

Java Server Pages

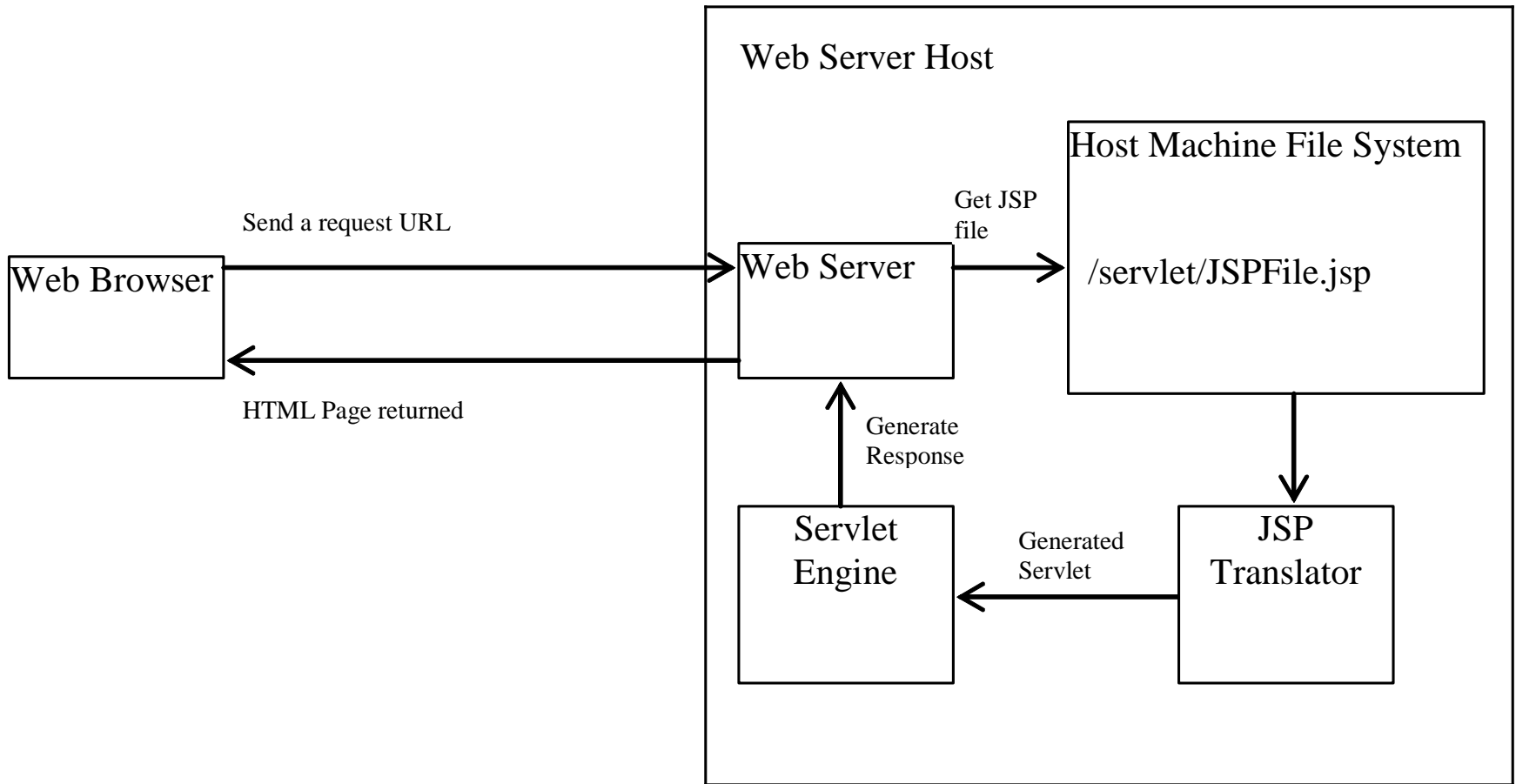
CSE 305 – Principles of Database Systems

Paul Fodor

Stony Brook University

<http://www.cs.stonybrook.edu/~cse305>

How Is a JSP Processed?



```
<!-- CurrentTime.jsp -->

<HTML>

<HEAD>

<TITLE>

    CurrentTime

</TITLE>

</HEAD>

<BODY>

    Current time is <%= new java.util.Date() %>

</BODY>

</HTML>
```

JSP Constructs

There are three types of scripting constructs you can use to insert Java code into the resultant servlet. They are *expressions*, *scriptlets*, and *declarations*.

expression

A JSP expression is used to insert a Java expression directly into the output.

scriptlet

It has the following form:

`<%= Java-expression %>`

declaration

The expression is evaluated, converted into a string, and sent to the output stream of the servlet.

JSP Constructs

There are three types of scripting constructs you can use to insert Java code into the resultant servlet. They are *expressions*, *scriptlets*, and *declarations*.

expression

scriptlet

declaration

A JSP scriptlet enables you to insert a Java statement into the servlet's `jspService` method, which is invoked by the service method. A JSP scriptlet has the following form:

`<% Java statement %>`

JSP Constructs

There are three types of scripting constructs you can use to insert Java code into the resultant servlet. They are *expressions*, *scriptlets*, and *declarations*.

expression

scriptlet

declaration

A JSP declaration is for declaring methods or fields into the servlet. It has the following form:

```
<% ! Java method or field declaration %>
```

JSP Comment

HTML comments have the following form:

`<!-- HTML Comment -->`

If you don't want the comment appear in the resultant HTML file, use the following comment in JSP:

`<%-- JSP Comment --%>`

Computing Factorials

```
<HTML>
<HEAD>
<TITLE>
Factorial
</TITLE>
</HEAD>
<BODY>
```

JSP scriptlet

```
<% for (int i = 0; i <= 10; i++) { %>
```

```
Factorial of <%= i %> is
```

```
<%= computeFactorial(i) %> <br />
```

```
<% } %>
```

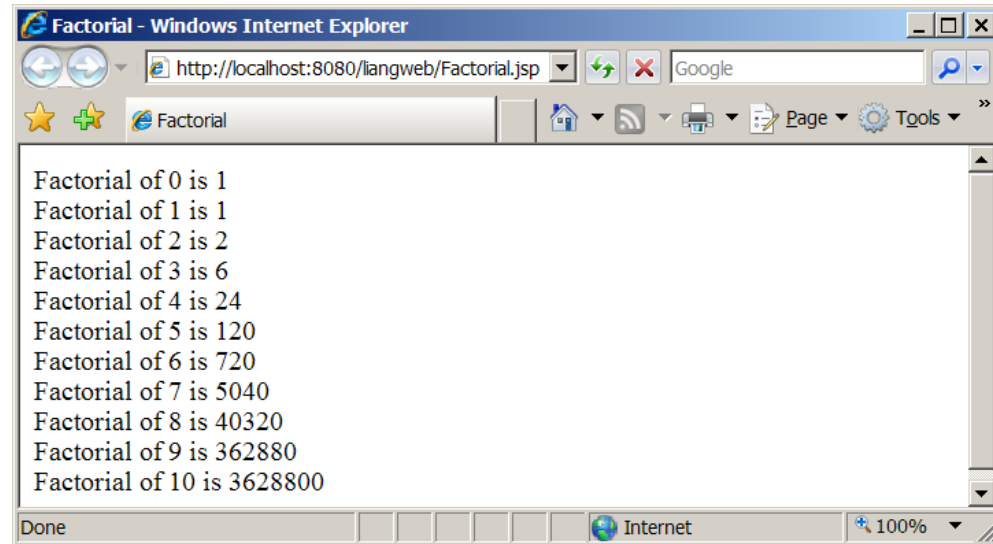
JSP expression

```
<%! private long computeFactorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * computeFactorial(n - 1);
}
%>
```

JSP declaration

```
</BODY>
```

```
</HTML>
```



JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request

response

out

session

application

config

pagecontext

page

Represents the client's request, which is an instance of `HttpServletRequest`.

You can use it to access request parameters, HTTP headers such as cookies, hostname, etc.

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request
response
out
session
application
config
pagecontext
page

Represents the servlet's response, which is an instance of `HttpServletResponse`.

You can use it to set response type and send output to the client.

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request
response
out
session
application
config
pagecontext
page

Represents the character output stream, which is an instance of `PrintWriter` obtained from `response.getWriter()`.

You can use it to send character content to the client.

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request
response
out
session
application
config
pagecontext
page

Represents the HttpSession object associated with the request, obtained from
`request.getSession()`

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request

response

out

session

application

config

pagecontext

page

Represents the ServletContext object for storing persistent data for all clients.

The difference between session and application is that session is tied to one client, but application is for all clients to share persistent data.

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request
response
out
session
application
config
pagecontext
page

Represents the ServletConfig object for the page.

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request

response

out

session

application

config

pagecontext

page

Represents the PageContext object.

PageContext is a new class introduced in JSP to give a central point of access to many page attributes.

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*.

request
response
out
session
application
config
pagecontext
page

Page is an alternative to **this**.

Computing Loan

An HTML page that prompts the user to enter loan amount, annual interest rate, and number of years. Clicking the Compute Loan Payment button invokes a JSP to compute and display the monthly and total loan payment.

```
<!-- ComputeLoan.html -->
```

```
<html>
```

```
<head>
```

```
<title>ComputeLoan</title>
```

```
</head>
```

```
<body>
```

```
Compute Loan Payment
```

```
<form method="get" action="ComputeLoan.jsp">
```

```
<p>Loan Amount
```

```
<input type="text" name="loanAmount"><br>
```

```
Annual Interest Rate
```

```
<input type="text" name="annualInterestRate"><br>
```

```
Number of Years <input type="text" name="numberOfYears" size="3"></p>
```

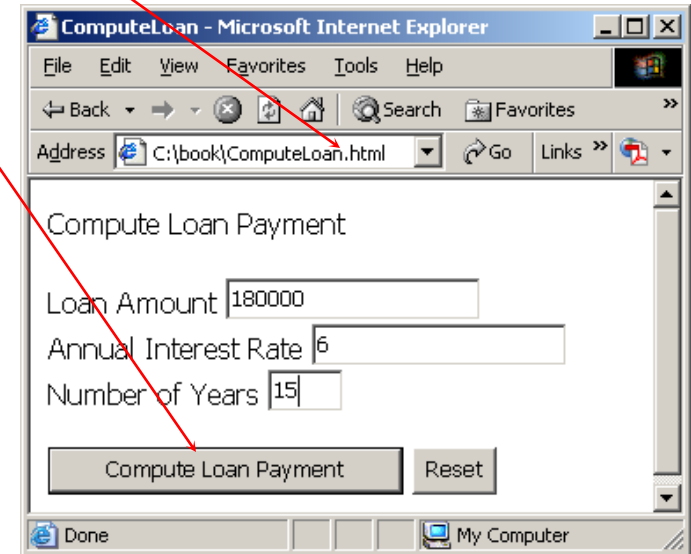
```
<p><input type="submit" name="Submit" value="Compute Loan Payment">
```

```
<input type="reset" value="Reset"></p>
```

```
</form>
```

```
</body>
```

```
</html>
```



```
<!-- ComputeLoan.jsp -->
```

```
<html>
```

```
<head>
```

```
<title>ComputeLoan</title>
```

```
</head>
```

```
<body>
```

```
<% double loanAmount = Double.parseDouble(
```

```
    request.getParameter("loanAmount");
```

```
    double annualInterestRate = Double.parseDouble(
```

```
    request.getParameter("annualInterestRate"));
```

```
    double numberOfYears = Integer.parseInt(
```

```
    request.getParameter("numberOfYears"));
```

```
    double monthlyInterestRate = annualInterestRate / 1200;
```

```
    double monthlyPayment = loanAmount * monthlyInterestRate /  
        (1 - 1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12));
```

```
    double totalPayment = monthlyPayment * numberOfYears * 12; %>
```

```
Loan Amount: <%= loanAmount %><br>
```

```
Annual Interest Rate: <%= annualInterestRate %><br>
```

```
Number of Years: <%= numberOfYears %><br>
```

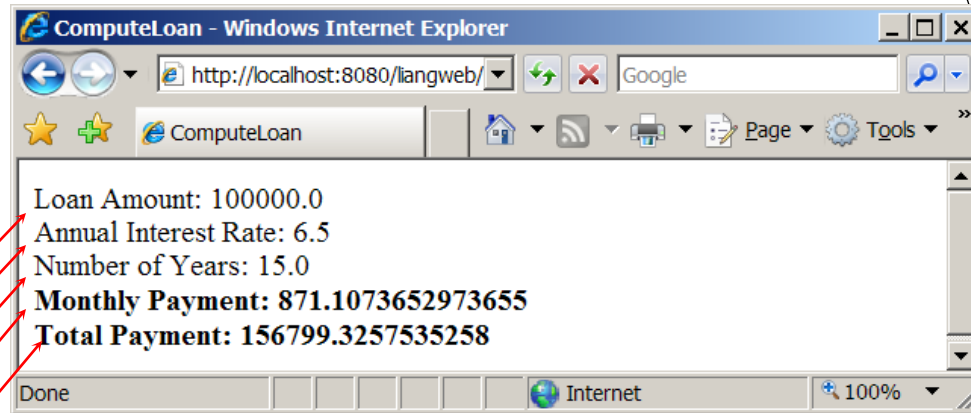
```
<b>Monthly Payment: <%= monthlyPayment %><br>
```

```
Total Payment: <%= totalPayment %><br></b>
```

```
</body>
```

```
</html>
```

Predefined
variable



JSP Directives

A JSP directive is a statement that gives the JSP engine information about the JSP page.

For example, if your JSP page uses a Java class from a package other than the `java.lang` package, you have to use a directive to import this package.

The general syntax for a JSP directive is as follows:

`<% @ directive attribute="value" %>`, or

`<% @ directive attribute1="value1" ...`

`attributen="valuen" %>`

For example, the directive

`<% @ page import="java.util.*, java.text.*" %>`

Three JSP Directives

Three possible directives are the following: page, include, and tablib.

page

include

tablib

page lets you provide information for the page, such as importing classes and setting up content type.

The page directive can appear anywhere in the JSP file.

Three JSP Directives

Three possible directives are the following: page, include, and tablib.

page

include

tablib

include lets you insert a file to the servlet when the page is translated to a servlet.

The include directive must be placed where you want the file to be inserted.

Three JSP Directives

Three possible directives are the following: page, include, and tablib.

page

include

tablib

tablib lets you define custom tags.

Attributes for page Directives

import

contentType

session

buffer

autoFlush

isThreadSafe

errorPage

isErrorPage

Specifies one or more packages to be imported for this page.

For example, the directive

```
<% @ page import="java.util.*, java.text.*" %>
```

imports java.util.* and java.text.*.

Attributes for page Directives

import

contentType

session

buffer

autoFlush

isThreadSafe

errorPage

isErrorPage

Specifies the MIME type for the resultant JSP page.

By default, the content type is text/html for JSP.

The default content type for servlets is text/plain.

Attributes for page Directives

import
contentType
session
buffer
autoFlush
isThreadSafe
errorPage
isErrorPage

Specifies a boolean value to indicate whether the page is part of the session.

By default, session is true.

Attributes for page Directives

import
contentType
session
buffer
autoFlush
isThreadSafe
errorPage
isErrorPage

Specifies the output stream buffer size.

By default, it is 8KB.

For example, the directive

`<% @ page buffer="10KB" %>`

specifies that the output buffer size is 10KB.

The directive

`<% @ page buffer="none" %>`

specifies that a buffer is not used.

Attributes for page Directives

import
contentType
session
buffer
autoFlush
isThreadSafe
errorPage
isErrorPage

Specifies a boolean value to indicate whether the output buffer should be automatically flushed when it is full or whether an exception should be raised when the buffer overflows.

By default, this attribute is true.

In this case, the buffer attribute cannot be none.

Attributes for page Directives

import
contentType
session
buffer
autoFlush
isThreadSafe
errorPage
isErrorPage

Specifies a boolean value to indicate whether the page can be accessed simultaneously without data corruption.

By default, it is true.

If it is set to false, the JSP page will be translated to a servlet that implements the SingleThreadModel interface.

Attributes for page Directives

import
contentType
session
buffer
autoFlush
isThreadSafe
errorPage
isErrorPage

errorPage specifies a JSP page that is processed when an exception occurs in the current page.

For example, the directive

```
<% @ page errorPage="HandleError.jsp" %>
```

specifies that HandleError.jsp is processed when an exception occurs.

isErrorPage specifies a boolean value to indicate whether the page can be used as an error page. By default, this attribute is false.

Example: Computing Loan Using the Loan Class

Use the Loan class to simplify Example 38.2. You can create an object of Loan class and use its `monthlyPayment()` and `totalPayment()` methods to compute the monthly payment and total payment.

```
<!-- ComputeLoan.jsp -->
<html>
<head>
<title>ComputeLoan Using the Loan Class</title>
</head>
<body>
<%@ page import = "cse305.Loan" %>
<% double loanAmount = Double.parseDouble(
    request.getParameter("loanAmount"));
    double annualInterestRate = Double.parseDouble(
    request.getParameter("annualInterestRate"));
    int numberOfYears = Integer.parseInt(
    request.getParameter("numberOfYears"));
    Loan loan = new Loan(annualInterestRate, numberOfYears,
        loanAmount);
    %>
    Loan Amount: <%= loanAmount %><br>
    Annual Interest Rate: <%= annualInterestRate %><br>
    Number of Years: <%= numberOfYears %><br>
    <b>Monthly Payment: <%= loan.monthlyPayment() %><br>
    Total Payment: <%= loan.totalPayment() %><br></b>
</body>
</html>
```

Import a class. The class must be placed in a package (e.g. package `cse305`).

JavaBeans Component in JSP

A class is a JavaBeans component if it has the following three features:

The class is public.

The class has a public constructor with no arguments.

The class is serializable. (This requirement is not necessary in JSP.)

Using JavaBeans in JSP

To create an instance for a JavaBeans component, use the following syntax:

```
<jsp:useBean id="objectName"  
scope="scopeAttribute" class="ClassName" />
```

This syntax is equivalent to

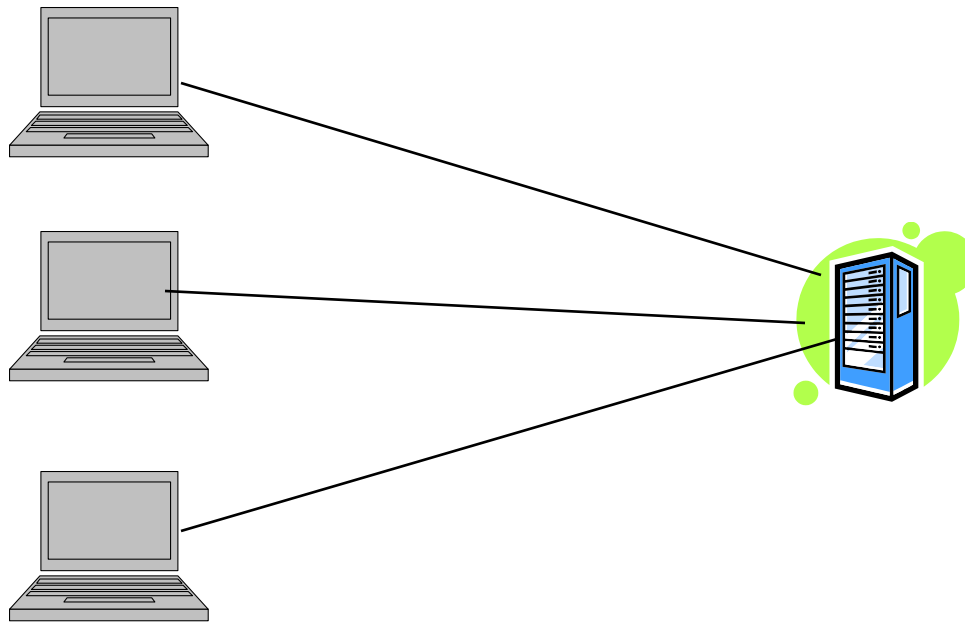
```
<% ClassName objectName = new ClassName() %>
```

except that the scope attribute specifies the scope of the object.

Scope Attributes

application
session
page
request

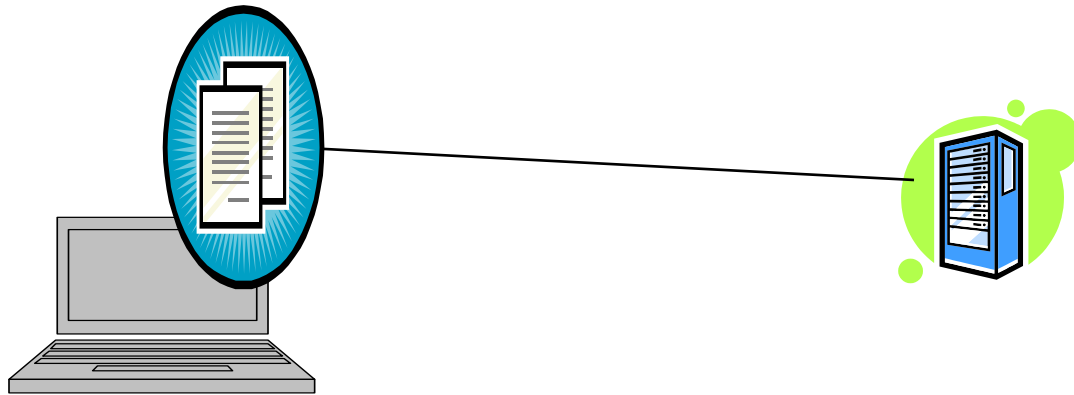
Specifies that the object is bound to the application. The object can be shared by all sessions of the application.



Scope Attributes

application
session
page
request

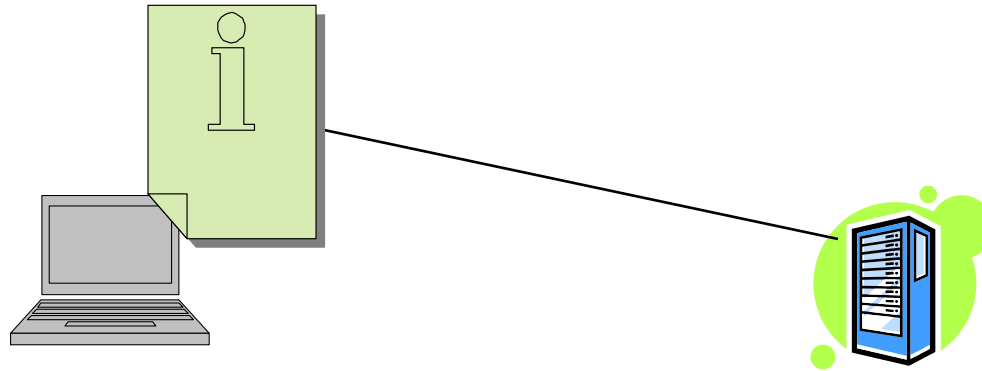
Specifies that the object is bound to the client's session. A client's session is automatically created between a Web browser and Web server. When a client from the same browser accesses two servlets or two JSP pages on the same server, the session is the same.



Scope Attributes

application
session
page
request

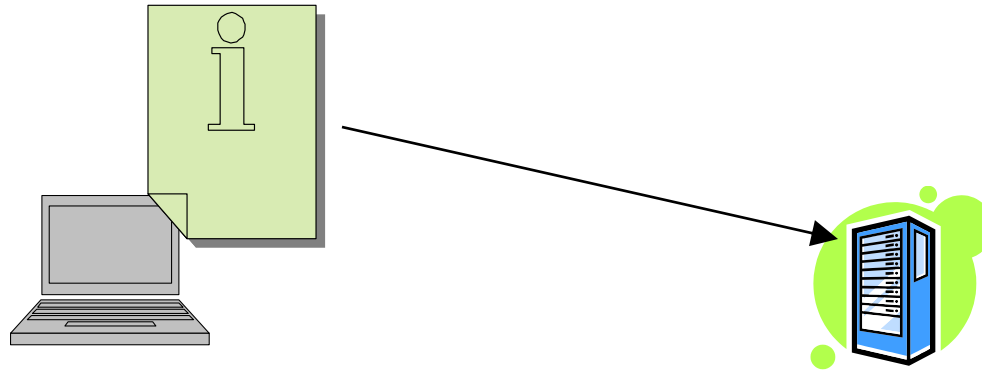
The default scope, which specifies that the object is bound to the page.



Scope Attributes

application
session
page
request

Specifies that the object is bound to the client's request.



How Does JSP Find an Object When

```
<jsp:useBean id="objectName"  
scope="scopeAttribute" class="ClassName" />
```

is processed, the JSP engine first searches for the object of the class with the same id and scope.

If found, the preexisting bean is used; otherwise, a new bean is created.

Another Syntax for Creating a Bean

Here is another syntax for creating a bean using the following statement:

```
<jsp:useBean id="objectName" scope="scopeAttribute"  
            class="ClassName" >
```

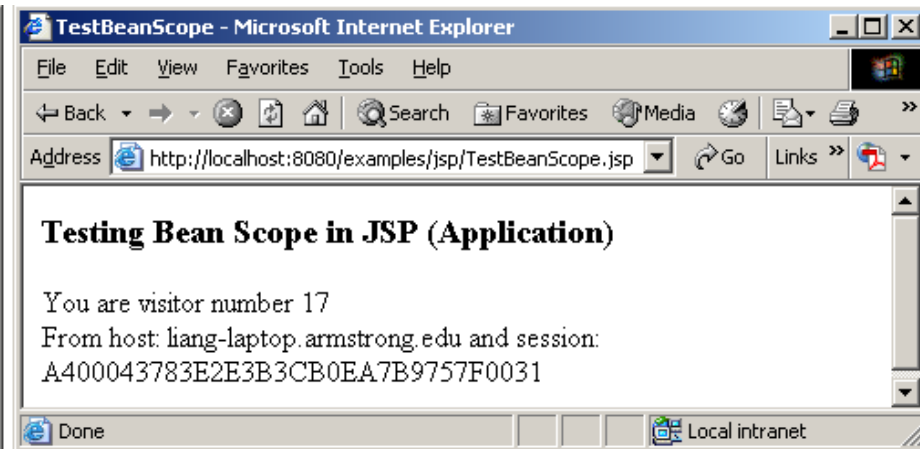
some statements

```
</jsp:useBean>
```

The statements are executed when the bean is created. If the bean with the same id and className already exists, the statements are **not executed**.

Example: Testing Bean Scope

This example creates a JavaBeans component named Count and uses it to count the number of visits to a page.



```
<!-- TestBeanScope.jsp -->
```

```
<%@ page import = "cse305.Count" %>
```

```
<jsp:useBean id="count" scope="application" class="cse305.Count">
```

```
</jsp:useBean>
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>TestBeanScope</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<H3>
```

```
Testing Bean Scope in JSP (Application)
```

```
</H3>
```

```
<% count.increaseCount(); %>
```

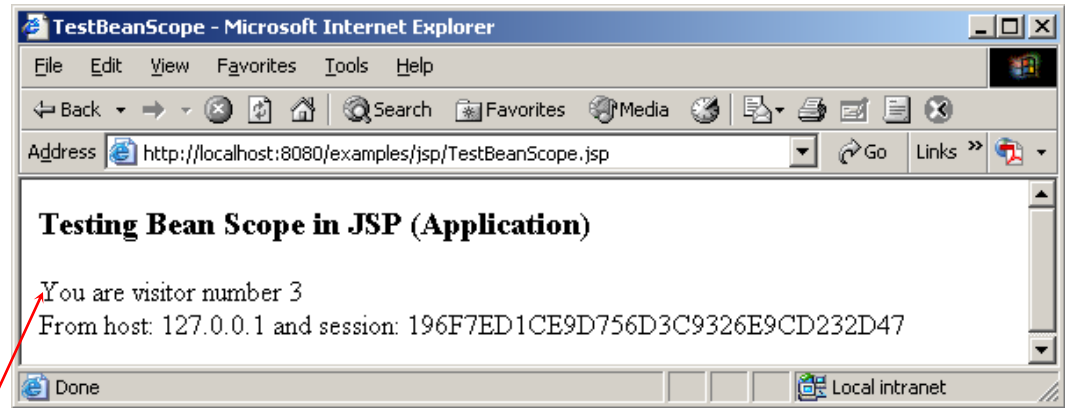
```
You are visitor number <%= count.getCount() %> <br>
```

```
From host: <%= request.getRemoteHost() %>
```

```
and session: <%= session.getId() %>
```

```
</BODY>
```

```
</HTML>
```



```
package cse305;
```

```
public class Count {  
    private int count = 0;
```

```
    /** Return count property */  
    public int getCount() {  
        return count;  
    }
```

```
    /** Increase count */  
    public void increaseCount() {  
        count++;  
    }  
}
```


Getting and Setting Properties

By convention, a JavaBeans component provides the get and set methods for reading and modifying its private properties.

You can get the property in JSP using the following syntax:

```
<jsp:getProperty name="beanId" property="age" />
```

This is equivalent to

```
<%= beanId.getAge() %>
```

Getting and Setting Properties, cont.

You can set the property in JSP using the following syntax:

```
<jsp:setProperty name="beanId" property="age"  
value="30" />
```

This is equivalent to

```
<% beanId.setAge(30); %>
```

Associating Properties with Input Parameters

Often properties are associated with input parameters. Suppose you want to get the value of the input parameter named score and set it to the JavaBeans property named score. You may write the following code:

```
<% double score = Double.parseDouble(  
    request.getParameter("score")); %>
```

```
<jsp:setProperty name="beanId" property="score"  
value="<%= score %>" />
```

Associating Properties with Input Parameters

This is cumbersome. JSP provides a convenient syntax that can be used to simplify it:

```
<jsp:setProperty name="beanId" property="score"  
    param="score" />
```

Instead of using the value attribute, you use the param attribute to name an input parameter.

The value of this parameter is set to the property.

Associating All Properties

Often the bean property and the parameter have the same name.

You can use the following convenient statement to associate all the bean properties in beanId with the parameters that match the property names.

```
<jsp:setProperty name="beanId" property="*" />
```

Example: Computing Loan Using JavaBeans

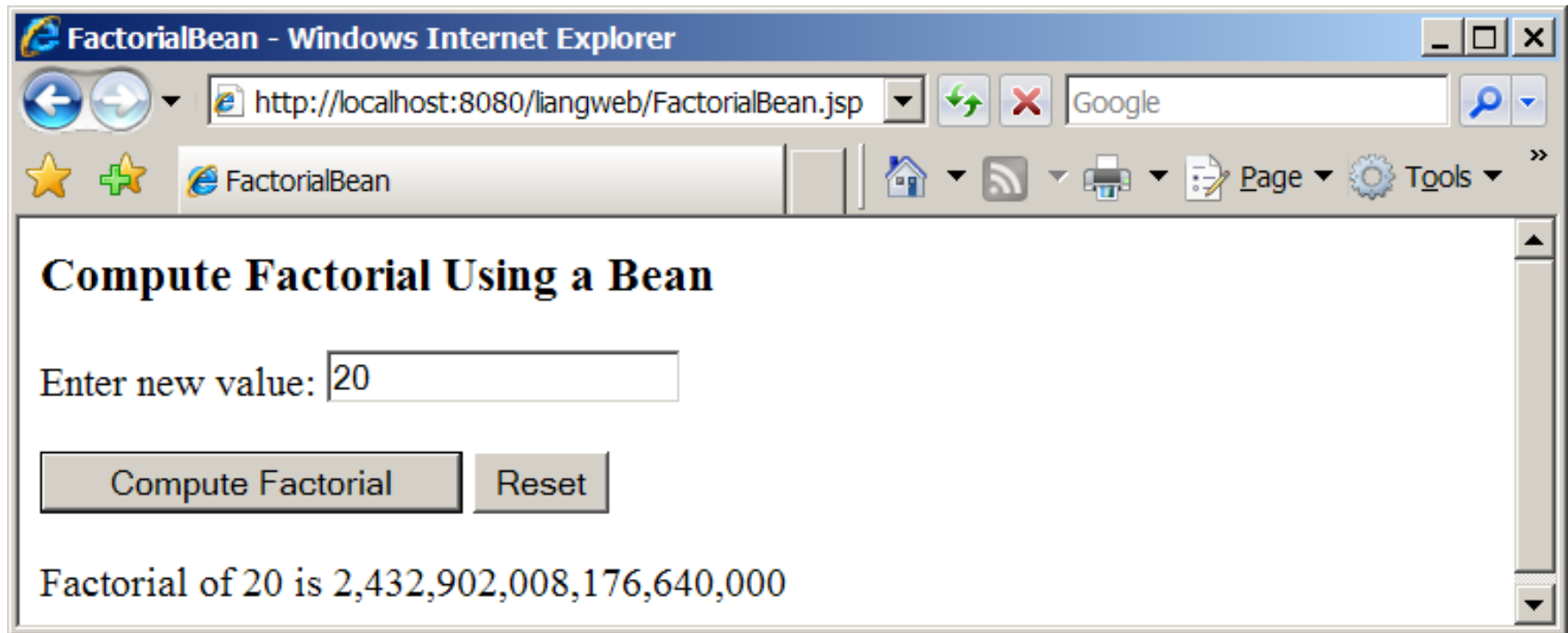
Associate the bean properties with the input parameters:

```
<!-- ComputeLoan.jsp -->
<html>
<head>
<title>ComputeLoan Using the Loan Class</title>
</head>
<body>
<%@ page import = "cse305.Loan" %>
<jsp:useBean id="loan" class="cse305.Loan"></jsp:useBean>
<jsp:setProperty name="loan" property="*" />
Loan Amount: <%= loan.getLoanAmount() %><br>
Annual Interest Rate: <%= loan.getAnnualInterestRate() %><br>
Number of Years: <%= loan.getNumOfYears() %><br>
<b>Monthly Payment: <%= loan.monthlyPayment() %><br>
Total Payment: <%= loan.totalPayment() %><br></b>
</body>
</html>
```

Associating the bean properties with the input parameters.

Example: Computing Factorials Using JavaBeans

Create a JavaBeans component named FactorialBean and use it to compute the factorial of an input number in a JSP page named FactorialBean.jsp.



```
<!-- FactorialBean.jsp -->
<%@ page import = "cse305.FactorialBean" %>
<jsp:useBean id="factorialBeanId" class="cse305.FactorialBean" >
</jsp:useBean>
<jsp:setProperty name="factorialBeanId" property="*" />
```

Associating the bean properties with the input parameters.

```
<HTML>
<HEAD>
<TITLE>
FactorialBean
</TITLE>
</HEAD>
<BODY>
<H3>
Compute Factorial Using a Bean
</H3>
<FORM method="post">
Enter new value: <INPUT NAME="number"><BR><BR>
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Compute Factorial">
<INPUT TYPE="RESET" VALUE="Reset">
<P>Factorial of
<jsp:getProperty name="factorialBeanId" property="number" />
```

Getting number

```
is
<%@ page import="java.text.*" %>
<% NumberFormat format = NumberFormat.getNumberInstance(); %>
<%= format.format(factorialBeanId.getFactorial()) %>
</FORM>
</BODY>
</HTML>
```



```
public class FactorialBean {  
    private int number;  
  
    /** Return number property */  
    public int getNumber() {  
        return number;  
    }  
  
    /** Set number property */  
    public void setNumber(int newValue) {  
        number = newValue;  
    }  
  
    /** Obtain factorial */  
    public long getFactorial() {  
        long factorial = 1;  
        for (int i = 1; i <= number; i++)  
            factorial *= i;  
        return factorial;  
    }  
}
```

DESIGN GUIDE

Mixing a lot of Java code with HTML in a JSP page makes the code difficult to read and to maintain.

You should move the Java code to a .java file as much as you can.

getting NewFactorialBean

```
<!-- NewFactorialBean.jsp -->
<%@ page import = "cse305.NewFactorialBean" %>
<jsp:useBean id = "factorialBeanId"
  class = "cse305.NewFactorialBean" scope = "page" >
</jsp:useBean>
<jsp:setProperty name = "factorialBeanId" property = "*" />
<html>
  <head>
    <title>
      FactorialBean
    </title>
  </head>
  <body>
    <h3>Compute Factorial Using a Bean</h3>
    <form method = "post">
      Enter new value: <input name = "number" /><br /><br />
      <input type = "submit" name = "Submit"
        value = "Compute Factorial" />
      <input type = "reset" value = "Reset" /><br /><br />
      Factorial of
      <jsp:getProperty name = "factorialBeanId"
        property = "number" /> is
      <%= NewFactorialBean.format(factorialBeanId.getFactorial()) %>
    </form>
  </body>
</html>
```

<!-- DisplayTime.jsp -->
<%@page pageEncoding = "GB18030"%>
<%@ page import = "cse305.TimeBean" %>
<jsp:useBean id = "timeBeanId"
class = "cse305.TimeBean" scope = "application" >
</jsp:useBean>
<jsp:setProperty name = "timeBeanId" property = "*" />
<html>
<head>
<title>
Display Time
</title>
</head>
<body>
<h3>Choose locale and time zone</h3>
Current time is
<%=
timeBeanId.currentTimeString(timeBeanId.getLocaleIndex(),
timeBeanId.getTimeZoneIndex()) %>
</body>
</html>

getting
TimeBean

Forwarding Requests from JavaServer Pages

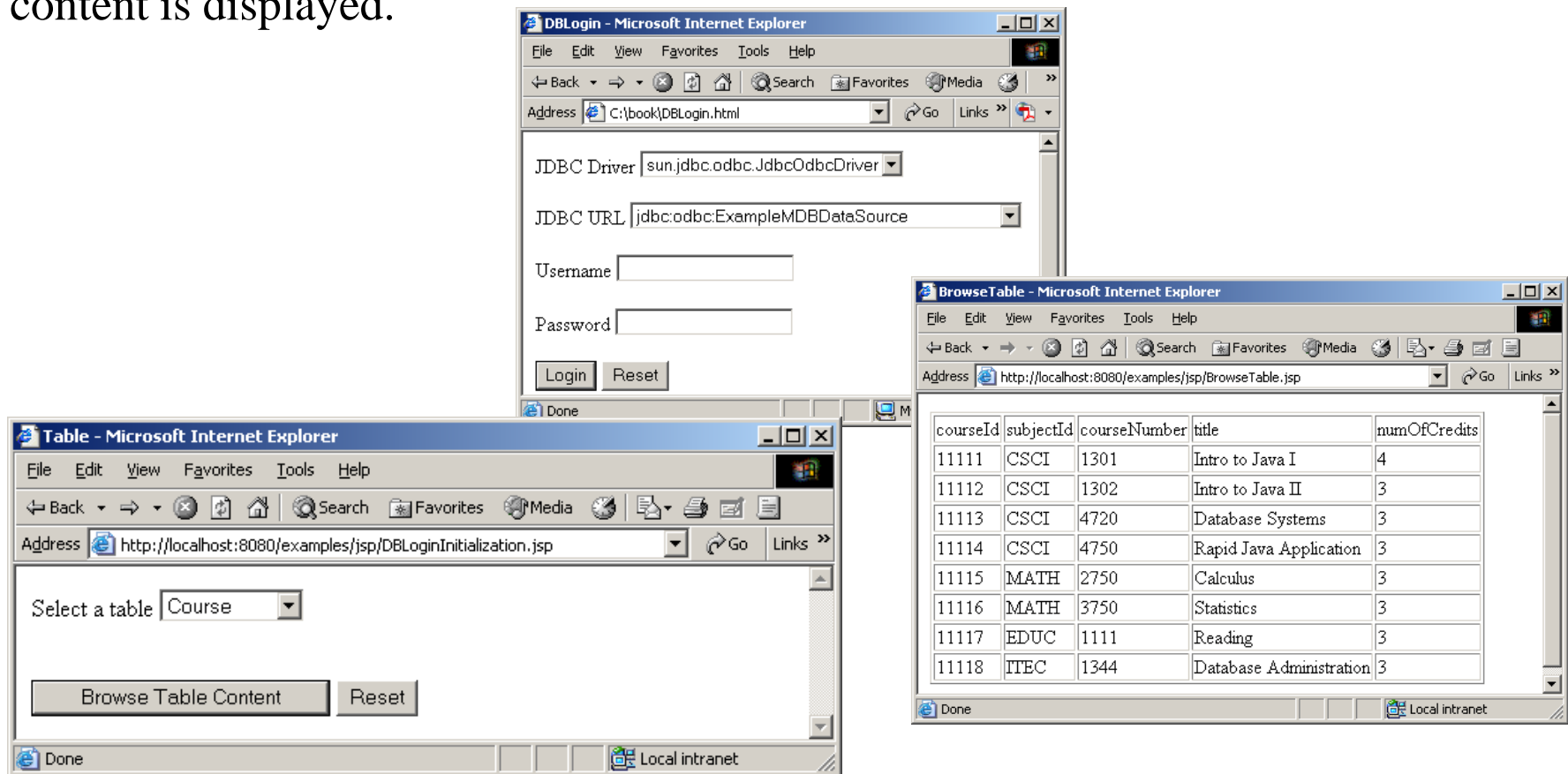
Web applications developed using JSP generally consist of many pages linked together.

JSP provides a forwarding tag in the following syntax that can be used to forward a page to another page.

```
<jsp:forward page="destination" />
```

Example: Browsing Database Tables

This example creates a JSP database application that browses tables. When you start the application, the first page prompts the user to enter the JDBC driver, URL, username, and password for a database. After you login to the database, you can select a table to browse. Upon clicking the Browse Table Content button, the table content is displayed.



```
<!-- DBLogin.html -->
<html>
  <head>
    <title>
      DBLogin
    </title>
  </head>
  <body>
    <form method = "post" action = "/DBLoginInitialization.jsp">
      JDBC Driver
      <select name = "driver" size = "1">
        <option>com.mysql.jdbc.Driver</option>
        <option>oracle.jdbc.driver.OracleDriver</option>
      </select><br /><br />
      JDBC URL
      <select name = "url" size = "1">
        <option>jdbc:mysql://localhost:3306/test</option>
        <option>jdbc:odbc:ExampleMDBDataSource</option>
      </select><br /><br />
      Username <input name = "username" /><br /><br />
      Password <input name = "password" /><br /><br />
      <input type = "submit" name = "Submit" value = "Login" />
      <input type = "reset" value = "Reset" />
    </form>
  </body>
</html>
```

```

package cse305;

import java.sql.*;

public class DBBean {

    private Connection connection = null;
    private String username;
    private String password;
    private String driver;
    private String url;

    /** Initialize database connection */
    public void initializeJdbc() {
        try {
            System.out.println("Driver is " + driver);
            Class.forName(driver);
            // Connect to the sample database
            connection = DriverManager.getConnection(url, username, password);
        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /** Get tables in the database */
    public String[] getTables() {
        String[] tables = null;
        try {
            DatabaseMetaData dbMetaData = connection.getMetaData();
            ResultSet rsTables = dbMetaData.getTables(null, null, null, new String[] {"TABLE"});

```



```

        int size = 0;
        while (rsTables.next()) size++;
        rsTables = dbMetaData.getTables(null, null, null, new String[] {"TABLE"});
        tables = new String[size];
        int i = 0;
        while (rsTables.next())
            tables[i++] = rsTables.getString("TABLE_NAME");
    }
    catch (Exception ex) {
        ex.printStackTrace();
    }
    return tables;
}

/** Return connection property */
public Connection getConnection() {
    return connection;
}

public void setUsername(String newUsername) {
    username = newUsername;
}

public String getUsername() {
    return username;
}

public void setPassword(String newPassword) {
    password = newPassword;
} ...
}

```

```
<!-- DBLoginInitialization.jsp -->
<%@ page import = "cse305.DBBean" %>
<jsp:useBean id = "dBBeanId" scope = "session" class = "cse305.DBBean">
</jsp:useBean>
<jsp:setProperty name = "dBBeanId" property = "*" />
<html>
  <head>
    <title>DBLoginInitialization</title>
  </head>
  <body>

    <!-- Connect to the database --%>
    <% dBBeanId.initializeJdbc(); %>

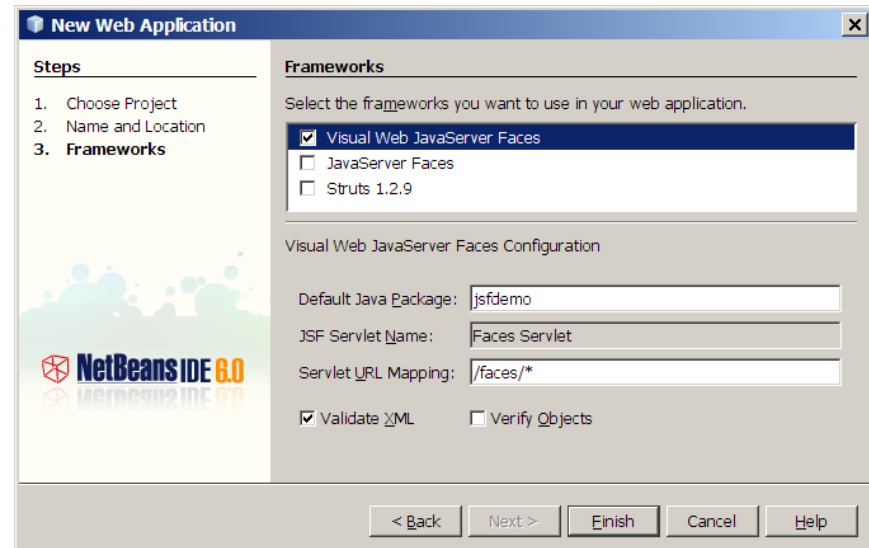
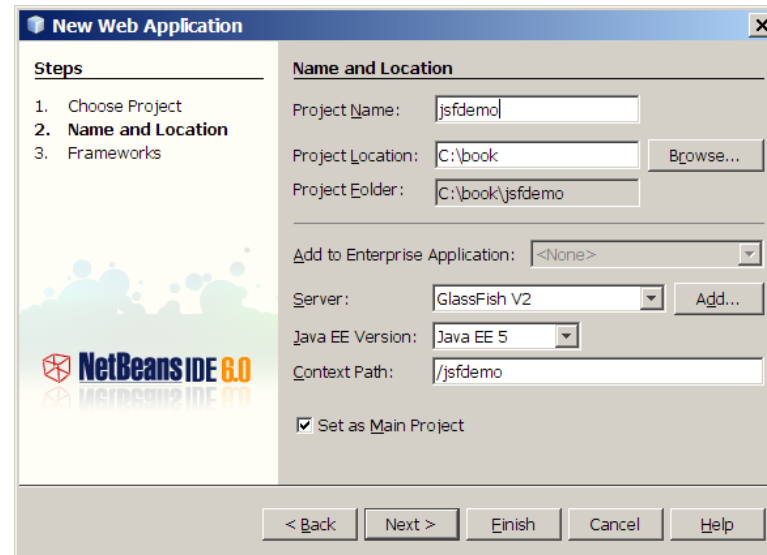
    <% if (dBBeanId.getConnection() == null) { %>
      Error: Login failed. Try again.
    <% }
    else {%>
      <jsp:forward page = "Table.jsp" />
    <% } %>
  </body>
</html>
```

Java Server Faces (JSF)

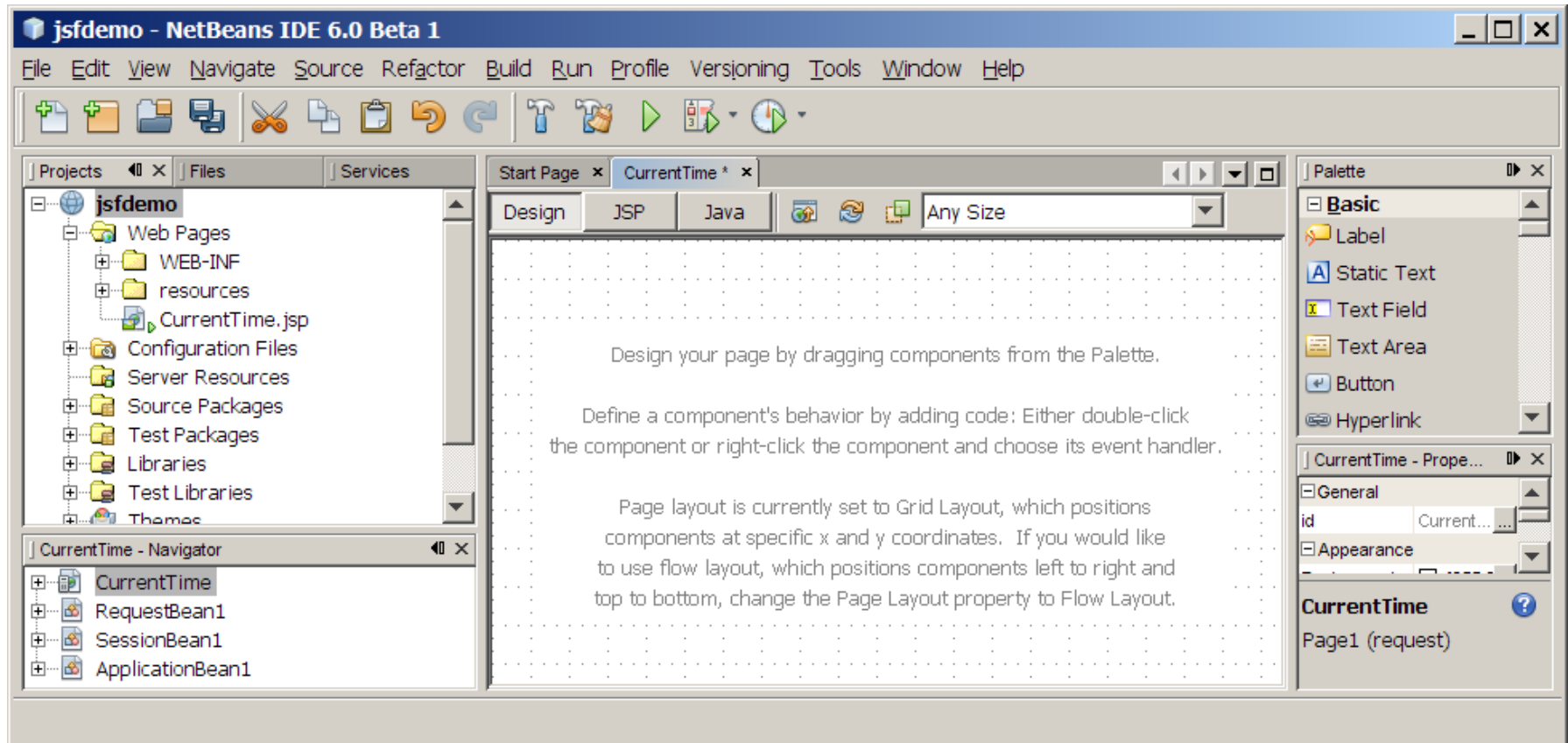
- JSF supports visual Web development.
- You can create a Web user interface using a tool without writing any code.
- JSF completely separates Web UI from Java code so the application developed using JSF is easy to debug and maintain.

Visual Web Design Using NetBeans

Create a Web project with Visual Web JavaServer Faces.



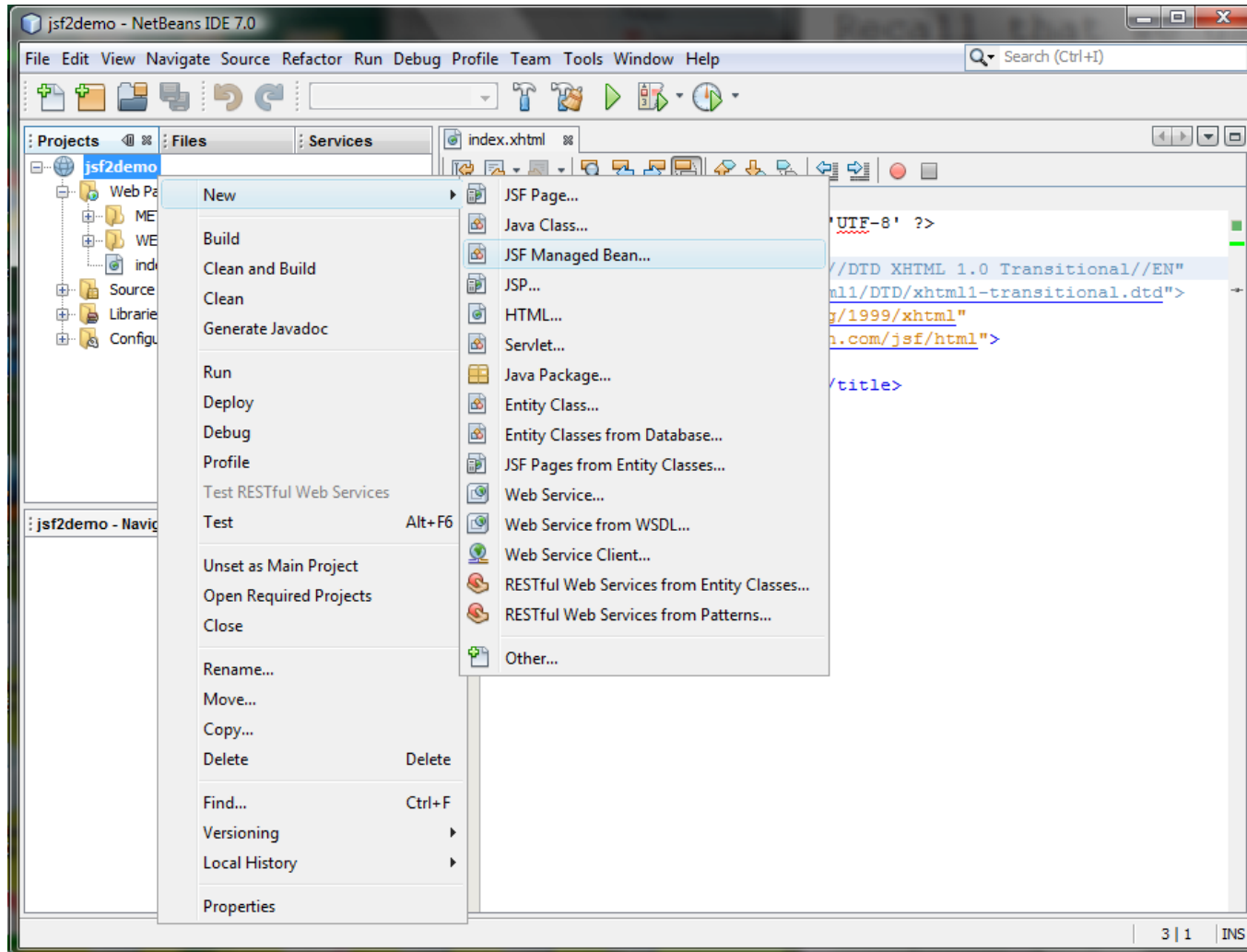
Creating UI in the Design Pane



A Basic JSF Page

```
<?xml version='1.0' encoding='UTF-8' ?>
<!-- index.xhtml -->
<!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://java.sun.com/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        Hello from Facelets
    </h:body>
</html>
```

Managed JavaBeans for JSF



Managed JavaBeans for JSF

```
package jsf2demo;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
@ManagedBean
@RequestScoped
public class TimeBean {
    public String getTime() {
        return new java.util.Date().toString();
    }
}

#{expression}

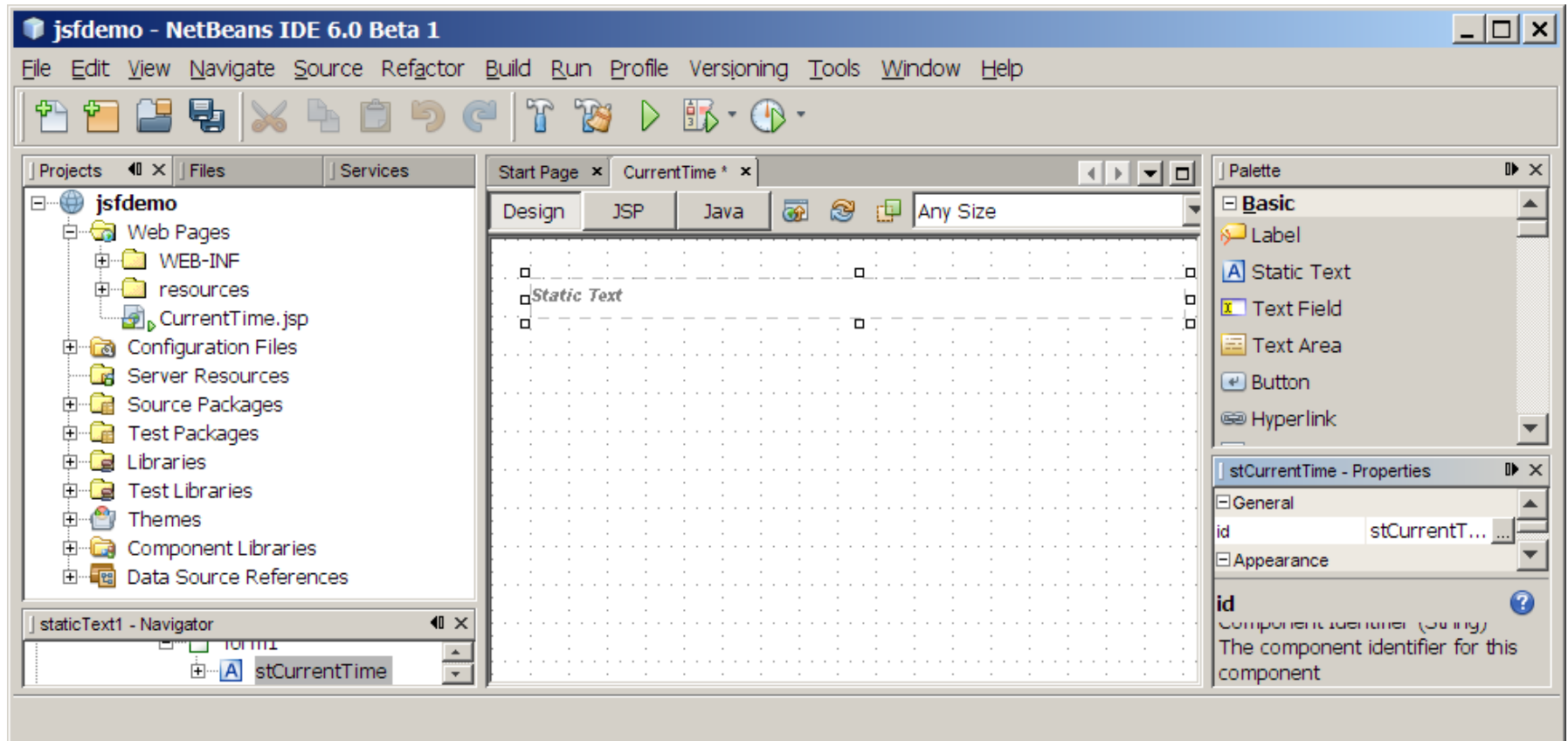
#{timeBean.time}
#{timeBean.getTime()}
```


CurrentTime.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://java.sun.com/jsf/html">
  <h:head>
    <title>Display Current Time</title>
    <meta http-equiv="refresh" content="60" />
  </h:head>
  <h:body>
    The current time is #{timeBean.time}
  </h:body>
</html>
```

Creating UI in the Design Pane

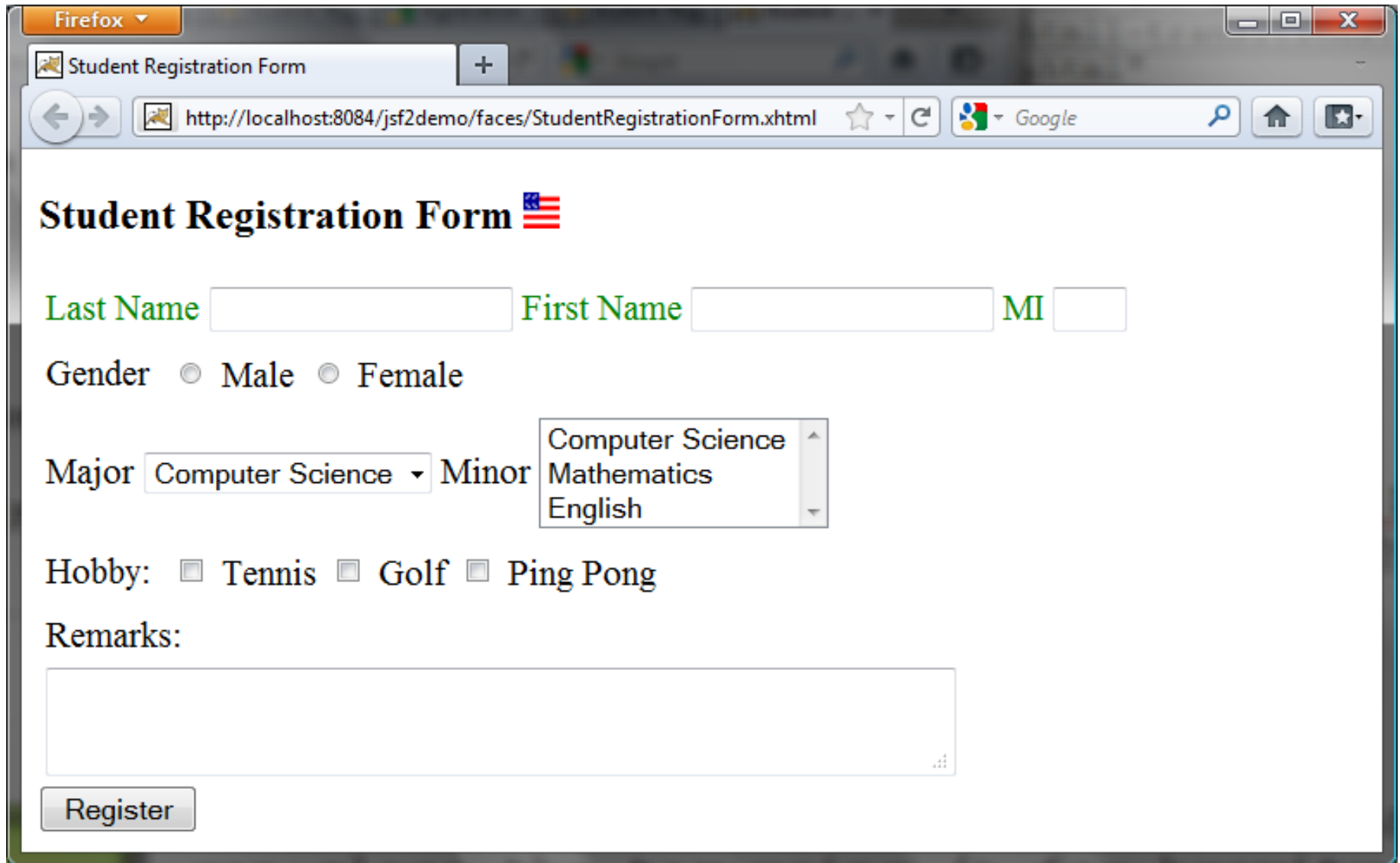
Drop a Static Text and set its properties



JSF GUI Components

JSF Tag	Description
<code>h:form</code>	inserts an XHTML form into a page.
<code>h:panelGroup</code>	similar to a Java flow layout container.
<code>h:panelGrid</code>	similar to a Java grid layout container.
<code>h:inputText</code>	displays a textbox for entering input.
<code>h:outputText</code>	displays a textbox for displaying output.
<code>h:inputTextArea</code>	displays a textarea for entering input.
<code>h:inputSecret</code>	displays a textbox for entering password.
<code>h:outputLabel</code>	displays a label.
<code>h:outputLink</code>	displays a hypertext link.
<code>h:selectOneMenu</code>	displays a combo box for selecting one item.
<code>h:selectOneRadio</code>	displays a set of radio button.
<code>h:selectBooleanCheckbox</code>	displays a checkbox.
<code>h:selectOneListbox</code>	displays a list for selecting one item.
<code>h:selectManyListbox</code>	displays a list for selecting multiple items.
<code>f:selectItem</code>	specifies an item in an <code>h:selectOneMenu</code> , <code>h:selectOneRadio</code> , or <code>h:selectManyListbox</code> .
<code>h:message</code>	displays a message for validating input.
<code>h:dataTable</code>	displays a data table.
<code>h:column</code>	specifies a column in a data table.
<code>h:graphicImage</code>	displays an image.


JSF UI Containers



The screenshot shows a Firefox browser window with a single tab titled "Student Registration Form". The address bar displays the URL "http://localhost:8084/jsf2demo/faces/StudentRegistrationForm.xhtml". The page content includes a title "Student Registration Form" with a US flag icon. Below the title are input fields for "Last Name", "First Name", and "MI". A "Gender" section has radio buttons for "Male" and "Female". The "Major" is set to "Computer Science" in a dropdown menu, and the "Minor" is shown in a separate dropdown menu with options "Computer Science", "Mathematics", and "English". The "Hobby" section has checkboxes for "Tennis", "Golf", and "Ping Pong". A "Remarks:" label is followed by a large text area. At the bottom left is a "Register" button.

Firefox ▾

Student Registration Form +

← →  http://localhost:8084/jsf2demo/faces/StudentRegistrationForm.xhtml ☆ ↺  Google 🔍 🏠 ☆

Student Registration Form

Last Name First Name MI

Gender ☐ Male ☐ Female

Major Minor

Computer Science

Mathematics

English

Hobby: ☐ Tennis ☐ Golf ☐ Ping Pong

Remarks:

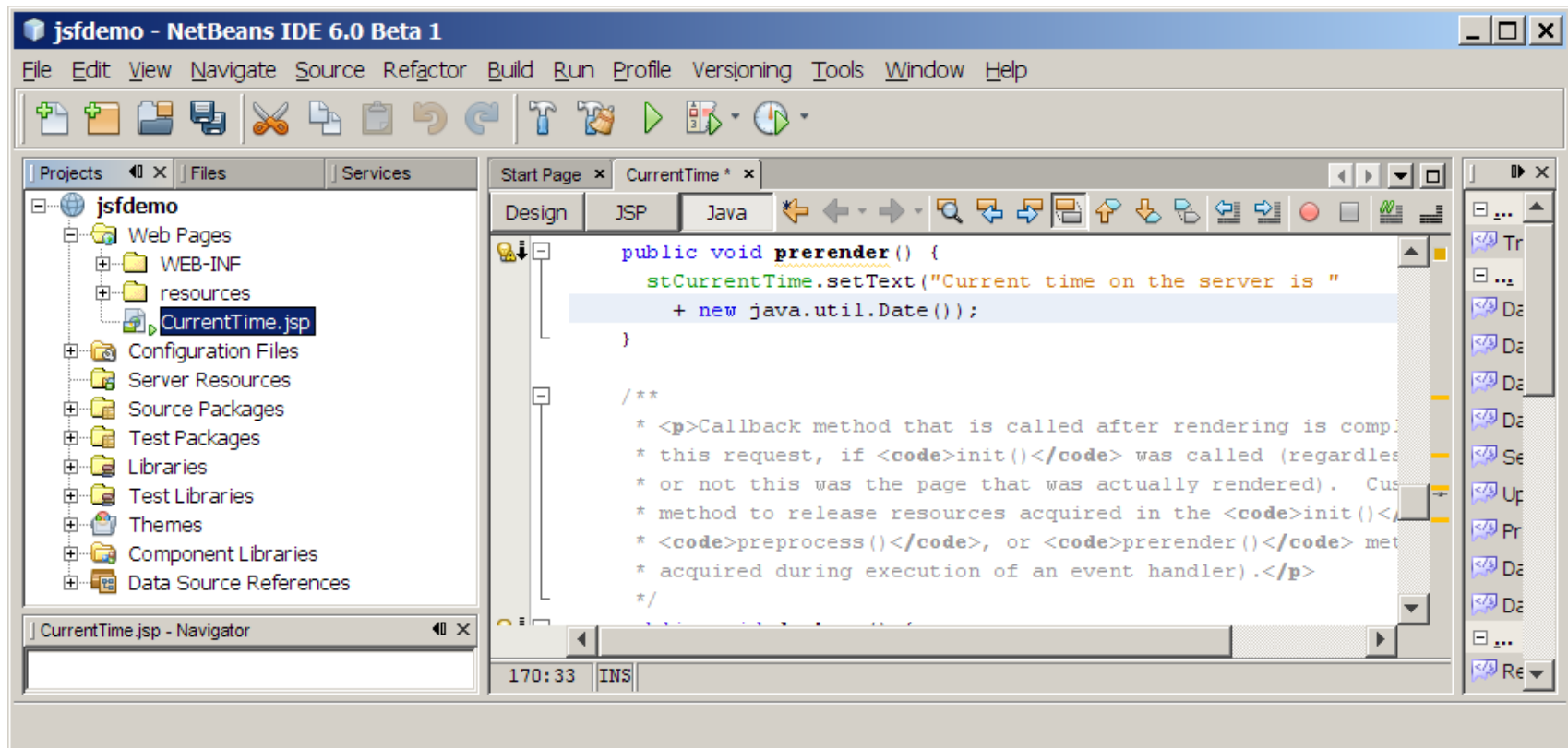
Register

Validating Input

JSF Tag	Description
<code>f:validateLength</code>	validates the length of the input.
<code>f:validateDoubleRange</code>	validates whether numeric input falls within acceptable range of double values.
<code>f:validateLongRange</code>	validates whether numeric input falls within acceptable range of long values.
<code>f:validateRequired</code>	validates whether a field is not empty.
<code>f:validateRegex</code>	validates whether the input matches a regular expression.
<code>f:validateBean</code>	invokes a custom method in a bean to perform custom validation.

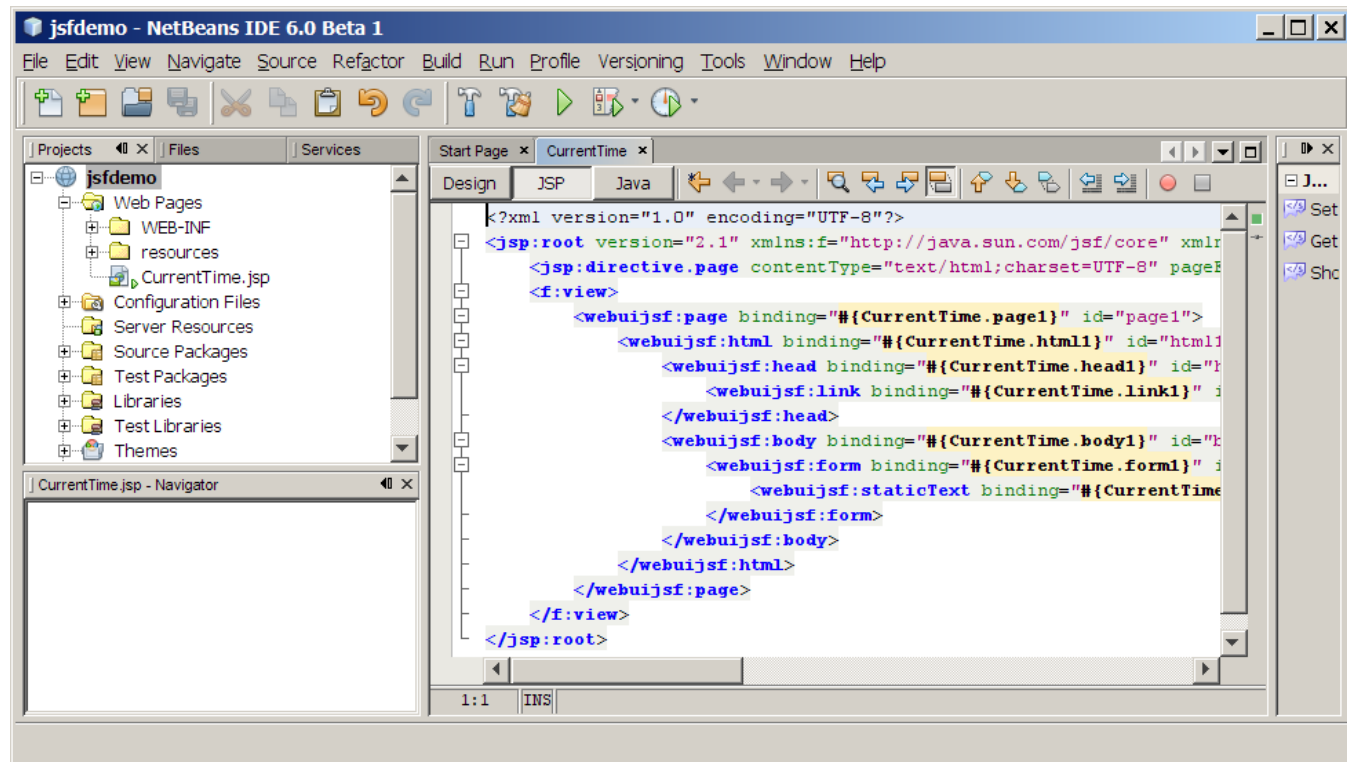
Creating UI in the Design Pane

Write the code in the Java tab.



Examining the JSP File

Click the JSP tab to see the JSP file. Whenever you add, remove, or change the UI components in the Design pane, the contents in the JSP are also updated. It is possible to modify the JSP file directly, but it is not recommended for the new users. Modifying the JSP file mistakenly could corrupt the entire project. You can completely ignore the JSP file when using this tool.



JSF UI Components

http://localhost:8080/jsfdemo/faces/Registrations.jsp - Windows Internet Explorer

http://localhost:8080/jsfdemo/faces/Registrations.jsp

Student Registration Form

Last Name: MI: First Name:

Gender: ☒ Male ☐ Female

Major: Minor:

Hobby: ☒ Tennis ☒ Golf ☐ Ping Pong

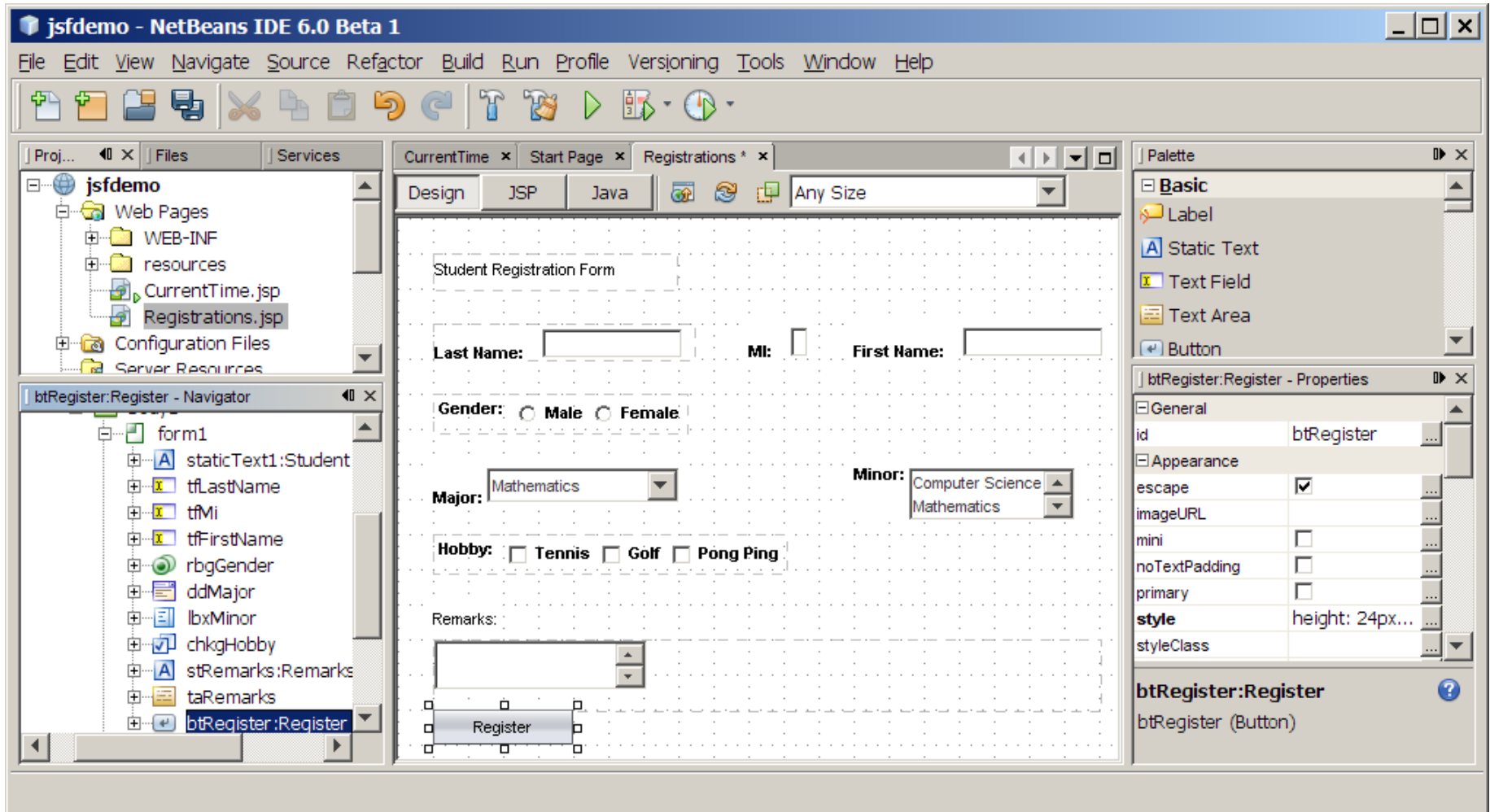
Remarks:

http://localhost:8080/jsfdemo/faces/Registrations.jsp - Windows Internet Explorer

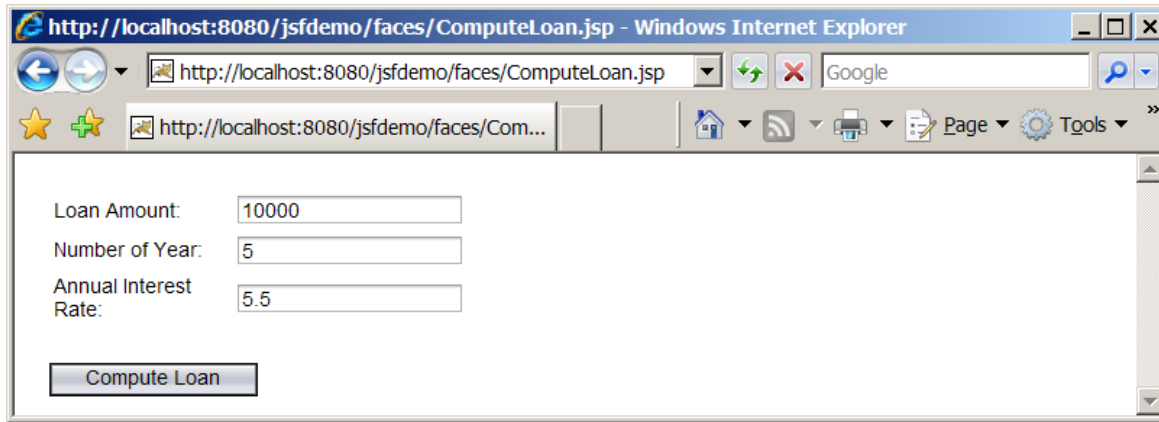
http://localhost:8080/jsfdemo/faces/Registrations.jsp

Last name is Smith
MI is T
First name is John
Selected gender is Male
Selected major is Computer Science
Selected minors are Mathematics
Selected hobbies are TennisGolf
Remarks are No remarks

JSF UI Components



JSF UI Containers



http://localhost:8080/jsfdemo/faces/ComputeLoan.jsp - Windows Internet Explorer

http://localhost:8080/jsfdemo/faces/ComputeLoan.jsp

Google

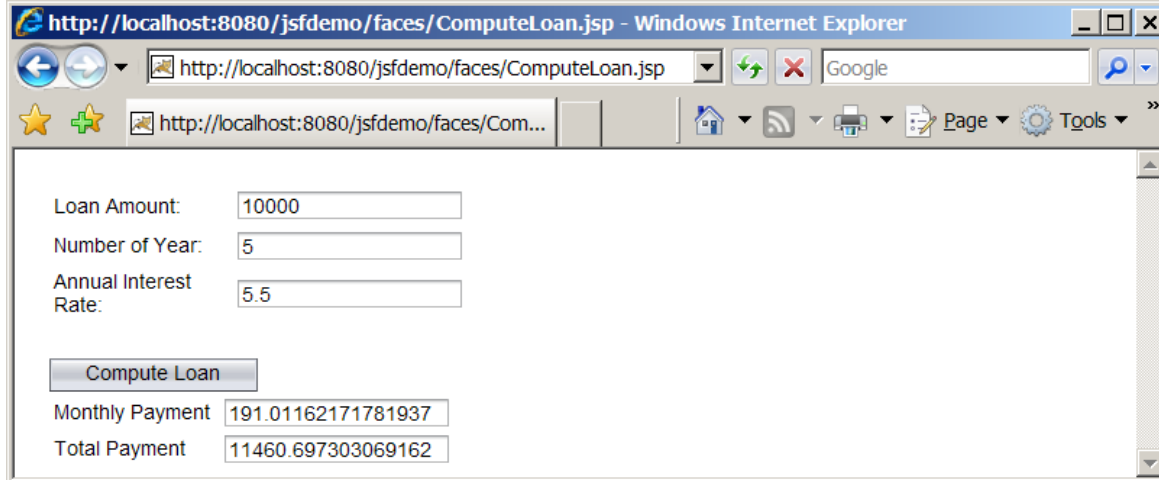
http://localhost:8080/jsfdemo/faces/Com...

Loan Amount: 10000

Number of Year: 5

Annual Interest Rate: 5.5

Compute Loan



http://localhost:8080/jsfdemo/faces/ComputeLoan.jsp - Windows Internet Explorer

http://localhost:8080/jsfdemo/faces/ComputeLoan.jsp

Google

http://localhost:8080/jsfdemo/faces/Com...

Loan Amount: 10000

Number of Year: 5

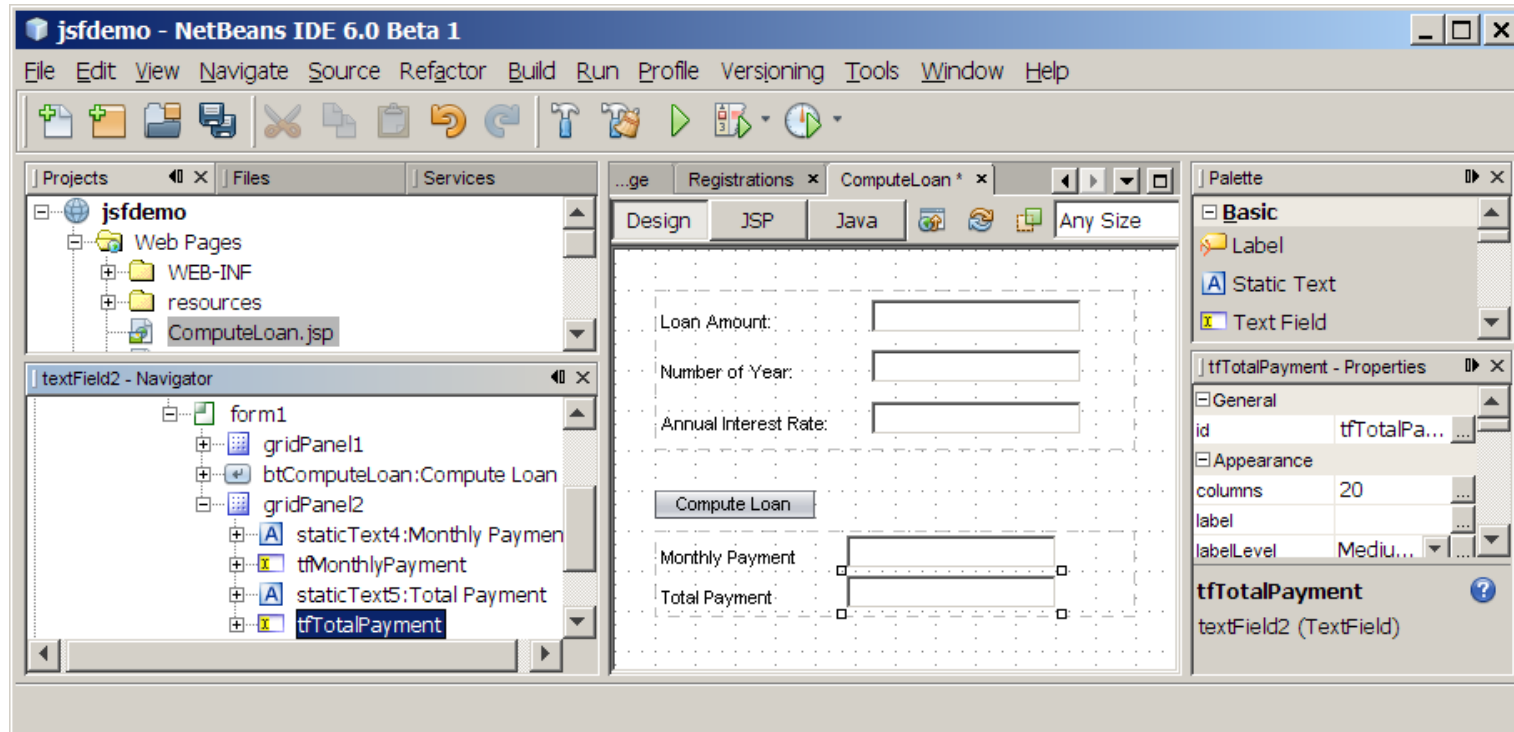
Annual Interest Rate: 5.5

Compute Loan

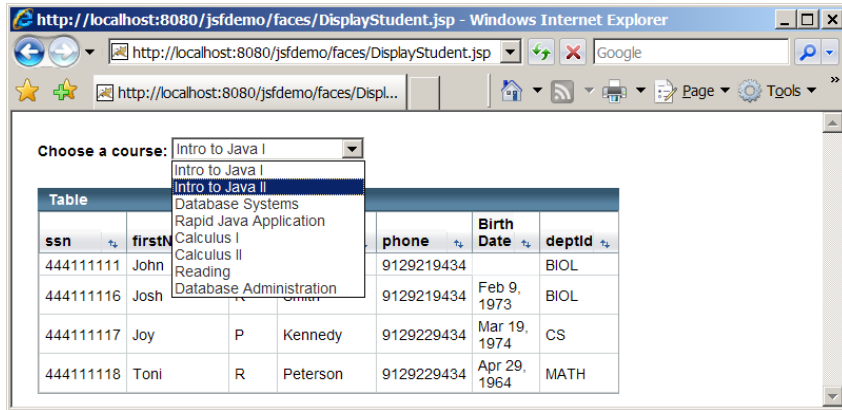
Monthly Payment 191.01162171781937

Total Payment 11460.697303069162

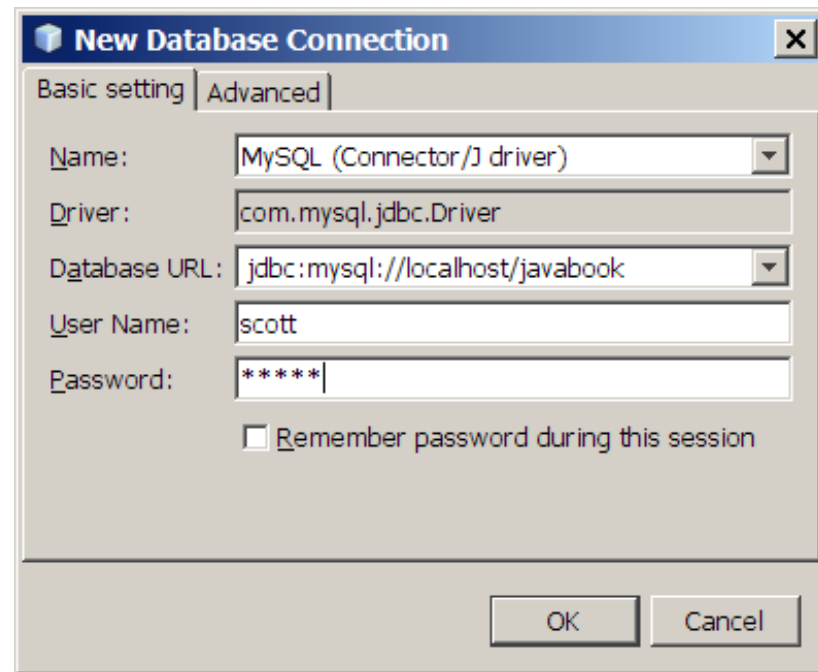
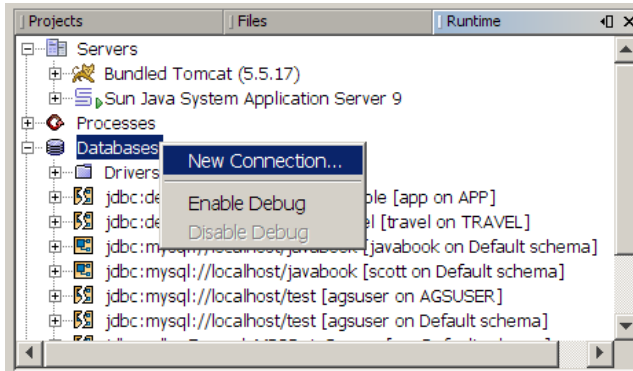
JSF UI Containers



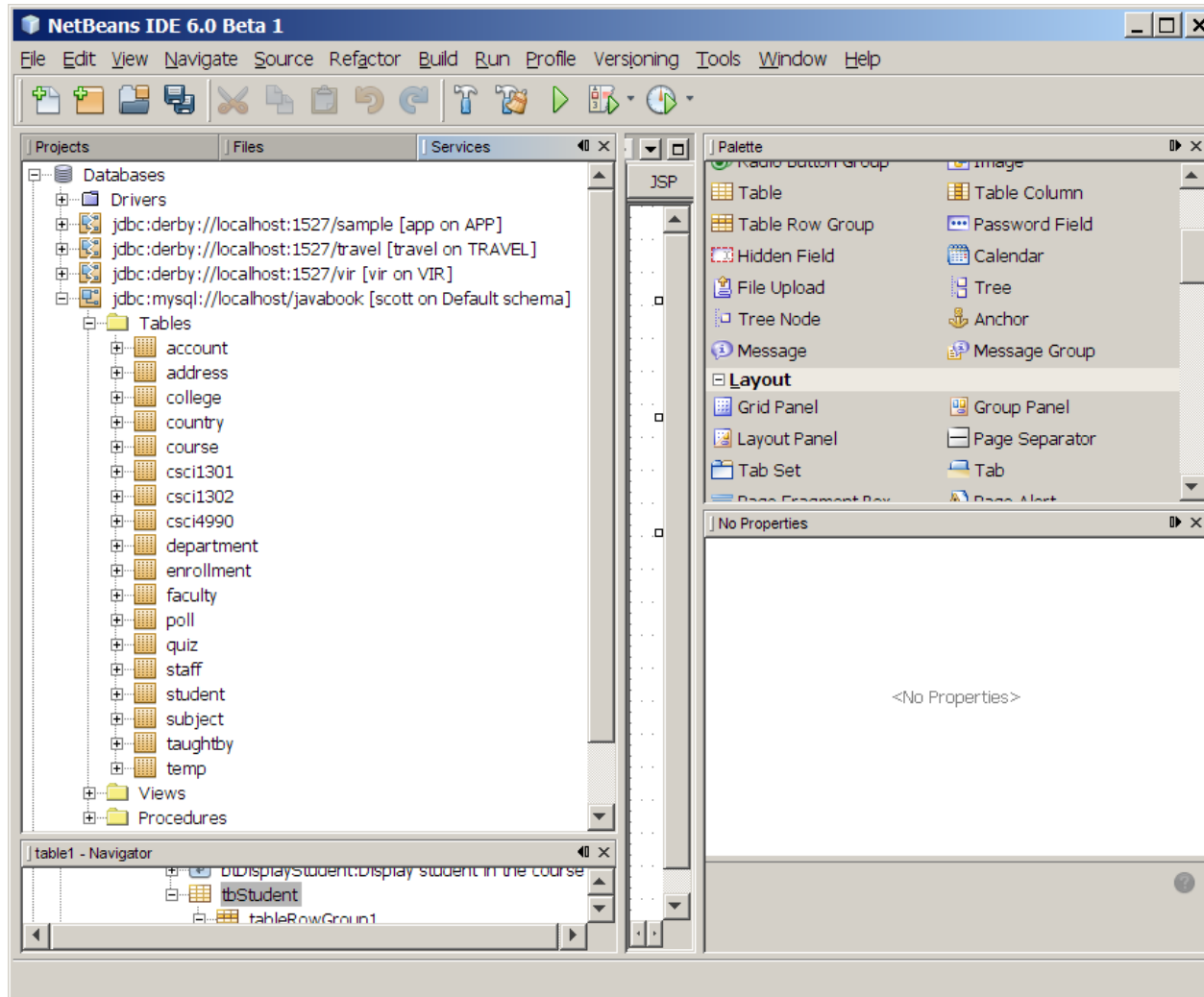
Binding Data with UI Components



Creating a New Database Connection



Creating a New Database Connection



Designing UI

jsfdemo - NetBeans IDE 6.0 Beta 1

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

Pro... x Files Services

jsfdemo

- Web Pages
- WEB-INF
- resources

table1 - Navigator

- DisplayStudent
 - page1
 - html1
 - head1
 - body1
 - form1
 - ddCourse
 - tbStudent

Design JSP Java Any Size

Choose a course: Item 1

column1	column2	column3
abc	abc	abc
abc	abc	abc
abc	abc	abc

tbStudent - Properties

General

id tbStudent

Appearance

augmentTitle

tbStudent

table1 (Table)

Bind to Data - dropDown1

Current Items property setting

#[DisplayStudent.courseDataProvider.options['course.courseId,course.title']]

Bind to Data Provider Bind to an Object

Choose a Data Provider to bind to dropDown1:

courseDataProvider (DisplayStudent)

Value field:

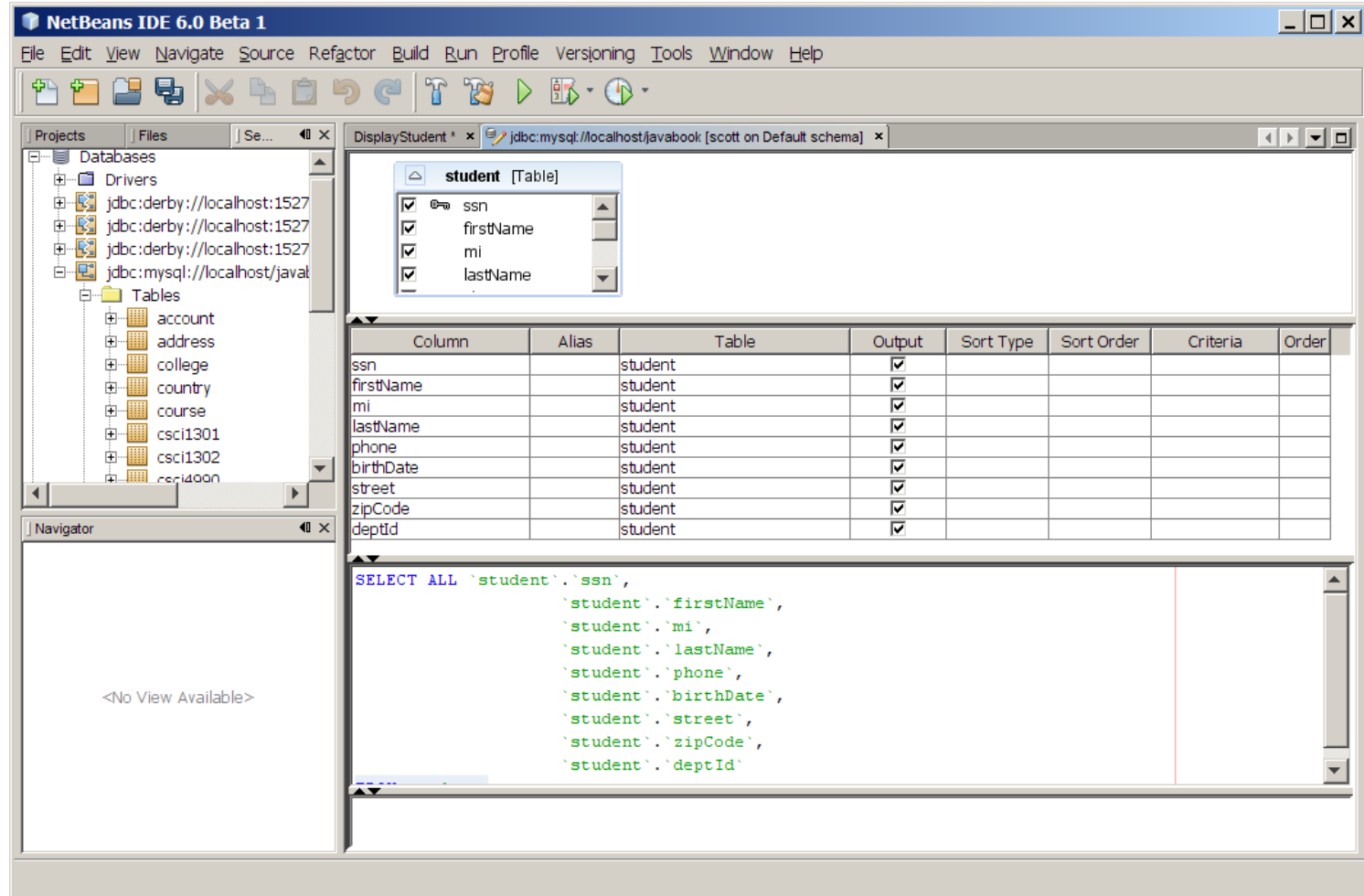
- <none>
- course.courseId String
- course.subjectId String
- course.courseNumber Integer
- course.title String
- course.numOfCredits Integer

Display field:

- <use value field>
- course.courseId String
- course.subjectId String
- course.courseNumber Integer
- course.title String
- course.numOfCredits Integer

OK Cancel Apply Help

Modifying Query



Modifying Query

NetBeans IDE 6.0 Beta 1

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

DisplayStudent * x jdbc:mysql://localhost/javabook [scott on Default schema] x

student [Table]

- ☒ ssn
- ☒ firstName
- ☒ mi
- ☒ lastName

enrollment [Table]

- ☐ ssn
- ☐ courseId
- ☐ dateRegistered
- ☐ grade

course [Table]

- ☐ courseId
- ☐ subjectId
- ☐ courseNumber
- ☐ title

Column	Alias	Table	Output	Sort Type	Sort Order	Criteria	Order
ssn		student	<input checked="" type="checkbox"/>				
firstName		student	<input checked="" type="checkbox"/>				
mi		student	<input checked="" type="checkbox"/>				
lastName		student	<input checked="" type="checkbox"/>				
phone		student	<input checked="" type="checkbox"/>				
birthDate		student	<input checked="" type="checkbox"/>				
street		student	<input checked="" type="checkbox"/>				

```
student.birthDate,
student.street,
student.zipCode,
student.deptId
FROM student, enrollment, course
WHERE course.courseId = ?
      AND student.ssn = enrollment.ssn
      AND enrollment.courseId = course.courseId
```

Changing Table Layout

Table Layout - tbStudent

Columns | Options

Get Data From: **studentDataProvider** (DisplayStudent)

Available		Selected	
student.street	>	student.ssn	Up
student.zipCode	<	student.firstName	Down
	<<	student.mi	
	>>	student.lastName	
		student.phone	
		student.birthDate	New
		student.deptId	

Column Details

Header Text: birthDate

Footer Text:

Component Type: Static Text

Value Expression: #{currentRow.value['student.birthDate']}

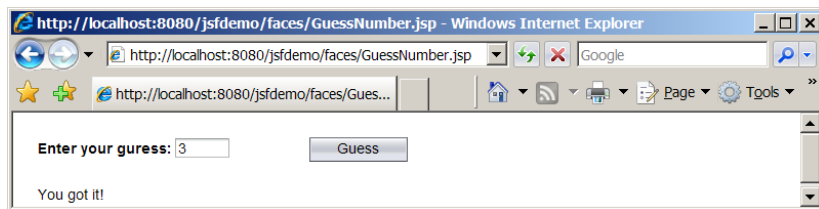
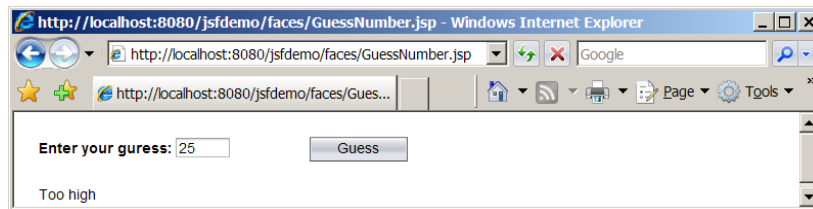
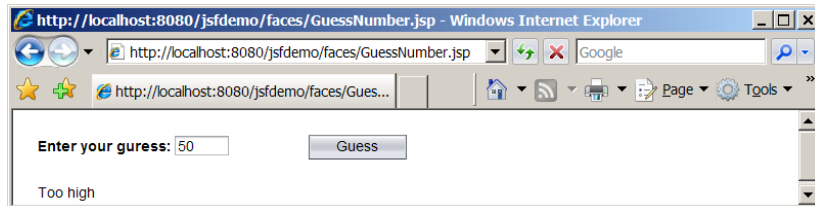
Width:

Horizontal Align: Left Vertical Align: <not set>

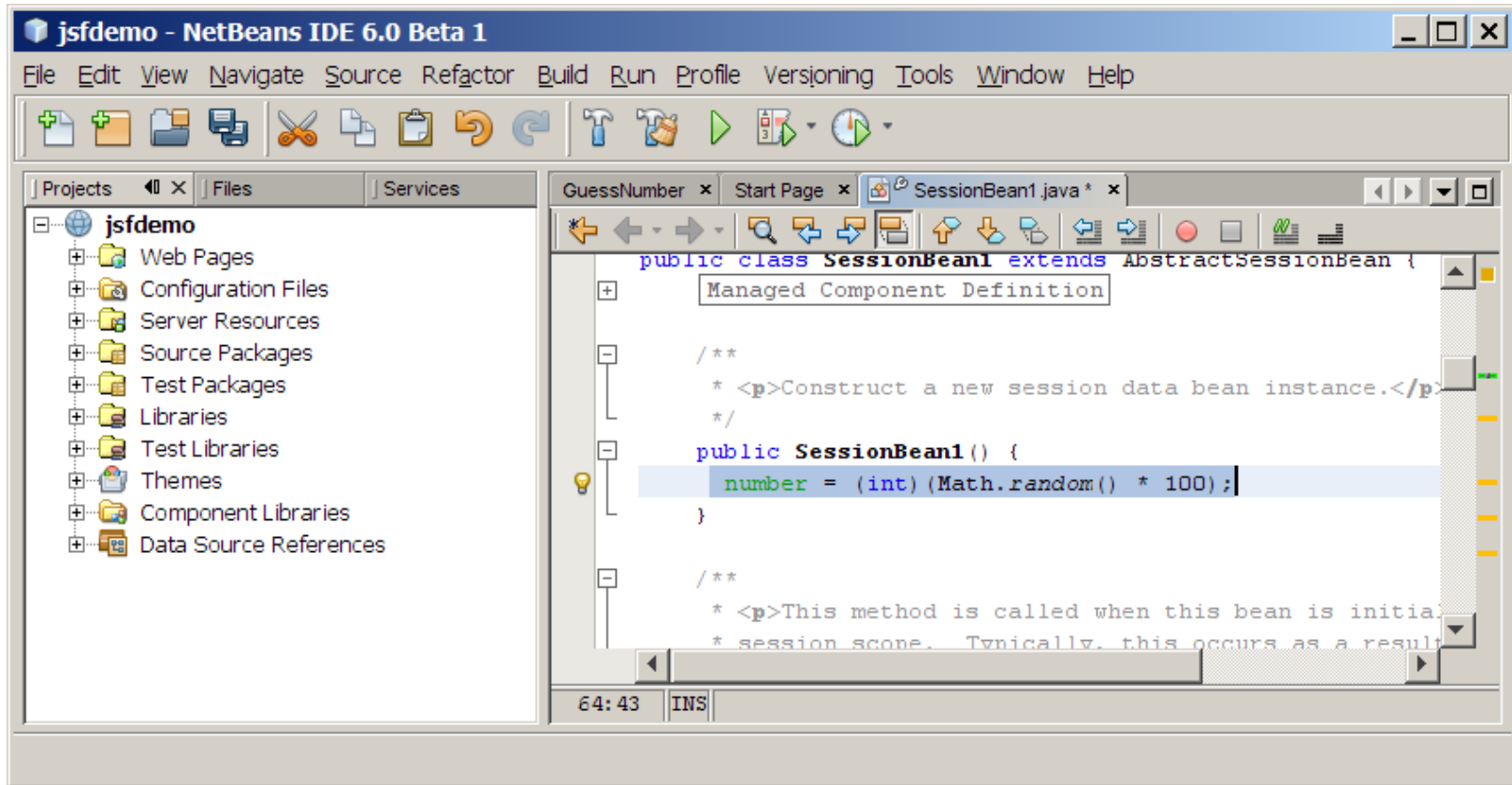
☒ Sortable

OK Cancel Apply Help

Session Tracking



Session Bean



Validating Input

http://localhost:8080/jsfdemo/faces/ValidateForm.jsp - Windows Internet Explorer

http://localhost:8080/jsfdemo/faces/ValidateForm.jsp

Name:

SSN:

Age:

Height:

Submit

form1:tfName: Validation Error: Value is required.
Enter a valid SSN, e.g., 123-44-5678
form1:tfAge: Validation Error: Value is required.
form1:tfHeight: Validation Error: Value is required.

