

Java Web

AULA 06 – JAVA BEANS

Objetivos e Conceitos

- Objetivos:
 - Apresentar Java Beans e seu uso em aplicações Java Web através de escopos.
- Conceitos:
 - Java Beans. Propriedades. Parâmetros . Escopos

Tópicos

- Java Beans
- Servlets e Java Beans
- JSP e Java Beans
- Escopo dos Beans
- loadOnStartup e o Escopo da Aplicação

Java Beans

JavaBean é uma classe Java com uma padronização nos nomes dos métodos

Também conhecidos como **Beans**

Há o conceito de **propriedade** que são acessadas por estes métodos padronizados

São componentes de software

- Componentes encapsulados
- Seus atributos só podem ser acessados via métodos
- Propriedades (Atributos) privados
- Métodos acessadores públicos

Java Beans

Padrão de métodos acessadores

```
getXXX
isXXX (quando a propriedade é booleana)
setXXX (opcional)
```

Onde Xxx é o nome do atributo da classe

Construtor

- Deve ser sem parâmetros

JavaBeans devem implementar a interface Serializable

Pode conter métodos diversos para realizar outras tarefas

Java Beans.

```
import java.io.Serializable;

public class Aluno implements Serializable {
    private int idade;
    private boolean matriculado;

    public Aluno() {
    }
    public int getIdade() {
        return this.idade;
    }
    public void setIdade(int idade) {
        this.idade = idade;
    }
    public boolean isMatriculado() {
        return this.matriculado;
    }
    public void setMatriculado(boolean matriculado) {
        this.matriculado = matriculado;
    }
}
```

Propriedades

Os métodos setXxx e getXxx/isXxx apresentam o conceito de propriedade

Assim:

- Atributo: dado/característica do objeto, contém um nome
- Propriedade: acesso ao atributo, que pode conter outro nome. Ex. `getNome()` / `setNome()` criam uma propriedade chamada `nome`

```
import java.io.Serializable;

public class Aluno implements Serializable {
    private int nIdade;
    private boolean matriculado;

    public Aluno() {
    }
    public int getIdade() {
        return this.nIdade;
    }
    public void setIdade(int nIdade) {
        this.nIdade = nIdade;
    }
    ...
}
```

Propriedades

```
import java.io.Serializable;

public class Aluno implements Serializable {
    private int nIdade;
    private boolean matriculado;

    public Aluno() {
    }
    public int getIdade() {
        return this.nIdade;
    }
    public void setIdade(int nIdade) {
        this.nIdade = nIdade;
    }
    ...
}
```

Atributo: `nIdade`

Propriedade: `idade`

Propriedades.

Propriedades Indexadas - Vetores:

```
private int[] numeros;
```

Acesso ao vetor inteiro:

```
public int[] getNumeros();
public void setNumeros(int[] value);
```

Acesso a um elemento específico:

```
public int getNumeros(int index);
public void setNumeros(int index, int value);
```

Exemplo

```
import java.io.Serializable;

public class Teste implements Serializable {
    private int[] numeros;

    public Teste() {
    }
    public int[] getNumeros() {
        return numeros;
    }
    public void setNumeros(int[] numeros) {
        this.numeros = numeros;
    }
    public int getNumeros(int index) {
        return this.numeros[index];
    }
    public void setNumeros(int index, int value) {
        this.numeros[index] = value;
    }
}
```

Exemplo..

```
import java.io.Serializable;

public class Teste implements Serializable {
    private String[] nomes;

    public Teste() {
    }
    public String[] getNomes() {
        return nomes;
    }
    public void setNomes(String[] nomes) {
        this.nomes = nomes;
    }
    public String getNomes(int index) {
        return this.nomes[index];
    }
    public void setNomes(int index, String value) {
        this.nomes[index] = value;
    }
}
```

Servlets e Java Beans

Servlets e Java Beans.

Usar um Bean em um Servlet é muito natural

Como um Bean é uma classe, seu pacote pode ser importado e objetos podem ser criados normalmente

O acesso às propriedades se dá pelos métodos *setters/getters*

Uso normal, como uma classe qualquer

Exemplo

```
package com.cursojava.beans;

import java.io.Serializable;

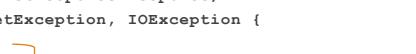
public class Aluno implements Serializable {
    private int idade;
    private String nome;

    public Aluno() {
    }
    public int getIdade() {
        return this.idade;
    }
    public void setIdade(int idade) {
        this.idade = idade;
    }
    public String getNome() {
        return this.nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Exemplo..

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.cursojava.beans.Aluno;

@WebServlet(urlPatterns = {" /Teste"})
public class Teste extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        Aluno a = new Aluno();
        a.setNome("Razer");
        PrintWriter out = response.getWriter();
        resp.setContentType("text/html");
        out.println("<html><head><title>Teste</title></head><body>");
        out.println("Teste de JavaBeans: " + a.getNome());
        out.println("</body></html>");
        out.flush();
    }
}
```



Acesso ao Java Bean

JSPs e Java Beans

Prof. Dr. Razer A N R Montaño

JAVA WEB

JSPs e Java Beans.

Usa-se os Java Beans através de tags

Todo Java Bean (objeto) em JSP existe em um escopo

- Escopo default é a página: o objeto existe enquanto a página JSP estiver sendo processada
- Escopo define em que momentos o objeto está criado e é visível

Tag de Criação/Referência a um Bean:

```
<jsp:useBean />
```

Tag para setar uma Propriedade de um Bean

```
<jsp:setProperty />
```

Tag para obter uma Propriedade de um Bean:

```
<jsp:getProperty />
```

jsp:useBean

Exemplo

```
<jsp:useBean id="nome" class="pacote.Classe" />
```

Onde

- **id** : identificador da variável que irá referenciar o Bean
- **class** : é o nome da classe, contendo nome do pacote, a classe precisa ter o construtor sem parâmetros (ou não ter construtor) e não pode ser abstrata
- **scope**: define em que escopo o bean será criado, por default na página

Se o *bean* já existir (na página), referencia-o (recupera) para uso com o nome "**nome**"

Se o *bean* ainda não existir, cria-o :

```
pacote.Classe nome = new pacote.Classe()
```

jsp:useBean.

jsp:useBean pode ser condicional, adicionando um corpo à tag

```
<jsp:useBean id="nome" class="pacote.Classe" >  
    ...  
</jsp:useBean>
```

Neste caso, se o bean for CRIADO ao invés de referenciado, executa os comandos do corpo da tag

Exemplo 1.

JSP

```
<jsp:useBean id="data" class="java.util.Date" scope="session" >  
    <h3> Data Criada </h3>  
</jsp:useBean>  
<%  out.println(data); %>
```

Exemplo 1.

JSP

Este trecho só é executado quando a data for criada

```
<jsp:useBean id="data" class="java.util.Date" scope="session">  
    <h3> Data Criada </h3>  
</jsp:useBean>  
<%  out.println(data); %>
```

Mostra a data criada ou recuperada

scope="session"

Veremos escopos depois.

Isso é para que a data esteja criada após o refresh

Exemplo 2

```
package beans;  
import java.io.Serializable;  
public class Pessoa implements Serializable {  
    private String nome;  
    public Pessoa() {  
        this.nome = "Desconhecido";  
    }  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
}
```

Exemplo 2.

```
<%@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>
        <h1>Teste de Propriedade</h1>
        <jsp:useBean id="p" class="beans.Pessoa" scope="session">
            Bean criado
        </jsp:useBean>
    </body>
</html>
```

jsp:getProperty.

Para obter o valor de uma propriedade de um Bean:

```
<jsp:useBean id="aluno" class="com.cursojava.Aluno" />
...
<jsp:getProperty name="aluno" property="nome" />
```

Onde

- **name**: identificador do Bean, igual ao atributo "id" de `<jsp:useBean />`
- **property**: é o nome da propriedade a ser lida

Comportamento

- Na posição em que for colocado, a tag será substituída pelo valor da propriedade "nome" do Bean "aluno"

Equivale a:

```
out.println( aluno.getNome() );
```

Exemplo

```
package beans;
import java.io.Serializable;
public class Pessoa implements Serializable {
    private String nome;
    public Pessoa() {
        this.nome = "Desconhecido";
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Exemplo.

```
<%@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>
        <h1>Teste de Propriedade</h1>
        <jsp:useBean id="p" class="beans.Pessoa" />
        <jsp:getProperty name="p" property="nome" />
    </body>
</html>
```

jsp:setProperty

Para setar uma propriedade de um Bean usa-se `<jsp:getProperty />`:

```
<jsp:useBean      id="aluno"    class="com.cursojava.Aluno" />
...
<jsp:setProperty name="aluno"  property="nome"
                 value="Razer" />
```

Onde

- **name**: identificador do Bean, igual ao atributo "id" do `<jsp:useBean />`
- **property**: é o nome da propriedade a ser setada
- **value**: valor que a propriedade receberá. Os dados são automaticamente convertidos

Comportamento

- Colocará o valor "Razer" na propriedade "nome" do Bean "aluno" (com.cursojava.Aluno)

Mais ou menos equivale a:

```
aluno.setNome("Razer");
```

jsp:setProperty.

O `<jsp:setProperty />` também é usado comumente de forma condicional na criação/referência de um Bean

- Usado "dentro" de `<jsp:useBean />`
- Se o *bean* já existir, não sobrescreve a propriedade
- Se o *bean* não existir, cria e inicializa a propriedade

Use-se useBean com corpo

```
<jsp:useBean id="aluno" class="com.cursojava.Aluno" >
  <jsp:setProperty name="aluno"  property="nome"
                   value="Razer" />
</jsp:useBean>
```

Exemplo

```
package beans;
import java.io.Serializable;
public class Pessoa implements Serializable {
    private String nome;
    public Pessoa() {
        this.nome = "Desconhecido";
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Exemplo.

```
<%@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>
        <h1>Teste de Propriedade</h1>
        <jsp:useBean id="p" class="beans.Pessoa" />
        <jsp:setProperty name="p" property="nome" value="Razer" />
        <jsp:getProperty name="p" property="nome" />
    </body>
</html>
```

Preenchimento por Parâmetros

Neste caso assumimos que o JSP é invocado a partir de um formulário

Em MVC não usaremos esta característica

- Um JSP será sempre invocado a partir de uma Servlet
- Aqui serão apresentados apenas exemplos

```
<html>
<head><title>Teste</title></head>
<body>
<form action="teste.jsp" method="post">
    Nome: <input type="text" name="usuario"/>
    <input type="submit" value="Enviar"/>
</form>
</body>
</html>
```

Preenchimento por Parâmetros

Para se obter os dados passados em `teste.jsp`, pode-se (PODE, MAS NÃO DEVE) usar Scriptlet para setar uma propriedade:

```
<jsp:useBean id="p" class="pacote.Pessoa" />
<% p.setNome(request.getParameter("usuario")); %>
```

Preenchimento por Parâmetros

Ou pode-se usar setProperty com *scriptlet* (Argh!)

```
<jsp:useBean id="p" class="pacote.Pessoa" />
<jsp:setProperty name="p" property="nome"
    value='<%=request.getParameter("usuario") %>' />
```

Preenchimento por Parâmetros

Solução interessante:

- Usar atributo **param** no setProperty

O seguinte trecho de código

```
<jsp:useBean id="p" class="pacote.Pessoa" />
<jsp:setProperty name="p" property="nome" param="usuario" />
```

Colocará o parâmetro 'usuario' passado do formulário como valor da propriedade 'nome' de 'p'

Neste caso, a tag <jsp:setProperty /> automaticamente executa um

```
p.setNome( request.getParameter("usuario") );
```

Preenchimento por Parâmetros

Melhor forma de usar:

- Alterar o formulário para que o nome dos campos seja igual ao nome das propriedades do bean

```
<html>
<head><title>Teste</title></head>
<body>
    <form action="teste.jsp">
        Nome: <input type="text" name="nome" />
        <input type="submit" value="Enviar" />
    </form>
</body>
</html>
```

Preenchimento por Parâmetros

O seguinte trecho de código

```
<jsp:useBean id="p" class="pacote.Pessoa" />
<jsp:setProperty name="p" property="nome" />
```

Executa automaticamente um `getParameter()` do parâmetro 'nome' passado do formulário, e faz o `setNome()` em seguida

A tag `<jsp:setProperty />` automaticamente executa um

```
p.setNome( request.getParameter("nome") );
```

Preenchimento por Parâmetros.

Para obter vários parâmetros com um só comando:

Fazer todos os campos do formulário terem nomes iguais às propriedades do bean sendo preenchido

Então usa-se a propriedade asterisco : "*"

```
<jsp:useBean id="p" class="pacote.Pessoa" />
<jsp:setProperty name="p" property="*" />
```

- Varre os parâmetros da requisição
- Encontra os que coincidem com propriedades do bean
- Seta o valor da propriedade com o valor do parâmetro

Exemplo

Por exemplo:

- Seja um formulário com NOME e ENDEREÇO
- Seja um *bean* com as propriedades Nome e Endereço

Exemplo

```
package com.cursojava.beans;

import java.io.Serializable;

public class Pessoa implements Serializable {
    private String nome;
    private String endereco;

    public Pessoa() {
    }
    public String getName() {
        return this.nome;
    }
    public void setName(String nome) {
        this.nome = nome;
    }
    public String getEndereco() {
        return this.endereco;
    }
    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }
}
```

Prof. Dr. Razer A N R Montaño

JAVA WEB

39

Exemplo

```
<html>
<head><title>Teste de Bean</title></head>
<body>
    <form method="post" action="mostrar.jsp">
        Nome: <input type="text" name="nome" /><br/>
        Endereco: <input type="text" name="endereco" /><br/>
        <input type="submit" name="Ok" /><br/>
    </form>
</body>
</html>
```

Prof. Dr. Razer A N R Montaño

JAVA WEB

40

Exemplo.

```
<html>
<head><title>Teste de Bean</title></head>
<body>
<jsp:useBean id="p" class="com.cursojava.beans.Pessoa" />
<jsp:setProperty name="p" property="*" />

<br/>
<h1>Processado</h1>
    Nome:    <jsp:getProperty name="p" property="nome" /><br/>
    Endereco: <jsp:getProperty name="p" property="endereco" /><br/>
</body>
</html>
```



Exercícios..

1. Executar os exemplos e exercícios de Servlets e JSP neste slide

Escopo dos Java Beans

Escopos.

Páginas JSP manipulam Beans

Alguns objetos são criados pelo próprio Contêiner, outros pelo desenvolvedor

Quando o desenvolvedor cria, precisa indicar o quanto disponível estará o objeto

- Por default, um objeto criado em um JSP é visível enquanto o JSP estiver sendo processado
- Também conhecido como escopo da PÁGINA (**page**)
- Alguns escopos estão também disponíveis em Servlets

Define-se o escopo pelo atributo **scope** da tag **jsp:useBean**

Quatro tipos:

- Página (**page**)
- Requisição (**request**)
- Sessão (**session**)
- Aplicação (**application**)

Escopo da Página

page: Página (default)

- Se nada for indicado, este é o escopo default
- Válidos somente na página em que são criados
- Quando a resposta é enviada de volta ao navegador, o objeto é perdido
- Se for feito um redirecionamento, o objeto também é perdido
- Não está disponível em Servlets
- Referenciado por: `pageContext`

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

Escopo da Página.

pageContext

- Objeto implícito em toda JSP
- Estende de `JspContext`
- Contém informações importantes sobre a página JSP
- Alguns métodos
 - `getRequest()` : obtém o objeto `request`
 - `getResponse()` : obtém o objeto `response`
 - `getSession()` : obtém o objeto `session`
 - `getServletContext()` : obtém o `ServletContext`
 - `setAttribute(String, Object)` : (`JspContext`) seta um bean no escopo da página
 - `getAttribute(String)` : (`JspContext`) obtém um bean do escopo da página
 - `setAttribute(String, Object, int - escopo)` : (`JspContext`) seta um bean no escopo passado como parâmetro
 - `getAttribute(String, int - escopo)` : (`JspContext`) obtém um bean guardado no escopo passado como parâmetro (ou `null` se não encontrar)
 - `findAttribute(String)` : (`JspContext`) procura e retorna um bean de algum escopo específico (`page`, `request`, `session`, `application`), se não encontrar retorna `null`

Exemplo: pageContext

```
<html>
<body>

<%
    String meuNome = "Razer";
    pageContext.setAttribute("usuario",
                            meuNome,
                            PageContext.SESSION_SCOPE);
%>

<a href="vaiLa.jsp">Outra Página</a>

</body>
</html>
```

Exemplo: pageContext.

```
<html>
    <head><title>JSP Page</title></head>
    <body>
        <%
            String usr = (String)pageContext
                            .getSession()
                            .getAttribute("usuario");
        %>
        <h1>Nome: <%=usr %></h1>
    </body>
</html>
```



Exercícios.

1. Executar o exercício de pageContext

Escopo da Requisição

request: Requisição

- Válidos durante a requisição
- Visíveis nas páginas que processam a mesma requisição onde são criados
- Depois que o Contêiner processa a requisição, o bean é perdido
- Fica visível mesmo que seja feito um forward, include ou se for usado um RequestDispatcher
- Referenciados por: **request** (ServletRequest)
 - Acessíveis em Servlets na mesma requisição

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

Escopo da Requisição

Em Servlets usa-se o escopo da Requisição com:

- Para setar um dado no escopo da requisição

```
request.setAttribute(<nome>, <objeto>);
```

- Para ler um dado do escopo da requisição

```
<Tipo> ret = (<Tipo>) request.getAttribute(<nome>);
```

Escopo da Requisição

Assim, se uma Servlet redireciona para uma JSP usando:

```
pacote.Pessoa p = new pacote.Pessoa();
request.setAttribute("eu", p);
RequestDispatcher rd = getServletContext().
                        getRequestDispatcher("/principal.jsp");
rd.forward(request, response);
```

Na JSP (principal.jsp) consegue-se obter o parâmetro com:

```
<jsp:useBean id="eu" class="pacote.Pessoa" scope="request" />
```

Escopo da Requisição

Assim, se uma Servlet redireciona para uma JSP usando:

```
pacote.Pessoa p = new pacote.Pessoa();
request.setAttribute("eu", p);
RequestDispatcher rd = getServletContext().
    getRequestDispatcher("/principal.jsp");
rd.forward(request, response);
```

Na JSP (principal.jsp) consegue-se obter o parâmetro com:

```
<jsp:useBean id="eu" class="pacote.Pessoa" scope="request" />
```

Escopo da Requisição

Da mesma forma, se um JSP armazena dados no escopo da requisição e redireciona:

```
<jsp:useBean id="eu" class="pacote.Pessoa" scope="request" />
<%
    RequestDispatcher rd = getServletContext().
        getRequestDispatcher("MinhaServlet");
    rd.forward(request, response);
%>
```

Na Servlet obtém-se da seguinte forma:

```
pacote.Pessoa p = (pacote.Pessoa) request.getAttribute("eu");
```



Exercícios

1. Escreva uma aplicação com os seguintes arquivos:
 - a. **Arquivo index.html**: com um formulário contendo nome e e-mail que submete para a Servlet Processa
 - b. **Java Bean Pessoa**: contendo nome e e-mail
 - c. **Servlet Processa**: que recebe o nome e o e-mail, cria uma pessoa preenchendo com nome e e-mail recebidos do formulário. Redireciona para a JSP principal.jsp, passando a pessoa preenchida como parâmetro no escopo da requisição
 - d. **JSP principal.jsp**: que recebe a pessoa no escopo da requisição e apresenta na tela



Exercícios

2. Escreva uma aplicação com os seguintes arquivos:
 - a. **Arquivo index.html**: com um formulário contendo nome e e-mail que submete para a JSP principal.jsp
 - b. **Java Bean Pessoa**: contendo nome e e-mail
 - c. **JSP principal.jsp**: obtém os dados do formulário, preenche um objeto Pessoa no escopo da requisição (usando `jsp:useBean`) e redireciona para a servlet Processa
 - d. **Servlet Processa**: obtém a pessoa a partir da requisição e mostra seus dados

DICA para receber dados na JSP:

```
<jsp:useBean id="eu" class="pacote.Pessoa" scope="request" />
<jsp:setProperty name="eu" property="nome"
                 value='<%=request.getParameter("nome") %>' />
```



Exercícios.

2. Escreva uma aplicação com os seguintes arquivos:

- Arquivo principal (request.getParameter())
- Java Bean que recebe e e-mail que submete para a JSP
- JSP principal que recebe um objeto e coloca no value do bean
- Servlet que processa a requisição e mostra seus dados

Prestar atenção em:

- Apóstrofe no valor da value
- <%= para executar uma expressão
- Sem ; no final da expressão %>

DICA para receber e-mail:

```
<jsp:useBean id="eu" class="pacote.Pessoa" scope="request" />
<jsp:setProperty name="eu" property="nome"
    value='<%=request.getParameter("nome") %>' />
```

Coloca o objeto no escopo da requisição

Escopo da Sessão

session: Sessão

- Validos durante o uso de sessão
- Automaticamente armazena e recupera da sessão
- Visíveis por todas as Servlets e JSPs que são acessíveis por um usuário único
- Visíveis mesmo se o usuário navegar entre várias páginas
- Referenciados por: **session** (HttpSession)
 - Acessíveis em servlets e JSPs

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

Escopo da Sessão

Nas Servlets são acessíveis a partir de HttpSession

- Para setar um dado na Sessão

```
HttpSession session = request.getSession();
...
session.setAttribute(<nome>, <objeto>);
```

- Para obter um dado da Sessão

```
HttpSession session = request.getSession();
...
<Tipo> ret = (<Tipo>) session.getAttribute(<nome>);
```

Escopo da Sessão

Assim, se uma Servlet armazena dados na sessão:

```
pacote.Pessoa p = new pacote.Pessoa();
HttpSession session = request.getSession();
session.setAttribute("recebedor", p);
```

Na JSP (principal.jsp) consegue-se obtê-lo com:

```
<jsp:useBean id="recebedor" class="pacote.Pessoa"
              scope="session" />
```

Escopo da Sessão

Assim, se uma Servlet armazena dados na sessão:

```
pacote.Pessoa p = new pacote.Pessoa();
HttpSession session = request.getSession();
session.setAttribute("recebedor", p);
```

Na JSP (principal.jsp) consegue-se obtê-lo com:

```
<jsp:useBean id="recebedor" class="pacote.Pessoa"
              scope="session" />
```

Escopo da Sessão

Da mesma forma, se a JSP armazena algo nas sessão

```
<jsp:useBean id="eu" class="pacote.Pessoa"
              scope="session" />
```

Na Servlet pode-se obter da seguinte forma:

```
HttpSession session = request.getSession();
pacote.Pessoa p = (pacote.Pessoa) session.getAttribute("eu");
```

Escopo da Sessão.

Da mesma forma, se a JSP armazena algo nas sessão

```
<jsp:useBean id="eu" class="pacote.Pessoa"  
scope="session" />
```

Na Servlet pode-se obter da seguinte forma:

```
HttpSession session = request.getSession();  
pacote.Pessoa p = session.getAttribute("eu");
```



Exercícios.

1. Escreva uma aplicação com os seguintes arquivos:
 - a. **Arquivo index.html**: com um formulário contendo nome e e-mail que submete para a Servlet Processa
 - b. **Java Bean Pessoa**: contendo nome e e-mail
 - c. **Servlet Processa**: que recebe o nome e o e-mail, cria uma pessoa preenchendo com nome e e-mail recebidos do formulário. Armazena a pessoa na sessão e redireciona para a JSP principal.jsp.
 - d. **JSP principal.jsp**: obtém a pessoa da sessão (usando jsp:useBean) e apresenta seus dados
2. Escreva uma aplicação com os seguintes arquivos:
 - a. **Arquivo index.html**: com um formulário contendo nome e e-mail que submete para a JSP principal.jsp
 - b. **Java Bean Pessoa**: contendo nome e e-mail
 - c. **JSP principal.jsp**: obtém os dados do formulário, preenche um objeto Pessoa no escopo da sessão (usando jsp:useBean) e redireciona para a servlet Processa
 - d. **Servlet Processa**: obtém a pessoa a partir da sessão e mostra seus dados

Mesmas dicas do exercício de escopo da requisição.

Escopo da Aplicação

application: Aplicação

- Válidas em todas as páginas da aplicação
- São tipicamente criados e populados quando a aplicação inicia, depois somente lidos pelo resto da aplicação
- Referenciados por: **application** (ServletContext)
 - Acessíveis em Servlets e JSPs
 - Em ServletContext: `setAttribute()`/`getAttribute()`

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

Escopo da Aplicação

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <title>JSP Page</title>  
  </head>  
  <body>  
    <jsp:useBean id="p" class="beans.Pessoa" scope="application"/>  
    <jsp:setProperty name="p" property="nome" value="Razer" />  
  
    <a href="Teste">Teste</a>  
  </body>  
</html>
```

Escopo da Aplicação

```

@WebServlet(name = "Teste", urlPatterns = {"/*Teste"})
public class Teste extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
        throws ServletException, IOException {
        ServletContext ctx = getServletContext();
        Pessoa pessoa = (Pessoa)ctx.getAttribute("p");

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html><head><title>Teste</title></head>");
            out.println("<body>");
            out.println("<h1>Nome da Pessoa: " + pessoa.getNome() +
                       "</h1>");
            out.println("</body></html>");
        }
    }
}

+ HttpServlet methods. Click on the + sign on the left to edit the code.
}

```

Escopo da Aplicação.

```

@WebServlet(name = "Teste", urlPatterns = {"/*Teste"})
public class Teste extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
        throws ServletException, IOException {
        ServletContext ctx = getServletContext();
        Pessoa pessoa = (Pessoa)ctx.getAttribute("p");

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html><head><title>Teste</title></head>");
            out.println("<body>");
            out.println("<h1>Nome da Pessoa: " + pessoa.getNome() +
                       "</h1>");
            out.println("</body></html>");
        }
    }
}

+ HttpServlet methods. Click on the + sign on the left to edit the code.
}

```

Mesmo nome que foi usado
no "id" do useBean



Exercícios.

1. Executar o exercício de escopo da aplicação
 2. Executar o exemplo de login com sessão a seguir
 - Java Bean : **Pessoa.java**
 - Formulário : **index.html**
 - JSP de Login : **login.jsp**
 - JSP para Listar : **listar.jsp**
- Executar de duas formas diferentes
- a. Executar diretamente o **listar.jsp**
 - Não deve aparecer dado algum de usuário na sessão
 - b. Executar a partir do **index.html** e depois o **listar.jsp**
 - Deve aparecer o dado de um usuário na sessão

Exemplo

```
package com.cursojava.beans;

import java.io.Serializable;

public class Pessoa implements Serializable {
    private String nome;
    private String endereço;

    public Pessoa() {
    }
    public String getNome() {
        return this.nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getEndereço() {
        return this.endereço;
    }
    public void setEndereço(String endereço) {
        this.endereço = endereço;
    }
}
```

Exemplo

```
<html>
<head><title>index.html</title></head>
<body>
    <form action="login.jsp" method="post">
        Nome: <input type="text" name="nome"/> <br />
        Endereço: <input type="text" name="endereco"/> <br />
        <input type="submit" value="OK"/>
    </form>
</body>
</html>
```

Exemplo

```
<html>
<head><title>login.jsp</title></head>
<body>

    <jsp:useBean id="p" class="com.cursojava.beans.Pessoa"
                  scope="session"/>
    <jsp:setProperty name="p" property="*" />

    <h2>Sessão armazenada</h2>
    <a href="listar.jsp">Ver Sessão</a>

</body>
</html>
```

Exemplo..

```
<html>
<head><title>listar.jsp</title></head>
<body>
<jsp:useBean id="p" class="com.cursojava.beans.Pessoa"
scope="session"/>

<h2>Dados da Sessão</h2>
Nome: <jsp:getProperty name="p"
property="nome" /><br />
Endereço: <jsp:getProperty name="p"
property="endereco" /><br />

</body>
</html>
```

loadOnStartup e Escopo de Aplicação

loadOnStartup.

As Servlets são iniciadas em sua primeira requisição

Mas é possível iniciar uma Servlet quando a aplicação (ou o servidor) iniciar

- Faz a inicialização como na primeira requisição
- Chama o método `init()`

Atributo `loadOnStartup` na anotação `@WebServlet`

```
@WebServlet(name = "Teste", urlPatterns = {"/Teste"}, loadOnStartup = 1)
```

O valor indica a ordem em que as Servlets serão iniciadas

Pode-se usar para adicionar um bean no escopo da Aplicação

Exemplo

```
package beans;

import java.io.Serializable;

public class ConfigBean implements Serializable {
    private String email;

    public ConfigBean() {
    }
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

Exemplo

```
package servlets;
// importações

@WebServlet(name = "StartupServlet", urlPatterns = {"/StartupServlet"}, loadOnStartup = 1)
public class StartupServlet extends HttpServlet {

    public void init(ServletConfig config) throws ServletException {
        ConfigBean conf = new ConfigBean();
        conf.setEmail("razer.anthom@gmail.com");

        ServletContext ctx = config.getServletContext();
        ctx.setAttribute("configuracao", conf);
    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // ...
    }

    + HttpServlet methods. Click on the + sign on the left to edit the code.
}
```

Exemplo.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Escopo</title>
    </head>
    <body>
        <h1>Teste de Escopo da Aplicação</h1>
        <jsp:useBean id="configuracao" class="beans.ConfigBean"
                     scope="application" />
        E-mail: <jsp:getProperty name="configuracao"
                               property="email" />

    </body>
</html>
```



Exercícios..

1. Executar o exercício de escopo da aplicação apresentado anteriormente