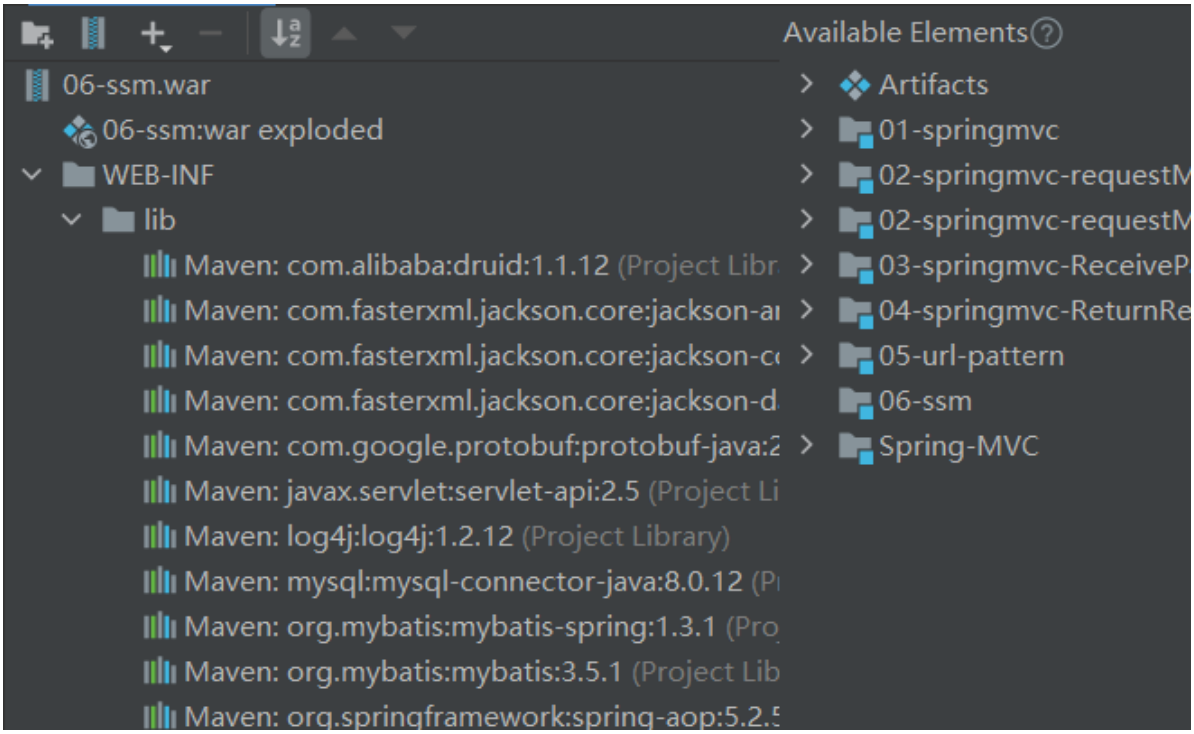
SSM整合步骤，IDEA

一、创建Maven+web工程

IDEA中手动添加web支持的时候，不会默认把Artifacts里面项目的jar包放到lib包中，这个得自己手动加，如果是采用骨架方式创建的工程则不用，但是我习惯手动，所以得记住每次弄完需要Artifacts中手动创建lib文件夹,切记啊，如果你运行tomcat有NoClassDefFoundError异常，可以看看有没有lib文件夹



二、pom.xml中添加相关依赖

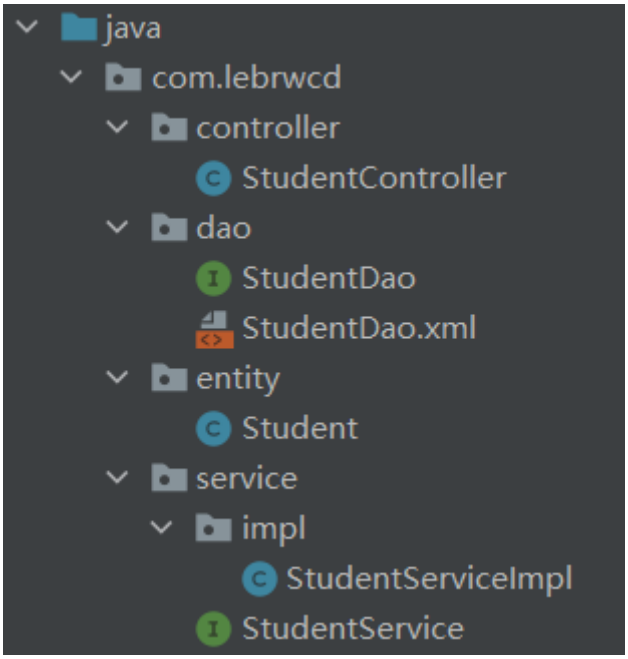
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>groupId</groupId>
    <artifactId>06-ssm</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>

    <dependencies>
        <!--spring核心jar-->
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>
        <!--spring-tx-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>
        <!--spring-jdbc-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>
        <!--servlet-->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
        </dependency>
        <!--springmvc-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>
        <!--jackson插件-->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.9.0</version>
        </dependency>
        <!--jackson插件-->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.9.0</version>
        </dependency>
        <!--mybatis-spring-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.3.1</version>
        </dependency>
        <!--mybatis-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.5.1</version>
        </dependency>
        <!--连接池-->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.1.12</version>
```

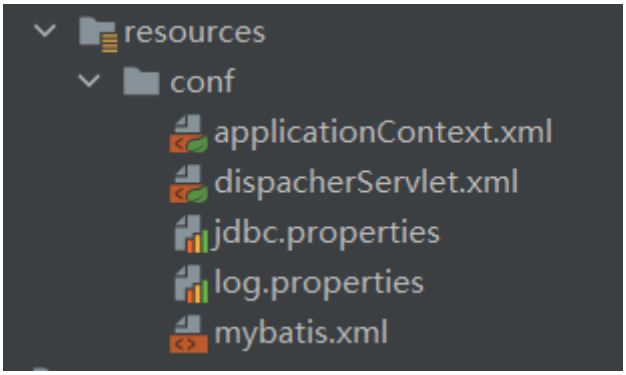
```
        </dependency>
        <!--log4j日志工具-->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.12</version>
        </dependency>
        <!--mysql驱动-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.12</version>
        </dependency>
    </dependencies>

    <!--当你出现这个异常
org.apache.ibatis.binding.BindingException: Invalid bound statement (not found),
尝试配置以下语句-->
    <build>
        <resources>
            <resource>
                <directory>src/main/java</directory>
                <includes>
                    <!--生成配置文件-->
                    <include>**/*.xml</include>
                </includes>
            </resource>
        </resources>
    </build>
</project>
```

```
用于在classes目录下生成配置文件
<build>
        <resources>
            <resource>
                <directory>src/main/java</directory>
                <includes>
                    <!--生成配置文件-->
                    <include>**/*.xml</include>
                </includes>
            </resource>
            <resource>
                <directory>src/main/resources</directory>
                <includes>
                    <!--生成配置文件-->
                    <include>**/*.xml</include>
                    <include>**/*.properties</include>
                </includes>
            </resource>
        </resources>
    </build>
```

三、创建相关包，controller,service,dao,entity



四、配置文件，包括spring配置文件，springMVC配置文件，mybatis主配置文件，mybatis映射文件，日志配置文件,jdbc配置文件



1.spring配置文件:applicationContext.xml

主要负责:声明service,dao,工具类，事务配置

```
<?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.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd">

 <!--spring容器负责创建service,dao对象-->

<!--扫描service注解以创建service对象-->
    <context:component-scan base-package="com.lebrwcd.service"/>
    <!--声明jdbc配置文件路径-->
    <context:property-placeholder location="classpath:conf/jdbc.properties"/>

<!--这是创建dao-->
    <!--数据源-->
```

```
    <bean id="dataSource" class="${dataSource}">
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.user}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <!--SqlSessionFactory-->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!--数据源-->
        <property name="dataSource" ref="dataSource"/>
        <!--Mybatis主配置文件路径-->
        <property name="configLocation" value="classpath:conf/mybatis.xml"/>
    </bean>

    <!--动态代理对象,用于创建dao接口的实现类，采用动态代理方式，
        默认实现类名称是dao接口首字母小写-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
        <property name="basePackage" value="com.lebrwcd.dao"/>
    </bean>
</beans>
```

2.springMVC配置文件:dispacherServlet.xml(名字随意，推荐这个)

主要负责:声明controller,视图解析器等web开发中的对象

```
<?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
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!--springmvc容器，负责视图解析器，控制器对象,springmvc以注解开发为主，所以也需要
    声明一个注解驱动，注解驱动的作用很大-->

    <!--组件扫描器-->
    <context:component-scan base-package="com.lebrwcd.controller"/>

    <!--视图解析器-->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/view/"/>
        <property name="suffix" value=".html"/>
    </bean>

    <!--注册注解驱动(核心)-->
    <mvc:annotation-driven/>
</beans>
```

3.mybatis主配置文件:mybatis.xml

主要负责:起别名，这样mapper映射文件就可以采用别名机制，同时指定mapper文件的位置

```
<?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>
    <!--起别名，实体类所在的包-->
    <typeAliases>
        <package name="com.lebrwcd.entity"/>
    </typeAliases>

    <!--sql mapper文件的位置-->
    <mappers>
        <package name="com.lebrwcd.dao"/>
    </mappers>
</configuration>
```

4.mybatis映射文件：xxxxDao.xml

主要负责：编写sql语句，与数据库进行交互返回数据，mybatis映射文件一般放在dao包中，与dao接口名字一致

```
<?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">
<!--namespace是dao接口的全限定名称-->
<mapper namespace="com.lebrwcd.dao.StudentDao">

    <!--插入学生-->
    <insert id="insertStudent">
        insert into t_student(id,name,email,age)values (#{id},#{name},#{email},#
{age}) ;
    </insert>
    <!--查询学生-->
    <!--只有select语句才有resultType-->
    <select id="selectStudent" resultType="Student">
       select id,name,email,age from t_student order by id desc;
    </select>
</mapper>
```

5.其他配置文件(可选)

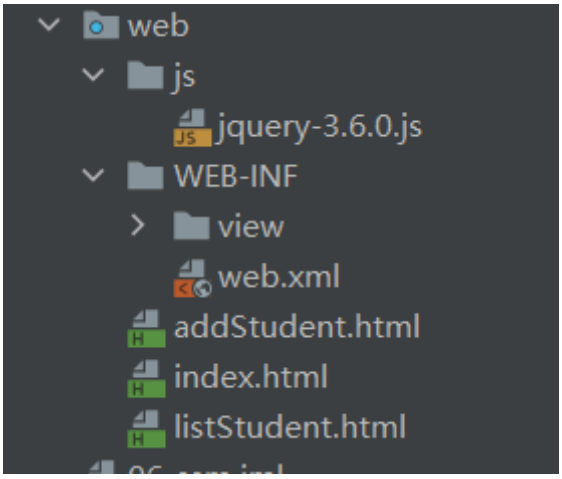
log4j

```
log4j.rootLogger=DEBUG,Console
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n
log4j.logger.org.apache=INFO
```

jdbc属性资源配置文件

```
dataSource=com.alibaba.druid.pool.DruidDataSource
jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/springdb
jdbc.user=root
jdbc.password=wcd0209
```

五、web层的配置



web.xml文件的配置

主要负责：ContextLoaderListener的声明(spring配置文件)，作用是创建WebapplicationContext,可以使得其可以在servletContext的作用域中，WebapplicationContext只创建一次；中央调度器的创建DispacherServlet(springMVC配置文件);字符串过滤器的创建:防止post请求乱码

```
<?xml version="1.0" encoding="UTF-8"?>
<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_4_0.xsd"
         version="4.0">

    <!--ContextLoaderListener-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:conf/applicationContext.xml</param-value>
    </context-param>
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!--log4j-->
    <!--字符集过滤器-->
    <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>forceRequestEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>forceResponseEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>characterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!--中央调度器-->
    <servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:conf/dispacherServlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <!--urlpattern可以是*.do或者/,  /的话就成为了默认servlet，访问不了静态资源，
            需要配置-->
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

</web-app>
```

======到这，基本环境已经搭建完毕==========

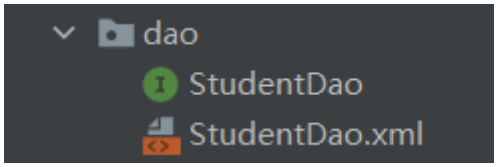
例子:简易完成学生的注册与查询

数据库:

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
id	int	10		☑	☐	🔑 1
name	varchar	255		☑	☐	
email	varchar	255		☑	☐	
age	int	10		☑	☐	

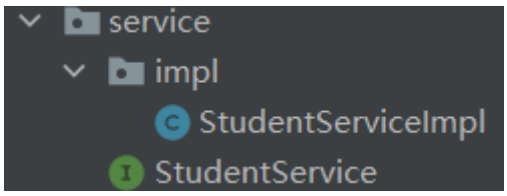
dao:



```java
package com.lebrwcd.dao;

import com.lebrwcd.entity.Student;

import java.util.List;

public interface StudentDao {
     int insertStudent(Student student);
     List<Student> selectStudent();
}
```

```xml
<?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.lebrwcd.dao.StudentDao">

    <!--插入学生-->
    <insert id="insertStudent">
        insert into t_student(id,name,email,age)values (#{id},#{name},#{email},#
{age}) ;
    </insert>
    <!--查询学生-->
    <select id="selectStudent" resultType="Student">
       select id,name,email,age from t_student order by id desc;
    </select>
</mapper>
```

sevice:




```java
package com.lebrwcd.service;

import com.lebrwcd.entity.Student;

import java.util.List;

public interface StudentService {
    int addStudent(Student student);
    List<Student> querryStudent();
}
```

```java
package com.lebrwcd.service.impl;

import com.lebrwcd.dao.StudentDao;
import com.lebrwcd.entity.Student;
import com.lebrwcd.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```java
import java.util.List;

/*sevice注解，由于spring配置文件中声明了组件扫描器
<context:component-scan base-package="com.lebrwcd.service"/>
所以之后spring之后会创建service对象，StudentDao是引用类型，才有@Autowired或者@Resource
完成对象的创建*/
@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentDao studentDao;
    @Override
    public int addStudent(Student student) {
        return studentDao.insertStudent(student);
    }

    @Override
    public List<Student> querryStudent() {
        return studentDao.selectStudent();
    }
}
```

entity实体类:

```java
package com.lebrwcd.entity;
public class Student {
    private Integer id;
    private String name;
    private String email;
    private Integer age;

    public Student() {
    }

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```
    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", email='" + email + '\'' +
                '}';
    }
}
```

controller:

```
package com.lebrwcd.controller;

import com.lebrwcd.service.StudentService;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
import com.lebrwcd.entity.Student;
import javax.annotation.Resource;
import java.util.List;

/*@RequestMapping注解放在类上，value值是模板    /student/add.do
                                                /student/querry.do  */
@RequestMapping("/student")
@Controller
public class StudentController {

    //service对象
    @Resource
    private StudentService studentService;

    //学生注册
    @RequestMapping(value = "/add.do")
    public ModelAndView doAdd(Student student){
        ModelAndView mv = new ModelAndView();
        int count = studentService.addStudent(student);
        if(count > 0 ){
            //添加成功
            mv.setViewName("success");
        }else{
            //注册失败
            mv.setViewName("error");
        }
        return mv;
    }
```

```
    //查询学生列表
    @RequestMapping(value = "/querry.do")
    @ResponseBody
    public List<Student> doQuerry(){
        return studentService.querryStudent();
    }
}
```

首页:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>首页</title>
</head>
<body>
    <div align="center">
        <a href="addStudent.html">学生注册</a>
        <hr color="blue" width="100%">
        <a href="listStudent.html">学生列表</a>
    </div>
</body>
</html>
```

添加学生:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>学生注册</title>
</head>
<body>
<div align="center">
    <form action="student/add.do" method="post">
        <table>
            <tr>
                <td>用户id</td>
                <td><input type="text" name="id"></td>
            </tr>
            <tr>
                <td>用户名</td>
                <td><input type="text" name="name"></td>
            </tr>
            <tr>
                <td>用户邮箱</td>
                <td><input type="text" name="email"></td>
            </tr>
            <tr>
                <td>年龄</td>
                <td><input type="text" name="age"></td>
            </tr>
            <tr>
                <td><input type="submit" value="注册"></td>
            </tr>
```

```
        </table>
    </form>
</div>
</body>
</html>
```

学生列表:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>学生列表</title>
    <script type="text/javascript" src="js/jquery-3.6.0.js"></script>
    <script type="text/javascript">
        $(function (){
            //点击学生列表超链接就可以显示所有学生信息表格
            getStudentInfo();
            $("#doAjax").click(function () {
                //发送ajax请求
                getStudentInfo();
            });
            function getStudentInfo(){
                $.ajax({
                    url:"student/querry.do",
                    dataType:"json",
                    success : function (response){
                        //清空
                        $("#stuInfo").empty();
                        $.each(response,function (i,n){
                            $("#stuInfo").append(
                                "<tr>" +
                                "<td>" + n.id +"</td>"
                                +"<td>"+ n.name +"</td>"
                                +"<td>" + n.email +"</td>"
                                +"<td>" + n.age + "</td>"+
                                "</tr>"
                            )
                        })
                    }
                });
            }


        });
    </script>
</head>
<body>
    <div align="center">
        <p>查看学生列表<button id="doAjax">获取学生信息</button></p>
        <table border="1px" width="50%">
            <thead>
                <tr>
                    <td>id</td>
                    <td>name</td>
                    <td>email</td>
                    <td>age</td>
```

```
                </tr>
            </thead>
            <tbody id="stuInfo">

            </tbody>
        </table>
    </div>
</body>
</html>
```

学生注册

学生列表

用户id

用户名

用户邮箱

年龄

注册

用户id 222

用户名 张三

用户邮箱 zs@163.cn

年龄 46

注册

11	222	张三	zs@163.cn	46

查看学生列表 获取学生信息

id	name	email	age
222	张三	zs@163.cn	46
81	wxx	wxx@qq.com	16
75	yr	yr@qqqq.cn	19
56	wsd	wsd@184.com	18
45	las	las@dd.com	45
31	chx	chx@hh.cn	21
15	trr	trr@qq.cn	16
14	wws	wws@qq.com	15
13	wkkd	wkkd@ee.com	19
12	wad	wad@qq.com	14
7	ppcdha	ppha@qq.com	18

简易版，有很多不规范的地方，请谅解，但主要是演示ssm整合的步骤

整合步骤出现的异常：

org.apache.ibatis.binding.BindingException: Invalid bound statement (not found)

网络搜集方案:

截图为网络中搜索到的常见原因：

Invalid bound statement (not found)错误的可能原因

其他原因导致此问题解决参考：

1.检查xml文件所在package名称是否和Mapper interface所在的包名

```
<mapper namespace="me.tspace.pm.dao.UserDao">
```

mapper的namespace写的不对！！！注意系修改。

2.UserDao的方法在UserDao.xml中没有，然后执行UserDao的方法会报此

3. UserDao的方法返回值是List<User>,而select元素没有正确配置ResultMap,或者只配置ResultType!

4. 如果你确认没有以上问题,请任意修改下对应的xml文件,比如删除一个空行,保存.问题解决

5.看下mapper的XML配置路径是否正确

```
mybatis:
  #配置mapper xml文件所在的路径
  mapper-locations: classpath*:com/fz/hr/modules/dao/**/*.xml
  #配置映射类所在的包名
  type-aliases-package: com.fz.hr.modules.domain
```

如果如上办法都解决不了，可以尝试在pom.xml中增加如下配置：

```
src/main/java

  */.xml
```

java.lang.NoClassDefFoundError: org/springframework/dao/support/DaoSupport

尝试在pom.xml中添加spring-tx依赖