

## Exercício 6 - MVC

Este exercício deve ser feito como **alteração do Exercício** feito anteriormente. As alterações são:

- Todas as telas devem ter BOOTSTRAP
- Todas as páginas JSP devem indicar que a página de erro é **erro.jsp**
- A página **erro.jsp** agora deve ser uma página de erro e tratar a exceção
- Todos os redirecionamentos em JSPs devem usar a tag **<jsp:forward />**. Cuidado nas JSPs que recebem dados de erros (**index.jsp**), eles podem vir via **request.getAttribute()** ou **request.getParameter()**. Tratar os dois casos.
- Para processar os dados de um cliente, somente a Servlet **ClientesServlet** deve ser invocada
- Usar EL para mostrar informações da sessão, requisição e onde for possível com o aprendido (**\${mensagem}**, **<c:forEach>**)
- Trocar os rodapés das páginas para usar EL para mostrar o e-mail do administrador, não precisando mais do **<jsp:useBean />** (ex.: **\${configuracao.email}**)
- Criar um Façade (**com.ufpr.tads.web2.facade.ClientesFacade**) contendo métodos estáticos que devem ser invocados pela servlet para efetuar as ações de uma Servlet. As Servlets só invocam a Façade **ClientesFacade**, não mais invocam as DAOs diretamente, pois isso é feito pelos seus métodos. Métodos:
  - **public static void inserir(Cliente c);**
  - **public static void alterar(Cliente c);**
  - **public static Cliente buscar(int id);**
  - **public static List<Cliente> buscarTodos();**

Agora todas as classes Java deverão estar dentro dos pacotes conforme indicado nas descrições abaixo.

Deve-se usar o banco de dados já criado contendo uma tabela chamada **tb\_usuario** contendo os campos **id\_usuario** (PK, inteiro), **login\_usuario** (VARCHAR(50)), **senha\_usuario** (VARCHAR(50)) e **nome\_usuario** (VARCHAR(100)).

Deve-se ter pelo menos 3 usuários nessa tabela. Esta tabela será usada para efetuar login no sistema.

Deve-se adicionar uma tabela chamada **tb\_cliente** contendo os campos **id\_cliente** (PK, inteiro), **cpf\_cliente** (CHAR(11)), **nome\_cliente** (VARCHAR(100)), **email\_cliente** (VARCHAR(100)), **data\_cliente** (DATE), **rua\_cliente** (VARCHAR(100)), **nr\_cliente**

(INTEGER), **cep\_cliente** (CHAR(8)), **cidade\_cliente** (VARCHAR(100)), **uf\_cliente** (CHAR(2)).

Adicionar pelo menos 10 clientes nessa tabela.

Todas as telas JSPs e HTMLs devem usar Bootstrap.

Link para básico do BOOTSTRAP:

[https://www.w3schools.com/bootstrap4/bootstrap\\_get\\_started.asp](https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp)

## DICA PÁGINA DE ERRO:

Em Servlets pode-se redirecionar para página de erro (marcada com **isErrorPage**) da seguinte forma:

```
request.setAttribute("javax.servlet.jsp.jspException", <objeto Exceção a ser mostrada> );
request.setAttribute("javax.servlet.error.status_code", 500);
RequestDispatcher rd = getServletContext().getRequestDispatcher("/erro.jsp");
rd.forward(request, response);
```

### 1) index.jsp

- Deve-se incluir BOOTSTRAP para formatação desta página
- Deve receber um parâmetro no escopo da requisição (chamado "msg"). Se este parâmetro for passado, deve ser apresentado em vermelho, como uma mensagem de aviso, junto com a página de login. Se não for passado, somente a página de login é apresentada.

No rodapé da página deve ser colocada a seguinte mensagem: **"Em caso de problemas contactar o administrador: "** juntamente com o e-mail armazenado no escopo da aplicação.

### 2) com.ufpr.tads.web2.servlets.StartupServlet

É uma servlet que deve ser inicializada no Startup da aplicação. Dentro do seu método **init()**, deve-se armazenar no escopo da aplicação (ServletContext) uma instância preenchida de um ConfigBean, com o nome "configuracao".

### 3) com.ufpr.tads.web2.servlets.LoginServlet

É uma servlet que verifica o login do usuário.

Se o par login/senha estiver no banco de dados:

- Preenche uma instância de LoginBean com os dados do usuário

- Armazena o LoginBean na sessão (indicando que o usuário está logado);
- Redireciona para a servlet **portal.jsp**.

Se o par login/senha não estiver no banco de dados:

- Redireciona para **index.jsp** (via *forward*), passando como parâmetro a mensagem de erro a ser apresentada (parâmetro "msg") "**Usuário/Senha inválidos.**"

#### 4) portal.jsp

A primeira coisa que **portal.jsp** faz é verificar se o usuário está logado.

- Abra um código scriptlet (<% ... %>)
- O JSP já possui um objeto **session** instanciado, basta acessá-lo
- Se não estiver logado:
  - Redireciona para **index.jsp** (via *forward*), passando como parâmetro a mensagem de erro a ser apresentada (parâmetro "msg") "**Usuário deve se autenticar para acessar o sistema.**"
- Se o usuário estiver logado:
  - Obtenha o objeto **LoginBean** da sessão usando `<jsp:useBean />`
  - Apresenta uma tela com o nome do usuário logado (exemplo: `${login.nome}`)
  - Deve-se incluir um MENU contendo as opções:
    - **Cadastro de Clientes:** que direciona para a servlet **ClientesServlet**
    - **Sair:** que direciona para a servlet **LogoutServlet**

No rodapé da página deve ser colocada a seguinte mensagem: "**Em caso de problemas contactar o administrador:** " juntamente com o e-mail armazenado no escopo da aplicação.

#### 5) com.ufpr.tads.web2.servlets.LogoutServlet

Efetua o logout do usuário e direciona, via **forward** para **index.jsp**, passando como parâmetro a mensagem "**Usuário desconectado com sucesso**"

#### 6) erro.jsp

Esta agora será uma página de erro, portanto adicione a diretiva que faz esta indicação.

Apresenta, em vermelho, a mensagem de erro de dentro da exceção que gerou a sua chamada (objeto implícito **exception**). Devem ser mostrados a mensagem e logo após o StackTrace:

Mostrar Mensagem:

```
${pageContext.exception.message}
```

Mostrar StackTrace:

```
${pageContext.out.flush()}
```

```
${pageContext.exception.printStackTrace(pageContext.response.writer)}
```

No rodapé da página deve ser colocada a seguinte mensagem: "Em caso de problemas contactar o administrador: " juntamente com o e-mail armazenado no escopo da aplicação.

## 7) com.ufpr.tads.web2.servlets.CientesServlet

Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Esta servlet processa TODAS as requisições relativas a clientes. Ela deve SEMPRE receber um parâmetro chamado **action**. A seguir os valores possíveis e ações a serem tomadas

1. Valor: "**list**" ou nulo (não passado)

Invoca a Façade para buscar todos os clientes da base de dados, como um **List<Cliente>**, e os coloca no escopo da requisição. Em seguida, efetua **forward** para **clientesListar.jsp**.

2. Valor: "**show**"

- Recebe um parâmetro (via parameter, GET) com o Id do cliente a ser mostrado
- Invoca a Façade para buscar no banco de dados o objeto do cliente
- Adiciona o objeto no escopo da requisição
- Efetua forward para **clientesVisualizar.jsp**

3. Valor: "**formUpdate**"

- Recebe um parâmetro (via parameter, GET) com o Id do cliente a ser alterado
- Invoca a Façade para busca no banco de dados o objeto do cliente
- Adiciona o objeto no escopo da requisição
- Efetua forward para **clientesAlterar.jsp**

#### 4. Valor: "remove"

- Recebe um parâmetro (via parameter, GET) com o Id do cliente a ser removido
- Invoca a Façade para remover o cliente do banco de dados
- Efetua forward para **ClientesServlet**

#### 5. Valor: "update"

- Recebe todos os dados do cliente passados via POST pelo formulário
- Invoca a Façade para realizar uma atualização na base de dados
- Efetua um redirecionamento para **ClientesServlet**.

#### 6. Valor: "formNew"

- Efetua um redirecionamento para **clientesNovo.jsp**.

#### 7. Valor: "new"

- Recebe todos os dados do cliente passados via POST pelo formulário
- Invoca a Façade para realizar uma inserção na base de dados
- Efetua um redirecionamento para **ClientesServlet**.

### 8) com.ufpr.tads.web2.beans.Cliente

Um Java Bean que contém os dados da tabela **tb\_cliente**. Este bean deve ser usado juntamente com as classes DAO para acessar o banco de dados.

### 9) clientesListar.jsp

Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Esta tela apresenta o mesmo formato que **portal.jsp**, alterando somente seu conteúdo. Deve apresentar, em forma de tabela, os dados de Cliente: CPF, Nome e E-mail. Adicionalmente, em cada linha, deve-se apresentar 3 pequenas imagens representando as ações: Visualizar, Alterar e Remover; com os seguintes links:

- Visualizar: para a Servlet **ClientesServlet?action=show&id=10** (onde id=10 é o id do cliente naquela linha)
- Alterar: para a Servlet **ClientesServlet?action=formUpdate&id=10** (idem)
- Remover: para a Servlet **ClientesServlet?action=remove&id=10** (idem)

Use **<c:forEach items="\${lista}" var="cliente" />** para apresentar esta lista. Para adicionar o id do cliente nos links, use por exemplo:  
**ClientesServlet?action=remove&id=\${cliente.id}**

Esta tela também deve apresentar um botão de Novo, com um link para a servlet **ClientesServlet?action=formNew**.

#### **40) com.ufpr.tads.web2.servlets.VisualizarClienteServlet**

Inicialmente verifica-se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Executa os seguintes passos:

- Recebe um parâmetro (via parameter, GET) com o Id do cliente a ser mostrado
- Busca no banco de dados o objeto do cliente
- Adiciona o objeto no escopo da requisição
- Efetua forward para **clientesVisualizar.jsp**

#### **41) com.ufpr.tads.web2.servlets.RemoverClienteServlet**

Inicialmente verifica-se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Executa os seguintes passos:

- Recebe um parâmetro (via parameter, GET) com o Id do cliente a ser removido
- Remove o cliente do banco de dados
- Efetua forward para **ClientesServlet**

#### **42) com.ufpr.tads.web2.servlets.FormAlterarClienteServlet**

Inicialmente verifica-se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Executa os seguintes passos:

- ~~Recebe um parâmetro (via parameter, GET) com o Id do cliente a ser alterado~~
- ~~Busca no banco de dados o objeto do cliente~~
- ~~Adiciona o objeto no escopo da requisição~~
- ~~Efetua forward para **clientesAlterar.jsp**~~

### 13) clientesAlterar.jsp

Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Deve apresentar um formulário com todos os dados do cliente, já preenchidos com os dados recebidos via request, um botão **Alterar** e um botão **Cancelar**.

Use EL (ex.: **`${cliente.nome}`**) para apresentar estes dados. Não é mais necessário usar a tag **`<jsp:useBean />`**

Ao pressionar **Alterar**, deve-se submeter para **ClientesServlet?action=update**. Ao pressionar **Cancelar**, deve-se direcionar para **ClientesServlet**.

### 14) clientesListar.jsp

Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Deve apresentar todos os dados do cliente passado no escopo da requisição, e um botão **Voltar**.

Use EL (ex.: **`${cliente.nome}`**) para apresentar estes dados. Não é mais necessário usar a tag **`<jsp:useBean />`**

Ao pressionar **Voltar**, deve-se submeter para **ClientesServlet?action=list**.

### ~~14) com.ufpr.tads.web2.servlets.AlterarClienteServlet~~

~~Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").~~

~~Recebe todos os dados do cliente passados via POST pelo formulário e realiza uma atualização na base de dados. Então, efetua um redirecionamento para **ClientesServlet**.~~

#### ~~15) com.ufpr.tads.web2.servlets.FormNovoClienteServlet~~

~~Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").~~

~~Efetua um redirecionamento para **clientesNovo.jsp**.~~

#### 15) clientesNovo.jsp

Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").

Deve apresentar um formulário com todos os dados do cliente (limpos para uma adição), um botão **Salvar** e um botão **Cancelar**. Ao pressionar Salvar, deve-se submeter para **ClientesServlet?action=new**. Ao pressionar Cancelar, deve-se direcionar para **ClientesServlet**.

#### ~~17) com.ufpr.tads.web2.servlets.NovoClienteServlet~~

~~Inicialmente verifica se o usuário está logado. Se não estiver, redireciona via **forward** para **index.jsp** passando como parâmetro a mensagem "**Usuário deve se autenticar para acessar o sistema**" (parâmetro "msg").~~

~~Recebe todos os dados do cliente passados via POST pelo formulário e realiza uma inserção na base de dados. Então, efetua um redirecionamento para **ClientesServlet**.~~

#### 16) com.ufpr.tads.web2.beans.LoginBean

Um Java Bean que contém os dados de login a serem armazenados na sessão. Contém o id do usuário, e o nome do usuário.

#### 17) com.ufpr.tads.web2.beans.ConfigBean

Um Java Bean que contém dados de configuração da aplicação, no caso o e-mail do administrador.



#### **18) com.ufpr.tads.web2.beans.Usuario**

Um Java Bean que contém os dados da tabela **tb\_usuario**. Este bean deve ser usado juntamente com as classes DAO para acessar o banco de dados.

#### **19) com.ufpr.tads.web2.dao.UsuarioDAO**

É a classe DAO de acesso do usuário ao banco de dados.

#### **20) com.ufpr.tads.web2.dao.ConnectionFactory**

É a classe que controla a conexão com o banco de dados.

#### **21) com.ufpr.tads.web2.dao.ClienteDAO**

É a classe DAO de acesso do usuário ao banco de dados para manipular os dados de clientes.