

Advanced Java

Lesson 2—Java Servlet I



Learning Objectives



- ✓ Discuss Java Servlet Architecture
- ✓ Explain the important terms related to Java Servlets
- ✓ Understand Servlet Lifecycle
- ✓ Explain ServletsRequest, ServletResponse, ServletConfig, and, ServletContext
- ✓ List the steps to configure and deploy Servlet
- ✓ Explain Servlet Collaboration

Advanced Java

Topic 1—Introduction to Java Servlet

- Static and Dynamic Response
- Client-Server Architecture with Java Servlet
- Web Container
- Client-Server Architecture with Web Container

Why Servlet?

We have learned that web server generates two types of responses: Static and Dynamic.

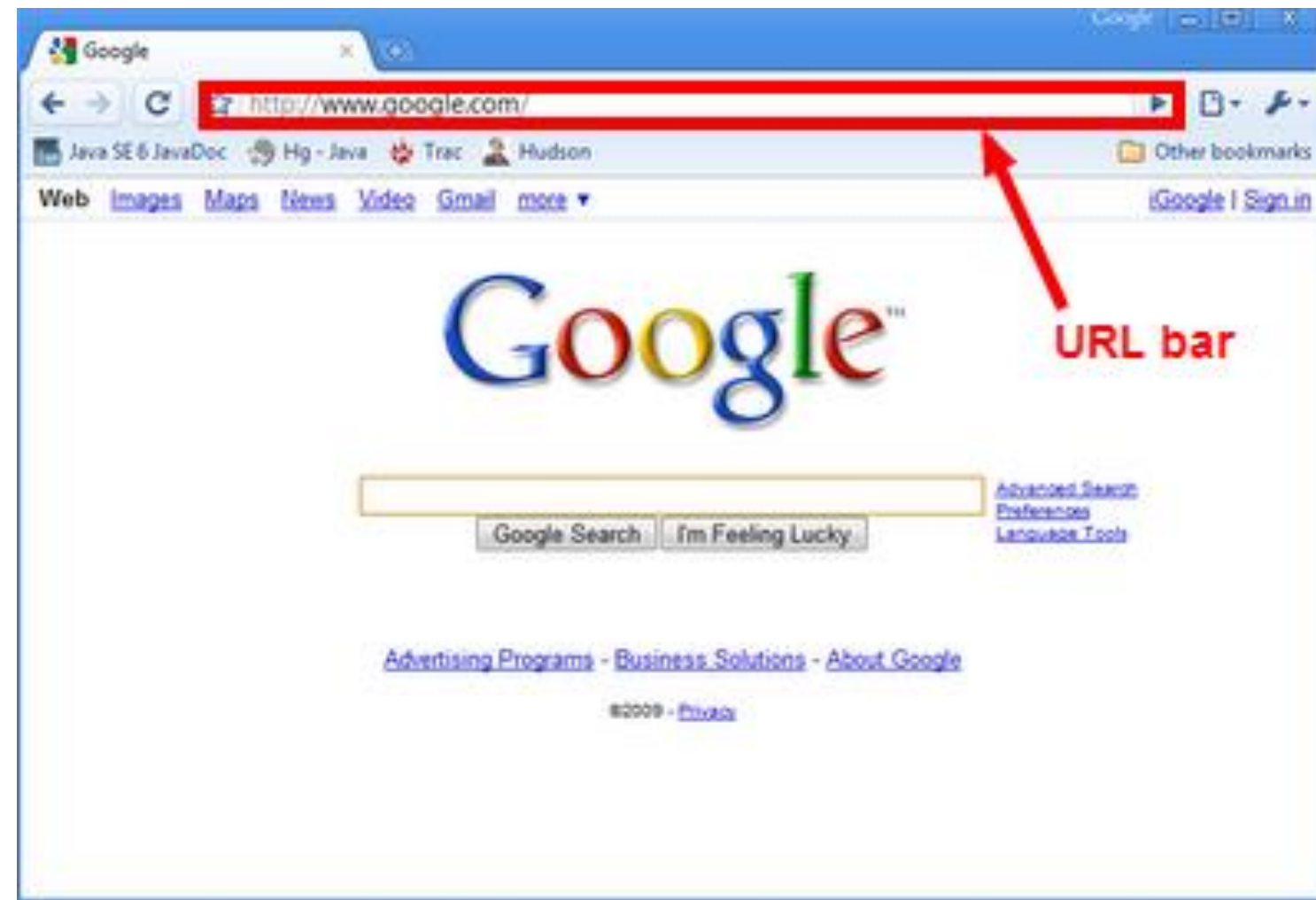
However, a web server can generate static responses in the form of HTML.

To generate dynamic response, a programming language that performs dynamic operations is needed.

Servlet is a Java program that can perform dynamic operations and send to web server. The web server then sends this response to web client.

Static Response

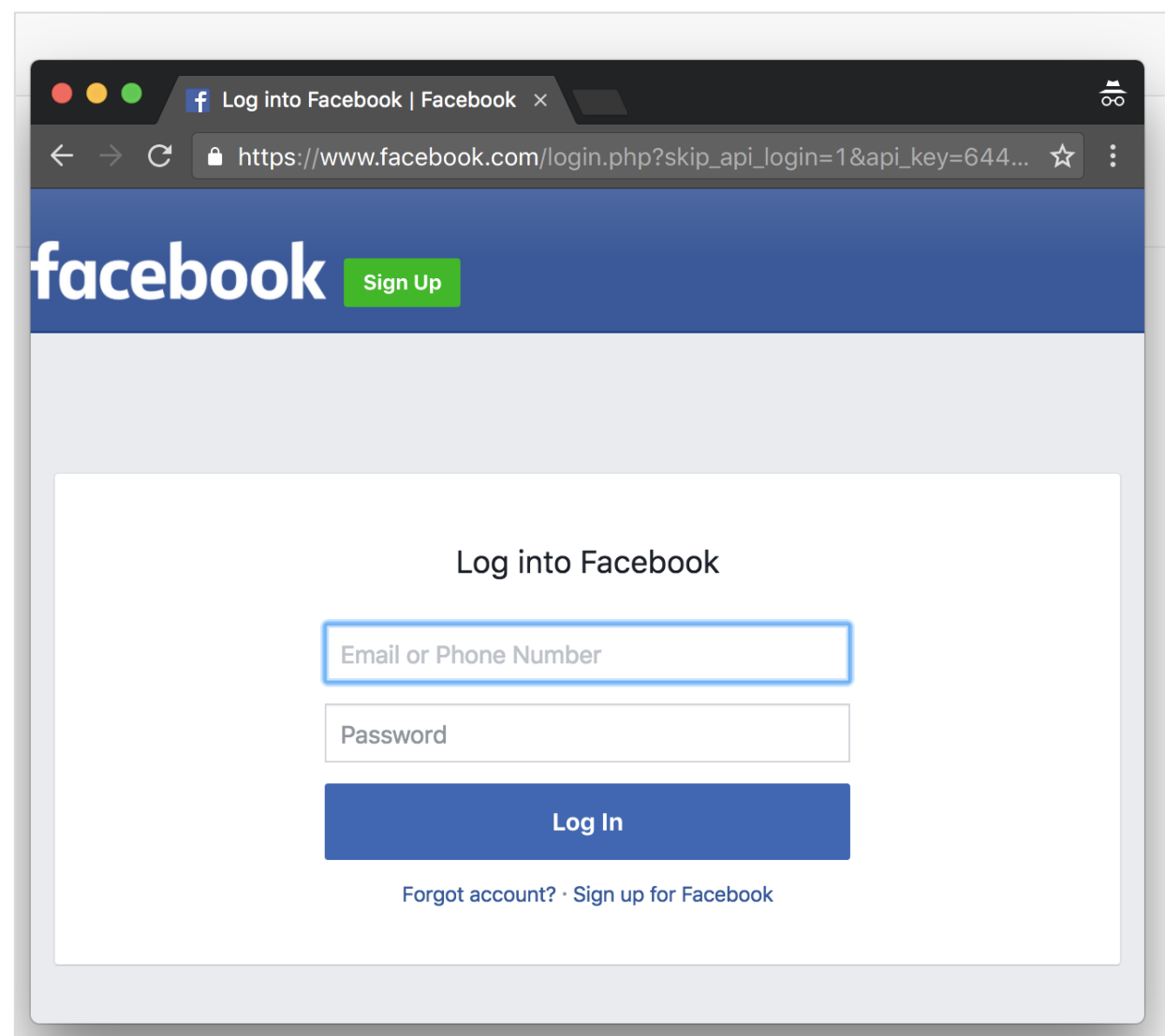
Every time you type <https://www.google.com/> in your browser, you are directed to the Google home page. This is called a static page as the web server displays the same HTML page every time the client requests.



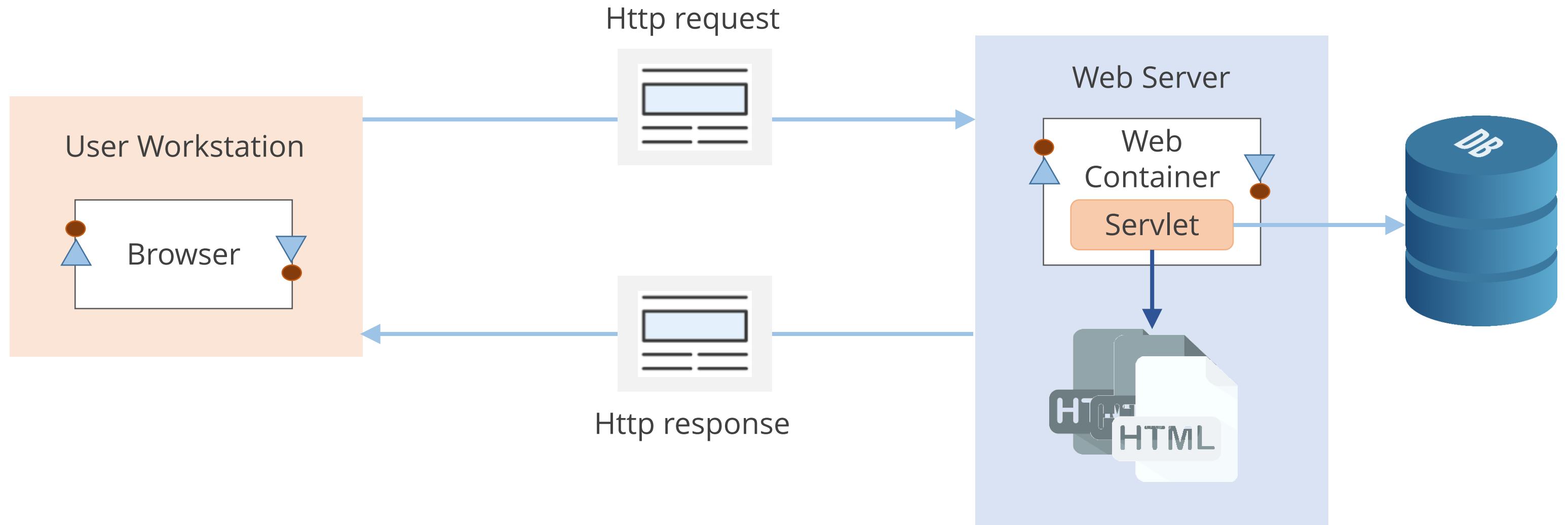
Dynamic Response

When users login to their Facebook account, different pages will be generated for different users. You would see the pages relevant to you.

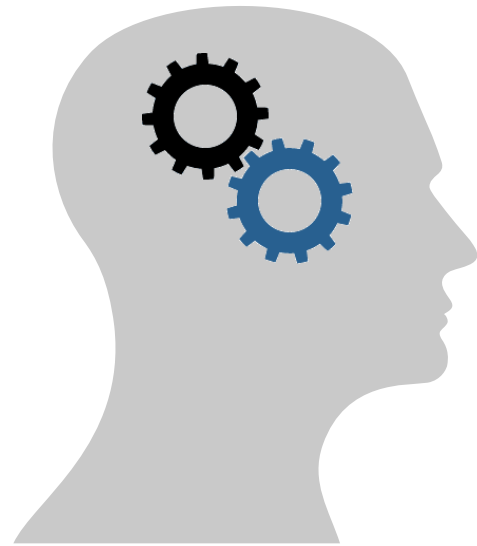
This requires multiple responses from the server, called dynamic response. This uses Servlets.



Client-Server Architecture with Java Servlet



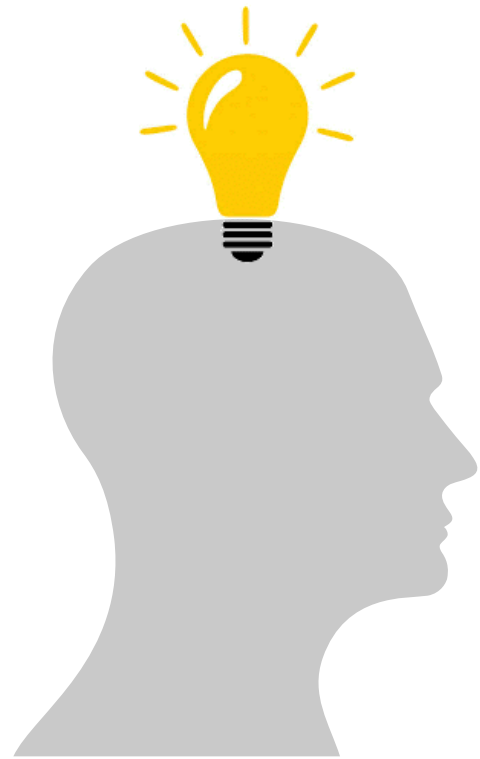
Think Through!



Web server is a program that sends HTTP response, and Servlet is a Java Program that deals with classes and objects.

How is the response generated in the HTTP format?

Think Through!



This is done using a “web container.”

Let’s understand what a web container is.

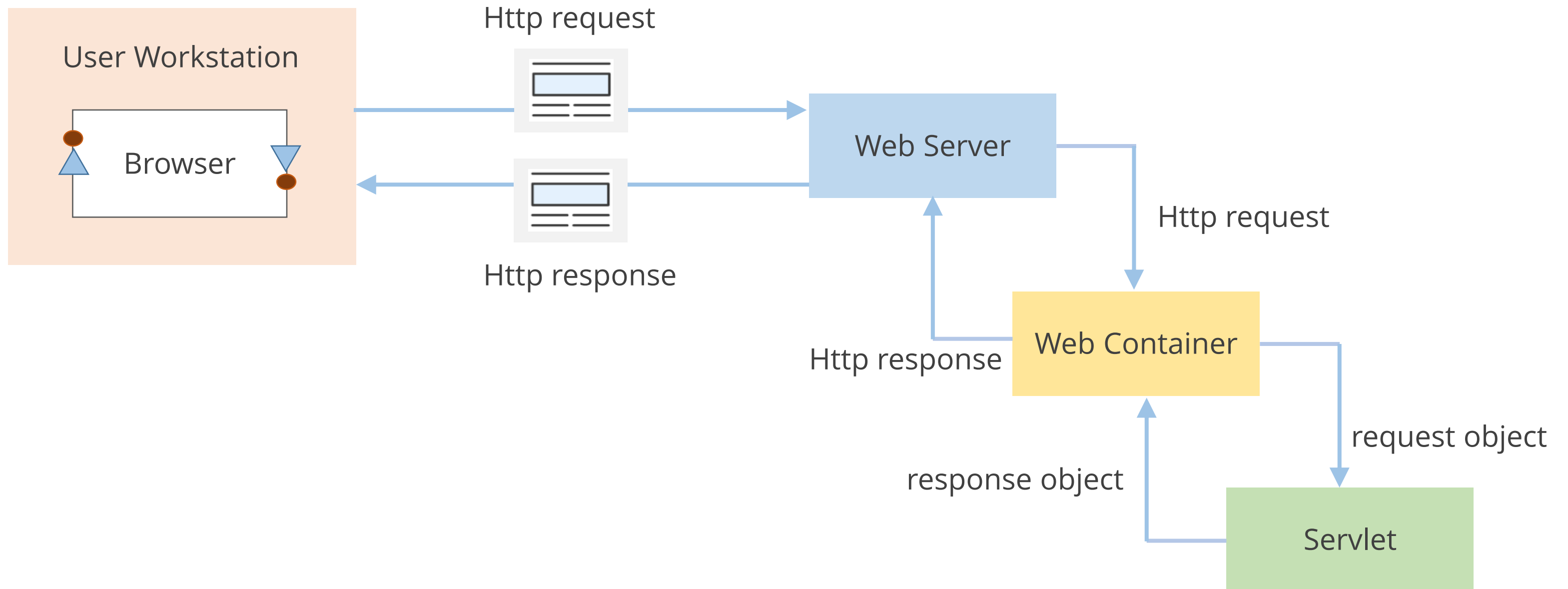
What is a Web Container?

A web container is built on top of the Java EE platform, and it implements the Servlet API and the services required to process HTTP (and other Transmission Control/Internet Protocol [TCP/IP]) requests.

- Java Servlets are components that must exist in a web container. Web container activates the Servlet that matches the requested URL by calling the service method on an instance of Servlet class.
- Activation of the service method for a given HTTP request is handled in a separate thread within the web container protocol.

Client-Server Architecture with Web Container

The lifecycle of a Servlet is controlled by the web container in which the Servlet has been deployed.



Advanced Java

Topic 2—Servlet API, Interface, and Methods

- Servlets API
- Servlet Interface and Classes
- Servlet API Interface
- Servlets API Hierarchy to Create Servlet
- Servlets Methods
- Generic Servlet Abstract Class
- Generic Servlet Abstract Class Methods
- HttpServlet Abstract Class Methods

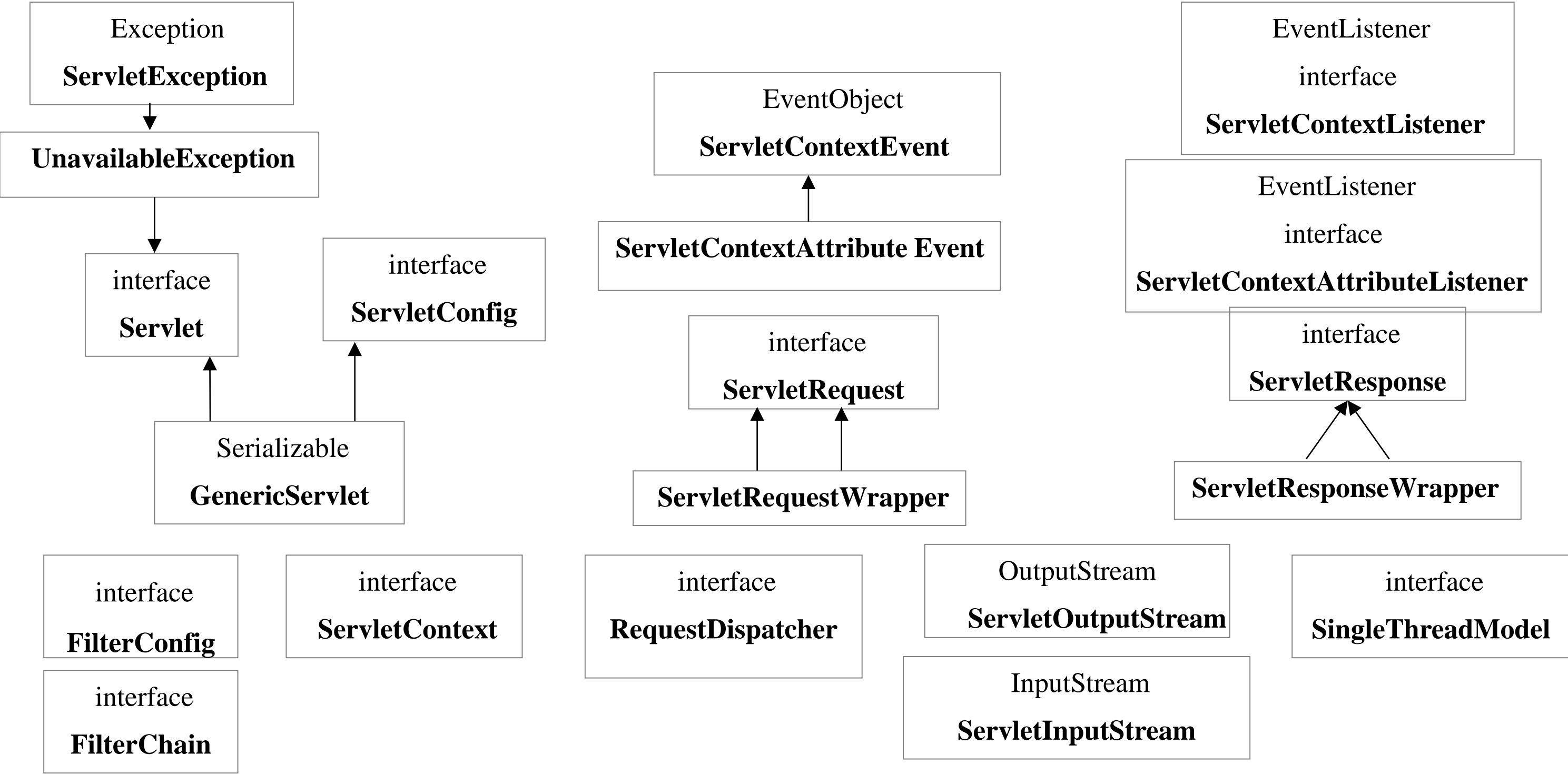
Servlets API

Servlet API contains a number of classes and interfaces that describe the contracts between a servlet class and the runtime environment provided for an instance by a conforming servlet container.

Servlet API provides the following two packages that contain its classes and interfaces:

<code>javax.servlet.*;</code>	Classes and interfaces defines the contracts between a Servlet class and the runtime environment provided for an instance of such a class by a conforming Servlet container.
<code>javax.servlet.http.*;</code>	Classes and interfaces defines the contracts between a Servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming Servlet container.

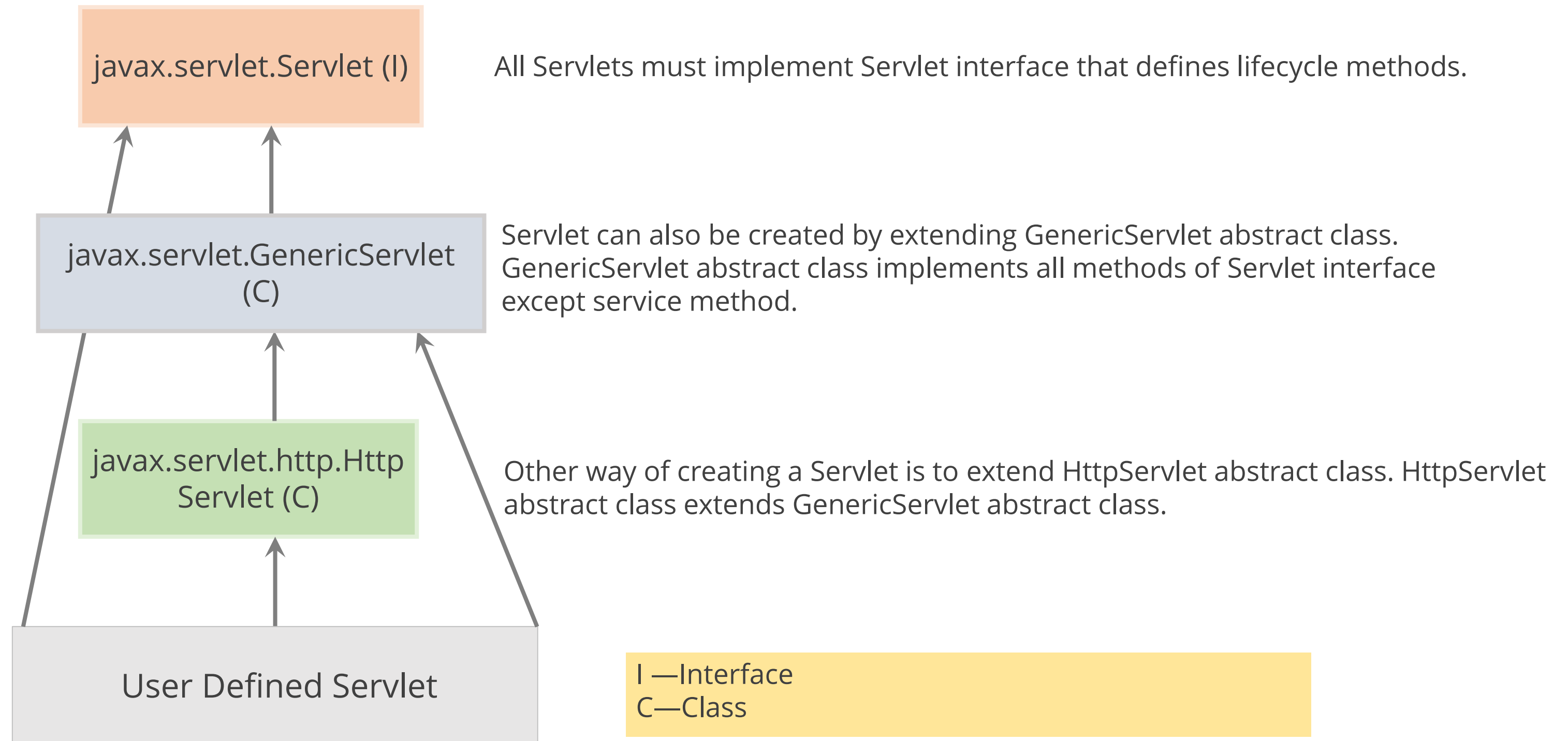
Servlet Interface and Classes



Servlet API Interface

javax.servlet.FilterConfig	Passes information to a filter during initialization
javax.servlet.RequestDispatcher	Sends request and response object to any resource (such as a Servlet, HTML file, or JSP file) on the server
javax.servlet.Servlet	Defines methods that all Servlets must implement
javax.servlet.ServletConfig	A Servlet configuration object used by a Servlet container to pass information to a Servlet during initialization
javax.servlet.ServletContext	Defines a set of methods that a Servlet uses to communicate with its Servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file
javax.servlet.ServletContextAttributeListener	Implementation of this interface receives notifications of changes to the attribute list on the Servlet context of a web application
javax.servlet.ServletContextListener	Implementation of this interface receive notifications about changes to the Servlet context of the web application they are a part of
javax.servlet.ServletRequest	Defines an object to provide client request information to a Servlet
javax.servlet.ServletRequestAttributeListener	A ServletRequestAttributeListener can be implemented by the developer interested in being notified of request attribute changes
javax.servlet.ServletRequestListener	A ServletRequestListener can be implemented by the developer interested in being notified of requests coming in and out of scope in a web component
javax.servlet.ServletResponse	Defines an object to assist a Servlet in sending a response to the client

Servlets API Hierarchy to Create Servlet



Servlet Methods



Servlet interface has 5 methods:

1. `init`
2. `service`
3. `destroy`
4. `getServletInfo`
5. `getServletConfig`

Servlet Methods

init

init
service
destroy
getServletInfo
getServletConfig

init method is called by Servlet container to indicate to a Servlet that it is being placed into service.

It is declared as: `void init (ServletConfig config)`

Servlet Methods

service

init
service
destroy
getServletInfo
getServletConfig

service method is called by the Servlet container to allow the Servlet to respond to a request.

It is declared as:

```
public void service(ServletRequest req, ServletResponse  
res) throws ServletException,  
                java.io.IOException
```

Servlet Methods

destroy

init
service
destroy
getServletInfo
getServletConfig

destroy() indicates that the Servlet is being taken out of service.

It is declared as: `public void destroy()`

Servlet Methods

getServletInfo

init

service

destroy

getServletInfo

getServletConfig

getServletInfo method returns information about the Servlet.

It is declared as: `public java.lang.String getServletInfo()`

Servlet Methods

getServiceConfig

init

service

destroy

getServletInfo

getServletConfig

Returns a ServletConfig object, that contains initialization and start-up parameters for this Servlet.

It is declared as: `public ServletConfig getServletConfig()`

Generic Servlet Abstract Class

- GenericServlet implements Servlet interface.
- It defines Servlet generic and is independent of protocols.
- It doesn't give implementation for service method of Servlet interface.
- If your Servlet extends Generic Class, implementing service method is necessary.

Generic Servlet Abstract Class Methods

<code>public abstract class GenericServlet</code>	Extends <code>java.lang.Object</code> implements <code>Servlet</code> , <code>ServletConfig</code> , and <code>java.io.Serializable</code>
<code>java.lang.String getInitParameter(java.lang.St ring name)</code>	Returns a <code>String</code> containing the value of the named initialization parameter, or returns null if the parameter does not exist
<code>java.util.Enumeration getInitParameterNames()</code>	Returns the names of the Servlet's initialization parameters as an <code>Enumeration</code> of <code>String</code> objects, or returns an empty <code>Enumeration</code> if the Servlet has no initialization parameters
<code>ServletContext getServletContext()</code>	Returns a reference to the <code>ServletContext</code> in which the Servlet is running
<code>java.lang.String getServletName()</code>	Returns the name of this Servlet instance
<code>void init()</code>	A convenient method which can be overridden so that there's no need to call <code>super.init(config)</code>

Generic Servlet Abstract Class Methods

<code>void init(ServletConfig config)</code>	Called by the Servlet container to indicate to a Servlet that the Servlet is being placed into service
<code>void log(java.lang.String msg)</code>	Writes the specified message to a Servlet log file, prepended by the Servlet's name
<code>void log(java.lang.String message, java.lang.Throwable t)</code>	Writes an explanatory message and a stack trace for a given Throwable -throwable exception to the Servlet log file, prepended by the Servlet's name
<code>abstract void service(ServletRequest req, ServletResponse res)</code>	Allows the Servlet to respond to a request.

HttpServlet Abstract Class Methods

It provides an abstract class to be sub-classed to create an HTTP Servlet suitable for a website.

Following is the list of methods:

<code>public abstract class HttpServlet</code>	Extends GenericServlet Implements java.io.Serializable
<code>doGet</code>	Used if the Servlet supports HTTP GET requests
<code>doPost</code>	Used for HTTP POST requests
<code>doPut</code>	Used for HTTP PUT requests
<code>doDelete</code>	Used for HTTP DELETE requests
<code>init and destroy</code>	Used to manage resources that are held for the life of the Servlet
<code>getServletInfo</code>	Used by the Servlet to provide information about itself

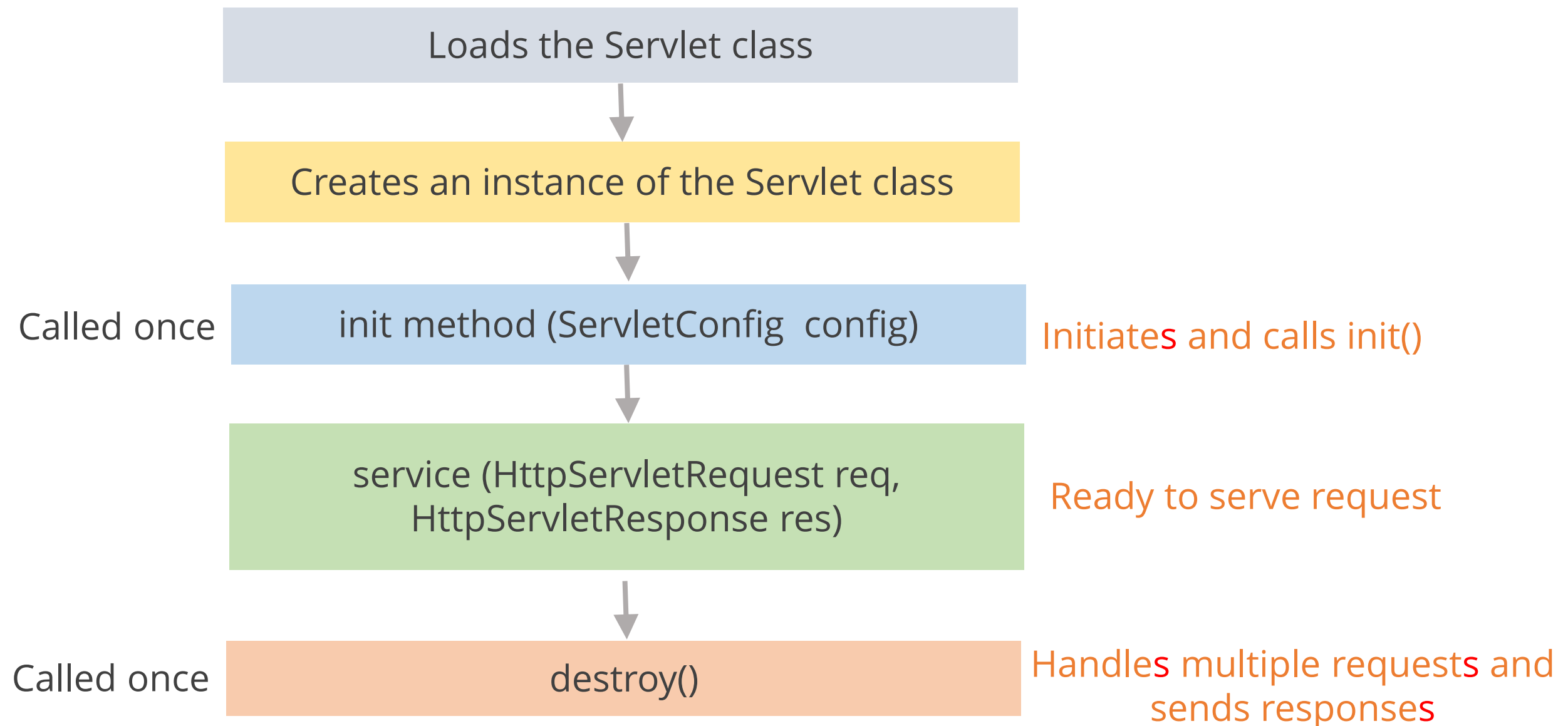
Topic 3—Servlet Lifecycle

Topic 3—Servlet Lifecycle

Servlet Lifecycle

The lifecycle of a Servlet is controlled by the container in which the Servlet has been deployed.

When a request is mapped to a Servlet, the container performs the following steps if an instance of the Servlet does not exist.



Topic 4—Configure and Deploy Servlet

Topic 4—Configure and Deploy Servlet

Configure and Deploy Servlet

1. Create dynamic web project in eclipse
2. Add Server to project
3. Run Server (Apache Tomcat)
4. Create HTML file
5. Click on src and create Servlet
6. Create web.xml file
7. Configure a Servlet Definition
8. Configure a Servlet Mapping
9. Activating the Servlet in Web

Configure and Deploy Servlet

STEPS TO CREATE DYNAMIC WEB PROJECT IN ECLIPSE

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

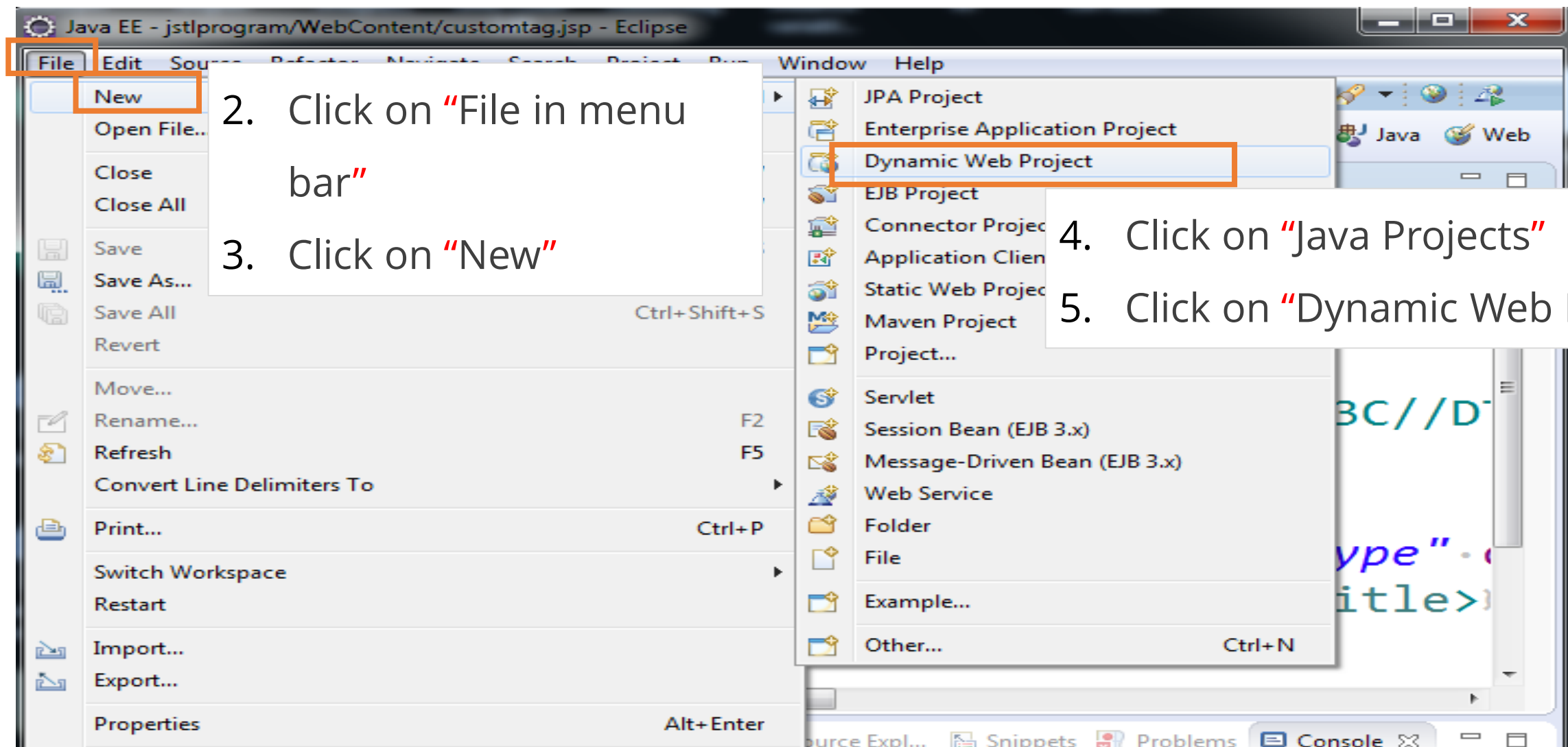
Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

Activating the Servlet in Web

Open eclipse for EE



Configure and Deploy Servlet

STEPS TO CREATE DYNAMIC WEB PROJECT IN ECLIPSE

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

Activating the Servlet in Web

New Dynamic Web Project

Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location
☒ Use default location
Location:

Target runtime

Dynamic web module version

6. Give name to project.

7. Add web.xml by selecting check box.

8. Click on ~~f~~"Finish"

Configure and Deploy Servlet

STEPS TO ADD SERVER TO PROJECT

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

Activating the Servlet in Web

1. Click on “Windows” in menu bar
2. Click on “preference”
3. Click on “Server”
4. Click on “Runtime Environment”
5. Add Server

Configure and Deploy Servlet

RUN SERVER (APACHE TOMCAT)

Create dynamic web project in eclipse
Add Server to project
Run Server (Apache Tomcat)
Create HTML file
Create Java class(Servlet)
Create web.xml file
Configure a Servlet definition
Configure a Servlet Mapping
Activating the Servlet in Web

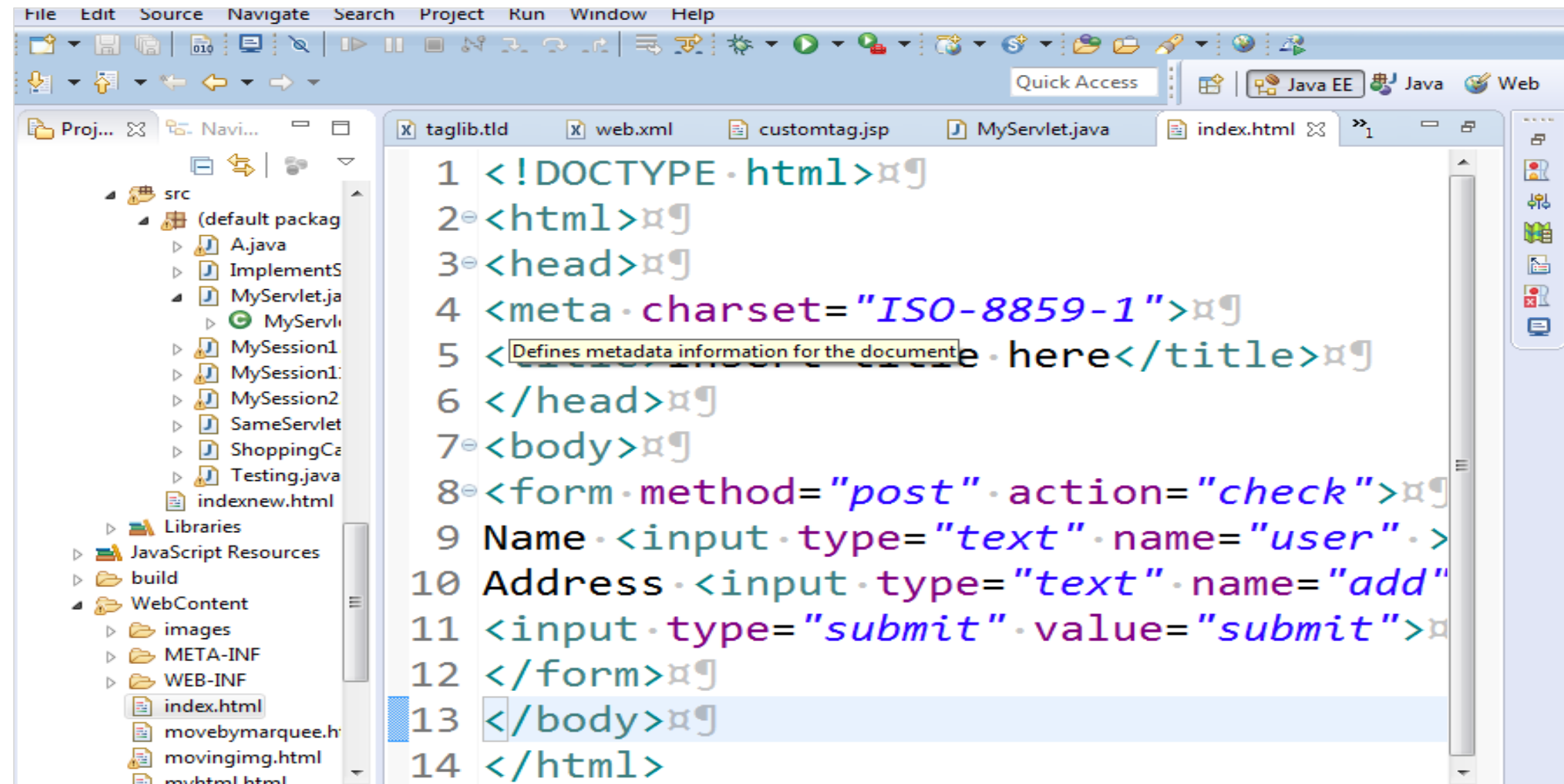
1. Run Apache with eclipse
2. Click on "Window"
3. Click on "show view"
4. Click on "server"
5. click on "new server"
6. Click on "next"
7. Click on "next", the server is now added to your project
8. To start server, right click on "Tomcat server"
9. Click on "start option"



We have already discussed the detailed steps in the installation guide that is available on your LMS for download.

Configure and Deploy Servlet

CREATE HTML FILE



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Defines metadata information for the document here</title>
6 </head>
7 <body>
8 <form method="post" action="check">
9   Name <input type="text" name="user">
10  Address <input type="text" name="add">
11  <input type="submit" value="submit">
12 </form>
13 </body>
14 </html>
```



All web pages like html, css, and jsp should be added in the WebContent folder

Configure and Deploy Servlet

CREATE JAVA CLASS (SERVLET)

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

Activating the Servlet in Web

```
30 import java.io.IOException;
10
11
12 public class MyServlet extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14     //
15     /**
16      * @see HttpServlet#HttpServlet()
17      */
18     public MyServlet() {
19         super();
20         // TODO Auto-generated constructor stub
21     }
22
23     /**
24      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
25      */
26     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27         // TODO Auto-generated method stub
28         response.setContentType("text/html;charset=UTF-8");
29         PrintWriter out = response.getWriter();
30         try {
31             //
32             String user = request.getParameter("user");
33             String add = request.getParameter("add");
34             out.println("<h2>name "+user+"</h2>");
35             out.println("<h2>address "+add+"</h2>");
36             //
37         } finally {
38             out.close();
39         }
40         System.out.println("hi");
41     }
42
43 }
```

Configure and Deploy Servlet

CREATE web.xml FILE

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

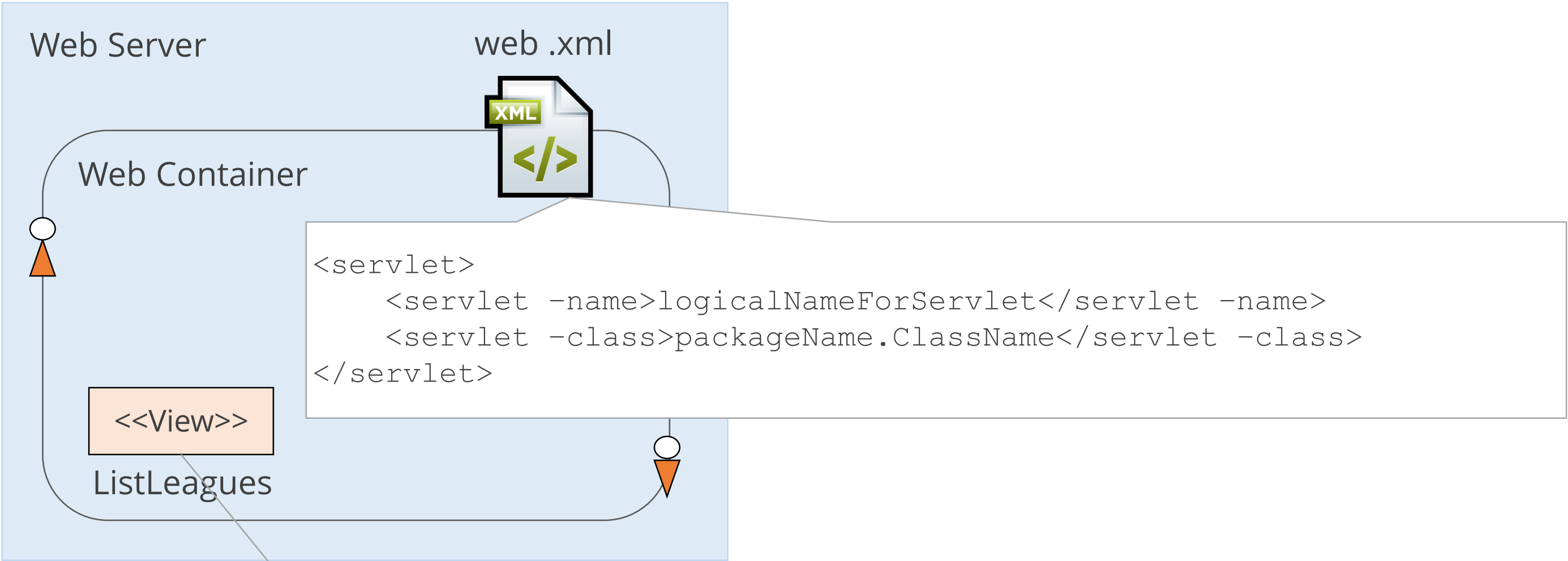
Activating the Servlet in Web

- The web.xml deployment descriptor is used by the web container to configure servlet component. The Servlet definition also specifies the fully qualified class that implements the component.
- The web container creates an instance of each Servlet definition in the deployment descriptor
- There can be multiple Servlet definitions

Configure and Deploy Servlet

CONFIGURE A SERVLET DEFINITION

Create dynamic web project in eclipse
Add Server to project
Run Server (Apache Tomcat)
Create HTML file
Create Java class(Servlet)
Create web.xml file
Configure a Servlet definition
Configure a Servlet Mapping
Activating the Servlet in Web



The web container will create only one, Servlet object for each definition in the deployment descriptor

Configure and Deploy Servlet

CONFIGURE A SERVLET MAPPING

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

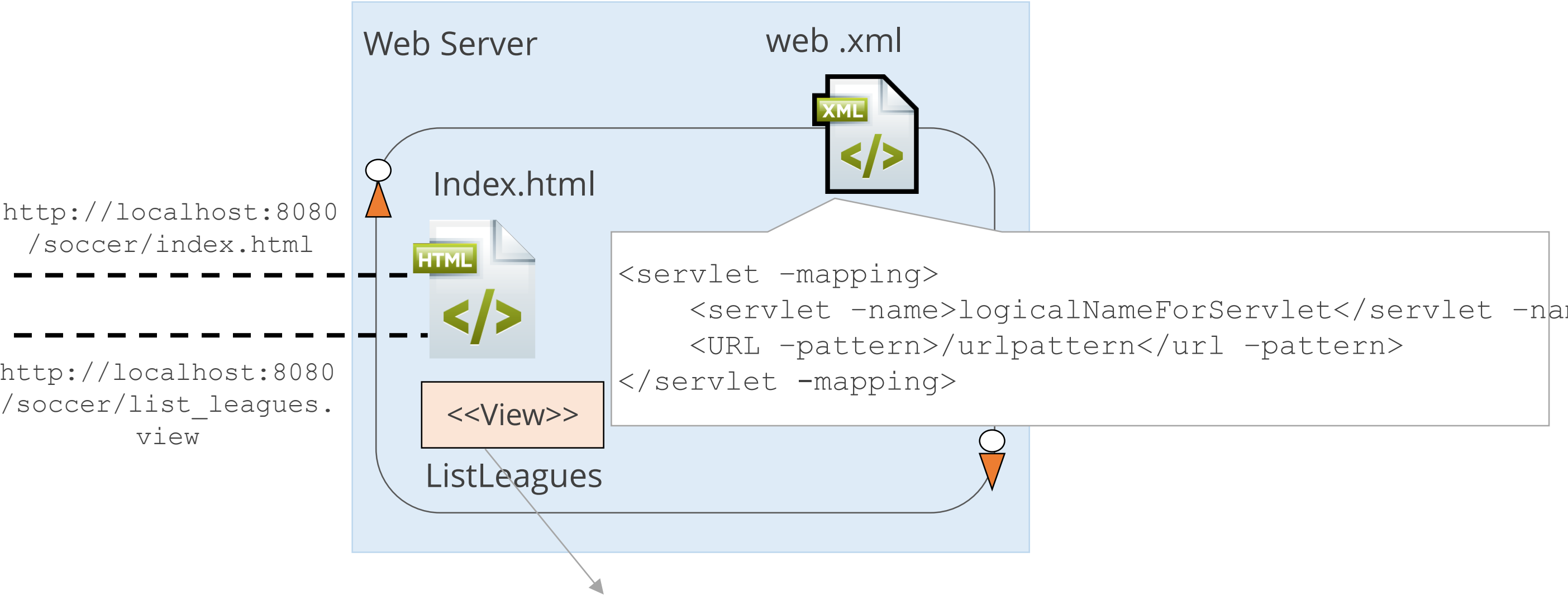
Activating the Servlet in Web

- The deployment descriptor is also used to configure a URL pattern that is used to invoke the Servlet component.
- The web container receives HTTP request for specific URLs, and it matches the URL to either a physical component or a logical mapping to a Servlet component.

Configure and Deploy Servlet

CONFIGURE A SERVLET MAPPING

Create dynamic web project in eclipse
Add Server to project
Run Server (Apache Tomcat)
Create HTML file
Create Java class(Servlet)
Create web.xml file
Configure a Servlet definition
Configure a Servlet Mapping
Activating the Servlet in Web



The web container maps the URL of each request to either a physical web resource (such as an HTML page) or to a logical web resource (such as a Servlet).

Configure and Deploy Servlet

CONFIGURE A SERVLET MAPPING

When URL pattern becomes <http://localhost:8080/projectname/check>, Web container looks in web.xml to find location of its servlet class.

```
<web-app>
<servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>MyServlet</servlet-class>
</servlet>

    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/check</url-pattern>
    </servlet-mapping>

</web-app>
```



Check is URL pattern, HelloServlet is logical name for servlet, and MyServlet is servlet Java class

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

Activating the Servlet in Web

Configure and Deploy Servlet

ACTIVATING THE SERVLET IN WEB

Create dynamic web project in eclipse

Add Server to project

Run Server (Apache Tomcat)

Create HTML file

Create Java class(Servlet)

Create web.xml file

Configure a Servlet definition

Configure a Servlet Mapping

Activating the Servlet in Web

When the user selects the home page by the way of URL <http://localhost:8080/projectname/urlpattren>, the web container responds with the HTML code for the home page.

DEMO—Configure and Deploy Servlet

DEMO—Configure and Deploy Servlet

Advanced Java

Topic 5—ServletsRequest, ServletResponse

- ServletRequest
- ServletRequest Interface
- Servlet Response

ServletRequest



public interface **ServletRequest**

- It defines an object to provide client request information to a Servlet.
- The Servlet container creates a ServletRequest object.
- Servlet Container calls service method by passing ServletRequest object.
- This object provides data including parameter name and values, attributes, and an input stream.

ServletRequest Interface

<code>java.lang.Object getAttribute(java.lang.String name)</code>	Returns the value of the named attribute as an object, or returns null if no attribute of the given name exists.
<code>java.util.Enumeration getAttributeNames()</code>	Returns an Enumeration containing the names of the attributes available to this request.
<code>java.lang.String getCharacterEncoding()</code>	Returns the name of the character encoding used in the body of this request.
<code>int getContentLength()</code>	Returns the length of the request body (in bytes) and is made available by the input stream, or -1 if the length is not known.
<code>java.lang.String getContentType()</code>	Returns the MIME type of the body of the request, or returns null if the type is not known.
<code>ServletInputStream getInputStream()</code>	Retrieves the body of the request as binary data using a ServletInputStream.
<code>java.util.Locale getLocale()</code>	Returns the preferred Locale that the client will accept content, based on the Accept-Language header.
<code>java.util.Enumeration getLocales()</code>	Returns an Enumeration of Locale objects indicating the locales (in decreasing order starting with the preferred locale) that are acceptable to the client based on the Accept-Language header.

ServletRequest Interface

<code>java.lang.String getLocalName()</code>	Returns the host name of the Internet Protocol (IP) interface on which the request was received
<code>int getLocalPort()</code>	Returns the Internet Protocol (IP) port number of the interface on which the request was received
<code>java.lang.String getParameter(java.lang.String name)</code>	Returns the value of a request parameter as a String, or null if the parameter does not exist
<code>java.util.Map getParameterMap()</code>	Returns a java.util.Map of the parameters of this request
<code>java.util.Enumeration getParameterNames()</code>	Returns an Enumeration of String objects containing the names of the parameters contained in this request
<code>java.lang.String[] getParameterValues(java.lang.String name)</code>	Returns an array of String objects containing all of the values the given request parameter has, or returns null if the parameter does not exist
<code>java.lang.String getProtocol()</code>	Returns the name and version of the protocol the request uses in the form protocol/majorVersion.minorVersion, for example, HTTP/1.1
<code>java.io.BufferedReader getReader()</code>	Retrieves the body of the request as character data using a BufferedReader
<code>void setAttribute(java.lang.String name, java.lang.Object o)</code>	Stores an attribute in this request

ServletResponse



public interface **ServletResponse**

- It defines an object to assist a Servlet in sending a response to the client.
- The Servlet Container creates these objects to call Servlet's service method.
- To send binary data in a MIME body response, use the ServletOutputStream returned by `getOutputStream()`.
- To send character data, use the PrintWriter object returned by `getWriter()`. To mix binary and text data, for example, to create a multipart response, use a ServletOutputStream and manage the character sections manually.

Advanced Java

Topic 5—ServletConfig, ServletContext

- ServletConfig, ServletContext
- ServletConfig Methods

ServletConfig, ServletContext

ServletConfig:

1. The config object is created by the web container based on the initialization parameters specified in the deployment descriptor
2. One ServletConfig per Servlet

ServletContext:

1. To share application-specific data across independent web components
2. One ServletContext per application
3. The context object is accessible to all Servlets in web application
4. A ServletContext object is the runtime representation of the web application

ServletConfig Methods

java.lang.String getInitParameter(java.lang.String name)	Returns a String containing the value of the named initialization parameter, or returns null if the parameter does not exist
java.util.Enumeration getInitParameterNames()	Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or returns an empty Enumeration if the servlet has no initialization parameters
ServletContext getServletContext()	Returns a reference to the ServletContext in which the caller is executing.
java.lang.String getServletName()	Returns the name of this servlet instance
public interface ServletContext	Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file
java.lang.Object getAttribute(java.lang.String name)	Returns the servlet container attribute with the given name, or returns null if there is no attribute by that name

ServletConfig Methods

java.util.Enumeration getAttributeNames()	Returns an Enumeration containing the attribute names available within this servlet context
ServletContext getContext(java.lang.String uripath)	Returns a ServletContext object that corresponds to a specified URL on the server
java.lang.String getInitParameter(java.lang.String name)	Returns a String containing the value of the named context-wide initialization parameter, or returns null if the parameter does not exist
java.util.Enumeration getInitParameterNames()	Returns the names of the context's initialization parameters as an Enumeration of String objects, or returns an empty Enumeration if the context has no initialization parameters

Advanced Java

DEMO—ServletConfig and ServletContext

Advanced Java

Topic 6—Servlet Scopes and Attributes

- Servlet Attributes
- Attribute Specific Methods

Servlet Attributes

Attributes allow data to be stored and used in inter servlet communication.

The methods associated with attribute manipulation are:

Attribute Scope	
Attribute	Scope of data
ServletRequest	Duration of the request
HttpSession	While the client is active
ServletContext	The life of the web application



We will learn about Session scope in lesson 3 and `management.pageContext` is described in JSP (lesson 4)

Attribute Specific Methods

1. `public void setAttribute (String name, Object object)`: This method is used to set the given object in the application scope.
2. `public Object getAttribute(String name)`: This method is used to return the attribute for the specified name.
3. `public Enumeration getInitParameterNames()`: This method is used to return the names of the context's initialization parameters as an Enumeration of String objects.
4. `public void removeAttribute(String name)`: This method is used to remove the attribute with the given name from the servlet context.

Advanced Java

Topic 7—Servlet Collaboration

- What is Servlet Collaboration?
- Ways of Servlet Collaboration
- Forwarding and Redirecting
- forward() method
- include() method
- RequestDispatcher Interface

What is Servlet Collaboration?

The Servlet collaboration involves sharing of information among the servlets.

Collaborating servlets is passing the common information that is to be shared directly by one servlet to another servlet of html or jsp through various invocations of methods.



Servlet should know about the other servlets with which it is collaborated.

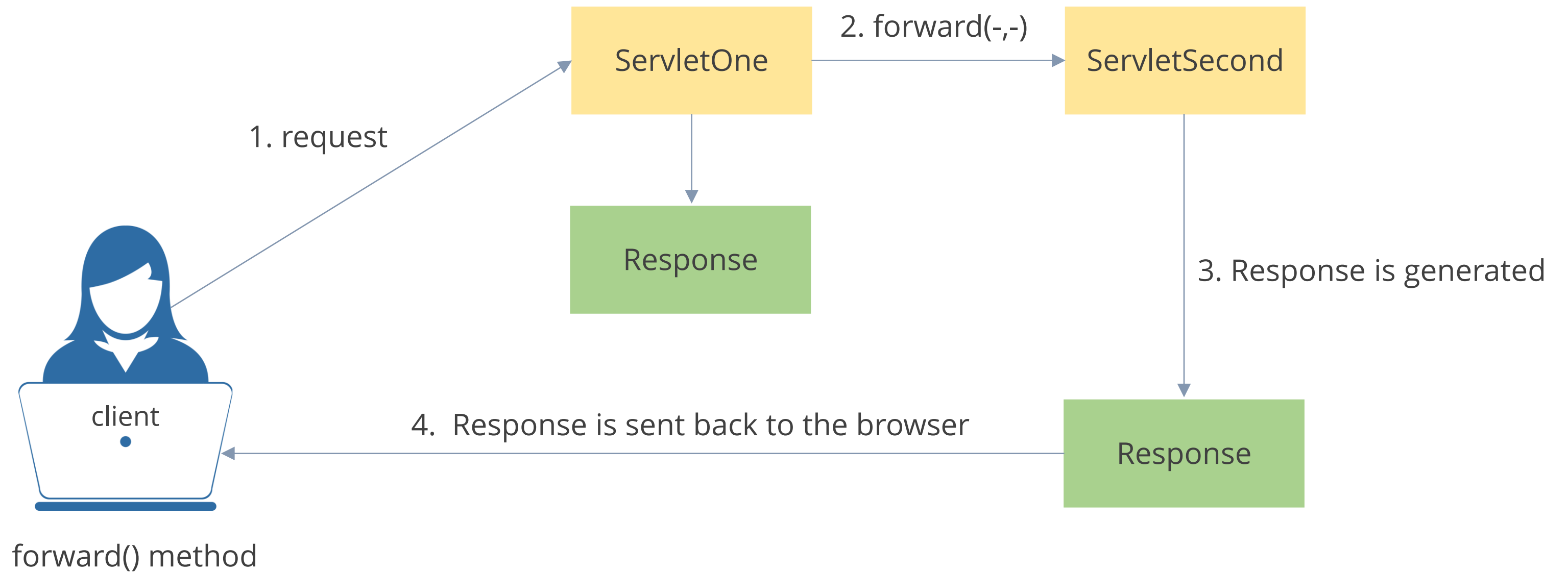
Ways of Servlet Collaboration

- Using RequestDispatchers *include()* and *forward()* method;
- Using HttpServletResponse *sendRedirect()* method;
- Using ServletContext *setAttribute()* and *getAttribute()* methods;

Forwarding and Redirecting

- RequestDispatcher interface is used to forward request and response objects to another Servlet
- sendRedirect method is used for Redirecting request and response object
- HttpServletResponse has sendRedirect method: `response.sendRedirect("url");`
- HttpServletRequest has getRequestDispatcher method that returns RequestDispatcher object
- There are two methods, forward and include

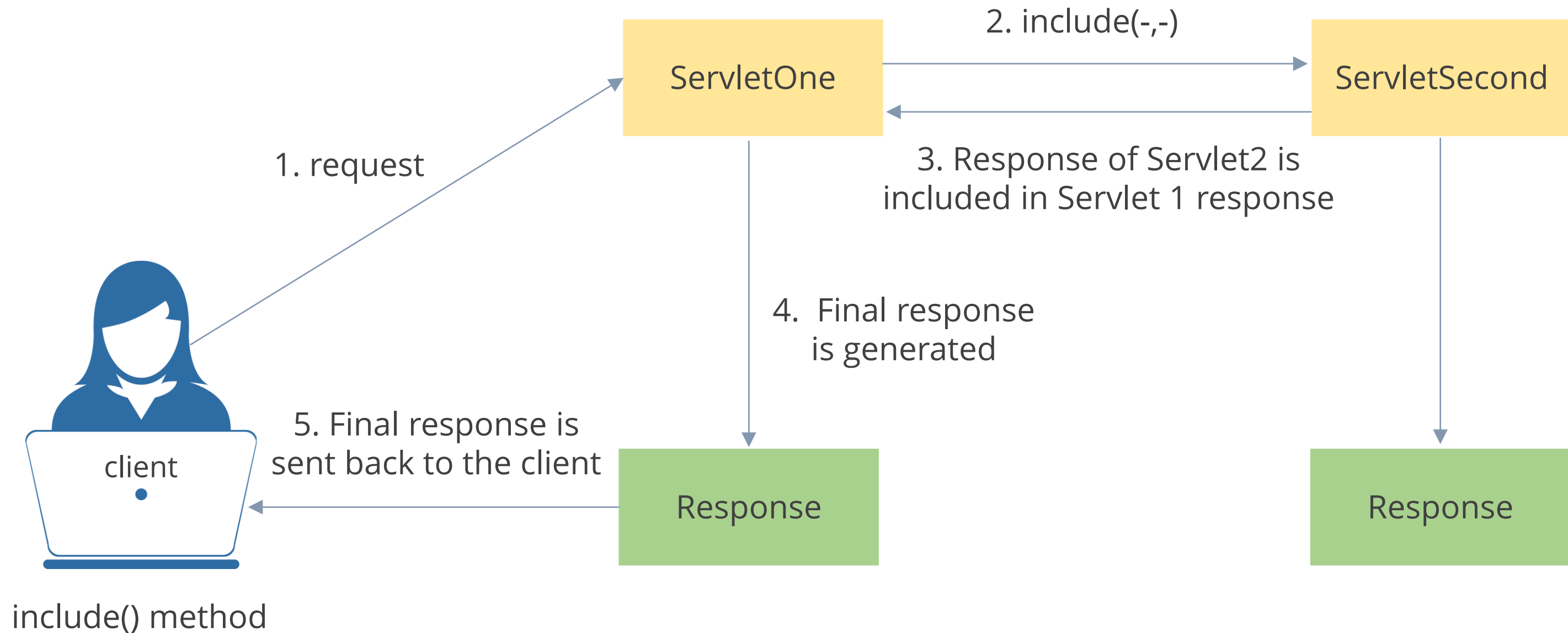
forward() method



forward() method: Syntax

```
RequestDispatcher rd=  
request.getRequestDispatcher("url_pattern_for_servlet or html_file_name");  
rd.forward(request, response)
```

include() method

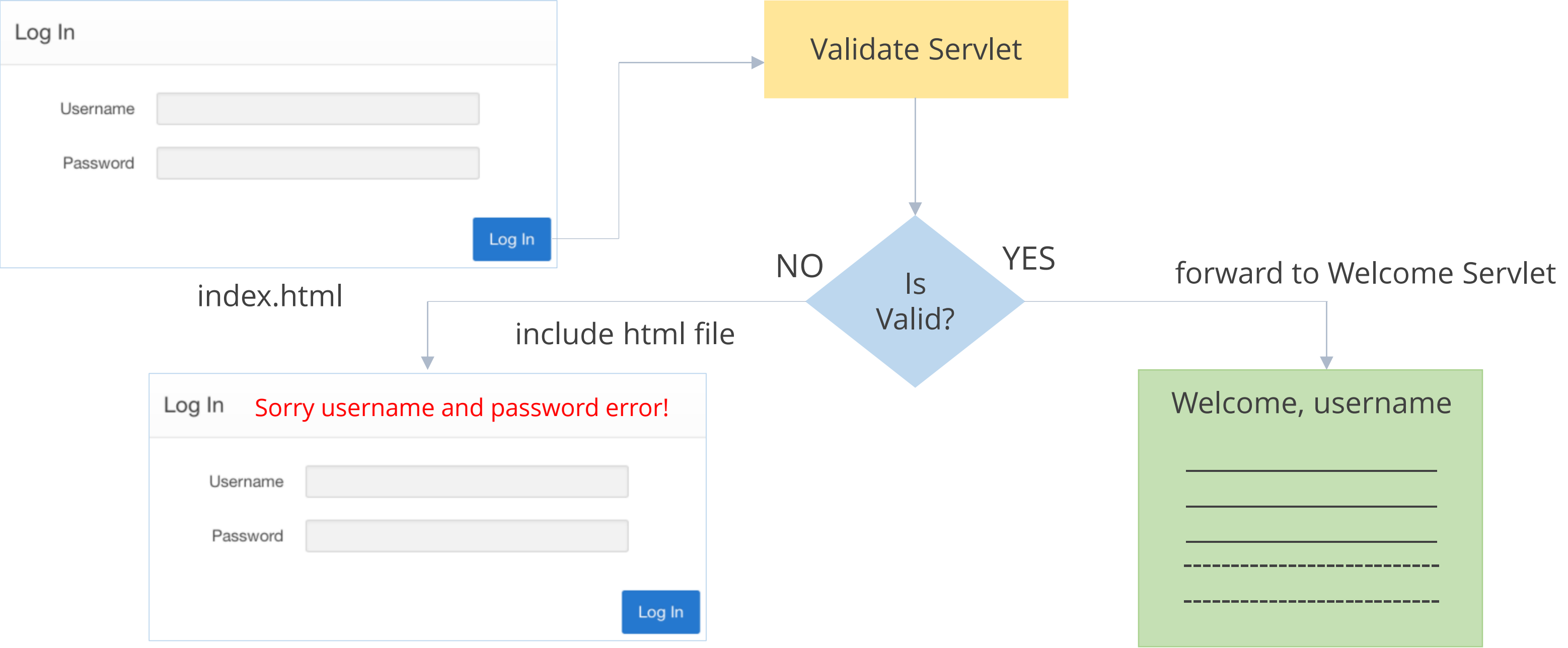


include() method: Syntax



```
RequestDispatcher rd= request.getRequestDispatcher("url_pattern_for_servlet or html_file_name");  
rd.include(request,response);
```


RequestDispatcher Interface



Key Takeaways



- ✓ Servlet is a Java program that can perform dynamic operations and send to web server. The web server then sends this response to web client.
- ✓ Web container is built on top of the Java SE platform and implements the Servlet API and the services required to process HTTP (and other Transmission Control/Internet Protocol [TCP/IP]) requests.
- ✓ The lifecycle of a Servlet is controlled by the web container in which the Servlet has been deployed.
- ✓ Servlet API contains a number of classes and interfaces that defines the contracts between a servlet class and the runtime environment provided for an instance by a conforming servlet container.
- ✓ Collaborating servlets is passing the common information that is to be shared directly by one servlet to another servlet of HTML or JSP through various invocations of the methods.



**QUIZ
1**

Which package contain Servlet interface?

- a. javax.Servlet
- b. javax.Servlet.http
- c. None Of the Above
- d. Both
- e. java.Servlet



The correct answer is **b**

**QUIZ
2**

Which method is called only once?

- a. init()
- b. service()
- c. destroy()
- d. getServletInfo()
- e. getServlet()



The correct answer is **b**



Thank You