

Information Management Resource Kit

Module on Management of Electronic Documents

UNIT 6. NETWORKING DOCUMENTS AND DATABASES

LESSON 4. DYNAMIC WEBSITES: JAVA SERVER PAGES (JSPS) AND SERVLETS

NOTE

Please note that this PDF version does not have the interactive features offered through the IMARK courseware such as exercises with feedback, pop-ups, animations etc.

We recommend that you take the lesson using the interactive courseware environment, and use the PDF version for printing the lesson and to use as a reference after you have completed the course.



© FAO, 2003

Objectives

At the end of this lesson, you will be able to:

- understand the **advantages** of using **Java technologies**;
- distinguish the different functions of **Servlets**, **Java Server Pages**, and **JavaScripts**.



Introduction

We could work with Java technologies:
I heard that they offer several benefits!



Java, introduced by Sun Microsystems in 1995, is a programming language highly suited to the distributed environment of the Internet.

Java Servlets and Java Server Pages (JSPs) were introduced to facilitate and standardize the use of Java in the web server.

Today Java, Servlet and JSP technology play an important role in web development.

Java technologies

To start, we have to distinguish between these different Java technologies:



Servlets: Java programs that run on a web server and dynamically generate a response to a web browser request. Typically this is an HTML page but it may also be an image, document, spreadsheet etc.

Java Server Pages (JSPs): text files containing HTML tags and Java codes. JSPs simplify the building of dynamic web pages by allowing the programmer to directly insert Java into regular HTML pages. The web server then automatically converts JSPs into Servlets.

Please don't confuse JSPs with...

JavaScript: an interpreted scripting language based on ideas found in Java. JavaScript is embedded in web pages to add interactivity. Unlike Servlets and JSPs, JavaScript typically executes directly on the user's computer and not on the remote web server.

Servlets

Servlets dynamically prepare a response to a web browser request. Typically this response is an HTML page. Building dynamic web pages with Servlets is useful and commonly done, because:

- They **generate web pages based on data submitted by the user** (e.g.: result pages from search engines).
- The **data can change frequently**. For example, a weather-report or news headlines might build the page dynamically.
- The web pages **use information from corporate databases** or other such sources. For example, an online store web page that lists current prices and number of items in stock.
- **Execute server side actions** such as processing of orders for e-commerce sites, login, making a reservation or payment.



Servlets



Servlets have several key advantages compared with the older technology of CGI.

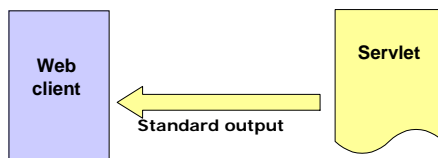
They are more efficient, and allow several things to be done easily that are difficult or impossible with regular CGI.

One major feature is that **Servlets can talk directly to the web server**; this simplifies operations that need to look up images and other data stored in standard places.

Servlets are written in Java and follow a well-standardized API (Application Programming Interface). They can be deployed with almost every commercial and open source web server, including Apache and Microsoft IIS.

Servlets

Servlets work in a similar way to CGI, in that output to be sent back to the web client must be generated on the **standard output**, frequently using the Java **println** statement.



This means that servlets which involve a lot of user interaction (and generate a lot of HTML code) can be rather difficult to write and maintain. Hence Servlets tend to get used more in scenarios where there is **little or no interaction** with the web browser client or the response is not an HTML page.

If there is a lot of HTML passed back to the client, it is more usual to use **Java Server Pages**.

Java Server Pages

Technologies such as CGI and Servlets make you generate the entire page via your program, even though most of it always remains the same.

JSPs let you **embed the program logic directly in a regular HTML page**. Here is an example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0  
Transitional//EN">  
<HTML>  
<HEAD><TITLE>Welcome to Our Store</TITLE></HEAD>  
<BODY>  
<H1>Welcome to Our Store</H1>  
<SMALL>Welcome,  
<!--User name is "New User" for first-time visitors -->  
<%=request.getRemoteUser()%>  
To access your account settings, click  
<A HREF="Account-Settings.html">here. </A></SMALL>  
<P>  
Regular HTML for all the rest of the on-line store's web page.  
</BODY></HTML>
```

The codes included between `<%=` and `%>` tags contain Java instructions.

In this example, the method `"getRemoteUser()"` of the Request object retrieves the session Username.

In order to be a Java Server Page, this file must be saved as **.jsp**.

Java Server Pages



Actually, a JSP can be considered as an HTML (or XML) page, a **text file**, with calls to Java functions embedded in it, whereas a Servlet is a **Java program** which has bits of HTML generated in print statements.

For this reason **JSPs can be created and maintained by people with only limited programming knowledge**: they can call functions that have been created by programmers, without knowing exactly how they work, just what they do and what their input and output parameters are.

The use of JSPs does not exclude the use of Servlets, indeed often both technologies are used in concert to make a website.

Java Server Pages

Embedding bits of Java code into JSPs is quite easy. However, when we have a lot of Java codes in the HTML, identifying the Java code among the HTML tags can become difficult.

The following solutions to this problem have been developed.



JavaBeans

A JavaBean is a reusable software component that can be visually manipulated in builder tools. Standard tags then allow for beans to be used and manipulated in the JSP page.

Extensible JSP Tags

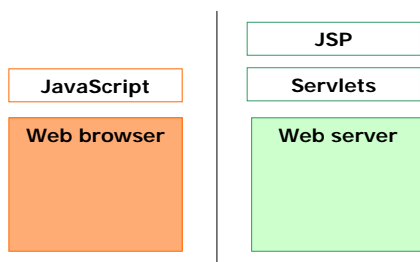
While JavaBeans can be used to encapsulate much of the Java code, using them in JSPs still requires content developers to have some knowledge of Java syntax.

JSP tag libraries allow the tags available in JSPs to be extended. Tag libraries can be created by the development team. Many tag libraries are available, of special interest **JSP Standard Tag Library (JSTL)** which covers many commonly used functions.

JavaScript

JavaScript is another Java technology that can generate HTML dynamically **on the client**.

This is a useful capability, but only handles situations where the dynamic information is based on the client's environment.



Since JavaScript runs on the client, **server-side resources cannot be accessed**. This includes page request data and HTTP information.

However, JavaScript **can access the contents of the local page and cookies**. It can also modify the page being displayed.

An advantage of using JavaScript is that the page can interact directly with the user without returning to the server.

However, you don't know the exact client environment, therefore you will never know exactly how JavaScript will run.

Note that JavaScript may be included in pages dynamically prepared by a JSP or Servlet on the server.

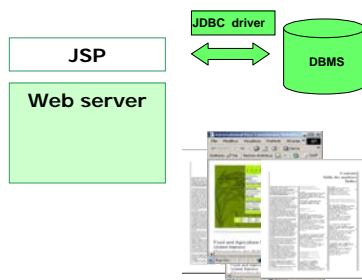
JavaScript

A user is visiting an online Travel service. In your opinion, which of the following web pages would use JavaScript?

- This flight corresponds to your request :
- 6:35 PM** Depart
Lisbon (LIS)
Arrive Tangier (TNG)
11:05 PM
- This is the price of the flight you selected :
- Price: **\$328.30**
Travelers:
Total \$656.60
- Your flight has been **confirmed** :
- Your reservation number is
1234567890

Select the answer of your choice

Building a dynamic website using JSPs



We are now going to create a simple **document management website** using a basic **JSP** design.

Required:

- **web server** (e.g.: Tomcat or Resin).
- **DBMS** (e.g. MySQL, Microsoft SQL Server or Oracle). In this example, we will use Microsoft SQL server.

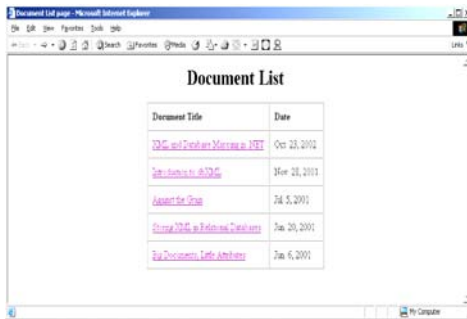
We also need a **JDBC driver** to connect to the database.

JDBC (Java Database Connection) is a core part of Sun Microsystem's Java Platform (<http://java.sun.com>).

It is a standardized programming interface that allows a program to interact with a relational database without having to know the specifics of the flavor of database server being used.

In order to achieve this standardization, the database vendor provides a **JDBC Driver** which translates the queries and actions requested by the program into equivalent commands understood by the specific database.

Building a dynamic website using JSPs



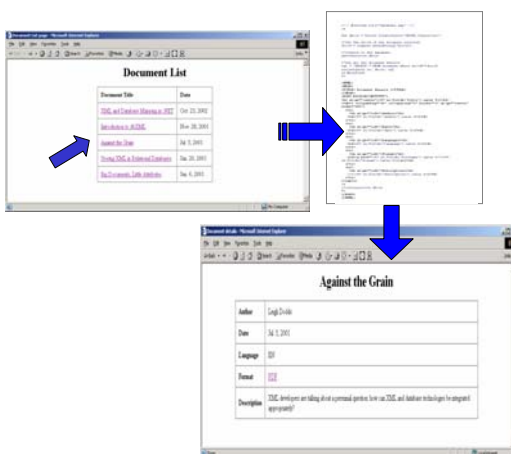
View Animation

The first page will be similar to the one on the left.

To access, for example, the first document, the user will have to:

- select the **document title**, then
- select the format, "**PDF**", to bring up the document itself.

Building a dynamic website using JSPs



When the user selects the title "Against the Grain" on the document list page, a **details page generates the screen** with the details of the relevant document.

We will only need one details page.

Let's look at how to do it.

Building a dynamic website using JSPs

Firstly, we need to **create a database**, e.g. called **documentstore**, with a table **document**, that has all the document details.

In this example, we use SQL to create a table for **five documents**. The "docid" value (1,2,3,4,5) is the primary key identifying each document.

Then, we set up JDBC in order to use it in our program.

Table document (fragment)

```
INSERT INTO document VALUES (1, 'XML and Database Mapping in .NET', 'Niel Bornstein', 'Oct. 23, 2002', 'EN', 'PDF', 'D1 - XML and Database Mapping.pdf', 'Continuing his look at .NET's XML processing from a Java point of view, Niel Bornstein discovers .NET's facilities for binding XML to databases.');
```

```
INSERT INTO document VALUES (2, 'Introduction to dbXML', 'Kimbro Staken', 'Nov. 28, 2001', 'EN', 'PDF', 'D2 - Introduction to dbXML.pdf', 'Following on from his introduction to native XML databases, Kimbro Staken introduces the dbXML open source native XML database');
```

 [Setting up JDBC](#)

 [View the entire Table document](#)


Building a dynamic website using JSPs

The first JSP page we will create gives the database calls required by the other pages. We create a file called **database.jsp**. The instruction `<sql:setDataSource .../>` in the fragment sets the connection to the database.

database.jsp (fragment)

```
<%@ taglib prefix="sql" uri="/WEB-INF/tld/sql.tld" %>
<%@ taglib prefix="c" uri="/WEB-INF/tld/c.tld" %>
<sql:setDataSource dataSource="jdbc/TestDB" />
<c:if test="${!empty param.populate}">
  <sql:transaction>
    <sql:update var="create_document_table">
      CREATE TABLE document (
        docid      INTEGER NOT NULL PRIMARY KEY,
        title      VARCHAR(80) NOT NULL,
```

(All the examples use the MySQL database. For references to the usage of MySQL with the Java language, consult the database provider website at <http://www.mysql.org>).

 [View the entire database.jsp file](#)

 [More information about database connection](#)

Building a dynamic website using JSPs

The database is defined for security reasons in a configuration file called web.xml.

Web.xml (fragment)

```
<resource-ref>
  <description>DB Connection</description>
  <res-ref-name>jdbc/TestDB</res-ref-name>
  <res-type>com.mysql.jdbc.jdbc2.optional.MysqlDataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

[View the entire web.xml file](#)

Building a dynamic website using JSPs

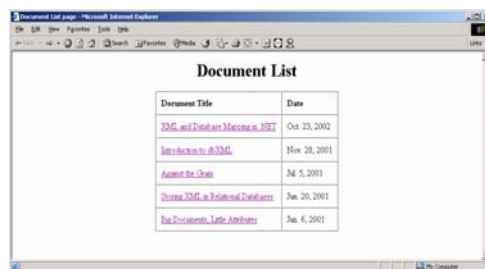
Now we are ready to create the first page that will contain the document list.

We call this page:

DynamicDocumentList.jsp.

To access this database, we use two tag libraries: the Standard Tag Library and the SQL Tag Library, declared at the very top of the page.

Thus the **DynamicDocumentList.jsp** can dynamically get the document list from the database.



Document Title	Date
JDBC and Database Migration JEE	Oct 23, 2002
Introduction to J2EE	Nov 28, 2001
Against the Grain	Jul 5, 2001
Getting J2EE in Relational Databases	Jan 26, 2001
The Document, Little Ambrosia	Jan 4, 2001

[View the DynamicDocumentList.jsp file](#)

Building a dynamic website using JSPs

To query such a database, we use the SQL tags as follows:

DynamicDocumentList.jsp (fragment)

```
<c:choose>
  <c:when test="{!empty param.docid}">
    <sql:query var="select_documents">
      SELECT *
      FROM document
      WHERE docid=?
      <sql:param value="{param.docid}"/>
    </sql:query>
  </c:when>
  <c:otherwise>
    <sql:query var="select_documents">
      SELECT docid, title, date
      FROM document
    </sql:query>
  </c:otherwise>
</c:choose>

<sql:query var="project">
  SELECT * FROM project
</sql:query>
```

There is a hypertext link to `details.asp` including the `docid` as a parameter. This is so that we only need one details page. The `docid` will tell the `details.jsp` page which document was selected.

Building a dynamic website using JSPs

To retrieve data from the tables, we use instead:

DynamicDocumentList.jsp (fragment)

```
<h1 align="center">Document List</h1>
<table cellpadding="10" cellspacing="0" border="1" align="center">
  <tr>
    <th align="left">Document Title</th>
    <th align="left">Date</th>
  </tr>

  <c:forEach var="row" items="{documents.rowsByIndex}">
    <tr>
      <td><c:out value="{row.}" /></td>

      <td><a href="details.jsp?docid={%=rs.getInt("docid")%}"> {%=rs.getString("title")%}</a></td>
      <td>{%=rs.getString("date")%}</td>

    </c:forEach>
  </tr>
</c:forEach>
</table>
```

This way you are able to retrieve the data without revealing sensitive information about your database.

Building a dynamic website using JSPs

The **details.jsp** page will then **get all the details** for each document from the database.

Adding a new document is now done by simply inserting a new row into the document table in the database; no change to the JSP is needed.

Details.jsp (fragment)

```
<c:if test="${!empty param.docid}">
  <c:forEach var="document" begin="0"
    items="${ document_query.rows}">
    <h1 align="center"><c:out
      value="${ document.title}"/></h1>
    <table cellpadding="10" cellspacing="0" border="1"
      align="center" width="80%">
      <tr>
        <th align="left">Author</th>
        <td><c:out value="${ document.author}"/></td>
      </tr>
```

[View the entire details.jsp file](#)

Summary

- There are three different **Java technologies** that allow us to build dynamic web pages: Servlets, Java Server Pages (JSPs) and JavaScript.
- **Servlets** are Java programs that run on a web server and **build web pages dynamically**; they work in a similar way to CGI, but they are more efficient.
- **Java Server Pages** simplify the building of dynamic html web pages by allowing the programmer to insert Java directly into regular HTML pages.
- **JavaScript** is a Java technology that can generate HTML dynamically **on the client**. This capability only handles situations where the dynamic information is based on the client's environment.
- Together these technologies allow effective implementation of powerful dynamic websites.



Exercises

The following three exercises will allow you to test your understanding of the concepts described up to now.

Good luck!



Exercise 1

Can you define the following Java technologies?

Servlets

JSPs

JavaScripts

Technology allowing the creation of web pages containing dynamically-generated content.

Program running on the server that generates dynamic HTML.

Scripts that generate HTML dynamically on the client.

Click on each option, drag it and drop it in the corresponding box.

Exercise 2

If there are ten simultaneous requests to the same servlet, then:

- the servlet is loaded into the memory ten times.
- the servlet is loaded into the memory only the first time.

Select the answer of your choice

Exercise 3

Can you associate the following actions with the relevant technologies?

Servlets

JSPs

JavaScripts

Embed Java functions in a regular HTML page on the server.

Generate the whole HTML page from a Java function.

Can't access resources in a database.

Click on each option, drag it and drop it in the corresponding box.

If you want to know more...

General JSP resources: <http://java.sun.com>

General JSP resources for developers:
<http://developer.java.sun.com/developer/onlineTraining/JSPIntro>

<http://java.sun.com/products/servlet/>

<http://jakarta.apache.org/tomcat/>

<http://developer.java.sun.com/developer/onlineTraining/Beans/Beans1/>

<http://jakarta.apache.org/tomcat/tomcat-4.1-doc/printer/jndi-datasource-examples-howto.html#MySQL%20DBCP%20Example>

<http://jakarta.apache.org/taglibs/index.html>

