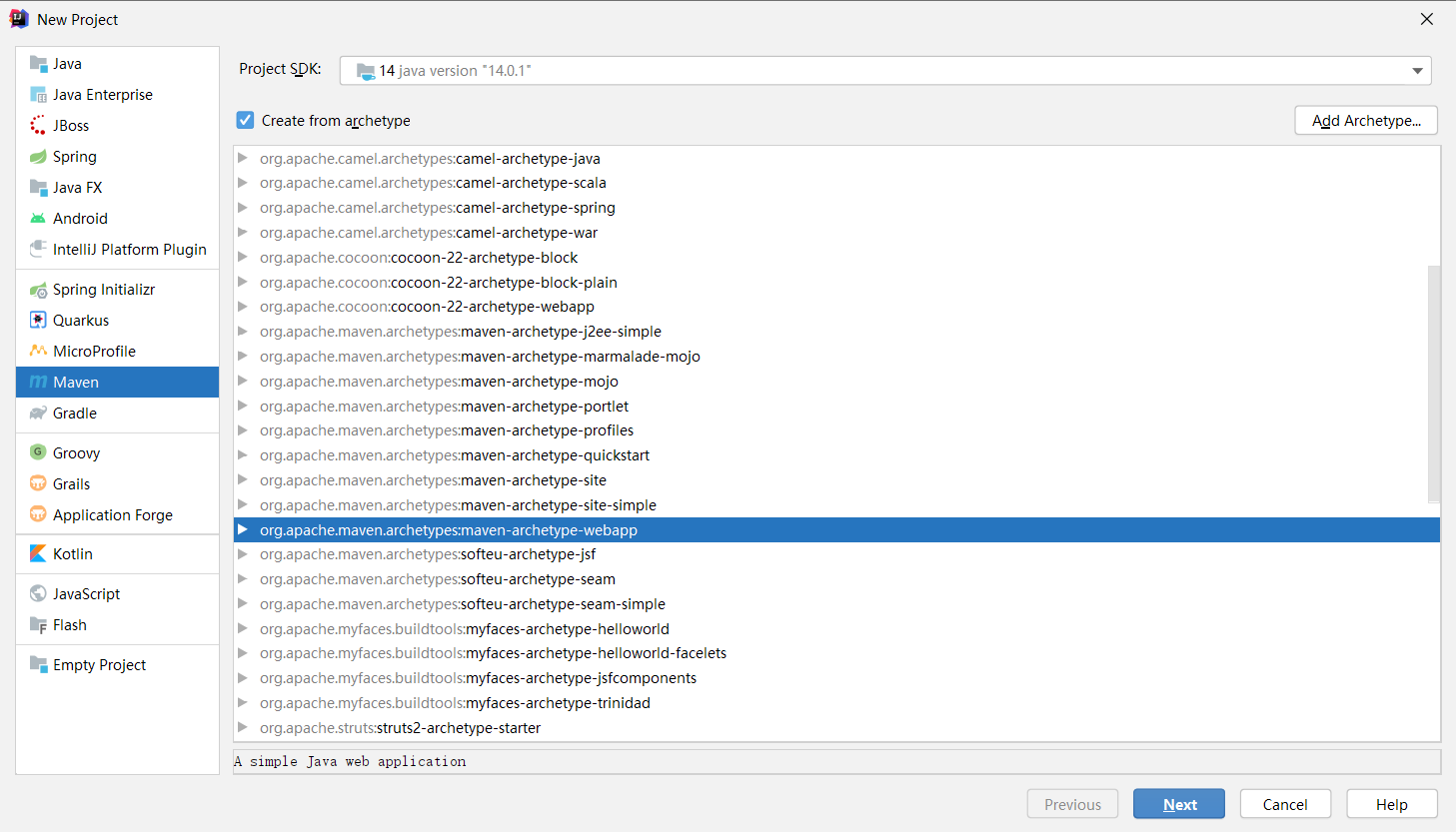
首先选好一个模板：



新建Maven工程，导入相关依赖：



SSM：Spring、SpringMVC、Mybatis。

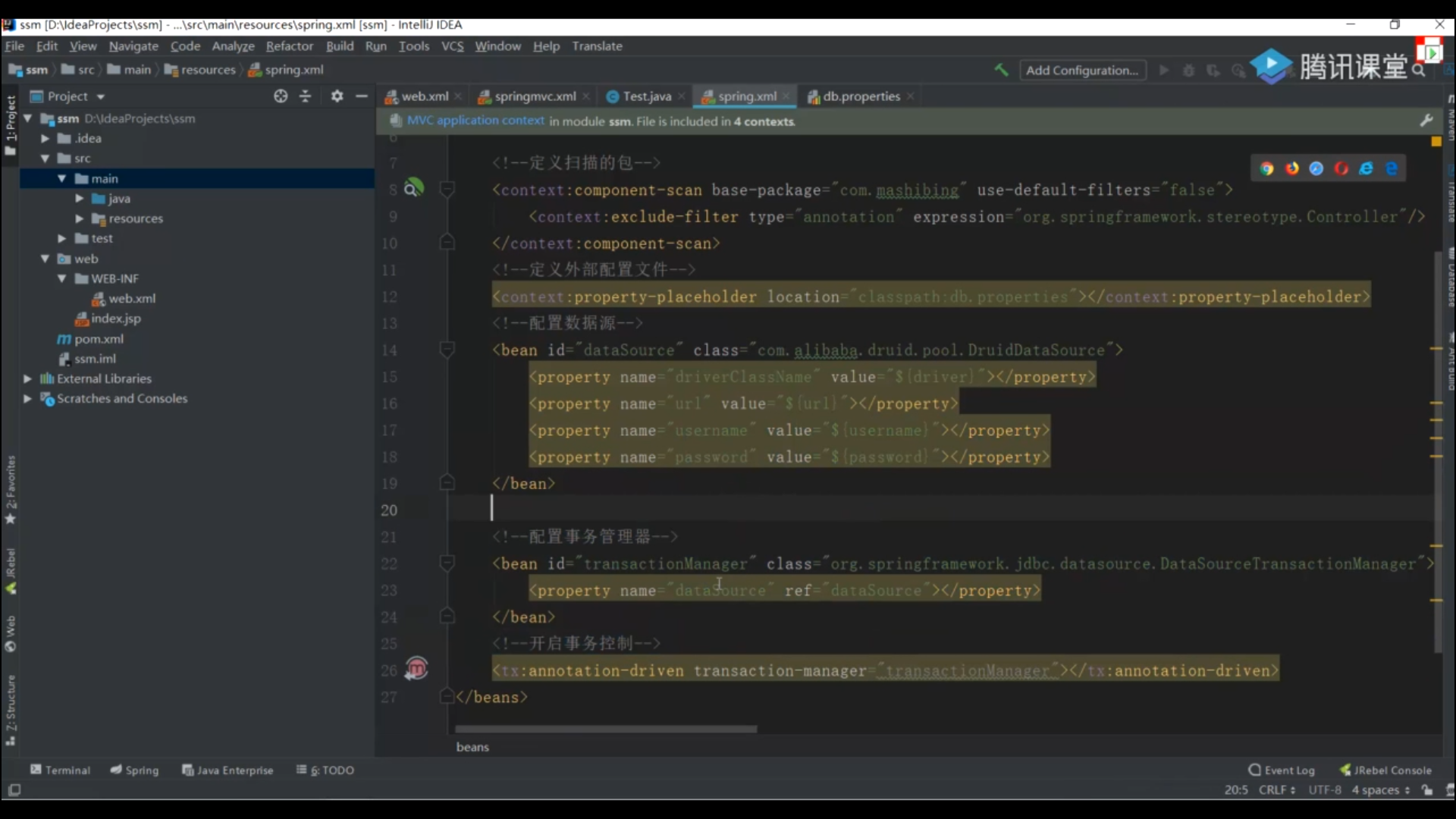
其中Spring负责管理这两个组件。在pom.xml层面，由于SpringMVC本身就是Spring的一个子模块，所以他们两个不需要再整合，只需要将Mybatis与Spring整合即可

相关依赖：

<dependencies>  
 *<!--SpringMVC 相关组件-->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.6</version>  
 </dependency>  
  
 *<!--Spring JDBC-->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>5.3.9</version>  
 </dependency>  
  
 *<!--Spring AOP-->* <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.6</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>5.2.12.RELEASE</version>  
 </dependency>  
  
 *<!--Mybatis-->* <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.5.7</version>  
 </dependency>  
  
 *<!--Mybatis整合Spring-->* <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>2.0.6</version>  
 </dependency>  
  
 *<!--MySQL驱动-->* <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.25</version>  
 </dependency>  
  
 *<!--C3P0-->* <dependency>  
 <groupId>c3p0</groupId>  
 <artifactId>c3p0</artifactId>  
 <version>0.9.1.2</version>  
 </dependency>  
  
 *<!--JSTL-->* <dependency>  
 <groupId>jstl</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
  
 *<!--ServletAPI-->* <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 </dependency>  
</dependencies>

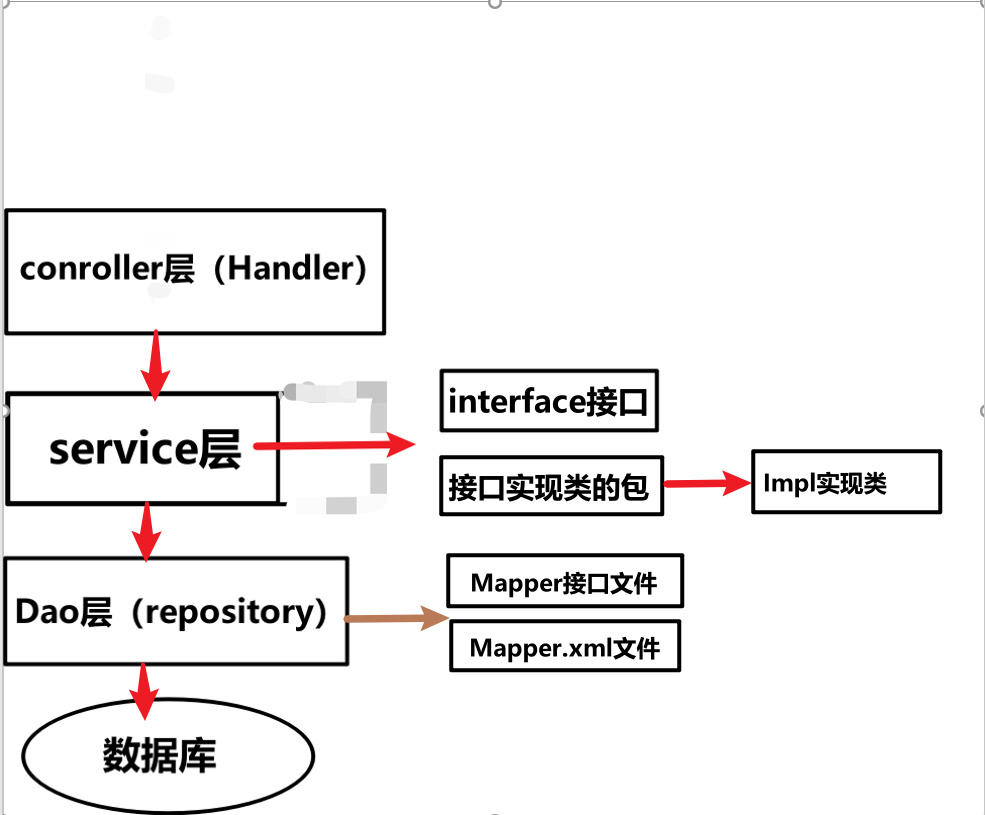
web.xml中配置SpringMVC、Spring、字符编码过滤器、加载静态资源

上面学习的是使用c3p0的数据库连接池，如果想使用Druid：其中db.properties写在resource文件夹下



这里使用语言进行整体的一个易于理解的总结，在配置好SSM框架的各种resources文件之后，对接好相应的数据库之后，然后需要在java包下创建entity包，用于存放javabean类，对接好数据库中表格的属性值。

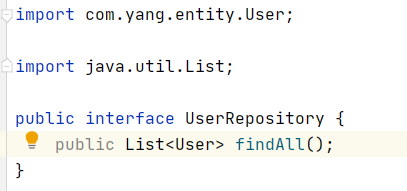
然后由底层向上。依次建包



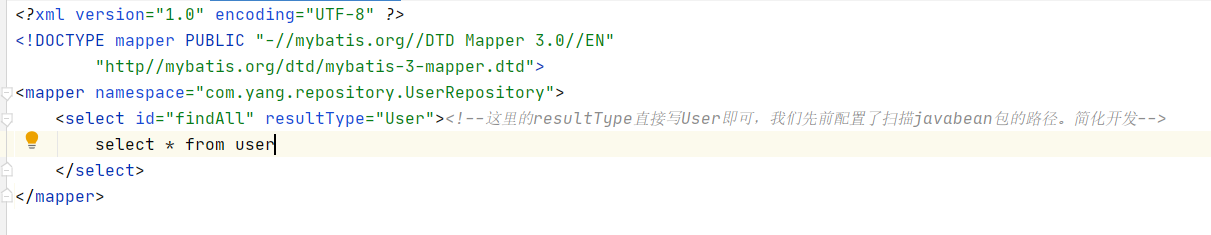
现在假设对数据库中的一个user表格使用SSM框架输出到客户端

首先建立repository包，其中包含：UserRepository接口，里面有一个findAll()方法，返回值是一个List集合类型。然后再创建UserRepository.xml文件，引入头文件后，使用Mapper标签的namespace属性指向刚刚创键的UserRepository接口，然后使用select标签，在里面编写相应的SQL语句。

**Repository：**



**Repository.xml:**



至此repository创建完毕，然后来创建service包，首先创建UserService接口，其中也包括一个findAll()方法，然后在service包下创建一个子包Impl，UserServiceImpl实现子类，实现service接口，实现findAll()方法，然后注入UserRepository对象，使用@AutoWired标签自动装载。

**然后编写controller层：**



创建对应controller包，然后创建一个UserHandler类，使用@Controller注解，创建一个findAll()方法，使用ModelAndView模型数据传递，但是具体传递的值，使用的是注入的UserService对象中的findAll()方法。然后返回的视图解析是showUser，在webapp下创建showUser.jsp:

因为传递的user数据是一个集合类型，包括所有的user



所以在jsp页面直接使用foreach标签将list集合中的数据遍历出来即可



调用关系就是：在客户端调用Handler中的findAllUser()方法，方法的返回值调用的是service中的实现子类UserServiceImpl的findAll()方法，在service中调用与数据库相关联的底层操作Dao（repository）层中的UserRepository的findAll()方法。

大致关系就是如此