```
        public String getUserName()
        {
                return userName;
        }
        public void setUserName(String userName)
        {
                this.userName = userName;
        }
        public int getUserType()
        {
                return userType;
        }
        public void setUserType (int userType)
        {
                this.age = userType;
        }
}
```

Example: JSP Page to send User Details to UserDetails.JSP <html>

```
<head><title> User Form </title></head>
<body>
<form action="UserDetails.jsp" method="post">
        User Name : <input type="text" name="userName"><br>
        User Type : <input type="text" name="userType"><br>
                        <input type="submit" value="Register">
</form>
</body>
</html>
```

Example: JSP page to write and read Bean Properties (File UserDetails.JSP)

```
<html>
<head><title> Print User Details
</title></head> <body>
        <jsp:useBean id="userInfo" class="UserBean" />
        <jsp:setProperty property="*" name="userInfo"/>
        <h1> Received User Detail as Below</h1>
        User Name:<jsp:getProperty property="userName"
        name="userInfo"/> <br>
        User Type: <jsp:getProperty property="userType" name="userInfo"/>
</body>
</html>
```

Http Session

The servlet container uses HttpSession interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user. A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using Cookies or URL Rewriting or Hidden Form Field or HttpSession Interface.

HttpSession interface allows servlets to View and manipulate information about a session, such as the session identifier, creation time, and last accessed time Bind objects to sessions, allowing user information to persist across multiple user connections.

```
HttpSession session = request.getSession(true);
```

true : Use existing session if exist or create one new session
false : Use existing session if exist or return null

```
HttpSession session = request.getSession();
                              //Same as request.getSession(true);
```

Setting inactive time and values in session

```
session.setAttribute("user", username);
session.setMaxInactiveInterval(30);
```

Destroy Session in JSP and Servlet

1- There are some way to prevent session in JSP. It will not create session for this page.

```
<%@ page session="false"%>
```

2- Use session object to destroy the session
```
sesssion.invalidate();
```

3- Set Time till session is valid. If no session is active no session is created and it expires immediately

```
HttpSession session = request.getSession (false);

if (session != null)
        session.setMaxInactiveInterval(1);
```

Example: HTML Page Login.html Which Calls Servlet That Creates Session.

```
<html><head><title>Insert title
here</title></head> <body>
        <h1> Http Session HTML Page </h1>
        <form action="LoginController" method="post">
                User name : <input type="text" name="userName"><br>
                Password : <input type="password" name="password"><br>
                <input type="submit" value="Login">
        </form>
</body>
</html>
```

Example: Servlet Which Creates Session.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginController extends HttpServlet
{
        protected void doPost(HttpServletRequest request,
                                HttpServletResponse response)
                                        throws ServletException, IOException
        {
                response.setContentType("text/html");

                PrintWriter out = response.getWriter();
                String uname = request.getParameter("userName");
                String pwd = request.getParameter("password");

                if (un.equals("GP Lohaghat"))

                {
                        out.print("Welcome, " + uname);
                        HttpSession session = request.getSession(true);
                        session.setAttribute("userName", uname);
                session.setMaxInactiveInterval(60);
                response.sendRedirect("Home.jsp");
            }
            else
            {
                RequestDispatcher rd = null;
                rd = request.getRequestDispatcher ("Login.html");
                out.println("<font color=red>Wrong
                Credentials</font>"); rd.include(request, response);
            }
        }
}
```

Example: Home.JSP Which Reads Session Attributes.

```
<html>
<head><title>Accessing Session Attribute</title></head>
<body>

        <%
            String name = null;
            if (session != null)
            {
                if (session.getAttribute("user") != null) {
                    name = (String) session.getAttribute("userName");
                    out.print("Hello, " + name + "Welcome");
                }
                else
                    response.sendRedirect("login.html");
            }
        %>

        <form action="LogoutController" method = "post">
            <input type="submit"
        value="Logout"> </form>
</body>
</html>
```

Example: Servlet Which Destroy Session.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LogoutController extends HttpServlet
{
	protected void doPost(HttpServletRequest request,
					HttpServletResponse response)
					Throws ServletException, IOException
	{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		out.println("Thank You. Session was destroyed successfully!!");
		HttpSession session = request.getSession(false);

		synchronized (session)
		{
			session.setAttribute("user", null);
			session.removeAttribute("user");
			session.getMaxInactiveInterval();
			session.invalidate();
		}
	}
}
```

MVC (Model View Controller)

MVC stands for Model View and Controller. It is a design pattern that separates the business logic and data access layers from the presentation layer.

Controller: Controller is a Servlet that acts as an interface between View and Model. Controller intercepts all the incoming requests.

Model: Model is a Java Bean which represents the state of the application i.e. data. It can also have business logic.

View: View is a JSP page which represents the presentation or User Interface.

Steps required to Implement MVC

1. Define beans to represent the data of application.

2. Use a Servlet to handle requests from clients.

3. Populate the beans by placing result obtained from accessing business logic and data access code in the beans.

4. Store the bean either in the request, session or Servlet context (application).