## 在线考试系统 需求说明书

## 版本信息

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： |  |
| 作 者： |  |
| 完成日期： |  |

广州小马信息科技有限公司

## 1 引言

### 1.1 编写目的

本文档的编写目的是果在线考试系统的开发提供：

* 软件总体要求，作为用户和软件开发人员之间了解的基础；
* 功能、性能和接口的要求，作为软件人员进行设计和编码的基础
* 验收标准，作为用户确认测试的依据。

### 1.2 文档约定

本需求规格说明书的主要内容有：介绍产品的背景，概述产品的功能需求、环境需求、性能需求、平台角色等内容。

\*文档提供了功能需求，功能模块，需要完成的文档还有：数据库设计，页面原型设计，详细功能清单，接口文档。

### 1.3 读者对象

该需求规格说明书提供给最终用户、研发人员、测试人员等相关人员阅读。

## 

## 系统主要概述

### 2.1前端主要功能模块

1. 登录，注册  
   2.个人中心
2. 用户答题

### 2.1后台主要功能模块

1. 基础管理 —— 后台用户管理，角色管理，菜单管理，权限管理，消息管理，字典管理，（区域管理）
2. 普通用户管理
3. 考题管理（考试主题管理和题目管理）
4. 成绩管理
5. 前端功能接口

## 3.主要功能需求描述

### 3.1登录，注册

1.用户通过验证邮箱进行注册登录,邮箱输入采用后缀免输入的设计方式。

### 3.2个人中心

1.显示和修改个人信息，包括用户头像、个人中心背景图、姓名、性别、手机号码、邮箱等。

2.显示用户个人成长曲线（成绩）。

3.修改密码。

### 3.3答题

1.主题选择。

2.主题详情描述（图文结合）。

3.题目列表，包括题目序号、考题、排序、得分情况。

4.题目详情,提交答题。

5.主题答题统计,显示总答题数、答题进度和总得分。

6.匹配功能，答完最后一道题，最后出现一个成绩，通过成绩筛选成绩最相近的一个用户，出现画面说明他的成绩与你的成绩最相近，还有一个击败百分之多少的用户的结果呈现。

### 4 建设原则和技术线路

### 4.1 性能需求

系统设计遵循以下原则：

* + 先进性

系统采用多层结构设计，采取XML技术进行数据的存储、传输。

* + 经济性

避免重复建设，力争节约资金，对现有的办公网络充分利用、发挥效益。

* + 扩展性

系统充分考虑今后功能扩展需要，预留了各种扩展接口。

* + 简单性

系统功能完善，但操作维护便利，界面友好，尤其对单位系统管理员而言，无须复杂的技术培训和繁琐的编程，即可对应用流程进行调整和维护。

* + 互联性

该系统可以与其它信息系统实现无缝联接，可以与各种关系型数据库平滑联接。

* + 可靠性

由于系统的特定用户群，在系统平台上将保证系统的高保密性、高安全性和高可靠性。系统设计中，采用可靠成熟的技术和平台产品，数据有适量冗余及其他保护措施，平台和应用软件具有容错性、健壮性等。

* + 安全性

系统采用多重安全措施确保系统多层次的安全需求。

### 4.2 技术线路

MVC开发模式

严格采用“模型-视图-控制器”（Model-View-Controller）开发模式，把业务逻辑封装在Model层中，显示逻辑放在View层中,而在View层中所能访问Model的信息完全由控制层（Controller）控制。Model提供统一的对外接口，而不管View端是基于什么样的开发语言。

MVC是一种多种设计模式组合而成的设计模式，指的是一种划分系统功能的方法，它是Model-View-Controller的缩写，最早是由Xerox（施乐）在20世纪80年代为Smaltalk-80语言发展提出的。随后，它成为了一种著名的用户界面设计架构。

在MVC经典架构中，它把应用程序（也可看做是应用程序的一个模块）分为3个部分：

**1．模型（Model）**

模型代表了该应用程序的核心功能，它负责处理用户的数据，实现业务逻辑。同时，模型还为视图的显示提供数据，并可被多个视图所共用。

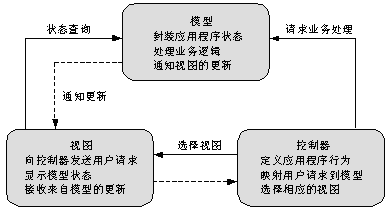
**2．视图（View）**

视图主要指与用户交互的界面，也即应用程序的外观。视图可以接收用户的输入，但它并不包括任何实际的业务处理，它只是将数据转交给控制器。同时，视图还负责展现模型传递给用户的数据，当后台模型更新数据时，视图也应当随之更新它的显示。

**3．控制器（Control）**

控制器负责接收用户的请求和数据，接着它会做出判断将请求和数据交由哪一个模型来处理，最后调用视图来显示模型返回的数据。

它们之间的关系如下图所示。

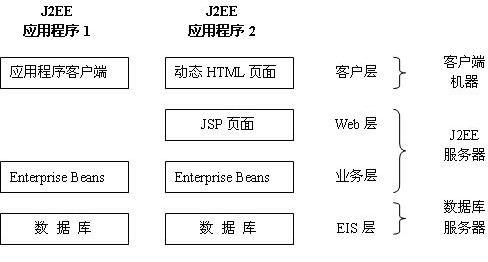


基于J2EE架构

充分利用JAVA语言的跨平台性，同时遵照SUN公司的J2EE架构标准。

J2EE是一个开放的、基于标准的平台，可以开发、部署和管理N层结构的、面向Web的、以服务器为中心的企业级应用，它是利用Java 2 平台来简化与多级企业解决方案的开发、部署和管理相关的诸多复杂问题的应用体系结构。

J2EE平台采用一个多层次分布式的应用模式。这意味着应用逻辑根据功能被划分成组件，组成J2EE应用的不同应用组件安装在不同的服务器上，这种划分是根据应用组件属于多层次J2EE环境中的哪一个层次来决定的。J2EE应用可以由三或四个层次组成，J2EE多层次应用一般被认为是三层应用，因为它们是被分布在三个不同的地点：客户端机器、J2EE服务器和数据库或后端的传统系统服务器。三层架构应用是对标准的客户端/服务器应用架构的一种扩展，即在客户端应用和后台存储之间增加一个多线程应用服务器。



J2EE多层应用

J2EE体系包括JSP、Servlet、EJB、WEB SERVICE等多项技术。这些技术的出现给电子商务时代的WEB应用开发提供了一个非常有竞争力的选择。怎样把这些技术组合起来，形成一个适应项目需要的稳定架构是项目开发过程中一个非常重要的步骤。

一个成功的软件需要有一个成功的架构，但软件架构的建立是一个复杂而又持续改进的过程，软件开发者们不可能对每个不同的项目做不同的架构，而总是尽量重用以前的架构，或开发出尽量通用的架构方案，Struts就是流行的基于J2EE的架构方案之一，其他常用的基于J2EE的架构方案还有Turbine、RealMothods等。

技术参考资料：

[Maven学习](http://www.cnblogs.com/izecsonLee/p/6155868.html)

[Spring-boot学习](http://blog.csdn.net/lxhjh/article/details/51711148)

[Thymeleaf模板使用](http://schy-hqh.iteye.com/blog/1961397)