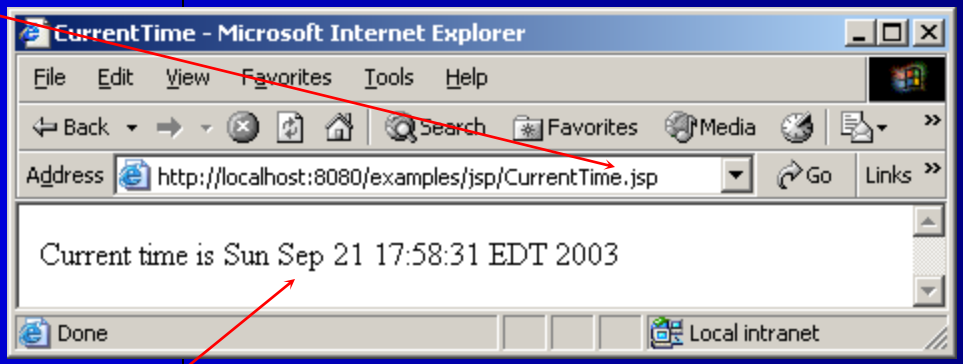
A Simple JSP

```
<!-- CurrentTime.jsp -->
<HTML>
<HEAD>
<TITLE>
CurrentTime
</TITLE>
</HEAD>
<BODY>
Current time is <%= new java.util.Date() %>
</BODY>
</HTML>
```



A Simple JSP

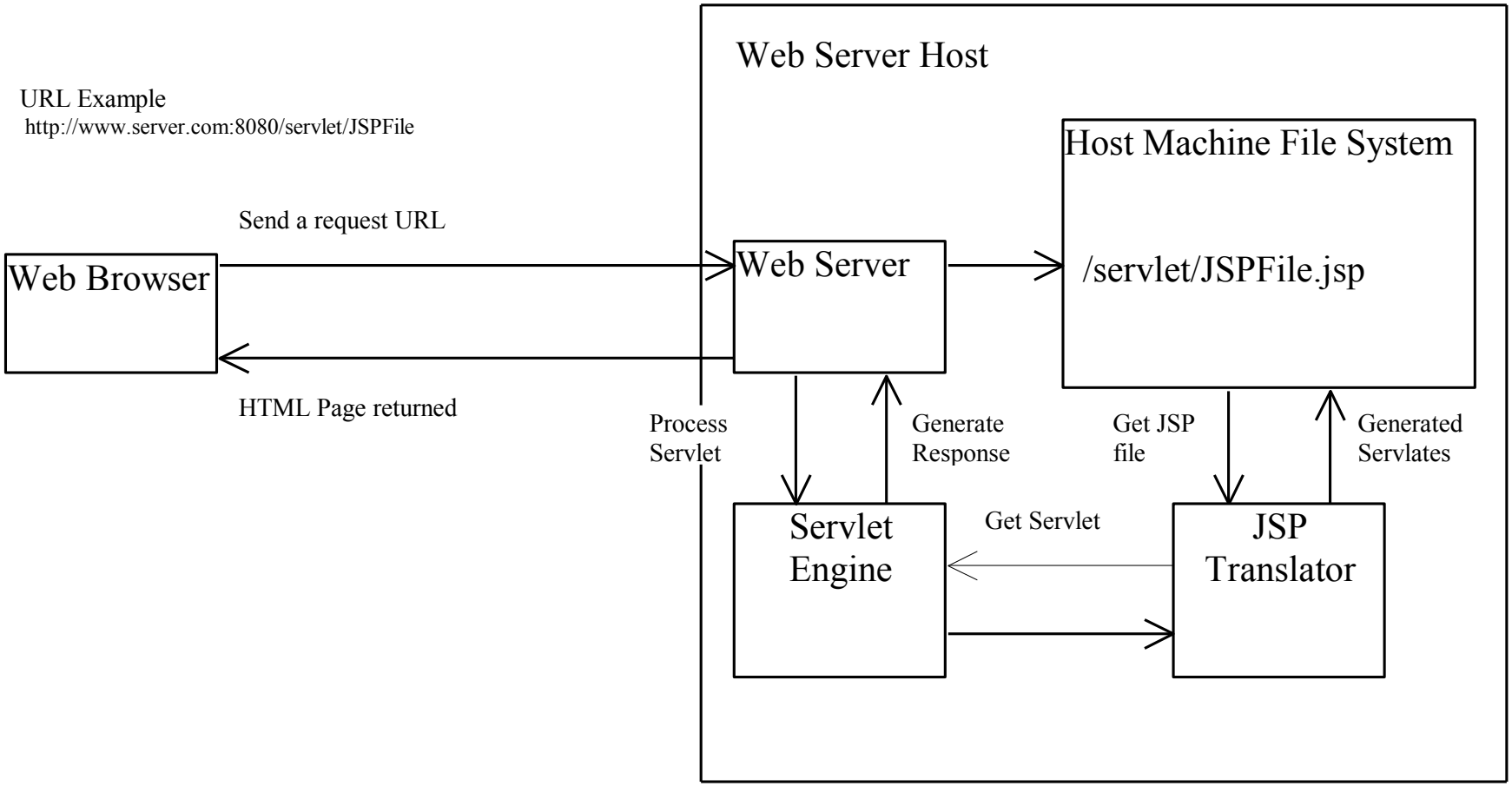
- Create a text file, name it currenttime.jsp
- You must store the file in the following :

tomcat/webapps/examples/jsp/currenttime.jsp

- Use the following URL:

http://localhost:8080/examples/jsp/currenttime.jsp

How Is a JSP Processed?



JSP Constructs

There are three types of scripting constructs. They are *expressions*, *scriptlets*, and *declarations*.

expression

scriptlet

declaration

A JSP expression is used to insert a Java expression directly into the output. It has the following form:

<%= Java-expression %>

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. 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 . They are *expressions*, *scriptlets*, and *declarations*

expression

scriptlet

declaration

A JSP declaration is for declaring methods. It has the following form:

<%! Java method or field declaration %>

JSP Comment

HTML comments have the following form:

<!-- HTML Comment -->

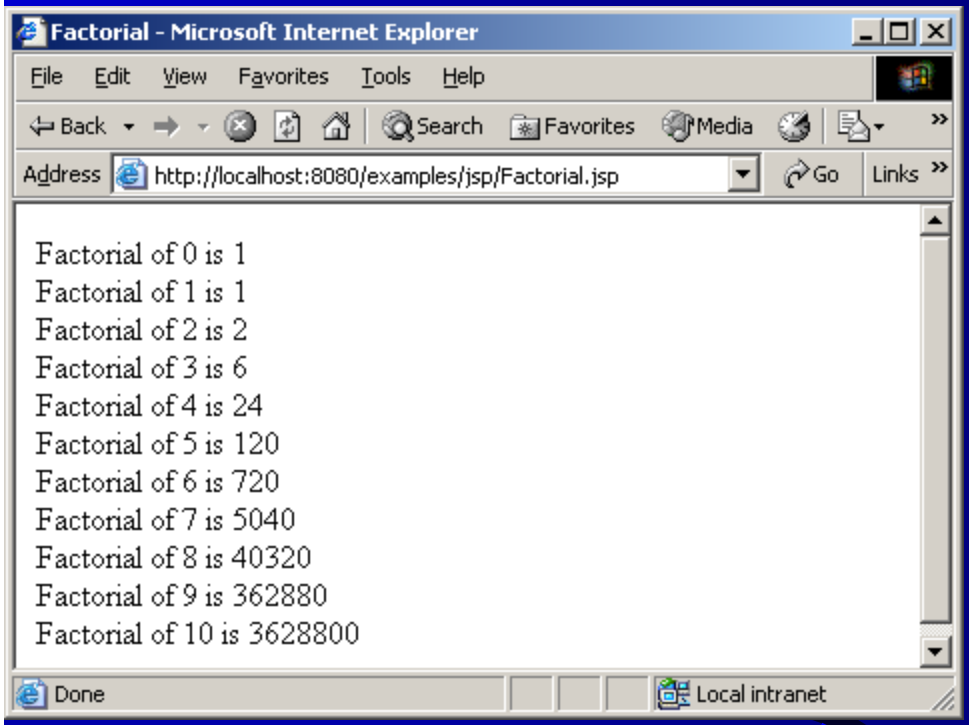
Example 27.1 Computing Factorials

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

<%  for (int i = 0; i <= 10; i++) { %>
Factorial of <%= i %> is
<%= computeFactorial(i) %> <br>
<%  } %>

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

</BODY>
</HTML>
```

JSP scriptlet

JSP expression

JSP declaration



JSP Predefined Variables

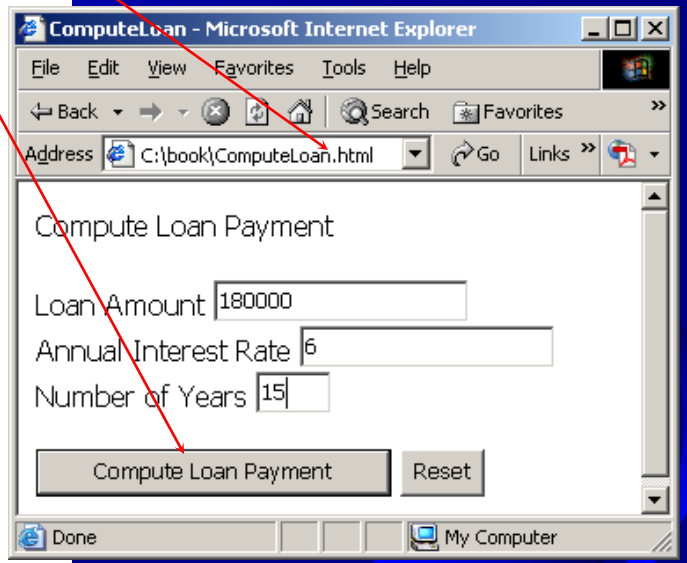
You can use variables in JSP.

- request
- response
- out
- session
- application
- config
- pageconte
  xt
- page

Example 27.2 Computing Loan

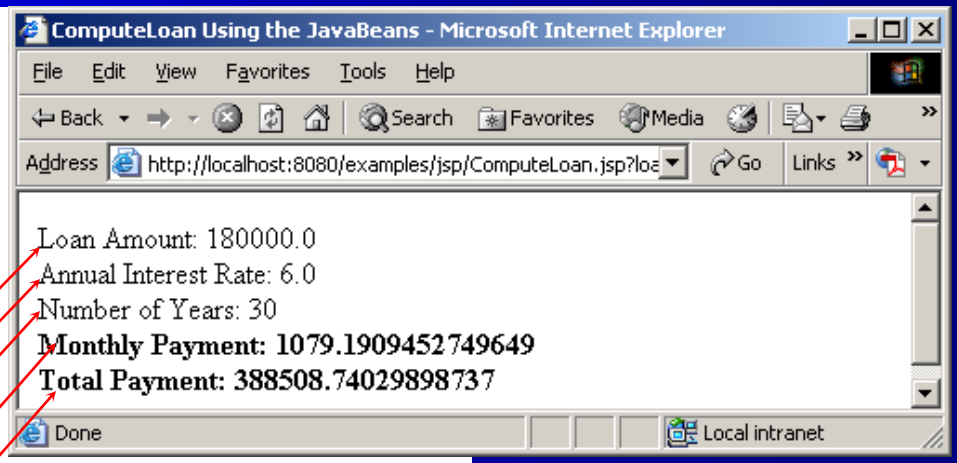
Write an HTML page that prompts the user to enter loan amount, annual interest rate, and number of years. Clicking the Compute Loan Payment button invokes a JSP to compute and display the monthly and total loan payment.

```
<!-- ComputeLoan.html -->
<html>
<head>
<title>ComputeLoan</title>
</head>
<body>
Compute Loan Payment

<form method="get"

action="http://localhost:8080/examples/jsp/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

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"
              attribute2="value2"
              ...
               attributen="vlauen" %>
```

Three JSP Directives

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

1. page
2. includ
   e
3. tablib

Attributes for *page* Directives

- import
- contentType
- session
- buffer
- autoFlush
- isThreadSafe
- errorPage
- isErrorPage

```
<!-- ComputeLoan.jsp -->
<html>
<head>
<title>ComputeLoan Using the Loan Class</title>
</head>
<body>
<%@ page import = "chapter27.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>
```

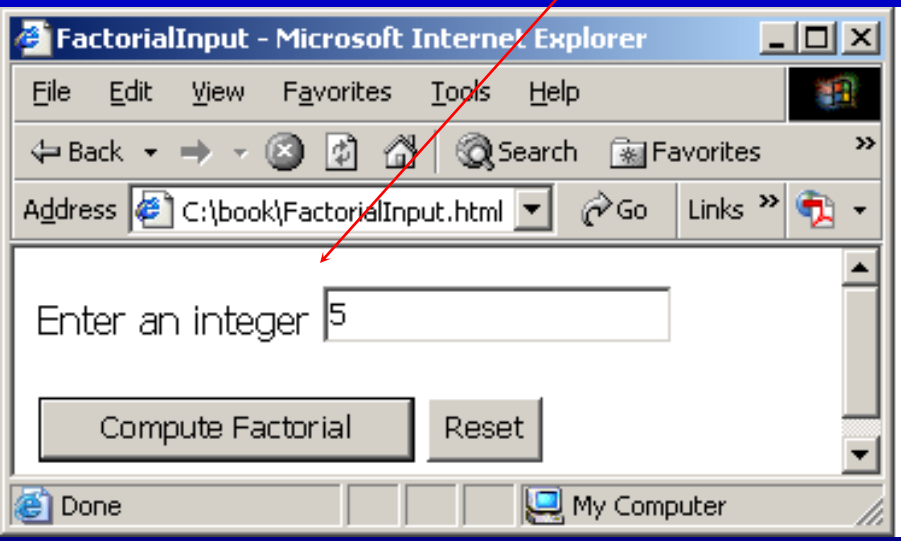
Example 27.3 Computing Loan Using the Loan Class

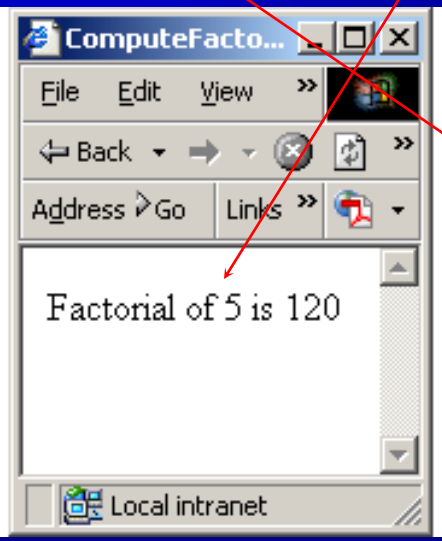
Use the Loan class to simplify Example 27.2. You can create an object of Loan class and use its monthlyPayment() and totalPayment() methods to compute the monthly payment and total payment.

Import a class. The class must be placed in a package (e.g. package chapter27.

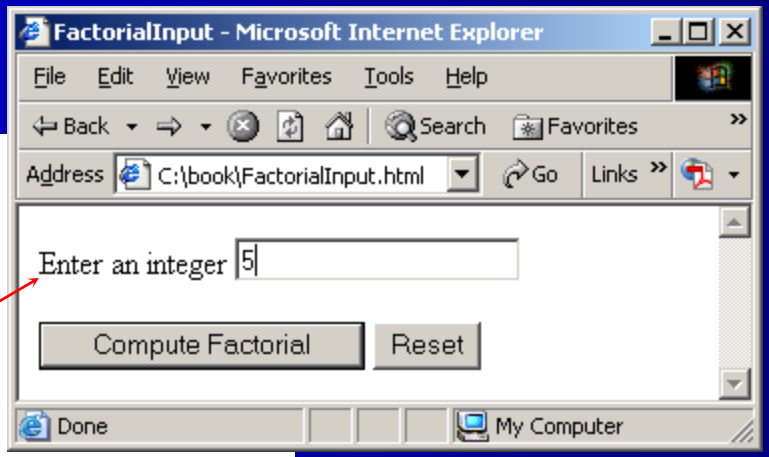
Example 27.4 Using Error Pages

This example prompts the user to enter an integer and displays the factorial for the integer. If a noninteger value is entered by mistake, an error page is displayed.



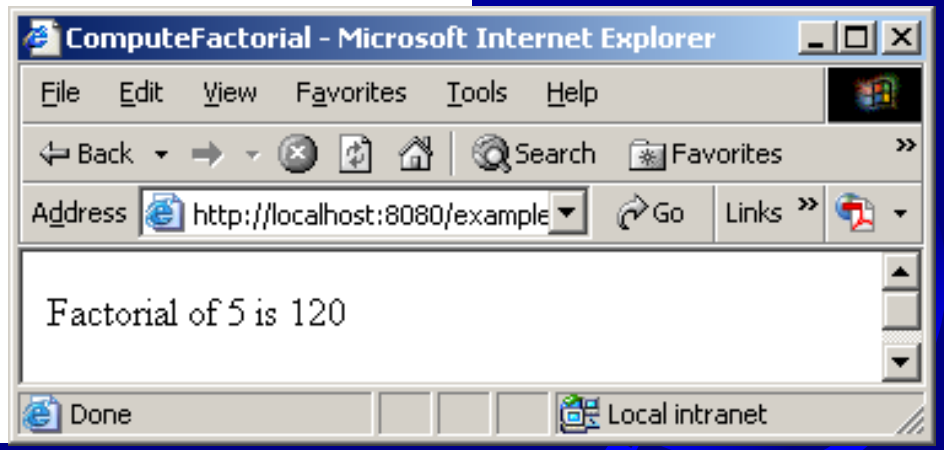


```
<!-- FactorialInput.html -->
<HTML>
<HEAD>
<TITLE>
FactorialInput
</TITLE>
</HEAD>
<BODY>
<FORM method="post"
  action="http://localhost:8080/examples/jsp/ComputeFactorial.jsp">
  Enter an integer <INPUT NAME="number"><BR><BR>
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Compute Factorial">
<INPUT TYPE="RESET" VALUE="Reset">
</FORM>
</BODY>
</HTML>
```



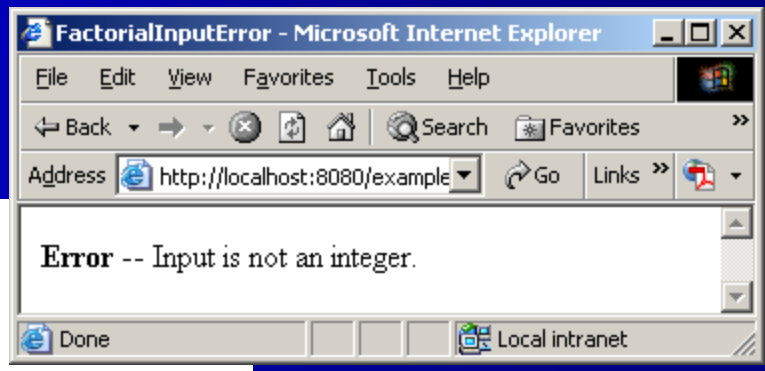
```
<!-- ComputeFactorial.jsp -->
<HTML>
<HEAD>
<TITLE>
ComputeFactorial
</TITLE>
</HEAD>
<BODY>

<%@ page errorPage = "FactorialInputError.jsp" %>
<% int number = Integer.parseInt(request.getParameter("number")); %>
Factorial of <%= number %> is
<%= computeFactorial(number) %> <p>
<%! private long computeFactorial(int n) {
      if (n == 0)
        return 1;
      else
        return n * computeFactorial(n - 1);
    }
%>
</BODY>
</HTML>
```

Error page



```
<!-- FactorialInputError.jsp -->
<HTML>
<HEAD>
<TITLE>
FactorialInputError
</TITLE>
</HEAD>
<BODY>
<%@ page isErrorPage = "true" %>

<b>Error</b> -- Input is not an integer.

</BODY>
</HTML>
```

Indicate it is error page



JavaBeans Component in JSP

Recall that 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. Recall that 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

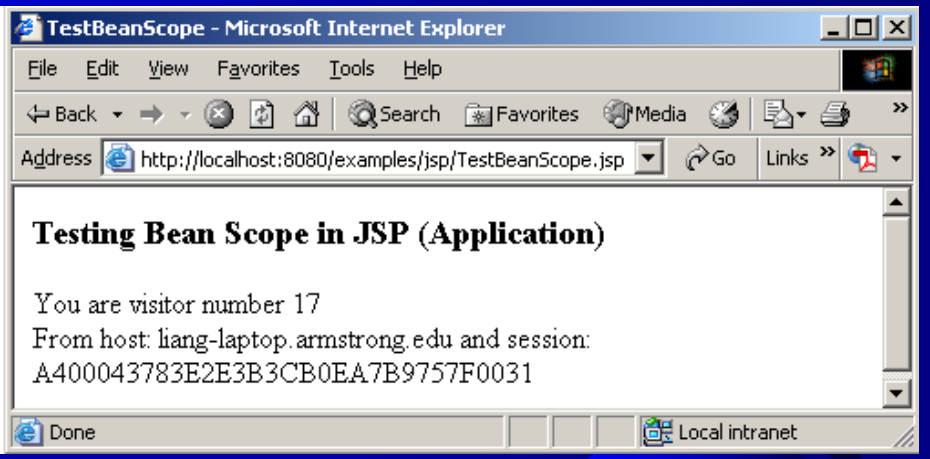
Here is another syntax for creating a bean using the following statement:

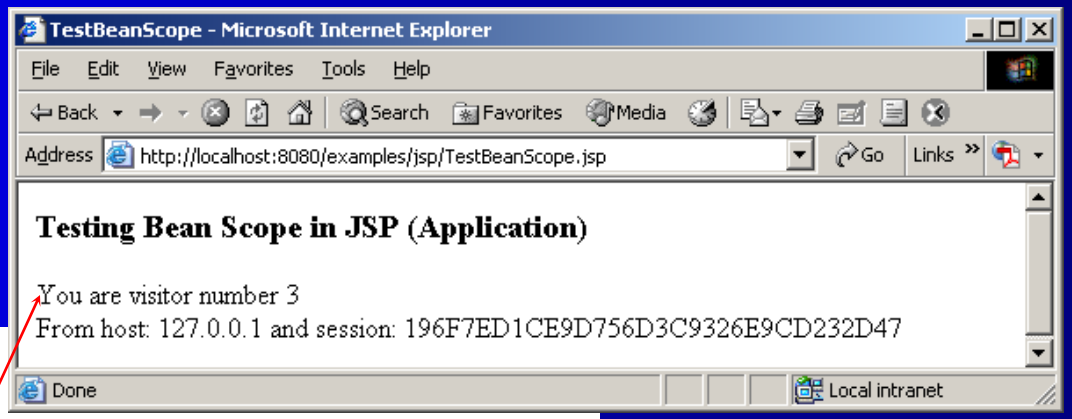
```
<jsp:useBean id="objectName"
    scope="scopeAttribute“ class="ClassName" >

 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 27.5 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 = "chapter27.Count" %>
<jsp:useBean id="count" scope="application" class="chapter27.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 chapter27;

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="sample" />
```

This is equivalent to

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

Getting and Setting Properties, cont.

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

```
<jsp:setProperty name="beanId“
property="sample“ value="test1" />
```

This is equivalent to

```
<% beanId.setSample("test1"); %>
```

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, cont.

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

```
<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 27.6 Computing Loan Using JavaBeans

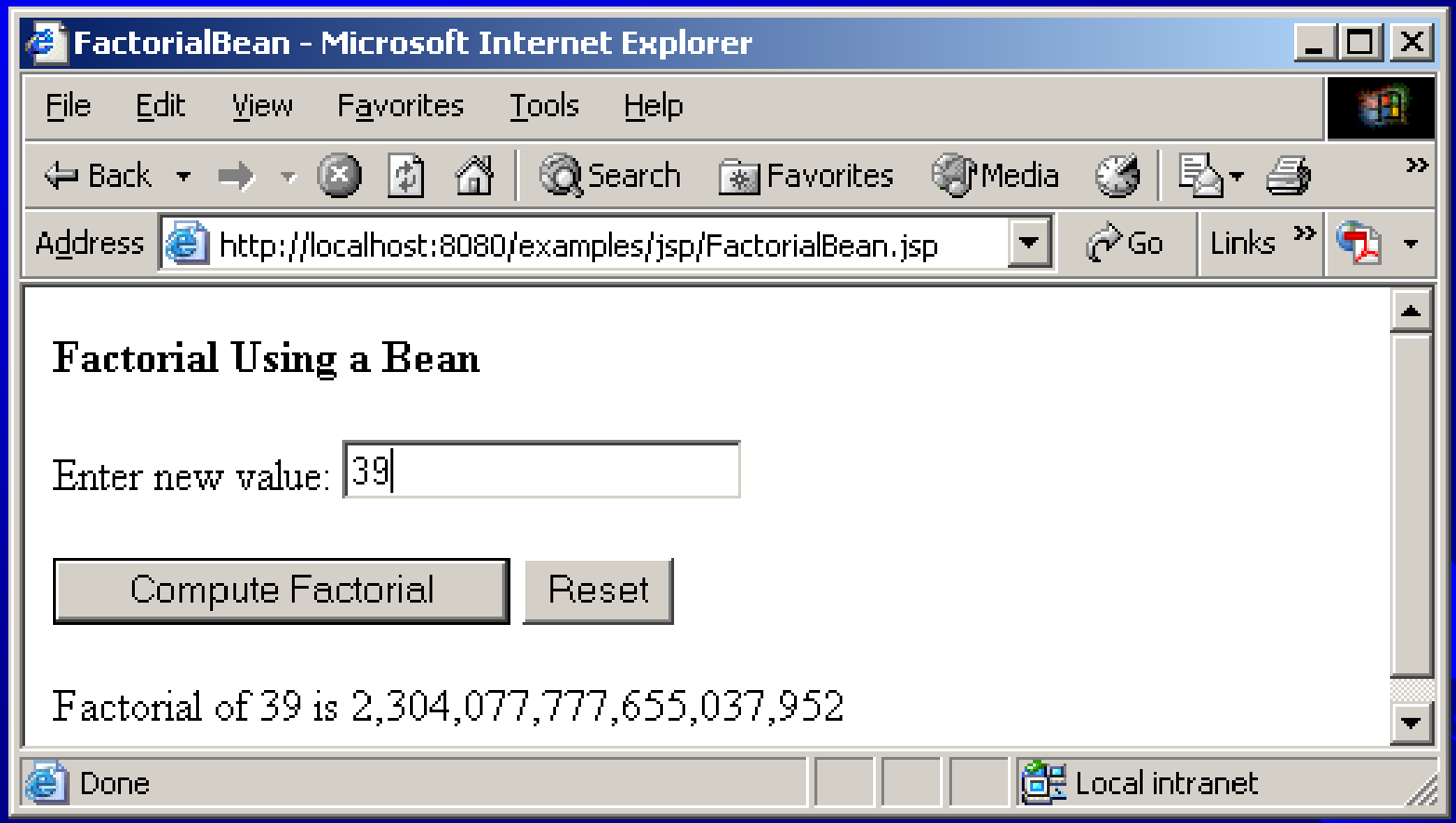
Use JavaBeans to simplify Example 27.3 by associating the bean properties with the input parameters.

```
<!-- ComputeLoan.jsp -->
<html>
<head>
<title>ComputeLoan Using the Loan Class</title>
</head>
<body>
<%@ page import = "chapter27.Loan" %>
<jsp:useBean id="loan" class="chapter27.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 27.7 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 = "chapter27.FactorialBean" %>
<jsp:useBean id="factorialBeanId" class="chapter27.FactorialBean" >
</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">
<P>Factorial of
<jsp:getProperty name="factorialBeanId" property="number" /> is

<%=factorialBeanId.getFactorial() %>
</FORM>
</BODY>
</HTML>
```

Associating the bean properties with the input parameters.

Getting number

```
package chatper28;

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;
  }
}
```

Example 27.8 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.