Java Server Pages

Môn học: Java Server Pages

Bài 1

Những vấn đề chính sẽ được đề cập trong khoá học:

- ✓ *Cài đặt JRUN.*
  - o *Cài đặt JDK1.3.*
  - o *Cài đặt JRUN.*
  - o *Cấu hình ứng dụng JSP*
- ✓ *Giới thiệu JSP.*
  - o *Script.*
  - o *Khai báo chỉ mục.*

1. CÀI ĐẶT JRUN

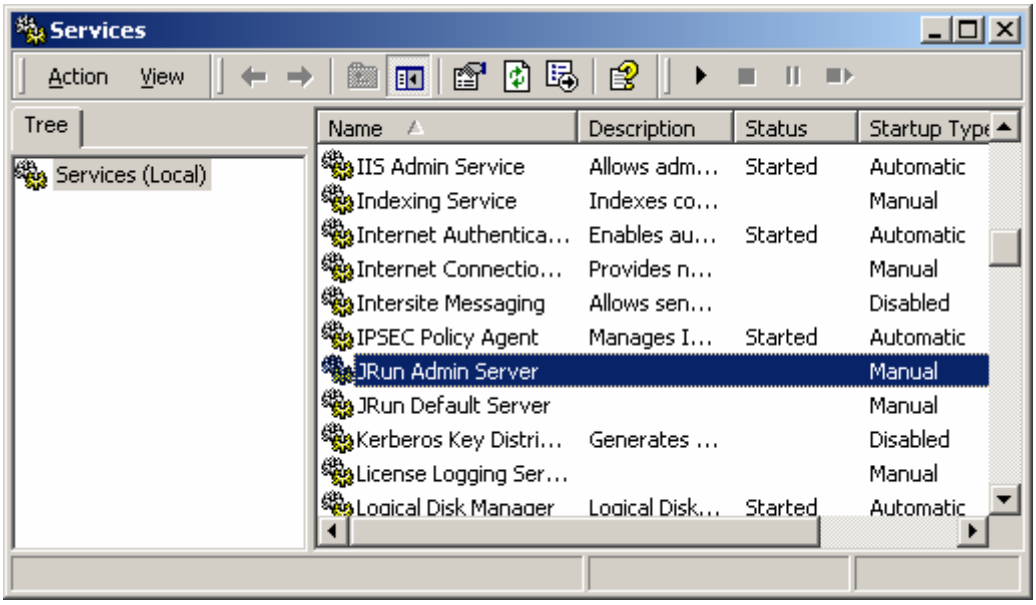
1.1. Cài đặt JDK

Để cấu hình JRUN 4.0 trên Windows, trước tiên bạn cài đặt bộ JDK1.3 hay JDK 1.4 lên đĩa cứng C hay D, sau khi cài đặt thành công bạn nên boot máy lại, trong ổ đĩa sẽ xuất hiện thư mục JDK1.3 với các thư viện của chúng.

1.2. Cài đặt JRUN

Để cài đặt JRUN, bạn mua đĩa CD với ứng dụng JRUN 4.0, sau đó cài đặt chúng vào máy có cài đặt JDK. Trong khi cài đặt thì JRUN có yêu cầu chòn thư mục nơi bạn cài đặt JDK.

Trong khi cài đặt JRUN, có thể yêu cầu bạn cài đặt chúng dưới dạng một dịch vụ của hợp đồng hành (mặc định là Yes), khi đó JRUN sẽ cài đặt với hai dịch vụ là JRUN Admin và JRUN Default trong cửa sổ Services (trong Control Panel hay Administrative Tools) của hệ điều hành Windows như hình 1-1 sau:



Hình 1-1: Dịch vụ JRUN trong Servies

Bạn phải bảo dảm rằng chúng đang ở chế độ Started, trong trường hợp đang ở chế độ STOP thì bạn chọn dịch vụ này và nhấn nút Start.

Nếu JRUN đang ở chế độ Start thì bạn sẽ gõ http://localhost:8000 trên trình duyệt và cung cấp username cung với password để đăng nhập vào màn hình uqản trị JRUN nếu muốn chương.

Nếu JRUN Default Server đang ở chế độ Start thì bạn có thể gõ địa chỉ sau trên browser http://localhost:8100 để chạy ứng dụng mặc định của JSP.

Ngoài ra, nếu bạn không cài đặt JRUN như một dịch vụ của hệ điều hành Windows thì mỗi lần bạn muốn chạy ứng dụng JSP thì phải Start nó lên bằng cách chọn Start | Programs | JRUN 4.0 | JRUN Default Server hay JRUN Admin Server roi sau do khởi động JRUN Default Server, phần này sẽ trình bày trong phần cấu hình.

1.3. Cấu hình ứng dụng JSP

Để triệu gọi trang JSP (tên mở rộng .jsp) trên trình duyệt, bạn có thể cấu hình một torng hai cách sau:

1.3.1. Cách 1:

Nếu không muốn tạo ứng dụng Web cho riêng mình mà sử dụng thư mục mặc định thì bạn có thể khai báo các trang JSP và bỏ vào thư mục JRUN4/Servers/Defaul/ default-ear/ default-war/.

Chẳng hạn, trong trường hợp này chúng ta khai báo trang test.jsp với nội dung như sau:

```
<HTML>
   <HEAD>
   <title>Test JSP</title>
   </HEAD>
   <BODY>
            <%="Hello JSP"%>
   </BODY>
</HTML>
```

Để kiểm tra trang JSP này, bạn khai báo trên trình duyệt như sau:

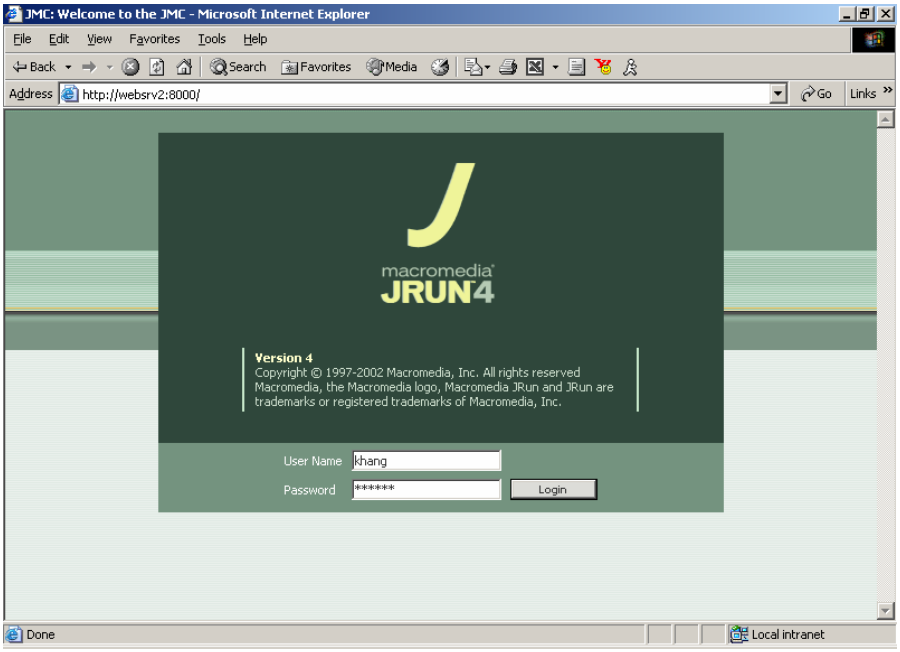
```
http://localhost1:8100/test.jsp
```

Tuy nhiên, bạn có thể khai báo các thư mục con bên trong của thư mục mặc định này, sau đó triệu gọi trên trình duyệt theo tên thư mục, chẳng hạn http://localhost:8100/**qlda/danhmucphongban.jsp.**

1.3.2. Cách 2:

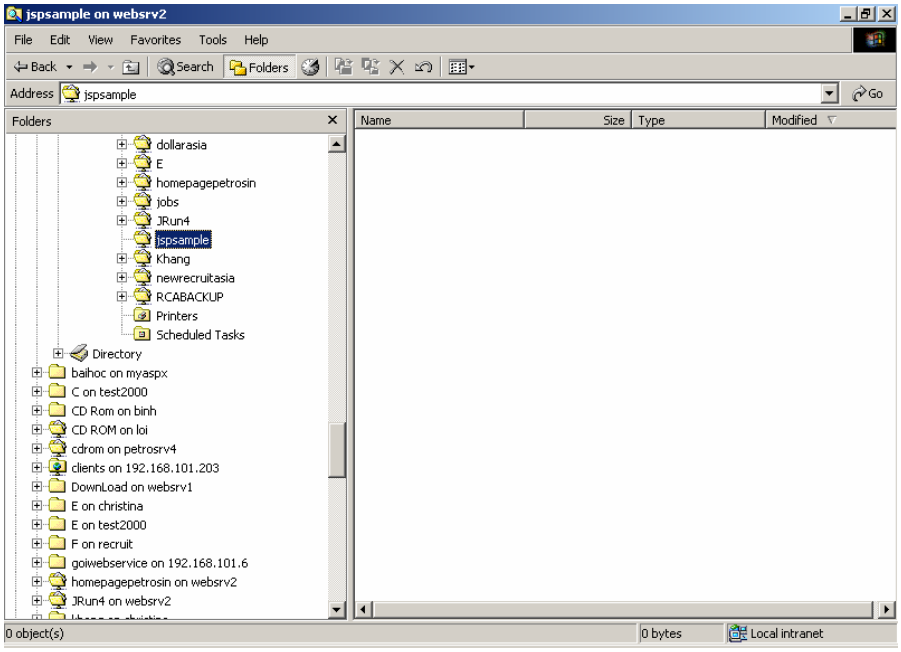
Để cấu hình ứng dụng JSP trên JRUN bạn có thể thao tác theo các bước sau:

1. Bạn đăng nhập vào JRUN Admin với http://localhost:8000 bằng cách cung cấp username/pwd như hình 1-2 sau.



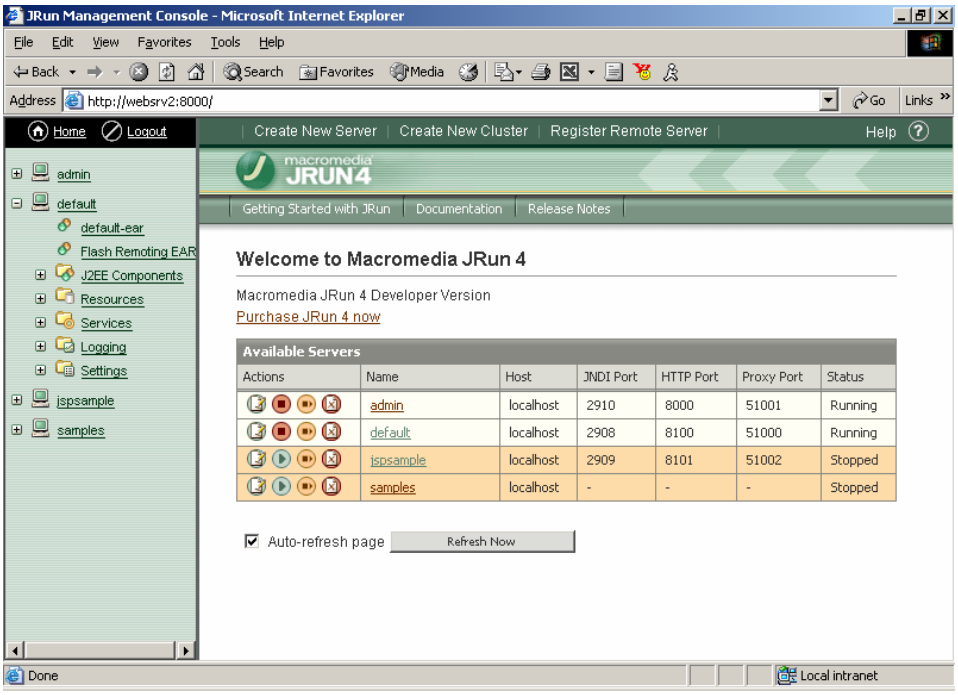
Hình 1-2: Đăng nhập JRUN Admin

2. Tạo thư mục lưu trữ web site trên ổ đĩa, chẳng hạn trong trường hợp này khai báo thư mục jspsample như hình 1-3.



Hình 1-3: Khai báo thư mục để lưu trữ trang JSP

3. Chọn vào menu Home, JRUN sẽ liệt kê danh sách các server đang cấu hình trên JRUN như hình 1-4.



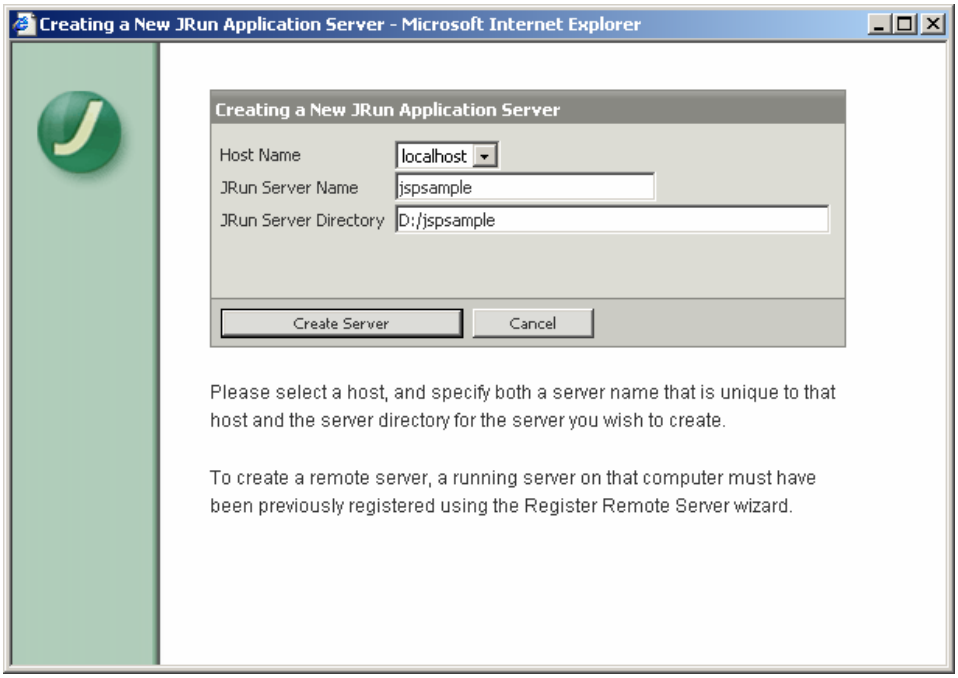
Hình 1-4: Danh sách các Server trên JRUN

4. Chọn URL có tên là Create New Server trên menu nằm ở đầu trang, trong cửa sổ vừa xuất hiện (Creating a New JRun Application Server) bạn cung cấp các tham số trong hình 1-5 như sau:

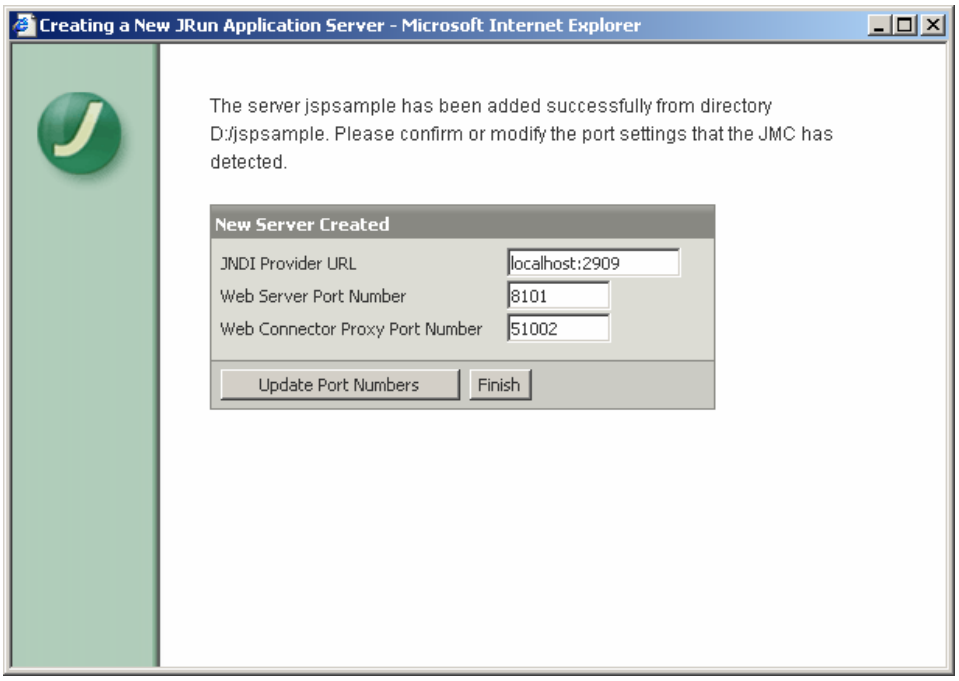
- Host Name:localhost (hoặc địa chỉ IP, tên server)
- JRun Server Name: Chọn tên của ứng dụng, ví dụ trong trường hợp này chọn jspsample.
- JRun Server Directory: Thư mục của ứng dụng bạn vừa tạo ra, chẳng hạn D:/jspsample (hay D:/thuvien/hr).

5. Nhấn nút Create chờ trong giây lát cửa sổ của HOME page sẽ xuất hiện tên ứng dụng web bạn vừa tạo (jspsample), nhấn nút Start để khởi động ứng dụng này.

   Trong trường hợp phát sinh lỗi do port bị đụng độ, mặc định của site default là 8100 (admin là 8000), khi bạn tạo ứng dụng xong, một port nào đó được gán cho ứng dụng của bạn vì dụ như 8101 như hình 1-6. Tuy nhiên, bạn có thể thay đổi port này bằng cách edit để cập nhật.

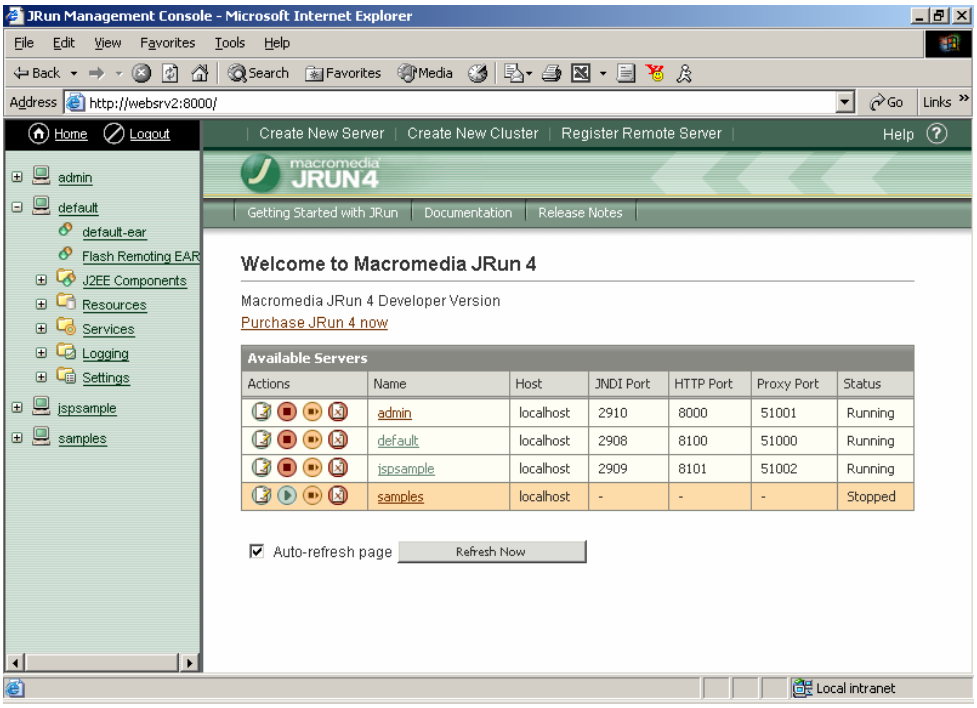


Hình 1-5: Tạo website



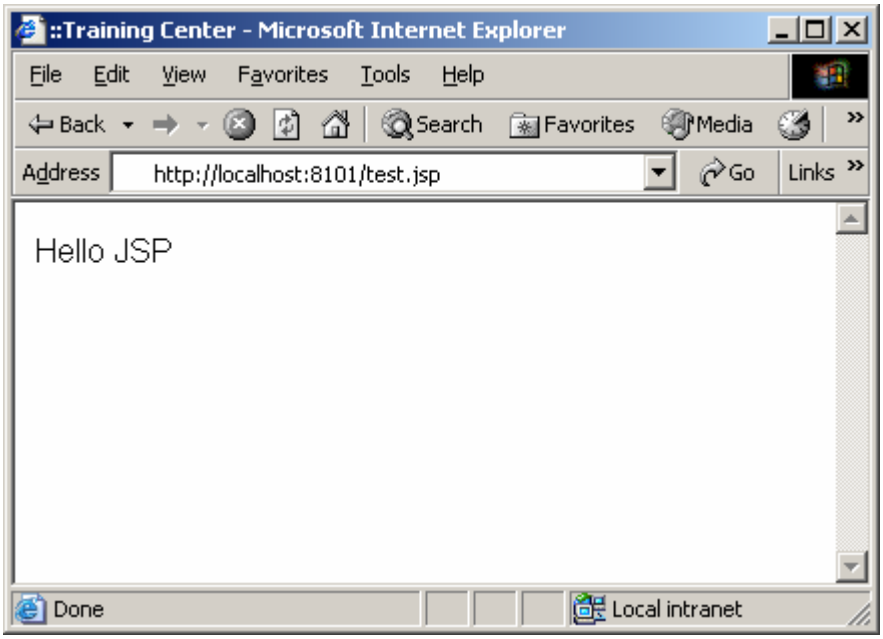
Hình 1-6: Tạo ứng dụng JSP thành công

Sau khi tạo thành công, tên ứng dụng vừa tạo sẽ xuất hiện trong danh sách server của JRUN như hình 1-7.



Hình 1-7: Tên site vừa tạo

Xem trên bảng danh sách server ứng với jspsample có phần HTTP Port là 8101, sau đó trên http, bạn gõ http://localhost:8101/test.jsp, kết quả sẽ xuất hiện như hình 1-8.



Hình 1-8: Kết quả trang test.jsp

2. GIỚI THIỆU JSP

2.1. Yêu cầu

JSP dựa trên cú pháp của ngôn ngữ lập trình Java, chính vì vậy khi làm việc với JSP bạn phải là người có kiến thức về ngôn ngữ này. Nếu bạn xây dựng ứng dụng JSP có kết nối cơ sở dữ liệu thì kiến thức về cơ sở dữ liệu SQL Server hay Oracle là điều cần thiết.

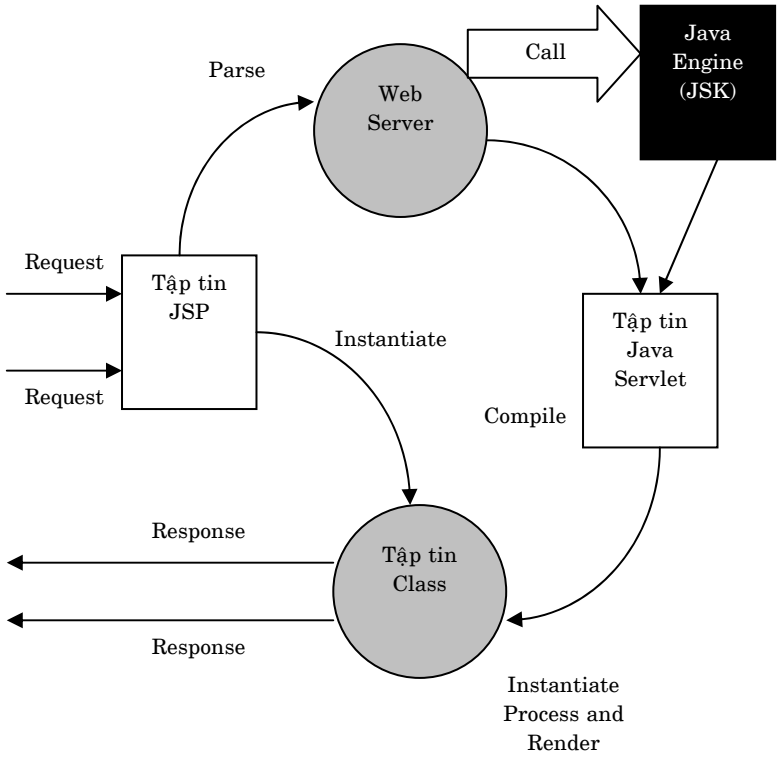
2.2. Giới thiệu

JSP là kịch bản trình chủ (Server Script) được chạy trên nền JDK 1.3 trở về sau, cùng với ứng dụng Web Server để quản lý chúng. Web Server thường sử dụng là Tomcate, Java Web Server, JRUN, WebLogic và Apache, ...

Tiền thân của JSP là xuất phát từ Java Servlet, khi làm việc với Java Servlet thì hầu hết các lập trình viwên gặp khó khăn khi xuất nhập dữ liệu, cụ thể là giao diện với người sử dụng. Chính vì vậy SUN Microsystem cung cấp kịch bản JSP là phần mở rộng của Java Servlet để cho phép quá trình lập trình ứng dụng Web trở nên đơn giản hơn. Tuy nhiên, những trang JSP này khi biên dịch đề thông qua trang trung gian là Java Sevrlet.

2.3. Biên dịch trang JSP

Khi người sử dụng gọi trang *JSP* lần đầu tiên, *Web Server* triệu gọi trình biên dịch dịch trang *JSP* (trong trường hợp này là JDK) thành tập tin Java, kế đến tập tin java (Java Servlet) này sẽ biên dịch ra *Class* . Sau đó, trang *class* thực thi và trả về kết quả cho người sử dụng như hình 1-9.



Hình 1-10: Quá trình biên dịch trang JSP

2.4. Kịch bản (script)

Nội dung của JSP có thể khai báo lẫn lộn với HTML, chính vì vậy bạn sử dụng cặp dấu <%= %> để khai báo mã JSP. Chẳng hạn, chúng ta khai báo:

```
<br>
1-Giá trị biến Str: <%=str%>
2-Giá trị biến i: <%=i%>
3-Giá trị cũ thể: <%=10%>
```

Trong trường hợp có nhiều khai báo, bạn sử dụng Scriptlet, đều này có nghĩa là sử dụng cặp dấu trên như <%%> với các khai báo JSP với cú pháp của Java như sau:

```
<%
int i=0;
String str="Select * from tblABC";
out.println(str);
%>
-Khai báo trên là Scriptlet
Giá trị của i: <br>
<%=i%> -Khai báo này là Script
```

3. KẾT LUÂN

Trong bài này, chúng ta tập trung tìm hiểu cách cài đặt JSK và JRUN, sau đó cấu hình ứng dụng JSP trong JRUN hay sử dụng cấu hình mặc định của chúng.

Ngoài ra, bạn làm quen cách khai báo mã JSP trong trang .jsp cùng với script hay scriptlet.

Môn học: Java Server Pages

Bài 2

Bài học này chúng ta sẽ làm quen và tìm hiểu cú pháp và một số phương thức cơ bản của JSP:

- ✓ *Câu lệnh.*
- ✓ *Biến và kiểu dữ liệu.*
- ✓ *Hằng.*
- ✓ *Bảng dãy*
- ✓ *Một số phương thức cơ bản*

1. KHÁI NIỆM VỀ CÚ PHÁP JSP

Cú pháp JSP chính là cú pháp trong ngôn ngữ Java, các bạn làm quen với ngôn ngữ Java thì có lợi thế trong lập trình JSP.

Để lập trình bằng ngôn ngữ JSP cần chú ý những điểm sau:

- ❖ Cuối câu lệnh có dấu ;
- ❖ Mỗi phương thức đều bắt đầu { và đóng bằng dấu }
- ❖ Khi khai báo biến thì kiểu dữ liệu nằm trước tên biến
- ❖ Nên có giá trị khởi đầu cho biến khai báo
- ❖ Phải có chi chú (comment) cho mỗi feature mới
- ❖ Sử dụng dấu // để giải thích cho mỗi câu ghi chú
- ❖ Sử dụng /* và */ cho mỗi đoạn ghi chú
- ❖ Khai báo biến có phân biệt chữ hoa hay thường
- ❖ Tên file và lớp cũng như như khai báo biến

2. KHAI BÁO BIẾN

Khi thực hiện một việc khai báo biến trong java, bạn cần phải biết tuân thủ quy định như: kiểu dữ liệu trước tên biến và có giá trị khởi đầu

Xuất phát từ những điều ở trên, khai báo biến trong Java như sau:

- ❖ Datatype variable name [initial value];

```
int licount=0;
String lsSQL="Select * from tblusers where active=1";
double account[];
boolean checkerror=false;
```

3. KIỂU DỮ LIỆU

Bảng các kiểu dữ liệu thông thường

Type	Bytes	Range
Boolean	2	

Byte	1	
Char	2	
Double	8 cho âm, 4 số dương	
Float	4	
Int	4	
Long	8	
Short	2	
Connection		
Statement		
ResultSet		

3.1. Kiểu Array

Kiểu mảng là một mảng số liệu do người dùng định nghĩa, chúng có cú pháp như sau:

```
double account[]; // mảng số double
```

hay có thể khai báo như sau

```
double account[]={0,0,1,45.95,6.5};
```

thứ tự index trong mảng bắt đầu từ vị trí 0. Nếu như bạn khai báo mảng hai chiều, thì cú pháp khai báo như sau:

```
double account[][]=new double[2][5];
```

Chẳng hạn khai báo như sau:

```
<%
    double account[][]=new double[2][100];
    account[0][3]=43.95;
    account[1][3]=43.95000;
    out.println("Account 0-3 is " + account[0][3] );
    out.println("Account 1-3 is " + account[1][3] );
%>
```

Khai báo biến String

```
<%
    String strSQL="select * from tblusers ";
    String strWhere=" where active=0";
```

```
        out.println("SQL Statement is " + strSQL+strWhere );
    %>
```

Khai báo với nhiều loại dữ liệu

```
    <%
     boolean bo;
     byte by;
     char c;
     short s;
     int i;
     long l;
     float f;
     double d;
     object o;
      int[] intArray = new int[2];
      object[] objectArray = new Object[2];
      out.println("boolean: "+bo);
      out.println("byte: "+by);
      out.println("char: "+c);
      out.println("short: "+s);
      out.println("int: "+i);
      out.println("long: "+l);
      out.println("float: "+f);
      out.println("double: "+d);
      out.println("Object: "+o);
      out.println("int[2]: "+intArray[0]+" "+intArray[1]);
      out.println("Object[2]: "+objectArray[0]+" "+objectArray[1]);
    %>
```

4. CÁC PHƯƠNG THỨC VÀ PHƯƠNG THỨC TRONG JAVA

4.1. Phương thức trả về chiều dài mảng

Khi quan tâm đến chiều dài của mảng thì bạn cần theo cú pháp sau:

Array.length

Giả sử rằng, bạn khai báo biến với chiều dài mãng một và hai chiều

```
<%
double account[]={88,11,2.5,77};
double sum;
sum=account.length;
```

```
out.println("Length of Account  is " + sum);%>
```

4.2. Chuyển sang kiểu chuỗi

Khi bạn cần chuyễn đổi từ kiểu số liệu khác sang kiểu chuỗi, thì cần khai báo như sau:

String.valueOf(data);

Ví dụ chuyển đổi kiểu sang kiểu chuỗi

```
<%
    double account[]={88,11,2.5,77};
    String str;
    str=String.valueOf(account[2]);
    out.println("String of Account 2 is " + str);
%>
```

4.3. Nối chuỗi

Khi cần thiết nối hai hay nhiều chuổi lại với nhau, bạn sử dụng phương thức concat, thông thường chúng ta hay dung phép toán + để nối hai hay nhiều chuỗi lại với nhau.

Cú pháp concat như sau:

Str1.conact(Str2);

Kết nối chuỗi

```
<%
    String str1="Select * from tblemplyers";
    String str2=" where paid=1";
    str1=str1.conact(str2);
    out.println("String of Str1 is " + str1);
%>
```

4.4. Chuỗi con

Khi bạn cần lấy một chuỗi con trong chuỗi lớn, bạn cần dùng đến phương thức với cú pháp như sau:

str1=str2.substring(start,chiều dài)

Ví dụ khai báo để lấy chuỗi con

```
<%
    String str1="Select * from tblemplyers";
    String str2=str1.substring(9,5);
    out.println("Sub String of Str1 is " + str2);
```

```
%>
```

Nhưng nếu có nhu cầu lấy ra một ký tự nào đó trong chuỗi, thì bạn không cần dùng substring mà chỉ sử dụng cú pháp charAt như sau:

```
Char=Str1.charAt(number);
```

Chẳng hạn, khai báo để lấy 1 ký tự

```
<%
    String str1="Select * from tblemplyers";
    String str2=str1.charAt(5);
    Out.println("charAt of Str1 is " + str2);
%>
```

4.5. Chuyễn đổi String sang Array

Thông thường trong khi tính toá chuỗi, đôi khi cũng cần đến chúng như một mãng, lý do đó chúng ta có phương thức chuyễn đổi như sau:

```
char char1[]=str1.toCharArray();
```

Chuyển chuỗi sang mảng

```
<%
    String str1="Select * from tblemplyers";
    char char1=str1.toCharArray();
    out.println("Char of Str1[1] is " + char1[5]);
%>
```

4.6. Thay thế chuỗi

Khi cần thay thế một chuỗi con nào đó trong chuỗi mẹ thành chuỗi con khác, chúng ta cần đến phương thức replace có cú pháp như sau:

```
str1=str2.replace("","""");
str1=str2.replace("a","k");
```

Ví dụ khai báo thay thế chuỗi

```
<%
String str1="Select * from tblemplyurs";
str1= replaceString (str1,"u","o");
str1= replaceString (str1,"","""");
out.println("Char of Str1 is " + str1);
%>
```

Trong đó, khai báo phương thức replaceString như sau

```
<%!
public String replaceString(String sStr,String oldStr,String newStr)
   {
      sStr=(sStr==null?"":sStr);
      String strVar=sStr;
      String tmpStr="";
      String finalStr="";
      int stpos=0,endpos=0,strLen=0;
      while (true)
         {
            strLen=strVar.length();
            stpos=0;
            endpos=strVar.indexOf(oldStr,stpos);
            if (endpos==-1)
               break;
            tmpStr=strVar.substring(stpos,endpos);
            tmpStr=tmpStr.concat(newStr);

   strVar=strVar.substring(endpos+oldStr.length()>sStr.length()?endpos:endpos+old
Str.length(),strLen);
            finalStr=finalStr.concat(tmpStr);
            stpos=endpos;
         }
         finalStr=finalStr.concat(strVar);
         return finalStr;
      }
%>
```

4.7. Vị trí ký tự trong chuỗi

Khi cần biết vị trí cũa ký tự hay chuỗi con nào đó trong chuỗi, bạn sử dụng phương thức sau:

```
virti=str1.indexOf("select");
virti=str1.indexOf("o");
```

Chẳng hạn tìm vị trí chuỗi hay ký tự trong Chuỗi

```
<%
```

```
String str1=”Select * from tblemplyurs”;
int vitri=str1.indexOf(“o”);
Out.println(“Location of \”o\” is ” + vitri);
%>
```

4.8. Kiểu chữ

Nếu muốn chuyễn đổi chữ hoa sang thường hay ngược lại, thì bạn dùng phương thức có cú pháp như sau:

Từ hoa sang thường: str1.toLowerCase();

Từ thường saung hoa: str1.toUpperCase();

Ví dụ

```
<%
String str1=”Select * from tblemplyors”;
str1=str1.toLowerCase();
out.println(“LowerCase is ” + str1);
str1=str1.toUpperCase();
out.println(“UpperCase is ” + str1);
%>
```

5. TÓM TẮT

Trong bài học này chúng tôi giới thiệu đến cho các bạn cách khai báo biến, các kiểu dữ liệu, đồng thời giúp cho các bạn những phương thức trên chuỗi và ký tự trong JSP.

Bài 3

PHÉP TOÁN VÀ PHÁT BIỂU CÓ ĐIỀU KIỆN TRONG JSP

Chương này chúng ta sẽ làm quen và tìm hiểu toán tử, phát biểu có điều kiện và vòng lặp của JSP.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ *Toán tử.*
- ✓ *Phép gán trong Java*
- ✓ *Phát biểu có điều khiển.*
- ✓ *Vòng lặp.*

1. KHÁI NIỆM VỀ CÁC TOÁN TỬ TRONG JSP

Khi bạn lập trình trên JSP là sử dụng cú pháp của ngôn ngữ Java. Tương tự như những ngôn ngữ lập trình khác, toán tử giúp cho bạn thực hiện những phép toán như số học hay trên chuỗi.

Bảng sau đây giúp cho bạn hình dung được những toán tử sử dụng tron Java

Java định nghĩa toá tử toán học, quan hệ, số học, bit, caste, class, selection, và nột só phép toán gán.

Loại toán tử	Toán tử	Diễn giải	Ví dụ
Arithmetic	+	Addition	a + b
	-	Subtraction	a - b
	*	Multiplication	a * b
	/	Division	a / b
	%	Modulus	a % b
Relational	>	Greater than	a > b
	<	Less than	a < b
	>=	Greater than or equal	a >= b
	<=	Less than or equal	a <= b
	!=	Not equal	a != b
	==	Equal	a == b
Logical	!	Not	!a
	&&		a && b

	||	AND OR	a || b
Bit-manipulation	~ & | ^ << >> >>>	Complement AND OR Exclusive OR Left shift Right shift Zero-filled right shift	~a a & b a | b a ^ b a << b a >> B a >>> b
Assignment	= ++ -- += -= *= /= %= |= &= ^= <<= >>= >>>=	Assignment Increment and assign Decrement and assign Add and assign Subtract and assign Multiply and assign Divide and assign Take modulus and assign OR and assign AND and assign XOR and assign Left shift and assign Right shift and assign Zero-filled left shift and assign	a = b a++ a-- a += b a -= b a *= b a /= b a %= b a |= b a &= b a ^= b a <<= b a >>= b a >>>= b
Caste	(type)	Convert to type	(char) b
Instance	instance of	Is instance of class?	a instanceof b
Allocation	new	Create a new object of a class	new A()
Selection	? :	If...Then selection	a ? b : c

2. GIỚI THIỆU TOÁN TỬ

Khi nói đến toán tử, chúng ta luôn liên tưởng đến thứ tự xử lý, cũng như trong toán học, toán tử trong java cũng co độ ưu tiên add-subtract-multi-divide.

2.1. Toán tử AND

Khi thực hiện một việc tăng lên giá trị thì bạn sử dụng cú pháp như sau:

int i=0,j=0;

j=i++;// i tăng sau khi gán i vào j, chính vì vậy sau khi gán i vào j, j vẫn không thay đổi

j=++i;// i tăng trước khi gán i vào j, chính vì vậy sau khi gán i vào j, j thay đổi.

Ví dụ 3.1: Phép toán AND.

```
<%
    int i=0,j=0;
    j=i++;
    out.println("Value of j is " + j);
    j=++i;
    out.println("Value of j is " + j);
%>
```

2.2. Toán tử Not: ~ And !

Toán tử ~ đảo nghịch tất cả các bit của tham số, còn toán tử ! đảo nghịch giá trị của giá trị trước đó

Ví dụ 3.2: Phép toán ~ and !

```
<%
    short i=32767;
    boolean b=true;
    out.println("Value of ~ short is " + ~i);
    out.println("Value of !b is " +!b);
%>
```

2.3. Toán tử nhân và chia: * and /

Bạn có thể tham khảo ví dụ sau

Ví dụ 3.3: Phép toán * và /, + và -

```
<%
    int i=767;
    double j=10.5;
    out.println("Value of multi is " + i*j);
    out.println("Value of divide is " +i/5);
```

```
    out.println("Value of add is " +i+5);
    out.println("Value of subtract is " +i-5);
%>
```

2.4. Toán tử modulus: %

Khi chia một số cho một số, bạn cần kết quả là số dư của phép chia đó thì dùng toán tử modulus

Ví dụ 3.4: Phép toán %

```
<%
    int i=10;
    int j=3;
    out.println("Value of i%j is " + i%j);
%>
```

2.5. Toán tử quan hệ: >=,>,<,<=,==,!=

Khi cần so sánh kết quả giữa hai toán hạn với nhau, thông thường bạn nghĩ đến phép toán so sánh như là bằng, lớn hơn, nhỏ hơn, ví dụ sau diễn giải cho bạn các toán tử trên:

Ví dụ 3.5: Phép toán >,>=,<,<=,==,!=

```
<%
    int i=10;
    int j=3;
    if(i>=j)
        out.println("result is true");
    else
        out.println("result is false");
    if(i!=j)
        out.println("result is not equals");
    else
        out.println("result is equals");

%>
```

2.6. Toán tử && và ||

&& là toá tử and trong só học

|| là toán tử or trong số học

Hai toán tử này rất thường dùng trong khi lập trình trên Java, ví dụ dưới đây diễn giải cho bạn đầy đủ hai toán tử này. Chú ý rằng khi sử dụng toán tử đều có kèm phát biểu có điều kiện.

Ví dụ 3.6: Phép toán && và ||

```
<%
    boolean b=true;
    int j=3;
    if((j>=3) &&(b!=true))
        out.println("result is true");
    if((j<3) ||(b==true))
        out.println("result is false");
%>
```

2.7. Toán tử ?:

Toán tử này thay thế cho phát biểu có điều kiện if.. then .. else, khi bạn cần lấy kết quả theo điều kiện nào đó, nếu có thể không cần phát biểu if-else, thì hãy thay thế bằng toán tử ?:, cú pháp của chúng như sau:

str1=str2.equals("khang")?"Welcome to Java":"Good bye JSP";

Ví dụ 3.7: Phép toán ?:

```
<%
    String str1="Pham Huu Khang";
    String str2 ="Khang";
        out.println("result is true"+ (str1.equals(str2)?"Welcome to
                                        Java":"Good bye JSP"));

%>
```

3. PHÉP GÁN

Khi gán một giá trị hay biến vào một biến trong Java, bạn phải dùng đến phép gán, nhưng trong Java cũng giốngnhư trong C thì có những phép gán được đơn giản hoá hay nói đúng hơn là chuẩn hoá để rút gọn lại trong khi viết.

3.1. Phép gán thông thường nhất như sau:

```
int   j=i;
String  str1 =" Hello!";
boolean    b=true;
```

3.2. Phép gán thêm một giá trị là 1

```
int k=0;
k++;
```

3.3. Phép gán thêm một với chính nó giá trị

```
int k=0,j=1;
k+=j;
```

tương tự như vậy chúng ta có k*=2, nghĩa là k=k*2

4. PHÁT BIỂU CÓ ĐIỀU KIÊN

Các phat biểu có điều kiện như :

- IF (điều kiện) { câu lệnh; }
- IF (điều kiện) { câu lệnh; }ELSE { câu lệnh; }
- switch (điều kiện)

```
{
    case Value1
            câu lệnh1;
            break;
    }
```

- While (điều kiện)
- Do - While (điều kiện)
- Break
- Continue

4.1. Phát biểu IF (điều kiện) { câu lệnh; }

Sử dụng phát biểu if để chọn lọc kết quả khi điều kiện đúng, ví dụ như sau:

Ví dụ 3.8: Phát biểu IF

```
<%
    boolean b=true;
    int j=3;
    if((j>=3) &&(b!=true))
       out.println(“result is true”);
    if((j<3) ||(b==true))
       out.println(“result is false”);
  %>
```

4.2. Phát biểu IF (điều kiện) { câu lệnh; }ELSE { câu lệnh; }

Sử dụng phát biểu if để chọn lọc kết quả khi điều kiện đúng, và xuất ra kết quả khi điều kiện sai, ví dụ như sau:

Ví dụ 3.9: Phát biểu IF - ELSE

```
<%
       boolean b=true;
       int j=3;
       if((j>=3) &&(b!=true))
          out.println(“result is true”);
       else
          out.println(“result is false”);

    %>
```

4.3. Phát biểu Switch (điều kiện)

Phát biểu switch là phần của phát biểu if else nhiều nhánh, khi có nhiều điều kiện chọn lựa thì bạn sử dụng switch, cú pháp của chúng như sau:

```
Switch(điều kiện)
   {
```

```
    case Value1
        câu lệnh1;
        break;
    case Value2
        câu lệnh2;
        break;
    …
        default:
        câu lệnh default;
}
```

Break: dùng để thoát ra khỏi switch khi thoả một case nào đó trong switch, default: khi không có bất kỳ giá trị nào thoản trong các case thì giá trị cuối cùng là defaule statement

Ví dụ 3.10: Phát biểu Switch

```
<%
    int j=3;
    switch(j)
    {
     case 1:
        out.println("Today is  Monday");
        break;
     case 2:
        out.println("Today is Thurday");
        break;
     case 3:
        out.println("Today is Tueday");
        break;
     default:
        out.println("Today is Sunday");
    }
%>
```

4.4. Phát biểu While(điều kiện)

Phát biểu while thực thi những câu lệnh trong while khi điều kiện cò đúng.

Ví dụ 3.11: Phát biểu While

```
<%
    int j=1;
    while(j<=30)
    {
        out.println("Number of j" + j);
        j++;

%>
```

4.5. Phát biểu For

Phát biểu for dùng cho vòng lặp có giới hạn cho trước, cú pháp có dạng như sau:

```
int  i=0;
for(i=1;i<10;i++)
{
   câu lệnh;
}
```

Ví dụ 3.12: Phát biểu For

```
<%
    int mang[]={3,5,6,7,8,9};
    for (int j=0;j<mang.length;j++)
    {
        out.println("Phan tu mang "+j+" : "+mang[j]);
    }
%>
```

5. TÓM TẮT

Trong bài học này chúng tôi giới thiệu đến cho các bạn các phép gán, các toán tử, đồng thời giúp cho các bạn hiểu thêm vào các phát biểu có điều kiện như while, for, switch, ...

Bài 4

ĐỐI TƯỢNG SESSION, REQUEST, RESPONSE TRONG JSP

Chương này chúng ta sẽ làm quen các đối tượng thường sử dụng trong quá trình thiết kế trang JSP.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ *Đối tượng Request.*
- ✓ *Đối tượng Response.*

1. ĐỐI TƯỢNG REQUEST

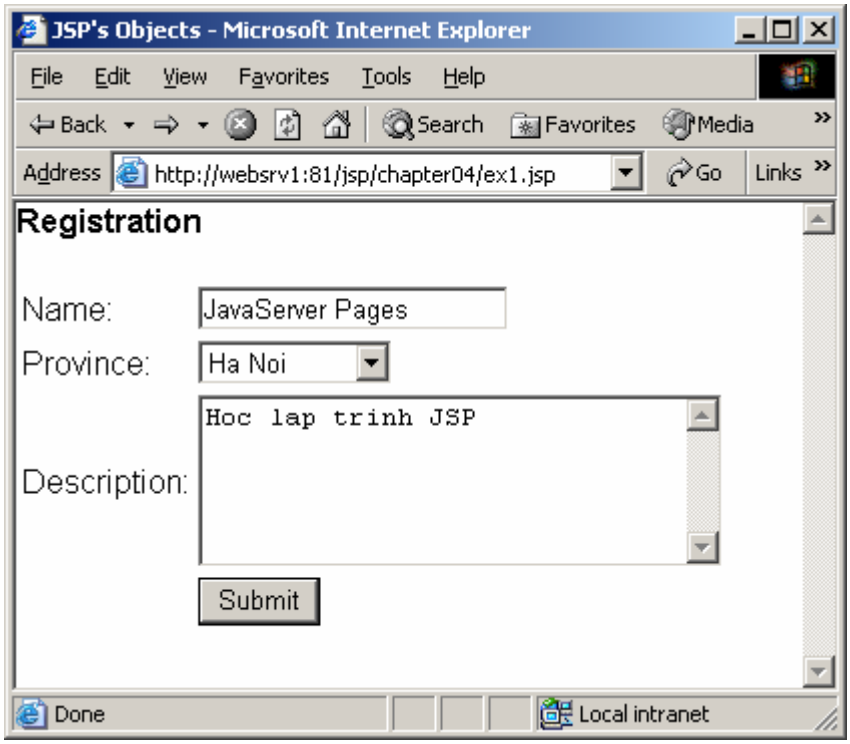
Khi muốn lấy giá trị từ một thẻ nhập liệu đệ trình (submit) từ form hay từ chuỗi QueryString, điều này có nghĩa là cho phép lấy giá trị từ client gởi lên server, đối trường hợp này bạn sử dụng đối tượng Request. Đối tượng này thuộc lớp javax.servlet.ServletRequest.

1.1. Request với từ thẻ nhập liệu

Ví dụ chúng ta khai báo trang HTML hay JSP có thẻ form, bên trong thẻ form khai báo các thẻ nhập liệu như input, select, textarea như ví dụ 4-1.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP's Objects</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" >
<tr><td></td><td>Registration</td></tr>
<form name=form1 action=ex2.jsp method=post>
<table border=0>
<tr>
<td>Name: </td><td><input type=text name=txtname></td></tr>
<tr><td>Province: </td><td><select name=province>
<option value=HAN> Ha Noi </option>
<option value=HUE> Hue </option>
<option value=HCM> Ho Chi Minh </option>
</select></td></tr>
<tr><td>Description: </td><td>
<textarea name=txtdesc cols=30 rows=5></textarea></td>
</tr>
<tr><td></td><td><input type=submit value="Submit"></td></tr>
</table>
</form>
</body>
</html>
```

Khi người sử dụng gọi trang JSP này trên trình duyệt và nhập các thông tin yêu cầu như hình 4-1.

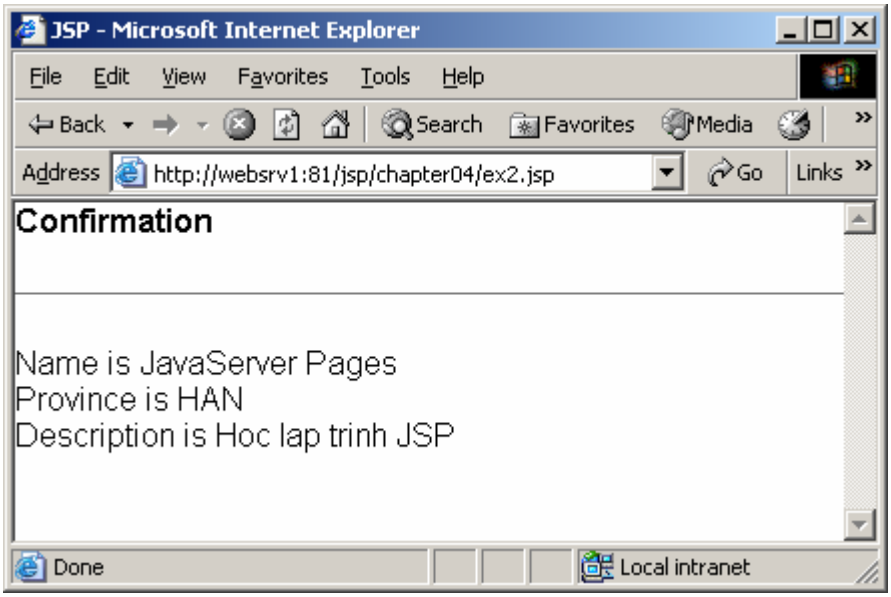


Hình 4-1: Nhập liệu

Để lấy được giá trị của các phần đã nhập trên hình 4-1, chúng ta sử dụng phương thức getParameter của đối tượng request như ví dụ 4-2.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<h4>Confirmation</h4>
<hr size=1><br>
<%
     String name=request.getParameter("txtname");
     String province=request.getParameter("province");
     String desc=request.getParameter("txtdesc");
     out.println("Name is "+ name +"<br>");
     out.println("Province is " + province +"<br>");
     out.println("Description is " + desc +"<br>");
%>
</body>
</html>
```

Kết quả trình bày như hình 4-2 nếu thông tin nhập giống như hình 4-1.



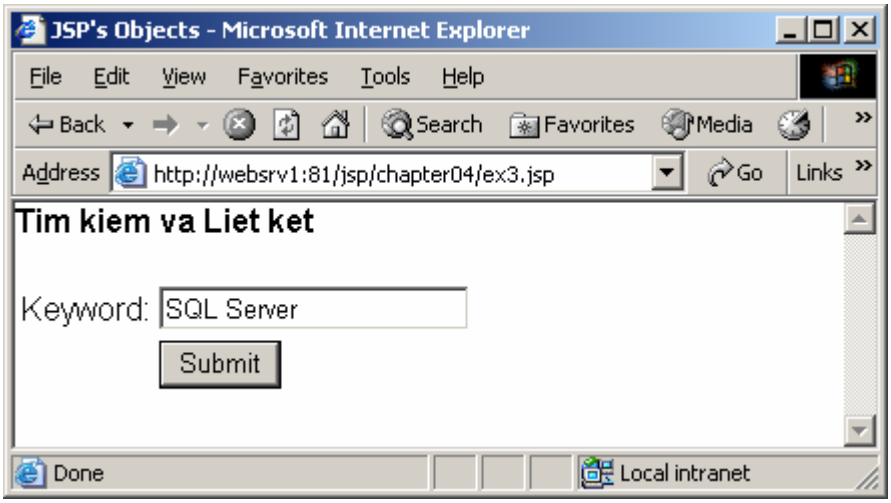
Hình 4-2: Sử dụng phương thức getParameter

1.2. Request với từ tham số trên QueryString

Tương tự như trên, trong trường hợp bạn muốn lấy các giá trị của các tham số trên QueryString, bạn cũng sử dụng phương thức getParameter của đối tượng rquest. Chẳng hạn, chúng ta khai báo trang JSP có thẻ input và sử dụng phương thức get trong thẻ form như ví dụ 4-3.

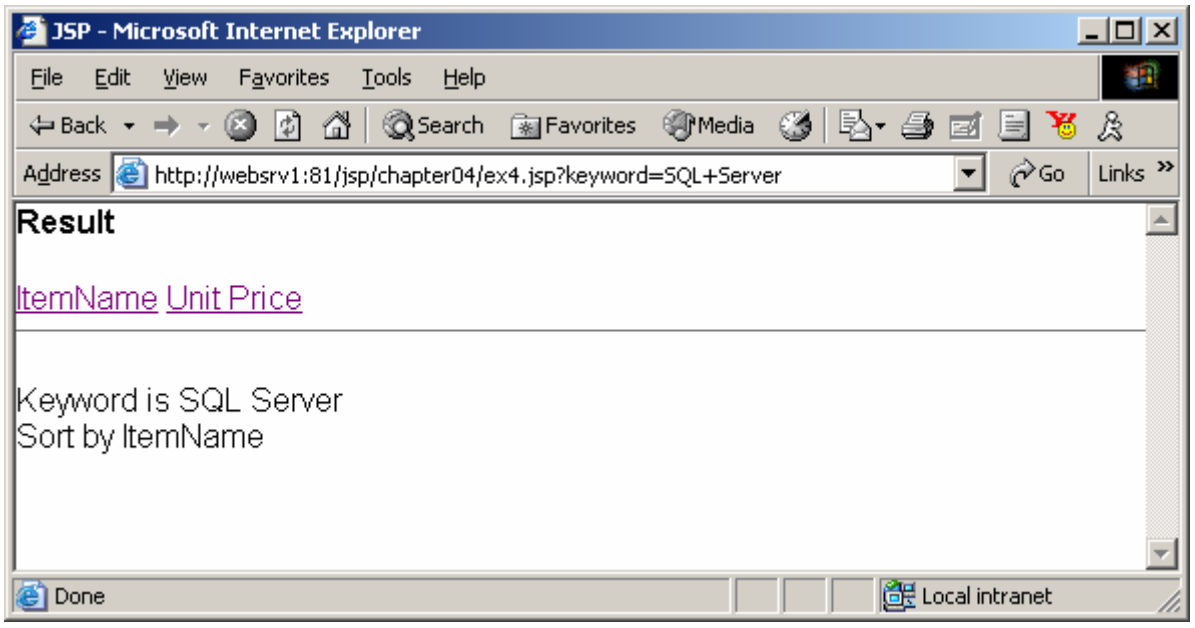
```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP's Objects</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" >
<tr><td></td><td><b>Tim kiem va Liet ket</b></td></tr>
<form name=form1 action=ex4.jsp method=get>
<table border=0>
<tr>
<td>Keyword: </td><td><input type=text name=keyword></td></tr>
<tr><td></td><td><input type=submit value="Submit"></td></tr>
</table>
</form>
</body>
</html>
```

Khi người sử dụng nhập một từ khoá nào đó và nhấn Submit, trang ex4.jsp sẽ được triệu gọi. Chẳng hạn, chúng ta nhập từ khoá là SQL Server như hình 4-3.



Hình 4-3: Phương thức GET

Khi trang ex4.jsp triệu gọi, trên chuỗi QueryString xuất hiện http://localhost:81/jsp/chapter04/ex4.jsp?keyword=SQL+Server như hình 4-4.



Hình 4-4: Lấy giá trị từ QueryString

Để lấy giá trị từ chuỗi QueryString, chúng ta sử dụng phương thức getParemeter của đối tượng request như ví dụ 4-4.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%
```

```
      // Lấy giá trị keyword từ trang ex3.jsp
      String keyword=request.getParameter("keyword");
      // Lấy giá trị sort từ trang QueryString
      String sort=request.getParameter("sort");
%>
<html>
<head>
<title>JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<h4>Result</h4>
<!--Khai báo các link để giữ lại keyword
và khai báo giá trị cho tham số sort-->
<a href="ex4.jsp?keyword=<%=keyword%>&sort=ItemName">ItemName</a>
<a href="ex4.jsp?keyword=<%=keyword%>&sort=UnitPrice">Unit Price</a>
<br>
<hr size=1><br>
<%
      // xuất giá trị lấy được
      out.println("Keyword is "+ keyword +"<br>");
      sort=(sort==null)?"ItemName":sort;
      out.println("Sort by " + sort +"<br>");
%>
</body>
</html>
```

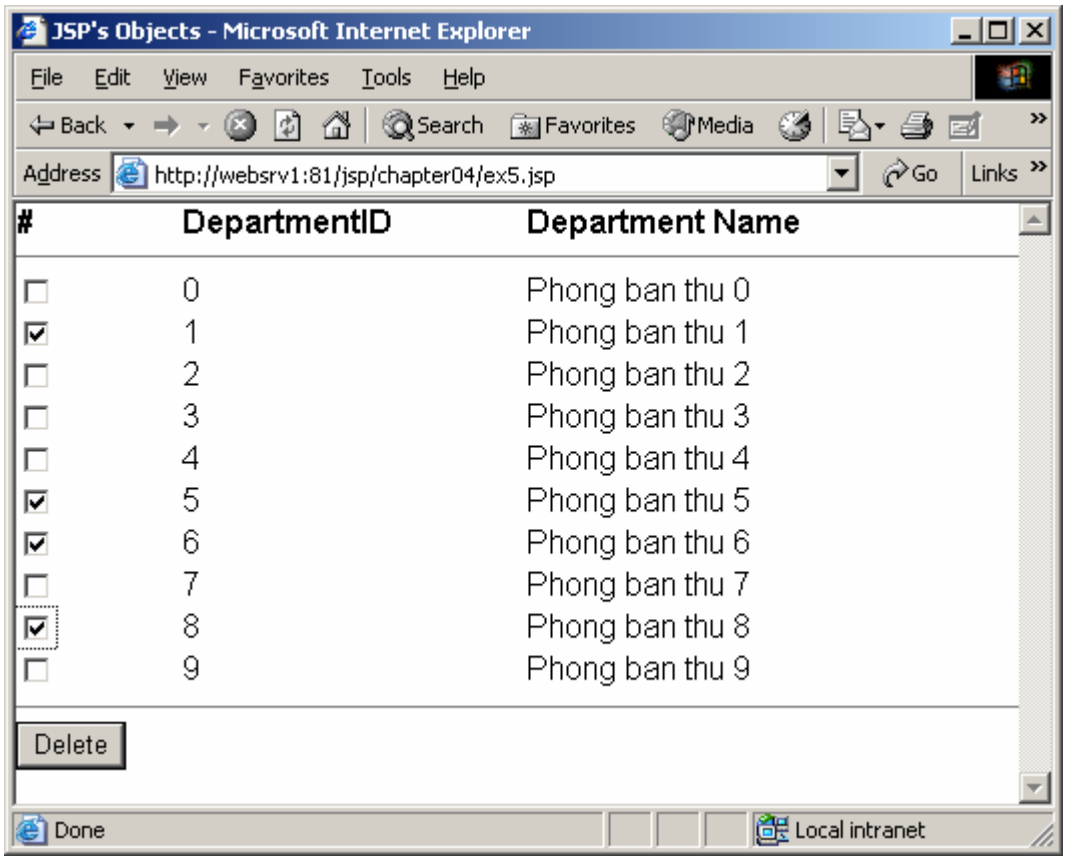
1.3. Request với mảng tham số

Nếu như có nhiều thẻ trên form hay tham số trên QueryString cùng tên, chúng ta phải sử dụng phương thức getParameterValues của đối tượng request. Ví dụ, chúng ta khai báo trang JSP có nhiều thẻ input dạng checkbox cùng tên nhưng khác giá trị như ví dụ 4-5.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP's Objects</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" >
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<form action=ex6.jsp method=post>
<tr><td><b>#</b></td><td><b>DepartmentID</b></td><td><b>Department
Name</b></td></tr>
<tr><td colspan=3><hr size=1></td></tr>
<%
int i=0;
while(i<10)
      {
      out.println("<tr><td width=50 valign=top><input type=checkbox "):
      out.println("name=chkid value='"+i+"'></td>");
      out.println("<td width=100 valign=top>"+ i +"</td>");
      out.println("<td width=150 valign=top>Phong ban thu " + i+ " </td><tr>");
      i++;
      }
%>
```

```
<tr><td colspan=3><hr size=1></td></tr>
<tr><td colspan=3><input type=submit value="Delete"></td></tr>
</table>
</form>
</body>
</html>
```

Khi triệu gọi trang JSP này trên trình duyệt, bằng cách chọn một vài phòng ban để xoá bạn có giao diện như hình 4-5.



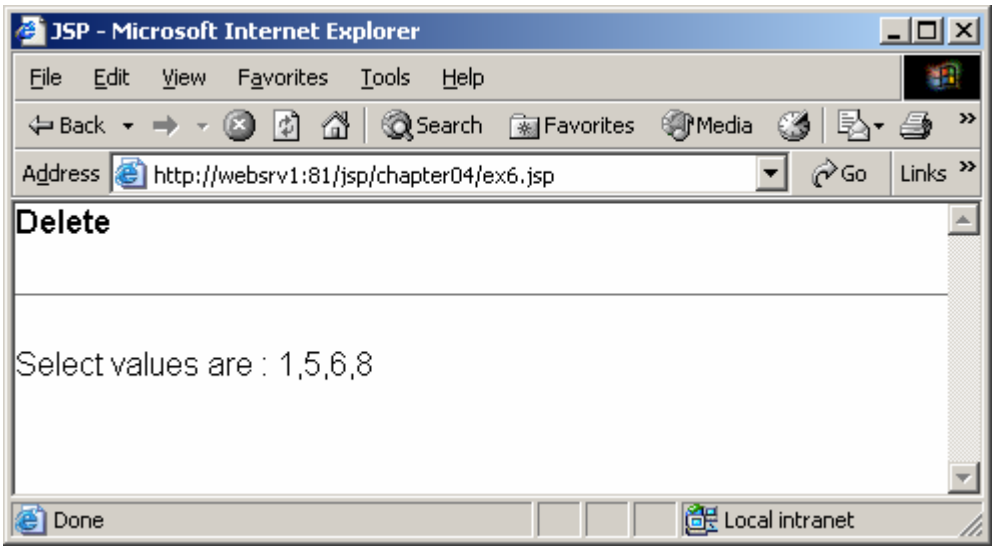
Hình 4-5: Thẻ cùng tên và khác giá trị

Như hình trên, bạn chọn các phòng ban có giá trị là 1,5,6,8, khi submit form này trang ex6.jsp sẽ được triệu gọi, để lấy được giá trị 1,5,6,8 như đã chọn bằng cách sử dụng phương thức getParameterValues, bạn khai báo như ví dụ 4-6.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<h4>Delete </h4>
<hr size=1><br>
```

```
<%
     String[] deleterecord=null;
     String delStr="";
     deleterecord=request.getParameterValues("chkid");
     if(deleterecord!=null){
        for(int k=0;k<deleterecord.length;k++){
          delStr+= deleterecord[k]+",";
        }
        delStr=delStr.substring(0,delStr.length()-1);
     }
     out.println("Select values are : " + delStr);
%>
</body>
</html>
```

Khi đó, kết quả trình bày trên trình duyệt như hình 4-6, trong thực tế bạn có thể gán giá trị này với chuỗi SQL và sử dụng phép toán IN hay NOT IN để truy vấn hay thao tác trong dữ liệu.



Hình 4-6: Kết quả lấy mảng giá trị

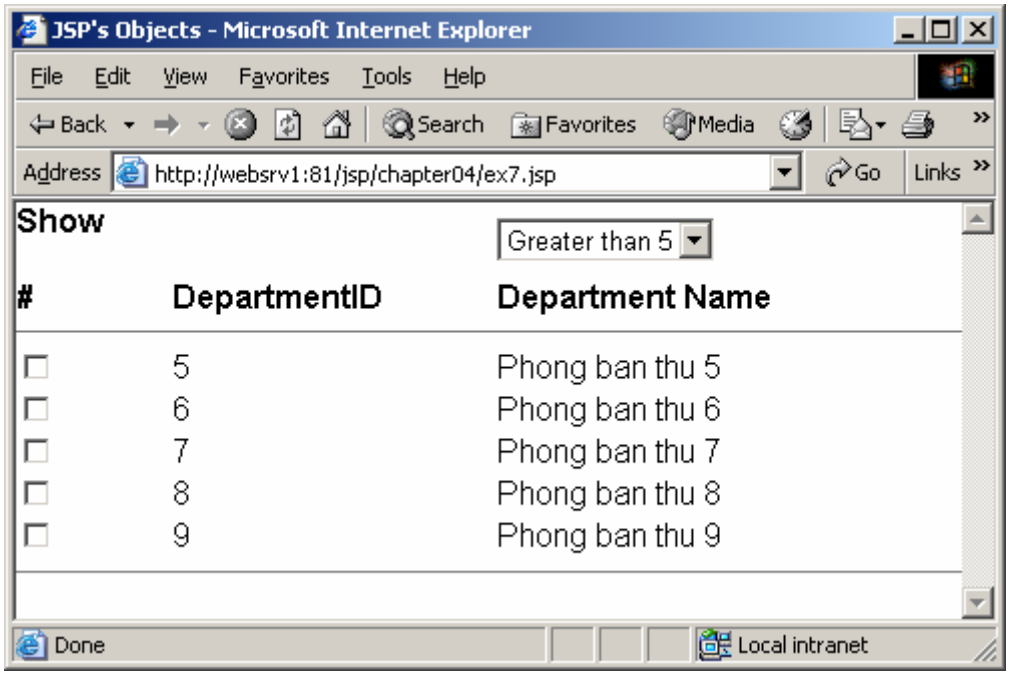
1.4. Request kết hợp với JavaScript

Ngoài các cách trên, khi làm ứng dụng thường chúng ta trình bày danh sách mẩu tin phụ thuộc vào giá trị chọn trên thẻ select hay checkbox hoặc readiobutton, bạn có thể sử dụng phương thức getParameter của đối tượng request để thực hiện ý định này như ví dụ 4-7.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%
// Lấy giá trị chọn trong thẻ select có tên selectid
int selectid=0;
// Chuyển qua số nguyên nếu có chọn
if (request.getParameter("selectid")!=null)
{
     try {
        selectid=Integer.parseInt(request.getParameter("selectid"));
```

```
      }
      catch(Exception ex)
      {
        selectid=0;
      }
}
%>
<html>
<head>
<title>JSP's Objects</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" >
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<!--Khai báo thẻ select có tên selectid-->
<form action=ex7.jsp method=post name=form1>
<tr><td colspan=2><h4>Show</h4></td><td>
<select name=selectid onchange="document.form1.submit();">
      <option value=0>Greater than 0</option>
      <option value=1>Greater than 1</option>
      <option value=2>Greater than 2</option>
      <option value=3>Greater than 3</option>
      <option value=5>Greater than 5</option>
</select>
<script>
      // Chọn lại giá trị chọn trước đó trong thẻ select có tên selectid
      for(var j=0;j<form1.selectid.length;j++)
      {
        if(form1.selectid[j].value=="<%=selectid%>")
          form1.selectid.selectedIndex=j;
      }
</script>
</td></tr>
</form>
<tr><td><b>#</b></td><td><b>DepartmentID</b></td><td>
<b>Department Name</b></td></tr>
<tr><td colspan=3><hr size=1></td></tr>
<%
// Trình bày số mẩu tin >= giá trị chọn trong thẻ select có tên selectid
int i=selectid;
while(i<10)
      {
      out.println("<tr><td width=50 valign=top>");
      out.println("<input type=checkbox name=chkid value='"+i+"'></td>");
      out.println("<td width=100>"+ i +"</td>");
      out.println("<td width=150>Phong ban thu " + i+ " </td><tr>");
      i++;
      }
%>
<tr><td colspan=3><hr size=1></td></tr>

</table>
</body>
</html>
```

Lần đầu tiên gọi đến trang ex7.jsp thì kết quả trình bày là 10 phòng ban, nếu người sử dụng chọn giá trị trong phần Show "Greater than " thì số mẩu tin trình bày lơn hơn hoặc bằng giá trị đang chọn như hình 4-7.



Hình 4-7: Trình bày mẩu tin chọn lọc

2. ĐỐI TƯỢNG RESPONSE

Khi muốn trả giá trị từ server xuống client thì sử dụng đối tượng response. Đối tượng này thuộc lớp javax.servlet.ServletResponse.

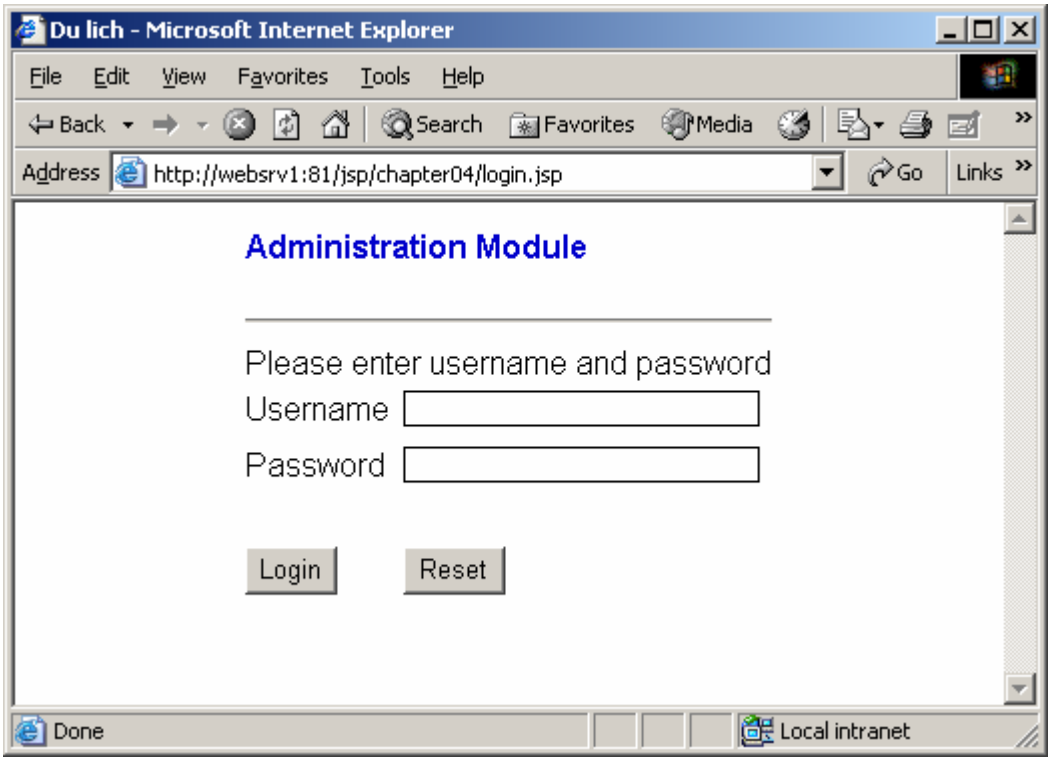
Chẳng hạn, sau khi kiểm tra giá trị username và password hợp lệ thì tự động chuyển đến trang myaccount.jsp trong trang login_authentication.jsp.

Để làm điều này, chúng ta khai báo trang login.jsp có hai thẻ nhập liệu là username và password như ví dụ 4-8.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP</title>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
<script language="JavaScript">

// Khai báo kiểm tra giá trị nhập trên trình khách
function checkinput(){
     if(document.form1.username.value==""){
        alert("Xin vui long nhap username");
        document.form1.username.focus();
        return false;
     }
     if(document.form1.password.value=="" ){
        alert("Xin vui long nhap password");
        document.form1.password.focus();
        return false;
     }

     return true;
```

```
}
</script>

</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="5" cellpadding="5">
<tr><td>
<form action=login_authentication.jsp name=form1
method=post onSubmit="return checkinput();">
     <table align=center>
      <tr>
             <td colspan="2" height="19">
              <h4><b><font color="#FF0000">
            <font color="#0000CC">
            Administration Module</font>
            </font></b></h4><hr>
             </td>
            </tr><tr>
             <td colspan=2 height="19">
              <div align="left">Please enter username and password</div>
             </td>

       </tr>
       <tr>
             <td height="19">
              <div align="left">Username</div>
             </td>
             <td height="19">
              <div align="left">
               <input type="text" name="username" size="30" maxlength="50">
              </div>
             </td>
            </tr>
            <tr>
             <td height="31">
              <div align="left">Password</div>
             </td>
             <td height="31">
              <div align="left">
               <input type="password" name="password"
                      size="30" maxlength="10">
              </div>
             </td>
            </tr>
            <tr>
             <td > </td>
             <td> </td>
            </tr><tr>
             <td ><input type=submit value="Login"></td>
             <td ><input type=reset value=Reset></td>
            </tr>
</table></form>
</body>
</html>
```

Khi triệu gọi trang login.jsp trên trình duyệt như hình 4-8, nếu bạn không nhập giá trị cho username hay password thì cửa sổ thông báo yêu cầu xuất hiện, quá trình submit lên server chỉ xảy ra khi bạn nhập cả hai giá trị hợp lệ.



Hình 4-8: Đăng nhập

Khi submit thành công, trang login_authentication.jsp được gọi, nếu username và password bạn nhập tương ứng là khang, 123456 thì trang myaccount.jsp trở đến ngược lại thì gọi lại trang login.jsp như ví dụ 4-9.

```
<%
String username="";
String password="";
username= request.getParameter("username");
password= request.getParameter("password");
if (username.equals("khang")){
     if(password.equals("123456")){
        response.sendRedirect("myaccount.jsp");
     }
     else{
        /*sai password*/
        response.sendRedirect("login.jsp");
        }
}
else{
     /*sai user*/
     response.sendRedirect("login.jsp");
}
%>
```

3. TÓM TẮT

Trong bài học này chúng tôi giới thiệu đến cho các bạn hai đối tượng chính là request và response.

Bài 5

ĐỐI TƯỢNG SESSION

TRONG JSP

Trong bài trước chúng ta sẽ làm quen các đối tượng thường sử dụng trong quá trình thiết kế trang JSP như Request, Response. Trong bài này chúng ta tiếp tục làm việc với đối tượng Session.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ *Đối tượng Session.*
- ✓ *Ứng dụng đối tượng Session.*

1. ĐỐI TƯỢNG SESSION

Khi muốn truyền giá trị từ trang trang web này sang trang web khác trong một phiên làm việc thì sử dụng dụng đối tượng Session. Đối tượng này thuộc lớp HttpSession.

http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/http/HttpSession.html

1.1. Nhận dạng một phiên làm việc

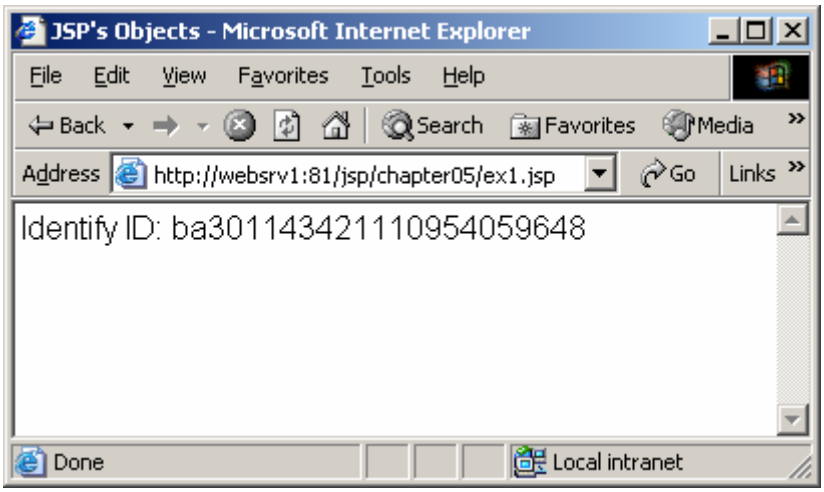
Mỗi khi có một phiên làm việc được tạo ra, thì trình chủ web sẽ cấp một định danh là một chuỗi bao gồm các ký tự và số cho phiên làm việc đó. Chẳng hạn, bạn mở một browser và gọi trang jsp từ web site, khi đó phiên làm việc được tạo ra cùng với định danh duy nhất.

Khi kết thúc phiên làm việc, định danh này bị thu lại và phân phát lại cho phiên làm việc khác mới tạo ra.

Để lấy được định danh này, bạn sử dụng phương thức getId như ví dụ 5-1.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP's Objects</title>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" >
<table>
   <tr><td>
      Identify ID: <%=session.getId()%>
   </td></tr>
   </table>
</body>
</html>
```

Khi người sử dụng gọi trang JSP này trên trình duyệt giá trị của định danh trình bày như hình 4-1.



Hình 5-1: Định danh của phiên làm việc

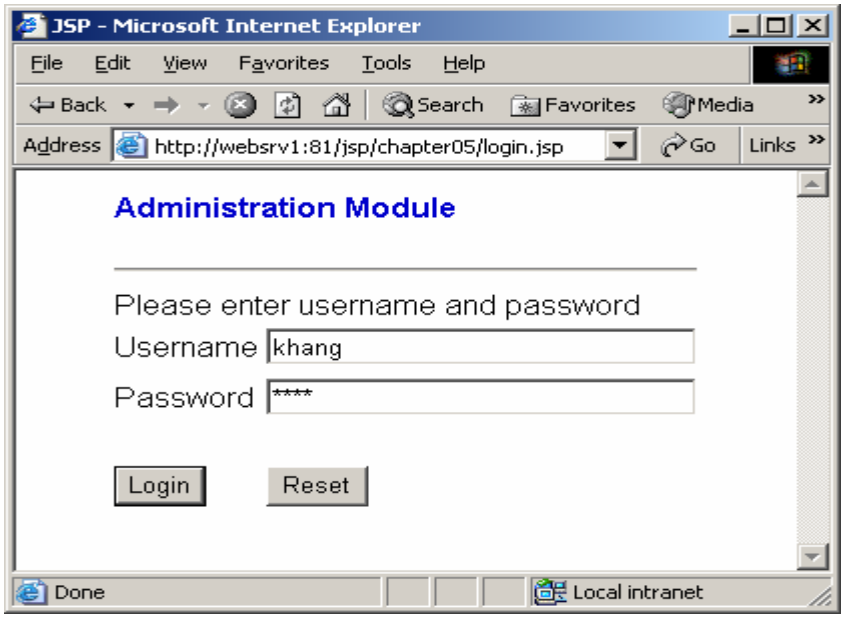
1.2. Khởi tạo và gán giá trị cho session

Khi cần truyền gt từ trang JSP này sang trang JSP khác, bằng cách khởi tạo dt Session và gán cho chúng thời gian tương ứng, sau đó bạn có thể truy cập vào đối tượng này trong trang JSP khác trong cùng phiên làm việc.

Để làm điều này, bạn sử dụng phương thức setValue với cú pháp như sau:

```
session.putValue("tên session","giá trị");
```

Ví dụ sau khi cung cấp username/password và nhấn nút Submit từ trang login.jsp như hình 5-2, trang login_authentication.jsp sẽ được triệu gọi.



Hình 5-2: Trang login.jsp

Nếu username là khang và password là 123456 thì trang myaccount.jsp sẽ được triệu gọi. Trước khi triệu gọi trang myaccount.jsp, chúng ta khởi tạo một số session dùng để sử dụng cho phiên làm việc này chẳng hạn userid, username, fullname, email như ví dụ 5-2 (login_authentication.jsp).

```
<%
String username="";
String password="";
username= request.getParameter("username");
password= request.getParameter("password");
if (username.equals("khang")){
   if(password.equals("123456")){
      session.putValue("userid","123");
      session.putValue("username",username);
      session.putValue("fullname","huukhang.com");
      session.putValue("email","admin@huukhang.com");
      response.sendRedirect("myaccount.jsp");
   }
   else{
      /*sai password*/
      response.sendRedirect("login.jsp");
      }
}
else{
   /*sai user*/
   response.sendRedirect("login.jsp");
}
%>
```

1.3. Lấy giá trị từ session

Sau khi đăng nhập thành công, những đối tượng session được tạo ra, bằng cách sử dụng phương thức getValue, bạn có thể lấy giá trị từ các session này như ví dụ 5-3 (myaccount.jsp).

Hình 5-3: Lấy giá trị từ Session

Để lấy giá trị từ các session khai báo trong trang login_authentication.jsp và trình bày trên trang myaccount.jsp, bạn khai báo như ví dụ 5-3.

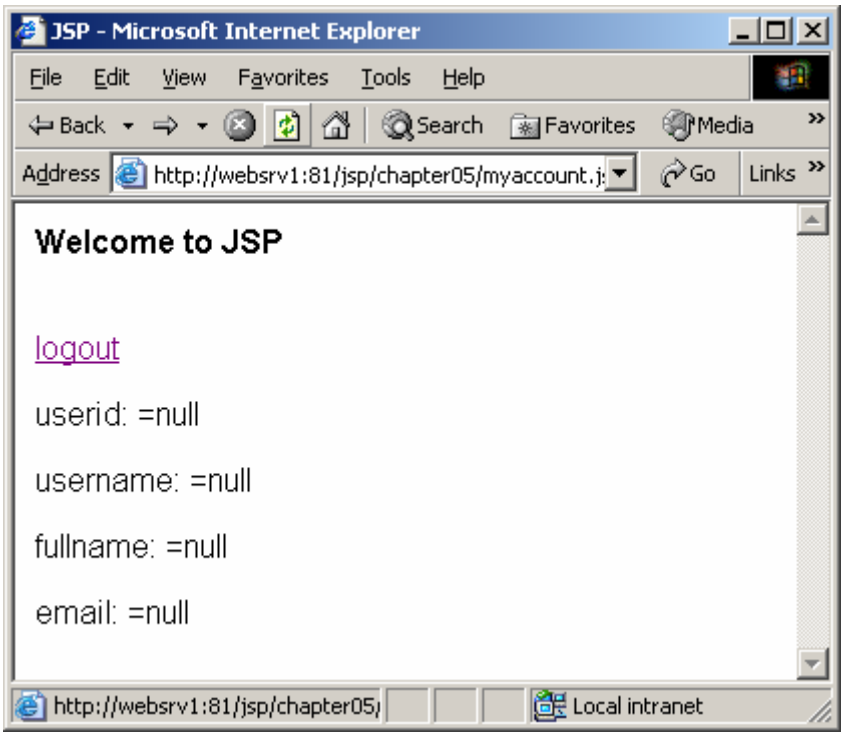
```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="5" cellpadding="5">
<tr><td>
     <h4>Welcome to JSP</h4>
</td></tr>
<tr><td><a href="logout.jsp">logout</a></td></tr>
<tr><td>userid: =<%=session.getValue("userid")%></td></tr>
<tr><td>username: =<%=session.getValue("username")%></td></tr>
<tr><td>fullname: =<%=session.getValue("fullname")%></td></tr>
<tr><td>email: =<%=session.getValue("email")%>
</td></tr>
</table>
</body>
</html>
```

1.4. Huỷ session

Sau khi không sử dụng đối tượng session, bạn cần sử dụng phương thức để huỷ session đó thày vì để chúng tồn tại, bởi ví nếu chúng không sử dụng mà vẫn tồn thì web server vẫn phải quảnl lý. Chẳng hạn, khi chúng ta logout khỏi ứng dụng website thì trang logout.jsp sẽ được tiệu gọi, ba72ng cách sử dụng các phương thức removeValue ví dụ 5-4.

```
<%
session.removeValue("userid");
session.removeValue("username");
session.removeValue("fullname");
session.removeValue("email");
response.sendRedirect("login.jsp");
%>
```

Nếu bạn triệu gọi lại trang trang myaccount.jsp trên trình duyệt, thì kết quả trình bày như hình 5-4.



Hình 5-4: Huỷ các session

Như hình trên, khi session chưa tạo ra, nếu bạn truy cập đến giá trị của chúng thì giá trị đó là null, chính vì vậy trong một số trang bắt buộc người sử dụng phải đăng nhập rồi mới sử dụng thì bạn cần phải kiểm tra session, nếu session bằng null thì trỏ đến trang login.jsp.

Chẳng hạn trong trường hợp này chúng ta có ví dụ 5-5, cho phép sử dụng khi người sử dụng đã đăng nhập, điều này có nghĩa là session có tên userid phải tồn tại.

```
     <%@ page contentType="text/html; charset=UTF-8" %>
<%
if (session.getValue("userid")==null)
     response.sendRedirect("login.jsp");
%>
<html>
<head>
<title>JSP</title>
<LINK href="style.css" rel=stylesheet>
```

```
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="5" cellpadding="5">
<tr><td>
     <h4>Welcome to JSP</h4>
</td></tr>
</table>
</body>
</html>
```

2. TÓM TẮT

Trong bài học này chúng tôi giới thiệu đến cho các bạn đối tượng chính là session, sau khi có đối tượng này, bạn có thể kết hợp với hai đối tượng request và response để xây dựng chức năng đăng nhập hệ thống.

Bài 6

PHƯƠNG THỨC VÀ CHÈN TẬP TIN

Khi xây dựng ứng dụng Web nói chung và ứng dụng JSP nói riêng, việc thống nhất hoá giao diện là điều cần thiết, chẳng hạn mọi trang JSP của ứng dụng đều phải có kích thước phần top, left, right, bottom và thân giống nhau. Tương tự như vậy mọi font chữ và kích thước cho từng phần nội dung cũng là điều bạn phải thực hiện để ứng dụng mang tính chuyên nghiệp hơn

Ngoài ra, như những bài kế tiếp chúng ta sử dụng chuỗi kết nối hay khai báo đối tượng Connection sử dụng trong mỗi trang, tất cả những điều này đều dẫn đến việc quản lý khó khăn khi cần thay đổi một trong những phần liên quan.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ *Xây dựng tập tin định dạng nội dung*
- ✓ *Thống nhất kích thước của mọi trang JSP*
- ✓ *Khai báo hàm và thủ tục dùng chung*
- ✓ *Sử dụng hàm và thủ tục dùng chung*

1. XÂY DỰNG TẬP TIN ĐỊNH DẠNG NỘI DUNG

Khi trình bày nội dung trên trang *HTML* hay trang *JSP*, để thống nhất định dạng chuỗi trong thẻ *body* hay thẻ *div* chẳng hạn bạn cần khai báo thẻ *style* trong thẻ <*head*>.

```
<style>
   A {
     COLOR: #003063;
     TEXT-DECORATION: none
   }
   A:hover {
     COLOR: #003063;
     TEXT-DECORATION: underline
   }
   A:link {
     FONT-WEIGHT: bold;
     COLOR: red;
     TEXT-DECORATION: none
   }
   A:visited {
     FONT-WEIGHT: bold;
     COLOR: black;
     TEXT-DECORATION: none
   }
   .title {
     FONT-WEIGHT: normal;
     FONT-SIZE: 22px
   }
   .text{
     FONT: 11px Arial, Helvetica, sans-serif
   }
```

```
</style>
```

Trong đó, A tương ứng với liên kết (chuỗi trong thẻ <a>) có định dạng ứng với trường hợp liên kết, di chuyển con chuột, chọn liên kết.

```
A {
  COLOR: #003063;
  TEXT-DECORATION: none
}
A:hover {
  COLOR: #003063;
  TEXT-DECORATION: underline
}
A:link {
  FONT-WEIGHT: bold;
  COLOR: red;
  TEXT-DECORATION: none
}
A:visited {
  FONT-WEIGHT: bold;
  COLOR: black;
  TEXT-DECORATION: none
}
```

Chẳng hạn, chúng ta khai báo trang *JSP* với nội dung được áp dụng với kiểu định dạng khai báo trong thẻ *style* như vú dụ 6-1.

Ví dụ 6-1: Khai báo thẻ style

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>Style trong JSP</title>
<style>
A {
     COLOR: #003063;
     TEXT-DECORATION: none
}
A:hover {
     COLOR: #003063;
     TEXT-DECORATION: underline
}
A:link {
     FONT-WEIGHT: bold;
     COLOR: red;
     TEXT-DECORATION: none
}
A:visited {
     FONT-WEIGHT: bold;
     COLOR: black;
     TEXT-DECORATION: none
}
.title {
     FONT-WEIGHT: normal;
     FONT-SIZE: 22px;
     COLOR: #003063;
}
.text{
     FONT: 11px Arial, Helvetica, sans-serif
}
</style>
</head>
<body>
```

```
        <h4>Style Tag</h4>
        <TABLE cellSpacing=0 cellPadding=0
            width="100%" border=0>
        <TR>
           <TD vAlign=top class=title>
           *** Quản Trị SQL Server 2000 ***          </TD>
        </TR>
        <TR>
           <TD class=text>
           <div align=justify>
           Tìm hiểu cách cài đặt, cấu hình, quản trị,
           backup & restore, import & export, thiết
           kế, lập trình, tự động hoá tác vụ quản trị,
           bản sao dữ liệu, bảo mật và chống thâm nhập
           dữ liệu bằng.
           <b>SQL Injection</b>.</div>
           </TD>
        </TR>
        <TR><TD><hr size=1 color=red></TD</TR>
        <TR><TD>Welcome to
        <a href="www.huukhang.com" class=>
        www.huukhang.com</a></TD
        </TR>
      </TABLE>
</body>
</html>
```

Khi triệu gọi trang *ex1.jsp* trên trình duyệt, nội dung của trang *web* được định dạng theo thẻ *style* như hình 6-1.

Hình 6-1: Áp dụng thẻ style

Tương tự như vậy khi bạn muốn thống nhất nội dung trong những thẻ khác của một trang *web* thì khai báo một định dạng trong thẻ *style*. Tuy nhiên, khi đặt tên trùng với thẻ *HTML*, mọi thẻ đó trong trang sẽ cùng chung một định dạng. Chẳng hạn, bạn khai báo định dạng cho thẻ *td* như sau:

```
TD {
     FONT: 10px Arial, Helvetica, sans-serif
}
```

Mọi nội dung trình bày trong thẻ *td* sẽ có định dạng như trên. Nếu bạn muốn có định dạng khác thì khai báo thuộc tính *class* cho thẻ *td* đó, ví dụ sử dụng định dạng khác cho thẻ *td*:

```
<td class=text>ABC</td>
```

Thay vì chuỗi *ABC* sẽ có định dạng là *FONT: 10px Arial*, *Helvetica, sans-serif* thì chúng sẽ có định dạng của *FONT: 11px Arial, Helvetica, sans-serif*.

Chú ý rằng, trong mỗi trang *web* bạn phải khai báo thẻ *style* và định nghĩa thống nhất cho các thẻ. Khi có sự thay đổi bạn phải thay đổi trong mọi trang *web*. Để sử dụng chung cho mọi trang *web* trong ứng dụng, bạn cần xây dựng một tập tin *style*, tập tin được biết đến với tên gọi *custom style sheet* (*css*).

Bất kỳ trang *web* nào trong ứng dụng, muốn áp dụng kiểu định dạng trong tập tin *css* thì khai báo liên kết tập tin *css* bằng thẻ *link*.

Ví dụ, chúng ta khai báo tập tin *style.css* bao gồm các định dạng như ví dụ 6-2.

Ví dụ 6-2: Khai báo tập tin css

```
A {
     COLOR: #003063;
     TEXT-DECORATION: none
}
A:hover {
     COLOR: #003063;
     TEXT-DECORATION: underline
}
A:link {
     FONT-WEIGHT: bold;
     COLOR: red;
     TEXT-DECORATION: none
}
A:visited {
     FONT-WEIGHT: bold;
     COLOR: black;
     TEXT-DECORATION: none
}
.title {
     FONT-WEIGHT: bold;
     FONT-SIZE: 14px;
     COLOR: #003063;
}
.text{
     FONT: 11px Arial, Helvetica, sans-serif
}
```

Sau đó trong trang *JSP*, bạn khai báo liên kết tập tin này bằng thẻ *link*, nếu muốn áp dụng định dạng này trong mỗi thẻ *HTML* bạn sử dụng thuộc tính *class* như khai báo định dạng của thẻ *style* ngay trong trang đó như ví dụ 6-3.

Ví dụ 6-3: Khai báo sử dụng tập tin css

```
<html>
<head>
<title>
     Welcome to Link Style Sheet File
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body>
     <h4>Style File</h4>
     <TABLE cellSpacing=0 cellPadding=0
          width="100%" border=0>
     <TR>
        <TD vAlign=top class=title>
        *** Quản Trị SQL Server 2000 ***          </TD>
     </TR>
     <TR>
        <TD class=text>
        <div align=justify>
        Tìm hiểu cách cài đặt, cấu hình, quản trị,
        backup & restore, import & export, thiết
        kế, lập trình, tự động hoá tác vụ quản trị,
        bản sao dữ liệu, bảo mật và chống thâm nhập
        dữ liệu bằng.
        <b>SQL Injection</b>.</div>
        </TD>
     </TR>
     <TR><TD><hr size=1 color=red></TD</TR>
     <TR><TD>Welcome to
     <a href="www.huukhang.com" class=>
     www.huukhang.com</a></TD
     </TR>
   </TABLE>
</body>
</html>
```

Triệu gọi trang *ex.jsp* trên trình duyệt như hình 6-3, màu và kích thước *font* cùng với kiểu chữa của nội dung không thay đổi so với *ex1.jsp*, bởi vì phần thẻ *style* được tách ra thành tập tin *style.css*, sau đó dùng thẻ *link* để liên kết tập tin *css* này vào trang *jsp* trở lại.

Hình 6-3: Liên kết tập tin css

Chú ý rằng, nếu khai báo thuộc tính class trong thẻ *<table>* thì những nội dung trong thẻ *<table>* sẽ có định dạng *theo* định dạng khai báo trong thuộc tính *class*. Tương tự, nếu khai báo thuộc tính *class* trong thẻ *<tr>* thì nội dung trong thẻ <tr> sẽ có định dạng giống như định dạng khai báo trong thông tin *class*.

2. THỐNG NHẤT KÍCH THƯỚC CỦA MOI TRANG JSP

Khi xây dựng ứng dụng *web* chuyên nghiệp, điều đầu tiên bạn nên quan tâm là sự thống nhất về kích thước của các phần trên trang *web*. Điều này có nghĩa là khi người sử dụng thay đổi trang *web* khi duyệt, phần *top, left, right, bottom* có kích thước như nhau.

Để làm điều này, bạn chia trang *web* ra thành 5 phần: *top*, *left*, *right, body* và *bottom*.

Phần top thường trình bày các thuộc tính như quản cáo (*baner*), *logo* (biểu tượng của công ty), *menu* (thực đơn của ứng dụng) và một số thông tin khác.

Phần *left* là thông tin về các *menu* phụ hay còn gọi là *menu* của *menu* chính, bên cạnh *menu* con này trang *web* thường có các liên kết về liên hệ, quảng cáo, *mailing list* (đăng ký *email*), gởi đến bạn bè (*send to friend*), ...

Đối với phần *right*, thường là phần giới thiệu về các thông đặc biệt và quảng cáo, chẳng hạn đối với ứng dụng bán sách, phần *right* thường là danh sách các nhóm sách bán chạy, sắp phát hành, ...

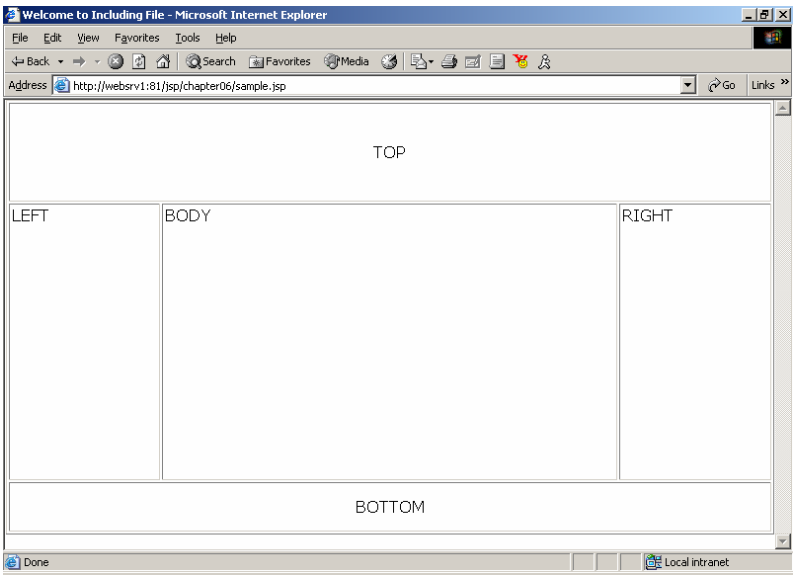
Phần *bottom* thường thông tin liên lạc của công ty, chủ nhân của *web site* và bản quyền. Ngoài ra, phần *bottom* đôi khi là danh sách các *menu* con khác.

Tóm lại, tuỳ thuộc vào ý tưởng thiết kế mỗi phần như trên bao gồm các thuộc tính mà nhà thiết kế cần trình bày sao cho phù hợp. Tuy nhiên, phần *body* là phần trình bày nội dung chính của mỗi trang *web*. Ngoài ra, tuỳ vào từng trường hợp cụ thể, trang *web* có thể không có phần *left* và *right*.

Như vậy, chúng ta sẽ chia trang *web* ra thành 5 phần, phần *body* chính là phần chính của trang *web* đó, còn 4 phần còn lại được chèn vào khi có nhu cầu.

Chẳng hạn, có những trang *web* do thông tin trình bày trong phần *body* nhiều, nên cần không gia lớn hơn, bạn có thể không cần sử dụng hai phần *left* và *right*.

Để làm điều này, trước tiên chúng ta thiết kế trang *sample.jsp* có 5 phần như hình 6-3.



Hình 6-3: Trang sample.jsp

Lưu ý:

- *Tạo một table gồm 3 hàng 3 cột và khai báo border=1 để dễ canh lề sau đó bạn có thể khai báo lại thuộc tính này bằng 0.*
- *Phần top và bottom là một hàng và merge 3 cột thành 1.*
- *Bên trong mỗi phần có thể có một hay nhiều thẻ table khác.*
- *Có thể không có phần left và right nhưng bắt buộc phần top và bottom phải có.*
- *Bạn có thể sử dụng chiều rộng của table theo kích thước tương đối (%) hay số chỉ định, đối với màn hình 600*800 thì chiều rộng thường sử dụng là 780, khi người sử dụng chọn độ phân giải của màn hình lớn hơn thì kích thước của table này không thay đổi, trong khi đó nội dung sẽ phủ đầy màn hình khi bạn khai báo kích thước theo 100%.*

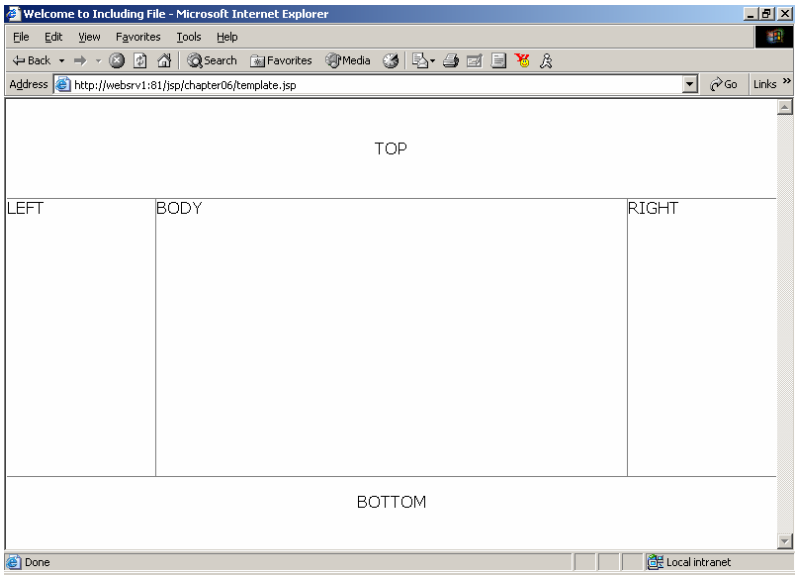
Để có giao diện như trang *sample.jsp* như trên, bạn có thể khai báo như ví dụ 6-3.

Ví dụ 6-3: Nội dung trang sample.jsp

```
<html>
<head>
<title>
     Welcome to Including File
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
```

```
</head>
<body bottomMargin=0 leftMargin=0
     topMargin=0 rightMargin=0>
     <TABLE cellSpacing=2 cellPadding=2
          width="778" border=1 align=center>
      <TR HEIGHT="100">
        <TD  Align=center colspan=3>
          TOP
        </TD>
      </TR>
      <TR HEIGHT="280">
        <TD vAlign=top width="20%">
             LEFT
        </TD>
        <TD vAlign=top width="60%">
             BODY
        </TD>
        <TD vAlign=top width="20%">
             RIGHT
        </TD>
      </TR>
      <TR HEIGHT="50">
      <TD colspan=3 align=center>
          BOTTOM
      </TD
      </TR>
     </TABLE>
</body>
</html>
```

Trong trường hợp bạn muốn có đường phân cách giữa mỗi phần bằng *image*, bạn có thể khai báo lại trang *sample.jsp* có 5 hàng và 5 cột như *template.jsp* như hình 6-4.



Hình 2-4: Phân cách có viền

Để trình bày trang *tempale.jsp* như hình 6-4, bạn khai báo nội dung trang này như ví dụ 6-4.

Ví dụ 6-4: Khai báo template.jsp

```
<html>
<head>
<title>
     Welcome to Including File
</title>
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body bottomMargin=0 leftMargin=0
     topMargin=0 rightMargin=0>
     <TABLE width="778" border=0 cellSpacing=0
      cellPadding=0 align=center>
     <TR HEIGHT="100">
       <TD  Align=center colspan=5>
          TOP
       </TD>
     </TR>
     <!---Khai báo đường phân cách--->
     <TR HEIGHT="1">
       <TD colspan=5 bgcolor=gray></TD>
     </TR>
     <TR HEIGHT="280">
       <TD vAlign=top width="150">LEFT</TD>
       <!---Khai báo đường phân cách--->
       <TD bgcolor=gray width="1"></TD>
       <TD vAlign=top width="476">BODY</TD>
       <!---Khai báo đường phân cách--->
       <TD bgcolor=gray width="1"></TD>
       <TD vAlign=top width="150">RIGHT</TD>
     </TR>
     <!---Khai báo đường phân cách--->
     <TR HEIGHT="1">
       <TD colspan=5 bgcolor=gray></TD>
     </TR>
     <TR HEIGHT="50">
     <TD colspan=5 align=center>
       BOTTOM
     </TD
     </TR>
     </TABLE>
</body>
</html>
```

Sau đó tách trang *template.jsp* này thành 5 trang khác nhau được đặt tên tương ứng là *top.htm, left.htm, right.htm* và *bottom.htm*, trong đó phần *body* tương ứng với trang *templates.jsp*.

Để khai báo chèn tập tin trong trang *jsp*, bạn sử dụng cú pháp như sau:

```
<%@include file="filename"%>
```

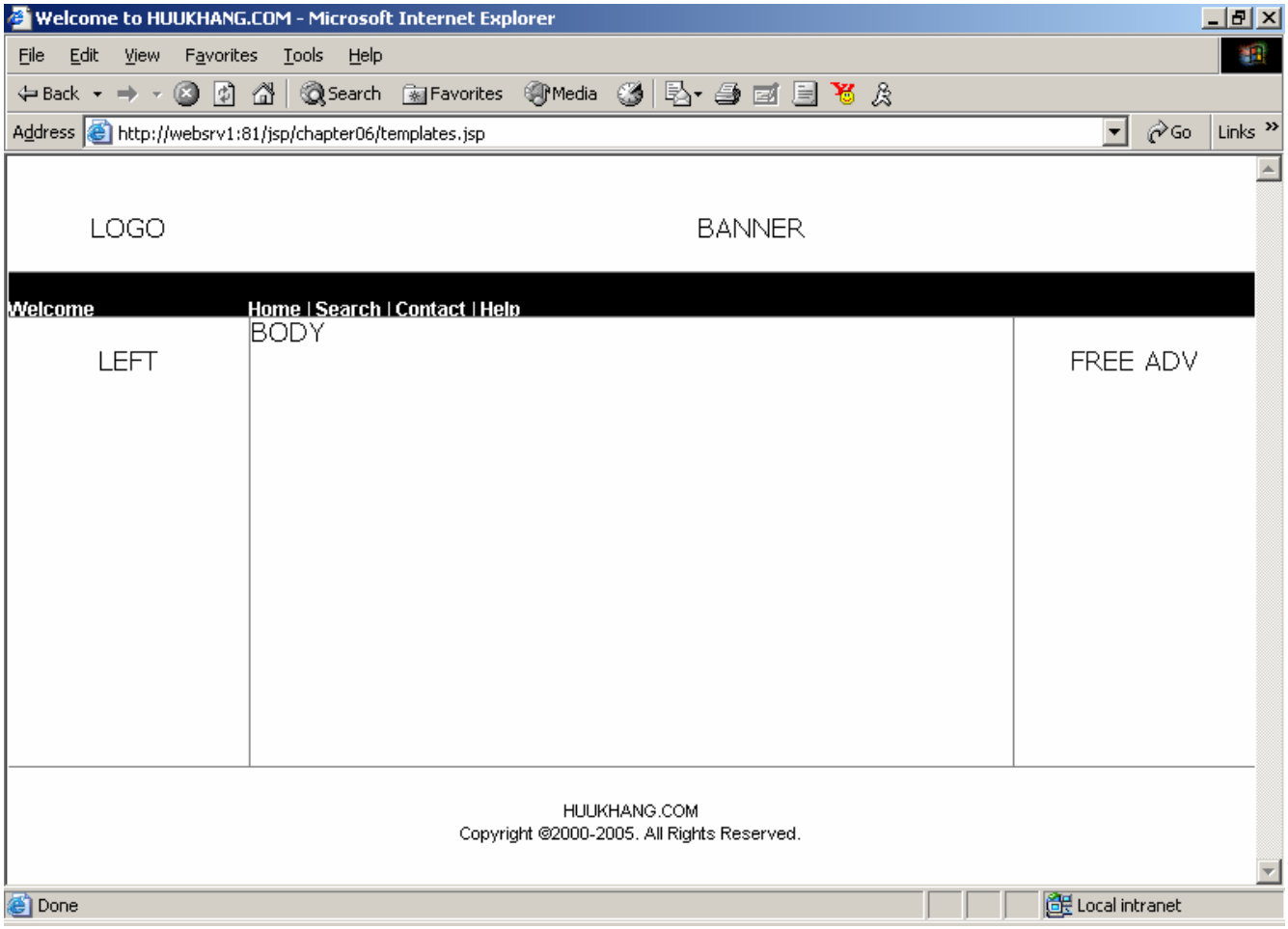
Trong đó trang *templates.jsp* khai báo chèn *top.htm, left.htm, right.htm* và *bottom.htm* như ví dụ 6-5.

Ví dụ 6-5: Khai báo chèn tập tin trong templates.jsp

```
<html>
<head>
<title>
     Welcome to HUUKHANG.COM
</title>
```

```
<LINK href="style.css" rel=stylesheet>
     <META http-equiv=Content-Type
     content="text/html; charset=utf-8">
</head>
<body bottomMargin=0 leftMargin=0
     topMargin=0 rightMargin=0>
     <TABLE width="778" border=0 cellSpacing=0
      cellPadding=0 align=center>
     <TR HEIGHT="100">
       <TD  Align=center colspan=5>
          <%@include file="top.htm"%>
       </TD>
     </TR>
     <!---Khai báo đường phân cách--->
     <TR HEIGHT="1">
       <TD colspan=5 bgcolor=gray></TD>
     </TR>
     <TR HEIGHT="280">
       <TD vAlign=top width="150">
       <%@include file="left.htm"%>
       </TD>
       <!---Khai báo đường phân cách--->
       <TD bgcolor=gray width="1"></TD>
       <TD vAlign=top width="476">BODY</TD>
       <!---Khai báo đường phân cách--->
       <TD bgcolor=gray width="1"></TD>
       <TD vAlign=top width="150">
       <%@include file="right.htm"%>
       </TD>
     </TR>
     <!---Khai báo đường phân cách--->
     <TR HEIGHT="1">
       <TD colspan=5 bgcolor=gray></TD>
     </TR>
     <TR HEIGHT="50">
     <TD colspan=5 align=center>
       <%@include file="bottom.htm"%>
     </TD
     </TR>
     </TABLE>
</body>
</html>
```

Khi triệu gọi trang *templates.jsp*, nội dung của 4 tang *left.htm*, *right.htm, top.htm, bottom.htm* chèn vào trang *templates.jsp* như hình 6-5.



Hình 6-5: Trang templates.jsp sau khi chèn

Trong đó, nội dung của trang *top.htm* định nghĩa tương tự như ví dụ 6-5-1.

Ví dụ 6-5-1: Nội dung trang top.htm

```
<TABLE width="100%" border=0 cellSpacing=0
     cellPadding=0 HEIGHT="100%" align=center>
         <TR >
           <TD width="150" Align=center>
        LOGO
      </TD>
      <TD  Align=center>
        BANNER
      </TD>
     </TR>
     <TR HEIGHT="1">
          <TD colspan=2 bgcolor=gray></TD>
     </TR>
     <TR HEIGHT="20%" bgcolor=black class=menu>
          <TD width="150" >
        Welcome
      </TD>
      <TD>
        Home | Search | Contact | Help
      </TD>
     </TR>
</TABLE>
```

Nội dung của tập tin *left.htm* được định nghĩa tương tự như ví dụ 6-5-2.

Ví dụ 6-5-2: Nội dung trang left.htm

```
<TABLE width="100%" border=0 cellSpacing=0
    cellPadding=0 HEIGHT="100%" align=center>
       <TR >
         <TD width="150" Align=center>
         LEFT
    </TD>
    </TR>
</TABLE>
```

Nếu có sử dụng trang right.htm thì nội dung của tập tin này được định nghĩa tương tự như ví dụ 6-5-3.

Ví dụ 6-5-3: Nội dung trang right.htm

```
<TABLE width="100%" border=0 cellSpacing=0
    cellPadding=0 HEIGHT="100%" align=center>
       <TR >
         <TD width="150" Align=center>
         FREE ADV
    </TD>
    </TR>
</TABLE>
```

Tương tự như vậy, trang *bottom.htm* có nội dung như ví dụ 6-5-4.

Ví dụ 6-5-4: Nội dung trang bottom.htm

```
<TABLE width="100%" border=0 cellSpacing=0
    cellPadding=0 HEIGHT="100%" align=center>
       <TR class=text>
         <TD Align=center>
         HUUKHANG.COM<br>
       Copyright ©2000-2005.
       All Rights Reserved.
    </TD>
    </TR>
</TABLE>
```

Chú ý rằng, trong mỗi trang khai báo chèn không có các thẻ đóng và mở *html*, *body* bởi khi chèn thì nội dung của tập tin được chèn sẽ được chèn vào tập tin bị chèn và trong tập tin bị chèn đã có hai thẻ này.

Kịch bản trình chủ jsp hỗ trợ các tập tin được chèn với các tên mở rộng như *htm, jsp, inc, lib, html*. Do thực chất của việc khai báo chèn là chèn đoạn mã trong tập tin chèn vào tập tin bọ chèn, trong trường hợp này trang chèn *htm* hay *jsp* đều giống nhau đó là lý do tại sao các trang chèn ở trên đều có tên mở rộng là *htm*.

Tuy nhiên, khi bạn gọi trang chèn này một mình ví dụ *tom.htm*, nếu bên trong có mã *jsp* thì mã đó không được thông dịch. Nếu những trang chèn này có nhu cầu gọi một mình thì bạn có thể chuyển chúng thành trang *jsp* thay vì *htm* như đã trình bày.

Sau khi có được trang *templates.jsp*, bạn có thể sử dụng trang này là mẫu cho các trang khác bằng cách *save as* thành các trang *jsp* khác khi lập trình. Khi khai báo chèn tập tin, bạn có thể sử dụng đường dẫn tương đối hoặc tuyệt đối của tập tin chèn so với ập tin bị chèn.

3. KHAI BÁO HÀM VÀ THỦ TỤC DÙNG CHUNG

Sau khi khai báo chèn tập tin thành một tập tin mẫu, chúng ta có thể sử dụng tập tin này để *save as* thành trang nhiều trang *jsp* khác.

Chẳng hạn, chúng ta khai báo một trang *jsp* có tên common.jsp, trong tập tin bao gồm những phương thức sử dụng nhiều lần trong các trang jsp khác. Ví dụ chúng ta khai báo phương thức nhận giá trị kiểm tra, nếu giá trị null htì trả về rỗng như ví dụ 6-6.

Ví dụ 6-6: Nội dung thủ tục getVal trong common.jsp

```
public String getVal(String strVal,String def)
     {
       return (strVal==null?def:strVal);
     }
```

Tương tự như vậy, chúng ta khai báo phương thức nhận một chuỗi cha sStr, chuỗi con oldStr và chuỗi thay thế newStr nếu tìm thấy chuỗi con, sau đó tìm kiếm trong chuỗi cha, nếu tìm thấy chuỗi con thì thay thế chuỗi thay thế như ví dụ 6-7.

Ví dụ 6-7: Nội dung thủ tục replaceString trong common.jsp

```
public String replaceString(String sStr,String oldStr,String newStr)
     {
       sStr=(sStr==null?"":sStr);
       String strVar=sStr;
       String tmpStr="";
       String finalStr="";
       int stpos=0,endpos=0,strLen=0;
       while (true)
          {
            strLen=strVar.length();
            stpos=0;
            endpos=strVar.indexOf(oldStr,stpos);
            if (endpos==-1)
                      break;
            tmpStr=strVar.substring(stpos,endpos);
            tmpStr=tmpStr.concat(newStr);

    strVar=strVar.substring(endpos+oldStr.length()>sStr.length()?endpos:e
ndpos+oldStr.length(),strLen);
            finalStr=finalStr.concat(tmpStr);
            stpos=endpos;
          }
          finalStr=finalStr.concat(strVar);
          return finalStr;
       }
```

Lưu ý rằng, khi khai báo phương thức trong JSP để dùng chung bạn sử dụng cặp dấu <%! %>

Để sử dụng phương thức này, bạn khai báo chè tập tin common.jsp vào trang jsp và gọi các phương thức này như phương thức được khai báo ngay trong trang jsp đó.

Ví dụ, chúng ta khai báo trang login.jsp cho phép người sử dụng nhập username và password, sau đó submit, trong trang login_authentication.jsp sẽ lấy hai thời gian này bằng cách sử dụng đối tượng Request. Tuy nhiên, trong trường hợp người sử dụng nhập giá trị có dấu ‘ hay giá trị là null thì kết nối cơ sở dữ liệu sẽ phát sinh lỗi.

Chính vì vậy trong trường hợp này chúng ta sử dụng phương thức getVal và replaceString trong tập tin common.jsp để thay thế nếu tìm thấy dấu ‘ trong giá trị vừa lấy ra như ví dụ 6-8.

Ví dụ 6-8: Dùng thủ tục replaceString trong common.jsp

```
<%@include file="common.jsp"%>
<%
     String username="";
     String password="";
     username= getVal(request.getParameter("username"),"");
     password= getVal(request.getParameter("password"),"");
     username= replaceString(username,"'","''");
     password= replaceString(password,"'","''");
     if (username.equals("khang")){
      if(password.equals("123456")){
        response.sendRedirect("myaccount.jsp");
      }
      else{
        /*sai password*/
        response.sendRedirect("login.jsp");
        }
     }
     else{
      /*sai user*/
      response.sendRedirect("login.jsp");
     }
%>
```

4. KẾT CHƯƠNG

Trong bài này chúng ta tập trung tìm hiểu cách khai báo tập tin dùng chung sau đó chèn vào trang *jsp*.

Ngoài ra, bạn tham khảo chi tiết các khai báo biến, phát biểu, hàm, thủ tục trong tập tin *jsp*, sau đó khai báo chèn và sử dụng các khai báo này.

Trong bài kế tiếp, chúng ta tiếp tục tìm hiểu cách tương tác với cơ sở dữ liệu.

Môn học: Java Server Pages

BÀI 7: JAVA DATABASE CONNECTIVITY

JavaSoft Inc giới thiệu JDBC (Java Database Connectivity) cho phép các ứng dụng Java truy cập vào cơ sở dữ liệu. với JDBC bạn có thể cập nhật, thêm và truy vấn dữ liệu bất kỳ.

Để kết nối cơ sở dữ liệu SQL Server trong Java, chúng ta có nhiều cách ứng với nhiều phương thức kết nối cơ sở dữ liệu, trong phần này chúng ta tập trung tìm hiểu cách kết nối cơ sở dữ liệu SQL Server từ Java bằng cầu nối JDBC-ODBC và các gói hỗ trợ khác.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ Giới thiệu JDBC API
- ✓ Các đối tượng kết nối cơ sở dữ liệu.
- ✓ Khai báo ODBC.
- ✓ Sử dụng cầu nối JDBC-ODBC
- ✓ Truy vấn, cập nhật, thêm, xoá dữ liệu

1. GIỚI THIỆU JDBC API

JDBC API (Application Programming Interface) là một tập đặt tả cho phép định nghĩa làm thế nào để Java kết nối với cơ sở dữ liệu. JDBC API được phát triển bởi công ty JavaSoft. Nhiều khái niệm của JDBC API được vay mượn từ các tài nguyên khác như ODBC (Open Database Connectivity) của Microsoft.

1.1. Trình điều khiển của JDBC

JDBC API định nghĩa để thể hiện dữ liệu như thế nào, thực thi tập lệnh để thêm, xoá hay cập nhật dữ liệu như thế nào từ yêu cầu của người sử dụng.

1.2. Sản phẩm của JDBC

Sản phẩm của JDBC bao gồm ba loại chính là gói java.sql, Test suite, cầu nối JDBC-ODBC và java.sql.package.

JDBC API là tập các interface và class cho phép kết nối cơ sở dữ liệu, chúng chứa đựng tong gói java.sql và những interface này bao gồm:

- CallableStatement: Chứa đựng các phương thức để thực thi thủ tục SQL có hỗ trợ các tham số In và Out.
- Connection: Bảo trì và theo giỏi kết nối cơ sở dữ liệu.
- DatabaseMetaData: Cung cấp các thông tin của cơ sở dữ liệu.
- Driver: Tạo ra đối tượng kết nối.
- PrepareStatement: Biên dịch phát biểu SQL trước khi thực thi.
- ResultSet: Cung cấp các phương thức để truy cập dữ liệu từ phát biểu SQL.

- ResultSetMetaData: Thu thập thông tin siêu dữ liệu phù hợp với đối tượng ResultSet cuối cùng.
- Statement: Thực thi phát biểu SQL và truy vấn dữ liệu trả về từ đối tượng ResultSet.

1.2.1. Test Suite

Test Suite dùng để kiểm tra các chức năng của trình điều khiển JDBC, nó bảo đảm rằng tất cả phương thức và lớp cài đặt trong JDBC API được cài đặt.

1.2.2. Cầu nối JDBC-ODBC

Cầu nối JDBC-ODBC là trình điều khiển JDBC cho phép chương trình Java sử dụng ODBC để kết nối cơ sở dữ liệu. Trong trường hợp này chúng ta sử dụng ODBC của hệ điều hành Windows để kết nối cơ sở dữ liệu Windows hỗ trợ.

1.3. Các trình điều khiển JDBC

JDBC Driver bao gồm 4 loại chính: JDBC-ODBC Bridge, Native API Java, JDBC Network và Native Protocol.

2. CÁC ĐỐI TƯỢNG KẾT NỐI CƠ SỞ DỮ LIỆU

Như giới htiệu ở trên, chúng ta sẽ có 3 đối tượng chình để làm việc với cơ sở dữ liệu là Connection, Statement và ResultSet.

Trước khi sử dụng các đối tượng này, bạn phải khai báo import java.sql.* và khai báo trình điều khiển JdbcODBC như sau:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

2.1. Connection

Connection là đối tượng dùng để mở kết nối cơ sở dữ liệu bằng trình điều khiển JDBC, bằng cách sử dụng phương thức DriverManager.getConnection của đối tượng Connection bạn có thể mở kết nối cơ sở dữ liệu với loại cơ sở dữ liệu cùng với các tham số khác.

2.1.1. Khai báo

Khai báo đối tượng Connection, bạn sử dụng cú pháp như sau:

```
Connection con =
DriverManager.getConnection("jdbc:odbc:Northwind","hocvien","");
```

Hay

```
Connection con = null;
con =
DriverManager.getConnection("jdbc:odbc:Northwind","hocvien","");
```

2.1.2. Tạo đối tượng Statement

Sau khi mở kết nối cơ sở dữ liệu bằng đối tượng Connection, bạn có thể sử dụng phương thức createStatement để trả về đối tượng Statement bằng cú pháp:

```
con.createStatement();
```

2.1.3. Đóng kết nối

Để đóng kết nối cơ sở dữ liệu, bạn sử dụng cú pháp:

```
con.close();
```

2.1.4. Giải phóng kết nối

Tương tự như trên, khi không có nhu cầu sử dụng đối tượng connection, bạn cần huỷ đối tượng này để giải phóng bộ nhớ mà đối tượng đã chiếm.

```
con.dispose();
```

2.2. Statement

Đối tượng Statament dùng để thực thi phát biểu SQL dùng cho hành động truy vấn và cập nhật, thay đổi xoá dữ liệu.

2.2.1. Khai báo

Khai báo đối tượng Statement, bạn sử dụng cú pháp như sau:

```
Statement st = con.createStatement();
```

Hay

```
Statement st = null;
st = con.createStatement();
```

2.2.2. Thực thi phát biểu SQL

Sau khi mở kết nối cơ sở dữ liệu bằng đối tượng Connection và khai báo đối tượng Statement, bạn có thể sử dụng phương thức executeUpdate để thực thi phát biểu SQL dạng hành động:

```
st.executeUpdate(sql);
```

Hay khai báo biến int để nhận số mẩu tin được thực thi

```
int records= st.executeUpdate(sql);
```

2.2.3. Đóng đối tượng

Để đóng kết nối cơ sở dữ liệu, bạn sử dụng cú pháp:

```
st.close();
```

2.2.4. Giải phóng đối tượng

Tương tự như trong trường hợp đối tượng connection, khi không có nhu cầu sử dụng đối tượng statement, bạn cần huỷ đối tượng này để giải phóng bộ nhớ mà đối tượng đã chiếm.

```
st.dispose();
```

2.3. ResultSet

Đối tượng ResultSet nắm giữ một tập dữ liệu cho phép bạn thao tác trên tập dữ liệu bằng các phương thức và thuộc tính của nó.

2.3.1. Khai báo

Khai báo đối tượng ResultSet, bạn sử dụng cú pháp như sau:

```
ResultSet rs = st.executeQuery(sql);
```

Hay

```
ResultSet rs = null;
// hoặc ResultSet rs;
rs= st.executeQuery(sql);
```

2.3.2. Đọc dữ liệu từ đối tượng ResultSet

Sau khi mở kết nối cơ sở dữ liệu bằng đối tượng Connection, bạn có thể sử dụng phương thức createStatement để trả về đối tượng Statement, bạn có thể đọc tập dữ liệu để điền vào đối tượng ResultSet và có thể đọc đối tượng bằng phương thức next() như sau:

```
rs.next()
```

Nếu cần kiểm tra đối tượng tồn tại mẩu tin hay không, bạn có thể sử dụng phát biểu oif như sau:

```
if(rs.next())
{

}
```

Trong trường hợp duyệt từng mẩu tin bên trong đối tượng, bạn sử dụng phát biểu while như sau:

```
while(rs.next())
{

}
```

Để đọc giá trị từ field, bạn sử dụng phương thức của đối tượng này, chẳng hạn đối với trường hợp này chúng ta sử dụng phương thức getString(fieldname) như sau:

```
string x=rs.getString("CustomerID")
```

2.3.3. Đóng đối tượng ResultSet

Để đóng đối tượng ResultSet, bạn sử dụng cú pháp:

```
rst.close();
```

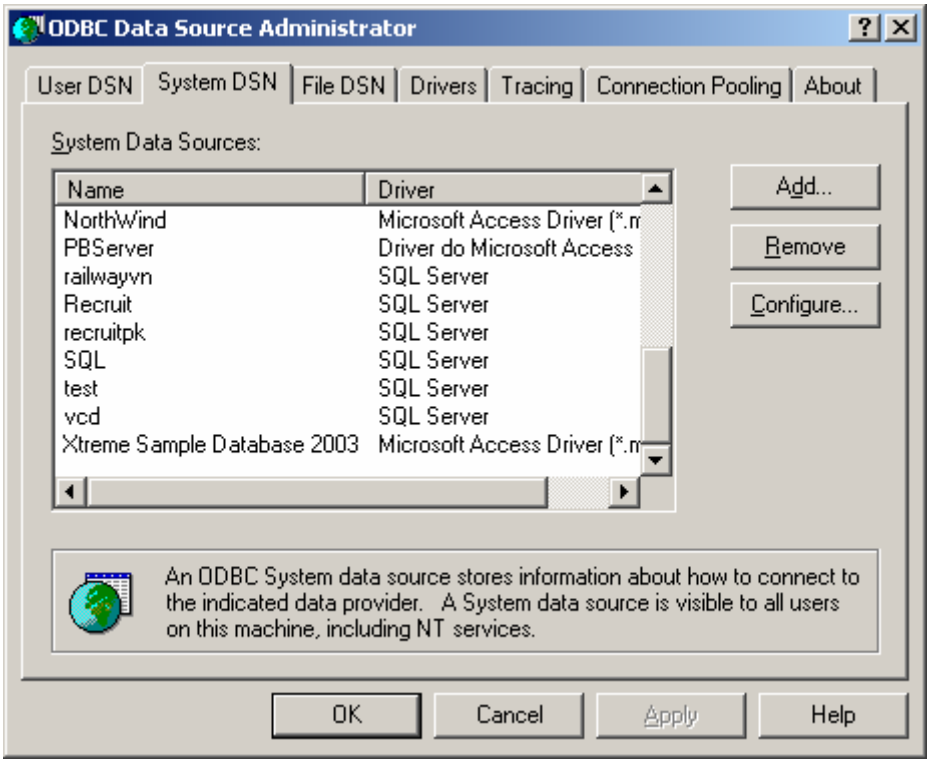
2.3.4. Giải phóng đối tượng ResultSet

Tương tự như trên, khi không có nhu cầu sử dụng đối tượng ResultSet, bạn cần huỷ đối tượng này để giải phóng bộ nhớ mà đối tượng đã chiếm.

```
rst.dispose();
```

3. KHAI BÁO ODBC

Để sử dụng ODBC trong khai báo cầu nối JDBC-ODBC, trước tiên bạn khai báo trình điều khiển trong ODBC. Để làm điều này, bạn chọn ODBC trong Cntrol panel hay từ Administrative Tools, cửa sổ xuất hiện như hình 9-1 sau:



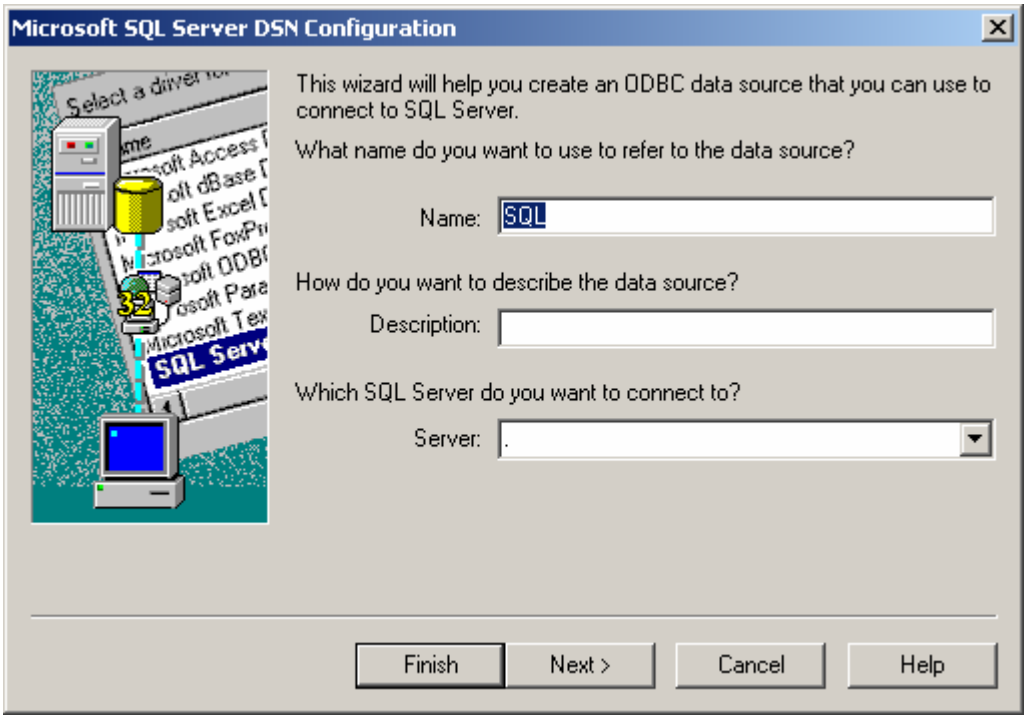
Hình 9-1: Tạo Data Source Name

Chọn vào DSN System | Add để thêm tên tham chiếu, cửa sổ xuất hiện yêu cầu bạn nhập tên (sẽ sử dụng trong khai báo) và chỉ định server (trường hợp này dùng dấu . tương đương với SQL của máy cục bộ) như hình 9-2.

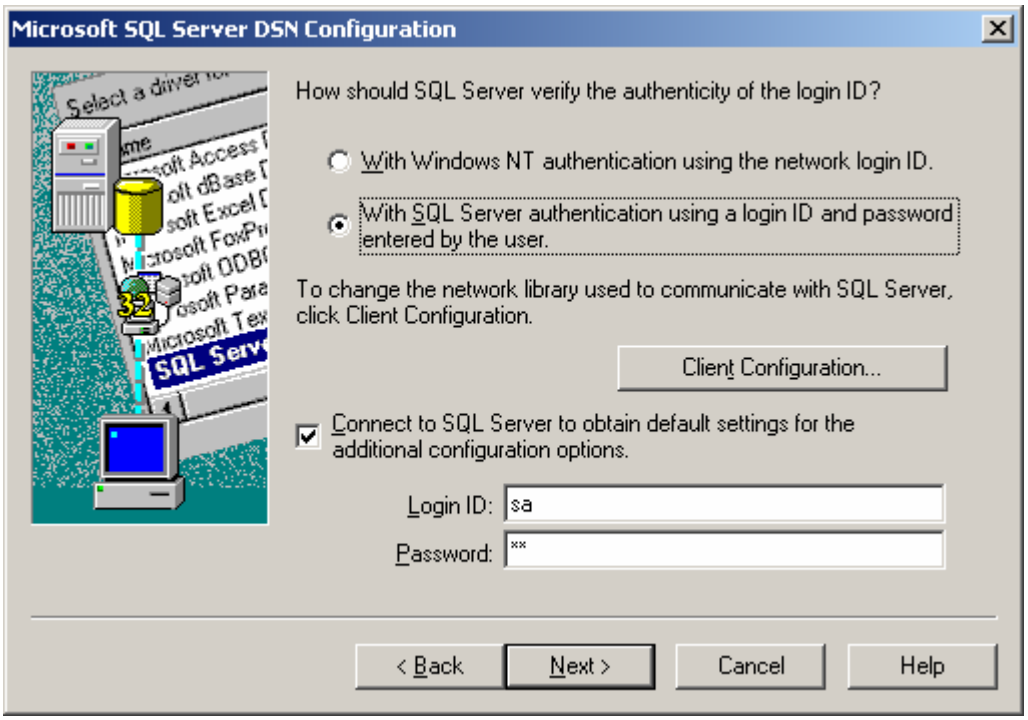
Bước kế tiếp chọn vào đặt quyền kết nối cơ sở dữ liệu bằng SQL hay Windows, trong trường hợp này chúng ta chọn vào tuỳ chọn thứ hai như hình 9-3. Khi chọn vào tuỳ chọn này, bạn cần cung cấp username và password để định nghĩa cơ sở dữ liệu SQ Server.

Nếu các tham số trên cung cấp hợp lệ, chọn Next cửa sổ kế tiếp xuất hiện như hình 9-4 yêu cầu bạn chọn tên cơ sở dữ liệu cần làm việc. Lưu ý rằng, cơ sở dữ liệu mặc định chính là cơ sở dữ liệu khai báo mặc định trong SQL Server khi người sử dụng với username đó được tạo ra.

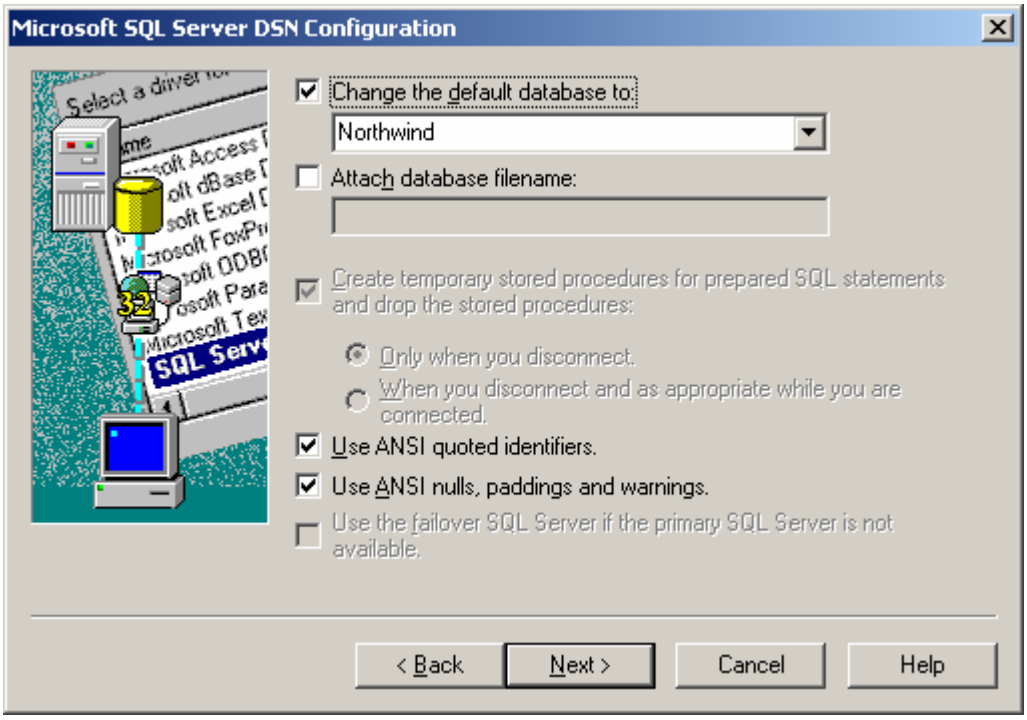
Chọn Next và đến bước cuối cùng, bạn có thể kiểm tra kết nối cơ sở dữ liệu vừa chọn có thành công hay không bằng cách nhấn vào nút Test Conneciton như hình 9-5.



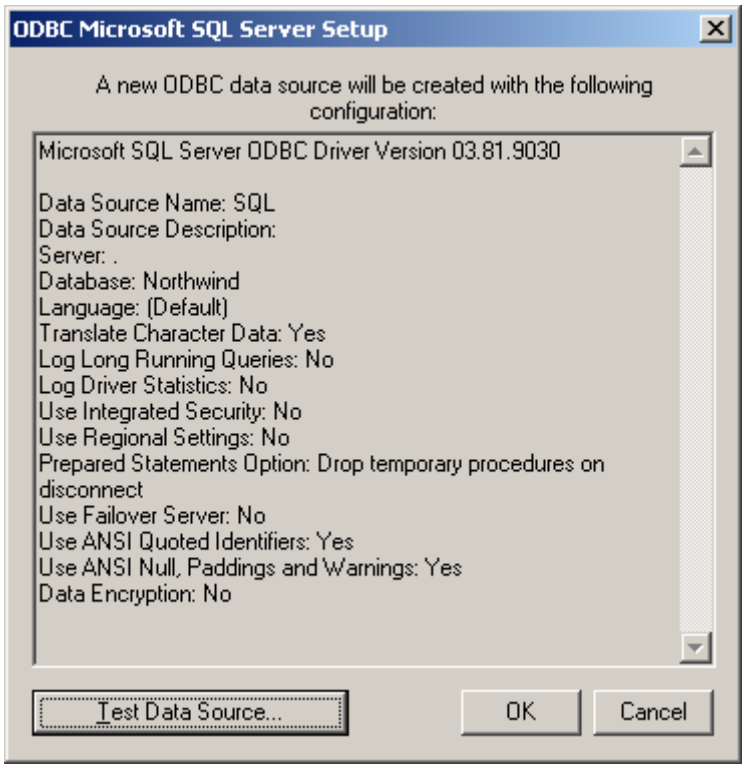
Hình 9-2: Khai báo DSN



Hình 9-3: Nhập username và password



Hình 9-4: Chọn tên cơ sở dữ liệu



Hình 9-5: Kiểm tra kết nối cơ sở dữ liệu

4. SỬ DUNG CẦU NỐI JDBC-ODBC

Sau khi khai báo xong ODBC với tên chỉ định, giả sử trong trường hợp này chúng ta chọn tên SQL với cơ sở dữ liệu Northwind cùng với tài khoản sa và password là sa.

```
try{
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  Connection con =
   DriverManager.getConnection("jdbc:odbc:SQL","sa","sa");
  ...
 }
 catch(Exception e)
 {
  System.out.println("Error : " + e);
 }
}
```

5. SỬ DUNG ĐỐI TƯỢNG CONNECTION, STATEMENT VÀ RESULTSET

5.1. Sử dụng phương thức executeQuery

Trở lại tập tin java, bạn có thể khai báo để kết nối cơ sở dữ liệu và truy vấn dữ liệu bạn có thể khai báo và sử dụng phương thức executeQuery và đối tượng Resultset như ví dụ sau:

```
...
try{
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  Connection con =
   DriverManager.getConnection("jdbc:odbc:SQL","sa","sa");
  Statement st = con.createStatement();
  String sql = "Select * from Customers";
  ResultSet rs = st.executeQuery(sql);

  while(rs.next())
  {
out.println(rs.getString("CustomerID"));
  }
  con.close();
 }
 catch(Exception e)
 {
out.println("Error : " + e);
 }
}
...
```

Chẳng hạn, chúng ta tham khảo ví dụ về chức năng đăng nhập hệ thống bằng cách sử dụng ODBC-JDBC như sau:

```
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<%
   int err=0;
   String userid="";
   String username="";
   String email="";
   String password="";
   String fullname="";
 try{
   Connection cn;
   Statement smt;
   ResultSet rst;
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
cn = DriverManager.getConnection(odbc,sysuser,syspwd);
smt = cn.createStatement();

username= request.getParameter("username");
username=replaceString(username,"'", "''");
session.putValue("username",username);
password= request.getParameter("password");
String strSQL="";
String pwd="";
strSQL="select * from tblusers where username = '" + username + "'";
rst=smt.executeQuery(strSQL);

if (rst.next()){
     userid= rst.getString("UserID");
     pwd=rst.getString("Password");
     fullname= rst.getString("Fullname");
     email= rst.getString("Email");
     if(pwd.equals(password)){
        session.putValue("userid",userid);
        session.putValue("email",email);
        session.putValue("fullname",fullname);
        response.sendRedirect("myaccount.jsp");
     }
     else{
        /*sai password*/
        session.putValue("userid","0");
        response.sendRedirect("login.jsp");
     }
   }
else{
   /*sai user*/
   session.putValue("userid","-1");
   response.sendRedirect("login.jsp");
}
smt.close();
cn.close();
}
catch (Exception e){
   /*sai ket noi*/
   session.putValue("userid","-2");
   out.println(e);
   response.sendRedirect("login.jsp");
}


%>
```

Từ trang login.jsp, người sử dụng nhập username/password và nhấn OK, bạn triệu gọi trang login_authentication.jsp, bằng cách sử dụng 3 đối tượng Connection, Statement và Resultset để kết nối cơ sở dữ liệu và kiểm tra. Nếu username/password hợp lệ thì trang myaccount.jsp sẽ được triệu gọi, nếu không chúng ta sẽ trở về trang login.jsp.

5.2. Sử dụng phương thức executeNonQuery

Trong trường hợp thêm hay cập nhật, thêm, xoá dữ liệu hay thực thi SP bạn sử dụng phương thức executeUpdate, bạn có thể khai báo như ví dụ sau:

```
try{
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  Connection con =
   DriverManager.getConnection("jdbc:odbc:SQL","sa","sa");
  Statement st = con.createStatement();
  String sql = "Delete from Customers where Country='Vietnam'";
  int records = st.executeUpdate(sql);
```

```
   out.println("Effected records : " + records.ToString());
  con.close();
 }
 catch(Exception e)
 {
out.println("Error : " + e);
 }
}
```

Chẳng hạn, chúng ta khai báo trang them.jsp để thêm mẩu tin vào bảng tblCategories như sau:

```
%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<%
 try{
   Connection cn;
   Statement smt;
   ResultSet rst;
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   cn = DriverManager.getConnection(odbc,sysuser,syspwd);
   smt = cn.createStatement();
   String strSQL="";
   strSQL="insert into tblCategories values('ABC')";
   smt.executeNonQuery(strSQL);
   smt.close();
   cn.close();
   }
   catch (Exception e){
   /*sai ket noi*/
   out.println(e);
}
%>
```

6. KẾT LUÂN

Trong bài này, chúng ta tìm hiểu JDBC API, các đối tượng làm việc với cơ sở dữ liệu, khai báo ODBC và các phương thức của các đối tượng Connection, Statement và ResultSet.

Trong bài kế tiếp chúng ta tiếp tục tìm hiểu về các cách trình bày dữ liệu bằng 3 đối tượng trên với cơ sở dữ liệu SQL Server.

Môn học: Java Server Pages

BÀI 8: THÊM, XOÁ, CẬP NHẬT DỮ LIỆU

Sau khi chúng ta đã làm quen với JDBC (Java Database Connectivity), bạn có thể sử dụng cầu nối JDBC-ODBC hay các gói kết nối khác để xoá, cập nhật, thêm và truy vấn dữ liệu bất kỳ.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ Thêm dữ liệu
- ✓ Xoá dữ liệu
- ✓ Cập nhật dữ liệu
- ✓ Thực thi thủ tục nội tại của SQL Server

1. THÊM DỮ LIỆU

Để thêm mẩu tin vào bảng dữ liệu SQL Server, bạn sử dụng phương thức executeUpdate như ví dụ trang insert.jsp sau:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<html>
<head>
<title>Thêm mẩu tin vào cơ sở dữ liệu trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<%
 boolean isok=false;
 try{
  Connection cn;
  Statement smt;
  ResultSet rst;
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  cn = DriverManager.getConnection(odbc,sysuser,syspwd);
  smt = cn.createStatement();
  String strSQL="";
  strSQL="insert into tblCategories values('Database')";
  smt.executeUpdate(strSQL);
  isok=true;
  smt.close();
  cn.close();
  }
  catch (Exception e)
  {
     /*sai ket noi*/
     out.println(e);
     isok=false;
  }
if( isok)
  out.println("Thêm mẩu tin thành công");
else
  out.println("Thêm mẩu tin không thành công ");
```

```
%>

</body>
</html>
```

2. CẬP NHẬT DỮ LIỆU

Trong trường hợp cập nhật dữ liệu cũng tương tự như thêm mẩu tin, bạn sử dụng phương thức executeUpdate để thực thi phát biểu SQL dạng Update. Chẳng hạn, chúng ta tham khảo ví dụ về cập nhật dữ liệu như trang update.jsp sau:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<html>
<head>
<title>Cập nhật mẩu tin vào cơ sở dữ liệu trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<%
  boolean isok=false;
  int j=0;
 try{
  Connection cn;
  Statement smt;
  ResultSet rst;
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  cn = DriverManager.getConnection(odbc,sysuser,syspwd);
  smt = cn.createStatement();
  String strSQL="";
  strSQL="update tblCategories set CateName='Databases' "
  strSQL+= " where CateID=41";
  j= smt.executeUpdate(strSQL);
  isok=true;
  smt.close();
  cn.close();
  }
  catch (Exception e)
  {
     /*sai ket noi*/
     out.println(e);
     isok=false;
  }
if( isok)
  out.println("Cập nhật " + j + " mẩu tin thành công");
else
  out.println("Cập nhật mẩu tin không thành công ");
%>

</body>
</html>
```

3. XOÁ MẨU TIN

Trong trường hợp xoá mẩu tin trong cơ sở dữ liệu bạn cũng sử dụng phương thức executeUpdate như trang delete.jsp sau:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
```

```
<%@ include file="common.jsp"%>
<html>
<head>
<title>Xoá mẩu tin vào cơ sở dữ liệu trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<%
  boolean isok=false;
  int j=0;
 try{
  Connection cn;
  Statement smt;
  ResultSet rst;
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  cn = DriverManager.getConnection(odbc,sysuser,syspwd);
  smt = cn.createStatement();
  String strSQL="";
  strSQL="delete from tblCategories where CateID>=30 ";
  j= smt.executeUpdate(strSQL);
  isok=true;
  smt.close();
  cn.close();
  }
  catch (Exception e)
  {
     /*sai ket noi*/
     out.println(e);
     isok=false;
  }
if( isok)
  out.println("Xoá " + j + " mẩu tin thành công");
else
  out.println("Xoá mẩu tin không thành công ");
%>

</body>
</html>
```

4. THƯC THI THỦ TUC NÔI TAI CỦA SQL SERVER

4.1. Thủ tục không có giá trị trả về

Bạn có thể thực thi một thủ tục nội tại **không có giá trị trả về** của SQL Server cũng như một phát biểu SQL dạng hành động, chẳng hạn chúng ta có thủ tục thêm mẩu tin vào tblCategories như trang procedure.jsp sau:

```
create proc AddCategories
  @name nvarchar(50)
as
  Insert into tblCategories values(@name)
```

Lưu ý rằng, khi gọi thủ tục này trong SQL Server bạn sử dụng cú pháp như sau:

```
  AddCategories 'Phan Tich'
```

Sau đó, từ trang JSP bạn khai báo để thực thi thủ tục này như sau:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<html>
<head>
<title>Thêm mẩu tin vào cơ sở dữ liệu trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<%
  boolean isok=false;
  int j=0;
 try{
  Connection cn;
  Statement smt;
  ResultSet rst;
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  cn = DriverManager.getConnection(odbc,sysuser,syspwd);
  smt = cn.createStatement();
  String myName ="Phan Tich";
  String strSQL="";
  strSQL="AddCategories '" + myName +"'";
  j= smt.executeUpdate(strSQL);
  isok=true;
  smt.close();
  cn.close();
  }
  catch (Exception e)
  {
    /*sai ket noi*/
    out.println(e);
    isok=false;
  }
if( isok)
  out.println("Thêm " + j + " mẩu tin thành công");
else
  out.println("Thêm mẩu tin không thành công ");
%>

</body>
</html>
```

4.2. Thực thi thủ tục có giá trị trả về

Bạn có thể thực thi một thủ tục nội tại **có giá trị trả về** của SQL Server cũng như một phát biểu SQL dạng Select, chẳng hạn chúng ta có thủ tục thêm mẩu tin vào tblCategories và lấy số tự động là mã của Category đó phát sinh như trang procedurewithvalue.jsp sau:

```
create proc getCategoryID
  @name nvarchar(50)
as
  insert into tblCategories values(@name)
  select @@identity as No
```

Lưu ý rằng, khi gọi thủ tục này trong SQL Server bạn sử dụng cú pháp như sau:

```
getCategoryID 'Phan Tich He Thong'
```

Thì kết quả trả về là số tự động tương ứng với mã Category. Sau đó, từ trang JSP bạn khai báo để thực thi thủ tục này bằng đối tượng ResultSet như trang procedurewithvalue.jsp sau:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<html>
<head>
<title>Thêm mẩu tin vào cơ sở dữ liệu trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<%
  boolean isok=false;
  int j=0;
 try{
  Connection cn;
  Statement smt;
  ResultSet rst;
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  cn = DriverManager.getConnection(odbc,sysuser,syspwd);
  smt = cn.createStatement();
  String myName ="Phan Tich He Thong";
  String strSQL="";
  strSQL="getCategoryID '" + myName +"'";
  rst= smt.executeQuery(strSQL);
  if (rst.next())
    j=rst.getInt("No");
  isok=true;
  smt.close();
  cn.close();
  }
  catch (Exception e)
  {
    /*sai ket noi*/
    out.println(e);
    isok=false;
  }
if( isok)
  out.println("CategoryID is " + j);
else
  out.println("Thêm mẩu tin không thành công ");
%>

</body>
</html>
```

4.3. Thực thi thủ tục có giá trị trả về là tập dữ liệu

Bạn có thể thực thi một thủ tục nội tại **có giá trị trả về là một tập dữ liệu** của SQL Server cũng như một phát biểu SQL dạng Select, chẳng hạn chúng ta có thủ tục thực thi những hành động nào đó, sau đó liệt kê mẩu tin của bảng tblCategories như trang procedurewithresultset.jsp sau:

```
create proc getResultset
  @id int
```

```
as
   --Nhiều tính toán và hành động ơ đây
   select * from tblCategories where CateID>@id
```

Lưu ý rằng, khi gọi thủ tục này trong SQL Server bạn sử dụng cú pháp như sau:

```
   getResultset 20
```

Thì kết quả trả về là danh sách mẩu tin có mã Category lớn hơn 20. Sau đó, từ trang JSP bạn khai báo để thực thi thủ tục này bằng đối tượng ResultSet như trang procedurewithresulset.jsp sau:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<html>
<head>
<title>Liệt kê mẩu tin vào cơ sở dữ liệu trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<%
   boolean isok=false;
   int j=0;
 try{
   Connection cn;
   Statement smt;
   ResultSet rst;
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   cn = DriverManager.getConnection(odbc,sysuser,syspwd);
   smt = cn.createStatement();
   String strSQL="";
   strSQL="getResultset '" + j +"'";
   rst=smt.executeQuery(strSQL);
   while(rst.next())
      {
      out.println(rst.getString("CateID")+"-");
      out.println(rst.getString("CateName"));
      out.println("<br>");
      }
   isok=true;
   smt.close();
   cn.close();
   }
   catch (Exception e)
   {
      /*sai ket noi*/
      out.println(e);
      isok=false;
   }
if(!isok)
   out.println("Liệt kê mẩu tin không thành công ");
%>

</body>
</html>
```

5. LIÊT KÊ DỮ LIÊU THEO TUỲ CHON

Bằng cách liệt kê danh sách của bảng có quan hệ cha, cho phép người sử dụng chọn một phần tử, bạn có thể liệt kê danh sách các mẩu tin có quan hệ trong bảng có quan hệ N. Để làm điều này, trước tiên bạn liêt kê danh sách trong bảng tblCategories trên thẻ select, sau đó mỗi lần người sử dụng chọn một CategoryID thì bạn liêt kê danh sách các mẩu tin trong bảng tblSubCategories như ví dụ trang chooseandshow.jsp.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<html>
<head>
<title>Liệt kê mẩu tin trong JSP</title>
<LINK href="style.css" rel=stylesheet>
<LINK href="newstyle.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type"
CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<br>
<form action=chooseandshow.jsp method=post name=form1>
Category: <select name=cateid onchange="document.form1.submit();">
<option value=''></option>
<%
//Lấy giá trị từ thẻ select có tên cateid nếu có submit
String cateid=request.getParameter("cateid");
Connection cn;
Statement smt;
ResultSet rst=null;
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
cn = DriverManager.getConnection(odbc,sysuser,syspwd);
smt = cn.createStatement();
try{
   // Liệt kê danh sách Category trong thẻ select có tên cateid
   String strSQL="select * from tblCategories";
   rst=smt.executeQuery(strSQL);
   while(rst.next())
      {
      String id=rst.getString("CateID") ;
      out.println("<option value='" + id + "' ");
      if(cateid.toString().equals(id))
        out.println(" selected");
      out.println(">");
      out.println(rst.getString("CateName"));
      out.println("</option>");
      }
   }
   catch (Exception e)
   {
      /*sai ket noi*/
      out.println(e);
   }
out.println("</select></form></br>");
rst.close();
//Nếu có submit thì liệt danh sách subcategory theo cateid
if(cateid!=null)
{
 int j=0;
 try{

   String strSQL="";
   strSQL="select * from tblSubCategories Where cateid='" + cateid +"'";
```

```
   rst=smt.executeQuery(strSQL);
   while(rst.next())
      {
      j++;
      out.println(rst.getString("SubCateID")+"-");
      out.println(rst.getString("SubCateName"));
      out.println("<br>");
      }
   }
   catch (Exception e)
   {
      /*sai ket noi*/
      out.println(e);
   }
  rst.close();
  if(j==0)
   out.println("SubCategory not found!");
}
smt.close();
cn.close();
%>
</body>
</html>
```

6. KẾT LUẬN

Trong bài này, chúng ta tìm hiểu trình bày dữ liệu, thêm, xoá, cập nhật và thực thi thủ tục nội tại SQL Server bằng 3 đối tượng Connection, Statement và ResultSet với cơ sở dữ liệu SQL Server.

Môn học: Java Server Pages

BÀI 9: THÊM, XOÁ, CẬP NHẬT DỮ LIỆU

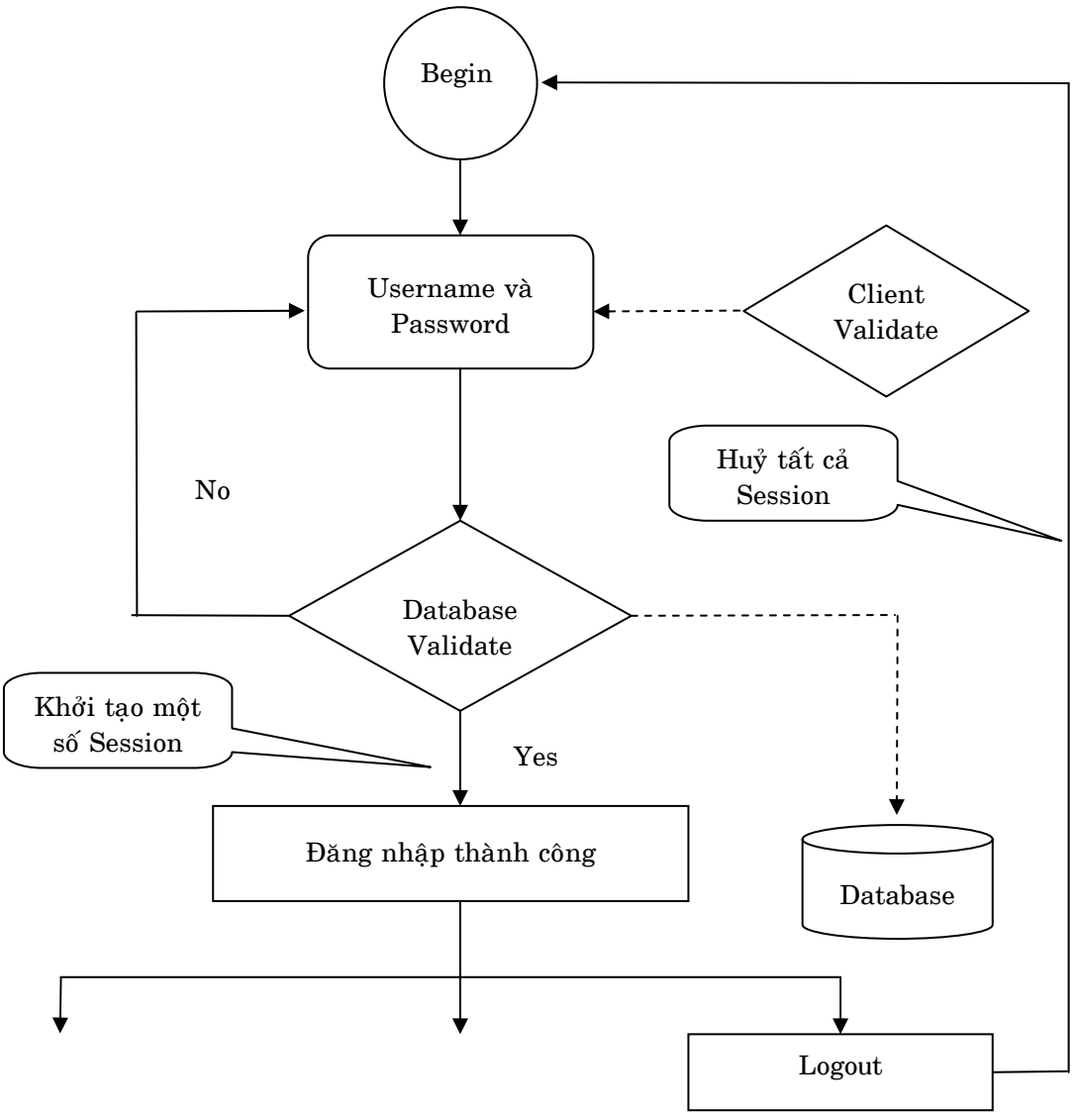
Sau khi chúng ta đã làm quen với thao tác cơ sở dữ liệu, trong bài này chúng ta tiếp tục tìm hiểu xây dựng chức năng đăng nhập hệ thống ứng dụng.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ Mô hình chức năng đăng nhập
- ✓ Login, Trang chính và Logout

1. MÔ HÌNH CHỨC NĂNG ĐĂNG NHẬP

Rong mọi ứng dụng Web, nếu cho phép người sử dụng đăng nhập hệ thống bạn đều phải sử dụng trang login, trang chính sau khi đăng nhập thành công và trang logout theo mô hình sau.



1. Người sử dụng nhập username và password.
2. Nhấn nút Login.
3. Kiểm tra trên client (JavaScript) người sử dụng đã nhập username (email) và Password (nếu bắt buộc phải có Password). Nếu username và password hợp lệ thì nhảy đến bước 4, ngược lại trở về bước 1.
4. Kết nối cơ sở dữ liệu và kiểm tra username có tồn tại hay không?
   - Nếu tồn tại thì so sánh Password
     - Nếu trùng Password thì khởi tạo một số Session cần thiết và trỏ đến trang chính của tài koản người sử dụng.
     - Ngược lại thì trở về bước 1 với thông báo "Sai password".
   - Trong trường hợp không tồn tại thì trở về bước 1 với thông báo "Không tồn tại username".
5. Sau khi đăng nhập thành công, nếu người sử dụng Logout thì huỷ tất cả các Session đã tạo ra và trở về bước 1.

Lưu ý: Dựa vào đâu chúng ta trình bày chuỗi thông báo trong trang login.

1. Lần đầu tiên hay sau khi logout nếu người sử dụng gọi trang login thì chúng ta thông báo "Xin vui lòng cung cấp username và password để đăng nhập hệ thống".
2. Trong trường hợp sai password mà gọi lại trang login thì chúng ta thông báo " Sai password, xin vui lòng nhập lại password".
3. Tương tự như vậy, khi kiểm tra không tìm thấy username mà gọi lại trang login thì chúng ta thông báo " Không tồn tại username, xin vui lòng nhập lại username".

Như vậy, chúng ta dựa và session có tên userid, nếu đăng nhập thành công thì session có giá trị là số id của người sử dụng. Trong trường hợp sai password thì chúng ta gán giá trị cho session này là 0, nếu username không tồn tại thì giá trị gán là –1. Khi lỗi phát sinh thì gán giá trị là -2.

2. LOGIN, TRANG CHÍNH VÀ LOGOUT

2.1. Trang LOGIN

Để xây dựng trang login, bạn khai báo thành ba phân đoạn, trước tiên là khai báo đoạn JavaScript để kiểm tra username / password như ví dụ sau:

```
<script language="JavaScript">
  function checkinput(){
    if(document.form1.username.value==""){
      alert("Xin vui long nhap username");
      document.form1.username.focus();
      return false;
    }
    if(document.form1.password.value=="" ){
      alert("Xin vui long nhap password");
      document.form1.password.focus();
      return false;
    }
```

```
    return true;
  }
</script>
```

Sau đó, dựa vào giá trị của session có tên userid, bạn có thể thông báo chuỗi thông báo như sau:

```
<%
  String userid=getVal((String)session.getValue("userid"),"");
  String err="Xin vui lòng nhập username và password";
  if(userid.equals("0")){
    err="<font color=\"#FF0000\">Sai password,Xin vui lịng nhập
  password</font>";}
  if(userid.equals("-1")){
    err="<font color=\"#FF0000\">Sai username,Xin vui lịng nhập
  username</font>";}
  if(userid.equals("-2")){
    err="<font color=\"#FF0000\">Lỗi hệ thống, Xin vui lòng thử
  lại</font>";}
%>
```

Sau đó, khai báo thẻ form cùng với các thẻ input cho phép người sử dụng nhập username/password và gọi phương thức checkinput():

```
<form action=login_authentication.jsp name=form1
  method=post onSubmit="return checkinput();">
  <table align=center>
   <tr>
            <td colspan="2" height="19">
             <h4><b><font color="#FF0000">
        <font color="#0000CC">Nhân viên quản lý</font>
        </font></b></h4><hr>
            </td>
           </tr><tr>
            <td colspan=2 height="19">
             <div align="left"><%=err%></div>
            </td>

           </tr>
        <tr>
            <td height="19">
             <div align="left">Username</div>
            </td>
            <td height="19">
             <div align="left">
              <input type="text" name="username"
                   size="30" maxlength="50" class=textbox>
             </div>
            </td>
           </tr>
           <tr>
            <td height="31">
             <div align="left">Password</div>
            </td>
            <td height="31">
             <div align="left">
              <input type="password" name="password"
                   size="30" maxlength="10" class=textbox>
             </div>
            </td>
           </tr>
           <tr>
            <td > </td>
```

```
            <td> </td>
          </tr><tr>
            <td ></td>
           <td ><input type=submit value="Đăng nhập">
                    <input type=reset value=Huỷ></td>
          </tr>
</table>
</form>
```

Khi người sử dụng triệu gọi trang login lần đầu tiên, màn hình xuất hiện như hình 9-1.

Hình 9-1: Trang login

Nếu người sử dụng không nhập user/password mà nhấn nút "Đăng nhập" thì cửa sổ thông báo do phương thức checkinput khai báo ở trên phun ra như hình 9-2.

Hình 9-2: Thông báo yêu cầu nhập username/password

Trong trường hợp đã nhập username mà chưa cung cấp passwprd thì cửa sổ xuất hiện như hình 9-3.

Hình 9-3: Yêu cầu nhập password

Lưu ý rằng, sau khi thông báo yêu cầu người sử dụng cung cấp username hay password, nếu người sử dụng nhấn nút IK lập tức con trỏ sẽ chỉ vào textbox tương ứng.

Nếu cả hai giá trị đếu cung cấp, trang khai báo trong thuộc tính action của thẻ form sẽ được triệu gọi. Trong trường hợp này chính là trang login_authentication.jsp.

2.2. Trang kiểm tra cơ sở dữ liệu

Sau khi submit đến trang login_authentication.jsp, bạn kết nối cơ sở dữ liệu và đọc trong bảng tblusers với phát biểu Select và mệnh đề Where ứng với cột username.

Nếu tìm thấy mẩu tin tồn tại bạn tiếp tục so sánh giá trị trong cột password với passsword nhập từ trang login.jsp, nếu password hợp lệ thì khởi tạo session và trỏ đến trang myaccount.jsp.

Trong trường hợp không tìm thấy mẩu tin nào có username bằng với username mà người sử dụng cung cấp thì bạn trở về trang login.jsp với giá trị của session có tên userid là –1.

Đối với trường hợp so sánh password không bằng nhau thì bạn trở về trang login.jsp với giá trị của session có tên userid là 0.

Nếu lỗi phát sinh do kết nối cơ sở dữ liệu thì bạn bạn trở về trang login.jsp với giá trị của session có tên userid là –2.

```
<%@ page import="java.sql.*" %>
<%@ include file="common.jsp"%>
<%
   int err=0;
   String userid="";
   String username="";
   String email="";
   String password="";
   String fullname="";
  try{
   Connection cn;
   Statement smt;
   ResultSet rst;
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   cn = DriverManager.getConnection(odbc,sysuser,syspwd);
   smt = cn.createStatement();

   username= request.getParameter("username");
   username=replaceString(username,"'", "''");
   session.putValue("username",username);
```

```
password= request.getParameter("password");
String strSQL="";
String pwd="";
strSQL="select * from tblusers where username = '" + username + "'";
rst=smt.executeQuery(strSQL);

if (rst.next()){
     userid= rst.getString("UserID");
     pwd=rst.getString("Password");
     fullname= rst.getString("Fullname");
     email= rst.getString("Email");
     if(pwd.equals(password)){
        session.putValue("userid",userid);
        session.putValue("email",email);
        session.putValue("fullname",fullname);
        response.sendRedirect("myaccount.jsp");
     }
     else{
        /*sai password*/
        session.putValue("userid","0");
        response.sendRedirect("login.jsp");
     }
   }
else{
   /*sai user*/
   session.putValue("userid","-1");
   response.sendRedirect("login.jsp");
}
smt.close();
cn.close();
}
catch (Exception e){
   /*sai ket noi*/
   session.putValue("userid","-2");
   out.println(e);
   response.sendRedirect("login.jsp");
}


%>
```

Chúng ta thay thế giá trị nhập có dấu ‘ thành hai dấu nháy đơn liên tiếp bằng phương thức `replaceString` khai báo trong trang common.jsp như sau:

```
username= request.getParameter("username");
username=replaceString(username,"'", "''");
```

Lưu ý rằng, khi username và password đều hợp lệ thì chúng ta khai báo như sau:

```
if(pwd.equals(password))
{
       session.putValue("userid",userid);
       session.putValue("email",email);
       session.putValue("fullname",fullname);
       response.sendRedirect("myaccount.jsp");
}
```

Nếu sai password thì bạn khai báo như sau:

```
else{
```

```
            /*sai password*/
            session.putValue("userid","0");
            response.sendRedirect("login.jsp");
         }
```

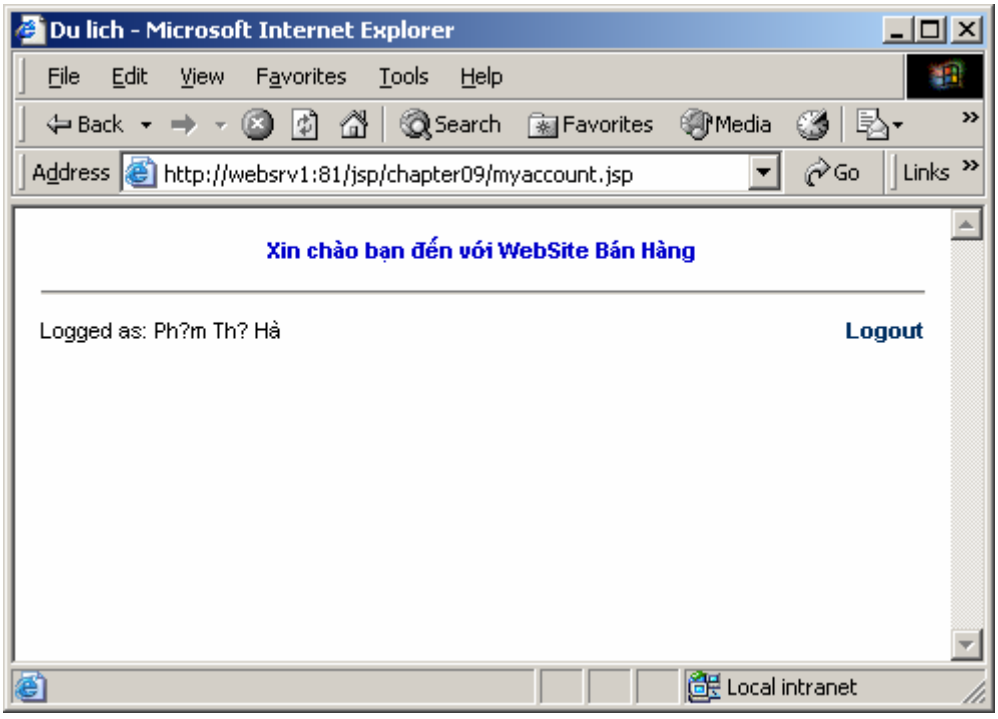
Trong trường hợp không tìm thấy username tồn tại trong cơ sở dữ liệu thì bạn khai báo như sau:

```
else{
     /*sai user*/
     session.putValue("userid","-1");
     response.sendRedirect("login.jsp");
   }
```

Nếu lỗi phát sinh do hệ thống hay kết nối cơ sở dữ liệu, bạn khai báo trong phát biểu catch như sau:

```
catch (Exception e){
   /*sai ket noi*/
   session.putValue("userid","-2");
   response.sendRedirect("login.jsp");
}
```

Sau khi đăng nhập thành công, trang myaccount.jsp xuất hiện như hình 9-4.



Hình 9-4: Trang myaccount.jsp

Bạn có thể trình bày nhiều loại thông tin trong trang này, tuy nhiên chúng tôi chỉ xuất ra giá trị của session có tên là fullname như ví dụ sau.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ include file="common.jsp"%>
<%@ include file="checksession.jsp"%>
<html>
<head>
<title>Du lich</title>
<LINK href="style.css" rel=stylesheet>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ; charset=utf-8">
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="5" cellpadding="5">
<tr><td>
  <table width="100%">
   <tr>
      <td colspan="2" height="19" align=center>
      <b><font color="#FF0000"><font color="#0000CC">
     Xin chào bạn đến với WebSite Bán Hàng </font></b>
     </td></tr>
     <tr><td colspan=2><hr></td></tr>
     <tr><td>Logged as:
     <%=getVal((String)session.getValue("fullname"),"")%>
             </td><td align=right>
              <b><a href="logout.jsp">Logout</a></b>
             </td>
            </tr>

</table>
</td></tr></table>
</body>
</html>
```

Tuy nhiên, trang myaccount.jsp chỉ được sử dụng khi người sử dụng đã đăng nhập, điều này có nghĩa là bạn phải kiểm tra session userid phải có giá trị lớn hơn 0 mới cho phép trình bày thông tin trên trang này nếu không thì triệu gọi trang login.jsp để yêu cầu người sử dụng đăng nhập.

Trong trường hợp này chúng khai báo trức tiếp trong trang myaacount.jsp hay khai báo thành trang sử dụng chung là checksession.jsp sau đó khai báo chèn vào myaccount.jsp.

```
<%
String userid=getVal((String)session.getValue("userid"),"");
if(userid.equals("")|| userid.equals("0") || userid.equals("-1")||
userid.equals("-2"))
  response.sendRedirect("login.jsp");

%>
```

Trên trang myaccount.jsp chúng ta khai báo một URL cho phép người sử dụng thoát (logout) khỏi ứng dụng bằng cách triệu gọi trang logout.jsp. Trang này thực hiện quá trình huỷ tất cả các session đã tạo ra trong quá trình sử dụng của người dùng và trỏ đến trang login.jsp.

```
<%
session.putValue("userid","");
session.putValue("username","");
session.putValue("email","");
session.putValue("fullname","");
response.sendRedirect("login.jsp");
%>
```

3. KẾT LUÂN

Trong bài này, chúng ta tìm hiểu chức năng đăng nhập hệ thống, bài kế tiếp chúng ta tiếp tục tìm hiểu về xáo dữ liệu dạng mảng control trên form.

Lưu ý rằng, các trang jsp trên không sử dụng chè tập tin, trong thực tế bạn cần khai báo các trang top, left, right và bottom sau đó chèn vào mỗi trang web khác.

Môn học: Java Server Pages

BÀI 10: XOÁ, CẬP NHẬT DỮ LIỆU DẠNG MẢNG

Trong bài trước chúng ta đã làm quen với cách xoá mẩu tin trong cơ sở dữ liệu SQL Server. Đối với trường hợp xoá một lúc nhiều mẩu tin, chúng ta phải xây dựng trang JSP có sử dụng thẻ input dạng checkbox.

Những vấn đề chính sẽ được đề cập trong bài học này

- ✓ Liệt kê dữ liệu dạng danh sách
- ✓ Xoá nhiều mẩu tin
- ✓ Cập nhật nhiều mẩu tin

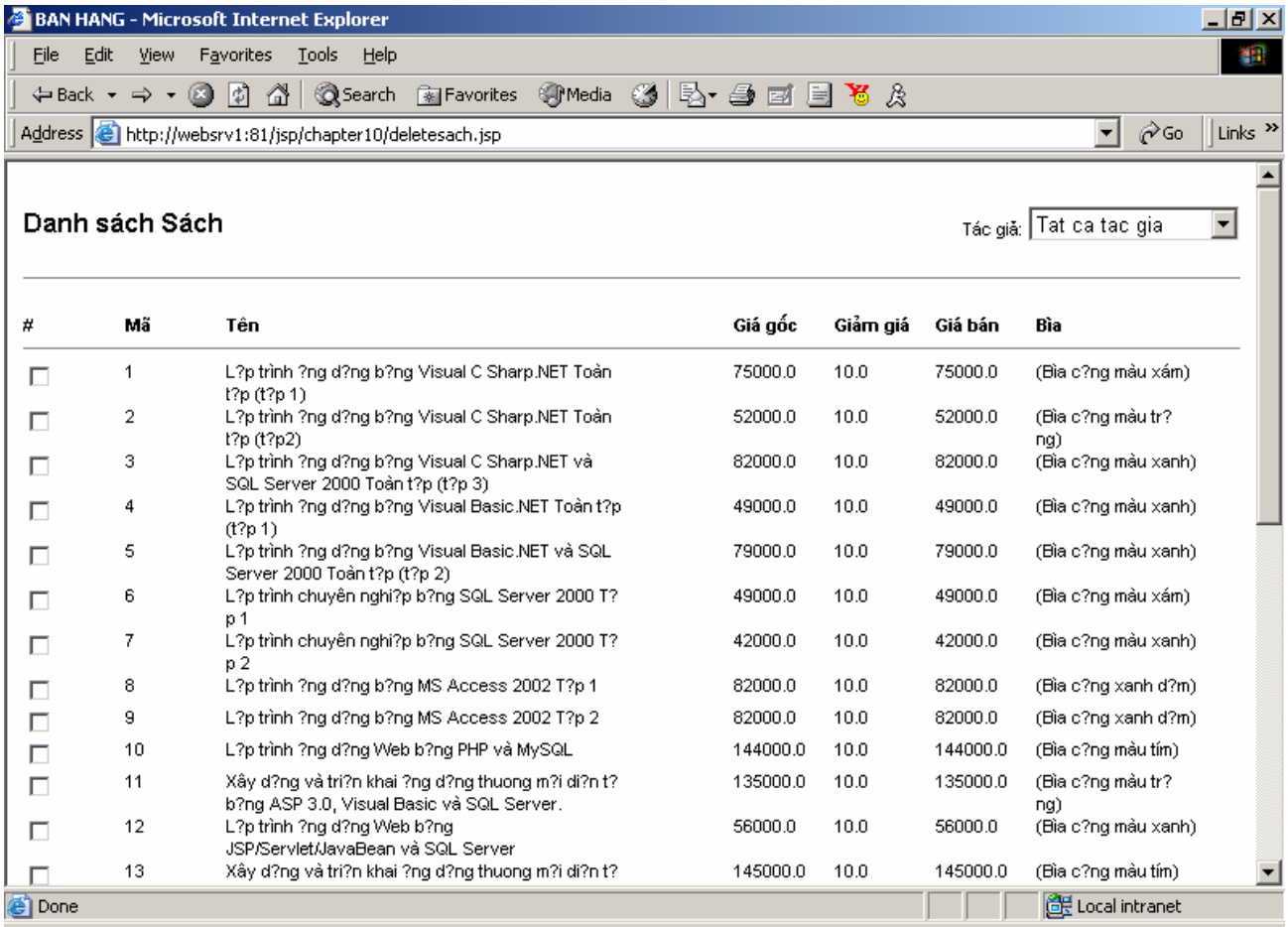
1. LIỆT KÊ DỮ LIỆU

Để xoá nhiều mẩu tin cùng một lúc, trước tiên bạn khai báo trang JSP để liệt kê danh sách mẩu tin trong mảng dữ liệu chẳng hạn, mỗi lập trình xuất hiện một checkbox tương ứng. Checkbox này có giá trị là mã nhận dạng của mẩu tin đó. Trong trường hợp này chúng ta dùng cột khoá của mã sách trong bảng tblItems.

Tuy nhiên, để cho phép lấy được nhiều giá trị chọn của sản phẩm, bạn khai báo các checkbox này cùng tên (giả sử tên là chkid) và giá trị trị là ItemID của mỗi sản phẩm như ví dụ 10-1 trong trang deletesach.jsp.

```
try{
   rst=smt.executeQuery(strSQL);
   while(rst.next())
      {
         ID=rst.getString("ItemID");
         out.println("<tr><td width=50 ><input type=checkbox ");
         out.println("name=chkid value='"+ID+"'></td>");
         out.println("<td width=50 valign=top>"+ID+"</td>");
         strO =rst.getString("ItemName");
         out.println("<td width=250 valign=top>");
         out.println( strO + "</td>");
         strO =rst.getString("ListPrice");
         out.println("<td width=50 valign=top>");
         out.println( strO + "</td>");
         strO =getVal(rst.getString("SalesDiscount"),"");
         out.println("<td width=50 valign=top>");
         out.println( strO + "</a></td>");
         strO =getVal(rst.getString("SalesPrice"),"");
         out.println("<td width=50 valign=top>");
         out.println( strO + "</a></td>");
         strO =rst.getString("ItemStyle");
         out.println("<td width=100 valign=top>");
         out.println( strO + "</a></td>");
         out.println( "</tr>");
         totalRecords++;
   }
...
```

Tuy nhiên, do nhiều cuốn sách thuộc các tác giả khác nhau, chính vì vậy bạn khai báo danh sách tác giả trên thẻ select cho phép người sử dụng liệt kê sách theo nhóm tác giả như hình 10-1.



Hình 10-1: Liệt kê danh sách sách

Để liệt kê danh sách tác giả trong bảng tblAuthors, bằng cách khai báo phương thức nhận chuỗi SQL dạng Select và giá trị mặc định trả về nhiều phần tử thẻ option trong tập tin database.jsp như ví dụ 10-2.

```
<%!
public String getOption(String strSQL,String strSelect)
{
String OptionString="";
Connection cns;
Statement smts;
ResultSet rsts;
try{
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     cns = DriverManager.getConnection(odbc,sysuser,syspwd);
     smts = cns.createStatement();
     String ID="";
     String strO;
     String strU;
     rsts=smts.executeQuery(strSQL);
     while(rsts.next())
        {
          ID=rsts.getString("ID");
          OptionString+="<option value='"+ID+"' ";
```

```
            if(!strSelect.equals(""))
              if(strSelect.equals(ID)) OptionString+=" selected ";
            strO =rsts.getString("Name");
            strU=new String(strO.getBytes("ISO-8859-1"),"UTF-8");
            OptionString+= ">" + strO + "</option>";
          }
          rsts.close();
        smts.close();
        cns.close();
        }
        catch (Exception e){
          OptionString=e.toString();
        }
    return OptionString;
    }

    %>
```

Sau đó, gọi phương thức này trong trang deletesach.jsp như ví dụ 10-3.

```
<%@ include file="database.jsp"%>
<%
String strID=getVal(request.getParameter("selectid"),"");
String strOptions="<option value=''>Tat ca tac gia</option>";
strOptions+=getOption("Select AuthorID AS ID, AuthorName AS NAME from
tblAuthors",strID);
%>
```

Lần đầu tiên bạn có thể chọn mặc định một tác giả hoặc liệt kê tất cả, khi người sử dụng chọn tác giả nào đó thì trang deletesach.jsp sẽ liệt kê danh sách sách của tác giả đó. Để làm điều này, bạn khai báo thẻ form với thẻ select như ví dụ 10-4.

```
<form name=frmSelect method=post action=deletesach.jsp>
<tr><td>
<font size=3><b>Danh ́sch Śch</b></font></td><td align=right>
Tác giả: <select name=selectid
     onchange="document.frmSelect.submit();">
<%=strOptions%></select></td>
</tr>
</form>
```

Khi người sử dụng chọn các mẩu tin như hình 10-2 và nhấn nút Delete, dựa vào giá trị của nút có tên action (trong trường hợp này là Delete), bạn có thể khai báo biến để lấy giá trị chọn bằng cách khai báo như ví dụ 10-5.

```
deleterecord=request.getParameterValues("chkid");
if(deleterecord!=null){
     for(int k=0;k<deleterecord.length;k++){
       delStr+="'"+deleterecord[k]+"'"+",";
     }
     delStr=delStr.substring(0,delStr.length()-1);
}
```

Dựa vào thẻ hidden khai báo trong các trang trình bày danh sách mẩu tin như sau:

```
<input name="from" type=hidden value="deletesach">
```

Bạn có thể biết từ trang nào gọi đến trang dosql.jsp để quay trở về khi thực hiện xong tác vụ xử lý.

Ngoài ra, dựa vào giá trị của nút action để thực hiện phát biểu SQL. Chẳng hạn, trong trường hợp này nếu người sử dụng nhấn hút Delete thì bạn khai báo như ví dụ 10-6 sau:

```
if(action.equals("Delete"))
{
     strSQL=" delete from tblItems Where ItemID ";
     msg="Xố ́ sch thnh cng";
}
```

Sau đó, sử dụng đối tượng Connection, Statement bạn có thể thực thi phát biểu SQL vừa khai báo ở trên như ví dụ 10-7.

```
if(!strSQL.equals("") )
{
     strSQL+=" in (" + delStr +")";
     Connection cn;
     Statement smt;
     ResultSet rst;
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     cn = DriverManager.getConnection(odbc,sysuser,syspwd);
     smt = cn.createStatement();
     int i = smt.executeUpdate(strSQL);
     smt.close();
     cn.close();
}
```

Lưu ý rằng, bạn cũng nên khai báo try catch trong khi làm việc với cơ sở dữ liệu. Ngoài ra, bạn cũng phải xác nhận trước khi thực thi hành động xoá mẩu tin chọn bằng cách khai báo đoạn Javascript như sau:

```
<script>
function isok()
 {
     return confirm('Are you sure to delete?');
 }
</script>
```

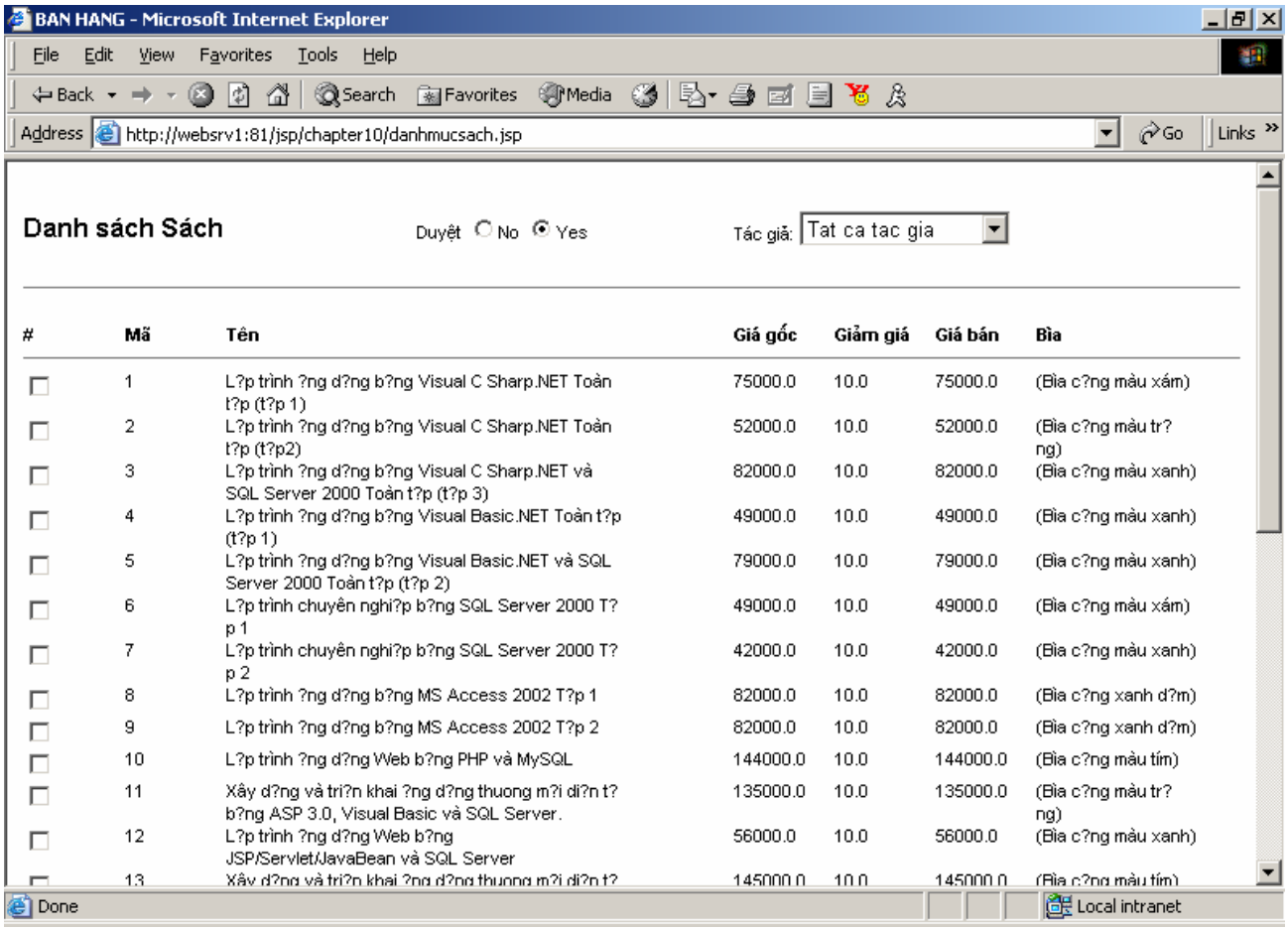
Sau đó gọi trong biến cố onsubmit của form như sau:

```
<form action=dosql.jsp method=post onsubmit="return isok();">
```

2. CẬP NHẬT NHIỀU MẨU TIN

Tương tự như trường hợp Delete, khi bạn duyệt (approval) một số mẩu tin theo một cột dữ liệu nào đó, chẳng hạn, trong trường hợp này chúng ta cho phép sử dụng những sản phẩm đã qua sự đồng ý của nhà quản lý thì cột dữ liệu Activate của bảng tbltems có giá trị là 1.

Để làm điều này, trước tiên bạn liệt kê danh sách sản phẩm như hình 10-3.



Hình 10-3: Liệt kê danh sách sản phẩm duyệt hay chưa

Mặc định là danh sách sản phẩm chưa duyệt, để làm điều này bạn hai báo hai thẻ input loại radio và mặc định là chọn vào tuỳ chọn thứ nhất, trong trường hợp người sử dụng chọn vào tuỳ chọn thứ hai thì sau khi submit bạn chọn lại tuỳ chọn thứ hai như ví dụ 10-8.

```
<table border=0 width=100%><tr><td>
Duyệt <input type=radio name=selectstatus
onclick="document.frmSelect.submit();" value="0" checked>No
<input type=radio name=selectstatus
onclick="document.frmSelect.submit();" value="1"
<%if(strStatus.equals("1")){ %> checked<%}%>>Yes</td><td>
Tác giả: <select name=selectid onchange="document.frmSelect.submit();">
<%=strOptions%></select>
</td></tr></table>
```

Để lấy giá trị người sử dụng chọn trên hai tuỳ chọn này, bạn khai báo như sau:

```
String strStatus=getVal(request.getParameter("selectstatus"),"0");
```

Mặc định phát biểu SQL dạng select có mệnh đề Where với cột Activate tuỳ thuộc vào giá trị chọn của một trong hai tuỳ chọn trên bạn khai báo như ví dụ 10-9:

```
String strSQL="select ItemID, ItemName,ListPrice, SalesDiscount,
SalesPrice, ItemStyle,Activate from tblItems ";
strSQL+=" where Activate='" + strStatus+ "' ";
if(!strID.equals(""))
  strSQL+=" and AuthID='" + strID+ "'";
```

Tuỳ thuộc vào số mẩu tin liệt kê thuộc trạng thái chưa duyệt thì nút Approve hiện ra, ngược lại là nút Unapprove. Điều này có nghĩa là nút Approve dùng cho những mẩu tin chưa duyệt và ngược lại. Để làm điều này bạn khai báo như ví dụ 10-10.

```
<input name=action type=submit
value="<%if(strStatus.equals("1")){
%>Unapprove<%}else{%>Approve<%}%>"> </td></tr>
```

Như vậy, dựa vào nút action có giá trị là Delete, Approve hay Unapprove bạn định nghĩa phát biểu SQL trong trang dosql.jsp. Đối với trường hợp này chúng ta khai báo thêm trong trang dosql.jsp như ví dụ 10-11.

```
if(action.equals("Approve"))
{
  strSQL=" update tblItems set Activate=1 Where ItemID ";
  msg="Duyệt sách thành công";
}
if(action.equals("Unapprove"))
{
  strSQL=" update tblItems set Activate=0 Where ItemID ";
  msg="Huỷ duyệt sách thành công";
}
```

3. KẾT LUẬN

Trong bài này, chúng ta tìm hiểu chức năng xoá, cập nhật nhiều mẩu tin bằng cách sử dụng thẻ input loại checkbox cùng tên và khác giá trị, bài kế tiếp chúng ta tiếp tục tìm hiểu về cách tìm kiếm và phân trang trong JSP.

Môn học: Java Server Pages

BÀI 11: TÌM KIẾM, PHÂN TRANG

Chúng ta vừa làm quen cách trình bày dữ liệu dưới nhiều hình thức khác nhau. Trong chương này, chúng ta tiếp tục tìm hiểu cách xây dựng cơ sở dữ liệu và trang JSP cho phép người sử dụng tìm kiếm theo hình thức đơn giản hay phức tạp.

Sau khi cung cấp các tiêu chuẩn tìm kiếm, nếu người sử dụng submit thì trang kết quả sẽ trình bày danh sách mẩu tin thoả điều kiện đó.

Trong trường hợp số mẩu tin liệt kê có số lượng nhiều, bạn có thể phân chia ra nhiều trang nhằm giúp cho người sử dụng dễ xem thông tin do họ tìm kiếm.

Các vấn đề chính sẽ được đề cập:

- ✓ Xây dựng bảng dữ liệu phục vụ tìm kiếm
- ✓ Tìm kiếm đơn giản
- ✓ Tìm kiếm nâng cao
- ✓ Phân trang và điều hướng

1. XÂY DỰNG BẢNG DỮ LIỆU PHUC VU TÌM KIẾM

Khi thiết kế dữ liệu, bảng lưu trữ thông tin chi tiết của sản phẩm có thể có nhiều cột, nếu người sử dụng cung cấp một từ khoá để tìm kiếm mẩu tin thoả yêu cầu này, bạn phải khai báo phát biểu *Select* có mệnh đề *Where* dựa trên các cột dữ liệu cho phép tìm.

Tuy nhiên, khi người sử dụng cung cấp từ khoá tìm kiếm không thuộc đề tài của cuốn sách mà chỉ có một chương hay phần nào đó trình bày về vấn đề đó, đối với trường hợp này chúng ta xử lý như thế nào.

Để cho phép kết quả tìm kiếm trả về ứng với từ khoá do người sử dụng cung cấp mà cuốn sách có thể cùng đề tài hay một phần của chúng trình bày về vấn đề có từ khoá này, chúng ta khai báo cột dữ liệu có tên *keywords*.

Cột *keywords* cho phép bạn nhập tất cả các từ khoá chính mà sản phẩm có đề cập. Chẳng hạn, chúng ta có bảng *tblItems* lưu trữ danh sách nhiều loại sách, trong bảng này có cột *keywords* lưu trữ các từ khoá như: *COM, COM+, DLL, API, Access, Excel, World, PowerPoint, Outlook, Crystal Report, Unicode, Grid, SQL* ứng với cuốn sách "Kỹ xảo lập trình *Visual Basic 6.0*" và *SQL Server, Database, Servlet, JSP, JavaScript, SQL, Access*, *Shopping Cart*, *Payment*, *RMI, EJB, Class* ứng với cuốn sách "Xây dựng ứng dụng thương mại điện tử bằng *JSP và Servlet*".

Như vậy, khi người sử dụng tìm kiếm từ khoá *SQL* thì những cuốn sách có trình bày về *SQL* sẽ được liệt kê, đối với trường hợp này hai cuốn sách trên vẫn nằm trong danh sách liệt kê mặc dù đề tài cuốn sách là *Java* và *JSP*.

Với cấu trúc như trên, bạn sử dụng thuật toán tìm kiếm theo độ ưu tiên và tách chuỗi để liệt kê những cuốn sách có tựa đề về đề tài này liệt kê trước sau đó liệt kê những cuốn sách có trình bày một phần về *SQL*.

Tóm lại, bạn khai báo cột *keywords* để lưu trữ tất cả các từ khoá mà cuốn sách có trình bày về chúng. Thay vì so sánh chuỗi tìm kiếm theo yêu cầu của người sử dụng trên những cột dữ liệu liên quan thì bạn chỉ cần so sánh trên chính cột này.

Đối với trường hợp này, bạn sử dụng phép toán *LIKE* với ký hiệu % chẳng hạn, cho phép bạn so sánh gần đúng với chuỗi yêu cầu tìm kiếm.

Tuy nhiên, *Microsoft* cung cấp một dịch vụ tìm kiếm gọi là *Full-Text Search* cho phép bạn sử dụng các hàm của *SQL Server* để tìm kiếm từ trên mọi cột của bảng dữ liệu.

2. TÌM KIẾM ĐƠN GIẢN

Sau khi khai báo cột dữ liệu có tên *keywords* trong bảng *tblItems*, bạn có thể nhập các từ khoá của từng sản phẩm có toàn bộ hay một phần nội dung trình bày về từ khoá này vào cột *keywords*.

Từ ứng dụng *JSP*, bạn thiết kế trang tìm kiếm đơn giản (*basic search*) như hình 11-1 với *textbox* cho phép người sử dụng *Internet* nhập từ khoá bất kỳ. Lưu ý rằng, trang này có thể là một trang hay một phần trong trang chính khác.

Hình 11-1: Trang tìm kiếm đơn giản

Để thực hiện điều này, bạn khai báo nội dung *HTML* của phần *body* trong trang *timkiemcoban.jsp* như ví dụ 11-1.

Ví dụ 11-1: Phần BODY của trang tìm kiếm đơn giản

```
...
<body topmargin="0" leftmargin="0" rightmargin="0">
        <form action="ketquacoban.jsp" method="GET" name="frmView"
             onsubmit="return checkinput();">
          <table width="100%" border="0" cellspacing="3" cellpadding="3">
            <tr>
              <td colspan="2" ><b>
                <font  face="Arial">Xin vui lòng cung cấp các thông tin
                tìm kiếm vào ô từ khoá.<br>
                </font>  </td>
            </tr>
            <tr>
              <td width="30%"><font face="Arial">T□ khố</font></td>
              <td width="70%"> <font face="Arial">
                <input name="keyword" id="word"  class=text size="30" >
                </font></td>
            </tr>
            <tr align="left">
              <td  height="51"> </td><td>
                <input type=submit value="Tìm Ki□m">
               <input type=reset value="Hu□">
                <input type=hidden name= "Searchfrom" value="CB">
              </td>
            </tr>
          </table>
        </form>
</body>
```

Trong ví dụ trên, chúng ta sử dụng phương thức *GET* cho thẻ *form* nhằm cho phép truyền giá trị tìm kiếm của người sử dụng lên *QueryString* thay vì dùng phương thức *Post*.

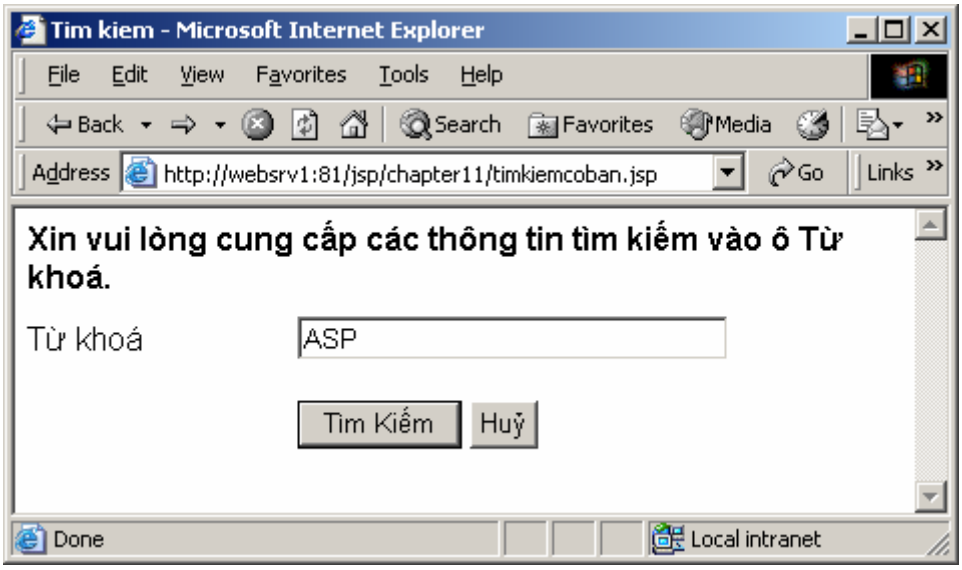
Ngoài ra, bạn khai báo đoạn JavaScript để yêu cầu nhập từ khoá trước khi nhấn nút Tìm Kiếm trong trang *timkiemcoban.jsp* như ví dụ 11-2.

Ví dụ 11-2: Phần khai báo chèn file

```
<SCRIPT language=JavaScript>
        function checkinput()
```

```
        {
        if (document.frmView.keyword.value=="" )
        {
            alert("Xin vui long nhap ten, tua de, ten tac gia, nha xuat ban");
            document.frmView.keyword.focus();
            return false;
        }
        return true;
        }
</SCRIPT>
```

Sau khi người sử dụng nhập chuỗi *ASP* vào phần từ khoá như hình 11-2 và nhấn nút *Tìm Kiếm*, trang *ketquacoban.jsp* sẽ được triệu gọi, kết quả trình bày như hình 11-3.



Hình 11-2: Tìm kiếm từ khoá

Để lấy giá trị nhập từ trang *timkiemcoban.jsp* và kết nối cơ sở dữ liệu, kế đến liệt kê những mẩu tin trong bảng *tblItems* có từ trùng với từ *ASP* tại cột *keywords* thì bạn khai báo mệnh đề Where trong trang *ketquacoban.jsp* bạn như ví dụ 11-3.

Ví dụ 11-3: Khai báo tìm kiếm đơn giản

```
if (!(keyword==null)){
    strShow += keyword + ", ";
    strWhere+= " and (ItemName like '%" ;
```

```
    strWhere+= keyword;
    strWhere+= "%' or Keywords like '%" ;
    strWhere+= keyword;
    strWhere+= "%' or SubCateName like '%" ;
    strWhere+= keyword;
    strWhere+= "%')";
}
```

Trong đó, phần định nghĩa phát biểu *SQL* dạng *Select* với giá trị tìm kiếm từ trang tìm kiếm như ví dụ 11-4 sau:

Ví dụ 11-4: Khai báo kiểm tra

```
<%
String strShow="";
String strWhere="";
String keyword=request.getParameter("keyword");
if (keyword==null)
    {
      response.sendRedirect("timkiemcoban.jsp");
    }
if (!(keyword==null)){
    strShow += keyword + ", ";
    strWhere+= " and (ItemName like '%" ;
    strWhere+= keyword;
    strWhere+= "%' or Keywords like '%" ;
    strWhere+= keyword;
    strWhere+= "%' or SubCateName like '%" ;
    strWhere+= keyword;
    strWhere+= "%')";
}
%>
```

Kết quả tìm kiếm liệt kê như hình 11-3 bao gồm danh sách các sản phẩm có một trong những từ khoá bằng *ASP*.

Hình 11-3: Kết quả tìm kiếm

Để làm điều này, bạn khai báo kết nối cơ sở dữ liệu và đọc các field trình bày như ví dụ 11-5.

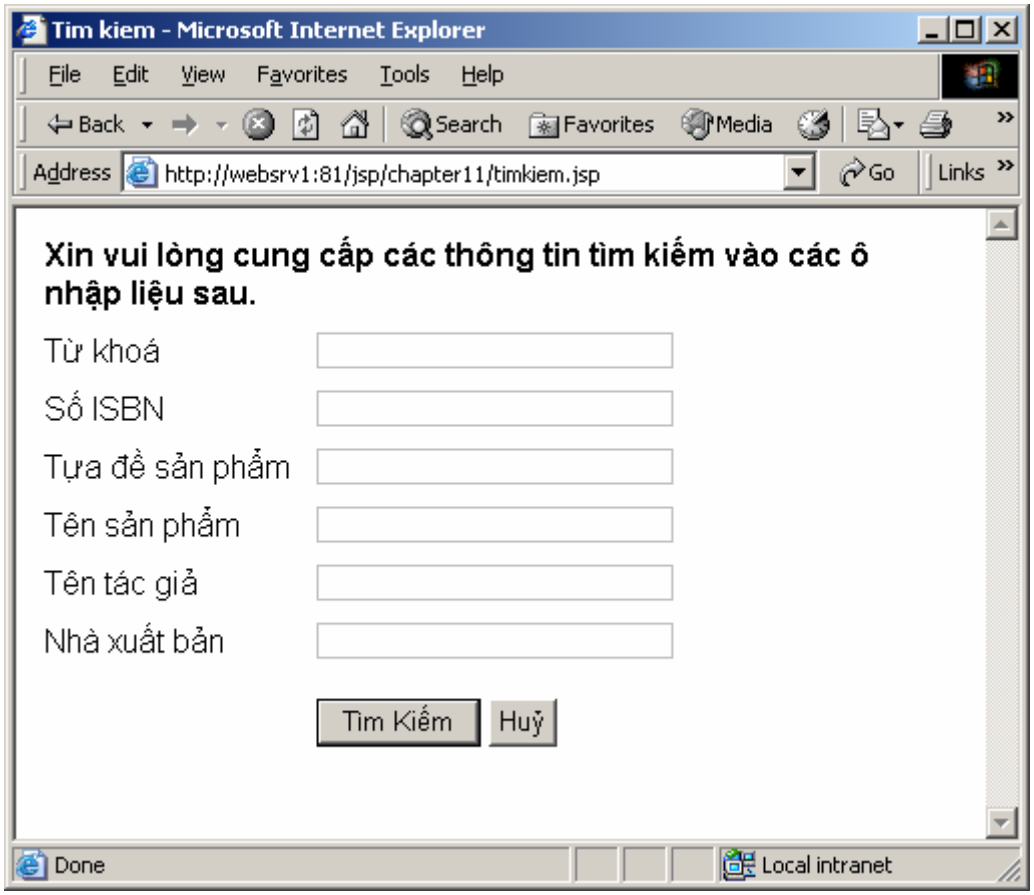
Ví dụ 11-5: Khai báo tìm kiếm đơn giản

```
<%
Connection cn;
Statement smt;
ResultSet rst;
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
cn = DriverManager.getConnection(odbc,sysuser,syspwd);
String strSQL="select * from vwItems where ItemID>0 " ;
strSQL += " " + strWhere;

String sorts=request.getParameter("sort");
String x="ItemName,ProName,AuthorName";
if( !(sorts==null) && (x.indexOf(sorts)!=-1) )
{
    strSQL=strSQL + " order by CustomerRating ASC,";
    strSQL=strSQL + sorts;
    strSQL=strSQL +  " ASC";
```

```
}
String qryString="keyword=" + keyword ;
smt = cn.createStatement();
int viewrow=1;
try{
    rst=smt.executeQuery(strSQL);
    while(rst.next())
      {
%>
```

3. TÌM KIẾM NÂNG CAO

Tìm kiếm nâng cao là cho phép người sử dụng cung cấc tiêu chuẩn tìm kiếm phức tạp hơn, chẳng hạn trong trường hợp này chúng ta thiết kế trang timkiem.jsp như hình 11-4.



Hình 11-4: Tìm kiếm nâng cao

Đối với trang này, yêu cầu người sử dụng ít nhất phải nhập giá trị ứng với field, bằng cách khai báo đoạn JavaScript để kiểm tra quá trình ràng buộc này.

Ví dụ 11-6: Khai báo ràng buộc tìm kiếm nâng cao

```
<SCRIPT language=JavaScript>
    function checkinput()
   {
     if(document.frmView.word.value=="" &&
     document.frmView.subject.value=="" &&
     document.frmView.name.value=="" &&
     document.frmView.author.value=="" &&
     document.frmView.publish.value=="" &&
     document.frmView.ISBN.value=="")
     {
       alert("Xin vui long nhap tu khoa/ten/tua de/ten
       tac gia/nha xuat ban de tim kiem");
       document.frmView.word.focus();
       return false;
     }
   return true;
   }
</SCRIPT>
```

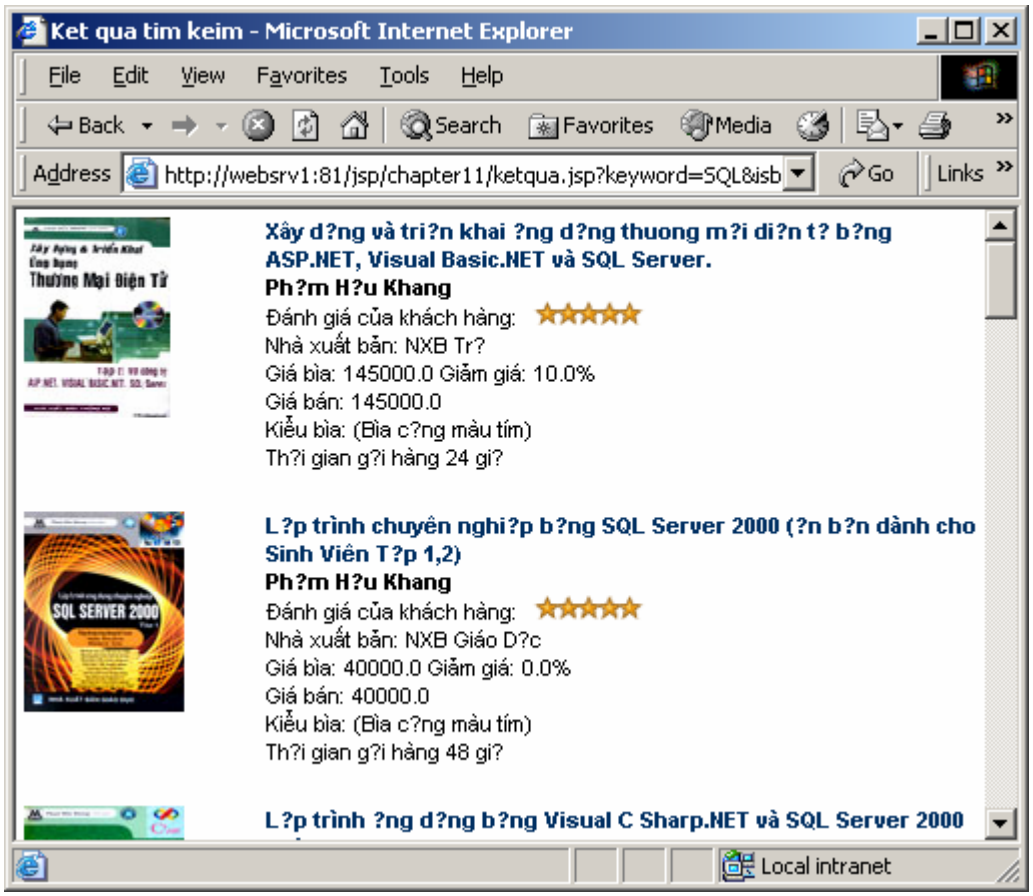
Sau khi nhấn nút Tìm kiếm, trang ketqua.jsp sẽ đựoc triệu gọi, tương tự như trường hợp tìm kiếm cơ bản, bạn khai báo mệnh đề Where như ví dụ 11-7.

Ví dụ 11-7: Khai báo mệnh đề Where

```
<%
Connection cn;
Statement smt;
ResultSet rst;
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
cn = DriverManager.getConnection(odbc,sysuser,syspwd);
String strSQL="select * from vwItems where ItemID>0 " ;
String strCount="select count(*) as No from vwItems where
ItemID>0 ";
strSQL += " " + strWhere;
strCount+=strWhere;
String sorts=request.getParameter("sort");
String x="ItemName,ProName,AuthorName";
if( !(sorts==null) && (x.indexOf(sorts)!=-1) )
{
    strSQL=strSQL + " order by CustomerRating ASC,";
    strSQL=strSQL + sorts;
    strSQL=strSQL +  " ASC";
}
```

```
String qryString="keyword=" + keyword + "&name=" + name
+"&subject="+subject+"&author="+author+"&publish="+pu
blish+"&isbn="+isbn+"&sort=" + sorts;
smt = cn.createStatement();
int viewrow=1;
try{
    rst=smt.executeQuery(strSQL);
    while(rst.next())
      {
%>
```

Giả sử bạn tìm kiếm từ khoá SQL, kết quả trả về như hình 11-5.



Hình 11-5: Kết quả tìm kiếm nâng cao

4. PHÂN TRANG VÀ ĐIỀU HƯỚNG

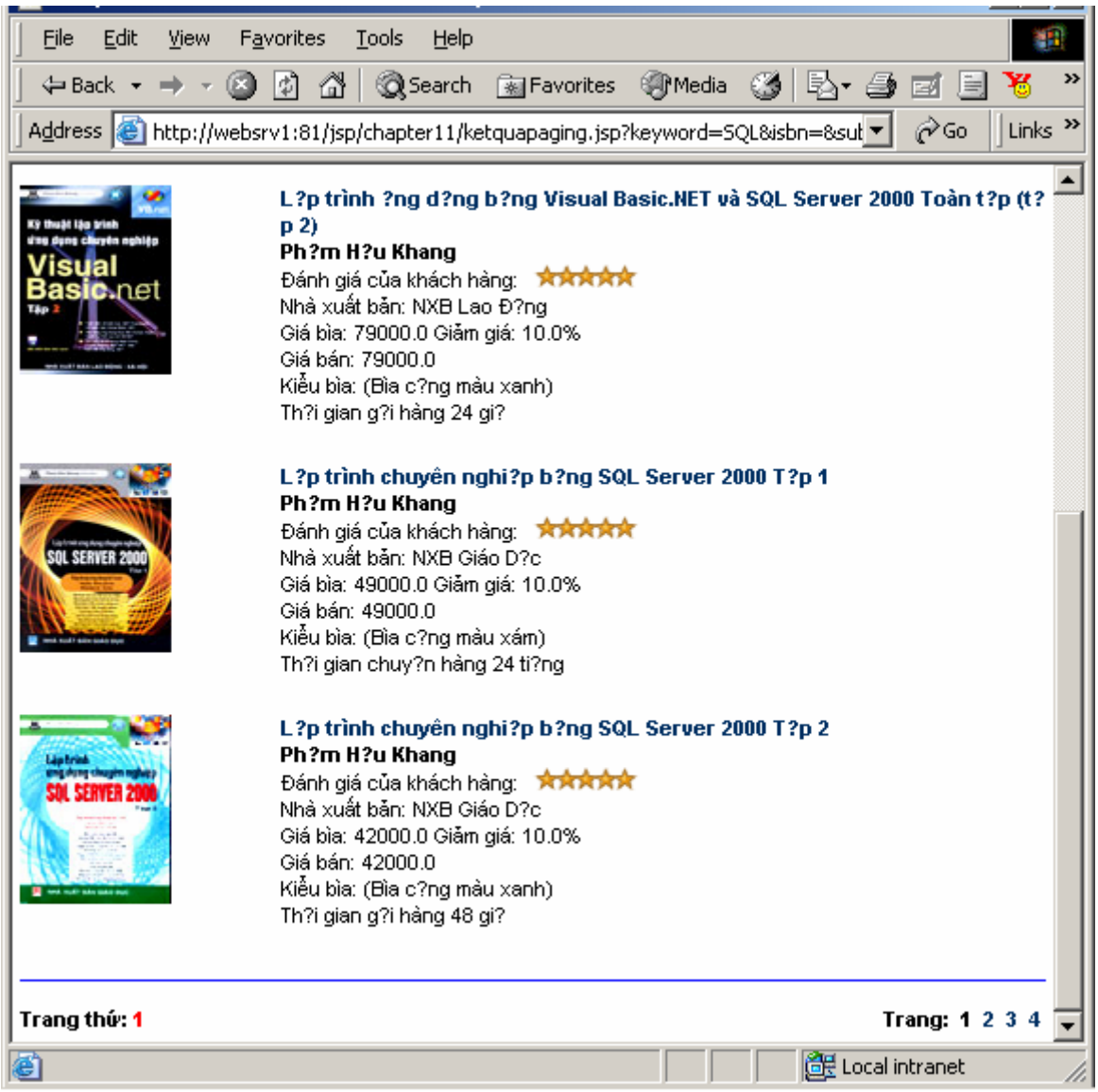
Khi trình bày dữ liệu trên trang kết quả tìm kiếm hay liệt kê, số mẩu tin có thể vượt quá không gian trang *web* có thể thể hiện, chính vì vậy phân trang là điều cần thiết.

Giả sử rằng, bạn muốn trình bày bày 10 mẩu tin trên mỗi trang, mỗ lần xuất hiện 5 trang, nếu số trang nhiều hơn 5 sẽ xuất hiện thêm liên kết *Next* cho phép trình bày 5 trang kế tiếp.

Tương tự như vậy, khi người sử dụng đang đứng sau 5 trang đầu tiên, xuất hiện liên kế *Previous* cho phép người sử dụng trở về 5 trang trước đó.

Để làm điều này, chúng ta sử dụng những thuộc tính và phương thức của đối tượng *ResultSet* bằng *Absolute(i)*: Trỏ con trỏ đến mẩu tin đầu tiên của trang thư i.

Chẳng hạng, khi liệt kê danh sách sản phẩm theo *alphabet*, số lượng mẩu tin là 23 mẩu tin, bạn chia chúng ra thành 5 trang mỗi trang 5 mẩu tin và mỗi lần trình bày 3 trang như hình 13-7.



Hình 11-6: Phân trang

Để làm điều này, bạn khai báo hai biến *record* và *pages* tương ứng với số mẩu tin cần trình bày trên một trang và số trang cần trình bày trong một phân đoạn:

```
int pageCurrent=1;
//trang lay tu QueryString
int pages=5;
if (request.getParameter("pages")!=null)
{
    try {

    pages=Integer.parseInt(request.getParameter("pages
"));
    }
    catch(Exception ex)
    {
      pages=5;
    }
}
int record=5;// moi lan hien 5 mau tin
if (request.getParameter("rows")!=null)
{
    try {

    record=Integer.parseInt(request.getParameter("rows
"));
    }
    catch(Exception ex)
    {
      record=5;
    }
}
```

Kế đến, trước khi sử dụng phương thức *executeQuery* của đối tượng *Statement* và sử dụng thuộc tính:

```
if(pageCurrent>1)
{
        row=(pageCurrent-1)*record;
        rst.absolute(row);
}
```

Mỗi khi người sử dụng triệu gọi trang *JSP* này, bằng cách sử dụng đối tượng *Request* để lấy số trang từ tham số *page*:

```
    if (request.getParameter("page")!=null)
{
    try {

    pageCurrent=Integer.parseInt(request.getParameter(
"page"));
    }
    catch(Exception ex)
    {
      pageCurrent=1;
    }
}
```

Lưu ý rằng, mặc định trang thứ 1 được gọi trong trường hợp triệu gọi trang *JSP* đầu tiên.

Sau khi có được trang thứ i cần trình bày, bằng cách sử dụng phương thức *Absolute* để định vị con trỏ đến mẩu tin đầu tiên của trang thứ i:

```
if(pageCurrent>1)
       {
         row=(pageCurrent-1)*record;
         rst.absolute(row);
       }
      viewrow=1;
      while(rst.next()&& viewrow<=record)
         {
```

Khi con trỏ được định vị trại mẩu tin đầu tiên của trang thư i (biến *PageCurrent*), bạn chỉ cần trình bày đúng với số mẩu tin dựa trên biến *record*:

```
    while(rst.next()&& viewrow<=record)
      {

      %>
      <tr>
            <td width="25%" valign="top">
       <img src="hinh/
        <%=rst.getString("ImagePathSmall")%>"
       height="100"></td>
          <td width="75%" valign="top"> <b>
      <a href="chitiet.jsp?itemid=
      <%=rst.getString("ItemID")%>
      &aid=<%="AuthID"%>">
      <font face="Arial">
      <%=rst.getString("ItemName")%>
```

```
        </font></a></b><br>
             <strong>
          <%=rst.getString("AuthorName")%></strong><br>
               nh gí c a kích ng:
        <%String CR=rst.getString("CustomerRating");%>
        <img src="icons/stars
        <%=(CR==null)?"A1":CR%>.gif"
        width="64" height="12"><br>
             N xu t b n:
          <%=rst.getString("ProName")%><br>
               Gí bìa: <%=rst.getString("ListPrice")%>
          Gi m gí: <%=rst.getString("SalesDiscount")%>%
          <br>
          Gí n: <%=rst.getString("SalesPrice")%><br>
          Ki u bìa: <%=rst.getString("ItemStyle")%> <br>
               <%=rst.getString("Available")%>
          <br><br></td>
           </tr>
      <%viewrow++;
      }
      rst.close();
    }
    catch (Exception e){
      out.println(e);
    }
    smt.close();
    cn.close();
```

Sau khi có được số trang *Pages*, số *record* mẩu tin trên một trang, mỗi lần trình bày *page* trang và trang hiện hành *PageCurrent*, bạn có thể khai báo đoạn mã *JSP* để có được chuỗi phân trang như ví dụ 13-7.

Ví dụ 13-7: Phân trang

```
<tr>
 <td width="25%" height="19" align="left" valign="middle"><strong>Trang
th :<font color="#FF0000"> <%=pageCurrent%></font></strong></td>
   <td width="75%" height="19" align="right"><b>
%if (totalRecords>record){%>Trang: <%=getPage(pageCurrent,record,
pages,totalRecords,  totalPages, "ketqua.jsp?"+qryString)%><%}%></b></td>
</tr>
```

Như vậy, để sử dụng chung đoạn phân trang này cho mọi trang *JSP* có phân trang, bạn khai báo ví dụ 13-7 thành một tập tin *JSP* với hai biến *record* và *page* cùng với hàm *getPage* như ví dụ 11-8.

Ví dụ 13-8: phương thức trả về chuỗi phân trang

```
<%!
public String getPage(int pageCurrent,int records, int pages,
   int totalRecords, int totalPages, String strURL)
   {
        int startNo=0;int endNo=0;int Segment=0;
        String strPage="";
        totalPages=totalRecords/records;
        if((totalRecords%records)>0)
                totalPages+=1;
        if(totalPages<=1)
                strPage="";
        else
        {
                Segment=pageCurrent/pages;
                if ((pageCurrent % pages)>0) Segment+=1 ;
                startNo=Segment*pages-pages+1;
                if (Segment*pages>totalPages) endNo=totalPages;
                else endNo=Segment*pages;
                if (Segment>1)
                {
                strPage=strPage + "<A href='"+strURL;
                strPage=strPage + "&page=" ;
                strPage=strPage + (Segment-1)*pages + "');"  ;
                        strPage=strPage + ">Previous</A>  " ;
                }
                for (int No=startNo;No<=endNo;No++)
                {
                if (No==pageCurrent)
                strPage=strPage + " "  +  No + " ";
                else
                   {
                strPage = strPage + " <A
                        href='"+strURL+"&page=";
                strPage=strPage +  No + "');>" + No + "</A> ";
                }
                }
                if (totalPages>pages*Segment)
                {
                strPage = strPage + " <A href='"
```

```
                        +strURL+"&page=";
                                strPage = strPage + ((Segment*pages) + 1);
                                strPage = strPage +  "');>Next</A>" ;
                        }
                }
                return strPage;
        }
%>
```

Do hàm *getPage* trong trang *paging.jsp* sẽ được dùng chung cho mọi trang *JSP*, chính vì vậy bạn khai báo tham số *querystring* để nhận tên của trang *JSP* có sử dụng hàm *getPage* khi phân trang.

Chú ý rằng, nếu người sử dụng chọn vào chữ *L* lập tức danh sách các cuốn sách có tiền tố là *L* được trình bày, như vậy khi người sử dụng chọn trang thứ *i* thì ký tự *L* cũng sẽ được gán cho tham số *al* trên *QueryString*.

Tương tự như vậy trong trường hợp trình bày kết quả tìm kiếm, bạn phải bảo đảm các tham số trong trang tìm kiếm được xuất hiện mỗi khi người sử dụng chọn trang thứ i.

Ví dụ, chúng ta xây dựng trang tìm kiếm nâng cao có tên *timkiempaging.jsp* có cấu trúc như trang *ketquapaging.jsp*, khi người sử dụng nhập các tiêu chuẩn tìm kiếm và nhấn nút *Tìm Kiếm* thì trang *ketquapaging.JSP* được triệu gọi, kết quả tìm kiếm xuất hiện như hình 11-6.

5. KẾT LUÂN

Chúng ta vừa tập trung tìm hiểu cách cài đặt phân trang và điều hướng trong trang JSP.

Trong chương kế tiếp, chúng ta tiếp tục tìm hiểu cách trình bày thông tin chi tiết của sản phẩm, xây dựng giỏ hàng và cho phép người sử dụng đơn đặt hàng qua mạng.

Môn học: Java Server Pages

BÀI 12: CHI TIẾT SẢN PHẨM

Sau khi trình bày kết quả tìm kiếm, nếu người sử dụng chọn một sản phẩm nào đó thì bạn trình bày các thông tin chi tiết của sản phẩm đó. Ngoài những thông tin chi tiết của sản phẩm, bạn nên trình bày các thuộc tính liên quan khác như sản phẩm bổ sung, sản phẩm thay thế, sản phẩm được các khách hàng khác mua khi chọn sản phẩm này.

Các vấn đề chính sẽ được đề cập:

- ✓ Chi tiết độ phân giải
- ✓ Sản phẩm bổ sung
- ✓ Sản phẩm của khách hàng thường mua

1. CHI TIẾT SẢN PHẨM

Khi thiết kế dữ liệu, bảng lưu trữ thông tin chi tiết của sản phẩm có thể có nhiều cột, nếu người sử dụng cung cấp một từ khoá để tìm kiếm mẩu tin thoả yêu cầu này, bạn phải khai báo phát biểu *Select* có mệnh đề *Where* dựa trên các cột dữ liệu cho phép tìm.

Sau khi khách hàng chọn một sản phẩm trên trang kết quả tìm kiếm, toàn bộ thông tin chi tiết của sản phẩm đó cần được trình bày.

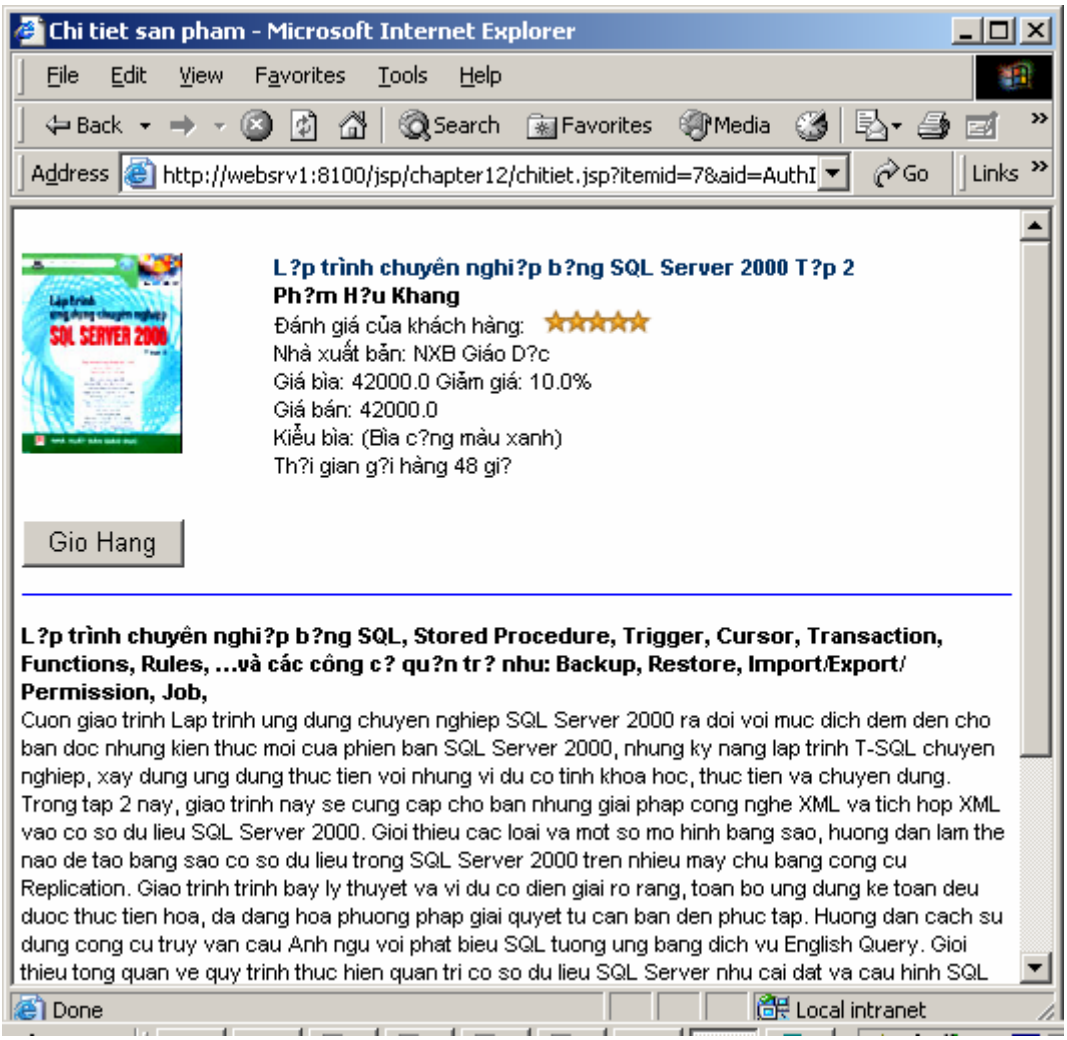
Những thông tin trình bày tuỳ thuộc vào giải thuật của bạn, chẳng hạn, ngoài các thông tin chi tiết của sản phẩm, bạn trình bày các thông tin liên quan như sản phẩm thay thế, sản phẩm bổ sung và những sản phẩm mà người sử dụng khác mua chúng khi mua sản phẩm này.

Để trình bày thông tin chi tiết của sản phẩm, bạn có thể khai báo để định nghĩa mệnh đề Where như ví dụ 12-1.

Ví dụ 12-1: Khai báo mệnh đề Where

```
<%
String strShow="";
String strWhere=" ";
String id=request.getParameter("itemid");

if (id==null)
    {
      response.sendRedirect("ketqua.jsp");
    }
else
    {
      strWhere+= " and I.ItemID= '" + id + "'";
```

```
        }
%>
```



Hình 12-1: Chi tiết sản phẩm

Sau đó, bằng cách định nghĩa phát biểu Select như ví dụ 12-2, bạn có thể đọc dữ liệu để trình bày như hình 12-1.

Ví dụ 12-2: Định nghĩa phát biểu Select

```
<%
Connection cn;
Statement smt;
ResultSet rst;
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
cn = DriverManager.getConnection(odbc,sysuser,syspwd);
String strSQL="select ImagePathSmall,ItemName,AuthorName, ";
```

```
strSQL+="CustomerRating,ProName,ListPrice,SalesDiscount, ";
strSQL+="SalesPrice, ItemStyle,Available,cast(Subjects ";
strSQL+="As varchar(8000)) as Subj,cast(Descriptions ";
strSQL+="As varchar(8000)) as Description";
strSQL+=" from tblitems I inner join tblitemdetails D on ";
strSQL+=" I.ItemID=D.ItemID right join tblAuthors A ";
strSQL+="on A.AuthorID=I.AuthID right join tblProductions ";
strSQL+="P on P.ProID=I.ProID where Activate=1 " ;
strSQL += " " + strWhere;
smt = cn.createStatement();
try{
```

Bằng cách sử dụng các phương thức đọc dữ liệu của từng cột, như ví dụ 12-3.

Ví dụ 12-3: Chi tiết sản phẩm

```
<tr>
<td width="25%" valign="top">
        <img src="hinh/<%=rst.getString("ImagePathSmall")%>"
            height="100"></td>
              <td width="75%" valign="top"> <b>
           <a href="chitiet.jsp?itemid=<%=id%>&aid=<%="AuthID"%>">
           <%Name=rst.getString("ItemName");%>
           <font face="Arial"><%=Name%>
           </font></a></b><br>
        <strong><%=rst.getString("AuthorName")%></strong><br>
                  Đánh giá của khách hàng:
           <%CR=rst.getString("CustomerRating");%>
           <img src="icons/stars<%=(CR==null)?"A1":CR%>.gif"
           width="64" height="12"><br>
                Nhà xuất bản: <%=rst.getString("ProName")%><br>
                  Giá bìa: <%=rst.getString("ListPrice")%>
        Giảm giá: <%=rst.getString("SalesDiscount")%>%<br>
                  Giá bán: <%=rst.getString("SalesPrice")%><br>
                Kiểu bìa: <%=rst.getString("ItemStyle")%> <br>
                  <%=rst.getString("Available")%> <br><br></td>
    </tr>
```

Sau đó, khai báo nút cho phép người sử dụng thêm sản phẩm này vào giỏ hàng cùng với các thẻ hidden liên quan.

Ví dụ 12-4: Khai báo nút để thêm sản phẩm vào giỏ hàng

```
<form action=them.jsp method=post>
    <tr>
      <td height="19" colspan="2">
      <input type=hidden value="<%=id%>" name=id>
      <input type=hidden value="<%=Name%>" name=name>
                  <input type=submit value="Gio Hang">
        </td>
       </tr>
```

```
</form>
```

Chẳng hạn, trong trường hợp này chúng ta khai báo thẻ hidden lưu trữ mã và tên của sản phẩm đang xem chi tiết la id, name và price.

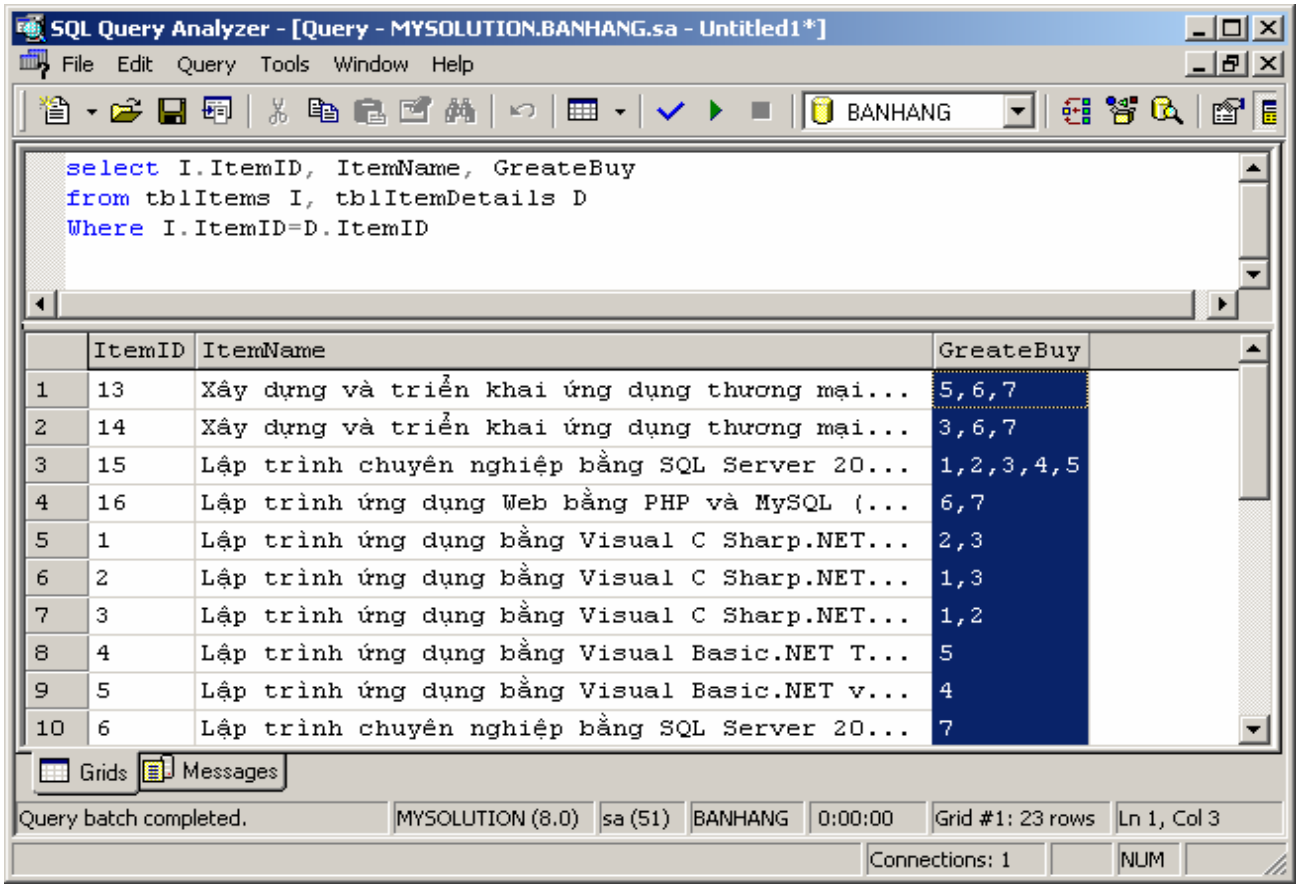
2. SẢN PHẨM BỔ SUNG

Sản phẩm bổ sung là sản phẩm được sử dụng để hỗ trợ cho sản phẩm trước đó co hiệu quả hoàn thiện hơn. Chẳng hạn, khi bạn mua một cuốn sách Visual Basic 6.0 thì nên mua thêm cuốn sách SQL Server bở vì cuốn sách SQL Server cung cấp kiến thức bổ sung để học Visual Basic 6.0 tốt hơn.

Để làm điều này, bạn khai báo thêm cột GreateBuy lưu trữ mã của những sản phẩm mà nhà quản lý cho là có thể bổ sung cho sản phẩm đó. Ví dụ những cố sách có sản phẩm bổ sung là như sau:

```
select I.ItemID, ItemName, GreateBuy
from tblItems I, tblItemDetails D Where I.ItemID=D.ItemID
```

Kết quả trình bày như hình 12-2 trong Query Analyzer như sau.

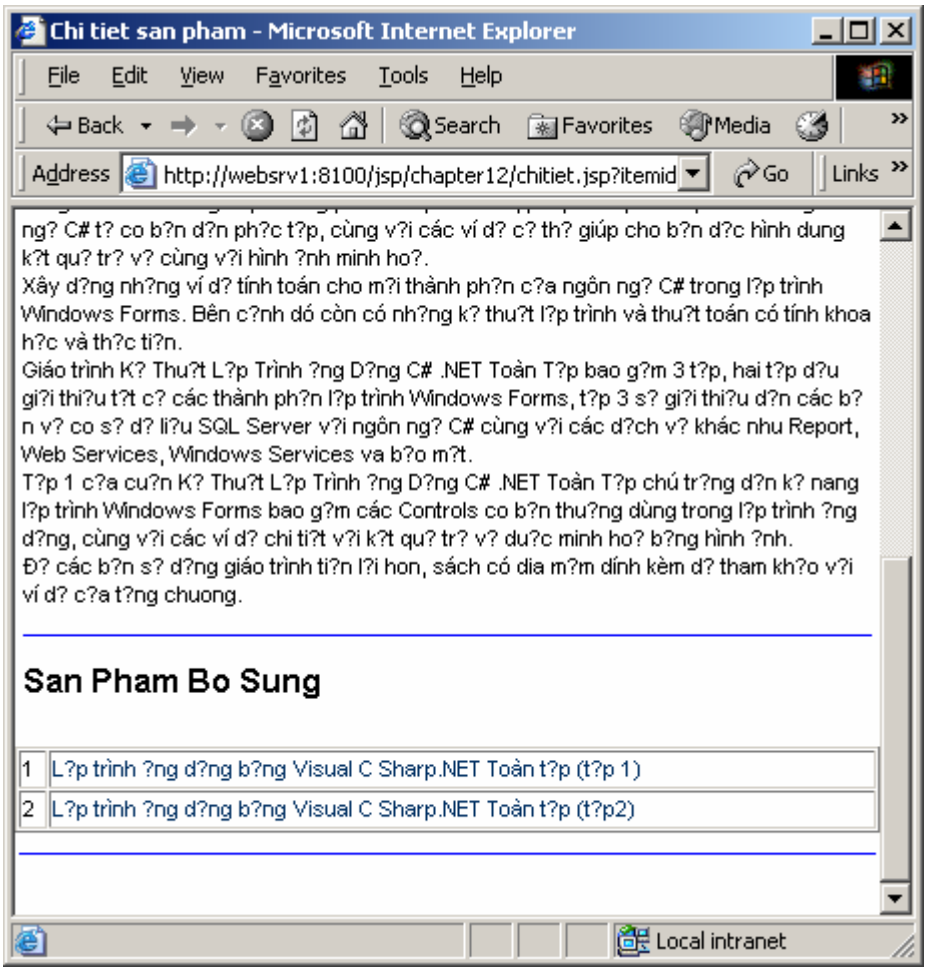


Hình 12-2: Sản phẩm bổ sung

Bằng cách lấy giá trị trong cột GreateBuy của từng sản phẩm, bạn có thể khai báo chuỗi Select ứng với những sản phẩm có mã nằm trong chuỗi giá trị này:

```
String GreateBuy="";

...

GreateBuy=rst.getString("GreateBuy");
...

strSQL="select I.ItemID, ItemName, GreateBuy ";
strSQL+=" from tblItems I, tblItemDetails D
strSQL+=" Where I.ItemID=D.ItemID";
strSQL+=" and I.ItemID in (" + GreateBuy + ")";
```

Sau đó duyệt tất cả sản phẩm và trình bày như hình 12-3.



Hình 12-3: Sản phẩm bổ sung

Để làm điều này, bạn khai báo như sau:

```
...
strSQL="select I.ItemID, ItemName, GreateBuy ";
strSQL+=" from tblItems I, tblItemDetails D
strSQL+=" Where I.ItemID=D.ItemID";
strSQL+=" and I.ItemID in (" + GreateBuy + ")";
rst=smt.executeQuery(strSQL);
while(rst.next())
    {
      id=rst.getString("ItemID");
    %>
    <tr>
          <td height="19"><%=id%></td>
      <td height="19"><a href="chitiet.jsp?id=<%=id%>">
      <%=rst.getString("ItemName")%></a></td>
        </tr>

    <%
}
rst.close();

...
```

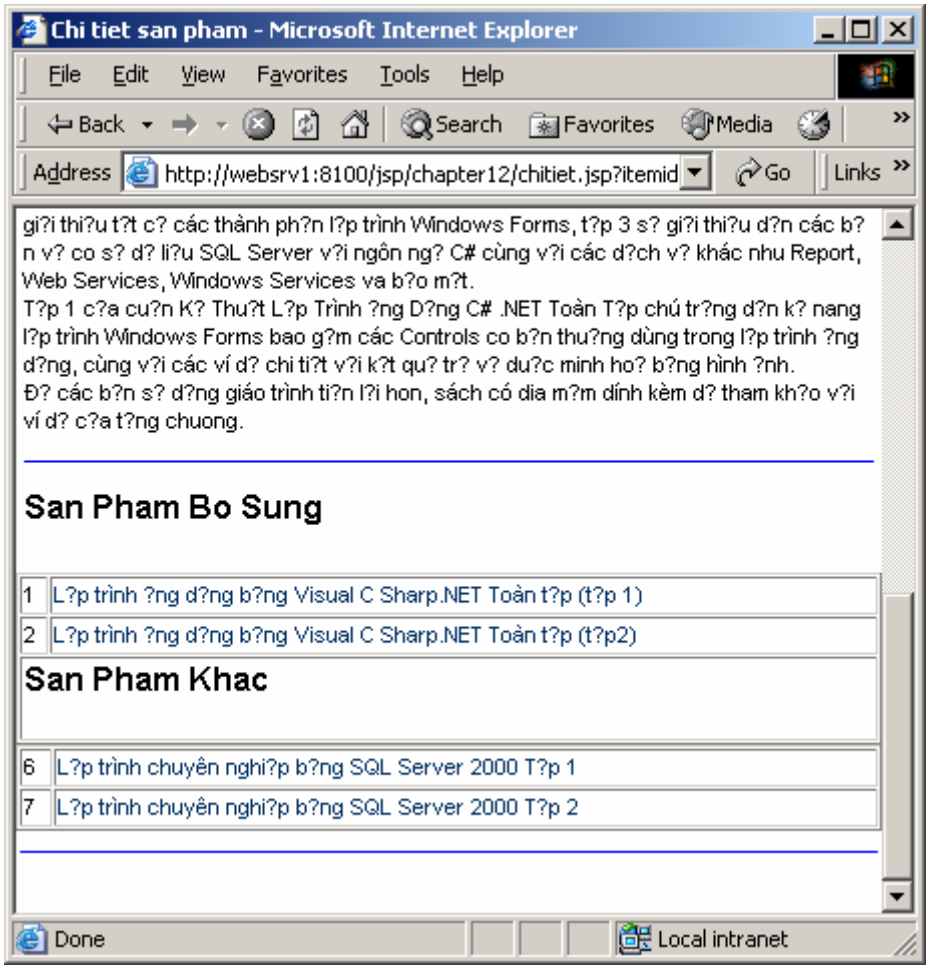
3. SẢN PHẨM CỦA KHÁCH HÀNG THƯỜNG MUA

Ngoài danh sách các sản phẩm bổ sung, nhà quản lý nên cung cấp những cuốn sách mà thường khách hàng mua đính kèm khi mua cuốn sách hiện hành. Để làm điều này, bạn khai báo cột dữ liệu trong bảng tbltemDetails có tên là Relations.

Bằng cách khai báo biến Relations và lấy giá trị từ cột này ra, sau đó khai báo phát biểu Select để liệt kê danh sách các sản phẩm này.

```
String Relations="";

...


Relations=rst.getString("Relations");

...

strSQL="select I.ItemID, ItemName, GreateBuy ";
strSQL+=" from tblItems I, tblItemDetails D Where " ;
strSQL+=" I.ItemID=D.ItemID";
strSQL+=" and I.ItemID in (" + Relations + ")";
rst=smt.executeQuery(strSQL);
```

Kết quả trình bày như hình 12-4.



Hình 12-4: Danh sản phẩm liên quan khác

Để làm điều này, bạn khai báo đoạn chương trình như sa:

```
<table width="100%" border="1"
    cellspacing="1" cellpadding="1">
    <%
    strSQL="select I.ItemID, ItemName, GreateBuy ";
    strSQL+=" from tblItems I, tblItemDetails D ";
    strSQL+=" Where I.ItemID=D.ItemID";
    strSQL+=" and I.ItemID in (" + Relations + ")";
    rst=smt.executeQuery(strSQL);
    while(rst.next())
    {
      id=rst.getString("ItemID");
    %>
    <tr>
          <td height="19"><%=id%></td>
```

```
        <td height="19"><a href="chitiet.jsp?id=<%=id%>">
        <%=rst.getString("ItemName")%></a></td>
          </tr>

      <%
      }
      rst.close();
```

4. KẾT LUÂN

Chúng ta vừa tập trung tìm hiểu cách cài đặt giỏ hàng, cho phép người sử dụng thêm, cập nhật số lượng và huỷ giỏ hàng bằng JSP.

Môn học: Java Server Pages

BÀI 13: GIỎ HÀNG VÀ CHI TIẾT SẢN PHẨM

Bất kỳ ứng dụng thương mại điển tử nào có bán hàng qua mạng đều có chức năng giỏ hàng, giỏ hàng giúp cho người sử dụng lưu trữ sản phẩm trong khi đang chọn những sản phẩm khác trên mạng.

Sau khi kết thúc quá trình thêm vào giỏ hàng, người sử dụng có thể xem, xoá hay cập nhập số lượng của giỏ hàng.

Ngoài ra, người sử dụng quyết định đặt hàng sau khi xem xét giỏ hàng.

Các vấn đề chính sẽ được đề cập:

- ✓ Xây dựng chức năng giỏ hàng
- ✓ Cập nhật, xoá giỏ hàng
- ✓ Đặt hàng

1. XÂY DỰNG GIỎ HÀNG

Sau khi khách hàng chọn một sản phẩm trên trang kết quả tìm kiếm, toàn bộ thông tin chi tiết của sản phẩm đó cần được trình bày.

Hình 13-1: Chi tiết sản phẩm

Để xây dựng giỏ hàng trong JSP bạn có thể sử dụng nhiều cách khác nhau như vector, session, cơ sở dữ liệu, cookie. Tuy nhiên, đối với bài học này chúng ta sử dụng cách dùng session kết hợp với mảng hai hciều để xây dựng và quản lý giỏ hàng trong JSP.

Khi người sử dụng chọn nút "Gio Hang" trong trang chi tiết sản phẩm, thì trang them.jsp được triệu gọi. Trang này lấy 3 giá trị chính khai báo hidden trong trang chi tiết là id, name,price và update để xây dựng giỏ hàng.

```
String id=getVal(request.getParameter("id"),"");
String name=getVal(request.getParameter("name"),"");
String price=getVal(request.getParameter("price"),"0");
String update=getVal(request.getParameter("update"),"");
```

Bằng cách sử dụng mảng dữ liệu hai chiều gồm tối đa 10 sản phẩm và mỗi sản phẩm chúng ta lưu trữ 4 thông tin chính là mã, tên, số lượng và đơn giá của sản phẩm.

```
String cart[][] =new String[10][4];
int cartno=0;
```

Trong đó, biến cart là mảng lưu trữ thông tin của sản phẩm, cartno là số sản phẩm có trong mảng cart.

Trong trường hợp lần đầu tiên người sử dụng thêm sản phẩm vào giỏ hàng, chúng ta dựa vào biến session có tên session.getValue("cartno") với giá trị là 0 thì thêm sản phẩm vào giỏ hàng tại phần tử thứ 0, còn các phần tử còn lại chúng ta khởi tạo giá trị là rỗng như sau:

```
}else
{
    if(update.equals(""))
    {
      cart[0][0]=id;
      cart[0][1]=name;
      cart[0][2]="1";
      cart[0][3]=price;
      for (int j=cartno+1;j<cart.length;j++)
      {
        cart[j][0]="";
        cart[j][1]="";
        cart[j][2]="";
        cart[j][3]="";
      }
    }
}
```

Nếu sản phẩm được thêm vào trong lần kế tiếp, chúng ta lấy giá trị từ biến session có tên cart và cartno, sau đó ép kiểu sang mảng và kiểm tra nếu mã sản phẩm chưa tồn tại thì thêm vào.

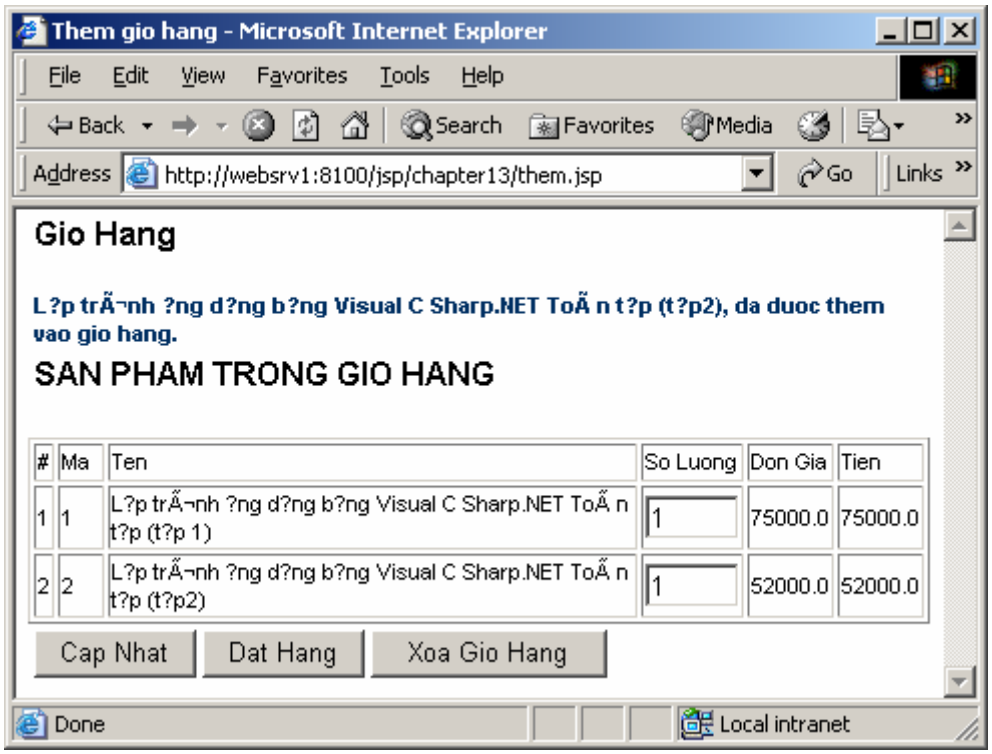
```
if(cartno<=9)
      {
        boolean check=false;
        for (int j=0;j<cartno;j++)
        {
          if(cart[j][0].equals(id))
          {
                    check=true;
          }
        }
        if (check==false)
        {
          cart[cartno][0]=id;
          cart[cartno][1]=name;
          cart[cartno][2]="1";
          cart[cartno][3]=price;
          cartno++;
        }
      }
```

Lưu ý rằng, chỉ cho phép thêm vào giỏ hàng tối đa 10 sản phẩm, chính vì vậy chúng ta chỉ thêm khi giá trị của biến cartno nhỏ hoặc bằng 9.
Sau đó, chúng ta trình bày danh sách những sản phẩm đã có trong giỏ hàng bằng cách sử dụng vòng lặp for như sau:

```
<table width=450 border="1" align=center>
    <tr>
          <td height="19">#</td>
          <td height="19">Ma</td>
          <td height="19"> Ten</td>
          <td height="19">So Luong</td>
          <td height="19">Don Gia</td>
          <td height="19">Tien</td>
    </tr>
    <%for (int j=0;j<cart.length;j++)
    {
    if(cart[j][0]==null || cart[j][0].equals("")) break;
    %>
    <tr>
          <td width=5><%=j+1%></td>
          <td width=20> <%=cart[j][0]%></td>
          <td width=300><%=cart[j][1]%></td>
          <td width=2>
          <input size=2 maxlength=3 value="<%=cart[j][2]%>"
          name="chk<%=cart[j][0]%>">
        </td>
        <td width=25><%=cart[j][3]%></td>
```

```
        <td width=25>
        <%= Integer.parseInt((String)cart[j][2])*
        Double.parseDouble((String)cart[j][3])%></td>
         </tr>
    <%}%>
    </table>
```

Kết quả trình bày như hình 13-2.



Hình 13-2: Danh sản phẩm trong giỏ hàng

Lưu ý: Trong trường hợp không có sản phẩm được thêm vào giỏ hàng hoặc trường hợp cập nhật thì trang timkiem.jsp sẽ được triệu gọi.

2. CẬP NHẬT GIỎ HÀNG

Mặc định của mỗi sản phẩm khi thêm vào giỏ hàng là 1, sau khi trinh bày như hình 13-2, bạn khai báo nút Cập Nhật cho phép người sử dụng thay đổi số lượng trong giỏ hàng cho mỗi sản phẩm.

Để làm điều này, trước tiên khai báo thẻ input có tên là chk và mã sản phẩm như sau:

```
<input size=2 maxlength=3
value="<%=cart[j][2]%>" name="chk<%=cart[j][0]%>">

...
```

```
<input type=submit name=update value="Cap Nhat">
```

Sau đó, bạn khai báo biến update để lấy giá trị của nút Update khi người sử dụng submit để phân biệt trường hợp cập nhật số lượng trong giỏ hàng với trường hợp khi thêm sản phẩm vào giỏ hàng từ trang chi tiết.

```
String update=getVal(request.getParameter("update"),"");
```

Trong trường hợp này, dựa vào mã sản phẩm trong giò hàng, chúng ta cập nhật sl như sau:

```
    /*trường hợp thêm mới từ trang chi tiết*/
else
    {
      for (int j=0;j<cartno;j++)
        {
        cart[j][2]=getVal(request.getParameter("chk" +
                                    cart[j][0]),"1");
        }
    }
```

Lưu ý rằng, chúng ta chỉ duyệt số phần tử từ 0 đến số phần tử đang có, và cập nhật phần tử số lượng bằng sớ lượng từ màn hình.

Ngoài ra, bạn có thể khai báo đoạn Javascript để kiểm soát số nhập hợp lệ trên từng phần số lượng của sản phẩm.

Chẳng hạn, chúng ta thay đổi số lượng của sản phẩm thứ nhất lên 2 và sản phẩm thứ hai thành 3, kết quả trình bày như hình 13-3.

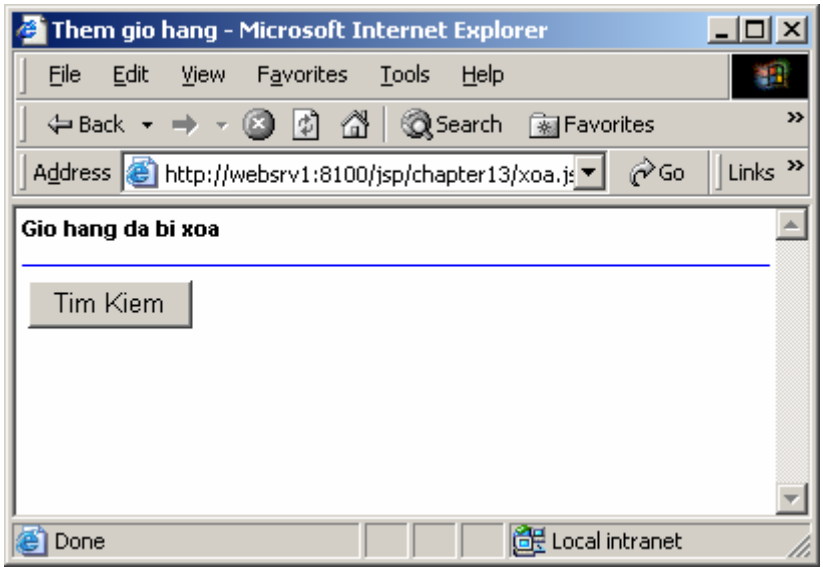
Hình 12-3: Cập nhật số lượng

3. HUỶ GIỎ HÀNG

Để huỷ giò hàng, bạn sử dụng phương thức removeAttribute để huỷ hai đối tượng session có tên cart và cartno. Để làm điều này, bạn khai báo trang xoa.jsp được triệu gọi, bạn khai báo như sau:

```
<%  session.removeAttribute("cart");
    session.removeAttribute("cartno");%>
```

Sau khi xoá giỏ hàng, thông báo xuất hiện như hình 13-4.



Hình 13-4: Xoá giỏ hàng

4. KẾT LUÂN

Chúng ta vừa tập trung tìm hiểu cách cài đặt giỏ hàng, cho phép người sử dụng thêm, cập nhật số lượng và huỷ giỏ hàng bằng JSP.