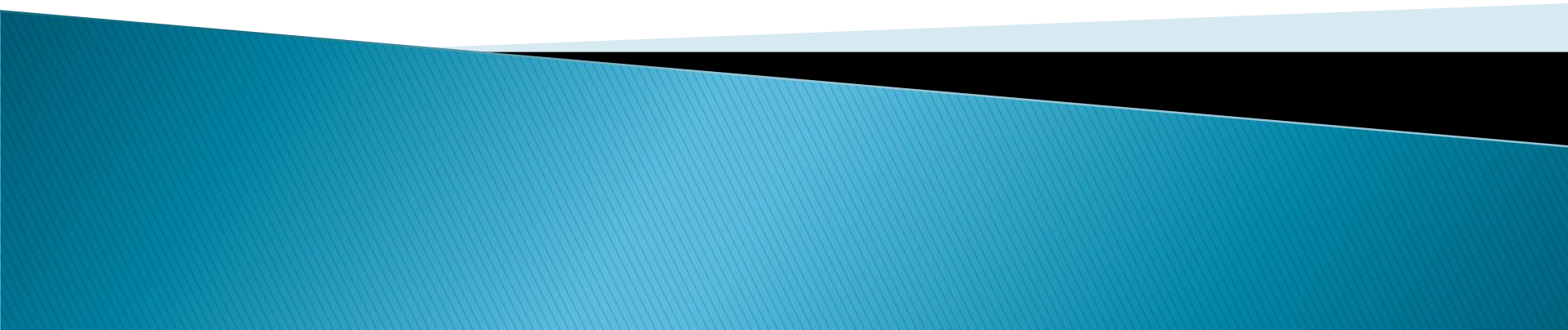


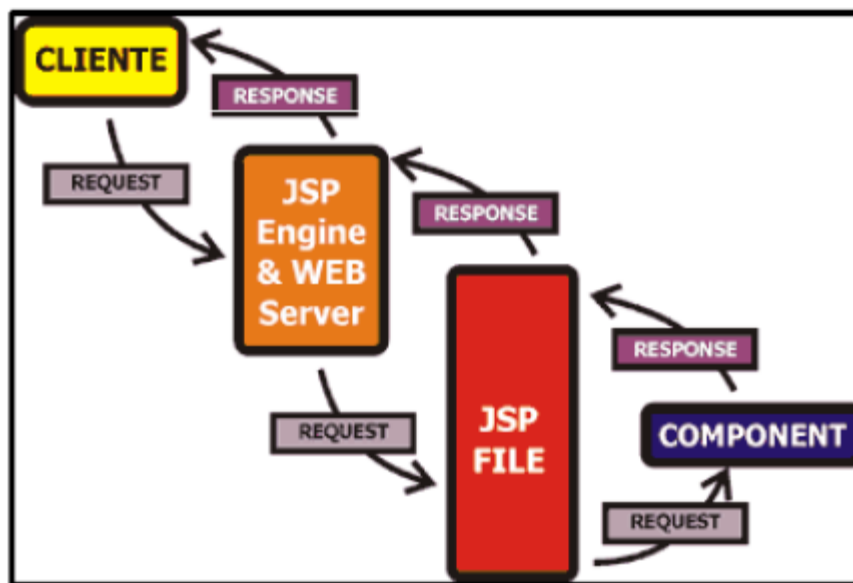
Java Server Pages (JSP)

Curso de Java Módulo 2.



O que são JavaServer Pages?

- ▶ Tecnologia Java para desenvolvimento de aplicações Web com conteúdo dinâmico.
- ▶ Funcionalmente são equivalentes aos Servlets.



JSP X Servlets

- ▶ Servlets incluem código de apresentação (HTML) nos componentes de processamento (código Java)
- ▶ JSP coloca instruções de processamento (código Java) no resultado a ser apresentado. A este resultado é dado o nome de “página” JSP.
- ▶ Em resumo:
 - JSP consiste de uma página HTML com código dinâmico

JSP: Quando usar?

- ▶ JSP é mais interessante quando a ênfase da requisição é apresentação de informações para o usuário
 - Pouco código Java e mais código HTML
- ▶ Servlets levam vantagem quando a ênfase é o processamento
 - Muito código Java e pouco de apresentação

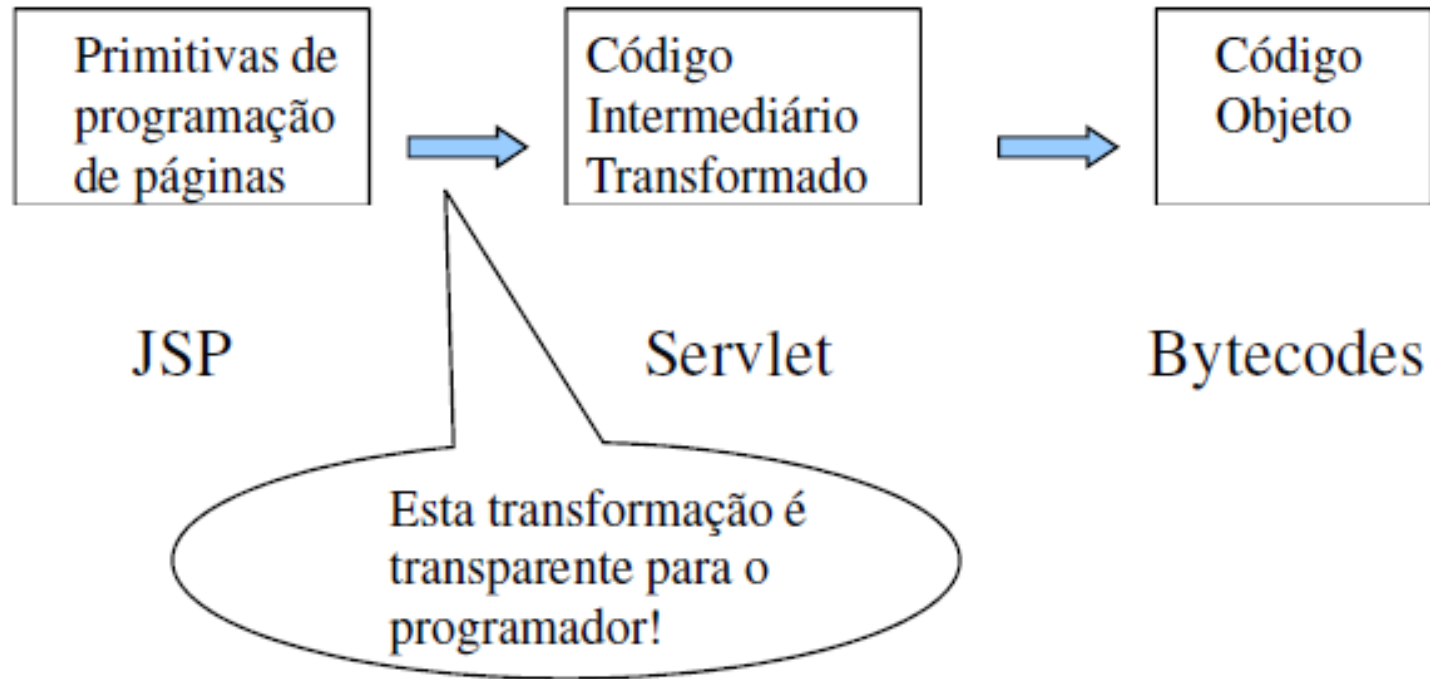
Vantagens

- ▶ Introduz novos comandos que facilitam o desenvolvimento de páginas
- ▶ Facilita a manutenção da apresentação. Para o designer, o sistema é composto apenas por páginas.

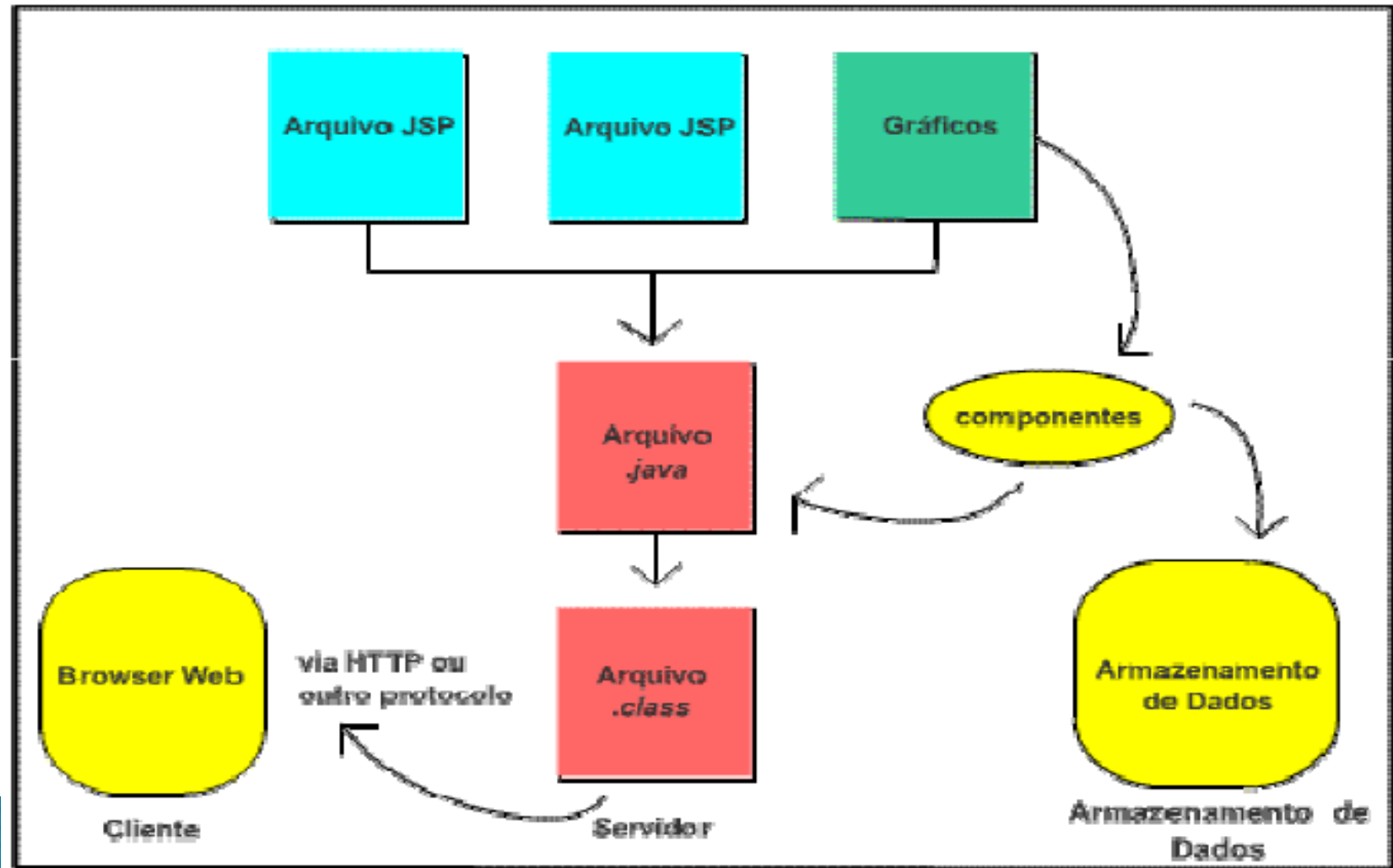
Desvantagens

- ▶ Falta de uniformidade na programação.
Uma página JSP é algo entre HTML e Java.
Isso pode afetar:
 - Legibilidade
 - Modularidade

JSP é um Servlet



JSP Engine



Funcionamento do JSP Engine

- ▶ O JSP Engine é um software para
 - Atender requisições de páginas JSP
 - Transformar uma página JSP requisitada em um servlet
 - Executar o servlet para atender às requisições JSP
- ▶ Pode ser acoplado a um servidor de páginas ou servidor de aplicações

Funcionamento do JSP Engine

- ▶ As operações de transformação, compilação e processamento da requisição são feitas automaticamente pelo engine
 - O JSP é traduzido e compilado na primeira vez que for acessado.
- ▶ O desenvolvedor apenas precisa disponibilizar o arquivo JSP em um local acessível ao engine.

Sintaxe JSP

Anatomia de uma página JSP

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

— JSP element

```
<html>
<body bgcolor="white">
```

— template text

```
<jsp:useBean
  id="userInfo"
  class="com.ora.jsp.beans.userInfo.UserInfoBean">
  <jsp:setProperty name="userInfo" property="*" />
</jsp:useBean>
```

— JSP element

```
The following information was saved:
<ul>
  <li>User Name:
```

— template text

```
    <jsp:getProperty name="userInfo"
      property="userName" />
```

— JSP element

```
  <li>Email Address:
```

— template text

```
    <jsp:getProperty name="userInfo"
      property="emailAddr" />
```

— JSP element

```
  </li>
</ul>
</body>
</html>
```

— template text

■ Uma página JSP

- Elementos JSP
- Template (tudo o que não for elemento da sintaxe JSP)
 - HTML
 - XML
 - etc.

Elementos básicos da sintaxe

▶ Diretivas

- Usadas na configuração do funcionamento da página no servidor

▶ Scripting

- Usados para codificar a página JSP com scripts escritos em Java.

Diretivas

- ▶ Instruções fornecidas ao container para configuração do JSP:
 - Como a página deve funcionar
 - Os recursos que ela irá usar
- ▶ Sintaxe:
 - `<%@ diretiva opção="valor" ... %>`
- ▶ Onde diretiva pode ser:
 - Page
 - Include
 - taglib (veremos com mais detalhes na Unidade II)

Diretiva: page

- ▶ Controla a estrutura do servlet através da importação de classes, customização da superclasse do servlet e definição do tipo de conteúdo da página.
- ▶ Podem ser declaradas várias vezes, mas cada opção só pode aparecer apenas uma vez na página.
 - Exceto a opção import que pode ser declarada várias vezes

Diretiva: page

- ▶ Recomenda-se declarar no início do arquivo JSP, antes mesmo da tag `<HTML>`.
- ▶ Opções da diretiva page:
 - Autoflush
 - Buffer
 - contentType
 - isErrorPage
 - errorPage

Diretiva: page

- ▶ Opções da diretiva page (cont):
 - Extends
 - Info
 - Session
 - isThreadSafe
 - Language
 - import

Diretiva page com autoflush

- ▶ Controla se o buffer de saída será descarregado automaticamente quando estiver cheio.
- ▶ O valor true indica que o conteúdo deve ser enviado ao browser quando estiver cheio.
- ▶ O valor false indica que uma exceção deve ser lançada quando o buffer ficar cheio.
- ▶ Exemplo:
 - `<%@ page autoflush="true"%>` `<!-- padrão --%>`

Diretiva page com buffer

- ▶ Define o tamanho do buffer usado pelo JSPWriter para mandar a resposta para o cliente.
- ▶ Útil para dimensionar um buffer para páginas grandes

Diretiva page com buffer

- ▶ Valores possíveis:
 - sizekb (size deve ser um número)
 - none (none implica que todo dado inserido na resposta é enviado imediatamente para o cliente)
- ▶ Exemplo:
 - `<%@ page buffer="32kb"%>` `<%--acumula até 32kb de dados--%>`

Diretiva page com contentType

- ▶ Define o cabeçalho de resposta, indicando o tipo MIME do documento que está sendo enviado ao cliente.
- ▶ Exemplo:
 - `<%@ page contentType="text/plain"%> <!-- default--%>`
 - `<%@ page contentType="text/html"%>`

Diretiva page com isErrorPage

- ▶ Define se a página corrente pode ser usada como uma página de erro para outra página JSP.
- ▶ Exemplo:
 - `<%@ page isErrorPage="false"%> <!--default-->`
 - `<%@ page isErrorPage="true"%>`

Diretiva page com errorPage

- ▶ Define qual página JSP será invocada quando uma exceção for levantada na página corrente.
- ▶ A exceção será apresentada na página de erro definida pelo programador.
- ▶ A página de erro referenciada deverá estar com a opção isErrorPage com o valor true.
- ▶ Exemplo:
 - `<%@ page errorPage="/paginas/PaginaErro.jsp"%>`

Diretiva page com extends

- ▶ Define qual a superclasse do servlet que será gerada a partir da página JSP
- ▶ Exemplo:
 - `<%@ page extends="MinhaClasse.class"%>`

Diretiva page com info

- ▶ Define uma string que será recuperada pelo servlet através do método `getServletInfo`.
- ▶ Exemplo:
 - `<%@ page info="Minha informação..."%>`

Diretiva page com session

- ▶ Define se a página utilizará algum recurso da sessão HTTP ou não.
- ▶ Exemplo:
 - `<%@ page session="false"%> <%-- indica que a sessão não será usada na página --%>`
 - `<%@ page session="true"%> <%-- indica que a sessão será usada na página. Equivalente a request.getSession("true") --%>`

Diretiva page com isThreadSafe

- ▶ Define se o servlet que resulta de uma página JSP irá implementar a interface `SingleThreadModel`.
- ▶ Se o valor for `false`, indica que o seu JSP não suporta acesso concorrente. Caso contrário (`true`), indica que o JSP suporta várias requisições simultâneas (valor default).
- ▶ Exemplo:
 - `<%@ page isThreadSafe="true"%> <!-- default --%>`
 - `<%@ page isThreadSafe="false"%>`

Diretiva page com language

- ▶ Utilizado para especificar a linguagem de programação base que está sendo utilizada. No momento só suporta Java
- ▶ Exemplo:
 - `<%@ page language="Java"%>`

Diretiva page com import

- ▶ Da mesma maneira que uma classe Java, define os pacotes e classes que serão usados na página
- ▶ Exemplo:
 - `<%@ page import="java.util.Date"%>`

Diretiva include

- ▶ Faz o container incluir o conteúdo do recurso em tempo de compilação
- ▶ O recurso é incluído na página antes de sua transformação para servlet.
- ▶ O recurso pode ser uma página HTML, JSP, texto, javascript, etc.
- ▶ Sintaxe:
 - `<%@ include file="arquivo.extensão"%>`

Scripting

- ▶ Código Java embutido em uma página JSP.
- ▶ Há diferenças semânticas entre o processamento em tempo de:
 - Compilação (transformação)
 - Requisição (processamento da página)

Scripting

- ▶ Existem scriptings para:
 - Declarações
 - Scriptlets
 - Expressões
 - Comentários
- ▶ Enquanto as diretivas comandam o container, scripting comanda o processamento da requisição.

Scripting: declarações

- ▶ Utilizadas para declarar variáveis e métodos usadas em blocos de código Java
- ▶ As variáveis declaradas pertencem à instância do servlet.
- ▶ Sintaxe:
 - `<%! <declaração Java>; %>`
- ▶ Exemplo:

```
<%! int i = 0; %>  
<%! public String getTitulo() {  
    return "O Título";  
};  
%>
```

Scripting: *scriptlets*

- ▶ É um bloco de código Java interpretado a cada requisição, sequencialmente na página
- ▶ É mapeado no método `service()` do servlet gerado.
- ▶ Sintaxe:
 - `<% <Código Java> %>`
- ▶ Exemplo:

```
<% if (Calendar.getInstance().get(Calendar.AM_PM) ==  
    Calendar.AM) { %>  
    Bom dia!  
<% } else { %>  
    Boa tarde!  
<% } %>
```

Scripting: *scriptlets*

▶ Exemplo:

```
<html>
<body>
<% if (Math.Random()*10 < 5) { %>
    Reprovado
<% } else { %>
    Aprovado
<% } %>
</body>
</html>
```

Scripting: expressões

- ▶ São expressões válidas da linguagem Java, avaliadas pelo servlet e enviadas ao seu output stream
- ▶ Tem o mesmo efeito do seguinte comando usado no servlet:
 - `response.getWriter().println(<expressao>);`
- ▶ Sintaxe:
 - `<%= <expressão Java> %>`
- ▶ Exemplo:
 - `<%= 10*2 %>`

Scripting: declarações e expressões

▶ Exemplo:

```
<html>
<head><title>Exemplo de Declarações JSP</title></head>
<body>
<h1>Declarações JSP</h1>
  <%! private int contador = 0; %>
  <h2>Número de Acessos a essa página
  <%= ++contador %>
</h2>
</body>
</html>
```

Scripting: comentários


- ▶ Comentário no formato HTML:
 - `<!-- comentário -->`
 - Vira um comentário em HTML quando é feita a conversão e é enviado para o browser do cliente.
- ▶ Comentário no formato JSP:
 - `<%-- comentário --%>`
 - Não é enviado para o browser do cliente

Objetos implícitos JSP

Objetos implícitos

- ▶ Alguns objetos da API existem nas páginas JSP sem necessidade de declaração
- ▶ Em geral, são definidos nos pacotes `java.servlet.*` e `java.servlet.http`

Objetos implícitos

- ▶ Alguns objetos são acessíveis através de condições específicas:
 - request
 - response
 - out
 - session
 - exception
 - application
 - pageContext
 - config
 - page (this)
- 

Objetos implícitos: request

- ▶ É um objeto do tipo `HttpServletRequest`
- ▶ Representa a requisição que causou a chamada do serviço

```
<html>
<body>
<%
    String nome = request.getParameter("nome");
    Cookie[] cookies = request.getCookies();
%>
</body>
</html>
```

Objetos implícitos: response

- ▶ É um objeto do tipo HttpServletResponse
- ▶ Permite configurar a resposta para o cliente como resultado de um requisição

```
<html><body>
<%   response.setContentType("text/html");
      File file = new File("c:\\temp\\figura.gif");
      if (!file.exists()){
          response.sendError(404,"arquivo não
                              encontrado");
      }%>
</body>
</html>
```

Objetos implícitos: out

- ▶ É um objeto do tipo `JspWriter`
- ▶ Permite a escrita no stream de saída de forma “bufferizada”

```
<%   if (numero>10) {  
        out.println("número é maior que 10");  
    } else {  
        out.println("número é menor que 10");  
    }  
%>
```

Objetos implícitos: session

- ▶ É um objeto do tipo HttpSession
- ▶ Permite acessar a sessão do usuário

```
<%@ page session="true" %>
<html><body>
<%   String nome = session.getAttribute("nome");
      Usuario usuario = fachada.getUsuario(nome);
      session.setAttribute("user", usuario);
%>
</body>
</html>
```

Objetos implícitos: application

- ▶ É um objeto do tipo ServletContext
- ▶ Permite acesso ao contexto da aplicação

```
<%@ page session="true" %>
<html><body>
<%   String url =

    application.getInitParameter("URLBanco");
    Connection con = fachada.getConexaoBanco(url);
    application.setAttribute("conexao", con);

%>
</body>
</html>
```

Objetos implícitos: exception

- ▶ É um objeto do tipo Throwable
- ▶ Representa a exceção dentro de uma página de erro
- ▶ Exceções surgidas em uma página são automaticamente transferidas para o objeto exception da página de erro

Objetos implícitos: exception

- ▶ O objeto exception existe apenas em páginas de erro

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

```
<html><body>
```

Aconteceu um erro no processamento:

```
<%= exception.getMessage %>
```

```
</body>
```

```
</html>
```


Outros objetos implícitos

- ▶ **pageContext:**
 - Dá acesso a informações e objetos implícitos com implementações mais específicas do servidor
- ▶ **page:**
 - Equivalente ao `this` de um programa típico Java.
- ▶ **config:**
 - Permite acesso a informações de configuração do JSP
 - É um objeto do tipo `ServletConfig`