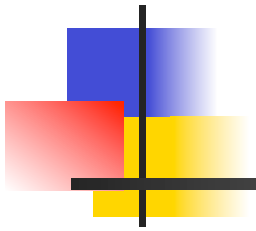
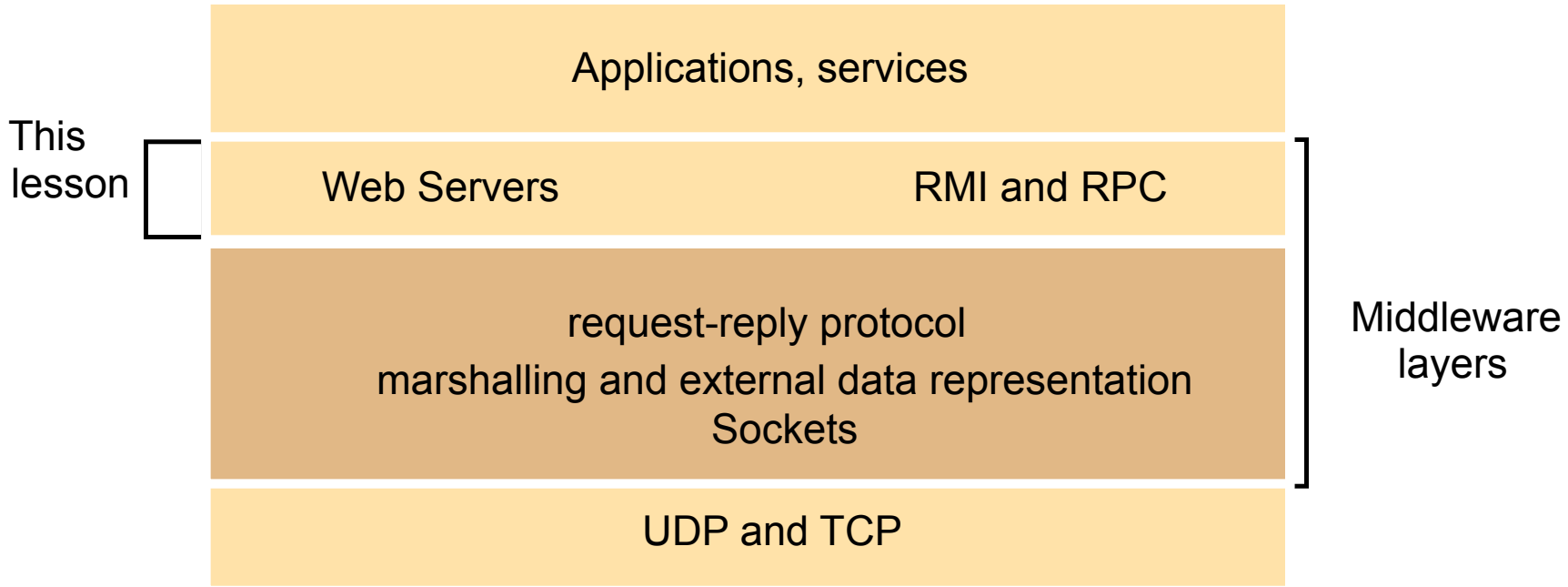


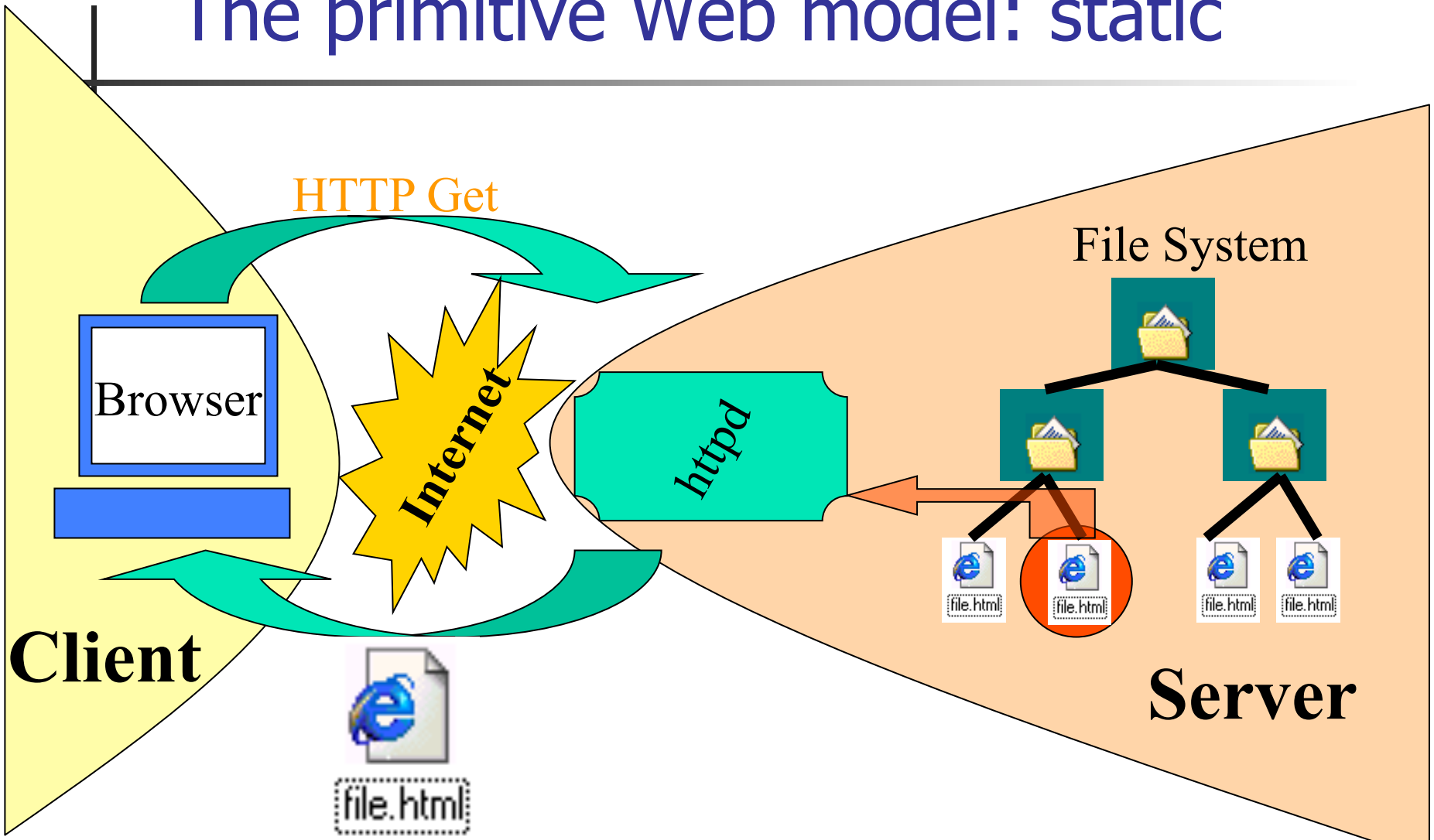
Web Architectures

Adapted from Marco Ronchetti,
corso "Sistemi Distribuiti: progettazione"

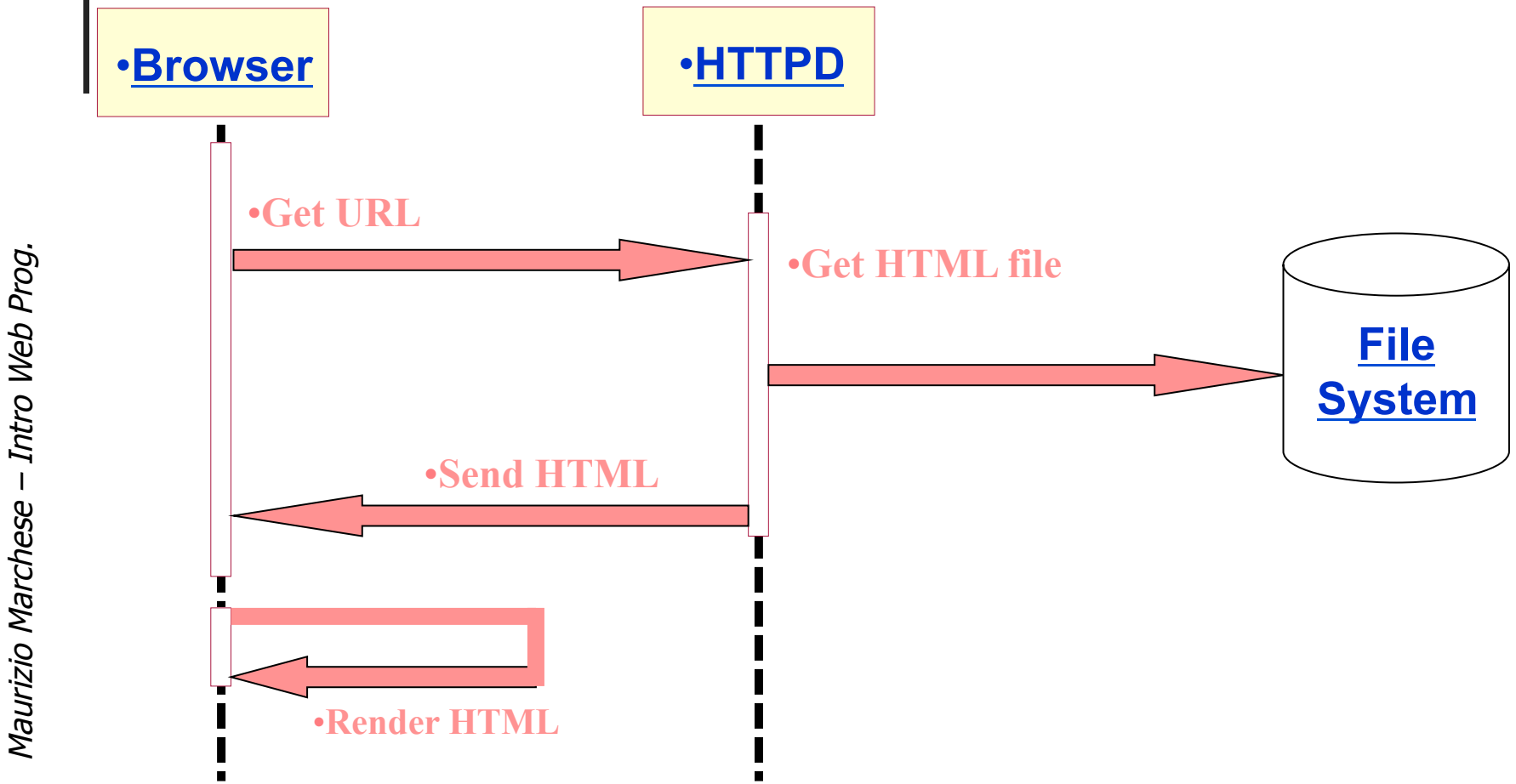
Middleware layers



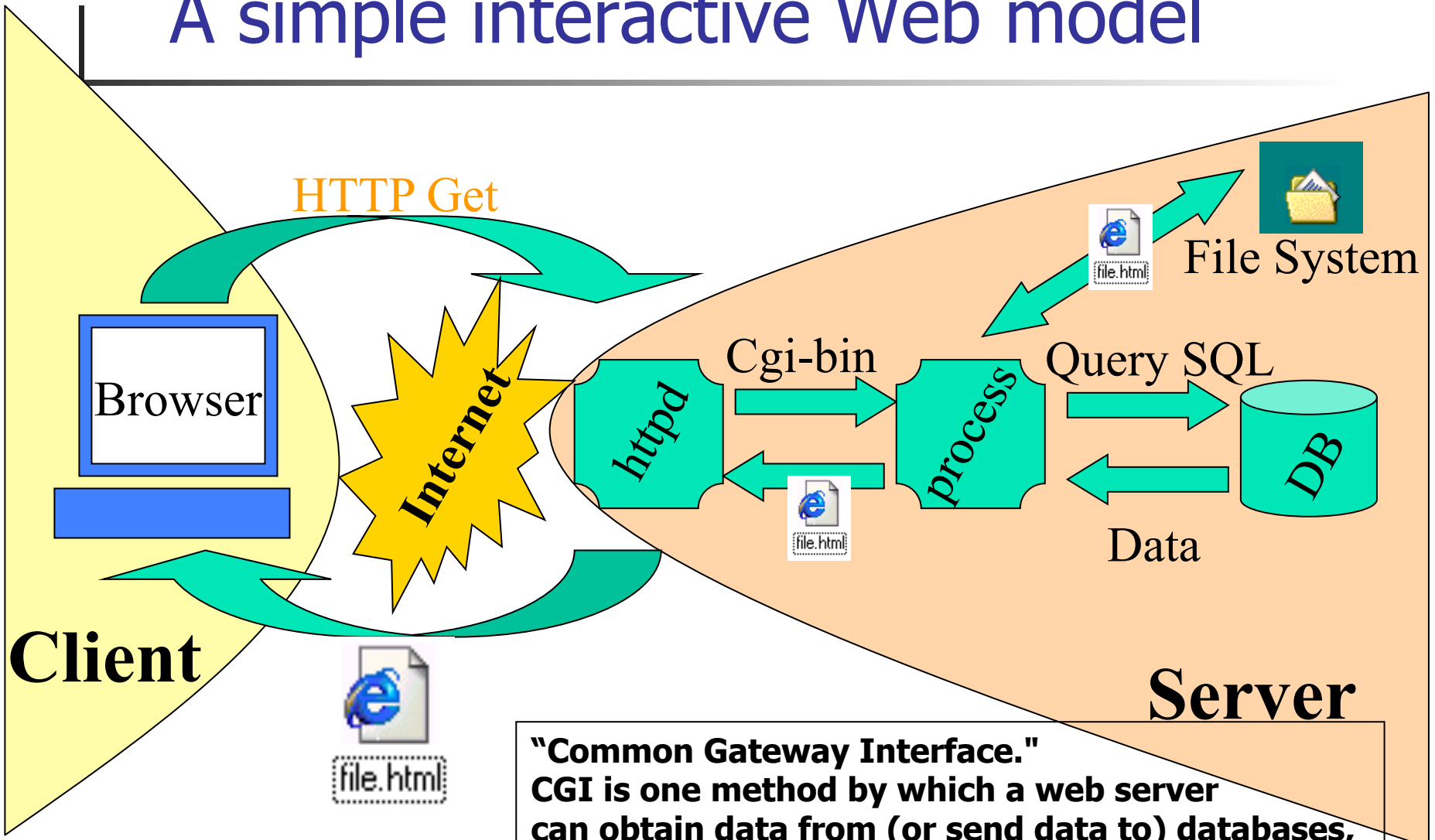
The primitive Web model: static



The primitive Web model: static

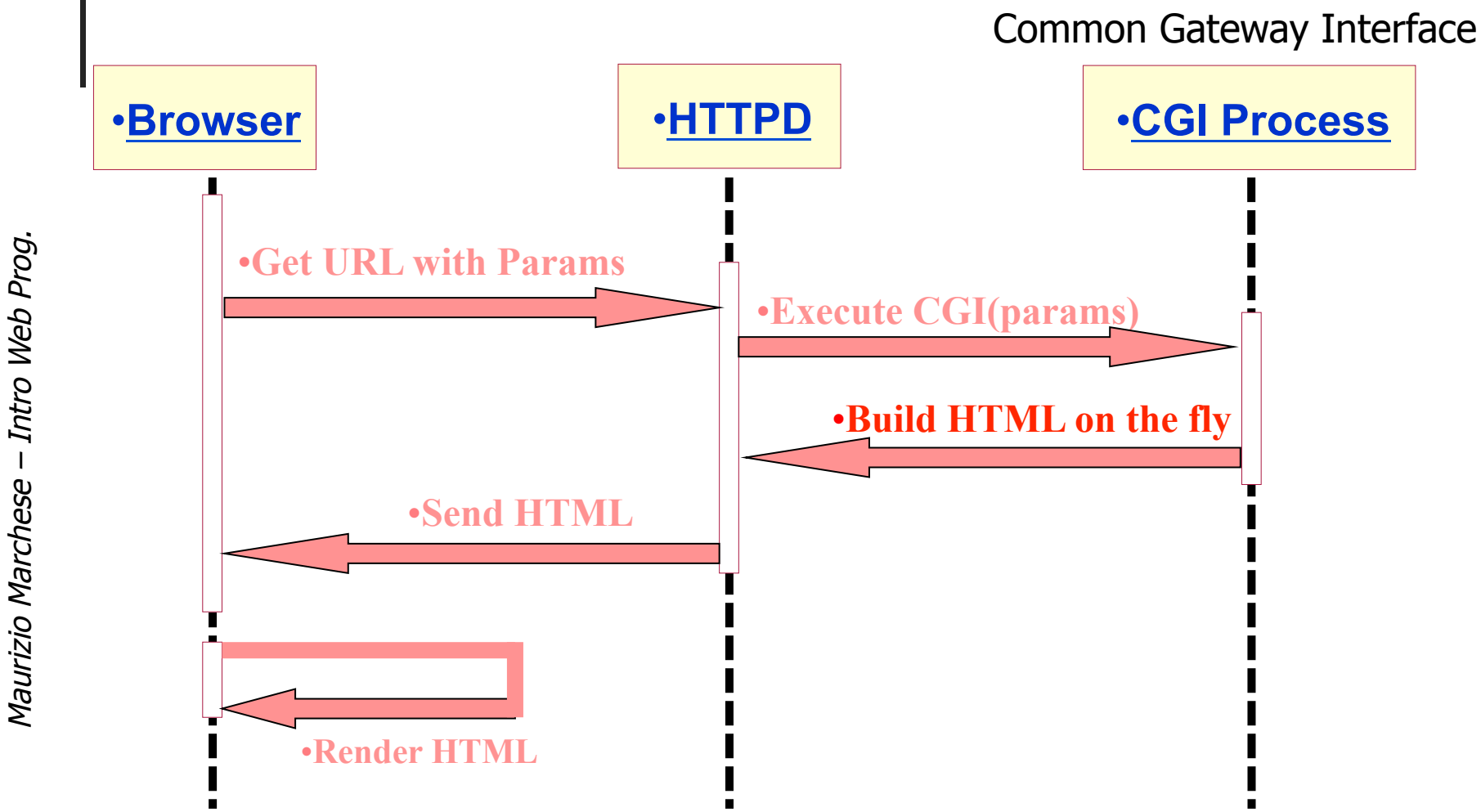


A simple interactive Web model

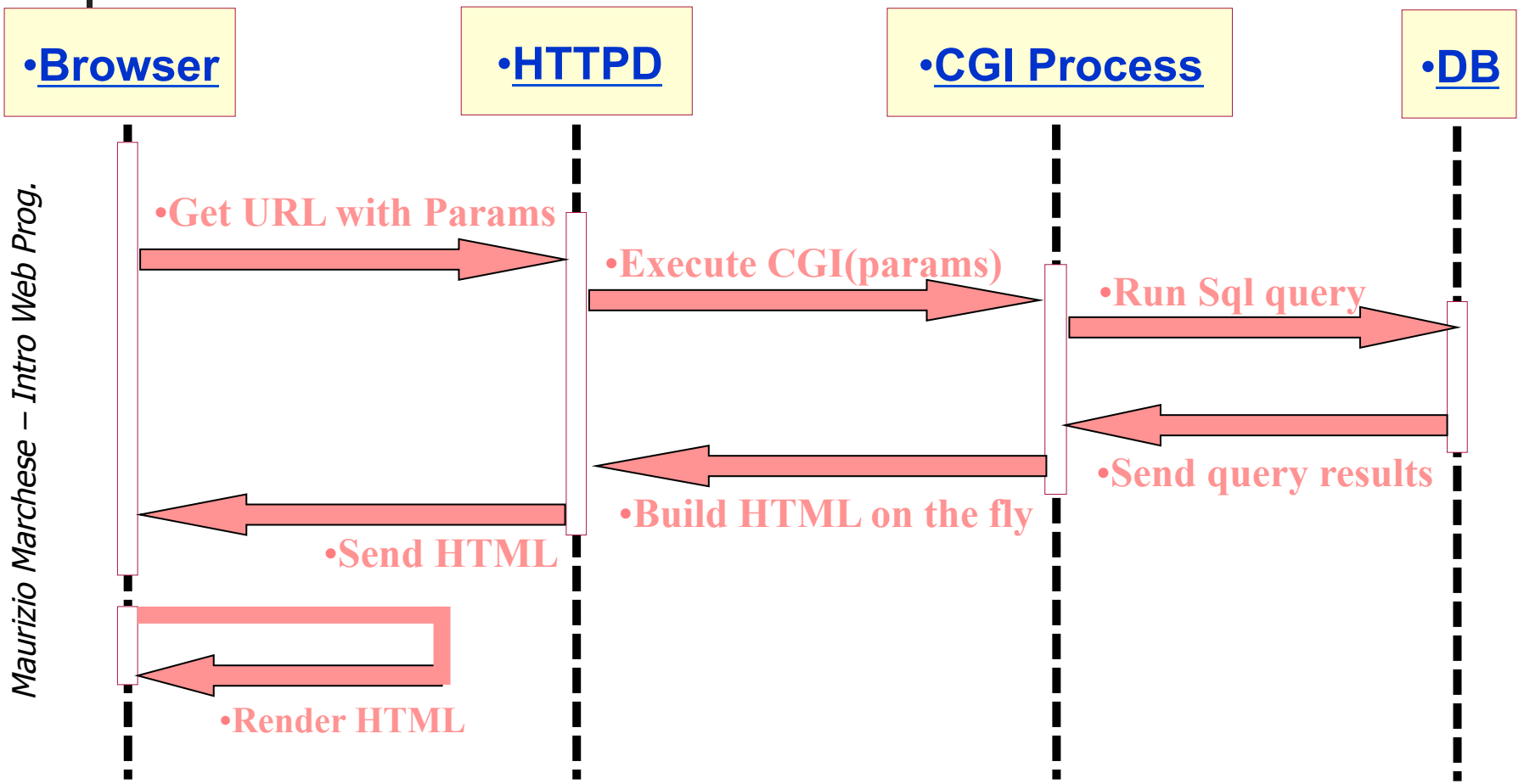


"Common Gateway Interface."
CGI is one method by which a web server can obtain data from (or send data to) databases, documents, and other programs, and present that data to viewers via the web.

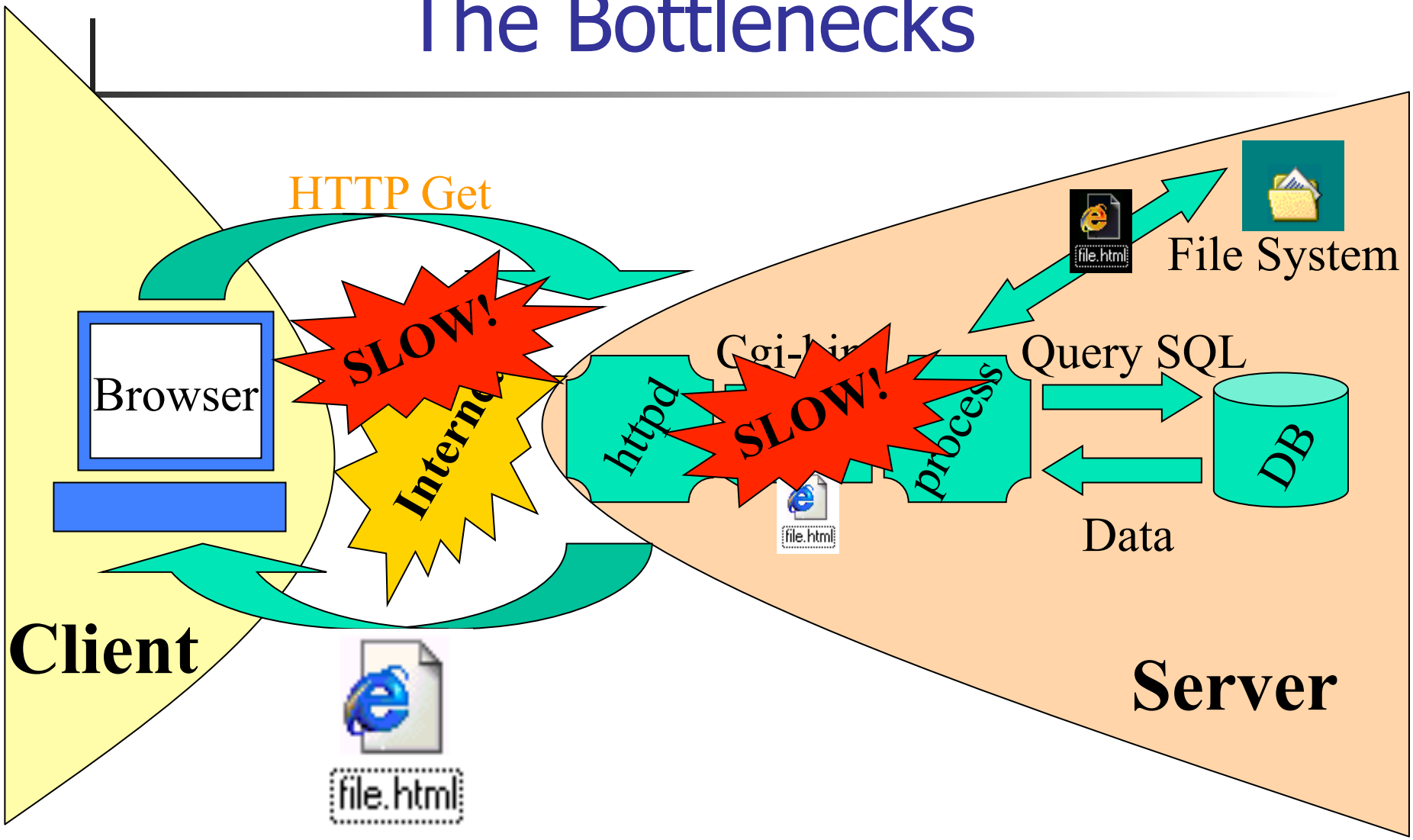
A simple interactive Web model



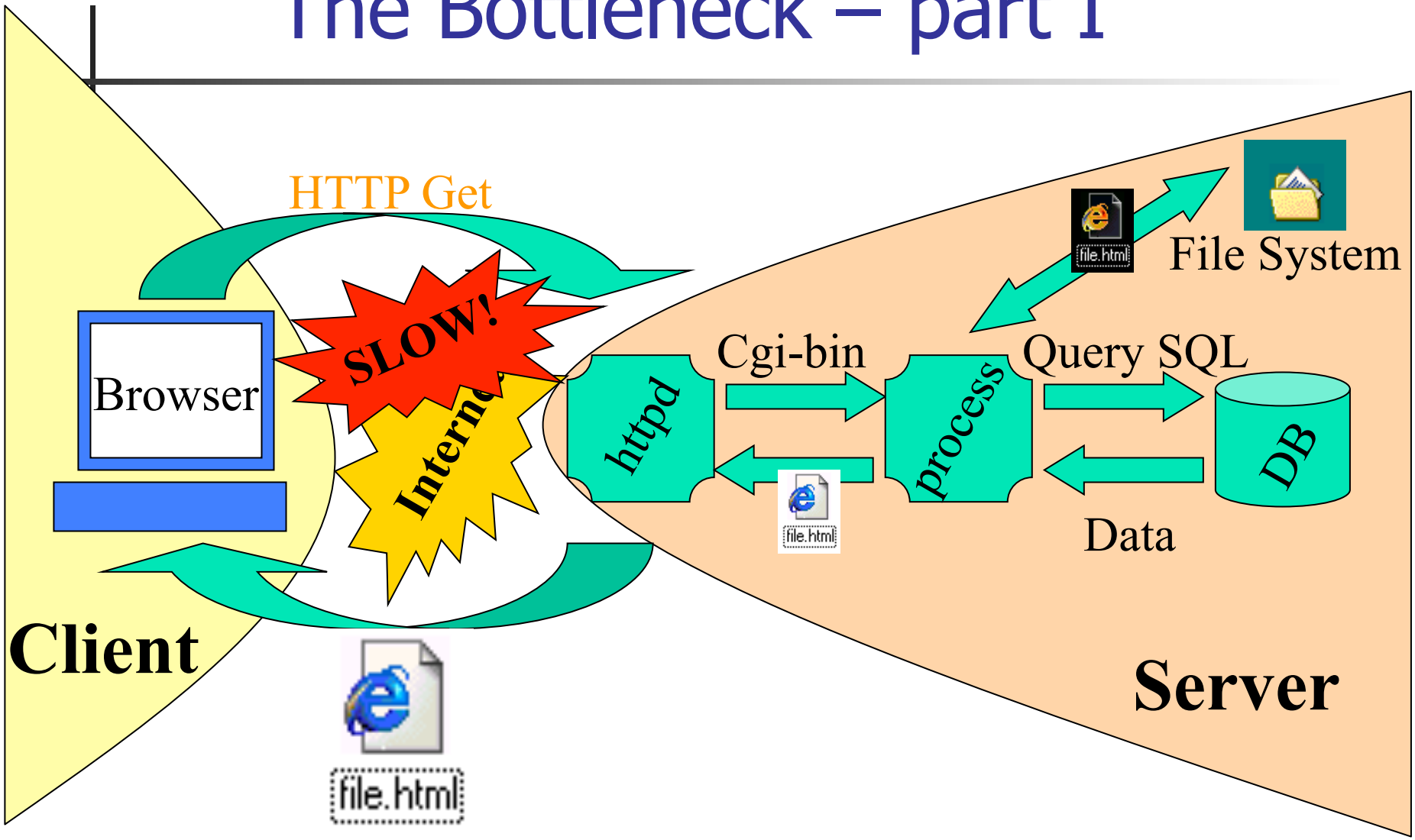
An evolved interactive Web model



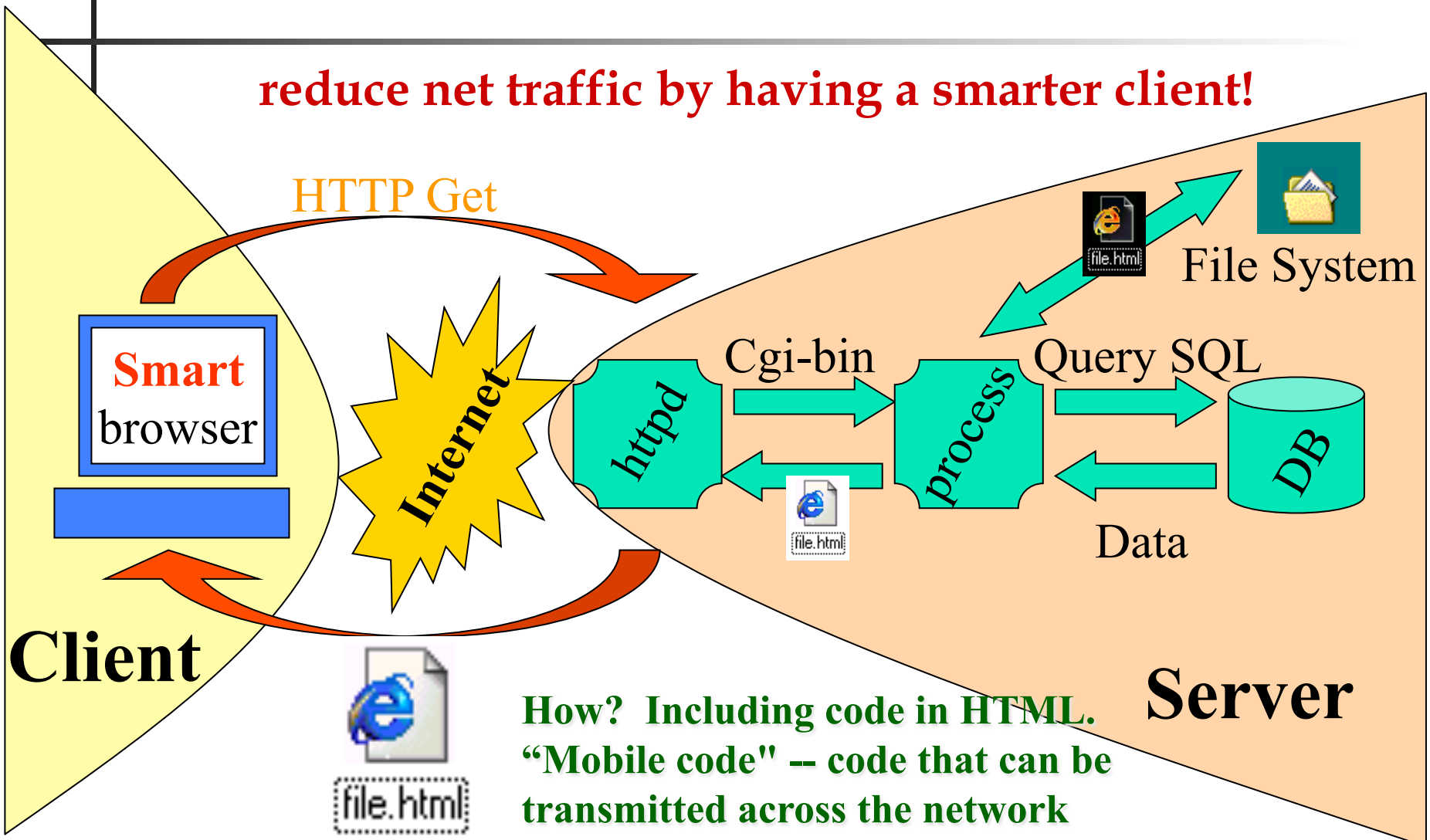
The Bottlenecks



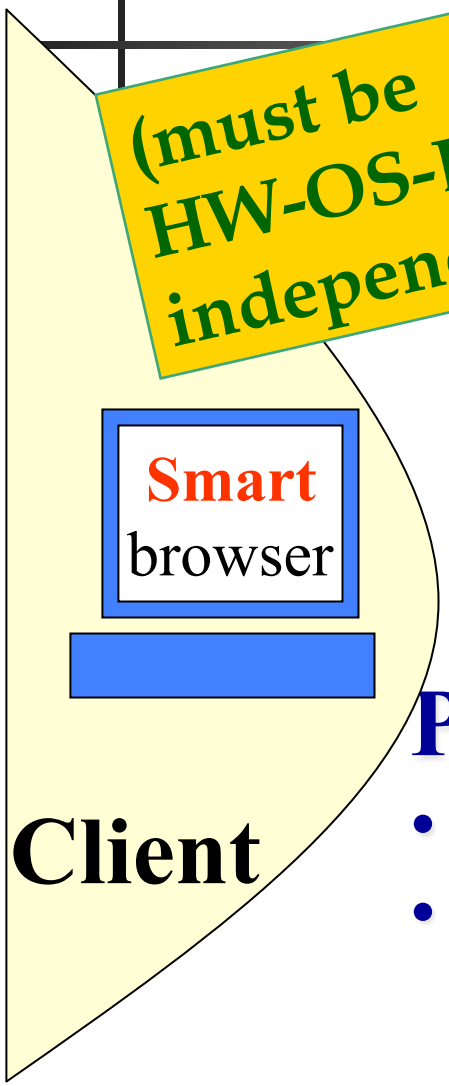
The Bottleneck – part I



A solution:


reduce net traffic by having a smarter client!
HTTP Get
Smart
browser
Internet
Client
file.html
httpd
Cgi-bin
process
Query SQL
DB
Data
file.html
file.html
File System
Server
How? Including code in HTML.
"Mobile code" -- code that can be
transmitted across the network
and executed on the other end

Enabling technologies



Scripting languages

- Javascript
- VBscript
- Perlscript
- Python
- …

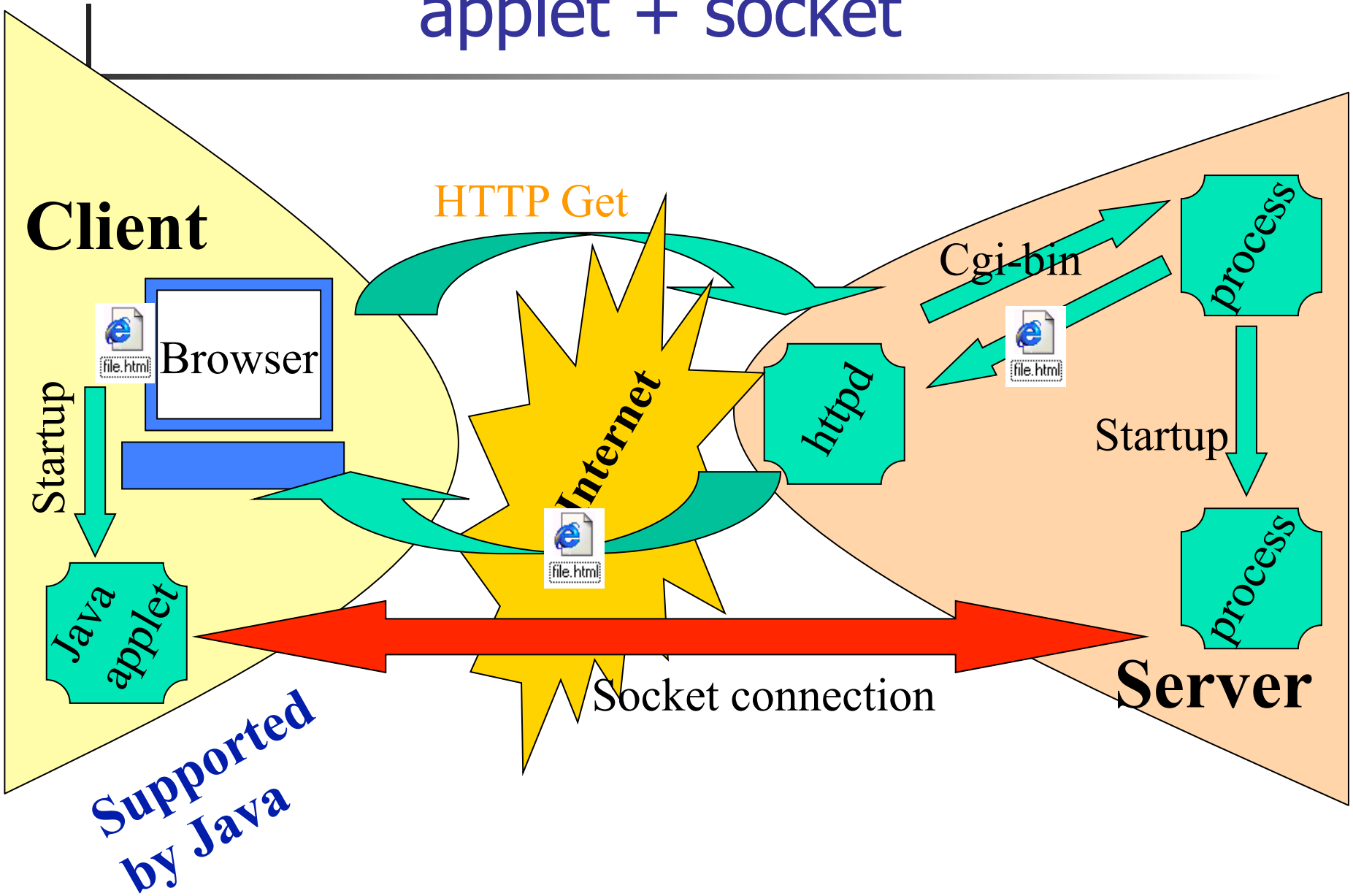
Interpreted:
Source code travels

Programming languages

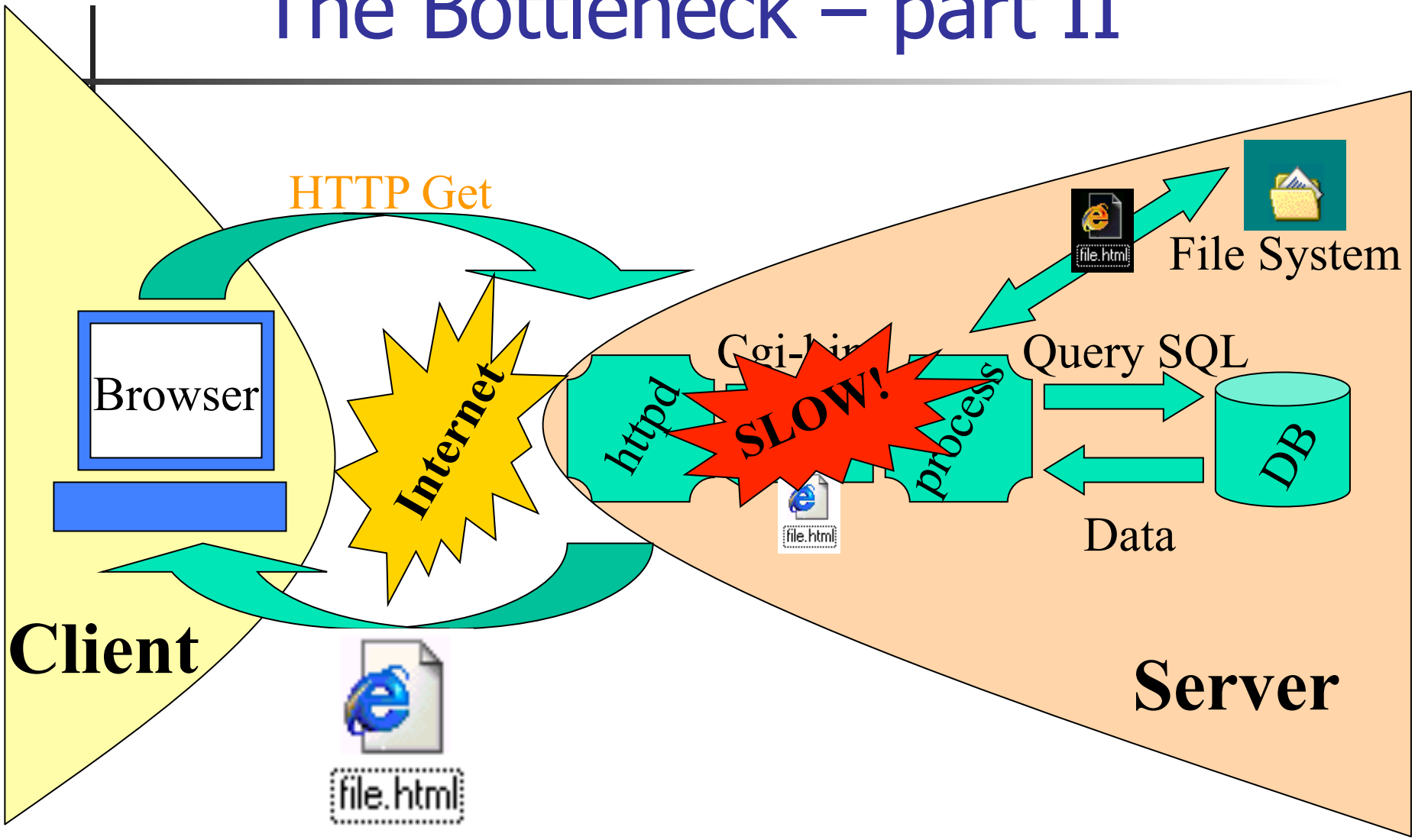
- Java (Applet)
- .NET
  (ActiveX – only Windows)

Compiled:
Executable travel

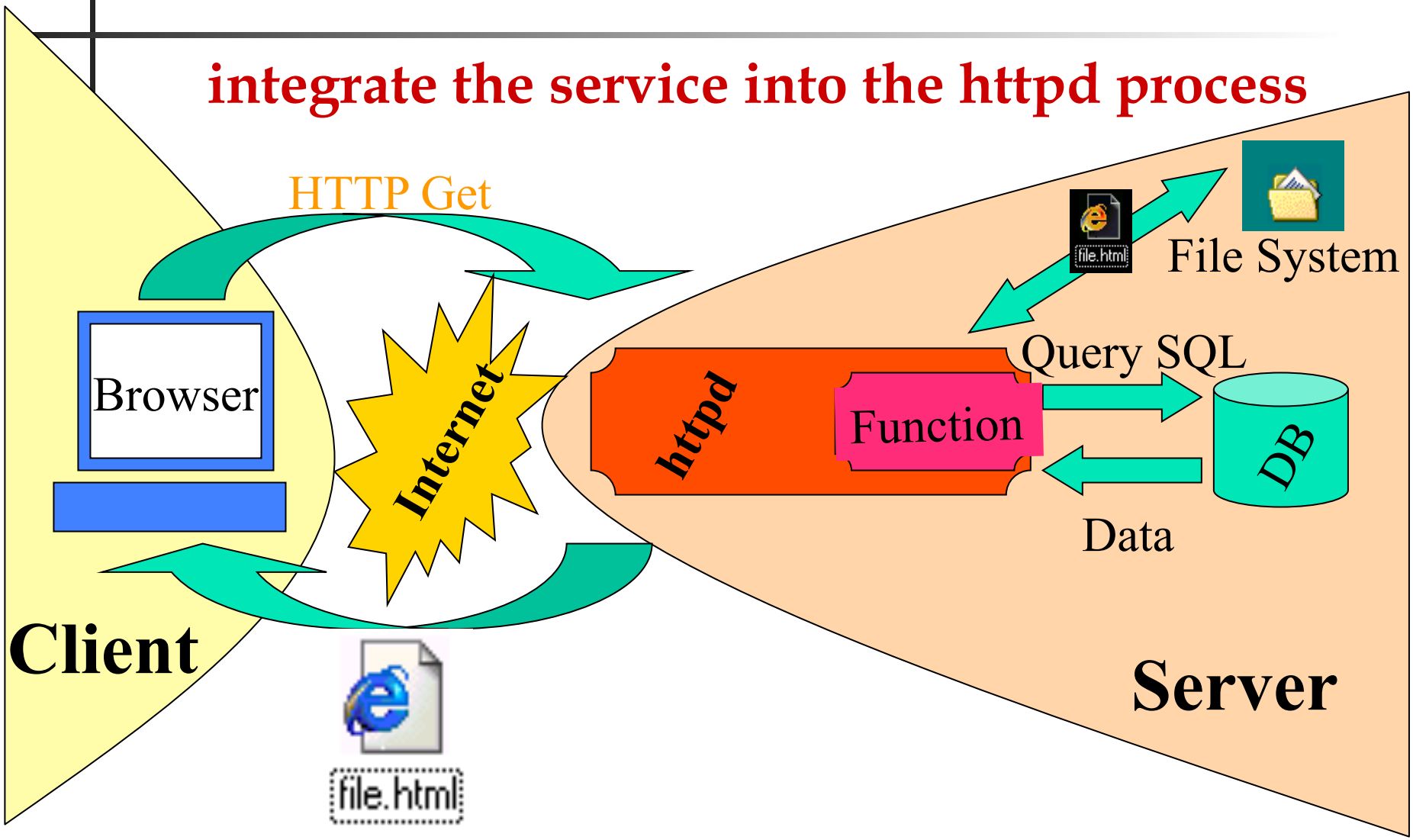
A more radical solution: applet + socket



The Bottleneck – part II



A solution:


integrate the service into the httpd process
HTTP Get
Browser
Internet
Client
file.html
httpd
Function
Query SQL
DB
Data
file.html
File System
Server

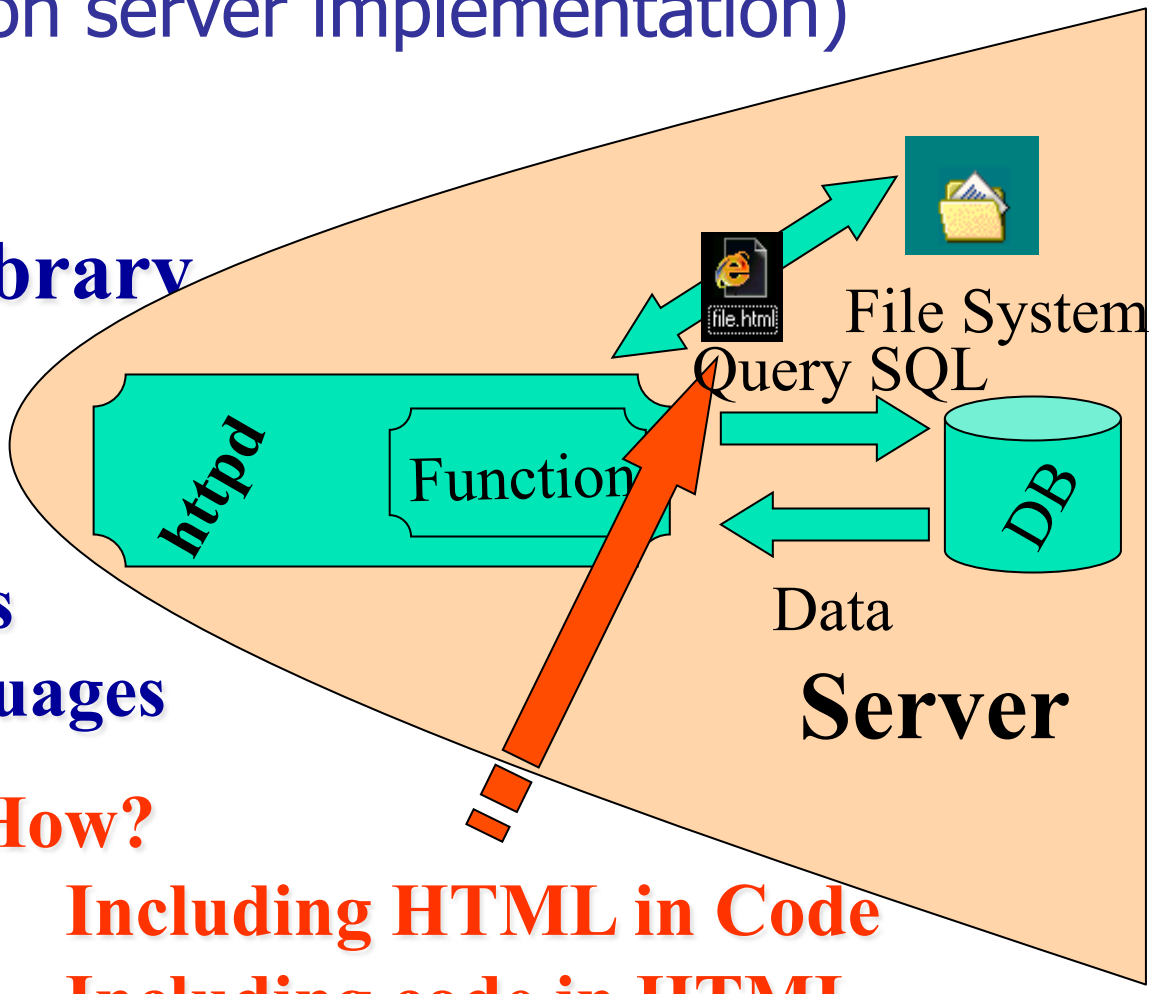
Enabling technologies

(depending on server implementation)

- **Multithreading**
- **DynamicLinkLibrary**
- **Servlets/JSP**

Using…

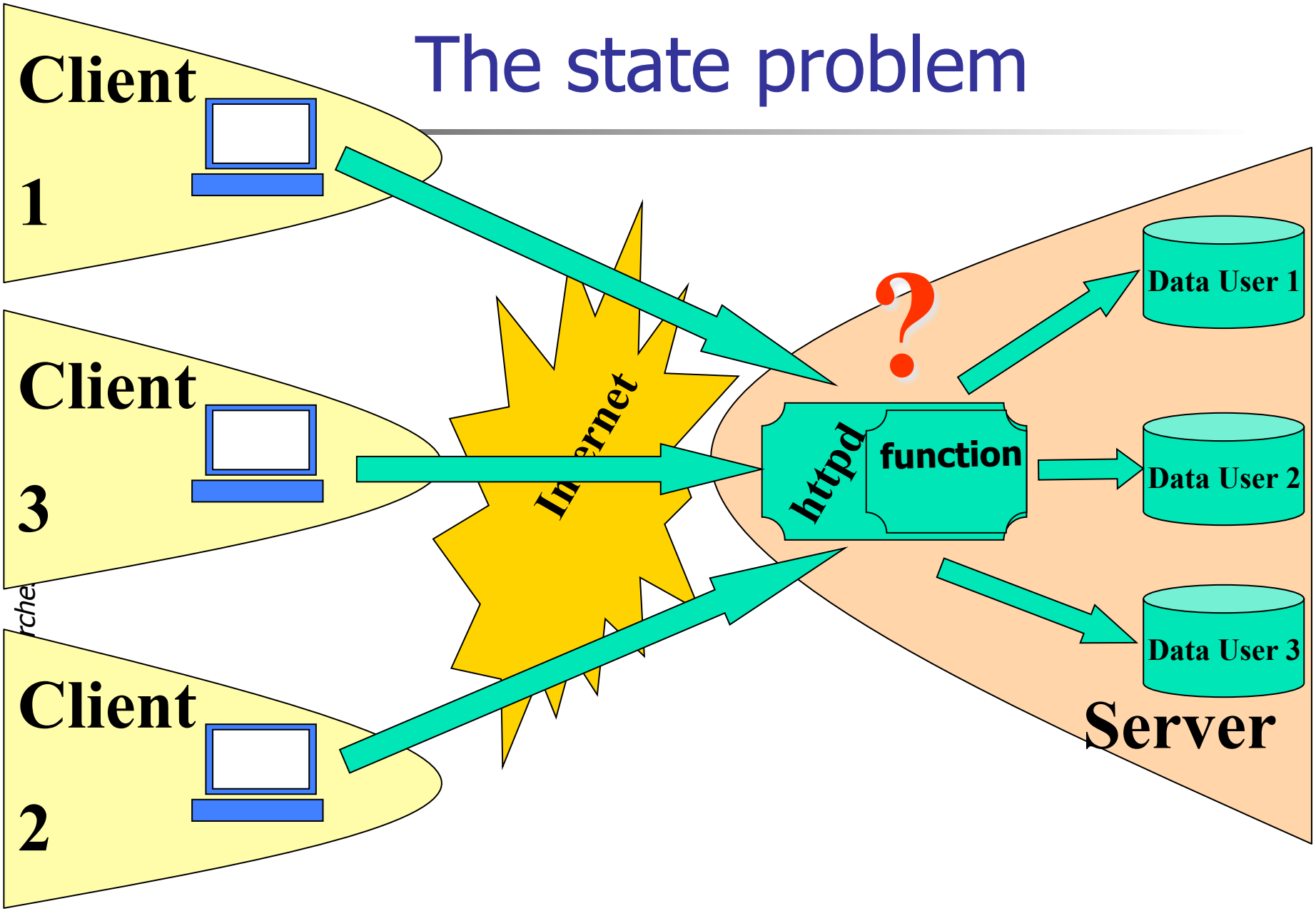
- **Scripting languages**
- **Programming languages**



How?

- **Including HTML in Code**
- **Including code in HTML**

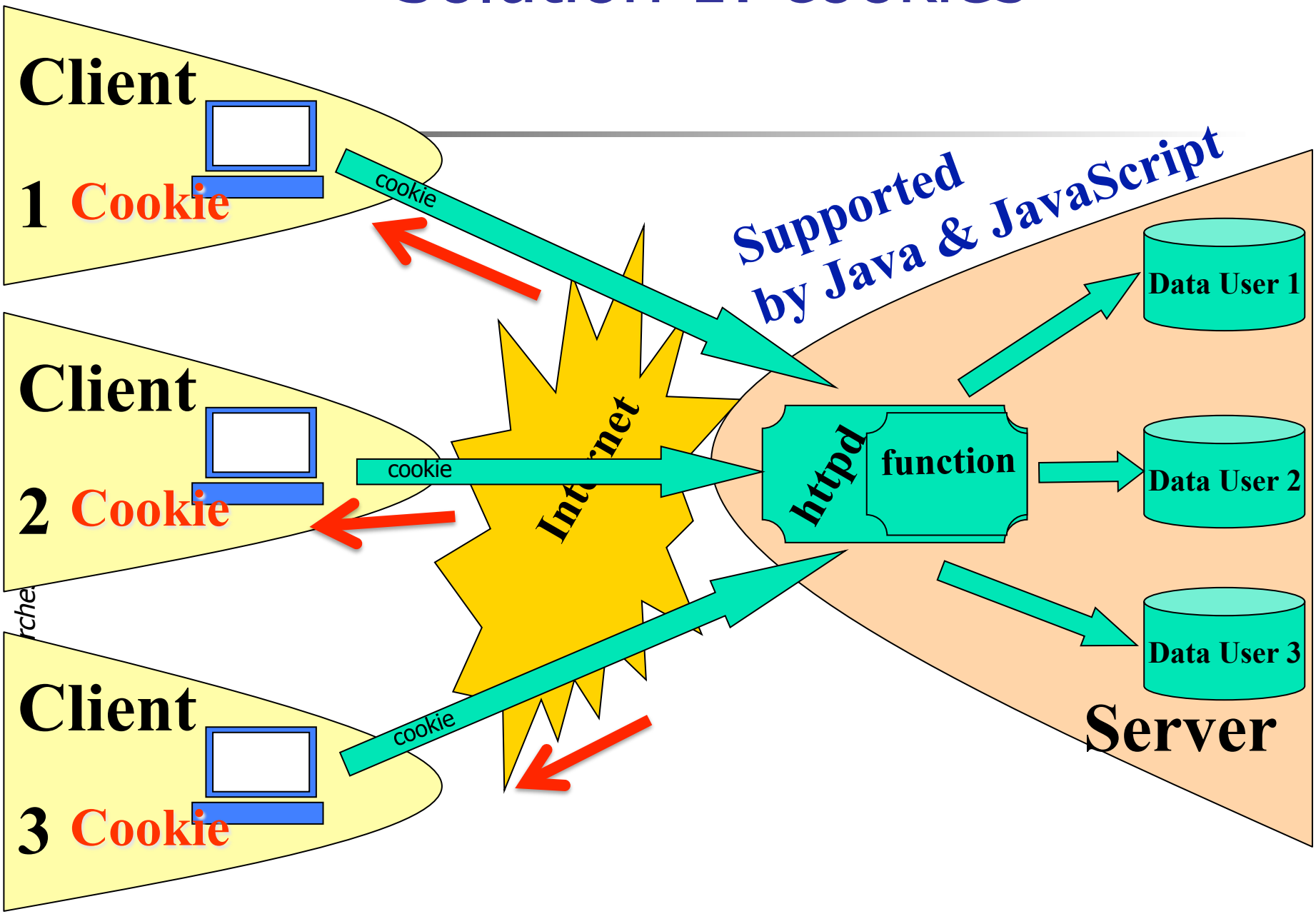
The state problem



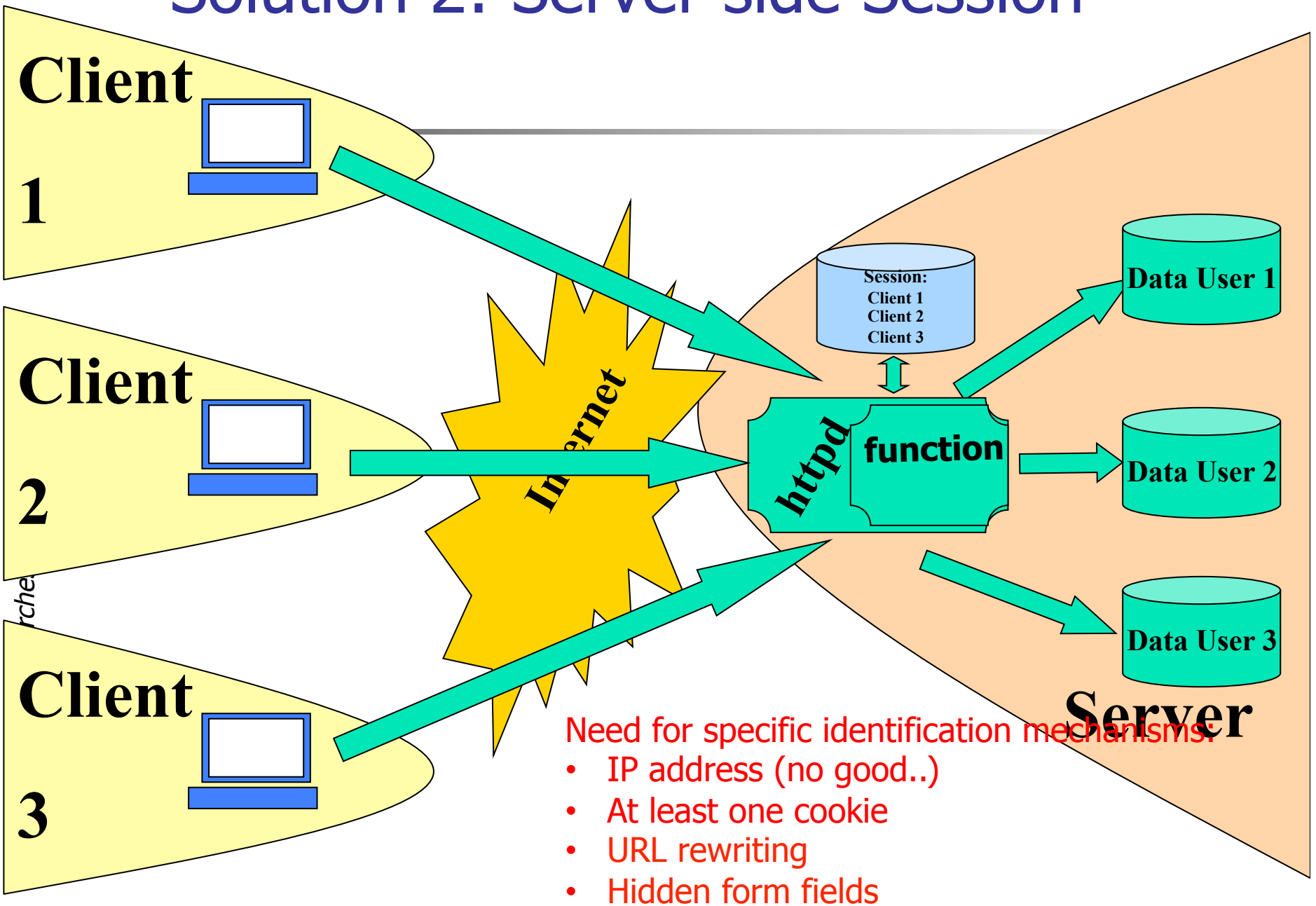
The state problem

- HTTP is a ***stateless*** protocol. This means that if a user inputs some data on a Web page and then goes to another page, the second page does not "know" what has been done on the first.
- ***Session tracking*** allows the server application to remember the user's input and carry it from page to page.
- A ***session*** is some logical task that a user tries to accomplish on a Web site.
- Example: shopping cart application
  - the process of buying a book may involve several steps—book selection, input of billing and shipping information, and so on.
  - Multiple users connect to the same "function", but each of them has a personal shopping cart.
- Session information can be stored either in the client, in the server tier or in an appropriate middle-tier (middle-ware)
  - **Client** → cookies
  - **Server** → send applet / server db
  - **Middleware** → third party service

Solution 1: cookies

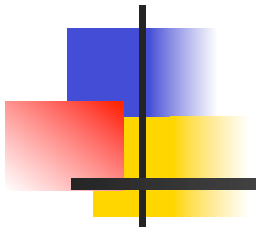


Solution 2: Server side Session



Need for specific identification mechanisms:

- IP address (no good..)
- At least one cookie
- URL rewriting
- Hidden form fields



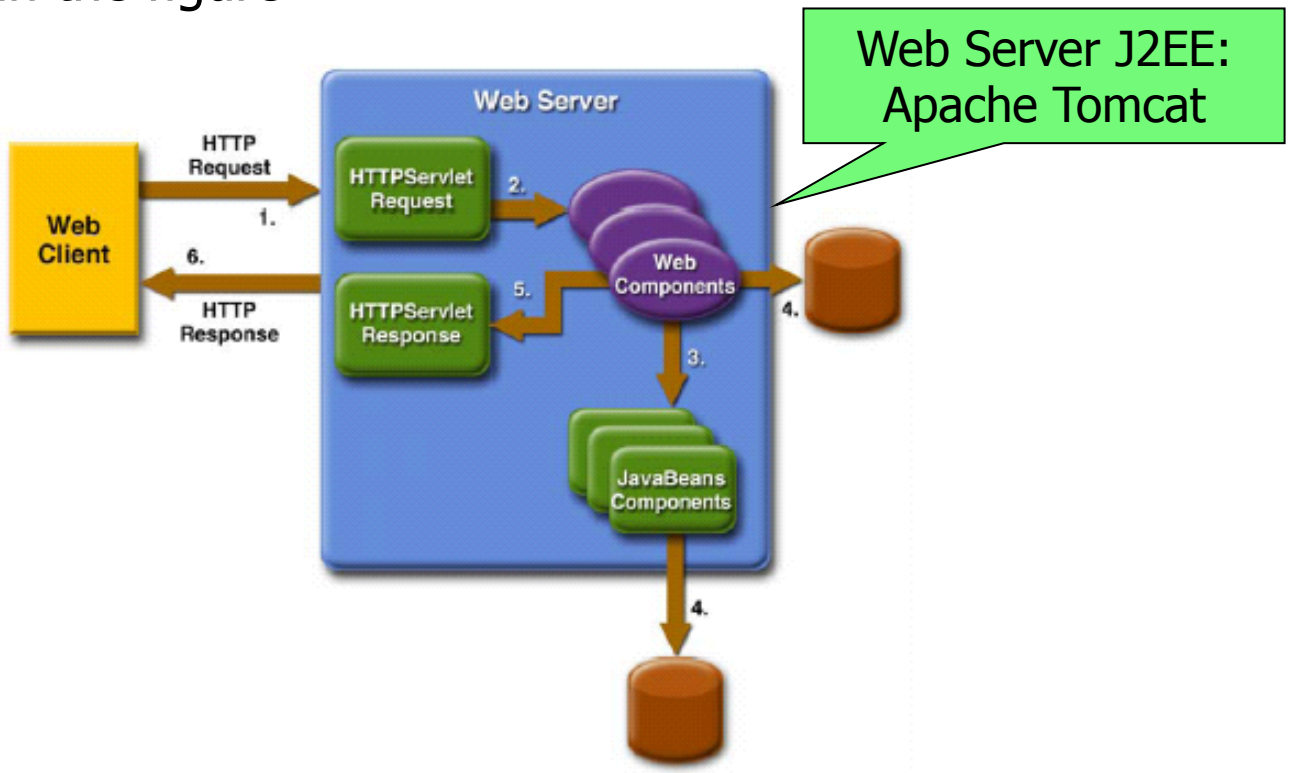
Java Web Applications

Java Web Application

- A Java Web application is a dynamic extension of a Java Web or application server.
- There are two types of Web applications:
  - *Presentation-oriented* : A presentation-oriented Web application generates interactive Web pages containing various types of markup language (HTML, XML, and so on) and dynamic content in response to requests.
  - *Service-oriented* : A service-oriented Web application implements the endpoint of a Web service. Presentation-oriented applications are often clients of service-oriented Web applications.

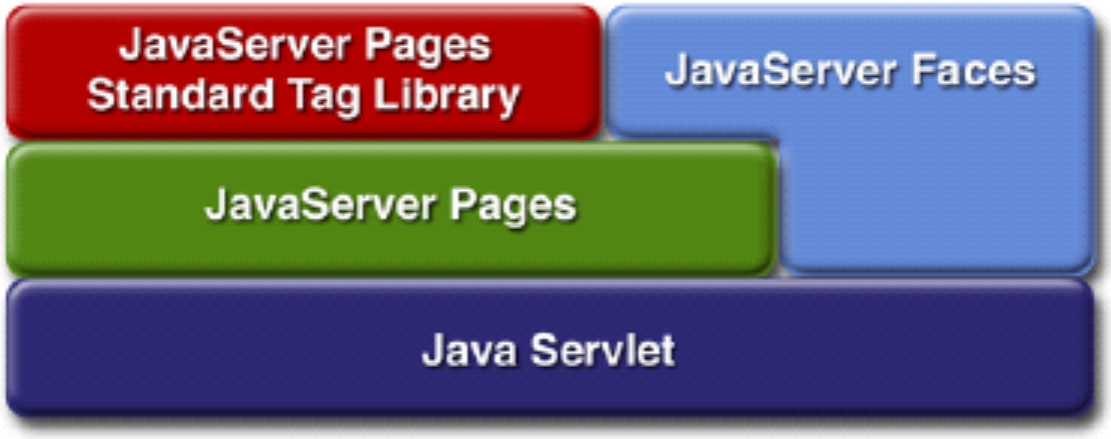
Web components

- In the Java 2 platform, *Web components* provide the dynamic extension capabilities for a Web server. Web components are either Java servlets, JSP pages, or Web service endpoints. The interaction between a Web client and a Web application is illustrated in the figure



Java Web Application Technologies

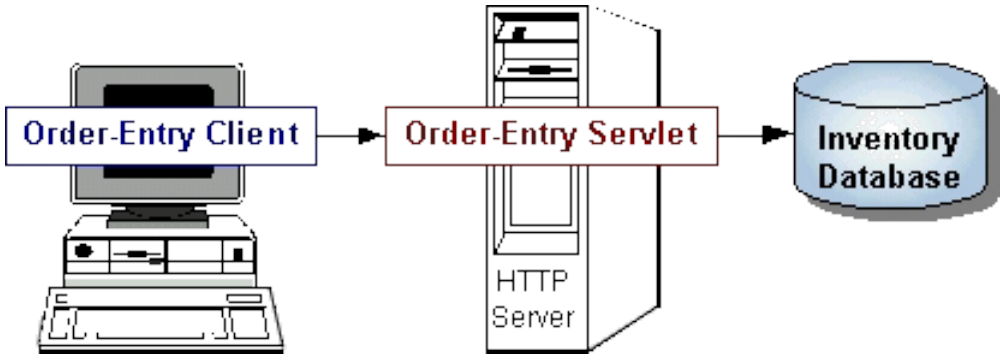
- Since the introduction of Java Servlet and JSP technology, additional Java technologies and frameworks for building interactive Web applications have been developed.



NOTE: all are based on Java Servlet

Servlets

- Servlets are Java Classes that extend Java-enabled web servers.
  - For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.



- Servlets have no graphical user interface.
- For a full tutorial, see http://java.sun.com/docs/books/tutorial/servlets/overview/index.html

Servlet properties

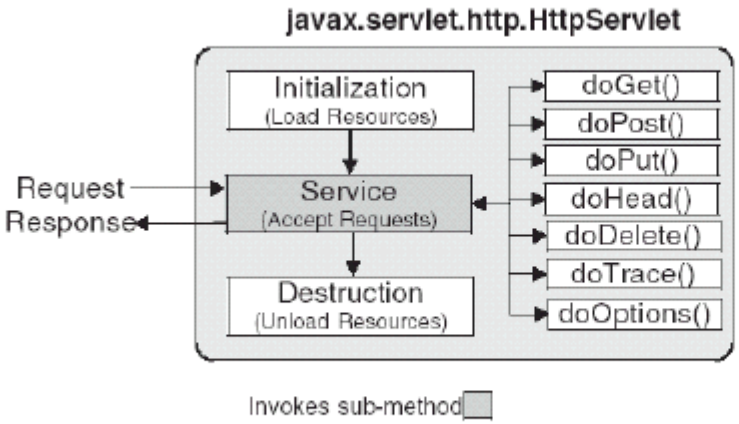
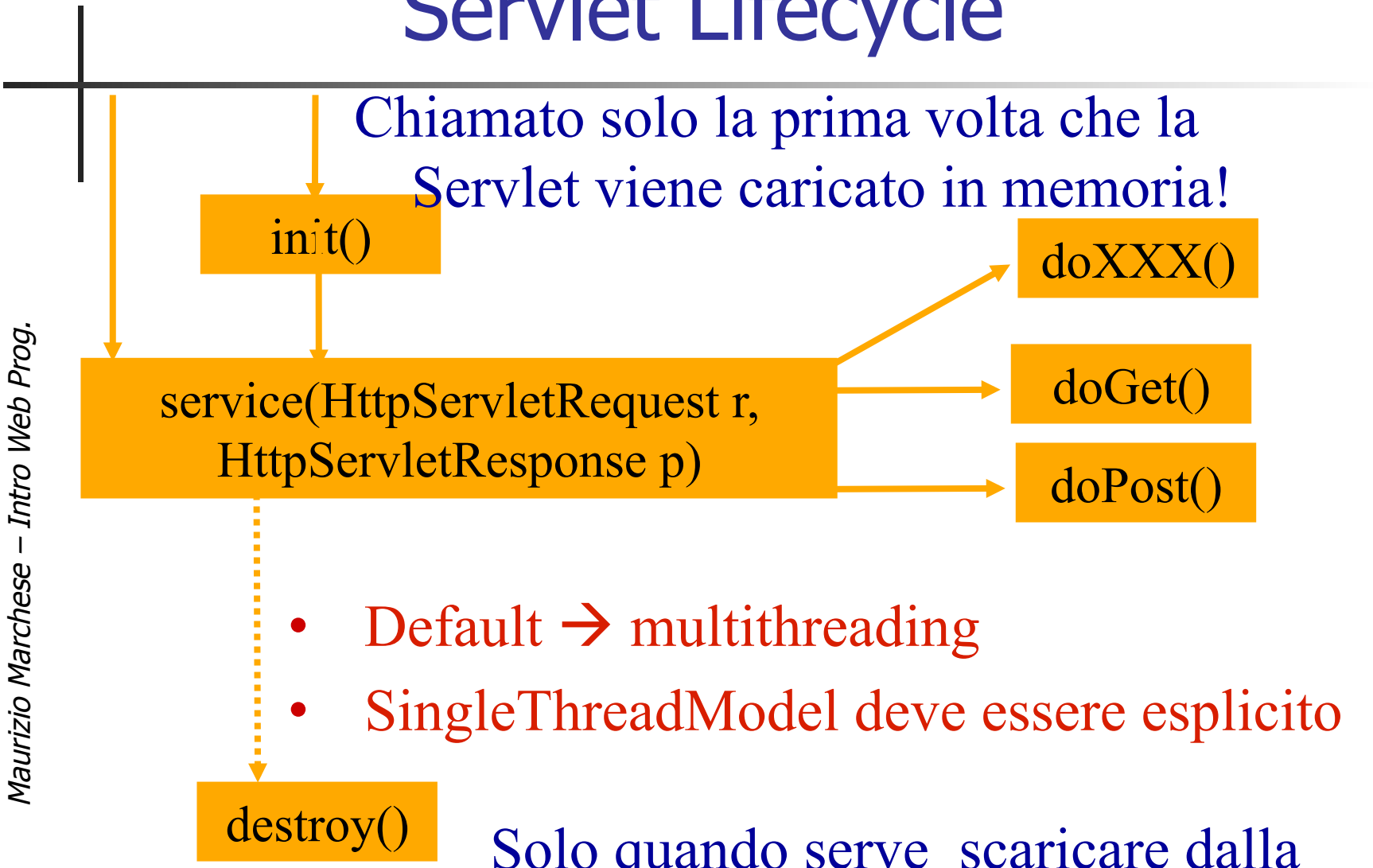


Figure 2-5 HttpServlet Life Cycle

- Extend javax.servlet.http.HttpServlet
- Server side
- Without main
- Without GUI
- Main method: Service();

Servlet Lifecycle



- Default → multithreading
- SingleThreadModel deve essere esplicito

HttpServletRequest Class

```
1  package javax.servlet.http;
2
3  import java.io.IOException;
4  import java.security.Principal;
5  import java.util.Collection;
6  import java.util.Enumeration;
7  import javax.servlet.ServletException;
8  import javax.servlet.ServletRequest;
9
10 public interface HttpServletRequest extends ServletRequest {
11
12     public static final String BASIC_AUTH = "BASIC";
13     public static final String FORM_AUTH = "FORM";
14     public static final String CLIENT_CERT_AUTH = "CLIENT_CERT";
15     public static final String DIGEST_AUTH = "DIGEST";
16
17     public String getAuthType();
18
19     public Cookie[] getCookies();
20
21     public long getDateHeader(String name);
22
23     public String getHeader(String name);
24
25     public Enumeration<String> getHeaders(String name);
26
27     public Enumeration<String> getHeaderNames();
28
29     public int getIntHeader(String name);
30
31     public String getMethod();
32
33     public String getPathInfo();
34
35     public String getPathTranslated();
36
37     public String getContextPath();
38
39     public String getQueryString();
40
41     public String getRemoteUser();
42
43     public boolean isUserInRole(String role);
```

Basic Servlet Structure 1

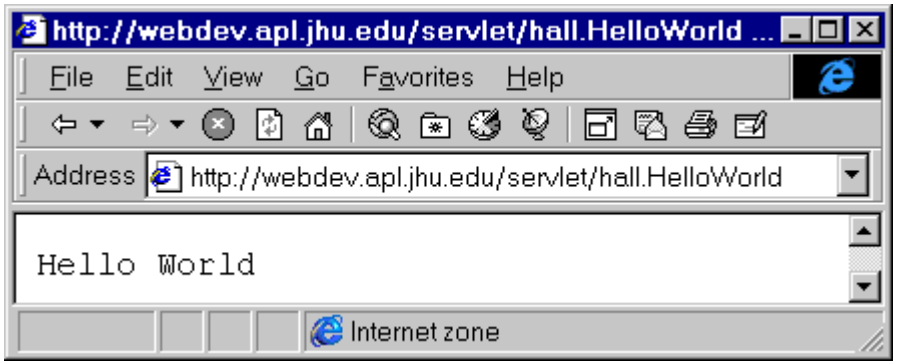
- outline of a basic servlet that handles GET requests
  - GET requests are made by browsers when the user types in a URL on the address line, follows a link from a Web page, or makes an HTML form that does not specify a METHOD.
  - Servlets can also very easily handle POST requests, which are generated when someone creates an HTML form that specifies METHOD="POST" (later)

Basic Servlet Structure 2

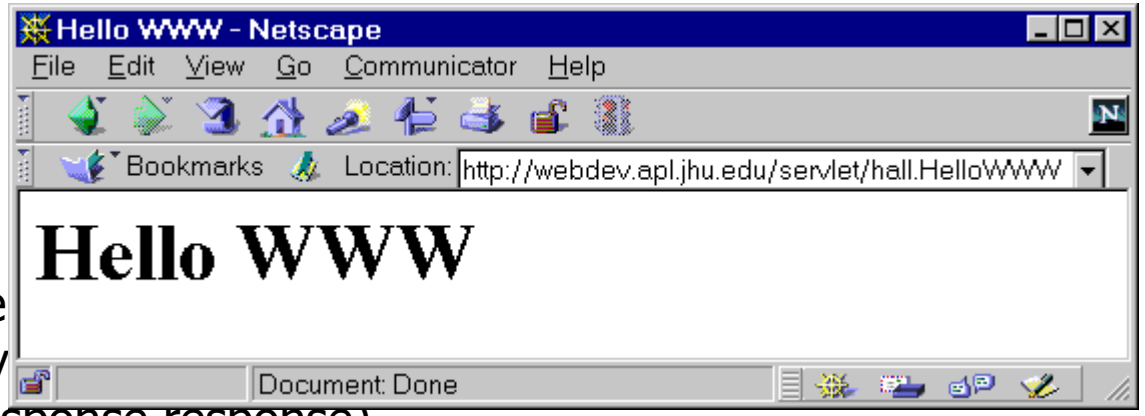
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SomeServlet extends HttpServlet {

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

    // Use "request" to read incoming HTTP headers (e.g. cookies)
    // and HTML form data (e.g. data the user entered and submitted)

    // Use "response" to specify the HTTP response line and headers
    // (e.g. specifying the content type, setting cookies).

    PrintWriter out = response.getWriter();
    // Use "out" to send content to browser
  } }
```

HelloWorld.java

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

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();

    out.println("Hello World");
  }
}
```



HelloWWW.java

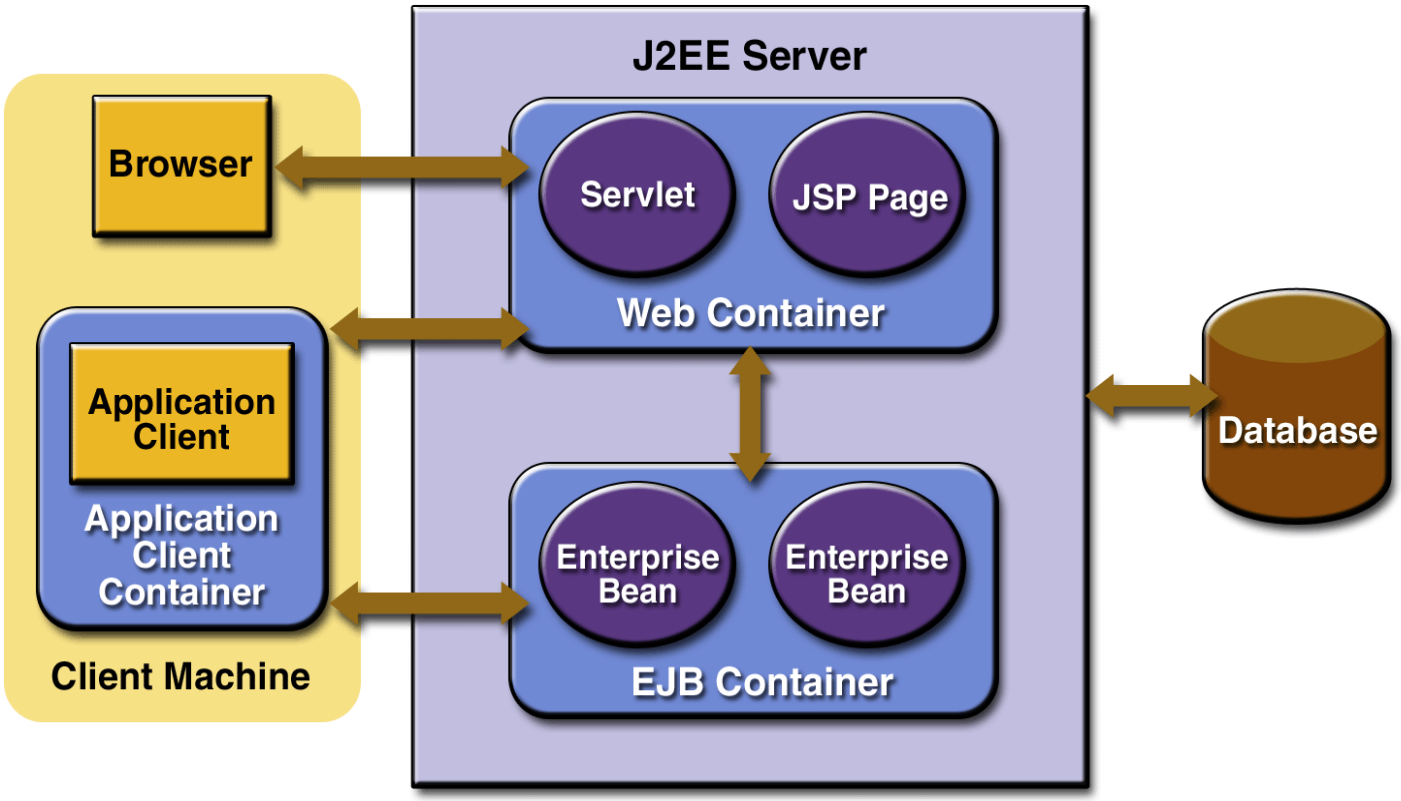



```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWWW exte
  public void doGet(HttpServ
                    HttpServletResponse response)
      throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
                                "Transitional//EN\">\n" +
            "<HTML>\n" +
            "<HEAD><TITLE>Hello WWW</TITLE></HEAD>\n" +
            "<BODY>\n" +
            "<H1>Hello WWW</H1>\n" +
            "</BODY></HTML>"); } }
```

Other uses of servlets

- Forwarding requests.
  - Servlets can forward requests to other servers and servlets. Thus servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.
- Allowing collaboration between people.
  - A servlet can handle multiple requests concurrently, and can synchronize requests.
    This allows servlets to support systems such as on-line conferencing.

Dynamic model: Java Servlet & JSP



Web application deployment descriptor

- Web components (such as Servlet) are supported by the services of a runtime platform called a *Web container*.
- A Web container provides services such as request dispatching, security, concurrency, and life-cycle management. It also gives Web components access to APIs such as naming, transactions, and email.
- Certain aspects of Web application behavior can be configured when the application is installed, or *deployed*, to the Web container.
  The configuration information is maintained in a text file in XML format called a *Web application deployment descriptor* (DD). A DD must conform to the Java Servlet Specification XML schema.

Web Application Life Cycle

- A Web application consists of Web components, static resource files such as images, and helper classes and libraries. The Web container provides many supporting services that enhance the capabilities of Web components and make them easier to develop.

- However, because a Web application must take these services into account, the process for creating and running a Web application is different from that of traditional stand-alone Java classes.

Web Application Life Cycle

- The process for creating, deploying, and executing a Web application can be summarized as follows:
  - Develop the Web component code.
  - Develop the Web application deployment descriptor.
  - Compile the Web application components and helper classes referenced by the components.
  - *Optionally* package the application into a deployable unit.
  - Deploy the application into a Web container.
  - Access a URL that references the Web application.

Web Modules

- In the J2EE architecture, Web components and static Web content files such as html files and images are called *Web resources*.
- A *Web module* is the smallest deployable and usable unit of Web resources.
- A *J2EE Web module* corresponds to a *Web application*
- In addition to Web components and Web resources, a Web module can contain other files:
  - Server-side utility classes (database beans, shopping carts, and so on). Often these classes conform to the JavaBeans component architecture.
  - Client-side classes (applets and utility classes).

Web Module Structure

- **A Web module has a specific structure**
- The top-level directory of a Web module is the *document root* of the application. The document root is where JSP pages, *client-side* classes and archives, and static Web resources, such as images, are stored.
- The document root contains a subdirectory named /WEB-INF/, which contains the following files and directories:
  - web.xml: The Web application deployment descriptor
  - Tag library descriptor files
  - classes: A directory that contains *server-side classes*: servlets, utility classes, and JavaBeans components
  - tags: A directory that contains tag files, which are implementations of tag libraries
  - lib: A directory that contains JAR archives of libraries called by serverside classes
  - You can also create application-specific subdirectories (that is, package directories) in either the document root or the /WEB-INF/ classes/ directory.

Web Module Structure