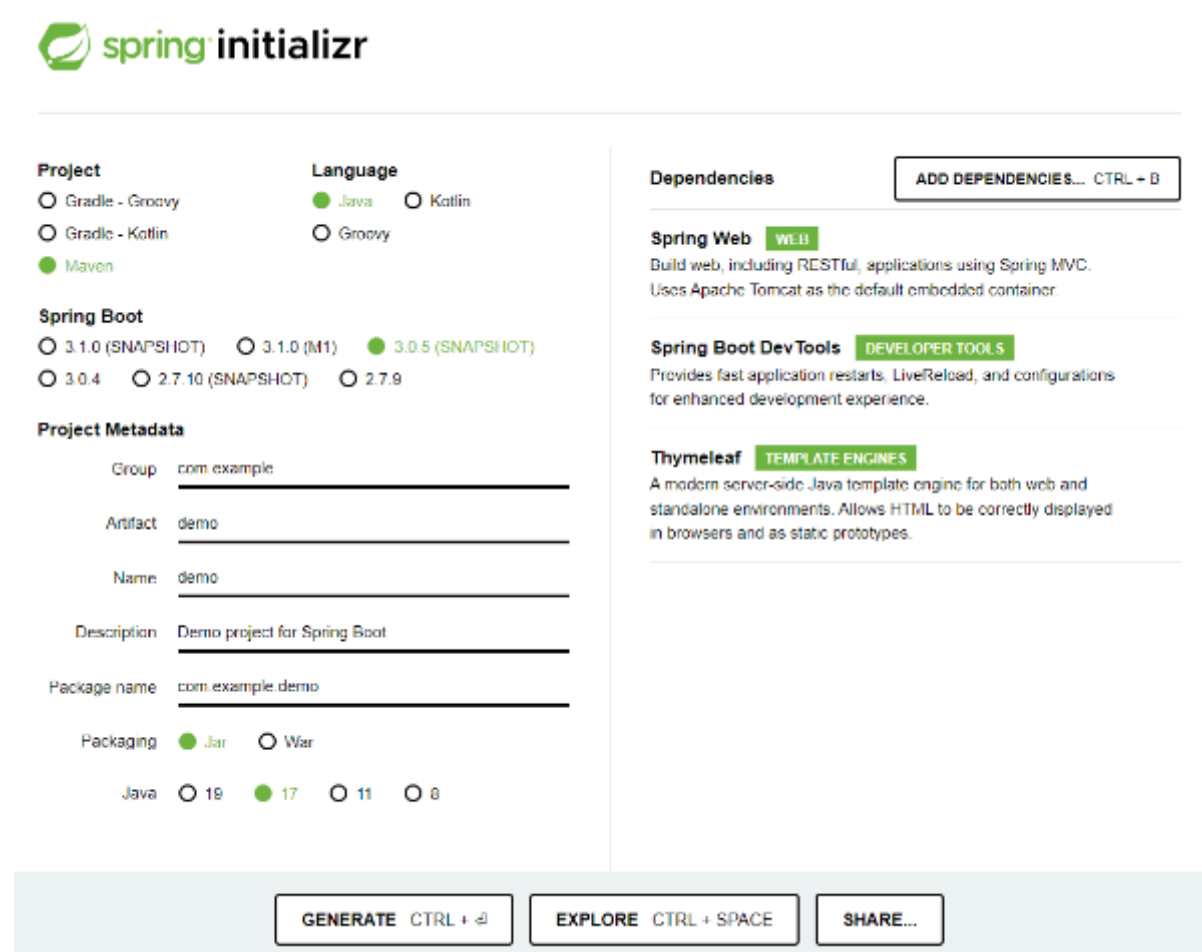


Checkpoint 01 - Spring e Maven

Spring

Spring - Framework java mais conhecido do mundo e utilizamos o padrão mvc para criação de aplicações web

<https://start.spring.io/> - Spring Initializr - Aplicação que gera configurações default



The screenshot displays the Spring Initializr web application interface. At the top, the 'spring initializr' logo is visible. The main configuration area is divided into several sections:

- Project:** Radio buttons for 'Gradle - Groovy', 'Gradle - Kotlin', and 'Maven' (selected).
- Language:** Radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'.
- Spring Boot:** Radio buttons for '3.1.0 (SNAPSHOT)', '3.1.0 (M1)', '3.0.5 (SNAPSHOT)' (selected), '3.0.4', '2.7.10 (SNAPSHOT)', and '2.7.9'.
- Project Metadata:** Text input fields for 'Group' (com.example), 'Artifact' (demo), 'Name' (demo), 'Description' (Demo project for Spring Boot), and 'Package name' (com.example.demo). Below these is a 'Packaging' section with radio buttons for 'Jar' (selected) and 'War', and a 'Java' version section with radio buttons for '19', '17' (selected), '11', and '8'.
- Dependencies:** A section with a button 'ADD DEPENDENCIES... CTRL + D'. It lists three default dependencies: 'Spring Web' (WEB), 'Spring Boot DevTools' (DEVELOPER TOOLS), and 'Thymeleaf' (TEMPLATE ENGINES), each with a brief description.

At the bottom of the form, there are three buttons: 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

Configurações utilizadas na aula

Dependências (Injeção de Dependência) - Auxilia na estruturação do códigos utilizando dependências (semelhantes a API) para reduzir o trabalho

Dependências usadas:

- Spring Web
- DevTools
- Thymeleaf
- Data JPA
- MySQL Driver

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.



Spring Boot DevTools

DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.



Thymeleaf

TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.



Spring Data JPA

SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.



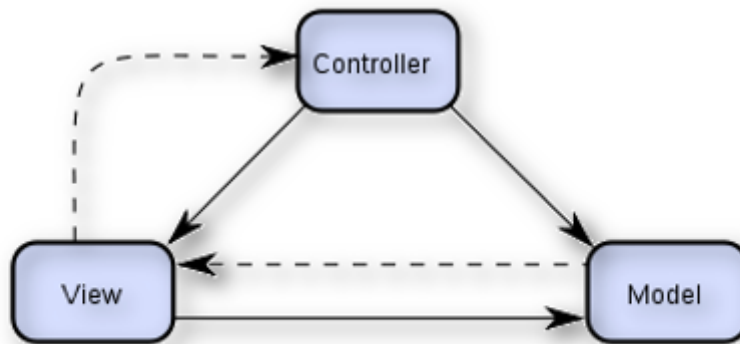
Injeção de Dependência é um é um padrão de projeto usado para evitar o alto nível de acoplamento; Vantagens: mais fácil manutenção e facilidade na implementação de testes.

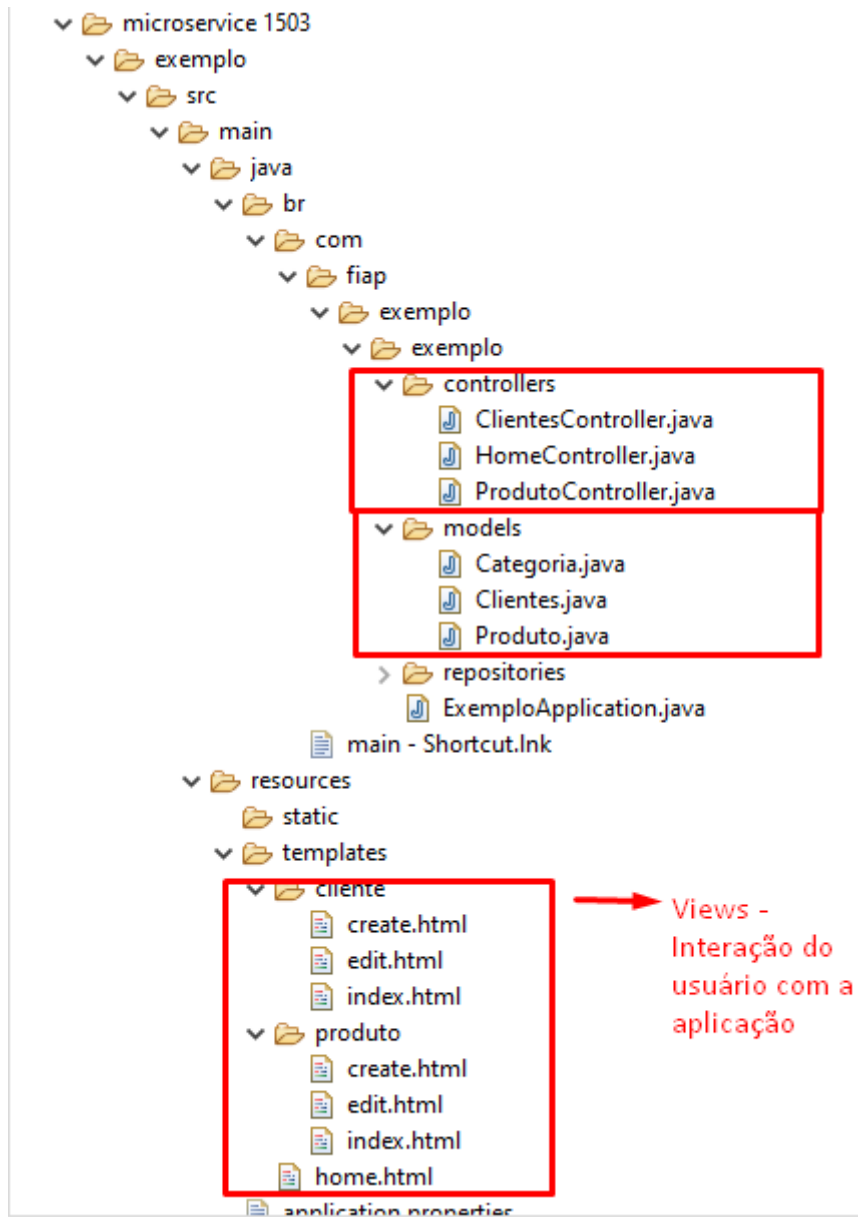
› MVC

MVC - Model, View e Controller. Utilizamos essa padrão para a codificação em java

- **Pattern Arquitetural - MVC**

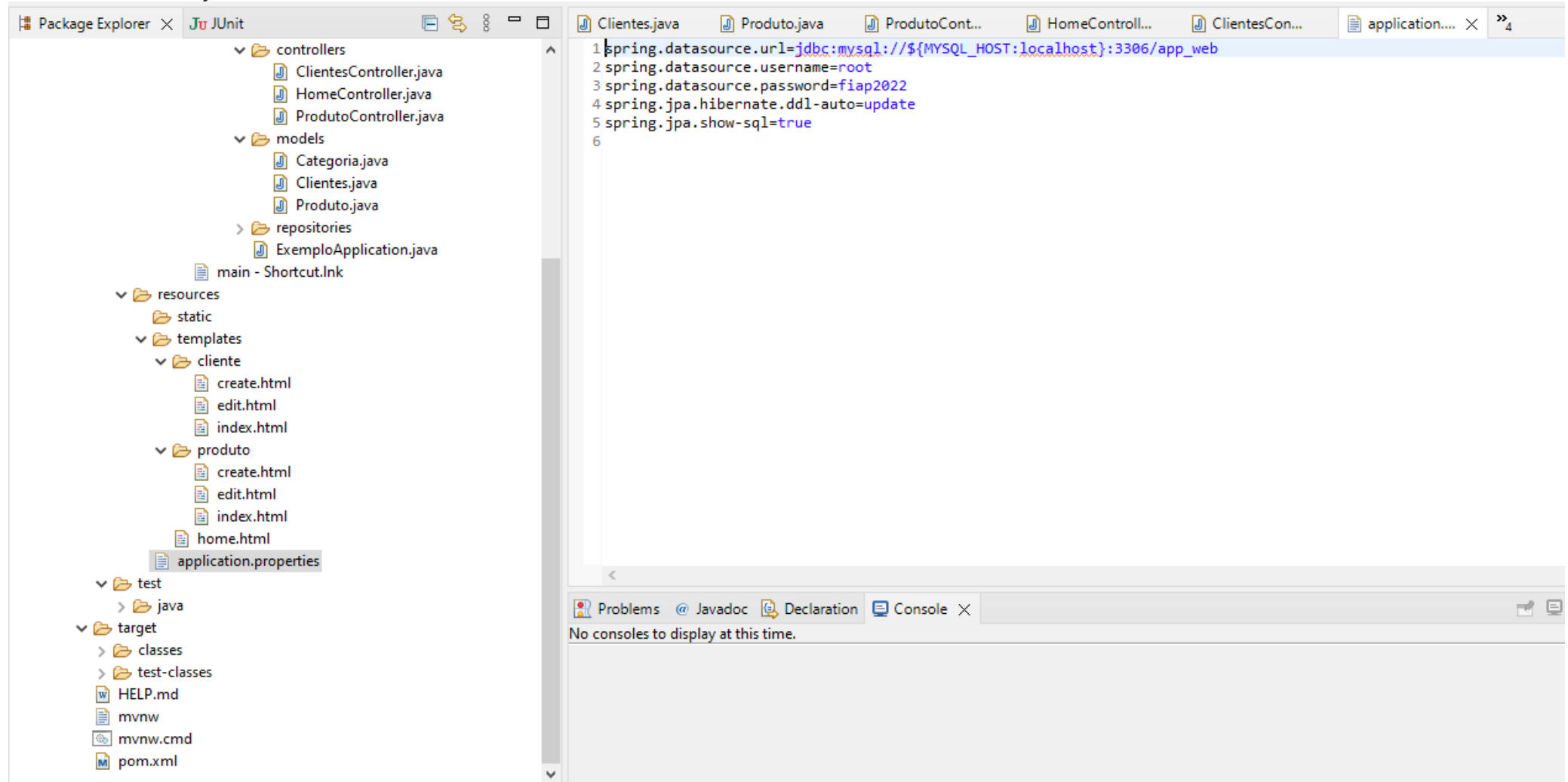
- Model - View - Controller





Views - Interação com o usuário Models - Definição das variáveis e getters and setters Controller - Direcionamento do usuário pelas páginas, listagens.

Conexão com o MySQL



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.

@GetMapping atende a uma requisição http

Exemplos:

```
@GetMapping("/hello")
public String hello(HttpServletRequest request) {
    request.setAttribute("nome", "mundo");
    return "hello";
}

@GetMapping("/edit/{idade}")
public String getById(Model model, @PathVariable("idade") Integer idIdade) {
    // model.addAttribute("")
    return "cliente/edit";
}
```

Exemplo do index arquivo prof.

Identificador para edição: .

th:text="\${idade}" - Serve para passarmos uma variável para o html da página.

Código da controller para a variável idade utilizada acima

```
@GetMapping("/edit/{idade}")
public String getById(Model model, @PathVariable("idade") Integer idIdade) {
    // model.addAttribute("")
    return "cliente/edit";
}
```

Exemplo site - <https://www.treinaweb.com.br/blog/o-que-e-o-thymeleaf>

Abaixo temos um exemplo de código escrito com o Thymeleaf:

Copiar

```
<ul>
  <li th:each="user : ${users}" >
    <a
      th:href="/user/{username} (username=${user.username})"
      th:text="${user.firstname} + ' ' + ${user.lastname}"
    ></a>
  </li>
</ul>
```

Controller

Ex Produto

```
package br.com.fiap.exemplo.exemplo.controllers;
```

```
import java.util.ArrayList; import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import
org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import
org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.servlet.ModelAndView;
```

```
import br.com.fiap.exemplo.exemplo.models.Categoria; import br.com.fiap.exemplo.exemplo.models.Produto; import
br.com.fiap.exemplo.exemplo.repositories.ProdutoRepository;
```

```
@Controller @RequestMapping("/produto") public class ProdutoController {
```

```

@Autowired
private ProdutoRepository produtoRepository;

@GetMapping("")
public ModelAndView get() {
    ModelAndView model = new ModelAndView("produto/index");

    /*
     * List<Produto> listaProduto = new ArrayList<Produto>();
     *
     * Produto produtoUm = new Produto(); produtoUm.setId(new Long(1));
     * produtoUm.setNome("Nike Lebron");
     *
     * Produto produtoDois = new Produto(); produtoDois.setId(new Long(2));
     * produtoDois.setNome("Nike do naldo");
     *
     * listaProduto.add(produtoUm); listaProduto.add(produtoDois);
     */

    List<Produto> listaProduto = produtoRepository.findAll();
    model.addObject("produtos", listaProduto);
    return model;
}

@GetMapping("/edit/{id}")
public String getById(Model model, @PathVariable("id") Integer idProduto) {
    // model.addAttribute("")
    return "produto/edit";
}

@GetMapping("/create")
public String create() {
    return "produto/create";
}

```



```

@PostMapping("/create")
public String create(@ModelAttribute("produto") Produto objProduto) {
    produtoRepository.save(objProduto);

    /*
     * // enviar para base de dados
     *
     * System.out.println(objProduto.getId());
     * System.out.println(objProduto.getNome());
     */

    return "redirect:/produto";
}

```

```

@GetMapping("/categoria")
@ResponseBody
public Categoria getCategoria() {
    Categoria categoria = new Categoria();
    categoria.setDescricao("masculino");
    categoria.setId(1);

    return categoria;
}

```

```

}

```

Model

Responsavel dentro do Controller de mandar informações para a view

```
public String home(Model model) List pedidos = Arrays.asList(pedido); model.addAttribute("pedidos", pedido);
```

Exemplo produto model

```

package br.com.fiap.exemplo.exemplo.models;

import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import
jakarta.persistence.Id;

@Entity public class Produto {

    @Id // Primary key
    @GeneratedValue(strategy = GenerationType.AUTO) // Auto incremenet

    private Long id;
    private String nome;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

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

}

```

› Repositories

Repositories do produto (banco de dados MySQL)

```
package br.com.fiap.exemplo.exemplo.repositories;

import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;

import br.com.fiap.exemplo.exemplo.models.Produto;

@Repository public interface ProdutoRepository extends JpaRepository<Produto, Long>{

}
```