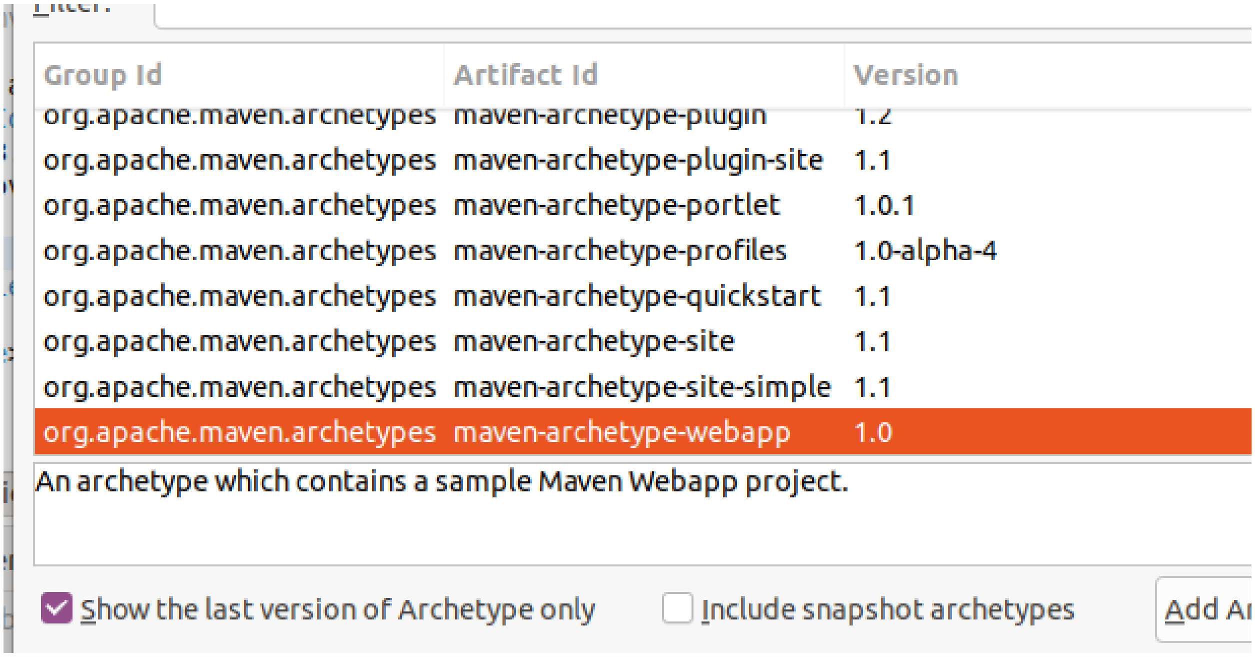
Registration Page

- Create a Maven project of the type web app. After creating the project, it will give one error in the HTML page.

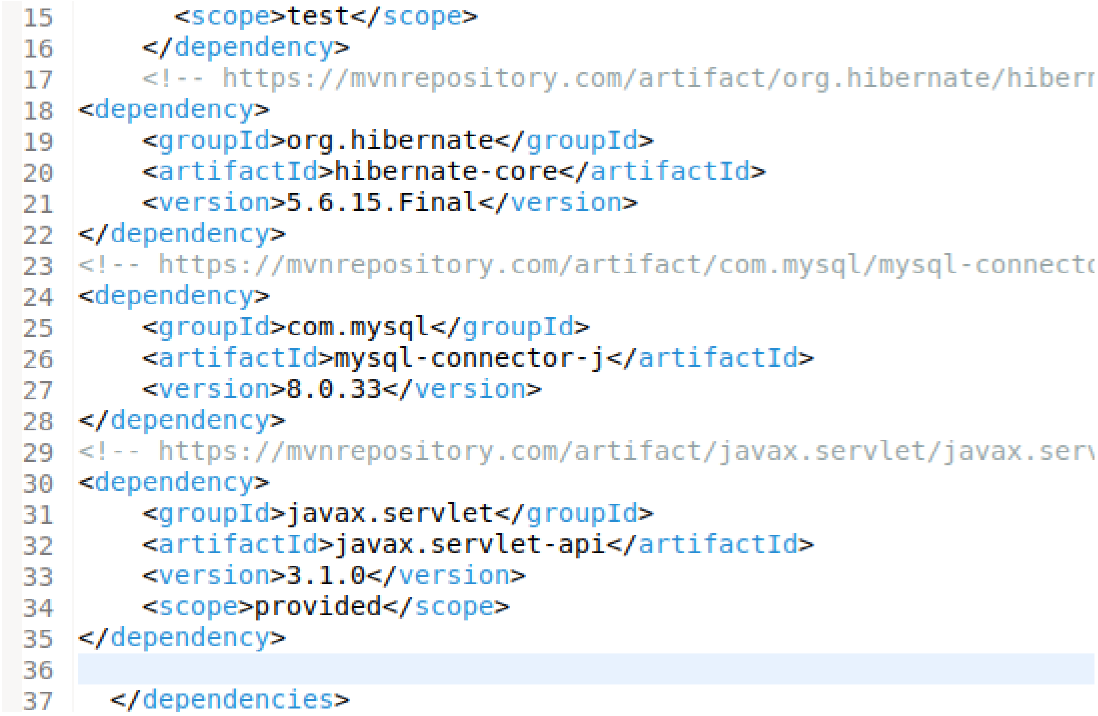


Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1
org.apache.maven.archetypes	maven-archetype-webapp	1.0

An archetype which contains a sample Maven Webapp project.

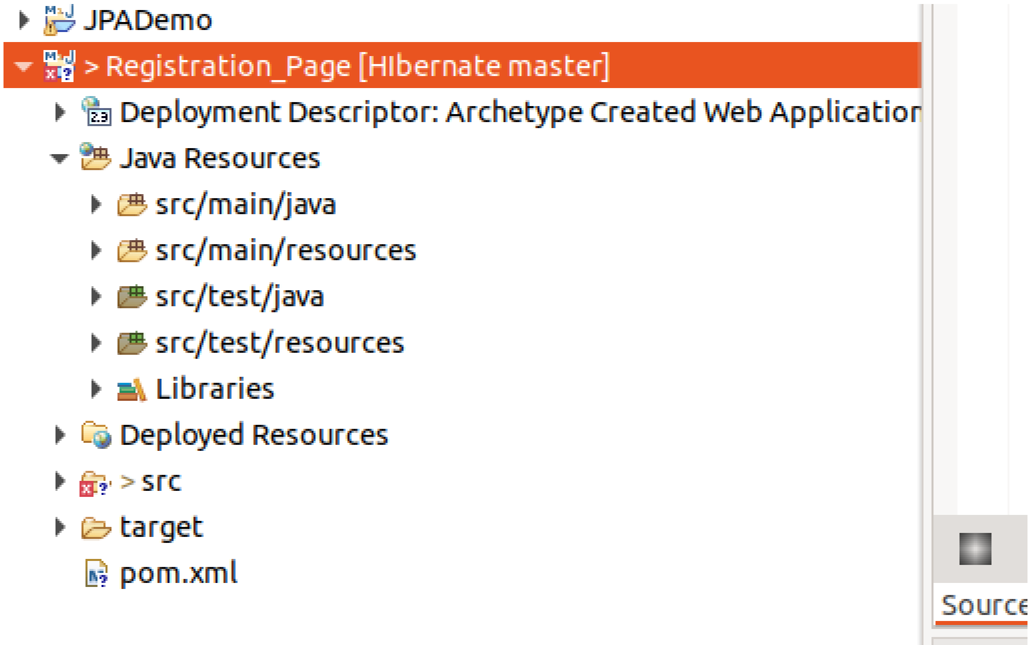
☑ Show the last version of Archetype only ☐ Include snapshot archetypes

- Import following dependency



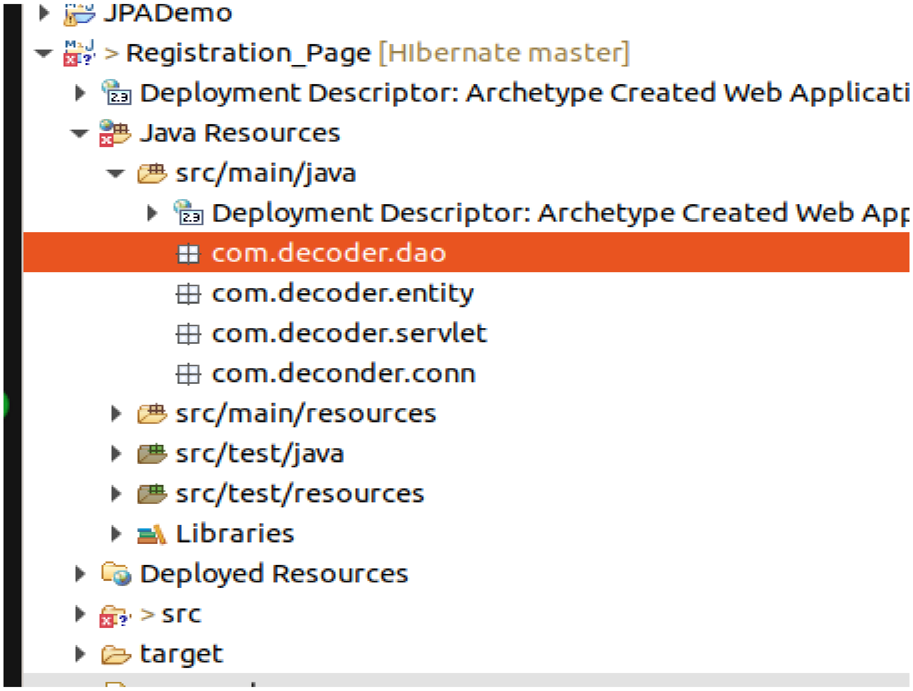
```
   <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hiberr
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.6.15.Final</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.mysql/mysql-connecto
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <version>8.0.33</version>
</dependency>
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.serv
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>

 </dependencies>
```

- The note folder structure should be the following: if the main/java file is not there, please change your java version; it should be your system version; by default, maven selects an older version of java.



```
JPADemo
> Registration_Page [HIbernate master]
    Deployment Descriptor: Archetype Created Web Application
    Java Resources
        src/main/java
        src/main/resources
        src/test/java
        src/test/resources
        Libraries
    Deployed Resources
    > src
    target
    pom.xml
```

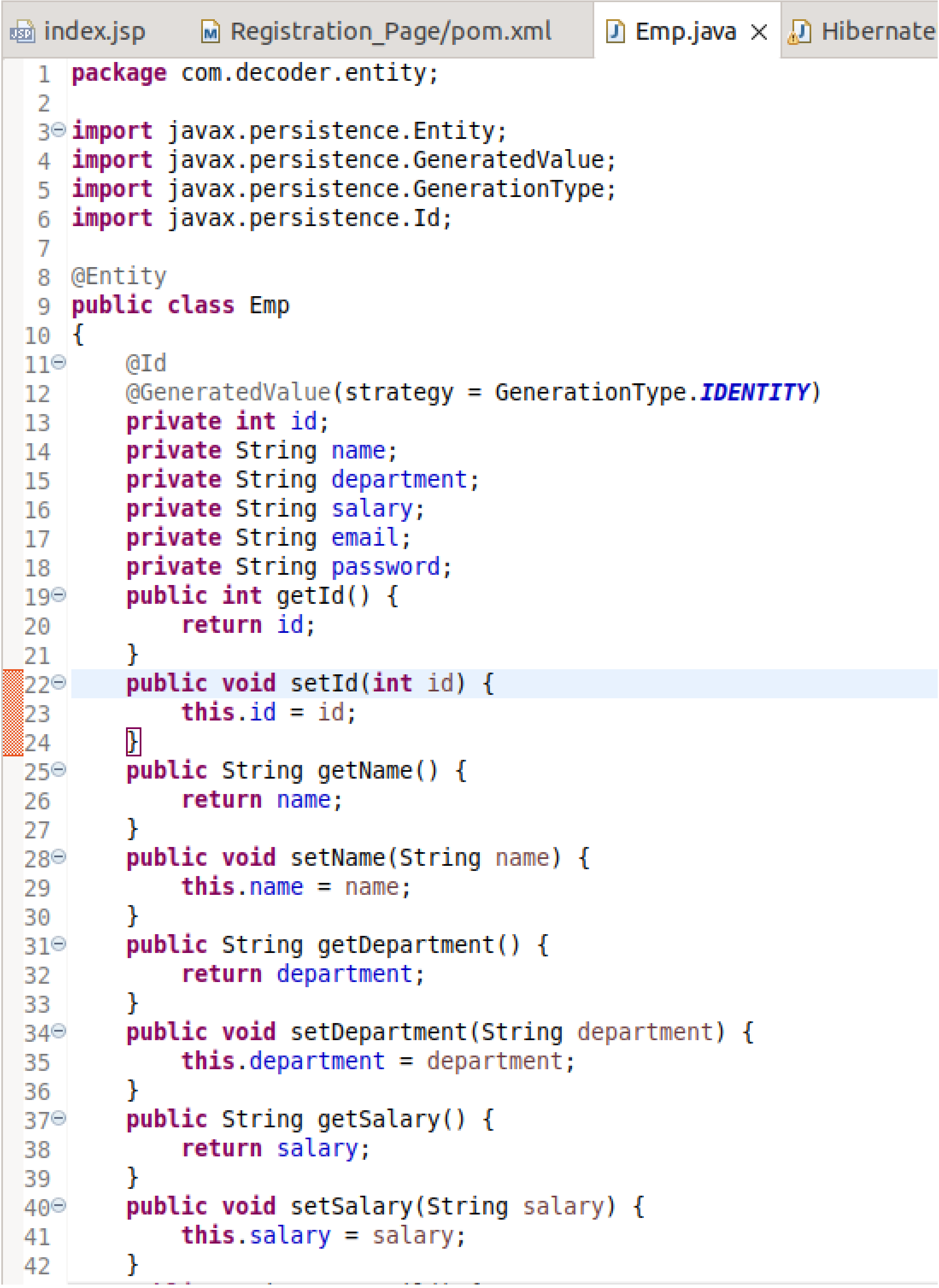
-
- Create three packages, each package used for each specific task
    - Conn
    - Entity
    - Servlet
    - Dao



```
JPADemo
> Registration_Page [HIbernate master]
    Deployment Descriptor: Archetype Created Web Applicati
    Java Resources
        src/main/java
            Deployment Descriptor: Archetype Created Web App
            com.decoder.dao
            com.decoder.entity
            com.decoder.servlet
            com.deconder.conn
        src/main/resources
        src/test/java
        src/test/resources
        Libraries
    Deployed Resources
    > src
    target
```

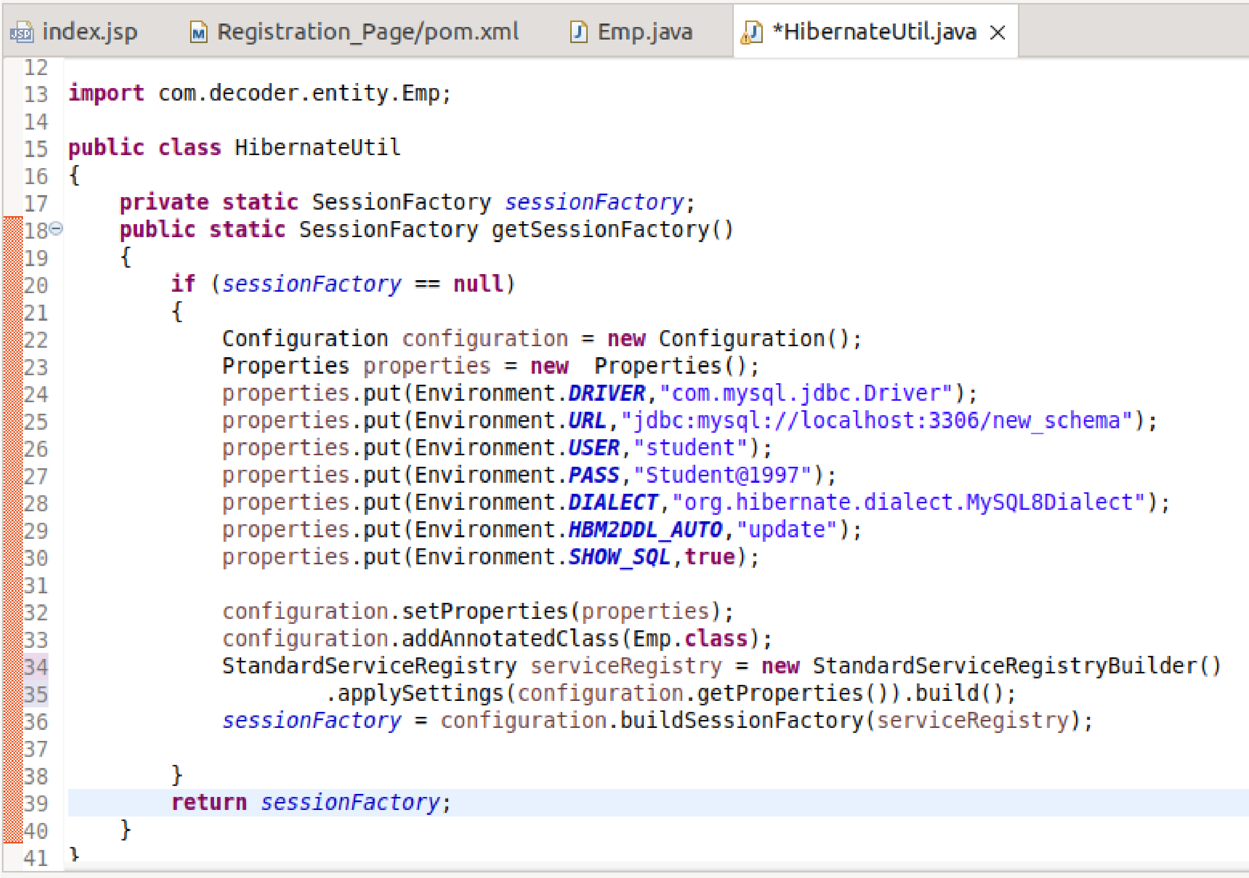
-

- For Entity, I create an “Employee/Emp” class entity
    - In this Emp class, I added Enity, id annotation, and “`@GeneratedValue(strategy = GenerationType.IDENTITY)`,” which generates autoincrement ID.



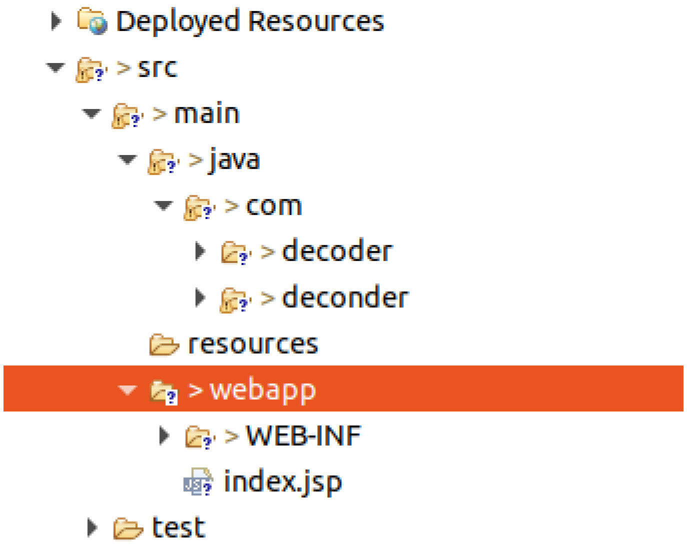
```java
package com.decoder.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Emp
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String department;
    private String salary;
    private String email;
    private String password;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
    }
    public String getSalary() {
        return salary;
    }
    public void setSalary(String salary) {
        this.salary = salary;
    }
```

- For Configuration with database, I created a single Java file that contains the property.



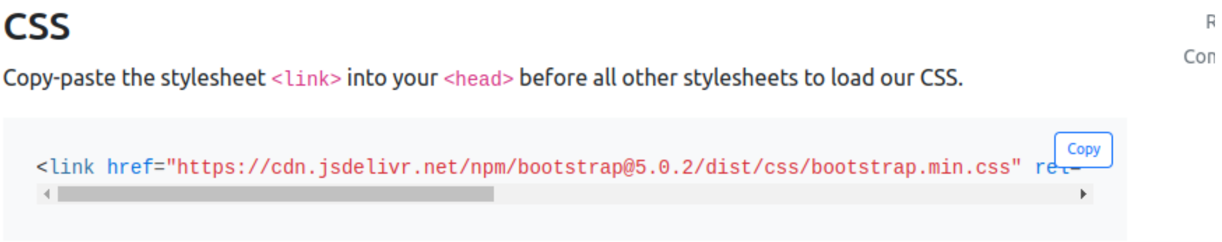
```java

import com.decoder.entity.Emp;

public class HibernateUtil
{
    private static SessionFactory sessionFactory;
    public static SessionFactory getSessionFactory()
    {
        if (sessionFactory == null)
        {
            Configuration configuration = new Configuration();
            Properties properties = new  Properties();
            properties.put(Environment.DRIVER,"com.mysql.jdbc.Driver");
            properties.put(Environment.URL,"jdbc:mysql://localhost:3306/new_schema");
            properties.put(Environment.USER,"student");
            properties.put(Environment.PASS,"Student@1997");
            properties.put(Environment.DIALECT,"org.hibernate.dialect.MySQL8Dialect");
            properties.put(Environment.HBM2DDL_AUTO,"update");
            properties.put(Environment.SHOW_SQL,true);

            configuration.setProperties(properties);
            configuration.addAnnotatedClass(Emp.class);
            StandardServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
                    .applySettings(configuration.getProperties()).build();
            sessionFactory = configuration.buildSessionFactory(serviceRegistry);

        }
        return sessionFactory;
    }
}
```

- Now, front End, Delete the existing index.jsp file in the web app
- Create a new index.jsp file in the same folder



```
▸ Deployed Resources
▾ > src
    ▾ > main
        ▾ > java
            ▾ > com
                ▸ > decoder
                ▸ > deconder
            resources
        ▾ > webapp
            ▸ > WEB-INF
              index.jsp
    ▸ test
```

- In index.jsp file, for creating a web page, I directly use “boostrap5” → search in Google and copy CSS link.

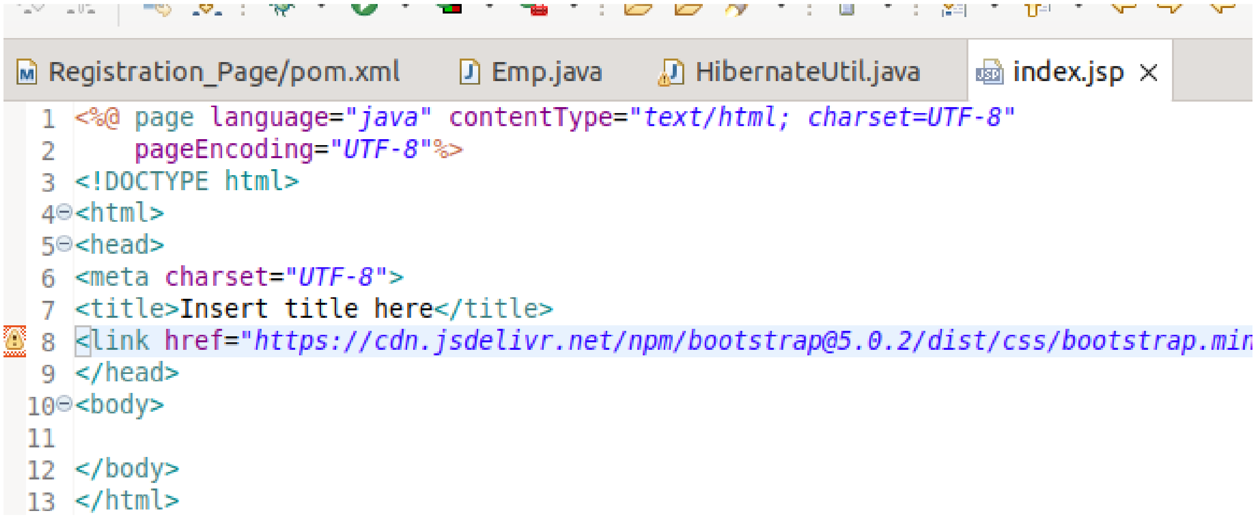


CSS

Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load our CSS.

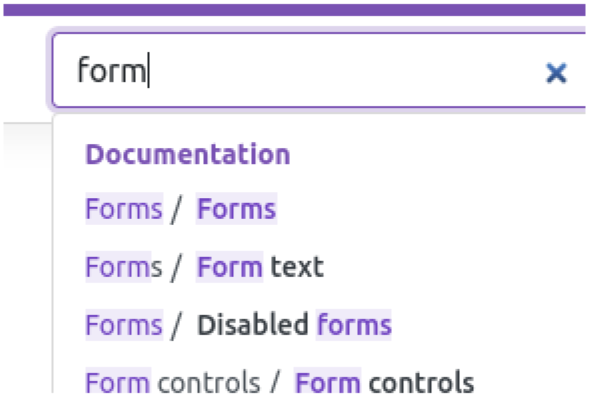
```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" re
```

- Paste it in the “head” tag.



```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min
</head>
<body>

</body>
</html>
```

- To create a form page, I search for forms on the Bootstrap website.



form

Documentation

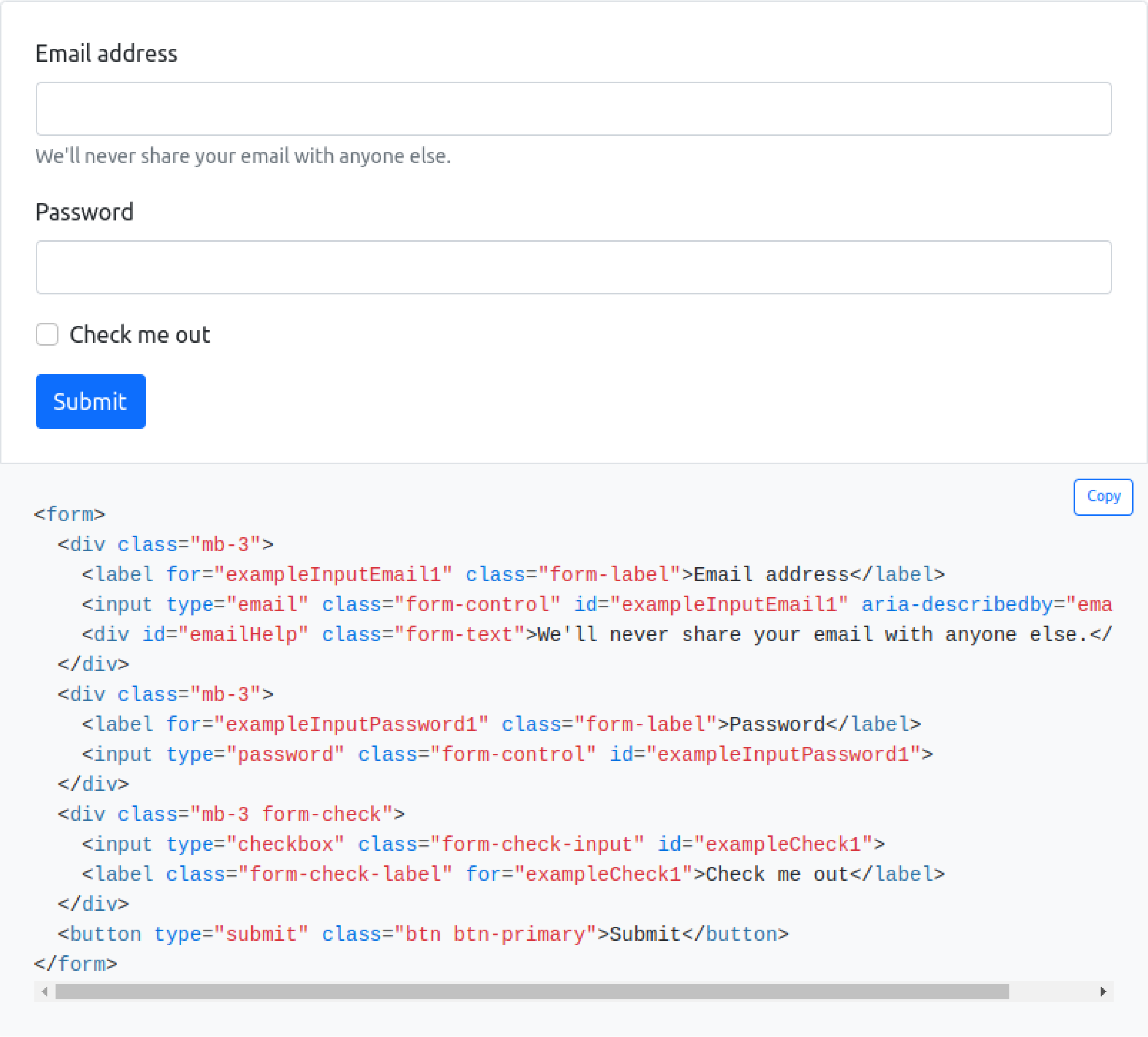
Forms / Forms

Forms / Form text

Forms / Disabled forms

Form controls / Form controls

- Copy this form code.



Email address

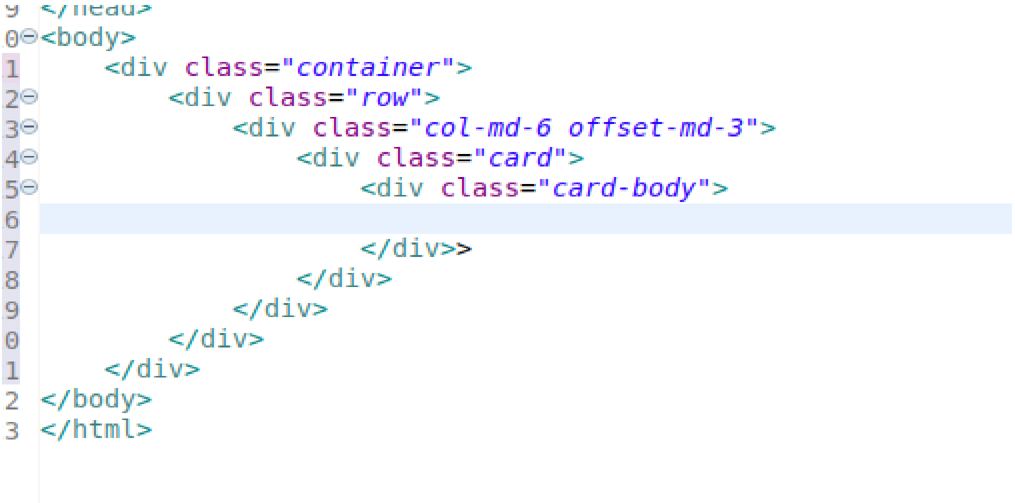
We'll never share your email with anyone else.

Password

□ Check me out

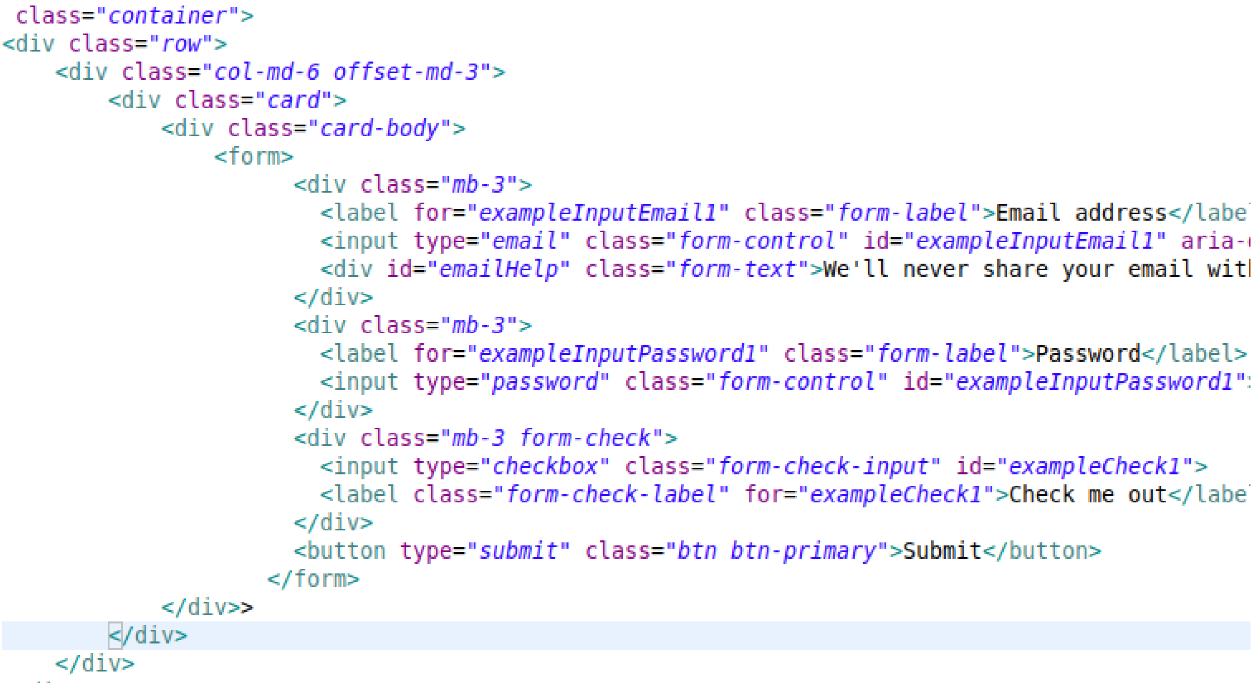
```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="ema
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

-
- Before the paste, make some modifications to the index.jsp file, add some div tags, and paste the form code here



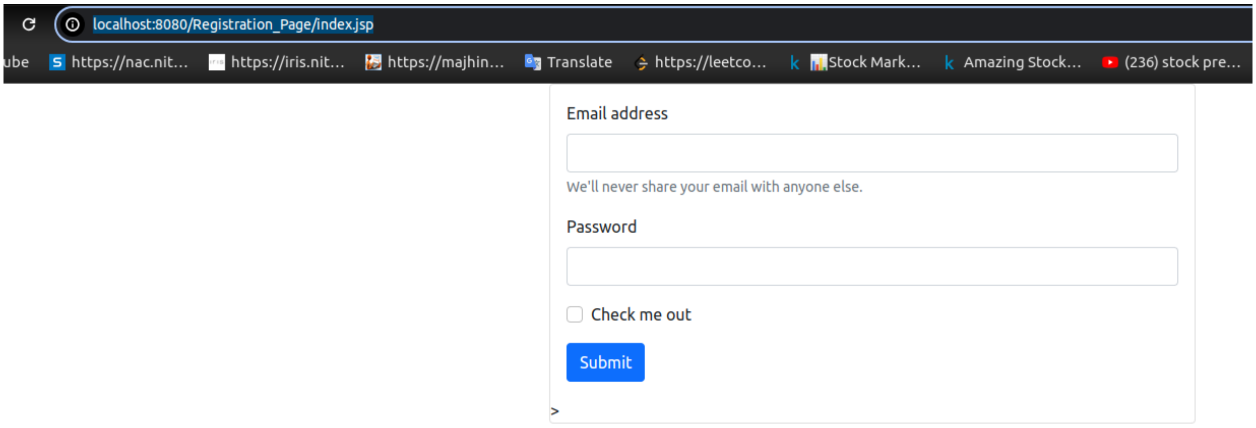
```
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-md-6 offset-md-3">
                <div class="card">
                    <div class="card-body">

                    </div>>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

-



```
 class="container">
<div class="row">
    <div class="col-md-6 offset-md-3">
        <div class="card">
            <div class="card-body">
                <form>
                      <div class="mb-3">
                        <label for="exampleInputEmail1" class="form-label">Email address</labe
                        <input type="email" class="form-control" id="exampleInputEmail1" aria-
                        <div id="emailHelp" class="form-text">We'll never share your email wit
                      </div>
                      <div class="mb-3">
                        <label for="exampleInputPassword1" class="form-label">Password</label>
                        <input type="password" class="form-control" id="exampleInputPassword1"
                      </div>
                      <div class="mb-3 form-check">
                        <input type="checkbox" class="form-check-input" id="exampleCheck1">
                        <label class="form-check-label" for="exampleCheck1">Check me out</labe
                      </div>
                      <button type="submit" class="btn btn-primary">Submit</button>
                    </form>
            </div>>
        </div>
    </div>
```

-
- Running this application →, install Topcat 8.5 server → click on the run, find the server, and install.
- After installation, run the code → open chrome → "http://localhost:8080/Registration_Page/index.jsp"



Email address

We'll never share your email with anyone else.

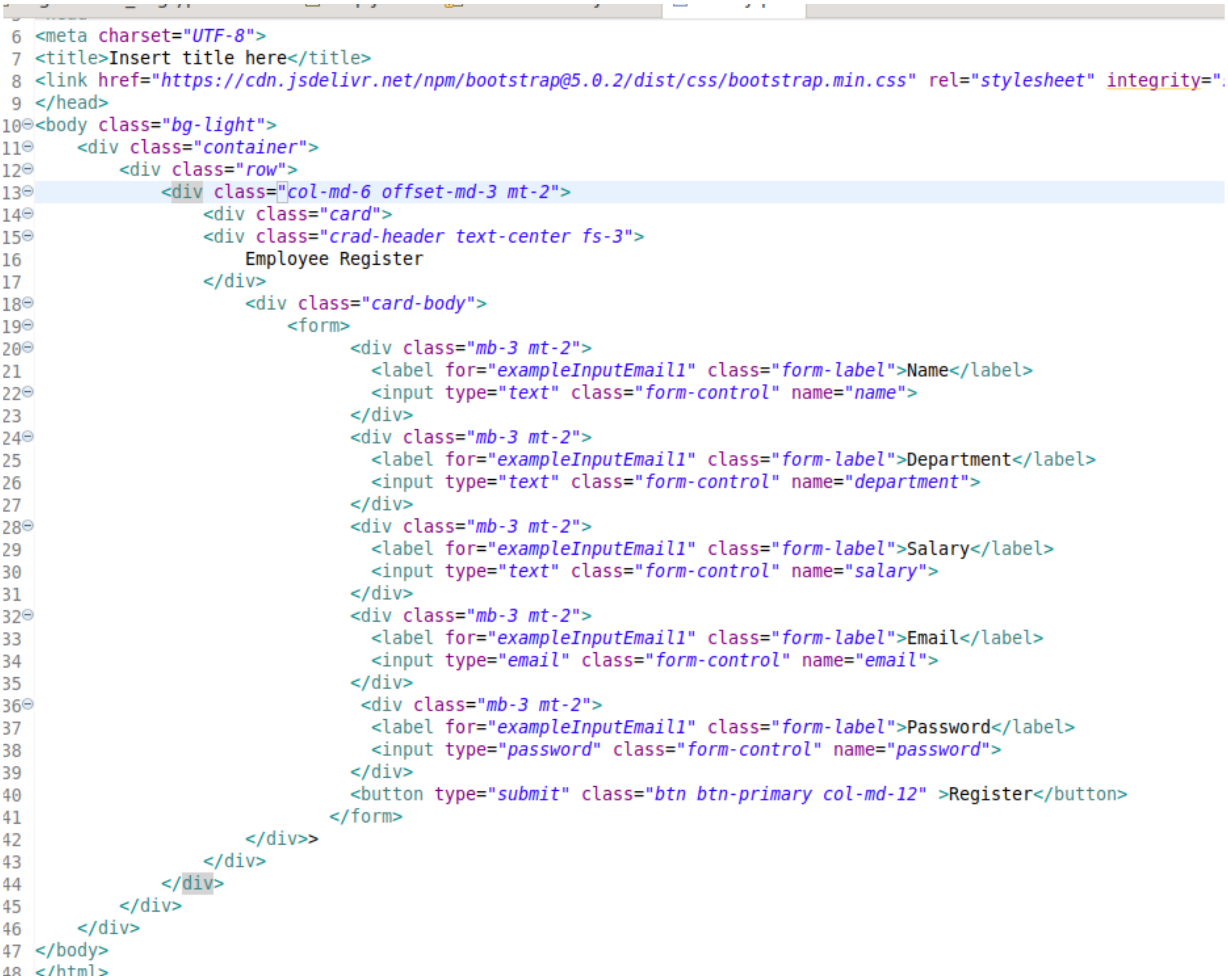
Password

□ Check me out

Submit

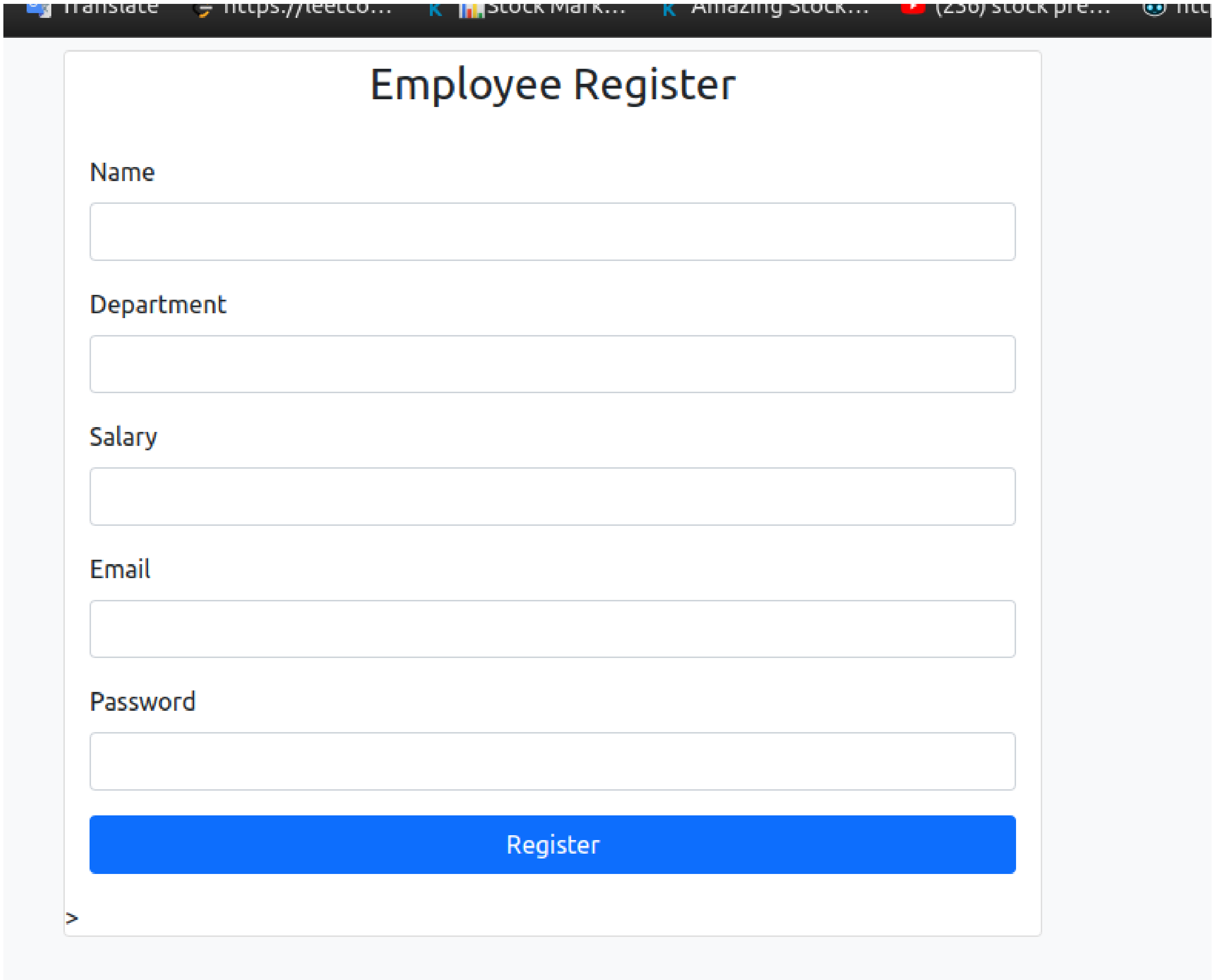
>

-
- The server is running correctly.
- In index.jsp file, make frontend changes according to your needs. I made all the changes referring to the following code.



```
<meta charset="UTF-8">
<title>Insert title here</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="
</head>
<body class="bg-light">
    <div class="container">
        <div class="row">
            <div class="col-md-6 offset-md-3 mt-2">
                <div class="card">
                <div class="crad-header text-center fs-3">
                    Employee Register
                </div>
                    <div class="card-body">
                        <form>
                              <div class="mb-3 mt-2">
                                <label for="exampleInputEmail1" class="form-label">Name</label>
                                <input type="text" class="form-control" name="name">
                              </div>
                              <div class="mb-3 mt-2">
                                <label for="exampleInputEmail1" class="form-label">Department</label>
                                <input type="text" class="form-control" name="department">
                              </div>
                              <div class="mb-3 mt-2">
                                <label for="exampleInputEmail1" class="form-label">Salary</label>
                                <input type="text" class="form-control" name="salary">
                              </div>
                              <div class="mb-3 mt-2">
                                <label for="exampleInputEmail1" class="form-label">Email</label>
                                <input type="email" class="form-control" name="email">
                              </div>
                               <div class="mb-3 mt-2">
                                <label for="exampleInputEmail1" class="form-label">Password</label>
                                <input type="password" class="form-control" name="password">
                              </div>
                              <button type="submit" class="btn btn-primary col-md-12" >Register</button>
                            </form>
                    </div>>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

-
- The page looks like this:

- 

  Employee Register

  Name

  Department

  Salary

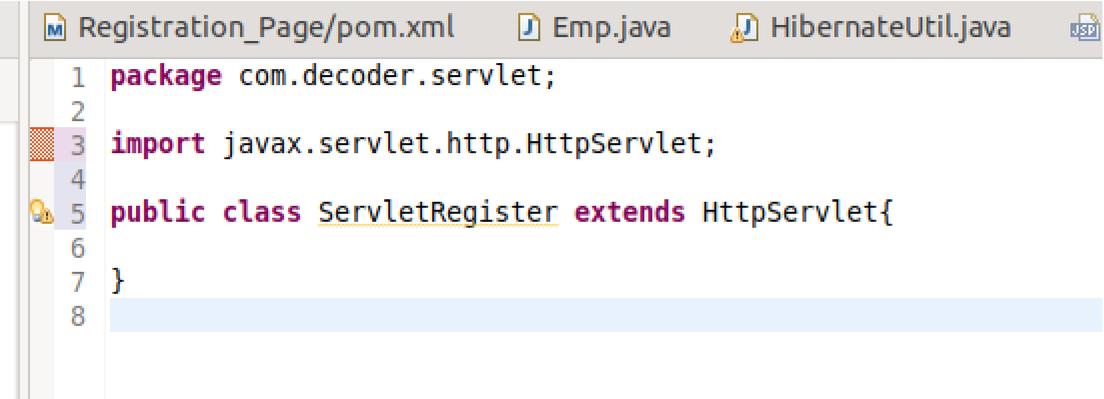
  Email

  Password

  Register

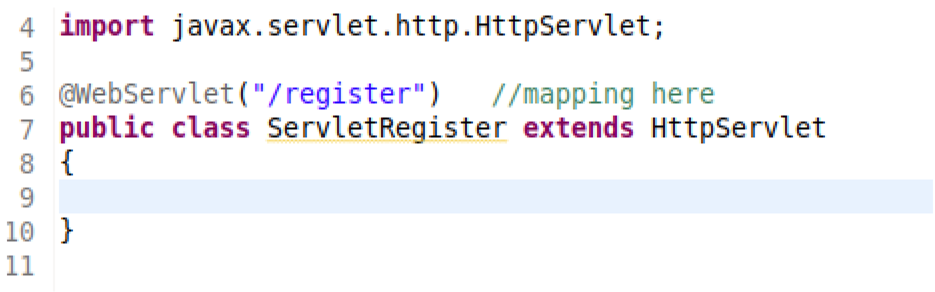
  >

- By using servlet, we give input, so we need to create a class in the servlet package “ServletRegister” and extend Httpservlet.

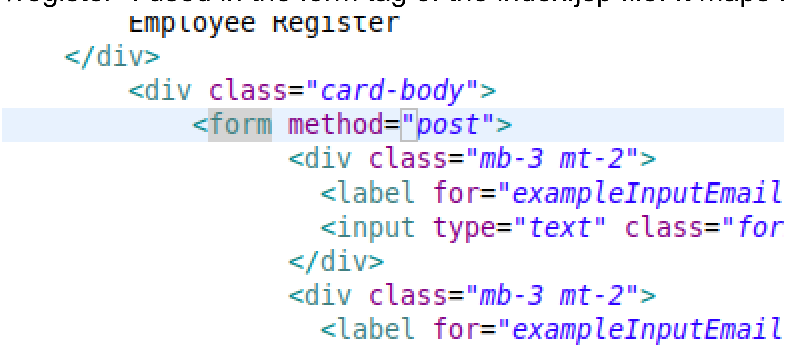
```
package com.decoder.servlet;

import javax.servlet.http.HttpServlet;

public class ServletRegister extends HttpServlet{

}

```

- Add annotation in servletRegister and add method in an index.jsp → form tag → post method

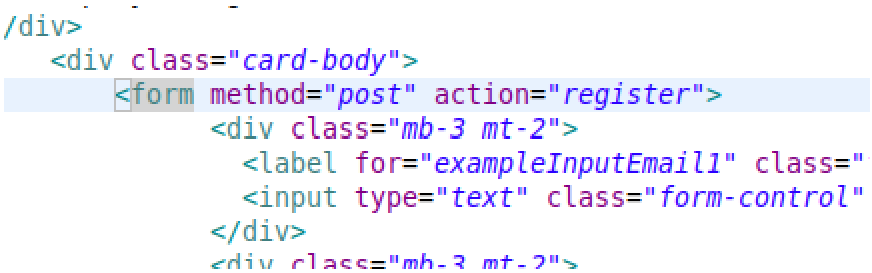


```
import javax.servlet.http.HttpServlet;

@WebServlet("/register")    //mapping here
public class ServletRegister extends HttpServlet
{

}
```

- “/register” I used in the form tag of the index.jsp file. It maps form to Java servlet

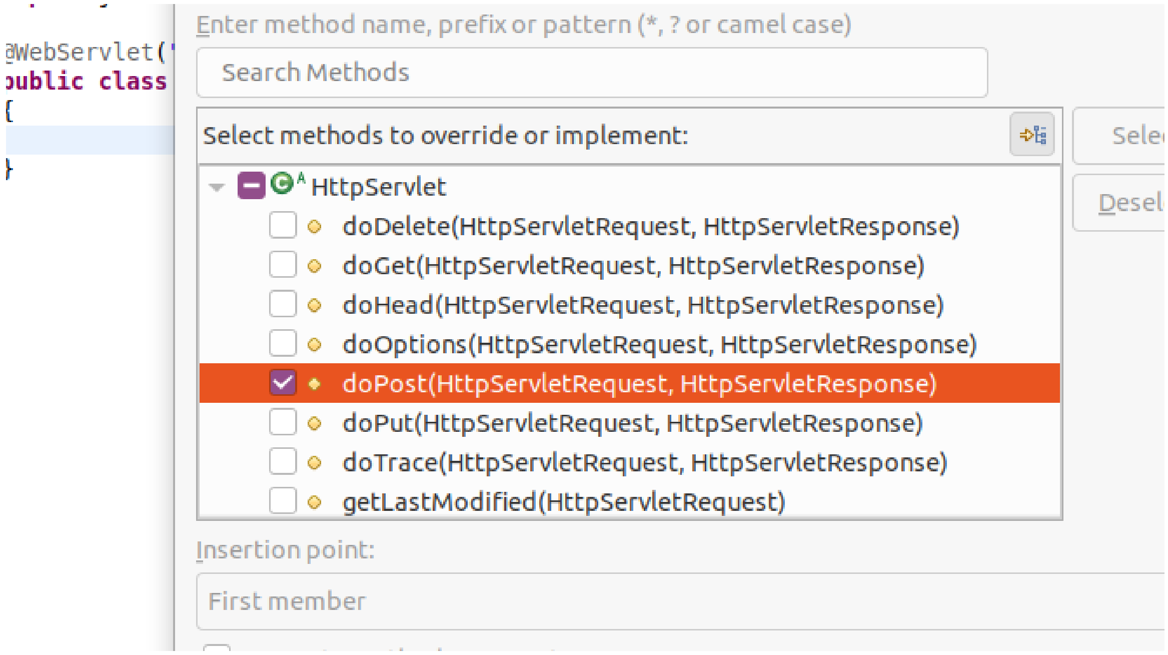


```
        Employee Register
     </div>
        <div class="card-body">
            <form method="post">
                <div class="mb-3 mt-2">
                  <label for="exampleInputEmail
                  <input type="text" class="for
                </div>
                <div class="mb-3 mt-2">
                  <label for="exampleInputEmail
```



```
/div>
   <div class="card-body">
       <form method="post" action="register">
           <div class="mb-3 mt-2">
             <label for="exampleInputEmail1" class="
             <input type="text" class="form-control"
           </div>
           <div class="mb-3 mt-2">
```

- Now override the post method in “ServletRegister” class
  - resource→override/implement method → doPost



```
@WebServlet('
public class
{

}
```

Enter method name, prefix or pattern (*, ? or camel case)

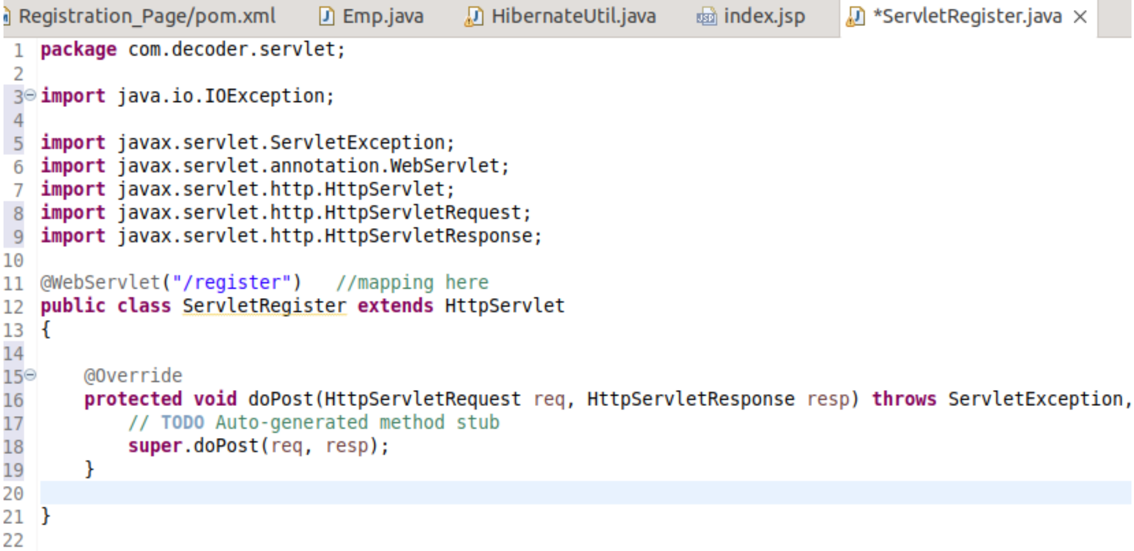
Search Methods

Select methods to override or implement:

- HttpServlet
  - □ doDelete(HttpServletRequest, HttpServletResponse)
  - □ doGet(HttpServletRequest, HttpServletResponse)
  - □ doHead(HttpServletRequest, HttpServletResponse)
  - □ doOptions(HttpServletRequest, HttpServletResponse)
  - ☑ doPost(HttpServletRequest, HttpServletResponse)
  - □ doPut(HttpServletRequest, HttpServletResponse)
  - □ doTrace(HttpServletRequest, HttpServletResponse)
  - □ getLastModified(HttpServletRequest)

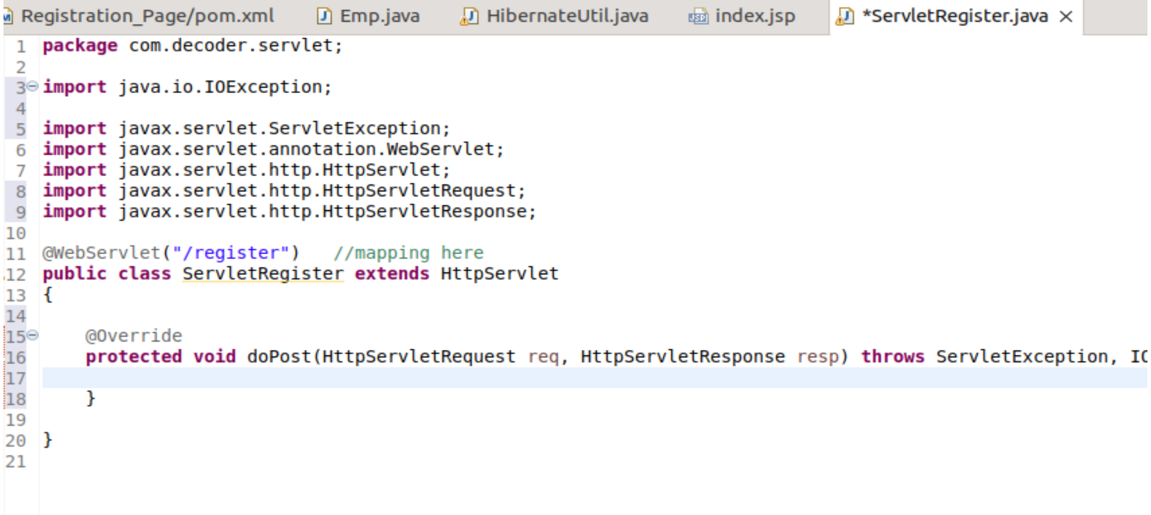
Insertion point:

First member



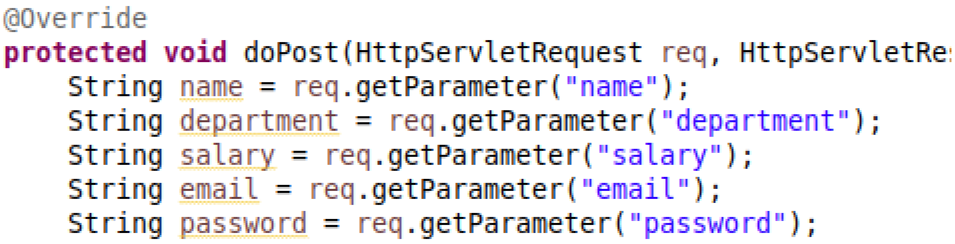
```java
package com.decoder.servlet;

import java.io.IOException;

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("/register")   //mapping here
public class ServletRegister extends HttpServlet
{

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
        // TODO Auto-generated method stub
        super.doPost(req, resp);
    }

}

```

- Remove the extra super method.



```java
package com.decoder.servlet;

import java.io.IOException;

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("/register")   //mapping here
public class ServletRegister extends HttpServlet
{

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IC

    }

}

```

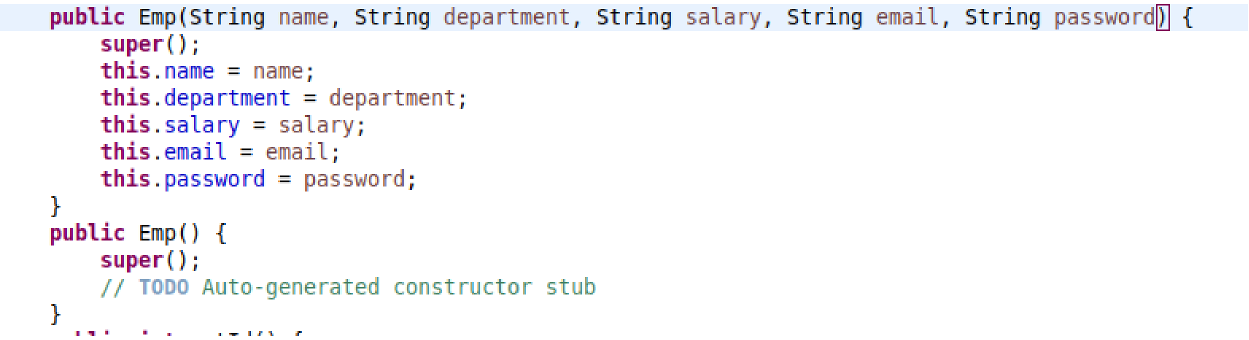
- Now, here, whatever value is put by the user in the front end will be taken by the Servlet register
- In index.jsp file use name=” name,” name=” password”, name=” salary.” These are variables that hold value from the front end; fetching in a java file, we use the names



```java
@Override
protected void doPost(HttpServletRequest req, HttpServletRe
    String name = req.getParameter("name");
    String department = req.getParameter("department");
    String salary = req.getParameter("salary");
    String email = req.getParameter("email");
    String password = req.getParameter("password");
```

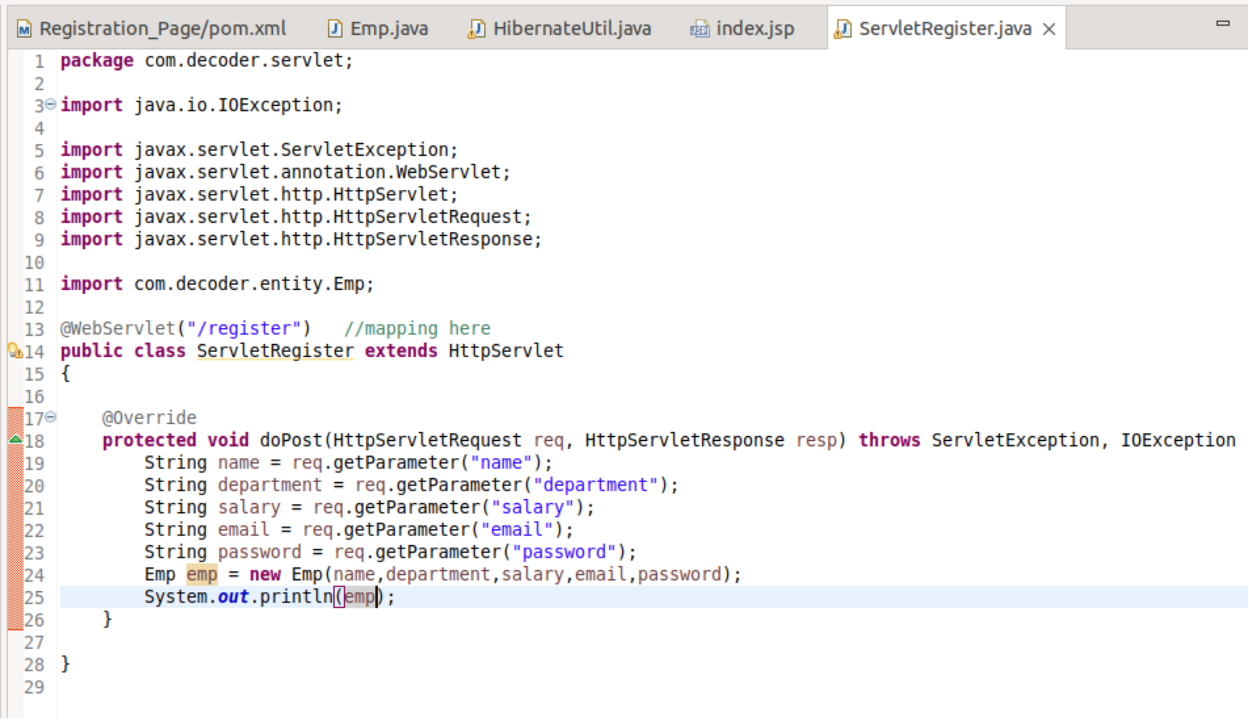
- }
- To value the thrown object, we must create a default and a parameterized constructor in the Entity class.

- For default constructor → resource → generate constructor from super class → click ok
- For parameterized constructor → resource → generated field constructor → select all



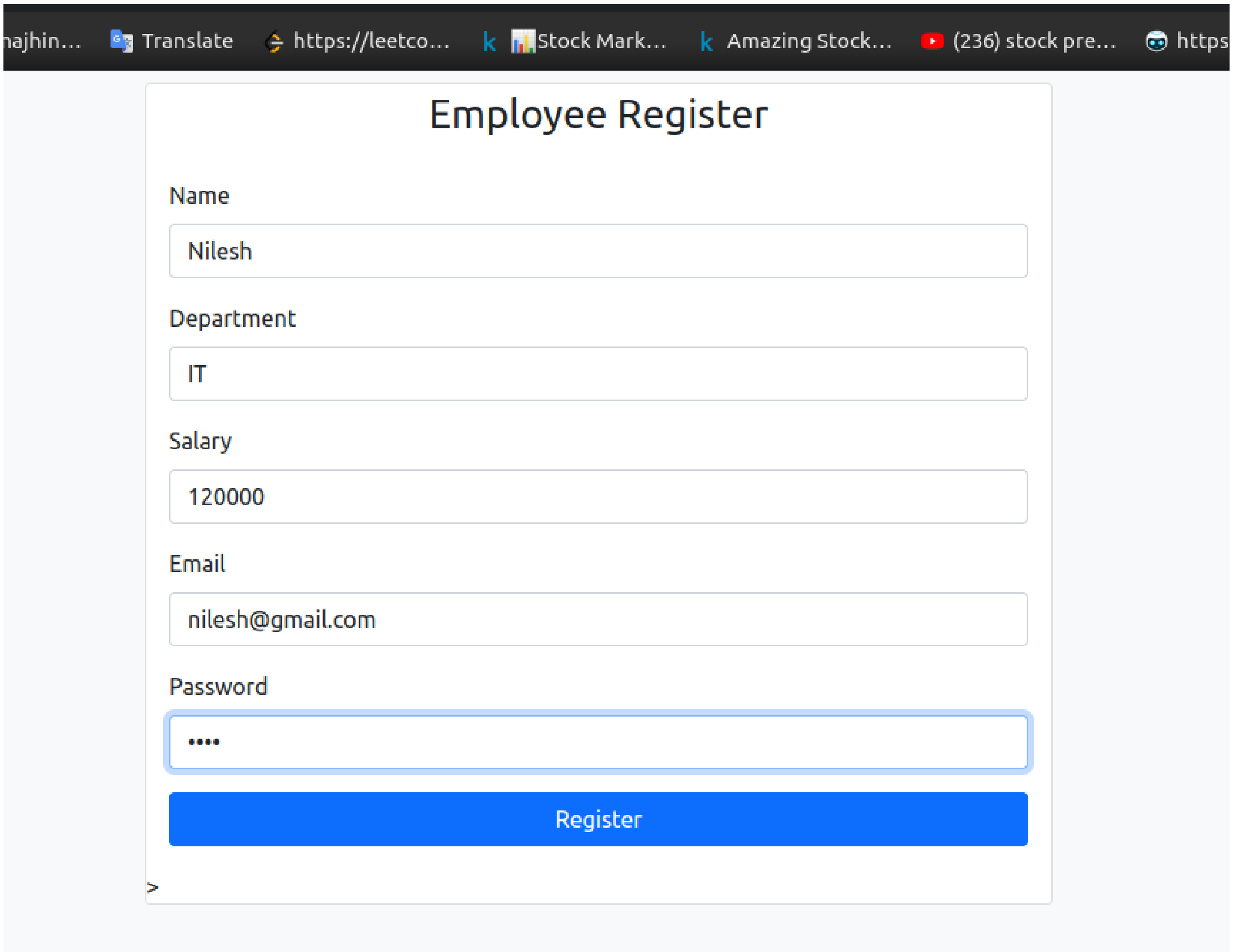
```java
    public Emp(String name, String department, String salary, String email, String password) {
        super();
        this.name = name;
        this.department = department;
        this.salary = salary;
        this.email = email;
        this.password = password;
    }
    public Emp() {
        super();
        // TODO Auto-generated constructor stub
    }
```

- There is no need for an id constructor because automatically getting that value
- Now, the servletregister class file creates the object of the entity and prints:



```java
package com.decoder.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.decoder.entity.Emp;

@WebServlet("/register")   //mapping here
public class ServletRegister extends HttpServlet
{

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
        String name = req.getParameter("name");
        String department = req.getParameter("department");
        String salary = req.getParameter("salary");
        String email = req.getParameter("email");
        String password = req.getParameter("password");
        Emp emp = new Emp(name,department,salary,email,password);
        System.out.println(emp);
    }

}
```

- Run index.jsp file → enter details and check values we are getting in the console:

- 

  Employee Register

  Name

  Nilesh

  Department

  IT

  Salary

  120000

  Email

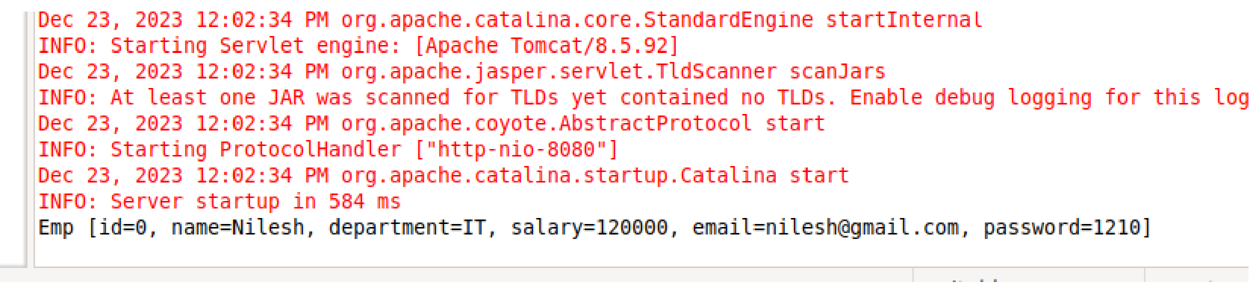
  nilesh@gmail.com

  Password

  ••••

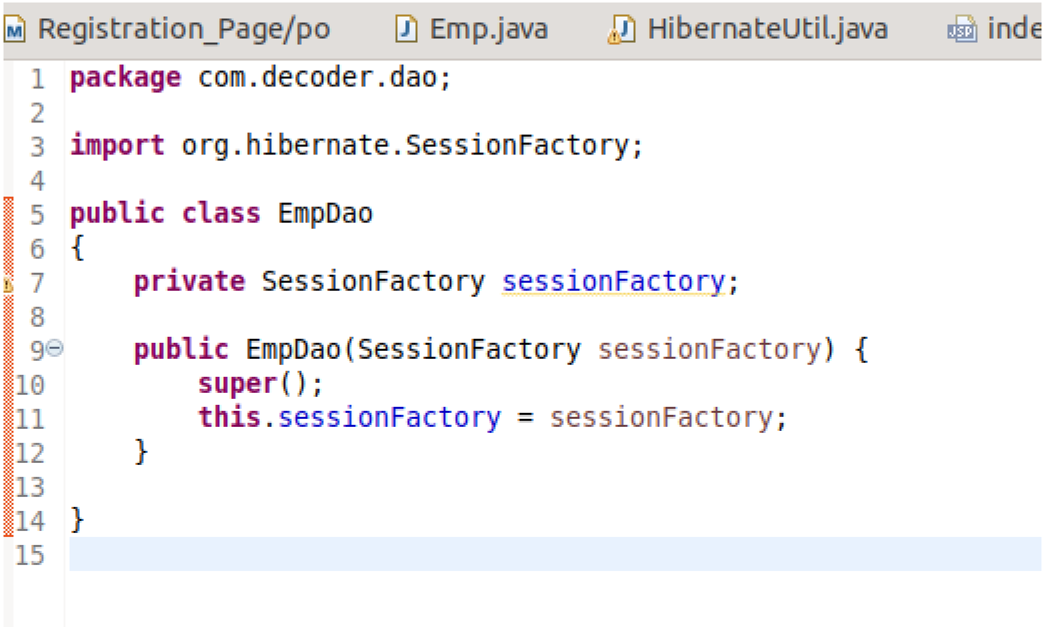
  Register

  >

- 

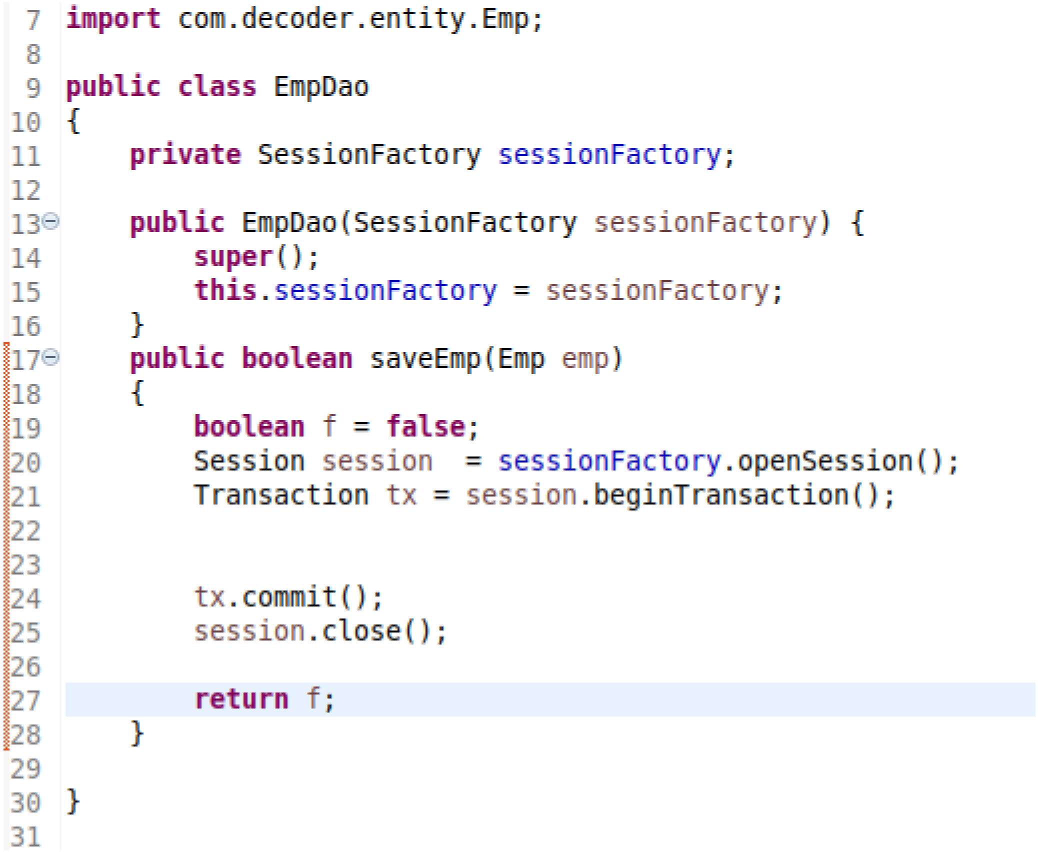
```
Dec 23, 2023 12:02:34 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/8.5.92]
Dec 23, 2023 12:02:34 PM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this log
Dec 23, 2023 12:02:34 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8080"]
Dec 23, 2023 12:02:34 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 584 ms
Emp [id=0, name=Nilesh, department=IT, salary=120000, email=nilesh@gmail.com, password=1210]
```

- We are getting data from the front end, so we need to save that data in DB.
- Create an “EmpDao” class file in the Dao package
- All operations like save, delete, and update are written in an EmpDao file.
- Create variable as Session factory and create parameterized constructor pass session factor object,

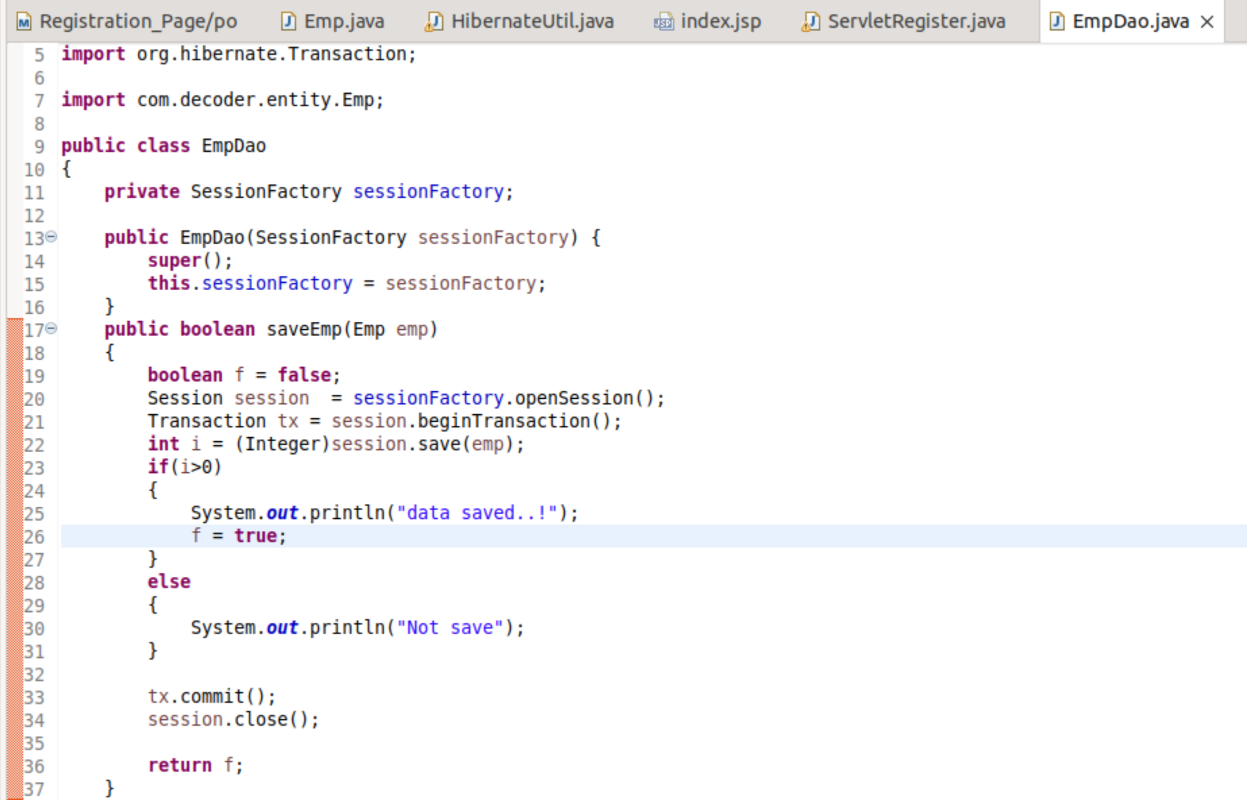


```java
package com.decoder.dao;

import org.hibernate.SessionFactory;

public class EmpDao
{
    private SessionFactory sessionFactory;

    public EmpDao(SessionFactory sessionFactory) {
        super();
        this.sessionFactory = sessionFactory;
    }

}

```

- Now, create a save method that returns an object; the way contains the emp thing.
- In the EmpDao.class file, we made a save method, so to save objects in Db, we have to use all configurations we used in the standard hibernate structure.
- We are not returning anything in the save method, so we created a boolean save method for simplicity.

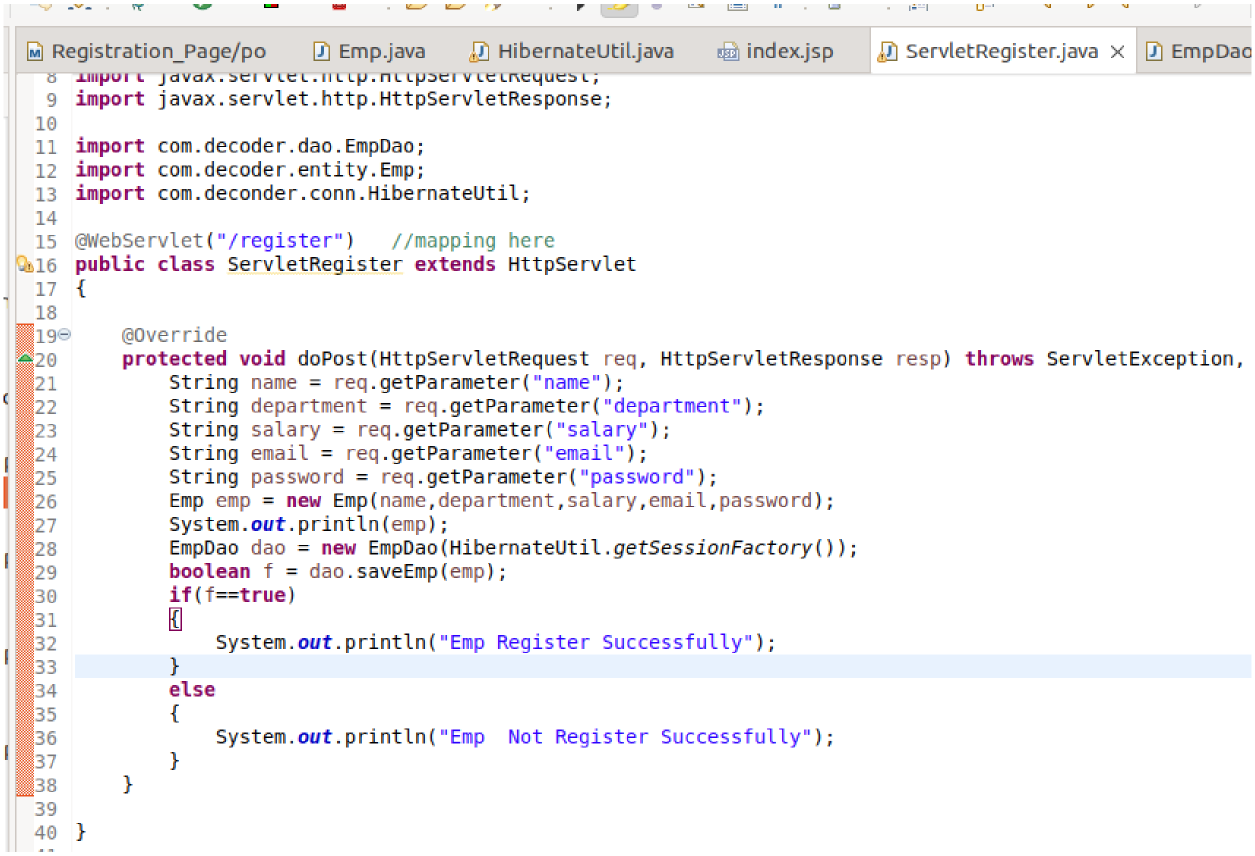


```java
import com.decoder.entity.Emp;

public class EmpDao
{
    private SessionFactory sessionFactory;

    public EmpDao(SessionFactory sessionFactory) {
        super();
        this.sessionFactory = sessionFactory;
    }
    public boolean saveEmp(Emp emp)
    {
        boolean f = false;
        Session session  = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();


        tx.commit();
        session.close();

        return f;
    }

}

```

- 

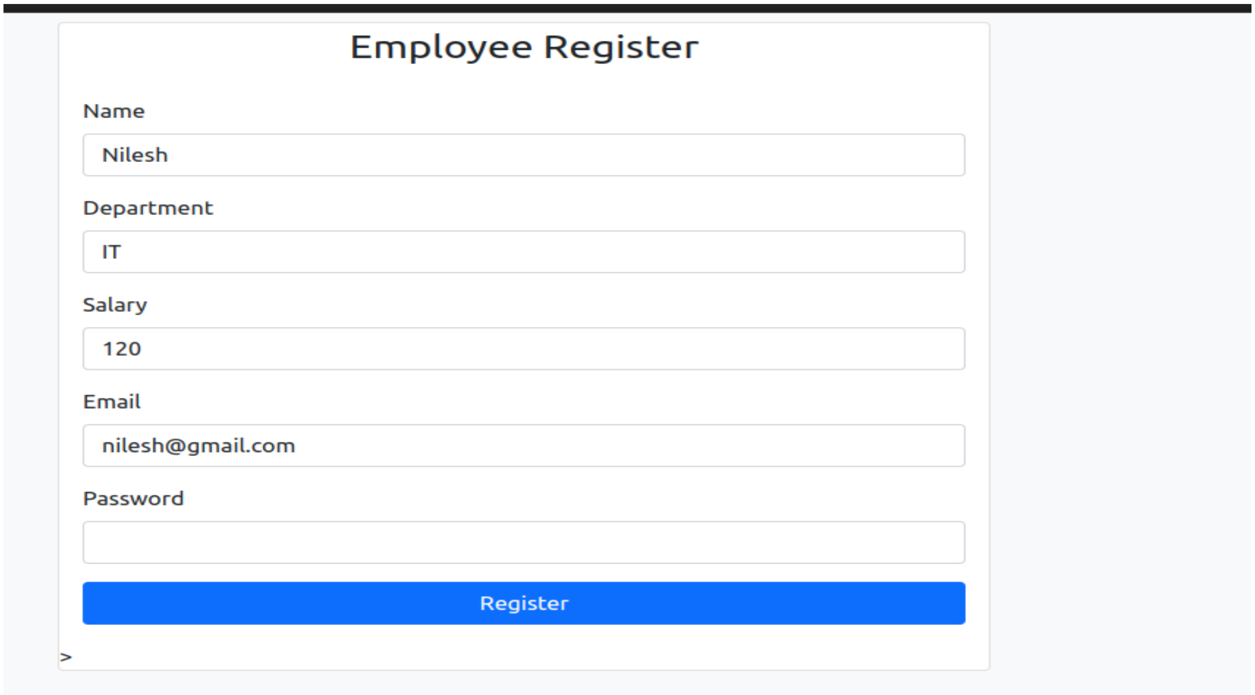
```java
import org.hibernate.Transaction;

import com.decoder.entity.Emp;

public class EmpDao
{
    private SessionFactory sessionFactory;

    public EmpDao(SessionFactory sessionFactory) {
        super();
        this.sessionFactory = sessionFactory;
    }
    public boolean saveEmp(Emp emp)
    {
        boolean f = false;
        Session session  = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        int i = (Integer)session.save(emp);
        if(i>0)
        {
            System.out.println("data saved..!");
            f = true;
        }
        else
        {
            System.out.println("Not save");
        }

        tx.commit();
        session.close();

        return f;
    }
```

-
- Now, save value in DB we have to call this method in ServletRegister class,
- Create an object of EmpDao → by passing the session-factory object from the HIbernateUnit class, because we need to create connect as well.
- And call save method of EmpDao class.



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

import com.decoder.dao.EmpDao;
import com.decoder.entity.Emp;
import com.deconder.conn.HibernateUtil;

@WebServlet("/register")   //mapping here
public class ServletRegister extends HttpServlet
{

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
        String name = req.getParameter("name");
        String department = req.getParameter("department");
        String salary = req.getParameter("salary");
        String email = req.getParameter("email");
        String password = req.getParameter("password");
        Emp emp = new Emp(name,department,salary,email,password);
        System.out.println(emp);
        EmpDao dao = new EmpDao(HibernateUtil.getSessionFactory());
        boolean f = dao.saveEmp(emp);
        if(f==true)
        {
            System.out.println("Emp Register Successfully");
        }
        else
        {
            System.out.println("Emp  Not Register Successfully");
        }
    }

}
```

Now run index.jsp file → Enter data → check in DB



Employee Register

Name

Nilesh

Department

IT

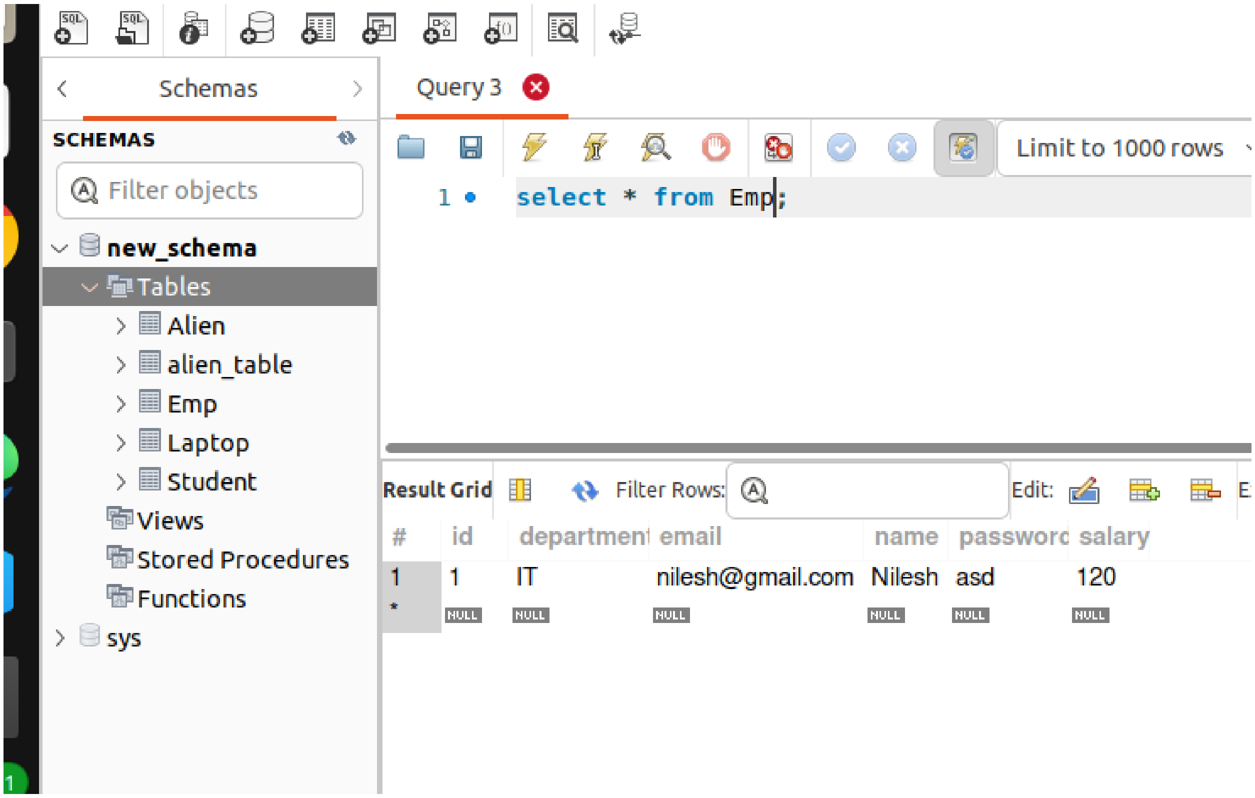
Salary

120

Email

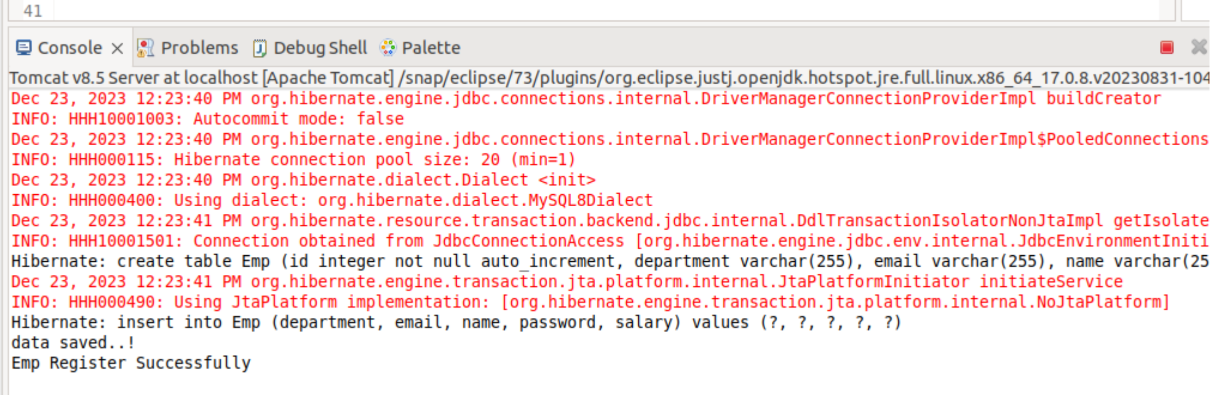
nilesh@gmail.com

Password



```
select * from Emp;
```

#	id	departmen	email	name	passworc	salary
1	1	IT	nilesh@gmail.com	Nilesh	asd	120
*	NULL	NULL	NULL	NULL	NULL	NULL



```
Tomcat v8.5 Server at localhost [Apache Tomcat] /snap/eclipse/73/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.8.v20230831-104
Dec 23, 2023 12:23:40 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Dec 23, 2023 12:23:40 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
Dec 23, 2023 12:23:40 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
Dec 23, 2023 12:23:41 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolate
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentIniti
Hibernate: create table Emp (id integer not null auto_increment, department varchar(255), email varchar(255), name varchar(25
Dec 23, 2023 12:23:41 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate: insert into Emp (department, email, name, password, salary) values (?, ?, ?, ?, ?)
data saved..!
Emp Register Successfully
```

Final index.jsp code:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmL
ASjC" crossorigin="anonymous">
</head>
<body class="bg-light">
      <div class="container">
            <div class="row">
                  <div class="col-md-6 offset-md-3 mt-2">
                        <div class="card">
                        <div class="crad-header text-center fs-3">
                              Employee Register
                        </div>
                              <div class="card-body">
                                    <form method="post" action="register">
                                            <div class="mb-3 mt-2">
                                              <label for="exampleInputEmail1"
class="form-label">Name</label>
                                              <input type="text"
class="form-control" name="name">
                                            </div>
                                            <div class="mb-3 mt-2">
                                              <label for="exampleInputEmail1"
class="form-label">Department</label>
                                              <input type="text"
class="form-control" name="department">
                                            </div>
                                            <div class="mb-3 mt-2">
                                              <label for="exampleInputEmail1"
class="form-label">Salary</label>
                                              <input type="text"
class="form-control" name="salary">
                                            </div>
                                            <div class="mb-3 mt-2">
                                              <label for="exampleInputEmail1"
class="form-label">Email</label>
```

```
                                               <input type="email"
class="form-control" name="email">
                                             </div>
                                              <div class="mb-3 mt-2">
                                               <label for="exampleInputEmail1"
class="form-label">Password</label>
                                               <input type="password"
class="form-control" name="password">
                                             </div>
                                             <button type="submit" class="btn
btn-primary col-md-12" >Register</button>
                                           </form>
                               </div>
                         </div>
                   </div>
             </div>
      </div>
</body>
</html>
```