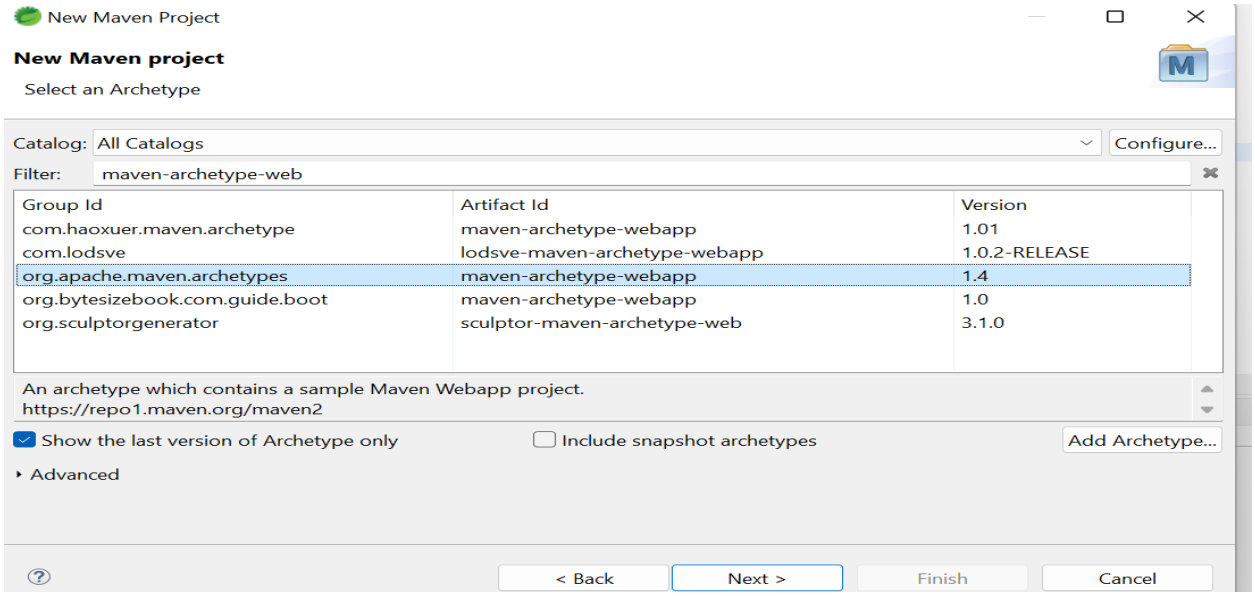
**SPRING MVC Using Collection**

- 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



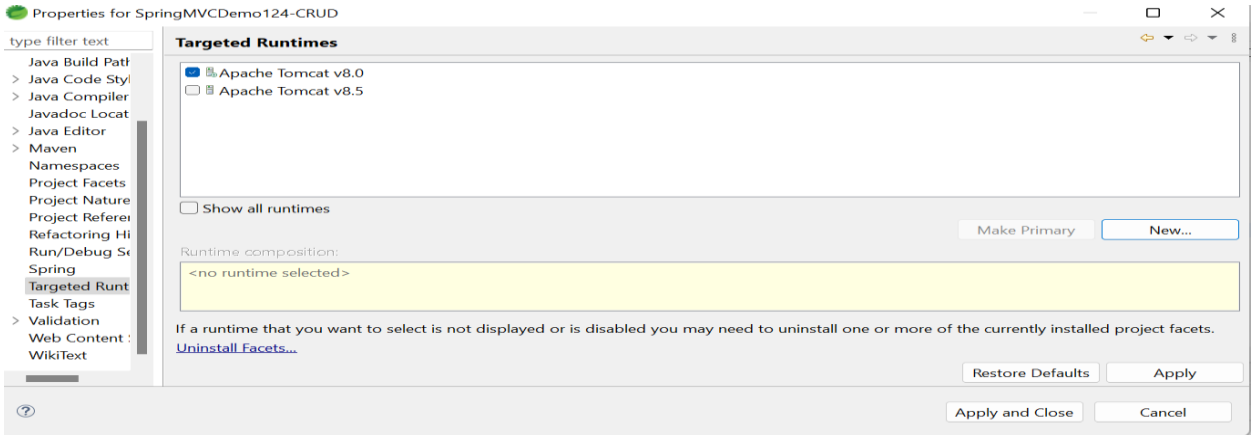

- After successfully creating project, set jre1.8 by right clicking on project select build path option->configure build path option->libraries tab->click on add libraray button->select jre system option->click on Installed jre->click on add button->select standard VM->click next->select directory->click 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>5.3.21</version>
            </dependency>
            <!--
https://mvnrepository.com/artifact/org.springframework/spring-context
-->
            <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-context</artifactId>
                  <version>5.3.21</version>
            </dependency>


            <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-webmvc</artifactId>
                  <version>5.3.20</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>taglibs</groupId>
                  <artifactId>standard</artifactId>
                  <version>1.1.2</version>
            </dependency>

      </dependencies>
```

- Integrate 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->browse 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
        - Create bean for required classes
        - 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/sch
ema/context"
      xmlns:p="http://www.springframework.org/schema/p"
      xmlns:c="http://www.springframework.org/schema/c"
      xsi:schemaLocation="http://www.springframework.or
g/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

      class="org.springframework.web.servlet.view.Inter
nalResourceViewResolver"
            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;
```

```java
import java.util.Map;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import
org.springframework.web.bind.annotation.DeleteMapping;
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.RequestBody;
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;
import com.psl.training.Service.UserServiceImpl;

@Controller
public class UserController // resource class
{
	@Autowired
	private UserService service;

	@RequestMapping("getUsers")
	public ModelAndView getUser(ModelMap model) {
		List<User> users = service.getUsers();
		model.addAttribute("users", users);
		ModelAndView m = new ModelAndView();
		m.setViewName("displayUsers");
		return m;
	}
```

```
	@RequestMapping("show/{id}")
	public String showusers(@PathVariable Long id,
ModelMap model) {
		{
			User users = service.findOne(id);
			model.addAttribute("users", users);
			return "displayUsers";
		}

	}

	@RequestMapping("newUsers")
	public ModelAndView getnewUsers(ModelMap model) {
		User users = service.createUser();
		model.addAttribute("users", users);
		ModelAndView m = new ModelAndView();
		m.setViewName("message");
		return m;
	}

}
```

- **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>
```

```
<h3>${users}</h3>
</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>User inserted successfully</h2>
</body>
</html>
```

- **Create Entity/Model class**

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

import org.springframework.stereotype.Component;

public class User {

   private Long user_id;
   private String user_name;

   public User(Long user_id, String user_name) {
         super();
         this.user_id = user_id;
         this.user_name = user_name;

   }

   public Long getUser_id() {
         return user_id;
   }

   public void setUser_id(Long user_id) {
         this.user_id = user_id;
   }

   public String getUser_name() {
         return user_name;
   }
```

```
    public void setUser_name(String user_name) {
        this.user_name = user_name;
    }

    @Override
    public String toString() {
        return "User [user_id=" + user_id + ",
user_name=" + user_name + "]";
    }

}
```

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

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

import java.util.List;

import com.psl.training.Entity.User;

public interface UserDAO {

    public List<User> ReadAll();//Read all users from
collection

    public User findOne(Long id);//find particular user from
collection

    public User create();//insert data into collection

}
```

**Class to implement interface**

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

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Repository;
import com.psl.training.Entity.User;

@Repository
public class UserDAOImpl implements UserDAO// Dao
class or service class
```

```
{

	private static List<User> list = new
LinkedList<User>();

	static {

		list.add(new User(3L, "Alex"));
		list.add(new User(1L, "Alen"));
		list.add(new User(2L, "Bhagyashri"));
		list.add(new User(4L, "Jack"));
	}

	@Override
	public List<User> ReadAll() {// Read opeartion
		return list;
	}

	public User findOne(Long id) {
		for (User user : list) {
			if (user.getUser_id().compareTo(id) ==
0) {
				return user;
			}

		}
		return null;

	}

	@Override
	public User create() {
		User u = new User(450000L, "Bhagyashri");
		list.add(u);

		return u;
	}

}
```

- **Create Service Layer**

**Service interface**

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

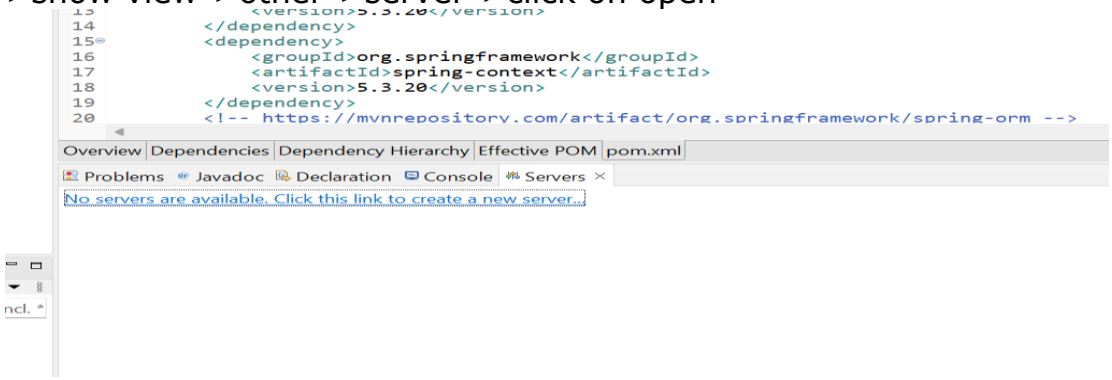
```java
import java.util.List;

import com.psl.training.Entity.User;

public interface UserService {

	public List<User> getUsers();

	public User findOne(Long id);

	public User createUser();
}
```

**Class to implement service**

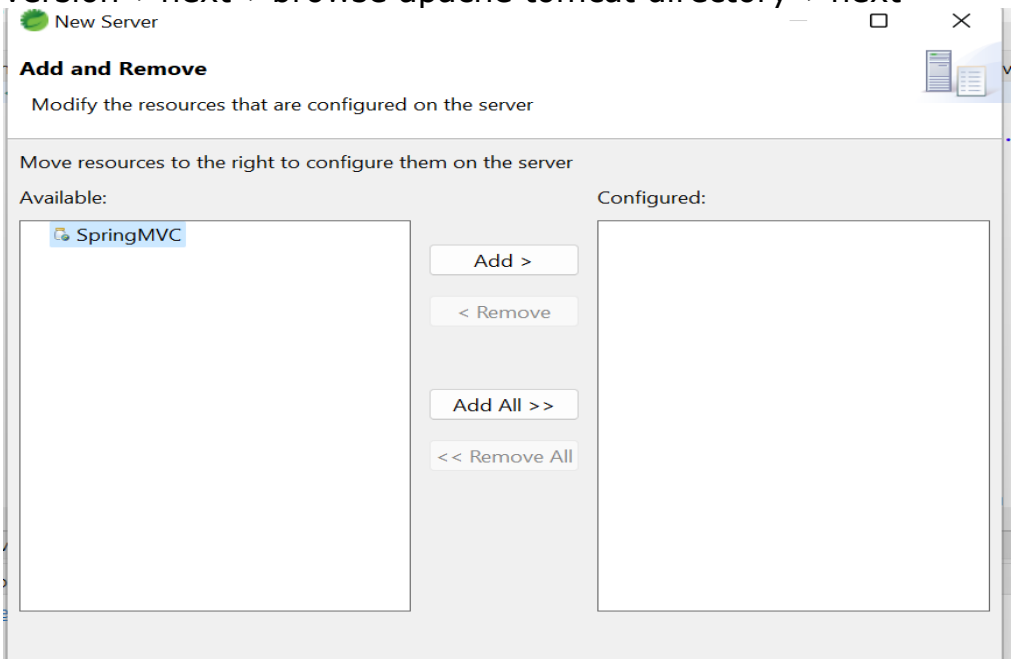
```java
package com.psl.training.Service;

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

@Service
public class UserServiceImpl implements UserService {

	@Autowired
	private UserDAO dao;

	@Override
	public List<User> getUsers() {
		// TODO Auto-generated method stub
		return dao.ReadAll();
	}

	@Override
	public User findOne(Long id) {
		// TODO Auto-generated method stub
		return dao.findOne(id);
	}

	@Override
	public User createUser() {
		// TODO Auto-generated method stub
```

```
        return dao.create();
    }

}
```

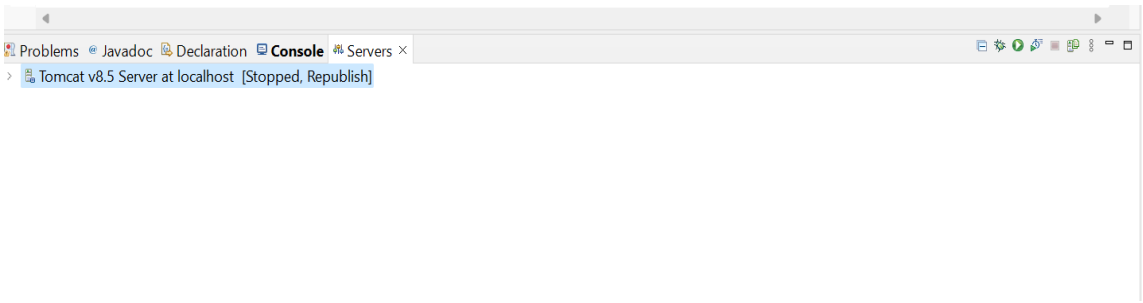
- To run code, need to add server on console. Go to Windows->show view->other->server->click on open




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"/ `getUsers` -> here you will get collection data
2.Loacthost:8080/"your project name"/show/101→here you will get data who's id=101
3. Loacthost:8080/"your project name"/newUsers;

- 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