目录

[1 系统 3](#_Toc494028851)

[1.1 系统设置 3](#_Toc494028852)

[1.1.1 查找项目基本信息接口（不需要） 3](#_Toc494028853)

[1.1.2 修改项目基本信息接口 3](#_Toc494028854)

[1.1.3 查找系统扩展变量列表接口 3](#_Toc494028855)

[1.1.4 查找系统扩展详情变量接口 5](#_Toc494028856)

[1.1.5 增加/修改系统变量接口 5](#_Toc494028857)

[1.1.6 删除系统变量接口 6](#_Toc494028858)

[1.1.7 修改系统参数-基本设置接口 7](#_Toc494028859)

[1.1.8 修改系统参数-用户设置接口 7](#_Toc494028860)

[1.1.9 修改系统参数-文件设置接口 7](#_Toc494028861)

[1.1.10 修改系统参数-信息设置接口 7](#_Toc494028862)

[1.1.11 修改系统参数-消息推送设置接口 7](#_Toc494028863)

[1.1.12 修改系统参数-支付设置接口 7](#_Toc494028864)

[1.1.13 修改安全参数设置接口 7](#_Toc494028865)

[1.1.14 修改网络防火墙接口 7](#_Toc494028866)

[1.2 数据表与系统模型 7](#_Toc494028867)

[1.2.1 增加/修改数据表接口 7](#_Toc494028868)

[1.2.2 查找数据表列表接口 9](#_Toc494028869)

[1.2.3 查找数据表信息接口 11](#_Toc494028870)

[1.2.4 修改数据表顺序接口 12](#_Toc494028871)

[1.2.5 批量删除数据表接口 13](#_Toc494028872)

[1.2.6 查找字段列表接口 14](#_Toc494028873)

[1.2.7 查找字段信息接口 16](#_Toc494028874)

[1.2.8 字段查重接口 18](#_Toc494028875)

[1.2.9 删除字段接口 19](#_Toc494028876)

[1.2.10 增加/修改字段接口 21](#_Toc494028877)

[1.2.11 查找父表的ID和表名接口 24](#_Toc494028878)

[1.2.12 修改字段顺序接口 25](#_Toc494028879)

[1.2.13 显示字段html代码接口 26](#_Toc494028880)

[1.2.14 查找系统模型列表接口 28](#_Toc494028881)

[1.2.15 查找系统模型信息接口 30](#_Toc494028882)

[1.2.16 增加/修改系统模型接口 32](#_Toc494028883)

[1.2.17 删除系统模型接口 33](#_Toc494028884)

[1.2.18 根据classid查系统模型接口 34](#_Toc494028885)

[1.2.19 查询分表列表接口 34](#_Toc494028886)

[1.2.20 增加分表接口 35](#_Toc494028887)

[1.2.21 删除分表接口 36](#_Toc494028888)

[1.2.22 设置默认系统模型接口 37](#_Toc494028889)

[1.2.23 查询分表记录数接口 38](#_Toc494028890)

[1.2.24 获取不同基础表字段接口 39](#_Toc494028891)

[1.2.25 字段数据转移接口 40](#_Toc494028892)

[1.2.26 内容页html生成接口 41](#_Toc494028893)

[1.2.27 列表页html生成接口 42](#_Toc494028894)

[1.2.28 条件列表页html生成接口 43](#_Toc494028895)

[1.3 工作流模块 44](#_Toc494028896)

[1.3.1 增加工作流模型接口 44](#_Toc494028897)

[1.3.2 修改工作流模型 46](#_Toc494028898)

[1.3.3 修改工作流模型查找信息接口 47](#_Toc494028899)

[1.3.4 查找工作流模型列表接口 48](#_Toc494028900)

[1.3.5 删除工作流模型 49](#_Toc494028901)

[1.3.6 部署工作流模型接口 50](#_Toc494028902)

[1.3.7 执行（完成）一个流程任务接口 51](#_Toc494028903)

[1.3.8 启动（也可以叫申请或产生）一个新的流程接口 53](#_Toc494028904)

[1.3.9 查询某人已完成或未完成的任务 54](#_Toc494028905)

[1.3.10 查询流程实例信息 57](#_Toc494028906)

[1.3.11 查询流程实例的过程（任务列表） 58](#_Toc494028907)

[1.3.12 查询流程定义列表 59](#_Toc494028908)

[1.3.13 查询流程实例列表 60](#_Toc494028909)

[1.3.14 查找节点候选人名单接口 62](#_Toc494028910)

[1.3.15 拾取任务接口 63](#_Toc494028911)

[1.4 附件管理 64](#_Toc494028912)

[1.4.1 增加/修改文件夹 64](#_Toc494028913)

[1.4.2 查找机构根文件夹 64](#_Toc494028914)

[1.4.3 根据pid查找子所有子文件夹 64](#_Toc494028915)

[1.4.4 根据文件夹id查找文件夹下所有附件 64](#_Toc494028916)

[1.4.5 上传单个文件 64](#_Toc494028917)

[1.4.6 传附件和属性接口 66](#_Toc494028918)

[1.4.7 查找附件列表接口 67](#_Toc494028919)

[1.4.8 附件下载 69](#_Toc494028920)

[1.4.9 附件查看 70](#_Toc494028921)

[1.4.10 查询根级目录接口 71](#_Toc494028922)

[1.4.11 裁剪图片接口 73](#_Toc494028923)

[1.5 规则管理 75](#_Toc494028924)

[1.5.1 查找规则列表接口 75](#_Toc494028925)

[1.5.2 新增/修改规则接口 75](#_Toc494028926)

[1.5.3 查找规则详情接口 75](#_Toc494028927)

[1.5.4 删除规则接口 75](#_Toc494028928)

[1.6 信息查询 75](#_Toc494028929)

[1.6.1 查找SQL模板列表接口 75](#_Toc494028930)

[1.6.2 新增/修改SQL模板接口 75](#_Toc494028931)

[1.6.3 查找SQL模板详情接口 75](#_Toc494028932)

[1.6.4 删除SQL模板接口 75](#_Toc494028933)

[1.7 导入数据 75](#_Toc494028934)

[1.7.1 批量导入excel接口 75](#_Toc494028935)

[1.7.2 批量导入word接口 75](#_Toc494028936)

[1.7.3 批量导出excel接口 75](#_Toc494028937)

[1.7.4 批量导出word接口 75](#_Toc494028938)

[1.8 函数 75](#_Toc494028939)

[1.9 对象 76](#_Toc494028940)

[2 暂不实现接口 76](#_Toc494028941)

[2.1 系统设置 76](#_Toc494028942)

[2.2 数据表与系统模型 76](#_Toc494028943)

[2.3 工作流模块 76](#_Toc494028944)

[2.4 附件管理 76](#_Toc494028945)

[2.5 规则管理 76](#_Toc494028946)

[2.6 信息查询 76](#_Toc494028947)

[2.7 数据导入 76](#_Toc494028948)

1. 系统
   1. 系统设置
      1. 增加/修改项目基本信息接口

**a.接口url:**

system/setSystem/basicInformation.do

相关信息：

1. projectnameCN:中文--数据存本地
2. projectnameEN:英文名--数据存本地
3. orgname：机构名称--数据存本地
4. orgid：机构id--数据存本地
5. orglogo：项目logo—文件：pub/admin/assets/images/logo.png
6. orgico：项目ico—文件：pub/admin/assets/images/logo.ico

文字是数据保存本地，图片直接同名覆盖

**b.接口说明:**

增加或修改项目基本信息。

**c. 接口定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口定义 | | | | | |
| 接口名 | | 对象名 | | | 返回值类型 |
| basicInformation | | Info | | | Json |
| 参数说明： | | | |  | |
| 参数名称 | 参数类型 | | 备注 | | |
| projectnameCN | String | | 项目中文名 | | |
| projectnameEN | String | | 项目英文名 | | |
| orgname | String | | 机构名称 | | |
| orgid | String | | 机构id | | |
| orglogo | String | | 机构logo | | |
| orgico | String | | 机构页面图标 | | |

**d.返回值示例与说明**{

{

"result": { //结果

"msg": "操作成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①提交成功后创建本地文档并存入数据。

②修改该文档即全部覆盖之前的文档。

* + 1. 查找系统扩展变量列表接口

**a.接口url:**

system/setSystem/findSysparamList.do

**b.接口说明:**

查找扩展变量列表。

**c. 接口定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口定义 | | | | | |
| 接口名 | | 对象名 | | | 返回值类型 |
| findSysparamList | | Info | | | Json |
| 参数说明： | | | |  | |
| 参数名称 | 参数类型 | | 备注 | | |
| page | String | | 当前页码 | | |
| num | String | | 每页显示多少条数据 | | |

**d.返回值示例与说明**{

"page": { //分页信息

"page": 2, //当前页

"pages": 10, //总页数

"lines": 100, //信息总条数

"pageSize": "10", //每页显示信息数

"pageStart": 1 //起始页

},

"total": 1, //当前页信息数

"infos": [ //信息

{

"id": "1", //系统参数id

"title": "sitename", //参数英文名

"cname": "站点名称", //参数中文名

"content": "学习城测试", //内容

"type": "基本属性", //参数类型

"iscache": "1" //是否写入缓存

}

]

}

**e.功能实现说明**

①检查page参数、为空则设置默认值1，检查num参数、为空则设置默认值30。

②根据page和num生成页码条件（limit）

③组合sql语句查询数据，根据返回值格式进行数据返回。

* + 1. 查找系统扩展变量详情接口

**a.接口url:**

system/setSystem/findSysparamInfo.do

**b.接口说明:**

查找扩展变量详情。

**c. 接口定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口定义 | | | | | |
| 接口名 | | 对象名 | | | 返回值类型 |
| findSysparamInfo | | Info | | | Json |
| 参数说明： | | | |  | |
| 参数名称 | 参数类型 | | 备注 | | |
| classid | String | | 栏目id | | |
| id | String | | 信息id | | |

**d.返回值示例与说明**{

"infos": [ //信息

{

"id": "1", //系统参数id

"title": "sitename", //参数英文名

"cname": "站点名称", //参数中文名

"content": "学习城测试", //内容

"type": "基本属性", //参数类型

"iscache": "1" //是否写入缓存

}

]

}

**e.功能实现说明**

①根据栏目id和信息id 查找信息详情。

* + 1. 增加/修改系统变量接口

**a.接口url:**

system/setSystem/addSysparam.do

**b.接口说明:**

增加扩展变量。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addSysparam | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
|  | string | 参数中文名 |
|  | string | 参数英文名 |
|  | string | 是否写入缓存 |
|  | string | 变量内容 |
| id | string | “”  注意：添加系统变量的时候该字段传空字符串，修改系统变量的时候该字段传信息id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "增加成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①添加系统变量的时候该字段传空字符串，修改系统变量的时候该字段传信息id

②返回更新成功信息

* + 1. 删除系统变量接口

**a.接口url:**

system/setSystem/deleteSysparam.do

**b.接口说明:**

删除指定的扩展变量。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteSysparam | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | 扩展变量id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①接收参数扩展变量id，在系统参数表中通过扩展变量id删除指定的扩展变量。

②根据缓存的key为id清除服务器上application的缓存。

③返回删除成功信息

* + 1. 配置项目系统参数设置-基本设置接口

1. **接口url:**

system/setSystem/addBasicSetting.do

1. **接口说明：**

系统参数中基本信息设置。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addBasicSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 站点名称 |
|  | String | 网站地址 |
|  | String | 附件地址 |
|  | String | 管理员邮箱 |
|  | String | 网站关键字 |
|  | String | 网站简介 |
|  | String | 用户单点登录 |
|  | String | 管理员单点登陆 |
|  | String | 不开启操作的时间点 |
|  | String | 验证码字符组成 |
|  | String | 验证码过期时间 |
|  | String | 验证码配色 |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* + 1. 配置项目系统参数设置-用户设置接口

1. **接口url:**

system/setSystem/addUserSetting.do

1. **接口说明：**

系统参数中用户信息设置。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addUserSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 实时刷新角色菜单 |
|  | String | 后台登录验证码 |
|  | String | 后台登录次数限制 |
|  | String | 重新登录时间间隔 |
|  | String | 登录超时限制 |
|  | String | 注册用户名限制 |
|  | String | 注册密码限制 |
|  | String | 注册用户邮箱唯一性检查 |
|  | String | 同一IP注册间隔限制 |
|  | String | 用户名保留关键字 |
|  | String | 激活帐号链接有效期 |
|  | String | 帐号激活邮件内容 |
|  | String | 开启取回密码功能 |
|  | String | 取回密码链接有效期 |
|  | String | 取回密码邮件内容 |
|  | String | 禁止 IP 访问列表:(前端) : (范围:192.168.0.0～192.168.255.255) |
|  | String | 禁止 IP 访问列表:(后台) :  (范围:192.168.0.0～192.168.255.255) |
|  | String | 允许IP 访问列表:(前端) :  (范围:192.168.0.0～192.168.255.255) |
|  | String | 允许 IP 访问列表:(后台):  (范围:192.168.0.0～192.168.255.255) |
|  | String | 控制动作 |
|  | String | 禁止 IP 提交列表（前台及后台有效）:  (范围:192.168.0.0～192.168.255.255) |
|  | String | 允许 IP 提交列表（前台及后台有效）:  (范围:192.168.0.0～192.168.255.255) |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* + 1. 配置项目系统参数设置-文件设置接口

1. **接口url:**

system/setSystem/addFileSetting.do

1. **接口说明：**

系统参数中用户信息设置。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addFileSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 附件存放目录类型 |
|  | String | 附件存放目录 |
|  | String | 后台上传附件大小 |
|  | String | 后台上传文件扩展名 |
|  | String | 前台投稿附件设置，最大图片大小 |
|  | String | 前台投稿附件设置，图片扩展名 |
|  | String | 前台投稿附件设置，最大附件大小 |
|  | String | 前台投稿附件设置，附件扩展名 |
|  | String | 文件生成权限 |
|  | String | 数据备份存放目录 |
|  | String | 压缩包存放目录 |
|  | String | 备份只选择当前数据库 |
|  | String | 支持MYSQL查询方式 |
|  | String | 默认值 |
|  | String | 超出部分是否截取 |
|  | String | 图片水印 |
|  | String | 水印位置 |
|  | String | 文字水印 |
|  | String | 图片水印，图片文件 |
|  | String | 图片水印，图片质量 |
|  | String | 图片水印，水印透明度 |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* + 1. 配置项目系统参数设置-信息设置接口

1. **接口url:**

system/setSystem/addInfoSetting.do

1. **接口说明：**

系统参数中用户信息设置。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addInfoSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 后台管理信息 |
|  | String | 后台信息默认显示时间范围 |
|  | String | 后台附件默认显示时间范围 |
|  | String | 信息置顶设置 |
|  | String | 动态内容页变量缓存 |
|  | String | 注册用户投稿限制 |
|  | String | 投稿数量限制 |
|  | String | 重复投稿时间限制 |
|  | String | 选择终极栏目的背景颜色 |
|  | String | 九级头条名称 |
|  | String | 九级推荐名称 |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* + 1. 配置项目系统参数设置-消息推送接口

1. **接口url:**

system/setSystem/add MessagePushSetting.do

1. **接口说明：**

系统参数中消息推送设置。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| MessagePushSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 邮件发送模式 |
|  | String | SMTP服务器 |
|  | String | SMTP端口 |
|  | String | 发信人地址 |
|  | String | 发信人昵称 |
|  | String | 是否需要登录验证 |
|  | String | 邮箱登录用户名 |
|  | String | 邮箱登录密码 |
|  | String | 安装形式 |
|  | String | 启用SSL连接 |
|  | String | 被动模式(pasv)连接 |
|  | String | FTP服务器地址 |
|  | String | FTP用户名 |
|  | String | FTP密码 |
|  | String | 传送模式 |
|  | String | FTP传输超时时间 |
|  | String | 系统根目录(FTP) |
|  | String | 测试FTP服务器 |
|  | String | 短信服务商 |
|  | String | alidayu\_sms\_key |
|  | String | alidayu\_sms\_sign\_method |
|  | String | alidayu\_sms\_sign |
|  | String | alidayu\_sms\_free\_sign\_name |
|  | String | alidayu\_sms\_param |
|  | String | alidayu\_sms\_template\_code |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* + 1. 配置项目系统参数设置-支付设置接口（没用到）
    2. 配置项目安全参数设置接口

1. **接口url:**

system/setSystem/securityParamSetting.do

1. **接口说明：**

配置项目安全参数。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| securityParamSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 后台安全相关配置 |
|  | String | 后台登录认证码 |
|  | String | 后台开启验证登录IP |
|  | String | 记录登陆日志 |
|  | String | 记录操作日志 |
|  | String | 开启访问来源 |
|  | String | 开启后台来源认证码 |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* + 1. 配置项目网络防火墙设置接口

1. **接口url:**

system/setSystem/firewallSetting.do

1. **接口说明：**

系统参数中用户信息设置。

1. **接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| firewallSetting | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| id | String | “” |
|  | String | 开启防火墙 |
|  | String | 防火墙加密密钥 |
|  | String | 允许后台登录的域名 |
|  | String | 信息置顶设置允许登陆后台的时间点 (不选为不限制) |
|  | String | 态内动容页变量缓存允许登陆后台的星期 (不选为不限制) |
|  | String | 防火墙后台预登陆验证变量名 |
|  | String | 防火墙后台登录认证码 |
|  | String | 屏蔽提交敏感字符 (多个用半角逗号格开; 设置屏蔽前台所有提交内容及后台登陆内容) |

1. **返回值示例与说明:**

{

"result": { //结果

"msg": "设置成功", //信息

"flag": "1" //标识

}

}

1. **功能实现说明：**

①系统创建id

②返回设置成功信息

* 1. 数据表与系统模型
     1. 增加/修改数据表接口

注意：1.数据表表名只能由只能由字母、数字、下划线组成；

**a.接口url:**

system/tableModel/addOrUpdateTable.do

**b.接口说明:**

（1）如果传入zjxxEnewstable对象的tid为空，则为增加数据表接口。

（2）如果传入zjxxEnewstable对象的tid不为空，则为修改数据表接口。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addOrUpdateTable | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| zjxxEnewstable | zjxxEnewstable | 表名 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "操作成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

（1）如果zjxxEnewstable对象的id为空：

①传入参数zjxxEnewstable的对象，生成UUID作为zjxxEnewstable对象的id属性，设置默认顺序值为0。

② 判断zjxxEnewstable对象中的ismaster字段值为1或0：

1. 如果ismaster = 1，则待建的这个表为主表，执行③④⑤⑥
2. 如果 ismaster = 0，则待建的这个表为分表，执行③④⑤⑥，然后根据 zjxxEnewstable对象中的主表表名字段masterb更新zjxx\_enewstable表中tbname字 段值为zjxxEnewstable对象中的主表表名字段值的数据，更新其分表集合字段datatbs， 也就是将zjxxEnewstable对象中的id字段拼接到datatbs字段中（多个分表ID用逗号 隔开）

③ 根据zjxxEnewstable对象建立表a、归档表a\_gd、微客表a\_wk，其中a为zjxxEnewstable对象中的字段tbname的值。

④ 为表a 、归档表a\_gd、微客表a\_wk添加系统默认字段，系统默认字段记录在指定的txt文本中，json格式存储（字段名以及字段的属性）。

⑤ 读取文件（服务器本地文件），获取系统可录入项系统字段，将这些字段记录到模型字段表zjxx\_enewsf （模型字段表）中，并且与表a相关联

⑥ a\_wk表额外增加weike\_id（varchar48）、modifyauthor(varchar100)、modifytime(double)三个字段。将zjxxEnewstable对象数据插入zjxx\_enewstable表中，返回result。

（2）如果zjxxEnewstable对象的id不为空：

①接收参数zjxxEnewstable对象，修改zjxx\_enewstable表中id字段值为zjxxEnewstable对象中id字段值的数据，修改为对象参数的对应字段数据。

②返回result。

注：修改数据表的时候，只允许修改描述字段，新建的数据表的表中字段以及字段属性都一样。

* + 1. 查找数据表列表接口

**a.接口url:**

system/tableModel/findTableList.do

**b.接口说明:**

查找数据表列表数据。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findTableList | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| page | Int | 1，当前页码 |
| num | Int | 10，每页显示多少条数据 |
| tbname | String | 数据表表名 |

**d.返回值示例与说明**

{

"page": { //分页信息

"page": 2, //当前页

"pages": 10, //总页数

"lines": 100, //信息总条数

"pageSize": "10", //每页显示信息数

"pageStart": 1 //起始页

},

"total": 1, //当前页信息数

"infos": [ //信息

{

"datatbs": "zjxx\_t2\_1,", //分表集合

"ismaster": true, //是否是主表

"parenttb": "", //父表

"sontb": "zjxx\_t2\_son,", //子表集合

"tbname": "zjxx\_t2", //数据表名

"tbtype": 6, //表类型

"tid": "338055a8-15aa-4174-ac7b-0a65148125fb", //数据表ID

"tname": "t2", //数据表标识

"tsay": "t2", //备注

"tbs": "name,age,", //对应字段

"masterb": "", //主表名称

"torder": "1", //顺序

"yhid": "1", //优化方案ID

"orgid": "1" //机构

}

]

}

**e.功能实现说明**

1. 检查page参数、为空则设置默认值1，检查num参数、为空则设置默认值30。
2. 检查tbname参数，若非空则根据tbname进行模糊查询（like）、查找相似表名数据
3. 根据page和num生成页码条件（limit）
4. 根据以上条件查询zjxx\_enewstable （模型数据表）查询数据，返回数据Info。
   * 1. 查找数据表信息接口

**a.接口url:**

system/tableModel/findTable.do

**b.接口说明:**

按条件查找数据表数据。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findTable | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tid | String | 数据表id |
| tbname | String | 数据表表名 |

**d.返回值示例与说明**

{

"result": { //识别结果

"msg": "查询成功", //结果信息

"flag": "1" //1为成功，0为失败

}

"infos" :{

"datatbs": "zjxx\_t2\_1,", //分表集合

"ismaster": true, //是否是主表

"parenttb": "", //父表

"sontb": "zjxx\_t2\_son,", //子表集合

"tbname": "zjxx\_t2", //数据表名

"tbtype": 6, //表类型

"tid": "338055a8-15aa-4174-ac7b-0a65148125fb", //数据表ID

"tname": "t2", //数据表标识

"tsay": "t2", //备注

"tbs": "name,age,", //对应字段

"masterb": "", //主表名称

"torder": "1", //顺序

"yhid": "1", //优化方案ID

"orgid": "1" //机构

}

}

**e.功能实现说明**

1. 分别检测参数tbname、tid、若两个参数皆为空值、则返回“来源错误”信息。
2. 若tid非空值、则拼接查询条件、查询zjxx\_enewstable （模型数据表）中tid为参数tid的数据
3. 若tbname非空值并且tid为空值或为NULL、则拼接查询条件、使用精确查询查询查询zjxx\_enewstable （模型数据表）中tbname为参数tbname的数据
4. 根据生成的查询条件、查询对应数据库、找到数据后进行返回。
5. 若找不到数据则提示“无对应数据”信息
   * 1. 修改数据表顺序接口

**a.接口url:**

system/tableModel/updateTableOrder.do

**b.接口说明:**

修改、批量修改数据表显示顺序。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| updateTableOrder | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| orders | String[] | 需要修改数据表的序号的数组 |
| tids | String[] | 数据表id的数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "修改成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①接收参数tids和orders，遍历tids数组，每一条tids[i]对应一个orders[i],生成sql语句

“UPDATE zjxx\_enewstable SET torder = ‘orders[i]’ WHERE tid = ‘tids[i]’”

②执行sql语句更新数据库，返回result。

* + 1. 批量删除数据表接口

注意：数据表的删除是真删除

**a.接口url:**

system/tableModel/deleteTable.do

**b.接口说明:**

删除、批量删除数据表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteTable | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tids | String[] | 数据表id数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①传入参数tids，遍历tids得到每个tid，使用finById(tid)找到该表数据，得到该表的父表数据，得到该表的表名tbname，修改其父表的子表集合。

②使用deleteByRecursion(tbname)方法递归删除该表下的系统模型、删除字段表中属于该表的字段、删除该表数据、删除数据库中的该表、删除该表的归档和微客表。

③得到该表的分表集合，遍历分表集合删除所有分表数据、删除分表字段、删除数据库中分表、删除分表的归档表和微客表。

④得到该表的子表集合，如果子表集合不为空，遍历子表集合，每次将一个子表的表名sontbname作为递归参数再次传入deleteByRecursion(sontbname)进行下一轮递归。

* + 1. 查找字段列表接口

**a.接口url:**

system/tableModel/findFieldList.do

**b.接口说明:**

查询字段数据列表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findFieldList | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tid | String | 数据表id |

**d.返回值示例与说明**

{

"infos": [ //信息

{

"f": "name", //字段名

"fform": "linkfield", //输入表单显示元素

"fhtml": "classid=<?=$classid?>", //输入表单替换html代码

"fid": "08a049db-2815-4004-91a3-c232c4bb0830", //字段ID

"flen": "255", //字段长度

"fname": "姓名", //字段标识

"ftype": "VARCHAR", //字段类型

"fvalue": "", //字段初始值

"fzs": "副标题", //注释

"isadd": false, //是否是系统字段

"iscj": false, //是否为采集项

"iskey": false, //是否为索引字段

"isonly": false, //字段值是否唯一

"ispage": false, //是否分页字段

"isshow": false, //是否显示字段

"issmalltext": false, //是否简介字段

"jsonstring": "{/"fformsize/":/"60/",/"fformwidth/":/"/",/"fformheight/":/"/",/"linkfieldtb/":/"/",/"linkfieldshow/":/"/",/"linkfieldval/":/"/",/"samedata/":/"1/",/"editorys/":/"0/"}", //

"qfhtml": "<input name=/"zjxxEnewsf.getF()/" type=/"text/" id=/"zjxxEnewsf.getF()/" value=/"<?=$ecmsfirstpost==1?/"/":ehtmlspecialchars(stripSlashes($r[zjxxEnewsf.getF()]))?>/" size=/"45/">/n <a onclick=/"window.open('ecmseditor/FileMain.php?type=2", //选项

"savetxt": false, //是否内容存文本

"tbdataf": 0, //是否为分表字段

"tbname": "zjxx\_t2", //数据表明

"tid": "338055a8-15aa-4174-ac7b-0a65148125fb" //数据表ID

}

],

[{…},{…}]

}

**e.功能实现说明**

①检查参数 tid ，若为空则提示返回“来源错误”

②根据tid查找对应数据表数据。将该表的表名、以及分表名存入一个数组tbnames[]

③遍历tbnames[]，依次根据各个表名再去获取对应字段的列表。存入返回值对象。

返回一个Info对象

* + 1. 查找字段信息接口

**a.接口url:**

system/tableModel/findField.do

**b.接口说明:**

按条件查询字段数据。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findField | Infor | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tid | String | 数据表id |
| fid | String | 字段id |
| f | String | 字段名 |

**d.返回值示例与说明**

{infos:[ {

"f": "name", //字段名

"fform": "linkfield", //输入表单显示元素

"fhtml": "classid=<?=$classid?>", //输入表单替换html代码

"fid": "08a049db-2815-4004-91a3-c232c4bb0830", //字段ID

"flen": "255", //字段长度

"fname": "姓名", //字段标识

"ftype": "VARCHAR", //字段类型

"fvalue": "", //字段初始值

"fzs": "副标题", //注释

"isadd": false, //是否是系统字段

"iscj": false, //是否为采集项

"iskey": false, //是否为索引字段

"isonly": false, //字段值是否唯一

"ispage": false, //是否分页字段

"isshow": false, //是否显示字段

"issmalltext": false, //是否简介字段

"jsonstring": "{/"fformsize/":/"60/",/"fformwidth/":/"/",/"fformheight/":/"/",/"linkfieldtb/":/"/",/"linkfieldshow/":/"/",/"linkfieldval/":/"/",/"samedata/":/"1/",/"editorys/":/"0/"}", //

"qfhtml": "<input name=/"zjxxEnewsf.getF()/" type=/"text/" id=/"zjxxEnewsf.getF()/" value=/"<?=$ecmsfirstpost==1?/"/":ehtmlspecialchars(stripSlashes($r[zjxxEnewsf.getF()]))?>/" size=/"45/">/n <a onclick=/"window.open('ecmseditor/FileMain.php?type=2", //选项

"savetxt": false, //是否内容存文本

"tbdataf": 0, //是否为分表字段

"tbname": "zjxx\_t2", //数据表明

"tid": "338055a8-15aa-4174-ac7b-0a65148125fb" //数据表ID

}

]

}

**e.功能实现说明**

1. 检查参数、若fid为空且f和tid有一个及以上为空值时、返回“来源错误”信息。
2. 若fid非空值、则拼接查询条件、使用精确查询（=）
3. 若tid非空值、则拼接查询条件、使用精确查询（=）
4. 若f非空值、则拼接查询条件、使用精确查询（=）
5. 根据生成的查询条件、查询对应数据库、找到数据后进行返回。
6. 若找不到数据则提示“无对应数据”信息
7. 返回infor
   * 1. 字段查重接口（没用到）

**a.接口url:**

system/tableModel/findSameField.do

**b.接口说明:**

添加和修改字段时，按条件查询字段数据判断是否重复。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findSameField | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tbname | String | 数据表名 |
| tid | String | 数据表id |
| fid | String | 字段id |
| f | String | 字段名 |

**d.返回值示例与说明**

{

"flag": "1", //标识：2重复，1成功 ，0出错；

" msg": "", //

}

**e.功能实现说明**

1. 检查参数、若f和tbname有一个及以上为空值时、返回“来源错误”信息。
2. 根据tbname找到其主表和所有分表并根据并f和tbnames拼接查询条件查找数据库。
3. 若fid为空值、判断查找结果是否为空，若存在则设置flag=2，不存在设置flag=1
4. 若fid非空值、判断查找结果是否为空，若存在则判断其fid是否等于fid，若相等，则设置flag=1，若不相等，则设置flag=2；若不存在设置flag=1；
5. 查询失败，设置flag=1，返回Result。
   * 1. 删除字段接口

**a.接口url:**

system/tableModel/deleteField.do

**b.接口说明:**

删除字段。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteField | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| fid | String | 字段id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

**e.功能实现说明**

① 通过fid找到对应的字段数据，得到字段名f、以及字段所属数据表tbname。

② 判断tbname是否为主表（根据字段数据的tid在模型数据表中查找该数据表记录， 查看记录的Masterb字段是否为空，若不为空，则则将Masterb赋给tbname）

③ 通过得到的tbname在系统模型表中查找所有tbname字段值为tbname的list对象集合，遍历该list集合。查找每一记录中的以上字段：mtemp、cj、enter、tempvar、searchvar、qenter、mustqenterf、listandf、listtempvar、canaddf、caneditf、orderf、uk、enterjson是否存在原字段信息，若存在则删除各字段中的原字段的信息（在删除时，注意连后面的“，”一并删除，可以使用replace替换为空），并更新系统模型表记录

④在数据库中删除tbname表中的字段f，删除对应归档表、微客表中的字段f。

⑤删除zjxx\_enewsf表中该符合该fid的字段数据。

* + 1. 增加/修改字段接口

注意：1.修改字段时禁止分表改主表。

2.主表如果有分表则调用接口把分表也查找出来,供选择基本设置中存放表使用。

**a.接口url:**

system/tableModel/addOrUpdateField.do

**b.接口说明:**

（1）如果传入参数zjxxEnewsf对象的fid为空，则为增加字段接口。

（2）如果zjxxEnewsf对象的fid不为空，则为修改字段、转移字段接口。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addOrUpdateField | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| zjxxEnewsf | ZjxxEnewsf | 字段对象 |
| tid | String | 数据表id |
| toTbname | String | 字段转移的目标表名 |
| fid | String | 字段id  注意：若为增加字段，则该字段传空字符串；若为修改字段，则该字段传该条信息的id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "修改成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

（1）判断zjxxEnewsf对象的flen为空，则设置默认字段长度为255

（2）如果传入参数zjxxEnewsf对象的fid为空：为增加字段

①、生成uuid，并将其设置为字段的fid；

②、判断fthml是否为空或空字符串，若为空，则调用showHtml接口，设置其zjxxEnewsf的fhtml和qhtml的值。

③接收参数zjxxEnewsf对象，得到该字段的tbname。

④得到字段名、字段类型、字段长度、是否为空、是否为主键等信息。

⑤、判断zjxxEnewsf的ftype字段类型是否为TEXT，若是，则单独拼接sql语句（不考虑字段的值唯一和加索引属性）往数据库tbname表中添加该字段，往tbname的归档表和微客表中添加该字段。

⑥、判断zjxxEnewsf的ftype字段类型是否为Longtext、mediumtext，若是，则单独拼接对应的sql语句（不考虑字段的值唯一和加索引属性）往数据库tbname表中添加该字段，往tbname的归档表和微客表中添加该字段。

⑦、判断zjxxEnewsf的ftype字段类型是否有长度，分为有长度与无长度。

再判断字段的值isonly是否唯一,

根据不同的条件，拼接不同的sql语句。

分别给数据库tbname表中添加该字段，往tbname的归档表和微客表中添加该字段。

⑧、判断zjxxEnewsf的值不唯一 且 加索引 且 ftype的值不等于text、mediumtext、longtext时，给数据表对应的字段添加索引，生成sql语句。往数据库tbname表中添加该字段，往tbname的归档表和微客表中添加该字段。

~~③生成添加字段的sql语句，往数据库tbname表中添加该字段，往tbname的归档表和微客表中添加该字段。~~

⑨将zjxxEnewsf对象数据插入zjxx\_enewsf表中。

10.返回result。

（2）如果传入参数zjxxEnewsf对象的fid不为空：

①如果toTbname为空，则为修改字段

1. 通过zjxxEnewsf对象的fid在字段表中查找该字段的原数据oldzjxxEnewsf
2. 判断修改后的字段zjxxEnewsf是否有长度，拼接sql语句，修改数据表、归档表以及威客表中对应的字段
3. 若oldzjxxEnewsf的ftype不等于text、Longtext、mediumtext时，判断修改前字段oldzjxxEnewsf的属性是否存在值唯一或索引，若存在，则先拼接sql语句，去除字段的索引属性，修改数据表、归档表以及威客表中对应的字段
4. 若zjxxEnewsf的值唯一，且zjxxEnewsf的字段类型ftype不等于text、Longtext、mediumtext时，给字段添加唯一属性，修改数据表、归档表以及威客表中对应的字段
5. 若zjxxEnewsf的值不唯一且要加索引且zjxxEnewsf字段类型ftype不等于text、Longtext、mediumtext时，给字段添加索引属性，修改数据表、归档表以及威客表中对应的字段。

（以下6、7）为修改系统模型表操作

1. 再通过oldzjxxEnewsf的该字段的tbname，并判断其是否为主表（根据oldzjxxEnewsf的tid在模型数据表中查找该数据表记录，查看记录的Masterb字段是否为空，若不为空，则则将Masterb赋给tbname）
2. 通过得到的tbname在系统模型表中查找所有tbname字段值为tbname的list对象集合，遍历该list集合。查找每一记录中的以上字段：mtemp、cj、enter、tempvar、searchvar、qenter、mustqenterf、listandf、listtempvar、canaddf、caneditf、orderf、uk、enterjson是否存在原字段信息，若存在则根据zjxxEnewsf信息依次替换（可以使用replace替换），并更新系统模型表记录

②~~通过zjxxEnewsf得到tbname。~~

③更新zjxx\_enewsf表中的字段信息。

~~④修改数据库tbname表、tbname\_gd表和tbname\_wk表中对应字段的属性，返回result。~~

④、返回result；

⑤如果toTbname不为空，则：为转移字段

⑥通过zjxxEnewsf得到原数据表tbname。

⑦、设置zjxxEnewsf2为值为zjxxEnewsf，并修改其fid为空，再修改其tbname为toTbname，tid为相对与的值，调用添加字段接口，往数据库toTbname表中添加zjxxEnewsf字段，往数据库toTbname的归档表和微客表中添加zjxxEnewsf字段。

~~⑦往数据库toTbname表中添加zjxxEnewsf字段，往数据库toTbname的归档表和微客表中添加zjxxEnewsf字段。~~

（8、9）为转移字段时要修改的系统模型表信息

⑧ 通过zjxxEnewsf的该对象的tbname，并判断其是否为主表（根据zjxxEnewsf的tid在模型数据表中查找该数据表记录，查看记录的Masterb字段是否为空，若不为空，则则将Masterb赋给tbname）

⑨通过得到的tbname在系统模型表中查找所有tbname字段值为tbname的list对象集合，遍历该list集合。查找每一记录中的以上字段：mtemp、cj、enter、tempvar、searchvar、qenter、mustqenterf、listandf、listtempvar、canaddf、caneditf、orderf、uk、enterjson中是否存在zjxxEnewsf字段信息，若存在则根据totbname修改每个字段值中的tbname（可以使用replace替换），并更新系统模型表记录

⑨返回result。

\*字段转移逻辑中，前台人员还需调用信息转移接口和字段删除接口以完成全部逻辑。

* + 1. 修改字段顺序接口

**a.接口url:**

system/tableModel/updateFieldOrder.do

**b.接口说明:**

修改、批量修改字段顺序。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| updateFieldOrder | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| fid | String[] | 字段id的数组 |
| myorder | String[] | 字段显示顺序的数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "修改成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①接收参数fid和myorder数组，遍历fid数组，得到fid[i]，结合对应myorder[i]生成sql语句

“UPDATE zjxx\_enewsf SET myorder = ‘myorder[i]’ WHERE fid = ‘fid[i]’”

②执行该sql更新数据库，返回result。

* + 1. 显示字段html代码接口（没用到）

**a.接口url:**

system/tableModel/showHtml.do

**b.接口说明:**

利用freemarker读取模板html.txt文件，并根据前台传入的值替换模板中的值，生成html代码。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| showHtml | String | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| fform | String | 输入表单显示元素 |
| f | String | 字段名 |
| jsonstring | String | 选项json字符串，记录不同fform需要的不同选项 |
| fvalue | String | 字段初始值 |

**d.返回值示例与说明**

{

"html": "<input name=/"zjxxEnewsf.getF()/" type=/"text/" id=/"zjxxEnewsf.getF()/" value=/"<?=$ecmsfirstpost==1?/"/":ehtmlspecialchars(stripSlashes($r[zjxxEnewsf.getF()]))?>/" size=/"45/">/n <a onclick=/"window.open('ecmseditor/FileMain.php?type=2" //html代码

}

**e.功能实现说明**

①接收参数fform、f、jsonstring、fvalue。

②根据fform从html.txt文本中读取对应的默认html，例如fform=password，从模板html.txt中读取形如“<#elseif (fform)=="password">”的默认html。

③根据f、fform的值封装在Map中。

④当fform值为select、radio、checkbox时需要对fvalue进行解析（此时fvalue：value==key多个值之间用”\n”分割），取出每一组key、value。将其封装到Map中

~~⑤当fform值为dictionaries、dictionarylist时需要对解析jsonstring，得到其中的fformsize，通过fformsize的值（即为数据词典的fid）来调用数据词典controller中的getTitles（）接口，得到数据词典关联单选或多选的titles字符串（之间用“，”分割），解析titles字符串得到一组title作为单选或多选的值，再将其封装到Map中。~~

⑥当fform值为classselectinfo、roles、labels、people、institutions和dictionaries时，为关联字段，jsonstring中的fformsize即为关联栏目信息的classid，角色关联的父id，机构关联的父id，数据词典关联的父id，将其封装到Map中。（在后面的录入信息时，在添加信息页面将调用html，生成关联字段的button，点击其中的button，将分别调用要查找的关联信息接口，将返回值在新的弹出div中显示出来，以供添加信息时选择，选中那条信息，则返回id）。

⑥新添加了发布时间、所属机构、标题图片、标题四大基础系统字段的输出类型。（主要是为了解决系统字段被修改输出类型后，一样可以还原成系统字段的输出类型）

⑦解析jsonstring，jsonstring:{"fformsize":60,"fformwidth":"","fformheight":"","linkfieldtb":"","linkfieldshow":"","linkfieldval":"","samedata":"","editorys":""}其中jsonstring共包含8个字段，分别代表：表单元素长度；元素宽度，高度；关联显示值字段名；关联数据表名；关联显示字段；关联是否数据同步；编辑器样式；

可将其转化为json对象，获取其中对应的值，将需要用到的数据封装到Map中。

⑧利用freemark模板中的数据用Map中的值替换（例如Map中的键为key，将会自动替换模板中的${key}）。

⑨返回修改后的html。

* + 1. 查找系统模型列表接口

**a.接口url:**

system/tableModel/findModelList.do

**b.接口说明:**

该接口用于查找系统模型列表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findModelList | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tbname | String | 数据表名 |
| tid | String | 数据表id |
| page | String | 页码 |
| num | String | 每页显示信息数 |

**d.返回值示例与说明**

{

"page": { //分页信息

"page": 2, //当前页

"pages": 10, //总页数

"lines": 100, //信息总条数

"pageSize": "10", //每页显示信息数

"pageStart": 1 //起始页

},

"total": 1, //当前页信息数

"infos": [ //信息

{

"canaddf": "表名:字段名, 表名:字段名, 表名:字段名", //可增加项

"caneditf": "表名:字段名, 表名:字段名, 表名:字段名", //可修改项

"cj": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}", //采集项

"definfovoteid": 0, //信息预设投票ID

"enter": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}特殊属性${a\_specialfield}发布时间${a\_newstime}标题图片${a\_titlepic}", //录入项

"isdefault": false, //是否默认模型

"listandf": "表名:字段名, 表名:字段名, 表名:字段名", //结合项

"listfile": "", //信息列表名称

"listtempvar": "表名:字段名, 表名:字段名, 表名:字段名", //列表模板项

"mid": "fc943551-a5d3-4603-9dcf-08d0506bb823", //模型ID

"mname": "t2的系统模型1", //模型名称

"mtemp": "[{/"fileds/":/"account,password,a\_title,a\_ftitle,a\_specialfield,a\_newstime,a\_titlepic/",/"htmls/":/"用户名:<input type='condlinkfield' name='account' value=''/><br>密码:<input type='password' name='password' value=''/><br>标题:<input type='text' name='a\_title' value=''/><br>副标题:<input type='text' name='a\_ftitle' value=''/><br>特殊属性:<input type='text' name='a\_specialfield' value=''/><br>发布时间:<input type='text' name='a\_newstime' value=''/><br>标题图片:<input type='text' name='a\_titlepic' value=''/>/"}]", //录入表单模板

"mustqenterf": "account,password,a\_title,a\_ftitle,a\_specialfield,a\_newstime,a\_titlepic,", //必填项

"myorder": 0, //排序

"mzs": "", //注释

"orderf": "account,password,a\_title,a\_ftitle,", //排序项

"printtempid": 0, //打印模板ID

"qenter": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}", //投稿项

"qmname": "t2的系统模型1", //模型别名

"searchvar": "account,password,a\_title,a\_ftitle,", //搜索项

"setandf": false, //结合相匹配

"showmod": true, //是否显示到前台导航

"tbname": "zjxx\_t2", //数据表名

"tempvar": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}", //内容模板项

"tid": "338055a8-15aa-4174-ac7b-0a65148125fb", //数据表ID

"uk": "account,password,a\_title,a\_ftitle,", //联合主键

"usemod": true //是否启用

}

]

}

**e.功能实现说明**

1. 检查page参数、为空则设置默认值1，检查num参数、为空则设置默认值30。
2. 检查tbname参数，若非空则根据tbname的值拼接查询条件、使用精确查询（=）
3. 根据page和num生成页码条件（limit）
4. 根据生成的sql语句查询数据，返回数据。
   * 1. 查找系统模型信息接口

**a.接口url:**

system/tableModel/findModel.do

**b.接口说明:**

通过不同方式查找一个系统模型。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findModel | ZjxxModel | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| mid | String | 系统模型id |
| mname | String | 系统模型名称 |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，1为成功，0为操作失败

"msg": "配置员工角色成功" //返回的信息；

}

“infos”：{

"canaddf": "表名:字段名, 表名:字段名, 表名:字段名", //可增加项

"caneditf": "表名:字段名, 表名:字段名, 表名:字段名", //可修改项

"cj": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}", //采集项

"definfovoteid": 0, //信息预设投票ID

"enter": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}特殊属性${a\_specialfield}发布时间${a\_newstime}标题图片${a\_titlepic}", //录入项

"isdefault": false, //是否默认模型

"listandf": "表名:字段名, 表名:字段名, 表名:字段名", //结合项

"listfile": "", //信息列表名称

"listtempvar": "表名:字段名, 表名:字段名, 表名:字段名", //列表模板项

"mid": "fc943551-a5d3-4603-9dcf-08d0506bb823", //模型ID

"mname": "t2的系统模型1", //模型名称

"mtemp": "[{/"fileds/":/"account,password,a\_title,a\_ftitle,a\_specialfield,a\_newstime,a\_titlepic/",/"htmls/":/"用户名:<input type='condlinkfield' name='account' value=''/><br>密码:<input type='password' name='password' value=''/><br>标题:<input type='text' name='a\_title' value=''/><br>副标题:<input type='text' name='a\_ftitle' value=''/><br>特殊属性:<input type='text' name='a\_specialfield' value=''/><br>发布时间:<input type='text' name='a\_newstime' value=''/><br>标题图片:<input type='text' name='a\_titlepic' value=''/>/"}]", //录入表单模板

"mustqenterf": "account,password,a\_title,a\_ftitle,a\_specialfield,a\_newstime,a\_titlepic,", //必填项

"myorder": 0, //排序

"mzs": "", //注释

"orderf": "account,password,a\_title,a\_ftitle,", //排序项

"printtempid": 0, //打印模板ID

"qenter": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}", //投稿项

"qmname": "t2的系统模型1", //模型别名

"searchvar": "account,password,a\_title,a\_ftitle,", //搜索项

"setandf": false, //结合相匹配

"showmod": true, //是否显示到前台导航

"tbname": "zjxx\_t2", //数据表名

"tempvar": "用户名${account}密码${password}标题${a\_title}副标题${a\_ftitle}", //内容模板项

"tid": "338055a8-15aa-4174-ac7b-0a65148125fb", //数据表ID

"uk": "account,password,a\_title,a\_ftitle,", //联合主键

"usemod": true //是否启用

}

}

**e.功能实现说明**

1. 检查参数mid、mname,若均为空、则返回“来源错误”
2. 若mid非空、则拼接加入查询条件、使用精确查询（=）
3. 若mname非空、则拼接加入查询条件、使用精确查询（=）
4. 根据组合的查询条件查询数据库、返回对应数据
   * 1. 增加/修改系统模型接口

**a.接口url:**

system/tableModel/addOrUpdateModel.do

**b.接口说明:**

（1）如果传入的mid为空，则该接口用于增加、复制系统模型。

（2）如果传入的mid不为空，则该接口用于修改系统模型。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addOrUpdateModel | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| mid | String | 系统模型id  注意：如果传入的mid为空，则该接口用于增加、复制系统模型；如果传入的mid不为空，则该接口用于修改系统模型。 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "增加成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

（1）如果传入参数zjxxModel对象的mid为空：

①接收参数zjxxModel对象，生成UUID并赋值给zjxxModel对象的mid属性。

②将zjxxModel对象插入zjxx\_model表中，返回result。

（2）如果传入参数zjxxModel对象的mid不为空：

①接收参数zjxxModel对象，根据zjxxModel对象的属性更新zjxx\_model表中的数据，返回result。

* + 1. 删除系统模型接口

**a.接口url:**

system/tableModel/deleteModel.do

**b.接口说明:**

该接口用于删除系统模型

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findModel | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| mid | String | 系统模型id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①传入参数mid，在栏目表查询是否有栏目绑定此系统模型，如果有，提示不能删除。

②如果没有，执行SQL语句“DELETE FROM zjxx\_model WHERE mid = ‘mid’;”删除对应mid的系统模型，返回result。

* + 1. 根据classid查系统模型接口(没有使用)

**a.接口url:**

system/tableModel/findModelByClassid.do

**b.接口说明:**

该接口用于通过classid查找系统模型数据。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findModelByClassid | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |

**d.返回值示例与说明**

//信息

{

“infos”：{……}

}

**e.功能实现说明**

1. 接收参数classid，为空值则返回“来源错误”
2. 根据classid找到栏目信息中的系统模型mid。
3. 根据通过mid找到系统模型信息。
4. 返回info。
   * 1. 查询分表列表接口

**a.接口url:**

system/tableModel/findPartTableList.do

**b.接口说明:**

该接口用于通过一个主表的tbname查询所有分表列表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findPartTableList | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tbname | String | 主表数据表名 |

**d.返回值示例与说明**

{

"infos": [ //信息

{

"ismaster": false, //是否为主表

"masterb": "zjxx\_t2", //主表名

"tbname": "zjxx\_t2\_1", //表名

"tid": "b7683f90-5bff-42e0-b063-7002eb1af66e" //表ID

},

{…}

]

}

**e.功能实现说明**

①检查参数tbname，若为空则返回信息“来源错误”

②非空则根据tbname查找信息。若是主表、则直接查询所有分表信息。若是分表、则找到对应主表后在查询所有分表信息

③最终返回主表于其全部分表的信息。

* + 1. 增加分表接口

**a.接口url:**

system/tableModel/addPartTable.do

**b.接口说明:**

该接口用于增加分表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addPartTable | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| masterb | String | 主表名称 |
| tbname | String | 分表数据表名 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "增加成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①接收参数masterb和tbname，生成一个ZjxxEnewstable对象zjxxEnewstable。

②生成UUID，将UUID、masterb、tbname赋值给zjxxEnewstable对象，设置zjxxEnewstable对象的ismaster属性为false。

③将zjxxEnewstable对象插入zjxx\_enewstable数据表。

④在数据库建立名为tbname、tbname\_gd、tbname\_wk的三张表。

⑤通过masterb找到该分表的主表数据，通过主表数据得到主表的分表集合datatbs，将该tbname拼接到datatbs中。

⑥更新主表数据，返回result。

* + 1. 删除分表接口

**a.接口url:**

system/tableModel/deletePartTable.do

**b.接口说明:**

该接口用于删除一个分表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deletePartTable | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tid | String | 分表id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①接收参数tid，通过tid找到符合的分表对象zjxxEnewstable，得到zjxxEnewstable对象的属性tbname和masterb。

②执行SQL语句“DELETE FROM zjxx\_enewstable WHERE tid = ‘tid’;”删除zjxx\_enewstable表中符合该tid的数据。

③执行SQL语句“DROP TABLE tbname; DROP TABLE tbname\_gd; DROP TABLE tbname\_wk;”删除数据库中的tbname表和其归档表、微客表。

④通过masterb找到该分表的主表，得到主表的分表集合字段datatbs，使用replace(tbname+”,”,””)修改datatbs，更新主表数据，返回result。

* + 1. 设置默认系统模型接口

**a.接口url:**

pub/admin/system/tableModel/setDefaultModel.do

**b.接口说明：**

设置当前系统模型为默认系统模型，其他系统模型为非默认。

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| setDefaultModel | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| mid | String | | 系统模型id | |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "增加成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

1. 接收参数mid，通过mid找到系统模型model。
2. 通过model获得tbname。
3. 设置属于该tbname的所有系统模型为非默认。（isdefault为0）。设置属于该tbname且mid=mid的系统模型为默认。（isdefault为1）
4. 返回result。
   * 1. 查询分表记录数接口

**a.接口url:**

system/tableModel/findCountByTbname.do

**b.接口说明:**

该接口用于查询分表记录数。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findCountByTbname | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tbname | String | 数据表名 |

**d.返回值示例与说明**

{

result:{flag:1,msg:查询成功},

total:10

}

**e.功能实现说明**

①检查参数tbname，若为空则返回信息“来源错误”

②非空则根据tbname查找对应表的记录数。

③返回info。

* + 1. 获取不同基础表字段接口（没有使用）

**a.接口url:**

system/tableModel/getZjxxBaseTableField.do

**b.接口说明:**

通过基础表的tbname获取该表的字段。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| getZjxxBaseTableField | ZjxxBaseTableField  （说明见6.1.5） | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| baseTbname | String[] | 基础表的表名数组 |

**d.返回值示例与说明**

{

“infos”：[ {

"columnName": "id", //字段名

"dataType": "varchar", //字段类型

"characterMaxImumLength": "48", //字段长度

"columnComment": "姓名", //字段备注

"isNullable": "no", //是否为空

"columnKey": "PRI" //是否为主键

},

{}

]

}

**e.功能实现说明**

①接收参数baseTbname数组，每次遍历该数组得到一个基础表名'tbname'，依次执行SQL语句

select column\_name,data\_type,character\_maxImum\_length,column\_comment,is\_nullable,column\_key

from information\_schema.COLUMNS

where TABLE\_SCHEMA='javaplat' and TABLE\_NAME='tbname'

②即可得到该表的所有字段信息，返回ZjxxBaseTableField类。

* + 1. 字段数据转移接口（没有使用）

**a.接口url:**

/pub/admin/system/tableModel/transferFieldData.do

**b.接口说明:**

字段修改所属表时数据转移用

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| transferFieldData | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| formTbname | String | 转出数据表名 |
| toTbname | String | 转入数据表名 |
| fname | String | 转移的字段名 |
| start | Int | 转移的起点下标 |
| size | Int | 转移多少条数据 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "200", //成功转移数量

"flag": "1" //标识

}

}

**e.功能实现说明**

1. 检查参数、formTbname、toTbname、fname，有任意一项为空值则返回信息 “来源错误”
2. 检查参数start、size、若为空则设置默认值start=0、size=200
3. 根据formTbname、fname编写查询语句、根据start、size控制limit
4. 遍历查询结果、依次执行update语句修改toTbname中的fname值。并记录转移成功数
5. 最后将成功转移成功数返回。

* 1. 工作流模块
     1. 增加工作流模型接口

**a.接口url:**

pub/admin/system/workflow/ addWorkflow.do

**b.接口说明：**

创建一个工作流模型

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| addWorkflow | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| wfname | String | | 工作流名称 | |
| wftext | String | | 工作流描述 | |
| sysModelId | String | | 系统模型Id | |
| myorder | String | | 顺序 | |
| orgId | String | | 机构Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "添加模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 产生一个Activiti的model对象。
2. 把wfname设置到对象name中，wfname,wftest, sysModelId,myorder，orgId以json字符串的形式存到对象中的MetaInfo变量中，执行repositoryService接口的saveModel方法，传入model对象参数，就把model对象存入数据库
3. 创建一个json对象editorNode，editorNode对象放id，resourceId和stencilset这3个变量，id的值为canvas，resourceId的值为canvas，stencilset是一个json对象，里面放namespace变量，值为http://b3mn.org/stencilset/bpmn2.0#。在执行repositoryService接口的addModelEditorSource方法，传入Model对象的id和editorNode对象的字符化两个参数。
4. 创建Resultd对象，设置msg为添加模型成功，flag为0，并返回这个Result对象。
   * 1. 修改工作流模型

**a.接口url:**

pub/admin/system/workflow/updateWorkflow.do

**b.接口说明：**

修改已经创建的工作流模型

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| updateWorkflow | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| wfname | String | | 工作流名称 | |
| wftext | String | | 工作流描述 | |
| sysModelId | String | | 系统模型Id | |
| myorder | String | | 顺序 | |
| modelId | String | | 工作流模型Id | |
| orgId | String | | 机构Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "修改模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 把参数wftext，sysModelId，myorder，orgId放到一个json对象中。在把这个json对象字符化赋值到MetaInfo变量中。
2. 把wfname，modelId，MetaInfo这四个变量放到一个map中。
3. 执行对应dao层的接口的修改方法，参入map参数。
4. 创建Resultd对象，设置msg为修改模型成功，flag为0，并返回这个Result对象。
   * 1. 修改工作流模型查找信息接口

**a.接口url:**

pub/admin/system/workflow/searchWorkflowinfo.do

**b.接口说明：**

修改工作流模型前查找该模型的数据

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchWorkflowinfo | | Map<String,Object> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelId | String | | 工作流模型Id | |

**d.返回值示例与说明**

{

" sysModelId ": "HI9385938 ", //配置工作流模型Id

"description": "aa", //工作流描述

"wfname": "aaaa", //工作流名称

"myorder": "34", //顺序

“orgId”：“” //机构Id

}

**e.功能实现说明**

1. 根据modelId查到这个model对象。
2. 把model中的name,description, sysModelId,myorder，orgId数据放到map中，其中description, sysModelId,myorder，orgId这三个数据存在model中的MetaInfo字段中。
3. 返回这个map。
   * 1. 查找工作流模型列表接口

**a.接口url:**

pub/admin/system/workflow/searchWorkflow.do

**b.接口说明：**

查询工作流模型列表，如果传modelName或orgId参数，用这个orgId 或modelName来模糊查找，如果不传，是对所有模型的搜索。

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchWorkflow | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelName | String | | 用模型名作为条件搜索 | |
| orgId | String | | 机构Id | |
| pageSize | Integer | | 每页显示的尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"sysModelId ": "837423", //系统模型Id

"modelid": "15001", //模型的id

"addtime": "Fri Aug 05 21:15:06 CST 2016",//模型增加的时间

"wfname": "aaaa", //工作流模型的名称

“orgId”：“” //机构Id

},

{}

]

}

**e.功能实现说明**

1. 根据参数拼接sql语句，把得到的list数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
2. 根据参数拼接sql语句，把得到的counts数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
3. 最后返回infor对象。
   * 1. 删除工作流模型

**a.接口url:**

pub/admin/system/workflow/deleteWorkflow.do

**b.接口说明：**

把已经存在的工作流模型删除掉

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| deleteWorkflow | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelId | String | | 模型Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "删除模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 执行repositoryService接口的deleteMode方法，传入modelId参数，就可以删除这个模型。
2. 创建Resultd对象，设置msg为删除模型成功，flag为0，并返回这个Result对象。
   * 1. 部署工作流模型接口

**a.接口url:**

pub/admin/system/workflow/deploy.do

**b.接口说明：**

部署工作流模型

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| deploy | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelId | String | | 模型Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "部署工作流模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 执行repositoryService接口的getMode方法，传入modelId参数，就可以得到这个模型。
2. 读这个模型的EditorSource文件放到ObjectNode中去，在把Object转化为BpmnModel，在从BpmnModel中得到byte[]数据。
3. 在把byte[]数据放到字符串bpmnString中去，从工作流模型得到模型名modelName，流程名processName设置为modelName加上‘.bpmn20.xml’的组合。
4. 执行repositoryService接口的.createDeployment() . name(modelName). addString(processName ,bpmnString) .deploy()方法.就可以部署好这个流程pi。
5. 从模型中得到配置的模型的id号sysModelId.
6. 执行processEngine.getIdentityService().newUser(pi.getd)方法,传入流程定义的Id号，创建了一个User对象，User对象的firstName设置模型的id号，User对象的lastName设置为sysModelId变量的值。
7. 创建Resultd对象，设置msg为部署流程成功，flag为0，并返回这个Result对象。
   * 1. 执行（完成）一个流程任务接口

**a.接口url:**

pub/admin/system/workFlow/completeTask.do

**b.接口说明：**

执行（完成）一个流程任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| completeTask | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskId | String | | 任务id | |
| jsonInfo | String | | 上传的Json数据 | |
| completeName | String | | 完成人 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "完成任务成功" //返回的信息；

}

}

**f.功能实现说明**

1. 根据任务id查到流程定义id，在查询activiti数据库的User表来得到系统模型id。
2. 执行updatData函数（该函数会返回信息id），传入jsonInfo，任务id,系统模型id三个参数根据任务id来得到任务往下走的逻辑变量list。
3. 遍历这个list，根据变量名来从jsonInfo中取数据，再把变量名当成key值，取得数据当成value值。放到一个map中。如果取不到数据，根据任务id调前面的数据来取值。
4. 让completeName人来拾取这个任务。
5. 把信息id绑定到这个任务上。
6. 在这个任务上设置一个qTaskId的参数，值为此任务的Id
7. 执行完成任务的方法，传入任务id和map。
8. 创建Resultd对象，设置msg为执行任务成功，flag为0，并返回这个Result对象。
   * 1. 启动（也可以叫申请或产生）一个新的流程接口

**a.接口url:**

pub/admin/system/workFlow/ startProcess.do

**b.接口说明：**

申请流程

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| startProcess | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processkey | String | | 流程的key值 | |
| aName | String | | 申请人 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "申请成功" //返回的信息；

}

}

**e.功能实现说明**

1. 把参数aName放到map中。
2. 执行runtimeService().startProcessInstanceByKey(processkey,map)方法。
3. 传入参数processkey和map。
4. 创建Resultd对象，设置msg为申请成功，flag为0，并返回这个Result对象。
   * 1. 查询某人已完成或未完成的任务

**a.接口url:**

pub/admin/system/workFlow/decisionMakerTask.do

**b.接口说明：**

查询一个人已经或未完成的任务（根据节点id来查任务，根据决策人姓名来查任务或根据决策人姓名和流程定义id来查任务）

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| decisionMakerTask | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| nodeId | String | | 节点id | |
| processDefinitionId | String | | 流程定义id | |
| decisionMaker | String | | 决策人姓名 | |
| complete | Integer | | 0表示未完成的任务，1表示已经完成的任务 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

“inforId”:”” //信息Id

},

{}

]

}

**f.功能实现说明**

1. 当complete参数为0的时候。
   1. 如果nodeId不为空且decisionMake不为空，执行TaskService接口的createTaskQuery. taskDefinitionKey(nodeId). taskCandidateOrAssigned (decisionMaker) . listPage(nowPage, pageSize)方法
   2. 如果nodeId不为空，执行TaskService接口的createTaskQuery. taskDefinitionKey(nodeId) .listPage(nowPage, pageSize)方法.（包括一个流程下节点下所有的任务，比如申请的话，所有申请人任务都在）
   3. 如果decisionMake不为空且processDefinitionId不为空，执行TaskService接口的createTaskQuery. processDefinitionId(processDefinitionId) . taskCandidateOrAssigned(decisionMaker).listPage(nowPage,pageSize)方法.（一个流程下，一个人的所有任务，可能有其他路径）
   4. 如果decisionMake不为空，执行TaskService接口的createTaskQuery. taskCandidateOrAssigned (tdecisionMaker) .listPage(nowPage, pageSize)方法.（一个人所有的任务，可能多个流程）
   5. 往方法中传入tdecisionMaker,nowPage和pageSize这三个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
   6. 查找每一个任务的上一个任务绑定的信息id变量，放到infor对象的list变量的一个对象中。
   7. 执行相应的查询数量方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
   8. 返回这个infor对象。
2. 当complete参数为1的时候。
   1. 如果nodeId不为空且decisionMake不为空,执 行historyService(). taskDefinitionKey(nodeId) . taskAssignee(decisionMaker). Finished ( ).listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   2. 如果nodeId不为空,执 行historyService(). taskDefinitionKey(nodeId) .finished().listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   3. 如果decisionMake不为空且processDefinitionId不为空执行historyService()..createHistoricTaskInstanceQuery().processDefinitionId(processDefinitionId).taskAssignee(decisionMaker).finished().listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   4. 如果decisionMake不为空,执 行historyService()..createHistoricTaskInstanceQuery().taskAssignee(decisionMaker).finished().listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   5. 往方法中传入tdecisionMaker,（nowPage\*pageSize-pageSize）和pageSize这三个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
   6. 执行historyService接口的相应的count()方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
   7. 返回这个infor对象。
      1. 查询流程实例信息

**a.接口url:**

pub/admin/system/workFlow/sezrch HistoricProcessInstance.do

**b.接口说明：**

查询单个流程的基本信息

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| sezrchHistoricProcess Instance | | Map<String,Object> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processInstanceId | String | | 流程实例Id | |

**d.返回值示例与说明**

{

"processDefinitionId": "leaveProcess:1:4", //流程定义Id

"businessKey": null, //流程的key值

"name": null, //流程的名称

"description": null, //描述

"startTime": "Wed Sep 14 11:02:51 CST 2016", //流程的开始时间

"endTime": "Wed Sep 14 11:49:32 CST 2016", //流程的结束时间

"id": "2501" //流程的id

}

**f.功能实现说明**

1. 执行historyService接口的createHistoricProcessInstanceQuery () . processInstanceId(businessKey). singleResult();
2. 得到HistoricProcessInstance的数据。
3. 定义一个map，把HistoricProcessInstance的一些参数放到这个map中
4. 返回map。
   * 1. 查询流程实例的过程（任务列表）

**a.接口url:**

pub/admin/system/workFlow/searchTaskHis.do

**b.接口说明：**

查询一个流程的执行了哪些任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchTaskHis | | List<Map<String, Object>> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processInstanceId | String | | 流程实例Id | |

**d.返回值示例与说明**

[

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

"endTime": "", //结束时间

}

]

**f.功能实现说明**

1. 执行historyService接口的createHistoricTaskInstanceQuery() . processInstanceId(processInstanceId). list();
2. 得到List<HistoricTaskInstance>的数据。
3. 再把List<HistoricTaskInstance>返回到页面
   * 1. 查询流程定义列表

**a.接口url:**

pub/admin/system/workFlow/sezrch ProcessDefinitions.do

**b.接口说明：**

查询已经部署的流程有哪些

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| sezrch ProcessDefinitions | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"DeploymentId": "1", //部署Id

"id": "leaveProcess:1:4", //流程定义Id

"Key": "leaveProcess", //流程定义的Key

"Name": "leaveMyProcess" //流程名

} ,

{}

]

}

**f.功能实现说明**

1. 执行repositoryService接口的createProcessDefinitionQuery () .listPage(nowPage, pageSize);
2. 往方法中传入nowPage和pageSize这两个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
3. 执行repositoryService接口的createHistoricProcessInstanceQuery().count()方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
4. 返回这个infor对象。
   * 1. 查询流程实例列表

**a.接口url:**

pub/admin/system/workFlow/sezrch HistoricProcessInstances.do

**b.接口说明：**

求一个流程产生了多少条流程实例

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| sezrchHistoricProcess Instances | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processDefinitionId | String | | 流程定义id | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"processDefinitionId": "leaveProcess:1:4", //流程定义Id

"name": null, //流程的名称

"description": null, //描述

"startTime": "Wed Sep 14 11:02:51 CST 2016", //流程的开始时间

"endTime": "Wed Sep 14 11:49:32 CST 2016", //流程的结束时间

"id": "2501" //流程的id

},

{}

]

}

**f.功能实现说明**

1. 执行HistoryService(). createHistoricProcessInstanceQuery(). processDefinitionId(processDefinitionId) .listPage(nowPage, pageSize)。
2. 往方法中传入processDefinitionId,nowPage和pageSize这三个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
3. 执行HistoryService接口的createHistoricProcessInstanceQuery(). processDefinitionId(processDefinitionId. count()方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
4. 返回这个infor对象。
   * 1. 查找节点候选人名单接口

**a.接口url:**

pub/admin/system/workFlow/findCandidater.do

**b.接口说明：**

查询此节点候选人名单

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| findCandidater | | List<Map<String,String>> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskId | String | | 任务id | |

**d.返回值示例与说明**

[

{

"personName": "" //候选人姓名

},

{

"personName": "" //候选人姓名

}

]

**e.功能实现说明**

1. 执行taskService接口的getIdentityLinksForTask(taskId)方法。
2. 传入参数taskId，会得到IdentityLink对象的list，遍历list，把IdentityLink对象的UserId字段值赋值给personName变量，把personName变量放到一个map中，这个map中的数据放到list中
3. 返回list。
   * 1. 拾取任务接口

**a.接口url:**

pub/admin/system/workFlow/claimTask.do

**b.接口说明：**

拾取任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| claimTask | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskId | String | | 任务id | |
| userId | String | | 拾取人名称 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "拾取成功" //返回的信息；

}

}

**e.功能实现说明**

1. TaskService接口执行claim方法，传入taskId和userId两个变量。
2. 创建Resultd对象，设置msg为拾取成功，flag为0，并返回这个Result对象。
   1. 附件管理
      1. 增加/修改文件夹

**a.接口url:**

pub/admin/system/enclosure/addOrUpdatefloder.do

**b.接口说明：**

增加或修改文件夹。

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| addOrUpdatefloder | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| classid | String | | 栏目id | |
| id | String | | “”  注意：增加时，id传空字符串，修改时，id传该条信息的id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "操作成功" //返回的信息；

}

}

**e.功能实现说明**

* + 1. 删除单个/批量删除附件接口

**a.接口url:**

pub/admin/system/enclosure/deleteAttachment.do

**b.接口说明:**

删除、批量删除数据表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteAttachment | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| ids | String[] | 附件id数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①传入参数tids，遍历tids得到每个tid，使用finById(tid)找到该表数据，得到该表的父表数据，得到该表的表名tbname，修改其父表的子表集合。

②使用deleteByRecursion(tbname)方法递归删除该表下的系统模型、删除字段表中属于该表的字段、删除该表数据、删除数据库中的该表、删除该表的归档和微客表。

③得到该表的分表集合，遍历分表集合删除所有分表数据、删除分表字段、删除数据库中分表、删除分表的归档表和微客表。

④得到该表的子表集合，如果子表集合不为空，遍历子表集合，每次将一个子表的表名sontbname作为递归参数再次传入deleteByRecursion(sontbname)进行下一轮递归。

* + 1. 清除失效文件接口

**a.接口url:**

pub/admin/system/enclosure/deleteLoseFile.do

**b.接口说明:**

删除、批量删除数据表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteLoseFile | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| ids | String[] | 失效文件id数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①传入参数tids，遍历tids得到每个tid，使用finById(tid)找到该表数据，得到该表的父表数据，得到该表的表名tbname，修改其父表的子表集合。

②使用deleteByRecursion(tbname)方法递归删除该表下的系统模型、删除字段表中属于该表的字段、删除该表数据、删除数据库中的该表、删除该表的归档和微客表。

③得到该表的分表集合，遍历分表集合删除所有分表数据、删除分表字段、删除数据库中分表、删除分表的归档表和微客表。

④得到该表的子表集合，如果子表集合不为空，遍历子表集合，每次将一个子表的表名sontbname作为递归参数再次传入deleteByRecursion(sontbname)进行下一轮递归。

* + 1. 查找机构根文件夹列表

**a.接口url:**

pub/admin/system/enclosure/findOrgRootFileList.do

**b.接口说明:**

该接口用于查找机构的根文件夹列表。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findOrgRootFileList | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| orgid | String | 机构id |
| classid | String | 栏目id |
| nowPage | Integer | 当前页 |
| pageSize | Integer | 每页显示的尺寸 |

**d.返回值示例与说明**

{

"infos": [ //信息

{…}

]

}

**e.功能实现说明**

* + 1. 查找子文件夹

**a.接口url:**

pub/admin/system/enclosure/findChildrenFolder.do

**b.接口说明:**

该接口用于查找子文件夹。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findChildrenFolder | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
|  | String | 父文件夹id |
|  | String | 文件夹id |

**d.返回值示例与说明**

{

"infos": [ //信息

{…}

]

}

**e.功能实现说明**

* + 1. 根据文件夹id查找文件夹下所有附件

**a.接口url:**

pub/admin/system/enclosure/findAllAttachmentById.do

**b.接口说明:**

该接口用于查找子文件夹。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findAllAttachmentById | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
|  | String | 父文件夹id |

**d.返回值示例与说明**

{

"infos": [ //信息

{…}

]

}

**e.功能实现说明**

* + 1. 上传单个文件（可以不要，与1.4.8重复）

**a.接口url:**

pub/admin/system/enclosure/uploadFileOne.do

**b.接口说明**

上传单个附件，返回文件的信息

检查上传文件类型是否符合系统设置中的文件设置

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| uploadFileOne | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| file | MultipartFile | | 上传的文件变量 | |
| filetype | String | | 文件类型，文件类型用数据词典做的 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [ //信息;

{

" fileid ": "547a7578defe483b9d7629a1881cdb5b", //文件id;

" filename ": "1.jpg", //文件名;

" adduser ": "张三", //增加者;

" filesize ": "14000", //文件的大小;

" filetime ": "2016-08-13 16:16:08", //文件的增加时间;

“filetype”:””, //文件类型

“path”:””, //文件的路径

“classid”:””, //栏目id

“no”:””, //文件别名

“onclick”:””, //下载次数

“id”:””, //信息id

“cjid”：“”， //文件临时识别编码

“md5”：“”， //文件MD5值

“fid”：“”， //文件的父id

“ftype”：“”， //文件类型

}

]

}

**e.功能实现说明**

1. 把上传的判断是否存在，
   1. 存在就只存文件的信息到数据库
   2. 不存在，存文件信息到数据库中，在把文件存到服务器上
      1. 上传附件和属性接口

注意：1.按照系统参数设置中文件设置来存储设置图片。

2.上传图片可以设置图片宽高度，可以生成缩略图，添加水印。

**a.接口url:**

pub/admin/system/enclosure/uploadFile.do

**b.接口说明**

1. 上传单个附件或多个附件
2. 对上传图片可以生成缩略图、添加水印

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| uploadFile | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| getsmall | String | | 生成缩略图标志（为1生成缩略图，其他不生成） | |
| no | String[] | | 给多个文件设置别名，不写的文件别名设置成原始文件名 | |
| width | Integer | | 生成缩略图图片的宽度（默认值200） | |
| height | Integer | | 生成缩略图图片的高度（默认值200） | |
| file | MultipartFile[] | | 上传的文件变量 | |
| classid | string | | 所属栏目id、可为空 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "上传附件成功" //返回的信息；

}

}

**e.功能实现说明**

1. 对file进行遍历，假如其中有的文件是空的，结束本次循环，进行下一次循环。遍历结束，跳到第8步。
2. 取得服务器上工程的根路径。
3. 首先计算文件MD5值，搜索数据库这个MD5值是否存在，如果不存在，继续执行下一步。若果存在，跳过第4步，从第5步执行。
4. 把附件的文件名改为MD5值，在把这个附件存储到根路径下pub/admin/upload中。（备注：存储附件的时候，先建一个日期文件夹，再来存储附件，日期文件夹用年月日来命名：比如20160817，如果日期文件夹存在，就不需要再建，附件就直接放到这个日期文件夹下）。
5. 在new一个zjxxEnewsfile对象，给对象字段设置值（**注意新增了classid字段**），设置好了把对象对象存储到数据库中。（如果这个文件的MD5值已经存在，zjxxEnewsfile对象的path字段从MD5相同的一条数据中得到）。
6. 假如getsmall值是1，根据width 和 height把图片文件生成缩略图。在执行3,4,5步，用来保存缩略图数据和附件，在执行第5步的时候，必须要设置fid和ftype这两个字段，fid表示这张缩略图元文件的id，ftype设置为0字符串。（fid和ftype用来设置缩略图与原图的逻辑）结束后执行第7步。
7. 跳到第1步。直到遍历结束。
8. 创建Resultd对象。
9. 如果上传成功，设置msg为上传附件成功，flag为0，否则设置msg为上传附件失败，flag为1.
10. 返回这个Result对象。
    * 1. 查找附件列表接口

**a.接口url:**

pub/admin/system/enclosure/searchFile.do

**b.接口说明：（**nowPage和pageSize两个参数必须传**）**

1. 搜索附件列表接口
2. 条件搜索附件
   * 1. 根据文件夹名来搜索附件，在加上folderName参数
     2. 根据文件名，编号，上传者条件来搜索附件，在加上filename或no或adduser参数。

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchFile | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| nowPage | Integer | | 当前页 | |
| pageSize | Integer | | 每页显示的尺寸 | |
| folderName | String | | 文件夹名 | |
| filename | String | | 文件名 | |
| no | String | | 编号 | |
| adduser | String | | 上传者 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [ //信息;

{

" fileid ": "547a7578defe483b9d7629a1881cdb5b", //文件id;

" filename ": "1.jpg", //文件名;

" adduser ": "张三", //增加者;

" filesize ": "14000", //文件的大小;

" filetime ": "2016-08-13 16:16:08", //文件的增加时间;

“path”:””, //文件的路径

“classid”:””, //栏目id

“no”:””, //文件别名

“onclick”:””, //下载次数

“id”:””, //信息id

“cjid”：“”， //文件临时识别编码

“md5”：“”， //文件MD5值

“fid”：“”， //文件的父id

“ftype”：“”， //文件类型

},

{}

]

}

**e.功能实现说明**

1. 把（nowPage\*pageSize-nowPage），pageSize，folderName, no, adduser,filename这六个参数放入map中。
2. 执行dao层的查询数据方法，把得到的数据放到infor对象的list中。把得到数据的长度放到infor对象的total字段中。
3. 执行dao层的查询数据长度方法，传入map，把得到的数据放到infor对象的page对象的lines中。在给page对象设置page,pages,pageSize,pageStart字段。
4. 返回这个infor对象
   * 1. 附件下载（没有使用）

**a.接口url:**

pub/admin/system/enclosure/Downloadfile.do

**b.接口说明：**

下载附件到本地

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| Downloadfile | | ResponseEntity<byte[]> | | 二进制流 |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| fileid | String | | 文件的id | |

**d.返回值示例与说明**

返回文件

**e.功能实现说明**

1. 根据文件的id，执行dao层方法，传入id。得到zjxxEnewsfile对象
2. 从zjxxEnewsfile中的到onclick字段值，加上1，在设置到zjxxEnewsfile对象的onclick字段中。
3. 执行dao层的修改方法，传入zjxxEnewsfile参数。
4. 从zjxxEnewsfile对象中的得到文件的路径path。
5. 取得服务器上的工程路径和pub/admin/uploa和path组成一个文件的路径，从而得到这个文件
6. ResponseEntity<byte[]>方法把文件以二进制流的方式写出去。
   * 1. 附件查看（没有使用）

**a.接口url:**

pub/admin/system/enclosure/lookfilejson.do

**b.接口说明：**

返回文件的具体信息

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| lookfilejson | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| fileid | String | | 文件的id | |

**d.返回值示例与说明**

{

“infos”:

{

" fileid ": "547a7578defe483b9d7629a1881cdb5b", //文件id;

" filename ": "1.jpg", //文件名;

" adduser ": "张三", //增加者;

" filesize ": "14000", //文件的大小;

" filetime ": "2016-08-13 16:16:08", //文件的增加时间;

“path”:””, //文件的路径

“classid”:””, //栏目id

“no”:””, //文件别名

“onclick”:””, //下载次数

“id”:””, //信息id

“cjid”：“”， //文件临时识别编码

“md5”：“”， //文件MD5值

“fid”：“”， //文件的父id

“ftype”：“”， //文件类型

}

}

**e.功能实现说明**

1. 根据文件的id，执行dao层方法，传入id。得到zjxxEnewsfile对象
2. 返回zjxxEnewsfile对象
   * 1. 查询根级目录接口

**a.接口url:**

pub/admin/system/enclosure/searchCatalog.do

**b.接口说明：**

搜索目录

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchCatalog | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [ //信息;

{

" folderName": "201415", //文件夹名;

}

]

}

**e.功能实现说明**

1. 先取得服务器上工程的根路径，把根路径和pub/admin/upload组成一个新路径path2.
2. 取得path2路径下的文件数组m，遍历这个文件数组，假如此文件是一个文件夹，把这个文件放到文件数组a中
3. 把a数组的长度放到infor对象的page对象中的lines字段中。
4. 取a数组中（nowPage\*pageSize-nowPage）到（nowPage\*pageSize）的数据放到list<map>中，每一个map的key值为folderName。
5. 在把list<map>设置到infor对象的list中，把list<map>的长度设置到infor对象的total字段中。
6. 给page设置其他字段。
7. 返回这个infor对象.
   * 1. 裁剪图片接口 （没有使用）

**a.接口url:**

pub/admin/system/enclosure/cutImage.do

**b.接口说明：**

裁剪图片

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| cutImage | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| fileid | String | | 文件id | |
| width | Integer | | 剪切文件的宽度 | |
| height | Integer | | 剪切文件的高度 | |
| x | Integer | | 剪切图片起始点x坐标 | |
| y | Integer | | 剪切图片起始点y坐标 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "剪切图片成功" //返回的信息；

}

}

**e.功能实现说明**

1. 根据文件id的来得到文件的存储路径，从而得到这个文件。
2. 根据width，height,x和y对这张图片进行剪切，得到剪切后的文件file2.
3. 计算文件file2的MD5值，搜索数据库这个MD5值是否存在，如果不存在，继续执行下一步。若果存在，跳过第4步，从第5步执行。
4. 把file2附件存储到根路径下pub/admin/upload文件夹中。（备注：存储附件的时候，先建一个日期文件夹，再来存储附件，日期文件夹用年月日来命名：比如20160817，如果日期文件夹存在，就不需要再建，附件就直接放到这个日期文件夹下）
5. 在new一个zjxxEnewsfile对象，给对象字段设置值，必须设置fid和ftype这两个字段，fid的值为fileid，ftyppe的值为1。（fid，ftype用来设置剪切图与原图的逻辑）设置好了把对象对象存储到数据库中。（如果这个文件的MD5值已经存在，zjxxEnewsfile对象的path字段从MD5相同的一条数据中得到）
6. 创建Resultd对象。
7. 如果剪切成功，设置msg为剪切成功，flag为0，否则设置msg为剪切失败，flag为1.
8. 返回这个Result对象。
   * 1. 删除文件夹

**a.接口url:**

pub/admin/system/enclosure/deleteFolder.do

**b.接口说明:**

删除文件夹。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteFolder | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| id | String | 文件夹id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

* + 1. 查询文件夹里面的文件及文件夹

**a.接口url:**

pub/admin/system/enclosure/findFileInFolder.do

**b.接口说明:**

该接口用于查找文件夹里面的文件及文件夹。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findFileInFolder | Info | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
|  | String | 文件夹id |

**d.返回值示例与说明**

{

"infos": [ //信息

{…}

]

}

**e.功能实现说明**

* 1. 信息查询
     1. 查找SQL模板列表接口

**a.接口url:**

system/setSystem/findSQLModalList.do

**b.接口说明:**

查找SQL模板列表。

**c. 接口定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口定义 | | | | | |
| 接口名 | | 对象名 | | | 返回值类型 |
| findSQLModalList | | Info | | | Json |
| 参数说明： | | | |  | |
| 参数名称 | 参数类型 | | 备注 | | |
| classid | String | | 栏目id | | |
| page | String | | 当前页码 | | |
| num | String | | 每页显示多少条数据 | | |

**d.返回值示例与说明**{

"page": { //分页信息

"page": 2, //当前页

"pages": 10, //总页数

"lines": 100, //信息总条数

"pageSize": "10", //每页显示信息数

"pageStart": 1 //起始页

},

"total": 1, //当前页信息数

"infos": [ //信息

{

····

}

]

}

**e.功能实现说明**

①检查page参数、为空则设置默认值1，检查num参数、为空则设置默认值30。

②根据page和num生成页码条件（limit）

③组合sql语句查询数据，根据返回值格式进行数据返回。

* + 1. 新增/修改SQL模板接口

**a.接口url:**

system/setSystem/addOrUpdateSQLModal.do

**b.接口说明:**

增加扩展变量。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| addOrUpdateSQLModal | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
|  | string | SQL模板中文名称 |
|  | string | SQL模板英文名 |
|  | string | 增加查询条件 |
|  | string | SQL查询语句 |
|  | string | sql功能描述 |
|  | string | 刷新周期 |
|  | string | 显示排序 |
| id | string | “”  注意：增加传空，修改传该信息id |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "增加成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

①添加系统变量的时候该字段传空字符串，修改系统变量的时候该字段传信息id

②返回更新成功信息

* + 1. 查找SQL模板详情接口

**a.接口url:**

system/setSystem/findSQLModalInfo.do

**b.接口说明:**

查找SQL模板列表。

**c. 接口定义：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口定义 | | | | | |
| 接口名 | | 对象名 | | | 返回值类型 |
| findSQLModalInfo | | Info | | | Json |
| 参数说明： | | | |  | |
| 参数名称 | 参数类型 | | 备注 | | |
| classid | String | | 栏目id | | |
| id | String | | SQL模板id | | |

**d.返回值示例与说明**{

"page": { //分页信息

"page": 2, //当前页

"pages": 10, //总页数

"lines": 100, //信息总条数

"pageSize": "10", //每页显示信息数

"pageStart": 1 //起始页

},

"total": 1, //当前页信息数

"infos": [ //信息

{

····

}

]

}

**e.功能实现说明**

* + 1. 删除/批量删除SQL模板接口

**a.接口url:**

system/setSystem/deleteSQLModal.do

**b.接口说明:**

删除/批量删除SQL模板。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| deleteSQLModal | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| ids | String[] | SQL模板id数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "删除成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

* + 1. 批量修改SQL模板顺序

**a.接口url:**

system/setSystem/updateSQLModalOrder.do

**b.接口说明:**

修改、批量修改字段顺序。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| updateSQLModalOrder | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| id | String[] | SQL模板id的数组 |
| order | String[] | 字段显示顺序的数组 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "修改成功", //信息

"flag": "1" //标识

}

}

**e.功能实现说明**

* 1. 导入数据
     1. 批量导入excel接口
     2. 批量导入word接口
     3. 批量导出excel接口
     4. 批量导出word接口
  2. 函数
  3. 对象

1. 暂不实现接口
   1. 系统设置
      1. 查找项目基本信息接口（不需要）
      2. 修改系统参数-基本设置接口
      3. 修改系统参数-用户设置接口
      4. 修改系统参数-文件设置接口
      5. 修改系统参数-信息设置接口
      6. 修改系统参数-消息推送设置接口
      7. 修改系统参数-支付设置接口
      8. 修改安全参数设置接口
      9. 修改网络防火墙接口
   2. 数据表与系统模型
      1. 查找父表的ID和表名接口

**a.接口url:**

system/tableModel/findparenttidtbname.do

**b.接口说明:**

（1）如果传入参数zjxxEnewsf对象的fid为空，则为增加字段接口。

（2）如果zjxxEnewsf对象的fid不为空，则为修改字段、转移字段接口。

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| findparenttidtbname | tidtbname | String |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| tbname | String | 数据表表名 |

**d.返回值示例与说明**

tidtbname

**e.功能实现说明**

判断

1. 判断tbname为空或为空字符串时，返回tidtbname值为error1
2. 根据tbname在模型数据表中查找该表的记录，去该表的parenttb，
3. 判断parenttb为空或为空字符串时，返回tidtbname值为error2，否者在根据parenttb在模型数据表中查找该父表的记录，通过父表的记录分别取到父表的tid和tbname，将其用“|”拼接，赋值给tidtbname，
4. 返回tidtbname；
   * 1. 内容页html生成接口

**a.接口url:**

pub/admin/information/Infohtml/createContentHtml.do

**b.接口说明:**

用于生成内容页html

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| createContentHtml | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| id | String | 信息id |
| classid | String | 栏目id |
| filename | String | 文件名 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "操作成功", //提示信息

"flag": "1" //标识

}

}

**e.功能实现说明**

1. 检查参数、id、classid、filename，皆为空值则返回信息 “来源错误”，filename为空且id、classid中有任意一个空值同样返回信息“来源错误”
2. 若id、classid任意一值为空，分解filename、取出第一位字符找到对应缓存信息表、再根据缓存信息表和filename找到classid和id。
3. 如果classid和id不为空，将classid和infoid传入[生成文件名方法](#_5.1.1__生成文件名方法)，生成文件名。
4. 把classid、id、filename、缓存信息表分表cacheInfoTbname传入**[生成页面信息方法](#_5.1.2__生成页面信息方法)**
5. 返回result
   * 1. 列表页html生成接口

**a.接口url:**

pub/admin/information/Infohtml/createListHtml.do

**b.接口说明:**

用于生成列表页html

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| createListHtml | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| filename | String | 文件名 |
| page | int | 指定页码 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "操作完成", //信息提示

"flag": "1" //标识

}

}

**e.功能实现说明**

1. 检查参数、classid、filename，都为空值则返回信息 “来源错误”
2. classid为空时、分解filename、取出第一位字符找到对应缓存信息表，再根据缓存信息表和filename找到classid。
3. 如果classid不为空，将classid传入[生成文件名方法](#_5.1.1__生成文件名方法)，生成文件名。
4. 把classid、page（为空默认查第一页）、filename、缓存信息表分表cacheInfoTbname传入**[生成列表信息方法](#_5.1.3__生成列表信息方法)**
5. 返回result
   * 1. 条件列表页html生成接口

**a.接口url:**

pub/admin/information/Infohtml/createConditionListHtml.do

**b.接口说明:**

用于生成列表页html

**c. 接口定义：**

|  |  |  |
| --- | --- | --- |
| 接口定义 | | |
| 接口名 | 对象名 | 返回值类型 |
| createConditionListHtml | Result | Json |
| 参数说明： | | |
| 参数名称 | 参数类型 | 备注 |
| classid | String | 栏目id |
| filename | String | 文件名 |

**d.返回值示例与说明**

{

"result": { //结果

"msg": "操作完成", //信息提示

"flag": "1" //标识

}

}

**e.功能实现说明**

1. 检查参数、classid、filename，都为空值则返回信息 “来源错误”
2. classid为空时、分解filename、取出第一位字符找到对应缓存信息表、再根据第二位开始的字符串找到对应信息的classid
3. 根据classid查询栏目信息、生成栏目对象

\*\*\*\*\*\*

* 1. 工作流模块
  2. 附件管理
  3. 规则管理
     1. 查找规则列表接口
     2. 新增/修改规则接口
     3. 查找规则详情接口
     4. 删除规则接口
  4. 信息查询
  5. 数据导入