

Laboratório de Programação de Web Sites Dinâmicos

Java Server Pages

Tassio Sirqueira – 2019/01

JSP | Por que precisamos de algo além dos Servlets?

ExemploServlet.java

```
public class ExemploServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<html><head><title>Página</title><style>\n" +
"h1 {font-size: 2rem; font-weight: bolder; margin-bottom: 1rem;}\n" +
"p {margin-bottom: 1rem; color: tomato;}\n" +
"button {cursor: pointer; appearance: none; border-radius: 4px; font-size: 1.25rem; padding: 0.75rem 1rem; border: 1px solid navy;}\n" + "</style></head>\n" +
" <body>\n" +
"   <h1>I am a headline made with HTML</h1>\n" +
"   <p>And I am a simple text paragraph. The color of this text is styled with CSS. Click the button below to remove me through
the" + "power JavaScript.</p>\n" +
"   if (request.getSession().getAttribute("cadastro")!=null) {
        <button>Hide the text above</button>\n" +
    }
    <script>$('button').on('click', function() {\n" +
"      $('p').css('opacity', 0);\n" +
"   });</script>\n" +
" </body> \n" +
"</html>");

    }
}
```

CSS

Java

JavaScript

HTML

Java

| JSP | Introdução



- ❑ Podemos escrever conteúdo dinâmico através de Servlets.
 - ❑ Teremos muitos problemas na manutenção das nossas páginas e também na legibilidade do nosso código
- ❑ Para melhorar a manutenabilidade de websites dinâmicos, o JavaEE oferece a tecnologia **JavaServer Pages (JSP)**.
- ❑ Essa tecnologia é baseada nos Servlets que vimos na aula anterior, portanto todos os conceitos aprendidos devem ser considerados aqui.
 - ❑ Muitas vezes vamos combinar Servlet e JSPs no desenvolvimento do aplicação web dinâmicas.

JSP | Introdução



❑ Exemplo de JSP:

bemvindo.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>
        <h1>Hello World!</h1>
    </body>
</html>
```

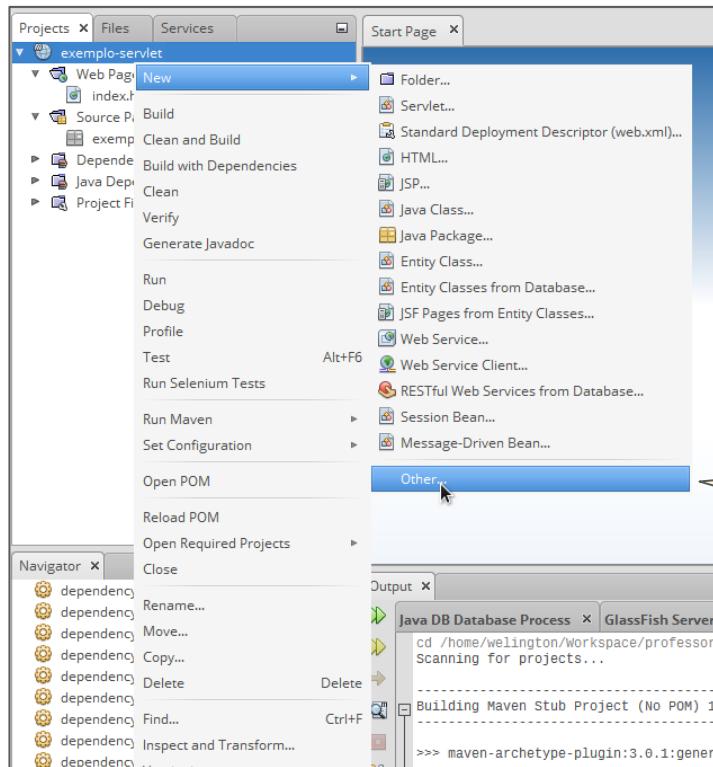
Podemos adicionar diretivas especiais por meio de tags no formato <%@ %>

Podemos escrever HTML normal em JSPs



JSP | Criando JSPs no Netbeans

☐ Criando um JSP:



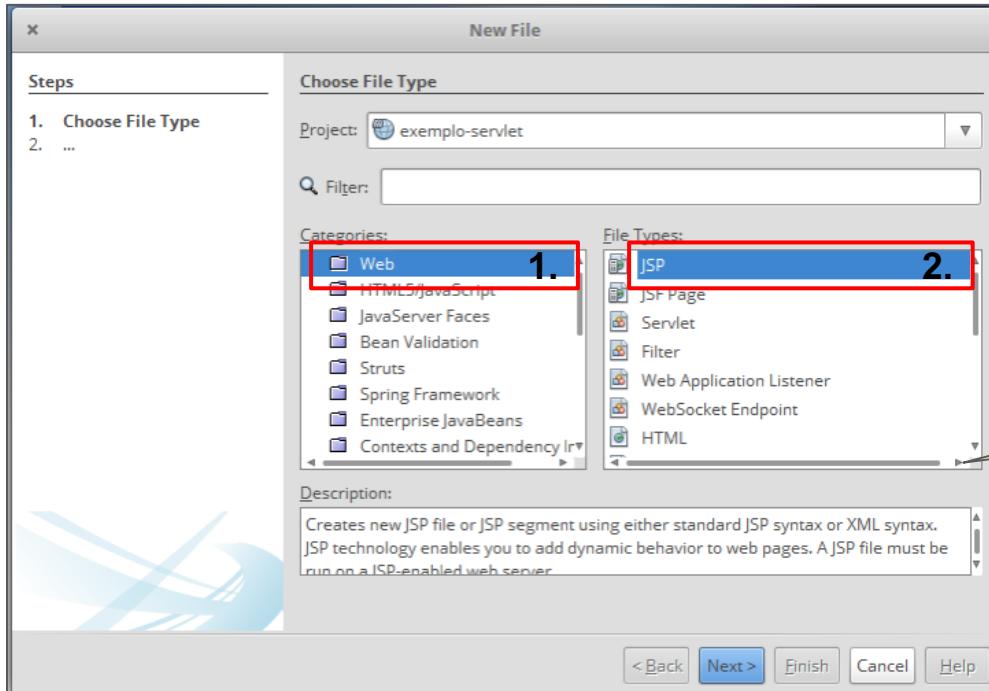
Clique com o botão direito no nome do projeto.

Selecione **New > Other...**

JSP | Criando JSPs no Netbeans



Criando um JSP:

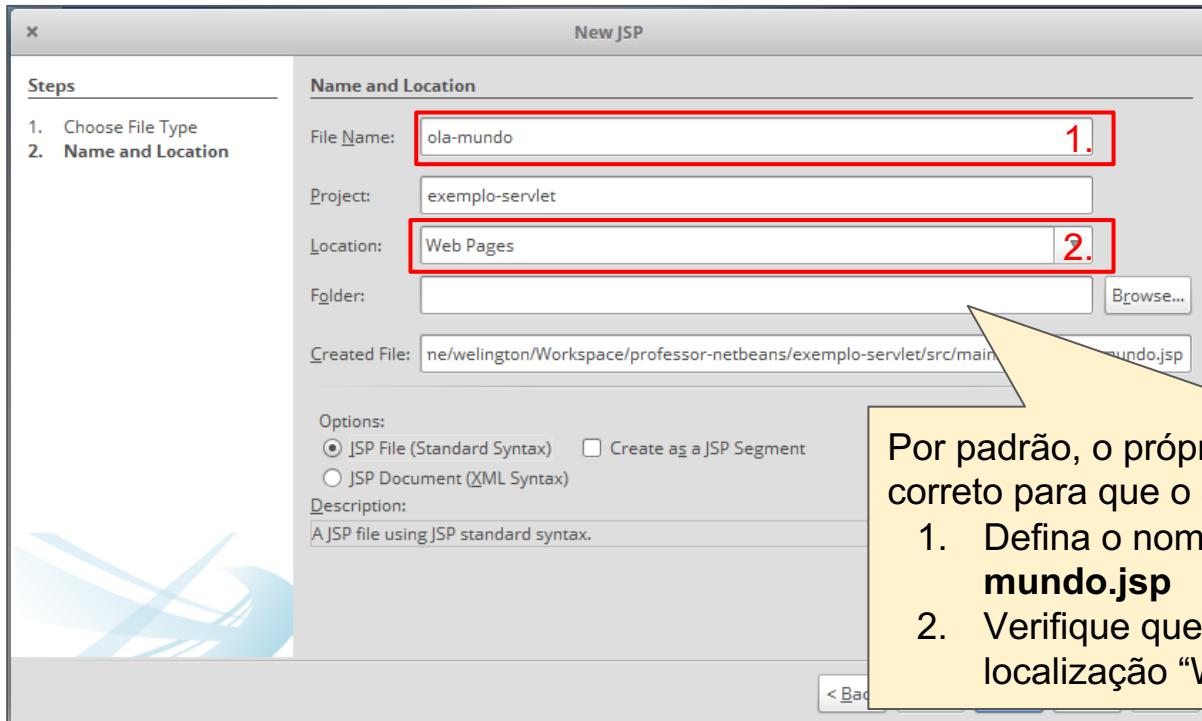


Selecione, na escolha do arquivo, a categoria
(1.) Web e o tipo de arquivo **(2.) JSP**



JSP | Criando JSPs no Netbeans

❑ Criando um JSP:



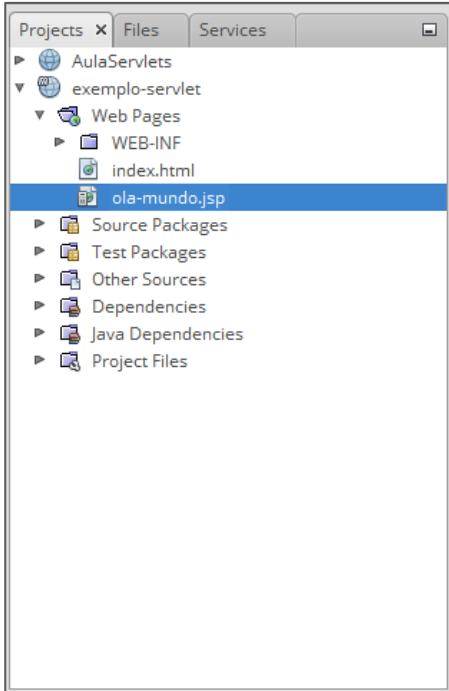
Por padrão, o próprio Netbeans irá indicar o local correto para que o JSP criados.

1. Defina o nome da página JSP. No exemplo **ola-mundo.jsp**
2. Verifique que o mesmo deve ser criado na localização “Web Pages”

JSP | Criando JSPs no Netbeans



Criando um JSP:



```
ola-mundo.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

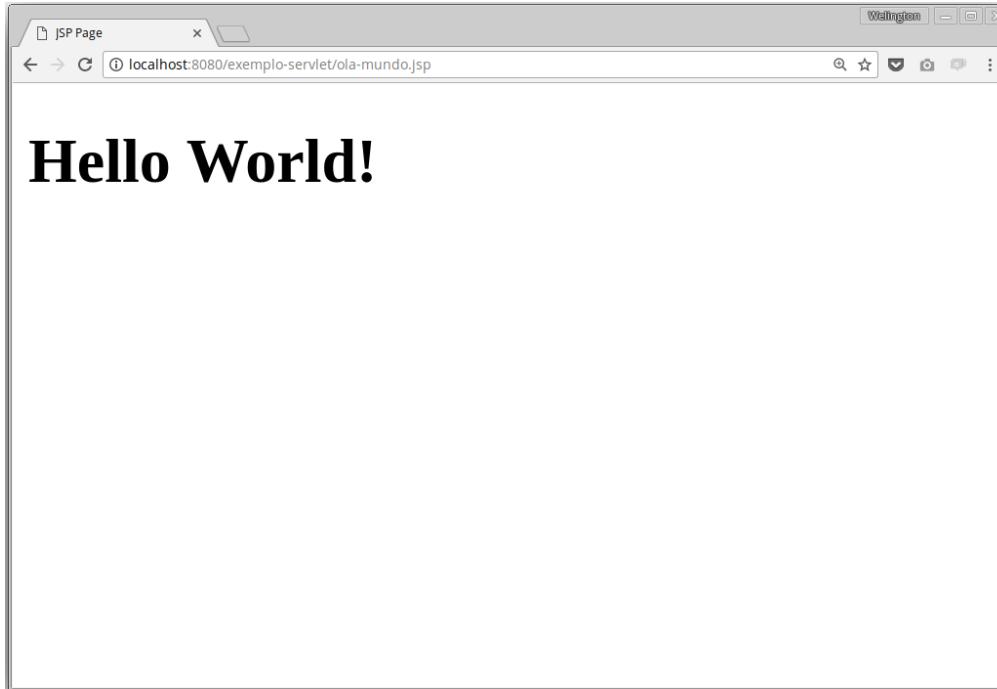
<html>

    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

| JSP | Criando JSPs no Netbeans



- ❑ Acessando o JSP criado:
 - ❑ <http://localhost:8080/exemplo-servlet/ola-mundo.jsp>



| JSP | Scriptlet

- ❑ Embora suporte HTML estático, a vantagem de se usar JSPs consiste na possibilidade de adicionar **elementos dinâmicos** às páginas.
- ❑ A forma mais simples de se fazer isso é por meio de **Scriptlets**.
 - ❑ Pequenas porções de lógica de negócio resolvidas na página.
 - ❑ Escritas em Java.
- ❑ Podemos escrever scriptlets por meio da sintaxe `<% ... %>`





JSP | Scriptlet

❑ Exemplo de scriptlet

bemvindo.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>
        <h1><% out.println("Olá Mundo!");%></h1>
    </body>
</html>
```

Podemos escrever código Java em qualquer parte da página.



JSP | Scriptlet

❑ Exemplo de scriptlet

ola-mundo.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>
        <%
            String nome = "Tassio Sirqueira";
        %>
        <h1><% out.println("Visitante: "+ nome);%></h1>
    </body>
</html>
```

Podemos escrever código Java em qualquer parte da página.

| JSP | Scriptlet | Objetos Implícitos

- ❑ Existem algumas variáveis especiais disponíveis no escopo de uma página JSP.
- ❑ São conhecidos como **Objetos Implícitos**.
- ❑ Os principais Objetos Implícitos são familiares:
 - ❑ **out**: OutputStreamer da resposta atual.
 - ❑ **request**: Requisição atual.
 - ❑ **response**: Resposta atual.
 - ❑ **session**: Sessão de usuário atual.





JSP | Scriptlet | Objetos Implícitos

Exemplo

```
contador.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>
        <%
            Integer count = (Integer) session.getAttribute("count");
            if (count == null) {
                count = 0;
            }
            count++;
            session.setAttribute("count", count);
        %>
        <h1>Você visitou essa página <% out.println(count);%> vezes</h1>
    </body>
</html>
```

Podemos re-implementar o contador usando apenas objetos implícitos do JSP.

| JSP | Scriptlet | Sintaxes especiais

- ❑ Além da sintaxe <% ... %>, existem outras sintaxes bastante úteis para a construção de páginas dinâmicas.
- ❑ Para declarar métodos e variáveis no JSP: <%! ... %>.

```
<%
    int count;
    private int incrementa() {
        return ++count;
    }
%>
```

- ❑ Para imprimir sem necessidade de **out.println** <%= ... %>.

```
<%= incrementa() %>
```

| JSP | Scriptlet | Sintaxes especiais

- ❑ Para fazer comentários que não farão parte da página:

`<%-- ... --%>.`

```
<%-- This is JSP comment --%>
```

- ❑ Para importar bibliotecas : `<%@page import="..." %>`.

```
<%@page import="java.util.Date"%>
```

| JSP | Scriptlet | Controle de Fluxo

- ❑ Podemos adicionar instruções if...else

```
<%! int day = 3; %>
<html>
    <head><title>IF...ELSE Example</title></head>

    <body>
        <% if (day == 1 | day == 7) { %>
            <p> Today is weekend</p>
        <% } else { %>
            <p> Today is not weekend</p>
        <% } %>
    </body>
</html>
```

Podemos alternar instruções Java com HTML
ou usar **out.println**.

| JSP | Scriptlet | Controle de Fluxo

- ❑ Podemos adicionar instruções switch...case

```
<%! int day = 3; %>
<html>
  <head><title>SWITCH...CASE Example</title></head>
  <body>
    <% switch(day) {
      case 1:
        out.println("It's Monday.");
        break;
      case 2:
        out.println("It's Tuesday.");
        break;
      case 3:
        out.println("It's Wednesday.");
        break;
      case 4:
        out.println("It's Thursday.");
        break;
      case 5:
        out.println("It's Friday.");
        break;
    } %>
  </body>
</html>
```

| JSP | Scriptlet | Controle de Fluxo

- ❑ Podemos adicionar instruções **for, while, e do...while**

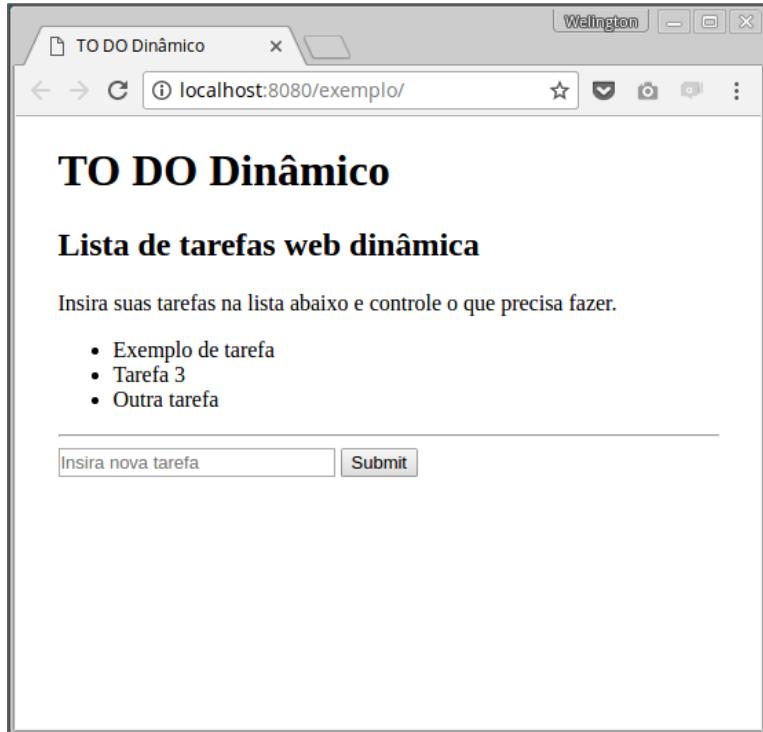
```
<%! int fontSize; %>
<html>
    <head><title>FOR LOOP Example</title></head>

    <body>
        <%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
            <font color = "green" size = "<%= fontSize %>">
                Texto
            </font><br />
        <%}%>
    </body>
</html>
```

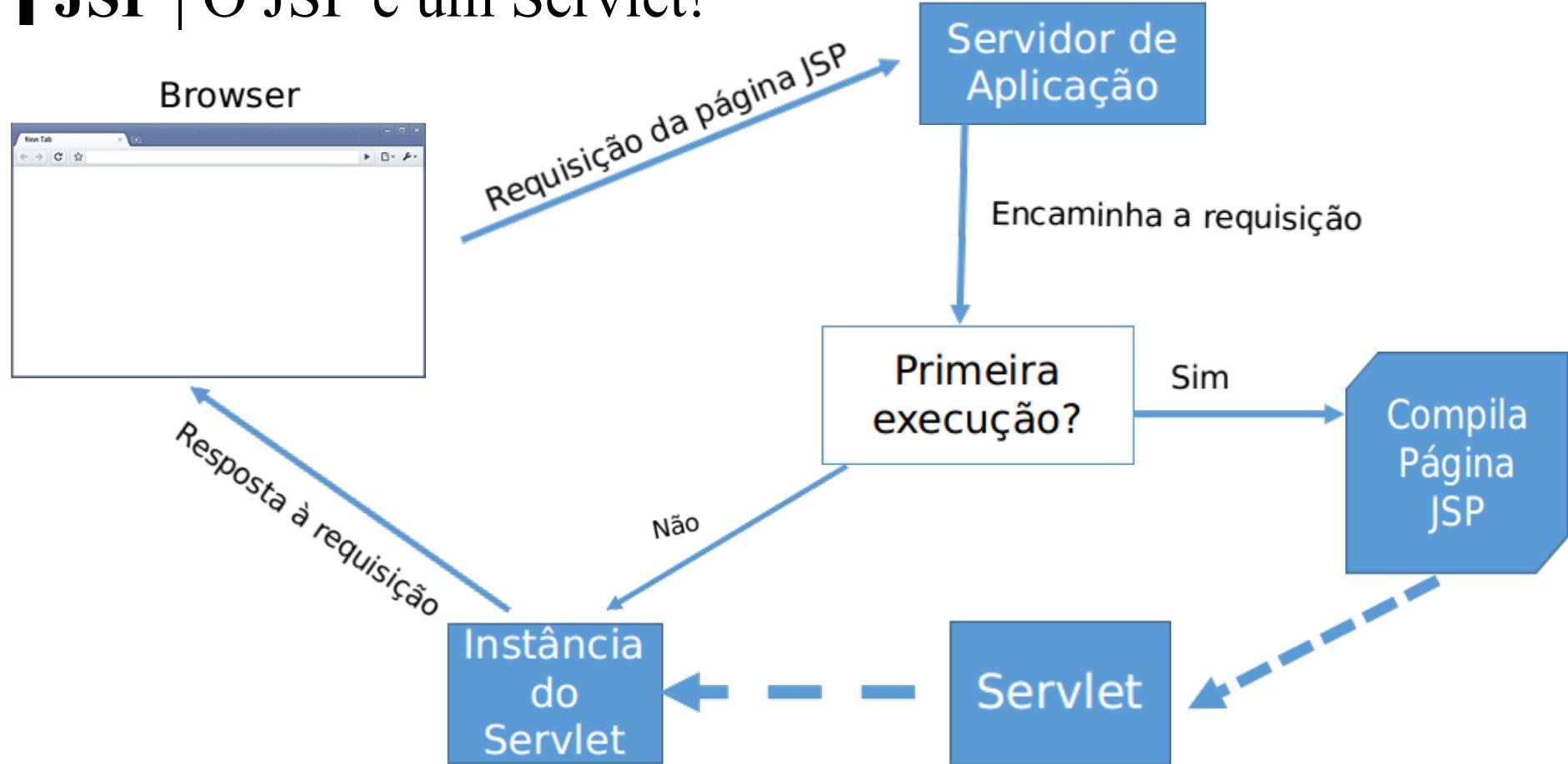
Atividade | Implementando JSP

- ❑ Utilizando o template de TO-DO list desenvolvido na atividade 4, altere a implementação para utilizar JSP.
- ❑ Envie para tassio@tassio.eti.br com o assunto

Atividade 6 - Todo List JSP com seu nome no corpo do e-mail até 02/09.



JSP | O JSP é um Servlet!



JSP | Servlets + JSP

bemvindo.jsp

```
<%@page contentType="..." pageEncoding="..."%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="..." content="...">
    <title>JSP Page</title>
  </head>
  <body>
    <h1><% out.println("Olá Mundo!");%></h1>
  </body>
</html>
```

ola_002dmando_jsp.jsp

```
out.write("\n");
out.write("\n");
out.write("\n");
out.write("\n");
out.write("<!DOCTYPE html>\n");
out.write("<html>\n");
out.write("  <head>\n");
out.write("    <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
out.write("    <title>JSP Page</title>\n");
out.write("  </head>\n");
out.write("  <body>\n");
out.write("    <!-- Comentário HTML --&gt;\n");
out.write("    ");
out.write("\n");
out.write("    ");
out.println("Olá Mundo!");
out.write("\n");
out.write("  &lt;/body&gt;\n");
out.write("&lt;/html&gt;\n");</pre>
```

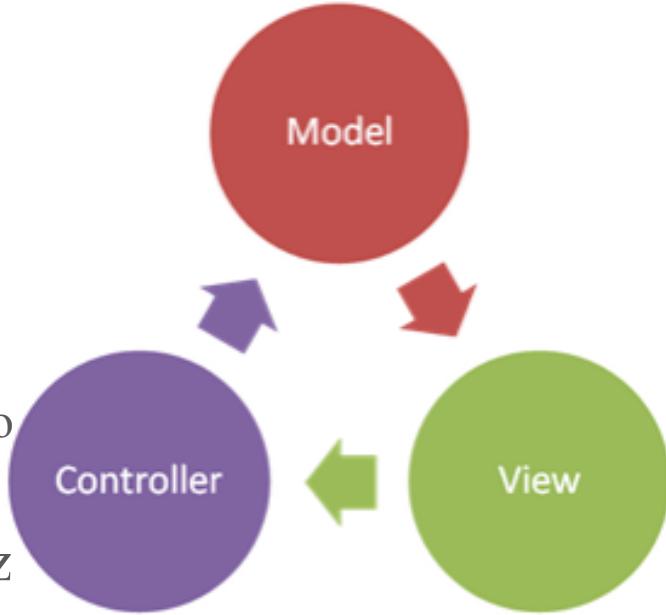
| JSP | Servlets + JSP

- ❑ O uso de JSPs possui um inconveniente equivalente ao que vimos no Servlets.
 - ❑ No **servlet** adicionamos código **HTML** no meio do código **Java**.
 - ❑ No **JSP** adicionamos código **Java** no meio do **HTML**.
- ❑ São dois tipos diferentes de código “espaguete”.
- ❑ Mas a combinação das duas tecnologias permite uma separação mais adequada de responsabilidades.
 - ❑ Para resolver essa questão temos alguns padrões arquiteturais relevantes.
 - ❑ O mais utilizado é o padrão MVC.



| JSP | Servlets + JSP | MVC

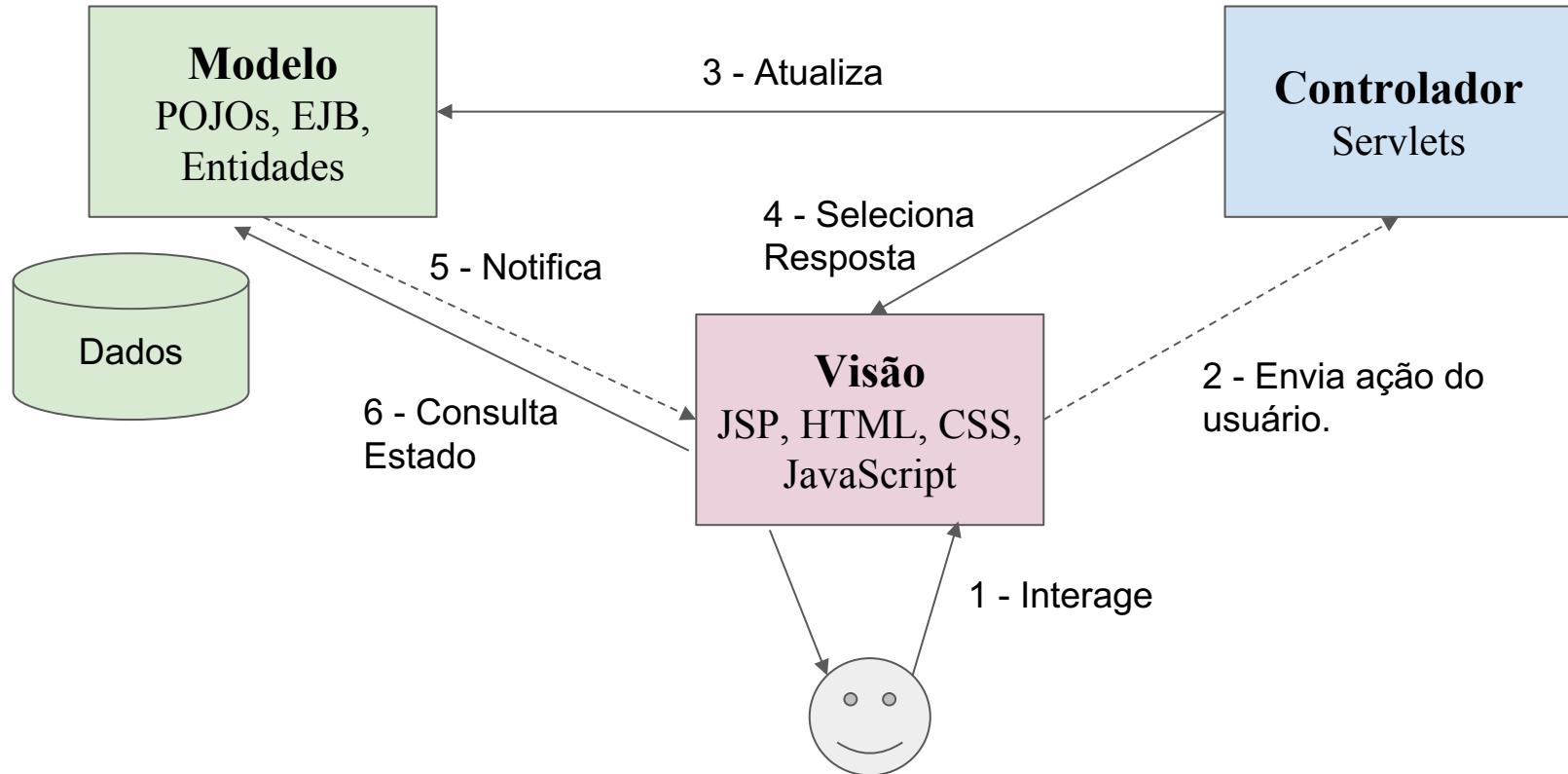
- ❑ Model View Controller é um padrão arquitetural em 3 camadas.
 - ❑ Em português: Modelo-Visão-Controlador.
 - ❑ Separa a representação da informação da interação do usuário com ela.
- ❑ O padrão MVC foi descrito pela primeira vez em 1979 por Trygve Reenskaug, que trabalhava no Smalltalk, na Xerox PARC.
- ❑ É o padrão mais utilizado na Web para a arquitetura de aplicações dinâmicas.



| JSP | Servlets + JSP | MVC

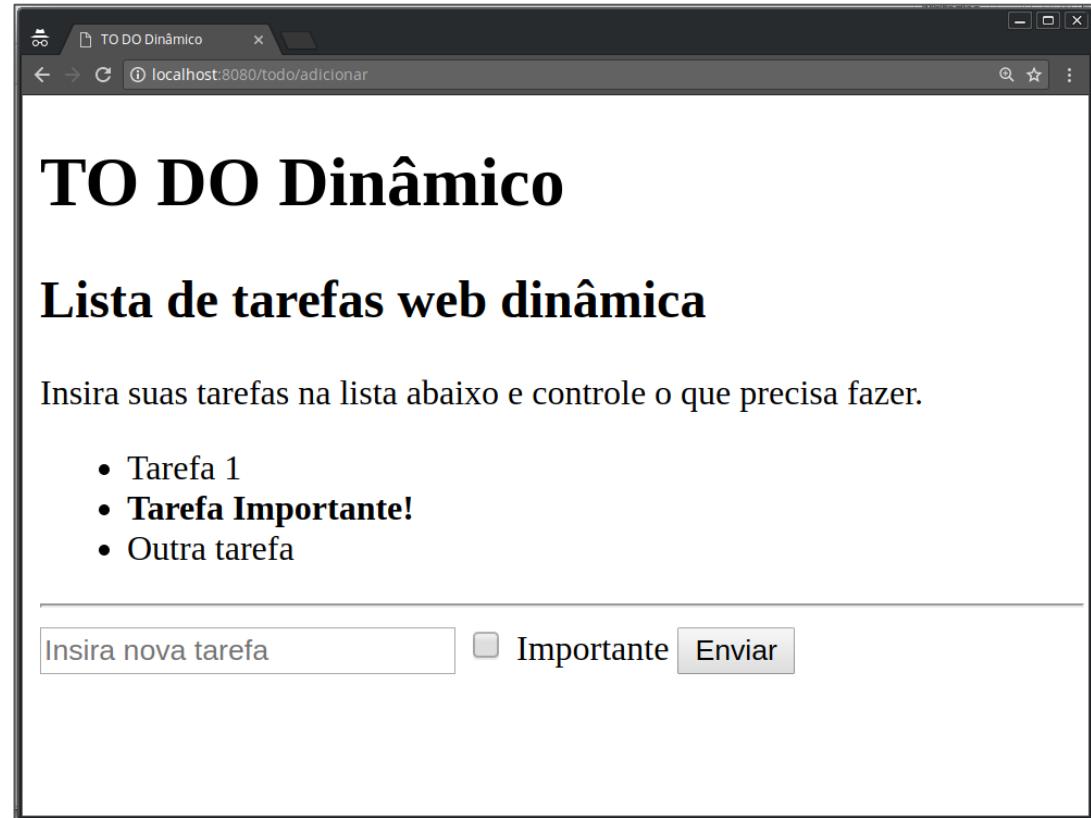
- ❑ Além de dividir a aplicação em três tipos de componentes, o desenho MVC define as interações entre eles.
 - ❑ **Um controlador (controller)** envia comandos para o modelo de acordo com as ações do usuário. O controlador também pode enviar comandos para a visão associada para alterar a apresentação da visão do modelo.
 - ❑ **Um modelo (model)** armazena dados, processa as regras de negócio, e altera o estado interno da aplicação, podendo notificar as visões sobre seu estado atual.
 - ❑ **A visão (view)** Gera uma representação (Visão) dos dados presentes no modelo solicitado.

| JSP | Servlets + JSP | MVC



| JSP | Servlets + JSP | MVC | Modelo

- ❑ Utilizando como base o Todo List desenvolvido nas atividades anteriores, vamos desenvolver uma versão MVC.



JSP | Servlets + JSP | MVC | Modelo

```
package todo.model;

public class Tarefa {

    private String descricao;
    private Boolean importante;

    public Tarefa(String descricao, Boolean importante) {
        this.descricao = descricao;
        this.importante = importante;
    }
    // Getters e Setters omitidos ...

    @Override
    public String toString() {
        return "Tarefa{" + "descricao=" + descricao + ", importante=" + importante + '}';
    }
}
```

Modelos podem ser
classes simples em Java (POJOs),
EJBs, entidades do JPA, etc.

JSP | Servlets + JSP | MVC | Visão

```
<%@page import="java.util.List"%>
<%@page import="todo.model.Tarefa"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head> ... </head>
    <body>
        <div class="centro">
            <h1>TO DO Dinâmico</h1>
            <h2>Lista de tarefas web dinâmica </h2>
            <p>Insira suas tarefas na lista abaixo e controle o que precisa fazer.</p>
            <ul>
                <% List<Tarefa> tarefas = (List<Tarefa>) session.getAttribute("tarefas"); %>
                <% if(tarefas != null) {%
                    <% for (Tarefa tarefa : tarefas) { %
                        <% if(tarefa.getImportante()) {%
                            <li><strong><%= tarefa.getDescricao() %></strong></li>
                        <% } else {%
                            <li><%= tarefa.getDescricao() %></li>
                        <% }%
                    <% }%
                <% }%
            </ul>
            <hr />
            <form action="/todo/adicionar" method="post">
                <input type="text" name="descricao" placeholder="Insira nova tarefa" />
                <input type="checkbox" name="importante"/> Importante <input type="submit" name="enviar" value="Enviar" />
            </form>
        </div>
    </body>
</html>
```

Apenas a lógica de exibição das tarefas da sessão.

(Ainda tem muito Java aqui... podemos melhorar isso.)

JSP | Servlets + JSP | MVC | Controlador

```
package todo.controllers;

@WebServlet(urlPatterns = "/adicionar")
public class AdicionarController extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        // 1 - Garantimos que a sessão existe.
        HttpSession session = req.getSession(true);

        // 2 - Obtemos os dados enviados e montamos o modelo Tarefa.
        String descricao = req.getParameter("descricao");
        boolean importante = "on".equalsIgnoreCase(req.getParameter("importante"));
        Tarefa tarefa = new Tarefa(descricao, importante);

        // 3 - Adicionamos a tarefa às tarefas da sessão.
        List<Tarefa> tarefas = (List<Tarefa>) session.getAttribute("tarefas");
        if (tarefas == null) {
            tarefas = new ArrayList<>();
        }
        tarefas.add(tarefa);

        // 4 - Devolvemos as tarefas para a sessão.
        session.setAttribute("tarefas", tarefas);

        // 5 - Repassamos a resposta para a View (index.jsp)
        RequestDispatcher rd = req.getRequestDispatcher("/index.jsp");
        rd.forward(req,resp);
    }
}
```

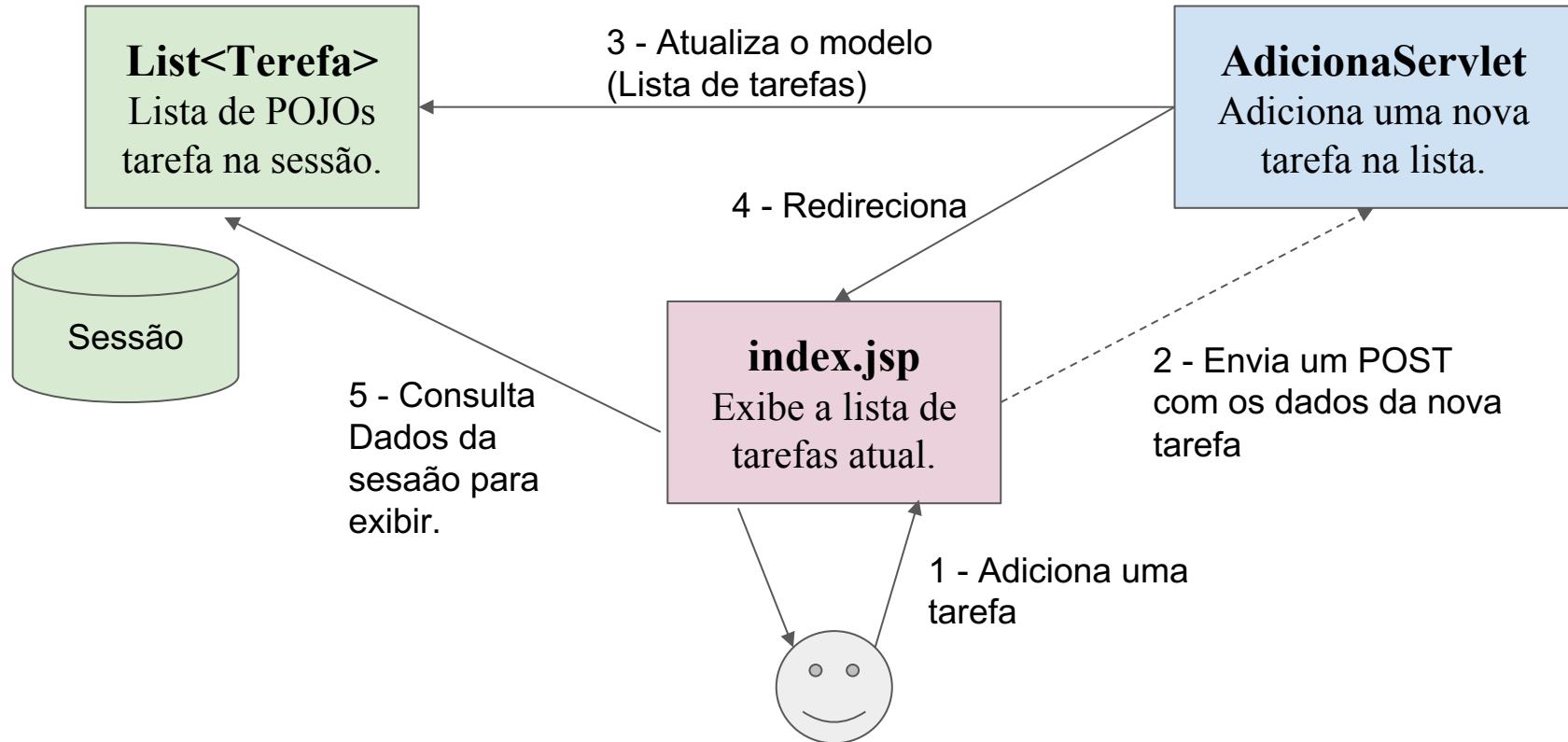
Recupera a sessão.

Atualiza o modelo
(Cria nova tarefa e adiciona nas tarefas existentes.)

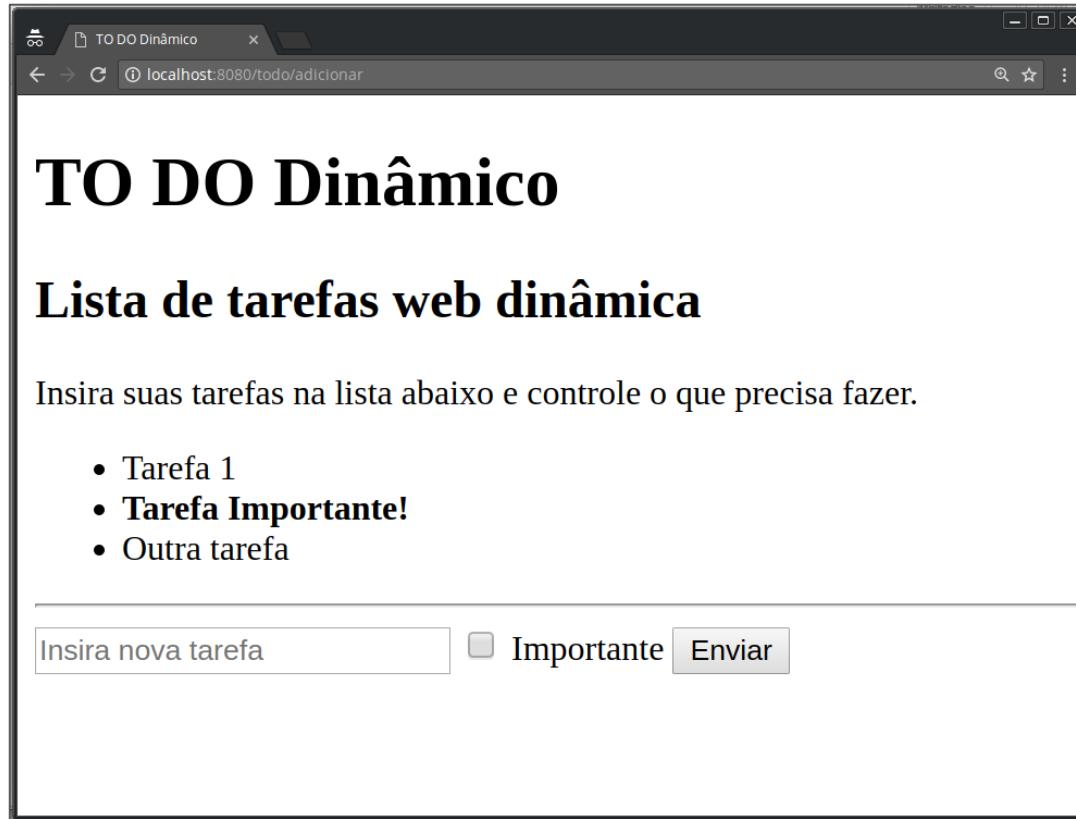
Atualiza a sessão.

Com o modelo atualizado, passa o controle para a View.

JSP | Servlets + JSP | MVC



| JSP | Servlets + JSP | MVC | Modelo



JSP | Scriptlets

□ Utilizando scriptlet o MVC ainda ficou “espaguete”.

```
<%@page import="java.util.List"%>
<%@page import="todo.model.Tarefa"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head> ... </head>
    <body>
        <div class="centro">
            <h1>TO DO Dinâmico</h1>
            <h2>Lista de tarefas web dinâmica </h2>
            <p>Insira suas tarefas na lista abaixo e controle o que precisa fazer.</p>
            <ul>
                <% List<Tarefa> tarefas = (List<Tarefa>) session.getAttribute("tarefas"); %>
                <% if(tarefas != null) {%
                    <% for (Tarefa tarefa : tarefas) { %
                        <% if(tarefa.getImportante()) {%
                            <li><strong><%= tarefa.getDescricao() %></strong></li>
                        <% } else {%
                            <li><%= tarefa.getDescricao() %></li>
                        <% }%
                    <% }%
                <% }%
            </ul>
            <hr />
            <form action="/todo/adicionar" method="post">
                <input type="text" name="descricao" placeholder="Insira nova tarefa" />
                <input type="checkbox" name="importante"/> Importante <input type="submit" name="enviar" value="Enviar" />
            </form>
        </div>
    </body>
</html>
```

Esse código ainda
não ficou bom!

| JSP | Scriptlets

- ❑ Porque separar código de negócio da visão?
 - ❑ Facilitar a manutenção da visão.
 - ❑ Permitir que uma equipe focada em frontend dê manutenção sem saber Java.
 - ❑ Permitir uma alteração visual com menor impacto.



| JSP | JSTL

- ❑ Para evitar Java no JSP a Sun sugeriu o uso da **JavaServer Pages Standard Tag Library**, a **JSTL**.
 - ❑ A JSTL encapsula tags simples funcionalidades como controle de laços (fors), controle de fluxo do tipo if else, formatação, etc.
 - ❑ A JSTL foi a forma encontrada de padronizar o trabalho de milhares de programadores de páginas JSP.
- ❑ O primeiro passo para utilizar JSTL é importá-lo na página:

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

| JSP | JSTL | Expression Language

- ❑ Juntamente com o JSTL podemos utilizar a **Expression Language (EL)**, que simplifica o acesso às informações.
 - ❑ Possuem a forma `${ valor }` .
 - ❑ Podem imprimir informações na tela sem necessidade de `<%= ... %>`:
 - ❑ `${ valor }`
 - ❑ Podem acessar getters e setters diretamente.
 - ❑ `${ objeto.propriedade }` e `${ objeto.propriedade.prop }`
 - ❑ `${ objeto.propriedade = valor }`
 - ❑ Podem acessar valores de listas...
 - ❑ `${ lista[3] }`
 - ❑ ... e mapas:
 - ❑ `${ mapa["chave"] }`

JSP | JSTL | Expression Language

```
<% List<Tarefa> tarefas = (List<Tarefa>)
session.getAttribute("tarefas"); %>
<% if(tarefas != null) {%
    <% for (Tarefa tarefa : tarefas) { %
        <% if(tarefa.getImportante()) {%
            <li>
                <strong>
                    <%= tarefa.getDescricao() %>
                </strong>
            </li>
        <% } else {%
            <li>
                <%= tarefa.getDescricao() %>
            </li>
        <% }%>
    <% }%>
<% }%>
```

```
<% List<Tarefa> tarefas = (List<Tarefa>)
session.getAttribute("tarefas"); %>
<% if(tarefas != null) {%
    <% for (Tarefa tarefa : tarefas) { %
        <% if(tarefa.getImportante()) {%
            <li>
                <strong>
                    ${tarefa.descricao}
                </strong>
            </li>
        <% } else {%
            <li>
                ${tarefa.descricao}
            </li>
        <% }%>
    <% }%>
<% }%>
```

| JSP | JSTL | Condicionais

- ❑ Utilizando JSTL e EL podemos adicionar tags para exibição condicional.

```
<c:if test="${tarefa.importante}">  
....  
</c:if>
```

```
<c:choose>  
  <c:when test="${tarefa.importante}">  
    ...  
  </c:when>  
  <c:when test="${tarefa.descricao == null}">  
    ...  
  </c:when>  
  <c:otherwise>  
    ...  
  </c:otherwise>  
</c:choose>
```

JSP | JSTL | Condicionais

```
<% List<Tarefa> tarefas = (List<Tarefa>)
session.getAttribute("tarefas"); %>
<% if(tarefas != null) {%
    <% for (Tarefa tarefa : tarefas) { %
        <% if(tarefa.getImportante()) {%
            <li>
                <strong>
                    <%= tarefa.getDescricao() %>
                </strong>
            </li>
        <% } else {%
            <li>
                <%= tarefa.getDescricao() %>
            </li>
        <% }%>
    <% }%>
<% }%>
```

```
<% List<Tarefa> tarefas = (List<Tarefa>)
session.getAttribute("tarefas"); %>
<% if(tarefas != null) {%
    <% for (Tarefa tarefa : tarefas) { %
        <li>
            <c:choose>
                <c:when test="${tarefa.importante}">
                    <strong>${tarefa.descricao}</strong>
                </c:when>
                <c:otherwise>
                    <li>${tarefa.descricao}
                </c:otherwise>
            </c:choose>
        </li>
    <% }%>
<% }%>
```

| JSP | JSTL | Laços

- ❑ Utilizando JSTL e EL podemos adicionar tags para exibição de laços.

```
<table>
    <!-- percorre contatos montando as linhas da tabela -->
    <c:forEach var="contato" items="${contatos}">
        <tr>
            <td>${contato.nome}</td>
            <td>${contato.email}</td>
            <td>${contato.endereco}</td>
        </tr>
    </c:forEach>
</table>
```

JSP | JSTL | Condicionais

```
<% List<Tarefa> tarefas = (List<Tarefa>)
session.getAttribute("tarefas"); %>
<% if(tarefas != null) {%
    <% for (Tarefa tarefa : tarefas) { %
        <% if(tarefa.getImportante()) {%
            <li>
                <strong>
                    <%= tarefa.getDescricao() %>
                </strong>
            </li>
        <% } else {%
            <li>
                <%= tarefa.getDescricao() %>
            </li>
        <% }%>
    <% }%>
<% }%>
```

```
<c:forEach var="tarefa"
           items="${sessionScope.tarefas}">
    <li>
        <c:choose>
            <c:when test="${tarefa.importante}">
                <strong>${tarefa.descricao}</strong>
            </c:when>
            <c:otherwise>
                ${tarefa.descricao}</li>
            </c:otherwise>
        </c:choose>
    </li>
</c:forEach>
```

JSP | JSTL | Escopo

```
<c:forEach var="tarefa"
           items="${sessionScope.tarefas}">
    <li>
        <c:choose>
            <c:when test="${tarefa.importante}">
                <strong>${tarefa.descricao}</strong>
            </c:when>
            <c:otherwise>
                ${tarefa.descricao}</li>
            </c:otherwise>
        </c:choose>
    </li>
</c:forEach>
```

- ❑ Existem diferentes escopos acessíveis na EL.
 - ❑ **pageScope** escopo do JSP atual.
 - ❑ **requestScope** escopo da requisição atual.
 - ❑ **sessionScope** escopo da sessão atual.
 - ❑ **applicationScope** escopo da aplicação atual.

JSP | JSTL | Escopo

```
<c:forEach var="tarefa"
           items="${tarefas}">
    <li>
        <c:choose>
            <c:when test="${tarefa.importante}">
                <strong>${tarefa.descricao}</strong>
            </c:when>
            <c:otherwise>
                ${tarefa.descricao}</li>
            </c:otherwise>
        </c:choose>
    </li>
</c:forEach>
```

- ❑ Se não informarmos o escopo, o atributo será buscado na ordem de precedência dos escopos.
- ❑ Pode haver conflito de nomes e bugs em função disso.

| JSP | Servlets + JSP | Atividade

- ❑ Implemente uma página para avaliação de fotos a partir do modelo/template disponibilizado.
 - ❑ Por enquanto, as informações serão armazenadas na sessão.
- ❑ Para isso crie 2 páginas e quantos servlets forem necessários:
 - ❑ **Página 1:** Página com a postagem de fotos e listagem de avaliações.
 - ❑ **Página 2:** Cadastro de novas avaliações.
- ❑ Crie os servlets que achar necessários.
- ❑ Lembre-se de seguir o padrão MVC e usar JSTL na visão.
- ❑ Envie para tassio@tassio.eti.br com o assunto Atividade 6 - PhotArt JSP até XX/XX

The image consists of two screenshots of a web application called PhotArt. The top screenshot shows a circular planter on a spiral staircase with several green plants. Below the image is a rating bar with four yellow stars and one empty star, followed by the text 'Avaliação 4.3 / 5'. The bottom screenshot shows the same image with an 'Add Review' form overlaid. The form includes fields for 'Title' (filled with 'Muito bom!'), 'User' (filled with '@johndoe'), 'Review' (empty), and 'Rating' (a row of five radio buttons). A large blue button at the bottom right says 'Publicar Avaliação'.

JSP | Autenticação

- ❑ É um requisito comum em aplicações web autenticar os usuários para que eles possam fazer alterações nos dados da aplicação.
 - ❑ Cliente em um sistema de compras.
 - ❑ Responsável por uma ação em um sistema de controle.
 - ❑ Nível de acesso do usuário para as informações.
- ❑ Existem vários mecanismos de autenticação:
 - ❑ Usuário e Senha.
 - ❑ Autenticação em 2 passos.
 - ❑ Tokens
 - ❑ Biometria



JSP | Autenticação vs Autorização

- ❑ É como que haja confusão entre os conceitos **autenticação e autorização**.
 - ❑ Autenticação:
 - ❑ Processo de verificação de uma identidade alegada, por meio de comparação das credenciais apresentadas pelo usuário com outras pré-definidas.
 - ❑ Autorização:
 - ❑ Processo que ocorre após a autenticação e que tem a função de diferenciar os privilégios atribuídos ao sujeito autenticado. c



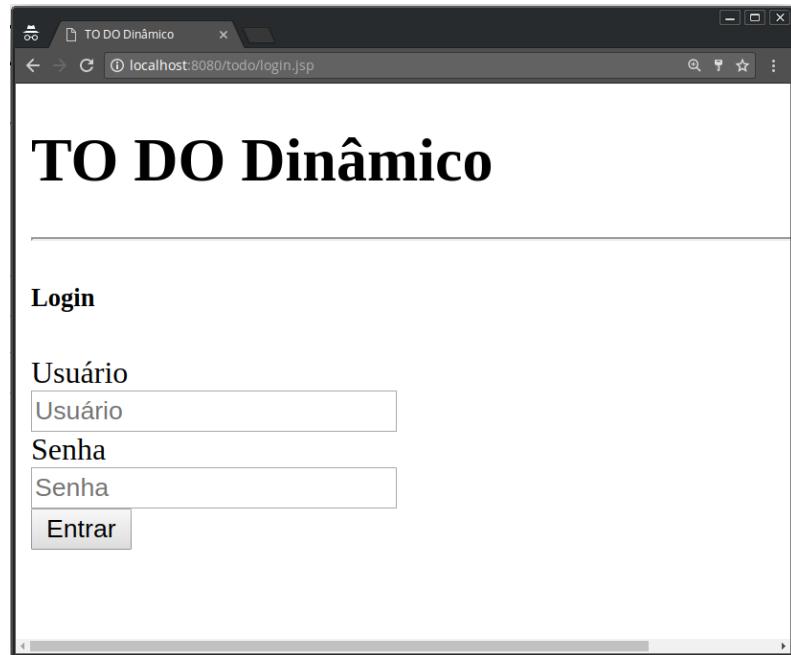
| JSP | Utilizando Servlets e JSPs para Autenticação

- ❑ Vamos estudar como implementar mecanismos simples de autenticação utilizando Servlets e JSPs:
 - a. Adicionando uma página para verificação se o usuário está autenticado.
 - b. Criar um servlet que vai gravar na sessão as informações do usuário autenticado através desta página.
 - c. Redirecionar para a página de login quando não houver informação de autenticação na sessão em algum outro controller ou página.



JSP | Autenticação

❑ Implementação de autenticação:



TO DO Dinâmico

Login

Usuário

Senha

Entrar



TO DO Dinâmico

Lista de tarefas web dinâmica

Insira suas tarefas na lista abaixo e controle o que precisa fazer.

- Tarefa 1
- **Tarefa Importante!**
- Outra tarefa

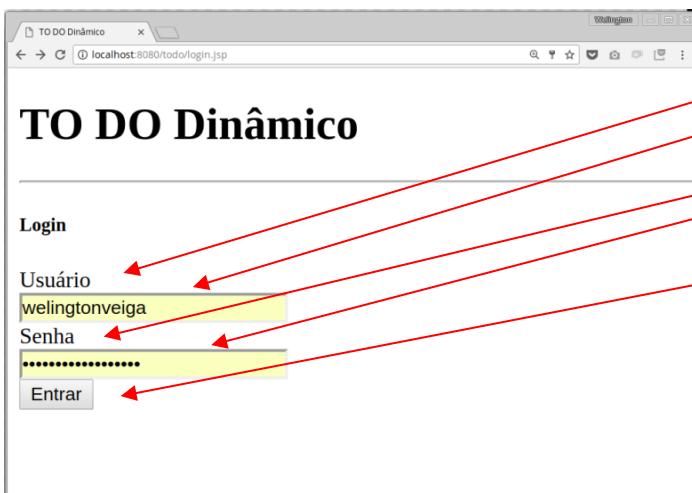
Insira nova tarefa

Importante

Enviar

JSP | Autenticação | 1 - Página de Login

1. Criamos um formulário de login, que deve ser informado no primeiro acesso do usuário.



```
login.jsp

<html>
  <head> ... </head>
  <body>
    <div class="centro">
      <h1>TO DO Dinâmico</h1>
      <hr />
      <h5>Login</h5>
      <form action="/todo/login" method="post">
        <label for="username">Usuário</label> <br />
        <input id="username" type="text"
               name="username" placeholder="Usuário" /><br />
        <label for="password">Senha</label> <br />
        <input id="password" type="password"
               name="password" placeholder="Senha"/><br />
        <input type="submit" name="entrar" value="Entrar" />
      </form>
    </div>
  </body>
</html>
```

JSP | Autenticação | 2 - Definimos o modelo de usuário

Usuario.java

```
public class Usuario {  
  
    private String name;  
    private String password;  
  
    public Usuario(String name, String password) {  
        this.name = name;  
        this.password = password;  
    }  
  
    // Getters e setters  
  
}
```

```
@Override  
public int hashCode() {  
    return // ...;  
}  
  
}
```

```
@Override  
public boolean equals(Object obj) {  
    return // ...;  
}  
  
@Override  
public String toString() {  
    return "Usuario{" + "name=" + name + ", password=" + password + '}';  
}  
}
```

O modelo de usuário precisa ter informações para identificação e autenticação do mesmo.
Muitas vezes precisa estar relacionado à informações de autorização.
Um usuário pode ou não estar ligado a uma outra entidade, como pessoa, cliente, aluno, etc.

Métodos definidos no contrato de Object precisam estar definidos:
equals, hashCode e toString devem ser implementados.

JSP | Autenticação | 3 - Servlet de Login

LoginController.java

```
@WebServlet(urlPatterns = "/login")
public class LoginController extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        // 1 - Garantimos que a sessão existe.
        HttpSession session = req.getSession(true);

        // 2 - Verificamos se o usuário existe.
        String username = req.getParameter("username");
        String password = req.getParameter("password");

        // 3 - Verificamos se reconhecemos o usuário.
        // Poderia ser um banco de dados, serviço externo, etc...
        if (username.equals("admin") && password.equals("admin")) {
            // 4 - Salvamos o login
            Usuario usuario = new Usuario(username, password);
            session.setAttribute("usuario", usuario);
            RequestDispatcher rd = req.getRequestDispatcher("/index.jsp");
            rd.forward(req,resp);
        } else {
            req.setAttribute("erro", "Usuário ou senha desconhecido.");
            RequestDispatcher rd = req.getRequestDispatcher("/login.jsp");
            rd.forward(req,resp);
        }
    }
}
```

Garantimos que
há sessão.

Obtemos da requisição
os dados usuário e senha.

Se os dados recebidos
são válidos, redirecionamos
para a página inicial.

Caso contrário retornamos
para a página de login
com uma mensagem de erro.

JSP | Autenticação | 4 - Restringir o acesso não autenticado

- ❑ Mas como restringir o acesso à páginas?
 - ❑ Verificando o parâmetro “usuario” na sessão.
 - ❑ Podemos adicionar essa verificação em todos os servlets menos no servlet que processa a autenticação.

```
@WebServlet(urlPatterns = "/adicionar")
public class AdicionarController extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        // 1 - Garantimos que a sessão existe.
        HttpSession session = req.getSession(true);
        if (session.getAttribute("usuario") == null) {
            req.setAttribute("erro", "Favor realize login para continuar.");
            RequestDispatcher rd = req.getRequestDispatcher("/login.jsp");
            rd.forward(req, resp);
        }

        // ... Regras do Servlet ...

        RequestDispatcher rd = req.getRequestDispatcher("/index.jsp");
        rd.forward(req, resp);
    }
}
```

JSP | Autenticação | 4 - Restringir o acesso não autenticado

- ❑ E para restringir o acesso à página JSP?

- ❑ Podemos criar um servlet que decide se redireciona para a página ou não.

IndexController.java

```
@WebServlet(name = "IndexController", urlPatterns = {" /index"})
public class IndexController extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        HttpSession session = request.getSession();
        if (session != null && session.getAttribute("usuario") != null) {
            RequestDispatcher rd = request.getRequestDispatcher("/index.jsp");
            rd.forward(request, response);
        } else {
            request.setAttribute("erro", "Usuário ou senha desconhecido.");
            RequestDispatcher rd = request.getRequestDispatcher("/login.jsp");
            rd.forward(request, response);
        }
    }
}
```

JSP | Autenticação | 4 - Restringir o acesso não autenticado

- Para que o servlet seja carregado antes da página, podemos utilizar o web.xml.

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ... >

<welcome-file-list>
    <welcome-file>index</welcome-file>
</welcome-file-list>
```

Quando a URL base da aplicação é informada, o glassfish procura pelas páginas iniciais padrão:
(index.jsp, index.html ...)

Usando a tag **welcome-list-files** é possível definir seu próprio JSP/Servlet inicial.
URL Pattern do servlet.
Atenção, sem "/"

```
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
</web-app>
```

JSP | Autenticação | 5 - Feedback

- ❑ Vamos imprimir a mensagem de erro na página de login!

Web.xml

```
<div class="centro">
    <h1>TO DO Dinâmico</h1>
    <hr />
    <h5>Login</h5>
    <c:if test="${not empty requestScope.erro}">
        <p style="color: red">${requestScope.erro}</p>
    </c:if>
    <p>Entre com suas credenciais</p>
    <form action="/todo/login" method="post">
        <label for="username">Usuário</label> <br />
        <input id="username" type="text" name="username" placeholder="Usuário" /><br />
        <label for="username">Senha</label> <br />
        <input id="password" type="password" name="password" placeholder="Senha"/><br />
        <input type="submit" name="entrar" value="Entrar" />
    </form>
</div>
```

Imprimimos um feedback de erro, caso haja.

JSP | Autenticação

❑ Exemplo de autenticação:



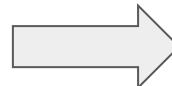
TO DO Dinâmico

Login

Usuário

Senha

Entrar



TO DO Dinâmico

Lista de tarefas web dinâmica

Insira suas tarefas na lista abaixo e controle o que precisa fazer.

- Tarefa 1
- **Tarefa Importante!**
- Outra tarefa

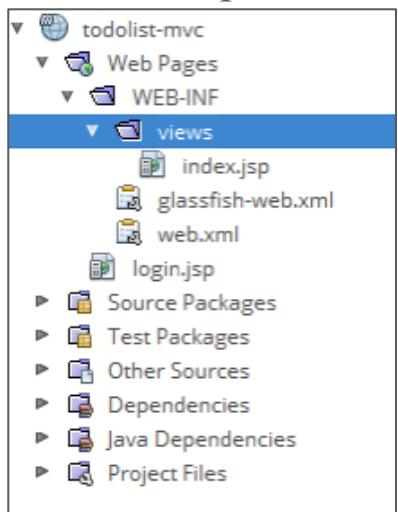
Insira nova tarefa

Importante

Enviar

JSP | Autenticação | Atenção!

- ❑ Nesse exemplo os JSPs ainda podem ser acessados diretamente.
 - ❑ Uma dica para evitar isso é adicioná-los onde não podem ser acessados, como na pasta WEB-INF:



```
HttpSession session = req.getSession(true);
if (session.getAttribute("usuario") != null) {
    RequestDispatcher rd;
    rd = req.getRequestDispatcher("/WEB-INF/views/index.jsp");
    rd.forward(req, resp);
}
```

| JSP | Autenticação | Atividade

- ❑ A partir da atividade anterior, utilizando [esse template de página de login](#):
 - a. Adicionar a página como **login.jsp** no projeto.
 - b. Crie um modelo de usuário.
 - c. Proteja os controladores existentes para aceitarem apenas autenticados.
 - d. Deixe a página de login como acesso padrão.
- ❑ Lembre-se de seguir o padrão MVC e usar JSTL na visão.
- ❑ Envie para tassio@tassio.eti.br com o assunto Atividade 6 - PhotArt JSP até xx/xx.

The diagram illustrates a process flow. On the left, there is a screenshot of a web application interface titled "PhotArt". It shows a circular garden scene with a central plant and several smaller potted plants on a balcony. Below the image are buttons for "Adicionar Avaliação" (Add Review) and "Avaliação 4.3 / 5" with a star rating. To the right of the main image is a large downward-pointing arrow. On the far right, there is another screenshot of the same "PhotArt" interface, but it is a review page. It shows the same circular garden image and includes a form for writing a review. This second screenshot is also accompanied by a large downward-pointing arrow. At the bottom left of the second screenshot, there is a large upward-pointing arrow. The entire sequence of screenshots and arrows is enclosed in a rectangular frame.

PhotArt

Simetria por @yuanzhanglang

Avaliação 4.3 / 5

Avaliações

Top aligned media por @ususio

Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.

Donec sed odio dui. N

parturient montes, n

© 2017 Laboratório de Web Sites Dinâmicos - CES JF.

PhotArt

Simetria por @yuanzhanglang

Escreva sua avaliação

Título: Muito bom!

Usuário: @johndoe

Avaliação:

Nota final: 1 2 3 4 5

Publicar Avaliação

© 2017 Laboratório de Web Sites Dinâmicos - CES JF.

HTTP | Referências adicionais

1. <https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html>
2. <https://www.caelum.com.br/apostila-java-web/servlets/>
3. <http://blog.caelum.com.br/java-ee6-comecando-com-as-servlets-3-0/>
4. <https://www.caelum.com.br/apostila-java-web/appendice-topicos-da-servlet-api/>
5. [https://www.tutorialspoint.com/servlets/servlets-form-data.htm](https://www.tutorialspoint.com/servlets/servlets_form_data.htm)
6. https://www.tutorialspoint.com/jsp/jsp_syntax.htm
7. https://www.owasp.org/index.php/Authentication_Cheat_Sheet
8. <https://martinfowler.com/articles/web-security-basics.html>