

Instituto de

computação



# Desenvolvimento Web Servlets

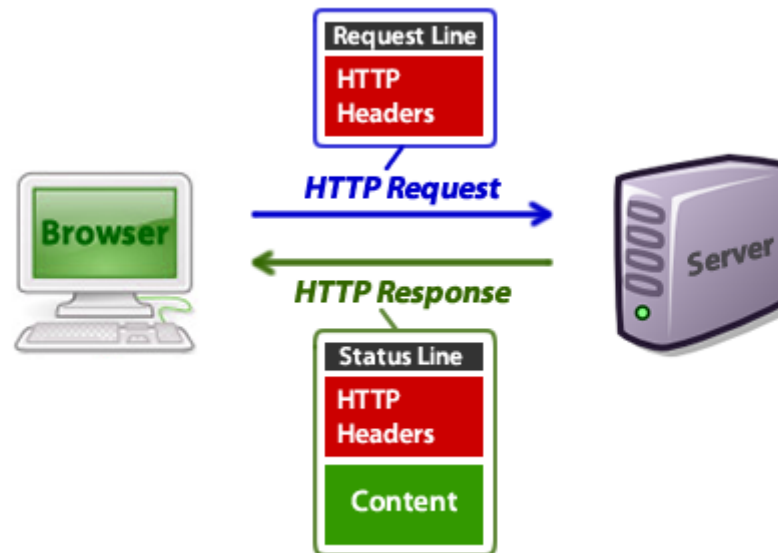
*Prof. Leonardo Cruz.*

leonardocruz@id.uff.br

Departamento de Computação, UFF

# Servlets

- O nome "servlet" vem da ideia de um pequeno servidor (*servidorzinho*, em inglês) cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente.

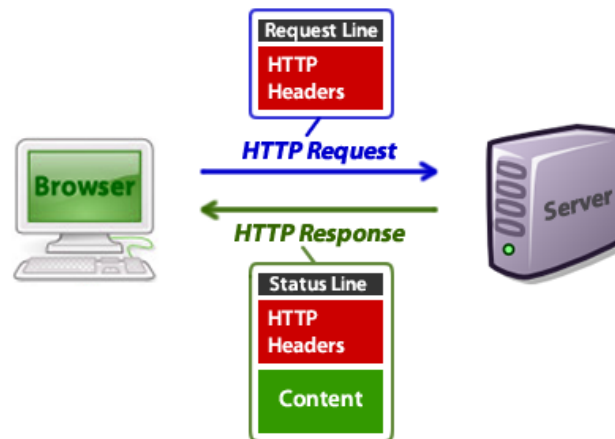


# Servlets

- Uma primeira ideia da servlet seria que cada uma delas é responsável por uma página, sendo que ela lê dados da requisição do cliente e responde com outros dados (uma página HTML, uma imagem GIF etc).
- Como no Java tentamos sempre que possível trabalhar orientado a objetos, nada mais natural que uma servlet seja representada como um objeto a partir de uma classe Java.

# Servlets

- Cada servlet é, portanto, um objeto Java que recebe tais requisições (**request**) e produz algo (**response**), como uma página HTML dinamicamente gerada.



# Servlets - Resumo

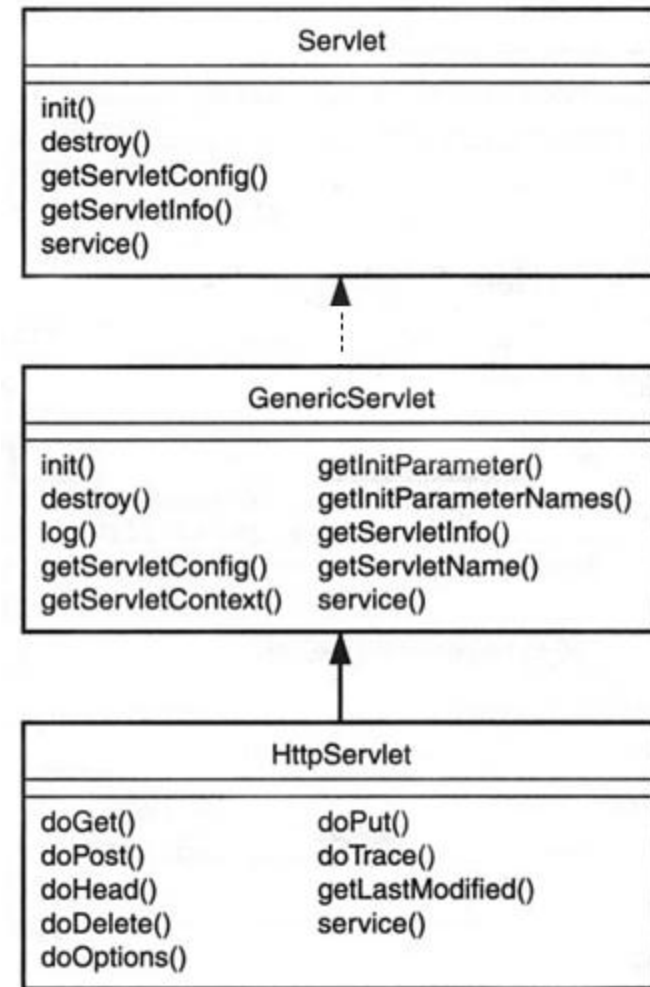
- São o bloco básico do desenvolvimento web em Java
- Servlets são classes hospedadas num servidor que respondem a requisições HTTP
- De maneira informal e resumida, servlets são classes java que geram páginas html
- Um servlet é uma especialização da **classe** **`javax.servlet.http.HttpServlet`**

# Servlets - ciclo de vida

- A implementação da **interface** Servlet (`javax.servlet.Servlet`) é usada para todos os servlets, porque é essa interface que encapsula os métodos do ciclo de vida do servlet.
- **public interface** Servlet

# Servlets - ciclo de vida

- O servlet é inicializado chamando o método **init ()**.
- O método de **service** para processar o pedido de um cliente.
- O servlet é encerrado chamando o método **destroy ()**.



# Servlets - ciclo de vida

## ■ O método init()

- Concebido para ser chamado somente uma vez.
- É chamado quando o servlet é criado pela primeira vez, e não chamado novamente para cada solicitação do usuário.



# Servlets - ciclo de vida

## ■ O método service ()

- O método de serviço () verifica o tipo de solicitação HTTP (GET, POST, PUT, DELETE, etc.) e chama doGet, doPost, doPut, doDelete, etc. adequado para tratar a solicitação.

# Servlets - ciclo de vida

## ■ O método `destroy()`

- quando o servlet não for mais necessário (por exemplo, quando a aplicação Web for removida do container), é chamado o método `destroy()`. Neste devem ser liberados os recursos alocados em `init()`..

# Container

- Os servlets não possuem um método `main()`.
- Eles estão sob o controle de outra aplicação Java chamada **Container**

# Container

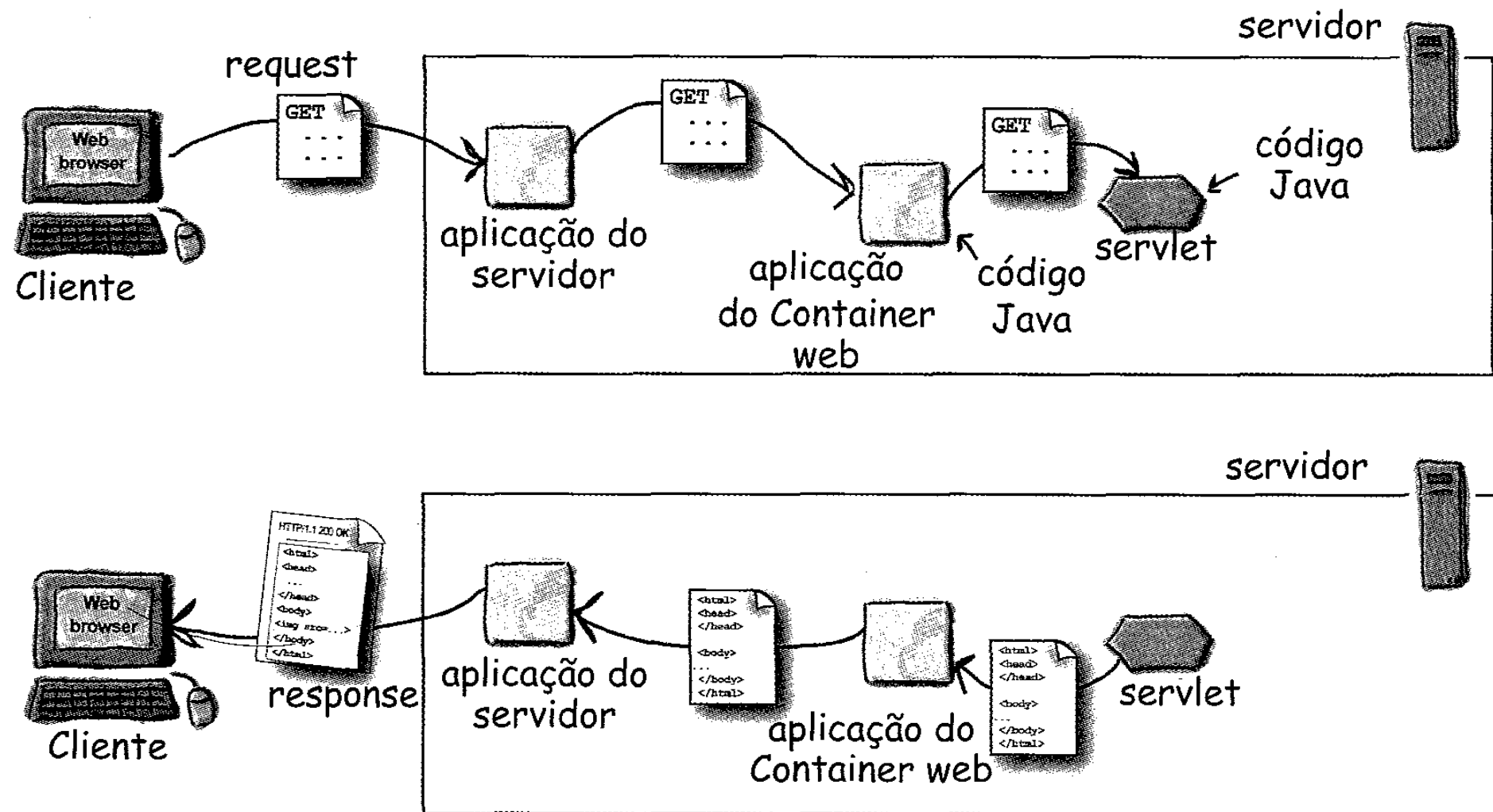
- Tomcat, Glassfish e JBoss são exemplos de Container



# Container

- Quando a sua aplicação web recebe uma solicitação para um servlet, o servidor entrega a solicitação **não ao servlet em si**, mas ao **container** no qual o servlet.
- É o container que entrega ao servlet a request e a response HTTP e chama os métodos do servlet doPost ou doGet.

# Container



# O que o Container oferece?

## ■ A infraestrutura...

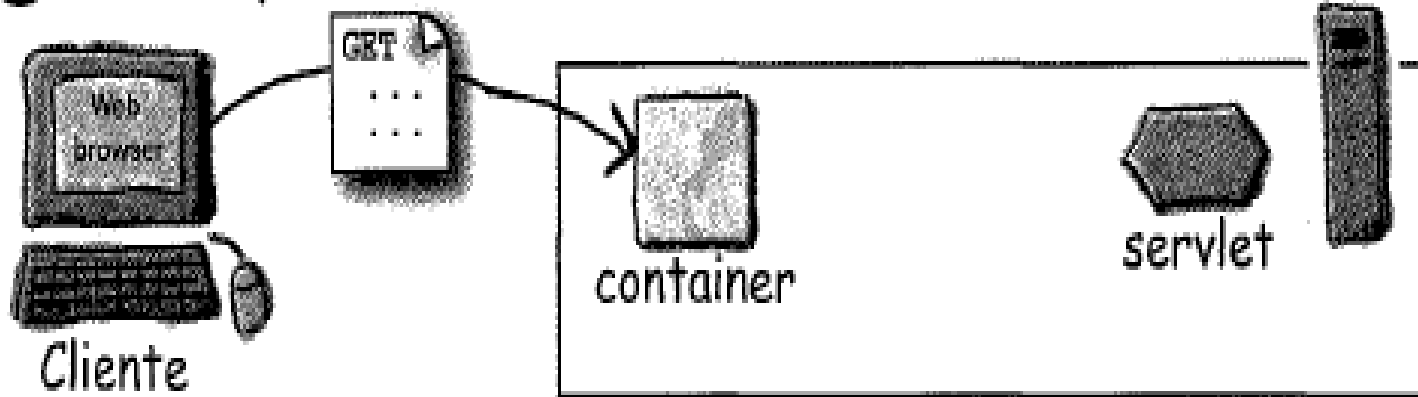
- ☐ Suporte para comunicações
- ☐ Gerenciamento do ciclo de vida do servlet
- ☐ Suporte a multithread
- ☐ Segurança
- ☐ Suporte JSP

- O programador se concentra na lógica do negócio e não na infraestrutura

# Como o Container trata uma solicitação?

1

request HTTP

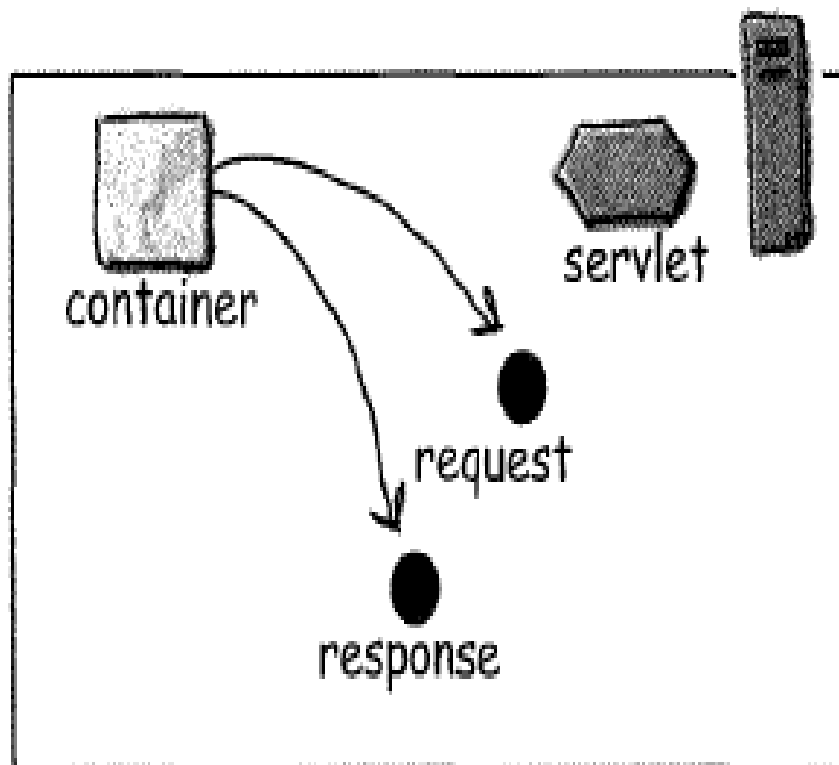


O usuário clica em um link que contém uma URL para um servlet, em vez de uma página estática.



# Como o Container trata uma solicitação?

2

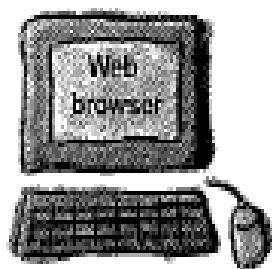


O container “vê” que a request é para um servlet e então ele cria dois objetos:

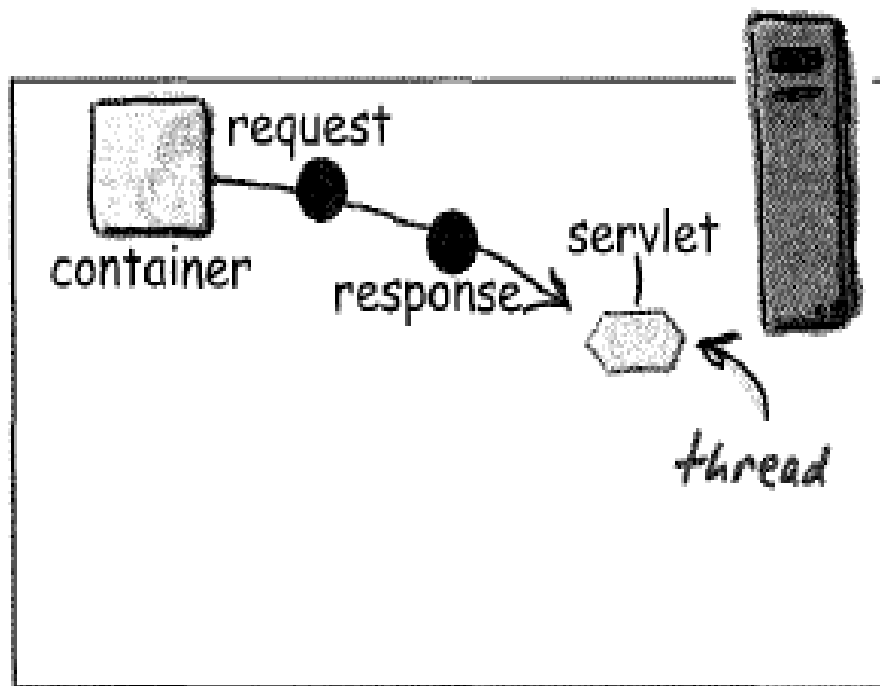
- 1) `HttpServletResponse`
- 2) `HttpServletRequest`

# Como o Container trata uma solicitação?

3



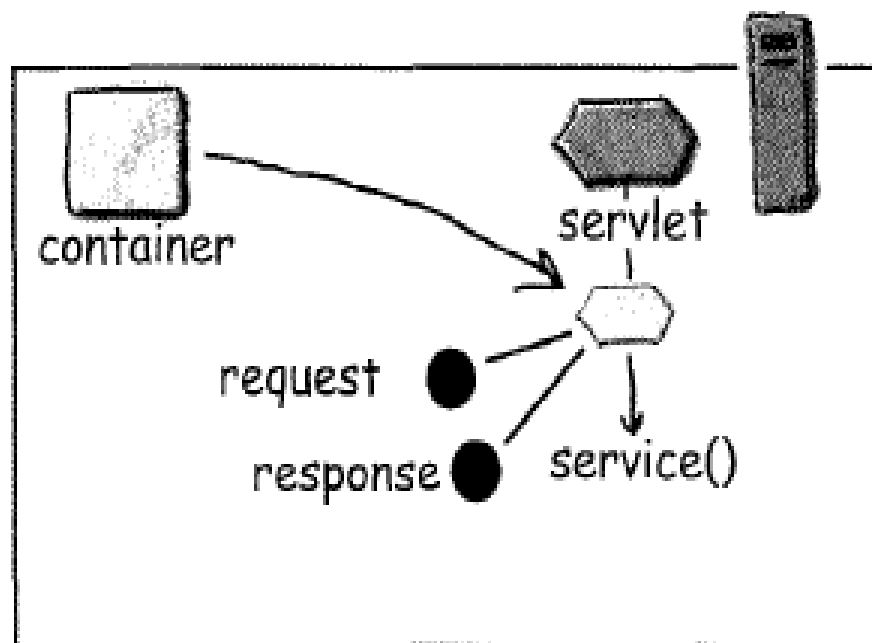
Cliente



O container encontra o servlet correto baseado na URL da request, cria ou aloca uma thread para essa request, e passa os objetos request e response para a thread do servlet.

# Como o Container trata uma solicitação?

4

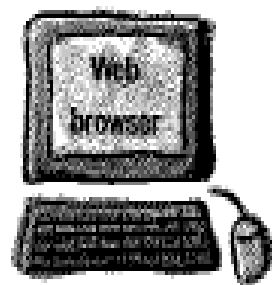


O container chama o método `service()` do servlet. Dependendo do tipo de request, o método `service()` chama ou o método `doGet()`, ou o método `doPost()`.

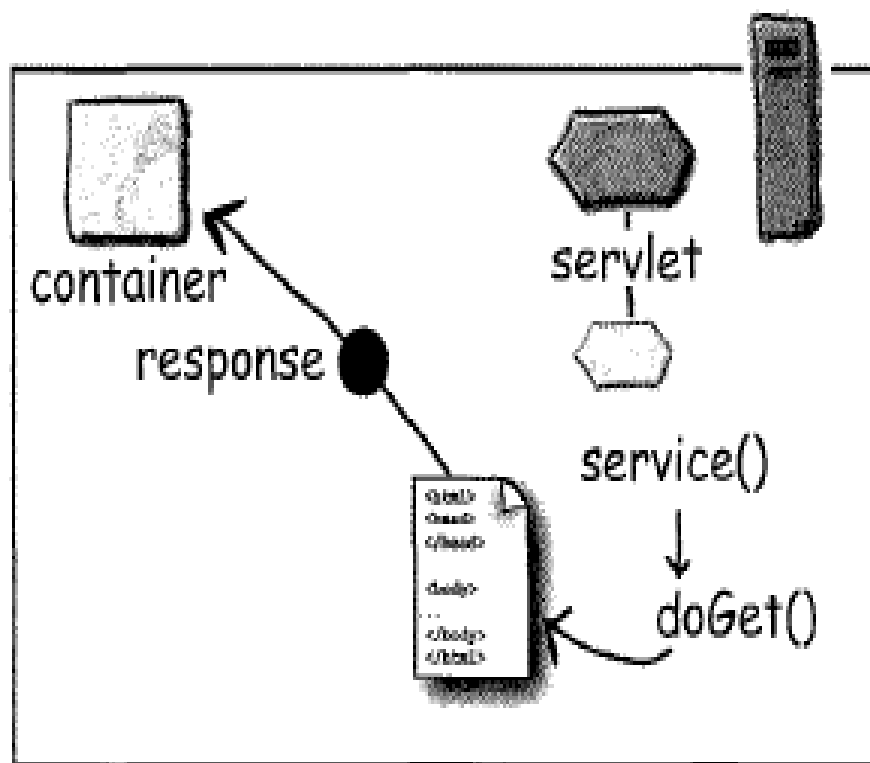
Para este exemplo, consideraremos que a request foi um HTTP GET.

# Como o Container trata uma solicitação?

5



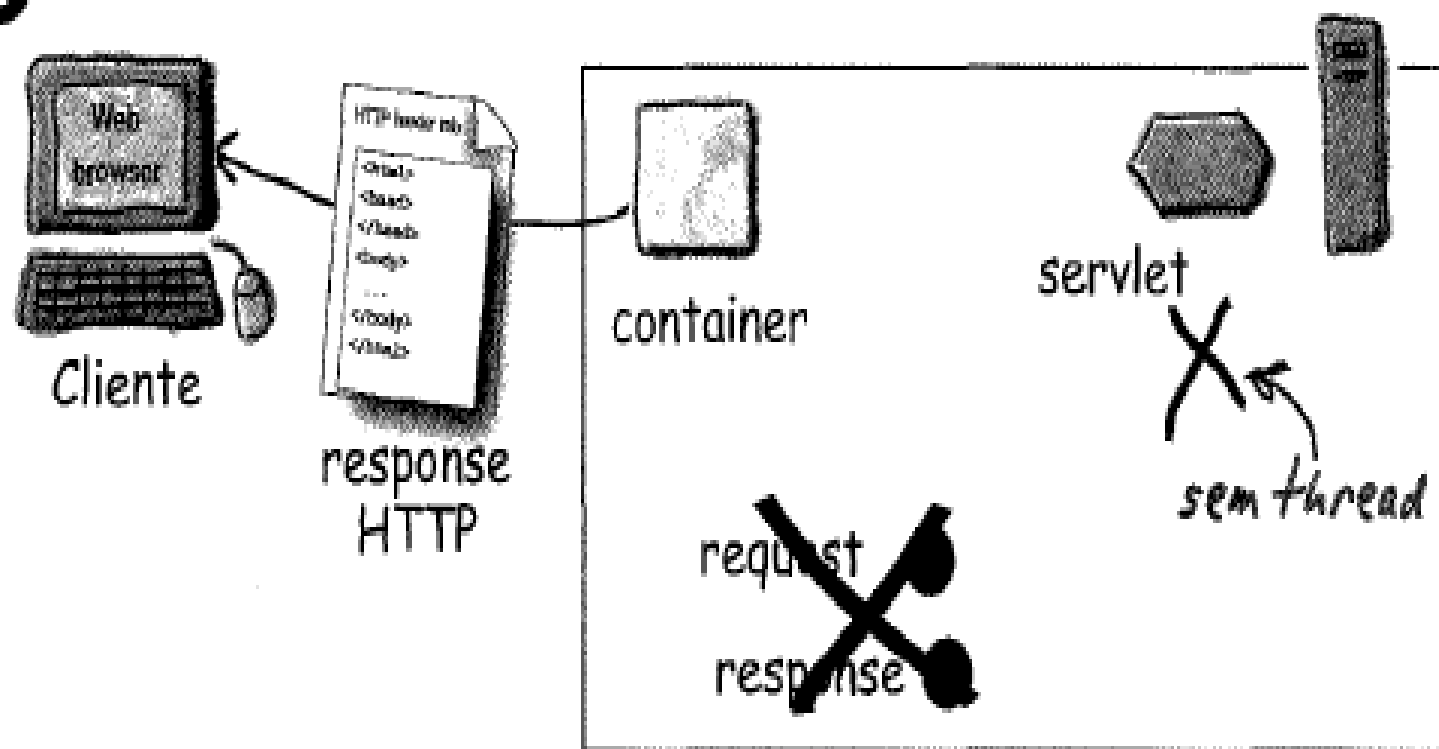
Cliente



O método doGet gera uma página dinâmica e a insere no objeto response. Lembre-se, o container ainda tem uma referência do objeto response!

# Como o Container trata uma solicitação?

6



O thread termina, o container converte o objeto response em uma response HTTP, envia de volta ao cliente e apaga os objetos request e response.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
public class Ch1Servlet extends HttpServlet {
```



```
    public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                       throws IOException {
```

**Método doGet**

**Recebe  
dois objetos**

```
        PrintWriter out = response.getWriter();
        java.util.Date today = new java.util.Date();
```

```
        out.println("<html> " +
                    "<body>" +
                    "<h1 align=center>HF\'s Chapter1 Servlet</h1>" +
                    + "<br>" + today + "</body>" + "</html>");
```



**Inserção do código html**

No mundo real, 99,9% de todos os servlets anulam ou o método `doGet()`, ou o `doPost()`.

99,9999% de todos os servlets são `HttpServlet`s.

Repare... nenhum método `main()`. Os métodos do ciclo de vida do servlet (como o `doGet()`) são chamados pelo Container.

```
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.io.*;
```

```
public class Ch2Servlet extends HttpServlet {
```

```
    public void doGet(HttpServletRequest request,  
                      HttpServletResponse response)  
        throws IOException {
```

É aqui que o seu servlet consegue as referências dos objetos `request` e `response` que o container cria.

```
        PrintWriter out = response.getWriter();  
        java.util.Date today = new java.util.Date();  
        out.println("<html> " +  
                    "<body>" +  
                    "<h1 style='text-align:center>" +  
                    "HF\'s Chapter2 Servlet</h1>" +  
                    "<br>" + today +  
                    "</body>" +  
                    "</html>");
```

Você pode conseguir um `PrintWriter` do objeto `response` que o seu servlet recebe do Container. Utilize o `PrintWriter` para escrever texto HTML no objeto `response`. (Você pode ter outras opções de saída além do `PrintWriter`, para escrever, digamos, uma figura, em vez de um texto HTML.)

## O método `doGet`

# Mas... Como o Container encontrou o servlet?





# Mas... Como o Container encontrou o servlet?

- A url que chega como parte da solicitação do cliente é mapeada para o servlet específico no servidor.
- Esse mapeamento pode ser feito de várias formas
- E cabe o programador configurar este mapeamento

# Mapeamento. Para começar...

- Um servlet pode ter três nomes...



# Um servlet pode ter três nomes...

- Um Nome da url (conhecido pelo mundo)
- Um nome interno (secreto) usado somente para a distribuição do servlet.
- Nome do arquivo (da classe com o seu pacote...)
- Mas para que isso tudo...

# Mapeamento

- Flexibilidade para mudanças e alterações...
  - Alterações de classes, pacotes não influem no mundo...
- Segurança
  - Não expõe a estrutura do servidor para o mundo

# Mapeamento

- Para isso, vamos fazer um mapeamento de uma servlet utiliza-se o arquivo **web.xml**, que fica dentro da pasta **WEB-INF**

# Mapeamento... O arquivo web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

# Mapeamento... O arquivo web.xml

Especificando o servlet ... Os três nomes...

```
<servlet>
```

```
<servlet-name>oi</servlet-name>
```

```
<servlet-class>br.uff.curso.dw.Oimundo</servlet-class>
```

```
<url-pattern>/oi</url-pattern>
```

```
</servlet>
```

# Mapeamento... O arquivo web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <servlet>
    <servlet-name>oi</servlet-name>
    <servlet-class>br.uff.curso.dw.Oimundo</servlet-class>
    <url-pattern>/oi</url-pattern>
  </servlet>
</web-app>
```



# Servlet - Implementação

- Netbeans... (o ambiente de desenvolvimento)

# NetBeans - (o ambiente de desenvolvimento)

Download o **NetBeans IDE 8.2**

8.1 | 8.2 | Desenvolvimento | Arquivo

Endereço de email (opcional):

Inscrever-se na newsletter: ☒ Mensal ☐ Semanal

☒ Permito me contatar neste email

Idioma do IDE: **Português (Brasil)**

Plataforma: **Windows**

Nota: Tecnologias em cinza não são suportadas para esta plataforma.

## Distribuições para baixar do NetBeans IDE

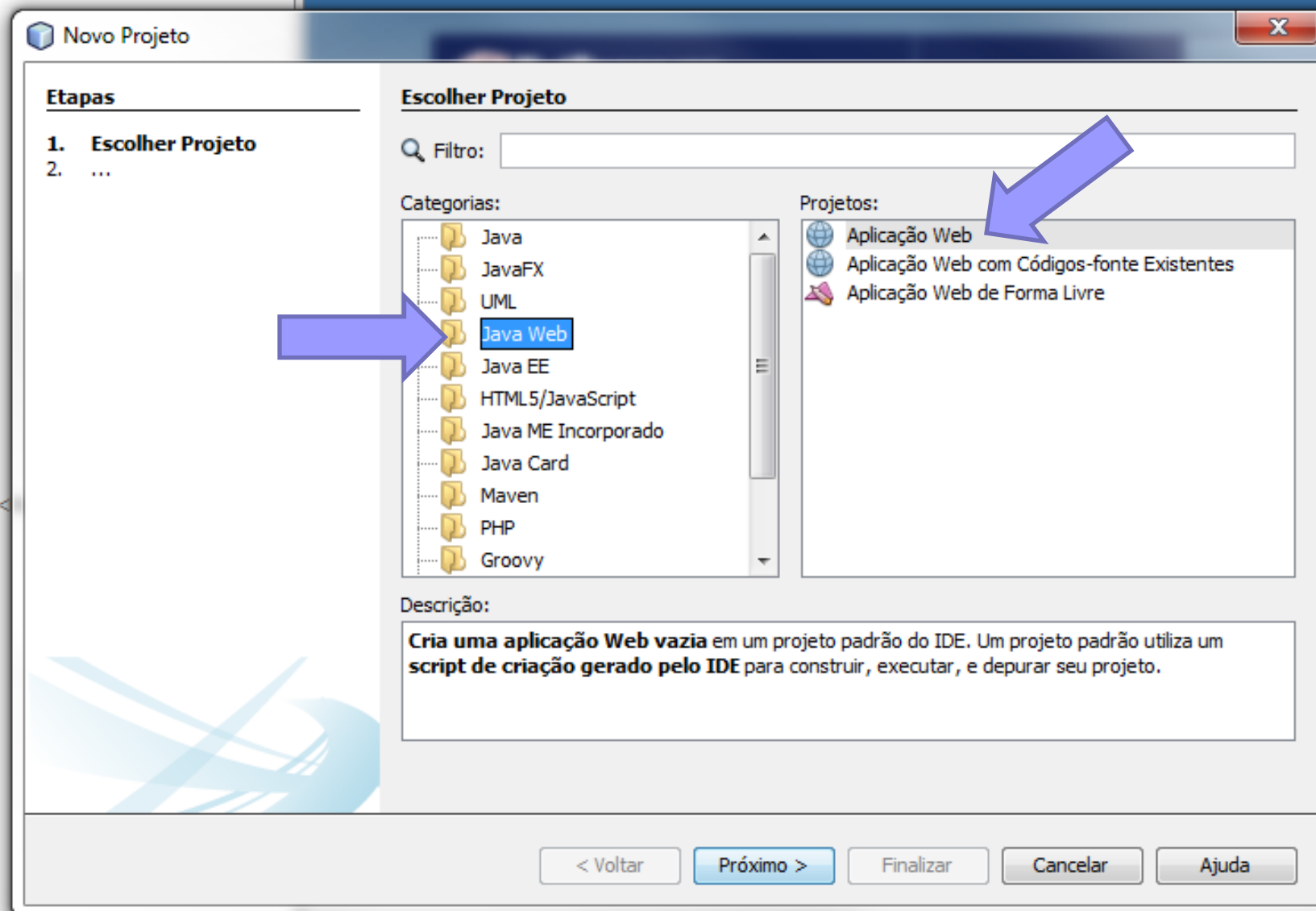
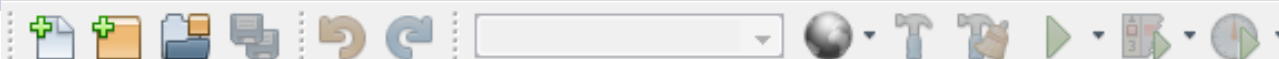
Tecnologias suportadas *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	Tudo
④ SDK da plataforma NetBeans	•	•				•
④ Java SE	•	•				•
④ Java FX	•	•				•
④ Java EE		•				•
④ Java ME						•
④ HTML5/JavaScript		•	•	•		•
④ PHP			•	•		•
④ C/C++					•	•
④ Groovy						•
④ Java Card(tm) 3 Connected						•
Servidores embutidos						
④ GlassFish Server Open Source Edition 4.1.1		•				•
④ Apache Tomcat 8.0.27		•				•

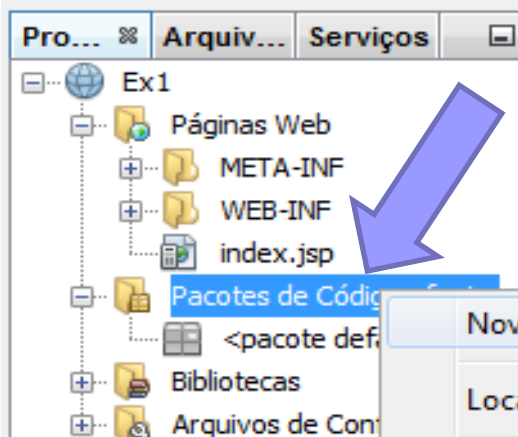
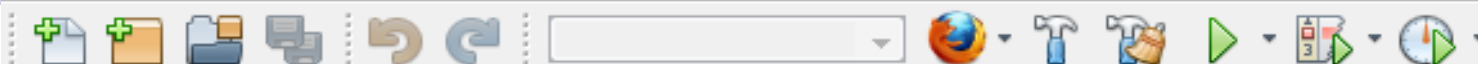


[Download](#)
[Download](#)
[Download x86](#)
[Download x86](#)
[Download x86](#)
[Download](#)

[Download x64](#)
[Download x64](#)
[Download x64](#)







Página Inicial index.jsp

Código-Fonte

Histórico



3

Created on : 05/09/2017, 10:

4

Author : Leo

Novo

Localizar...

Colar Ctrl+V

Histórico

Ferramentas

Propriedades

Pasta...

Interface Java...

Classe Java...

Class Diagram

Servlet...

Pacote Java...

JSP...

Arquivo JSON...

HTML...

Arquivo JavaScript...

Web Services RESTful de Classes de Entidade...

Web Service do WSDL...

Classe da Entidade...

Páginas JSF de Classes de Entidade...

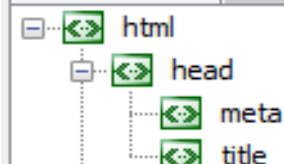
Classes de Entidade do Banco de Dados...

Outros

text/html"

uiv="Conten  
ge</title>

Navegador



Saída



viços

Página Inicial index.jsp

Código-Fonte

New Servlet

### Etapas

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

### Nome e Localização

Nome da Classe: OiMundo

Projeto: Ex1

Localização: Pacotes de Códigos-fonte

Pacote:

Arquivo Criado: C:\Users\Leo\Desktop\DW\fontes\Ex1\src\java\OiMundo.java

⚠ Advertência: é altamente recomendado que você não coloque classes Java no pacote default.

< Voltar

Próximo >

Finalizar

Cancelar

Ajuda

# Servlets

## Exemplo 1 – veja fonte Ex1

- primeiro exemplo **não executa nada de lógica** e apenas mostra uma mensagem estática de bem vindo para o usuário.
- **Escreve código em doGet.**

```
@WebServlet(urlPatterns = {"/OiMundo"})
```

```
public class OiMundo extends HttpServlet {
```

```
    @Override
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse
```

```
        throws ServletException, IOException {
```

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

```
        PrintWriter out = response.getWriter();
```

```
        try {
```

```
            out.println("<!DOCTYPE html>");
```

```
            out.println("<html>");
```

```
            out.println("<head>");
```

```
            out.println("<title>Servlet OiMundo</title>");
```

```
            out.println("</head>");
```

```
            out.println("<body>");
```

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

```
            out.println("</body>");
```

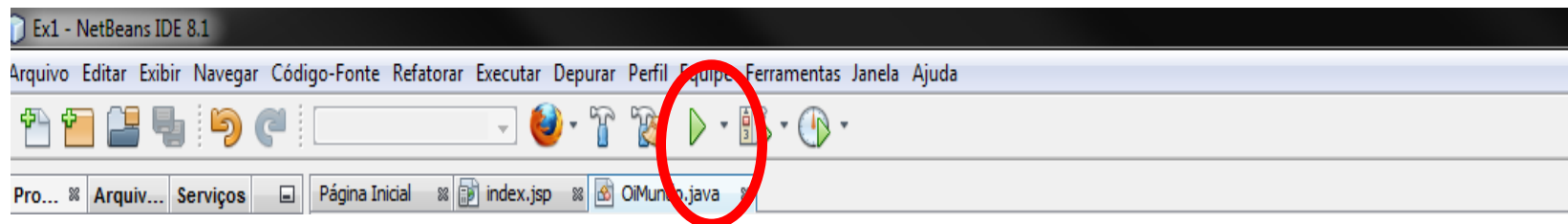
```
            out.println("</html>");
```

```
        } finally {
```

```
            out.close();
```

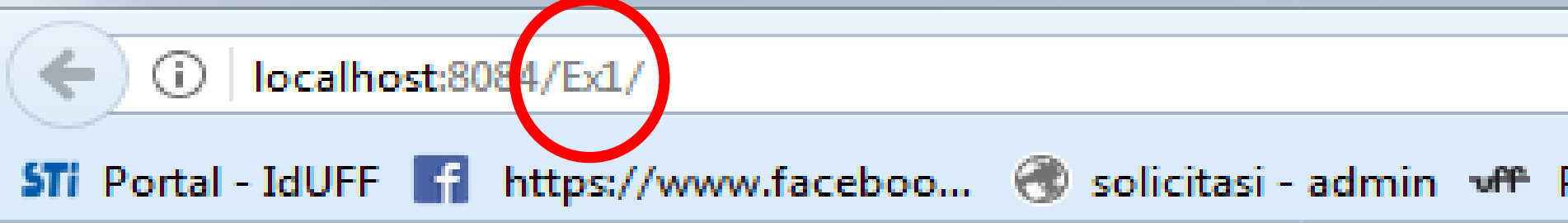
```
        }
```

# Colocar o serviço no ar



```
Ex1 (run) Log Apache Tomcat 7.0.34.0 Apache Tomcat 7.0.34.0
Using CATALINA_BASE: "C:\Users\Leo\AppData\Roaming\NetBeans\8.1\apache-tomcat-7.0.34.0_base"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Apache Tomcat 7.0.34"
Using CATALINA_TMPDIR: "C:\Users\Leo\AppData\Roaming\NetBeans\8.1\apache-tomcat-7.0.34.0_base\temp"
Using JRE_HOME: "C:\Program Files\Java\jdk1.8.0_65"
Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Apache Tomcat 7.0.34\bin\bootstrap.jar;C:\Program Files\Apache
set 09, 2017 9:08:21 AM org.apache.catalina.core.AprLifecycleListener init
INFORMAÇÕES: The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found
set 09, 2017 9:08:22 AM org.apache.coyote.AbstractProtocol init
INFORMAÇÕES: Initializing ProtocolHandler ["http-bio-8084"]
set 09, 2017 9:08:22 AM org.apache.coyote.AbstractProtocol init
INFORMAÇÕES: Initializing ProtocolHandler ["ajp-bio-8009"]
set 09, 2017 9:08:22 AM org.apache.catalina.startup.Catalina load
INFORMAÇÕES: Initialization processed in 682 ms
set 09, 2017 9:08:22 AM org.apache.catalina.core.StandardService startInternal
INFORMAÇÕES: Starting service Catalina
set 09, 2017 9:08:22 AM org.apache.catalina.core.StandardEngine startInternal
INFORMAÇÕES: Starting Servlet Engine: Apache Tomcat/7.0.34
```





# Hello World!



**Serviço no ar!**

**Ops...**

**Não seria Oi Mundo???**

```
@WebServlet(urlPatterns = {"/OiMundo"})
public class OiMundo extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet OiMundo</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Oi Mundo </h1>");
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }
}
```

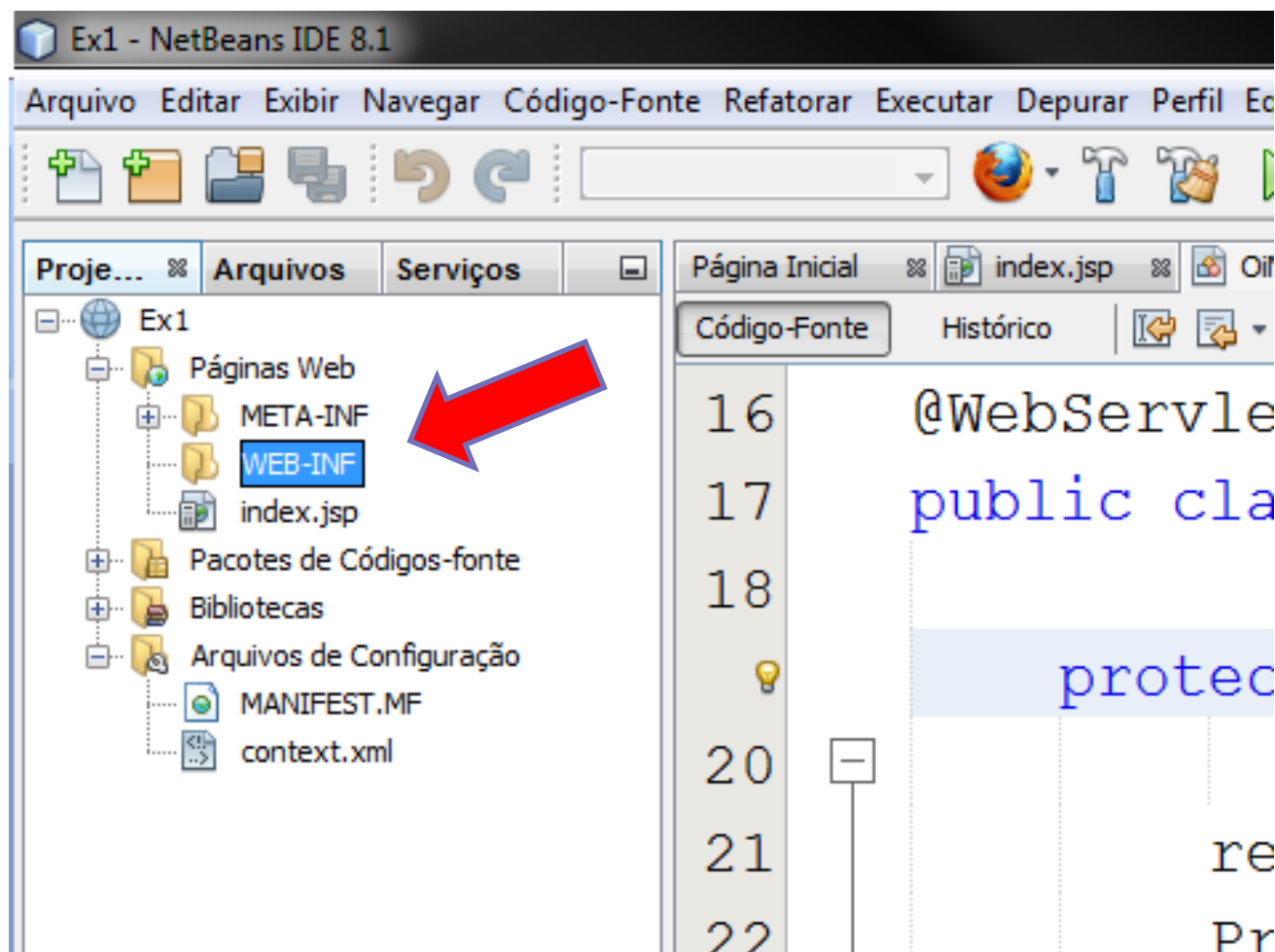
**Olha a url!**

# Oi Mundo

## ■ Requisição Get...

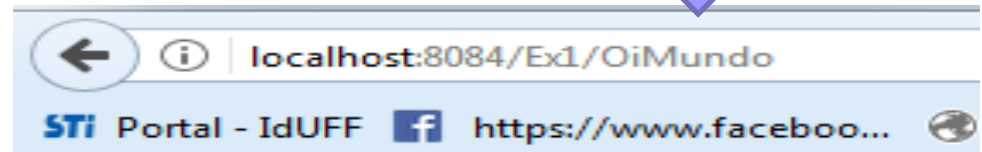
**Veja fonte Ex1**

# Como funcionou? Não configurei o web.xml?



# Servlets

- `@WebServlet("/OiMundo")` - Especificação Servlets 3.0 do Java EE 6.
- Isso é equivalente a configurar a Servlet a com a **url-pattern** configurada como **/OiMundo**.
- não é mais preciso configurar as nossas Servlets no web.xml, sendo suficiente usar a anotação `@WebServlet`



**Oi Mundo**

# Servlets



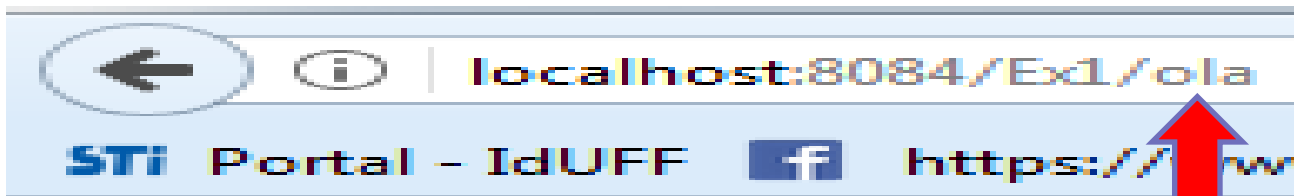
- `@WebServlet(name = "MinhaServlet",  
urlPatterns = {"/OiMundo"})`
- podemos colocar ainda um parâmetro **opcional** chamado `name` que define um nome para a Servlet
- Se não definirmos esse atributo, por padrão, o nome da Servlet é o nome completo da classe da sua Servlet (OiMundo)

# Servlets

- `@WebServlet(urlPatterns = {"/OiMundo",  
"/ola"})`



- Para definir mais de uma URL para acessar a Servlet, podemos utilizar o atributo `urlPatterns` e passar um vetor de URLs



**Oi Mundo**

# Servlets – Erro Comum...

- `@WebServlet(urlPatterns = {"/OiMundo", "/ola"})`



- **Não posso esquecer a barra....**



# Descrevendo o Servlet

- `import java.io.IOException;`
- `import java.io.PrintWriter;`
- `import javax.servlet.ServletException;`
- `import javax.servlet.annotation.WebServlet;`
- `import javax.servlet.http.HttpServlet;`
- `import javax.servlet.http.HttpServletRequest;`
- `import javax.servlet.http.HttpServletResponse;`
  
- Classes usadas no servlet

# Descrevendo o Servlet

```
public class OiMundo extends HttpServlet {
```

- **OiMundo** herda os métodos de **HttpServlet**

```
    public void doGet( HttpServletRequest request,  
                      HttpServletResponse response )  
        throws ServletException, IOException
```

- Method **doGet**
  - Responde a requisição GET

# Descrevendo o Servlet

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

## □ **setContentType**

- Especifica o tipo de conteúdo
- **Esse caso HTML**

- `response.setContentType("text/html");`
- `response.setContentType("text/plain");`
- `response.setContentType("text/css");`
- `response.setContentType("application/html");`
- `response.setContentType("image/gif");`
- `response.setContentType("application/zip");`
- `response.setContentType("application/pdf");`

# Descrevendo o Servlet

```
PrintWriter out = response.getWriter();
```

## □ **getWriter**

- Retorna o objeto **PrintWriter** do response
- Obter um objeto que represente a saída a ser enviada ao usuário através do método `getWriter` da variável `response`

# Descrevendo o Servlet

```
out.println("<!DOCTYPE html>");  
out.println("<html>");  
out.println("<head>");  
out.println("<title>Servlet OiMundo</title>");  
out.println("</head>");  
out.println("<body>");  
out.println("<h1>Oi Mundo </h1>");  
out.println("</body>");  
out.println("</html>");
```

- Mensagem HTML simples para os usuários que a requisitarem.

# Recebendo dados de um Formulário

**Exemplo 2 - Fazendo uma requisição Post**

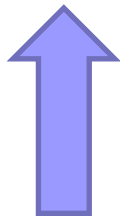
**veja Fonte Ex2**

# Recebendo dados de um Formulário

`<html>` arquivo: **exContatoServlet.html**  
`<body>`  
    `<form method="post"`  
        `action="http://localhost:8084/Ex2/ContatoServlet">`  
        Nome `<br />`  
        `<input type="text" name="nome" /><br />`  
        Endereco`<br />`  
        `<input type="text" name="endereco" /><br />`  
        `<input type="submit" value="Enviar Para o Servlet" />`  
    `</form>`  
    `</body>`  
`</html>`

# Recebendo dados de um Formulário

`<form action=http://localhost:8084/Ex2/ContatoServlet>`



**Servidor: localhost**  
**Porta: 8084**



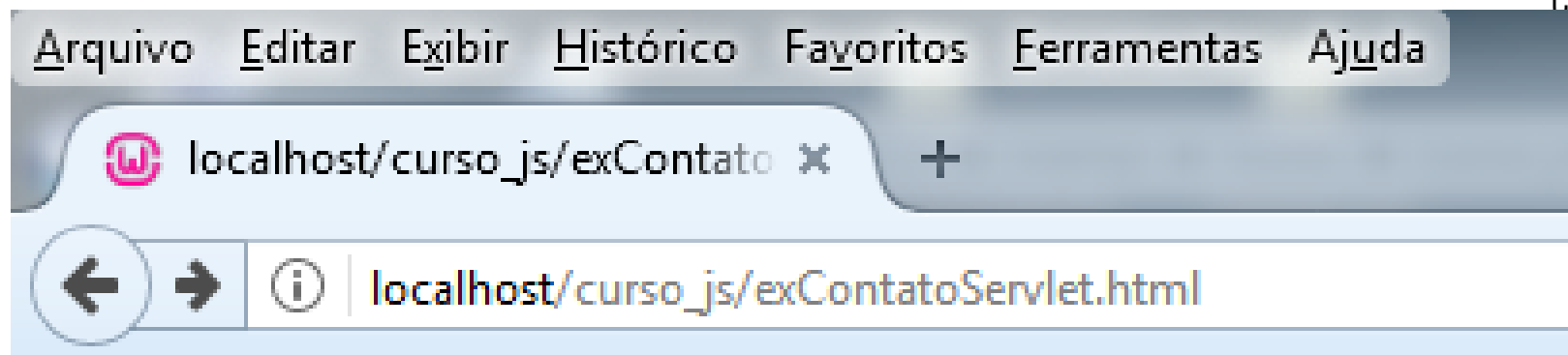
**Quem trata  
a solicitação**

`<form method="post">`



**Tipo de requisição**





Nome

Pedro

Endereco

Rua do lala

Enviar Para o Servlet

## Requisição Post

```
@WebServlet(urlPatterns = {"/ContatoServlet"})  
public class ContatoServlet extends HttpServlet {  
    protected void processRequest(HttpServletRequest request, HttpServletResponse  
        throws ServletException, IOException { response.setContentType  
        PrintWriter out = response.getWriter();  
        try {  
            // pegando os parâmetros do request  
            String nome = request.getParameter("nome");  
            String endereco = request.getParameter("endereco");  
            // gerando a resposta  
            out.println("<!DOCTYPE html>");  
            out.println("<html>");  
            out.println("<head>");  
            out.println("<title>Servlet ContatoServlet</title>");  
            out.println("</head>");  
            out.println("<body>");  
            out.println("<h1>Contato </h1>");  
            out.println("<p>Nome: "+nome+"</p>");  
            out.println("<p>Endereço: "+endereco+"</p>");  
            out.println("</body>");
```

```
@WebServlet(urlPatterns = {"/ContatoServlet"})
```



```
public class ContatoServlet extends HttpServlet {  
    protected void processRequest(HttpServletRequest request, HttpServletResponse  
        throws ServletException, IOException { response.setContentType  
        PrintWriter out = response.getWriter();  
        try {  
            // pegando os parâmetros do request  
            String nome = request.getParameter("nome");  
            String endereco = request.getParameter("endereco");  
            // gerando a resposta
```

```
<html>  
<body>  
    <form action="http://localhost:8084/Ex2/ContatoServlet">  
        Nome <br />  
        <input type="text" name="nome" /><br />  
        Endereco<br />  
        <input type="text" name="endereco" /><br />  
        <input type="submit" value="Enviar Para o Servet" />
```

```
        out.println("<hr>Contato </hr>");  
        out.println("<p>Nome: "+nome+"</p>");  
        out.println("<p>Endereço: "+endereco+"</p>");  
        out.println("</body>");
```

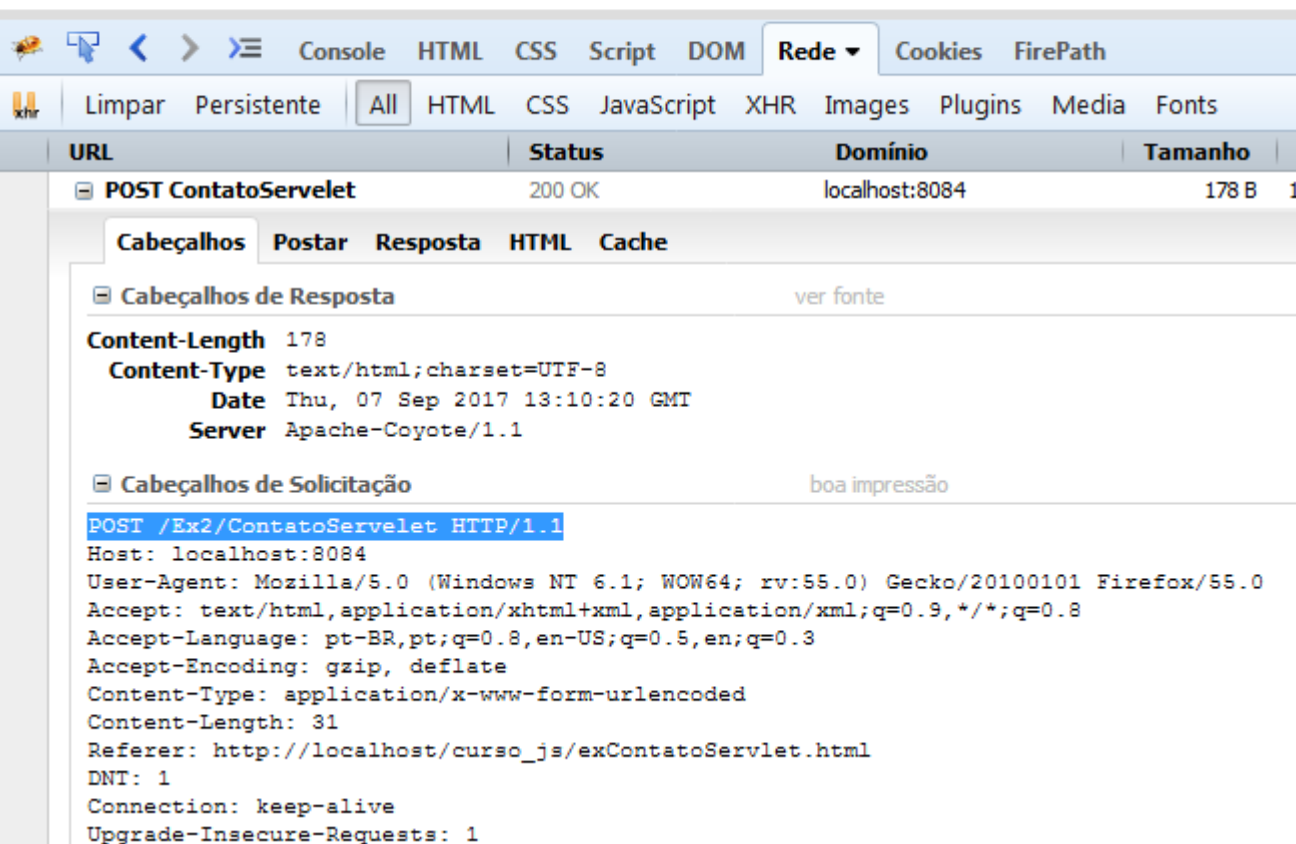


# Contato

Nome: Pedro

Endereço: Rua do lala

# Resposta do servlet



The screenshot shows the Firefox Developer Tools network tab. A POST request to 'ContatoServlet' is selected, showing a 200 OK status. The 'Cabeçalhos de Resposta' (Response Headers) section is expanded, displaying the following information:

- Content-Length:** 178
- Content-Type:** text/html; charset=UTF-8
- Date:** Thu, 07 Sep 2017 13:10:20 GMT
- Server:** Apache-Coyote/1.1

The 'Cabeçalhos de Solicitação' (Request Headers) section is also expanded, showing the following information:

- POST /Ex2/ContatoServlet HTTP/1.1**
- Host:** localhost:8084
- User-Agent:** Mozilla/5.0 (Windows NT 6.1; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0
- Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
- Accept-Language:** pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
- Accept-Encoding:** gzip, deflate
- Content-Type:** application/x-www-form-urlencoded
- Content-Length:** 31
- Referer:** http://localhost/curso\_js/exContatoServlet.html
- DNT:** 1
- Connection:** keep-alive
- Upgrade-Insecure-Requests:** 1

# Recebendo dados de um Formulário

- Através do método `getParameter("campo")` da classe `HttpServletRequest` podemos obter o valor de um determinado campo de formulário;
- No parâmetro desta função, inserimos **o mesmo nome de campo utilizado no formulário HTML** que submeteu os dados;
- O valor retornado por esta função será um dado do tipo **String**.
- Para atribuir tal valor a uma variável de outro tipo, devemos efetuar uma conversão;

# Tipo de Requisição

- Os requests podem ser de dois tipos:
- **POST** → Função é enviar dados para o servidor
- **GET** → Função é requisitar dados do servidor
  - POST: usualmente por forms
  - GET: usualmente por links/barra de url

localhost:8084/Ex2/ContatoServlet?nome="leo"&amp;endereco="rua lala"

# Contato

Nome: "leo"

Endereço: "rua lala"

The screenshot shows a web browser window with a tab titled 'Servlet ContatoServlet'. The address bar displays the URL 'localhost:8084/Ex2/ContatoServlet?nome="leo"&endereco="rua lala"'. A red arrow points from the browser's address bar to the 'Cabeçalhos de Solicitação' (Request Headers) section in the developer tools. The page content shows a form with the name 'leo' and address 'rua lala'. The developer tools show a GET request to the same URL with a 200 OK status. The request headers are expanded, showing the following details:

URL	Status	Domínio
GET ContatoServlet?nome=%22leo%22&endereco=%22rua%20lala%22	200 OK	localhost:8084

Below the table, the 'Cabeçalhos de Solicitação' (Request Headers) are listed:

```
GET /Ex2/ContatoServlet?nome=%22leo%22&endereco=%22rua%20lala%22 HTTP/1.1
Host: localhost:8084
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

A red arrow points to the 'User-Agent' header. The bottom of the developer tools shows '1 requisição' (1 request).

# Contato

Nome: "leo"

Endereço: "rua lala"



## Requisição com GET

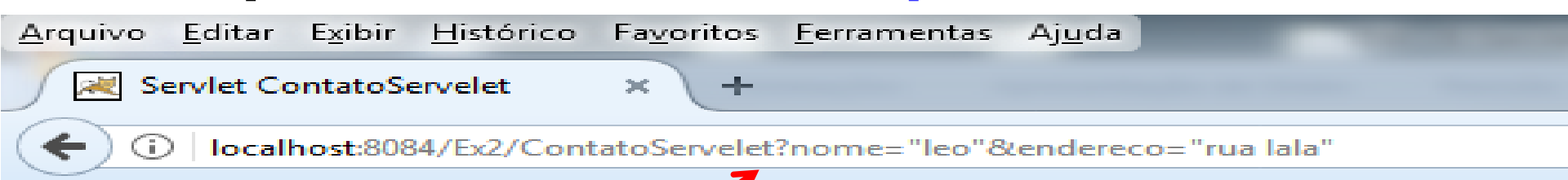


# Requisição com GET

- Podemos passar alguns **parâmetros pela** requisição do tipo GET por meio do seguinte esquema:
- `http://servidor/servlet?param1=valor1`
- O “macete” é a interrogação: **?**
- Esse caractere indica que:
  - ☐ o endereço **já acabou**
  - ☐ tudo que vem em seguida é parâmetro

# Requisição com GET

■ `http://servidor/servlet?param1=valor1`



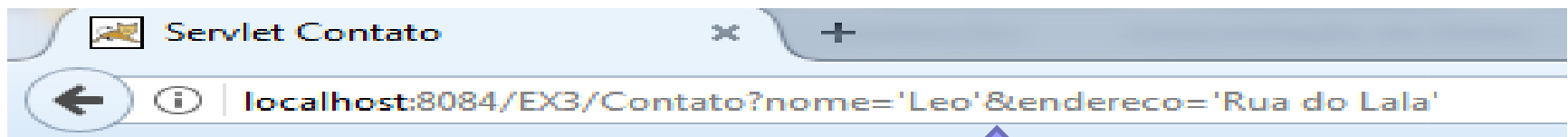
- Depois da ?
- Nome do parâmetro (no exemplo, **param1**)
  - Sinal de igualdade
  - Valor do parâmetro (no exemplo, **valor1**)

# Requisição com GET

- E se quiser passar mais de um parâmetro?

basta separá-los com o uso de um &

- `http://servidor/servlet?param1=valor1&param2=valor2`

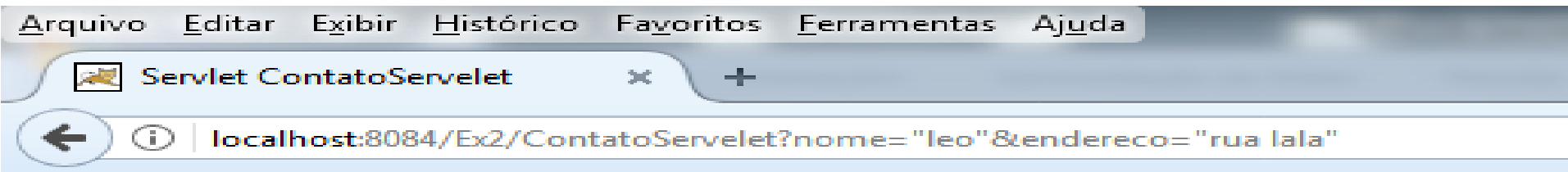


# Requisição com GET

- Podemos passar quantos parâmetros quisermos?
- **NÃO com o GET tem limitação com relação ao número de caracteres**

# Observação

- Usamos o mesmo servlet para processar uma requisição **Get** e uma requisição **Post**



<form action=http://localhost:8084/Ex2/**ContatoServlet**>

Nome

Endereco

Do ponto de vista do  
**servlet, o que muda?**

# Observação

- Usando NetBeans, **NADA**, os dados chegam, com o **GET**, da mesma forma que com o **POST**
- Quer dizer que não temos como diferenciar um do outro no **servlet**?

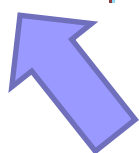
# Observação

```
// <editor-fold defaultstate="collapsed" desc="HttpServletRequest methods. Click on th
```

```
/**...*/
```

```
@Override
```

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



**Redireciona para processRequest**

```
/**...*/
```

```
@Override
```

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



**Redireciona para processRequest**

```
/**...*/
```

```
@Override
```

```
public String getServletInfo() {
    return "Short description";
}
```

# Rejeitando GET

- E se não quisermos que nosso **servlet** responda com requisições GET?
  - Podemos ir até o método **doGet()** e simplesmente remover a chamada ao **processRequest()**
  - Um outro jeito é encaminhando o usuário para uma página de erro específica...
  - Ou simplesmente para a tela correta de preenchimento!



# Rejeitando GET

- E se não quisermos que nosso **servlet** responda com requisições GET?

Retirando a chamada **processRequest** do **doGet**

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

# Rejeitando GET

- E se não quisermos que nosso **servlet** responda com requisições GET?
  - Excluir o método **processRequest** e escrever o método específico para **doGet** e **doPost** especificamente

# Rejeitando GET

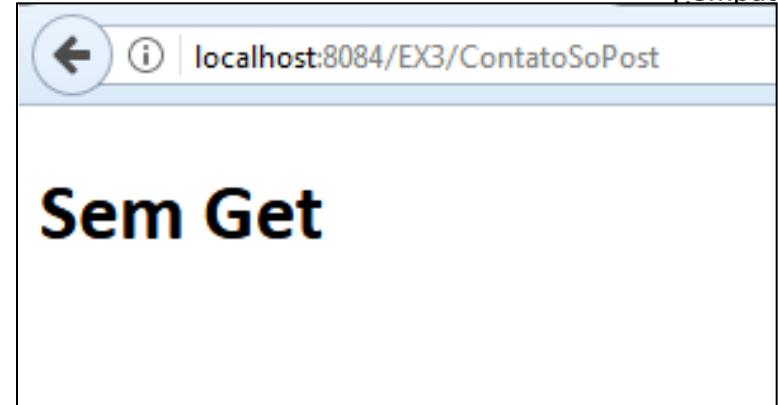
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException { response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    try {
        // pegando os parâmetros do request
        String nome = request.getParameter("nome");
        String endereco = request.getParameter("endereco");

        // gerando a resposta
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet ContatoServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Contato </h1>");
        out.println("<p>Nome: "+nome+"</p>");
        out.println("<p>Endereço: "+endereco+"</p>");
        out.println("</body>");
        out.println("</html>");
    } catch (Exception e) {
        // tratamento de exceção
    }
}
```

**Retirar processRequest**

# Rejeitando GET

## doGet específico



```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
    throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    try {
        // gerando a resposta
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet ContatoServelet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Sem Get </h1>");
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

Vide fonte ex3

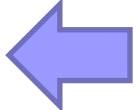
# Rejeitando GET

## doPost específico

```
protected void doPost(HttpServletRequest request, HttpServletResponse resp
    throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    try {
        // pegando os parâmetros do request
        String nome = request.getParameter("nome");
        String endereco = request.getParameter("endereco");
        // gerando a resposta
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet ContatoServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Contato </h1>");
        out.println("<p>Nome: "+nome+"</p>");
        out.println("<p>Endereço: "+endereco+"</p>");
        out.println("</body>");
        out.println("</html>");
    } finally {
```

# **Servlets - Mais um exemplo**

- **Servlets + POO...**
- **Vide fonte ex4**

public class **ContatoPessoal** {  **A Classe ContatoPessoal**

```
    private String Nome;  
    private String Endereco;
```

```
    public ContatoPessoal(String Nome, String Endereco) {  
        this.Nome = Nome;  
        this.Endereco = Endereco;    }
```

```
    public String getNome() { return Nome; }  
    public void setNome(String Nome) { this.Nome = Nome; }  
    public String getEndereco() { return Endereco; }  
    public void setEndereco(String Endereco) { this.Endereco =  
        Endereco;  
    }
```

```
public class Contato extends HttpServlet {  
    protected void processRequest(HttpServletRequest request, HttpServletResponse  
        throws ServletException, IOException { response.setContentType("text/html");  
    try {  
        // pegando os parâmetros do request  
        String nome = request.getParameter("nome");  
        String endereco = request.getParameter("endereco");  
        // criação do objeto ContatoPessoal  
        ContatoPessoal meuContato = new ContatoPessoal(nome, endereco);  
  
        out.println("<!DOCTYPE html>");  
        out.println("<html>");  
        out.println("<head>");  
        out.println("<title>Servlet Contato</title>");  
        out.println("</head>");  
        out.println("<body>");  
        out.println("<h1>Contato</h1>");  
        out.println("<p>Nome:" + meuContato.getNome() + "</p>");  
        out.println("<p>Nome:" + meuContato.getEndereco() + "</p>");  
        out.println("</body>");  
    }  
}
```

