**Documentação - Testes – Sistema ERP web M&V Estética Automotiva**

## **1. Erro: "No database selected"**

### **Contexto do Teste:**

**Cenário:**  
 Ao acessar o módulo de **Clientes** ou outros módulos que executam consultas no banco de dados, a aplicação retorna um erro de conexão.

**Mensagem de Erro:**

JDBC exception executing SQL [select c1\_0.id,c1\_0.nome,c1\_0.telefone from clientes c1\_0] [No database selected] [n/a]

### **Análise:**

O erro indica que o MySQL não sabe qual banco de dados utilizar, pois o nome do banco não foi especificado na URL de conexão.

### **Solução Aplicada:**

**Atualização do arquivo application.properties:**

Incluímos o nome do banco de dados na URL de conexão.

Exemplo:

spring.datasource.url=jdbc:mysql://localhost:3306/estetica\_auto?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC  
 spring.datasource.username=root  
 spring.datasource.password=

**Resultado:**  
 Após essa alteração, a conexão com o MySQL foi estabelecida corretamente e as consultas passaram a funcionar.

## **2. Erro: Dependência do MySQL ausente no pom.xml**

### **Contexto do Teste:**

**Cenário:**  
 Ao iniciar a aplicação, o Spring Boot não conseguia conectar-se ao banco de dados MySQL, pois a dependência do driver JDBC não estava adicionada ao pom.xml.

**Mensagem de Erro:**

Cannot load JDBC driver class 'com.mysql.cj.jdbc.Driver'

### **Análise:**

O erro ocorre porque o projeto não encontrava o driver necessário para se conectar ao MySQL.

A ausência da dependência mysql-connector-java no pom.xml impedia a comunicação entre a aplicação e o banco de dados.

### **Solução Aplicada:**

**Adição da dependência do**  **MySQL no pom.xml**

<dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>

**Resultado:**  
 Após a inclusão dessa dependência e a atualização do Maven, a aplicação conseguiu reconhecer o driver e conectar-se corretamente ao banco de dados.

## **3. Erro: Problema na Formatação de Data em Template Thymeleaf**

### **Contexto do Teste:**

### **Cenário:** Na tela de **Agendamentos**, ao tentar exibir a data e hora formatadas utilizando o atributo:

<td th:text="${#dates.format(agendamento.dataHora, 'dd/MM/yyyy HH:mm')}"></td>  
   
a aplicação retornava uma exceção durante o parsing do template.

**Mensagem de Erro:**

Exception evaluating SpringEL expression: "#dates.format(agendamento.dataHora, 'dd/MM/yyyy HH:mm')"   
...  
 Caused by: org.springframework.expression.spel.SpelEvaluationException: EL1004E: Method call: Method format(java.time.LocalDateTime,java.lang.String) cannot be found on type org.thymeleaf.expression.Dates  
   
**Análise:**

A expressão #dates não possui o método format para objetos do tipo java.time.LocalDateTime.

O Thymeleaf (versões recentes) disponibiliza o utilitário #temporals para manipulação e formatação de objetos da API java.time.

### **Solução Aplicada:**

**Alteração do template:**

Substituímos a expressão #dates.format(...) por #temporals.format(...).

Exemplo corrigido:

<td th:text="${#temporals.format(agendamento.dataHora, 'dd/MM/yyyy HH:mm')}"></td>  
   
**Resultado:**  
 Com a alteração, o template foi processado corretamente e as datas passaram a ser exibidas no formato desejado.

**Error 404**

Este aplicativo não tem mapeamento explícito para /error, então você está vendo isso como um fallback. Seg Mar 17 10:00:40 BRT 2025 Ocorreu um erro inesperado (tipo=Não Encontrado, status=404).

**Dito pelo ChatGPT:**

O erro que você está recebendo é uma página padrão do erro 404 (não encontrado), indicando que a aplicação não conseguiu encontrar o recurso solicitado. Isso pode ser causado por algumas razões. Vou te ajudar a identificar e corrigir o problema.

### **Possíveis Causas e Soluções:**

***1.*** *Rota de URL incorreta ou não mapeada no driver*

Esse erro pode ocorrer se uma URL que você está tentando acessar não tiver um driver que mapeie corretamente. Verifique se as rotas em seu ClienteController(ou outro driver relevante) estão corretas.

**Solução:**

**Verifique as rotas no controlador** : Se você tem um controlador que lida com /clientes, e está tentando acessar /clientes/novo, /clientes/editar/{id}, ou outras rotas relacionadas a clientes, as rotas precisam estar corretamente mapeadas.

No exemplo que você me enviou anteriormente, o driver ClienteControllerdeve mapear as URLs corretamente:

@Controller

@RequestMapping("/clientes")

public class ClienteController {

@Autowired

private ClienteRepository clienteRepository;

@GetMapping

public String listar(Model model) {

model.addAttribute("clientes", clienteRepository.findAll());

return "clientes/lista"; // A página de listagem de clientes

}

@GetMapping("/novo")

public String novoCliente(Model model) {

model.addAttribute("cliente", new Cliente());

return "clientes/form"; // A página para criar um novo cliente

}

@GetMapping("/editar/{id}")

public String editarCliente(@PathVariable("id") Integer id, Model model) {

Cliente cliente = clienteRepository.findById(id).orElse(null);

if (cliente == null) {

return "redirect:/clientes"; // Redireciona para a lista de clientes se não encontrar

}

model.addAttribute("cliente", cliente);

return "clientes/form"; // A página de edição do cliente

}

@PostMapping("/salvar")

public String salvarCliente(@Valid @ModelAttribute("cliente") Cliente cliente,

BindingResult result, Model model) {

if (result.hasErrors()) {

return "clientes/form"; // Retorna ao formulário em caso de erro

}

clienteRepository.save(cliente);

return "redirect:/clientes"; // Redireciona após salvar

}

@GetMapping("/deletar/{id}")

public String deletarCliente(@PathVariable("id") Integer id) {

clienteRepository.deleteById(id);

return "redirect:/clientes"; // Redireciona para a lista de clientes após excluir

}

}

***2.*** *Página de erro não definida (parágrafo* ***/error****)*

Se você receber um erro 404, isso pode ser causado pela falta de uma página de erro personalizada ou falta de configuração para lidar com erros. O Spring Boot, por padrão, tenta redirecionar para /errorquando um erro ocorre, mas se não houver uma página de configuração de erro, ele exibirá essa página genérica.

Você pode criar uma página personalizada de erro ou configurar o Spring Boot para exibir mensagens de erro mais desenvolvidas.

**Como criar uma página de erro personalizada:**

Você pode criar uma página de erro personalizada dentro da pasta src/main/resources/templatese chamar o arquivo de error.html. Exemplo de um arquivo simples error.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Erro</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body>

<div class="container mt-4">

<h2 class="text-danger">Ocorreu um erro inesperado!</h2>

<p>Erro: <span th:text="${status}">404</span></p>

<p th:text="${message}">Página não encontrada</p>

<a href="/" class="btn btn-primary">Voltar para a Página Inicial</a>

</div>

</body>

</html>

Ao criar esse arquivo, qualquer erro inesperado (como 404) será redirecionado para essa página e você receberá uma mensagem mais amigável ao usuário.

**Configurando uma página de erro personalizada no Spring Boot:**

Se você deseja personalizar a configuração de erro, pode adicionar uma classe que captura e processa erros:

@ControllerAdvice

public class ErrorController implements org.springframework.boot.web.servlet.error.ErrorController {

@Override

public String getErrorPath() {

return "/error"; // A página customizada de erro será retornada ao acessar /error

}

@RequestMapping("/error")

public String handleError() {

// Exibe a página de erro personalizada

return "error";

}

}

***3.*** *Verifique as URLs no Navegador*

Se você está tentando acessar uma URL diretamente no navegador, verifique se está digitando corretamente uma URL que corresponde ao mapeamento no driver.

Por exemplo:

/clientes/novo: Para criar um novo cliente.

/clientes/editar/{id}: Para editar um cliente específico.

/clientes/deletar/{id}: Para excluir um cliente.

***4.*** *Verifique se o método* ***salvar*** *está correto*

O erro também pode ser causado pela falta de um mapeamento adequado para o método salvar. No código de exemplo que passei, usamos o @PostMapping("/salvar"). Isso garante que o formulário seja processado corretamente.

Verifique se o formulário no Thymeleafestá apontando para o método correto e a URL que está sendo usada corresponde ao método do driver:

<form th:action="@{/clientes/salvar}" th:object="${cliente}" method="post">

Esse formulário chama uma URL /clientes/salvarpara o método salvarClienteno driver.

## **Conclusão**

Esses testes e as respectivas correções garantiram que:

A conexão com o banco de dados seja estabelecida corretamente, evitando erros de "No database selected".

O driver MySQL seja reconhecido pelo Spring Boot, permitindo a conexão com o banco.

Os templates Thymeleaf sejam renderizados corretamente, com a formatação de datas ajustada para trabalhar com objetos LocalDateTime.

Verifique se as rotas estão corretamente configuradas e se existe uma URL que você está tentando acessar. Adicionar um driver de erro pode ajudar a fornecer uma experiência de erro mais amigável para o usuário. Se o erro 404 continuar, tente depurar a aplicação para garantir que as rotas estejam mapeadas corretamente e que o Spring Boot esteja processando as requisições corretamente.