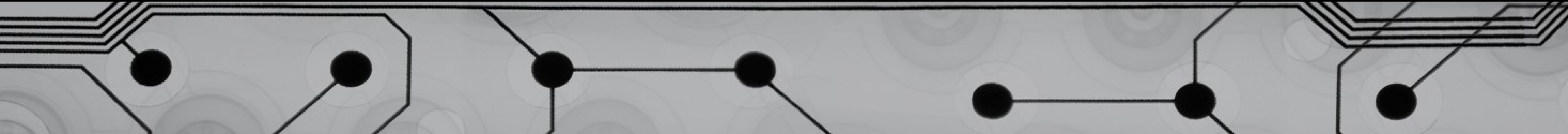
CST 243-3 Rapid Application Development

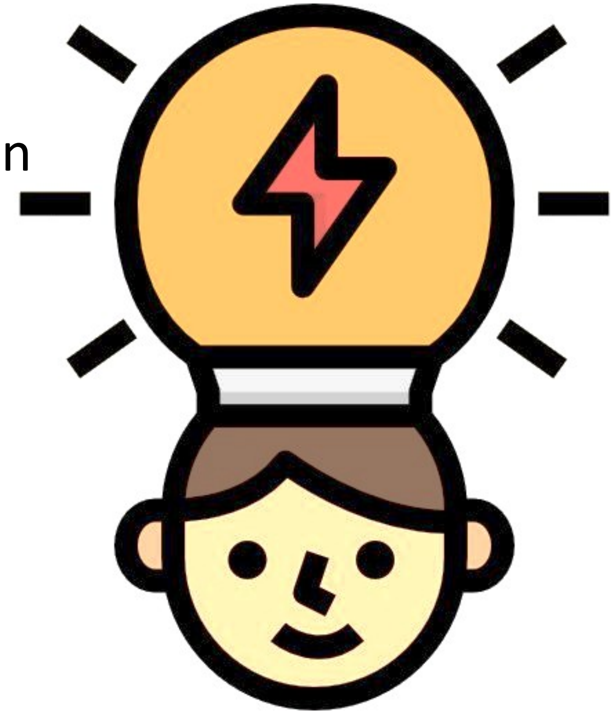
Lesson 04: Web Technologies for Rapid Development

By Subhash Ariyadasa



Lesson Learning Outcomes

- After successful completion of this lesson you will be able to,
  - **Understand** how **Java Servlets** and **JSP** fit into the overall architecture of web applications
  - **Handle different types of web requests** using Java Servlets, **extract data from requests**, **perform data processing and business logic**, and **generate appropriate responses to clients**
  - **Create dynamic web pages using JSP** by embedding Java code within HTML markup
  - **Understand** how to leverage **JSP tags**, **expressions**, and **scripting elements to generate dynamic content**, **iterate over collections**, **implement conditional logic**, and **access data sources**
  - **Gain awareness of common vulnerabilities** and understand how to **mitigate security risks** in web applications
  - **Apply the gained knowledge** about Java Servlet and JSP **to implement real-world dynamic web applications**



Lesson Outline

- Part I: Java Servlets
  - Dynamic Web Pages
  - Servlet Basics
  - Java Servlets with JDBC
  - Session Management and Cookies
- Part II: JSP
  - JSP basics
  - Processing a JSP File
  - JSP Life Cycle
  - JSP Syntax
  - JSP Directives

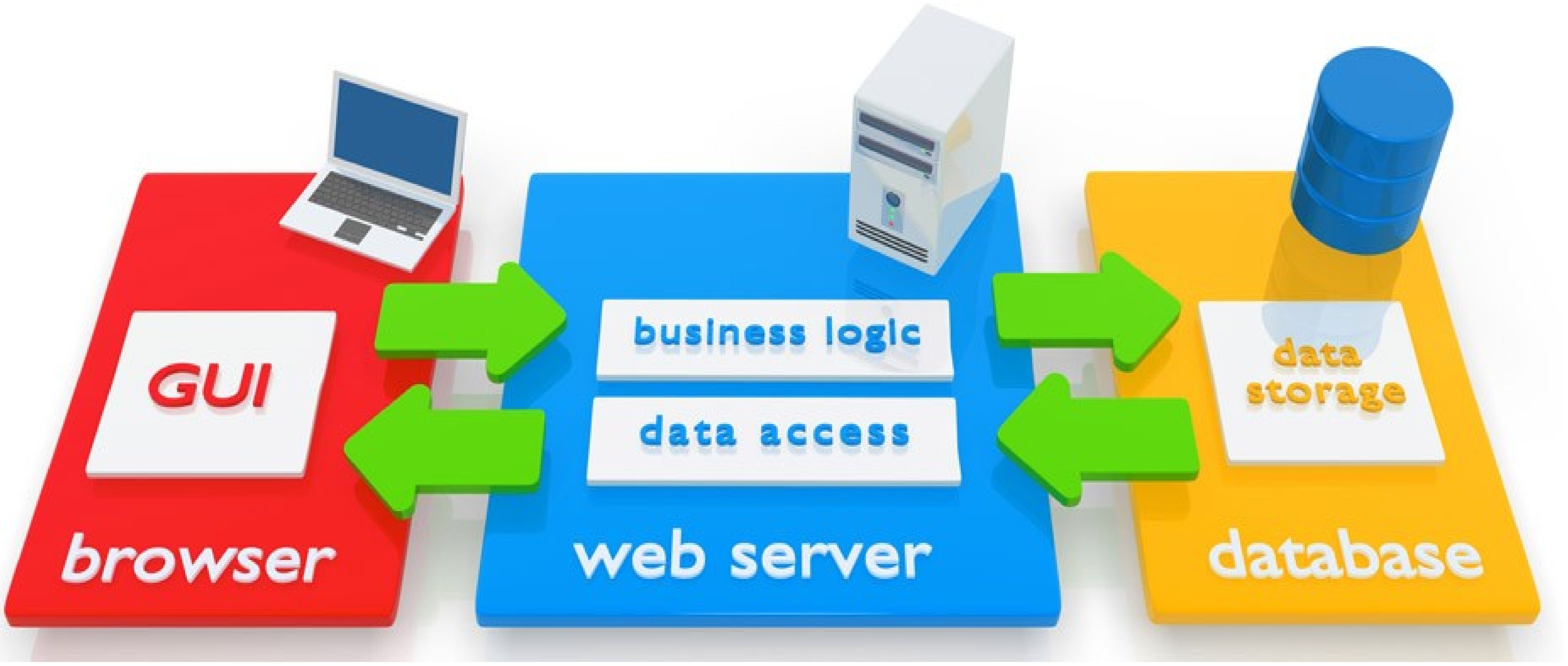
Part I

Java Servlets

Dynamic Web Pages

- Pages which displays different content for different users while retaining the same layout and design
- E.g.:
  - Facebook
  - Twitter
- The site contents change according to the time
- Java Servlet is there to generate dynamic web pages

Dynamic Web Pages…

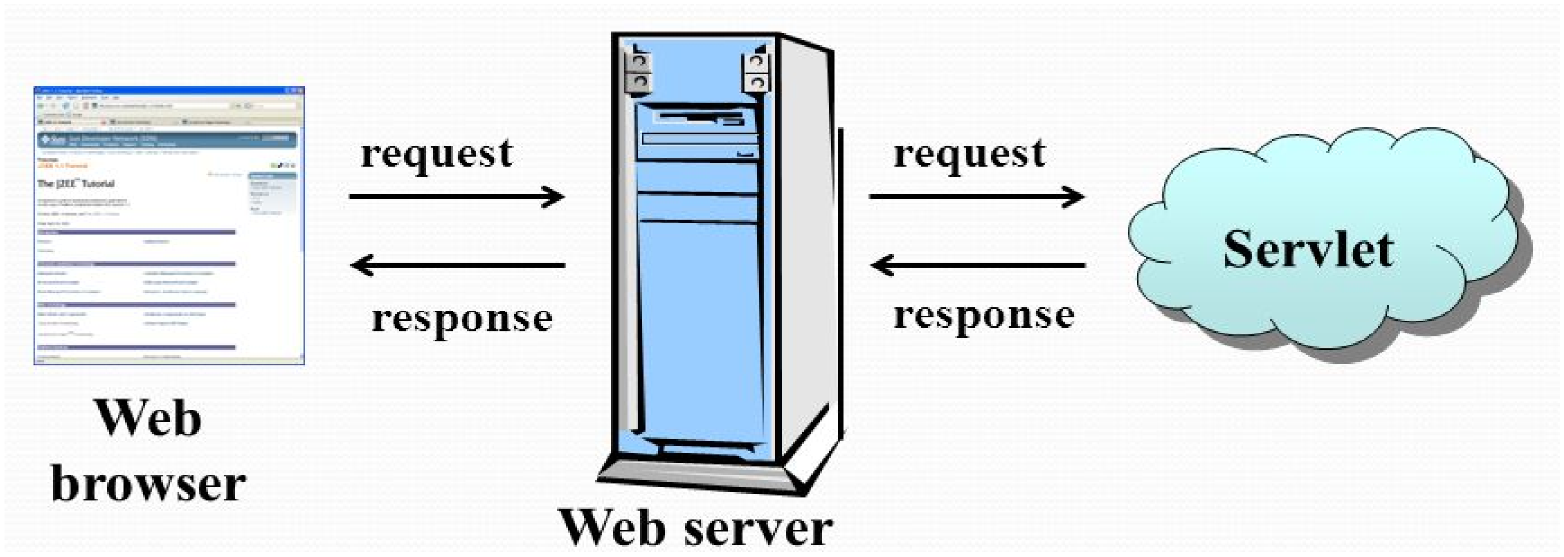


Java Servlets

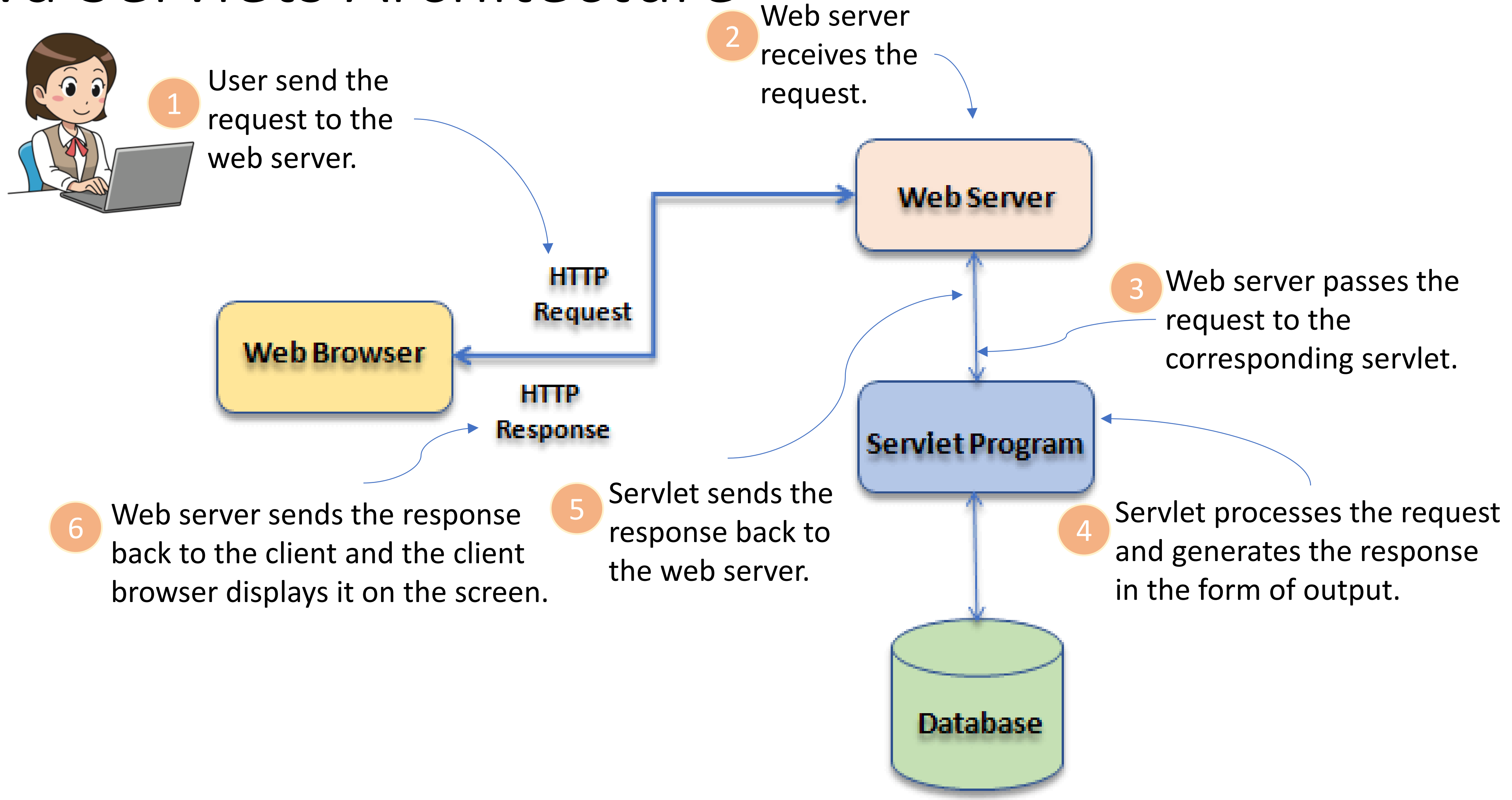
- Servlets are the Java programs that run on the Java-enabled web server or application server
  - Glassfish
  - JBoss EAP
  - Apache Tomcat
  - Apache TomEE
  - Jetty
  - Wildfly

Java Servlets…

- It can receive parameters in an HTTP request
- It will generate an HTTP response



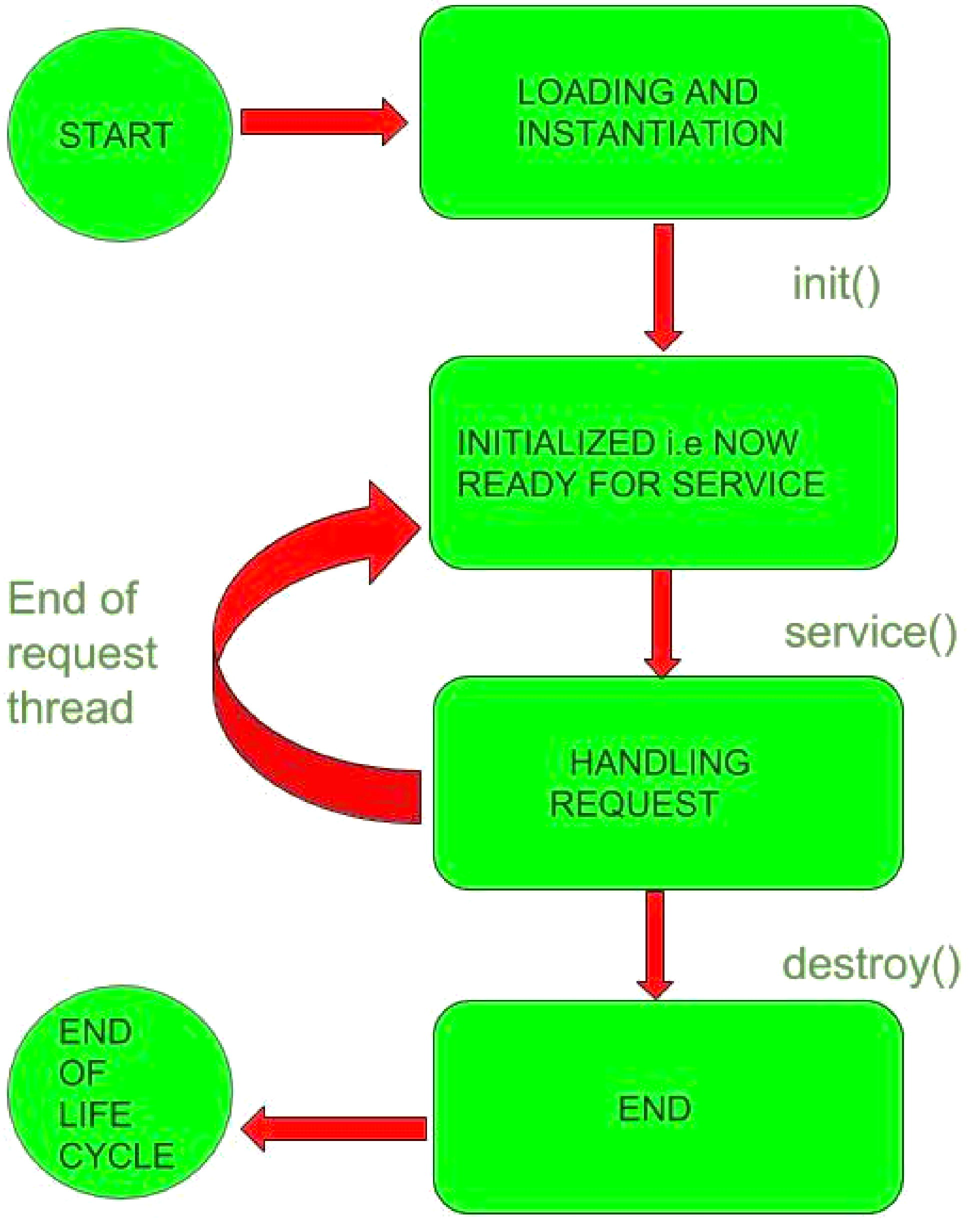
Java Servlets Architecture



1 User send the request to the web server.
2 Web server receives the request.
3 Web server passes the request to the corresponding servlet.
4 Servlet processes the request and generates the response in the form of output.
5 Servlet sends the response back to the web server.
6 Web server sends the response back to the client and the client browser displays it on the screen.
Web Server
HTTP Request
Web Browser
HTTP Response
Servlet Program
Database

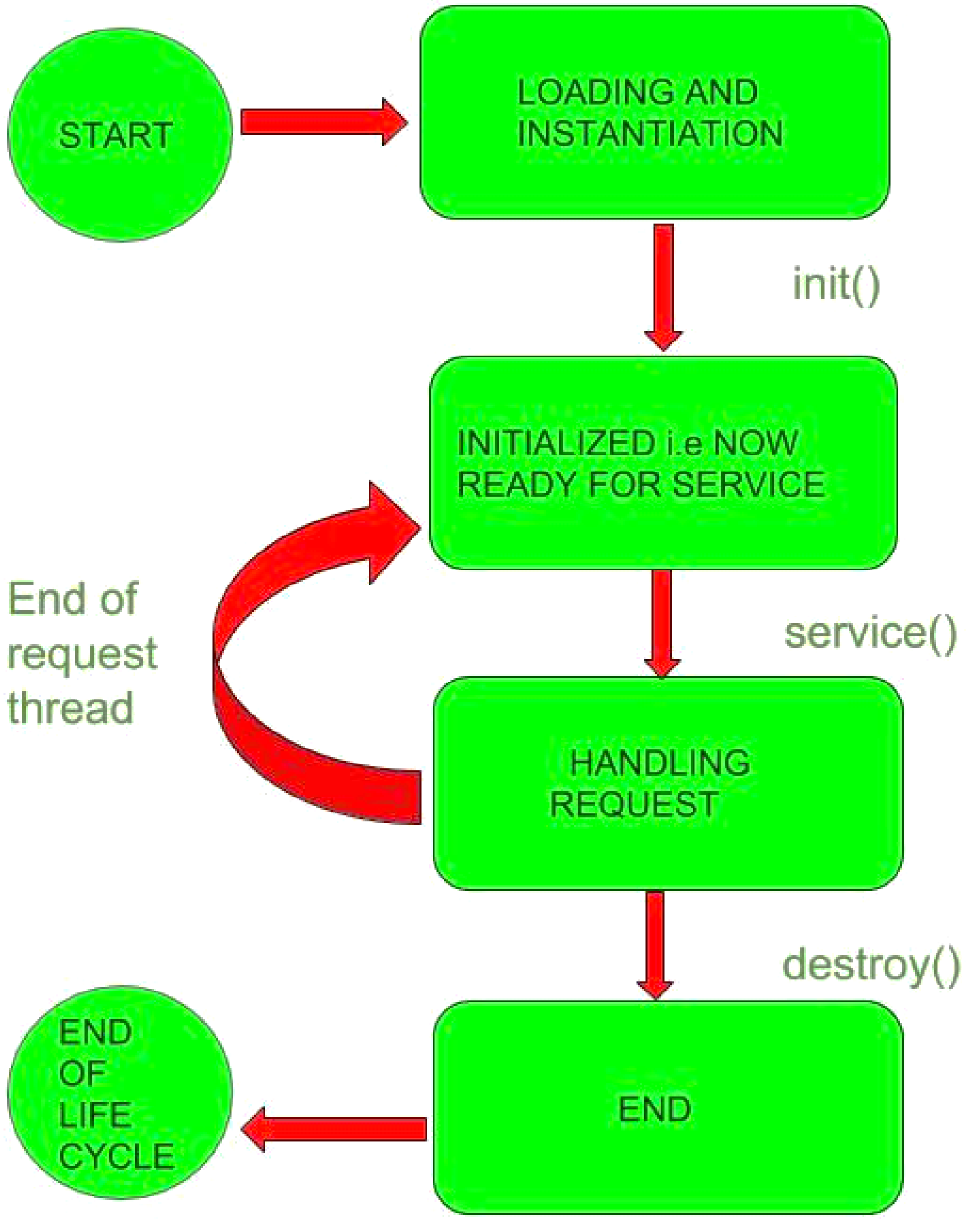
Servlet Life Cycle

- Loading a Servlet
- Initializing the Servlet
- Request handling
- Destroying the Servlet



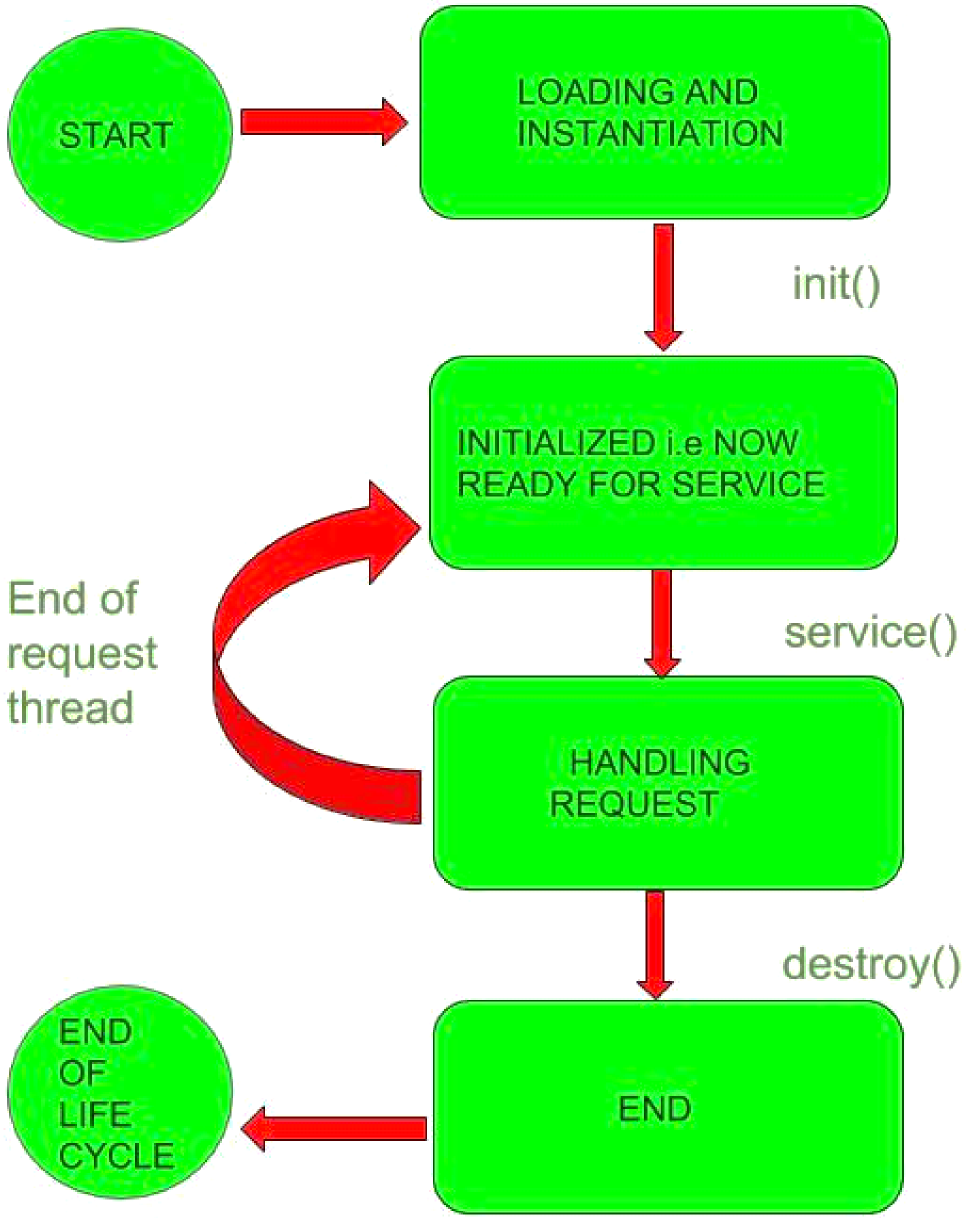
Servlet Life Cycle

- Loading a Servlet
  - When the web server starts up, the servlet container deploy and loads all the servlets.
- Initializing the Servlet
- Request handling
- Destroying the Servlet



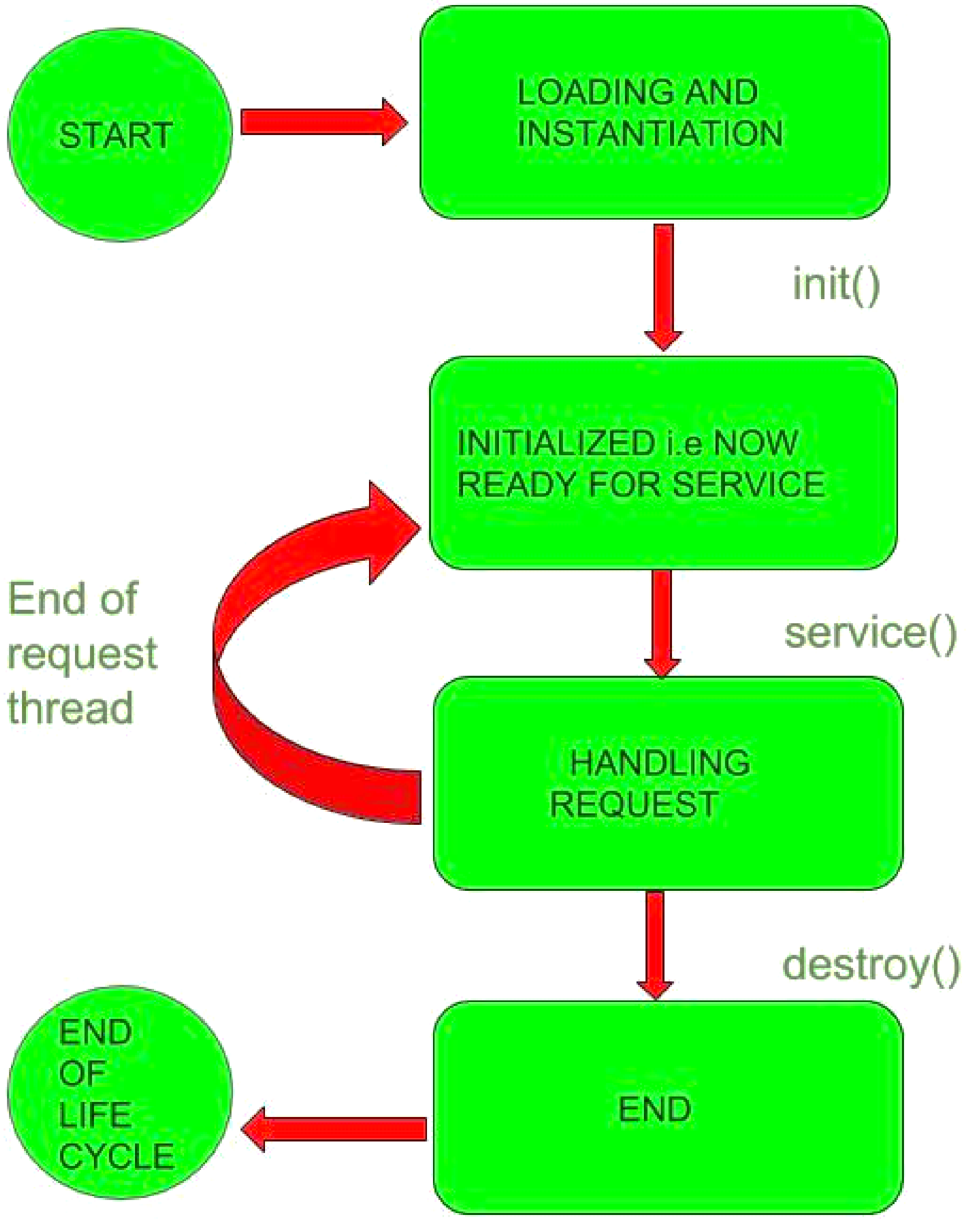
Servlet Life Cycle

- Loading a Servlet
- Initializing the Servlet
  - The Servlet.init() method is called by the Servlet container.
  - Check whether the Servlet instance is instantiated successfully and ready to serve.
  - This method is called only once.
- Request handling
- Destroying the Servlet



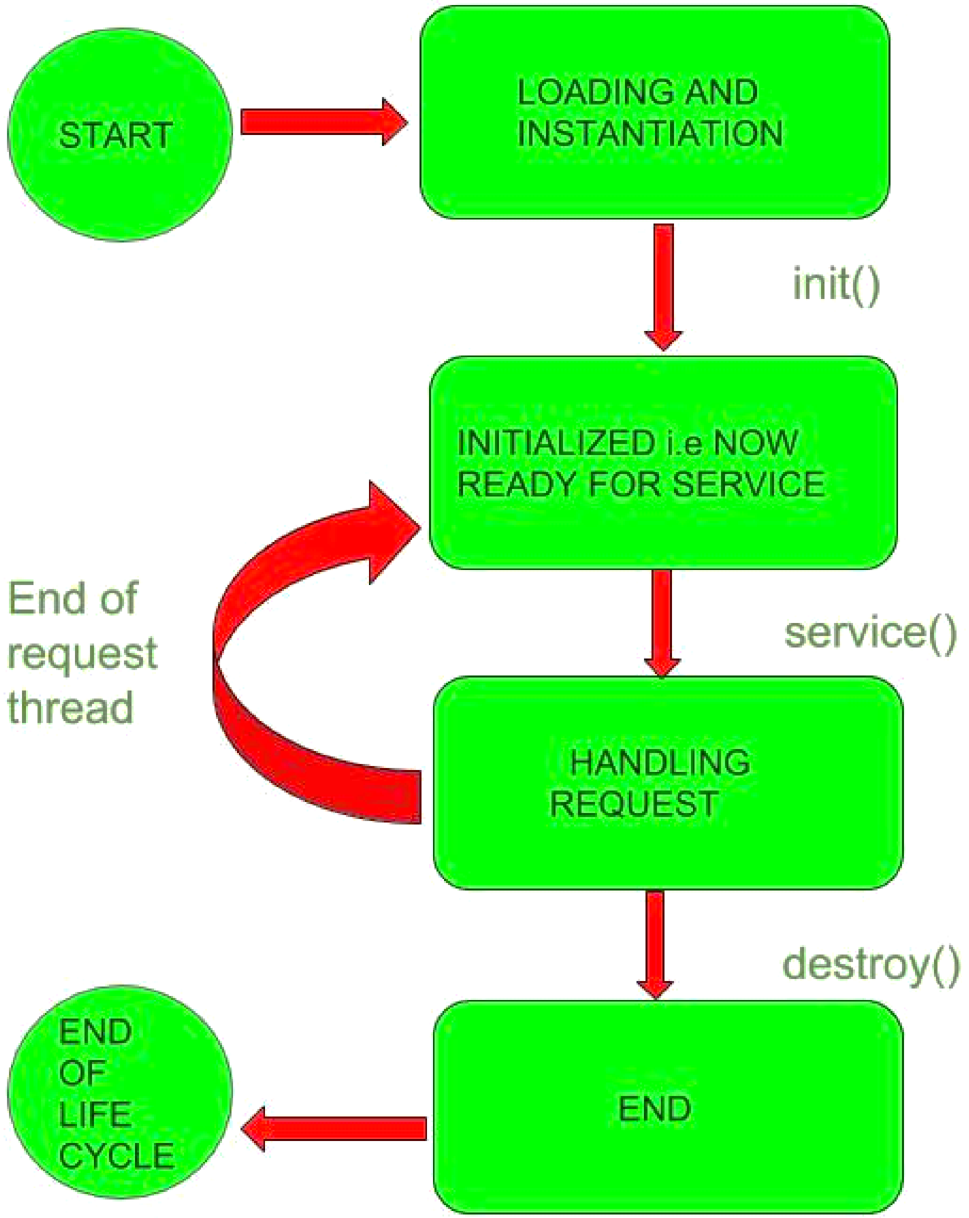
Servlet Life Cycle

- Loading a Servlet
- Initializing the Servlet
- Request handling
    - The servlet calls service() method to process a client's request
- Destroying the Servlet



Servlet Life Cycle

- Loading a Servlet
- Initializing the Servlet
- Request handling
- Destroying the Servlet
  - The destroy() method runs only once during the lifetime of a Servlet

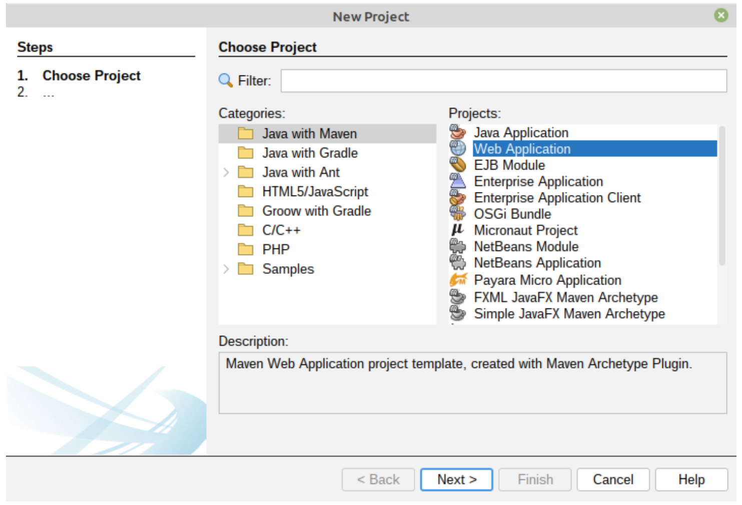


Steps to Create Servlet

- Create a directory structure
- Create and compile the Servlet
- Configure Servlet by adding mappings
- Start the server and deploy the project
- Access the servlet

Steps to Create Servlet…

- Create a directory structure



New Project

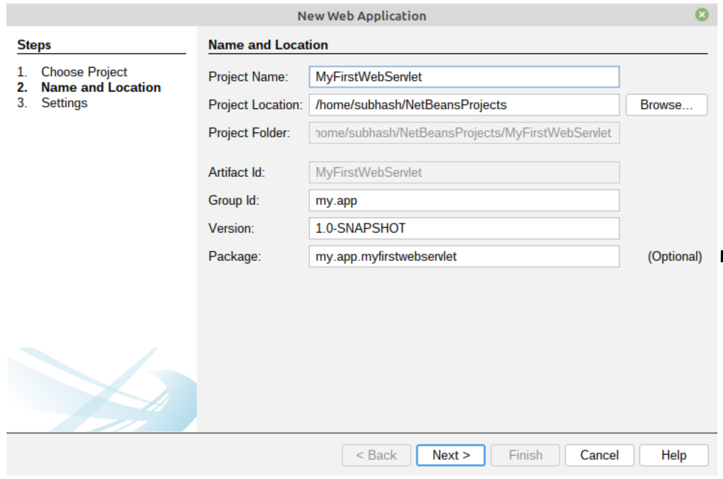
Steps
1. Choose Project
2. …

Choose Project

Filter:

Categories:	Projects:
Java with Maven	Java Application
Java with Gradle	Web Application
Java with Ant	EJB Module
HTML5/JavaScript	Enterprise Application
Groovy with Gradle	Enterprise Application Client
C/C++	OSGi Bundle
PHP	Micronaut Project
Samples	NetBeans Module
	NetBeans Application
	Payara Micro Application
	FXML JavaFX Maven Archetype
	Simple JavaFX Maven Archetype

Description:
Maven Web Application project template, created with Maven Archetype Plugin.

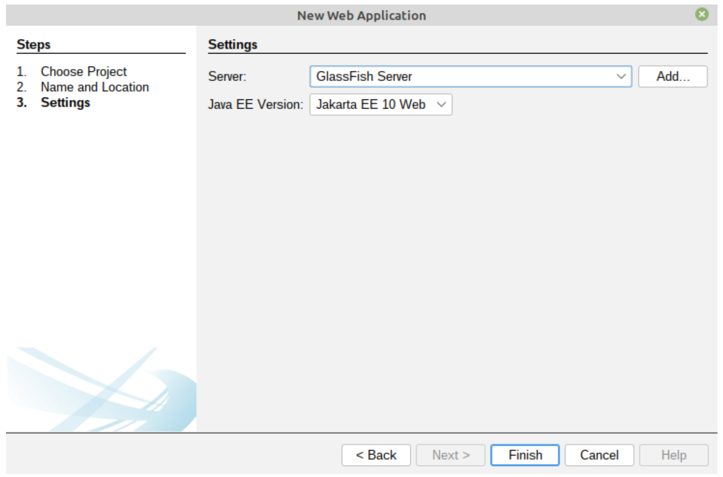


New Web Application

Steps
1. Choose Project
2. Name and Location
3. Settings

Name and Location

Project Name:	MyFirstWebServlet
Project Location:	/home/subhash/NetBeansProjects
Project Folder:	ıome/subhash/NetBeansProjects/MyFirstWebServlet
Artifact Id:	MyFirstWebServlet
Group Id:	my.app
Version:	1.0-SNAPSHOT
Package:	my.app.myfirstwebservlet (Optional)

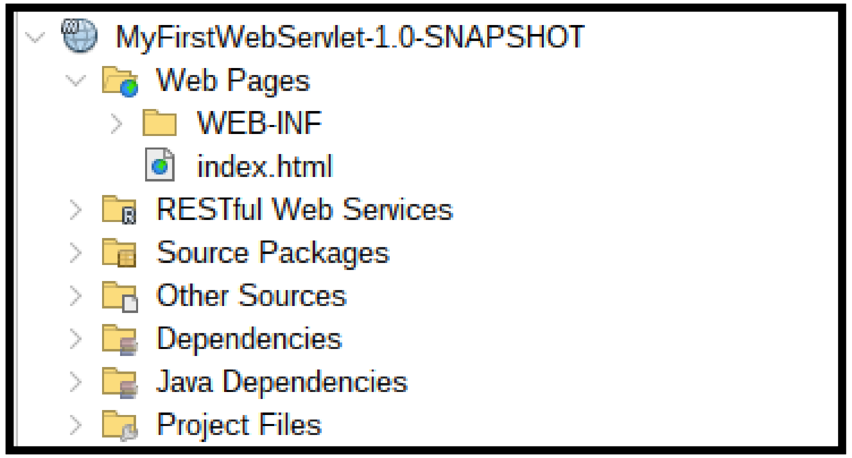


New Web Application

Steps
1. Choose Project
2. Name and Location
3. Settings

Settings

Server:	GlassFish Server
Java EE Version:	Jakarta EE 10 Web



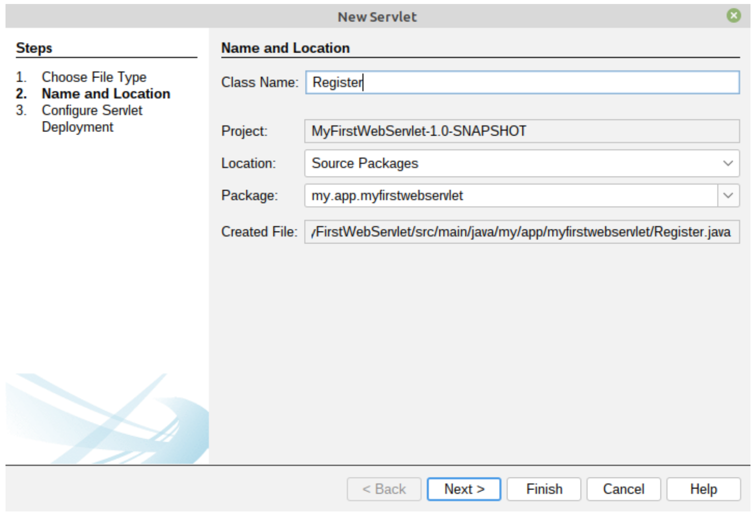
```
MyFirstWebServlet-1.0-SNAPSHOT
  Web Pages
    WEB-INF
    index.html
  RESTful Web Services
  Source Packages
  Other Sources
  Dependencies
  Java Dependencies
  Project Files
```

Steps to Create Servlet…

- WEB-INF folder
  - A **special directory** within a web application's directory structure
  - Serves several key purposes related to the **configuration** and **security** of the web application - provides a layer of security for **sensitive configuration files** and **resources**
  - Normally contains some configuration files like web.xml
    - web.xml – defines servlet mappings, welcome files, error pages, security constraints, context parameters, and other configuration details
  - Mainly **improves the security** of the application – not accessible directly via the web

Steps to Create Servlet…

- Create a Servlet and Compile it



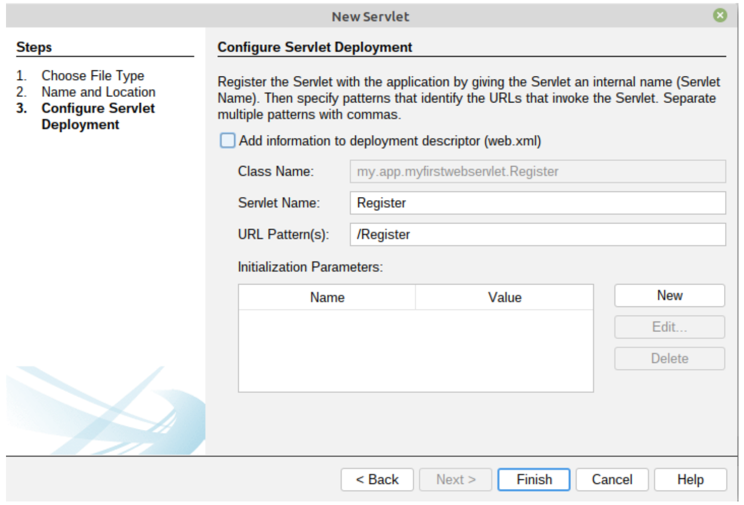
New Servlet

Steps

1. Choose File Type
2. **Name and Location**
3. Configure Servlet Deployment

Name and Location

Class Name:	Register
Project:	MyFirstWebServlet-1.0-SNAPSHOT
Location:	Source Packages
Package:	my.app.myfirstwebservlet
Created File:	/FirstWebServlet/src/main/java/my/app/myfirstwebservlet/Register.java



New Servlet

Steps

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

Configure Servlet Deployment

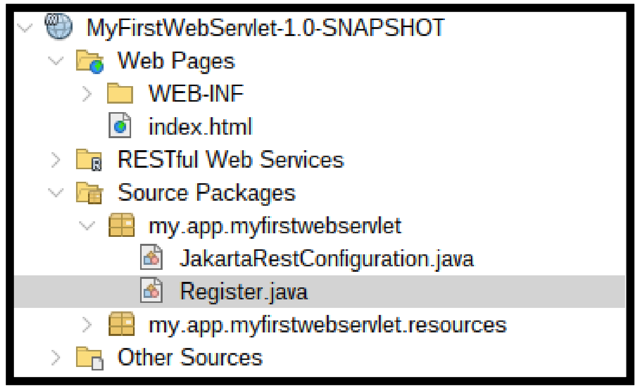
Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☐ Add information to deployment descriptor (web.xml)

Class Name:	my.app.myfirstwebservlet.Register
Servlet Name:	Register
URL Pattern(s):	/Register

Initialization Parameters:

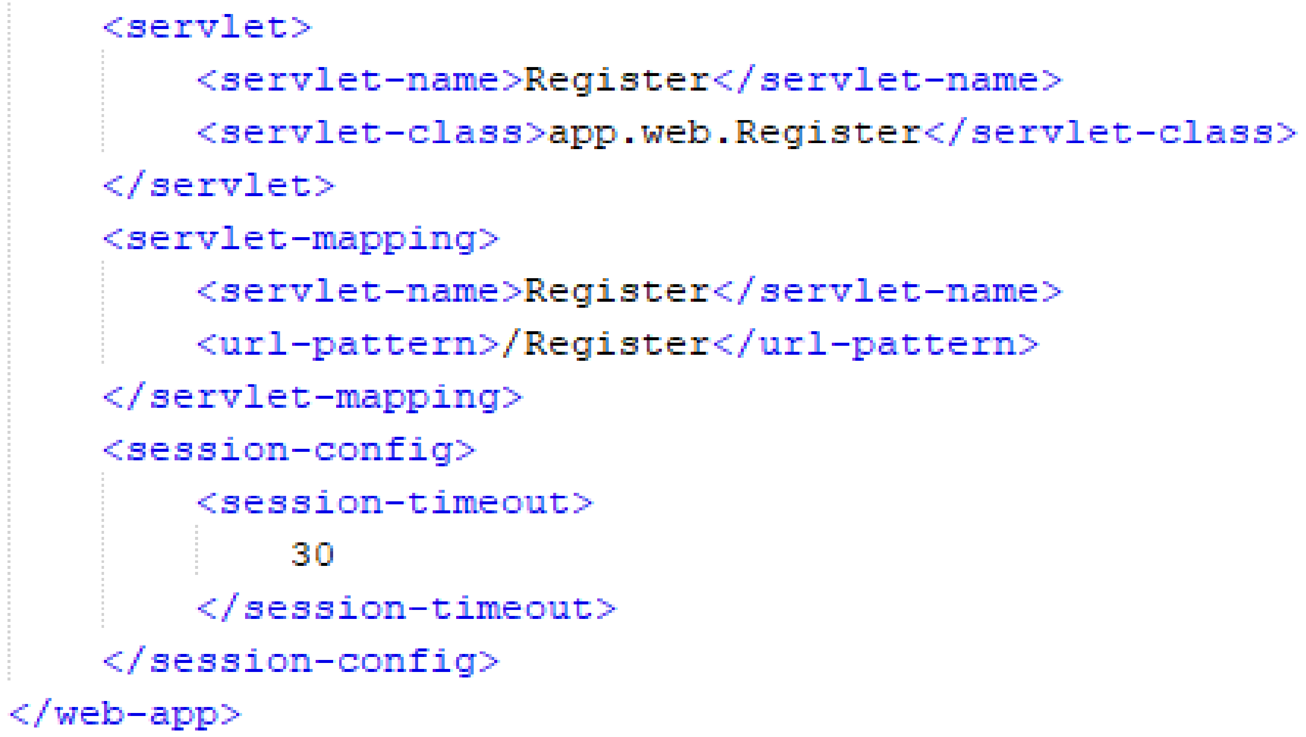
Name	Value



```
MyFirstWebServlet-1.0-SNAPSHOT
  Web Pages
    WEB-INF
    index.html
  RESTful Web Services
  Source Packages
    my.app.myfirstwebservlet
      JakartaRestConfiguration.java
      Register.java
    my.app.myfirstwebservlet.resources
  Other Sources
```

XML-based or Annotation-based

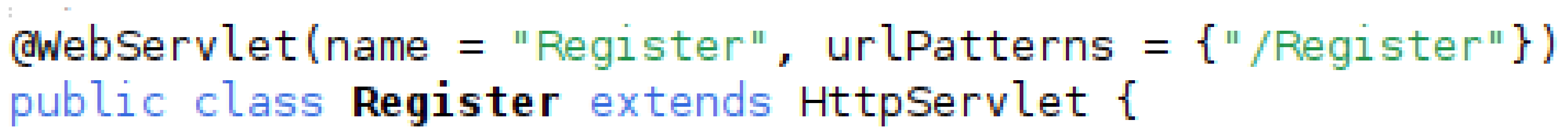
- XML-based Approach
  - Servlets are configured using deployment descriptor files (web.xml)



```
    <servlet>
        <servlet-name>Register</servlet-name>
        <servlet-class>app.web.Register</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Register</servlet-name>
        <url-pattern>/Register</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

XML-based or Annotation-based

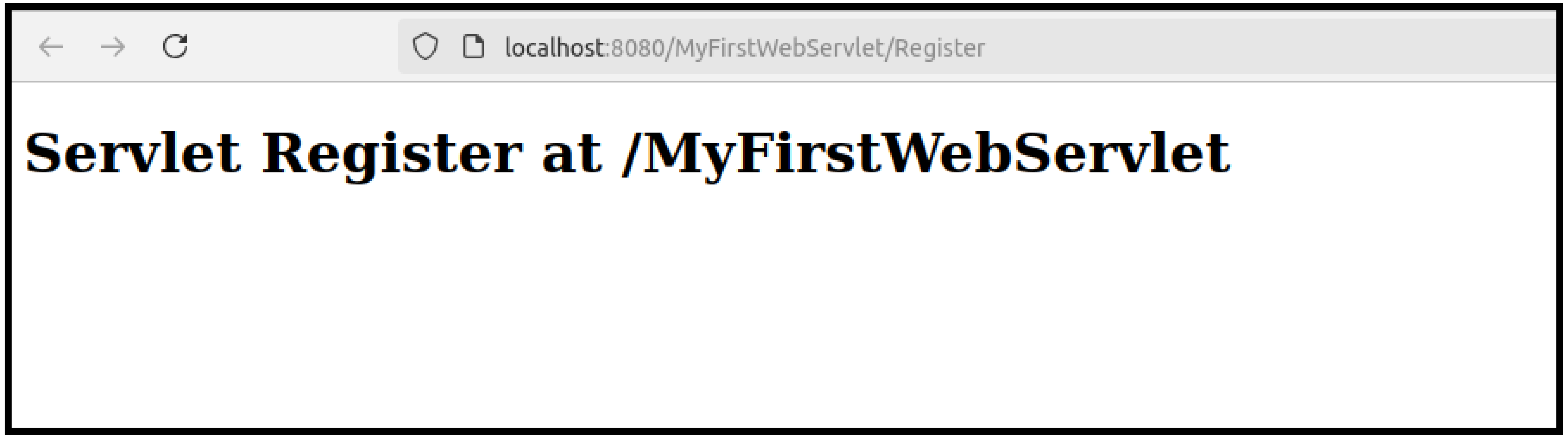
- Annotation-based Approach
  - A special kind of metadata that provides additional information about your code (E.g., @Override)
  - Used directly in the servlet to define their mappings and configurations



```java
@WebServlet(name = "Register", urlPatterns = {"/Register"})
public class Register extends HttpServlet {
```

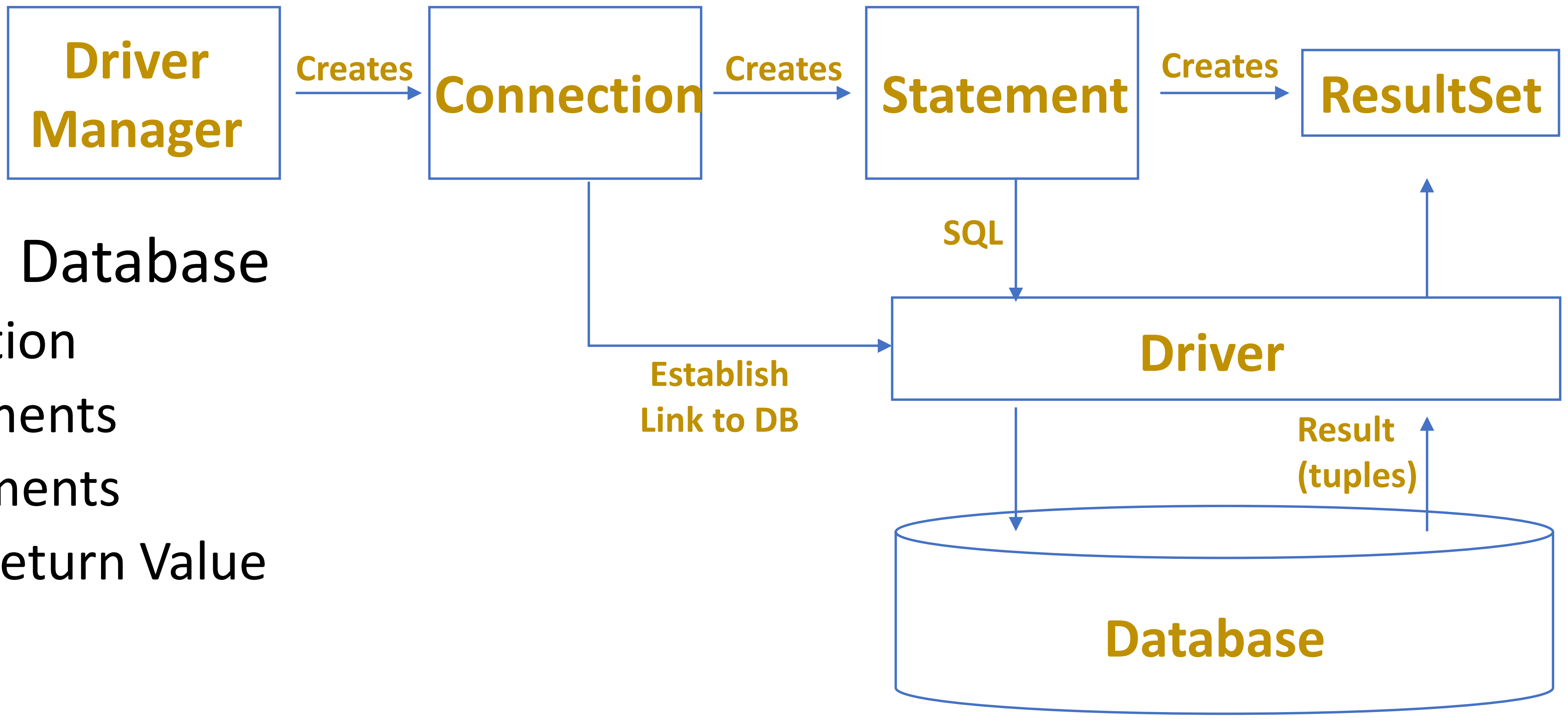
Steps to Create Servlet…

- Start the server and deploy the project
- Access the servlet


localhost:8080/MyFirstWebServlet/Register

Servlet Register at /MyFirstWebServlet

How to Connect with a Database?

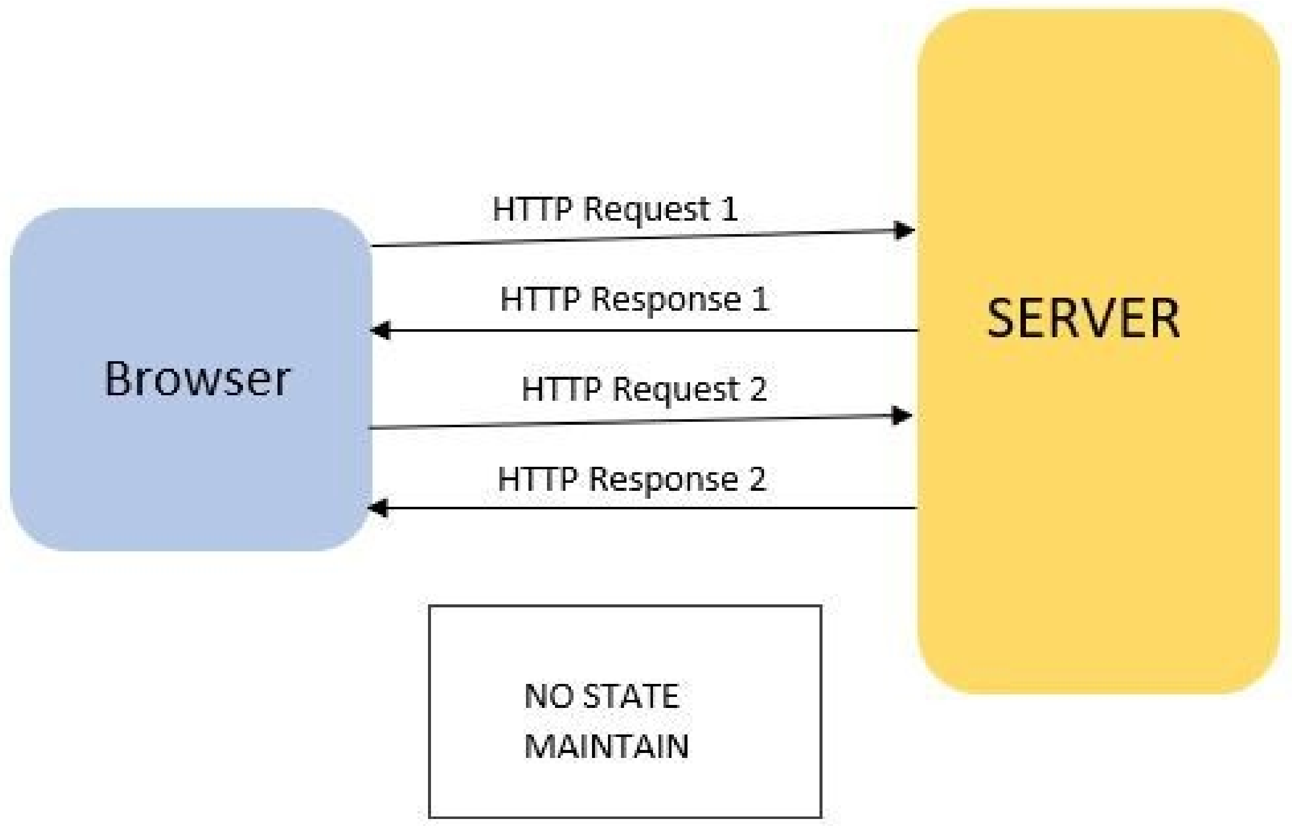


- Basic Steps to use a Database
  - Establish a connection
  - Create JDBC Statements
  - Execute SQL Statements
  - Get ResultSet OR Return Value
  - Close Connections

Session Management and Cookies

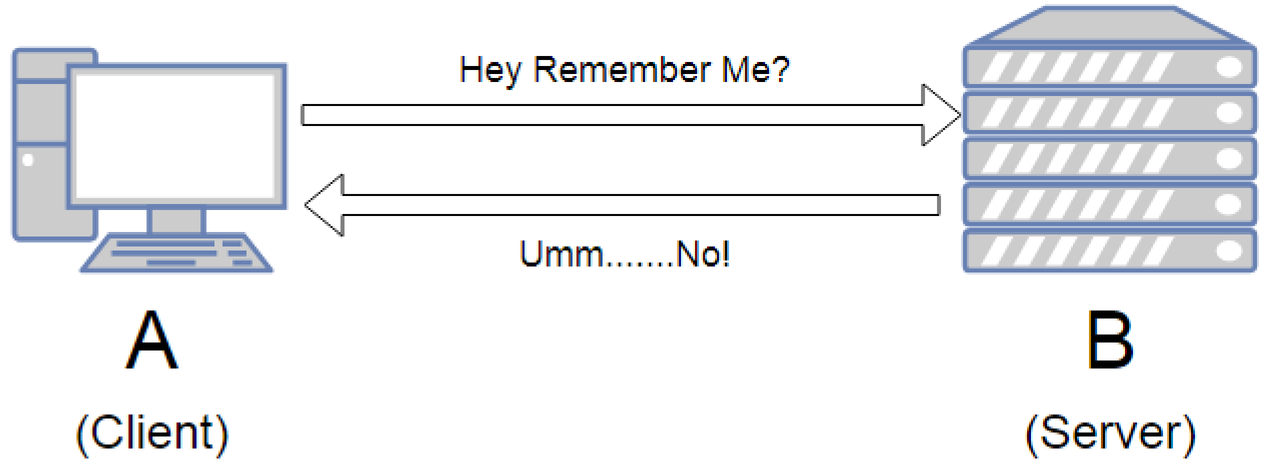
Why do We Need Sessions?

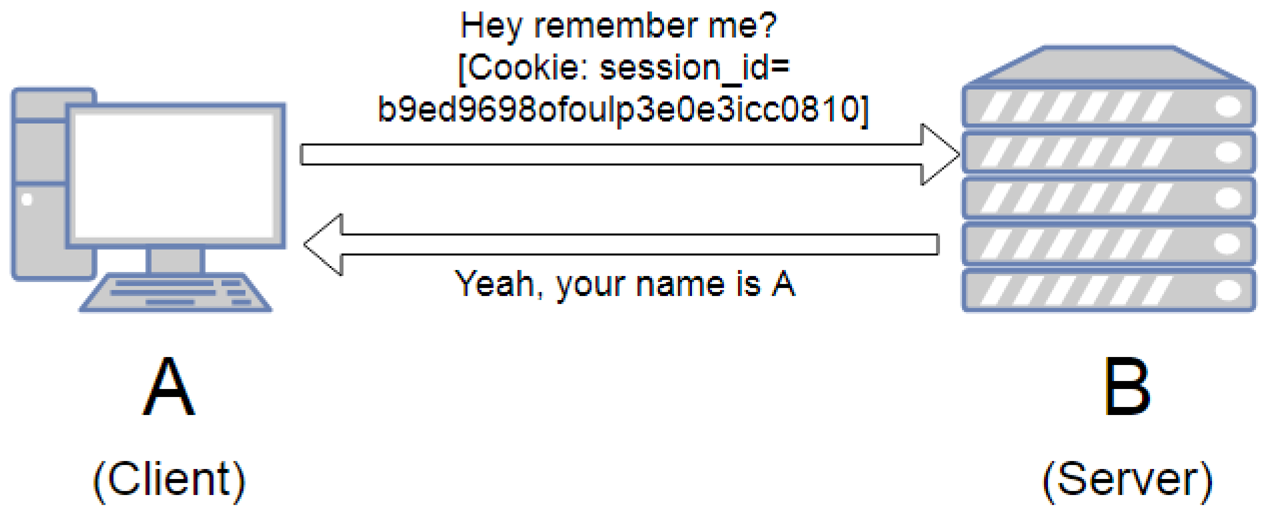
- HTTP is a **stateless protocol**.
- Once a user request some URL, **web-server serve the requested page and closes** the connection.
- **Each request is unique for a server** and isolated from previous requests.



Why do We Need Sessions?...

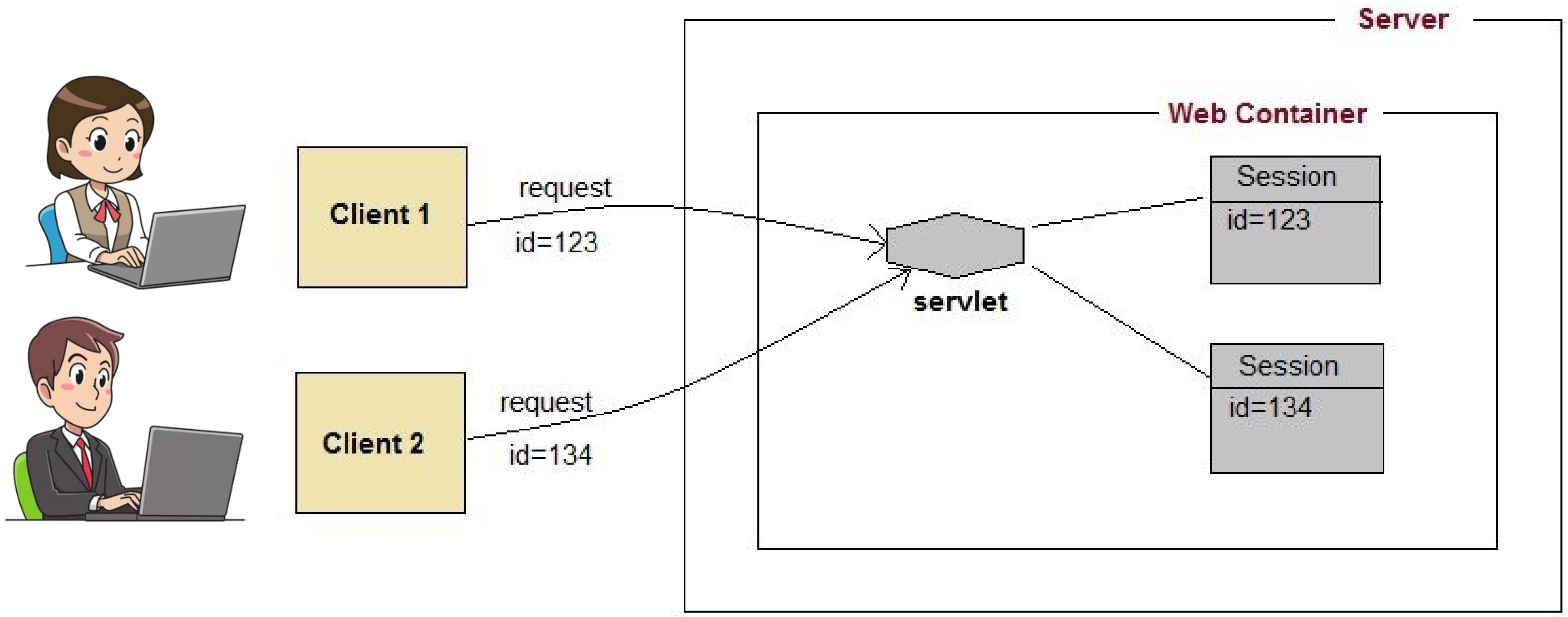
- Identify who made a particular request to serve **personalized contents** to the user
- **Temporary storage** named Session is essential





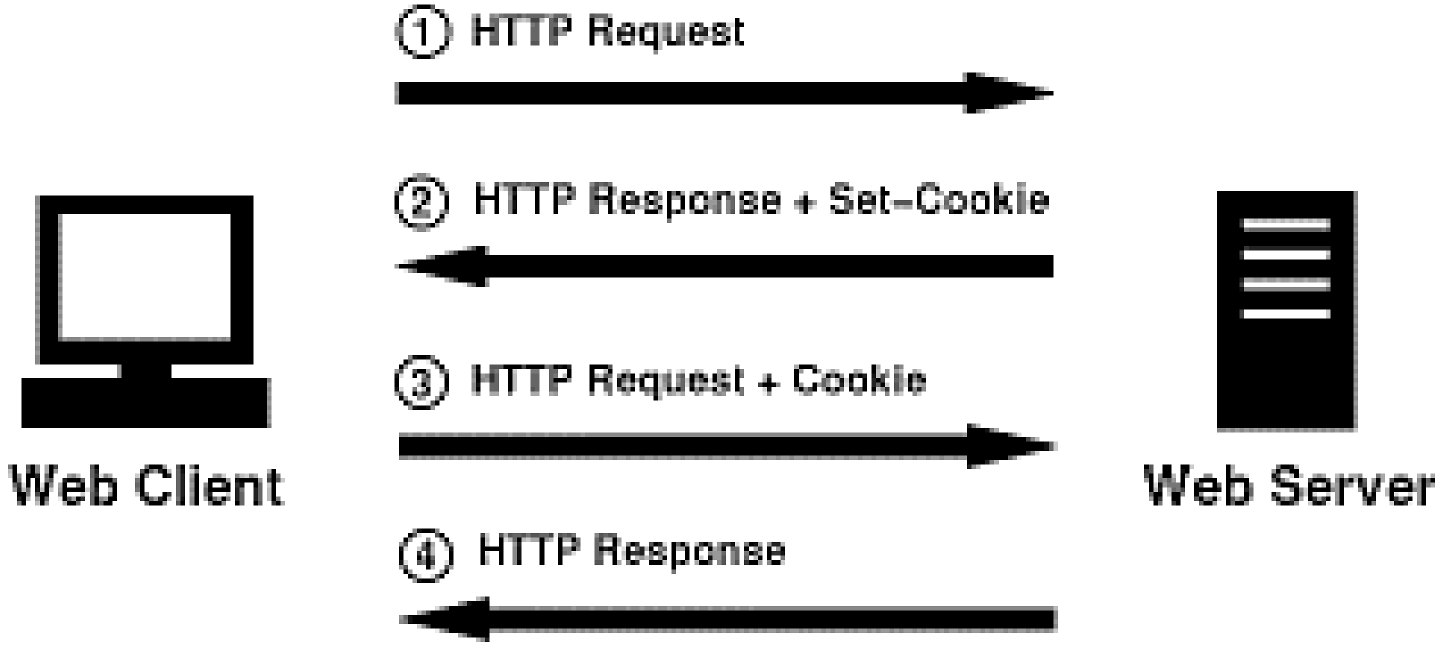
What is a Session?

- Session is a **temporary storage** at web server
- Each user, there is **unique session** are at server



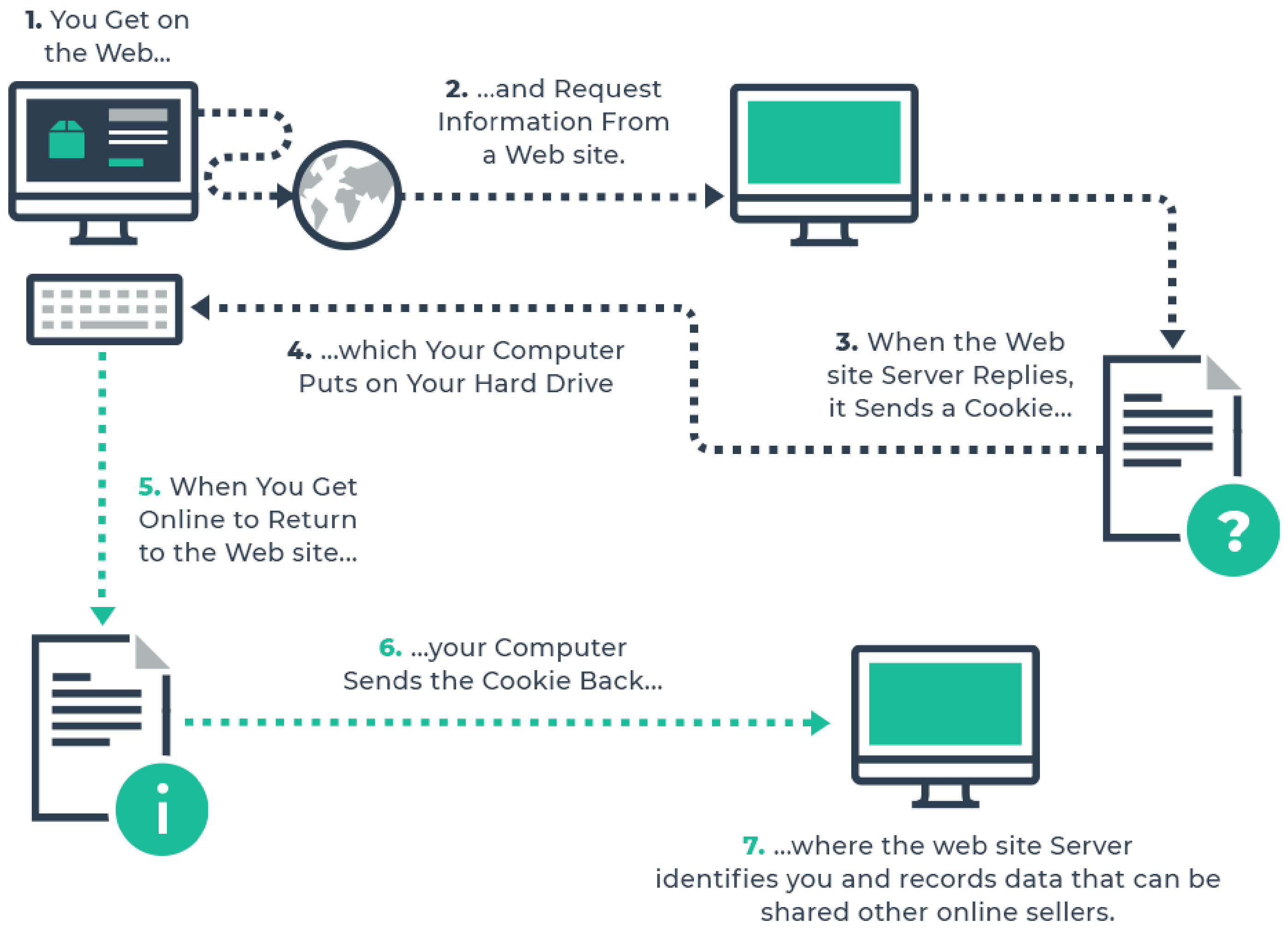
Session Management (Session Tracking)

- A way of maintaining the state of the user
- Four ways:
  - Cookie
  - Hidden form field
  - Url rewriting
  - HttpSession



Cookies

- Cookie: A **small piece of information** stored as a text file on client's machine
- This information is used to **identify new and previous users**
- Two types:
  - **Non-persistent** cookies
  - **Persistent** cookies


1. You Get on the Web...
2. ...and Request Information From a Web site.
3. When the Web site Server Replies, it Sends a Cookie...
4. ...which Your Computer Puts on Your Hard Drive
5. When You Get Online to Return to the Web site...
6. ...your Computer Sends the Cookie Back...
7. ...where the web site Server identifies you and records data that can be shared other online sellers.

Creating Cookies in Servlets

- **HttpServletResponse** interface's **addCookie(Cookie ck)** method is used.

Create a cookie object and add it to the response:

```
Cookie cookie=new Cookie("cookieName", "cookieValue");
response.addCookie(cookie);
```

Example:

```
Cookie cookie=new Cookie("user", "abc");
response.addCookie(cookie);
```

Accessing Cookies in Servlets

- HttpServletRequest interface's getCookies() method is used.

Get all cookie objects and iterate to get individuals:

```
Cookie[] ck = request.getCookies();
      if (ck != null) {
         for (Cookie cookie : ck) {
            if (cookie.getName().equals("user")) {
               String user = cookie.getValue();
}}}
```

Deleting/Removing Cookies in Servlets

- setMaxAge(*value*) method is used.
  - Value can be 0 or -1
  - If 0, the cookie will be immediately removed
  - If -1, it will be removed once the browser closed

Remove value and set expiration time to 0:

```
Cookie cookie = new Cookie(“cookieName”, “”);
cookie.setMaxAge(0);
response.addCookie(cookie);
```

Servlet Hidden Fields

- An input text with hidden type
- Simple approach

```
<input name="fieldname" value="fieldValue" type="hidden"/>
String value = request.getParameter("fieldname");
```

- Not secure
- The hidden value can be submitted only when a form is used

URL Rewriting

- A way of appending data at the end of URL

```
https://abc.com?param1=value1& param2=value2
String value = request.getParameter(“param1”);
```

- Not secure since the data available in the URL
- Cannot add many since URL has a character limit

HttpSession

- Inbuild architecture to identify a user in different requests
- This can create a session object using **HttpSession class**

```
HttpSession session = request.getSession();
session.setAttribute(“uname”, “abc”);

String value = (String) session.getAttribute(“uname”);

session.invalidate();
```