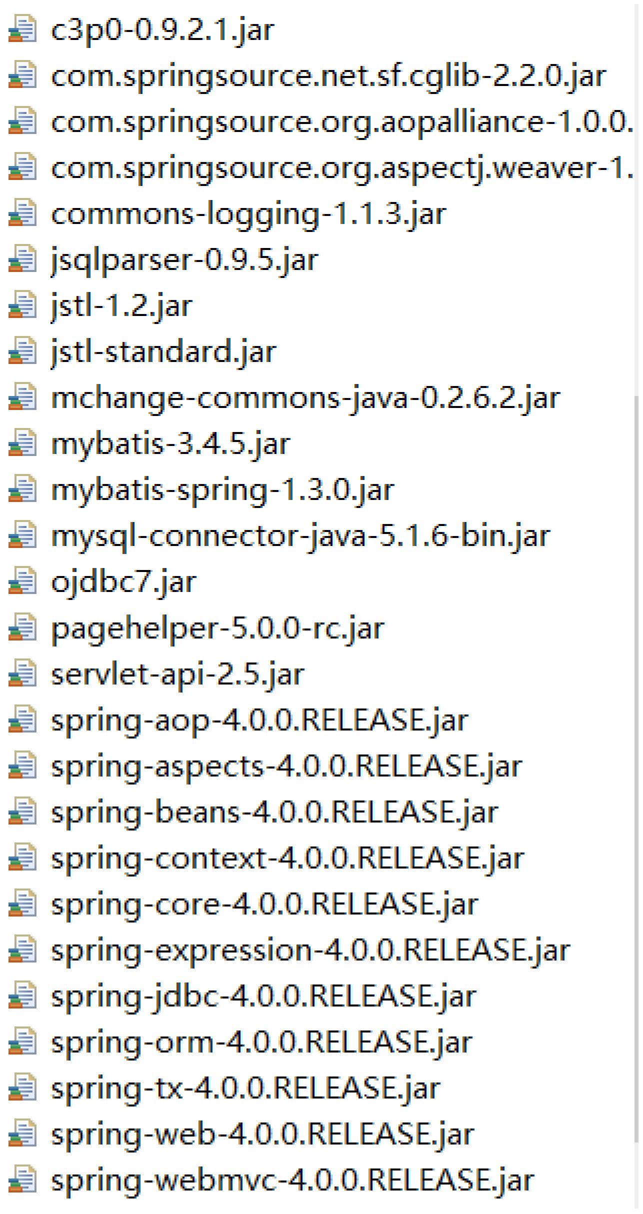
SSM搭建

```
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-
spring-1.2.xsd   http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">   <!--
Spring管理所有的业务逻辑组件,不扫描@Controll控制器 -->  <context:component-
scan base-package="com.zkx">   <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller" />  </context:component-
scan>     <!-- 引入数据库的配置文件 -->   <context:property-placeholder
location="classpath:dbconfig.properties" />         <!-- 数据源 c3p0 -->   <bean
id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
<property name="jdbcUrl" value="${orcl.url}"></property>   <property
name="driverClass" value="${orcl.driver}"></property>   <property name="user"
value="${orcl.username}"></property>   <property name="password"
value="${orcl.password}"></property>  </bean>    <!-- Spring内置数据源配置 -->
<!-- <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
驱动类         <property name="driverClassName"
value="oracle.jdbc.driver.OracleDriver"></property>         url地址         <property
name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"></property>
<property name="username" value="C##RZ"></property>         <property
name="password" value="123"></property>     </bean> -->     <!-- spring事务管理 --
>  <bean id="dataSourceTransactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<property name="dataSource" ref="dataSource"></property>  </bean>   <!--
开启基于注解的事务 --> <tx:annotation-driven transaction-
manager="dataSourceTransactionManager"/>     <!--
整合mybatis,spring管理mapper的实现,spring用来管理事务  -->    <!--
创建出SqlSessionFactory对象  -->  <bean id="sqlSessionFactoryBean"
class="org.mybatis.spring.SqlSessionFactoryBean">   <property name="dataSource"
ref="dataSource"></property>   <!-- configLocation指定全局配置文件的位置 -->
<property name="configLocation" value="classpath:mybatisconfig.xml"></property>
<!--mapperLocations: 指定mapper文件的位置-->   <property
name="mapperLocations" value="classpath:com/zkx/dao/*.xml"></property>
</bean>       <!--
扫描所有的mapper接口的实现，让这些mapper能够自动注入；base-
package：指定mapper接口的包名  老版：<property name="basePackage"
value="com.softjx.dao"></property>  新版： <mybatis-spring:scan base-
package="com.softjx.dao"/>   -->              <bean
```

1. lib

1.1.



```
c3p0-0.9.2.1.jar
com.springsource.net.sf.cglib-2.2.0.jar
com.springsource.org.aopalliance-1.0.0.
com.springsource.org.aspectj.weaver-1.
commons-logging-1.1.3.jar
jsqlparser-0.9.5.jar
jstl-1.2.jar
jstl-standard.jar
mchange-commons-java-0.2.6.2.jar
mybatis-3.4.5.jar
mybatis-spring-1.3.0.jar
mysql-connector-java-5.1.6-bin.jar
ojdbc7.jar
pagehelper-5.0.0-rc.jar
servlet-api-2.5.jar
spring-aop-4.0.0.RELEASE.jar
spring-aspects-4.0.0.RELEASE.jar
spring-beans-4.0.0.RELEASE.jar
spring-context-4.0.0.RELEASE.jar
spring-core-4.0.0.RELEASE.jar
spring-expression-4.0.0.RELEASE.jar
spring-jdbc-4.0.0.RELEASE.jar
spring-orm-4.0.0.RELEASE.jar
spring-tx-4.0.0.RELEASE.jar
spring-web-4.0.0.RELEASE.jar
spring-webmvc-4.0.0.RELEASE.jar
```

2. web.xml

2.1. <!-- 配置启动 Spring IOC 容器的 Listener,启动spring容器 -->

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>

  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
```

2.2. <!-- 表单提交controller获得中文参数后乱码解决方案 注意：

jsp页面编码设置为UTF-8

form表单提交方式为必须为post，get方式下面spring编码过滤器不起效果 -->

```
   <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>
```

2.3.<!-- 可以把 POST 请求转为 DELETE 或 PUT 请求 -->

```
  <filter>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-
class>
  </filter>

  <filter-mapping>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

2.4.<!-- 配置 DispatcherServlet -->

```
  <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:springmvc.xml</param-value>
    </init-param>

    <load-on-startup>1</load-on-startup>

  </servlet>

  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
```

3. applicationContext.xml

3.1. <?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:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
   xmlns:tx="http://www.springframework.org/schema/tx"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
   http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-
spring-1.2.xsd
      http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
      http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">



<!-- Spring管理所有的业务逻辑组件,不扫描@Controll控制器 -->
   <context:component-scan base-package="com.zkx">
      <context:exclude-filter type="annotation"
         expression="org.springframework.stereotype.Controller" />
   </context:component-scan>


   <!-- 引入数据库的配置文件 -->
    <context:property-placeholder location="classpath:dbconfig.properties" />


    <!-- 数据源 c3p0 -->
```

```
 <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
   <property name="jdbcUrl" value="${orcl.url}"></property>
   <property name="driverClass" value="${orcl.driver}"></property>
   <property name="user" value="${orcl.username}"></property>
   <property name="password" value="${orcl.password}"></property>
</bean>
 <!-- Spring内置数据源配置 -->
<!-- <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">

    驱动类
    <property name="driverClassName"
value="oracle.jdbc.driver.OracleDriver"></property>
    url地址
    <property name="url"
value="jdbc:oracle:thin:@localhost:1521:orcl"></property>
    <property name="username" value="C##RZ"></property>
    <property name="password" value="123"></property>
 </bean> -->



<!-- spring事务管理 -->
<bean id="dataSourceTransactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="dataSource"></property>
</bean>

<!-- 开启基于注解的事务 -->
<tx:annotation-driven transaction-manager="dataSourceTransactionManager"/>
```

```
  <!--
  整合mybatis,spring管理mapper的实现,spring用来管理事务
  -->

  <!--创建出SqlSessionFactory对象  -->
  <bean id="sqlSessionFactoryBean"
class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"></property>
    <!-- configLocation指定全局配置文件的位置 -->
    <property name="configLocation"
value="classpath:mybatisconfig.xml"></property>
    <!--mapperLocations: 指定mapper文件的位置-->
    <property name="mapperLocations"
value="classpath:com/zkx/dao/*.xml"></property>
  </bean>



  <!-- 扫描所有的mapper接口的实现，让这些mapper能够自动注入；base-
package：指定mapper接口的包名
  老版：<property name="basePackage" value="com.softjx.dao"></property>
  新版： <mybatis-spring:scan base-package="com.softjx.dao"/>
   -->



  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="com.zkx.dao"></property>
  </bean>
```

```
</beans>
```

4. springmvc.xml

4.1.
```
<?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
http://www.springframework.org/schema/context/spring-context-4.0.xsd
    http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">


  <!-- 配置自定扫描的包 -->
  <context:component-scan base-package="com.zkx" use-default-filters="false">
    <context:include-filter type="annotation"
      expression="org.springframework.stereotype.Controller"/>
  </context:component-scan>


  <!-- 配置视图解析器: 如何把 handler 方法返回值解析为实际的物理视图 -->
  <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value=""></property>
    <property name="suffix" value=".jsp"></property>
  </bean>
```

```
    <!-- 处理静态资源 -->
    <mvc:default-servlet-handler/>



    <!-- 默认配置方案。
    并提供了：数据绑定支持，@NumberFormatannotation支持，
    @DateTimeFormat支持，@Valid支持，读写XML的支持（JAXB），
    读写JSON的支持（Jackson）。
        后面，我们处理响应ajax请求时，就使用到了对json的支持。
     -->
    <mvc:annotation-driven></mvc:annotation-driven>



</beans>
```

5. mybatis.xml

```
5.1. <?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>

  <settings>
    <!--
显式的指定每个我们需要更改的配置的值，即使他是默认的。防止版本更新带来
```

```
的问题 -->
    <setting name="lazyLoadingEnabled" value="true"/>
    <setting name="aggressiveLazyLoading" value="false"/>
  </settings>
  <!-- <plugins>
    <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>

  </plugins> -->
</configuration>
```

6. 步骤

6.1. 1.建立POJO实体对象

6.2. 2.DAO层

6.2.1. UserMapper.java(接口)

6.2.2. UserMapper.xml

namespace 中的要和接口名保持一致,sql 的ID 要和接口中的方法一致

```
<mapper namespace="com.zkx.dao.UserMapper" >
</mapper>
```

6.3. 3.Service层

6.3.1. UserService.java(接口)(方法和DAO层的接口保持一致)

6.3.2. UserServiceImpl.java

类前面需要注解@Service("userService")

@Transactional

该类需要私有的成员变量,通过spring自动注入的方式获取DAO层的对象,改对象可以直接使用DAO层接口中的方法

```
        @Autowired
        private UserMapper userMapper;
```

6.4. 4.Test测试

6.4.1. private static ApplicationContext context = null;

```
    private UserService userService = null;


@Before
    public void init() throws Exception {
        context = new ClassPathXmlApplicationContext("applicationContext.xml");
        userService = (UserService) context.getBean("userService");


    }
@Test
    public void TestUser() {


    }
```

6.5. 5.Controller层

6.5.1. 类注解,表明它是一个控制器,以便让springMVC扫描到:

```
    @Controller
    @RequestMapping("user")
```

自动注入一个Service层的对象:

```
    @Autowired
    private UserService userService;
```

方法注解,提供uri的路径:

```
    @RequestMapping("/selAll")
```

返回值:String 代表返回的视图路径

形参:可以传回到视图层

6.6. 6.视图层

6.6.1. JSTL会自动识别对象里面的属性,EL表达式可以取出四大域中的对象

```
<c:forEach var="user" items="${ List}">
        <td>${user.id}</td>
</c:forEach>
```