Tecnologias Web

/

Arquitectura cliente-servidor

Objectivos da aula

- Dar uma visão geral do Web e das suas tecnologias
- Abordar as arquitecturas e protocolos de comunicação
- Diferenciar as tecnologias cliente vs servidor
- Apresentar um largo espectro de tecnologias (apesar da constante evolução e aparição de novas tecnologias)

Plano

- Apresentação da internet e WWW
  - História da internet
  - W3C
- Arquitectura e software para as redes
  - *Peer to peer*
  - Clientes-Servidores
  - DNS
  - Browsers
  - Proxy
- Tecnologias do lado do cliente
  - HTML
  - CSS
  - Javascript
  - Applet
  - ActiveX
- Tecnologias do lado do servidor
  - Cookies
  - CGI
  - Servlet / JSP
  - PHP
  - ASP / ASP.NET

História da internet

- 1962: estudo para a criação de uma rede para controlar a infra-estrutura americana, capaz de resistir a um ataque nuclear.
- 1969: construção da primeira rede física (4 máquinas, 50kbps).
- 1972: envio do primeiro e-mail (23 máquinas, 50kbps).
- 1973: criação do TCP/IP.
- 1974: primeira utilização do termo internet.
- 1982: TCP/IP torna-se o standard da internet.
- 1983: criação do Domain Name System (DNS).

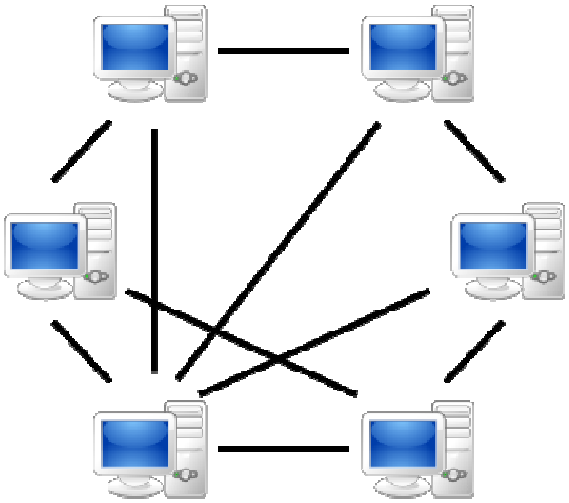
- 1990: criação de um sistema hipertexto por Tim Berners-Lee.
- 1992: o CERN apresenta o World Wide Web.
- 1993: criação de Mosaic, o primeiro browser.
- 1994: criação do W3C.
- 1995: IE 1.0
- 1997: IE 4.0, Netscape tem 72% e IE tem 18%.
- 1998: compra de Netscape por AOL.
- 2002: Mozilla 1.0
- 2004: início da 2ª guerra dos browsers.

W3C

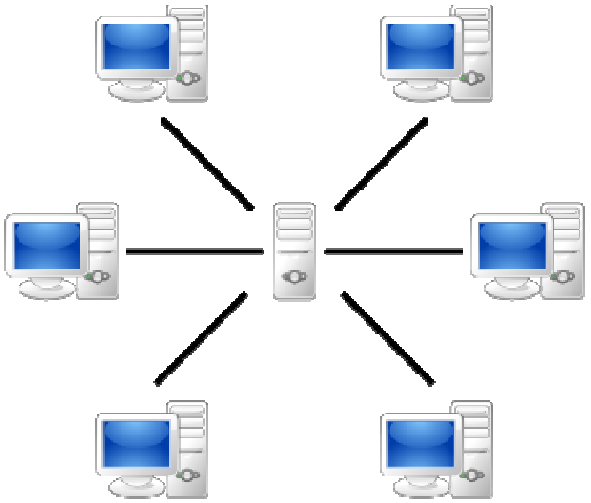
- World Wide Web Consortium.
- 3 objectivos:
  - Acesso universal à Web através de tecnologias que tomam em conta as ≠ culturas, línguas, deficiências…
  - Web semântica.
  - Web of trust: guia para desenvolver a Web tomando em conta os aspectos legais, comerciais e sociais ligados às novas tecnologias.
- Papel do W3C:
  - Interoperabilidade: as especificações dos protocolos e linguagens da Web devem poder funcionar em conjunto.
  - Evolução: assegurar-se de que novas tecnologias possam ser acrescentadas à Web.
  - Estandardização: dar *recomendações* que descrevem as tecnologias da Web.

Arquitecturas e softwares para as redes

Peer to Peer

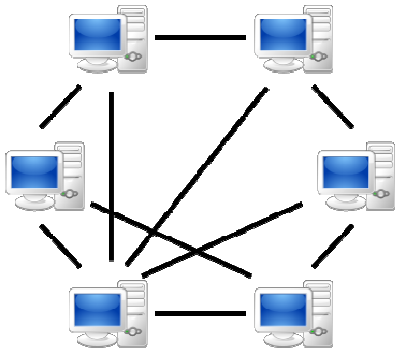


Clientes-Servidor



I. Arquitectura Peer to Peer

- Sem computador centralizado.
- Cada computador tem o papel de cliente e servidor.



- Desvantagens:
  - A falta de centralização deixa o sistema difícil de administrar.
  - Fraca segurança.
  - Nenhum elo do sistema é fiável.
- Vantagens:
  - Custos reduzidos.
  - Simplicidade de instalação.

II. Arquitectura cliente-servidor

☞2 partes distintas:
- Cliente
- Servidor

☞Analogia:
- Consumidor
- Fornecedor

II.1. O cliente

- Está conectado a uma rede.
- Utiliza os serviços de 1 ou vários servidores.
- Pede a execução de 1 ou várias tarefas.
- 2 formas:
  - computador
  - programa
- Recupera os resultados do servidor.
- Oferece uma interface utilizador convivial.
- O utilizador está situado ao nível do cliente.

II.2. O servidor

- Está conectado à rede.
- Coloca serviços à disposição do cliente.
- Tem que cumprir tarefas.
- 2 formas:
  - computador
  - programa
- Papéis:
  - responde às solicitações do cliente.
  - executa as tarefas pedidas pelo cliente.
- Pode tratar vários pedidos simultâneos.

II.3. Comunicação cliente-servidor

- Diálogo entre processos 2 a 2.
- Resultado: troca de dados.
- O cliente inicia o diálogo.
- O servidor é perpetuamente em espera de um eventual pedido.

Pedido

Espera

Realização/Execução

Recepção

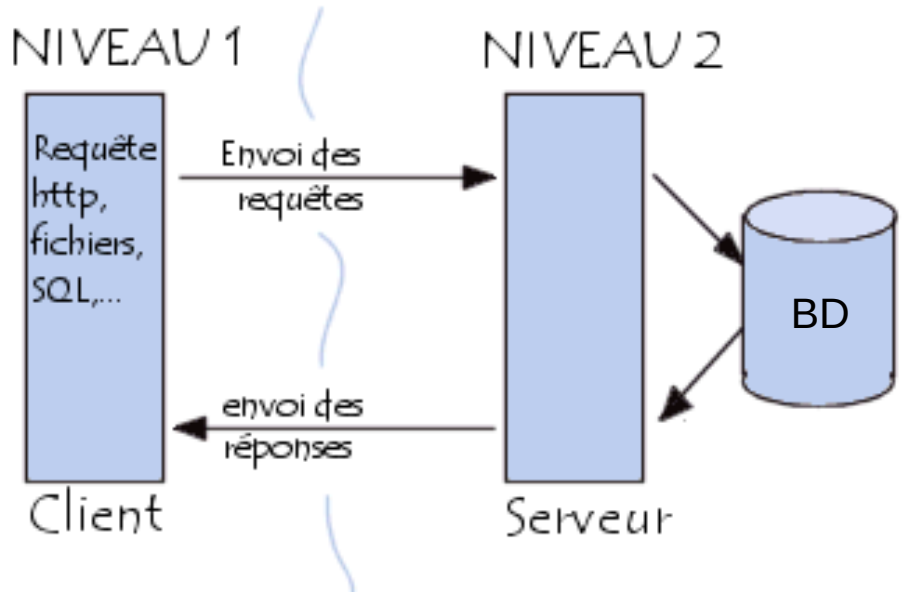
Envio

II.4. Configurações cliente/servidor

- **Vantagens:**
  - Os clientes vêem apenas o servidor.
  - Recursos centralizados no servidor: => distribuição de recursos comuns a todos os utilizadores (ex: database centralizada).
  - Melhora segurança: limitação dos pontos de acesso aos dados.
  - Administração ao nível do servidor: a administração ao nível do cliente é secundária.
  - Rede evolutiva: possibilidade de juntar ou eliminar clientes sem alterar a rede.
- **Limites:**
  - Custo elevado: alta tecnicidade de administração do servidor.
  - Elo fraco: problema no servidor => rede abaixo.

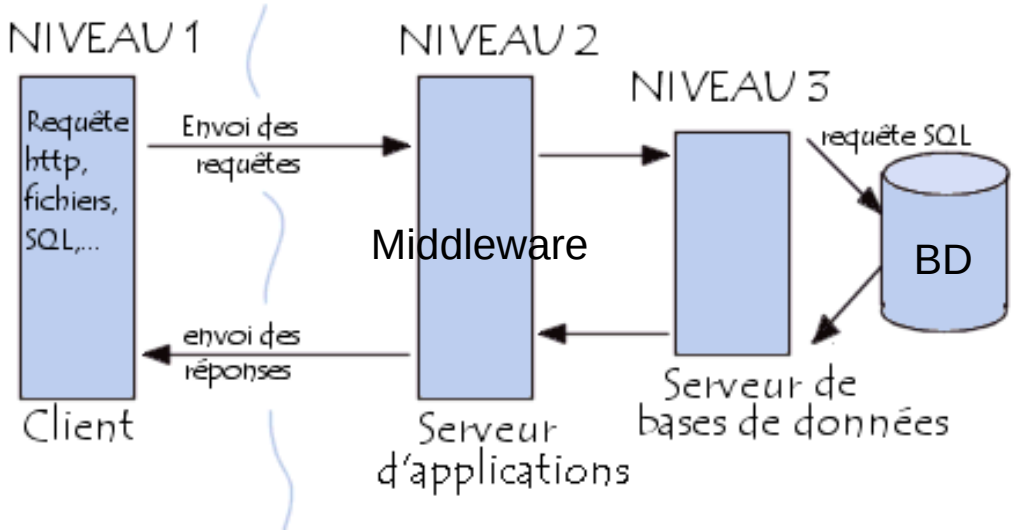
☞ **Arquitectura a 2 níveis (ou 2-tier):**

- = modelo clássico.
- o servidor fornece <u>directamente</u> o recurso pedido pelo cliente.
- => o servidor não chama uma outra aplicação para fornecer o serviço.



☞ **Arquitectura a 3 níveis (ou 3-tier):**

- nível 1 = o computador cliente pede um recurso.
- nível 2 = o servidor de aplicação (= middleware) fornece o recurso graças ao contacto com o servidor de BD.
- nível 3 = o servidor de BD fornece ao middleware os dados que necessita.



☞ **Comparação entre os 2 tipos de arquitectura:**

- Arquitectura 2-tier:

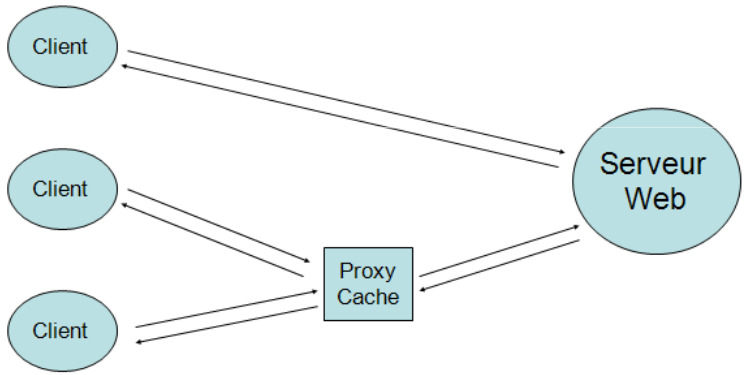
- servidor polivalente.
- => servidor capaz de fornecer todos os recursos pedidos pelo cliente.

- Arquitectura 3-tier:

- especialização de cada servidor numa tarefa determinada.
- => maior flexibilidade.
- => melhor configuração da segurança segundo o nível e o serviço.
- => melhor desempenho dado a repartição das tarefas.

II.5. Funcionamento dos servidores Web

- Armazenamento de páginas Web.
- Espera em permanência os pedidos do cliente.
- Sequência de comunicação:



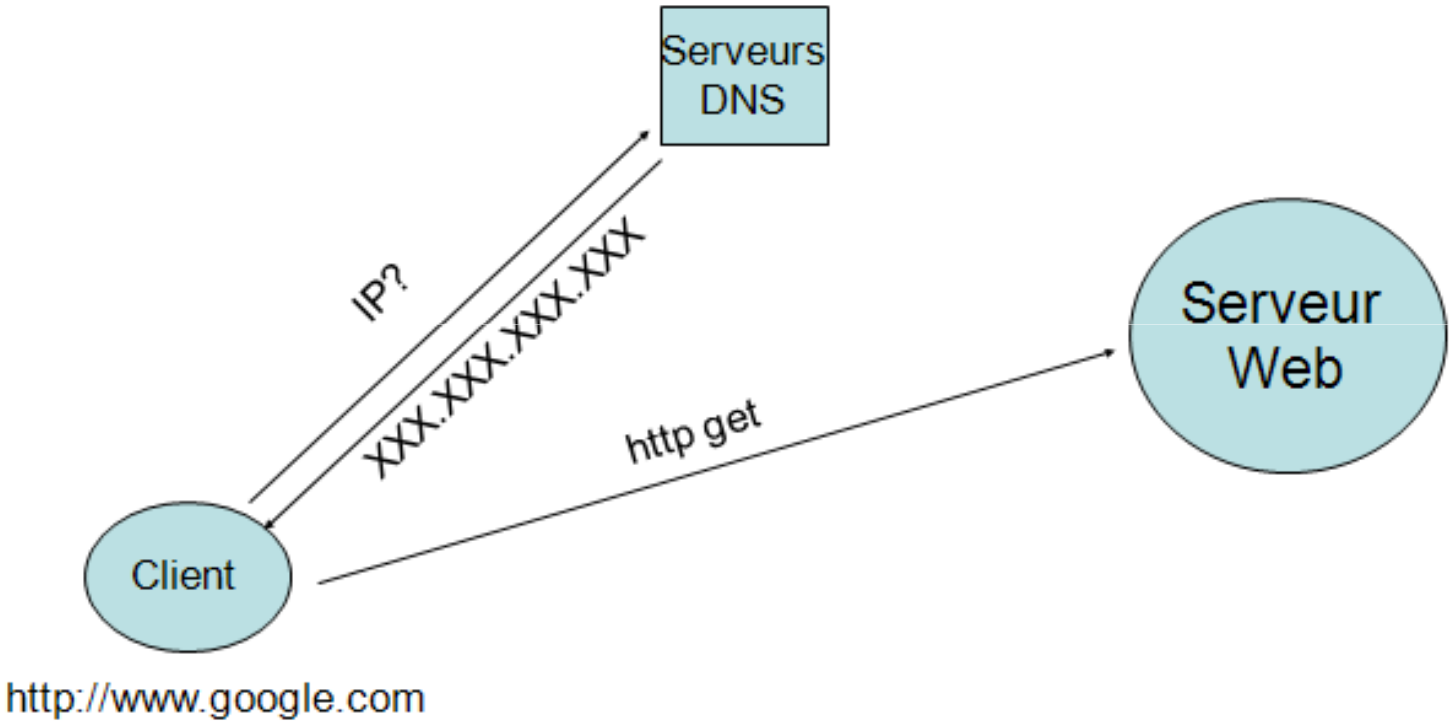
1) Cliente:
   - pede a página Web.
   - entra o endereço URL no seu browser
     = protocolo (http) / pedido de serviço.

2) Servidor:
   - recepção do request.
   - tratamento = pesquisa do código da página.
   - envio da página Web.

3) Cliente:
   - recepção do código (HTML, CSS, …).
   - interpretação e apresentação do resultado no ecrã.

II.5.1. Nome de domínio

- É mais fácil lembrar-se de uma série de palavras do que de uma série de números.
- Os nomes de domínios são uma sequência alfanumérica.
- Cada segmento tem um tamanho arbitrário (mas limitado).
- O segmento final é normalizado: .com, .edu, .gov, .org, …
- A transformação do nome de domínio em endereço IP é realizado através da conexão ao servidor de domínio (DNS).
- Cada instituição pode instalar o seu servidor DNS.



- Para comunicar, um cliente tem que conhecer o IP do servidor e o número do porto.
- Este endereço é obtido através do servidor DNS.

II.5.2. Os browsers

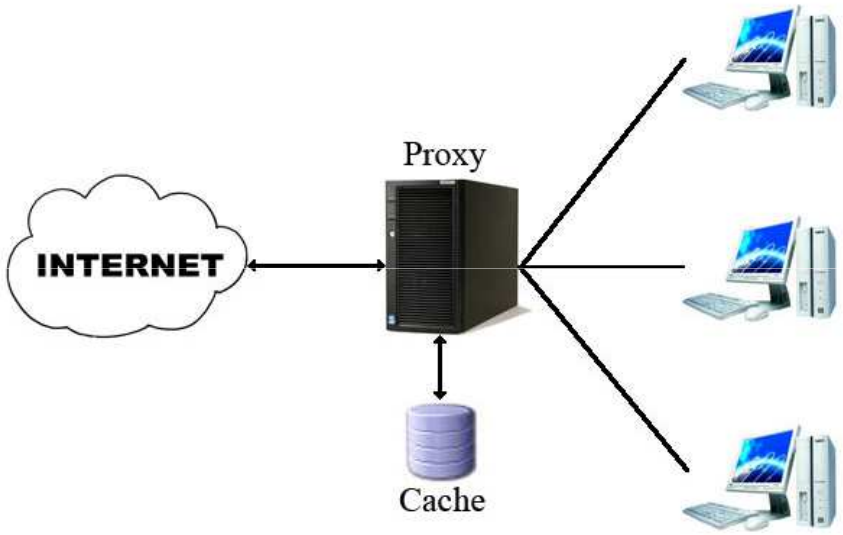
- ≠ plataformas:
  - Windows
  - Linux
  - Mac
  - …
- ≠ softwares:
  - IE
  - Firefox
  - Safari
  - …
- Características comunas:
  - Gestão do HTML/XHTML
  - Gestão do CSS
  - Gestão de Javascript
  - Gestão de plug-in
  - …

II.5.3. Os servidores Web

- São programas que respondem aos pedidos de clientes Web.
- Também chamados servidores http.
- Escutam no porto 80 (convenção) da máquina.
- 2 tipos de páginas:
  - Estáticas: sem tratamento do lado do servidor.
  - Dinâmicas: necessidade de acrescentar operações específicas.
- Números de softwares disponíveis:
  - Apache
  - Internet Information Service (Microsoft©)
  - Oracle iPlanet Web Server (novo Sun Java System Web Server)
  - …

II.5.4. O proxy (*cache*)

- Porque pedir várias vezes a mesma coisa ao servidor?
- Certos clientes Web têm um *cache* personalizado para o utilizador. O proxy trabalha ao nível do domínio (= conjunto de clientes).
- Os clientes pedem ao proxy, depois o proxy pede ao servidor:
  - Melhora a reactividade.
  - ↓ a carga do servidor.
  - ↓ a utilização da largura de banda.
- Papel do proxy-cache:
  - Manter em memória (*caching*) as páginas recentemente pedidas.
  - Filtrar o acesso Web (em função dos sites, pessoas, …).



II.5.5 Páginas estáticas vs dinâmicas

☞ **Páginas estáticas:**

- Código HTML “pré-formado” armazenado no servidor.

- Código enviado ao cliente, sem efectuar de tratamentos.

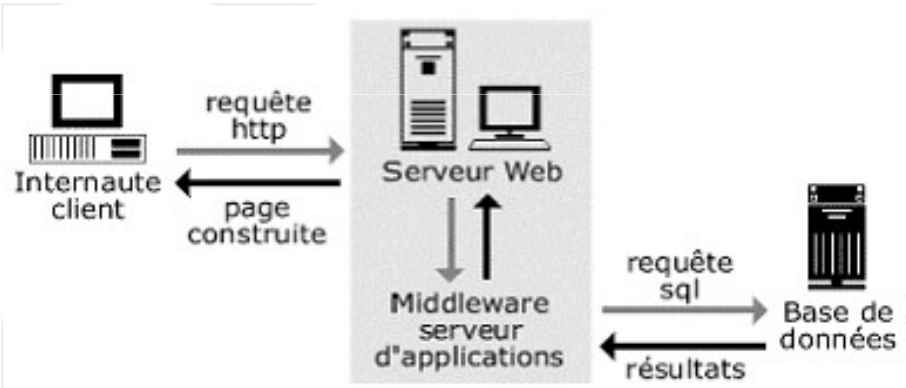
- Página idêntica qualquer que seja:
  - o cliente
  - o momento

- Exemplo: frontpage do site de uma empresa.

☞ Páginas dinâmicas:

- Utilizam outras linguagens além do HTML/CSS:
  - PHP,
  - Java (applet, servlet…),
  - …
- Arquitectura:

Cliente → (http request) → servidor Web → (interrogação da BD – sql request) → construção da página Web.



- Dados armazenados numa BD => são dissociados do código que restitui as informações ao browser.
- Permitem diferenciar a apresentação:
  - Em função do momento (e.g., apresentação da date actual).
  - Em função do cliente (e.g., língua diferente segundo o país do cliente).

☞ 2 maneiras para introduzir o dinamismo

- Do lado do servidor:

- Recepção do request do cliente.
- O servidor realiza o tratamento.
- Envio do código correspondente.
- Vantagens:
  - → independência em relação ao cliente
  - → descarrega o cliente
- Inconveniente:
  - interactividade limitada

- **Do lado do cliente:**
  - Recepção do request do cliente.
  - Envio dos elementos ao cliente (código html + ficheiros de class + …).
  - O cliente realiza o tratamento.
  - Vantagens:
    → maior interactividade
    → descarrega o servidor
  - Inconveniente:
    dependência em relação ao cliente (e.g., o browser tem que suportar certas linguagens)

Tecnologias da Web

HTML/XHTML

CSS

Javascript

ActiveX

Applet

CGI

Servlet

ASP

JSP

PHP

Diálogo

Cliente

Servidor

Tecnologias da Web

Cliente side…

☞ **HTML**

- HyperText Markup Language.
- Linguagem não proprietário.
- A língua franca para publicar hipertextos na WWW.
- Derivado do SGML (*Standard Generalized Markup Language*).
- Mistura estrutura e apresentação.
- Usa tags (=> facilita a integração de sistemas):

  <tag> … </tag>
- Agora XHTML (o seu sucessor).

- Estrutura de uma página XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt">
  <head>
    <title>Bem-vindo no meu site !</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  </head>
  <body>
  </body>
</html>
```

1) DOCTYPE: indique ao browser que o ficheiro é uma página XHTML e a versão da linguagem.
2) 1º mark html: tem atributos para indicar, nomeadamente, em que língua está escrita a página.
3) head marks: indique, nomeadamente, o título da página (<title>).
4) body marks: entrar as infos contidas na página Web.

☞ CSS

- Cascading Style Sheets.
- Descreve como um documento é apresentado no ecrã.
- Permite acrescentar um estilo à página Web.
- As folhas de estilo são colocadas dentro do tag <head>:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt">
  <head>
    <title>Exemplo de utilização de um ficheiro externo CSS</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <link rel="stylesheet" media="screen" type="text/css" title="Design"
    href="design.css" />
  </head>
  <body>
    <p>
      ...
    </p>
  </body>
</html>
```

Localização do ficheiro .css (aqui, na mesma pasta do que o .html)

☞ **Javascript**

- Linguagem de script:
  - linguagem interpretada
  - orientada pelo objecto
  - sintaxe próxima do C e do Java
- Código inserido no código html da página.
- Gere os eventos principais do rato e do teclado.
- A página é enviada e, depois, interpretada pelo cliente.
- Exemplo das potencialidades do Javascript:
  - a página personalizada de Google.
  - => possibilidade de mexer os elementos da página sem ter que refrescá-la (é tratada pelo browser).

- Tipos, variáveis, funções:
  - 5 tipos de base:
    - → Strings
    - → Números
    - → Booleanos
    - → Objectos
    - → Funções
  - Declaração das variáveis (não são diferenciadas):

```
var minhaVariavel = valor;
```

  - Declaração de funções:

```
function minhaFuncao(argumento_1, argumento_2, …) {
        …
}
```

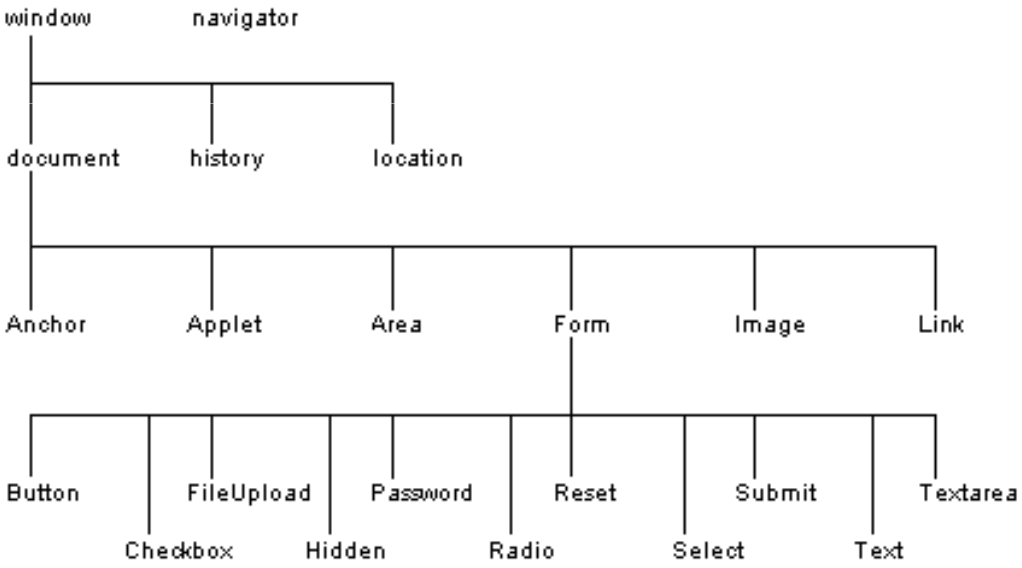
  - Chamar a função:

```
minhaFuncao(arg_1, arg_2,…);
```

- Hierarquia de objectos:

- Esses objectos são criados automaticamente pelo javascript se os elementos correspondentes existirem na página.

- Alguns existem sempre:
  - → navigator
  - → window
  - → document
  - → location
  - → history



- Objectos por defeito:

  - navigator:

    Contém o nome e a versão do navegador, plugins instalados, …
  - window:

    Propriedades aplicadas à janela inteira.
  - document:

    Propriedades acerca dos elementos do documento (título, core, …).
  - location:

    URL actual.
  - history:

    URLs visitados.

- O script “Hello World”:

```
<html>
 <head>
  <title>O meu primeiro script!</title>
 </head>
 <body>
 <script type="text/javascript">
 <!--

 document.write('<p>Hello World</p>');
  /* Isto é um comentário javascript */

 //-->
 </script>
 </body>
</html>
```

Início do script*

Tags de delimitação do script

document: elemento (ou objecto) que contem todos os elementos da página.
write(): função que apaga a página e, a seguir, escreve o texto.

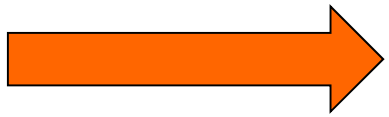
Pode-se inserir tags html no javascript

Fim do script

* É possível integrar um javascript num documento html através de um ficheiro externo:
<script type=“text/javascript” src=“ficheiro_javascript.js”></script>

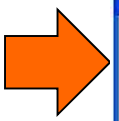
- As caixas de diálogo:
  - A função *alert* (apenas uma mensagem)

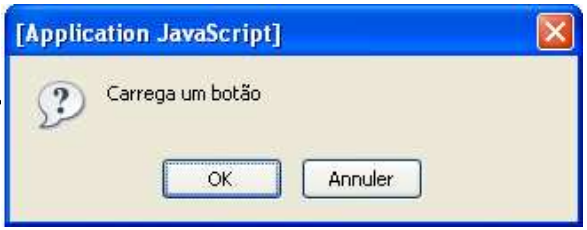
```
alert('Hello World!');
```



  - A função *confirm* (escolha entre 2 alternativas)

```
if(confirm('Carrega um botão'))  /* se clique no OK… */
{
alert('Carregou no botão ok!'); /* …imprime este alerta */
}
else
{
alert('Carregou no botão annuler!'); /* senão imprime este */
}
```



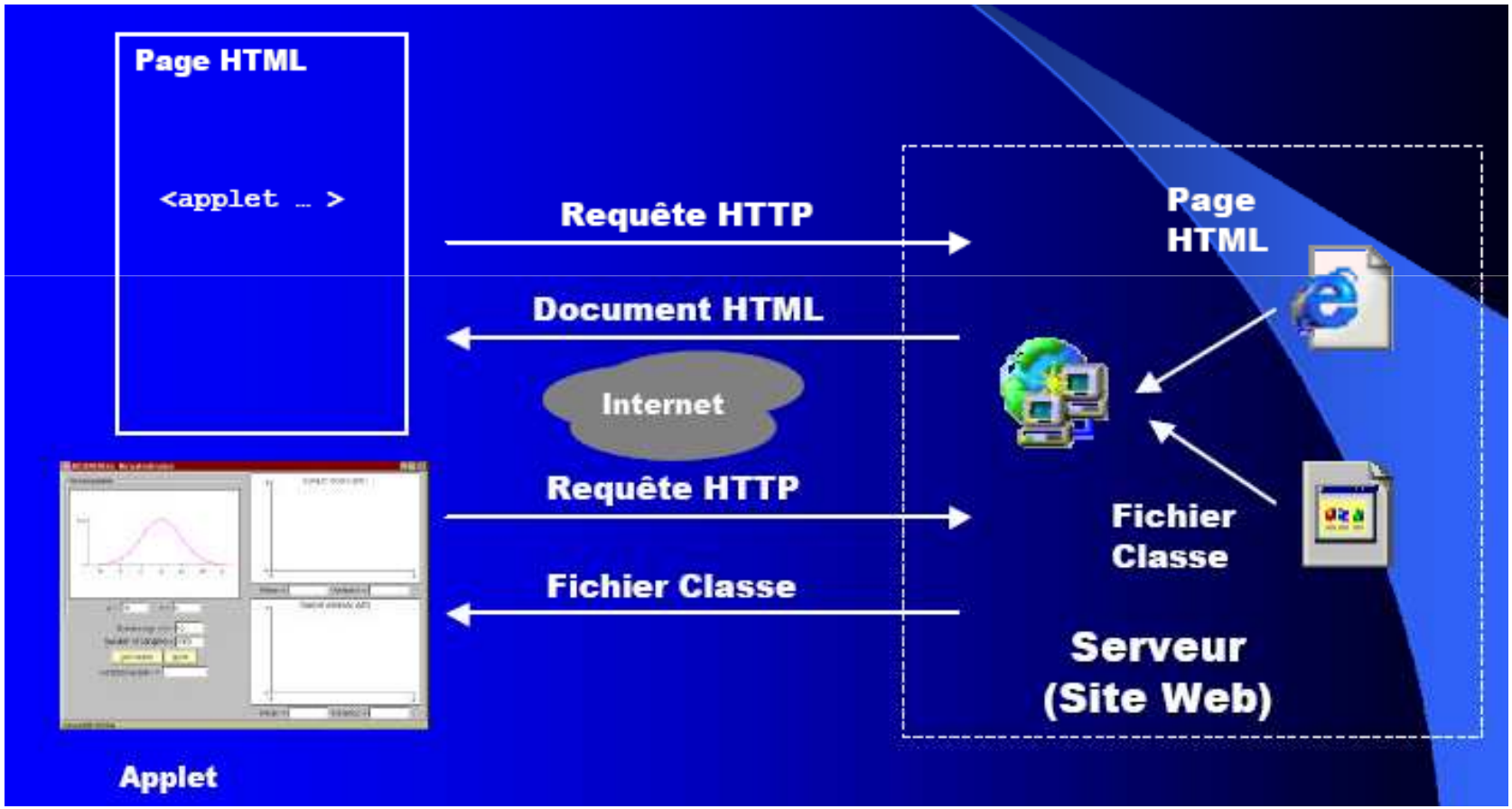


☞ **Applet**

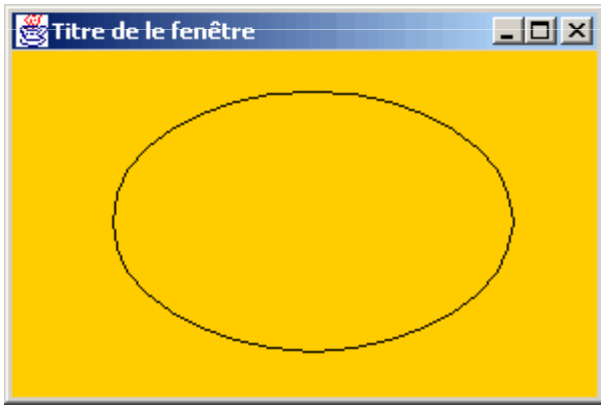
- = programa Java chamado por um documento html, através dos tags seguintes:

```
<applet code="nome_applet.class" height="…" width="…">
</applet>
```

- Um plugin permite lançar uma máquina virtual (JVM) dentro do browser.
- O browser download o código Java que é, depois, compilado em .class e executado localmente.
- Tem um ciclo de vida determinado pelo browser: inicia a applet quando a sua janela é visível no ecrã e pára-a quando desaparece do ecrã.


Page HTML
<applet … >
Requête HTTP
Document HTML
Internet
Requête HTTP
Fichier Classe
Page HTML
Fichier Classe
Serveur (Site Web)
Applet

- Exemplo de código
  - Aplicação Java standard:




```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class TestApplication extends JFrame {

  public static void main(String[] args) {
    new TestApplication().show();
  }
  public TestApplication() {
    setTitle("Titre de le fenêtre");
    setSize(300, 200);
    getContentPane().add(new Panneau());
  }
}

class Panneau extends JPanel {
  public void paintComponent(Graphics surface) {
    super.paintComponent(surface);
    setBackground(Color.orange);
    surface.drawOval(50, 20, 200, 130);
  }
}
```

En héritant de JFrame, notre application devient graphique et une fenêtre complète sera automatiquement construite

c'est bien une application puisque nous avons la méthode main

- Transformação da aplicação em applet:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class TestApplet extends JApplet {

  public void init() {
    getContentPane().add(new Panneau());
  }
}

class Panneau extends JPanel {
  public void paintComponent(Graphics surface) {
    super.paintComponent(surface);
    setBackground(Color.orange);
    surface.drawOval(50, 20, 200, 150);
  }
}
```

Cette fois-ci, la classe TestApplet hérite de JApplet au lieu de JFrame, c'est donc le navigateur qui gère l'affichage

La méthode main n'existe plus. Elle est remplacée par la méthode init

Ensuite tout le reste du code est identique

Para a class principal ser reconhecida como um applet tem que herdar de JApplet.

Desaparição do método main. A JVM lança o método init para inicializar e executar o applet no browser.

- Código html que integra o applet:

```
1   <html>
2     <head><title>Votre première applet</title>
3     </head>
4     <body bgcolor="navy" text="yellow">
5       <h2 align="center">Votre première applet</h2><hr />
6       <p align="center">
7         <applet code="TestApplet.class" height="200" width="300">
8         </applet>
9       </p>
10    </body>
11  </html>
```

Codage html classique

Placement de l'applet sur la page web en indiquant son nom et ses dimensions

- **ActiveX**
  - Tecnologia da Microsoft.
  - Ponto comum com as applets Java:
    Módulos de programa podem ser incorporados no ficheiro html.
  - Vantagem em relação aos applets:
    Permite trocar informações entre um formulário e um quadro (Excel) ou uma BD (Access).
  - Desvantagem em relação aos applets:
    Obriga a armazenar ficheiros (.vbx, .ocx, .dll ou .exe) no disco rígido do cliente => problema de segurança.
  - Pode ser criado através de várias linguagens:
    Visual Basic, Delphi, C, C++, …

Tecnologias da Web

Server side…

☞ **Cookies**

- String gerada pelo servidor e armazenada pelo browser.
- Usada para manter as informações de navegação.
- Viagem dentro dos headers http.
- Estrutura:
  - Nome
  - Valor
  - Date de expiração
  - Caminho de validade
  - Domínio de validade
  - Atributo de segurança

☞ CGI

- Common Gateway Interface.
- A tecnologia mais antiga para gerar paginas Web dinâmicas.
- Ainda muito utilizada.
- Execução de um programa (pré-compilado) no servidor:

  Configuração do servidor para indicar a pasta em que se encontram os scripts (geralmente: “cgi-bin”).
- Várias linguagens possíveis:
  - A linguagem PERL é a mais utilizada.
  - C, C++, Fortran, Shell, Python, … são ainda possíveis.

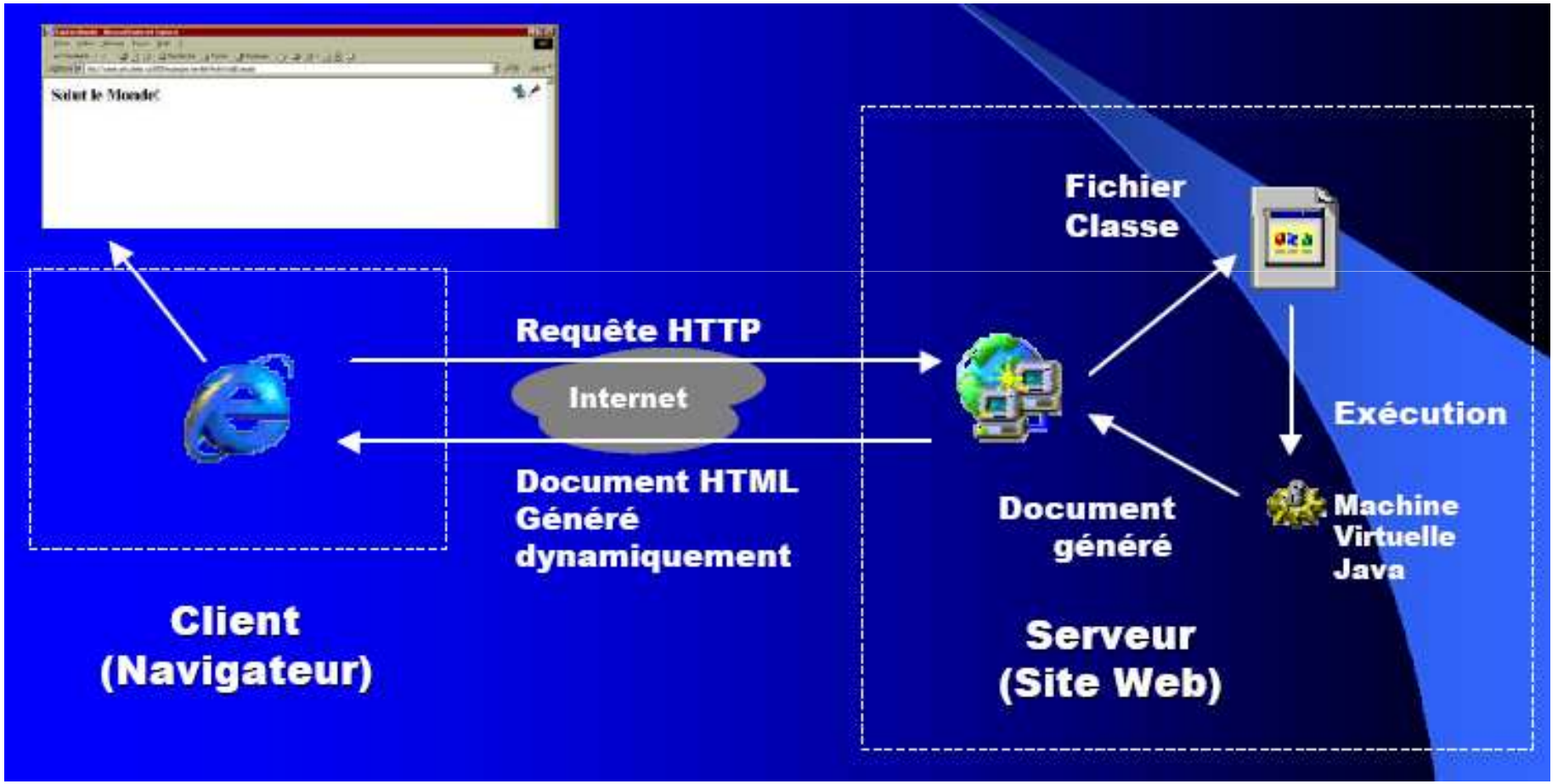
- Utilização através de formulários:
  - Método GET (visível) vs método POST (stdin).
  - Tratamento do request pelo programa.
  - Construção do código HTML correspondente.

- Inconveniente:

  Pede muito recurso sistema (a criação de uma nova cópia do programa a cada request => sobrecarga rápida do servidor).

☞ Servlets

- Applets (browser) vs Servlets (servidor).
- Processo:
  - http request para o URL desejado.
  - o servidor Web envia uma página estática para o cliente no formato html.
  - o cliente preenche o formulário, de modo não conectado ao servidor:
    - → poupa os recursos do servidor e a largura de banda.
    - → ↑ quantidade de clientes tratados em simultâneo.
  - envio do formulário → tratamento por um programa especializado = servlet.
  - servidor Web executa o servlet → criação da página dinâmica.


Salut le Monde!
Fichier
Classe
Requête HTTP
Internet
Document HTML
Généré
dynamiquement
Exécution
Document
généré
Machine
Virtuelle
Java
Client
(Navigateur)
Serveur
(Site Web)

- Exemplo de código de um servlet:

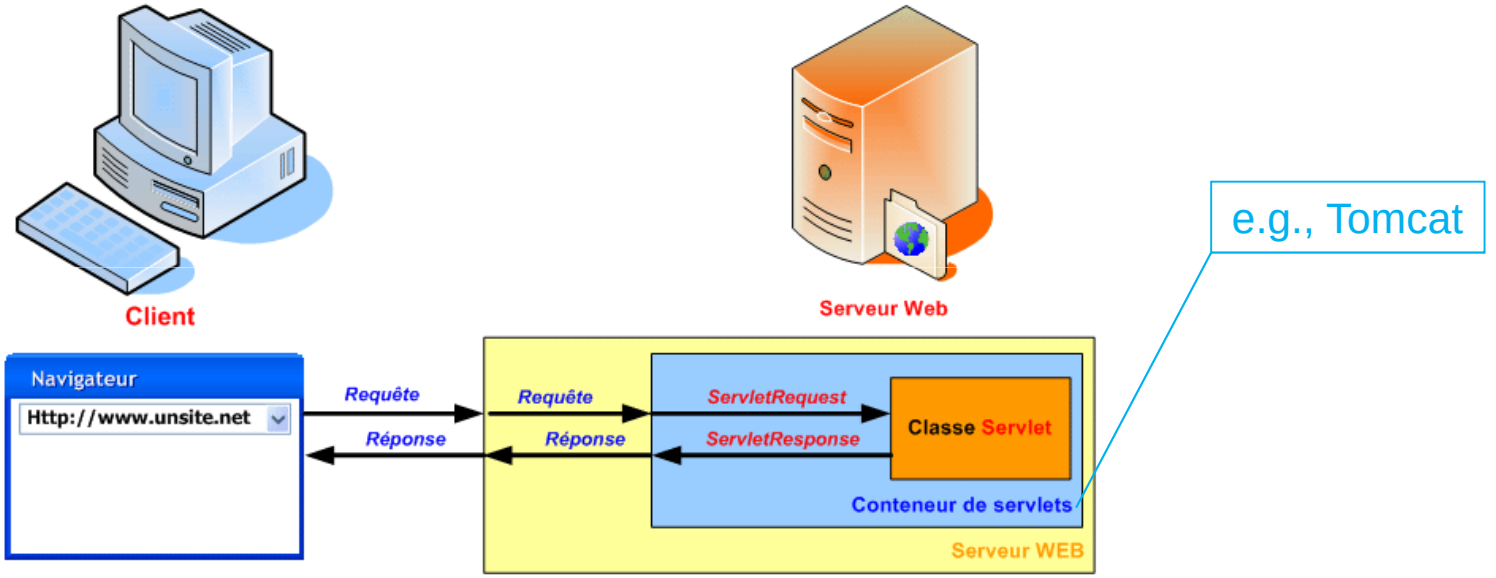
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

- Ferramentas para implementar:
  - **Java Servlet Development Kit** (JSDK), para programar.
  - Acrescentar o modulo **Tomcat** ao servidor Apache.

- Arquitectura de uma rede que utiliza servlets:



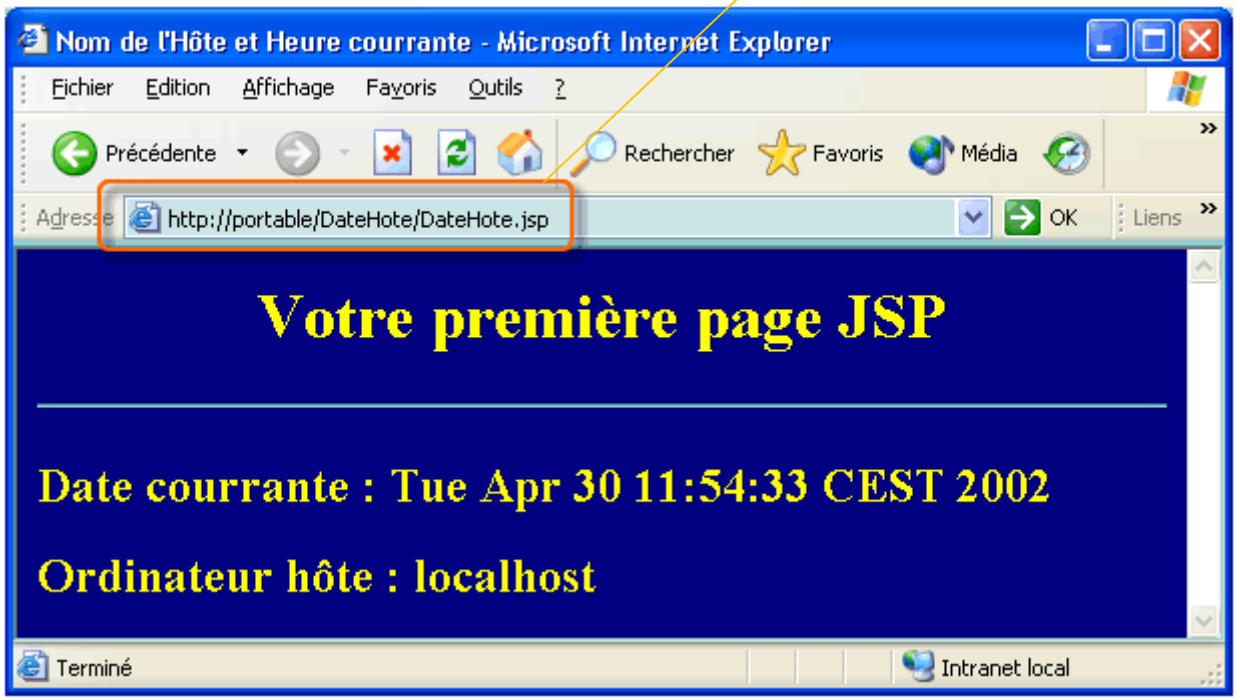
- Vantagens:
  - Java => exportável para qualquer tipo de plataforma.
  - Execução num motor de servlet distinto do servidor Web (e.g., Tomcat) => não gera a conexão.
  - Pode comunicar com uma BD e aplicações exteriores.

☞ JSP

- Tecnologia Java (Java Server Script).
- Linguagem de script.
- Permite inserir o script dinâmico dentro do código html:
  tag de inserção: <% … %>
- Extensão do ficheiro JSP: .jsp
- Processo em 4 etapas:
  - request recebido pelo servidor.
  - a página JSP pedida é convertida em servlet.
  - o servlet é compilado.
  - o servlet é executado para elaborar a página html+script dinâmico que será enviado ao cliente.
- Processo mais pesado do que as servlets (conversão).

```
<html>
<head>
<title>Nom de l'Hôte et Heure courrante</title>
</head>
<body bgcolor="#000080" text="yellow">
<h1 align="center">Votre première page JSP</h1><hr>
<h2>Date courrante : <%= new java.util.Date() %></h2>
<h2>Ordinateur hôte : <%= request.getRemoteHost()%></h2>
</body>
</html>
```

O servidor Tomcat é colocado no computador portátil, com o porto 80

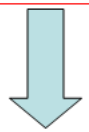


☞ **PHP**

- Hypertext Processor.
- Linguagem de script:
  - orientado pelo objecto
  - sem tipo
  - sintaxe ≈ C
- Integra-se no código html.
- Ficheiros .php
- O servidor interpreta o código PHP e envia a página modificada ao cliente.
- Permite um acesso fácil às bases de dados.

- Exemplo de integração num código html:

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
     <?php echo '<p>Hello World</p>'; ?>
</body>
</html>
```



Exécution par le serveur

```
<html>
<head>
<title>PHP Test</title>
</head>
<body>
      <p>Hello World</p>
</body>
</html>
```

- Exemplo de comunicação com um formulário html:

```
<form action="action.php" method="post">
  <p> Your name: <input type="text" name="name" /> </p>
  <p> Your age: <input type="text" name="age" />      </p>
  <p> <input type="submit" />                         </p>
</form>
```

Client

```
Hi <?php echo $_POST['name']; ?>.
You are <?php echo $_POST['age']; ?> years old.
```

Serveur

```
Hi Joe. You are 22 years old.
```

Client

☞ ASP / ASP.NET

- Active Server Page (framework .net).
- Tecnologia Microsoft.
- Linguagem de script.
- Extensão do ficheiro ASP: .asp
- Mesmo princípio do que JSP:
  - inserção dentro do código html através da utilização de tags.
  - arquitectura 3-tier, em que o servidor de ASP serve de intermédio entre o browser e as BD, através de um acesso baseado na tecnologia ADO (ActiveX Data Objecto).