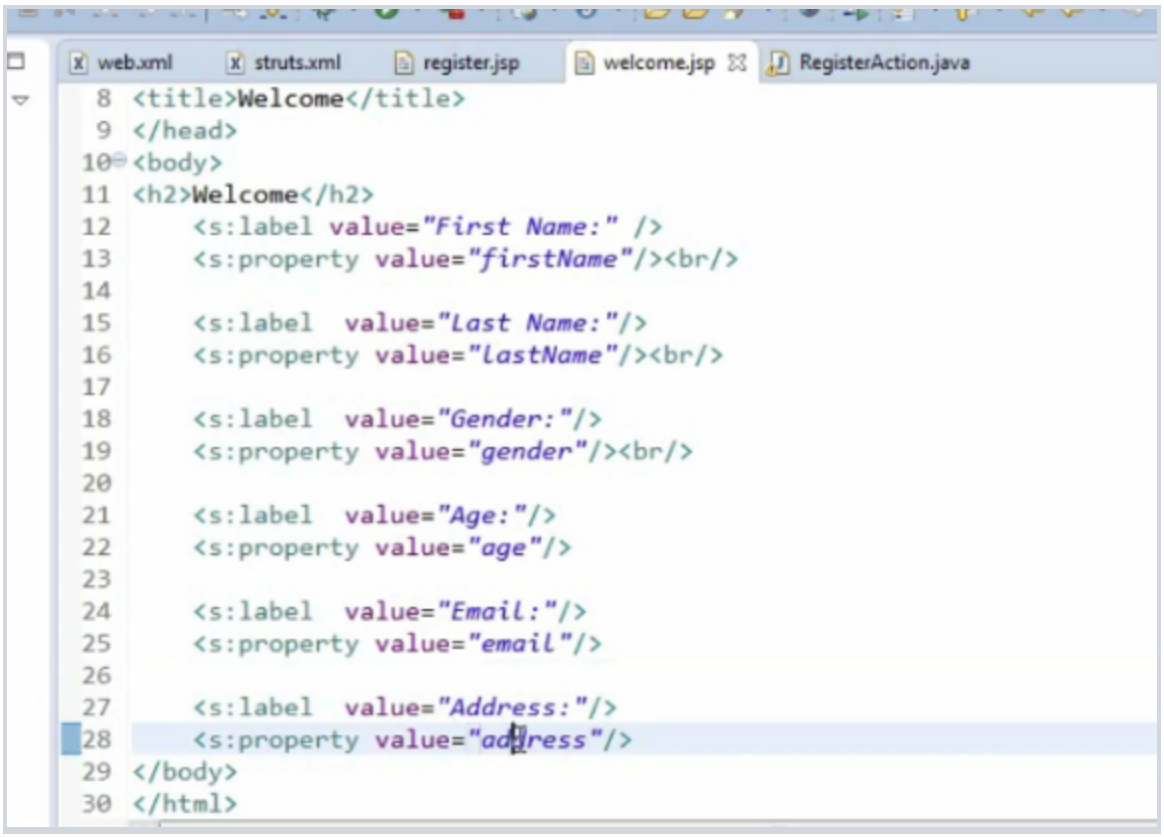
Struts - 2 Tags:

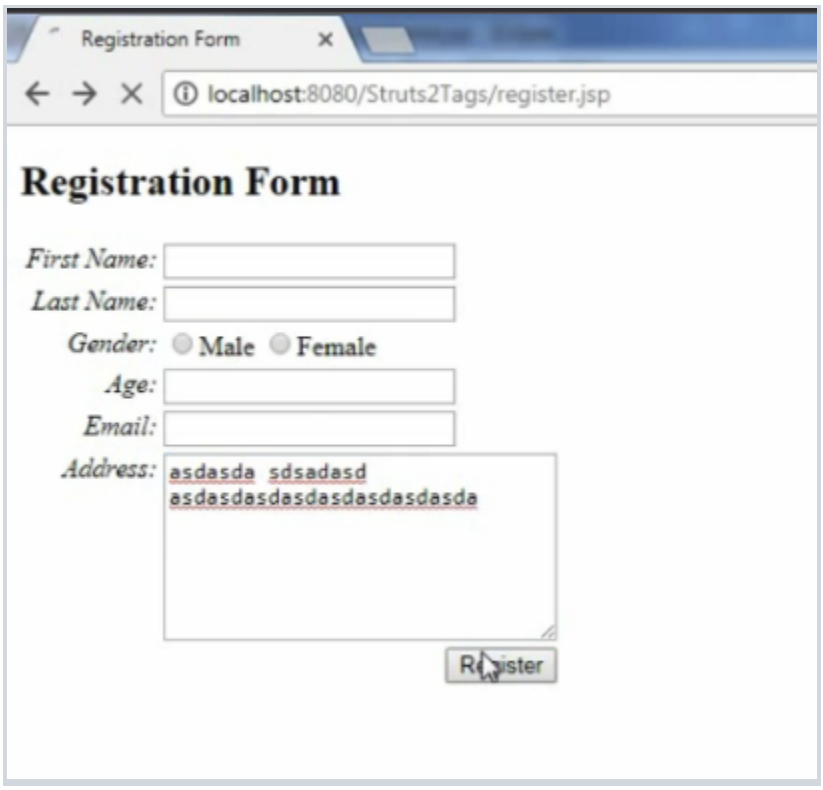
21. TextArea and Reset Tag

```
14
15 <s:form action="registerAction">
16     <s:textfield name="firstName" label="First Name" />
17     <s:textfield name="lastName" label="Last Name" />
18     <s:radio name="gender" list="{'Male','Female'}" label="Gender" />
19     <s:textfield name="age" label="Age" />
20     <s:textfield name="email" label="Email" />
21     <s:textarea name="address" cols="30" rows="7" label="Address"/>
22     <s:submit value="Register" />
23 </s:form>
24
```

web.xml | struts.xml | register.jsp | welcome.jsp | RegisterAction.java

```
1  import com.opensymphony.xwork2.ActionSupport;
2
3  public class RegisterAction extends ActionSupport{
4
5      String firstName;
6      String lastName;
7      String gender;
8      Integer age;
9      String email;
10     String address;
11
12     public String execute()
13     {
14         System.out.println("execute() method called");
```


web.xml
struts.xml
register.jsp
welcome.jsp
RegisterAction.java
8 <title>Welcome</title>
9 </head>
10 <body>
11 <h2>Welcome</h2>
12 <s:label value="First Name:" />
13 <s:property value="firstName"/>
14
15 <s:label value="Last Name:"/>
16 <s:property value="lastName"/>
17
18 <s:label value="Gender:"/>
19 <s:property value="gender"/>
20
21 <s:label value="Age:"/>
22 <s:property value="age"/>
23
24 <s:label value="Email:"/>
25 <s:property value="email"/>
26
27 <s:label value="Address:"/>
28 <s:property value="address"/>
29 </body>
30 </html>


Registration Form
localhost:8080/Struts2Tags/register.jsp
Registration Form
First Name:
Last Name:
Gender: Male Female
Age:
Email:
Address: asdasda sdsadasd
asdasdasdasdasdasdasdasda
Register

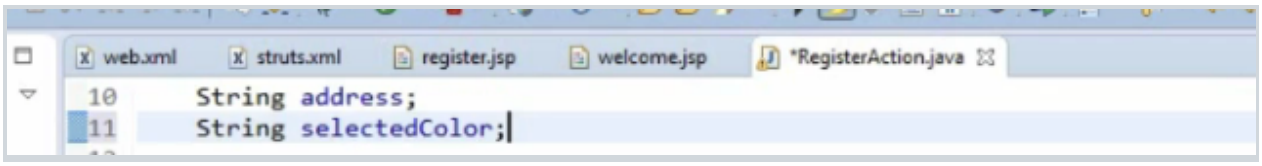
Reset Tag

```
20      <s:textfield name="email" label="Email" />
21      <s:textarea name="address" cols="30" rows="7" label="Address"/>
22      <s:reset value="Reset" />
23      <s:submit value="Register" />
24  </s:form>
```

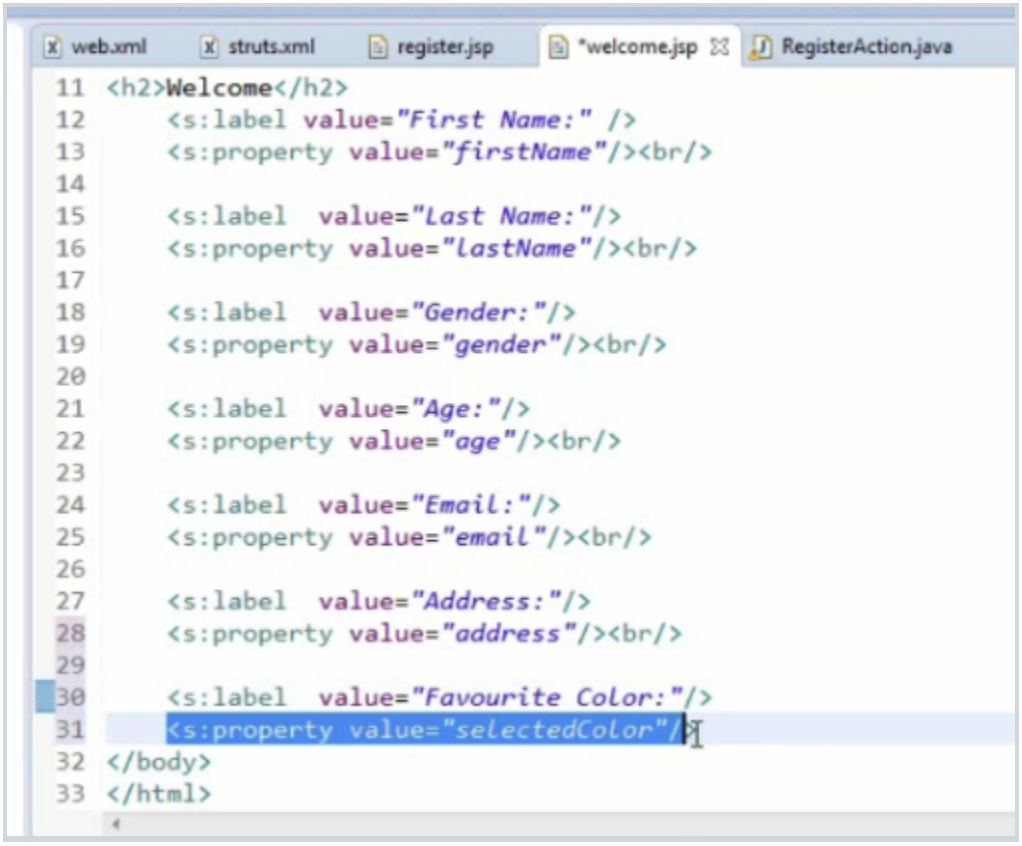
23. Select Tag: Part 2

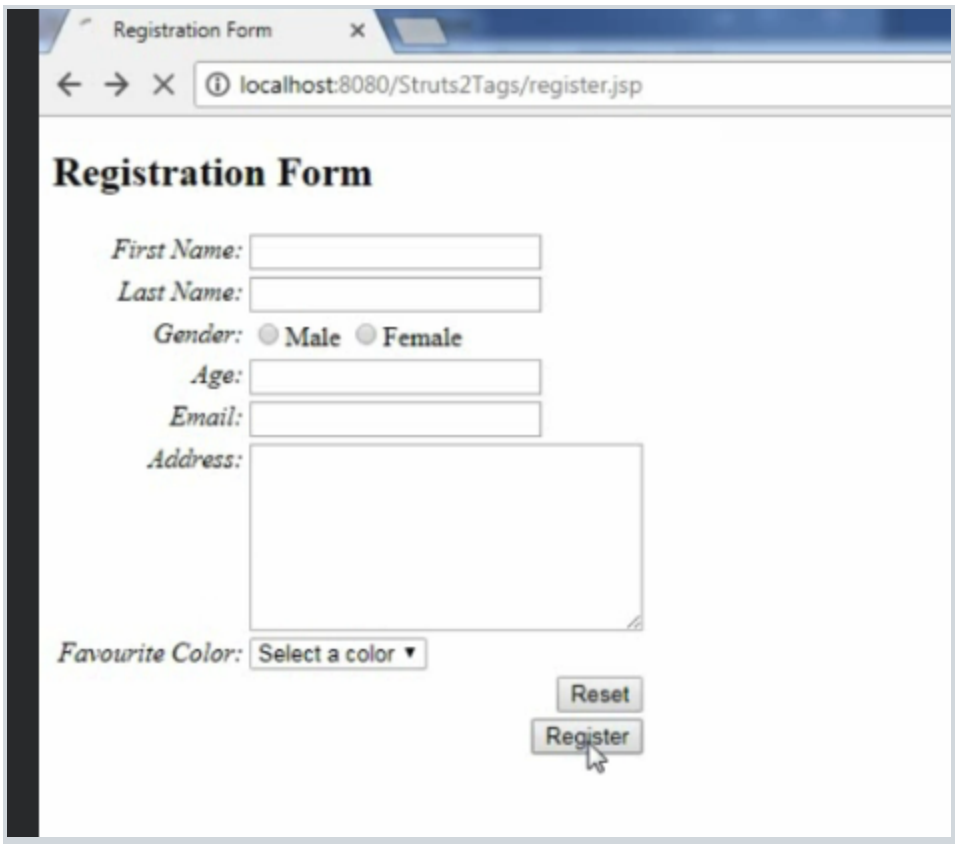
```
14
15  <s:form action="registerAction">
16      <s:textfield name="firstName" label="First Name" />
17      <s:textfield name="lastName" label="Last Name" />
18      <s:radio name="gender" list="{'Male','Female'}" label="Gender" />
19      <s:textfield name="age" label="Age" />
20      <s:textfield name="email" label="Email" />
21      <s:textarea name="address" cols="30" rows="7" label="Address"/>
22      <s:select list="{'Blue','Red','Green','White'}" name="selectedColor" headerKey="None" headerValue="Select a color"
23      <s:reset value="Reset" />
24      <s:submit value="Register" />
25  </s:form>
```

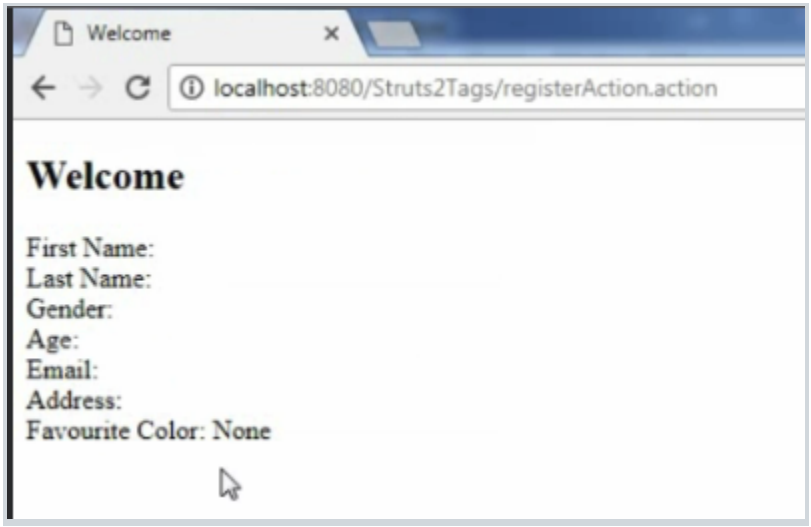
```
"7" label="Address"/>
te'}" name="selectedColor" headerKey="None" headerValue="Select a color" label="Favourite Color" />
```



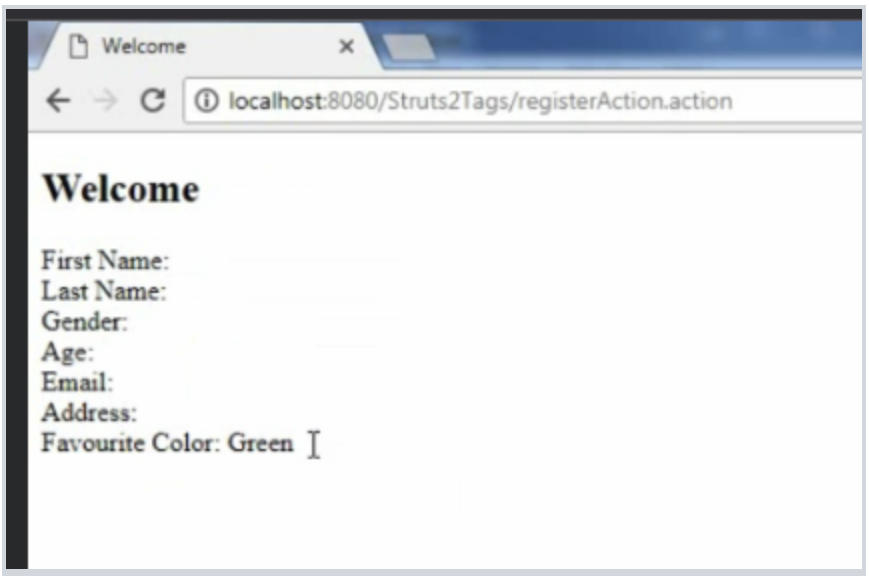
Add setters and getters







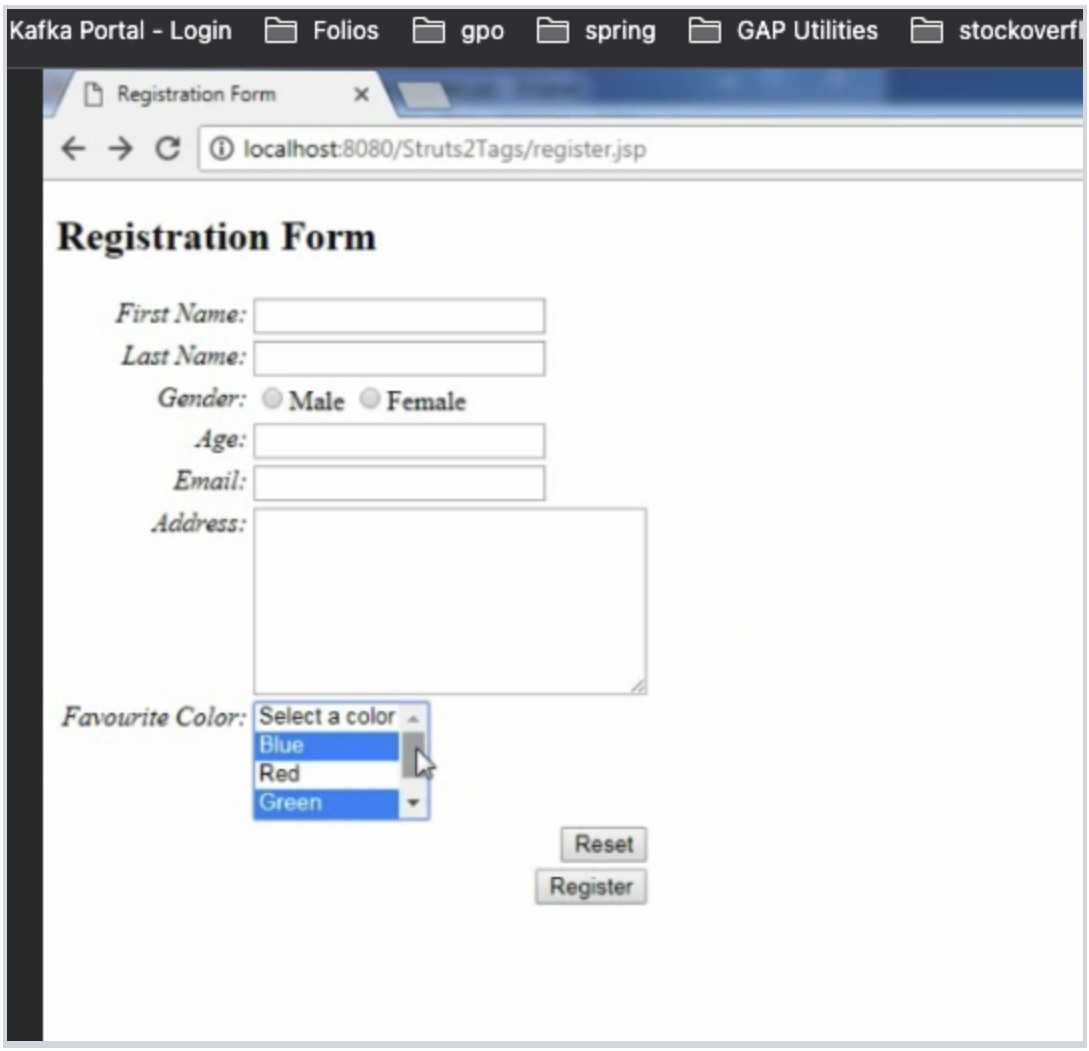
When we wont select then we will see None.



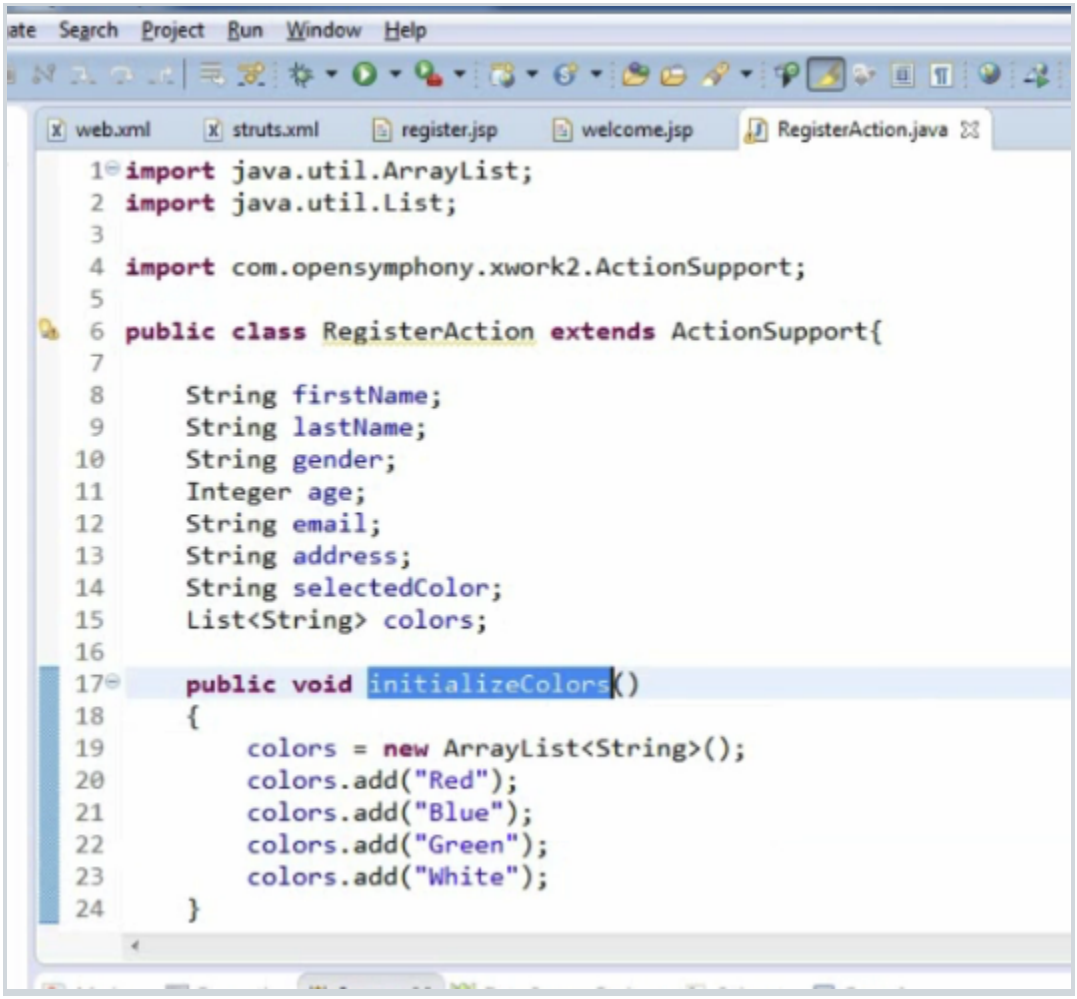
When we select some thing we will see that value.

```
<s:textarea name="address" cols="30" rows="7" label="Address"/>
<s:select multiple="true" list="{'Blue','Red','Green','White'}" name="selectedColor" headerKey="None" headerValue=
<s:reset value="Reset" />
<s:submit value="Register" />
```

Select multiple tag above



Otherway of selecting list


ate Search Project Run Window Help
web.xml
struts.xml
register.jsp
welcome.jsp
RegisterAction.java
1 import java.util.ArrayList;
2 import java.util.List;
3
4 import com.opensymphony.xwork2.ActionSupport;
5
6 public class RegisterAction extends ActionSupport{
7
8 String firstName;
9 String lastName;
10 String gender;
11 Integer age;
12 String email;
13 String address;
14 String selectedColor;
15 List<String> colors;
16
17 public void initializeColors()
18 {
19 colors = new ArrayList<String>();
20 colors.add("Red");
21 colors.add("Blue");
22 colors.add("Green");
23 colors.add("White");
24 }


public String initializeFormFields()
{
initializeColors();
return "input";
}
public void initializeColors()
{
colors = new ArrayList<String>();
colors.add("Red");
colors.add("Blue");

web.xml | struts.xml | register.jsp | welcome.jsp | *RegisterAction.java

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>
    <package name="register" extends="struts-default">
        <action name="registerAction" class="RegisterAction">
            <result name="success">/welcome.jsp</result>
            <result name="input">/register.jsp</result>
        </action>
        <action name="formLoadAction" method="initializeFormFields" class="RegisterAction">
            <result name="input">/register.jsp</result>
        </action>
    </package>
</struts>
```

Add action tag. Inorder call custom method apart from execute method use method attribute.

Replace array in jsp with variable name colors

```
    <s:textfield name="age" label="Age" />
    <s:textfield name="email" label="Email" />
    <s:textarea name="address" cols="30" rows="7" label="Address"/>
    <s:select list="colors" name="selectedColor" headerKey="None" headerValue="Select a
    <s:reset value="Reset" />
    <s:submit value="Register" />
</s:form>
```

24. Checkbox Tag

```
    <s:textarea name="address" cols="30" rows="7" label="Address"/>
    <s:select list="colors" name="selectedColor" headerKey="None" headerValue="Select a colo
    <s:checkbox name="subscription" value="true" label="Subscribe to our newsletter"/>
    <s:reset value="Reset" />
    <s:submit value="Register" />
```

23rd line

```
    List<String> colors;
    Boolean subscription;
```

Add setters and getters

```
18    List<String> hobbies;
19    String selectedHobbies;
20
21    public String initializeFormFields()
22    {
23        initializeColors();
24        initializeHobbies();
25        return "input";
26    }
```

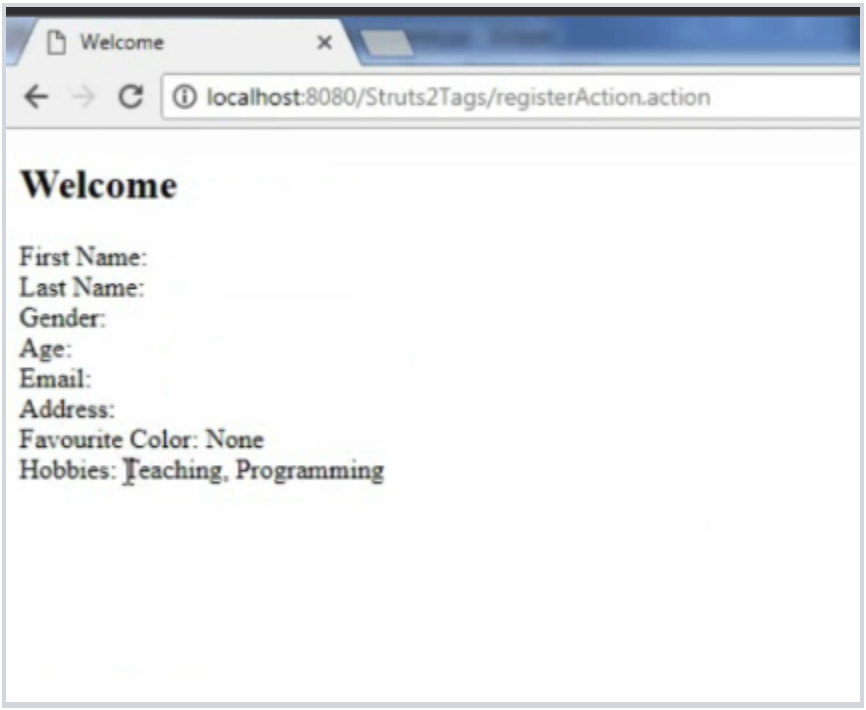
Introduce hobbies and selected hobbies and add setters and getters and create method initializeHobbies in initializeFormFields Method.

```
35
36    public void initializeHobbies()
37    {
38        hobbies = new ArrayList<String>();
39        hobbies.add("Drawing");
40        hobbies.add("Teaching");
41        hobbies.add("Singing");
42        hobbies.add("Programming");
43    }
44
```

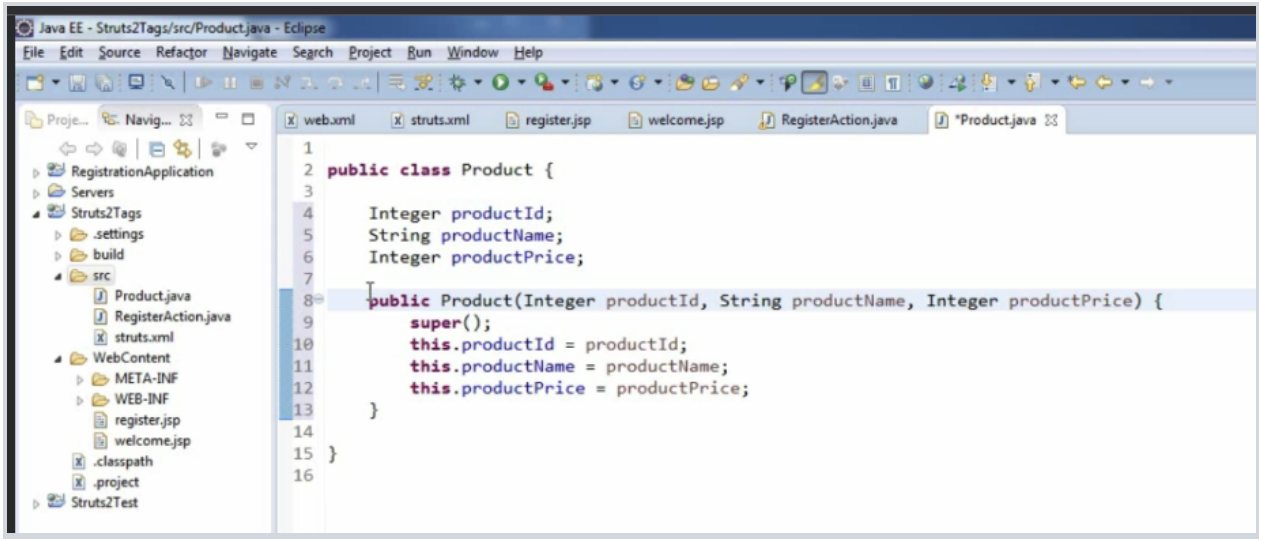
web.xml | struts.xml | register.jsp | welcome.jsp | RegisterAction.java

```
13    <s:property value="firstName"/><br/>
14
15    <s:label  value="Last Name:"/>
16    <s:property value="lastName"/><br/>
17
18    <s:label  value="Gender:"/>
19    <s:property value="gender"/><br/>
20
21    <s:label  value="Age:"/>
22    <s:property value="age"/><br/>
23
24    <s:label  value="Email:"/>
25    <s:property value="email"/><br/>
26
27    <s:label  value="Address:"/>
28    <s:property value="address"/><br/>
29
30    <s:label  value="Favourite Color:"/>
31    <s:property value="selectedColor"/><br/>
32
33    <s:label  value="Hobbies:"/>
34    <s:property value="selectedHobbies"/>
35 </body>
36 </html>
```

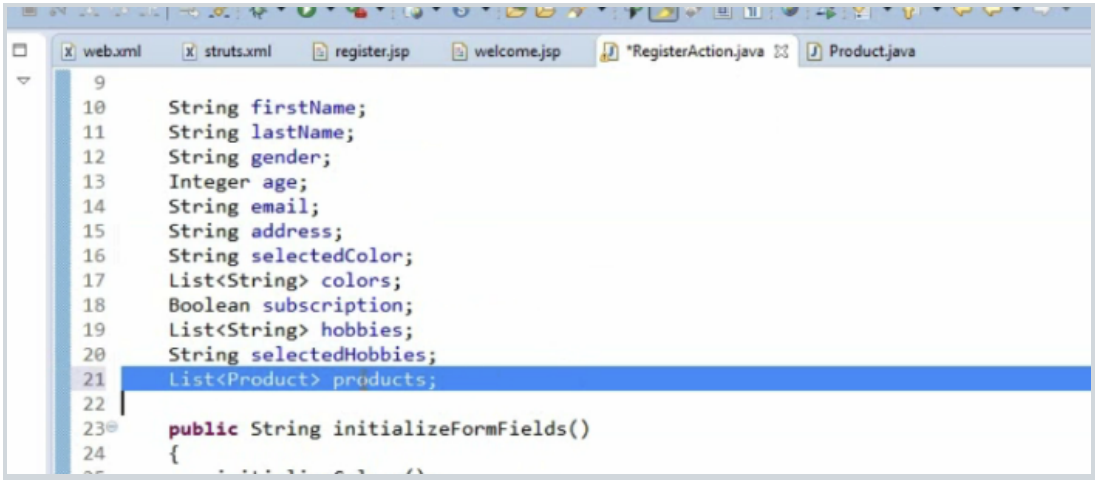

Registration Form
localhost:8080/Struts2Tags/formLoadAction
Registration Form
First Name:
Last Name:
Gender: Male Female
Age:
Email:
Address:
Favourite Color: Select a color
Subscribe to our newsletter
Hobbies: Drawing Teaching Singing Programming
Reset
Register


Welcome
localhost:8080/Struts2Tags/registerAction.action
Welcome
First Name:
Last Name:
Gender:
Age:
Email:
Address:
Favourite Color: None
Hobbies: Teaching, Programming

26. Iterator Tag

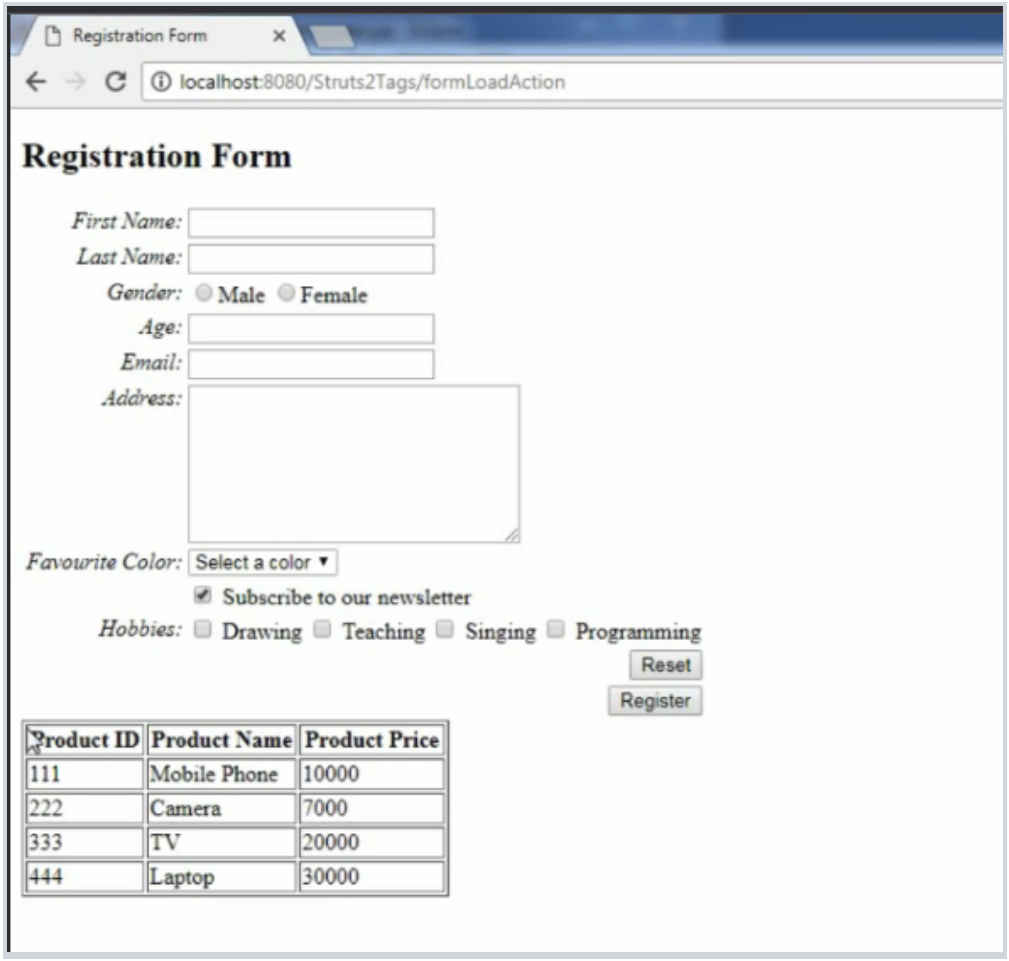


Add getters and setter methods for the above.



```
22
23      public String initializeFormFields()
24      {
25          initializeColors();
26          initializeHobbies();
27          initializeProducts();
28          return "input";
29      }
```

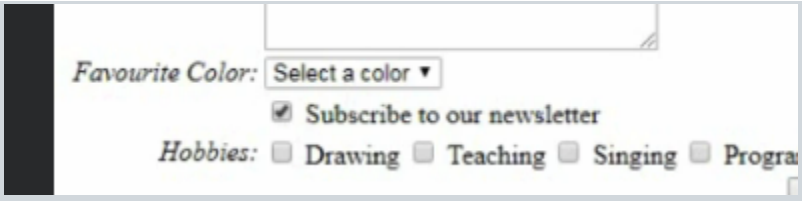
```
27  </s:form>
28      <table border="1" width="300">
29          <tr>
30              <th>Product ID</th>
31              <th>Product Name</th>
32              <th>Product Price</th>
33          </tr>
34          <s:iterator value="products" var="product">
35              <tr>
36                  <td>
37                      <s:property value="#product.productId"/>
38                  </td>
39                  <td>
40                      <s:property value="#product.productName"/>
41                  </td>
42                  <td>
43                      <s:property value="#product.productPrice"/>
44                  </td>
45              </tr>
46          </s:iterator>
47      </table>
```

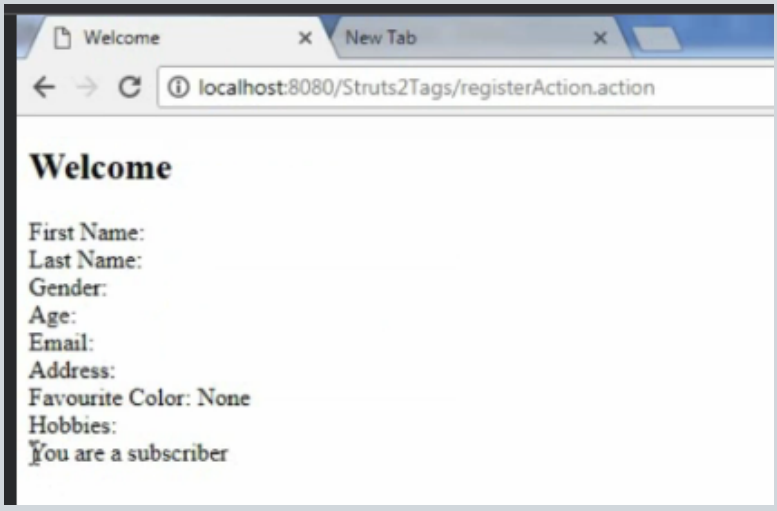


27. If Else Tag

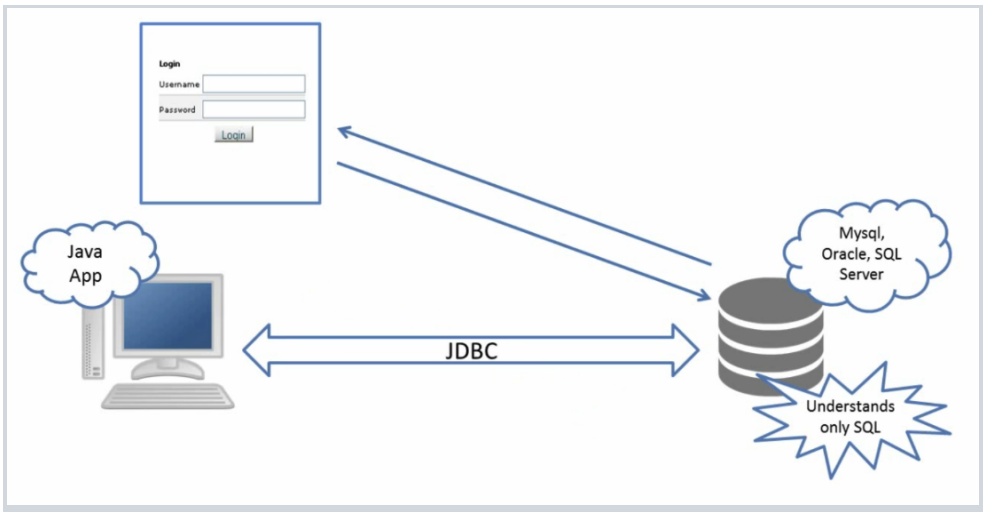
web.xml | struts.xml | register.jsp | welcome.jsp | RegisterAction.java | Product.java

```
22      <s:property value="age"/><br/>
23
24      <s:label  value="Email:"/>
25      <s:property value="email"/><br/>
26
27      <s:label  value="Address:"/>
28      <s:property value="address"/><br/>
29
30      <s:label  value="Favourite Color:"/>
31      <s:property value="selectedColor"/><br/>
32
33      <s:label  value="Hobbies:"/>
34      <s:property value="selectedHobbies"/>
35
36      <s:if test="%{subscription == true}">
37          <div>You are a subscriber</div>
38      </s:if>
39      <s:else>
40          <div>You are NOT a subscriber</div>
41      </s:else>
42  </body>
43  </html>
```





When un check



What is JDBC?

- JDBC (Java Database Connectivity) is an API which allows our Java applications to interact with an RDBMS(Relational Database Management System).
- It is independent of any particular database.

Why do we need a database driver?

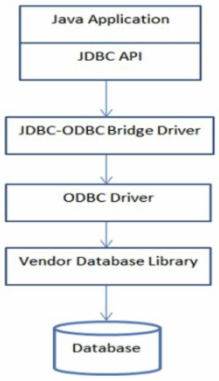
- We can't directly talk to the database server.
- Driver converts JDBC calls to database-specific calls.
- For every database, we need a driver.

Types of JDBC Drivers

- Type-1: JDBC-ODBC Bridge Driver
- Type-2: Native API Driver
- Type-3: Network Protocol Driver
- Type-4: Thin Driver

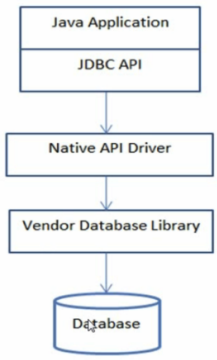
JDBC-ODBC Bridge Driver

- Converts JDBC calls to ODBC calls.



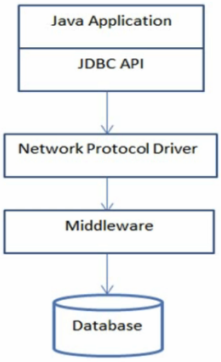
Native API Driver

- Converts JDBC calls to native calls of the database API.



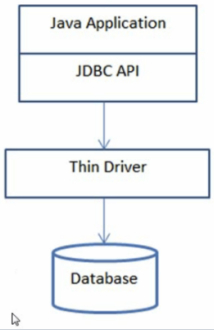
Network Protocol Driver

- Uses a middleware between the calling program and database.
- Middleware converts JDBC calls to vendor-specific database protocol.



Thin Driver

- Converts JDBC calls directly to vendor-specific database protocol.



Commonly used classes/interfaces

- DriverManager
- Connection
- Statement
- PreparedStatement
- CallableStatement
- ResultSet
- ResultSetMetaData
- DatabaseMetaData

Meta data meaning data about data.

Step 1 : Load and Register the driver

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Step 2 : Create connection

```
Connection con=DriverManager.getConnection("jdbc:oracle:thi
n:@localhost:1521:xe","system","password");
```

Step 3 : Create statement

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

Step 4 : Execute query

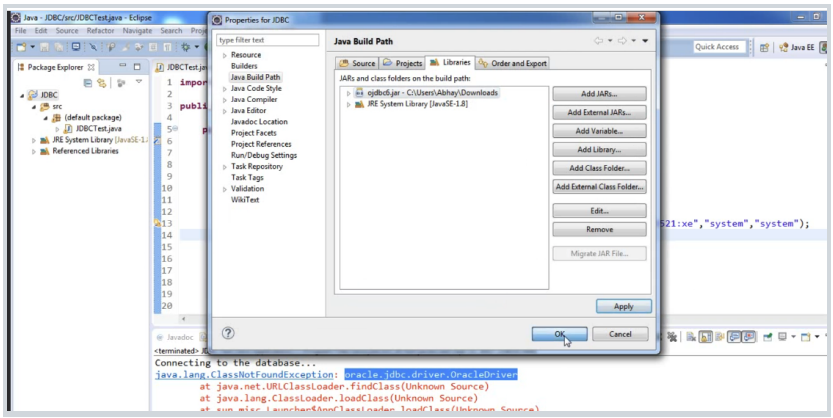
```
ResultSet rs=stmt.executeQuery("select * from emp");
```

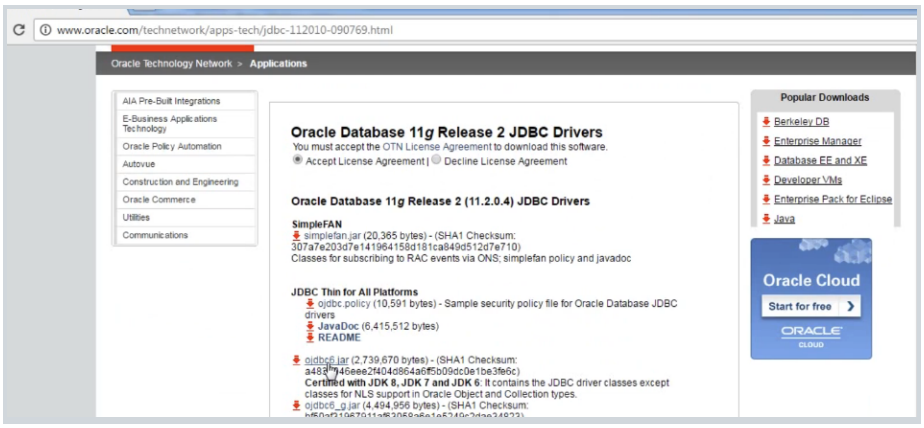
Step 5 : Close connection

```
con.close();
```

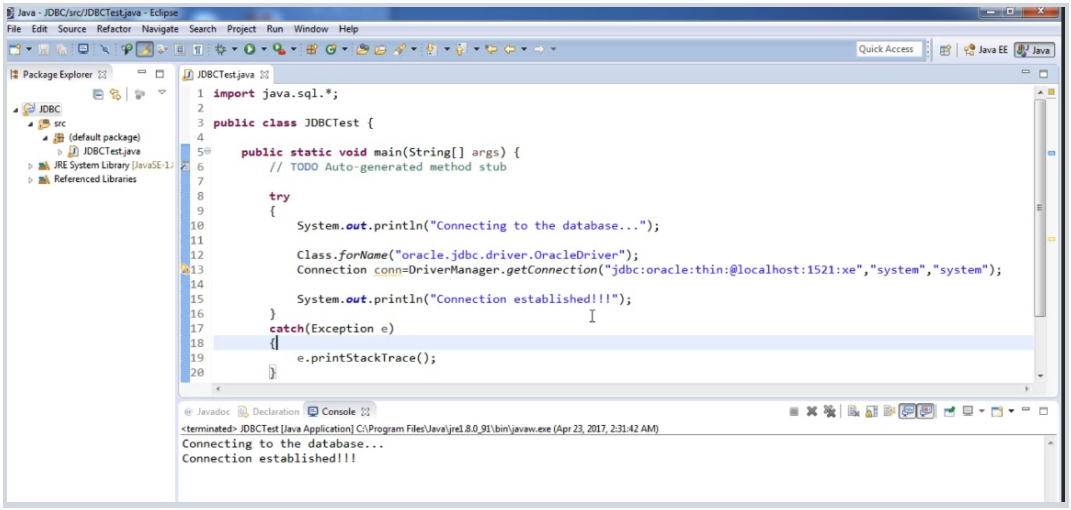
JDBC Connectivity Steps:-

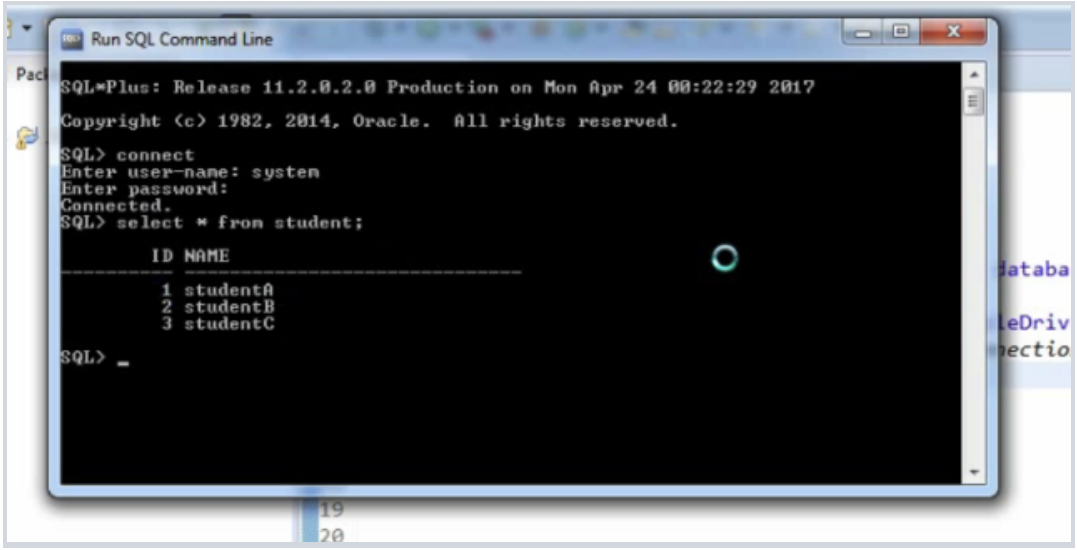
1. Load and Register the driver.
2. Create connection.
3. Create statement.
4. Execute query.
5. Close connection.





Jar location in internet.





```
JDBCTest.java

4
5    public static void main(String[] args) {
6        // TODO Auto-generated method stub
7
8        try
9        {
10           System.out.println("Connecting to the database...");
11
12           Class.forName("oracle.jdbc.driver.OracleDriver");
13           Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");
14
15           Statement st = conn.createStatement();
16           ResultSet rs = st.executeQuery("SELECT * FROM student");
17
18           while(rs.next())
19           {
20               System.out.println(rs.getInt("ID")+" --> "+rs.getString("NAME"));
21           }
22
23       }

Javadoc  Declaration  Console
<terminated> JDBCTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (Apr 24, 2017, 12:27:02 AM)
Connecting to the database...
1 --> studentA
2 --> studentB
3 --> studentC
```

Create Query

```
JDBCTest.java  CRUDTest.java

1  import java.sql.*;
2
3  public class CRUDTest {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          try
8          {
9
10             Class.forName("oracle.jdbc.driver.OracleDriver");
11             Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");
12
13             Statement st = conn.createStatement();
14             int n = st.executeUpdate("CREATE TABLE employee(emp_id int, emp_name varchar2(30), emp_salary int)");
15             System.out.println("Number of rows affected: "+n);
16
17             conn.close();
18
19         }
20         catch(Exception e)
21         {
22             e.printStackTrace();
23         }

Javadoc  Declaration  Console
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (Apr 30, 2017, 12:42:26 AM)
```

```
SQL> select * from employee;

no rows selected

SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMP_ID                                             NUMBER(38)
 EMP_NAME                                           VARCHAR2(30)
 EMP_SALARY                                         NUMBER(38)

SQL> _
```

Insert Query

```
JDBCTest.java  CRUDTest.java
1  import java.sql.*;
2
3  public class CRUDTest {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          try
8          {
9
10             Class.forName("oracle.jdbc.driver.OracleDriver");
11             Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");
12
13             Statement st = conn.createStatement();
14             int n = st.executeUpdate("INSERT INTO employee values(101,'employeeA',1000)");
15             System.out.println("Number of rows affected: "+n);
16
17             conn.close();
18
19         }
20         catch(Exception e)
21         {
22             e.printStackTrace();
23         }
Javadoc  Declaration  Console
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 2, 2017, 12:38:04 AM)
Number of rows affected: 1
```

```
SQL> select * from employee;

    EMP_ID EMP_NAME                       EMP_SALARY
---------- ------------------------------ ----------
       101 employeeA                            1000

SQL> _
```

Insert Query - only specific fields

```
JDBCTest.java  CRUDTest.java
1  import java.sql.*;
2
3  public class CRUDTest {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          try
8          {
9
10             Class.forName("oracle.jdbc.driver.OracleDriver");
11             Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");
12
13             Statement st = conn.createStatement();
14             int n = st.executeUpdate("INSERT INTO employee(emp_id,emp_name) values(104,'employeeD')");
15             System.out.println("Number of rows affected: "+n);
16
17             conn.close();
18
19         }
20         catch(Exception e)
21         {
22             e.printStackTrace();
23         }
Javadoc  Declaration  Console
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 2, 2017, 12:38:50 AM)
Number of rows affected: 1
```

104 employee has no salary.

```
103 employeeC                                7000

SQL> select * from employee ORDER BY emp_id;

    EMP_ID EMP_NAME                       EMP_SALARY
---------- ------------------------------ ----------
       101 employeeA                            1000
       102 employeeB                            3000
       103 employeeC                            7000
       104 employeeD
```

Update Query

```java
import java.sql.*;

public class CRUDTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");

            Statement st = conn.createStatement();
            int n = st.executeUpdate("UPDATE employee SET emp_salary = 11000 where emp_id = 104");
            System.out.println("Number of rows affected: "+n);

            conn.close();

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

```
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 2, 2017, 1:09:55 AM)
```

One more update statement

```java
import java.sql.*;

public class CRUDTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");

            Statement st = conn.createStatement();
            int n = st.executeUpdate("UPDATE employee SET emp_salary = 5000 where emp_salary > 1000");
            System.out.println("Number of rows affected: "+n);

            conn.close();

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

```
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 2, 2017, 1:15:05 AM)
Number of rows affected: 3
```

Delete Query

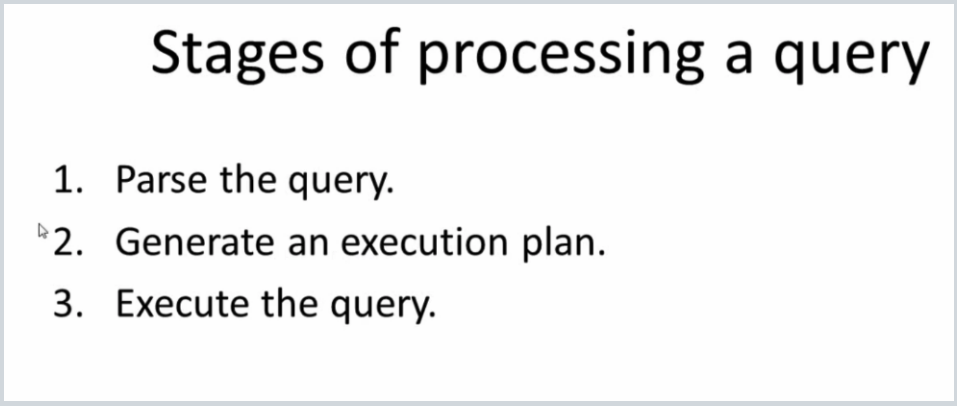
```java

public class CRUDTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");

            Statement st = conn.createStatement();
            int n = st.executeUpdate("DELETE FROM employee where emp_id = 104");
            System.out.println("Number of rows affected: "+n);

            conn.close();

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

```
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 2, 2017, 1:22:36 AM)
Number of rows affected: 1
```

One more delete

```
Statement st = conn.createStatement();
int n = st.executeUpdate("DELETE FROM employee");
System.out.println("Number of rows affected: "+n);
```

Statement Vs PreparedStatement



In case of a statement, object with hardcoded values it will compile db again and again for every insert statement. It will load/affect memory of the oracle database server.

Incase of prepare statement, compile will happens only once.

```java
import java.sql.*;

public class CRUDTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");

            PreparedStatement ps = conn.prepareStatement("INSERT INTO employee values(?,?,?)");
            ps.setInt(1, 101);
            ps.setString(2, "employeeA");
            ps.setInt(3, 1000);
            int n = ps.executeUpdate();
            System.out.println("Number of rows affected: "+n);

            conn.close();

        }
        catch(Exception e)
```

```
<terminated> CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 3, 2017, 1:00:11 AM)
Number of rows affected: 1
```

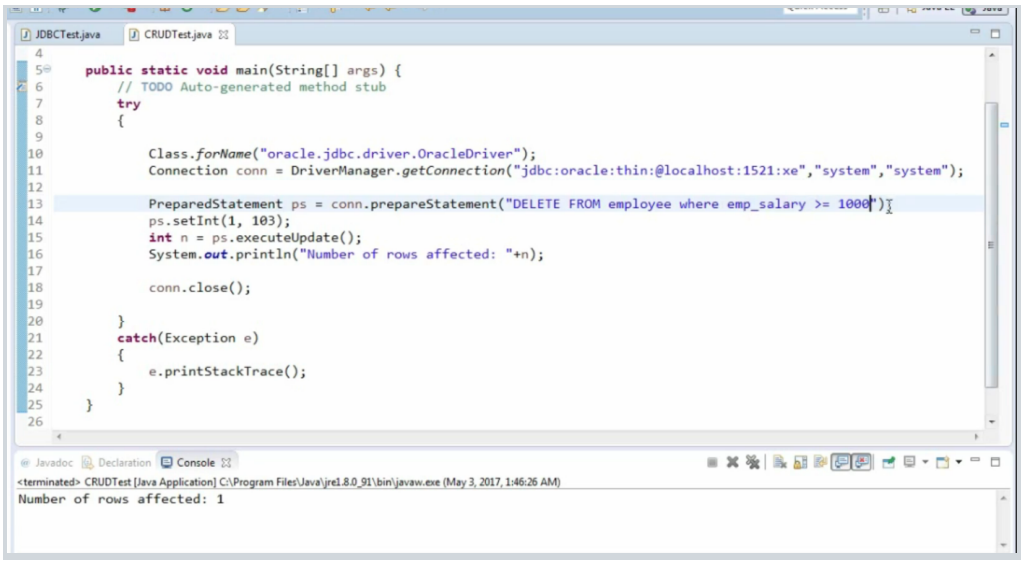
```
Connected.
SQL> select * from employee order by emp_id;

no rows selected

SQL> select * from employee order by emp_id;

    EMP_ID EMP_NAME                       EMP_SALARY
---------- ------------------------------ ----------
       101 employeeA                            1000
```

Update with PrepareStatement.

```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");

            PreparedStatement ps = conn.prepareStatement("UPDATE employee SET emp_salary = ? where emp_id = ?");
            ps.setInt(1, 20000);
            ps.setInt(2, 102);
            int n = ps.executeUpdate();
            System.out.println("Number of rows affected: "+n);

            conn.close();

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

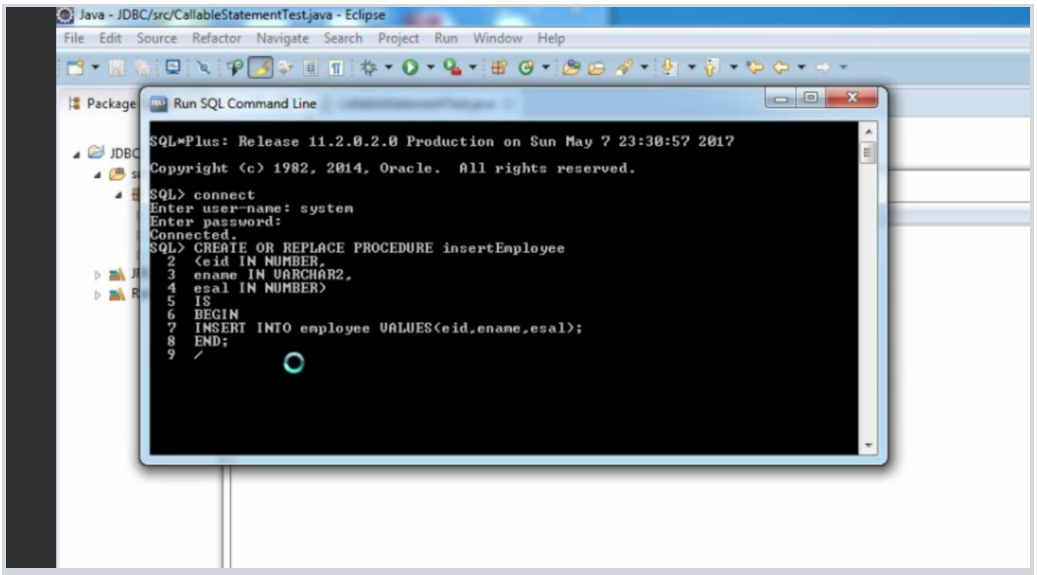
```
CRUDTest [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (May 3, 2017, 1:26:30 AM)
```

```
    EMP_ID EMP_NAME                       EMP_SALARY
---------- ------------------------------ ----------
       101 employeeA                            1000
       102 employeeB                           20000
       103 employeeC                           10000

SQL> _
```



Callable Statement:

Create store procedure in oracle db as shown below.



And callable statement in program.

```java
import java.sql.*;

public class CallableStatementTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");

            CallableStatement stmt=conn.prepareCall("{call insertEmployee(?,?,?)}");
            stmt.setInt(1,101);
            stmt.setString(2,"employeeA");
            stmt.setInt(3,9000);
            int n = stmt.executeUpdate();
            System.out.println("Number of rows affected: "+n);

            conn.close();

        }
```

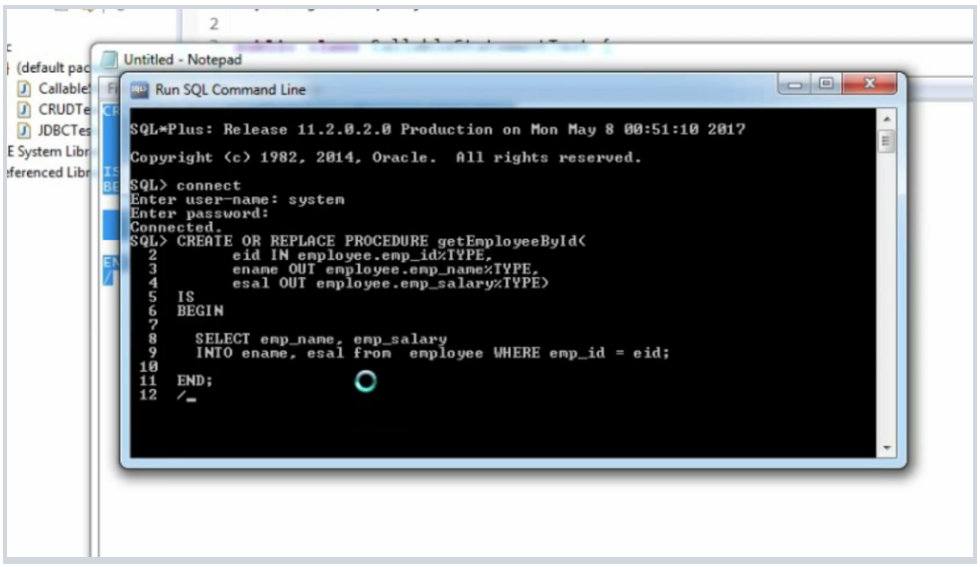
```
Number of rows affected: 1
```

```
no rows selected
SQL> select * from employee order by emp_id;

    EMP_ID EMP_NAME                       EMP_SALARY
---------- ------------------------------ ----------
       101 employeeA                            9000
SQL> _
```

```
int n = stmt.executeUpdate();
System.out.println("Number of rows
conn.close();
}
```

```
Javadoc  Declaration  Console
<terminated> CallableStatementTest [Java Application] C:\Program Files\Java\jre1.8.0_
Number of rows affected: 1
```

One more example with select statement



Above we have one input parameter and 2 output parameters in our store procedure and its program below.