Programação IV
Aula 05 – JSP
Parte 2

Prof. Alex Marin

alex.marin@iffarroupilha.edu.br

Ações (*Actions*)

- Ações são tags que afetam o comportamento em JSP e a resposta retornada ao cliente.

- Durante a tradução, o container substitui uma *Action* por um código Java que corresponda a seu efeito.

- Em outras palavras, as ações são um conjunto padronizado de elementos do JSP que permitem a realização de inúmeras tarefas sem a necessidade de adição de código Java (scriptlets ou expressões).

Ações (*Actions*)

INSTITUTO FEDERAL
FARROUPILHA

- Em JSP existem 6 tipos de ações:
  - `<jsp:include>`
  - `<jsp:forward>`
  - `<jsp:useBean>`
  - `<jsp:setProperty>`
  - `<jsp:getProperty>`
  - `<jsp:plugin>`

Ação <jsp:include>

- A ação de inclusão **<jsp:include>** permite incluir uma outra página JSP na página em que o comando include é executado, tal como se o seu conteúdo estivesse nesta página.

- Sintaxe:

```
<jsp:include page="pagina.jsp" flush="true" />
```

  - O atributo flush controla se o buffer de saída da página original deve ou não ser esvaziado antes que o novo output comece a ser incluído. O valor default é `"false"`.

Ação <jsp:include>

- Ou seja, a ação de inclusão permite a incorporação do conteúdo gerado por outro documento local no output da página atual.
  - Quando uma tag <jsp:include> é encontrada, a solicitação é redirecionada do arquivo original para a página incluída, o container realiza as tarefas comuns de execução de uma página (carrega o servlet caso necessário, recompila caso tenha havido alguma alteração, etc), o output gerado é inserido no output da página original e depois o processamento da página original continua normalmente.

Ação <jsp:include>

Confira: **topo.jsp**

Confira: **Exemplo_ActionInclude.jsp**

Essa tag de ação pode também conter um corpo e, com ajuda de outra tag, podemos passar parâmetros.

Confira: **topo2.jsp**

Confira: **Exemplo_ActionInclude2.jsp**

Diretiva Include x Ação Include

- Diretiva Include:
  - Todas as páginas incluídas são compiladas juntas criando um único servlet.
  - Compartilhamento de variáveis entre as páginas (já que na verdade geram um único servlet).
  - Melhor performance tendo em vista que não requer o overhead de despachar a solicitação para a página incluída e depois incorporar a resposta no output da página original.
  - Quando uma página incluída é alterada, para ter efeito a alteração é necessário realizar forçosamente uma alteração na página principal, para assim o servlet ser recompilado.
  - A página a ser incluída não pode ser especificada dinamicamente.
  - Pode incluir apenas páginas estáticas.

Diretiva Include x Ação Include

- Ação Include:
  - Cada página incluída é um servlet separado, que deve ser carregado e executado pelo container.
  - Troca de informações entre as páginas através do objeto session ou request.
  - Como cada página incluída é um servlet separado, uma alteração em alguma delas não requer uma alteração forçada na página principal para a alteração ter efeito (recompilação automática).
  - Menores tamanhos de classe, já que o código correspondente ao arquivo incluído não é repetido nos servlets para todas as páginas JSP que fazem a inclusão.
  - Possibilidade de especificar em tempo de execução a página a ser incluída.
  - Pode incluir páginas estáticas, CGI's, servlets ou outra página JSP.

Ação <jsp:forward>

- A ação de encaminhamento **<jsp:forward>** é usada para redirecionar o controle (o objeto `request`) de uma página JSP para um outro local no servidor, o qual pode ser um documento estático, um CGI, um servlet ou outra página JSP.

- Sintaxe:

```
<jsp:forward page="pagina.jsp" />
```

Ação `<jsp:forward>`

- Quando o controle é transferido para uma outra página JSP, o container irá automaticamente atribuir um novo objeto `pageContext` à página encaminhada.

- O conteúdo gerado anteriormente é descartado.

- No entanto, os objetos `request` e `session`, serão os mesmos tanto para página original quanto para a página encaminhada.

Ação <jsp:forward>

Confira: **redirecionando.jsp**

Confira: **Exemplo_ActionForward.jsp**

Assim como na ação `include`, podemos, com a ajuda tag **<jsp:param />**, passar parâmetros:

Confira: **redirecionando2.jsp**

Confira: **Exemplo_ActionForward2.jsp**

Ação <jsp:forward>

Confira: **Formulario_ActionForward.jsp**

Confira: **Formulario_Processa.jsp**

Confira: **Formulario_Maior.jsp**

Confira: **Formulario_Menor.jsp**

Ação <jsp:useBean>

- Para tornar um bean (veremos este conceito mais adiante) disponível para uma página JSP, é necessário utilizar a sua correspondente ação.

- Sintaxe:

```
<jsp:useBean id="beanName" class="beanClass" />
...
</jsp:useBean>
```

Ação <jsp:setProperty>

- Esta ação permite ajustar o valor de uma ou mais propriedades em um bean.
  - É necessário declarar o bean com `<jsp:useBean>` antes de ajustar uma propriedade.

- Sintaxe:

```
<jsp:setProperty name="beanName"
    property="propertyName" value="propertyValue" />
```

Ação <jsp:getProperty>

- Esta ação permite acessar (obter) o valor de uma propriedade específica em um bean.

- Sintaxe:

```
<jsp:getProperty name="beanName"
                 property="propertyName" />
```

Ação <jsp:plugin>

- Esta ação possibilita a execução ou exibição de um objeto (geralmente um applet) no navegador do cliente, utilizando para tal o plug-in Java que está embutido no navegador.
  - Por questões de prioridade, o funcionamento desta ação não será vista em maiores detalhes nesta disciplina.

JavaBeans

- Para facilitar o desenvolvimento de projetos complexos, em nosso caso, aplicações web dotadas de muitas páginas e funcionalidades, é conveniente dividir o projeto em partes menores
  - As quais possuirão responsabilidades específicas, serão mais fáceis de desenvolver e testar, além de permitir a divisão do trabalho em equipe.

JavaBeans

- A presença de código Java nas páginas JSP dificulta o trabalho dos web designers.

- Além disso, cria outro problema, pois o código de controle e lógica dos negócios (Java) mistura-se ao código da apresentação (HTML).

- O uso de JavaBeans pode auxiliar na separação da camada de apresentação das demais camadas da aplicação web.

JavaBeans

- **JavaBeans** são componentes de *software* escritos na linguagem de programação Java.
  - Segundo a especificação da Sun os JavaBeans são "componentes reutilizáveis de *software* que podem ser manipulados visualmente com a ajuda de uma ferramenta de desenvolvimento".
  - Praticamente são classes escritas de acordo com uma convenção em particular. São usados para encapsular muitos objetos em um único objeto (o bean), assim eles podem ser transmitidos como um único objeto em vez de vários objetos individuais.

JavaBeans

- JavaBeans são simples classes Java, nas quais existem algumas exigências adicionais para que tal manipulação seja padronizada.

- Resumidamente, tais exigências são:
  - A classe correspondente ao bean deve possuir um construtor público (*public class MeuBean*)
  - Todos os seus campos, denominados de propriedades, independente de seus tipos, devem ser declarados como privados.

JavaBeans

- Quando for possível a obtenção do valor de um campo, deve ser implementado um método de leitura ou acesso.
- Quando for possível a alteração do valor de um campo, deve ser implementado um método de escrita ou alteração.
- Quando o bean produz eventos, devem ser implementados métodos com denominação específica.

- Em outras palavras, um bean é uma classe qualquer que pode desempenhar tarefas específicas: cálculo de valores, formatação de dados, consulta a BD, etc.

JavaBeans – Propriedades e Métodos

- As propriedades e métodos dos beans devem seguir um conjunto de regras de denominação.

- **Propriedades simples e seus métodos**

```
private <tipo> <nome>;     //propriedade
public <tipo> get<Nome>();    //método de leitura
public void set<Nome>(<tipo> valor);  //método de escrita
```

Exemplos:

```
private int idade;     //propriedade
public int getIdade();    //método de leitura
public void setIdade(int valor);  //método de escrita
```

JavaBeans – Propriedades e Métodos

- **Propriedades lógicas e seus métodos**

```
private boolean <nome>;     //propriedade
public boolean is<Nome>();    //método de leitura
public void set<Nome>(boolean valor);  //método de escrita
```

Exemplos:

```
private boolean active;     //propriedade
public boolean isActive();    //método de leitura
public void setActive(boolean valor);  //método de escrita
```

JavaBeans – Propriedades e Métodos

- **Propriedades indexadas e seus métodos**

```
private <tipo>[] <nome>;     //propriedade
```

  - Leitura e acesso aos elementos individuais

```
public <tipo> get<Nome>(int indice); //leitura individual
public void set<Nome>(int indice, <tipo> valor);
                                        //escrita individual
```

  - Leitura e acesso ao array de elementos

```
public <tipo> get<Nome>(); //leitura do array de elementos
public void set<Nome>(<tipo>[] valor);  //escrita do array
```

JavaBeans – Propriedades e Métodos

- **Propriedades indexadas e seus métodos**

Exemplos:

```
private String[] nomes;     //propriedade

public String getNomes(int indice);  //leitura individual

public void setNomes(int indice, String valor);
                                        //escrita individual

public String[] getNomes(); //leitura do array

public void setNomes(String[] valor); //escrita do array
```

Utilizando JavaBeans em JSP

- Para tornar um bean disponível para uma página JSP, é necessário utilizar uma tag de ação própria:

```
<jsp:useBean id="beanName" class="beanClass" />
```

PS. Não esqueça de criar um pacote para os beans

- Os identificadores (atributo `id`) devem ser únicos e não podem usar palavras reservadas do JSP.

Confira: **MeuPrimeiroJavaBean.java**

Confira: **UsandoMeuPrimJavaBean.jsp**

Acesso às propriedades dos JavaBeans

- Para obtermos o valor de uma propriedade específica de um bean, devemos usar outra tag de ação padrão:

```
<jsp:getProperty name="beanName"
                 property="propertyName" />
```

  - Onde o atributo `name` indica o id de um bean já disponibilizado na página, enquanto o atributo `property` se relaciona a uma propriedade do bean.

Ajuste das propriedades dos JavaBeans

- Também é possível ajustarmos o valor de uma propriedade específica de um bean, para tal devemos usar a tag de ação:

```
<jsp:setProperty name="beanName"
                 property="propertyName"
                 value="propertyValue" />
```

  - Onde o atributo `value` indica o valor a ser ajustado na propriedade especificada no atributo `property`.

Ajuste das propriedades dos JavaBeans

- Ainda, quando em uma página existe um formulário HTML dotado de um campo com o mesmo nome que uma propriedade do bean usado em outra página (carregada como resultado da ação do formulário) é possível escrever:

```
<jsp:setProperty name="beanName"
                        property="propertyName" />
```

- Caso o formulário possua campos idênticos ao bean, podemos utilizar:

```
<jsp:setProperty name="beanName" property="*" />
```

Exemplo

INSTITUTO FEDERAL
FARROUPILHA

Confira: **clienteBean.java**

Confira: **interagindoComJavaBean.jsp**

Confira: **exibeDadosClientes.jsp**

Expression Language (EL)

- Permite a realização de operações aritméticas e lógicas, bem como efetuar a leitura de propriedades de JavaBeans.

- Seu propósito é facilitar o acesso a informações contidas no servidor e exibi-las na página do cliente.
  - Bastante utilizada pois sua sintaxe é mais simples do que a empregada na tags padrão do JSP.

Expression Language (EL)

- As expressões EL são delimitadas pelos símbolos **`${ }`**
  - O conteúdo desta estrutura é interpretado no momento de execução da página e imediatamente avaliado.

- Exemplos:

`${true}`	→	tipo lógico (**`boolean`**)
`${75}`	→	tipo inteiro (**`int`**, **`long`**, ...)
`${-14.245}`	→	tipo ponto flutuante (**`float`** ou **`double`**)
`${"Alex Marin"}`	→	tipo caractere (**`String`**)
`${null}`	→	tipo nulo (**`null`**)

Expression Language (EL)

- O valor de campos de objetos previamente declarados também pode ser utilizado, desde que exista um método público **`getNomeDoCampo()`** para a obtenção do valor do campo denominado **`nomeDoCampo`**.

- Assim, um objeto aluno de uma classe **`Aluno`** dotada de um campo denominado **`idade`** e de um método **`getIdade()`** poderia ser utilizado em uma EL:

```
${aluno.idade >= 18}
```

Expression Language (EL)

Confira: **WelcomeBean.java**

Confira: **Welcome.jsp**