IDEA手动搭建Spring mvc+Mybatis开发环境（推荐）

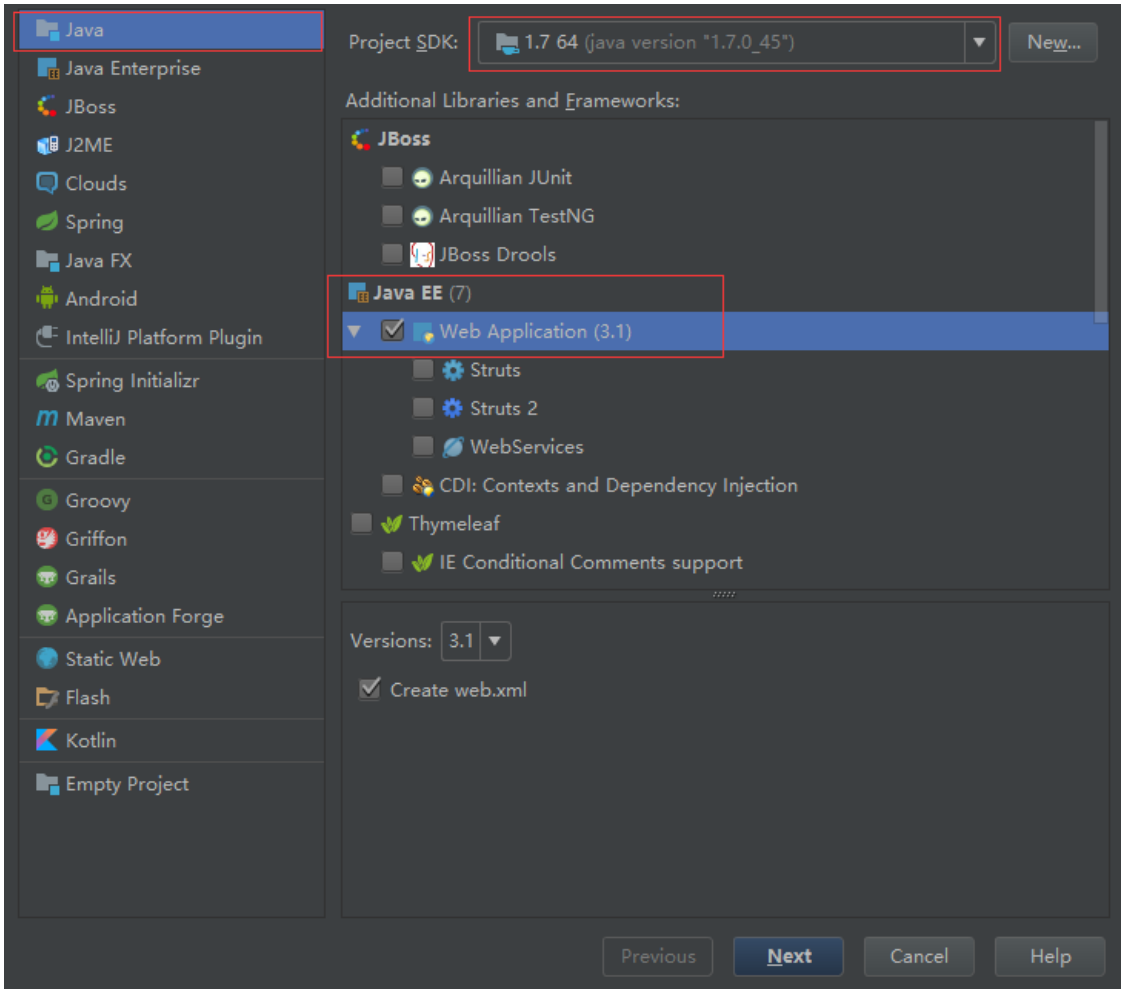
前言：

虽然IDEA创建项目时有自动创建Spring mvc的项目选项，但是这里推荐手动搭建，一是手动搭建可以加深对Spring mvc框架的理解，二是少踩一些不必要的坑。

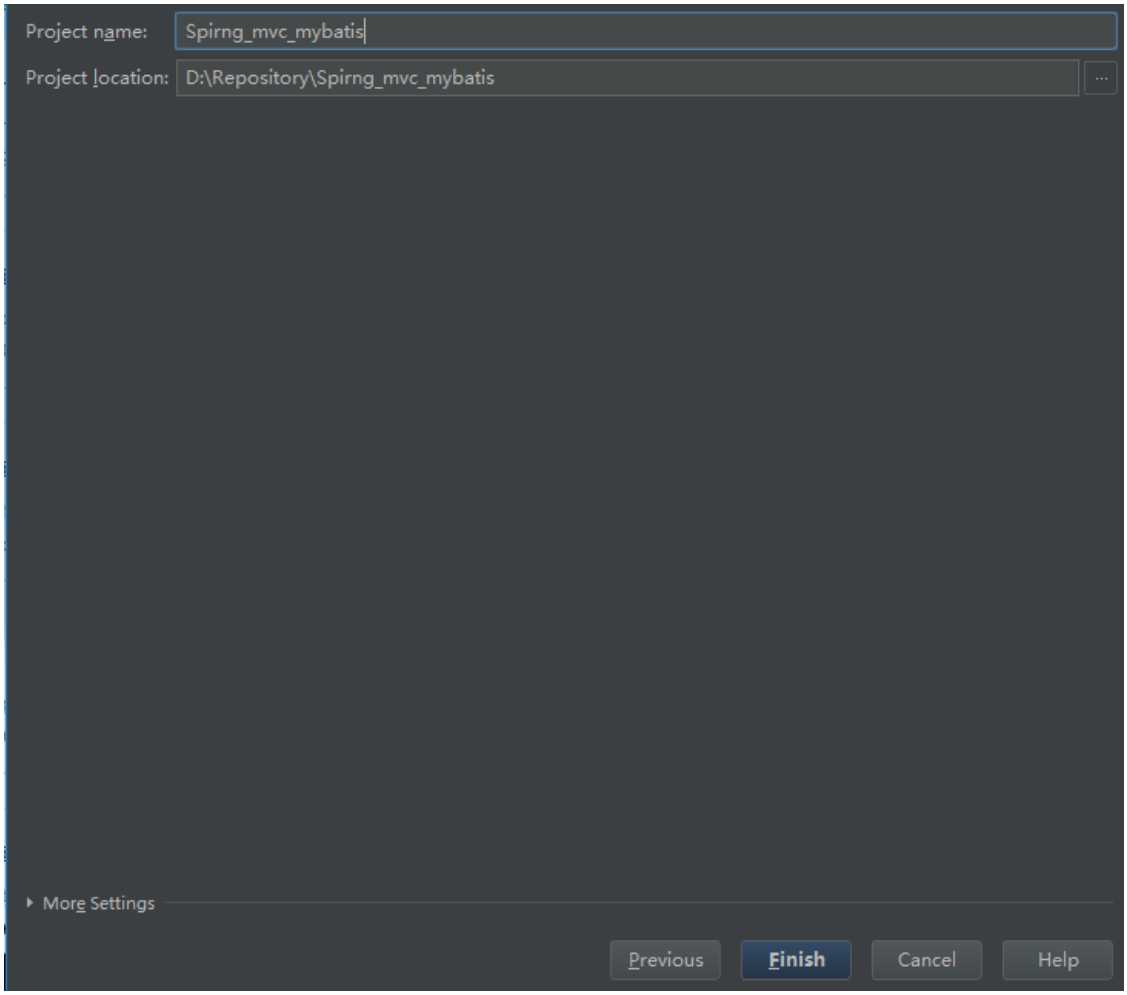
一、创建Java web项目

1. 打开IDEA选择 Create New Project 创建新的项目

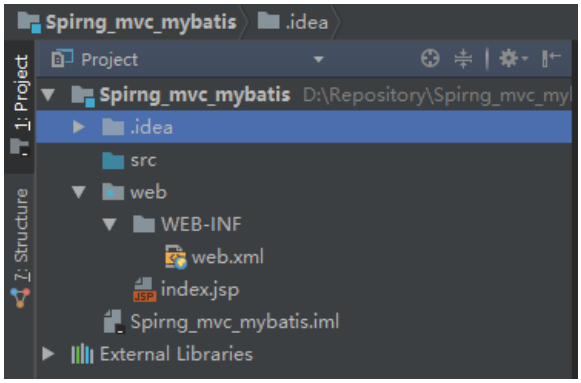
2. 选择创建Java项目，然后选择sdk版本，勾选创建Web Application项目



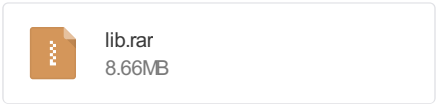
3. 选择项目保存的路径，然后给项目命名



4. 点击Finish开始创建，创建成功后按快捷键Alt+1来查看目录



二、导入Sping+mybatis +log4j的依赖jar包



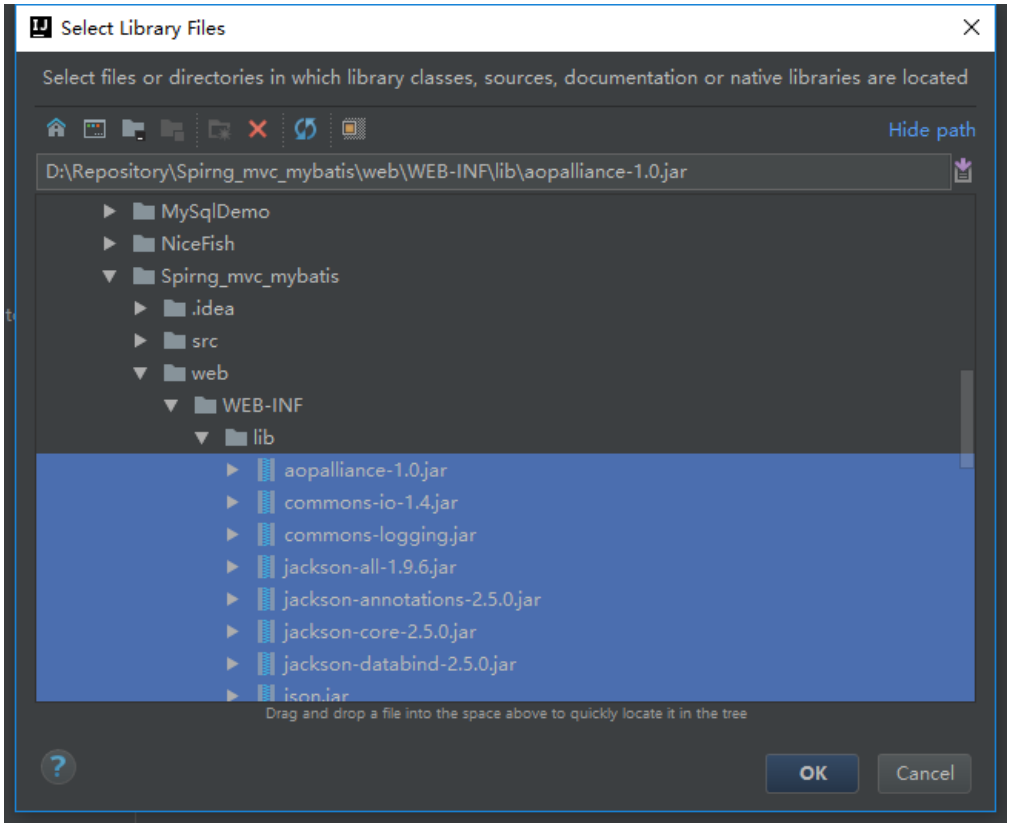
这里spring版本是3.2

mybatis版本是3.2.6

1. 在web/WEB-INF/下创建lib目录，并将所需jar包复制进去。
2. 在项目中导入这些jar包

   a.右击项目名Spirng_mvc_mybatis 选择 **Open Model Settings** 或者快捷键 **F4**打开项目设置

   b. 选中Libraries 然后点击绿色的+号，选中lib目录下的所有jar包，然后点ok导入



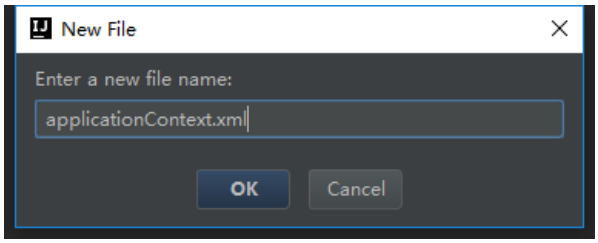
三、配置Spring框架

在web/WEB-INF/下创建spring_mybatis目录用来存放spring和mybatis的配置信息

配置applicationContext.xml

1. 在spring_mybatis目录下创建applicationContext.xml文件

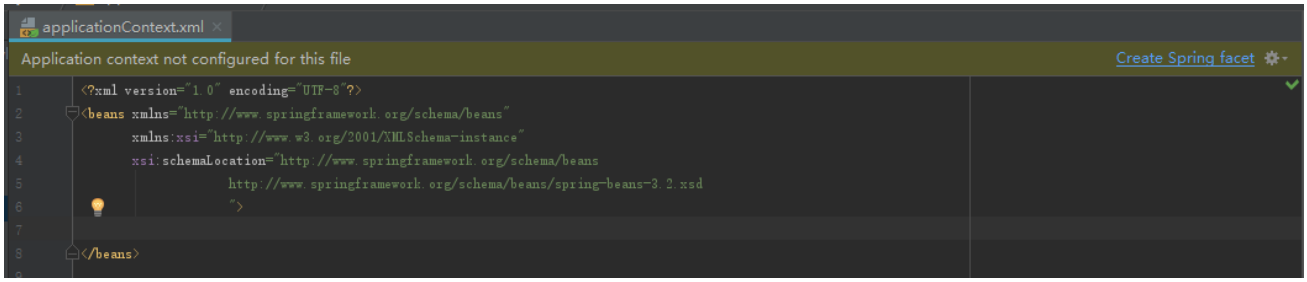
右击目录然后new一个file，输入applicationContext.xml即可创建。



2. 在applicationContext.xml中添加配置信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
            ">
</beans>
```

3. 这时IDEA会提醒你Application context notconfigured for this file 选择右边的 Create Spring facet先创建facet



然后 Configure application context（这里默认选项就可以）

配置DispatchServlet

1. 在web/WEB-INF/spring_mybatis/下创建spring-servlet.xml文件，用于开启基于注解的Springmvc功能，创建方法跟上面一样

2. 配置信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
                    http://www.springframework.org/schema/context
                    http://www.springframework.org/schema/context/spring-context-3.2.xsd
                    http://www.springframework.org/schema/mvc
                    http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
                    ">
    <!-- 启动注解驱动的Spring MVC功能，注册请求url和注解POJO类方法的映射 -->
    <mvc:annotation-driven />
    <!-- 启动包扫描功能，以便注册带有@Controller、@Service、@repository、@Component等注解的类成为spring的bean -->
    <context:component-scan base-package="com"></context:component-scan>

    <!-- 对模型视图名称的解析，在请求时模型视图名称添加前后缀 -->
    <bean
            class="org.springframework.web.servlet.view.InternalResourceViewResolver"
            p:prefix="/jsp/"
            p:suffix=".jsp" >
    </bean>
</beans>
```

3. 同理需要 Configure application context

配置web.xml

修改web/WEB-INF/下的web.xml文件，配置信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
    <display-name>mvc_demo</display-name>
    <!-- 配置需要装载的Spring配置文件 -->
    <!-- 默认的路径是"/WEB-INF/applicationContext.xml，多个路径用“，”号隔开 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring_mybatis/applicationContext.xml
        </param-value>
    </context-param>
```

```
    <!-- Spring监听配置，Web容器启动自动装配ApplicationContext的配置信息 -->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- Spring字符编码过滤器配置，处理中文乱码，针对post请求 -->
    <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>

    <!-- Spring Servlet配置 -->
    <servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <!-- 默认的路径是"/WEB-INF/spring-servlet.xml，多个路径用“，”号隔开 -->
            <param-value>/WEB-INF/spring_mybatis/spring-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/service/*</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>jsp/index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

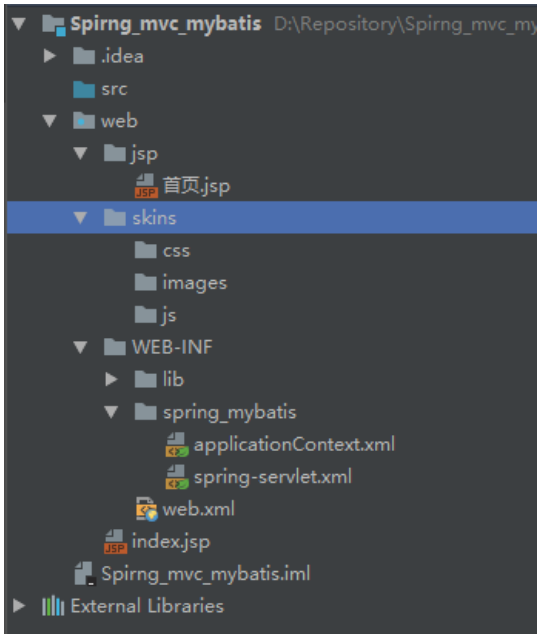
四、创建web目录结构

1.在web/下创建jsp目录用来存放jsp文件

2.创建skins目录

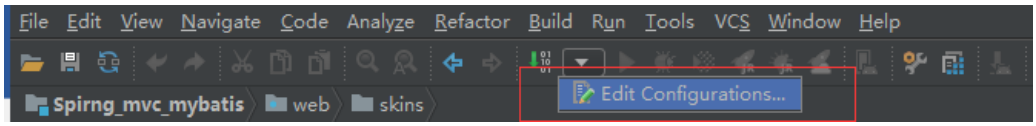
在skins目录下分别创建js、css、images目录

目录结构如下图所示：

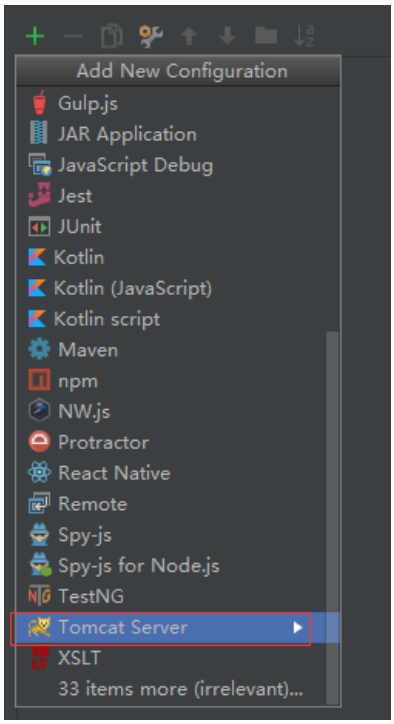


五、部署tomcat

1.在工具栏点一下向下的箭头，然后选择Edit Configurations

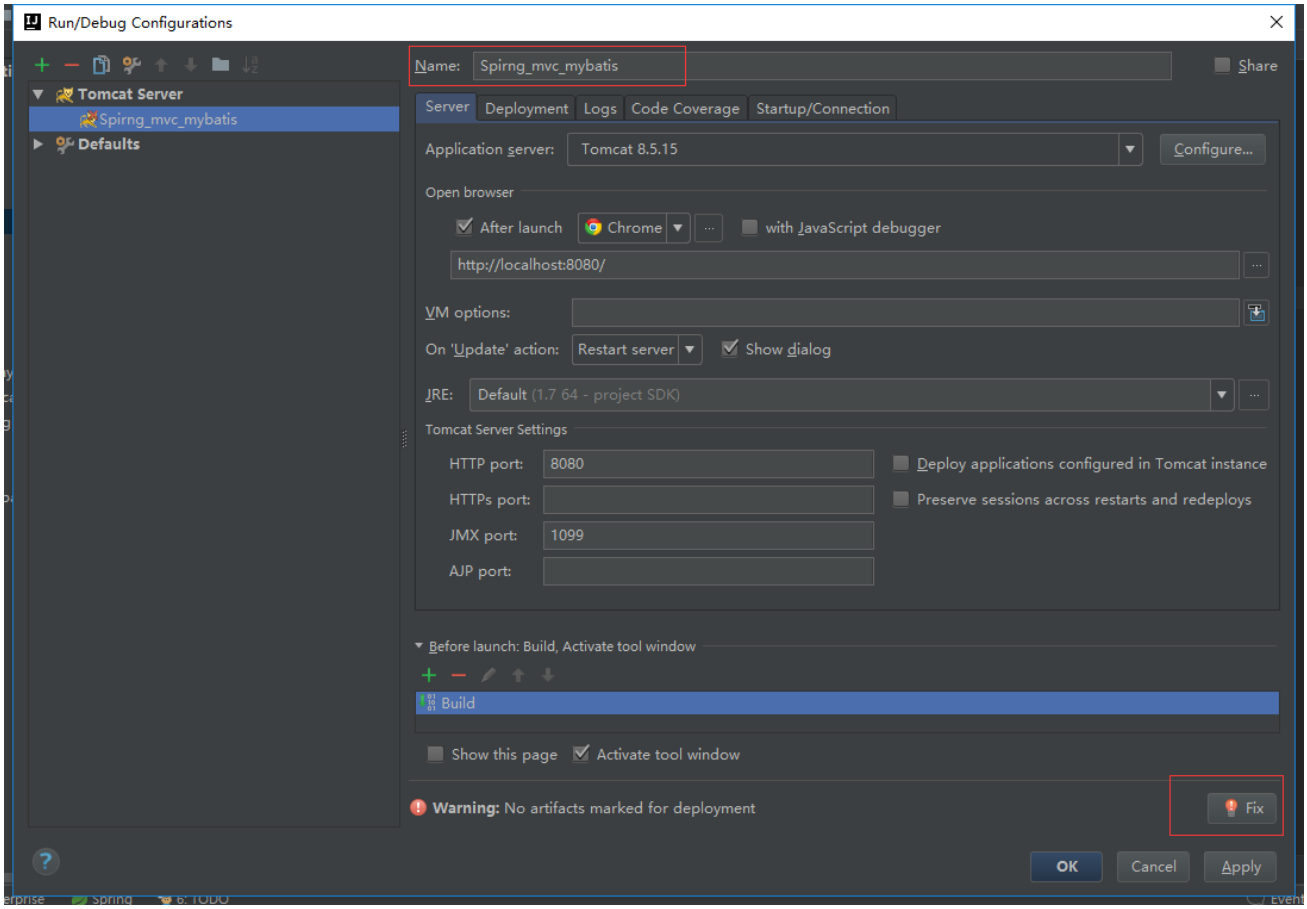


2. 点击绿色的加号然后选择tomcat，选择local



3.配置信息

先给tomcat起个名字，然后点击右下角的fix来配置Deployment里的信息，如下所示



六、测试spring环境

1. 在jsp目录下创建一个index.jsp文件

代码如下：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>首页</title>
</head>
<body>
<h1>Spring环境搭建成功</h1>
</body>
</html>
```

2. 然后点击运行tomcat

七、配置Mybatis

1.添加datasource.xml数据源文件，用来配置数据源信息

a. 在spring-mybatis目录下创建datasource.xml文件

b. 配置信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
">
    <!-- 配置数据源 -->
```

```
    <bean id="dataSource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName">
            <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url">
            <value>jdbc:mysql://localhost:3306/databaseName?characterEncoding=UTF-8
            </value>
        </property>
        <property name="username">
            <value>username</value>
        </property>
        <property name="password">
            <value>password</value>
        </property>
    </bean>
</beans>
```

代码说明：

driverClassName:

数据库驱动如果是mysql的话value值为：com.mysql.jdbc.Driver

如果是oracle的话value的值为：oracle.jdbc.driver.OracleDriver（当然需要引入相应的jar包）

url:

连接数据库服务器的地址以及数据库的名字databaseName是填写需要连接数据库的名字

username:

value填用户名

password:

value填密码

注意：黄色标注的这几个value值需要根据自己数据库的情况正确填写。

2.添加mybatis-config.xml文件，用来配置mybatis

a. 在spring-mybatis目录下创建mybatis-config.xml文件

b. 配置信息如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "classpath:/mybatis-3-config.dtd">

<configuration>
    <settings>
        <setting name="jdbcTypeForNull" value="NULL" />
        <setting name="callSettersOnNulls" value="true" />
        <setting name="logImpl" value="LOG4J" />
    </settings>
</configuration>
```

3.添加mybatis-context.xml 上下文配置文件，用来配置和spring框架的依赖

a. 在spring-mybatis目录下创建mybatis-context.xml文件

b. 配置信息如下：

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

```
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.2.xsd"
>

    <bean id="mybatisTransactionManager"
            class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource" />
    </bean>

    <tx:annotation-driven transaction-manager="mybatisTransactionManager" />

    <!-- define the SqlSessionFactory -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="configLocation" value="/WEB-INF/spring_mybatis/mybatis-config.xml"/>
    </bean>
    <!-- 定义扫描路径 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.inspur.**.dao" />
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
    </bean>
</beans>
```

4.修改web.xml文件增加以下代码：

```
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring_mybatis/datasource.xml
            /WEB-INF/spring_mybatis/mybatis-context.xml
            /WEB-INF/spring_mybatis/applicationContext.xml
        </param-value>
    </context-param>
```

八、编写查询数据库小例子测试mybatis+Spring mvc

需求：在前端jsp页面显示查询的某个数据

1.前端jsp

因为我们只是测试一下能否连接数据库并获取数据，所以前端只是写几行代码能显示后台传来的数据即可。

在jsp目录下新建一个名字为testState的jsp文件，内部代码为：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>testMybatis</title>
</head>
<body>
<h1>Mybatis测试状态：${requestScope.message}</h1>
</body>
</html>
```

2.Bean层 data

用来封装查询数据库状态信息的属性和方法

a. 在src目录下创建包com.springmvc.demo.data

在src目录上右击选择new—>package

b. 在data包下创建Sata类

内容如下：

```
package com.springmvc.demo.data;

import javax.persistence.Column;
import javax.persistence.Id;
import javax.persistence.Table;

@Table(name = "myBatisState")
public class State {
    @Id
    private Integer id;
    @Column(name = "testState")
    private String testState;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getTestState() {
        return testState;
    }

    public void setTestState(String testState) {
        this.testState = testState;
    }
}
```

3. 持久层 dao

持久层用来与数据库交互，服务层可以调用持久层的相应方法，来进行业务逻辑的处理

a. 在src目录下创建包com.springmvc.demo.dao

方法跟如上

b. 在dao包下创建StateMapper接口

内容如下：

```
package com.springmvc.demo.dao;
import com.springmvc.demo.data.State;

public interface StateMapper {
    public State getInfo();
}
```

c. 创建dao层后，需要在mybatis-context.xml配置文件中添加dao层的扫描

修改路径：web/WEB-INF/spirng_mybatis/mybatis-context.xml

添加内容如下黄色背景：

```
<?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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx-3.2.xsd"
>

    <bean id="mybatisTransactionManager"
          class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource" />
    </bean>

    <tx:annotation-driven transaction-manager="mybatisTransactionManager" />

    <!-- define the SqlSessionFactory -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="configLocation" value="/WEB-INF/spring_mybatis/mybatis-config.xml"/>
    </bean>
    <!-- 定义扫描路径 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.springmvc.*.dao" />
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
    </bean>
</beans>
```

d. 在dao包下创建StateMapper.xml文件

在StateMapper接口中定义了getInfo方法，当调用这个方法的时候，会从StateMapper.xml中查询出所需数据。所以我们需要创建一个StateMapper.xml文件，并在里编写sal语句对数据库操作

内容如下：

```
<?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">

<mapper namespace="com.springmvc.demo.dao.StateMapper">

    <resultMap type="com.springmvc.demo.data.State" id="stateResultMap">
        <result property="testState" column="testState" />
    </resultMap>

    <select id="getInfo" resultMap="stateResultMap" >
        SELECT * FROM myBatisState
    </select>

</mapper>
```

4. 服务层Service

a. 在src目录下创建包com.springmvc.demo.service

b. 创建service层后，需要在applicationContext.xml配置文件中添加dao层的扫描

修改路径：web/WEB-INF/spirng_mybatis/applicationContext.xml

修改内容如下黄色背景内容：

```
<?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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
          http://www.springframework.org/schema/context
          http://www.springframework.org/schema/context/spring-context.xsd">
    <context:component-scan base-package="com.springmvc.*.service"></context:component-scan>
</beans>
```

c.在service包下创建IState接口

代码如下：

```
package com.springmvc.demo.service;
import com.springmvc.demo.data.State;

public interface IState {
    /*因为就一条信息所以直接查询*/
    public State getInfo();
}
```

c.在service包下创建impl包然后创建IState接口的实现类 StateServiceImpl

代码如下：

```
package com.springmvc.demo.service.impl;

import com.springmvc.demo.dao.StateMapper;
import com.springmvc.demo.data.State;
import com.springmvc.demo.service.IState;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service("stateService")
public class StateServiceImpl implements IState {
    @Autowired
    private StateMapper stateMapper;
    public State getInfo() {
        return stateMapper.getInfo();
    }
}
```

5. 控制层controller

前端和后台数据连接的桥梁，用来获取前端传来的数据，并调用服务层来处理相应的业务逻辑

a. 在src目录下创建包com.springmvc.demo.controller

b.在controller包下创建testController类

内容如下：

```
package com.springmvc.demo.controller;
import com.springmvc.demo.data.State;
import com.springmvc.demo.service.IState;
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;
//声明为控制层
@Controller
@RequestMapping("/")
public class TestController {
    @Autowired
    private IState iState;
    @RequestMapping("/testMybatis")
    public ModelAndView testState(){
        State state = iState.getInfo();
        String restState=state.getTestState();
        return new ModelAndView("/testMybatis","message",restState);
    }
}
```

c. 创建controller类后，需要在spring-servlet.xml配置文件中添加controller的注册扫描

修改路径：web/WEB-INF/spirng_mybatis/sprign-servlet.xml

修改内容如下黄色背景内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
                    http://www.springframework.org/schema/context
                    http://www.springframework.org/schema/context/spring-context-3.2.xsd
                    http://www.springframework.org/schema/mvc
                    http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
                    ">
    <!-- 启动注解驱动的Spring MVC功能，注册请求url和注解POJO类方法的映射 -->
    <mvc:annotation-driven />
    <!-- 启动包扫描功能，以便注册带有@Controller、@Service、@repository、@Component等注解的类成为spring
的bean -->
    <context:component-scan base-package="com.springmvc.*.controller"></context:component-scan>

    <!-- 对模型视图名称的解析，在请求时模型视图名称添加前后缀 -->
    <bean
            class="org.springframework.web.servlet.view.InternalResourceViewResolver"
            p:prefix="/jsp/"
            p:suffix=".jsp" >
    </bean>
</beans>
```

6. 初始化Sql创建表

在要查询的数据库中创建表mybatisstate

```
CREATE TABLE mybatisstate(teststate VARCHAR(20));
INSERT INTO `mybatisstate` (`testState`) VALUES ('ok')
```

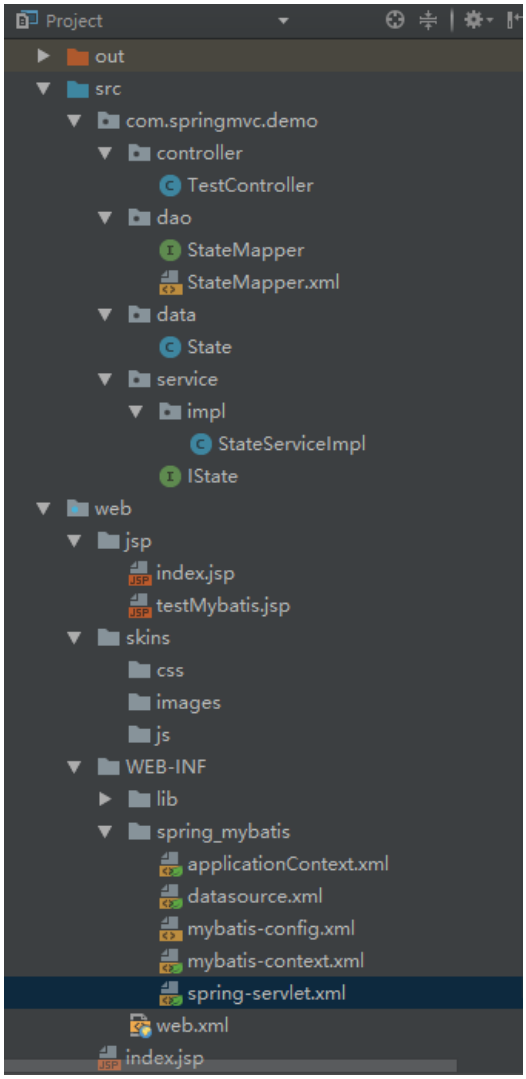
7. 测试效果

重启tomcat，输入网址http://localhost:8080/service/testMybatis

Mybatis测试状态：ok

出现这个页面就说明我们的环境搭建成功了！！

我们整个项目的目录结构就是这个样子



小技巧：

环境搭建好之后，我们不要急着去做开发，我们需要把刚搭建好的项目备份一下，以后每次需要用到springmvc+mybatis框架时，复制一份，然后改改名字直接用就ok了。

IDEA版本信息：

IntelliJ IDEA 2017.2.1

Build #IU-172.3544.35, built on July 31, 2017

Licensed to shirukai

JRE: 1.8.0_152-release-915-b6 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Windows 10 10.0

另外注意：此项目jdk环境为1.7