

# 国家公路网综合养护管理系统 帮助文档

中公高科养护科技股份有限公司

养护管理信息化事业部

2018 年 8 月

# 国家公路网综合养护管理系统

## 帮助文档

文件状态：

☐ 大纲稿

☐ 征求意见稿

☐ 正式发布

拟制	王求卿	日期	2018.08.03	文件编号	
审核		日期			
批准		日期		版本号	V0.1
单位	中公高科养护科技股份有限公司				

# 目录

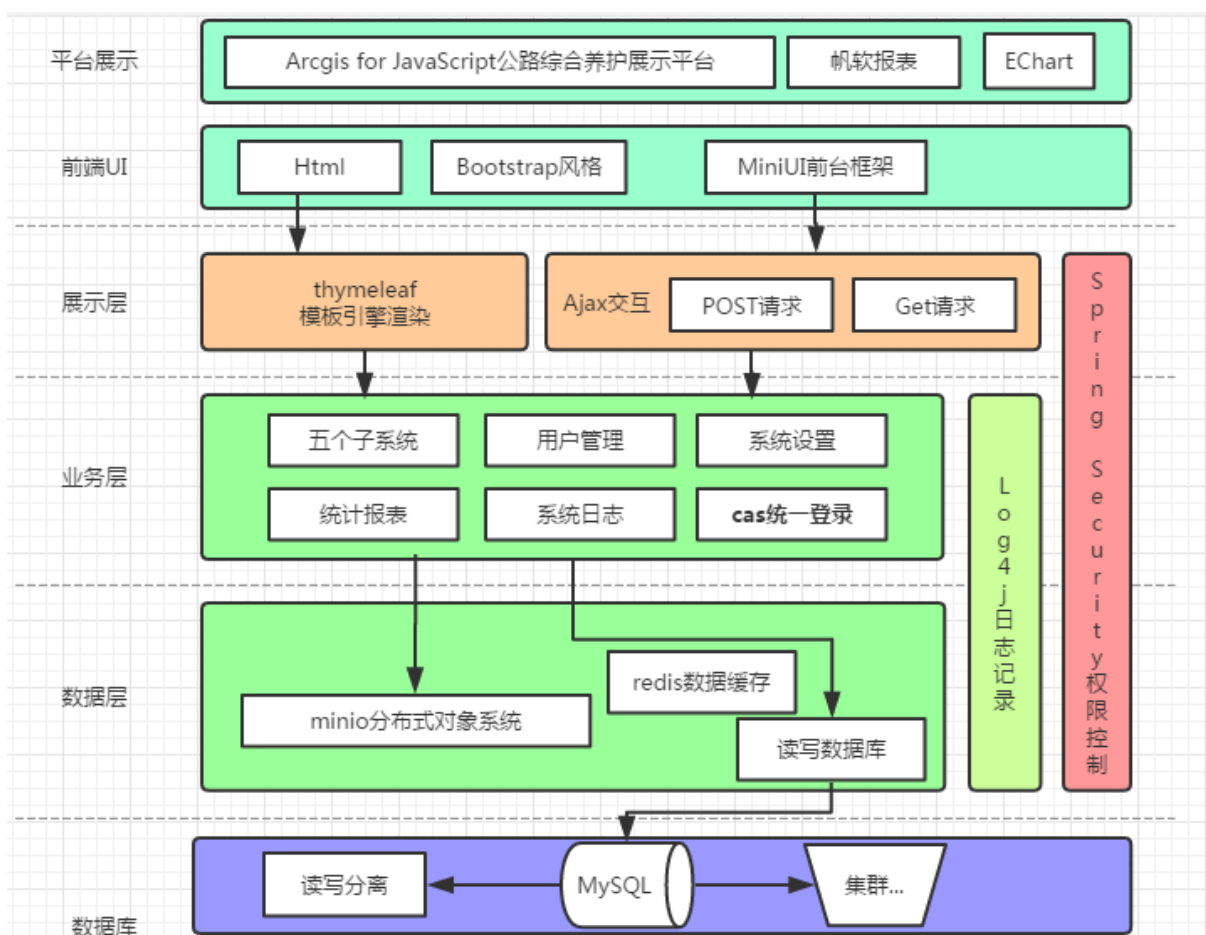
<b>第 1 章 系统开发框架 .....</b>	<b>1</b>
1.1 整体框架 .....	1
1.2 框架结构 .....	1
1.3 基础类库与和接口应用说明 .....	1
1.4 工具类库与接口应用说明 .....	4
1.5 组件库应用说明 .....	5
1.6 公用 CSS 库说明 .....	6
1.7 UI 应用说明 .....	6
<b>第 2 章 开发规范 .....</b>	<b>9</b>
2.1 编程规约 .....	9
2.1.1 命名规约 .....	9
2.1.2 格式规约 .....	10
2.1.3 OOP 规约 .....	13
2.1.4 集合处理 .....	15
2.1.5 注释规约 .....	16
2.1.6 其他 .....	17
2.2 异常日志规范 .....	18
2.2.1 异常处理 .....	18
2.2.2 日志规约 .....	19
2.3 数据库规约 .....	20
2.3.1 PDM 设计规约 .....	20
2.3.2 建表规约 .....	20
2.3.3 索引规约 .....	21
2.3.4 SQL 规约 .....	22
2.3.5 ORM 规约 .....	23
2.3.6 公用字段命名规约 .....	24
2.4 工程规约 .....	27
2.4.1 应用路径 .....	27
2.4.2 应用规范 .....	27
2.4.3 工程包规约 .....	28
2.4.3.1 包名 .....	28
2.4.3.2 类与接口名 .....	28
2.4.3.3 方法名 .....	28
2.4.3.4 属性名 .....	28
2.4.3.5 常量命名 .....	29
2.5 安全规约 .....	29
<b>第 3 章 帆软报表系统应用 .....</b>	<b>31</b>
3.1 环境搭建 .....	31

# 第1章 系统开发框架

## 1.1 整体框架

系统整体框架是基于 SpringMVC 的技术，数据访问层（DAO）使用 spring data jpa 进行数据查询。提供了原生 SQL 查询的能力。数据层的事务采用 @Transactional 注解的方式进行控制，前台与后台的交互采用 restful api 的形式，利用 spring security 来实现 OAUTH2 的用户鉴权。为了提高查询效率，前台采用 thymeleaf 模板进行展示。

## 1.2 框架结构



## 1.3 基础类库与接口应用说明

基础工具类的 API 提供: String, json, date, excel, number, web page 的工具类。

详细代码见: framework/com/platform/common/utils 包

字符串工具类:

限定符和类型	方法和说明
static java.lang.String	<a href="#"><code>convertFirstLowerCase</code></a> (java.lang.String source) 描述: 把一个字符串的第一个字母小写。
static java.lang.String	<a href="#"><code>convertFirstUpperCase</code></a> (java.lang.String source) 描述: 把一个字符串的第一个字母大写。
static boolean	<a href="#"><code>isAllLowerCase</code></a> (java.lang.String cs) 描述: 检查字符串是否全部为小写。
static boolean	<a href="#"><code>isAllUpperCase</code></a> (java.lang.String cs) 描述: 检查是否都是大写。
static boolean	<a href="#"><code>isNotNull</code></a> (java.lang.String str) 描述: 判断字符串不为空。
static boolean	<a href="#"><code>isNull</code></a> (java.lang.String str) 描述: 判断字符串为空。
static java.lang.String	<a href="#"><code>left</code></a> (java.lang.String str, int len) 描述: 截取一个字符串的前几个。
static java.lang.String	<a href="#"><code>leftPad</code></a> (java.lang.String str, int size) 描述: 扩大字符串长度, 从左边补充空格。
static java.lang.String	<a href="#"><code>leftPad</code></a> (java.lang.String str, int size, char padChar) 描述: 扩大字符串长度, 从左边补充指定的字符。
static java.lang.String	<a href="#"><code>leftPad</code></a> (java.lang.String str, int size, java.lang.String padStr) 描述: 扩大字符串长度, 从左边补充指定的字符。
static java.lang.String	<a href="#"><code>removeEnd</code></a> (java.lang.String str, java.lang.String remove) 描述: 只从源字符串中移除指定结尾的子字符串。
static java.lang.String	<a href="#"><code>removeStart</code></a> (java.lang.String str, java.lang.String remove) 描述: 只从源字符串中移除指定开头子字符串。
static java.lang.String	<a href="#"><code>repeat</code></a> (char ch, int repeat) 描述: 将某个字符重复 N 次。
static java.lang.String	<a href="#"><code>repeat</code></a> (java.lang.String str, int repeat) 描述: 将一个字符串重复 N 次。
static java.lang.String	<a href="#"><code>repeat</code></a> (java.lang.String str, java.lang.String separator, int repeat) 描述: 将一个字符串重复 N 次, 并且中间加上指定的分隔符。
static java.lang.String	<a href="#"><code>reverse</code></a> (java.lang.String str) 描述: 反转字符串。
static java.lang.String	<a href="#"><code>right</code></a> (java.lang.String str, int len) 描述: 从右边截取字符串。
static java.lang.String	<a href="#"><code>rightPad</code></a> (java.lang.String str, int size, char padChar) 描述: 字符串长度达不到指定长度时, 在字符串右边补指定的字符。
static java.lang.String	<a href="#"><code>rightPad</code></a> (java.lang.String str, int size, java.lang.String padStr) 描述: 扩大字符串长度, 从左边补充指定字符。
static java.lang.String	<a href="#"><code>substringBetween</code></a> (java.lang.String str, java.lang.String tag) StringUtils.substringBetween("tagabctag","tag")="abc" 描述: 得到 tag 字符串中间子字符串, 只返回第一个匹配项。
static java.lang.String	<a href="#"><code>substringBetween</code></a> (java.lang.String str, java.lang.String open, java.lang.String close) 描述: 得到两个字符串中间子字符串, 只返回第一个匹配项。
static java.lang.String	<a href="#"><code>swapCase</code></a> (java.lang.String str)

	描述: 切换字符串中的所有字母大小写。
static java.lang.String <a href="#">toCamelCase</a> (java.lang.String param)	描述: 下划线字段转换成驼峰规则。.
static java.lang.String <a href="#">toString</a> (java.lang.Object obj)	描述: 进行 toString 操作, 如果传入的是 null, 返回空字符串。
static java.lang.String <a href="#">toString</a> (java.lang.Object obj, java.lang.String nullStr)	描述: 进行 toString 操作, 如果传入的是 null, 返回指定的默认值。.
static java.lang.String <a href="#">toUnderlineName</a> (java.lang.String param)	描述: 驼峰转换成下划线规则。.

## JSON 工具类:

限定符和类型	方法和说明
static <T> java.lang.String	<a href="#">dealBootstrapPage</a> (org.springframework.data.domain.Page<T> pages) 描述: 处理 Jpa 的 PAGE, 返回的 BootstrapPage 需要的 json 格式 当前第几页: "+pages.getNumber()+"-当前记录行数-"+pages.getSize()+"-总共多少页--"+pages.getTotalPages()+"-总记录条数--"+pages.getTotalElements().
static java.lang.String	<a href="#">dealJson</a> (java.lang.Object obj) 描述: 对象, 集合转字符串 .
static <T> java.util.List<T>	<a href="#">dealJsonToList</a> (java.lang.String json, java.lang.Class<T> clazz) 描述: JSON 转数组 .
static <T> T	<a href="#">dealJsonToObject</a> (java.lang.String json, java.lang.Class<T> clazz) 描述: JSON 转对象 .
static <T> java.lang.String	<a href="#">dealMiniPage</a> (org.springframework.data.domain.Page<T> pages) 描述: 处理 Jpa 的 PAGE, 返回的 BootstrapPage 需要的 json 格式 当前第几页: "+pages.getNumber()+"-当前记录行数-"+pages.getSize()+"-总共多少页--"+pages.getTotalPages()+"-总记录条数--"+pages.getTotalElements().

## 日期工具类:

限定符和类型	方法和说明
static java.lang.String	<a href="#">getCurrentDate</a> () 描述: 获取当前时间“日期 ”字符串。
static java.lang.String	<a href="#">getCurrentShort</a> () 描述: 获取时间 小时:分;秒 HH:mm:ss.
static java.lang.String	<a href="#">getCurrentTime</a> () 描述: 获取当前时间 “日期 时分秒”字符串。

## excel 工具类:

限定符和类型	方法和说明
static void	<a href="#">checkFile</a> (org.springframework.web.multipart.MultipartFile file)
static java.lang.Object	<a href="#">getCellValue</a> (org.apache.poi.ss.usermodel.Cell cell) 描述： 对表格中数值进行格式化。
java.lang.String	<a href="#">getFormat</a> (java.lang.String str)
java.lang.Integer	<a href="#">getFormats</a> (java.lang.Integer str)
static org.apache.poi.ss.usermodel.Workbook	<a href="#">getWorkbook</a> (org.springframework.web.multipart.MultipartFile file) 描述： 根据文件后缀，自动适应上传文件的版本。
static java.util.List<java.util.Map<java.lang.String,java.lang.Object>>	<a href="#">readExcel</a> (org.springframework.web.multipart.MultipartFile file) 读取 excel 文件后，把第一行中的值，用表头组成 map 集合 所有的行作为一个集合返回。

处理 web 页面的工具类：

限定符和类型	方法和说明
static org.springframework.data.domain.Pageable	<a href="#">dealbootRequest</a> (javax.servlet.http.HttpServletRequest request) 描述： 解析 current=1,rowCount=10,sort[sender]=asc,searchPhrase=,id=bodf282a-od67-40e5-8558-c9e93b7befed Sort sort = new Sort(Sort.Direction.DISC, "id"); Pageable pageable = new PageRequest(0, 10, sort);.
static org.springframework.data.domain.Pageable	<a href="#">dealminiRequest</a> (javax.servlet.http.HttpServletRequest request) 描述： 处理前台接收的 miniUI 传入的参数，封装成 page 需要的结构数据 ,sortField,sortOrder,pageSize,pageIndex.

数字工具类：

限定符和类型	方法和说明
static java.lang.Double	<a href="#">add</a> (java.lang.Number value1, java.lang.Number value2) 提供精确的加法运算。
static java.lang.Double	<a href="#">mul</a> (java.lang.Number value1, java.lang.Number value2) 提供精确的乘法运算。
static java.lang.Double	<a href="#">sub</a> (java.lang.Number value1, java.lang.Number value2) 提供精确的减法运算。

## 1.4 工具类库与接口应用说明

导入：导入支持标准的行和列的格式的数据，导入，对导入的数据读取对应的行和列的数据生成 List 集合，再调用平台提供的 batchInsert 方法 进行批量插入操作。

详细 demo 见：framework/com/platform/common/utils/POIUtils.java

导出:采用 miniui 的前台操作。解析对应的代码。

数据转换

文件上传下载：文件上传采用 file input 的上传控件，采用灵活的配置来设定文件上传的配置文件，详细 demo 见：example/person/import.html.



## 1.5 组件库应用说明

查询条件

查询条件：对于开发过程中的用到的，都进行了封装，详细 demo 见：

templates/home1 ~home6 包含了所有可能出现的查询页面的设计如图：



树:采用 miniui 的提供的 tree 组件。根据不同的数据生成不同的 tree。



详细代码可见：`crms\src\main\resources\templates\rights\org\index.html`

## 1.6 公用 CSS 库说明

整个系统框架提供统一的 `commonbase` 文件，要求每个页面必须引入，为了后续统一的调整样式，现已提供样式库：`public.css`（公共样式）和 `font-awesome.min.css`（公共图标）。

`public` 样式包含：

全局滚动条

全局以及 `table` 字体大小，字体样式、字体颜色

`table` 的字体大小调整。

`miniui` 的按钮样式的重写。

公共图标样式采用 `html5` 编写，可自定义大小及颜色：

图标地址：<http://www.fontawesome.com.cn/faicons/>

## 1.7 UI 应用说明

UI 应用主要是值应用支撑页面，应用支撑页面包含：

应用集成配置：

配置平台需要集成的系统工程及端口，平台采用 `iframe` 结构，可通过平台总集成界面生成针对不同系统的 URL。如图：

+ 添加   编辑   删除			请输入名称或编码   查询		
<input type="checkbox"/>	应用上下文	子系统服务名称	子系统服务IP	子系统服务端口	备注
<input type="checkbox"/>	cbms	桥梁	106.37.229.146	6004	桥梁子系统1
<input type="checkbox"/>	chms	长大桥隧	106.37.229.146	6003	02包系统
<input type="checkbox"/>	cims	投资决策	106.37.229.146	6005	中咨集团
<input type="checkbox"/>	WebReport	帆软	106.37.229.146	8075	

用户管理：

用户管理包含用户的维护，以及角色绑定，绑定角色之后，用户就拥有该用户的角色权限。

+ 添加

✎ 编辑

✕ 删除

 角色分配

请输入名称或编码

🔍 查询

<input type="checkbox"/>	用户账号 △	用户姓名	所属机构	电话	邮箱	备注	是否启用
<input type="checkbox"/>	admin	超级管理员		13922301658	16973066@qq.com	超级管理员	是
<input type="checkbox"/>	cbms	<div><div> 角色分配</div><div><div><input type="checkbox"/> 角色名称</div><div><input type="checkbox"/> 全数据权限</div><div><input type="checkbox"/> 查看权限</div><div><input checked="" type="checkbox"/> 报表角色-中公高科</div><div><input type="checkbox"/> 报表角色-公规院</div><div><input type="checkbox"/> 报表角色-北京新桥</div><div><input type="checkbox"/> manager管理角色</div><div><input type="checkbox"/> 报表角色-中咨集团</div></div></div>					是
<input type="checkbox"/>	chms						是
<input type="checkbox"/>	cims						是
<input checked="" type="checkbox"/>	crms				@163.com		是
<input type="checkbox"/>	guest02					JAVA开发	是
<input type="checkbox"/>	guest03						是
<input type="checkbox"/>	manager				@qq.com	系统管理员	是
<input type="checkbox"/>	sd				@qq.com	公路工程师	是
<input type="checkbox"/>	test						是
<input type="checkbox"/>	xq						是
<input type="checkbox"/>	yn				qq.com	桥梁工程师	是
<input type="checkbox"/>	zggk						是

角色管理:

角色管理是用于绑定改角色对应的功能项以及按钮项，来控制整个平台的访问权限。通过 security 标签来绑定对应的 restful api 是否可以访问。

+ 添加	编辑	删除	绑定功能			
<input type="checkbox"/>	序号	角色名称 ▾	是否启用	备注		
<input checked="" type="checkbox"/>	1	查看权限	是	所有用户的查看权限		
<input type="checkbox"/>	2	报表角色-北京新桥				
<input type="checkbox"/>	3	报表角色-公规院				
<input type="checkbox"/>	4	报表角色-中咨集团				
<input type="checkbox"/>	5	报表角色-中公高科				
<input type="checkbox"/>	6	全数据权限				
<input type="checkbox"/>	7	manager管理角色				

### 功能管理：

功能管理是基于功能项的配置菜单。如下图

名称: 请输入功能名称 查询

国家公路网综合养护管理系统

- 首页
- 展示平台
- 数据集成
- 路况监测
- 路面管理
- 桥梁管理
- 隧道管理
- 投资决策
- 长大桥隧监测
- 应用支撑

+ 添加 编辑 删除

请输入名称或编码 查询

<input type="checkbox"/>	功能名称	链接URL	权限标识	来源系统	上级菜单	是否启用	是否菜单	排序字段	样式	备注
<input checked="" type="checkbox"/>	国家公路网综合...			默认		是	菜单			
<input type="checkbox"/>	首页			默认	国家公路网综合...	否	菜单	0		
<input type="checkbox"/>	展示平台	gist/index		默认	国家公路网综合...	是	菜单	0	📍	
<input type="checkbox"/>	数据集成			默认	国家公路网综合...	是	菜单	1	📊	
<input type="checkbox"/>	路况监测			默认	国家公路网综合...	是	菜单	2	👥	
<input type="checkbox"/>	路面管理			默认	国家公路网综合...	是	菜单	3	👥	
<input type="checkbox"/>	桥梁管理			默认	国家公路网综合...	是	菜单	4	📊	
<input type="checkbox"/>	隧道管理			默认	国家公路网综合...	是	菜单	5	📊	
<input type="checkbox"/>	投资决策			投资决策	国家公路网综合...	是	菜单	6	¥	
<input type="checkbox"/>	长大桥隧监测			长大桥隧	国家公路网综合...	是	菜单	7	📍	
<input type="checkbox"/>	应用支撑			默认	国家公路网综合...	是	菜单	8	👥	
<input type="checkbox"/>	路线信息集成应...			默认	数据集成	是	菜单	0	📊	
<input type="checkbox"/>	首页			长大桥隧	长大桥隧监测	是	菜单	0		
<input type="checkbox"/>	路面技术状况评...			默认	路面管理	是	菜单	0		
<input type="checkbox"/>	基于GIS的智能搜...			默认	展示平台	是	菜单	0		
<input type="checkbox"/>	公路技术状况评...			投资决策	投资决策	是	菜单	1		
<input type="checkbox"/>	路面信息集成应...			默认	数据集成	是	菜单	1	👥	

20 / 22 1 / 22 每页 20 条,共 422 条

组织机构管理：

组织结构是对于现有路网系统所有用户的组织机构管理页面，用于后期的上传及上报的根据，可根据组织机构数据来配置对应管养单位以及管养单位所拥有的权限。

+ 添加 编辑 删除

请输入名称或编码 查询

<input type="checkbox"/>	组织机构名称	组织机构编号	所属上级机构	联系人	联系电话	邮编	地址	是否启用	备注
<input checked="" type="checkbox"/>	全国	000000						是	
<input type="checkbox"/>	东部	1						是	
<input type="checkbox"/>	交通运输部路网...	1001						是	
<input type="checkbox"/>	中部	2						是	
<input type="checkbox"/>	西部	3						是	
<input type="checkbox"/>	北京市	110000						是	
<input type="checkbox"/>	天津市	120000						是	
<input type="checkbox"/>	河北省	130000						是	
<input type="checkbox"/>	山西省	140000						是	
<input type="checkbox"/>	内蒙古自治区	150000						是	
<input type="checkbox"/>	辽宁省	210000						是	
<input type="checkbox"/>	吉林省	220000						是	
<input type="checkbox"/>	黑龙江省	230000						是	
<input type="checkbox"/>	上海市	310000						是	
<input type="checkbox"/>	江苏省	320000						是	
<input type="checkbox"/>	浙江省	330000						是	
<input type="checkbox"/>	安徽省	340000						是	

20 / 188 1 / 188 每页 20 条,共 3742 条

## 第2章 开发规范

### 2.1 编程规约

#### 2.1.1 命名规约

1. 【强制】 代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

反例： `_name` / `__name` / `$Object` / `name_` / `name$` / `Object$`

2. 【强制】 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。

说明： 正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式也要避免采用。

反例： `DaZhePromotion` [打折] / `getPingfenByName()` [评分] / `int 某变量 = 3`

正例： `alibaba` / `taobao` / `youku` / `hangzhou` 等国际通用的名称， 可视同英文。

3. 【强制】类名使用 UpperCamelCase 风格，必须遵从驼峰形式，但以下情形例外：（领域模型的相关命名） `DO` / `BO` / `DTO` / `VO` 等。

正例： `MarcoPolo` / `UserDO` / `XmlService` / `TcpUdpDeal` / `TaPromotion`

反例： `macroPolo` / `UserDo` / `XMLService` / `TCPUDPDeal` / `TAPromotion`

4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从驼峰形式。

正例： `localValue` / `getHttpMessage()` / `inputUserId`

5. 【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。

正例： `MAX_STOCK_COUNT`

反例： `MAX_COUNT`

6. 【强制】中括号是数组类型的一部分，数组定义如下： `String[] args`;

反例： 请勿使用 `String args[]`的方式来定义。

7. 【强制】 POJO 类中布尔类型的变量，都不要加 `is`，否则部分框架解析会引起序列化错误。

反例： 定义为基本数据类型 `boolean isSuccess`；的属性，它的方法也是 `isSuccess()`，

RPC 框架在反向解析的时候，“以为”对应的属性名称是 `success`，导致属性获取不到，进而抛出异常。

8. **【强制】**包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用

单数形式，但是类名如果有复数含义，类名可以使用复数形式。

**正例：**应用工具类包名为 `com.alibaba.open.util`、类名为 `MessageUtils`（此规则参考 `spring` 的框架结构）

9. **【强制】**杜绝完全不规范的缩写，避免望文不知义。

**反例：**`AbstractClass`“缩写”命名成 `AbsClass`；`condition`“缩写”命名成 `condi`，此类随意缩写严重降低了代码的可阅读性。

10. 接口和实现类的命名有两套规则：

1) **【强制】**对于 `Service` 和 `DAO` 类，基于 `SOA` 的理念，暴露出来的服务一定是接口，内部的实现类用 `Impl` 的后缀与接口区别。

**正例：**`CacheServiceImpl` 实现 `CacheService` 接口。

2) **【推荐】**如果是形容能力的接口名称，取对应的形容词做接口名（通常是 `-able` 的形式）。

**正例：**`AbstractTranslator` 实现 `Translatable`。

11. **【参考】**各层命名规约：

`Service/DAO` 层方法命名规约

- 1) 获取单个对象的方法用 `get` 做前缀。
- 2) 获取多个对象的方法用 `list` 做前缀。
- 3) 获取统计值的方法用 `count` 做前缀。
- 4) 插入的方法用 `save`（推荐）或 `insert` 做前缀。
- 5) 删除的方法用 `remove`（推荐）或 `delete` 做前缀。
- 6) 修改的方法用 `update` 做前缀。

## 2.1.2格式规约

1. **【强制】**大括号的使用约定。如果是大括号内为空，则简洁地写成 `{}` 即可，不需要换行；如果是非空代码块则：

- 1) 左大括号前不换行。
- 2) 左大括号后换行。

正例: 

```
if (flag == 1) {  
    System.out.println("world");  
    // 右大括号前换行, 右大括号后有 else, 不用换行  
} else {  
    System.out.println("ok");  
    // 在右大括号后直接结束, 则必须换行  
}
```

2. 【强制】左括号和后一个字符之间不出现空格; 同样, 右括号和前一个字符之间也不出现空格。详见第 5 条下方正例提示。
3. 【强制】if/for/while/switch/do 等保留字与左右括号之间都必须加空格。
4. 【强制】任何运算符左右必须加一个空格。说明: 运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号、三目运行符等。
5. 【强制】缩进采用 4 个空格, 禁止使用 tab 字符。

说明: 如果使用 tab 缩进, 必须设置 缩进, 必须设置 缩进, 必须设置 缩进, 必须设置 缩进, 必须设置 缩进, 必须设置 1 个 tab 为 4 个空格。IDEA 设置 tab 为 4 个空格时, 请勿勾选 Use tab character ; 而在 eclipse 中, 必须勾选 insert spaces for tabs 。

正例: (涉及 1-5 点)

```
public static void main(String args[]) {  
    // 缩进 4 个空格  
    String say = "hello";  
    // 运算符的左右必须有一个空格  
    int flag = 0;  
    // 关键词 if 与括号之间必须有一个空格, 括号内的 f 与左括号, 0 与右括号不需要空格  
    if (flag == 0) {  
        System.out.println(say);  
    }
```

```
// 左大括号前加空格且不换行；左大括号后换行
if (flag == 1) {
    System.out.println("world");
// 右大括号前换行，右大括号后有 else，不用换行
} else {
    System.out.println("ok");
// 在右大括号后直接结束，则必须换行
}
}
```

6. 【强制】单行字符数限不超过 120 个，超出需要换行，换行时遵循如下原则：

- 1) 第二行相对一缩进 4 个空格，从第三行开始不再继续缩进参考示例。
- 2) 运算符与下文一起换行。
- 3) 方法调用的点符号与下文一起换行。
- 4) 在多个参数超长，逗号后进行换行。
- 5) 在括号前不要换行，见反例。

正例：

```
StringBuffer sb = new StringBuffer();
```

```
//超过 120 个字符的情况下，换行缩进 4 个空格，并且方法前的点符号一起换行
sb.append("zi").append("xin")...
    .append("huang")...
    .append("huang")...
    .append("huang");
```

反例：

```
StringBuffer sb = new StringBuffer();
```

```
//超过 120 个字符的情况下，不要在括号前换行
```

```
sb.append("zi").append("xin")...append
("huang");
```

```
//参数很多的方法调用可能超过 120 个字符，不要在逗号前换行
```

```
method(args1, args2, args3, ...
, argsX);
```

7. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。

正例：下例中实参的"a", 后边必须要有一个空格。

```
method("a", "b", "c");
```

8. 【强制】IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式，不要使用 windows 格式。

9. 【推荐】方法体内的执行语句组、变量的定义语句组、不同的业务逻辑之间或者不同的语义之间插入一个空行。相同业务逻辑和语义之间不需要插入空行。说明：没有必要插入多行空格进行隔开。

### 2.1.3 OOP 规约

1. 【强制】所有的要标明控制组件的方法，必须是加@RestController 注解。

说明：默认为返回 json 数据，如需返回页面需返回 ModelAndView

2. 【强制】controller 层引入 dao/service 的时候必须采用@Autowired 注解。

正例： @Autowired

```
FwplServerService fwPlserverService;
```

说明：可变参数必须放置在参数列表的最后。（提倡同学们尽量不用可变参数编程）

正例： public User getUsers(String type, Integer... ids)

3. 【强制】对于有需要用到功能权限的功能时候，统一添加@PreAuthorize 注解，详细见代码。

4. 【推荐】针对 modelAndView 返回到 thymeleaf 模板时，路径地址说明。

说明：return new ModelAndView("rights/server/index") 所返回的路径对应的资源路径为 main/resources/rights/server/index.html, 路径命名规则按照业务模块功能的含义。主界面用 index.html，编辑界面用：edit.html.

5. 【强制】Object 的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用 equals。

正例： "test".equals(object);

反例： object.equals("test");

6. 【强制】需要处理 Json 函数统一采用 com.platform.common.utils.JsonUtils 类进行处理。提供了将 list, object 转 JSON 字符串方法。以及 miniui, bootstrap 的分页



数据对象转 JSON 字符串方法, 或者使用 `com.alibaba.fastjson.*` 下的基础类进行处理 json, 不可以再引入其他的 json 处理包。

7. **【强制】** 构造方法里面禁止加入任何业务逻辑, 如果有初始化逻辑, 请放在 `init` 方法中。

8. **【强制】** POJO 类必须写 `toString` 方法。使用 IDE 的中工具: `source>generate toString` 时, 如果继承了另一个 POJO 类, 注意在前面加一下 `super.toString`。

**说明:** 在方法执行抛出异常时, 可以直接调用 POJO 的 `toString()` 方法打印其属性值, 便于排查问题。

9. **【推荐】** 类内方法定义顺序依次是: 公有方法或保护方法 > 私有方法 > `getter/setter` 方法。

**说明:** 公有方法是类的调用者和维护者最关心的方法, 首屏展示最好; 保护方法虽然只是子类关心, 也可能是“模板设计模式”下的核心方法; 而私有方法外部一般不需要特别关心, 是一个黑盒实现; 因为方法信息价值较低, 所有 `Service` 和 `DAO` 的 `getter/setter` 方法放在类体最后。

10. **【推荐】** `setter` 方法中, 参数名称与类成员变量名称一致, `this.成员名=参数名`。在 `getter/setter` 方法中, 尽量不要增加业务逻辑, 增加排查问题的难度。

**反例:**

```
public Integer getData() {  
    if(true) {  
        return data + 100;  
    } else {  
        return data - 100;  
    }  
}
```

11. **【推荐】** 循环体内, 字符串的联接方式, 使用 `StringBuilder` 的 `append` 方法进行扩展。

**反例:**

```
String str = "start";  
for(int i=0; i<100; i++){  
    str = str + "hello";  
}
```

```
}
```

**说明：**反编译出的字节码每次循环都会 new 出一个 StringBuilder 对象，然后进行 append 操作，最后通过 toString 方法返回 String 对象，造成内存资源浪费。

### 2.1.4 集合处理

1. **【强制】**工具类操作使用 com.google.common.collect.\*下的包进行操作。

例子：`List<test> list = Lists.newArrayList();`

说明：该包处理泛型的提示，以及使用的安全问题。详细资料可自行查找。苹果。

2. **【强制】**不要在 foreach 循环里进行元素的 remove/add 操作。remove 元素请使用 Iterator 方式，如果并发操作，需要对 Iterator 对象加锁。

**反例：**

```
List<String> a = new ArrayList<String>();  
a.add("1");  
a.add("2");  
for (String temp : a) {  
    if("1".equals(temp)) {  
        a.remove(temp);  
    }  
}
```

**说明：**以上代码的执行结果肯定会出乎大家的意料，那么试一下把“1”换成“2”，会是同样的结果吗？

**正例：**

```
Iterator<String> it = a.iterator();  
while(it.hasNext()) {  
    String temp = it.next();  
    if(删除元素的条件) {  
        it.remove();  
    }  
}
```

3. **【推荐】**高度注意 Map 类集合 K/V 能不能存储 null 值的情况，如下表格：

集合类	Key	Value	Super	说明
Hashtable	不允许为 null	不允许为 null	Dictionary	线程安全
ConcurrentHashMap	不允许为 null	不允许为 null	AbstractMap	分段锁技术
TreeMap	不允许为 null	允许为 null	AbstractMap	线程不安全
HashMap	允许为 null	允许为 null	AbstractMap	线程不安全

**反例：**由于 HashMap 的干扰，很多人认为 ConcurrentHashMap 是可以置入 null 值，注意存储 null 值时会抛出 NPE 异常。

3. **【强制】** SimpleDateFormat 是线程不安全的类，一般不要定义为 static 变量，如果定义为 static，必须加锁，或者使用 DateUtils 工具类。

**正例：**注意线程安全，使用 DateUtils。亦推荐如下处理：

```
private static final ThreadLocal<DateFormat> df = new
ThreadLocal<DateFormat>() {
    @Override
    protected DateFormat initialValue() {
        return new SimpleDateFormat("yyyy-MM-dd");
    }
};
```

**说明：**如果是 JDK8 的应用，可以使用 Instant 代替 Date，LocalDateTime 代替 Calendar，DateTimeFormatter 代替 Simpledateformatter，官方给出的解释：simple beautiful strong immutable thread-safe。

## 2.1.5 注释规约

1. **【强制】** 类、类属性、类方法的注释必须使用 Javadoc 规范，使用 `/**内容*/` 格式，不得使用 `//xxx` 方式。

**说明：**在 IDE 编辑窗口中，Javadoc 方式会提示相关注释，生成 Javadoc 可以正确输出相应注释；在 IDE 中，工程调用方法时，不进入方法即可悬浮提示方法、参数、返回值的意义，提高阅读效率。注释模板由总集提供。

2. **【强制】** 所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、异常说明外，还必须指出该方法做什么事情，实现什么功能。

**说明：**对子类的实现要求，或者调用注意事项，请一并说明。

3. 【强制】所有的类都必须添加创建者信息。
4. 【强制】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释使用/\* \*/注释，注意与代码对齐。
5. 【强制】所有的枚举类型字段必须要有注释，说明每个数据项的用途。
6. 【推荐】与其“半吊子”英文来注释，不如用中文注释把问题说清楚。专有名词与关键字保持英文原文即可。

**反例：**“TCP 连接超时”解释成“传输控制协议连接超时”，理解反而费脑筋。

7. 【参考】注释掉的代码尽量要配合说明，而不是简单的注释掉。

**说明：**代码被注释掉有两种可能性：1) 后续会恢复此段代码逻辑。2) 永久不用。前者如果没有备注信息，难以知晓注释动机。后者建议直接删掉（代码仓库保存了历史代码）。

8. 【参考】对于注释的要求：第一、能够准确反应设计思想和代码逻辑；第二、能够描述业务含义，使别的程序员能够迅速了解到代码背后的信息。完全没有注释的大段代码对于阅读者形同天书，注释是给自己看的，即使隔很长时间，也能清晰理解当时的思路；注释也是给继任者看的，使其能够快速接替自己的工作。

9. 【推荐】代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑等的修改。

**说明：**代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后，就失去了导航的意义。

## 2.1.6其他

1. 【强制】在使用正则表达式时，利用好其预编译功能，可以有效加快正则匹配速度。  
**说明：**不要在方法体内定义：Pattern pattern = Pattern.compile(规则);
2. 【强制】velocity 调用 POJO 类的属性时，建议直接使用属性名取值即可，模板引擎会自动按规范调用 POJO 的 getXxx()，如果是 boolean 基本数据类型变量（boolean 命名不需要加 is 前缀），会自动调用 isXxx() 方法。  
**说明：**注意如果是 Boolean 包装类对象，优先调用 getXxx() 的方法。
3. 【强制】后台输送给页面的变量必须加\${var}——中间的感叹号。  
**说明：**如果 var=null 或者不存在，那么\${var}会直接显示在页面上。
4. 【强制】注意 Math.random() 这个方法返回是 double 类型，注意取值的范围  $0 \leq x < 1$ （能够取到零值，注意除零异常），如果想获取整数类型的随机数，不要将 x 放大 10 的若干倍然后取整，直接使用 Random 对象的 nextInt 或者 nextLong 方法。
5. 【强制】获取当前毫秒数 System.currentTimeMillis(); 而不是 new Date().getTime();  
**说明：**如果想获取更加精确的纳秒级时间值，用 System.nanoTime()。在 JDK8 中，针对统计时间等场景，推荐使用 Instant 类。

6. 【推荐】尽量不要在 vm 中加入变量声明、逻辑运算符，更不要在 vm 模板中加入任何复杂的逻辑。
7. 【推荐】任何数据结构的构造或初始化，都应指定大小，避免数据结构无限增长吃光内存。
8. 【推荐】对于“明确停止使用的代码和配置”，如方法、变量、类、配置文件、动态配置属性等要坚决从程序中清理出去，避免造成过多垃圾。

## 2.2 异常日志规范

### 2.2.1 异常处理

1. 【强制】不要捕获 Java 类库中定义的继承自 RuntimeException 的运行时异常类，如：IndexOutOfBoundsException / NullPointerException，这类异常由程序员预检查来规避，保证程序健壮性。

正例： `if(obj != null) {...}`

反例： `try { obj.method() } catch(NullPointerException e){...}`

2. 【推荐】防止 NPE，是程序员的基本修养，注意 NPE 产生的场景：

- 1) 返回类型为包装数据类型，有可能是 null，返回 int 值时注意判空。

反例： `public int f(){return Integer 对象}`；如果为 null，自动解箱抛 NPE。

- 2) 数据库的查询结果可能为 null。

- 3) 集合里的元素即使 isEmpty，取出的数据元素也可能为 null。

- 4) 远程调用返回对象，一律要求进行 NPE 判断。

- 5) 对于 Session 中获取的数据，建议 NPE 检查，避免空指针。

- 6) 级联调用 `obj.getA().getB().getC()`；一连串调用，易产生 NPE。

2. 【强制】异常不要用来做流程控制，条件控制，因为异常的处理效率比条件分支低。

3. 【强制】对大段代码进行 try-catch，这是不负责任的表现。catch 时请分清稳定代码和非稳定代码，稳定代码指的是无论如何不会出错的代码。对于非稳定代码的 catch 尽可能进行区分异常类型，再做对应的异常处理。

4. 【强制】捕获异常是为了处理它，不要捕获了却什么都不处理而抛弃之，如果不想处理它，请将该异常抛给它的调用者。最外层的业务使用者，必须处理异常，将其转化为用户可以理解的内容。

5. 【强制】有 try 块放到了事务代码中，catch 异常后，如果需要回滚事务，一定要注意手动回滚事务。

6. 【强制】finally 块必须对资源对象、流对象进行关闭，有异常也要做 try-catch。说明：

如果JDK7，可以使用try-with-resources方式。

7. 【强制】不能在finally块中使用return，finally块中的return返回后方法结束执行，不会再执行try块中的return语句。

8. 【强制】捕获异常与抛异常，必须是完全匹配，或者捕获异常是抛异常的父类。说明：如果预期对方抛的是绣球，实际接到的是铅球，就会产生意外情况。

9. 【推荐】方法的返回值可以为null，不强制返回空集合，或者空对象等，必须添加注释充分说明什么情况下会返回null值。调用方需要进行null判断防止NPE问题。说明：本规约明确防止NPE是调用者的责任。即使被调用方法返回空集合或者空对象，对调用者来说，也并非高枕无忧，必须考虑到远程调用失败，运行时异常等场景返回null的情况。

10. 远程调用返回对象，一律要求进行NPE判断。 5) 对于Session中获取的数据，建议NPE检查，避免空指针。 6) 级联调用obj.getA().getB().getC();一连串调用，易产生NPE。

11. 【推荐】在代码中使用“抛异常”还是“返回错误码”，对于公司外的http/api开放接口必须使用“错误码”；而应用内部推荐异常抛出；跨应用间RPC调用优先考虑使用Result方式，封装isSuccess、“错误码”、“错误简短信息”。说明：关于RPC方法返回方式使用Result方式的理由： 1) 使用抛异常返回方式，调用方如果没有捕获到就会产生运行时错误。 2) 如果不加栈信息，只是new自定义异常，加入自己的理解的error message，对于调用端解决问题的帮助不会太多。如果加了栈信息，在频繁调用出错的情况下，数据序列化和传输的性能损耗也是问题。

12. 【推荐】定义时区分unchecked / checked 异常，避免直接使用RuntimeException抛出，更不允许抛出Exception或者Throwable，应使用有业务含义的自定义异常。推荐业界已定义过的自定义异常，如：DAOException / ServiceException等。

13. 【参考】避免出现重复的代码（Don't Repeat Yourself），即DRY原则。说明：随意复制和粘贴代码，必然会导致代码的重复，在以后需要修改时，需要修改所有的副本，容易遗漏。必要时抽取共性方法，或者抽象公共类，甚至是共用模块。正例：一个类中有多个public方法，都需要进行数行相同的参数校验操作，这个时候请抽取：

```
private boolean checkParam(DTO dto) {...}
```

## 2.2.2 日志规约

日志主要指的是用户增加，修改，删除数据时候，用通用方法记录操作日志。

### 【样例】

service 层：首先集成总集提供的 ServiceImpl 方法，里面提供了统一的日志操作类。第二调用日志操作类的 save 方法，参数为 log 对象（操作类型，操作的表名称，操作信息）

1) 继承 extends BaseServiceImpl

2) this.getLogDao().save(this.getOperationLog(OPERATE\_TYPE\_ADD, 操作的表名称, "操作信息"));

## 2.3 数据库规约

### 2.3.1 PDM 设计规约

1. **【强制】** Comment 备注项必须列出。
2. **【强制】** 字典字段 Comment 需列出字典值说明。
3. **【强制】** 外键字段 Comment 需说明关联字段和关联表，字段名和主表字段名需一致。
4. **【强制】** Owner 必须选上

### 2.3.2 建表规约

字典编号编码 CODE, TYPE

1. **【强制】** 在定义数据库表、列的时候全部采用小写字母加下划线的方式，不使用任何大写字母。

说明：数据表名，字段全部采用小写，有助于避免数据库版本区分大小写问题。

2. **【强制】** 数据库、表、字段等所有名称使用英文单词或英文短语或相应缩写，禁止使用汉语拼音，且均使用单数名。

例如：对存储客人信息的表命名为 customer 而不是 customers。

3. **【强制】** 所有表中同意义的字段名称，字段类型应一致，字段名称应统一。

说明：数据设计阶段，各包多交流，尤其是对专业性比较强的字段命名更应该一致。

4. **【强制】** 对系统关键数据表设计，应加入创建人, 创建时间, 修改人, 修改时间, 备注说明：为了满足三级等保审计要求

字段命名为 create\_user\_id varchar(40), create\_time timestamp, update\_user\_id varchar(40), update\_time timestamp, remark varchar(300)

4. **【强制】** 表引擎类型应该为 InnoDB

说明：如果为其他类型，则该表不拥有事务的能力

5. **【推荐】** 表规则 数据表字符集统一采用 utf8, 排序规则采用 utf8\_general\_ci.

6. **【强制】** 表名命名规则. 所属系统简称\_所属业务\_所属模块

正例：fw\_right\_user, rd\_prevent\_cost, fw\_operation\_log 等等

反例：t\_person, users\_dept , fw\_zuzhijigou

说明：统一的正确的命名规则，能让开发者简单了解后找到对应的数据表。

7. **【强制】表主键规则** . 表字段必须包含 ID 主键，作为唯一标示。主键名称就叫做 ID，不要加上业务名称。主键 ID 字段必须为 varchar 类型，长度为 40。

说明：统一的主键命名，方便开发者进行调用，杂乱的命名，会影响使用者的理解，混淆主键。主键采用 UUID 类型，所以长度只需 40。 varchar 类型方便数据迁移，以及免去自增序列值的维护工作。方便处理分布式存储数据，不会超出 INT 取值范围

8. **【推荐】表字段长度规则** 字段长度应贴合实际业务进行定义，如不确定，可前期将长度设置成 100，后续不够再进行扩展

说明：长度的准确定义可以大大缩小表空间的使用

9. **【推荐】表字段类型规则** 多媒体数据格式的存储，表字段禁止使用 blob 类型，统一采用路径存储。日期类型定义成：timestamp 或者 date 类型，根据实际业务定义。

10. **【强制】数据库底层访问接口**，强制要求采用 spring data JPA 进行读取。避免写入一些特定函数的 SQL。

说明：spring data JPA 很好的兼容 spring。兼容任何的数据库, 以后可以快速迁移数据库以及后续平台进行云部署。

### 2.3.3 索引规约

1. **【强制】业务上具有唯一特性的字段**，即使是组合字段，也必须建成唯一索引。说明：不要以为唯一索引影响了 insert 速度，这个速度损耗可以忽略，但提高查找速度是明显的；另外，即使在应用层做了非常完善的校验和控制，只要没有唯一索引，根据墨菲定律，必然有脏数据产生。

2. **【强制】超过三个表禁止 join**。需要 join 的字段，数据类型保持绝对一致；多表关联查询时，保证被关联的字段需要有索引。说明：即使双表 join 也要注意表索引、SQL 性能。

3. **【强制】在 varchar 字段上建立索引时**，必须指定索引长度，没必要对全字段建立索引，根据实际文本区分度决定索引长度。说明：索引的长度与区分度是一对矛盾体，一般对字符串类型数据，长度为 20 的索引，区分度会高达 90%以上，可以使用 `count(distinct left(列名, 索引长度))/count(*)` 的区分度来确定。

4. **【强制】页面搜索严禁左模糊或者全模糊**，如果需要请走搜索引擎来解决。说明：索引文件具有 B-Tree 的最左前缀匹配特性，如果左边的值未确定，那么无法使用此索引。

5. **【推荐】如果有 order by 的场景**，请注意利用索引的有序性。order by 最后的字段是组合索引的一部分，并且放在索引组合顺序的最后，避免出现 file\_sort 的情况，影响查询性能。正例：where a=? and b=? order by c; 索引：a\_b\_c 反例：索引中有范围查找，那么索引有序性无法利用，如：WHERE a>10 ORDER BY b; 索引 a\_b 无法排序。



6. 【推荐】利用覆盖索引来进行查询操作，来避免回表操作。说明：如果一本书需要知道第 11 章是什么标题，会翻开第 11 章对应的那一页吗？目录浏览一下就好，这个目录就是起到覆盖索引的作用。正例：能够建立索引的种类：主键索引、唯一索引、普通索引，而覆盖索引是一种查询的一种效果，用 explain 的结果，extra 列会出现：using index。
7. 【推荐】利用延迟关联或者子查询优化超多分页场景。说明：MySQL 并不是跳过 offset 行，而是取 offset+N 行，然后返回放弃前 offset 行，返回 N 行，那当 offset 特别大的时候，效率就非常的低下，要么控制返回的总页数，要么对超过特定阈值的页数进行 SQL 改写。正例：先快速定位需要获取的 id 段，然后再关联：SELECT a.\* FROM 表 1 a, (select id from 表 1 where 条件 LIMIT 100000,20 ) b where a.id=b.id
8. 【推荐】SQL 性能优化的目标：至少要达到 range 级别，要求是 ref 级别，如果可以是 consts 最好。说明：1) consts 单表中最多只有一个匹配行（主键或者唯一索引），在优化阶段即可读取到数据。2) ref 指的是使用普通的索引（normal index）。3) range 对索引进行范围检索。反例：explain 表的结果，type=index，索引物理文件全扫描，速度非常慢，这个 index 级别比较 range 还低，与全表扫描是小巫见大巫。
9. 【推荐】建组合索引的时候，区分度最高的在最左边。正例：如果 where a=? and b=?，a 列的几乎接近于唯一值，那么只需要单建 idx\_a 索引即可。说明：存在非等号和等号混合判断条件时，在建索引时，请把等号条件的列前置。如：where a>? and b=? 那么即使 a 的区分度更高，也必须把 b 放在索引的最前列。
10. 【参考】创建索引时避免有如下极端误解：1) 误认为一个查询就需要建一个索引。2) 误认为索引会消耗空间、严重拖慢更新和新增速度。3) 误认为唯一索引一律需要在应用层通过“先查后插”方式解决。

## 2.3.4SQL 规约

1. 【强制】不要使用 count(列名)或 count(常量)来替代 count(\*), count(\*) 就是 SQL92 定义的标准统计行数的语法，跟数据库无关，跟 NULL 和非 NULL 无关。说明：count(\*) 会统计值为 NULL 的行，而 count(列名)不会统计此列为 NULL 值的行。
2. 【强制】count(distinct col) 计算该列除 NULL 之外的不重复数量。注意 count(distinct col1, col2) 如果其中一列全为 NULL，那么即使另一列有不同的值，也返回为 0。
3. 【强制】当某一列的值全是 NULL 时，count(col) 的返回结果为 0，但 sum(col) 的返回结果为 NULL，因此使用 sum() 时需注意 NPE 问题。正例：可以使用如下方式来避免 sum 的 NPE 问题：SELECT IF(ISNULL(SUM(g)),0,SUM(g)) FROM table;
4. 【强制】使用 ISNULL() 来判断是否为 NULL 值。注意：NULL 与任何值的直接比较都为 NULL。说明：1) NULL<>NULL 的返回结果是 NULL，而不是 false。2) NULL=NULL 的返回结果是 NULL，而不是 true。3) NULL<>1 的返回结果是 NULL，而不是 true。
5. 【强制】在代码中写分页查询逻辑时，若 count 为 0 应直接返回，避免执行后面的分页语句。
6. 【强制】不得使用外键与级联，一切外键概念必须在应用层解决。说明：（概念解释）学生表中的 student\_id 是主键，那么成绩表中的 student\_id 则为外键。如果更新学生表中的 student\_id，同时触发成绩表中的 student\_id 更新，则为级联更新。外键与级联更新适用于单机低并发，不适合分布式、高并发集群；级联更新是强阻塞，存在数据库更新风暴的风险；外键影响数据库的插入速度。
7. 【强制】禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。

8. **【强制】**数据订正时，删除和修改记录时，要先 select，避免出现误删除，确认无误才能执行更新语句。
9. **【推荐】**in 操作能避免则避免，若实在避免不了，需要仔细评估 in 后边的集合元素数量，控制在 1000 个之内。
10. **【参考】**如果有全球化需要，所有的字符存储与表示，均以 utf-8 编码，那么字符计数方法注意：说明：SELECT LENGTH("轻松工作"); 返回为 12 SELECT CHARACTER\_LENGTH("轻松工作"); 返回为 4 如果要使用表情，那么使用 utfmb4 来进行存储，注意它与 utf-8 编码的区别。
11. **【参考】**TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少，但 TRUNCATE 无事务且不触发 trigger，有可能造成事故，故不建议在开发代码中使用此语句。说明：TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同。

### 2.3.5ORM 规约

1. **【强制】**在表查询中，一律不要使用 \* 作为查询的字段列表，需要哪些字段必须明确写明。说明：1) 增加查询分析器解析成本。2) 增减字段容易与 resultMap 配置不一致。
2. **【强制】**POJO 类的 boolean 属性不能加 is，而数据库字段必须加 is\_，要求在 resultMap 中进行字段与属性之间的映射。说明：参见定义 POJO 类以及数据库字段定义规定，在 sql.xml 增加映射，是必须的。
3. **【强制】**不要用 resultClass 当返回参数，即使所有类属性名与数据库字段一一对应，也需要定义；反过来，每一个表也必然有一个与之对应。说明：配置映射关系，使字段与 DO 类解耦，方便维护。
4. **【强制】**xml 配置中参数注意使用：#{}, #param# 不要使用 \${} 此种方式容易出现 SQL 注入。
5. **【强制】**iBATIS 自带的 queryForList(String statementName,int start,int size)不推荐使用。说明：其实现方式是在数据库取到 statementName 对应的 SQL 语句的所有记录，再通过 subList 取 start,size 的子集合，线上因为这个原因曾经出现过 OOM。  
 正例：在 sqlmap.xml 中引入 #start#, #size#  

```
Map<String, Object> map = new HashMap<String, Object>();
map.put("start", start);
map.put("size", size);
```
6. **【强制】**不允许直接拿 HashMap 与 Hashtable 作为查询结果集的输出。
7. **【强制】**更新数据表记录时，必须同时更新记录对应的 gmt\_modified 字段值为当前时间。

8. 【推荐】不要写一个大而全的数据更新接口，传入为 POJO 类，不管是不是自己的目标更新字段，都进行 `update table set c1=value1,c2=value2,c3=value3`；这是不对的。执行 SQL 时，尽量不要更新无改动的字段，一是易出错；二是效率低；三是 binlog 增加存储。

9. 【参考】@Transactional 事务不要滥用。事务会影响数据库的 QPS，另外使用事务的地方需要考虑各方面的回滚方案，包括缓存回滚、搜索引擎回滚、消息补偿、统计修正等。

10. 【参考】<isEqual>中的 compareValue 是与属性值对比的常量，一般是数字，表示相等时带上此条件；<isNotEmpty>表示不为空且不为 null 时执行；<isNotNull>表示不为 null 值时执行。

### 2.3.6 公用字段命名规约

1	create_time	创建日期			
2	create_user_id	创建人			
3	update_time	修改日期			
4	update_user_id	修改人			
1	road_code	路线编码	字符串	10	
2	road_name	路线名称	字符串	50	
3	start_stake	起点桩号	数字	7	3
4	end_stake	止点桩号	数字	7	3
5	tech_grade	技术等级	数字	1	0
1	mng_org_code	管养单位编码	字符串	7	
1	dstrct_code	政区编码	字符串	6	
1	DIRECT	行车方向	字符串	2	
2	PAVEMENT_TYPE	路面类型	字符串	4	
1	UPPER_LAYER_NM	上面层名称	字符串	50	
2	UPPER_LAYER_THK	上面层厚度	数字	7	3

3	MID_LAYER_NM	中面层名称	字符串	50	
4	MID_LAYER_THK	中面层厚度	数字	7	3
5	LOWER_LAYER_NM	下面层名称	字符串	50	
6	LOWER_LAYER_THK	下面层厚度	数字	7	3
7	PRY_STRUC_NM	基层名称	字符串	50	
8	PRY_STRUC_THK	基层厚度	数字	7	3
9	PRY_STRUC_NM	底基层名称	字符串	50	
10	PRY_STRUC_THK	底基层厚度	数字	7	3
11	PRY_STRUC_NM	垫层名称	字符串	50	
12	PRY_STRUC_THK	垫层厚度	数字	7	3
1	MAINTAIN_TYPE	养护类型	字符串	50	
2	MAINTAIN_PRJ	养护方案	数字	7	3
3	MAINTAIN_MONEY	养护金额	数字	7	3
4	MAINTAIN_LENGTH	养护里程	数字	7	3
5	MAINTAIN_YEAR	养护年度	数字	4	
1	MQI		数字	3	
2	PQI		数字	3	
3	SCI		数字	3	
4	BCI		数字	3	
5	TCI		数字	3	
1	PQI		数字	3	
2	PCI		数字	3	
3	RQI		数字	3	
4	RDI		数字	3	

5	SRI		数字	3	
6	PSSI		数字	3	
5	XHC	小型货车流量	数字	6	
6	ZHC	中型货车流量	数字	6	
7	DHC	大型货车流量	数字	6	
8	TDH	特大货车流量	数字	6	
9	JZX	集装箱车流量	数字	6	
10	XKC	中小客车流量	数字	6	
11	DKC	大型客车流量	数字	6	
1	MTC	摩托车流量	数字	6	
2	TLJ	拖拉机流量	数字	6	
1	XHCS	小型货车行车速度	数字	4	1
2	ZHCS	中型货车行车速度	数字	4	1
3	DHCS	大型货车行车速度	数字	4	1
4	TDHS	特大货车行车速度	数字	4	1
5	JZXS	集装箱车行车速度	数字	4	1
6	XKCS	中小客车行车速度	数字	4	1
7	DKCS	大型客车行车速度	数字	4	1
1	MTCS	摩托车行车速度	数字	4	1
2	TLJS	拖拉机行车速度	数字	4	1
1		桥梁代码			
2		桥梁名称			
3		中心桩号			
1		结构类型代码			
2		结构类型			
3		材料代码			

4		材料名称			
1		设计荷载等级代码			
2		设计荷载等级			
1		监管单位名称			
2		收费性质代码			
3		收费性质			

## 2.4 工程规约

### 2.4.1 应用路径

1. **【强制】**工程分为 controller, service, dao, entity , 每个目录下可包含对应的业务包。

如 rights(权限), logs(日志), bps(工作流)包等等

比如: com.zgk.framework.controller.log

com.zgk.framework.service.log

com.zgk.framework.dao.log

com.zgk.framework.entity.log

2. **【强制】**前台资源 js 或 html 规则

包统一放入到 src.main.resources 下

其中通用组件包放入到 src.main.resources.static 下

html 业务页面放入到 src.main.resources.templates 下

### 2.4.2应用规范

1. **【强制】** 各包的应用层的代码需严格按照总集提供的例子进行编写, 总集对各包提供了基础父类, 各包在 controller, service, dao 都应实现或继承 BaseController, BaseServiceImpl, JpaRepository

2. **【强制】** 针对 redis 的使用, 各包将数据放入 redis 当中, 应注意 redis key 值的编写, 现提出如下要求:

redis-key :所属系统:所属业务表:用途:值 比如: crms:user:getById:1, 将

redis-key 放入到 properties 文件当中进行读取. 各包不要滥用 redis 缓存, 应分析业务, 了解那些数据是很少变动的数据, 合理设置失效时长。

## 2.4.3 工程包规约

### 2.4.3.1 包名

总集规范了对应的包的命名规则

各包包名，命名规范	
com.platform.common	平台常用包
com.platform.framework	基础框架包
com.platform.example	demo 包 例子
com.cbms.*	新桥开发包
com.checc.*	中咨开发包
com.hpdi.*	公规院
com.zggk.*	中公高科开发包
说明：各包严格按照以上包命名进行，对于各包子目录，可根据业务再扩展一级节点	

### 2.4.3.2 类与接口名

类名和接口使用类意义完整的英文描述，每个英文单词的首字母使用大写、其余字母使用小写的大小写混合法。

示例：OrderInformation, CustomerList, LogManager, LogConfig

### 2.4.3.3 方法名

方法名使用类意义完整的英文描述：第一个单词的字母使用小写、剩余单词首字母大写其余字母小写的大小写混合法。方法名称需满足《命名规约》

示例：public void addNewOrder();

### 2.4.3.4 属性名

属性名使用意义完整的英文描述：第一个单词的字母使用小写、剩余单词首字母大写其余字母小写的大小写混合法。属性名不能与方法名相同。

示例：private customerName;

### 2.4.3.5 常量命名

常量名使用全大写的英文描述，英文单词之间用下划线分隔开，并且使用 `final static` 修饰。

示例：

```
public final static int MAX_VALUE = 1000;
```

名称引用属性名可以和公有方法参数相同，不能和局部变量相同，引用非静态成员变量时使用 `this` 引用，引用静态成员变量时使用类名引用。

示例：

```
public class Person
{
    private String name;
    private static List properties;

    public void setName (String name)
    {
        this.name = name;
    }

    public void setProperties (List properties)
    {
        Person.properties = properties;
    }
}
```

## 2.5 安全规约

1. **【强制】**隶属于用户个人的页面或者功能必须进行权限控制校验。说明：防止没有做水平权限校验就可随意访问、操作别人的数据，比如查看、修改别人的订单。

2. **【强制】**用户敏感数据禁止直接展示，必须对展示数据脱敏。说明：查看个人手机号码会显示成:158\*\*\*\*9119，隐藏中间 4 位，防止隐私泄露。

3. **【强制】**用户输入的 SQL 参数严格使用参数绑定或者 METADATA 字段值限定，防止 SQL 注入，禁止字符串拼接 SQL 访问数据库。

4. **【强制】**用户请求传入的任何参数必须做有效性验证。说明：忽略参数校验可能导致：page size 过大导致内存溢出, 恶意 order by 导致数据库慢查询, 任意重定向, SQL 注入, 反序列化注入



正则输入源串拒绝服务 ReDoS 说明： Java JavaJava 代码用 代码用 正则来验证客户端的输入，有些正则写法验证普通用户输入没有问题，但是如果攻击人员使用的是特殊构造的字符串来验证，有可能导致死循环的效果 。

5. **【强制】**禁止向 HTML 页面输出未经安全过滤或未正确转义的用户数据。

6. **【强制】**表单、AJAX 提交必须执行 CSRF 安全过滤。 说明：CSRF(Cross-site request forgery)跨站请求伪造是一类常见编程漏洞。对于存在 CSRF 漏洞的应用/网站，攻击者可以事先构造好 URL，只要受害者用户一访问，后台便在用户不知情情况下对数据库中用户参数进行相应修

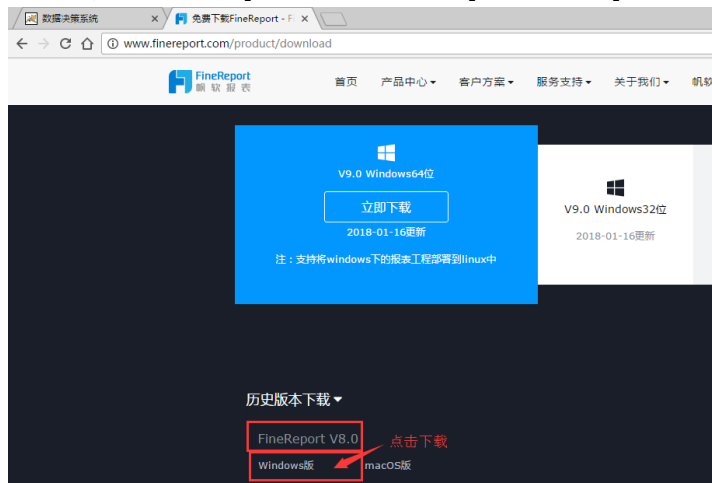
## 第3章 帆软报表系统应用

备注：路网中心项目采用的帆软报表 8.0 版本，后续 8.0 小版本会升级，不会影响各包已有功能的使用。

### 3.1 环境搭建

#### 1. 各包下载帆软 8.0 的设计器

客户端地址：<http://www.finereport.com/product/download>




#### 2. 首次安装需要激活码，激活码如下：

FineReport 8.0 激活码：bf2a3723-e33f54ad4-a6bf-70d72662311d

#### 3. 默认安装完打开。点击 文件 --> 切换工作目录 --> 其他

通用配置		
主机名/IP:	106. 37. 229. 146	
端口:	8075	
WEB 应用:	WebReport	
Servlet:	RemoteServer	
各包配置 登录 http://106. 37. 229. 146:8075/WebReport/RemoteServer?op=fs 及时修改密码		
	用户名	密码
中公高科	crms	crms123
中咨集团	cims	cims123
公规院	chms	chms123
北京新桥	cbms	cbms123

添加对应的远程服务器后，可点击  进行测试。

#### 4. 生成对应的远程服务器目录如下：各包安装会获得对应的操作目录。

01 包 将设计的报告放入到 zgk 和 checc 目录下

02 包 将设计的报告放入到 hpdi 目录下

## 03 包 将设计的报告放入到 cbms 目录下

