SQL

Class 5

Agenda

DB Testing with Selenium

Database Testing with Selenium

Selenium Webdriver is limited to testing your applications using Browser. To use Selenium Webdriver for Database Testing you need to use the JDBC ("Java Database Connectivity").

JDBC (Java Database Connectivity) is a SQL level API that allows you to execute SQL statements.

It is responsible for the connectivity between the Java Programming language and a wide range of databases.

JDBC is Database Independent

JDBC is Not SQL Independent

Database Testing with Selenium

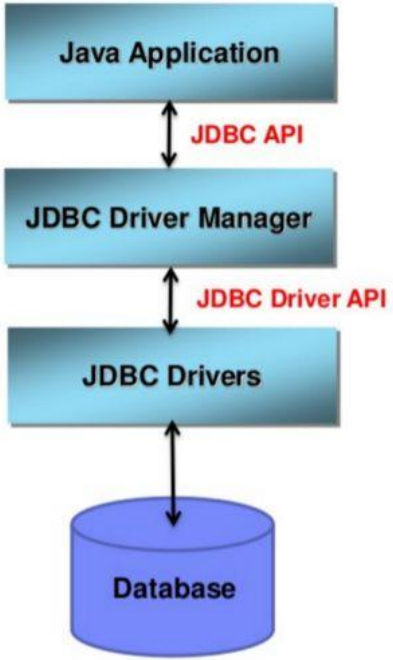
JDBC is used to interact with the various type of Database such as Oracle, MS Access, Mysql and SQL Server. To connect Java with different Databases we have different drivers and jar files.

Using JDBC we can execute the queries using Java Application and retrieve the results in Database.

To connect Oracle to our java Application, we need to download Oracle JDBC.

Database Testing with Selenium

JDBC architecture



Database Testing with Selenium

JDBC API provides the following classes and interfaces:

1. Driver Manager Class
2. Connection Interface
3. Statement Interface
4. ResultSet Interface
5. SQLException Class

Database Testing with Selenium

In order to test your Database using Selenium, you need to observe the following steps

1. Create a connection to the Database
2. Create Statement
3. Execute Queries
4. Process Results
5. Close Connection

Connecting to Oracle

First we will download Oracle JDBC and add it as external jar

https://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html

Now we need information about our DataBase:

HostName: syntaxdatabase.cdjflmucstpo.us-east-1.rds.amazonaws.com

Post: 1521

SID: orcl

UserName: Syntax

Password: syntax123

1) Make a connection to the Database

In order to make a connection to the database we need to use getConnection() method of DriverManager

```
DriverManager.getConnection(DatabaseURL, "userid", "password" )
```

- Userid is the username configured in the database
- Password of the configured user
- DatabaseURL is of format `jdbc:jdbctype:@ipaddress:portnumber:db_type`
- **jdbctype**- The driver for the database you are trying to connect. To connect to oracle database this value will be "oracle:thin"

1) Make a connection to the Database

- DataBaseURL connection for our database will be:

  "jdbc:oracle:thin:@syntaxdatabase.cdjflmucstpo.us-east-1.rds.amazonaws.com:1521:orcl"

- And the code to create connection looks like

```
Connection con = DriverManager.getConnection(DatabaseURL,username,password);
```

- The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.
- A Connection is the session between java application and database.

2) Create the Statement object

- The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

```
Statement stmt = con.createStatement();
```

The Statement interface provides methods to execute queries with the database.

3) Execute the query

- Once the statement object is created we can use executeQuery method to execute the SQL queries

```
ResultSet rSet=stmt.executeQuery(“select * from Departments”);
```

The object of ResultSet maintains a cursor pointing to a row of a table.

Results from the executed query are stored in the ResultSet Object.

4) Process the results

public boolean next()	used to move the cursor to the one row next from the current position.
public boolean previous()	iused to move the cursor to the one row previous from the current position.
public boolean last():	used to move the cursor to the last row in result set object.
public boolean absolute(int row)	used to move the cursor to the specified row number in the ResultSet object.
public int getInt(int columnIndex)	used to return the data of specified column index of the current row as int.
public int getInt(String columnName)	used to return the data of specified column name of the current row as int.
public String getString(int columnIndex)	used to return the data of specified column index of the current row as String
public String getString(String columnName)	is used to return the data of specified column name of the current row as String

```
public class JDBCDemo {

    String DbUrl = "jdbc:oracle:oci:@syntaxdatabase.cdjflmucstpo.us-east-1.rds.amazonaws.com:1521:orcl";
    String userName = "Syntax";
    String password = "syntax123";

    @Test
    public void firstTest() throws SQLException, ClassNotFoundException {

        Connection conn = DriverManager.getConnection(DbUrl, userName, password);

        Statement statement = conn.createStatement();

        ResultSet resultSet = statement.executeQuery("Select * from Departments");

        resultSet.next();
        System.out.println(resultSet.getString("department_name"));

        while (resultSet.next()) {
            System.out.println(resultSet.getInt("Department_Id") + "=" + resultSet.getString("Department_Name"));
        }
        conn.close();
    }
```

MetaData

The metadata means data about data i.e. we can get further information from the data.

JDBC Metadata supports:

- DatabaseMetaData
- ResultSetMetaData

DatabaseMetaData

DatabaseMetaData interface provides methods to get meta data of a database such as database product name, database product version, driver name, name of total number of tables, name of total number of views

public String getDriverName()	it returns the name of the JDBC driver
public String getUserName()	it returns the username of the database
public String getDatabaseProductName()	it returns the product name of the database
public String getDatabaseProductVersion()	it returns the product version of the database

ResultSetMetaData

ResultSetMetaData interface provides methods to get metadata from ResultSet such as total number of column, column name, column type.

public int getColumnCount()	it returns the total number of columns in the ResultSet object
public String getColumnName(int index)	it returns the column name of the specified column index
public String getColumnTypeName(int index)	it returns the column type name for the specified index
public String getTableName(int index)	it returns the table name for the specified column index.

```java
public class JDBCDemo {

    String DbUrl = "jdbc:oracle:oci:@syntaxdatabase.cdjflmucstpo.us-east-1.rds.amazonaws.com:1521:orcl";
    String userName = "Syntax";
    String password = "syntax123";

    @Test
    public void metaData() throws SQLException {
        Connection conn = DriverManager.getConnection(DbUrl, userName, password);
        Statement statement = conn.createStatement();
        ResultSet resultSet = statement.executeQuery("Select * from Employees");
        DatabaseMetaData databaseData = conn.getMetaData();
        String dbName = databaseData.getDatabaseProductName();
        String dbVersion = databaseData.getDatabaseProductVersion();
        String driverName = databaseData.getDriverName();
        System.out.println(dbName + "-" + dbVersion + " " + driverName);

        ResultSetMetaData rsData = resultSet.getMetaData();
        int cols = rsData.getColumnCount();
        String colName = rsData.getColumnName(1);

        for (int i = 1; i <= cols; i++) {
            System.out.println(rsData.getColumnName(i));
        }

        conn.close();
    }
```

Processing ResultSet

Results from the executed query are stored in the ResultSet Object.
Now we will need Java data structure to store those results
Result might contain multiple rows, so the best way is store it inside List of Maps

List<Map<String, Object>>

Map<String, V> - store column name
Map <K, Object> - store column value

Map<> stores 1 row as Key & Value
List<> stores all rows

Processing ResultSet

```
public class JDBCDemo {

    String DbUrl = "jdbc:oracle:oci:@syntaxdatabase.cdjflmucstpo.us-east-1.rds.amazonaws.com:1521:orcl";
    String userName = "Syntax";
    String password = "syntax123";

    @Test
    public void storeData() throws SQLException {
        Connection conn = DriverManager.getConnection(DbUrl, userName, password);
        Statement state = conn.createStatement();
        ResultSet result = state.executeQuery("Select first_name, last_name from employees where job_id='IT_PROG'");
        List<Map<String, Object>> rsdata = new ArrayList<>();
        Map<String, Object> rowData;
        while (result.next()) {
            rowData = new HashMap<>();
            rowData.put("first_name", result.getObject("first_name"));
            rowData.put("last_name", result.getObject("last_name"));
            rsdata.add(rowData);
        }

        System.out.println(rsdata.size());
        conn.close();
    }
```

Processing ResultSet

```
public class JDBCDemo {

    String DbUrl = "jdbc:oracle:oci:@syntaxdatabase.cdjflmucstpo.us-east-1.rds.amazonaws.com:1521:orcl";
    String userName = "Syntax";
    String password = "syntax123";

    @Test
    public void storeAllColumns() throws SQLException {
        Connection conn = DriverManager.getConnection(DbUrl, userName, password);
        Statement statement = conn.createStatement();
        ResultSet result = statement.executeQuery("Select * from employees where job_id='IT_PROG'");

        ResultSetMetaData rsetMetaData = result.getMetaData();
        int cols = rsetMetaData.getColumnCount();
        List<Map<String, Object>> rsList = new ArrayList<>();

        while (result.next()) {

            Map<String, Object> rowData = new HashMap<>();
            for (int i = 1; i <= cols; i++) {
                rowData.put(rsetMetaData.getColumnName(i), result.getObject(i));
            }
            System.out.println(rowData);
            rsList.add(rowData);
        }
        System.out.println(rsList);
        result.close();
        statement.close();
        conn.close();
    }
```