库存管理系统

软件构架文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2016-05-30 | 1.0 | 初始创建 | 徐元明 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 构架表示方式 4

3. 构架目标和约束 4

4. 用例视图 4

4.1 用例实现 4

5. 逻辑视图 4

5.1 概述 5

5.2 在构架方面具有重要意义的设计包 5

6. 进程视图 5

7. 部署视图 5

8. 实施视图 5

8.1 概述 5

8.2 层 5

9. 数据视图（可选） 5

10. 大小和性能 5

11. 质量 5

软件构架文档

# 简介

MVC它是一种设计模式，它把应用程序分成三个核心模块：模型、视图、控制器，它们各自处理自己的任务。

MVC（Model-View-Controller）应用程序结构被用来分析分布式应用程序的特征。这种抽象结构能有助于将应用程序分割成若干逻辑部件，使程序设计变得更加容易。   
  
MVC结构提供了一种按功能对各种对象进行分割的方法（这些对象是用来维护和表现数据的），其目的是为了将各对象间的耦合程度减至最小。MVC结构本来是为了将传统的输入（input）、处理（processing）、输出（output）任务运用到图形化用户交互模型中而设计的。但是，将这些概念运用于基于Web的企业级多层应用领域也是很适合的。   
  
在MVC结构中，模型（Model）代表应用程序的数据（data）和用于控制访问和修改这些数据的业务规则（business rule）。通常模型被用来作为对现实世界中一个处理过程的软件近似，当定义一个模型时，可以采用一般的简单的建模技术。   
  
当模型发生改变时，它会通知视图（View），并且为视提供查询模型相关状态的能力。同时，它也为控制器（Controller）提供访问封装在模型内部的应用程序功能的能力。   
  
一个视图（View）用来组织模型的内容。它从模型那里获得数据并指定这些数据如何表现。当模型变化时，视负责维持数据表现的一致性。视同时将用户要求告知控制器（Controller）。   
  
控制器（Controller）定义了应用程序的行为；它负责对来自视的用户要求进行解释，并把这些要求映射成相应的行为，这些行为由模型负责实现。在独立运行的GUI客户端，用户要求可能是一些鼠标单击或是菜单选择操作。在一个Web应用程序中，它们的表现形式可能是一些来自客户端的GET或POST的HTTP请求。模型所实现的行为包括处理业务和修改模型的状态。根据用户要求和模型行为的结果，控制器选择一个视作为对用户请求的应答。通常一组相关功能集对应一个控制器。

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

本文的目的为使软件开发人员更好的理解系统，并指导软件开发工作。

## 范围

**本文适用于架构师，软件设计师**

## 定义、首字母缩写词和缩略语

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

## 参考资料

参考SRS文档

## 概述

本文包含了架构建模，用例图，ACME组件连接器视图，部署图，数据库设计图

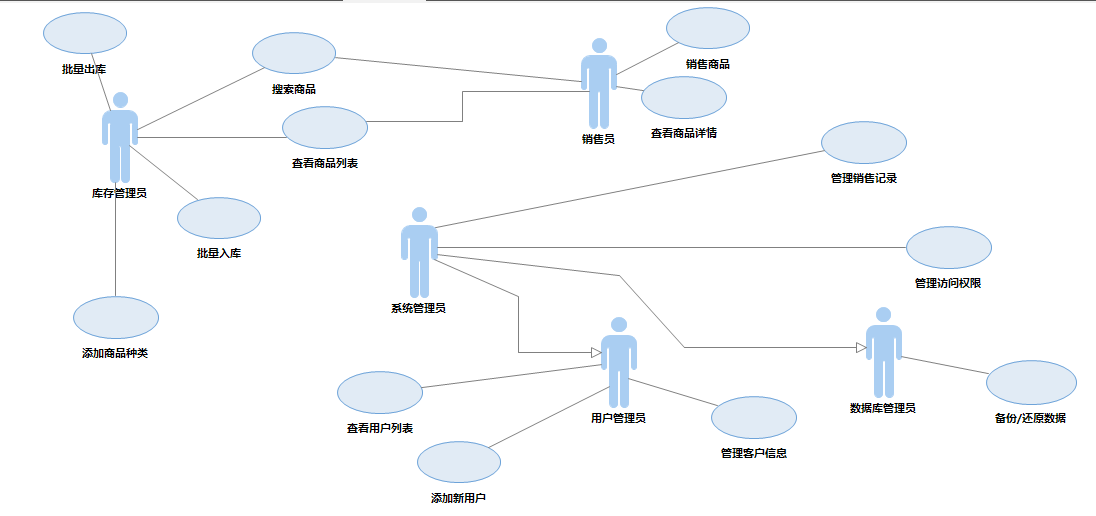
# 构架表示方式

当前系统使用用例图，ACME组件连接器视图，部署图，数据库建模图。

# 构架目标和约束

本架构使用MVC结构

# 用例视图



## 用例实现

库存管理系统需实现以下功能：

1. 查看商品列表，优先高亮显示将缺货的商品

2. 添加商品种类

3. 按商品名称、商品别名搜索商品

4. 查看商品详情

5. 批量出库

6. 批量入库

7. 查看、修改、删除销售记录

8. 标记销售单已收条/已开发票/已收款/已审计

9. 按时间/商品名称排序显示商品销售记录

10.按时间段（当天、本周、本月、本季度、本年、自定义）查询销售记录

11. 添加新客户

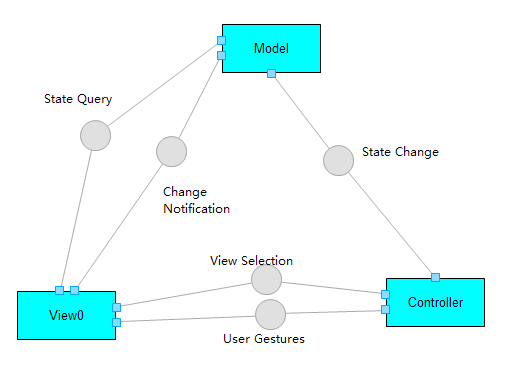
12. 查看客户列表

13.查询、修改、删除客户信息

14.访问控制

15.数据库导入导出

# 逻辑视图



## 概述

[本节按照设计模型中包的层次结构来说明设计模型的整体分解情况。]

## 在构架方面具有重要意义的设计包

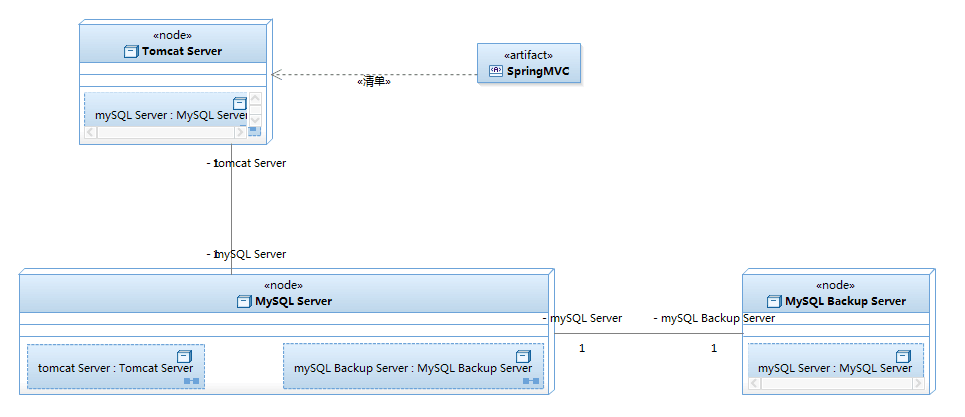
[对于每个重要的包，都用一个小节来加以说明，其中应包括该包的名称、简要说明以及显示该包中所有重要的类和包的图。

对于该包中的每个重要类，应包括其名称、简要说明，还可选择包括对其部分主要职责、操作和属性的说明。]

# 进程视图

[本节说明将系统分解为轻量级进程（单个控制线程）和重量级进程（成组的轻量级进程）的情况。本节的内容按照各个通信或交互的进程组来进行组织。说明进程之间的主要通信模式，例如消息传递、中断和会合。]

# 部署视图



# 实施视图

[本节说明实施模型的整体结构、软件分解为实施模型中的层和子系统的情况，以及所有在构架方面具有重要意义的构件。]

## 概述

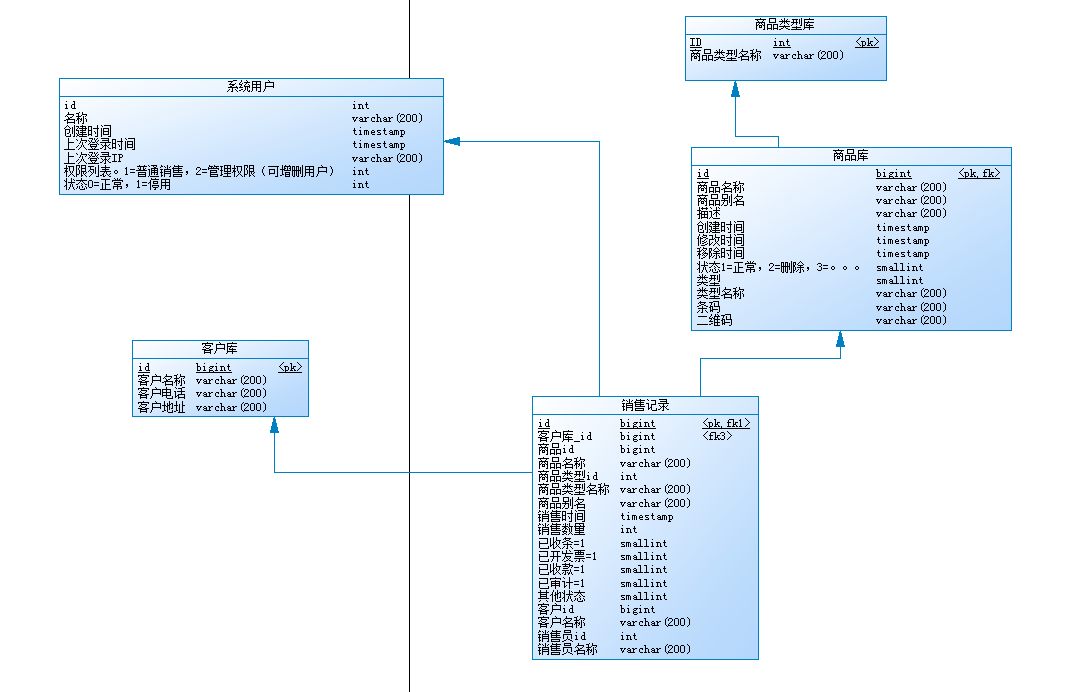
[本小节指定并定义各个层及其内容、添加到指定层时要遵循的规则以及各层之间的边界。还应包括一个显示层间关系的构件图。 ]

## 层

[对于每个层，都用一个小节来加以说明，其中包括该层的名称和一个构件图，并列举位于该层的子系统。]

# 数据视图（可选）

[从永久性数据存储方面来对系统进行说明。如果几乎或根本没有永久性数据，或者设计模型与数据模型之间的转换并不重要，那么本节就为可选。]



# 大小和性能

[说明软件中会对构架产生影响的主要尺寸特征，以及目标性能约束。]

# 质量

[说明软件构架如何促成诸如可扩展性、可靠性、可移植性等所有系统能力（而非功能）的实现。如果这些特征具有特殊的意义（例如在安全性或保密性方面的意义），则应该对它们进行详细的说明。]