# SERVLETS E JSP (JEE) - Servlets

Diogo Cezar Teixeira Batista

diogo@diogocezar.com.br

http://www.diogocezar.com

Universidade Tecnológica Federal do Paraná - UTFPR

Cornélio Procópio - 2012

# Servlets I

- O que é Servlet?
  - é a base do desenvolvimento de aplicativos java baseados na *web*;
  - de uma maneira simples, pode ser entendida como uma classe que é automaticamente carregada na memória e posteriormente executada por um servidor *web* especial;

# Servlets II

- o servidor *web* é chamado de *container*, escolhas comuns entre *conteiners* estão:
  - Tomcat
    `http://jakarta.apache.org/tomcat/`
  - JBoss
    `http://www.jboss.org/products/index`
  - Websphere
    `http://www-306.ibm.com/software/websphere/`

# Servlets III

- arquitetura servlet:
  - o pacote `javax.servlet` provê as classes e interfaces para o desenvolvimento de *servlets*;
  - a principal abstração no pacote *Servlet* é a interface *Servlet*.
  - todas as *servlets* implementam esta interface;
  - utilização indireta da interface *Servlet*;
  - o mais comum é a herança da classe `HttpServlet`, que por sua vez implementa a interface *Servlet*;
  - a interface *Servlet* declara, mas não implementa, métodos de gerenciamento e comunicação com os clientes;

# Servlets IV

- interação com o cliente:
  - quando um *servlet* aceita uma chamada de um cliente, ele recebe dois objetos:
    - um `ServletRequest` , que encapsula a comunicação entre o cliente e o servidor;
    - um `ServletResponse` , que encapsula a comunicação entre o servidor e o cliente;
  - `ServletRequest` and `ServletResponse` são interfaces definidas pelo pacote javax.servlet.

# Servlets V

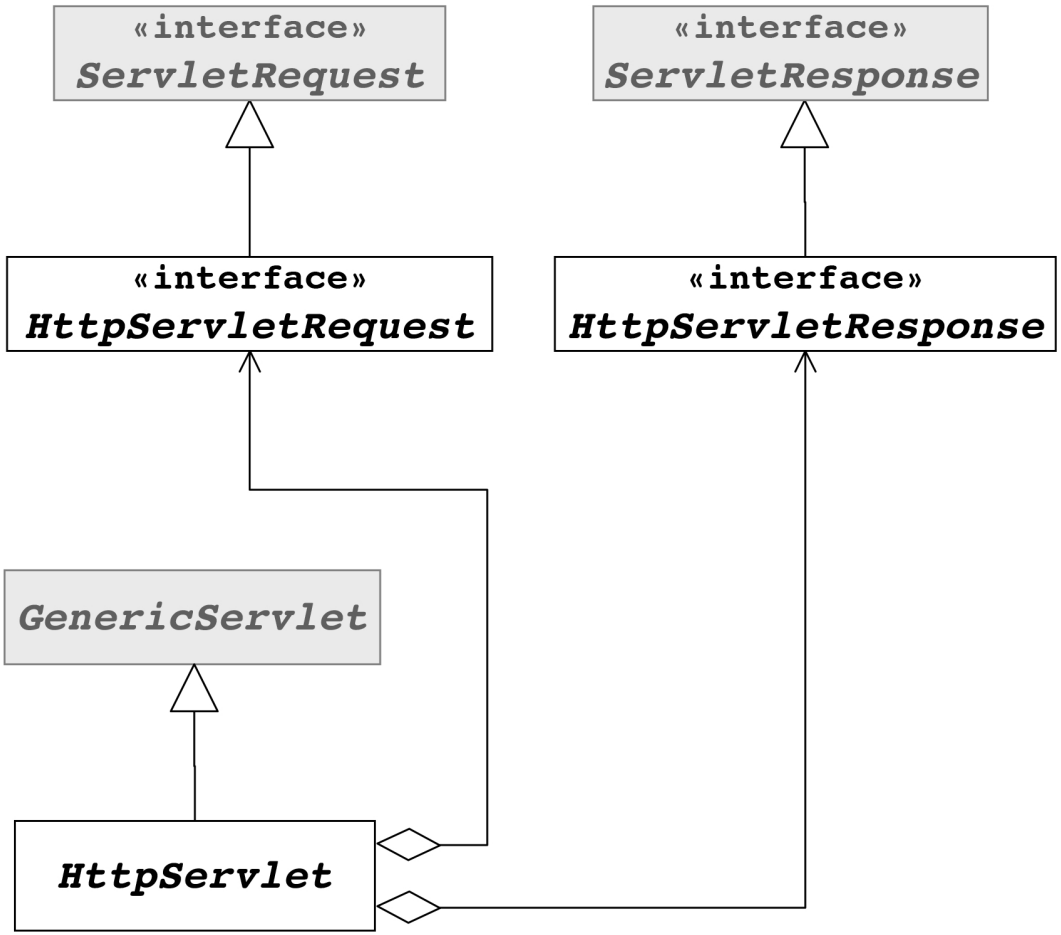
- a interface `ServletRequest`
  - a interface `ServletRequest` permite ao servlet ter acesso a:
    - informações como nomes de parâmetros passados pelo cliente, o protocolo usado pelo cliente e os nomes dos *hosts* remotos que fizeram a requisição ao servidor;
    - O *stream* de entrada, `ServletInputStream` . *Servlets* usam o *stream* de entrada para receber os dados do cliente, como é o caso do métodos HTTP `POST` e `GET`;
  - A interface `HttpServletRequest` é mais usual em ambientes HTTP;

# Servlets VI

- a interface `ServletResponse`
  - a interface `ServletResponse` permite ao *servlet* métodos para enviar ao cliente:
    - um *stream* de saída, `ServletOutputStream` , e um *Writer* através dos quais podem ser enviados dados de resposta ao cliente;
  - A interface `HttpServletResponse` é mais usual em ambientes HTTP.

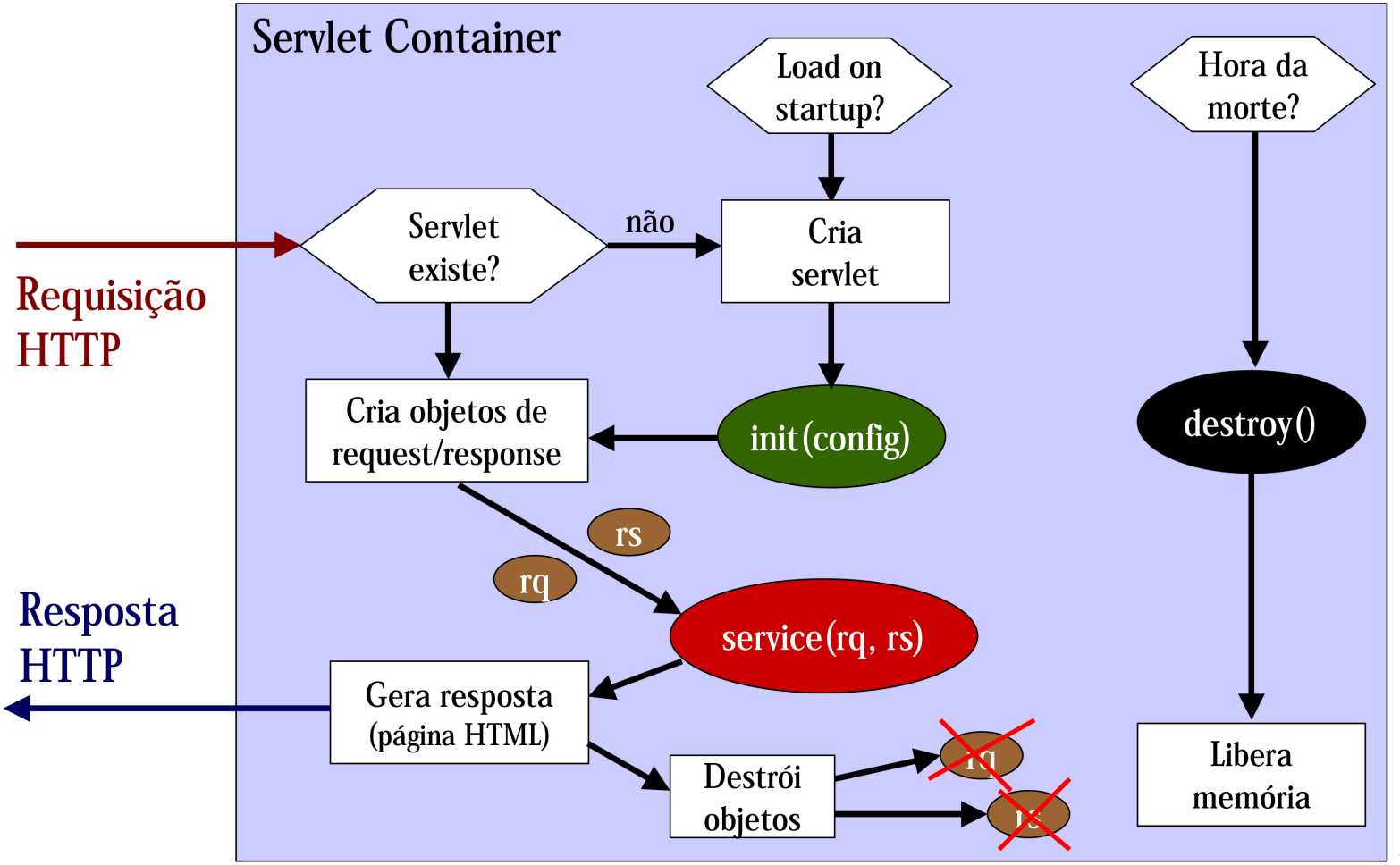
# Servlets VII

- hierarquia de classes de um *Servlet*



# Servlets VIII

- ciclo de vida de um *Servlet*



# Servlets IX

- método `init()` - chamado automaticamente pelo container após o carregamento da classe na memória;

```
public void init (ServletConfig config) throws ServletException
```

# Servlets X

- método `service()` - é chamado pelo container logo após o método `init()`, permitindo ao *servlet* responder a uma requisição.

```
  public void service (servletRequest request, Servlet response
response) throws ServletException,
java.io.Exception
```

# Servlets XI

- método `destroy()` - é chamado pelo container antes de remover a *servlet* da memória;

# Servlets XII

- como o *servlet* implementa *doGet()* e *doPost()*:

Código 1: implementado GET e POST

```
public class ServletWeb extends HttpServlet {
  public void doGet (HttpServletRequest request,
  HttpServletResponse response) {
    processar(request, response);
  }
  public void doPost (HttpServletRequest request,
  HttpServletResponse response) {
    processar(request, response);
  }
  public void processar(HttpServletRequest request,
  HttpServletResponse response) {
    ...
  }
}
```

# Servlets XIII

- como ler parâmetros de requisição:
  - formulários HTML codificam o texto ao enviar os dados automaticamente;
  - seja o método POST ou GET, os valores dos parâmetros podem ser recuperados pelo método `getParameter()` de *ServletRequest*, que recebe seu nome;

```
String parametro = request.getParameter("nome");
```

  - parâmetros de mesmo nome podem ser repetidos. Neste caso `getParameter()` retornará apenas a primeira ocorrência. Para obter todas use `String[] getParameterValues()`

```
String[] params = request.getParameterValues("nome");
```

# Servlets XIV

- como gerar uma resposta:
  - para gerar uma resposta, primeiro é necessário obter, do objeto *HttpServletResponse*, um fluxo de saída, que pode ser de caracteres (*Writer*) ou de bytes (*OutputStream*):

```
Writer out = response.getWriter(); // ou
OutputStream out = response.getOutputStream();
```

  - apenas um deve ser usado. Os objetos correspondem ao mesmo stream de dados;
  - deve-se também definir o tipo de dados a ser gerado. Isto é importante para que o cabeçalho *Content-type* seja gerado corretamente e o *browser* saiba exibir as informações:

```
response.setContentType("text/html");
```

# Servlets XV

- depois, pode-se gerar os dados, imprimindo-os no objeto de saída (*out*) obtido anteriormente:

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

# Exemplos I

Código 2: Exemplo index.jsp

```
<!DOCTYPE html>
<html>
    <head>
<meta http-equiv="Content-Type" content="text/html;
  charset=UTF-8">
<title>Teste sua Altura</title>
    </head>
    <body>
<h1>Teste a sua Altura</h1>
<p>Programa feito para testar a sua altura.</p>
<p>Esta página irá acessar um Servlet.</p>
<form method="POST" action="ServeletAltura">
    <input type="text" name="txtAltura" id="txtAltura" />
    <input type="submit" value="Enviar" />
</form>
    </body>
</html>
```

# Exemplos II

## Código 3: Exemplo ServletIdade.java

```
  protected void processRequest(HttpServletRequest request,
      HttpServletResponse response)
  throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
  float
altura = Float.parseFloat(request.getParameter("txtAltura"));
  out.println("<html>");
  out.println("<head>");
  out.println("<title>Servlet ServeletIdade</title>");
  out.println("</head>");
  out.println("<body>");
  out.println("<h1>Resultado para sua altura</h1>");
  if(altura <= 1.50){ out.println("<p>Você é baixo.</p>");}
  else if(altura <= 1.80){ out.println("<p>Você possui uma
altura mediana.</p>");}
  else {out.println("<p>Você é alto.</p>");}
  out.println("</body>");
  out.println("</html>");
      } catch(Exception e) {
  out.println("<h1>Houve um erro: " + e.getMessage() + "</h1>");
      }
      out.close();
  }
```

# Atividades

- crie um *servlet* que receba três informações:
  - nome;
  - idade;
  - data de nascimento;
- o servlet deve analisar as informações e exibir:
  - "Bom dia", "Boa tarde" ou "Boa noite" dependendo da hora de acesso do dia;
  - a idade da pessoa (tratar a *exception* se não for uma idade válida 0-150);
  - o signo da pessoa baseado em seu mês de nascimento;