Данный туториал направлен на ознакомление с технологией **servlet**, использованию **spring** **injection** в пределах **web** приложения.

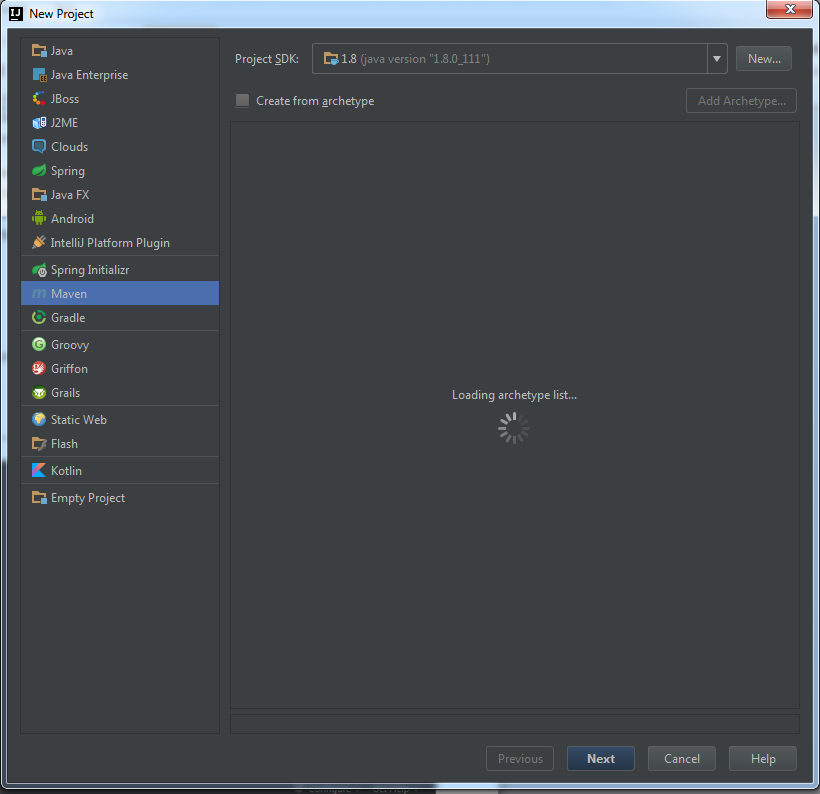
1. Создание веб-приложения
2. Деплой посредством встроенного веб-сервера jetty
3. Разработка model-service-controller системы
4. Spring inject -> конфигурация
5. Тестирование

Итак начнем с создания web приложения – для этого

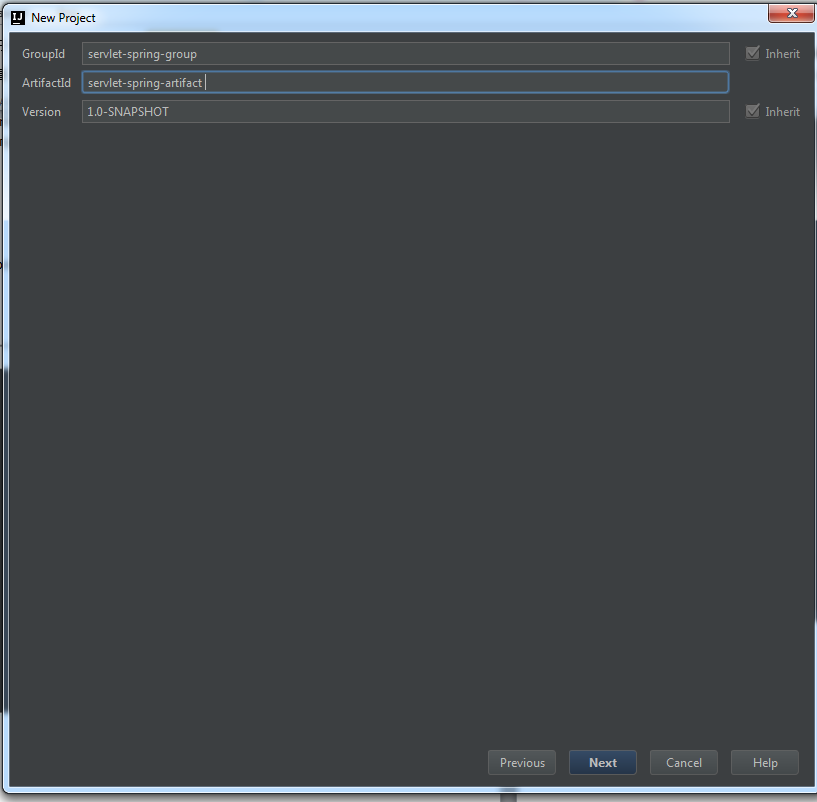
1. Запустим Intellij Idea



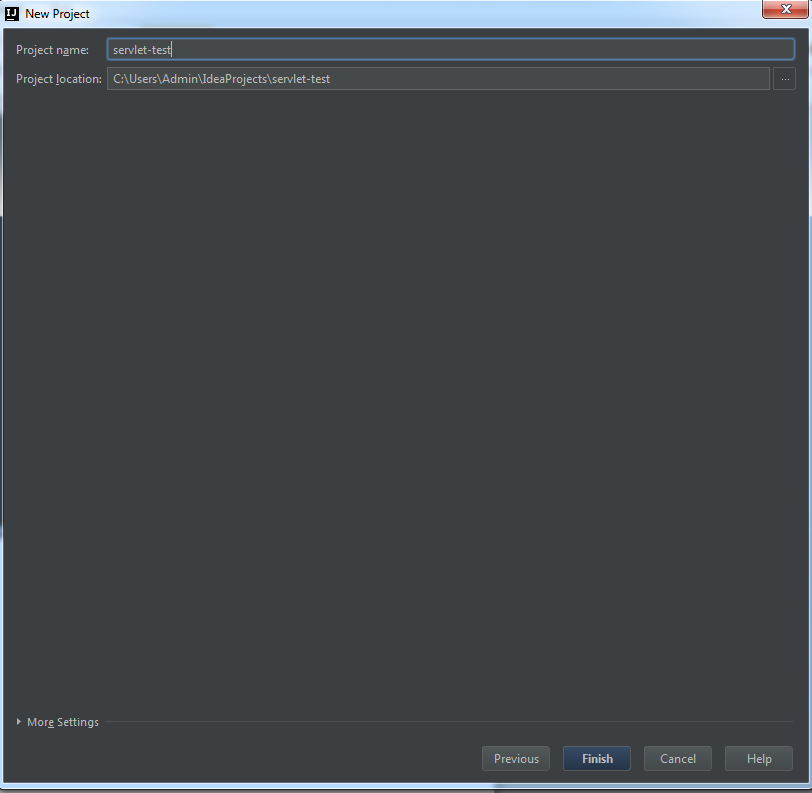
* Create New Project -> Maven



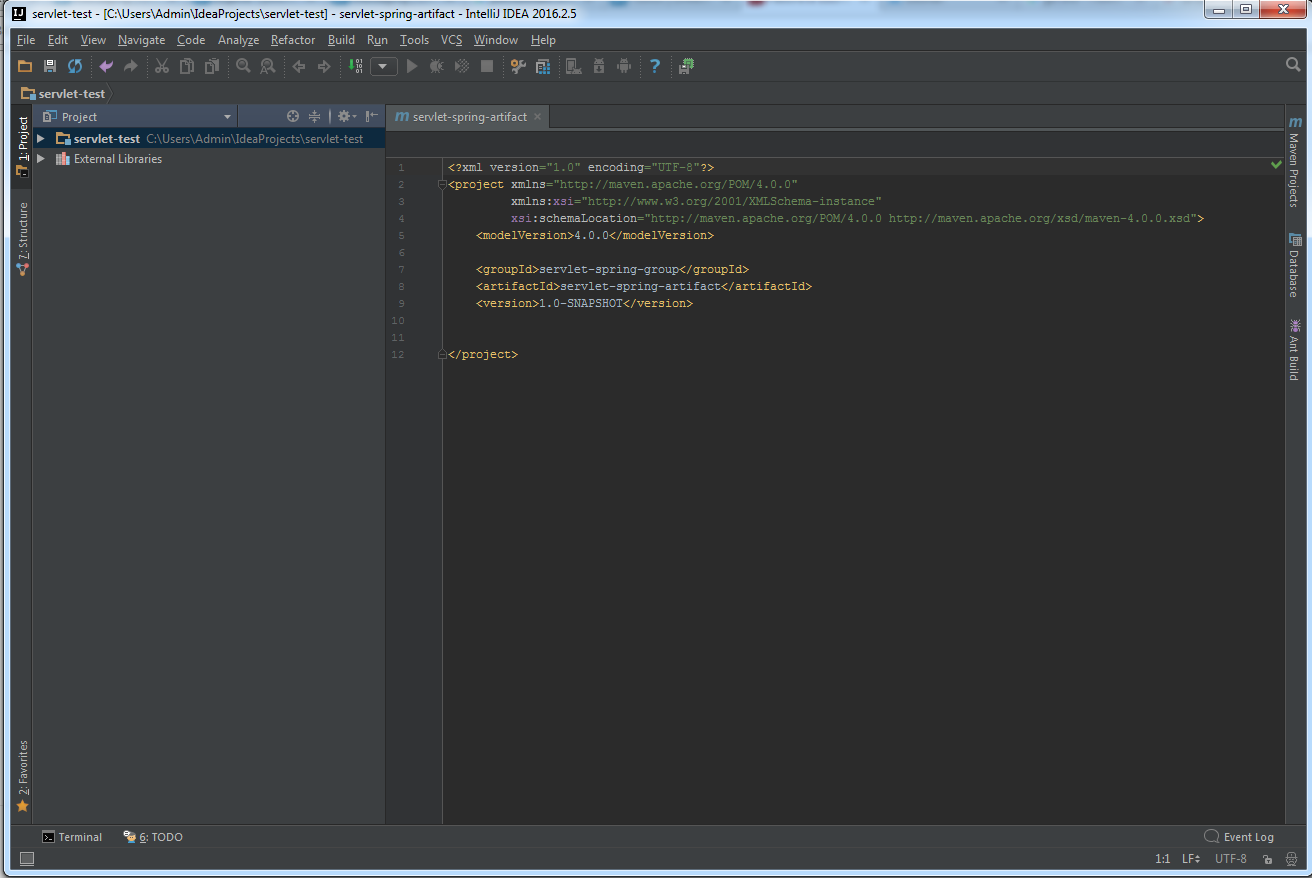
В качестве параметров для artifactId and groupId можно **указать servlet-spring-group** and **servlet-spring-artifact** соответственно



Укажите имя проекта:

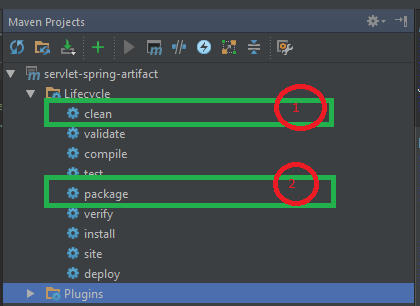


В результате должен получиться следующий вид проекта

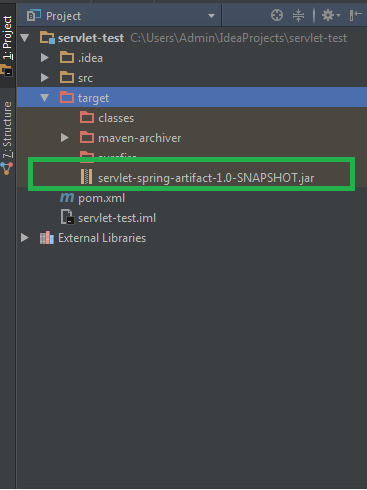


Прошу заметить – что на текущий момент это является обычным стандартным мавен проектом –> в результате сборки которого у нас получается jar

Cборка: **clean - package**

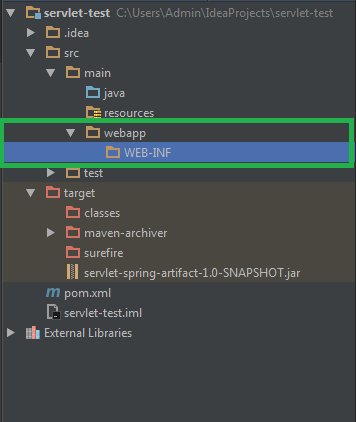


Результат сборки:



Создадим веб структуру – для этого добавим **webapp** and **web.xml** + поменяем тип упаковки в пределах ром хмл

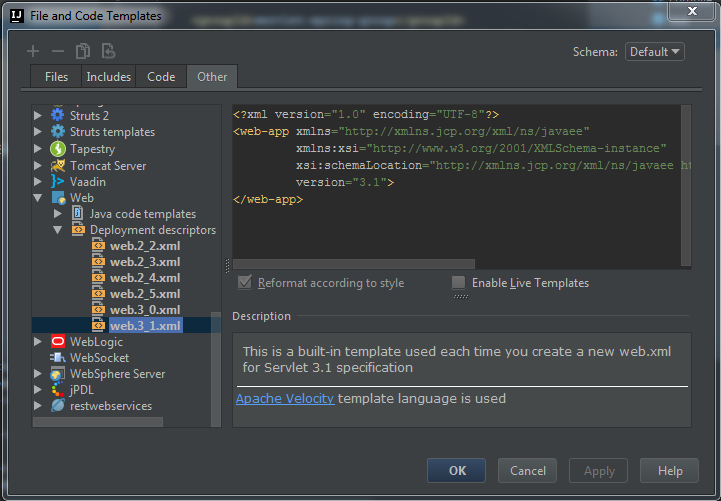
* src -> main -> webapp -> WEB-INF



* под **webapp-> WEB-INF** создайте web.xml

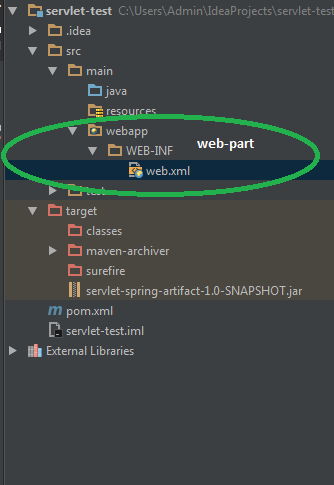
если у Вас Ultimate версия идеи – Вы можете создать web.xml на базе темплейта

правой кнопкой на папочке **WEB-INF -> Edit Files template -> tab Other -> web -> xsd(как вариант 3.0 или 3.1)**



Скопируйте и продублируйте в файлик webapp-> WEB-INF->web.xml

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  
 version="3.1">  
</web-app>

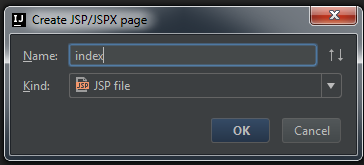


* pom.xml -> изменим packaging -> war

<?xml version="1.0" encoding="UTF-8"?>  
<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>servlet-spring-group</groupId>  
 <artifactId>servlet-spring-artifact</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>war</packaging>  
  
</project>

* В пределах **WEB-INF** создадим **-> index.jsp**

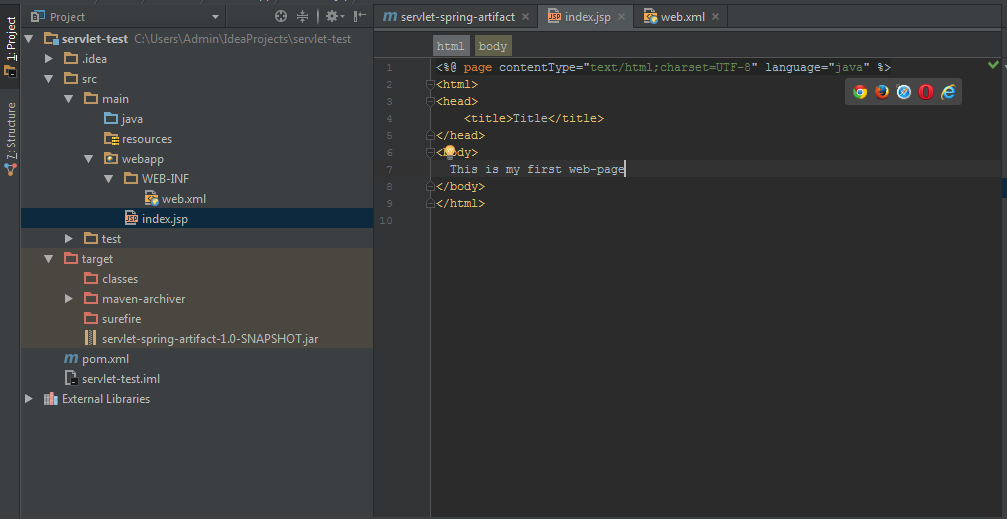
Правой кнопкой на **webapp** -> **File** **-> index.jsp**



Контент index.jsp:

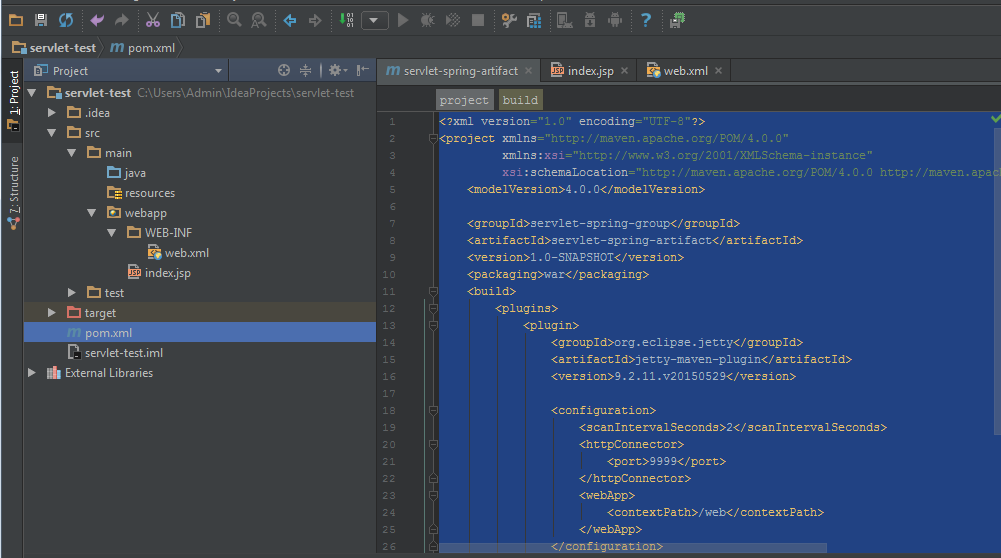
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
 This is my first web page)))  
</body>  
</html>

Общий вид проекта:



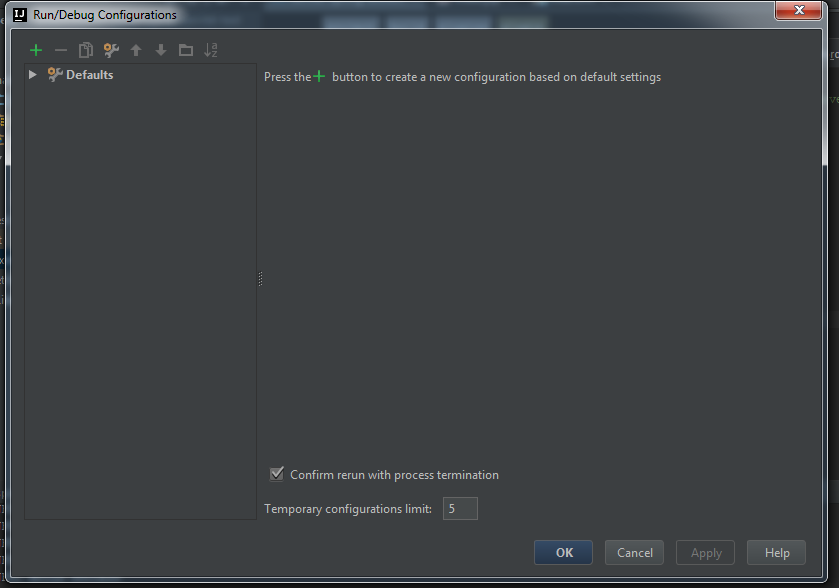
* Добавим в pom.xml => jetty server plugin

<?xml version="1.0" encoding="UTF-8"?>  
<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>servlet-spring-group</groupId>  
 <artifactId>servlet-spring-artifact</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>war</packaging>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.eclipse.jetty</groupId>  
 <artifactId>jetty-maven-plugin</artifactId>  
 <version>9.2.11.v20150529</version>  
  
 <configuration>  
 <scanIntervalSeconds>2</scanIntervalSeconds>  
 <httpConnector>  
 <port>9999</port>  
 </httpConnector>  
 <webApp>  
 <contextPath>/web</contextPath>  
 </webApp>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

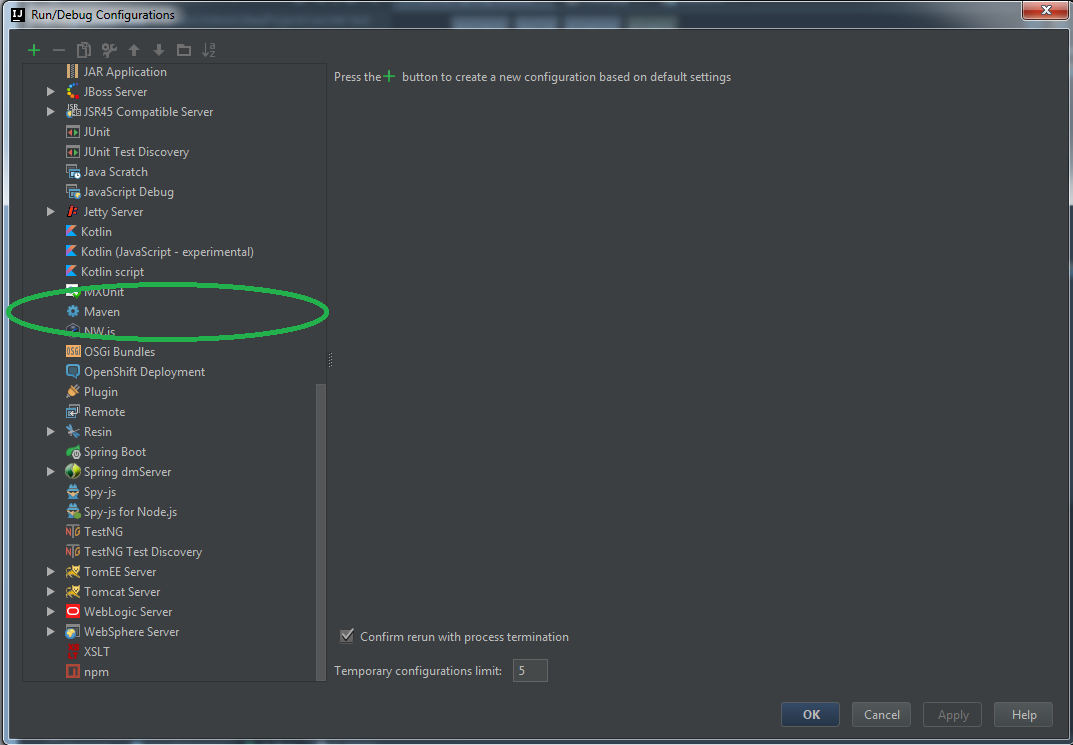


Создаем **maven** конфигурацию(для быстрого запуска приложения)

Выполняем **ctrl+shift+A** -> вводим **edit configurations**

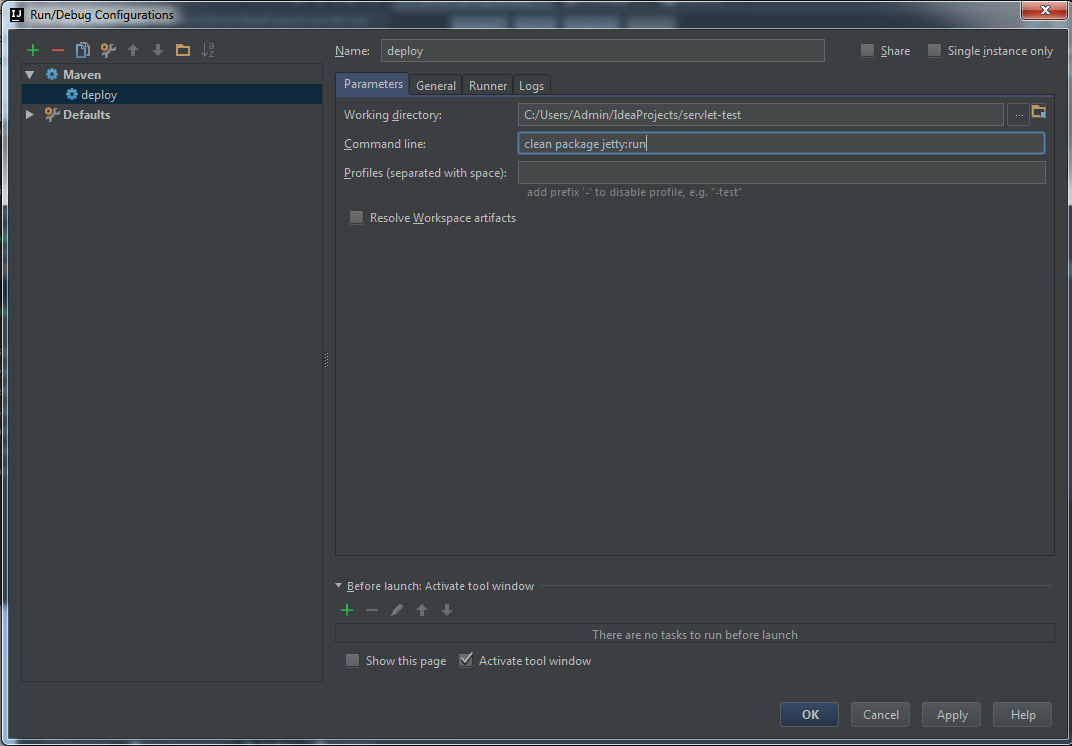


Нажимаем на «+» -> maven config



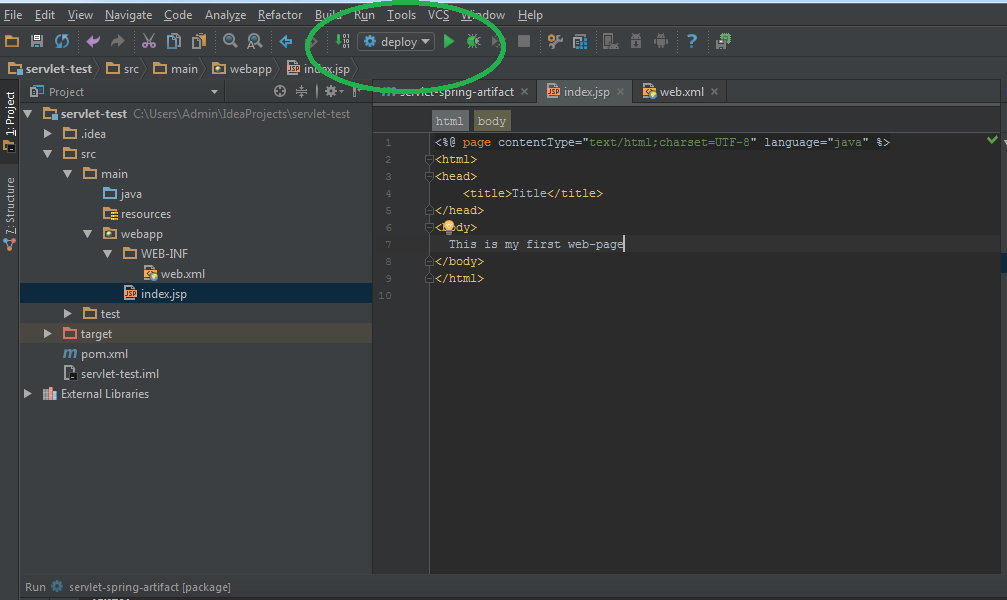
Command line -> clean package jetty:run

В результате должна получиться следующая конфигурация:



Затем **apply -> ok**

Попробуем задеплоить

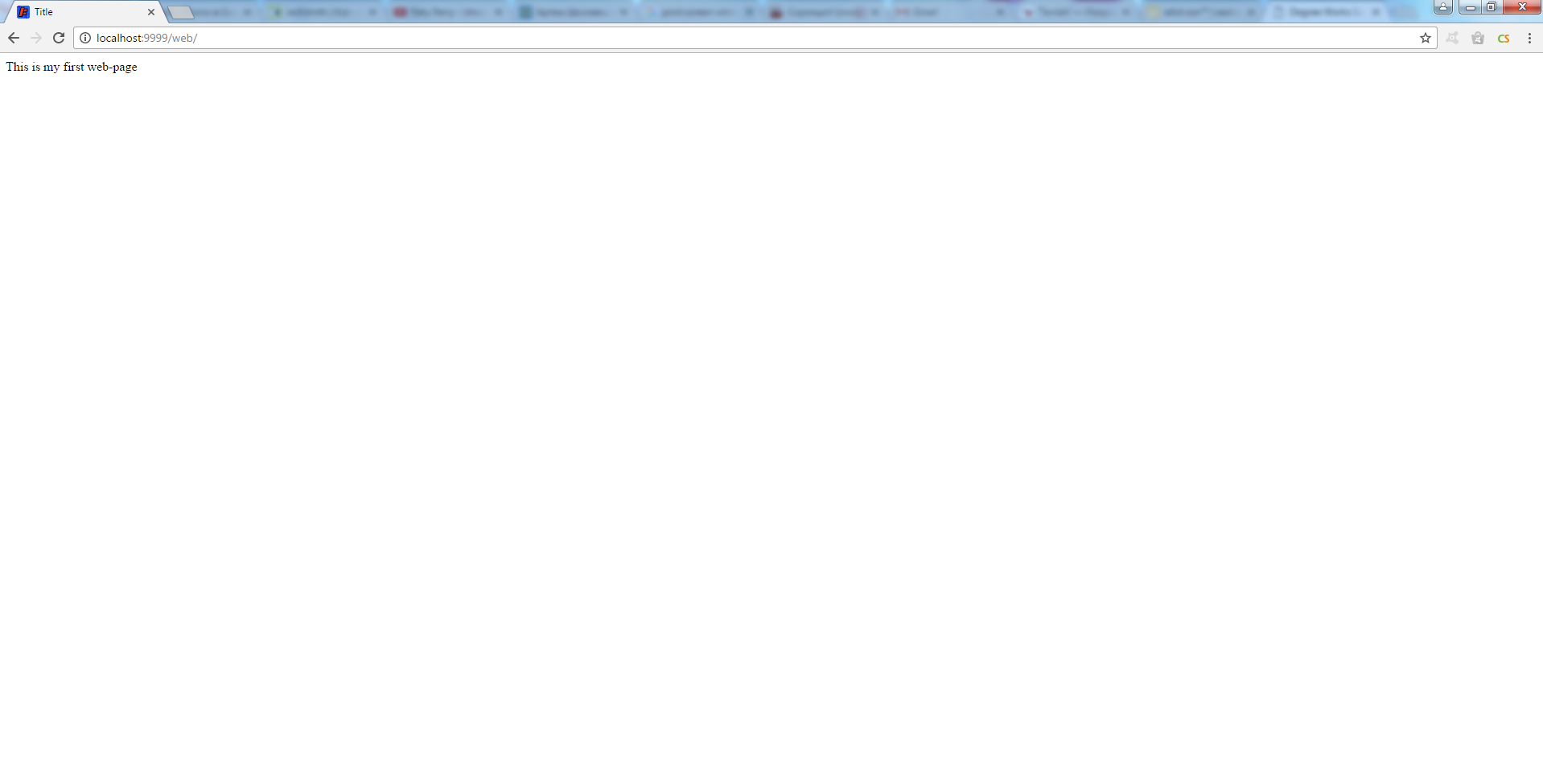


Выберите нашу конфигурацию, затем please start jetty via  or 

Откройте любой браузер – попробуйте ввести **localhost:9999/web**

Прошу обратить внимание на то, что параметры сервера мы указываем в пределах pom.xml ->

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.eclipse.jetty</groupId>  
 <artifactId>jetty-maven-plugin</artifactId>  
 <version>9.2.11.v20150529</version>  
  
 <configuration>  
 <scanIntervalSeconds>2</scanIntervalSeconds>  
 <httpConnector>  
 <port>9999</port>  
 </httpConnector>  
 <webApp>  
 <contextPath>/web</contextPath>  
 </webApp>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>



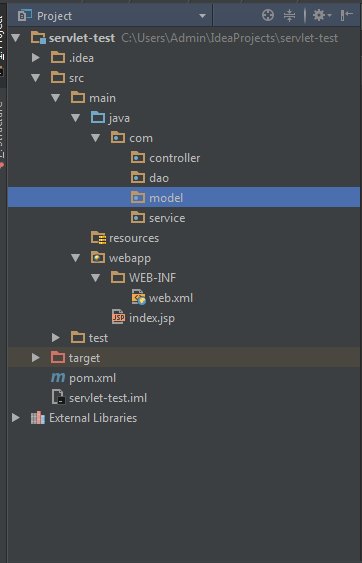
Мы большие молодцы ☺

1. на данном этапе у нас уже есть полноценное веб приложение – давайте попробуем разработать UserManagement system -> для этого

* обновим зависимости pom.xml(добавим spring && javax.servlet support)

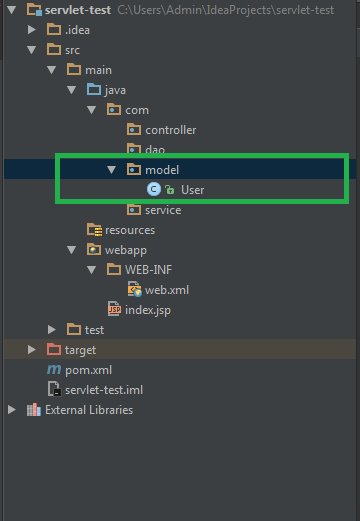
<properties>  
 <spring.version>3.0.6.RELEASE</spring.version>  
</properties>  
  
<dependencies>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>2.5</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
</dependencies>

* подготовим пакеты по dao- service- model- controller- levels



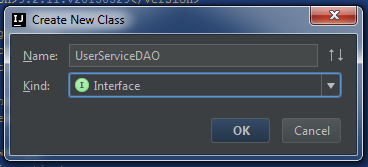
* создадим модель User (package com.model)

package com.model;  
  
public class User {  
 private String name;  
 private int age;  
  
 public User(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 User user = (User) o;  
  
 if (age != user.age) return false;  
 return name != null ? name.equals(user.name) : user.name == null;  
  
 }  
  
 @Override  
 public int hashCode() {  
 int result = name != null ? name.hashCode() : 0;  
 result = 31 \* result + age;  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "name='" + name + '\'' +  
 ", age=" + age +  
 '}';  
 }  
}



* создадим **dao-level**

определим интерфейс

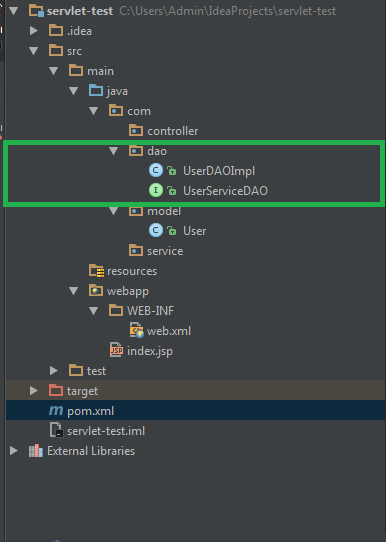


package com.dao;  
  
  
import com.model.User;  
  
import java.util.List;  
  
public interface UserServiceDAO {  
 List<User> getAllAvailableUsers();  
}

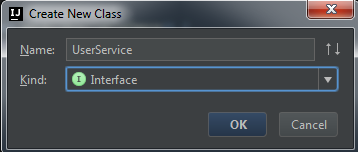
реализуем логику данного интерфейса в пределах UserDAOImpl(с использованием спринга)

package com.dao;  
  
import com.model.User;  
import org.springframework.stereotype.Repository;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
@Repository("userDAO")  
public class UserDAOImpl implements UserServiceDAO {  
 private List<User> users = new ArrayList<User>(Arrays.*asList*(new User("xxx", 12), new User("yyy", 14)));  
  
 public List<User> getAllAvailableUsers() {  
 return users;  
 }  
}

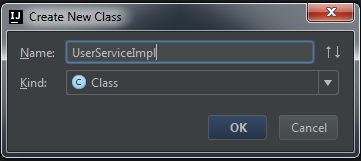
в результате получим следующую структуру проекта



* перейдем к реализации сервис-уровня(будем работать в пределах com->service пакета)

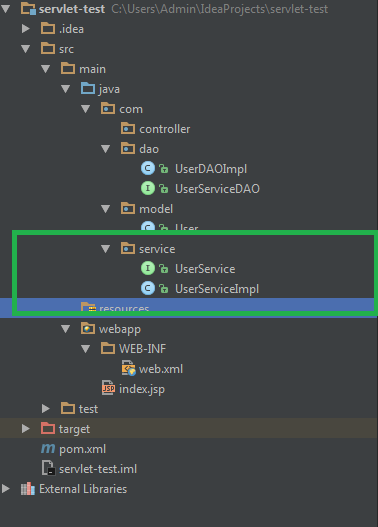


package com.service;  
  
import com.model.User;  
  
import java.util.List;  
  
public interface UserService {  
 List<User> getAllAvailableUsers();  
}



Реализация нашего интерфейса **UserService**

package com.service;  
  
import com.dao.UserServiceDAO;  
import com.model.User;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class UserServiceImpl implements UserService {  
  
 @Autowired  
 private UserServiceDAO userDAO;  
  
 public List<User> getAllAvailableUsers() {  
 return userDAO. getAllAvailableUsers();  
 }  
  
 public UserServiceDAO getUserDAO() {  
 return userDAO;  
 }  
  
 public void setUserDAO(UserServiceDAO userDAO) {  
 this.userDAO = userDAO;  
 }  
}



* перейдем к реализации нашего контроллера(будем работать в пределах com.controller)

package com.controller;  
  
import com.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.config.AutowireCapableBeanFactory;  
import org.springframework.web.context.support.WebApplicationContextUtils;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
public class UserController extends HttpServlet {  
  
 @Autowired  
 private UserService userService;  
  
 @Override  
 public void init() throws ServletException {  
 AutowireCapableBeanFactory factory = WebApplicationContextUtils.*getWebApplicationContext*(getServletContext()).getAutowireCapableBeanFactory();  
 factory.autowireBean(this);  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 resp.getWriter().println(userService.getAllAvailableUsers());  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 super.doPost(req, resp);  
 }  
}

Прошу обратить внимание на то, что инжект для сервлета мы проводим в «ручном режиме» посредством

@Override  
public void init() throws ServletException {  
 AutowireCapableBeanFactory factory = WebApplicationContextUtils.*getWebApplicationContext*(getServletContext()).getAutowireCapableBeanFactory();  
 factory.autowireBean(this);  
}

Давайте проанализируем, что у нас есть:

1. система UserManagement -> service- dao- controller- levels
2. spring injection (аннотации над основными классами, определение зависимостей)
3. server + web структура

Что отсутствует?

1. Механизм запуска spring-контекста(вспомните пожалуйста ClassPathXmlApplicationContext -> запуск из main, сейчас по факту main у нас нет)

Поэтому для решения данной проблемы:

* + Создадим **applicationContext.xml** -> для определения основных сетингов **spring**

Для этого – щелкаем правой кнопкой на уровне **WEB-INF** – создаем обычный xml файлик(**applicationContext.xml**) со следующим контентом

<?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:cnxt="http://www.springframework.org/schema/context"  
 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 ">  
  
 <cnxt:component-scan base-package="com"/>  
 <cnxt:annotation-config/>  
  
</beans>

При этом мы должны понимать, что

<cnxt:annotation-config/>

Подключение режима аннотаций

<cnxt:component-scan base-package="com"/>

Определяет какой пакет будет сканироваться для IOC

* + Для поднятия контекста во время запуска откроем **web.xml**
    - добавим следующую строчки

<context-param>  
 <param-name>applicationContext</param-name>  
 <param-value>/WEB-INF/applicationContext.xml</param-value>  
</context-param>

и

<listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>

В результате должны получить следующий вид **web.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  
 version="3.1">  
 <context-param>  
 <param-name>applicationContext</param-name>  
 <param-value>/WEB-INF/applicationContext.xml</param-value>  
 </context-param>  
  
 <listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
 </listener>  
</web-app>

У нас осталась еще одна небольшая проблема, а именно - >

1. Перенаправление запросов к нашему контролеру

Этот условный маппинг мы будем определять в пределах **web.xml**

<servlet>  
 <servlet-name>userController</servlet-name>  
 <servlet-class>com.controller.UserController</servlet-class>  
</servlet>  
  
<servlet-mapping>  
 <servlet-name>userController</servlet-name>  
 <url-pattern>/users</url-pattern>  
</servlet-mapping>

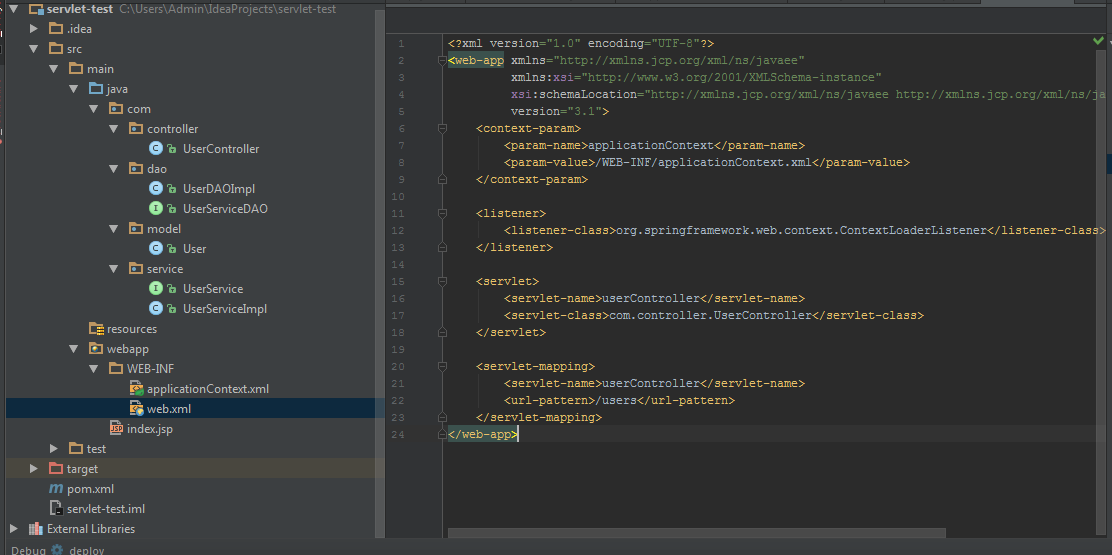
То-есть в случае указания в запросе после основного веб-контекста users (определяется в пределах **<url-pattern>/users</url-pattern>**) мы будем стучаться к нашему контролеру, а именно **UserController**

**Вариант url ->**

**<server-host>:<port>/<web-context>/users**

В результате последних изменений **web.xml** должен иметь следующий вид:

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  
 version="3.1">  
 <context-param>  
 <param-name>applicationContext</param-name>  
 <param-value>/WEB-INF/applicationContext.xml</param-value>  
 </context-param>  
  
 <listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
 </listener>  
  
 <servlet>  
 <servlet-name>userController</servlet-name>  
 <servlet-class>com.controller.UserController</servlet-class>  
 </servlet>  
  
 <servlet-mapping>  
 <servlet-name>userController</servlet-name>  
 <url-pattern>/users</url-pattern>  
 </servlet-mapping>  
</web-app>

 Пробуем поднять

Выберите нашу конфигурацию(смотрите пункт 1 – deploy) , затем запустите либо в обычном режиме  либо в дебаг моде 

Откройте любой браузер – попробуйте ввести **localhost:9999/web/users**

