# Curso de Java

## aula 13

Prof. Anderson Henrique

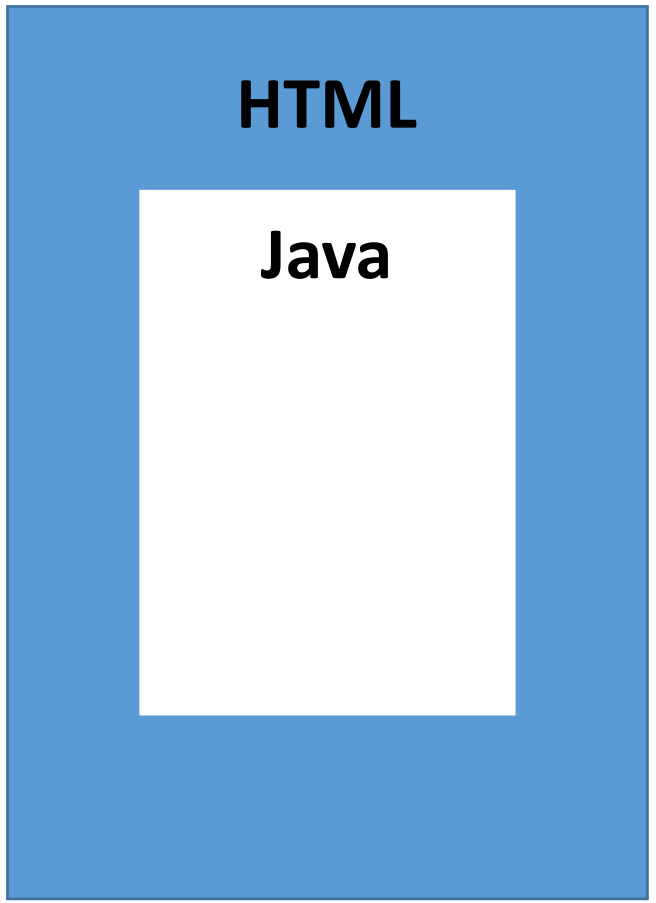
TECH
soluções em ti

# JavaEE (JSP e Servlets)

O desenvolvimento de aplicativos para a web utilizando a linguagem Java ocorre a partir de duas tecnologias desenvolvidas pela Sun Microsystems, em meados dos anos 90, **Java Server Pages** e **Servlets**.

O que se difere das duas tecnologias é a forma de se programá-las. No caso do JSP, páginas HTML (pré-requisito) contêm códigos escritos em Java.

Essa tecnologia permite a construção de páginas com conteúdo gerado dinamicamente. Ela consiste, basicamente, em componentes escritos na linguagem Java que ficam armazenados em um servidor de aplicações e atendem as requisições dos usuários que trafegam pela internet.

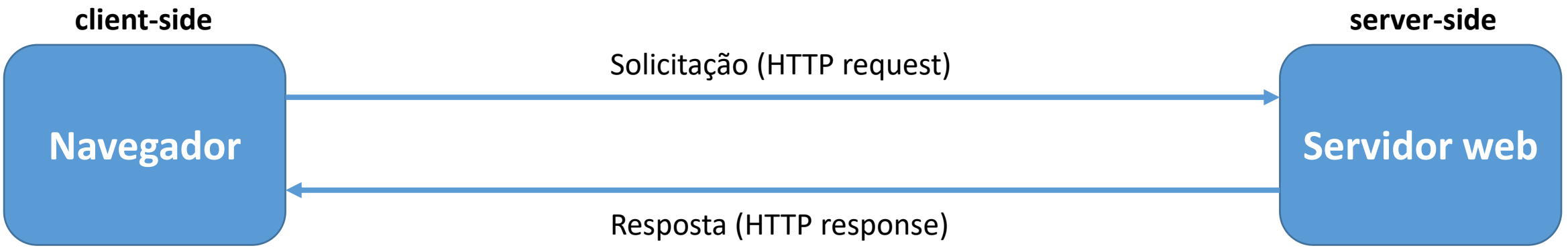



TECH
soluções em ti

Quando um desses usuários executam uma operação (como acesso a um banco de dados), esses componentes devolvem como resultado um documento padrão HTML que pode ser exibido pelo navegador.

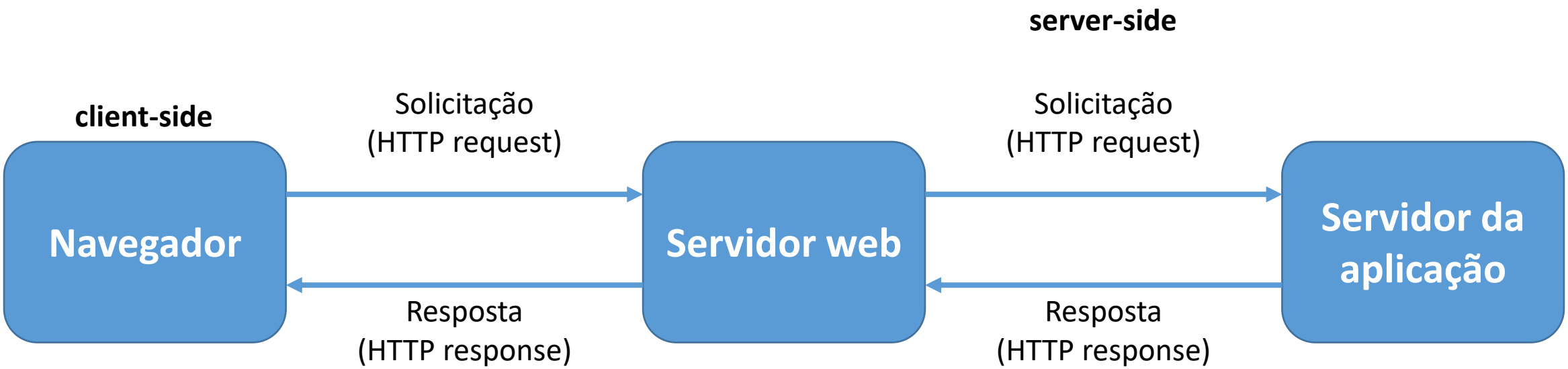
O processo normal que ocorre quando se acesso um site pode ser visualizado de forma gráfica a seguir. Esse processo se inicia com uma solicitação de serviço/recurso pela aplicação cliente (navegador).

Essa solicitação é encaminhada ao servidor web, o software responsável pela disponibilização dos documentos HTML, que a processa e devolve o documento referente à página que deve ser apresentada. (client-side/ server-side).


client-side
server-side
Navegador
Solicitação (HTTP request)
Servidor web
Resposta (HTTP response)

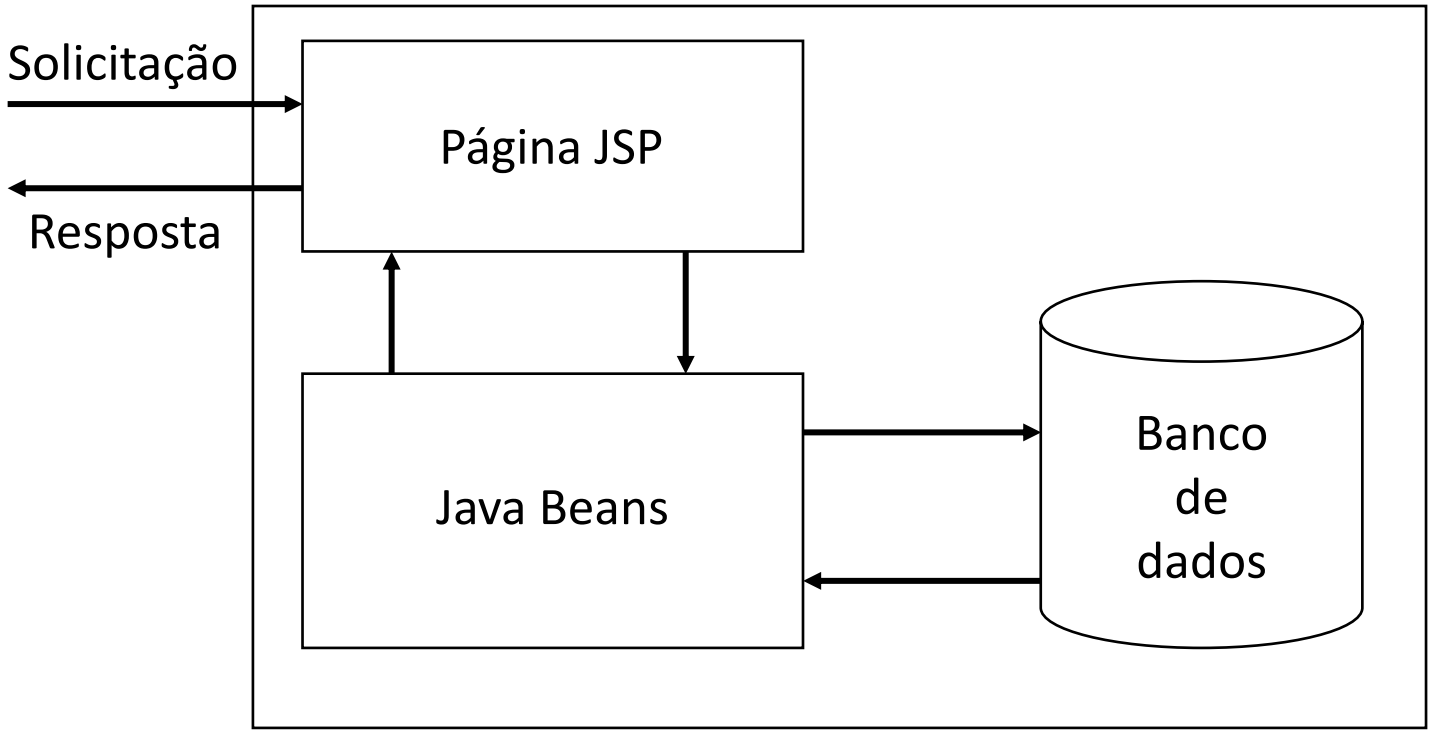
Nas aplicações JSP, como a página é gerada dinamicamente, não existe um arquivo/documento HTML no servidor web. O que ocorre é que o servidor web, ao receber uma requisição de um arquivo JSP, envia uma solicitação ao servidor da aplicação, que se encarrega de rodar o código correspondente.

O resultado do processamento é um documento padrão HTML que é retornado ao servidor web para envio de volta ao navegador. Esse processo é necessário porque o servidor web somente “reconhece” documentos no padrão HTML.

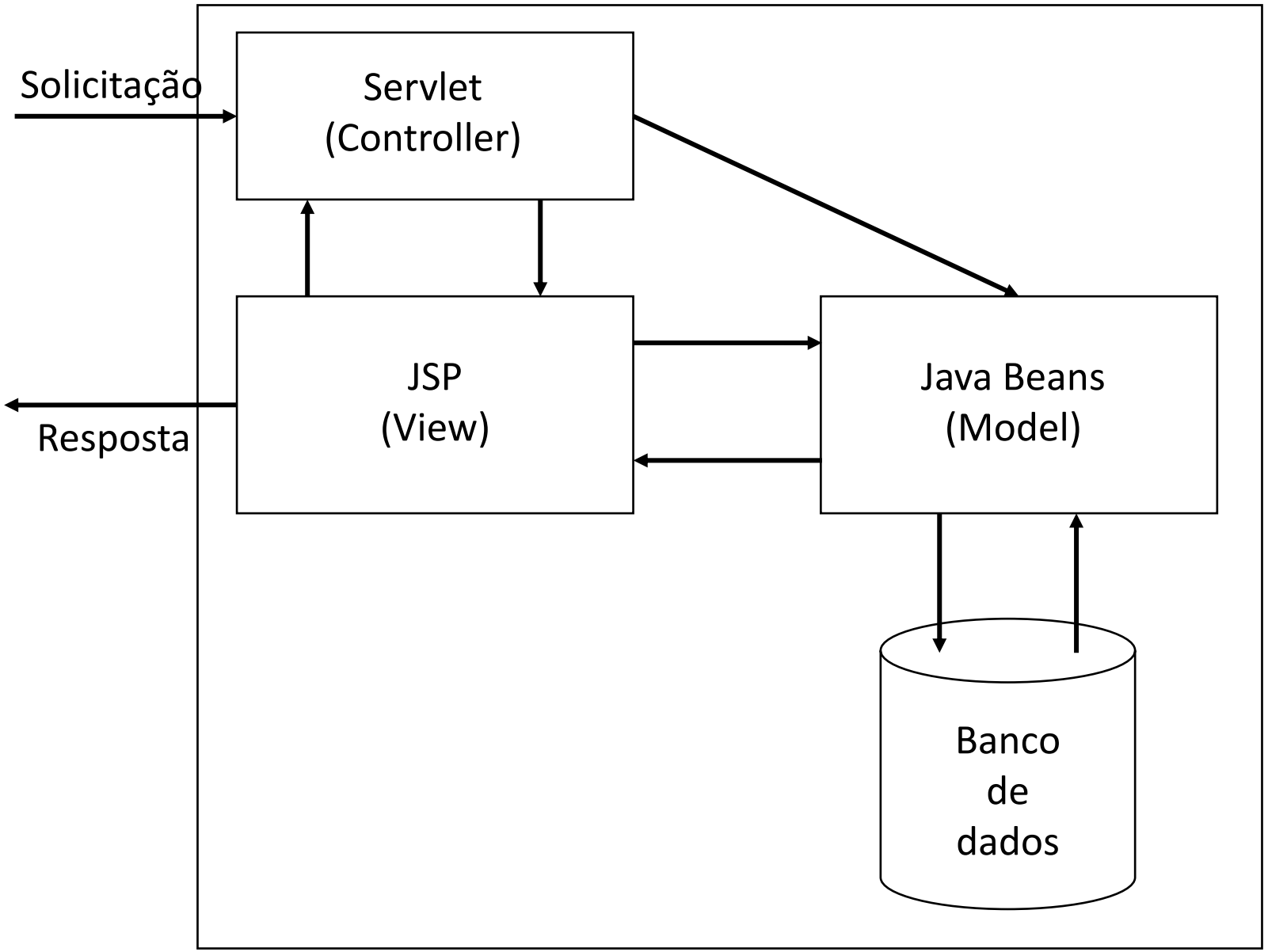


Construir aplicações complexas, formadas por centenas ou milhares de linhas de código, pode ser um trabalho árduo, principalmente nas futuras manutenções. Tendo isso em vista, as aplicações JSP podem ser classificadas em duas categorias: arquitetura **JavaBeans** e arquitetura **MVC**.

Verifique as imagens ilustrativas:




É caracterizada por ter a lógica de programação embutida em classes Java (Java Beans), que são invocadas pelas páginas JSP.




Segue o padrão MVC, ou seja, a lógica da aplicação é separada da apresentação. A camada Controller é manipulada por Servlets, enquanto a camada View fica a cargo das páginas JSP. Por outro lado, as classes Java Beans são responsáveis pela camada Model.

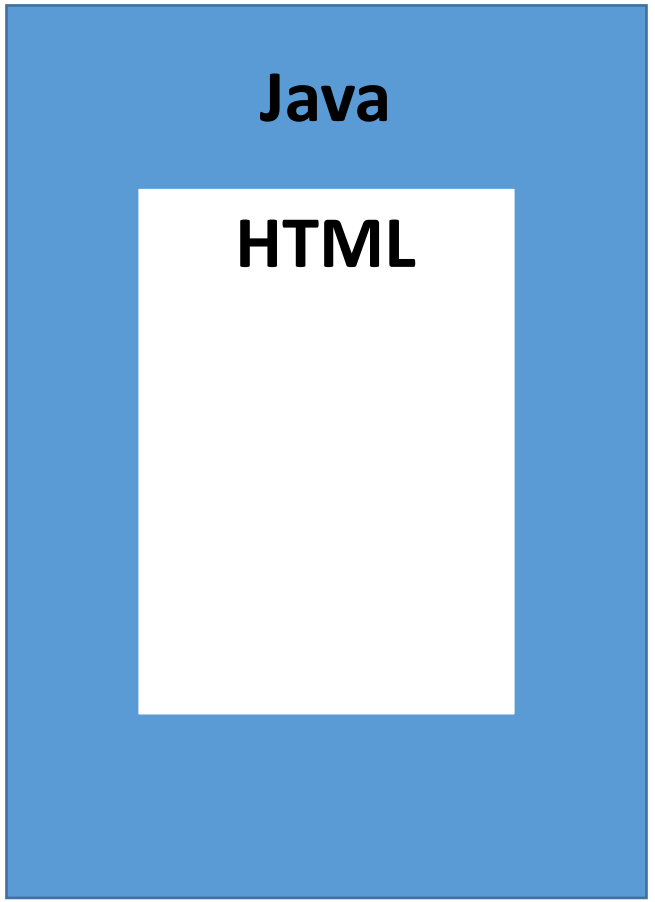
Podemos entender um Servlet como uma classe escrita em Java que contém código HTML para geração de páginas web. Ele é uma resposta a antiga técnica de desenvolvimento de aplicações web denominada **Common Gateway Interface** (CGI).

Da mesma forma que ocorre com páginas JSP, os Servlets são programas que rodam no servidor web, capturando as ações dos usuários enviadas pelo navegador e invocando rotinas também escritas em Java para acessar bancos de dados e apresentar os resultados na tela do usuário.

Se comparados à tradicional programação CGI, os Servlets oferecem maior eficiência, são mais fáceis de programar e mais seguros.

Dúvidas sobre HTML, CSS? Consulte o site w3schools (HTML e CSS).

Site: https://www.w3schools.com




TECH
soluções em ti

# JavaEE (Tomcat)

Após ter preparado o ambiente de desenvolvimento web (JDK, Tomcat, Netbeans) instalados, devemos executar mais alguns procedimentos preparativos para que seja possível criar e executar páginas **JSP**.

Precisamos criar um novo projeto, **Java Web**, dentro da pasta: **C:\webapps.**

No **Tomcat**, uma aplicação web é denominada Context (Contexto). Para armazená-la e distribui-la é preciso criar um diretório de contexto dentro da pasta **webapps**.

TECH
soluções em ti

Vamos criar o nosso primeiro documento JSP dentro da pasta **Páginas Web** chamado **OiMundo**.jsp

Como já foi informado, documentos JSP são documentos em formato HTML que possuem códigos embutidos escritos em Java. A inserção desses códigos  no interior de páginas HTML é possível graças ao novo par de tags **<% %>**.

Sendo assim, todo código em Java deve ser envolvido por esses marcadores, como mostra o seguinte exemplo:

Ex.:

```
<%
        out.println(“Programação Java para a Web”) ;
        out.println(“Oi Mundo!”) ;
%>
```

# JavaEE (JSP Elementos Sintáticos)

Para escrevermos códigos em Java inseridos em páginas HTML, dispomos de três diferentes, comumente conhecidos como elementos **sintáticos** e denominados **scripting**, **diretivas** e **ações**.

Cada uma delas tem o comportamento diferente no momento da execução do código.

# JavaEE (JSP Scripting)

O elemento estático scripting pode ser dividido em outros três tipos. O primeiro tipo é denominado **scriptlet**, e com ele o código é executado toda vez que a página for solicitada.

**Ex.:**

<% out.println(“Programação Java para a Web”); %>

O segundo tipo, conhecido como **declarações** é empregada para declarar métodos e variáveis utilizados quando a página é executada.

**Ex.:**

<%! int intMascaraBits = 204; %>

O terceiro e último tipo é a **expressão**, por meio da qual podemos exibir valores dentro de códigos HTML.

**Ex.:**

Valor da máscara de bits é <%= intMascaraBits %> 

# JavaEE (JSP Diretivas)

Elas são como instruções para o compilador, que informam como ele deve agir durante o processo de compilação do código-fonte. Do mesmo modo, as diretivas em programação JSP fazem com que o processamento da página JSP pelo Tomcat seja efetuado de uma maneira especial.

Para incluir uma diretiva no código, deve-se utilizar a seguinte sintaxe:

<%@ diretiva %>

Estão disponíveis três diretivas, a saber: **page**, **include** e **taglib**.

Com a diretiva **page** é possível especificar alguns atributos, como, por exemplo, a incorporação de uma biblioteca de classes ao código, como mostra o exemplo a seguir:

**Ex.:**

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

Essa diretiva informa ao Container que é necessário incluir a biblioteca de classes **Date** do pacote **java.util** para que a página possa ser executada com sucesso.

A diretiva **page** possui alguns atributos, dentre eles: **language**, **contentType**, **pageEncoding**.

A diretiva **include** permite que o conteúdo de um arquivo texto ou de código JSP, especificado por ela, seja inserido no código no momento da execução da página. Por exemplo, o código a seguir inclui o conteúdo do arquivo “copyright_pagina.txt”.

**Ex.:**

<%@ include file=“copyright_pagina.txt” %>

Por fim, temos a diretiva **taglib**, que permite o carregamento de uma biblioteca de tags definidas pelo próprio desenvolvedor. Por exemplo, suponha que você tenha criado sua própria biblioteca de tags JSP com o nome “minha_biblioteca_tags.jsp”. Em seu código, para fazer uso dessa biblioteca, você precisaria inserir a diretiva:

**Ex.:**

<%@ taglib “minha_biblioteca_tags.jsp” %>

# JavaEE (JSP Ações)

Normalmente, ações são utilizadas em conjunto com **JavaBeans**, que são bibliotecas de classes criadas pelo próprio programador para uso em seus trabalhos. Elas permitem especificar atividades que o Tomcat deve executar quando da solicitação de uma página por parte de um cliente.

Existem oito tipos de ações: **element**, **forward**, **getProperty**, **include**, **plugin**, **setProperty**, **text**, e **useBean**. Além das ações, existem também subações, que só podem ser utilizadas dentro do corpo de uma ação.

São elas: **atribute**, **body**, **fallback**, **param** e **params**.

O exemplo a seguir faz com que a saída de uma página JSP, denominada “processamento.jsp” seja incluída na página atual:

Ex.:

<jsp:include page=“processamento.jsp”/>

Na prática, como utilizamos estes elementos sintáticos?

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.util.Date"%>
<%@page import="java.text.SimpleDateFormat"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
        <title>Exemplo de Página JSP</title>
    </head>
    <body>
        <%! int intContaAcesso = 0; %>
        <%
            java.util.Date DataAtual = new java.util.Date();
            String strData = new SimpleDateFormat("dd/MM/yyyy")
                                                .format(DataAtual);
        %>
        <h3>Data Atual: <%= strData %></h3>
        <%
            out.println("<h4>Contagem de Atualizações de Página </h4>");
            intContaAcesso++;
        %>
        <h3>
            Número de vezes que você atualizou esta página:
            <%= intContaAcesso%>
        </h3>
    </body>
</html>
```

Diretiva page

Scripting declaração

Scripting expressão

Scripting expressão

TECH
soluções em ti

# JavaEE (JSP Objetos implícitos)

No exemplo anterior, vimos que a saída de dados é obtida por meio do método **println** do objeto **out**. Esse objeto faz parte de um grupo denominado **objetos implícitos**. Esses objetos são instanciados automaticamente para serem utilizados a qualquer momento.

Do mesmo modo temos o objeto implícito que permite a recuperação de dados digitados em campos de formulários. Esse objeto é denominado **request**. O método utilizado para recuperar dados é o **getParameter( )**.

# JavaEE (JSP e Formulário)

Teremos dois arquivos, um com a extensão **html** e o outro com a extensão **jsp**.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Form Trapezio</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h3>Cálculo da Área do Trapézio</h3>
        <form method="get" action="area_trapezio.jsp">
            <h3>Digite as informações</h3>
            <p>Digite o valor da base menor<input type="text" name="base_menor"></p>
            <p>Digite o valor da base maior<input type="text" name="base_maior"></p>
            <p>Digite o valor da altura    <input type="text" name="altura"></p>
            <p><input type="submit" value="calcular"></p>
        </form>
    </body>
</html>
```

Este é o arquivo que será chamado pelo atributo **action** da tag **form**, quando clicarmos no botão calcular.

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cálculo da Área do Trapézio</title>
    </head>
    <body>
        <%
            int intBaseMenor, intBaseMaior, intAltura, intArea;
            intBaseMenor = Integer.parseInt(request.getParameter("base_menor"));
            intBaseMaior = Integer.parseInt(request.getParameter("base_maior"));
            intAltura    = Integer.parseInt(request.getParameter("altura"));

            intArea = ((intBaseMenor + intBaseMaior) * intAltura) / 2;
        %>
        <h3>Valor da área do trapézio: <%= intArea %></h3>
    </body>
</html>
```

**Objetos implícitos**

TECH soluções em ti

Prosseguiremos no próximo slide... Com JEE (JSP e Servlets)

Professor: Anderson Henrique

Programador nas Linguagens Java e PHP