# 设备及巡检系统架构设计说明书

版本号：V0.1

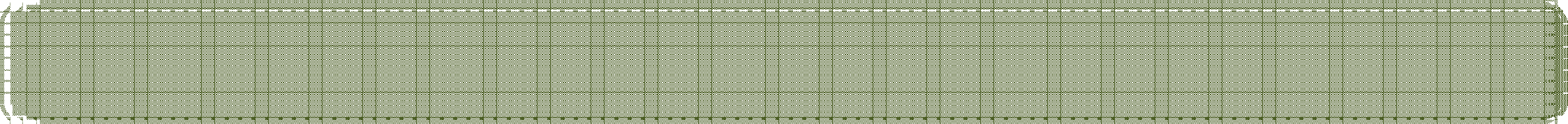
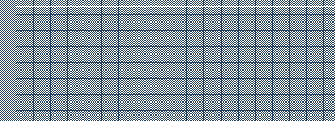
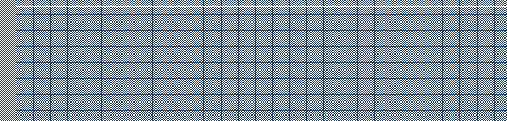
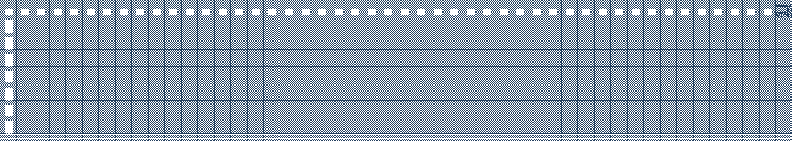
2015 年 11 月

## 1. 目的

本说明书的编写目的是描述系统的架构设计方案，包括系统的软件总体架构 设计及使用的框架说明，以及基于该架构的开发流程，并作为指导开发人员、测 试人员进行系统开发及测试的依据。

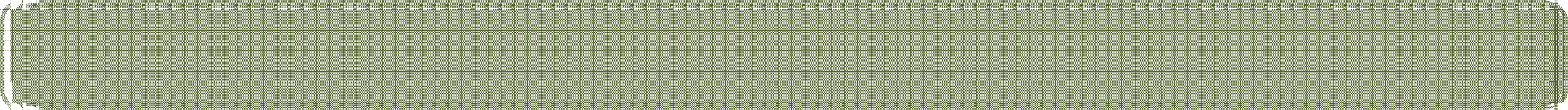
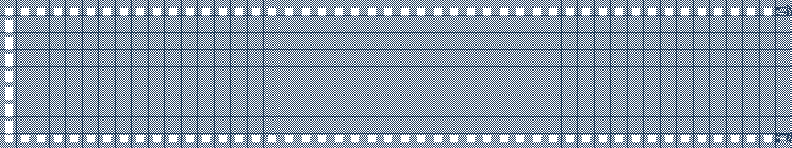
## 2.系统架构设计

整个软件架构方案采用分层、分布式的部署结构，明确地分离了表现层和业 务逻辑，能够保证应用服务逻辑的一致性和稳定性、结构的开放性、功能的可扩 展性和可维护性、开发的可并行性，同时采用一些开源的框架，兼顾了经济性。 框架是一种特殊的软件，它为软件开发带来了高度的重用性，是无数软件开发人 员的多年项目开发经验的总结。在一个优秀的框架上开发应用，而不是从零开始， 可以大量缩短项目的开发周期、降低开发风险、增强应用系统的稳定性。

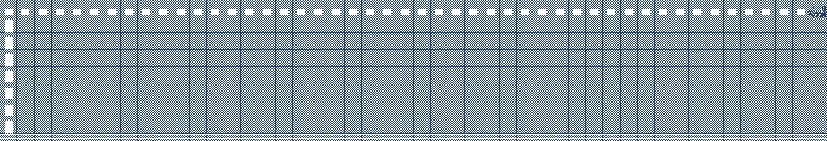


用户层 平板 客户端 WEB 浏览器（IE）

公共模块



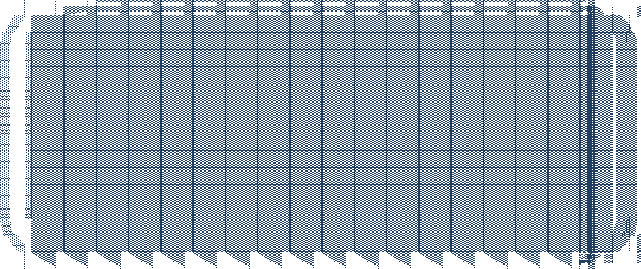
表示层 View



设备模块

WEB 应用 服务器

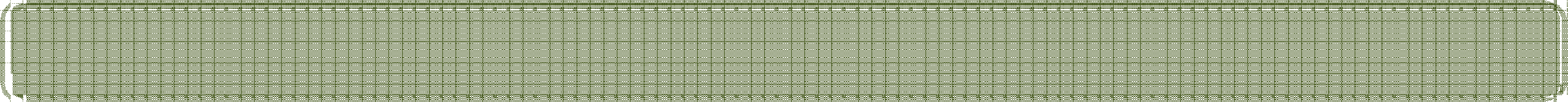
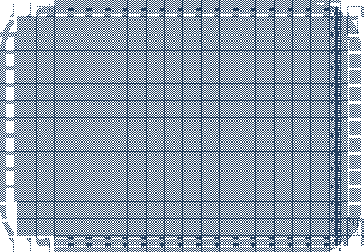
业务层 Controller



巡检模块

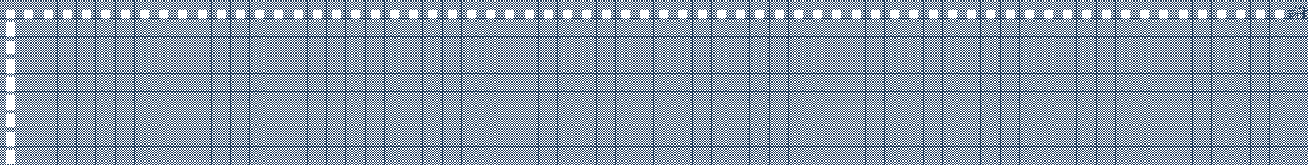
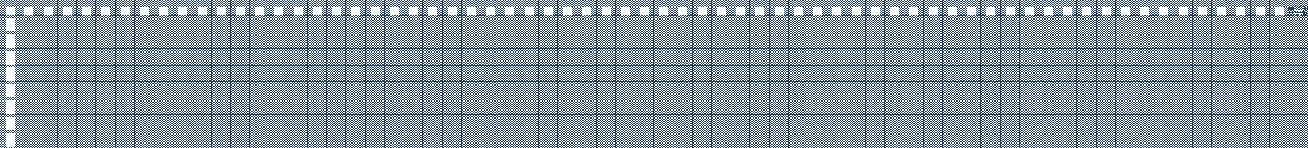
数据访问层 Model

OCI OCI



数据库

ORACLE

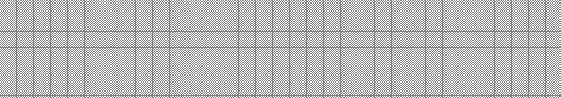
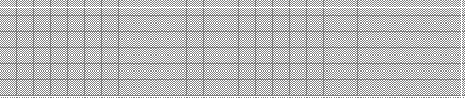


操作系统 LINUX

系统总体架构图

系统总体架构如上图所示，按功能可以分为 公共管理、设备管理、巡检管理、机房出入等模块；系统根据功能特点与业务需求采用基于web的B/S架构模式，项目基于PHP代码实现，运用CI开源框架，可以运行于 Linux 和 Windows 平台；

## 3. WEB系统架构设计



用户层 Web 浏览器（IE、Google）

Web 服务器

Nginx Server

表

示

层 html Json View

DTO DTO

业

务 PHP Controller

层

DTO DTO

DAO iBatis

数据访问层

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | Model |  |
|  | |  |

|  |  |
| --- | --- |
|  |  |
| ORM |  |
|  | |

OCI OCI

数据库

Oracle

操作系统 RedHat Linux

从架构图中可以看出系统分为四层：

用户层：浏览器

表示层：借助 html框架及json数据实现

业务层：借助 PHP进行业务流程开发。

 数据持久层：ORM模型操作数据库

为什么采用这样的四层架构？

 通过成熟的开源产品实现各层，同自己编写代码实现，相比之下能缩短开发周期， 且架构所用到的开源产品均有很广泛的用户群，经受过实践的考验，质量和性能 更有保障。

 层与层之间松散耦合，增加代码重用率。

 各层分工明确，这样也利于团队的明确分工。 系统的总体架构从结构上分为用户层、表示层、业务层、数据访问层以及在层间传递数 据的数据传输对象。下面针对各层加以描述。

1)．用户层

用户层作为客户端程序，用来与用户交互，并把来自系统的信息显示给用户。

系统的用户层采用的是 IE 浏览器作为交互方式。

2）．表示层 表示层主要控制页面外观，产生页面逻辑以及对用户输入的数据进行合法性验证。

系统中主要包括基于Bootstrap框架的 JavaScript 脚本及基于CI框架的 表单验证。其中 JavaScript 脚本可以增强用户体验，PHP负责视图的功能，由 HTML、 PHP程序片断和 JSON 数据构成。

3）．业务层

业务层处理应用的核心业务逻辑。业务逻辑对象把业务 规则、约束、活动和数据结合在一起，Controller负责对这些业务对象的管理。

4）．数据访问层

数据访问对象把底层的数据访问操作和上层的商务逻辑分开。

CI框架规范定义了 Web 应用程序的类和文件存放的目录结构。该层次结构由三个层 次构成。第一层是上下文，它是一个目录或者是多个目录，用来查找与客户请求关联的 Web 应用程序。在上下文中存在一个/Application目录，该目录存放应用程序，它包 含MVC三层目录，用来实现系统的业务逻辑。/public 包含所有样式文件，js文件和图片。CI框架采用但入口方式访问系统，即入口文件只有一个，通过参数控制（c=控制器名&m=方法名）访问不同控制器的方法以打开不同的页面。

表 1-1 系统目录结构表

|  |  |  |  |
| --- | --- | --- | --- |
| 目录名 | | | 说明 |
| /data\_center |  |  | 应用上下文根目录 |
|  | /public |  | Css 样式表文件,js文件和图片等目录 |
|  | /uploads |  | 上传的文件目录 |
|  | /system |  | 框架系统目录 |
|  | /downloads |  | 可下载的文件存放目录 |
|  | /Application |  |  |
|  |  | /config | 系统配置文件目录 |
|  |  | /controllers | 业务逻辑层目录（控制器） |
|  |  | /models | 模型层目录（模型） |
|  |  | /views | 视图层目录（视图） |
|  | index.php |  | 系统入口文件 |

controllers目录下的系统PHP文件的结构说明见表 1-2。

表 1-2 系统PHP文件结构说明表

|  |  |
| --- | --- |
| 包名 | 作用说明 |
| com\_\*.php | 系统管理模块相关控制器 |
| sys\_\*.php | 公共管理模块相关控制器 |
| equipment\_\*.php | 设备模块相关控制器 |
| interface\_\*.php | 接口模块相关控制器 |
| check\_\*.php | 巡检模块控制器 |
| room\_\*.php | 机房出入模块控制器 |

## 4. WEB子系统架构总体功能设计

4.1 CI框 架

CI 是一个实现了 MVC 模式的框架，对 Model、View 和 Controller 都提供了对应 的实现组件。如下图所示：

CI 入口文件

Brower

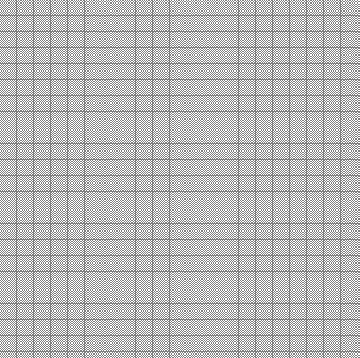
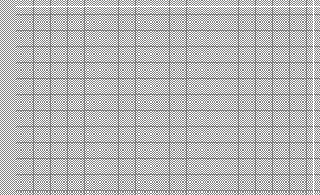
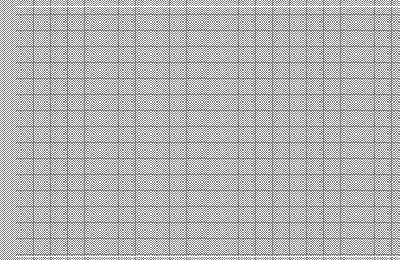
**Controller**

**Model**

Action

**View**

1．控制器（Controller）



控制器的作用是从客户端接受请求，并且通过模型层和和数据库交互，然后选择执行相应的业务逻辑，然后把响应结果返回到客户端（视图层）。

2．模型（Model）

MVC 系统中的 Model 部分从概念上可以分为两类--系统的内部状态，和改变系统 状态的动作。CI 为 Model 部分提供了 default 对象：所有的 Action 处 理器对象都是开发者从 Struts 的 Action 类派生的子类。Action 处理器对象封装了具体的

处理逻辑，调用业务逻辑模块，并且把响应提交到合适的 View 组件以产生响应。

1. 视图（View）

View通过html+php技术实现，view层提供了丰富的组件库，例如下拉框，列表、翻页等，可以非常好的和系统的controller交互，通过这些标签组件实现数据的录入和展现。

4.2 表 单 验 证

对 Web 应用来说，由于用户的行为是无法预测的，在表单数据在传递给业务类之前， 必须保证数据的合法性及有效性，而表单验证是保证数据合法性及有效性的重要手段。 对于基于 CI框架的表单验证一般有两种方式：基于 JavaScript 的表单验证（前端验证） 及基于 php 的表单验证（后端验证）。

前端验证：

可以针对基本类型（汉字、英文、整型、数字、日期、邮编）的有效性及合法性开发出一些共同 JavaScript 函数，以备整个系统统一调用。

后端验证：

主要采用基于 CI form组件 的表单验证。 由于前端验证不需要提交服务器，直接在客户端完成，从而减少了服务器的压力，

所以我们优先采用前端验证。

4.3 日 志 处 理

规范合理的日志记录能让开发人员和维护人员事半功倍，在记录日志时主要考虑对数据的一些敏感操作，如增加和删除修改等，并且日志易读。

CI作为一个开源的强大的php框架，有自己的日志处理方式，运行日志主要保存在/home/wwwroot/default/data\_center/application/logs

在application/config/config.php中设置$config['log\_threshold'] = 1/2/3/4，如果为0表示不输出错误日志，可自定义日志输出的级别；

另外，为了方便维护，系统还开发了操作日志记录功能（调用setLog（‘日志内容’）方法），用中文的方式记录主要的操作信息，操作日志目录在 /appxj/data\_cenetr/logs

5. 共同组件

5.1 分 页 组 件

数据列表是页面的一种常见页面表现形式，而分页功能又是列表的一个不可缺少 的功能，分页组件主要为各种形式的数据列表提供一个共同的分页功能。技术上实现为 首先查询出总的符合条件的数据记录数，然后每次只查询出一页的数据记录，这种方法对于大数据量的情况性能很好。本系统页面表现基于 Ajax 技术，实现 无刷新效果，组件调用接口要求简洁明了。分页组件主要通过前台js+ajax实现，后台控制器只需按ajax提交的参数返回总的条数和对应页面的数据即可，此处用的组件是基于jquery的datatable组件。

5.2 树 形 组 件

树形结构一般用于组织机构等具有层次结构的数据，也是页面常见的表现形式，树 形组件为各种形式的层次结构数据提供一个共同的分页功能。技术上可以分两种实现方 法：一是查询出所有的层次节点的符合条件的数据记录，在 html 页面进行分层显示。考虑到层次结构数据一般数据量不大的特点，我们 采用第一种方法。此组件主要用户巡检模块的参数设置，基于jquery的zTree树形组件。

5.3 下拉框组件

对于下拉选项比较多的情况，我们采用基于jquery的下拉组件select2，实现可以在下拉框中模糊搜索匹配，从而快速筛选出想要的下拉选项。

5.4 model基类（CI\_Model）

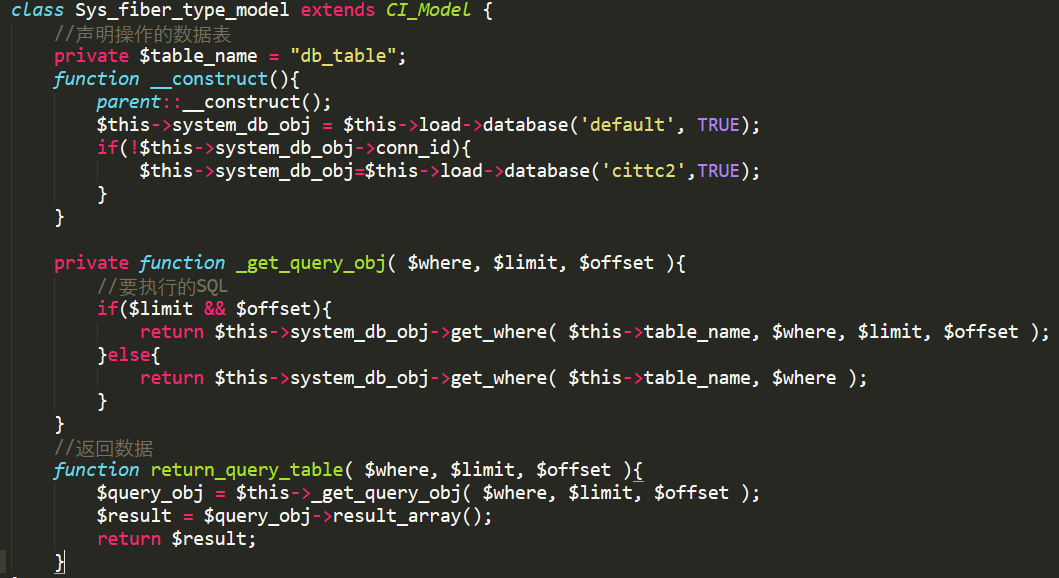
针对模型层的model基类继承CI的CI\_Model类, 实现数据对象的增删改查数据库操作，通过实现query()方法来执行SQL语句，通过result\_array()方法来设定返回的数据为二维数组。

5.5 controller基类（CI\_Controller）

该类作为业务层Controller类的基类，所有业务层Controller类必须继承该类。该类实现

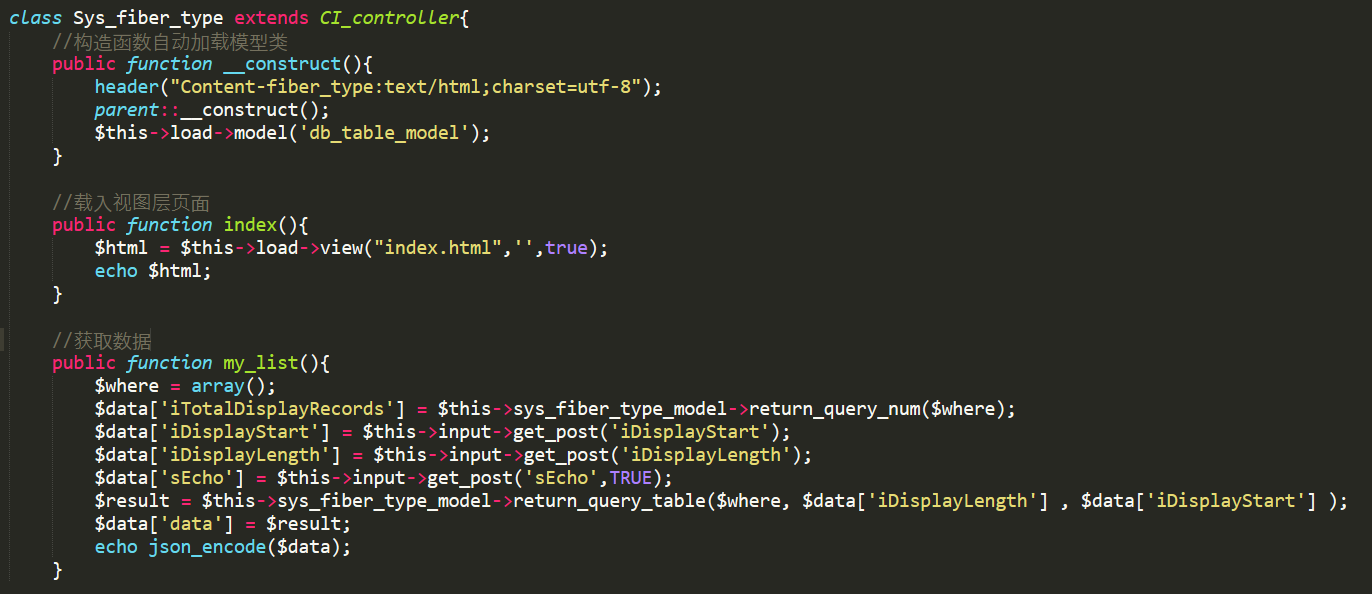
1. Web子模块基于架构的开发流程

6.1编写model类



6.2编写页面

6.3编写controller类



引入相应model类

$this->load->model('db\_table\_model');

调用相应页面

public function index(){

$html = $this->load->view("index.html",'',true);

echo $html;

}

6.4添加模块名称

在“系统管理”->“模块类别管理”和“模块管理”里面增加相应模块名称，模块名称是要在左侧菜单显示的名称，并设置此模块对应的控制器名称和方法名称。

6.5设置用户权限

刚添加的模块功能需要赋权限给用户才能访问，编辑“系统管理”->“角色管理”中的角色，将添加的模块权限赋给对应的角色，此角色下的用户将能访问到新模块。