jhnw程序开发文档

总体设计文档总体结构，功能模块设计。

数据库设计说明书数据库设计文档 、 ER图，对数据库中使用的所有标识、逻辑结构和物理结构做出具体的设计规定。

数据迁移方案有时涉及到数据迁移的话要有方案的文档，主要包括迁移内容、迁移方案、异常处理这三部分。

接口说明文档包括系统间的接口设计以及内部功能模块之间的接口设计，比较大的项目是需要这部分的详细文档的。

参考如下：引言：设计目标、使用范围、相关术语、参考资料；分析设计：设计思想、接口标准结构体、数据流分析；接口：可根据功能模块逐条列出，每条接口要具备函数原型、功能、声明头文件、参数说明、返回值说明、使用方法、提示等。

# 数据库设计文档

## 获取基本信息命令表 basic\_information

表名：basic\_information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 | 常见值 | 意义 |
| id | int | 11 | 主键 |  |  |
| command | varchar | 255 | 命令 |  |  |
| problem\_id | varchar | 20 | 分析ID |  |  |

主键自增，

命令为获取基本信息命令，以“，”分隔。

分析ID 为 问题扫描逻辑表主键。

## 问题扫描逻辑表 problem\_scan\_logic

表名：problem\_scan\_logic

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 | 常见值 | 意义 |
| id | varchar | 20 | 主键索引 |  |  |
| matched | varchar | 12 | 匹配 | 精确匹配、模糊匹配、不存在 |  |
| relative\_position | varchar | 10 | 相对位置 | （0,0）、 null | （行，列）、全文 |
| match\_content | varchar | 255 | 匹配内容 |  |  |
| action | varchar | 20 | 动作 | 取词 |  |
| r\_position | int | 2 | 位置 |  |  |
| length | varchar | 3 | 长度 |  |  |
| exhibit | varchar | 4 | 是否显示 | 是、否 | 显示、不显示 |
| word\_name | varchar | 100 | 取词名称 | 设备品牌、设备型号、内部固件版本、子版本号 |  |
| compare | varchar | 50 | 比较 |  |  |
| t\_line | varchar | 2 | true行号 |  | 成功分支行号 |
| t\_next\_id | varchar | 20 | true下一条分析索引 |  | 成功下一条分析ID |
| t\_com\_id | varchar | 20 | true下一条命令索引 |  | 成功下一条命令ID |
| problem\_id | varchar | 50 | 问题索引 | 无问题107、有问题105 | 安全、异常加交换机问题ID |
| f\_line | varchar | 2 | false行号 |  | 失败分支行号 |
| f\_next\_id | varchar | 20 | false下一条分析索引 |  | 失败下一条分析ID |
| f\_com\_id | varchar | 20 | false下一条命令索引 |  | 失败下一条命令ID |
| return\_cmd\_id | int | 11 | 返回命令 |  |  |
| cycle\_start\_id | varchar | 20 | 循环起始ID |  | 循环到某一分析ID（会有命令可能吗） |

主键ID 由前端根据时间邮戳生成。 建立索引。

## 命令逻辑表command\_logic

表名：command\_logic

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 | 常见值 | 意义 |
| id | varchar | 20 | 主键索引 |  |  |
| state | varchar | 2 | 状态 |  |  |
| c\_line | varchar | 2 | 命令行号 | 1 | 分析逻辑中位于第几行 |
| command | varchar | 100 | 命令 |  |  |
| result\_check\_id | varchar | 1 | 结果检查id | 0、1 | 0为 分析problem\_id，1为下一条命令end\_index |
| problem\_id | varchar | 20 | 分析id |  | 问题扫描逻辑表 ID |
| end\_index | varchar | 20 | 下一结束索引 |  | 命令表 下一条命令ID |

## 问题描述表problem\_describe

表名：problem\_describe

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| id | int | 11 | 问题描述ID |
| problem\_describe | text | 0 | 问题描述 |

## 交换机返回信息表return\_record

表名：return\_record

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| id | int | 11 | 主键 |
| user\_name | varchar | 100 | 系统登录人名称 |
| switch\_ip | varchar | 100 | 交换机IP |
| brand | varchar | 100 | 品牌 |
| type | varchar | 100 | 型号 |
| fireware\_version | varchar | 100 | 内部固件版本 |
| sub\_version | varchar | 100 | 子版本号 |
| current\_comm\_log | varchar | 255 | 当前通信日志 |
| current\_return\_log | text | 0 | 当前返回日志 |
| current\_identifier | text | 0 | 当前标识符 |
| create\_time | datetime | 0 | 创建时间 |

## 交换机错误表switch\_error

表名：switch\_error

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| error\_id | int | 11 | 错误主键 |
| brand | varchar | 50 | 交换机品牌 |
| switch\_type | varchar | 50 | 型号 |
| fireware\_version | varchar | 50 | 内部固件版本 |
| sub\_version | varchar | 50 | 子版本号 |
| error\_keyword | varchar | 255 | 错误关键字 |
| error\_name | varchar | 255 | 错误名称 |

## 交换机故障表switch\_failure

表名：switch\_failure

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| failure\_id | int | 11 | 故障主键 |
| brand | varchar | 50 | 交换机品牌 |
| switch\_type | varchar | 50 | 型号 |
| fireware\_version | varchar | 50 | 内部固件版本 |
| sub\_version | varchar | 50 | 子版本号 |
| failure\_keyword | varchar | 255 | 故障关键字 |
| failure\_name | varchar | 255 | 故障名称 |

## 操作日志记录switch\_oper\_log

表名：switch\_oper\_log

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| oper\_id | bigint | 20 | 日志主键 |
| title | varchar | 50 | 模块标题 |
| business\_type | int | 2 | 业务类型（0其它 1新增 2修改 3删除） |
| method | varchar | 500 | 方法名称 |
| request\_method | varchar | 10 | 请求方式 |
| operator\_type | int | 1 | 操作类别（0其它 1后台用户 2手机端用户） |
| oper\_name | varchar | 50 | 操作人员 |
| dept\_name | varchar | 50 | 部门名称 |
| oper\_url | varchar | 255 | 请求URL |
| oper\_ip | varchar | 128 | 主机地址 |
| oper\_location | varchar | 255 | 操作地点 |
| oper\_param | varchar | 2000 | 请求参数 |
| json\_result | varchar | 2000 | 返回参数 |
| status | int | 1 | 操作状态（0正常 1异常） |
| error\_msg | varchar | 2000 | 错误消息 |
| oper\_time | datetime | 0 | 操作时间 |

## 交换机问题表switch\_problem（废弃）

表名：switch\_problem

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| id | int | 11 | 主键 |
| switch\_ip | varchar | 255 | 交换机ip |
| brand | varchar | 50 | 品牌 |
| switch\_type | varchar | 50 | 型号 |
| fireware\_version | varchar | 50 | 内部固件版本 |
| sub\_version | varchar | 50 | 子版本号 |
| switch\_name | varchar | 255 | 交换机姓名 |
| switch\_password | varchar | 255 | 交换机密码 |
| login\_method | varchar | 10 | 登录方式 |
| port\_number | int | 5 | 登录端口号 |
| problem\_id | varchar | 11 | 问题索引 |
| if\_question | varchar | 20 | 是否有问题 |
| com\_id | varchar | 20 | 命令索引 |
| value\_id | int | 11 | 参数索引 |
| resolved | varchar | 4 | 是否解决 |
| create\_time | datetime | 0 | 扫描时间 |
| user\_name | varchar | 20 | 登录名称 |
| phonenumber | varchar | 100 | 登录手机号 |

## 交换机扫描结果switch\_scan\_result

表名：switch\_scan\_result

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| id | int | 11 | 主键 |
| switch\_ip | varchar | 255 | 交换机ip |
| brand | varchar | 50 | 品牌 |
| switch\_type | varchar | 50 | 型号 |
| fireware\_version | varchar | 50 | 内部固件版本 |
| sub\_version | varchar | 50 | 子版本号 |
| switch\_name | varchar | 255 | 交换机姓名 |
| switch\_password | varchar | 255 | 交换机密码 |
| configureCiphers | varchar | 255 | 配置密码 |
| login\_method | varchar | 10 | 登录方式 |
| port\_number | int | 5 | 登录端口号 |
| problem\_id | varchar | 11 | 问题索引 |
| type\_problem | varchar | 50 | 问题种类 |
| tem\_pro\_name | varchar | 50 | 范本问题名称 |
| problem\_name | varchar | 50 | 问题名称 |
| remarks | varchar | 255 | 备注 |
| problem\_describe\_id | int | 11 | 问题详细说明和指导索引 |
| if\_question | varchar | 20 | 是否有问题 |
| com\_id | varchar | 20 | 命令索引 |
| dynamic\_information | varchar | 255 | 动态信息 |
| create\_time | datetime | 0 | 扫描时间 |
| user\_name | varchar | 20 | 登录名称 |
| phonenumber | varchar | 100 | 登录手机号 |

## 交换机定义问题表total\_question\_table

表名：total\_question\_table

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| id | int | 11 | 主键索引 |
| brand | varchar | 50 | 品牌 |
| type | varchar | 50 | 型号 |
| fireware\_version | varchar | 50 | 内部固件版本 |
| sub\_version | varchar | 50 | 子版本号 |
| not\_finished | varchar | 255 | 未完成(包含某个字符串说明返回未完成) |
| required\_items | int | 1 | 是否必选(0 非必选，1必选) |
| command\_id | varchar | 20 | 启动命令ID |
| problem\_solving\_id | varchar | 20 | 解决问题命令ID |
| type\_problem | varchar | 50 | 问题种类 |
| tem\_pro\_name | varchar | 50 | 范本问题名称 |
| problem\_name | varchar | 50 | 问题名称 |
| remarks | varchar | 255 | 备注 |
| problem\_describe\_id | int | 11 | 问题详细说明和指导索引 |

## 取值信息存储表value\_information（废弃）

表名：value\_information

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 注释 |
| id | int | 11 | 主键 |
| exhibit | varchar | 4 | 是否显示 |
| dynamic\_Vname | varchar | 100 | 动态信息名称 |
| dynamic\_information | varchar | 255 | 动态信息 |
| out\_id | int | 11 | 下一信息ID |
| display\_information | varchar | 100 | 动态信息(显示或密文) |

# 接口说明文档

## 获取交换机基本信息定义

### 获取交换机基本信息定义

方法名：insertInformationAnalysis

|  |  |
| --- | --- |
| 定义获取基本信息命令插入 | |
| 方法名 | insertInformationAnalysis |
| 路径 | /sql/DefinitionProblemController/insertInformationAnalysis |
| 参数 | @RequestBody List<String> jsonPojoList 分析逻辑集合 @RequestParam String command 获取交换机基本信息命令以“，”分割 |
| 逻辑 | 先将命令插入数据库，并返回插入命令数据的ID。 将分析逻辑集合及获取交换机基本信息命令ID，传入 引用方法：insertInformationAnalysisMethod 中。 |
| 引用方法 | insertInformationAnalysisMethod |
| 返回结果 | boolean true：插入成功 false：插入失败 |

### 2.获取交换机基本信息分析数据的插入

方法名：insertInformationAnalysisMethod

|  |  |
| --- | --- |
| 定义获取基本信息分析数据插入 | |
| 方法名 | insertInformationAnalysisMethod |
| 路径 | /sql/DefinitionProblemController/insertInformationAnalysisMethod |
| 参数 | LoginUser loginUser 系统登陆人信息，用于错误代码WebSocket显示前端 @RequestBody List<String> jsonPojoList 分析逻辑集合 Long basicInformationId 交换机基本信息命令ID |
| 逻辑 | 遍历分析数据 如果分析数据中含有"command" 则 是命令 则 进入 String转命令实体类方法 如果分析数据中不含有"command" 则 是分析 则 进入 String转分析实体类方法 注：如果当前集合元素是分析 则 需要考虑下一集合元素是 命令 还是分析 如果是命令 则analysisCommandLogic 在 Sting转分析实体类中 传入 命令属性值 如果是分析 则analysisProblemScanLogic 在 Sting转分析实体类中 传入 分析属性值 这会影响到 前端数据传入的 下一ID 的 赋值问题  再经历，definitionProblem相同ID的分析实体类需要放到一个实体类中(因为这里是ture和false的原因，造成了一个实体类分割成了两个相同ID的实体类)  获取交换机基本信息第一条数据为ID 需要传送给 获取交换机基本信息命令的分析ID |
| 引用方法 | analysisCommandLogic、analysisProblemScanLogic、definitionProblem |
| 返回结果 | boolean true：插入成功 false：插入失败 |

### 3.字符串解析CommandLogic实体类

方法名：analysisCommandLogic

|  |  |
| --- | --- |
| 字符串解析CommandLogic实体类 | |
| 方法名 | analysisCommandLogic |
| 路径 | /sql/DefinitionProblemController/analysisCommandLogic |
| 参数 | String jsonPojo 可以解析为 命令表实体类的字符串 |
| 逻辑 | 第一步：去掉“{”“}”，然后以“，”分割为数组。格式为：（属性名：属性值） 第二步：创建hashMap散列表，存入 以命令表字段名为key值的 键值对，value值为null或默认值 第三步：遍历属性数组，以“：”分割为[“属性名”，“属性值”]的数组 ，使用 属性名 匹配 hashmap中的key值，给key值赋值 第四步：如果为常规校验的话，resultCheckId = 1；则分析数据的下一条ID为下一命令ID。则nextIndex属性值 应赋值给 实体类endIndex字段。 |
| 引用方法 |  |
| 返回结果 | CommandLogic 命令实体类 |

### 字符串解析ProblemScanLogic实体类

方法名：analysisProblemScanLogic

|  |  |
| --- | --- |
| 字符串解析ProblemScanLogic实体类 | |
| 方法名 | analysisProblemScanLogic |
| 路径 | /sql/DefinitionProblemController/analysisProblemScanLogic |
| 参数 | String jsonPojo 可以解析为 命令表实体类的字符串 String ifCommand 值为： 命令 分析 影响下一条ID的归属 |
| 逻辑 | 字符串转化为分析实体类比较复杂。关键字在会有“，”“：”存在。 第一步，去掉“{”“}”。然后 单引号 替换为 双引号。通过替换，使属性名和属性值的结构都变化为--- “key”：“value”。 第二步，创建hashMap散列表，存入 以命令表字段名为key值的 键值对，value值为null或默认值 第三步：遍历属性数组，以“：”分割为[“属性名”，“属性值”]的数组 ，使用 属性名 匹配 hashmap中的key值，给key值赋值   当匹配方式不为空 且 包含 按行 时 则为 按行匹配 则需要 拼接 relativePosition 值 如果下一条分析数据为命令时 则 下一条ID 赋值给 命令ID 如果 trueFalse 为 失败时 则 成功行号、成功下一条分析、成功下一条命令 都复制给 失败对应 属性 如果action动作属性不为空 且动作属性参数为 循环时 需要清空动作属性 如果动作属性不为空 且动作属性参数为 比较时 需要清空动作属性 如果动作属性不为空 且动作属性参数为 有无问题时 需要清空动作属性 action 前端传入后端有值，后端返回前端有值，根据不同字段来确定。 |
| 引用方法 |  |
| 返回结果 | problemScanLogic 分析实体类 |

### 将时间戳相同的实体类放到一个实体

方法名：definitionProblem

|  |  |
| --- | --- |
| 将时间戳相同的实体类放到一个实体 | |
| 方法名 | definitionProblem |
| 路径 | /sql/DefinitionProblemController/definitionProblem |
| 参数 | List<ProblemScanLogic> pojoList 分析实体类集合 |
| 逻辑 | 第一步：创建hashset集合。将实体类集合放入set确保ID不重复。 第二步：将hashset集合的元素，作为ID创建一个新的实体类集合NewList。 第三步：遍历NewList，拿实体类ID匹配接口传入实体类集合。当ID一致时，实体类信息放入newlist的该元素中的对应字段 |
| 引用方法 |  |
| 返回结果 | List<ProblemScanLogic> 分析实体类集合 |

## 回显定义获取交换机基本信息数据

### 查询获取基本信息命令列表

方法名：getPojolist

|  |  |
| --- | --- |
| 查询获取基本信息命令列表 | |
| 方法名 | getPojolist |
| 路径 | /sql/basic\_information/getPojolist |
| 参数 |  |
| 逻辑 | mybatise查询集合，传入参数为null，则获取所有数据 |
| 引用方法 |  |
| 返回结果 | List<BasicInformation> 获取基本信息命令列表集合 |

### 回显获取交换机基本信息逻辑数据

方法名：getBasicInformationProblemScanLogicTimeouts

路径：/sql/DefinitionProblemController/getBasicInformationProblemScanLogicTimeouts

是方法：getBasicInformationProblemScanLogic的超时方法

方法名：getBasicInformationProblemScanLogic

|  |  |
| --- | --- |
| 回显获取交换机基本信息逻辑数据 | |
| 方法名 | getBasicInformationProblemScanLogic |
| 路径 | /sql/DefinitionProblemController/getBasicInformationProblemScanLogic |
| 参数 | String problemId 获取交换机基本信息命令 分析ID LoginUser loginUser 登陆人信息 |
| 逻辑 | 第一步：引用方法problemScanLogicList，根据分析ID，获取分析表实体类集合，并拆分true和false。 第二步：遍历分析表实体类集合，通过调用方法problemScanLogicSting将分析表实体类转化为String类型。并返回(行号:String)集合  第三步：根据行号排序，返回String集合。 |
| 引用方法 | *problemScanLogicList*，*problemScanLogicSting* |
| 返回结果 | List<String> 分析表实体类转化为String的集合 |

### 根据分析ID获取分析实体类集合并拆分true、false

方法名：problemScanLogicList

|  |  |
| --- | --- |
| 根据分析ID获取分析实体类集合并拆分true、false | |
| 方法名 | problemScanLogicList |
| 路径 | /sql/DefinitionProblemController/problemScanLogicList |
| 参数 | String problemScanLogicID 获取交换机基本信息分析ID LoginUser loginUser 登陆人信息 |
| 逻辑 | 第一：根据分析逻辑ID在分析逻辑表中查询分析逻辑数据。并存入map集合中。 第二：查出来的分析逻辑数据的 正确下一步ID 和 错误下一步ID 拼接起来分析逻辑ID字符串。 第三：进行下一轮查询，放入map中。查询前先在map中查询是否已经查询过。  查询过，则进行本轮中下一ID的查询。直到拼接起的字符串为空 则 退出并返回。 |
| 引用方法 |  |
| 返回结果 | List<String> 分析表实体类转化为String的集合 |

### 分析逻辑实体类转Sting

方法名：problemScanLogicSting

|  |  |
| --- | --- |
| 分析实体类转Sting | |
| 方法名 | problemScanLogicSting |
| 路径 | /sql/DefinitionProblemController/problemScanLogicSting |
| 参数 | ProblemScanLogic problemScanLogic 分析实体类 String id 交换机问题ID |
| 逻辑 | 第一步：根据前端传输数据的属性名，定义一个新的实体类VO。 第二步：将对应属性值传给实体类VO的对应属性名。(匹配字段matched会根据相对位置字段RelativePosition的值来确定是按行还是全文；比较逻辑字段Compare和问题ID字段ProblemId 来决定 动作字段Action 为"比较"还是"问题"；) 第三步：根据前端需要格式拼接 |
| 引用方法 |  |
| 返回结果 | String 分析表实体类转化为String 格式：（行号：前端需要的格式String） |

## 删除获取交换机基本信息逻辑

方法名：deleteBasicInformationProblemScanLogic

|  |  |
| --- | --- |
| 删除获取交换机基本信息逻辑 | |
| 方法名 | deleteBasicInformationProblemScanLogic |
| 路径 | /sql/DefinitionProblemController/deleteBasicInformationProblemScanLogic |
| 输入参数 | Long id 交换机问题ID |
| 逻辑 | 第一步：根据交换机问题ID，查询交换机问题实体类 第二步：获取交换机问题实体类分析ID 第三步：根据分析ID调用problemScanLogicList获取分析实体类集合。 第四步：利用set集合 给分析实体类集合ID 去重。 第五步：set集合 转换为 String数组。 第六步：根据String数组删除分析实体类集合，成功后删除交换机问题实体类 |
| 调用方法 | *problemScanLogicList* |
| 返回结果 | boolean true & false |

## 获取交换机基本信息逻辑（单步）

方法名：testToObtainBasicInformation

|  |  |
| --- | --- |
| 测试获取交换机基本信息逻辑执行结果 | |
| 方法名 | testToObtainBasicInformation |
| 路径 | /sql/SwitchInteraction/testToObtainBasicInformation |
| 参数 | @PathVariable String ip, IP地址 @PathVariable String name, 用户名 @PathVariable String password, 密码 @PathVariable String port, 端口号 @PathVariable String mode, 连接方式 @PathVariable String configureCiphers, 配置密码 @PathVariable String command, 命令 @RequestBody List<String> pojoList 分析逻辑String集合 |
| 逻辑 | 第一步：创建用户及交换机信息Map user\_String 存放交换机信息、获取交换机基本信息时间等。 第二步：创建交换机连接Map user\_Object 及当前登陆人信息LoginUser 第三步：通过传输用户及交换机信息Map user\_String，调用连接交换机方法requestConnect。连接交换机。如果连接失败，返回失败。如果连接成功，则得到 对应方法的 与交换机的连接方式（自己编写的代码）和连接工具（查找的代码库），得到这些是为了应对多线程。 第四步：遍历分析逻辑字符串集合：List<String> pojoList 通过 调用 analysisProblemScanLogic 方法 将字符串 转化为 分析逻辑实体类，并放入 分析逻辑实体类集合 List<ProblemScanLogic> problemScanLogicList。 第五步：通过调用方法definitionProblem 将相同ID(时间戳)的实体类放到一个实体中 第六步：调用getBasicInformationTest 获取交换机基本信息  第七步：以固定格式String 返回字符串 |
| 调用方法 | *requestConnect、analysisProblemScanLogic、 definitionProblem、getBasicInformationTest* |
| 返回结果 | String 返回字符串 |

## 连接交换机

方法名：requestConnect

|  |  |
| --- | --- |
| 连接交换机方法 | |
| 方法名 | requestConnect |
| 路径 | /sql/SwitchInteraction/requestConnect |
| 参数 | Map<String,String> user\_String   user\_String.put("mode",mode);//登录方式 user\_String.put("ip",ip);//ip地址 user\_String.put("name",name);//用户名 user\_String.put("password",password);//用户密码 user\_String.put("configureCiphers",configureCiphers);//配置密码 user\_String.put("port",port+"");//登录端口号 //扫描时间 获取当前时间 时间格式为 "yyyy-MM-dd HH:mm:ss" 字符串 user\_String.put("ScanningTime",scanningTime); //交换机信息 预设为空 //设备型号 user\_String.put("deviceModel",null); //设备品牌 user\_String.put("deviceBrand",null); //内部固件版本 user\_String.put("firmwareVersion",null); //子版本号 user\_String.put("subversionNumber",null); |
| 逻辑 | 第一步：取出连接方式、IP、用户名、密码、配置密码、端口号 第二步：通过连接方式选择SSH还是Telnet ，通过IP、用户名、密码、配置密码、端口号连接交换机，如果返回结果不为空，则连接成功。 |
| 调用方法 |  |
| 返回结果 | AjaxResult  如果连接失败 返回：AjaxResult.error("交换机连接失败"); 如果连接成功则 返回：AjaxResult.success(objectMap); Map集合:   Map<String,Object> objectMap = new HashMap<>();  //设定返回值 Map集合 //元素0 ：是否连接成功  objectMap.put("TrueAndFalse",is\_the\_connection\_successful);  //元素1 ：连接方法  objectMap.put("way",way);  //元素2 ：交换机ID  objectMap.put("hostIp",hostIp);  //元素3 ：交换机登录用户  objectMap.put("userName",userName);  //元素4 ：交换机登录用户密码  objectMap.put("userPassword",userPassword);  //元素5 ：交换机连接端口号  objectMap.put("portID",portID);  //元素6 ：ssh连接对象：如果连接方法为telnet则connectMethod为空，插入connectMethod失败  objectMap.put("connectMethod",connectMethod);  //元素7 ：telnet连接对象：如果连接方法为ssh则telnetSwitchMethod为空，插入telnetSwitchMethod失败  objectMap.put("telnetSwitchMethod",telnetSwitchMethod);  //元素8 ：ssh连接工具对象  objectMap.put("sshConnect",sshConnect);  //元素9 ：telnet连接工具对象  objectMap.put("telnetComponent",telnetComponent);  //元素10 ： 配置密码configureCiphers  objectMap.put("configureCiphers",configureCiphers); |

## 获取交换机基本信息（单步）

方法名：getBasicInformationTest

|  |  |
| --- | --- |
| 测试获取交换机基本信息 | |
| 方法名 | getBasicInformationTest |
| 路径 | /sql/SwitchInteraction/getBasicInformationTest |
| 参数 | Map<String,String> user\_String, 交换机信息 Map<String,Object> user\_Object, 登陆人信息 String commands , 获取交换机信息 命令 List<ProblemScanLogic> problemScanLogicList 获取交换机信息 分析逻辑 |
| 逻辑 | 第一步：命令以","分割为 命令数组。遍历命令数组，获取返回字符串，调用方法：switchfailure判断命令返回结果是否有错误， 第二步：命令返回结果如果有错误，则将返回信息及故障问题插入数据库，并重新执行当前命令(此时要求命令不会陷入死循环，故交换机故障表，故障信息不会一直存在) 当交换机故障跳出时，拆入数据库的ID需要记录，并是指查询数据出来。 第三步：调用方法removeLoginInformation，去除交换机登录信息 第四步：调用方法trimString 交换机返回信息 修整字符串 去除多余 "\r\n" 连续空格 为插入数据美观。 第五步：截取返回日志和标识符，修改刚插入的，交换机返回信息数据库数据。调用judgmentError查看交换机命令是否错误。错误中止，如果无误则拼接在一起。并进行命令数组的下一命令。 第六步：调用方法analysisReturn 获取分析结果。 |
| 调用方法 | *switchfailure、removeLoginInformation、 trimString、judgmentError、analysisReturn* |
| 返回结果 | AjaxResult |

### analysisReturn 根据交换机返回信息、分析ID 分析逻辑集合获取交换机基本信息

方法名：analysisReturn

|  |  |
| --- | --- |
| 根据交换机返回信息、分析ID 分析逻辑集合获取交换机基本信息 | |
| 方法名 | analysisReturn |
| 路径 | /sql/SwitchInteraction/analysisReturn |
| 参数 | Map<String,String> user\_String, 交换机信息 Map<String,Object> user\_Object, 登录人信息 TotalQuestionTable totalQuestionTable, 问题表信息 String resultString, 交换机返回结果 String first\_problem\_scanLogic\_Id, 分析ID  List<ProblemScanLogic> problemScanLogicList 分析逻辑实体类（用于预执行） |
| 逻辑 | 第一步：交换机返回信息 按 \r\n 分割，获取配置文件设置最大超时时间。 第二步：调用selectProblemScanLogicById 进行分析流程。返回分析结果 第三步：调用removeIdentifier 去除 配置文件中设置的标识符。 |
| 调用方法 | *selectProblemScanLogicById、removeIdentifier* |
| 返回结果 | String 提取出的数据 |

## 分析逻辑流程

方法名：selectProblemScanLogicById

|  |  |
| --- | --- |
| 分析逻辑流程 | |
| 方法名 | selectProblemScanLogicById |
| 路径 | /sql/SwitchInteraction/selectProblemScanLogicById |
| 输入参数 | Map<String,String> user\_String, 交换机信息 Map<String,Object> user\_Object, 用户信息 TotalQuestionTable totalQuestionTable, 问题表信息 String[] return\_information\_array, 交换机返回结果以\r\n分割数组 String current\_Round\_Extraction\_String, 单次分析提取数据 String extractInformation\_string, 循环分析提取数据 int line\_n, 交换机返回信息字符串分析索引位置(光标) String firstID , 第一条分析ID List<ProblemScanLogic> problemScanLogicList, 分析逻辑实体类集合（预获取基本信息） String currentID, 当前分析ID Integer insertsInteger, 插入问题数据次数 为了“完成” Integer loop, 循环次数 Integer numberOfCycles 最大循环次数 |
| 逻辑 | 第一步：判断当前分析ID(currentID)是否为空。如果为空则用第一条分析ID(firstID).如果当前分析ID(currentID)不为空，说明是第二次调用本方法，则使用当前分析ID(currentID)，都赋值给ID。 第二步，判断输入参数problemScanLogicList分析逻辑实体类集合是否为空，不为空则，分析逻辑数据通过problemScanLogicList来存储。如果为空，则需要查询数据库。都是通过ID来获取具体分析逻辑数据。  第三步：如果获取的具体分析逻辑数据，循环字段不为空的话，则说明是循环逻辑，比较循环次数和最大循环测试。需要调用本方法，调出循环ID，当做当前分析ID继续进行分析。  第四步：查看问题ID(ProblemId)字段。如果该字段不为空，则分析出问题了。1：如果ProblemId字段包含"问题" 或者 ProblemId字段不包含"问题"并且ProblemId字段不包含"完成"(自定义的问题名称)。  如果是自定义的问题名称，则像前端报告，且写入日志。 得出结果，然后调用insertSwitchScanResult方法，插入扫描结果数据库。插入问题数据次数insertsInteger 加1，单次分析提取数据 清空。 调用获取扫描结果方法getSwitchScanResultListByData以websocket传到前端实时显示。 如果tNextId下一分析ID(此时tNextId默认为下一分析ID)不为空时，则tNextId赋值给当前分析ID 调用本方法，继续分析流程。 如果tComId不为空时，则调用方法executeScanCommandByCommandId发送新命令，通过analysisReturnResults进行新的分析。  第五步：根据Matched 取出匹配字段，根据Action 取出 动作，根据Compare 取出取词。 |
| 第六步：根据光标遍历交换机返回信息的字符串数组，根据第五步三个字段获得分析操作。  第七步：匹配字段不为空时：通过matchAnalysis匹配方法，返回匹配结果。true和false， 当匹配成功时，取出成功命令ID，如果成功命令ID不为空，则进行下一个命令并进行分析。如果成功命令ID为空，则查看成功下一分析ID，并将成功下一分析ID赋值给当前分析ID，调用本方法继续分析。 当匹配失败时，查看是否为全文匹配，是则遍历交换机返回信息的字符串数组下一元素，遍历到最后一个元素截止。如果不是全文匹配，则进行错误的命令ID或者错误下一分析ID。  第八步：取词逻辑：包含取品牌、型号、内部固件版本、子版本号 及 自定义取词。 取词方法wordSelection 返回取词内容，如果取词内容为null 则去次失败，则返回去次失败。如果取词内容不为null，则拼接取词字符串。取出成功命令ID，如果成功命令ID不为空，则进行下一个命令并进行分析。如果成功命令ID为空，则查看成功下一分析ID，并将成功下一分析ID赋值给当前分析ID，调用本方法继续分析。  第九步： 比较逻辑不为空，则 compareVersion 比较内容， 成功进行成功的下一命令或分析，失败进行失败的下一命令和分析 |
| 调用方法 | *insertSwitchScanResult、getSwitchScanResultListByData、executeScanCommandByCommandId、analysisReturnResults、matchAnalysis、wordSelection、compareVersion* |
| 返回结果 | String 取词信息 |

### 扫描结果插入数据库

方法名：*insertSwitchScanResult*

|  |  |
| --- | --- |
| 分析逻辑流程 | |
| 方法名 | selectProblemScanLogicById |
| 路径 | /sql/SwitchInteraction/selectProblemScanLogicById |
| 输入参数 | Map<String,String> user\_String, 交换机信息 Map<String,Object> user\_Object, 登陆人信息 TotalQuestionTable totalQuestionTable, 交换机问题 ProblemScanLogic problemScanLogic, 分析逻辑数据 String parameterString 单次提取信息 |
| 逻辑 | 取出：品牌、型号、内部固件版本，子版本。 ip、用户名、密码、配置密码、连接方式、端口号 问题索引，问题名称(三个)，备注，注释索引 解决问题，单次提取信息，扫描人员姓名，手机号 |
| 调用方法 |  |
| 返回结果 |  |

### websocket查询扫描出的问题

*方法名：getSwitchScanResultListByData*

|  |  |
| --- | --- |
| websocket查询扫描出的问题表 | |
| 方法名 | getSwitchScanResultListByData |
| 路径 | /sql/SwitchInteraction/getSwitchScanResultListByData |
| 输入参数 | Map<String,String> user\_String 交换机信息，扫描时间 Map<String,Object> user\_Object 登录人信息 |
| 逻辑 | 根据用户名和扫描时间，从交换机扫描结果表switchScanResult中查询，本次扫描结果数据。 |
| 调用方法 |  |
| 返回结果 | List<ScanResultsVO> 前端需要格式集合 |

### 根据命令ID获取具体命令执行并返回结果

*方法名：executeScanCommandByCommandId*

|  |  |
| --- | --- |
| 根据命令ID获取具体命令执行并返回结果 | |
| 方法名 | executeScanCommandByCommandId |
| 路径 | /sql/SwitchInteraction/executeScanCommandByCommandId |
| 输入参数 | Map<String,String> user\_String, 交换机信息 TotalQuestionTable totalQuestionTable, 扫描交换机问题 String commandId, 扫描交换机问题命令ID String notFinished, 交换机返回信息未结束的标志 String way, 连接交换机方法 Map<String,Object> user\_Object 登录人信息 |
| 逻辑 | 第一步：扫描交换机问题命令ID 获取 扫描问题需要发送的命令CommandLogic，通过way选择发送方式。接受返回结果。 第二步：调用方法switchfailure。粗略查看是否存在故障，存在则websocket故障信息，写入日志，并将故障信息插入数据库。如果不存在故障，则将交换机返回信息插入数据库。记录最后数据ID 第三步：通过数据ID，得到交换机返回信息的实体类信息。 第四步：通过removeLoginInformation 去除其他交换机登录信息 第五步：通过trimString修整返回信息去除多余 "\r\n" 连续空格 第六步：截取返回日志和标识符，添加入交换机返回信息的实体类信息。 第七步：通过交换机返回信息的实体类信息，修改数据库信息。 第八步：通过judgmentError判断命令是否错误 第九步：判断是否简单检验 1L为简单校验 默认0L为分析数据表自定义校验，查看是否进行下一命令。 //连续多个命令后 执行分析情况。 |
| 调用方法 | *switchfailure、removeLoginInformation、trimString、judgmentError* |
| 返回结果 | List<Object> List<Object> objectList = new ArrayList<>(); objectList.add(command\_string);//交换机返回信息 System.err.println("\r\n交换机返回信息:"+command\_string+"\r\n"); objectList.add(first\_problem\_scanLogic\_Id);//分析第一条ID |

### 执行分析

*方法名：analysisReturnResults*

|  |  |
| --- | --- |
| 执行分析 | |
| 方法名 | analysisReturnResults |
| 路径 | /sql/SwitchInteraction/analysisReturnResults |
| 输入参数 | Map<String,String> user\_String, 交换机信息 Map<String,Object> user\_Object, 登陆人信息 TotalQuestionTable totalQuestionTable, 交换机问题 List<Object> executeScanCommandByCommandId\_object, 执行命令的返回结果(交换机返回信息、分析ID) String current\_Round\_Extraction\_String, 单次分析提取数据 String extractInformation\_string 循环分析提取数据 |
| 逻辑 | 第一步：交换机返回信息、分析ID、最大循环次数。 第二步：调用方法selectProblemScanLogicById 返回String取词信息 |
| 调用方法 | *selectProblemScanLogicById* |
| 返回结果 | String String取词信息不为null则取词信息 否则 null |

## 定义交换机问题

*方法名：add*

|  |  |
| --- | --- |
| 定义交换机问题 | |
| 方法名 | add |
| 路径 | /sql/total\_question\_table/add |
| 输入参数 | TotalQuestionTable totalQuestionTable 交换机问题实体类 |
| 逻辑 | 第一步：型号、内部固件版本、子版本 如果为""，则赋值为"\*" 第二步：根据实体类，查询实体类集合。如果不为null，则返回"问题已存在" 第三步：插入实体类 |
| 调用方法 |  |
| 返回结果 | AjaxResult true 返回实体类ID & false 返回error |

## 定义交换机问题分析

*方法名：definitionProblemJson*

*直接调用 definitionProblemJsonPojo 方法*

*因为需要 日志 definitionProblemJsonPojo 几个方法会调用。*

*方法名：definitionProblemJsonPojo*

|  |  |
| --- | --- |
| 定义交换机问题分析 | |
| 方法名 | definitionProblemJsonPojo |
| 路径 | /sql/DefinitionProblemController/definitionProblemJsonPojo |
| 输入参数 | List<String> jsonPojoList 交换机分析实体类 |
| 逻辑 | 遍历分析数据 如果分析数据中含有"command" 则 是命令 则 进入 String转命令实体类方法 如果分析数据中不含有"command" 则 是分析 则 进入 String转分析实体类方法 注：如果当前集合元素是分析 则 需要考虑下一集合元素是 命令 还是分析 如果是命令 则analysisCommandLogic 在 Sting转分析实体类中 传入 命令属性值 如果是分析 则analysisProblemScanLogic 在 Sting转分析实体类中 传入 分析属性值 这会影响到 前端数据传入的 下一ID 的 赋值问题  再经历，definitionProblem相同ID的分析实体类需要放到一个实体类中(因为这里是ture和false的原因，造成了一个实体类分割成了两个相同ID的实体类)  遍历分析实体类 插入数据库，并查看包含的"有问题"或"无问题"的ProblemId，截取问题ID，  遍历命令和分析，行号等于1的数据ID，需要"命令"+数据ID ("分析"+数据ID)，给到交换机问题表CommandId字段。  根据截取的问题ID，查询交换机问题。给实体类CommandId字段赋值。修改实体类。 |
| 调用方法 | *analysisCommandLogic、analysisProblemScanLogic、definitionProblem* |
| 返回结果 | boolean true & false |

## 回显交换机问题及分析

### 查询交换机问题列表PojoList

*方法名：selectPojoList*

|  |  |
| --- | --- |
| 查询交换机问题列表PojoList | |
| 方法名 | selectPojoList |
| 路径 | /sql/total\_question\_table/selectPojoList |
| 输入参数 | TotalQuestionTable totalQuestionTable 交换机问题实体类 |
| 逻辑 | 根据交换机问题实体类，查询交换机问题实体类列表  如果传入参数为null，则查询所有。 |
| 调用方法 |  |
| 返回结果 | List<TotalQuestionTable> 交换机问题实体类列表 |

### 获取交换机问题的详细信息

*方法名：*getInfo

|  |  |
| --- | --- |
| 获取交换机问题的详细信息 | |
| 方法名 | getInfo |
| 路径 | /sql/total\_question\_table/{id} |
| 输入参数 | Long id 交换机问题实体类ID |
| 逻辑 | 根据交换机问题实体类ID，查询交换机问题实体类 |
| 调用方法 |  |
| 返回结果 | AjaxResult success(交换机问题实体类) |

### 根据交换机问题实体类查询问题分析逻辑数据

*方法名：getAnalysisListTimeouts*

*是getAnalysisList 方法的 定时方法*

*方法名：getAnalysisList*

|  |  |
| --- | --- |
| 根据交换机问题实体类查询问题分析逻辑数据 | |
| 方法名 | getAnalysisList |
| 路径 | /sql/DefinitionProblemController/getAnalysisList |
| 输入参数 | TotalQuestionTable totalQuestionTable 交换机问题实体类 LoginUser loginUser 登陆人信息 |
| 逻辑 | 第一步：查看交换机问题实体类CommandId字段是否为空。如果CommandId字段为空，则是传输的七个字段，需要查询详细信息。不为空，则为实体类详细信息可以直接使用。 第二步：将CommandId字段取出，去除"命令"或"分析"。 第三步： 如果 CommandId字段 为分析ID，则需要调用 problemScanLogicList方法 根据分析ID 获取 分析实体类集合 并拆分 true false。 并将 分析ID 置空。 第四步：遍历分析实体类集合，如果hashmap中不存在，则存放hashmap中，并筛选出 命令ID； 第五步：如果 CommandId字段 为命令ID，则需要查询命令实体类，如果hashmap中不存在，则存放hashmap中，通过ProblemId调用 problemScanLogicList方法 根据分析ID 获取 分析实体类集合 并拆分 true false。 并将 命令ID 置空。 第六步：遍历分析实体类集合，如果hashmap中不存在，则存放hashmap中，并筛选出 命令ID； 第七步：判断 命令ID 是否为空，为空回到第三步。 第八步：遍历命令实体类集合，通过commandLogicString得到命令实体的字符串及行号，放入hashmap中，（行号，命令实体的字符串） 第九步：遍历分析实体类集合，通过problemScanLogicSting得到分析实体的字符串及行号，放入hashmap中，（行号，分析实体的字符串） 第十步：遍历hashmap，通过行号排大小。存入List<String>； |
| 调用方法 | *problemScanLogicList、commandLogicString、problemScanLogicSting* |
| 返回结果 | List<String> 命令及分析的字符串集合 |

### commandLogic命令表转化为String

*方法名：*commandLogicString

|  |  |
| --- | --- |
| commandLogic命令表转化为String | |
| 方法名 | commandLogicString |
| 路径 | /sql/DefinitionProblemController/commandLogicString |
| 输入参数 | CommandLogic commandLogic 交换机命令实体类 |
| 逻辑 | 第一步：创建CommandLogicVO类，贴近前端需要的属性和各式。 第二步：如果命令包含":",则说明命令带参数。需要根据":"分割。[0]为命令，[1]为参数。 第三步：字段resultCheckId为0时，下一ID为自定义分析ProblemId();字段resultCheckId为1时，常规检验执行下一命令,所以下一ID为下一命令EndIndex(); 第四步：返回 行号 + 前端需要的字符串格式 |
| 调用方法 |  |
| 返回结果 | String 行号 + 前端需要的字符串格式 |

## 交换机扫描

### 扫描全部问题

方法名：multipleScans

|  |  |
| --- | --- |
| 扫描全部问题 | |
| 方法名 | multipleScans |
| 路径 | /sql/SwitchInteraction/multipleScans |
| 输入参数 | List<String> switchInformation 交换机登录信息字符串集合 Long scanNum 线程数 |
| 逻辑 | 第一步：通过分割交换机登录信息String，得到交换机登录信息集合。 第二步：获取扫描时间，和登录人信息 第三步：通过调用线程方法switchLoginInformations 开启扫描线程 |
| 调用方法 | *switchLoginInformations* |
| 返回结果 | String "扫描结束" |

### 扫描线程

方法名：switchLoginInformations

|  |  |
| --- | --- |
| 扫描线程 | |
| 方法名 | switchLoginInformations |
| 路径 | ScanFixedThreadPool.switchLoginInformations |
| 输入参数 | List<Object[]> objects, 交换机登录信息 String ScanningTime, 扫描时间 LoginUser login, 登陆人信息 int threads 线程数 |
| 逻辑 | 第一步：遍历交换机登录信息集合，取出 登陆方式mode, ip, 用户名name, 密码password,配置密码configureCiphers, 端口号port 第二步：获取扫描时间，和登录人信息。 第三步：调用logInToGetBasicInformation方法进行扫描 |
| 调用方法 | *logInToGetBasicInformation* |
| 返回结果 |  |

### 扫描方法

方法名：logInToGetBasicInformation

|  |  |
| --- | --- |
| 扫描方法 | |
| 方法名 | logInToGetBasicInformation |
| 路径 | /sql/SwitchInteraction/logInToGetBasicInformation |
| 输入参数 | String mode, 连接方法 String ip, ip地址 String name, 用户名 String password, 密码 String configureCiphers , 配置密码 int port, 端口号 LoginUser loginUser, 登陆人信息 String ScanningTime, 扫描时间 List<TotalQuestionTable> totalQuestionTables 需要扫描的交换机问题 |
| 逻辑 | 第一步：创建user\_String交换机信息map集合。存储：登录方式 mode，IP地址 ip，用户名 name，用户密码 password，端口号 port，扫描时间 ScanningTime，设备型号deviceModel，设备品牌deviceBrand，内部固件版本firmwareVersion，子版本号subversionNumber 第二步：创建user\_Object的map集合，存储ssh连接方法，telnet连接方法，系统登录人信息loginUser 第三步：通过requestConnect方法连接交换机，如果连接交换机失败返回"交换机连接失败" 第四步：密码和配置密码 通过MD5加密 并覆盖user\_String。根据连接交换机返回 的连接方法，将ssh：connectMethod参数 telnet：telnetSwitchMethod参数 存放入user\_Object的map集合。 第五步：通过getBasicInformationList 获取交换机基本信息。 第六步：通过scanProblem方法，获取匹配的交换机可执行的命令ID并循环执行分析操作 |
| 调用方法 | *requestConnect、getBasicInformationList、scanProblem* |
| 返回结果 | AjaxResult |

### 获取交换机基本信息

方法名：getBasicInformationList

|  |  |
| --- | --- |
| 获取交换机基本信息 | |
| 方法名 | getBasicInformationList |
| 路径 | /sql/SwitchInteraction/getBasicInformationList |
| 输入参数 | Map<String,String> user\_String 交换机信息Map Map<String,Object> user\_Object 登陆人信息 |
| 逻辑 | 第一步：查询获取基本信息命令表中的全部命令，得获取基本信息命令集合。 第二步：遍历获取基本信息命令集合，根据","分割命令，得到命令数组。 第三步：预设交换机返回结果、交换机返回结果拼接长字符串 第四步：遍历命令数组，发送命令，获取交换机返回结果。 第五步：调用switchfailure方法查看是否存在故障。有故障，报告前端及写入日志，交换机返回故障问题插入数据库，重新执行该命令。无问题，则交换机返回信息插入数据库。记录插入数据库数据ID。 第六步：根据数据ID查询插入数据库交换机返回结果。 第七步：调用removeLoginInformation去除其他交换机登录信息 第八步：调用trimString修整交换机返回信息字符串去除多余"\r\n" 连续空格 第九步：分割返回日志内容和标识符，赋值交换机返回信息。 第十步：调用judgmentError判断命令是否错误，正确为true，错误为false并报告前端及写入日志，执行下一个 命令数组元素。 第十一步：通过 analysisReturn 方法，根据交换机返回信息、分析ID 分析逻辑集合获取交换机基本信息。返回总提取信息。 第十二步：拆分基本信息，当四个基本信息都存在后，返回结果(基本属性得map) |
| 调用方法 | *switchfailure、removeLoginInformation、trimString、judgmentError、analysisReturn* |
| 返回结果 | AjaxResult |

### 获取交换机可扫描的问题并执行分析操作

方法名：scanProblem

|  |  |
| --- | --- |
| 获取交换机可扫描的问题并执行分析操作 | |
| 方法名 | scanProblem |
| 路径 | /sql/SwitchInteraction/scanProblem |
| 输入参数 | Map<String,String> user\_String, 交换机信息 Map<String,Object> user\_Object, 登陆人信息 List<TotalQuestionTable> totalQuestionTables 交换机定向扫描，交换机问题 |
| 逻辑 | 第一步：当交换机定向扫描的交换机问题不为空时，则遍历问题集合，选出基本属性匹配的数据。否则，根据交换机基本信息，数据库查询交换机可扫描问题。 赋值交换机问题集合List<TotalQuestionTable> totalQuestionTableList。 第二步：遍历交换机问题集合totalQuestionTableList，根据同一问题扫描多次，选取基本信息更详细问题。筛选集合列表。存入map中key为范式类型和范式名称。value值为交换机问题实体类。 第三步：map集合转化为交换机问题实体类list集合TotalQuestionTablePojoList。 第四步：遍历TotalQuestionTablePojoList集合，根据交换机问题实体类CommandId字段，包含"命令"还是"分析"。 第五步：如果包含"命令"，则通过NotFinished字段获取交换机返回信息未结束标识。调用 executeScanCommandByCommandId 向交换机发送命令 返回 交换机返回信息及分析ID。调用analysisReturnResults方法，进行分析，返回取词。 第六步：如果包含"分析"，则创建List<Object>集合，第一元素赋值""空字符充当交换机返回结果，第二元素赋值分析ID,调用analysisReturnResults方法，进行分析，返回取词。 |
| 调用方法 | *executeScanCommandByCommandId、analysisReturnResults* |
| 返回结果 | AjaxResult |

### 获取交换机可扫描的问题并执行分析操作

*方法名：selectProblemScanLogicById*

|  |  |
| --- | --- |
| 获取交换机可扫描的问题并执行分析操作 | |
| 方法名 | scanProblem |
| 路径 | /sql/SwitchInteraction/scanProblem |
| 输入参数 | Map<String,String> user\_String, 登交换机信息 Map<String,Object> user\_Object, 登陆方法 TotalQuestionTable totalQuestionTable, 扫描问题 String[] return\_information\_array, 交换机返回信息字符串 String current\_Round\_Extraction\_String, 单次分析提取数据 String extractInformation\_string, 循环分析提取数据  int line\_n, 扫描到的位置 String firstID , 第一条分析ID  List<ProblemScanLogic> problemScanLogicList, 扫描逻辑，与扫描获取基本信息用 String currentID, 当前分析ID  Integer insertsInteger, 插入问题数据次数  Integer loop,循环次数 Integer numberOfCycles 最大循环次数 |
| 逻辑 | 第一步：获取当前登陆人信息。判断当前分析ID(currentID)是否为空。如果为空则用第一条分析ID(firstID).如果当前分析ID(currentID)不为空，说明是第二次调用本方法，则使用当前分析ID(currentID)，都赋值给ID。 第二步：判断输入参数problemScanLogicList分析逻辑实体类集合是否为空，不为空则，分析逻辑数据通过problemScanLogicList来存储。如果为空，则需要查询数据库。都是通过ID来获取具体分析逻辑数据。 第三步：判断如果循环ID不为空的话 说明 分析数据为循环分析 则 需要调出循环ID 当做 当前分析ID 继续执行。循环分析数据 不需要分析 功能指向循环位置 第四步：如果 问题索引字段 不为空 null 则 说明 分析数据 是 分析出问题或者可以结束了problemScanLogic.getProblemId() 可以为 有问题(前端显示:异常) 无问题(前端显示:安全) 完成 第五步：判断出是否有问题，显示到前端，使用insertSwitchScanResult和getSwitchScanResultListByData方法，插入和显示前端。 第六步：根据RelativePosition获取全文或是按行。获取RelativePosition得到相对位置，行或列。 第七步：当行位置不为空时，光标位置 line\_number 添加行位置数，实现光标跳动。 第八步：获取匹配 matched 、取词 action 、比较 compare 逻辑关键字。 第九步：根据光标 line\_number 遍历交换机返回信息数组，当 matched 不为空的时候为匹配逻辑，当action不为空时为取词逻辑，当 compare 不为空时为比较逻辑。  匹配逻辑：调用 matchAnalysis 实现匹配 ，当匹配成功时，显示取词成功，并调用 trueLogic 成功逻辑，当匹配失败时，显示取词失败，并调用 falseLogic 失败逻辑。 取词逻辑：调用 wordSelection 实现取词，当取词为null时，显示取词错误并返回。取词结果拼接取词字符串。并执行 trueLogic 成功操作。 比较逻辑：调用 compareVersion 实现比较，比较结果显示前端。当比较成功时，并调用 trueLogic 成功逻辑，当比较失败时，并调用 falseLogic 失败逻辑。 |
| 调用方法 | *insertSwitchScanResult、getSwitchScanResultListByData、 matchAnalysis、trueLogic、falseLogic、wordSelection、compareVersion* |
| 返回结果 | String |

### 扫描结果插入数据库

*方法名：*insertSwitchScanResult

|  |  |
| --- | --- |
| 扫描结果插入数据库 | |
| 方法名 | insertSwitchScanResult |
| 路径 | /sql/SwitchInteraction/insertSwitchScanResult |
| 输入参数 | Map<String,String> user\_String, 交换机登录信息 Map<String,Object> user\_Object, 交换机登陆方法 TotalQuestionTable totalQuestionTable, 交换机问题 ProblemScanLogic problemScanLogic, 逻辑 String parameterString 单词提取信息 |
| 逻辑 | 第一步：系统登录人的用户名、手机号、有问题还是无问题、问题ID 第二步：给SwitchScanResult实体类赋值，插入数据库 |
| 调用方法 |  |
| 返回结果 |  |

### 成功逻辑 trueLogic

*方法名：*trueLogic

|  |  |
| --- | --- |
| 成功逻辑 trueLogic | |
| 方法名 | trueLogic |
| 路径 | /sql/SwitchInteraction/trueLogic |
| 输入参数 |  |
| 逻辑 | 判断tComId成功命令ID是否为null，如果命令不为空则进executeScanCommandByCommandId执行命令并返回结果。根据分析ID调用analysisReturnResults返回分析。  如果tNextId正确下一分析ID不为null，调用 selectProblemScanLogicById 进行下一步分析 |
| 调用方法 | *executeScanCommandByCommandId、analysisReturnResults、selectProblemScanLogicById* |
| 返回结果 | String |

### 失败逻辑 falseLogic

*方法名：*falseLogic

|  |  |
| --- | --- |
| 失败逻辑 falseLogic | |
| 方法名 | falseLogic |
| 路径 | /sql/SwitchInteraction/falseLogic |
| 输入参数 |  |
| 逻辑 | 判断fComId失败命令ID是否为null，如果命令不为空则进executeScanCommandByCommandId执行命令并返回结果。根据分析ID调用analysisReturnResults返回分析。  如果fNextId正确下一分析ID不为null，调用 selectProblemScanLogicById 进行下一步分析 |
| 调用方法 | *executeScanCommandByCommandId、analysisReturnResults、selectProblemScanLogicById* |
| 返回结果 | String |

## 定义修复问题

|  |  |
| --- | --- |
| 定义修复问题 | |
| 方法名 | insertModifyProblemCommandSet |
| 路径 | /sql/command\_logic/insertModifyProblemCommandSet |
| 输入参数 | Long totalQuestionTableId 交换机问题ID List<String> commandLogicList 修复问题命令 |
| 逻辑 | 第一步：获取系统登陆人信息 第二步：判断修复交换机命令集合是否为空，如果为空，则先前端并返回false 第三步：根据 totalQuestionTableId交换机问题ID 查询交换机问题实体类 TotalQuestionTable 第四步：调用analysisCommandLogic使Sting转化为实体类。List<String>转化为List<CommandLogic> 第五步：遍历List<CommandLogic>插入数据库，第一个元素的ID的记录并存储到 TotalQuestionTable 的修复问题ID字段。  修改 totalQuestionTable 添加 修复问题ID字段 |
| 调用方法 | *executeScanCommandByCommandId、*  *analysisReturnResults、*  *selectProblemScanLogicById* |
| 返回结果 | boolean |

## 删除修复问题

|  |  |
| --- | --- |
| 删除修复问题 | |
| 方法名 | deleteProblemSolvingCommand |
| 路径 | /sql/command\_logic/deleteProblemSolvingCommand |
| 输入参数 | Long totalQuestionTableId |
| 逻辑 | 第一步：根据 totalQuestionTableId 问题表问题ID查询问题数据实体类。 第二步：获取实体类的 ProblemSolvingId 修复问题ID。 第三步：根据 ProblemSolvingId 获取修复命令集合。 第四步：获取 修复命令ID的集合 第五步：根据 修复命令ID的集合 批量删除修复命令 第六步：totalQuestionTable实体类的ProblemSolvingId字段置为null，执行update修改 |
| 调用方法 |  |
| 返回结果 | boolean |

## 修改修复问题

*方法名：updateProblemSolvingCommand*

先调用删除：deleteProblemSolvingCommand

再调用新增：insertModifyProblemCommandSet

## 修复逻辑

方法名：batchSolutionMultithreading

|  |  |
| --- | --- |
| 修复问题 | |
| 方法名 | batchSolutionMultithreading |
| 路径 | /sql/SolveProblemController/batchSolutionMultithreading |
| 输入参数 | List<Object> userinformation, 交换机登录信息 List<String> problemIdList, 交换机扫描结果ID Long scanNum , 线程数 List<String> allProIdList 交换机所有扫描结果ID |
| 逻辑 | 第一步：获取系统当前登陆人，交换机扫描结果ID集合转化为ID数组，并查询 扫描结果集合。 第二步：交换机登录信息集合去重，并解析成交换机登录信息map的集合。 第三步：根据扫描结果集合，与交换机登录信息map的集合，得到登录信息与扫描结果集合对应的两组集合。 第四步：调用 RepairFixedThreadPool . Solution 多线程 |
| 调用方法 | *Solution* |
| 返回结果 |  |

## 修复问题线程

|  |  |
| --- | --- |
| 修复问题 线程 | |
| 方法名 | Solution |
| 路径 | RepairFixedThreadPool . Solution |
| 输入参数 | LoginUser user, 当前登陆人 List<Map<String,String>> userinformation, 交换机登录信息 List<List<SwitchScanResult>> problemList, 交换机问题数据 List<String> problemIdStrings, 交换机所有问题ID int threads 线程 |
| 逻辑 | 第一步：遍历交换机登录信息集合，获取交换机问题数据。 第二步：开启线程，调用 solveProblemController.batchSolution 方法，进行交换机问题修复 |
| 调用方法 | *batchSolution* |
| 返回结果 |  |

## 修复问题

方法名：batchSolution

|  |  |
| --- | --- |
| 修复问题 | |
| 方法名 | batchSolution |
| 路径 | /sql/SolveProblemController/batchSolution |
| 输入参数 | Map<String,String> user\_String, 交换机登录信息 LoginUser loginUser, 系统登陆人信息 List<SwitchScanResult> switchScanResultList , 交换机扫描结果集合 List<String> problemIds ， 交换机所有结果ID |
| 逻辑 | 第一步：SwitchInteraction.requestConnect连接交换机，获取交换机连接方法和连接工具。 第二步：通过getBasicInformationList交换机基本信息 第三步：判断交换机扫描结果集合 与 获取的交换机基本信息是否一致，不一致的显示到前端。 第四步：遍历筛选后的交换机扫描结果集合，根据修复命令ID queryCommandSet 获取命令String集合 第五步：调用separationParameters方法，获得属性与属性值的map集合。 第六步：调用solveProblem执行解决问题 |
| 调用方法 | *requestConnect、getBasicInformationList、separationParameters、solveProblem* |
| 返回结果 | AjaxResult |

## 执行修复问题命令

方法名：solveProblem

|  |  |
| --- | --- |
| 执行修复问题命令 | |
| 方法名 | solveProblem |
| 路径 | /sql/SolveProblemController/solveProblem |
| 输入参数 | Map<String,String> user\_String, 交换机信息 LoginUser loginUser, 系统登陆人 Map<String,Object> objectMap, 交换机连接工具及方法 SwitchScanResult switchScanResult, 交换机扫描结果 List<String> commandList, 修复问题命令 HashMap<String,String> valueHashMap 修复问题参数 |
| 逻辑 | 第一步：遍历修复交换机问题命令，根据参数名 修改交换机命令，并放入修复问题命令集合 第二步：根据连接交换机方法，获取交换机连接工具 第三步：遍历执行修复命令。如果交换机出现异常则重复执行命令，如果出现错误，则跳出修复。交换机返回信息，插入交换机返回信息表。 |
| 调用方法 | *requestConnect、getBasicInformationList、separationParameters、solveProblem* |
| 返回结果 | String |

## 交换机问题、分析、修复表数据导出

|  |  |
| --- | --- |
| 交换机问题、分析、修复表数据导出 | |
| 方法名 | scanningSQL |
| 路径 | /sql/DefinitionProblemController/scanningSQL |
| 输入参数 | totalQuestionTableId 问题ID |
| 逻辑 | 第一步：如果问题ID不为空，则根据ID查询对应交换机问题。如果问题ID为空，则查询所有交换机问题。 第二步：调用 totalQuestionTableExport 方法，将交换机问题集合制成excel表格，并返回名称及路径。 第三步：预先定义扫描命令集合、定义分析集合、定义修复命令集合 第四步：遍历 交换机问题集合 ，根据交换机问题，获取扫描命令集合、分析集合、修复命令集合。并放入预先定义的集合中。 第五步：将预先定义的扫描命令集合、分析集合、修复命令集合，通过commandLogicExport、DefinitionProblemController、repaircommandLogicExport 导出并返回Excel表名。 |
| 调用方法 | *totalQuestionTableExport、commandLogicExport、DefinitionProblemController、repaircommandLogicExport* |
| 返回结果 | AjaxResult |

## 交换机问题、分析、修复表数据导入

|  |  |
| --- | --- |
| 交换机问题、分析、修复表数据导入 | |
| 方法名 | importData |
| 路径 | /sql/DefinitionProblemController/importData |
| 输入参数 | MultipartFile file boolean updateSupport |
| 逻辑 | 第一步：通过 getOriginalFilename();获取导入文件名称。 第二步：判断 表名种类：问题表、分析表、命令表 第三步：通过若依方法，获得Excel 表格中数据 插入数据库。 |
| 调用方法 |  |
| 返回结果 | AjaxResult |

## 数据表自填数据表数据删除

方法名:deleteAllTable

路径：/sql/DefinitionProblemController/deleteAllTable

调用几张表的 Mapper.xml 文件的 sql 文件

执行 删除表中所有数据

delete from 表名