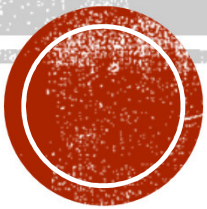
CHAPTER FOUR

Advanced Enterprise Software Development

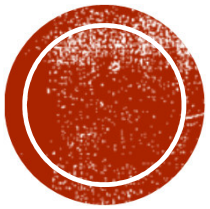


CHAPTER OUTLINE

- MVC Design Pattern
  - Model: Java Bean
  - View: Java Server Pages (JSP)
  - Control: Java Servlet
- Software Framework
- Microservices
- Restful Web Service

WEB PROGRAMMING IN JAVA

- Fundamental Java Web Technologies
  - Java Servlet
  - Java Server Pages (JSP)
  - Enterprise JavaBean (EJB)
- Advanced Technologies
  - Software Framework (Portlet, struts, spring, JSF, etc.)
    - Require some advanced object oriented programming skills
  - Software Platform (Liferay, Jakarta Pluto, EXO Platform, etc.)
    - Require little programming skills
    - Require system administrative skills
- Technologies Standardization: Java Enterprise Edition (Java EE)
  - Provides many web technology standards for development.
  - The Java Community Process (JCP) defines the standard detail in Java Specification Requests (JSR) http://jcp.org/en/jsr/all (List of all JSR).



3.1 JAVA SERVER PAGES

WHAT IS JSP?

- A Java Server Pages (JSP) is a technology that let's software developers create dynamically generated web pages based on HTML.
- A JSP is similar to PHP in that it is a markup that is a combination of HTML and Java where as PHP was a combination of HTML and PHP. It is also similar to ASP from Microsoft that was a combination of C# and HTML.
- A JSP is compiled into a Servlet!

JAVA SERVLET EXAMPLE

```
01. import java.io.*;
02. import javax.servlet.*;
03.
04. public class SecondServlet extends HttpServlet {
05.   public void doGet(HttpServletRequest req, HttpServletResponse resp)
06.           throws IOException, ServletException {
07.     doPost(req, resp);
08.   }
09.
10.   public void doPost(HttpServletRequest req, HttpServletResponse resp)
11.           throws IOException, ServletException {
12.     PrintWriter out = resp.getWriter();
13.     out.println("<html>");
14.     out.println("<head>");
15.     out.println("<title>First Servlet</title>");
16.     out.println("</head>");
17.     out.println("<body>");
```

JAVA SERVLET EXAMPLE (CONT.)

```
18.     out.println("<h3>Monthly Report</h3>");
19.     out.println("<table width=90% align=center border=1");
20.     out.println("  <tr>");
21.     out.println("    <td>Transaction Date</td>");
22.     out.println("    <td>Transaction Number</td>");
23.     out.println("    <td>Account Number</td>");
24.     out.println("    <td>Currency</td>");
25.     out.println("    <td>Debit Amount</td>");
26.     out.println("    <td>Credit Amount</td>");
27.     out.println("    <td>Balance</td>");
28.     out.println("  </tr>");
29.     out.println("  <tr>");
30.     out.println("    <td>2012-07-01</td>");
31.     out.println("    <td>163257937</td>");
32.     out.println("    <td>01-10-21-123456</td>");
33.     out.println("    <td>MOP</td>");
34.     out.println("    <td>0.00</td>");
```

JAVA SERVLET EXAMPLE (CONT.)

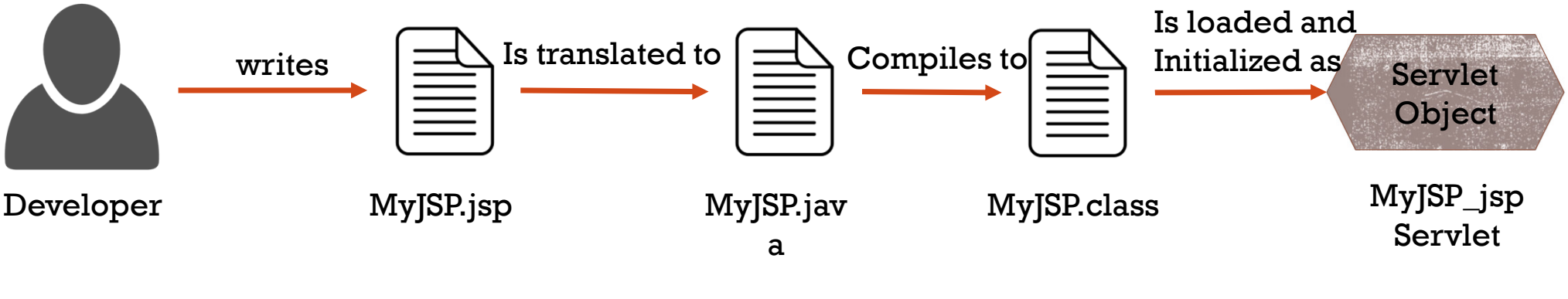
```
35.         out.println("    <td>100.00</td>");
36.         out.println("    <td>5,000.00</td>");
37.         out.println("  </tr>");
38.         out.println("  <tr>");
39.         out.println("    <td>2012-07-02</td>");
40.         out.println("    <td>163257948</td>");
41.         out.println("    <td>01-10-21-123456</td>");
42.         out.println("    <td>MOP</td>");
43.         out.println("    <td>0.00</td>");
44.         out.println("    <td>200.00</td>");
45.         out.println("    <td>4,800.00</td>");
46.         out.println("  </tr>");
47.         out.println("</table>");
48.         out.println("</body>");
49.         out.println("</html>");
59.     }
51. }
```

JAVA SERVLET PROBLEM

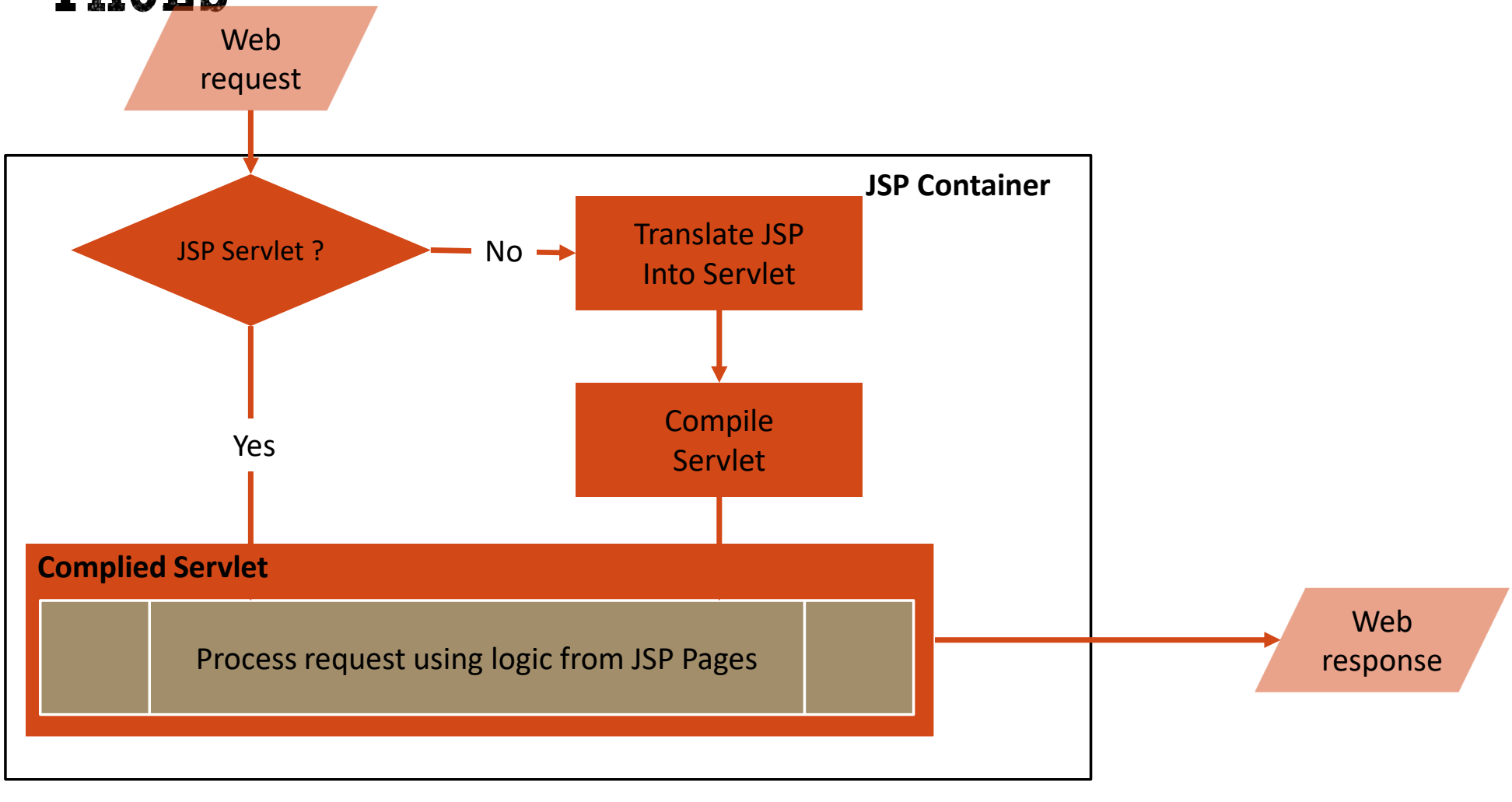
- Using Java Servlet to show lengthy HTML content and to show special characters are awkward.
- Since HTML codes are embedded inside Java Servlets, it is very hard for web designers to change the look-and-view of the web pages.
- Mixing HTML with Java code is simply a mess (spaghetti code).
- So, how to simplify on coding a large web page in Java Servlet?

INTRODUCTION TO JAVA SERVER PAGES (JSP)

- Writing plenty of HTML code inside a Java Servlet is a nightmare
- JSP on the other hand is writing Java code inside the HTML files
- JSP helps to present the HTML response objects easily
- Changing a Java Servlet may require restarting server to take effect, but JSP is complied in run-time. So, we can always get the latest version of our web applications
- JSP is the extension of Java Servlet, and container will translate JSP into servlet (Java file), so you don't need to compile JSP file



PROCESS OF TRANSLATING AND RUNNING JSP PAGES



JSP EXAMPLE

- Developing JSP program is liked writing normal HTML file.
- If you want to inject some Java codes, simply surround the Java codes with <% ..%> tags.
- SystemTime.jsp

```
01. <%@page import="java.util.Calendar;" %>
02. <html>
03. <head>
04. <title>JSP Example</title>
05. </head>
06. <body>
07. Time is: <% out.println(Calendar.getInstance().getTime()); %>
08. </body>
09. </html>
```

CONVERT JSP TO JAVA CLASS

- Web container converts JSP to a Java class or a Java Servlet.

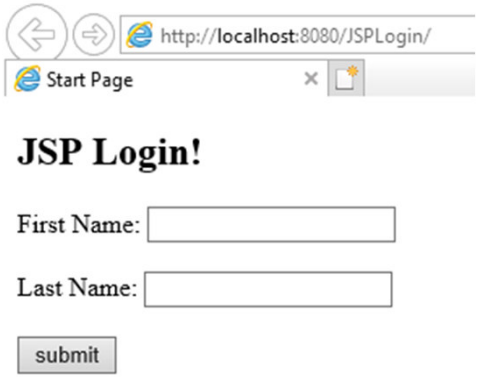
```java
/*
 * Generated by the Jasper component of Apache Tomcat
 * Version: Apache Tomcat/8.5.58
 * Generated at: 2021-09-05 07:52:29 UTC
 * Note: The last modified time of this file was set to
 *       the last modified time of the source file after
 *       generation to assist with modification tracking.
 */
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import java.util.Calendar;

public final class firstJSP_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
                 org.apache.jasper.runtime.JspSourceImports {

  private static final javax.servlet.jsp.JspFactory _jspxFactory =
          javax.servlet.jsp.JspFactory.getDefaultFactory();

  private static java.util.Map<java.lang.String,java.lang.Long> _jspx_dependants;

  private static final java.util.Set<java.lang.String> _jspx_imports_packages;
```

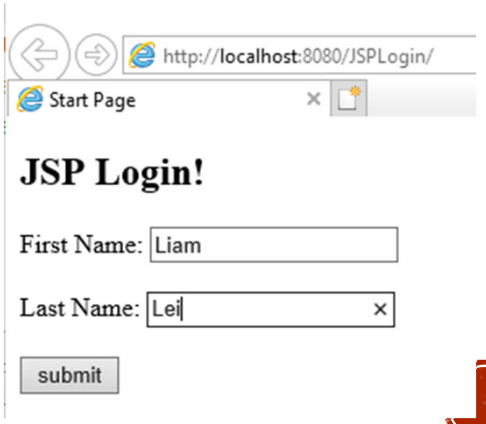
.....

```java
      out = pageContext.getOut();
      _jspx_out = out;

      out.write("\n");
      out.write("\n");
      out.write("\n");
      out.write("<!DOCTYPE html>\n");
      out.write("<html>\n");
      out.write("<head>\n");
      out.write("    <meta charset=\"utf-8\">\n");
      out.write("<title>Insert title here</title>\n");
      out.write("</head>\n");
      out.write("    <body>\n");
      out.write("        <h3>This is a JSP </h3>\n");
      out.write("        Time is: ");
out.println(Calendar.getInstance().getTime());
      out.write("\n");
      out.write("    </body>\n");
      out.write("</html>\n");
      out.write("\n");
    } catch (java.lang.Throwable t) {
      if (!(t instanceof javax.servlet.jsp.SkipPageException)){
```

JSP LOGIN EXAMPLE (INDEX.HTML)

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>Start Page</title>
5           <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6       </head>
7       <body>
8           <h2>JSP Login!</h2>
9           <form action="LogJSP" method="get">
10
11          First Name: <input type="text" name="FirstName"/> <br><br>
12          Last Name: <input type="text" name="LastName" /> <br><br>
13          <input type="submit" value="submit">
14          </form>
15      </body>
16  </html>
```



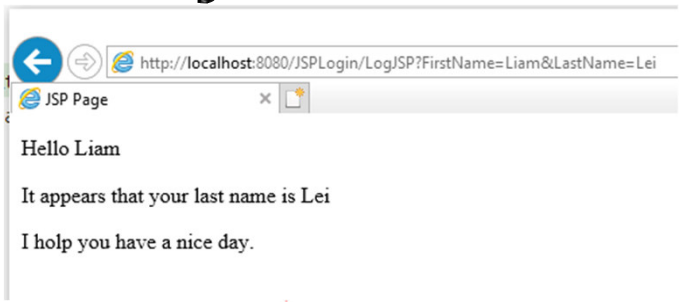


JSP LOGIN EXAMPLE （WEB.XML）

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5               http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
6       <servlet>
7           <servlet-name>HandelForm</servlet-name>
8           <servlet-class>edu.ipm.jsplogin.HandelFormInput</servlet-class>
9       </servlet>
10      <servlet-mapping>
11          <servlet-name>HandelForm</servlet-name>
12          <url-pattern>/LogJSP</url-pattern>
13      </servlet-mapping>
14  </web-app>
15
```

JSP LOGIN EXAMPLE (RESPONSE.JSP)

```
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <!DOCTYPE html>
9   <html>
10      <head>
11          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12          <title>JSP Page</title>
13      </head>
14      <body>
15          <p>Hello <%= request.getAttribute("firstName") %></p>
16          <p>It appears that your last name is <%= request.getAttribute("lastName") %></p>
17          <p>I holp you have a nice day.</p>
18
19
20      </body>
21  </html>
```

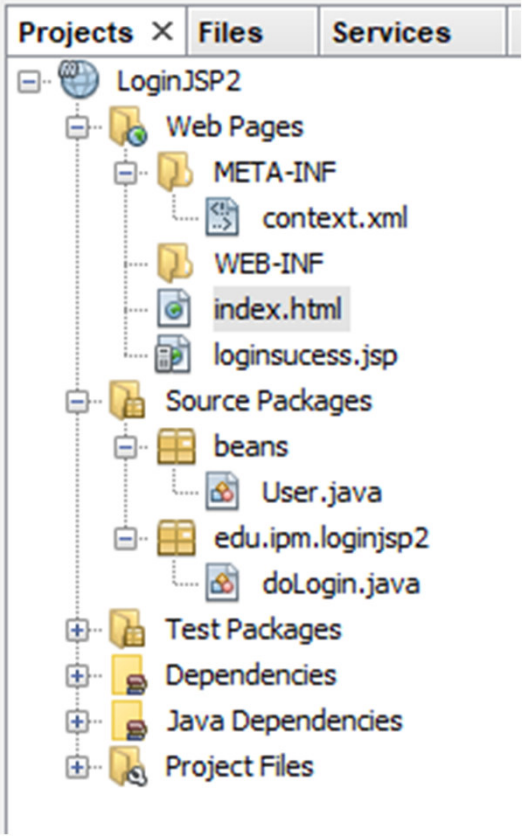


JSP LOGIN EXAMPLE (DOPOST METHOD)

```
18
19      @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
21              throws ServletException, IOException {
22          String firstName = request.getParameter("FirstName");
            String lastName = request.getParameter("LastName");
24
25          request.setAttribute("firstName", firstName);
26          request.setAttribute("lastName", lastName);
27          request.getRequestDispatcher("ResponsePage.jsp").forward(request, response);
28      }
```

JSP + SERVLET LOGIN EXAMPLE

- Project Structure
  - `index.html` (the web application home page).
  - `doLogin.java` (servlet for handling request and response)
  - `User.java` (the JavaBean for modeling the user data)
  - `loginsucess.jsp` (a jsp document for handling how to represent the response to the client)



JSP + SERVLET LOGIN EXAMPLE

index.html

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>JSP Login 2</title>
5           <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6       </head>
7       <body>
8           <h1>JSP + Servlet + HTML</h1>
9           <form action="doLogin" method="post">
10              User Name: <input type="text" name="userName"/> <br><br>
11              Password: <input type="password" name="userPass"/><br>
12              <input type="submit" value="Login">
13          </form>
14      </body>
15  </html>
```

JSP + SERVLET LOGIN EXAMPLE

doLogin.java: doGet()

```
1
2   package edu.ipm.loginjsp2;
3
4   import beans.User;
5   import java.io.IOException;
6   import javax.servlet.ServletException;
7   import javax.servlet.annotation.WebServlet;
8   import javax.servlet.http.HttpServlet;
9   import javax.servlet.http.HttpServletRequest;
10  import javax.servlet.http.HttpServletResponse;
11
12  @WebServlet(name="myLoginServlet",
13              urlPatterns="/doLogin")
14  public class doLogin extends HttpServlet {
15
16      @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
18              throws ServletException, IOException {
19          doPost(request, response);
20      }
```

Import dependences libraries

Mapping a Servlet to an URL

doGet() method

JSP + SERVLET LOGIN EXAMPLE

doLogin.java: doPost()

```
22      @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
24              throws ServletException, IOException {
25
26          String userName = request.getParameter("userName");
27          String userPass = request.getParameter("userPass");
28
29          User loginUser = new User();
30          loginUser.setName(userName);
31          loginUser.setPassWord(userPass);
32
33          request.getSession().setAttribute("loginUser", loginUser);
34          response.sendRedirect("loginsucess.jsp");
35      }
36
37  }
```

Get user's information from the request

Create an User object to store user's information

Send the User Bean to JSP for forward processing

JSP + SERVLET LOGIN EXAMPLE

User.java (JavaBean)

```
1
2   package beans;
3
4   public class User {
5
6       private String name;
7       private String passWord;
8
9       public User(){
10          name = "testUser";
11          passWord = "123456";
12      }
```

member variable

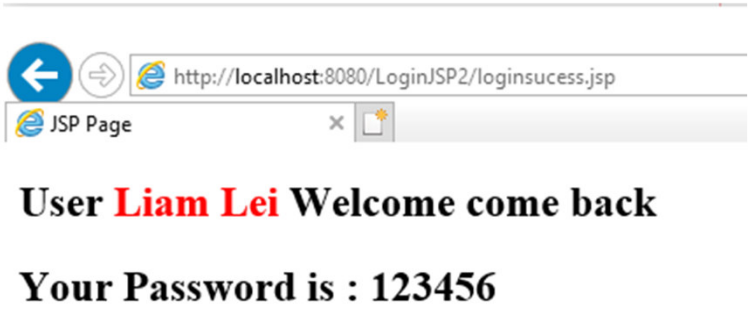
Constructor

```
13
14      public String getName() {
15          return name;
16      }
17
18      public void setName(String name) {
19          this.name = name;
20      }
21
22      public String getPassWord() {
23          return passWord;
24      }
25
26      public void setPassWord(String passWord) {
27          this.passWord = passWord;
28      }
29
30
31  }
```

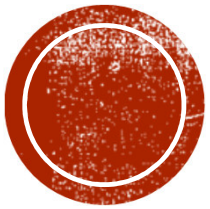
Getters and setters

JSP + SERVLET LOGIN EXAMPLE

loginsucess.jsp



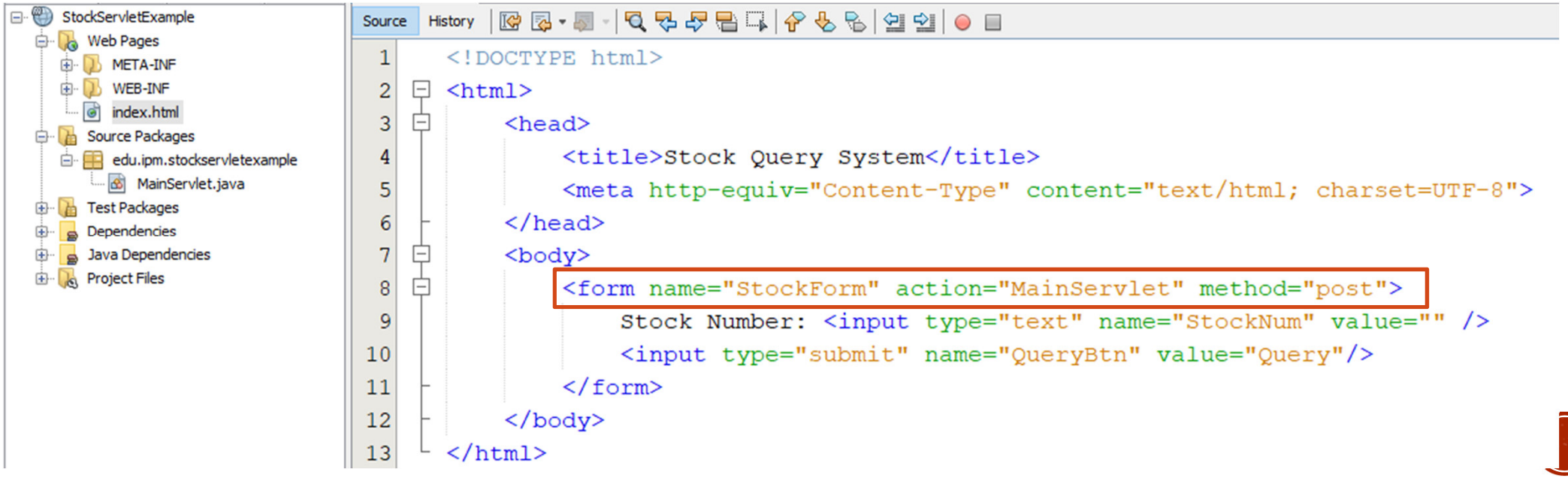

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h2>User <spand style="color:red">${loginUser.name}</spand> Welcome come back </h2>
        <h2>Your Password is : ${loginUser.passWord}</h2>
    </body>
</html>
```



MVC DESIGN PATTERN

STOCK SYSTEM EXAMPLE 1

- An HTML page submits a stock number to the server, and it replies the stock information and current price.
- HTML form submits to `MainServlet` to process the transaction.



SINGLE PROGRAM (JAVA SERVLET) TO PROCESS A TRANSACTION

- Import dependencies libraries.

```
1   package edu.ipm.stockservletexample;
2
3   import java.io.IOException;
4   import java.io.PrintWriter;
5   import java.text.DecimalFormat;
    import java.util.Hashtable;
7   import javax.servlet.ServletException;
8   import javax.servlet.http.HttpServlet;
9   import javax.servlet.http.HttpServletRequest;
10  import javax.servlet.http.HttpServletResponse;
```

SINGLE PROGRAM (JAVA SERVLET) TO PROCESS A TRANSACTION

```
12  public class MainServlet extends HttpServlet {
13
14      // Create a Hashtable to store stock's Names and number
        private static final Hashtable<String, String> stockName = new Hashtable<String, String>();
16
17      // static code block for store's information
18      static {
19          stockName.put("00001", "AAA Company");
20          stockName.put("00002", "BBB Company");
21          stockName.put("00003", "CCC Company");
22          stockName.put("00004", "DDD Company");
23          stockName.put("00005", "EEE Company");
24          stockName.put("00006", "FFF Company");
25          stockName.put("00007", "GGG Company");
26          stockName.put("00008", "HHH Company");
27          stockName.put("00009", "III Company");
28          stockName.put("00010", "JJJ Company");
29      }
30      @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
32              throws ServletException, IOException {
33          doPost(request, response);
34      }
```

Store information data

doGet( )

SINGLE PROGRAM (JAVA SERVLET) TO PROCESS A TRANSACTION

```
36      @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
38              throws ServletException, IOException {
39          // configure out put format
40          DecimalFormat df = new DecimalFormat();
41          df.applyPattern("#,###,##0.00");
42          // get the stock number from user
43          String number = request.getParameter("StockNum");
44          // query stock's name and price according stock' number
45          String name = stockName.get(number) == null ? "n/a" : stockName.get(number);
46          String price = df.format(stockName.get(number) == null ? 0.0 : Math.random() * 100);
```

Handel and processing User Input

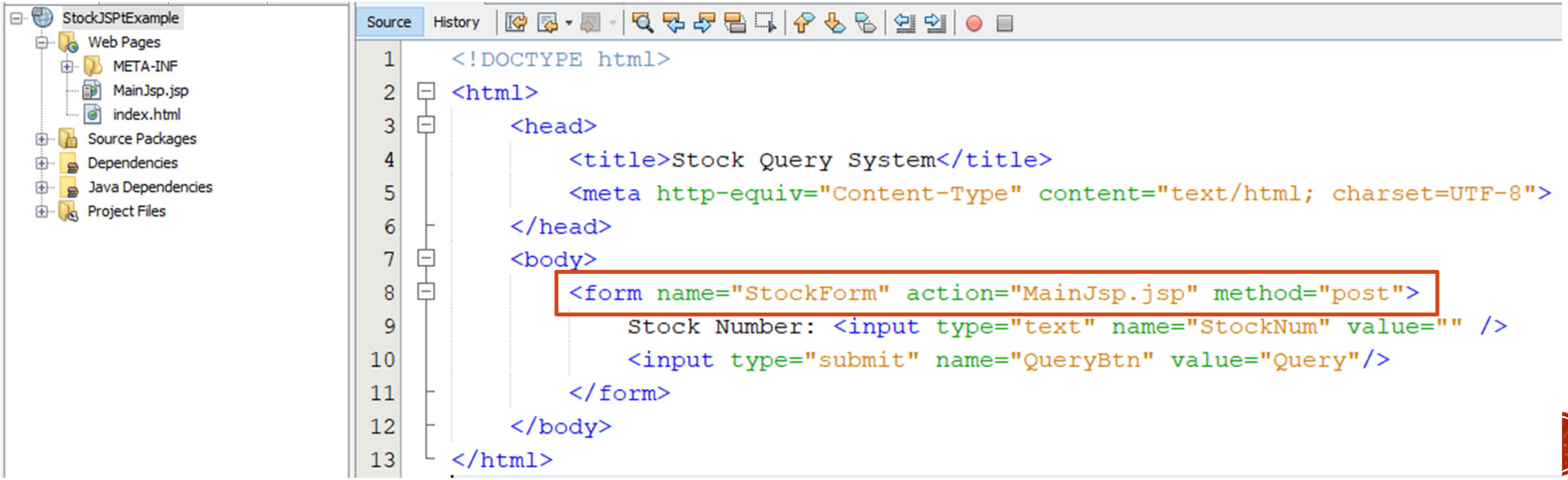
SINGLE PROGRAM (JAVA SERVLET) TO PROCESS A TRANSACTION

```
47          // generate a response html file
48          PrintWriter out = response.getWriter();
49          out.println("<html>");
50          out.println("<head>");
51          out.println("<title> Stock Query Response</title>");
52          out.println("</head>");
53          out.println("<body>");
54          out.println("<h2>The stock information you want to find is:</h2>");
55          out.println("<table border=\"1\">");
56          out.println("   <tr>");
57          out.println("       <td>Stock Number</td>");
58          out.println("       <td>Stock Name</td>");
59          out.println("       <td>Stock Price");
60          out.println("   </tr>");
61          out.println("       <td>" + number +"</td>");
62          out.println("       <td>" + name + "</td>");
63          out.println("       <td>" + price + "</td>");
64          out.println("   </tr>");
65          out.println("</body>");
66          out.println("</html>");
67      }
68
69  }
```

Display query result to user

STOCK SYSTEM EXAMPLE 2

- An HTML page submits a stock number to the server, and it replies the stock information and current price.
- HTML form submits to `MainJsp.jsp` to process the transaction.



SINGLE PROGRAM (JSP) TO PROCESS A TRANSACTION

- Import dependencies libraries.

```
<%@page import="java.text.DecimalFormat"%>
<%@page import="java.util.Hashtable"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Stock Query Response</title>
    </head>
```

SINGLE PROGRAM (JSP) TO PROCESS A TRANSACTION

```
10  <%
11  // Create a Hashtable to store stock's Names and number
        Hashtable<String, String> stockName = new Hashtable<String, String>();
13
14  // static code block for store's information
15
16      stockName.put("00001", "AAA Company");
17      stockName.put("00002", "BBB Company");
18      stockName.put("00003", "CCC Company");
19      stockName.put("00004", "DDD Company");
20      stockName.put("00005", "EEE Company");
21      stockName.put("00006", "FFF Company");
22      stockName.put("00007", "GGG Company");
23      stockName.put("00008", "HHH Company");
24      stockName.put("00009", "III Company");
25      stockName.put("00010", "JJJ Company");
26
27      // configure out put format
28      DecimalFormat df = new DecimalFormat();
29      df.applyPattern("#,###,##0.00");
30      // get the stock number from user
31      String number = request.getParameter("StockNum");
32      // query stock's name and price according stock' number
33      String name = stockName.get(number) == null ? "n/a" : stockName.get(number);
34      String price = df.format(stockName.get(number) == null ? 0.0 : Math.random() * 100);
35  %>
```

SINGLE PROGRAM (JSP) TO PROCESS A TRANSACTION

```
36      <body>
37
38          <h2>The stock information you want to find is:</h2>
39          <table border="1">
40              <tr>
41                  <td>Stock Number</td>
42                  <td>Stock Name</td>
43                  <td>Stock Price</td>
44              </tr>
45              <tr>
46                  <td><%=number%></td>
47                  <td><%=name%></td>
48                  <td><%=price%></td>
49              </tr>
50          </table>
51      </body>
52  </html>
```

PROBLEM & SOLUTION

- Problems
  - Presentation, Business Logic, Control flow, and Data layers are bounced together. This all-in-one program is lengthy and is not easy to maintain.
  - Logic and presentation layer cannot be reused. It is not flexible and extensible for future requirements.
  - Any modification may affect the correctness of processing the whole transaction.
- Solutions
  - We should separate the program according to their functional purposes into different modules.
  - We should adopt the MVC design pattern.
  - We can use Java Servlet, JSP, and JavaBean web technologies to architect this transaction.

INTRODUCTION TO THE MVC DESIGN PATTERN

The MVC architecture pattern turns complex application development into a much more manageable process. It allows several developers to simultaneously work on the application.

- The MVC design pattern is used to build the Presentation (Web) Layer.
- The MVC design pattern helps to enforce separation of concerns to help you avoid mixing presentation logic, business logic, and data access logic together.
- What is MVC?
  - MVC stands for model-view-controller. Here's what each of those components mean:
    - **Model**: manages the behavior and data from the application layers
    - **View**: manages the display of data (The frontend or graphical user interface)
    - **Controller**: handles page events and navigation between pages (calls Business Services to get the (Object) Model and navigates to the next View)

RECAP ON JAVA SERVLET, JSP, JAVABEAN

- Java Servlet
  - Receives requests from client and processes the business logic.
  - Chooses the JavaBean model to hold the result data.
  - Controls the transaction workflow: calls sub functions and chooses the resulting page.
  - Should not have page content in most situation
- Java Server Pages
  - Responses to client and processes the data presentation.
  - Retrieves the JavaBean data from request or session object.
  - Formats the data objects and present the data in HTML back to client.
  - Should not have business logic in most situation

RECAP ON JAVA SERVLET, JSP, JAVABEAN

- JavaBean
  - Keeps the transactional data temporary.
  - Models the data structure such as mapping a table or a transaction
  - Restricts the data range and format in setter methods
  - Should not have page content and business logic in all situations.

RECAP ON JAVABEAN

- Server side program designed for reuse and to hold data
- A package of objects: to encapsulate many objects into a single object (the bean)
- It is serializable Java object to persist data through servers
- Modern software frameworks use JavaBean to transform data to different style formats (database table, XML, JSON, etc.)
- Enterprise JavaBeans (EJB) is a different story
  - Specified in Java EE where JavaBean is not
  - A component architecture for modular construction
  - Designed to handle persistence, transactional integrity and security
  - EJB Members: SessionBean, MessageDrivenBean, EntityBean

JAVABEAN CHARACTERISTICS

- A Java object implements java.io.Serializableto persist data through servers
- It has no argument constructor and no main method
- It contains many properties for holding data
- Allow access to properties using GETTE Rand SETTER methods with its naming convention
- Accessor methods (getter) to retrieve value
  - getXXX() method
- Mutatormethods (setter) to change value
  - setXXX() method

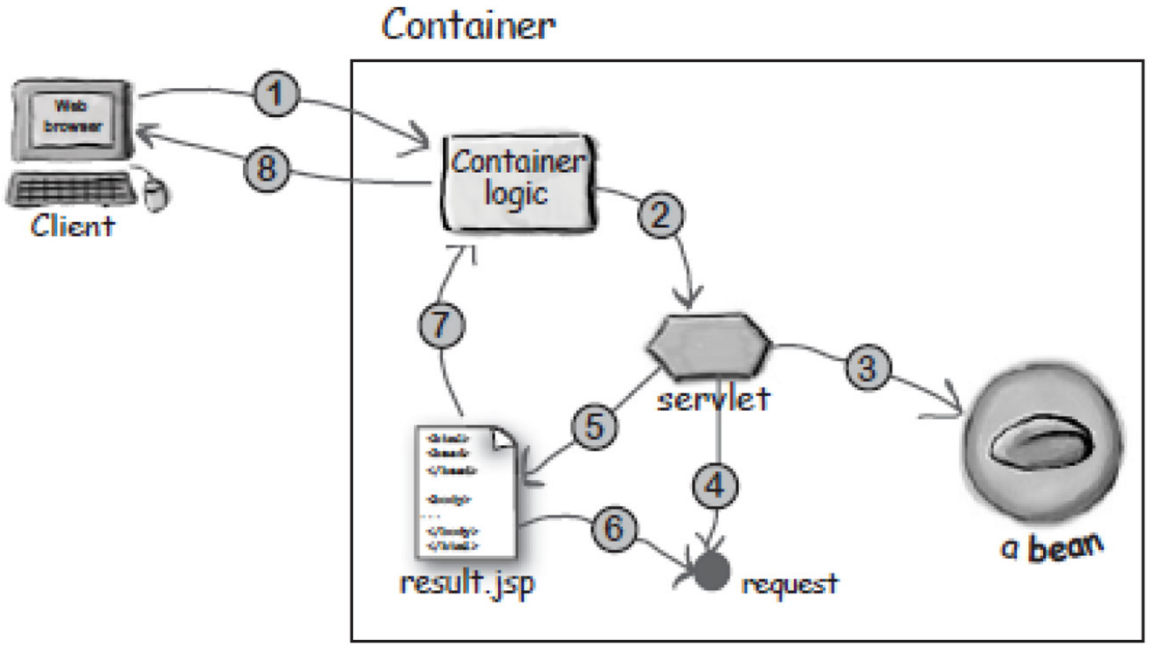
JAVABEAN EXAMPLE: PERSONBEAN

- Create a JavaBean (PersonBean) with two properties: name and gender.
- Modern IDE can help to generate the getter and setter methods.

```
01.  public class PersonBean implements java.io.Serializable {
02.    private String name, gender;
03.
04.    public String getName() { return name; }
05.
06.    public void setName(String name) { this.name = name; }
07.
08.    public String getGender() { return gender; }
09.
10.    public void setGender (String gender) { this.gender = gender; }
11.  }
```

TRANSACTION WORKFLOW DIAGRAM

- Using different web technologies (Java Servlet, JSP, JavaBean) together on a backend process for an online transaction.

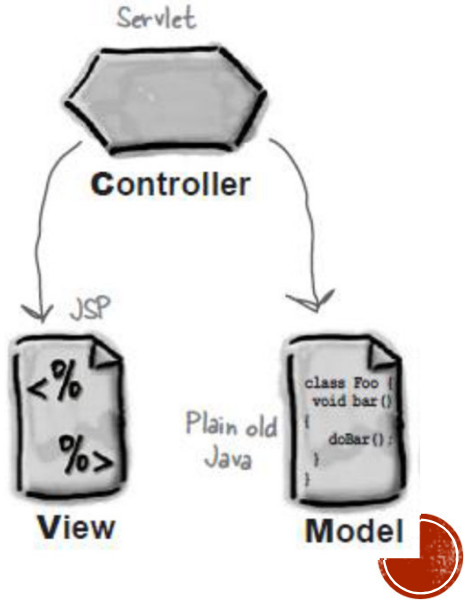


TRANSACTION WORKFLOW

1. Client sends a request to the web container;
2. Container finds the corresponding servlet and pass the request data to that servlet;
3. Servlet processes the business logic and store the result data object into a JavaBean;
4. Servlet adds that JavaBean to the session object inside the request object;
5. Servlet forwards to JSP;
6. JSP gets the JavaBean from the session object;
7. JSP converts JavaBean into HTML page format for the Container;
8. Container translates the JSP into plain HTML codes and returns back to client.

MODEL VIEW CONTROLLER

- Design Pattern for building web application
  - Making program more reusable, extensible, and easier for maintenance
  - Use to deal with large scale project
- Break down the functional sectors into pieces instead of writing everything inside a single program
- Separate the development roles
  - Data Modeling for system architect
  - Business logic and workflow for programmer
  - Presentation for web designer
- Categorize the web technologies
  - Model: JavaBean, XML, etc.
  - View: JSP, HTML, JavaScript, Swing, JavaFX, etc.
  - Controller: Java Servlet, etc.
- Many web development frameworks are based on MVC design pattern



MVC (MODEL)

- A Java program to contain static data which is part of the data model.


```java
package ipm.esap.comp413.util;

import java.util.Hashtable;

public class StockInfo {
  private static Hashtable<String, String> stockName = new Hashtable<String, String>();

  static {
    stockName.put("00001", "AAA Company");
    stockName.put("00002", "BBB Company");
    stockName.put("00003", "CCC Company");
    stockName.put("00003", "KKK Company");
    stockName.put("00004", "DDD Company");
    stockName.put("00005", "EEE Company");
    stockName.put("00006", "FFF Company");
    stockName.put("00007", "GGG Company");
    stockName.put("00008", "HHH Company");
    stockName.put("00009", "III Company");
    stockName.put("00010", "JJJ Company");
  }

  public static String getStockName(String num) {
    return (stockName.get(num) == null ? "n/a" : stockName.get(num));
  }

  public static double getStockPrice(String num) {
    String stock = stockName.get(num);
    if (stock == null) {
      return 0.0;
    } else {
      return Math.random() * 100;
    }
  }
}
```

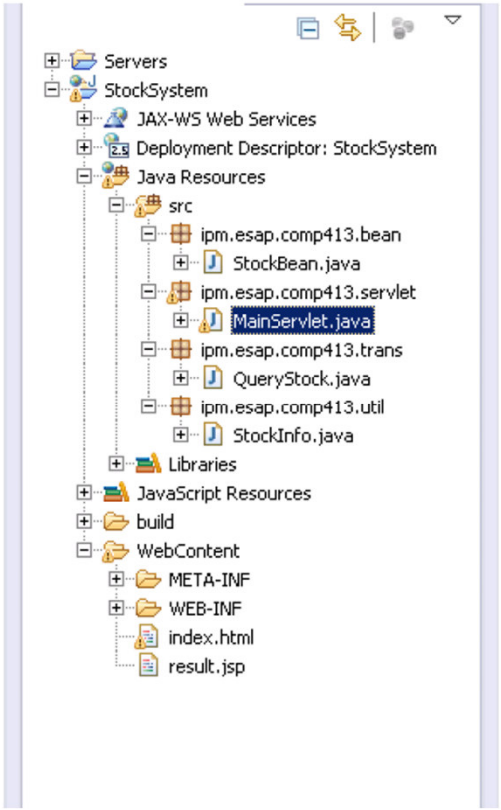
MVC (MODEL)

- A Java Bean to hold the transactional data for passing between programs.
- The transaction data model is defined according to user requirements.


```
1  package ipm.esap.comp413.bean;
2
3  public class StockBean {
4    private String stockNum, stockName, stockPrice;
5
6    public StockBean() {
7      super();
8    }
9
10   public String getStockNum() {
11     return stockNum;
12   }
13
14   public void setStockNum(String stockNum) {
15     this.stockNum = stockNum;
16   }
17
18   public String getStockName() {
19     return stockName;
20   }
21
22   public void setStockName(String stockName) {
23     this.stockName = stockName;
24   }
25
26   public String getStockPrice() {
27     return stockPrice;
28   }
29
30   public void setStockPrice(String stockPrice) {
31     this.stockPrice = stockPrice;
32   }
33 }
```

MVC (CONTROL)

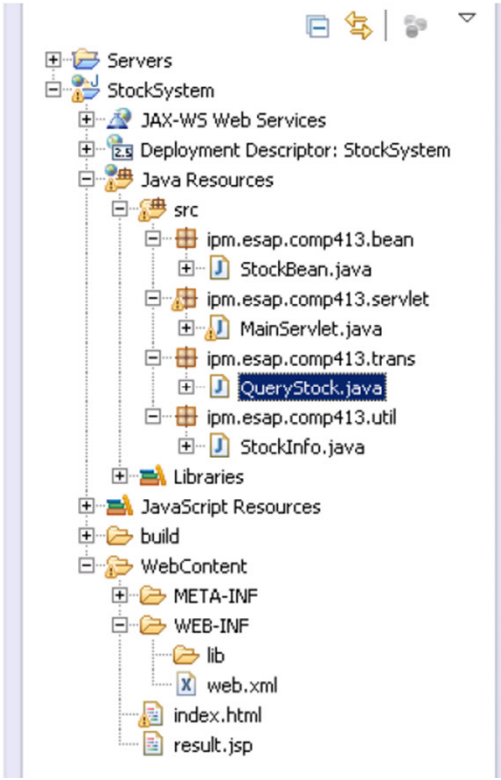
- A Java Servlet to control the transaction flow.
- Changing the flow will not affect other programs.




```java
package ipm.esap.comp413.servlet;

import ipm.esap.comp413.bean.StockBean;
import ipm.esap.comp413.trans.QueryStock;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class MainServlet extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    doPost(request, response);
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response) {
    try {
      QueryStock trans = new QueryStock();
      StockBean bean = trans.doTransaction(request);
      HttpSession session = request.getSession(true);
      session.setAttribute("Bean", bean);
      response.sendRedirect("result.jsp");
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
}
```

MVC (CONTROL)

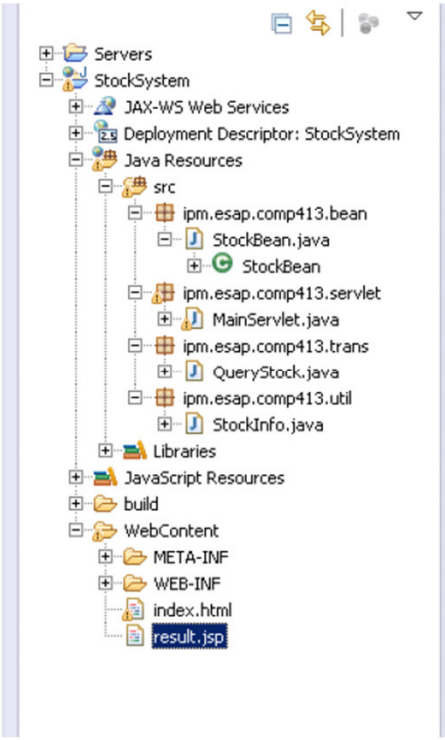
- A Java program to create a transaction to query data.
- The core program to apply the business logic.
- This query transaction can be reused.




```java
package ipm.esap.comp413.trans;

import ipm.esap.comp413.bean.StockBean;
import ipm.esap.comp413.util.StockInfo;

import java.text.DecimalFormat;

import javax.servlet.http.HttpServletRequest;

public class QueryStock {
  public StockBean doTransaction(HttpServletRequest request) throws Exception {
    DecimalFormat df = new DecimalFormat();
    df.applyPattern("#,###,##0.00");
    String stockNum = request.getParameter("StockNum");
    StockBean bean = new StockBean();
    bean.setStockNum(stockNum);
    bean.setStockName(StockInfo.getStockName(stockNum));
    bean.setStockPrice(df.format(StockInfo.getStockPrice(stockNum)));
    return bean;
  }
}
```

MVC (VIEW)

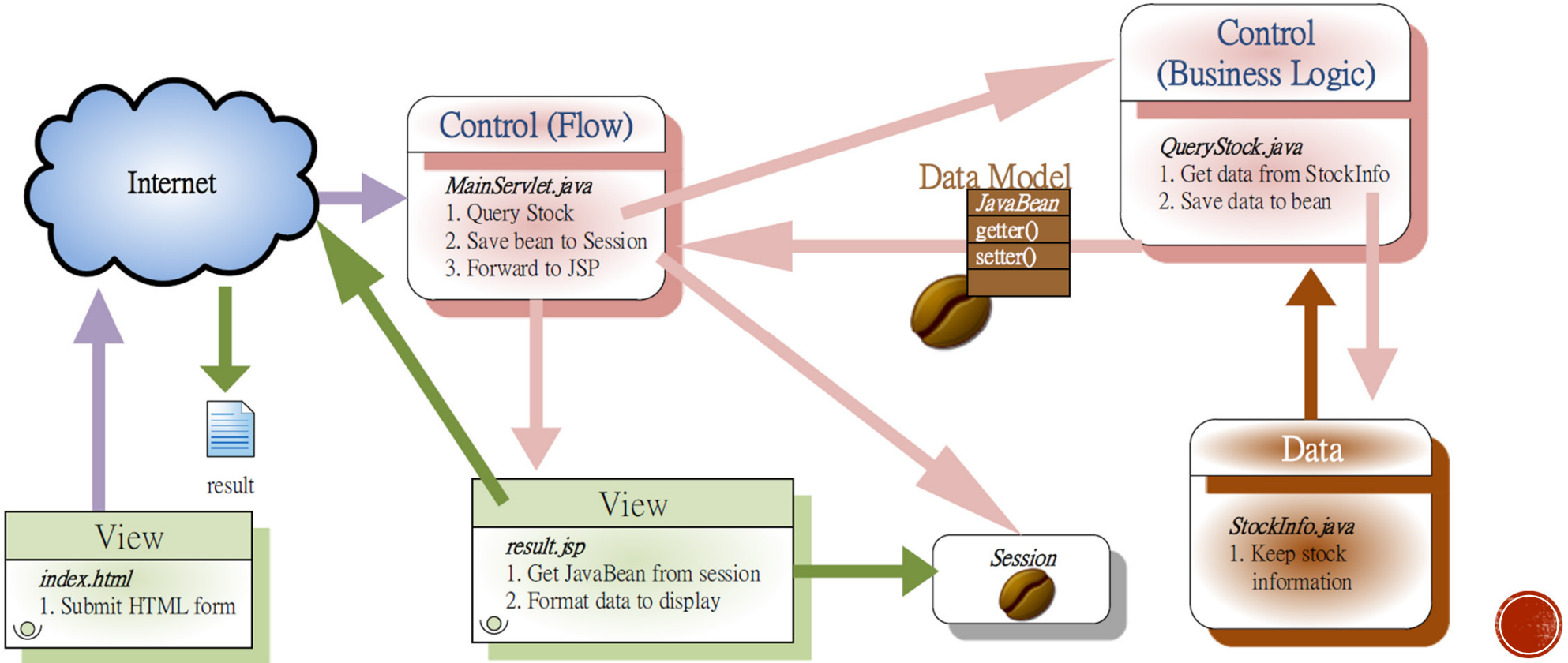
- JSP to present the result data (JavaBean) to client.
- Clients can have different presentation to show the JavaBean.
- Allow web designer to modify the JSP as ordinary web pages.




```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="ipm.esap.comp413.bean.StockBean" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Stock Enquiry System</title>
</head>

<body>
<%
  session = request.getSession(true);
  StockBean bean = (StockBean) session.getAttribute("Bean");
%>
<table border="1">
  <tr>
    <td>Stock Number</td>
    <td>Stock Name</td>
    <td>Stock Price</td>
  </tr>
  <tr>
    <td><%=bean.getStockNum()%></td>
    <td><%=bean.getStockName()%></td>
    <td><%=bean.getStockPrice()%></td>
  </tr>
</table>
</body>
</html>
```

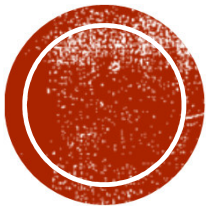
TRANSACTION FLOW DIAGRAM

- This transaction is now broken down into modules by adopting the MVC design pattern.



MVC FRAMEWORK

- MVC is a design pattern mainly to separate logic, flow, display, and data model into different programs.
- Since the MVC programs are following a pattern, they must share some similarities. Therefore, software engineers will make abstraction to minimize the repeating codes. As a result, new software framework is formed.
- Software framework that based on MVC model
- PHP –AppFlower, CakePHP, CodeIgniter, Fakoli, FuelPHP, Joomla, Kajona, PHPixie, Prado, Symfony, Yii, Zend, Zikula, etc.
- Java –Apache OfBiz, Sling, Struts, Tapestry, JavaServerFaces, JBossSeam, Spring, Stripes, Wavemaker, WebObjects, ztemplates, ZKoss, etc.
- http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks



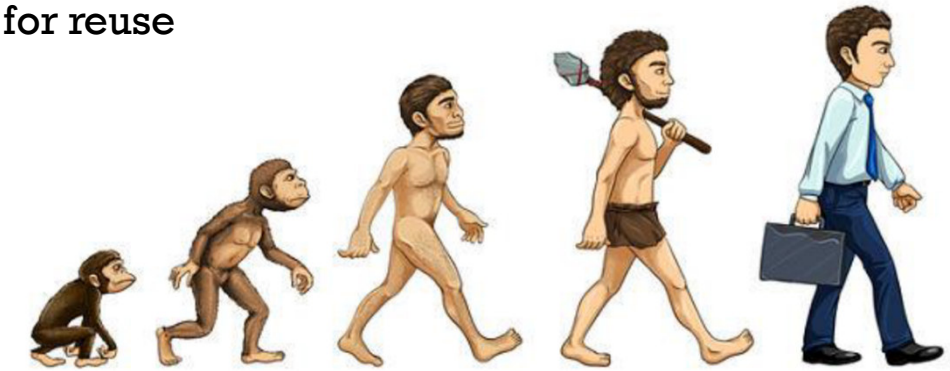
SOFTWARE FRAMEWORK

HOW TO START BUILDING AN ENTERPRISE APPLICATION?

- We know MVC design pattern, database manipulation technologies but where should I start to build an enterprise application first?
- When a new project comes, I need to build an application from scratch on each time but I know there are many common parts between projects.
- It turns out that we are keep repeating develop the same modules (user management, database connection, transaction flow, etc.).
- Is it a better way to reuse the developed programs from our previous projects?

DEVELOPMENT EVOLUTION

- All-in-one Approach
  - Mixing all processes in a single program
- Pattern Design
  - Design the programs to follow the same pattern.
  - Distribute the MVC modules (Data Model, View, Business Logic, and workflow) into different programs.
- Software Framework
  - Transform the pattern into a higher level for reuse



MVC DESIGN PATTERN

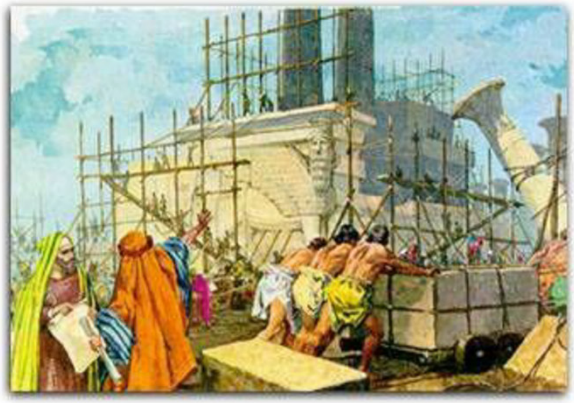
- A transaction can be broken down into MVC modules.
- When there are many transactions, similar codes and patterns are formed. Then, people start to make abstraction for simplification.
- As a result, a software framework can be created based on those abstractions.

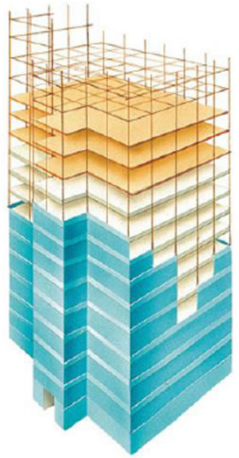
DESIGNING A SOFTWARE FRAMEWORK

- The general usage of software framework is to facilitate people on building applications.
- Focus on different parts of a transaction to make abstraction or optimization.
  - Controller
    - Application types: Portlet, Servlet, JApplet, etc.
    - Abstract to XML as a framework: struts, Spring, etc.
  - Data Model
    - Abstract to XML as a framework: Hibernate, NoSQL, RESTful, etc.
  - Presentation
    - Application types: HTML (web), Pane (standalone), View (mobile), etc.
    - Allows to write HTML on its program: ASP, JSP, PHP, etc.
    - Abstract to XML as a framework: JSF, Android, etc.

BUILDING A SKYSCRAPER

- Building modern skyscrapers will not be the same as building the a temple in the ancient time.
  - Using a lot of manpower, stone by stone, and taking ages.
  - Using steel framework for building the skeleton and then just plug the pre-built rooms or the curtain walls with different functions (kitchen, toilet, etc.) for finishing.





- Building an enterprise application can use the ideas of framework. Developers will focus only on making "rooms".

WHAT EXACTLY IT IS?

- Software Framework is an abstraction of software to provide generic functions that can be selectively changed by users.
- It is a universal, reusable software module used to develop applications, products, and solutions.
- Tries to make generalizations about the common tasks and workflow.
- In an object oriented (OO) environment, a framework consists of abstract and concrete classes. Instantiate a framework consists of composing and sub-classing the existing classes.
- Allow programmers to develop a project in a shorter time and to develop more reliable systems.

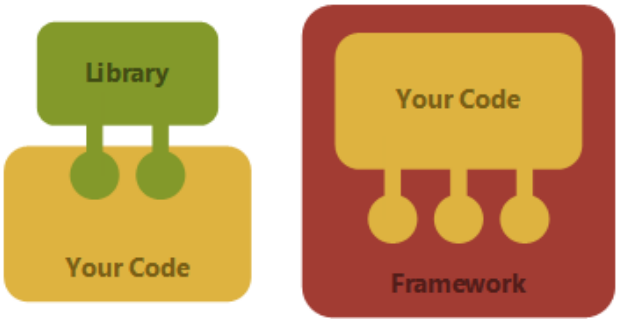
SOFTWARE FRAMEWORK VS LIBRARY

Library

- Library is a set of reusable functions used by your program
- They are important in program linking and binding process
- Code calls to library
- Javax.awt is a Java library for 2D Graphics.

Framework

- It is a piece of code that dictates the architecture of our project
- They provide a proper standard way to build an deploy program
- Framework calls your code
- JSF is a frame work for view



FRAMEWORK CHARACTERISTICS

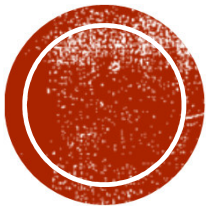
- The following features make software framework different from normal libraries:
- Inversion of Control (IoC)
  - Unlike procedural programming style as the flow of the business logic is statically assigned, IoC will assign the business logic at run time.
- Default Behavior
  - Framework has default behavior to help finish some tasks.
- Extensibility
  - It is a software that required to extend and is not finalized product.
- Non-modifiable code
  - The framework code in general is not allowed to be modified.

WEB APPLICATION FRAMEWORK

- Web application projects often adopt more than one frameworks.
- There are plenty of frameworks in the market and is expanding. You will see new frameworks appear over the time.
- Java: Grails, Google Web, Hibernate, IceFaces, JSF, Liferay, Seam, Oracle ADF, Sling, Spring, Struts, WebObjects, ZK, ztemplates, etc.
- PHP: CakePHP, Drupal, Joomla! Platform, PRADO, Symfony, ZendFramework, etc.
- ASP.NET: ASP.NET MVC, BFC, CSLA, DotNetNuke, Entity Framework, MVVM, Nancy, PostSharp, Prism, Web Forms MVP, WPF, etc.
- Most of the web application framework are based on the latest web technologies such as web 3.0, HTML 5, UTF-8, JSON, etc.
- http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

THE REASON OF USING FRAMEWORK

- Aim to facilitate software developers to focus on meeting the software requirements rather than dealing with **low-level or tedious development**. Thereby reducing the development time for building everything from scratch.
- Some companies might have their own design patterns or in-house developed framework. By using the modern framework, you don't need to spend time on training your development team to learn that non standard framework.
- In modern framework, the control flows are commonly written in XML files unlike traditional MVC pattern are written in Java Servlet.
- Provide more latest fancy interfaces (mobile, HTML5, JavaScript, etc.) for better user experiences.
- Some frameworks even provide the fundamental modules (user management, etc.).



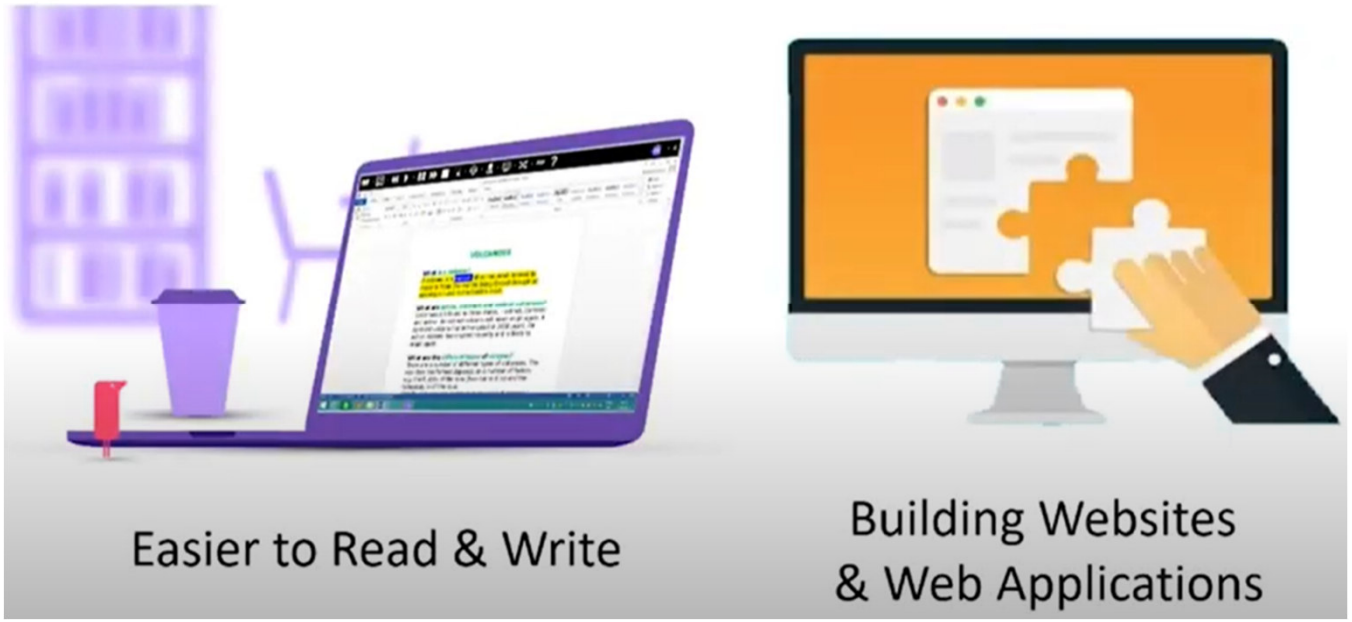
JAVA SERVER FACES (JSF)

DISCUSSION – WHY JSP + SERVLET?

- Both Java Server Pages and Java Servlet technologies are pretty old (almost 20 years now).
- Pure JSP has been replaced by Java Server Faces (JSF) and other MVC frameworks, which is where we will focus on moving forward
- Even Java Servlets, for the most part, have been replaced by Controllers in all MVC frameworks although they are used by the MVC frameworks.

JAVA SERVER FACES

- What is JSF?
  - JSF is a server side component framework for building websites and web applications.
  - JSF makes websites easier to read and write.



BENEFITS OF JSF

- Separates presentation code and behavioral code from each other
  - Presentation code is the code that make what you see and interact with
  - Behavioral code is the code that what the website or application is doing

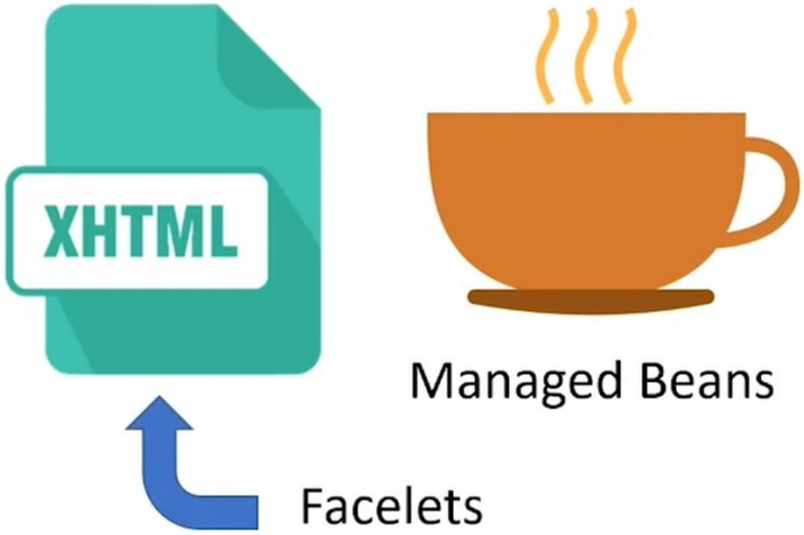
Presentational Code

Behavioural Code

BENEFITS OF JSF

- Before JSF

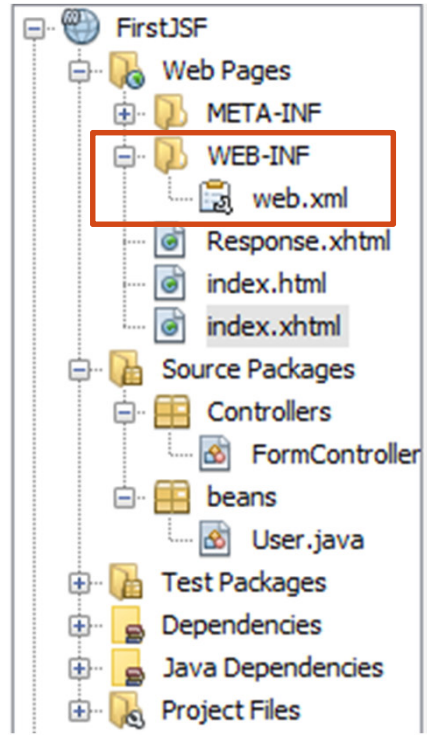
- After JSF



THE FIRST JSF EXAMPLE: (WEB.XML)

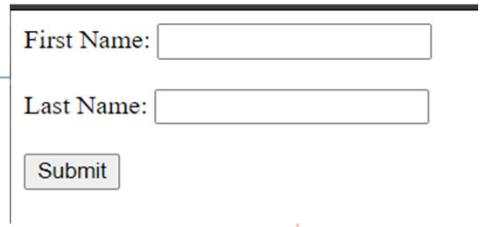
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

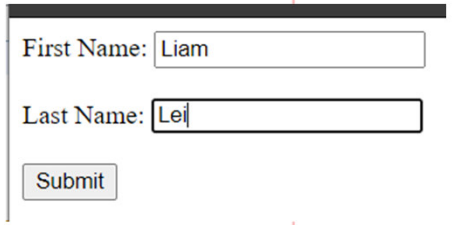
Welcome Page



THE FIRST JSF EXAMPLE: (INDEX.XHTML)

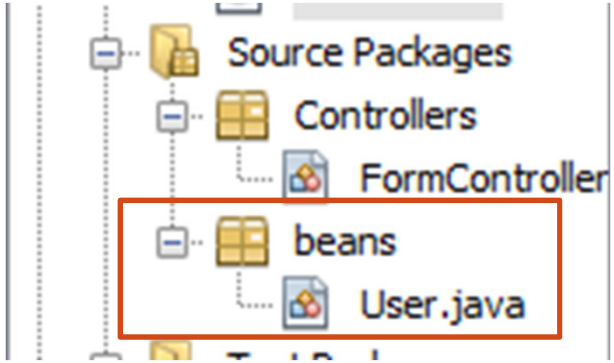
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>The First JSF</title>
    </h:head>
    <h:body>
        <h:form>
            First Name:   <h:inputText value="#{user.firstName}" /> <br/><br/>
            Last Name:    <h:inputText value="#{user.lastName}" /><br/><br/>
            <h:commandButton value="Submit" action="#{formController.onSubmit()}"/>
        </h:form>
    </h:body>
</html>
```





THE FIRST JSF EXAMPLE: (USER.JAVA)

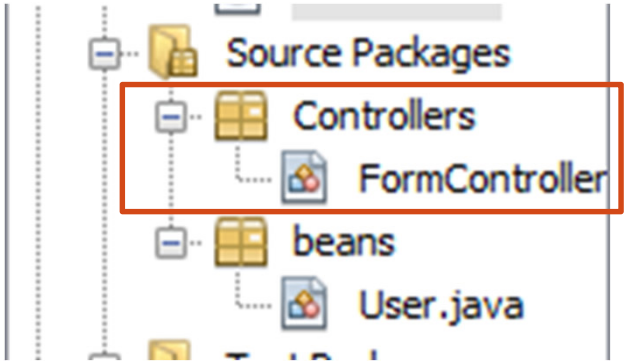
- ManagedBean




```
1   package beans;
2
4   import javax.faces.bean.ManagedBean;
5
6   @ManagedBean
7
8   public class User {
9
10      private String firstName;
11      private String lastName;
12
13      public User() {...2 lines }
15
16      public String getFirstName() {...3 lines }
19
20      public void setFirstName(String firstName) {...3 lines }
23
24      public String getLastName() {...3 lines }
27
28      public void setLastName(String lastName) {...3 lines }
31
32  }
```

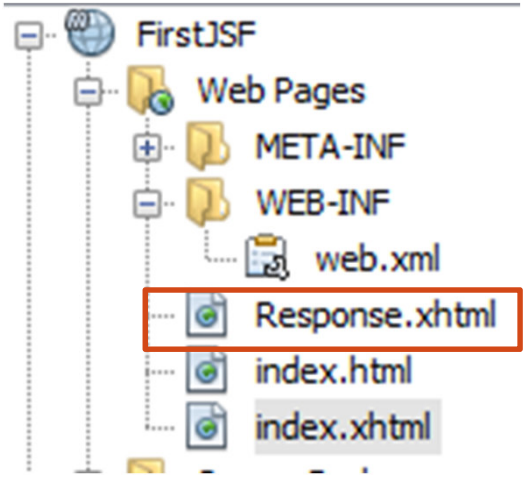
THE FIRST JSF EXAMPLE: (FORMCONTROLLER.JAVA)

- ManagedBean




```java
package Controllers;

import beans.User;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

@ManagedBean
public class FormController {

    public String onSubmit(){

        // get the user values from the input form.
        FacesContext context = FacesContext.getCurrentInstance();
        User user = context.getApplication().evaluateExpressionGet(context, "#{user}", User.class);

        // show the user object data in the console log
        System.out.println("your first name is :" + user.getFirstName());
        System.out.println("Your last name is: " + user.getLastName());
        System.out.println("You clikc the submit button.");

        // put the user object into the POST request
        FacesContext.getCurrentInstance().getExternalContext().getRequestMap().put("user", user);

        // show the next page
        return "Response.xhtml";
    }
}
```

THE FIRST JSF EXAMPLE: (RESPONSE.XHTML)




```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html>
3  <html xmlns="http://www.w3.org/1999/xhtml">
4      <head>
5          <title>JSF Response Page</title>
6          <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
7      </head>
8      <body>
9          <h1>Response Page</h1>
10         <p>Hello #{user.firstName}, I holp you doing well</p>
11         <p>I believe your last name is #{user.lastName}</p>
12
13     </body>
14 </html>
```

WHAT ARE ALL THESE JSF HTML TAGS?

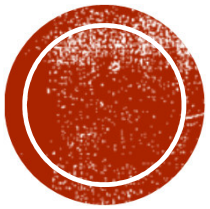
- When you build JSF Pages most of the HTML tags have been replaced with an equivalent JSF tag.
- The JSF tags are generally pretty similar to their equivalent HTML tag but add support for data binding, AJAX, etc.
- Remember JSF DOES NOT have to render HTML so these tags are "wrapper" classes over what they ultimately render.
- One of the better resources to learn Enterprise Java is from the Oracle Tutorials.
- For Java EE 7 go to: https://docs.oracle.com/javaee/7/tutorial/index.html
- For Java EE 8 go to: https://javaee.github.io/tutorial/toc.html

WHAT ARE JSF FACELETS?

- Facelets enable you to build reusable UI Components and Templates.
- One common use of Facelets is to support Layouts, which allows you to create "page fragments" that can be used to build common partial pages that can be assembled to form a final complete HTML page.
- This allows you to build common headers, footers, and menus that can be combined with your core page content to build a single HTML page.
- For documentation:
  - Java EE 7 docs: https://docs.oracle.com/javaee/7/tutorial/
  - For Facelets drill down into Part III and Chapter 8.4

WHAT STEPS ARE REQUIRED TO DESIGN LAYOUTS IN JSF?

- Design a Layout View
- Design one or more reusable page fragments
- Design each of your JSF Pages to use the Layout.



ENTERPRISE JAVA BEAN (EJB)

Introduction to EJB

INTRODUCTION

- Enterprise Java Beans (EJB) is
  - Enterprise JavaBeans are server-side components that encapsulate the business logic of an application. The business logic is the code that fulfills the purpose of the application. It DOES NOT perform display of business data (presentation) or perform operations directly the database (persistence).
  - Enterprise JavaBeans (EJB) simplify application development by automatically taking care of transaction management and security.
  - There are two types of Enterprise JavaBean:
    - session beans, which perform business logic,
    - and message-driven beans, which act as a message listener.

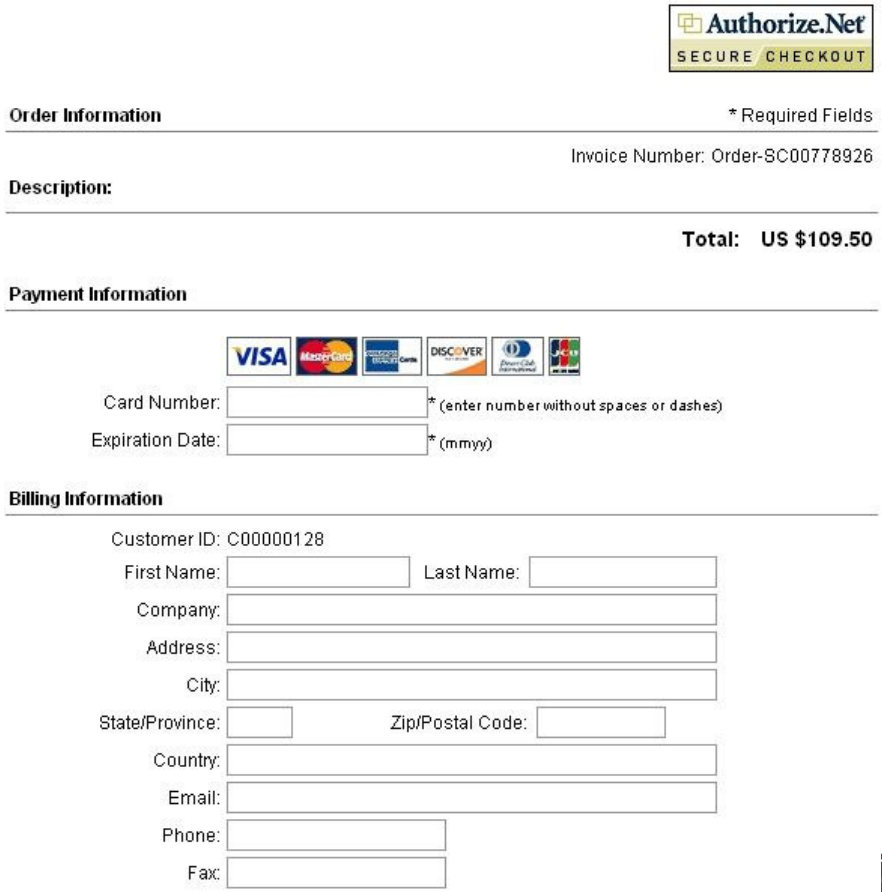
WHAT IS BUSINESS LOGIC?

- Prescribes how business objects interact with one another.
- Business logic is the portion of an enterprise system which determines how data is transformed or calculate, and how it is routed to people or software (workflow)

- For example: Tracking the package:
- Order workflow:
  1. Customer creates and order
  2. Warehouse prepares order
  3. Shipping puts the order on a truck
  4. Confirm delivery at your door.

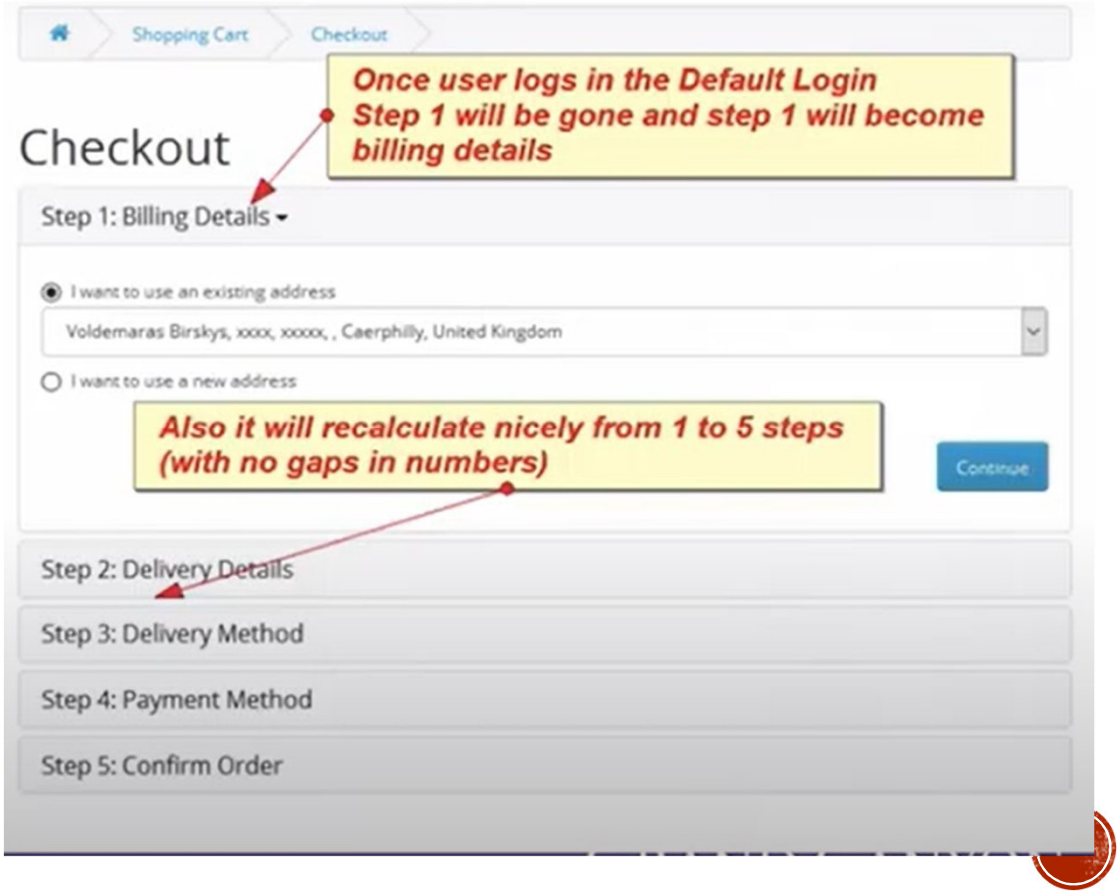
WHAT IS BUSINESS LOGIC?

- For example for an eCommerce site: Allow visitors to add items to a shopping cart, specify a shipping address, and supply payment information.



WHAT IS BUSINESS LOGIC?

- The business logic of the website might include workflow such as: The sequence of events that happens during the checkout
  - First asks for the shipping address
  - Then Billing address
  - Next page will contain the payment method
  - And last page will show congratulations
- Think workflow, what tasks, and what order.



BUSINESS LOGIC – DECISIONS TO BE MADE

	Rules		
Conditions	**1**	**2**	**3**
C1. Destination is in outlying island	Y		
C2. Transaction > $80		Y	N
C3. Weight > 10 kg		N	
Actions	**1**	**2**	**3**
A1. Provide free delivery		X	
A2. Refuse delivery	X		X

WHAT IS A BUSINESS RULE?

- A business rule that defines or constrains some aspect of business and always resolves to either true or false
- Business rules describe the operations, definitions and constraints that apply to an organization.
- Business rules can apply to people, processes, corporate behavior and computing systems in an organization, and are put in place to help the organization achieve its goals.
- For example:
  - No credit check is to be performed on return customers.
  - Cannot withdraw more money from your account then your balance.
  - Cannot buy items with a credit card if the card authorization fails.

EXAMPLE BUSINESS LOGIC FOR A SHOPPING CART

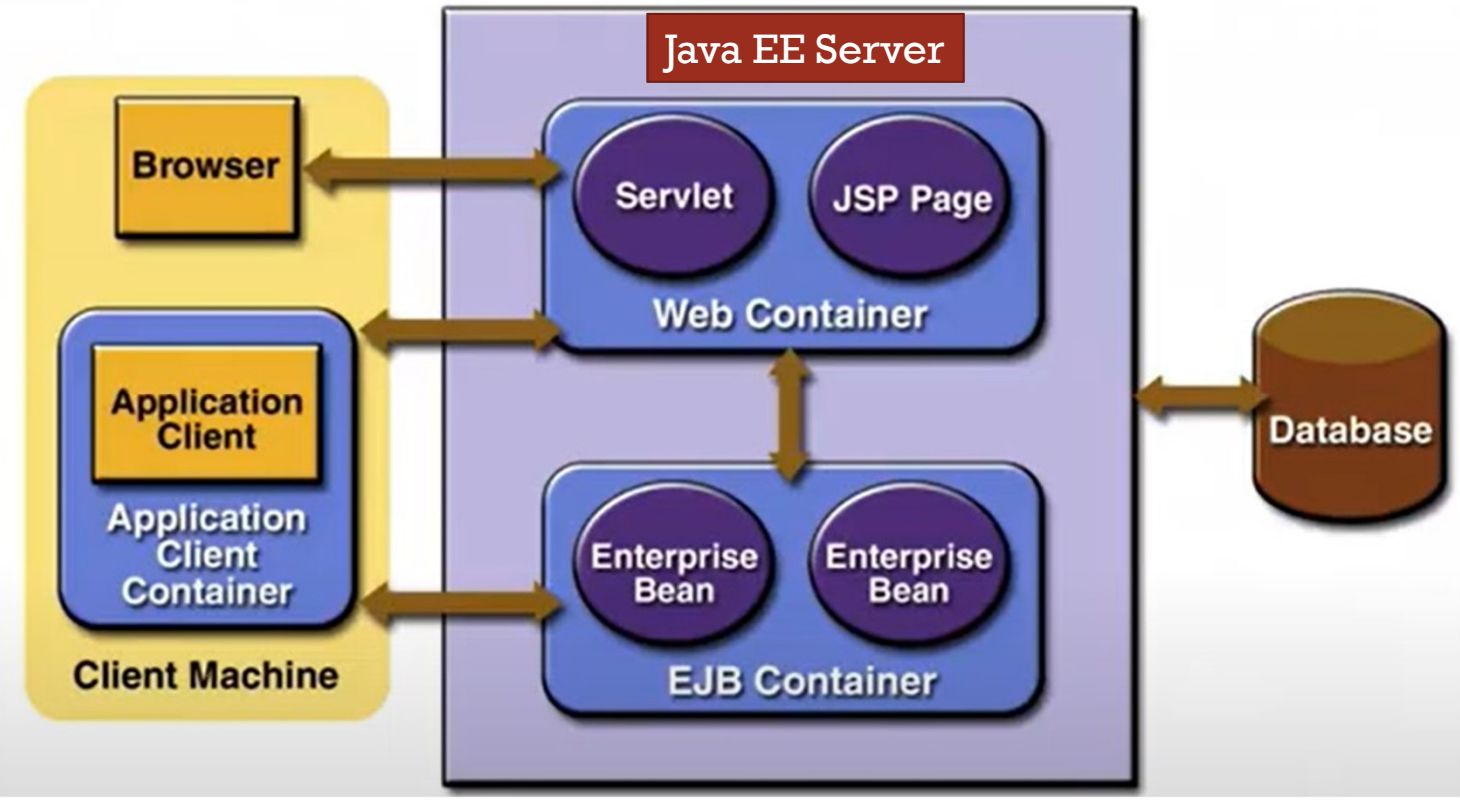
```
public class Cart {

private int itemCount;
private double totalPrice;
private static int capacity;
private static Item[] cart = new
Item[capacity];

public Cart(){
    itemCount = 10;
    totalPrice = 0.0;
    capacity = 0;
}
```

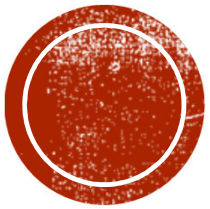
```
public void add(int itemID, String itemName,
double itemPrice, String itemDescription, int
itemQuantity, double itemTax){
    Item item = new Item(itemID, itemName,
itemPrice, itemDescription, itemQuantity,
itemTax);
    totalPrice += (itemPrice * itemQuantity);
    cart[itemCount] = item;
    itemCount += 1;
    if(itemCount==capacity)
    {
        increaseSize();
    }
}

more methods to follow...
```

EJB CONTAINER

- An EJB container manages the enterprise beans contained within it.
- For each enterprise bean, the container is responsible for
  - registering the object,
  - providing a remote interface for the object,
  - creating and destroying object instances,
  - checking security for the object,
  - managing the active state for the object,
  - and coordinating distributed transactions.

EJB CONTAINER





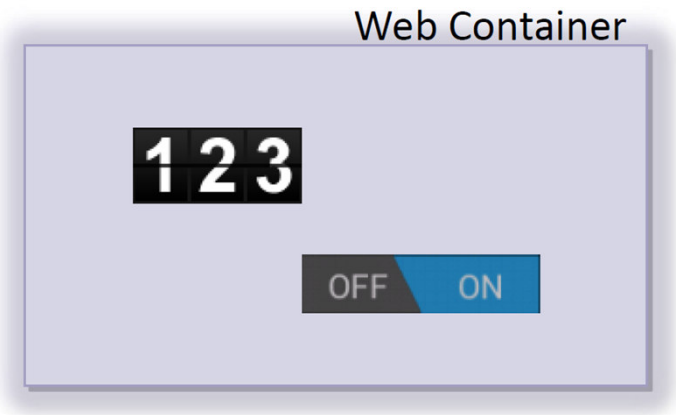
SCOPE OF EJB

STATELESS AND STATEFUL

- What are the differences?
  - Stateless objects for all users (application level).
  - Stateful objects for individual user (session level).
- Stateless objects can be a service for all users
  - They created when a method/application is called/started.
  - They are the same objects visible to all users but don't hold state for each user.
- Stateful objects are not shared among users
  - They hold state for individual user.
  - They will die when client sessions are ended or timeout.

EXAMPLE OF STATELESS AND STATEFUL

- Stateless
- unique
  - Application level
  - counters, switches, etc.
  - Every user will retrieve the same object

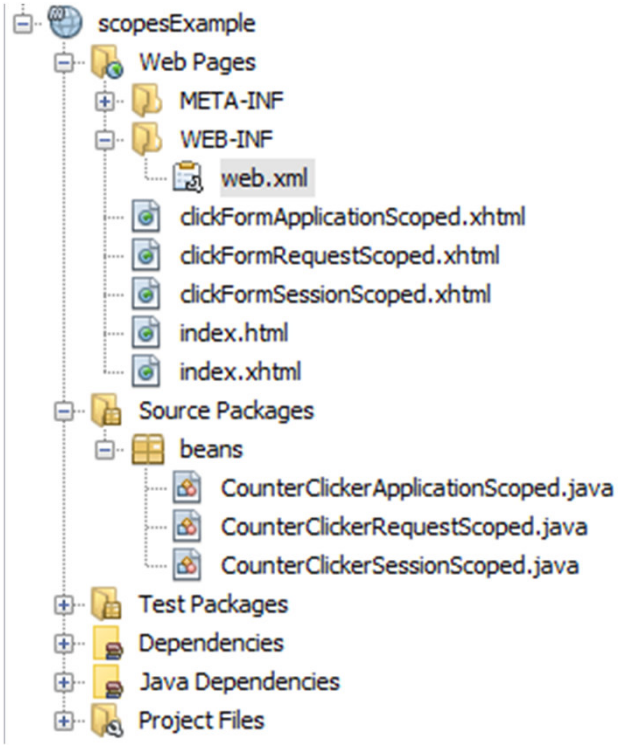


- Stateful
  - multiple
  - Session level
  - shopping carts, etc.
  - Every user has its own object to hold different value

MANIPULATING OBJECTS WITH SESSION

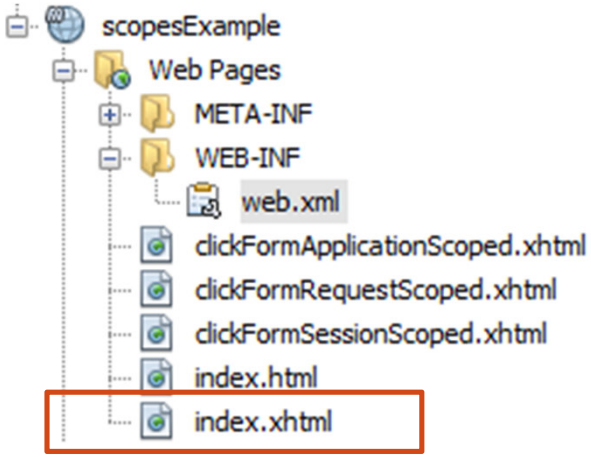
- It is the most common approach in online system. Store the data in a session, and any pages or programs can get the data back from the session.
- Once the session is created, no need to handle passing the objects page by page.
- Session objects will be killed or terminated when user logout the application or close the browser.
- Server side can control the maximum idle time for a session to timeout.

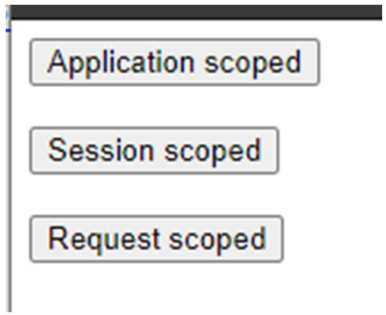
SCOPE EXAMPLE: (WEB.XML)




```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">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

SCOPE EXAMPLE: (INDEX.HTML)






```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml"
5        xmlns:h="http://xmlns.jcp.org/jsf/html">
6      <h:head>
7          <title>Scoped Examples</title>
8      </h:head>
9      <h:body>
10         <h:form>
11             <h:commandButton action="clickFormApplicationScoped.xhtml"
12                              value="Application scoped"/> <br/><br/>
13             <h:commandButton action="clickFormSessionScoped.xhtml"
14                              value="Session scoped"/> <br/><br/>
15             <h:commandButton action="clickFormRequestScoped.xhtml"
16                              value="Request scoped"/>
17         </h:form>
18     </h:body>
19 </html>
```

SCOPE EXAMPLE: (CLICKFORMSCOPE.XHTML)

- Application Scope
  - clickFormApplicationScoped.xhtml
- Session Scope
  - clickFormSessionScoped.xhtml
- Request Scope
  - clickFormRequestScoped.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <head>
        <title>Request Scoped</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    </head>
    <body>
        <p>
            Request scoped, Run this URL in different browsers.
            You should see that each request (click) has its own version of variable.
            The value is continually reset to its initial value 0 and then incremented.
            Use this type of scope for short term values such as a form submission.
            this is the default scope of a managed bean.
        </p>
        Counter Value: #{counterClickerRequestScoped.clickValue}

        <h:form>
            <h:commandButton action="#{counterClickerRequestScoped.increment()}"
                             value="Click to increment"/>
        </h:form>
    </body>
</html>
```

SCOPE EXAMPLE: (CLICKFORMSCOPE.XHTML)

```
15          Counter Value: #{counterClickerApplicationScoped.clickValue}
16
17          <h:form>
18              <h:commandButton action="#{counterClickerApplicationScoped.increment()}"
19                               value="Click to increment"/>
20          </h:form>
```

```
15          Counter Value: #{counterClickerSessionScoped.clickValue}
16
17          <h:form>
18              <h:commandButton action="#{counterClickerSessionScoped.increment()}"
19                               value="Click to increment"/>
20          </h:form>
```

```
16          </p>
17          Counter Value: #{counterClickerRequestScoped.clickValue}
18
19          <h:form>
20              <h:commandButton action="#{counterClickerRequestScoped.increment()}"
21                               value="Click to increment"/>
22          </h:form>
```

SCOPE EXAMPLE: MANAGED BEANS

```java
package beans;


import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

@ManagedBean
@RequestScoped

public class CounterClickerRequestScoped {

    private int clickValue = 0;

    public String increment(){
        setClickValue(getClickValue() + 1);
       return "clickFormRequestScoped.xhtml";
    }

    public int getClickValue() {
        return clickValue;
    }

    public void setClickValue(int clickValue) {
        this.clickValue = clickValue;
    }

}
```

```java
package beans;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped

public class CounterClickerSessionScoped {
 private int clickValue = 0;

    public String increment(){
        setClickValue(getClickValue() + 1);
       return "clickFormSessionScoped.xhtml";
    }

    public int getClickValue() {
        return clickValue;
    }

    public void setClickValue(int clickValue) {
        this.clickValue = clickValue;
    }

}
```

```java
package beans;

import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;

@ManagedBean
@ApplicationScoped

public class CounterClickerApplicationScoped {

    private int clickValue = 0;

    public String increment(){
        setClickValue(getClickValue() + 1);
       return "clickFormApplicationScoped.xhtml";
    }

    public int getClickValue() {
        return clickValue;
    }

    public void setClickValue(int clickValue) {
        this.clickValue = clickValue;
    }

}
```