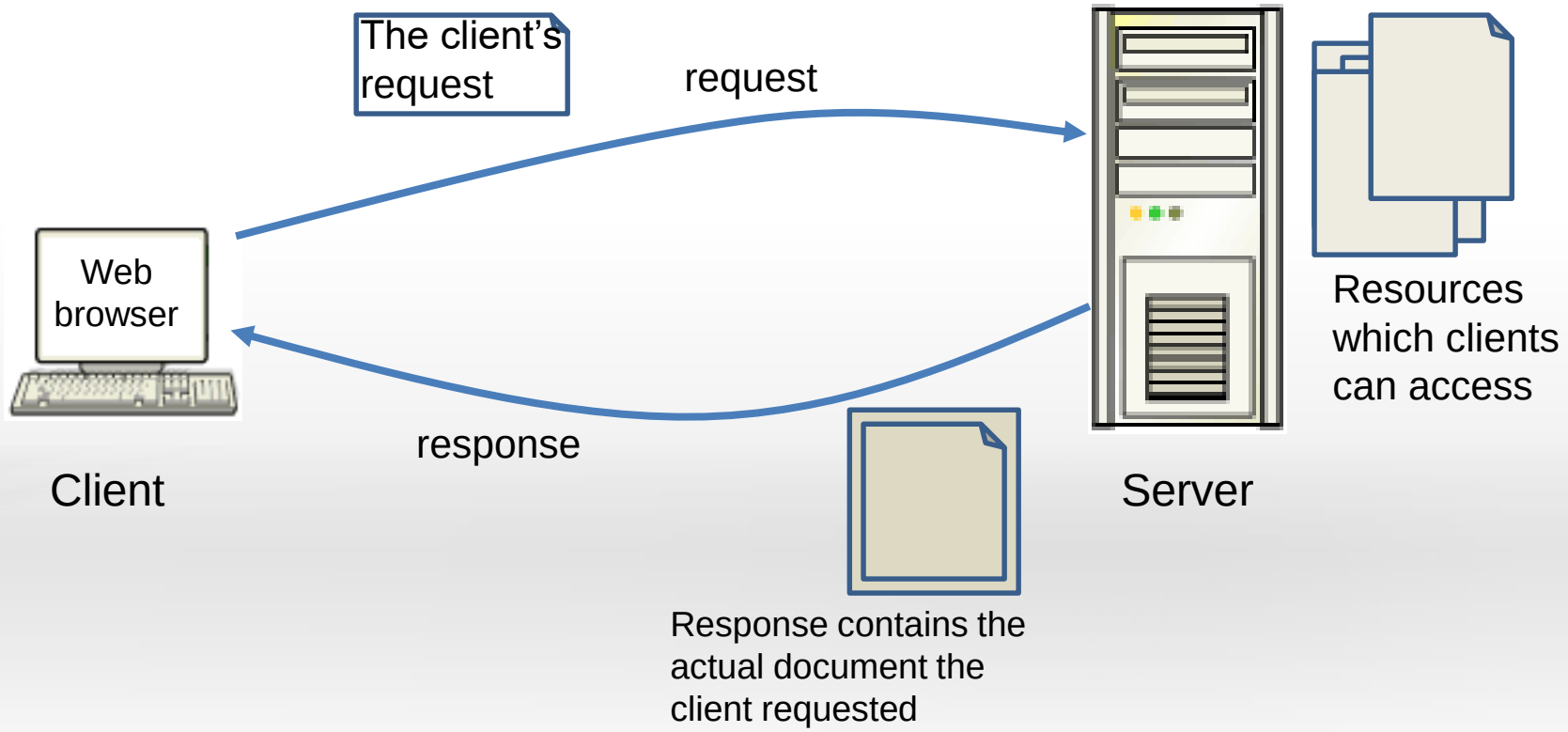
Introduction to JEE

Banu Prakash

Web Server

- A web server takes a client request and sends back the response to the client.



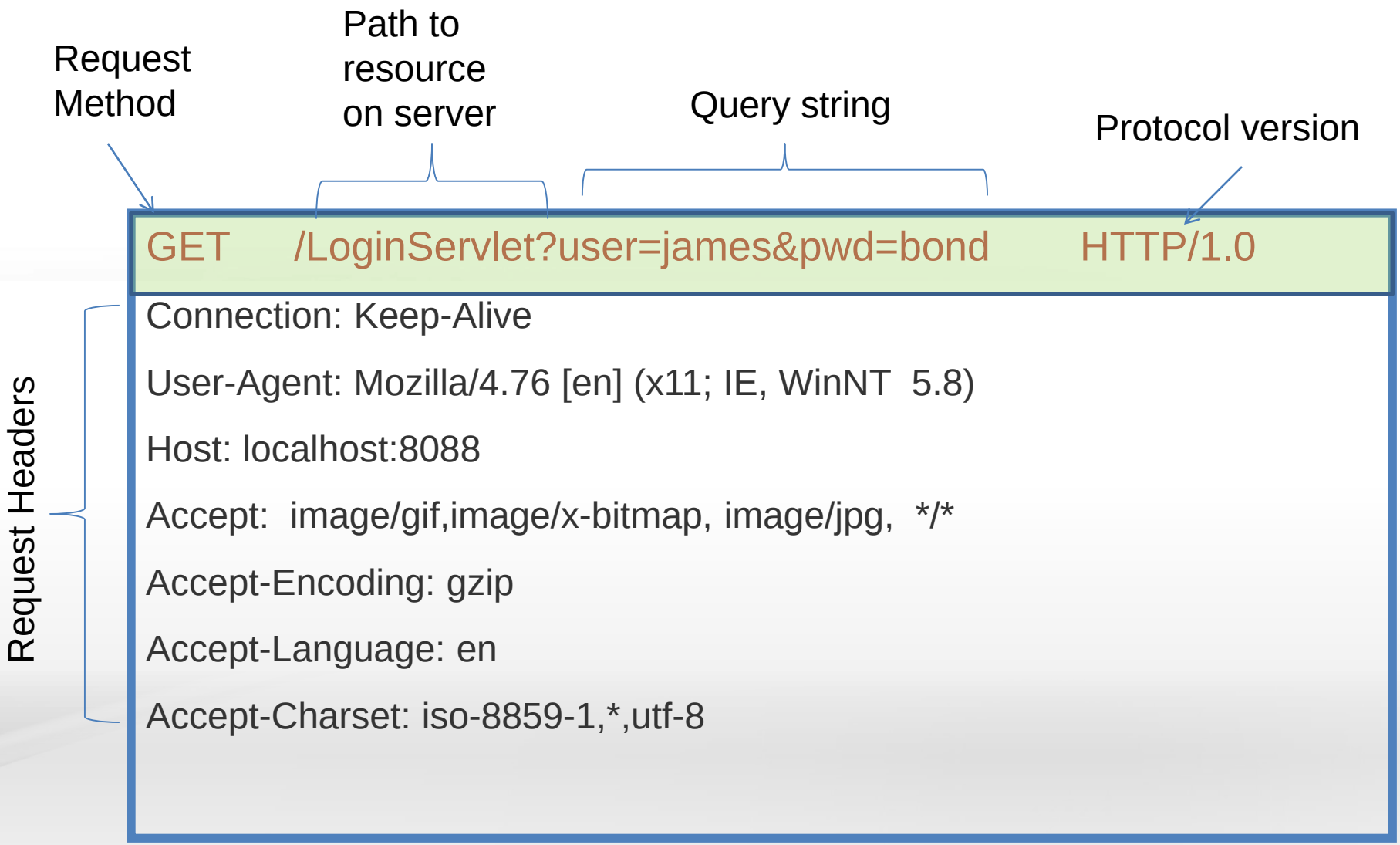
HTTP

- HTTP functions as a request-response protocol in the client-server computing model.
- In HTTP, a web browser, for example, acts as a *client*, while an application running on a computer hosting a web site functions as a *server*.
- The client submits an HTTP *request* message to the server. The server, which has *resources* such as HTML files, or performs other functions on behalf of the client, returns a response message to the client.

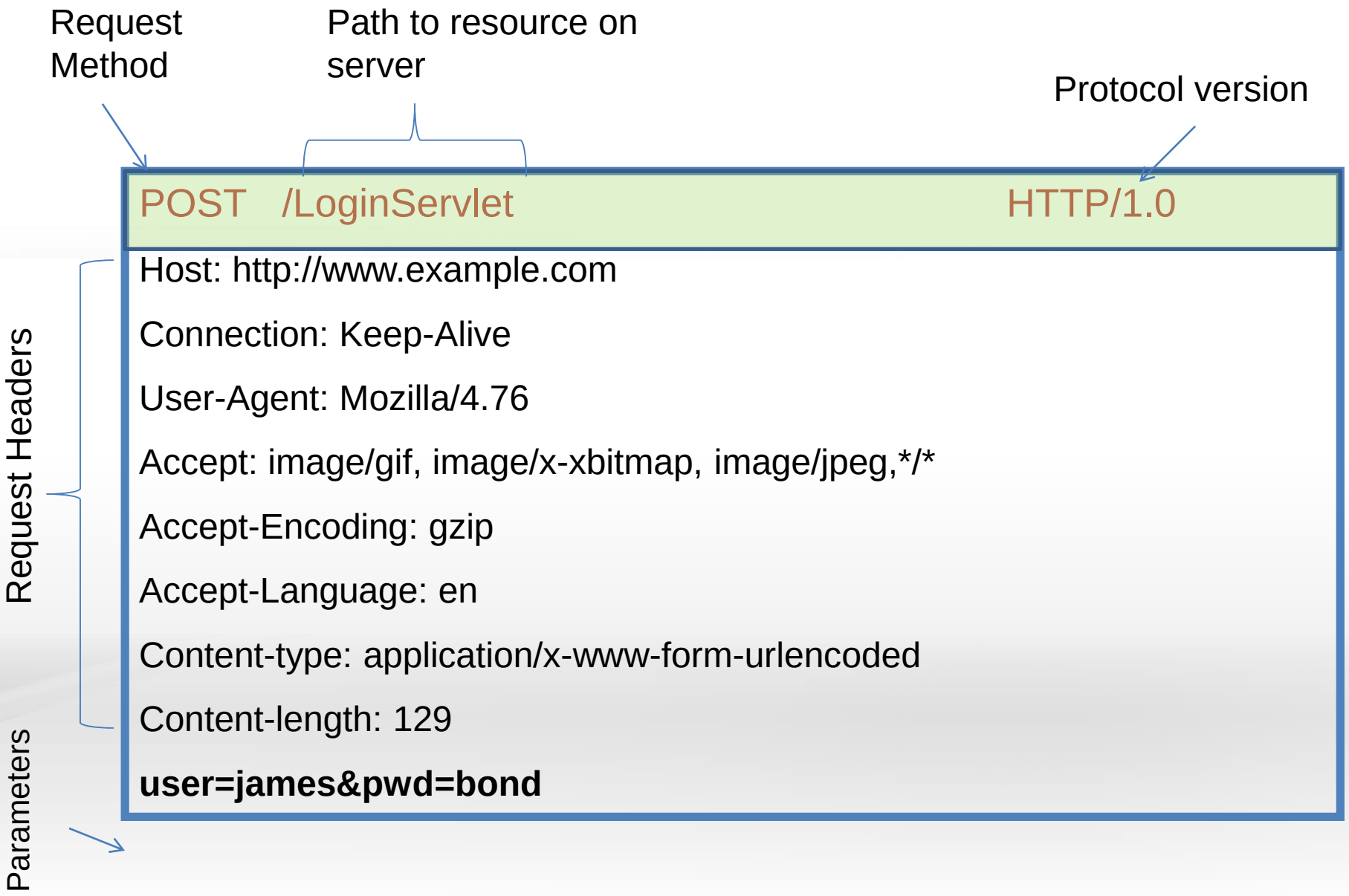
HTTP Request Methods

- **GET**
  - Requests a representation of the specified resource. Requests using GET "SHOULD NOT have the significance of taking an action other than retrieval".
- **POST**
  - Submits data to be processed (HTML form data) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.
- **HEAD**
  - The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response. This method is often used for testing hypertext links for validity, accessibility, and recent modification
- **PUT**
  - The PUT method requests that the enclosed entity be stored under the supplied Request-URI

GET request


Request Method
Path to resource on server
Query string
Protocol version
GET /LoginServlet?user=james&pwd=bond HTTP/1.0
Request Headers
Connection: Keep-Alive
User-Agent: Mozilla/4.76 [en] (x11; IE, WinNT 5.8)
Host: localhost:8088
Accept: image/gif,image/x-bitmap, image/jpg, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8

POST request



HTTP Response

- After receiving and interpreting a request message, a server responds with an HTTP response message.
- A response from a Web server normally consists of a status line, one or more response headers, a blank line, and the document.
- Response =
  - Status-Line ;
  - header
  - CRLF
  - [message-body]

HTTP Response

Answer this

GET or POST ?

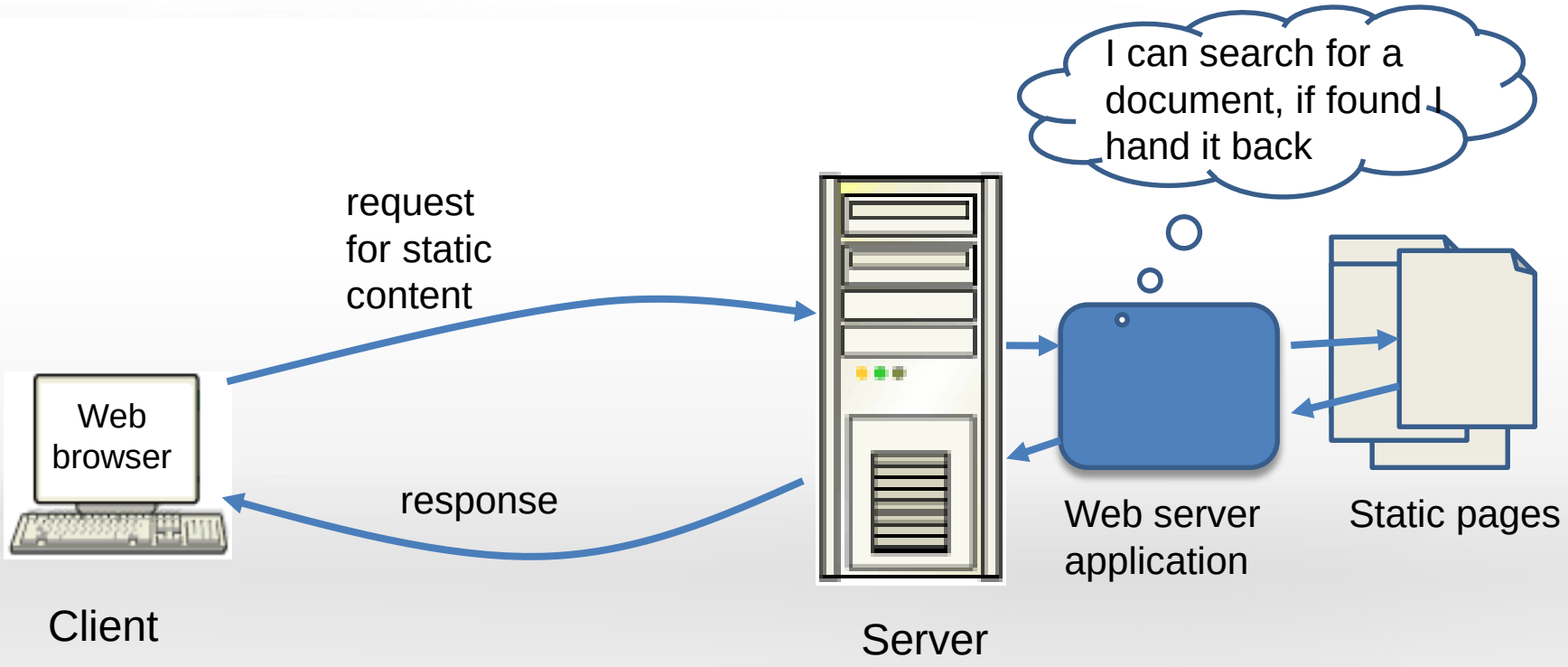
- A user is requesting a new page by clicking on an hyperlink
- A user is uploading his profile photo
- A user is submitting his credentials
- A user is submitting search text to an search engine.

- What is the default port number used by HTTP protocol?

- State True or False
  - GET can be used to pass parameters to server

Static Content

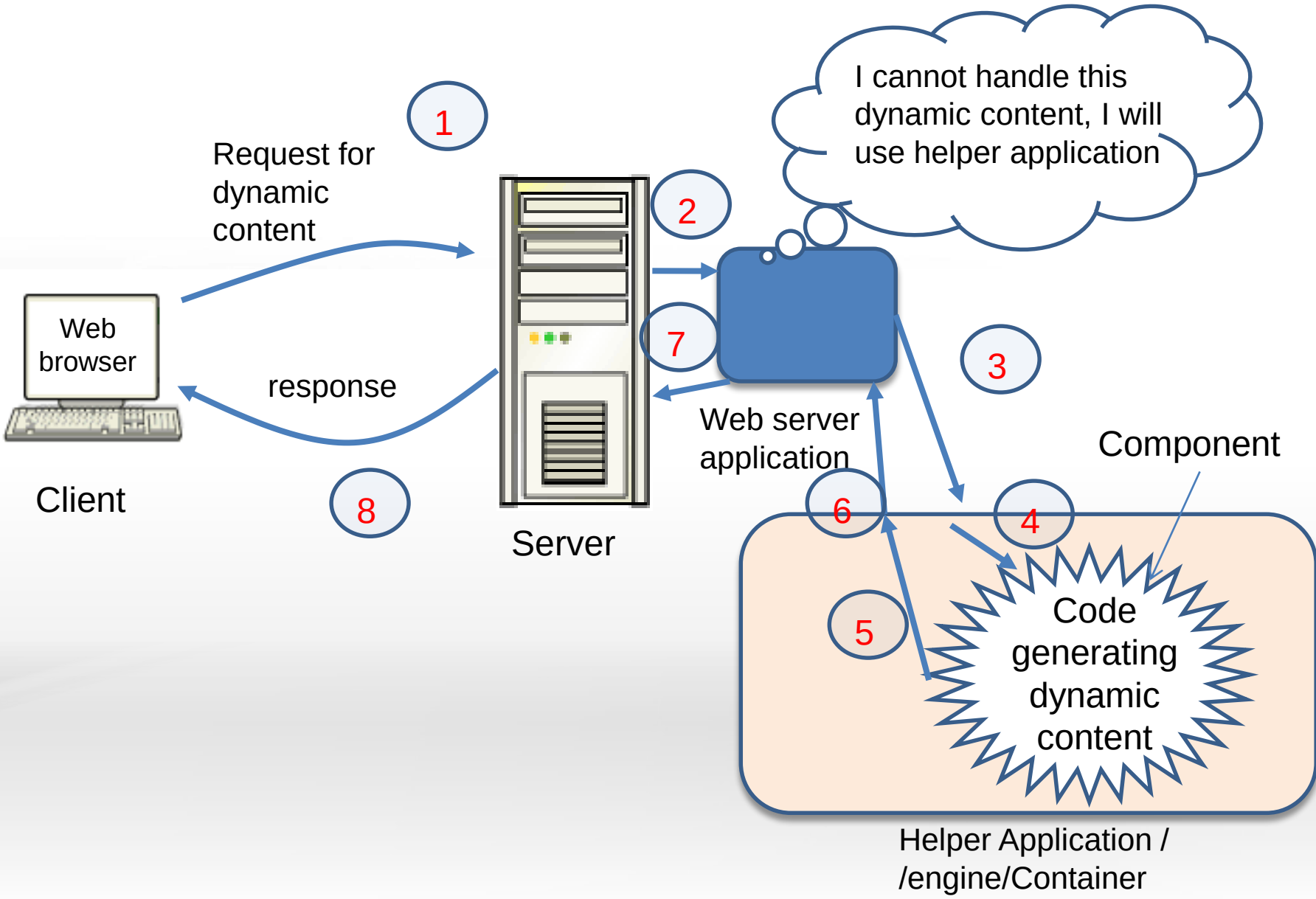
- Web server application by itself can just server static pages
- A static page resides in some folder on a web server machine.
- Every client sees the same content.



Dynamic Content

- To generate dynamic content, we need another helper application on web server commonly known as engine/container.
- If an helper application is ASP engine, the web server can serve dynamic contents using applications written using ASP.
- If an helper application is Servlet engine (Web Container) , then the web server can server dynamic content using Servlet/JSP technology.
- Similarly for CGI, it can serve PERL, C , Python or Unix shell scripts.

Dynamic Content



JEE
Java Enterprise Edition

An introduction to JEE

What is enterprise application?

- Enterprise applications usually involves the safe storage, retrieval and manipulation of business data: customer invoices, flight bookings, and so on.
- Enterprise applications might have multiple user interfaces: a web and mobile interface for customers, and a GUI application in the branch office.
- Enterprise applications have to deal with communication between remote systems.
- If the business grows, the application needs to grow with it.
- One application server crashing or one computer accidentally turned off should not stop the system as a whole from working. Enterprise applications need to be able to handle far more concurrent requests than one server can handle (Clustering).

Java Enterprise Edition (JEE)

- JEE is a platform for building enterprise applications written in Java language.
- JEE provides portable, multiuser, secure and standard platform.

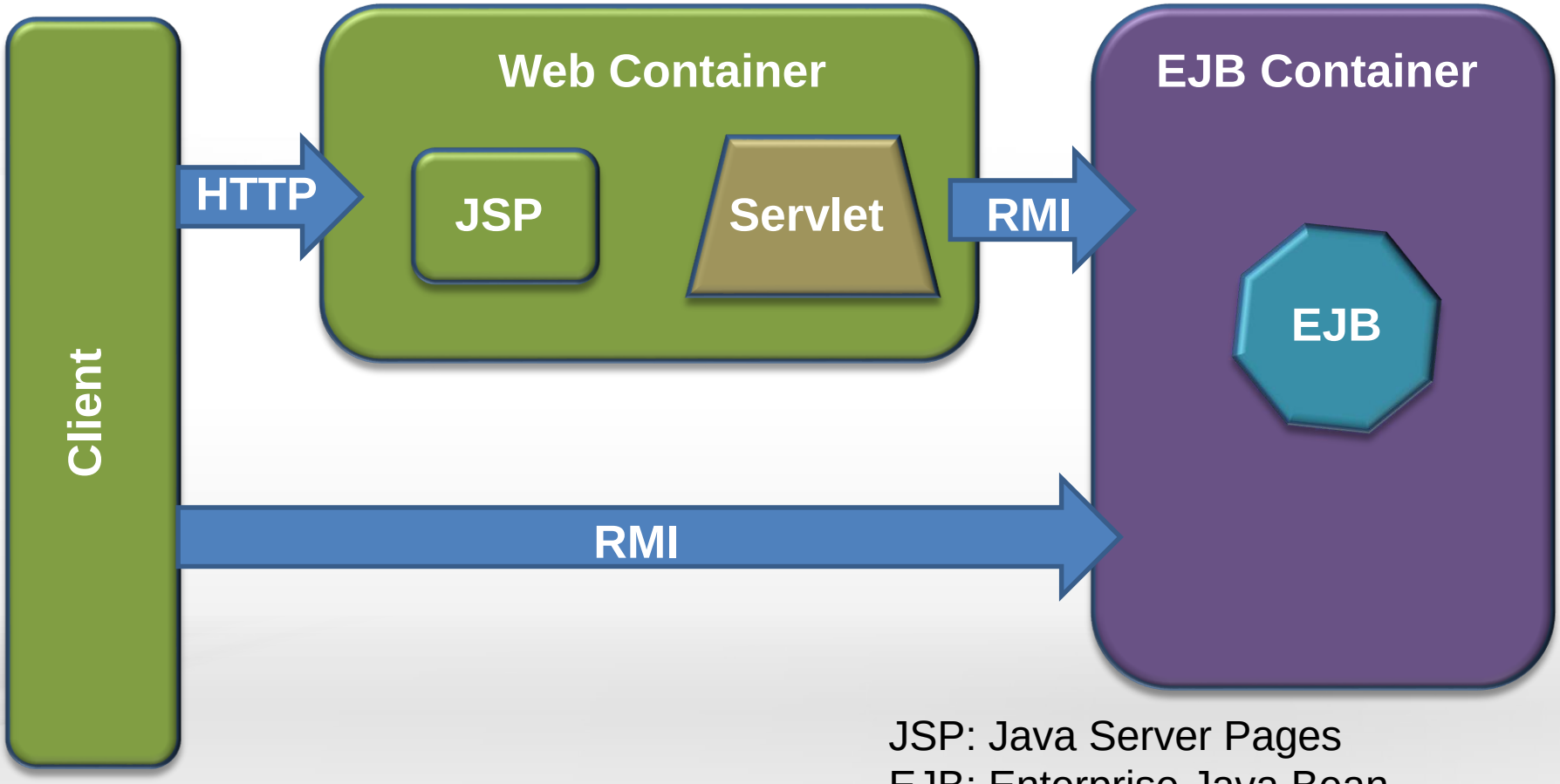
Java Enterprise Edition (JEE)

- JEE technologies:
  - Web Application Technologies
    - Java Servlet
    - Java Server Pages (JSP)
    - Java Server Faces (JSF)
  - Enterprise Application Technologies
    - Enterprise Java Beans (EJB)
    - Java Persistence API
    - Java Mail
    - Connector Architecture
    - Java Message Service API (JMS)
    - Java Transaction API (JTA)

Java Enterprise Edition

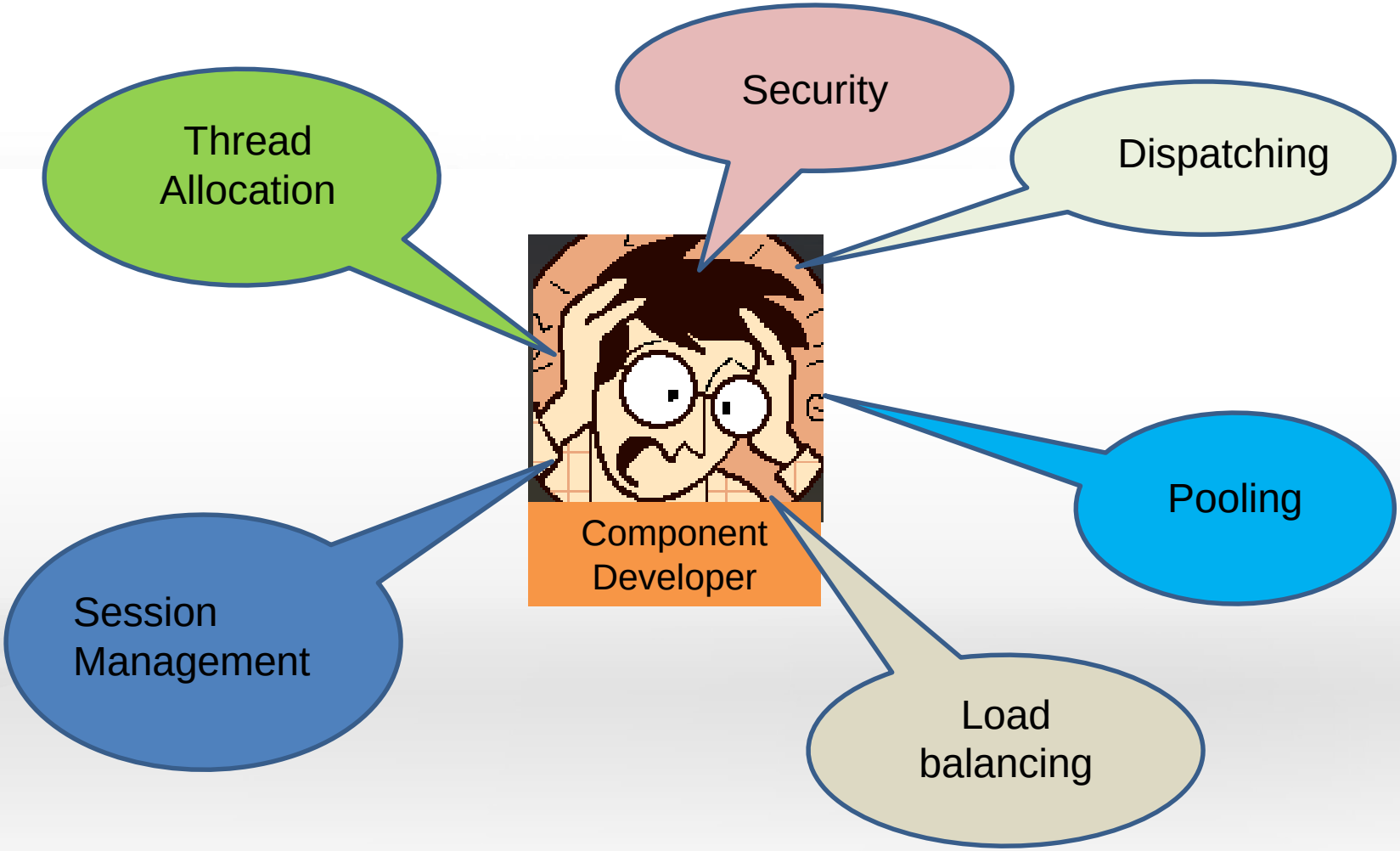
- JEE technologies:
  - Web Services Technologies
    - Java API for XML-based Web Services (JAX- WS)
    - Java API for XML-based RPC (JAX-RPC)
    - Java Architecture for XML Binding
    - SOAP with Attachments API for Java (SAAJ)
    - Streaming API for XML
    - Web Service Metadata for the Java Platform
  - Management and Security Technologies
    - JEE Management
    - Authentication and Authorization

JEE Containers

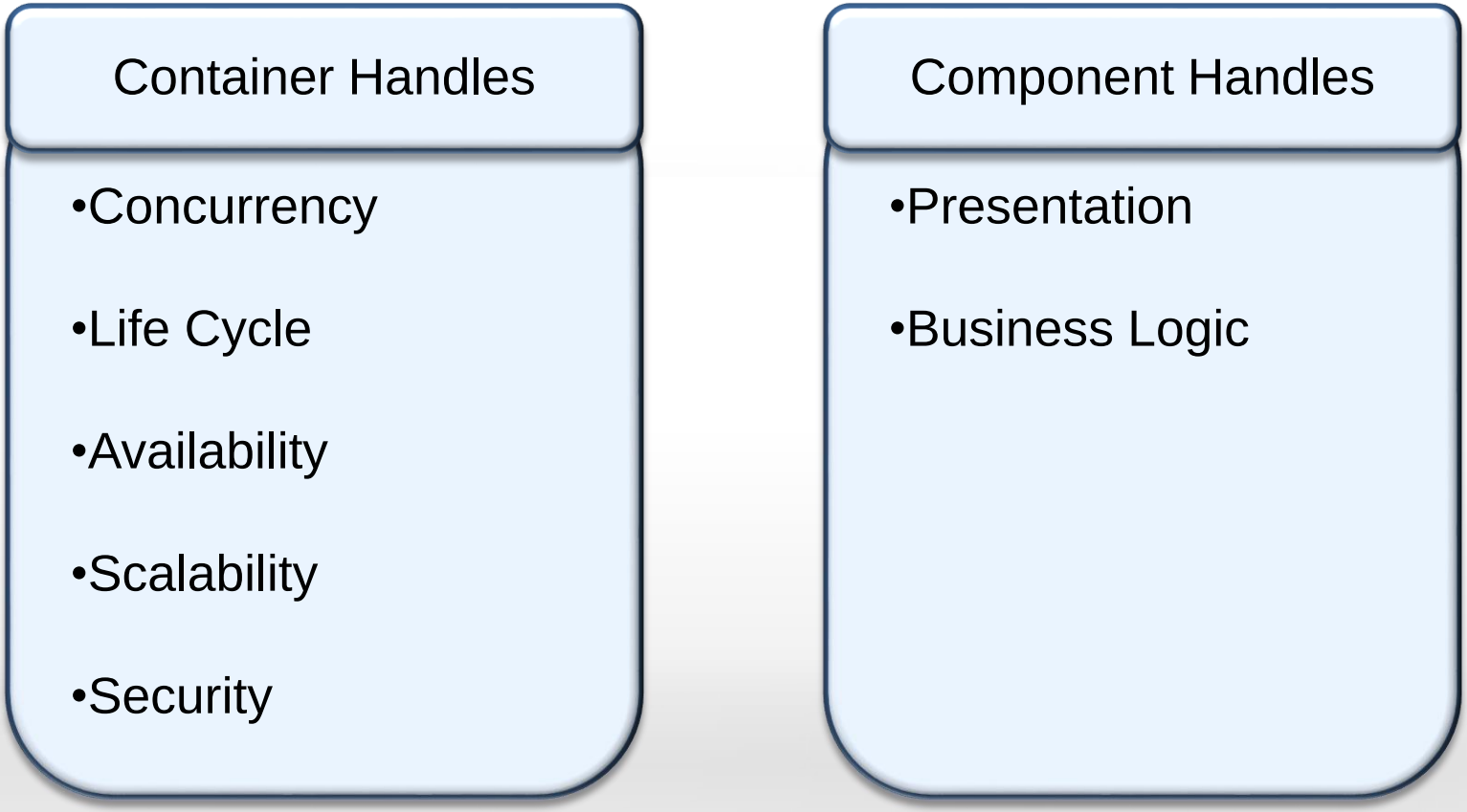


JSP: Java Server Pages
EJB: Enterprise Java Bean
RMI: Remote Method Invocation
HTTP: Hyper Text Transfer Protocol

JEE server key services



Role of Container and Components



Servlet

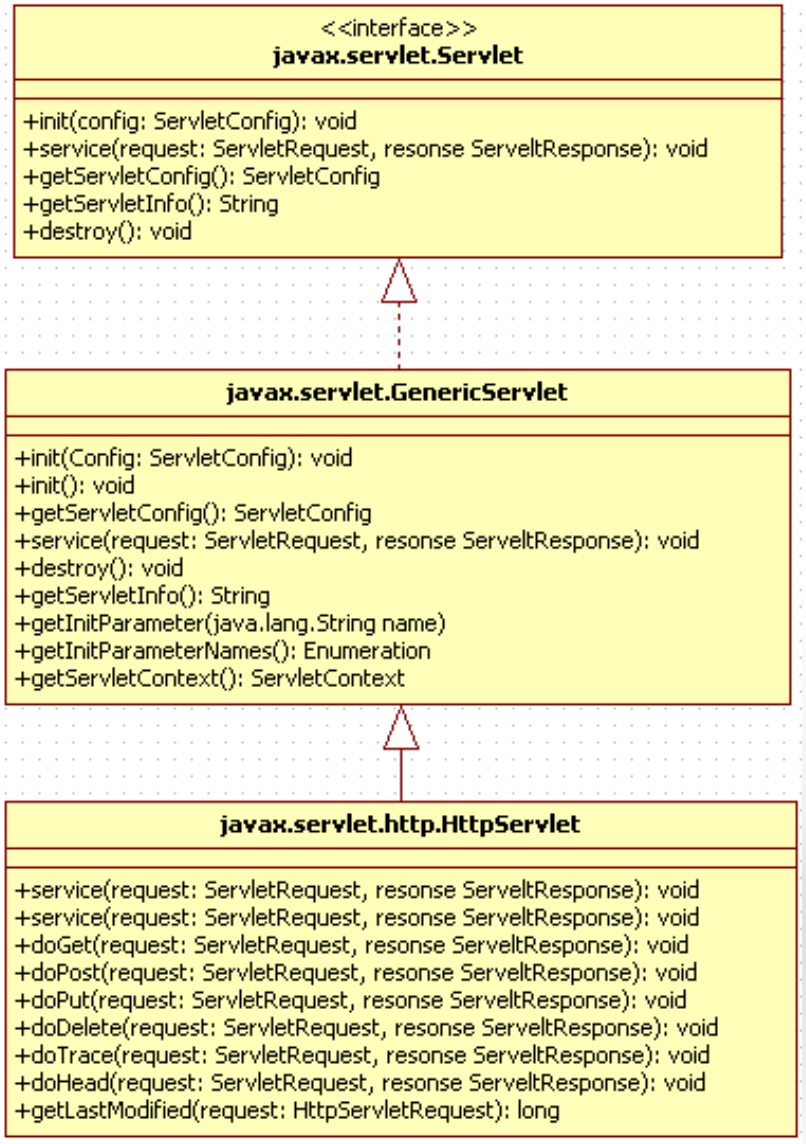
- Servlet technology is used to build dynamic web component using Java programming language
- Servlet's executes within a Web Container.
- Servlet's receives the request and generates the response dynamically based on the request.
- Servlet's are not tied to a specific client-server protocol, but are most often used with the HTTP protocol. Therefore, the word "Servlet" is often used in the meaning of "HTTP Servlet".

Servlet

- Servlet’s typically run on multithreaded servers, so be aware that a Servlet must handle concurrent requests and be careful to synchronize access to shared resources.

- Shared resources include in-memory data such as instance or class variables and external objects such as files, database connections, and network connections.

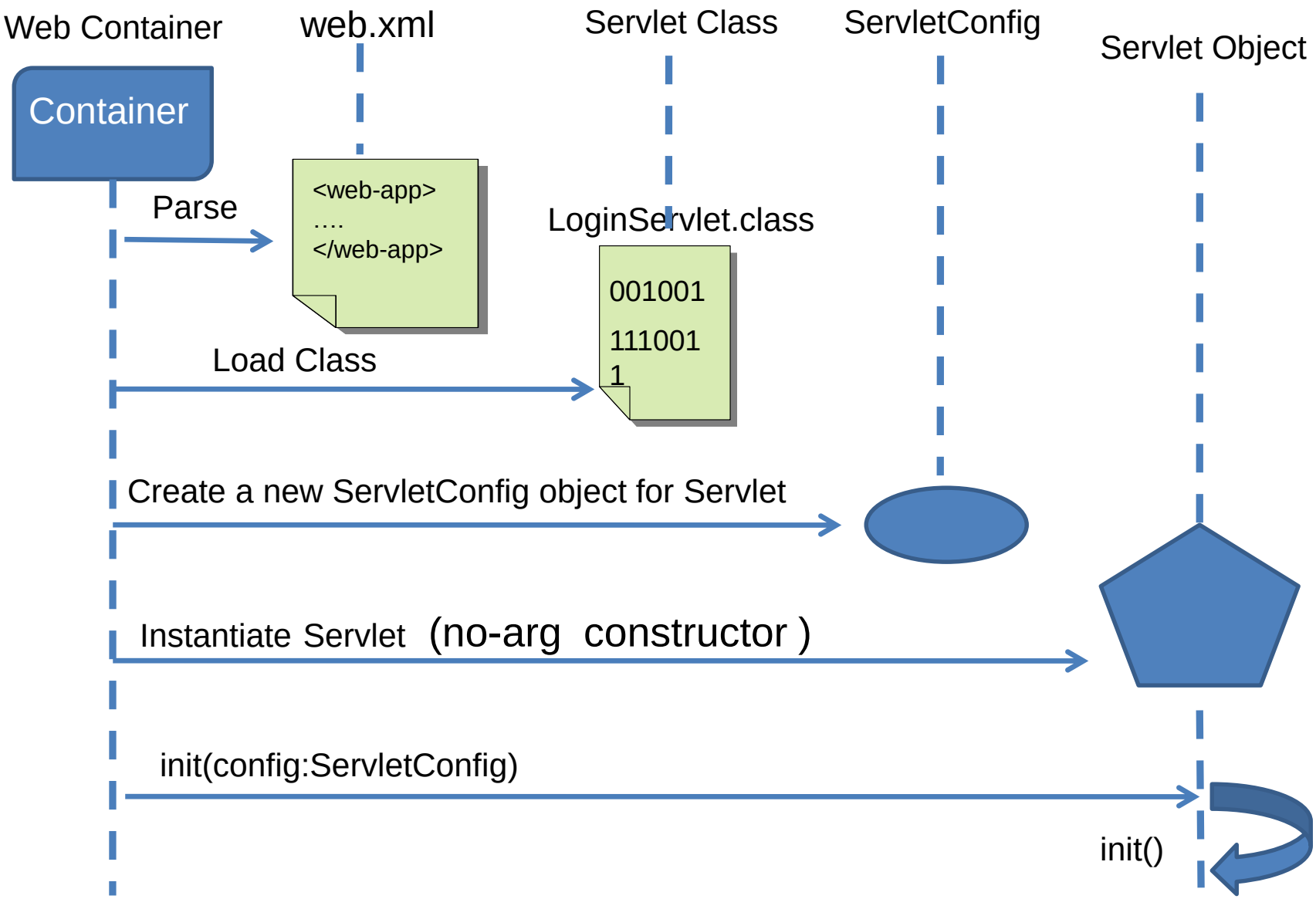
Servlet API


<<interface>>
javax.servlet.Servlet
+init(config: ServletConfig): void
+service(request: ServletRequest, resonse ServeltResponse): void
+getServletConfig(): ServletConfig
+getServletInfo(): String
+destroy(): void
javax.servlet.GenericServlet
+init(Config: ServletConfig): void
+init(): void
+getServletConfig(): ServletConfig
+service(request: ServletRequest, resonse ServeltResponse): void
+destroy(): void
+getServletInfo(): String
+getInitParameter(java.lang.String name)
+getInitParameterNames(): Enumeration
+getServletContext(): ServletContext
javax.servlet.http.HttpServlet
+service(request: ServletRequest, resonse ServeltResponse): void
+service(request: ServletRequest, resonse ServeltResponse): void
+doGet(request: ServletRequest, resonse ServeltResponse): void
+doPost(request: ServletRequest, resonse ServeltResponse): void
+doPut(request: ServletRequest, resonse ServeltResponse): void
+doDelete(request: ServletRequest, resonse ServeltResponse): void
+doTrace(request: ServletRequest, resonse ServeltResponse): void
+doHead(request: ServletRequest, resonse ServeltResponse): void
+getLastModified(request: HttpServletRequest): long

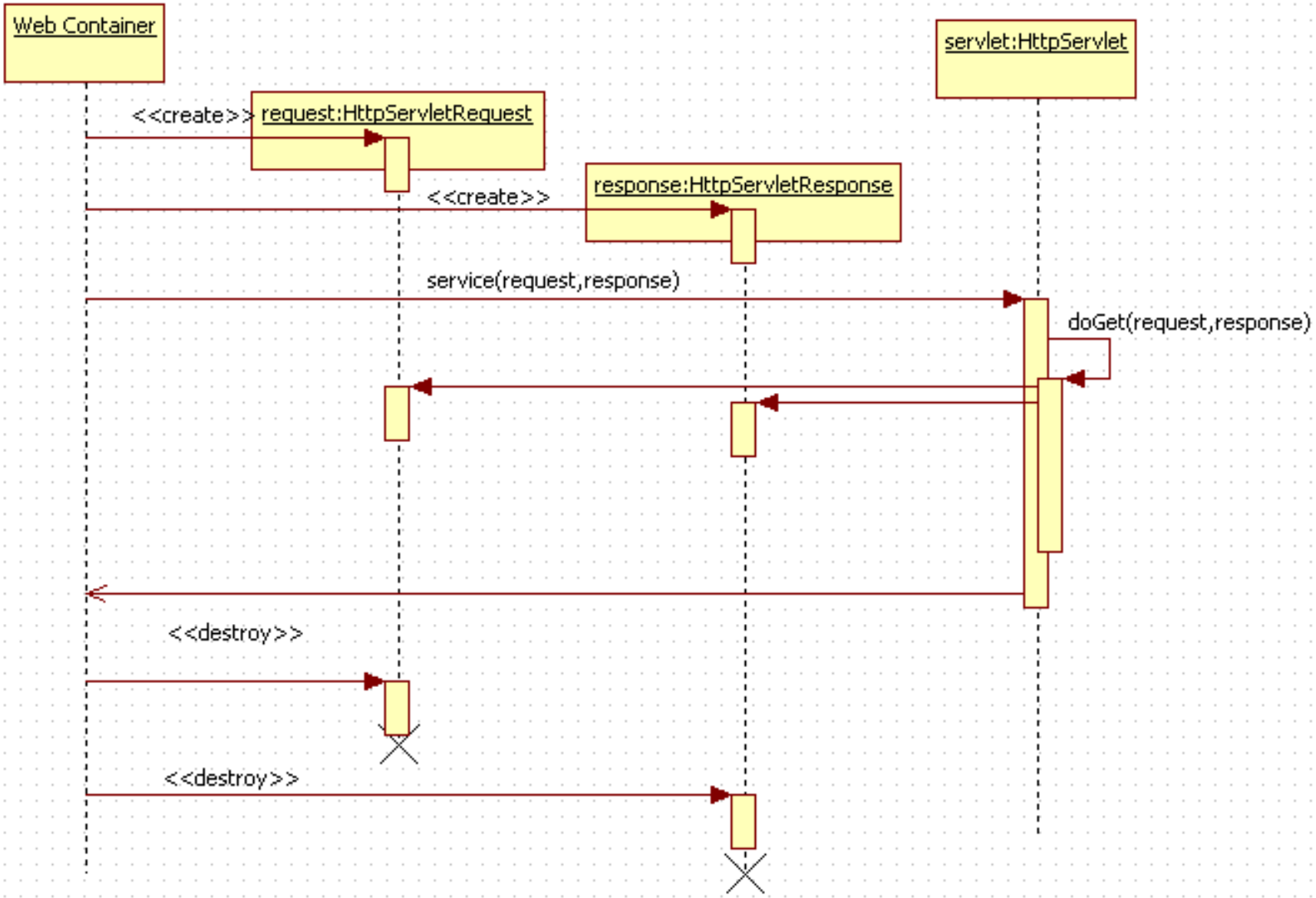
The Deployment Descriptor

- A **deployment descriptor** (DD) refers to a configuration file for an object that is deployed to a container.
- XML is used for the syntax of these deployment descriptors.
- The web.xml is the deployment descriptor for web applications.
  - File should reside in WEB-INF folder
- The persistence.xml is the deployment descriptor for Java Persistence API
  - File should reside in META-INF folder
- The application.xml is the deployment descriptor for complete Enterprise application
  - File should reside in META-INF folder

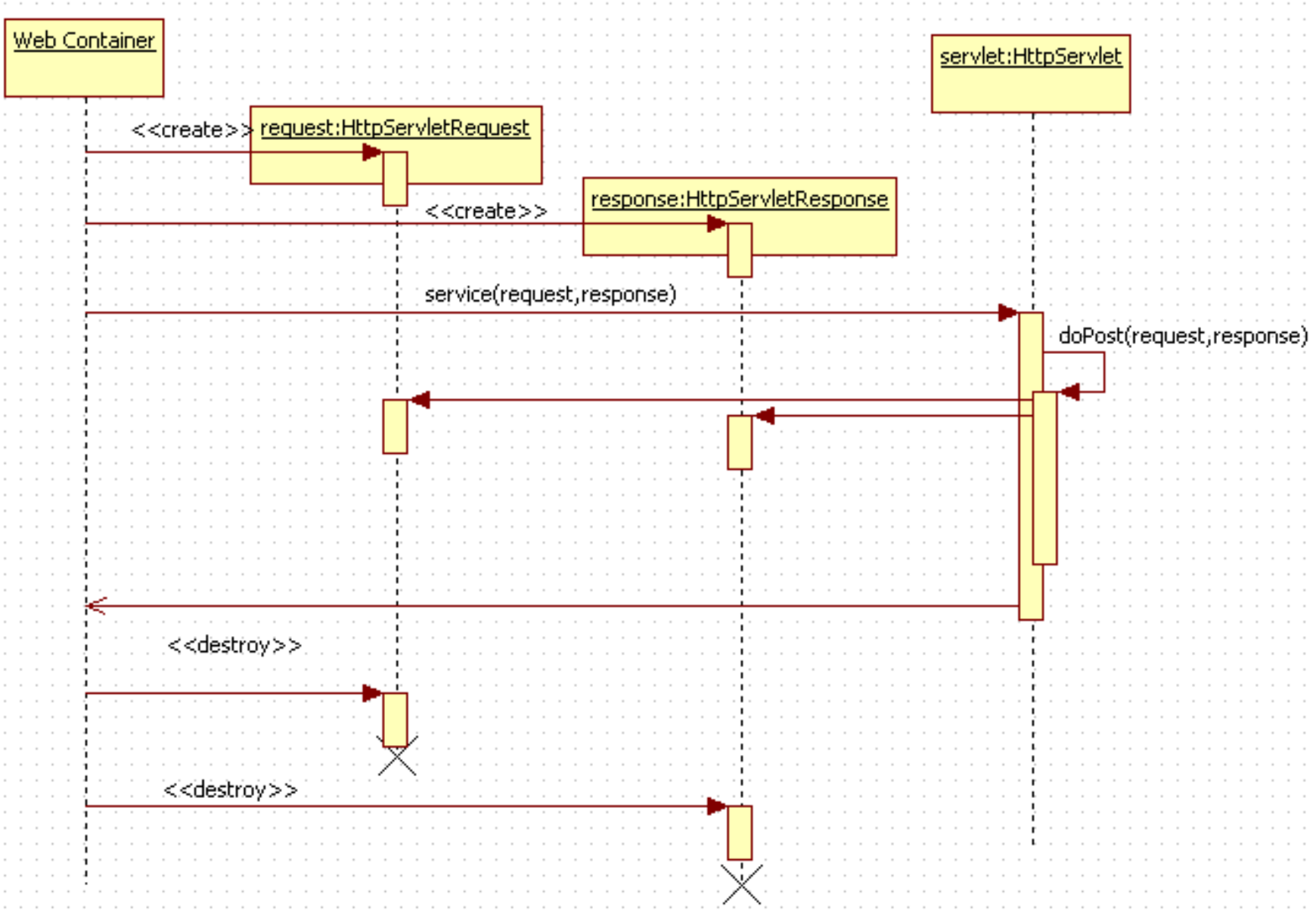
Servlet's Deployment



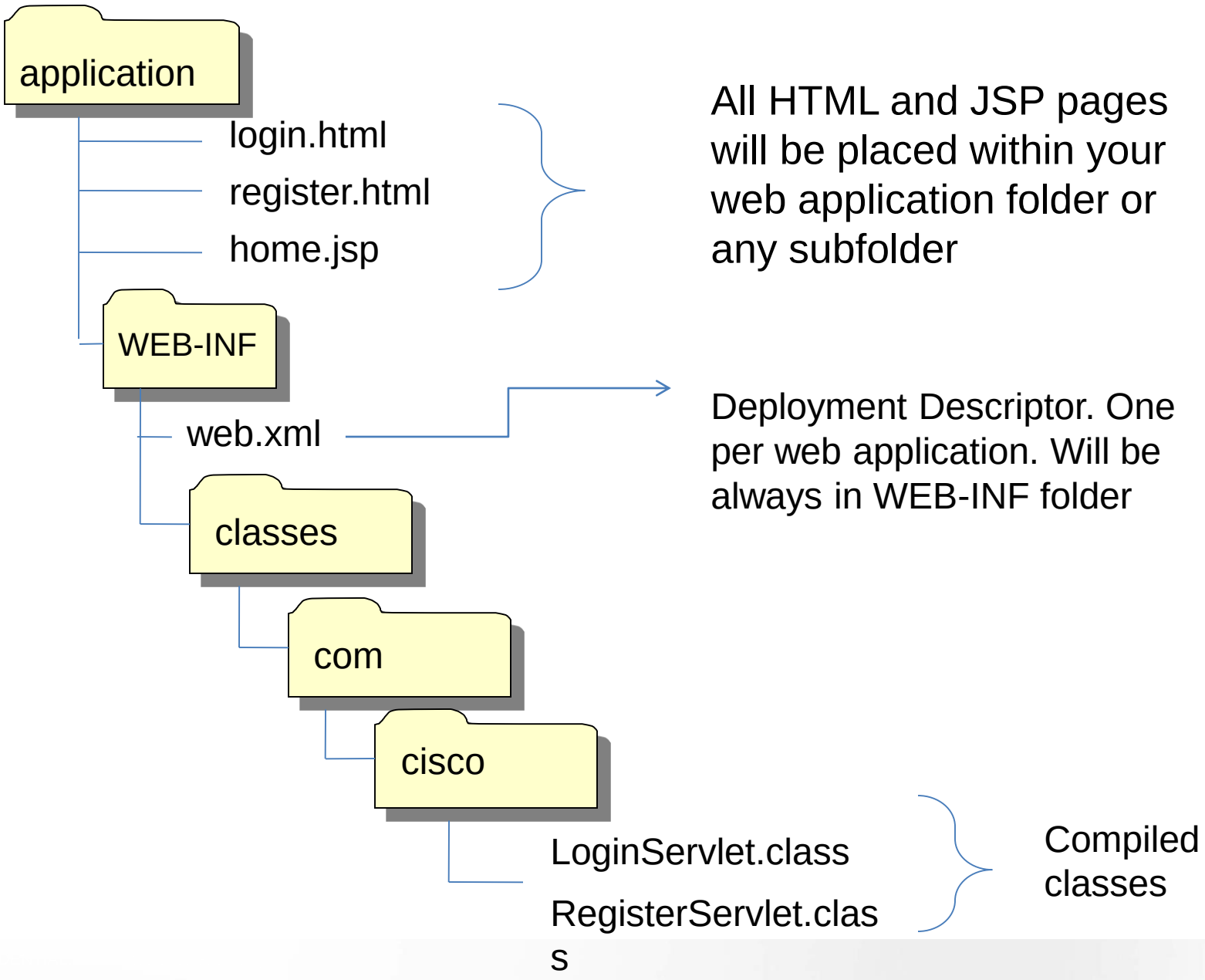
Sequence diagram for HTTP GET request



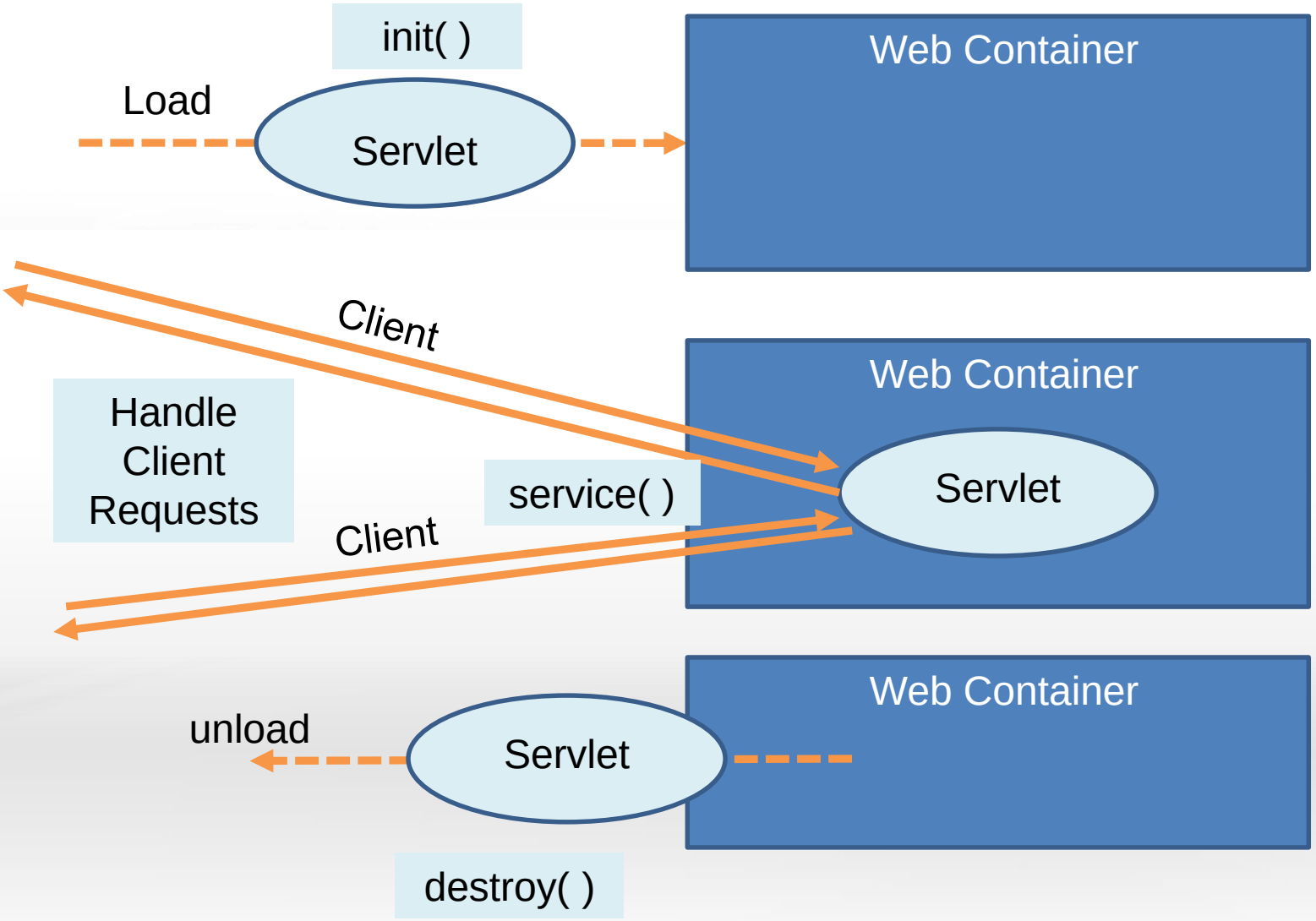
Sequence diagram for HTTP POST request



Web application folder structure

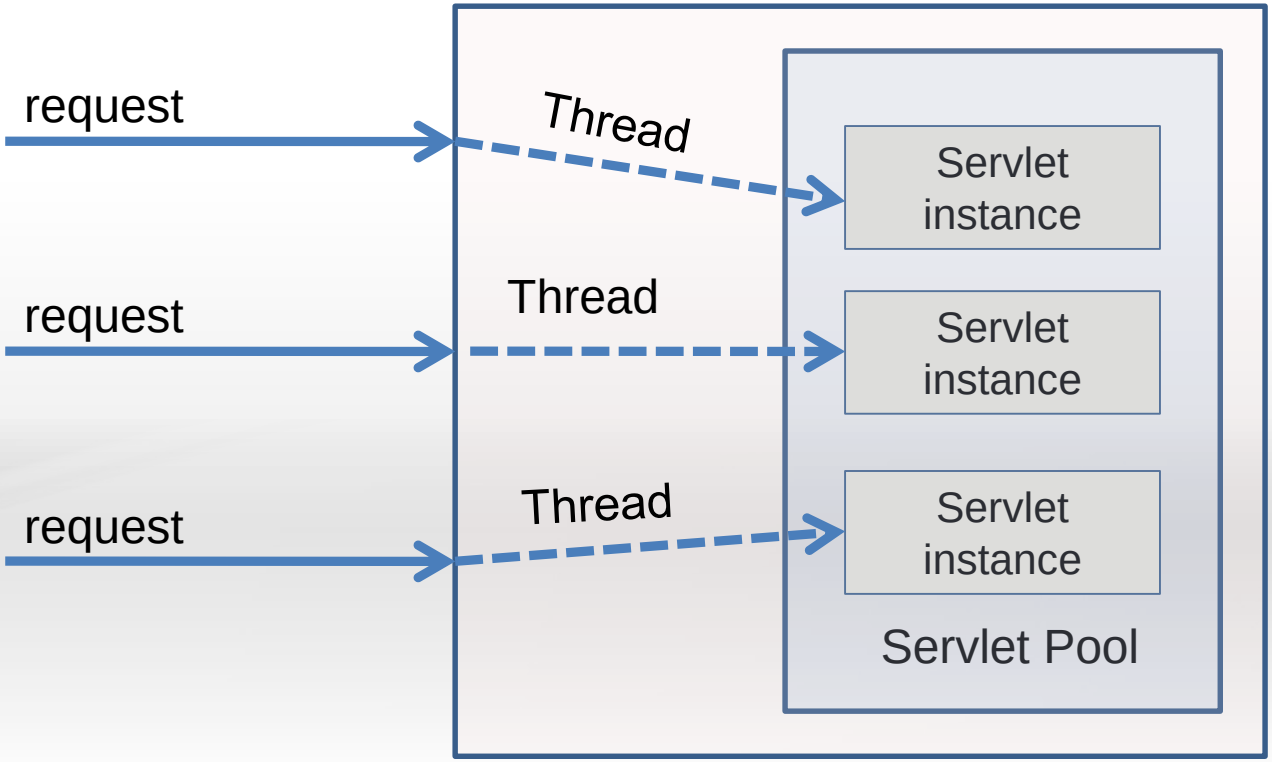


Servlet's Life-Cycle



Single-Thread Model

- By default one servlet instance is created per registered servlet name, that servlet will be multi threaded.
- We can have a pool of servlet instances created for each of its names by marking the Servlet as single threaded by SingleThreadModel marker interface.



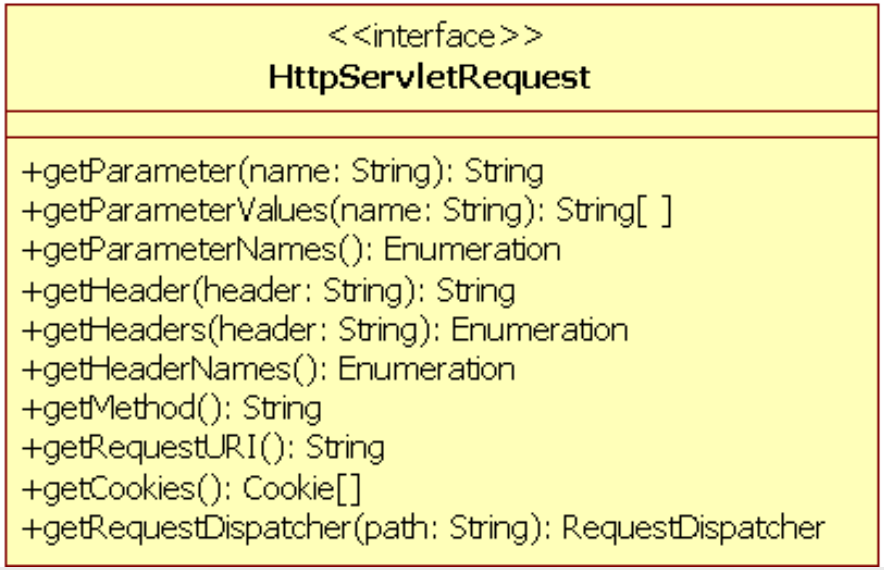
HttpServletRequest

- An HttpServletRequest object provides access to HTTP header data, and arguments that the client sent as part of the request.

String **getParameter**(String name)
Returns the value of a request parameter as a String, or null if the parameter does not exist.

String[ ] **getParameterValues**(String name)
Returns an array of String objects containing all of the values for the given request parameter , or null if the parameter does not exist.

public String **getHeader**(String name)
Returns the value of the specified request header as a String.



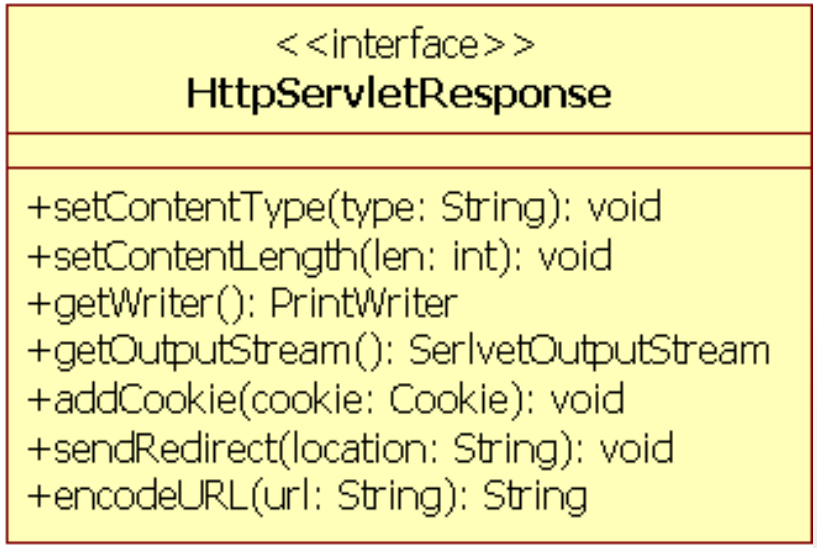
HttpServletResponse

- HttpServletResponse object assists a servlet in sending a response to the client.

•ServletOutputStream getOutputStream().
This method returns ServletOutputStream to send binary data in a body response.

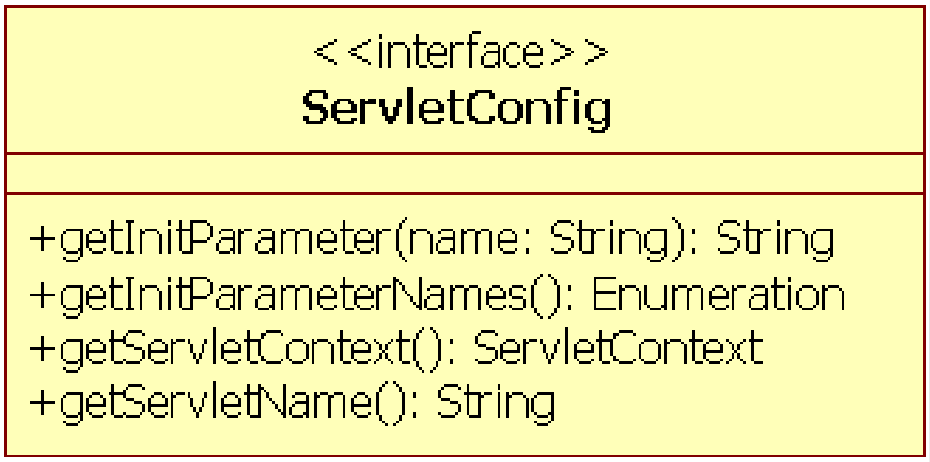
•PrintWriter getWriter( )
This method returns Printriter to send character data,.

•void sendRedirect (String location)
Sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer



ServletConfig

- An Object of type ServletConfig is created by the Web Container to pass information to servlet during initialization.
- For every Servlet Web container creates one object of type ServletConfig.
- Web Container passes the ServletConfig object created to init(ServletConfig config) method.



Java Server Pages (JSP)

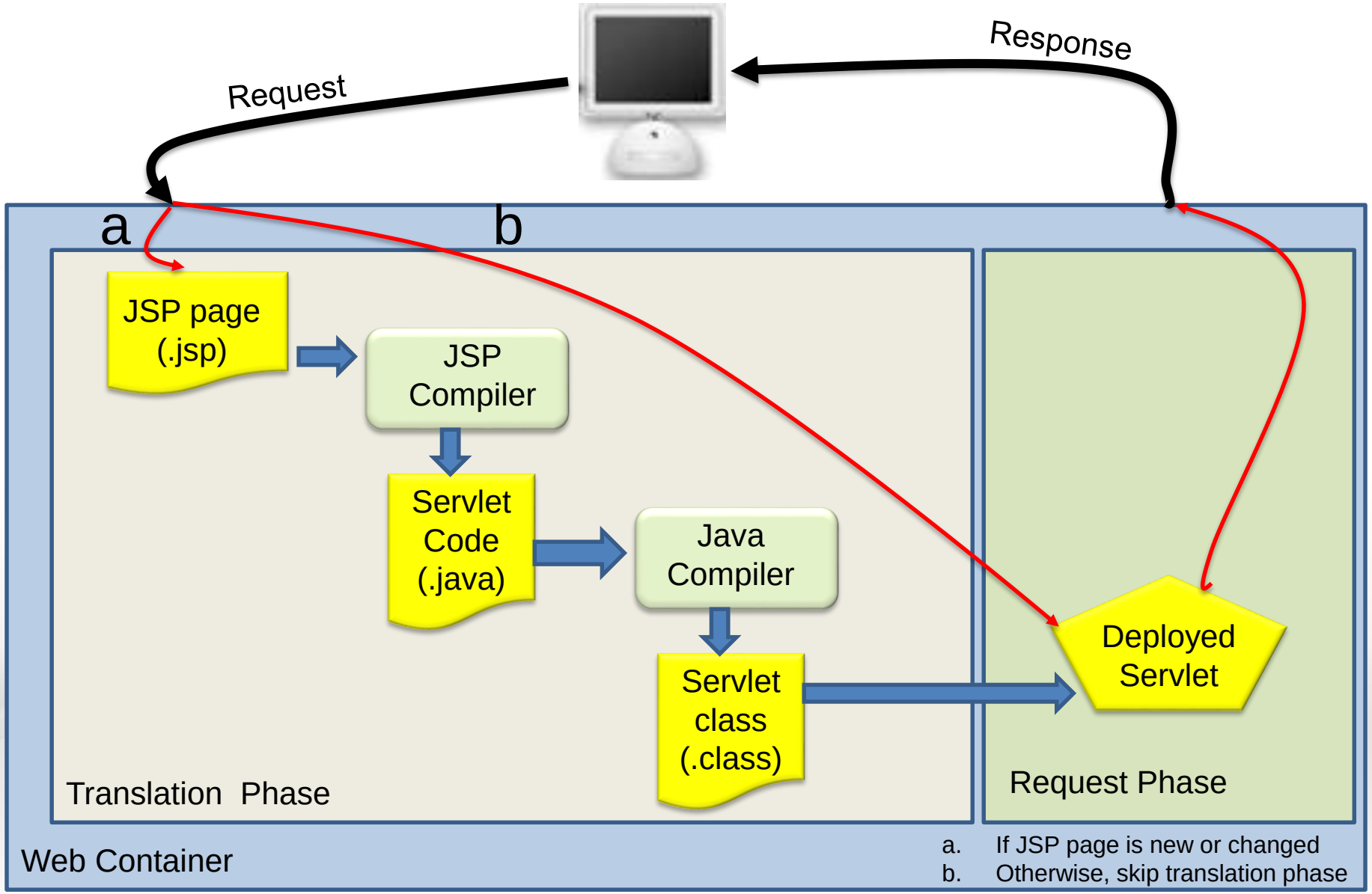
- JSP technology allows you to easily create web content that has both static and dynamic content.
- JSP static content can consist of:
  - HTML tags and plain text.
- JSP dynamic content can be generated using:
  - Java code and XML tags.

Java Server Pages (JSP)

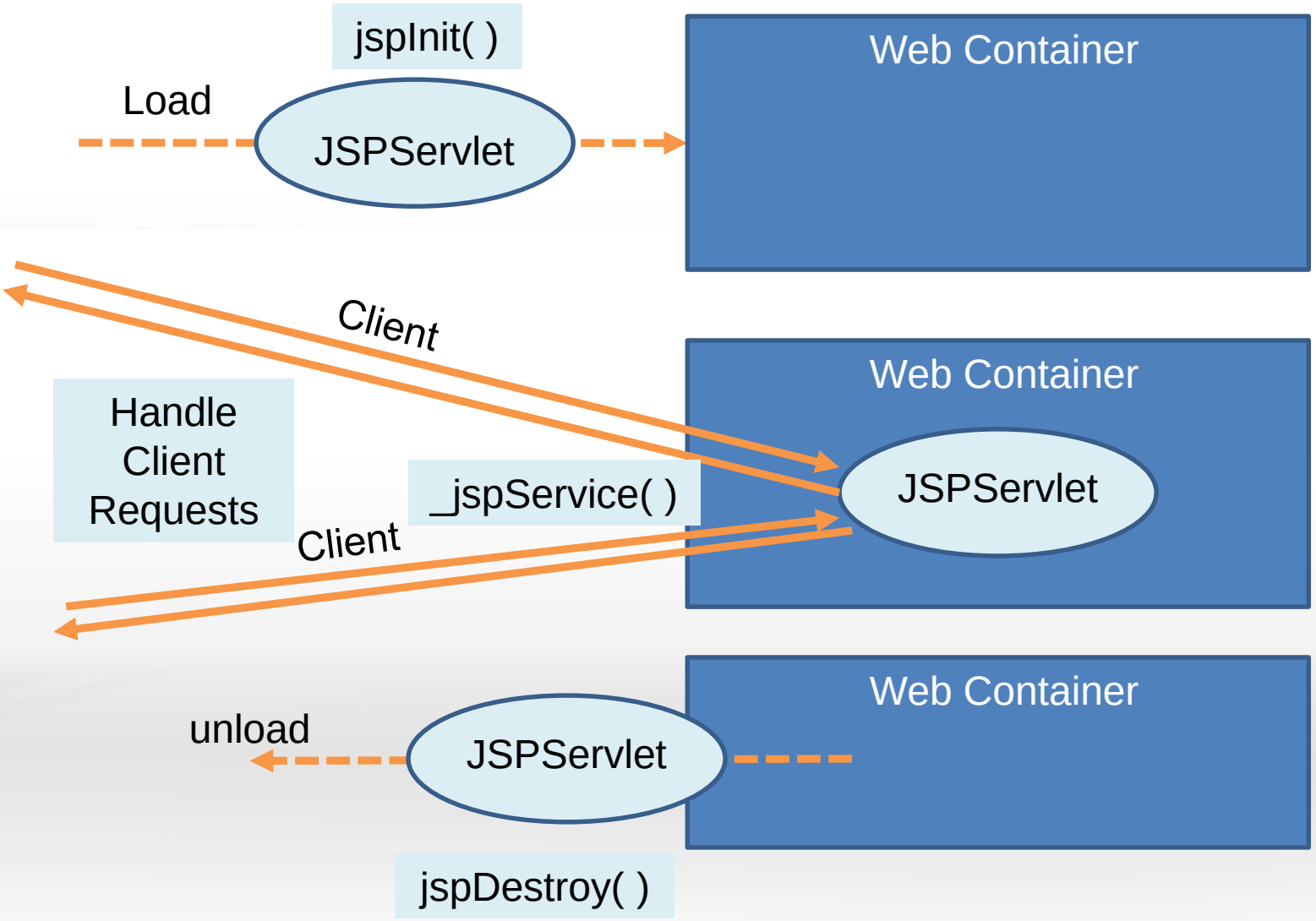
- A simple JSP page: (Blue: Static, Red: Dynamic)

```
<html>
<body>
    Hello World! <br />
    Current time in server is : <%= new java.util.Date( ) %>
 </body>
 </html>
```

JSP Translation Phase / Request Processing Phase



JSP life cycle methods



JSP implicit objects

JSP container provides you a list of instantiated objects which are called *implicit object*

- request ----> Object of type HttpServletRequest
- response ----> Object of type HttpServletResponse
- out ----> Object of type JspWriter
- config ----> Object of type ServletConfig

- Other implicit Objects (explained in later sessions)
- session ----> Object of type HttpSession
- application ----> Object of type ServletContext
- pageContext ----> Object of type PageContext

JSP Scripting elements

- JSP scripting elements are used to create and access objects, define methods, and manage the flow of control.

- JSP Scripting elements are
  - Declarations
  - Scriptlets
  - Expressions
  - JSP comments

Declarations

- A *JSP declaration* is used to declare instance /static variables and methods in a page's scripting language.
- The syntax for a declaration is as follows:

  <%! scripting language declaration %>

Example:

```
<%!
    private double computeSum(double d1, double d2) {
       return (d1 + d2);
    }
%>
```

JSP Scripting elements

Scriptlets

- A *JSP scriptlet* is used to contain any code fragment that is valid for the scripting language used in a page.
- The syntax for a scriptlet is as follows:
  - <% scripting language statements %>

Example:

```
<%
    String queryData = request.getQueryString();
    out.println("Data: " + queryData);
    out.println("Sum : " + computeSum(1.2,4.5);
%>
```

JSP Scripting elements

Expressions

- A *JSP expression* is used to insert the value of a expression. The value of the expression is converted into a String object and inserts it into the implicit out object.
- The syntax for an expression is as follows:
  - <%= scripting language expression %>

Example:

- Display current time using Date class

  Current time: <%= new java.util.Date() %>

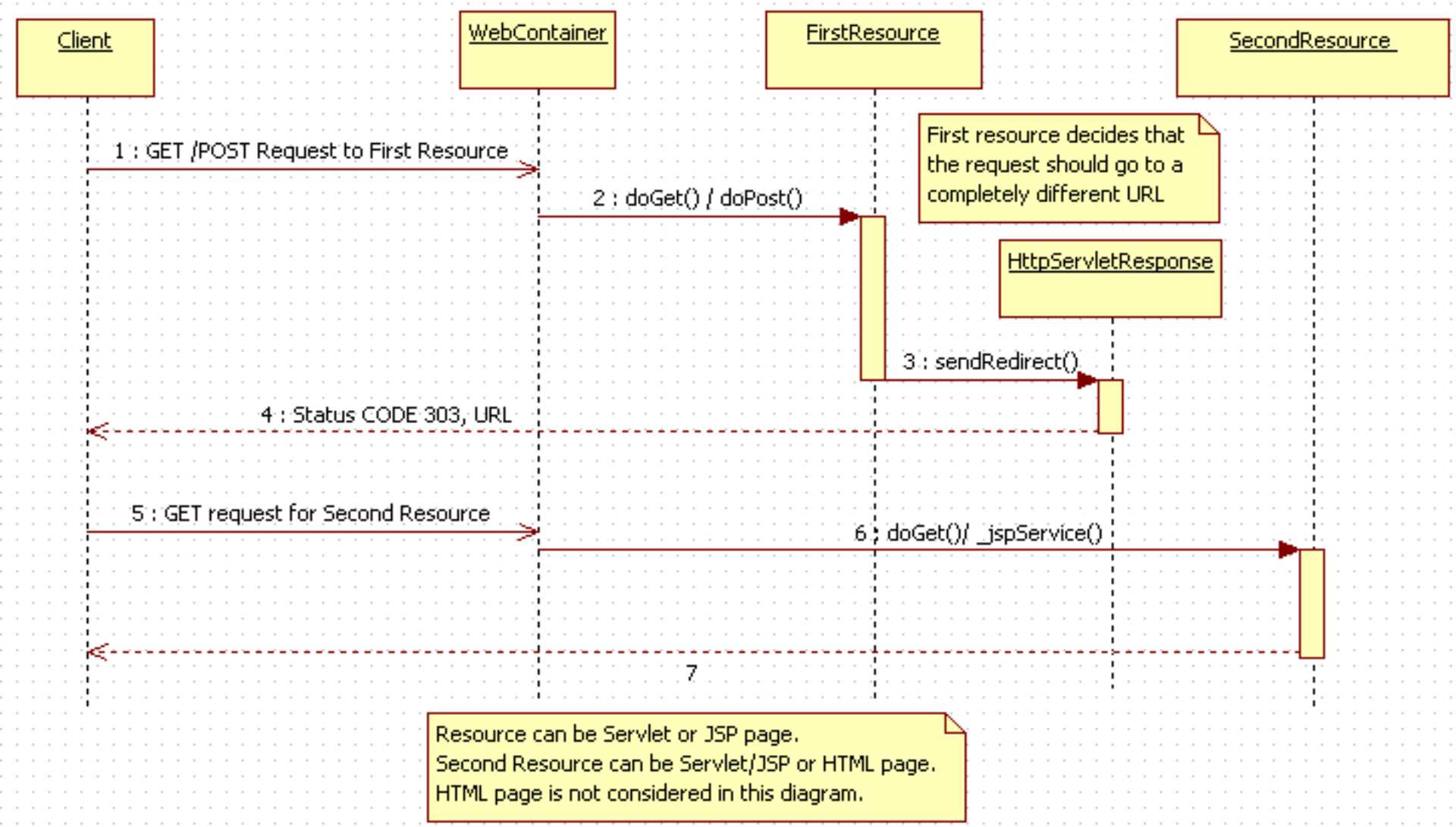
- Use implicit objects

  Name:<%= request.getParameter("name") %>

Redirection

- Client-Side Redirection
  - Redirection done by the web browser
  - Use response.sendRedirect(String URL)
- Server-Side Redirection
  - Redirection done by the web container
  - In Servlet use forward(HttpServletRequest, HttpServletResponse) method of ServletRequestDispatcher
  - In JSP use <jsp:forward /> tag

Client Side Redirection



Code Snippet: Client Side Redirection

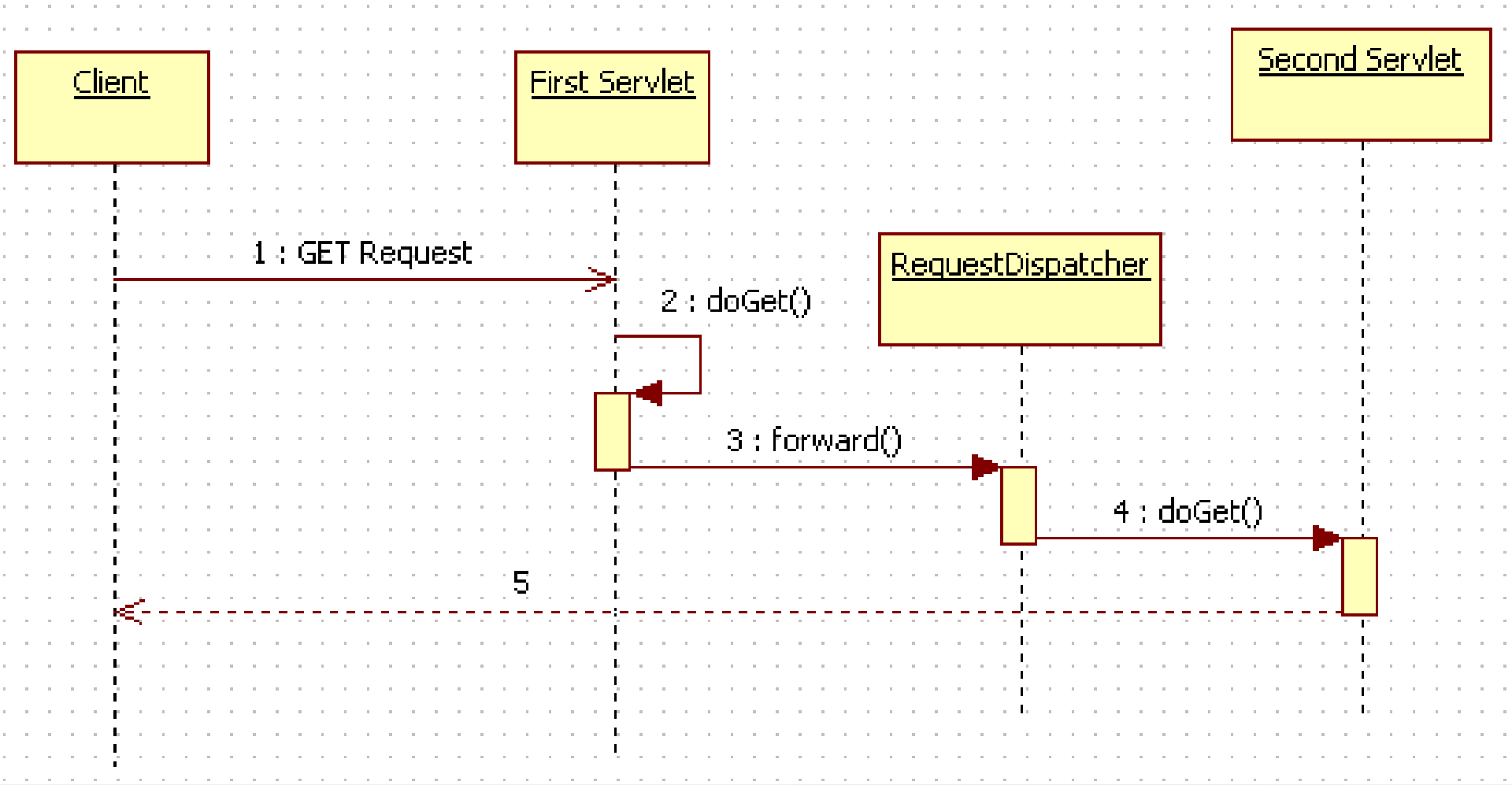
```
/*
 * read parameters userName and password
 */
String userName = request.getParameter("userName");
String password = request.getParameter("password");
/*
 * Assume userName = "James" with password="secret123"
 * is valid user credentials.
 */
if (userName.equals("James") && password.equals("secret123")) {
    /*
     * if user has entered valid credentials redirect to home.jsp
     */
    response.sendRedirect("home.jsp");
} else {
    /*
     * if user has not entered valid credentials
     * redirect him/her to login.jsp with Invalid User/Password message.
     */
    response.sendRedirect("login.jsp?message=Invalid User/Password");
}
```

RequestDispatcher

- Transferring Control to another Web Component
  - RequestDispatcher can be used to transfer control to another web component by invoking forward() method.
  - This mechanism is useful for multistage processing of data.

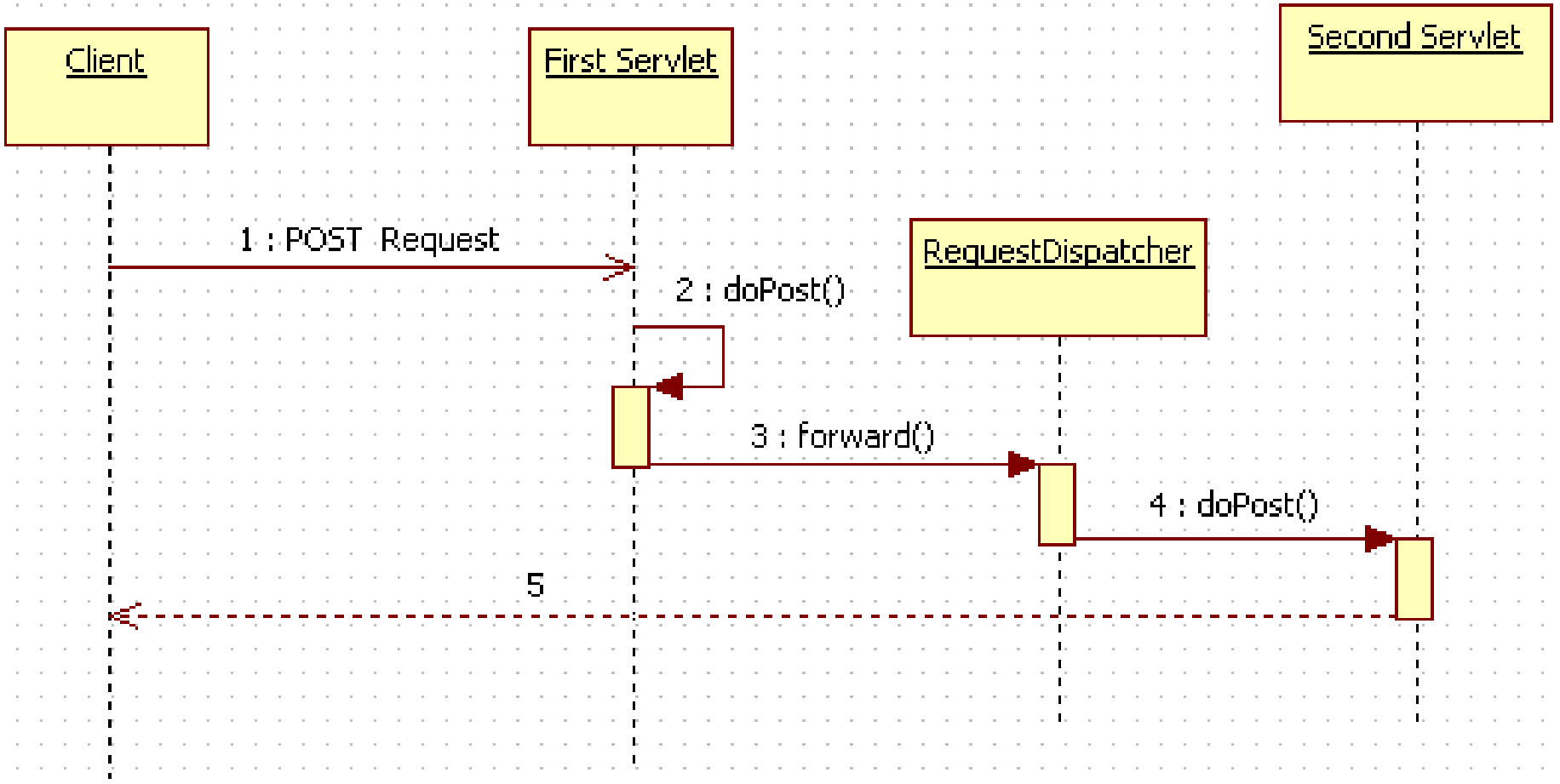
- **Including Other Resources in the Response**
  - RequestDispatcher can also be used to include another Web resource using include() method, in the response returned from a Web Component.
  - This mechanism is useful for including common contents like banner, copyright information and menus as response from all the servlets.

RequestDispatcher's forward method



- Server Side Redirection using forward() method of RequestDispatcher.

RequestDispatcher's forward method



The method of REQUEST does not change in server side redirection.

Code Snippet: Server Side Redirection

```
String target = "login.jsp?message=Invalid User/Password";
/*
 * read parameters userName and password
 */
String userName = request.getParameter("userName");
String password = request.getParameter("password");
/*
 * Assume userName = "James" with password="secret123"
 * is valid user credentials.
 */
if (userName.equals("James") && password.equals("secret123")) {
    /*
     * if user has entered valid credentials redirect to home.jsp
     */
     target = "home.jsp";
}
/*
 * You can get a RequestDispatcher object from either a request or the ServletContext;
 * A request can take a relative path (that is, one that does not begin with a /),
 * but the ServletContext requires an absolute path
 */
RequestDispatcher dispatcher = request.getRequestDispatcher(target);
dispatcher.forward(request, response);
```