HttpClient 案例设计

# 更新历史

|  |  |  |
| --- | --- | --- |
| 日期 | 更新人 | 更新内容 |
| 2017-09-26 | pangchuguan | Create |
|  |  |  |
|  |  |  |

# 目录

[更新历史 2](#_Toc495998130)

[目录 3](#_Toc495998131)

[1 简介 4](#_Toc495998132)

[2 主要对象 4](#_Toc495998133)

[3 HTTP(S)访问介绍 4](#_Toc495998134)

[3.1.1 HTTP(S)访问信息列表 4](#_Toc495998135)

[4 功能测试 6](#_Toc495998136)

[4.1 HttpClient对象 6](#_Toc495998137)

[4.1.1 HttpClient对象用例（PB系统对象） 6](#_Toc495998138)

[4.1.2 Request Functions 测试 6](#_Toc495998139)

[4.1.3 Response Functions 测试 10](#_Toc495998140)

[4.1.4 Properties 11](#_Toc495998141)

[4.2 RestClient 12](#_Toc495998142)

[4.2.1 RestClient（PB对象测试） 12](#_Toc495998143)

[4.2.2 功能测试 12](#_Toc495998144)

[4.2.3 属性测试 15](#_Toc495998145)

[4.2.4 使用场景 15](#_Toc495998146)

# 

# 简介

为支持PB下对RESTful Web API的信息交互，在appeon powerbuilder下加入HttpClient、RestClient对象用来发送请求到Web API，同时接受Web API返回的信息；

# 主要对象

HttpClient：作为HTTP客户端(类似浏览器)与RESTful Web API服务器进行通讯，发送HTTP请求信息，并获取服务器端响应信息。

RestClient：实现将web api返回的数据输出到datawindow和datastore中方便数据的展示(包括ChildDataWindow)。

# HTTP(S)访问介绍

### HTTP(S)访问信息列表

能发送和获得所有的HTTP报文；

#### 请求信息头列表(.Net支持的略有不同)

Accept:\*/\*

Accept-Encoding:gzip

Accept-Language:zh-CN,zh

Accept-Charset:ISO-8859-1,utf-8;q=0.7,\*;q=0.7

Allowautoredirect:false

Authorization:Basic bXl1c2VyOm15cGFzcw==

Connection:keep-alive

Cache-Control:no-cache

Cookie:user=appeon;pwd=appeon

ClientCertificate:

Content-Type:application/json

Content-Length:43

Date:

Expect:100-continue

From:abc@appeon.com

Host:www.appeon.com

If-Match:

If-Modified-Since:Sat, 28 Nov 2017 06:38:19 GMT

KeepAlive:300

Method:GET/POST/PUT/PATCH/DELETE

Origin:https://www.appeon.com

Postdata:a=1&b=2

Pragma:no-cache

ProtocolVersion:

ProxyIp:

ProxyPwd:

ProxyUserName:

ReadWriteTimeout:5000

Referer:https://www.appeon.com/

ResultType:Byte[]

Timeout:5000

URL:https://www.appeon.com/index.php

User-Agent:Chrome/60.0.3112.113

Upgrade-Insecure-Requests:1

X-Requested-With:XMLHttpRequest

#### 响应信息头列表

Accept-Ranges:

Allow：

Cache-Control：

Connection：

Content-Encoding:

Content-Length：

Content-Security-Policy:

Content-Type:

Date:

Etag:

Expires:

Last-Modified:

Location：

P3P:

Pragma:

Proxy-Authenticate：

Refresh：

Retry-After：

Server:

Set-Cookie:

Strict-Transport-Security:

Transfer-Encoding:

Vary:

Via：

X-Cache-Lookup:

X-Content-Type-Options:

X-Daa-Tunnel:

X-NWS-LOG-UUID:

X-Powered-By：

X-XSS-Protection:

WWW-Authenticate：

#### HTTP状态列表

200 ok - 成功返回状态，对应，GET、PUT、PATCH、DELETE。

201 created - 成功创建。

302 found – 跳转Location。

304 not modified - HTTP缓存有些。

400 bad request - 请求格式错误。

401 unauthorized – 未授权。

403 forbidden – 鉴权成功，但是该用户没有权限。

404 not found – 请求的资源不存在。

405 method not allowed – 该http方法不被允许。

410 gone – 这个url对应的资源现在不可用。

415 unsupported media type – 请求类型错误。

422 unprocessable entity –校验错误时用。

429 too many request – 请求过多。

# 功能测试

## HttpClient对象

HttpClient各个功能函数，测试用例设计

### HttpClient对象用例（PB系统对象）

* 通过classname，获取新加对象的类名
* Typeof、ClassName函数的返回值
* 通过create和destroy，创建和销毁对象
* 手工封装HttpClient对象，自定义常用函数，方便继承调用
* 语法异常测试？
* 组合测试，场景设计？
* 变量范围，放全局变量？重复调用？HttpClient是否加到已有的用户对象列表里？
* 安全验证怎么测试？
* 作为参数传递application、window(事件，函数，message)，UserObject，Functions，structure

### Request Functions 测试

#### ClearRequestHeaders

语法：Integer HttpClient. ClearRequestHeaders ()

* HttpClient 未构建的情况下调用（这个是异常调用，可以在systemerror?中捕捉异常，看是否可以捕捉）
* 重置HttpClient的信息头和响应正文
* 无参数，看函数的返回值，1成功,-1有错误
* 对比ResetRequest前后的Headers值和返回值
* 执行HttpClient.SendRequest()前测试并查看GetRequestHeaders
* 执行HttpClient.SendRequest()后测试并查看GetRequestHeaders

#### SetRequestHeader

语法：Integer HttpClient.SetRequestHeader(string headerName, string headerValue)

* 参数HeaderName 是请求信息头名称(参照3.1.1.1列表)，参数Header Value是请求信息头的值
* 设置HttpClient请求的信息头
* 返回值，1成功,-1有错误，是否会有null？
* 参数为null时，查看返回值
* 参数为空字符串时，查看返回值
* 参数字符数超大时，5000字符以上，查看返回值
* headerValue:Cookie, postdata, ProxyPwd 设计时需要支持多语言
* 版本号放到header的测试？版本号放url的测试

#### GetRequestHeader

语法：String HttpClient.GetRequestHeader(string headerName)

* 参数HeaderName 是请求信息头名称(参考3.1.1.1列表)
* 获取HttpClient指定信息头值
* 返回值，字符串型，指定信息头的值。(返回空字符为失败？)
* 参数为null时，查看返回值
* 参数为空字符串时，查看返回值
* 参数为不存在的headerName时，查看返回值
* 参数字符数超大时，5000字符以上，查看返回值

#### SetRequestHeaders

语法：Integer HttpClient.SetRequestHeaders(string headers)

* 参数Headers 是请求信息头集合(参照3.1.1.1列表)，
* 格式：headerName1:headerValue1~r~nheaderName2:headerValue2
* 批量设置HttpClient请求的信息头
* 返回值，1成功,-1有错误，是否会有null？
* 参数为null时，查看返回值
* 参数为空字符串时，查看返回值
* 参数字符数超大时，5000字符以上，查看返回值

#### GetRequestHeaders

语法：String HttpClient.GetRequestHeader()

* 获取HttpClient信息头集合
* 查看返回值，字符串型，信息头的集合。(返回空字符为失败？)
* 执行HttpClient.SendRequest()前测试并查看返回值
* 执行HttpClient.SendRequest()后测试并查看返回值

#### SendRequest

语法1：Integer HttpClient.SendRequest(string methodName, string urlName)

* 参数methodName 是请求方法(GET/POST/PUT/PATCH/DELETE)，参数urlName是Web API 的URI，包含？后面的参数
* 发送请求到Web 服务器，并响应服务器返回的信息
* 返回值，1成功,-1有错误，-2非法的URL，-3无网络，-4超时，是否会有null？
* 各参数为null时，查看返回值
* 各参数为空字符串时，查看返回值
* 各参数字符数超大时，5000字符以上，查看返回值
* 使用错误的URL，查看返回值 (考虑无网络怎么测试？)
* GET/PUT/PATCH/DELETE方法测试，查看返回值，并查看HttpClient.GetResponseStatusCode()返回值（参照3.1.1.3）
* POST方法测试，需要增加信息头postdata，和”Content-Type:application/x-www-form-urlencoded;”， 查看返回值，并查看HttpClient.GetResponseStatusCode()返回值
* SSL 5个版本协议验证；

语法2：Integer HttpClient.SendRequest(string methodName, string urlName，string data)

* 参数methodName 是请求方法(POST/PUT/PATCH/DELETE)，参数urlName是Web API 的URI，参数data是提交的数据信息，即postdata
* 发送请求到Web 服务器，并响应服务器返回的信息
* 返回值，1成功,-1有错误
* 各参数为null时，查看返回值
* 各参数为空字符串时，查看返回值
* 各参数字符数超大时，5000字符以上，查看返回值
* 使用错误的URL，查看返回值
* POST/PUT/PATCH/DELETE方法测试，需要增加信息头”Content-Type:application/x-www-form-urlencoded;”，查看返回值，并查看HttpClient.GetResponseStatusCode()返回值（参照3.1.1.3）

语法3：Integer HttpClient.SendRequest(string methodName, string urlName，blob data)

* 参数methodName 是请求方法(POST/PUT/PATCH/DELETE)，参数urlName是Web API 的URI，参数data是提交的大文本数据，即postdata （转换成Byte64？）
* 发送请求到Web 服务器，并响应服务器返回的信息,一般用于大文本信息提交
* 返回值，1成功,-1有错误，-2非法的URL，-3无网络，-4超时，是否会有null？
* POST/PUT/PATCH/DELETE方法测试，需要增加信息头”Content-Type:application/x-www-form-urlencoded;”(还有其他?)，查看返回值，并查看HttpClient.GetResponseStatusCode()返回值（参照3.1.1.3）

#### PostDataStart

语法1：Integer HttpClient.PostDataStart(string urlName)

* 参数urlName是Web API 的URI
* 开始发送数据到Web 服务器，并响应服务器返回的信息，主要是POST方法
* 返回值，1成功,-1有错误，-2非法的URL，-3无网络，-4超时，是否会有null？
* 参数为null时，查看返回值
* 参数为空字符串时，查看返回值
* 参数字符数超大时，5000字符以上，查看返回值
* 使用错误的URL，查看返回值 (考虑无网络怎么测试？)
* 需配合PostData、PostDataEnd测试，需要增加信息头” Content-Length:9999999”， 查看返回值

#### PostData

语法1：Integer HttpClient.PostData(string data, long bufferSize)

* 参数data是传输的字符串，参数bufferSize是表明传输字符串的大小
* 持续发送数据到Web 服务器，并响应服务器返回的信息，主要是POST方法
* 返回值，1成功,-1有错误，-2超时，是否会有null？
* 各参数为null时，查看返回值
* 各参数为空字符串时，查看返回值
* 参数data字符数超1024字符以上，查看返回值
* 需配合PostDataStart、PostDataEnd测试，需要增加信息头” Content-Length:9999999”， 查看返回值

语法2：Integer HttpClient.PostData(Blob data, long bufferSize)

* 参数data是传输的大文本数据(实际传输是转换byte？)，参数bufferSize是表明传输数据的大小
* 持续发送数据到Web 服务器，并响应服务器返回的信息，主要是POST方法
* 返回值，1成功,-1有错误，-2超时，是否会有null？
* 各参数为null时，查看返回值
* 各参数为空字符串时，查看返回值
* 参数data数据数超1024字符以上，查看返回值
* 需配合PostDataStart、PostDataEnd测试，需要增加信息头” Content-Length:9999999”， 查看返回值

#### PostDataEnd

语法1：Integer HttpClient.PostDataEnd()

* 结束发送数据到Web 服务器，并响应服务器返回的信息，主要是POST方法
* 返回值，1成功,-1有错误，-2超时，是否会有null？
* 需配合PostDataStart、PostDataEnd测试，需要增加信息头” Content-Length:9999999”， 查看返回值

### Response Functions 测试

#### GetResponseHeader

语法1：Integer HttpClient.GetResponseHeader(string headerName)

* 参数headerName是响应信息头名称(参考3.1.1.2列表)
* 获取服务器响应HttpClient返回的指定信息头值
* 返回值，字符串型，指定信息头的值。(返回空字符为失败？)
* 参数为null时，查看返回值
* 参数为空字符串时，查看返回值
* 参数为不存在的headerName时，查看返回值
* 参数字符数超大时，5000字符以上，查看返回值
* 获取常用信息头Cookie、Response等测试(参考3.1.1.2列表)

#### GetResponseHeaders

语法：String HttpClient.GetResponseHeaders()

* 获取HttpClient与服务器交互后响应的信息头集合
* 查看返回值，字符串型，信息头的集合。(返回空字符为失败？)
* 执行HttpClient.SendRequest()前测试并查看返回值
* 执行HttpClient.SendRequest()后测试并查看返回值

#### GetResponseBody

语法1：Integer HttpClient.GetResponseBody(ref string data)

* 获取HttpClient与服务器交互后响应的正文
* 查看返回值，1成功，-2有错误，-2超时
* 执行HttpClient.SendRequest()后测试并查看返回值及data内容
* 通过修改请求信息头Content-Type，测试返回正文类型：JSON、XML、JEPG 等

语法2：Integer HttpClient.GetResponseBody(ref string data, encoding EncodingType)

* 获取HttpClient与服务器交互后响应的指定编码正文
* 查看返回值，1成功，-2有错误，-2超时
* 执行HttpClient.SendRequest()后测试并查看返回值及data内容
* 通过修改请求信息头Content-Type，测试返回正文类型：JSON、XML、JEPG 等
* 通过修改请求信息头Accept-Encoding？、Accept-Charset，测试返回正文编码：UTF-8、UTF-16、UNICODE等

语法3：Integer HttpClient.GetResponseBody(ref blob data)

* 获取HttpClient与服务器交互后响应的大文本的正文
* 查看返回值，1成功，-2有错误，-2超时
* 执行HttpClient.SendRequest()后测试并查看返回值及data大文本内容

#### ReadData

语法1：Integer HttpClient.ReadData(ref blob data, long bufferSize)

* 获取HttpClient与服务器交互后响应的正文
* 查看返回值，1成功，0读取完成，-2有错误，-2超时
* 执行HttpClient.SendRequest()后，按1024字节持续读取大文本data内容，查看返回值

#### GetResponseStatusCode

语法1：Integer HttpClient.GetResponseStatusCode()

* 获取HttpClient与服务器交互后响应的状态编码
* 查看返回值，（参照3.1.1.3列表 ）
* 执行HttpClient.SendRequest()后，查看GetResponseStatusCode()返回值
* 验证返回值200，201，304,400,401,404等

#### GetResponseStatusText

语法1：Integer HttpClient.GetResponseStatusText()

* 获取HttpClient与服务器交互后响应的状态文本
* 查看返回值，（参照3.1.1.3列表 ）
* 执行HttpClient.SendRequest()后，查看GetResponseStatusText()返回值
* 验证返回值ok, created,not modified,not found 等

### Properties

#### AutoReadData

语法1： HttpClient.AutoReadData = True/false

* 设置HttpClient是否自动读取响应的数据
* True –自动
* False – 不自动

语法2：Boolean lb\_temp = HttpClient.AutoReadData

* 获取当前HttpClient是否自动读取响应的数据
* True –自动
* False – 不自动

#### SecureProtocol

语法1：HttpClient.SecureProtocol = 0/1/2/3/4/5

* 设置HttpClient安全验证协议版本
* 0 – All secure protocols. This is a default value.
* 1 – SSL2
* 2 – SSL3
* 3 – TLS1.0
* 4 – TLS1.1
* 5 – TLS1.2

语法2：Integer li\_rc = HttpClient.SecureProtocol

* 获取当前HttpClient安全验证协议版本
* 0 – All secure protocols. This is a default value.
* 1 – SSL2
* 2 – SSL3
* 3 – TLS1.0
* 4 – TLS1.1
* 5 – TLS1.2

#### Timeout

语法1：HttpClient.Timeout = 8

* 设置HttpClient与服务器交互的超时时间，如果超过设置的时间没有返回值则认为超时
* 单位：秒

语法1：Long ll\_rc = HttpClient.Timeout

* 获取当前HttpClient与服务器交互设置的超时时间
* 单位：秒

## RestClient

### RestClient（PB对象测试）

* 测试classname函数
* 测试typeof函数
* 测试create和destroy

### 功能测试

#### ClearResquestHeaders

语法：Integer ClearRequestHeaders()

* 未发出任何请求，调用该函数
* 发送各方式消息（post、get等），调用该函数
* 多次重复掉哦那个该函数

#### SetRequestHeader

语法：Integer SetRequestHeader(string headerName, string headerValue)

* 参数headerName错误或不存在的情况
* 参数headerName和headervalue不匹配的情况
* 各正常参数情况
* 参数为Null的情况
* Headervalue为多字节情况
* Headervalue为特殊字符情况
* Headervalue为超长字符情况

#### GetRequestHeader

语法：String GetRequestHeader(string headerName)

* 参数headerName错误或不存在的情况
* 参数为Null的情况
* HeaderName为多字节或特殊字符情况
* 不同类型的HeaderName，看能否正常返回
* 未执行信息请求情况下，该函数的返回值情况
* Headername对应的数值为多字节情况

#### SetRequestHeaders

语法：Integer SetRequestHeaders(string headers)

* 参数headers只有一个header，没有values的情况
* 参数只有一组header数据的情况
* 参数有多组数据，header1:value1~r~nheader2~r~nheader3:values3~r~n
* 参数~r~n在最前面的情况~r~nheader1:value1~r~nheader2:values2~r~nheader3:values3
* 通过char(13)+char(10)，形成换行的参数header1:value1 char(13)+char(10)header2:values2
* 多组正常header值的情况
* 只传空格加换行符的情况 space(10)+ char(13)+char(10)+ space(10)+ char(13)+char(10)

#### GetRequestHeaders

语法：String GetRequestHeaders()

* 执行完setrequestheader(str header,str value)，查看该函数的返回值
* 未执行信息请求情况下，该函数的返回值情况
* 正常执行SetRequestHeaders后，该参数的返回值
* SetRequsetHeaders执行异常后，该函数的返回值

#### Retrieve

语法一：Long Retrieve( dw dwControl/ds datastore, string urlName) GetMethod

* Urlname为Null的情况下执行
* Ds未实例化的情况下执行
* Dw/ds 的dataobject不存在的情况下执行
* Dw/ds的dataobject数据列与url不匹配的情况下执行效果
* 通过该方法得到的数据，能否通过数据共享，展示到其他datawindow控件
* 该方法的datawindow数据，如何update（暂时未做）
* 各风格的datawindow，是否能正常展示数据

语法二：Long Retrieve(Dw dwControl/ds datastore, string urlName, string data) Post

* data为Null的情况下执行
* urlName不合法和网络不通的情况
* data为多字节数据的情况
* data为大文本数据的情况
* data不符合网站请求数据格式的情况
* 正常传输data数据的情况

语法三：Long Retrieve(Dw dwControl/ds datastore, string urlName, blob data) Post

* data为Null的情况下执行
* urlName不合法和网络不通的情况
* data的字符集与url不匹配的情况
* data数据传输过程中网路不稳定中断的情况
* 正常传输data数据的情况

#### GetResponseHeader

语法：String GetResponseHeader(string headerName)

* 参数headerName错误或不存在的情况
* 参数为Null的情况
* 参数headername正常，但未执行setrequestheader的情况
* 执行完setrequestheaders，查看该函数的返回情况
* 网络不通时查看函数的执行情况
* 正常执行完setrequestheader后，查看该函数的执行情况

#### GetResponseHeaders

语法：String GetResponseHeaders()

* 未执行setrequestheaders，查看函数的执行情况
* 执行完setrequestheader，查看该函数的执行情况
* 执行setrequestheaders异常后，查看该函数的执行情况
* 正常执行完setrequestheaders后，查看该函数的执行情况
* 多次反复执行该函数查看，函数的执行情况

#### GetResponseStatuscode

语法：Long GetResponseStatusCode()

* 网络不通时查看执行情况
* 未发出任何请求，查看该函数的执行情况
* 当请求的url不存在时，查看函数的返回值
* 当请求信息异常时，查看该函数的返回值
* 正常请求成功下，该函数的返回值

#### GetResponseStatusText

语法：String GetResponseStatusText()

* 网络不通时查看执行情况
* 未发出任何请求，查看该函数的执行情况
* 当请求的url不存在时，查看函数的返回值
* 当请求信息异常时，查看该函数的返回值
* 正常请求成功下，该函数的返回值

### 属性测试

#### SecureProtocol

主要包括：

0 – All secure protocols. This is a default value.

1 – SSL2

2 – SSL3

3 – TLS1.0

4 – TLS1.1

5 – TLS1.2

* 该属性设置为-1的情况
* 该属性设置为大于5的情况
* 属性设置为正常值的情况

#### Timeout

服务器地址不可达，测试超时情况

* 未设置该数据时，是否60S超时
* 设置为0时，是否不会超时
* 设置为-1时，查看能否超时
* 设置为0.01s时，是否时时超时
* 设置为120S时，2分钟后是否超时

### 使用场景

一周内天气情况汇总，将一周内的数据通过restclient从气象站获取，以graph dw方式展示到datawindow中，观察天气变化