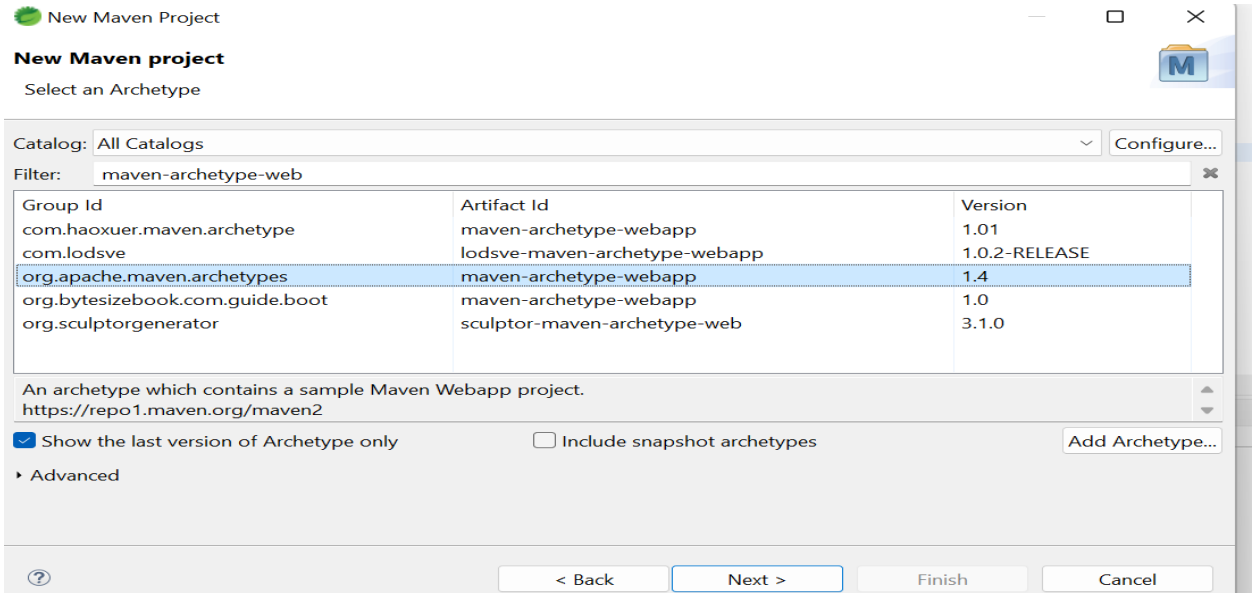
SPRING MVC Using Hibernate-ORM tool

- First Create database, use database,create table
- Open a new maven project
- Do not select checkbox for "create a simple project". Just click next.
- Then filter on "maven-archetype-webapp"
- Select "org.apache.maven.archetypes" – "maven-archetype-webapp"-1.4 version
- Note: make sure version should be 1.4



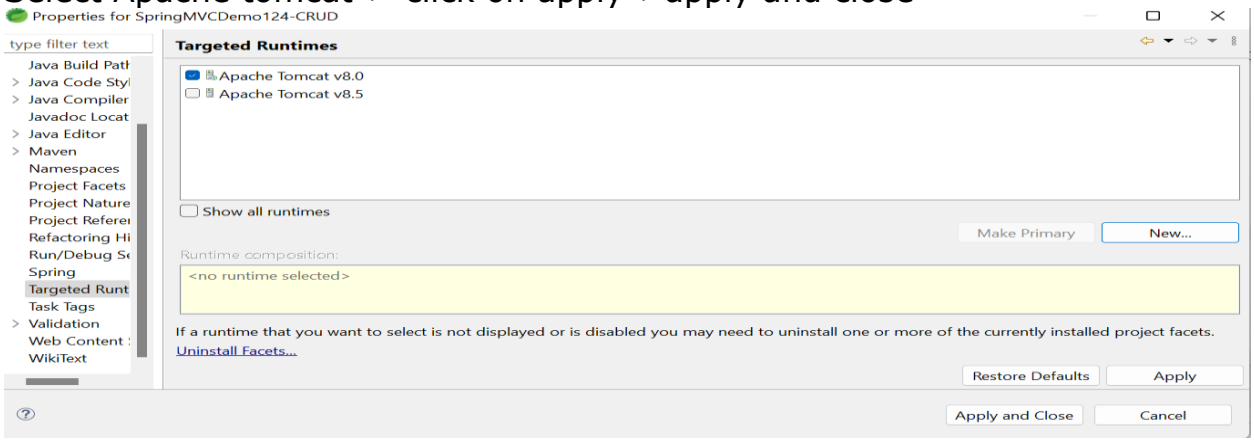
- After successfully creating project, set jre1.8 by right clicking on project select build path option->configure build path option->libraries tab->click on add libraray button->select jre system option->click on Installed jre->click on add button->select standard VM->click next->select directory->click on finish->apply->apply and close->apply->apply and close.

- Add following dependency to the pom.xml file.

```
<dependencies>
<dependency>
	<groupId>junit</groupId>
	<artifactId>junit</artifactId>
	<version>4.11</version>
	<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-core
-->
<dependency>
```

```
                <groupId>org.springframework</groupId>
                <artifactId>spring-core</artifactId>
                <version>5.3.25</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.springframework/spring-
context -->
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.25</version>
            </dependency>


            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-webmvc</artifactId>
                <version>5.3.25</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-
servlet-api -->
            <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat-servlet-api</artifactId>
                <version>10.1.0-M16</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-orm</artifactId>
                <version>5.3.25</version>
            </dependency>
            <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>5.6.8.Final</version>
            </dependency>
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>8.0.29</version>
            </dependency>
            <dependency>
                <groupId>jstl</groupId>
                <artifactId>jstl</artifactId>
                <version>1.2</version>
            </dependency>
            <dependency>
                <groupId>taglibs</groupId>
                <artifactId>standard</artifactId>
                <version>1.1.2</version>
            </dependency>

        </dependencies>
```

- Inregrate apache tomcat server with our project, right click on project->build path->configure build path->select targeted runtime option->click on new button->select apache tomcat version 8.5 or 8.0(check in your system apache tomcat version then select that appropriate version)->click next->brouse apache tomcat directory->click on finish
- Select Apache tomcat-> click on apply->apply and close



- Steps in creating Spring MVC application
  - Configure the dispatcher servlet (add following tags to web.xml)
  - Note->Explicitly we have to add web-app tags here which is highlighted in green color.

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
	   http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
	version="3.0">


	<servlet>
		<servlet-name>dispatcher</servlet-name>
		<servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
	</servlet>

	<servlet-mapping>
		<servlet-name>dispatcher</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>

</web-app>
```

  - Create spring configuration (follow name convention)
    - add "<servlet-name>-servlet.xml" under WEB-INF
    - e.g. "dispatcher-servlet.xml"
  - In "dispatcher-servlet.xml" file

- Configure the view resolver
- Configure Sessionfactory
- Add below configuration code

```
<?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:context="http://www.springframework.org/schema/conte
xt"
	xmlns:p="http://www.springframework.org/schema/p"
	xmlns:c="http://www.springframework.org/schema/c"
	xmlns:tx="http://www.springframework.org/schema/tx"
	xsi:schemaLocation="http://www.springframework.org/schema/
beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-
context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">


	<context:component-scan
		base-package="com.psl.training" />
	<tx:annotation-driven />

	<bean

	class="org.springframework.jdbc.datasource.DriverManagerDa
taSource"
		name="dataSource"
p:driverClassName="com.mysql.jdbc.Driver"
		p:url="jdbc:mysql://localhost/user123"
p:username="root"
		p:password="root" />

	<bean

	class="org.springframework.orm.hibernate5.LocalSessionFact
oryBean"
		name="sessionFactory" p:dataSource-ref="dataSource">
		<property name="hibernateProperties">
			<props>
				<prop
key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
				<prop
key="hibernate.show_sql">true</prop>
			</props>
		</property>
		<property name="annotatedClasses">
			<list>

	<value>com.psl.training.Entity.User</value>
			</list>
		</property>
```

```
		</bean>

		<bean

		class="org.springframework.orm.hibernate5.HibernateTransac
	tionManager"
			name="transactionManager" p:sessionFactory-
	ref="sessionFactory" />

		<bean

		class="org.springframework.web.servlet.view.InternalResour
	ceViewResolver"
			name="viewResolver">
			<property name="prefix">
				<value>/WEB-INF/views/</value>
			</property>
			<property name="suffix">
				<value>.jsp</value>
			</property>
		</bean>
	</beans>
```

- Create controller
  - Inside src/main/java create package and add class MyController.java and marked that class with @Controller
  - Add method to return ModelAndView

```
package com.psl.training;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import com.psl.training.Entity.User;
import com.psl.training.Service.UserService;

@Controller
```

```
public class UserController {

	@Autowired
	private UserService service;

	@RequestMapping("registrationUser")
	// @RequestMapping(value = "/registrationUser", method =
RequestMethod.GET)
	public ModelAndView showRegistrationPage() {
		ModelAndView m = new ModelAndView();
		m.setViewName("UserReg");
		return m;
	}

	// @PostMapping("registerUser")
	@RequestMapping(value = "registerUser", method =
RequestMethod.POST)
	public String registerUser(@ModelAttribute("user") User user,
ModelMap model) {
		service.save(user);// insert data
		model.addAttribute("result", "User inserted in to database
table");
		return "UserReg";
	}

	@RequestMapping("getUsers")
	public String getUser(ModelMap model) {
		List<User> users = service.getUsers();// select
		model.addAttribute("users", users);
		return "displayUsers";
	}

	@RequestMapping("delete")
	public String getUser(@RequestParam("id") int id) {
		service.deleteEmployee(id);
		return "message";
	}

}
```

- **Create View folder**
  - Right click on WEB-INF folder and create new folder->give name view
  - Under view folder create .jsp files.

- **Displayusers.jsp**

```
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <table border="1">
        <tr>
            <th>id</th>
            <th>name</th>
            <th>email</th>
        </tr>
        <c:forEach items="${users}" var="user">
            <tr>

                <td>${user.id}</td>
                <td>${user.name}</td>
                <td>${user.email}</td>



            </tr>
        </c:forEach>

    </table>
</body>
</html>
```

- **Registrationuser.jsp**

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">

<title>Insert title here</title>


</head>
<body>
    <h2>Registration page</h2>
    <form action="registerUser" method="POST">
            <pre>
Id: <input type="text" name="id" id="id" />
Name: <input type="text" name="name" />
Email: <input type="text" name="email" />
<input type="submit" name="register" />
</pre>

    </form>
<br />${result}


</body>
</html>
```

- **Message.jsp**

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>Data deleted successfully</h2>
</body>
</html>
```

- **Create Entity/Model class**

```
package com.psl.training.Entity;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="userdata")
public class User {

    @Id
    private int id;
    private String name;
    private String email;
    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 getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ",
email=" + email + "]";
    }


    }
```

- **Create DAO Layer**
- **Interface**

```
package com.psl.training.DAO;

import java.util.List;
import com.psl.training.Entity.User;
```

```
	public interface UserDAO {

		public boolean create(User user);//insert-- create

		public List<User> readAll();//select--read

		public void deleteEmployee(int id);


	}
```

Class to implement interface

```
package com.psl.training.DAO;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.orm.hibernate5.HibernateTemplate;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import com.psl.training.Entity.User;

@Repository
public class UserDAOImpl implements UserDAO {

	@Autowired
	SessionFactory sessionFactory;

	public boolean create(User user) {
		Session session = sessionFactory.openSession();
		session.beginTransaction(); // transaction started
		session.saveOrUpdate(user);// insert query
		session.getTransaction().commit();
		return true;
	}

	public List<User> readAll() {
		Session session = sessionFactory.openSession();
		return session.createQuery("from User database", User.class).list();
	}

	public void deleteEmployee(int id) {
		Session session = sessionFactory.openSession();
		User e = session.get(User.class, id);
		session.beginTransaction();
		session.delete(e);
		session.getTransaction().commit();

	}
}
```

- **Create Service Layer**

Service interface

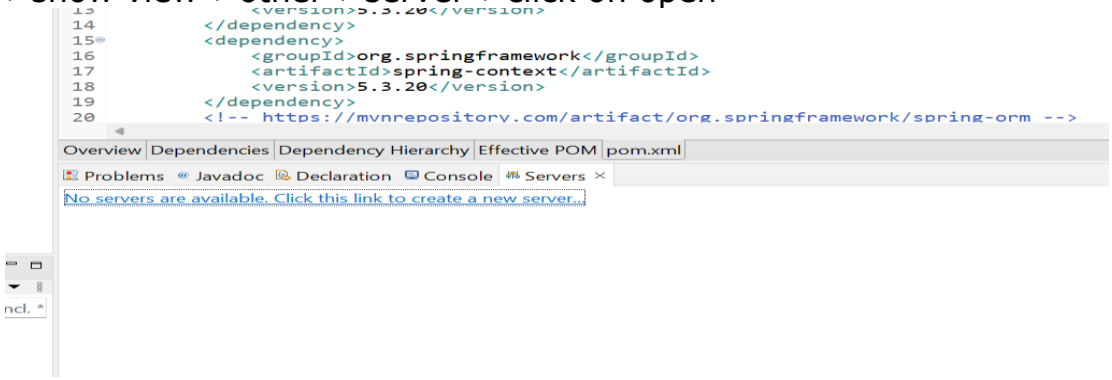
```
package com.psl.training.Service;

import java.util.List;
import com.psl.training.Entity.User;

public interface UserService {

	public boolean save(User user);// insert

	public List<User> getUsers();// select

	public void deleteEmployee(int id);

}
```

Class to implement service

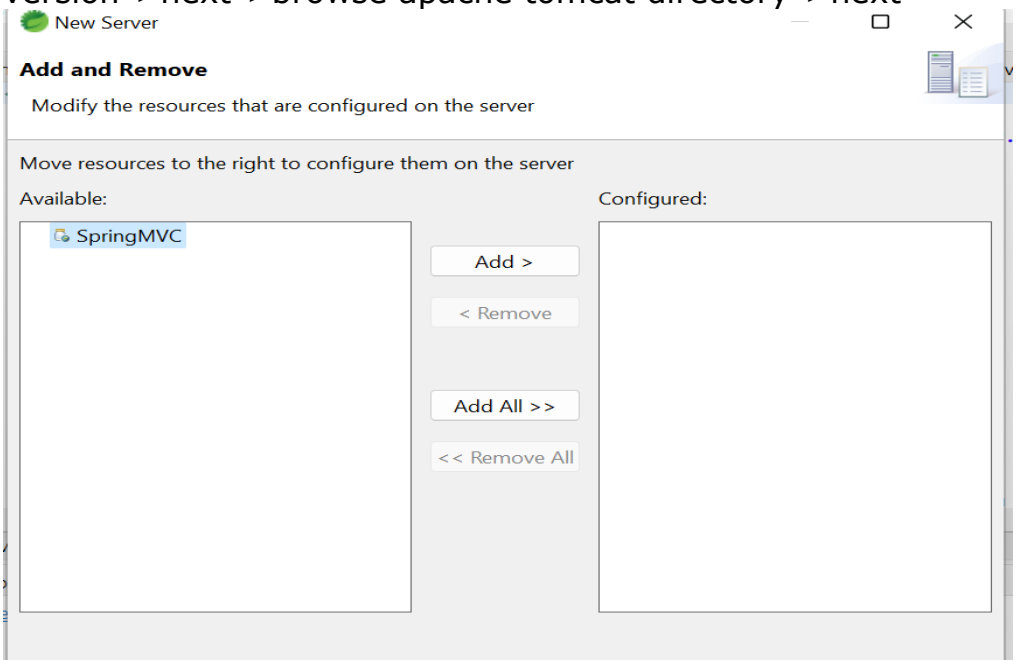
```
package com.psl.training.Service;

import java.util.List;
import javax.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.psl.training.DAO.UserDAO;
import com.psl.training.Entity.User;

@Service
public class UserServiceImpl implements UserService{

	@Autowired
	UserDAO dao;


	public boolean save(User user) {
		//business logic
		return dao.create(user);

	}
	public List<User> getUsers() {

		return dao.readAll();
	}

	public void deleteEmployee(int id) {
		 dao.deleteEmployee(id);

	}
```

}

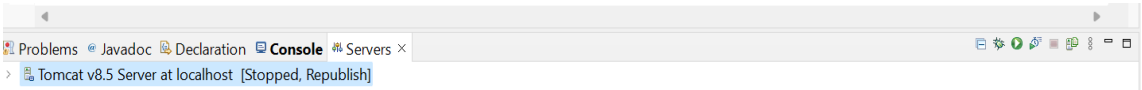
- To run code, need to add server on console. Go to Windows->show view->other->server->click on open



Click on above blue color link->select apache tomcat version->next->browse apache tomcat directory->next



Click on add->finish



You will get server on console window-- just start server here

Once server started successfully->hit below url on any browser
1.Loacthost:8080/"your project name"/ `registrationUser` -> here you will get registration page to insert data
2.Loacthost:8080/"your project name"/getUsers→here you will get table data
3. Loacthost:8080/"your project name"/delete?id=1;

- If run on server option present then right click on project -> run on server
- Select Apache tomcat v8 or v8.5 whichever is installed
- Click finish
- If -> error about port 8005 or 8080 in use
  - Double click server which is available in console window
  - Change port number
- Re-run application