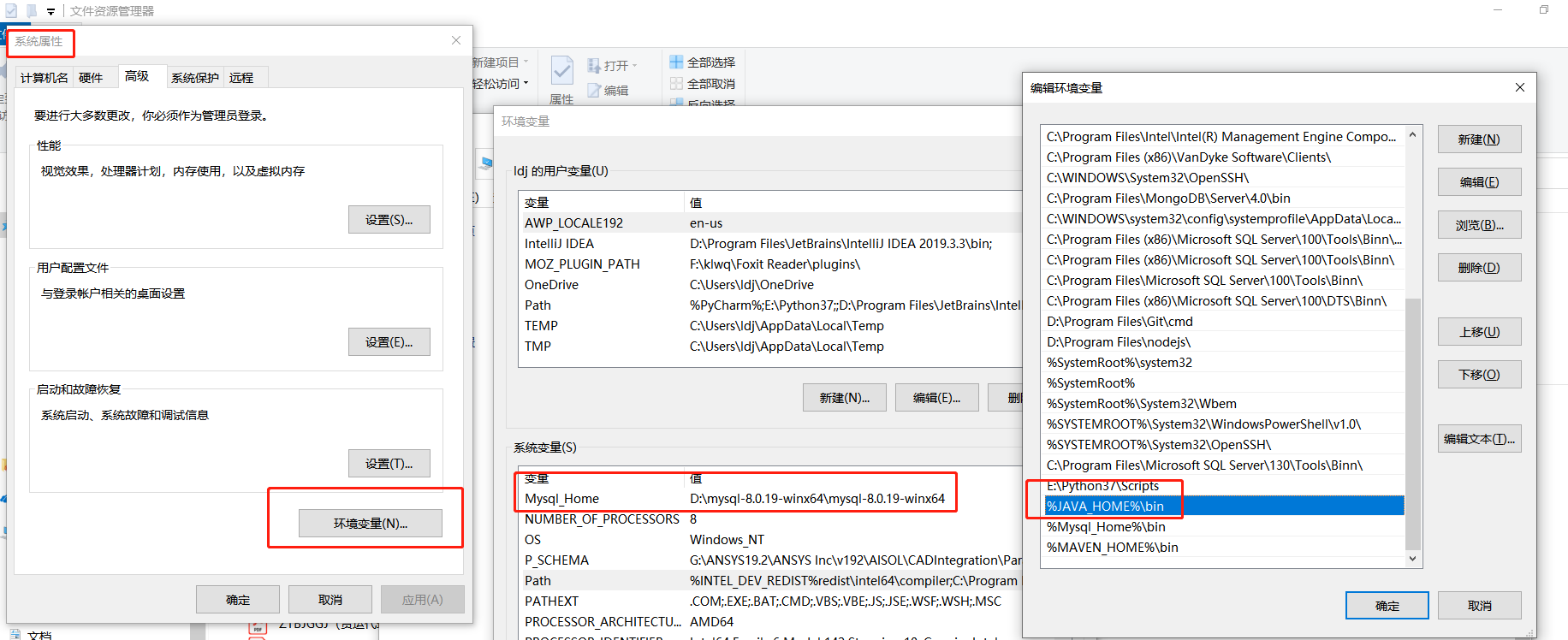
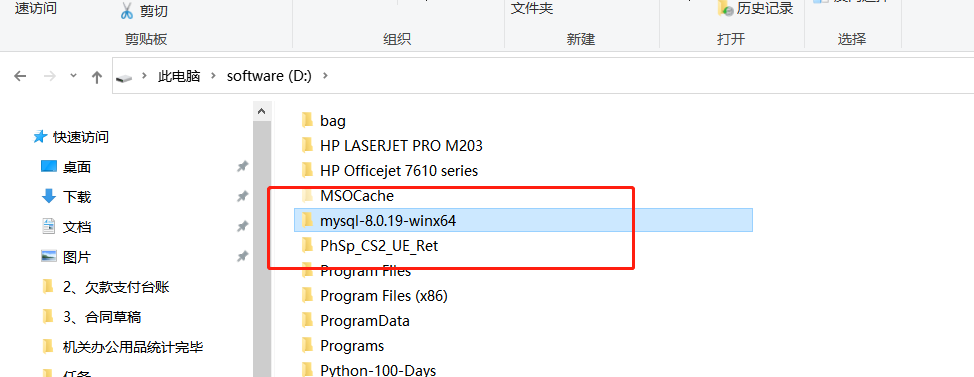
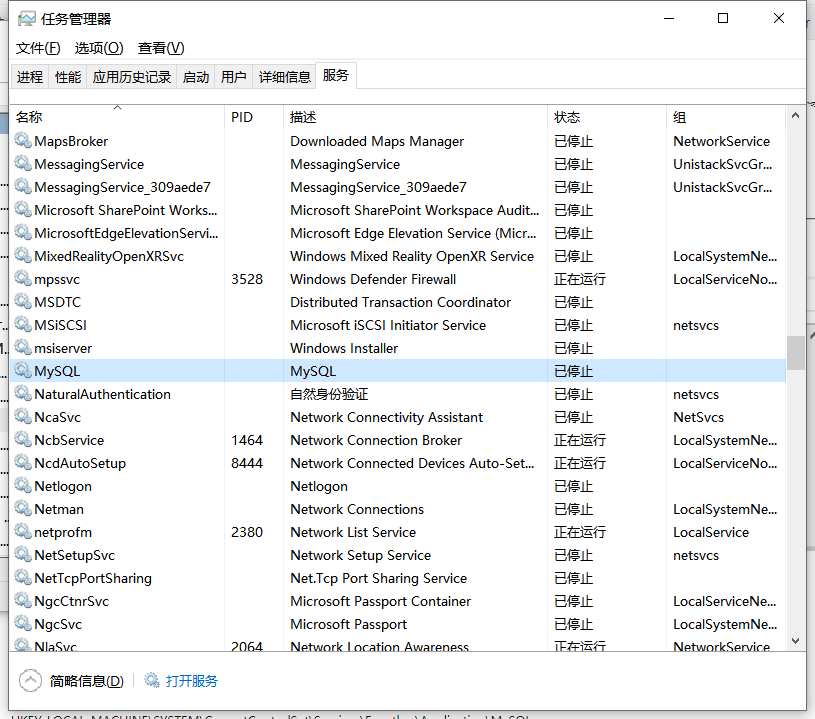
任务一

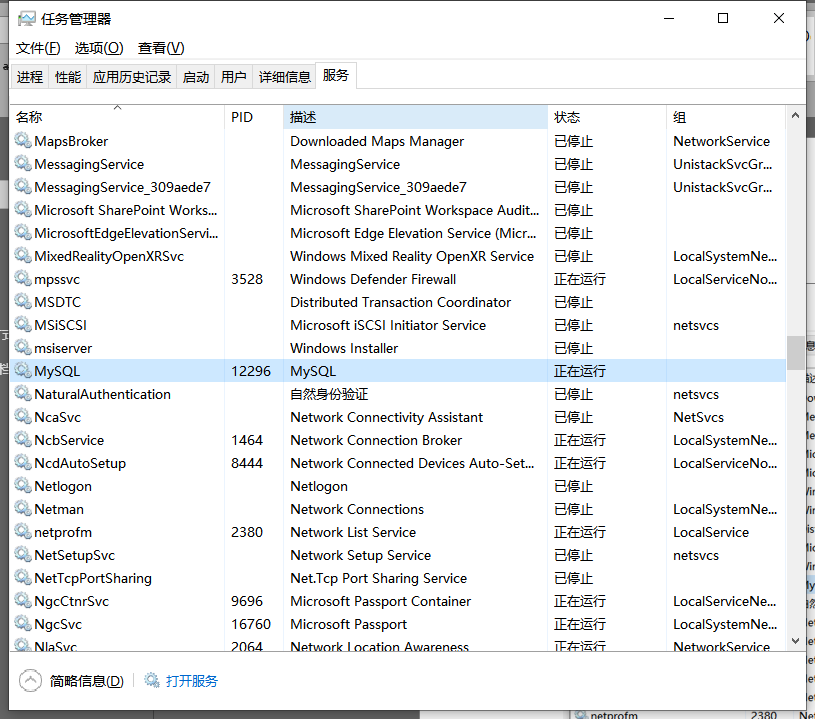
步骤15至

首先启动MYSQL服务



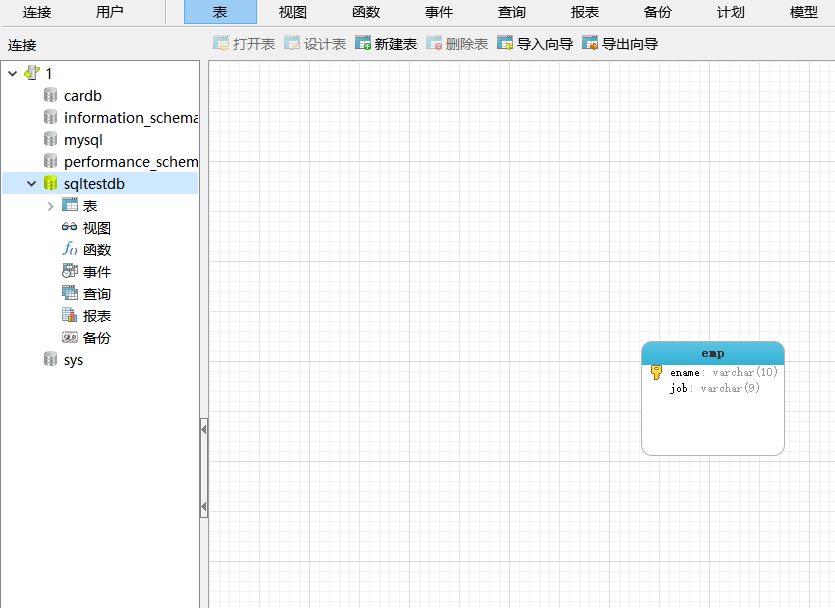




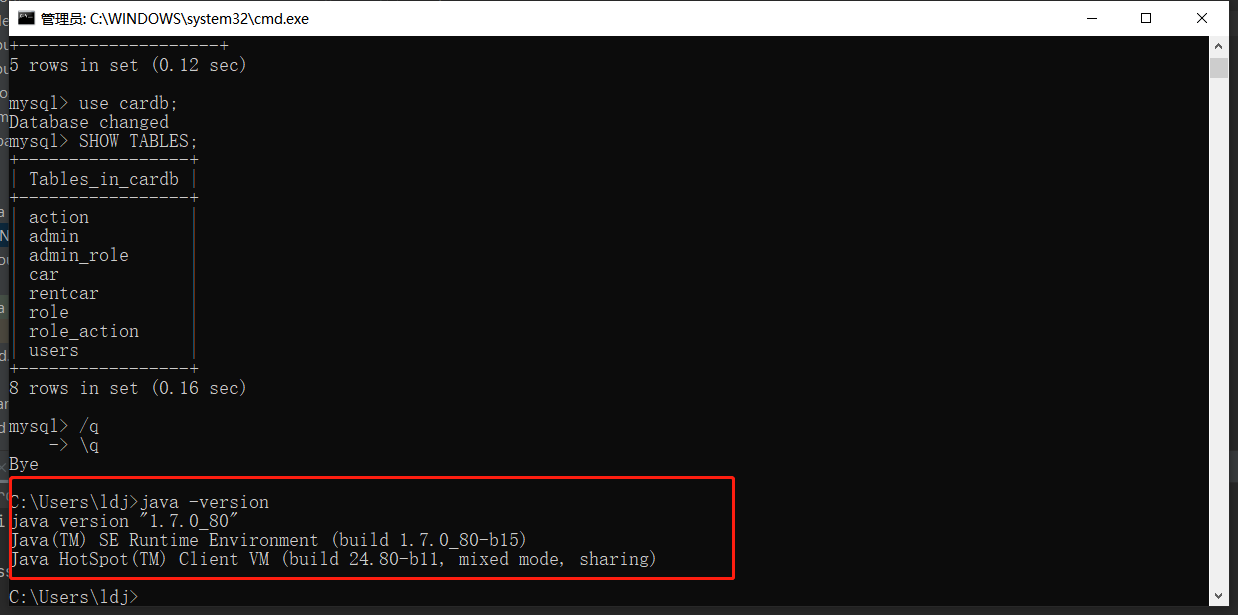
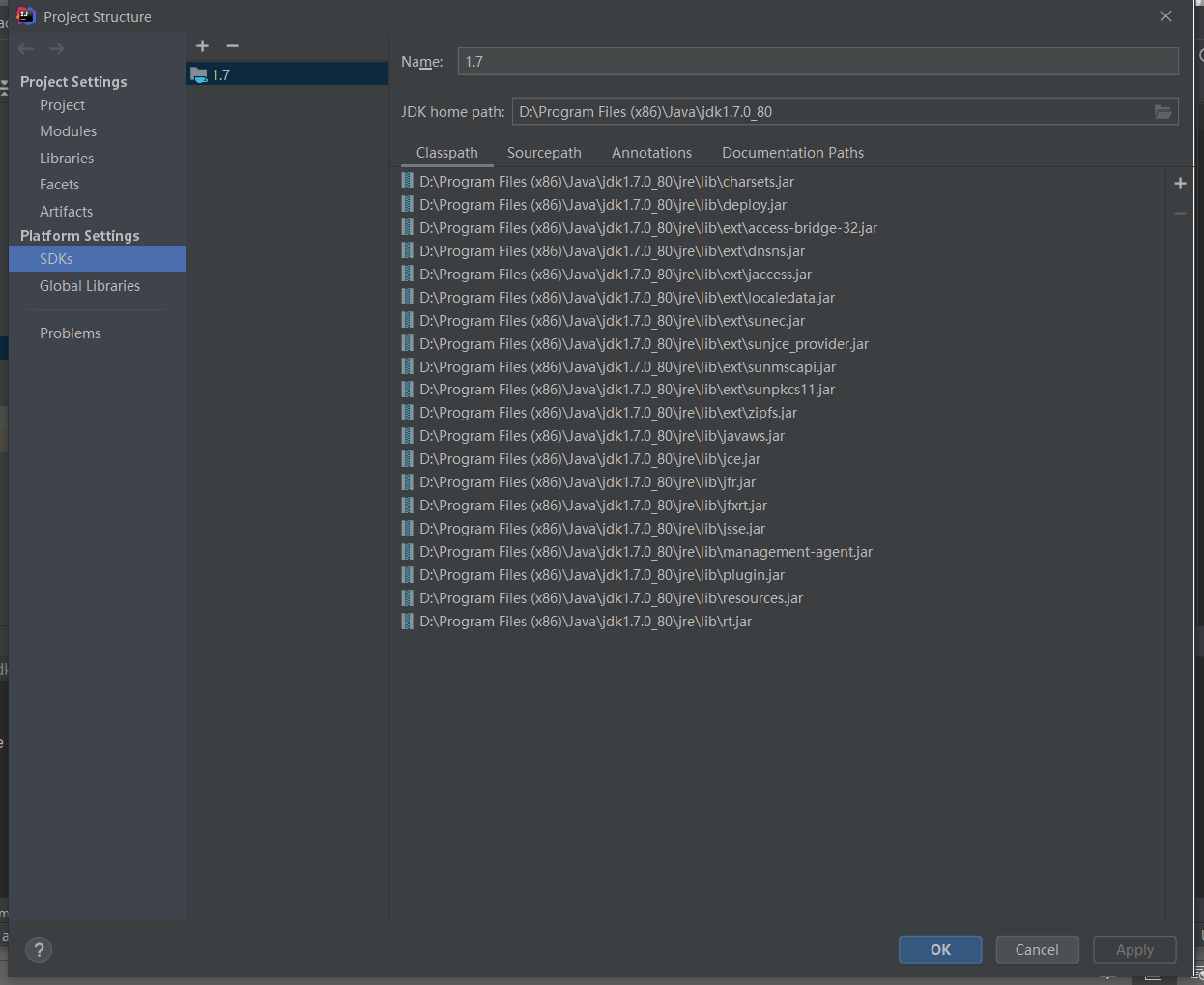




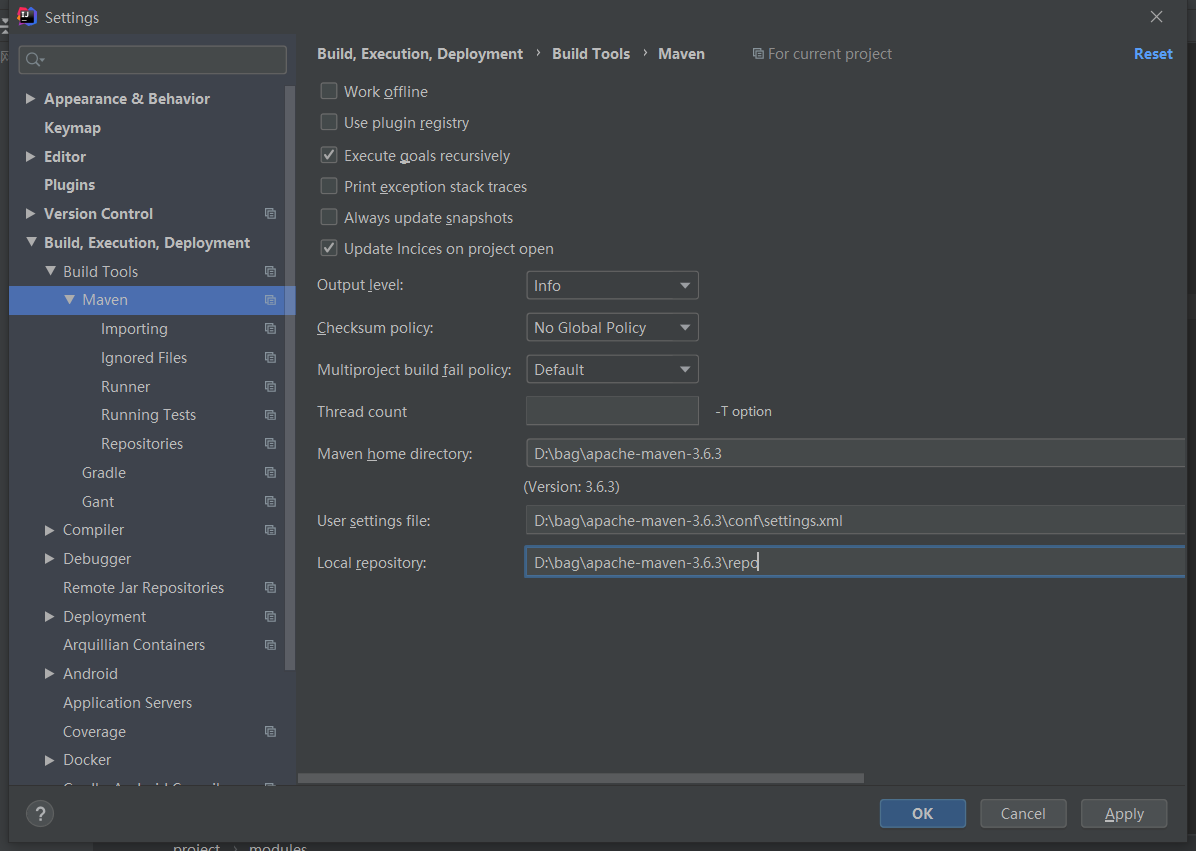
数据库内容



再检查JAVA是否就绪

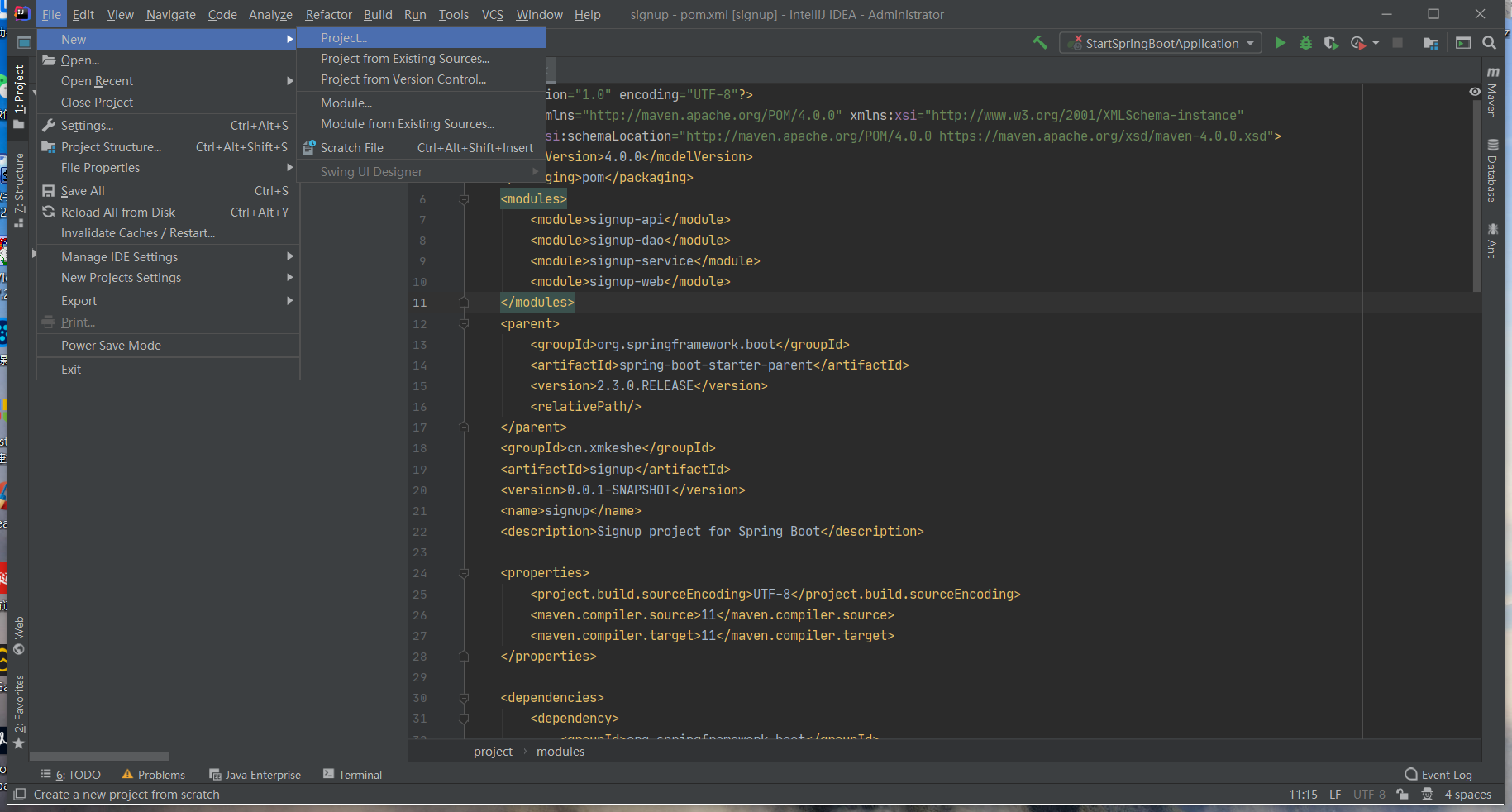


检查MAVEN配置

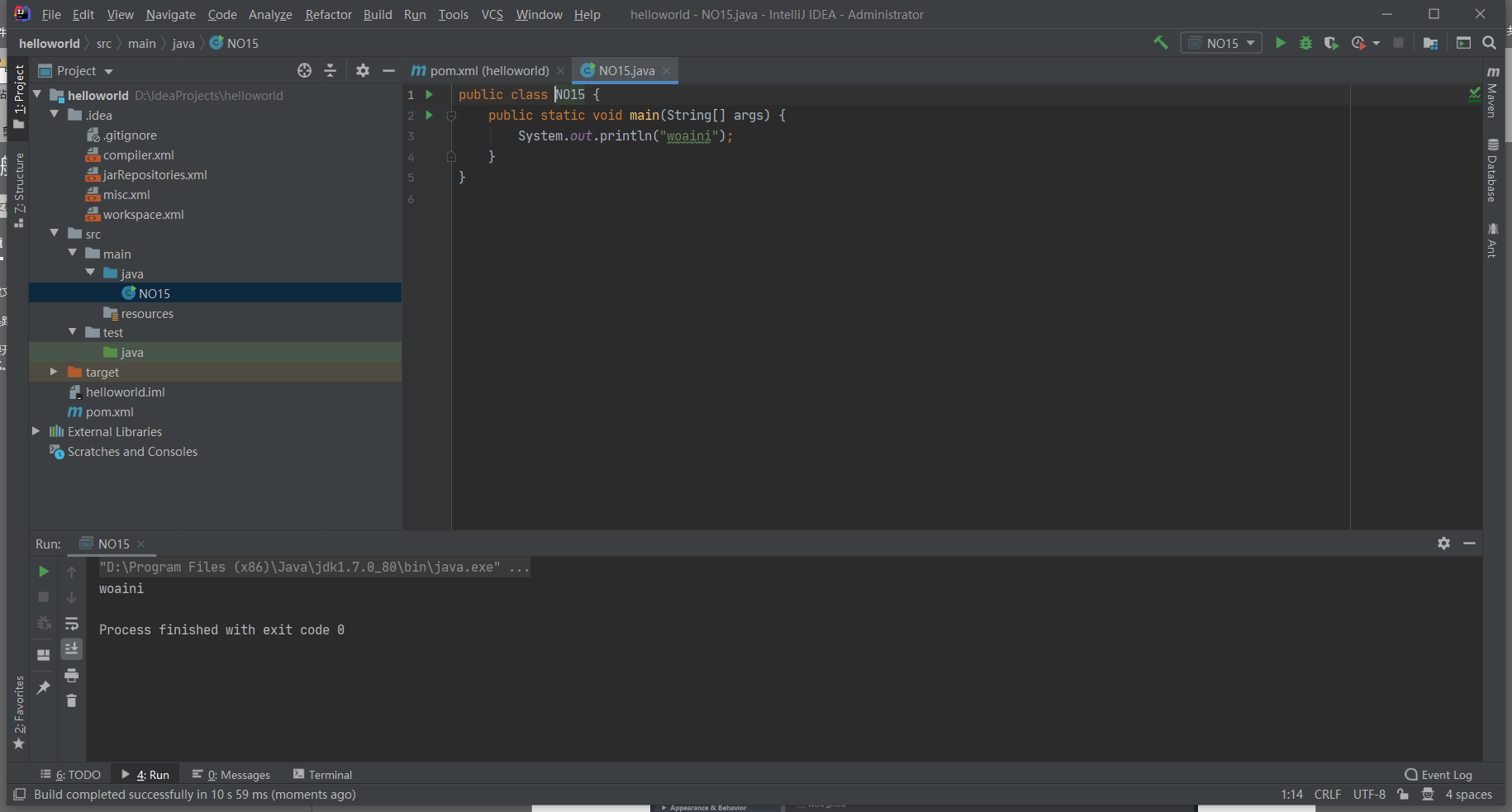




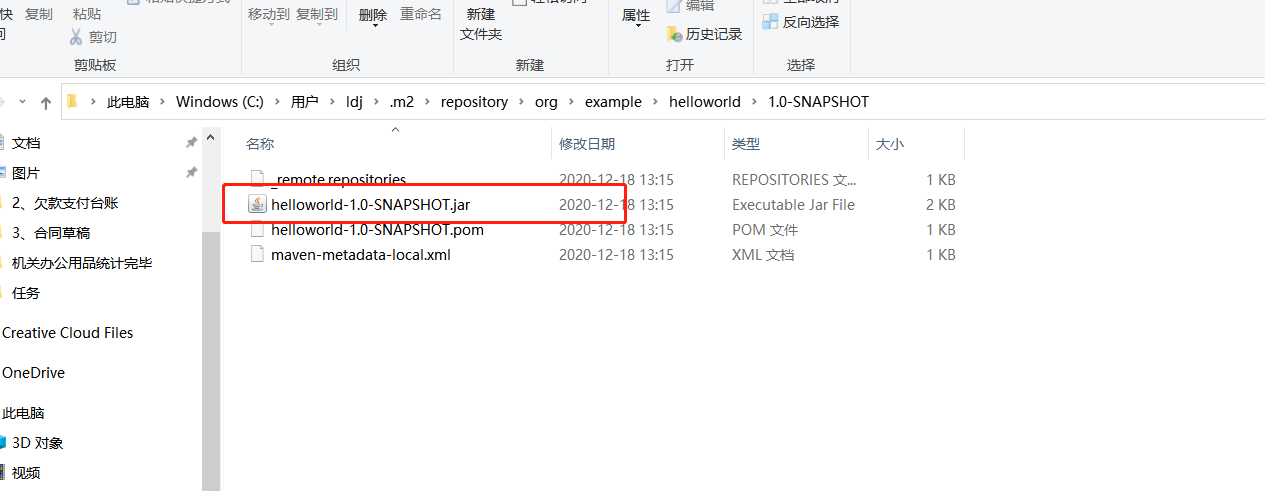
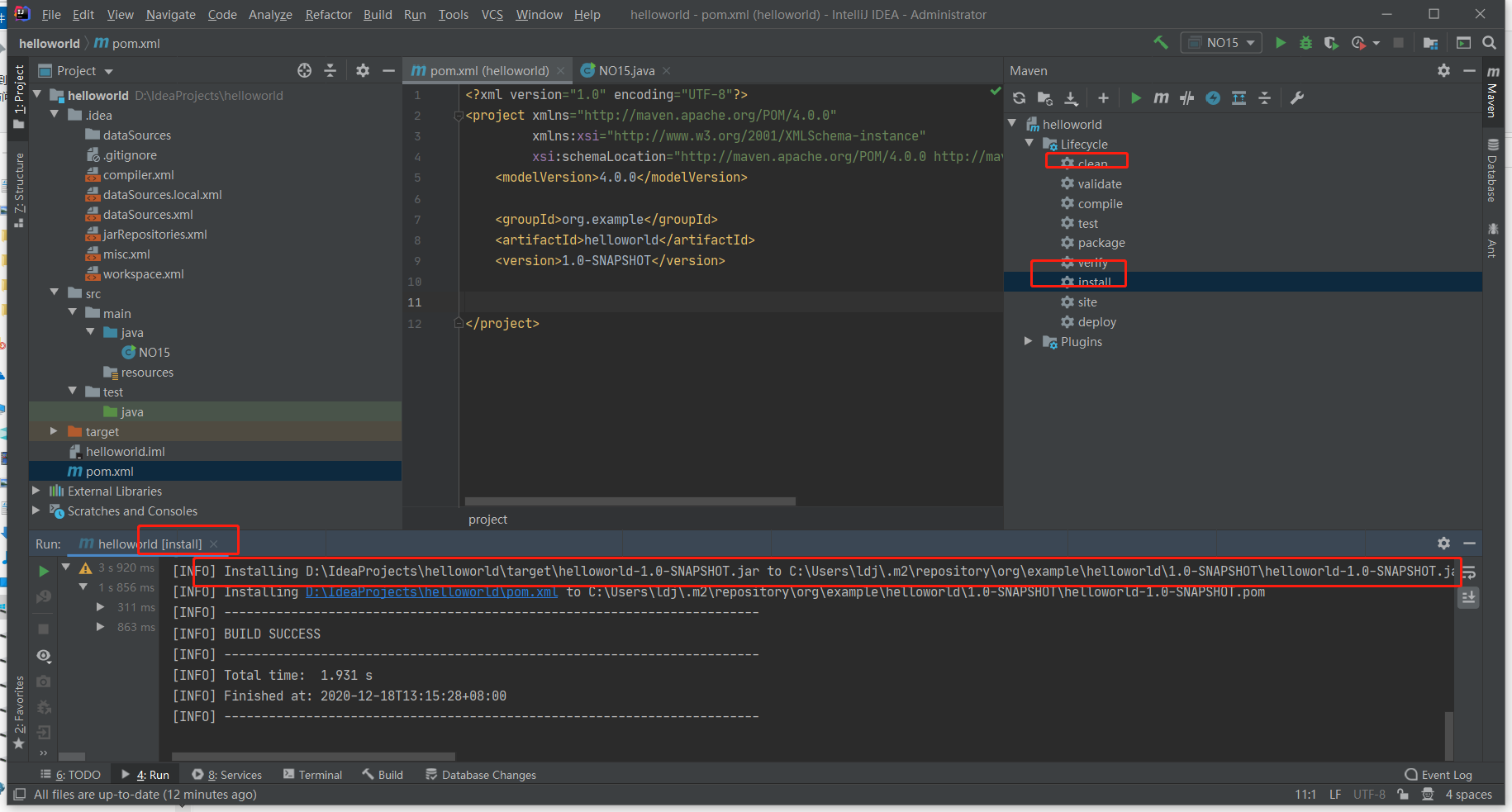
15.创建一个新的maven项目





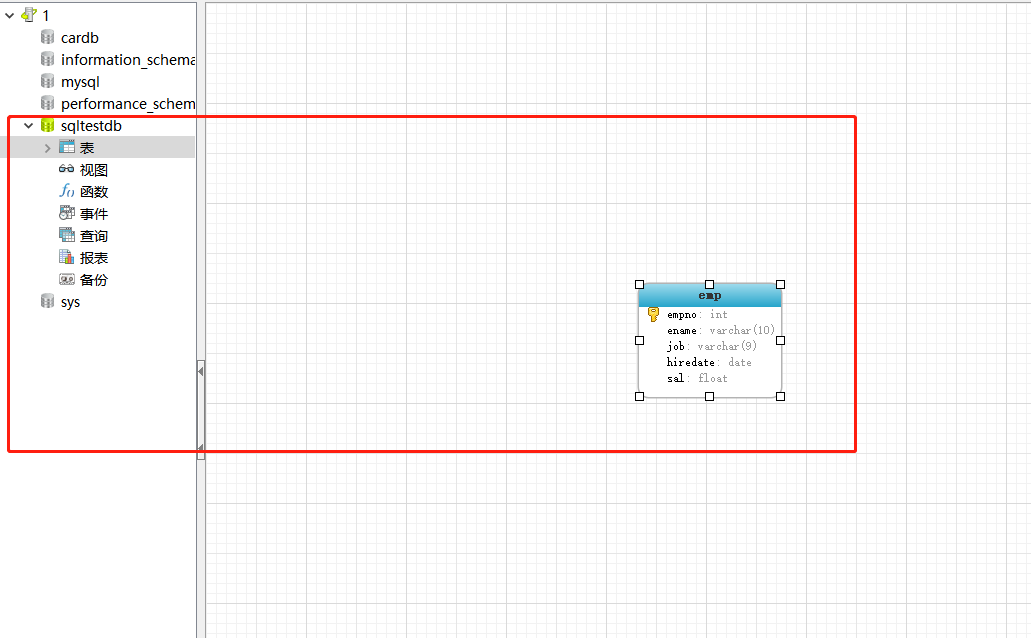


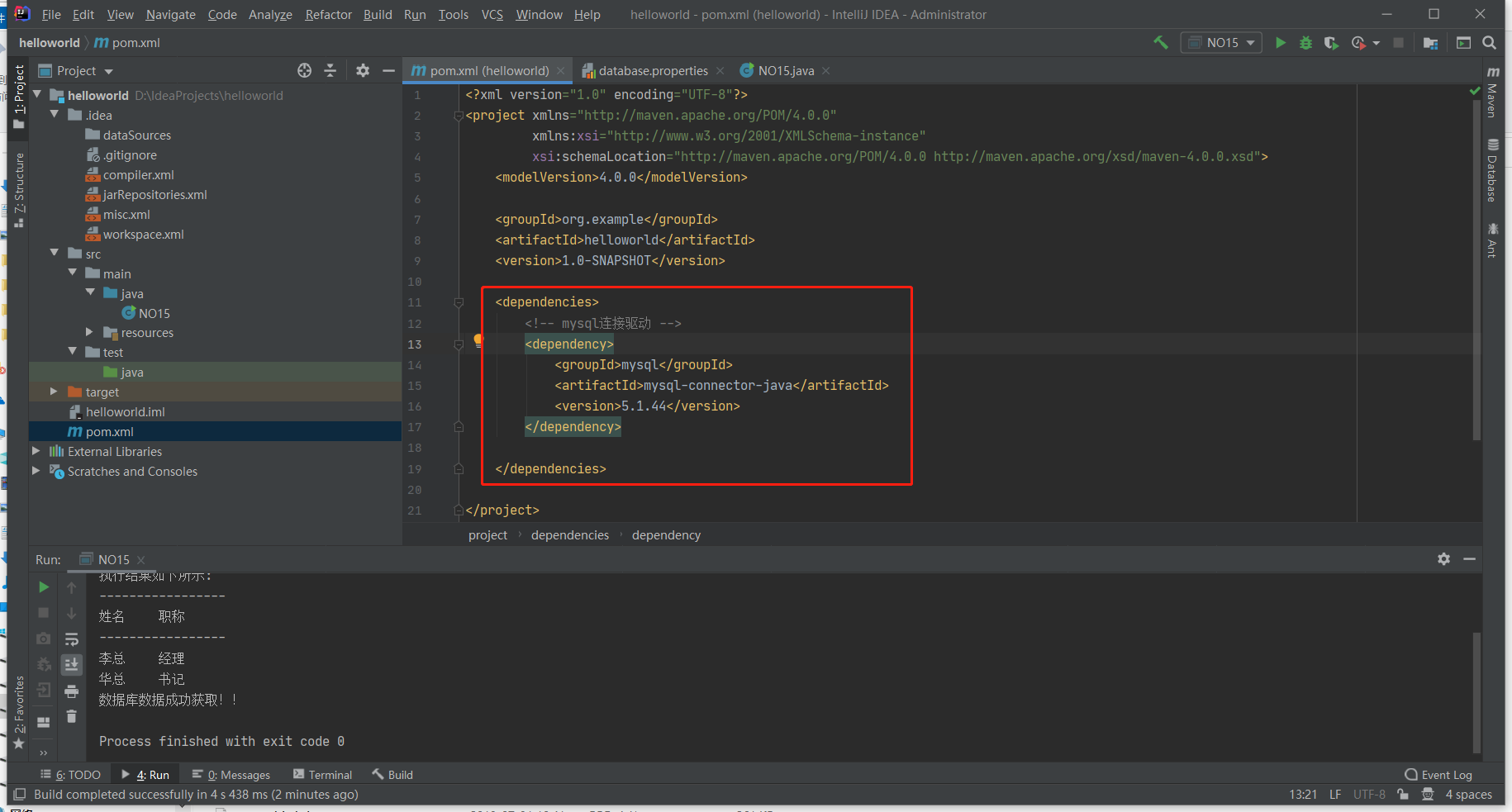
在intellij idea环境下，使用maven打jar包



17、编写DAO，分别JdbcTemplate和Mybatis连接数据库

Jdbc连接数据库





import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class NO15 {

public static void main(String[] args) {

System.out.println("woaini");

//声明Connection对象

Connection con;

//驱动程序名

String driver = "com.mysql.jdbc.Driver";

//URL指向要访问的数据库名mydata

String url = "jdbc:mysql://localhost:3306/sqltestdb?useSSL=false";

//MySQL配置时的用户名

String user = "root";

//MySQL配置时的密码

String password = "123456";

//遍历查询结果集

try {

//加载驱动程序

Class.forName(driver);

//1.getConnection()方法，连接MySQL数据库！！

con = DriverManager.getConnection(url,user,password);

if(!con.isClosed())

System.out.println("Succeeded connecting to the Database!");

//2.创建statement类对象，用来执行SQL语句！！

Statement statement = con.createStatement();

//要执行的SQL语句

String sql = "select \* from emp";

//3.ResultSet类，用来存放获取的结果集！！

ResultSet rs = statement.executeQuery(sql);

System.out.println("-----------------");

System.out.println("执行结果如下所示:");

System.out.println("-----------------");

System.out.println("姓名" + "\t" + "职称");

System.out.println("-----------------");

String job = null;

String id = null;

while(rs.next()){

//获取stuname这列数据

job = rs.getString("job");

//获取stuid这列数据

id = rs.getString("ename");

//输出结果

System.out.println(id + "\t" + job);

}

rs.close();

con.close();

} catch(ClassNotFoundException e) {

//数据库驱动类异常处理

System.out.println("Sorry,can`t find the Driver!");

e.printStackTrace();

} catch(SQLException e) {

//数据库连接失败异常处理

e.printStackTrace();

}catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}finally{

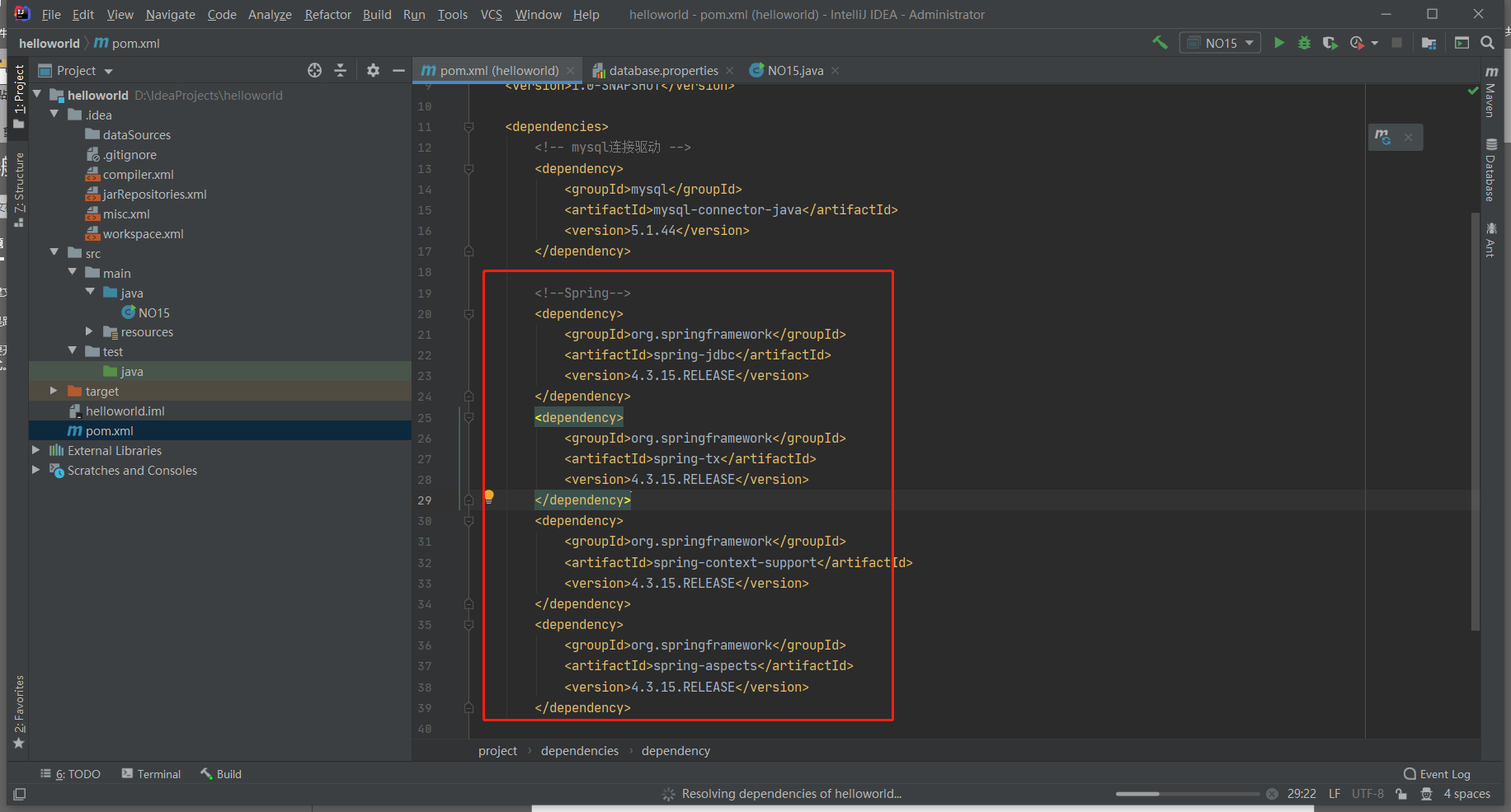
System.out.println("数据库数据成功获取！！");

}

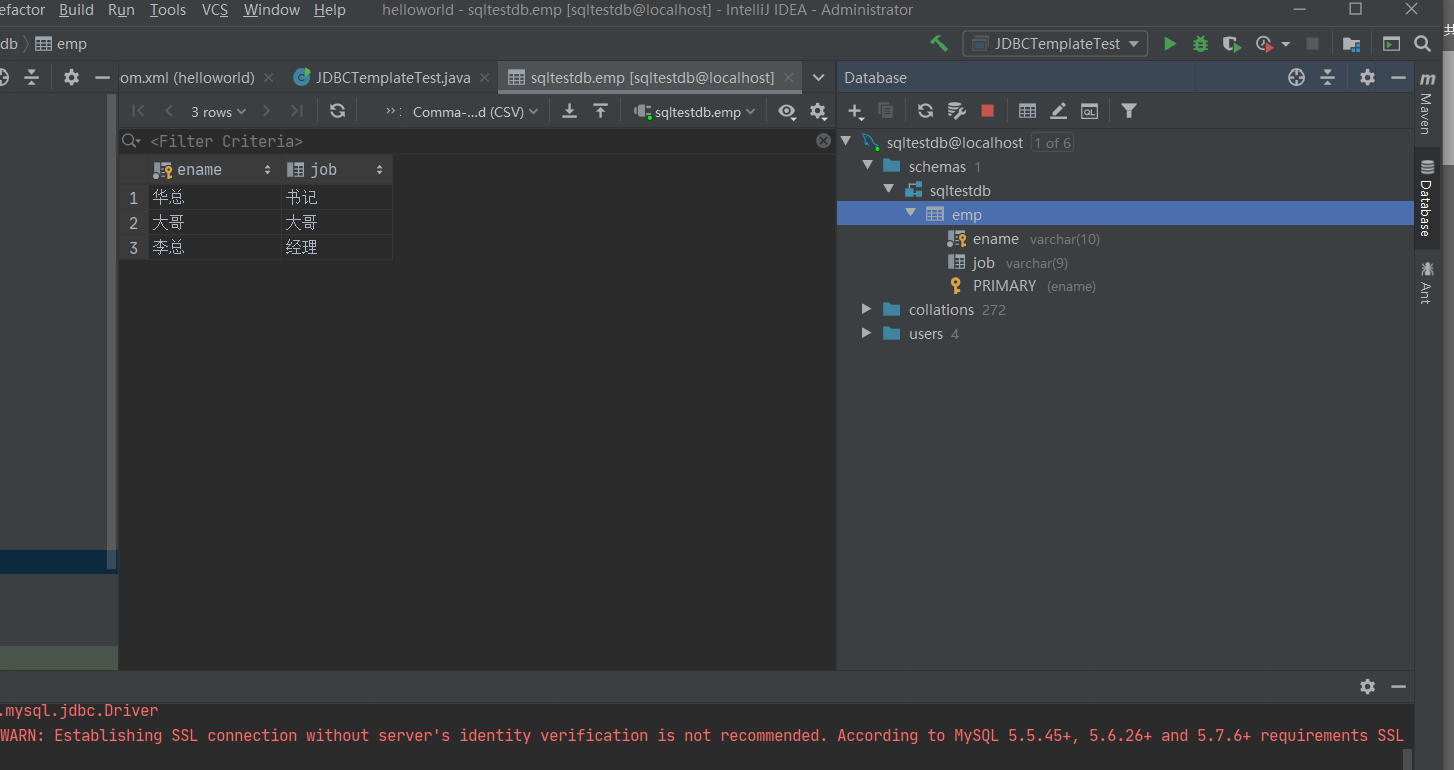
}

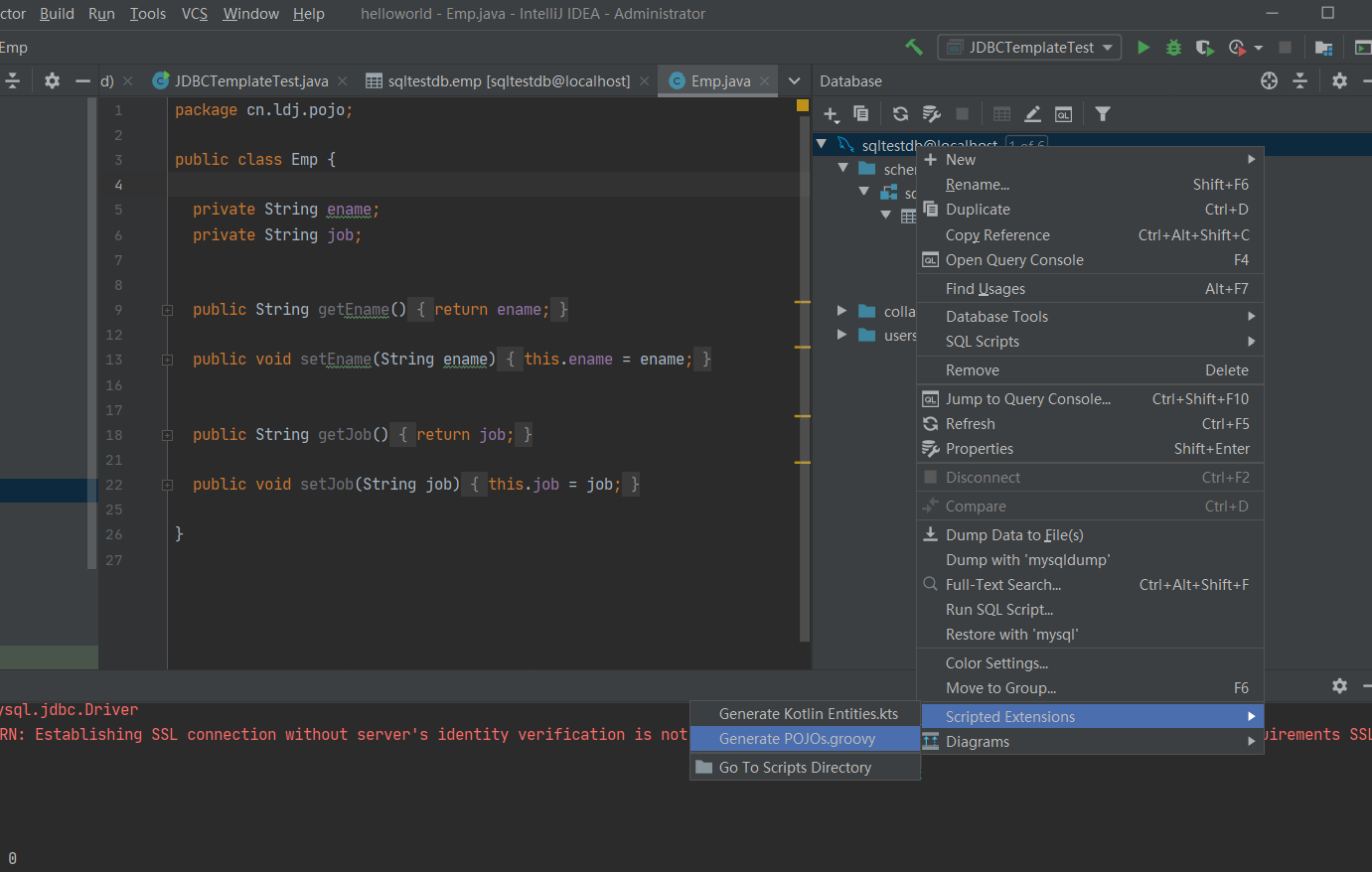
}

JdbcTemplate连接数据库（最简单的数据持久层方案）



用IDEA直接连接MYSQL数据库并生成pojo实体类



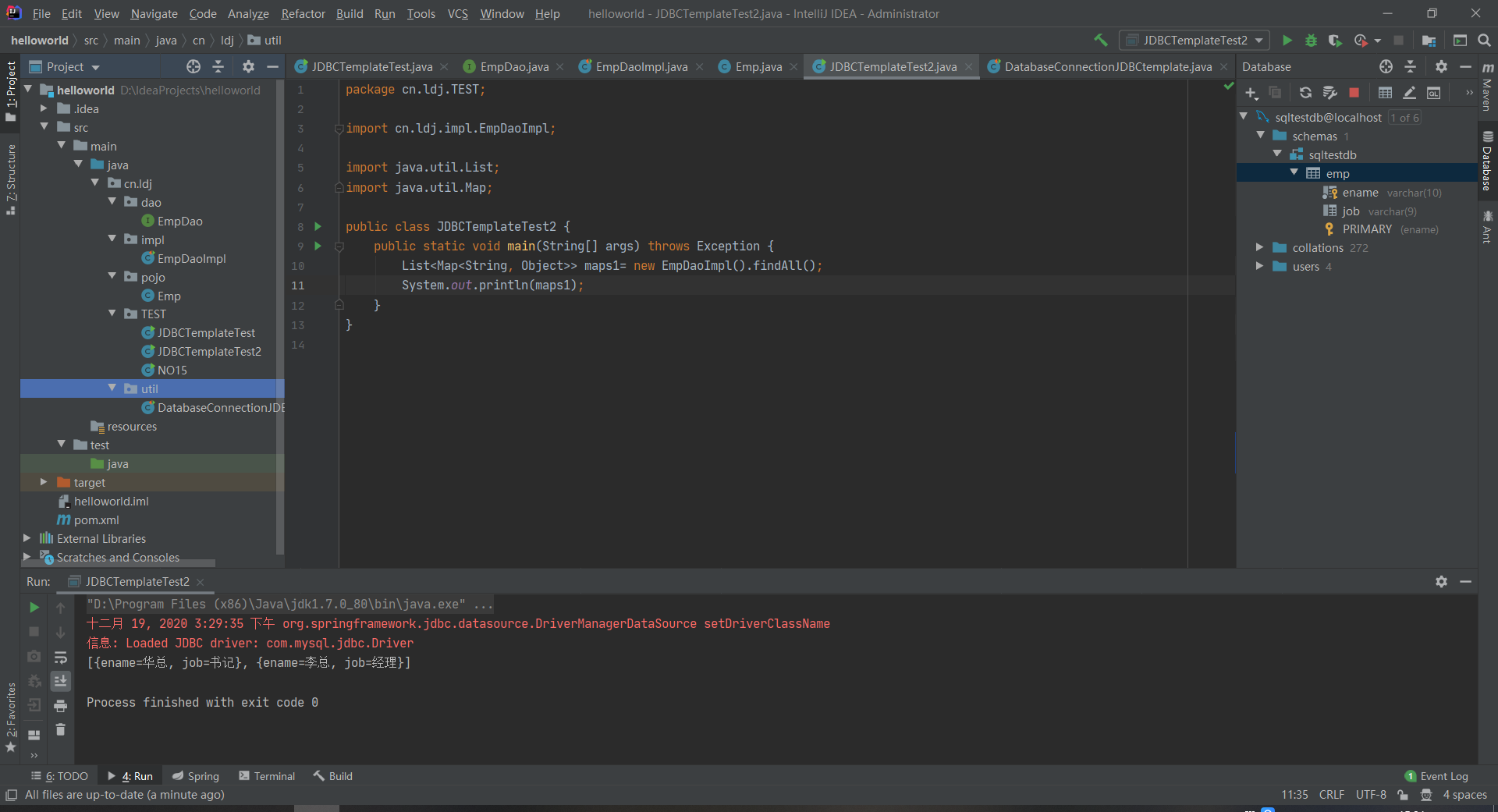


JdbcTemplate实现CRUD



package cn.ldj.TEST;  
import cn.ldj.pojo.Emp;  
import org.springframework.jdbc.core.BeanPropertyRowMapper;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import java.util.List;  
import java.util.Map;  
public class JDBCTemplateTest {  
 public static void main(String[] args) {  
 //创建数据源  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
 dataSource.setDriverClassName("com.mysql.jdbc.Driver");  
 dataSource.setUrl("jdbc:mysql://localhost:3306/sqltestdb?useSSL=false");  
 dataSource.setUsername("root");  
 dataSource.setPassword("123456");  
  
 //创建JdbcTemplate对象，设置数据源  
 JdbcTemplate template = new JdbcTemplate(dataSource);  
  
 /\*插入数据template.update("insert into emp values(?,?)","大哥","大哥");\*/  
 /\*修改数据template.update("UPDATE emp SET ename = '修改名字' WHERE job = '大哥'");\*/  
 /\*查询所有并返回结果\*/  
 //String sql="select \* from emp";  
 //List<Map<String, Object>> maps = template.queryForList(sql);  
 //System.out.println(maps);  
 //条件查询  
 //String sql1="select \* from emp where ename = '李总'";  
 //List<Map<String, Object>> maps1 = template.queryForList(sql1);  
 //System.out.println(maps1);  
 //删除数据  
 //template.update("DELETE from emp where ename = '修改名字'");  
  
 }  
}

JdbcTemplate添加数据库工具类.DAO.impl.juit



package cn.ldj.util;  
  
import org.junit.Test;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
  
import java.util.List;  
import java.util.Map;  
  
public class DatabaseConnectionJDBCtemplate {  
  
 private static final String *DBDRIVER* = "com.mysql.jdbc.Driver";  
 private static final String *DBURL* = "jdbc:mysql://localhost:3306/sqltestdb?useSSL=false";// 数据库地址  
 private static final String *DBUSER* = "root";  
 private static final String *DBPASSWORD* = "123456";  
 private JdbcTemplate template = null;  
 public DatabaseConnectionJDBCtemplate() throws Exception {  
 //创建数据源  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
 dataSource.setDriverClassName(*DBDRIVER*);  
 dataSource.setUrl(*DBURL*);  
 dataSource.setUsername(*DBUSER*);  
 dataSource.setPassword(*DBPASSWORD*);  
 //创建JdbcTemplate对象，设置数据源  
 this.template = new JdbcTemplate(dataSource); // 加载驱动// 连接数据库  
 }  
 public JdbcTemplate getJdbcTemplate() {  
 return this.template;// 取连接  
 }  
  
 @Test  
 public void test(){  
 //测试该工具类  
 String sql1="select \* from emp where ename = '李总'";  
 List<Map<String, Object>> maps1 = this.getJdbcTemplate().queryForList(sql1);  
 System.*out*.println(maps1);  
 }  
  
  
}

package cn.ldj.dao;  
  
import cn.ldj.pojo.Emp;  
  
import java.util.List;  
import java.util.Map;  
  
public interface EmpDao {  
  
 */\*\*  
  
 \* 查询  
  
 \*/* public List<Map<String, Object>> findAll() throws Exception ;  
}

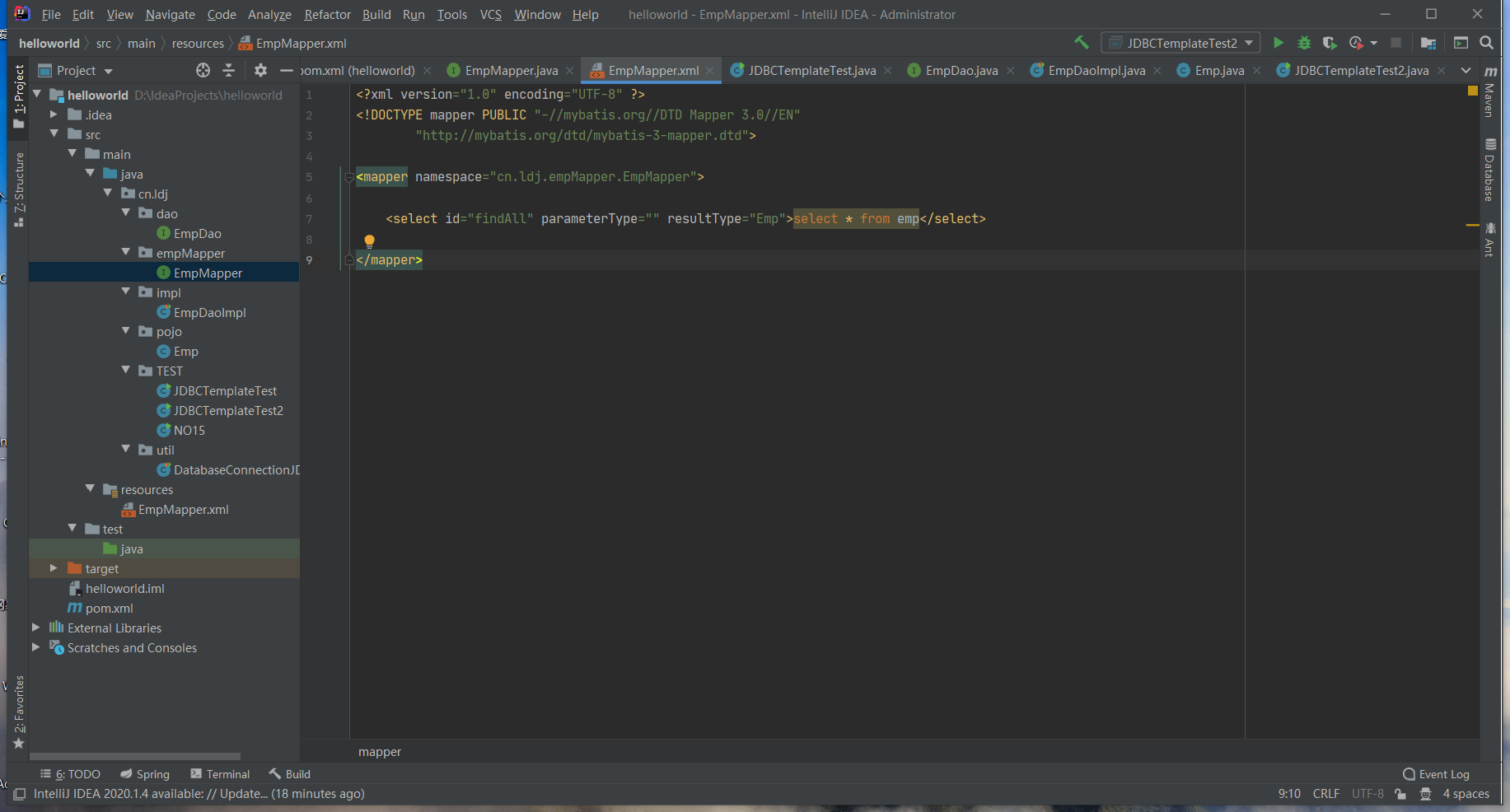
package cn.ldj.impl;  
  
import cn.ldj.dao.EmpDao;  
import cn.ldj.pojo.Emp;  
import cn.ldj.util.DatabaseConnectionJDBCtemplate;  
import org.junit.Test;  
import org.springframework.jdbc.core.JdbcTemplate;  
  
import java.util.List;  
import java.util.Map;  
  
public class EmpDaoImpl implements EmpDao {  
 private DatabaseConnectionJDBCtemplate databaseConnectionJDBCtemplate;  
 private JdbcTemplate jdbcTemplate;  
 public EmpDaoImpl() throws Exception {//构造方法中实例化连接与实例化DAO对象  
 this.databaseConnectionJDBCtemplate = new DatabaseConnectionJDBCtemplate();//连接数据库  
 this.jdbcTemplate = databaseConnectionJDBCtemplate.getJdbcTemplate();  
 }  
  
 @Override  
 public List<Map<String, Object>> findAll() throws Exception {  
  
 String sql="select \* from emp";  
 List<Map<String, Object>> maps = jdbcTemplate.queryForList(sql);  
 return maps;  
 }  
  
 @Test  
 public void test2() throws Exception {  
 System.*out*.println(this.findAll());  
 }  
}

package cn.ldj.TEST;  
  
import cn.ldj.impl.EmpDaoImpl;  
  
import java.util.List;  
import java.util.Map;  
  
public class JDBCTemplateTest2 {  
 public static void main(String[] args) throws Exception {  
 //测试查询  
 List<Map<String, Object>> maps1= new EmpDaoImpl().findAll();  
 System.*out*.println(maps1);  
 }  
}

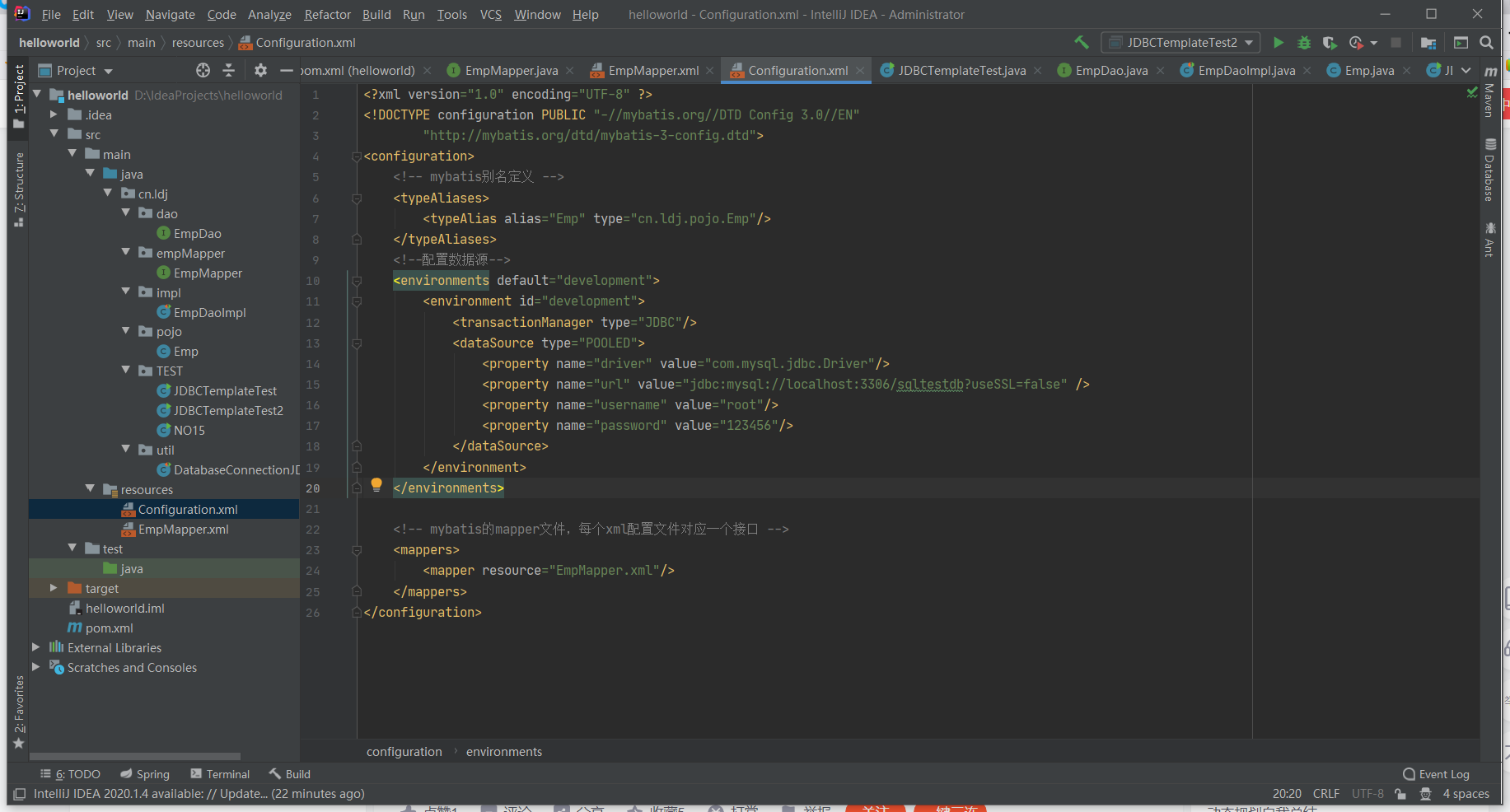
Mybatis连接数据库

<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.4.2</version>  
</dependency>

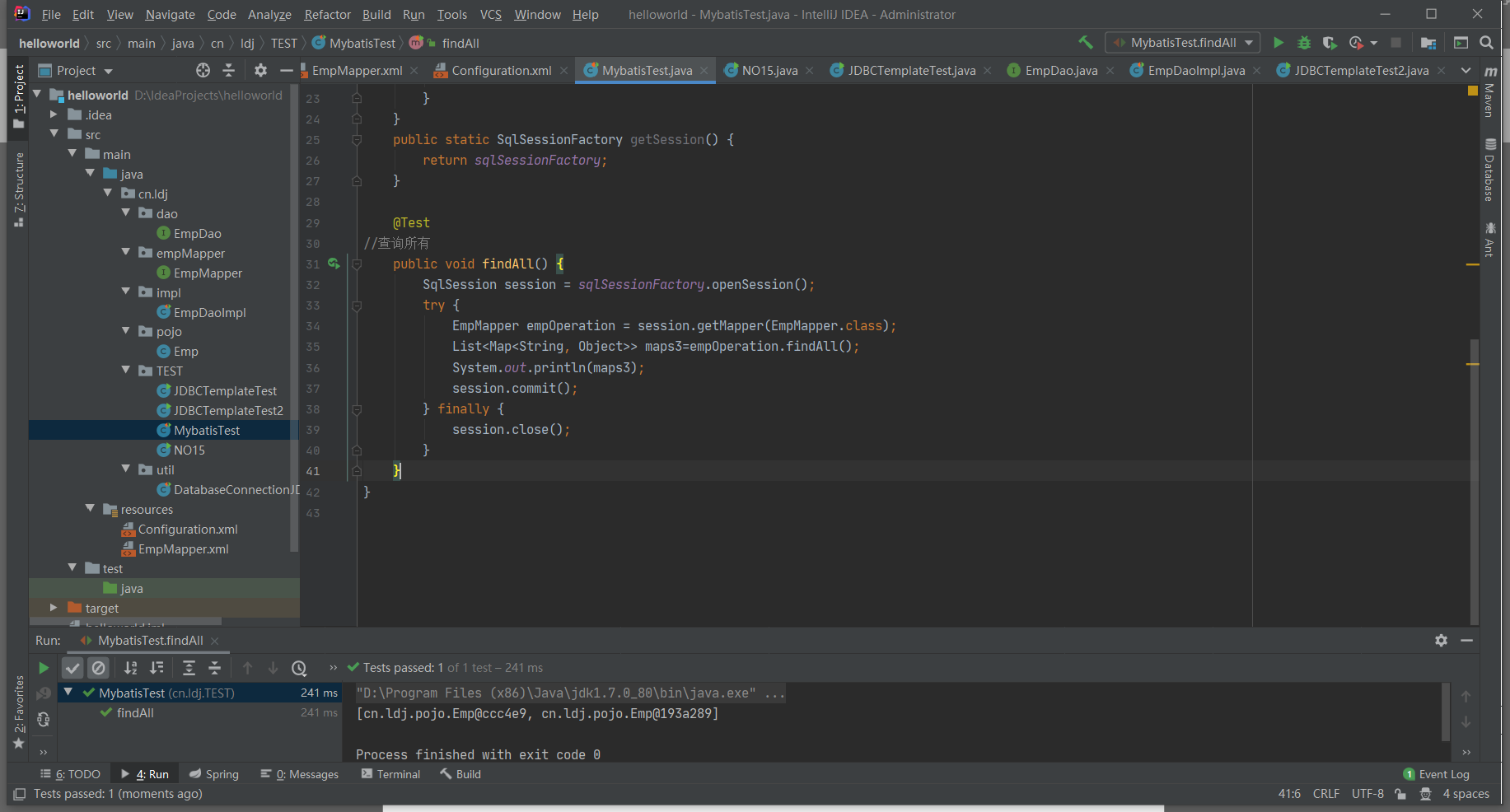
package cn.ldj.empMapper;  
  
import java.util.List;  
import java.util.Map;  
  
public interface EmpMapper {  
 public List<Map<String, Object>> findAll();  
}



<?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">  
  
<mapper namespace="cn.ldj.empMapper.EmpMapper">  
  
 <select id="findAll" resultType="Emp">select \* from emp</select>  
  
</mapper>



<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <!-- mybatis别名定义 -->  
 <typeAliases>  
 <typeAlias alias="Emp" type="cn.ldj.pojo.Emp"/>  
 </typeAliases>  
 <!--配置数据源-->  
 <environments default="development">  
 <environment id="development">  
 <transactionManager type="JDBC"/>  
 <dataSource type="POOLED">  
 <property name="driver" value="com.mysql.jdbc.Driver"/>  
 <property name="url" value="jdbc:mysql://localhost:3306/sqltestdb?useSSL=false" />  
 <property name="username" value="root"/>  
 <property name="password" value="123456"/>  
 </dataSource>  
 </environment>  
 </environments>  
  
 <!-- mybatis的mapper文件，每个xml配置文件对应一个接口 -->  
 <mappers>  
 <mapper resource="EmpMapper.xml"/>  
 </mappers>  
</configuration>



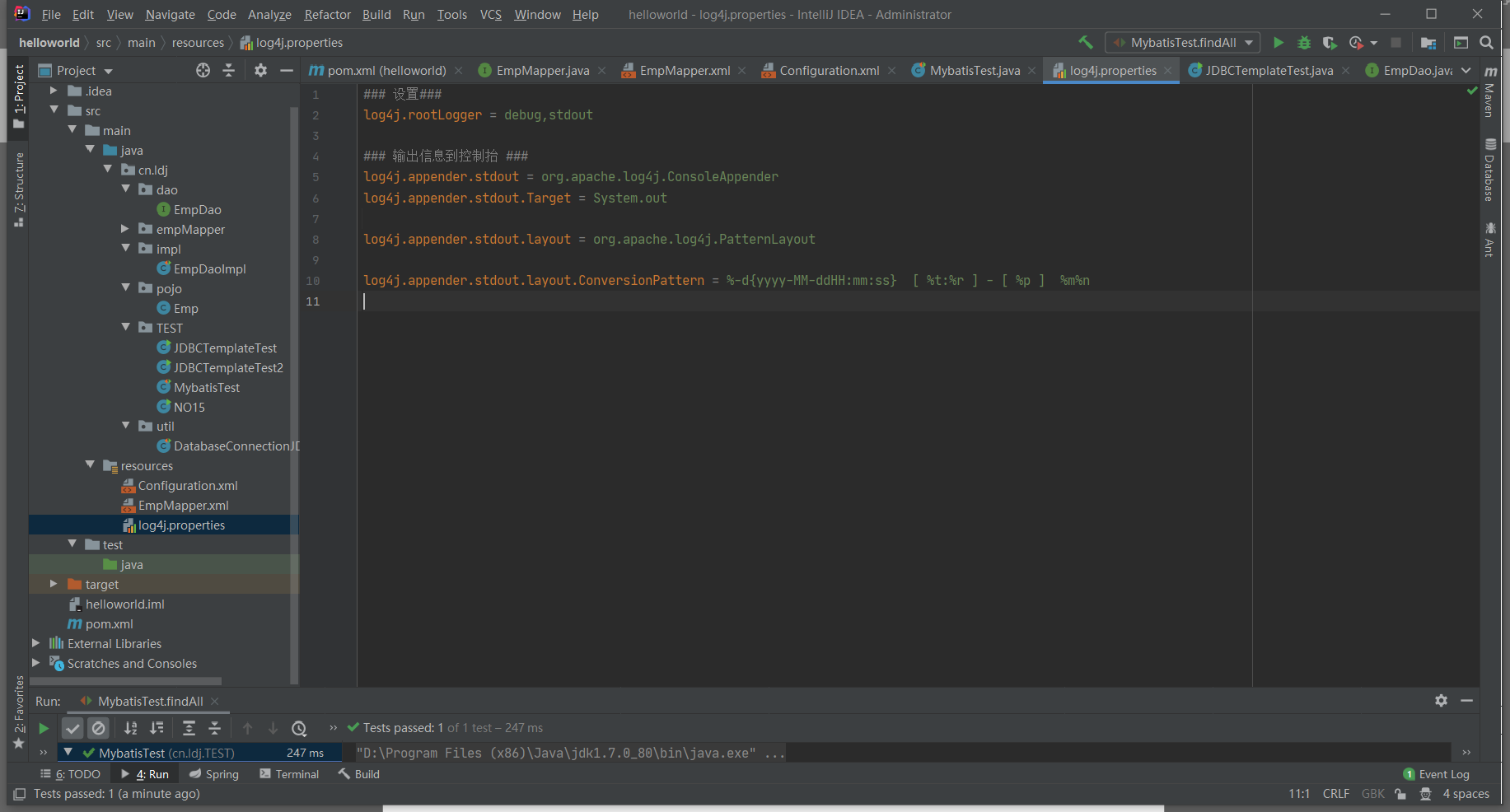
package cn.ldj.TEST;  
  
import cn.ldj.empMapper.EmpMapper;  
import org.apache.ibatis.io.Resources;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
import org.junit.Test;  
  
import java.io.Reader;  
import java.util.List;  
import java.util.Map;  
  
public class MybatisTest {  
 private static SqlSessionFactory *sqlSessionFactory*;  
 private static Reader *reader*;  
 static {  
 try {  
 *reader* = Resources.*getResourceAsReader*("Configuration.xml");  
 *sqlSessionFactory* = new SqlSessionFactoryBuilder().build(*reader*);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public static SqlSessionFactory getSession() {  
 return *sqlSessionFactory*;  
 }  
  
 @Test  
//查询所有  
 public void findAll() {  
 SqlSession session = *sqlSessionFactory*.openSession();  
 try {  
 EmpMapper empOperation = session.getMapper(EmpMapper.class);  
 List<Map<String, Object>> maps3=empOperation.findAll();  
 System.*out*.println(maps3);  
 session.commit();  
 } finally {  
 session.close();  
 }  
 }  
}

Mybatis的时候注意理解为什么不需要Impl？

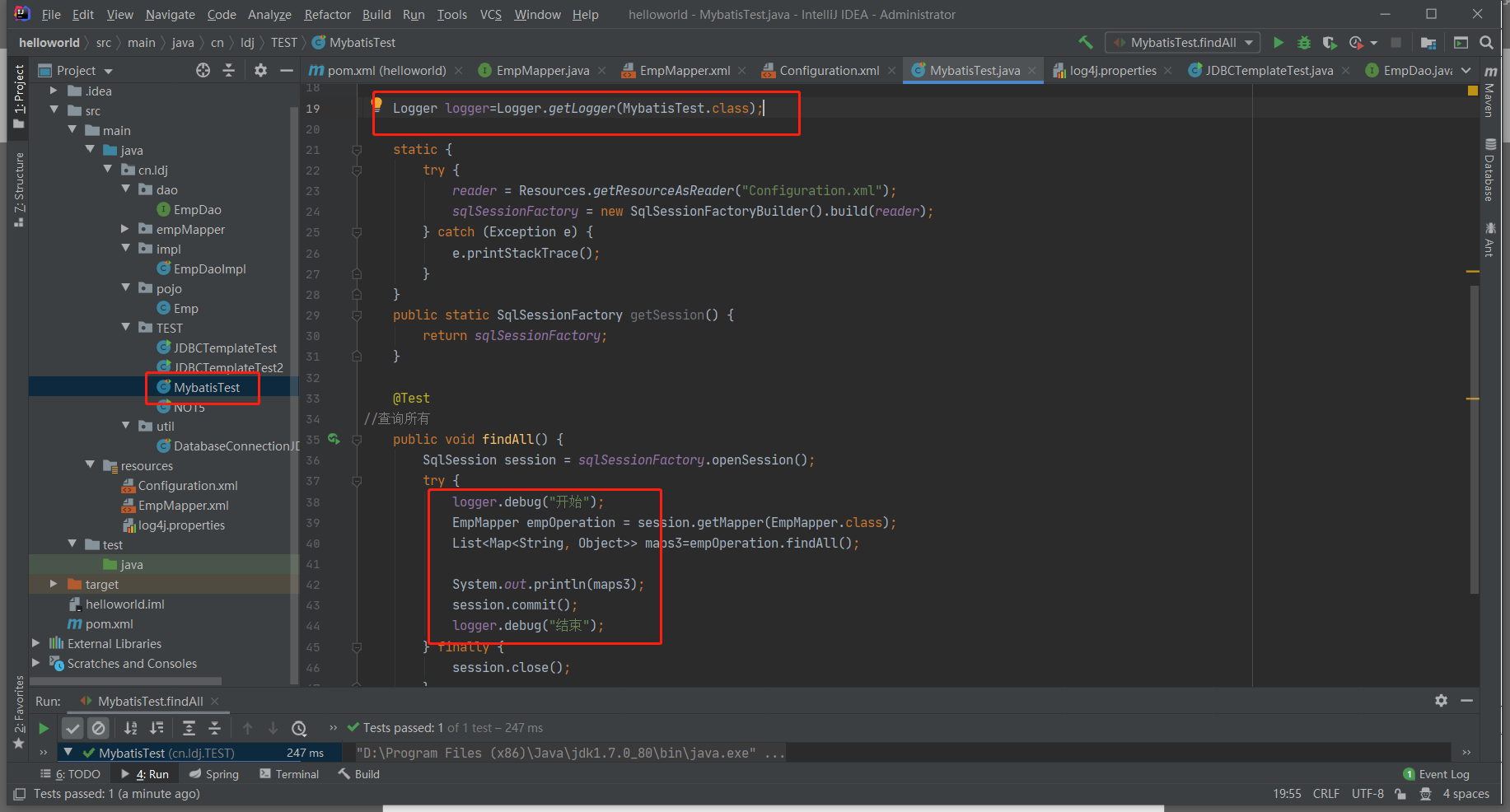
Pom.xml中引入mybatis依赖， configuration.xml链接实体类、数据库（mapper.xml中记录双方交流的内容）。Impl的功能在mybatis中通过mapper和mapper.xml实现，连接数据库在configuration.xml中实现。

21.日志LOG4J

<dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
</dependency>

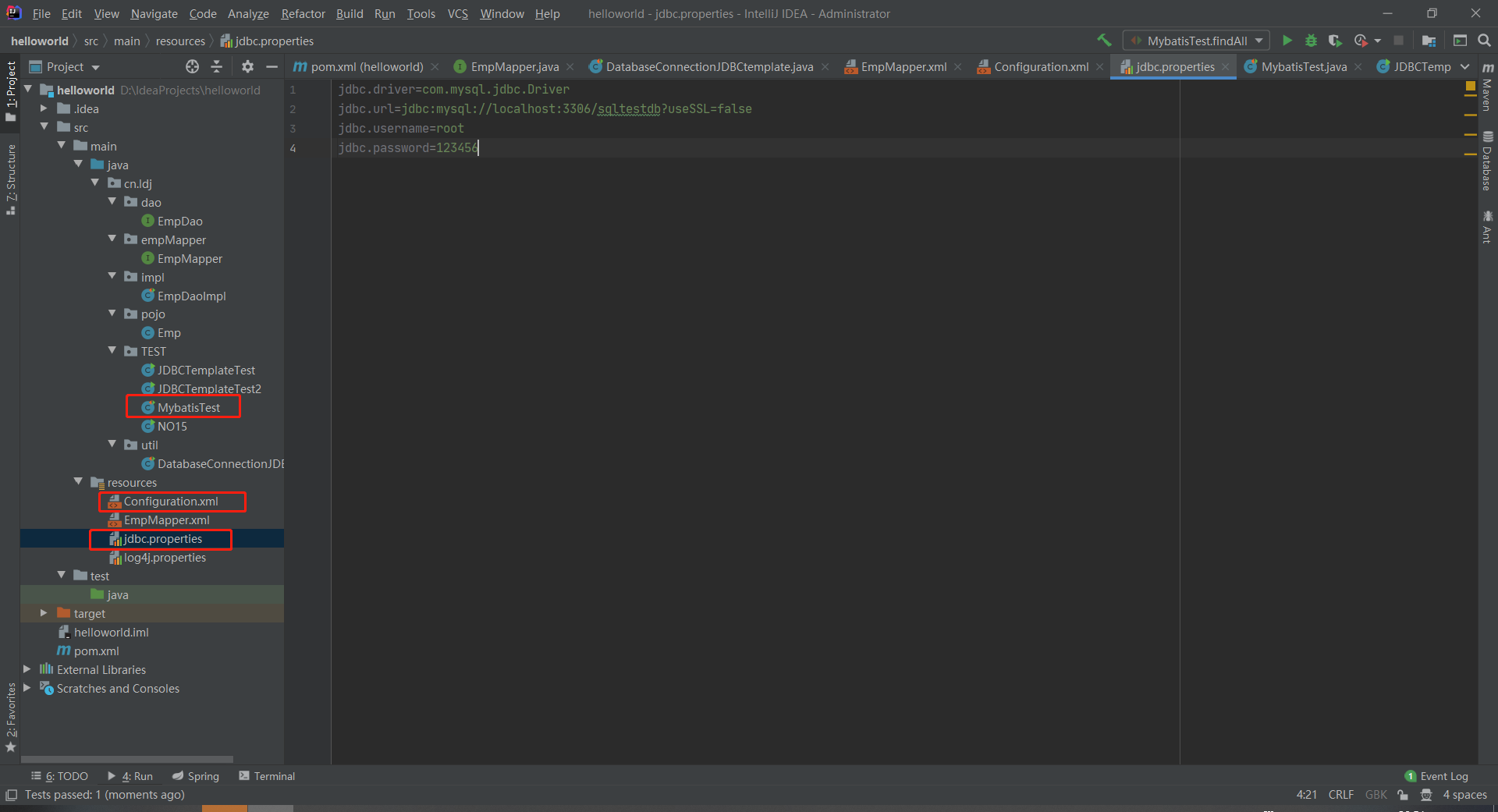


### 设置###  
log4j.rootLogger = debug,stdout  
  
### 输出信息到控制抬 ###  
log4j.appender.stdout = org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.Target = System.out  
  
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout  
  
log4j.appender.stdout.layout.ConversionPattern = %-d{yyyy-MM-ddHH:mm:ss} [ %t:%r ] - [ %p ] %m%n



package cn.ldj.TEST;  
  
import cn.ldj.empMapper.EmpMapper;  
import org.apache.ibatis.io.Resources;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
import org.junit.Test;  
import org.apache.log4j.Logger;  
  
import java.io.Reader;  
import java.util.List;  
import java.util.Map;  
  
public class MybatisTest {  
 private static SqlSessionFactory *sqlSessionFactory*;  
 private static Reader *reader*;  
  
 Logger logger=Logger.*getLogger*(MybatisTest.class);  
  
 static {  
 try {  
 *reader* = Resources.*getResourceAsReader*("Configuration.xml");  
 *sqlSessionFactory* = new SqlSessionFactoryBuilder().build(*reader*);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public static SqlSessionFactory getSession() {  
 return *sqlSessionFactory*;  
 }  
  
 @Test  
//查询所有  
 public void findAll() {  
 SqlSession session = *sqlSessionFactory*.openSession();  
 try {  
 logger.debug("开始");  
 EmpMapper empOperation = session.getMapper(EmpMapper.class);  
 List<Map<String, Object>> maps3=empOperation.findAll();  
  
 System.*out*.println(maps3);  
 session.commit();  
 logger.debug("结束");  
 } finally {  
 session.close();  
 }  
 }  
}

引用 jdbc.properties 文件，修改Configuration.xml，分离数据库连接



jdbc.driver=com.mysql.jdbc.Driver  
jdbc.url=jdbc:mysql://localhost:3306/sqltestdb?useSSL=false  
jdbc.username=root  
jdbc.password=123456

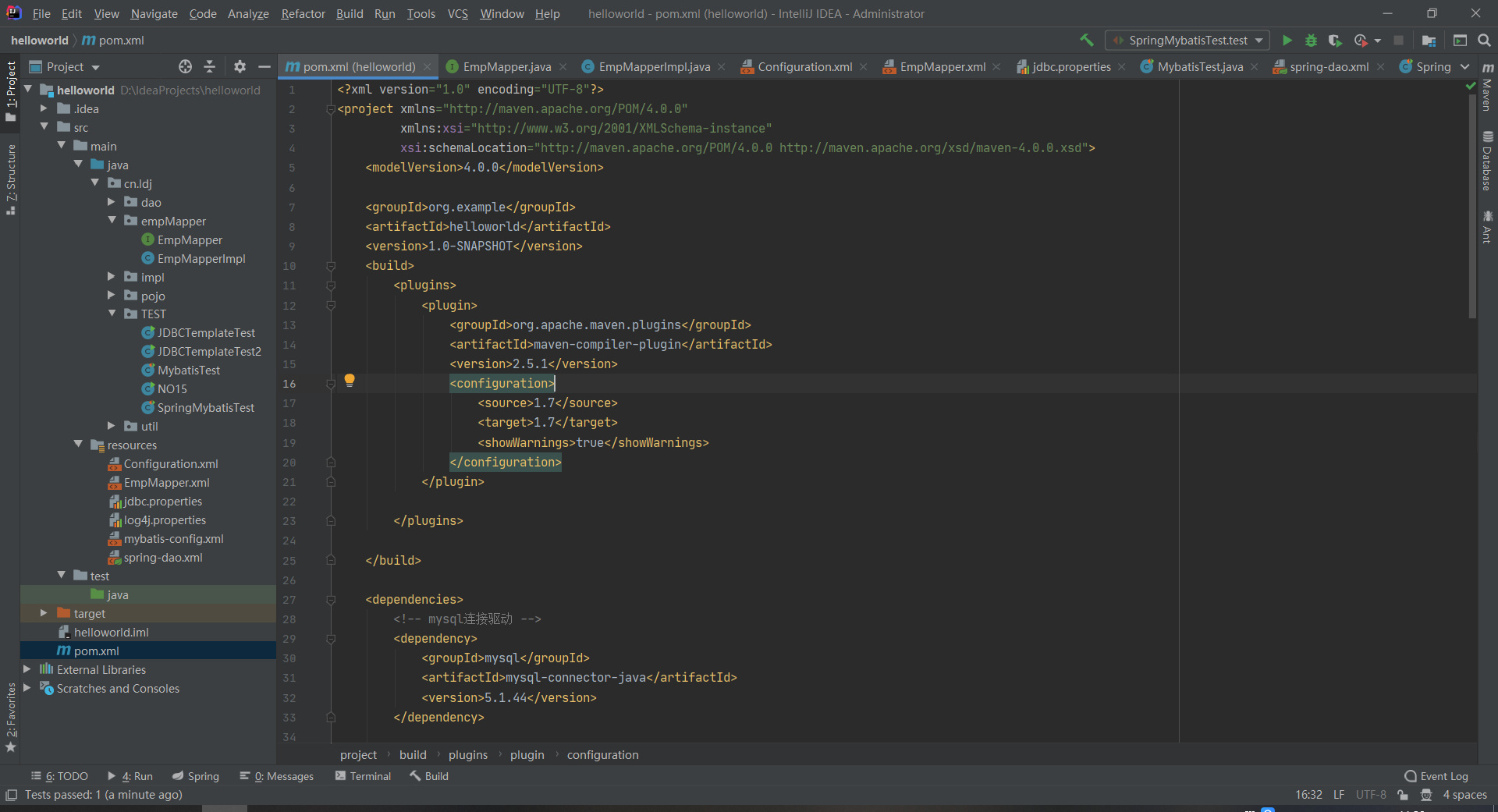
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <properties resource="jdbc.properties"/>  
 <!-- mybatis别名定义 -->  
 <typeAliases>  
 <typeAlias alias="Emp" type="cn.ldj.pojo.Emp"/>  
 </typeAliases>  
 <!--配置数据源-->  
  
 <environments default="development">  
 <environment id="development">  
 <transactionManager type="JDBC"/>  
 <dataSource type="POOLED">  
 <property name="driver" value="${jdbc.driver}"/>  
 <property name="url" value="${jdbc.url}" />  
 <property name="username" value="${jdbc.username}"/>  
 <property name="password" value="${jdbc.password}"/>  
 </dataSource>  
 </environment>  
 </environments>  
  
 <!-- mybatis的mapper文件，每个xml配置文件对应一个接口 -->  
 <mappers>  
 <mapper resource="EmpMapper.xml"/>  
 </mappers>  
</configuration>

package cn.ldj.TEST;  
  
import cn.ldj.empMapper.EmpMapper;  
import org.apache.ibatis.io.Resources;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
import org.junit.Test;  
import org.apache.log4j.Logger;  
  
import java.io.Reader;  
import java.util.List;  
import java.util.Map;  
  
public class MybatisTest {  
 private static SqlSessionFactory *sqlSessionFactory*;  
 private static Reader *reader*;  
  
 Logger logger=Logger.*getLogger*(MybatisTest.class);  
  
 static {  
 try {  
 *reader* = Resources.*getResourceAsReader*("Configuration.xml");  
 *sqlSessionFactory* = new SqlSessionFactoryBuilder().build(*reader*);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public static SqlSessionFactory getSession() {  
 return *sqlSessionFactory*;  
 }  
  
 @Test  
//查询所有  
 public void findAll() {  
 SqlSession session = *sqlSessionFactory*.openSession();  
 try {  
 logger.debug("开始");  
 EmpMapper empOperation = session.getMapper(EmpMapper.class);  
 List<Map<String, Object>> maps3=empOperation.findAll();  
  
 System.*out*.println(maps3);  
 session.commit();  
 logger.debug("结束");  
 } finally {  
 session.close();  
 }  
 }  
}

21.整合Spring+Mybatis+MYSQL

参考：https://blog.csdn.net/youif/article/details/108352943

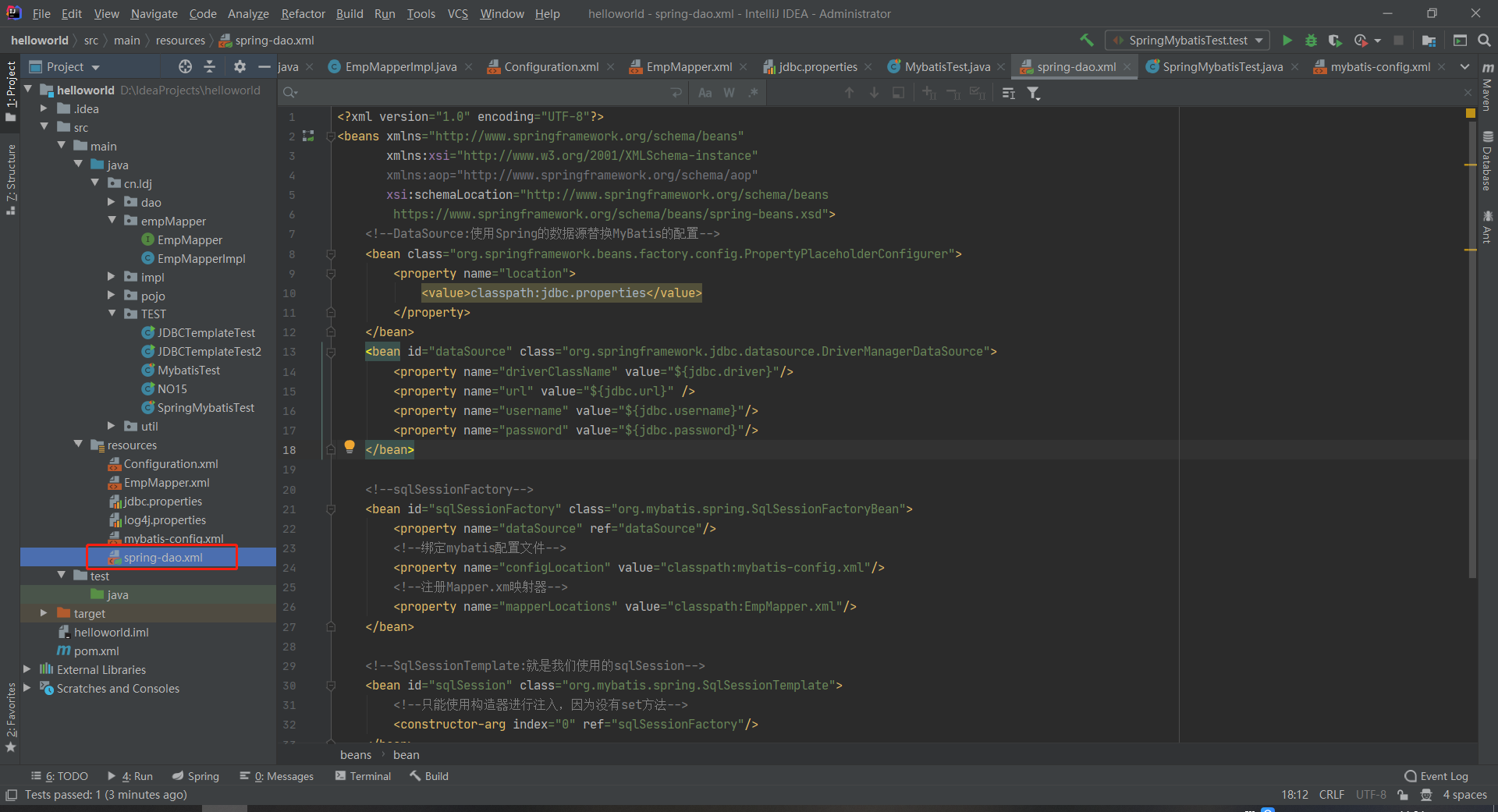
1. 修改pom.xml引入配置文件，修正所有依赖版本



<?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>org.example</groupId>  
 <artifactId>helloworld</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>2.5.1</version>  
 <configuration>  
 <source>1.7</source>  
 <target>1.7</target>  
 <showWarnings>true</showWarnings>  
 </configuration>  
 </plugin>  
  
 </plugins>  
  
 </build>  
  
 <dependencies>  
 <!-- mysql连接驱动 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.44</version>  
 </dependency>  
  
 <!--Spring-->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>3.2.9.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>3.2.9.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context-support</artifactId>  
 <version>3.2.9.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>3.2.9.RELEASE</version>  
 </dependency>  
 <!--Spring+Mybatis+Mysql-->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.7.4</version>  
 </dependency>  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>1.3.1</version>  
 </dependency>  
  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 <scope>compile</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.4.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
 </dependency>  
  
  
  
  
 </dependencies>  
  
</project>

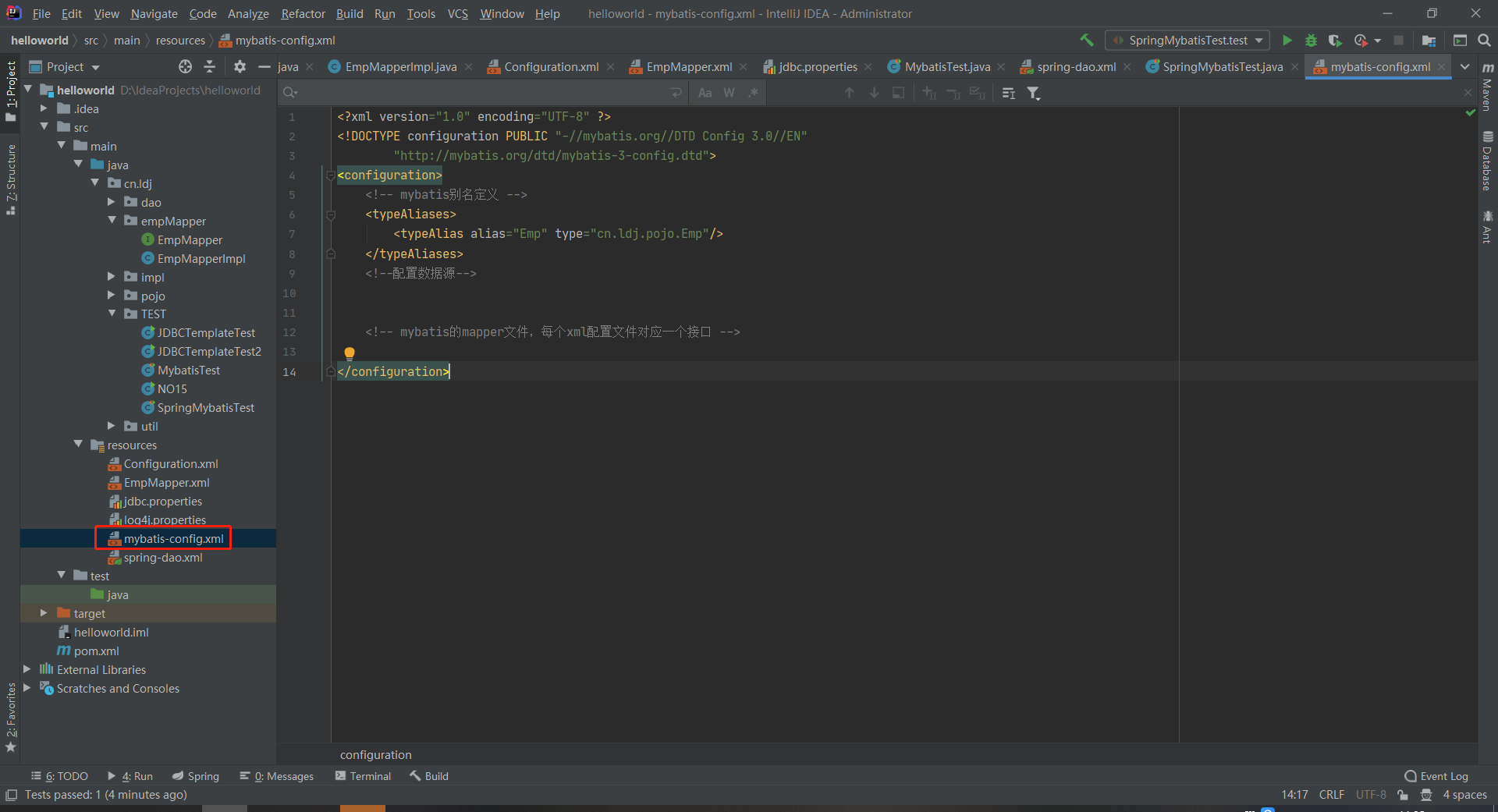
1. spring配置

在resource目录下新建spring-dao.xml



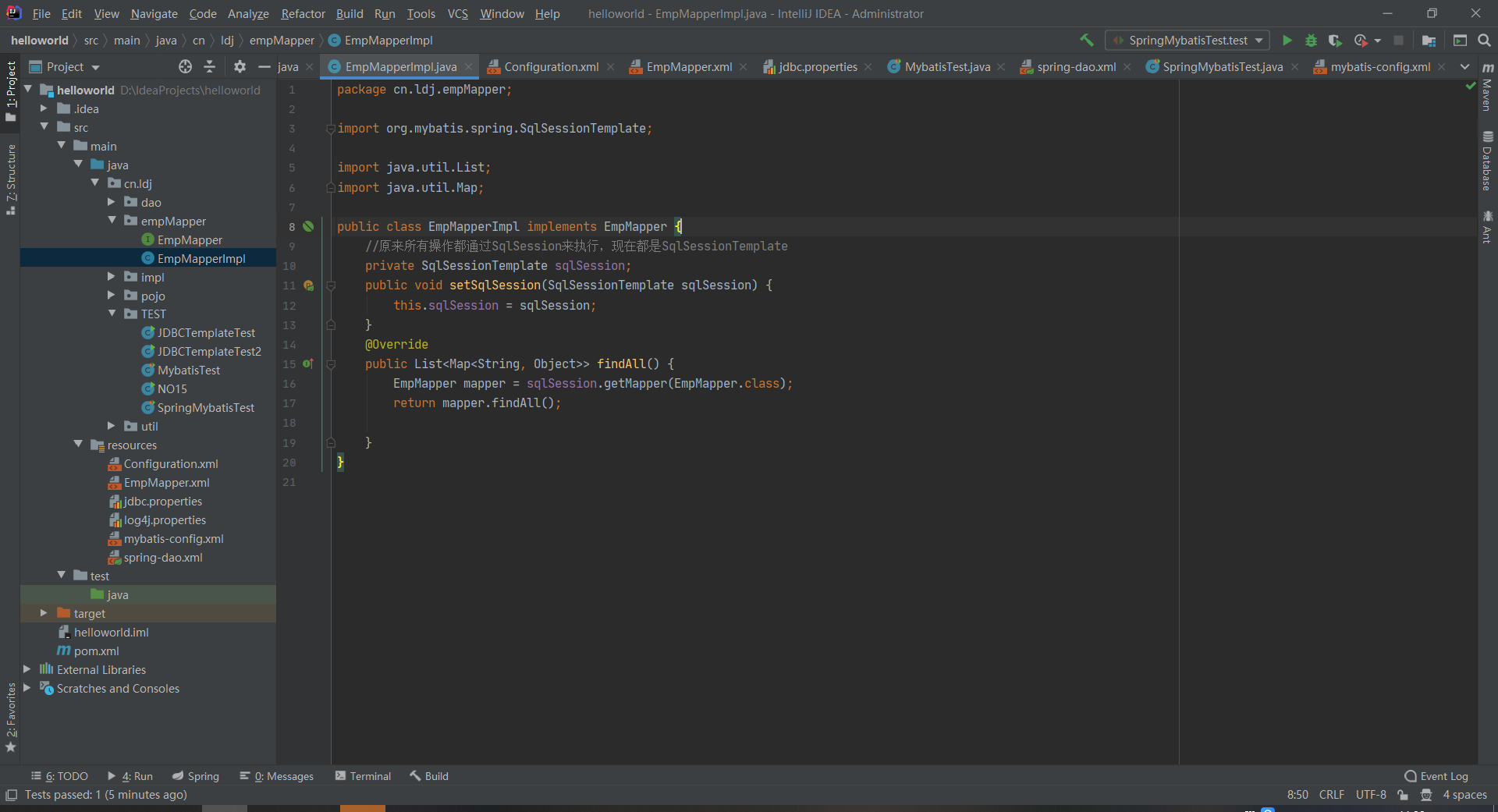
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
 <!—spring内置连接池DataSource:使用Spring的数据源替换MyBatis的配置-->  
 <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <property name="location">  
 <value>classpath:jdbc.properties</value>  
 </property>  
 </bean>  
 <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property name="driverClassName" value="${jdbc.driver}"/>  
 <property name="url" value="${jdbc.url}" />  
 <property name="username" value="${jdbc.username}"/>  
 <property name="password" value="${jdbc.password}"/>  
 </bean>  
  
 <!--sqlSessionFactory-->  
 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <property name="dataSource" ref="dataSource"/>  
 <!--绑定mybatis配置文件-->  
 <property name="configLocation" value="classpath:mybatis-config.xml"/>  
 <!--注册Mapper.xm映射器-->  
 <property name="mapperLocations" value="classpath:EmpMapper.xml"/>  
 </bean>  
  
 <!--SqlSessionTemplate:就是我们使用的sqlSession-->  
 <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">  
 <!--只能使用构造器进行注入，因为没有set方法-->  
 <constructor-arg index="0" ref="sqlSessionFactory"/>  
 </bean>  
  
 <!--利用spring创建EmpMapperImpl接口实现类对象，取名为empMapper，并注入上述创建好的sqlSession对象-->  
 <bean id="empMapper" class="cn.ldj.empMapper.EmpMapperImpl">  
 <property name="sqlSession" ref="sqlSession"/>  
 </bean>  
</beans>

2.Mybatis配置



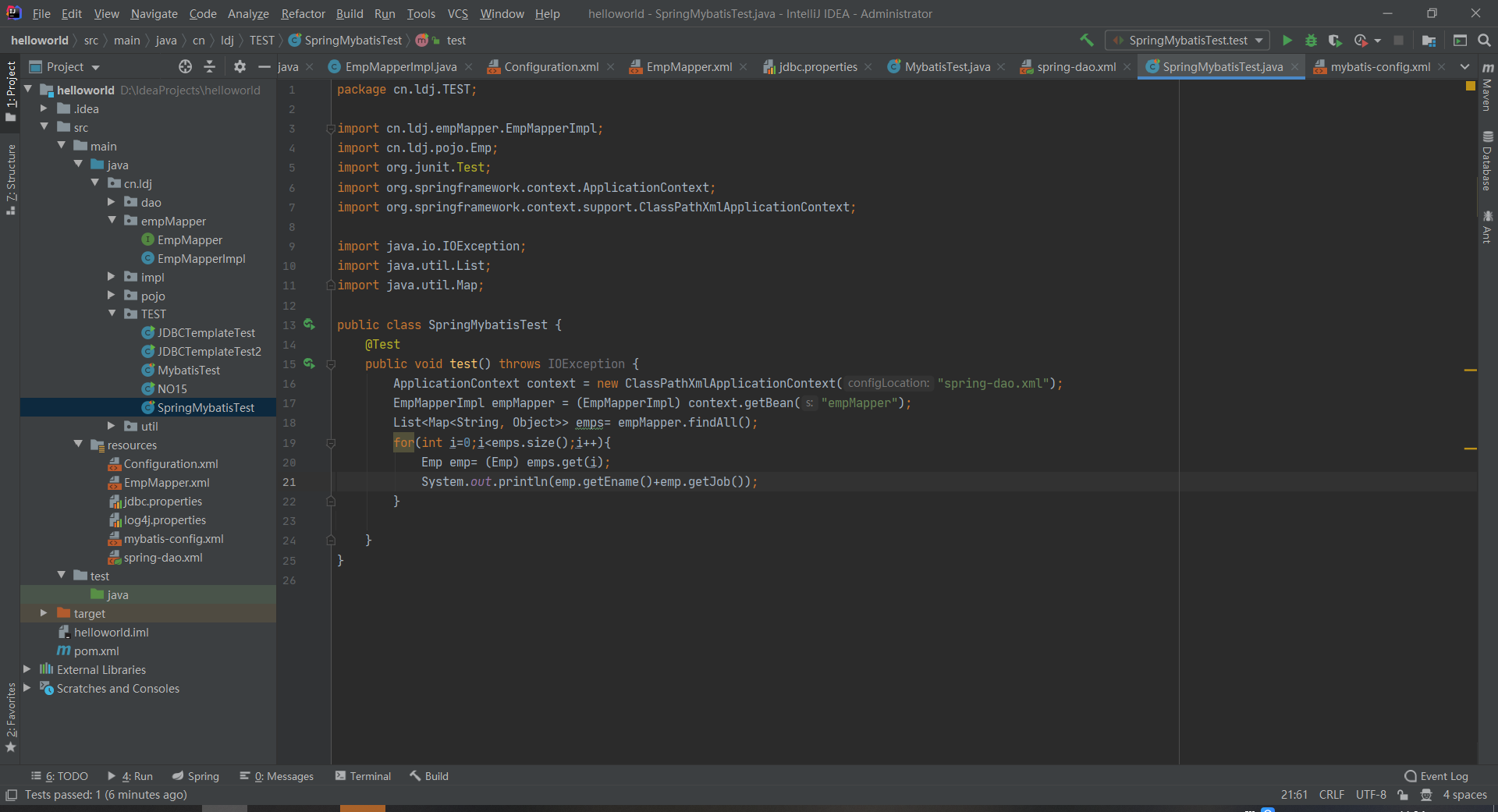
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <!-- mybatis别名定义 -->  
 <typeAliases>  
 <typeAlias alias="Emp" type="cn.ldj.pojo.Emp"/>  
 </typeAliases>  
 <!--配置数据源-->  
  
  
 <!-- mybatis的mapper文件，每个xml配置文件对应一个接口 -->  
  
</configuration>

4.impl

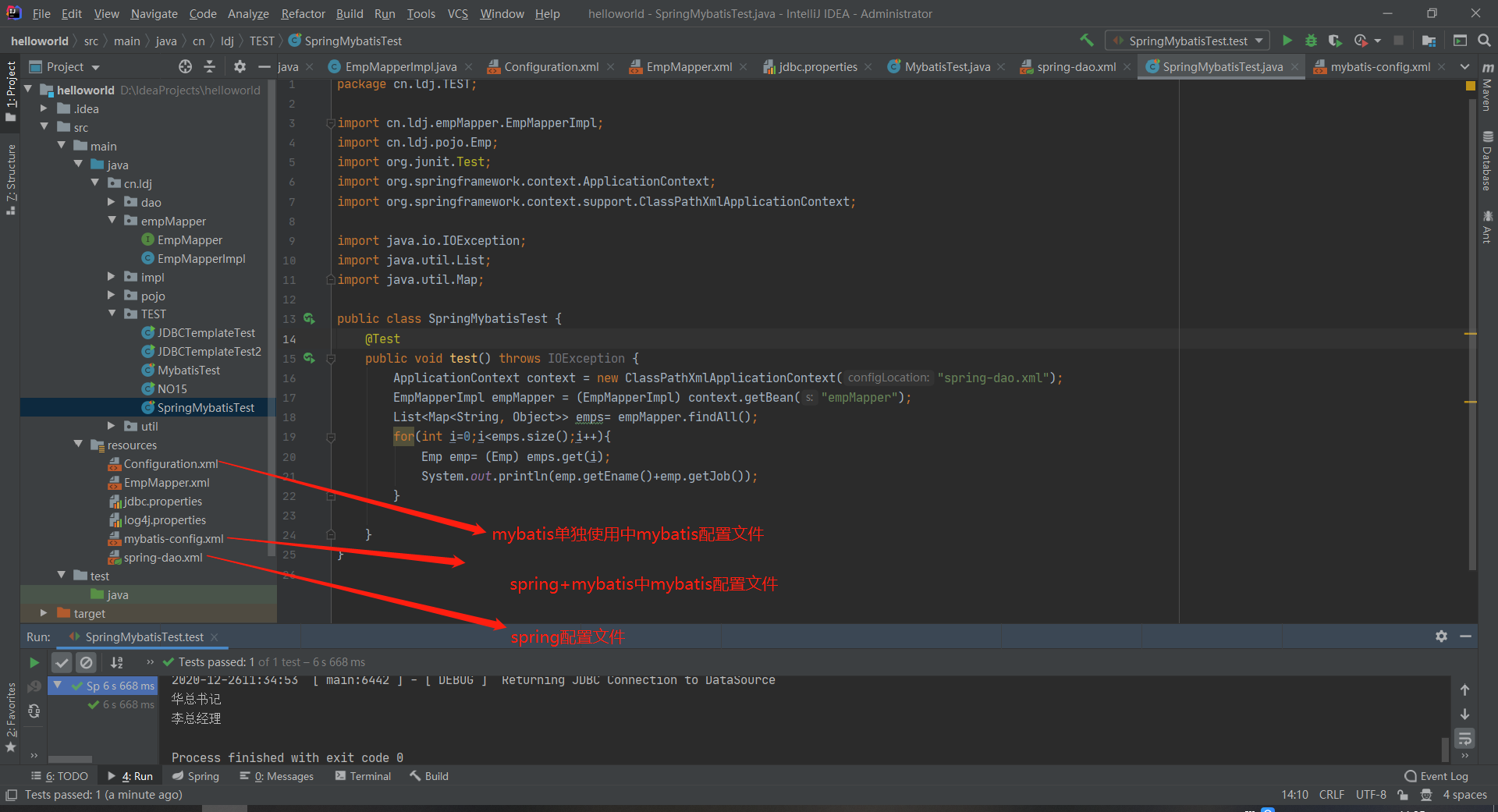


package cn.ldj.empMapper;  
  
import org.mybatis.spring.SqlSessionTemplate;  
  
import java.util.List;  
import java.util.Map;  
  
public class EmpMapperImpl implements EmpMapper {  
 //原来所有操作都通过SqlSession来执行，现在都是SqlSessionTemplate  
 private SqlSessionTemplate sqlSession;  
 public void setSqlSession(SqlSessionTemplate sqlSession) {  
 this.sqlSession = sqlSession;  
 }  
 @Override  
 public List<Map<String, Object>> findAll() {  
 EmpMapper mapper = sqlSession.getMapper(EmpMapper.class);  
 return mapper.findAll();  
  
 }  
}

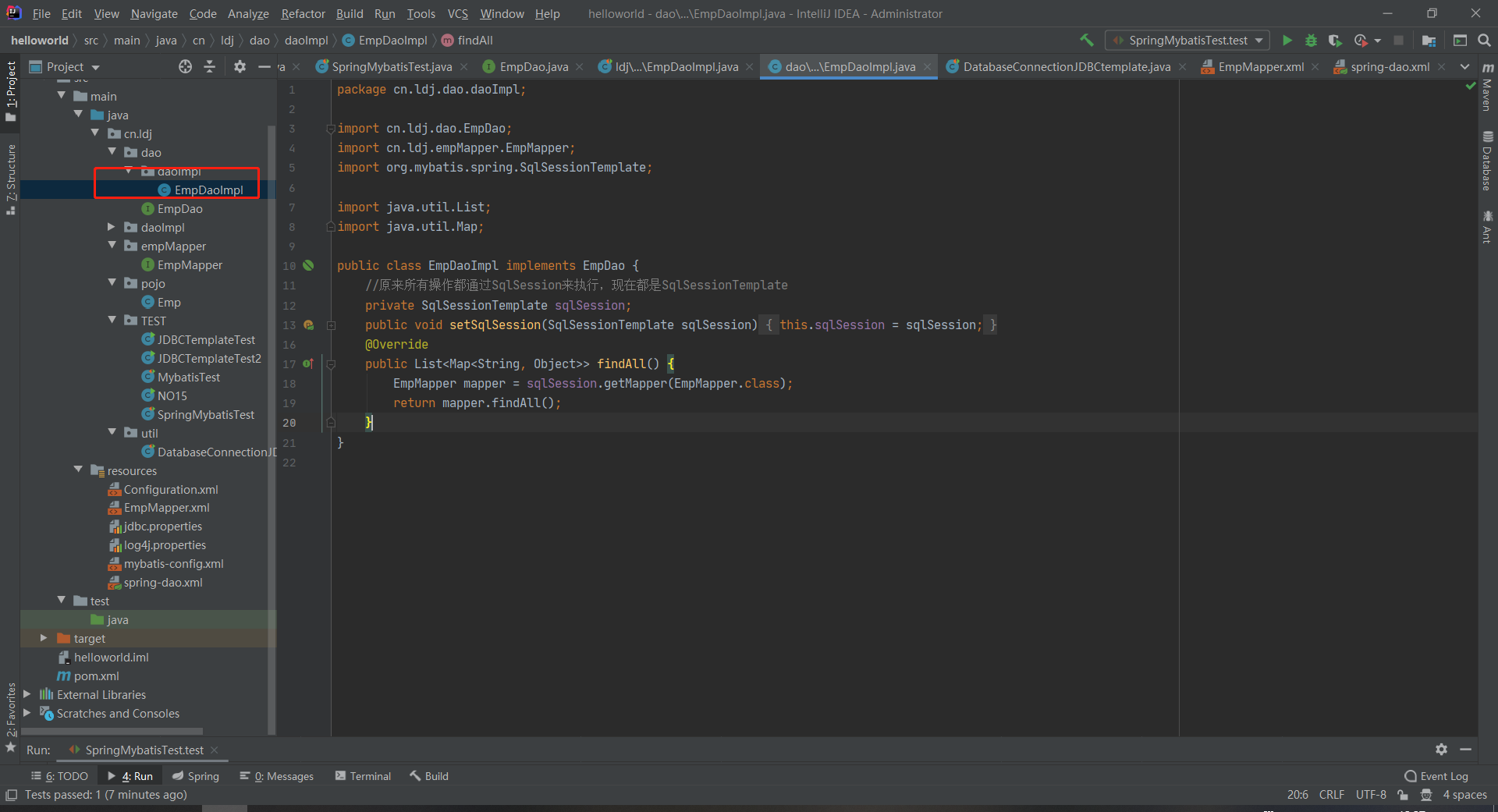
5.完成测试



package cn.ldj.TEST;  
  
import cn.ldj.empMapper.EmpMapperImpl;  
import cn.ldj.pojo.Emp;  
import org.junit.Test;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
import java.io.IOException;  
import java.util.List;  
import java.util.Map;  
  
public class SpringMybatisTest {  
 @Test  
 public void test() throws IOException {  
 ApplicationContext context = new ClassPathXmlApplicationContext("spring-dao.xml");  
 EmpMapperImpl empMapper = (EmpMapperImpl) context.getBean("empMapper");  
 List<Map<String, Object>> emps= empMapper.findAll();  
 for(int i=0;i<emps.size();i++){  
 Emp emp= (Emp) emps.get(i);  
 System.*out*.println(emp.getEname()+emp.getJob());  
 }  
  
 }  
}

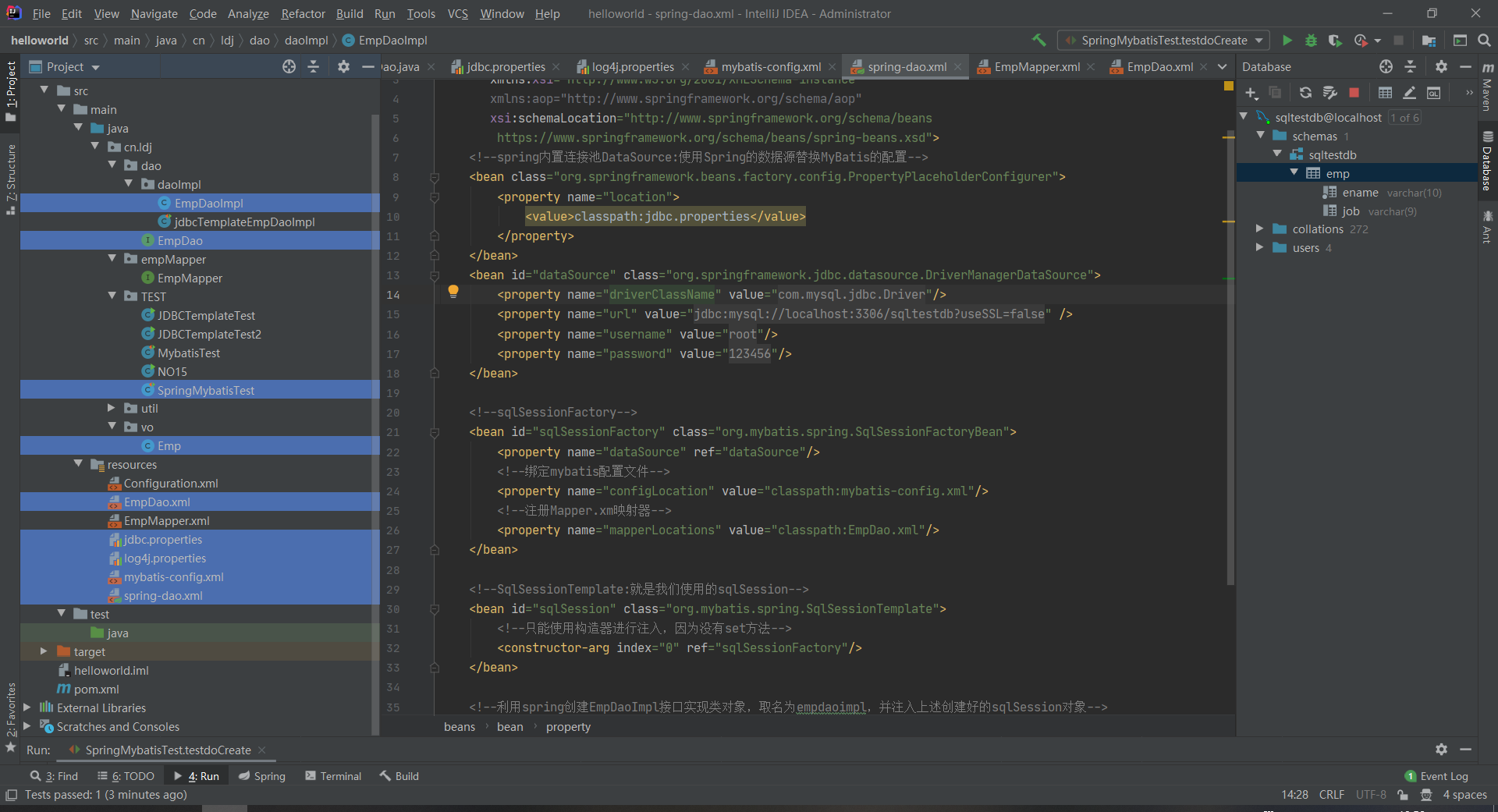


6.将imp修改为实现EmpDao并移动至dao-daoImpl包下，并重命名为EmpDaoimpl。重命名和移动时配置文件中引用该文件的地方会自动变化。将pojo包更名为vo。



25．修正项目Spring+Mybatis+MYSQL

目前项目只有查询所有功能，针对任务要求，现添加“新增”功能，标色部分即为修改后的代码。代码依次如下



package cn.ldj.dao.daoImpl;  
  
import cn.ldj.dao.EmpDao;  
import cn.ldj.empMapper.EmpMapper;  
import cn.ldj.vo.Emp;  
import org.mybatis.spring.SqlSessionTemplate;  
  
import java.util.List;  
import java.util.Map;  
  
public class EmpDaoImpl implements EmpDao {  
 //原来所有操作都通过SqlSession来执行，现在都是SqlSessionTemplate  
 private SqlSessionTemplate sqlSession;  
 public void setSqlSession(SqlSessionTemplate sqlSession) {  
 this.sqlSession = sqlSession;  
 }  
 @Override  
 public List<Map<String, Object>> findAll() throws Exception {  
 EmpDao mapper = sqlSession.getMapper(EmpDao.class);  
 return mapper.findAll();  
 }  
  
 @Override  
 public void doCreate(Emp emp) throws Exception {  
 EmpDao mapper = sqlSession.getMapper(EmpDao.class);  
 mapper.doCreate(emp);  
 }  
}

package cn.ldj.dao;  
  
import cn.ldj.vo.Emp;  
  
import java.util.List;  
import java.util.Map;  
  
public interface EmpDao {  
  
 */\*\*  
 \* 查询  
 \*/* public List<Map<String, Object>> findAll() throws Exception ;  
  
 */\*\*  
 \* 实现数据增加操作  
 \*/* public void doCreate(Emp emp) throws Exception;  
}

package cn.ldj.TEST;  
  
import cn.ldj.dao.daoImpl.EmpDaoImpl;  
import cn.ldj.vo.Emp;  
import org.junit.Test;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
import java.io.IOException;  
import java.util.List;  
import java.util.Map;  
  
public class SpringMybatisTest {  
 @Test  
 public void test() throws Exception {  
 ApplicationContext context = new ClassPathXmlApplicationContext("spring-dao.xml");  
 EmpDaoImpl empdaoimpl = (EmpDaoImpl) context.getBean("empDaoimpl");  
 List<Map<String, Object>> emps= empdaoimpl.findAll();  
 for(int i=0;i<emps.size();i++){  
 Emp emp= (Emp) emps.get(i);  
 System.*out*.println(emp.getEname()+emp.getJob());  
 }  
 }  
 @Test  
 public void testdoCreate() throws Exception {  
 Emp emp=new Emp();  
 emp.setEname("笑笑xiao3");  
 emp.setJob("副总经理");  
 ApplicationContext context = new ClassPathXmlApplicationContext("spring-dao.xml");  
 EmpDaoImpl empdaoimpl = (EmpDaoImpl) context.getBean("empDaoimpl");  
 try {  
 empdaoimpl.doCreate(emp);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 List<Map<String, Object>> emps= empdaoimpl.findAll();  
 for(int i=0;i<emps.size();i++){  
 Emp emp2= (Emp) emps.get(i);  
 System.*out*.println(emp2.getEname()+emp2.getJob());  
 }  
  
 }  
}

package cn.ldj.vo;  
  
public class Emp {  
  
 private String ename;  
 private String job;  
  
  
 public String getEname() {  
 return ename;  
 }  
  
 public void setEname(String ename) {  
 this.ename = ename;  
 }  
  
  
 public String getJob() {  
 return job;  
 }  
  
 public void setJob(String job) {  
 this.job = job;  
 }  
  
}

<?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">  
  
<mapper namespace="cn.ldj.dao.EmpDao">  
 <select id="findAll" resultType="Emp">  
select \* from emp  
</select>  
 <insert id="doCreate" useGeneratedKeys="true" parameterType="Emp">  
 insert into emp(ename,job) values(#{ename},#{job})  
</insert>  
</mapper>

jdbc.driver=com.mysql.jdbc.Driver  
jdbc.url=jdbc:mysql://localhost:3306/sqltestdb?useSSL=false  
jdbc.username=root  
jdbc.password=123456

### 设置###  
log4j.rootLogger = debug,stdout  
  
### 输出信息到控制抬 ###  
log4j.appender.stdout = org.apache.log4j.ConsoleAppender  
log4j.appender.stdout.Target = System.out  
  
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout  
  
log4j.appender.stdout.layout.ConversionPattern = %-d{yyyy-MM-ddHH:mm:ss} [ %t:%r ] - [ %p ] %m%n

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <!-- mybatis别名定义 -->  
 <typeAliases>  
 <typeAlias alias="Emp" type="cn.ldj.vo.Emp"/>  
 </typeAliases>  
 <!--配置数据源-->  
  
  
 <!-- mybatis的mapper文件，每个xml配置文件对应一个接口 -->  
  
</configuration>

<?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:aop="http://www.springframework.org/schema/aop"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
 <!--spring内置连接池DataSource:使用Spring的数据源替换MyBatis的配置-->  
 <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <property name="location">  
 <value>classpath:jdbc.properties</value>  
 </property>  
 </bean>  
 <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property name="driverClassName" value="${jdbc.driver}"/>  
 <property name="url" value="${jdbc.url}" />  
 <property name="username" value="${jdbc.username}"/>  
 <property name="password" value="${jdbc.password}"/>  
 </bean>  
  
 <!--sqlSessionFactory-->  
 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <property name="dataSource" ref="dataSource"/>  
 <!--绑定mybatis配置文件-->  
 <property name="configLocation" value="classpath:mybatis-config.xml"/>  
 <!--注册Mapper.xm映射器-->  
 <property name="mapperLocations" value="classpath:EmpDao.xml"/>  
 </bean>  
  
 <!--SqlSessionTemplate:就是我们使用的sqlSession-->  
 <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">  
 <!--只能使用构造器进行注入，因为没有set方法-->  
 <constructor-arg index="0" ref="sqlSessionFactory"/>  
 </bean>  
  
 <!--利用spring创建EmpDaoImpl接口实现类对象，取名为empdaoimpl，并注入上述创建好的sqlSession对象-->  
 <bean id="empDaoimpl" class="cn.ldj.dao.daoImpl.EmpDaoImpl">  
 <property name="sqlSession" ref="sqlSession"/>  
 </bean>  
</beans>