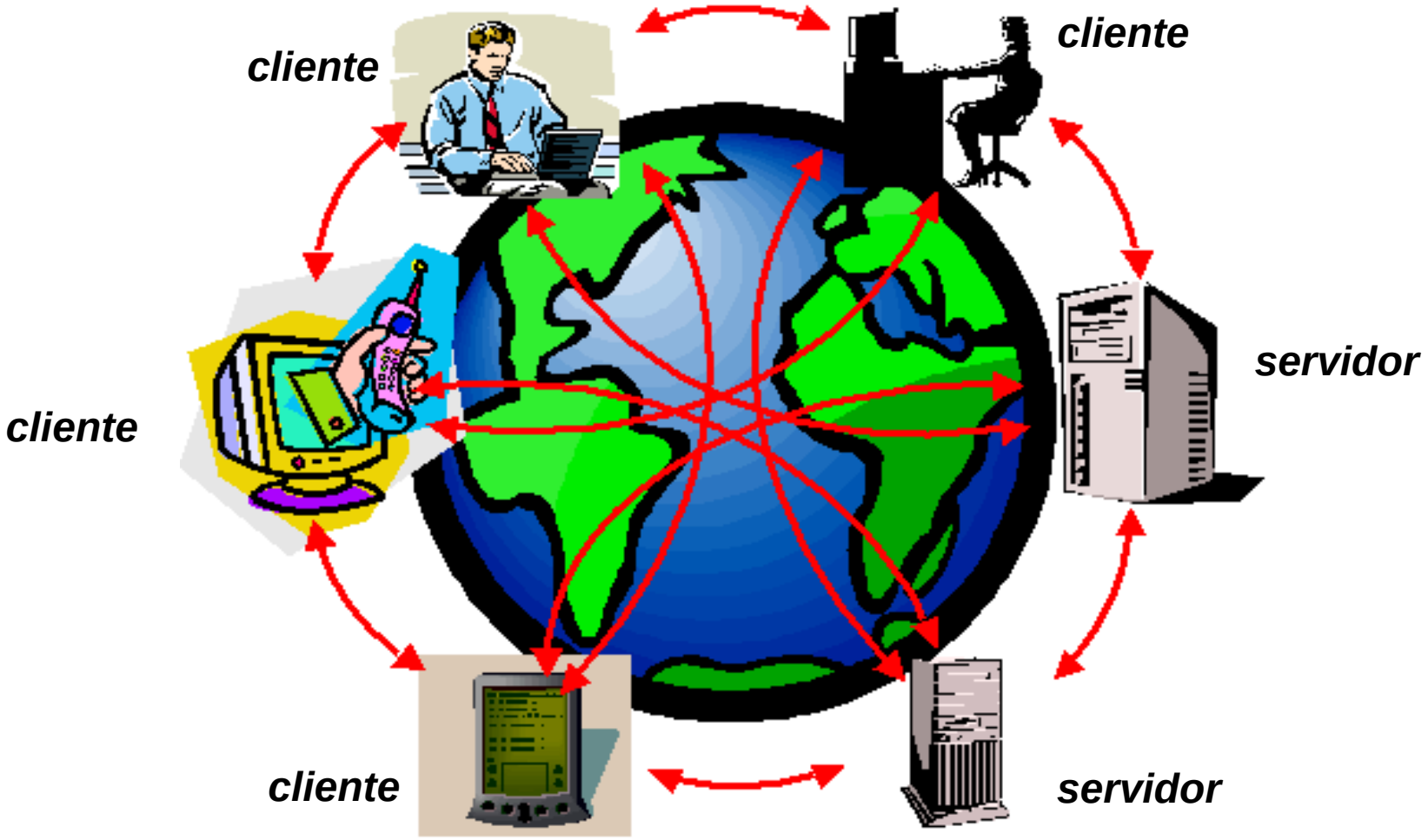
Introdução à Programação Java para Web

Prof. Leonardo Vianna do Nascimento
Disc. de Desenvolvimento de Sistemas I

A Internet



A Internet

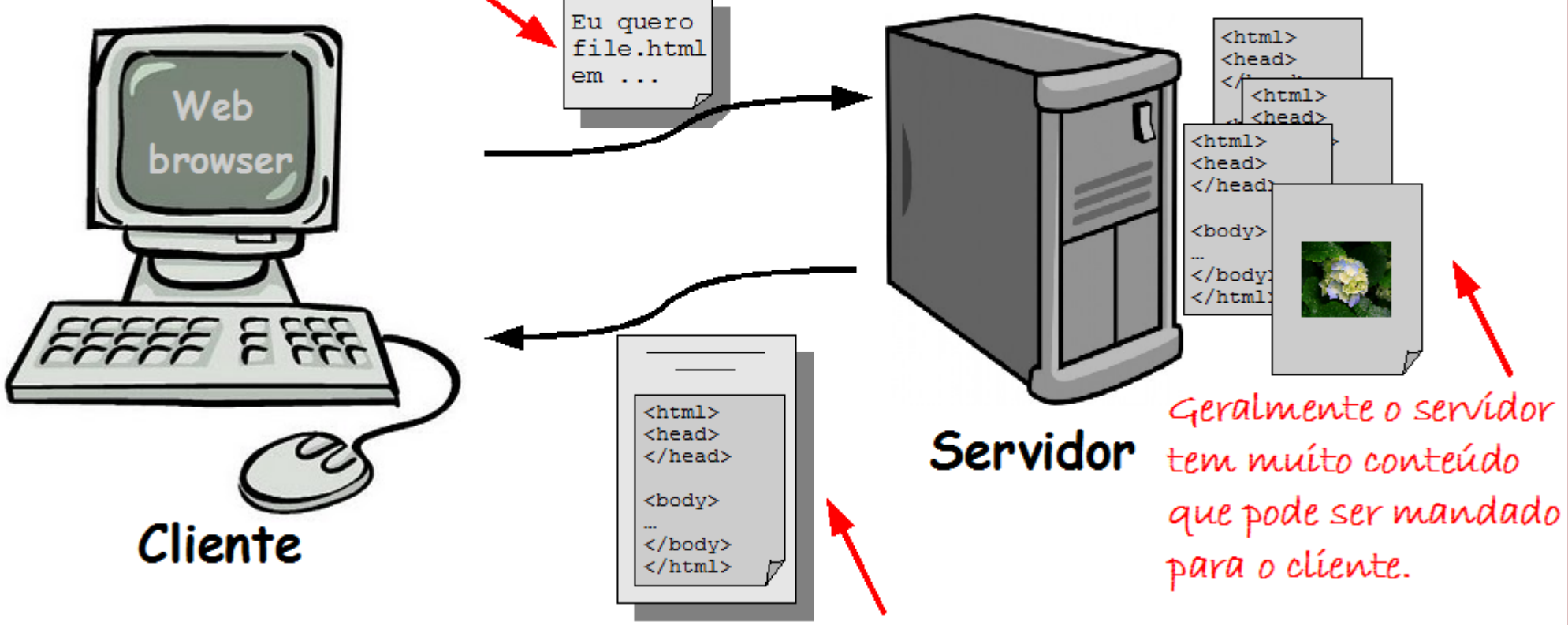
A Internet consiste em milhares de milhares de clientes (usando navegadores como Mozilla ou o Safari) e servidores (rodando aplicações como o Apache), conectados através de redes com fio e wireless.

O que faz um Servidor?

Um servidor recebe uma solicitação e devolve algo para o cliente

Exemplo: Servidor Web

A solicitação do cliente contém o nome e endereço (a URL) daquilo que o cliente está procurando



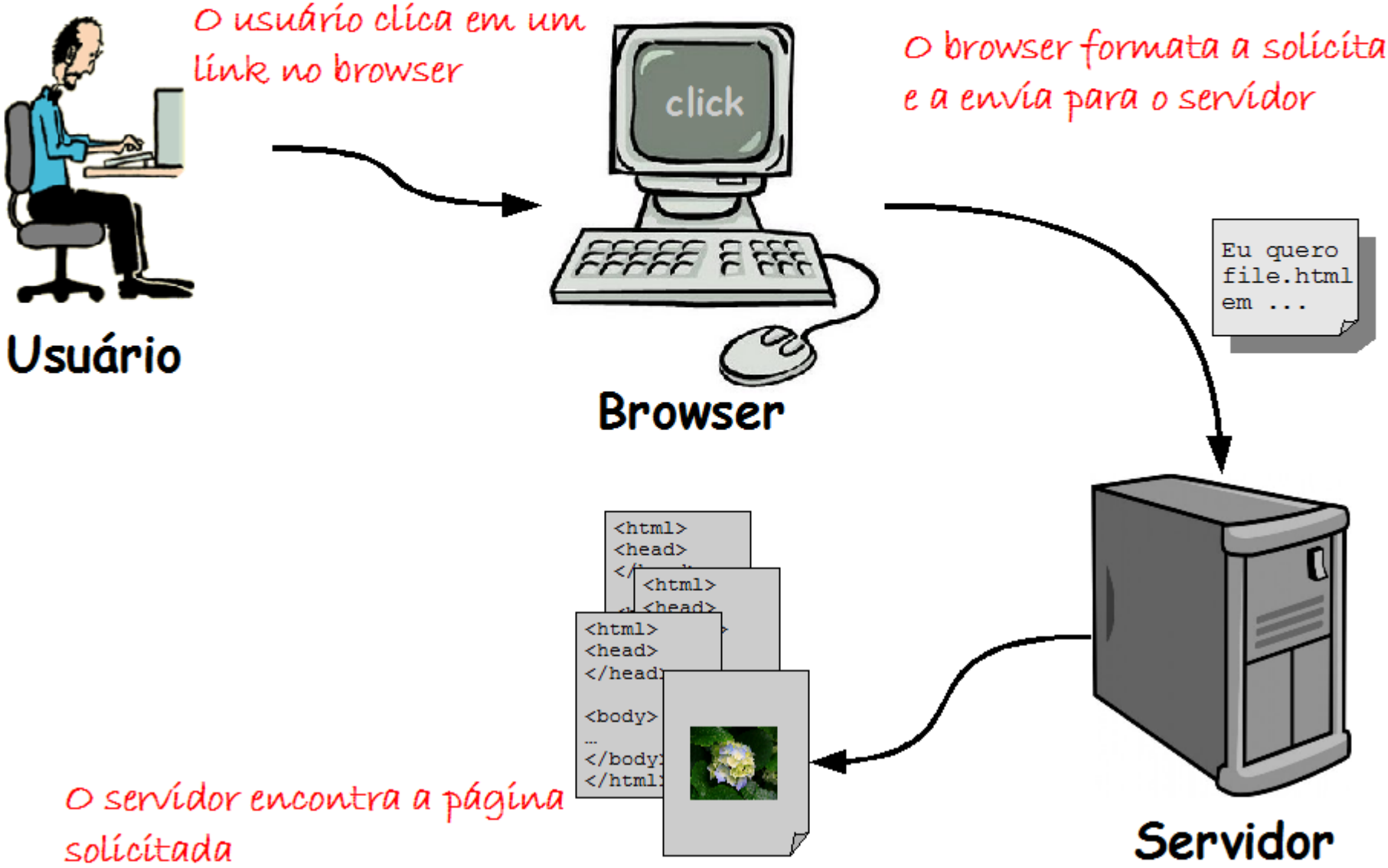
Geralmente o servidor tem muito conteúdo que pode ser mandado para o cliente.

A resposta do servidor contém o documento verdadeiro que o cliente solicitou (ou o código de erro se o pedido não puder ser processado).

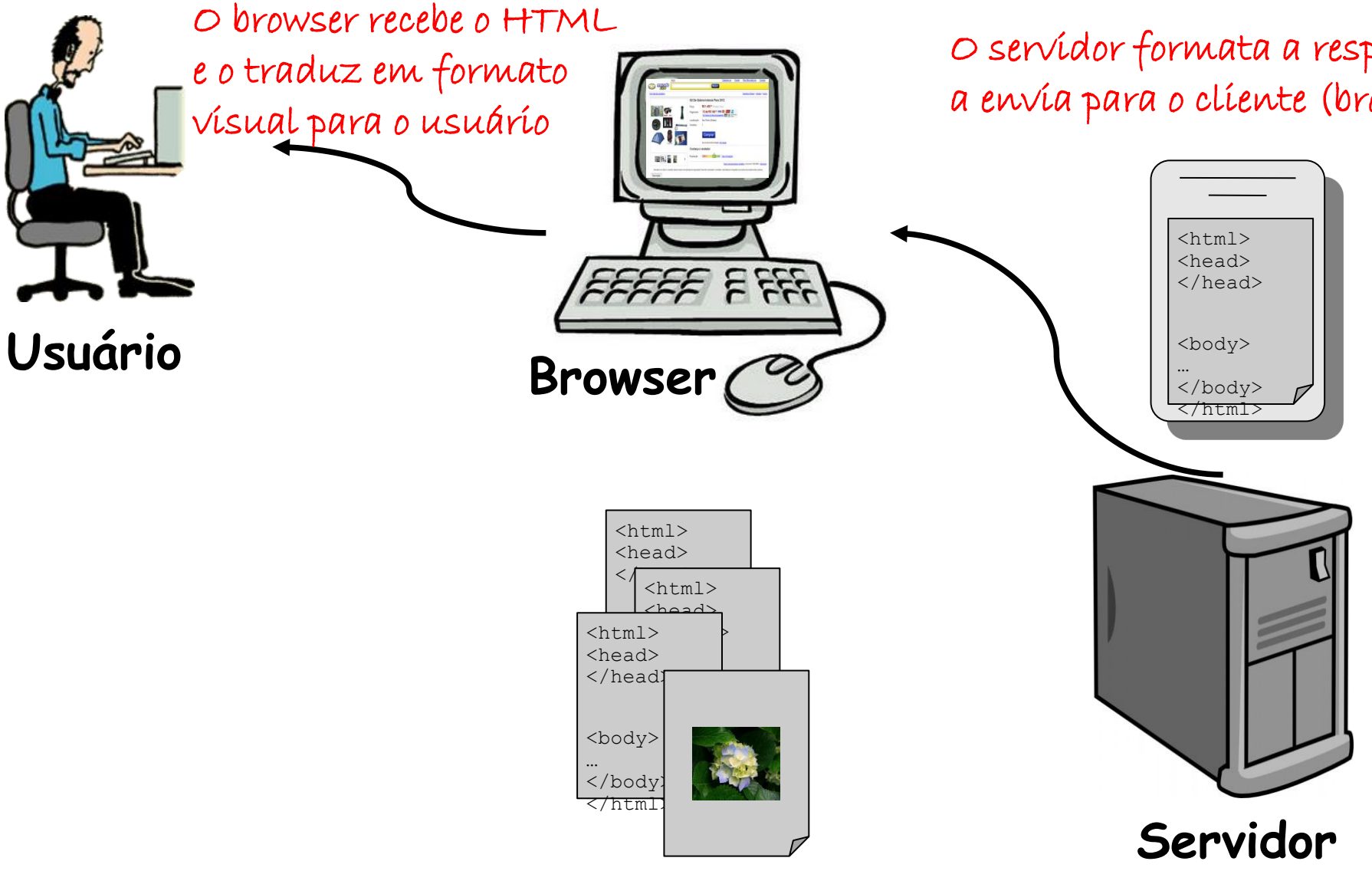
O que faz um Cliente?

Um cliente permite ao usuário fazer solicitações ao servidor, exibindo para ele o resultado do pedido

Exemplo: Cliente Web (1)



Exemplo: Cliente Web (2)



Mas como os clientes e os servidores comunicam-se?

Sábia pergunta, jovem Padawan. Para se comunicarem, eles têm que compartilhar uma linguagem em comum. Na Web, os clientes e os servidores devem falar HTTP e os browsers devem conhecer o HTML.

HTML

- Quando um servidor responde a uma solicitação, ele geralmente envia algum tipo de conteúdo para o browser, para que este possa exibi-lo
- Esse conteúdo é expresso em HTML (*HyperText Markup Language*)
- O HTML diz ao browser como apresentar o conteúdo ao usuário

Exemplo HTML

```
<html>
<!-- Um HTML exemplo -->
<head>
        <title>Uma Página de Login</title>
</head>
<body>
        <h1 align="center">Página de Login</h1>

        <p align="right">
                <img src="pessoa.jpeg" width="130" height="150"/>
        </p>

        <form action="acao" method="post">
                Login: <input type="text" name="param1"/><br/>
        Senha: <input type="text" name="param2"/><br/><br/><br/>

                <center>
                        <input type="submit"/>
                </center>
        </form>
</body>
</html>
```

A

B

C

D

E

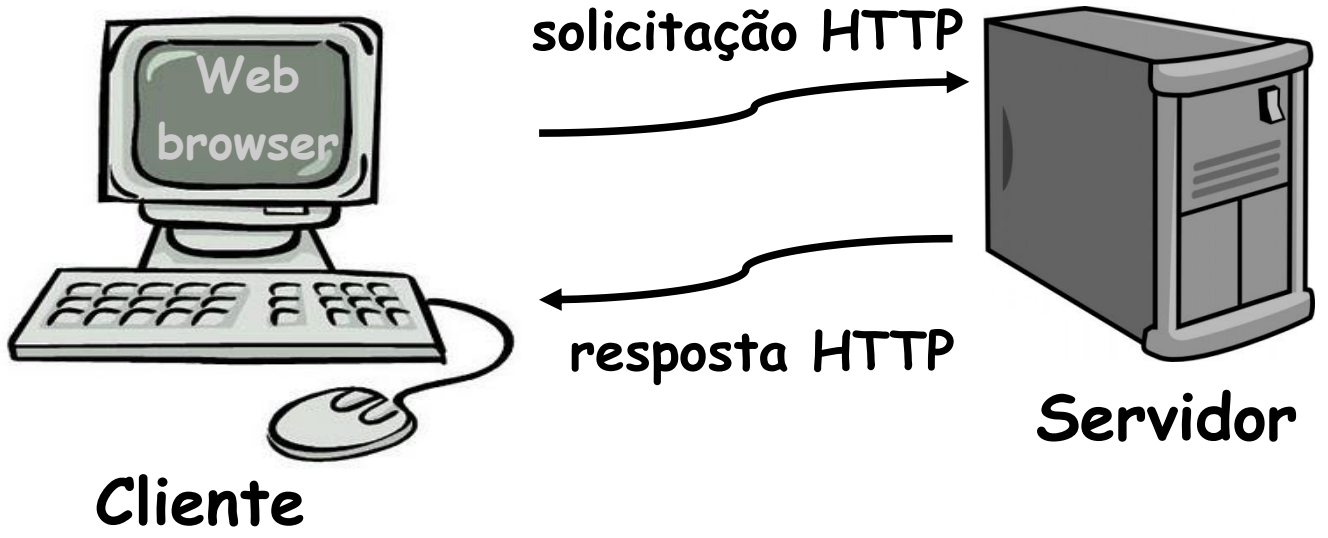
O que o browser mostra...

HTTP

- A maioria das conversas que ocorre na Web entre clientes e servidores é mantida através de um protocolo chamado HTTP
- O cliente envia uma solicitação HTTP e o servidor retorna uma resposta HTTP
- Quando o servidor envia uma página em HTML para o cliente, o faz usando o HTTP

Protocolo HTTP

- O HTTP depende do TCP e do IP para obter a solicitação e a resposta completas de um lugar para o outro
- Como todo protocolo, o HTTP estabelece regras de como devem ser as “conversas” entre clientes e servidores na Web



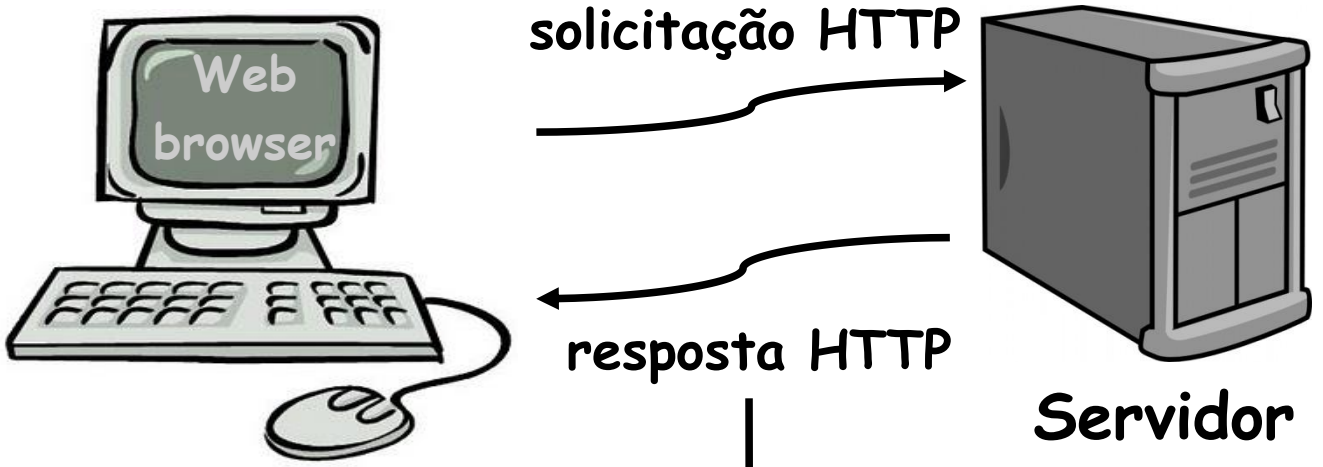
Principais elementos do fluxo de solicitação:

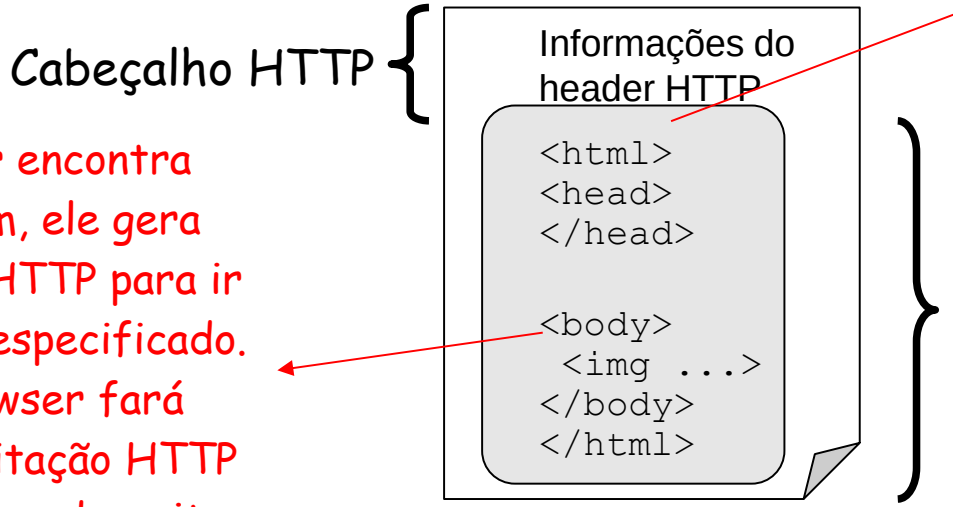
- O método HTTP (a ação a ser executada)
- A página que será acessada (uma URL)
- Os parâmetros do formulário (como argumentos para um método)

Principais elementos do fluxo de resposta:

- Um código de status
- Tipo de conteúdo (texto, imagem, HTML, etc.)
- O conteúdo (o HTML real, imagem, etc.)

HTML está na resposta HTTP





Quando o browser encontra a tag <html> de abertura, ele entra em modo de processamento do HTML e exibe a página para o usuário

Quando o browser encontra uma tag de imagem, ele gera outra solicitação HTTP para ir buscar o recurso especificado. Neste caso, o browser fará uma segunda solicitação HTTP para obter a imagem descrita na tag <img>.

E a solicitação HTTP?

- A primeira coisa que encontramos é o nome do método HTTP
  - Estes não são métodos Java, mas a ideia é semelhante
  - O nome do método informa ao servidor o tipo de solicitação que está sendo feita
- Principais métodos
  - GET
  - POST

GET

- Método mais simples do HTTP
- Serve para pedir ao servidor que consiga um recurso (página HTML, imagem, etc.) e o envie de volta
- Permite passagem limitada de dados por parâmetros (estes são anexados na URL)

POST

- Solicitação mais poderosa
- Permite enviar dados ao servidor para serem processados e solicitar uma resposta
- Permite envio de dados sem as limitações do GET (os parâmetros são passados no corpo da solicitação)

URL

Porta TCP (pode ser omitido caso o servidor web esteja rodando na porta 80, como neste caso)

Protocolo

```
http://www.site.com.br:80/pasta/pagina.html
```

Servidor

Caminho que indica a localização do recurso no servidor

Páginas Estáticas

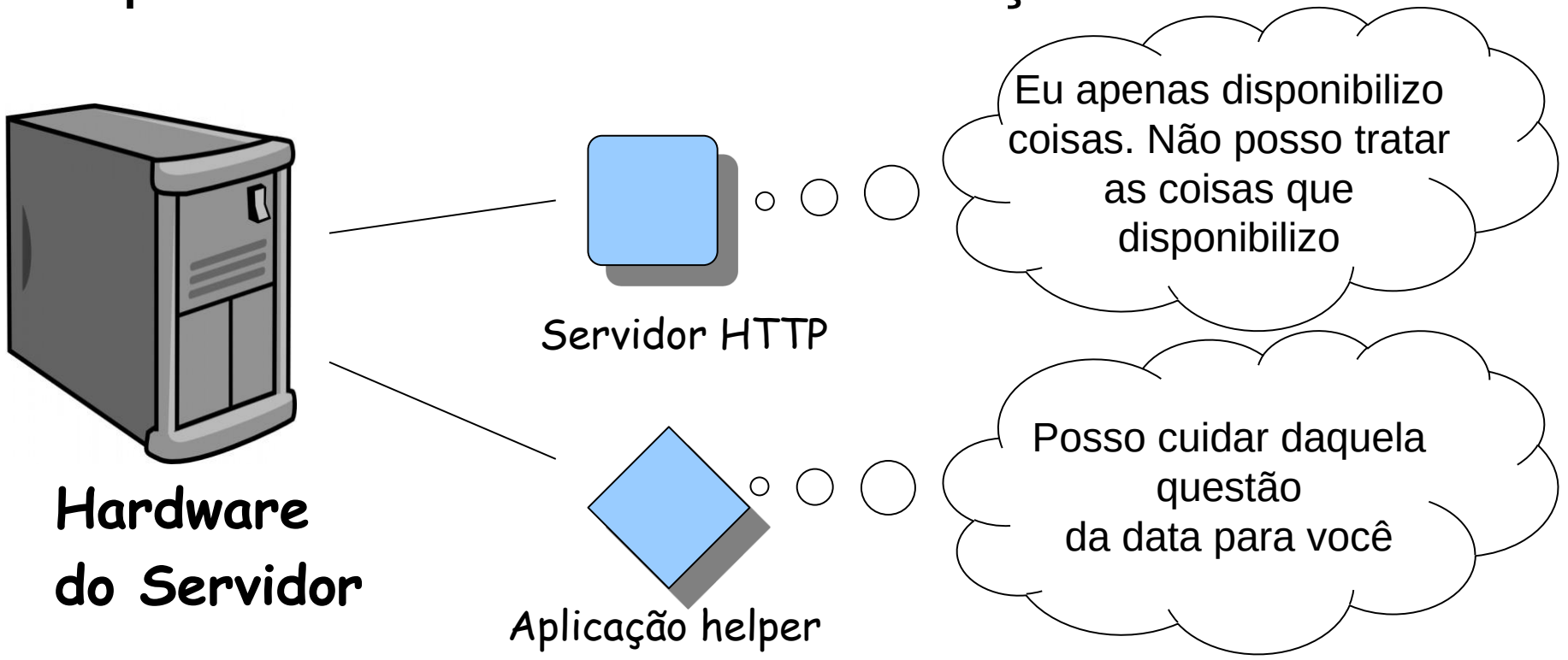
- Os servidores web sabem muito bem servir os clientes com conteúdo estático
- Esses recursos são fixos ao longo do tempo
  - O recurso é enviado ao cliente exatamente da mesma forma como está armazenado no servidor

Mas se eu quiser, digamos, que a hora atual
apareça na minha página? E seu eu quiser uma página com conteúdo dinâmico?
Eu não poderia ter algo como uma variável no HTML?

```
<html>
<body>
    A hora atual é [insertTimeOnServer]
</body>
</html>
```

Dinamismo no Servidor

- Os servidores HTTP não sabem lidar com conteúdo dinâmico
- Eles precisam terceirizar o serviço



Quando o servidor terceiriza?

- Solicitação de conteúdo dinâmico
  - Entenda-se conteúdo dinâmico processado no servidor, o que não inclui Javascript (que é processado no cliente)
- Processamento de dados de formulários

Aplicações Helper

- Podem ser implementadas de várias formas:
  - Programas CGI (*Common Gateway Interface*)
    - Programas em C, scripts em Perl, Python, PHP, etc.
    - Programas executados pelo servidor Web que jogam na saída padrão código HTML
  - Módulos de servidores Web
    - A aplicação helper é incluída no servidor Web como um módulo
    - O PHP pode funcionar dessa forma
  - **Plataforma Java EE**
    - **Servlets, JSPs, JSFs, e outros, são executados por uma aplicação *container* (Tomcat, etc.)**

MVC

- ***M****odel* ***V****iew* ***C****ontroller*
- Separa a lógica da aplicação (modelo) e a sua visualização (view)
- Modelo em três camadas
  - View
  - Modelo
  - Controlador

View

- Responsável pela apresentação (interface) da aplicação
- Recebe o estado do modelo através do controlador
  - Em muitos casos, o controlador apenas coloca os dados do modelo em um lugar onde a View possa encontrá-los
- Também é responsável por receber os dados de entrada do usuário e repassá-los ao controlador

Controlador

- Recebe os dados da solicitação do usuário e interpreta o seu significado para o modelo
- Obriga o modelo a se atualizar a partir dos dados de entrada
- Disponibiliza o novo estado do modelo para a view

Modelo

- Abriga a verdadeira lógica e o estado do modelo de negócios da aplicação
  - Ele conhece as regras para obtenção e atualização dos dados da aplicação
  - Exemplo: O conteúdo de um carrinho de compras (e as regras sobre o que fazer com isso) seria parte do modelo em uma arquitetura MVC

MVC na Plataforma Java

- Controlador
  - Servlets
- View
  - JSP
  - HTML, CSS, Javascript, ...
- Modelo
  - POJO
    - *Plain Old Java Object*
      - Classes Java tradicionais

Exemplo

- Aplicação de soma de números inteiros
  - Usuário informa dois números inteiros em um formulário web e a aplicação mostra o resultado da soma dos números
- Projeto *Aula01-Exemplo1*

Instalando a Aplicação

- Os arquivos Java compilados (controlador e modelo) devem ser colocados na pasta WEB-INF/classes de nossa aplicação
- O arquivo JSP e o arquivo CSS devem ser colocados na pasta raiz da aplicação
- O arquivo web.xml é colocado na pasta WEB-INF