

MySQL JDBC Connection and State Management

Outline

- ❖ JDBC(Java Database Connectivity)
- ❖ State Management
 - Cookie
 - Session



JDBC(Java Database Connectivity)

- ❖ JDBC is the standard developed in java to interact with any Data-Source to store data onto the secondary storage permanently.
- ❖ The data source can be the spreadsheet, RDBMS and mainframe.
- ❖ JDBC API exist in the java.sql Package



Database Drivers

Each and every DBMS vendors provides the user interface for the user and their proprietary API(Set of function) for the Application Developer to work with the DBMS.



JDBC Driver(Thin Driver)

- ❖ The implemented classes are provided by the DBMS vendors.
- ❖ The classes in this driver directly access the database without invoking to the methods of Native API.
- ❖ This is the fastest JDBC driver mostly used in the industry. Different DBMS vendor provides this driver in the form of .jar file.
 - In case of MySQL ---- mysql-con.jar
 - In case of Oracle ---- ojdbc14.jar



Interfaces in the java.sql(package)

❖ **Connection**

- The object of the implemented class of this interface represents to the connection between the java class & DBMS.
- All the operation on the DBMS always be performed after getting the connection object.

❖ **Statement**

- Object of the implemented class of Statement is used to invoke all types of queries.



Continue...

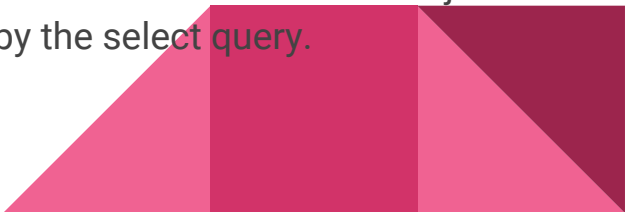
❖ **PreparedStatement**

- An instance of the PreparedStatement is use to execute the parameterised queries.

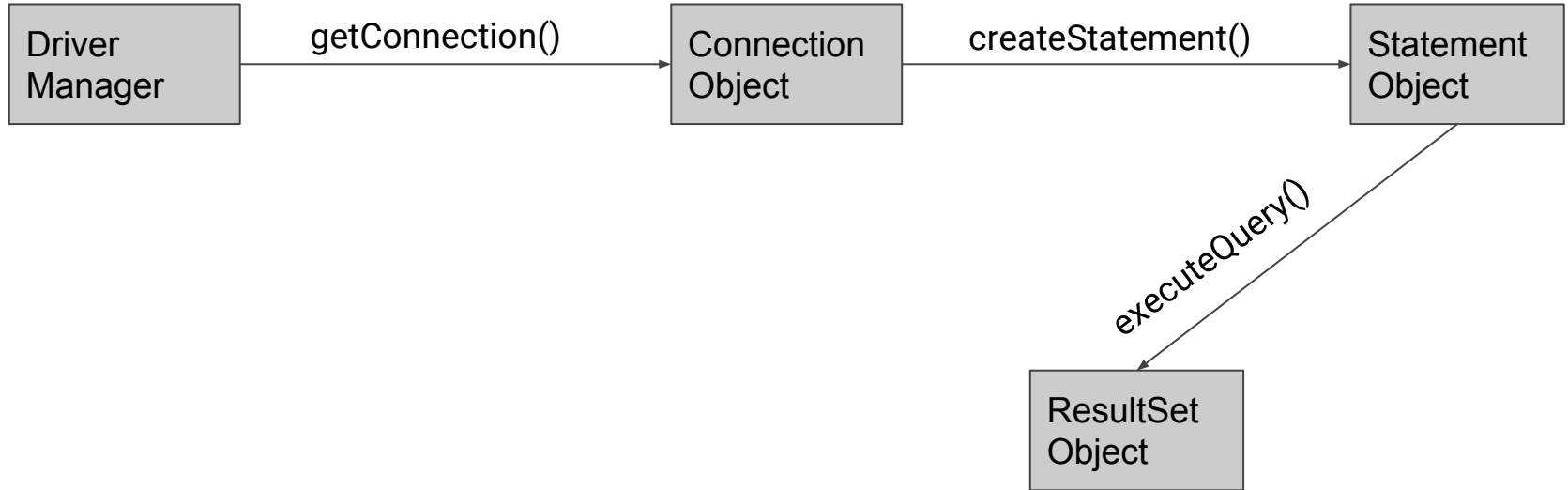
❖ **CallableStatement**

- Object of the implemented class of callable statement use to invoke the PL/SQL stored procedure and function.

❖ **ResultSet**

- An instance of ResultSet represent to the cursor of the selected records that means in the java Application the ResultSet object contains the records selected by the select query.
- 

Flow Diagram to implement JDBC



Steps to implement the JDBC

❖ Load the driver class

- Each and every JDBC driver contains the driver class. All the driver classes are the implemented classes of java.sql.Driver interface.
- Eg.
 - `Class.forName("com.mysql.jdbc.Driver")`

❖ Create the Connection

- Driver Manager class provides the static method to create the connection object.
- Methods
 - `public static Connection getConnection(String connectionUrl)`
 - `public static Connection getConnection(String connectionURL, String username, String password)`
- Eg.
 - `Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/database1", "root", "abc123")`

Continue...

❖ Create the Statement Object

- Connection provides the method to create the statement object.
- Method
 - public Statement *createStatement()*
- Eg. Statement st = con.createStatement();

❖ Execute the query

- Statement interface provide the method to execute the queries.
- Methods
 - public int executeUpdate(String nonSelectQuery)
 - public ResultSet executeQuery(String selectQuery)
 - Eg. ResultSet rs = st.executeQuery("select * from Student");

Continue...

❖ The ResultSet provides the methods of two types

➤ Methods to navigate the record pointer.

- public boolean next()
 - This method will move the record pointer on to the next record. If pointer is already on to the last record then this method returns false.
- public boolean previous()
 - This method will move the record pointer on to the previous record, return false if pointer is already on to the first record.
- public boolean first()
 - This method move the record pointer on to the first record.
- public boolean last()
 - This method move the record pointer directly on to the last record.
- public void absolute(int RecordNumber)

Continue...

❖ Methods to retrieve the values from the selected record-

- public String getString(int columnNumnber)
 - This method return the value of the specified column.
- public String getString(String columnName)
- public int getInt(String columnNumber)
- public int getInt(String columnName)



Example

```
import java.io.IOException;  
import java.io.PrintWriter;
```

```
import java.sql.*;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;
```

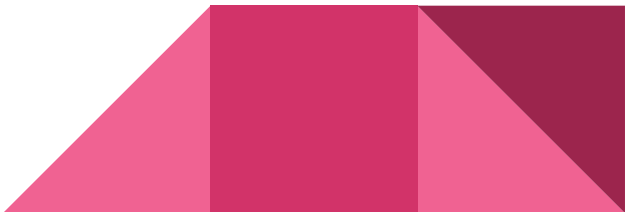
```
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;
```



```
public class DatabaseShow extends HttpServlet {  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException  
    {  
        response.setContentType("text/html");  
        try {  
            // Database Connection  
            String dbDriver = "com.mysql.cj.jdbc.Driver";  
            String dbURL = "jdbc:mysql:// localhost:3306/";  
            // Database name to access  
            String dbName = "database1";  
            String dbUsername = "root";  
            String dbPassword = "abc123";
```

```
Class.forName(dbDriver); //load the class
Connection con = DriverManager.getConnection(dbURL + dbN, dbUsername,
dbPassword); // Initialize the database
Statement st = con.createStatement();    // Create a SQL query to select data from the
table Student

ResultSet rs = st.executeQuery("Select * from Student");    // Execute query
PrintWriter out = response.getWriter();    // get the writer object to print
out.println("List of Students");
out.print("<br>");
out.print("<table><tr>");
out.print("<td>Name</td>");
out.print("<td>Subject</td>");
out.print("<td>Marks</td>");
out.print("</tr>");
```



```
while(rs.next()){
    out.print("<tr>");
    out.print("<td>" + rs.getString(1) + "</td>");
    out.print("<td>" + rs.getString(2) + "</td>");
    out.print("<td>" + rs.getString(3) + "</td>");
    out.print("</tr>");
}
out.print("</table>");
st.close();                // Close all the connections
con.close();

}
catch (Exception e)
{ e.printStackTrace(); }
} }
```



State Management

❖ Http is the stateless protocol

- Whenever the client browser submit the request a new Http packet is created and the request data stored into it and then that packet transmitted over the internet towards the web server.
- The web server gets the request parameter and process the request and then loaded the response content into that packet. The packet returned back on to the browser and web browser retrieves the header and response content after that immediately the HttpServlet gets destroyed.
- For the new request new packet will be created. Due to this behaviour of Http protocol the submitted data in the one request cannot be accessed in the further any other request. So we have to maintain some logic on the server side to manage the state of the client(web browser) that means to access the data of one request into further any other request from the same browser.

- ❖ For the new request new packet will be created. Due to this behaviour of Http protocol the submitted data in the one request cannot be accessed in the further any other request. So we have to maintain some logic on the server side to manage the state of the client(web browser) that means to access the data of one request into further any other request from the same browser.
- ❖ Most of the web application developed based on the state management for example Shopping site, Online reservation(IRCTC).
- ❖ State Management can be done by following ways:
 - Cookie
 - Session



Cookie

- ❖ Cookies are small files which are stored on a user's computer. They are designed to hold a modest amount of data specific to a particular client and website, and can be accessed either by the web server or the client computer.
- ❖ Cookies always be created onto the server-side and transmitted towards the client side with the response to be stored in the cache of the client browser either temporary or permanently



javax.servlet.http.Cookie

❖ An instance of this class represents a cookie onto the server.

❖ **Constructor**

➤ public Cookie(String name, String value)

❖ **Methods**

■ public String getName()

- This method will return the name of the cookie.

■ public String getValue()

- This method will return the name of the cookie.

■ public void setMaxAge(int sec)

- This method will have make the cookie persistent or non-persistent.
- Negative value means non-persistent cookies and positive values makes the cookies persistent for the specified time(sec).
- If age is 0 then browser immediately destroy to that cookie.

Additional methods of HttpServletResponse and HttpServletRequest to handle Cookies

❖ Methods

- public void addCookie(Cookie c)
 - method of the **HttpServletResponse** is used to add the cookie onto the Response.
- public Cookie[] getCookies()
 - Method of the HttpServletRequest is used to get the cookies from the request.



Example

// index.html

```
<html>
```

```
<body>
```

```
  <form action = "login" method = "get">
```

```
    Enter name: <input type = "text" value = " " name = "name1">
```

```
    <input type = "submit" value = "login" >
```

```
  </form>
```


```
</body>
```

```
</html>
```



```
//LoginServlet
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
public class LoginServlet extends HttpServlet{
protected void doGet(HttpServletRequest req,HttpServletResponse res)
                    throws IOException,ServletException{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    String name = req.getParameter("name1");
    out.println("Welcome "+name);
    Cookie ck = new Cookie("user",name); // creating a cookie and setting the name-value pair
    ck.setMaxAge(600);
    res.addCookie(ck); //adding cookie to response object
    out.println("<br><a href = 'home'>Home</a>");
}
}
```



// HomeServlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.http.Cookie;

public class HomeServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest req,HttpServletResponse res)
        throws IOException,ServletException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        Cookie ck[] = req.getCookies();
```




```
String nm = null;
for(Cookie c : ck)
{
    if(c.getName().equals("user"))
    {
        nm =c.getValue();
        break;
    }
}
out.println("Sorry "+nm);

out.println("<br><center><h1>Site is not ready</h1></center>");
}
}
```



//web.xml

<web-app>

 <servlet>

 <servlet-name>servlet1</servlet-name>

 <servlet-class>LoginServlet</servlet-class>

 </servlet>

 <servlet-mapping>

 <servlet-name>servlet1</servlet-name>

 <url-pattern>/login</url-pattern>

 </servlet-mapping>



```
<servlet>  
  <servlet-name>servlet2</servlet-name>  
  <servlet-class>HomeServlet</servlet-class>  
</servlet>
```

```
<servlet-mapping>  
  <servlet-name>servlet2</servlet-name>  
  <url-pattern>/home</url-pattern>  
</servlet-mapping>
```

```
</web-app>
```

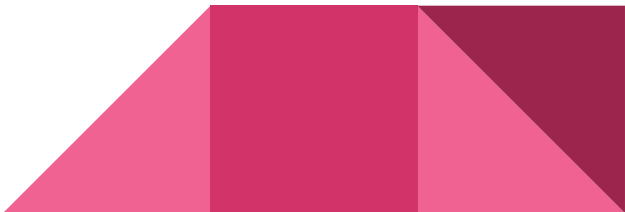


Session

- ❖ Session is the object on to the web server for any specific client(web browser).
- ❖ Session object is also capable to maintain the attributes.
- ❖ In the form of attribute the client specific data can be stored into the session.
- ❖ From each client web server maintain the separate session object. The session object of any specific client is the concept of session tracking.



javax.servlet.http.HttpSession

- ❖ This is an interface and implemented class is provided by all the web-servers.
 - ❖ An instance of this class represents the session.
 - ❖ **Methods**
 - `public void setAttribute(String attributeName, Object attributeValue)`
 - `public Object getAttribute(String attributeName)`
 - `public void removeAttribute(String attributeName)`
 - `public boolean isNew()`
 - `public void invalidate()`
 - `public HttpSession getSession(boolean b)`
- 

Additional methods of HttpServletRequest to handle Session

❖ Methods

➤ public HttpSession getSession(boolean b)


- This method is used to returns either new session object or the existing session object.
- If in the current request, the session Id is exist but the matched session object is not available onto the server(already been invalidate). Also new session object returns.
- If in the request, sessionId is exist and also the matched session object found then the matched existing session object return.
- The argument boolean variable denotes the whether the new session object will be created or not if existing session object does not traced.



// LoginServlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class LoginServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest req,HttpServletResponse res)
        throws IOException,ServletException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        String name = req.getParameter("name1");
        out.println("Welcome "+name);
    }
}
```



```
HttpSession session = req.getSession();  
session.setAttribute("user",name);    //setting session with attribute user and value name  
out.println("<br><a href = 'home'>Home</a>");
```

```
}
```


```
}
```



// HomeServlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.http.Cookie;

public class HomeServlet extends HttpServlet{
    protected void doGet(HttpServletRequest req,HttpServletResponse res)
        throws IOException,ServletException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        HttpSession session = req.getSession(); // getting Session object
        String nm = (String)session.getAttribute("user"); //getting User value
        out.println("Sorry "+nm);
        out.println("<br><center><h1>Site is not ready</h1></center>");
    }
}
```



//web.xml

<web-app>

 <servlet>

 <servlet-name>servlet1</servlet-name>

 <servlet-class>LoginServlet</servlet-class>

 </servlet>

 <servlet-mapping>

 <servlet-name>servlet1</servlet-name>

 <url-pattern>/login</url-pattern>

 </servlet-mapping>



```
<servlet>  
  <servlet-name>servlet2</servlet-name>  
  <servlet-class>HomeServlet</servlet-class>  
</servlet>
```

```
<servlet-mapping>  
  <servlet-name>servlet2</servlet-name>  
  <url-pattern>/home</url-pattern>  
</servlet-mapping>
```

```
</web-app>
```

