==========J-SUtil 使用手册==========

此 Jar 文件提供的功能有：日志、数据库 DAO、日期操作。

1. 将 jsUtil-1.x.x.jar 文件放入 web 应用的 classPath 中，一般在 XX/WEB-INF/lib 下。
2. 在 web.xml 文件加入如下配置：

```
<servlet>
  <servlet-name>SysInit</servlet-name>
  <servlet-class>com.utility.init.WebInitUtil</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

3. 将 log4j.properties 和 SysSetings.properties 文件分别拷贝到 web 应用到 XX/WEB-INF/下，
   并根据项目的实际情况，在 SysSetings.properties 文件中记入数据库的配置信息。
   <<特别注意>>
   本 jar 文件所依赖的第三方库如下：
   （1）、log4j-1.2.17.jar (log4j 必须)
   （2）、mysql-connector-java-3.1.12-bin.jar (Mysql5.0+ 数据库时 必须)
   （3）、sqljdbc4.jar (SQL server2008+ 数据库时 必须)
   （4）、classes12.jar (Oracle10g+ 数据库时 必须)
   （5）、c3p0-0.9.5.jar、mchange-commons-java-0.2.9.jar (启用链接池时 必须)

以上配置完了后，基本使用方法如下：

1. 日志

在类中首先定义如下：

```
private static LoggerUtil log = new LoggerUtil(XX.class);
```

输出错误日志：log.error(e); //可直接传入 Exception 对象也可传入 String 对象

输出一般日志：log.info([message]);

2. 数据库-DAO 层引入

在你的 DAO 层的类需要先继承 【extends MysqlBaseDao】类，（如果是 SQL Server 数据库则继承 extends SqlServerBaseDao，Oracle 数据库则继承 extends OracleBaseDao）。

此时在类的方法中就能对数据库进行操作了（即 有了 dbu 对象）。例如：

1）查询

```
this.dbu.prepareSQL("select * from table01 where CurrentTime>=? and CurrentTime<=?");
this.dbu.setParameter(1, "2015-01-28 00:00:00");
this.dbu.setParameter(2, "2015-01-28 23:59:59");
List<Map<Object, Object>>  rs = this.dbu.execQuery();
获取值
rs.get(0).get("[数据表的字段名称]")
```

2）插入

```
this.dbu.beginTransaction();//需要开启事物时用
this.dbu.prepareSQL("insert into t_historyquerydetail_day(uuid,Ecode,Ename) values(?,?,?)");
this.dbu.setParameter(1, "123344");
this.dbu.setParameter(2, "t001");
this.dbu.setParameter(3, "张三");
this.dbu.execSQL();
this.dbu.commit();//需要开启事物时用
```

3. 日期操作。

对于日期的操作 DateUtil 类为你提供了各种常用的方法。

==========完了==========

轻松几步你的项目就同时有了“日志”和“便捷的数据库 DAO”了。祝你好运！

4. **查询 Record 与 Bean 对象的映射**

如果希望将查询的记录直接映射成对象话，可以通过下记方法简单实现：

① 建立与数据库表(table)、查询（view）对应的 JavaBean 类，并给类和属性的 set 方法加批注。

```
@TableAnnotation(targetTable = "t_menu")
public class MenuBean {
    private String uuid;
    private String name;
    private String icon;
    private String url;
    private int showIdx;
    private String type;
    private String enable;
    public String getUuid() {
        return uuid;
    }
    @BeanAnnotation(targetField="uuid")
    public void setUuid(String uuid) {
        this.uuid = uuid;
    }
    public String getName() {
        return name;
    }
    @BeanAnnotation(targetField="name")
    public void setName(String name) {
        this.name = name;
    }
    public String getIcon() {
        return icon;
    }
    @BeanAnnotation(targetField="icon")
    public void setIcon(String icon) {
        this.icon = icon;
    }
    public String getUrl() {
        return url;
```

对应的数据库表名称或查询名称

表、查询的“字段名称或查询的别名

② 在 DAO 层的只需如下调用即可。

```
public List<MenuBean> findMenu(MenuBean menu,int pageNum){
    List<MenuBean> rs = null;
    try {
        StringBuffer buff = new StringBuffer("select t.* from (select * from t_menu where 1=1");
        if(menu!=null&&menu.getName()!=null&&menu.getName().trim().length()>0){
            buff.append(" and name like '%"+menu.getName()+"%'");
        }
        buff.append(" order by showIdx asc,name asc) t ");
        dbu.setPageNumber(pageNum);
        dbu.setPageSize(Constant.pageSize);
        dbu.prepareSQL(buff.toString());
        rs = (List<MenuBean>)dbu.execQuery(MenuBean.class);
    } catch (Exception e) {
        e.printStackTrace();
        log.error(e);// 打印日志
        throw new SystemException(e);
    }
    return rs;
}
```

③ 在存储过程调用。

```
public List<User> getUsers3(String name) throws SystemException {
    List<User> rs = null;
    try {
        Map<String, Object> input = new HashMap<String, Object>();
        input.put("name", name);
        rs = (List<User>)dbu.prepareCall(User.class,"FindUsers", input);
        return rs;
    } catch (Exception e) {
        e.printStackTrace();
        log.error(e);// 打印日志
        throw new SystemException(e);
    }
}
```

5. 根据表或查询（view）的主键取得唯一 record

事前需要先通过 Map 的形式存入各个主键字段以及对应的值。.

① 返回 Bean 对象数据

```
public MenuBean queryMenu(String uuid) throws SystemException{
    Map<String,Object> key=new HashMap<String,Object>();
    key.put("uuid", uuid);
    return (MenuBean)dbu.getDataByPrimaryKey(MenuBean.class, key);

}
```

② 返回 Map 数据

```
public Map<Object,Object> queryMenu(String uuid) throws SystemException{
    Map<String,Object> key=new HashMap<String,Object>();
    key.put("uuid", uuid);
    return dbu.getDataByPrimaryKey("t_menu",key);

}
```

6. c3p0 链接池

① 将 config/c3p0.properties 拷贝到你的 web 应用的 classPath 中，一般为：XX\WEB-INF\classes 目录下。

② 根据项目实际情况修改 c3p0.properties 文件。

③ 打开 XX\WEB-INF\SysSetings.properties 文件将对应数据库的“连接池开启”（即设为 true）

注：如果你采用连接池的方式，SysSetings.properties 文件中的数据库连接参数可以不配置。

④ 将 Dependent-libs 下的 c3p0-plugin-1.0.0.jar、c3p0-0.9.5.jar 和 mchange-commons-java-0.2.9.jar 放到项目的 lib 目录下。

⑤ 再次启动 Tomcat

通过以上几步，你的项目就可以使用连接池啦

7. 分页查询

```
dbu.setPageNumber(pageNum);//页码
dbu.setPageSize(Constant.pageSize);//页大小（即，每页显示的记录数）
dbu.prepareSQL(sql.toString());
rs = (List<MenuBean>)dbu.execQuery(MenuBean.class);
```

注：分页以上两个方法需要在 dbu.prepareSQL()方法之前调用。