



# Curso de Java

## aula 15

Prof. Anderson Henrique

# JavaEE (Uso de Servlets)

Como já foi mencionado, toda aplicação escrita em **JSP** é convertida automaticamente pelo **Tomcat** em um **Servlet**, pois é isso que ele realmente executa.

No entanto, não é muito produtivo desenvolver toda nossa aplicação em Servlet, principalmente porque temos o código responsável pela montagem da página HTML (a interface com o usuário) embutido no código Java.

Além disso, há uma complexidade maior no desenvolvimento de Servlets, pois não é como no caso do JSP, que simplesmente codificamos e já podemos executar via navegador.

O que se costuma fazer é utilizar **JSP** para criar a camada de apresentação (**View**) e deixar a cargo dos **Servlets** a responsabilidade pelo processamento dos dados, ou seja, das regras do negócio (**Controller**).

Mas precisamos primeiramente entender a estrutura de um Servlet e o que é necessário para distribuir uma aplicação criada com essa técnica.

Um Servlet nada mais é do que uma classe Java e, sendo assim, o código deve ser gravado com a extensão **.java**, e não **.jsp**, como fizemos até agora.

Toda classe que implementa um Servlet deve ser gravada em um pacote. Isso facilita a implantação/distribuição da aplicação.

Uma classe Java precisa estender a classe **HttpServlet** para ser um Servlet de fato. Por outro lado, essa classe é abstrata, o que significa que devemos implementar as funcionalidades dos seus métodos.

Dois desses métodos, muito utilizados, são **doGet( )** e **doPost( )**, que servem para tratar requisições do cliente efetuadas por GET ou POST, definidas no atributo **method** do formulário HTML.

Falamos que um Servlet demanda um trabalho maior no desenvolvimento, pois exige algumas configurações extras para poder ser rodado. A principal diferença entre **JSP** e **Servlet** é a presença de um arquivo em padrão **XML** denominado **descritor de implantação**.

Esse arquivo deve ter o nome **web.xml**, e sua estrutura básica é a seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <servlet>
    <servlet-name>CalculoAreaCircunferencia</servlet-name>
    <servlet-class>CalculoAreaCircunferencia</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CalculoAreaCircunferencia</servlet-name>
    <url-pattern>/CalculoAreaCircunferencia</url-pattern>
  </servlet-mapping>
</web-app>
```

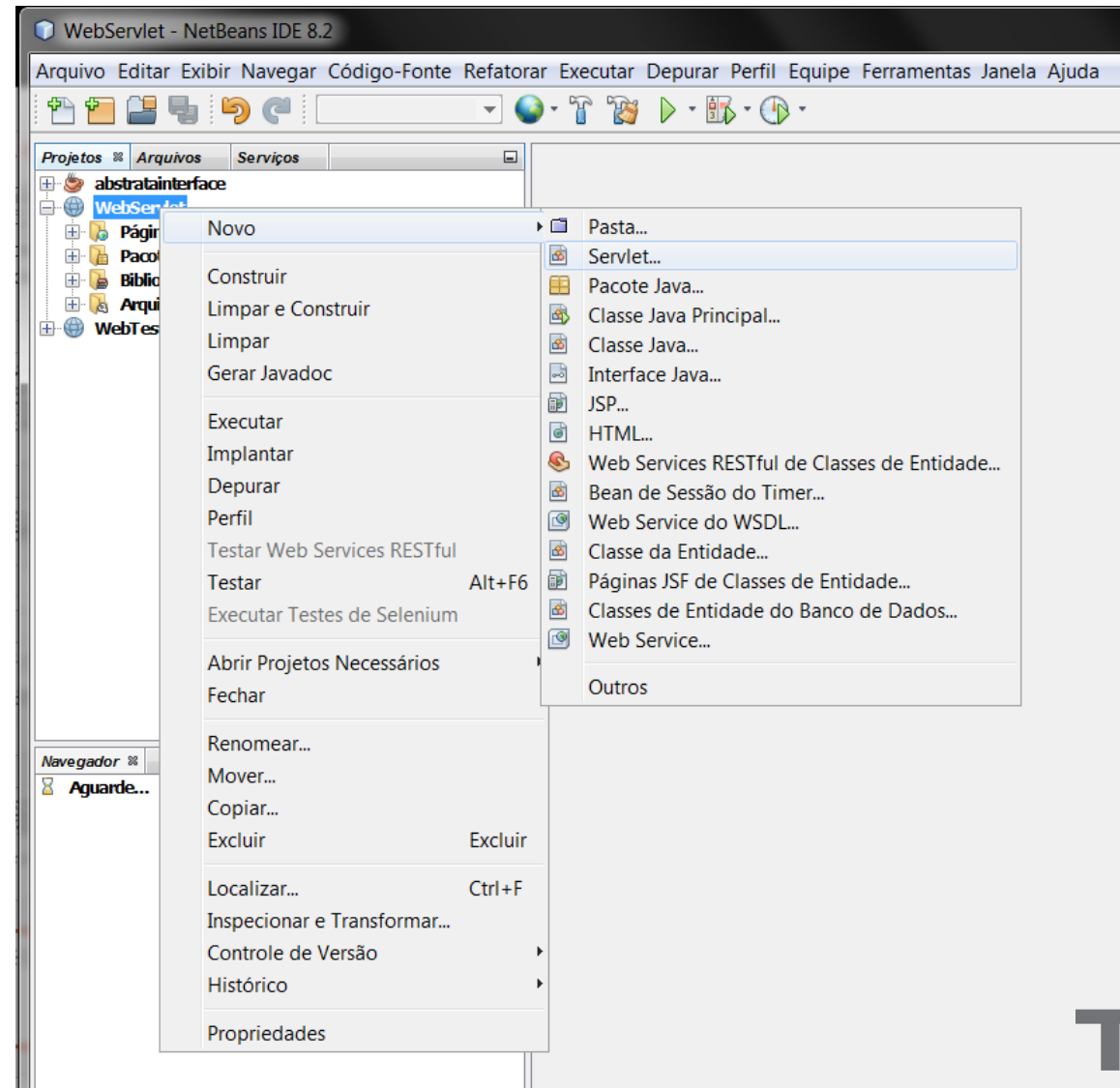
Nesse arquivo encontramos o par de tags **<web-app>** e **</web-app>**, responsáveis por agrupar os diversos Servlets que devem ser reconhecidos pelo Tomcat. A definição propriamente dita do Servlet é obtida com a configuração de dois grupos de elementos.

O primeiro grupo, definido por **<servlet>** e **</servlet>**, é utilizado para se especificar o nome interno do Servlet com o elemento **<servlet-name>** e **</servlet-name>** e o nome da classe (nome do arquivo **.class**) por meio do elemento **<servlet-class>** e **</servlet-class>**.

O segundo grupo de elementos, definido pelo elemento **<servlet-mapping>** e **</servlet-mapping>**, permite fazer o mapeamento do Servlet. Um nome deve ser especificado por meio do par de tags **<servlet-name>** e **</servlet-name>**. O valor desse elemento deve ser o mesmo atribuído ao elemento **<servlet>** e **</servlet>**. Já o elemento definido por **<url-pattern>** e **</url-pattern>** é utilizado para a especificação de parte da URL que usamos para acessar o Servlet.

Você pode criar esse arquivo manualmente, ou se preferir, deixar que o Netbeans o crie durante a configuração de parâmetros para criação do Servlet.

Vamos construir um pequeno exemplo para ver como tudo funciona. Crie um **novo projeto web: WebServlet**. Após a criação desse projeto, clique com o botão auxiliar do mouse sobre o ícone do projeto > **novo** > Servlet



**Obs.:** antes de criar o Servlet, devemos criar um Pacote dentro da pasta Pacotes de Código Fonte, Vamos nomear esse pacote: **Servlets**



Certifique-se que o Servlet será criado no pacote **Servlets**, e o nome do Servlet: **CalculoAreaCircunferencia**. Clique no botão próximo...

**New Servlet**

**Etapas**

1. Escolher Tipo de Arquivo
2. Nome e Localização
3. Configurar Implantação do Servlet

**Nome e Localização**

Nome da Classe:

Projeto:

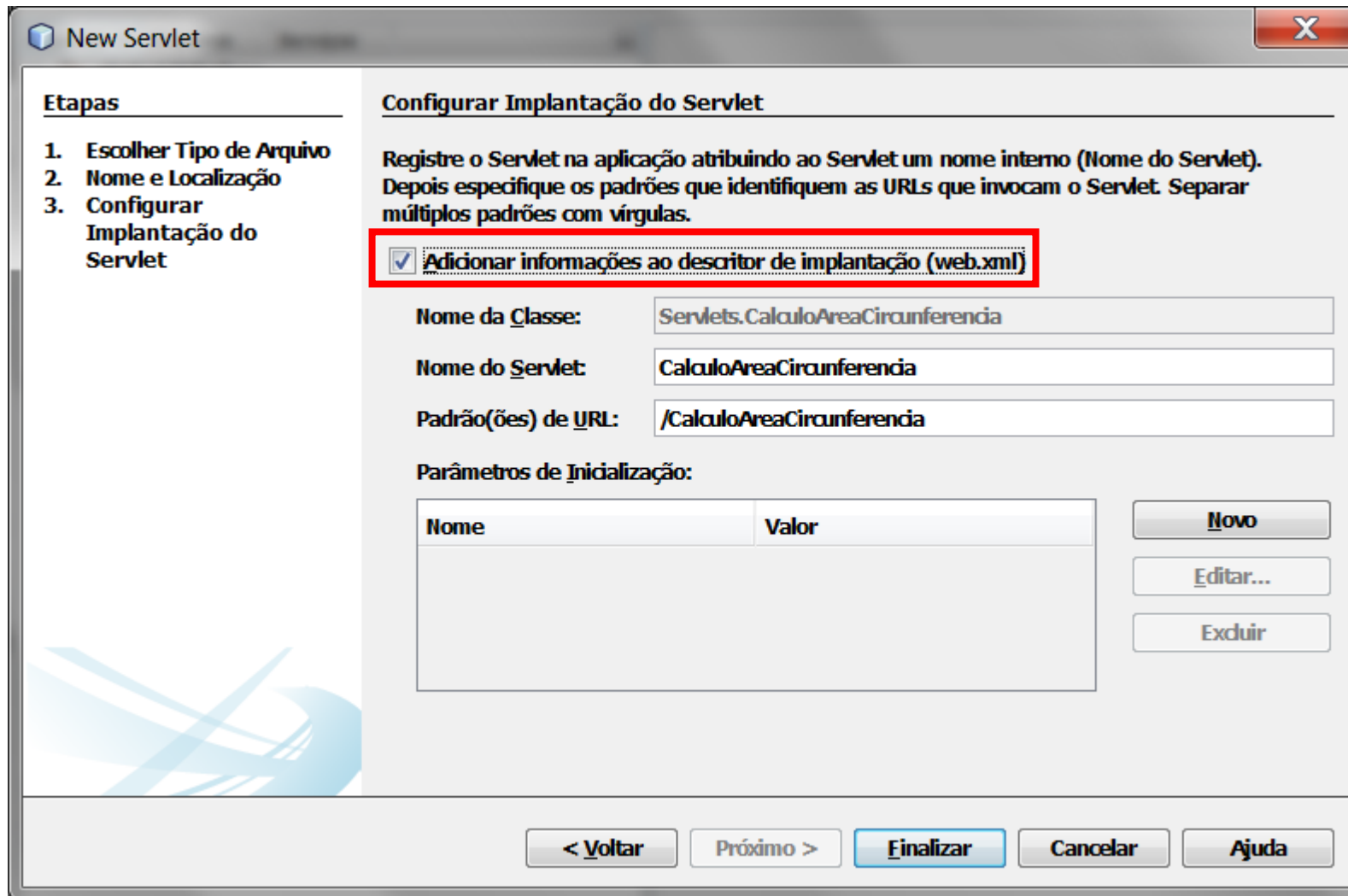
Localização:

Pacote:

Arquivo Criado:

< Voltar   **Próximo >**   Finalizar   Cancelar   Ajuda

Certifique-se de marcar a caixa de verificação: **Adicionar informações ao descritor de implantação (web.xml)**. Clique em finalizar...



**New Servlet**

**Etapas**

1. Escolher Tipo de Arquivo
2. Nome e Localização
3. Configurar Implantação do Servlet

**Configurar Implantação do Servlet**

Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.

☒ Adicionar informações ao descritor de implantação (web.xml)

Nome da Classe: Servlets.CalculoAreaCircunferencia

Nome do Servlet: CalculoAreaCircunferencia

Padrão(ões) de URL: /CalculoAreaCircunferencia

Parâmetros de Inicialização:

Nome	Valor
------	-------

**Novo**

**Editar...**

**Excluir**

**< Voltar** **Próximo >** **Finalizar** **Cancelar** **Ajuda**

Será exibido o arquivo Java (Servlet), você poderá ver os métodos **doGet( )** e **doPost( )**. A única coisa que eles fazem é chamar o método **processRequest( )**, cujo código precisa ser alterado no seguinte trecho:

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

    int intRaio;
    double dblArea;

    if(request.getParameter("fldRaio") != null){
        intRaio = Integer.parseInt(request.getParameter("fldRaio"));
    }else{
        intRaio = 0;
    }
    dblArea = (intRaio * intRaio) * Math.PI;

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Cálculo da Área de uma Circunferência</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h2>O valor da área da circunferência é: " + dblArea + "</h2>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

A primeira linha do código contém a diretiva **package**, que é responsável pela especificação do pacote ao qual a classe pertence (**Servlets**). As linhas seguintes são necessárias para que os pacotes utilizados pela nossa classe seja importados.

**java.io.IOException** - tratar qualquer erro que eventualmente ocorra durante o processamento de fluxo de saída de dados.

**javax.servlet.ServletException** – usa a classe ServletException, para poder lançar uma exceção no caso de o Servlet não ter sucesso no processamento da requisição do cliente.

Os últimos três pacotes precisam ser importados para dentro da classe tanto porque ela estende a classe **HttpServlet** como porque faz uso das interfaces **HttpServletRequest** e **HttpServletResponse**.

A primeira é responsável pelo atendimento das requisições pelos métodos **doGet( )** e **doPost( )**, enquanto a segunda é utilizada para se fornecer um objeto que represente a resposta ao cliente.

O método **processRequest( )**, por meio dos parâmetros **request** e **response**, recebe dois objetos, que representam uma **requisição** do cliente e uma **resposta** que deve ser retornada a ele.

A requisição é obtida graças à interface **HttpServletRequest**, enquanto a resposta é tratada com a interface **HttpServletResponse**. Esse método pode lançar exceções.

O objeto **response** chama o método **setContentType** para definir o tipo de resposta, que, no caso, é um documento no formato **text/html**, com o character set ajustado para **UTF-8**.

Um novo objeto do tipo **PrintWriter** é declarado e a ele é associado o objeto **response**, o que significa que todo o fluxo de saída de dados com destino ao objeto **out** será redirecionado para **response** automaticamente.

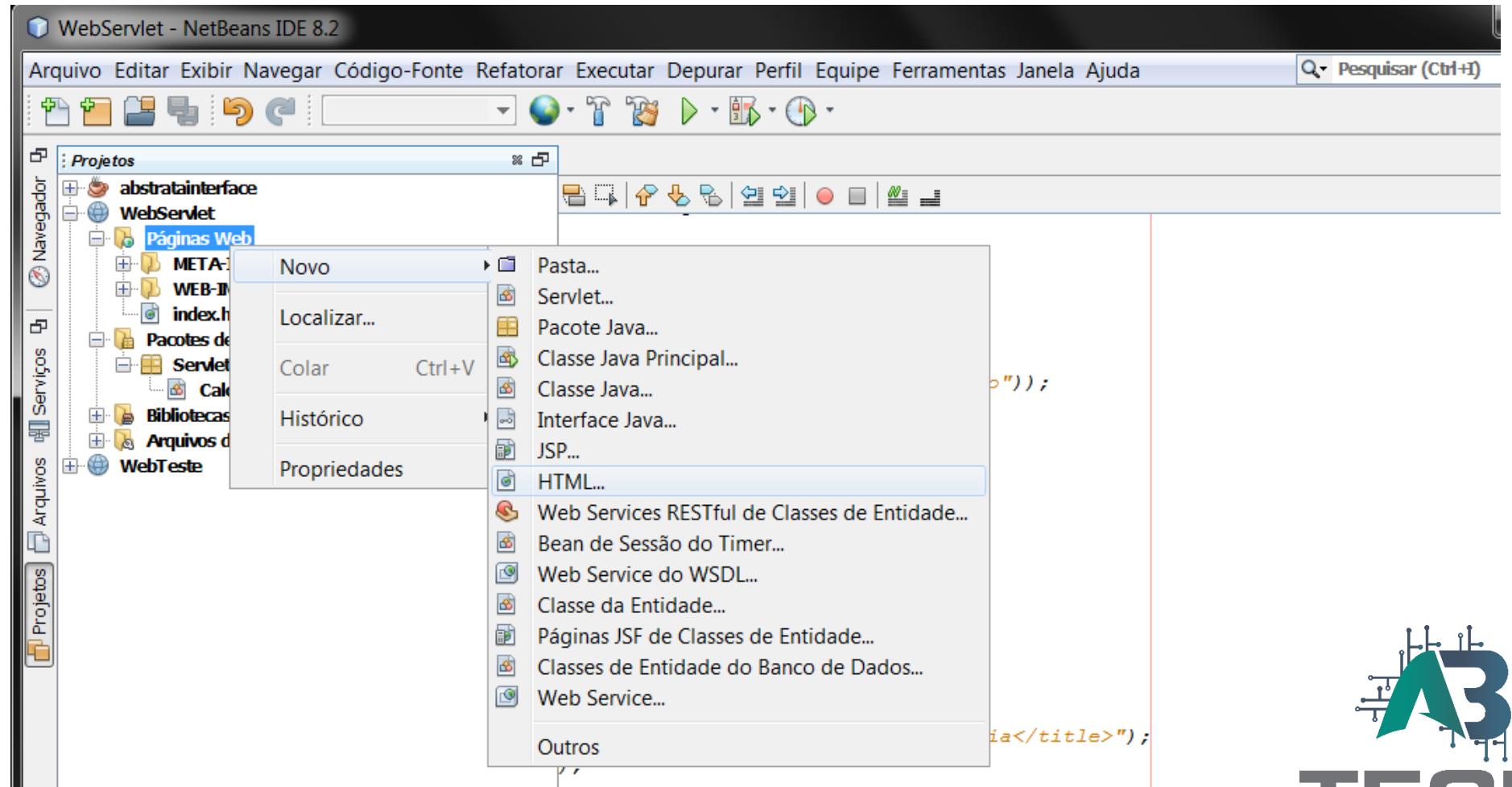
A saída de dados para o objeto **out** é obtida do mesmo modo que vimos quando trabalhamos com JSP, ou seja, chamando o método **println( )**. É aqui que construímos o documento **HTML** que será **retornado** ao usuário.

Os métodos **doGet( )** e **doPost( )** simplesmente invocam o método **processRequest( )** passando-lhe os mesmos parâmetros que receberam.

Preferimos definir um método específico para a execução das tarefas pertinentes ao Servlet. Se assim não tivesse sido feito, precisaríamos duplicar o código entre esses dois métodos, que não é nada eficiente.

# Testando o Servlet...

Precisamos criar uma página **HTML** que contenha o formulário com sua ação vinculada a classe. Nomeie o arquivo: **area\_circunferencia**





Após a criação do arquivo HTML, altere o código seguindo o exemplo abaixo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cálculo da área de uma circunferência</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <h3>Cálculo da área de uma circunferência</h3>
    <form method="get" action="CalculoAreaCircunferencia">
      <p>Digite o valor do raio da circunferência</p>
      <p><input type="text" name="fldRaio"></p>
      <p><input type="submit" value="Calcular"></p>
    </form>
  </body>
</html>
```

Note que no atributo **action** do formulário foi especificado o nome da classe que define o nosso Servlet, e ele não possui qualquer extensão.

Então concluímos que o grande objetivo de Servlets é separar a camada de apresentação (**View**) da camada de negócios (**Controller**).

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

Professor: Anderson Henrique

Programador nas Linguagens Java e PHP

