18

Alliance College of Engineering and Design

Department of Computer Science and Engineering

Sem: VI Year: III Batch: 2021 - 2025

CSL607 – Advanced Java Development Lab Manual

LIST OF EXPERIMENTS

1. To implement program with Commandline arguments, 2. Inheritance 3. To implement program with Polymorphism 4. Interface 5. Abstract classes 6. Upcasting and downcasting
1. To implement exception handling using java 2. To implement XML concepts, DTD and XSL concepts 3. Develop a project using Objects, Methods and Classes with its databases. 4. Program with AWT and Databases 5. Program with Swing and Databases
1. Write a JDBC specification to establish the connection with the Mysql 2. Write a JDBC specification to insert a record into the Employee database. 3. Write a JDBC specification to establish the connection with the SQL server 4. Write a JDBC specification to implement batch-updates. 5. Write a program to implement JDBC transaction
1. Construct a Servlet program to print a welcome message and its number of user visiting page 2. Construct a Servlet program to validate the user login 3. Construct a servlet to handle sessions using URL-rewriting 4. Implement a program with session management using servlet 5. Develop a program in cookie and identify cookie information using servlets 6. Write a program to display CRUD application using Servlet with Databases
1. Implement a program to display date and time in JSP 2. Write a JSP program to include and forward web pages 3. Write a JSP program to display user details from HTML web page 4. Write a program to implement implicit objects in JSP 5. Write a program to display CRUD application using JSP with Databases 6. Create a program to display the values of User information using Hibernate.

7. Create an application in Spring to make injecting codes in string based information
8. Create an application with restful Web Services Project with Spring Boot

1. To implement program with Commandline arguments,

```
public class cmd1 {

        public static void main(String[] args)
        {
 int jm,cm,rm,regno;  int tot;   float
avg;
                String name,remarks;
regno=Integer.parseInt(args[0]);
                name=args[1];
                jm=Integer.parseInt(args[2]);
cm=Integer.parseInt(args[3]);
                rm=Integer.parseInt(args[4]);
                tot=jm+cm+rm;
                avg=(float) (tot/3.0);
if(jm>=50 && cm>=50 && rm>=50)
remarks="Pass in all Subjects";             else
remarks="Fail in subjects";
                System.out.println("Regno is:"+regno);
                System.out.println("Name is:"+name);
                System.out.println("java Mark is:"+jm);
  .out.println("CSystem Mark is:"+cm);
  .out.println("RSystemmark is:"+rm);
                System.out.println("Total is:"+tot);
                System.out.println("Average is:"+avg);
                System.out.println("Remarks is:"+remarks);
        }
}
```

```
Regno is:101
Name  is:CSE
java Mark is:75 C
Mark is:85
R mark is:95
Total is:255
Average is:85.0
Remarks is:Pass in all Subjects
```

2. Inheritance

```
class Box
{
    double width,height,depth;
    Box()
    {height=10; width=10; depth=20;          }
```

```
    Box(double l)
    {
            width=height=depth=l;
    }
    Box(double w,double h,double d)
    {
    width=w;
    height=h;
    depth=d;        }
    Box(Box ob)
    {               width=ob.width;
    height=ob.height;
            depth=ob.depth;
    }
    double volume()
    {       System.out.print("The volume of Box is:");
    return (width*height*depth);
    }
}
class BoxWeight extends Box
{
    double weight;
    BoxWeight()
    {super();       }
    BoxWeight(double l)
    {super(l);
            weight=l;
    }
    BoxWeight(double w,double h,double d,double w1)
    {               super(w,h,d);
            weight=w1;      }
    BoxWeight(BoxWeight ob)
    {               super(ob);
    weight=ob.weight;
    }
}

class BoxShiftment extends BoxWeight
{ double cost;
    BoxShiftment()
    {super();       }
    BoxShiftment(double l)
    {super(l);
            cost=l; }
    BoxShiftment(double w,double h,double d,double w1,double c)
    {               super(w,h,d,w1);
            cost=c; }
    BoxShiftment(BoxShiftment ob)
    {               super(ob);
    cost=ob.cost;   }} class BoxEx
{   public static void main(String a[])
```

```
{       BoxShiftment bs1=new BoxShiftment();
        System.out.println(bs1.volume());
        BoxWeight bw1=new BoxWeight(10);
        System.out.println(bw1.volume());
        //bs1=bw1;
        bw1=bs1;
        System.out.println(bw1.volume());
}}
```

3. To implement program with Polymorphism

```
import java.util.Scanner;

class Bank {
    //int roi;
    int getroi()
    {               return 2;
    }
}
class SBI extends Bank
{   int getroi()
    {
    return 4;
    }
}
class HDFC extends Bank
{   int getroi()
    {               return 5;
    }
} class poly
{
    public static void main(String[] args)
    {
            double p,n,r,r1,r2,si;
    double ci;
            Scanner s=new Scanner(System.in);
            System.out.println("Enter p,n,r");
            p=s.nextInt();
    n=s.nextInt();  Bank b=new Bank();
    r=b.getroi();   si=(p*n*r)/100;
    double amt=p+si;
ci=p*Math.pow((1+r/100),n);
            System.out.println("Amt is:"+amt+"\n"+"CI is:"+ci);
                    SBI sb=new SBI();
            r1=sb.getroi();
            si=(p*n*r1)/100;
            ci=p*Math.pow((1+r1/100),n);
            System.out.println("SI is:"+si+"\n"+"CI is:"+ci);
                    HDFC h=new HDFC();
```

```
            r2=h.getroi();
    si=(p*n*r2)/100;
    ci=p*Math.pow((1+r2/100),n);
            System.out.println("SI is:"+si+"\n"+"CI is:"+ci);
            }
}
```

4. Interface public

```
interface int1 {    int
regno=101;
    void input();
    String compute();
    void display(String s1);

}
interface int2 extends int1
{ void
display();
}


import java.util.Scanner;


public class stuint implements int2
{
    int jm,cm,pm,tot,avg;
    String remarks;
    Scanner s;
    public void
input()
    {
            s=new Scanner(System.in);
            System.out.println("Enter the Java mark, Cm, PM");
            jm=s.nextInt();
    cm=s.nextInt();
    pm=s.nextInt();
    }
    public String compute()
    {       tot=jm+cm+pm;
    avg=tot/3;
            if(avg>=75)
                    remarks="Passed with Distinction";
            else if(avg>=60 && avg<75)
            remarks="Passed with I class";
            else if(avg>=50 && avg<60)
                    remarks="Pass";
            else remarks="Fail";
                    return remarks;
```

```
        }
        public void display(String s)
        {       System.out.println("Reg no is:"+regno);
        System.out.println("Total is:"+tot);
                System.out.println("Avg is:"+avg);
                System.out.println("Remarks"+s);
        }
        public static void main(String[] args)
        {       int1 i1=new stuint();
        i1.input();
                String s1=i1.compute();
        i1.display(s1); }
    }
```

5. Abstract classes

```
import java.util.Scanner; public
abstract class theory
{
        int s1,s2;
void getdata1()
        {       Scanner s=new Scanner(System.in);
System.out.println("Enter the marks1:");                  s1=s.nextInt();
                System.out.println("Enter the marks2:");
s2=s.nextInt();
        }
        abstract String display();
}

import java.util.Scanner; public
class practical extends theory
{String rem;
        void getdata2()
        {       Scanner s=new Scanner(System.in);
System.out.println("Enter the marks1:");                  s1=s.nextInt();
                System.out.println("Enter the marks2:");
s2=s.nextInt();
        }
        public String display()
        {       if(s1>=50 && s2>=50)
        rem="pass";
                else rem="fail";

                return rem;
        }
}

public class result extends practical
{       String s1,s2;
        public static void main(String[] args)
```

```
        {
                result r=new result();
r.getdata1();
                r.s1=r.display();
                r.getdata2();
                r.s2=r.display();
                if((r.s1.equals("pass")) && (r.s2.equals("pass")))
        {       System.out.println("Overall Pass in Subjects");
                }
                else
                {       System.out.println("Overall Fail in Subjects");
                }
        }
}
```

6. Upcasting and downcasting

```
 class Alliance
{   }
class ACED extends Alliance {
    static void method(Alliance
a)
    {       if(a instanceof ACED)
            {       ACED d=(ACED)a;
                    System.out.println("Student of ACED");
            }
    }}
class ASOB extends Alliance {
    static void method(Alliance
a)
    {       if(a instanceof ASOB)
            {       ASOB d=(ASOB)a;
                    System.out.println("Student of ASOB");
    }       }
} class downcast { public static void
main(String[] args)
    {       Alliance a1=new ACED();
    Alliance a2=new ASOB();
            ACED.method(a1);
            ASOB.method(a2);
    }
}
```

7. To implement exception handling using java(Own Exception)

```
 import java.util.Scanner;

class fund extends Exception
```

```
{       Scanner s;
 int bal,amt;  String m;
fund(String s)
        {       bal=500;
                m=s;
                System.out.println(m);
        }
        void compute()throws fund
        {
try
                {
                        System.out.println("Enter Amount");
        s=new Scanner(System.in);
amt=s.nextInt();
                        if(bal<amt)
                                throw new fund("Insufficient");
                        else
                                bal=bal-amt;
                                System.out.println(bal);
                }

                catch(fund e)
                {
                        System.out.println(e.getMessage());
                }
                finally{

                }
        }
} class ex1
{

        public static void main(String[] args) {
                try
                {

                fund f=new fund("User Define Exception:");
                f.compute();
        }catch(Exception e)
                {
                System.out.println(e.getMessage());
                }
                }
        }
```

8. To implement XML concepts, DTD and XSL concepts

A. Write a Xml program to display Automobile details using DTD structure.

Cars.dtd

```
<?xml version = "1.0"  encoding = "utf-8"?>
```

<!ELEMENT car_catalog (car+)>

```
<!ELEMENT car (year, model, color, engine, number_of_doors, transmission_type, accessories)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT engine (number_of_cylinders, fuel_system)>
<!ELEMENT number_of_cylinders (#PCDATA)>
<!ELEMENT fuel_system (#PCDATA)>
<!ELEMENT number_of_doors (#PCDATA)>
<!ELEMENT transmission_type (#PCDATA)>
<!ELEMENT accessories (#PCDATA)>
<!ATTLIST accessories radio CDATA #REQUIRED>
<!ATTLIST accessories air_conditioning CDATA #REQUIRED>
<!ATTLIST accessories power_windows CDATA #REQUIRED>
<!ATTLIST accessories power_steering CDATA #REQUIRED>
<!ATTLIST accessories power_brakes CDATA #REQUIRED>
```

Cars.xml

```
<?xml version = "1.0"  encoding = "utf-8"?>
<!DOCTYPE car_catalog SYSTEM "Cars.dtd">
  <car_catalog>
    <car>
      <year> 1997 </year>
      <model> MARUTHI </model>
      <color> Light blue </color>
      <engine>
        <number_of_cylinders> 8 cylinder
        </number_of_cylinders>
        <fuel_system> multi-port fuel injected </fuel_system>
      </engine>
      <number_of_doors> 4 door </number_of_doors>
      <transmission_type> 4 speed automatic
      </transmission_type>
      <accessories radio = "yes" air_conditioning = "yes"
                power_windows = "yes"
power_steering = "yes"                    power_brakes
= "yes" />
    </car>
    <car>
      <year> 1965 </year>
      <model> FORD </model>
      <color> White </color>
      <engine>
        <number_of_cylinders> 8 cylinder
        </number_of_cylinders>
        <fuel_system> 4BBL carburetor </fuel_system>
      </engine>
      <number_of_doors> 2 door </number_of_doors>
      <transmission_type> 3 speed manual </transmission_type>
<accessories radio = "yes" air_conditioning = "no"
power_windows = "no"  power_steering = "yes"
                power_brakes = "yes" />
    </car>
```

```
  <car>
    <year> 1985 </year>
    <model> HYUNDAI </model>
    <color> Blue </color>
    <engine>
      <number_of_cylinders> 4 cylinder
      </number_of_cylinders>
      <fuel_system> fuel injected </fuel_system>
    </engine>
    <number_of_doors> 4 door </number_of_doors>
    <transmission_type> 4 speed manual </transmission_type>
<accessories radio = "yes" air_conditioning = "yes"
power_windows = "no"  power_steering = "yes"
                power_brakes = "yes" />
  </car>
 </car_catalog>
```

B. Write a Xml program to display student details using XSL structure.

Stud.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="stud.xsl"?>
 <student>
  <s>
   <name> Abraham </name>
   <branch> CSE</branch>
   <age>18</age>
   <city> Mumbai </city>
  </s>
  <s>
   <name> Issac </name>
   <branch> IT</branch>
   <age> 20</age>
   <city> Chennai </city>
  </s>
  <s>
   <name> Moses </name>
   <branch> CSE</branch>
   <age> 23</age>
   <city> Bengaluru</city>
  </s>
  <s>
   <name> Joshua </name>
<branch> CSE</branch>
   <age> 17</age>
   <city> Delhi</city>
  </s>
  <s>
   <name> Sumathi </name>
   <branch> IT</branch>
   <age> 25</age>
```

```
  <city> Indore</city>
 </s>
</student>
```

Stud.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
  <h1 align="center">Student Basic Details</h1>
   <table border="3" align="center" >
   <tr>
    <th>Name</th>
    <th>Branch</th>
    <th>Age</th>
    <th>City</th>
   </tr>
    <xsl:for-each select="student/s">
   <tr>
    <td><xsl:value-of select="name"/></td>
    <td><xsl:value-of select="branch"/></td>
    <td><xsl:value-of select="age"/></td>
    <td><xsl:value-of select="city"/></td>
   </tr>
    </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Output:

Student Basic Details

Name	Branch	Age	City
Abraham	CSE	18	Mumbai
Issac	IT	20	Chennai
Moses	CSE	23	Bengaluru
Joshua	CSE	17	Delhi
Sumathi	IT	25	Indore

C. Write a Xml program to display student details using xsd schema structure.

Order.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="order">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="orderperson" type="xs:string"/>
```

```
    <xs:element name="ordername">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="address" type="xs:string"/>
          <xs:element name="city" type="xs:string"/>
          <xs:element name="country" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="item" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="title" type="xs:string"/>
          <xs:element name="note" type="xs:string" minOccurs="0"/>
          <xs:element name="quantity" type="xs:positiveInteger"/>
          <xs:element name="price" type="xs:decimal"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:sequence>
   <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Order.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderid="T5987" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Order.xsd">
  <orderperson>John Smith</orderperson>
  <ordername>
    <name>Jeremiah</name>
    <address>Langgt 23</address>
    <city>Mumbai</city>
    <country>India</country>
  </ordername>
  <item>
    <title>Service Arch</title>
    <note>Special Edition</note>
    <quantity>2</quantity>
    <price>100.90</price>
  </item>
  <item>
    <title>Advanced Java</title>
    <quantity>3</quantity>
    <price>90.55</price>
  </item>
</order>
```

9. Develop a project using Objects, Methods and Classes with its databases.

```
 import java.sql.*; class
Dbmd{
public static void main(String args[]){ try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/paul","root","root");
DatabaseMetaData dbmd=con.getMetaData();

System.out.println("Driver Name: "+dbmd.getDriverName());
System.out.println("Driver Version: "+dbmd.getDriverVersion());
System.out.println("UserName: "+dbmd.getUserName());
System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

10. Write a JDBC specification to insert a record into the Employee database.

```
import java.sql.*;
//maysql.connector.jar -> jre/lib/ext file public
class sql1 {

                public static void main(String args[]){
                try{
                Class.forName("com.mysql.jdbc.Driver");
                Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/paul","root","root");

//Class.forName("oracle.jdbc.driver.OracleDriver");
//Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
// String database="student.mdb";//Here database exists in the current directory
//  String url="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};
//DBQ=" + database + ";DriverID=22;READONLY=true";
 //  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 //  Connection c=DriverManager.getConnection("jdbc:odbc:mydsn");

                Statement stmt=con.createStatement();
stmt.executeUpdate("drop table emp5;");
  stmt.executeUpdate("create table emp5(eno int(5),ename varchar(20),esal int(5), edes varchar(10))");

                stmt.executeUpdate("insert into emp5
values('101','aaa','5000','manager'),('102','bbb','6000','officer'),('104','ddd',8000,'accountant');");

                stmt.executeUpdate("delete from emp5 where eno=101;");

                stmt.executeUpdate("update emp5 set esal=10700 where eno=104;");
```

```
ResultSet rs=stmt.executeQuery("select * from emp5");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getInt(3)+"  "+rs.getString(4));
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

11. Write a JDBC specification to implement batch-updates.

```
import java.sql.*;
  class DBbatup
  {
        public static void main(String args[])
        {
            Connection con=null;
Statement stmt=null;
              try
              {
                    Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/my123","root","root"); stmt =
con.createStatement();
stmt.addBatch("update people set firstname='nasan',lastname='ellil' where id=1");
stmt.addBatch("update people set firstname='Eric',lastname='john' where id=2"); stmt.addBatch("update
people set firstname='May',lastname='julian' where id=3");
        System.out.println("records updated");
        stmt.executeBatch();
        //String sql = "update people set firstname=? , lastname=? where id=?";
              PreparedStatement ps = null;
ps = con.prepareStatement("update people set firstname=? , lastname=? where id=?");
  ps.setString(1, "Gary");    ps.setString(2,
"Larson");          ps.setLong       (3,    2);
ps.addBatch();
       ps.setString(1, "Stan");          ps.setString(2,
"Lee");    ps.setLong    (3,    3);    ps.addBatch();
ps.executeBatch();
        System.out.println("Records Updated using Prepared Statement");
    ResultSet rs=stmt.executeQuery("select * from people");
        while(rs.next())
        {
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
        }
        }
              catch(Exception e)
              {
                    System.out.println(e+" File not found");
                    System.exit(0);
          }
              finally
              {
              }
}    }
```

records updated
Records Updated using Prepared Statement
1 nasan ellil
2 Gary Larson
3 Stan Lee

12. Write a JDBC transaction specification to establish the connection with the Mysql

```
import java.sql.*;
  class DBbatup
  {
          public static void main(String args[])
          {
                  Connection con;
        try
                  { Class.forName("com.mysql.jdbc.Driver");
                  con=DriverManager.getConnection("jdbc:mysql://localhost:3306/my123","root","root");
                        Statement stmt=null;
                        try{
                                con.setAutoCommit(false);
        stmt = con.createStatement(); stmt.executeUpdate("update stud1 set
lname='John' where rollno=101");
                        System.out.println("Row updated successfully");
                        }
        finally {
                        if(stmt != null) {
          stmt.close();
                        }
                    }

    CallableStatement cs;       Statement s2;
                        try
                        { cs=con.prepareCall("{call proc1(?,?,?)}");
                          cs.setInt(1,102);
                                cs.setString(2,"Gary");
                          cs.setString(3,"Larson");
            cs.addBatch();
cs.setInt(1,104);
cs.setString(2,"Stan");
cs.setString(3,"Lee");                          cs.addBatch();
                    cs.executeBatch();
        System.out.println("Two Records Updated using Callable Statement");
s2=con.createStatement();
                         ResultSet rs=s2.executeQuery("select * from stud1");
                      while(rs.next())
                      {
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+" "+rs.getString(3));
                      }
                          }
```

```
                    finally
        {       }
                     con.commit();
                  } catch(Exception e)
                {       System.out.println("Error"+e.getMessage());
                }
                finally
                  {       }
  } }
```

Row updated successfully
Two Records Updated using Callable Statement
101 AAA John
103 bbb ccc
102 Gary Larson
104 Stan Lee

13. Construct a Servlet program to print a welcome message and its number of user visiting page

```
 import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.ServletException;
import
javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletReque
st; import
javax.servlet.http.HttpServletRespo
nse; import
javax.servlet.http.HttpSession;

public class ServSession extends HttpServlet {

   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
   {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
     HttpSession session=request.getSession();
     String Heading;
try        {
       Integer cnt=(Integer)session.getAttribute("cnt");
out.println("<html>");          out.println("<head>");
         out.println("<title>Servlet ServSession</title>");
out.println("</head>");          out.println("<body><br/>");
         if(cnt==null)
         {
            cnt=new Integer(1);
           Heading="Welcome you for the first time";
```

```
        }
else
        {
           Heading="Welcome Once again";
           cnt=new Integer(cnt.intValue()+1);
        }
        session.setAttribute("cnt", cnt);
 out.println("<h1>Welcome to session Program: You are Visiting this site as :"+cnt +
" Visitor </h1>");
out.println("<h1>You are Working Servlet Program at " + request.getContextPath () +" <br/><br/>"+Heading);
 out.println("</h1> <h2>Your Hidden Session ID is:"+session.getId()+"  :ID</h2>");
out.println("</body>");
        out.println("</html>");

      } finally
      {
         out.close();
      }
   }
}
```

14. Construct a Servlet program to validate the user login

```
 <!DOCTYPE html>
<html><head><meta charset="ISO-8859-1">
<title>Servlet Program for Login</title></head>
<body>
<form name="frm" method="get" action="Ex2">
<h1 style="font-size:40px; font-variants:small-caps; font-weight:bolder">Login Registration Page </h1>
User name is: <input type=text name=username><br/>
Password is: <input type=password name=pass><br/>
<input type= submit value=Enter>
</form></body></html>

import java.io.*;
import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
@WebServlet("/Ex2")
public class Ex2 extends HttpServlet
{
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException
        {
                PrintWriter out=response.getWriter();
                String user;
String pwd;
```

```
                try{
                response.setContentType("text/html");
user=request.getParameter("username");
pwd=request.getParameter("pass");                out.println("<html><body><h1>");
                out.println("Welcome to Servlet Servlet Program <br/>");
                out.println("Username is: "+user);
out.println("<br/>Password is: "+pwd);
                out.println("</h1></body></html>");
                out.println(".....Successful Execution. .....");
                }
                catch(Exception e)
                {
                        out.println(e.getMessage());
                }
}}
```

Login Registration Page

User name is:
Password is:
Enter

Welcome to Servlet Servlet Program
Username is: Alliance
Password is: CSE/IT

.....Successful Execution.....

15. Construct a servlet to handle sessions using URL- rewriting.

```
<!DOCTYPE html>
<html><head><meta charset="ISO-8859-1">
<title>URL - reWriting</title></head>
<body>
<form action="Ex4" method="get">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="submit"/>
</form>  </body>  </html>
```

```
import java.io.*; import javax.servlet.*; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
@WebServlet("/Ex4")
public class Ex4 extends HttpServlet {
        private static final long serialVersionUID = 1L;
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            {
              try {
response.setContentType("text/html");
PrintWriter out = response.getWriter();

                String n = request.getParameter("userName");

                out.print("Welcome  " + n);

               out.print("<a href='Ex41?uname=" + n +  "'>  visit </a>");

                out.close();
      }
              catch (Exception e) {
System.out.println(e);
  }   }    }
```

```
import java.io.*; import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
@WebServlet("/Ex41")
public class Ex41 extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
            {
              try {

                response.setContentType("text/html");
PrintWriter out = response.getWriter();            String n =
request.getParameter("uname");
out.print("Good.... This is " + n+"University");
out.close();            }
              catch (Exception e) {
                System.out.println(e.getMessage());
              }
            }
}
```

16. Implement a program to display date and time in JSP

```
<%@ page import = "java.io.*,java.util.*, javax.servlet.*" %>
<html>
  <head>
    <title>Display Current Date & Time</title>
  </head>
  <body>
    <center>
      <h1>Display Current Date & Time</h1>
    </center>
    <%
      Date date = new Date();
      out.print( "<h2 align = \"center\">" +date.toString()+"</h2>");
    %>
  </body>
</html>
```

17. Write a JSP program to include and forward web pages

index.jsp

```
<html>
<head>
<title>JSP forward action tag example</title>
</head>
<body>
<p align="center">My main JSP page</p>
<jsp:forward page="display.jsp" />
</body>
</html>
```

display.jsp

```
<html>
<head>
<title>Display Page</title>
</head>
<body>
Hello this is a display.jsp Page
<jsp:include page="printdate.jsp" />
<h2>end section of index page</h2>
</body>
</html>
```

printdate.jsp

```
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
```

18. Write a program to display CRUD application using JSP with Databases

index.jsp

```
<!DOCTYPE
html>
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP CRUD Example</title>
</head>
<body>
<h1>JSP CRUD Example</h1>
<a href="adduserform.jsp">Add User</a>
<a href="viewusers.jsp">View Users</a>
</body>
</html>
```

adduserform.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Add User Form</title>
</head>
<body>
<jsp:include page="userform.html"></jsp:include>
</body>
</html>
```

userform.html

```
<a href="viewusers.jsp">View All Records</a><br/>
<h1>Add New User</h1>
<form action="adduser.jsp" method="post">
<table>
<tr><td>Name:</td><td><input type="text" name="name"/></td></tr> <tr><td>Password:</td><td>
<input type="password" name="password"/></td></tr>
<tr><td>Email:</td><td><input type="email" name="email"/></td></tr>
<tr><td>Sex:</td><td>
<input type="radio" name="sex" value="male"/>Male
<input type="radio" name="sex" value="female"/>Female </td></tr>
<tr><td>Country:</td><td>
<select name="country" style="width:155px">
<option>India</option>
<option>Pakistan</option>
<option>Afghanistan</option>
<option>Berma</option>
<option>Other</option>
</select>
</td></tr>
<tr><td colspan="2"><input type="submit" value="Add User"/></td></tr>
</table>
</form>
```

adduser.jsp

```
<%@page import="com.javatpoint.dao.UserDao"%>
<jsp:useBean id="u" class="com.javatpoint.bean.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>
<%
int i=UserDao.save(u);
if(i>0){
response.sendRedirect("adduser-success.jsp");
}else{
response.sendRedirect("adduser-error.jsp");
}
%>
```

User.java

```
package com.javatpoint.bean;
public class User {  private
int id;
private String name,password,email,sex,country;
//generate getters and setters
}
```

UserDao.java package

```
com.javatpoint.dao; import
java.sql.*; import
java.util.ArrayList; import
java.util.List; import
com.javatpoint.bean.User;
public class UserDao {
public static Connection getConnection(){
   Connection con=null;
   try{
      Class.forName("com.mysql.jdbc.Driver");
      con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","","");
   }catch(Exception e){System.out.println(e);}
return con;
}
public static int save(User u){
   int status=0;
try{
      Connection con=getConnection();
      PreparedStatement ps=con.prepareStatement(
"insert into register(name,password,email,sex,country) values(?,?,?,?,?)");
ps.setString(1,u.getName());          ps.setString(2,u.getPassword());
ps.setString(3,u.getEmail());          ps.setString(4,u.getSex());
ps.setString(5,u.getCountry());          status=ps.executeUpdate();
   }catch(Exception e){System.out.println(e);}
return status;
}
public static int update(User u){
```

```
   int status=0;
try{
      Connection con=getConnection();
      PreparedStatement ps=con.prepareStatement(
"update register set name=?,password=?,email=?,sex=?,country=? where id=?");
ps.setString(1,u.getName());          ps.setString(2,u.getPassword());
ps.setString(3,u.getEmail());          ps.setString(4,u.getSex());
ps.setString(5,u.getCountry());          ps.setInt(6,u.getId());
status=ps.executeUpdate();
   }catch(Exception e){System.out.println(e);}
return status;
}
public static int delete(User u){
   int status=0;
try{
      Connection con=getConnection();
      PreparedStatement ps=con.prepareStatement("delete from register where id=?");
ps.setInt(1,u.getId());          status=ps.executeUpdate();
   }catch(Exception e){System.out.println(e);}

   return status;
}
public static List<User> getAllRecords(){
List<User> list=new ArrayList<User>();

   try{
      Connection con=getConnection();
      PreparedStatement ps=con.prepareStatement("select * from register");
      ResultSet rs=ps.executeQuery();
      while(rs.next()){
User u=new User();
         u.setId(rs.getInt("id"));
         u.setName(rs.getString("name"));
         u.setPassword(rs.getString("password"));
         u.setEmail(rs.getString("email"));
         u.setSex(rs.getString("sex"));
         u.setCountry(rs.getString("country"));
list.add(u);
      }
   }catch(Exception e){System.out.println(e);}
return list;
}
public static User getRecordById(int id){
   User u=null;
try{
      Connection con=getConnection();
      PreparedStatement ps=con.prepareStatement("select * from register where id=?");
ps.setInt(1,id);
      ResultSet rs=ps.executeQuery();
      while(rs.next()){
u=new User();
```

```
            u.setId(rs.getInt("id"));
            u.setName(rs.getString("name"));
            u.setPassword(rs.getString("password"));
            u.setEmail(rs.getString("email"));
            u.setSex(rs.getString("sex"));
            u.setCountry(rs.getString("country"));
        }
    }catch(Exception e){System.out.println(e);}
return u;
}
}
```

adduser-success.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Add User Success</title>
</head>
<body>
<p>Record successfully saved!</p>
<jsp:include page="userform.html"></jsp:include>
</body>
</html>
```

adduser-error.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Add User Error</title>
</head>
<body>
<p>Sorry, an error occurred!</p>
<jsp:include page="userform.html"></jsp:include>
</body>
</html>
```

viewusers.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>View Users</title>
</head>
<body>
<%@page import="com.javatpoint.dao.UserDao,com.javatpoint.bean.*,java.util.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<h1>Users List</h1>
<%
```

```
List<User> list=UserDao.getAllRecords();
request.setAttribute("list",list);
%>
<table border="1" width="90%">
<tr><th>Id</th><th>Name</th><th>Password</th><th>Email</th>
<th>Sex</th><th>Country</th><th>Edit</th><th>Delete</th></tr>
<c:forEach items="${list}" var="u">
<tr><td>${u.getId()}</td><td>${u.getName()}</td><td>${u.getPassword()}</td>
<td>${u.getEmail()}</td><td>${u.getSex()}</td><td>${u.getCountry()}</td>
<td><a href="editform.jsp?id=${u.getId()}">Edit</a></td>
<td><a href="deleteuser.jsp?id=${u.getId()}">Delete</a></td></tr>
</c:forEach>
</table>
<br/><a href="adduserform.jsp">Add New User</a>
</body>
</html>

editform.jsp <!DOCTYPE
html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Edit Form</title>
</head>
<body>
<%@page import="com.javatpoint.dao.UserDao,com.javatpoint.bean.User"%> <%
String id=request.getParameter("id");
User u=UserDao.getRecordById(Integer.parseInt(id));
%>
<h1>Edit Form</h1>
<form action="edituser.jsp" method="post">
<input type="hidden" name="id" value="<%=u.getId() %>"/>
<table>
<tr><td>Name:</td><td>
<input type="text" name="name" value="<%= u.getName()%>"/></td></tr>
<tr><td>Password:</td><td>
<input type="password" name="password" value="<%= u.getPassword()%>"/></td></tr>
<tr><td>Email:</td><td>
<input type="email" name="email" value="<%= u.getEmail()%>"/></td></tr>
<tr><td>Sex:</td><td>
<input type="radio" name="sex" value="male"/>Male
<input type="radio" name="sex" value="female"/>Female </td></tr>  <tr><td>Country:</td><td>
<select name="country">
<option>India</option>
<option>USA</option>
<option>Pakistan</option>
<option>Australia</option>
<option>England</option>
</select>
</td></tr>
```

```
<tr><td colspan="2"><input type="submit" value="Edit User"/></td></tr>
</table>
</form>
</body>
</html>
```

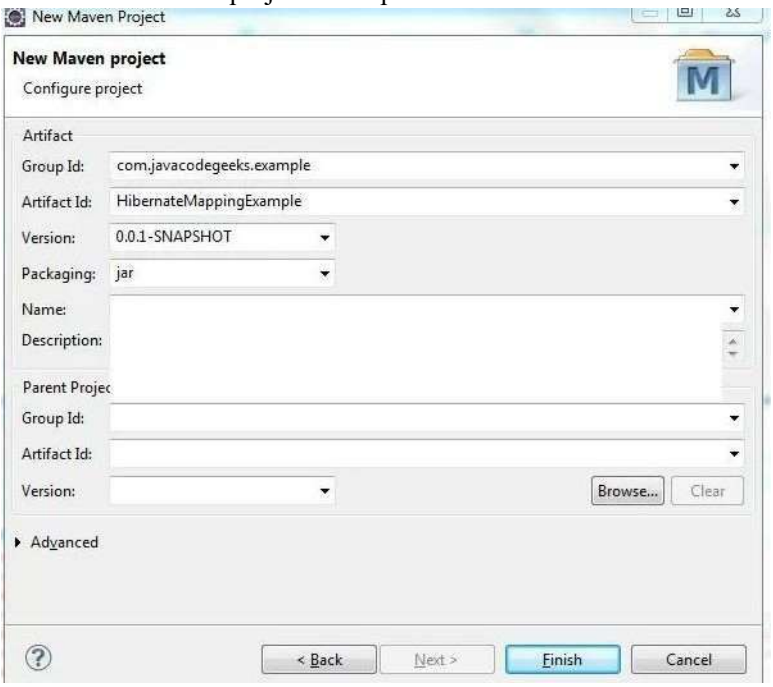
edituser.jsp

```
<%@page import="com.javatpoint.dao.UserDao"%>
<jsp:useBean id="u" class="com.javatpoint.bean.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>
<%
int i=UserDao.update(u);
response.sendRedirect("viewusers.jsp");
%>  deleteuser.jsp
<%@page import="com.javatpoint.dao.UserDao"%>
<jsp:useBean id="u" class="com.javatpoint.bean.User"></jsp:useBean>
<jsp:setProperty property="*" name="u"/>
<%
UserDao.delete(u);
response.sendRedirect("viewusers.jsp");
%>
```

19. Create a program to display the values of User information using Hibernate

Create a Maven Project

Create a new maven project in eclipse. Fill in the details



Add hibernate dependencies

Now we will add the required dependencies for Hibernate. Here we will use Maven's feature of dependency management in pom.xml. *pom.xml*

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.javacodegeeks.example</groupId>
  <artifactId>HibernateMappingExample</artifactId>
  <version>0.0.1-SNAPSHOT</version>
    <dependencies>
    <dependency>
       <groupId>org.hibernate</groupId>
       <artifactId>hibernate-core</artifactId>
       <version>4.3.11.Final</version>
    </dependency>
  </dependencies>
</project>
```

Create Hibernate Configuration File

Let's create a hibernate configuration file `hibernate.cfg.xml` under resources directory in the project.

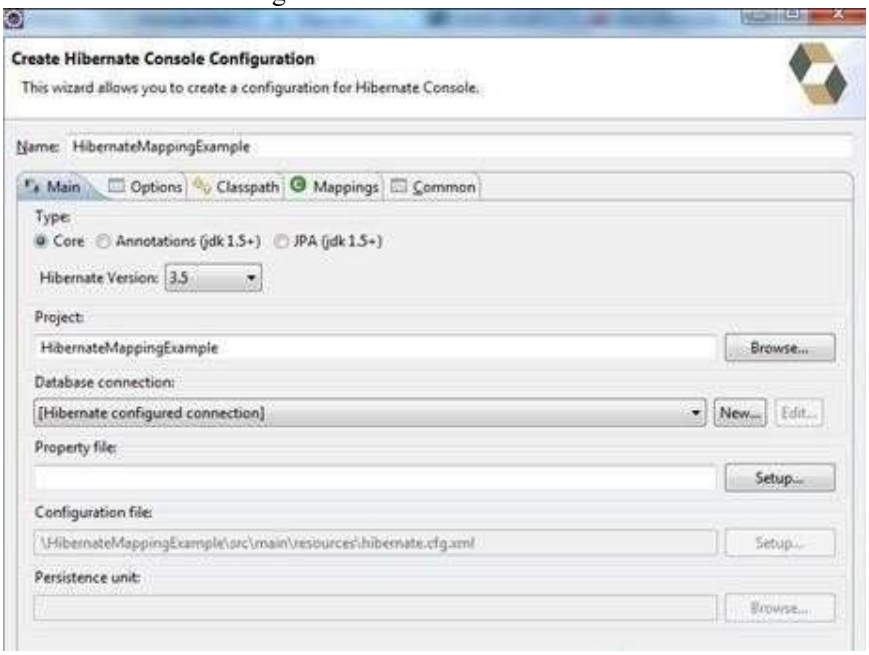
"Create a console configuration" checkbox on the screen as shown above, to configure Hibernate Console.

The `hibernate.cfg.xml` file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">org.gjt.mm.mysql.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://127.0.0.1/hbtutorial</property>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
```

```
    </session-factory>
</hibernate-configuration>
```

Hibernate Console Configuration



Once you have selected the checkbox for "Create a console configuration", click Next to go on the screen shown below for hibernate console and click Finish.

Open Hibernate Console Configuration view in eclipse and it will show database and all the tables of that database.

Create Java Objects

Since we are showing one-to-many relationship mapping in this example, we will create two different Java classes. We will create Employee.java and Department.java objects for our example. *Department.java*

```
package com.javacodegeeks.example;

import java.util.Set;

public class Department {

    private Long id;

    private String departmentName;

    private Set employees;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
this.id = id;
```

```
}

    public String getDepartmentName() {
        return departmentName;
    }

    public void setDepartmentName(String departmentName) {
 this.departmentName = departmentName;
    }

    public Set getEmployees() {
        return employees;
    }

    public void setEmployees(Set employees) {
        this.employees = employees;
    }
}
```

Employee.java

```
package com.javacodegeeks.example
import java.sql.Date;

public class Employee {
        private Long id;

    private String firstname;

    private String lastname;

    private Department department;

    public Employee() {
    }

    public Employee(String firstname, String lastname) {
        this.setFirstname(firstname);
        this.setLastname(lastname);
    }

    public Long getId() {
 return id;
    }

    public void setId(Long id) {
 this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }
```

```
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public Department getDepartment() {
        return department;
    }

    public void setDepartment(Department department) {
this.department = department;
    }
}
```

Map Java Objects to Database

Let's create a mapping file for each java object to database. In eclipse, under src -> main -> resources, create a file with option Hibernate XML Mapping file (hbm.xml). We will have `Employee.hbm.xml` and `Department.hbm.xml`.

Employee.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Jul 11, 2016 10:24:48 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="com.javacodegeeks.example.Employee" table="EMPLOYEE">
        <id name="id" type="java.lang.Long">
            <column name="ID" />
            <generator class="assigned" />
        </id>
        <property name="firstname" type="java.lang.String">
            <column name="FIRSTNAME" />
        </property>
        <property name="lastname" type="java.lang.String">
            <column name="LASTNAME" />
        </property>
        <many-to-one name="department" class="com.javacodegeeks.example.Department" fetch="join">
<column name="DEPARTMENT_ID" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

Department.hbm.xml

use<?xml version="1.0"?>

```
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Jul 11, 2016 10:24:48 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
   <class name="com.javacodegeeks.example.Department" table="DEPARTMENT">
      <id name="id" type="java.lang.Long">
         <column name="ID" />
         <generator class="assigned" />
      </id>
      <property name="departmentName" type="java.lang.String">
         <column name="DEPT_NAME" />
      </property>
      <set name="employees" table="EMPLOYEE" inverse="false" lazy="true">
         <key>
            <column name="DEPARTMENT_ID" />
         </key>
         <one-to-many class="com.javacodegeeks.example.Employee"/>
      </set>
   </class>
</hibernate-mapping>
```

Create a Hibernate Test Program

Let's create a test hibernate program to insert some employees for some departments. The source code looks like below

HibernateTestMappingProgram.java

```
package com.javacodegeeks.example;

import org.hibernate.HibernateException;
import org.hibernate.Session; import
org.hibernate.SessionFactory; import
org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder; import
org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;

public class HibernateTestMappingProgram {
private static SessionFactory factory;
   private static ServiceRegistry serviceRegistry;

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

      Configuration config = new Configuration();
config.configure();
      config.addAnnotatedClass(Department.class);
config.addResource("Department.hbm.xml");
config.addAnnotatedClass(Employee.class);        config.addResource("Employee.hbm.xml");
```

```
        serviceRegistry = new StandardServiceRegistryBuilder().applySettings(config.getProperties()).build();
factory = config.buildSessionFactory(serviceRegistry);

        HibernateTestMappingProgram hbTest = new HibernateTestMappingProgram();



        hbTest.insertEmployee(1,"Mark","Johnson","Sales",1);
hbTest.insertEmployee(2,"Jill","Johnson","Marketing",2);
    }

   private long insertEmployee(int id, String firstname, String lastname, String deptName, int deptId)
   {
        Session session = factory.openSession();
        Transaction tx = null;
Long empIdSaved = null;
        try {
            tx = session.beginTransaction();
Department d = new Department();
d.setDepartmentName(deptName);
            d.setId(new Long(deptId));
            session.save(d);
            Employee e = new Employee();
e.setFirstname(firstname);
            e.setLastname(lastname);
            e.setId(new Long(id));
            e.setDepartment(d);
empIdSaved = (Long) session.save(e);

            tx.commit();
        } catch(HibernateException ex) {
if(tx != null)                tx.rollback();
ex.printStackTrace();
        } finally {
session.close();
        }

        return empIdSaved;

   }

}
```

9. Create an application in Spring to make injecting codes in string based information

Employee.java

It is a simple class containing two fields id and name. There are four constructors and one method in this class.

```
package com.javatpoint;
  public class Employee {
private int id;
private String name;
```

```
  public Employee() {System.out.println("def cons");}
  public Employee(int id) {this.id = id;}
  public Employee(String name) { this.name = name;}
  public Employee(int id, String name) {
    this.id = id;
    this.name = name;
}
  void show(){
    System.out.println(id+" "+name);
}
}
```

applicationContext.xml

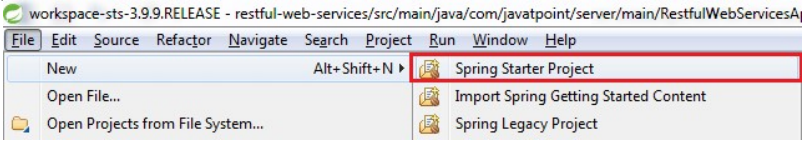
```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
  <bean id="e" class="com.javatpoint.Employee">
<constructor-arg value="10" type="int"></constructor-arg>
</bean>
  </beans>
```

Test.java

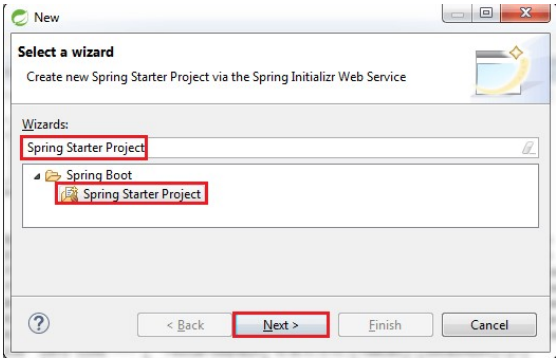
```
package com.javatpoint;
  import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.*;
  public class Test {
    public static void main(String[] args) {
              Resource r=new ClassPathResource("applicationContext.xml");
       BeanFactory factory=new XmlBeanFactory(r);
              Employee s=(Employee)factory.getBean("e");
       s.show();
     }
}
```

20. Create an application with restful Web Services Project with Spring Boot

Step 1: Download the **Spring Tool Suite (STS)** from https://spring.io/tools3/sts/all and extract it.
Step 2: Launch the **STS**.
Step 3: Click on **File menu -> New -> Spring Starter Project ->**

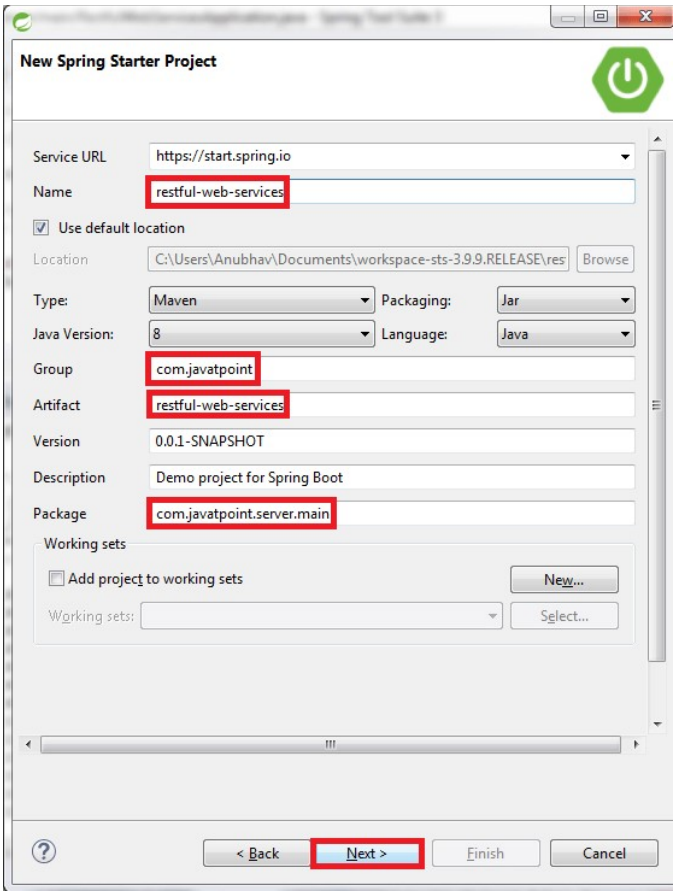


If the **Spring Starter Project** is not enlisted, then click on **Other** at the bottom of the menu. A dialog box appears on the screen. Type **Spring Starter Project** in the **Wizards** text box and click on the **Next** button.

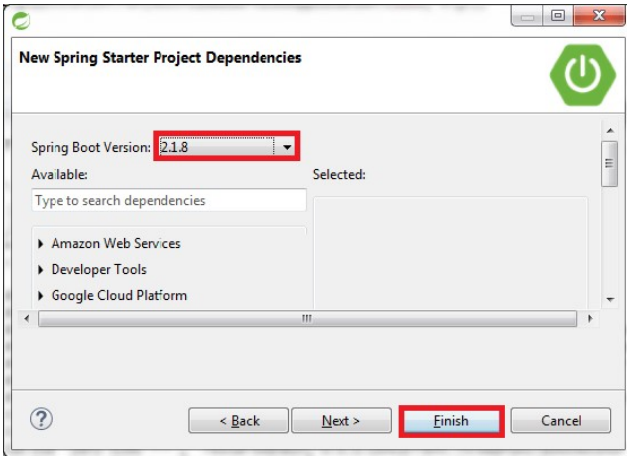


Step 4: provide the name, group, and package of the project. We have provided:

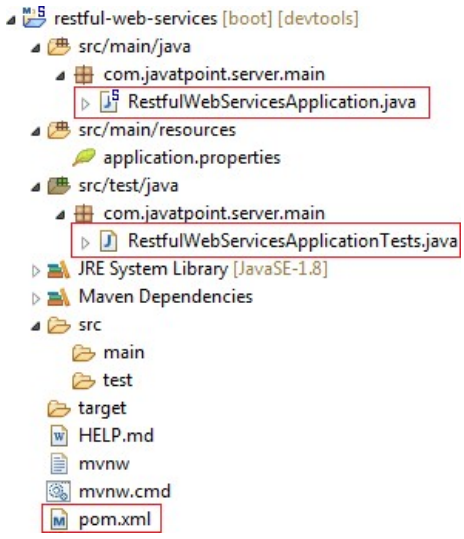
Name: **restful-web-services**
Group: **com.javatpoint**
Package: **com.javatpoint.server.main**
Click on the **Next** button.



Step 5: Choose the Spring Boot Version **2.1.8**.



Step 6: We can see the project structure in the project explorer window.



Step 7: Go to the Maven Repository https://mvnrepository.com/ and add **Spring Web MVC, Spring Boot DevTools, JPA,** and **H2** dependencies in the pom.xml. After adding the dependencies, the pom.xml file looks like the following:

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.1.8.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
```

```
<groupId>com.javatpoint</groupId>
<artifactId>restful-web-services</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>restful-web-services</name>
<description>Demo project for Spring Boot</description>
<properties>
<java.version>1.8</java.version>
</properties>
<dependencies>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.apache.maven</groupId>
<artifactId>maven-archiver</artifactId>
<version>2.5</version>
</dependency>
</dependencies>
<build> <plugins> <plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin> </plugins> </build> </project>
```

Step 8: Now open the **RestfulWebServicesApplication.java** file and Run the file as Java Application.

```
package com.javatpoint.server.main;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class RestfulWebServicesApplication
{ public static void main(String[] args)
{ SpringApplication.run(RestfulWebServicesApplication.class, args);
}
}
```

4. Program with Swing and Databases
5. Develop a project using Objects, Methods and Classes with its databases.
6. (a) Write a JDBC specification to establish the connection with the Mysql,
   (b). Write a JDBC specification to insert a record into the Employee database.
7. Write a JDBC specification to implement batch-updates.
8. Write a program to implement JDBC transaction
9. Construct a Servlet program to print a welcome message and its number of user visiting page
10. Construct a servlet to handle sessions using URL-rewriting
11. Develop a program in cookie and identify cookie information using servlets
12. (a)Implement a program to display date and time in JSP
    (b).Write a JSP program to include and forward web pages