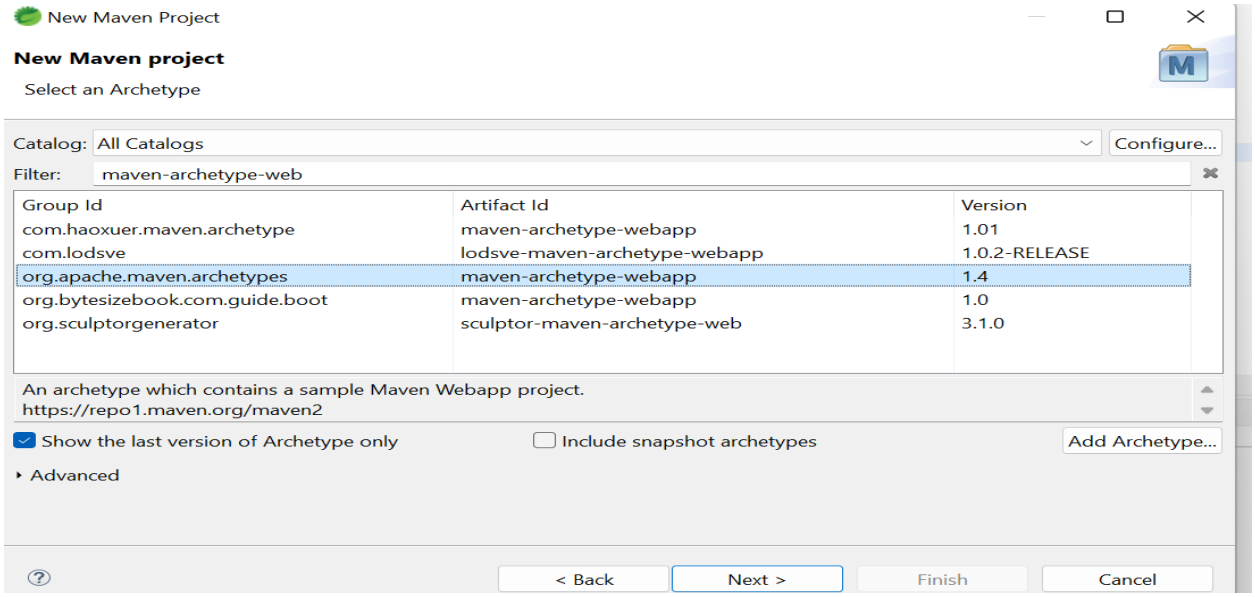
SPRING MVC Using JDBCTemplate

- First Create database, use database and 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



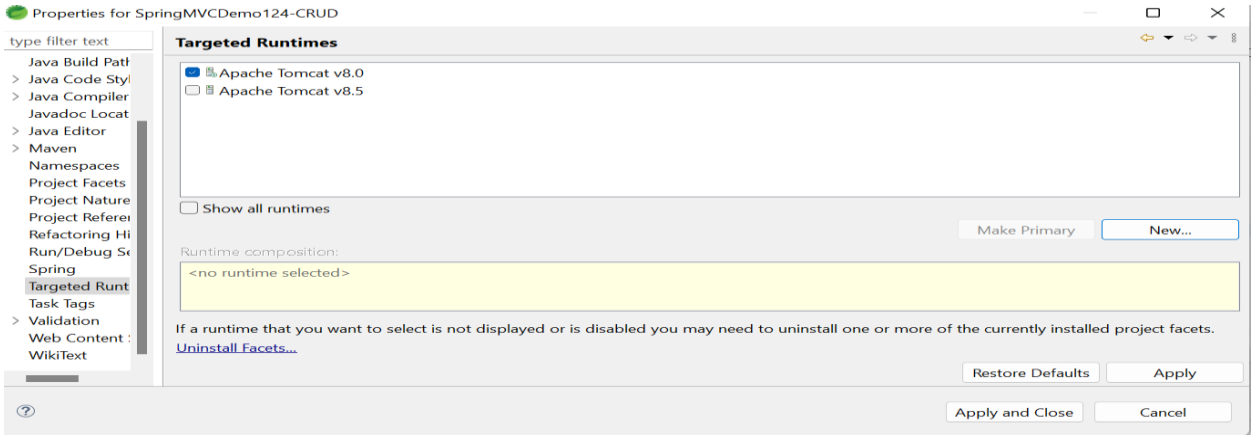
- After successfully creating project, set jre1.8 by right clicking on project select build path option->configure build path option->libraries tab->click on add libraray button->select jre system option->click on Installed jre->click on add button->select standard VM->click next->select directory->click 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>
<dependency>
	<groupId>org.springframework</groupId>
	<artifactId>spring-core</artifactId>
```

```
		<version>5.3.25</version>
	</dependency>

	<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>
	 <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-jdbc</artifactId>
		<version>5.3.25</version>
	</dependency>
	<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.28</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 JDBCTemplate
        - 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/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    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">

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

    <bean name="dataSource"

    class="org.springframework.jdbc.datasource.DriverManagerDataSourc
    e"
          p:driverClassName="com.mysql.cj.jdbc.Driver"
          p:url="jdbc:mysql://localhost:3306/empdetails"
    p:username="root"
          p:password="root">
    </bean>

    <bean name="jdbctemplate"
          class="org.springframework.jdbc.core.JdbcTemplate"
          p:dataSource-ref="dataSource">
    </bean>

    <bean

    class="org.springframework.web.servlet.view.InternalResourceViewR
    esolver"
          name="viewResolver">
          <property name="prefix" value="/WEB-INF/view/">
          </property>
          <property name="suffix" value=".jsp">
          </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.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import com.psl.training.model.Employee;
import com.psl.training.service.EmployeeService;
```

```java
@Controller
public class Mycontroller {

	@Autowired
	EmployeeService service;

	@RequestMapping("registration")
	public ModelAndView gethomePage()
	{
		ModelAndView model=new ModelAndView();
		model.setViewName("empregistartion");
		return model;

	}

	@RequestMapping(value = "registerEmp", method = RequestMethod.POST)
	public String registerEmp(@ModelAttribute("emp") Employee emp) {
		service.createEmp(emp);
		return "message";
	}

	@RequestMapping("getEmps")
	public String getUser(ModelMap model) {
		List<Employee> emps = service.readEmp();// select
		model.addAttribute("emps", emps);
		return "displayUsers";
	}



}
```

- 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="${emps}" var="emp">
                <tr>
                    <td>${emp.id}</td>
                    <td>${emp.name}</td>
                    <td>${emp.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="registerEmp" method="POST">
        <pre >
Id: <input type="text" name="id" id="id" /><br>
Name: <input type="text" name="name" /><br>
Email: <input type="text" name="email" /><br>
```

```
<input type="submit" name="register" />
</pre>

    </form>


</body>
</html>
```

- **Message.jsp**

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>Emp inserted successfully</h2>
</body>
</html>
```

- **Create Entity/Model class**

```
package com.psl.training.model;

public class Employee {

	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 "Employee [id=" + id + ", name=" + name + ", email=" +
email + "]";
	}

}
```

- Create DAO Layer
- **Interface**

```
package com.psl.training.dao;

import java.util.List;
import com.psl.training.model.Employee;

public interface EmployeeDAO {
	void create(Employee emp);// insert record

	List<Employee> readall();// read all employees

	void update(Employee emp);// insert record

	void select(int id);// read single record

}
```

- **Class to implement interface**

```
package com.psl.training.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import com.psl.training.model.Employee;

@Repository
public class EmployeeDAOImpl implements EmployeeDAO {

	@Autowired
	JdbcTemplate jdbctemplate;
```

```
	@Override
	public void create(Employee emp) {
		String sql = "insert into emp values(?,?,?)";
		jdbctemplate.update(sql, emp.getId(), emp.getName(),
emp.getEmail());

	}

	@Override
	public List<Employee> readall() {
		EmployeeRowMapperImpl r = new EmployeeRowMapperImpl();
		String sql = "select * from emp";
		ArrayList<Employee> list = (ArrayList<Employee>)
jdbctemplate.query(sql, r);
		return list;
	}

	@Override
	public void update(Employee emp) {
		// TODO Auto-generated method stub

	}

	@Override
	public Employee select(int id) {
		return null;
	}

}// end of class

final class EmployeeRowMapperImpl implements RowMapper<Employee> {

	@Override
	public Employee mapRow(ResultSet rs, int rowNum) throws SQLException {
		Employee e = new Employee();
		e.setId(rs.getInt(1));
		e.setName(rs.getString(2));
		e.setEmail(rs.getString(3));
		return e;
	}

}
```

- Create Service Layer

  Service interface
  package com.psl.training.service;

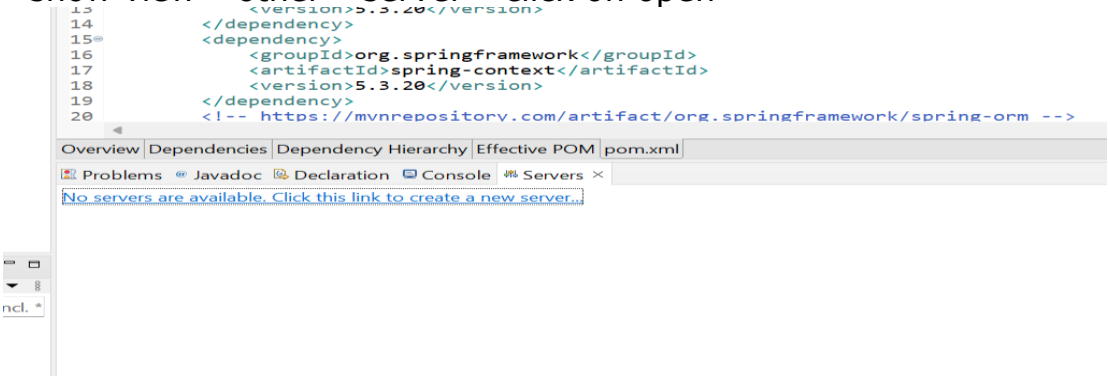
  import java.util.List;

```
import com.psl.training.model.Employee;

public interface EmployeeService {
    void createEmp(Employee emp);// insert record

    List<Employee> readEmp();// read all employees

    void updateEmp(Employee emp);// insert record

    Employee selectEmpbyId(int id);// read single record
}
```

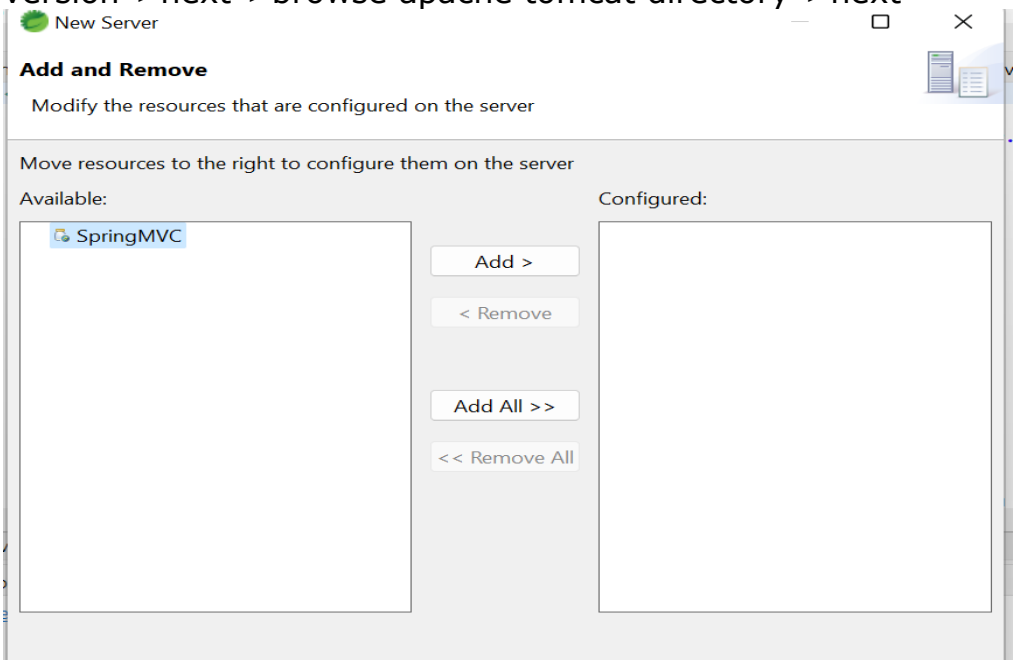
Class to implement service

```
package com.psl.training.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.psl.training.dao.EmployeeDAO;
import com.psl.training.model.Employee;

@Service
public class EmployeeServiceImpl implements EmployeeService {

	@Autowired
	EmployeeDAO dao;

	public void createEmp(Employee emp) {
		dao.create(emp);

	}

	public List<Employee> readEmp() {
		return dao.readall();
	}

	public void updateEmp(Employee emp) {
		// TODO Auto-generated method stub

	}

	public Employee selectEmpbyId(int id) {
		return null;

	}

}
```

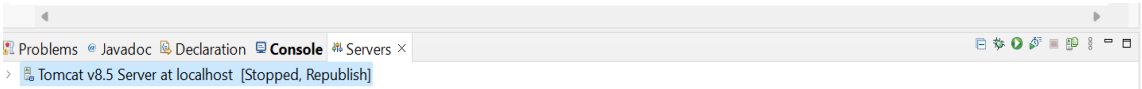
- To run code, need to add server on console. Go to Windows->show view->other->server->click on open



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"/registration-> here you will get registration page to insert data
2.Loacthost:8080/"your project name"/getEmps→here you will get table data

- 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