Aplicações de Linguagem de Programação Orientada a Objeto

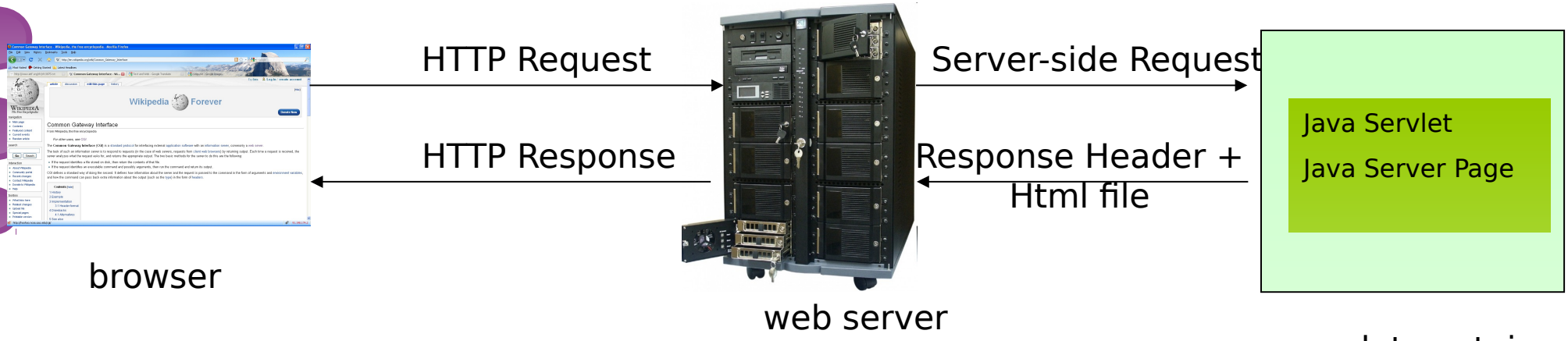
WEB
Servlets

Slides baseados no material de Daniel Arraes Pereira, cin.ufpe.br.

Professora Sheila Cáceres

Server-side Java for the web

- Um servlet é um programa Java que gera como saida uma página html.
- É uma tecnologia que roda no lado do servidor.



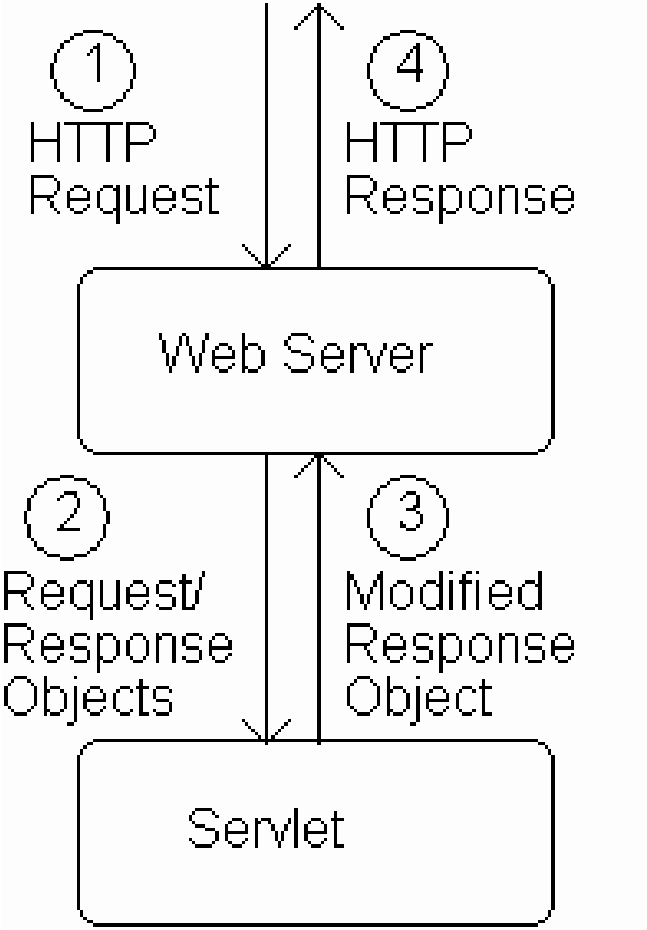
Servidor Web

- É um programa de computador responsável por:
  - aceitar pedidos HTTP de clientes (geralmente de navegadores)
  - servi-los com respostas HTTP, incluindo opcionalmente dados. Essas respostas geralmente são páginas web, tais como documentos HTML com objetos embutidos (imagens, etc.)
- Também é considerado servidor web a um computador que executa um programa que provê a funcionalidade descrita anteriormente.
- Exemplos: Servidor Apache, tomcat.

Server-side Programming

- A resposta do servidor web pode ser estatica ou dinamica.
  - Statica: O documento HTML é recuperado do sistema de arquivos e retornado ao cliente.
  - Dinamica: O documento HTML é gerado por um programa em resposta (HTTP response) a uma solicitud HTTP (HTTP request)
- O servlets Java são uma tecnologia para produzir respostas de servidor dinamicas.
- Servlet é uma classe instanciada pelo servidor para produzir uma resposta dinamica.

Servlet Overview



Servlets

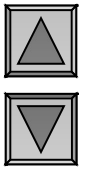
- Por que usá-los?
  - Facilidade de uso.
  - Facilidade de desenvolvimento.
  - Maturidade da linguagem Java. Servlets são classes Java.
  - Eficientes
  - Independentes de browsers.
  - Robustez, segurança, Multiplataforma, possuem praticamente toda a plataforma Java disponível.

Servlets

- Como usá-los?
  - Escreve uma classe que estenda a classe Servlet sobrescrevendo os métodos relativos ao tipos de requisição.
  - Cria a estrutura de diretórios necessária e faz o deploy do servlet em tal estrutura.
  - Realiza o mapeamento do servlet em uma(s) URL através do deployment descriptor da aplicação

Servlet Overview

1. When server starts it instantiates servlets
2. Server receives HTTP request, determines need for dynamic response
3. Server selects the appropriate servlet to generate the response, creates request/response objects, and passes them to a method on the servlet instance
4. Servlet adds information to response object via method calls
5. Server generates HTTP response based on information stored in response object



Outline

```
1  // Fig. 24.5: WelcomeServlet.java
2  // A simple servlet to process get requests.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet extends HttpServlet {
9
10    // process "get" requests from clients
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       response.setContentType( "text/html" );
16       PrintWriter out = response.getWriter();
17
18       // send XHTML page to client
19
20       // start XHTML document
21       out.println( "<?xml version = \"1.0\"?>" );
22
23       out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
24          "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
25          "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
26
```

WelcomeServlet

Lines 4-5

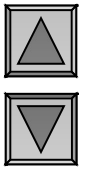
Import the `javax.servlet` and `javax.servlet.http` packages.

Extends `HttpServlet` to handle HTTP get requests and HTTP post

Override method `doGet` to provide custom get request processing.

Uses the `response` object's `getWriter` method to obtain a reference to the `PrintWriter` object that enables the servlet to send content to the client.

Create the XHTML document by writing strings with the `out` object's `println` method.



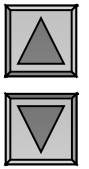
Outline

WelcomeServlet

Line 41

```
27          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
28
29          // head section of document
30          out.println( "<head>" );
31          out.println( "<title>A Simple Servlet Example</title>" );
32          out.println( "</head>" );
33
34          // body section of document
35          out.println( "<body>" );
36          out.println( "<h1>Welcome to Servlets!</h1>" );
37          out.println( "</body>" );
38
39          // end XHTML document
40          out.println( "</html>" );
41          out.close();  // close stream to complete the page
42       }
43    }
```

Closes the output stream, flushes the output buffer and sends the information to the client.



Outline

WelcomeServlet.html

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 24.6: WelcomeServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP Get Request</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp5/welcome1" method = "get">
14
15       <p><label>Click the button to invoke the servlet
16          <input type = "submit" value = "Get HTML Document" />
17       </label></p>
18
19    </form>
20 </body>
21 </html>
```

Servlets vs. Java Applications

- Servlets do not have a `main()`
  - The `main()` is in the server
  - Entry point to servlet code is via call to a method (`doGet()` in the example)
- Servlet interaction with end user is indirect via request/response object APIs
  - Actual HTTP request/response processing is handled by the server
- Primary servlet output is typically HTML

Servlet Hello World

```
public class HelloWorldServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                            throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html><head><title>Hi</title></head>");
      out.println("<body>");
      out.println("Hello World!!");
      out.println("</body>");
      out.println("</html>");
      out.close();
   }
}
```

Servlet Hello World - Deploy

- webapps
  - ROOT – Contexto principal. URL default.
  - [Nome do contexto]
    - Arquivos JSP
    - META-INF
    - WEB-INF
      - **classes** – Classes da aplicação (Servlets, ...)
      - **lib** – bibliotecas específicas da aplicação. *.jar.
      - **Deployment Descriptor (web.xml)**

Servlet Hello World

Deployment Descriptor (web.xml)

- Arquivo que faz o mapeamento entre URLs e Servlets além de configurações de segurança, eventos, filtros, ...

Servlet Hello World
Deployment Descriptor (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" 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">
<display-name>HelloWorld</display-name>
<servlet>
<description></description>
<display-name>HelloWorldServlet</display-name>
<servlet-name>HelloWorldServlet</servlet-name>
<servlet-class>HelloWorldServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HelloWorldServlet</servlet-name>
<url-pattern>/HelloWorldServlet</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>index.htm</welcome-file>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Servlet Hello World

- Executando
  - http://localhost:8080/HelloWorld/HelloWorldServlet

Servlets e requisições

- Suportam todos os tipos de requisições. Mapeadas em chamadas de métodos na instancia do Servlet em questão.
  - GET – doGet(...)
  - POST – doPost(...)
  - PUT – doPut(...)
  - DELETE – doDelete(...)
  - TRACE – doTrace(...)
  - HEAD – doHead(...)
  - OPTIONS – doOptions(...)

Servlets e requisições

Características	GET	POST
Tipo de dados	Texto	Texto ou binário
Quantidade de dados	Máximo de 255 caracteres	Ilimitado
Visibilidade	Dados fazem parte da URL e podem ser **vistos**	Dados não fazem parte da URL e sim do corpo da mensagem. Não podem ser vistos

Servlets e requisições

- Exemplos de URL usando Get e Post
  - GET

    http://www.cin.ufpe.br/servlet/ServletProcurar?numero=12&codigo=1
  - POST

    http://www.cin.ufpe.br/servlet/ServletProcura

Servlets e requisições: analisando

- Interface **ServletRequest**
  - Provê métodos de acesso ao conteúdo da requisição que são relevantes a qualquer protocolo.
  - Pacote javax.servlet.*;
- Interface **HttpServletRequest**
  - Extende ServletRequest
  - Provê métodos relativos ao protocolo HTTP.
  - Pacote javax.servlet.http.*;

ServletRequest

Método	Descrição
String **getParameter**(String paramName)	Retorna o valor associado ao determinado parâmetro.
String[] **getParameterValues**(String name)	Retorna todos os valores associados ao determinado parâmetro. (List boxes, Check boxes)
Enumeration **getParameterNames**()	Retorna os nomes dos parâmetros passados na requisição.

HttpServletRequest

Método	Descrição
String **getHeader**(String headerName)	Retorna um dos valores associados ao Header.
Enumeration **getHeaderValues**(String headerName)	Retorna todos os valores associados ao Header
Enumeration **getHeaderNames**()	Retorna os nomes dos Headers.

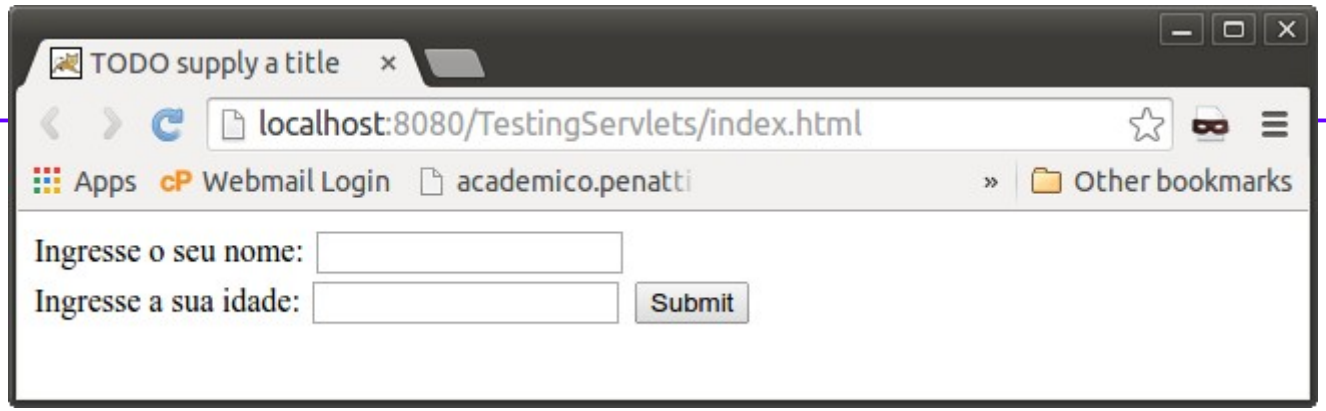
Request Headers example :

```
"headers" : {
        "Host" : "mkyong.com",
        "Accept-Encoding" : "gzip,deflate",
        "X-Forwarded-For" : "66.249.x.x",
        "X-Forwarded-Proto" : "http",
        "User-Agent" : "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)",
        "X-Request-Start" : "1389158003923",
        "Accept" : "*/*",
        "Connection" : "close",
        "X-Forwarded-Port" : "80",
        "From" : "googlebot(at)googlebot.com"
}
```

Exemplo

Envio de uma requisição desde html

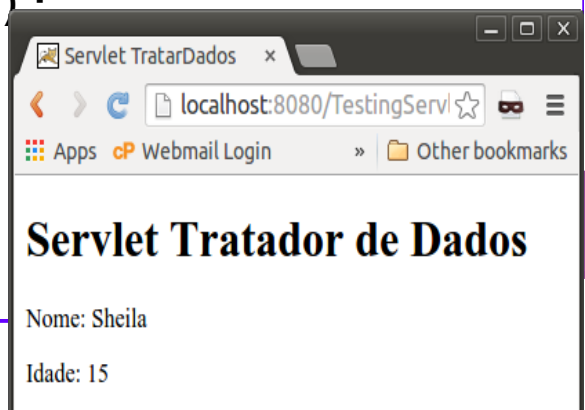
```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    </head>
    <body>
        <form method="POST" action="tratador">
            Ingresse o seu nome: <input type="text"
name="nome"><br />
            Ingresse a sua idade: <input type="text" name="idade">
            <input type="submit">
        </form>
    </body>
</html>
```



Tratamento da requisição

```
public class TratarDados extends HttpServlet {
    protected void doGet(HttpServletRequest request,
  HttpServletResponse response)throws ServletException, IOException{
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            String nome = request.getParameter("nome");
            String idade = request.getParameter("idade");
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet TratarDados</title>");

            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Tratador de Dados </h1>");
            out.println("<p>Nome: "+nome+" </p>");
            out.println("<p>Idade: "+idade+" </p>");
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```



Servlets e requisições: Exemplo 2

Dentro do servlet

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

    String nome = request.getParameter("name");
    int idade = Integer.parseInt(request.getParameter("age"));
    fachada.cadastrarCliente(new Cliente(nome, idade));
    response.sendRedirect("sucesso.jsp");
}
```

Servlets - Enviando respostas

- Interface **ServletResponse**
  - Provê métodos de resposta que são relevantes a qualquer protocolo.
  - Pacote javax.servlet.*;
- Interface **HttpServletResponse**
  - Extende ServletResponse
  - Provê métodos relativos ao protocolo HTTP.
  - Pacote javax.servlet.http.*;

ServletResponse

Método	Descrição
PrintWriter **getWriter**()	Retorna um java.io.PrintWriter, que pode ser usado para enviar texto ao cliente.
ServletOutputStream **getOutputStream**()	Retorna um javax.servlet.ServletOutputStream, que pode ser usado para enviar dados binários ao cliente.
void **setContentType**(String type)	Usado para ajustar o tipo do conteúdo da resposta. Ex: “text/html”, “image/jpeg”, “application/jar”

HttpServletResponse

Método	Descrição
void **setXXXHeader**(...)	Seta pares nome / valor para o Header. (String, Inteiros, Datas)
boolean **containsHeader**(String name)	Retorna se um header com este nome já está setado.
void **sendRedirect**(String location)	Redireciona a chamada à URL especificada.

Servlets – Enviando Repostas: Exemplo

```
protected void doGet(HttpServletRequest req,
                     HttpServletResponse res)
                     throws ServletException, IOException
                                                          {
    res.setContentType("application/jar");
    File f = new File("test.jar");
    byte[] bytearray = new byte[(int) f.length()];
    FileInputStream is = new FileInputStream(f);
    is.read(bytearray);
    OutputStream os = res.getOutputStream();
    os.write(bytearray);
    os.flush();
}
```

Bibliografia

- Material de Daniel Arraes Pereira, cin.ufpe.br.
- Slides de Guy-Vincent Jourdan, Universidade de Ottawa, Faculdade de Engenharia, Canada.