告警管理（FM）复用文档

组 号： 0 0 1

撰 写 人： 吴 逸 菲

完成日期: 2015年04月29日

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **构件名称** | 告警管理（FM） | | | | | | | | |
| **功能描述** | 1. 接收应用程序的告警信息 | | | | | | | | |
| 2.输出告警信息到一个单独的告警文件 | | | | | | | | |
| **语言描述** | Java | | | | | | | | |
| **环境描述** | JVM | | | | | | | | |
| **详细说明** | | | | | | | | | |
| **类名称** | 功能说明 | | 输入 | | | 输出 | | 过程 | 说明 |
| **ErrorUtil** | 从根目录下error\_list文件中读取告警信息个数及其具体内容，并且将告警信息及其请求时间录入日志文件，及功能描述中所述单独的告警文件 | | 1. 告警信息个数(error\_list中配置) 2. 告警信息内容(error\_list中配置) | | | 1. 请求时间 2. 告警信息内容   皆输出到error.log文件中，即日志 | | 从error\_list中读取告警信息，然后输出到告警日志中 | 无 |
| **源码** | **public** **class** ErrorUtil {  **private** String error\_list\_path = "error\_list";  **private** String log\_file\_path = "error.log";  **private** Map< Integer, String > error\_summary = **new** HashMap< Integer, String >();    **public** ErrorUtil() {  FileInputStream fis = **null**;  **try** {  fis = **new** FileInputStream(error\_list\_path);  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  System.***out***.println("error list not exist");  e.printStackTrace();  }  InputStreamReader isr = **new** InputStreamReader(fis);  BufferedReader bd = **new** BufferedReader(isr);  **int** num = 0;  **try** {  num = bd.read();  bd.readLine();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  System.***out***.println("can not read the error list");  e.printStackTrace();  }    **for**(**int** i=1; i<=num; i++){  String summary = **new** String();  **try** {  summary = bd.readLine();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  System.***out***.println("can not read the error list");  e.printStackTrace();  }  error\_summary.put(i,summary);  }  **try** {  bd.close();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  System.***out***.println("no file are opened");  e.printStackTrace();  }  }    @SuppressWarnings("resource")  **public** **boolean** report\_an\_error(Integer error\_id) {  **if**(error\_id > error\_summary.size())  **return** **false**;  FileOutputStream fos = **null**;  **try** {  fos = **new** FileOutputStream(log\_file\_path, **true**);  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  System.***out***.println("can not open the log file");  e.printStackTrace();  }  OutputStreamWriter osw = **new** OutputStreamWriter(fos);  BufferedWriter bw = **new** BufferedWriter(osw);  **try** {  bw.write(**new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(**new** Date())+" "+error\_summary.get(error\_id));  bw.newLine();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  System.***out***.println("failed to write the log");  e.printStackTrace();  }  **try** {  bw.close();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  **return** **true**;  }  } | | | | | | | | |
| **ErrorUtilTest** | 同ErrorUtilTest | 同ErrorUtilTest | | 同ErrorUtilTest | 利用ErrorUtil模块中report\_an\_error进行动态测试 | | 该模块为测试模块，结合JUnit进行测试 | | |
| **源代码** | **public** **class** ErrorUtilTest {  **private** ErrorUtil EU = **new** ErrorUtil();  @Before  **public** **void** setUp() **throws** Exception {  }  @After  **public** **void** tearDown() **throws** Exception {  }  @Test  **public** **void** testReport\_an\_error() {  *assertEquals*(**true**, EU.report\_an\_error(1));  *assertEquals*(**true**, EU.report\_an\_error(2));  *assertEquals*(**true**, EU.report\_an\_error(3));  *assertEquals*(**true**, EU.report\_an\_error(4));  }  } | | | | | | | | |