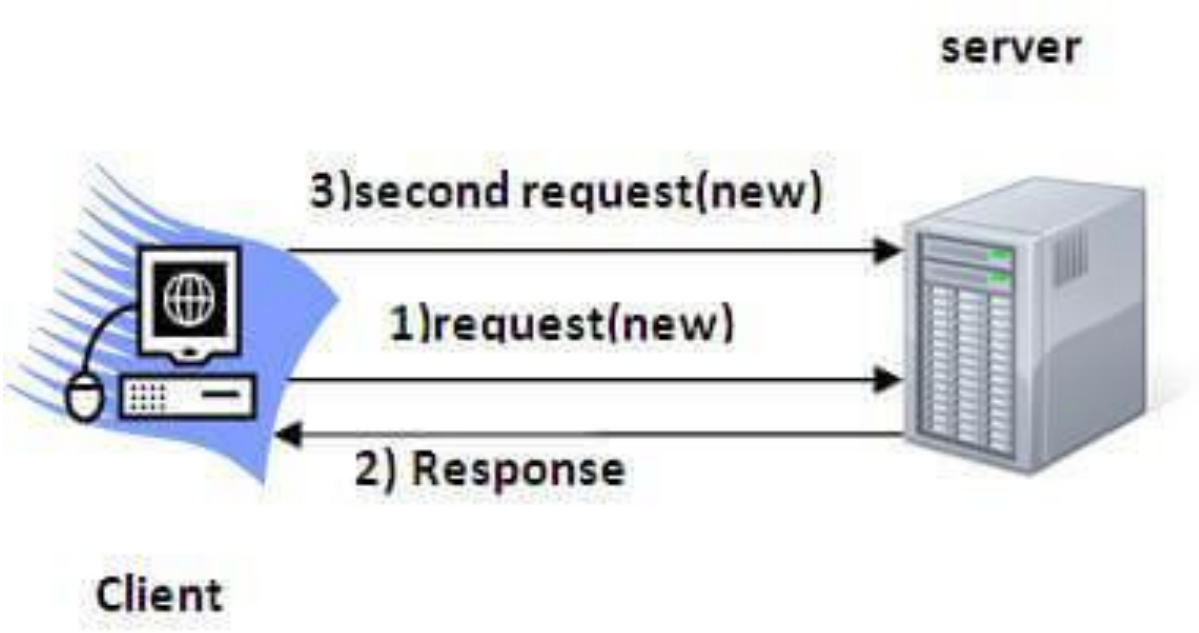
Lecture 2

Session Tracking in Servlet

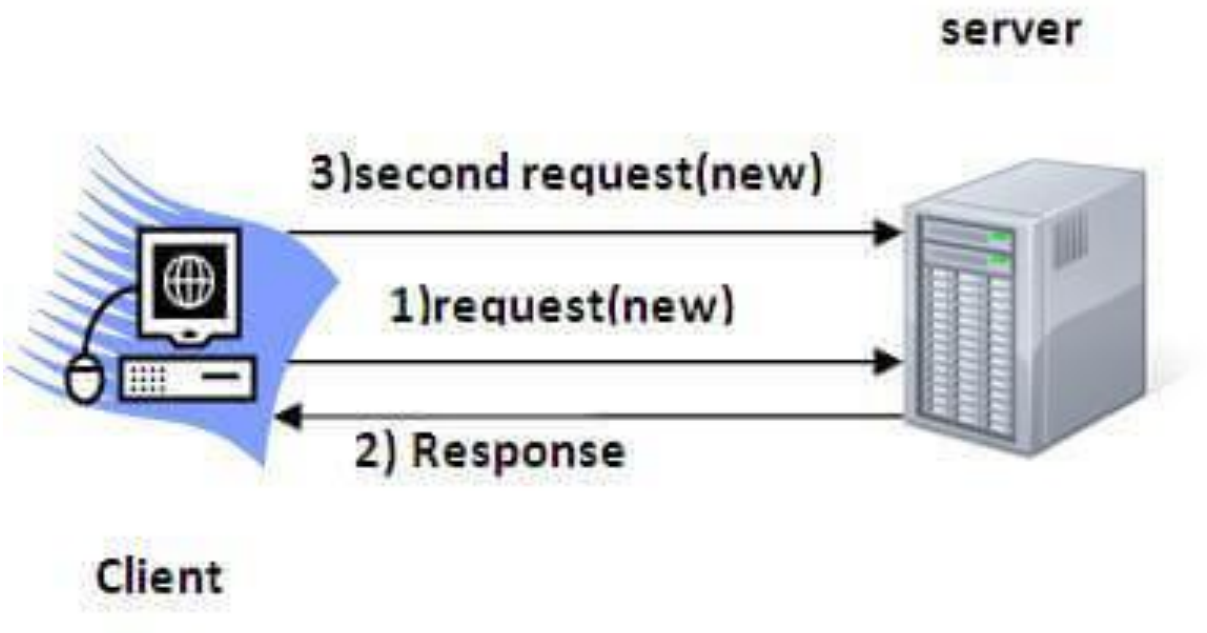
- Session simply means a particular interval of time.
- Session Tracking is a way to maintain state (data) of a user.
- It is also known as session management in servlet.



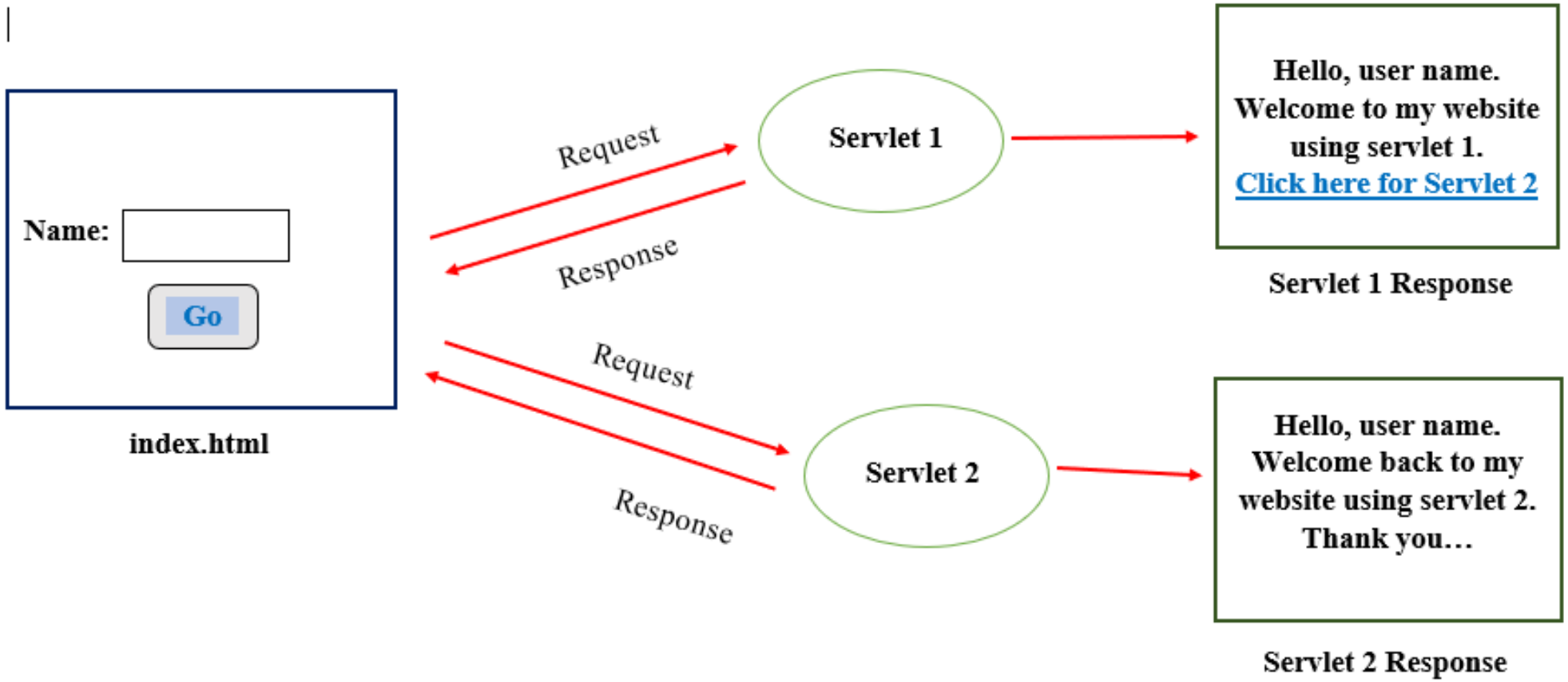
Session Tracking in Servlet

- Why use Session Tracking?
  - To recognize the user, It is used to recognize the particular user.

- Session Tracking Techniques
  - Cookies
  - Hidden Form Field
  - URL Rewriting
  - HttpSession



Why to track Session? Stateless Problem Example

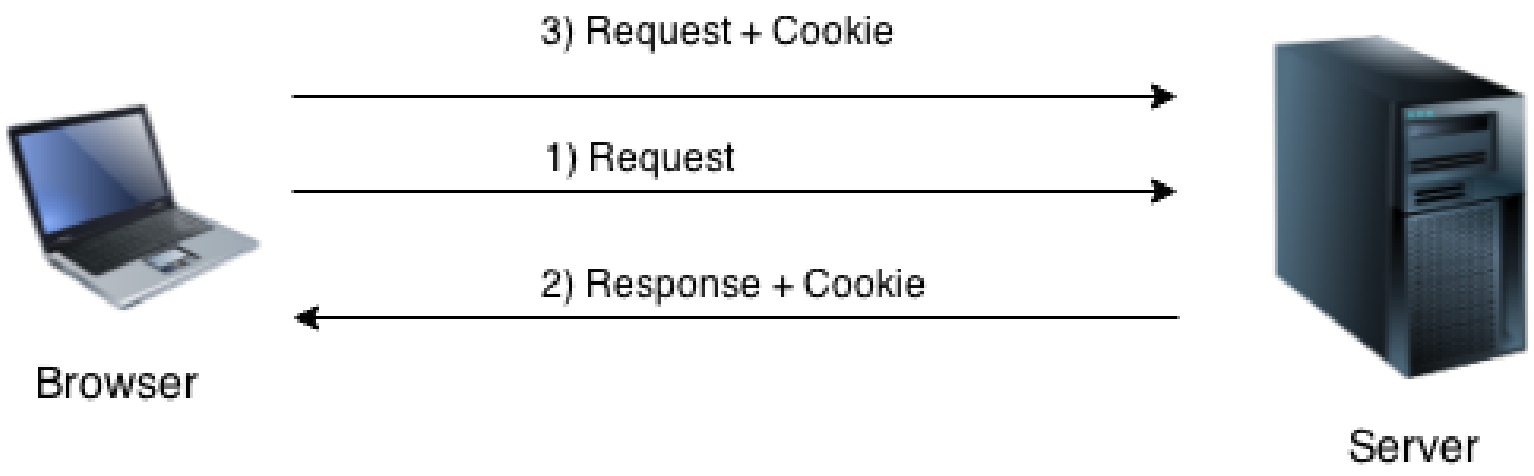


Cookies in Servlet

- Cookies are the textual information which are stored in a key value pair format to the clint's browser during multiple request.
- A cookie is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works?

- In cookies technique, we add cookie with response from the servlet.
- So, cookie is stored in the cache of the browser.
- After that if request is sent by the user, cookie is added with request by default.



Types of Cookies

- Non-persistent cookie
  - It is valid for single session only. It is removed each time when user closes the browser.
- Persistent cookie
  - It is valid for multiple session . It is not removed each time when user closes the browser. It is removed only if user logout or sign-out.

How to create Cookies in java?

- *javax.servlet.http.Cookie* class provides the functionality of using cookies.
- To create a cookie, just create an object of Cookie class and pass a name and value wit it.
- For example,

```
//creating cookie object
Cookie ck = new Cookie("user","sonoo jaiswal");
//adding cookie in the response
response.addCookie(ck);
```

How to get Cookies?

```
Cookie ck[]=request.getCookies();
If(ck==null){
        // It is a new user
}
else{
        for(int i=0;i<ck.length;i++)
            {
                String tname = ck[i].getName();
```

```
if(tname.equals("user_name"))
                    {
                        f=true; //It is a old user.
                        name=ck[i].getValue();
                    }
                }
}
```

How to delete Cookie?

- //deleting value of cookie

```
Cookie ck=new Cookie("user","");
```

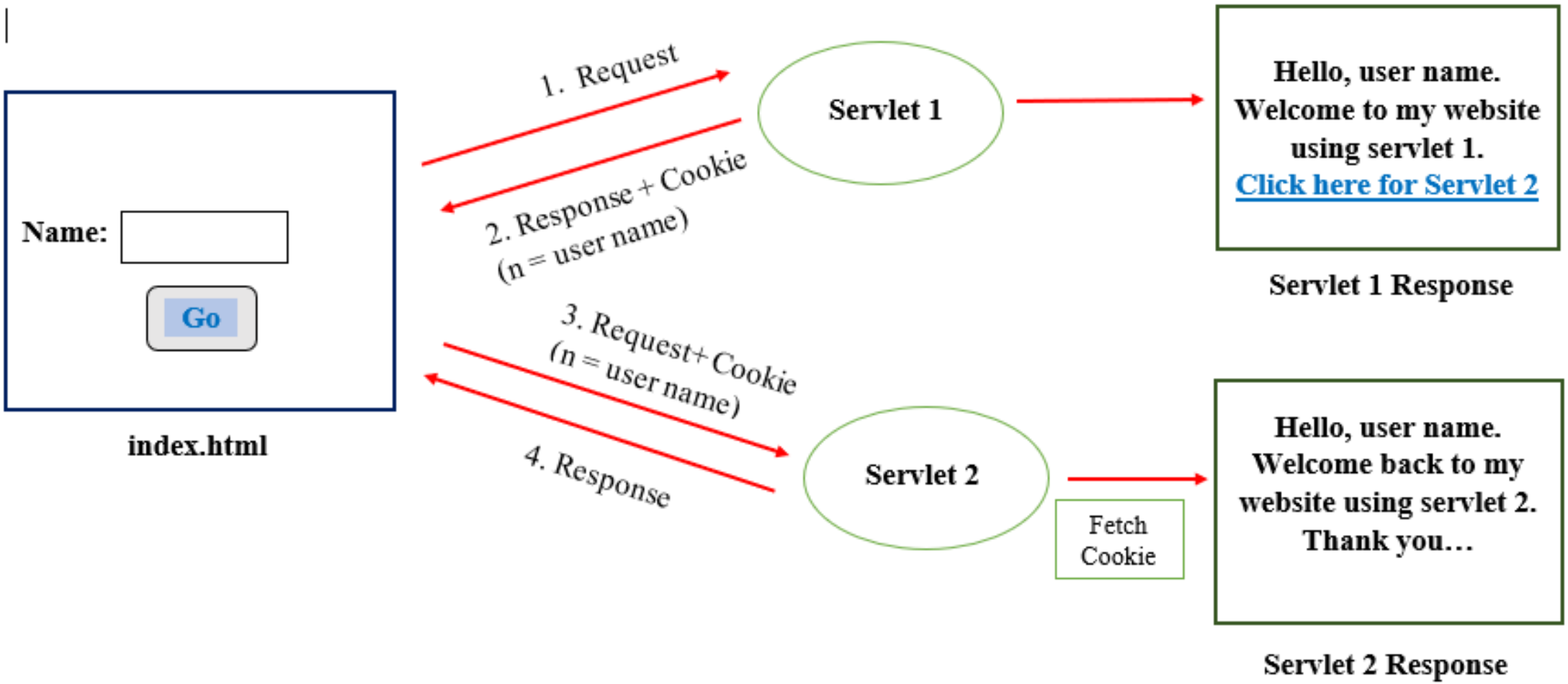
- //changing the maximum age to 0 seconds

```
ck.setMaxAge(0);
```

- //adding cookie in the response

```
response.addCookie(ck);
```

Cookie Example

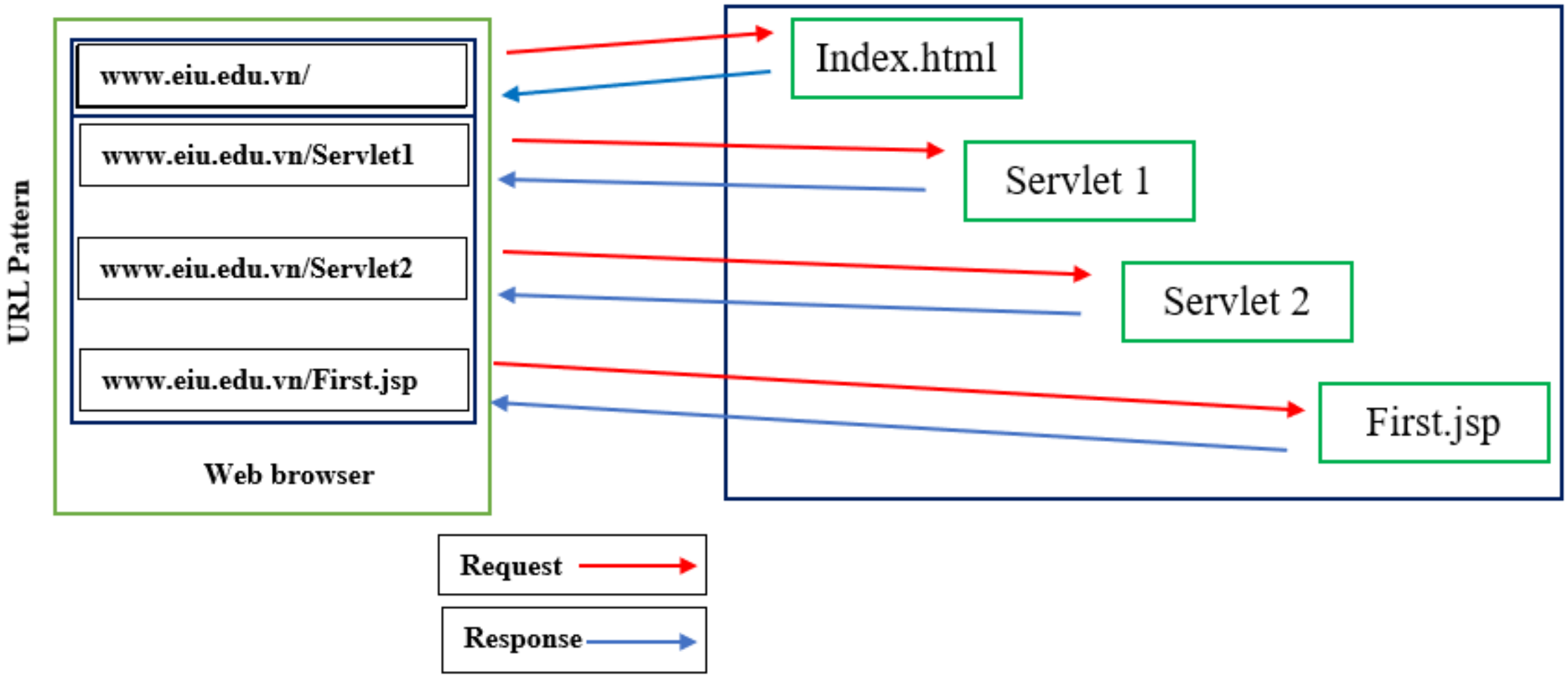


What is URL Rewriting?

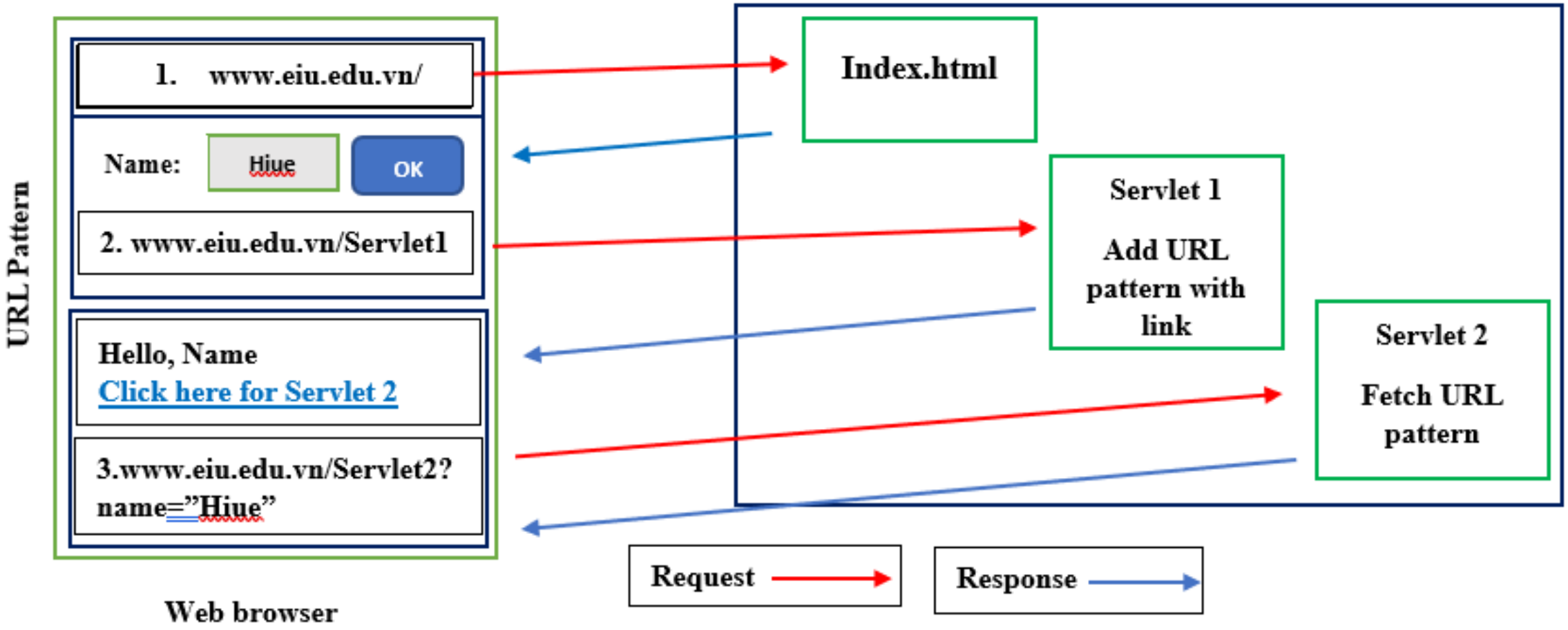
- It is a process of appending or modifying any URL structure while loading a page.
- URL rewriting is another way to support anonymous session tracking.
- With URL rewriting, every local URL the user might click on is dynamically modified, or rewritten, to include extra information.
- For example,

http://www.eiujava.com/MyFirstServlet?name="admin"&message="never give up!"

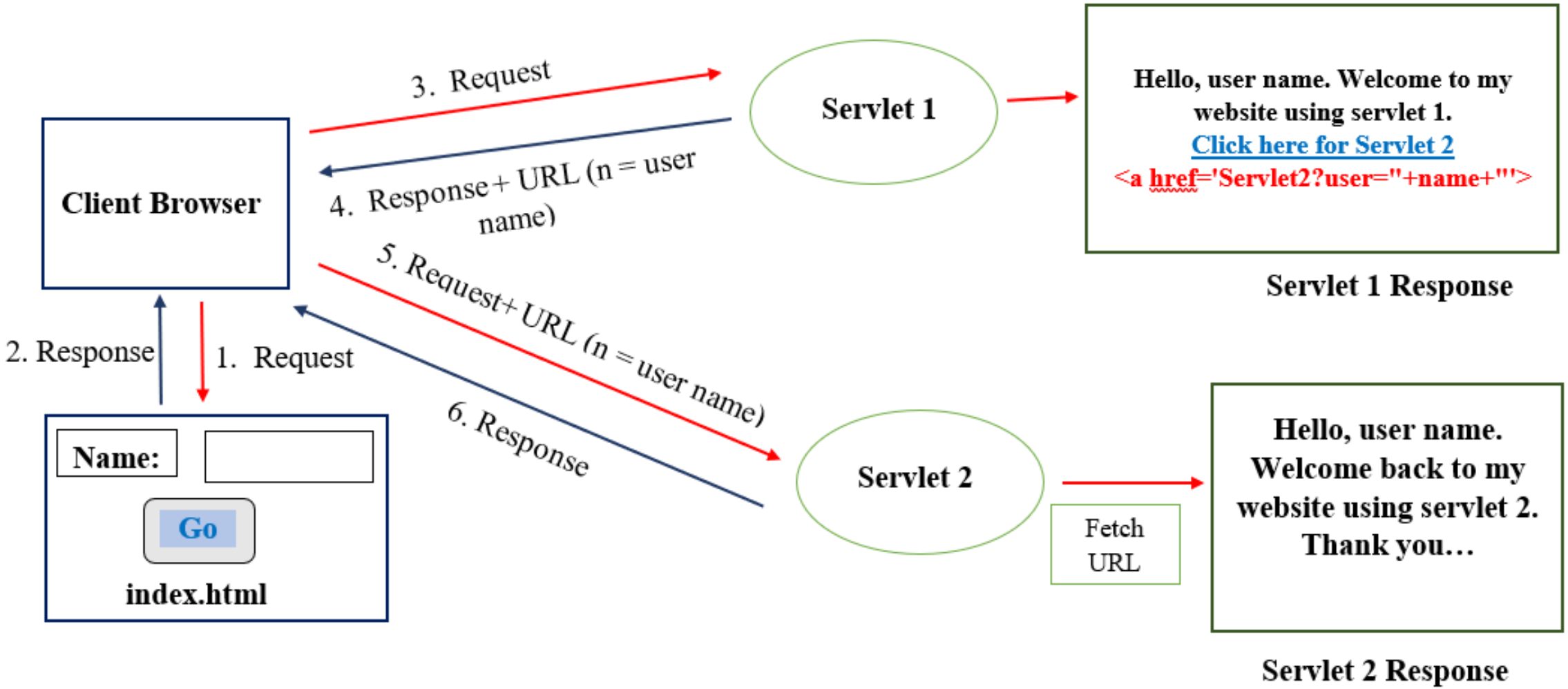
What is URL Rewriting?



What is URL Rewriting?



Example of URL Rewriting



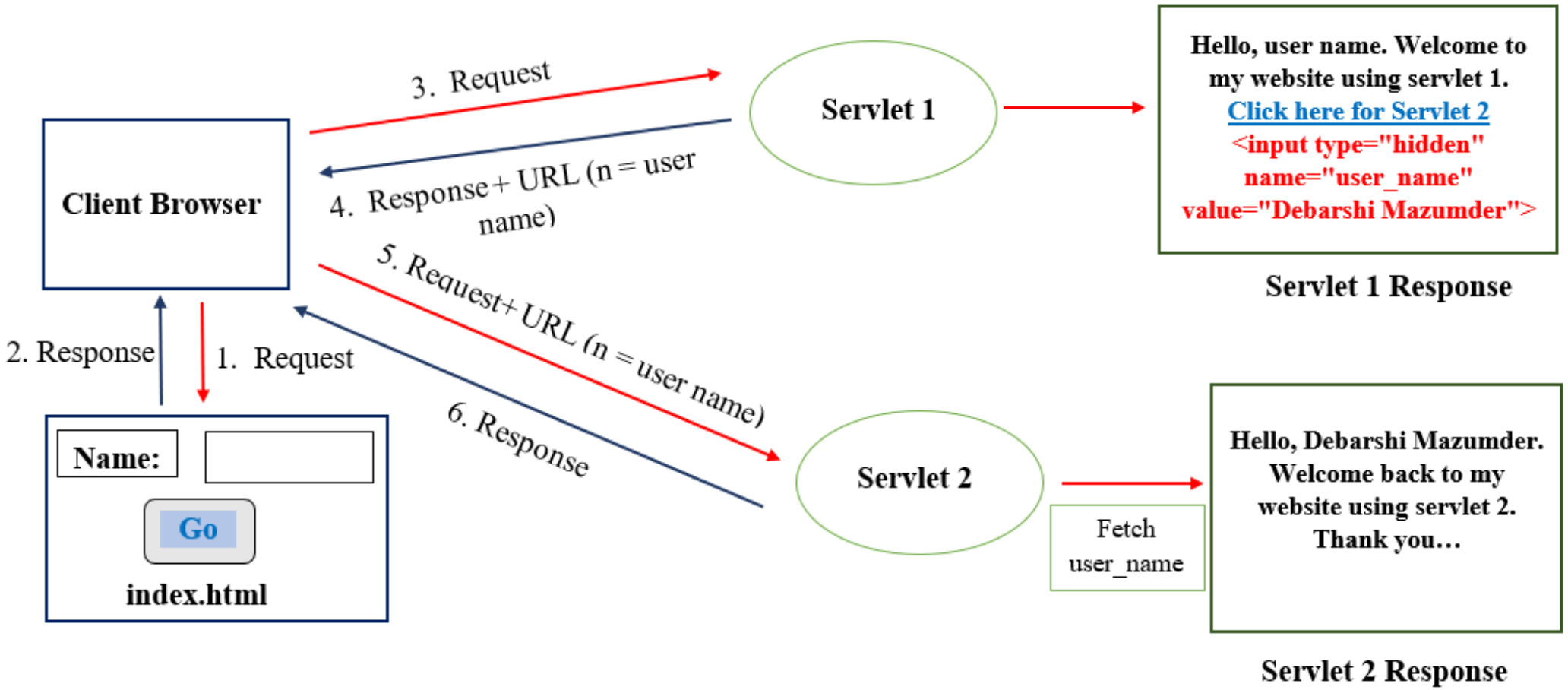
Example of URL Rewriting

- Servlet1
  - String name=request.getParameter("name");
  - out.println ("<h1>Click here for Servlet 2</h1>");
- Servlet 2
  - String name=request.getParameter("user");

Hidden Form Field

- Hidden Form Field a hidden (invisible) text field is used for maintaining the state of a user.

```
<input type="hidden" name="uname" value="Debarshi ">
```



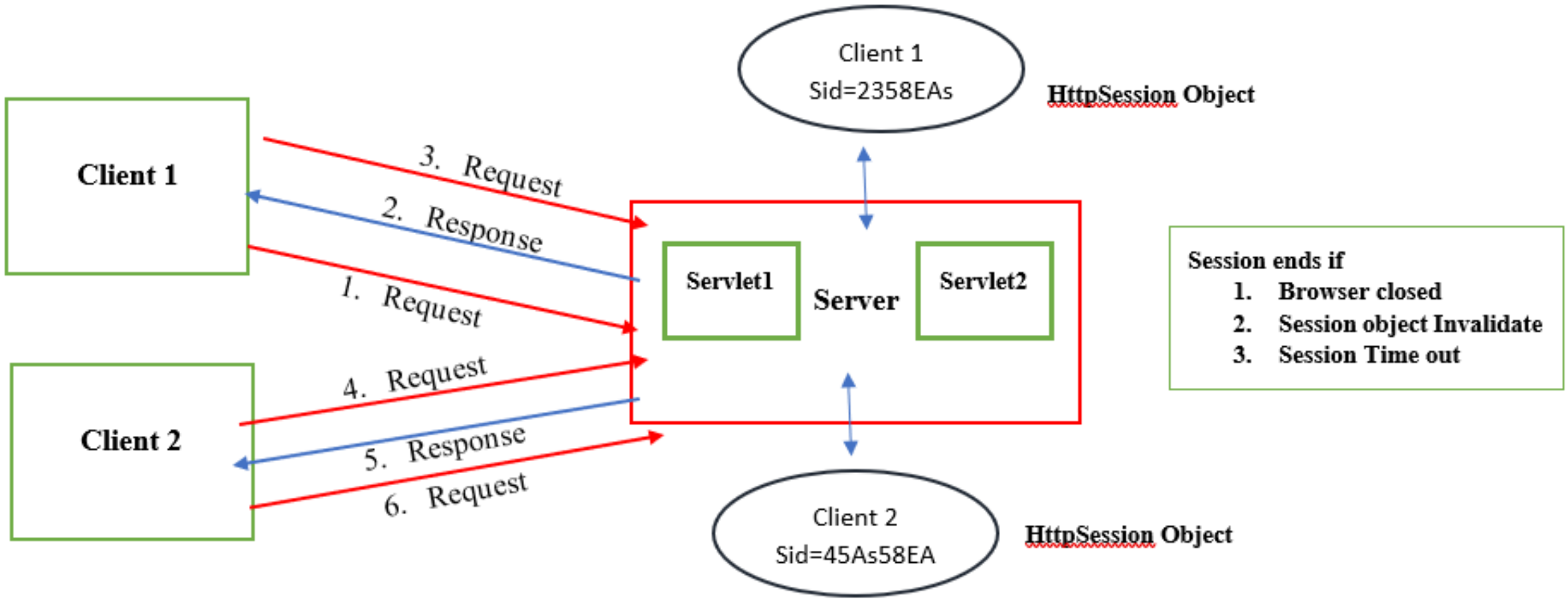
HttpSession

- HttpSession is an interface that provides a way to identify a user in multiple page requests.
- A unique session id is given to the user when first request comes.
- Servlet web container creates a session id for each user.
- This id is stored in a request parameter or in a cookie.
- The container uses this id to identify the particular user.

HttpSession

- An object of HttpSession can be used to perform two tasks:
  - Bind objects
  - View and manipulate information about a session, such as the session identifier, creation time, and last accessed time.

HttpSession



How to get the HttpSession object ?

- The HttpServletRequest interface provides two methods to get the object of HttpSession:
- **public HttpSession getSession()**
  - Returns the current session associated with this request, or if the request does not have a session, creates one.
- **public HttpSession getSession(boolean create)**
  - Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

Methods Of HttpSession Interface

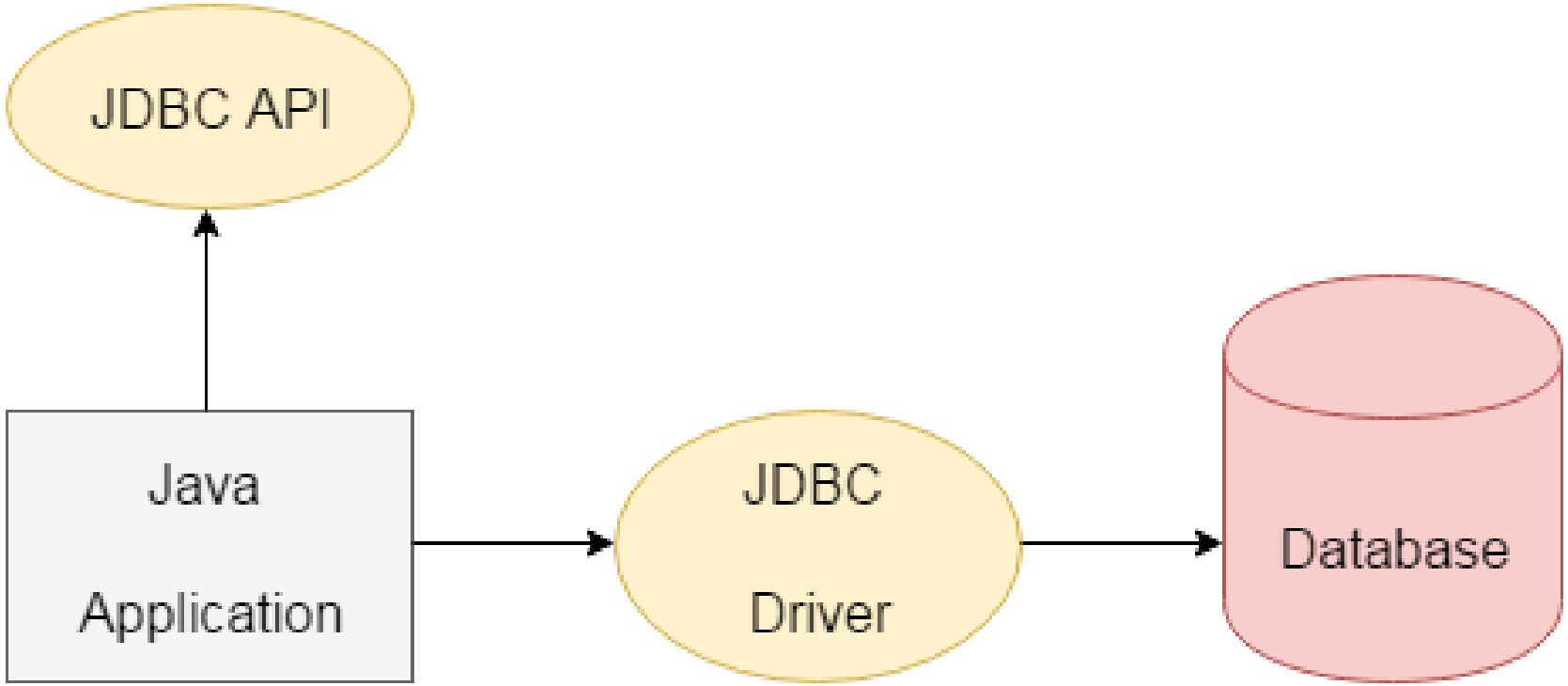
- public String getId()
  - Returns a string containing the unique identifier value.
- public long getCreationTime()
  - Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
- public long getLastAccessedTime()
  - Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970, GMT.
- public void invalidate()
  - Invalidates this session then unbinds any objects bound to it.

Java Database Connectivity (JDBC)

- JDBC stands for Java Database Connectivity.
- JDBC is a Java API to connect and execute the query with the database. It is a part of Java SE (Java Standard Edition).
- JDBC API uses JDBC drivers to connect with the database.
- There are four types of JDBC drivers:
  - JDBC-ODBC Bridge Driver,
  - Native Driver,
  - Network Protocol Driver, and
  - Thin Driver

Java Database Connectivity (JDBC)

- JDBC is a Java API to connect and execute the query with the database.
- It is a part of JavaSE (Java Standard Edition).



JDBC with 7 Steps

- There are 7 steps to connect any java application with the database using JDBC. These steps are as follows:
  - 1. Import the package
  - 2. Load & register the Driver.
  - 3. Create the Connection.
  - 4. Create the Statement.
  - 5. Execute queries
  - 6. Process the Results
  - 7. Close connection

JDBC with 7 Steps

- 1. import the package
  - import java.sql.*;
- 2. Load & register the Driver.
  - a. Load the MySQL driver.
    - com.mysql.jdbc.Driver
    - using mySQL-connector
  - b. Register the driver
    - Class.forName(“com.mysql.jdbc.Driver");
- The forName() method of class is used to register the driver class. This method is used to dynamically load the driver class.

JDBC with 7 Steps

- 3. Create the Connection.
  - The getConnection() method of DriverManager class is used to establish connection with the database.
  - Connection con = DriverManager.getConnection (dbURL + dbName, dbUsername, dbPassword);
  - String dbDriver = "com.mysql.jdbc.Driver";
  - String dbURL = "jdbc:mysql://localhost:3306/ ";
  - String dbName = “debarshi”
  - String dbUsername = "debarshi";
  - String dbPassword = "root";

JDBC with 7 Steps

- 4. Create the Statement.
  - The createStatement() method of Connection interface is used to create statement.
  - The object of statement is responsible to execute queries with the database.
  - Statement st=con.createStatement();
- Types of Statement
  - Statement
  - Prepared Statement
  - Callable Statement

JDBC with 7 Steps

- 5. Execute queries
  - The executeQuery() or executeUpdate(); method of Statement interface is used to execute queries to the database.
  - This method returns the object of ResultSet that can be used to get all the records of a table.
  - select * from emp
- 6. Process the Results
  - ResultSet rs=st.executeQuery("select * from emp");
- 7. Close connection
  - st.close();
  - con.close();

ServletConfig and ServletContext

- ServletConfig and ServletContext, both are objects created at the time of servlet initialization and used to provide some initial parameters or configuration information to the servlet.
- The main difference lies in between these is:
  - Information shared by ServletConfig is for a specific servlet,
  - while information shared by ServletContext is available for all servlets in the web application.

ServletConfig

- An object of ServletConfig is created by the web container for each servlet.
- This object can be used to get configuration information from web.xml file.
- Advantage of ServletConfig
  - The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

ServletConfig Example

- ***NewServletConfig.java***

```
ServletContext sct = getServletContext();
String name = sct.getInitParameter("name");
```

- ***Web.xml***

```
<servlet>
        <servlet-name>NewServletConfig</servlet-name>
        <servlet-class>com.servlet.NewServletConfig</servlet-class>
        <init-param>
                <param-name>phone</param-name>
                <param-value>0902367175</param-value>
        </init-param>
</servlet>
```

ServletContext

- ServletContext is the object created by Servlet Container to share initial parameters or configuration information to the whole application.
- If any information is shared to many servlet, it is better to provide it from the web.xml file using the <context-param> element.
- Advantage of ServletContext
  - Easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet.

ServletContext Example

- ***ServletContext.java***

```
ServletContext sct=getServletContext();
String name = sct.getInitParameter("name");
```

- ***Web.xml***

```
<?xml version="1.0" encoding="UTF-8"?> <web-app version="3.1"
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">
  <!--
      No declaration & No mapping
  -->
</web-app>
```

Annotation Example

- ***NewServletContext.java***

```
import javax.servlet.annotation.WebServlet;
@WebServlet("/NewServletContext")
```

- ***NewServletContext.java***

```
import javax.servlet.annotation.WebServlet;
@WebServlet("/NewServletContext")
```