Faculdade de Informática e Administração Paulista

Java Advanced

Atividade

Prof. Me. Orlando C. Patriarcha

Descrição do Problema: Lets Play a game

Existem muitos erros espalhados pela aplicação em anexo. Você deve corrigi-los todos se quiser ver o código funcionar corretamente. Mas atenção, a tarefa não é tão simples quanto parece. Cada erro é uma peça do quebra-cabeça, e sua solução é a chave para avançar. Vocês tem até o final da aula para tanto.

Entretanto, se você deseja ir além, há um desafio adicional, algo que testará sua verdadeira habilidade: adicione proteção de rotas utilizando Spring Security nas rotas que "ALTERAM O ESTADO DO SERVIDOR". Se fizer isso com sucesso, um ponto extra será seu... Mas cuidado, o tempo está passando, e as consequências da falha são irreversíveis.

Os jogadores podem tentar dividir suas equipes entre segurança e resolução de problemas ou atacar de maneira coordenada cada um dos desafios. Existem multiplas resoluções para esse problema e a heurística utilizada faz parte da solução

Aos que sobreviverem ao desafio, sua próxima tarefa é clara: entregar o projeto em um arquivo zipado, mas lembrem-se... O diretório de binários deve ser excluído. Um erro nesse detalhe, e todo o esforço será em vão. Vocês já sabem o que está em jogo. O tempo não para, e as decisões serão sempre suas.

O jogo começou. Faça suas escolhas.

Desafio

O projeto que você recebeu contém:

- Um **controlador** Java com **5 erros fatais**.
- Dois **templates Thymeleaf** — `login.html` e `dashboard.html` — cada um com **5 falhas técnicas**.

Sua missão é simples... em teoria:

1. Corrigir todos os erros e inconsistências.
2. Fazer o sistema inicializar sem exceções.
3. Garantir que o controlador e as views cooperem em perfeita harmonia.

Falhe — e sua nota escorregará como um stack trace infinito.

O Controlador Amaldiçoado

GameController.java

```
package com.example.demo.web;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
// import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.PostMapping;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@org.springframework.web.bind.annotation.RestController
public class GameController {

    private static final List<String> store = new ArrayList<>();

    @GetMappin("/")
    public String home(Model model) {
        model.addAttribute("now", LocalDateTime.now());
        model.addAttribute("itens", List.of("Relatório", "Métricas", "Exportação"));
        return "dashbord";
    }

    @GetMapping("/login")
    public String login(Model model) {
        model.addAttribute("loginMessage", "Bem-vindo!");
        return "login";
    }

    @GetMapping("/do-something")
    public String doSomething(Model model) {
        store.add("Novo registro " + LocalDateTime.now());
        model.addAttribute("message", "Feito com sucesso!");
        model.addAttribute("itens", new ArrayList<>(store));
        model.addAttribute("now", LocalDateTime.now());
        model.addAttribute("pageTitle", "Dashboard");
        return "dashboard";
    }
}
```

Dica de Jigsaw: Um método que altera o estado do sistema não deve ser invocado por um `@GetMapping`. Ou será que você quer mesmo que o avaliador descubra suas práticas REST hereges?

Template do Login

login.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
```

```
    <title th:text="${pageTitle">Login</title>
  </head>
  <body>

    <h1 th:text="${pageTitle}">Tela de Login</h1>

    <form method="post"
          th:action="@{/login">
      <label for="username">Usuário</label>
      <input id="username" th:field="*{username}" type="text" placeholder="Seu usuário">

      <label for="password">Senha</label>
      <input id="password" th:field="*{password}" type="password" placeholder="Sua senha">

      <button type="submti">Entrar</button>
    </form>

    <p><a th:href="@{/}">Voltar ao painel</a></p>

  </body>
</html>
```

Jigsaw sussurra: "Cada tag quebrada é um ponto a menos... cada sintaxe errada é um pedaço do seu sono perdido."

Template do Dashboard

dashboard.html

```
<!DOCTYPE html>
<html lang="pt-BR" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8">
    <title th:text="${pageTitle">Dashboard</title>
  </head>
  <body>

    <h1 th:texts="${pageTitle}">Dashboard</h1>

    <p>
      Agora é:
      <strong th:text="${now.toString()} )">2025-10-08T12:34</strong>
    </p>

    <ul>
      <li th:each="item ${itens}" th:text="${item}">Item</li>
    </ul>

    <div th:if="${message}" th:text="${message}">

    <p><a th:href="@{/}">Voltar</a></p>
  </body>
</html>
```

Jigsaw murmura: "Feche suas tags. Feche suas lacunas. Ou eu fecho seu semestre."

Condições de Vitória

- O projeto compila e executa sem erros.
- Todas as views renderizam corretamente.
- O método que altera o estado usa `@PostMapping`.
- Nenhum aluno foi reprovado no processo (ainda).

"Corrija o código, passe de ano, sobreviva ao semestre."
— **Jigsaw, Engenheiro do Spring MVC**

CONTINUA NA PROXIMA PAGINA

Figure 1: Lets play a SPRING game