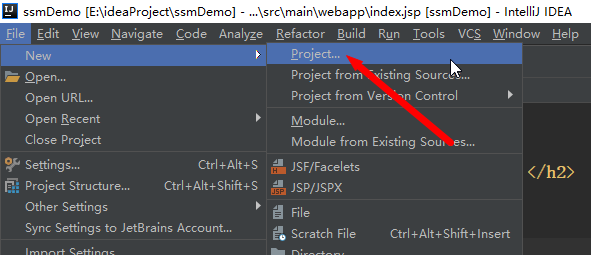
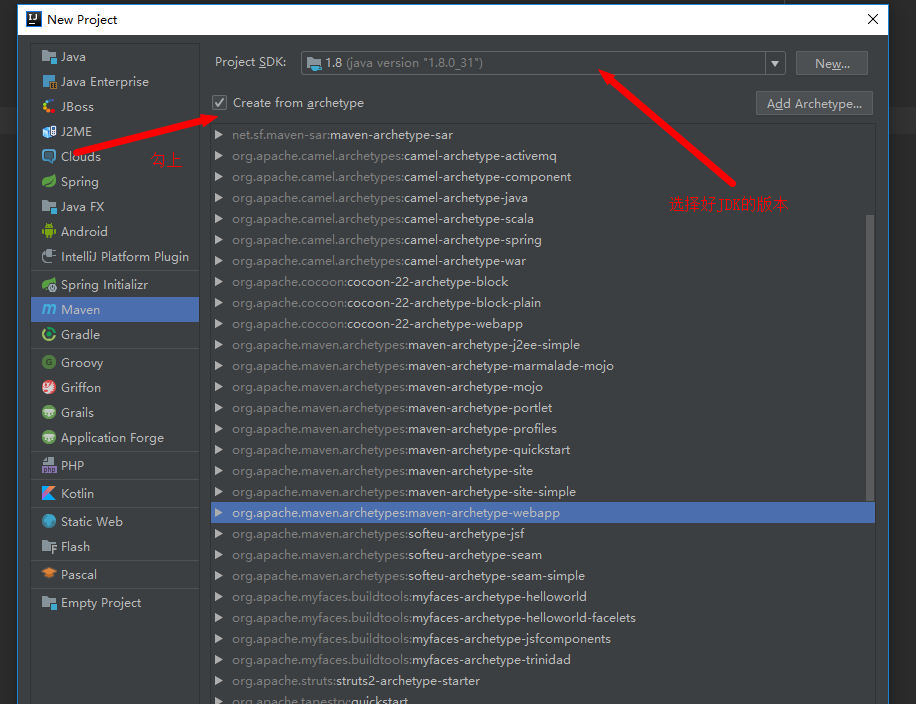
## **创建项目**

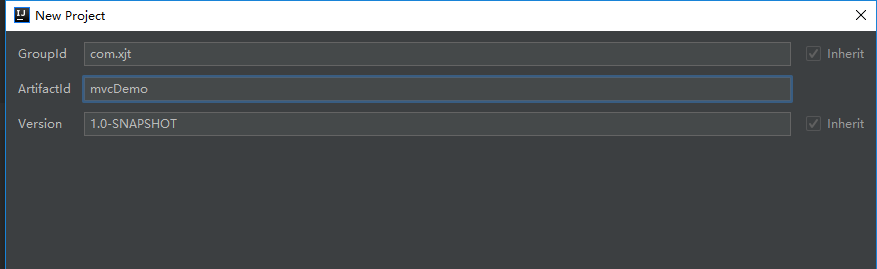
搭建好环境后，打开idea，点击New - Project



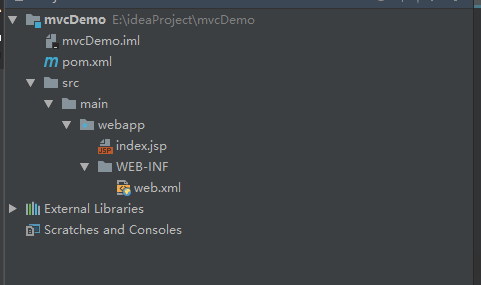
找到Maven一栏，因为要搭建的SpringMvc项目，所以选择webapp模板



填写好GroupId和ArtifactId后，一步步next，最后finish完成

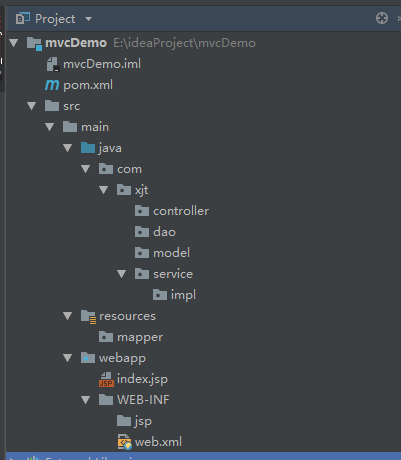


创建成功后，可以看到项目是这样的目录结构



除了配置相关依赖的pom.xml，目录中还有一个文件夹src，src的main目录提供了一个webapp文件夹，webapp文件夹下有一个WEB-INF文件夹，放置的是前端页面的文件，以及web.xml文件。

除了模版提供的目录结构，为了后面项目能成功运行，我们还需要添加一些文件夹，让项目的目录结构变成这样：



## **数据库文件**

首先，先准备好数据库文件，并初始化一条记录

DROP TABLE IF EXISTS `user`;CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '用户ID',

`email` varchar(255) NOT NULL COMMENT '用户邮箱',

`username` varchar(255) NOT NULL COMMENT '用户昵称',

`role` varchar(255) NOT NULL COMMENT '用户身份',

`mobile` varchar(50) DEFAULT '' COMMENT '手机号码',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf-8;

-- ------------------------------ Records of user-- ----------------------------INSERT INTO `user` VALUES ('1', '1589585621@qq.com', 'xjt', 'root', '15678635432');

## **配置文件**

### **pom.xml**

<?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>com.xjt</groupId>

<artifactId>mvcDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<name>mvcDemo Maven Webapp</name>

<!-- 用来设置版本号 -->

<properties>

<srping.version>4.0.2.RELEASE</srping.version>

<mybatis.version>3.2.8</mybatis.version>

<slf4j.version>1.7.12</slf4j.version>

<log4j.version>1.2.17</log4j.version>

<druid.version>1.0.9</druid.version>

</properties>

<!-- 用到的jar包 -->

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<!-- 表示开发的时候引入，发布的时候不会加载此包 -->

<scope>test</scope>

</dependency>

<!-- spring框架包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${srping.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${srping.version}</version>

</dependency>

<!-- spring框架包 -->

<!-- mybatis框架包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<!-- mybatis框架包 -->

<!-- 数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.35</version>

</dependency>

<!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<!-- jstl标签类 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- log -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- 连接池 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>${druid.version}</version>

</dependency>

</dependencies>

<build>

<!-- java编译插件,如果maven的设置里配置好jdk版本就不用 -->

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.2</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

</build></project>

**注：以下四个配置文件都是放置在resources文件夹下**

### **log4j.properties**

#日志输出级别

log4j.rootLogger=debug,stdout,D,E

#设置stdout的日志输出控制台

log4j.appender.stdout=org.apache.log4j.ConsoleAppender#输出日志到控制台的方式，默认为System.out

log4j.appender.stdout.Target = System.out#设置使用灵活布局

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout#灵活定义输出格式

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} -[%p] method:[%c (%rms)] - %m%n

### **jdbc.properties**

driver=com.mysql.jdbc.Driver url=jdbc:mysql://localhost:3306/test?characterEncoding=utf8&useSSL=false #数据库的用户名username=root #数据库的密码，别像我一样不设置password= #定义初始连接数initialSize=0 #定义最大连接数maxActive=20 #定义最大空闲maxIdle=20 #定义最小空闲minIdle=1 #定义最长等待时间maxWait=60000

### **applicationContext.xml**

这是Spring的核心配置文件，包括Spring结合Mybatis和数据源的配置信息

<?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:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">

<!-- 加载properties文件 -->

<bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value="classpath:jdbc.properties"/>

</bean>

<!-- 配置数据源 -->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">

<property name="driverClassName" value="${driver}"/>

<property name="url" value="${url}"/>

<property name="username" value="${username}"/>

<property name="password" value="${password}"/>

</bean>

<!-- mybatis和spring完美整合，不需要mybatis的配置映射文件 -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<!-- 扫描model包 -->

<property name="typeAliasesPackage" value="com.xjt.model"/>

<!-- 扫描sql配置文件:mapper需要的xml文件-->

<property name="mapperLocations" value="classpath:mapper/\*.xml"/>

</bean>

<!-- Mapper动态代理开发，扫描dao接口包-->

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<!-- 注入sqlSessionFactory -->

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>

<!-- 给出需要扫描Dao接口包 -->

<property name="basePackage" value="com.xjt.dao"/>

</bean>

<!-- 事务管理 -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<!--数据库连接池-->

<property name="dataSource" ref="dataSource"/>

</bean>

### **spring-mvc.xml**

<?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:mvc="http://www.springframework.org/schema/mvc"

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/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

<!-- 扫描注解，这样com.xjt包下的文件都能被扫描 -->

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

<!-- 开启SpringMVC注解模式 -->

<mvc:annotation-driven/>

<!-- 静态资源默认servlet配置 -->

<mvc:default-servlet-handler/>

<!-- 配置返回视图的路径，以及识别后缀是jsp文件 -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>

<property name="prefix" value="/WEB-INF/jsp/"/>

<property name="suffix" value=".jsp"/>

</bean></beans>

### **web.xml**

该文件不是放在resources，而是webapp的WEB-INF文件夹下，文件内容如下：

<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">

<display-name>mvcDemo</display-name>

<!--项目的欢迎页，项目运行起来后访问的页面-->

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

<!-- 注册ServletContext监听器，创建容器对象，并且将ApplicationContext对象放到Application域中 -->

<listener>

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

</listener>

<!-- 指定spring核心配置文件 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

<!-- 解决乱码的过滤器 -->

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置前端控制器 -->

<servlet>

<servlet-name>springmvc</servlet-name>

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

<!-- 指定配置文件位置和名称 如果不设置,默认找/WEB-INF/<servlet-name>-servlet.xml -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mvc.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

<async-supported>true</async-supported>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern>

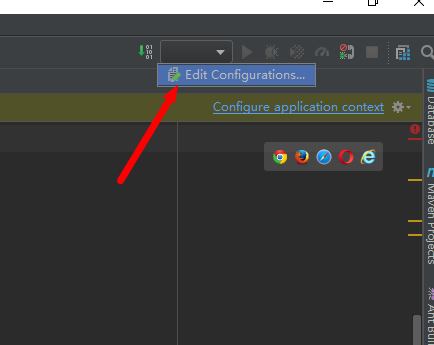
</servlet-mapping>

</web-app>

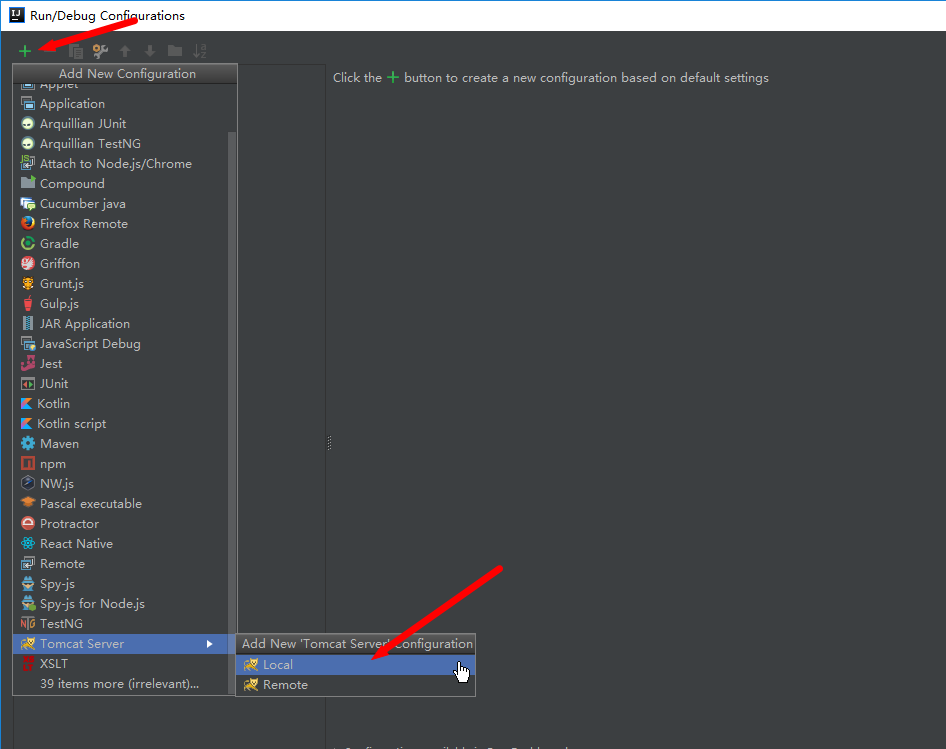
## **运行项目**

上面的配置文件编写好之后，其实我们就可以尝试着启动项目了，**启动项目前先把applicationContext.xml的内容都注释了**，然后就可以创建运行环境了。

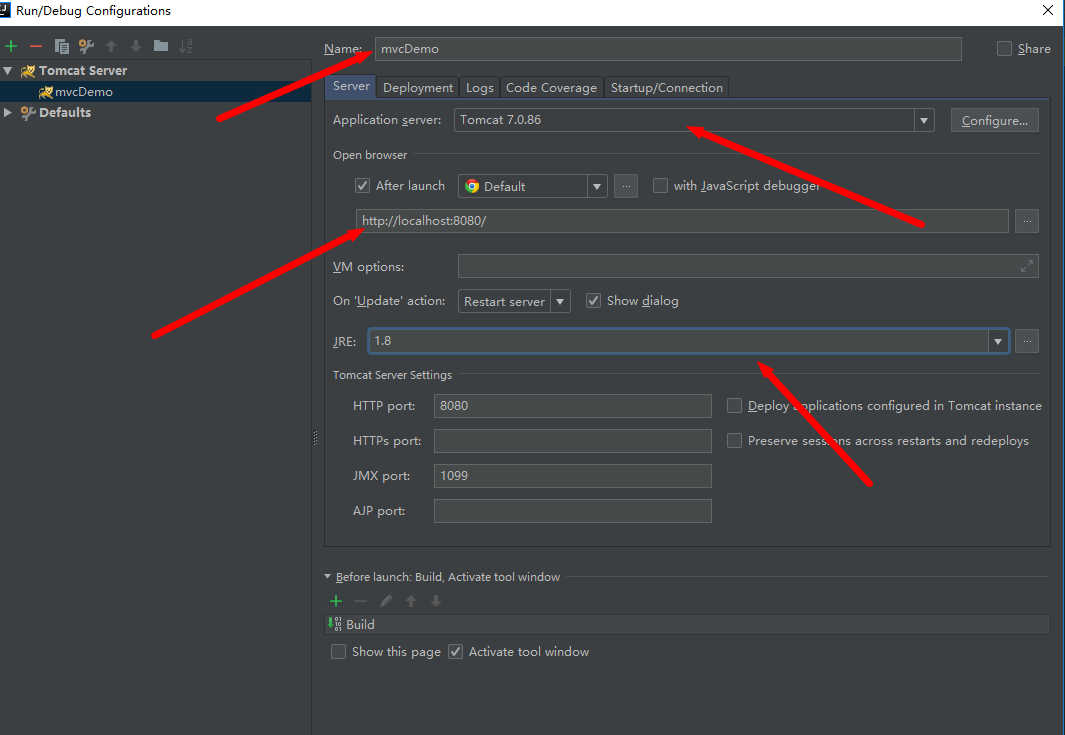
点击idea右上角的 Edit Configurations...



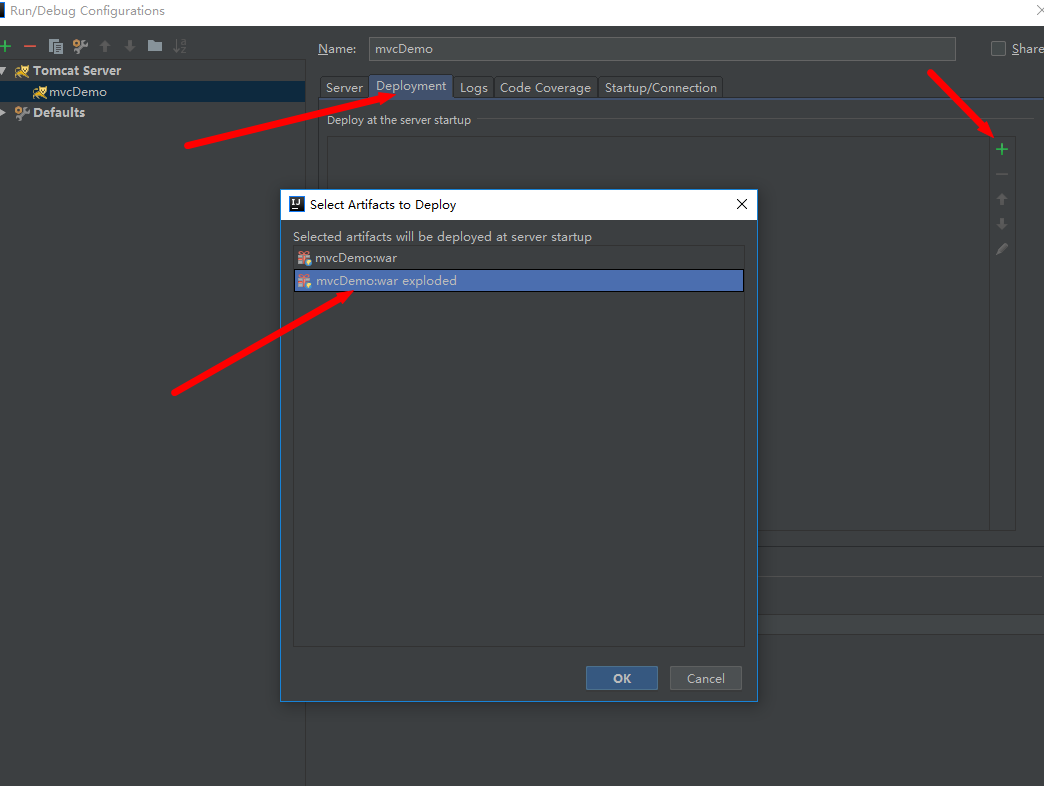
选择Tomcat Server - Local，

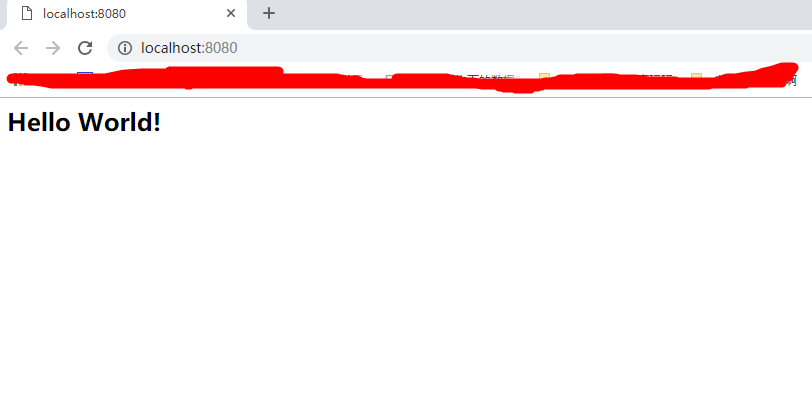


编辑好项目的启动信息，包括项目名，jdk版本，tomcat以及端口



选择Deployment，添加Atifact，选择第二项，否则Tomcat运行会报错



保存后，启动项目，成功后在浏览器输入<http://localhost:8080>，返回结果如下：  


这是index.jsp文件中的内容，因为index.jsp是启动页，所以项目启动后返回的结果是启动页的内容

<html><body><h2>Hello World!</h2></body></html>

## **编写代码**

项目能跑起来，说明我们前端控制器的配置是成功的，接下来要做的是测试能否访问数据库。首先，把基本的类文件创建并编写好代码。

**实体类：User.java**

package com.xjt.model;

public class User {

private long id;

private String email;

private String mobile;

private String username;

private String role;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

**dao文件：IUserDao**

package com.xjt.dao;

import com.xjt.model.User;

public interface IUserDao {

User selectUser(long id);

}

**mapper文件：UserDao.xml**

位于resources - mapper文件夹下

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<!-- 设置为IUserDao接口方法提供sql语句配置 --><mapper namespace="com.xjt.dao.IUserDao">

<select id="selectUser" resultType="User" parameterType="long">

SELECT \* FROM user WHERE id = #{id}

</select>

</mapper>

**service接口：IUserService**

package com.xjt.service;

import com.xjt.model.User;

public interface IUserService {

public User selectUser(long userId);

}

**UserServiceImpl**

package com.xjt.service.impl;

import com.xjt.dao.IUserDao;import com.xjt.model.User;import com.xjt.service.IUserService;import org.springframework.stereotype.Service;

import javax.annotation.Resource;

@Service("userService")public class UserServiceImpl implements IUserService {

@Resource

private IUserDao userDao;

public User selectUser(long userId) {

return userDao.selectUser(userId);

}

}

**controller文件：UserController**

package com.xjt.controller;

import com.xjt.model.User;import com.xjt.service.IUserService;import org.springframework.beans.factory.annotation.Autowired;import org.springframework.stereotype.Controller;import org.springframework.web.bind.annotation.RequestMapping;import org.springframework.web.servlet.ModelAndView;

@Controller@RequestMapping("/user")public class UserController {

@Autowired

private IUserService userService;

@RequestMapping("/select")

public ModelAndView selectUser() throws Exception {

ModelAndView mv = new ModelAndView();

User user = userService.selectUser(1);

mv.addObject("user", user);

mv.setViewName("user");

return mv;

}

}

UserController定义了一个方法selectUser，读取id为1的User信息，并返回user.jsp页面，访问的路径是/user/select，user.jsp位于 WEB-INF 的 jsp 文件夹下，代码如下：

<%@ page contentType="text/html;charset=UTF-8" language="java" %><html><head>

<title>user</title></head><body>

id:${requestScope.user.id}<br/>

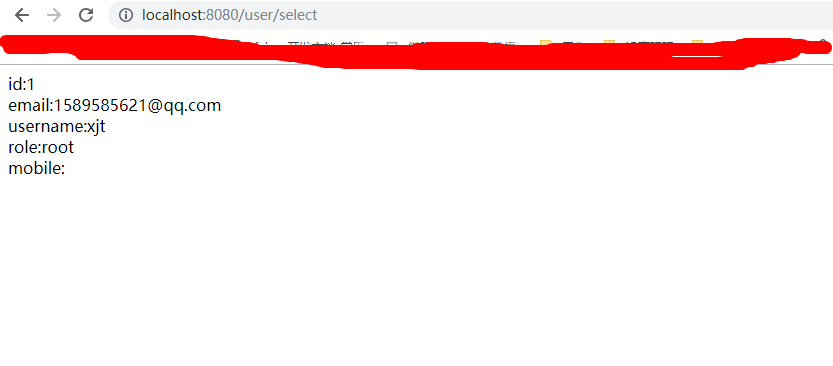
email:${requestScope.user.email}<br/>

username:${requestScope.user.username}<br/>

role:${requestScope.user.role}<br/>

mobile:${requestScope.user.mobile}<br/></body></html>

都编写完毕以后，撤销applicationContext.xml文件的注释符号，然后开启项目，成功后在浏览器输入<http://localhost:8080/user/select>



页面成功的返回了User信息，说明，我们成功访问到了数据库的内容。