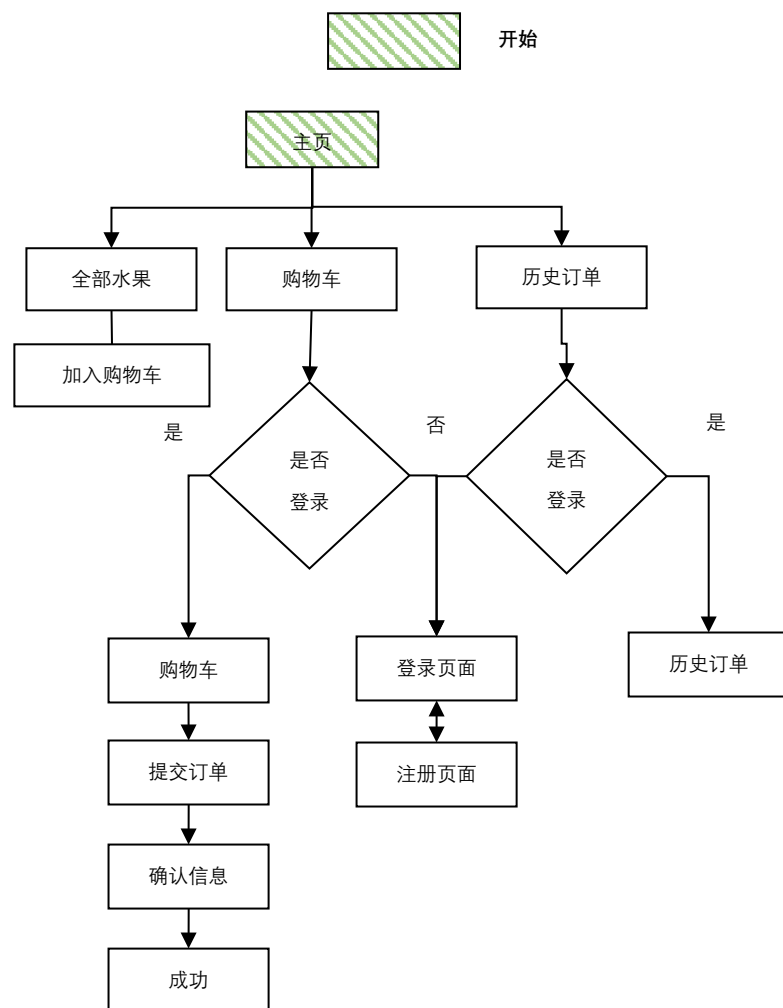


ws-fruitshop

一. 项目介绍

1. 项目简介

基于 SOA 的分布式电商客户端系统，各服务和总线之间采 soap1.2 协议



2. 采用关键技术

CXF3.7 框架

Spring

JSP&Servlet

JSTL

EL 表达式

c3p0 连接池

DbUtils 数据访问框架

BeanUtils

jQuery

Ajax

自定义事务管理机制 `MannagerThreadLocal`

SOAP1.2 协议

3. 开发环境

jdk1.7

tomcat8.5、8.0

MySQL5.5.27

MyEclipse Enterprise Workbench 2017 CI 7

字符集编码: utf-8

4. 项目演示

- 主页



欢迎使用水果商店

新鲜果蔬在线订购，送单到寝



欢迎使用水果商店

新鲜果蔬在线订购，送单到寝

● 登录



[返回首页](#) [注册](#)

欢迎登陆Fruit Shop
新鲜果蔬，便捷订购

登录



[返回首页](#) [注册](#)

欢迎登陆Fruit Shop
新鲜果蔬，便捷订购

!用户ID不能为空

密码不能为空

登录



[返回首页](#) [注册](#)

欢迎登陆Fruit Shop
新鲜果蔬，便捷订购
登录失败，帐号或密码错误

登录



登录中...

● 注册



[返回首页](#) [登录](#)

欢迎注册Fruit Shop

新鲜果蔬，便捷订购

用户ID

用户名

密码

确认密码

性别
男

手机号码

● 购物车



序号	商品名	单价	数量	库存	小计	操作
1	猕猴桃	26.0	<input type="button" value="减"/> 1 <input type="button" value="加"/>	9	26.0	删除
2	橘子	16.0	<input type="button" value="减"/> 2 <input type="button" value="加"/>	7	32.0	删除

合计: 58.0元

结算

首页全部水果购物车历史订单赵煜注册

加入购物车

序号	商品名	单价	数量	小计
1	猕猴桃	26.0	2	52.0
2	橘子	16.0	2	32.0

收货人姓名
赵煜

手机号码
18640378235

配送地址
昆明

配送时间
2018/02/24 14:00

合计: 84.0元


提交

小计	操作
52.0	删除
32.0	删除


结算

● 水果列表






苹果
16.0元/份
4个*1份 200g以上/个
加入购物车



青芒
20.0元/份
4个*1份 200g以上/个
加入购物车



蛇果
17.0元/份
4个*1份 约190g/个
加入购物车

« 第2页/共2页 »

● 历史订单

FRUIT SHOP

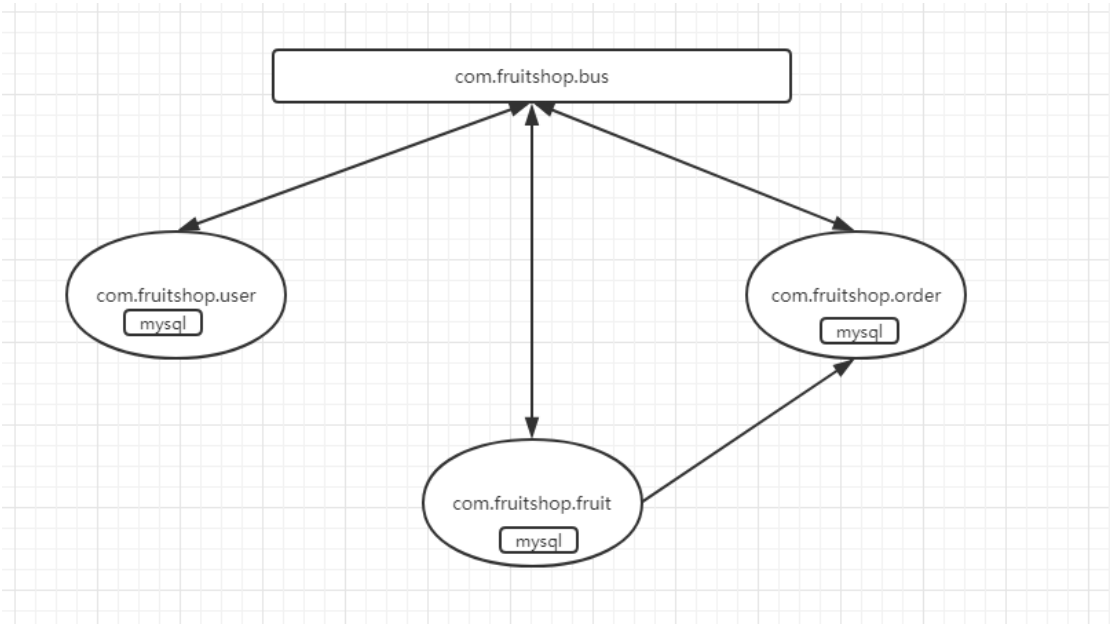
历史订单

序号	订单号	收货人	地址	联系方式	收货时间	提交时间	状态	总价	操作
1	20181205092558510	赵煜	昆明	18640378235	2018-02-24 14:00:00.0	2018-12-05 09:25:58.0	未支付	22.0	查看
2	20181205092834230	赵煜	昆明	18640378235	2018-02-24 14:00:00.0	2018-12-05 09:28:34.0	未支付	22.0	查看
3	20181205103522822	赵煜	昆明	18640378235	2018-02-24 14:00:00.0	2018-12-05 10:35:22.0	未支付	24.0	查看
4	20181207034712923	赵煜	昆明	18640378235	2018-02-24 14:00:00.0	2018-12-07 03:47:12.0	未支付	14.0	查看
5	20181209023906943	赵煜	昆明	18640378235	2018-02-24 14:00:00.0	2018-12-09 02:39:06.0	未支付	84.0	查看

二. 源码简析

● 架构设计

ws-fruitshop-bus	总线
ws-fruitshop-fruit	提供水果服务
ws-fruitshop-order	提供订单服务
ws-fruitshop-user	提供用户服务



● 暴露接口描述

服务接口	方法	描述	发布地址	Wsd1 文件
UserServiceInterface	login()	为总线提供登录服务	http://127.0.0.1:8081/ws-fruitshop-user/ws	UserServiceInterface_wsdl.xml
	regist()	为总线提供注册服务		
FruitServiceInterface	findFruitPage()	为总线提供水果页服务	http://127.0.0.1:8084/ws-fruitshop-fruit/ws	FruitServiceInterface_wsdl.xml
	findFruitById()	为总线提供查询水果服务		
FruitServiceInterface2	updateReserve()	为order服务提供更新水果库存服务		FruitServiceInterface2_wsdl.xml
OrderServiceInterface	findOrderByUserID()	为总线提供根据用户ID查找订单服务	http://127.0.0.1:8087/ws-fruitshop-order/ws	FruitServiceInterface2_wsdl.xml
	addOrder()	为总线提供添加订单服务		

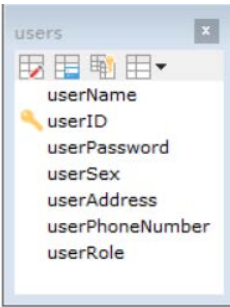
Available SOAP services:	
UserServiceInterface <ul style="list-style-type: none">loginregist	Endpoint address: http://127.0.0.1:8081/ws-fruitshop-user/ws/user WSDL: (http://service.user.com/UserServiceInterfaceService) Target namespace: http://service.user.com/

Available SOAP services:	
FruitServiceInterface <ul style="list-style-type: none">findFruitPagefindFruitById	Endpoint address: http://127.0.0.1:8084/ws-fruitshop-fruit/ws/fruit WSDL: (http://service.fruit.com/FruitServiceInterfaceService) Target namespace: http://service.fruit.com/
FruitServiceInterface2 <ul style="list-style-type: none">updateReserve	Endpoint address: http://127.0.0.1:8084/ws-fruitshop-fruit/ws/fruit2 WSDL: (http://service.fruit.com/FruitServiceInterface2Service) Target namespace: http://service.fruit.com/

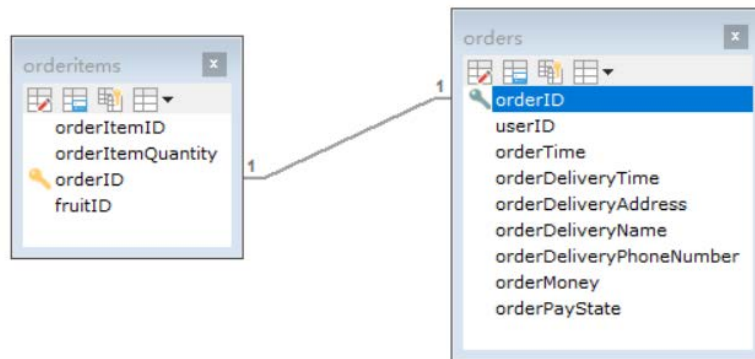
Available SOAP services:	
OrderServiceInterface <ul style="list-style-type: none">findOrderByUserIDaddOrder	Endpoint address: http://127.0.0.1:8087/ws-fruitshop-order/ws/order WSDL: (http://service.order.com/OrderServiceInterfaceService) Target namespace: http://service.order.com/

● 数据库

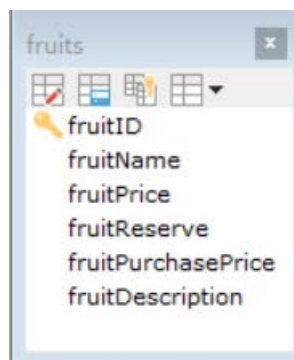
ws-fruitshop-user 项目使用 mysql5.5.27



ws-fruitshop-order 项目使用 mysql5.5.27



ws-fruitshop-fruit 项目使用 mysql5.5.27



● webservice 服务开发

1. 下载 cxf 框架，并配置系统环境变量。本次开发使用 apache-cxf-3.2.7。
2. 创建 web 项目，并导入 CXF 下 lib 文件夹里的所有 jar 包。
3. 创建 SEI 接口，注意加入 '@WebService' 标签，如：

```
@WebService
@BindingType(SOAPBinding.SOAP12HTTP_BINDING) // 使用SOAP1.2版本
public interface UserServiceInterface {
    public User login(String userID, String userPassword);

    public boolean regist(User user);
}
```

4. 创建 SEI 实现类，如

```
public class UserDaoInterfaceImpl implements UserDaoInterface {

    @Override
    public User findUserByIdAndPassword(String userID, String userPassword) throws SQLException {
        QueryRunner qr = new QueryRunner(C3P0Util.getDataSource());
        return qr.query("SELECT * FROM users WHERE userID=? AND userPassword=?", new BeanHandler<User>(User.class), userID,
            userPassword);
    }

    @Override
    public boolean addUser(User user) throws SQLException {
        QueryRunner qr = new QueryRunner(C3P0Util.getDataSource());

        String sql = "INSERT INTO users(userID,userName,userPassword,userSex,userAddress,userPhoneNumber,userRole) VALUES(?,?,?,?,?,?,?)";
        qr.update(sql, user.getUserID(), user.getUserName(), user.getUserPassword(), user.getUserSex(),
            user.getUserAddress(), user.getUserPhoneNumber(), user.getUserRole());
        return true;
    }
}
```


5. 配置 spring 配置文件 applicationContext.xml 发布服务，配置服务地址、服务接口和服务实现类。如：

```
<?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:jaxws="http://cxf.apache.org/jaxws"
       xmlns:jaxrs="http://cxf.apache.org/jaxrs" xmlns:cxfrs="http://cxf.apache.org/core"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd
                           http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd
                           http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd">

    <jaxws:server address="/user"
                 serviceClass="com.user.service.UserServiceInterface">
        <jaxws:serviceBean>
            <ref bean="userServiceInterface"></ref>
        </jaxws:serviceBean>
    </jaxws:server>

    <!-- 配置服务实现类 -->
    <bean name="userServiceInterface" class="com.user.service.impl.UserServiceInterfaceImpl"></bean>

</beans>
```

6. 配置 web.xml，配置 spring 配置文件和加载的 listener，配置 CXF 的 servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         id="WebApp_ID" version="3.1">
    <display-name>ws-fruitshop-user</display-name>

    <!-- 设置spring的环境 -->
    <context-param>
        <!-- contextConfigLocation不能改 -->
        <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>

    <!-- 配置CXF的Servlet -->
    <servlet>
        <servlet-name>CXF</servlet-name>
        <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CXF</servlet-name>
        <url-pattern>/ws/*</url-pattern>
    </servlet-mapping>
</web-app>
```

7. 将项目部署到 tomcat 下，启动 tomcat

8. 测试服务

WSDL 地址规则：http://ip:端口号/项目名称/servlet 拦截路径

如：<http://localhost:8081/ws-fruitshop-user/ws>

Available SOAP services:

UserServiceInterface <ul style="list-style-type: none">loginregist	Endpoint address: http://localhost:8081/ws-fruitshop-user/ws/user WSDL : http://service.user.com/?UserServiceImplService Target namespace: http://service.user.com/
--	--

● webservice 服务生成客户端并部署

1. 在客户端项目下引入 CXF 下 lib 文件夹里的所有 jar 包。

2. 使用 CXF 提供的 wsdl2java 工具生成客户端代码。

如：`wsdl2java -p com.fruitshop.service.user -d . http://localhost:8081/ws-fruitshop-user/ws/user?wsdl`

3. 配置客户端 spring 配置文件 applicationContext.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:jaxws="http://cxf.apache.org/jaxws"
       xmlns:jaxrs="http://cxf.apache.org/jaxrs" xmlns:cxf="http://cxf.apache.org/core"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd
                           http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd
                           http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd">

    <!-- http://127.0.0.1:8081/ws-fruitshop-user/ws -->
    <jaxws:client id="userServiceClient"
        address="http://127.0.0.1:8081/ws-fruitshop-user/ws/user"
        serviceClass="com.fruitshop.user.service.UserServiceInterface"/>


```

4. 根据 wsdl 说明书使用生成代码来调用远程服务。

```
public void login(HttpServletRequest request, HttpServletResponse response, String userID, String userPassword) throws ServletException, IOException {
    ApplicationContext context = new ClassPathXmlApplicationContext("classpath:applicationContext.xml");
    UserServiceInterface userServiceInterface = (UserServiceInterface) context.getBean("userServiceClient");
    User user = userServiceInterface.login(userID, userPassword);

    if(null==user){
        System.out.println("用户不存在");
        request.setAttribute("msg_login", "登录失败, 帐号或密码错误");
        request.getRequestDispatcher("/login.jsp").forward(request, response);
    }else{
        System.out.println(user.getUserName());
        System.out.println(user.getUserSex());
        request.getSession().setAttribute("user", user);
        request.getRequestDispatcher("/index.jsp").forward(request, response);
    }
}
```

● 服务业务关键代码

1. 三个服务项目和一个总线项目统一采用 MVC 风格，包的命名格式类似下图：

```
> com.fruit.dao
> com.fruit.dao.impl
> com.fruit.pojo
> com.fruit.service
> com.fruit.service.impl
> com.fruit.util
```

2. Dao 层使用 C3P0 连接池和 Apache.commons.dbutils 框架对数据库进行操作。

```
public class C3P0Util {
    private static DataSource dataSource = new ComboPooledDataSource();

    public static DataSource getDataSource() {
        return dataSource;
    }

    public static void setDataSource(DataSource dataSource) {
        C3P0Util.dataSource = dataSource;
    }

    public static Connection getConnection() {
        try {
            return dataSource.getConnection();
        } catch (SQLException e) {
            throw new RuntimeException("服务器错误");
        }
    }

    public boolean addUser(User user) throws SQLException {
        QueryRunner qr = new QueryRunner(C3P0Util.getDataSource());

        String sql = "INSERT INTO users(userID,userName,userPassword,userSex,userAddress,userPhoneNumber,userRole) VALUES(?,?,?,?,?,?,?)";
        qr.update(sql, user.getUserID(), user.getUserName(), user.getUserPassword(), user.getUserSex(),
            user.getUserAddress(), user.getUserPhoneNumber(), user.getUserRole());
        return true;
    }
}
```

3. 事务管理，自定义线程管理类 ManagerThreadLocal。

```

~
public void addOrder(Order order) throws Exception {

    try {
        ManagerThreadLocal.startTransaction();
        orderDaoInterface.addOrder(order);
        orderItemDaoInterface.addOrderItem(order);

        ArrayList<UpdateEntry> entryList = new ArrayList<UpdateEntry>();
        for (int i = 0; i < order.getOrderItems().size(); i++) {
            UpdateEntry entry = new UpdateEntry();
            entry.setKey(order.getOrderItems().get(i).getFruitID());
            entry.setValue(order.getOrderItems().get(i).getOrderItemQuantity());
            entryList.add(entry);
        }

        ApplicationContext context = new ClassPathXmlApplicationContext("classpath:applicationContext.xml");
        FruitServiceInterface2 fruitServiceInterface2 = (FruitServiceInterface2) context
            .getBean("fruitServiceClient2");

        ManagerThreadLocal.commit();
        fruitServiceInterface2.updateReserve(entryList);

    } catch (Exception e) {
        ManagerThreadLocal.rollback();
        throw new Exception("提交订单失败! ");
    }
}

```

4. 使用 Apache.commons.beanutils 工具封装从前端传入后端的数据。

```
BeanUtils.populate(user, request.getParameterMap());
```

5. 使用 JSTL 的 <c:forEach> 标签水果展示、购物车信息、历史订单信息等都采用了动态表单。

```

<c:forEach items="${fpb.fruits}" var="f">
    <div class="col-xs-4 col-sm-4 col-md-4 col-lg-4">
        <div class="thumbnail">
            
            <div class="caption">
                <h3>${f.fruitName}</h3>
                <div class="h5" style="color: orange">${f.fruitPrice}元/册</div>
                <div class="h4">${f.fruitDescription}</div>
                <p>
                    <a href="#" onclick="addFruitIntoCart('${f.fruitID}')"
                        class="btn btn-primary" role="button"
                        style="background: rgba(38, 80, 91, 0.65)"> 加入购物车</a>
                </p>
            </div>
        </div>
    </div>
</c:forEach>

```

6. 登录、注册、修改信息，使用 jQuery 在前台拦截可以不访问数据库确认的错误，减少数据库约束。以注册的 jQuery 为例，提前写好错误提示的 div 文字，当检测出需要拦截时，jQuery.css() 方法响应对应 div 的 id，更改样式。

```

$(document).ready(function(){
    $("#signup").click(function(){
        $("#username-error").css("display","none");
        $("#pass-error").css("display","none");
        $("#pass-error1").css("display","none");
        $("#repass-error1").css("display","none");
        $("#repass-error2").css("display","none");
        $("#userID-error").css("display","none");
        $("#address-error").css("display","none");
        $("#phone-error1").css("display","none");
        //注册

        switch(true){
            case $("#ID").val().length<6:
                $("#ID").css("border-color","rgb(204, 0, 0, 0.88)");
                $("#userID-error").css("display","block");
                break;
            case $("#username").val() == "":
                $("#username").css("border-color","rgb(204, 0, 0, 0.88)");
                $("#username-error").css("display","block");
                break;
            case $("#password").val() == "":
                $("#password").css("border-color","rgb(204, 0, 0, 0.88)");
                $("#pass-error").css("display","block");
                break;
            case $("#repassword").val() != $("#password").val():
                $("#repassword").css("border-color","rgb(204, 0, 0, 0.88)");
                $("#repass-error1").css("display","block");
                break;
            case $("#phone").val() == "":
                $("#phone").css("border-color","rgb(204, 0, 0, 0.88)");
                $("#phone-error1").css("display","block");
                break;
            case $("#address").val() == "":
                $("#address").css("border-color","rgb(204, 0, 0, 0.88)");
                $("#address-error").css("display","block");
                break;
            default:
                $("#right").hide();
                $("#upgrade-tips").hide();
                $("#success").css("display","block");
                $("#left-image").animate({left:'250px'},0.600);
                setTimeout("document.getElementById('regist-form').submit();","1500");
        }
    }
}

```

三. 项目部署简介

注：因为在本地测试，所有的服务和总线都部署在本地。

1. 在 MyEclipse 或 Eclipse 下打开四个项目。
2. 分别右键每个“工程 -> Properties -> Java Build Path -> Add JARs”导入所有“外部 jar 包”文件夹下的 jar 包。
3. 在项目文件中，已经包含了服务所生成的客户端代码。注意生成的客户端代码还是 java 源码，并没有进行打包处理，已经导进了项目源码中。
4. 向数据库中导入三个数据库脚本 fruit.sql、order.sql、user.sql，会生成三个数据库，数据库中已经存在一些开发时测试用的数据。
5. 根据本地数据库信息配置 ws-fruitshop-fruit、ws-fruit-order 和 ws-fruit-user 三个服务项目 src 下的 c3p0-config.xml。

```

<?xml version="1.0" encoding="utf-8"?>
<c3p0-config>
    <default-config>
        <property name="user">root</property>
        <property name="password">123</property>
        <property name="driverClass">com.mysql.jdbc.Driver</property>
        <property name="jdbcUrl">jdbc:mysql:///user</property>
    </default-config>
</c3p0-config>

```

6. 准备 4 个 tomcat 服务器，先将提供服务的 ws-fruitshop-fruit、ws-fruit-user 项目部署在相应的两个服务器上。
7. 由于服务项目 ws-fruitshop-order 和总线 ws-fruit-bus 都用到了其他服务提供的接口，所以先配置 ws-fruitshop-order 项目 config 下的 applicationContext.xml 的 fruit 服务地址。只需根据 ws-fruitshop-fruit 所在服务器的端口号修改端口号即可。配置完成后，将项目 ws-fruitshop-order 部署在第三个服务器上，并启动。

```
<jaxws:client id="fruitServiceClient2"
    address="http://127.0.0.1:8084/ws-fruitshop-fruit/ws/fruit2"
    serviceClass="com.fruit.service2.FruitServiceInterface2" />
```

8. 再配置 ws-fruitshop-bus 项目 config 下的 applicationContext.xml。根据前三个服务所在服务器的端口号修改端口号即可。

```
<!-- http://127.0.0.1:8081/ws-fruitshop-user/ws -->
<jaxws:client id="userServiceClient"
    address="http://127.0.0.1:8081/ws-fruitshop-user/ws/user"
    serviceClass="com.fruitshop.user.service.UserServiceInterface"/>

<!-- http://127.0.0.1:8084/ws-fruitshop-fruit/ws/fruit -->
<jaxws:client id="fruitServiceClient"
    address="http://127.0.0.1:8084/ws-fruitshop-fruit/ws/fruit"
    serviceClass="com.fruitshop.fruit.service.FruitServiceInterface"/>

<!-- http://127.0.0.1:8087/ws-fruitshop-order/ws/order -->
<jaxws:client id="orderServiceClient"
    address="http://127.0.0.1:8087/ws-fruitshop-order/ws/order"
    serviceClass="com.fruitshop.order.service.OrderServiceInterface"/>
```

9. 再将 ws-fruitshop-bus 部署到第四个服务器上并启动访问：localhost:8080/ws-fruitshop-bus/index.jsp。当然端口号还是修改为总线项目部署服务器的端口号。
注意：

1. 在步骤“3”中讲到，所有服务的客户端代码已经生成，并且以 java 源代码的形式添加到相应项目的相应包中。当然，可按照该文档“webservice 服务生成客户端并部署”进行重新生成。生成之后在替换相应包中的源码即可，所以下面给出各项目用到其他外部服务的客户端代码所放的位置。

1. 在 ws-fruitshop-bus 中 com.fruitshop.fruit.service 存放 ws-fruitshop-fruit 提供的 FruitServiceInterface 服务。com.fruitshop.order.service 存放 ws-fruitshop-order 提供的 OrderServiceInterface 服务。com.fruitshop.user.service 存放 ws-fruitshop-user 提供的 UserServiceInterface 服务。
2. 在 ws-fruitshop-order 中 com.fruit.service2 存放 ws-fruitshop-fruit 提供的 FruitServiceInterface2 服务

2. 各服务提供方的发布地址也是能修改的，例如 <http://127.0.0.1:8084/ws-fruitshop-fruit/ws/fruit?wsdl>

- ‘127.0.0.1’ 是服务发布的服务器地址；
- ‘8084’ 是服务器发布服务的相应端口号；

- ‘ws-fruitshop-fruit’ 是服务项目名称;
- ‘ws’ 是 CXF 的 servletd 的 url-pattern, 可在 WebRoot/WEB-INF/web.xml 中修改;
- ‘fruit’ 是服务接口的地址可在 config/applicationContext.xml 中配置。