库存管理系统

软件构架文档

版本 <1.0>

[注：以下提供的模板用于 Rational Unified Process。其中包括用方括号括起来并以蓝色斜体（样式=InfoBlue）显示的文本，它们用于向作者提供指导，在发布此文档之前应该将其删除。按此样式输入的段落将被自动设置为普通样式（样式=Body Text）。]

[要定制 Microsoft Word 中的自动字段（选中时显示灰色背景），请选择 File>Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit>Select All（或 Ctrl-A）并按 F9，或只是在字段上单击并按 F9，可以在整个文档中更新自动字段。对于页眉和页脚，这一操作必须单独进行。按 Alt-F9，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。]

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

[**软件构架文档**的简介应提供整个**软件构架文档**的概述。它应包括此**软件构架文档**的目的、范围、定义、首字母缩写词、缩略语、参考资料和概述。]

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。

[本节确定此**软件构架文档**在整个项目文档中的作用或目的，并对此文档的结构进行简要说明。应确定此文档的特定读者，并指出他们应该如何使用此文档。]

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

[简要说明此软件构架文档适用的范围和影响的范围。]

## 定义、首字母缩写词和缩略语

[本小节应提供正确理解此**软件构架文档**所需的全部术语的定义、首字母缩写词和缩略语。 这些信息可以通过引用项目词汇表来提供。]

## 参考资料

[本小节应完整地列出此**软件构架文档**中其他部分所引用的所有文档。每个文档应标有标题、报告号（如果适用）、日期和出版单位。列出可从中获取这些参考资料的来源。这些信息可以通过引用附录或其他文档来提供。]

## 概述

[本小节应说明此**软件构架文档**中其他部分所包含的内容，并解释此**软件构架文档**的组织方式。]

# 构架表示方式

[本节说明当前系统所使用的软件构架及其表示方式。还会从**用例视图**、**逻辑视图**、**进程视图**、**部署视图**和**实施视图**中列出必需的那些视图，并分别说明这些视图包含哪些类型的模型元素。]

# 构架目标和约束

[本节说明对构架具有某种重要影响的软件需求和目标，例如：安全性、保密性、市售产品的使用、可移植性、分销和重复使用。还应记录可能适用的特殊约束：设计与实施策略、开发工具、团队结构、时间表、遗留代码等。]

# 用例视图

[本节列出用例模型中的一些用例或场景，这些用例或场景应体现最终系统中重要的、核心的功能；或是在构架方面涉及范围很广（使用了许多构架元素）；或强调或阐明了构架的某一具体的细微之处。]

## 用例实现

[本节通过几个精选的用例（场景）实现来阐述软件的实际工作方式，并解释不同的设计模型元素如何促成其功能的实现。]

# 逻辑视图

[本节说明设计模型在构架方面具有重要意义的部分，例如设计模型被分解为多个子系统和包。而每个重要的包又被分解为多个类和类实用程序。您应该介绍那些在构架方面具有重要意义的类，并说明它们的职责，以及几项非常重要的关系、操作和属性。]

## 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

[本节说明用来部署和运行该软件的一种或多种物理网络（硬件）配置。对于每种配置，它至少应该指出执行该软件的物理节点（计算机、CPU）及其互连情况（总线连接、LAN 连接、点到点连接等）。另外还要包括**进程视图**中的各进程到物理节点的映射。]

# 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。]