**一、Maven下SSM的环境的搭建**

1、创建maven的war项目（见maven笔记第四点和第七点），创建完后maven项目缺少文件夹可以像笔记中那样手动补全，也可以使用eclipse集成的工具：右键项目，选择javaEE-tools,选择generate…那一项。

2、在pom.xml文件中添加项目依赖包，一般包括以下几点：

1）三个java EE的必要基础包

2）spring相关包

3）mybatis相关包

4）spring和mybatis整合包

5）mysql驱动包

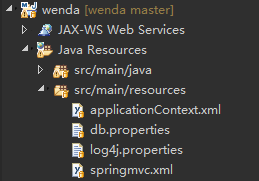
6）log4j日志包

7）jason包

8) apache文件上传

最后配置tomcat插件和资源加载插件，进行热部署（如果要放到云上和linnux上要配置，可以先不配置，部署的时候配置）

3、配置三个最基本的配置文件（xml）,除了web.xml房子web-info文件夹下，下述文件都放在resource下，这个是maven给出的虚拟目录，最后都会被加载到类路径下。



1）web.xml文件(放在web-info下):这是全局配置文件，目前主要是四点内容：

**上下文配置参数：**主要配置了spring的配置文件的路径

**监听器：**监听文件加载

**springMVC前端控制器**Distpacher Servlet配置，一般配置成拦截所有(只有拦截所有，才能让前端的请求都到controller中处理)，但是 .jsp 文件不会拦截，会当成拦截静态资源。

**字符编码过滤器：**用于统一编码格式

2）、applicationContext.xml文件，这个是最重要的部分，是spring的配置文件。目前主要配置了以下内容（红色是基本配置）：

**（重要）注解扫描:**只要使用到spring注解的文件都要加入到扫描中（mapper不用，下文会配置mapper的扫描器），目前只有service.impl包中需要被扫描。

加载配置文件：数据库参数配置的properties文件需要先加载，才能读出来，这样的方式更好维护。

数据源：将上面读取的参数注入到dataSource中

SqlSessionFactory:注入dataSource

扫描器:主要为了扫描mapper包，这里采用的是接口绑定方案，扫描后会自动声场mapper下相应的bean.

事务管理:利用的aop方式。一般切点为业务层，进行通知中进行回滚和提交等事务操作。用到了事务管理器包

（后期扩展）配置日志AOP：等于配置后置通知，将日志内容写到通知中，配置给相应的切点。

3）、springMVC.xml文件，目前主要包括了（红色是基本配置）：

注解扫描：注意controller包是由springMVC管理，springMVC是spring的子容器。在controller中可以注入spring管理的bean,如serviceImpl进行自动注入

注解驱动：驱动是handlerMapping和handlerAdapter

静态资源:因为dispacherServlet配置了拦截所有出了jsp,所以对于其他都会被认为是控制器别名来寻找控制器，这样无法访问到静态资源，因此在此设置放行

配置Multipart解析器:如果涉及到接收文件，需要配置

配置拦截器

4数据库参数配置文件和日志配置文件（properties），文件放置位置也是在resource下，和applicationContext.xml、springMVC.xml放在一起。

1）、db.properties:主要就是配置数据库访问的参数，如果改变数据库只需在这里改，与项目解耦。

2）、log4j.properties：配置日志输出格式，优先级，位置等等

至此，一个基本的SSM框架搭建成功。

**二、Maven下SSM引入velocity模板渲染。**

1、配置pom文件，在原来的基础上加入以下依赖，这里需要注意的是velocity-tools这个依赖要引入图中显示的这个，如果引入的是

<dependency>

<groupId>org.apache.velocity</groupId>

<artifactId>velocity-tools</artifactId>

<version>2.0</version>

</dependency>

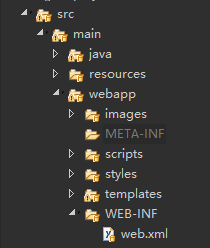
这个文件会报冲突。



2、配置springMVC文件，在原来的基础上加入以下内容，注意视图解析器一定要配置，以前在用jsp的时候是默认配置的，所以可以不写，但是用新的模板就一定要配置。其中还要注意的是，在maven管理下器webapp文件夹相当于以前的webContent。下图中模板设置路径的地方设置了velocity文件的存放路径，这里就相当于都放在webapp/templates文件夹中，后缀可以根据需求来，可加可不加。



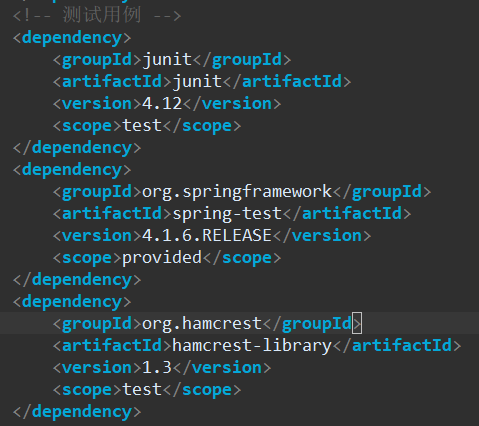
3、webapp文件夹下创建templates放html文件（这个和velocity在springMVC文件中的配置要对应，然后创建images、scripts、styles用于存放一些静态资源）



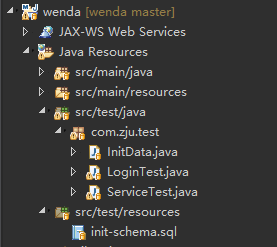
参考文章：<https://blog.csdn.net/tjcyjd/article/details/78495504>

**三、项目引入Junit单元测试。**

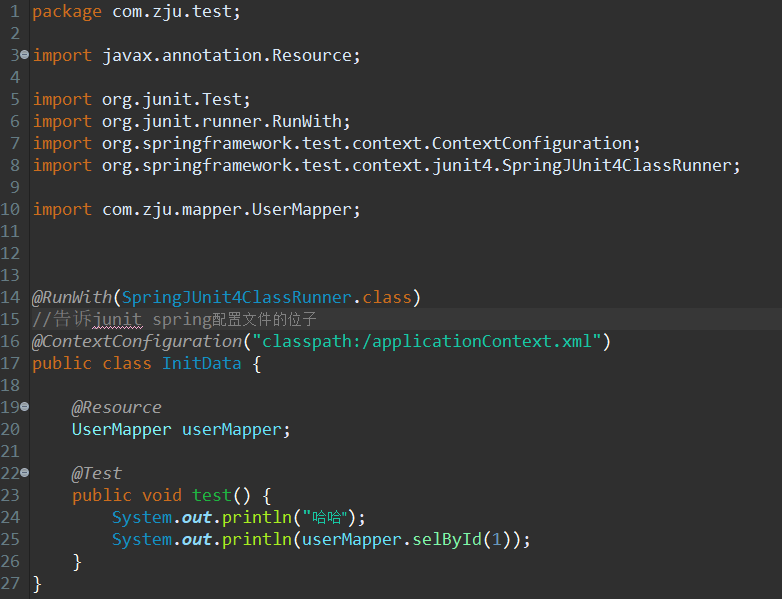
1、配置pom文件，在原来的基础上加入下图所示依赖包，第二包应该是属于spring的测试支持包，可以和前面的spring相关包放在一起，需要注意的是版本号，如果第二个包选择4.2.4会出错。因此没有特殊要求不要改动下面的版本号



2、写单元测试样例类。Maven管理下的文件目录结构可知，在src下有两组相同的目录，即main分支和test分支，单元测试类就写在test分支，和main一样，test分支同样的也有resources目录，用于存放一些配置文件或者资源，而java下就用于写测试代码，进行普通的建包写类。



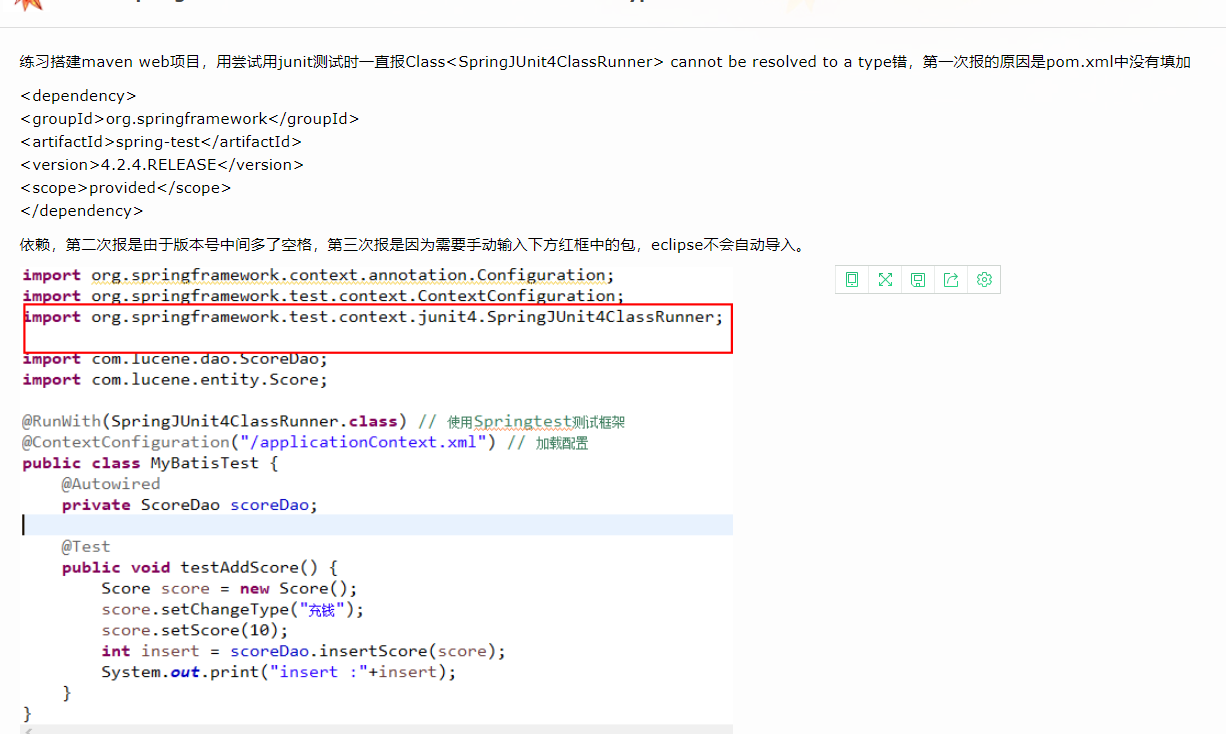
如上图所示新建InitData.java测试样例，格式如下图所示。（下面用到了注解，但是并不用在spring或者springMVC的配置文件中进行注解扫描配置）。



3、出现过的问题：

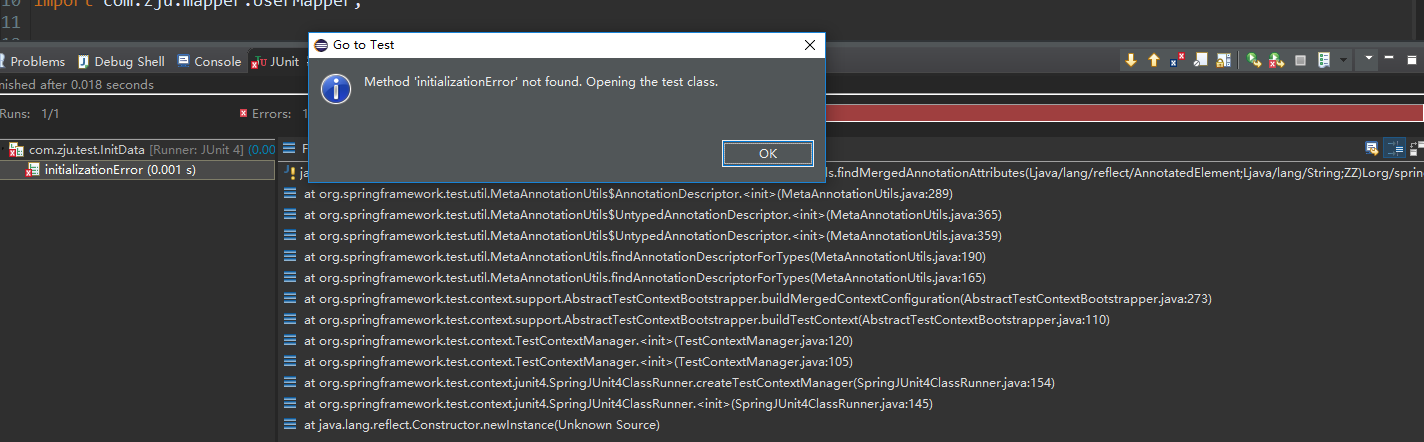
1、Class<SpringJUnit4ClassRunner> cannot be resolved to a type

解决方案：

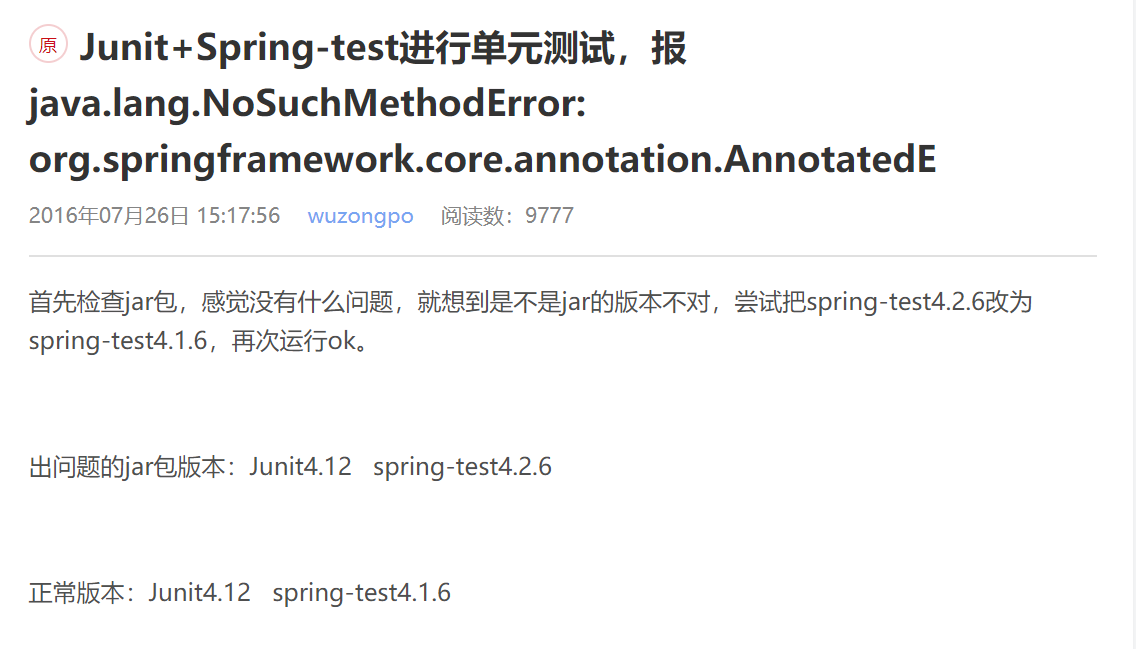


参考：<https://www.cnblogs.com/xiaopu123/p/7396103.html>，注意该文章中依赖的版本号和我们配置的会出现错误，因此产生下面第2个问题

2、导包版本错误

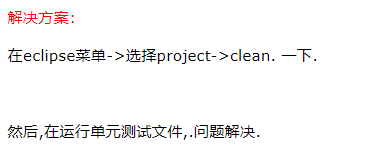


解决：



参考<https://blog.csdn.net/wuzongpo/article/details/52036096>

3、[junit单元测试时，出现java.lang.ClassNotFoundException](https://www.cnblogs.com/Jacob186/articles/4898616.html)，测试样例有时候会突然无法加载，而之前使用一直正常，没有任何改动。



参考：<https://www.cnblogs.com/Jacob186/articles/4898616.html>