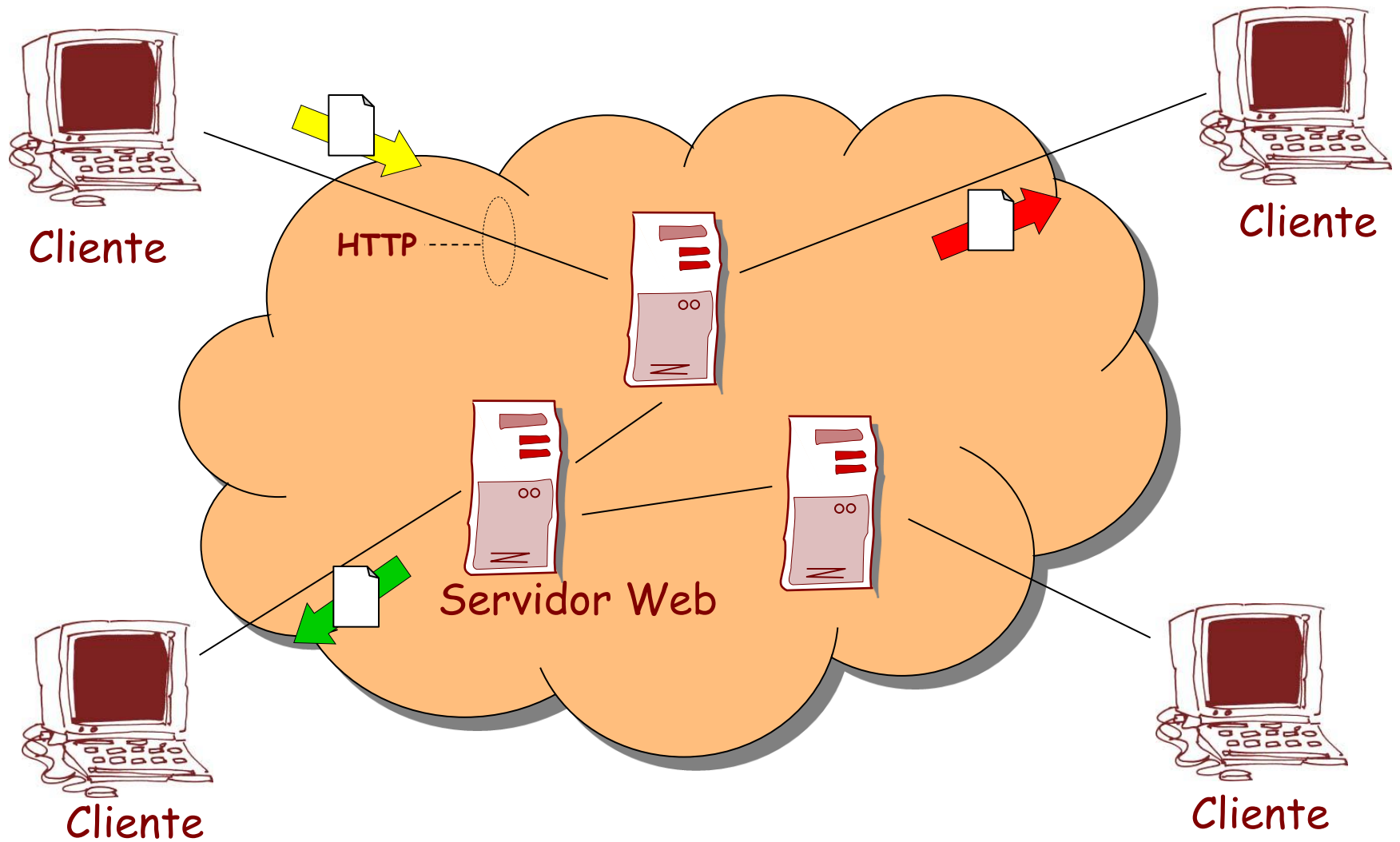


Introdução e Visão Geral

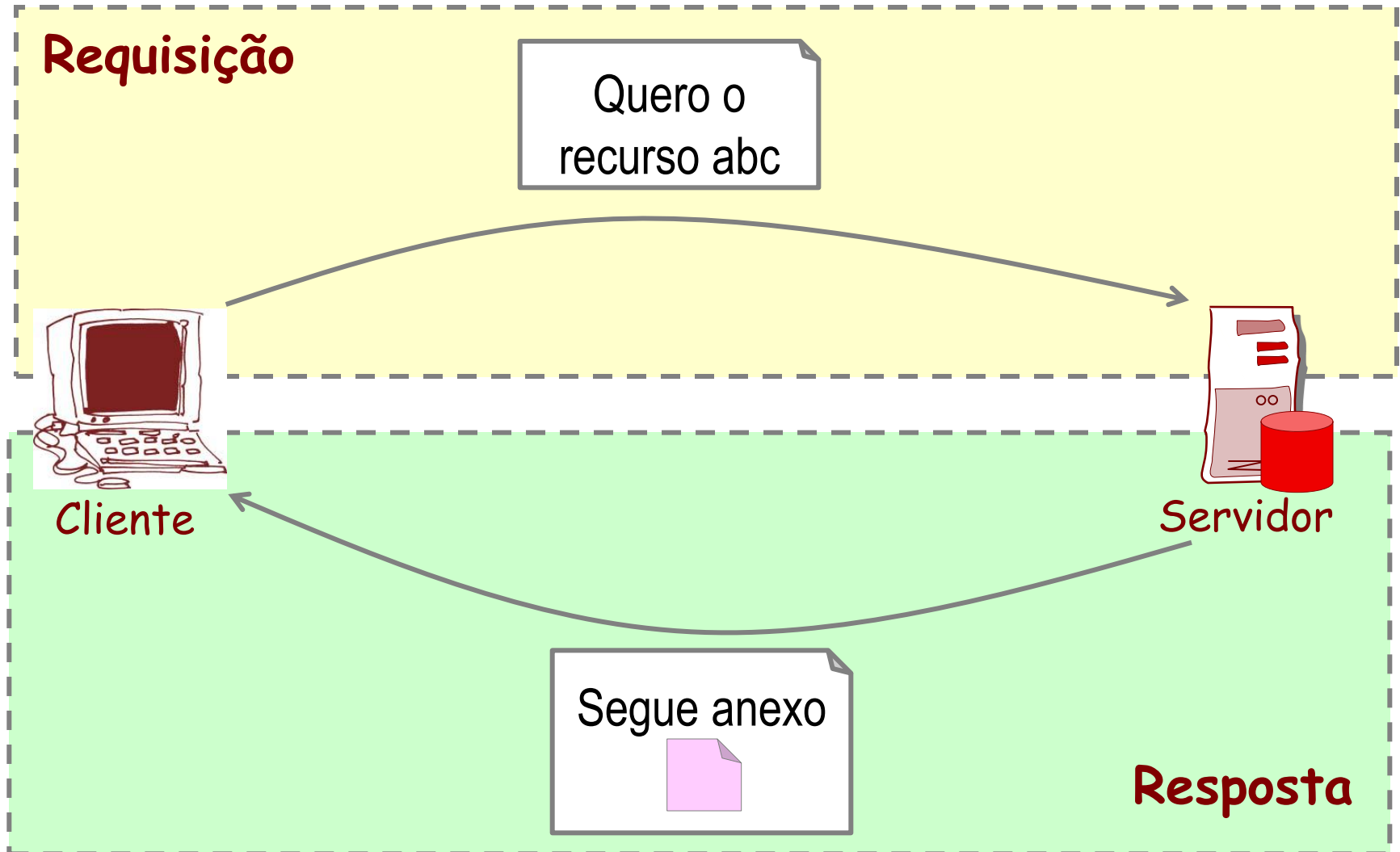
Protocolo HTTP
Aplicações web
Servlets e JSP



A web



Clientes e Servidores Web

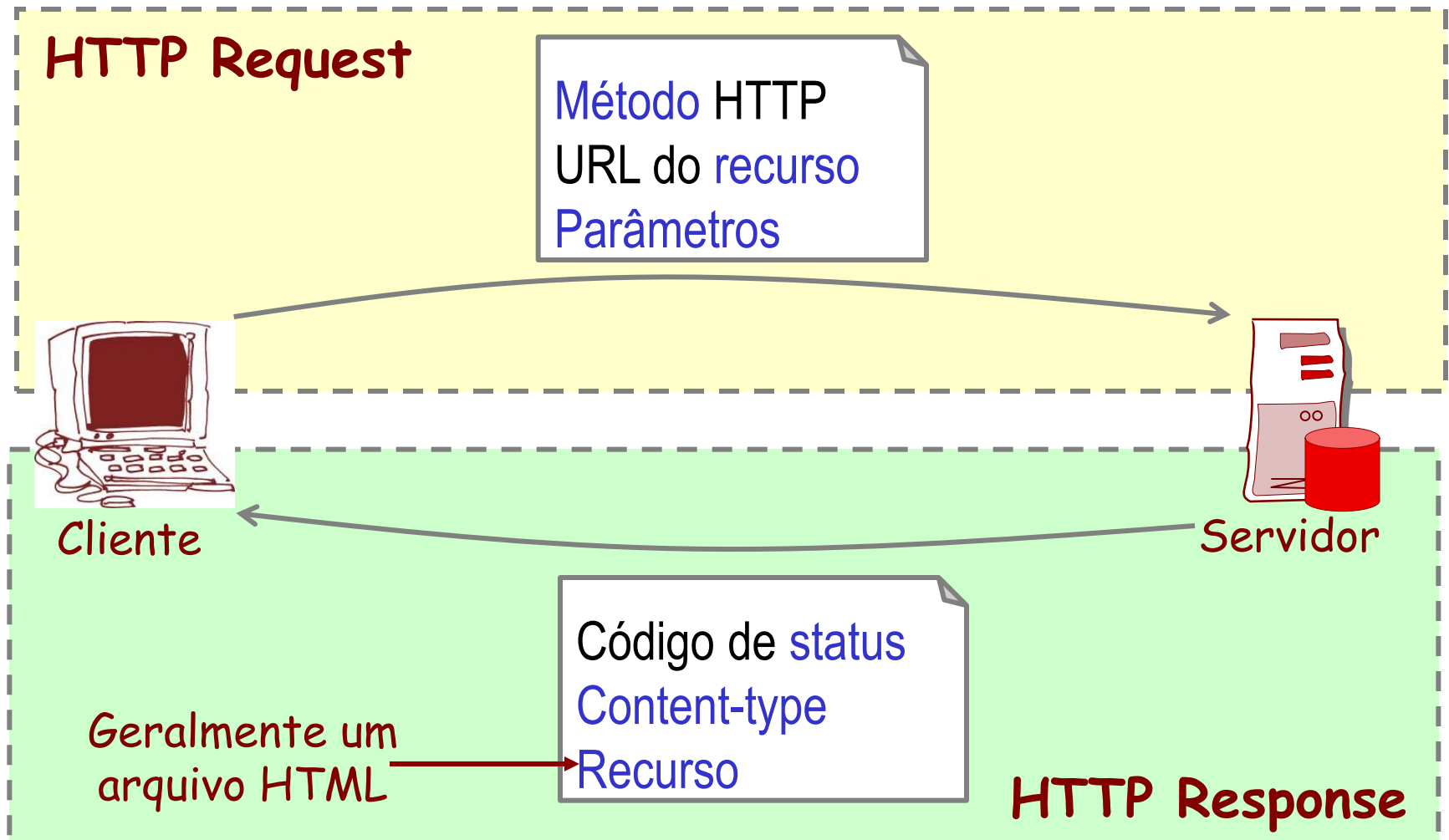


Cientes e Servidores Web

■ Glossário

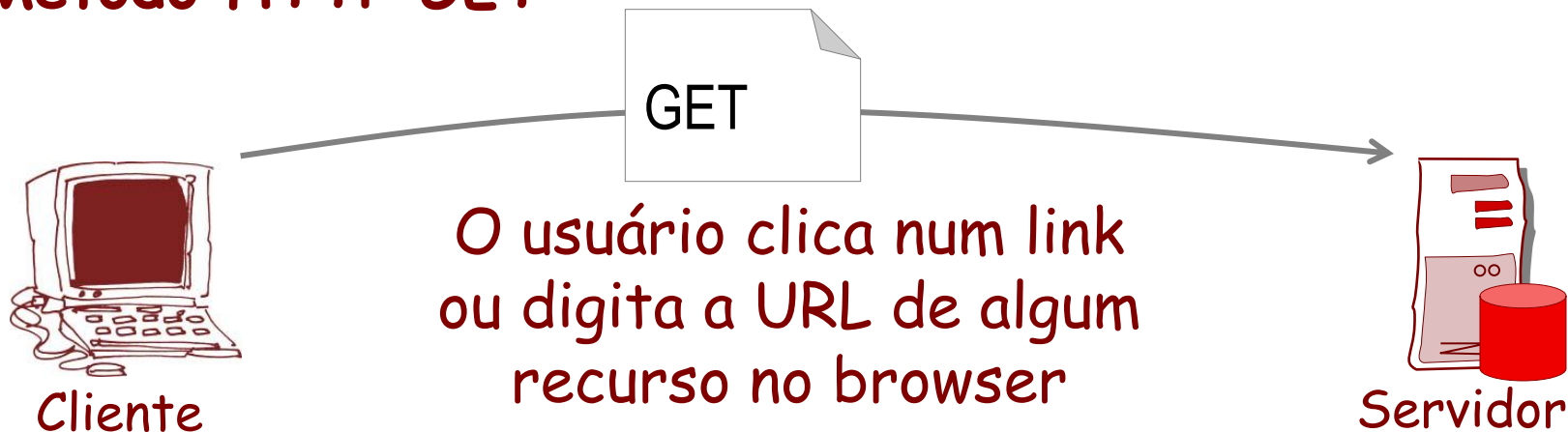
- **HTTP**: protocolo que define o formato de como o cliente pede recursos ao servidor e de como o servidor responde
- **HTML**: um dos formatos em que um recurso pode estar, geralmente contendo informações que serão formatadas pelo cliente (browser)
- **Requisição**: frame HTTP com um pedido de recurso a um servidor
- **Resposta**: frame HTTP com a resposta (sucesso ou erro) a uma requisição

Protocolo HTTP



Protocolo HTTP

Método HTTP GET



Método HTTP POST



Protocolo HTTP

- Diferenças entre o POST e o GET
 - A quantidade de dados que se pode passar no GET é limitada pelo servidor (256 bytes, em geral)
 - Os dados enviados via GET são exibidos na barra de endereços do browser (expõe dados secretos)
 - Não é possível marcar (favoritos) o resultado de um request que usa o POST

Protocolo HTTP

■ Detalhes de um GET:

método	path do recurso	dados	versão do protocolo
GET	/loja/consulta.jsp?	id=123&tipo=10	HTTP/1.1

```
Host: www.lojatal.com.br
User-Agent: IE/7.0
Accept: text/html,application/xml,text/html,text/
plain,image/gif,image/jpeg
Connection: keep-alive
Cookie: jsessionid=12abd4f67cc34
```

→ parâmetros
do request

Protocolo HTTP

■ Detalhes de um POST:

método	path do recurso	versão do protocolo
POST	/loja/cadastro.do	HTTP/1.1

Host: www.lojatal.com.br
User-Agent: IE/7.0
Accept: text/html,application/xml,text/html,text/plain,image/gif,image/jpeg
Connection: keep-alive
Cookie: jsessionid=12abd4f67cc34

nome=Frederico+Costa&curso=DSI&turma=26012

dados

Protocolo HTTP

■ Detalhes de uma resposta:

versão código e mensagem de resposta
HTTP/1.1 200 OK

Content-type: text/html obrigatório

Content-length: 435

Date: Sat, 11 Set 2005 03:33:33 GMT

Server: Apache/1.1

Conection: close

<html>

...

</html>

código HTML (recurso solicitado)

Protocolo HTTP

■ Detalhes de uma resposta:

versão código e mensagem de resposta

HTTP/1.1 **200 OK**

Content-type: application/zip obrigatório

Content-length: 435

Date: Sat, 11 Set 2005 03:33:33 GMT

Server: Apache/1.1

Conection: close

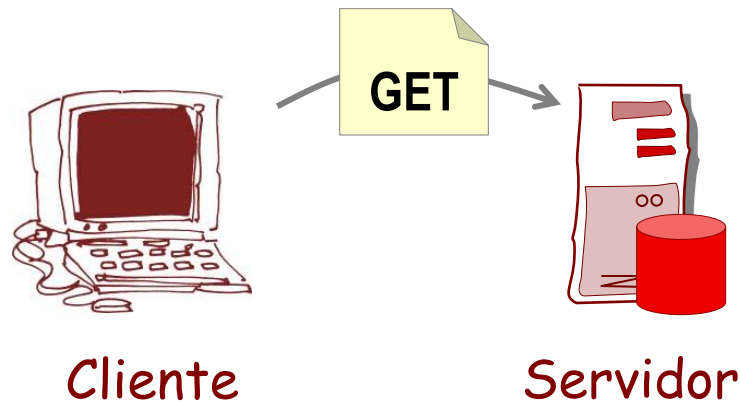
A10BC6F70B
673648FFD2
8994000AE4

←

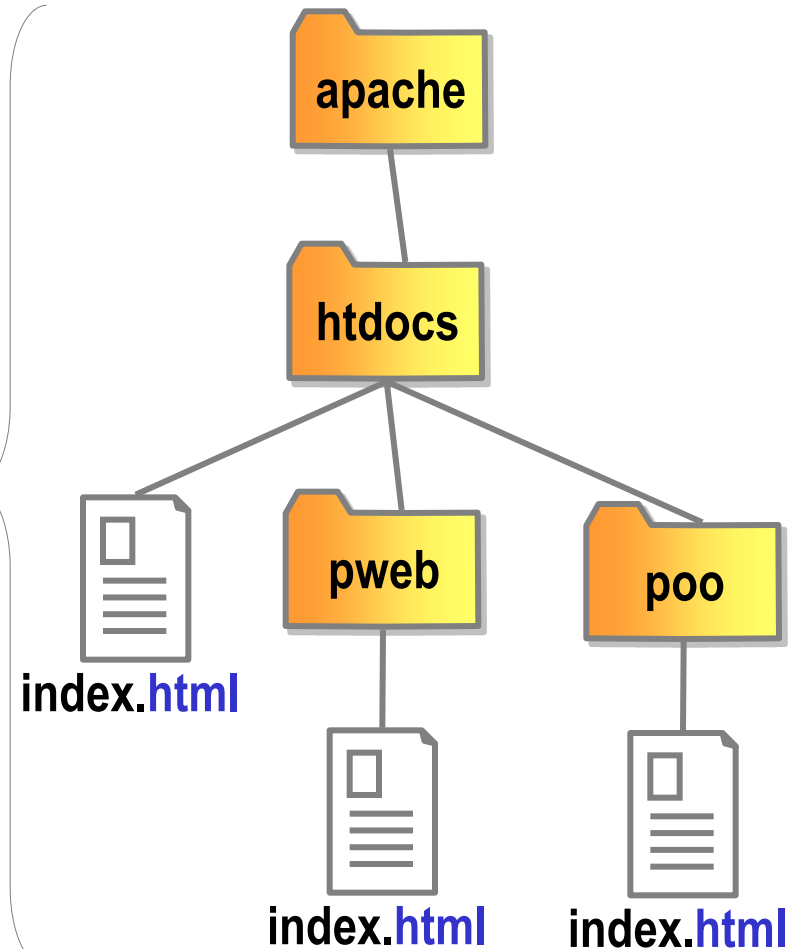
arquivo Zip (recurso solicitado)

Protocolo HTTP

■ Servindo conteúdo estático

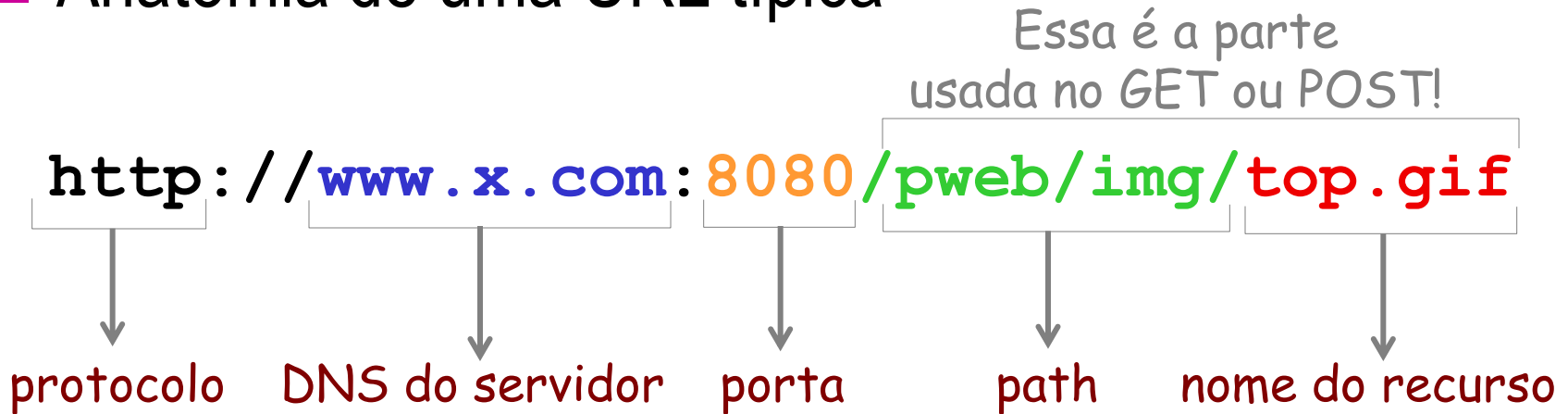


Mapeando URL no path do recurso:
`www.x.com/pweb/index.html`



Protocolo HTTP

■ Anatomia de uma URL típica



- O path é contado a partir de um diretório base que é específico de cada servidor web (o do apache é htdocs)
- Se o recurso não estiver no path indicado ou o nome dele estiver errado, o servidor devolve uma resposta de erro

Protocolo HTTP



E se eu quiser obter uma
informação gerada
dinamicamente ou
processar dados enviados
por um formulário HTML?

Protocolo HTTP



É aí que entra em ação a segunda habilidade de um servidor web, além de servir recursos estáticos: **executar aplicações externas**

Aplicação CGI

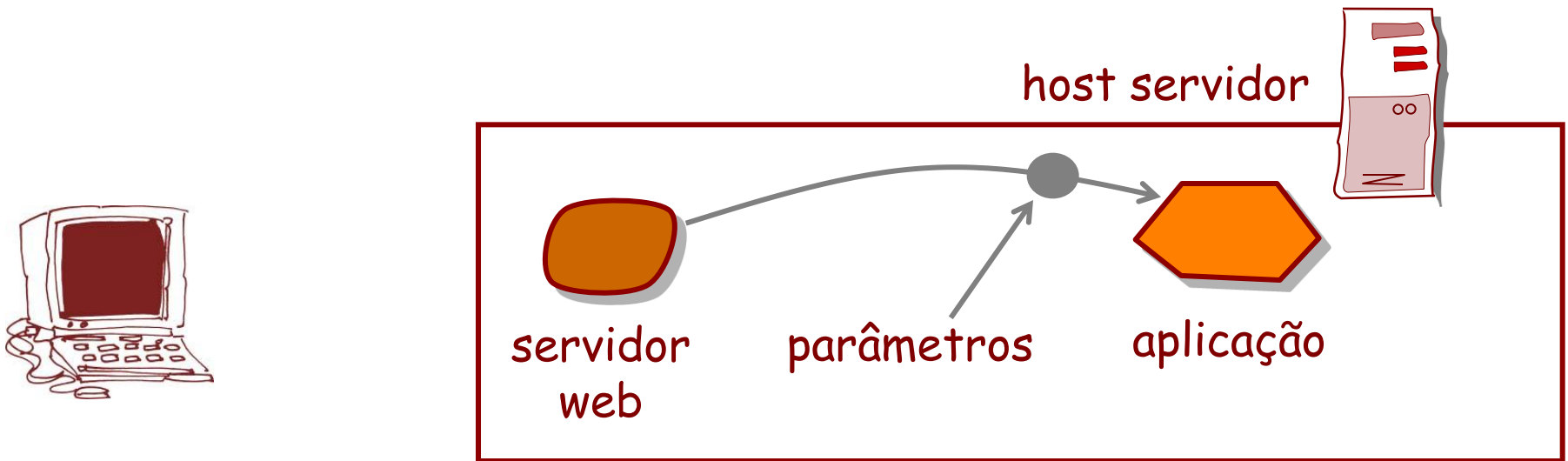
- 1 O servidor web recebe um pedido de execução de uma aplicação (via GET ou POST)

dados de formulários



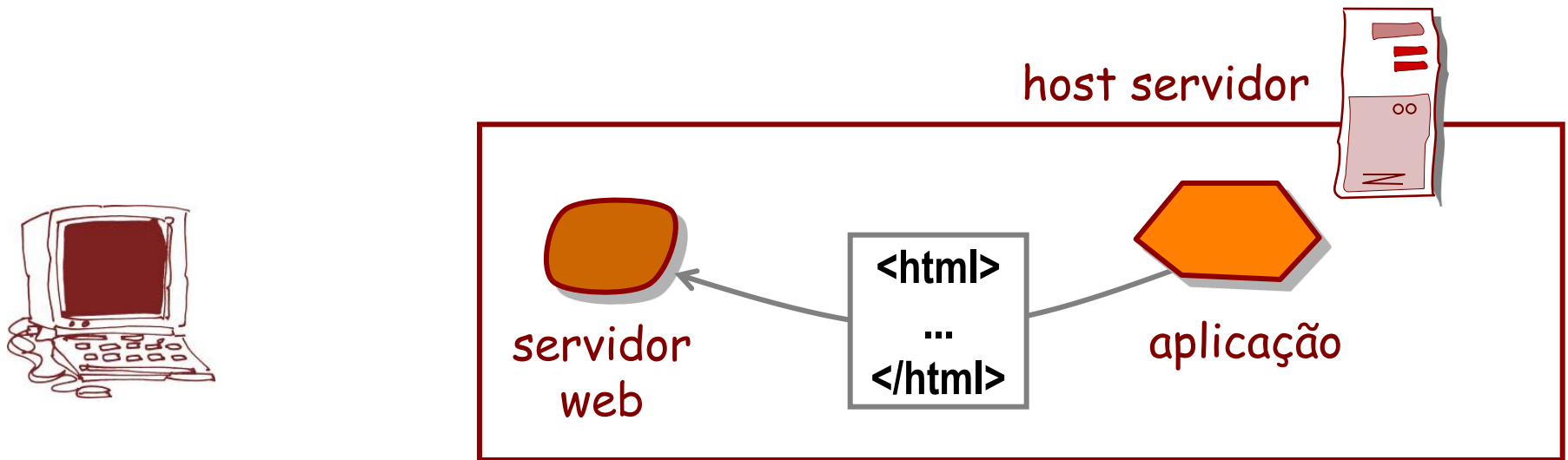
Aplicação CGI

- 2 O servidor web cria um processo para a aplicação e passa alguns parâmetros em variáveis especiais



Aplicação CGI

- 3 A aplicação executa e devolve o resultado (código HTML) para o servidor web pela saída padrão



Aplicação CGI

- 4 O servidor web monta um frame de resposta HTTP, põe o HTML dentro dele e devolve para o cliente



Comparativo

CGI

- Um processo para cada requisição
- Podem ser escritos em qualquer linguagem
- Você está sozinho, vire-se!

Servlets

- Threads para cada requisição
- São classes Java especiais
- Executam sob um container J2EE e têm toda a ajuda e proteção dele

Codificando e implantando um servlet

- Escreva uma classe Java que herde de HttpServlet

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

public class AloMundoServlet extends HttpServlet {
    public void doGet(HttpServletRequest req,
                       HttpServletResponse resp)
        throws IOException, ServletException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.println("<h1>Alo, Mundo!</h1>");
    }
}
```

É assim que ele serve o GET

Tipo MIME da resposta

Tudo que ele escrever aqui vai para o servidor web

HTML gerado pelo servlet

Codificando e implantando um servlet

- Escreva o descritor da aplicação web: **web.xml**

```
<webapp>
  <servlet>
    <servlet-name>Alo</servlet-name>
    <servlet-class>AloMundoServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Alo</servlet-name>
    <url-pattern>/show.do</url-pattern>
  </servlet-mapping>
</web-app>
```

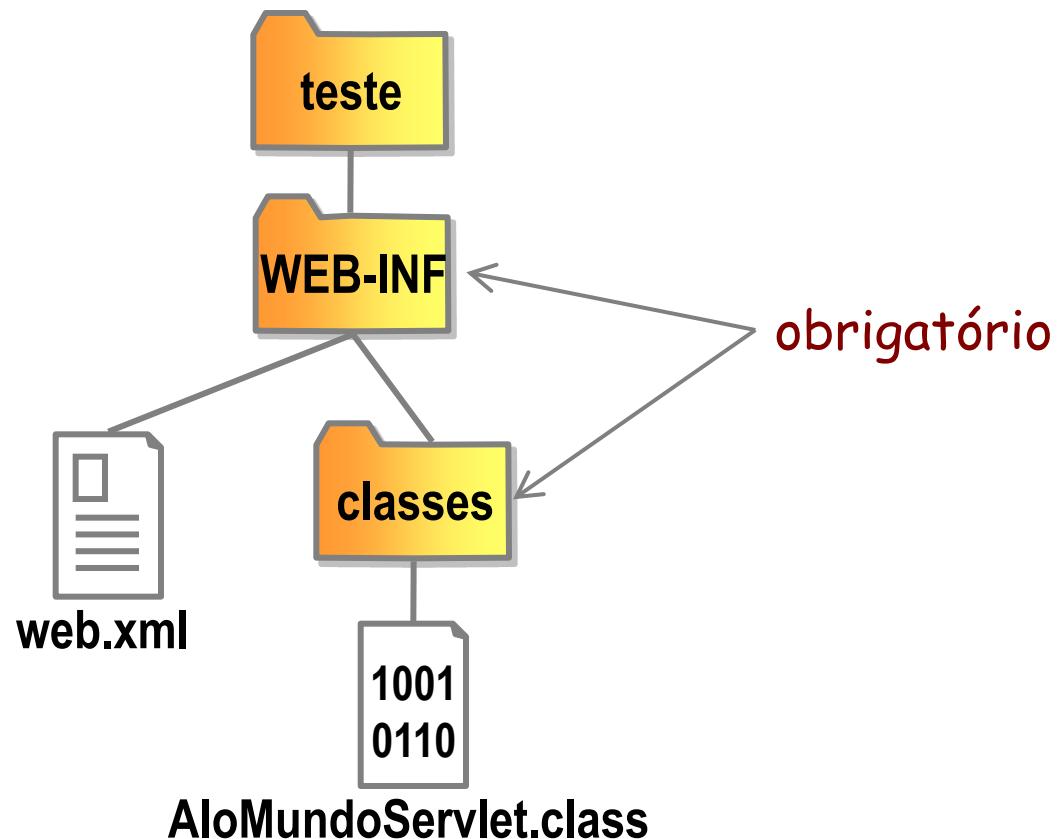
Nome interno do servlet

Nome da classe

Como o servlet será chamado na URL

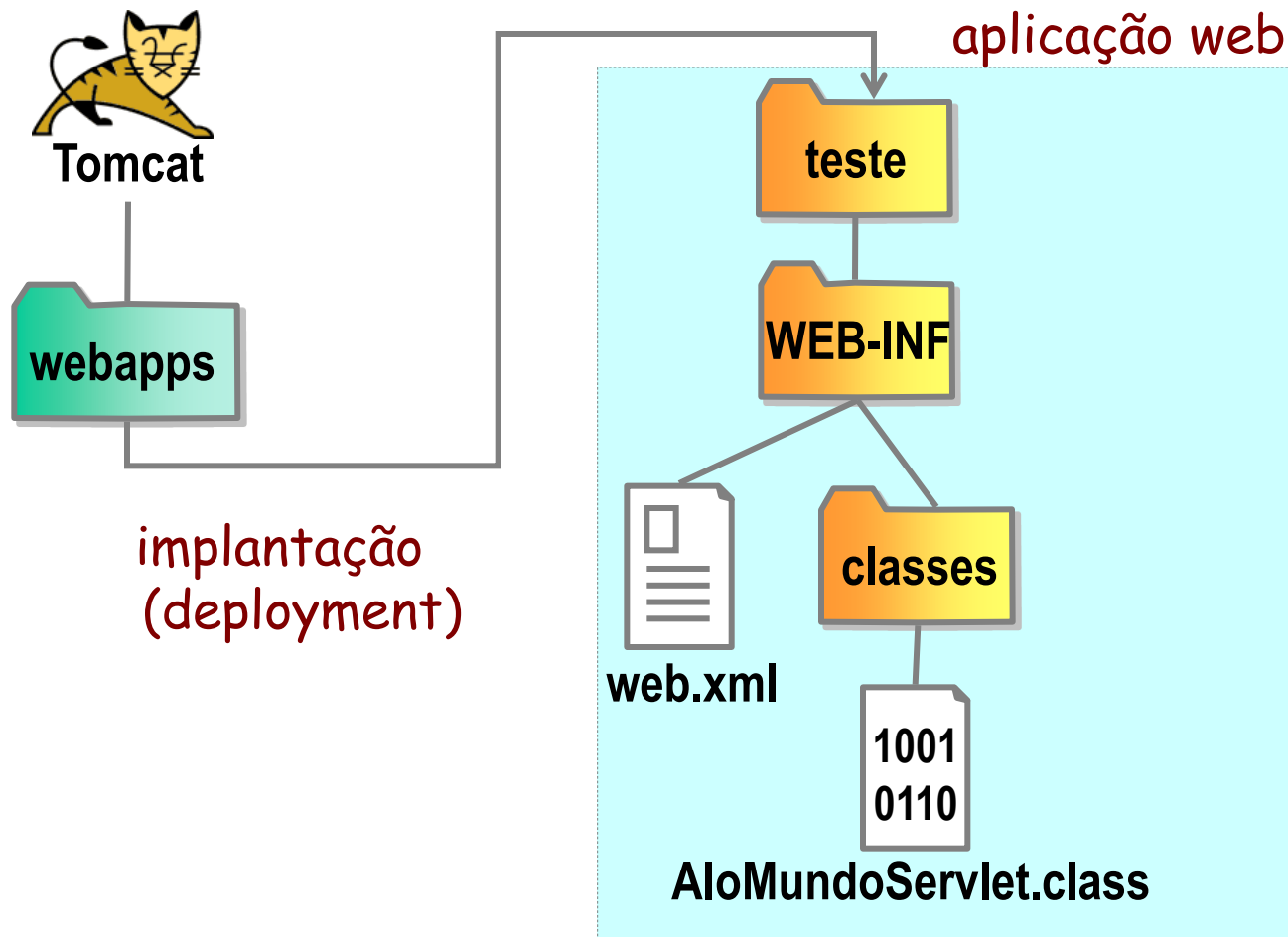
Codificando e implantando um servlet

- Compile o servlet e crie a seguinte estrutura de diretórios:



Codificando e implantando um servlet

- Implante a estrutura no container



Executando a aplicação web

- Ponha o servidor web () no ar (detalhes depois)
Tomcat
- Abra o browser e aponte uma URL para o servlet

`http://localhost:8080/teste/show.do`

Nome da aplicação web
(diretório base)

Nome "URL" do
servlet

- O servlet executa e o browser recebe o HTML
`<h1>Alo, Mundo</h1>` para ser exibido

JavaServer Pages




Legal, mas... e esse código HTML dentro de um String Java? E se eu quiser mudar o *layout* da minha aplicação, tenho que editar código Java para isso?

JavaServer Pages

- JSP é código Java dentro de um HTML

```
<html>
<body>
<h1>Pagina Pessoal</h1>
Nome do cliente ${cliente.nome}
</body>
</html>
```

Uma expressão JSP



- Agora é o *web designer* que tem que aprender Java?
- Nem tanto, os elementos JSP são parecidos com outros de soluções parecidas como ASP, PHP e outros.

JavaServer Pages

■ O JSP na estrutura:

