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 |

# 目录

[目录 2](#_Toc405244704)

[1 文档介绍 5](#_Toc405244705)

[1.1 文档目的 5](#_Toc405244706)

[1.2 文档范围 5](#_Toc405244707)

[1.3 缩写词列表 5](#_Toc405244708)

[1.4 参考内容 6](#_Toc405244709)

[2 系统范围 7](#_Toc405244710)

[2.1 业务流程图 7](#_Toc405244711)

[2.2 网络拓扑图 8](#_Toc405244712)

[2.3 系统架构图 8](#_Toc405244713)

[2.4 系统模块图 9](#_Toc405244714)

[2.5 系统数据ER图 9](#_Toc405244715)

[3 系统实现技术选型 10](#_Toc405244716)

[3.1 系统架构 10](#_Toc405244717)

[3.1.1 系统分层模型 - MVC 10](#_Toc405244718)

[3.1.2 系统分层模型实现 - SSH2 11](#_Toc405244719)

[3.2 Struts2 11](#_Toc405244720)

[3.2.1 概述 11](#_Toc405244721)

[3.2.2 优点与不足 12](#_Toc405244722)

[3.2.3 拦截器 12](#_Toc405244723)

[3.2.4 标签 12](#_Toc405244724)

[3.3 全功能栈JavaEE应用程序框架 - Spring 12](#_Toc405244725)

[3.3.1 概述 12](#_Toc405244726)

[3.3.2 控制反转 - IOC 13](#_Toc405244727)

[3.3.3 面向切面编程 13](#_Toc405244728)

[3.3.4 声明式事务管理 13](#_Toc405244729)

[3.3.5 安全 14](#_Toc405244730)

[3.4 数据持久层 - Hibernate 14](#_Toc405244731)

[3.4.1 概述 14](#_Toc405244732)

[3.4.2 好处与不足 14](#_Toc405244733)

[3.4.3 ORM关系对象模型 14](#_Toc405244734)

[3.5 Web前端技术 15](#_Toc405244735)

[3.5.1 jQuery 15](#_Toc405244736)

[3.5.2 Ajax技术 15](#_Toc405244737)

[3.6 Web Service 15](#_Toc405244738)

[3.6.1 概念 15](#_Toc405244739)

[3.6.2 RESTful 15](#_Toc405244740)

[3.6.3 JSON介绍 16](#_Toc405244741)

[3.6.4 实现Web Service的中间件介绍 16](#_Toc405244742)

[3.7 数据库技术 16](#_Toc405244743)

[3.7.1 概念 16](#_Toc405244744)

[3.7.2 MySQL 17](#_Toc405244745)

[3.8 应用服务器 17](#_Toc405244746)

[3.8.1 Apache HTTP服务器 17](#_Toc405244747)

[3.8.2 Tomcat应用服务器 18](#_Toc405244748)

[3.8.3 Tomcat 双机负载平衡 18](#_Toc405244749)

[3.8.4 HTTPS技术 18](#_Toc405244750)

[4 系统构建技术选型 19](#_Toc405244751)

[4.1 持续集成 19](#_Toc405244752)

[4.1.1 概念 19](#_Toc405244753)

[4.1.2 Maven 19](#_Toc405244754)

[4.2 测试驱动开发 19](#_Toc405244755)

[4.2.1 概念 19](#_Toc405244756)

[4.2.2 TDD 19](#_Toc405244757)

[5 系统概要设计 20](#_Toc405244758)

[5.1 开发环境 20](#_Toc405244759)

[5.2 运行平台 20](#_Toc405244760)

[5.3 异常处理 20](#_Toc405244761)

[5.3.1 异常信息封装 20](#_Toc405244762)

[5.3.2 异常处理 20](#_Toc405244763)

[5.4 日志管理 20](#_Toc405244764)

[5.4.1 日志的作用 20](#_Toc405244765)

[5.4.2 日志配置 20](#_Toc405244766)

[6 数据字典 20](#_Toc405244767)

[7 附录 24](#_Toc405244768)

# 文档介绍

## 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。

## 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。

## 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 |
|  |  |
|  |  |

## 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

# 系统范围

## 业务流程图



## 网络拓扑图



## 系统架构图



## 系统模块图



## 系统数据ER图



# 系统实现技术选型

## 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。

### 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。

### 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。

## Struts2

### 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:

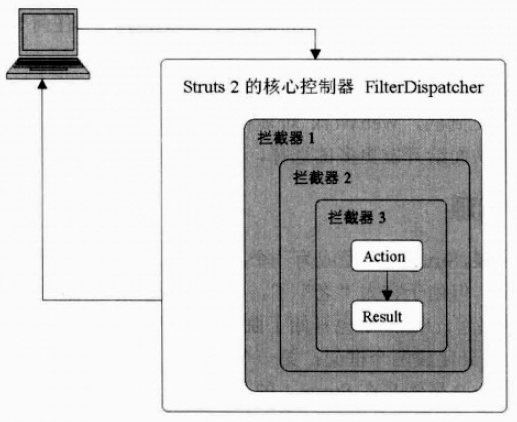


图3 - 1 Struts2 体系结构

Struts 2以WebWork为核心，采用拦截器的机制来处理用户的请求，这样的设计也使得业务逻辑控制器能够与Servlet API完全脱离开，所以Struts 2可以理解为WebWork的更新产品。虽然从Struts 1到Struts 2有着太大的变化，但是相对于WebWork，Struts 2只有很小的变化。

### 优点与不足

1.Struts2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。

2.使用OGNL进行参数传递。

3.强大的拦截器

4.易于测试

5.易于扩展的插件机制

6.模块化

7.全局结果与声明式异常

### 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Actiion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。

Struts2拦截器的实现原理相对简单，当请求Struts2的Action时，Struts 2会查找配置文件，并根据其配置实例化相对的 拦截器对象，然后串成一个列表，最后一个一个地调用列表中的拦截器。

### 标签

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。

## 全功能栈JavaEE应用程序框架 - Spring

### 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，使得程序的编写或者的最大的灵活度，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。

### 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。

### 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。

### 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。

### 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。

## 数据持久层 - Hibernate

### 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。

### 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

### 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。

## Web前端技术

### 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。

### 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。

## Web Service

### 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。

Web Service技术， 能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、 平台或内部协议是什么， 都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署， 因为它们基于一些常规的产业标准以及已有的一些技术，诸如标准通用标记语言下的子集XML、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

### RESTful

一种软件架构风格，设计风格而不是标准，只是提供了一组设计原则和约束条件。它主要用于客户端和服务器交互类的软件。基于这个风格设计的软件可以更简洁，更有层次，更易于实现缓存等机制。

RESTful指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个唯一的地址。所有资源都共享统一的接口，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。

### 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)。

### 实现Web Service的中间件介绍

JSR 311 或 JAX-RS（用于 RESTful Web Services 的 Java API）的目的是提供一组 API 以简化 REST 样式的 Web 服务的开发。

在 JAX-RS 规范之前，已经有 Restlet 和 RestEasy 之类的框架，可以帮助您实现 RESTful Web 服务，但是它们不够直观。Jersey 是 JAX-RS 的参考实现，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 还提供可以轻松集成 Spring、Guice、Apache Abdera 的库。

## 数据库技术

### 概念

数据库是依照某种数据模型组织起来并存放二级存储器中的数据集合。这种数据集合具有如下特点：尽可能不重复，以最优方式为某个特定组织的多种应用服务，其数据结构独立于使用它的应用程序，对数据的增、删、改、查由统一软件进行管理和控制。

主要特点

⑴ 实现数据共享

数据共享包含所有用户可同时存取数据库中的数据，也包括用户可以用各种方式通过接口使用数据库，并提供数据共享。

⑵ 减少数据的冗余度

同文件系统相比，由于数据库实现了数据共享，从而避免了用户各自建立应用文件。减少了大量重复数据，减少了数据冗余，维护了数据的一致性。

⑶ 数据的独立性

数据的独立性包括逻辑独立性（数据库中数据库的逻辑结构和应用程序相互独立）和物理独立性（数据物理结构的变化不影响数据的逻辑结构）。

⑷ 数据实现集中控制

文件管理方式中，数据处于一种分散的状态，不同的用户或同一用户在不同处理中其文件之间毫无关系。利用数据库可对数据进行集中控制和管理，并通过数据模型表示各种数据的组织以及数据间的联系。

⑸数据一致性和可维护性，以确保数据的安全性和可靠性

主要包括：①安全性控制：以防止数据丢失、错误更新和越权使用；②完整性控制：保证数据的正确性、有效性和相容性；③并发控制：使在同一时间周期内，允许对数据实现多路存取，又能防止用户之间的不正常交互作用。

⑹ 故障恢复

由数据库管理系统提供一套方法，可及时发现故障和修复故障，从而防止数据被破坏。数据库系统能尽快恢复数据库系统运行时出现的故障，可能是物理上或是逻辑上的错误。比如对系统的误操作造成的数据错误等。

### 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。

## 应用服务器

### 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。

### Tomcat应用服务器

Tomcat是Apache 软件基金会的Jakarta 项目中的一个核心项目，由Apache、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现，Tomcat 5以及后续版本支持最新的Servlet 2.4 和JSP 2.0 规范。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 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。

### 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。

### 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。

# 系统构建技术选型

## 持续集成

### 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。

### 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。

## 测试驱动开发

### 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。

### 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。

# 系统概要设计

## 开发环境

JDK 1.7+

Windows 7+

Tomcat 6+

MySQL 5+

## 运行平台

Linux

## 异常处理

### 异常信息封装

建一个异常信息封装类，将项目中常见异常信息进行封装，常见信息包括：

1.异常种类：

如： 数据库操作异常； 应用操作异常（业务异常等等） ； 系统异常（如 应用服务器异常）

2.具体异常：

如："该数据库不存在！" 等等

### 异常处理

在持久层或业务逻辑层捕获处理，然后层层向上抛，直到Action层，在Action进行捕获和处理。

## 日志管理

### 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。

### 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。

# 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 | user\_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 | 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 | user\_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 | 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 |

# 附录