# 支付系统架构方案

下图是目前的支付页面，以及支付页面的url

<http://www.gxyj.com/security/shoppingcart/verify.jhtml?orderId=020170315PO0074016&channel=00>



## 支付系统总体架构

对接多个订单系统



## B2C支付流程

B2B系统 零批系统 支付流程与下图基本一致

支付系统作为单独系统其数据库也是独立的

这里涉及到支付成功后分布式事物问题：更新订单库和更新支付库必须是原子性的



订单系统需要提供两个接口：

**查询订单状态**

* 请求参数：订单号
* 返回结果：订单是否已支付，支付金额

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段解释** | **类型** | **备注** |
| result | 返回结果 | int | 100 订单可以支付  200 订单已经支付  201 订单号不存在  202 请求参数错误  300 服务器端错误 |
| money | 支付金额 | string | 格式： 123.45 |
| message | 备注信息 | string |  |

* 数据格式：json

**更新订单状态**

会涉及到减库存 优惠券 加积分等业务

* 请求参数：第三方支付传过来的所有参数
* 返回结果：处理结果

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段解释** | **类型** | **备注** |
| result | 返回结果 | int | 100 更新成功  200 请求参数错误  300 服务器端错误 |
| message | 备注信息 | string |  |

* 数据格式：json

支付系统需要对外提供的接口（Service）

查询订单支付信息

## 支付系统分层

下图默认上次依赖下层，箭头关系表示可直接依赖，如开放接口层可直接依赖于web层，也可以依赖service层。



* **开发接口层**：可直接封装Servcie层接口暴露成RPC接口，通过web封装成http接口，网关控制层等
* **终端显示层**：各个终端的模板渲染并执行显示。当前主要是velocity渲染 js渲染 jsp渲染 移动端展示层
* **Web层**：主要对访问控制进行转发，各类基本参数校验，或者不复用的业务简单处理等
* **Service层**：具体的业务逻辑层
* **Manager层**:通用业务处理层，它有如下特征：

1. 对第三方平台封装的层，预处理返回结果及转化异常信息；
2. 对service层通用能力的下沉，如缓存方案、中间件通用处理；
3. 与DAO层交互，对DAO的业务通用能力的封装

* **DAO层**：数据访问层，与底层mysql oracle hbase进行数据处理
* **外部接口或第三方平台**：包括其他部门RPC开放接口，基础平台，其他公司的http接口

## 支付流程图



## 第三方支付通知处理逻辑





## 支付系统部署图

* OpenResty集群作为网关入口，负责**请求认证 流量控制** 等工作
* Tomcat集群负责收集请求数据，然后调用服务层
* 服务层采用SOA架构，对外发布服务
* 数据库层采用主从加redis方式 提高性能



## 支付系统拆分顺序

1 pc端支付

2 手机安卓端

3 手机ios端

4 微商城端