Developing Dynamic Web Pages Using Servlets

You Already Know

Course(s):

Java Certification Training Course

Recap

- Explain servlet life cycle
  - Servlet
  - Servlet life cycle

- Configure and deploy servlet
  - Configuring of servlet with Eclipse
  - Deploying servlet

- Explain servlet API, interfaces, and methods
  - Servlet API
  - Servlet classes and interfaces
  - Servlet methods

Recap

- Manage a session
  - Session management in servlets

- Explain listeners in Java EE
  - Listeners in Java EE

- Explain filters in Java EE
  - Filters in Java EE

A Day in the Life of a Full Stack Developer

We have met Joe before. He is working as a Full Stack Developer in Abq Inc. He has upskilled himself. Due to his excellence, a task related to an e-commerce website has been assigned to him. This website is new, and Joe has to build it from scratch.

Joe has to develop a servlet-based login page for the website. He has to write a program such that on successful login, a dashboard will appear where the logout link will be provided. And on incorrect login, an error message will be displayed.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

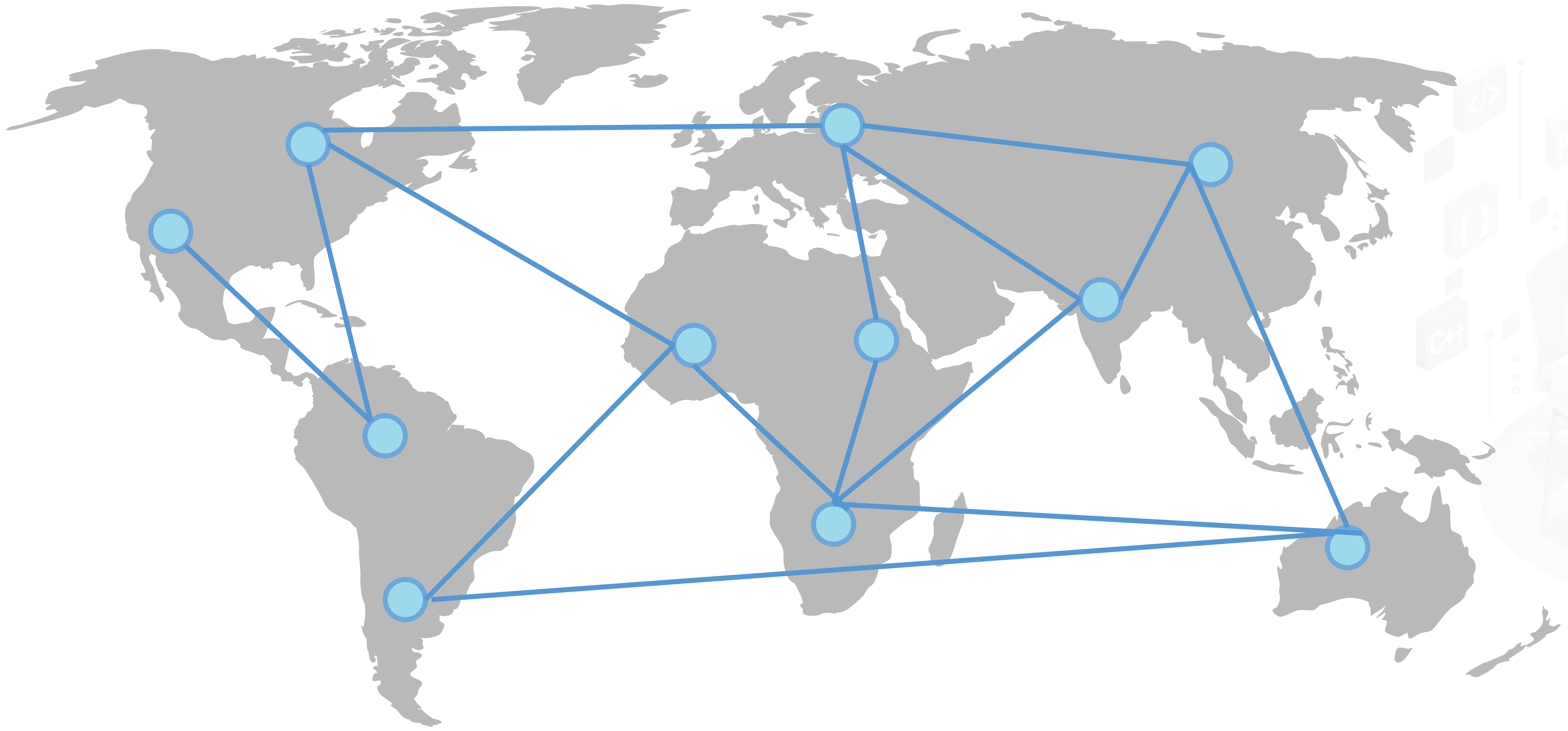
- Explain web technology
- Define servlets and servlet architecture
- Configure servlets and deploy them on Eclipse IDE
- Explain generic servlets, servlet classes, and interfaces
- Design servlet filters
- Implement session tracking and session tracking techniques

FULL STACK

Web Technology

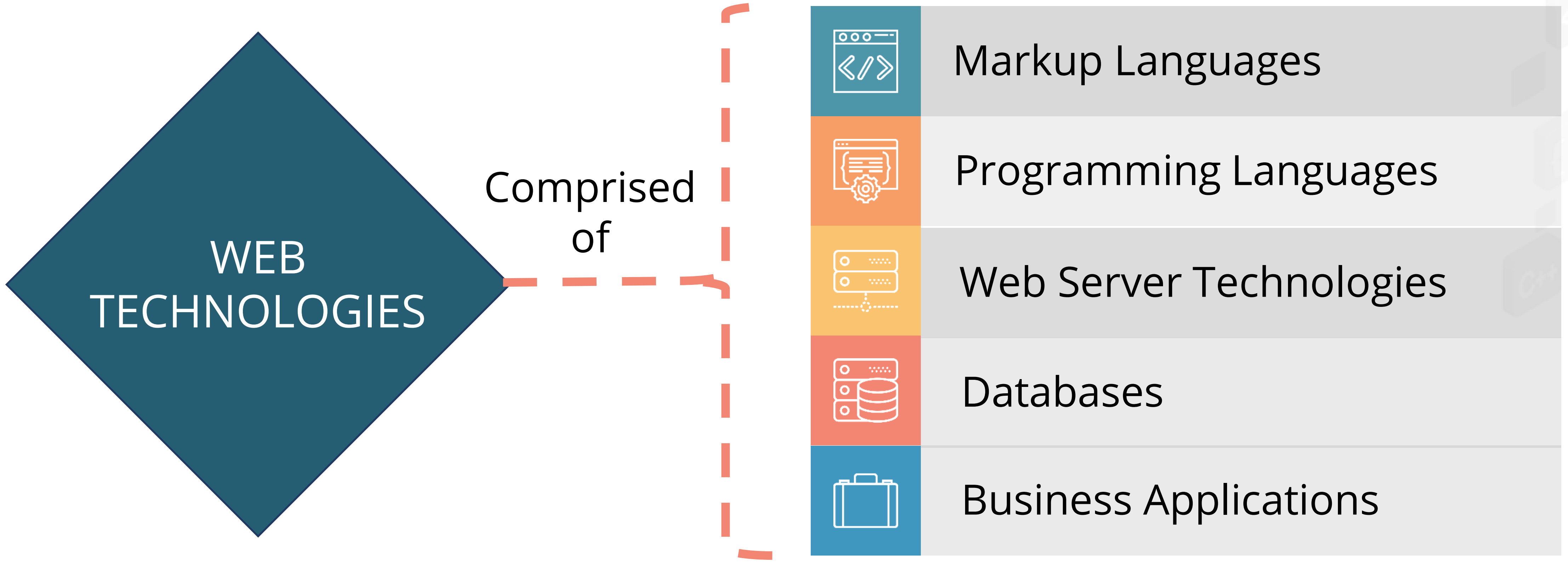
Internet: An Overview

Internet is a global system of interconnected computer networks.
It uses Internet Protocol Suite (TCP/IP).



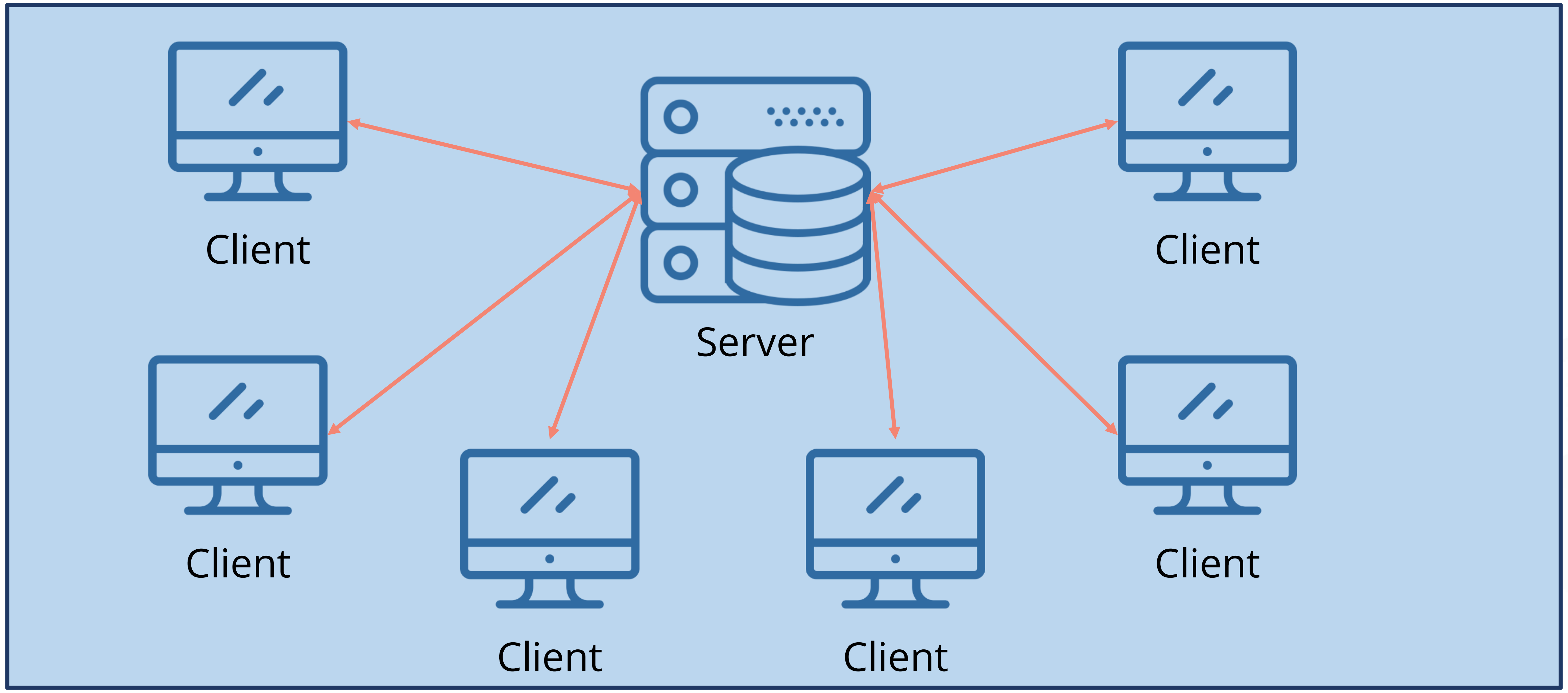
Web Technologies

Web technologies are mechanisms that enable computers to communicate with each other over a network.



Client-Server Architecture

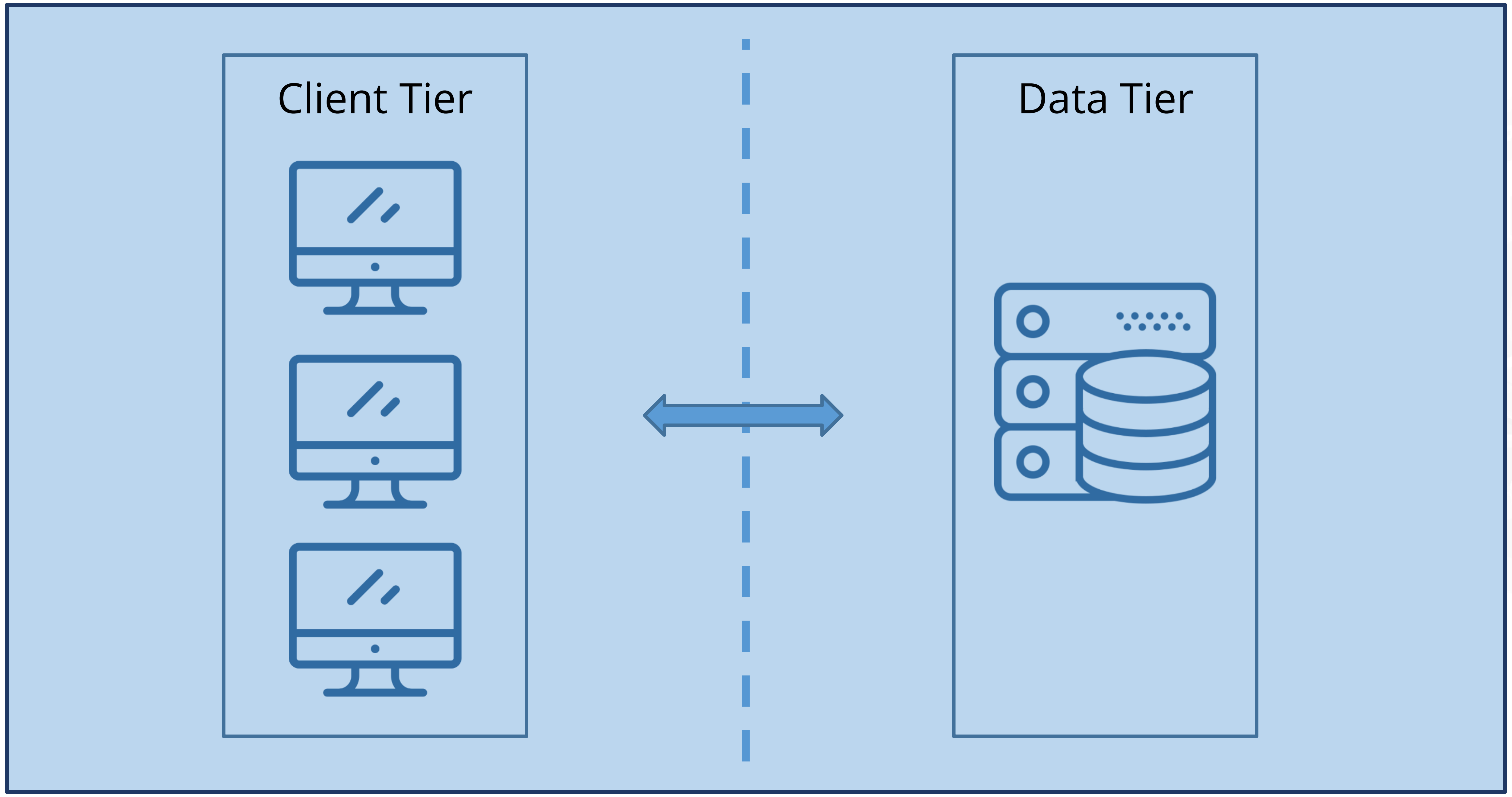
It is a computing model where the server hosts, delivers, and manages most of the services and resources that will be consumed by the client.



One or more **client** computers connected to a central **server** over a **network**

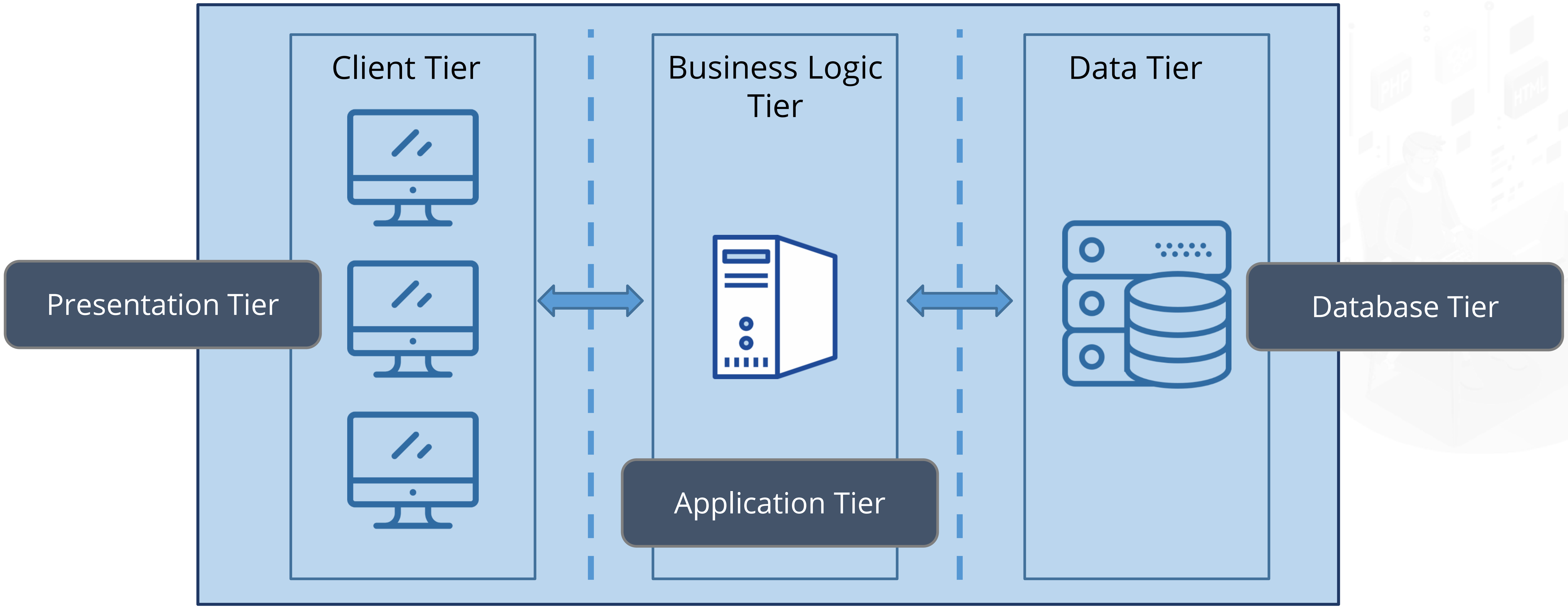
Client-Server Architecture: Two-Tier

In a two-tier architecture, the presentation tier runs on the client, and data gets stored on the server.

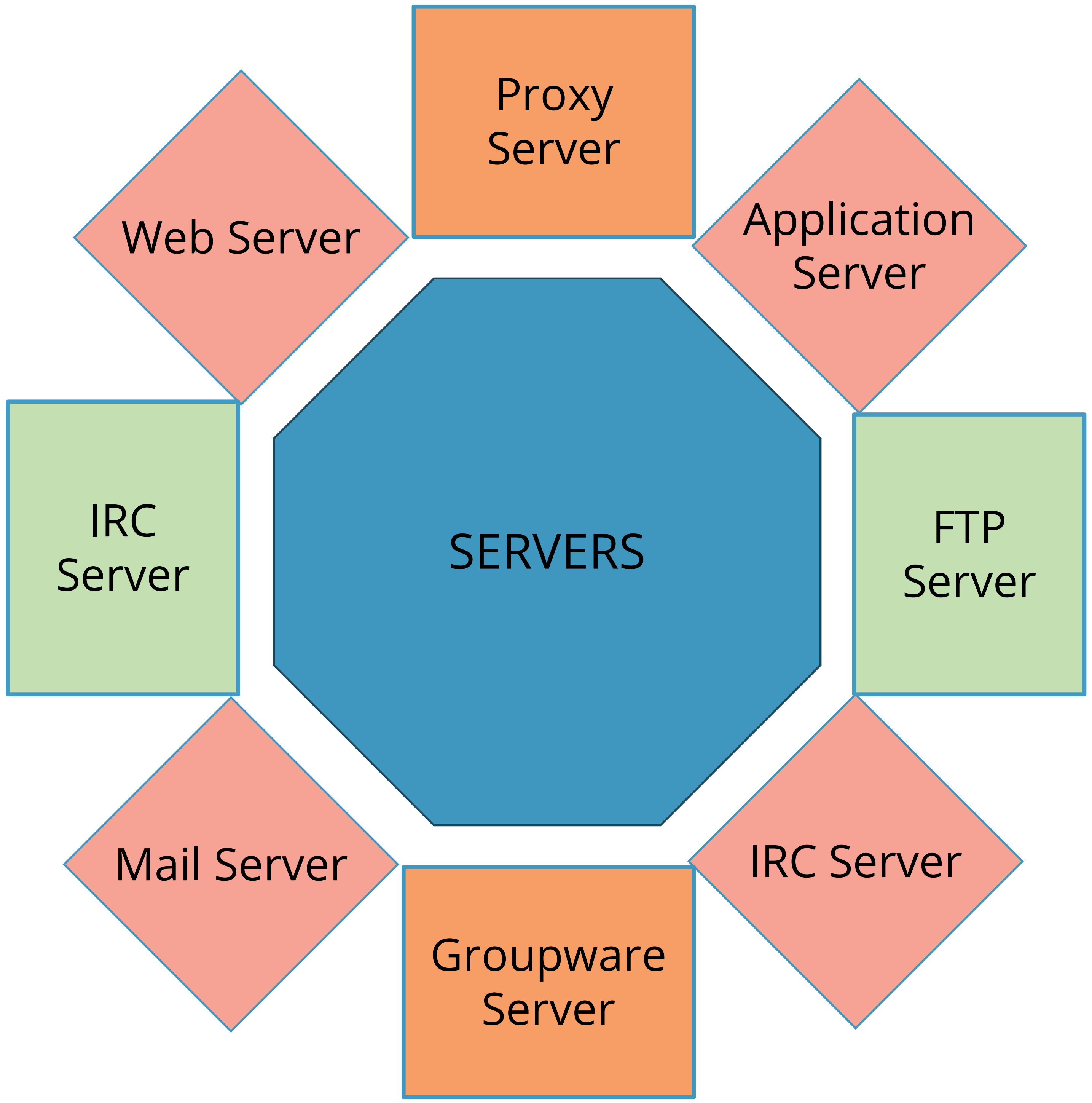


Client-Server Architecture: Three-Tier

In a three-tier architecture, the business logic and the presentation tier are separated. The presentation tier, application tier, and database tier constitute a three-tier architecture.

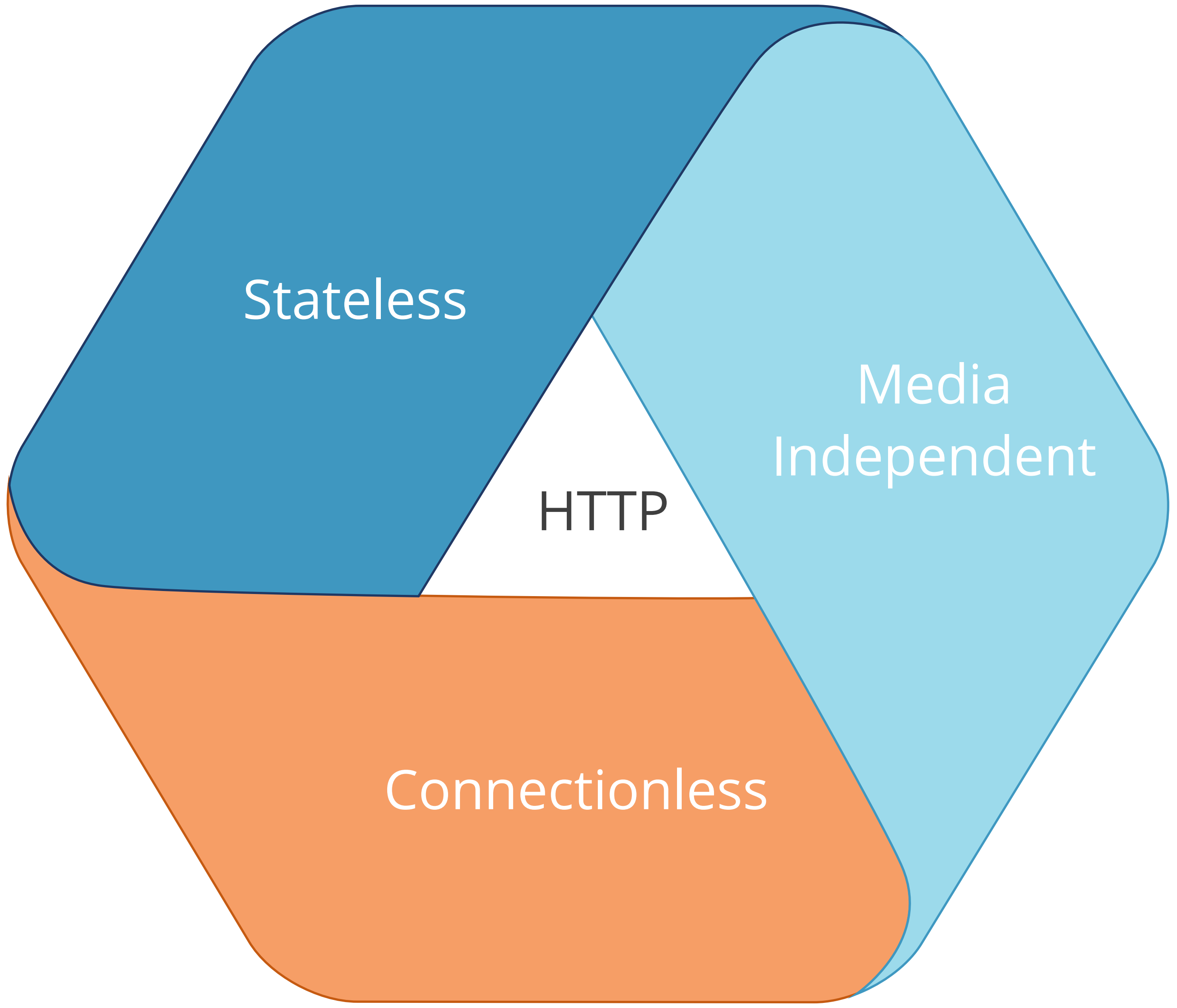


Server Types



Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is an application-level protocol for collaborative and distributed systems.



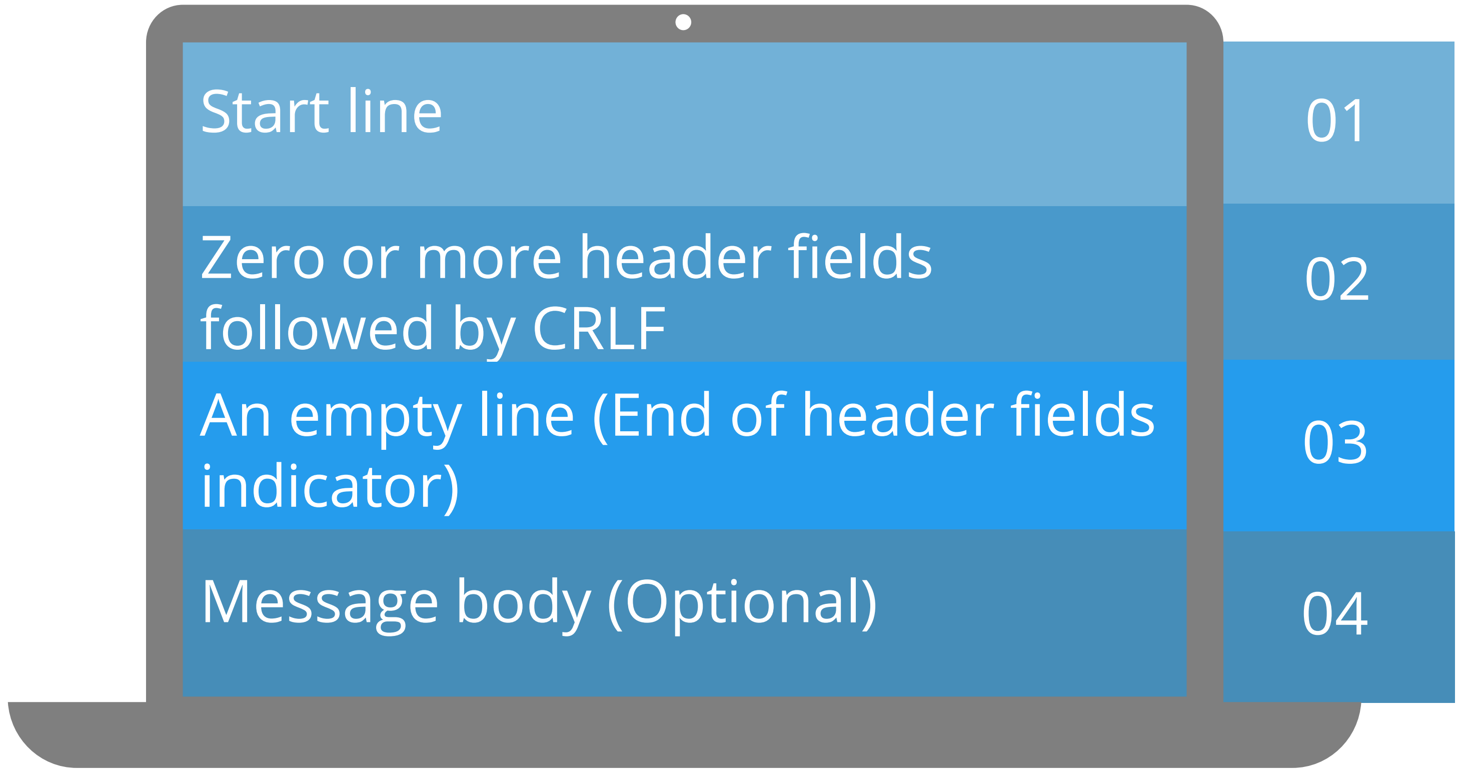
Hypertext Transfer Protocol (HTTP)

The client and server use ASCII messages to communicate. The client sends a **request** to the server, and the server sends back a **response**.

HTTP Message

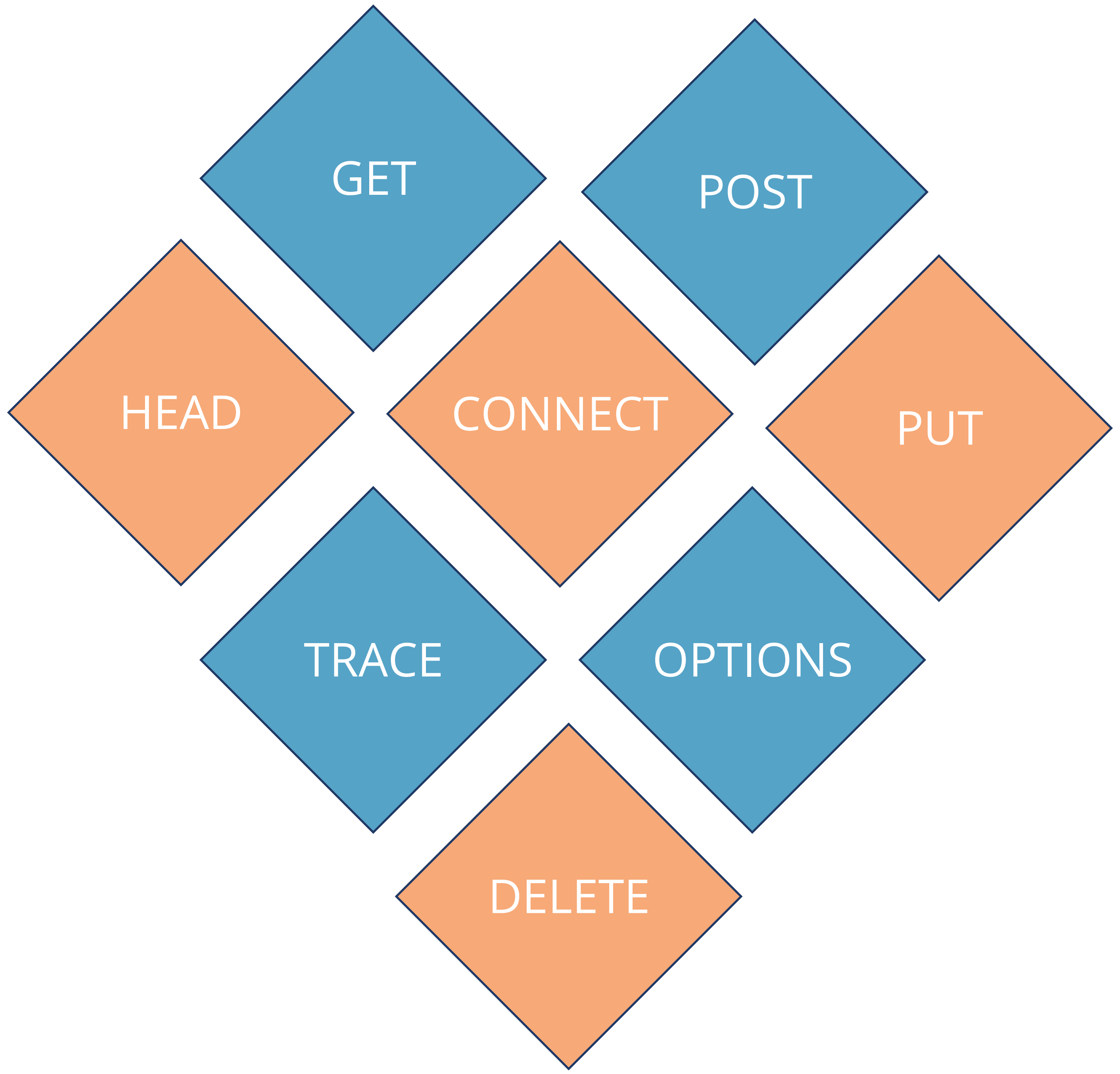
HTTP-message = <Request> | <Response> ; HTTP/1.1 messages

HTTP Message Format


Start line
01
Zero or more header fields followed by CRLF
02
An empty line (End of header fields indicator)
03
Message body (Optional)
04

HTTP Methods

HTTP method names are case sensitive. They must be used in uppercase only.



Difference between HTTP GET and POST

GET	POST
Request parameters are passed in the URL string	Request parameters are passed in the request body
Requests are usually used for viewing a record	Requests are mainly used for updating a record
Requests can be cached	Requests are never cached
Requests remain in browser history	Requests do not remain in browser history
Requests can be bookmarked	Requests cannot be bookmarked
Requests can contain only ASCII characters	Requests have no restrictions on data type
Requests have length restrictions	Requests have no restrictions on data length

Example of Hypertext Transfer Protocol (HTTP)

In Java, HTTP requests can be sent using the **HttpURLConnection** class.

```
URL url = new URL("https://somedomain.com");
HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("User-Agent", "Java Browser");
BufferedReader rdr = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
String line = null;
StringBuilder sb = new StringBuilder("");
while ((line = rdr.readLine()) != null) {
   sb.append(line);
}

rdr.close();
System.out.println(sb.toString());
```

Differences between GET and POST

Duration: 30 min.

Problem Statement:
Write a program to demonstrate the differences between GET and POST.

Assisted Practice: Guidelines

Steps to demonstrate differences between GET and POST:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate the differences between GET and POST methods.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push the code to your GitHub repository.

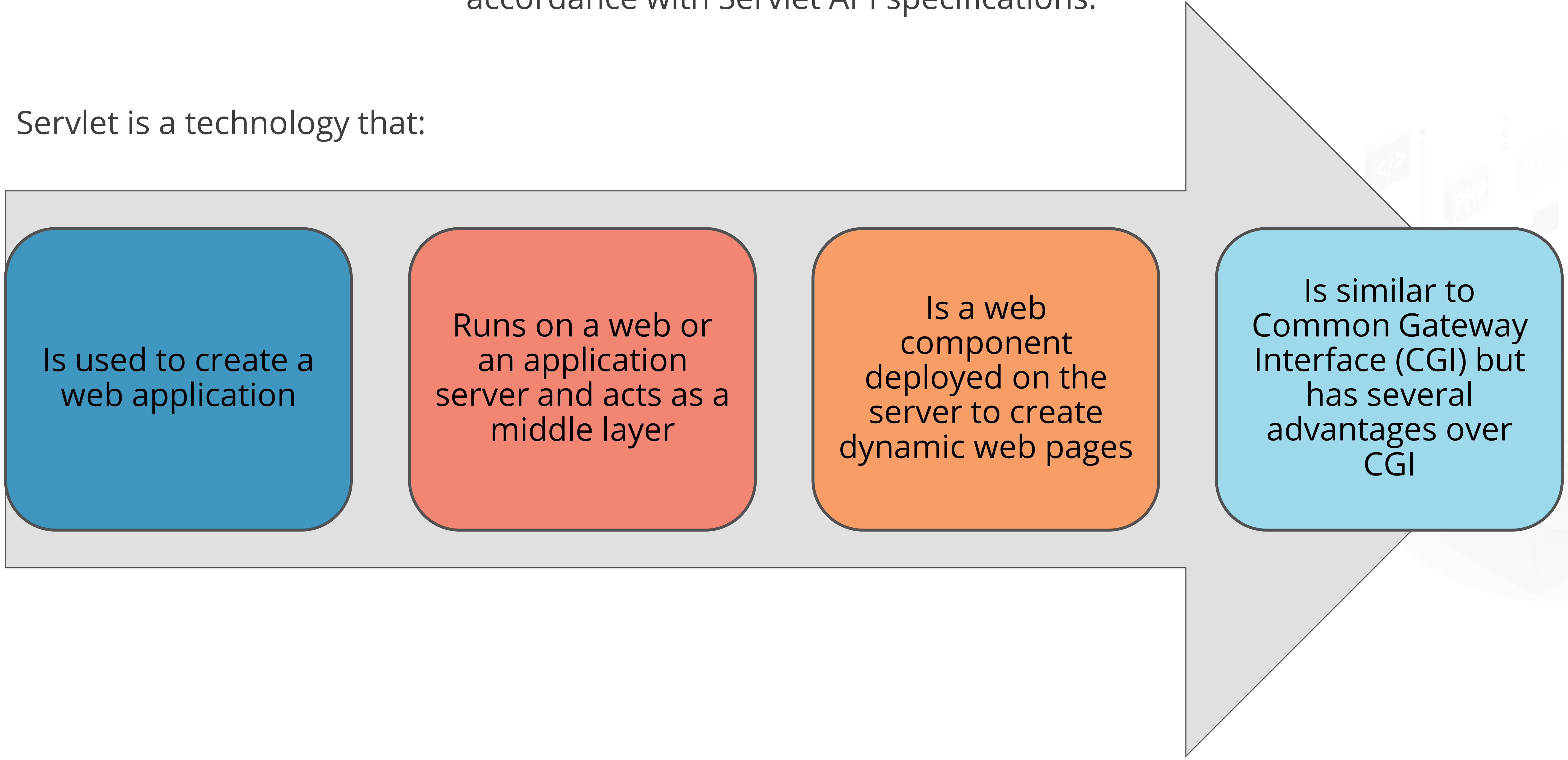
FULL STACK

Overview of Servlets

Servlet

In Java, a servlet is a server-side web component that is written in accordance with Servlet API specifications.

Servlet is a technology that:


Is used to create a web application
Runs on a web or an application server and acts as a middle layer
Is a web component deployed on the server to create dynamic web pages
Is similar to Common Gateway Interface (CGI) but has several advantages over CGI

Common Gateway Interface (CGI)

CGI is a set of rules for running programs and scripts on a web server

CGI is a standard interface that can be used on multiple hardware platforms

CGI specifies how information is communicated and transmitted between web browsers and web server

CGI programs can be written in programming languages such as Java, Perl, and C++

Difference between CGI and Servlets

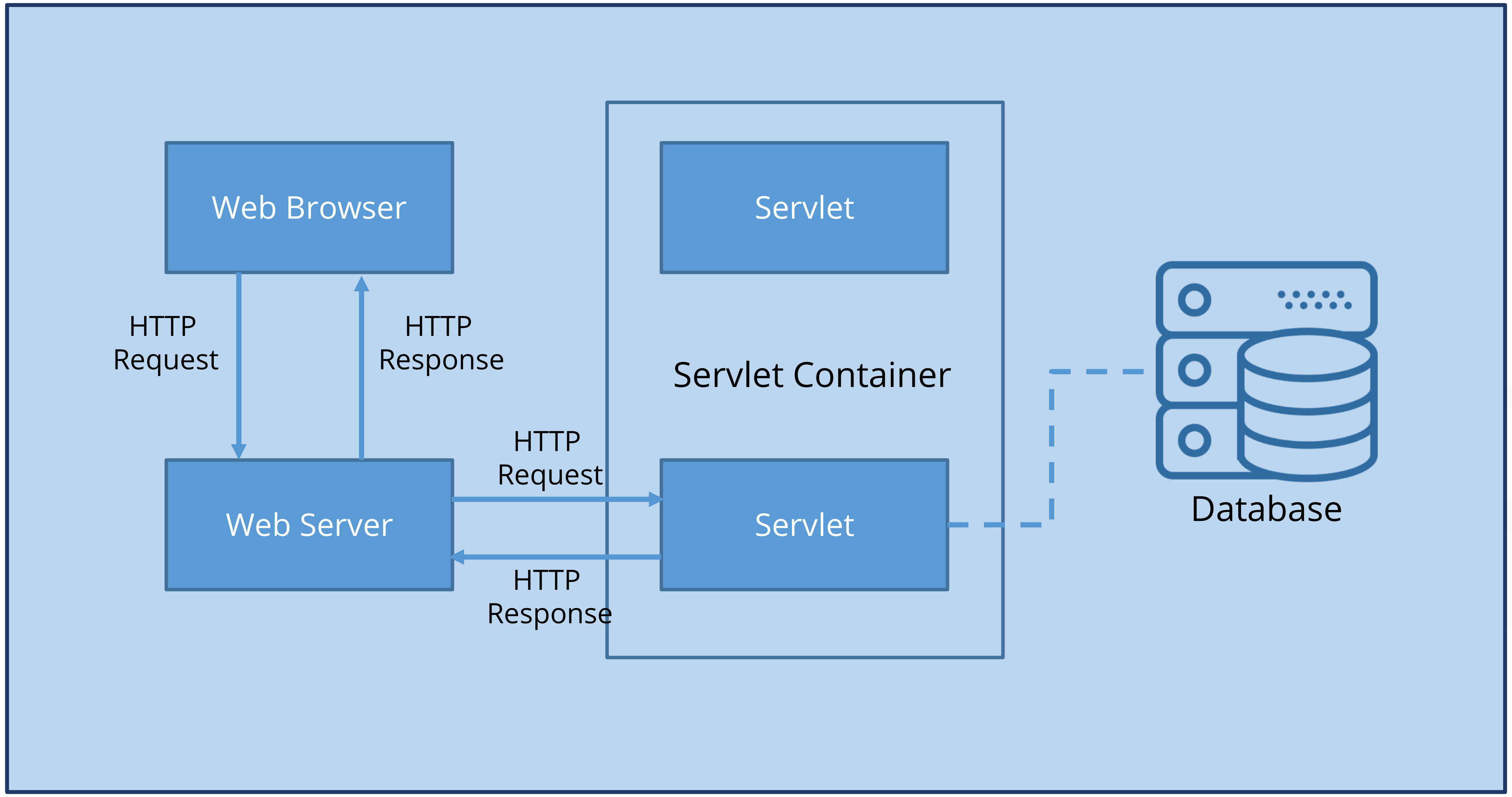
CGI	Servlet
Not portable	Portable
Data cannot be shared	Data can be shared
Cannot directly link to the web server	Can directly link to the web server
Does not allow session tracking and caching	Allows session tracking and caching
No automatic parsing and decoding of HTML form data	Automatic parsing and decoding of HTML form data
HTTP headers cannot be read or set	HTTP headers can be read and set
Cookies cannot be handled	Cookies can be handled

FULL STACK

Servlet Architecture

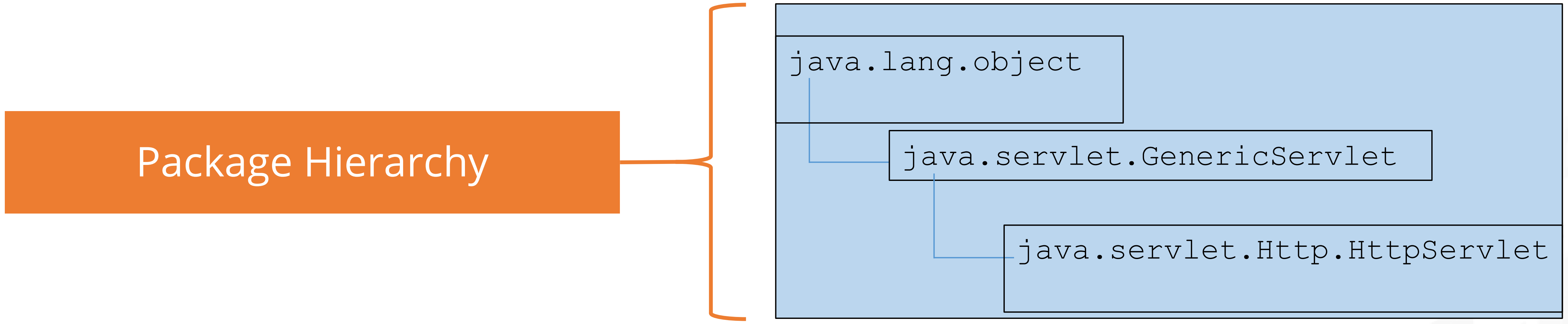
Servlet Architecture

The main components of servlet architecture are the web browser, the web server, the servlet container, and the database.



Java Servlet API: javax.servlet Package

The core of Servlet API is **javax.servlet** package that has all the classes and interfaces required to create a standard and protocol-independent servlet.



Javax.servlet

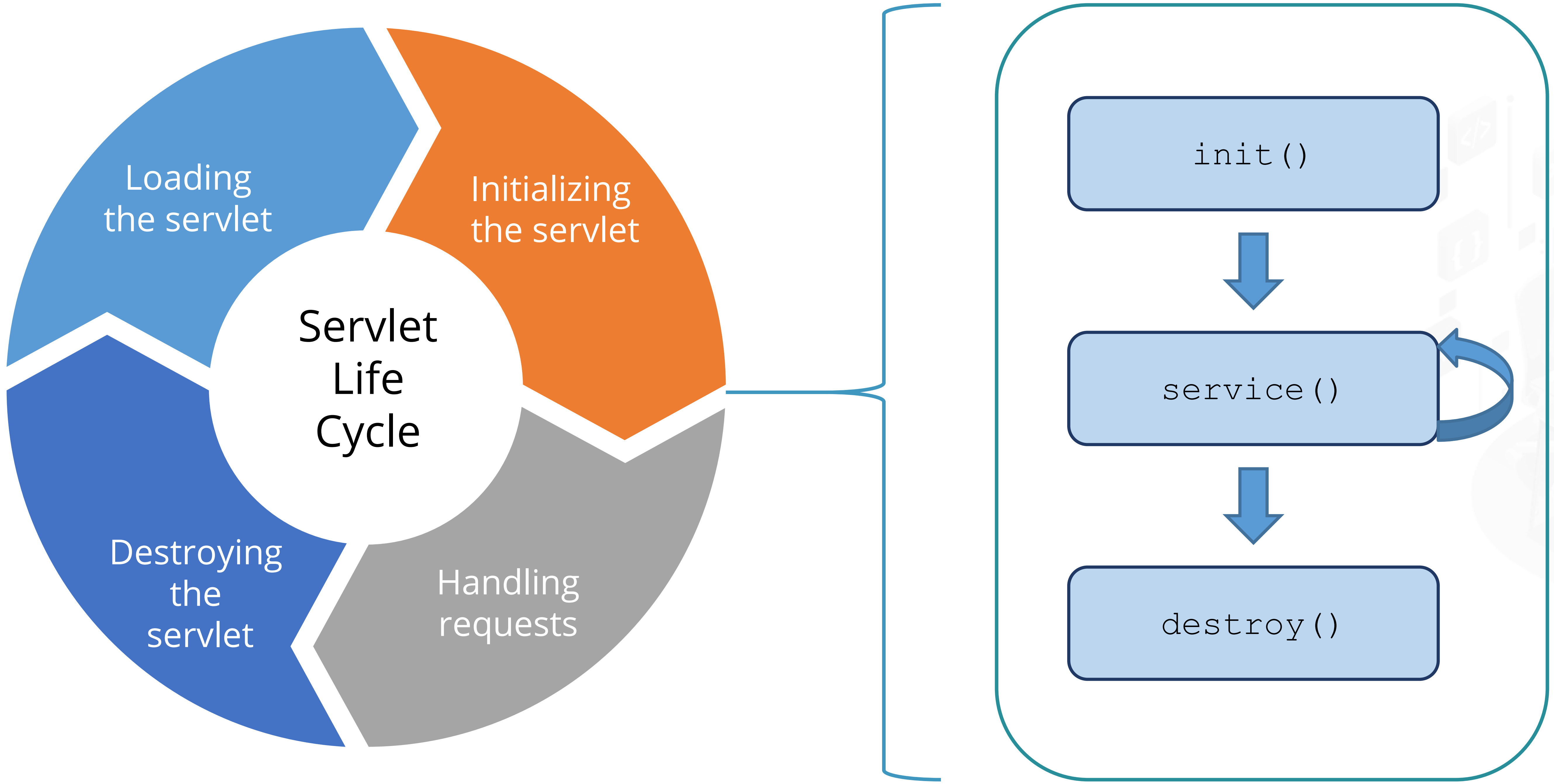
Contains classes and interfaces used by the servlet or web container and is not protocol specific

Javax.servlet.Http.HttpServlet

Contains classes and interfaces specific to creating HTTP servlets

Servlet Life Cycle

Servlet life cycle is managed by the servlet container.



Servlet Types

There are two types of servlets: **Generic** and **HTTP.**

Generic Servlet	HTTP Servlet
Protocol independent	Protocol dependent
Extends the object class and implements Servlet, ServletConfig, and Serializable interfaces	Extends the GenericServlet class and implements the Serializable interface
service() is an abstract method	service() is not an abstract method
Not commonly used	Commonly used
Package – javax.servlet	Package – javax.servlet.Http

Configuring a Servlet with Eclipse

Duration: 30 min.

Problem Statement:
Configure a servlet in Eclipse IDE.

Assisted Practice: Guidelines

Steps to configure Servlet in Eclipse IDE:

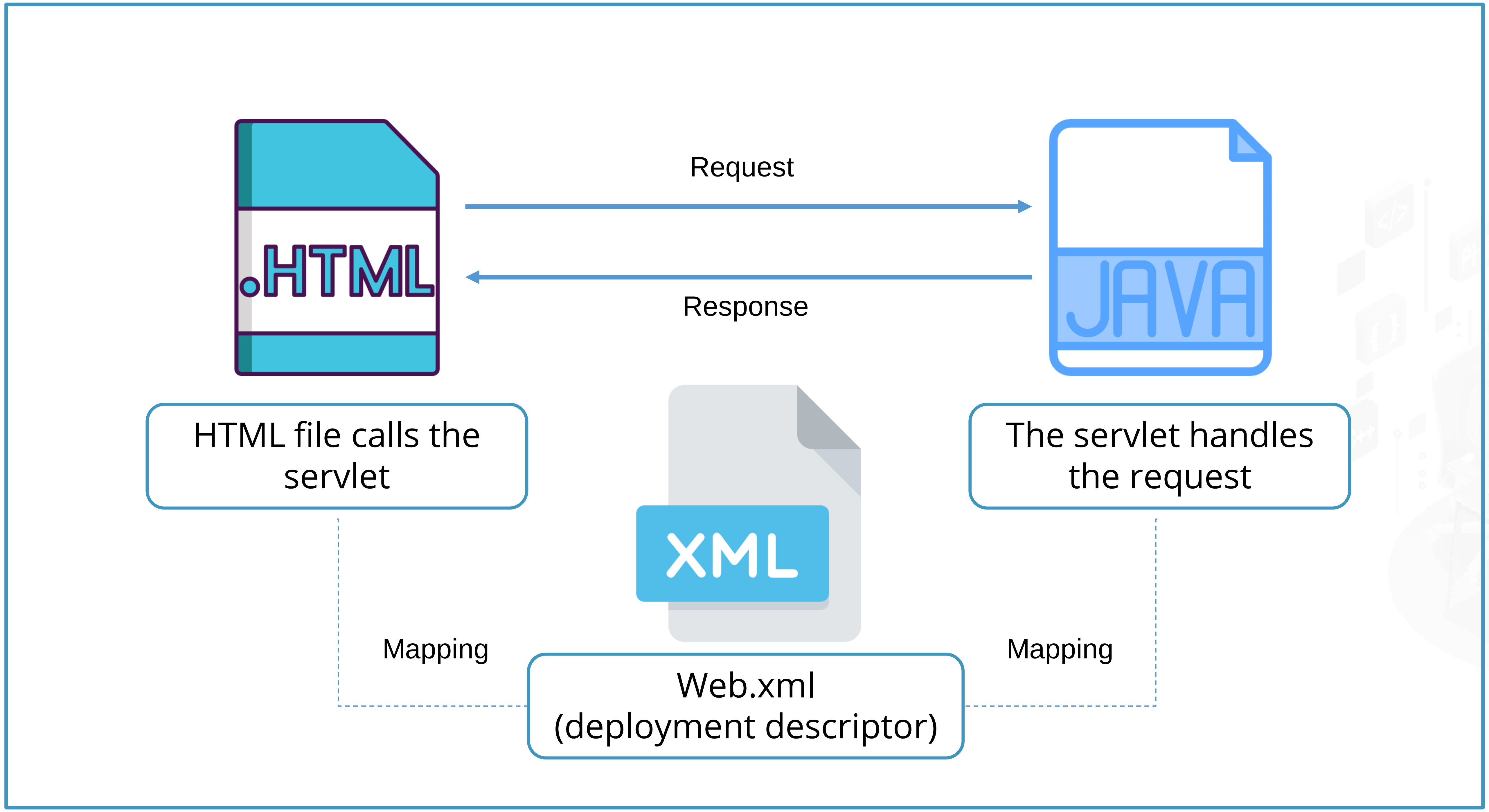
1. Create a dynamic web project in Eclipse IDE.
2. Name your project and generate the deployment descriptor (web.xml).
3. Click on the directory structure of your project in Eclipse. Go to Java Resources, right click on src, select New, and select Servlet.
4. Name your servlet class, servlet name and click Finish.
5. Next, add servlet-api.jar to the project. Click on Libraries, right click on Web App Libraries, and select Build Path, Configure Build Path.
6. Click Add External JARs. Select servlet-api.jar from Apache Tomcat Directory. Configure web.xml and add the webapp to the server. Start the server to run the application.

FULL STACK

Generic Servlets

simplilearn

Generic Servlets


.HTML
Request
Response
JAVA
HTML file calls the servlet
XML
The servlet handles the request
Mapping
Mapping
Web.xml (deployment descriptor)

Generic Servlet: Example

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body><center>
<form name=frmName method="POST"
action="process">
<table width="50%" border="0">
<tr valign="top">
<td width="20%">Enter your name</td>
<td><input name="mname" id="mname"
maxlength=100></td>
</tr>
<tr valign="top">
<td colspan=2 width="100%">
<button>Submit</button>
</td></tr>
</table>
</form></center>
</body>
</html>
```

MyServlet.java

```
import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import java.io.*;

public class MyServlet extends
GenericServlet {
public void service(ServletRequest req,
ServletResponse res) throws
ServletException, IOException
      String sbasic =
req.getParameter("mname");
            res.setContentType("text/html");
            PrintWriter out=res.getWriter();
            out.print("<html><body>");
            out.print("Name:" + sbasic +
"<Br>");
            out.print("</body></html>");
}
}
```

Generic Servlet: Example

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml
/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-
app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>ServletDemo</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-
file>
```

```
<welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
      <servlet-name>MyServlet</servlet-name>
      <servlet-class>MyServlet</servlet-
class>
  </servlet>

  <servlet-mapping>
      <servlet-name>MyServlet</servlet-name>
      <url-pattern>/process</url-pattern>
   </servlet-mapping>

</web-app>
```

Generic Servlets

Duration: 30 min.

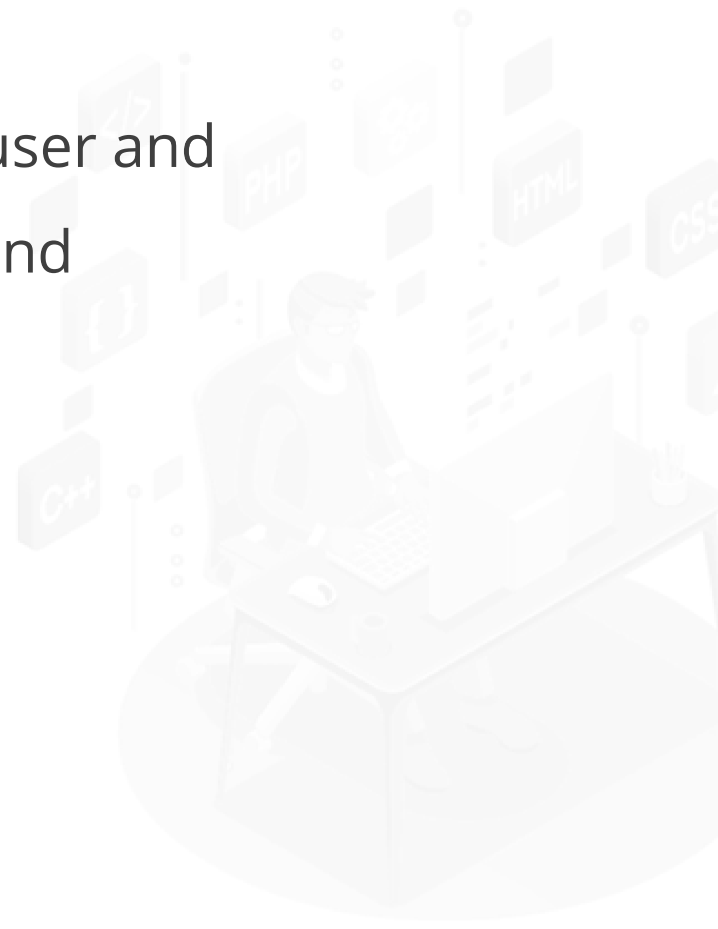
Problem Statement:
Write a program to demonstrate the concept of generic servlets.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the concept of Generic Servlets:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a servlet program in Java that accepts first name and last name from the user and prints the full name. (Create a HTML file that accepts first name and last name and sends this information to the servlet).
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.



FULL STACK

Servlet Interfaces

Servlet Interfaces

Servlet	**Servlet** interface defines methods that all servlets must implement.
ServletConfig	**ServletConfig** object is created when the servlet is initialized. ServletConfig object is used to get configuration information from web.xml.
ServletContext	**ServletContext** object is used to access context initialization parameters and communicate with the servlet container. There can be only one ServletContext object per application.
ServletRequest	**ServletRequest** object is used to provide client request information to a servlet.

Servlet Interfaces: Servlet

All the servlets must implement the servlet interface. This interface provides life cycle methods for initializing the servlet, destroying the servlet, and for providing service as per the requests.

Methods

Method	Description
public void init(ServletConfig config)	Life cycle method to initialize the servlet that is invoked only once
public void service(ServletRequest request,ServletResponse response)	Method to provide response to the incoming request that is invoked at each request
public void destroy()	Life cycle method to invalidate a servlet that is invoked only once
public ServletConfig getServletConfig()	Method to retrieve a ServletConfig object
public String getServletInfo()	Method to get information about the servlet

Servlet Interfaces: ServletConfig

ServletConfig object is used to get servlet configuration information from deployment descriptor. When a servlet is initialized, a ServletConfig object is created.

Methods

Method	Description
public String getInitParameter(String name)	Method returns parameter value for a specified parameter name
public Enumeration getInitParameterNames()	Method returns an enumeration of all the initialization parameter names
public String getServletName()	Method returns the name of the servlet
public ServletContext getServletContext()	Method returns ServletContext object

Servlet Interfaces: ServletContext

The web container creates a ServletContext object when the project is deployed. It is used to access configuration information from a deployment descriptor.

Methods

Method	Description
public String getInitParameter(String name)	Method returns parameter value for a specified parameter name
public Enumeration getInitParameterNames()	Method returns an enumeration of all the initialization parameter names
public void setAttribute(String name,Object object)	Method to set the given object in the application scope
public Object getAttribute(String name)	Method returns the attribute for the specified name
public Enumeration getInitParameterNames()	Method returns an enumeration of all the context initialization parameters
public void removeAttribute(String name)	Method removes the attribute with the given name from the servlet context

Servlet Interfaces: ServletRequest

The ServletRequest object is used to provide client information to a servlet.

Methods

Method	Description
public String getParameter(String name)	Method returns parameter value for a specified parameter name
public String[] getParameterValues(String name)	Method returns an array of string containing all values of the given parameter name
java.util.Enumeration getParameterNames()	Method returns an enumeration of all the request parameter names
public int getContentLength()	Method returns the character set encoding for the input of the request

Servlet Interfaces: ServletRequest

Methods

Method	Description
public String getContentType()	Method returns Internet Media Type of the request entity data or null if not known
public ServletInputStream getInputStream() throws IOException	Method returns an input stream for reading binary data in the request body
public abstract String getServerName()	Method returns the host name of the server that received the request
public int getServerPort()	Method returns port number on which this request was received

Servlet Classes and Interfaces

Duration: 30 min.

Problem Statement:
Write a program to demonstrate the concept of servlet classes and interfaces.

Assisted Practice: Guidelines

Steps to demonstrate the Servlet Interfaces:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.

2. Write a servlet program in Java to demonstrate servlet classes and interfaces.

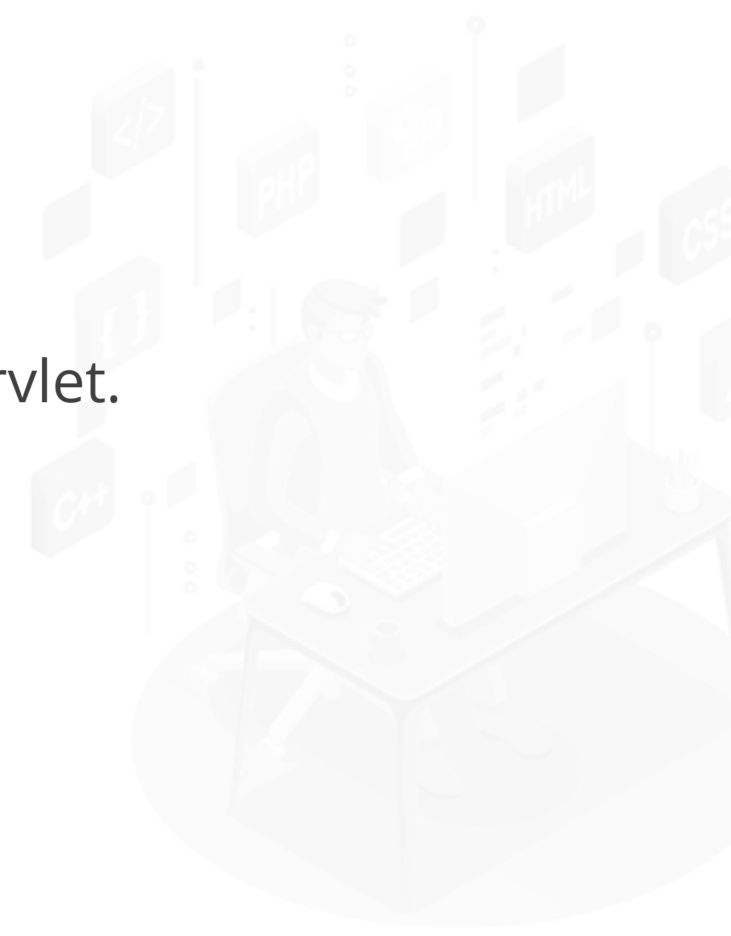
3. Create a HTML file that accepts user input and sends this information to the servlet.

4. Run the HTML code on your browser.

5. Initialize the .git file.

6. Add and commit the program files.

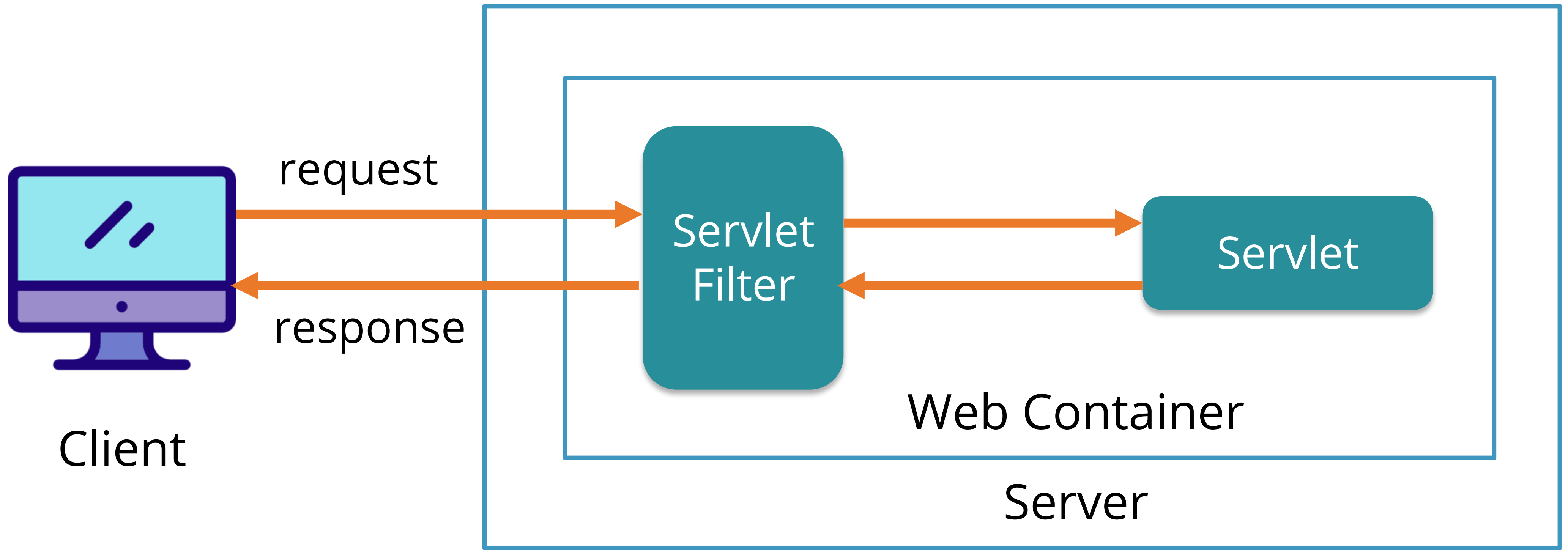
7. Push the code to your GitHub repository.



Servlet Filters

Servlet Filters

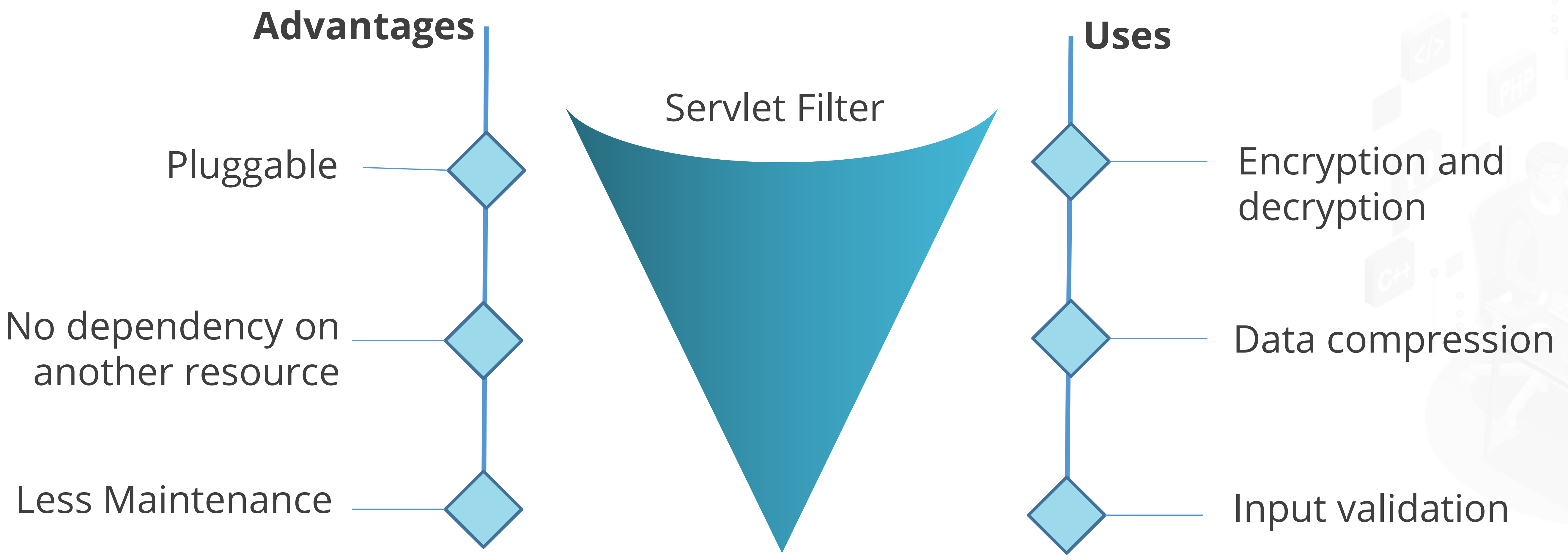
Java filters are **pluggable** Java components that are used to intercept and process requests before they are sent to the servlet and responses sent by the servlet.



No cap on the number of filters

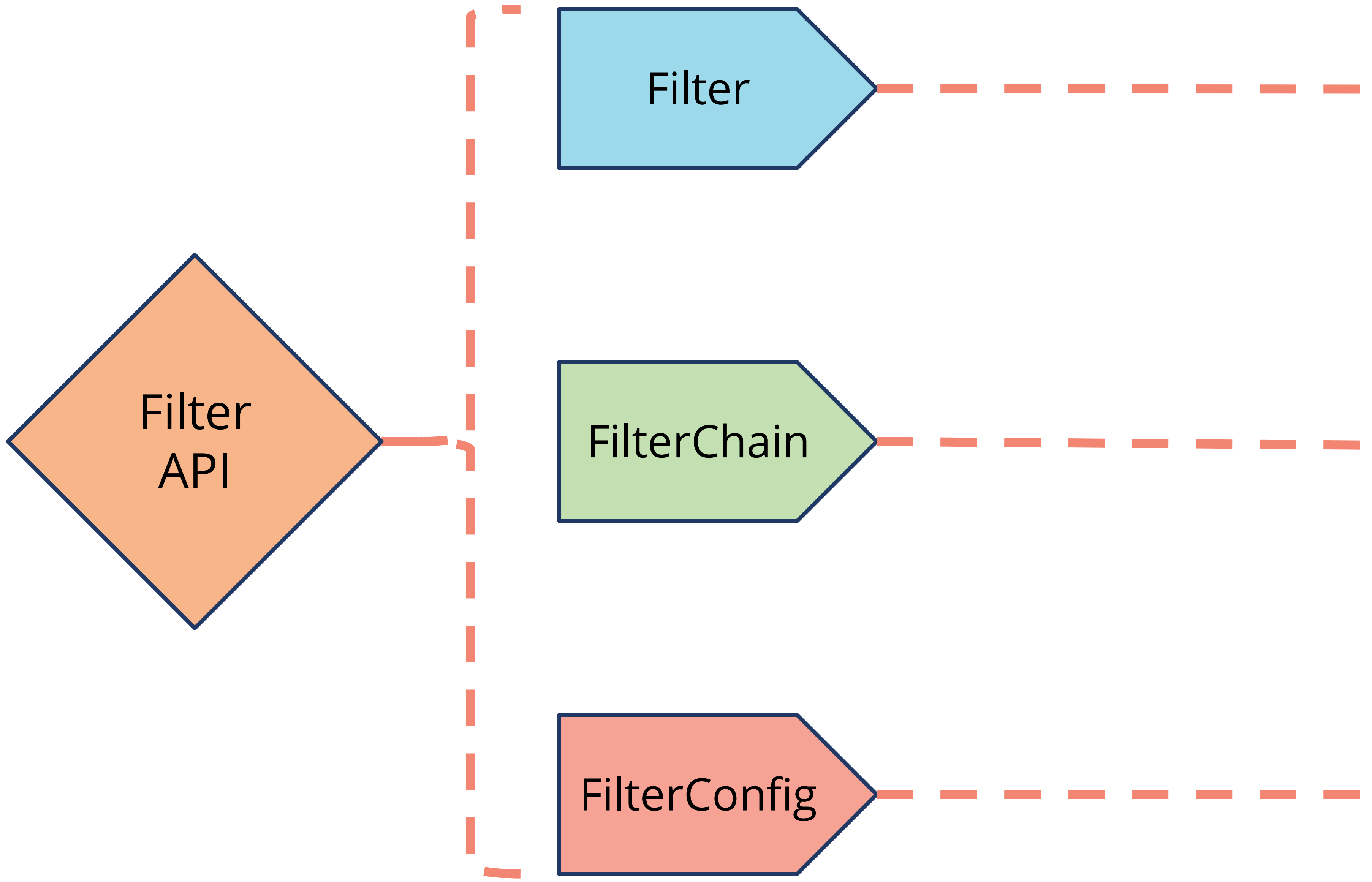
Configured in the deployment descriptor

Servlet Filters: Advantages and Uses



Filter API

Filter API is a part of Servlet API. **Filter**, **FilterChain,** and **FilterConfig** are the three interfaces of Filter API.



A filter must implement a **Filter** interface.

A filter uses a **FilterChain** object to call the next filter in the chain.

A filter uses a **FilterConfig** object to get the configuration information.

Authentication Filter

Authentication filter performs pre- and post-processing of authentication functions.

Authentication filter – An extension to filter object

Acts as a helper to an authentication provider

Servlet Filter: Example

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

@WebFilter("/AuthenticationFilter")
public class AuthenticationFilter
implements Filter {

     private ServletContext context;

     public void init(FilterConfig
fConfig) throws ServletException {
          this.context =
fConfig.getServletContext();
     } public void
doFilter(ServletRequest sreq,
ServletResponse sresp, FilterChain chain)
throws IOException, ServletException {
```

```
HttpServletRequest req =
(HttpServletRequest) sreq;
HttpServletResponse res =
(HttpServletResponse) sresp;

           HttpSession session =
req.getSession(false);
           if(session == null){

     this.context.log("Unauthorized
access request");

     res.sendRedirect("login.html");
           }else{
                chain.doFilter(request,
response);
           }
     }
     public void destroy() {
     }
}
```

Servlet Filters

Duration: 15 min.

Problem Statement:
Write a program to demonstrate a servlet filter.

Assisted Practice: Guidelines

Steps to demonstrate a Servlet Filter:

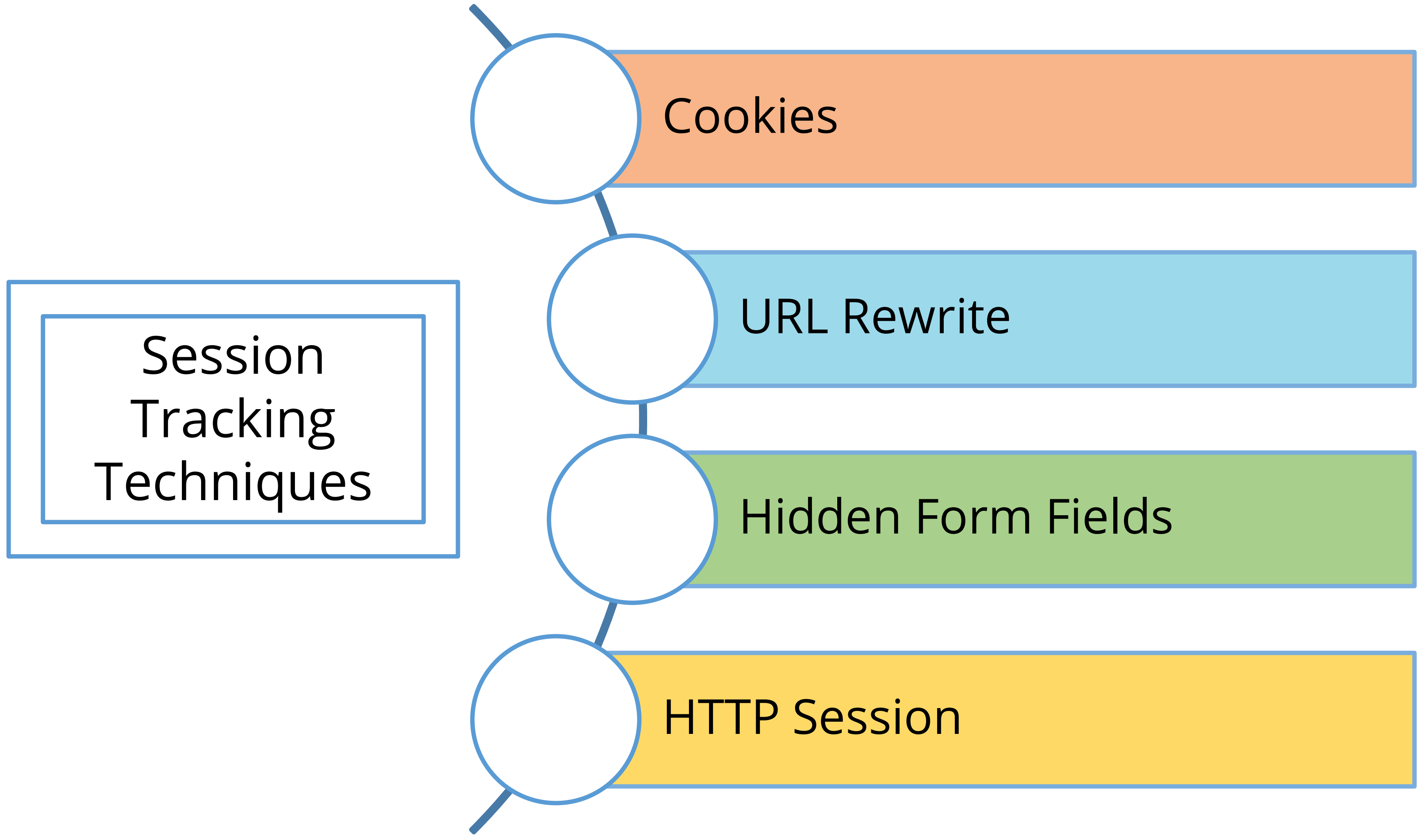
1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a servlet program to demonstrate the working of a servlet filter.
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.

FULL STACK

Session Tracking

Servlet Session Tracking

Session tracking is a mechanism that the web container uses to store session information of a user.



Session Tracking Using Cookies

Cookies are used to maintain the session.

- Unique session ID is assigned to each client by the web server
- Web server uses cookies to recognize subsequent requests from the client
- Not a recommended technique

Cookies are small pieces of data sent by the web server to the web browser.

Session Tracking Using Cookies: Example

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

@WebServlet("/CheckCookie")
public class CheckCookie extends
HttpServlet {

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

           Cookie[] requestCookies =
request.getCookies();
           boolean userIdExists = false;
```

```
if(requestCookies != null){
            for(Cookie c :
requestCookies){
               if
(c.getName().equals("userid") &&
c.getValue() != null)
                userIdExists = true;
            }
if (userIdExists)
     response.sendRedirect("/accountDashb
oard");
             else
     response.sendRedirect("/sessionError
");
           }
           else
     response.sendRedirect("/sessionError
");
}
}
```

Session Tracking Using Cookies

Duration: 15 min.

Problem Statement:
Write a program to demonstrate a session tracking using cookies.

Assisted Practice: Guidelines

Steps to demonstrate session tracking using cookies:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a program in Java to demonstrate session tracking using cookies.
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.

FULL STACK

Session Tracking Using URL Rewrite

Session Tracking Using URL Rewrite

A session is maintained by rewriting the URL.

- Session ID is appended at the end of the URL.
- The server associates the session identifier with the session data.
- Disadvantage – To assign a session ID, URL has to be dynamically generated.

Example

```
http://sampleSite.com/sampleHTM.htm;sessionid
=98745
```

Session Tracking Using URL Rewrite: Example

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

    public class SessionCheckServlet
extends HttpServlet {

    public void doGet(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
            try{
            boolean userIdExists = false;
            String
n=request.getParameter("userid");
                  if (n != null)
             userIdExists = true;
```

```
if (userIdExists)

response.sendRedirect("/accountDashboard"
);
         else
response.sendRedirect("/sessionError");
            } catch(Exception e){
     response.sendRedirect("/sessionError
");

         }
       }


   }
```

Session Tracking Using URL Rewrite

Duration: 15 min.

Problem Statement:
Write a program to demonstrate a session tracking using URL rewrite.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate session tracking using URL rewrite:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a program in Java to demonstrate session tracking using URL rewrite.
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.

FULL STACK

Session Tracking Using Hidden Form Fields

Session Tracking Using Hidden Form Fields

A session is maintained using a hidden form field.

- Session ID is stored in the client browser and sent to the server when the form is submitted.
- The hidden fields are not directly visible to the users.
- Disadvantage – Increased network traffic

Example

```
input type="hidden'  name="sessionid"
       value="98765"
```

Session Tracking Using Hidden Form Fields: Example

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

    public class SessionCheckServlet
extends HttpServlet {

    public void doGet(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
            try{
             boolean userIdExists = false;
String n =
request.getParameter("field_userid");
                  if (n != null)
             userIdExists = true;
```

```
 if (userIdExists)

response.sendRedirect("/accountDashboard"
);
             else

response.sendRedirect("/sessionError");

             } catch(Exception e){

     response.sendRedirect("/sessionError
");
   }
         }

    }
```

Session Tracking Using Hidden Form Fields

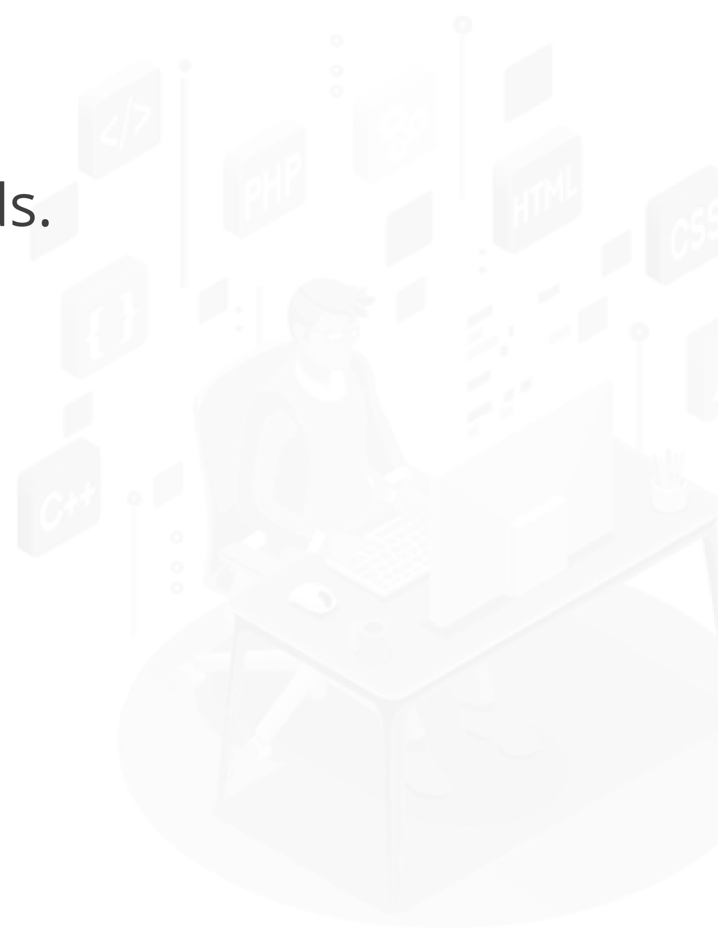
Duration: 15 min.

Problem Statement:
Write a program to demonstrate session tracking using hidden form fields.

Assisted Practice: Guidelines

Steps to demonstrate session tracking using hidden form fields:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a program in Java to demonstrate session tracking using hidden form fields.
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.



FULL STACK

Session Tracking Using HTTP Session

Session Tracking Using HTTP Session

A session is maintained using a HTTP Session object.

- Servlet container uses HTTPSession interface to create a session.
- User session is maintained across multiple HTTP requests.
- Attributes can be stored, removed, or retrieved from an HTTPSession object.

Session Tracking Using HTTP Session: Example

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

    public class SessionCheckServlet
extends HttpServlet {

    public void doGet(HttpServletRequest
request, HttpServletResponse response)
throws ServletException, IOException {
            try{
             boolean userIdExists = false;
             HttpSession
session=request.getSession(false);
             String
n=(String)session.getAttribute("userid");
                   if (n != null)
             userIdExists = true;
```

```
 if (userIdExists)

response.sendRedirect("/accountDashboard"
);
            else

response.sendRedirect("/sessionError");

            } catch(Exception e){

      response.sendRedirect("/sessionError
");

          }
        }
    }
```

Session Tracking Using HTTP Session

Duration: 15 min.

Problem Statement:
Write a program to demonstrate session tracking using an HTTP session object.

Assisted Practice: Guidelines

Steps to demonstrate session tracking using HTTP session:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a program in Java to demonstrate session tracking using HTTP session.
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.

Session Login and Logout

Session Login and Logout

User session is maintained using session login and logout.

Session Login

Starts a user session

```
//Create a new session
HttpSession session=request.getSession();

//Also used to create a new session
HttpSession session = request.getSession(true);

//Get a pre-existing session
HttpSession session =
request.getSession(false);
```

Session Logout

Ends a user session

```
//Destroy (invalidate) a session
session.invalidate();
```

Session Login: Example

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

    public class LoginServlet extends
HttpServlet {

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

            try{

           String userid =
request.getParameter("userid");
           String pwd =
request.getParameter("pwd");

           if (userid.equals("admin") &&
pwd.equals("12345")) {
```

```
 HttpSession
session=request.getSession();

session.setAttribute("userid",userid);

     response.sendRedirect("/accountDashb
oard");
            } else

     response.sendRedirect("/loginError")
;

             } catch(Exception e){

     response.sendRedirect("/loginError")
;
           }
         }
    }
```

Session Logout: Example

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

    public class LogoutServlet extends
HttpServlet {

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

            try{HttpSession
session=request.getSession();
           session.invalidate();
```

```
      response.sendRedirect("/login");

             } catch(Exception e){

      response.sendRedirect("/logoutError"
);

          }
        }
    }
```

Session Login and Logout

Duration: 15 min.

Problem Statement:
Write a program to demonstrate session login and logout.

Assisted Practice: Guidelines

Steps to demonstrate session login and logout:

1. Create a dynamic web project in Eclipse IDE and configure a servlet.
2. Write a program in Java to demonstrate session login and logout.
3. Run the HTML code on your browser.
4. Initialize the .git file.
5. Add and commit the program files.
6. Push the code to your GitHub repository.

Key Takeaways

- Web technology allows computers to communicate with each other using markup languages and multimedia packages.
- Java servlets are used on server side, can handle complicated requests from web servers, and are used to develop dynamic web pages.
- Advantages of servlets are better performance, portability, robustness, and security.
- Servlet interfaces are used to provide common behavior to all servlets.
- Servlet filters are pluggable and used to perform filtering operations such as logging, encryption, decryption, and input validation.
- To maintain the state of the data of a user, session tracking is used.

Validation of the User Login

Duration: 30 min.

Problem Statement:

Create a servlet-based web application that shows a login page and validates it. The correct values are hard-coded. On successful login, a dashboard page is shown. The dashboard will provide a link for logging out. Incorrect logins need to be handled by showing an error message page.

LESSON-END PROJECT

Before the Next Class

Course: SQL Training

You should be able to:

- Explain what is a database
- Demonstrate SQL queries
- Filter results using queries
- Group data using queries