JDBC

Shin-Jie Lee (李信杰)
Assistant Professor
Computer and Network Center
Department of Computer Science and Information Engineering
National Cheng Kung University

Java and Database Connections: SQL

- Structured Query Language (SQL) is a language for formulating queries for a relational database
  - SQL is not a part of Java, but Java has a library (JDBC) that allows SQL commands to be embedded in Java code
- SQL works with relational databases
  - Most commercial database management systems are relational databases

Java and Database Connections: SQL

- A relational database can be thought of as a collection of named tables with rows and columns
  - Each table relates together certain information, but the same relationship is not repeated in other tables
  - However, a piece of information in one table may provide a bridge to another

Relational Database Tables (Part 1 of 3)

Relational Database Tables

Names

AUTHOR	AUTHOR_ID	URL
Adams, Douglas	1	http:// ...
Simmons, Dan	2	http:// ...
Stephenson, Neal	3	http:// ...

(continued)

Relational Database Tables (Part 2 of 3)

Relational Database Tables

Titles

TITLE	ISBN
Snow Crash	0-553-38095-8
Endymion	0-553-57294-6
The Hitchhiker's Guide to the Galaxy	0-671-46149-4
The Rise of Endymion	0-553-57298-9

(continued)

Relational Database Tables (Part 3 of 3)

Relational Database Tables

BooksAuthors

ISBN	AUTHOR_ID
0-553-38095-8	3
0-553-57294-6	2
0-671-46149-4	1
0-553-57298-9	2

A Sample SQL Command

- The following is a sample SQL command that can be used in conjunction with the tables from the previous slide:

```
SELECT Titles.Title, Titles.ISBN,
  BooksAuthors.Author_ID
FROM Titles, BooksAuthors
WHERE Titles.ISBN = BooksAuthors.ISBN
```

- The above command will produce the table shown on the following slide

Result of SQL Command in Text

Result of SQL Command in Text

Result

TITLE	ISBN	AUTHOR_ID
Snow Crash	0-553-38095-8	3
Endymion	0-553-57294-6	2
The Hitchhiker's Guide to the Galaxy	0-671-46149-4	1
The Rise of Endymion	0-553-57298-9	2

Common SQL Statements (1 of 2)

`CREATE TABLE`	Create a new table named `newtable` with fields `field1`, `field2`, etc. Data types are similar to Java and include: `int`, `bigint`, `float`, `double`, and `var(size)` which is equivalent to a String of maximum length `size`.	`CREATE TABLE newtable (field1 datatype, field2 datatype, ...)`
`INSERT`	Insert a new row into the table `tableName` where `field1` has the value `field1Value`, `field2` has the value `field2Value`, etc. The data types for the values must match those for the corresponding fields when the table was created. String values should be enclosed in single quotes.	`INSERT INTO tableName VALUES (field1Value, field2Value, ...)`

Common SQL Statements (2 of 2)

`UPDATE`	Change the specified fields to the new values for any rows that match the `WHERE` clause. `Op` is a comparison operator such as =, <> (not equal to), <, >, etc.	`UPDATE tableName`
		`SET field1 = newValue,`
		`field2 = newValue, ...`
		`WHERE fieldName Op`
		`someValue`

`SELECT`	Retrieve the specified fields for the rows that match the `WHERE` clause. The `*` may be used to retrieve all fields. Omit the `WHERE` clause to retrieve all rows from the table.	`SELECT field1, field2`
		`FROM tableName`
		`WHERE fieldname Op`
		`someValue`

SQL Examples

- CREATE TABLE names(author varchar(50), author_id int, url varchar(80))
- INSERT INTO names VALUES ('Adams, Douglas', 1, 'http://www.douglasadams.com')
- UPDATE names SET url = 'http://www.douglasadams.com/dna/bio.html' WHERE author_id = 1
- SELECT author, author_id, url FROM names
- SELECT author, author_id, url FROM names WHERE author_id > 1

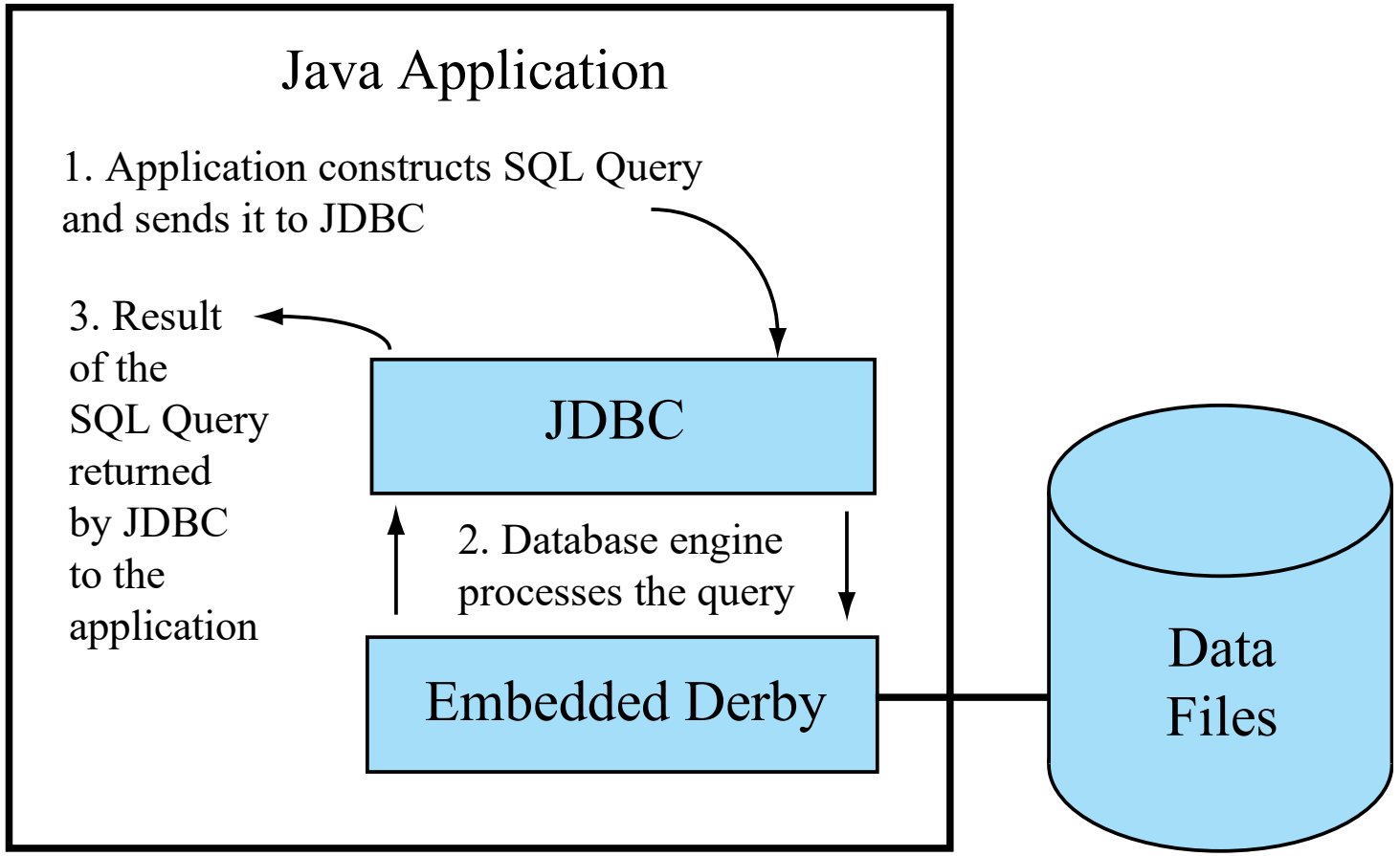
JDBC

- *Java Database Connectivity (JDBC)* allows SQL commands to be inserted into Java code
  - In order to use it, both JDBC and a database system compatible with it must be installed
  - A JDBC driver for the database system may need to be downloaded and installed as well
- Inside the Java code, a connection to a database system is made, and SQL commands are then executed

Java DB

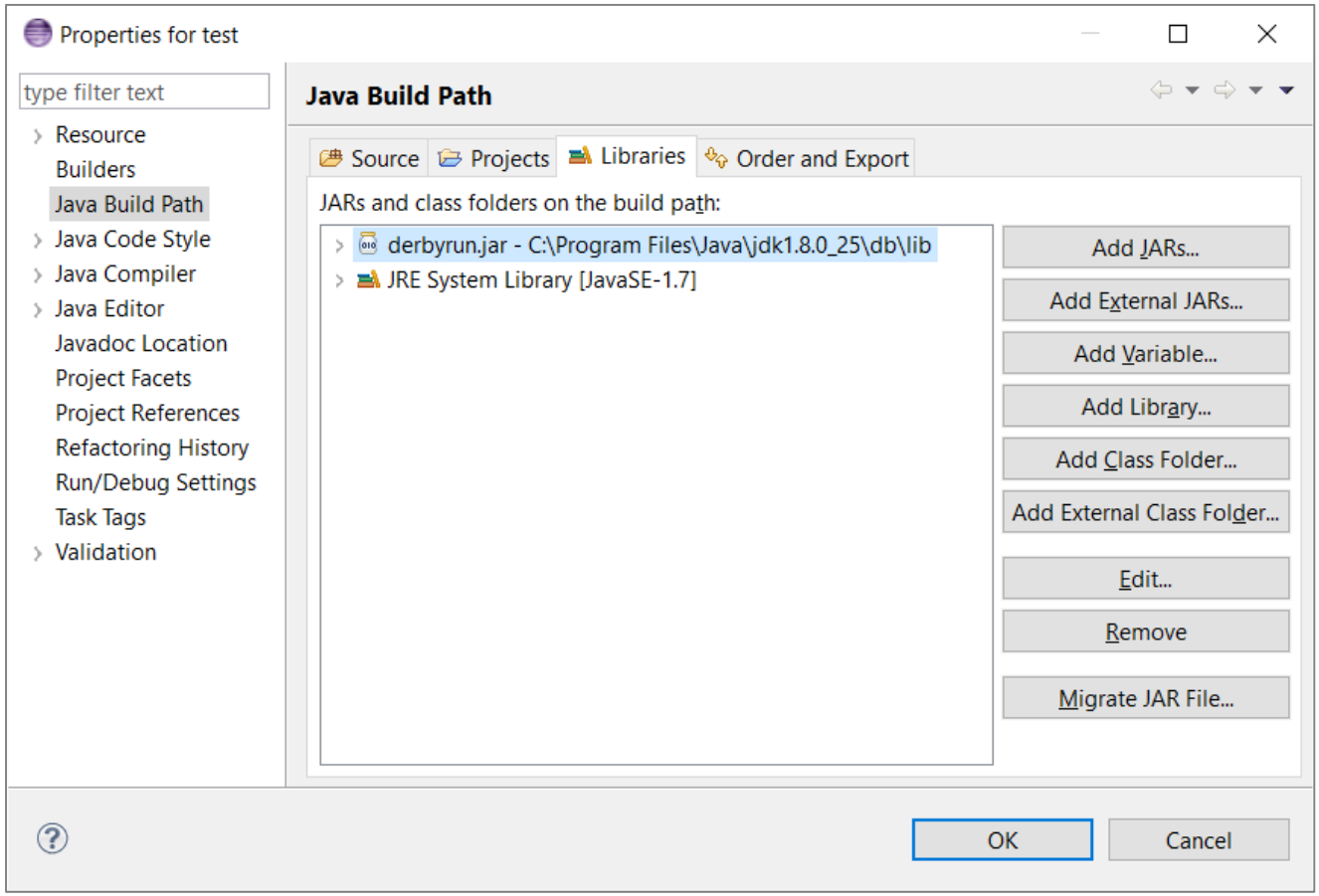
- In the following examples we will use *Java DB*
  - Packaged with version 6 or higher of the Java SDK
  - Based on the open source database known as *Apache Derby*
  - See http://www.oracle.com/technetwork/java/javadb/index.html

Data Flow of an Embedded Derby Application



Lab (Setup for Embedded Mode)

- Add "c:\Program Files\Java\jdkxxx\db\lib\derbyrun.jar" to the Java Build Path



Lab (Create DB)

```
import java.sql.*;

public class CreateDB {

 public static void main(String[] args) {
  try {
    Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
    Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase;create=true");

    Statement s = conn.createStatement();
    s.execute("CREATE TABLE names (author varchar(50), author_id int, url varchar(80))");

    System.out.println("Database is created with a table.");

    conn.close();
   } catch (Exception err) {
       err.printStackTrace();
   }
  }
}
```

Lab (Insert Data)

```
import java.sql.*;

public class InsertData {
   public static void main(String[] args) {
      try {
         Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
         Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase");

         Statement s = conn.createStatement();
         s.execute("INSERT INTO names VALUES ('Douglas', 1, 'http://A')");
         s.execute("INSERT INTO names VALUES ('Dan', 2, 'http://B')");
         s.execute("INSERT INTO names VALUES ('Neal', 3, 'http://C')");

         System.out.println("Authors inserted.");

         s.close();
         conn.close();
      } catch (Exception err) {
         err.printStackTrace();
      }
   }
}
```

Lab (Select Data)

```
import java.sql.*;
public class SelectData {
    public static void main(String[] args) {
        try {
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase");

            Statement s = conn.createStatement();
            ResultSet rs = s.executeQuery("SELECT author, author_id, url FROM names");

            while (rs.next()){
                String author = rs.getString("author");
                int id = rs.getInt("author_id");
                String url = rs.getString("url");
                System.out.println(author + "," + id + ","+url);
            }
           rs.close();
           s.close();
           conn.close();
        } catch (Exception err) {   err.printStackTrace();    }
    }
}
```

Lab (Update Query)

```
import java.sql.*;

public class UpdateData {
    public static void main(String[] args) {
        try {
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase");

            Statement s = conn.createStatement();
            s.execute("UPDATE names SET url = 'http://mynewurl' WHERE author_id = 1");
            System.out.println("Data is updated");

            s.close();
            conn.close();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Lab (Delete Data)

```
import java.sql.*;
public class DeleteData {
    public static void main(String[] args){
        try {
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase");

            Statement s = conn.createStatement();
            s.execute("DELETE FROM names WHERE author_id=1");
            System.out.println("Authors with author_id=1 are deleted.");

            s.close();
            conn.close();
        } catch (Exception err) {
            err.printStackTrace();
        }
    }
}
```

Java Server Page (JSP)

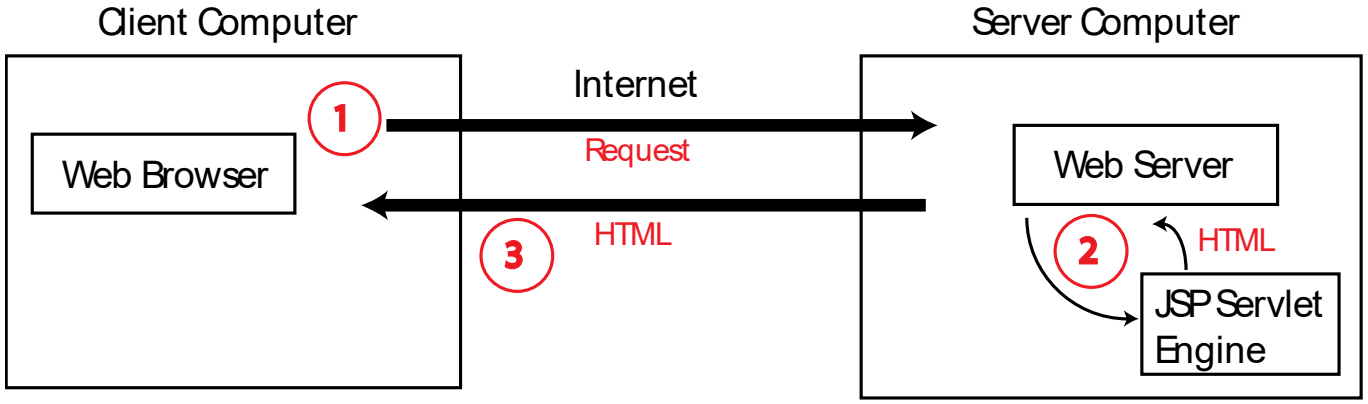
Shin-Jie Lee (李信杰)
Assistant Professor
Computer and Network Center
Department of Computer Science and Information Engineering
National Cheng Kung University

Web Programming with Java Server Pages

- Many technologies exist that allow programs to run within a web browser when visiting a website

- JSP
  - Dynamically compiles to Servlets and integrated with the server

Running a Java Server Page (JSP) Program



1. The client's web browser sends a request to the server for a web page that contains JSP code.

2. The JSP Servlet engine dynamically compiles the JSP source code into a Java servlet if a current, compiled servlet doesn't exist. The servlet runs and outputs HTML that is returned to the web server.

3. The web server sends the servlet's HTML to the client's web browser to be displayed.

JSP Requirements

- Web server capable of running JSP servlets
- Here we use the Tomcat web server
  - Download Tomcat 9.0 (32-bit/64-bit Windows Service Installer) from http://tomcat.apache.org/
  - Install Tomcat on D:\
  - Start Tomcat

JSP Tags - Declarations

- Declarations
  - Use to define variables and methods
  - Variable declarations are compiled as instance variables for a class that corresponds to the JSP page
  - Syntax:

```
<%!
        Declarations
 %>
```

- Example

```
<%!
   private int count = 0;
   private void incrementCount()
   {
        count++;
   }
%>
```

JSP Tags - Expressions

- Expressions
  - Use to access variables defined in declarations
  - Syntax:

```
<%=
        Expression
 %>
```

- Example

```
The value of count is <b> <%= count %> </b>
```

JSP Tags - Scriptlet

- Scriptlet
  - Use to embed blocks of Java Code
  - Syntax:

```
<%
        Java Code
 %>
```

  - Use `out.println()` to output to the browser
- Example

```
<%
  incrementCount();
  out.println("The counter's value is " + count +
"<br />");
%>
```

Lab (counter.jsp)

1. Create a count.jsp file with the following code
2. Put the file in the D:\Tomcat 9.0\webapps\ROOT\lab
3. Open browser and connect to http://localhost/lab/count.jsp

```
<%!
  private int count = 0;
  private void     incrementCount(){ count++; }
%>
  Before invoking incrementCount(), count is <%= count %>
<br>
<%
  incrementCount();
  out.println("After invoking incrementCount(), count is " +
count );
%>
```

Lab (heading.jsp)

```
<html>
<title> Displaying Heading Tags with JSP </title> <body>
<%! private static final int LASTLEVEL = 6; %>
<p>
This page uses JSP to display Heading Tags from Level 1 to
Level <%= LASTLEVEL %>
</p>
<%
  int i;
  for (i = 1; i <= LASTLEVEL; i++){
    out.println("<H" + i + ">" + "This text is in Heading
Level " + i + "</H" + i + ">");
  }
%>
</body>
</html>
```

HTML Generated by JSP Example

```
<html>
<title>
Displaying Heading Tags with JSP
</title>
<body>
<p>
This page uses JSP to display Heading Tags from
Level 1 to Level 6
</p>
<H1>This text is in Heading Level 1</H1>
<H2>This text is in Heading Level 2</H2>
<H3>This text is in Heading Level 3</H3>
<H4>This text is in Heading Level 4</H4>
<H5>This text is in Heading Level 5</H5>
<H6>This text is in Heading Level 6</H6>
</body>
</html>
```

Browser View of JSP Page

This page uses JSP to display Heading Tags from Level 1 to Level 6

This text is in Heading Level 1

This text is in Heading Level 2

This text is in Heading Level 3

This text is in Heading Level 4

This text is in Heading Level 5

This text is in Heading Level 6

HTML Forms

- Syntax for HTML Form

```
<FORM ACTION="Path_To_CGI_Program" METHOD="GET or POST">
Form_Elements
</FORM>
```

- ACTION identifies the program to execute
  - In our case, a JSP program
- GET or POST identify how data is transmitted
  - GET sends data as the URL, POST over the socket

Some HTML Form Elements

- Input Textbox

```
<INPUT TYPE="TEXT" NAME="Textbox_Name" VALUE="Default_Text"
 SIZE="Length_In_Characters"
 MAXLENGTH="Maximum_Number_Of_Allowable_Characters">
```

- Submission Button

```
<INPUT TYPE="SUBMIT" NAME="Name" VALUE="Button_Text">
```

- Many others form elements exist
  - E.g. radio buttons, drop down list, etc.

Lab (bookstore.html)

```
<html> <body>
<h1>Change Author's URL</h1>
<p> Enter the ID of the author you would like to change along
with the new URL. </p>

<form ACTION = "edit.jsp" METHOD = POST>

  Author ID:
  <input TYPE = "TEXT" NAME = "AuthorID" VALUE = "">
  <br>

  New URL:
  <input TYPE = "TEXT" NAME = "URL" VALUE = "http://" >
  <br>

  <INPUT TYPE="SUBMIT" VALUE="Submit">

</form>

</body> </html>
```

Browser View of HTML Form Document

Change Author's URL

Enter the ID of the author you would like to change along with the new URL.

Author ID:

New URL: http://

Submit

Reading HTML Form Input

- The **`request.getParameter`** method takes a String parameter as input that identifies the name of an HTML form element and returns the value entered by the user for that element on the form.
  - For example, if there is a textbox named AuthorID then we can retrieve the value entered in that textbox with the scriptlet code:

```
String value = request.getParameter("AuthorID");
```

- If the user leaves the field blank then getParameter returns an empty string.

Lab (edit.jsp)

```
<html>
<body>
<h2>Edit URL</h2>
<%
  String url = request.getParameter("URL");
  String stringID = request.getParameter("AuthorID");
  int author_id = Integer.parseInt(stringID);

  out.println("The submitted author ID is: " + author_id);
  out.println("<br/>");
  out.println("The submitted URL is: " + url);
%>
</body>
</html>
```

JSP Tags - Directive

- Directives
  - Instruct the compiler how to process a JSP program. Examples include the definition of our own tags, including the source code of other files, and importing packages.
  - Syntax:

```
<%@
        Directives
%>
```

- Example

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

Lab (JSP + JDBC)

1. Copy C:\Program Files\Java\jdk1.8.0_25\db\lib\derby.jar to D:\Tomcat 9.0\lib
2. Create and run insertdb.jsp
3. Create and run selectdb.jsp
4. Create a new version of edit.jsp
5. Run bookstore.html again

insertdb.jsp

```
<%@
  page import="java.sql.*"
%>
<html>
<body>
<h2>Insert DB</h2>

<%

  try {
                Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
                Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase;create=true");

                Statement s = conn.createStatement();
                s.execute("CREATE TABLE names (author varchar(50), author_id int, url varchar(80))");

                out.println("Database is created with a table.");

                s = conn.createStatement();
                s.execute("INSERT INTO names VALUES ('Douglas', 1, 'http://www.douglasadams.com')");
                s.execute("INSERT INTO names VALUES ('Dan', 2, 'http://www.dansimmons.com')");
                s.execute("INSERT INTO names VALUES ('Neal', 3, 'http://www.nealstephenson.com')");
                out.println("Authors inserted.");

                s.close();
                conn.close();
   }catch(Exception e){
                out.print(e);
   }

%>
</body>
</html>
```

selectdb.jsp

```
<%@
  page import="java.sql.*"
%>
<html>
<body>
<h2>Select DB</h2>
<%
  try {
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase");

            Statement s = conn.createStatement();
            ResultSet rs = s.executeQuery("SELECT author, author_id, url FROM names");
            while (rs.next()){
                        String author = rs.getString("author");
                        int id = rs.getInt("author_id");
                        String url = rs.getString("url");
                        out.print(author + "," + id + ","+url);
                        out.print("<br>");
            }

            rs.close();
            s.close();
            conn.close();
   }catch(Exception e){
            out.print(e);
   }

%>
</body>
</html>
```

edit.jsp (Version 2)

```
<%@
  page import="java.sql.*"
%>
<html>
<body>
<h2>Edit URL</h2>

<%
  String url = request.getParameter("URL");
  String stringID = request.getParameter("AuthorID");
  int author_id = Integer.parseInt(stringID);

  try {
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:derby:BookDatabase");
            Statement s = conn.createStatement();
            s.execute("UPDATE names SET url = '"+ url +"' WHERE author_id = "+author_id);
            out.println("Data is updated");

            s.close();
            conn.close();
   }catch(Exception e){
            out.print(e);
  }

%>
</body>
</html>
```

Reference

- "Absolute Java". Walter Savitch and Kenrick Mock. Addison-Wesley; 5 edition. 2012