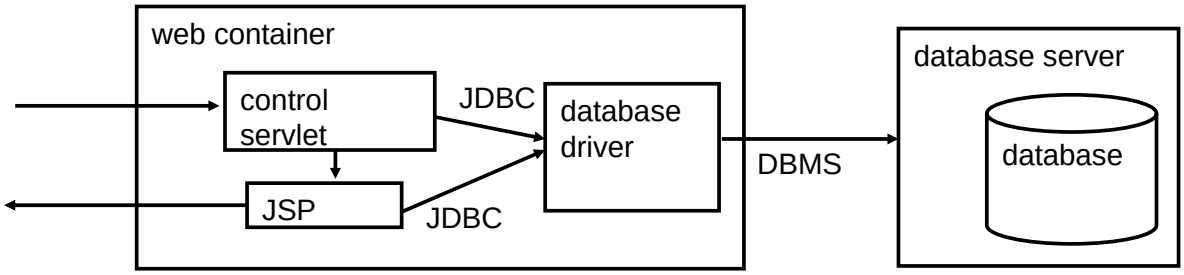
Server-side Web Programming

Lecture 13:
JDBC Database Programming

JDBC Definition

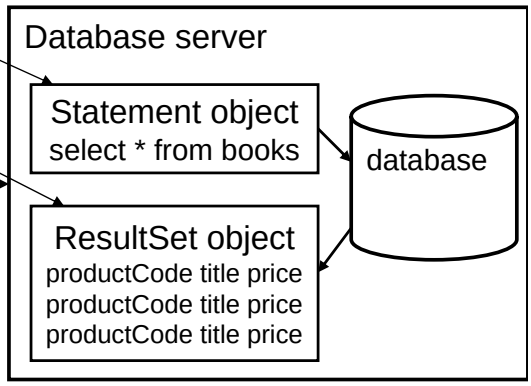
- Java Database Connectivity (JDBC): set of classes that provide methods to
  - **Connect** to a database through a database server (using a driver)
  - **Query** database using **SQL syntax**, getting "list" of records that match query
  - **Manipulate** database by executing SQL commands to modify, insert, and delete records



JDBC Components

- Major objects involved:
  - **`Connection`**: represents connection to a database through a server
  - **`Statement`**: represents SQL statement executed on database via that connection
  - **`ResultSet`**: represents "list" of records matching a query

```
<%@ page import="java.sql.*"  %>
  <%
    Connection connection = null;
    Statement statement = null;
    ResultSet books = null;
```



Connecting to the Database Server

- Load the database driver
  - Not necessary in most recent version, but safe thing to do

  Syntax:

  `Class.forName("driver class").newInstance();`

- Name of driver class based on url of provider
  Example: **`com.mysql.jdbc.Driver`**

```
15      try {
          Class.forName("com.mysql.jdbc.Driver").newInstance();  // load the driver
17        connection = DriverManager.getConnection(
18                      "jdbc:mysql://localhost/TestDB", "john", "sesame");
19        }
```

Connecting to the Database Server

- Connect to the database
  - Need to provide username and password
- Need to provide url of database
  Usual form: **jdbc:*server type*:*url of server*/*database name***

  Example: **jdbc:mysql://localhost/TestDB**
- Syntax:
  connectionobject **=**
  DriverManager.getConnection("*databaseURL*",
  "*username*", "*password*");

```
connection = DriverManager.getConnection(
                "jdbc:mysql://localhost/TestDB", "john", "sesame");
}
```

Exception Handling in JDBC

- Any database-related statement may throw an **SQLException**
  - Your code must put in try/catch block
  - May also need to catch other exceptions
    - **ClassNotFoundException** for missing database driver

```
try {
  Class.forName("com.mysql.jdbc.Driver").newInstance();  // load the driver
  connection = DriverManager.getConnection(
                "jdbc:mysql://localhost/TestDB", "john", "sesame");
  }
catch (ClassNotFoundException e) { %> NO DRIVER <% }
catch (SQLException e) { %> NO CONNECTION <% }
```

Diagnostic message displayed

Better idea: Redirect to an error page

Executing Queries

- Create new statement object using the connection
- Execute an SQL query using that statement
- Store results in a ResultSet object

- Syntax:
  `statement = connection.createStatement();`
  `statement.executeQuery(“SQL query”);`

```
try {
  statement = connection.createStatement();
  books = statement.executeQuery("SELECT * FROM books");
  }
catch (SQLException e) { %> BAD QUERY <% }
```

Reading ResultSets

- Can only do simple access:
  - Read in field values from current record
  - Move to next record

- Syntax to move to next record: **`ResultSetObject.next();`**
  - Returns false if no next record, true otherwise
  - Must execute once before reading first record
  - Usually while loop to read until no more records

```
while(ResultSetObject.next()) {
code to read in current record
}
```

```
while (books.next()) {
  String productCode = books.getString("productCode");
  String title = books.getString("title");
```

Reading ResultSets

- Syntax to read field from current record:

```
value = ResultSetObject.getType(fieldname);
```

Specify type data is to be read in as

varChar → **getString**

int → **getInt**

double → **getDouble**

Specify field name used in database

Reading ResultSets

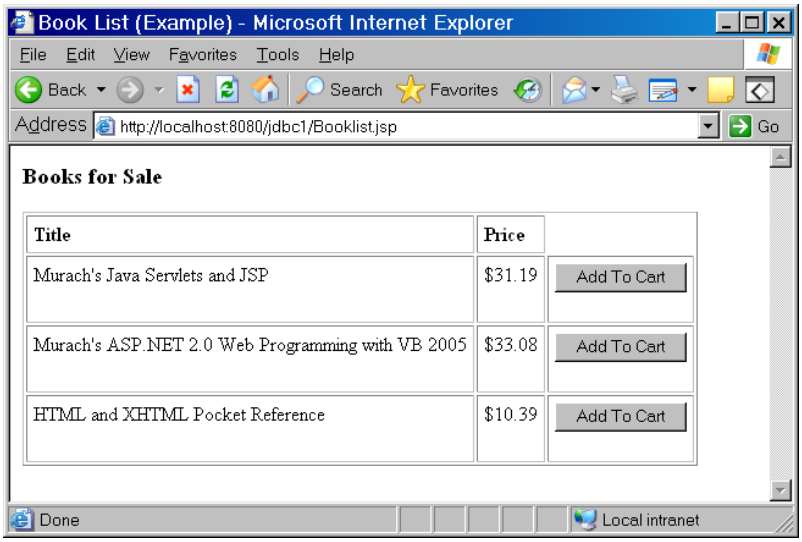
```
mysql> select * from books;
+-------------+-----------------------------------------------------+-------+
| productCode | title                                               | price |
+-------------+-----------------------------------------------------+-------+
| 0001        | Murach's Java Servlets and JSP                      | 31.19 |
| 0002        | Murach's ASP.NET 2.0 Web Programming with VB 2005    | 33.08 |
| 0003        | HTML and XHTML Pocket Reference                     | 10.39 |
+-------------+-----------------------------------------------------+-------+
3 rows in set (0.00 sec)

mysql>
```

```
<%
  while (books.next()) {
    String productCode = books.getString("productCode");
    String title = books.getString("title");
    double price = books.getDouble("price");
%>
```

Reading ResultSets

- Once ResultSet read in, can use in own code
  - Display in JSP
  - Store in array for future use, etc.



Reading ResultSets

```
40  <%
41    while (books != null && books.next()) {
42      String productCode = books.getString("productCode");
43      String title = books.getString("title");
44      double price = books.getDouble("price");
45  %>
46    <tr valign="top">
47      <td><%= title %></td>
48      <td>$<%= price %></td>
49      <td>
50        <form action ='<%= response.encodeURL("/jdbc1/servlet/CartServlet?&quantity=1") %>'
51              method = "get">
52          <input type="hidden" name="productCode" value="<%= productCode %>">
53          <input type="submit" value="Add To Cart">
54        </form>
55      </td>
56    </tr>
57
58  <%
59    }
60  %>
```

Display title and price in next table row

Create form that passes productCode of selected book to servlet if button on this row is pressed

Executing Update Statements

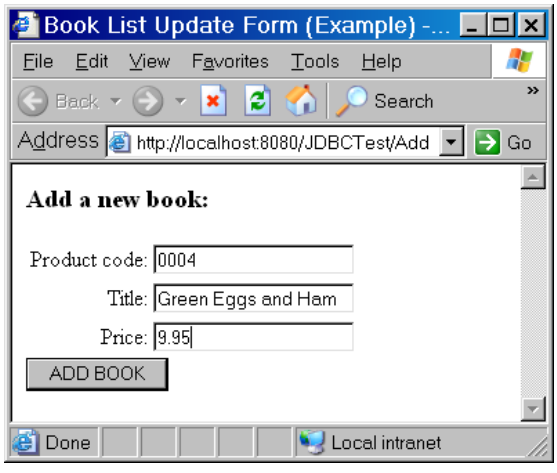
- Syntax:
  `Statement statement = connection.createStatement();`
  `statement.executeUpdate("`*`SQL statement`*`");`
- Example:

```
statement.executeUpdate("INSERT INTO books
        (productCode, title, price)
        VALUES ('0004', 'Green Eggs and Ham', 9.95)");
```

```
mysql> select * from books;
+-------------+-----------------------------------------------+-------+
| productCode | title                                         | price |
+-------------+-----------------------------------------------+-------+
| 0001        | Murach's Java Servlets and JSP                | 31.19 |
| 0002        | Murach's ASP.NET 2.0 Web Programming with VB 2005 | 33.08 |
| 0003        | HTML and XHTML Pocket Reference               | 10.39 |
| 0004        | Green Eggs and Ham                            |  9.95 |
+-------------+-----------------------------------------------+-------+
4 rows in set (0.00 sec)
```

Inserting Parameter Values

- User often decides how database is updated
  - Enters parameter on form
  - Parameters used for update




```
<h3>Add a new book:</h3>
<form action ='/jdbc2/servlet/AddServlet' metho
<table>
  <tr>
    <td align="right">Product code:</td>
    <td><input type="text" name="productCode">
  </tr>
  <tr>
    <td align="right">Title:</td>
    <td><input type="text" name="title">
  </tr>
  <tr>
    <td align="right">Price:</td>
    <td><input type="text" name="price">
  </tr>
</table>
<input type="submit" value="ADD BOOK">
</form>
```

Inserting Parameter Values

- Read in parameter values
  - Validate if necessary

```
protected void processRequest(HttpServletRequest request, HttpServl
throws ServletException, IOException {
// Get parameters to create new record
String productCode = request.getParameter("productCode");
String title = request.getParameter("title");
double price = Double.parseDouble(request.getParameter("price"));
```

Inserting Parameter Values

- Insert into SQL statement
  - Will need to use **+** to append values into SQL statement string
  - Note that string values must be inside **‘ ’** to avoid syntax errors

```
// Create query to put new record into database
try {
  statement = connection.createStatement();
  statement.executeUpdate("INSERT INTO books (productCode, title, price) " +
                          "VALUES ('"+productCode+"', '"+title+"', "+price+")");
```

0004 — Green Eggs and Ham — 9.95

Creates string of form:

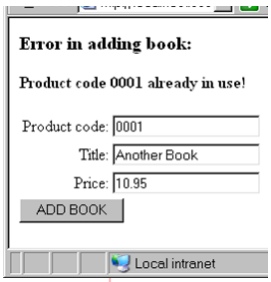
INSERT INTO books (productCode, title, price) VALUES (‘0004’, ‘Green Eggs and Ham’, 9.95)

Validation and Updates

- Usually need to validate update with database to avoid problems
  - Don't add item if already in database
  - Don't update or remove if not in database
    - Won't cause database error, but probably want to inform user
- Checking whether item in database involves query
  - Create query fro item in question
  - If no results then not in database
  - Key idea: use statement of form **`if(`*`ResultSetObject`*`.next())`**
    - True if at least one result
    - False if no matches

Validation and Updates

- Example:
  Validating that no book with given productCode exists in database before adding it
  - Query for book with that productCode
  - Redirect to error page if **`books.next()`** is true




```
// First check whether the given product code already exists
// by creating a query for it.
try {
  statement = connection.createStatement();
  books = statement.executeQuery("SELECT * FROM books WHERE productCode = '"+productCode+"'");
  if (books.next()) {  // At least one record with this product code
    // Redirect to the error JSP
    RequestDispatcher dispatcher =
            getServletContext().getRequestDispatcher("/AddError.jsp");
    dispatcher.forward(request, response);
    return;
    }
```