**DataWindowHTTP V1.00产品说明书**

**<http://datawindow.ltd/> <http://powerbuilder.ltd/> <http://mis2erp.com/>**

 **QQ: 27-3939-617/Mob: 137-9029-9411**

**Mail: chengang0769@21cn.com/273939617@qq.com**

**2021-10-06**

目录

* **[历史由来](#历史由来)**
* **[产品定位](#产品定位)**
* **[产品特色](#产品特色)**
* **[开发语言](#开发语言)**
* **[适用版本](#适用版本)**
* **[授权方式](#授权方式)**
* **[使用建议](#使用建议)**
* **[后续改进](#后续改进)**
* **[编程实例](#编程实例)**
* **[在线演示](#在线演示)**
* **[接口说明](#接口说明)**

**历史由来**

PB的优点就是连接数据库简单，设计方便。其datawindow控件集设计，展示，报表，存储，导入，导出于一体。本设计在datawindow的基础上做加强设计，将其连接数据库的功能，从传统局域网延展到互联网。

PB连接数据库的技术很传统，很直接，我们习惯称之为： 直连数据库。直连数据库是许多开发工具连接数据库的模式，是最直接的CS模式没有N层。虽然要借助于数据库驱动，但本质仍然是一种server/client模式。我们通常只管连接，并不在意如何连接。是一种简单可靠的数据访问模式，常用于中小系统设计，正好契合企业管理系统的开发要求。这是PB的优点。

但PB写的程序，在局域网使用还算不错。不过也仅限于PC机+网线模式。如果是用笔记本电脑和WIFI接入，或者是使用外网的服务器和数据库时，会经常遇到网络中断：PPPOE上网产生的中断，电脑睡眠产生的中断，WIFI掉线产生的中断，移动笔记本带来的中断……从而导致程序无法提交或者崩溃退出。直接的后果是，我们无法向客户解释，它看起来像软件BUG，而且是频繁发生的。这是PB的缺点。

2017年开发一个广东肇庆的项目，不想直连数据库，因为：断线让程序显得很崩溃，操作员频繁报故障，设计重连也非常不完美，重连失败或者等待时间太长。突发奇想，想采用网络接口访问数据库。花了两到三个月时间，架构了DataWindowHTTP（曾用名DataWindowJSON）第一个版本。几年前有做过PB与通讯计费系统通过JSON数据交互方面的运用，积累了PB成功使用互联网接口方面的一点点经验。这是DataWindowHTTP的可行性。

在某些PB版本中，可以设计基于webservice的dw，这个模式的优点是可以方便地调用web接口了。缺点是：它只是针对部分对外服务而设计的。并不能采用这个模式来设计所有dw，因为那样需要无数个接口。而DataWindowHTTP是设计成类似“桥（brige）”或“路由（router）”的功能，只需要一个“公共接口”，就可以让所有dw访问互联网数据库。对于如今的管理系统而言，连接数据库变得复杂了，这个变化是因为互联网这样的异构造成的：服务器（数量增多，机房增多，局内局外都有），数据库（数据库类型变多，各种用途的库增多）和办公地点（省，市分公司模式，镇区分支，连锁零售点等），这些是分散的，他们的互通，都是通过互联网这个纽带。在整个公司范畴来看，数据是整体和统一的。所以：“PB要以较小成本继续高效率开发的话，它需要DataWindowHTTP。DataWindowHTTP让PB轻松连接到基于互联网的服务器，而不需要在接口上增加一分钱成本。”。DataWindowHTTP采用非持久连接，让它可以轻松连接到互联网数据库。DataWindowHTTP就是基于这样的设计目的。这就是DataWindowHTTP的技术思路。

**产品定位**

基于Datawindow的基本功能： 设计，展示，增删改查等丰富特性，在其上嫁接互联网接口发送和接收功能。采用DatawindowHTTP+JSON封装数据+AES动态密码+RSA加密AES的动态密码+AES对数据加密+服务器接口+PHP8.0+APACHE2.4+数据库的模式。这样PB开发的程序不再需要安装数据库接口。这个访问链主要分三段：

1. DatawindowHTTP+JSON封装数据+AES动态密码+RSA加密AES的动态密码+AES数据加密： 全部封装到一个控件中。开发者可以无视它们的复杂内部构造，它已经封装成组件了。
2. PHP接口 + PHP8.0 + APACHE2.4：这是固定的接口，只需要简单安装配置，它等于一个“桥（brige）”或“路由(router)”，把DW的数据提交到数据库。
3. 数据库。目前经过测试的数据库有：

MSSQL2000，2005，2008，2012，2014，2017

Oracle10g，11g，12c

Mysql5.7，8.0

MariaDB10.5

PostgreSQL12.6

Sybase16.0（待测试）

Db2V11.1（待测试）

**产品特色**

1. 轻松连接互联网云主机的数据库。
2. 做减法设计，为简化而设计。
3. 实现互联网与局域网有机统一。
4. 无需改变设计的习惯。设计阶段采用传统直连模式，而运行阶段采用互联网接口模式。
5. 程序发行时无需数据库驱动，无需管理员远程安装和配置。
6. 服务器带负载能力强，负载容易均衡，数据库压力小。
7. 同时连接多个局域网/互联网数据库时很方便。
8. 实现了程序的真正移动运用，用户可以随时随地在笔记本上面漫游使用，一笔数据即使延迟几天也可以成功提交。
9. 使用AES加密上传下载的数据，安全性有了保证。
10. 采用久经验证的经典配置：Apache+PHP来驱动，对服务器的硬件配置要求低。

开发语言

组件代码采用PB代码实现，确保可以移植到多个版本。

适用版本

初始版本于2017年采用PB12.5开发，并基于它开发了ERP。鉴于它的实用性，于2021年整理和简化接口，并成功移植到PB9,10.0,10.5,11.0,12.5,2019。这个组件支持≥9.0的所有版本。dwhttp.pbd开发组件将以PBD作为发行组件，提供给开发者。dwhttp.pbd将不会提供PB源代码。

授权方式

开发者在购买后是授权【公司名称+联系电话】或者【软件工作室名称+联系电话】。这个信息会在组件产生提示时显示出来。

这个授权不限制开发者使用的电脑数量，最终开发的软件系统也不受限制（即不限制安装的服务器台数，也不限制连接的客户数量，也不限制最终产品可使用的时间）。

使用建议

本组件特别适用于基于互联网云主机的数据库管理系统。它彻底解决直连中产生的种种断线问题。提供了极具特色的驱动模式。

首先，它适合新的企业管理系统开发。

组件没有依赖其他框架，功能很单纯，您完全可以将其融合进界面框架，可以融合进业务架构，可以融合进报表系统，从而构建复杂的系统。

其次，它适合老项目的改造。

您可能觉得对于现有系统的整体改造很困难。但有折中的办法，可以把一些频繁使用的主要操作界面用本组件进行改造，极少使用的界面可以不用改造，不改造的界面仍然使用SQLCA连接。我曾经统计了一些项目，发现虽然界面很多，但是极其频繁使用的界面非常少，通常不超过5-10%。比如库存系统中，入库单出库单使用的频率极高；调整单和盘点单使用频率就极低。用这个组件改造，必然能起到立竿见影的效果。

改造一个简单的界面：

第一步：将dw\_1,dw\_2等控件的祖先由datawindow改为datawindow\_http；

第二步：将settransobject(sqlca)改为settranshttp(serverinfo)；

第三步：将retrieve(parm1,parm2...)改为retrievehttp(parm1,parm2...)；

第四步：将update()改为updatehttp()；

后续改进

后续版本，将完善“基于消息队列的分布式事务”和“基于XA协议的两阶段提交分布式事务”的例子。

* 目前虽然可以将多个SQL动态语句或者多个dw语句包在一个事务里面执行。但没法把中间结果放在一个事务中。
* “基于消息队列的分布式事务”是最实用的分布式事务。目前组件在技术上是支持的，只是缺少一个实际的例子程序。
* “基于XA协议的两阶段提交分布式事务”是Tuxedo提出，多数商业数据库实现了XA，目前组件在技术上是支持的，只是缺少一个实际的例子程序。

如有需要可提供一个Linux+Mysql+Apache2.4+PHP8.0的安装文档。

编程实例

Code\ : 演示代码源码（为减少文件体积，压缩包只含有12.5版本，如需其他版本请联系）

Test\ : 已编译直接运行的测试程序（为减少文件体积，压缩包只含有12.5版本，如需其他版本请联系）

Server\apache-php-driver: apache，php，sqlsrv等软件和数据库驱动。

Server\db：数据库

Server\interface: dwhttp服务器接口PHP代码

在线演示

提供一台服务器用于在线演示。这样大家无需搭建环境而快速测试。请运行Test目录的程序。

有兴趣深度测试的，可以加QQ索取自己需要的PB版本的例子程序，按文档安装服务器，运行程序例子。

接口说明

四个组件：

datastore\_http，datawindow\_http，datawindowchild\_http，embeddedsql\_http

1. datastore\_http

用于不可视的DataStore的检索

1.1 public subroutine settranshttp (str\_serverinfo serverinfo)

作用：

初始化接口信息

用例：

str\_serverinfo gstr\_serverinfo

embeddedsql\_http gds\_sql

//服务器，端口，目录，主页面

gstr\_serverinfo.server = “\*\*\*.\*\*\*.\*\*\*” //主页网址或者IP

gstr\_serverinfo.port = 8080 //端口

gstr\_serverinfo.path = "DataWindowHTTP” //子目录

gstr\_serverinfo.page = "query.php" //接口主页面

//使用接口的用户名和密码

gstr\_serverinfo.userid = "\*\*\*"

gstr\_serverinfo.pwd = "\*\*\*"

//指定rsa公钥，两种方式：

//1. public.pem以文件形式存在，指定：路径+文件名，方便定期更换公钥。

gstr\_serverinfo.rsapubkey = curpath + "\public.pem"

//2.用字符串把pubkey写下来,这样可以不放到文件夹里面, 也可以动态拼接，相对更安全，但是不好定期更换公钥。

gstr\_serverinfo.rsapubkey = "-----BEGIN PUBLIC KEY-----~n"+&

"MIGfMA0GCSqESIb3EQECAQUAA4GNADCBiQKBgQDkfZ9VABDgVktkss3njg7sVdjY~n"+&

"8mP4mrUVUBAJlcIGtzFg6rjRcjaR2N/KXtQJ3kU319Nn1fl0SVQLS+0lCFa10gKu~n"+&

"mRMMGAg2WiOoYs4r9fzbkukbkWCpWZxoJNnG18ruVn7dc2tQx6gscfO5ZSh394qT~n"+&

"b9RmOfgmX7qgknj+SQIDAQAB~n"+&

"-----END PUBLIC KEY-----~n"

//server的信息赋给控件并初始化, 等同于dw.settransobect()

gds\_sql.settranshttp(gstr\_serverinfo)

1.2 public subroutine retrievehttp ()

作用:

无参数检索

用例:

gds\_fk.retrievehttp()

1.3 public function long retrievehttp (any pvalue1, ..., any pvalueN)

作用:

带参数检索

其中，参数个数固定为1-30个，35个，40个，45个，50个。

如果接口恰好需要33个，请采用35个参数调用。没有的参数送入字符串空值或者0皆可。

用例：

//检索，有参数

Long p1

String p2

//与retrieve接口一致。可以送入变量。或者常量，参数类型自动识别。

//但是参数顺序和设计时必须严格一致。

gds\_fk.retrievehttp(p1,p2,”test”)

1.4 public subroutine updatehttp (boolean ab\_accepttext, boolean ab\_resetupdate)

作用：

更新数据

参数：

ab\_accepttext：执行accepttext()

ab\_resetupdate: 执行resetupdate()

属性

public statictext ist\_prompt

设置一个外部statictext，可以实时显示数据检索进度，可以为空则不显示。

public int ret

执行检索后的返回值，判断检索成功以否的标志

public int nrows

执行检索后的行数

1. datawindow\_http

用于可视的DataWindow的检索

2.1 public subroutine settranshttp (str\_serverinfo newserver)

2.2 public subroutine updatehttp (boolean ab\_accepttext, boolean ab\_resetupdate)

2.3 public function long retrievehttp ()

2.4 public function long retrievehttp (any pvalue1,...any pvalueN)

同datastore\_http的同名函数相同，此处略。

2.5 public subroutine bindparm (string pname, any pvalue)

绑定参数，一次绑定一个参数。

Pname：参数名

Pvalue：值

2.6 public subroutine turntopage (long al\_page)

翻页

al\_page：指定页码

2.7 public subroutine updatemerge (ref datawindow\_http adw\_mergedw1,...,ref datawindow\_http adw\_mergedwN,

 boolean ab\_accepttext, boolean ab\_resetupdate)

多个dw合并sql语句，一次性更新（在一个事务里面，确保一致性）。最多支持1-10个dw。

这个不同于传统的多个dw更新逐个提交模式。这个在提交上面是一次的，被包在一个trans中，效率更高。相对来说，事务锁定时间缩短。

属性：

public statictext ist\_prompt

public int ret

public int nrows

同datastore\_http的同名属性

public boolean setredraw = true

设置是否组件取得数据后刷新，有时需要检索后再过滤可以取消，避免不必要的刷新

public long pagerows = 0

分页时，每页的行数

public long pages = 0

分页时，根据检索 nRows/pagerows得到的页数。

1. datawindowchild\_http

用于下拉dddw的检索和composite报表的检索

* 1. public subroutine settranshttp (str\_serverinfo serverinfo)
	2. public function long retrievehttp ()
	3. public function long retrievehttp (any pvalue1, ..., any pvalueN)

同datastore\_http的同名函数

属性

public statictext ist\_prompt

public int ret

public int nrows

同datastore\_http的同名属性

1. embeddedsql\_http

用于内嵌（动态）sql的检索，包括内嵌sql语句和调用存储器。并可将动态语句生成的结果集放置于ds或者dw中。

* 1. public subroutine settranshttp (str\_serverinfo serverinfo)
	2. public subroutine bindparm (string pname, any pvalue)

同datawindow\_http中同名函数

* 1. public subroutine selectinto (string ptype1)
	2. public subroutine selectinto (string ptype1,..., string ptypeN)

模拟执行select into查询

ptype1：接收返回值的变量的类型,如”string”,”long”。和系统变量类型相同

* 1. public subroutine selectintodw (ref datastore ds\_ext)

4.6 public subroutine selectintodw (ref datawindow dw\_ext)

将动态sql语句结果集放入datastore/datawindow中，比如存储器的结果。

ds\_ext/dw\_ext: datastore/datawindow控件名

* 1. public function integer addsql (string sql)

添加动态执行的sql语句。可以多次添加，执行时按添加顺序执行。执行后清零。

4.7 public function integer addsql (string sql,boolean savesql)

添加动态执行的sql语句。可以多次添加，执行时按添加顺序执行。执行后保持语句。

如执行一个循环，可以写在循环外，只需要一次初始化赋值。

Savesql= QSAVESQL/QCLEARSQL

这个开关一旦打开，是针对所有SQL的，直到executesql后仍然保持。而且后续addsql也生效。

4.8 public subroutine executesql ()

执行动态语句，无参数。

4.9 public subroutine executesql (integer ai\_intrans)

执行动态语句

ai\_intrans: 是否将语句包括在一个事务(begin trans...end trans)中

ai\_intrans = QNOTRANS/QTRANS

属性

public statictext ist\_prompt

public int ret

public int nrows

同Datastore\_http中同名属性

public str\_dwparm intovar[]

执行select into模式的查询后，放置into变量取得的结果。

boolean ib\_dontmsg = false

是否显示错误提示消息

有时不要组件的提示，直接根据ret返回值做静默处理