客户名字和 logo

餐厅订餐系统-系统架构设计文档

XU LIN

日期：2014-12-02

文档版本历史

版本号	修订日期	修订人	审核人	变更内容
0.1	2014/11/21	Xu Lin		初始化
0.2	2014/11/22	Xu Lin		更新目录
1.0	2014/11/23	Wanglinglong		增加 Android 端技术方案
	2014/11/23	Nie Annie		修改格式
	2014-11-27	LUO ZHI		添加 ER 图
	2014/11/28	Xu Lin		增加第 1、2、3、5 章内容
	2014-11-28	LUO ZHI		修正文档格式
	2014-11-29	LUO ZHI		增加业务流程图，网络拓扑图，系统架构图，系统模块图
	2014/12/1	Xu Lin		更新缩写词和参考文档
	2014-12-01	LUO ZHI		1. 修改 3.1 以及 3.1.1 2. 修改 3.3
	2014/12/2	Xu Lin		更新 Struts2 相关内容
	2014/12/2	Xu Lin		更新数据库字段
	2014-12-02	LUO ZHI		更新格式 合并 Android 设计部分

目录

1 文档介绍

1.1 文档目的

本文描述 eOrder 餐厅订餐系统架构设计，用于指导开发人员进行系统详细设计；需求人员进行系统需求实现评估；测试人员进行系统测试用例编制。

1.2 文档范围

本文从系统 5 视图角度对 eOrder 餐厅订餐系统进行描述，包括系统逻辑视图（类图/时序图），开发视图（开发环境，依赖第三方 JDK，技术框架选型），架构视图，网络拓扑图，系统功能模块图以及数据库 ER 图。

系统模块的具体实现部分不在本文描述范围内，请参考系统详细设计文档。

1.3 缩写词列表

缩写词	解释
MVC	Model View Controller
SSH2	Struts2 Spring Hibernate
WebWork	WebWork 是由 OpenSymphony 组织开发的，致力于组件化和代码重用的 J2EE Web 框架。
IOC	Invest of Control 控制反转是关于一个对象如何获取他所依赖的对象的引用，这个责任的反转。
AOP	Aspect Oriented Programming 通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。
DI	Dependency Injection 依赖注入
DJ	
OOP	Object Oriented Programming 面向对象编程
EJB	Enterprise JavaBean EJB 是 Oracle 的 JavaEE 服务器端组件模型，设计目标与核心应用是部署分布式应用程序。
J2EE	Java 2 Platform,Enterprise Edition
CMP	
HTML	HyperText Markup language 超级文本标记语言
POM	Project Object Model 项目对象模型
TDD	Test-Driven Development 测试驱动开发
JavaEE	Java Enterprise Edition Java 企业级实现

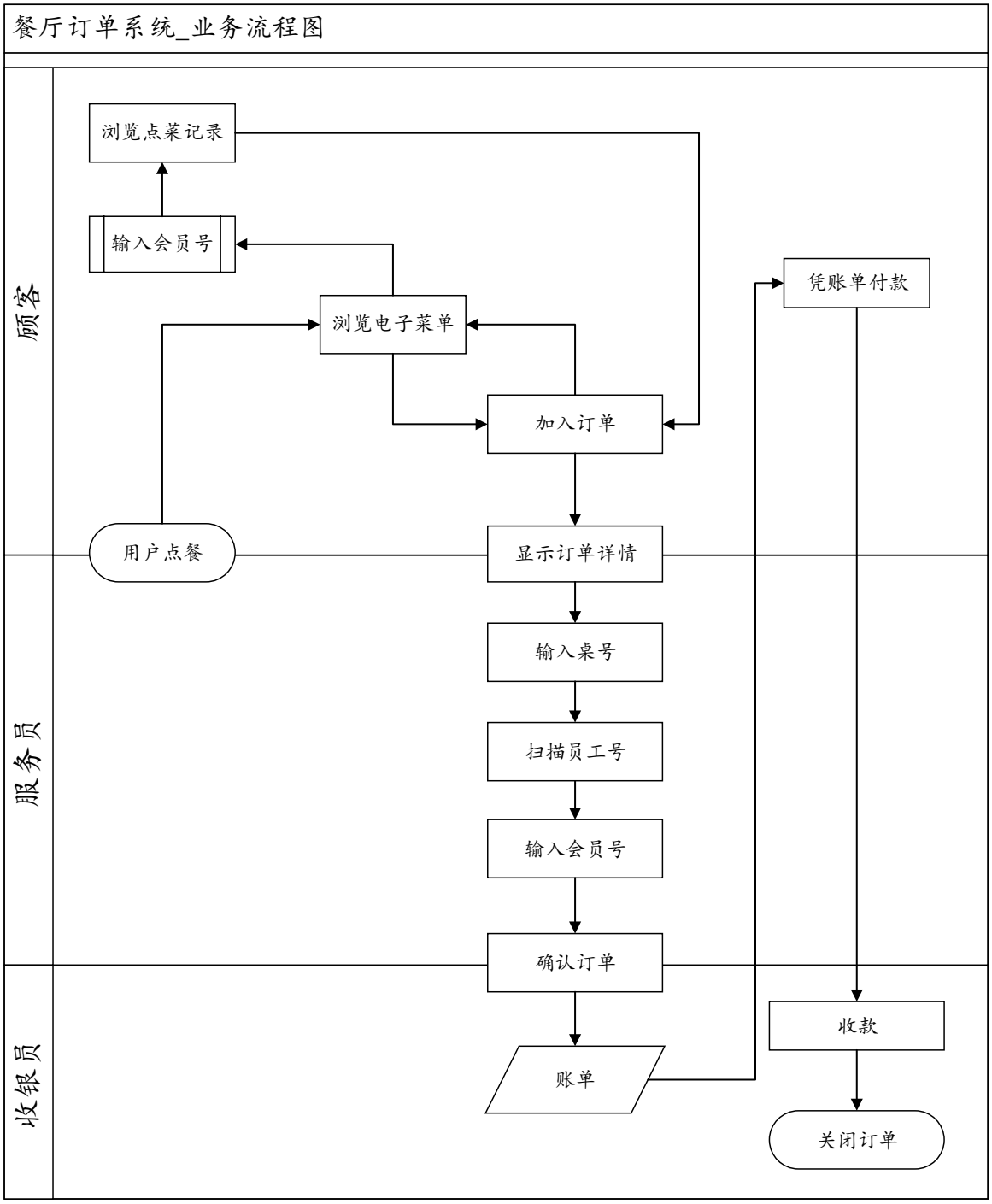
1.4 参考内容

- Spring 百度百科 http://baike.baidu.com/subview/23023/11192342.htm
- Struts2 百度百科 http://baike.baidu.com/view/1566725.htm
- Hibernate 百度百科 http://baike.baidu.com/view/7291.htm

2 系统范围

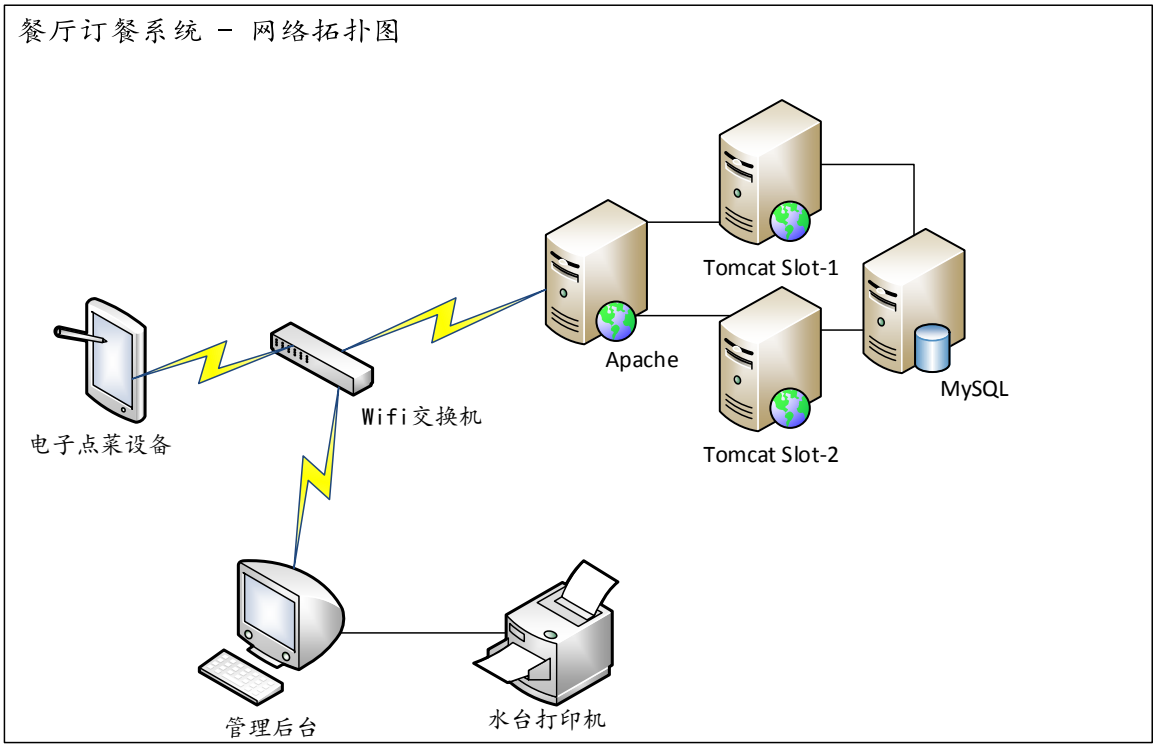
2.1 业务流程图

eOrder 订餐系统为餐饮业主提供了图形化点餐，下单，结帐以及会员管理的解决方案。整个业务流程图如下图所示：



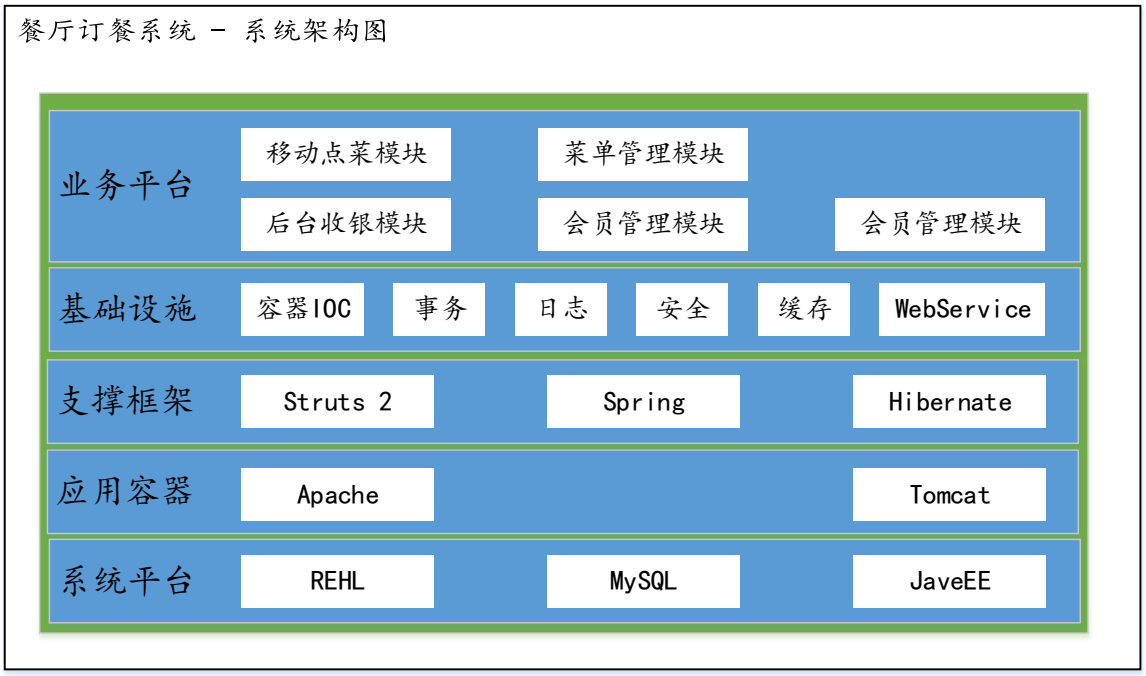
2.2 网络拓扑图

eOrder 订餐系统由服务器和客户端组成，下图是 eOrder 订餐系统的网络拓扑图。eOrder 订餐系统架构于餐饮业主餐厅组建的局域网内，与 Internet 没有连接，通过交换机连接服务器和客户端。服务器由 Apache HTTP 与 2 个 Tomcat 应用服务器一起构成负载均衡方案，后台数据库使用开源的 MySQL。前端架构使用 B/S 和 C/S 设计。服务员和顾客使用手持设备上的点餐应用进行点菜，下单；后台管理使用网页浏览器进行账单结算，菜品维护和用户维护等功能。



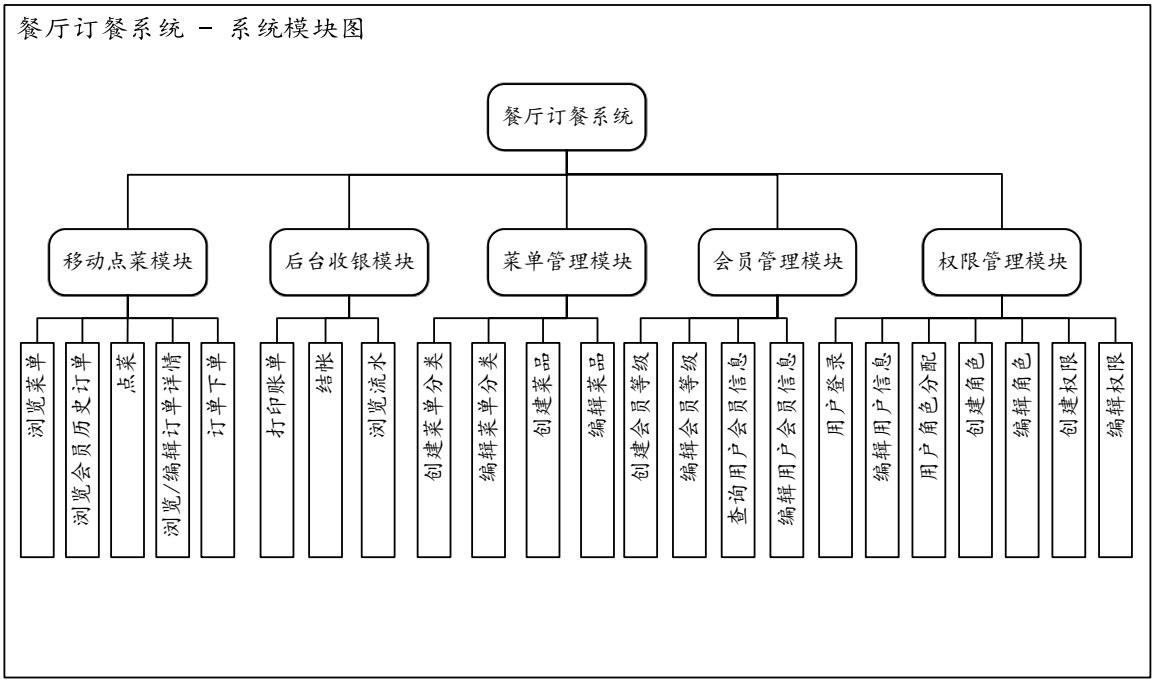
2.3 系统架构图

eOrder 订餐系统使用开源服务器和组件实现，下图描述了 eOrder 订餐系统的系统架构图。系统平台由红帽 Linux 企业版，MySQL 和 JavaEE 7+运行时构成。应用容器使用 Apache HTTP 和 Tomcat，开发框架选用传统 SSH 架构，使用容器 IOC，事务处理，日志，安全等功能实现业务模块。

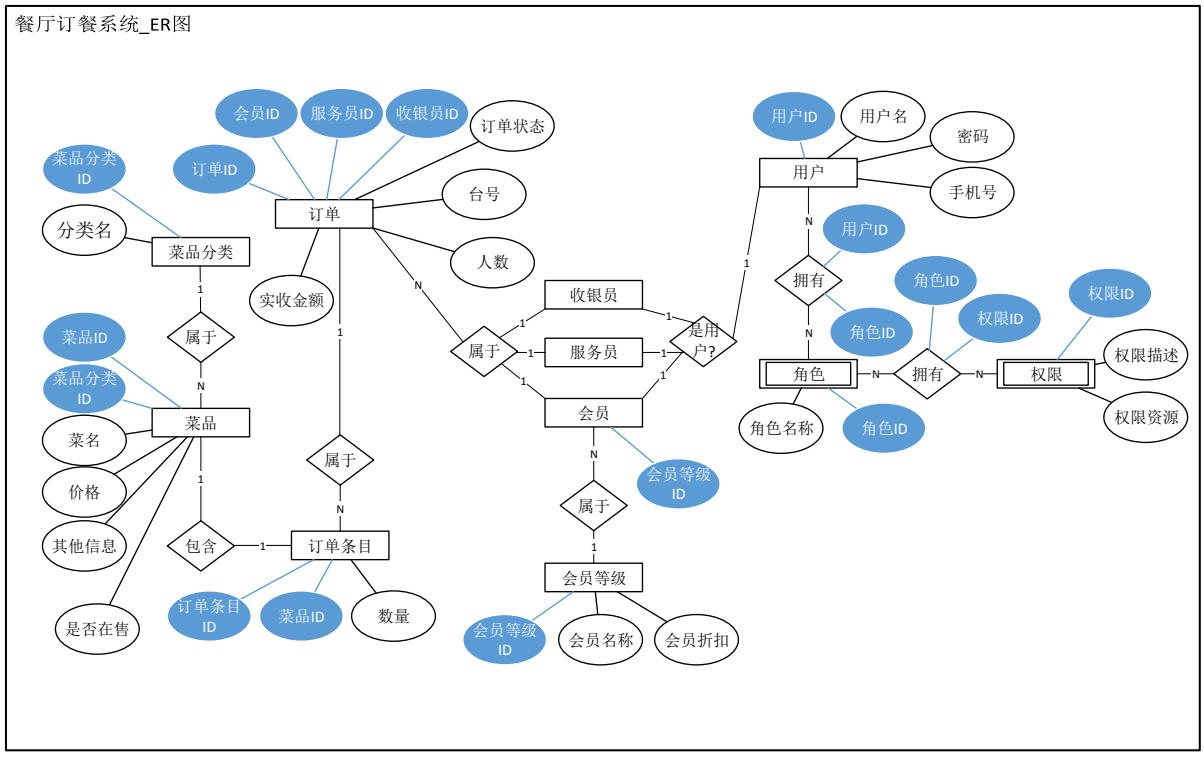


2.4 系统模块图

eOrder 订餐系统可以分为移动点餐模块，后台收银模块，菜单管理模块，会员管理模块和用户权限管理模块一共 5 大模块，其各自功能分解如下图所示



2.5 系统数据 ER 图


餐厅订餐系统_ER图
会员ID
服务员ID
收银员ID
订单状态
菜品分类ID
订单ID
订单
台号
分类名
菜品分类
人数
实收金额
菜品ID
菜品分类ID
属于
菜品
菜名
价格
其他信息
是否在售
包含
订单条目
订单条目ID
菜品ID
数量
属于
收银员
属于
服务员
是用户?
会员
会员等级ID
属于
会员等级
会员等级ID
会员名称
会员折扣
用户ID
用户名
密码
用户
手机号
用户ID
拥有
角色ID
角色ID
权限ID
权限ID
角色
拥有
权限
权限描述
角色名称
角色ID
权限资源

3 系统实现技术选型

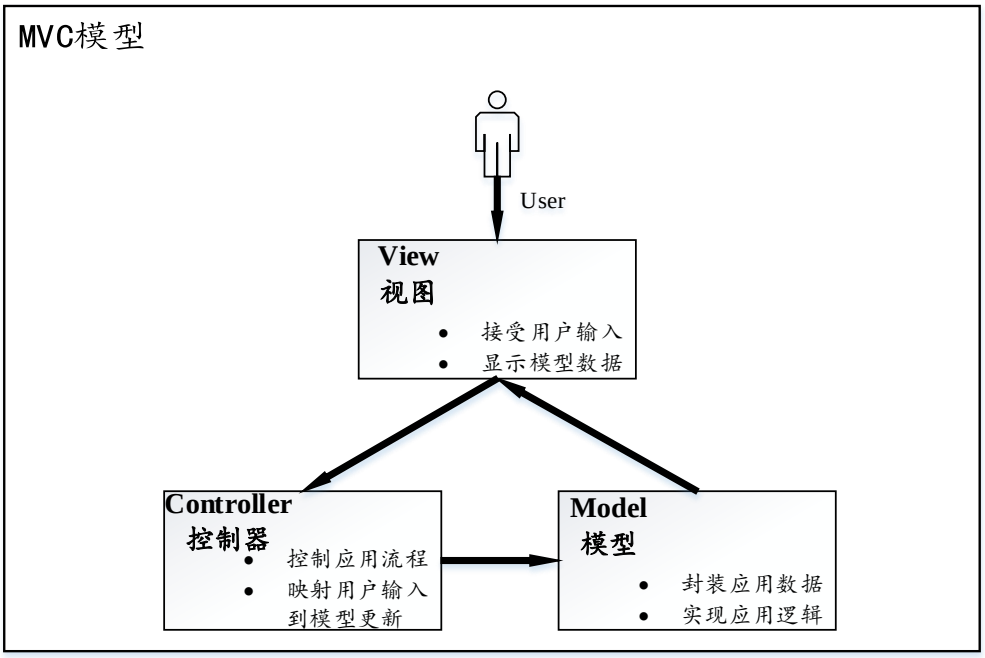
3.1 系统架构

现如今主流的企业应用主要基于浏览器/服务器(B/S)和客户端/服务器(C/S)架构。这两者的区别主要在于 B/S 利用浏览器作为系统的展现层接受用户输入，显示服务器返回，主要计算压力在服务器端，这主要是为了解决个人电脑元算能力较弱的场景。而 C/S 架构的应用这相反，业务逻辑的计算压力由客户端承担。但随着 Ajax 技术和个人电脑运算能力的提高，这两者之间的界限也就不那么明显了。eOrder 系统使用 C/S 架构实现移动点菜功能，而是用 B/S 架构实现后端管理功能。

3.1.1 系统分层模型 - MVC

MVC 是模型（Model）、视图（View）和控制器（Controller）的缩写。MVC 模型是一种经典的软件架构模型，它将软件系统分为 Model，View 和 Controller 三个部分。这三个部分的功能分别是，

- Model – 对应用中的数据实体进行封装，实现业务逻辑。在实际开发过程中，Model 层会进步一步希细分为代表应用实体的 POJO 和提供业务逻辑的服务 Service 层。Model 层的更新由视图层显示给用户
- Controller – 控制器的作用在于对用户输入请求进行处理，将用户的请求映射到正确的 Model 上，进行 Model 层更新
- View – 视图层的作用在于以图形化的方式将 Model 层的业务实体数据以符合业务逻辑的方式展现给终端用户。同时提供用户输入接口，提交用户请求到 Controller 层



MVC 模型如上图所示，由用户请求开始，到用户得到视图响应为止，形成一个闭环。这种系统架构模型的好处是有效的将显示，控制与业务数据逻辑分离，团队成员可以集中在某一个层面进行代码工作，从而提高团队开发效率。

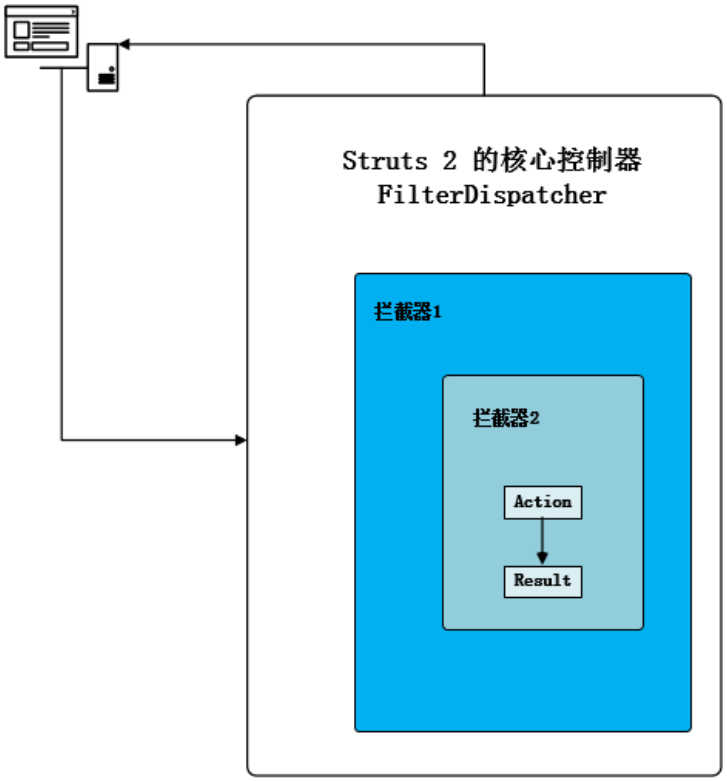
3.1.2 系统分层模型实现 - SSH2

SSH2 是 **S**truts**2**，**S**pring 和 **H**ibernate 的缩写。Hibernate 是持久层的解决方案，对应于 MVC 分层中的 M。Struts2 则实现了 MVC 分层系统中的 V 和 C，当然除了可以使用 Struts2 自带的视图实现方案，也可以集成其他的视图开源组件。Spring 是系统各层组件的粘合剂。使用 Spring 的依赖注-IOC 无需人工管理 MVC 各层组件的生命周期；结合面向接口编程-AOP，可以将公共功能进行抽象，灵活配置，达到解耦模块的目的；使用 Spring 的事务管理，实现容器托管事务处理，实现更为简单与灵活。

3.2 Struts2

3.2.1 概述

Apache Struts2 是一个优雅的，可扩展的 JAVA EE web 框架。框架设计的目标贯穿整个开发周期，从开发到发布，包括维护的整个过程。Struts2 的体系结构如下图所示:



3.2.2 优点与不足

优点：

- 基于 MVC 架构,框架结构清晰，开发流程一目了然，开发人员可以很好的掌控开发的过程。
- 使用 OGNL 进行参数传递。
- 强大的拦截器
- 易于测试
- 易于扩展的插件机制
- 模块化
- 全局结果与声明式异常

不足：

- 页面转到展示层时，需要配置 forward，每一次转到展示层，相信大多数都是直接转到 jsp，而涉及到转向，需要配置 forward。
- Struts2 的 Action 必需是 thread－safe 方式，它仅仅允许一个实例去处理所有的请求。所以 Action 用到的所有的资源都必需统一同步，这个就引起了线程安全的问题。
- 测试不方便。 Struts2 的每个 Action 都同 Web 层耦合在一起，这样它的测试依赖于 Web 容器，单元测试也很难实现。

3.2.3 拦截器

许多的 Struts2 中的 Action 需要共享一些共用信息或者是模块，有些 Action 需要对输入进行验证，另外一些 Action 或许需要对文件上传之前做一些逻辑处理，又或者一些 Action 需要对重复提交进行保护，还有一些 Action 需要在页面显示前，初始化下拉框或一些页面组件。

Struts2 框架使用了 Interceptor (拦截器) 策略使共享这些关心的模块更简单的使用起来。当需要使用到一些映射到 Action 的资源时，框架生成了 Action 对象，但在该 Action 被执行前，执法该 Action 中的方法被另外一个对象拦截了，在 Action 执行时，可能再次被拦截，我们亲切地称此对象为拦截器。

3.2.4 标签

Struts2 标签是标准 HTML 标签的扩充，提供了 HTML 标签不能实现的功能，大大简化的用户在展现层的工作。

Struts2 标签库包括用户界面标签库（表单/非表单标签库），非用户界面标签库（控制标签库，数据访问标签库），Ajax 支持标签库。

3.3 全功能 JavaEE 应用程序框架 - Spring

3.3.1 概述

在 Spring 出现之前，进行 JavaEE 企业级应用开发只能使用笨重的 EJB，开发过程陷入泥潭式的各种 Bean 和配置文件的编写，程序员的重心无法集中于业务流程本身。Spring 是一个开源的，全功能的轻量级 JavaEE 应用程序框架，它很好的弥补的 EJB 开发的不足，将开发人员从繁重的重复劳动中解放出来。使用 Spring，开发人员无需手动维护程序运行上下文，无需进行复杂的数据库连接对象编写，无需进行编程式事务管理等等。Spring 注重灵活的配置，面向接口编程，面向切面编程，使得程序的编写获得的最大的灵活度，和最小的代码量。Spring 犹如一管强大的胶水，可以很好的与 Struts，Hibernate 等其他框架集成。

Spring 应用框架有如下特性，

- 控制反转 – Spring 对应用运行上下进行管理，控制应用对象的生命周期而无需编码参与
- 面向切面编程 – 对软件系统中的通用功能，如日志，进行抽象，通过不入侵代码的方式实现通用功能
- 声明式事务处理 – 将事务处理成抽象，允许声明式的事务处，隔离事务逻辑与底层数据库无关
- 安全框架 – Spring 内置符合企业应用安全需要的 Security 框架，是 ACE 的一个实现，与 Spring 的天然集成可大大减少开发工作量

eOrder 订餐系统中将使用 Spring 应用框架的以上特性，下面各章节会对以上特性做进一步的说明。

3.3.2 控制反转 - IOC

控制反转（Inversion of Control，英文缩写为 IOC）是一个重要的面向对象编程的法则来削减计算机程序的耦合问题，也是轻量级的 Spring 框架的核心。 控制反转一般分为两种类型，依赖注入（Dependency Injection，简称 DI）和依赖查找（Dependency Lookup）。应用控制反转，对象在被创建的时候，由一个调控系统内所有对象的外界实体，将其所依赖的对象的引用，传递给它。也可以说，依赖被注入到对象中。所以，控制反转是，关于一个对象如何获取他所依赖的对象的引用，这个责任的反转。

3.3.3 面向切面编程

在软件中，面向切面编程（Aspect Oriented Programming，简称 AOP），是指通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。AOP 是 OOP 的延续，是软件开发中的一个热点，也是 Spring 框架中的一个重要内容，是函数式编程的一种衍生范型。利用 AOP 可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。

3.3.4 声明式事务管理

Spring 的声明式事务顾名思义就是采用声明的方式来处理事务。这里所说的声明，就是指在配置文件中申明。用在 Spring 配置文件中声明式的处理事务来代替代码式的处理事务。这样的好处是，事务管理不侵入开发的组件，具体来说，业务逻辑对象就不会意识到正在事务管理之中，事实上也应该如此，因为事务管理是属于系统层面的服务，而不是业务逻辑的一部分，如果想要改变事务管理策划的话，也只需要在定义文件中重新配置即可；在不需要事务管理的时候，只要在设定文件上修改一下，即可移去事务管理服务，无需改变代码重新编译，这样维护起来极其方便。

3.3.5 安全

Spring Security 是一个能够为基于 Spring 的企业应用系统提供声明式的安全访问控制解决方案的安全框架。它提供了一组可以在 Spring 应用上下文中配置的 Bean，充分利用了 Spring 的 IOC，DI 和 AOP 技术，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作。

Spring Security 对 Web 安全性的支持大量地依赖于 Servlet 过滤器。这些过滤器拦截进入请求，并且在应用程序处理该请求之前进行某些安全处理。 Spring Security 提供有若干个过滤器，它们能够拦截 Servlet 请求，并将这些请求转给认证和访问决策管理器处理，从而增强安全性。根据自己的需要，可以使用多个过滤器来保护自己的应用程序。

3.4 数据持久层 - Hibernate

3.4.1 概述

Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的用户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

3.4.2 好处与不足

优点：

- Hibernate 使用 Java 反射机制而不是字节码增强程序来实现透明性
- Hibernate 避免用户编码大量的原生 SQL 语句，提高开发质量和效率
- Hibernate 支持各种关系数据库，从一对一到多对多的各种复杂关系

不足：

- Hibernate 效率比 JDBC 略差
- Hibernate 不适合批量操作

3.4.3 ORM 关系对象模型

对象-关系映射（Object/Relation Mapping，简称 ORM），是随着面向对象的软件开发方法发展而产生的。面向对象的开发方法是当今企业级应用开发环境中的主流开发方法，关系数据库是企业级应用环境中永久存放数据的主流数据存储系统。对象和关系数据是业务实体的两种表现形式，业务实体在内存中表现为对象，在数据库中表现为关系数据。内存中的对象之间存在关联和继承关系，而在数据库中，关系数据无法直接表达多对多关联和继承关系。因此，对象-关系映射(ORM)系统一般以中间件的形式存在，主要实现程序对象到关系数据库数据的映射。

3.5 Web 前端技术

3.5.1 jQuery

jQuery 是一个优秀的轻量级 JavaScript 库，它兼容 CSS3，还兼容各种浏览器 jQuery 使用户能更方便地处理 HTML DOM、Events、实现动画效果，并且方便地为网站提供 AJAX 交互。jQuery 还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。

3.5.2 Ajax 技术

Ajax 指异步 JavaScript 及 XML（Asynchronous JavaScript and XML）。AJAX 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。简而言之，XmlHttpRequest 可以使用 JavaScript 向服务器提出请求并处理响应，而不阻塞用户。Ajax 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的 Web 应用程序的技术。

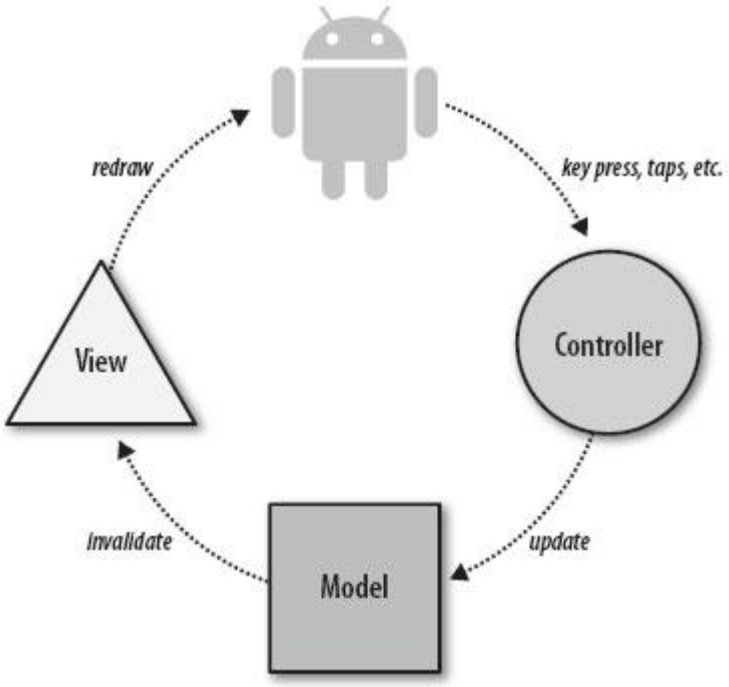
Ajax 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面。Ajax 可使因特网应用程序更小、更快、更友好。Ajax 是一种独立于 Web 服务器软件的浏览器技术。

3.6 客户端构建技术选型

目前主流的移动端开发平台为谷歌公司的安卓(Android)和苹果公司的 iOS，相比 iOS 平台，Android 平台因为其开源特性而被 eOrder 订餐系统选择为其移动客户端的实现平台。

3.6.1 系统模型

Android 客户端采用 MVC 架构，实际上在原生 Android 系统中已经采用了 MVC 架构的设计思想。如下图所示：



在 Android 系统中，MVC 分别指如下几类：

- 视图（View）

包括列表 List、网格 Grid、文本框 Textbox、按钮 Button、自定义 View 等等，上述控件组成的 Xml 文件放在 res/layout/目录中，实现了数据层与表示层的分离。

- 控制器（Controller）

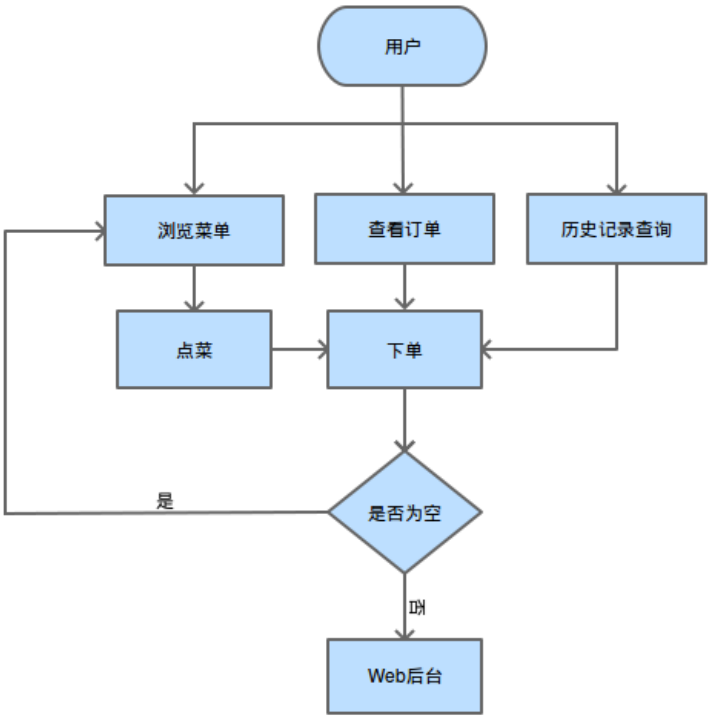
控制器的功能在 Intent（意图）、IntentFilter（意图过滤）、Broadcast Receiver（广播接收器）、BroadcastIntent（广播意图）、Service（服务）、Notification（通知）、Alarm（警告）、SMS（短信）、电话等逻辑功能代码中实现

- 模型（Model）

模型通常对应 Android 应用系统的业务部分（.java），放在 src 目录中。对数据库的操作、对网络等的操作都应该在 Model 里面处理，当然对业务计算等操作也是必须放在的该层的。

3.6.2 客户端流程图

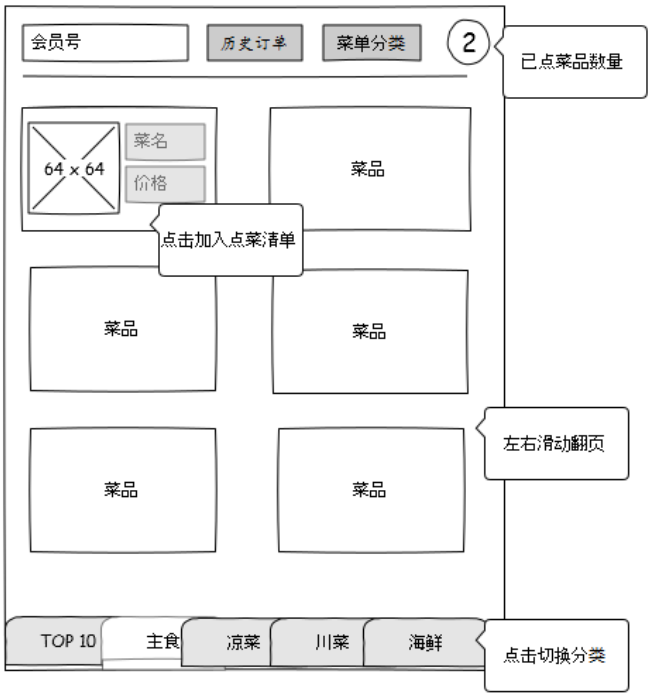
客户端流程图如图所示:



3.6.3 UI 实现技术

eOrder 订餐系统移动端主界面采用底部 Tab 栏的设计，通过底部 Tab 栏可以方便的切换不同的 Activity 或者自定义 View。然后在大部分界面中采用 GridView 来显示不同的分类项。

UI 设计原型图如下图所示



3.6.4 二维码扫描实现技术

二维码的定义：二维码（2-dimensional bar code），是用某种特定的几何图形按一定规律在平面（二维方向上）分布的黑白相间的图形记录数据符号信息的。在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图象输入设备或光电扫描设备自动识读以实现信息自动处理。

ZXing 是一个开放源码的，用 Java 实现的多种格式的 1D/2D 条码图像处理库，它包含了联系到其他语言的端口。ZXing 可以实现使用手机的内置的摄像头完成条形码的扫描及解码。该项目可实现的条形码编码和解码。目前支持以下格式：UPC-A、UPC-E、EAN-8、EAN-13、39 码、93 码。

3.7 Web Service

3.7.1 概念

Web Service 是一个平台独立的，低耦合的，自包含的、基于可编程的 web 的应用程序，可使用开放的 XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

3.7.2 REST

在 REST 出现之前，Web Service 的主要实现方式是 SOAP 和 XML-RPC，这就好比 Spring 之于 EJB，REST 大大的简化的开发人员的工作，通过实现 HTTP 协

议的 GET，POST，PUT 和 DELETE 方法，使得用户对资源操作的动作可以由 URI 进行直接传递。REST 也可以选择更为简洁和灵活的 JSON 作为传输协议。

3.7.3 JSON 介绍

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 JavaScript（Standard ECMA-262 3rd Edition - December 1999）的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。这些特性使 JSON 成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成(网络传输速度快)。

3.7.4 实现 Web Service 的中间件介绍

Jersey 是对 JSR311（用于 RESTful Web Services 的 Java API）的官方参考实现，它包含三个主要部分。

- 核心服务器（Core Server）：通过提供 JSR 311 中标准化的注释和 API 标准化，您可以用直观的方式开发 RESTful Web 服务
- 核心客户端（Core Client）：Jersey 客户端 API 帮助您与 REST 服务轻松通信
- 集成（Integration）：Jersey 还提供可以轻松与集成 Spring

4 数据库 - MySQL

MySQL 是一个开放源码的小型关联式数据库管理系统。MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

5 应用服务器

5.1.1 Apache HTTP 服务器

Apache HTTP Server，是 Apache 软件基金会的一个开放源码的网页服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是最流行的 Web 服务器端软件之一。它快速、可靠并且可通过简单的 API 扩展，将 Perl/Python 等解释器编译到服务器中。

5.1.2 Tomcat 应用服务器

Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。

5.1.3 Tomcat 双机负载平衡

对于 Web 应用集群的技术实现而言，最大的难点就是如何能在集群中的多个节点之间保持数据的一致性，会话（Session）信息是这些数据中最重要的一块。要实现这一点，大体上有两种方式，一种是把所有 Session 数据放到一台服务器上或者数据库中，集群中的所有节点通过访问这台 Session 服务器来获取数据；另一种就是在集群中的所有节点间进行 Session 数据的同步拷贝，任何一个节点均保存了所有的 Session 数据。两种方式都各有优点，第一种方式简单、易于实现，但是存在着 Session 服务器发生故障会导致全系统不能正常工作的风险；第二种方式可靠性更高，任一节点的故障不会对整个系统对客户访问的响应产生影响，但是技术实现上更复杂一些。常见的平台或中间件如 Microsoft ASP.net 和 IBM WAS 都会提供对两种共享方式的支持，Tomcat 也是这样，但是一般采用第二种方式。

5.1.4 HTTPS 技术

HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。 它是一个 URI scheme（抽象标识符体系），句法类同 http:体系。用于安全的 HTTP 数据传输。https:URL 表明它使用了 HTTP，但 HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP 与 TCP 之间）。这个系统的最初研发由网景公司(Netscape)进行，并内置于其浏览器 Netscape Navigator 中，提供了身份验证与加密通讯方法。现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面。

6 系统构建技术选型

6.1 持续集成

6.1.1 概念

持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试)来验证，从而尽快地发现集成错误。许多团队发现这个过程可以大大减少集成的问题，让团队能够更快的开发内聚的软件。

6.1.2 Maven

Maven 是基于项目对象模型 (Project Object Model，简称 POM) ，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。Maven 项目清晰定义了发布项目信息的方式，以及在多个项目中共享 JAR 的方式。

6.2 测试驱动开发

6.2.1 概念

测试驱动开发（Test-Driven Development，简称 TDD），是一种不同于传统软件开发流程的新型的开发方法。它要求在编写某个功能的代码之前先编写测试代码，然后只编写使测试通过的功能代码，通过测试来推动整个开发的进行。这有助于编写简洁可用和高质量的代码，并加速开发过程。

6.2.2 TDD

测试驱动开发的基本思想就是在开发功能代码之前，先编写测试代码，然后只编写使测试通过的功能代码，从而以测试来驱动整个开发过程的进行。这有助于编写简洁可用和高质量的代码，有很高的灵活性和健壮性，能快速响应变化，并加速开发过程。

测试驱动开发的基本过程如下:

- 快速新增一个测试
- 运行所有的测试（有时候只需要运行一个或一部分），发现新增的测试不能通过
- 做一些小小的改动，尽快地让测试程序可运行，为此可以在程序中使用一些不合情理的方法
- 运行所有的测试，并且全部通过
- 重构代码，以消除重复设计，优化设计结构
- 简单来说，就是不可运行/可运行/重构——这正是测试驱动开发的口号。

7 系统概要设计

7.1 开发环境

- JDK 1.7+
- Windows 7+
- Tomcat 6+
- MySQL 5+

7.2 运行平台

- REHL 6.x

7.3 异常处理

eOrder 系统不允许使用 Java 自带异常类进行异常处理，必须使用 Checked 异常派生出对应不用系统分层（MVC）的异常类。所有系统异常被捕捉后，使用自定义异常类进行重新封装，并按持久层->业务层->控制器层的顺序逐级向上抛出，统一在控制器层进行异常处理。

自定义异常类派生至 Exception 类，自定义异常类必须包含异常行为代码和异常消息。所有的异常消息均在需求文档附录中定义。

7.4 日志管理

7.4.1 日志的作用

- 系统运行日志：记录系统的运行情况，跟踪代码运行时轨迹；
- 异常和错误日志：记录异常堆栈信息，以供开发人员查看分析；
- 业务日志：记录业务信息和用户操作，例如用户登录、删除数据、更新数据等。

7.4.2 日志配置

系统中配置两个日志记录器，一个为异常记录器，专门记录异常信息，日志文件到达一定大小后将产生新的日志文件；另一个为系统运行记录器，按照日期记录所有的日志信息。

8 数据字典

数据表名	菜品分类表		t_dish_category		
数据项描述	数据项名	类型	长度	默认值	限制
菜品类别 ID	id	整型	32	n/a	主键，自动增加
类别名称	name	字符型	12	n/a	非空
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	菜品表		t_dish		
数据项描述	数据项名	类型	长度	默认值	限制
菜品类别 ID	id	整型	32	n/a	主键，自动增加
菜品分类 ID	category_id	整型	32	0	非空
菜品名称	name	字符型	12	n/a	非空
价格	price	浮点型	32	0.0	精确到小数点后一位
是否在售	on_sell	布尔型	1	True	非空
其他信息	misc	字符串	128	n/a	n/a
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	订单表		t_order		
数据项描述	数据项名	类型	长度	默认值	限制
订单 ID	id	整型	32	n/a	主键，自动增加
订单流水号	order_seq	字符型	16	n/a	格式为 <yyyymmddxxxxx> yyyy 四位数年份 mm 两位数月份 dd 两位日期 xxxxx 当天订单序号
订单状态	order_status	字符型	16	‘new’	非空
台号	table_num	整型	16	n/a	
人数	attendee_num	整型	16	1	
实收金额	total_price	浮点型	32	0.0	精确小数点后一位
服务员 ID	servent_id	整型	32	n/a	非空
会员 ID	member_id	整型	32	n/a	n/a
收银员 ID	casher_id	整型	32	n/a	n/a
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	订单条目表		t_order_item		
数据项描述	数据项名	类型	长度	默认值	限制
订单条目 ID	id	整型	32	n/a	主键，自动增加
订单 ID	order_id	整型	32	0	
菜品 ID	dish_id	整型	32	0	非空
份数	dish_acc	整型	32	1	最大 99
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	用户表		t_user		
数据项描述	数据项名	类型	长度	默认值	限制
用户 ID	id	整型	32	n/a	主键，自动增加
用户名	username	字符型	16	n/a	非空，英文字母和数字组成不少于 6 位
密码	pass	字符型	128	n/a	非空，英文字母和数字组成不少于 6 位
手机号	cellphone	字符型	16		
会员等级 ID	level_id	整型	32		
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	角色表		t_role		
数据项描述	数据项名	类型	长度	默认值	限制
角色 ID	role_id	整型	32	n/a	主键
角色名称	role_name	字符串	32	n/a	非空
角色描述	role_desc	字符串	128	n/a	非空
角色状态	role_status	布尔类型	1	n/a	非空
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	用户角色关联表		t_user_role		
数据项描述	数据项名	类型	长度	默认值	限制
用户 ID	user_id	整型	32	n/a	主键
角色 ID	role_id	整型	32	n/a	主键
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	功能表		t_function		
数据项描述	数据项名	类型	长度	默认值	限制
功能 ID	function_id	整型	32	n/a	主键
功能名称	function_name	字符串	32	n/a	非空
功能描述	function_desc	字符串	128	n/a	非空
功能路径	function_path	字符串	128	n/a	非空
父功能 ID	function_parent	整型	32	n/a	可为空
功能排序	function_order	字符串	32	n/a	非空
功能状态	function_status	布尔类型	1	n/a	非空
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

数据表名	角色功能关联表		t_role_function		
数据项描述	数据项名	类型	长度	默认值	限制
功能 ID	function_id	整型	32	n/a	主键
角色 ID	role_id	整型	32	n/a	主键
创建时间	create_at	日期型		当前日期	非空
修改时间	update_at	日期型		当前日期	非空

9 附录