

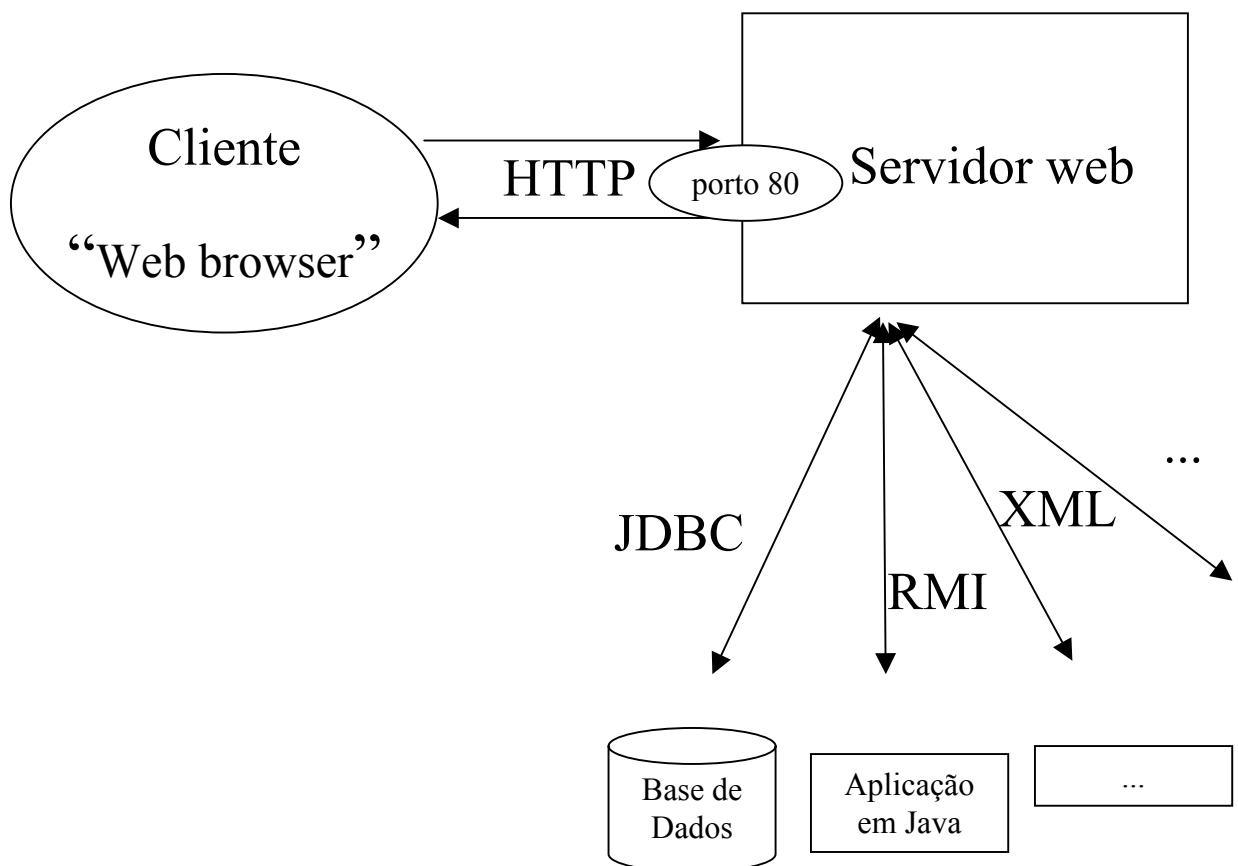
Capítulo VII – Aplicações para a Web

“Web Browser” como o processo cliente

Servidor web com páginas estáticas

Vs

Aplicações dinâmicas para a Web:



Capítulo VII – Aplicações para a Web

Páginas dinâmicas são usadas quando queremos:

- que haja interacção com o utilizador
- interacção com bases de dados
- interacção com outras aplicações

Páginas dinâmicas podem ser construídas através:

- CGI.s (Common Gateway Interface)

Consiste num programa, colocado no servidor, que é invocado numa página Web.

- cada invocação (por um cliente) dá origem à criação de um sub processo no servidor
- podem ser escritos em qualquer linguagem suportada pelo servidor, por exemplo, C, Perl, shell scripts, ...

Capítulo VII – Aplicações para a Web

- Linguagens de “scripting”

-Do lado do cliente:

-Geração de conteúdo dinâmico do lado do cliente.
Não podem aceder a recursos do servidor.

- . JavaScript

- . VB Script

-

-Do lado do servidor:

- PHP

Tecnologias:

- . ASP (Microsoft Active Server Pages),

- . Servlets e JSP (Java Server Pages)

-

Capítulo VII – Aplicações para a Web

Servlets

- . Uma servlet é um componente web que é processado por um “Container/Engine” e que vai gerar o conteúdo dinâmico de uma página web.
- . Actua como uma camada intermédia entre um pedido do cliente HTTP e aplicações ou bases de dados do lado do servidor
- . Para cada pedido do cliente é gerado uma nova Thread.
- . Uma servlet é carregada para memória apenas uma vez, permanecendo entre os vários pedidos.
- . Assim, é eliminado o “overhead” de criação do objecto e permite a persistência de recursos do servidor usados pela servlet (por exemplo uma ligação a uma base de dados).

Alguns contentores de Servlets: (“Servlets engines”)

Tomcat (<http://jakarta.apache.org/tomcat>)

JavaServer Web Development Kit (JSWDK)
<http://java.sun.com/products/servlet/>

Blazix (<http://www.blazix.com/>)

Capítulo VII – Aplicações para a Web

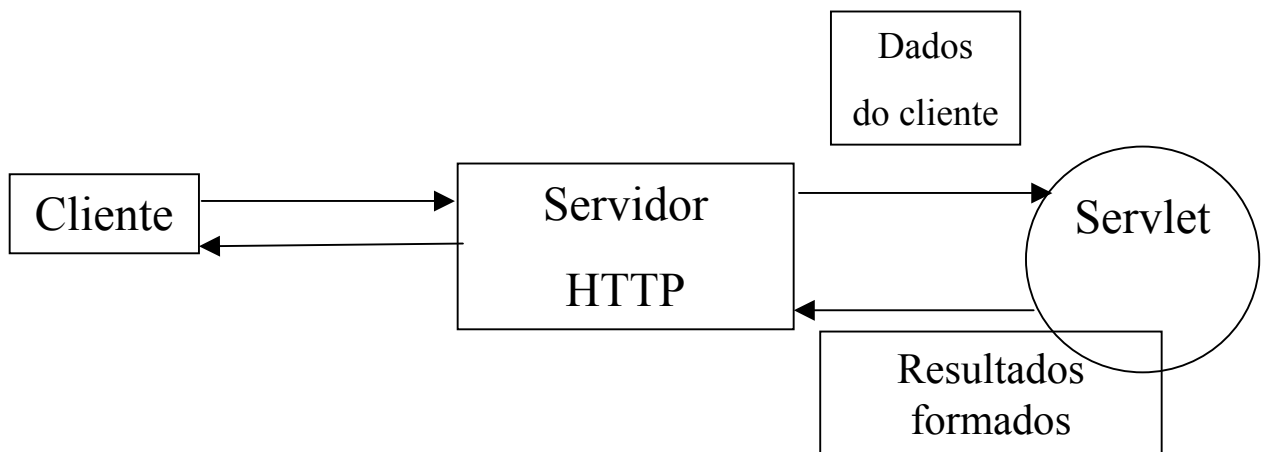
Servlets Vs CGI

Servlets têm melhor performance do que CGI's

Se existem N pedidos simultâneos a um mesmo CGI, o seu código é carregado para memória N vezes.

Com uma Servlet vão existir N threads, mas apenas uma cópia da classe da Servlet.

Arquitectura Servlet – Servidor Web



A interface Servlet

(javax.servlet.Servlet)

- Define um método para manipular os pedidos dos clientes:

```
public void service ( ServletRequest rep,  
                    ServletResponse res) throws  
ServletException, IOException;
```

Capítulo VII – Aplicações para a Web

A classe `GenericServlet` implementa a interface anterior e a classe abstracta `HttpServlet` é subclasse de `GenericServlet`.

Para criarmos uma Servlet podemos construir uma subclasse de `HttpServlet` e redefinir (sobrepor) os métodos `doGet` ou `doPost` (ou ambos).

```
public class PrimeiraServlet extends HttpServlet{

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

        PrintWriter out = response.getWriter();
        out.println( "Hello World");
    }
}
```

A interface `HttpServletRequest`,

define métodos que podem ser usados para obter informação sobre o pedido do cliente.

A interface `HttpServletResponse`,

representa a resposta para o cliente. Permite aceder a uma stream de output, `PrintWriter`, para envio de dados para o cliente.

Capítulo VII – Aplicações para a Web

Para que o processo cliente possa visualizar a resposta, em formato HTML, termos de:

- informar o browser de qual o formato em que é enviada a resposta,
- modificar a informação enviada pela `PrintWriter` para que represente HTML válido.

```
public class PrimeiraServlet extends HttpServlet{

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

        PrintWriter out = response.getWriter();
        String type= "<!DOCTYPE HTML PUBLIC ... "
        out.println( type + "<HTML>\n"+
                      "<BODY>" + "Hello world"+
                      "</BODY>" +
                      "</HTML>");
    }
}
```

Pode não ser muito produtivo, gerar HTML em instruções `println` ...

Uma solução é usar Java Server Pages (JSP)

Capítulo VII – Aplicações para a Web

Java Server Pages (JSP)

Tecnologia que permite misturar HTML estático, com HTML gerado dinamicamente.

“Tags” JSP são traduzidas para o código de uma servlet que fornecerá conteúdo dinâmico.

Essa tradução é efectuada a primeira vez que ocorre um acesso à página e quando a página é modificada.

Por sua vez também a Servlet é compilada apenas quando ocorre o primeiro acesso à página e quando esta é modificada

Sintaxe do JSP:

- . Elementos de scripting

Permitem inserir código na servlet que vai ser gerada pela página JSP (expressions, scriptlets, declarations)

- . Directivas

Mensagens para o motor de páginas JSP

- . Acções standard

Permitem instanciar objectos e comunicar com os recursos do servidor.

Capítulo VII – Aplicações para a Web

Expressões JSP

`<%= Expressão Java %>`

Permitem inserir o valor calculado pela expressão em código Java no output de uma servlet.

Por exemplo, criar o ficheiro random.jsp:

`<HTML>`

`<BODY>`

Número aleatório: `<%= Math.random()*100 %>`

`</BODY>`

`</HTML>`

-o conteúdo da expressão é convertido para uma String (ou é gerada uma exceção)

Scriptlets

Com scriptlets podemos incluir porções de código entre `<% ... %>`

O código da scriptlet será executado cada vez que a página JSP for acedida.

- . Podemos ter várias linhas de código
- . Alternativas,
- . Ciclos,

...

Capítulo VII – Aplicações para a Web

Exemplo:

```
<HTML>
```

```
<BODY>
```

```
<% System.out.println(“Calcular data do sistema”);
```

```
// O output desta instrução será escrito no ficheiro de log do servidor
```

```
// Esta scriptlet declara e inicializa a data:
```

```
Java.util.Date date = new java.util.Date();
```

```
%>
```

```
Hello! The time is <%= date %>
```

```
</BODY>
```

```
</HTML>
```

Variáveis pré definidas para utilização em scriptlets:

- request

(do tipo javax.servlet.http.HttpServletRequest)

ex.lo: <%=request.getRemoteHost() %>

- response

(do tipo javax.servlet.http.HttpServletResponse)

-out

(do tipo javax.servlet.jsp.JspWriter)

. Stream para enviar o output para o cliente

Capítulo VII – Aplicações para a Web

A scriptlet não gera código HTML,
para isso é necessário usar a variável out.

Exemplo:

```
<HTML>
<BODY>
<%
String cliente = request.getRemotetHost();
if ( cliente.equals("alunos.di.ubi.pt") )
    out.println ( "Olá, aluno do DI - UBI");
else
    out.println ("Quem és tu?");
%>
</BODY>
</HTML>
```

...

Ver “tutorial” sobre JSP

<http://www.jsptut.com>