Java

Advance Programming

CSE 456

CSE 456 – Advance Java Programming

Bachelor of Software Engineering

Department of Computer networks and Data Communications

School of Computing and Information Technology

Course Overview

EASTERN INTERNATIONAL UNIVERSITY

Course Description

- This course is an elective course for SE program. This course aims to give a structured overview of web application development using J2EE. The course highlights the fundamental concepts about the technology, development and architectures of Servlet, JSP, Hibernate, Spring, Spring Boot.
- Students will learn how to create enterprise-based apps by encapsulating an application's business logic. This course helps students strengthen their programming skills in order to design dynamic web applications utilizing server-side technologies and Java Database Connectivity. Students may learn about Model-View-Controller (MVC) architecture. Various Java frameworks, such as Spring, Spring Boot and Hibernate, can help students to improve their web application development skills and meet industry needs. The prerequisite for this course are CSE 102.

Course Objectives

- Understand the technological trends of web application development using java, different components of server-side programming, MVC architecture, modern frameworks and their applications.
- Familiarize the student with the JDBC frameworks such as Hibernate to create dynamic web pages, using Servlets, JSP with the help of MVC architecture.
- Understand and develop web-based enterprise applications using Spring, Spring Boot, Spring Security.
- Analyze the mode of operation of different types of frameworks that are used to interconnect a distributed community of a web application and various interfacing standards.
- Apply the advance java concepts, preparing the student for evaluate the challenges in building a web application project using java.

Course Learning Outcomes

- Knowledge:
  - **CLO1**: Understand the knowledge of Web Server, Web Container, and Application Server, as well as J2EE architecture, MVC architecture.
  - **CLO2**: Apply Server-Side Programming by developing dynamic web pages, using Servlets, JSP, JSTL, JDBC, Hibernate.
  - **CLO3**: Analyze different frameworks such as Spring, Spring Boot and Hibernate to develop the multi-tier architecture of web-based enterprise applications.
  - **CLO4**: Apply and analyze security on web-based applications using Spring, Spring Boot frameworks.

Course Learning Outcomes

- Skills:
  - **CLO5**: Design and develop various applications by combining any of Servlets, JSPs with JDBC, Spring, Spring Boot, and Hibernate by analyzing requirements and evaluating existing systems.
  - **CLO6**: Use Spring and Spring Boot frameworks to map Java classes and object associations to relational database tables with Hibernate mapping files.
- Attitudes
  - **CLO7**: Having honest, hard- working attitude , awareness and responsibility when using laboratory equipment.
  - **CLO8**: Use communication skills, skills to work independently and as part of a team.

Course Learning Outcomes

- Ability, responsibility and career
  - **CLO9**: Train and improve your self-study ability

Books and Teaching Materials

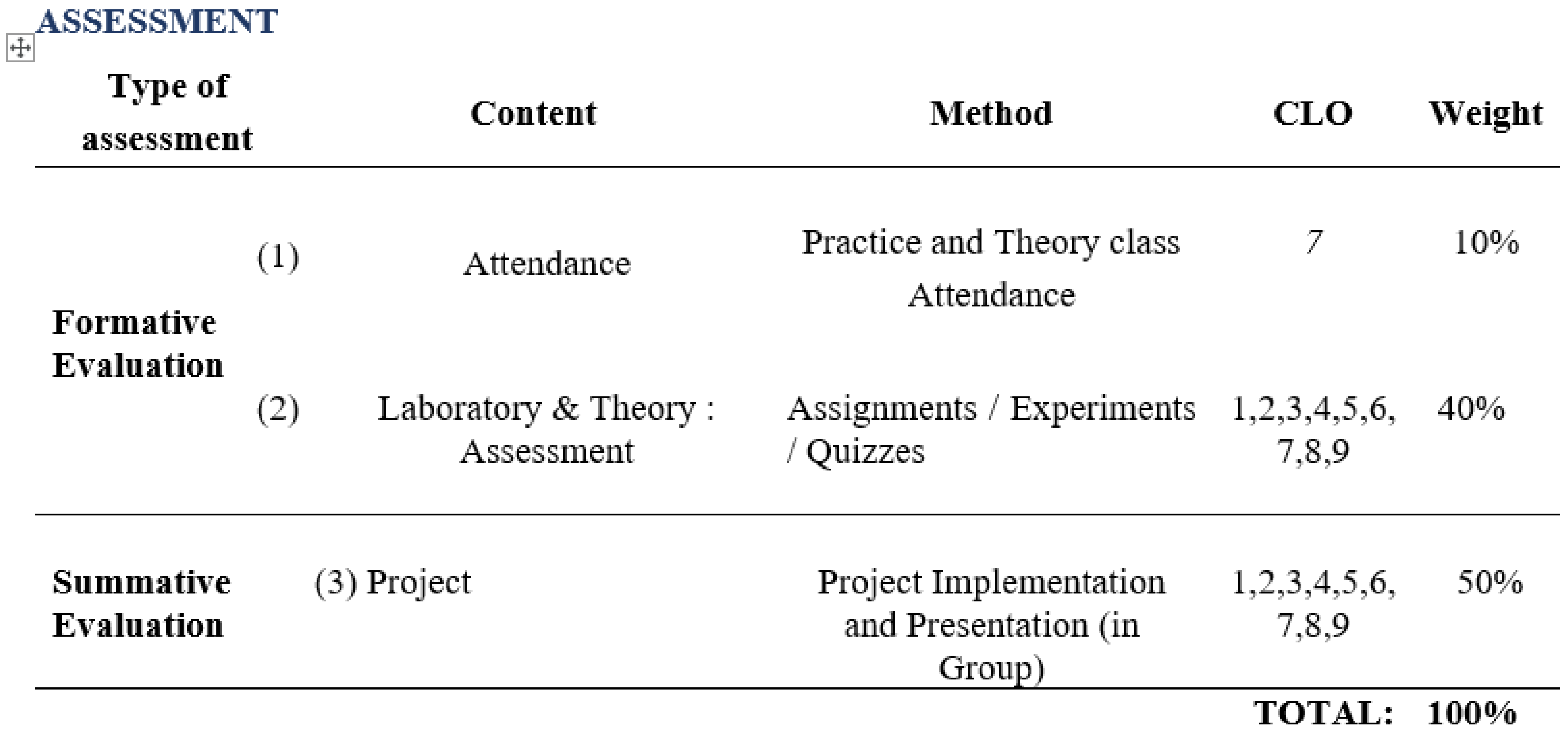
Teaching materials

- [1]. Đặng, Thanh Dũng, TS, 2016. *Giáo trình lập trình web với Servlet và JSP / Đặng Thanh Dũng*. TP. Hồ Chí Minh : Đại học Quốc gia.
- [2]. Karanam, Ranga Rao. 2017. *Mastering Spring 5.0 : Master reactive programming microservices, cloud native applications, and more*. Birmingham, UK : Packt Publishing.

References

- [3]. Bryan Basham, Kathy Sierra, and Bert Bates. 2008. *Head First Servlets and JSP™* (2nd Edition). Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol.
- [4]. Jim Keogh.2002. *J2EE: The Complete Reference (1st Edition)*. McGraw-Hill Osborne Media.
- [5]. Joseph B. Ottinger, Jeff Linwood, Dave Minter. 2016. *Beginning Hibernate: For Hibernate 5* (4th Edition). Apress.

Evaluation Process



ASSESSMENT

Type of assessment		Content	Method	CLO	Weight
Formative Evaluation	(1)	Attendance	Practice and Theory class Attendance	7	10%
	(2)	Laboratory & Theory : Assessment	Assignments / Experiments / Quizzes	1,2,3,4,5,6, 7,8,9	40%
Summative Evaluation	(3) Project		Project Implementation and Presentation (in Group)	1,2,3,4,5,6, 7,8,9	50%
				TOTAL:	100%

CSE 456

Advance Java Programming

Lecture 1

Chapter 1: Servlets

What is Advance Java?

- It is an advanced version of the Java programming language.
- is specifically designed to construct web-based, network-centric, or enterprise applications.
- It incorporates concepts like as Servlet, JSP, JDBC, RMI, Socket programming, and others.
- It is a Web & Enterprise application development platform which basically follows client & server architecture.

Why advance Java?

- It reduces the difficulty of creating a multi-tiered application.
- API standardization across components and application server containers.
- Framework services are provided by JEE application server and containers.

CORE JAVA

VS

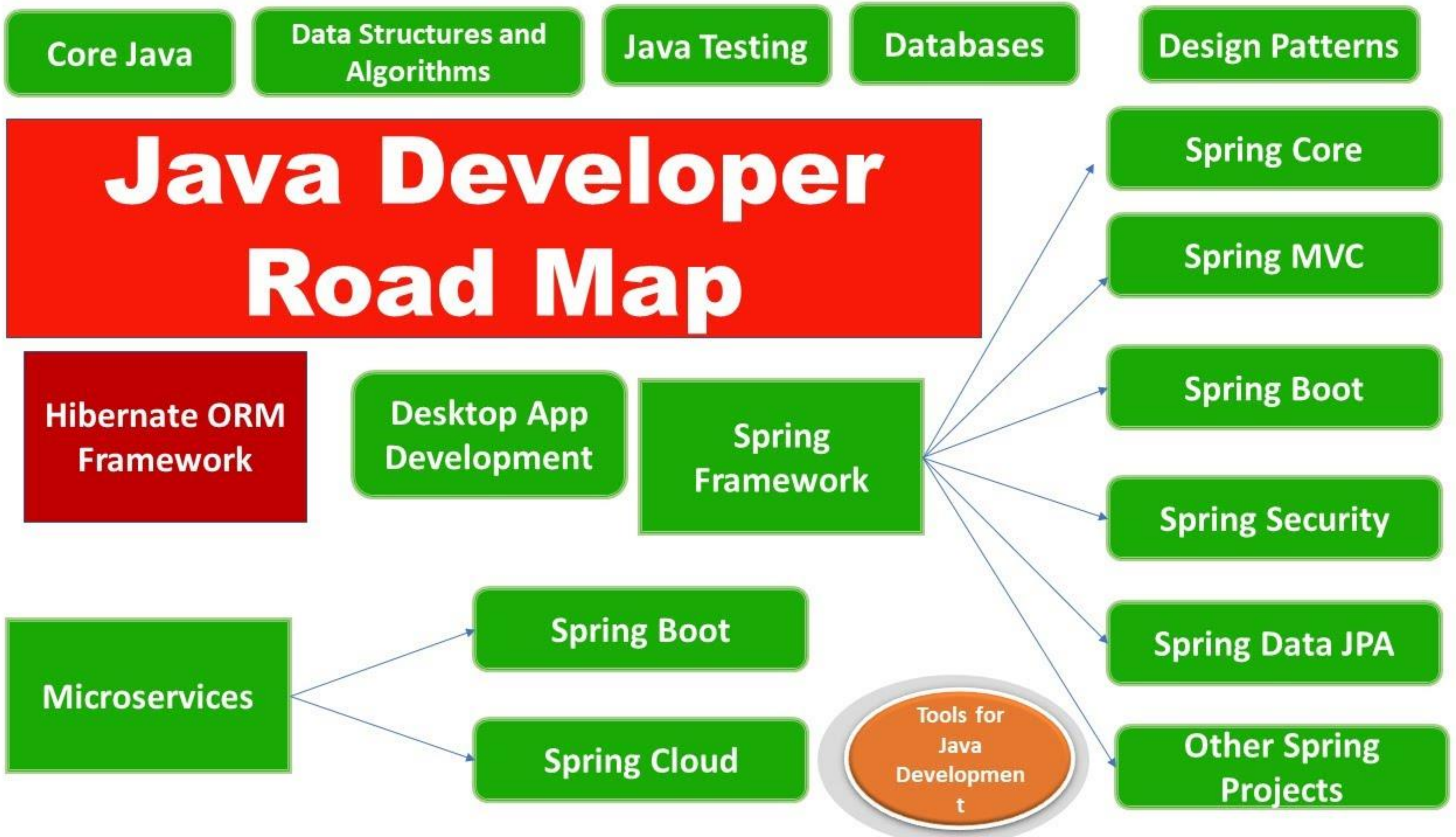
ADVANCED JAVA

Core Java vs Advance Java

- Core Java is used to develop general purpose application. Advance Java is used to develop web-based applications.
- Core Java does not deal with database, socket programming, etc. Advance Java deals with socket programming, DOM, and networking applications.
- Core Java is a single tier architecture. Advance Java is a multi-tier architecture.
- Core Java is a Java Standard Edition. Advance Java is a Java Enterprise Edition.
- Core Java provides java.lang.* package. Advance Java provides java.servlet.* package.

Advance Java Frameworks

Java Developer Road Map

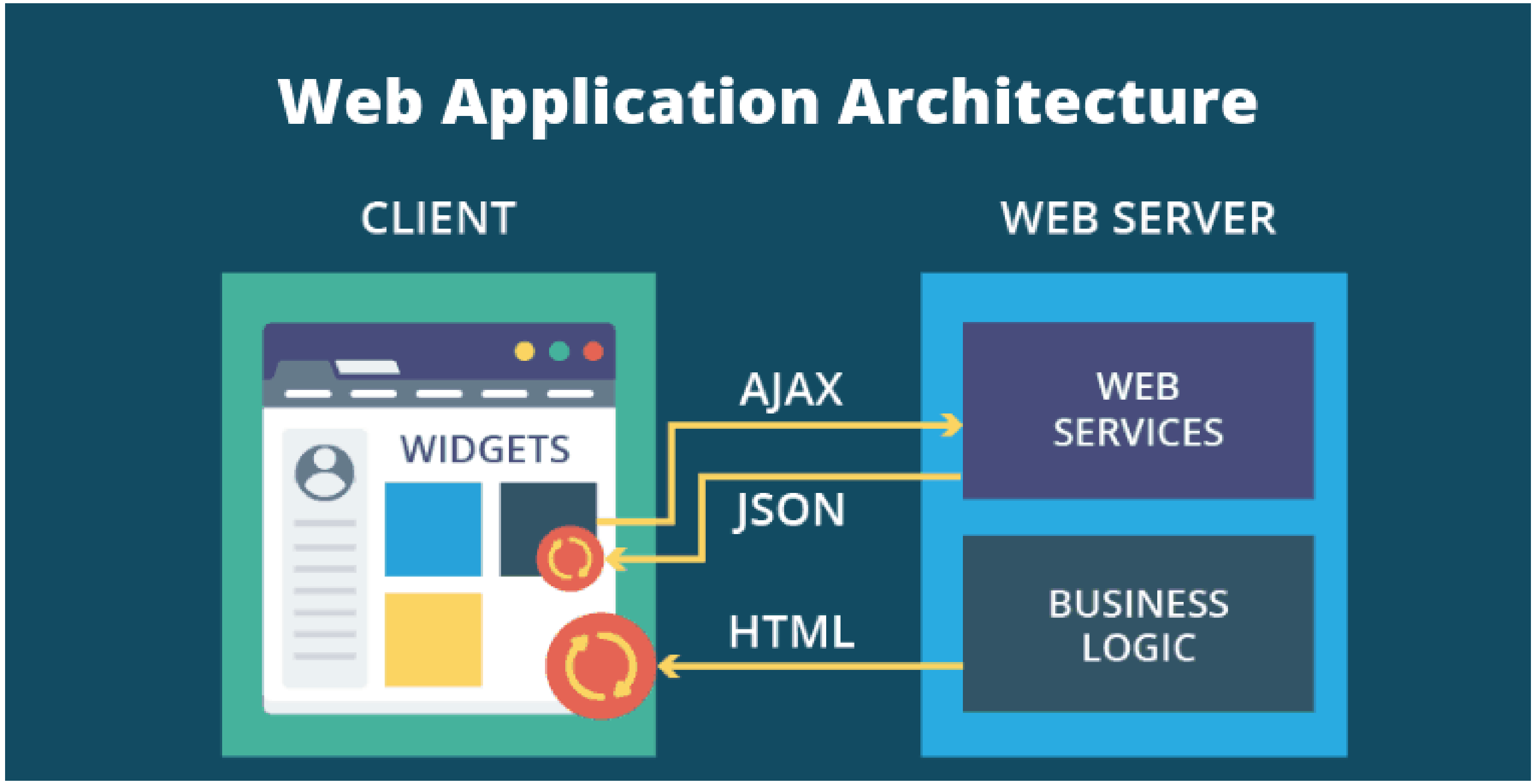


WEBSITE

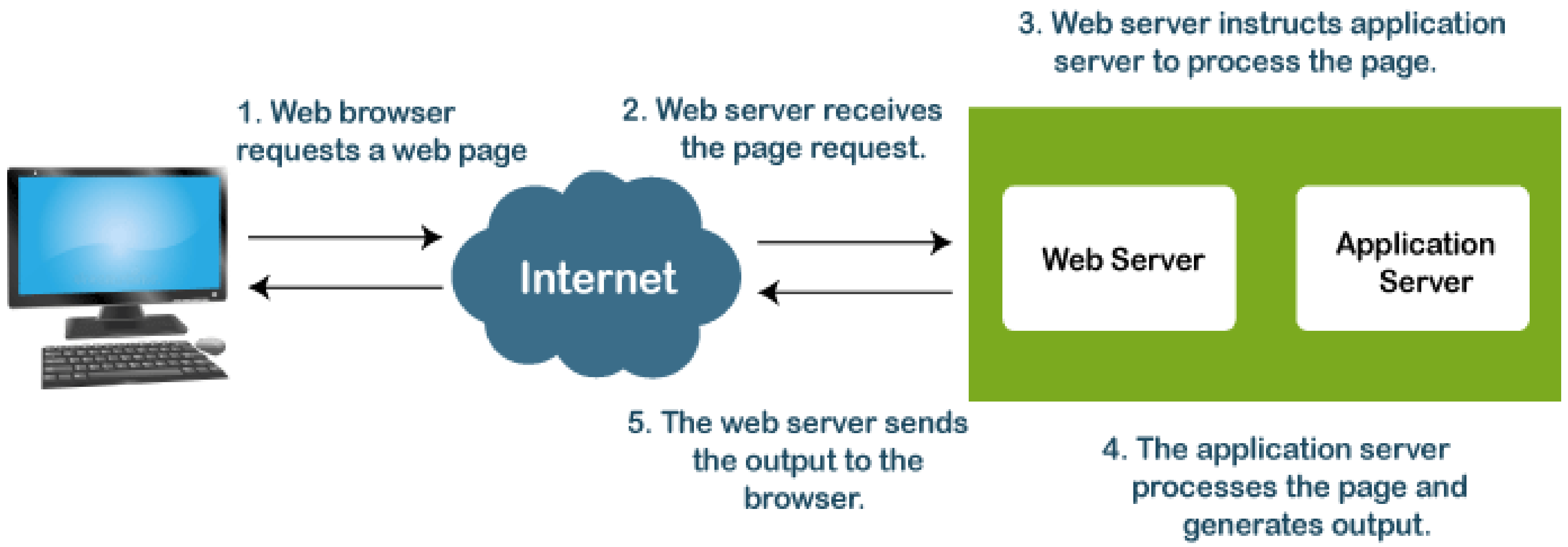
WEB APPLICATION

What is a Web Application?

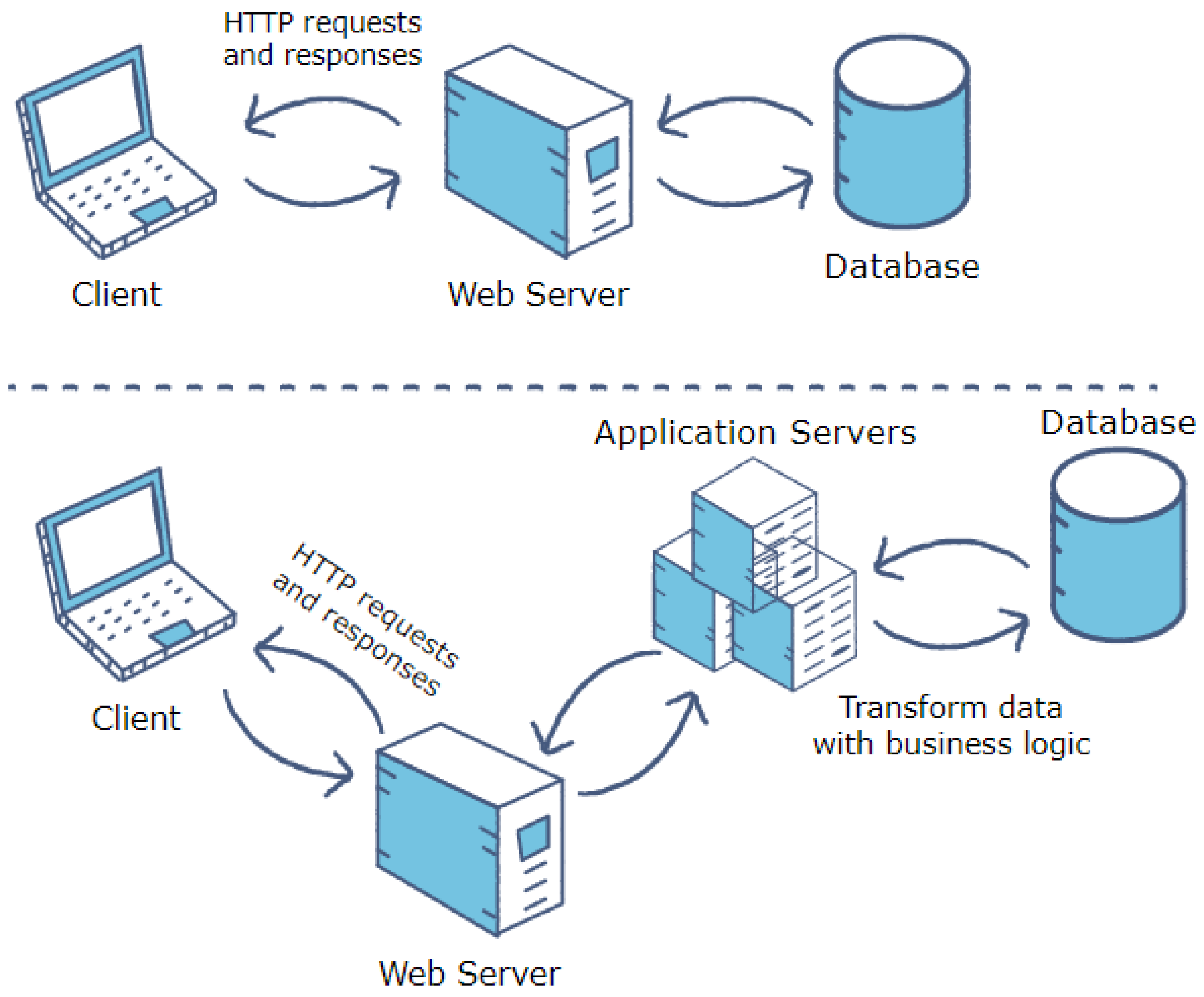
- A web-application is an application program that is usually stored on a remote server, and users can access it using Software known as web-browser.



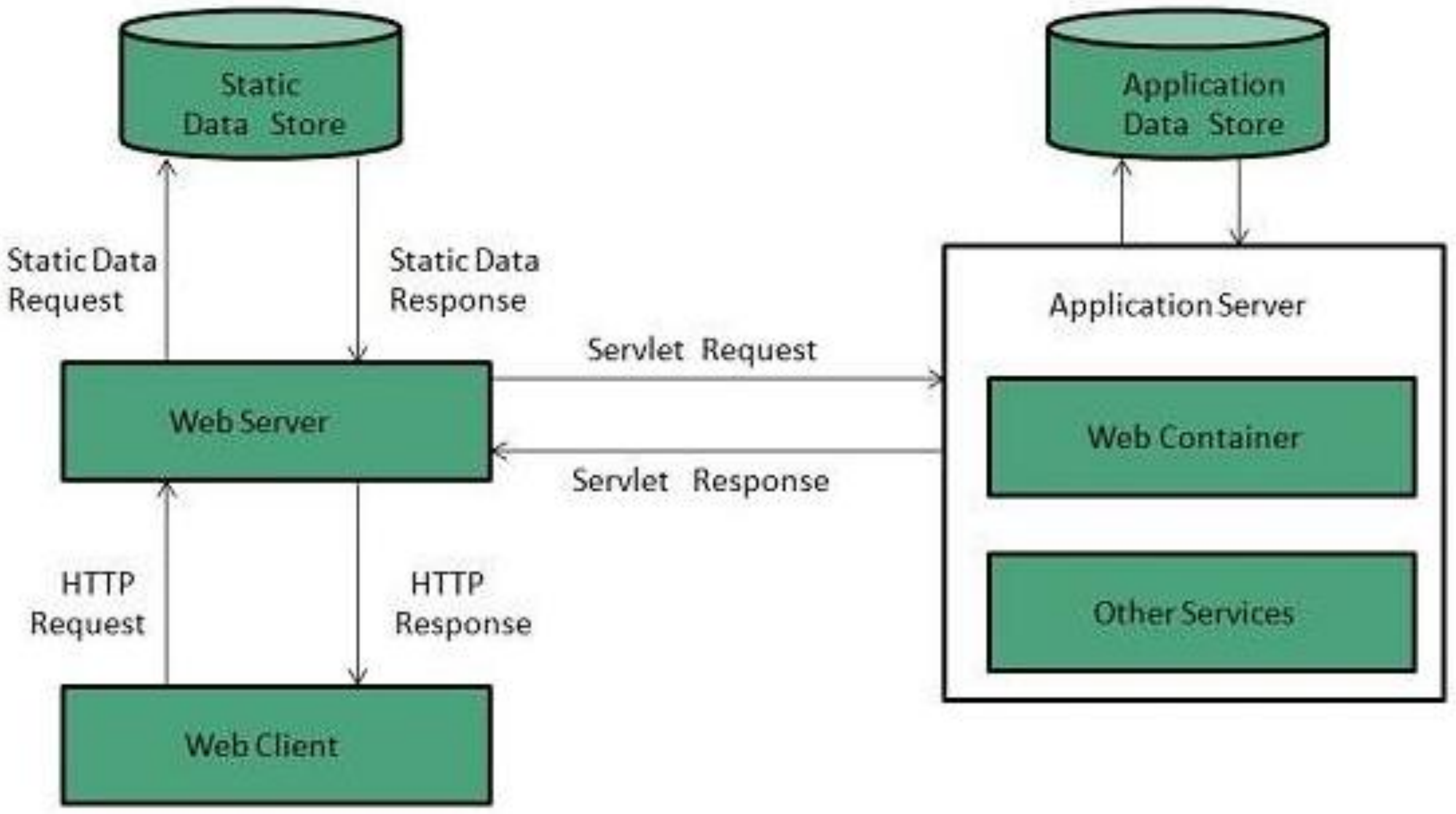
How does a web- application work?


3. Web server instructs application server to process the page.
1. Web browser requests a web page
2. Web server receives the page request.
Internet
Web Server
Application Server
5. The web server sends the output to the browser.
4. The application server processes the page and generates output.

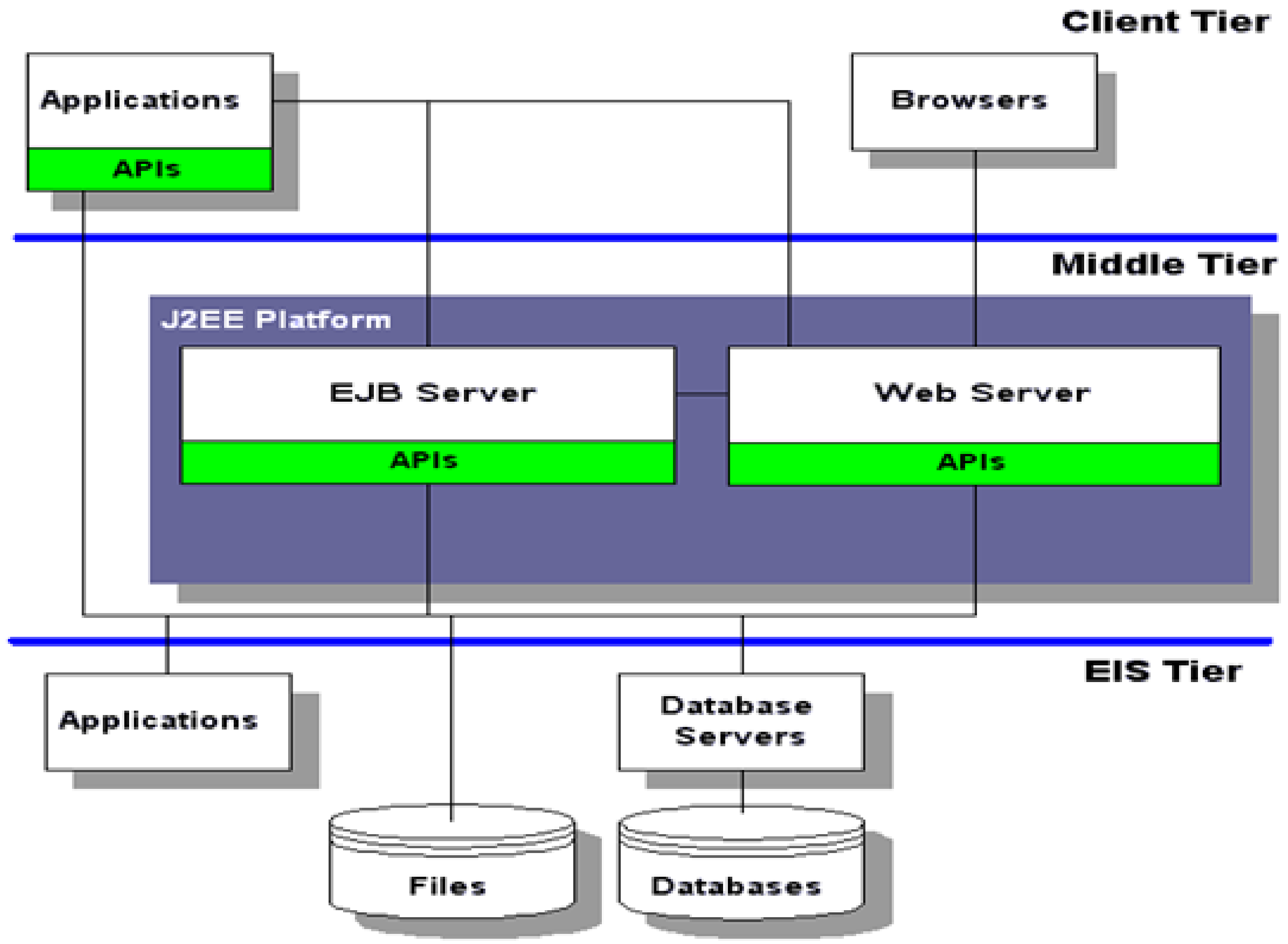
Server: Web Server vs. Application Server


HTTP requests and responses
Client
Web Server
Database
Application Servers
Database
Client
HTTP requests and responses
Transform data with business logic
Web Server

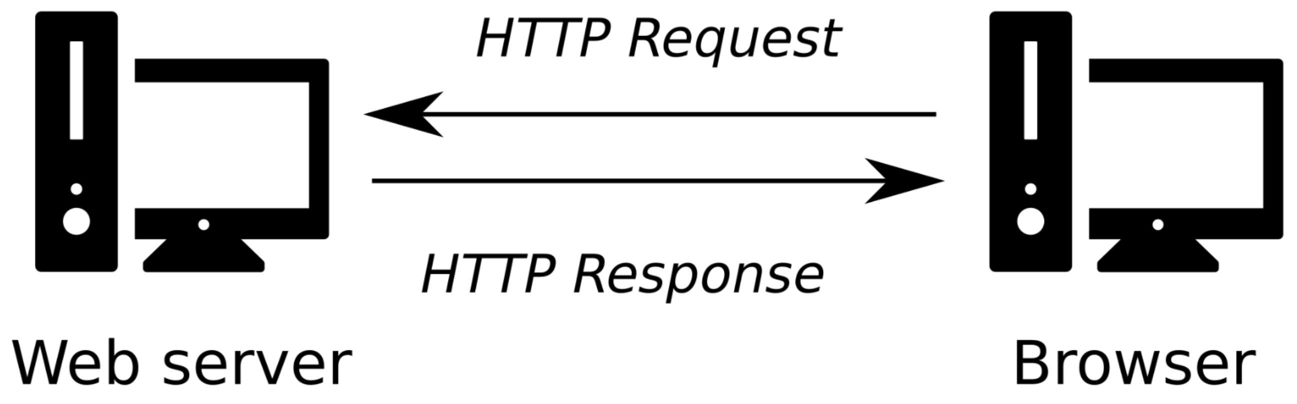
Block Diagram of Web Server

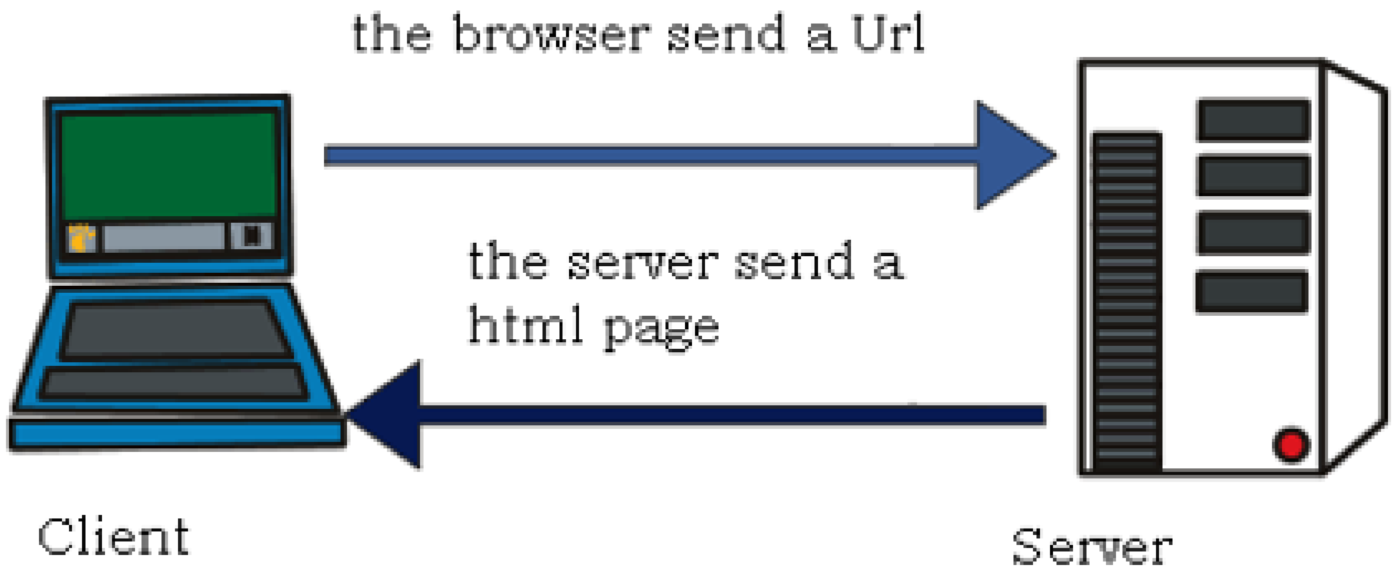


Block Diagram of Application Server



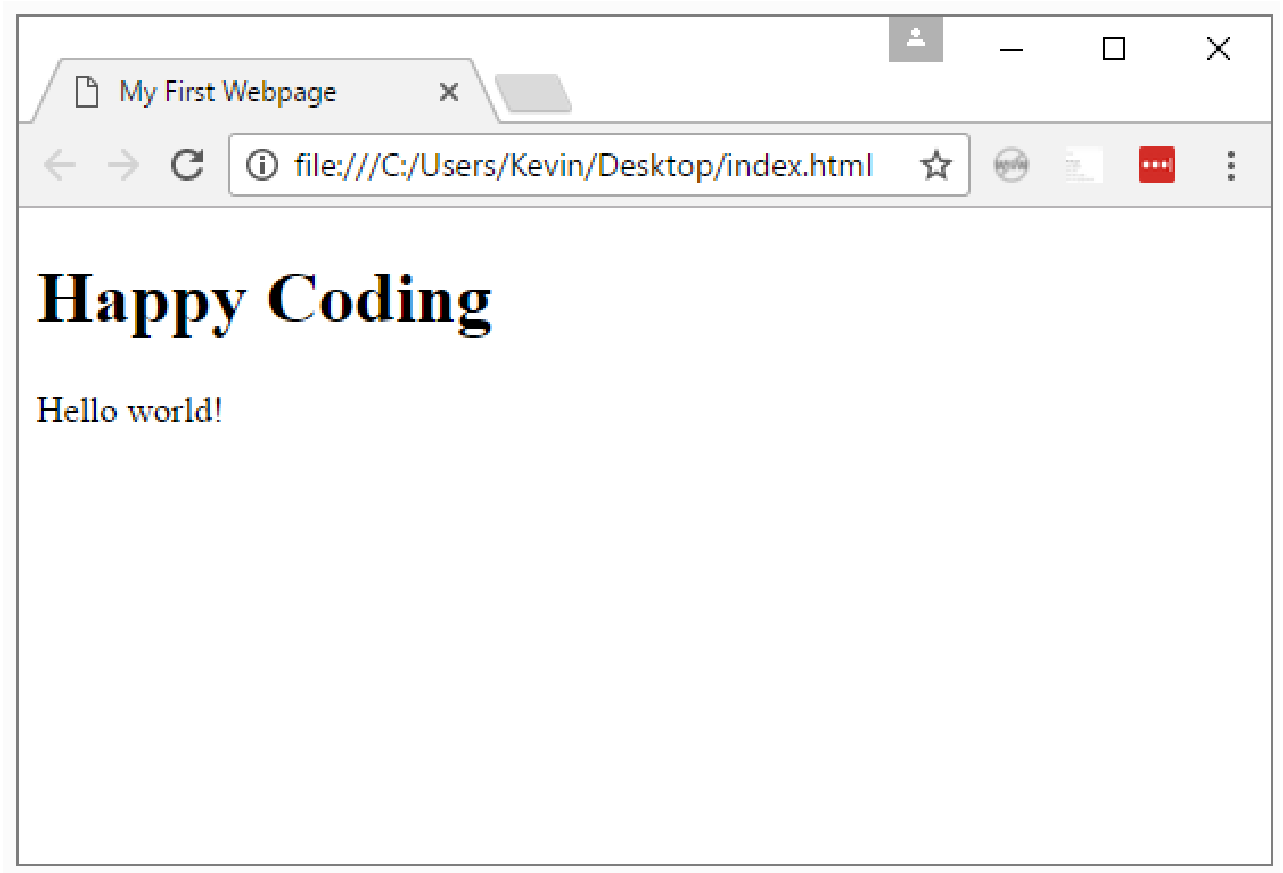
How do client and server communicate?





HTML

```
<!DOCTYPE html>
<html>
     <head>
               <title>
                         My First Webpage
               </title>
     </head>
     <body>
               <h1>Happy Coding</h1>
               <p>Hello world!</p>
     </body>
</html>
```

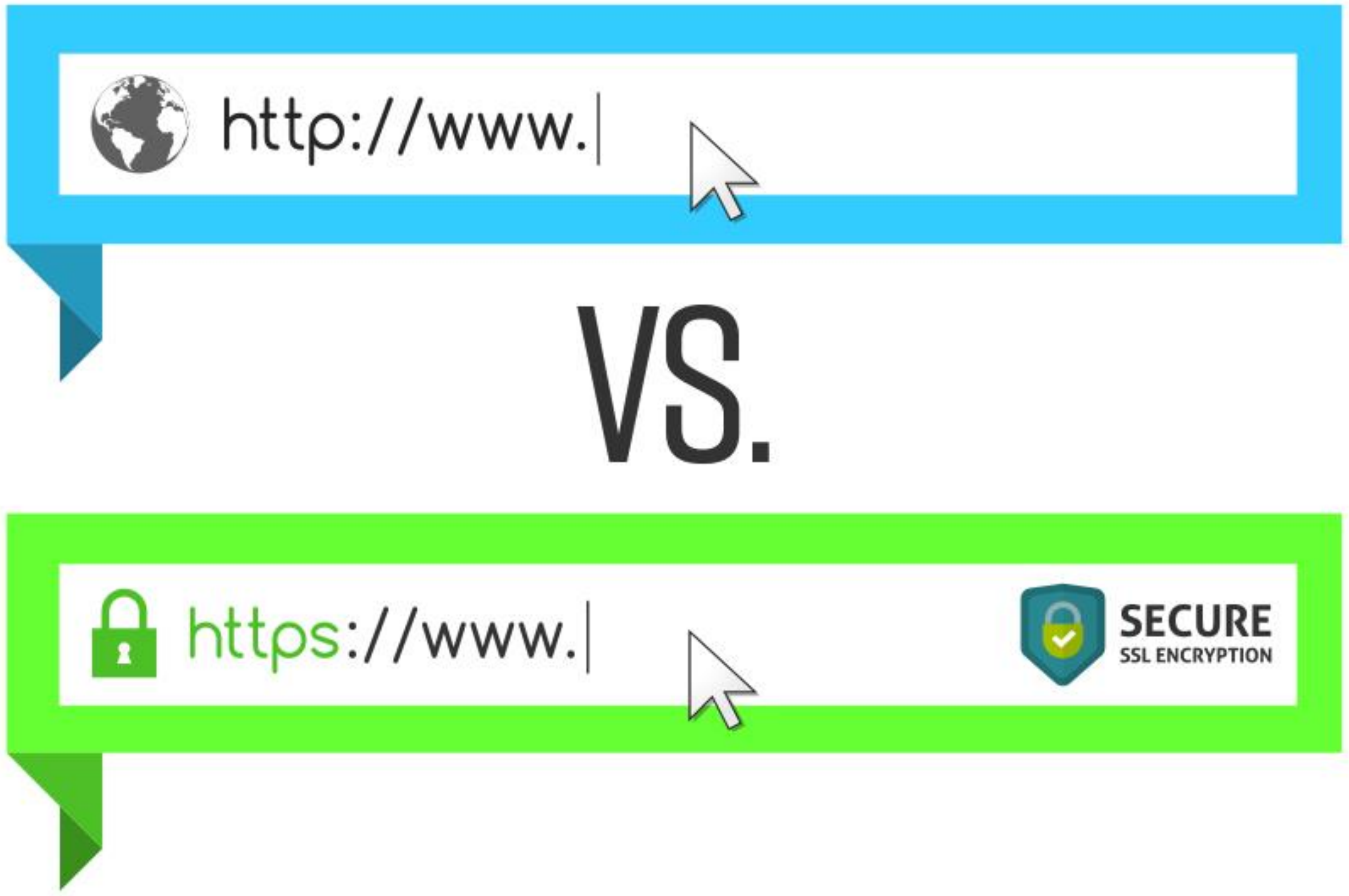

My First Webpage

file:///C:/Users/Kevin/Desktop/index.html

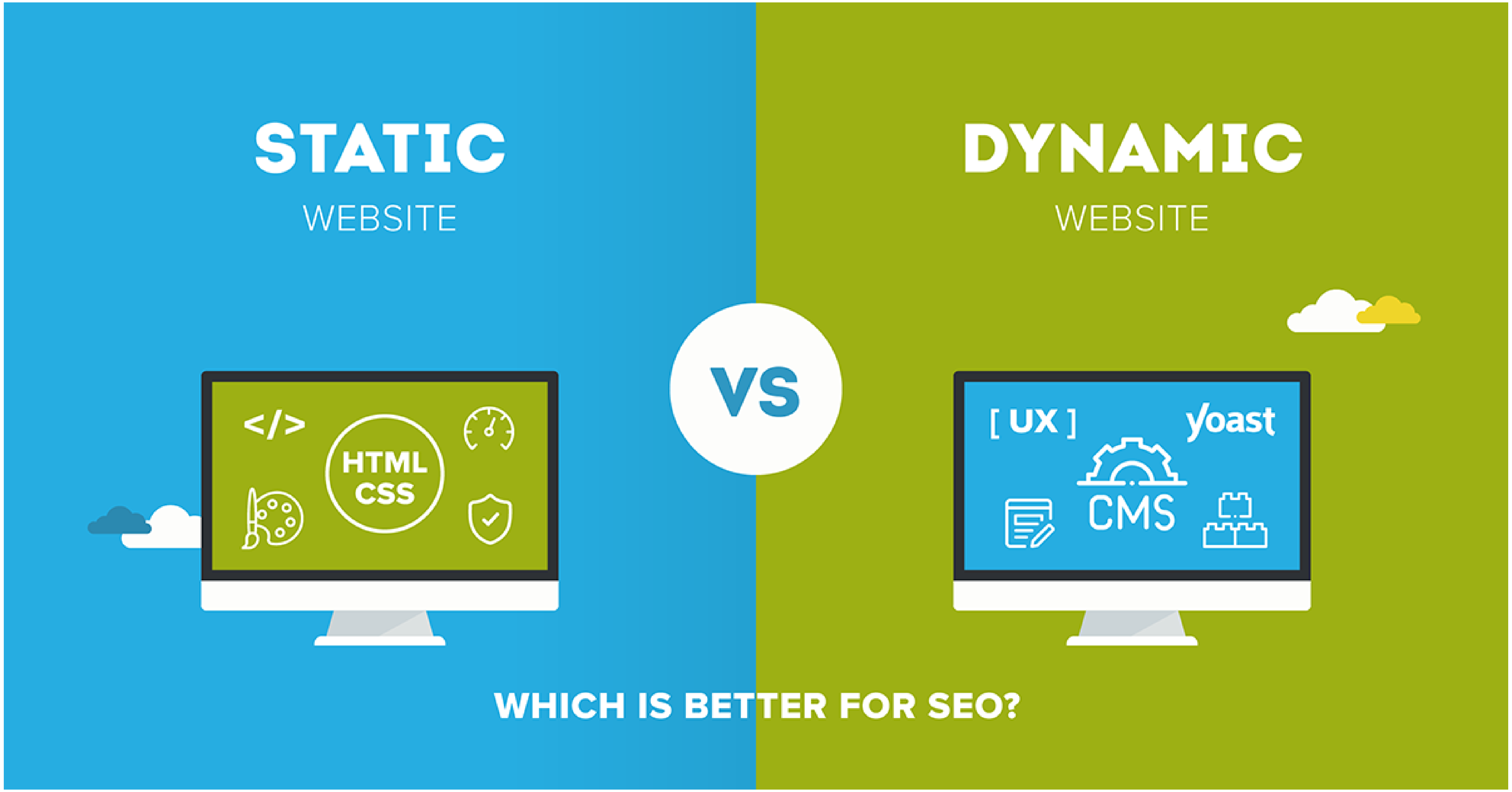
Happy Coding

Hello world!

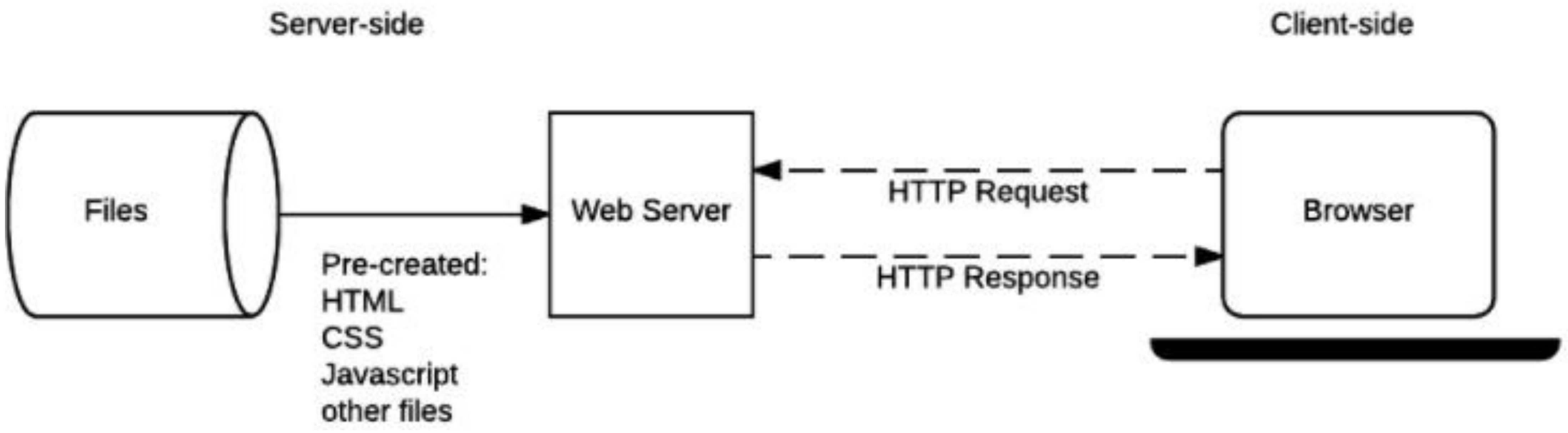
HTTP vs HTTPs



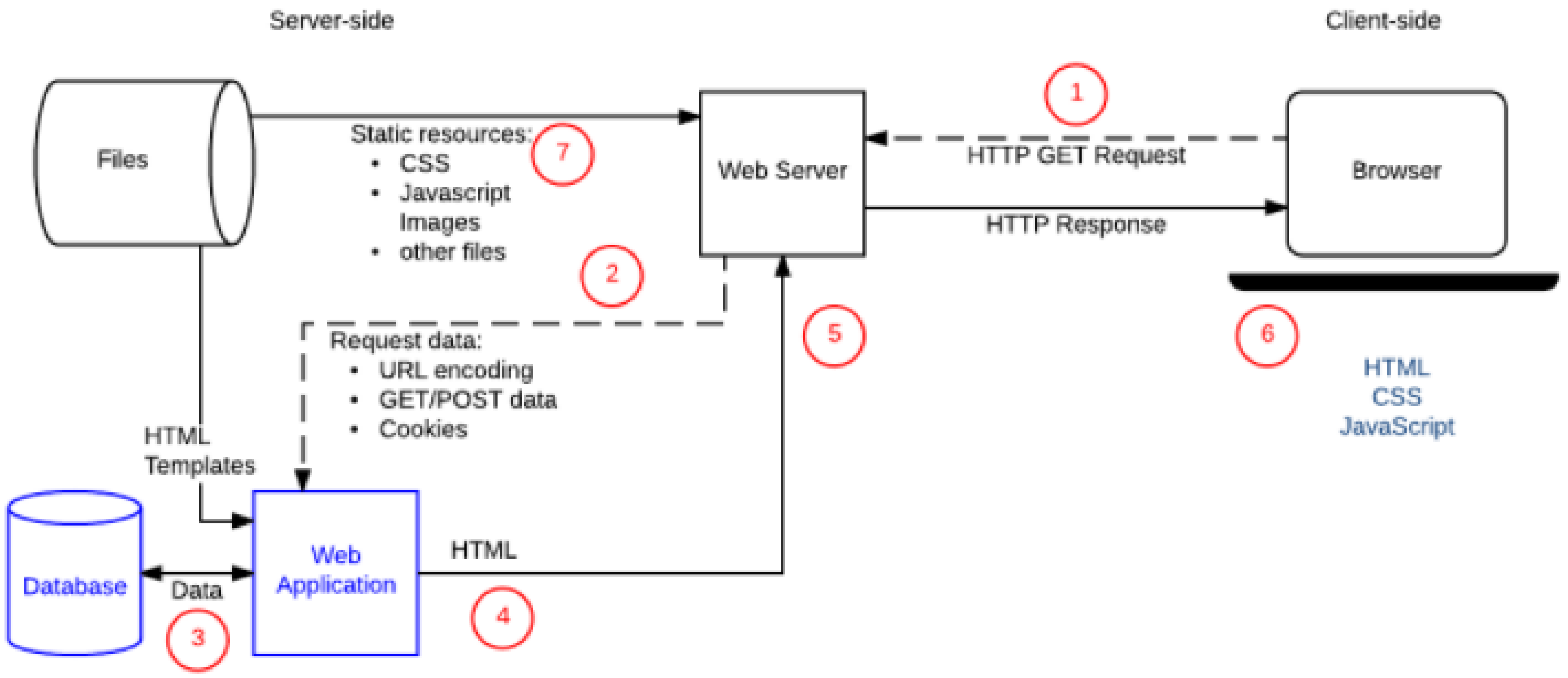
Static vs Dynamic Web page



Static Web page

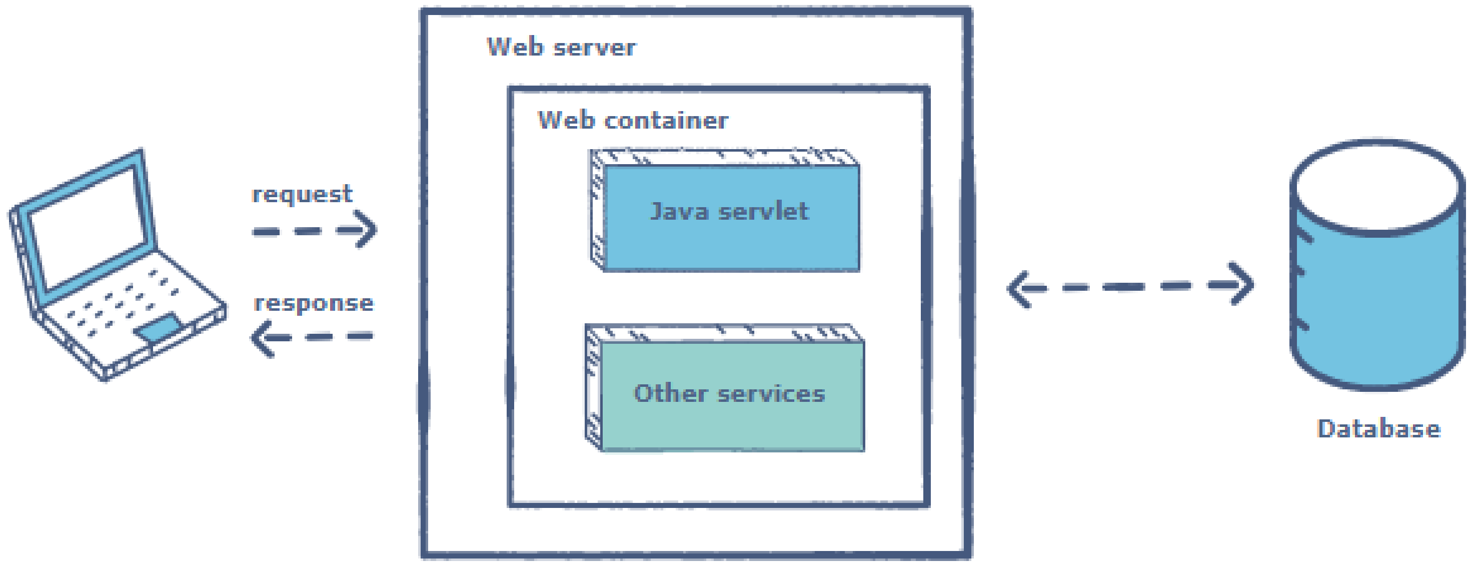


Dynamic Web page

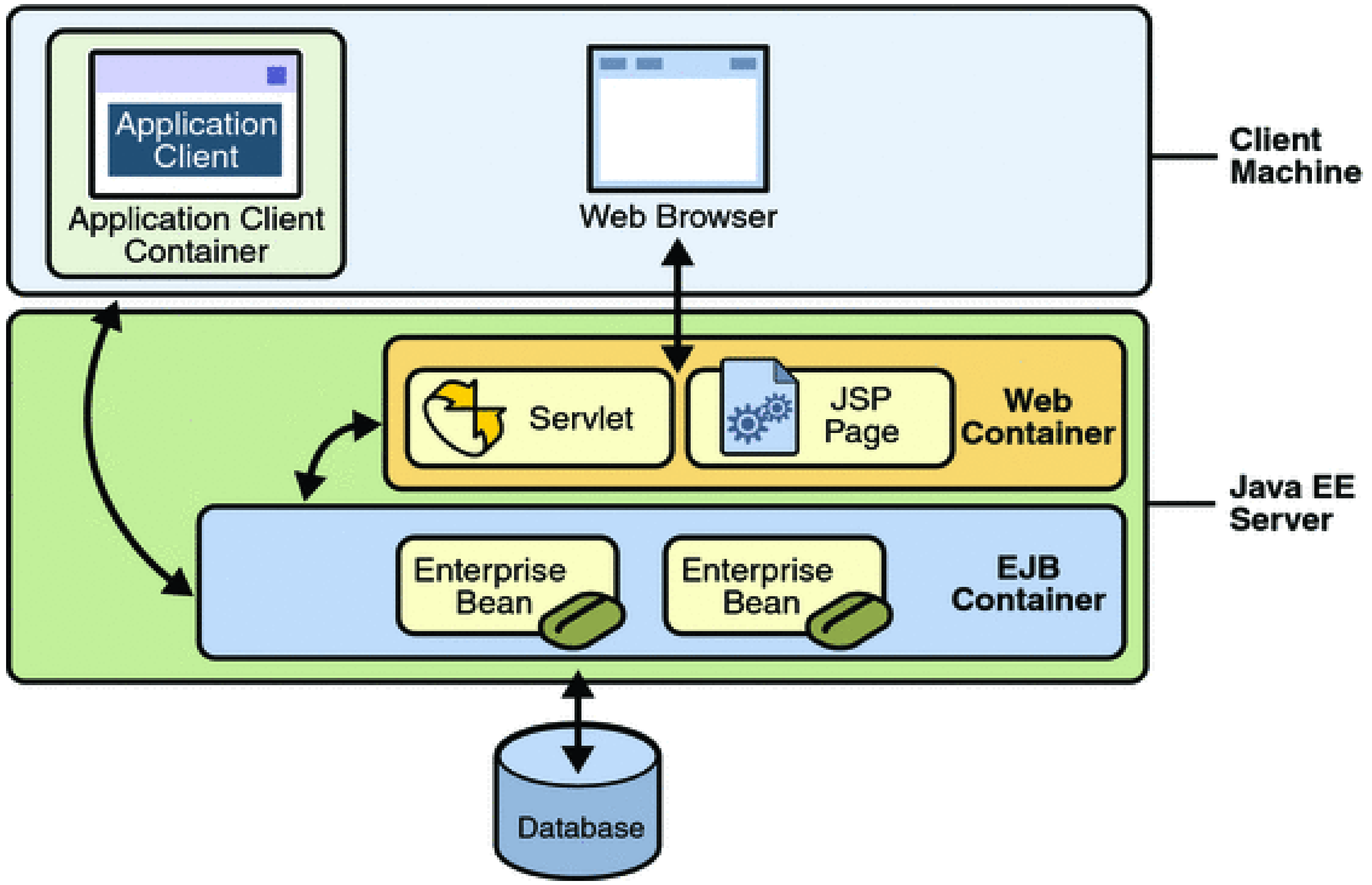


What is a web container?

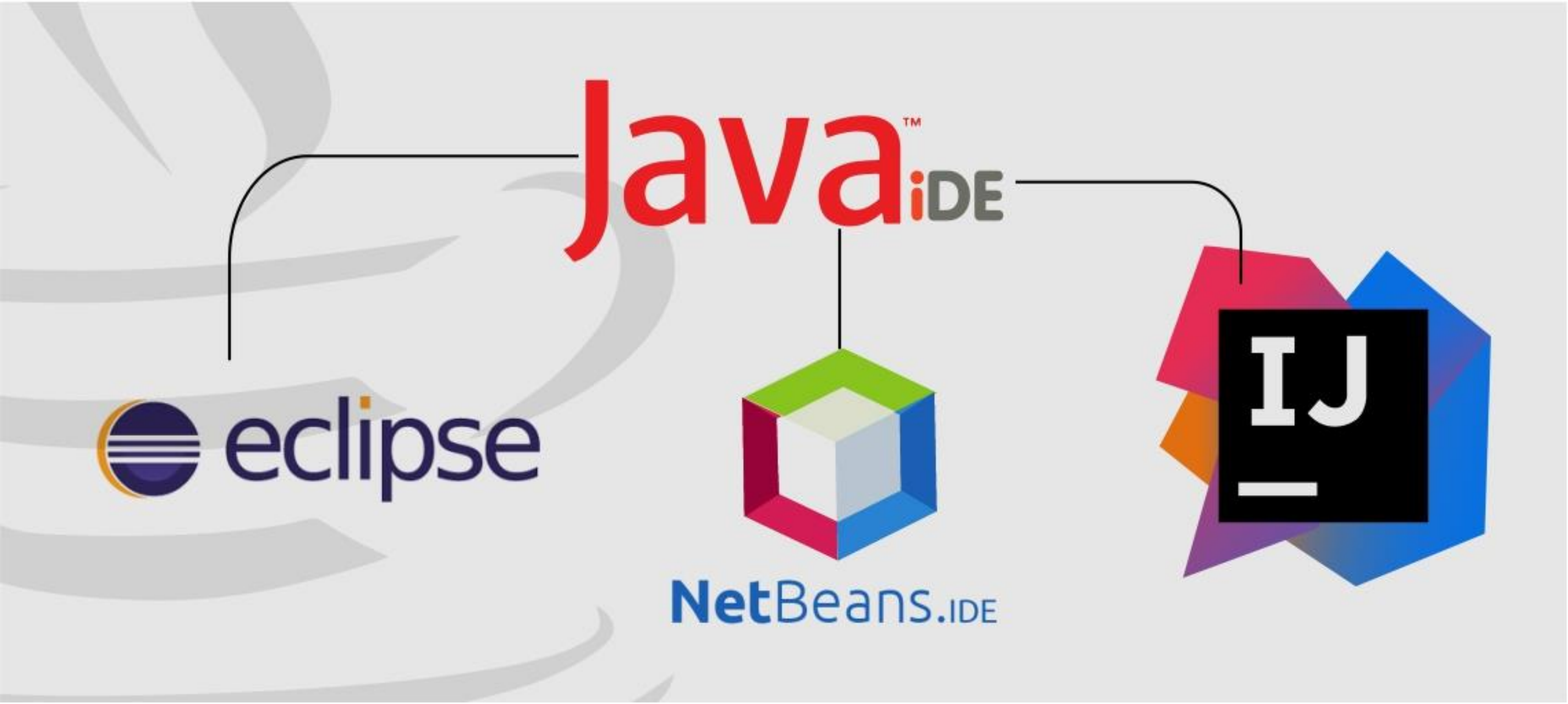
- The web container is the interface between web components and the web server. A web component can be a servlet, a Java Server Faces or a JSP page.
- The container manages the component's lifecycle, dispatches requests to application components, and provides interfaces to context data, such as information about the current request.



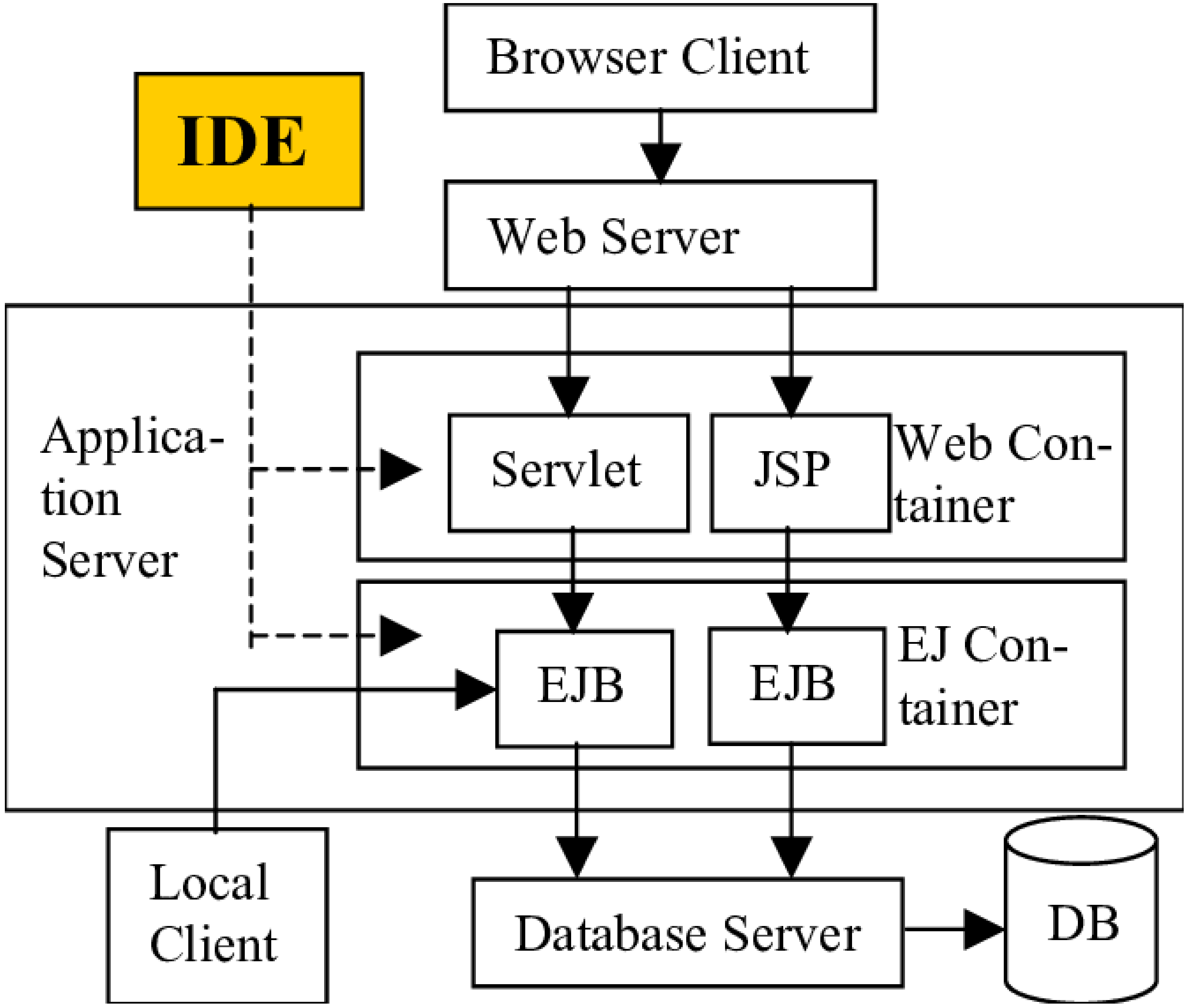
What is a web container?



J2EE IDE



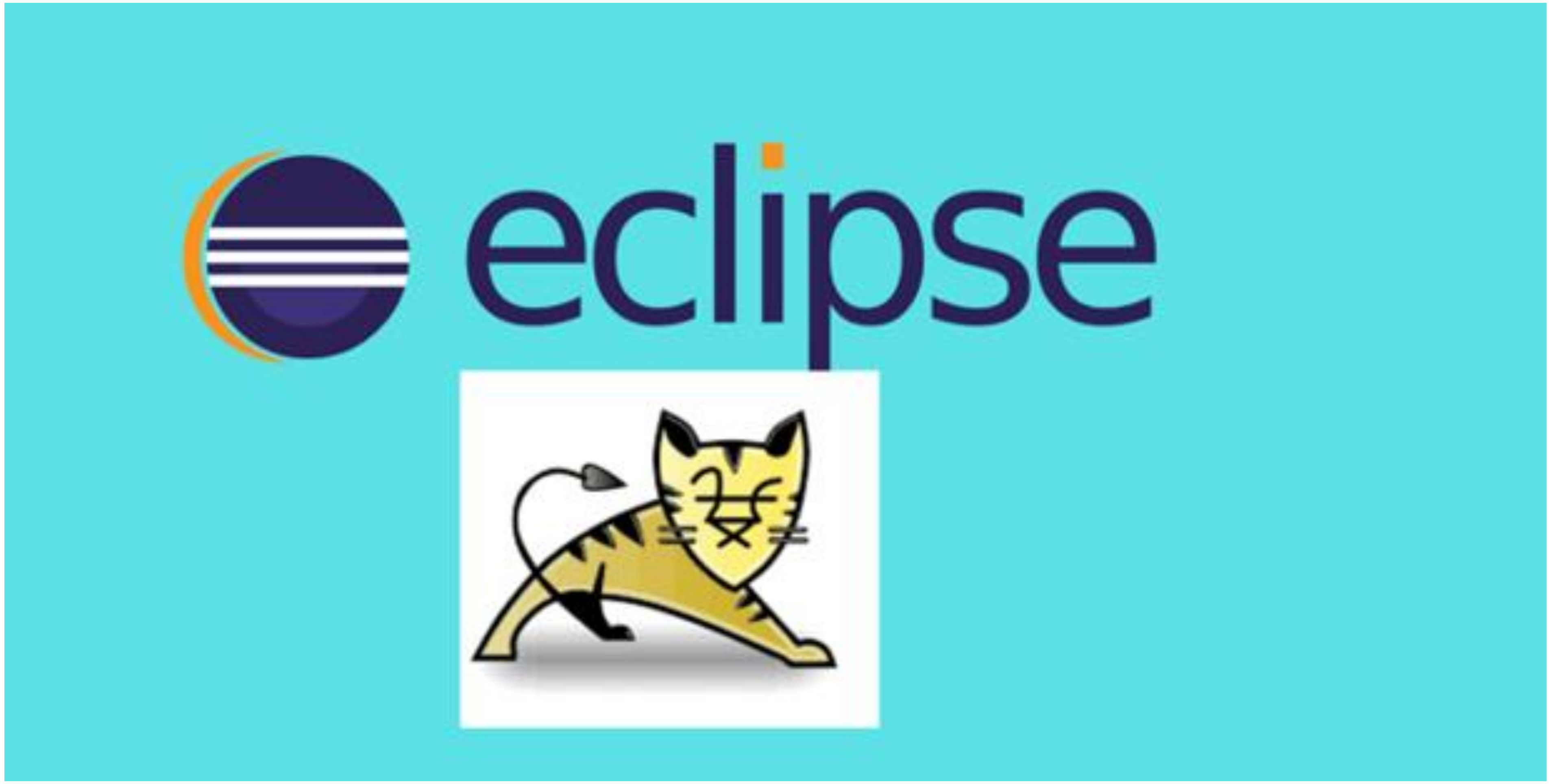
J2EE IDE Helps



How to Configure Tomcat with NetBeans?

- See this Link: https://www.youtube.com/watch?v=psvuLBVeTzQ.

How to Configure Tomcat with Eclipse?

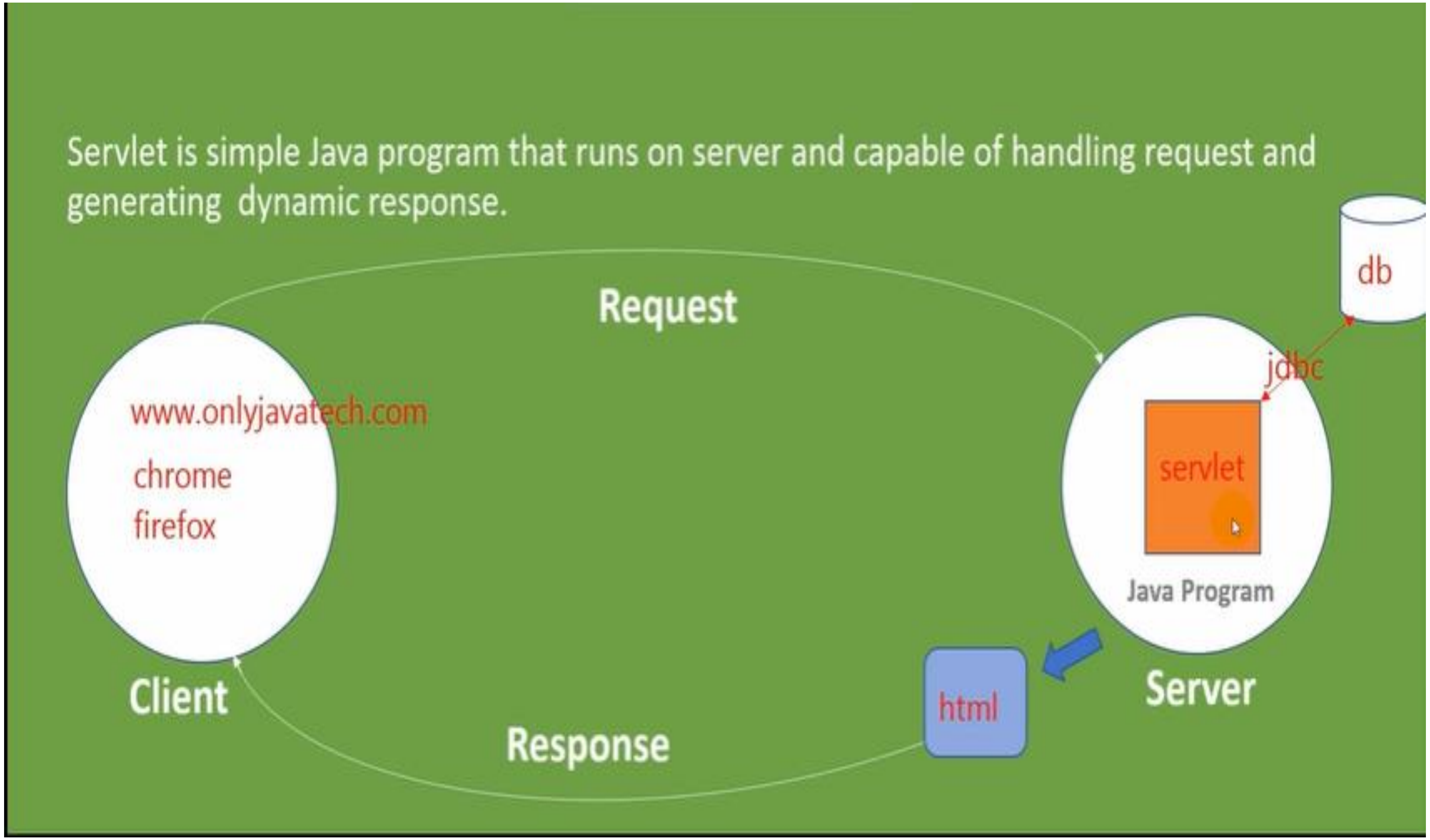


- See this Link: https://www.youtube.com/watch?v=PH-bK3g2YmU.

Servlets

- Servlet is a simple java program that runs on server and capable of handling request and generating dynamic response.

Servlets

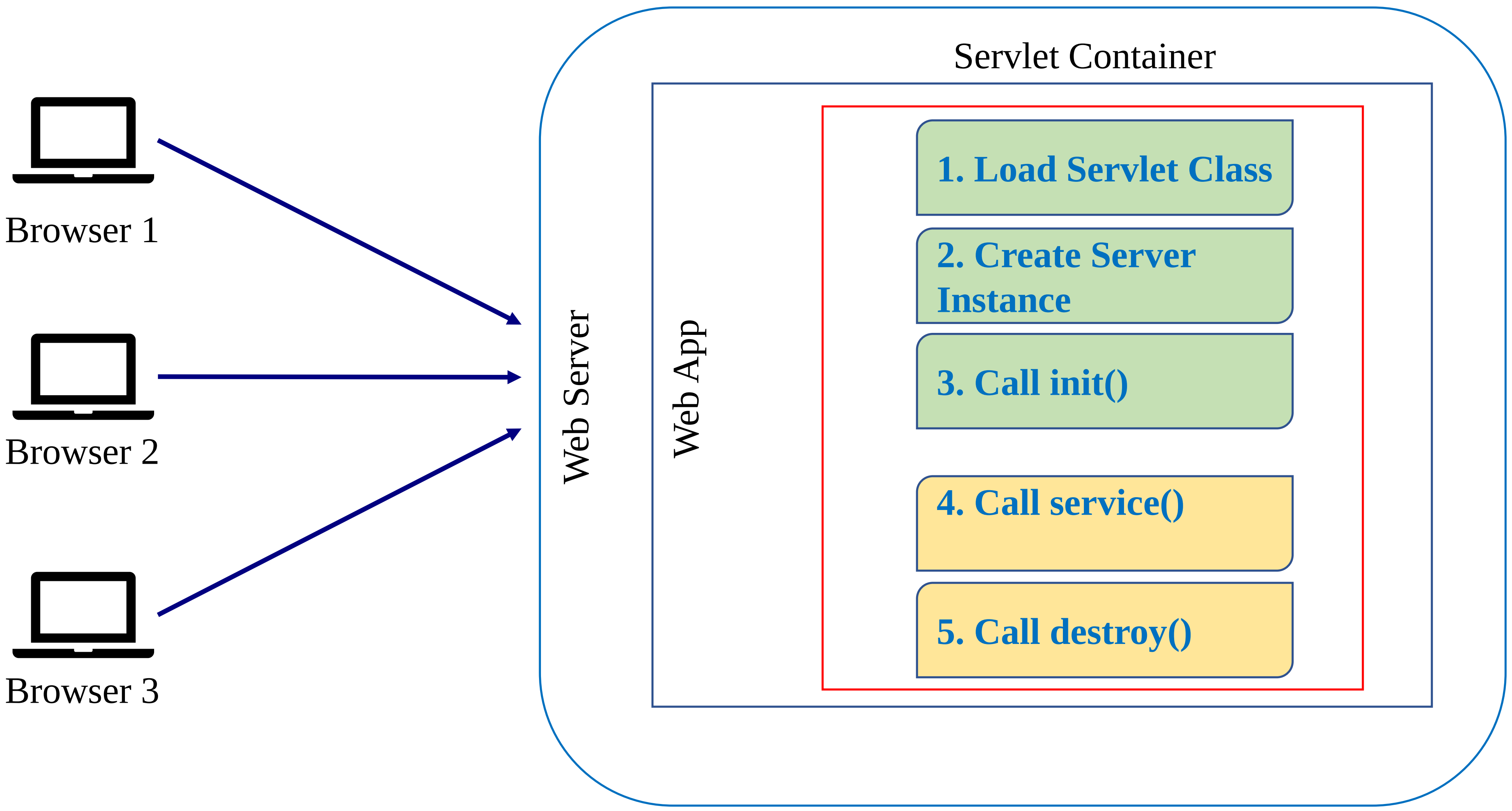

Servlet is simple Java program that runs on server and capable of handling request and generating dynamic response.

What is a Servlet Container?

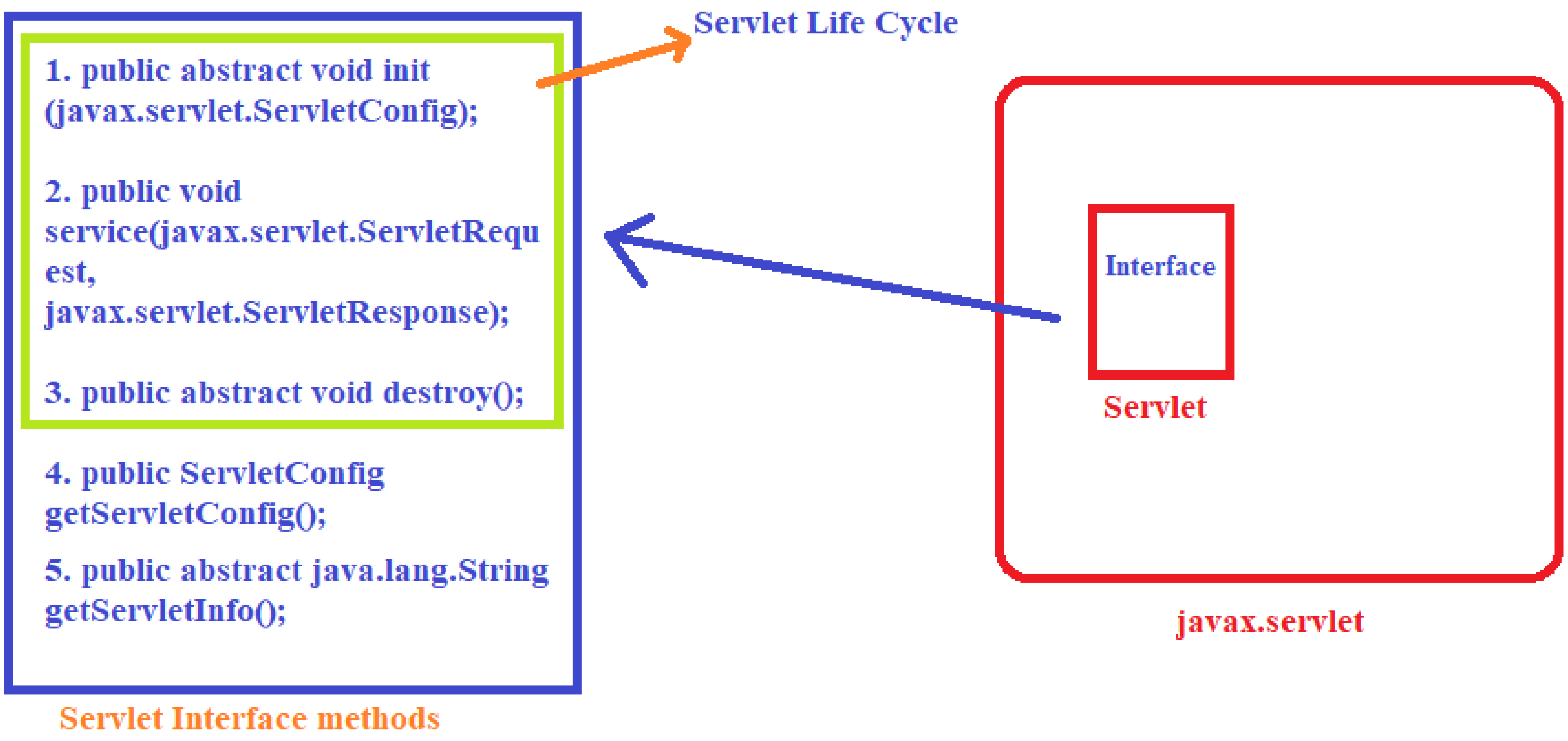
Servlets

- Servlet container is the essential part of the web server that interacts with the java servlets.
- Servlet Container communicates between client Browsers and the servlets.
- Servlet Container managing the life cycle of servlet.
- Servlet container loading the servlets into memory, initializing and invoking servlet methods and to destroy them.
- There are a lot of Servlet Containers like
  - ✓ Jboss
  - ✓ Apache Tomcat
  - ✓ WebLogic
  - ✓ etc.

How does this Servlet Container work?



How to create servlet using javax.servlet.Servlet?



1. public abstract void init (javax.servlet.ServletConfig);

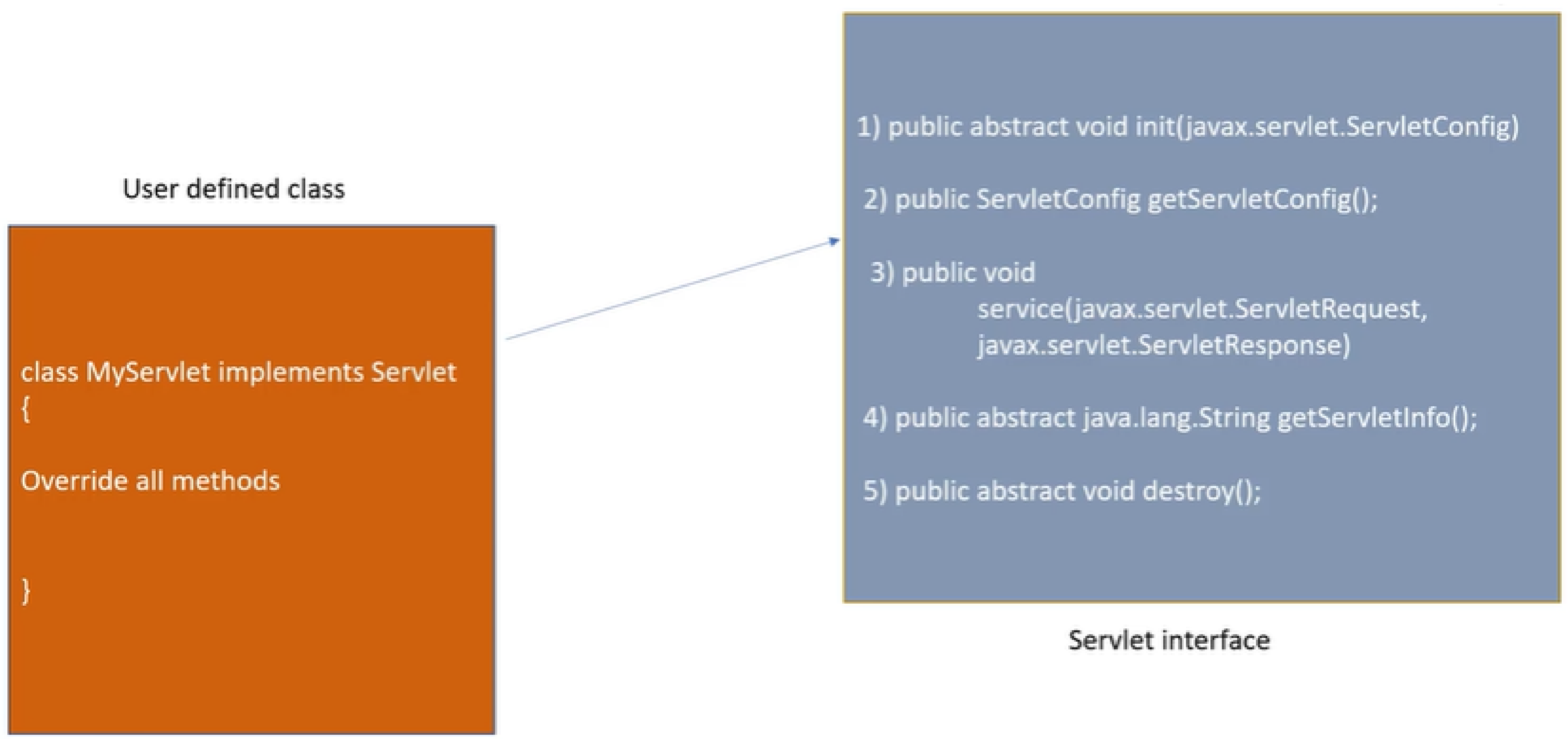
2. public void service(javax.servlet.ServletRequest, javax.servlet.ServletResponse);

3. public abstract void destroy();

4. public ServletConfig getServletConfig();

5. public abstract java.lang.String getServletInfo();

How to create servlet using javax.servlet.Servlet?



```
class MySerlvet implements Servlet
{

Override all methods


}
```

1) public abstract void init(javax.servlet.ServletConfig)

2) public ServletConfig getServletConfig();

3) public void
        service(javax.servlet.ServletRequest,
        javax.servlet.ServletResponse)

4) public abstract java.lang.String getServletInfo();

5) public abstract void destroy();

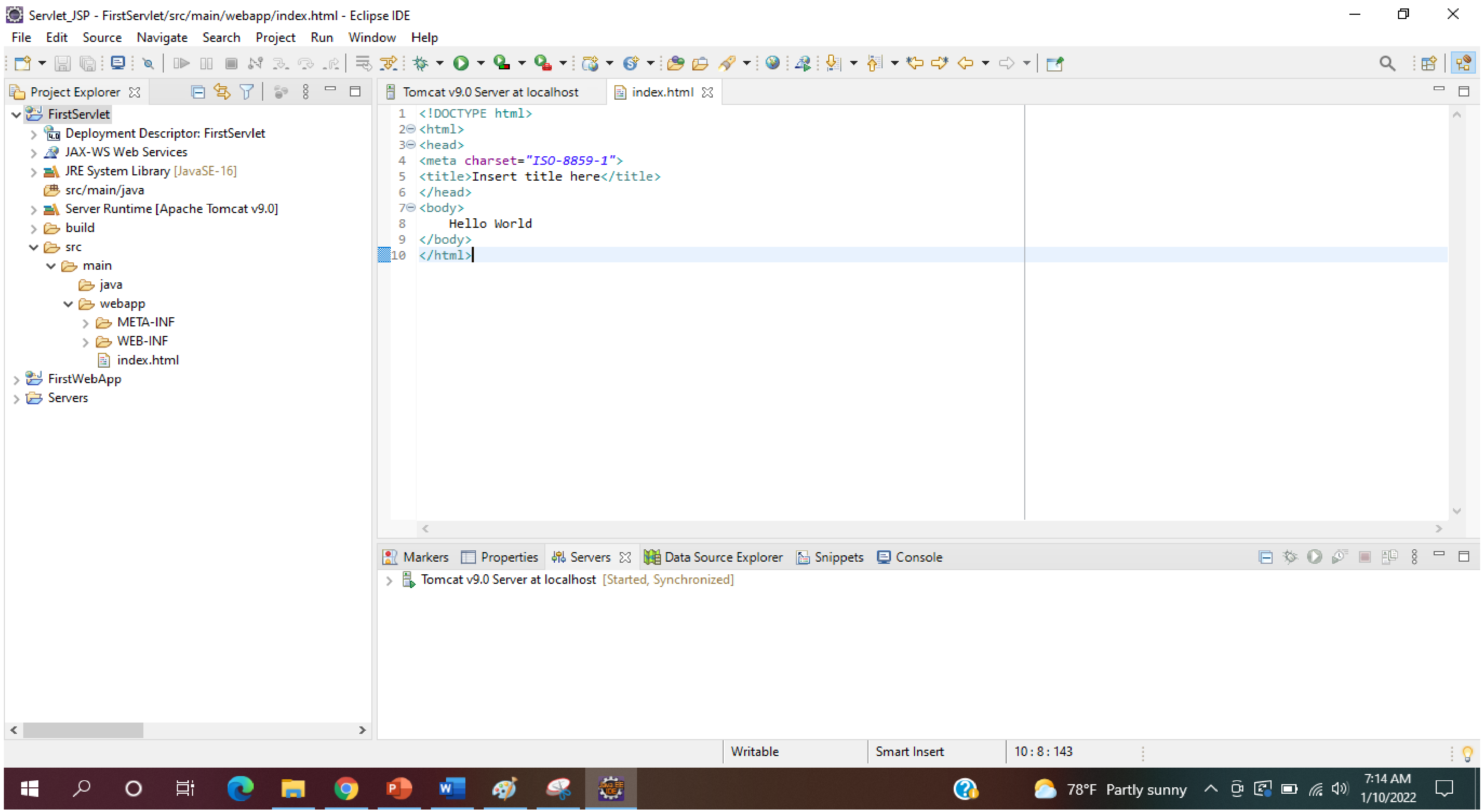
My First Servlets

- ❑ Basic java Package requires
  - ✓ package com.servlets;
  - ✓ import javax.servlet.*;
  - ✓ import java.io.IOException;

- ❑ User-define servlet class “FirstServlet.java”

```
public class FirstServlet implements Servlet
{
        public void init(ServletConfig conf){}
        public void service(ServletRequest req,
ServletResponse res) {}
        public void destroy(){}
        public ServletConfig getServletConfig() {}
        public String getServletInfo() {}
}
```

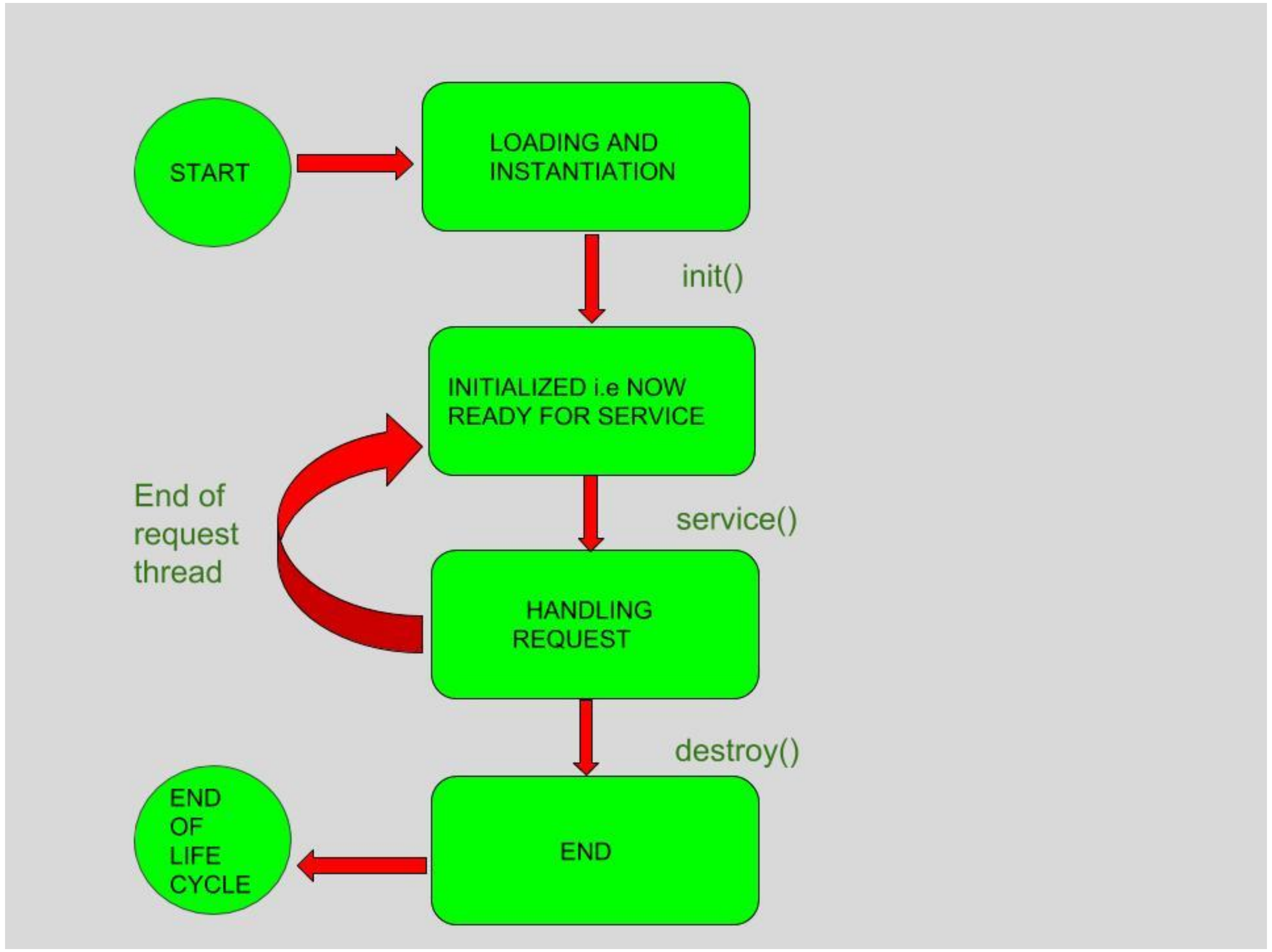
Creating First Servlet using Servlet interface in Eclipse


Servlet_JSP - FirstServlet/src/main/webapp/index.html - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
Project Explorer
FirstServlet
Deployment Descriptor: FirstServlet
JAX-WS Web Services
JRE System Library [JavaSE-16]
src/main/java
Server Runtime [Apache Tomcat v9.0]
build
src
main
java
webapp
META-INF
WEB-INF
index.html
FirstWebApp
Servers
Tomcat v9.0 Server at localhost
index.html
Markers Properties Servers Data Source Explorer Snippets Console
Writable
Smart Insert
10 : 8 : 143

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Hello World
</body>
</html>
```

Tomcat v9.0 Server at localhost [Started, Synchronized]

Life Cycle of a Servlet (Servlet Life Cycle)



public void init (ServletConfig conf) {}

- The Servlet container calls the init method, once the Instance for the servlet is created.
- The servlet won't process any service request until the init method finish.
- The servlet container cannot place the servlet into service if the init method
  - a. Throws a ServletException
  - b. Does not return within a time period defined by the Web server

```
ServletConfig conf; //Creating a ServletConfig object
    public void init(ServletConfig conf) //Overwridding init()
    {
        this.conf = conf; //Passing current object reference
        System.out.println("Creating Object:...");//Object Printing
    }
```

public void service(ServletRequest req, ServletResponse res) {}

- ❑ Servlet calls the service method to process client requests. Whenever a request for a servlet comes, the server spawns a new thread and calls service method of the servlet.
- ❑ The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

```
public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
      System.out.println("Servicing:......");
}
```

public void destroy(){}

- Whenever the servlet is out of service, then the servlet container calls the destroy method of this servlet.
- This method is only called once all threads within the servlet's service method have exited or after a timeout period has passed.
- After the servlet container calls this method, it will not call the service method again on this servlet.
- Destroy() method generally contains code to free any resources.

```
public void destroy()
   {
      System.out.println("Servlet Object is going to destroy....");
   }
```

Non-Life Cycle Methods

- public ServletConfig getServletConfig()
  - Returns a ServletConfig object, which contains initialization and startup parameters for this servlet. A servlet configuration object used by a servlet container to pass information to a servlet during initialization.
- public String getServletInfo()
  - Returns information about the servlet, such as author, version, and copyright.

```
public ServletConfig getServletConfig()
  {
     return this.conf;
  }
```

```
public String getServletInfo()
  {
     return "This servlet getServletInfo ";
  }
```

Non-Life Cycle Methods

- public ServletConfig getServletConfig()
  - Returns a ServletConfig object, which contains initialization and startup parameters for this servlet. A servlet configuration object used by a servlet container to pass information to a servlet during initialization.
- public String getServletInfo()
  - Returns information about the servlet, such as author, version, and copyright.

```
public ServletConfig getServletConfig()
   {
      return this.conf;
   }
```

```
public String getServletInfo()
   {
      return "This servlet getServletInfo ";
   }
```

Mapping with web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
           http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"  version="3.1">

<session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

What is Deployment Descriptor (web.xml)?

- A web application's deployment descriptor describes the classes, resources and configuration of the application and how the web server uses them to serve web requests.
- When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the code that ought to handle the request.
- The deployment descriptor is a file named web.xml. It resides in the app's WAR under the WEB-INF/ directory.
- The file is an XML file whose root element is <web-app>.

What is Deployment Descriptor (web.xml)?

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee     http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd"
        version="3.1">
        <servlet>
                <servlet-name> comingsoon </servlet-name>
                <servlet-class> mysite.server.ComingSoonServlet </servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>comingsoon</servlet-name>
                <url-pattern>/FirstServlet</url-pattern>
        </servlet-mapping>
</web-app>
```

Servlet Declaration & Mapping in web.xml

- Servlet Declaration:

```
<servlet>
        <servlet-name>first</servlet-name>
        <servlet-class>com.servlets.FirstServlet</servlet-class>
</servlet>
```

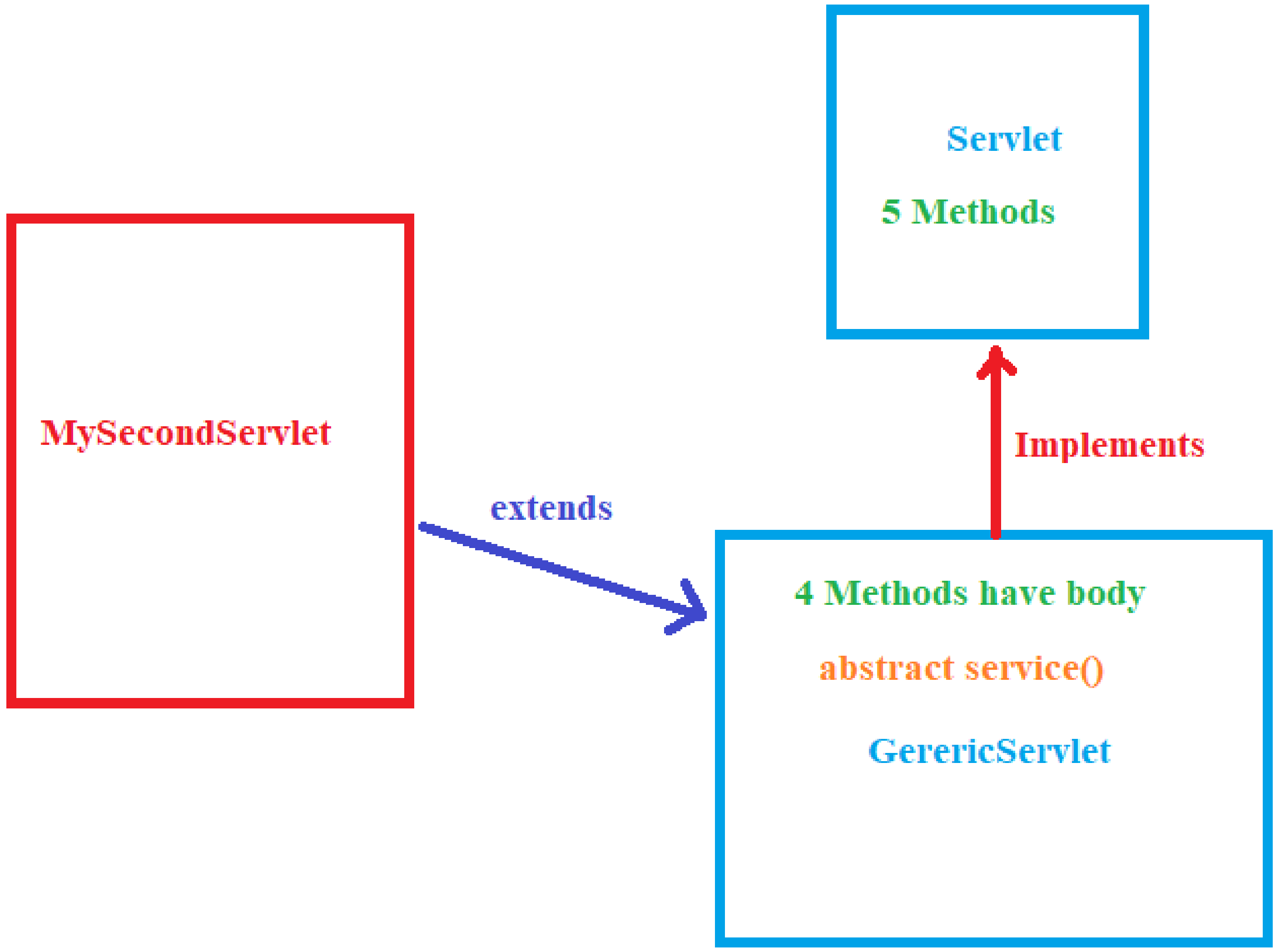
- Servlet Mapping:

```
<servlet-mapping>
        <servlet-name> first</servlet-name>
        <url-pattern>/MyFirstWeb</url-pattern>
</servlet-mapping>
```

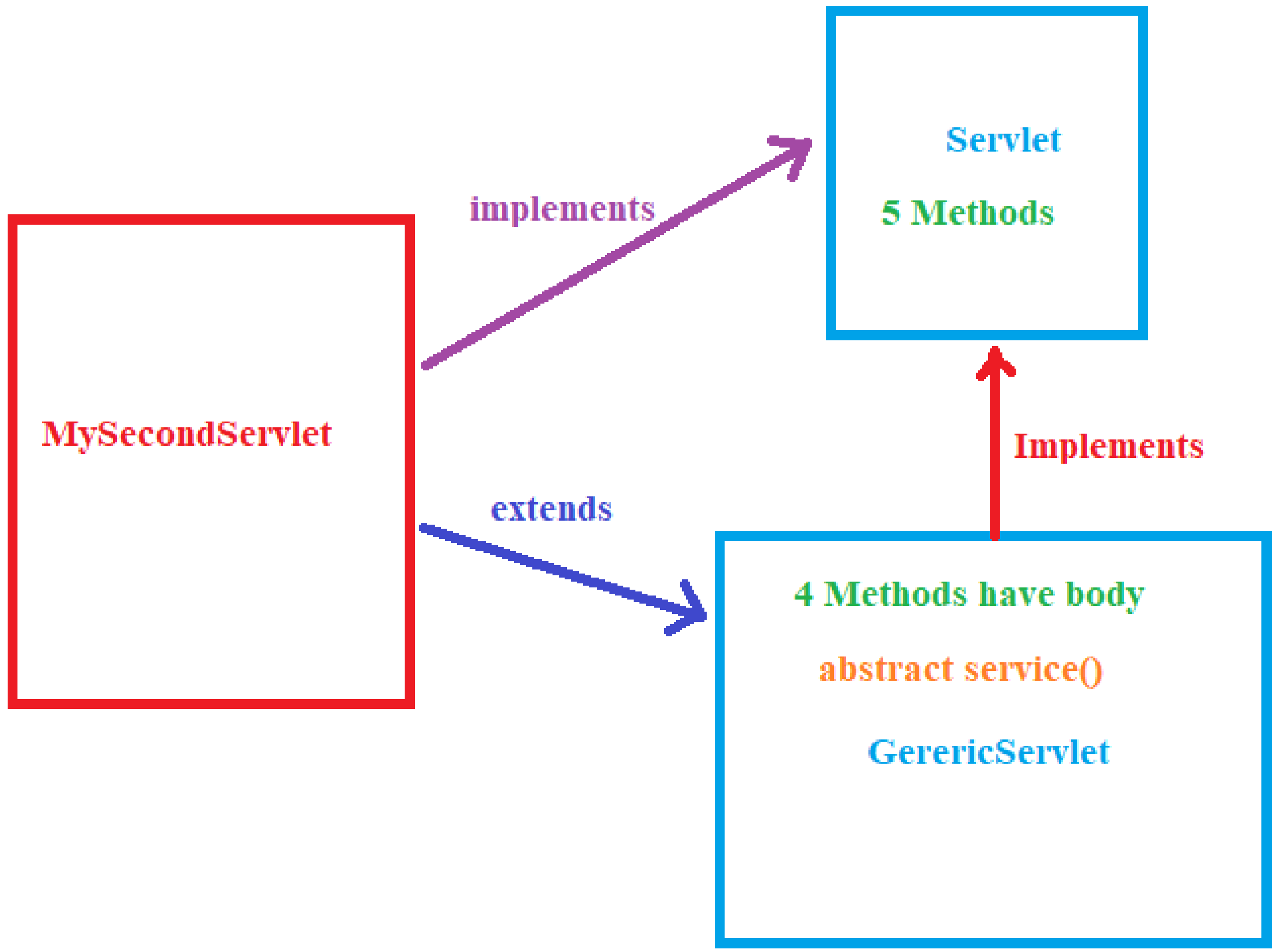
Dynamic Servlet Response

```
public void service (ServletRequest req, ServletResponse res) throws ServletException,
IOException
  {
     System.out.println("Servicing:......");
     //Set the content-Type of the response
     res.setContentType("text/html");
     PrintWriter out = res.getWriter();
     out.println("<h1>This is my output from servlet method</h1>");
     out.println("<h1>Today's Date & Time : "+new Date().toString()+"</h1>");
  }
```

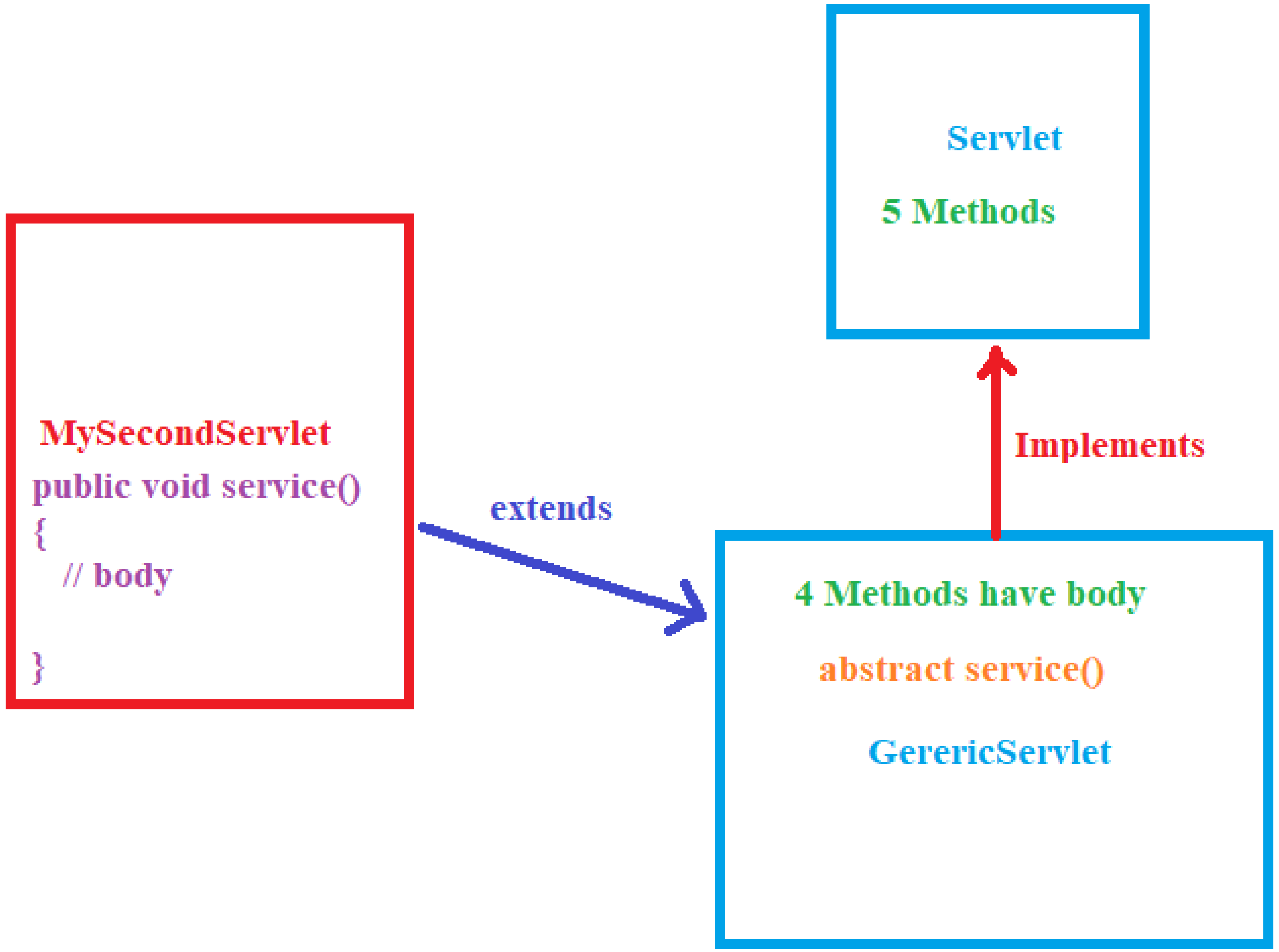
Creating Servlet using GenericServlet Class



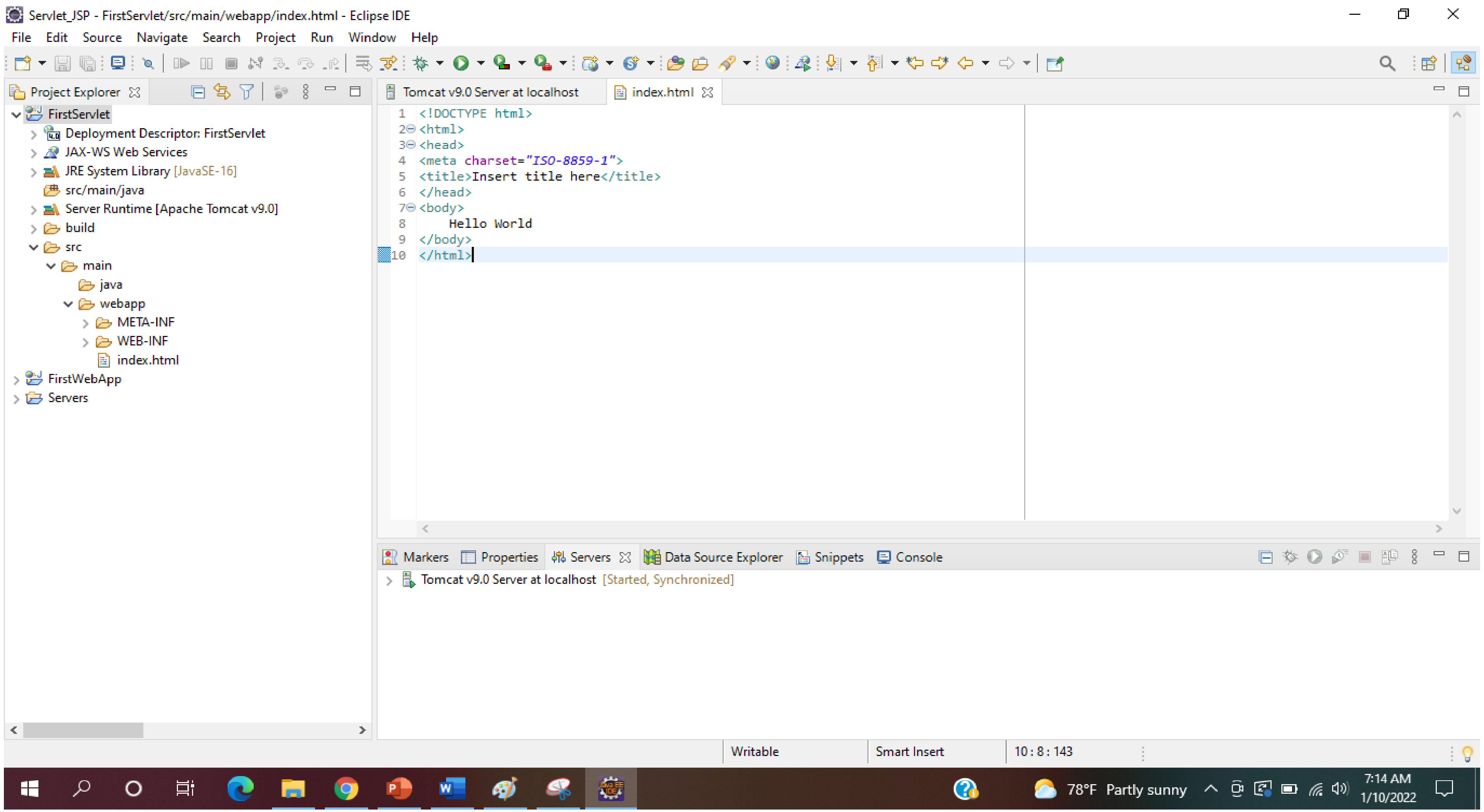
Creating Servlet using GenericServlet Class



Creating Servlet using GenericServlet Class



Creating Second Servlet using GenericServlet Class



Servlet_JSP - FirstServlet/src/main/webapp/index.html - Eclipse IDE

Project Explorer

- FirstServlet
  - Deployment Descriptor: FirstServlet
  - JAX-WS Web Services
  - JRE System Library [JavaSE-16]
  - src/main/java
  - Server Runtime [Apache Tomcat v9.0]
  - build
  - src
    - main
      - java
      - webapp
        - META-INF
        - WEB-INF
        - index.html
- FirstWebApp
- Servers

Tomcat v9.0 Server at localhost | index.html

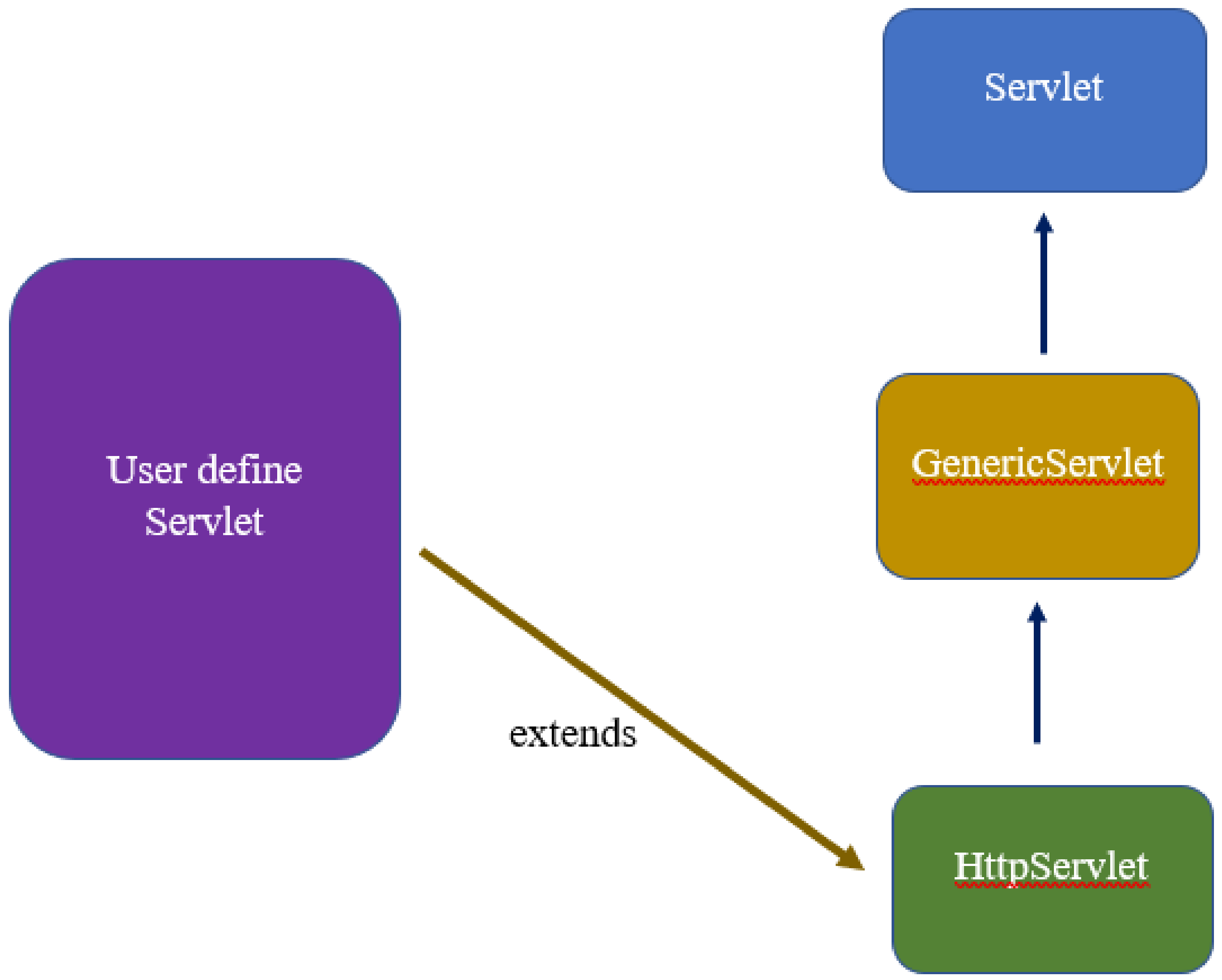
```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Hello World
</body>
</html>
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Console

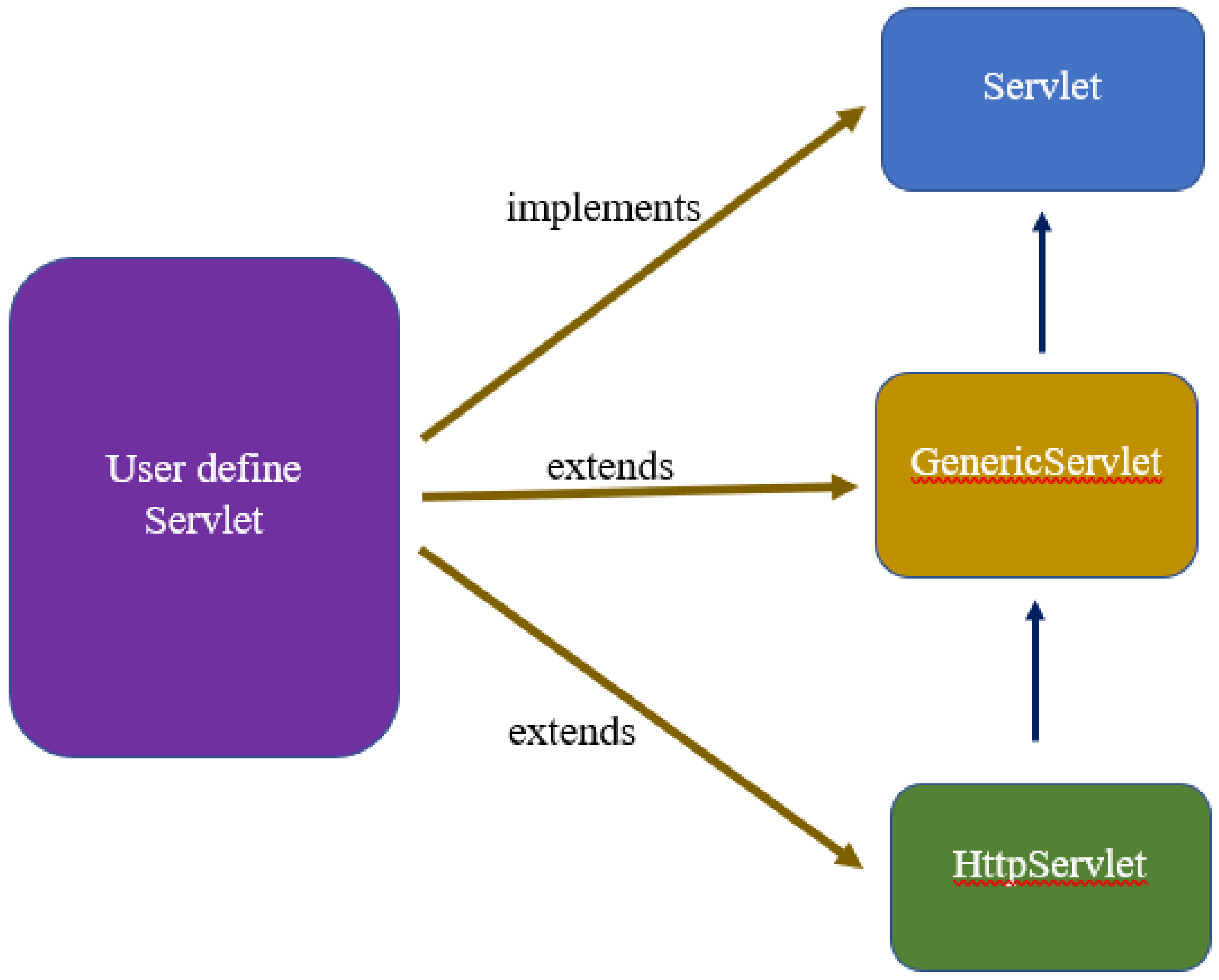
Tomcat v9.0 Server at localhost [Started, Synchronized]

Writable | Smart Insert | 10 : 8 : 143

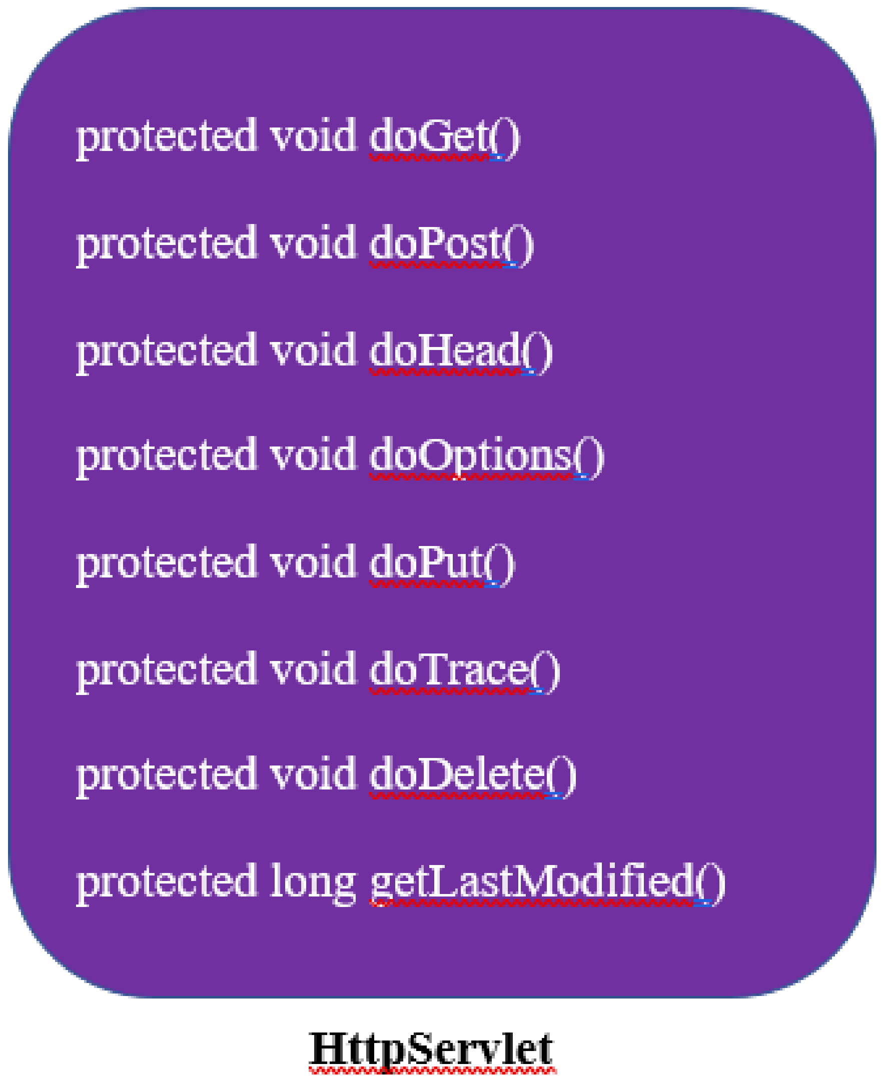
Creating Servlet using HttpServlet Class



Creating Servlet using HttpServlet Class



HttpServlet Class Methods



protected void doGet()

protected void doPost()

protected void doHead()

protected void doOptions()

protected void doPut()

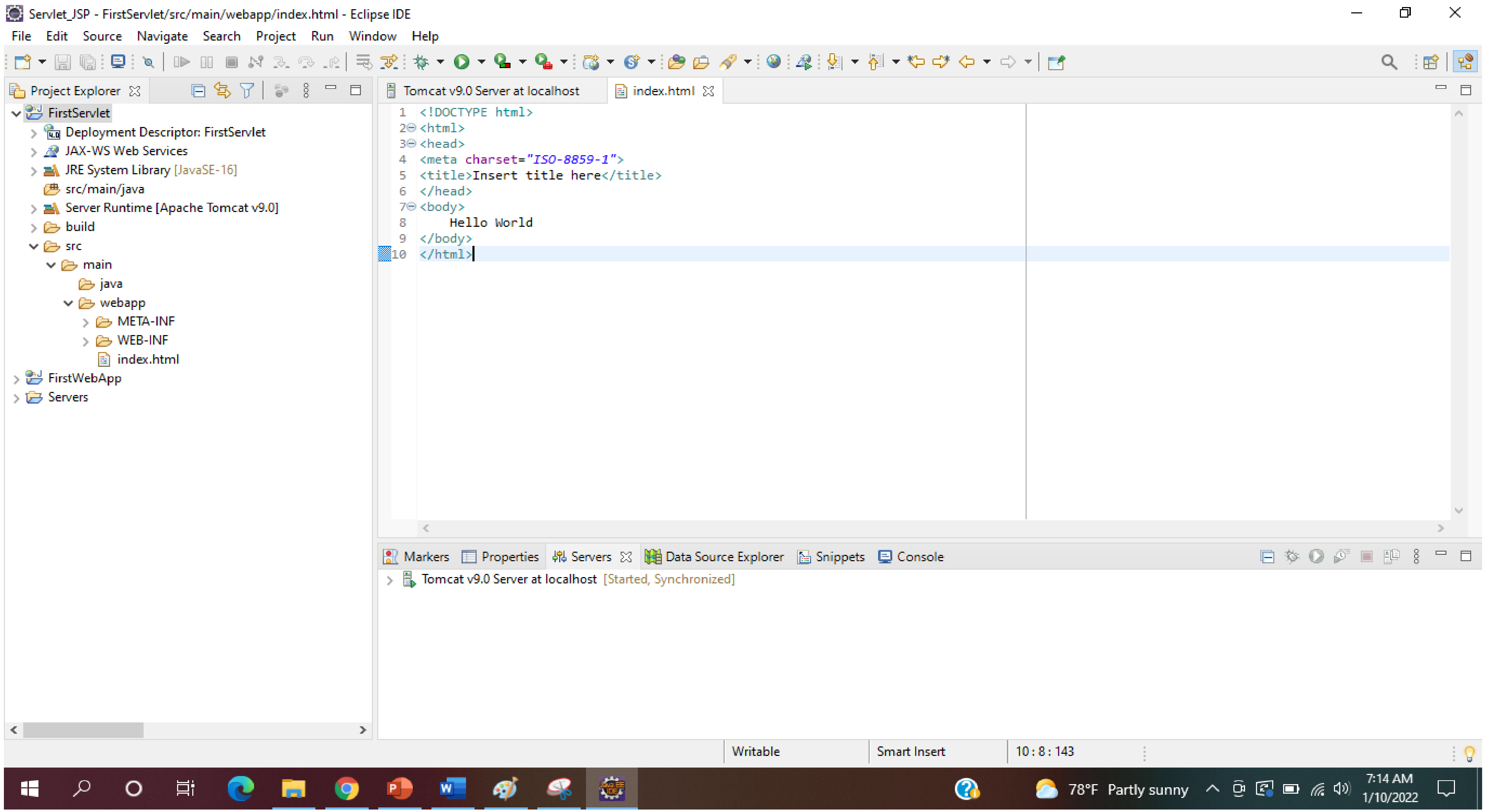
protected void doTrace()

protected void doDelete()

protected long getLastModified()

HttpServlet

Creating Third Servlet using HttpServlet Class



Servlet_JSP - FirstServlet/src/main/webapp/index.html - Eclipse IDE

File Edit Source Navigate Search Project Run Window Help

Project Explorer

- FirstServlet
  - Deployment Descriptor: FirstServlet
  - JAX-WS Web Services
  - JRE System Library [JavaSE-16]
  - src/main/java
  - Server Runtime [Apache Tomcat v9.0]
  - build
  - src
    - main
      - java
      - webapp
        - META-INF
        - WEB-INF
        - index.html
- FirstWebApp
- Servers

Tomcat v9.0 Server at localhost | index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Hello World
</body>
</html>
```

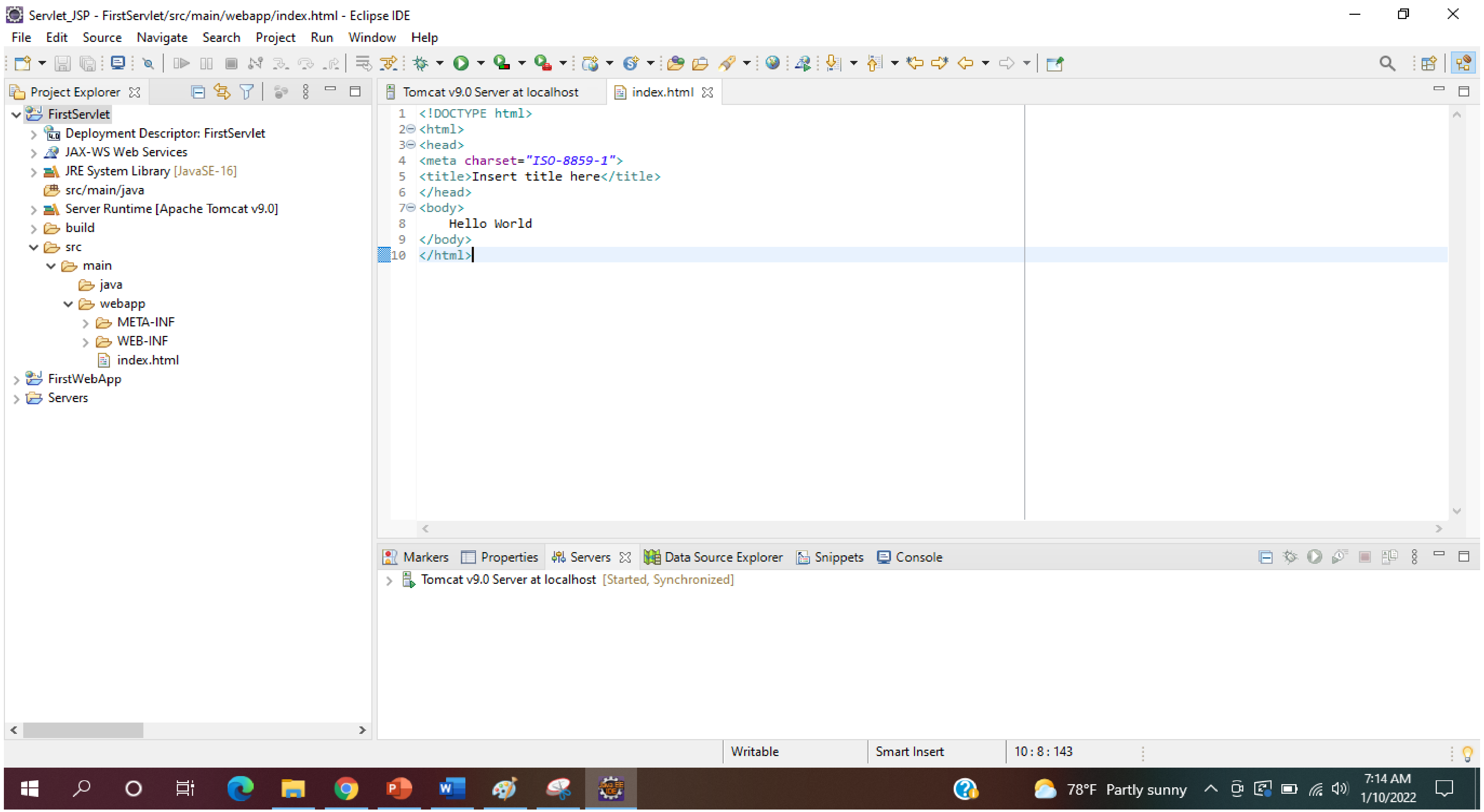
Markers | Properties | Servers | Data Source Explorer | Snippets | Console

Tomcat v9.0 Server at localhost [Started, Synchronized]

Writable | Smart Insert | 10 : 8 : 143

78°F Partly sunny 7:14 AM 1/10/2022

Creating Fourth Servlet using doGet()

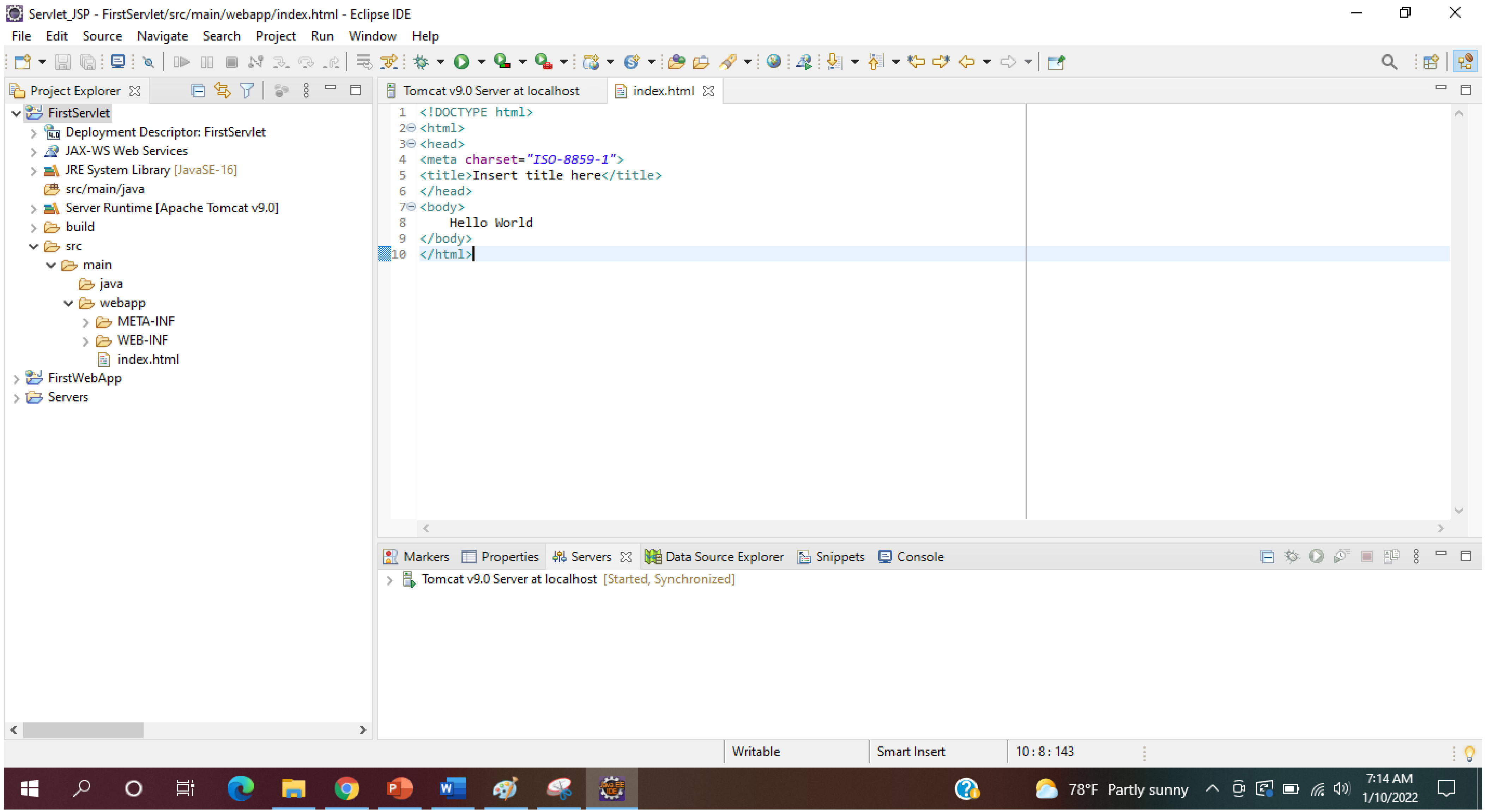


Servlet_JSP - FirstServlet/src/main/webapp/index.html - Eclipse IDE

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Hello World
</body>
</html>
```

Tomcat v9.0 Server at localhost [Started, Synchronized]

Creating Fifth Servlet using doPost()

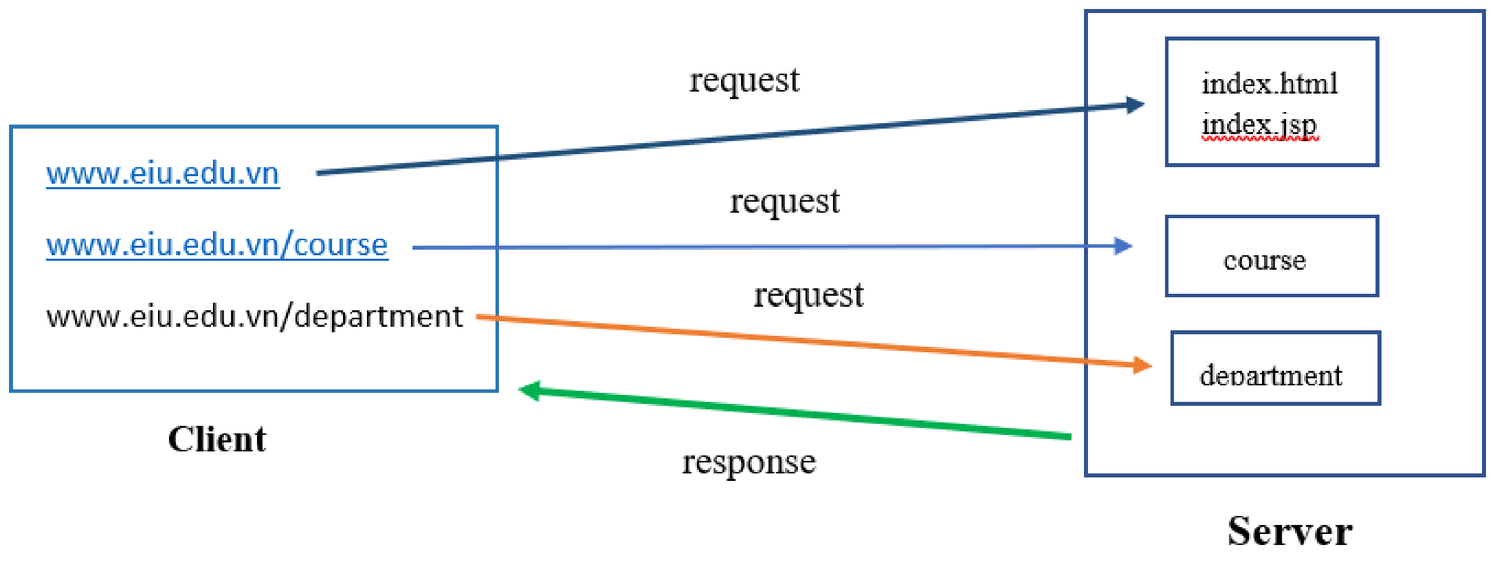


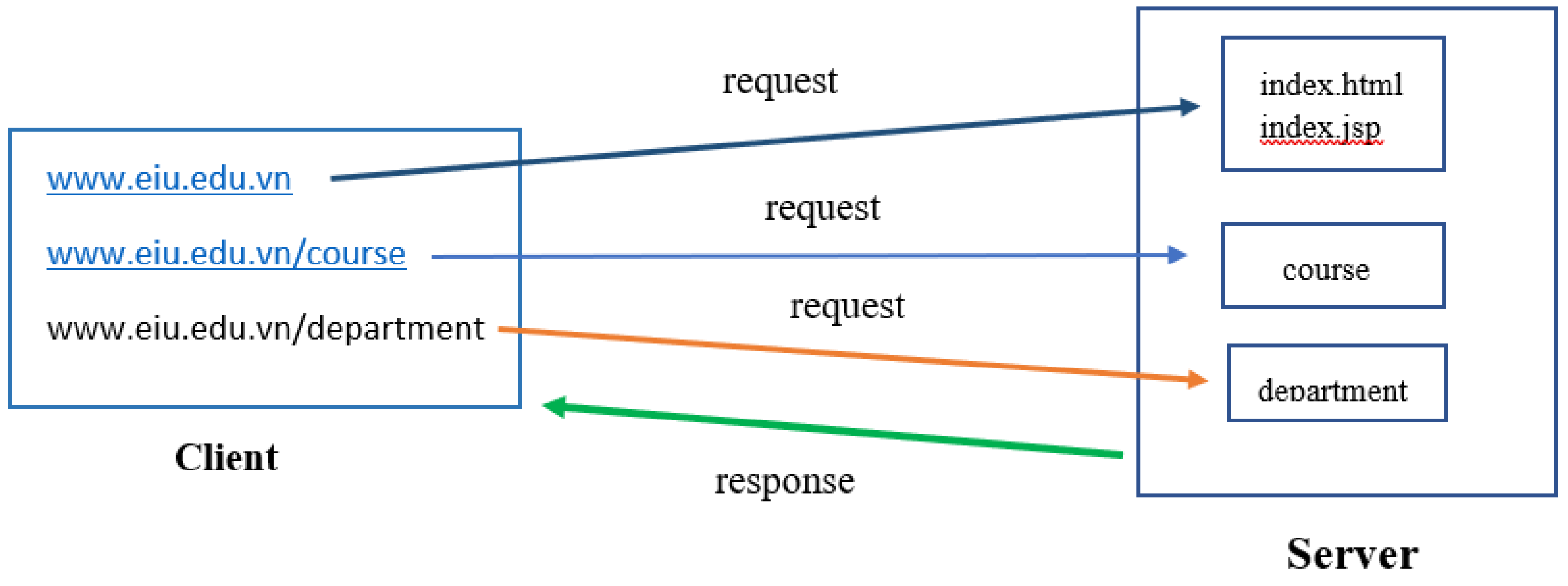
Servlet_JSP - FirstServlet/src/main/webapp/index.html - Eclipse IDE

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Hello World
</body>
</html>
```

Tomcat v9.0 Server at localhost [Started, Synchronized]

Welcome File List



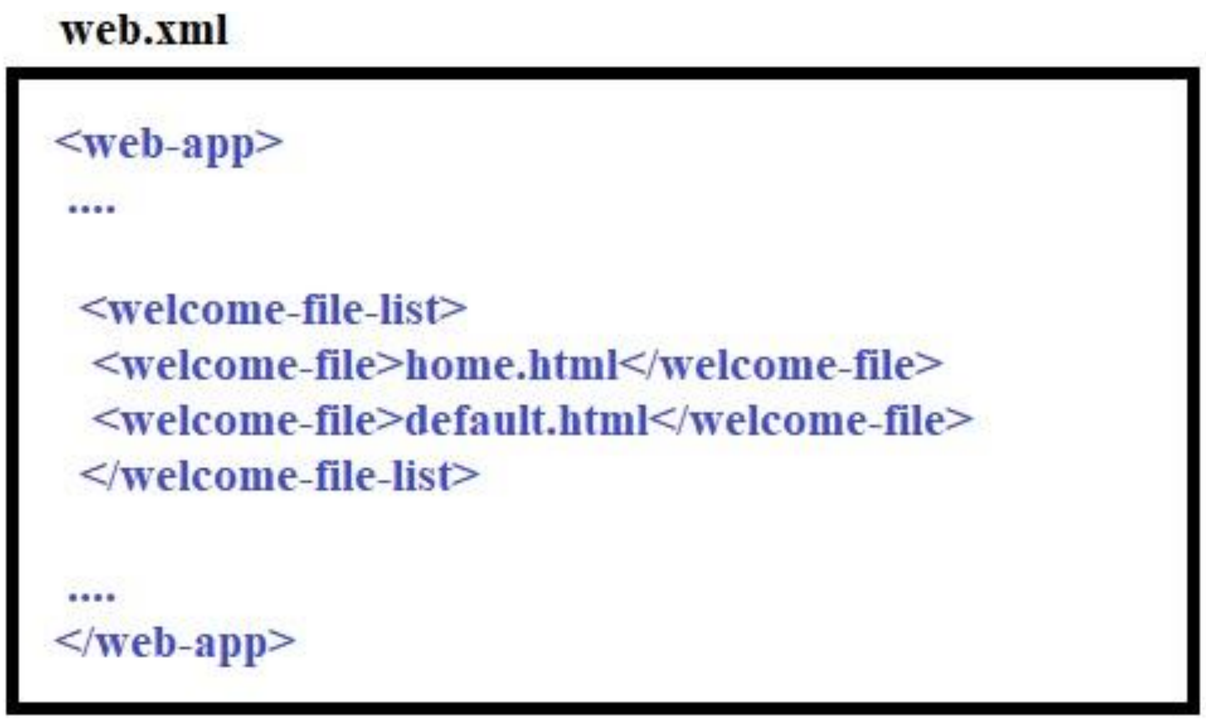


Welcome File List

- The welcome-file-list element of web-app, is used to define a list of welcome files.
- Its sub element is welcome-file that is used to define the welcome file.
- A welcome file is the file that is invoked automatically by the server, if you don't specify any file name.
- By default, server looks for the welcome file in following order:
  - index.html
  - index.htm
  - index.jsp
  - …
- If none of these files are found, server renders 404 error.

Welcome File List

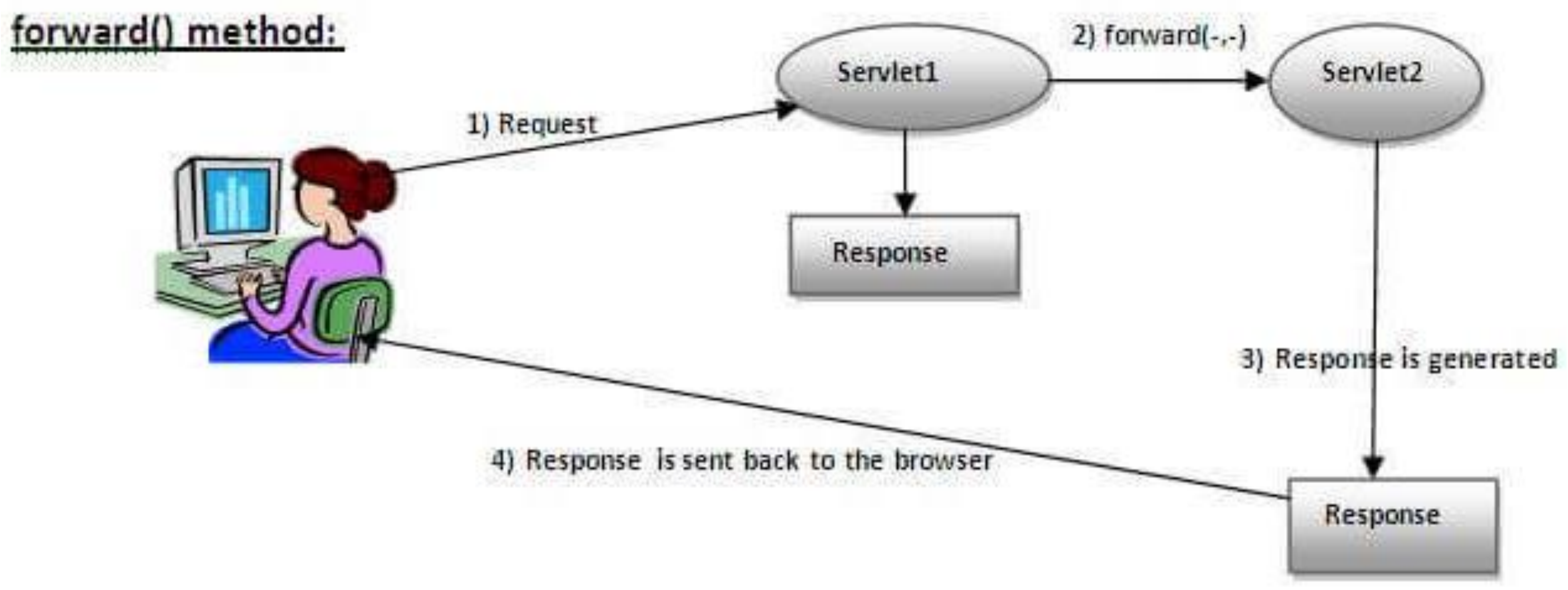
- If you have specified welcome-file in web.xml, and all the files index.html, index.htm and index.jsp exists, priority goes to welcome-file.
- If welcome-file-list entry doesn't exist in web.xml file, priority goes to index.html file then index.htm and at last index.jsp file.


web.xml

```
<web-app>
 ....

  <welcome-file-list>
   <welcome-file>home.html</welcome-file>
   <welcome-file>default.html</welcome-file>
  </welcome-file-list>

 ....
</web-app>
```

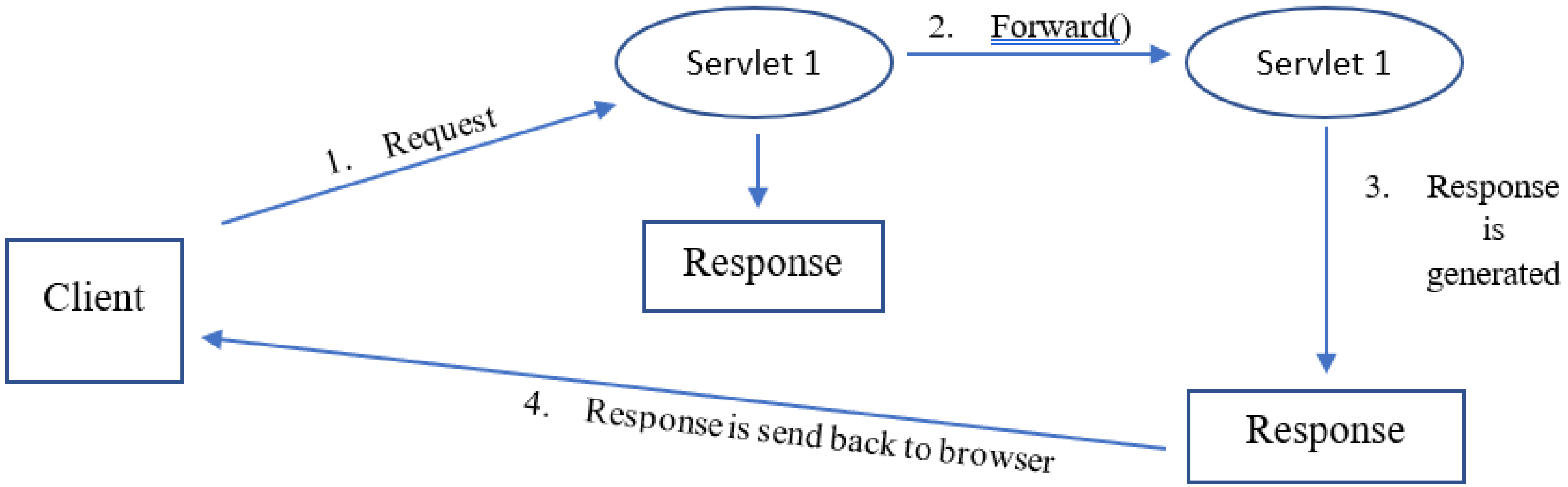
RequestDispatcher

- It is responsible for dispatching the request to another resource it may be html, servlet, jsp or others.
  - Forward()
  - Include()



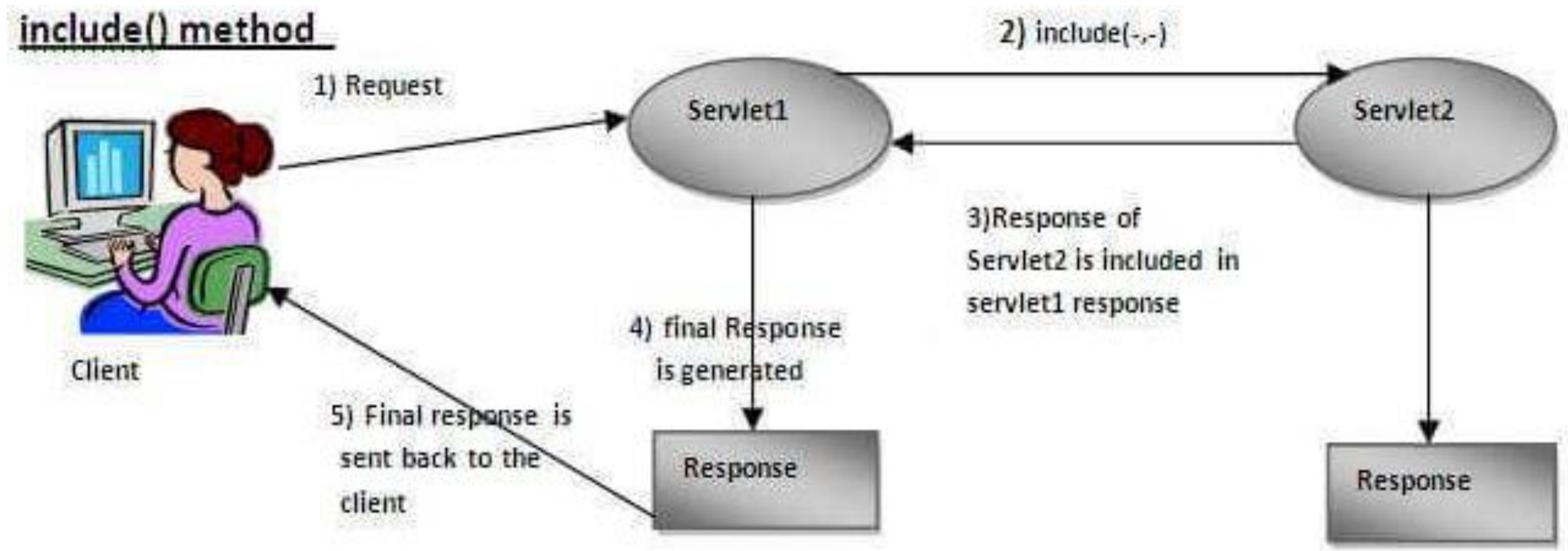
RequestDispatcher - forward()

- public void forward (ServletRequest request,ServletResponse response) throws ServletException, java.io.IOException:
- It Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.



RequestDispatcher – include()

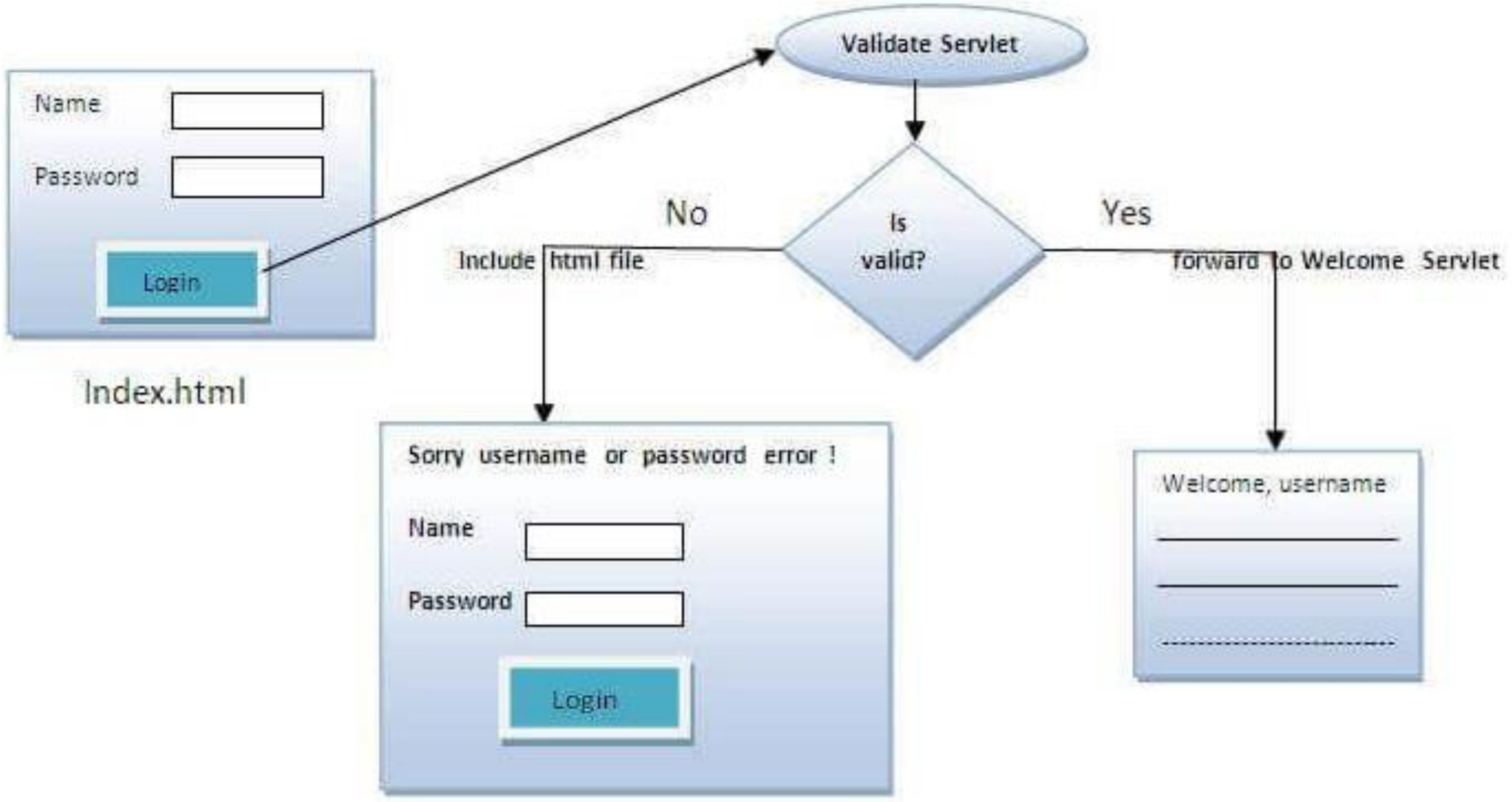
- public void include (ServletRequest request,ServletResponse response) throws ServletException, java.io.IOException:
- It Includes the content of a resource (servlet, JSP page, or HTML file) in the response


include() method
1) Request
Servlet1
2) include(-,-)
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
Client
5) Final response is sent back to the client
Response
Response

RequestDispatcher

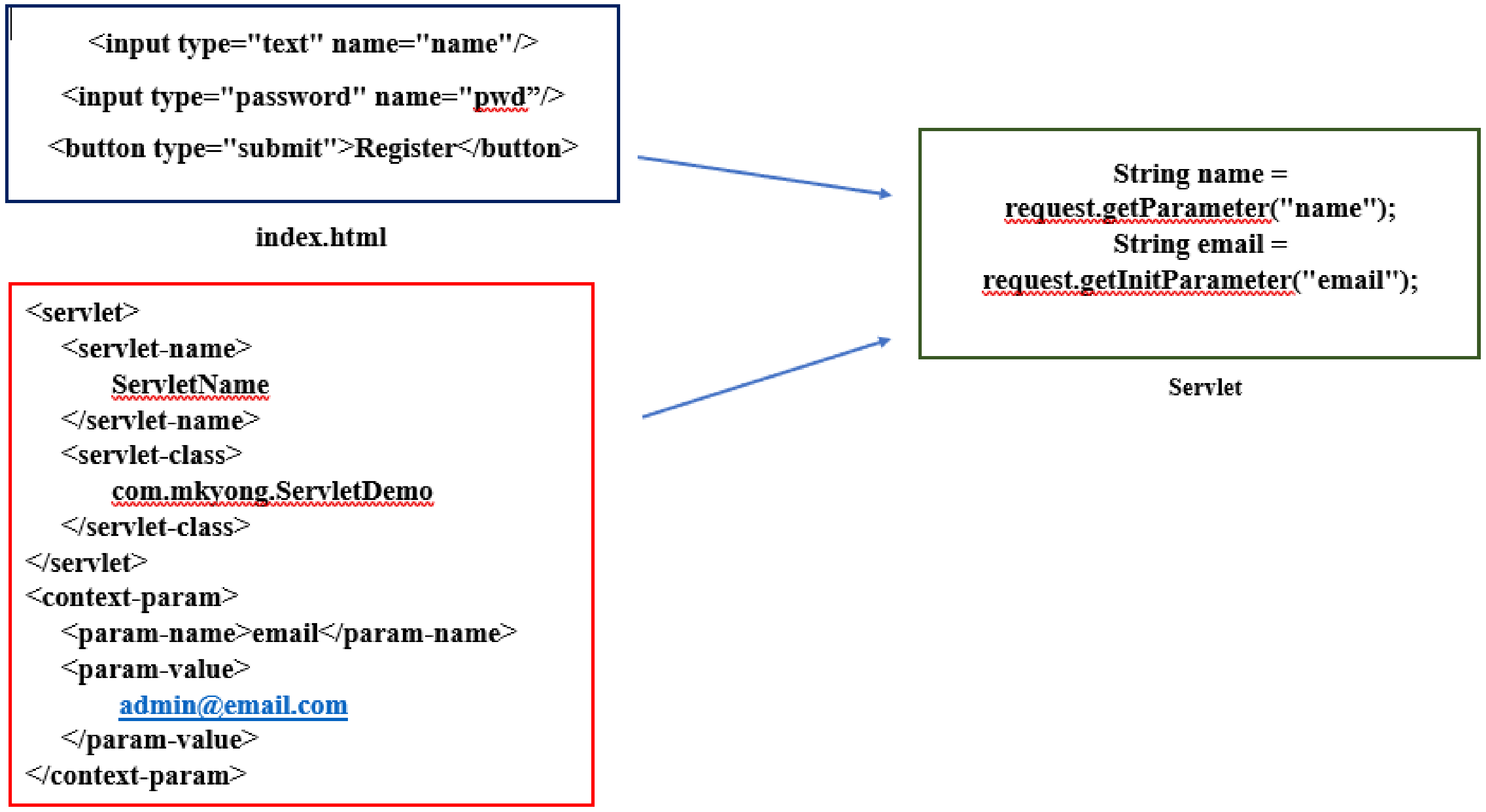
```
RequestDispatcher rd=request.getRequestDispatcher("servlet2");

//servlet2 is the url-pattern of the second servlet

rd.forward(request, response);//method may be forward

rd.include(request, response);//method may be forward
```

RequestDispatcher



What are the parameters in java web application?

- Parameters are those values which are provided by client/web.xml to any servlet to process the request during the request operation.
  - getParameter()
  - getInitParameter()



```
<input type="text" name="name"/>
<input type="password" name="pwd"/>
<button type="submit">Register</button>
```

index.html

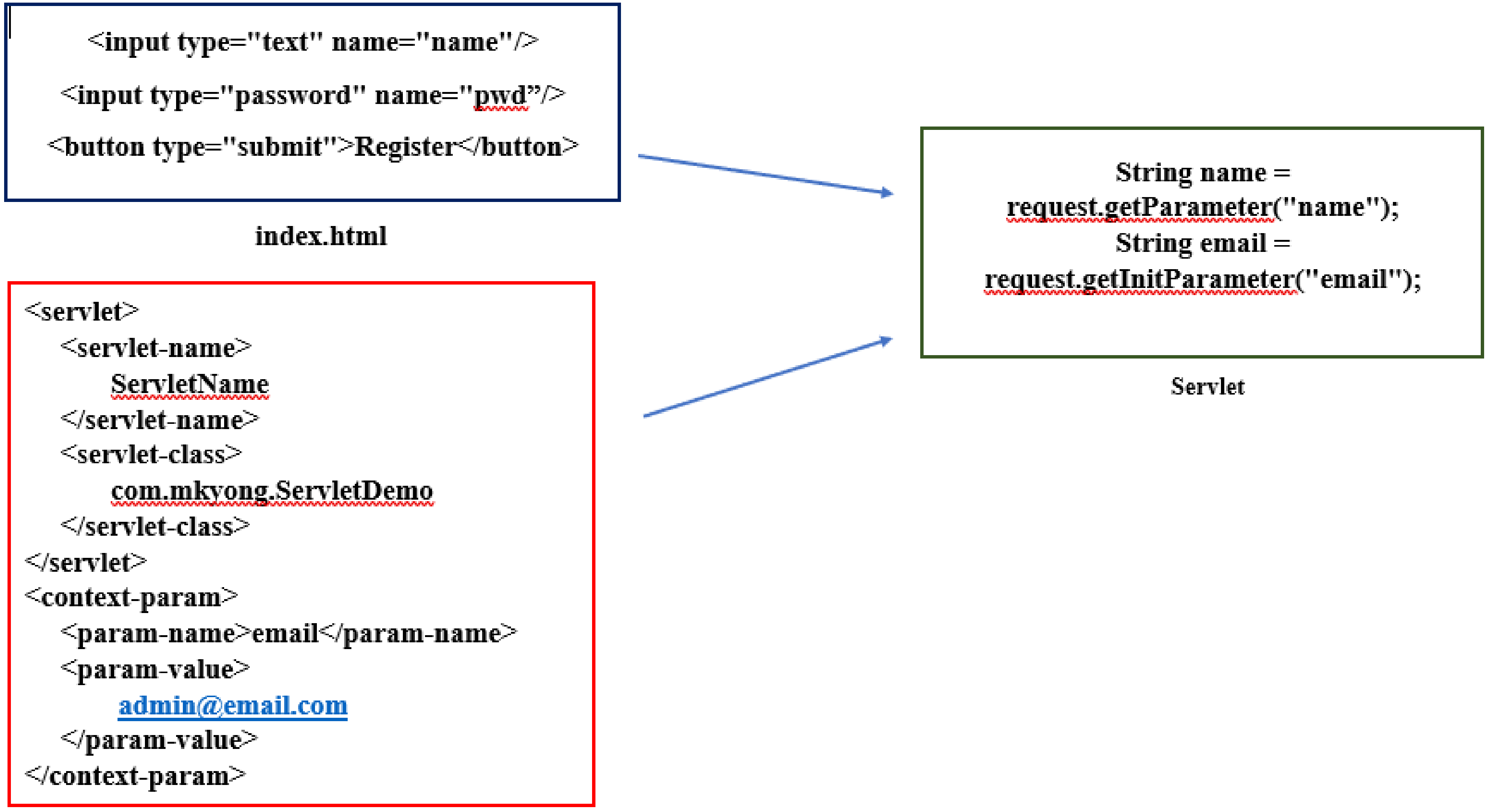
```
<servlet>
    <servlet-name>
        ServletName
    </servlet-name>
    <servlet-class>
        com.mkyong.ServletDemo
    </servlet-class>
</servlet>
<context-param>
    <param-name>email</param-name>
    <param-value>
        admin@email.com
    </param-value>
</context-param>
```

web.xml

```
String name =
request.getParameter("name");
String email =
request.getInitParameter("email");
```

Servlet

getParameter(), getInitParameter()



```
<input type="text" name="name"/>
<input type="password" name="pwd”/>
<button type="submit">Register</button>
```

index.html

```
<servlet>
    <servlet-name>
        ServletName
    </servlet-name>
    <servlet-class>
        com.mkyong.ServletDemo
    </servlet-class>
</servlet>
<context-param>
    <param-name>email</param-name>
    <param-value>
        admin@email.com
    </param-value>
</context-param>
```

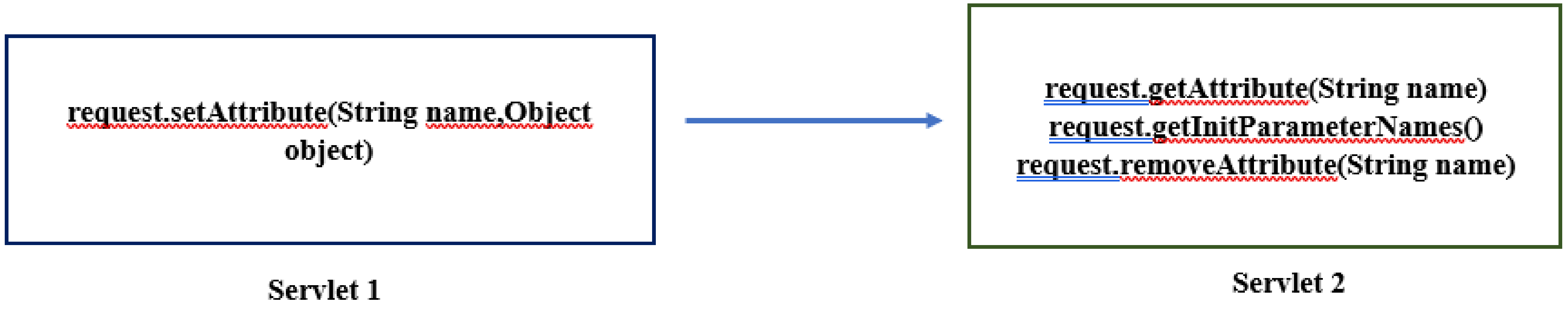
web.xml

```
String name =
request.getParameter("name");
String email =
request.getInitParameter("email");
```

Servlet

What are the attributes in java web application?

- Attributes are the objects that are attached by one servlet to object (session, request, config, context, etc.) and other servlet can fetch that object to process to logic.
- Servlet can easily modify, add and remove the content of attribute when required.



```
request.setAttribute(String name,Object object)
```

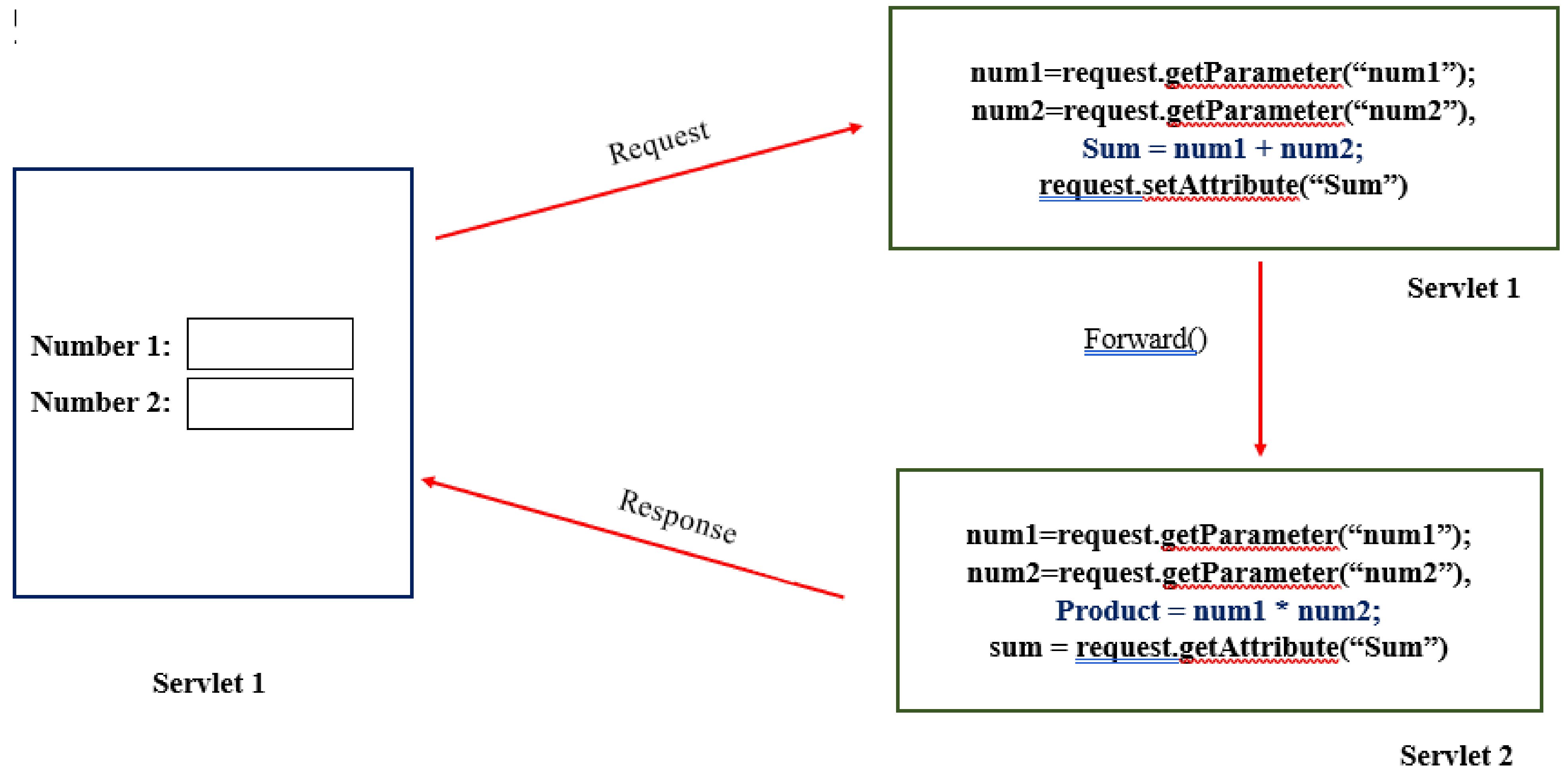
```
request.getAttribute(String name)
request.getInitParameterNames()
request.removeAttribute(String name)
```

What are the attributes in java web application?

- There are following 4 attribute specific methods. They are as follows:

- public void setAttribute(String name,Object object)
  - sets the given object in the application scope.
- public Object getAttribute(String name)
  - Returns the attribute for the specified name.
- public Enumeration getInitParameterNames()
  - Returns the names of the context's initialization parameters as an Enumeration of String objects.
- public void removeAttribute(String name)
  - Removes the attribute with the given name from the servlet context.

Example of Parameter & Attributes



```
num1=request.getParameter("num1");
num2=request.getParameter("num2"),
Sum = num1 + num2;
request.setAttribute("Sum")
```

```
num1=request.getParameter("num1");
num2=request.getParameter("num2"),
Product = num1 * num2;
sum = request.getAttribute("Sum")
```

Example of Parameter & Attributes

- There are following 4 attribute specific methods. They are as follows:
- public void setAttribute(String name,Object object)
  - sets the given object in the application scope.
- public Object getAttribute(String name)
  - Returns the attribute for the specified name.
- public Enumeration getInitParameterNames()
  - Returns the names of the context's initialization parameters as an Enumeration of String objects.
- public void removeAttribute(String name)
  - Removes the attribute with the given name from the servlet context.