JDBC Demonstration Courseware Using Servlets and Java Server Pages

Suzanne W. Dietrich, Susan D. Urban and Ion Kyriakides
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406
{dietrich | s.urban}@asu.edu

Abstract

This paper describes the design and functionality of courseware developed to demonstrate the JDBC API. The courseware is used in an advanced undergraduate database course that emphasizes the use of Web access to database systems. The JDBC courseware is written using Java Servlets and Java Server Pages, allowing the user to view the metadata associated with a database, to view and browse the information in a database according to the database metadata, and to query and/or manipulate data using SQL statements. The advantage of the courseware is that it demonstrates the main functionality of the JDBC API in an application-independent manner. The courseware can access any ODBC-compliant database, emphasizing the generality of the JDBC API and helping students understand how JDBC can be used to query the metadata of the database as well as the database contents.

1 Introduction

The interaction of databases and the Web is becoming an important component of database education. At Arizona State University, we are covering this interaction in a new advanced database concepts course for undergraduates [10]. Assuming a course on relational database concepts as a prerequisite, our advanced undergraduate database course addresses the use of Java Database Connectivity (JDBC) [4, 6] and the Extensible Markup Language (XML) [11] in the context of relational, object-oriented, and object-relational database technology. As part of the course, students implement a semester-long group project that uses JDBC to access a relational database with Web programming technology such as Java Servlets [7] and/or Java Server Pages (JSP) [8].

One of the challenges of using new technology in the classroom is that the technology is always changing. Based on our past work with the use of technology in a database curriculum [9], our educational approach is to teach only the basic concepts of the technology in class, providing demonstration software that uses the latest version of the technology. Through the use of cooperative group exercises and programming assignments, students study the demonstration software and actively acquire knowledge of the latest technology using a hands-on approach.

Our work in [2] describes the way in which we have used demonstration software in the advanced undergraduate database course to actively teach the use of XML for the import and export of data from an object-oriented database system. This paper describes the design and functionality of courseware developed to demonstrate the JDBC Application Programming Interface (API). JDBC is a mechanism that allows Java to communicate with databases using a standard API to access databases regardless of the driver and the database product. JDBC consists of a set of classes and interfaces written in Java, thus making its use platform independent. The JDBC API is primarily used to connect to databases and manipulate data through the use of method calls and SQL statements.

The JDBC example application presented in this paper is written using Java Servlets and JSP technologies. The example makes use of the JDBC API to connect to any database that is registered as an Open Database Connectivity (ODBC) data source [5]. The application can be run using a Web browser such as Netscape 4.0 or Internet Explorer 5.0 or higher. The functionality of the database allows the user to view the metadata associated with the database schema, to view and browse the information in the database according to the database metadata, and to query and/or manipulate the data using SQL statements. The JDBC example therefore demonstrates the main functionality of the JDBC API in an application-independent manner. The fact that the JDBC example software can be run on any ODBC-compliant database emphasizes to students the generality of the JDBC API, helping students understand how JDBC can be used to query the metadata of the database as well as the database contents. We encourage students to run the example on different databases and to study the code of the example to learn how to use the JDBC API.

The remainder of this paper describes the JDBC demonstration courseware and the manner in which it is implemented using Java Servlets and JSP. Section 2 provides an overview of the software architectural design. Section 3 provides a brief overview of JDBC. The user interface is covered briefly in Section 4. Section 5 provides an overview of the JDBC interface for database metadata. Section 6 covers the database manipulation commands. Section 7 concludes the paper by summarizing the contributions of the JDBC demonstration courseware.

2 Design

Figure 1 illustrates a high-level design of the JDBC demonstration courseware. The application is implemented using a three-tier architectural model. The top tier is the browser where the client can send Requests and receive Responses from the server using JSP pages. A Java Servlet and a Java Bean act as the middle tier, communicating with the underlying data source. The third tier uses the JDBC API through a JDBC-ODBC driver.

Servlets are small pieces of code, written in Java, that add functionality to a web server using the Request/Response model. The client sends a Request and the Server replies with a Response. JSP technology uses HTML-like tags written in Java to encapsulate the presentation of the web page. JSP pages are usually used in combination with Java Beans, which are Java objects with get and set methods. The Java Server Pages are compiled into Servlets during run time.

The application design was also influenced by the Model-View-Controller (MVC) [3] design pattern. In the MVC, three types of components are used to decouple the functionality of an application: the Model, the View, and the Controller. The Model exposes the business logic, which is the JavaBean class in this application that stores and manipulates the data. The View renders the model, which corresponds to the JSP pages, where the presentation of the data stored in the model is rendered using HTML. The Controller defines the behavior of the application, which is represented by the Java Servlet, performing different actions depending on the client Request.

This architectural design allows the application access to any registered, ODBC-compliant data source. Ideally, the student has the ability to run the application on a class server. The courseware documentation also provides directions on how the student can register the data source and install the application on the local host (see http://www.eas.asu.edu/~cse494db).

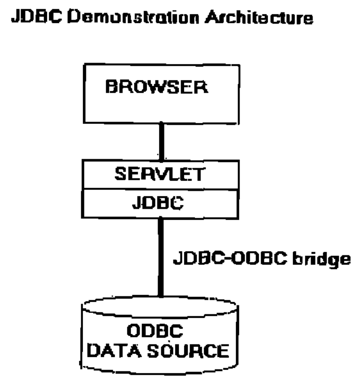


Figure 1 JDBC Courseware Architecture

3 JDBC Overview

The JDBC API, as given in the java.sql package [6], consists of a collection of interfaces and classes, allowing Java programs to interact with a database. The interfaces that are illustrated in the courseware include: Connection, DatabaseMetaData, Statement, ResultSet, and ResultSetMetaData. A Connection represents the interconnection between the Java client program and the database. The DatabaseMetaData provides information about the data contained in the database, such as table names and the names and types of attributes. The Statement interface provides methods for executing queries, and is created in the context of a Connection. The result of a Statement or metadata retrieval is returned as a ResultSet, with methods for iterating through the set of results. The ResultSetMetaData provides information about the ResultSet such as attribute names and types.

The students are provided with an overview of these basic JDBC concepts in class. The JDBC courseware is then demonstrated to the students, illustrating the use of the courseware on several different databases. The students are then given a cooperative assignment to study the code to find out how the JDBC API is being used.

The remainder of the paper describes examples of how the courseware demonstrates the various interfaces of the JDBC API. Due to the generality of the courseware, it does not demonstrate the PreparedStatement interface for executing parameterized, pre-compiled queries or the CallableStatement interface for executing stored procedures.

4 User Interface

The generic nature of the application required an appropriate user interface in the browser. The initial page prompts the user to enter a registered ODBC data source name. After the connection to the database has been established, two browser windows are opened: a Database MetaData window and a Database Manipulation window.

These windows are shown in the appendix using the Employee Training database [1].

The Database MetaData window allows the user to browse the database metadata in the top frame, and to examine the database instance in the bottom frame. The top frame displays the table names and their attributes from the currently opened database. When a table name is selected in the top frame, the bottom frame displays the selected table's instance, showing the attribute names and types in the header row followed by the tuples in the table.

The Database Manipulation window allows the user to query the database and to manipulate the data stored in the database including insertions, updates, and deletions. The submit button executes the SQL statement entered in the text window, displaying the result of the statement execution in the bottom frame. The Clear button clears the text window. The Open MetaData button opens the Database MetaData window if the one opened by default was inadvertently closed. The Close Database button closes the current database, returning the application to its main page.

The following sections describe the JDBC operations used to establish the database metadata and manipulation windows.

5 Database Metadata

The Database MetaData window uses the JDBC API metadata functions to display the metadata that corresponds to the database on which the connection is established. Figure 2 shows code fragments illustrating some of the methods in the DatabaseMetaData and ResultSet interfaces. The getMetaData() method on the Connection object (con) returns an object of type DatabaseMetaData. The getTables() method returns the tables associated with the database metadata into the rsTables ResultSet object. The code then iterates through the result set and selects user-defined tables and their associated names (and types), storing the metadata information into a JavaBean for later display by the JSP page that generates the top frame of the Database MetaData window.

As a result of running the example on different databases and studying the code associated with the MetaData window, students actively learn about the concept of metadata. Using the JDBC metadata API helps students to understand that metadata is an important part of the database that can be queried and used in a manner similar to the way that they query the data in the database.

```
 DatabaseMetaData dbmd;
 ResultSet rsTables = null;
 ResultSet rsColumns = null;
 dbmd = con.getMetaData();
 rsTables = dbmd.getTables(null, null, null, null);
while (rsTables.next())
{
   tableName = rsTables.getString("TABLE_NAME");
   ttype=rsTables.getString("TABLE_TYPE");
   /* checks if it's a user-defined table*/
   if (ttype.equals("TABLE"))
   { /* store the table names for later display */
      ...
    /* Get the column names in the ResultSet */
     rsColumns=dbmd.getColumns(null,null,tableName,null);
     while (rsColumns.next())
       { /* store the attribute names for later display
           in the top frame of the metadata window */ ...
       }
  ...}
}
/* transfer control to the appropriate JSP page */
```

Figure 2 Code Fragment Illustrating MetaData

6 Database Manipulation

The Database Manipulation window uses the JDBC API to manipulate the database, allowing the user to query, insert, update and delete data in the table by issuing ad hoc SQL statements. Figure 3 shows a code fragment for part of the database manipulation. A statement is created in the context of the database connection and the string query is initialized with the contents of the text box in the database manipulation window.

Figure 3 illustrates the case where the string in the text box begins with the SELECT identifier. The application calls the executeQuery method on the Statement object to return the result of the query in a ResultSet object. The metadata interface for a ResultSet object is used to store the column names and types in the JavaBean associated with the query. Then the code iterates through the result set instance, populating the JavaBean. Control is then passed to the appropriate Java Server Page to display the information stored in the Query Bean in the bottom frame of the Database Manipulation window.

If the text entered in the text box begins with the INSERT, UPDATE, or DELETE identifier, then the application calls the executeUpdate method on the Statement object. The bottom frame displays the result of the execution. Any

```
Statement stmt = con.createStatement();
ResultSet rs = null;
String query = req.getParameter("query");
...
rs = stmt.executeQuery(query);
ResultSetMetaData rsmd = rs.getMetaData();
int numCols = rsmd.getColumnCount ();
qBean.setNumColumns(numCols);
for ( int i = 1; i <= numCols; i++)
{ /* Store the column names and their types in the
   QueryBean using rsmd.getColumnLabel(i) and
   rsmd.getColumnTypeName(i)*/        ... }
while(rs.next())
{ /* Iterate through the ResultSet and store the actual
   data in the QueryBean  using rs.getString(i) */
}
...
    /* transfer control to the appropriate JSP page */
```

Figure 3 Code Fragment Illustrating Queries

other type of statement will not be processed and the application will respond with an error message.

As with the metadata API, when students study the code associated with the Database Manipulation window, they begin to understand how the JDBC API supports the capability for application programs to communicate queries to a database and manipulate the results that are returned from such queries. Students also learn to appreciate the ability to query the metadata of the database since they can use the MetaData window to review the database schema as they formulate queries to be submitted to the database.

7 Summary

This paper has presented an overview of courseware that demonstrates the JDBC API using Java Servlets and Java Server Pages. The students are provided with a concrete example that illustrates the use of the JDBC API in action, including the important use of metadata for both the database and the result set. Besides demonstrating this technology, the courseware also provides the students with a tool to issue ad hoc SQL statements to any registered ODBC data source. The courseware is available for educational use as part of the dissemination of the curriculum materials developed for a grant sponsored by the National Science Foundation to develop a second database course for undergraduates (see http://www.eas.asu.edu/~cse494db).

Acknowledgements

This work was partially supported by the National Science Foundation's DUE CCLI-EMD program (DUE-9980417). We thank Mathangi Ranganathan for her work on an earlier version of this courseware.

References

[1] Dietrich, S. W., *Understanding Relational Database Query Languages*, Prentice Hall, 2001.

[2] Dietrich, S. W., Suceava, D., Cherukuri, C. and Urban, S. D. "A Reusable Graphical User Interface for Manipulating Object-Oriented Databases using Java and XML", *Proceedings of ACM SIGCSE 2001*, Charlotte, North Carolina, February 21-26, 2001, 362-366.

[3] Gamma, E., Helm, R., Johnson, R., and Vlissides, V., *Design Patterns Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.

[4] Hamilton, G., Cattell, R., and Fisher, M. *JDBC™ Database Access with Java™ A Tutorial and Annotated Reference*, Addison Wesley, 1997.

[5] Microsoft Corporation, Open Database Connectivity, http://www.microsoft.com/data/odbc/default.htm

[6] Sun Microsystems, Inc., JDBC™ Data Access API, http://java.sun.com/products/jdbc/ (for java.sql package summary see http://java.sun.com/j2se/1.4/docs/api/java/sql/package-summary.html)

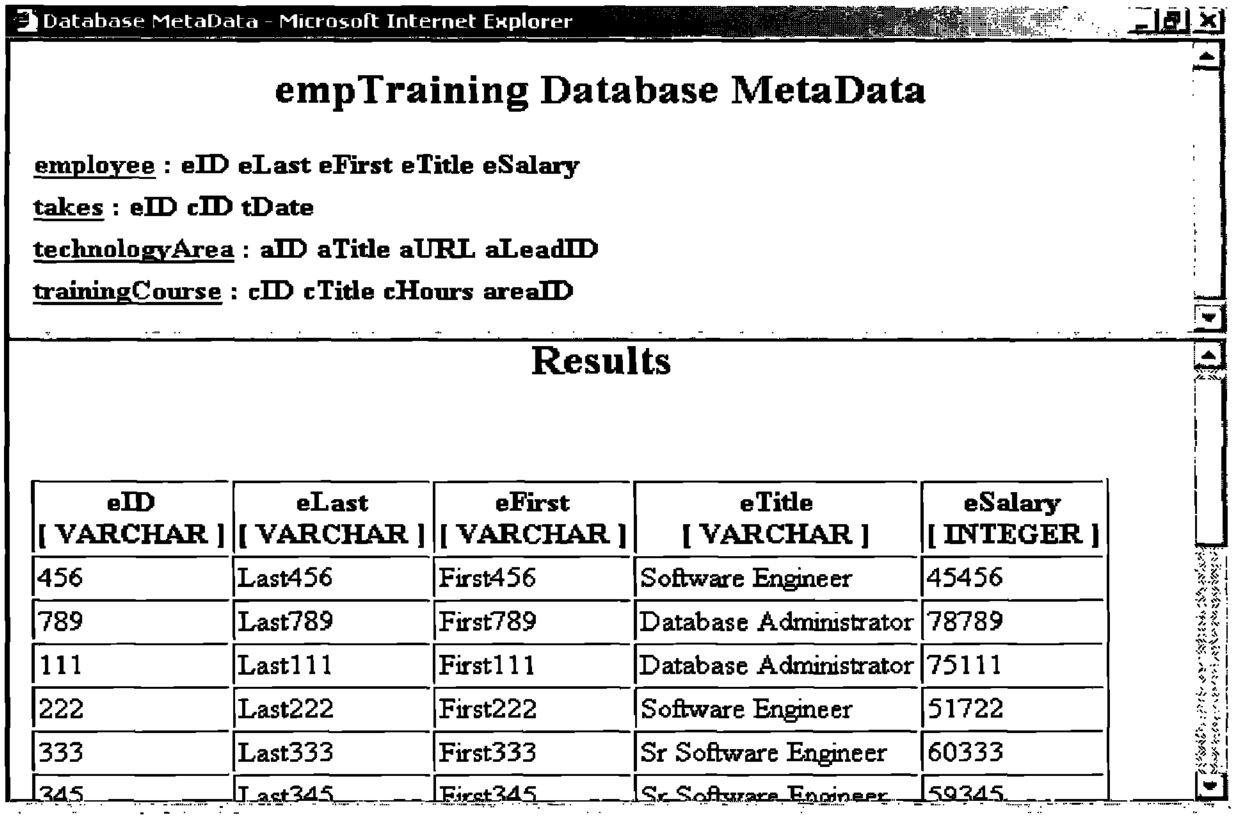
[7] Sun Microsystems, Inc., Java™ Servlet API Specification, Version 2.2, http://java.sun.com/products/servlet/2.2/

[8] Sun Microsystems, Inc., Java Server Pages™ Specification 1.2, http://java.sun.com/products/jsp/

[9] Urban, S. D. and Dietrich, S. W., Integrating the Practical Use of a Database Product into a Theoretical Curriculum, *Proceedings of the 28th ACM SIGCSE Technical Symposium on Computer Science Education*, San Jose, California, February 27 - March 1, 1997, pp. 121-125

[10] Urban, S. D. and Dietrich, S. W., "Advanced Database Concepts for Undergraduates: Experience with Teaching a Second Course", *Proceedings of ACM SIGCSE 2001*, Charlotte, North Carolina, February 21-26, 2001, 357-361.

[11] World Wide Web Consortium (W3C)'s XML web page, 2001. http://www.w3.org/XML/

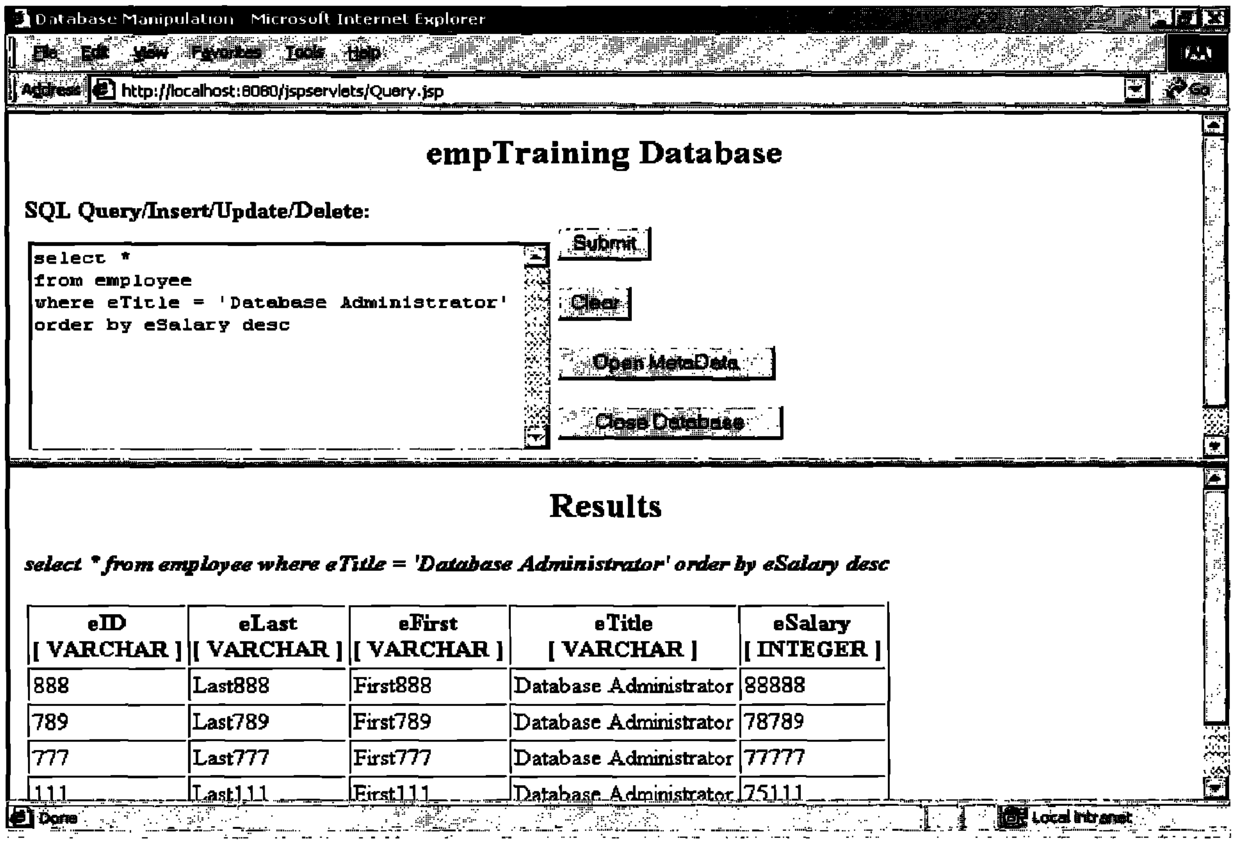


empTraining Database MetaData

employee : eID eLast eFirst eTitle eSalary
takes : eID cID tDate
technologyArea : aID aTitle aURL aLeadID
trainingCourse : cID cTitle cHours areaID

Results

eID [VARCHAR]	eLast [VARCHAR]	eFirst [VARCHAR]	eTitle [VARCHAR]	eSalary [INTEGER]
456	Last456	First456	Software Engineer	45456
789	Last789	First789	Database Administrator	78789
111	Last111	First111	Database Administrator	75111
222	Last222	First222	Software Engineer	51722
333	Last333	First333	Sr Software Engineer	60333
345	Last345	First345	Sr Software Engineer	59345


Database Manipulation Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/jspservlets/Query.jsp
Submit
Clear
Open MetaData
Close Database
Done
Local intranet

empTraining Database

SQL Query/Insert/Update/Delete:

```
select *
from employee
where eTitle = 'Database Administrator'
order by eSalary desc
```

Results

select * from employee where eTitle = 'Database Administrator' order by eSalary desc

eID [VARCHAR]	eLast [VARCHAR]	eFirst [VARCHAR]	eTitle [VARCHAR]	eSalary [INTEGER]
888	Last888	First888	Database Administrator	88888
789	Last789	First789	Database Administrator	78789
777	Last777	First777	Database Administrator	77777
111	Last111	First111	Database Administrator	75111