软件复用课程设计

Client-Server 应用程序

复用文档

CM构件

小组： Team10

成员： 谢志杰 1352975

计鹏玥 1352914

王思尧 1352896

谢明玥 1352937

徐锦程 1353012

[1． 构件简介 1](#_Toc448329172)

[2． 接口说明 1](#_Toc448329173)

[2.1 GetConfiguration.java 1](#_Toc448329174)

[(1) GetConfiguration() 1](#_Toc448329175)

[(2) ReadJSONFile(String Path) 1](#_Toc448329176)

[(3) writeJSONFile(String path,String[] key,String[] value) 2](#_Toc448329177)

[(4) getStringByKey(String key) 2](#_Toc448329178)

[(5) getIntByKey(String key) 3](#_Toc448329179)

[(6) load() 3](#_Toc448329180)

[(7) loadData() 4](#_Toc448329181)

[(8) getSERVER\_IP() 4](#_Toc448329182)

[(9) getSERVER\_PORT() 5](#_Toc448329183)

[(10) getMAX\_MESSAGE\_PER\_SECOND() 5](#_Toc448329184)

[(11) getMAX\_MESSAGE\_FOR\_TOTAL() 5](#_Toc448329185)

[(12) getDBUSER() 6](#_Toc448329186)

[(13) getDBPW() 6](#_Toc448329187)

[3．Demo 7](#_Toc448329188)

# 构件简介

|  |  |
| --- | --- |
|  | **获得配置信息 － GetConfiguration** |
| 构件介绍 | 这一构件的主要功能是通过读写文件来对配置信息进行操作，为服务端与客户端的连接、消息发送限制及数据库中存储用户名和密码提供相关信息。 |
| 功能说明 | 用户可以使用该构件实现对配置文件的读写。 |

# 接口说明

## 2.1 GetConfiguration.java

### (1) GetConfiguration()

用于构造GetConfiguration对象，无参数与返回值。

**作用描述：**

实例化获取到的配置文件的对象。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有输入参数和返回值，形如：

GetConfiguration getConfig = new GetConfiguration();

Private String SERVER\_IP = getConfig.getSERVER\_IP();

**注意事项：**

此方法是一个构造函数，在调用该方法前必须先有一个非空的配置文件。

### (2) ReadJSONFile(String Path)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| Path | String | 配置文件的路径 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| laststr | String | 配置文件中的内容 |

**作用描述：**

该方法读取配置文件，将其内容以字符串的形式返回。

**使用方法：**

传入一个配置文件的路径，返回该配置文件的内容，形如：

Sring content = config.ReadJSONFile(“D:\\”);

**注意事项：**

必须先有一个实际存在的对应路径的配置文件。

### (3) writeJSONFile(String path,String[] key,String[] value)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| path | String | 准备写入的文件所在的路径 |
| key | String[] | 键值 |
| value | String[] | 键值所对应的值 |

**作用描述：**

该方法用于将指定的配置信息写入指定路径下的配置文件中。

**使用方法：**

传入文件路径及将要写入的string数组中的配置信息，形如：

getConfig.writeJSONFile(“D:\\”, key, value);

**注意事项：**

写入配置信息的配置文件的路径必须有效，且key与value需要先进行初始化。

### (4) getStringByKey(String key)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | String | 键值 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| value | String | 键值所对应的值 |

**作用描述：**

为配置管理提供查询接口，根据key值查询对应的value，返回值为字符串类型。

**使用方法：**

对于实例化的JSON对象，可以直接调用该方法，输入key值，可以获取到对应的value值，该value值是字符串类型，形如：

String server\_ip = GetConfig.getStringByKey(“SERVER\_IP”);

### (5) getIntByKey(String key)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | String | 键值 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| value | Int | 键值所对应的值 |

**作用描述：**

为配置管理提供查询接口，根据key值查询对应的value，返回值为整型。

**使用方法：**

对于实例化的JSON对象，可以直接调用该方法，输入key值，可以获取到对应的value值，该value值可以是字符串类型或是整型，形如：

Int max\_msg\_per\_sec = GetConfig.getIntByKey(“MAX\_MESSAGE\_PER\_SECOND”);

### (6) load()

动态加载配置信息。

**作用描述：**

动态加载配置信息。

**使用方法：**

通过ReadJSONFile()方法，将读取到的配置文件内容存储到一个JSON对象当中，并可对mutable的JSON对象进行修改，形如：

getConfig.load();

**注意事项：**

1. 只有具有可更改权限的内容才可进行修改，immutable的内容只是可读的。

可更改的信息分别为MAX\_MESSAGE\_PER\_SECOND和MAX\_MESSAGE\_FOR\_TOTAL。

### (7) loadData()

每30秒动态加载一次配置信息。

**作用描述：**

每30秒动态加载一次配置信息。

**使用方法：**

创建计时器，每30秒就在LoadFileTimerTask类中进行一次load操作，形如：

getConfig.loadData();

### (8) getSERVER\_IP()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| SERVER\_IP | String | 服务器的IP地址 |

**作用描述：**

获取服务器的IP地址。

**使用方法：**

对于一个实例化的GetConfiguration对象，可以直接调用该方法，该方法没有输入参数，返回值为string类型的服务器IP地址，形如：

String ip = getConfig.getSERVER\_IP();

**注意事项：**

SERVER\_IP需要先进行初始化。

### (9) getSERVER\_PORT()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| SERVER\_PORT | String | 服务器的端口号 |

**作用描述：**

获取服务器的端口号。

**使用方法：**

对于一个实例化的GetConfiguration对象，可以直接调用该方法，该方法没有输入参数，返回值为string类型的服务器端口号，形如：

String port= getConfig.getSERVER\_PORT();

**注意事项：**

SERVER\_PORT需要先进行初始化。

### (10) getMAX\_MESSAGE\_PER\_SECOND()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| MAX\_MESSAGE\_PER\_SECOND | Int | 每分钟可收发的最大消息数 |

**作用描述：**

获取每分钟可以发送的最大消息数。

**使用方法：**

对于一个实例化的GetConfiguration对象，可以直接调用该方法，该方法没有输入参数，返回值为int类型的MAX\_MESSAGE\_PER\_SECOND，形如：

int max\_msg\_per\_second = getConfig.getMAX\_MESSAGE\_PER\_SECOND();

**注意事项：**

MAX\_MESSAGE\_PER\_SECOND需要先进行初始化。

### (11) getMAX\_MESSAGE\_FOR\_TOTAL()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| MAX\_MESSAGE\_FOR\_TOTAL | Int | 可进行收发的最大消息总数 |

**作用描述：**

获取可以进行收发的最大消息数。

**使用方法：**

对于一个实例化的GetConfiguration对象，可以直接调用该方法，该方法没有输入参数，返回值为int类型的MAX\_MESSAGE\_FOR\_TOTAL，形如：

int max\_msg\_for\_total= getConfig.getMAX\_MESSAGE\_FOR\_TOTAL();

**注意事项：**

MAX\_MESSAGE\_FOR\_TOTAL需要先进行初始化。

### (12) getDBUSER()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| DBUSER | String | 数据库名称 |

**作用描述：**

获取数据库名称。

**使用方法：**

对于一个实例化的GetConfiguration对象，可以直接调用该方法，该方法没有输入参数，返回值为string类型的数据库名称，形如：

string DB\_User = getConfig.getDBUSER();

**注意事项：**

DBUSERD需要先进行初始化。

### (13) getDBPW()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| DBPW | String | 数据库密码 |

**作用描述：**

获取数据库名称。

**使用方法：**

对于一个实例化的GetConfiguration对象，可以直接调用该方法，该方法没有输入参数，返回值为string类型的数据库密码，形如：

string DB\_PW = getConfig.getDBPW();

**注意事项：**

DBPW需要先进行初始化。

# 3．Demo

public class getConfigurationDemo extends ServerSocket{  
  
  
 public getConfigurationDemo(int SERVER\_PORT) throws IOException {  
 super(SERVER\_PORT);  
 // *TODO Auto-generated constructor stub* }  
  
 public void run() throws IOException{  
 try {  
 while(true){//监听客户端请求，启动线程处理  
 Socket socket = accept();  
 new ServerThread(socket);  
 }  
 }catch (Exception e) {  
 }finally{  
 close();  
 }  
 }  
  
 public class ServerThread extends Thread{  
 //将从Json获取配置信息  
 private int MAX\_MESSAGE\_PER\_SECOND;  
 private int MAX\_MESSAGE\_FOR\_TOTAL;  
 public GetConfiguration getConfiguration;  
 ServerThread(Socket socket){  
 getConfiguration = new GetConfiguration();  
 //动态配置，配置文件信息更改时将自动读入  
 getConfiguration.loadData();  
 //打印动态配置信息  
 System.out.println("MAX\_MESSAGE\_PER\_SECOND:"+getConfiguration.getMAX\_MESSAGE\_PER\_SECOND());  
 System.out.println("MAX\_MESSAGE\_FOR\_TOTAL:"+getConfiguration.getMAX\_MESSAGE\_FOR\_TOTAL());  
  
 }  
  
 @Override  
 public void run() {  
 //执行线程run方法  
 }  
 }  
  
}