# 版本变更

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间 | 原因 | 概述 | 变更人 | 相关文件 |
| 2018-9-29 | 初始化 | 初始化登录认证接口文档 | elvin |  |
|  |  |  |  |  |

# 符号,名词使用规则

|  |  |  |
| --- | --- | --- |
| 符号 | 意义 | 示例 |
| [ ] | 中括号,内部内容为对象 | 根据[user]是否存在判断登录认证结果 |
| ( ) | 小括号,用来修饰参数,例如:当不同功能参数名一样时,可用其修饰 | (userInfo)key  (userGroup)key |
| 🡪 | 执行下一步 | If(1>0) 🡪 return false; |
| DB | dataBase,文档中专指物理内存数据库 |  |
| >> | 过程中调用方法返回的中间或最终参数 | 根据入参username , password查询DB >> [user] |
| !=,==,equals  &&, || 等基本运算符号及逻辑与句 | 与java语法含义相同 |  |
| MIP | MethodInitialParam,方法初始化参数,指在流程开始时,外部调用传入的参数,在流程开始时用MIP声明 | MIP >> [url]  在认证时,url是浏览器通过httpServletRequest传入的参数,在撰写相关方法流程时,在流程开始时第(0)步声明即可 |
| return | 流程方法向调用方返回结果 |  |

# 全局请求&响应规范

## 1请求

|  |  |
| --- | --- |
| 字段 | 备注 |
| requestHead | 请求头(json形式)  applicationName : 请求系统名称  operationDatetime : 请求操作时间  operationLoginName : 请求操作人登录名  operationName : 请求操作人姓名(昵称)  operationId : 请求操作人Id |
| info | 请求参数(jsonObject形式),具体由业务决定 |

## 2响应

|  |  |
| --- | --- |
| 字段 | 备注 |
| responseHead | 返回响应头(json形式)  applicationName : 响应系统名称  operationDatetime : 响应时间  requestHead : 包装请求头(json形式,规范同请求一样) |
| code | 系统内响应状态码  “S”: 成功  “F”: 失败 |
| msg | 系统内响应信息 |
| errorNum | 系统内响应错误码 |
| business | 第三方业务响应&系统内响应数据对象(json形式)  code : 第三方业务响应状态码  “S”: 成功  “F”: 失败  errorNum : 第三方业务响应错误码  msg : 第三方业务响应信息  dataList : 第三方业务响应&系统内响应数据集合 |

# 登录认证授权

## 1登录

### 1.1登录参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型(长度) | 必填性 | 页面形式 | 备注 |
| username | String(32) | 必填 | input | 用户名 |
| password | String(32) | 必填 | input | 密码 |
| strategy | Integer(1) | 必填 | redio | 登录策略  1 : 缓存登录信息一周  0 : 缓存登录信息5分钟 |

### 1.2登陆过程

#### step1 : 对比DB数据

1. MIP >> [username],[ password]
2. 根据入参[username],[password]查询DB数据 >> [user],   
   if(user == null) 🡪 return 用户名密码错误;   
   if(user != null ) 🡪 进入step2;

#### step2 : 缓存用户信息+cookie

1. 使用工具自动生成全局唯一的字符串作为(userInfoCache) key, 将用户信息 [user]作为value缓存到缓存服务器;
2. 以userId作为参数,查询DB中该用户的数据隔离组信息,返回对象[userList]; 使用工具自动生成全局唯一的字符串作为(userGroupInfo)key,该用户的数据隔离组信息[userList] 作为value缓存到缓存服务器;
3. 构造(userInfo)cookie,以(userInfoCache)key作为cookie的value,以”ELOG\_KEY”作为cookie的key,将cookie写回httpServletResponse;
4. 构造(userGroupInfo)cookie,以(userGroupInfo)key作为cookie的value,以”ELOG\_GROUP\_KEY”作为cookie的key,将cookie写回httpServletResponse;

### 1.3登录结果

根据全局响应规范返回即可;

## 2认证

### 2.1认证参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型(长度) | 必填性 | 页面形式 | 备注 |
| cookieValue | string | 必填 | httpServletRequest | 请求中用户cookie值 |

### 2.2认证过程

#### step1 : 根据cookieValue 获取缓存中用户信息

1. MIP >> [cookieValue]
2. 以[cookieValue]为key,查询缓存中用户信息 >> [userInfo]   
   if([userInfo] ==null ) 🡪 return true  
   else 🡪 return false;

### 2.3认证结果

返回boolean结果即可;

## 3授权

### 3.1授权参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型(长度) | 必填性 | 页面形式 | 备注 |
| url | String | 必填 | httpServletRequest | 请求url |
| cookieValue | string | 必填 | httpServletRequest | 请求中用户cookie值 |

### 3.2授权过程

#### step1 : 对比全局权限url列表

1. MIP >> [url], [cookieValue]
2. 从缓存中获取全局权限url列表,key : “ELOG\_GLOBAL\_USER\_MODULE\_KEY:” >> [globalModuleList]   
   if(null == [globalModuleList] ) 🡪 查询DB,获取全局权限url列表 >> [globalModuleList]以”ELOG\_GLOBAL\_USER\_MODULE\_KEY:”作为key,将[globalModuleList]缓存到cache服务器;
3. If([globalModuleList].contains( [url] )) 🡪 进入step2;  
   else 🡪 放行;

#### step2 : 对比用户权限url列表

1. 从缓存中获取用户权限列表,key 值来源 : cookieValue + “ELOG\_USER\_MODULE”; >> [userModuleList]  
   if (null == [userModuleList] ) 🡪 查询缓存中用户信息 >> [userInfo] 🡪  
   [userInfo].userId 作为条件,查询DB,获取用户权限列表 >> [userModuleList]  
   以cookieValue + “ELOG\_USER\_MODULE”作为key,缓存[userModuleList]  
   到cache服务器;
2. If([userModuleList] .contains([url])) 🡪 放行  
   else 🡪 拦截

### 3.3授权结果

返回boolean结果即可;