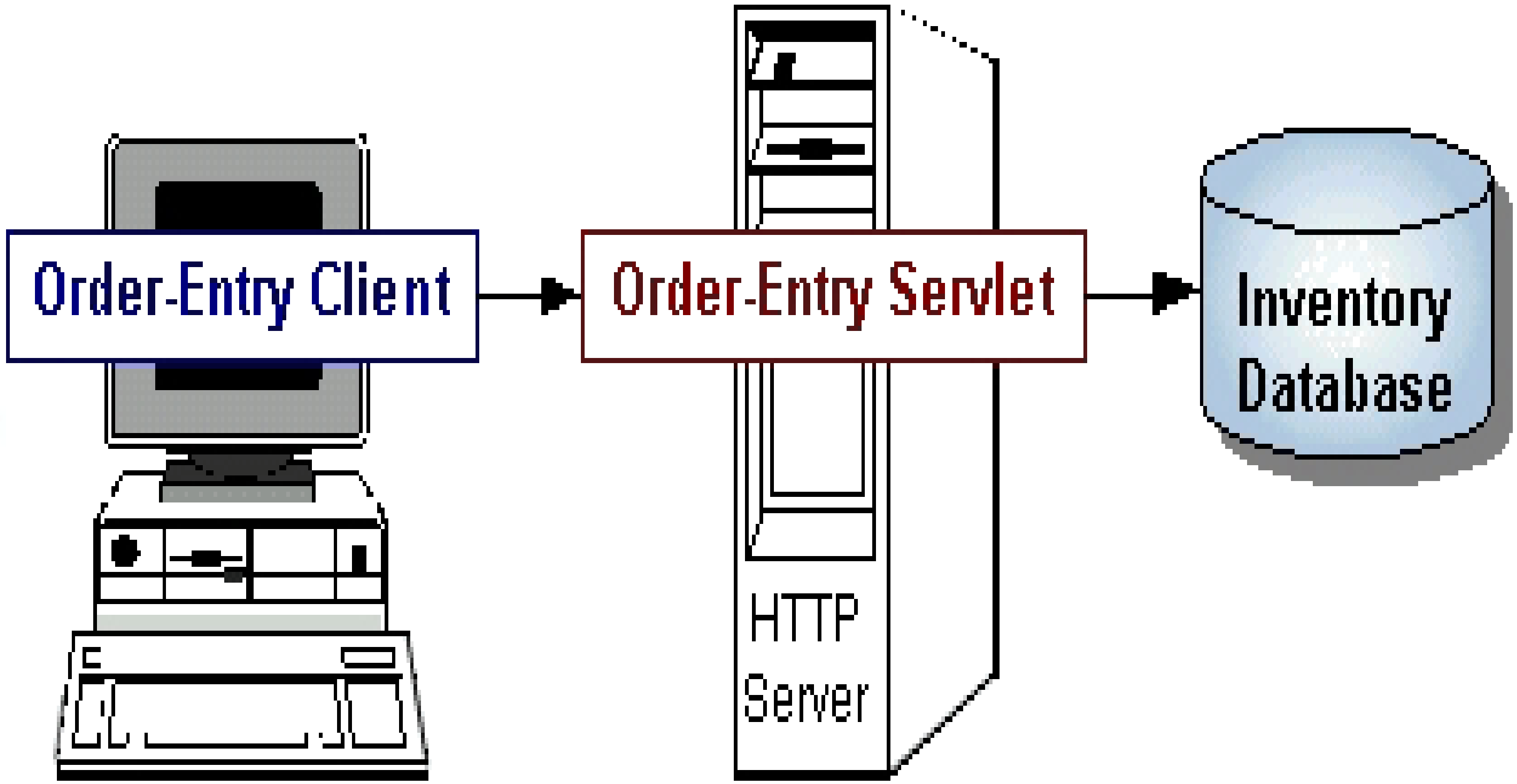
Servlet API

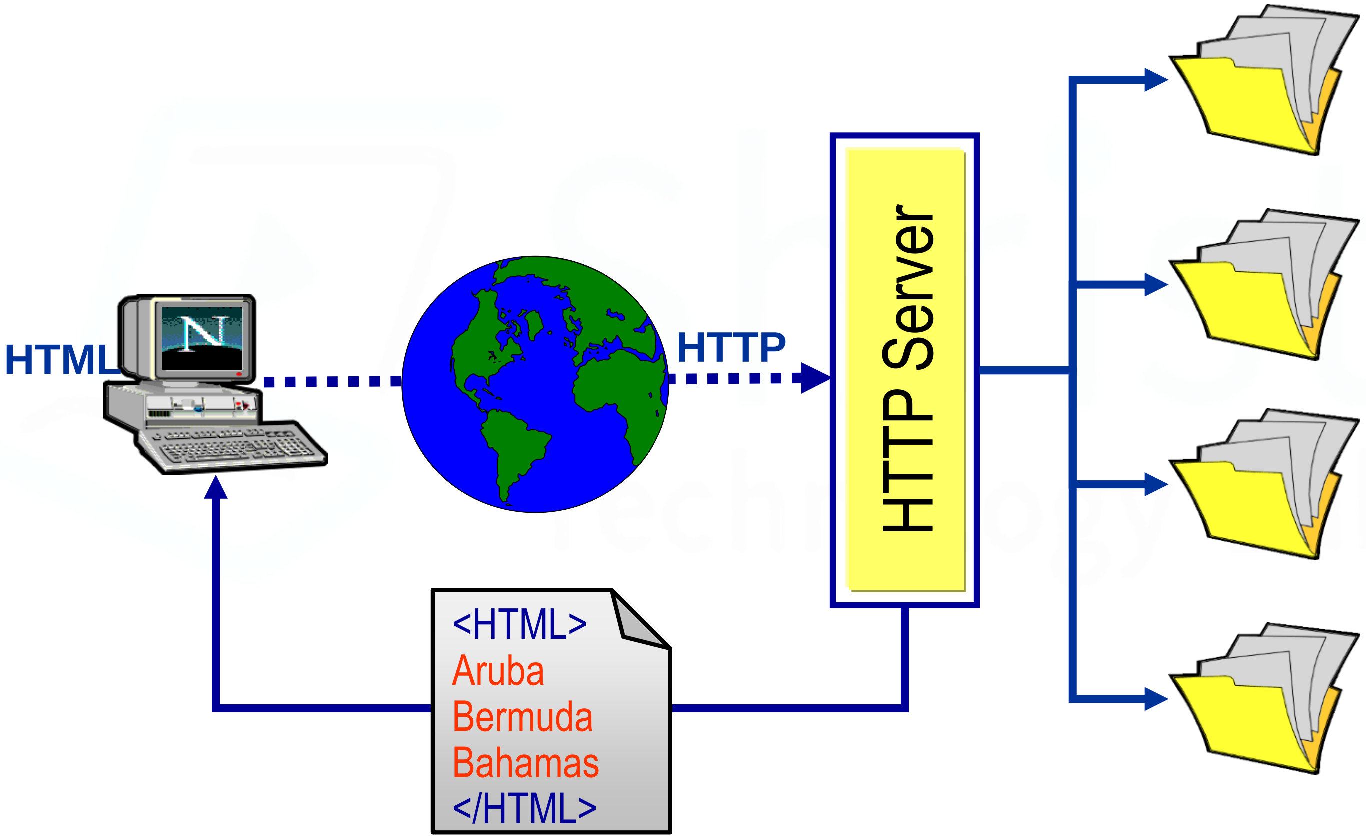
Shristi Technology Labs

Contents

Web Application Flow

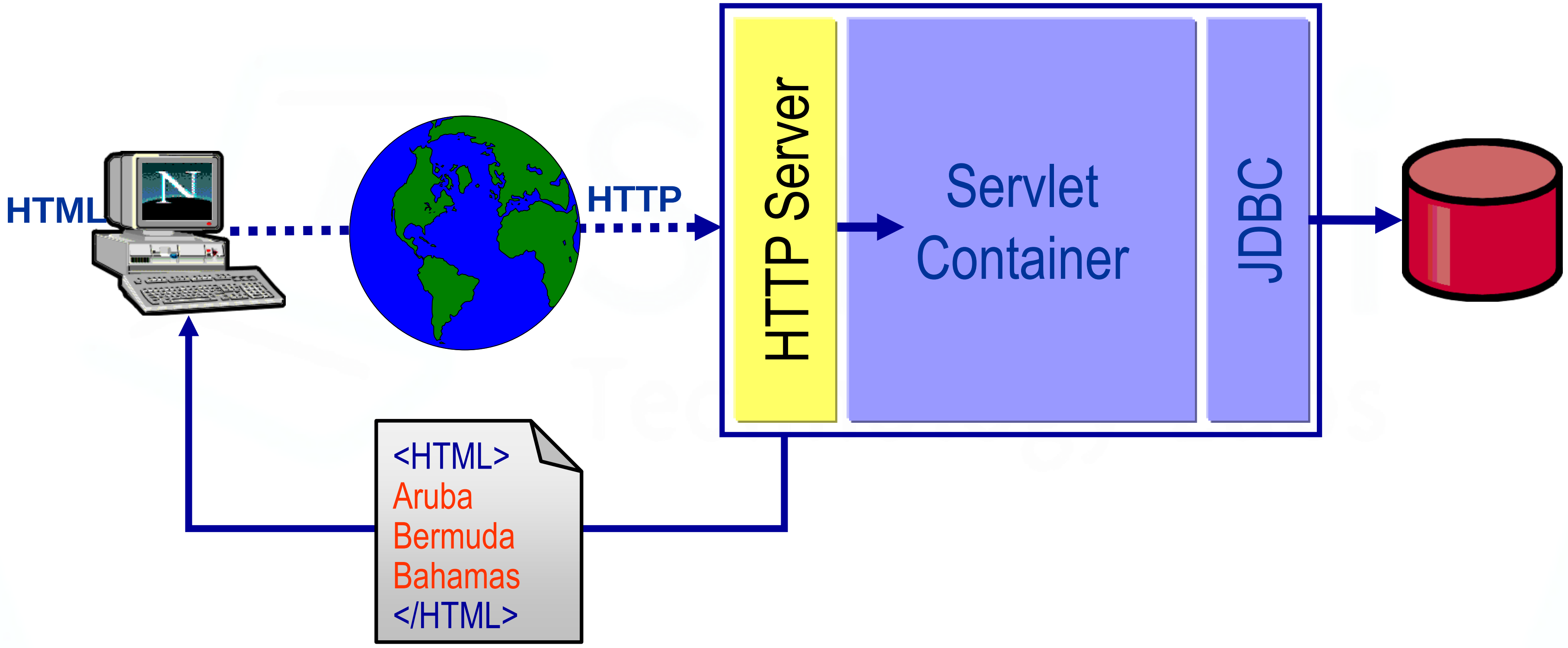


Static Web Site



<HTML>
Aruba
Bermuda
Bahamas
</HTML>

Dynamic page generation using servlets



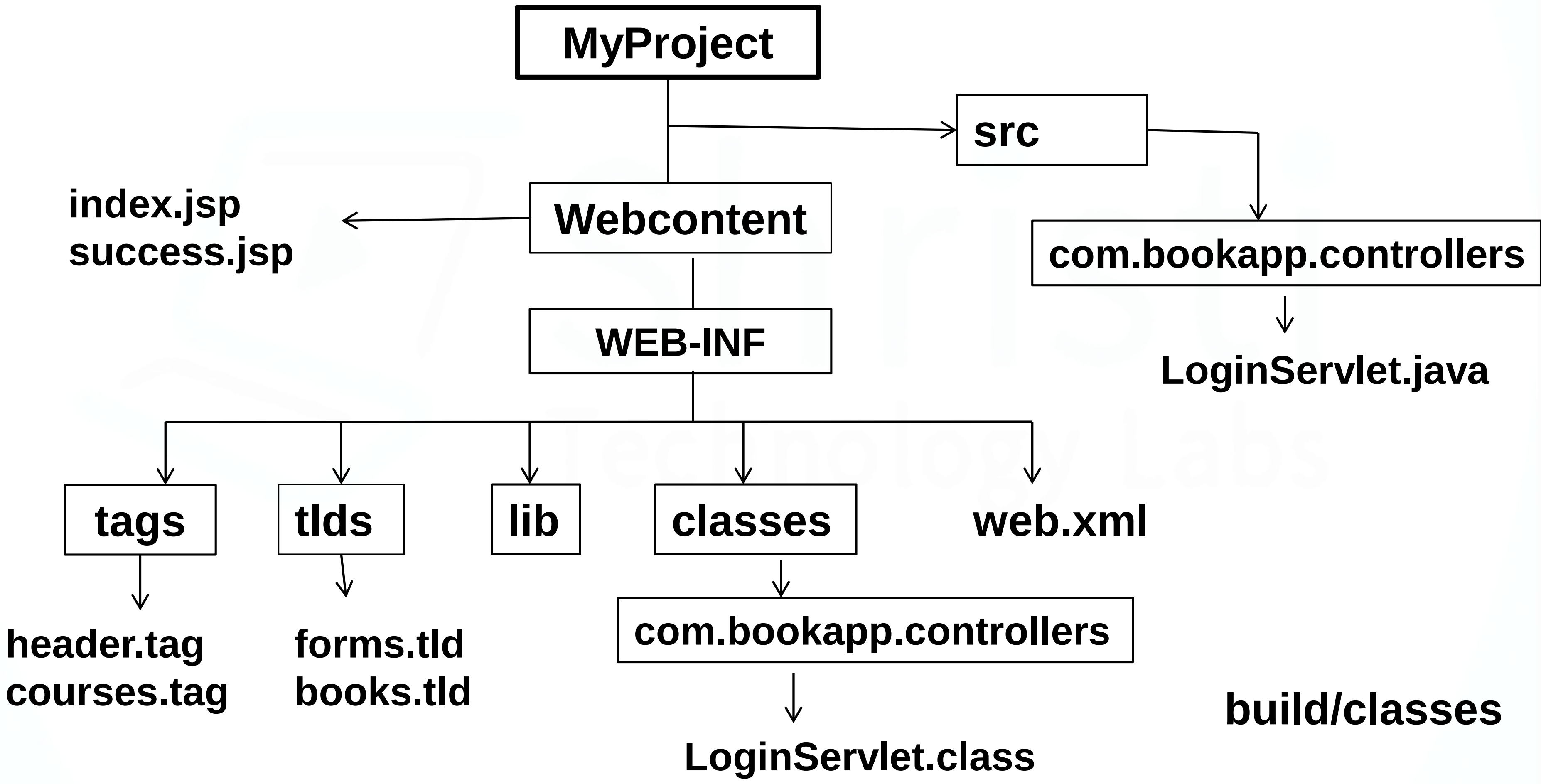
What is a Servlet?

- Server side Java program that extends the functionality of a Web Server
- Used to dynamically generate HTML documents
- Servlets run on the web server platform as part of the same process as the web server itself.
- The web server is responsible for initializing, invoking, and destroying each servlet instance.

Advantages of Servlet

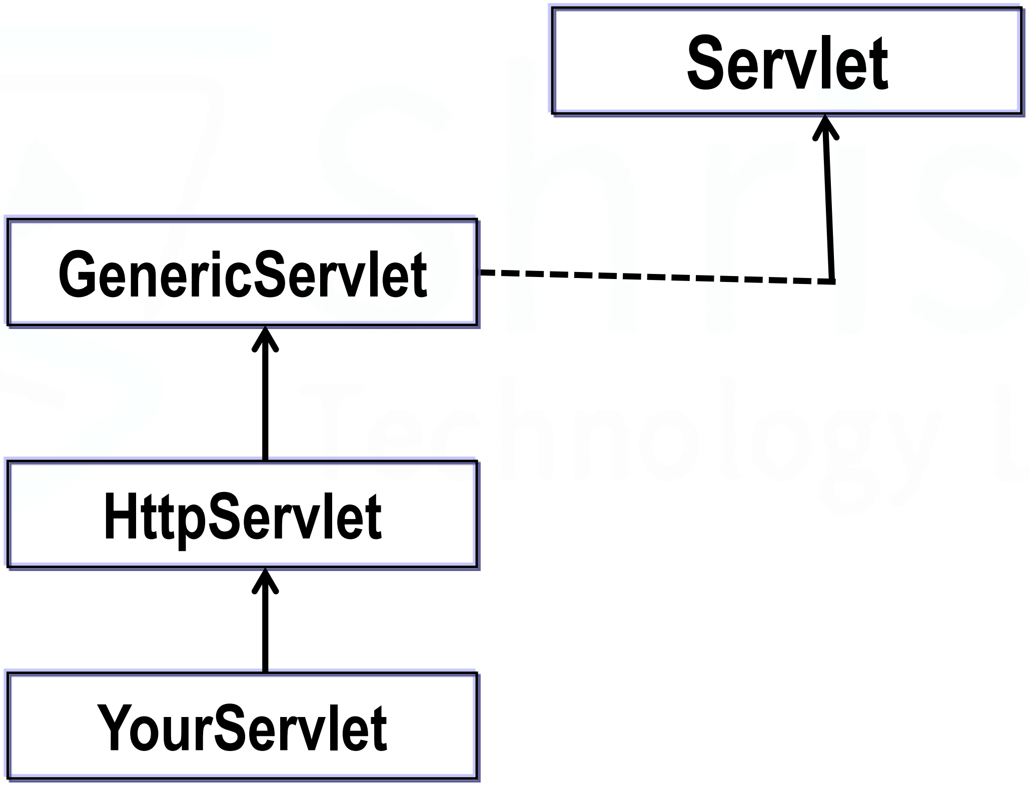
- Written in pure Java
  - Platform independent
  - Can take advantage of Java APIs
- Server independent
- Scalability
  - Do not start new process for each request
  - Can run in same server process as HTTP server
  - Multi-threaded

Structure of web application

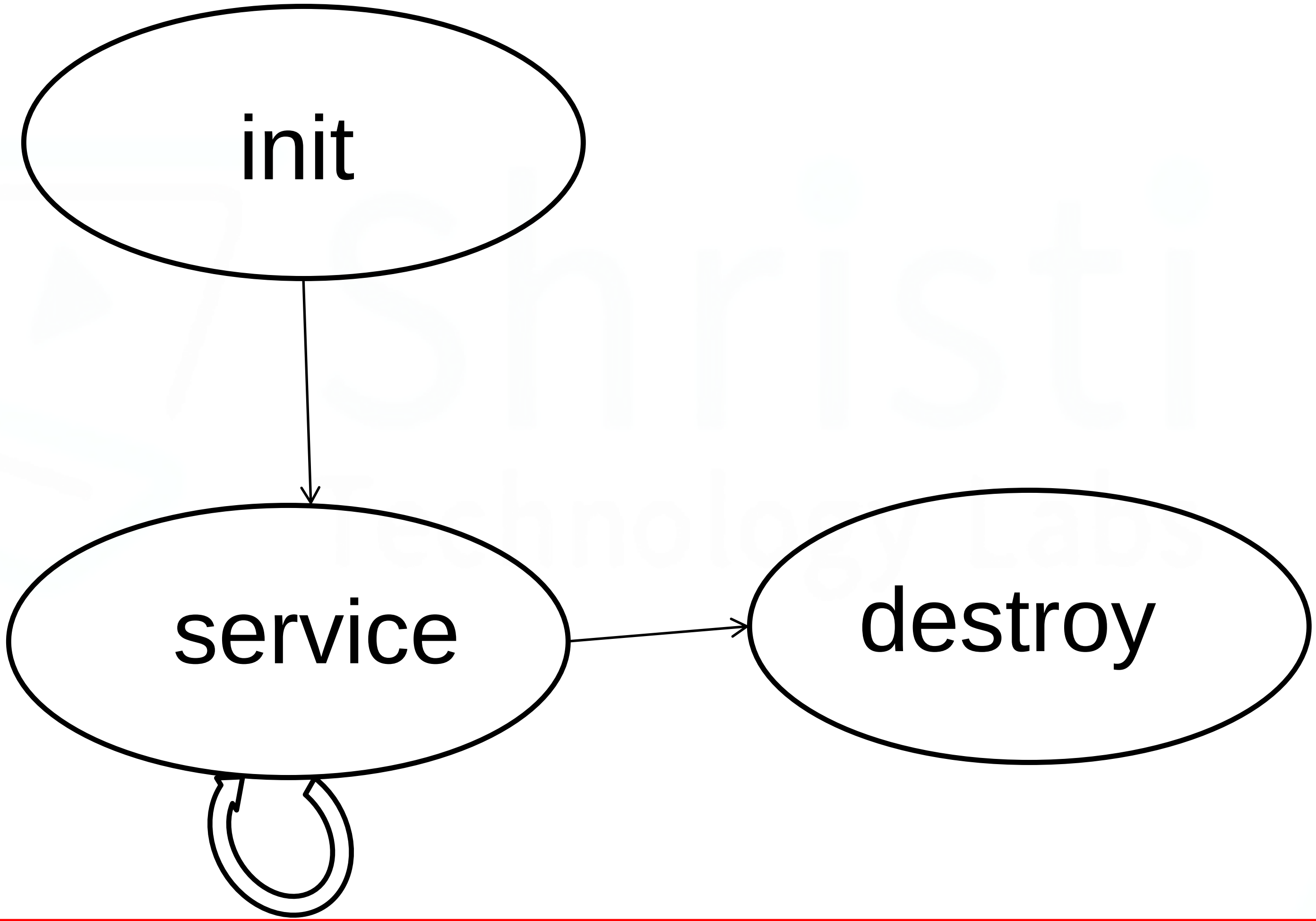


Servlet API

- Create a servlet that extends HttpServlet



Servlet Lifecycle



Lifecycle methods

public void init() throws ServletException

- is for initialization of the servlet and called only once during the life of a servlet
- To provide configuration details that can be shared by multiple clients

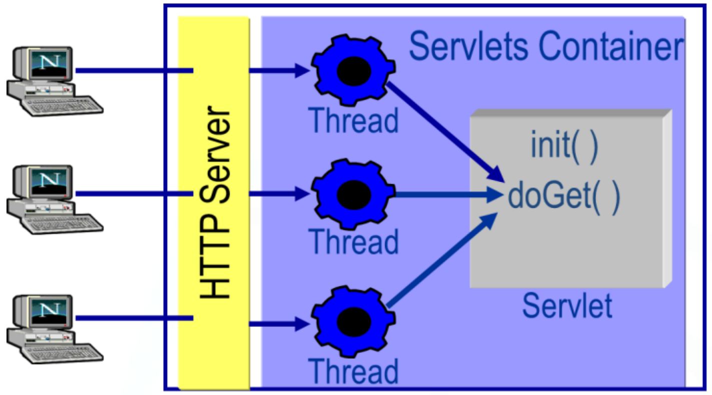
public void Service(ServletRequest, ServletResponse) throws ServletException, IOException

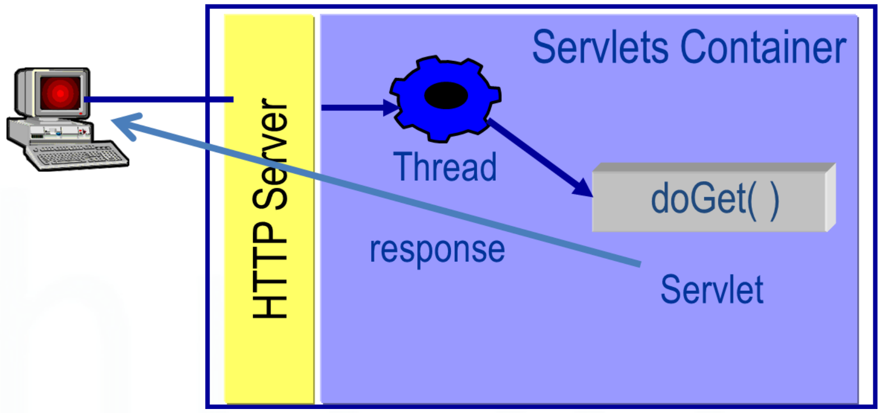
- is called once per client.
- Reads the request and produces the response message from its two parameters: ServletRequest object, ServletResponse Object

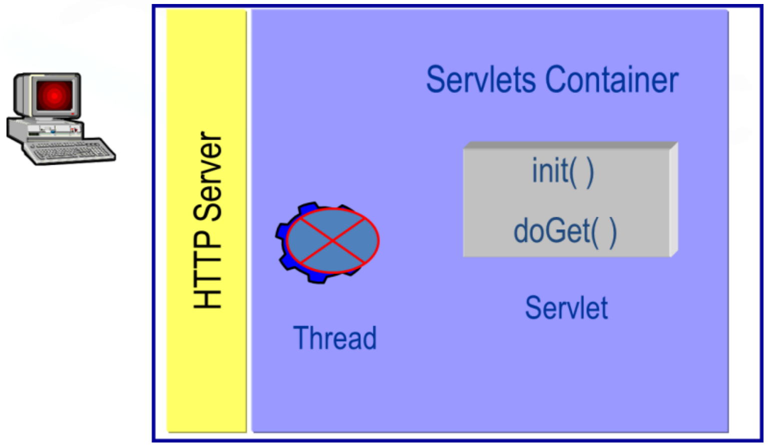
public void destroy()

- Is called to destroy the servlet instance and release the resources
- Is called once during the lifecycle of a servlet

Handling Requests using Servlet







Servlet Interface

- A servlet is Java class that implements the javax.servlet.Servlet interface
- This interface defines only five methods:
  - service()
  - init()
  - getServletConfig()
  - destroy()
  - getServletInfo()

HttpServlet

- Can be used with Http protocol
- Processing and/or storing data submitted by an HTML form
- Providing dynamic content
- Managing state information
- Has seven methods to override service method

Methods of HttpServlet

- doGet - Requests data from a specified resource
- doPut - to upload data to be processed to a specified resource
- doPost - Submits data to be processed to a specified resource
- doTrace - to acknowledge back what we sent
- doHead - to get HTTP header details only
- doDelete - to delete the specified resource
- doOptions - Returns the HTTP methods that the server supports

Takes two arguments

- **An HttpServletRequest object**, encapsulates the request from the client
- **An HttpServletResponse object**, encapsulates the response to the client

Reading Servlet Parameters

Methods to retrieve values from the form

From input form fields

- getParameter(String pname)

From check box

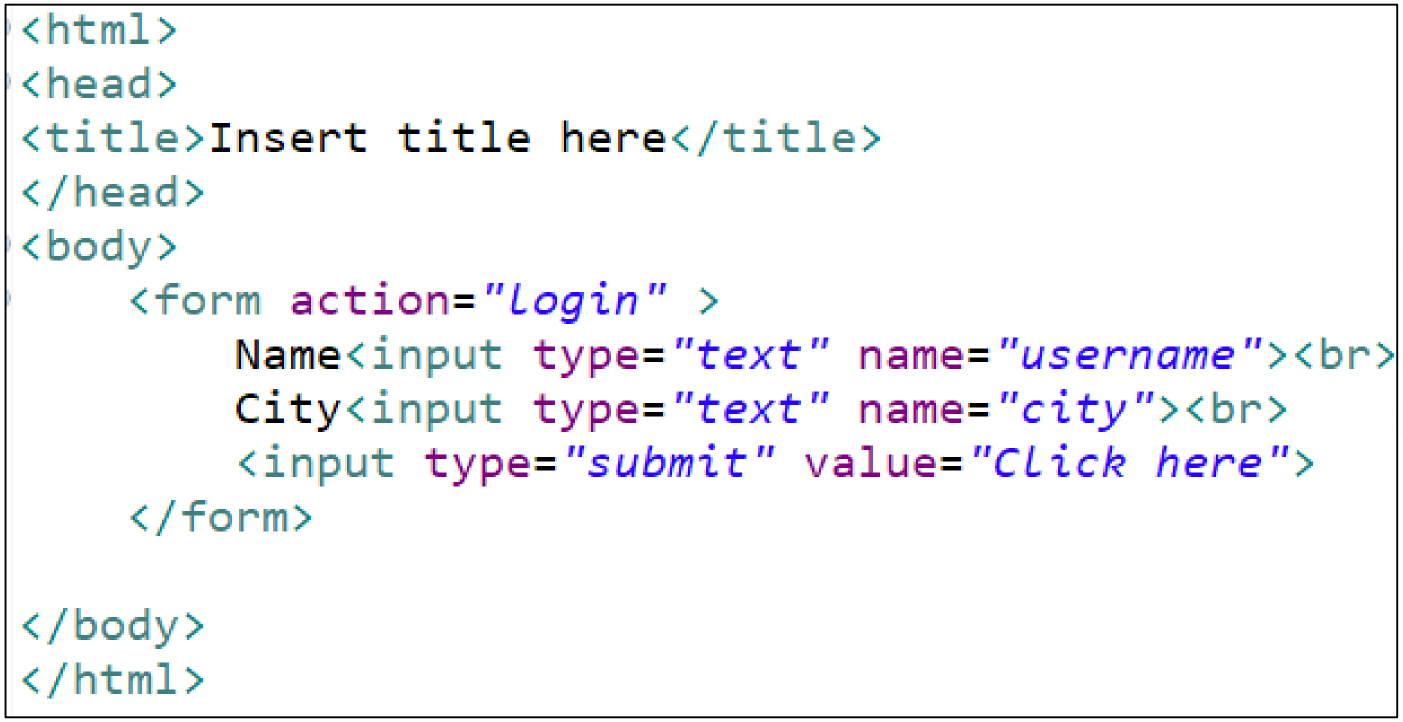
- getParameterValues(String pname)

Example

Create the following files

- index.jsp(WebContent)
- Login.java (src/com/training/controllers)- Servlet

index.jsp

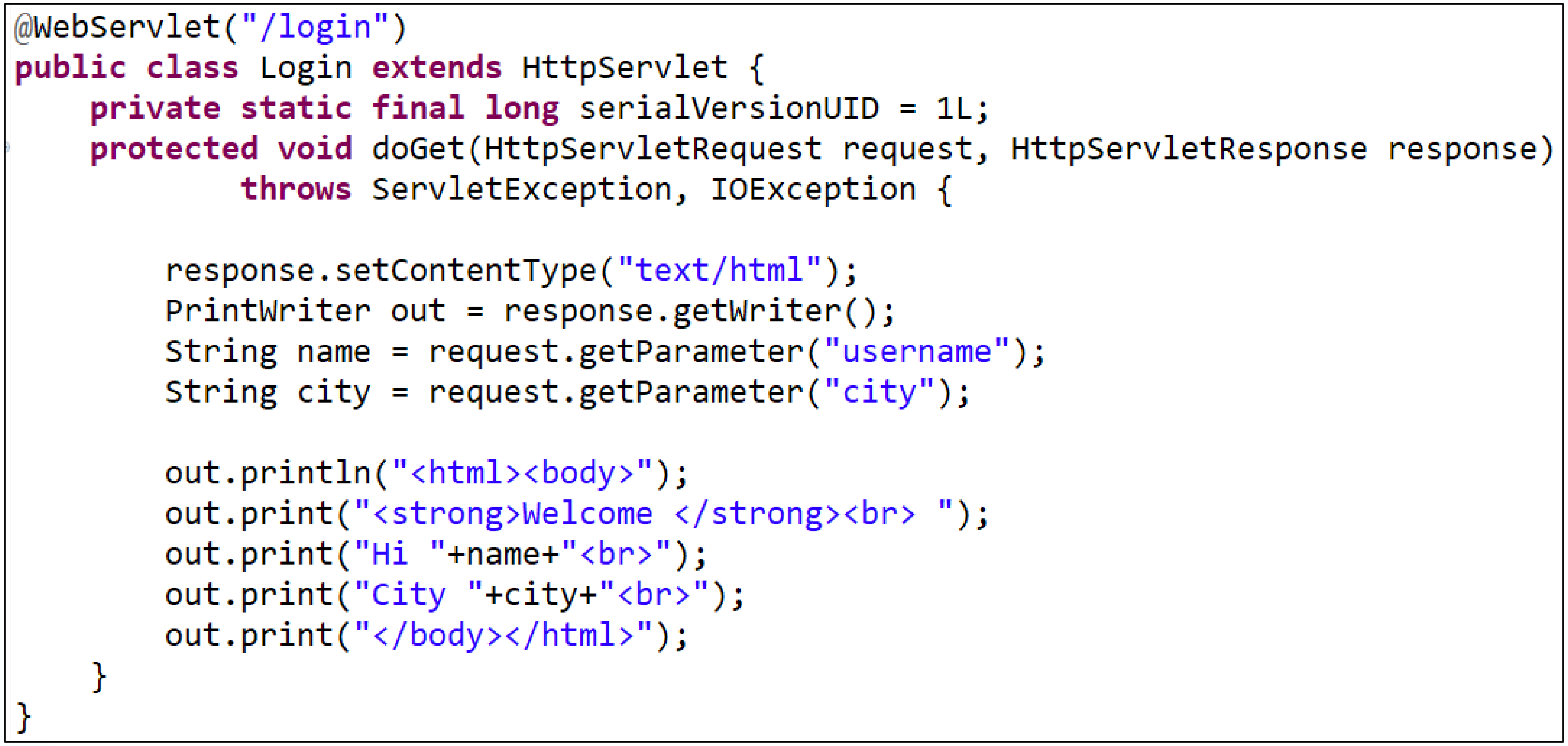


```
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <form action="login" >
        Name<input type="text" name="username"><br>
        City<input type="text" name="city"><br>
        <input type="submit" value="Click here">
    </form>

</body>
</html>
```

default method : get

Login.java(Servlet)



```java
@WebServlet("/login")
public class Login extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("username");
        String city = request.getParameter("city");

        out.println("<html><body>");
        out.print("<strong>Welcome </strong><br> ");
        out.print("Hi "+name+"<br>");
        out.print("City "+city+"<br>");
        out.print("</body></html>");
    }
}
```

Request Headers

Header Name	Header Value(s)
accept	*/*
accept-language	en-us
user-agent	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; InfoPath.2; MS-RTC LM 8)
accept-encoding	gzip, deflate
host	localhost:8080
connection	Keep-Alive
cache-control	no-cache

Shristi Technology Labs

Response Headers

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ... ...
HeaderN: ...
(Blank Line)
<!doctype ...>
<html> <head>...</head>
<body> ... </body>
 </html>
```

get Vs post

get

- Has only header
- all form values carried in query string

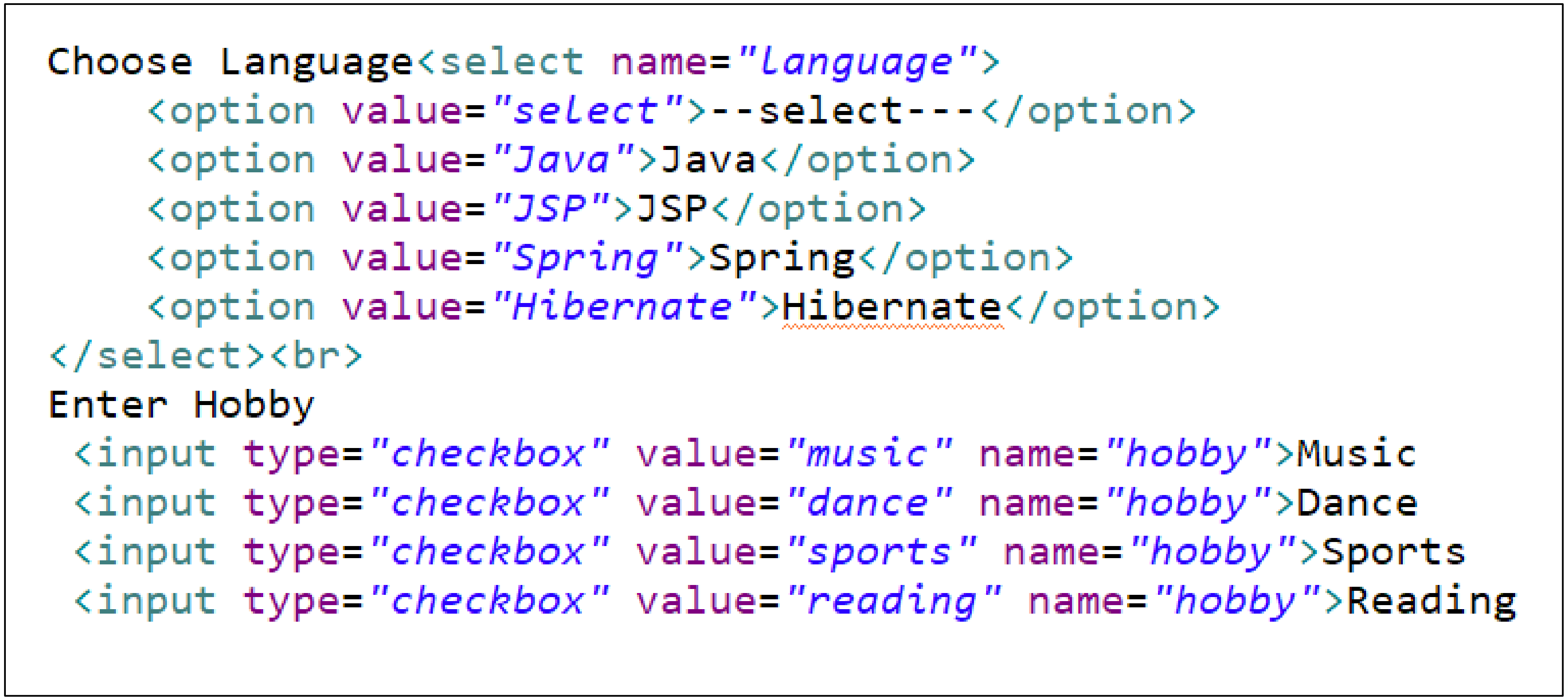
http://localhost:8080/MyProject1/login?username=mnm&city=mnmn&mobile=909

post

- Change method="post" in form tag
- Has both header and body
- all form values carried in body

http://localhost:8080/MyProject1/login

Dropdown menu and Checkbox



```
Choose Language<select name="language">
    <option value="select">--select---</option>
    <option value="Java">Java</option>
    <option value="JSP">JSP</option>
    <option value="Spring">Spring</option>
    <option value="Hibernate">Hibernate</option>
</select><br>
Enter Hobby
 <input type="checkbox" value="music" name="hobby">Music
 <input type="checkbox" value="dance" name="hobby">Dance
 <input type="checkbox" value="sports" name="hobby">Sports
 <input type="checkbox" value="reading" name="hobby">Reading
```

Get values of dropdown menu & checkbox

```
String language = request.getParameter("language");
String hobbies[] = request.getParameterValues("hobby");
```

RequestDispatcher & SendRedirect

RequestDispatcher

- Used to send the request and response to the next page(servlet/jsp)
- Set the attributes and send to the next page
- Has two methods ***forward, include***
- Is an interface
- The url shows the dummy name of the calling page (o/p is from success.jsp)

http://localhost:8080/MyProject1/login?username=pp&city=pp&mobile=909

Example

```
//retrieving
String name = request.getParameter("username");
String city = request.getParameter("city");
String mobile = request.getParameter("mobile");
long phone = Long.parseLong(mobile);
String language = request.getParameter("language");
String hobbies[] = request.getParameterValues("hobby");
//bundling
request.setAttribute("myname",name);
request.setAttribute("city",city);
request.setAttribute("mobile",phone);
request.setAttribute("language",language);
request.setAttribute("hobbies",hobbies);
//sending to view
RequestDispatcher rd = request.getRequestDispatcher("success.jsp");
rd.include(request, response);
```

forward Vs include

forward

- Forwards the request and response to the next page
- The output from the invoked(jsp) page alone will be shown in browser

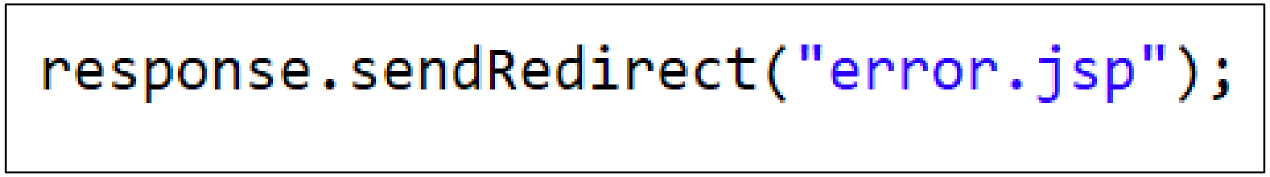
include

- Carries the output from the current page also to the next page
- The output from the current (servlet)page and the invoked page(jsp) together will be shown in browser

sendRedirect

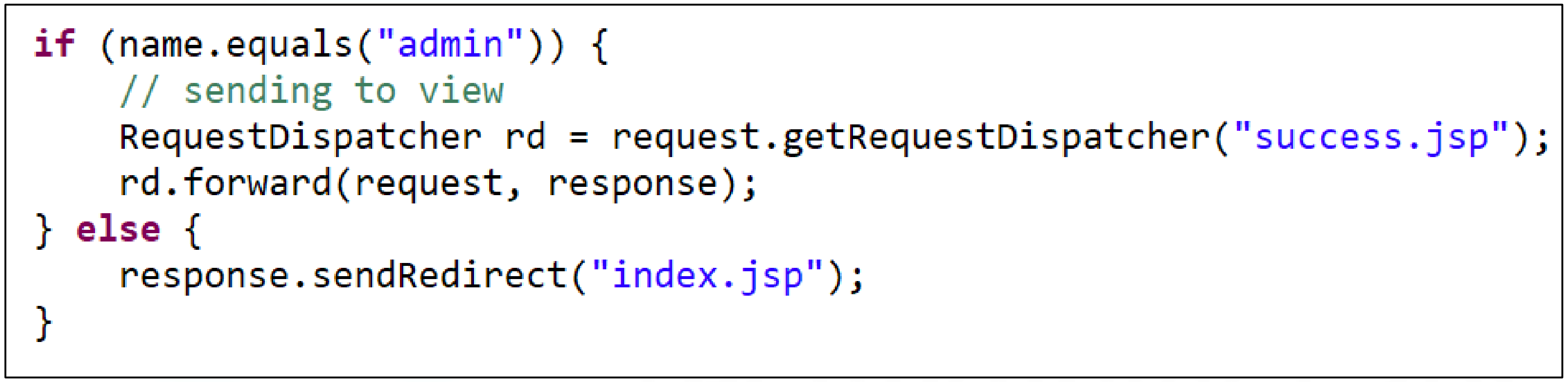
- Is a method
- Called on response object
- Will not carry the request attributes to the next page
- The url shows the final page requested(ie. **“error.jsp”**)

http://localhost:8080/MyProject1/error.jsp



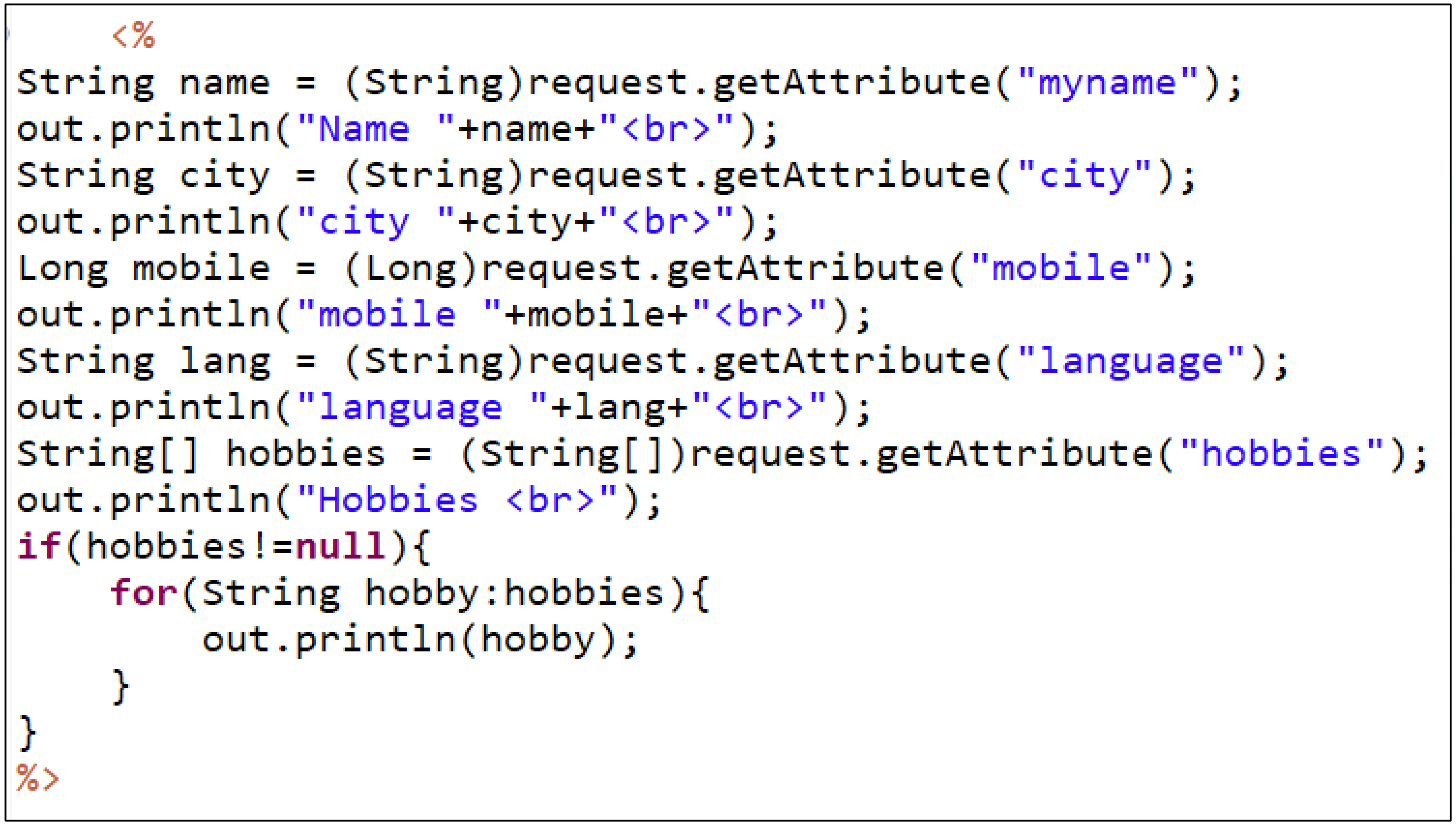
```
response.sendRedirect("error.jsp");
```

RequestDispatcher Vs sendRedirect



```java
if (name.equals("admin")) {
    // sending to view
    RequestDispatcher rd = request.getRequestDispatcher("success.jsp");
    rd.forward(request, response);
} else {
    response.sendRedirect("index.jsp");
}
```

To retrieve values in JSP



```
    <%
String name = (String)request.getAttribute("myname");
out.println("Name "+name+"<br>");
String city = (String)request.getAttribute("city");
out.println("city "+city+"<br>");
Long mobile = (Long)request.getAttribute("mobile");
out.println("mobile "+mobile+"<br>");
String lang = (String)request.getAttribute("language");
out.println("language "+lang+"<br>");
String[] hobbies = (String[])request.getAttribute("hobbies");
out.println("Hobbies <br>");
if(hobbies!=null){
    for(String hobby:hobbies){
        out.println(hobby);
    }
}
%>
```

ServletContext and ServletConfig

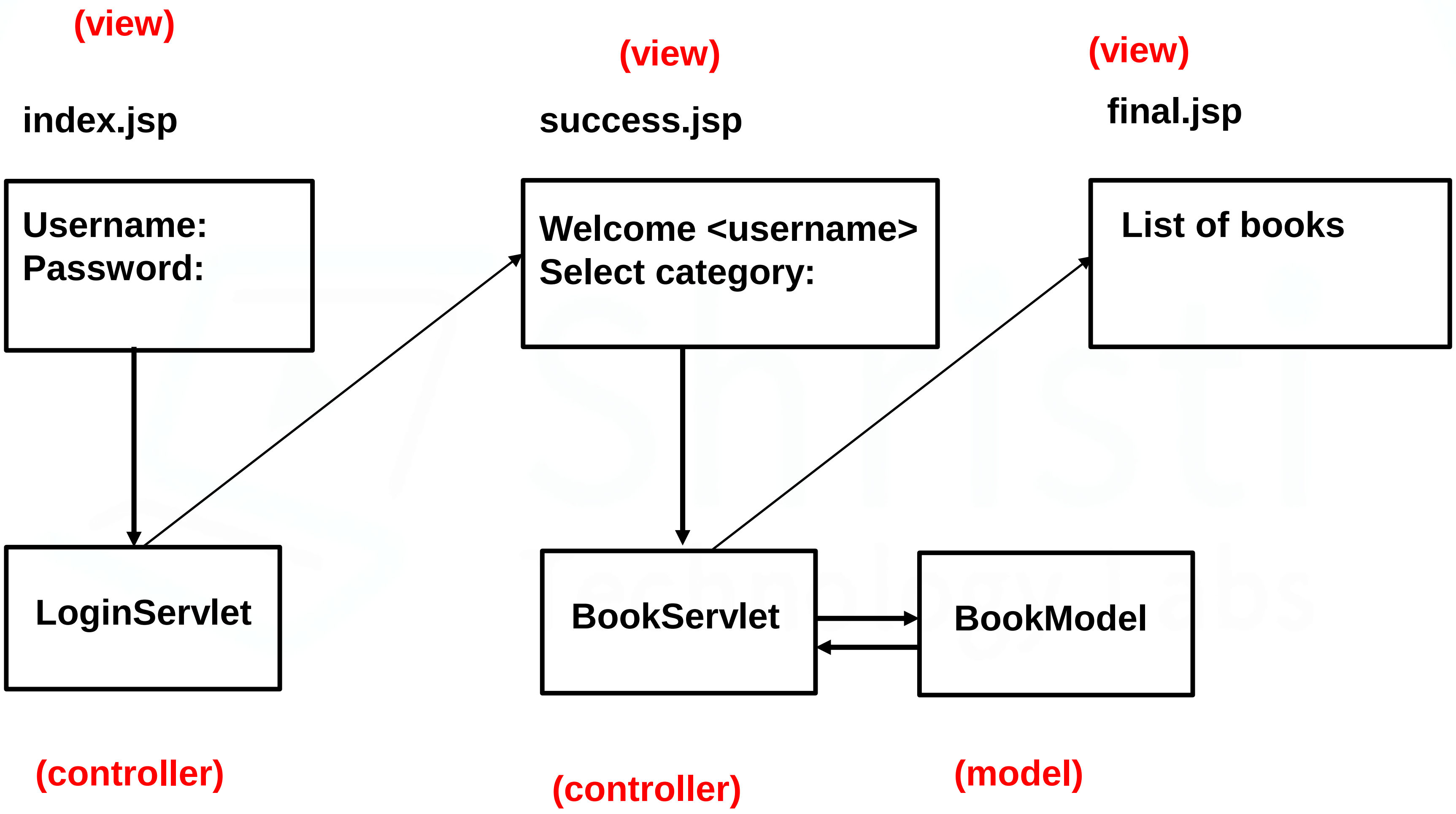
ServletContext Interface

- Is implemented by Servlet container in order to pass configuration information(database JNDI name)
- Is one per web application
- Interface defines following methods
  - getServletContext()
  - getInitParameter ()
  - getInitParameterNames()
  - getServletName()

ServletConfig Interface

- Is implemented by Servlet container in order to pass configuration information for individual servlets
- Is one per servlet
- Interface defines following methods
  - getServletConfig()
  - getInitParameter ()
  - getInitParameterNames()
  - getServletName()

Project



Project using MVC

Session Tracking

Session Tracking

- Http is a stateless protocol
- To save state information, so that information can be collected from several interactions between a browser and a server across pages
- Session Tracking with
  - HttpSession
  - Cookies
  - HiddenForms

HttpSession Interface

To get/ create session

HttpSession session = request.getSession();

-- returns an existing session if found or else creates a new session

HttpSession session = request.getSession(false);

-- returns an existing session if found or returns null

Methods of HttpSession Interface

MaxInactiveInterval

- session.getMaxInactiveInterval()

MaxInactiveInterval

- session.setMaxInactiveInterval(2); // 2 sec default 30 minutes

Session Id

- session.getId()

Creation time

- session.getCreationTime()

To Set Attribute

- session.setAttribute(“myname”,name);

Methods of HttpSession Interface

To Check for a new session

- session.isNew();

To invalidate a session(logout)

- session.invalidate();

<u>To set Session timeout in web.xml</u>

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

- **This is in minutes**

Cookie Class

- Is a small text file and stored in the clients machine
- Contains state information
- Call ***addCookie() method on HttpServletResponse*** *object*
- Use ***getCookies() method of the HttpServletRequest*** to read any cookies that are included in the HTTP get request

Example

To create cookie

```
Cookie cookie = new Cookie(“cook1”,”JSP Welcomes”);
response.addCookie(cookie);
```

To retrieve the cookie

```
Cookie cookarray[] = request.getCookies();
for(Cookie cook:cookarray){
    out.print("Name "+cook.getName()+" "+" Value "+cook.getValue());
    out.print("<br> MaxAge"+cook.getMaxAge());
    }
```

Hidden Forms

```
<input type = “hidden”
            name=“bookId”
            value=“<% = session.getBookId() %>”
```

This field is used to carry the session id from this jsp page to other pages.

To use in Servlet

```
String bookId = request.getParameter(“bookId”);
```

Summary

- What is a Servlet
- Servlet lifecycle
- Servlet API
- Structure of web Application
- Request and Response Model
- RequestDispatcher and sendRedirect
- Difference between forward and include
- Retreive values in JSP
- ServletContext and ServletConfig
- Project
- Session Tracking

Thank You