Java 实验报告

徐云凯 1713667

项目部署地址：https://www.milkyship.cn/sosoMobile

目录

问题分析描述

项目基于嗖嗖移动业务大厅 2.0 改进而来。要求使用 ssm 框架实现，并使用ＭＶＣ模型实现。

实验环境

开发端
Java 版本：java version "1.8.0_121" Java(TM) SE Runtime Environment (build 1.8.0_121-b13) Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)。Tomcat 版本：7.0.92. Maven 版本：3.6.1.　IDE: IntelliJ IDEA 2019

服务器端
服务器操作系统：Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-54-generic x86_64)。 Java 版本：java version "1.8.0_221" Java(TM) SE Runtime Environment (build 1.8.0_221-b11) Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)。 数据库：腾讯云服务器，mysql Ver 14.14 Distrib 5.7.27, for Linux (x86_64)。 数据库连接池：c3p0 0.9.5.2。 Mybatis 版本：3.4.1。 Spring 版本：4.3.5.RELEASE

实现思路

项目中数据流主要涉及电话卡信息、通话记录与套餐信息，故数据库中设置三个表，本别存储。

前端部分使用了 ajax 实时返回轻量级查询业务的结果，只在较复杂的业务时才跳转到新的页面。

后台实现中，将全部逻辑切分到 controller, service 与 dao 层中。

Controller 只负责识别前端传回的 http 请求，调用 service 中的相关方法。

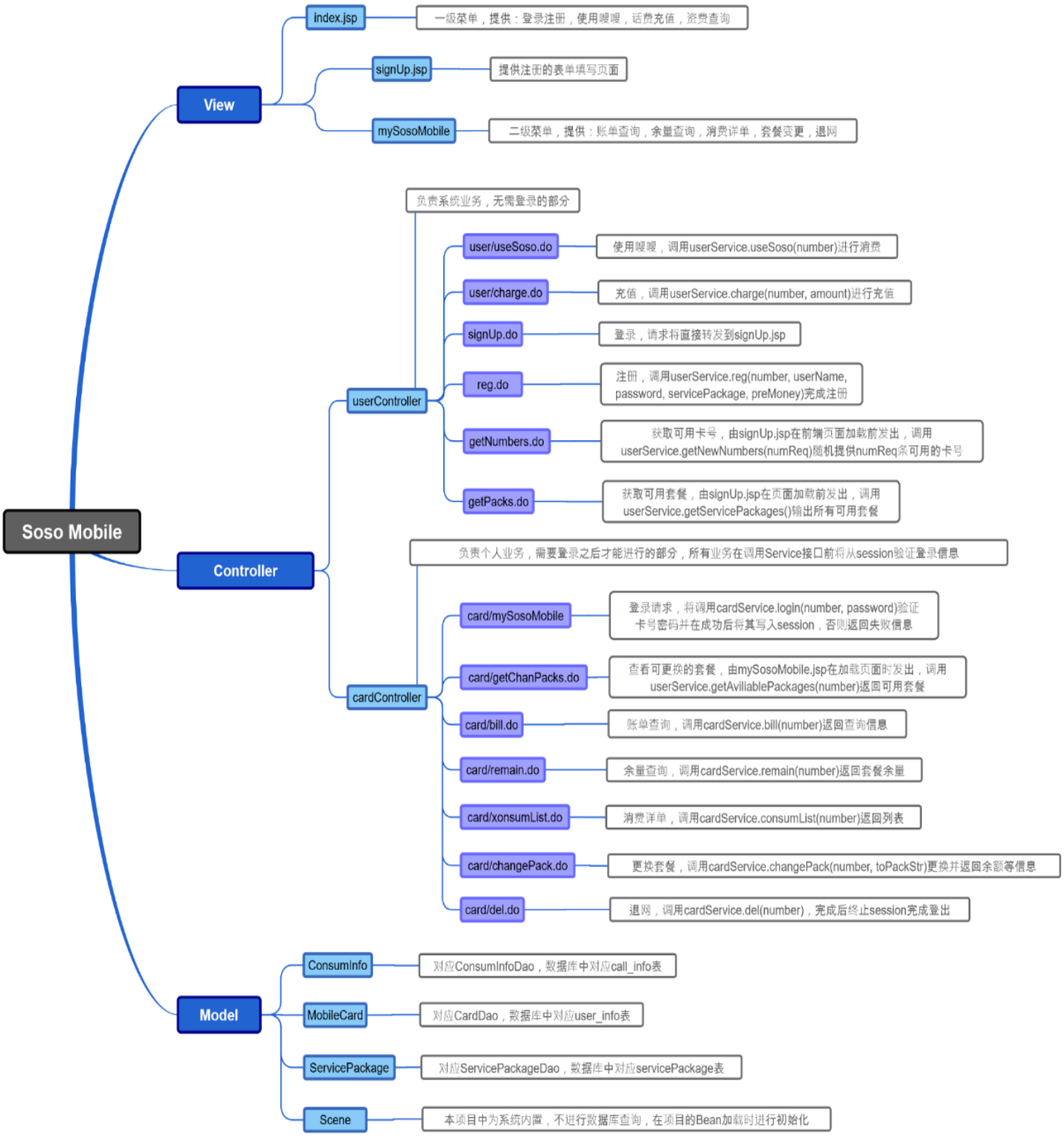
Service 负责对接 dao，提供专门的业务功能。

Dao 使用 mybatis，实现对数据库灵活快速的访问。

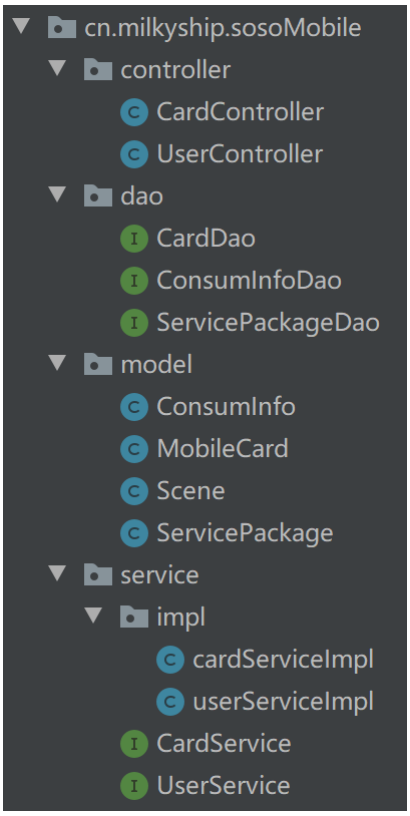
服务器端部署时，使用 nginx 反向代理 8080 接口，使用 apache 服务器访问静态文件如 css, html 等，使用 tomcat 实现 jsp 等动态网页访问。动静分离提高相应速度。

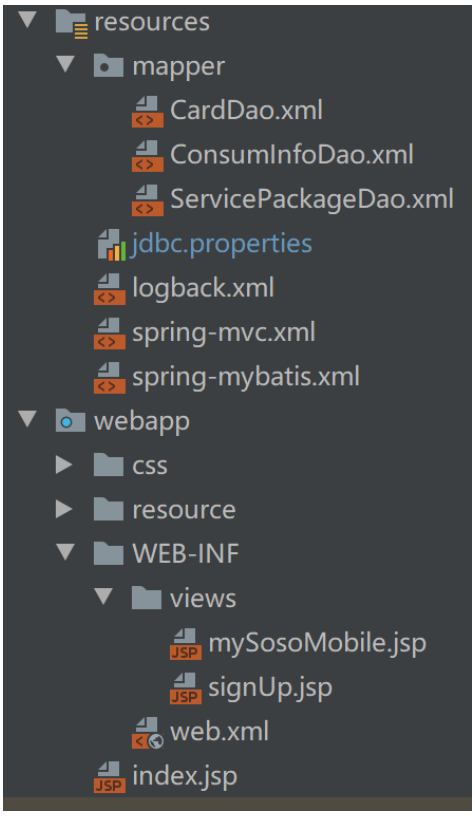
具体实现

一. 整体结构


Soso Mobile
View
index.jsp
一级菜单，提供：登录注册，使用嘤嘤，话费充值，资费查询
signUp.jsp
提供注册的表单填写页面
mySosoMobile
二级菜单，提供：账单查询，余量查询，消费详单，套餐变更，退网
Controller
userController
负责系统业务，无需登录的部分
user/useSoso.do
使用嘤嘤，调用userService.useSoso(number)进行消费
user/charge.do
充值，调用userService.charge(number, amount)进行充值
signUp.do
登录，请求将直接转发到signUp.jsp
reg.do
注册，调用userService.reg(number, userName, password, servicePackage, preMoney)完成注册
getNumbers.do
获取可用卡号，由signUp.jsp在前端页面加载前发出，调用userService.getNewNumbers(numReq)随机提供numReq条可用的卡号
getPacks.do
获取可用套餐，由signUp.jsp在页面加载前发出，调用userService.getServicePackages()输出所有可用套餐
cardController
负责个人业务，需要登录之后才能进行的部分，所有业务在调用Service接口前将从session验证登录信息
card/mySosoMobile
登录请求，将调用cardService.login(number, password)验证卡号密码并在成功后将其写入session，否则返回失败信息
card/getChanPacks.do
查看可更换的套餐，由mySosoMobile.jsp在加载页面时发出，调用userService.getAviliablePackages(number)返回可用套餐
card/bill.do
账单查询，调用cardService.bill(number)返回查询信息
card/remain.do
余量查询，调用cardService.remain(number)返回套餐余量
card/xonsumList.do
消费详单，调用cardService.consumList(number)返回列表
card/changePack.do
更换套餐，调用cardService.changePack(number, toPackStr)更换并返回余额等信息
card/del.do
退网，调用cardService.del(number)，完成后终止session完成登出
Model
ConsumInfo
对应ConsumInfoDao，数据库中对应call_info表
MobileCard
对应CardDao，数据库中对应user_info表
ServicePackage
对应ServicePackageDao，数据库中对应servicePackage表
Scene
本项目中为系统内置，不进行数据库查询，在项目的Bean加载时进行初始化

故依据以上要求，项目各个类与接口结构如下：





二. 前端页面

1. Index.jsp

1.1. Ajax 请求

```
function userSoso() {
    var number = document.getElementById("userUseSoso").value;
    if (number < 13900000000 || number > 17822011172)
        return false;
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState === 4 && xmlhttp.status === 200) {
            var infoLen = xmlhttp.responseText.length;
            document.getElementById("userSosoOutput").innerHTML = xmlhttp.responseText.substring(1, infoLen - 1);
        }
    }
    xmlhttp.open("POST", "user/useSoso.do", true);
    xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xmlhttp.send("number=" + number);
}
```

所有的轻量级通信均使用 ajax 实时返回信息。

1.2. 外部 css 样式表

```
<link rel="stylesheet" type="text/css" href="${pageContext.request.contextPath }/css/sosomobile.css">
```

1.3. form 表单

```
<div class="login">
    <form id="login" class="form" action="card/mySosoMobile" method="post">
        <label for="number">用户名： </label><input id="number" name="number" type="number" min="13900000000"
                                                max="17822011172" step="1">
        <label for="number">密码：</label><input id="password" name="password" type="password">
        <input class="submit" type="submit" value="登录">
        <output>${info}</output>
        <input class="submit" type="submit" formaction="user/signUp" value="没有账号？去注册">
    </form>
</div>
```

form 不会直接向 jsp 文件提交 http 请求，所有请求均会被 spring MVC 拦截并分类处理。任何涉及密码的请求一定使用 post 提交

对于 index.jsp，可能需要处理上一次登录失败的信息，使用 spring MVC 的 model 进行传值，将登陆失败时写入 model 对象的失败信息进行输出。

1.4. 文件链接跳转

```
<div class="fun">
    <div class="form">
        <a href="${pageContext.request.contextPath }/resource/FeesDetails.pdf">查看资费详情</a>
    </div>
</div>
```

使用${pageContext.request.contextPath }获取当前路径，进行链接跳转。

2. signUp.jsp

2.1. 加载页面时获取随机的可用卡号与全部可用套餐

```
<script>
    var xmlhttp0 = new XMLHttpRequest();
    xmlhttp0.onreadystatechange = function func0() {
        if (xmlhttp0.readyState === 4 && xmlhttp0.status === 200) {
            var textSize = xmlhttp0.responseText.length - 2;
            document.getElementById("numbers").innerHTML = xmlhttp0.responseText.substr(1, textSize);
        }
    }
    xmlhttp0.open("POST", "getNumbers.do", true);
    xmlhttp0.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xmlhttp0.send("num=9");
    var xmlhttp1 = new XMLHttpRequest();
    xmlhttp1.onreadystatechange = function func1() {
        if (xmlhttp1.readyState === 4 && xmlhttp1.status === 200) {
            var textSize = xmlhttp1.responseText.length - 2;
            document.getElementById("packages").innerHTML = xmlhttp1.responseText.substr(1, textSize);
        }
    }
    xmlhttp1.open("POST", "getPacks.do", true);
    xmlhttp1.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xmlhttp1.send();
</script>
```

2.2. JavaScript 检查输入是否合法并提交

```
<script>
    function reg() {
        var number = document.getElementById("number").value;
        if (number < 13900000000 || number > 17822011172) {
            document.getElementById("info").innerHTML = "卡号格式不合法";
            return false;
        }
        var userName = document.getElementById("userName").value;
        var pwd = document.getElementById("password").value;
        var rePwd = document.getElementById("rePassword").value;
        if (pwd !== rePwd) {
            document.getElementById("info").innerHTML = "两次输入密码不一致";
            return false;
        }
        var pack = document.getElementById("package").value;
        var money = document.getElementById("money").value;

        var xmlhttp = new XMLHttpRequest();

        xmlhttp.onreadystatechange = function () {
            if (xmlhttp.readyState === 4 && xmlhttp.status === 200) {
                var infoLen = xmlhttp.responseText.length;
                document.getElementById("info").innerHTML = xmlhttp.responseText.substring(1, infoLen - 1);
            }
        }
        xmlhttp.open("POST", "reg.do", true);
        xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xmlhttp.send("info=" + number + ";" + userName + ";" + pwd + ";" + pack + ";" + money);
    }
</script>
```

3. mySosoMobile.jsp

3.1. 页面加载时加载变更套餐功能需要用到的可用套餐

```
<script>
    var xmlhttp1 = new XMLHttpRequest();
    xmlhttp1.onreadystatechange = function func1() {
        if (xmlhttp1.readyState === 4 && xmlhttp1.status === 200) {
            var textSize = xmlhttp1.responseText.length - 2;
            document.getElementById("packages").innerHTML = xmlhttp1.responseText.substr(1, textSize);
        }
    }
    xmlhttp1.open("POST", "getChanPacks.do", true);
    xmlhttp1.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xmlhttp1.send("number=" + document.getElementById("Number").innerHTML);
</script>
```

3.2. Ajax 提交

```
function bill() {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState === 4 && xmlhttp.status === 200) {
            var infoLen = xmlhttp.responseText.length;
            document.getElementById("billOutput").innerHTML = xmlhttp.responseText.substring(1, infoLen - 1);
        }
    }
    xmlhttp.open("POST", "bill.do", true);
    xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xmlhttp.send();
}
```

三． 后台-Controller

1. UserController

使用 SpringMVC 注解模式。

1.0. 类头与静态资源

```
@Controller
@RequestMapping ("/user")
public class UserController {

    @Resource
    private UserService userService;
```

1.1. useSoso 使用嗖嗖

```
@RequestMapping ("/useSoso.do")
public void useSoso(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String number = request.getParameter( name: "number");
    String info = userService.useSoso(number);
    ObjectMapper mapper = new ObjectMapper();
    response.getWriter().write(mapper.writeValueAsString(info));
    response.getWriter().close();
}
```

1.2. charge 充值

```
@RequestMapping ("/charge.do")
public void charge(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String parameter = request.getParameter( name: "infos");
    String[] parameters = parameter.split( regex: ";");
    double amount = Double.parseDouble(parameters[1]);
    String info = userService.charge(parameters[0], amount);
    ObjectMapper mapper = new ObjectMapper();
    response.getWriter().write(mapper.writeValueAsString(info));
    response.getWriter().close();
}
```

1.3. signUp 登录，直接返回 signUp.jsp

```
@RequestMapping ("/signUp")
public String jumpToRegPage(HttpServletRequest request, HttpServletResponse response) throws IOException {
    return "/signUp";
}
```

1.4. reg 注册

```
@RequestMapping ("/reg.do")
public void reg(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String parameter = request.getParameter( name: "info");
    String[] parameters = parameter.split( regex: ";");
    String number = parameters[0];
    String userName = parameters[1];
    String password = parameters[2];
    String servicePackage = parameters[3];
    Double preMoney = Double.parseDouble(parameters[4]);
    String info = userService.reg(number, userName, password, servicePackage, preMoney);
    ObjectMapper mapper = new ObjectMapper();
    response.getWriter().write(mapper.writeValueAsString(info));
    response.getWriter().close();
}
```

1.5. getNewNumbers 获取随机的多个可用号码

```
@RequestMapping ("/getNumbers.do")
public void getNewNumbers(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    int numReq = Integer.parseInt(request.getParameter( name: "num"));
    List<String> nums = this.userService.getNewNumbers(numReq);
    StringBuilder output = new StringBuilder();
    for (String i : nums)
        output.append("<option value='").append(i).append("'></option>");
    ObjectMapper mapper = new ObjectMapper();
    response.getWriter().write(mapper.writeValueAsString(output));
    response.getWriter().close();
}
```

1.6. getServicePackages 获取可用的全部套餐

```
@RequestMapping ("/getPacks.do")
public void getServicePackages(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    List<ServicePackage> packs = this.userService.getServicePackages();
    StringBuilder output = new StringBuilder();
    for (ServicePackage i : packs)
        output.append("<option value='").append(i.getName()).append("'></option>");
    ObjectMapper mapper = new ObjectMapper();
    response.getWriter().write(mapper.writeValueAsString(output));
    response.getWriter().close();
}
```

2. CardController

1.1. 类头与静态资源

```
@Controller
@RequestMapping ("/card")
public class CardController {
```

```
    @Resource
    private CardService cardService;

    @Resource
    private UserService userService;
```

1.2. 登录请求，检查密码并向 session 写入登录信息，登录失败时使用 spring MVC model 对象将错误信息返回至下一个页面。

```
@RequestMapping ("/mySosoMobile")
public String logIn(HttpServletRequest request, HttpServletResponse response, Model model) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String number = request.getParameter( name: "number");
    String password = request.getParameter( name: "password");
    ObjectMapper mapper = new ObjectMapper();
    int returnParam = cardService.login(number, password);
    if (returnParam != 0) {
        if (returnParam == 1) {
            model.addAttribute( s: "info", o: "账号不存在");
        }
        else {
            model.addAttribute( s: "info", o: "密码错误");
        }
        return "/../../index";
    }
    request.getSession().setAttribute( name: "number", number);
    request.getSession().setAttribute( name: "password", password);
    return "/mySosoMobile";
}
```

1.3. del 退网，执行完成后终止 session

```
@RequestMapping ("/del.do")
public void del(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String number = request.getSession().getAttribute( name: "number").toString();
    cardService.del(number);
    request.getSession().invalidate();
}
```

1.4. 菜单功能

以更换套餐功能为例。

```
@RequestMapping ("/changePack.do")
public void changePack(HttpServletRequest request, HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    String number = request.getSession().getAttribute( name: "number").toString();
    String toPackStr = request.getParameter( name: "pack");
    String info = cardService.changePack(number, toPackStr);
    ObjectMapper mapper = new ObjectMapper();
    response.getWriter().write(mapper.writeValueAsString(info));
    response.getWriter().close();
}
```

四. 后台-Service

1. UserService

1.1. 提供接口如下：

```
public String useSoso(String number);
public String charge(String number, double money);
public String reg(String number, String userName, String password, String servicePackage, Double preMoney);
public List<String> getNewNumbers(int numberCount);
public List<ServicePackage> getServicePackages();
public List<ServicePackage> getAviliablePackages(String number);
```

1.2. 实现：UserServiceImpl 类

略。

2. CardService

2.1. 提供接口如下：

```
public int login(String number, String password);
public String bill(String number);
public String remain(String number);
public String consumList(String number);
public String changePack(String number, String toPack);
public void del(String number);
```

2.2. 实现：CardServiceImpl 类

使用 SpringMVC 注解模式

2.2.1. 类头与静态资源

```
@Service ("CardService")
public class cardServiceImpl implements CardService {

    @Resource
    private CardDao cardDao;

    @Resource
    private ServicePackageDao servicePackageDao;

    @Resource
    private ConsumInfoDao consumInfoDao;
```

2.2.2. login 登录

成功时返回 0，卡号不正确时返回 1，密码不正确时返回 2

```
@Override
public int login(String number, String password) {
    if (cardDao.findCardByNumber(number) == null) {
        return 1;
    }
    if (cardDao.checkCard(number, password) == null) {
        return 2;
    }
    return 0;
}
```

2.2.3. 账单查询

```
@Override
public String bill(String number) {
    MobileCard mobileCard = cardDao.findCardByNumber(number);
    ServicePackage servicePackage = servicePackageDao.findPackage(mobileCard.getSerPackage());
    StringBuilder out = new StringBuilder();
    DecimalFormat formatData = new DecimalFormat( pattern: "#0.00");
    out.append("<p>您的卡号：").append(number)
            .append("</p><p>当月账单：").append(formatData.format(mobileCard.getConsumAmount()))
            .append("</p><p>套餐资费：").append(formatData.format(servicePackage.getPrice())).append("元</p>")
            .append("<p>合计：").append(formatData.format( number: mobileCard.getConsumAmount() + servicePackage.getPrice())).append("元</p>")
            .append("<p>账户余额：").append(formatData.format(mobileCard.getMoney())).append("元</p>");
    return out.toString();
}
```

2.2.4. 余量查询

```
@Override
public String remain(String number) {
    MobileCard mobileCard = cardDao.findCardByNumber(number);
    ServicePackage servicePackage = servicePackageDao.findPackage(mobileCard.getSerPackage());
    StringBuilder out = new StringBuilder();
    DecimalFormat formatData = new DecimalFormat( pattern: "#0.00");
    out.append("<p>您的卡号是").append(number)
            .append("</p><p>套餐内剩余：</p>")
            .append("<p>通话时长：").append(Math.max(servicePackage.getTalkTime() - mobileCard.getRealTalkTime(), 0)).append("分钟</p>")
            .append("<p>短信条数：").append(Math.max(servicePackage.getSmsCount() - mobileCard.getRealSMSCount(), 0)).append("条</p>")
            .append("<p>上网流量：").append(formatData.format(Math.max(servicePackage.getFlow() - mobileCard.getRealFlow(), 0))).append("MB</p>
    return out.toString();
}
```

2.2.5. 消费详单

```
    @Override
    public String consumList(String number) {
//...
        StringBuilder content = new StringBuilder();
        content.append("<table border='1'><caption>").append("消费记录</caption>")
                .append("<thead><tr><th>序号</th><th>类型</th><th>数据[通话(分钟)/上网(MB)/短信(条)]</th></tr></thead>");
        ConsumInfo[] list = consumInfoDao.findRecByNumber(number);
        int ind = 1;
        content.append("<tbody>");
        for (ConsumInfo i : list) {
            if (i != null) {
                content.append("<tr><td>")
                        .append(ind++).append("</td><td>")
                        .append(i.getRecType()).append("</td><td>")
                        .append(i.consumData).append("</td></tr>");
            }
        }
        content.append("</tbody>");
        return content.toString();
    }
```

2.2.6. 更换套餐

```
@Override
public String changePack(String number, String toPack) {
    MobileCard mobileCard = cardDao.findCardByNumber(number);
    if (mobileCard.getSerPackage().equals(toPack)) {
        return "转出套餐与当前套餐相同";
    }
    ServicePackage servicePackage = servicePackageDao.findPackage(toPack);
    if (mobileCard.getMoney() < servicePackage.getPrice()) {
        return "余额不足以支付新套餐首月资费";
    }
    mobileCard.setSerPackage(toPack);
    cardDao.update(mobileCard);
```

```
    StringBuilder info = new StringBuilder();
    info.append("套餐转换成功。");
    DecimalFormat formatData = new DecimalFormat( pattern: "#.0");
    info.append(" 当前余额：").append(formatData.format(mobileCard.getMoney())).append("元。")
            .append("当前套餐：").append(mobileCard.getSerPackage())
            .append(String.format(",通话时长为%d分钟/月，短信条数为%d条/月，上网流量为%dMB/月",
                    servicePackage.getTalkTime(), servicePackage.getSmsCount(), servicePackage.getFlow()));
    return info.toString();
}
```

2.2.7. 退网

```
@Override
public void del(String number) {
    cardDao.del(number);
}
```

五. 后台-model

1. ConsumInfo

```
public class ConsumInfo {

    private String cardNumber;
    private String recType;
    public int consumData;
```

2. MobileCard

```
public class MobileCard {

    private String cardNumber;
    private String userName;
    private String password;
    private String serPackage;
    private double consumAmount = 0;
    private double money;
    private int realTalkTime = 0;
    private int realSMSCount = 0;
    private int realFlow = 0;
```

3. ServicePackage

```
public class ServicePackage {

    private String name = null;

    private int talkTime = 0;
    private int smsCount = 0;
    private int flow = 0;
    private double price = 0;
```

4. Scene

```
public class Scene {

    public String type;
    public int data;
    public String description;
```

六. 后台-dao

1. ConsumInfoDao

```
public interface ConsumInfoDao {
    /*** ...*/
    void save(ConsumInfo consumInfo);

    /*** ...*/
    void del(ConsumInfo consumInfo);

    /*** ...*/
    void update(ConsumInfo consumInfo);

    /*** ...*/
    ConsumInfo[] findRecByNumber(String Number);

}
```

2. CardDao

```java
public interface CardDao {
    /** Title: save ...*/
    void save(MobileCard mobileCard);

    /*** ...*/
    void del(String number);

    /*** ...*/
    void update(MobileCard mobileCard);

    /*** ...*/
    MobileCard findCardByNumber(String Number);

    /*** ...*/
    MobileCard checkCard(String Number, String password);
}
```

3. ServicePackageDao

```java
public interface ServicePackageDao {
    /*** ...*/
    ServicePackage findPackage(String name);

    /*** ...*/

    ServicePackage[] findAllPackage();
}
```

七. ssm 配置文件

1. Spring-Mybatis 与 DAO

1.1. Spring-Mybatis.xml

扫描 service 包

```xml
<!-- 扫描service包下所有使用注解的类型 -->
<context:component-scan base-package="cn.milkyship.service"/>
```

读取 jdbc.properties.xml

```xml
<!-- 配置数据库相关参数properties的属性: ${url} -->
<context:property-placeholder location="classpath:jdbc.properties"/>
```

将 jdbc.properties.xml 中的配置信息设置到 c3p0 中

```
<!-- 数据库连接池 -->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="${jdbc.driver}"/>
    <property name="jdbcUrl" value="${jdbc.url}"/>
    <property name="user" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>
    <property name="maxPoolSize" value="${c3p0.maxPoolSize}"/>
    <property name="minPoolSize" value="${c3p0.minPoolSize}"/>
    <property name="autoCommitOnClose" value="${c3p0.autoCommitOnClose}"/>
    <property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"/>
    <property name="acquireRetryAttempts" value="${c3p0.acquireRetryAttempts}"/>
</bean>
```

配置 sqlSessionFactory，注入连接池 bean，扫描 model 实体类，扫描 dao 配置文件。

```
<!-- 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"/>
    <!-- 扫描model包 使用别名 -->
    <property name="typeAliasesPackage" value="cn.milkyship.sosoMobile.model"/>
    <!-- 扫描sql配置文件:mapper需要的xml文件 -->
    <property name="mapperLocations" value="classpath:mapper/*.xml"/>
</bean>
```

配置事务管理器

```
<!-- 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>
```

启用注解模式

```
<!-- 配置基于注解的声明式事务 -->
<tx:annotation-driven transaction-manager="transactionManager"/>
```

1.2. CardDao.xml

```
<!-- 设置为cardDao接口方法提供sql语句配置 -->
<mapper namespace="cn.milkyship.sosoMobile.dao.CardDao">

    <insert id="save" parameterType="MobileCard">
        INSERT INTO user_info( cardNumber, userName, password, serPackage, money, consumAmount, realTalkTime, realSMSCount, realFlow)
            VALUES (#{cardNumber}, #{userName}, #{password}, #{serPackage}, #{money}, 0, 0, 0, 0)
    </insert>

    <delete id="del" parameterType="String">
        DELETE FROM user_info WHERE cardNumber = #{number}
    </delete>

    <update id="update" parameterType="MobileCard">
        UPDATE user_info SET
                cardNumber = #{cardNumber},
                userName = #{userName},
                password = #{password},
```

```
            serPackage = #{serPackage},
            money = #{money},
            consumAmount = #{consumAmount},
            realTalkTime = #{realTalkTime},
            realSMSCount = #{realSMSCount},
            realFlow = #{realFlow}
            WHERE cardNumber = #{cardNumber}
    </update>

    <select id="findCardByNumber" parameterType="String" resultType="MobileCard">
        SELECT * FROM user_info WHERE cardNumber = #{number}
    </select>

    <select id="checkCard" resultType="MobileCard">
        SELECT * FROM user_info WHERE cardNumber = #{arg0} and password = #{arg1}
    </select>
</mapper>
```

1.3. ConsumInfo.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">

<!-- 设置为ConsumInfoDao接口方法提供sql语句配置 -->
<mapper namespace="cn.milkyship.sosoMobile.dao.ConsumInfoDao">

    <insert id="save" parameterType="ConsumInfo">
        INSERT INTO call_record( cardNumber, recType, consumData)
            VALUES (#{cardNumber}, #{recType}, #{consumData})
    </insert>

    <delete id="del" parameterType="ConsumInfo">
        DELETE FROM call_record WHERE cardNumber = #{cardNumber}, recType = #{recType}, consumData = #{consumData}
    </delete>

    <update id="update" parameterType="ConsumInfo">
        UPDATE call_record SET
                cardNumber = #{cardNumber},
                recType = #{recType},
                consumData = #{consumData}
                WHERE cardNumber = #{cardNumber}
    </update>

    <select id="findRecByNumber" resultType="consumInfo" parameterType="String">
        SELECT cardNumber, recType, consumData FROM call_record WHERE cardNumber = #{Number}
    </select>

</mapper>
```

1.4. ServicePackage.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">

<!-- 设置为cardDao接口方法提供sql语句配置 -->
<mapper namespace="cn.milkyship.sosoMobile.dao.ServicePackageDao">

    <select id="findPackage" parameterType="String" resultType="ServicePackage">
        SELECT * FROM servicePackage WHERE name = #{name}
    </select>

    <select id="findAllPackage" resultType="ServicePackage">
        SELECT * FROM servicePackage
    </select>

</mapper>
```

2. 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">

    <!-- 扫描web相关的bean -->
    <context:component-scan base-package="cn.milkyship.sosoMobile.controller"/>
    <context:component-scan base-package="cn.milkyship.sosoMobile.service.impl"/>

    <!-- 开启SpringMVC注解模式 -->
    <mvc:annotation-driven/>

    <!-- 静态资源默认servlet配置 -->
    <mvc:default-servlet-handler/>

    <!-- 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/views"/>
        <property name="suffix" value=".jsp"/>
    </bean>

</beans>
```

3. Log, Maven 与 web.xml

3.1. logback.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="true">
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>
    <root level="debug">
        <appender-ref ref="STDOUT"/>
    </root>
</configuration>
```

3.2. web.xml

网站信息与首页

```
<?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_3_1.xsd"
         version="3.1">
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <display-name>SosoMobile</display-name>
    <description>SosoMobile_Alpha_0.0.1</description>
```

编码

```
<!-- 编码过滤器 -->
<filter>
    <filter-name>encodingFilter</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>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

配置 servlet 到 SpringMVC 并扫描 xml

```
<!-- 配置DispatcherServlet -->
<servlet>
    <servlet-name>SpringMVC</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 配置springMVC需要加载的配置文件-->
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring-*.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
    <async-supported>true</async-supported>
</servlet>

<servlet-mapping>
    <servlet-name>SpringMVC</servlet-name>
    <!-- 匹配所有请求，此处也可以配置成 *.do 形式 -->
    <!-- *.do是对所有以.do结尾的请求做处理，/*是对所有请求做处理-->
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

设置 session10 分钟超时

```
<session-config>
    <session-timeout>10</session-timeout>
</session-config>
```

八. 数据库

使用 mysql 5.7.27 数据库名称 sosoMobile，包含 3 个表 call_record, servicePackage, user_info。

1. call_record

名	类型	长度	小数点	不是 null	虚拟	键
id	int	10	0	☑	☐	🔑1
cardNumber	varchar	12	0	☑	☐	🔑2
recType	varchar	63	0	☑	☐	
consumData	int	16	0	☐	☐	

2. servicePackage

名	类型	长度	小数点	不是 null	虚拟	键
id	int	10	0	☑	☐	🔑1
name	varchar	255	0	☑	☐	🔑2
price	double	10	2	☐	☐	
talkTime	int	11	0	☐	☐	
smsCount	int	11	0	☐	☐	
flow	int	11	0	☐	☐	

3. user_info

名	类型	长度	小数点	不是 null	虚拟	键
id	int	8	0	☑	☐	🔑1
cardNumber	varchar	12	0	☑	☐	🔑2
userName	varchar	255	0	☑	☐	🔑3
password	varchar	255	0	☐	☐	
serPackage	varchar	63	0	☐	☐	
money	double	16	2	☐	☐	
consumAmount	double	16	2	☐	☐	
realTalkTime	int	12	0	☐	☐	
realSMSCount	int	12	0	☐	☐	
realFlow	int	12	0	☐	☐	