



Servlets Java: Conceitos Básicos

Prof. Edson A. O. Junior

edson@edsonjr.pro.br



Tópicos

- Papel do Servlet – Introdução
- Ciclo de Vida de um Servlet
- Exemplo de Servlet
- HttpRequest e HttpResponse
- Redirecionamento
- Parâmetros de Inicialização e de Contexto
- Atributos e seus Escopos



Introdução

- O papel de um servlet é:
 - receber uma requisição HTTP de um cliente (navegador);
 - processá-la; e
 - enviar uma resposta de volta ao cliente.
- Os servlets são controlados pelo container Web
- Servlets são classes Java que implementam métodos para cada um dos métodos HTTP:
 - get, post, head, etc..
 - instâncias da classe `javax.servlet.http.HttpServlet`



Ciclo de Vida de um Servlet

- O container Web:
 1. carrega a classe Servlet
 2. instancia o servlet – chama o construtor padrão
 3. o método `init()` é chamado:
 1. chamado uma única vez enquanto o servlet existir
 2. finalizado antes do método `service()` ser chamado
 4. o método `service()` é chamado: já é um objeto
 1. onde o servlet “gasta” mais tempo enquanto existir
 2. trata as requisições do cliente: `doGet()`, `doPost()`, etc
 3. cada requisição em uma thread separada
 5. o método `destroy()` é chamado:
 1. contem códigos de finalização do servlet
 2. chamado uma única vez

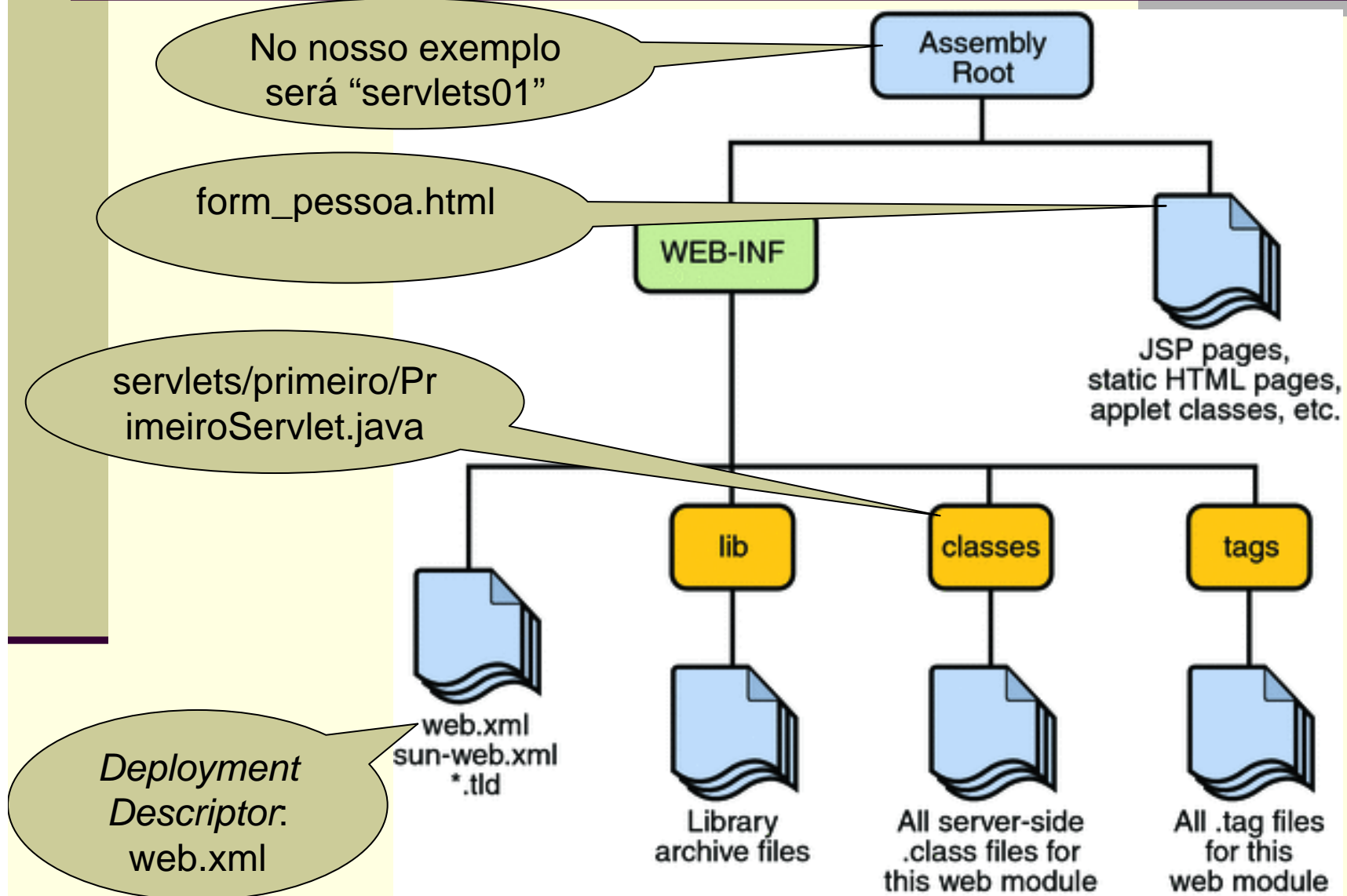


Ciclo de Vida de um Servlet

- O método `init()` pode ser sobreposto para, por exemplo:
 - iniciar conexões com BD
 - registrar objetos
- O método `service()` dificilmente é sobreposto, pois ele é responsável por chamar outros métodos para tratar as requisições:
 - `doGet()`, `doPost()`, etc...
- Os métodos `doGet()` e `doPost()` são **SEMPRE** sobrepostos, pelo menos um deles, pois:
 - é onde o usuário escreve o código para tratar as requisições do cliente
 - se não sobrepor, por exemplo, o método `doGet()`, indica que o servlet não suporta requisições HTTP do tipo `get`



Estrutura: Aplicações Web c/ Servlets





Exemplo de Servlet (WEB-INF/classes/)

```
package servlets.primeiro;

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

public class PrimeiroServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.print("Este servlet não suporta requisições GET...!!!");
    }

    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.print("Os dados digitados são: <BR>");
        out.print("Nome: " + req.getParameter("nome") + "<BR>");
        out.print("Idade: " + req.getParameter("idade") + "<BR>");
        out.print("Cidade: " + req.getParameter("cidade") + "<BR>");
        out.print("Estado: " + req.getParameter("estado") + "<BR>");

        out.print("Muito obrigado!!!");
    }
}
```



Criando o web.xml (WEB-INF/web.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">

  <display-name>Exemplo de Servlet</display-name>

  <servlet>
    <servlet-name>PrimeiroServlet</servlet-name>
    <servlet-
      class>servlets.primeiro.PrimeiroServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>PrimeiroServlet</servlet-name>
    <url-pattern>/primeiroServlet</url-pattern>
  </servlet-mapping>

</web-app>
```



Obtendo o form_pessoa.html

- Fazer o download do arquivo form_pessoa.html a partir do endereço:
 - http://www.edsonjr.pro.br/ensino/uem/form_pessoa.zip



Exemplo Servlet em Execução

1. Configurar o Eclipse para o Exemplo de Servlet
2. Iniciar o Container Tomcat
3. O exemplo está dividido em 02 etapas:
 1. Acessar o endereço:
<http://localhost:8080/servlets01/index.html>
 2. Acessar o endereço:
<http://localhost:8080/servlets01/primeiroServlet>
4. Observar a diferença no acesso aos endereços



Exemplo Servlet em Execução

The screenshot shows a Windows Internet Explorer browser window titled "Exemplo de Página HTML com Form - Windows Internet Explorer". The address bar shows the URL "http://localhost:8081/servlets01/". The page content is titled "Cadastro de Pessoas..." and contains a registration form with the following fields:

- Nome: Edson Jr.
- Idade: 29
- Cidade: Maringá
- Estado: PR

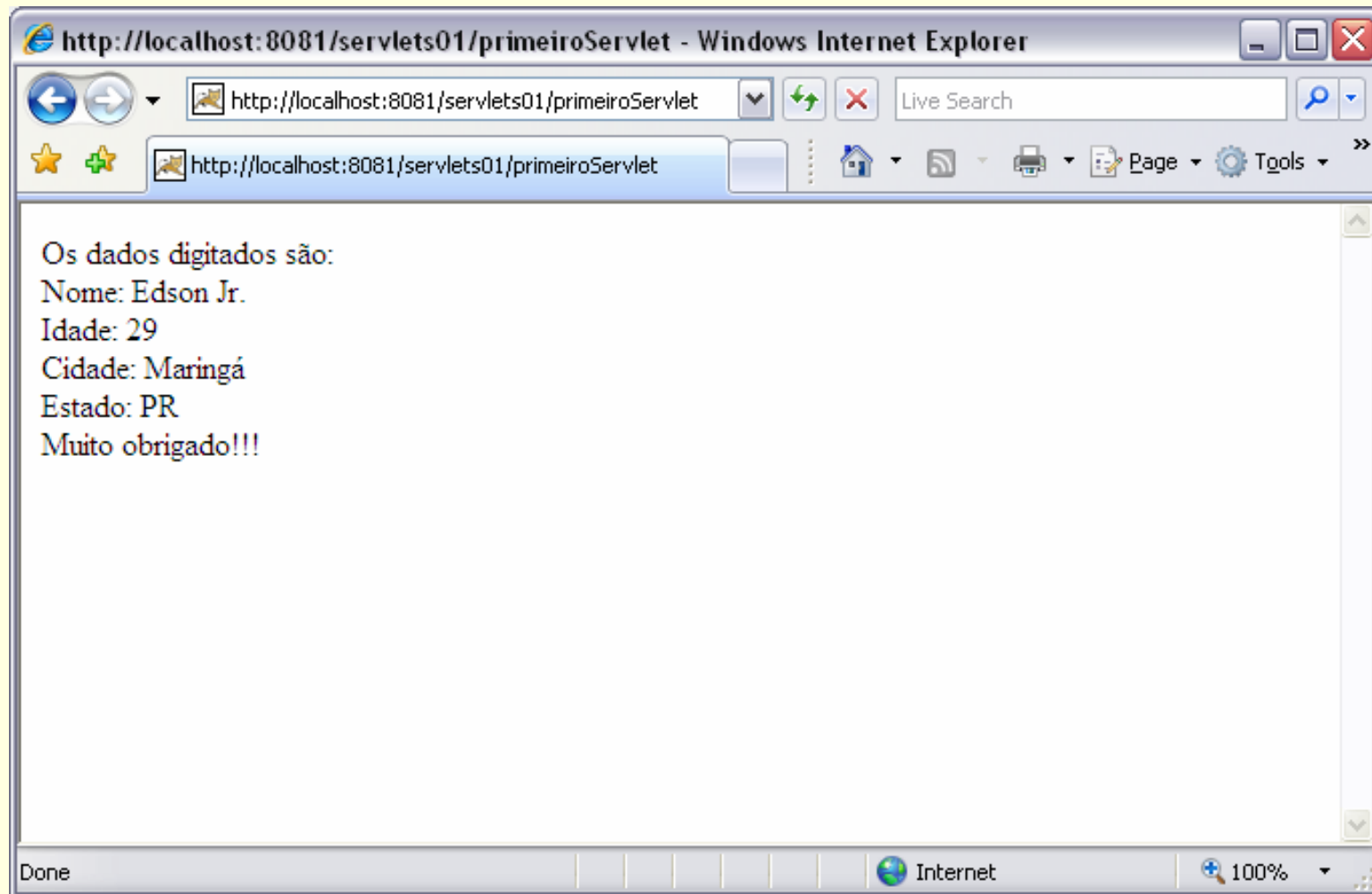
Below the form is a button labeled "Enviar dados". A dialog box titled "Windows Internet Explorer" is overlaid on the page, displaying a warning icon and the following text:

Você digitou os seguintes dados:
Nome: Edson Jr.
Idade: 29
cidade: Maringá
Estado: PR

An "OK" button is visible at the bottom of the dialog box. The browser's status bar at the bottom shows "primeiroServlet" on the left, "Internet" in the center, and "100%" on the right.

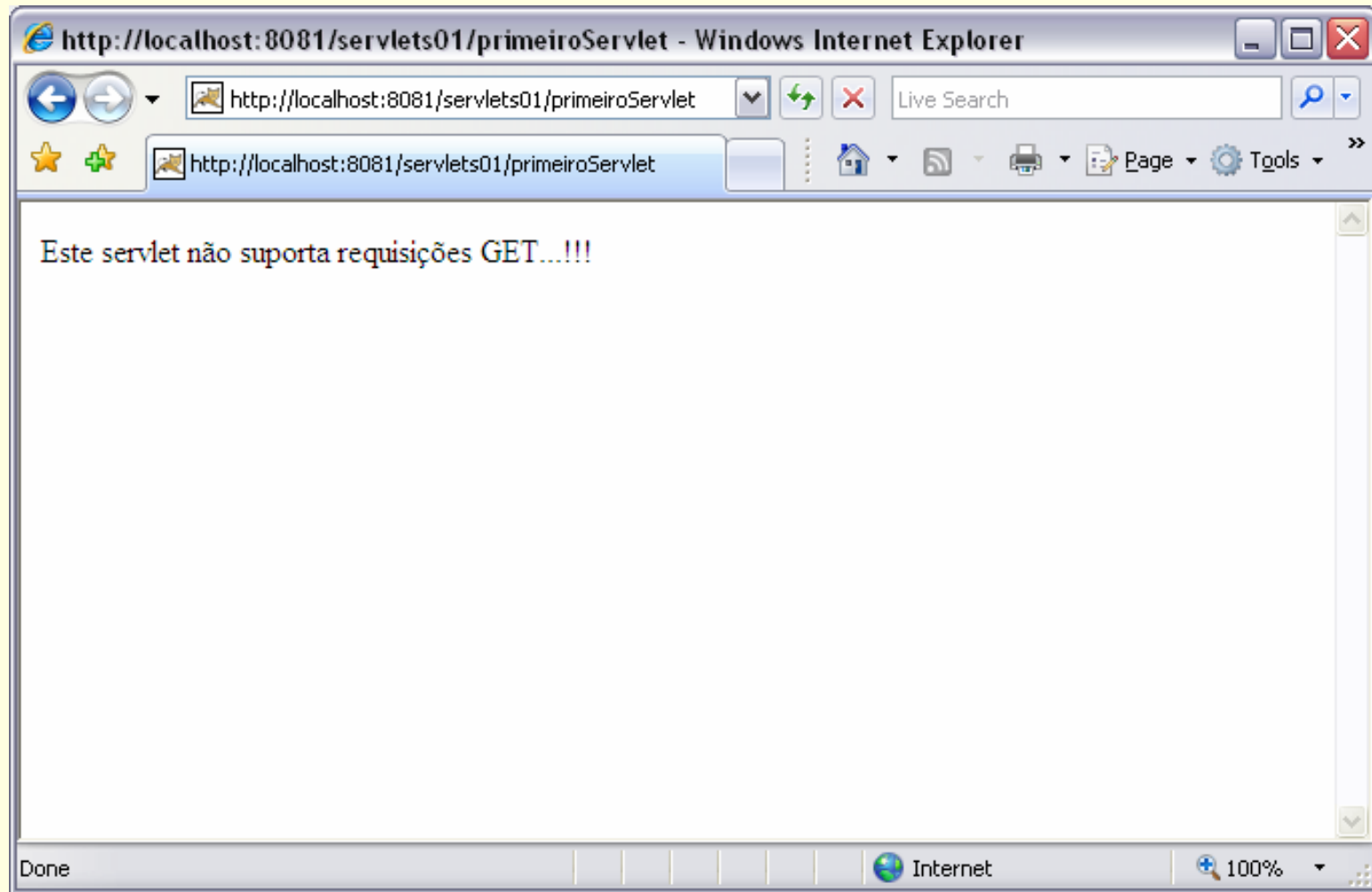


Exemplo Servlet em Execução





Exemplo Servlet em Execução





Exercícios - 05

1. Qual a ordem de execução dos métodos de um servlet: doPost(), service(), doGet() e init()?
2. Qual(is) desses métodos é (são) executado(s) toda vez que uma nova requisição chega ao servlet? Qual(is) dele(s) não são? Justifique.
3. Como o Container Web trata cada uma das requisições HTTP com relação à concorrência?
4. Por quê os servlets devem ser mapeados no web.xml? Por quê não acessá-los por uma URL direta até eles?



HttpRequest

- Além dos parâmetros, é possível acessar:
 - Informações sobre a plataforma do cliente e do navegador:

```
String cliente = request.getHeader("User-Agent");
```

- Os cookies associados com a requisição:

```
Cookeis[] cookies = request.getCookies();
```

- A sessão associada com a requisição:

```
HttpSession sessao = request.getSession();
```

- O método HTTP da requisição:

```
String metodo = request.getMethod();
```



HttpResponse

- Usada na maioria das vezes para enviar dados ao cliente
- 02 métodos principais:
 - `setContentTypes(String)`
 - `text/html, application/jar, application/pdf, etc...`
 - `getWriter()`
- É possível usar a resposta para definir:
 - Headers
 - Erros de envio
 - Cookies
- 02 tipos de saída:
 - Caracteres (`PrintWriter`) ou bytes (`ServletOutputStream`)
- Pode redirecionar a saída com o método `sendRedirect(String)`



Redirect vs. Request Dispatch

- O método `sendRedirect(String)` da resposta redireciona para uma URL e deixa o navegador tratar a URL:
 - pode ser um endereço externo à aplicação, por exemplo, <http://www.din.uem.br>
 - `request.sendRedirect("http://www.din.uem.br");`
- Já o método `getRequestDispatcher(String)` da requisição permite que o seu parâmetro seja um endereço relativo à aplicação ou um recurso, exemplo:
 - `request.getRequestDispatcher("resultado.jsp");`



Servlets Java: Conceitos Básicos

Parâmetros de Inicialização e de
Contexto



Parâmetros de Inicialização

- Fornecem valores iniciais para um determinado servlet
 - estão disponíveis somente para um único servlet em questão
- São definidos no *Deployment Descriptor* (web.xml), dentro da marcação <servlet>
- Não podem ser alterados, somente recuperados
- Só estão disponíveis depois que o servlet for inicializado, método init()

Exemplo de definição (web.xml):

```
<servlet>
  <servlet-name>PrimeiroServlet</servlet-name>
  <servlet-class>servlets.primeiro.PrimeiroServlet</servlet-class>

  <init-param>
    <param-name>email</param-name>
    <param-value>edson@edsonjr.pro.br</param-value>
  </init-param>
</servlet>
```

Exemplo de acesso (classe Servlet):

- `getServletConfig().getInitParameter("email");`



Exemplo: Parâmetros de Inicialização

- Adicionar ao web.xml o código a seguir:

```
<servlet>
  <servlet-name>ParamInicializacaoServlet</servlet-name>
  <servlet-class>servlets.parametros.inicializacao.
    ParametroInicializacaoServlet</servlet-class>

  <init-param>
    <param-name>nome</param-name>
    <param-value>Edson Alves de Oliveira Junior</param-value>
  </init-param>

  <init-param>
    <param-name>email</param-name>
    <param-value>edson@edsonjr.pro.br</param-value>
  </init-param>

  <init-param>
    <param-name>disciplina</param-name>
    <param-value>Linguagem de Programação para Web III</param-value>
  </init-param>

</servlet>
```



Exemplo: Parâmetros de Inicialização

```
package servlets.parametros.inicializacao;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class ParametroInicializacaoServlet extends HttpServlet {
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    resp.setContentType("text/html"); PrintWriter out = resp.getWriter();

    out.println("<h1>Parâmetros de Inicialização do Servlet: " +
this.getClass() + "</h1><BR><BR>");

    out.println("<b>Nome:</b> " +
this.getServletConfig().getInitParameter("nome") + "<BR>");

    out.println("<b>E-mail:</b> " +
this.getServletConfig().getInitParameter("email") + "<BR>");

    out.println("<b>Disciplina:</b> " +
this.getServletConfig().getInitParameter("disciplina") + "<BR>");
}

protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    resp.setContentType("text/html"); PrintWriter out = resp.getWriter();
    out.print("Este servlet não suporta requisições POST...!!!");
}
}
```



Parâmetros de Contexto

- Funcionam como os parâmetros de inicialização, porém os de contexto estão disponíveis para todos os recursos de uma aplicação Web
- 01 parâmetro de contexto por aplicação
- Vários parâmetros de inicialização por servlet

Exemplo de definição (web.xml):

```
<servlet>  
  <servlet-name>PrimeiroServlet</servlet-name>  
  <servlet-class>servlets.primeiro.PrimeiroServlet</servlet-class>  
</servlet>
```

```
<context-param>  
  <param-name>emailSuporte</param-name>  
  <param-value>suporte@din.uem.br</param-value>  
</context-param>
```

Exemplo de acesso (classe Servlet):

- `getServletContext().getInitParameter("emailSuporte");`



Exemplo: Parâmetros de Contexto

- Adicionar ao web.xml o código a seguir:

```
<servlet>
  <servlet-name>ParamContextoServlet</servlet-name>
  <servlet-class>servlets.parametros.contexto.ParametroContextoServlet</servlet-
    class>
</servlet>

<servlet-mapping>
  <servlet-name>ParamContextoServlet</servlet-name>
  <url-pattern>/paramContexto</url-pattern>
</servlet-mapping>

  <context-param>
    <param-name>cargaHorariaTotal</param-name>
    <param-value>360 h/a</param-value>
  </context-param>

  <context-param>
    <param-name>maxAlunos</param-name>
    <param-value>30</param-value>
  </context-param>

  <context-param>
    <param-name>notaMinima</param-name>
    <param-value>7,0</param-value>
  </context-param>
```



Exemplo: Parâmetros de Contexto

```
package servlets.parametros.contexto;
import java.io.*; import javax.servlet.ServletException; import
    javax.servlet.http.*;

public class ParametroContextoServlet extends HttpServlet {
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
    ServletException, IOException {

    resp.setContentType("text/html"); PrintWriter out = resp.getWriter();

    out.println("<h1>Parâmetros de Contextos da Aplicação: " + this.getClass() +
        "</h1><BR><BR>");

    out.println("<b>Carga Horária:</b> " +
        this.getServletContext().getInitParameter("cargaHorariaTotal") +
        "<BR>");

    out.println("<b>Número Máximo de Alunos:</b> " +
        this.getServletContext().getInitParameter("maxAlunos") + "<BR>");

    out.println("<b>Nota Mínima para Aprovação:</b> " +
        this.getServletContext().getInitParameter("notaMinima") + "<BR>");
}

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
    ServletException, IOException {
    resp.setContentType("text/html");
    PrintWriter out = resp.getWriter();
    out.print("Este servlet não suporta requisições POST...!!!");
}
}
```



Exercício - 06

- Implementar uma aplicação Web com Servlets para uma empresa que deseja ter um cadastro da carteira de seus clientes no ramo de TI, contendo:
 - O DD (web.xml) com:
 - o mapeamento dos servlets necessários
 - parâmetros de contexto:
 - email de contato da empresa, nome do gerente da empresa, email do gerente e cnpj da empresa
 - parâmetros de inicialização dos servlets:
 - número de clientes a cadastrar por dia
 - Uma página HTML com um formulário de cadastro de empresas da área de TI, contendo:
 - cnpj, razão social, área de TI (consultoria, treinamento, desenvolvimento de software, venda de equipamentos)



Exercício – 06 cont...

- Um servlet que gera como saída:
 - os dados da empresa cadastrada
 - a quantidade de empresas que ainda falta cadastrar para chegar ao valor do parâmetro de inicialização do servlet:
 - quando chegar nesse valor, redirecionar para uma página com uma mensagem ao usuário de que o número máximo de cadastros foi alcançado!!!
 - **dica:** criar uma variável de instância (atributo) no servlet para incrementar a cada novo cadastro e comparar com o valor do parâmetro de inicialização



Servlets Java:

Conceitos Básicos

Gerenciamento de Objetos usando
Atributos (Contexto, Sessão e Requisição)



O que são atributos?

- Atributos são objetos Java que podem ser compartilhados entre os recursos de uma aplicação Web
- As 02 grandes perguntas sobre atributos:
 - Qual recurso pode acessar os atributos?
 - Quanto tempo cada atributo fica disponível para ser acessado
 - Em outras palavras:
 - Qual o escopo de cada atributo?
- Atributos **NÃO** são parâmetros!!!



Escopo de Atributos

- **Contexto:** não é *thread-safe*!
 - **Acessível por:** qualquer recurso da aplicação
 - **Por quanto tempo existe:** enquanto existir a aplicação
 - **Exemplo de uso:** número de usuários ativos
- **Sessão:** não é *thread-safe*!
 - **Acessível por:** recursos de uma sessão
 - **Por quanto tempo existe:** enquanto a sessão existir
 - **Exemplo de uso:** carrinho de compras
- **Requisição:** é *thread-safe*!
 - **Acessível por:** recurso da requisição
 - **Por quanto tempo existe:** enquanto a requisição existir
 - **Exemplo de uso:** dados de um determinado item a venda



Métodos Utilizados para Atributos

- Todos os escopos possuem pelo menos os 04 métodos a seguir:
 - `getAttribute(String)`
 - `setAttribute(String, Object)`
 - `removeAttribute(String)`
 - `getAttributeNames()`
- Os atributos são armazenados na forma de um Map, com nome e valor



Exemplo com Atributos

```
public void doPost...{  
  
    resp.setContentType("text/html");  
    PrintWriter out = resp.getWriter();  
    out.print("Os dados armazenados como atributos são: <BR><BR>");  
  
    if(req.getAttribute("nome") == null) {  
        req.setAttribute("nome" , req.getParameter("nome"));  
        req.setAttribute("idade" , req.getParameter("idade"));  
        req.setAttribute("cidade" , req.getParameter("cidade"));  
        req.setAttribute("estado" , req.getParameter("estado"));  
    }  
  
    if(req.getSession().getAttribute("qtidade") == null) {  
        req.getSession().setAttribute("qtidade", 1);  
    } else {  
        req.getSession().setAttribute("qtidade",  
            (Integer)req.getSession().getAttribute("qtidade")+1);  
    }  
  
    out.print("Nome: " + req.getAttribute("nome") + "<BR>");  
    out.print("Idade: " + req.getAttribute("idade") + "<BR>");  
    out.print("Cidade: " + req.getAttribute("cidade") + "<BR>");  
    out.print("Estado: " + req.getAttribute("estado") + "<BR><BR>");  
    out.print("Quantidade de Cadastros: " + req.getSession().getAttribute("qtidade") +  
        "<BR><BR>");  
    out.print("Muito obrigado!!!");  
}
```

Atributos de
requisição

Atributos de
sessão

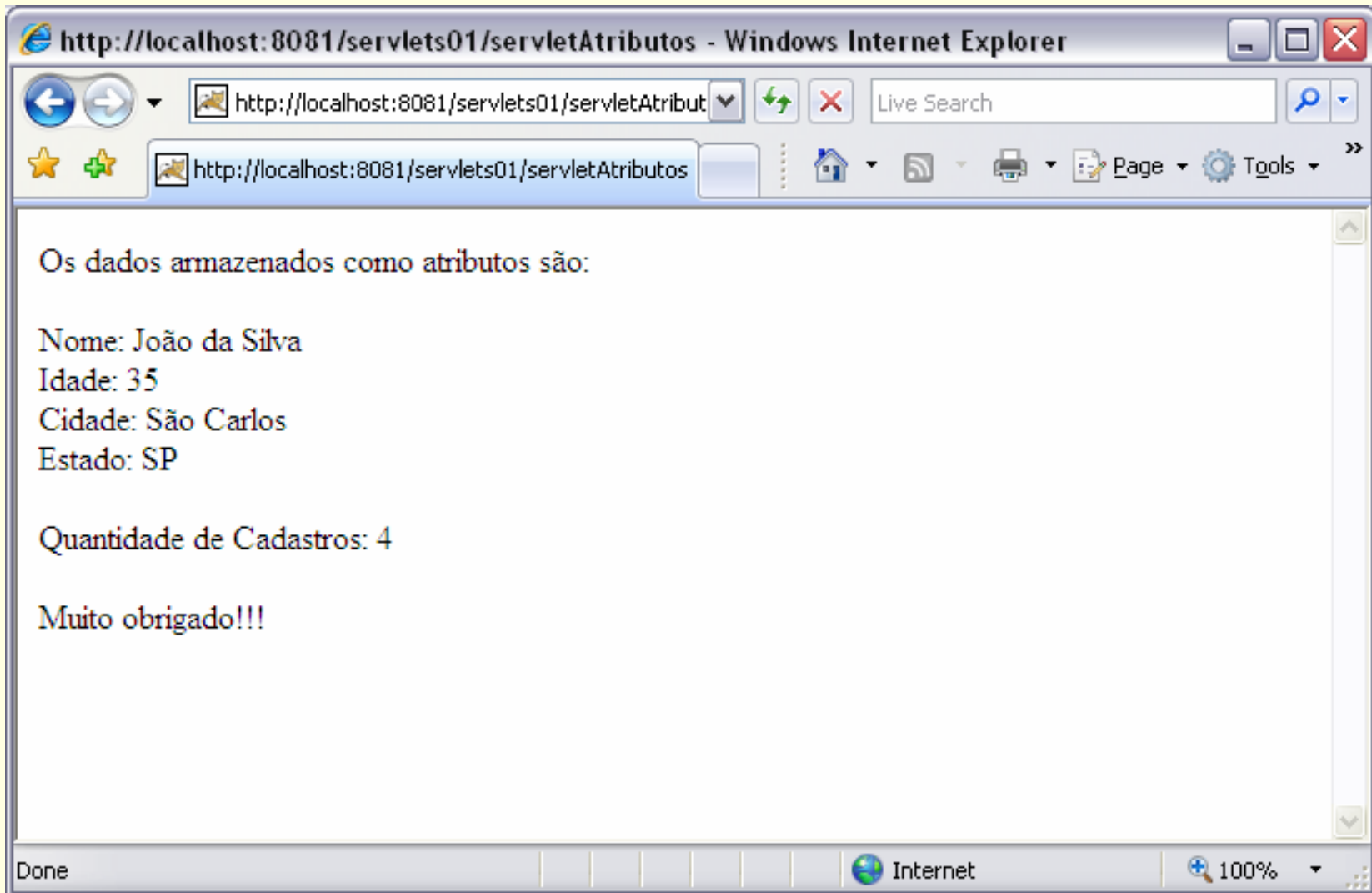


Obtendo o form_atributos.html

- Fazer o download do arquivo form_pessoa.html a partir do endereço:
 - http://www.edsonjr.pro.br/ensino/uem/form_atributos.zip



Exemplo com Atributos





Exercício - 07

- Dado um **formulário HTML**, construa um **servlet** que armazene os dados vindo desse formulário em um **Java Bean** e o **compartilhe** com outros recursos da **mesma sessão** por meio de **atributos**:
- Os dados do formulário são:
 - CPF, Nome da Pessoa, Data de Nascimento, Profissão
- O código do Java Bean está no **próximo slide**
- Este exercício envolve o uso dos seguintes conceitos já vistos:
 - Definição de servlets e mapeamento no web.xml
 - Criação de formulários em HTML
 - Envio de requisições para servlets
 - Respostas HTML a partir de servlets
 - Recuperação de parâmetros de requisição
 - Compartilhamento de objetos por meio de atributos
- **CUIDADO!!!**
 - não confundam PARÂMETRO com ATRIBUTO!!!



Exercício – 07 cont...

```
package beans;

public class Pessoa {
    private String cpf;
    private String nome;
    private String dataNascimento;
    private String profissao;

    public Pessoa() { super(); }

    public Pessoa(String cpf, String nome, String dataNascimento, String
profissão) {
        super();
        this.cpf = cpf; this.nome = nome; this.dataNascimento =
dataNascimento;
        this.profissão = profissão;
    }
    public String getCpf() { return cpf; }
    public void setCpf(String cpf) { this.cpf = cpf; }
    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }
    public String getDataNascimento() { return dataNascimento; }
    public void setDataNascimento(String dataNascimento) { this.dataNascimento =
dataNascimento; }
    public String getProfissao() { return profissao; }
    public void setProfissao(String profissao) { this.profissão = profissao; }
}
```



Exercícios - 08

- Extra-classe sobre:
 - Sistemas Web baseados em Java
 - Conceitos Básico sobre Servlets

- **OBS.: entregar na próxima aula!!!**
- **Disponíveis em:**
 - http://www.edsonjr.pro.br/ensino/uem/03_exercicios_sistemas_web_servlets.pdf