```
++++++++++++++++++++++++++++
Request Dispatching Mechanism
++++++++++++++++++++++++++++
 1. using include
 2. using forward

forward
 => To transfer the request to one more component then we need to use forward
approach.

Syntax
  RequestDispatcher rd = request.getRequestDispatcher("resourceinformation");
      rd.foward(request,response);

Scenario
 1. Adding the data to request object by the first resource and checking whether
the second resource is capable of
    reading it or not.
Ans. Yes possible, but along with the request object data send the user, container
will also few more details as shown below.

Forward Request Attribute
javax.servlet.forward.request_uri /RequestDispatching-04/first
javax.servlet.forward.context_path /RequestDispatching-04
javax.servlet.forward.servlet_path /first
javax.servlet.forward.mapping
org.apache.catalina.core.ApplicationMapping$MappingImpl@71982d6a
nitin java


FirstServlet.java
+++++++++++++++++
package in.pwskills.nitin.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/first")
public class FirstServlet extends HttpServlet {

      // Code used by JVM during De-Serialization
      private static final long serialVersionUID = 1L;

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

            // Adding the data to request object
            request.setAttribute("nitin", "java");

            //Transferring the control to second component(/second)
            ServletContext context = getServletContext();
            RequestDispatcher rd = context.getRequestDispatcher("/second");
            rd.forward(request, response);
      }
}

SecondServlet.java
++++++++++++++++++
package in.pwskills.nitin.controller;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/second")
public class SecondServlet extends HttpServlet {
      private static final long serialVersionUID = 1L;

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

            //Getting the Writer object to write the response
            PrintWriter out = response.getWriter();

            out.println("<h1>Forward Request Attribute</h1>");

            //Accessing the attributes
            Enumeration<String> names = request.getAttributeNames();

            //Retrieving all the attributes
            while (names.hasMoreElements()) {
                  String name = (String) names.nextElement();

                  Object value = request.getAttribute(name);
                  out.println(name + " " + value + "<br/>");
            }
      }
}

+++++++++++++
using include
+++++++++++++
 => Incase of include approach the container will not remove the response added by
first component rather the container will add
    the response from the first resource and it will forward it to second resource.
 => The response will be from
      TotalResponse = FirstResource + SecondResource

FirstServlet.java
+++++++++++++++++
package in.pwskills.nitin.controller;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/first")
public class FirstServlet extends HttpServlet {
      private static final long serialVersionUID = 1L;

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

            // Get the writer object to write the response
            PrintWriter out = response.getWriter();

            //Writing the response
            out.println("<h1>Hello this is FirstServlet</h1>");

            //Forwarding the request to second component(/second)
            RequestDispatcher rd = request.getRequestDispatcher("./second");
            rd.include(request, response);

            //Writing the response
            out.println("<h1>Hi this is FirstServlet Again...</h1>");

            // close the writer object
            out.close();

      }

}
```

SecondServlet.java
++++++++++++++++++

```
package in.pwskills.nitin.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/second")
public class SecondServlet extends HttpServlet {
      private static final long serialVersionUID = 1L;


      public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            PrintWriter out = response.getWriter();
            out.println("<h1>This is Second Servlet</h1>");
      }

}
```

```
Output
http://localhost:9999/RequestDispatching-05/first

Hello this is FirstServlet
This is Second Servlet
Hi this is FirstServlet Again...
```

++++++++++++++++++
Session Management
++++++++++++++++++

```
 => Client and Server can communicate with some common language, which is nothing
but HTTP.
 => The basic limitation of HTTP is it is "statless",meaning it can't remember the
client information for future purpose across
    multiple requests.Every request the server would treat it as new request.
 => Hence some mechanims is required at the server side to remember the client
information across multiple request.
 => This mechanism is nothing but "session management" mechanism.
 => Session Tracking can be implemented in 3 ways
      a. Session API(discussed)
      b. Cookie
      c. URLRe-Writing(developer created approach)
```

login.html
++++++++++

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Page</title>
</head>
<body>
      <form action='./test1'>
            <h1>Enter Books information</h1>
            <table>
                  <tr>
                        <th>NAME</th>
                        <td>
                              <input type='text' name='name'/>
                        </td>
                  </tr>
                  <tr>
                        <th>VALUE</th>
                        <td>
                              <input type='text' name='value'/>
                        </td>
                  </tr>
                  <tr>
                        <th></th>
                        <td>
                              <input type='submit' value='ADD TO CART'/>
                        </td>
                  </tr>
            </table>
      </form>
<a href='./test2'>SHOW CART</a>
</body>
</html>
```

SessionServlet1.java
++++++++++++++++++++

```
package in.pwskills.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/test1")
public class SessionServlet1 extends HttpServlet {
      private static final long serialVersionUID = 1L;

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

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            // Getting the session object to track the request information of
client
            HttpSession session = request.getSession();

            if (session.isNew()) {
                  out.println("<h2>New Session created with the id :: " +
session.getId() + "</h2>");
            } else {
                  out.println("<h2>Existing session only with the session id :: " +
session.getId() + "</h2>");
            }

            //Retrieving the user-information from request object
            String name = request.getParameter("name");
            String value = request.getParameter("value");

            //Keeping the user-information in session object
            session.setAttribute(name, value);

            //Specify the max active time
            session.setMaxInactiveInterval(60);

            //Sending the response to the end-user
            RequestDispatcher rd = request.getRequestDispatcher("login.html");
            rd.include(request, response);

            out.close();
      }
```

```
}


SessionServlet2.java
++++++++++++++++++++
package in.pwskills.controller;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/test2")
public class SessionServlet2 extends HttpServlet {
      private static final long serialVersionUID = 1L;

      public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            PrintWriter out = response.getWriter();

            // Getting the old session object for the same user
            HttpSession session = request.getSession(false);
            if (session == null) {
                  out.println("<h1>NO session information available w.r.t the
user</h1>");
            } else {
                  out.println("<table
border='2'><tr><th>AttributeName</th><th>AttributeValue</th></tr>");

                  // Retrieve the information from session object and process the
data
                  Enumeration<String> names = session.getAttributeNames();

                  // Processing the information using while loop from enumeration
object
                  while (names.hasMoreElements()) {
                        String name = (String) names.nextElement();
                        Object value = session.getAttribute(name);
                        out.println("<tr><td>" + name + "</td><td>" + value +
"</td></tr>");
                  }
                  out.println("</table>");

                  // Extra information is also retrieved
                  long creationTime = session.getCreationTime();
                  long lastAccessedTime = session.getLastAccessedTime();
                  int maxInactiveInterval = session.getMaxInactiveInterval();
                  out.println("<h1>CreationTime         is :: " + new
Date(creationTime) + "</h1>");
                  out.println("<h1>LastAccessedTime     is :: " + new
Date(lastAccessedTime) + "</h1>");
                  out.println("<h1>MaxInactiveInterval  is :: " + new
Date(maxInactiveInterval) + "</h1>");
            }

            out.close();
      }
}
```

run the program to see the output with the session object

DisAdvantage associated with HTML
 a. HTML is static, it can't be made dynamic like java language to retrieve the data from java object
 b. To resolve this problem SUNMS team had introduced a language which is like HTML,but it is made dynamic that technology only
    is "Java Server Pages(JSP)".

index.jsp
=========

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Search Page</title>
</head>
<body>
      <h1 style='color: red; text-align: center'>EMPLOYEE DATA PAGE</h1>
      <form method='GET' action='./retrieve'>
            <table>
                  <tr>
                        <th>USERID</th>
                        <td><input type='text' name='userid' /></td>
                  </tr>
                  <tr>
                        <th></th>
                        <td><input type='submit' value='getRecord'></td>
                  </tr>
            </table>
      </form>
</body>
</html>
```

RetrievalApp.java
+++++++++++++++++

```
package in.pwskills.controller;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import in.pwskills.entity.Employee;
import in.pwskills.utility.JdbcUtil;

@WebServlet("/retrieve")
public class RetrievalApp extends HttpServlet {
      private static final long serialVersionUID = 1L;
      private static final String SQL_SELECT_QUERY = "select
eid,ename,eage,eaddress from employee where eid = ? ";

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

            // Collect input supplied by the user
            String userId = request.getParameter("userid");

            try {
                  Connection connection = JdbcUtil.getDbConnection();

                  PreparedStatement pstmt =
connection.prepareStatement(SQL_SELECT_QUERY);
                  pstmt.setInt(1, Integer.parseInt(userId));
                  ResultSet resultSet = pstmt.executeQuery();

                  RequestDispatcher rd = null;

                  if (resultSet.next()) {

                        Employee employee = new Employee();
                        employee.setEid(resultSet.getInt(1));
                        employee.setEname(resultSet.getString(2));
                        employee.setEage(resultSet.getInt(3));
                        employee.setEaddress(resultSet.getString(4));

                        //forwarding to jsp page
                        rd = request.getRequestDispatcher("success.jsp");
                        request.setAttribute("employee", employee);
                        rd.forward(request, response);


                  } else {
                        //forwarding to jsp page
                        rd = request.getRequestDispatcher("failure.jsp");
                        request.setAttribute("userId", userId);
                        rd.forward(request, response);
                  }

            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }
}

success.jsp
===========
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<%@ page import="in.pwskills.entity.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>OUTPUT PAGE</title>
</head>
<body>
      <h1 style='color: red; text-align: center'>EMPLOYEE DATA</h1>
      <% Employee emp = (Employee)request.getAttribute("employee");%>
      <table border='1' align="center">
            <tr>
                  <th>EID</th>
                  <th>ENAME</th>
                  <th>EAGE</th>
                  <th>EADDRESS</th>
            </tr>
            <tr>
                  <td><%=emp.getEid()%></td>
                  <td><%=emp.getEname()%></td>
                  <td><%=emp.getEage()%></td>
                  <td><%=emp.getEaddress()%></td>
            </tr>
      </table>
</body>
</html>

failure.jsp
===========
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>FAILURE PAGE</title>
</head>
<body>
      <h1 style='color:red; text-align: center'>
            Record not available for the given id ::
                  <%=request.getAttribute("userId") %>
      </h1>
</body>
</html>
```