|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实训X班X组 | 文 档 编 号 | | 产品版本 | 密级 |
| 文件状态：  [ ] 草 稿  [ √ ] 正式发布  [ ] 正在修改 | JXPBIT200807 | | 1.0 | 内部 |
| 项目名称 | XX项目 | | 共X页 |

XXX项目

架构设计报告

**文档作者： XXX小组**

**项目组长： XXXX**

**批 准 人： XX**

**批准日期：XXXX年X月X日**

**国家发展改革委-微软软件创新中心IT人才实训基地**

Add: 江西省南昌高新区京东大道698号 浙大科技园 创业大厦D区

Tel: 0791-8188888

Fax: 0791-8186856 Code: 330096

http:// [www.jxmstc.com](http://www.jxmstc.com)

**版本状态**

| **版本** | **作者** | **参与者** | **起止日期** | **注释** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

**审阅者**

| **团队** | **版本** | **日期** | **签名** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**分发**

| **姓名** | **位置** | **分发方式** | **分发日期** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 引言 1](#_Toc361406599)

[1.1 编写目的 1](#_Toc361406600)

[1.2 编写约定 1](#_Toc361406601)

[1.3 读者对象 1](#_Toc361406602)

[1.4 术语解释 1](#_Toc361406603)

[1.5 参考资料 1](#_Toc361406604)

[2 架构设计思想 2](#_Toc361406605)

[2.1 架构原则 2](#_Toc361406606)

[2.2 实现策略 2](#_Toc361406607)

[2.3 架构模式 2](#_Toc361406608)

[3 用例视图 2](#_Toc361406609)

[4 逻辑视图 2](#_Toc361406610)

[4.1 概述 2](#_Toc361406611)

[4.2 逻辑架构 3](#_Toc361406612)

[5 开发视图 3](#_Toc361406613)

[6 物理视图 3](#_Toc361406614)

[7 进程视图 3](#_Toc361406615)

[8 数据存储设计 3](#_Toc361406616)

[9 性能设计 3](#_Toc361406617)

[10 可用（可靠）性和容错性设计 4](#_Toc361406618)

# 引言

## 编写目的

本文编写目的在于规定和控制项目开发的内容，保证本项目的需求分析活动在受控状态下进行，在进行软件开发前，明确本系统应达到的目标，对系统目标做出完整、准确、清晰、具体的要求。保证本需求分析的结果能够完整、无遗漏地反映待开发系统的要求。

## 编写约定

详见《工程文档格式编写规范要求》：

## 读者对象

*提示：阅读此文档的读者对象有哪些*。

## 术语解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
| [*MDA*](http://www.itisedu.com/phrase/200603051312555.html) | *(Model Driven Architecture)是模型驱动架构，它是由OMG定义的一个软件开发*[*框架*](http://www.itisedu.com/phrase/200603061723295.html)*。它是一种基于UML以及其他工业标准的框架，支持软件设计和模型的可视化、存储和交换。和UML相比，MDA能够创建出机器可读和高度抽象的模型，这些模型独立于实现技术，以标准化的方式储存。* |
|  |  |
|  |  |
|  |  |
| … |  |

## 参考资料

*提示：编写需求分析报告的输入产物或者参考的相关资料*

# 架构设计思想

*提示：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。*

## 架构原则

*提示：本小节描述设计架构的原则比如安全性设计原则等;*

## 实现策略

*提示：架构构件的实现遵循如下原则：*

*1．复用优先于购买，购买优先于编码实现；*

*2．使用可靠的框架，提高开发效率以及稳定程度*；

## 架构模式

*提示：本小节简介和约定本系统架构设计中主要使用的架构和设计模式。*

*结构模式：分层；分布式系统：代理、外观、服务模式；交互系统：ＭＶＣ*

# 用例视图

*提示：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）；主要影响架构的需求有哪些？详述关键的重要功能。*

# 逻辑视图

*提示：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

*提示：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。*

## 逻辑架构

*提示：简要介绍本系统采用的逻辑架构比如逻辑分层方式及各层之间的关系、功能分配等。*

# 开发视图

*提示：开发视图关注程序包，不仅包括要编写的源程序，还包括可以直接使用的第三方SDK和现成框架、类库，以及开发的系统将运行于其上的系统软件或中间件。开发视图和逻辑视图之间可能存在一定的映射关系：比如逻辑层一般会映射到多个程序包等、关键实现等;*

# 物理视图

*提示：物理视图：和部署相关的架构决策*

*软件最终要驻留、安装或部署到硬件才能运行，而软件架构的物理视图关注"目标程序及其依赖的运行库和系统软件"最终如何安装或部署到物理机器，以及 如何部署机器和网络来配合软件系统的可靠性、可伸缩性等要求。*

# 进程视图

*提示：此项内容如本项目未涉及，可以不填写。处理视图关注进程、线程、对象等运行时概念，以及相关的并发、同步、通信等问题。处理视图和开发视图的关系：开发视图一般偏重程序包在编译时期的静态依赖关系，而这些程序运行起来之后会表现为对象、线程、进程，处理视图比较关注的正是这些运行时单元的交互问题。*

# 数据存储设计

*提示：此项内容如本项目未涉及，可以不填写。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。*

# 性能设计

*提示：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。比如：1表示层性能设计考虑、应用层性能设计考虑、数据层性能设计考虑*

*在数据库设计方面，提供的架构级服务包括：*

*针对焦点表，进行表分割，按业务特性划分活动表和非活动表，提高OLTP（针对活动表的处理）效率；优化存取路径，检查索引效率；*

# 可用（可靠）性和容错性设计

*提示：对可靠性和容错性方面进行考虑的方面。*