Controle - reduzindo o número de servlets

- **Problema de administração:** uma servlet para cada lógica diferente de cada modelo é inviável de manter em projetos reais;
- **Uma opção:** para cada modelo, agrupamos todas as operações dentro de uma mesma servlet.

Exercício 1: Modifique o projeto aula5, agrupando em um única servlet as operações de inserção, alteração e remoção do modelo aluno. Primeiro crie um servlet chamado `ControllerServlet` com o seguinte código:

```
@WebServlet("/mvc")
public class ControllerServlet extends HttpServlet {

@Override
protected void service(HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException {
      String objetivo = request.getParameter("objetivo");

      String nome = request.getParameter("nome");
      String email = request.getParameter("email");
      String endereco = request.getParameter("endereco");

      Aluno aluno = new Aluno();
      aluno.setNome(nome);
      aluno.setEmail(email);
      aluno.setEndereco(endereco);
      AlunoDAO dao = new AlunoDAO();

      if (objetivo.equals("Inserir")){
            dao.insere(aluno);
            RequestDispatcher rd =
                  request.getRequestDispatcher("/listataglib.jsp");
            rd.forward(request, response);
      } else if (objetivo.equals("Alterar")){
            aluno.setId(Long.valueOf(request.getParameter("id")));
            dao.altera(aluno);
            RequestDispatcher rd =
                  request.getRequestDispatcher("/listataglib.jsp");
            rd.forward(request, response);
      } else if (objetivo.equals("Remover")){
            aluno.setId(Long.valueOf(request.getParameter("id")));
            dao.remove(aluno);
            RequestDispatcher rd =
                  request.getRequestDispatcher("/listataglib.jsp");
            rd.forward(request, response);
      }
}
}
```

Com o cursor em cada linha com erro, acrescente os imports abaixo usando Ctrl + 1:

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import br.edu.ufabc.prograd.dao.AlunoDAO;
import br.edu.ufabc.prograd.modelo.Aluno;
```

Modifique o arquivo `listataglib.jsp` no qual construiremos uma listagem dos alunos com opção para as operações de inserção, alteração e remoção.

Acrescente o campo id de cada aluno em uma nova coluna na tabela:

```
<td align="center">${aluno.id}</td>
```

Embaixo da tabela **coloque inputs para fornecermos os dados de um novo aluno e/ou alterarmos os dados de um aluno existente**. **Não esqueça de inserir a tag <form></form> com a action indicando o servlet.** O campo do id do aluno não deve permitir alterações, como segue:

```
Id:<input type="text" id="id" name="id" readonly style="color:#AAAAAA"/>
```

- **Adicionando radiobox na tabela de listagem dos funcionários.** Crie uma nova coluna na tabela e dentro dela um radiobox cujo valor é exatamente o valor do id do aluno no Banco de Dados.

```
<td align="center">
      <input type="radio" name="group1" value="${aluno.id}">
</td>
```

- **Implementando cópia dos dados na linha da tabela para os campos abaixo.** Para isso, precisamos poder identificar a linha que contém o `radiobox` clicado, a fim de pegar as informações da linha e colocá-las nos `input's`. Associamos então um id para cada linha:

```
<tr id="row_${aluno.id}" bgcolor="#${contador.count % 2 == 0 ?
'99FFFF' : 'FFFF99' }">
```

No evento onclick do radiobox, ou seja, toda vez que o usuário clicar nesse radiobox, chamaremos uma função chamada checaRadio(value) feita em Javascript:

```
<td align="center">
      <input type="radio" name="group1" value="${aluno.id}"
      onClick="checaRadio(value)">
</td>

<script type="application/javascript">
1     function checaRadio(name){
2           document.getElementById('id').value=name;
3           var elTableRow = document.getElementById('row_'+name);
4           var elTableCells = elTableRow.getElementsByTagName("td");
5           document.getElementById('nome').value=trim(elTableCells[2
                  ].textContent);
6           document.getElementById('email').value=trim(elTableCells[3]
                  .textContent);
7           document.getElementById('endereco').value=trim(elTableCells[4]
                  .textContent);
8     }

      function trim(str) {
            return str.replace(/^\s+|\s+$/g, '');
      }
</script>
```

Na linha 2 da função `checaRadio(name)`, nós pegamos o id do aluno na tabela do Banco de Dados (que é também o nome do `radiobox` daquela linha) e colocamos dentro do `input` Id. Na linha 3, procuramos a linha que contém o `radiobox` clicado. Na linha 4, pegamos as colunas contidas na linha da tabela. Na linha 5, 6 e 7, colocamos as informações na tabela nos `input's` correspondentes.

- **Colocando vários botões de `submit` no formulário.** Adicione os botões com o valor igual ao que é esperado no servlet (Inserir, Alterar ou Remover). Essa informação deve ser colocada no `input hidden` cujo nome é `"objetivo"`. É nesse parâmetro que a servlet vai buscar a informação sobre qual a ação que deve ser realizada. Assim, no `onclick` do botão colocaremos essa informação no `input hidden`.

```
<input type="hidden" id="objetivo" name="objetivo" value="" /><br />
<input type="submit" value="Inserir" onclick="setar(value)"/>
<input type="submit" value="Alterar" onclick="setar(value)"/>
<input type="submit" value="Remover" onclick="setar(value)"/>

<script type="application/javascript">
	function setar(acao){
		document.getElementById('objetivo').value=acao;
	}
</script>
```

- **Retornando para a mesma página depois do formulário submetido.** Ao invés de redirecionar para outra página, redirecionamos a aplicação para a mesma página que requisitou a servlet.

```
RequestDispatcher rd = request.getRequestDispatcher("/listataglib.jsp");
rd.forward(request, response);
```

No entanto, se o usuário agora der um reload/refresh na página, o sistema poderá fazer novamente a operação que acabou de ser executada. Para evitar isso, podemos substituir as duas linhas acima pela linha abaixo que apenas redireciona sem repassar os dados:

```
response.sendRedirect("/aula5/listataglib.jsp");
```

- **Adicionando link "Excluir" para cada linha da tabela.** Remova o botão remover do formulário. Acrescente uma nova coluna na tabela com um link "Excluir":

```
<td>
	<a href="#" id="link_${aluno.id}"
		onclick="setar('Remover'); prepararExcl(this.id);
		document.getElementById('formulario').submit(); return
		false;">Excluir</a>
</td>
```

Modifique o form acrescentando um id "formulario" para ele:

```
<form name="formulario" id="formulario" action="mvc" method="POST">
```

No `"a href"` do link colocamos `"#"` para indicar que a ação do link será programada em outro lugar. No `onclick` do link, colocamos três funções, duas delas programadas por nós em Javascript. A primeira seta o `input hidden` na operação de remoção e a segunda coloca as informações da linha selecionada nos `input's` (é a mesma utilizada pelos `radiobox`). A última é a ação que submete o formulário. E finalmente temos o `return false`, para o caso do usuário não estar com Javascript ativado em seu browser.

```
<script type="application/javascript">
	function prepararExcl(str){
		var i=str.indexOf("_");
		str=str.substr(i+1,str.length);
		checaRadio(str);
	}
</script>
```