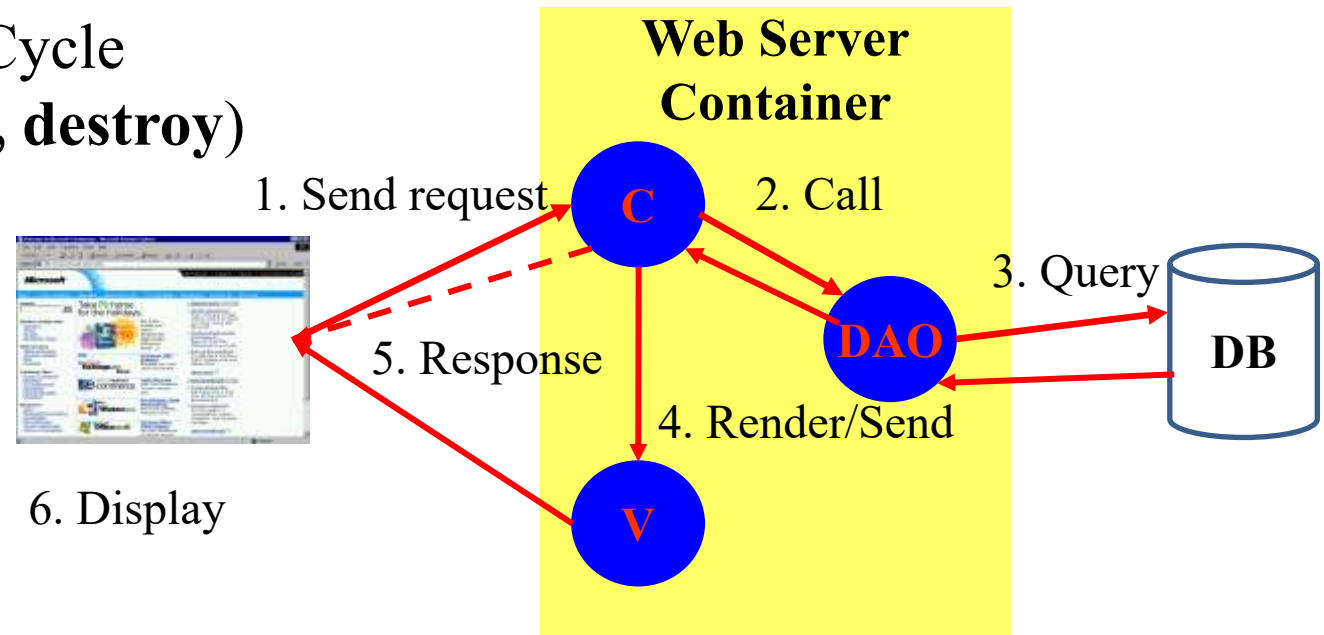


Web Application Interacting with Database

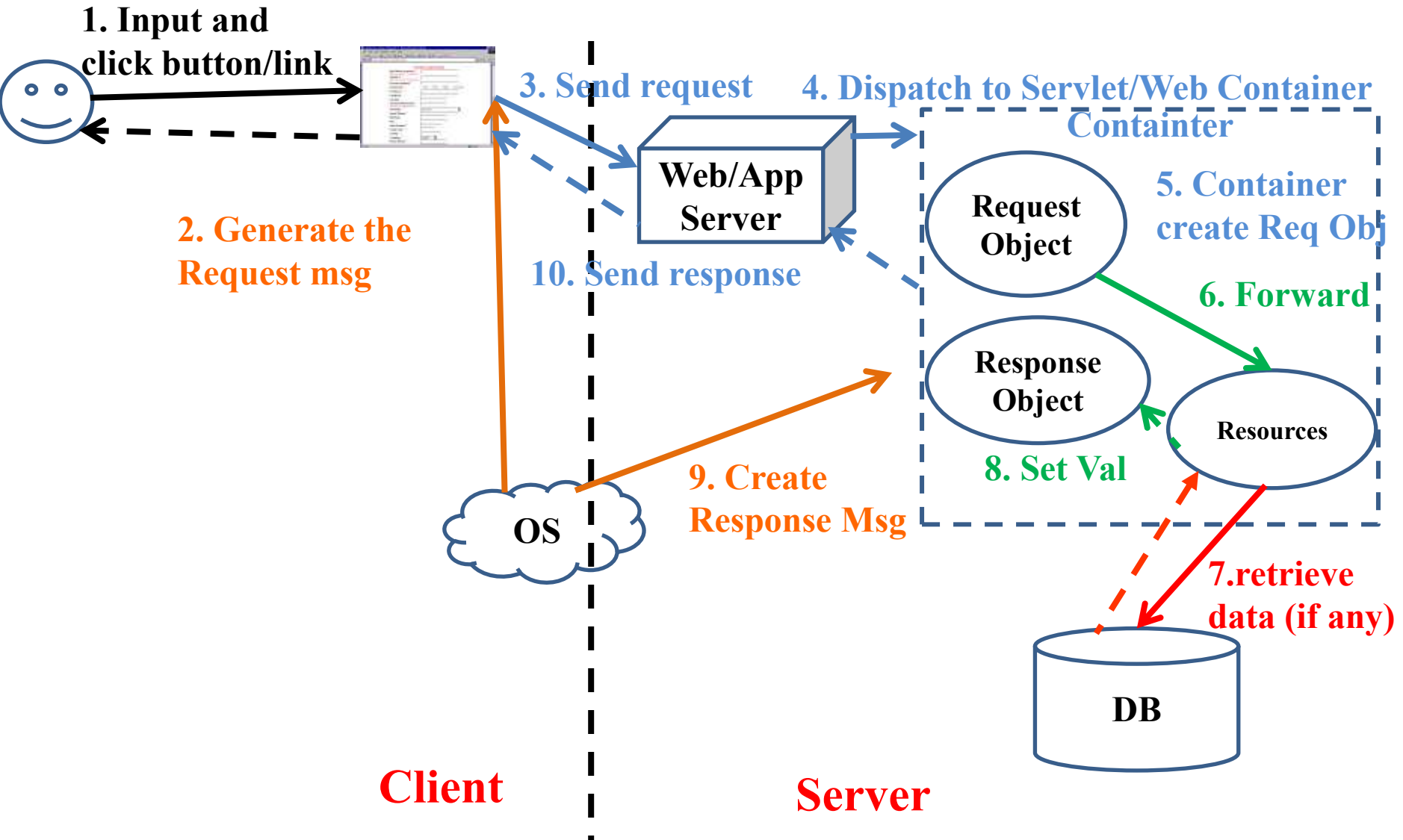
JDBC

Review

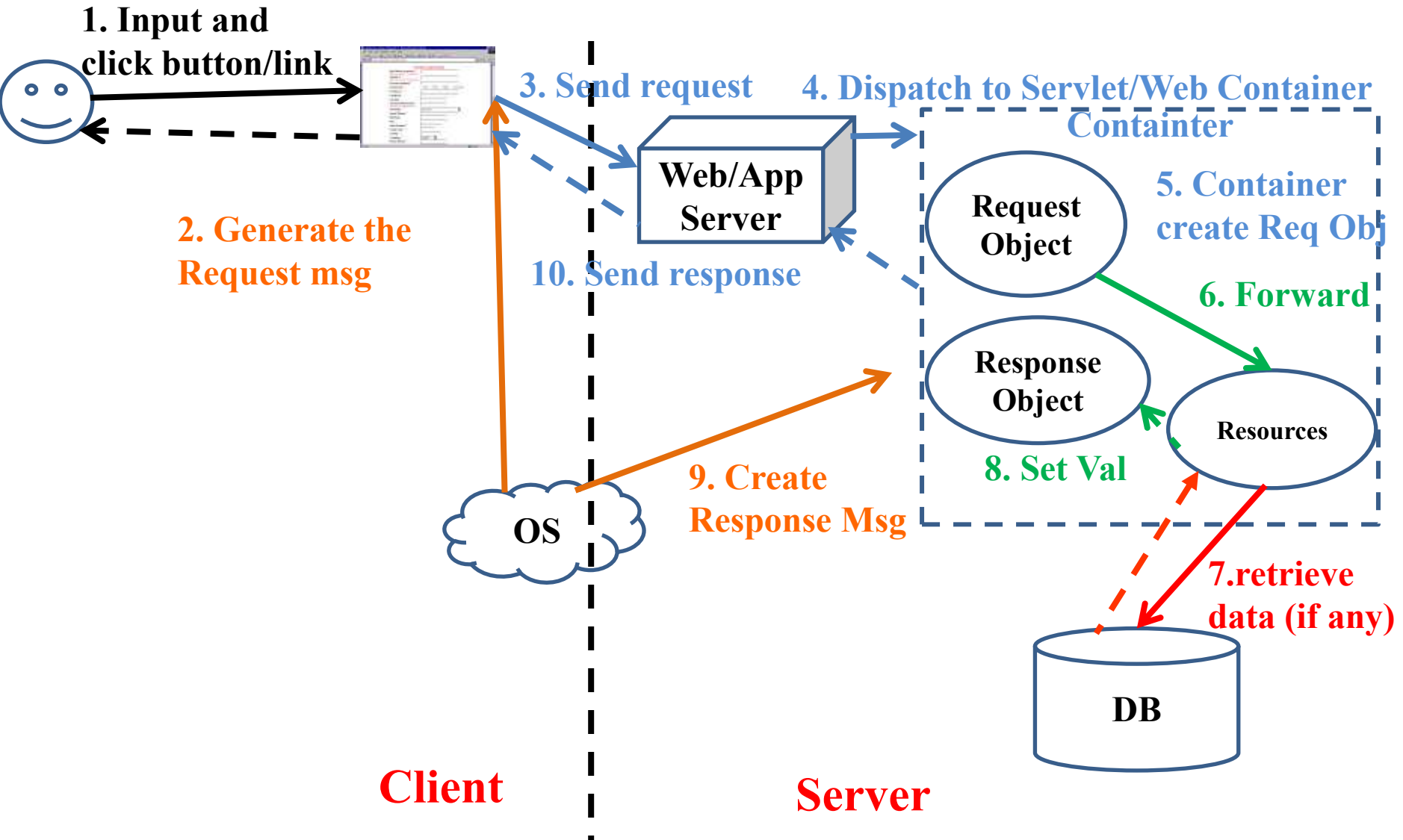
- **How to build the simple web site using html and servlet?**
 - Break down structure component in building web application
- **Some concepts**
 - Servlet vs. Java class, Parameter vs. Variable
 - Form Parameters
 - Http Protocol
 - HTTP Methods: **GET, POST, ...**
 - Servlet Life Cycle
(init, service, destroy)



Review



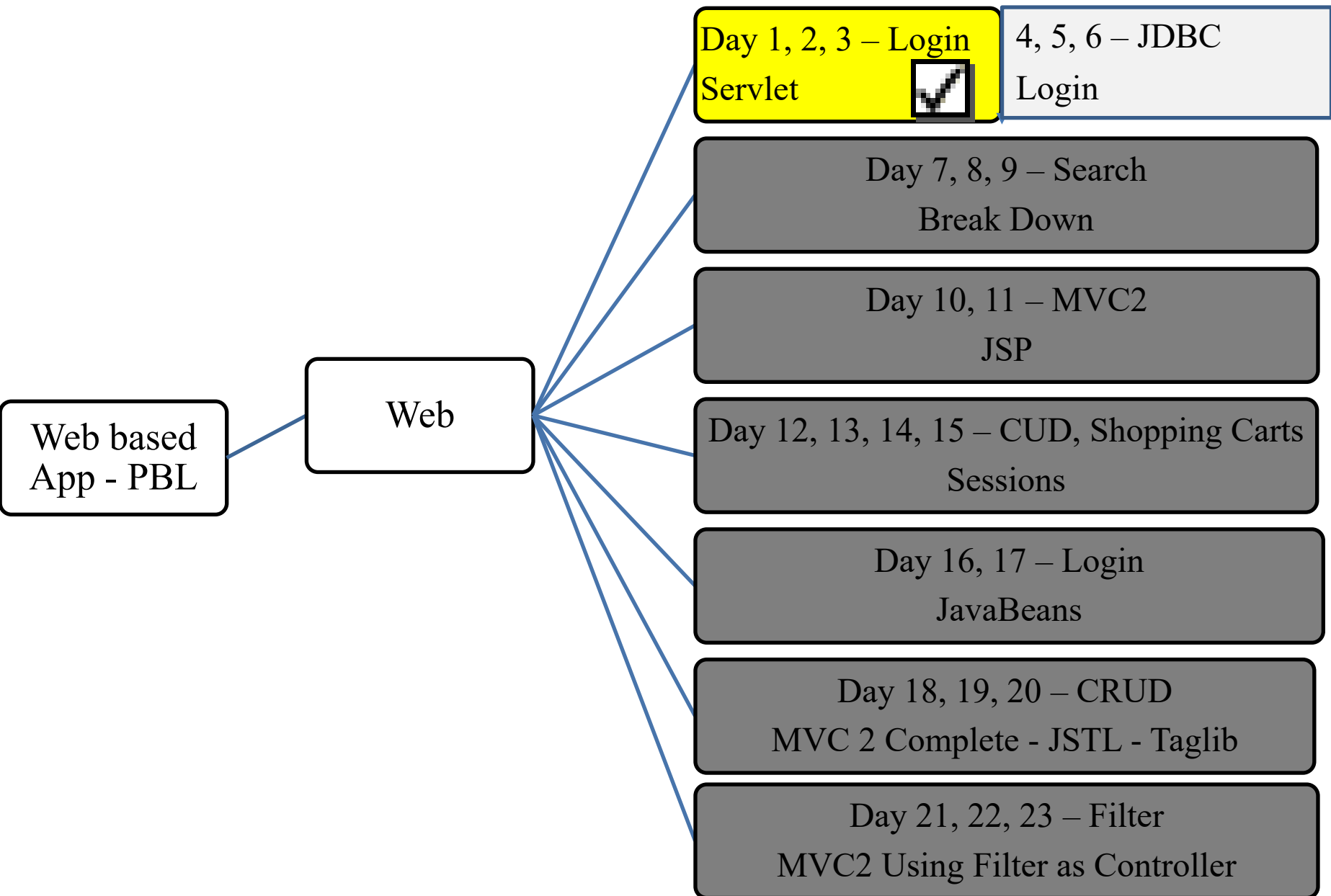
Review



Objectives

- **How to access database from web application?**
 - JDBC
 - Relational Database Overview
 - JDBC and JDBC Drivers
 - JDBC Basics: Processing SQL Statements
 - Implement CRUD application using MS SQL

Objectives



Overview

DB vs. DBMS

- **Databases**
 - Are **collection of related data** which are **stored** in **secondary** mass storage and are **used** by some processes **concurrently**.
 - Are **organized** in some ways in order to **reduce redundancies**.
- **DBMS: Database management system**
 - Is a **software** which manages some databases.
 - **Supports** ways to users/processes for creating, updating, manipulating on databases and security mechanisms are supported also.
 - **DBMS libraries** (C/C++ codes are usually used) support APIs for user programs to manipulate databases.

Overview

Relational DB

- Presents information in tables with **rows and columns**.
 - A table is referred to as a relation in the sense that it is a **collection of objects of the same type** (rows).
- A Relational Database Management System (**RDBMS**)
 - Handles the way data is **stored, maintained, and retrieved**.
 - **Ex:** MS Access, MS SQL Server, Oracle

KHANHKT\SQL2017.S...dbo.Registration			
	Column Name	Data Type	Allow Nulls
▶	username	varchar(20)	<input type="checkbox"/>
	password	varchar(30)	<input type="checkbox"/>
	lastname	nvarchar(100)	<input type="checkbox"/>
	isAdmin	bit	<input type="checkbox"/>

KHANHKT\SQL2017.S...dbo.Registration				
	username	password	lastname	isAdmin
▶	hutruc	123456	Hu Truc	False
	IA1301	123456	Class IA1301	False
	khanh	kieu123	Khanh Kieu	True
	khanh@Spring	123456	Spring annotati...	False
	khanhSpring	123456	khanh	False
	khanhSpringa	123456	dddscaaaa	False

RDBMS

Structure Query Language (SQL)

- Common DML – **Data Manipulating Language** queries.
 - **SELECT** columns **FROM** tables [**WHERE** condition]
 - **UPDATE** table **SET** column=value,... **WHERE** condition
 - **DELETE FROM** table **WHERE** condition
 - **INSERT INTO** table (col1, col2,...) **VALUES** (val1, val2,...)

JDBC

Overview

- Java Database Connectivity
 - Is an application programming interface (**API**) for the programming language **Java**, which defines how a **client** may **access a database**
 - The JDBC™ API
 - Was designed to keep simple things simple such as **executing** common **SQL** statements, and **performing** other **objectives** common to database applications
 - Is a Java API that can **access any kind** of tabular data, especially data stored in a **Relational Database**.

JDBC

API

- JDBC APIs has 02 parts in the **java.sql** package

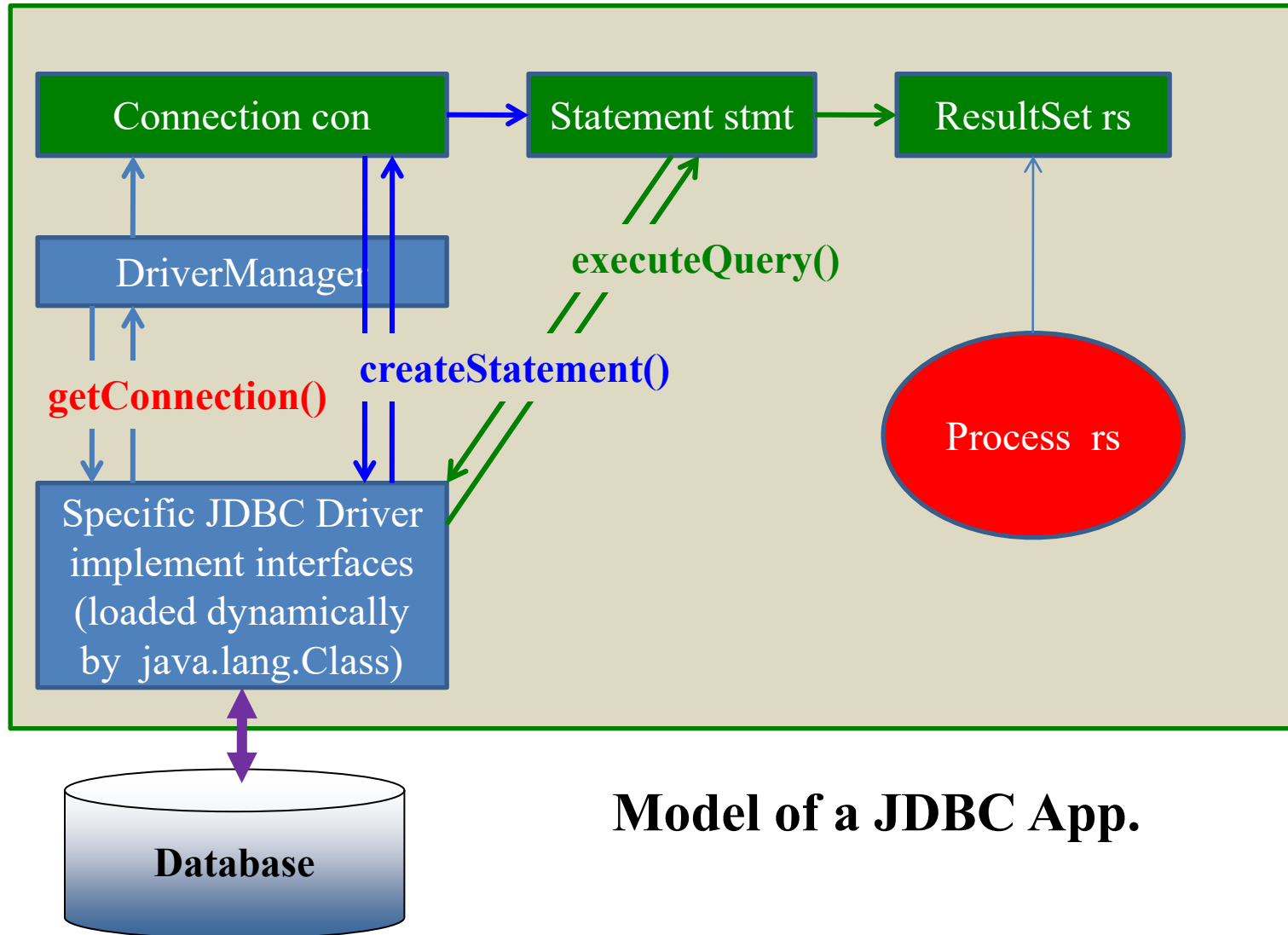
Part	Details	Purposes
JDBC Driver	DriverManager class	<p>Java.lang.Class.forName(DriverClass) will dynamically load the concrete driver class, provided by a specific provider for a specific database. This class implemented methods declared in JDBC interfaces.</p> <p>The class DriverManager will get a connection to database based on the specific driver class loaded.</p>
JDBC API	<u>Interfaces:</u> Connection, Statement ResultSet DatabaseMetadata ResultSetMetadata <u>Classes</u> SQLException	<p>For creating a connection to a DBMS</p> <p>For executing SQL statements</p> <p>For storing result data set and achieving columns</p> <p>For getting database metadata</p> <p>For getting resultset metadata</p>

Refer to the java.sql package for more details in Java documentation

JDBC

API

Java App.



Model of a JDBC App.

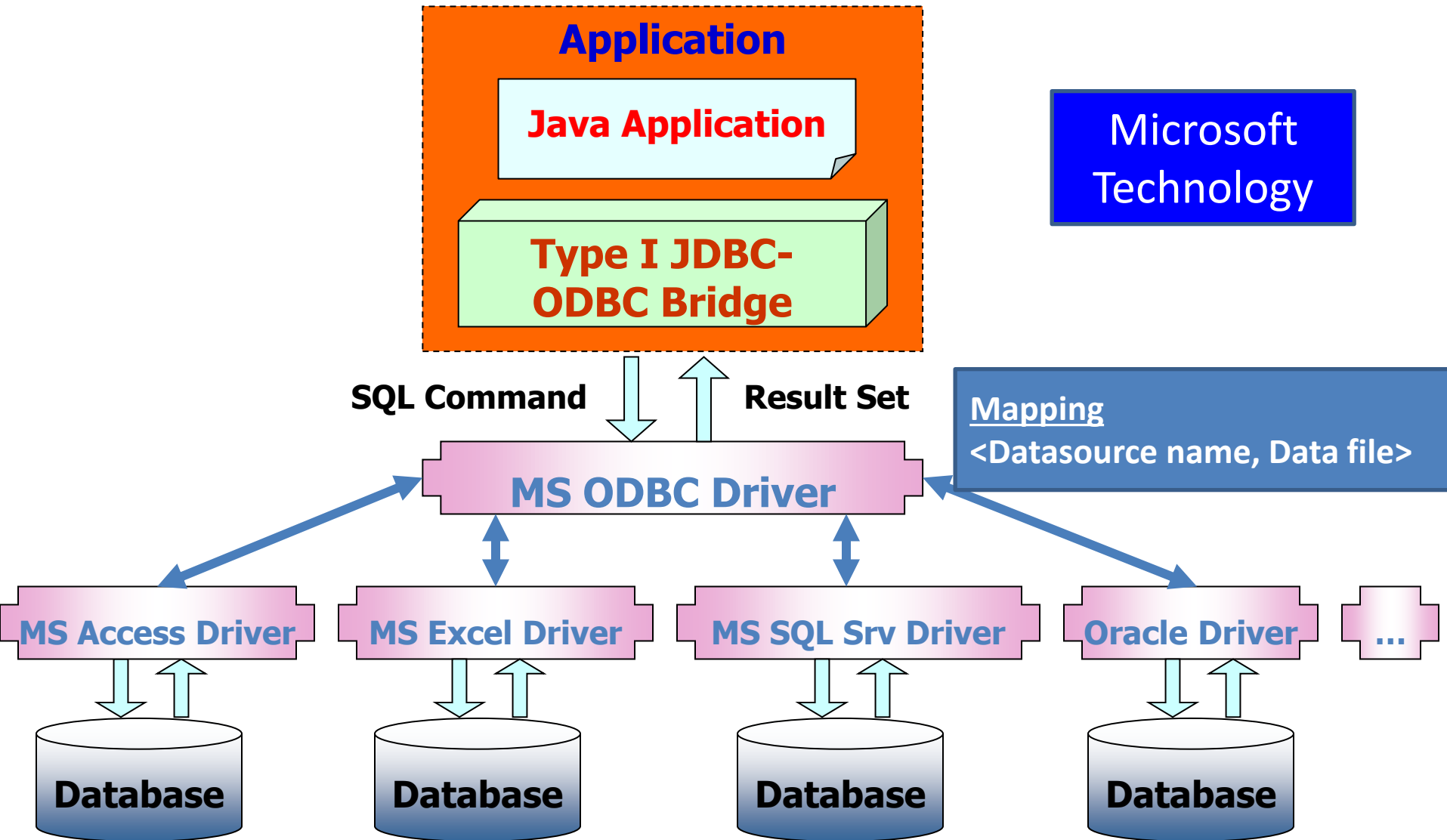
JDBC

API

- **DBMS provider/developer** will supply a package in which **specific classes** implementing **standard JDBC driver (free)**.
- Based on characteristics of DBMSs, **four types** of JDBC drivers are:
 - **Type 1: JDBC ODBC**
 - **Type 2: Native API**
 - **Type 3: Network Protocol**
 - **Type 4: Native Protocol**
- **Type 1 and Type 4 are populated.**

JDBC

Type 1-Driver: JDBC-ODBC Bridge



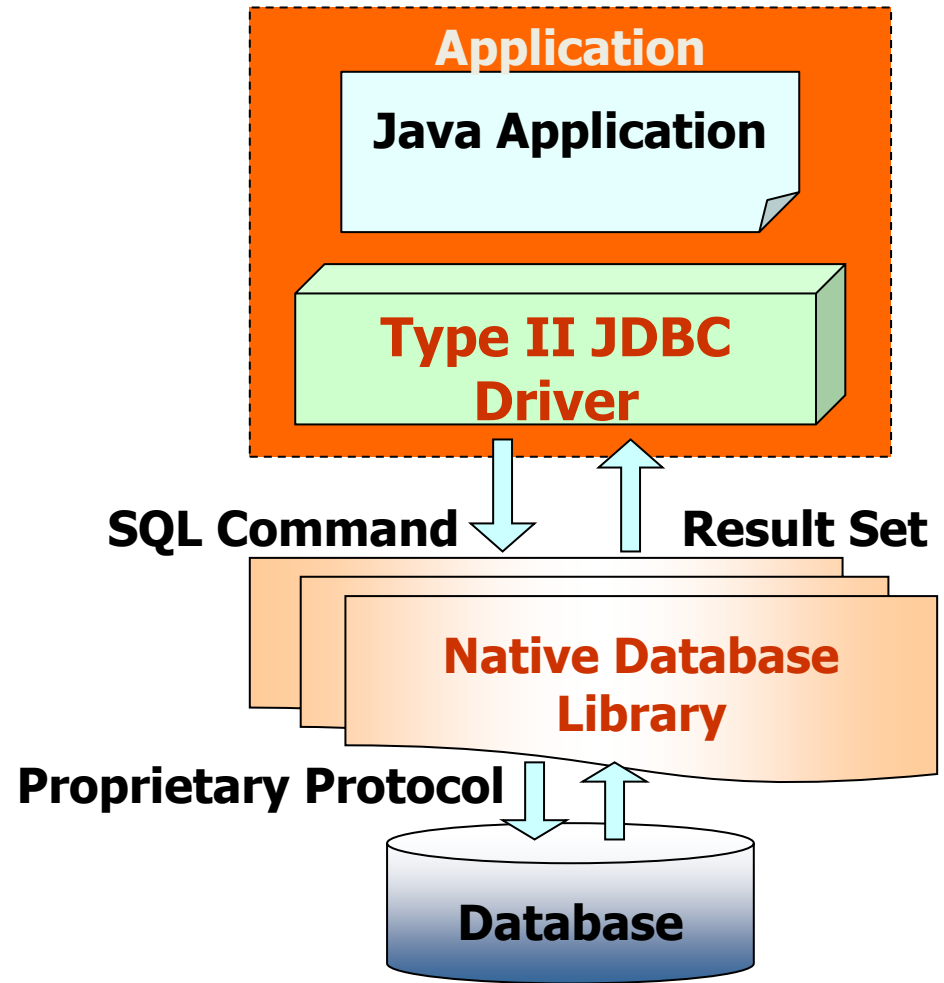
Type 1-Driver: JDBC-ODBC Bridge

- This package is in the JDK as default.
- **Translates** JDBC APIs to ODBC APIs
- **Enables** the Java applications to **interact** with any database supported by Microsoft.
- **Provides platform dependence**, as JDBC ODBC bridge driver uses ODBC
- **JDBC-ODBC bridge is useful when Java driver is not available for a database but it is supported by Microsoft.**
- **Disadvantages**
 - Platform **dependence** (Microsoft)
 - The performance is comparatively slower than other drivers
 - Require the ODBC driver and the client DB to be on the server.
- **Usage:** DSN is registered to use connecting DB (a data source is declared in Control Panel/ODBC Data sources)

JDBC

Type 2-Driver: Native API

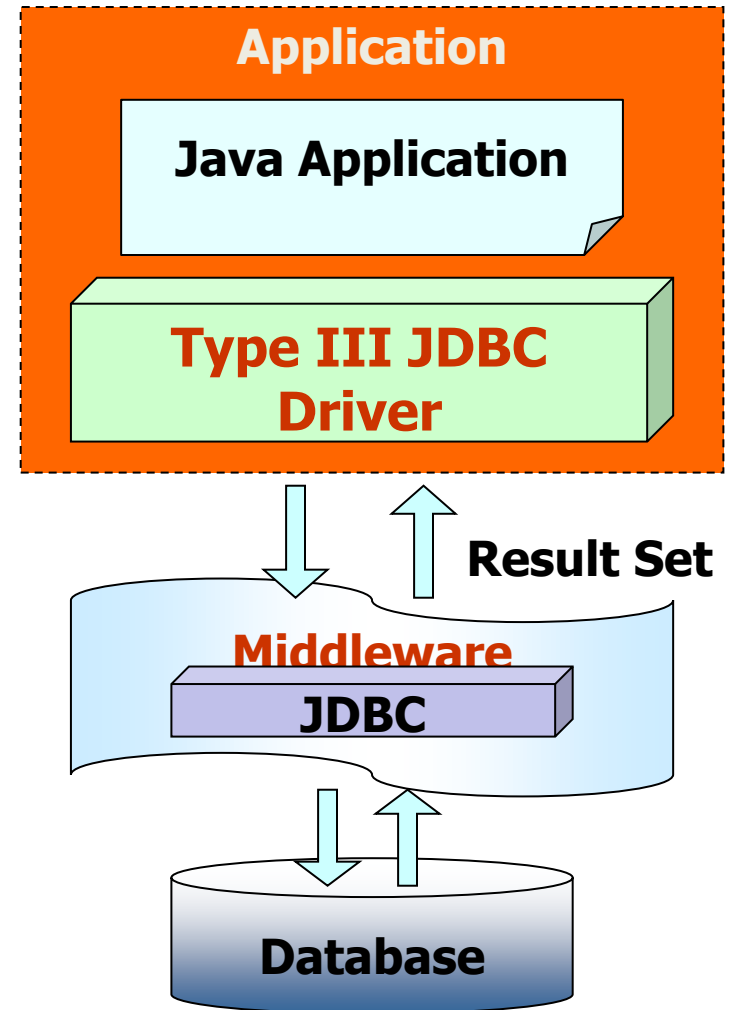
- Provides **access** to the database **through C/C++ codes**.
- Developed **using native code libraries**
- Native code libraries provide access to the database, and **improve the performance**
- Java application sends a request for database connectivity as a normal JDBC call to the **Native API driver**
- **Establishes** the call, and **translates** the call to the particular database protocol that is **forwarded** to the database



JDBC

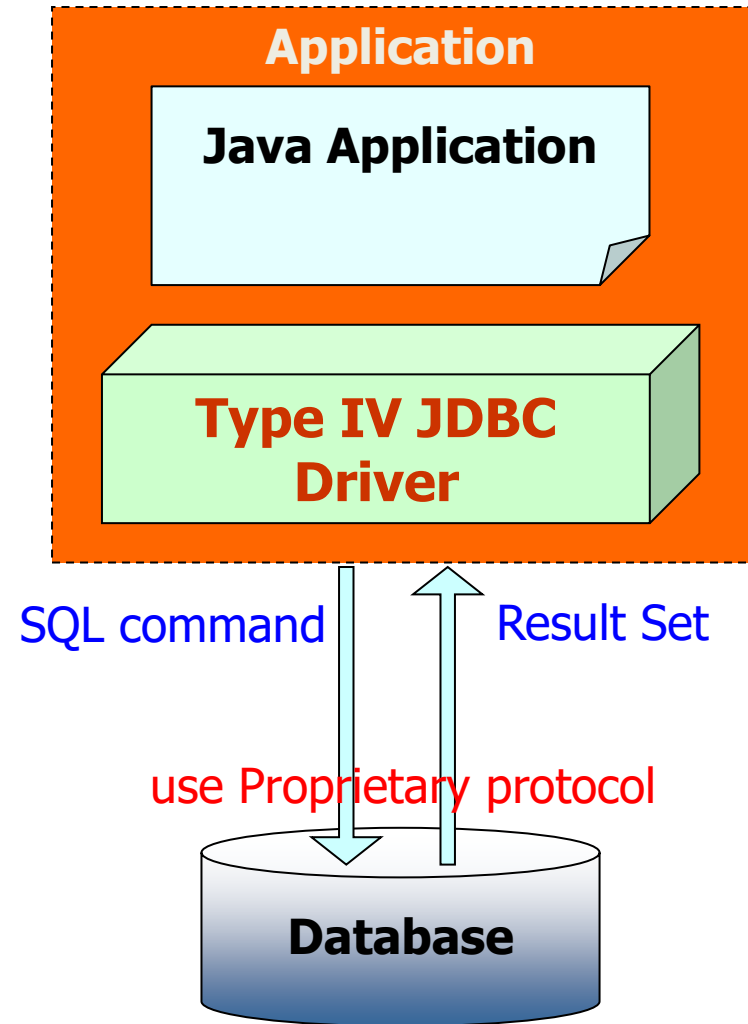
Type 3-Driver: Network Protocol

- Use a **pure Java client** and **communicate** with a **middleware server** using a **database-independent protocol**.
- The **middleware server** then **communicates** the **client's requests** to the **data source**
- **Manages multiple Java applications** connecting to **different databases**



Type 4-Driver: Native Protocol

- **Communicates directly** with the database **using Java sockets**
- **Improves the performance** as translation is not required
- Converts JDBC queries into **native calls** used by the particular RDBMS
- The **driver library** is required when it is **used and attached** with the **deployed application** (**sqlserver 2000**: mssqlserver.jar, msutil.jar, msbase.jar; **sqlserver 2005**: sqljdbc.jar; **jtds**: jtds.jar ...)
- **Independent platform**



Download Driver & Configure RDBMS

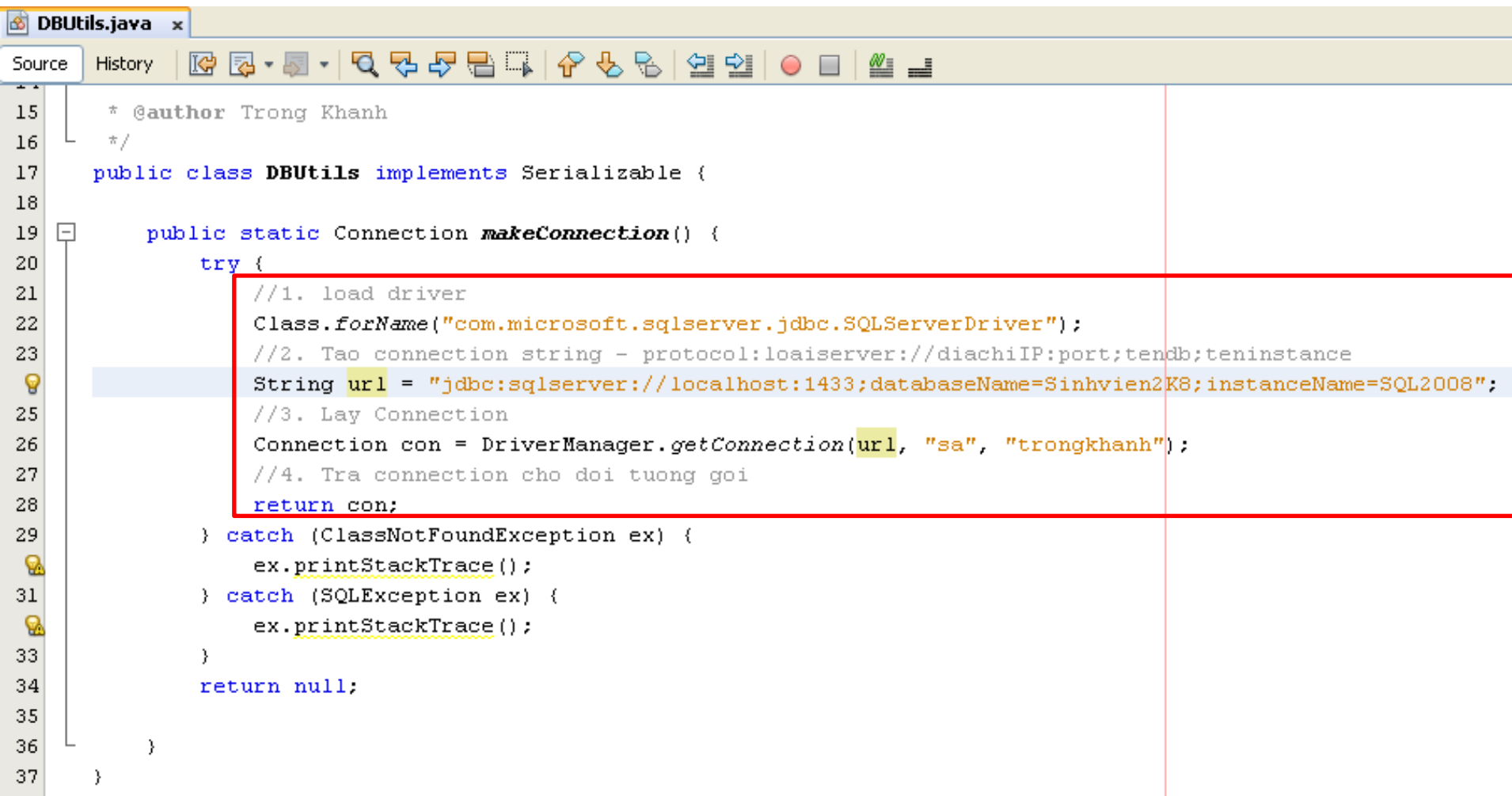
- **Download:**
 - [Download Microsoft JDBC Driver for SQL Server - SQL Server | Microsoft Docs](#)
- **Configuration**
 - Using SQLServer Config Manager
 - Or, services.msc
 - Configure ports, protocol

JDBC

- **How to connect DB using JDBC API**
 - Required
 - RDBMS: SQL Server
 - Driver Connection: sqljdbc4.jar
 - Steps
 - **Load Driver**
 - using **Class.forName** method
 - Driver string: **com.microsoft.sqlserver.jdbc.SQLServerDriver**
 - Exception: **ClassNotFoundException**
 - **Create connection String**
 - **protocol:server://ip:port;databaseName=DB[;instanceName=Instance]**
 - **Open connection**
 - **Connection con = DriverManager.getConnection(url, “user”, “pass”);**
 - Exception: **SQLException**

JDBC

- How to connect DB using JDBC API
 - Implementation



```

15  * @author Trong Khanh
16  */
17  public class DBUtils implements Serializable {
18
19      public static Connection makeConnection() {
20          try {
21              //1. load driver
22              Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
23              //2. Tao connection string - protocol:loaiserver://diachiIP:port;tendb;teninstance
24              String url = "jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;instanceName=SQL2008";
25              //3. Lay Connection
26              Connection con = DriverManager.getConnection(url, "sa", "trongkhanh");
27              //4. Tra connection cho doi tuong goi
28              return con;
29          } catch (ClassNotFoundException ex) {
30              ex.printStackTrace();
31          } catch (SQLException ex) {
32              ex.printStackTrace();
33          }
34          return null;
35      }
36  }
37  }
    
```

JDBC

- **How to access DB using JDBC API**
 - Required
 - DB is connected
 - Steps
 - Connect DB using method that you are built
 - **Check available DB connection**
 - Create **SQL String using DML**
 - Create Statement
 - Statement
 - **PreparedStatement (pass Parameter with ?)**
 - CallableStatement
 - **Execute Statement to get ResultSet**
 - **Process the ResultSet**
 - **Notes: must closed all objects that are created after process had finished**

JDBC

- How to access DB using JDBC API

- Implementation

```

26 Connection con = null;
27 PreparedStatement stm = null;
28 ResultSet rs = null;
29
30 try {
31     //1. connect DB
32     con = DBHelper.makeConnection();
33     //2. Create SQL String
34     if (con != null) {
35         String sql = "Select username "
36             + "From Registration "
37             + "Where username = ? And password = ?";
38         //3. Create Statement Object
39         stm = con.prepareStatement(sql);
40         stm.setString(1, username);
41         stm.setString(2, password);
42         //4. Execute Query
43         rs = stm.executeQuery();
44         //5. process result
45         if (rs.next()) {
46             return true;
47         } //end if
48     } //end if connection is opened
  
```

```

    } finally {
        if (rs != null) {
            rs.close();
        }
        if (stm != null) {
            stm.close();
        }
        if (con != null) {
            con.close();
        }
    }
  
```

JDBC

- **How to access DB using JDBC API**
 - Implementation

```

123         try {
124             //1. connect DB
125             con = DBHelper.makeConnection();
126             //2. Create SQL String
127             if (con != null) {
128                 String sql = "Delete From Registration "
129                     + "Where username = ?";
130                 //3. Create Statement Object
131                 stm = con.prepareStatement(sql);
132                 stm.setString(1, username);
133                 //4. Execute Query
134                 int row = stm.executeUpdate();
135                 //5. process result
136                 if (row > 0) {
137                     return true;
138                 }
139             } //end if connection is opened
140         } finally {
  
```


JDBC

- How to access DB using JDBC API
 - Implementation

```

158 //1. connect DB
159 con = DBHelper.makeConnection();
160 //2. Create SQL String
161 if (con != null) {
162     String sql = "Insert Into "
163         + "Registration(username, password, lastname, isAdmin) "
164         + "Values(?, ?, ?, ?)";
165 //3. Create Statement Object
166 stm = con.prepareStatement(sql);
167 stm.setString(1, username);
168 stm.setString(2, password);
169 stm.setString(3, fullName);
170 stm.setBoolean(4, role);
171
172 //4. Execute Query
173 int row = stm.executeUpdate();
174 //5. process result
175 if (row > 0) {
176     return true;
177 }
178 } //end if connection is opened
179 } finally {
    
```

Summary

- **How to access database from web application?**
 - JDBC
 - Relational Database Overview
 - JDBC and JDBC Drivers
 - JDBC Basics: Processing SQL Statements
 - Implement CRUD application using MS SQL

Q&A

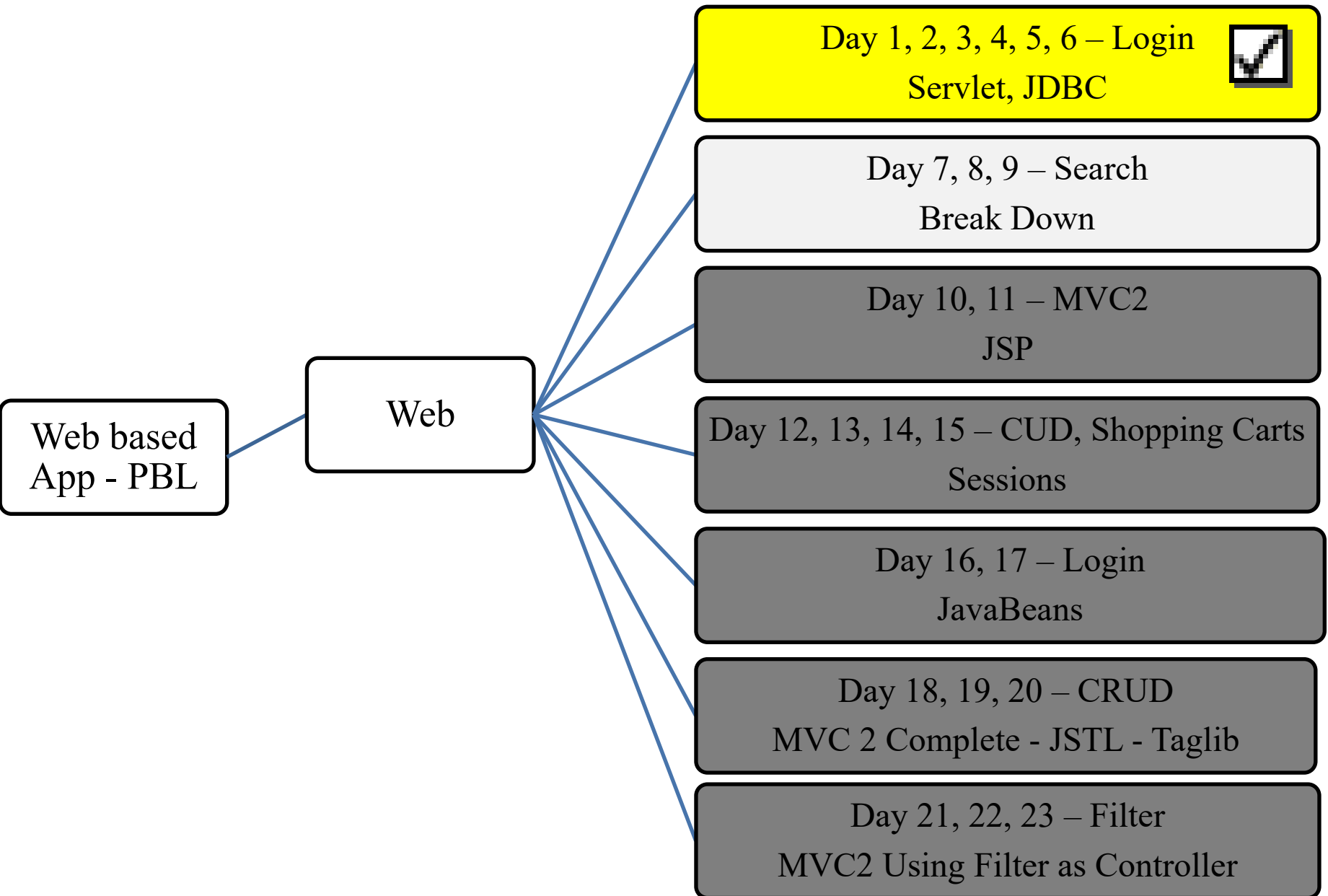
Exercises

- Do it again all of demos
- Using servlet to write the programs as the following requirement
 - Present the Login form (naming LoginServlet) with title Login, header h1 – Login, 02 textbox with naming txtUser and txtPass, and the Login button
 - Rewrite above Login application combining with DB
 - Writing the ColorServlet that presents “Welcome to Servlet course” with yellow in background and red in foreground
 - Writing the ProductServlet includes a form with a combo box containing Servlet & JSP, Struts & JSF, EJB, XMJ, Java Web Services, and the button with value Add to Cart

Next Lecture

- **How to deploy the Web Application to Web Server?**
 - Web applications Structure
 - Request Parameters vs. Context Parameters vs. Config/Servlet Parameters
 - Application Segments vs. Scope
- **How to transfer from resources to others with/without data/objects?**
 - Attributes vs. Parameters vs. Variables
 - Redirect vs. RequestDispatcher
 - RequestDispatcher vs. Filter

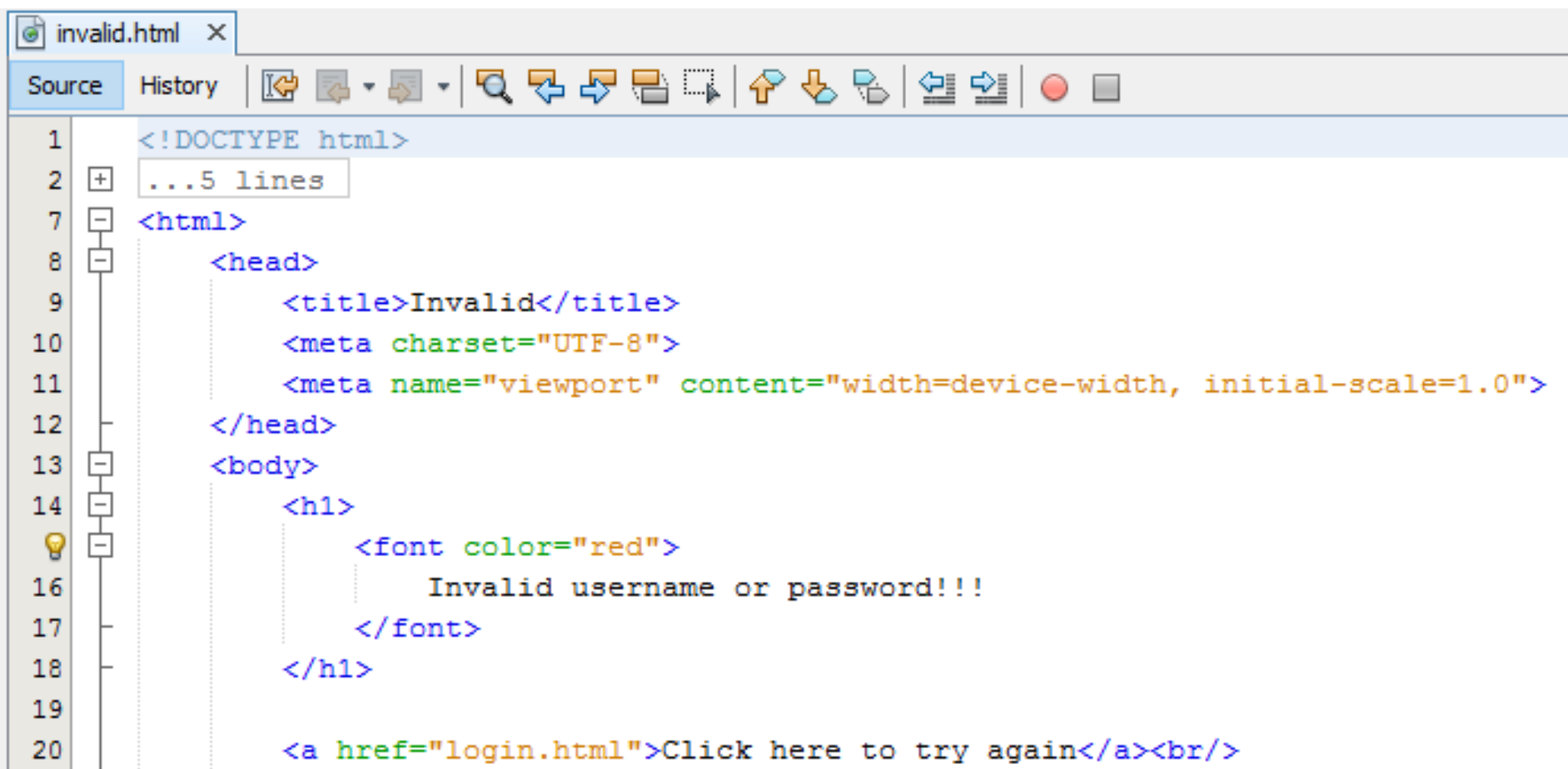
Next Lecture



Appendix – Build The Simple Web Login Page

```
login.html x
Source History
1 <!DOCTYPE html>
2 ...5 lines
7 <html>
8   <head>
9     <title>Login</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <h1>Login Page</h1>
15
16    <form action="SE1162Servlet" method="POST">
17      Username <input type="text" name="txtUsername" value="" /><br/>
18      Password <input type="password" name="txtPassword" value="" /><br/>
19      <input type="submit" value="Login" name="btAction" />
20      <input type="reset" value="Reset" />
21    </form>
```

Appendix – Build The Simple Web Invalid Page



```

1  <!DOCTYPE html>
2  ...5 lines
7  <html>
8      <head>
9          <title>Invalid</title>
10         <meta charset="UTF-8">
11         <meta name="viewport" content="width=device-width, initial-scale=1.0">
12     </head>
13     <body>
14         <h1>
15             <font color="red">
16                 Invalid username or password!!!
17             </font>
18         </h1>
19
20     <a href="login.html">Click here to try again</a><br/>
  
```

Appendix – Build The Simple Web Search Page

```

search.html x
Source History
1 <!DOCTYPE html>
2 ...5 lines
7 <html>
8   <head>
9     <title>Search</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <h1>Search Page</h1>
15    <form action="SE1162Servlet">
16      Search Value <input type="text" name="txtSearchValue" value="" /><br/>
17      <input type="submit" value="Search" name="btAction" />
18    </form>
19  </body>
20 </html>
21

```


Appendix – Build The Simple Web Servlet

```

SE1162Servlet.java x
Source History
23  * @author kieukhanh
24  */
25  public class SE1162Servlet extends HttpServlet {
26      private final String searchPage = "search.html";
27      private final String invalidPage = "invalid.html";
28      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines *
29
30      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31          throws ServletException, IOException {
32          response.setContentType("text/html;charset=UTF-8");
33          PrintWriter out = response.getWriter();
34          try {
35              String button = request.getParameter("btAction");
36              String url = invalidPage;
37              if (button.equals("Login")) {
38                  String username = request.getParameter("txtUsername");
39                  String password = request.getParameter("txtPassword");
40
41                  RegistrationDAO dao = new RegistrationDAO();
42                  boolean result = dao.checkLogin(username, password);
43
44                  if (result) {
45                      url = searchPage;
46                  }
47              }
48              response.sendRedirect(url);
49          } catch (NamingException ex) {
50              ex.printStackTrace();
51          } catch (SQLException ex) {
52              ex.printStackTrace();
53          } finally {
54              out.close();
55          }
56      }
57  }
58
59
60
61
  
```

Appendix – Build The Simple Web DAO

```

RegistrationDAO.java x
Source History
20 * @author kieukhanh
21 */
22 public class RegistrationDAO implements Serializable {
23     public boolean checkLogin(String username, String password)
24         throws SQLException, NamingException {
25         Connection con = null;
26         PreparedStatement stm = null;
27         ResultSet rs = null;
28         try {
29             con = DBUtils.makeConnection();
30             if (con != null) {
31                 String sql = "Select * From Registration Where username = ? And password = ?";
32
33                 stm = con.prepareStatement(sql);
34                 stm.setString(1, username);
35                 stm.setString(2, password);
36
37                 rs = stm.executeQuery();
38                 if (rs.next()) {
39                     return true;
40                 }
41             }
42         } finally {
43             if (rs != null) {
44                 rs.close();
45             }
46             if (stm != null) {
47                 stm.close();
48             }
49             if (con != null) {
50                 con.close();
51             }
52         }
53         con.close();
54     }
55 }
56
57 return false;
58 }

```