

Laboratório de Programação de Web Sites Dinâmicos

Servlets

Tassio Sirqueira – 2019/02

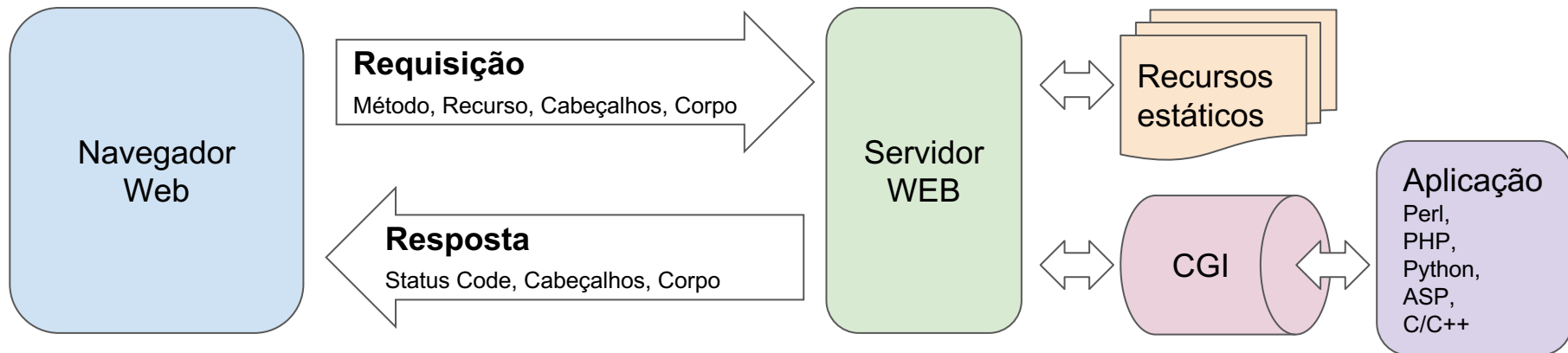
| Servlets | Introdução

- ❑ Quando a web surgiu, o objetivo era realizar a troca de documentos estáticos por meio de HTTP.
 - ❑ Páginas HTML
 - ❑ Imagens
 - ❑ Animações...
- ❑ Logo observou-se o potencial de comunicação da Web.
 - ❑ Páginas estáticas não seriam o suficiente.
- ❑ Era necessário responder de forma dinâmica às requisições dos usuários.
- ❑ Hoje boa parte do que acessamos na Web é baseado em conteúdos dinâmicos.
 - ❑ Portais de notícias
 - ❑ Blogs
 - ❑ Bancos...



| Servlets | Introdução

- ❑ A primeira forma de se gerar páginas dinâmicas foi por meio do CGI.
 - ❑ CGI - Common Gateway Interface
- ❑ Interface padrão a partir da qual o servidor Web executaria um programa externo.
 - ❑ Tipicamente PERL
 - ❑ Outras linguagens começaram a surgir/suportar o padrão, como PHP, ASP, C/C++...



| Servlets | Introdução



- ❑ Na plataforma Java, a primeira e principal tecnologia capaz de gerar páginas dinâmicas são as **Servlets**.
 - ❑ Surgiram no ano de 1997
 - ❑ As versões mais encontradas no mercado são a 2.4 (J2EE 1.4) e a 2.5 (Java EE 5).
 - ❑ A versão atual é a 3.1 (Java EE 7).
- ❑ Usaremos a própria linguagem Java para isso, criando uma classe que terá capacidade de gerar conteúdo HTML.
- ❑ O nome "servlet" vem da ideia de um pequeno servidor (servidorzinho, em inglês) .

| Servlets | Introdução



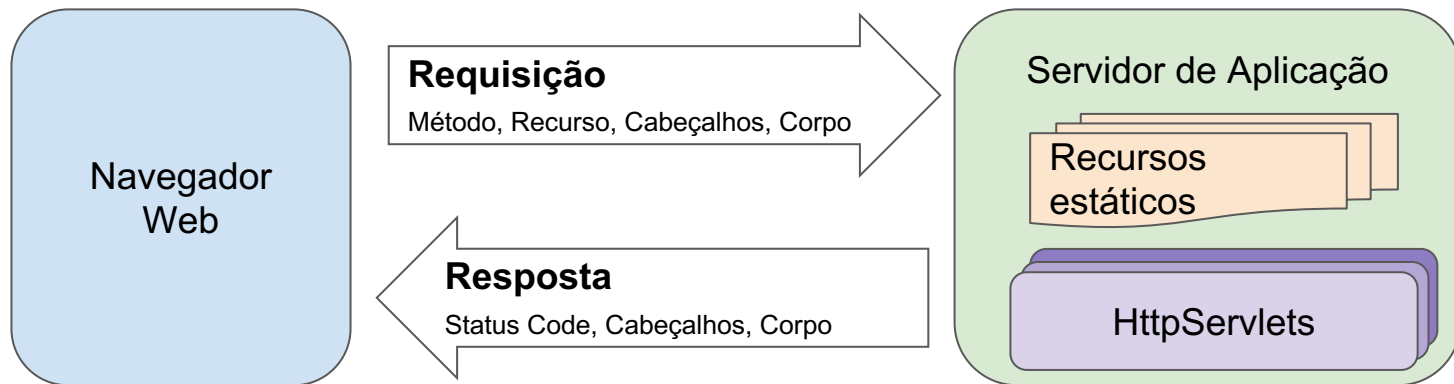
❑ Servlets x CGI

- ❑ Diversas requisições podem ser feitas à mesma servlet ao mesmo tempo em um único servidor.
- ❑ Fica na memória entre requisições, não precisa ser re-instanciada.
- ❑ O nível de segurança e permissão de acesso pode ser controlado em Java.
- ❑ Em CGI, cada cliente é representado por um processo, enquanto que com Servlets, cada cliente é representado por uma linha de execução (*Thread*).

| Servlets | Introdução



- ❑ Servlets são classes Java que implementam uma interface específica para tratar requisições e respostas.
- ❑ **HttpServlet**



| Servlets | Deployment Descriptor

- ❑ Para que possa tratar as requisições HTTP precisamos “dizer” ao servidor web quais são os Servlets do nosso projeto.
- ❑ Para isso utilizamos um arquivo específico o **web.xml**
 - ❑ web.xml é o ***Deployment Descriptor*** (Descritor de implantação).
- ❑ Esse arquivo deve ficar em uma pasta especial chamada **WEB-INF** dentro da pasta de recursos do projeto.
 - ❑ Nada dentro de META-INF será acessível via HTTP.

```
web.xml

<?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>olamundo</servlet-name>
    <servlet-
class>com.exemplo.controller.OlaMundo</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>olamundo</servlet-name>
    <url-pattern>/ola-mundo</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

| Servlets | Introdução



web.xml

```
<?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>olamundo</servlet-name>
    <servlet-
class>com.exemplo.controller.OlaMundo</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>olamundo</servlet-name>
    <url-pattern>/ola-mundo</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

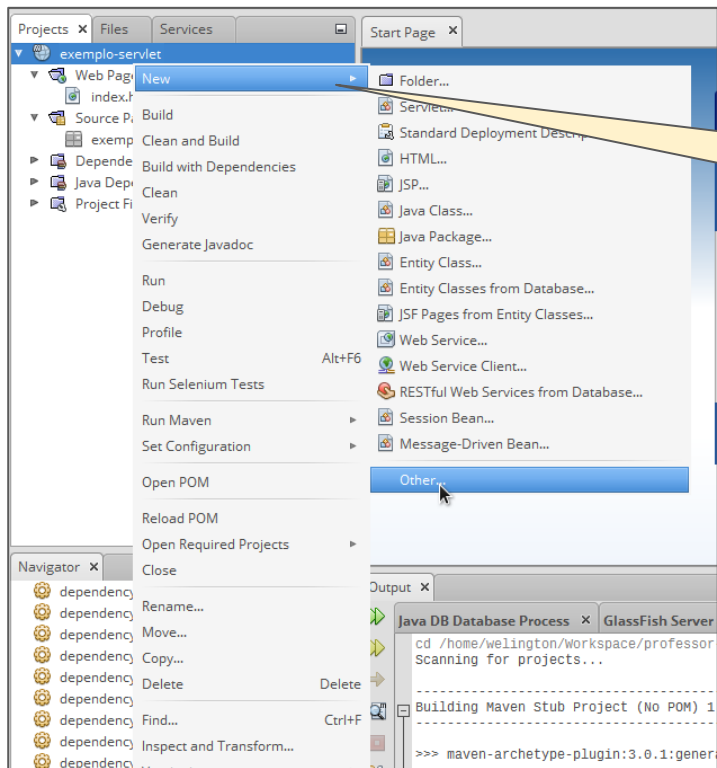
Qual o nome do Servlet
Qual classe implementa o Servlet indicado.

Mapeamos um recurso (URI) a
Um servlet declado nesse arquivo.

Existem uma série de configurações possíveis nesse arquivo.
Entre elas tempo de sessão, bibliotecas de tags customizadas,
Páginas de erro personalizadas, segurança, etc...

| Servlets | Deployment Descriptor | Netbeans

❑ Criando um **web.xml** no Netbeans.

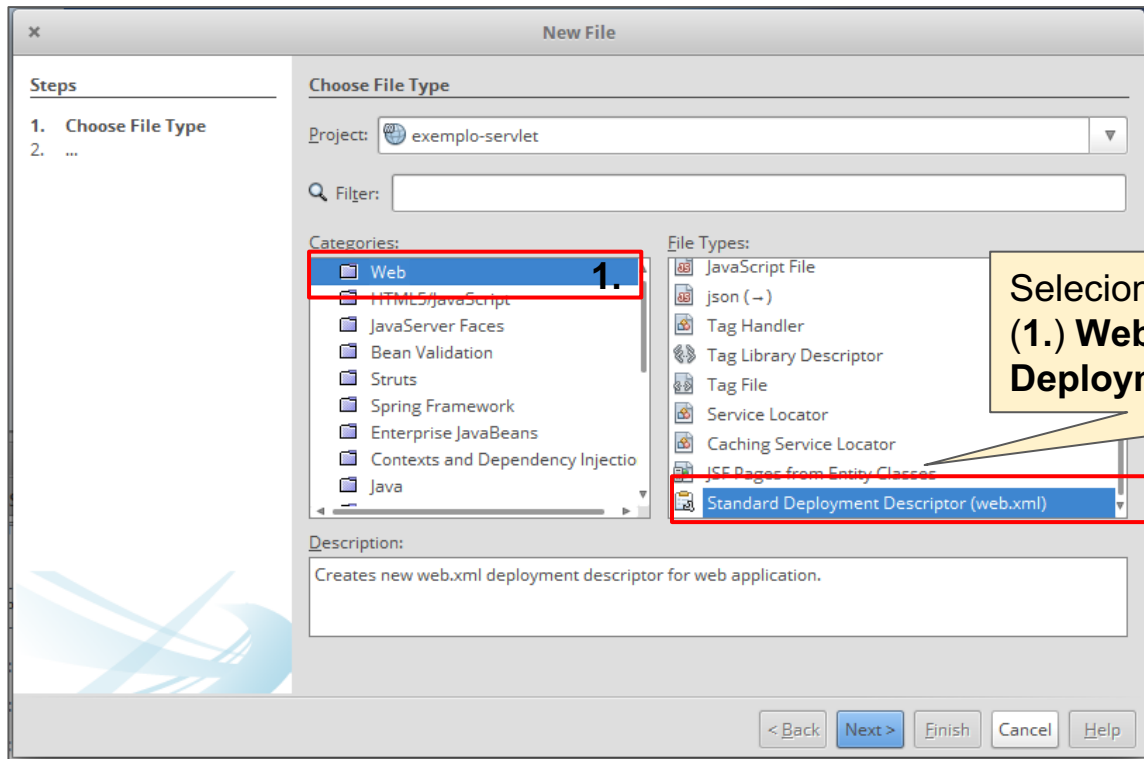


Clique com o botão direito no nome do projeto.

Selecione **New > Other...**

| Servlets | Deployment Descriptor | Netbeans

❑ Criando um **web.xml** no Netbeans.



Selecione, na escolha do arquivo, a categoria (1.) **Web** e o tipo de arquivo (2.) **Standard Deployment Descriptor (web.xml)**

| Servlets | Deployment Descriptor | Netbeans



❑ Criando um **web.xml** no Netbeans.

A screenshot of the 'New Standard Deployment Descriptor (web.xml)' dialog box in NetBeans. The dialog has a 'Steps' panel on the left with two steps: '1. Choose File Type' and '2. Name and Location'. The 'Name and Location' panel on the right contains four text fields: 'File Name:' with 'web.xml', 'Project:' with 'exemplo-servlet', 'Location:' with '/home/welington/Workspace/professor-netbeans/exemplo-servlet/src/main/webapp/WEB-INF', and 'Created File:' with '/home/welington/Workspace/professor-netbeans/exemplo-servlet/src/main/webapp/WEB-INF/web.xml'. The 'Location' and 'Created File' fields are highlighted with red boxes and numbered '1.' and '2.' respectively.

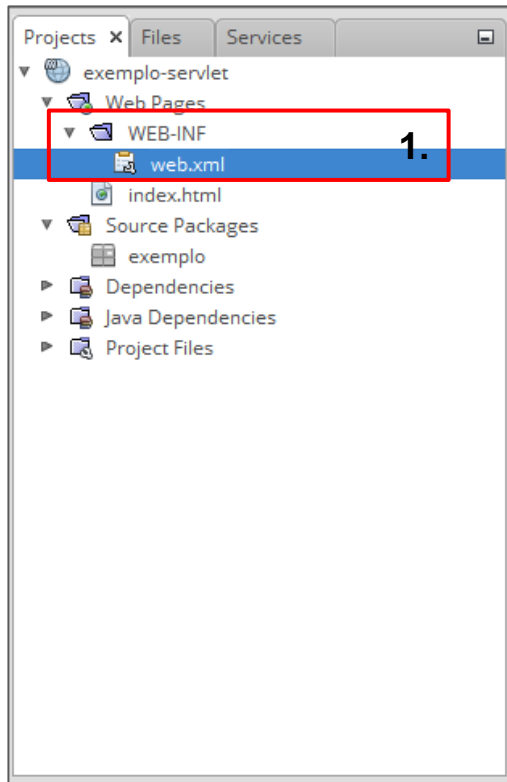
Por padrão, o próprio Netbeans irá indica o local correto para que os arquivos sejam exibidos.

Apenas verifique.

1. A pasta destino deve ser a **WEB-INF** dentro de **webapp** (para um projeto maven)
2. O arquivo deve ser **web.xml**

| Servlets | Deployment Descriptor | Netbeans

❏ Criando um **web.xml** no Netbeans.



web.xml

```
<?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">
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

O arquivo gerado irá exibir apenas uma configuração, com o tempo padrão de sessão.

Vamos discutir essa configuração adiante.

| Servlets | Implementando HttpServlet



- ❑ Para implementar um HttpServlet, podemos sobreescrever métodos importantes:

- ❑ **doGet**

- ❑ Recebe uma requisição e uma resposta para um requisição GET.

- ❑ **doPost**

- ❑ Recebe uma requisição e uma resposta para uma requisição POST.

- ❑ Precisamos escrever o conteúdo dinâmico na resposta.

Contador.java

```
public class Contador extends HttpServlet {

    private int count = 0;

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

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("    <head>");
            out.println("        <title>Servlet Contador</title>");
            out.println("    </head>");
            out.println("    <body>");
            out.println("        <h1>Requisição número " + (++count) + "</h1>");
            out.println("    </body>");
            out.println("</html>");

        }

    }

}
```

| Servlets | Implementando HttpServlet



web.xml

```
public class Contador extends HttpServlet {
```

```
    private int count = 0;
```

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

```
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("    <head>");
            out.println("        <title>Servlet Contador</title>");
            out.println("    </head>");
            out.println("    <body>");
            out.println("        <h1>Requisição número " + (++count) + "</h1>");
            out.println("    </body>");
            out.println("</html>");
        }
    }
```

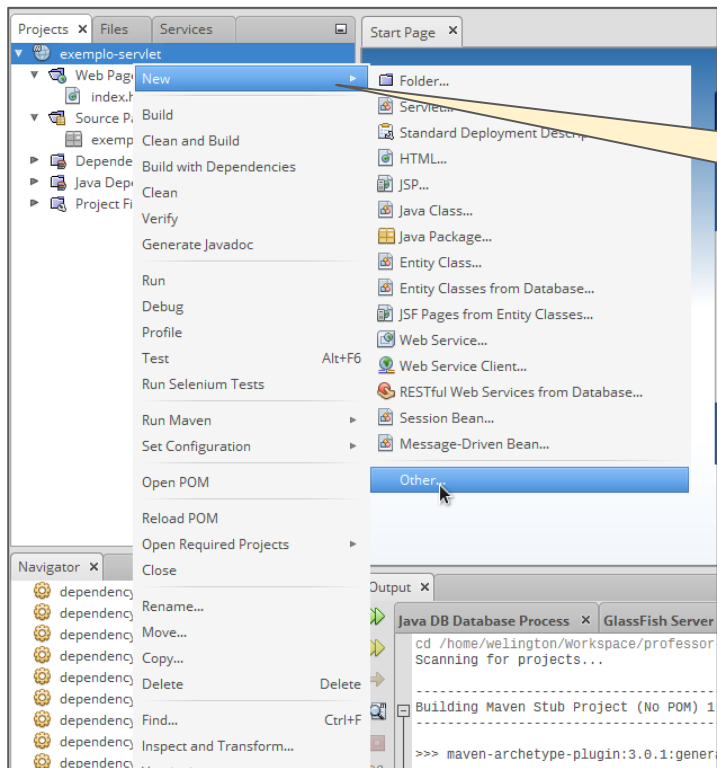
Precisamos herdar de **HttpServlet**.

O conteúdo, status code e cabeçalhos da resposta pode ser escrito no objeto response.

Sempre que recebermos uma requisição **GET** o código do método **doGet** será executado.

| Servlets | Implementando HTTPServlet | Netbeans

❑ Criando um Servlet no Netbeans.



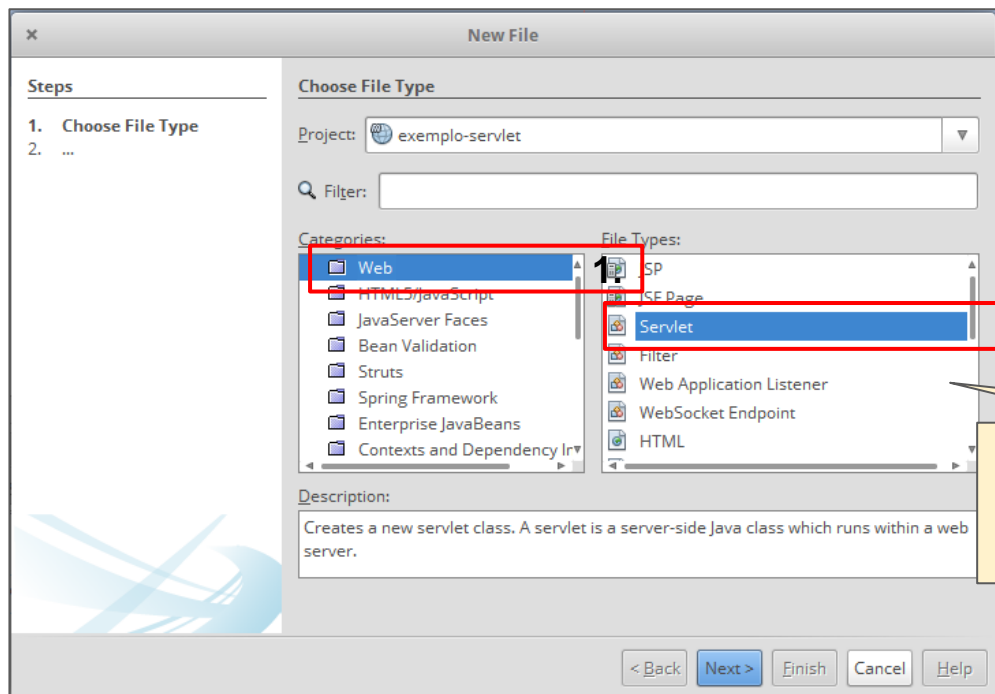
Clique com o botão direito no nome do projeto.

Selecione **New > Other...**

| Servlets | Implementando HTTPServlet | Netbeans



❑ Criando um **Servlet** no Netbeans.



Selecione, na escolha do arquivo, a categoria (1.) **Web** e o tipo de arquivo (2.) **Servlet**

| Servlets | Implementando HTTPServlet | Netbeans



❑ Criando um Servlet no Netbeans.

The image shows the "New Servlet" dialog box in the NetBeans IDE. The dialog is titled "New Servlet" and has a "Steps" panel on the left with three steps: "1. Choose File Type", "2. Name and Location", and "3. Configure Servlet Deployment". The "Name and Location" step is currently selected. The "Class Name" field is set to "Contador" and is highlighted with a red box and the number "1.". The "Project" field is set to "exemplo-servlet". The "Location" field is set to "Source Packages". The "Package" field is set to "exemplo" and is highlighted with a red box and the number "2.". The "Created File" field shows the path "rofessor-netbeans/exemplo-servlet/src/main/java/exemplo/Contador.java". At the bottom of the dialog are buttons for "< Back", "Next >", "Finish", "Cancel", and "Help".

1. Defina o nome da classe (ex: **Contador**).
]
2. Selecione o pacote em que o Servlet será criado.

| Servlets | Implementando HTTPServlet | Netbeans



❑ Criando um Servlet no Netbeans.

The screenshot shows the 'New Servlet' dialog box in NetBeans, specifically the 'Configure Servlet Deployment' step. The 'Steps' panel on the left lists three steps: '1. Choose File Type', '2. Name and Location', and '3. Configure Servlet Deployment'. The main area contains the following fields and options:

- ☒ Add information to deployment descriptor (web.xml) (labeled 1.)
- Class Name: exemplo.Contador
- Servlet Name: Contador (labeled 2.)
- URL Pattern(s): /contador (labeled 3.)
- Initialization Parameters: A table with columns 'Name' and 'Value', and a 'New' button.
- Buttons: < Back, Next >, Finish

1. Marque para que o servlet seja adicionado automaticamente ao **web.xml**.
2. Defina o nome do Servlet.
3. Defina a URL para acessar ao servlet.

| Servlets | Implementando HTTPServlet | Netbeans



❏ Criando um Servlet no Netbeans.

1. A classe que implementa o Servlet será criada no pacote de código fonte.

2. O mapeamento, conforme definido, será adicionado automaticamente ao **web.xml**.

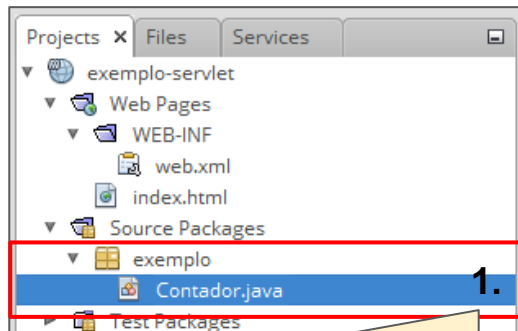
```
web.xml<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">

  <servlet>
    <servlet-name>Contador</servlet-name>
    <servlet-class>exemplo.Contador</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Contador</servlet-name>
    <url-pattern>/contador</url-pattern>
  </servlet-mapping>
</web-app>
```

| Servlets | Implementando HTTPServlet | Netbeans



❑ Criando um Servlet no Netbeans.



1. A classe que implementa o Servlet será criada no pacote de código fonte.
2. Os métodos **doGet** e **doPost** são automaticamente implementados.
3. O método **processRequest** gerado imprime uma mensagem padrão na tela.

```
web.xml

public class Contador extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Contador</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Contador at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

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

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

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

3.

2.

| Servlets | Implementando HTTPServlet | Netbeans



❑ Criando um **Servlet** no Netbeans.



http://localhost:8080/exemplo-servlet/contador		
Onde o Application Server está rodando	Contexto da Aplicação	Url pattern do Servlet



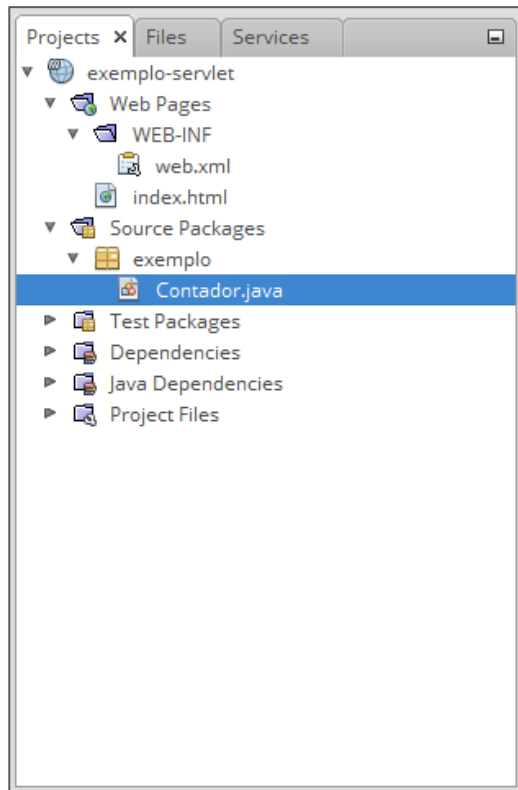
| Servlets | Implementando HttpServlet | Netbeans



| Servlets | Implementando HttpServlet | Netbeans



❏ Implementando o Servlet Contador.



Contador.java

```
public class Contador extends HttpServlet {

    private int count = 0;

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

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("    <head>");
            out.println("        <title>Servlet Contador</title>");
            out.println("    </head>");
            out.println("    <body>");
            out.println("        <h1>Requisição número " + (++count) + "</h1>");
            out.println("    </body>");
            out.println("</html>");
        }
    }
}
```

API completa: <http://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html>

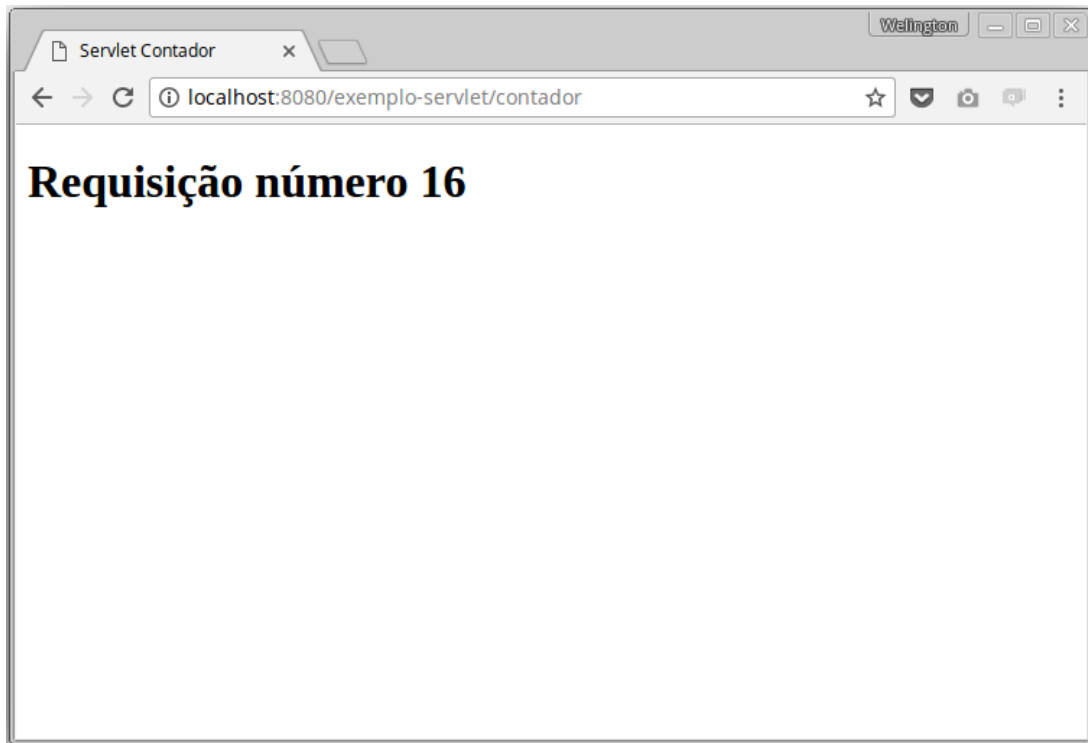
| Servlets | Implementando HTTPServlet | Netbeans



❑ Implementando o Servlet Contador.



http://localhost:8080/exemplo-servlet/contador



| Servlets | Implementando HTTPServlet | Netbeans



❑ Requisição/Resposta no Servlet Contador.



Requisição

```
GET /exemplo-servlet/contador HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/59.0.3071.115 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: pt,en-US;q=0.8,en;q=0.6
```

```
HTTP/1.1 200 OK
Server: GlassFish Server Open Source Edition
4.1.1
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish
Server Open Source Edition 4.1.1 Java/Oracle
Corporation/1.8)
Content-Type: text/html;charset=UTF-8
Date: Sun, 13 Aug 2017 22:22:17 GMT
Content-Length: 151
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Servlet Contador</title>
  </head>
  <body>
    <h1>Requisição número 17</h1>
  </body>
</html>
```

Resposta



| Servlets | HttpServletRequest | Principais métodos

❑ `getContextPath()`

- ❑ Parte da URL que corresponde à requisição realizada.

❑ `getHeader(String name)`

- ❑ Obtém um cabeçalho pelo Nome do mesmo.

❑ `getHeaderNames()`

- ❑ Retorna um enumerado com os nomes dos cabeçalhos.

❑ `getMethod()`

- ❑ Retorno o método HTTP da requisição.

❑ `getRequestURI()`

- ❑ Informações do caminho enviados pelo cliente na requisição.

❑ `getQueryString()`

- ❑ Parâmetros de consulta enviados na requisição.

Lendo o corpo de uma requisição POST

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

    StringBuilder sb = new StringBuilder();
    String line = null;
    try {
        BufferedReader reader = request.getReader();
        while ((line = reader.readLine()) != null)
            sb.append(line);
    } catch (Exception e) {
        /*report an error*/
    }
    System.out.println(sb.toString());
}
```

| Servlets | HttpServletResponse | Principais métodos

- ❑ **addHeader(String name, String value)**
 - ❑ Adiciona um cabeçalho à resposta.
- ❑ **setHeader(String name, String value)**
 - ❑ Define o valor de um cabeçalho.
- ❑ **setStatus(int sc)**
 - ❑ Define o código de status da resposta.
- ❑ **sendError(int sc)**
 - ❑ Limpa o corpo da mensagem e envia um código de erro para o cliente.
- ❑ **sendRedirect(String location)**
 - ❑ Redireciona a requisição para outro recurso.
- ❑ **setContentType(String type)**
 - ❑ Define o conteúdo da resposta.

Escrevendo o corpo de uma resposta GET/POST

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

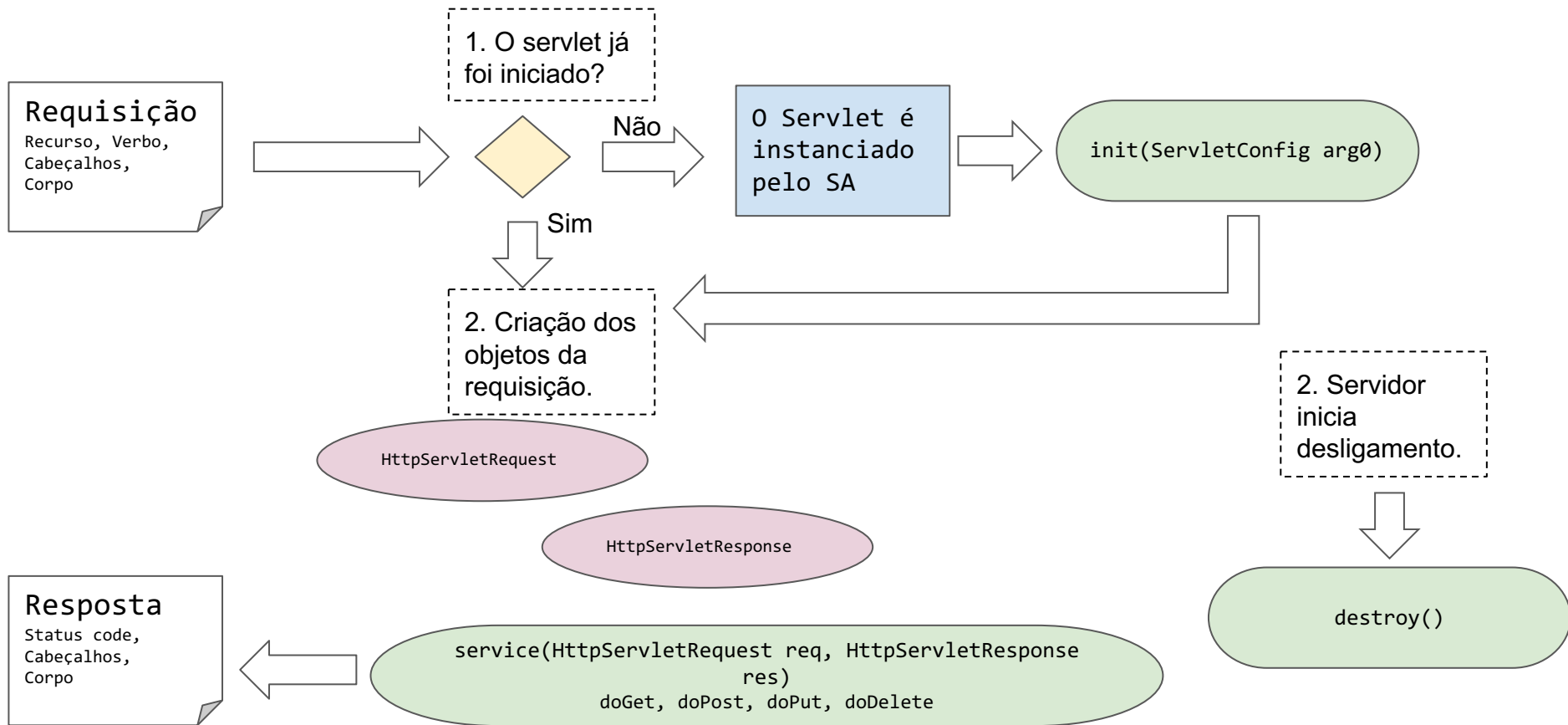
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code.
        */

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("    <head>");
        out.println("        <title>Servlet Contador</title>");
        out.println("    </head>");
        out.println("    <body>");
        out.println("        <h1>Requisição número " + (++count) +
"</h1>");

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

    }
}
```

| Servlets | Ciclo de Vida



| Servlets | Importância do ciclo de vida

- ❑ Alguns servlets podem ter regras complexas para serem iniciados:
 - ❑ Criar um diretório, estabelecer conexões, inicializar serviços, carregar recursos na memória, etc.
- ❑ É importante liberar recursos adicionais inicializados com o Servlet.

```
web.xml

<?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>olamundo</servlet-name>
    <servlet-
class>com.exemplo.controller.OlaMundo</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>olamundo</servlet-name>
    <url-pattern>/ola-mundo</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

| Servlets | Importância do ciclo de vida

- ❑ Alguns servlets podem ter regras complexas para serem iniciados:
 - ❑ Criar um diretório, estabelecer conexões, inicializar serviços, carregar recursos na memória, etc.
- ❑ É importante liberar recursos adicionais inicializados com o Servlet.



| Servlets | Importância do ciclo de vida

- ❑ Crie o Servlet **CicloDeVidaServlet** mapeado para a URL **/ciclo-vida**.
- ❑ Nesse Servlet sobrescreva os métodos **init**, **service** e **destroy** conforme exemplo ao lado.
- ❑ Realize testes e verifique o ciclo de vida.

CicloDeVida.java

```
public class CicloDeVida extends HttpServlet {
```

Executado quando o Servlet é acionado a primeira vez.

```
@Override
public void init(ServletConfig conf) throws ServletException {
    System.out.println("Método init("+conf+")");
}
```

Executado a cada requisição.

```
@Override
public void service(HttpServletRequest req, HttpServletResponse
res)
    throws ServletException, IOException {
    System.out.println("Método service("+req+", "+res+")");
}
```

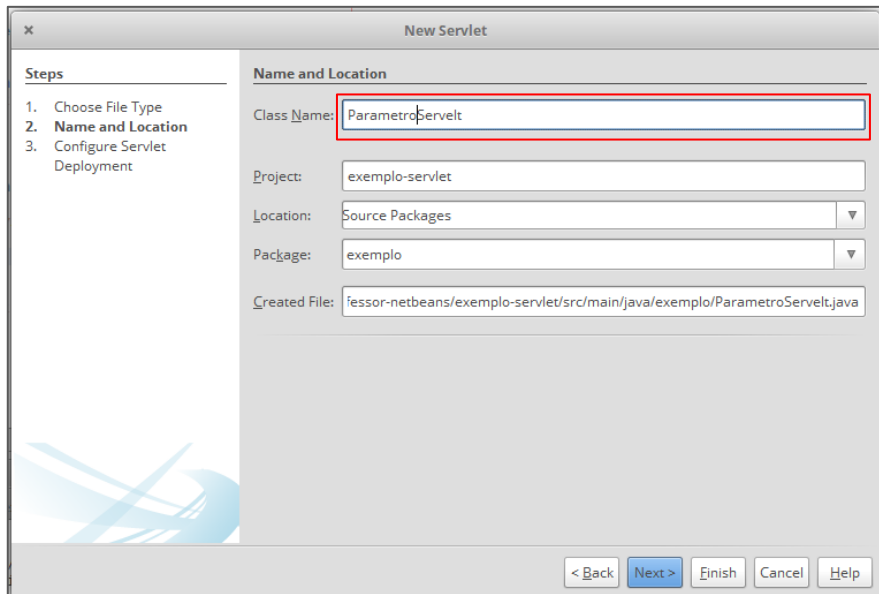
Executado quando o Servlet é finalizado.

```
@Override
public void destroy() {
    System.out.println("Método destroy()");
}
```

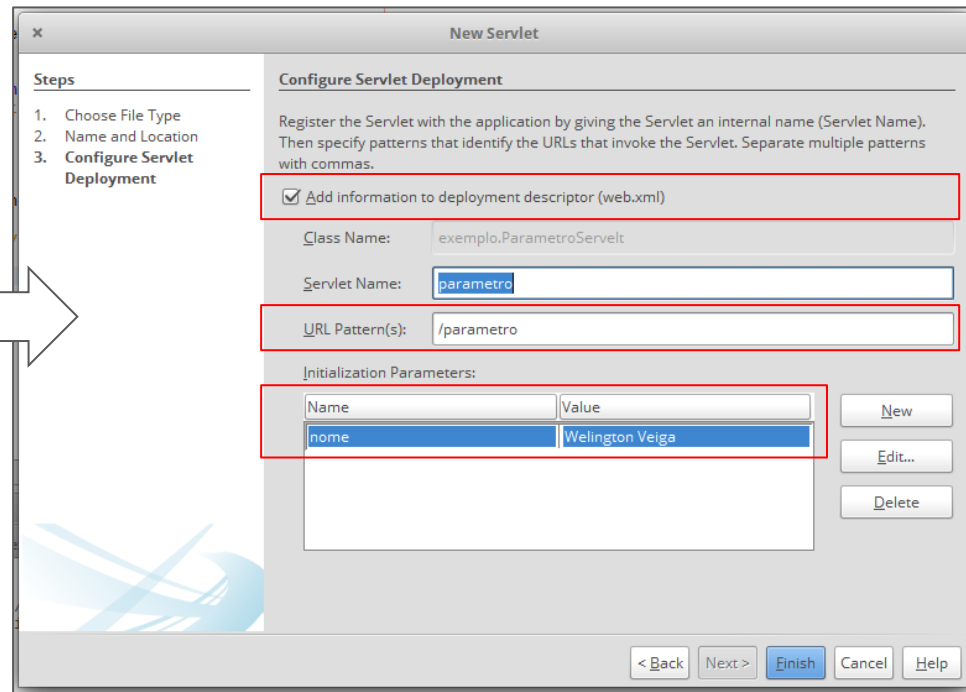
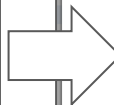
```
}
```

| Servlets | Parâmetros de Inicialização

- ❑ Servlets suportam parâmetros de inicialização, que permitem a configuração do comportamento do servlet.



The 'New Servlet' dialog is shown at the 'Name and Location' step. The 'Class Name' field is highlighted with a red box and contains the text 'ParametroServlet'. Other fields include 'Project' (exemplo-servlet), 'Location' (Source Packages), 'Package' (exemplo), and 'Created File' (fessor-netbeans/exemplo-servlet/src/main/java/exemplo/ParametroServlet.java). The 'Steps' list on the left shows 'Name and Location' as the current step.



The 'New Servlet' dialog is shown at the 'Configure Servlet Deployment' step. The 'Steps' list on the left shows 'Configure Servlet Deployment' as the current step. The 'Add information to deployment descriptor (web.xml)' checkbox is checked and highlighted with a red box. The 'Class Name' field contains 'exemplo.ParametroServlet'. The 'Servlet Name' field is highlighted with a red box and contains 'parametro'. The 'URL Pattern(s)' field is highlighted with a red box and contains '/parametro'. The 'Initialization Parameters' table is highlighted with a red box and contains one row with 'nome' as the Name and 'Wellington Veiga' as the Value.

Name	Value
nome	Wellington Veiga

| Servlets | Parâmetros de Inicialização

- ❑ Os parâmetros são definidos no **web.xml**.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app ...>
```

```
<servlet>
```

```
<servlet-name>parametro</servlet-name>
```

```
<servlet-class>exemplo.ParametroServlet</servlet-class>
```

```
<init-param>
```

```
<param-name>nome </param-name>
```

```
<param-value>Tassio Sirqueira</param-value>
```

```
</init-param>
```

```
</servlet>
```

```
<servlet-mapping>
```

```
<servlet-name>parametro</servlet-name>
```

```
<url-pattern>/parametro</url-pattern>
```

```
</servlet-mapping>
```

```
<session-config>
```

```
<session-timeout>
```

```
30
```

```
</session-timeout>
```

```
</session-config>
```

```
</web-app>
```

Nome e valor

| Servlets | Parâmetros de Inicialização

- ❑ Os parâmetros são definidos no **web.xml**.
- ❑ Podem ser obtidos no Servlet por meio do **ServletConfig**.
 - ❑ Disponível no método **init**.
 - ❑ Acessível via **getServletConfig()**

```
public class ParametroServlet extends HttpServlet {  
    @Override  
    public void init(ServletConfig config) throws ServletException {  
        System.out.println("Nome padrão: " + config.getInitParameter("nome"));  
    }  
  
    @Override  
    public void service(HttpServletRequest request, HttpServletResponse  
        response)  
        throws ServletException, IOException {  
        request.setCharacterEncoding("UTF-8");  
  
        String nome = request.getParameter("nome");  
        if (nome == null) {  
            nome = getServletConfig().getInitParameter("nome");  
        }  
  
        response.setContentType("text/html; charset=UTF-8");  
        try (PrintWriter out = response.getWriter()) {  
            /* TODO output your page here. You may use following sample code. */  
            out.println("<!DOCTYPE html>");  
            out.println("<html>");  
            out.println("    <head>");  
            out.println("    </head>");  
            out.println("    <body>");  
            out.println("        <h1>Nome: " + nome + "</h1>");  
            out.println("    </body>");  
            out.println("</html>");  
        }  
    }  
}
```

| Servlets | Servlet 3.0 e anotações

- ❑ Vimos que o deployment descriptor (**web.xml**) informa ao Servidor de Aplicação quais classes são Servlets e aos recursos (URLs) aos quais eles correspondem.
- ❑ A partir da versão 3.0, existe uma forma mais simples de criar Servlets.
- ❑ Por meio de anotações.
 - ❑ Recurso inserido na versão 6 do Java.
 - ❑ Permite “anotar” classes com informações adicionais de configuração.



| Servlets | Servlet 3.0 e anotações

- ❑ Mantenha o **web.xml** para configurações adicionais da aplicação.
- ❑ O arquivo ainda é obrigatório.

```
web.xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```



| Servlets | Servlet 3.0 e anotações

- ❑ Crie Servlets utilizando apenas a anotação **@WebServlet**

web.xml

```
@WebServlet(value="/olamundo", name="ServletOiMundo")
public class OiMundo extends HttpServlet {
    protected void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        String nome = request.getParameter("nome");
        PrintWriter out = response.getWriter();
        out.println("Ola Mundo Servlet 3: " + nome);
    }
}
```



| Servlets | Servlet 3.0 e anotações

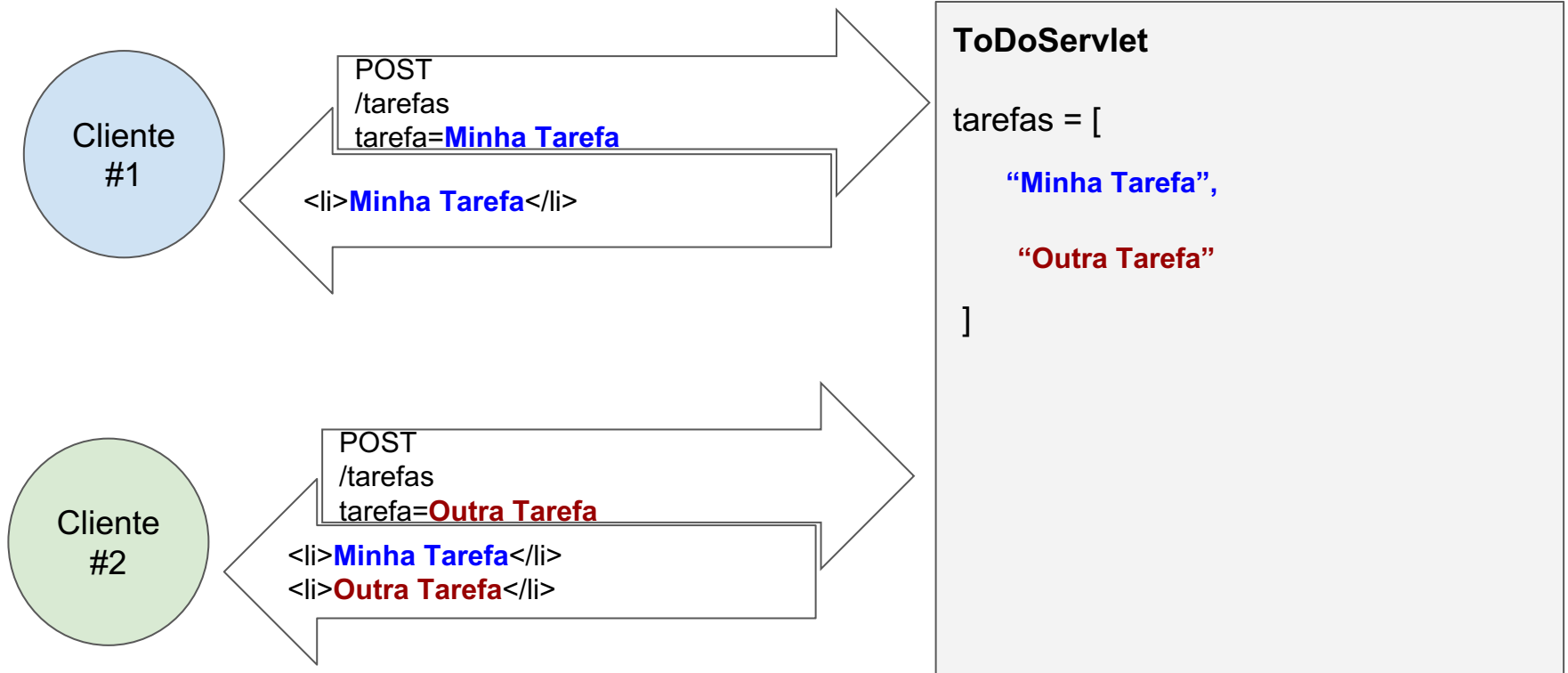


| Servlets | Servlet 3.0 e anotações

- ❑ **Atenção:** não misture no mesmo projeto anotações com XML. Mesmo funcionando, prejudica a legibilidade do código do projeto,



| Servlets | Armazenando Informações | Problema



| Servlets | Armazenando Informações | Problema

- ❑ Informações relacionadas à um cliente específico **não podem** ser armazenadas como atributo no Servlet.
- ❑ Todas as requisições compartilham a **mesma instância de Servlet**.



| Servlets | Armazenando Informações | Problema

- ❑ Informações relacionadas à um cliente específico **não podem** ser armazenadas como atributo no Servlet.
- ❑ Todas as requisições compartilham a **mesma instância de Servlet**.
- ❑ Como o **HTTP não mantém estado** de sessão, como resolver esse problema?
- ❑ Exemplos:
 - ❑ Acesso e login
 - ❑ Carrinho de compras.
 - ❑ Preferências



| Servlets | Armazenando Informações | Problema

- ❑ Como o protocolo não suporta sessões, as aplicações precisam fazer esse controle.
- ❑ Dessa forma é necessário uma forma de identificar um cliente específico.
- ❑ Para isso existem duas estratégias principais
 - ❑ Cookies.
 - ❑ Parâmetro adicional na URL.



| Servlets | Sessões

- ❑ O JavaEE oferece uma abstração para tratar dados de sessão.
 - ❑ A interface **HttpSession**
- ❑ Podemos criar ou obter a a sessão existente a partir de uma requisição.

Servlet.java

```
HttpSession session = request.getSession(true);
```

Obtém a sessão existente ou cria uma nova sessão para aquele cliente.

```
if (session.isNew()) {
```

O **isNew** verifica se a sessão acabou de ser criada ou já existia.

```
    session.setAttribute("corPreferida", "#000000");
```

Adicionamos atributos na sessão por meio do **setAttribute**.

```
}
```

```
String cor = (String) session.getAttribute("corPreferida");
```

Obtemos atributos na sessão por meio do **getAttribute**.

O cliente **não** tem acesso a esses dados que ficam apenas no servidor.

| Servlets | Sessões | Cookies

- ❑ Como o servidor reconhece um cliente para carregar a sessão correta?
 - ❑ Por meio do **JSESSIONID**, uma identificação randômica gerada para cada sessão nova.
 - ❑ Essa sessão pode ser armazenada em um cookie ou enviada como parâmetro na URL.
- ❑ Onde os dados da sessão são armazenados?
 - ❑ O Servidor de Aplicação pode armazenar na memória, no disco ou em um banco de dados, por exemplo.
 - ❑ Todo objeto na sessão deve ser **serializável**.



| Servlets | Sessões | Cookies

- ❑ Mas o que são **cookies**?
 - ❑ Um Cookie HTTP são pequenas porções de informações que o servidor pode enviar para o Navegador.
 - ❑ O navegador armazena essas informações e devolve para o servidor quando faz novas requisições.
 - ❑ Assim é possível guardar preferências, sessões, carrinhos de compras, histórico de buscas entre outras informações pequenas.
 - ❑ O navegador pode não suportar cookies e o usuário pode desabilitá-los.
- ❑ **Cookies não são seguros**, podem ser lidos no lado cliente, por isso não devem conter **Informações sensíveis**.



| Servlets | Sessões | Cookies

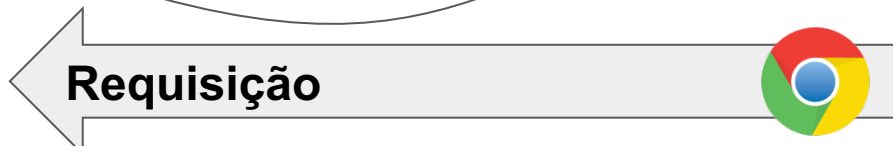
❑ Como funciona?



```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry

[page content]
```

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco;tasty_cookie=strawberry
```



Requisição

| Servlets | Sessões | Cookies

- ❑ Mas e se os **cookies** estiverem desabilitados?
 - ❑ Um parâmetro é adicionado pelo Servidor de Aplicação na URL
 - ❑ Exemplo:
 - ❑ `http://exemplo.com;JSESSIONID=udhofhoi4fojnio4no4n`



| Servlets | Sessões | Cookies | Exemplo

```
public class Contador extends HttpServlet {
```

```
    @Override
```

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

```
        HttpSession session = request.getSession(true);  
        if (session.isNew()) {  
            session.setAttribute("count", 0);  
        }
```

Vamos obter uma sessão e, caso ela não exista,
a mesma será criada.

```
        int count = (int) session.getAttribute("count");  
        count++;
```

```
        session.setAttribute("count", count);
```

Realizamos as operações necessárias e armazenamos
novamente os dados como atributo da sessão.

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

```
        try (PrintWriter out = response.getWriter()) {  
            /* TODO output your page here. You may use following sample code. */  
            out.println("<!DOCTYPE html>");  
            out.println("<html>");  
            out.println("    <head>");  
            out.println("        <title>Servlet Contador</title>");  
            out.println("    </head>");  
            out.println("    <body>");  
            out.println("        <h1>Requisição número " + count + "</h1>");  
            out.println("    </body>");  
            out.println("</html>");  
        }
```

```
    }
```

```
}
```

| HTTP | Referências adicionais

1. <https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html>
2. <https://www.caelum.com.br/apostila-java-web/servlets/>
3. <http://blog.caelum.com.br/java-ee6-comecando-com-as-servlets-3-0/>
4. <https://www.caelum.com.br/apostila-java-web/apendice-topicos-da-servlet-api/>
5. <https://www.tutorialspoint.com/servlets/servlets-form-data.htm>