

Java Web

AULA 06 – JSP

Objetivos e Conceitos

- Objetivos:
 - Apresentar JSP, tradução para Servlet, arquitetura completa de uma aplicação, usando MVC e padrões de projeto
- Conceitos:
 - JSP. MVC. Padrões de Projeto.


Tópicos

- JSP
- Misturando HTML e Java
- Redirecionamentos
- MVC
- Arquitetura da Aplicação

JSP

JSP Básico – Exemplo 1

```
<html>
  <head><title>Titulo</title>
</head>
  <body>
    <b>Página HTML</b><br/>
    <h1><% out.println("Hello World!"); %></h1>
  </body>
</html>
```



A diagram consisting of an orange bracket that spans the width of the JSP tag `<% out.println("Hello World!"); %>` in the code above. Below the bracket is a small orange rectangular box containing the text "Código Java".

JSP Básico – Exemplo 2

```
<html>
  <head><title>Titulo</title>
</head>
  <body>
    <b>Página HTML</b><br/>
    <h1><% out.println("Hello World!"); %></h1>
    <h2><%
      for (int i=0; i<10; i++) {
        out.println("Valor = " + i);
      }
    %></h2>
  </body>
</html>
```

JSP

Arquivo com formato HTML

Contém código JAVA dentro

Extensão .jsp

Do JSP é gerado uma SERVLET

- Ocorre na primeira invocação somente

O Servidor executa a Servlet gerada

JSP

Arquivo com formato HTML

Contém código JAVA dentro

Extensão .jsp

Do JSP é gerado uma SERVLET

- Ocorre na primeira invocação somente

O Servidor executa a Servlet gerada

JSP -> Servlet

```
<html><head>
  <title>Titulo</title>
</head><body>
  <%
    for (int i=0; i<10; i++) {
      out.println(i + " - ");
    }
  %>
</body></html>
```

JSP -> Servlet

```
public class meuJSP_jsp extends HttpServlet {
  public void _jspService(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException {
    PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    out.write("<html><head>");
    out.write("<title>Titulo</title>");
    out.write("</head><body>");
    for (int i=0; i<10; i++) {
      out.println(i + " - ");
    }
    out.write ("</body></html>");
  }
}
```

JSP -> Servlet

```

public class meuJSP_jsp extends HttpServlet {
    public void _jspService(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.write("<html><head>");
        out.write("<title>Teste</title>");
        out.write("</head><body>");
        for (int i=0; i<10; i++) {
            out.println(i + " - ");
        }
        out.write ("</body></html>");
    }
}

```

Diagram illustrating the mapping between JSP code and Servlet code:

- `<html><head>` maps to `out.write("<html><head>");`
- `<title>Titulo</title>` maps to `out.write("<title>Teste</title>");`
- `</head><body>` maps to `out.write("</head><body>");`
- `<% for (int i=0; i<10; i++) { out.println(i + " - "); } %>` maps to the `for` loop in the Servlet code.
- `</body></html>` maps to `out.write ("</body></html>");`

Objetos Implícitos

request : representa os dados passados na requisição da página
(`HttpServletRequest`)

```
str = (String) request.getParameter("nome");
```

response : representa os dados de saída para o cliente
(`HttpServletResponse`)

```
response.setContentType("text/html; charset=UTF-8");
```

out : representa a saída para o cliente (`javax.servlet.jsp.jspWriter`)

```
out.println("<h1>Oi mundo</h1>");
```

session : representa uma área de dados para armazenar dados durante a aplicação (`HttpSession`)

```
session.setAttribute("nome", "Razer");
```

Objetos Implícitos

config : Objeto de configuração do Servlet/JSP
(`javax.servlet.ServletConfig`)

application : Representa o contexto da aplicação, para acessar dados compartilhados entre todos os Servlets/JSP
(`javax.servlet.ServletContext`)

page : Sinônimo para **this**. (`java.lang.Object`)

pageContext : Acesso a todos os escopos, atributos de página (como requisição e resposta), objetos implícitos, etc.
(`javax.servlet.jsp.PageContext`)

exception : Representa uma exceção, para tratamento de erros, somente definido em páginas de erro (`java.lang.Throwable`)

Blocos de Código JSP

Scriptlets

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

Declarações

```
<%! ... %>
```

Expressões

```
<%= ... %>
```

Comentários

```
<!-- ... -->    <% //...%>    <% /*...*/ %>
<%-- ... --%>
```

Diretivas

```
<%@ ... %>
```

Scriptlets

```
<html>
  <head><title>Titulo</title>
</head>
  <body>
    <b>Página HTML</b><br/>
    <h1><% out.println("Hello World!"); %></h1>
    <h2><%
      for (int i=0; i<10; i++) {
        out.println("Valor = " + i);
      }
    %></h2>
  </body>
</html>
```

Declarações

Geram Atributos e Métodos na Servlet para a qual o JSP é gerado

Atributos

```
<%! String nome = "Razer"; %>
```

Métodos

```
<%!
  String getSaudacao() {
    return "opa";
  }
%>
```


Exemplo

```
<%! String frase = "JavaEE Rulez"; %>
<html><head><title>Teste</title></head>
<body>
    <b>Página HTML</b><br/>
    <h1><%
        out.println(frase);
    %></h1>
</body>
</html>
```

Exemplo

```
public class meuJSP_jsp extends HttpServlet {
    String frase = "JavaEE Rulez";
    public void _jspService(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.write("<html><head><title>Teste</title>/head><body>");
        out.write("<b>Página Teste</b><br/>");
        out.write("<h1>");
        out.println(frase);
        out.write("</h1></body></html>");
    }
}
```

Expressões

```
<html><head><title>Teste</title></head>
<body>
  <b>Página HTML</b><br>
  <h1><%= new java.util.Date() %></h1>
</body>
</html>
```

Diretivas

Diretiva **import**: Importação de pacotes

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

Pode ter vários imports na mesma declaração, separados por vírgula

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

Ou pode ter vários imports em várias declarações

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

Diretivas

Diretiva **include**

- Inclusão de arquivos na hora da compilação para *Servlet*
- Os dois arquivos são combinados para virar um só
- Pode incluir arquivos JSP
- O arquivo é colocado no lugar da diretiva

```
<%@ include file="arquivo.jsp" %>
```

Não usar tags `<html>`, `</html>`, `<body>`, `</body>` para não conflitar com as do JSP

Diretivas

Diretiva **page contentType**

- Indica qual o tipo de saída (MIME) que o JSP gerará

```
<%@ page contentType="text/html" %>
```

Diretivas

Diretiva **page session**

- Indica se a página utiliza ou não sessões
- Por default é true

```
<%@ page session="true" %>
```

Diretivas

Diretiva **page buffer**

- Indica se e quanto de buffer a página utiliza
- Por default é 8Kb

```
<%@ page buffer="32Kb" %>
```

- Quando o conteúdo da página atingir 32Kb ou estiver completa, é enviado para o cliente

Diretivas

Diretiva **page errorPage**

- Indica a página de erro para onde será direcionado se uma exceção ocorrer

```
<%@ page errorPage="erro.jsp" %>
```

Diretivas

Diretiva **page isErrorPage**

Indica se esta página é de erro ou não

```
<%@ page isErrorPage="true" %>
```

Usa-se a variável `exception` para mostrar a exceção

```
<%=exception %>
```

Diretivas

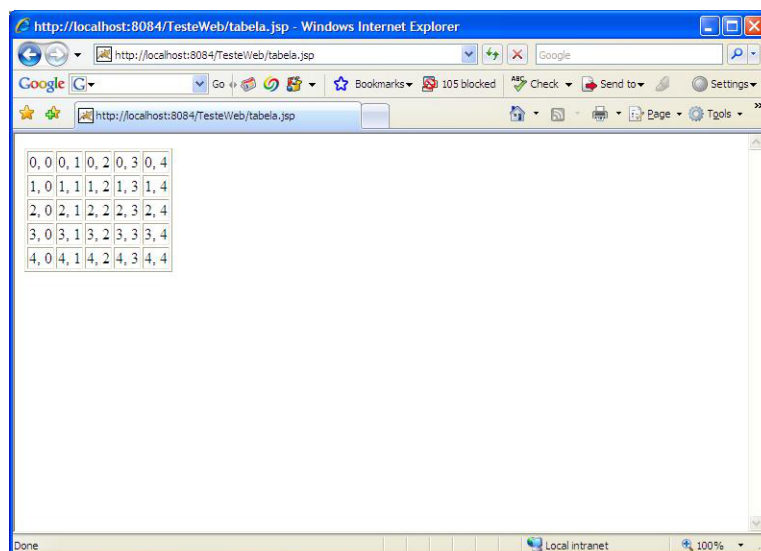
```
<%@ page import="java.util.*" %>
<html>
<head><title>Teste</title></head>
<body>
  <b>Página HTML</b><br/>
  <%
    ArrayList<Integer> arr = new ArrayList<>();
    arr.add(10);
    arr.add(20);
    arr.add(30);
    for (Integer i: arr)
      out.println("<h2>" + i + "</h2>");
  %>
</body>
</html>
```

Misturando HTML e Java

JSP

```
<html><head><title>Tabelas</title></head>
<body>
  <table border="1">
    <% String str = "";
      for (int i=0; i<5; i++) {
        str += "<tr>";
        for (int j=0; j<5; j++) {
          str += "<td style=\"text-align:center\">";
          str += i + ", " + j;
          str += "</td>";
        }
        str += "</tr>";
      }
      out.println(str);  %>
    </table>
  </body>
</html>
```

JSP



JSP

```
<html><html><head><title>Tabelas</title></head>
<body>
  <table border="1">
    <% for (int i=0; i<5; i++) { %>
      <tr>
        <% for (int j=0; j<5; j++) { %>
          <td style="text-align:center">
            <%= i + ", " + j %>
          </td>
        <% } %>
      </tr>
    <% } %>
  </table>
</body>
</html>
```

Redirecionamentos

Redirecionamentos: forward

Uso em JSP através de tags

Redireciona, via **forward**, para o recurso indicado

```
<jsp:forward page="<pagina>" />
<jsp:forward page="<pagina>">
    <jsp:param name="<parametro>" value="<valor>" />
</jsp:forward>
```

Onde

- **page** : Página HTML, JSP ou Servlet
- **name** : Nome do parâmetro passado no Request
- **value** : Valor do parâmetro passado no Request

Os parâmetros são obtidos via `request.getParameter()`

Redirecionamentos: forward

Cuidado

Por default, a página é *bufferizada*

Se for usado a diretiva:

```
<%@ page buffer="none" %>
```

Indicará que não está usando *buffer* de saída

Portanto:

- Ao usar o **jsp:forward**
- Se já foi enviado algum dado para a saída
- Será levantada uma exceção: **IllegalStateException**

Redirecionamentos: include

Uso em JSP através de tags

Inclui a saída do recurso indicado

```
<jsp:include page="<pagina>" />
<jsp:include page="<pagina>">
    <jsp:param name="<parametro>" value="<valor>" />
</jsp:include>
```

Onde

- **page** : Página HTML, JSP ou Servlet
- **name** : Nome do parâmetro passado no Request
- **value** : Valor do parâmetro passado no Request

Os parâmetros são obtidos via `request.getParameter()`

Include-File x Jsp-Include

Diretiva: **include file**

- Inclui o arquivo no local (HTML ou JSP)
- Ocorre antes da execução do JSP

```
<%@include file="teste.jsp" %>
```

Ação: **JSP include**

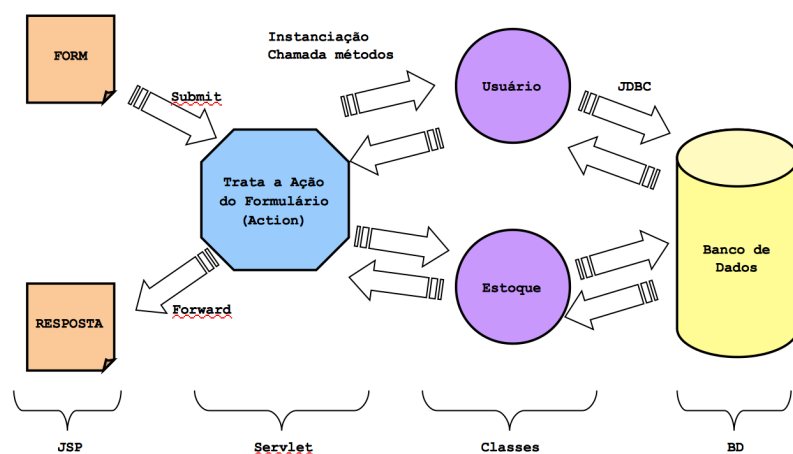
- Executa o arquivo e coloca a saída no local
- Ocorre na hora da execução do JSP

```
<jsp:include page="teste.jsp" %>
```

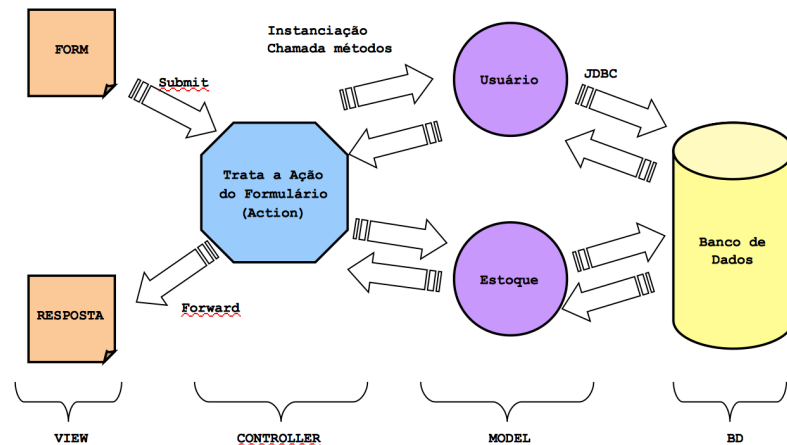
MVC

JSP + SERVLETS

Arquitetura de Aplicações



Arquitetura de Aplicações



MVC: Implementação

JSP (Visão)

- Contém formulário ou dados a serem mostrados
- Sempre que uma ação acontece, passa por uma Servlet controladora

Servlet (Controladora)

- Recebe as requisições do JSP
- Trata: acessando o banco, descobrindo a próxima tela para onde o sistema deve ir, etc
- Redireciona para outra JSP para mostrar os resultados

Controladora

Todas as controladoras executam 4 passos bem definidos

1. **PASSO 1:** Obter parâmetros
2. **PASSO 2:** Processar/Validar/Converter Parâmetros
3. **PASSO 3:** Efetuar Ação
4. **PASSO 4:** Redirecionamento (Passando ou não parâmetros)

Controladora

```
...
@WebServlet(urlPatterns = {"/Controladora"})
public class Controladora extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {

        // PASSO 1: Obter Parâmetros
        String strId = request.getParameter("id");

        // PASSO 2: Processar/Validar/Converter Parâmetros
        int id = Integer.parseInt(strId);

        // PASSO 3: Efetuar Ação
        ClientesDAO dao = new ClientesDAO();
        dao.remover (id);

        // PASSO 4: Redirecionamento
        RequestDispatcher rd = getServletContext().
            getRequestDispatcher("/mostrar.jsp");
        rd.forward(request, response);
    }
}
```

Exemplo

```
<html><head><title>index.jsp</title></head>

<body>

<form action="Controladora" method="POST" >

    Nome:  <input type="text" name="nome" /><br/>

    E-mail: <input type="text" name="email" /><br/>

    <input type="submit" value="Salvar" />

    <input type="reset" value="Limpar" />

</form>

</body>

</html>
```

Exemplo

```
...

public class Controladora extends HttpServlet {

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

        String nm = request.getParameter("nome");
        String em = request.getParameter("email");

        // faz o que tem que fazer

        RequestDispatcher rd = getServletContext().
            getRequestDispatcher("/mostrar.jsp");

        rd.forward(request, response);

    }

}
```

Exemplo

```
<html><head><title>mostrar.jsp</title></head>  
<body>  
    <h1>Ação efetuada com sucesso</h1>  
</body>  
</html>
```

Exercícios



- Executar a aplicação anterior

Passando Informações da Servlet para a JSP

Implementação

Na Servlet (Controladora), usa-se o request para passar um atributo

```
request.setAttribute("mensagem", "oi");
```

Na JSP destino, usa-se

```
(String) request.getAttribute("mensagem")
```

Para obter o atributo

OU, usando EL (Expression Language)

```
${mensagem}
```


Exemplo

```
<html><head><title>index.jsp</title></head>
<body>
<form action="Controladora" method="POST" >
    Nome:   <input type="text" name="nome" /><br/>
    E-mail: <input type="text" name="email" /><br/>
    <input type="submit" value="Salvar" />
    <input type="reset" value="Limpar" />
</form>
</body>
</html>
```

Exemplo

```
...
@WebServlet(urlPatterns = {"/Controladora"})
public class Controladora extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                           HttpServletResponse response)
        throws ServletException, IOException {
        String nm = request.getParameter("nome");
        String em = request.getParameter("email");
        // faz o que tem que fazer
        request.setAttribute("mensagem", "Dados corretos");
        RequestDispatcher rd = getServletContext().
            getRequestDispatcher("/mostrar.jsp");
        rd.forward(request, response);
    }
}
```

Exemplo

```
<html><head><title>mostrar.jsp</title></head>
<body>
  <h1>Ação efetuada com sucesso</h1>
  Mensagem: <%= (String)request.getAttribute("mensagem") %>
</body>
</html>
```

Exemplo

```
<html><head><title>mostrar.jsp</title></head>
<body>
  <h1>Ação efetuada com sucesso</h1>
  Mensagem: ${mensagem}
</body>
</html>
```

Exercícios



Criar a aplicação anterior, usando-se as duas alternativas para passagem de parâmetros e executá-la

- Na servlet não é necessário efetuar nenhuma ação, basta o redirecionamento com parâmetro

Passando Lista de Informações da Servlet para a JSP

Implementação

A diferença desta para o anterior é o tipo do dado que está sendo passado

Será passado uma Lista de Strings

```
List<String> arr = new ArrayList<String>();  
arr.add("Razer");  
request.setAttribute("dados", arr);
```

Na JSP destino, usa-se

```
(List<String>) request.getAttribute("dados");
```

Para obter o atributo

OU, usando EL (Expression Language)

```
${dados}
```

Exemplo

```
<html><head><title>index.jsp</title></head>  
<body>  
  <form action="Controladora" method="POST" >  
    Text:  <input type="text" name="texto" /><br/>  
    <input type="submit" value="Pesquisar" />  
    <input type="reset" value="Limpar" />  
  </form>  
</body>  
</html>
```

Exemplo

```
...
@WebServlet(urlPatterns = {"/Processa"})
public class Controladora extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                           HttpServletResponse response)
        throws ServletException, IOException {
        String texto = request.getParameter("texto");
        // faz o que tem que fazer
        List<String> lista = new ArrayList<String>();
        lista.add("Razer");
        request.setAttribute("dados", lista);
        RequestDispatcher rd = getServletContext().
            getRequestDispatcher("/mostrar.jsp");
        rd.forward(request, response);
    }
}
```

Exemplo

```
<html><head><title>mostrar.jsp</title></head>
<body>
    <h1>Ação efetuada com sucesso</h1>
    Dados: <br/>
    <%
        List<String> arr = (List<String>)request.getAttribute("dados");
        for (String x: arr) {
            out.println(x + "<br/>");
        }
    %>
</body>
</html>
```

Exemplo

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html><head><title>mostrar.jsp</title></head>
<body>
  <h1>Ação efetuada com sucesso</h1>
  Dados: <br/>
  <c:forEach items="${dados}" var="x">
    ${x}<br/>
  </c:forEach>
</body>
</html>
```

Para isso funcionar tem
Que adicionar a biblioteca
JSTL no projeto

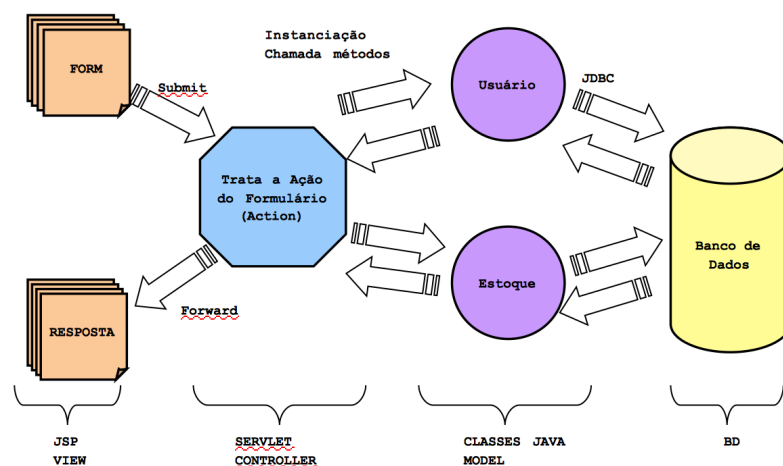
Exercícios



Criar a aplicação anterior das duas formas

Várias Ações, Uma Servlet

MVC – Implementação



Várias Ações

Passa-se um parâmetro (ex. **action**) para a Servlet, indicando qual é a ação a ser executada

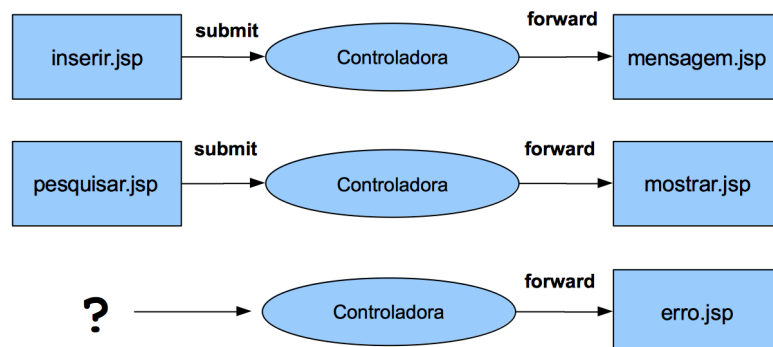
Na servlet, obtém-se este parâmetro e faz-se um IF para cada ação pertinente

Cada condição do IF efetua uma operação e redireciona para um local diferente

Exemplo:

- Tela Inserir
- Tela Pesquisar

Navegação



Exemplo

```
<html><head><title>inserir.jsp</title></head>

<body>

<form action="Controladora?action=inserir"
      method="POST" >

    Nome:   <input type="text" name="nome" /><br/>
    E-mail: <input type="text" name="email" /><br/>

    <input type="submit" value="Inserir" />
    <input type="reset" value="Limpar" />

</form>
</body>
</html>
```

Exemplo

```
<html><head><title>pesquisar.jsp</title></head>

<body>

<form action="Controladora?action=pesquisar"
      method="POST" >

    Texto:   <input type="text" name="texto" /><br/>

    <input type="submit" value="Pesquisar" />
    <input type="reset" value="Limpar" />

</form>
</body>
</html>
```

Exemplo

```
@WebServlet(urlPatterns = {"/Controladora"})
public class Controladora extends HttpServlet {
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String action = request.getParameter("action");
        if (action.equals("inserir")) {
            // insere
            RequestDispatcher rd = getServletContext().
                getRequestDispatcher("/mensagem.jsp");
            rd.forward(request, response);
        }
        else if (action.equals("pesquisar")) {
            // pesquisa
            RequestDispatcher rd = getServletContext().
                getRequestDispatcher("/mostrar.jsp");
            rd.forward(request, response);
        }
        else {
            RequestDispatcher rd = getServletContext().
                getRequestDispatcher("/erro.jsp");
            rd.forward(request, response);
        }
    }
}
```

Exemplo

```
<html><head><title>mensagem.jsp</title></head>

<body>

    <h1>Usuário inserido com sucesso</h1>

</body>

</html>
```

Exemplo

```
<html><head><title>mostrar.jsp</title></head>
<body>
  <h1>Dados para serem mostrado</h1>
</body>
</html>
```

Exemplo

```
<html><head><title>erro.jsp</title></head>
<body>
  <h1>Ação não reconhecida</h1>
</body>
</html>
```

Quantas Servlets Implementar?

1 servlet por ação

- Facilmente o número de arquivos/servlets fica impraticável

1 servlet para a aplicação toda

- Padrão de Projeto **FrontController**
- Bom para Frameworks
- Problemas que podem aparecer:
 - Ou a complexidade da servlet aumenta muito
 - Ou seu tamanho (quantidade de ifs) aumenta muito

Equilíbrio: 1 servlet por módulo, por assunto, etc

Exercícios



1. Criar um arquivo **index.jsp** que contém dois links, um para **inserir.jsp** e outro para **mostrar.jsp**
 - Criar a aplicação anterior, com as várias ações
2. Criar uma aplicação com 3 arquivos

dados.jsp – JSP com formulário para entrada de NOME e IDADE

Controladora – Servlet que recebe a requisição, deve colocar o nome todo em maiúsculas e multiplicar a idade por 2, então passar estes novos dados para mostrar.jsp

mostrar.jsp – JSP que deve mostrar os novos dados de NOME e IDADE

Exercícios



3. Criar uma aplicação com 3 arquivos

formulario.jsp – JSP com formulário para entrada de NOME e IDADE

Controladora – Servlet que recebe a requisição, deve inserir o nome e a idade em uma tabela de algum banco de dados e redirecionar para mostrar.jsp com uma mensagem de sucesso

mensagem.jsp – JSP que deve mostrar a mensagem de sucesso passada

Exercícios



4. Criar uma aplicação com 3 arquivos

formulario.jsp – JSP com formulário para entrada de um texto

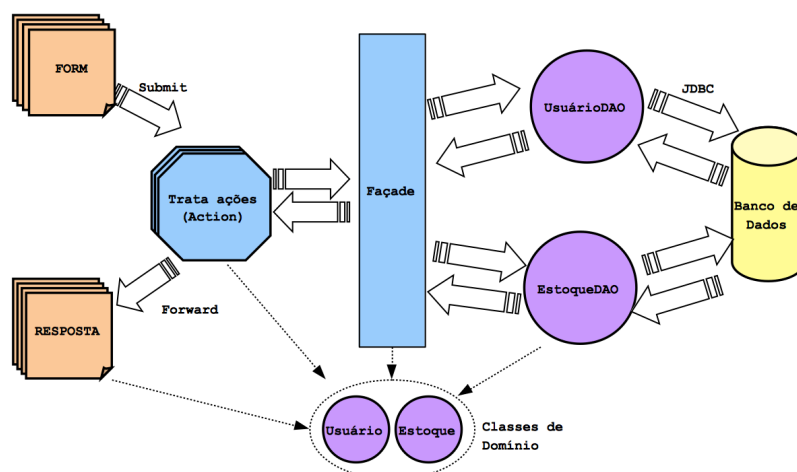
Controladora – Servlet que recebe a requisição, deve pesquisar todas as pessoas armazenadas no banco de dados que contém o texto passado como parte do nome e redirecionar para mostrar.jsp passando uma lista de nomes

mensagem.jsp – JSP que deve mostrar a lista de nomes

5. Integrar as duas aplicações anteriores na mesma aplicação, fazendo a Servlet tratar as duas requisições e redirecionar corretamente

Arquitetura da Aplicação

Arquitetura



Façade

Padrão de Projeto

Usado para "esconder" a complexidade de um subsistema ou parte do sistema

No nosso caso:

- Esconde a complexidade do acesso ao banco de dados
- Cada método do Façade implementa uma ação de negócio solicitada por uma requisição
- Métodos podem ser estáticos ou não. Ex.:

```
ClientesFacade.inserirCliente(c);
```

```
ClientesFacade facade = new ClientesFacade();
facade.inserirCliente(c);
```

Façade

```
package com.ufpr.tads.web2.facade;
```

```
public class ClientesFacade {
    public static void inserirCliente(Cliente c) {
        ...
    }
    public static void removerCliente(int id) {
        ...
    }
    ...
}
```

Faade na Servlet

```

...
@WebServlet(urlPatterns = {"/Controladora"})
public class Controladora extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {

        // PASSO 1: Obter Parâmetros
        String strId = request.getParameter("id");

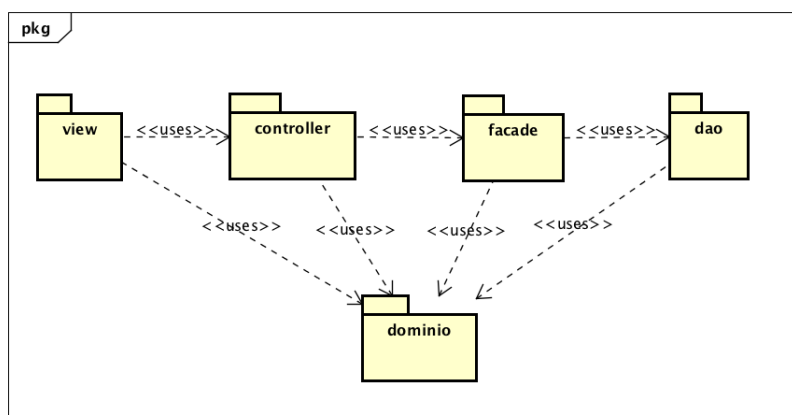
        // PASSO 2: Processar/Validar/Converter Parâmetros
        int id = Integer.parseInt(strId);

        // PASSO 3: Efetuar Ação (Usando Faade)
        ClientesFacade.removerCliente(id);

        // PASSO 4: Redirecionamento
        RequestDispatcher rd = getServletContext().
            getRequestDispatcher("/mostrar.jsp");
        rd.forward(request, response);
    }
}

```

Diagrama de Classes de Implementação



powered by Astah

Diagrama de Classes de Implementação

Para cada pacote, fazer seu diagrama de classes

- **view, controller, facade e dao:**
 - Não possuem associações entre si
 - Adicionar as classes e apresetar o diagrama
 - Não há necessidade de mostrar atributos e métodos da plataforma, a não ser que sejam invocados nos Diagramas de Sequência
- **dominio:**
 - Aqui estão as classes de negócio
 - Estas contém uma modelagem rica, com associações, herança, interface, polimorfismo etc
 - Este é o diagrama de classes de análise, ou diagrama de classes de modelo, etc