Java Web

AULA 09 – EL - EXPRESSSION LANGUAGE

Objetivos e Conceitos

- Objetivos:
  - Apresentar Expression Language (EL). Converter código Scriptlet para EL. Acesso a dados em escopos. Obtendo parâmetros de formulários. Objetos implícitos.
- Conceitos:
  - Expression Language.

Tópicos

- EL
- Acesso a Beans
- Formulários e Parâmetros
- Escopos
- Objetos Implícitos

EL

EL

Expression Language

- Linguagem de expressão
- Expressões usadas em JSPs para facilitar acesso a dados
- Uma maneira fácil de acessar
  - Java Beans
  - Listas/coleções
  - Atributos e objetos importantes (requisição, cookies, etc)
- Uma maneira de evitar a confusão dos *scriptlets*

EL

Expression Language

- Surgiu como parte do JSTL v1 (*JavaServer Pages Template Library*), para ser usada como atributos nas tags JSTL
- Foi movida para a especificação JSP 2.0, para ser usada em JSP como um todo (não somente nas tags JSTL)
- Com o JSF 1.0 foi criada uma EL específica, não conseguiram compatibilizar com a EL do JSP
- Assim, os grupos do JSF e JSP trabalharam em conjunto e criaram um EL unificada na JSR 245 (JSP 2.1 e JSF 1.2)
- JSR 341 foi a primeira JSR que apresenta a EL como uma especificação independente de tecnologia

Ler:

- https://download.oracle.com/otn-pub/jcp/el-3_0-fr-eval-spec/EL3.0.FR.pdf?AuthParam=1596565335_84e210152e3db29b6a1dbe4399aa8b0c

EL

Ativado por default

Se ocorrer um erro indicando que não é permitido, use no início da página

```
<%@page isELIgnored="false" %>
```


EL

As expressões são construídas com as seguintes sintaxes

- `${ expressão }`
- `#{ expressão }`

A diferença reside no tempo de avaliação da expressão

`${ expressão }`

- Usada em JSPs
- Avaliação imediata
- Expressão é compilada quando o JSP é compilado
- Expressão é executada quando o JSP é executado

`#{ expressão }`

- Usada em JSF
- Avaliação adiada, decidida pela tecnologia (ex, JSF)
- Expressão só é avaliada quando for seu valor for necessário

EL

Uma EL pode ser avaliada como *lvalue* ou *rvalue* de uma expressão

Ex. de expressão em Java:

```
x = 20;
```

Lado esquerdo, que recebe valor na atribuição:

lvalue

Lado direito, que é avaliado como um valor e pode ser usado em uma atribuição:

rvalue

Aqui usaremos expressões:

- Avaliadas como *rvalue*
- De avaliação imediata **`${ expressão }`**

EL

Quando encontra uma expressão dentro de **`${ expressão }`**

- Gera código para avaliar a expressão
- Coloca o resultado da expressão no local onde foi usada

Sintaxes comuns de Expressões dentro de **`${ expressão }`**

- Usando ponto : **`.`**
  - Acesso a propriedades de Java Beans (quando seguem as convenções)
  - Invocação de métodos
- Usando colchetes : **`[]`**
  - Operador genérico
  - Acesso a mapas, listas, array de objetos
  - Acesso a propriedades de Java Beans

EL

Exemplos

Acesso a propriedades de beans

```
${pessoa.nome}
```

Acesso a coleções

```
${lista[3]}
```

Acesso a mapas

```
${departamentos["depto1"]}
```

Invocação de métodos

```
${pessoa.efetuarAcao()}
```


EL

```
${pessoa.nome}
```

Bean => propriedade : retorna o valor da propriedade
Mapa => chave : retorna o valor do mapa para a chave

```
${lista[3]}
${pessoa["nome"]}
```

Bean => propriedade
Mapa => chave
Coleção => índice (inteiro ou string)
Array => índice (inteiro ou string)

EL

Executar o seguinte código

```
${pessoa.nome}
```

Comportamento é o mesmo que o seguinte *scriptlet*

```
<%
  Pessoa pessoa = (Pessoa) pageContext.findAttribute("pessoa");
  if (pessoa != null) {
    String nome = pessoa.getNome();
    if (nome != null) {
      out.print(nome);
    }
  }
%>
```


EL.

Neste exemplo

```
${pessoa.nome}
```

EL verifica se algo é **`null`** (ou o objeto, ou seu atributo)

NULL-safe : Não lança exceção

Desde o EL 2.2 (Servlets 3.0 / JSP 2.2) é possível invocar métodos

```
${pessoa.processar(param.id)}
```

EL não suporta sobrecarga de métodos, pois não faz verificação de tipos nos parâmetros

EL: Operadores

Operadores Aritméticos: `+, -, *, / (div), % (mod), - (- unário)`

Alteração da Precedência com parênteses: `(, )`

`${ (10 + 4) / 2 }`	Resultado: 7
`${ 3 div 4}`	Resultado: 0.75
`${ 1.2E4 + 1.4 }`	Resultado: 12001.4
`${ 10 mod 4 }`	Resultado: 2

EL: Operadores

Operadores relacionais: `== (eq), != (ne), < (lt), > (gt), <= (le), >= (ge)`

`${ 10.0 >= 5}`	Resultado: true
`${ (10*10) ne 100}`	Resultado: false
`${ 'a' < 'b'}`	Resultado: true
`${ 100.0 == 100}`	Resultado: true

Podem ser aplicados sobre outros valores booleanos, String, inteiro, valores em ponto flutuante

EL: Operadores

Operadores Lógicos: `&& (and), || (or), ! (not)`

```
${ (5 < 10) and (5 > 0) }
```

Condicional ternário: `? :`

```
${ (salario > 10000) ? "rico" : "pobre" }
```


EL: Operadores

Operador de teste Vazio

- **`empty`**: Verifica se um valor é nulo ou vazio

Teste: **`empty A`**

- Se **`A`** é nulo, retorna **`true`**
- Caso contrário, se **`A`** é uma string vazia, retorna **`true`**
- Caso contrário, se **`A`** é um array vazio, retorna **`true`**
- Caso contrário, se **`A`** é um mapa vazio, retorna **`true`**
- Caso contrário, se **`A`** é uma coleção vazia, retorna **`true`**
- Caso contrário, retorna **`false`**.

```
${ empty a }
```

Resultado: **`true`**, se a é nulo ou uma string vazia

```
${ !empty b }
```

Resultado: **`true`**, se b tiver algum valor; **`false`** se b for nulo ou uma string vazia

EL: Operadores

Operador de atribuição **=**

- Atribui o valor de B para A
- Se A não existe, cria no escopo da página
- Retorna o valor de B
- **${ A = B }**
- É associativo à direita: **${ A=B=C }** é o mesmo que **${ A=(B=C) }**

```
${ A = 10 } <br/>
${ B = 20 } <br/>
${ A + B }
```


EL: Operadores

Operador *;*

- **${ A ; B }**
- Primeiro avalia A e seu valor é descartado
- Depois avalia B e seu valor é retornado

```
${ x=30 ; x*2 }
```

EL: Operadores.

Precedência de Operadores

De cima para baixo, da esquerda para a direita

`[] .`
`()` (para mudar a precedência)
`-` (unário) `not ! empty`
`* / div % mod`
`+ -` (binário)
`+=`
`< > <= >= lt gt le ge`
`== != eq ne`
`&& and`
`|| or`
`? :`
`->`
`=`
`;`

EL: Literais.

Literais

- Valores lógicos : **`true`**, **`false`**
- Valor nulo : **`null`**
- Valores inteiros: **`10`**, **`50`**, **`0`**
- Valores em ponto flutuante : **`1.2`**, **`-5.8`**
- Strings : Delimitadas por `"` (`\"` para aspas) ou por `'` (`\'` para apóstrofe). Usar `\\` para barra

```
${ a != null }
${ "oi mundo!!" }
```

Scriptlets e EL.

Variáveis declaradas em *scriptlets* não estão disponíveis no EL

Precisam ser adicionadas a um escopo. Ex. escopo da página

```
<%
      int largura = 10;
      pageContext.setAttribute("largura", largura);
%>
${ largura * 2 }
```


Exemplo.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head><title>Teste de Bean</title></head>
<body>
   <%
      int idade = 20;
      pageContext.setAttribute("idade", idade);
   %>

   Total: ${ (10+20+30+40)/4 } <br />
   <b>
      ${(idade>=18) ? "Maior de idade" : "Menor de idade"}
   </b>

   <h1>${idade}</h1>
</body>
</html>
```

EL: Variáveis

O trecho anterior

```
<%
    int idade = 20;
    pageContext.setAttribute("idade", idade);
%>
```

Só foi usado porque ainda não foi visto JSTL

Este trecho cria uma variável **"idade"** e coloca-a disponível para o EL (no **pageContext**, isto é, escopo da página)

EL: Variáveis.

Pode-se usar a seguinte tag do JSTL:

```
<c:set var="idade" value="25" />
```

Cria a variável idade e atribui o valor 25

A tag **<c:set>** também tem o atributo **scope**

Deve-se adicionar a biblioteca do JSTL e a diretiva:

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Exemplo.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head><title>Teste de Bean</title></head>
<body>
   <c:set var="idade" value="25" />

   Total: ${ (10+20+30+40)/4 } <br />
   <b>
      ${(idade>=18) ? "Maior de idade" : "Menor de idade"}
   </b>

   <h1>${idade}</h1>
</body>
</html>
```


Palavras Reservadas.

```
and
or
not
eq
ne
lt
gt
le
ge
true
false
null
instanceof
empty
div
mod
```

Exercícios..

1. Execute os exemplos de EL apresentados nos slides anteriores
   a. Experimente usar variáveis com os operadores. Atribua valores usando:
      - Operador de Atribuição =
      - PageContext no Scriptlet
      - JSTL

Acesso a Beans

Acesso a Beans

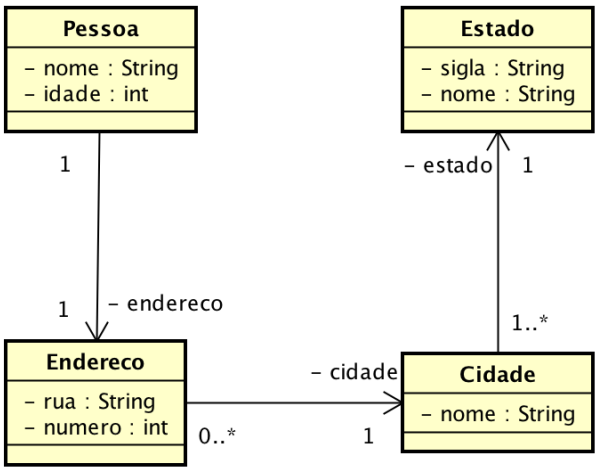
EL favorece o acesso a beans de forma simplificada

- Sintaxe `.` ou `[]`

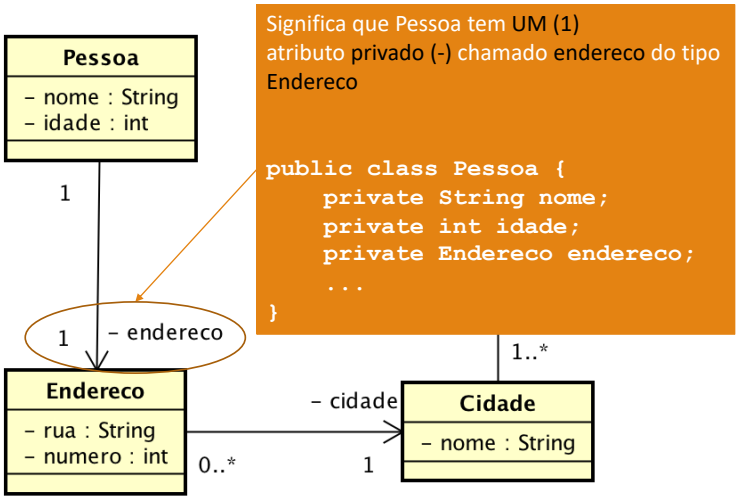
```
${pessoa.nome}
```

Também é possível caminhar pelo grafo de objetos

Acesso a Beans



Acesso a Beans



Acesso a Beans

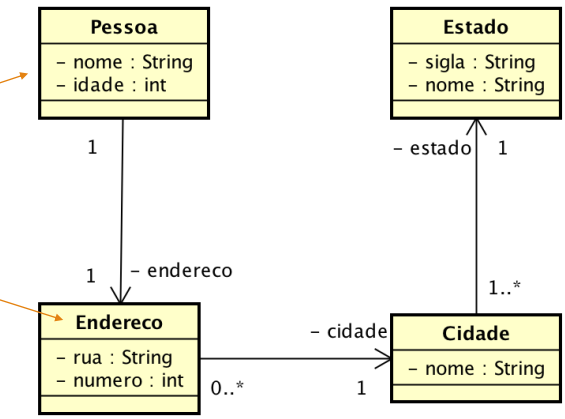
Se tivermos uma variável chamada **p** do tipo **Pessoa** em um JSP

```
<jsp:useBean id="p" class="pacote.Pessoa" />
```

Pode-se fazer em EL

```
${p.nome}
${p.idade}
```

```
${p.endereco}
```



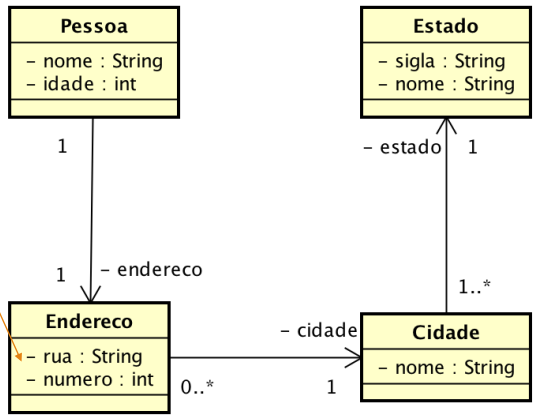
Acesso a Beans

Se tivermos uma variável chamada **p** do tipo **Pessoa** em um JSP

```
<jsp:useBean id="p" class="pacote.Pessoa" />
```

Pode-se fazer em EL

```
${p.endereco.rua}
${p.endereco.numero}
```



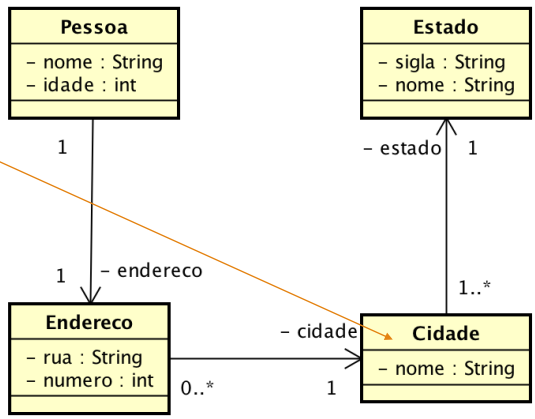
Acesso a Beans

Se tivermos uma variável chamada **p** do tipo **Pessoa** em um JSP

```
<jsp:useBean id="p" class="pacote.Pessoa" />
```

Pode-se fazer em EL

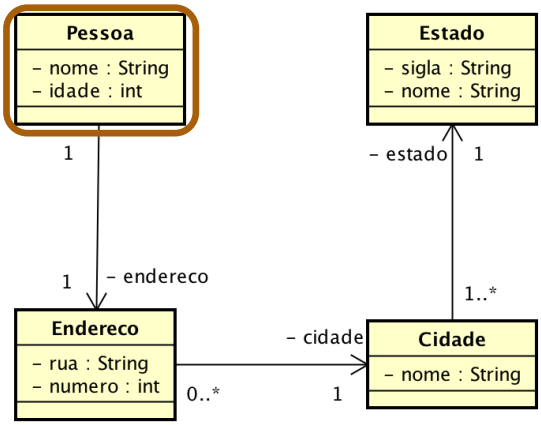
```
${p.endereco.cidade}
```



Acesso a Beans

`${p.nome}`

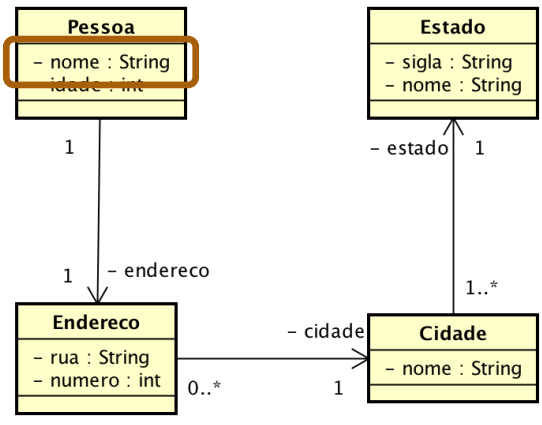
Objeto do tipo **Pessoa**



Acesso a Beans

`${p.nome}`

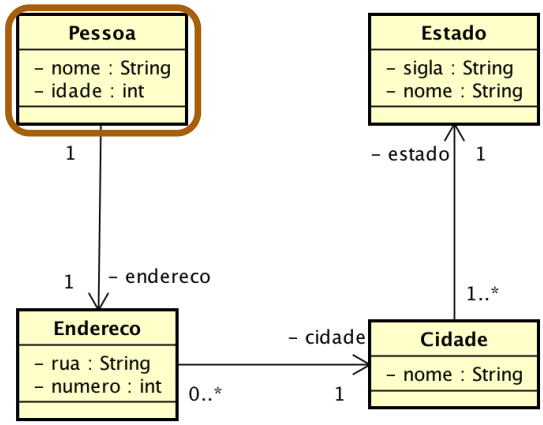
Atributo **nome** dentro de **Pessoa**



Acesso a Beans

```
${p.endereco.rua}
```

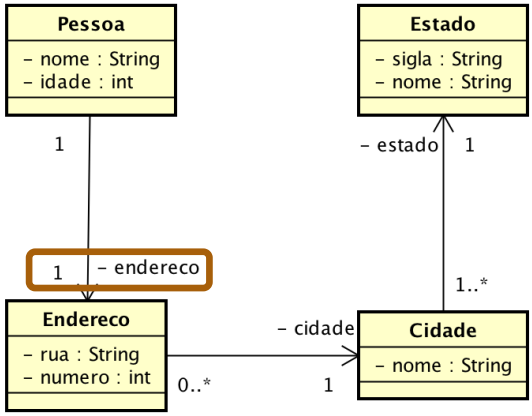
Objeto do tipo **Pessoa**



Acesso a Beans

```
${p.endereco.rua}
```

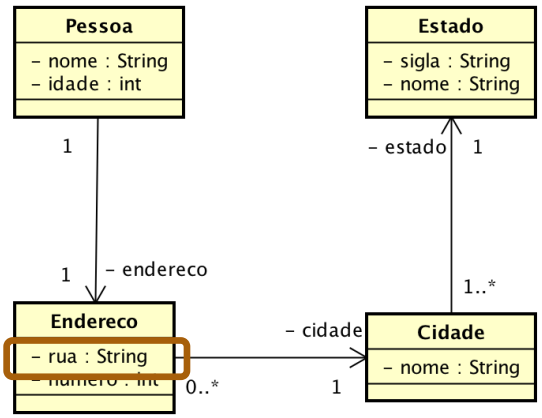
Objeto **endereco** do tipo **Endereco** que está dentro de **p**



Acesso a Beans

```
${p.endereco.rua}
```

Atributo **rua**, que está no **endereco** que está dentro de **p**



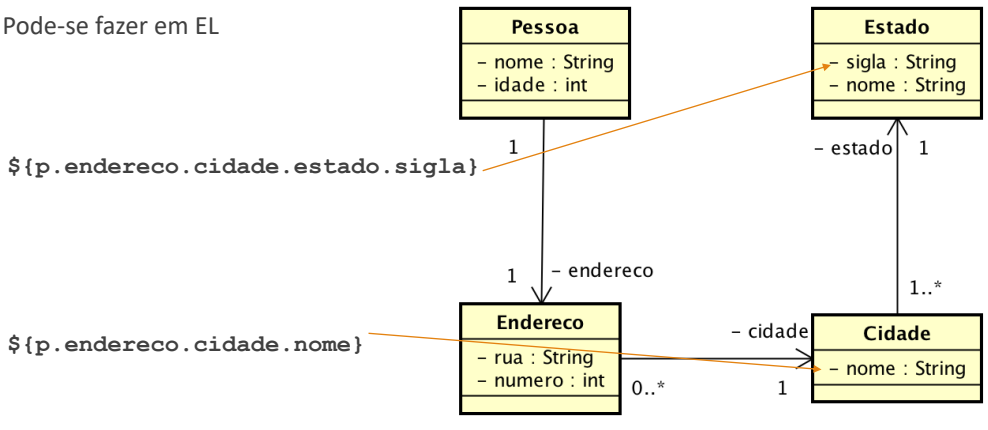
Prof. Dr. Razer A N R Montaño JAVA WEB 41

Acesso a Beans.

Se tivermos uma variável chamada **p** do tipo **Pessoa** em um JSP

```
<jsp:useBean id="p" class="pacote.Pessoa" />
```

Pode-se fazer em EL



Prof. Dr. Razer A N R Montaño JAVA WEB 42

Exemplo: Bean

```
package beans;

import java.io.Serializable;

public class Aluno implements Serializable {
   private String nome;
   private Endereco endereco;

   public Aluno() {
   }
   public String getNome() {
      return this.nome;
   }
   public void setNome(String nome) {
      this.nome = nome;
   }
   public Endereco getEndereco() {
      return this.endereco;
   }
   public void setEndereco(Endereco endereco) {
      this.endereco = endereco;
   }
}
```


Exemplo: Bean

```
package beans;

import java.io.Serializable;

public class Endereco implements Serializable {
    private String rua;
    private int numero;

    public Endereco() {
    }
    public String getRua() {
        return rua;
    }
    public void setRua(String rua) {
        this.rua = rua;
    }
    public int getNumero() {
        return numero;
    }
    public void setNumero(int numero) {
        this.numero = numero;
    }
}
```

Exemplo: Bean.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title></head><body>
      <jsp:useBean id="endereco" class="beans.Endereco" />
      <jsp:setProperty name="endereco" property="rua"
                                    value="Rua das Palmeiras" />
      <jsp:setProperty name="endereco" property="numero" value="500" />

      <jsp:useBean id="aluno" class="beans.Aluno" />
      <jsp:setProperty name="aluno" property="nome" value="Razer" />
      <jsp:setProperty name="aluno" property="endereco"
                                    value="${endereco}" />

     Nome do aluno: ${aluno.nome}<br />
     Rua do aluno: ${aluno.endereco.rua}<br />
     Número da Rua do aluno: ${aluno.endereco.numero}<br />
</body></html>
```


Exercícios..

1. Execute o exercício anterior de EL usando Beans

Formulários e Parâmetros

Recebendo Parâmetros com EL

Obter dados submetidos

Deve-se acessar os parâmetros da requisição
- Modo convencional: usar **`request.getParameter()`**

Usa-se o objeto implícito **`param`** para obter dados passados no **`request`**

Pode obter os dados passados:
- Via **POST** : através de um formulário
- Via **GET**: através de parâmetros em links:

```
"pagina.jsp?id=10&nome=Razer"
```

Recebendo Parâmetros com EL

No formulário:

```
<input type="text" name="idade" value=""/>
```

Na JSP para onde o formulário é submetido:

```
${param.idade}
```


Recebendo Parâmetros com EL

No formulário:

```
<input type="text" name="idade" value=""/>
```

Na JSP para onde o formulário é submetido:

```
${param.idade}
```

Recebendo Parâmetros com EL

Um link para a JSP:

```
<a href="teste.jsp?codigo=10">Meu link</a>
```

Na JSP para onde o link é enviado

```
${param.codigo}
```


Recebendo Parâmetros com EL.

Um link para a JSP:

```
<a href="teste.jsp?codigo=10">Meu link</a>
```

Na JSP para onde o link é enviado

```
${param.codigo}
```

Exemplo

```
<html><head><title>Teste</title></head>
<body>
<form action="inicio.jsp" method="post">
  Nome:   <input type="text" name="nome" value=""/><br/>
  E-mail: <input type="text" name="email" value=""/><br/>
  <input type="submit" value="Ok"/>
</form>
</body>
</html>
```


Exemplo

```
<html><head><title>inicio.jsp</title></head>
<body>
      <%
         String strNome = "";
         String strEmail = "";

         strNome = request.getParameter("nome");
         strEmail = request.getParameter("email");
      %>
      <h2><%= strNome %></h2>
      <h2><%= strEmail %></h2>
</body>
</html>
```

Exemplo.

```
<html><head><title>inicio.jsp</title></head>
<body>
      <h2>${param.nome}</h2>
      <h2>${param.email}</h2>
</body>
</html>
```


Exercícios..

1. Execute o exercício anterior de formulário
2. Altere o exercício anterior para, ao invés de ter um formulário passando parâmetros, ter um link passando parâmetros na URL e obtendo do JSP via **`param`**.

Escopos

Escopos

Acessa-se os 4 escopos usando os objetos implícitos:

- `pageScope`
- `requestScope`
- `sessionScope`
- `applicationScope`

Página: `${pageScope.pessoa.nome}`

Requisição: `${requestScope.pessoa.nome}`

Sessão: `${sessionScope.pessoa.nome}`

Aplicação: `${applicationScope.pessoa.nome}`

Escopos

Se for acessado como:

```
${pessoa.nome}
```

Será buscada uma propriedade **pessoa** na seguinte ordem:

1. Escopo da Página
2. Escopo da Requisição
3. Escopo da Sessão
4. Escopo da Aplicação

Usa o método **findAttribute()**

Se encontrado, chama **getNome()**

Senão, retorna vazio

Escopos

```
${pessoa.nome}
```

```
<%
  Pessoa pessoa = (Pessoa) pageContext.findAttribute("pessoa");
  if (pessoa != null) {
    String nome = pessoa.getNome();
    if (nome != null) {
      out.print(nome);
    }
  }
%>
```

Escopos

Na JSP, os dados podem ser inseridos nos escopos chamando-se o método **`setAttribute()`** em:

- **`PageContext`:** Escopo da página
- **`HttpServletRequest`:** Escopo da requisição
- **`HttpSession`:** Escopo da sessão
- **`ServletContext`:** Escopo da aplicação

Use-se:

- *Scriptlet:* `<% ... %>`
- Tag JSP: `<jsp:useBean scope="<escopo>" />`
- Tag JSTL: `<c:set scope="<escopo>" />`

Escopos..

Dentro de uma **Servlet**:

- **Não existe o escopo de página**, ele só está disponível na JSP
- O escopo da **requisição** obtém-se pelo objeto **`request`**
  - Parâmetro do **`doGet()`**/**`doPost()`**
- O escopo da **sessão** obtém-se por um **`HttpSession`:**

```
HttpSession session = request.getSession();
```

- O escopo da **aplicação** obtém-se por um **`ServletContext`**

```
ServletContext sc = getServletContext();
```

Objetos Pré-definidos

Objetos Pré-definidos: Contexto da Página

Objeto para acesso ao contexto da página JSP

- **`pageContext`** : Objeto do tipo **`PageContext`**, dando acesso a objetos como:
  - **`servletContext`**
  - **`session`**
  - **`request`**
  - etc

Em EL

```
${pageContext.request.contextPath}
```

Objetos Pré-definidos: Contexto da Página

Exemplos

- **`${pageContext.request.contextPath}`** : Obtém a raiz de contexto da aplicação
- **`${pageContext.request.cookies}`** : Obtém uma coleção dos cookies enviados
- **`${pageContext.request.method}`** : Obtém o método HTTP usado na invocação da página
- **`${pageContext.request.queryString}`** : Obtém os parâmetros da página (query string) que são passados após o "?": **`nome1=valor1&nome2=valor2`**
- **`${pageContext.request.requestURL}`** : Obtém a URL usada para acessar a página
- **`${pageContext.session.new}`** : Verdadeiro se a sessão é nova, Falso caso contrário
- **`${pageContext.servletContext.serverInfo}`** : Obtém informações sobre o contêiner

Objetos Pré-definidos: Contexto da Página.

Este exemplo de EL, em especial, é usado para criação de links:

```
<a href="${pageContext.request.contextPath}/teste.jsp">Teste</a>
```

Gera o link : **`/MinhaAplicacao/teste.jsp`**

Nota: Em JSTL há uma *tag* que cria links relativos à raiz de contexto.

Exercícios.

1. Crie dois arquivos:
   - **`origem.jsp`** : Contém um link para **`destino.jsp`**, mas este link deve ser feito usando-se:

     `${pageContext.request.contextPath}`
   - **`destino.jsp`** : Apresenta uma mensagem
   - Execute **`origem.jsp`**, verifique como o link é formado, teste o funcionamento do link

Objetos Pré-definidos: Escopos.

Objetos para acessar dados em escopos

- Mapas: chave/valor
- **`pageScope`** : Facilita o acesso a dados no escopo da **página**
- **`requestScope`** : Facilita o acesso a dados no escopo da **requisição**
- **`sessionScope`** : Facilita o acesso a dados no escopo da **sessão**
- **`applicationScope`** : Facilita o acesso a dados no escopo da **aplicação**

Em EL, para mostrar o usuário logado, por exemplo

(assumindo que o objeto usuarioLogado foi colocado na sessão e ele tem a propriedade nome)

```
${sessionScope.usuarioLogado.nome}
```

Exemplo: Bean Informacao

```
package beans;

import java.io.Serializable;

public class Informacao implements Serializable {
    private String texto;

    public Informacao() { }
    public String getTexto() {
        return texto;
    }
    public void setTexto(String texto) {
        this.texto = texto;
    }
}
```


Exemplo: Servlet InformacaoServlet

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        ServletContext ctx = getServletContext();
        Informacao infoContext = new Informacao();
        infoContext.setTexto("Bean no escopo da aplicação.");
        ctx.setAttribute("info", infoContext);

        HttpSession session = request.getSession();
        Informacao infoSession = new Informacao();
        infoSession.setTexto("Bean no escopo da sessão.");
        session.setAttribute("info", infoSession);

        Informacao infoRequest = new Informacao();
        infoRequest.setTexto("Bean no escopo da requisição.");
        request.setAttribute("info", infoRequest);

        RequestDispatcher rd = request.getRequestDispatcher("info.jsp");
        rd.forward(request, response);
    }
}
```

Exemplo: JSP info.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html><head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
</head><body>
        <%
            beans.Informacao bean = new beans.Informacao();
            bean.setTexto("Bean no escopo da página.");
            pageContext.setAttribute("info", bean);
         %>
        <h2>Escopo da Página:     ${pageScope.info.texto}</h2>
        <h2>Escopo da Requisição: ${requestScope.info.texto}</h2>
        <h2>Escopo da Sessão:     ${sessionScope.info.texto}</h2>
        <h2>Escopo da Aplicação:  ${applicationScope.info.texto}</h2>
        <h2>Sem o escopo:         ${info.texto}</h2>
</body></html>
```


Exercícios.

1. Implementar o exemplo anterior de escopo
   - Executar a Servlet **`InformacaoServlet`** e verificar o resultado que a **`info.jsp`** apresenta

Objetos Pré-definidos: Outros Dados

Objetos para acesso a outros dados

- Mapas: chave/valor
- **`cookie`** : Acessar os cookies
- **`initParam`** : Acessar dados de inicialização do contexto
- **`param`** : Obter os dados passados via parâmetro (como um **`request.getParameter()`**), retorna um valor simples
- **`paramValues`** : Obter os dados passados via parâmetro (como um **`request.getParameter()`**), retorna um array de valores
- **`header`** : Obter o valor de um *header* da mensagem HTTP
- **`headerValues`** : Obter o array de valores de um *header* da mensagem HTTP

Objetos Pré-definidos.

Em EL

```
${header["host"]}
${cookie.usuario.value}
${param.nome}
```

Exercícios..

1. Crie uma Servlet que gera um cookie e devolve uma tela HTML para o usuário. Nesta tela, coloque um link para **`teste.jsp`**. Dentro desta JSP, apresente, usando EL, o *cookie* recebido.