Java Web

AULA 03 – SERVLETS PARTE I

Objetivos e Conceitos

- Objetivos:
  - Apresentar Servlets e tecnologias relacionadas. Tornar o aluno apto a escrever servlets recebendo informações de formulários.
- Conceitos:
  - Servlets. Ciclo de Vida. Formulários.

Tópicos

1. Introdução
2. Estrutura de uma Servlet
3. Métodos de uma Servlet
4. Requisição e Resposta
5. Ciclo de Vida
6. Processamento de Formulários

1 Introdução

Servlets

Classes Java

Podem ser instaladas em um servidor que implementa um Servlet Container

Tratam requisições HTTP
- Ex.: Requisições de um navegador

Dão uma resposta ao cliente
- Ex.: Uma página HTML

Seguem uma estrutura definida
- Classe Java com elementos que a tornam uma Servlet

Servlets

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.
5.  @WebServlet("Teste")
6.  public class Teste extends HttpServlet {
7.     public void doGet(HttpServletRequest request,
8.                       HttpServletResponse response)
9.                   throws ServletException, IOException {
10.
11.       PrintWriter out = response.getWriter();
12.       response.setContentType("text/html");
13.       out.println("<html><head><title>Teste</title></head>");
14.       out.println("<body>Teste de Servlet</body></html>");
15.       out.flush();
16.    }
17. }
```

Servlets

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.
5.  @WebServlet("Teste")
6.  public class Teste extends HttpServlet {
7.      public void doGet(HttpServletRequest request,
8.                        HttpServletResponse response)
9.                    throws ServletException, IOException {
10.
11.       PrintWriter out = response.getWriter();
12.       response.setContentType("text/html");
13.       out.println("<html><head><title>Teste</title></head>");
14.       out.println("<body>Teste de Servlet</body></html>");
15.       out.flush();
16.   }
17. }
```


Servlets Básico - Netbeans

```
import *;

@WebServlet(urlPatterns = {"/TesteServlet1"})
public class TesteServlet1 extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
            throws ServletException, IOException {
        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>Servlet TesteServlet1</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet TesteServlet1 at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }
+   HttpServlet methods. Click on the + sign on the left to edit the code.
}
```

Servlets Básico – Exemplo 2

```
import *;

@WebServlet(urlPatterns = {"/TesteServlet1"})
public class TesteServlet1 extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
            throws ServletException, IOException {
        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>Teste TADS</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Minha Primeira Servlet</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }
+   HttpServlet methods. Click on the + sign on the left to edit the code.
}
```


Exercícios

1. Executar a primeira Servlet
   a. Criar novo projeto no Netbeans
   b. Adicionar uma Servlet
   c. Alterá-lo para o Exemplo 2 anterior
   d. Executar

2 Estrutura de uma Servlet

Estrutura de uma Servlet

Seguem uma estrutura definida

- Herdam de **`HttpServlet`**
- Anotadas com **`@WebServlet`**
- Implementam métodos, ex.: **`doGet()`** e **`doPost()`**
- Métodos **`doGet/doPost`** recebem um *request* (**`HttpServletRequest`**)
- Métodos **`doGet/doPost`** entregam um *response* (**`HttpServletResponse`**)

Exercícios

1. Encontre os elementos da estrutura de uma Servlet no último projeto criado
2. Criar uma aplicação contendo uma servlet que mostra uma página pessoal contendo:
   a) Nome completo
   b) Telefone e endereço completo
   c) Cidade natal
   d) Link para o Google

   Use cores para diferenciar cada um dos elementos acima

Servlets

A anotação **`@WebServlet`** indica como será acessada a página no navegador

```
@WebServlet("Teste")
```

- Indica que a página é acessada com **Teste**

```
http://localhost:8080/app/Teste
```

Neste exemplo

```
@WebServlet(urlPatterns = {"/TesteServlet1"})
```

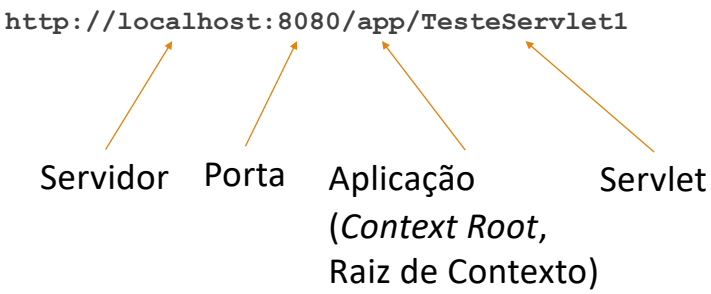
- Indica que a página é acessada com **`TesteServlet1`**

```
http://localhost:8080/app/TesteServlet1
```

O atributo **`urlPatterns`** indica o nome usado para acessar a Servlet

- Não use `.java` nem `.class` na URL para acessar uma Servlet

Servlets



Várias Servlets.

- Uma aplicação pode conter várias Servlets
  - Cada uma servindo uma página diferente
  - Podem ser ligadas através de *links*
- Se uma Servlet está anotada com:

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

- Então o link para ela deve ser colocado como:

```
<a href="SegundaServlet">Ir para Segunda Servlet</a>
```

- Na Servlet (sem a "/")

```
out.println("<a href=\"SegundaServlet\">Ir para Segunda
Servlet</a>");
```

Exercícios..

1. Na aplicação de página pessoal, adicione mais uma servlet contendo o nome de suas músicas favoritas. Altere o "Link para o Google" da servlet que já estava criada para ser um link para esta nova servlet.

3 Métodos de uma Servlet

Métodos de uma Servlet

Classe Abstrata **`HttpServlet`**

- Servlets herdam de **`HttpServlet`**

Contém métodos que tratam uma requisição

- **HTTP GET**: `doGet(HttpServletRequest req, HttpServletResponse resp)`
- **HTTP POST**: `doPost(HttpServletRequest req, HttpServletResponse resp)`
- **HTTP PUT**: `doPut(HttpServletRequest req, HttpServletResponse resp)`
- **HTTP DELETE**: `doDelete(HttpServletRequest req, HttpServletResponse resp)`

Método de Informação

- **`getServletInfo()`** : retornar alguma informação, ex, autor, data, etc

Métodos do ciclo de vida

- **`init(ServletConfig config)`**
- **`destroy()`**

Métodos de uma Servlet

Método **`service(HttpServletRequest request, HttpServletResponse response)`**

- Primeiro método a ser invocado em uma requisição
- **Não deve ser sobrescrito**
- Verifica o método de chamada e invoca **`doPost()`**, **`doGet()`**, etc.

Método **`doGet(HttpServletRequest request, HttpServletResponse response)`**

- Chamado quando a requisição for do tipo GET

Método **`doPost(HttpServletRequest request, HttpServletResponse response)`**

- Chamado quando a requisição for do tipo POST

Métodos de uma Servlet.

Se for necessário tratar, por exemplo, GET da mesma forma que POST, usar:

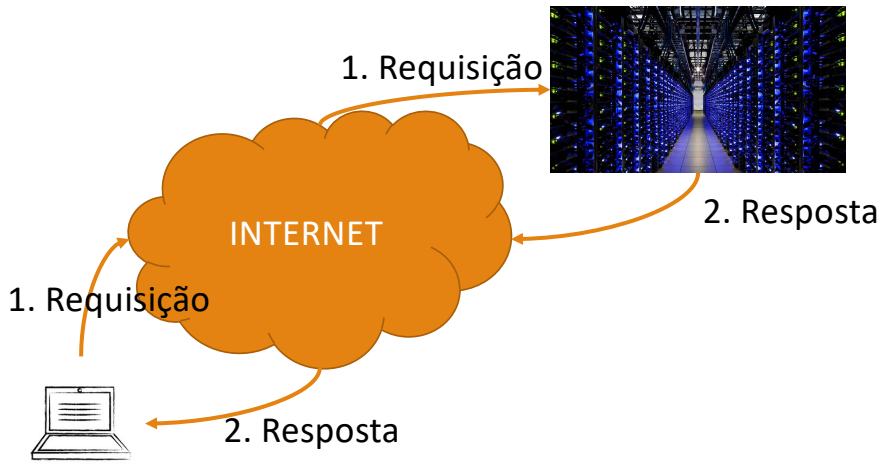
```
protected void doGet(HttpServletRequest req,
                     HttpServletResponse resp)
                  throws ServletException,IOException {
            doPost(req, resp);
}
protected void doPost(HttpServletRequest req,
                     HttpServletResponse resp)
                  throws ServletException,IOException {
            // Código aqui
}
```


Exercícios..

1. Crie uma nova aplicação contendo um HTML e uma SERVLET
   a. No HTML coloque um link para servlet e um formulário que invoca a servlet via POST
   b. Na servlet escreva os métodos doGet() e doPost()
      i. No método doGet(), mostre uma página mostrando que houve uma chamada via GET
      ii. No método doPost(), mostre uma página mostrando que ouve uma chamada via POST
   c. DICA: para funcionar no Netbeans:
      i. Apaque o método processRequest()
      ii. Abra o comentário escondido onde estão já implementados os métodos doGet() e doPost()
      iii. Escreva os conteúdos lá

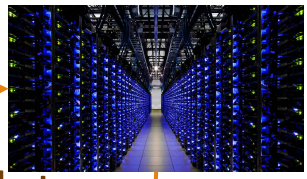
3 Requisição e Resposta

Requisição e Resposta



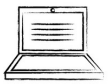
Requisição e Resposta

Requisição:
`HttpServletRequest`



Servlet

Resposta:
`HttpServletResponse`



✓ Objetos recebidos nos métodos **`doXXX()`**

Requisição e Resposta

Interface **`HttpServletRequest`**

- Herda da interface **`ServletRequest`**
- Encapsula a **requisição**
- Pode-se obter dados passados para a página (via formulários ou na URL com ? e &)
- Métodos
  - **`getContextPath()`**: retorna a raiz do contexto da aplicação. Ex, se a aplicação rodar em : **`http://www.servidor.com.br:8080/MinhaApp/index.jsp`**, ele retorna "MinhaApp"
  - **`getServerName()`** – ServletRequest: retorna nome do servidor (www.servidor.com.br)
  - **`getServerPort()`** – ServletRequest: retorna a porta da requisição (8080)
  - **`getHeader()`** : retorna o valor de um header que foi enviado na requisição
  - **`getCookies()`** : retorna um vetor contendo os cookies que o navegador enviou
  - **`getSession()`** : usado para obter uma sessão
  - **`getParameter()`** – ServletRequest: retorna o valor de um parâmetro da requisição
  - **`getAttribute()`** – ServletRequest: retorna o valor de um atributo guardado no escopo da requisição
  - **`setAttribute()`** – ServletRequest: armazena um dado no escopo da requisição
  - **`getRequestDispatcher()`** – ServletRequest: usado para fazer redirecionamentos

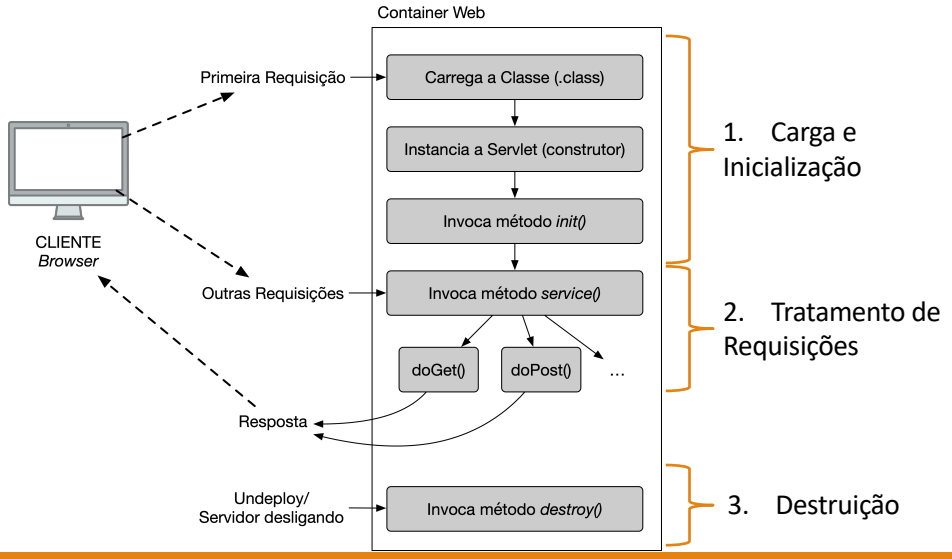
Requisição e Resposta..

Interface **`HttpServletResponse`**

- Herda da interface **`ServletResponse`**
- Encapsula a **resposta**
- Deve-se escrever o HTML neste objeto
- Pode-se escrever cookies a serem enviados para o navegador
- Métodos:
  - **`addCookie()`** : adiciona um cookie para ser enviado ao navegador
  - **`sendRedirect()`** : efetua um redirecionamento
  - **`setHeader()`** : adiciona um header na resposta a ser enviado ao navegador
  - **`getWriter()`** – ServletResponse : obtém um objeto PrintWriter, usado para escrever HTML para o cliente
  - **`setContentType()`** – ServletRequest : seta o conteúdo de saída da requisição

4 Ciclo de Vida

Ciclo de Vida de uma Servlet



Ciclo de Vida de uma Servlet

1. Criação e Inicialização
   - Pode ocorrer na primeira requisição OU quando a aplicação é iniciada
   - Método **`init(ServletConfig config)`**:
     - Contêiner invoca automaticamente
     - Invocado somente uma vez no ciclo de vida
2. Tratamento de Requisições
   - Método **`service(ServletRequest req, ServletResponse res)`**:
     - Trata requisições do usuário
     - Pode chamar **`doPost()`**, **`doGet()`**, etc
     - Cada requisição roda em uma *thread* separada
3. Destruição
   - Método **`destroy()`**:
     - Contêiner invoca automaticamente
     - Invocado no *undeploy,* ou servidor desligando, ou falta de memória, ou quando não requisitado por muito tempo
     - Servlet estará disponível para o GC

Ciclo de Vida de uma Servlet.

A requisição é tratada em *multithread*

- Cada requisição invoca o **`service()`** (e os **`doXXX()`**) em uma *thread* separada
- **O objeto da Servlet é o mesmo**

Cuidados

- A Servlet é uma classe, então ela pode conter atributos (membros)
- Estes membros são compartilhados entre TODAS as *threads*
- Cuidado ao acessar membros de Servlets
  - Ao alterar os dados, TODAS as *threads* receberão a alteração
  - Cuidado com *race-condition*
- Variáveis dentro dos métodos **`doXXX()`** são *thread-safe*
- Mas se elas apontarem para objetos compartilhados, deve-se ter cuidado
- Objetos *request* e *response* são *thread-safe*

Exercícios..

1. Crie uma nova aplicação contendo uma SERVLET
   a. Sobrescrever os métodos `init(ServletConfig config)` e `destroy()`
      i. No Netbeans - CTRL+i ou no menu Source | Insert Code
   b. No corpo dos métodos adicionar, respectivamente:

```
System.out.println("INICIANDO A SERVLET");

System.out.println("DESTRUINDO A SERVLET");
```

   c. Adicionar também um código destes no `processRequest()`
   d. Rodar o projeto e a Servlet
   e. Verificar no console do Glassfish se aparece o texto de inicialização e do atendimento da requisição
   f. Efetuar o *undeploy* da aplicação (No Netbeans - Aba Services | Servers)
   g. Verificar no console do Glassfish se aparece o texto de destruição

5 Processamento de Formulários

Processamento de Formulários

- NO HTML (cliente)
  - Formulário com campos e botão de submit
  - No campo "action" colocar o nome da Servlet
  - Definido no **urlPatterns**

```
<form action="MyServlet" method="post">
```

- NA SERVLET (servidor)
  - Parâmetro **urlPatterns** da anotação **@WebServlet**

```
@WebServlet(urlPatterns = {"/MyServlet"})
public class Processa extends HttpServlet {
```

Processamento de Formulários

Parâmetros de Formulários

- Conforme o método HTTP no formulário, chama um método da Servlet
  - `doGet(), doPost()`
- Campos do formulário são obtidos da requisição
- Atributo "`name`" do campo deve bater com o obtido com **`request.getParameter("...")`**
- Sempre retorna **String**

➢NO HTML (cliente)

```
<input type="text" name="nome" value=""/>
```

➢NA SERVLET (servidor)

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                throws ServletException, IOException {
    String usuario = request.getParameter("nome");
```


Processamento de Formulários.

Parâmetros

- **`request.getParameter()`** sempre retorna String

Em caso de números ou datas, deve-se fazer a conversão explícita na Servlet

```
String           nr     = request.getParameter("idade");
int              numero = Integer.parseInt(nr);
String           dt     = request.getParameter("data");
Date             data   = null;
SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
try {
    data = format.parse(dt);
    ...
}
catch (ParseException e) {
    ...
}
```

Exemplo: index.html

Criar um novo Projeto

Alterar o `index.html` para conter o código abaixo

Criar uma servlet para receber os campos "nome" e "email"

```
<html>
<head><title>Teste</title></head>
<body>
<form action="Processa" method="post">
  Nome:   <input type="text" name="nome" value="" />  <br/>
  E-mail: <input type="text" name="email" value="" /> <br/>
  <input type="submit" value="Ok">
</form>
</body>
</html>
```


Exemplo: Servlet Processa.java

```
package com.cursojava;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet(urlPatterns = {"/Processa"})
public class Processa extends HttpServlet {
   protected void doPost(HttpServletRequest request,
                         HttpServletResponse response)
                   throws ServletException, IOException {

      String usuario = request.getParameter("nome");
      String email   = request.getParameter("email");
```

Exemplo: Servlet Processa.java (cont.).

```
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<title>Teste</title></head><body>");

        out.println("<h1>Dados do formulario</h1>");
        out.println("Usuario: " + usuario + "<br/>");
        out.println("Email: "   + email   + "<br/>");

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


Exercícios.

1. Executar a aplicação anterior, que recebe os dados de um formulário e os apresenta.
2. Adicionar mais campos texto no formulário e apresentá-los:
   - Endereço
   - Número
   - Cidade
   - Estado

Processamento de Formulários

Outros tipos de Campos de Formulário (`"type"`, ou outras *tags*)

- **`text`**, **`password`**, **`textarea`**: obtidos da mesma forma já vista
- **`checkbox`**:
  - Quando marcado, envia o parâmetro com o valor do atributo `"value"`
  - Se não for marcado, não envia o parâmetro (recebe `null`)

➢ No HTML

```
<input type="checkbox" name="fumante" value="F"/>
```

➢ Na Servlet

```
String str = request.getParameter("fumante");
```

Se o campo foi check-ado, variável `str` recebe `"F"` (o que estiver em `value`)
Se não foi check-ado, variável `str` é `null`

Processamento de Formulários

Campos de Formulário

- **`radio`**:
  - Quando marcado, envia o parâmetro com o valor do atributo `"value"`
  - Se não for marcado, não envia o parâmetro (recebe `null`)

➢ No HTML

```
<input type="radio" name="status" value="A"/>
<input type="radio" name="status" value="I"/>
```

➢ Na Servlet

```
String st = request.getParameter("status");
```

Se algum marcado, variável `st` recebe ou `"A"` ou `"I"`
Se nenhum marcado, variável `st` é `null`

Processamento de Formulários

Campos de Formulário

- **`select`**: ComboBox
  - Envia o valor (atributo `value`) do elemento (`option`) selecionado.
  - O nome do componente a ser obtido na Servlet é o atributo `name` do `select`

➢No HTML

```
<select name="estado">
   <option value="PR"> Paraná </option>
   <option value="SC"> Santa Catarina </option>
</select>
```

➢Na Servlet

```
String est = request.getParameter("estado");
```

Variável `est` recebe ou `"PR"` ou `"SC"`, conforme o que foi escolhido
Se nenhum marcado, variável `est` é `null`

Processamento de Formulários.

Lista de campos com o mesmo nome

- Exemplo: **ListBox** onde é possível selecionar mais de um valor
- Usa-se: **`request.getParameterValues()`**

```
String[] dados = request.getParameterValues("dado");
```

➢No HTML

```
<input type="text" name="dado" value=""/>
<input type="text" name="dado" value=""/>
<input type="text" name="dado" value=""/>
```

➢Na Servlet

```
String[] dados = request.getParameterValues("dado");
```

Obtém uma lista com todos os elementos que foram passados, e somente os que foram passados

Exercícios..

1. Criar e executar a aplicação anterior
2. Na aplicação anterior, adicionar um campo "idade" e "data de nascimento". Fazer o tratamento correto de valor inteiro e data.
3. Na aplicação anterior (execício 2), criar uma tabela com os dados nome, e-mail, idade e data de nascimento. Inserir os dados nesta tabela ao tratar o formulário.
4. Na aplicação anterior (exercício 3), adicione um campo sexo (radio button) , fumante (check box) e observação (text area). Alterar a tabela para permitir a inserção destes campos e alterar a aplicação para persistir estes dados.