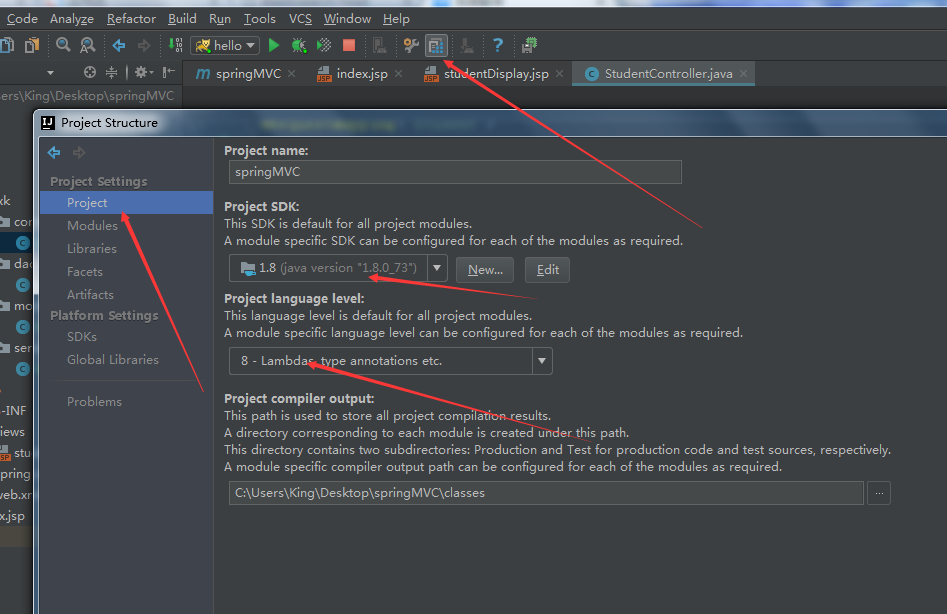
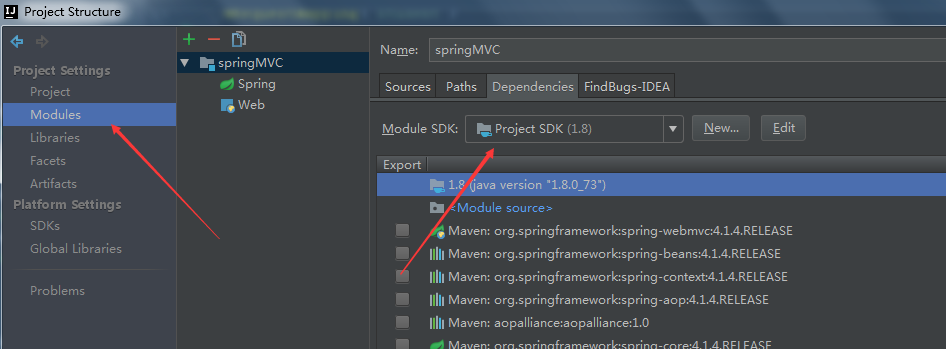
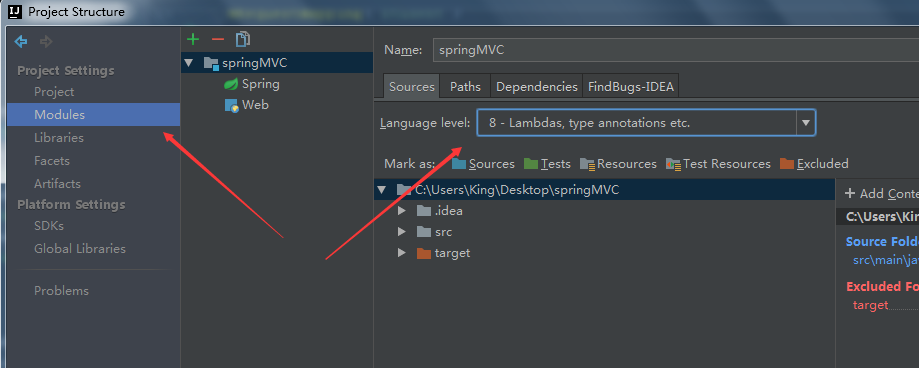
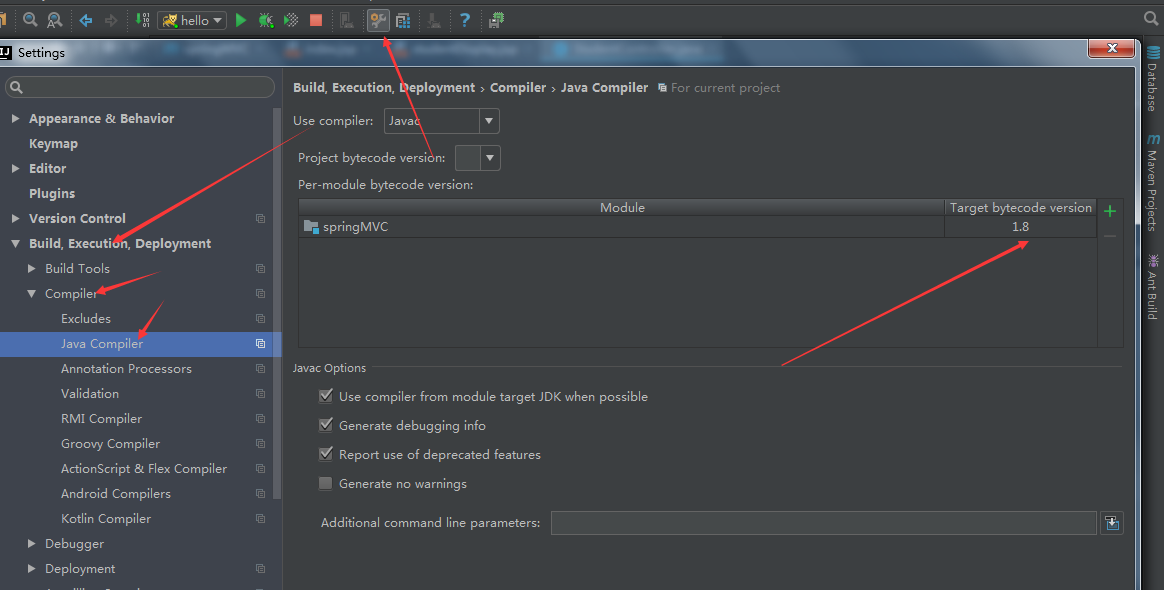
1. 下载IntelliJ专业版并使用license

下载安装jdk 1.8并设置jdk为java compiler









1. 安装使用总目录：<http://blog.csdn.net/qq_27093465/article/details/77449117>
2. 名词解释：

Java Bean：Java语言欠缺属性、事件、多重继承功能。所以，如果要在Java程序中实现一些面向对象编程的常见需求，只能手写大量胶水代码，Java Bean正是编写这套胶水代码的惯用模式或约定，包括getXxx、setXxx、isXxx、addXxxListener、XxxEvent等。

Spring：轻量级的控制反转（IoC）和面向切面（AOP）的容器框架，管理bean，包括很多开源项目SpringMVC：spring 处理web层请求的一个模块

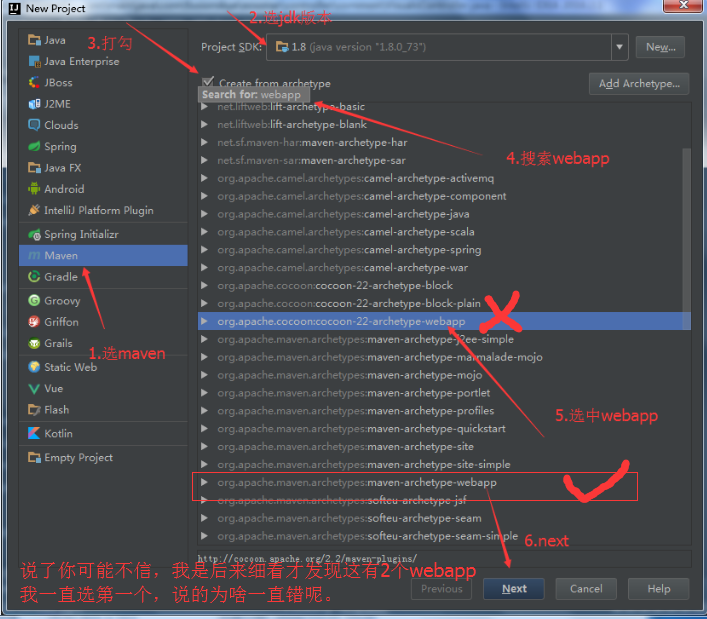
Spring流程：收到http请求，容器（如tomact）解析http生成一个request，通过映射关系（路径，方法，参数）被springmvc分发器找到可以处理这个请求的bean， omcat里面就由spring管理bean的一个池子（bean容器）里面找到，处理完了就把响应返回去。

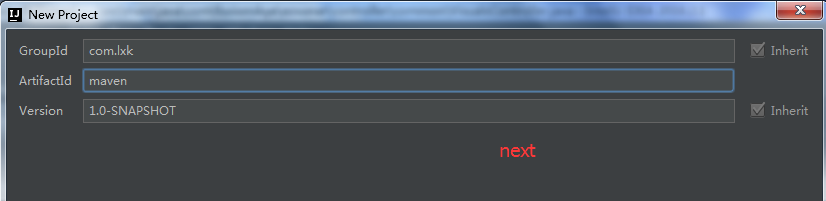
Maven：用于管理项目（文件，目录，配置等）的插件可自动下载需要的JAVA库（jar）

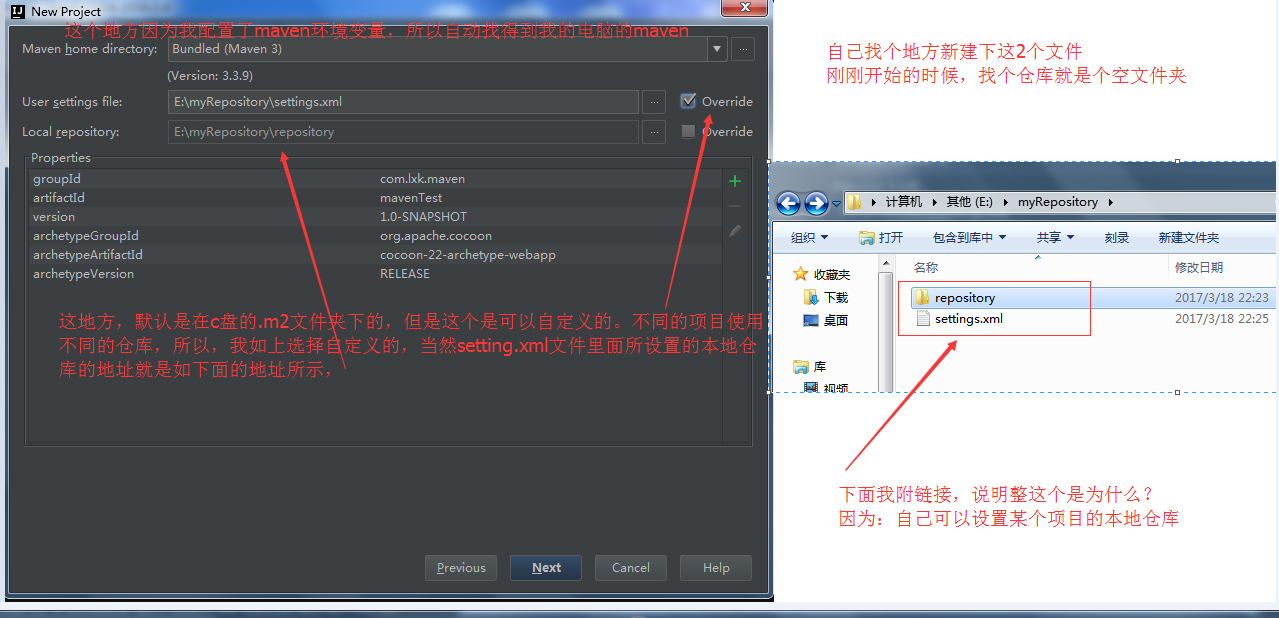
Module：

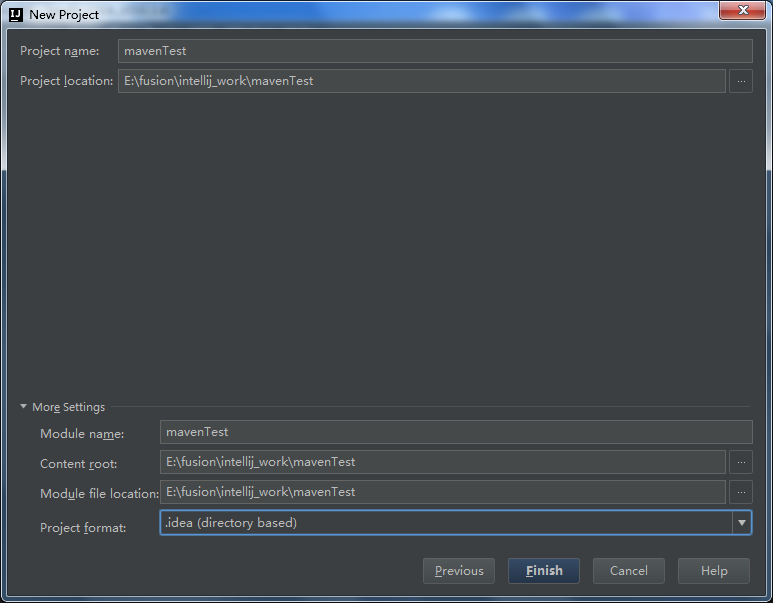
Aritfact：maven中的一个概念，表示某个module要如何打包，war exploded、war、jar、ear等等

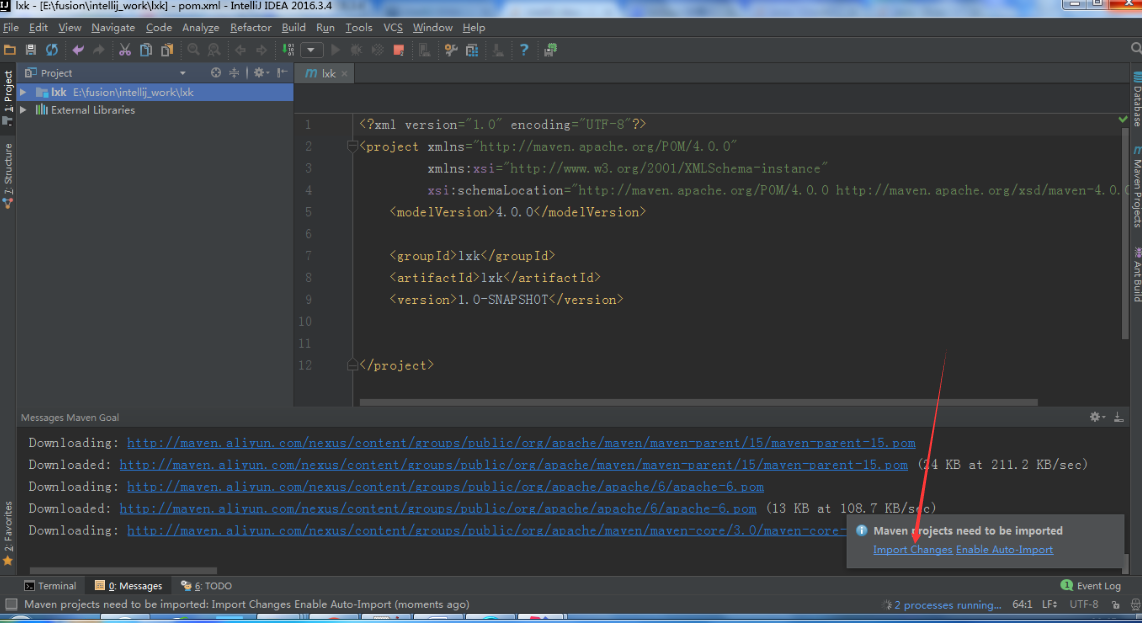
一个module有了 Artifacts 就可以部署到应用服务器中了！

1. 创建JAVA maven hello world程序
2. 创建maven项目mavenTest  
   

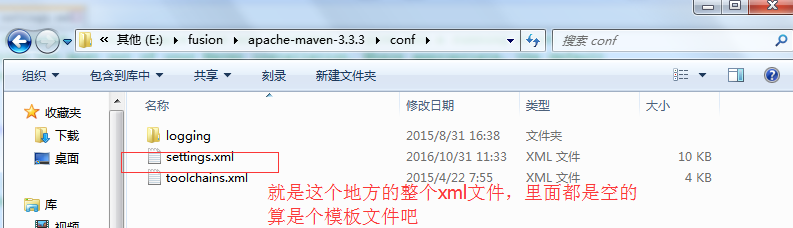


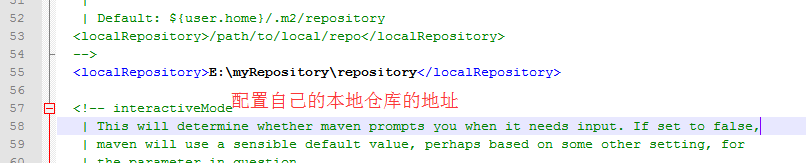


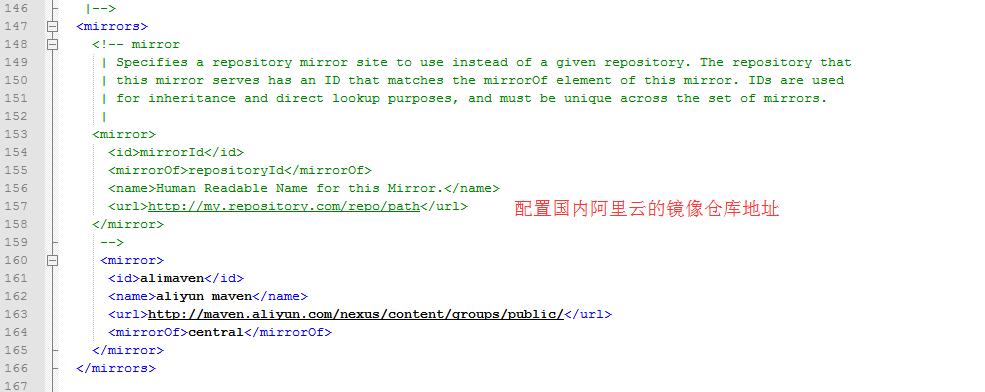




墙内可以使用阿里的maven服务器







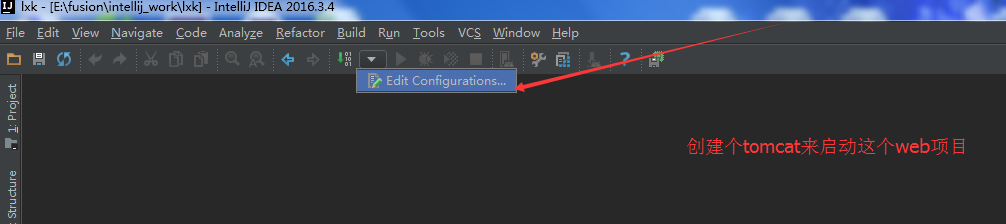
关于图中的镜像仓库的地址的文字如下：

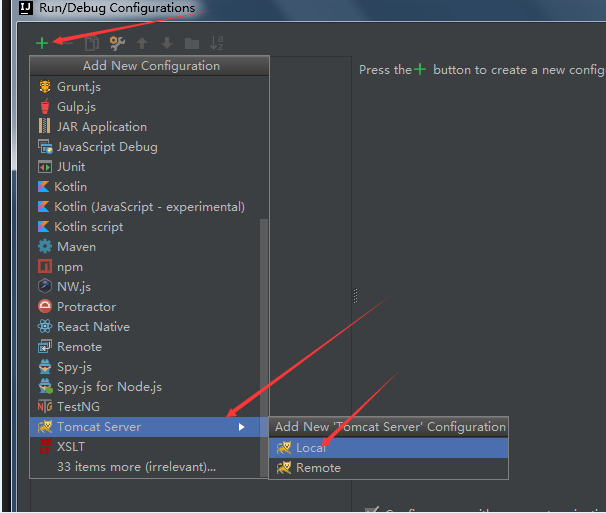
**[html]** [view plain](http://blog.csdn.net/qq_27093465/article/details/63683873) [copy](http://blog.csdn.net/qq_27093465/article/details/63683873)

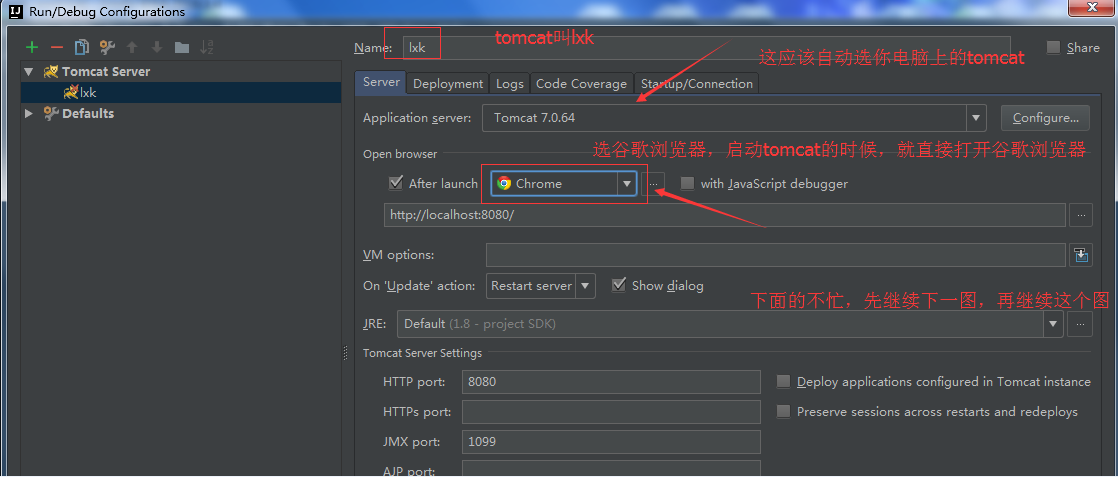
1. **<mirror>**
2. **<id>**alimaven**</id>**
3. **<name>**aliyun maven**</name>**
4. **<url>**http://maven.aliyun.com/nexus/content/groups/public/**</url>**
5. **<mirrorOf>**central**</mirrorOf>**
6. **</mirror>**

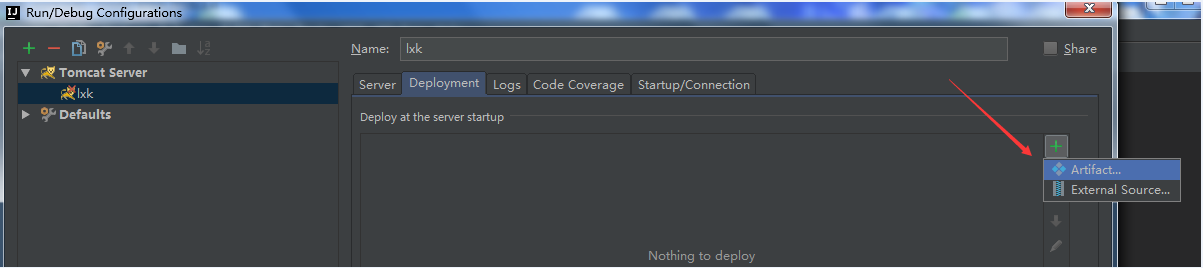
maven项目需要引入新jar包的时候，直接修改pom.xml再点击图上的**“Import Changes”**，maven会自动检查下包的依赖，没有的自己会去下载的。

1. 创建JAVA后台服务器（默认tomcat）

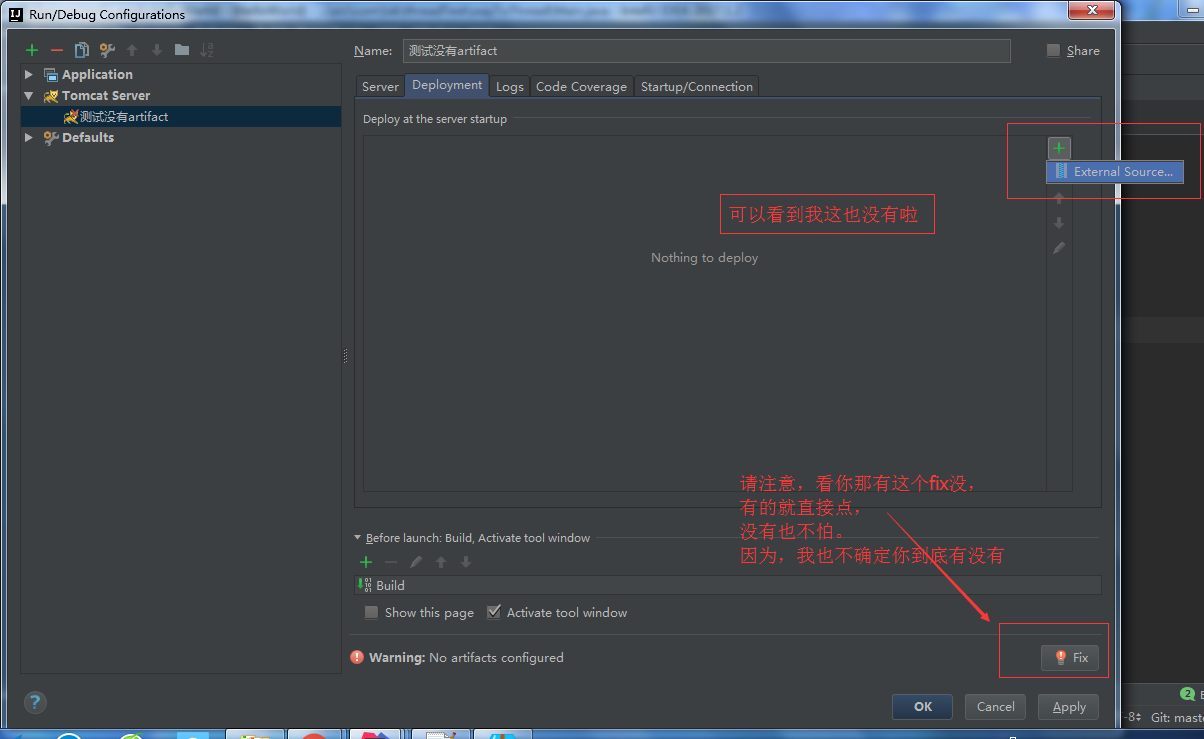


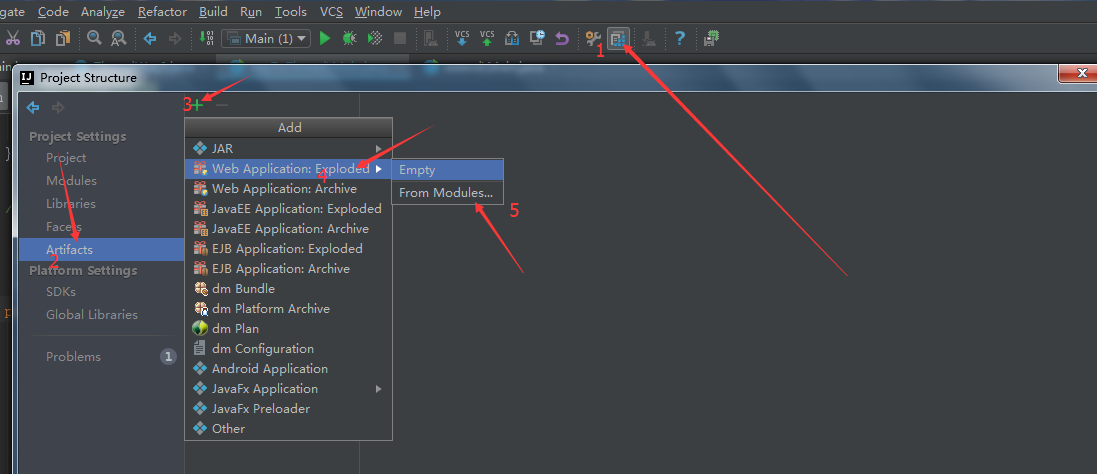


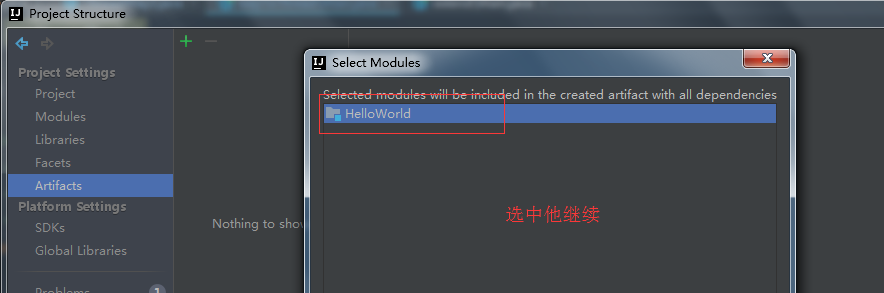


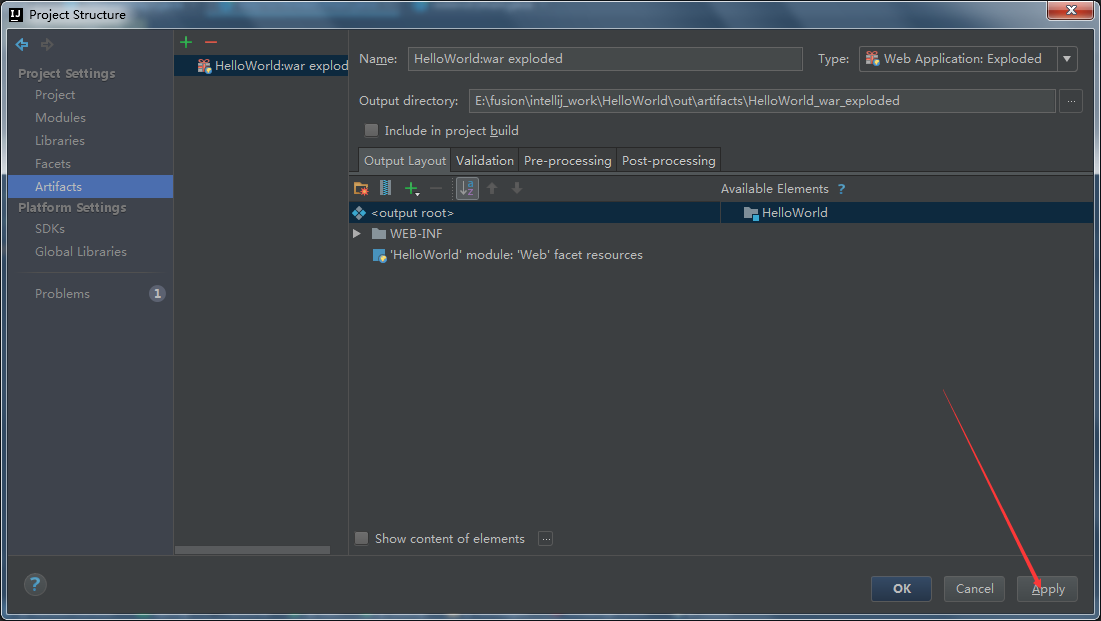


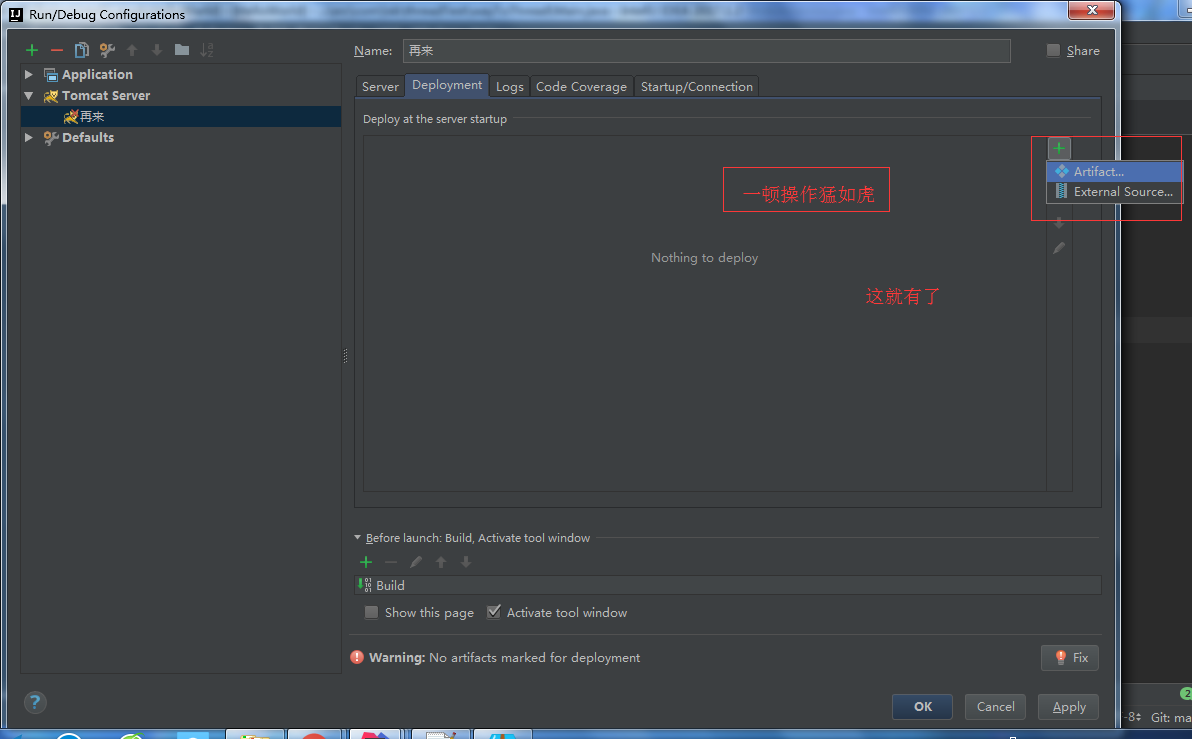
如果此时发现没有Artifact选项，可fix：







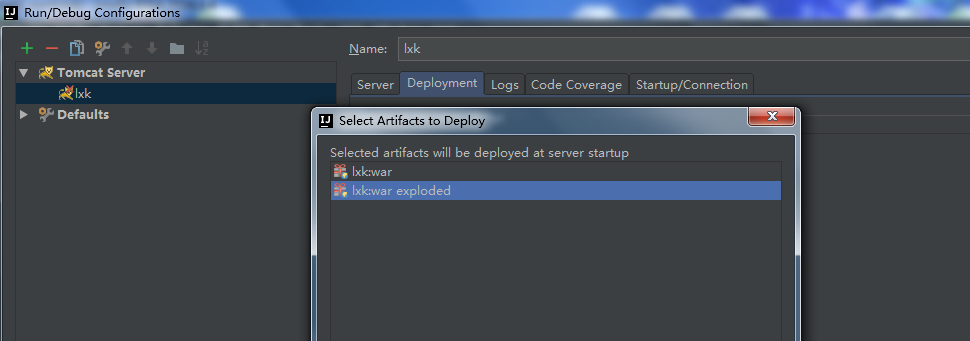


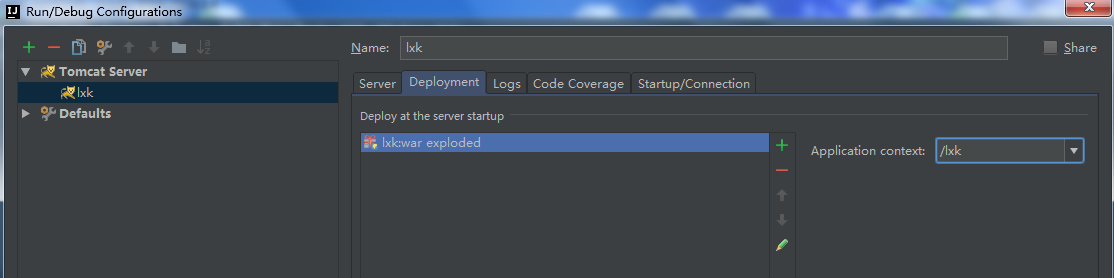


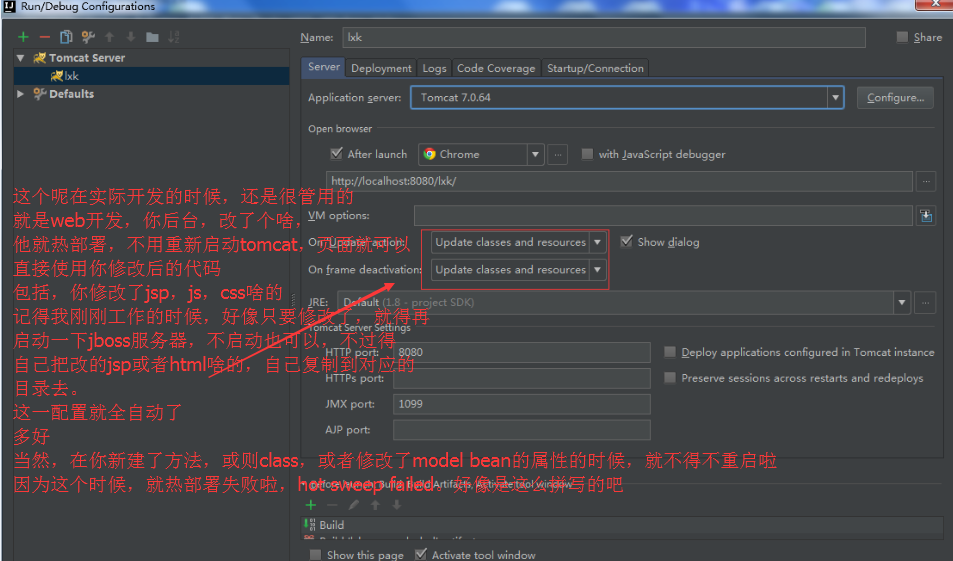
这样就有了

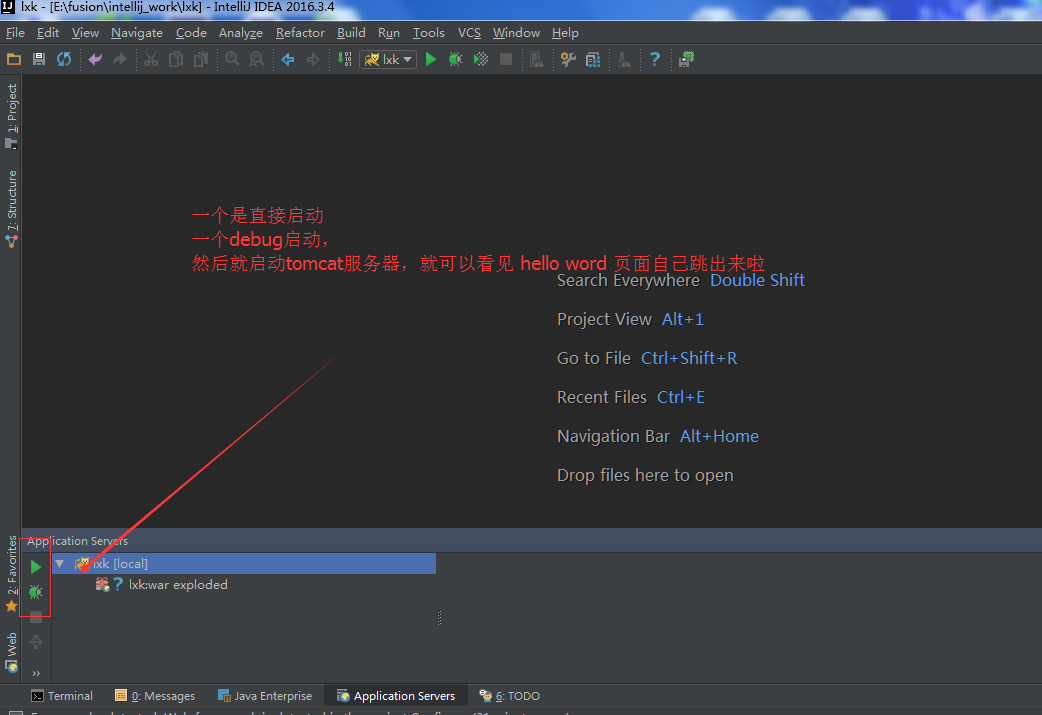
1. 配置热启动Tomcat服务器

选择explded(explode为展开不压缩的意思，也就是war jar等产出物没压缩前的目录结构。建议在开发的时候使用这种模式，便于修改了文件的效果立刻显现出来。)

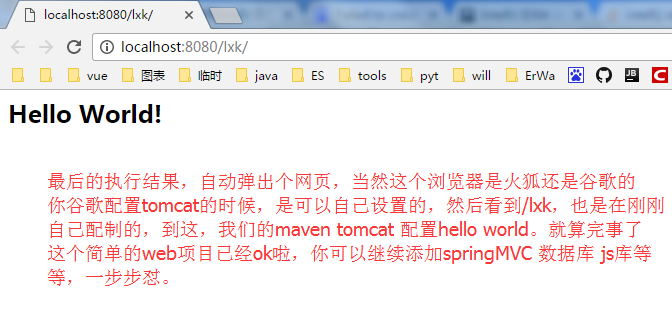




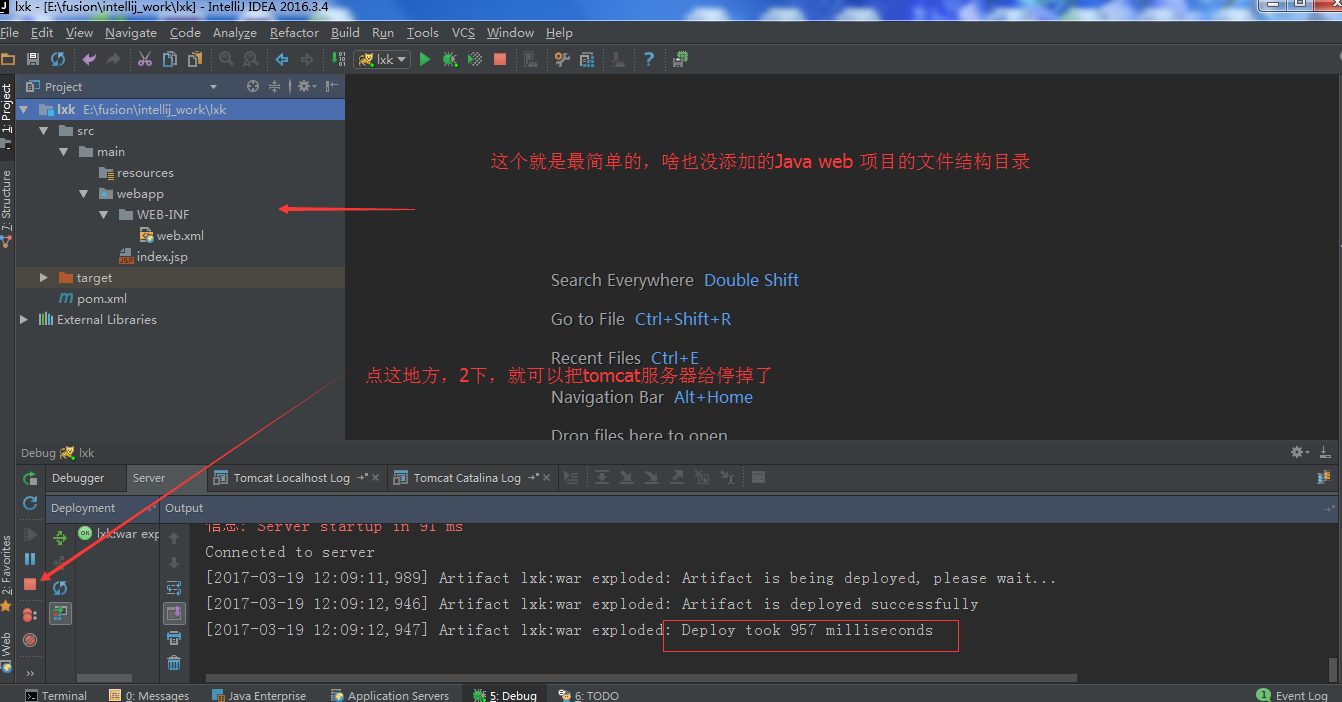




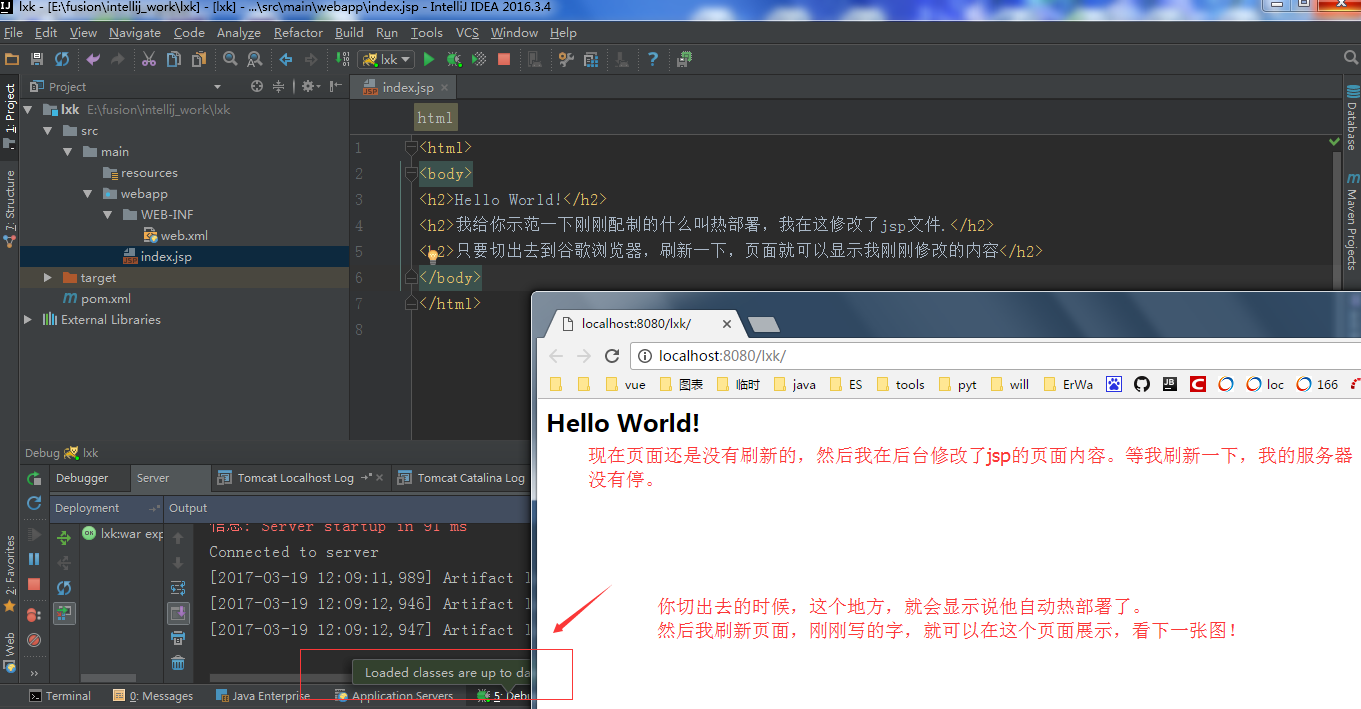
结果如下：



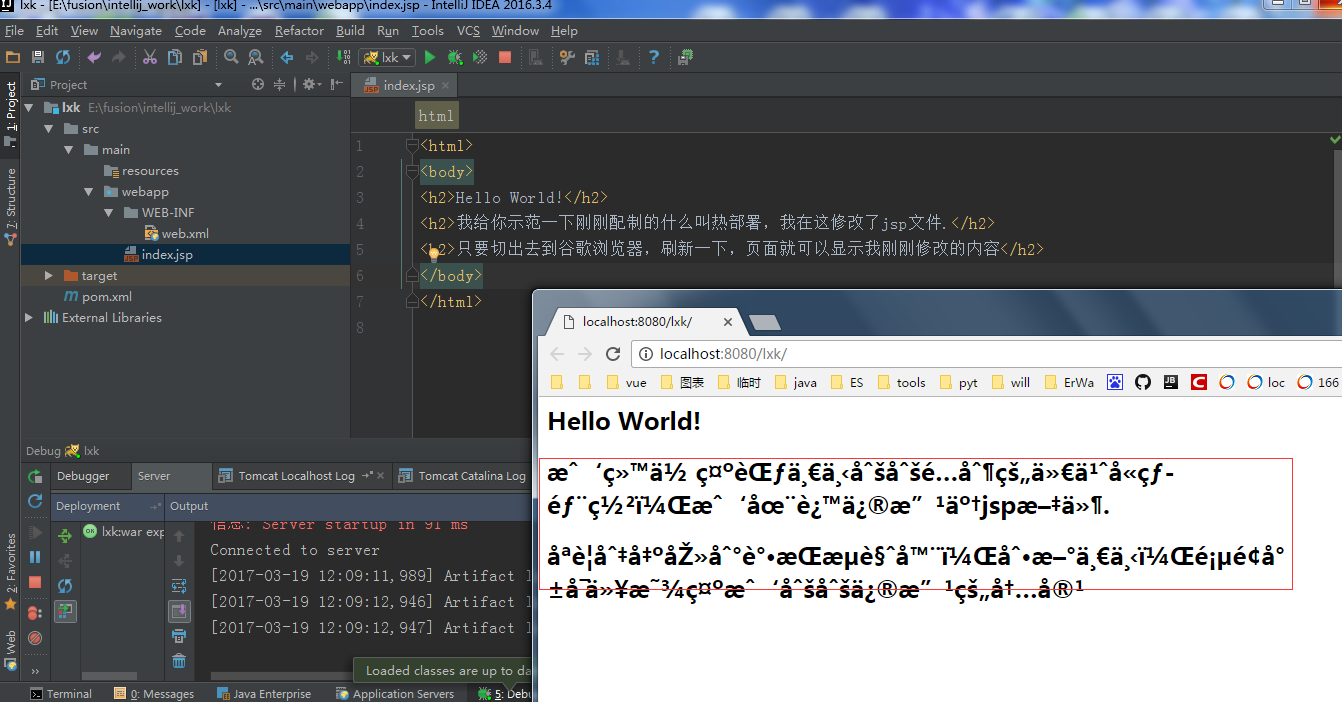
目录结构如下：

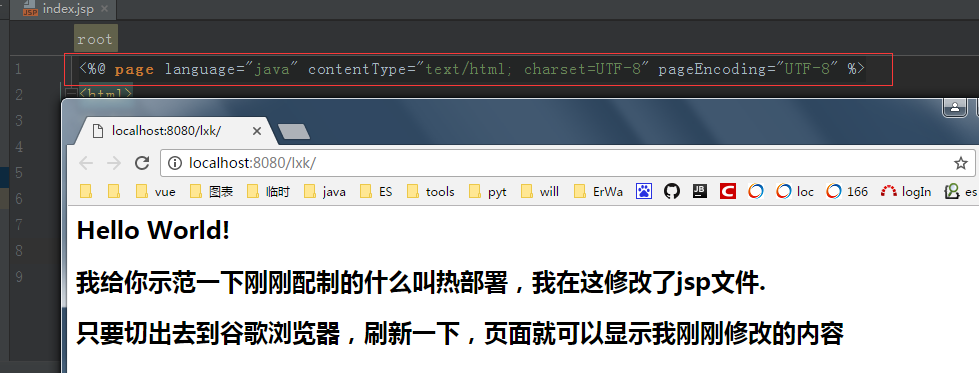


热启动效果：



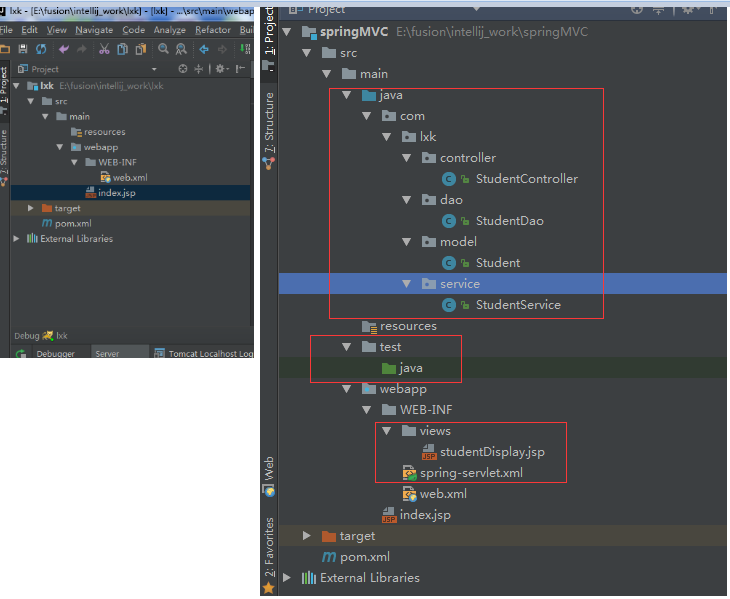
乱码处理：





在jsp文件中设置编码格式为utf8即可

1. **<**%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%**>**
2. 在JAVA maven hello world的基础上添加SpringMVC框架
   1. 为SpringMVC添加文件夹



目录结构：

有controller层，前台URL的映射关系，都在这个controller层处理。

有dao层，访问数据库，一般的crud都是在这写

有service层，一般的服务端的业务逻辑都是在service层，可以通用的方法都在这地方。

有前台页面views下面的都是前台页面的jsp，下面可以看到根据返回的ModelAndView类型，来绑定到对应的前台页面

有spring的配置spring-servlet.xml，spring配置相关的设置，扫描哪个目录下的文件注解等的设置

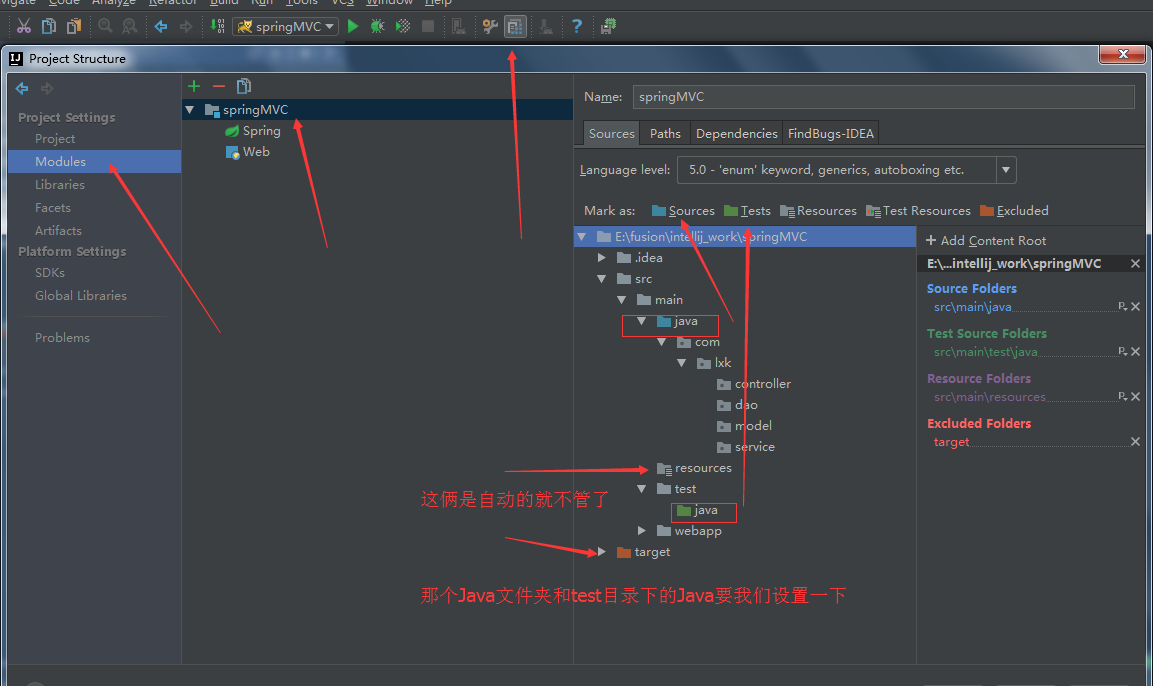
有web项目必备的web.xml 。。。。

有maven项目必备的pom.xml，存放的是项目所需的jar包依赖，以及项目打包所需的一些配置相关

有web项目必备的target文件夹，是项目打包自动生成的目录。

有test目录，这个也是项目的测试目录，一些测试代码就是在这文件夹下写的。

设置文件夹属性



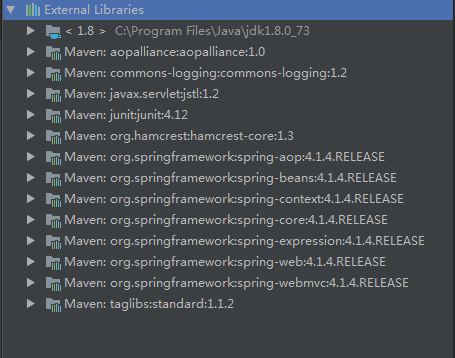
* 1. 配置xml文件

pom.xml文件中包含spring mvc依赖及为编写jsp文件提供支持的taglib相关依赖

1. **<project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"**>**
3. **<modelVersion>**4.0.0**</modelVersion>**
4. **<groupId>**lxk**</groupId>**
5. **<artifactId>**springMVC**</artifactId>**
6. **<packaging>**war**</packaging>**
7. **<version>**1.0-SNAPSHOT**</version>**
8. **<name>**springMVC Maven Webapp**</name>**
9. **<url>**http://maven.apache.org**</url>**
11. **<properties>**
12. **<org.springframework-version>**4.1.4.RELEASE**</org.springframework-version>**
13. **</properties>**
15. **<dependencies>**
16. <!-- Spring MVC support -->
17. **<dependency>**
18. **<groupId>**org.springframework**</groupId>**
19. **<artifactId>**spring-webmvc**</artifactId>**
20. **<version>**${org.springframework-version}**</version>**
21. **</dependency>**
22. **<dependency>**
23. **<groupId>**org.springframework**</groupId>**
24. **<artifactId>**spring-web**</artifactId>**
25. **<version>**${org.springframework-version}**</version>**
26. **</dependency>**
28. <!-- Tag libs support for view layer -->
29. **<dependency>**
30. **<groupId>**javax.servlet**</groupId>**
31. **<artifactId>**jstl**</artifactId>**
32. **<version>**1.2**</version>**
33. **<scope>**runtime**</scope>**
34. **</dependency>**
35. **<dependency>**
36. **<groupId>**taglibs**</groupId>**
37. **<artifactId>**standard**</artifactId>**
38. **<version>**1.1.2**</version>**
39. **<scope>**runtime**</scope>**
40. **</dependency>**
42. <!-- junit -->
43. **<dependency>**
44. **<groupId>**junit**</groupId>**
45. **<artifactId>**junit**</artifactId>**
46. **<version>**4.12**</version>**
47. **<scope>**test**</scope>**
48. **</dependency>**
49. **</dependencies>**

52. **<build>**
53. **<finalName>**springMVC**</finalName>**
54. **</build>**
55. **</project>**

更新pom.xml后maven会自动更新依赖的jar包



配置web.xml文件

1. **<web-app** id="WebApp\_ID" version="2.4"
2. xmlns="http://java.sun.com/xml/ns/j2ee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
5. http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"**>**
6. **<display-name>**Archetype Created Web Application**</display-name>**
8. **<servlet>**
9. **<servlet-name>**spring**</servlet-name>**
10. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
11. **<init-param>**
12. **<param-name>**contextConfigLocation**</param-name>**
13. **<param-value>**/WEB-INF/spring-servlet.xml**</param-value>**
14. **</init-param>**
15. **<load-on-startup>**1**</load-on-startup>**
16. **</servlet>**
17. **<servlet-mapping>**
18. **<servlet-name>**spring**</servlet-name>**
19. **<url-pattern>**/**</url-pattern>**
20. **</servlet-mapping>**
22. **</web-app>**

配置spring-servlet.xml的路径

1. **<beans** xmlns="http://www.springframework.org/schema/beans"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns:mvc="http://www.springframework.org/schema/mvc"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
6. http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
7. http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd"**>**
9. <!-- 开启spring的扫描注入，使用如下注解 -->
10. <!-- @Component,@Repository,@Service,@Controller-->
11. **<context:component-scan** base-package="com.lxk"**/>**
13. <!-- 开启springMVC的注解驱动，使得url可以映射到对应的controller -->
14. **<mvc:annotation-driven** **/>**
16. <!-- 视图解析 -->
17. **<bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
18. **<property** name="prefix" value="/WEB-INF/views/"**/>**
19. **<property** name="suffix" value=".jsp"**/>**
20. **</bean>**
22. **</beans>**
    1. mvc框架代码

controller代码

1. **package** com.lxk.controller;
3. **import** com.lxk.service.StudentService;
4. **import** org.springframework.stereotype.Controller;
5. **import** org.springframework.web.bind.annotation.RequestMapping;
6. **import** org.springframework.web.bind.annotation.RequestMethod;
7. **import** org.springframework.web.servlet.ModelAndView;
9. **import** javax.annotation.Resource;
11. /\*\*
12. \* Created by lxk on 2017/3/27
13. \*/
14. @Controller
15. @RequestMapping("student")
16. **public** **class** StudentController {
18. @Resource(name = "studentService")
19. **private** StudentService studentService;
21. @RequestMapping(value = "/getAllStudent", method = RequestMethod.GET)
22. **public** ModelAndView getAllStudent() {
23. ModelAndView mav = **new** ModelAndView();
24. mav.setViewName("studentDisplay");
25. mav.addObject("students", studentService.getAllStudent());
26. **return** mav;
27. }
28. }

Service代码

1. **package** com.lxk.service;
3. **import** com.lxk.dao.StudentDao;
4. **import** com.lxk.model.Student;
5. **import** org.springframework.stereotype.Service;
7. **import** javax.annotation.Resource;
8. **import** java.util.List;
10. /\*\*
11. \* Created by lxk on 2017/3/27
12. \*/
13. @Service("studentService")
14. **public** **class** StudentService {
16. @Resource(name = "studentDao")
17. **private** StudentDao dao;
19. **public** List<Student> getAllStudent() {
20. **return** dao.findAll();
21. }
22. }

Model代码，为了简单没有访问数据库而是直接返回了一个list集合

1. **package** com.lxk.dao;
3. **import** com.lxk.model.Student;
4. **import** org.springframework.stereotype.Repository;
6. **import** java.util.ArrayList;
7. **import** java.util.List;
9. /\*\*
10. \* Created by lxk on 2017/3/27
11. \*/
12. @Repository("studentDao")
13. **public** **class** StudentDao {
15. **public** List<Student> findAll() {
16. List<Student> all = **new** ArrayList<Student>();
17. all.add(**new** Student("1", "tom", 18, **true**));
18. all.add(**new** Student("2", "cat", 28, **true**));
19. all.add(**new** Student("3", "lxk", 18, **true**));
20. all.add(**new** Student("4", "cms", 18, **true**));
21. **return** all;
22. }
23. }

Student定义为一个JAVA Bean：

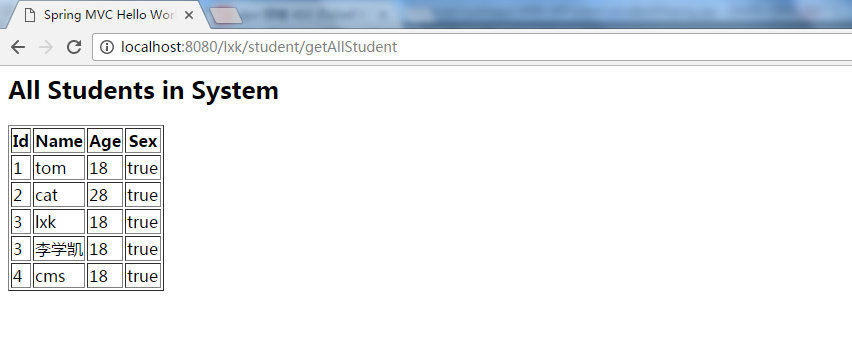
1. **package** com.lxk.model;
3. **import** java.io.Serializable;
5. /\*\*
6. \* Created by lxk on 2017/3/27
7. \*/
8. **public** **class** Student **implements** Serializable {
9. **private** String id;
10. **private** String name;
11. **private** **int** age;
12. **private** **boolean** sex;

15. **public** Student() {
16. }
18. **public** Student(String id, String name, **int** age, **boolean** sex) {
19. **this**.id = id;
20. **this**.name = name;
21. **this**.age = age;
22. **this**.sex = sex;
23. }
25. //getter and setter
26. ......
27. }

用jsp写前台

1. **<**%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%**>**
2. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%**>**
3. **<**%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%**>**
5. **<html>**
6. **<head>**
7. **<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**
8. **<title>**Spring MVC Hello World**</title>**
9. **</head>**
11. **<body>**
12. **<h2>**All Students in System**</h2>**
14. **<table** border="1"**>**
15. **<tr>**
16. **<th>**Id**</th>**
17. **<th>**Name**</th>**
18. **<th>**Age**</th>**
19. **<th>**Sex**</th>**
20. **</tr>**
21. **<c:forEach** var="student" items="${students}"**>**
22. **<tr>**
23. **<td>**${student.id}**</td>**
24. **<td>**${student.name}**</td>**
25. **<td>**${student.age}**</td>**
26. **<td>**${student.sex}**</td>**
27. **</tr>**
28. **</c:forEach>**
29. **</table>**
31. **</body>**
32. **</html>**
33. 结果

输入如下URL  
http://localhost:8080/lxk/student/getAllStudent  
就可以看到如下的执行结果啦。



1. SpringMVC使用JDBC连接MySQL
   1. 运行MySQL
   2. Maven添加MySQL依赖

Pom.xml添加

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.25</version>

</dependency>

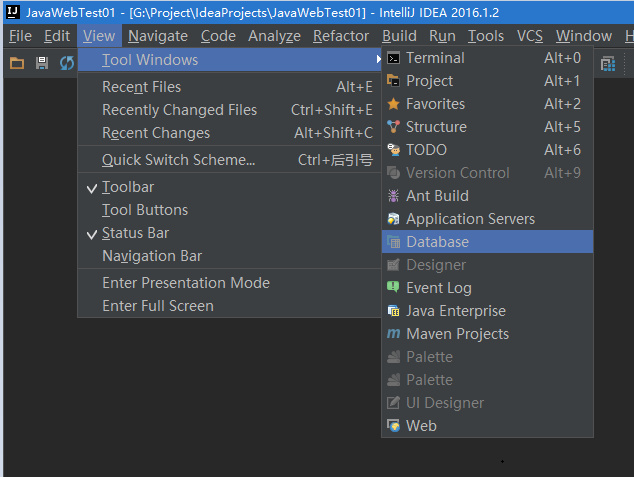
确保MySQL与mysql-connector-java版本一致，否则会报java.sql.SQLNonTransientConnectionException: CLIENT\_PLUGIN\_AUTH is required错误。

* 1. 重写SpringMVC的model层，使用JDBC连接MySQL

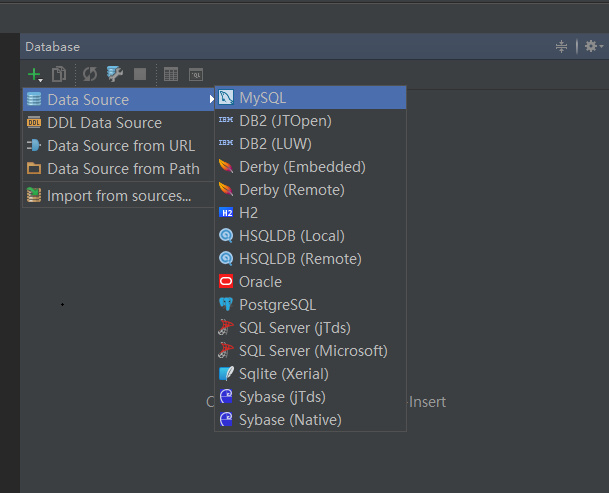
**public class** StudentDao {  
 **public** List<Student> findAll() {  
 List<Student> all = **new** ArrayList<Student>();  
 **try** {

**//需要注册一下否则会提示No suitable driver found for 'jdbc:mysql://localhost:3306/mysql**  
 DriverManager.*registerDriver*(**new** com.mysql.jdbc.Driver ());  
 Connection connection = *getConnection*();  
 Statement statement = connection.createStatement();  
 **try** (ResultSet resultSet = statement.executeQuery(**"SELECT** *\** **FROM students"**)) {  
 **while** (resultSet.next()) {  
 System.***out***.println(resultSet.getString(**"name"**));  
 all.add(**new** Student(resultSet.getString(**"id"**),  
 resultSet.getString(**"name"**),  
 resultSet.getInt(**"age"**),  
 resultSet.getBoolean(**"sex"**))  
 );  
 }  
 }  
 } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** all;  
 }  
  
 **private static** Connection getConnection() **throws** ClassNotFoundException, SQLException {  
 String serverName = **"localhost"**;  
 String database = **"StudentServer"**;  
 String url = **"jdbc:mysql://"** + serverName + **"/"** + database;  
  
 String user = **"root"**;  
 String password = **""**;  
  
 **return** DriverManager.*getConnection*(url, user, password);  
 }  
}

* 1. 配置MySQL服务器



添加MySQL数据库



测试MySQL是否成功运行

