



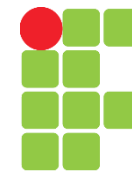
Desenvolvimento para WEB II

Prof. Roberto Alencar

O que veremos nesta aula ?

- Implementando a Pesquisa de Produtos
- Implementando a Pesquisa de Produtos **com AJAX**
- Melhorando a aparência da aplicação





Implementando a Pesquisa de Produtos





Implementando Pesquisa

Implementando um filtro para a pesquisa

→ Implemente um filtro para a tela de pesquisa de Produtos:

[Home](#) | [Cadastrar Usuário](#) | [Listar Usuario](#) | [Cadastrar Produto](#) | [Listar Produto](#) | [Cadastrar Categ. de Produto](#) | [Listar Categ. de Produto](#) | [Logout](#)

Pesquisar Produto

ID	CATEGORIA	CÓDIGO	DESCRIÇÃO	PREÇO DE CUSTO	PREÇO DE VENDA	GARANTIA	QUANTIDADE	IMAGEM	AÇÕES
55	Monitor	RTFA	Monitor LCD	250.0	340.0	02/01/2019	31		Editar Remover
58	Notebook	D14T	Notebook ACER	1420.0	1650.0	02/01/2019	15		Editar Remover
56	Notebook	ROC8	Notebook DELL i7	2300.0	2600.0	01/02/2019	9		Editar Remover
59	Teclado	C3PO	Teclado ABNT	12.5	25.0	02/01/2019	32		Editar Remover

Implementando um filtro para a pesquisa

➔ Implemente um filtro para a tela de pesquisa de Produtos:

[Home](#) | [Cadastrar Usuário](#) | [Listar Usuario](#) | [Cadastrar Produto](#) | [Listar Produto](#) | [Cadastrar Categ. de Produto](#) | [Listar Categ. de Produto](#) | [Logout](#)

Pesquisar Produto

Descrição:





Categoria:

Selecione

Limpar

Pesquisar

Lista de Produtos

ID	CATEGORIA	CÓDIGO	DESCRIÇÃO	PREÇO DE CUSTO	PREÇO DE VENDA	GARANTIA	QUANTIDADE	IMAGEM	AÇÕES
55	Monitor	RTFA	Monitor LCD	250.0	340.0	02/01/2019	31		Editar Remover
58	Notebook	D14T	Notebook ACER	1420.0	1650.0	02/01/2019	15		Editar Remover
56	Notebook	ROC8	Notebook DELL i7	2300.0	2600.0	01/02/2019	9		Editar Remover
59	Teclado	C3PO	Teclado ABNT	12.5	25.0	02/01/2019	32		Editar Remover

Implementando um filtro para a pesquisa

→ Desta forma, o código do JSP da pesquisa ficaria:

```
...  
<hr> <h3>Pesquisar Produto</h3>  
<div>  
  <form action="pesquisarProduto">  
    <p>  
      Descrição: <br />  
      <input type="text" id="descricao" name="descricao" value="${produto.descricao}">  
    </p>  
    <p>  
      Categoria: <br />  
      <select id="categoriaProduto" name="categoriaProduto" >  
        <option value=""> Selecione </option>  
        <c:forEach items="${listaCategoriaProduto}" var="obj">  
          <option value="${obj.id}"> ${obj.descricao} </option>  
        </c:forEach>  
      </select>  
    </p>  
    <p>  
      <input type="reset" value="Limpar"> &nbsp; &nbsp; &nbsp;  
      <input type="submit" value="Pesquisar">  
    </p>  
  </form>  
</div>  
<hr> <h3>Lista de Produtos </h3> <br />  
...
```

Implementando um filtro para a pesquisa

➔ Código do método `pesquisarProduto` no `ProdutoController`:

```
@RequestMapping("/pesquisarProduto")
public String pesquisarProduto(Produto produto, Model model) {

    // Código para popular o combo de categoria de produto
    CategoriaProdutoDao dao = new CategoriaProdutoDao();
    List<CategoriaProduto> listaCategoriaProduto = dao.listar();
    model.addAttribute("listaCategoriaProduto", listaCategoriaProduto);

    ProdutoDao dao2 = new ProdutoDao();
    List<Produto> listaProduto = dao2.pesquisar(produto);
    model.addAttribute("listaProduto", listaProduto);

    return "produto/pesquisarProduto";
}
```

Implementando um filtro para a pesquisa

→ Código do método `pesquisar` no `ProdutoDao`:

```
public List<Produto> pesquisar(Produto produto) {
    try {
        List<Produto> listaProduto = new ArrayList<Produto>();
        PreparedStatement stmt = null;
        if (!produto.getDescricao().equals("") && produto.getCategoriaProduto() == null) {
            stmt = this.connection.prepareStatement("SELECT * FROM produto WHERE descricao LIKE ? ORDER BY
descricao");
            stmt.setString(1, "%" + produto.getDescricao() + "%");
        } else if (produto.getDescricao().equals("") && produto.getCategoriaProduto() != null) {
            stmt = this.connection.prepareStatement("SELECT * FROM produto WHERE categoria_id = ? ORDER BY
descricao");
            stmt.setInt(1, produto.getCategoriaProduto().getId());
        } else if (!produto.getDescricao().equals("") && produto.getCategoriaProduto() != null) {
            stmt = this.connection.prepareStatement("SELECT * FROM produto WHERE descricao LIKE ? AND categoria_id
= ? ORDER BY descricao");
            stmt.setString(1, "%" + produto.getDescricao() + "%");
            stmt.setInt(2, produto.getCategoriaProduto().getId());
        } else {
            stmt = this.connection.prepareStatement("SELECT * FROM produto ORDER BY descricao");
        }
        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            listaProduto.add(montarObjeto(rs));
        }
        rs.close();
        stmt.close();
        connection.close();
        return listaProduto;
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```


Implementando um filtro para a pesquisa

→ Resultado do filtro da pesquisa:


[Home](#) | [Cadastrar Usuário](#) [Listar Usuario](#) | [Cadastrar Produto](#) [Listar Produto](#) | [Cadastrar Categ. de Produto](#) [Listar Categ. de Produto](#) | [Logout](#)

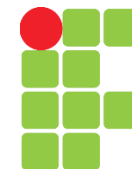
Pesquisar Produto

Descrição:

Categoria:

Lista de Produtos

ID	CATEGORIA	CÓDIGO	DESCRIÇÃO	PREÇO DE CUSTO	PREÇO DE VENDA	GARANTIA	QUANTIDADE	IMAGEM	AÇÕES
56	Notebook	ROC8	Notebook DELL i7	2300.0	2600.0	01/02/2019	9		Editar Remover



Implementando a Pesquisa de Produtos com AJAX

Implementando a pesquisa com AJAX





- Como pesquisa por produtos **sem precisar clicar no botão Pesquisar?** Queremos fazer uma pesquisa sem que a tela precise ser submetida e recarregada.

Pesquisar Produto

Descrição:

Categoria:

Lista de Produtos

ID	CATEGORIA	CÓDIGO	DESCRIÇÃO	PREÇO DE CUSTO	PREÇO DE VENDA	GARANTIA	QUANTIDADE	IMAGEM	AÇÕES
55	Monitor	RTFA	Monitor LCD	250.0	340.0	02/01/2019	31		Editar Remover
58	Notebook	D14T	Notebook ACER	1420.0	1650.0	02/01/2019	15		Editar Remover
56	Notebook	ROC8	Notebook DELL i7	2300.0	2600.0	01/02/2019	9		Editar Remover
59	Teclado	C3PO	Teclado ABNT	12.5	25.0	02/01/2019	32		Editar Remover

Implementando a pesquisa com AJAX

- Ou seja, de alguma forma precisamos mandar a requisição para o método `pesquisarProduto` no `ProdutoController`, mas ainda assim queremos manter a página do jeito que ela estava sem precisar recarregar nada.
- Podemos fazer isso através de uma técnica chamada **AJAX**, que significa *Asynchronous Javascript and XML*.

Implementando a pesquisa com AJAX

- AJAX nada mais é do que uma técnica que nos permite enviar requisições assíncronas, ou seja, manter a página que estava aberta intacta, e recuperar a resposta dessa requisição para fazermos qualquer processamento na tela.
- Essas respostas costumam ser XML, HTML ou um formato de transferência de dados chamado JSON (*Javascript Object Notation*).
- Para realizarmos uma requisição AJAX, precisamos utilizar Javascript. E neste curso, vamos utilizar o suporte que o jQuery nos fornece para trabalhar com AJAX.

Implementando a pesquisa com AJAX

- Implemente as funções em jQuery para capturar as alterações nos campos de descrição e categoria de produto:

```
<script type="text/javascript">
```

```
$(document).ready(function() {
```

```
    $("#descricao").keyup(function() {
```

```
        var texto = $('#descricao').val();
```

```
        var idCategoria = $('#categoriaProduto').val();
```

```
        ...
```

```
    });
```

```
    $("#categoriaProduto").change(function() {
```

```
        var texto = $('#descricao').val();
```

```
        var idCategoria = $('#categoriaProduto').val();
```

```
        ...
```

```
    });
```

```
});
```

```
</script>
```

Evento jQuery para quando uma tecla for digitada no seletor uma ação seja executada.

Evento jQuery para quando o seletor for alterado uma ação seja executada.

Implementando a pesquisa com AJAX

- Implemente uma requisição para um determinado método com o jQuery, para isso, basta definirmos qual método utilizaremos para enviar essa requisição (POST ou GET). O jQuery nos fornece duas funções: `$.post` e `$.get`

```
<script type="text/javascript">

$(document).ready(function() {
    $("#descricao").keyup(function() {
        var texto = $('#descricao').val();
        var idCategoria = $('#categoriaProduto').val();
        $.post("pesquisarProduto", {'descricao' : texto, 'idCategoria' : idCategoria},
function(dados) {
            $('#tabelaListaProduto').html(dados);
        });
    });

    $("#categoriaProduto").change(function() {
        var texto = $('#descricao').val();
        var idCategoria = $('#categoriaProduto').val();
        $.post("pesquisarProduto", {'descricao' : texto, 'idCategoria' : idCategoria},
function(dados) {
            $('#tabelaListaProduto').html(dados);
        });
    });
});
</script>
```

Implementando a pesquisa com AJAX

- Ajuste o método `pesquisarProduto` no `ProdutoController` para receber a requisição AJAX:

```
@RequestMapping("/pesquisarProduto")
public @ResponseBody String pesquisarProduto(@RequestParam String descricao, @RequestParam Integer idCategoria,
                                             HttpServletResponse response) {

    ProdutoDao dao = new ProdutoDao();
    List<Produto> listaProduto = dao.pesquisar(descricao, idCategoria);

    StringBuilder st = new StringBuilder();

    st.append("<tr style='background-color: #E6E6E6; font-weight: bold;'>");
    st.append("<td> CÓDIGO </td>");
    st.append("<td> CATEGORIA </td>");
    ...
    st.append("</tr>");

    for (Produto produto : listaProduto) {
        st.append("<tr>");
        st.append("<td> " + produto.getCodigo() + " </td>");
        st.append("<td> " + produto.getCategoriaProduto().getDescricao() + " </td>");
        st.append("<td> " + produto.getDescricao() + " </td>");
        ...
        st.append("<td>");
        st.append("<a href='exibirAlterarProduto?id=" + produto.getId() + "'>Editar</a> &nbsp;");
        st.append("<a href='removerProduto?id=" + produto.getId() + "'>Remover</a>");
        st.append("</td>");
        st.append("</tr>");
    }

    response.setStatus(200);
    return st.toString();
}
```


Implementando a pesquisa com AJAX

→ Ajuste o método `pesquisar` no `ProdutoDao`:

```
public List<Produto> pesquisar(String descricao, Integer idCategoria) {
    try {
        List<Produto> listaProduto = new ArrayList<Produto>();
        PreparedStatement stmt = null;

        if (!descricao.equals("") && idCategoria == null) {
            stmt = this.connection.prepareStatement("SELECT * FROM produto WHERE descricao LIKE ? ORDER BY descricao");
            stmt.setString(1, "%" + descricao + "%");
        } else if (descricao.equals("") && idCategoria != null) {
            stmt = this.connection.prepareStatement("SELECT * FROM produto WHERE categoria_id = ? ORDER BY descricao");
            stmt.setInt(1, idCategoria);
        } else if (!descricao.equals("") && idCategoria != null) {
            stmt = this.connection
                .prepareStatement("SELECT * FROM produto WHERE descricao LIKE ? AND categoria_id = ? ORDER BY
descricao");
            stmt.setString(1, "%" + descricao + "%");
            stmt.setInt(2, idCategoria);
        } else {
            stmt = this.connection.prepareStatement("SELECT * FROM produto ORDER BY descricao");
        }

        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            listaProduto.add(montarObjeto(rs));
        }

        rs.close();
        stmt.close();
        connection.close();
        return listaProduto;

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

Implementando a pesquisa com AJAX

- Como utilizaremos AJAX e jQuery, você pode retirar a tag `<form>` e os botões da `<DIV>` de pesquisa:

```
...  
<hr> <h3>Pesquisar Produto</h3>  
<div>  
<form action="pesquisarProduto">  
  <p>  
    Descrição: <br />  
    <input type="text" id="descricao" name="descricao" value="{produto.descricao}">  
  </p>  
  <p>  
    Categoria: <br />  
    <select id="categoriaProduto" name="categoriaProduto" >  
      <option value=""> Selecione </option>  
      <c:forEach items="{listaCategoriaProduto}" var="obj">  
        <option value="{obj.id}"> {obj.descricao} </option>  
      </c:forEach>  
    </select>  
  </p>  
<p>  
<input type="reset" value="Limpar"> &nbsp; &nbsp; &nbsp;  
<input type="submit" value="Pesquisar">  
</p>  
</form>  
</div>
```

Implementando a pesquisa com AJAX

→ Veja o resultado na página:

[Home](#) | [Cadastrar Usuário](#) [Listar Usuario](#) | [Cadastrar Produto](#) [Listar Produto](#) | [Cadastrar Categ. de Produto](#) [Listar Categ. de Produto](#) | [Logout](#)

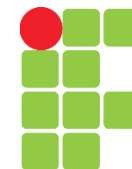
Pesquisar Produto

Descrição:

Categoria:

Lista de Produtos

CÓDIGO	CATEGORIA	DESCRIÇÃO	PREÇO DE CUSTO	PREÇO DE VENDA	GARANTIA	QUANTIDADE	IMAGEM	AÇÕES
ROC8	Notebook	Notebook DELL i7	2300.0	2600.0		9		Editar Remover

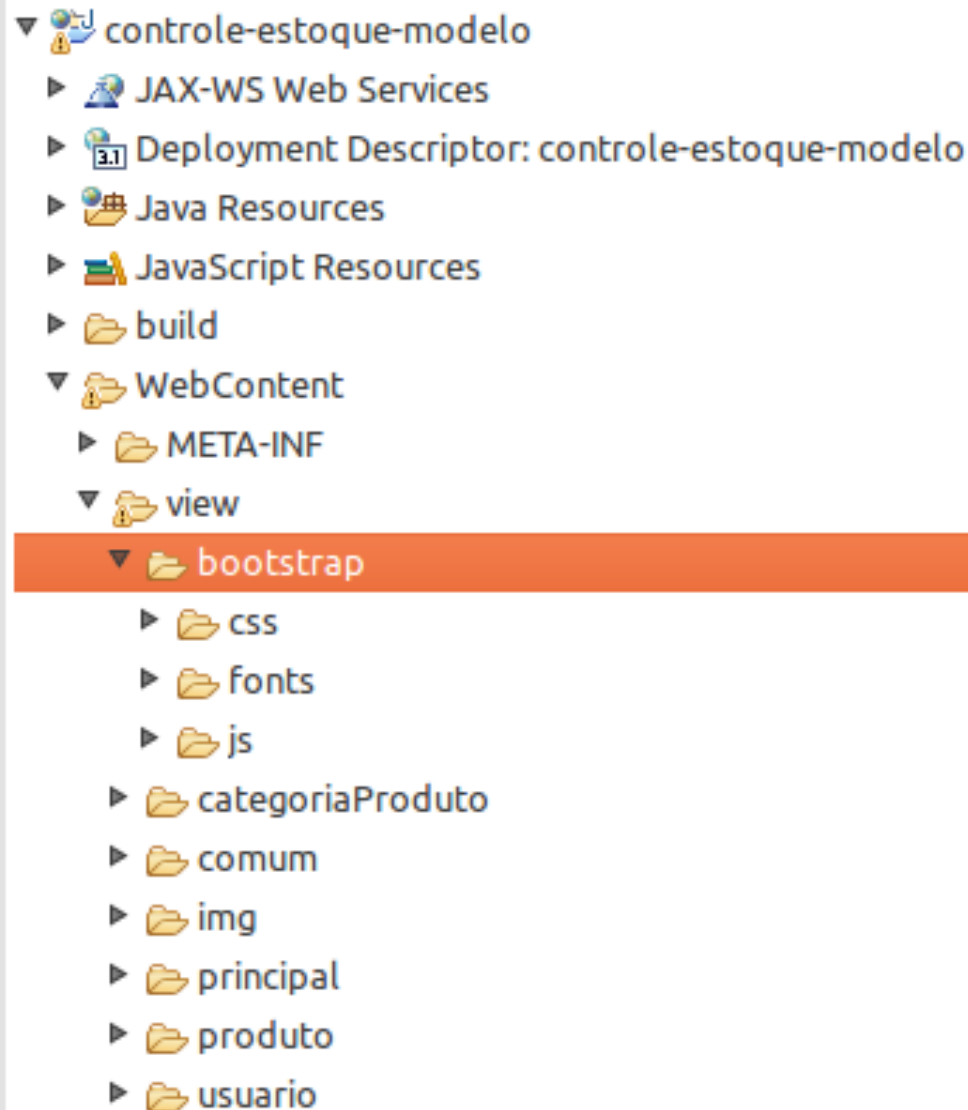


Melhorando a Aparência da Aplicação

Melhorando a Aparência da Aplicação

Adicionando o Bootstrap

→ Adicione os arquivos do *bootstrap* no projeto:



Melhorando a Aparência da Aplicação

Adicionando o Bootstrap

- Ajuste as páginas JSP, para que estas utilizem a aparência dos componentes do *bootstrap*:

Bem vindo ao sistema **EstoqueControl**, informe seu login e senha para acessar o sistema.



Login

Senha

ENTRAR

Melhorando a Aparência da Aplicação

Adicionando o Bootstrap

- Ajuste as páginas JSP, para que estas utilizem a aparência dos componentes do *bootstrap*:

Home | Estoque ▾ | Administração ▾ | Sair

Bem vindo, Roberto Alencar

EstoqueControll



Melhorando a Aparência da Aplicação

Adicionando o Bootstrap

- Ajuste as páginas JSP, para que estas utilizem a aparência dos componentes do *bootstrap*:

[Home](#) | [Estoque ▾](#) | [Administração ▾](#) | [Sair](#)

Clique **aqui** para exibir os campos de pesquisa de **Produtos**




Descrição:

Categoria:

Selecione ▾

Listagem de **Produtos**

Novo

Código	Categoria	Descrição	Preço de Custo	Preço de Venda	QTD	Imagem	Ações
RTFA	Monitor	Monitor LCD	250.0	340.0	31		<div>E</div> <div>R</div>
D14T	Notebook	Notebook ACER	1420.0	1650.0	15		<div>E</div> <div>R</div>
ROC8	Notebook	Notebook DELL i7	2300.0	2600.0	9		<div>E</div> <div>R</div>

Melhorando a Aparência da Aplicação

Adicionando o Bootstrap

- Ajuste as páginas JSP, para que estas utilizem a aparência dos componentes do *bootstrap*:

Home | Estoque ▾ | Administração ▾ | Sair

Incluir **Produto**



Código

Descrição

Categoria

Preço de Custo

Preço de Venda

Data de Garantia

Melhorando a Aparência da Aplicação

Adicionando o Bootstrap

- Ajuste as páginas JSP, para que estas utilizem a aparência dos componentes do *bootstrap*:

Home | Estoque ▾ | Administração ▾ | Sair

Alterar Produto



Código

C3PO

Descrição

Teclado ABNT

Categoria

Teclado ▾

Preço de Custo

12.5

Preço de Venda

25.0

Data de Garantia

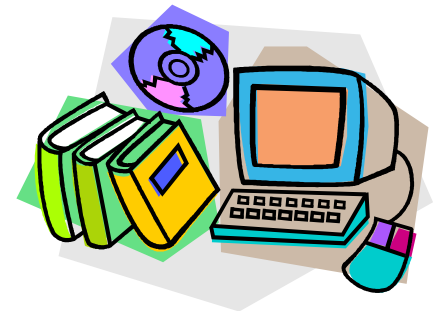
02/01/2019

Dúvidas



Incremente o Projeto Spring MVC utilizado na aula anterior:

- 1) Implemente uma pesquisa na tela de listagem de produtos utilizando jQuery e AJAX.
- 2) Melhore a aparência do seu projeto utilizando as bibliotecas do bootstrap.



Acesso a versão final do projeto

O projeto da disciplina está disponível no **GitHub**:

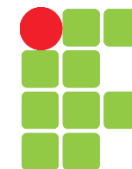
Repositório:

→ <https://github.com/robertoalencar/controle-estoque-modelo-springmvc>

URL:

→ <https://github.com/robertoalencar/controle-estoque-modelo-springmvc.git>





Obrigado pela Atenção !

Até a próxima aula.

Prof. Roberto Alencar

roberto.alencar@jaboatao.ifpe.edu.br