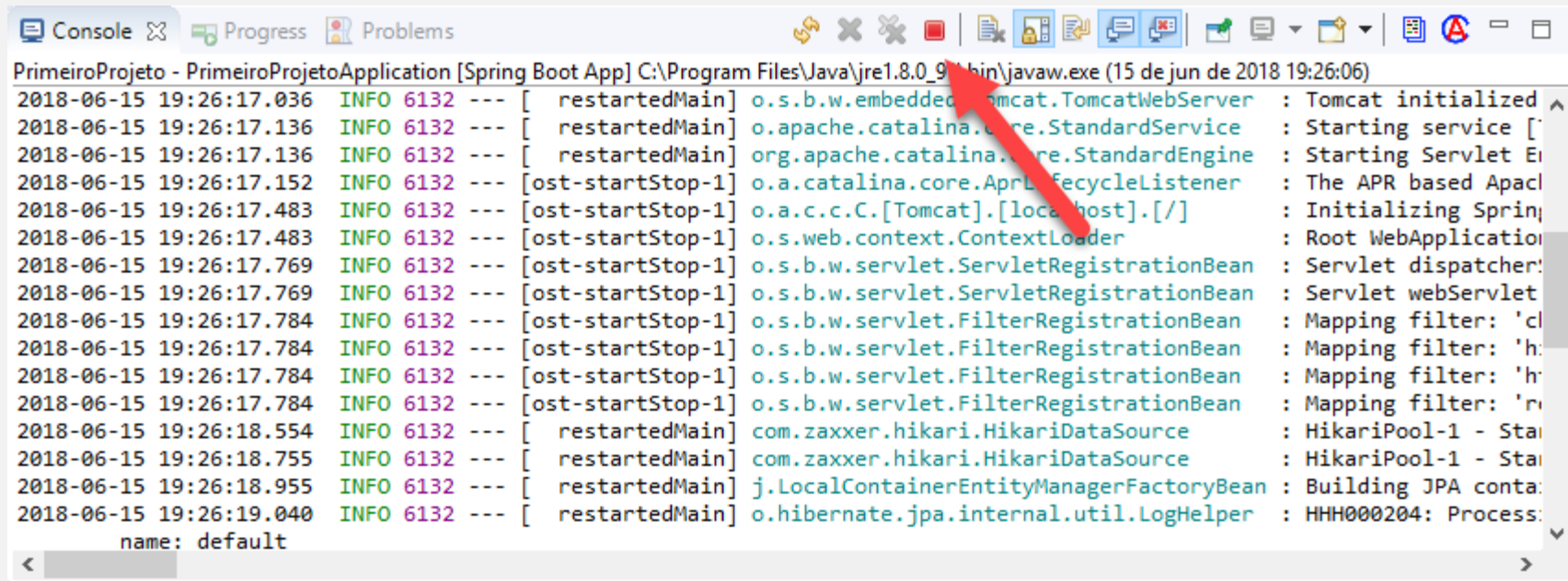
Run Run Selected Auto complete Clear SQL statement:

Important Commands

?		Displays this Help Page
⇅		Shows the Command History
▶	Ctrl+Enter	Executes the current SQL statement
▶	Shift+Enter	Executes the SQL statement defined by the text selection
	Ctrl+Space	Auto complete
🔌		Disconnects from the database

Sample SQL Script

Vamos parar a aplicação



The screenshot shows an IDE console window with the following elements:

- Tab Bar:** Contains 'Console', 'Progress', and 'Problems' tabs. 'Console' is selected.
- Toolbar:** Includes icons for running, debugging, and other IDE functions.
- Console Title:** 'PrimeiroProjeto - PrimeiroProjetoApplication [Spring Boot App] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (15 de jun de 2018 19:26:06)'
- Log Output:** A series of log entries with timestamps, log levels, and messages. A red arrow points to the line: `2018-06-15 19:26:17.136 INFO 6132 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized`

The log entries are as follows:

```
2018-06-15 19:26:17.036 INFO 6132 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized
2018-06-15 19:26:17.136 INFO 6132 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [
2018-06-15 19:26:17.136 INFO 6132 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet En
2018-06-15 19:26:17.152 INFO 6132 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener : The APR based Apac
2018-06-15 19:26:17.483 INFO 6132 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring
2018-06-15 19:26:17.483 INFO 6132 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplication
2018-06-15 19:26:17.769 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Servlet dispatcher:
2018-06-15 19:26:17.769 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Servlet webServlet
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'cl
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'h:
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'h:
2018-06-15 19:26:17.784 INFO 6132 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'r:
2018-06-15 19:26:18.554 INFO 6132 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Sta
2018-06-15 19:26:18.755 INFO 6132 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Sta
2018-06-15 19:26:18.955 INFO 6132 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Building JPA conta:
2018-06-15 19:26:19.040 INFO 6132 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Process:
name: default
```

Modelo MVC

Prof. Marcelo Estruc

O Modelo MVC

- O MVC é utilizado em muitos projetos devido à arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o **Model**, o **Controller** e a **View**, executa o que lhe é definido e nada mais do que isso.
- A utilização do padrão MVC trás como benefício isolar as regras de negócios da lógica de apresentação, a interface com o usuário.

O Modelo MVC

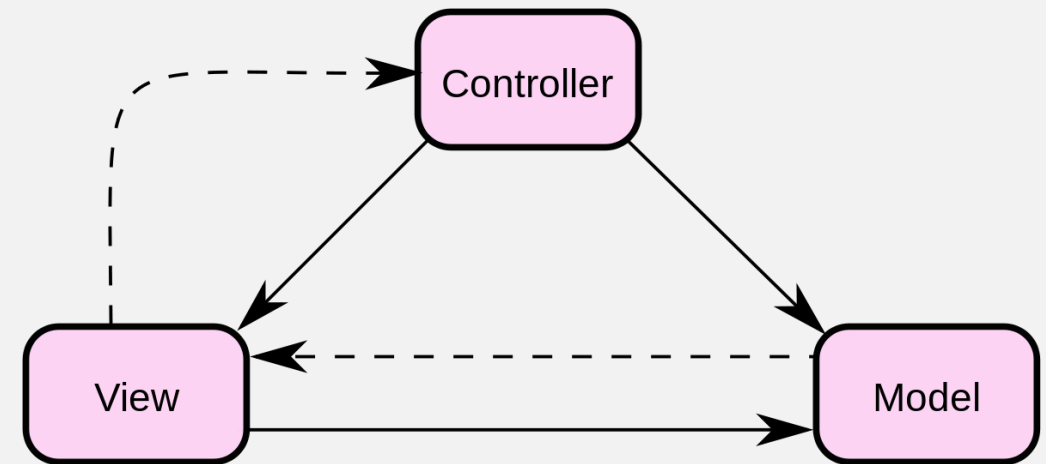
- Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reuso das classes.
- Uma das características de um padrão de projeto é poder aplicá-lo em sistemas distintos. O padrão MVC pode ser utilizado em vários tipos de projetos como, por exemplo, **desktop**, **web** e **mobile**.

O Modelo MVC

- A comunicação entre interfaces e regras de negócios é definida através de um controlador, e é a existência deste controlador que torna possível a separação entre as camadas. Quando um evento é executado na interface gráfica, como um clique em um botão, a interface irá se comunicar com o controlador que por sua vez se comunica com as regras de negócios.

O Modelo MVC

O usuário interage com a interface gráfica que é a camada **View**. A interface gráfica interage com um intermediador que é o **Controller**, e este interage com o **Model** que executa as regras de negócios do sistema.



Exemplo de projeto

Prof. Marcelo Estruc

O que vamos fazer?
Como iremos fazer?



Projeto

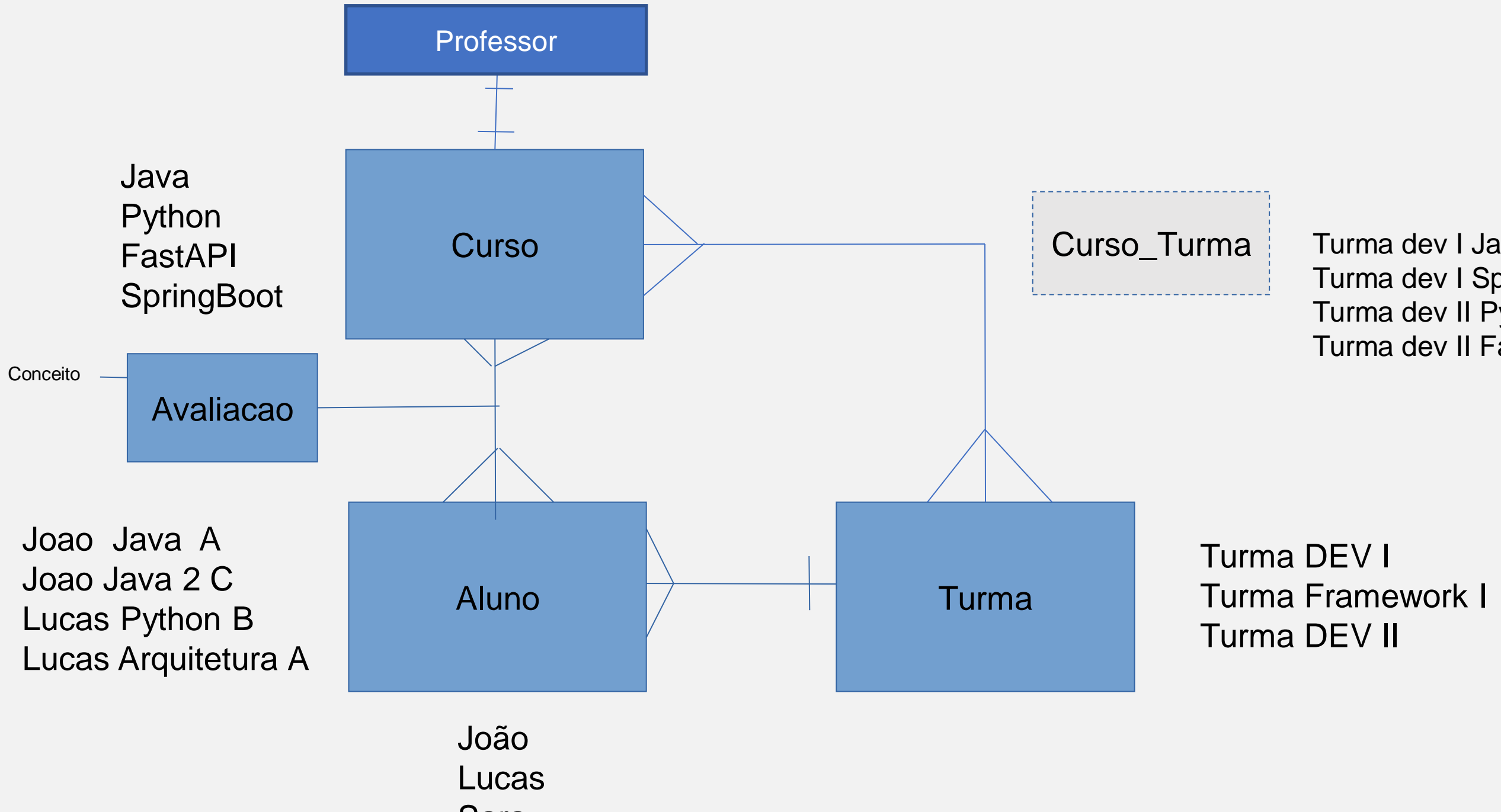
A empresa TiparaDev atua no mercado a mais de 10 anos e possui professores qualificados ministrando cursos para formar desenvolvedores nas principais linguagens e frameworks do mercado. Dentre os cursos estão: Java, NodeJS, Python, SpringBoot, MongoDB, FastAPI, SQL, ReactJS e Angular. Um professor pode lecionar somente um único curso. A empresa não permite que o professor ministre aulas em mais de um curso. Para o aluno se matricular ele deve abrir uma turma. Cada turma pode ser formado por um ou mais cursos. Ao final de cada curso o aluno é avaliado com uma nota com conceitos A,B,C e D.

A empresa deseja ter um sistema que possa controlar a avaliação de cada aluno, gerenciar as turmas e gerenciar os cursos.

Modelagem de dados

Diante do mini mundo apresentado, crie um modelo lógico que represente o sistema que será elaborado para a empresa Tipara Dev

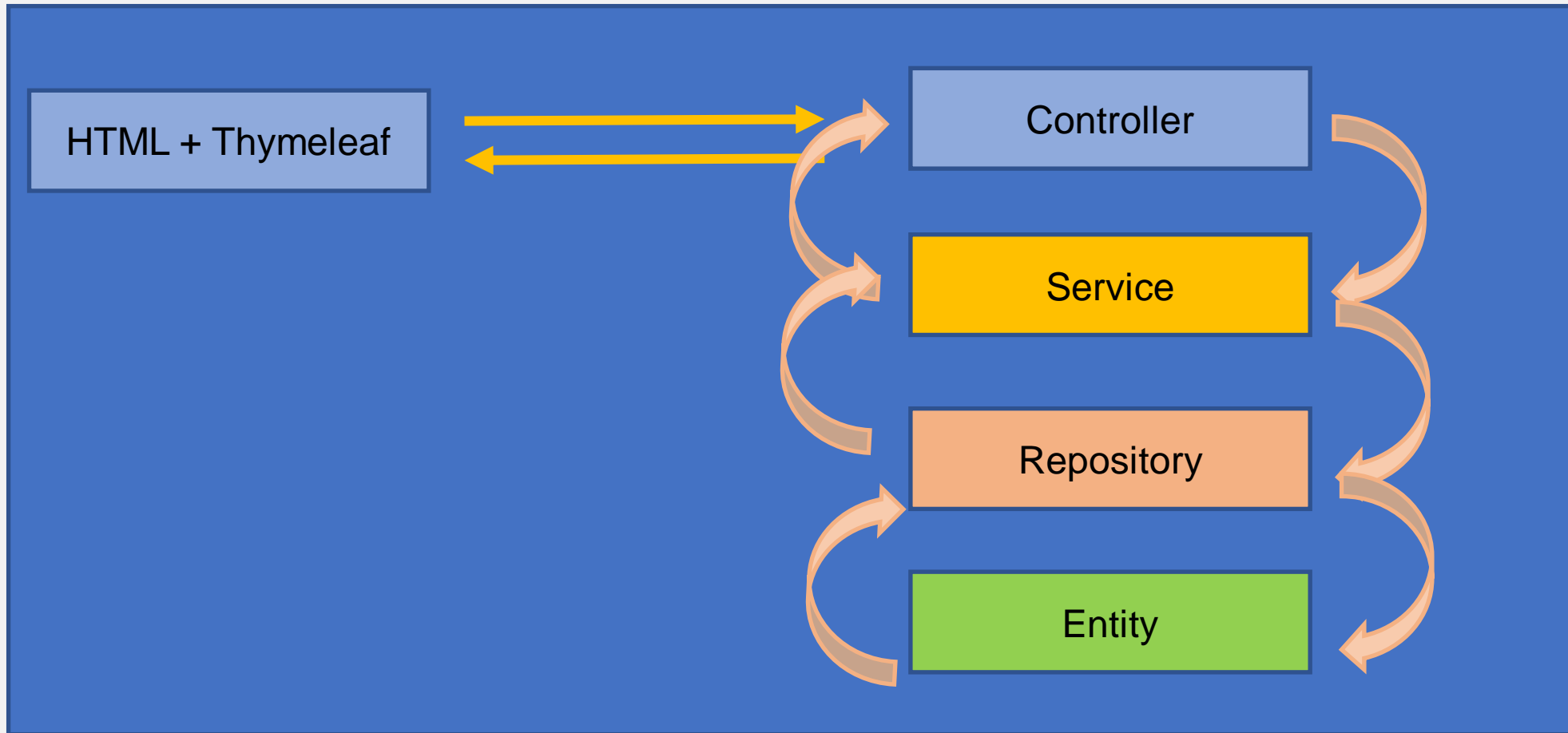
Modelo de Dados



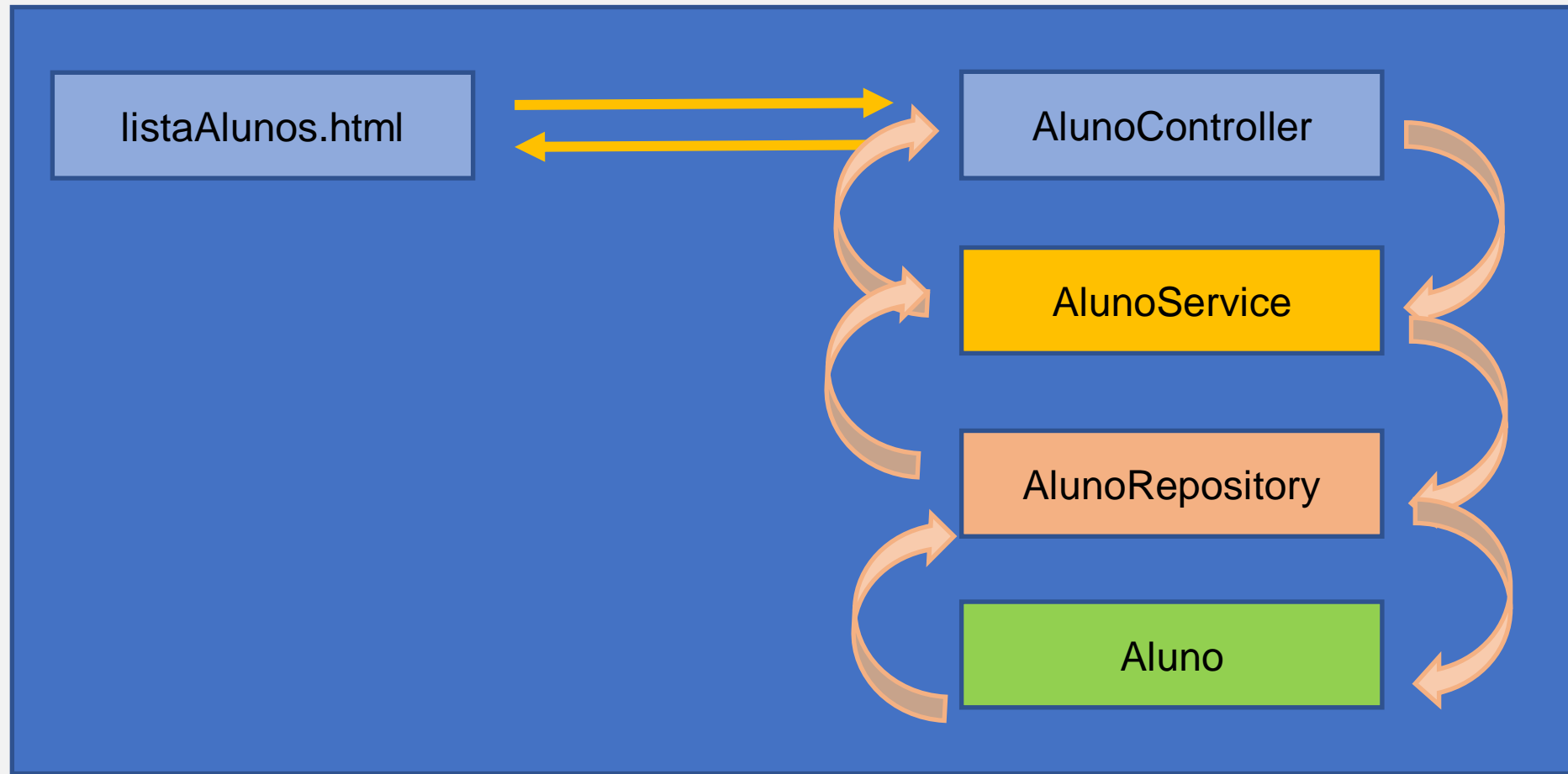
Camadas da aplicação

Prof. Marcelo Estruc

Camadas da Aplicação



Camadas da Aplicação



Repository e Entity

Prof. Marcelo Estruc

Entity

Implementando um entity com Spring Boot

Repository

- **Repository** (em português, repositório) é um padrão de projeto descrito no livro *Domain-Driven Design* (DDD) de Eric Vans.
- É um conceito muito semelhante ao padrão de projeto DAO, já que seu foco também é a camada de persistência de dados de uma aplicação. Muitas vezes a implementação de um DAO acaba sendo tratada como um **Repository**, ou vice-versa, mas estas abordagens se diferem em alguns pontos.

Repository

- Os repositórios são parte da camada de negócios da aplicação, e fornecem objetos a outras camadas como as de controle ou visão.
- Outra particularidade do repositório é que ele não conhece a infraestrutura da aplicação, como o tipo de banco de dados, ou se uma conexão será por JDBC, ODBC ou mesmo se vai trabalhar com um framework de persistência.

Mão na massa



workspaceSenac - CarrinhoCompras/src/main/java/br/com/senac

File Edit Source Refactor Navigate Search Project Run

Project Explorer

- Spring Elements
- JAX-WS Web Services
- Java Resources
 - src/main/java
 - br.com.senac
 - br.com.senac.controller
 - br.com.senac.domain
 - br.com.senac.exception
 - br.com.senac.inicializacao
 - Init.java
 - Init
 - br.com.senac.repository
 - br.com.senac.service
 - src/main/resources
 - static
 - templates
 - application.properties
 - src/test/java
 - Libraries
- JavaScript Resources
- Deployed Resources
- src
- target
- mvnw
- mvnw.cmd
- pom.xml
- Concessionaria
 - src
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

New Java Class

Java Class

Source folder: PrimeiroProjeto/src/main/java Browse...

Package: br.com.senac.domain Browse...

☐ Enclosing type: Browse...

Name: Aluno

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

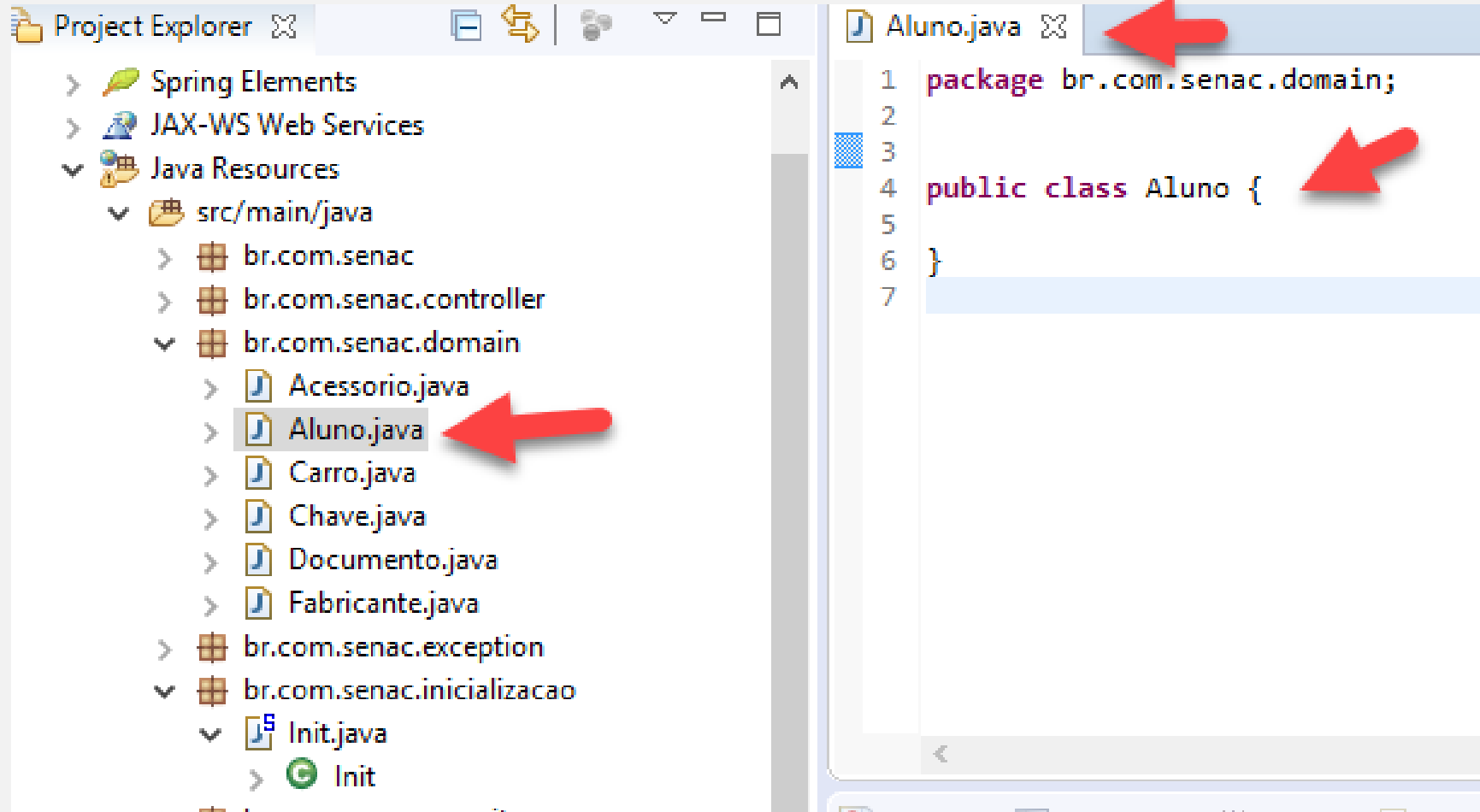
Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Finish Cancel



Para trabalharmos com transação de banco de dados. Iremos utilizar o Spring Data JPA.
Para isso devemos ter no pom.xml lib do spring jpa.

The screenshot shows an IDE with the Package Explorer on the left and the pom.xml file open in the editor. The Package Explorer shows the project structure for 'PrimeiroProjeto [boot]', including 'src/main/java', 'src/main/resources', 'src/test/java', 'JRE System Library [JavaSE-1.8]', 'Maven Dependencies', 'src', 'target', 'mvnw', 'mvnw.cmd', and 'pom.xml'. A red arrow points to 'pom.xml'. The editor shows the following XML code:

```
28 <dependency>
29   <groupId>org.springframework.boot</groupId>
30   <artifactId>spring-boot-starter-web</artifactId>
31 </dependency>
32
33 <dependency>
34   <groupId>org.springframework.boot</groupId>
35   <artifactId>spring-boot-starter-tomcat</artifactId>
36   <scope>provided</scope>
37 </dependency>
38 <dependency>
39   <groupId>org.springframework.boot</groupId>
40   <artifactId>spring-boot-starter-test</artifactId>
41   <scope>test</scope>
42 </dependency>
43 <dependency>
44   <groupId>org.springframework.boot</groupId>
45   <artifactId>spring-boot-starter-data-jpa</artifactId>
46 </dependency>
47 <dependency>
48   <groupId>com.h2database</groupId>
49   <artifactId>h2</artifactId>
50   <scope>runtime</scope>
51 </dependency>
52 <dependency>
53   <groupId>org.springframework.boot</groupId>
54   <artifactId>spring-boot-devtools</artifactId>
55 </dependency>
56 </dependencies>
```

Red arrows point to the following elements:

- The 'pom.xml' file in the Package Explorer.
- The 'spring-boot-starter-data-jpa' dependency entry (lines 43-46), which is highlighted with a red box.
- The 'spring-boot-starter-test' dependency entry (lines 38-42).
- The 'h2' dependency entry (lines 47-51).
- The 'spring-boot-devtools' dependency entry (lines 52-55).

```
1 package br.com.senac.domain;
2 import javax.persistence.Entity;
3 import javax.persistence.GeneratedValue;
4 import javax.persistence.GenerationType;
5 import javax.persistence.Id;
6
7 @Entity
8 public class Aluno {
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private Integer id;
13
14     private String nome;
15
16
17     public Integer getId() {
18         return id;
19     }
20
21
22     public void setId(Integer id) {
23         this.id = id;
24     }
25
26
27     public String getNome() {
28         return nome;
29     }
30
31
32     public void setNome(String nome) {
33         this.nome = nome;
34     }
35
36
37 }
```

Irá gerar um valor para a coluna ID

Informe que é representa uma tabela

Informe que é a chave primaria

O valor da coluna será preenchido conforme o auto incremento

Ir  gerar um valor para a coluna ID

GenerationType.IDENTITY	Informamos ao provedor de persist�ncia que os valores a serem atribu�dos ao identificador �nico ser�o gerados pela coluna de auto incremento do banco de dados. Assim, um valor para o identificador � gerado para cada registro inserido no banco. Alguns bancos de dados podem n�o suportar essa op��o.
GenerationType.SEQUENCE	Informamos ao provedor de persist�ncia que os valores ser�o gerados a partir de uma sequence. Caso n�o seja especificado um nome para a sequence, ser� utilizada uma sequence padr�o, a qual ser� global, para todas as entidades. Caso uma sequence seja especificada, o provedor passar� a adotar essa sequence para cria��o das chaves prim�rias. Alguns bancos de dados podem n�o suportar essa op��o.
GenerationType.TABLE	Com a op��o TABLE � necess�rio criar uma tabela para gerenciar as chaves prim�rias. Por causa da sobrecarga de consultas necess�rias para manter a tabela atualizada, essa op��o � pouco recomendada.

Aluno.java

```
1 package br.com.senac.domain;
2 import java.io.Serializable;
3
4 import javax.persistence.Column;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 @Entity
11 public class Aluno implements Serializable {
12
13     private static final long serialVersionUID = -918814843180942228L;
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Integer id;
18
19     @Column(name="nomeAluno")
20     private String nome;
21
22
23     public Integer getId() {
24         return id;
25     }
26
27
28     public void setId(Integer id) {
29         this.id = id;
30     }
31 }
```

Utilizado para persistir dados na base de dados

Utilizado para indicar o nome da
coluna na tabela. Se não tiver
especificado, será utilizado o nome
do atributo

Vamos Criar a Camada de Repositório

The screenshot shows the Spring Tool Suite IDE with a project named 'workspaceSenac'. The Project Explorer on the left displays the project structure, with the package 'br.com.senac.repository' highlighted. A red arrow points to this package. The 'New Java Interface' dialog is open, with the title 'Java Interface' circled in red. The dialog fields are as follows:

- Source folder: `PrimeiroProjeto/src/main/java`
- Package: `br.com.senac.repository`
- Enclosing type: (empty)
- Name: `AlunoRepository`
- Modifiers: ☒ public, ☐ package, ☐ private, ☐ protected
- Extended interfaces: (empty)
- Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Red arrows point to the 'Package' field and the 'Name' field. The 'Finish' and 'Cancel' buttons are at the bottom right. The bottom of the image shows a console window with log output.

```
2019-07-19 17:53:16.007 INFO 11136 --- [nio-9000-exec-1] o.h.h.i.QueryTranslator
2019-07-19 18:27:18.689 INFO 11136 --- [on(2)-127.0.0.1] inMXBeanRegistrar$Spring
2019-07-19 18:27:18.689 INFO 11136 --- [on(2)-127.0.0.1] ConfigServletWebServerA
2019-07-19 18:27:18.693 INFO 11136 --- [on(2)-127.0.0.1] o.s.j.e.a.AnnotationMBE
2019-07-19 18:27:18.693 INFO 11136 --- [on(2)-127.0.0.1] o.s.j.e.a.AnnotationMBE
2019-07-19 18:27:18.697 INFO 11136 --- [on(2)-127.0.0.1] j.LocalContainerEntityM
2019-07-19 18:27:18.697 INFO 11136 --- [on(2)-127.0.0.1] .SchemaDropperImpl$Dela
2019-07-19 18:27:18.714 INFO 11136 --- [on(2)-127.0.0.1] com.zaxxer.hikari.Hikar
```

AlunoRepository.java

```
1 package br.com.senac.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import br.com.senac.domain.Aluno;
7
8 @Repository ← Indica que é uma Interface que representa um repositório
9 public interface AlunoRepository extends JpaRepository<Aluno, Integer>{
10
11
12
13
14 }
```

<https://docs.spring.io/spring-data/jpa/docs/current/api/org.springframework.data.jpa.repository.JpaRepository.html>

Possui diversos métodos implementados como: buscar todos objetos, buscar por id, incluir, excluir

Service

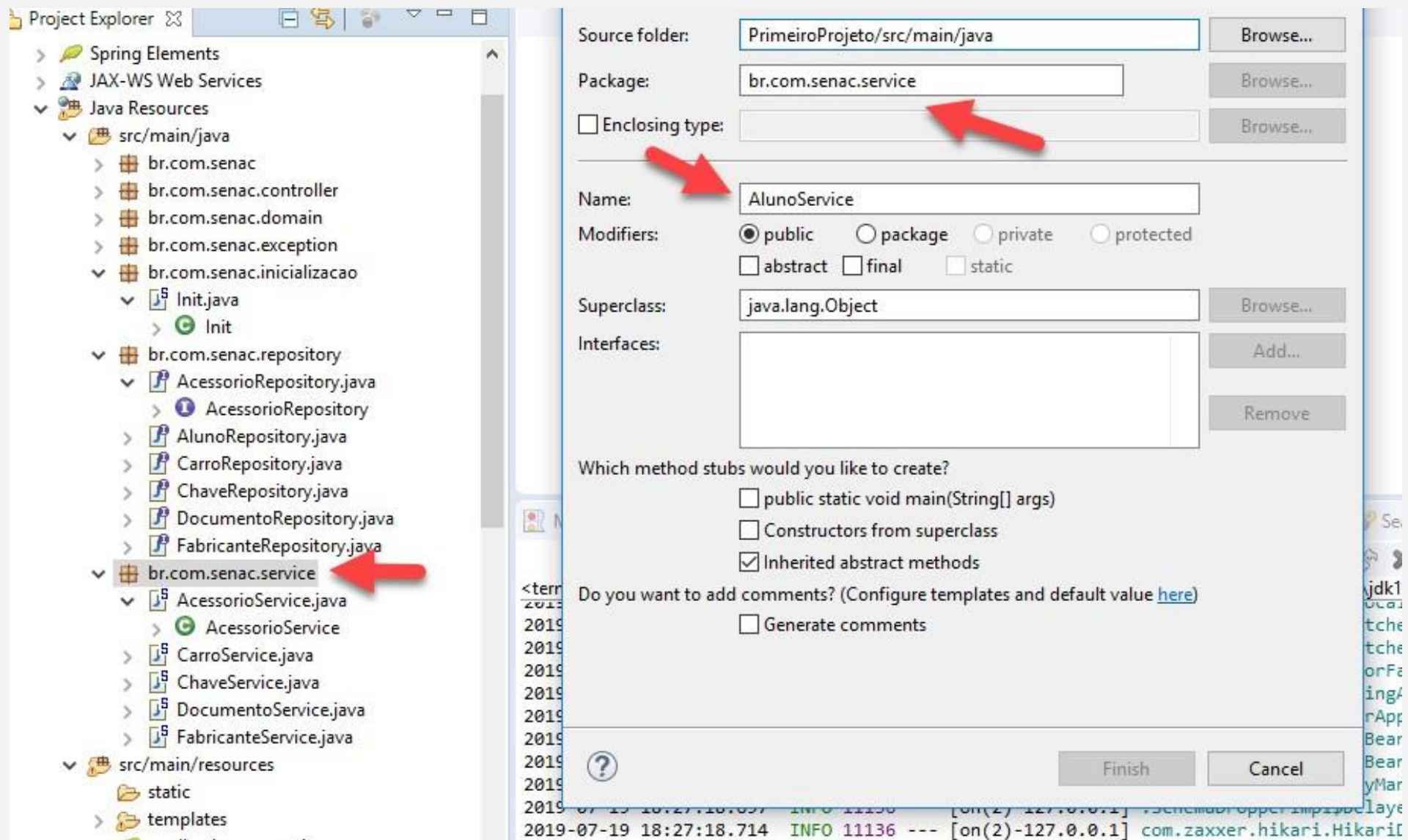
Prof. Marcelo Estruc

Service

- Serviços são classes que manipulam as regras de negócio para prover uma determinada funcionalidade.
- Têm anotação específica no universo Spring.

Mão na massa





Criando a classe Service


```

1 package br.com.senac.service;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import br.com.senac.domain.Aluno;
9 import br.com.senac.repository.AlunoRepository;
10
11 @Service
12 public class AlunoService {
13
14     @Autowired
15     AlunoRepository repoAluno;
16
17     public List<Aluno> buscarTodosAlunos(){
18         return repoAluno.findAll();
19     }
20
21     public Aluno salvar(Aluno aluno) {
22         return repoAluno.save(aluno);
23     }
24 }
25
26 }
27

```

Indica uma classe de Serviço

Indica que quando precisarmos utilizar AlunoRepository, ela vai estar instanciada

Método do JpaRepository que busca todos os dados da tabela

Método do JpaRepository que salva um objeto aluno na tabela

Classe de Inicialização

Prof. Marcelo Estruc

Classe de Inicialização

Mão na massa



Criando uma classe de Inicialização

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays the project structure. The package `br.com.senac.inicializacao` is highlighted with a red arrow. On the right, the 'New Java Class' dialog is open. It shows the source folder as `PrimeiroProjeto/src/main/java` and the package as `br.com.senac.inicializacao`. The class name is `Init`, and the modifiers are set to `public`. The superclass is `java.lang.Object`. The 'Which method stubs would you like to create?' section has `Inherited abstract methods` checked. The 'Do you want to add comments?' section has `Generate comments` unchecked. The 'Finish' button is highlighted.

workspaceSenac - CarrinhoCompras/src/main/java/br/com/senac

File Edit Source Refactor Navigate Search Project Run

Project Explorer

- Spring Elements
- JAX-WS Web Services
- Java Resources
 - src/main/java
 - br.com.senac
 - br.com.senac.controller
 - br.com.senac.domain
 - br.com.senac.exception
 - br.com.senac.inicializacao
 - br.com.senac.repository
 - AcessorioRepository.java
 - AcessorioRepository
 - AlunoRepository.java
 - CarroRepository.java
 - ChaveRepository.java
 - DocumentoRepository.java
 - FabricanteRepository.java
 - br.com.senac.service
 - AcessorioService.java
 - AcessorioService
 - AlunoService.java
 - CarroService.java
 - ChaveService.java
 - DocumentoService.java
 - FabricanteService.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - src/test/java
 - Libraries

```
1 package br.com.senac.inicializacao;
2 import org.springframework.beans.factory.annotation.Autowired;
3 import org.springframework.context.ApplicationListener;
4 import org.springframework.context.event.ContextRefreshedEvent;
5 import org.springframework.stereotype.Component;
6
7 import br.com.senac.domain.Aluno;
8 import br.com.senac.service.AlunoService;
9
10 @Component
11 public class Init implements ApplicationListener<ContextRefreshedEvent> {
12
13     @Autowired
14     AlunoService alunoService;
15
16     @Override
17     public void onApplicationEvent(ContextRefreshedEvent event) {
18
19         Aluno aluno1 = new Aluno();
20         aluno1.setNome("Lucas");
21         alunoService.salvar(aluno1);
22
23
24         Aluno aluno2 = new Aluno();
25         aluno2.setNome("Arthur");
26         alunoService.salvar(aluno2);
27
28         Aluno aluno3 = new Aluno();
29         aluno3.setNome("Jose");
30         alunoService.salvar(aluno3);
31
32     }
33 }
34
35
36
```

Vamos listar os alunos incluídos no console

```
@Override
public void onApplicationEvent(ContextRefreshedEvent event) {

    Aluno aluno1 = new Aluno();
    aluno1.setNome("Lucas");
    alunoService.salvar(aluno1);

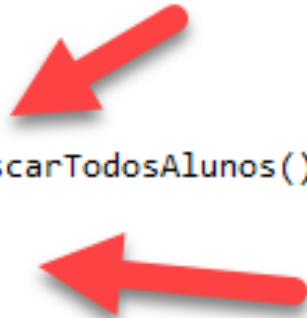
    Aluno aluno2 = new Aluno();
    aluno2.setNome("Arthur");
    alunoService.salvar(aluno2);

    Aluno aluno3 = new Aluno();
    aluno3.setNome("Jose");
    alunoService.salvar(aluno3);

    List<Aluno> listaAlunos = alunoService.buscarTodosAlunos();

    for(Aluno aluno:listaAlunos) {
        System.out.println(aluno.getNome());
    }

}
```



Executando aplicação

Prof. Marcelo Estruc

Mão na massa



Vamos executar o projeto

workspaceSenac - CarrinhoCompras/src/main/java/br/com/senac/inicializacao/Init.java - Spring Tool Suite

File Edit Source

Project Explorer

CarrinhoCo

Deploy

Spring B

JAX-WS

Java Res

src/

Build Path

Refactor

Import

Export

Refresh

Close Project

Close Unrelated Projects

Show in Remote Systems view

Validate

Run As

Debug As

Profile As

Restore from Local History...

Java EE Tools

Maven

Team

Compare With

Configure

Source

Spring Tools

Init.java

```
7 import org.springframework.stereotype
8
9 import br.com.senac.domain.Aluno;
10 import br.com.senac.service.AlunoServ
11
12 @Component
13 public class Init implements Applicat
14
15 @Autowired
16 AlunoService alunoService;
17
18 @Override
19 public void onApplicationEvent(Co
20
21 Aluno aluno1 = new Aluno();
22 aluno1.setNome("Lucas");
23 alunoService.salvar(aluno1);
24
```

Markers Properties Servers Snippets

1 Run on Server Alt+Shift+X, R

2 Java Application Alt+Shift+X, J

3 JUnit Test Alt+Shift+X, T

4 Maven build Alt+Shift+X, M

5 Maven build...

6 Maven clean

7 Maven generate-sources

8 Maven install


9 Maven test

Spring Boot App Alt+Shift+X, B

Spring Devtools Client

Vamos executar o projeto

```
2019-07-19 20:03:33.858 INFO 9296 --- [ r
2019-07-19 20:03:33.905 INFO 9296 --- [ r
2019-07-19 20:03:33.907 INFO 9296 --- [ r
2019-07-19 20:03:33.916 INFO 9296 --- [ r
2019-07-19 20:03:34.037 INFO 9296 --- [ r
Lucas
Arthur
Jose
2019-07-19 20:03:34.160 INFO 9296 --- [ r
2019-07-19 20:03:34.165 INFO 9296 --- [ r
```



<

Verificando no H2

← → ↻ 🏠 🟢 localhost:8080/h2-console/

English ▼ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▼

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:db

User Name: sa


Password:

Connect Test Connection


Vamos parar a aplicação

Auto commit Max rows: 1000

jdbc:h2:mem:db

- + ACESSORIO
- ALUNO 
 - + ID
 - + NOME_ALUNO
 - + Indexes
- + CARRO
- + CARRO_ACESSORIO
- + CHAVE
- + DOCUMENTO
- + FABRICANTE
- + INFORMATION_SCHEMA
- + Sequences
- + Users
- i H2 1.4.197 (2018-03-18)


Run Run Selected Auto complete Clear SC

SELECT * FROM ALUNO | 

SELECT * FROM ALUNO;

ID	NOME_ALUNO
1	Lucas
2	Arthur
3	Jose

(3 rows, 24 ms)

Edit 

Controller - Listagem

Prof. Marcelo Estruc

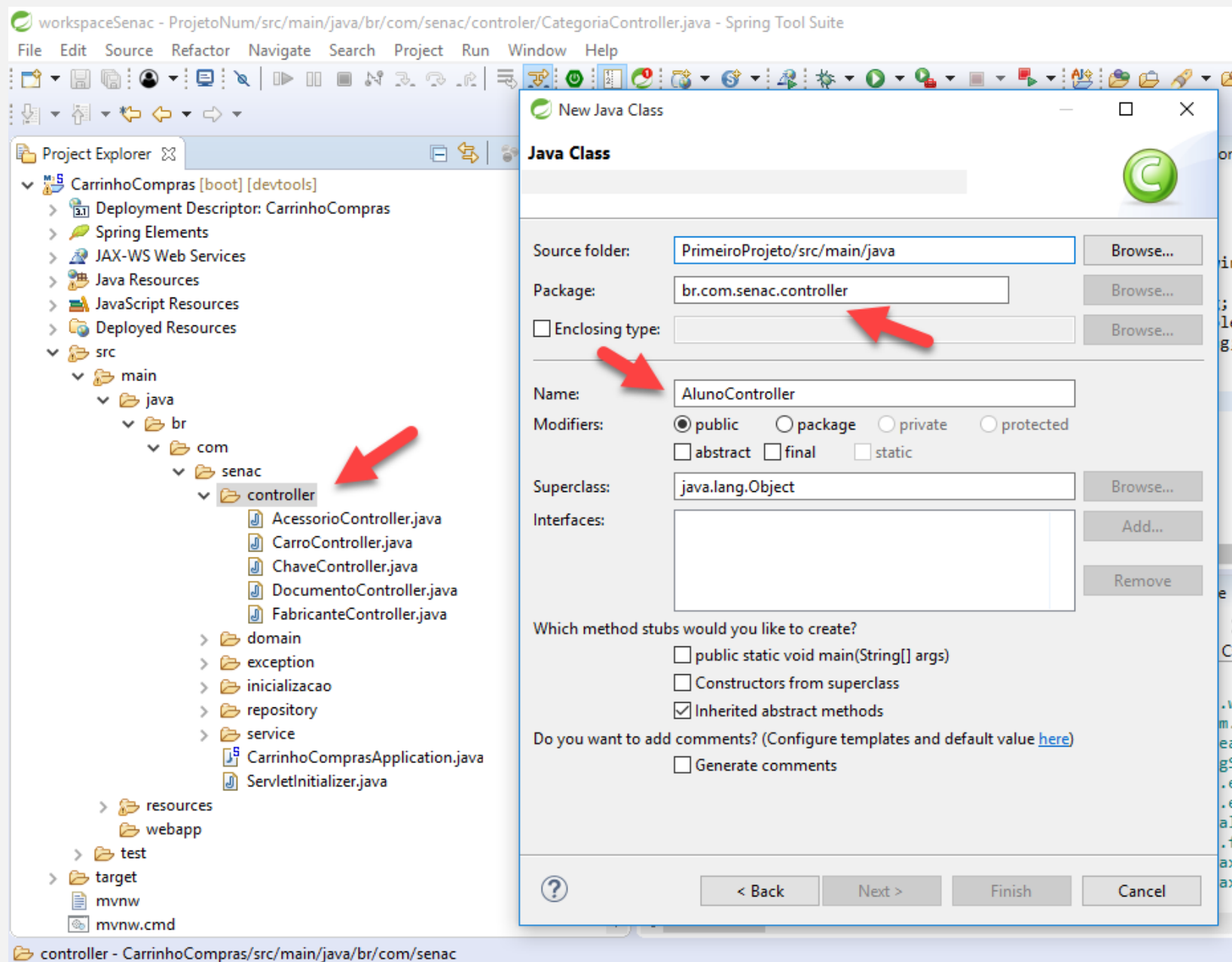
Controller

- O controlador serve como um intermediário que organiza os eventos da interface com usuário e os direciona para a camada de modelo, assim, torna-se possível um reaproveitamento da camada de modelo em outros sistemas já que não existe dependência entre a visualização e o modelo.
- Deste modo, toda requisição criada pelo usuário deve passar pelo **controller**, e este então se comunica com o **model**. Se o model gerar uma resposta para essas requisições, ele enviará as respostas ao **controller** que por sua vez repassa à camada **view**.

Mão na massa



Vamos implementar a classe de controller



```
1 package br.com.senac.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.servlet.ModelAndView;
8
9 import br.com.senac.service.AlunoService;
10
11 @Controller
12 @RequestMapping("aluno")
13 public class AlunoController {
14
15
16     @Autowired
17     private AlunoService alunoService;
18
19     @GetMapping("/listarAlunos")
20     public ModelAndView listaTodosAlunos() {
21         ModelAndView mv = new ModelAndView("aluno/paginaListaAlunos");
22         mv.addObject("alunos", alunoService.buscarTodosAlunos());
23         return mv;
24     }
25
26
27 }
```

Thymeleaf

Prof. Marcelo Estruc

Telas com o Thymeleaf

- O Thymeleaf é uma template engine para projetos Java que facilita a criação de páginas HTML. Sendo assim, ele serve para gerar páginas HTML no lado servidor de forma dinâmica, permitindo a troca de informações entre o código Java e as página HTML, de tal maneira ele garante que o desenvolvedor consiga criar *templates* de forma mais fácil para suas aplicações.
- O projeto pode ser acessado no seguinte link: <https://www.thymeleaf.org/>

Telas com o Thymeleaf

- Antes de mais nada a principal funcionalidade de um template engine é permitir que linguagens de programação possam ser incorporadas em páginas HTML. de tal forma, uma *template engine* permite que os programadores possam utilizar estruturas de condição, estruturas de repetição, herança e diversos outros recursos presentes apenas nas linguagens de programação em páginas HTML.
- Analogamente o **Thymeleaf** não é diferente, ele permite que desenvolvedores incorporem código Java em páginas HTML e também utilizem as principais características da linguagem em seus *templates*.

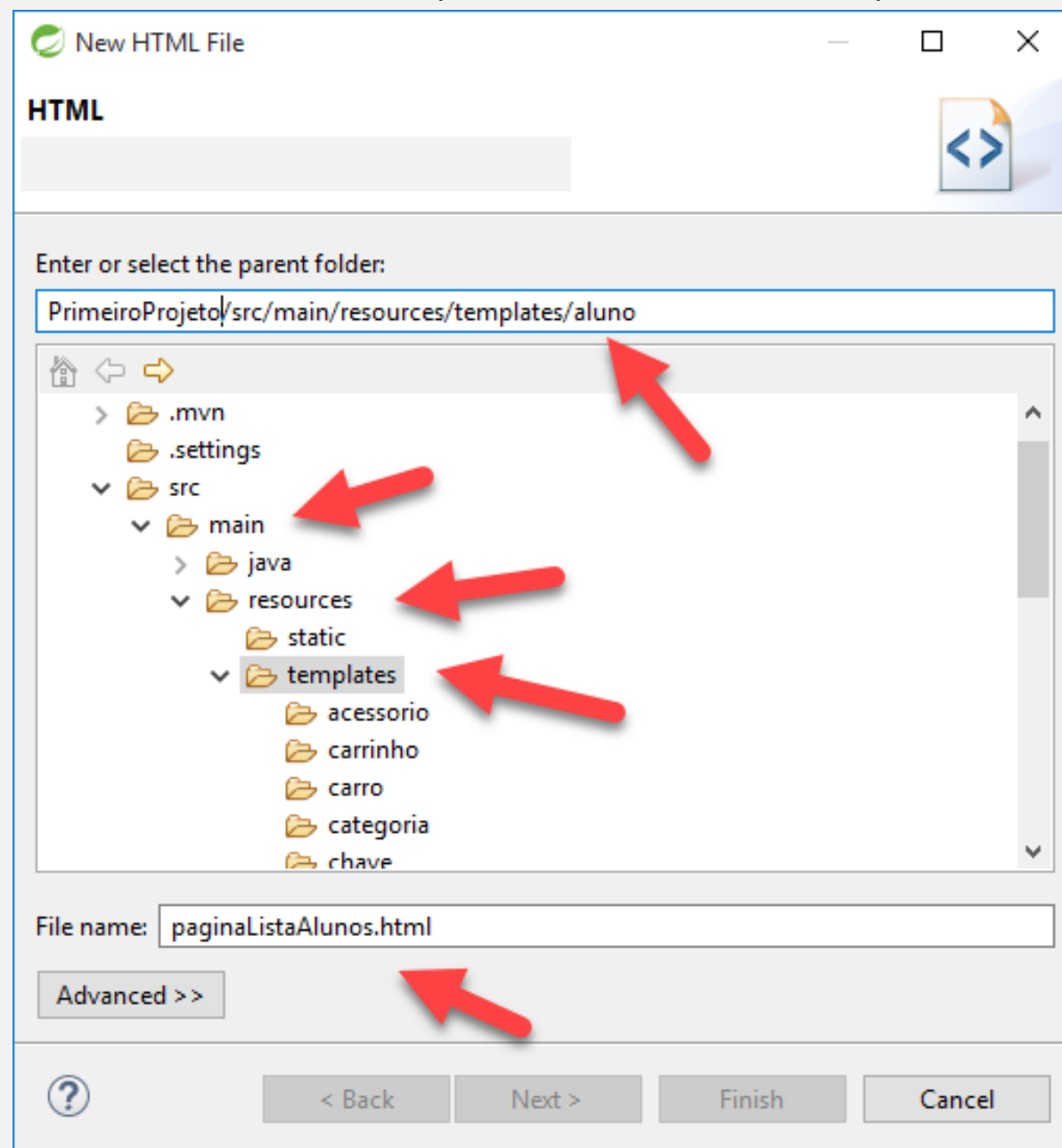
Telas com o Thymeleaf

- Dentre diversas características, podemos citar as principais:
 - Permite o uso de estruturas de condição e repetição em páginas HTML;
 - Com o **Thymeleaf** é possível utilizar herança de layouts, garantindo uma estrutura com um maior reaproveitamento de código;
 - Permite exibir o conteúdo de variáveis Java em páginas HTML;
 - Sistema de fragmentos de *templates*, dentre outros.

Mão na massa



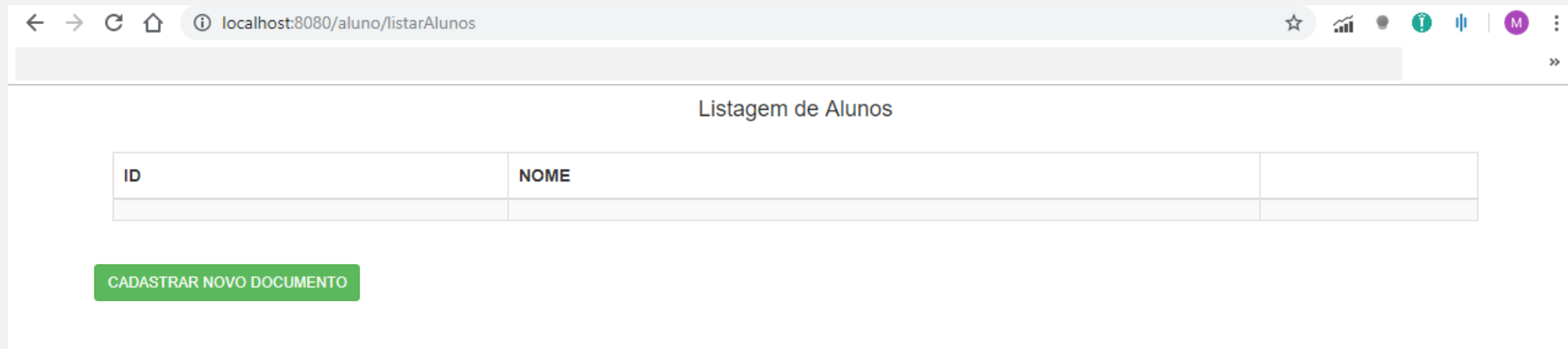
Vamos criar um html na pasta src/main/resources/templates



Implementando a pagina de listagem de alunos

```
paginaListaAlunos.html
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <title>Listagem de Alunos</title>
5     <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet"></link>
6     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
7 </head>
8 <body>
9     <div class="container">
10         <div class="panel-default">
11             <div class="text-center">
12                 <h1 class="h4 text-gray-900 mb-4">Listagem de Alunos</h1>
13             </div>
14             <div class="panel-body">
15                 <div class="table-responsive">
16                     <table class="table table-sm table-striped table-hover table-bordered">
17                         <thead>
18                             <tr>
19                                 <th>ID</th>
20                                 <th>NOME</th>
21                                 <th></th>
22                             </tr>
23                         </thead>
24                         <tbody>
25                             <tr>
26                                 <td></td>
27                                 <td></td>
28                                 <td></td>
29                             </tr>
30                         </tbody>
31                     </table>
32                 </div>
33             </div>
34         </div>
35         <a class="btn btn-sm btn-success">CADASTRAR NOVO DOCUMENTO</a>
36     </div>
37 </body>
38 </html>
```

Execute a aplicação



```
<tbody>
  <tr th:each="aluno : ${alunos}">
    <td th:text="${aluno.id}"></td>
    <td th:text="${aluno.nome}"></td>
    <td>
      <div class="btn-group pull-right">
        <a class="btn btn-sm btn-primary" >ALTERAR</a>
        <a class="delete btn btn-sm btn-danger" >EXCLUIR</a>
      </div>
    </td>
  </tr>
</tbody>
```

Vamos parar a aplicação

Listagem de Alunos

ID	NOME	
1	Lucas	ALTERAR EXCLUIR
2	Arthur	ALTERAR EXCLUIR
3	Jose	ALTERAR EXCLUIR

CADASTRAR NOVO DOCUMENTO

Controller - Cadastrar

Prof. Marcelo Estruc

Mão na massa

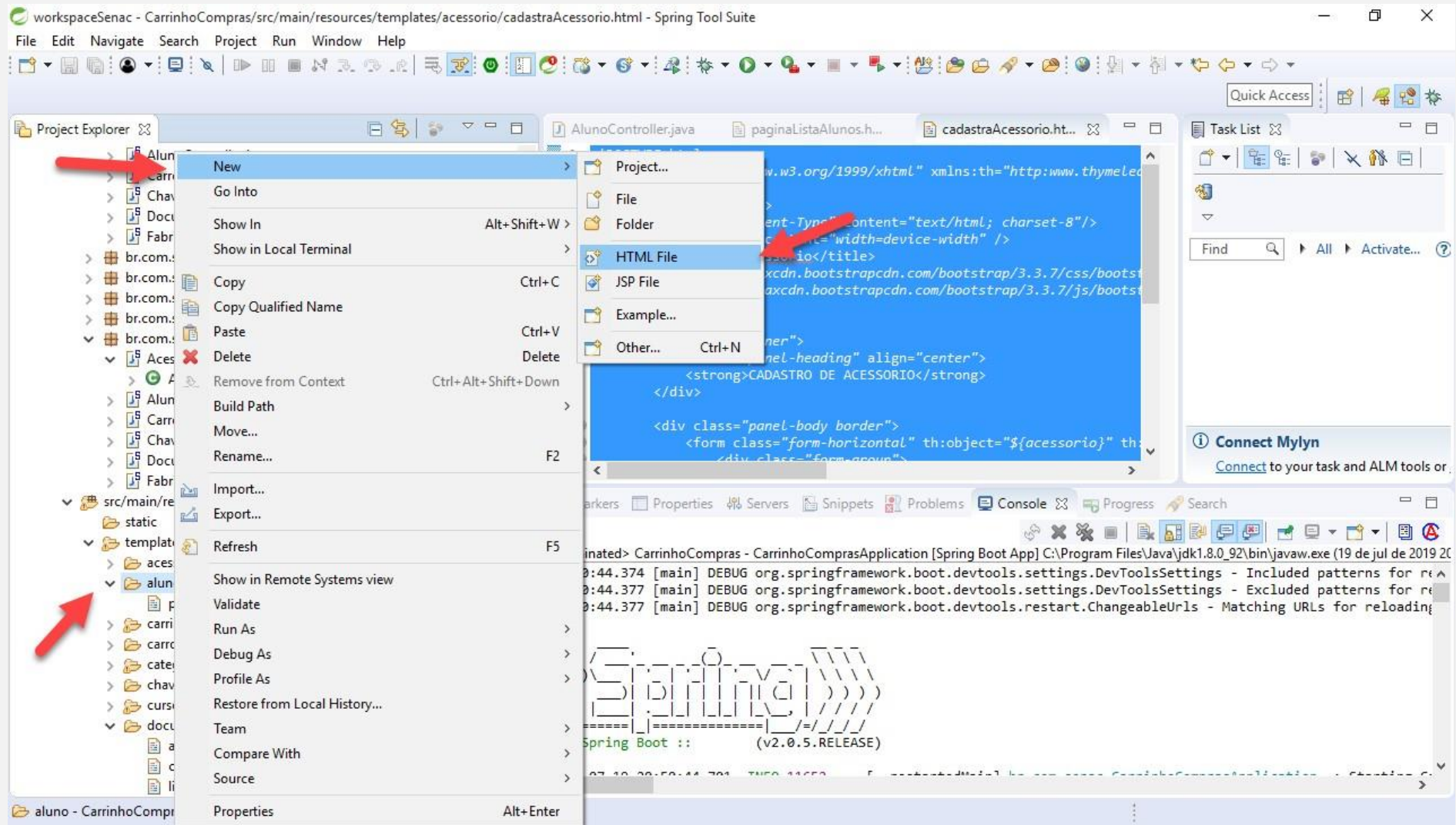


Vamos implementar o cadastrar

```
AlunoController.java
1 package br.com.senac.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7
8
9
10
11
12
13 @Controller
14 @RequestMapping("aluno")
15 public class AlunoController {
16
17
18     @Autowired
19     private AlunoService alunoService;
20
21
22     @GetMapping("/listarAlunos")
23     public ModelAndView listarTodosAlunos() {
24         ModelAndView mv = new ModelAndView("aluno/paginaListaAlunos");
25         mv.addObject("alunos", alunoService.buscarTodosAlunos());
26         return mv;
27     }
28
29     @GetMapping("/cadastrar")
30     public ModelAndView cadastrarAluno() {
31         ModelAndView mv = new ModelAndView("aluno/cadastreAluno");
32         mv.addObject("aluno", new Aluno());
33         return mv;
34     }
35
36     @PostMapping("/salvar")
37     public ModelAndView salvarAluno(Aluno aluno) {
38         alunoService.salvar(aluno);
39         return listarTodosAlunos();
40     }
41
42 }
```

```
7 </head>
8 <body>
9   <div class="container">
10     <div class="panel-default">
11       <div class="text-center">
12         <h1 class="h4 text-gray-900 mb-4">Listagem de Alunos</h1>
13       </div>
14       <div class="panel-body">
15         <div class="table-responsive">
16           <table class="table table-sm table-striped table-hover table-bordered">
17             <thead>
18               <tr>
19                 <th>ID</th>
20                 <th>NOME</th>
21                 <th></th>
22               </tr>
23             </thead>
24             <tbody>
25               <tr th:each="aluno : ${alunos}">
26                 <td th:text="${aluno.id}"></td>
27                 <td th:text="${aluno.nome}"></td>
28                 <td>
29                   <div class="btn-group pull-right">
30                     <a class="btn btn-sm btn-primary">ALTERAR</a>
31                     <a class="delete btn btn-sm btn-danger">EXCLUIR</a>
32                   </div>
33                 </td>
34               </tr>
35             </tbody>
36           </table>
37         </div>
38       </div>
39     </div>
40     <a class="btn btn-sm btn-success" th:href="@{/aluno/cadastrar}">CADASTRAR NOVO ALUNO</a>
41   </div>
42 </body>
43 </html>
```

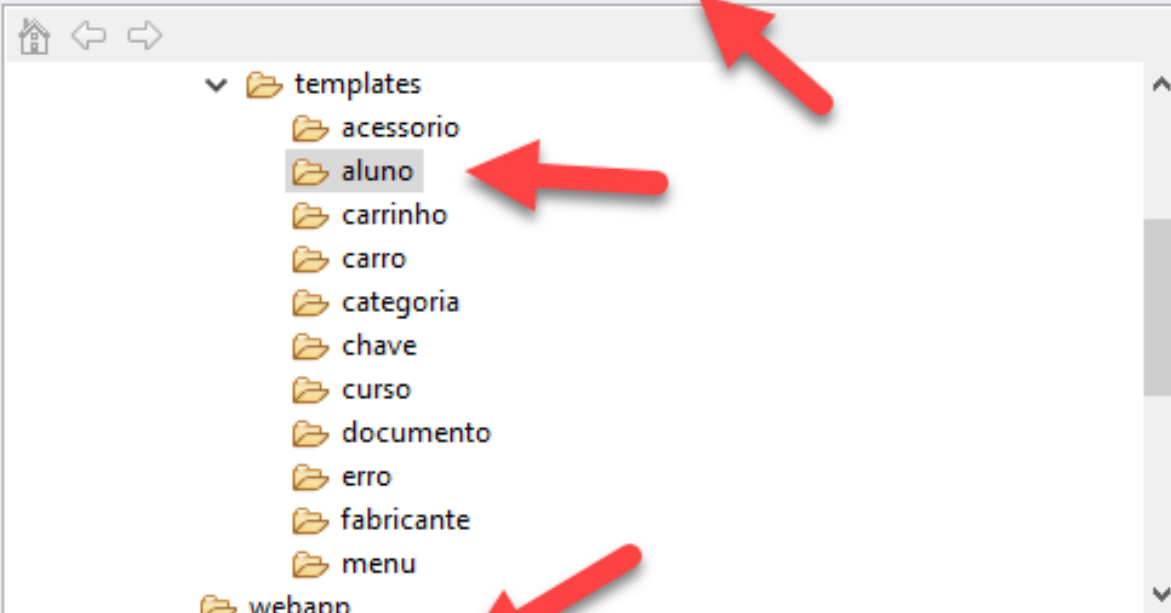






Enter or select the parent folder:

PrimeiroProjeto/src/main/resources/templates/aluno



File name: cadastraAluno.html

Advanced >>



< Back

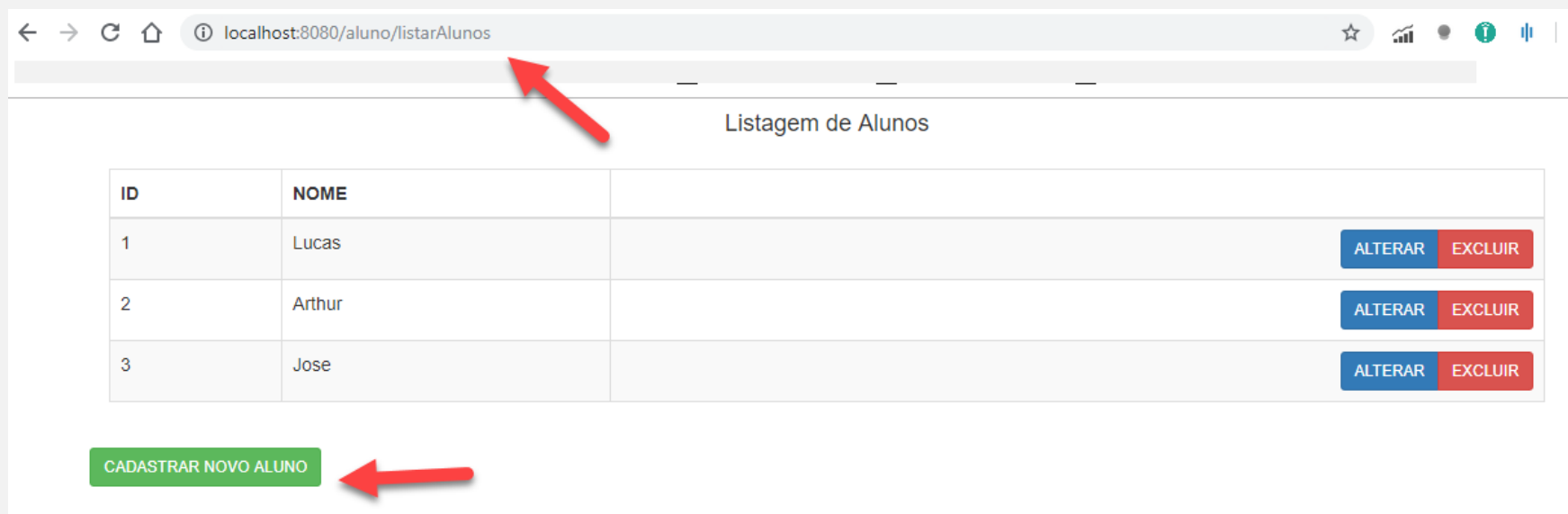
Next >

Finish

Cancel

```
AlunoController.java  paginaListaAlunos.html  cadastraAluno.html ✕
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="UTF-8"/>
5 <meta http-equiv="Content-Type" content="text/html; charset-8"/>
6 <meta name="viewport" content="width=device-width" />
7 <title>Cadastro de Aluno</title>
8 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet"></link>
9 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12 <div class="container">
13 <div class="panel-heading" align="center">
14 <strong>CADASTRO DE ALUNO</strong>
15 </div>
16
17 <div class="panel-body border">
18 <form class="form-horizontal" th:object="${aluno}" th:action="@{/aluno/salvar}" method="POST" style="margin: 10px">
19 <div class="form-group">
20 <fieldset>
21 <div class="form-group row">
22 <label>NOME</label>
23 <input type="text" class="form-control input-sm" th:field="*{nome}" autofocus="autofocus"
24 placeholder="Informe o nome" maxlength="50"/>
25 </div>
26 </fieldset>
27 </div>
28 <div align="center">
29 <button type="submit" class="btn btn-sm btn-success">SALVAR</button>
30 <a th:href="@{/aluno/listarAlunos}" class="btn btn-sm btn-default">CANCELAR</a>
31 </div>
32 </form>
33 </div>
34 </div>
35 </body>
36 </html>
```

Execute a aplicação



CADASTRO DE ALUNO

NOME

Guilherme



SALVAR

CANCELAR

Listagem de Alunos

ID	NOME	
1	Lucas	ALTERAR EXCLUIR
2	Arthur	ALTERAR EXCLUIR
3	Jose	ALTERAR EXCLUIR
4	Guilherme	ALTERAR EXCLUIR

[CADASTRAR NOVO ALUNO](#)

Controller – Alterar

Prof. Marcelo Estruc

Mão na massa



Busca por ID. Iremos utilizar quando for alterar um Aluno

```
AlunoService.java
1 package br.com.senac.service;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import br.com.senac.domain.Aluno;
10 import br.com.senac.repository.AlunoRepository;
11 import javassist.tools.rmi.ObjectNotFoundException;
12
13 @Service
14 public class AlunoService {
15
16
17     @Autowired
18     AlunoRepository repoAluno;
19
20     public List<Aluno> buscarTodosAlunos(){
21         return repoAluno.findAll();
22     }
23
24     public Aluno salvar(Aluno aluno) {
25         return repoAluno.save(aluno);
26     }
27
28     public Aluno buscaPorID(Integer id) throws ObjectNotFoundException {
29         Optional<Aluno> aluno = repoAluno.findById(id);
30         return aluno.orElseThrow(() -> new ObjectNotFoundException("Aluno não encontrado. id: " + id ));
31     }
32
33
34
35 }
36
```

```
paginaListaAlunos.html AlunoService.java AlunoController.java DocumentoController.java alteraAluno.html listaDocumento.html
7 </head>
8 <body>
9   <div class="container">
10     <div class="panel-default">
11       <div class="text-center">
12         <h1 class="h4 text-gray-900 mb-4">Listagem de Alunos</h1>
13       </div>
14       <div class="panel-body">
15         <div class="table-responsive">
16           <table class="table table-sm table-striped table-hover table-bordered">
17             <thead>
18               <tr>
19                 <th>ID</th>
20                 <th>NOME</th>
21                 <th></th>
22               </tr>
23             </thead>
24             <tbody>
25               <tr th:each="aluno : ${alunos}">
26                 <td th:text="${aluno.id}"></td>
27                 <td th:text="${aluno.nome}"></td>
28                 <td>
29                   <div class="btn-group pull-right">
30                     <a class="btn btn-sm btn-primary" th:href="@{/aluno/alterar/{idAluno}(idAluno=${aluno.id})}" >ALTERAR</a>
31                     <a class="delete btn btn-sm btn-danger" >EXCLUIR</a>
32                   </div>
33                 </td>
34               </tr>
35             </tbody>
36           </table>
37         </div>
38       </div>
39     </div>
40     <a class="btn btn-sm btn-success" th:href="@{/aluno/cadastrar}">CADASTRAR NOVO ALUNO</a>
41   </div>
42 </body>
43 </html>
```

```

17 public class AlunoController {
18
19
20     @Autowired
21     private AlunoService alunoService;
22
23     @GetMapping("/listarAlunos")
24     public ModelAndView listarTodosAlunos() {
25         ModelAndView mv = new ModelAndView("aluno/paginalistaAlunos");
26         mv.addObject("alunos", alunoService.buscarTodosAlunos());
27         return mv;
28     }
29
30     @GetMapping("/cadastrar")
31     public ModelAndView cadastrarAluno() {
32         ModelAndView mv = new ModelAndView("aluno/cadasttraAluno");
33         mv.addObject("aluno", new Aluno());
34         return mv;
35     }
36
37     @PostMapping("/salvar")
38     public ModelAndView salvarAluno(Aluno aluno) {
39         alunoService.salvar(aluno);
40         return listarTodosAlunos();
41     }
42
43
44     @GetMapping("/alterar/{id}")
45     public ModelAndView alterarAluno(@PathVariable("id") Integer idAluno) throws ObjectNotFoundException {
46         ModelAndView mv = new ModelAndView("aluno/alteraAluno");
47         mv.addObject("aluno", alunoService.buscaPorID(idAluno));
48         return mv;
49     }
50
51
52 }
53

```

Project Explorer

- br.com.senac.exception
- br.com.senac.inicializacao
- br.com.senac.repository
- br.com.senac.service
 - AcessorioService.java
 - AcessorioService
 - AlunoService.java
 - CarroService.java
 - ChaveService.java
 - DocumentoService.java
 - FabricanteService.java
- src/main/resources
 - static
 - templates
 - acessorio
 - aluno
 - alteraAluno.html
 - cadastraAluno.html
 - paginaListaAlunos.html
 - carrinho
 - carro
 - categoria
 - chave
 - curso
 - documento
 - erro
 - fabricante
 - menu
 - application.properties
 - src/test/java
 - Libraries
 - JavaScript Resources

Criar pagina

alteraAluno.html

```
1 <!DOCTYPE
2 <html >
3 <head>
4 <meta c
5 <meta f
6 <meta r
7 <title>
8 <link f
9 <script
10 </head>
11 <body>
12 <di
13
14
15
16
17
18
19
20
21
22
23
24
```


Markers

CarrinhoCompr
Hibernate:
select
alun
alun
from
alun
Hibernate:
select

```
alteraAluno.html AlunoController.java paginaListaAlunos.html AlunoService.java DocumentoController.java listaDocumento.html
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="UTF-8"/>
5 <meta http-equiv="Content-Type" content="text/html; charset-8"/>
6 <meta name="viewport" content="width=device-width" />
7 <title>Alterar Aluno</title>
8 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet"></link>
9 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12 <div class="container">
13 <div class="panel-heading" align="center"><strong>ALTERAÇÃO ALUNO</strong></div>
14 <div class="panel-body">
15 <form class="form-horizontal" th:object="${aluno}" th:action="@{/aluno/alterar}" method="POST" style="margin: 10px">
16 <div class="form-group">
17 <fieldset>
18 <div class="form-group row">
19 <label>ID</label>
20 <input type="text" class="form-control input-sm" id="id" th:field="*{id}" readOnly="readonly"/>
21 </div>
22 <div class="form-group row">
23 <label>NOME</label>
24 <input type="text" class="form-control input-sm" th:field="*{nome}" autofocus="autofocus" placeholder="Informe o nome" maxLength="50"/>
25 </div>
26 </fieldset>
27 </div>
28 <div align="center">
29 <button type="submit" class="btn btn-sm btn-success">SALVAR</button>
30 <a th:href="@{/aluno/listarAlunos}" class="btn btn-sm btn-default">CANCELAR</a>
31 </div>
32 </form>
33 </div>
34 </div>
35 </body>
36 </html>
```


AlunoService.java alteraAluno.html paginaListaAlunos.html DocumentoController.java listaDocumento.html

```
7 import org.springframework.stereotype.Service;
8
9 import br.com.senac.domain.Aluno;
10 import br.com.senac.repository.AlunoRepository;
11 import javassist.tools.rmi.ObjectNotFoundException;
12
13 @Service
14 public class AlunoService {
15
16
17     @Autowired
18     AlunoRepository repoAluno;
19
20     public List<Aluno> buscarTodosAlunos(){
21         return repoAluno.findAll();
22     }
23
24     public Aluno salvar(Aluno aluno) {
25         return repoAluno.save(aluno);
26     }
27
28     public Aluno buscaPorID(Integer id) throws ObjectNotFoundException {
29         Optional<Aluno> aluno = repoAluno.findById(id);
30         return aluno.orElseThrow(() -> new ObjectNotFoundException("Aluno não encontrado. id: " + id ));
31     }
32
33
34     public Aluno salvarAlteracao(Aluno alunoAlterado) throws ObjectNotFoundException {
35         Aluno aluno = buscaPorID(alunoAlterado.getId());
36         aluno.setId(alunoAlterado.getId());
37         aluno.setNome(alunoAlterado.getNome());
38         return salvar(aluno);
39     }
40
41
42
43 }
```



```

24 @GetMapping("/listarAlunos")
25 public ModelAndView listarTodosAlunos() {
26     ModelAndView mv = new ModelAndView("aluno/paginaListaAlunos");
27     mv.addObject("alunos", alunoService.buscarTodosAlunos());
28     return mv;
29 }
30
31 @GetMapping("/cadastrar")
32 public ModelAndView cadastrarAluno() {
33     ModelAndView mv = new ModelAndView("aluno/cadestraAluno");
34     mv.addObject("aluno", new Aluno());
35     return mv;
36 }
37
38 @PostMapping("/salvar")
39 public ModelAndView salvarAluno(Aluno aluno) {
40     alunoService.salvar(aluno);
41     return listarTodosAlunos();
42 }
43
44
45 @GetMapping("/alterar/{id}")
46 public ModelAndView alterarAluno(@PathVariable("id") Integer idAluno) throws ObjectNotFoundException {
47     ModelAndView mv = new ModelAndView("aluno/alteraAluno");
48     mv.addObject("aluno", alunoService.buscaPorID(idAluno));
49     return mv;
50 }
51
52
53 @PostMapping("/alterar")
54 public ModelAndView alterar(Aluno alunoAlterado) throws ObjectNotFoundException {
55     alunoService.salvarAlteracao(alunoAlterado);
56     return listarTodosAlunos();
57 }
58
59 }
60

```



ALTERAÇÃO ALUNO

ID

3

NOME

Jose



SALVAR

CANCELAR

Listagem de Alunos

ID	NOME	
1	Lucas	ALTERAR EXCLUIR
2	Arthur	ALTERAR EXCLUIR
3	Jose 2	ALTERAR EXCLUIR

CADASTRAR NOVO ALUNO

Controller – Excluir

Prof. Marcelo Estruc


Mão na massa



```

11 import javassist.tools.rmi.ObjectNotFoundException;
12
13 @Service
14 public class AlunoService {
15
16     @Autowired
17     AlunoRepository repoAluno;
18
19     public List<Aluno> buscarTodosAlunos(){
20         return repoAluno.findAll();
21     }
22
23     public Aluno salvar(Aluno aluno) {
24         return repoAluno.save(aluno);
25     }
26
27     public Aluno buscaPorID(Integer id) throws ObjectNotFoundException {
28         Optional<Aluno> aluno = repoAluno.findById(id);
29         return aluno.orElseThrow(() -> new ObjectNotFoundException("Aluno não encontrado. id: " + id ));
30     }
31
32     public Aluno salvarAlteracao(Aluno alunoAlterado) throws ObjectNotFoundException {
33         Aluno aluno = buscaPorID(alunoAlterado.getId());
34         aluno.setId(alunoAlterado.getId());
35         aluno.setNome(alunoAlterado.getNome());
36         return salvar(aluno);
37     }
38
39     public void excluir(Integer id) {
40         repoAluno.deleteById(id);
41     }
42
43 }
44
45
46
47 }

```



```
AlunoController.java AlunoService.java alteraAluno.html paginaListaAlunos.html DocumentoController.java
29     }
30
31     @GetMapping("/cadastrar")
32     public ModelAndView cadastrarAluno() {
33         ModelAndView mv = new ModelAndView("aluno/cadastreAluno");
34         mv.addObject("aluno", new Aluno());
35         return mv;
36     }
37
38     @PostMapping("/salvar")
39     public ModelAndView salvarAluno(Aluno aluno) {
40         alunoService.salvar(aluno);
41         return listaTodosAlunos();
42     }
43
44
45     @GetMapping("/alterar/{id}")
46     public ModelAndView alterarAluno(@PathVariable("id") Integer idAluno) throws ObjectNotFoundException {
47         ModelAndView mv = new ModelAndView("aluno/alteraAluno");
48         mv.addObject("aluno", alunoService.buscaPorID(idAluno));
49         return mv;
50     }
51
52
53     @PostMapping("/alterar")
54     public ModelAndView alterar(Aluno alunoAlterado) throws ObjectNotFoundException {
55         alunoService.salvarAlteracao(alunoAlterado);
56         return listaTodosAlunos();
57     }
58
59     @GetMapping("/excluir/{id}")
60     public ModelAndView excluirAluno(@PathVariable("id") Integer id) {
61         alunoService.excluir(id);
62         return listaTodosAlunos();
63     }
64
65 }
```



```
paginaListaAlunos.html AlunoController.java AlunoService.java alteraAluno.html DocumentoController.java listaDocumento.html DocumentoService.java
7 </head>
8 <body>
9   <div class="container">
10    <div class="panel-default">
11      <div class="text-center">
12        <h1 class="h4 text-gray-900 mb-4">Listagem de Alunos</h1>
13      </div>
14      <div class="panel-body">
15        <div class="table-responsive">
16          <table class="table table-sm table-striped table-hover table-bordered">
17            <thead>
18              <tr>
19                <th>ID</th>
20                <th>NOME</th>
21                <th></th>
22              </tr>
23            </thead>
24            <tbody>
25              <tr th:each="aluno : ${alunos}">
26                <td th:text="${aluno.id}"></td>
27                <td th:text="${aluno.nome}"></td>
28                <td>
29                  <div class="btn-group pull-right">
30                    <a class="btn btn-sm btn-primary" th:href="@{/aluno/alterar/{idAluno}(idAluno=${aluno.id})}" >ALTERAR</a>
31                    <a class="delete btn btn-sm btn-danger" th:href="@{/aluno/excluir/{idAluno}(idAluno=${aluno.id})}" >EXCLUIR</a>
32                  </div>
33                </td>
34              </tr>
35            </tbody>
36          </table>
37        </div>
38      </div>
39    </div>
40    <a class="btn btn-sm btn-success" th:href="@{/aluno/cadastrar}">CADASTRAR NOVO ALUNO</a>
41  </div>
42 </body>
43 </html>
```

stagem de Alunos

localhost:8080/aluno/listarAlunos

☆ 📶 ⓘ 🔊 | Pausada

Listagem de Alunos

ID	NOME	
1	Lucas	<div>ALTERAR EXCLUIR</div>
2	Arthur	<div>ALTERAR EXCLUIR</div>
3	Jose	<div>ALTERAR EXCLUIR</div>

CADASTRAR NOVO ALUNO

Listagem de Alunos

ID	NOME	
1	Lucas	<div>ALTERAREXCLUIR</div>
2	Arthur	<div>ALTERAREXCLUIR</div>

CADASTRAR NOVO ALUNO

Atividade para o Aluno

Prof. Marcelo Estruc

Atividade para o Aluno

O aluno deverá construir o gerenciamento de cursos.

Para isso comece criando o Entity, Repository e Service.

Verifique se todas as camadas estão funcionando corretamente utilizando a classe de inicialização.

Após essa verificação crie o Controller. Comece pelo listar, cadastrar, editar e excluir.