基于 SSM 框架的 Soso 移动业务大厅

Java 作业 4-潘巧巧-1613415

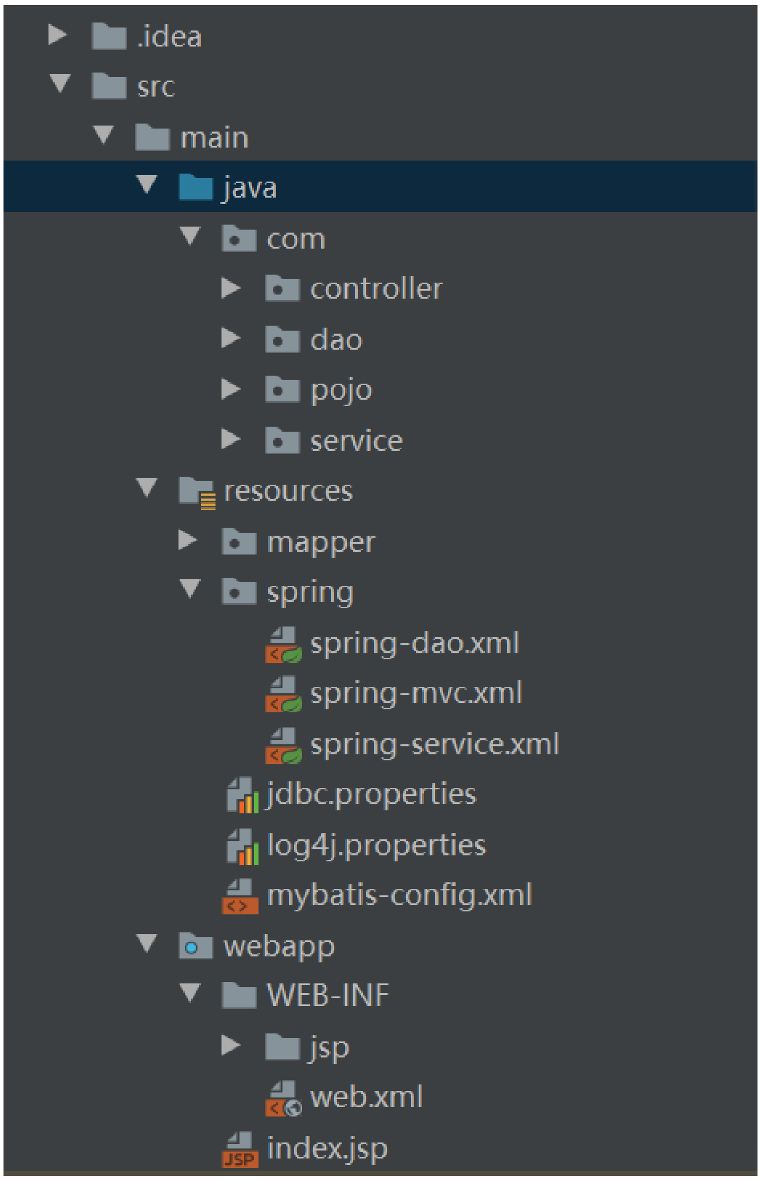
目录

一、整体基本实现情况

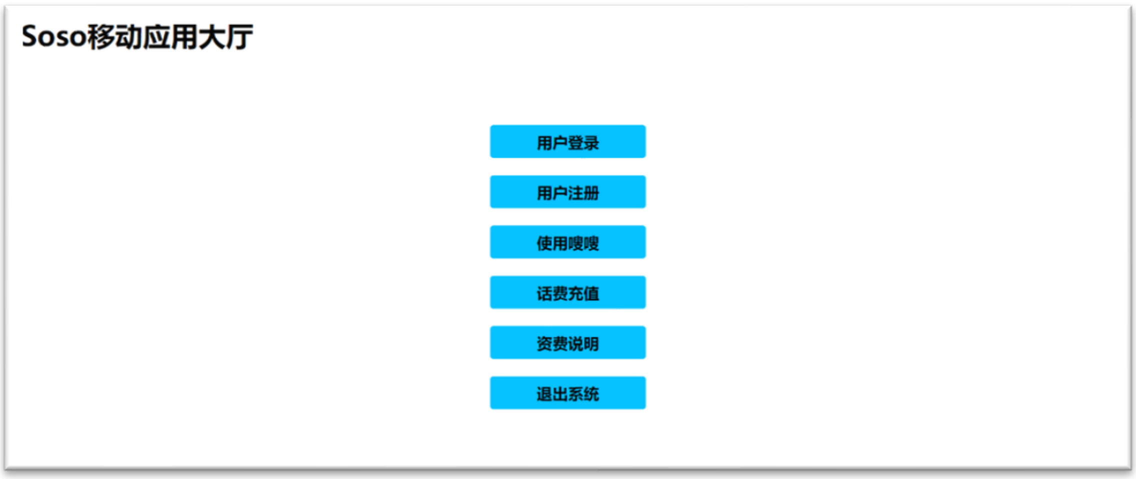
对本学期的 Java 作业 1 的 SOSO 移动大厅进行改进， 基于 SSM、JSP、Maven、Tomcat、MYSQL 等实现。

二、实现详情

1、工程结构图



```
▶ .idea
▼ src
    ▼ main
        ▼ java
            ▼ com
                ▶ controller
                ▶ dao
                ▶ pojo
                ▶ service
        ▼ resources
            ▶ mapper
            ▼ spring
                spring-dao.xml
                spring-mvc.xml
                spring-service.xml
            jdbc.properties
            log4j.properties
            mybatis-config.xml
        ▼ webapp
            ▼ WEB-INF
                ▶ jsp
                web.xml
            index.jsp
```



2、工程结构各部分实现

（1）java

pojo: 存放自定义的 java 类。每个类的属性设为 private，并提供 public 属性的 getter/setter 方法让外界访问。

```
// MobileCard.java

package com.pojo;

//手机卡类
public class MobileCard {
    private String cardNumber;          //卡号
    private String userName;            //用户名
    private String passWord;            //密码
    private String serPackage;          //服务包
    private double consumAmount;        //总消费额
    private double money;               //余额
    private int realTalkTime;           //实际通话时间
    private int realSMSCount;           //实际短信数量
    private int realFlow;               //实际流量消耗

    public MobileCard() {
        this.realTalkTime = 0;
        this.realSMSCount = 0;
        this.realFlow = 0;
```

```java
    }

    public String getcardNumber() {
        return cardNumber;
    }
    public String getuserName() {
        return userName;
    }
    public String getpassWord() {
        return passWord;
    }
    public String getserPackage() {
        return serPackage;
    }
    public double getconsumAmount() {
        return consumAmount;
    }
    public double getmoney() {
        return money;
    }
    public int getrealTalkTime() {
        return realTalkTime;
    }
    public int getrealSMSCount() {
        return realSMSCount;
    }
    public int getrealFlow() {
        return realFlow;
    }

    public void setcardNumber(String temp) {
        cardNumber = temp;
    }
    public void setuserName(String temp) {
        userName = temp;
    }
    public void setpassWord(String temp) {
        passWord = temp;
    }
    public void setserPackage(String temp) { serPackage = temp; }
    public void setconsumAmount(double temp) {
        consumAmount = temp;
    }
    public void setmoney(double temp) {
        money = temp;
    }
    public void setrealTalkTime(int temp) {
        realTalkTime = temp;
    }
    public void setrealSMSCount(int temp) {
        realSMSCount = temp;
    }
    public void setrealFlow(int temp) {
        realFlow = temp;
    }
}
```

service：定义接口，包含系统所提供的功能。此外还会在 service 包下再新建 impl 包。

```
//SosoService.java
package com.service;
import com.pojo.MobileCard;
import java.util.List;

public interface SosoService {
    int userRegister(MobileCard newUser);

    List<MobileCard> queryAllUser();
}
```

```
//SosoServiceInpl.java
package com.service.impl;
import com.dao.SosoDao;
import com.pojo.MobileCard;
import com.service.SosoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class SosoServicelmpl implements SosoService {
    @Autowired
    private SosoDao sosoDao;

    @Override
    public int userRegister(MobileCard newUser) {
        return sosoDao.userRegister(newUser);
```

```java
    }

    @Override
    public List<MobileCard> queryAllUser() {
        return sosoDao.queryAllUser();
    }
}
```

dao：定义接口，包含与数据库进行交互的功能。

```java
//SosoDao.java
package com.dao;
import com.pojo.MobileCard;
import java.util.List;

public interface SosoDao {
    int userRegister(MobileCard newUser);
    List<MobileCard> queryAllUser();
}
```

controller：控制器，负责接收页面请求，转发和处理。

```java
//SosoController.java
package com.controller;
import com.pojo.MobileCard;
import com.service.SosoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletRequest;
import java.util.List;

@Controller
@RequestMapping("/soso")
public class SosoController {
    @Autowired
    private SosoService sosoService;
    @Autowired
    HttpServletRequest request;

    @RequestMapping("/userRegisterWeb")
    public String userRegisterWeb(MobileCard newUser) {
        return "userRegister";
    }
    @RequestMapping(value = "/userRegister")
    public String userRegister() {
        MobileCard newUser = new MobileCard();
        newUser.setcardNumber(request.getParameter("cardNumber"));
        newUser.setuserName(request.getParameter("userName"));
        newUser.setpassWord(request.getParameter("passWord"));
        newUser.setserPackage(request.getParameter("serPackage"));

newUser.setconsumAmount(Double.valueOf(request.getParameter("consumAmount")
));
        newUser.setmoney(Double.valueOf(request.getParameter("money")));

newUser.setrealTalkTime(Integer.valueOf(request.getParameter("realTalkTime"
)));

newUser.setrealSMSCount(Integer.valueOf(request.getParameter("realSMSCount"
)));

newUser.setrealFlow(Integer.valueOf(request.getParameter("realFlow")));
        sosoService.userRegister(newUser);
        return "userRegister";
    }

    @RequestMapping("/allUserWeb")
    public String allUserWeb(Model model) {
        List<MobileCard> list = sosoService.queryAllUser();
        model.addAttribute("list", list);
        return "allUser";
    }
}
```

（2）resource

在resource包下有两个文件夹，“mapper”（用于存放xxxMapper.xml文件)和“spring”（用于存放 spring-xxx.xml 配置文件)。

jdbc.properties：mysql 数据库配置文件

log4j.properties：日志输出配置文件

mybatis-config.xml：mybatis 框架配置文件

```
//mapper 文件
<?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.dao.SosoDao">
    <resultMap type="MobileCard" id="sosoResultMap" >
        <id property="cardNumber" column="CardNumber"/>
        <result property="userName" column="UserName"/>
        <result property="passWord" column="PassWord"/>
        <result property="serPackage" column="SerPackage"/>
        <result property="consumAmount" column="ConsumAmount"/>
        <result property="money" column="Money"/>
        <result property="realTalkTime" column="RealTalkTime"/>
        <result property="realSMSCount" column="RealSMSCount"/>
        <result property="realFlow" column="RealFlow"/>
    </resultMap>

    <insert id="userRegister" parameterType="MobileCard">
        INSERT INTO
MobileCardMYSQL(CardNumber,UserName,PassWord,SerPackage,ConsumAmount,Money,
RealTalkTime, RealSMSCount, RealFlow) VALUE (#{cardNumber},#{userName},
#{passWord}, #{serPackage}, #{consumAmount},#{money}, #{realTalkTime},
#{realSMSCount}, #{realFlow})
    </insert>

    <select id="queryAllUser" resultMap="sosoResultMap">
        SELECT * FROM MobileCardMYSQL
    </select>
</mapper>
```

```
// jdbc.properties
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://127.0.0.1:3306/soso?useUnicode=true&characterEncoding
=utf8
jdbc.username=root
jdbc.password=root
```

```
// log4j.properties
log4j.rootLogger=ERROR, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
```

```
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n
```

```
// mybatis-config.xml
<?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>
    <!-- 配置全局属性 -->
    <settings>
        <!-- 使用 jdbc 的 getGeneratedKeys 获取数据库自增主键值 -->
        <setting name="useGeneratedKeys" value="true" />

        <!-- 使用列别名替换列名 默认:true -->
        <setting name="useColumnLabel" value="true" />

        <!-- 开启驼峰命名转换:Table{create_time} -> Entity{createTime} -->
        <setting name="mapUnderscoreToCamelCase" value="true" />
    </settings>
</configuration>
```

（3）jsp

Web 界面配置

```
// index.jsp 主界面
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" %>
<% pageContext.setAttribute("path", request.getContextPath()); %>
<!DOCTYPE HTML>
<html>
<head>
    <title>首页</title>
    <style type="text/css">
        a {
            text-align: center;
            text-decoration: none;
            color: black;
            font-size: 18px;
        }
        h3 {
            width: 180px;
            height: 38px;
            margin: 20px auto;
            text-align: center;
            line-height: 38px;
            background: deepskyblue;
            border-radius: 4px;
        }
    </style>
</head>
<body>
<div class="container">
    <div class="row clearfix">
        <div class="col-md-12 column">
            <div class="page-header">
                <h1> Soso 移动应用大厅 </h1>
            </div>
        </div>
    </div>
</div>
<br><br>
<h3> <a href="${path }/soso/allUserWeb">用户登录</a> </h3>
<h3> <a href="${path }/soso/userRegisterWeb">用户注册</a> </h3>
<h3> <a href="${path }/paper/addPaper">使用嗖嗖</a> </h3>
<h3> <a href="${path }/paper/updatePaper">话费充值</a> </h3>
<h3> <a href="${path }/paper/allPaper">资费说明</a> </h3>
<h3> <a href="${path }/paper/updatePaper">退出系统</a> </h3>
</body>
</html>
```

```
// 二级界面：用户注册（写数据库）
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>
<html>
<head>
    <title>Soso 用户注册（写数据库）</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- 引入 Bootstrap -->
    <link
href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
    <div class="row clearfix">
        <div class="col-md-12 column">
            <div class="page-header">
                <h1>Soso 用户注册</h1>
            </div>
        </div>
    </div>

    <form action="">
        手机号：<input type="text" name="cardNumber"><br>
        用户昵称：<input type="text" name="userName"><br>
        密码：<input type="text" name="passWord"><br>
        包类型：<input type="text" name="serPackage"><br>
        消费金额：<input type="text" name="consumAmount"><br>
        余额：<input type="text" name="money"><br>
        实际通话时间：<input type="text" name="realTalkTime"><br>
        实际短信条数：<input type="text" name="realSMSCount"><br>
        实际消费流量：<input type="text" name="realFlow"><br>
        <input type="button" value="添加" onclick="usrRegister()">
        <input type="button" value="返回" onclick="back()">
    </form>

    <script type="text/javascript">
        function usrRegister() {
            var form = document.forms[0];
            form.action = "<%=basePath %>soso/userRegister";
            form.method = "get";
            form.submit();
        }
        function back() {
            var form = document.forms[0];
            form.action = "<%=basePath %>";
            form.submit();
        }
    </script>
</div>
```

```
//二级界面：展示用户列表信息（读数据库）
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%  String appPath = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + appPath + "/";
%>
<head>
    <title>Soso 用户列表（读数据库）</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <link
href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet"/>
</head>-->

<body xmlns:c="http://java.sun.com/jsp/jstl/core">
<div class="container">

    <div class="row">
        <div class="col-md-4 column">
            <a class="btn btn-primary" href="<%=basePath %>">返回</a>
        </div>
    </div>
    <div class="row clearfix">
        <div class="col-md-12 column">
            <table class="table table-hover table-striped">
                <thead>
                <tr>
                    <th>手机号</th>
                    <th>用户名</th>
                    <th>密码</th>
                    <th>服务包</th>
                    <th>消费金额</th>
                    <th>余额</th>
                    <th>实际通话</th>
                    <th>实际短信</th>
                    <th>实际流量</th>
                </tr>
                </thead>
                <tbody>
                <c:forEach var="user" items="${requestScope.get('list')}"
varStatus="status">
                    <tr>
```

```
                        <td>${user.cardNumber}</td>
                        <td>${user.userName}</td>
                        <td>${user.passWord}</td>
                        <td>${user.serPackage}</td>
                        <td>${user.consumAmount}</td>
                        <td>${user.money}</td>
                        <td>${user.realTalkTime}</td>
                        <td>${user.realSMSCount}</td>
                        <td>${user.realFlow}</td>
<%--                        <td>111</td>--%>
<%--                        <td>222</td>--%>
<%--                        <td>3</td>--%>
<%--                        <td>4</td>--%>
<%--                        <td>5</td>--%>
<%--                        <td>6</td>--%>
<%--                        <td>7</td>--%>
<%--                        <td>8</td>--%>
<%--                        <td>9</td>--%>
                    </tr>
                </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
</body>
```

（4）pom.xml

在此文件中配置项目所需要的 jar 包。

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com</groupId>
  <artifactId>first</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
```

```xml
  <name>first Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
    <spring.version>5.0.3.RELEASE</spring.version>
    <mybatis.version>3.4.4</mybatis.version>
  </properties>
  <dependencies>
    <!-- 单元测试 -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <!-- 第一部分: Spring 配置-->
    <!-- Spring core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context-support</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!-- Spring DAO -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-tx</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!-- Spring mvc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-test</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!-- 第二部分：Servlet web -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.0.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.2</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>
    <dependency>
      <groupId>taglibs</groupId>
      <artifactId>standard</artifactId>
      <version>1.1.2</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.9.4</version>
    </dependency>
    <!-- 第三部分：数据库和 mybatis -->
    <!-- 数据库 -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.38</version>
    </dependency>
    <!-- 数据库连接池 -->
    <dependency>
      <groupId>com.mchange</groupId>
      <artifactId>c3p0</artifactId>
      <version>0.9.5.2</version>
    </dependency>
    <!-- MyBatis -->
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <version>${mybatis.version}</version>
    </dependency>
    <!-- mybatis-spring 整合包 -->
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis-spring</artifactId>
      <version>1.3.1</version>
    </dependency>
    <!-- 第四部分：日志 -->
    <!-- 实现 slf4j 接口并整合 -->
    <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
      <version>1.1.1</version>
    </dependency>
  </dependencies>
  <build>
    <finalName>first</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
```

```xml
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

（5）spring

spring-dao.xml：（spring-mybatis 整合配置文件）

```xml

<?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
    http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- 配置整合 mybatis 过程 -->
    <!-- 1.配置数据库相关参数 properties 的属性：${url} -->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!-- 2.数据库连接池 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <!-- 配置连接池属性 -->
        <property name="driverClass" value="${jdbc.driver}"/>
        <property name="jdbcUrl" value="${jdbc.url}"/>
        <property name="user" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>

        <!-- c3p0 连接池的私有属性 -->
        <property name="maxPoolSize" value="30"/>
        <property name="minPoolSize" value="10"/>
        <!-- 关闭连接后不自动 commit -->
        <property name="autoCommitOnClose" value="false"/>
        <!-- 获取连接超时时间 -->
        <property name="checkoutTimeout" value="10000"/>
        <!-- 当获取连接失败重试次数 -->
        <property name="acquireRetryAttempts" value="2"/>
    </bean>

    <!-- 3.配置 SqlSessionFactory 对象 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 注入数据库连接池 -->
        <property name="dataSource" ref="dataSource"/>
        <!-- 配置 MyBaties 全局配置文件:mybatis-config.xml -->
        <property name="configLocation" value="classpath:mybatis-
config.xml"/>
        <!-- 扫描 pojo 包 使用别名 -->
        <property name="typeAliasesPackage" value="com.pojo"/>
        <!-- 扫描 sql 配置文件:mapper 需要的 xml 文件 -->
        <property name="mapperLocations" value="classpath:mapper/*.xml"/>
    </bean>

    <!-- 4.配置扫描 Dao 接口包，动态实现 Dao 接口，注入到 spring 容器中 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 注入 sqlSessionFactory -->
        <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory"/>
        <!-- 给出需要扫描 Dao 接口包 -->
        <property name="basePackage" value="com.dao"/>
    </bean>
</beans>
```

Spring-mvc

```xml

<?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.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">
    <!-- 配置 SpringMVC -->
    <!-- 1.开启 SpringMVC 注解模式 -->
    <!-- 简化配置:
        (1)自动注册
```

```
DefaultAnootationHandlerMapping,AnotationMethodHandlerAdapter
        (2)提供一些列：数据绑定，数字和日期的 format @NumberFormat,
@DateTimeFormat, xml,json 默认读写支持
    -->
    <mvc:annotation-driven />

    <!-- 2.静态资源默认 servlet 配置
        (1)加入对静态资源的处理：js,gif,png
        (2)允许使用"/"做整体映射
     -->
    <mvc:default-servlet-handler/>

    <!-- 3.配置 jsp 显示 ViewResolver -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView" />
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <!-- 4.扫描 web 相关的 bean -->
    <context:component-scan base-package="com.controller" />
</beans>
```

Spring-service

```

<?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: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://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">
    <!-- 扫描 service 包下所有使用注解的类型 -->
    <context:component-scan base-package="com.service" />

    <!-- 配置事务管理器 -->
    <bean id="transactionManager"

```

```
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <!-- 注入数据库连接池 -->
        <property name="dataSource" ref="dataSource" />
    </bean>
    <!-- 配置基于注解的声明式事务 -->
    <tx:annotation-driven transaction-manager="transactionManager" />
</beans>
```

三、参考资料

https://blog.csdn.net/khxu666/article/details/79851070