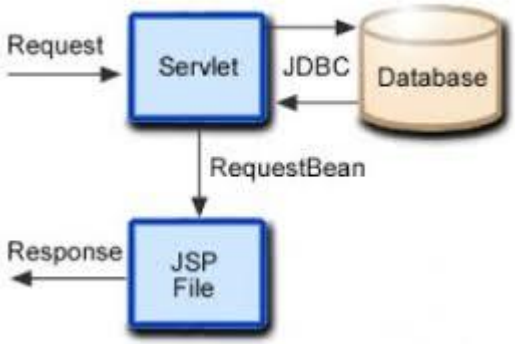
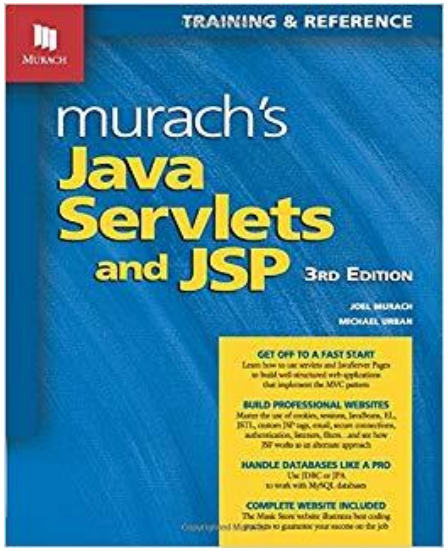
Unidade 1
Visão Geral da Tecnologia Servlet e JSP

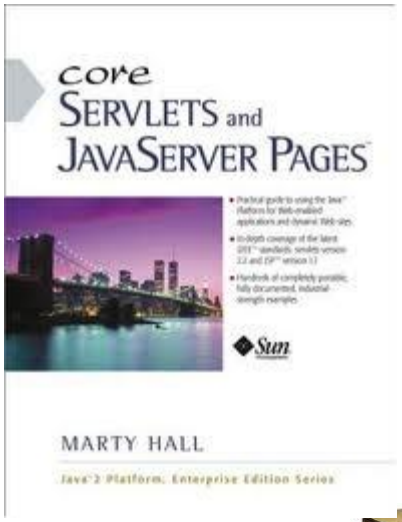


Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP

Bibliografia

- Head First Servlets & JSP – Bryan Basham, Kathy Sierra & Bert Bates
- Core Servlets and Java Server Pages – Marty Hall
- Java Servlets and JSP –Joel Murach – 3rd Edition

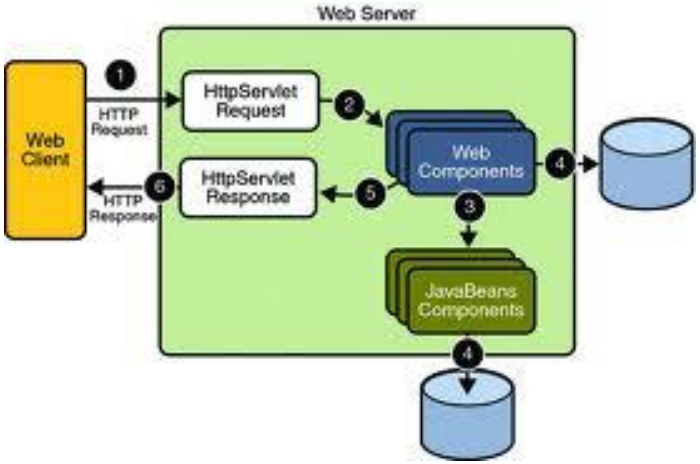




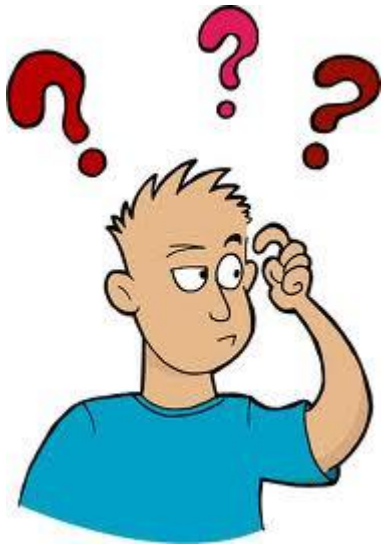
Aplicações WEB

- **WEB** sites (**páginas estáticas**) são limitados e geralmente não atendem em plenitude as necessidades de aplicações corporativas.

- Atualmente usuários necessitam aplicações **WEB** que sejam dinâmicas, interativas e configuráveis.

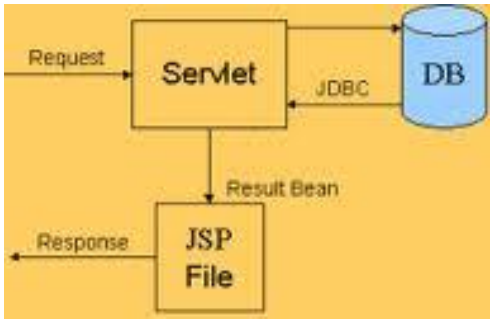


Porque usar Servlets e JSP ?

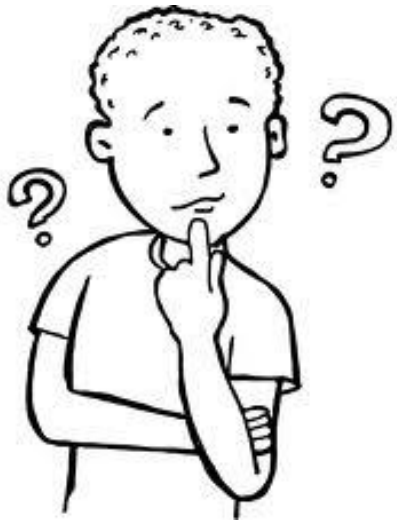


Tecnologia Servlet e JSP

- Permitem o desenvolvimento de aplicações **WEB** com páginas dinâmicas.

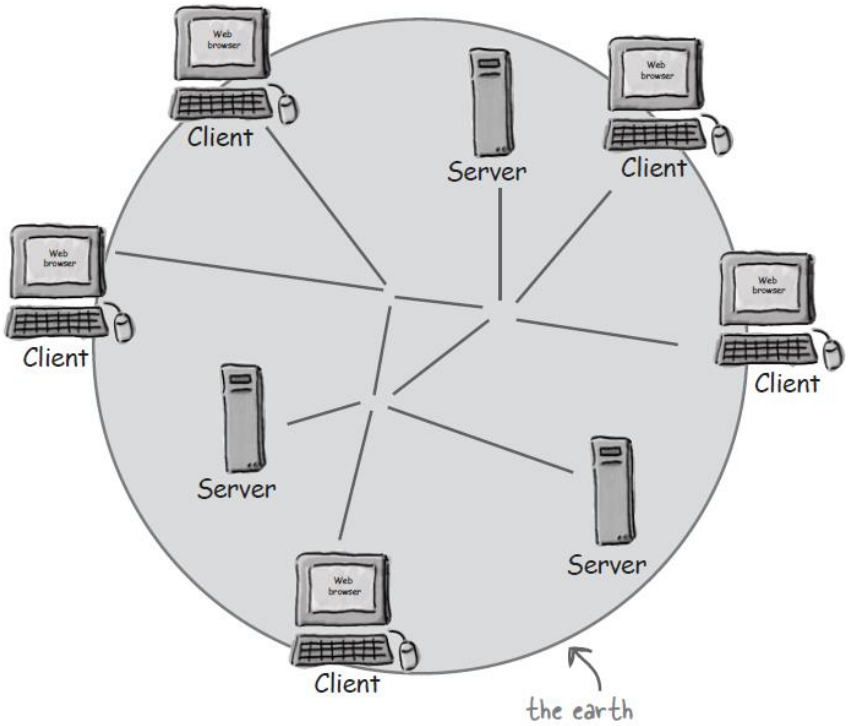


O que um servidor WEB faz ?

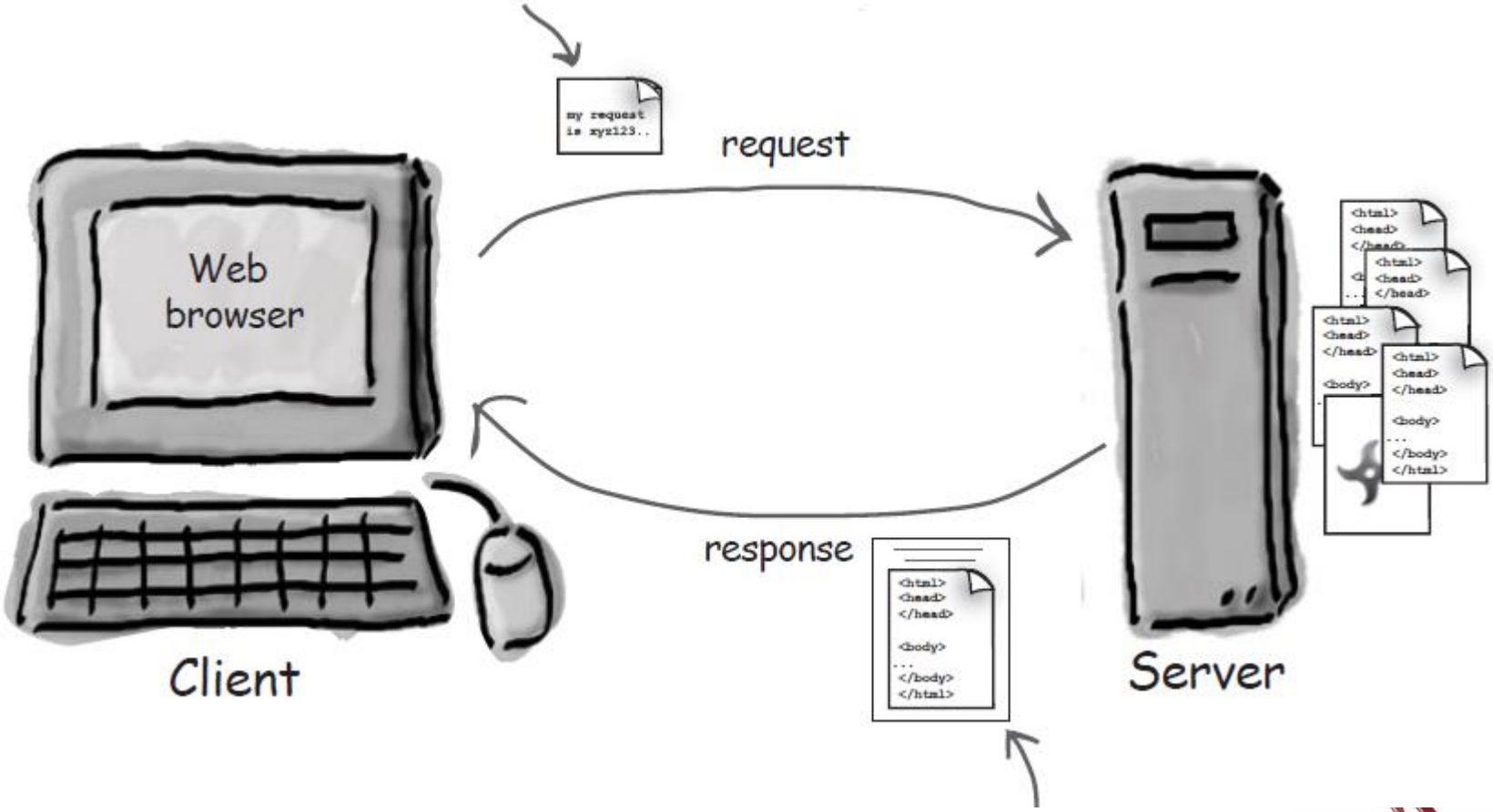


WEB Server

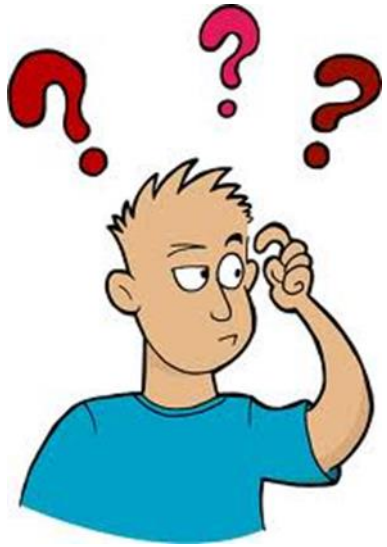
Recebe um *request* do cliente e devolve um *response* ...



Paradigma Request / Response

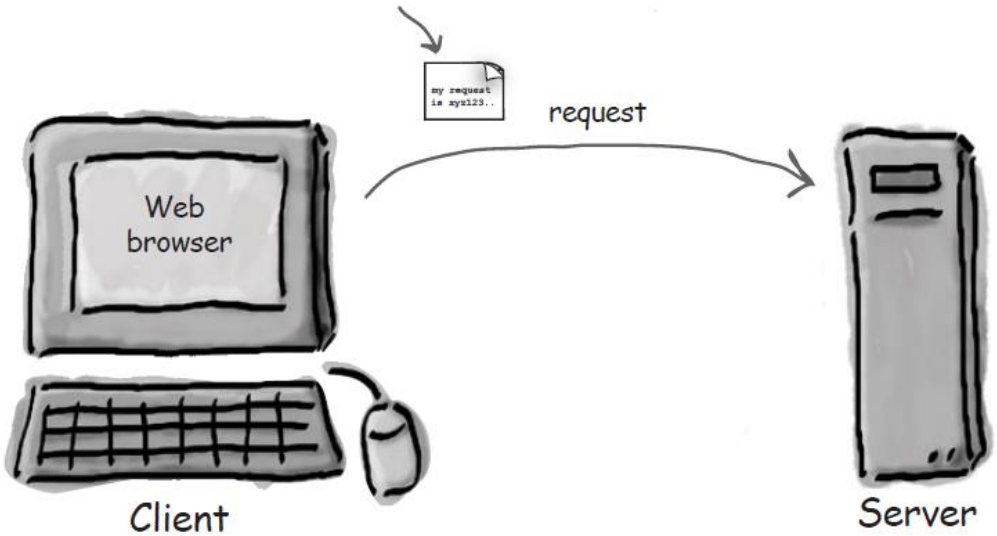


O que contém o *request* do cliente ?

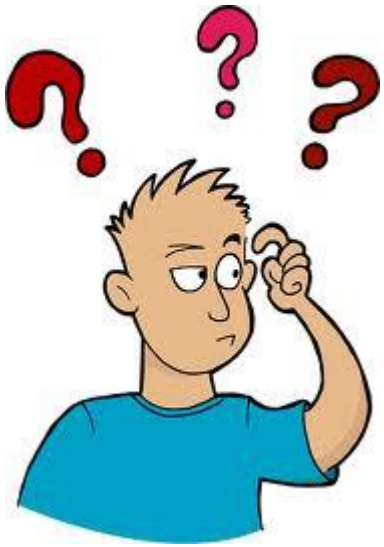


Request do Cliente

- Contém o nome e o endereço (**URL**) da informação que o cliente está procurando ...

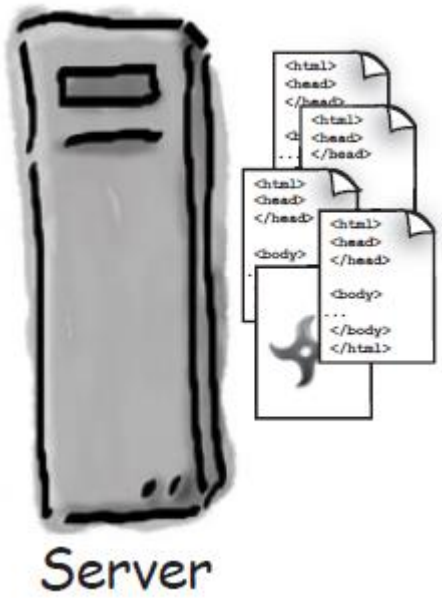


O que se armazena em um servidor WEB ?



Servidor WEB

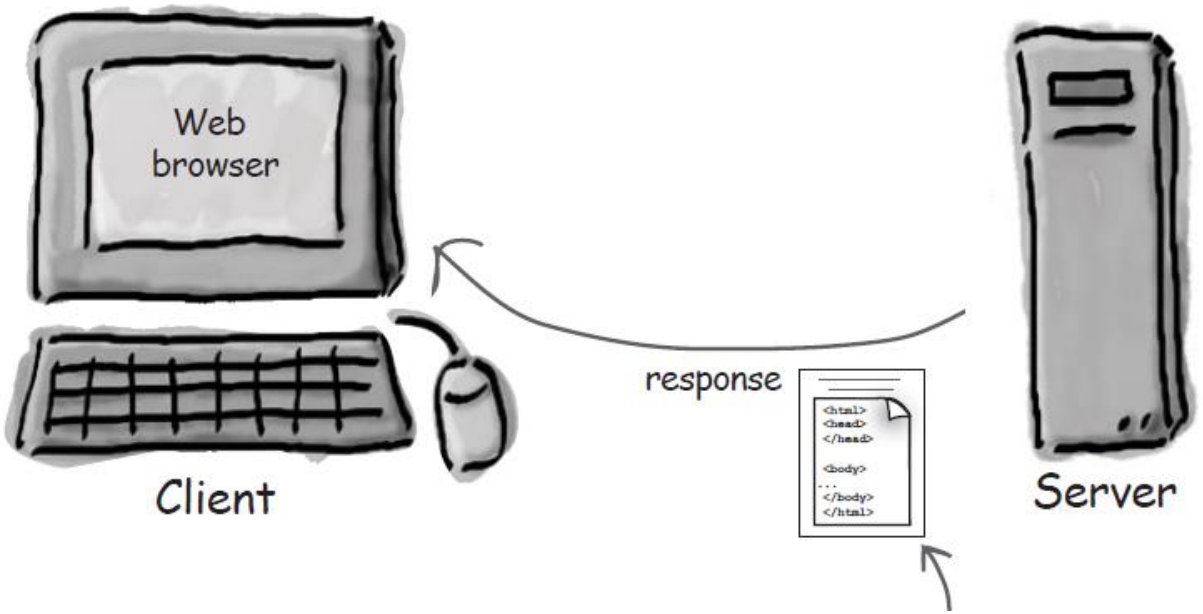
- O servidor **WEB** usualmente tem diversos “conteúdos” que ele pode enviar aos clientes.
- Este conteúdo pode ser páginas **HTML**, **JPEGs**, e outros recursos.



O que contém a resposta do servidor WEB ?

Response do Web Server

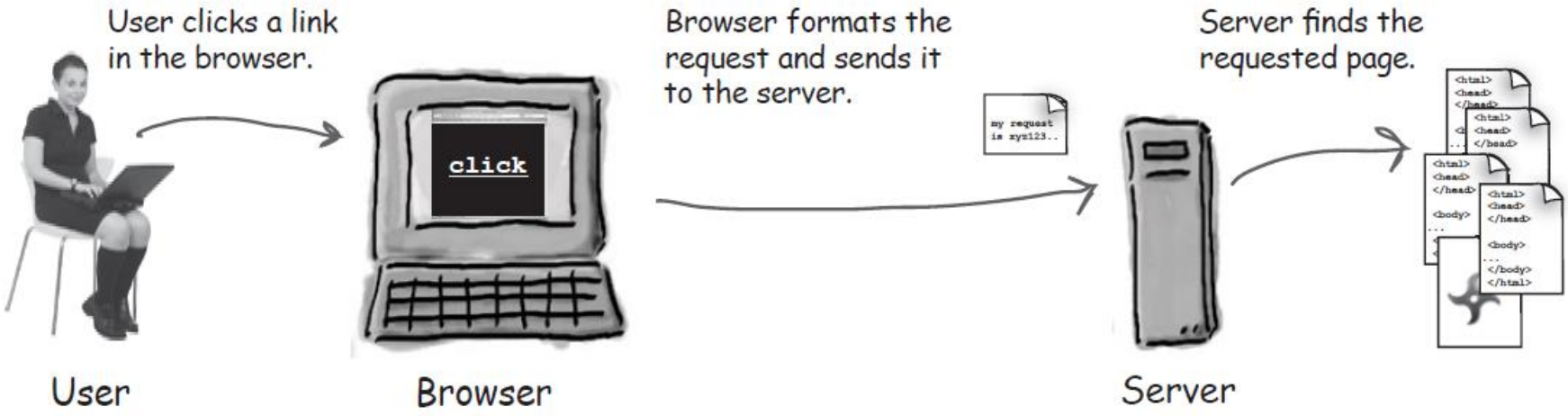
- A resposta do servidor contém o documento requisitado pelo cliente.
- Em caso de erro, o servidor envia uma mensagem informando que o documento não pode ser recuperado ou que não existe.



O que faz um cliente Web ?

Cliente Web

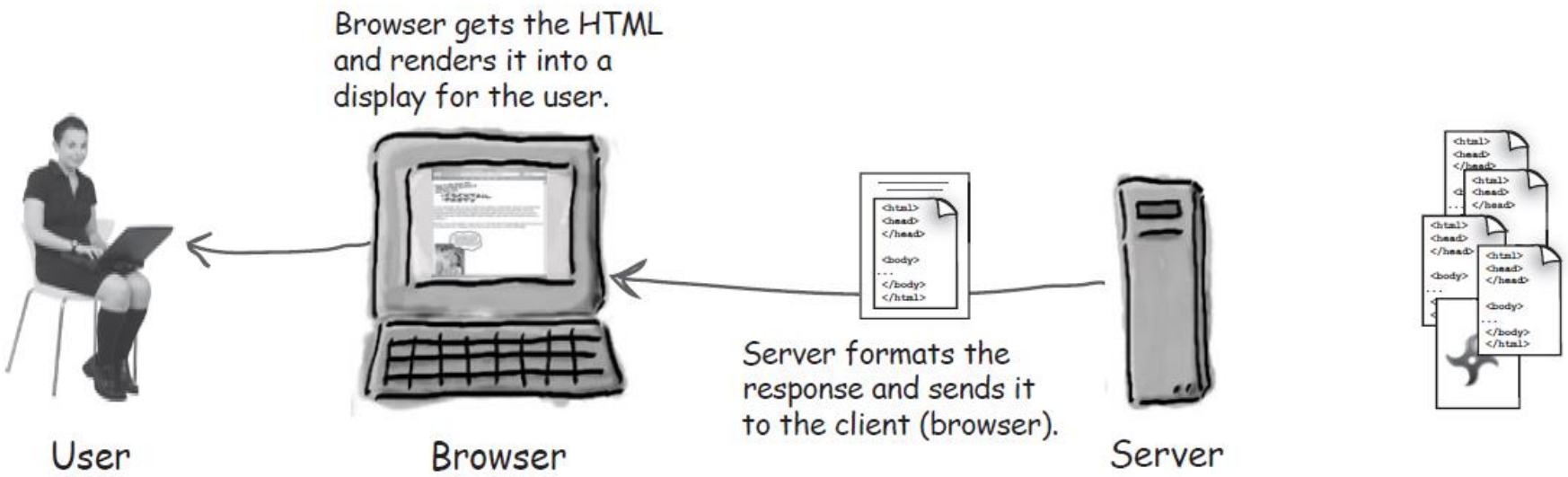
- Clientes podem representar o usuário (ser humano) ou uma aplicação Browser.
- O Browser sabe como se comunicar com o servidor.



O que faz o servidor Web ?

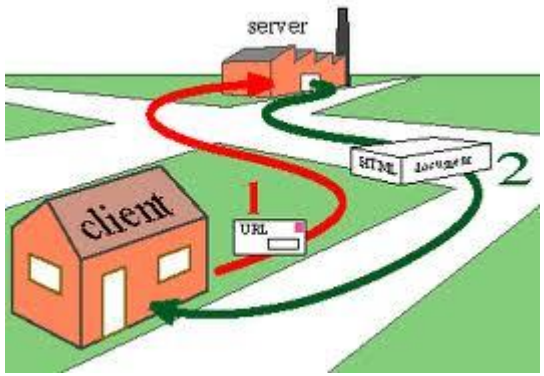
Servidor Web

- Formata a resposta e a envia para o **cliente** (Browser);
- Browser interpreta o código HTML e renderiza a página para o usuário.



Clientes e servidores conhecem HTML e HTTP

- Servidores frequentemente enviam ao **Browser** um conjunto de instruções escritas em **HTML**;
- A maioria das conversações entre cliente e servidor são processadas por meio do protocolo **HTTP;**
- O cliente envia um request **HTTP** e o server responde com um **HTTP** response.



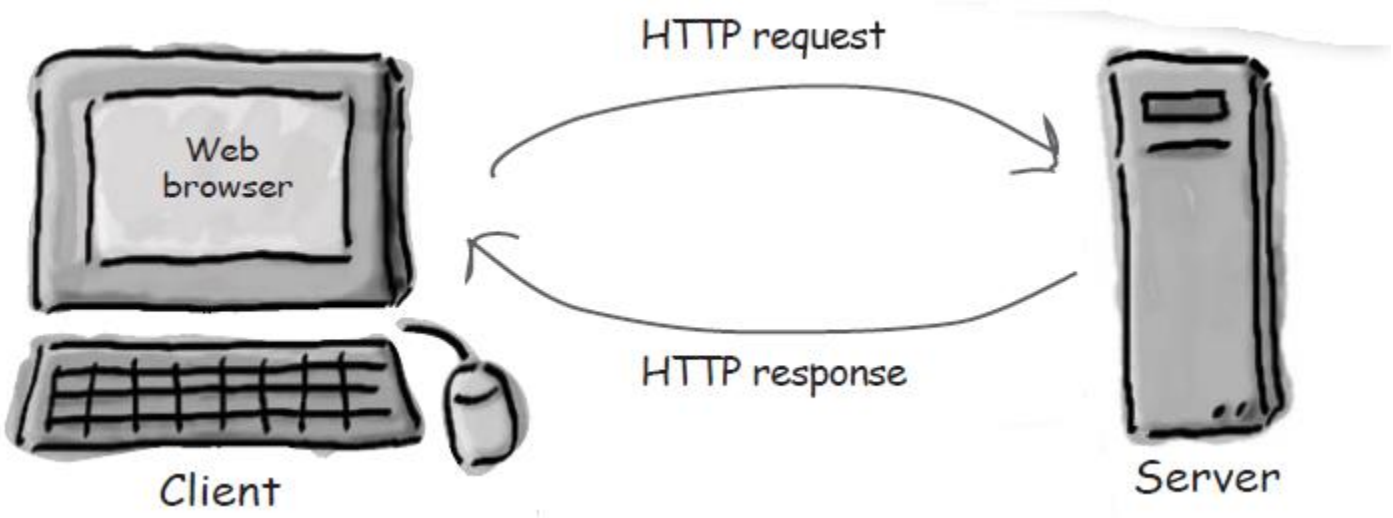
O que é o protocolo HTTP ?

Protocolo HTTP

- HTTP roda no topo do **TCP/IP**;
- TCP assegura que um arquivo seja enviado corretamente de um ponto a outro da rede;
- IP é o protocolo que roteia pacotes (sockets) entre pontos da rede;
- **HTTP** é outro protocolo aplicado especificamente para o ambiente WEB;
- A estrutura do protocolo HTTP é uma conversação baseada no paradigma **Request/Response**.

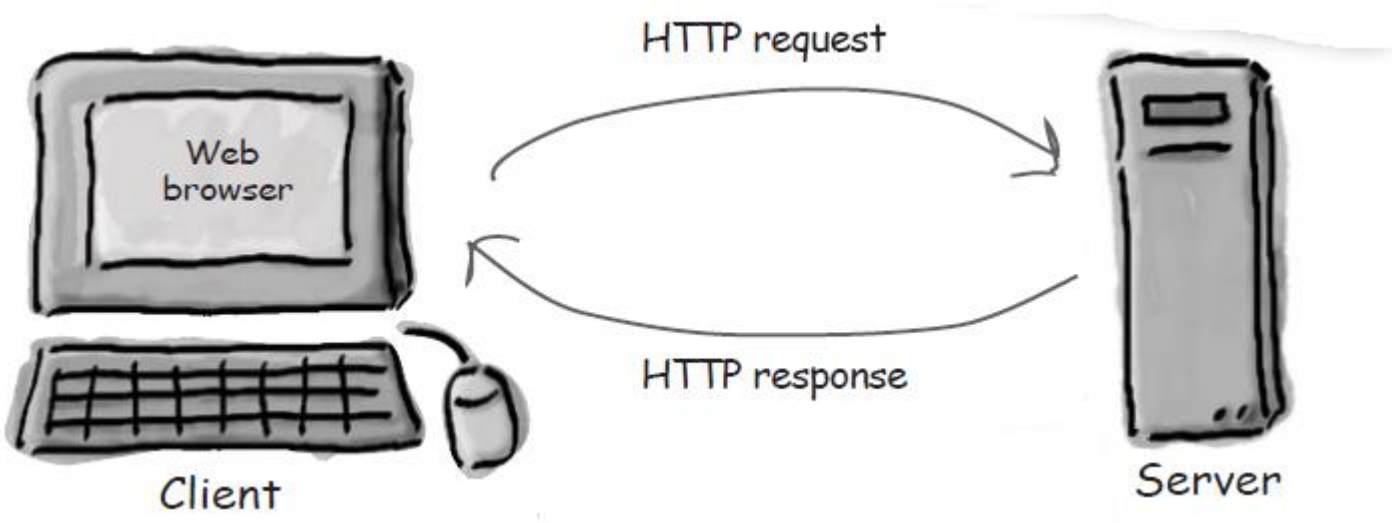
Elementos chave do Request

- Método **HTTP** (a ação a ser executada)
- Página a ser acessada (**URL**)
- **Parâmetros** do form (como argumentos em um método)



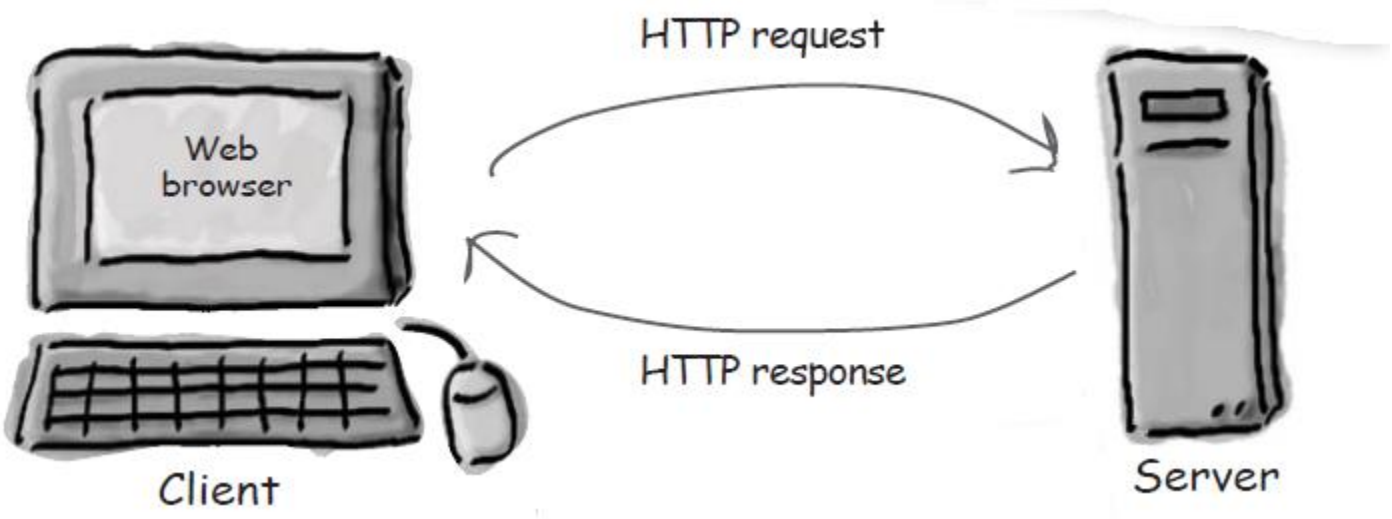
Elementos chave do Response

- Um **status code** (quando o request foi atendido).
- **Content-type** (text, picture, HTML, etc).
- O conteúdo (HTML, imagem, etc).



Especificação HTTP

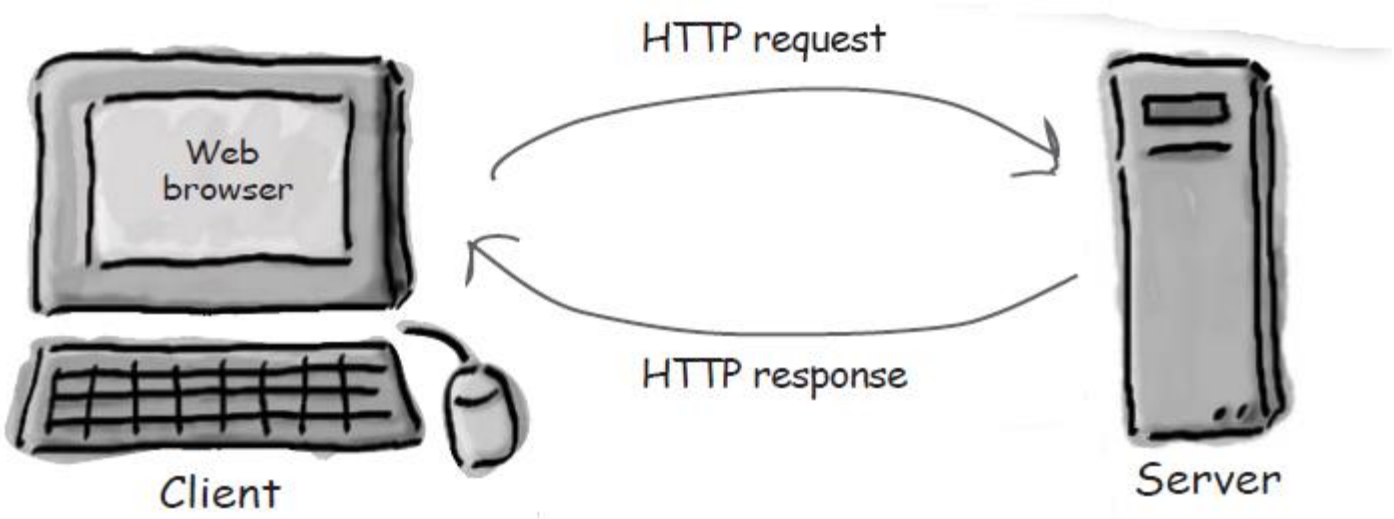
- O protocolo **HTTP** é um padrão IETF, **RFC 2616**;
- **Apache** é um exemplo de um Web Server que processa requests HTTP;
- **Mozilla** é um exemplo de um browser que encaminha requests HTTP e permite a visualização de documentos retornados pelo servidor.



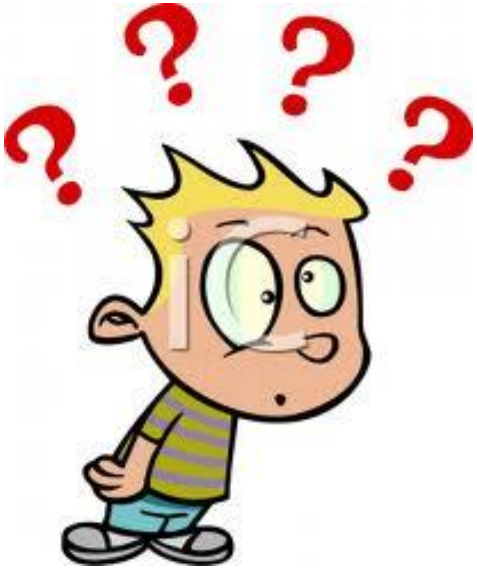
HTML é parte do request ou do response ?

HTML é parte do response

- HTTP adiciona informações de cabeçalho à qualquer conteúdo que esteja na resposta (**response**);
- O Browser utiliza estas informações para auxiliar no processamento da página HTML;
- Você pode imaginar o HTML como sendo o conteúdo passado no response do servidor.

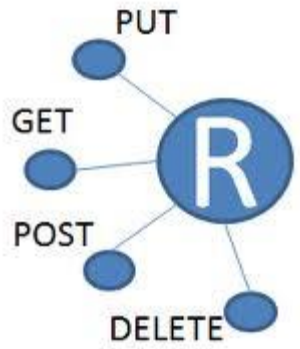


O que está no Request ?

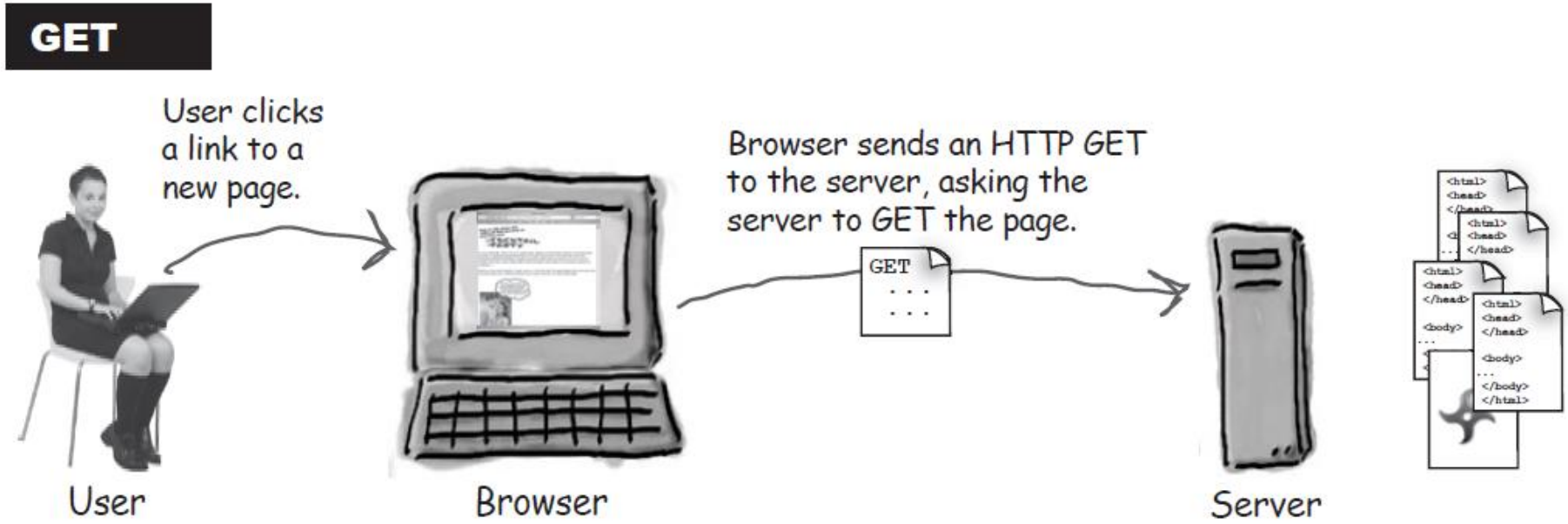


Formatação do Request

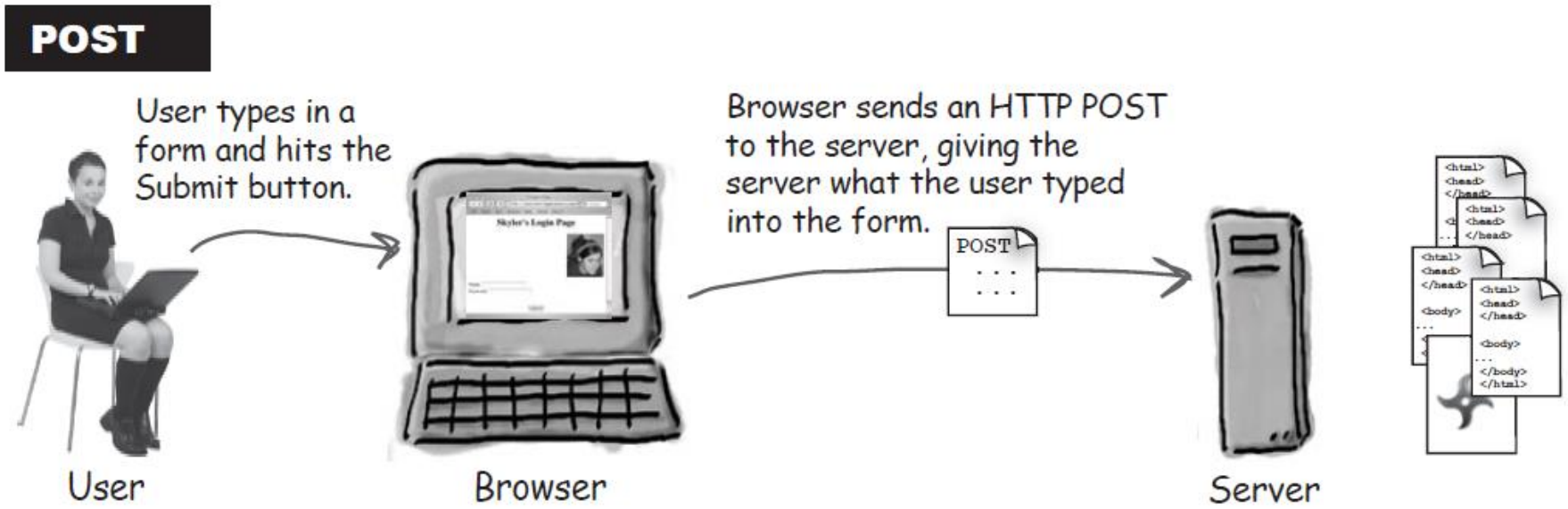
- No request a primeira informação que se vê é o método;
- O método diz ao servidor o tipo de request que está sendo solicitado ao servidor e como o resto da mensagem será formatada;
- Os métodos mais empregados são **GET** e **POST**.



Método Get


GET
User clicks a link to a new page.
Browser sends an HTTP GET to the server, asking the server to GET the page.
GET
User
Browser
Server

Método Post


POST
User types in a form and hits the Submit button.
Browser sends an HTTP POST to the server, giving the server what the user typed into the form.
POST
User
Browser
Server

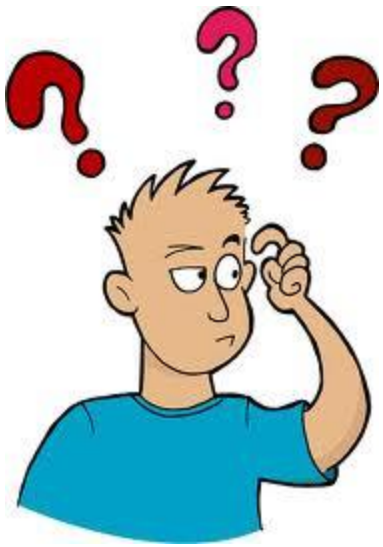
Quais as diferenças entre os métodos GET e POST ?

Métodos GET e POST

- **GET** é um método **HTTP** mais simples;
- A principal tarefa do método GET é pedir (**GET**) ao servidor um determinado recurso;
- Este recurso pode ser uma página **HTML**, um **JPEG**, um **PDF**, etc.
- A idéia básica do método **GET** é pedir algo ao servidor.

- O método **POST** é mais elaborado;
- É como um **GET**++.
- Com o método **POST** você pode pedir algo e ao mesmo tempo enviar dados por meio de um **FORM**.

É possível enviarmos dados por meio do Método GET ?



Enviando dados com GET ?

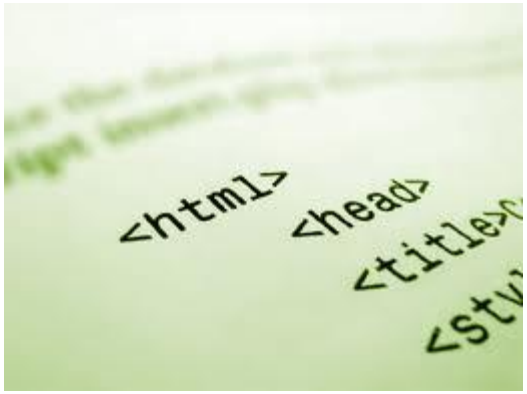
- **Sim**, é possível. Mas **POST** é mais recomendado para esta necessidade;
- A quantidade total de bytes enviadas pelo método **GET** é limitada (depende do servidor). Se o usuário digitar um longo texto, **GET** pode não trabalhar;
- O dado enviado no método **GET** é apenso à **URL**. Assim, a informação será visível (Senhas ???).

Opcional Query String

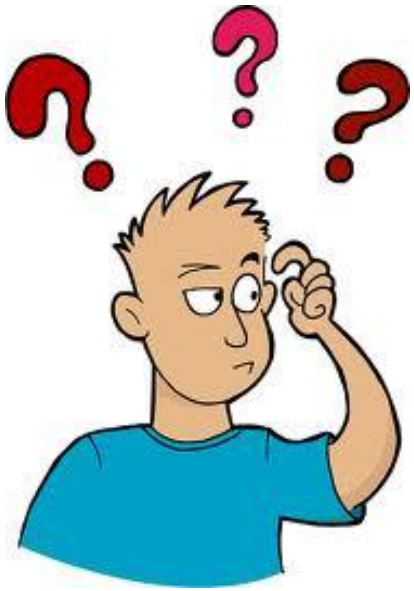
- Ao se empregar um request **GET**, pode-se entrar com informações extras (parâmetros) que são apensos ao final da **URL**;
- Estes parâmetros são iniciados com '**?**'.
- Cada parâmetro é definido pelo par (**nome/valor**) e separados por '**&**'

Páginas estáticas

- São páginas (arquivos) armazenadas nos diretórios do Web Server;
- O servidor as encontra e as devolve ao cliente justamente como estão codificadas;
- Todo o cliente vê uma página estática da mesma maneira.

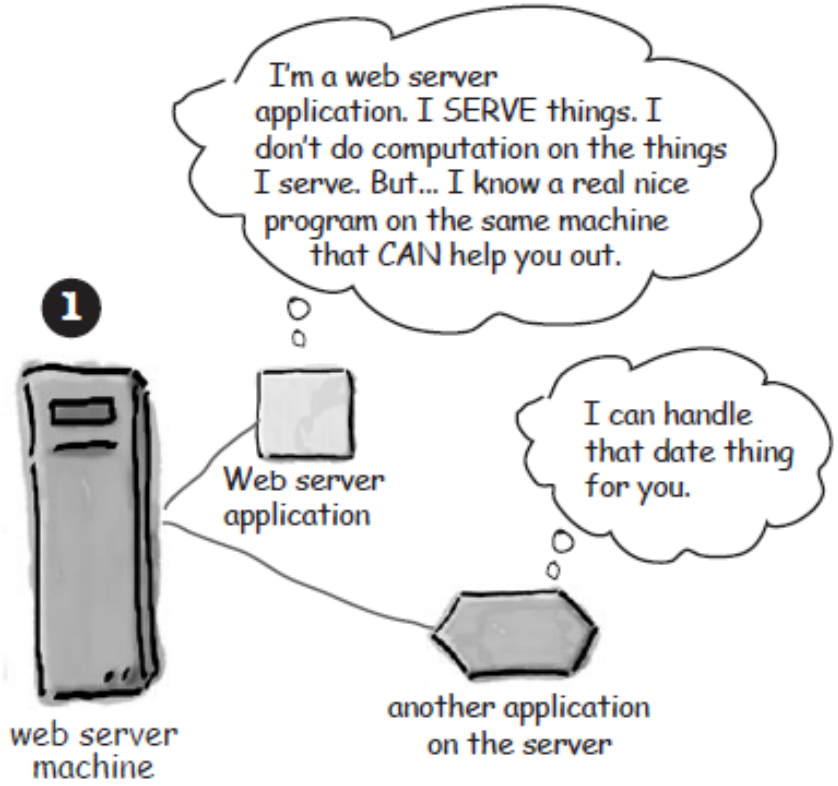


Mas, e se o usuário quizer uma página com alguma característica dinâmica ?

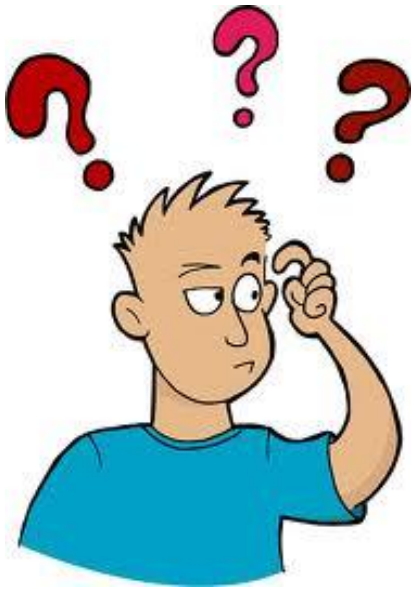


Páginas dinâmicas

- Uma aplicação Web Server serve páginas ...
- Aplicação Web Server não faz computação com estas páginas;
- No entanto, o Web Server pode chamar um programa na mesma máquina que pode efetuar alguma computação.



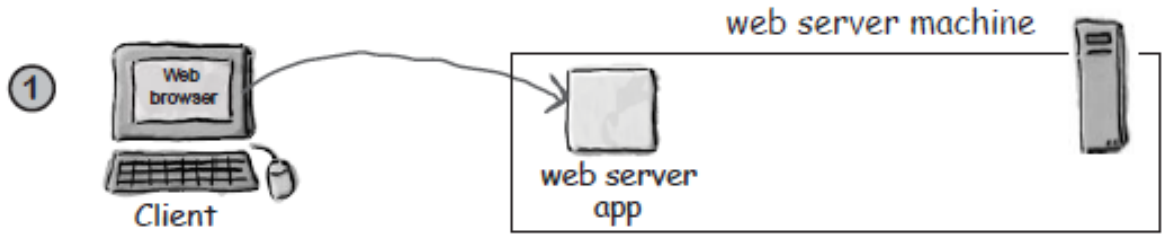
Para processamento de páginas dinâmicas o Web Server precisa de ajuda...



Coisas que o Web Server não faz sozinho ...

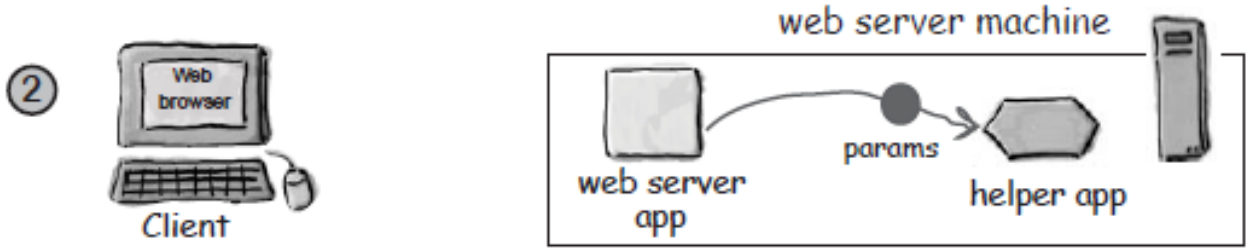
- Um **Web Server** serve apenas páginas estáticas, mas uma aplicação nos bastidores, no qual o **Web Server** se comunica, pode construir páginas não-estáticas, ou páginas just-in-time.
- Para processar dados de um form, será necessário chamar outra aplicação. O **Web Server** assume que os parâmetros são significativos para a aplicação a ser chamada e os transfere.
- O termo não-Java empregado para a aplicação a ser chamada é **CGI**.

Processamento de Programa CGI



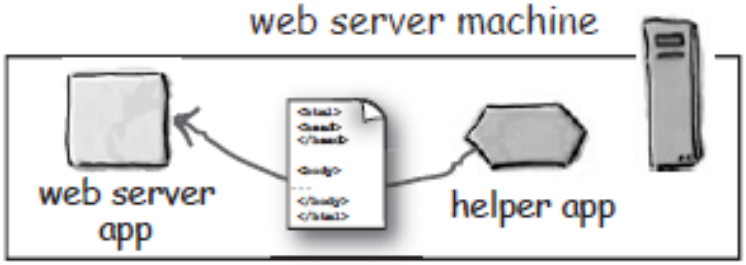
User clicks a link that has a URL to a CGI program instead of a static page.

Processamento de Programa CGI



Web server application "sees" that the request is for a helper program, so the web server launches and runs the program. The web server app sends along any parameters from a GET or POST.

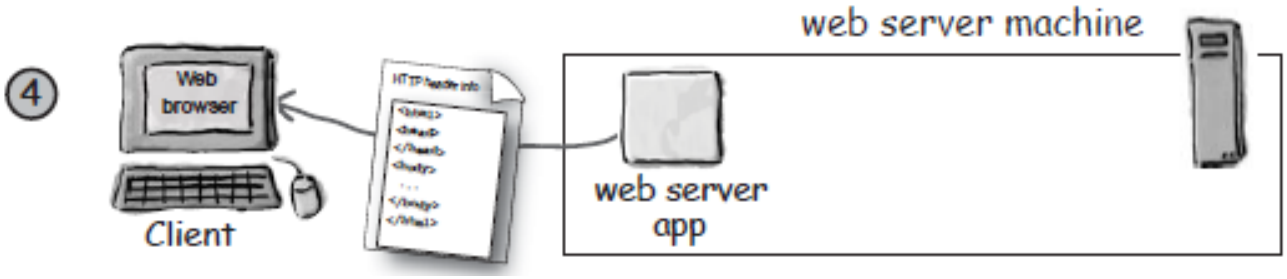
Processamento de Programa CGI



The helper app constructs the brand new page (that has the current date inserted) and sends the HTML back to the server.

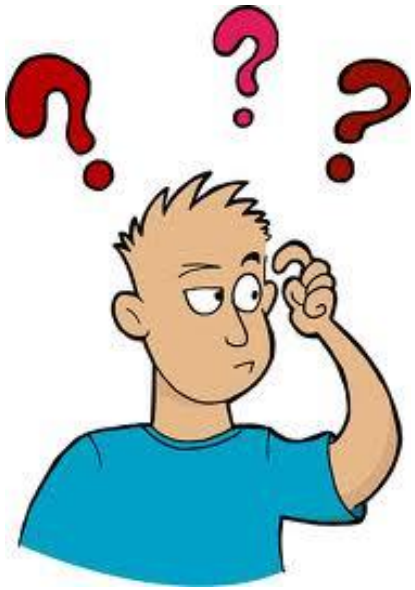
As far as the web server is concerned, the HTML from the helper app is a static page.

Processamento de Programa CGI



The helper application is shut down, and the client gets back an HTML page that has the current date as part of its now-static content.

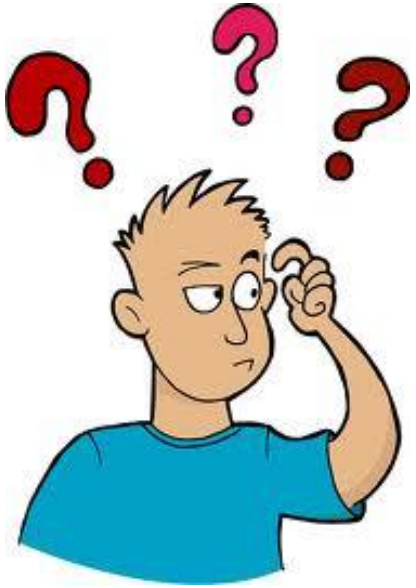
O que são Servlets ?



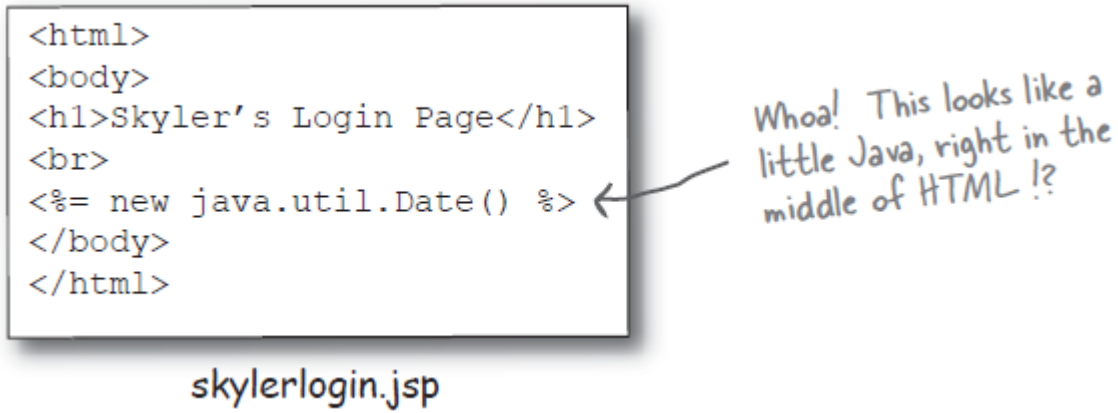
Servlets

- São programas **CGI** escritos na Linguagem **Java**.
- Necessitam portanto da infraestrutura Java para serem executados;
- Rodam em **JVM – Java Virtual Machine**;
- O servidor que fornece a infraestrutura para se executar **Servlets** é chamado **Web Container**.
- Exemplos de Web Container: **Tomcat**, **Jboss**, **WebSphere**, **Glassfish**, etc.

E se colocássemos Java em uma página HTML ao invés de HTML em uma classe Java?



Java Server Pages



- Uma página **JSP** na verdade é uma página **HTML**, exceto que se coloca Java e atributos do tipo Java dentro da página.

Java Server Pages

- Nem todos os designers **HTML** conhecem **Java**.
- Com JSP, desenvolvedores Java se concentram no código Java enquanto que designers **HTML** constróem as páginas **HTML**.