E-Commerce Database and Servlet Introduction

Jim Wilenius
Database Basics

- **What is a Database?**
  - Stores information (as one or more files)
  - Organized logically in Tables
  - Columns and Rows

<table>
<thead>
<tr>
<th>Table1</th>
<th>Table2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col1, ..., ColN</td>
<td>Col1, ..., ColN</td>
</tr>
<tr>
<td>row1</td>
<td>row1</td>
</tr>
<tr>
<td></td>
<td>rowM</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rowM</td>
</tr>
</tbody>
</table>
Database Basics

- Actions
  - Create Tables
  - Ask Questions (query) with criteria
  - Change information

```
SELECT ...
CREATE TABLE ...
UPDATE ...

DB
```
Database Basics

- Example table:
  - Person \{name, age, pNo\}

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arne</td>
<td>young</td>
<td>123456</td>
</tr>
<tr>
<td>Jim</td>
<td>infant</td>
<td>234567</td>
</tr>
<tr>
<td>Kim</td>
<td>infant</td>
<td>345678</td>
</tr>
</tbody>
</table>

- Query:
  
  SELECT name, pNo FROM Person WHERE age = 'infant'

returns the record set:
  
  Jim 234567
  Kim 345678
Database Basics

- SQL – structured query language
  - ANSI Standard for accessing Databases
  - DML – Data Manipulation Language
    - SELECT
    - UPDATE
    - DELETE
    - INSERT INTO
  - DDL – Data Definition Language
    - CREATE TABLE
    - DROP TABLE
    - ALTER TABLE
    - CREATE INDEX
    - DROP INDEX
Database Basics

- You now know the basic ideas...
  - What a DB is
  - How data is organized
  - That you can manipulate or query data

- How do you actually USE it in a practical application?
Database and Java

JAVA – JDBC (java database connectivity)

- Load the database driver
- Open a connection to the database
- Use connection to create/execute statements
- Get a result-set from the db (when applicable).

![Diagram of Java program connecting to a database](image-url)
Database and Java

- Load driver
  - Class.forName("org.hsqldb.jdbcDriver");
  - Throws ClassNotFoundException

- Create Connection – to myDB
  - con = DriverManager.getConnection("jdbc:hsqldb:file:myDB", "sa", "");
  - Throws SQLException
Database and Java

- Use the connection `con` to create a table
  ```java
  Statement stmt = con.createStatement();
  String e = “CREATE TABLE tblTest ( aField VARCHAR)”
  stmt.executeUpdate(e);
  stmt.close();
  ```

  - This will create the initial db file with one table
  - Information can be added to the database
Database and Java

- Use the connection `con` to add information

  ```java
  Statement stmt = con.createStatement();
  String e = "INSERT INTO tblTest (aField) VALUES ('test')";
  stmt.executeUpdate(e);
  stmt.close();
  ```

- Do it again: with 'hej'

- The table will have 2 rows

<table>
<thead>
<tr>
<th>tblTest</th>
<th>aField</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>test</td>
</tr>
<tr>
<td></td>
<td>hej</td>
</tr>
</tbody>
</table>
Database and Java

- Use the connection `con` to add information

```java
PreparedStatement p = con.prepareStatement("INSERT INTO tblTable ( aField ) VALUES( ? )");
p.setString(1, "saippuakauppias");
p.executeUpdate();
p.close();
```

- The table, from before, will now have 3 rows

<table>
<thead>
<tr>
<th>tblTest</th>
<th>aField</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td></td>
</tr>
<tr>
<td>hej</td>
<td></td>
</tr>
<tr>
<td>saippuakauppias</td>
<td></td>
</tr>
</tbody>
</table>

- Practical when setting binary data

```java
p.setBytes(N, myByteArray);
```
Database and Java

- Use the connection `con` to get information

  ```java
  Statement s = con.createStatement(
      "SELECT * FROM tblTable" );
  ResultSet r = s.executeQuery();
  ... use r ...
  s.close();
  ```

- The ResultSet will contain the three rows.
- `r.next()` - go to next row
- `r.getXXXX()` - methods to access columns
- Read the javadoc it is very helpful!
Database and Java

- Close everything nicely
  - Execute a statement with “SHUTDOWN”
  - Close the `con` object `con.close();`
Database and Java

- You now have the basic knowledge of
  - How to connect to a DB in java
  - How to create a statement and execute it
  - How to close the DB connection

- Use the provided links and google
  - More details about SQL syntax
  - Other types of statements (javadoc)

- Field Types
  - VARCHAR
  - INTEGER
  - VARBINARY
  - ... (see HSQLDB documentation, Data Types)
Web Servers

- The software that serves the web pages you see when you browse the Internet
- Many clients (web browsers)
- One server (web server)
  - Responds to clients requests for HTML pages
Web Servers

- **Static Server**
  - Serves only static HTML pages (somefile.html)
  - Not very powerful / useful

- **Tomcat (and others)**
  - Static HTML
  - Dynamic pages
    - Created when the request is made from the browser
    - JSP / Servlets
Web Server – dynamic page

- JSP
  - Dynamic contents in HTML file
  - Files are name .jsp instead of .html
    - Example: example.jsp

  ```html
  <HTML>
  <BODY>
  The time is <%= new java.util.Date() %>
  </BODY>
  </HTML>
  ```

  - Evaluated each time the page is requested
Web Server – Servlets

- Servlets – java programs
  - Called by the Tomcat server
Web Server – Servlets

- Setting up a Servlet in tomcat
  - See the lab instructions for directory structure

- `web.xml` – how Tomcat knows what to do

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>
  <servlet>
    <servlet-name>aTestServlet</servlet-name>
    <servlet-class>MyServletClass</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>aTestServlet</servlet-name>
    <url-pattern>/myservlet</url-pattern>
  </servlet-mapping>
</web-app>
```
What is a Servlet really...

- Normal java program (a class)
- No `public static void main(...)` required
- `default constructor` MUST exist
- Extends the `HTTPRequestServlet` class
- Override methods to customize Servlet
  - `doGet(...)`
  - `doPost(...)`
  - `Init(...)`
  - `destroy(...)`
Servlets - access

Ways of accessing the Servlet

• An URL http://server.com/myservlet
• With parameters – append ?paramName=value&...
Servlets - customizing

- **Override methods**
  - `doGet(...)`: called on each request of the page
  - `doPost(...)`: which one, depends on the request
  - `Init(...)`: called once when Servlet is loaded
  - `destroy(...)`: called once when Servlet is stopped

- `doGet(...)` and `doPost(...)`
  - `doGet(...)` is the default
  - 2 parameters
  - `HttpServletRequest`
  - `HttpServletResponse`
Servlets

- **HttpServletRequest object**
  - Contains request information
  - Form/fixed parameters
  - Use the `.getParameter("paramName")` when using forms or fixed parameters.
  - Has many useful methods, read the api documentation for useful information.
**Servlets**

- `HttpServletResponse res`
  - `res.getWriter()` returns a PrintWriter
    - Use to output your resulting HTML text.
    - Example:
      ```java
      PrintWriter out = res.getWriter();
      out.println("<html><body> static servlet page </body></html>");
      ```
  - Contains all HTML headers and error codes
    see the servlet-api documentation
Servlets

- init() and init(ServletConfig s)
  - Override this method to do Load-time initialization. (eg. create connection to DB)
  - Initialize state variables
  - ServletConfig has parameter values from the web.xml for a specific servlet.
Servlets

- Example: `init(ServletConfig c)` in the `web.xml` add to the `<servlet>` tag

```xml
...  
<servlet>
    <servlet-name>aTestServlet</servlet-name>
    <servlet-class>MyServletClass</servlet-class>
    <init-param>
        <param-name>testParam</param-name>
        <param-value>theValue</param-value>
    </init-param>
</servlet>
...  
```

- Access from `ServletConfig c`
  - `c.getParameter("testParam")`
Servlets

- `destroy()`
  - Called when the servlet is stopped.
  - (close the DB gracefully)