RTC脚本模型课堂 - 简介

2017-01-04 09:36:36 66 0

大家好！RTC脚本模型推出已经有段时间了，目前版本已经发展到1.2，目前的功能已经能够满足一般的网站制作或者接口api开发所需，现在，对它进行一个简单系统的介绍。

为什么要制作这么一个脚本模型呢？起因是这样的，做为一个Delphi开发人员，web开发总是我的弱项，而花费巨大的精力去重新学习其它语言，成本相对较高，也没法及时的解决手头上的项目，当然，学还是要学的，但一下子把web前端和web后端开发在短时间内学会，这也是不现实的。因为本人有一点前端html和css基础，后端asp基础，在参考php以后，想到一个解决方案，那就是找一个pascal语法的脚本引擎来实现php那样的功能，然后先把web前端开发先学会，以后有时间了再去学web后端开发，比如系统的学习一下php的开发。

RTC脚本模型的诞生：这几年，因为项目基本上都走http协议，所以比较喜欢使用RealThinClientSDK这个商业组件包，它是基于http应用层协议的轻量级传输和支撑组件，帮助我们解决数据传输和http服务支持的问题；在2016年4月中旬的时候，我首先尝试了使用rtc做为web服务端，paxCompiler做为脚本解析引擎的方案，开发出了最基本的框架，并实现了简单的web应答式脚本支持，当时的唯一功能，只有一个http post/get请求的实现，也就是在脚本中编写一个http post/get请求，将结果进行展示；在月底的时候，忽然发现rtc自己就带了脚本解析功能，于是又回去查看了它的示例，发现它的语法和脚本编写方式更加适合web开发，和php、asp一类的非常相似，于是将脚本解析部分，也换成rtc自己的了。就这样，我开始了边应用于实际项目，边完善脚本模型的历程，直到2016年11月19号，正式对外发布了rtc脚本模型1.0正式版本，当时已经具备了基本的web后端开发能力和api开发能力，也完成了公司一个项目管理后台的开发，包括给苹果ios app调用的api接口和网页版管理后台。

如何保持RTC脚本模型的活力呢？因为使用的是商业组件，所以直接开源是不可取的，而delphi开发人员在国内本身也已经不多了，使用rtc的群体更加的少，于是，我决定采用软件终身免费使用，功能及有效期限均不做限制，保留版权，不开源的方式（我记得有这样的授权协议的，具体忘了是哪个），让它与大家见面，让一些delphi开发人员，可以使用pascal语法进行web后端开发；当然，我也接受购买源码的请求，这样也就有一部分人可以在一起对框架进行讨论和改进，免费使用的用户也可以提出一些合理的要求及建议。基本上，只要它还能满足我自己项目需求，我就不会放弃更新它的。

那么，RTC脚本模型的原理是什么呢？其实也很简单，它本身就是一个web服务提供者，展示静态web资源，解析脚本后动态展示web资源；因为它只是将需要展示的数据进行输出（或者是解析处理后输出），所以对web前端的兼容全部交给了浏览器，也就不用于过关心web前端的兼容问题了，只要注意哪些web前端脚本是基于哪些版本的浏览器就行；web后端呢，则是类似php那样，由核心提供一堆的内置函数，供脚本调用，以实现各种功能，比如文件操作、数据转换等。基本流程就是用户从客户端（浏览器或者软件实现）发起一个请求，服务端（脚本模型）进行应答，基本上和web后端是一毛一样了。我也有考虑再加主动式执行的脚本功能，这个暂时还在计划中，具体实现时间还没定下来，如果加了这个功能，那就能解决更多的应用需求了。

对于delphi开发人员，如果您有兴趣用它进行web开发，我的建议是去了解一下bootstrap和jquery，其实bootstrap就是相当于delphi中的vcl了，比如你要使用一个按钮，那你只要查看一下bootstrap中的按钮的做对，复制过来，然后设置一下样式，绑定一下js函数就可以了，是不是很简单，很像delphi中的vcl呢？我就是这样折腾的。有兴趣的可以下载本站，也就是offeu.com的整站脚本，看看我是如何实现的。

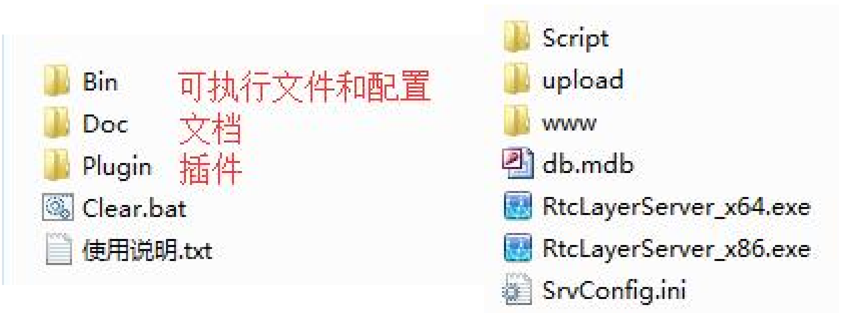
嗯，我写的比较乱，文笔不行，所以大家伙凑合着看就行了，后续我将写一些使用教学，希望对您有所帮助。

RTC脚本模型课堂 - hello world

2017-01-04 09:39:14 48 0

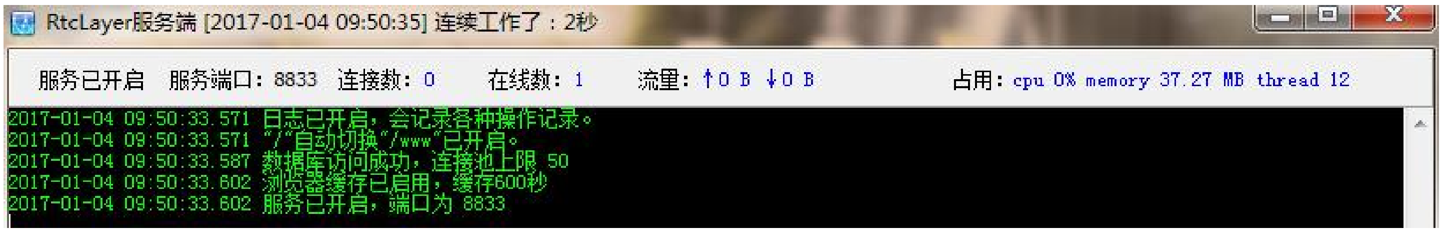
对程序员而言，是不是应该从hello world开始呢？

首先，介绍下如何使用脚本模型，从offeu.com下载最新的rtc脚本模型，它是一个压缩包，下载到本地，解压后，你会看到它的目录结构如下：



在开始之前，建议您先查看一下doc目录下的几个文档，分别是 部分说明 升级日志 和 注意事项，部分说明就是脚本模型内置函数的清单和说明，也包含简单的示例。

现在开始写咱们的第一个脚本，进入bin目录，运行RtcLayerServer_x86.exe，如下图：



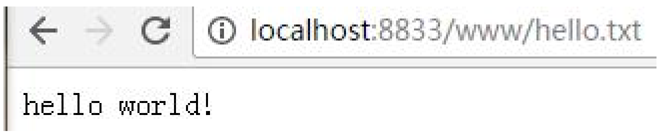
```
服务已开启  服务端口：8833  连接数：0    在线数：1    流量：↑0 B ↓0 B         占用：cpu 0% memory 37.27 MB thread 12
2017-01-04 09:50:33.571 日志已开启，会记录各种操作记录。
2017-01-04 09:50:33.571 "/"自动切换"/www"已开启。
2017-01-04 09:50:33.587 数据库访问成功，连接池上限 50
2017-01-04 09:50:33.602 浏览器缓存已启用，缓存600秒
2017-01-04 09:50:33.602 服务已开启，端口为 8833
```

在图中，您可以看到，默认设置已经让其开启了服务，8833就是服务端口。

进入bin\script目录，新建一个文本文件，将它命名为hello.txt，然后打开它，在里面输入以下内容：

```
<?'hello world!'?>
```

输入完成后，保存，然后打开您电脑上的浏览器，输入 http://localhost:8833/www/hello.txt，就能看到您的第一个脚本实现了



```
hello world!
```

解释一下，rtcscript的基本语法就是用<??>框起来的，单引号则表示需要输出的内容，注意！脚本模型要求所有脚本文件用utf-8编码格式保存，浏览器上也会自动输出utf-8编码格式的内容，比如你输入中文，也能正常的显示出来。

RTC脚本模型课堂 - 基本语法

2017-01-04 10:38:12 48 0

在rtc组件的目录下有个脚本演示，建议去看一下，在\QuickStart\rtcScript目录下。

```
脚本需要包含在 <? 和 ?> 之中。

//脚本中的字符串
Strings inside the Script:

    Long HTML strings are shown in HTML editor
    and may be spread across multiple lines:
//这是一个单行字符串的表现形式
    ?> this is a single-line string <?
//这是一个多行字符串的表现形式
        ?> this would be a multi-line string,
           which can spread across
           the whole document <?

    Short strings will NOT show in HTML editors
    and can NOT be spread across multiple lines
    (need to be opened and closed on the same line):
    " abc "
    ' abc '
//字符串用单引号或双引号括起来

//整型为0到9
Characters for Integer values:
    0-9

//浮点型为0到9和小数点
Characters for Floating-point values:
    0-9
    .

//数组，记录和函数调用参数的打开和关闭：
Arrays, Records and Function call parameters opening and closing:
    ( )
    [ ]
    { }
    begin end

//注释
Comments
//单行
    // abc        single-line comment
//多行
    (* abc *)     multi-line comment

//命令和元素分隔符（数组，记录集，函数调用参数）
Command and List element separators (arays, records, function call parameters):
    ,
    ;

//函数和变量名称中允许的字符
Characters allowed in Function and Variable names:
    a..z
    A..Z
    0..9
    _
    $

//用Delphi编写的函数需要由字母或者下划线开始
Functions written in Delphi begin with:
    a..z
    A..Z
    _

//脚本变量和函数以$号开始
Script Variables and Functions begin with:
    $

//使用变量名称作为字符串访问脚本变量
Accessing Script variables using a variable name as string:
//执行表达式以获取变量名称，返回变量内容
```

```
    $( expression ) - executes the expression to get variable name, returns variable content

    Examples://示例
    $('x') is the same as $x //$('x')和$x是一样的
    $('x.y') is the same as $x.y
    $y:="x"; $($y) - since variable $y is "x", $($y) returns the value of $x

//可以读取和更改变量（分配新值）。
    Variables can be read and changed (new values assigned) like this.

//访问数组中的字段
Accessing fields in an Array:
    .123        - get element at index 123
    ( 123 )     - get element at index 123

    ( expression )  - get element at index we get after evaluating the expression
              Expression will be converted to an Integer

//访问记录集中的字段
Accessing fields in a Record:
    .XYZ        - get field with name "XYZ"
    ('XYZ')     - get field with name "XYZ"

    ( expression )  - get element with the name we get after evaluating the expression
              Expression will be converted to a string if it isn't a string

//访问数据库记录集中的字段
Accessing fields in a DataSet:
    .123        - get field with index 123
    .XYZ        - get field with name "XYZ"
    ( 123 )     - get field with index 123
    ('XYZ')     - get field with name "XYZ"

    ( expression )  - if expression starts with a digit, get field at given index.
              Otherwise, get field with given name.

//数组变量上可用的属性
Properties available on Array variables:
    .COUNT      - number of elements in array
    .FIELDCOUNT - same as .COUNT

//记录集变量上可用的属性
Properties available on Record variables:
    .COUNT      - number of fields in record
    .FIELDCOUNT - same as .COUNT

//数据库记录集变量上可用的属性
Properties available on DataSet variables:
    .COUNT      - number of rows in dataset
    .ROWCOUNT   - same as .COUNT
    .FIELDCOUNT - number of fields in row
    .ROW        - current row number (read/write property: set to change row)

    .EOF        - returns TRUE if behind last row
    .BOF        - returns TRUE if before first row
    .EMPTY      - returns TRUE if dataset is empty (ROWCOUNT = 0)

//DataSet变量上可用的方法（方法返回NULL）
Methods available on DataSet variables (methods return NULL):
    .FIRST      - jumps to first row
    .LAST       - jump to last row
    .NEXT       - jumps to the next row
    .PRIOR      - jumps to the prior row

    .INSERT     - inserts a new row at the current positon
    .APPEND     - appends a new row at the end of the dataset
    .DELETE     - deletes the current row from the dataset

//函数调用
Function calls:
//无参数
    Without parameters ...

    FunctionName
    FunctionName()

    $FunctionName
    $FunctionName()

//有参数
```

```
    With parameters sent inside the "PARAMS" array:

    FunctioName (value1, value2, ...)
    FunctioName (value1; value2; ...)

    $FunctioName (value1, value2, ...)
    $FunctioName (value1; value2; ...)

//也可以带参数名
    With parameters sent using parameter names:

    FunctioName (paramName1:value1, paramName2:value2, ...)
    FunctioName (paramName1:value1; paramName2:value2; ...)

    $FunctioName (paramName1:value1, paramName2:value2, ...)
    $FunctioName (paramName1:value1; paramName2:value2; ...)

//构建数组
Constructing Arrays
    (value1, value2, value2 ...)
    (value1; value2; value2 ...)

//构建记录集
Constructing Records
    (name1:value1, name2:value2, name3:value3)
    (name1:value1; name2:value2; name3:value3)

//为空的表达形式
NULL or NILL values (the same effect):
    NULL
    NIL

//构建变量方式
Constructor:
    NEW ARRAY
    NEW RECORD
    NEW DATASET
    NEW STREAM  or NEW BYTESTREAM
    NEW INTEGER or NEW INT  or NEW INT32
    NEW LARGEINT    or NEW INT64
    NEW DATETIME    or NEW DATE or NEW TIME
    NEW FLOAT   or NEW DOUBLE
    NEW CURRENCY
    NEW BOOLEAN or NEW BOOL
    NEW WIDESTRING
    NEW TEXT

    Example:
    $x := new DataSet;

//操作符
Left-side operators, working as functions with 1 parameter:

    not not X

    !   ! X - same as not X
    -   - X

    #   # X

String, numeric and logical operators for functions with 2 parameters:

    +   X + Y
    -   X - Y
    #   X # Y   - X is string, Y is

    div X div Y
    mod X mod Y

    *   X * Y
    /   X / Y   - same as X div Y
    %   X % Y   - same as X mod Y

    shl X shl Y
    shr X shr Y

    <<    X << Y    - same as X shl Y
    >>    X >> Y    - same as X shr Y

    and X and Y
    or  X or Y
    xor X xor Y

    &   X & Y   - same as X and Y
    |   X | Y   - same as X or Y

    { not implemented }
    ^   X ^ Y


Comparison operators for functions with 2 parameters:
    >    X > Y
    <    X < Y
    =   X = Y
    ==  X == Y  - same as X = Y

    <=   X <= Y
    >=   X >= Y

    <>    X <> Y
    !=  X != Y  - same as X <>  Y

    { not implemented }
    in  X in Y
    is  X is Y

Assignment operators:
    :=  X := Y
    +=  X += Y
    -=  X -= Y
    *=  X *= Y
    /=  X /= Y
    %=  X %= Y
    &=  X &= Y
    |=  X |= Y

    { not implemented }
    ^=  X ^= Y

//操作符优先级
Operator priority:

    Assignment operators always evaluete the complete right side
    before assigning the result to the variable on the left side.

    High priority operators:
    * / % div mod

    High priority operators work on the result from the left side
    and a the first element from are right side.

    Low priority operators:
    < > <= >= == != <>

    Low priority operators work on the result from the left side
    and the result from the right side (executed as last command).

//在脚本中编写函数
Writing Functions inside Script:

    FUNCTION $FunctionName command;

    Function result has to be assigned to the $RESULT variable,
    which can also be used to store temporary data until the final result
    has been prepared. Once the function execution completes, only the data
    stored in the $RESULT variable will be passed on as function result.

    FUNCTION $FunctionName $result := command;

    If the function has more than a single command, put it isnide begin/end.
    Here is an example of a function to return a sum from 1 to the passed parameter:

    FUNCTION $test if $params <> nil then
      begin
      $result := 0;
      for $i := 1 to $params.0 do $result += $i;
      end;

    And here is how this simple function can be called:

```

```
    $test(100);

//条件表达形式
Conditional code branching:

    IF condition THEN command;

    IF condition THEN command1 ELSE command2;

//循环
Command Loops:

    REPEAT command-list UNTIL condition;

    WHILE condition DO command;

    FOR $variable := minvalue TO maxvalue DO command;
    FOR $variable := maxvalue DOWNTO minvalue DO command;

   { not implemented }
    FOREACH $variable := enumeration DO command;

//代码块
Code block declaration:

    CODE command

    Code Blocks are "blocks" of execution code which you can assign to local
    script variables or send as parameters to local script function calls.

    When a code block (command with a "code" prefix) is passed to a script function call,
    it will be executed in the scope of the script function.

    When a code block is assigned to a variable (or a variable property),
    each time the variable is evaluated, the code block (assigned to that variable)
    will be executed and the result returned as if the code was called as a function and
 stored into the variable.

    Examples:

    // define a code block for variable $x ...
    $x := code if $y>0 then $y else -$y;

    $y := 15;
    $x; // execute code x" output 15

    $a := code $x; // $a will be returning the value of $x

    // you can pass codeblocks to local function calls ...
    $myFunc(a:code $x);

    // you can pass any expression as a code block ...
    $myFunc(code if $x>0 then $x else $y);



Code block access:

    // To assign a codeblock to a variable, use ...

    $x := code $y;

    @x := 15;//this will assign "15" to variable $y, since it is stored in codeblock x

    $x := 25; // this will assign value "25" to variable $x,
 replacing any prior assignments to $x



NOTE: Function calls can be nested, all operators can be used anywhere inside a script.

Results from any function call, command and variables which are not assigned
to other variables will implicitly become part of the resulting output (HTML/XML).

All variable and function names are case-insensitive (NOT CASE SENSITIVE),
so you can use lowercase, uppercase or a combination of lower and uppercase names.

注意：函数调用可以嵌套，所有操作符都可以在脚本中的任何地方使用。

来自任何未分配的函数调用，命令和变量的结果
```

```
到其他变量将隐式地成为结果输出（HTML / XML）的一部分。

所有变量和函数名称不区分大小写（NOT CASE SENSITIVE），
因此您可以使用小写，大写或小写和大写名称的组合。


//其它一些对象的属性
Request parameters can be read and written using the "Request" variable:

    REQUEST.METHOD      = request method (examples: "GET", "POST", "PUT", ...)
    REQUEST.FILENAME    = request file name (example: "/myfile.rtc" )
    REQUEST.CLOSE       =should the connection be closed after respose was sent?(HTTP/1.0)
    REQUEST.URI     = Request URI (all exept the host)
    REQUEST.URL     = Request URL (host + URI) - read only

    REQUEST.CONTENTTYPE = "CONTENT-TYPE" HTTP reuqest header value
    REQUEST.CONTENTLENGTH   = "CONTENT-LENGTH" HTTP request header value
    REQUEST.HOST        ="HOST" HTTP request header value(example:www.realthinclient.com)
    REQUEST.AGENT       ="USER-AGENT" HTTP request header value(example:Mozila Firefox...)
    REQUEST.REFERER     = "REFERER" HTTP request header value
    REQUEST.FORWARDEDFOR    = "X-FORWARDED-FOR" HTTP request header value


    REQUEST.HEADER      or
    REQUEST.HEADER.TEXT = complete HTTP request header

    REQUEST.HEADER('abc')   or
    REQUEST.HEADER.abc  = value of the HTTP request header with name 'abc'
(example name: "CONTENT-TYPE")

    REQUEST.HEADER.COUNT        = number of HTTP request header variables
    REQUEST.HEADER.NAME( 123 )  =name of HTTP request header at index 123(starting at 0)
    REQUEST.HEADER.VALUE( 123 ) =value of HTTP request header at index 123(starting at 0)


    REQUEST.COOKIE      or
    REQUEST.COOKIE.TEXT = complete request cookie text

    REQUEST.COOKIE('abc')   or
    REQUEST.COOKIE.abc  = value of request cookie parameter with name 'abc'

    REQUEST.COOKIE.DELIMITER    = request cookie parameters delimiter
    REQUEST.COOKIE.COUNT        = number of parameters in a request cookie - read only
    REQUEST.COOKIE.NAME( 123 )
= name of the request cookie parameter at index 123 (starting at 0)
    REQUEST.COOKIE.VALUE( 123 )
= value of the request cookie parameter at index 123 (starting at 0)


    REQUEST.QUERY       or
    REQUEST.QUERY.TEXT  = complete request query text

    REQUEST.QUERY('abc')    or
    REQUEST.QUERY.abc   = value of request query parameter with name 'abc'

    REQUEST.QUERY.DELIMITER     =request query parameter delimiter (examples: ";", "&")
    REQUEST.QUERY.COUNT     = number of query parameters - read only
    REQUEST.QUERY.NAME( 123 )   = name of query parameter at index 123 (starting at 0)
    REQUEST.QUERY.VALUE( 123 )  =value of query parameter at index 123 (starting at 0)


    REQUEST.PARAMS      or
    REQUEST.PARAMS.TEXT = complete input as one string

    REQUEST.PARAMS('abc')   or
    REQUEST.PARAMS.abc  = value of input parameter with name 'abc'

    REQUEST.PARAMS.DELIMITER    = input parameters delimiter
    REQUEST.PARAMS.COUNT        = number of input parameters - read only
    REQUEST.PARAMS.NAME( 123 )= name of input parameter at index 123 (starting at 0)
    REQUEST.PARAMS.VALUE( 123 )= value of input parameter at index 123 (starting at 0)


    REQUEST.INFO('abc') or
    REQUEST.INFO.abc    = access to "Request.Info" variable 'abc'
                  All Request.Info variables can be accessed from Delphi.

    REQUEST('abc')      or
    REQUEST.abc     = Value of the HTTP request header with name "abc"
                  Example: Request("CONTENT-TYPE")
```

```



Response parameters can be read and written using the "Response" variable:


    RESPONSE.STATUS
= response Status Code+Text("200 OK", "404 File Not Found", etc)

    RESPONSE.STATUSCODE or
    RESPONSE.STATUS.CODE    = response Status Code (200, 404, ...)

    RESPONSE.STATUSTEXT or
    RESPONSE.STATUS.TEXT    = response Status Text ("OK", "File Not Found", etc)

    RESPONSE.CONTENTTYPE    = "CONTENT-TYPE" HTTP response header value
    RESPONSE.CONTENTLENGTH  = "CONTENT-LENGTH" HTTP response header value


    RESPONSE.HEADER     or
    RESPONSE.HEADER.TEXT    = complete HTTP response header

    RESPONSE.HEADER('abc')  or
    RESPONSE.HEADER.abc = value of the HTTP response header with name 'abc'
                (example name: "CONTENT-TYPE")

    RESPONSE.HEADER.COUNT       = number of HTTP response header variables
    RESPONSE.HEADER.NAME( 123 ) = name of HTTP response header at index 123
 (starting at 0)
    RESPONSE.HEADER.VALUE( 123 )    = value of HTTP response header at index 123
 (starting at 0)


    RESPONSE.COOKIE     or
    RESPONSE.COOKIE.TEXT    = complete response cookie text

    RESPONSE.COOKIE('abc')  or
    RESPONSE.COOKIE.abc = value of response cookie parameter with name 'abc'

    RESPONSE.COOKIE.COUNT       =number of parameters in a response cookie-read only
    RESPONSE.COOKIE.NAME( 123 )
= name of response cookie parameter at index 123 (starting at 0)
    RESPONSE.COOKIE.VALUE( 123 )
= value of response cookie parameter at index 123 (starting at 0)
    RESPONSE.COOKIE.DELIMITER   = response cookie parameters delimiter


    RESPONSE('abc')     or
    RESPONSE.abc        = Value of the HTTP response header with name "abc"
                Example: Response("CONTENT-TYPE")


You can work with sessions by using the "Session" variable:

    SESSION.ID      = Session ID

    SESSION.OPEN        or
    SESSION.OPEN.FWDLOCK    or
    SESSION.OPEN.PRIVATE    = Open and lock a new private user session.
                This is the default private session type,
              which works with users behind all kinds of Proxy servers.
                Locks the Session to current user's IP address or to
                the "X-FORWARDED-FOR" HTTP header (if it exists).

    SESSION.OPEN.PUBLIC = Open and lock a new PUBLIC Session, which should NOT be locked
 to this specific user.
           This Session can be accessed by any user who knows the Session ID.
                When working with PUBLIC sessions, always use SESSION.LOCK() to
 obrain a lock.

    SESSION.OPEN.IPLOCK = This option will NOT work for users behind Proxy Servers with
 changing IP.
                Lock the Session to users's IP address,
 ignoring "X-FORWARDED-FOR" HTTP header.

    SESSION.OPEN.SECURE or
    SESSION.OPEN.STRONG or
    SESSION.OPEN.IPFWDLOCK  = This option will NOT work for users behind Proxy Servers with
 changing IP.
                Opens and locks a new session with a strong and secure link to
 the current user.
                 Lock the Session to users's IP address *and* to the
"X-FORWARDED-FOR" HTTP header.

   SESSION.FIND('abc') or
   SESSION.FIND.abc    = Default method for finding and locking Private user Sessions.
                 Finds a Session with ID "abc" and tries to lock it,
 without waiting.
                 Returns FALSE if Session can not be locked because does
 NOT exist,
                   or if it is locked by anotuer user.
                 Returns TRUE and Locks the Session if Session exists and
                   is NOT currently locked by another user.

   SESSION.HAVE('abc') or
   SESSION.HAVE.abc    = Check if session with ID "abc" exists.
                 Returns TRUE if Session exists (even if locked), but does
 NOT lock the session.

   SESSION.LOCK('abc') or
   SESSION.LOCK.abc    = Lock a PUBLIC Session (which can be shared by multiple users).
                 Find Session with ID "abc" and wait until you can lock it
 for usage.
                 If session exists, it will be locked and TRUE will be returned.
                 If Session does NOT exist
 (or should be closed while waiting for a lock), returns FALSE.

   SESSION.CLOSE       = close current session

   SESSIOn.CLOSE('abc')    or
   SESSIOn.CLOSE.abc   = Find and close session 'abc'

   SESSION.KEEPALIVE   =number of seconds the session should be kept alive when not used
   SESSION.FINALEXPIRE = date/time when the Session has to expire, regardless
 of "SESSION.KEEPALIVE"

   SESSION.EXPIRETIME  = date/time when the Session would expire if not used (read-only)

   SESSION.COUNT       or
   SESSION.COUNT.TOTAL = Total number of currently active sessions (read-only)

   SESSION.COUNT.LOCKED    = Number of currently active and locked sessions (read-only)
   SESSION.COUNT.UNLOCKED  = Number of currently active but not locked sessions (read-only)

   SESSION('abc')  or
   SESSION.abc= access to current sessions variable "abc". Before you can access session
 variables,
             you need to find and lock an existing session using "Session.Find"
(or from a Delphi function),
             or open and lock a new session using "Session.Open". If no session
 is locked,
             all variables will return NULL on read access, but an exception will
 be raised on write access.


Data received through as Query parameters (part of the URL) can be accessed
by using the REQUEST.QUERY variable, or by using the QUERY variable
 (without the "REQUEST." prefix):

   QUERY           or
   QUERY.TEXT      = complete request query text

   QUERY.COUNT     = number of query parameters - read only
   QUERY.DELIMITER     = request query parameter delimiter (examples: ";", "&")
   QUERY.NAME( 123 )   = name of query parameter at index 123 (starting at 0)
   QUERY.VALUE( 123 )  = value of query parameter at index 123 (starting at 0)

   QUERY('abc')        or
   QUERY.abc       = value of request query parameter with name 'abc'


Data received from a FORM POST when using INPUT elements inside HTML can be accessed
 through the INPUT variable:

   INPUT           or
   INPUT.TEXT      = complete input as one string

   INPUT.COUNT     = number of input parameters - read only
   INPUT.DELIMITER     = input parameters delimiter
   INPUT.NAME( 123 )   = name of input parameter at index 123 (starting at 0)
```

```
    INPUT.VALUE( 123 )  = value of input parameter at index 123 (starting at 0)

    INPUT('abc')        or
    INPUT.abc       = value of input parameter with name 'abc'


Basic Client information can be obrained by using the CLIENT variable (read-only):

    CLIENT      = Client's IP address and Port in the form "IP:Port"
 (example: "196.45.32.112:1604")

    CLIENT.IP   or
    CLIENT.ADDR or
    CLIENT.ADDRESS  = Client's IP address

    CLIENT.PORT = Client's Port number (as string)

    CLIENT.COUNT    =Number of currently conencted clients(total client connection count)


Basic Server information can be obtained by using the SERVER variable (read-only):

    SERVER      = Server's IP address and Port in the form "IP:Port"
(example: "127.0.0.1:80")

    SERVER.IP   or
    SERVER.ADDR or
    SERVER.ADDRESS  = Our Server's Local IP address

    SERVER.PORT = Out Server's Port number (as string)
              Standard HTTP port = 80, Standard HTTPS (SSL) port = 443
```

RTC脚本模型课堂 - 基本语法

2017-01-04 10:38:12 48 0

在rtc组件的目录下有个脚本演示，建议去看一下，在\QuickStart\rtcScript目录下。

```
脚本需要包含在 <? 和 ?> 之中。

//脚本中的字符串
Strings inside the Script:

    Long HTML strings are shown in HTML editor
    and may be spread across multiple lines:
//这是一个单行字符串的表现形式
    ?> this is a single-line string <?
//这是一个多行字符串的表现形式
        ?> this would be a multi-line string,
           which can spread across
           the whole document <?

    Short strings will NOT show in HTML editors
    and can NOT be spread across multiple lines
    (need to be opened and closed on the same line):
    " abc "
    ' abc '
//字符串用单引号或双引号括起来

//整型为0到9
Characters for Integer values:
    0-9

//浮点型为0到9和小数点
Characters for Floating-point values:
    0-9
    .

//数组，记录和函数调用参数的打开和关闭：
Arrays, Records and Function call parameters opening and closing:
    ( )
    [ ]
    { }
    begin end

//注释
Comments
//单行
    // abc        single-line comment
//多行
    (* abc *)     multi-line comment

//命令和元素分隔符（数组，记录集，函数调用参数）
Command and List element separators (arays, records, function call parameters):
    ,
    ;

//函数和变量名称中允许的字符
Characters allowed in Function and Variable names:
    a..z
    A..Z
    0..9
    _
    $

//用Delphi编写的函数需要由字母或者下划线开始
Functions written in Delphi begin with:
    a..z
    A..Z
    _

//脚本变量和函数以$号开始
Script Variables and Functions begin with:
    $

//使用变量名称作为字符串访问脚本变量
Accessing Script variables using a variable name as string:
//执行表达式以获取变量名称，返回变量内容
```

```
    $( expression ) - executes the expression to get variable name, returns variable content

    Examples://示例
    $('x') is the same as $x //$('x')和$x是一样的
    $('x.y') is the same as $x.y
    $y:="x"; $($y) - since variable $y is "x", $($y) returns the value of $x

//可以读取和更改变量（分配新值）。
    Variables can be read and changed (new values assigned) like this.

//访问数组中的字段
Accessing fields in an Array:
    .123         - get element at index 123
    ( 123 )      - get element at index 123

    ( expression )  - get element at index we get after evaluating the expression
              Expression will be converted to an Integer

//访问记录集中的字段
Accessing fields in a Record:
    .XYZ         - get field with name "XYZ"
    ('XYZ')      - get field with name "XYZ"

    ( expression )  - get element with the name we get after evaluating the expression
              Expression will be converted to a string if it isn't a string

//访问数据库记录集中的字段
Accessing fields in a DataSet:
    .123         - get field with index 123
    .XYZ         - get field with name "XYZ"
    ( 123 )      - get field with index 123
    ('XYZ')      - get field with name "XYZ"

    ( expression )  - if expression starts with a digit, get field at given index.
              Otherwise, get field with given name.

//数组变量上可用的属性
Properties available on Array variables:
    .COUNT       - number of elements in array
    .FIELDCOUNT - same as .COUNT

//记录集变量上可用的属性
Properties available on Record variables:
    .COUNT       - number of fields in record
    .FIELDCOUNT - same as .COUNT

//数据库记录集变量上可用的属性
Properties available on DataSet variables:
    .COUNT       - number of rows in dataset
    .ROWCOUNT    - same as .COUNT
    .FIELDCOUNT - number of fields in row
    .ROW         - current row number (read/write property: set to change row)

    .EOF         - returns TRUE if behind last row
    .BOF         - returns TRUE if before first row
    .EMPTY       - returns TRUE if dataset is empty (ROWCOUNT = 0)

//DataSet变量上可用的方法（方法返回NULL）
Methods available on DataSet variables (methods return NULL):
    .FIRST       - jumps to first row
    .LAST        - jump to last row
    .NEXT        - jumps to the next row
    .PRIOR       - jumps to the prior row

    .INSERT      - inserts a new row at the current positon
    .APPEND      - appends a new row at the end of the dataset
    .DELETE      - deletes the current row from the dataset

//函数调用
Function calls:
//无参数
    Without parameters ...

    FunctionName
    FunctionName()

    $FunctionName
    $FunctionName()

//有参数
```

```
    With parameters sent inside the "PARAMS" array:

    FunctioName (value1, value2, ...)
    FunctioName (value1; value2; ...)

    $FunctioName (value1, value2, ...)
    $FunctioName (value1; value2; ...)

//也可以带参数名
    With parameters sent using parameter names:

    FunctionName (paramName1:value1, paramName2:value2, ...)
    FunctionName (paramName1:value1; paramName2:value2; ...)

    $FunctionName (paramName1:value1, paramName2:value2, ...)
    $FunctionName (paramName1:value1; paramName2:value2; ...)

//构建数组
Constructing Arrays
    (value1, value2, value2 ...)
    (value1; value2; value2 ...)

//构建记录集
Constructing Records
    (name1:value1, name2:value2, name3:value3)
    (name1:value1; name2:value2; name3:value3)

//为空的表达形式
NULL or NILL values (the same effect):
    NULL
    NIL

//构建变量方式
Constructor:
    NEW ARRAY
    NEW RECORD
    NEW DATASET
    NEW STREAM  or NEW BYTESTREAM
    NEW INTEGER or NEW INT  or NEW INT32
    NEW LARGEINT    or NEW INT64
    NEW DATETIME    or NEW DATE or NEW TIME
    NEW FLOAT   or NEW DOUBLE
    NEW CURRENCY
    NEW BOOLEAN or NEW BOOL
    NEW WIDESTRING
    NEW TEXT

    Example:
    $x := new DataSet;

//操作符
Left-side operators, working as functions with 1 parameter:

    not not X

    !   ! X - same as not X
    -   - X

    #   # X

String, numeric and logical operators for functions with 2 parameters:

    +   X + Y
    -   X - Y
    #   X # Y   - X is string, Y is

    div X div Y
    mod X mod Y

    *   X * Y
    /   X / Y   - same as X div Y
    %   X % Y   - same as X mod Y

    shl X shl Y
    shr X shr Y

    <<    X << Y    - same as X shl Y
    >>    X >> Y    - same as X shr Y

    and X and Y
    or  X or Y
    xor X xor Y

    &   X & Y   - same as X and Y
    |   X | Y   - same as X or Y

    { not implemented }
    ^   X ^ Y


Comparison operators for functions with 2 parameters:
    >    X > Y
    <    X < Y
    =   X = Y
    ==  X == Y  - same as X = Y

    <=   X <= Y
    >=   X >= Y

    <>    X <> Y
    !=  X != Y  - same as X <>  Y

    { not implemented }
    in  X in Y
    is  X is Y

Assignment operators:
    :=  X := Y
    +=  X += Y
    -=  X -= Y
    *=  X *= Y
    /=  X /= Y
    %=  X %= Y
    &=  X &= Y
    |=  X |= Y

    { not implemented }
    ^=  X ^= Y

//操作符优先级
Operator priority:

    Assignment operators always evaluete the complete right side
    before assigning the result to the variable on the left side.

    High priority operators:
    * / % div mod

    High priority operators work on the result from the left side
    and a the first element from are right side.

    Low priority operators:
    < > <= >= == != <>

    Low priority operators work on the result from the left side
    and the result from the right side (executed as last command).

//在脚本中编写函数
Writing Functions inside Script:

    FUNCTION $FunctionName command;

    Function result has to be assigned to the $RESULT variable,
    which can also be used to store temporary data until the final result
    has been prepared. Once the function execution completes, only the data
    stored in the $RESULT variable will be passed on as function result.

    FUNCTION $FunctionName $result := command;

    If the function has more than a single command, put it isnide begin/end.
    Here is an example of a function to return a sum from 1 to the passed parameter:

    FUNCTION $test if $params <> nil then
      begin
      $result := 0;
      for $i := 1 to $params.0 do $result += $i;
      end;

    And here is how this simple function can be called:

```

```
    $test(100);

//条件表达形式
Conditional code branching:

    IF condition THEN command;

    IF condition THEN command1 ELSE command2;

//循环
Command Loops:

    REPEAT command-list UNTIL condition;

    WHILE condition DO command;

    FOR $variable := minvalue TO maxvalue DO command;
    FOR $variable := maxvalue DOWNTO minvalue DO command;

   { not implemented }
    FOREACH $variable := enumeration DO command;

//代码块
Code block declaration:

    CODE command

    Code Blocks are "blocks" of execution code which you can assign to local
    script variables or send as parameters to local script function calls.

    When a code block (command with a "code" prefix) is passed to a script function call,
    it will be executed in the scope of the script function.

    When a code block is assigned to a variable (or a variable property),
    each time the variable is evaluated, the code block (assigned to that variable)
    will be executed and the result returned as if the code was called as a function and
 stored into the variable.

    Examples:

    // define a code block for variable $x ...
    $x := code if $y>0 then $y else -$y;

    $y := 15;
    $x; // execute code x" output 15

    $a := code $x; // $a will be returning the value of $x

    // you can pass codeblocks to local function calls ...
    $myFunc(a:code $x);

    // you can pass any expression as a code block ...
    $myFunc(code if $x>0 then $x else $y);



Code block access:

    // To assign a codeblock to a variable, use ...

    $x := code $y;

    @x := 15;//this will assign "15" to variable $y, since it is stored in codeblock x

    $x := 25; // this will assign value "25" to variable $x,
 replacing any prior assignments to $x



NOTE: Function calls can be nested, all operators can be used anywhere inside a script.

Results from any function call, command and variables which are not assigned
to other variables will implicitly become part of the resulting output (HTML/XML).

All variable and function names are case-insensitive (NOT CASE SENSITIVE),
so you can use lowercase, uppercase or a combination of lower and uppercase names.

注意：函数调用可以嵌套，所有操作符都可以在脚本中的任何地方使用。

来自任何未分配的函数调用，命令和变量的结果
```

```
到其他变量将隐式地成为结果输出（HTML / XML）的一部分。

所有变量和函数名称不区分大小写（NOT CASE SENSITIVE），
因此您可以使用小写，大写或小写和大写名称的组合。


//其它一些对象的属性
Request parameters can be read and written using the "Request" variable:

    REQUEST.METHOD      = request method (examples: "GET", "POST", "PUT", ...)
    REQUEST.FILENAME    = request file name (example: "/myfile.rtc" )
    REQUEST.CLOSE       =should the connection be closed after respose was sent?(HTTP/1.0)
    REQUEST.URI     = Request URI (all exept the host)
    REQUEST.URL     = Request URL (host + URI) - read only

    REQUEST.CONTENTTYPE = "CONTENT-TYPE" HTTP reuqest header value
    REQUEST.CONTENTLENGTH   = "CONTENT-LENGTH" HTTP request header value
    REQUEST.HOST        ="HOST" HTTP request header value(example:www.realthinclient.com)
    REQUEST.AGENT       ="USER-AGENT" HTTP request header value(example:Mozila Firefox...)
    REQUEST.REFERER     = "REFERER" HTTP request header value
    REQUEST.FORWARDEDFOR    = "X-FORWARDED-FOR" HTTP request header value


    REQUEST.HEADER      or
    REQUEST.HEADER.TEXT = complete HTTP request header

    REQUEST.HEADER('abc')   or
    REQUEST.HEADER.abc  = value of the HTTP request header with name 'abc'
(example name: "CONTENT-TYPE")

    REQUEST.HEADER.COUNT        = number of HTTP request header variables
    REQUEST.HEADER.NAME( 123 )  =name of HTTP request header at index 123(starting at 0)
    REQUEST.HEADER.VALUE( 123 ) =value of HTTP request header at index 123(starting at 0)


    REQUEST.COOKIE      or
    REQUEST.COOKIE.TEXT = complete request cookie text

    REQUEST.COOKIE('abc')   or
    REQUEST.COOKIE.abc  = value of request cookie parameter with name 'abc'

    REQUEST.COOKIE.DELIMITER    = request cookie parameters delimiter
    REQUEST.COOKIE.COUNT        = number of parameters in a request cookie - read only
    REQUEST.COOKIE.NAME( 123 )
= name of the request cookie parameter at index 123 (starting at 0)
    REQUEST.COOKIE.VALUE( 123 )
= value of the request cookie parameter at index 123 (starting at 0)


    REQUEST.QUERY       or
    REQUEST.QUERY.TEXT  = complete request query text

    REQUEST.QUERY('abc')    or
    REQUEST.QUERY.abc   = value of request query parameter with name 'abc'

    REQUEST.QUERY.DELIMITER     =request query parameter delimiter (examples: ";", "&")
    REQUEST.QUERY.COUNT     = number of query parameters - read only
    REQUEST.QUERY.NAME( 123 )   = name of query parameter at index 123 (starting at 0)
    REQUEST.QUERY.VALUE( 123 )  =value of query parameter at index 123 (starting at 0)


    REQUEST.PARAMS      or
    REQUEST.PARAMS.TEXT = complete input as one string

    REQUEST.PARAMS('abc')   or
    REQUEST.PARAMS.abc  = value of input parameter with name 'abc'

    REQUEST.PARAMS.DELIMITER    = input parameters delimiter
    REQUEST.PARAMS.COUNT        = number of input parameters - read only
    REQUEST.PARAMS.NAME( 123 )= name of input parameter at index 123 (starting at 0)
    REQUEST.PARAMS.VALUE( 123 )= value of input parameter at index 123 (starting at 0)


    REQUEST.INFO('abc') or
    REQUEST.INFO.abc    = access to "Request.Info" variable 'abc'
                  All Request.Info variables can be accessed from Delphi.

    REQUEST('abc')      or
    REQUEST.abc     = Value of the HTTP request header with name "abc"
                  Example: Request("CONTENT-TYPE")
```

```



Response parameters can be read and written using the "Response" variable:


    RESPONSE.STATUS
= response Status Code+Text("200 OK", "404 File Not Found", etc)

    RESPONSE.STATUSCODE or
    RESPONSE.STATUS.CODE    = response Status Code (200, 404, ...)

    RESPONSE.STATUSTEXT or
    RESPONSE.STATUS.TEXT    = response Status Text ("OK", "File Not Found", etc)

    RESPONSE.CONTENTTYPE    = "CONTENT-TYPE" HTTP response header value
    RESPONSE.CONTENTLENGTH  = "CONTENT-LENGTH" HTTP response header value


    RESPONSE.HEADER     or
    RESPONSE.HEADER.TEXT    = complete HTTP response header

    RESPONSE.HEADER('abc')  or
    RESPONSE.HEADER.abc = value of the HTTP response header with name 'abc'
                  (example name: "CONTENT-TYPE")

    RESPONSE.HEADER.COUNT       = number of HTTP response header variables
    RESPONSE.HEADER.NAME( 123 ) = name of HTTP response header at index 123
 (starting at 0)
    RESPONSE.HEADER.VALUE( 123 )    = value of HTTP response header at index 123
 (starting at 0)


    RESPONSE.COOKIE     or
    RESPONSE.COOKIE.TEXT    = complete response cookie text

    RESPONSE.COOKIE('abc')  or
    RESPONSE.COOKIE.abc = value of response cookie parameter with name 'abc'

    RESPONSE.COOKIE.COUNT       =number of parameters in a response cookie-read only
    RESPONSE.COOKIE.NAME( 123 )
= name of response cookie parameter at index 123 (starting at 0)
    RESPONSE.COOKIE.VALUE( 123 )
= value of response cookie parameter at index 123 (starting at 0)
    RESPONSE.COOKIE.DELIMITER   = response cookie parameters delimiter


    RESPONSE('abc')     or
    RESPONSE.abc        = Value of the HTTP response header with name "abc"
                  Example: Response("CONTENT-TYPE")


You can work with sessions by using the "Session" variable:

    SESSION.ID      = Session ID

    SESSION.OPEN        or
    SESSION.OPEN.FWDLOCK    or
    SESSION.OPEN.PRIVATE    = Open and lock a new private user session.
                  This is the default private session type,
                which works with users behind all kinds of Proxy servers.
                  Locks the Session to current user's IP address or to
                  the "X-FORWARDED-FOR" HTTP header (if it exists).

    SESSION.OPEN.PUBLIC = Open and lock a new PUBLIC Session, which should NOT be locked
 to this specific user.
             This Session can be accessed by any user who knows the Session ID.
                  When working with PUBLIC sessions, always use SESSION.LOCK() to
 obrain a lock.

    SESSION.OPEN.IPLOCK = This option will NOT work for users behind Proxy Servers with
 changing IP.
                  Lock the Session to users's IP address,
 ignoring "X-FORWARDED-FOR" HTTP header.

    SESSION.OPEN.SECURE or
    SESSION.OPEN.STRONG or
    SESSION.OPEN.IPFWDLOCK  = This option will NOT work for users behind Proxy Servers with
 changing IP.
                  Opens and locks a new session with a strong and secure link to
 the current user.
                 Lock the Session to users's IP address *and* to the
"X-FORWARDED-FOR" HTTP header.

    SESSION.FIND('abc') or
    SESSION.FIND.abc     = Default method for finding and locking Private user Sessions.
                 Finds a Session with ID "abc" and tries to lock it,
 without waiting.
                 Returns FALSE if Session can not be locked because does
 NOT exist,
                   or if it is locked by anotuer user.
                 Returns TRUE and Locks the Session if Session exists and
                   is NOT currently locked by another user.

    SESSION.HAVE('abc') or
    SESSION.HAVE.abc     = Check if session with ID "abc" exists.
                 Returns TRUE if Session exists (even if locked), but does
 NOT lock the session.

    SESSION.LOCK('abc') or
    SESSION.LOCK.abc     = Lock a PUBLIC Session (which can be shared by multiple users).
                 Find Session with ID "abc" and wait until you can lock it
 for usage.
                 If session exists, it will be locked and TRUE will be returned.
                 If Session does NOT exist
 (or should be closed while waiting for a lock), returns FALSE.

    SESSION.CLOSE        = close current session

    SESSIOn.CLOSE('abc')    or
    SESSIOn.CLOSE.abc    = Find and close session 'abc'

    SESSION.KEEPALIVE    =number of seconds the session should be kept alive when not used
    SESSION.FINALEXPIRE = date/time when the Session has to expire, regardless
 of "SESSION.KEEPALIVE"

    SESSION.EXPIRETIME  = date/time when the Session would expire if not used (read-only)

    SESSION.COUNT       or
    SESSION.COUNT.TOTAL = Total number of currently active sessions (read-only)

    SESSION.COUNT.LOCKED    = Number of currently active and locked sessions (read-only)
    SESSION.COUNT.UNLOCKED  = Number of currently active but not locked sessions (read-only)

    SESSION('abc')  or
    SESSION.abc= access to current sessions variable "abc". Before you can access session
 variables,
               you need to find and lock an existing session using "Session.Find"
(or from a Delphi function),
               or open and lock a new session using "Session.Open". If no session
 is locked,
               all variables will return NULL on read access, but an exception will
 be raised on write access.


Data received through as Query parameters (part of the URL) can be accessed
by using the REQUEST.QUERY variable, or by using the QUERY variable
 (without the "REQUEST." prefix):

    QUERY          or
    QUERY.TEXT     = complete request query text

    QUERY.COUNT     = number of query parameters - read only
    QUERY.DELIMITER     = request query parameter delimiter (examples: ";", "&")
    QUERY.NAME( 123 )   = name of query parameter at index 123 (starting at 0)
    QUERY.VALUE( 123 )  = value of query parameter at index 123 (starting at 0)

    QUERY('abc')        or
    QUERY.abc      = value of request query parameter with name 'abc'


Data received from a FORM POST when using INPUT elements inside HTML can be accessed
 through the INPUT variable:

    INPUT          or
    INPUT.TEXT     = complete input as one string

    INPUT.COUNT     = number of input parameters - read only
    INPUT.DELIMITER     = input parameters delimiter
    INPUT.NAME( 123 )   = name of input parameter at index 123 (starting at 0)
    INPUT.VALUE( 123 )  = value of input parameter at index 123 (starting at 0)

    INPUT('abc')        or
    INPUT.abc       = value of input parameter with name 'abc'


Basic Client information can be obrained by using the CLIENT variable (read-only):

    CLIENT      = Client's IP address and Port in the form "IP:Port"
 (example: "196.45.32.112:1604")

    CLIENT.IP   or
    CLIENT.ADDR or
    CLIENT.ADDRESS  = Client's IP address

    CLIENT.PORT = Client's Port number (as string)

    CLIENT.COUNT    =Number of currently conencted clients(total client connection count)


Basic Server information can be obtained by using the SERVER variable (read-only):

    SERVER      = Server's IP address and Port in the form "IP:Port"
(example: "127.0.0.1:80")

    SERVER.IP   or
    SERVER.ADDR or
    SERVER.ADDRESS  = Our Server's Local IP address

    SERVER.PORT = Out Server's Port number (as string)
              Standard HTTP port = 80, Standard HTTPS (SSL) port = 443
```

RTC脚本模型课堂 - 调用https接口

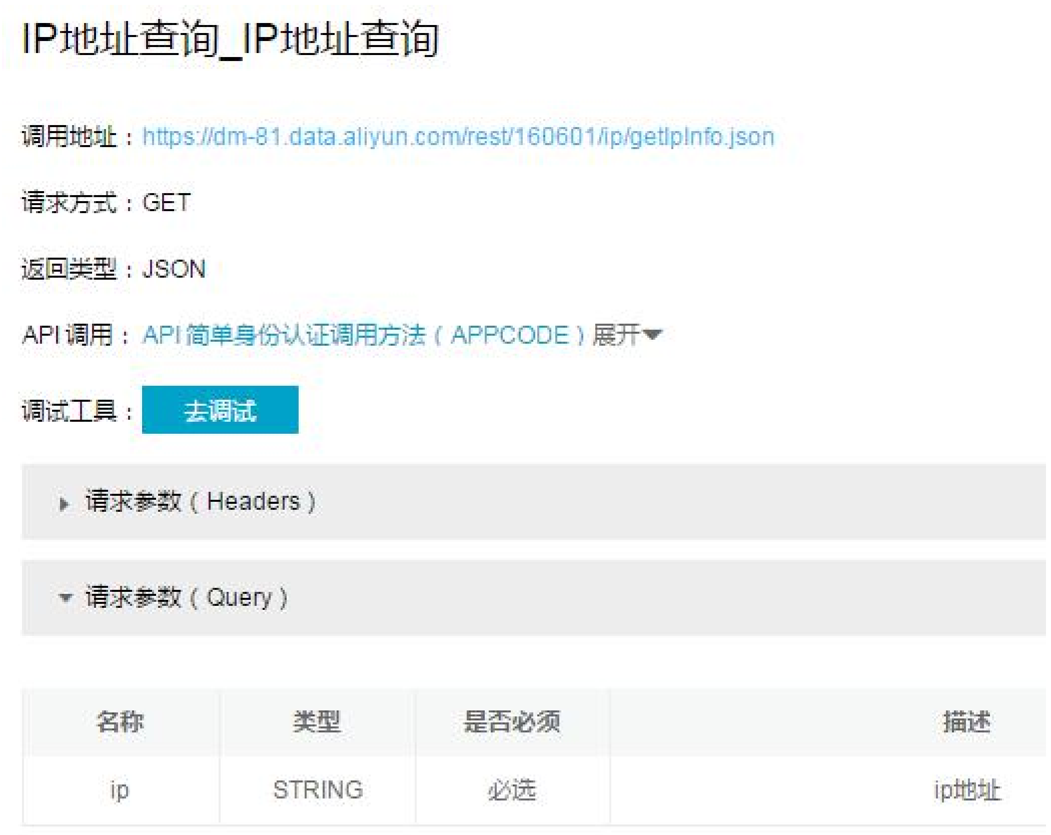
2017-01-04 10:47:01 39 0

阿里云市场api专区有大量的收费或免费的api接口可用，对我们的应用开发提供了便利。

今天我们来展示一下脚本模型中如何调用阿里云市场api之中由阿里云计算有限公司提供的ip地址查询api的使用：

首先，登录阿里云，在数据及API市场中找到免费的这个api，购买它（虽然是免费的，但还是要先购买，费用为0元），然后就可以在控制台中找到所需的APPCODE了；

然后我们看一下它的接口形式，如下：


IP地址查询_IP地址查询

调用地址：https://dm-81.data.aliyun.com/rest/160601/ip/getIpInfo.json

请求方式：GET

返回类型：JSON

API 调用：API 简单身份认证调用方法（APPCODE）展开

调试工具：去调试

▸ 请求参数（Headers）

▾ 请求参数（Query）

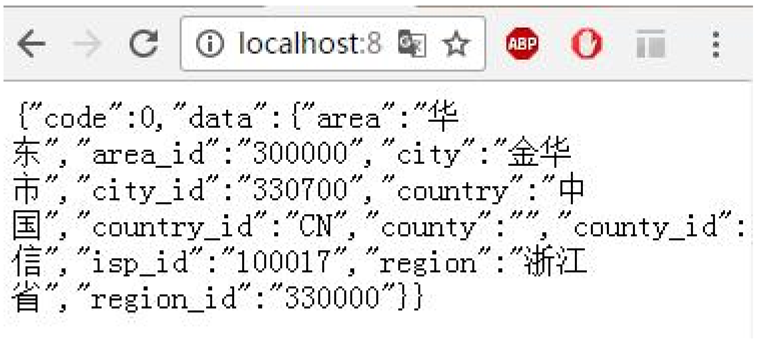
名称	类型	是否必须	描述
ip	STRING	必选	ip地址

接口说明中没有写Headers的相关描述，但实际上所有阿里云市场中的api都是需要在Headers中添加Authorization项目的，那么，我们在脚本中输入如下代码：

```
FnHTTPS('https://dm-81.data.aliyun.com/rest/160601/ip/getIpInfo.json','ip=60.191.244.5',
'Authorization='+FnURLencode('APPCODE 这里是你购买并获得的appcode',65001,true),
'get',10000,65001,65001,false,false);
```

注意，以上代码需要嵌套在<??>中，FnUrlencode是将appcode编码一下，因为中间有空格。

ok，在浏览器中测试一下效果，如下：



```
{"code":0,"data":{"area":"华
东","area_id":"300000","city":"金华
市","city_id":"330700","country":"中
国","country_id":"CN","county":"","county_id":
信","isp_id":"100017","region":"浙江
省","region_id":"330000"}}
```

RTC脚本模型课堂 - http跨域请求

2017-01-04 10:51:27 44 0

在web开发或者接口开发中，我们常常需要进行一些http请求的转发，也就是js中的跨域请求或域内请求，比如在http://www.offeu.com内去请求http://www.ykipa.com的接口，在js中我们是需要在被请求端设置一些参数，以允许跨域请求，在脚本模型中则不需要那么麻烦，直接调用内置函数FnHTTP或者FnHTTPS就可以了，这里，我推荐使用FnHTTPS，它是基于delphi xe8以后集成的httpclient来实现的，支持https的请求。

```
<?
(*
    0:vUrl.asText
    1:vParam.asText 内容需要url编码
    2:vHeaders.asText 内容需要url编码
    3:vMethod.asText='GET'
    4:vTimeout.asInteger=0
    5:RequestCodeing.asInteger=65001
    6:vResponseCoding.asInteger=65001
    7:vEncodeing.asBoolean=true
    8:vDecoding.asBoolean=false
*)
//获取指定编码文件内容
FnHTTPS('http://offeu.com/gbk.txt','','',
    'GET',10000,936,936,false,false);' >>> '
FnHTTPS('http://offeu.com/gbk.txt','','',
    'GET',10000,936,936,false,true);'<br>'
FnHTTPS('http://offeu.com/utf8.txt','','',
    'GET',10000,65001,65001,false,false);' >>> '
FnHTTPS('http://offeu.com/utf8.txt','','',
    'GET',10000,65001,65001,false,true);'<br>'
//传utf8过去，再取回来并解码
FnHTTPS('http://offeu.com/ask/echo.api','id='+FnUrlencode('测试',65001,true),'',
    'POST',10000,65001,65001,false,false);' >>> '
FnHTTPS('http://offeu.com/ask/echo.api','id=测试','',
    'POST',10000,65001,65001,true,true);'<br>'
//传gbk过去，再取回来并解码
FnHTTPS('http://offeu.com/ask/echo.api','id='+FnUrlencode('测试',936,true),'',
    'POST',10000,65001,65001,false,false);' >>> '
FnUrlencode(FnHTTPS('http://offeu.com/ask/echo.api','id=测试','',
    'POST',10000,936,65001,true,false),936,false);'<br>'
//阿里去市场api测试，ssl协议测试
$x:=FnJsonToRecord(FnHTTPS('http://offeu.com/ask/getip.api','','',
    'GET',5000,65001,65001,false,false));
FnHTTPS('https://dm-81.data.aliyun.com/rest/160601/ip/getIpInfo.json'
,'ip='+$x('ip'),//如果脚本在服务器上，直接CLIENT.IP就行，不用转一手
'Authorization='+FnURLencode('APPCODE 你自己申请的appcode',65001,true),
'get',10000,65001,65001,false,false);'<br>'
?>
```

之前还发过一篇文章专门介绍如何调用阿里云API市场中的IP查询接口，就是https的，大家可以去看一下，步骤也比较简单。

个人认为，http请求转发这块是比较常用的功能，所以优先进行介绍，这样就可以通过脚本实现很多功能，比如给你的手机app提供统一的后台接口，去转接天气预报、短信发送、条码生成等等三方接口。

RTC脚本模型课堂 - 正则表达式

2017-01-04 11:01:00 45 0

在开发过程中，特别是数据处理这块，正则表达式是非常方便和实用的东西，所以在脚本模型中也提供了相应的函数，分别是FnRegex和FnRegexCheck，调用示例如下：

```
<?//格式：FnRegex(内容,正则表达式);
$x:=FnRegex('dddaxxoobeee,eeeaooxxbddd','(?<=a).*?(?=b)');
if $x.count=0 then
'数据为空'
else
for $i:=0 to $x.count-1 do
begin
  '第' $i+1 '行:' $x($i);'<br>'
end;
?>
<?//格式：FnRegexCheck(内容,正则表达式);
$x:=FnRegexCheck('emailto:star5d@hotmail.com',
    '([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,5})+');
if $x='success.' then
'校验成功'
else if $x='fail.' then
'校验失败'
else if $x='Parameter error.' then
'参数错误';
?>
```

但要注意的是，delphi中的正则表达式和javascript中的，或者其它语言中的正则表达式还是有一点点区别的，所以，在使用中需要注意，建议在接口api开发过程中，使用delphi的语法，在web网站开发时，直接用javascript的语法就可以了。

RTC脚本模型课堂 - URLencode

2017-01-04 11:06:42 37 0

在web开发时，urlencode是很常用的，处理字符串编码的功能，比如我们在向某个接口提交数据时，一般都需要对这些数据进行编码，以防止在传输过程中出现乱码的情况。

```
<?//格式：FnURLencode(内容,936/65001,true/false);
'UTF-8编码：'FnURLencode('测试',65001,true);
'<br>GBK编码：'FnURLencode('测试',936,true);
'<br>UTF-8解码：'FnURLencode('%e6%b5%8b%e8%af%95',65001,false);
'<br>GBK解码：'FnURLencode('%b2%e2%ca%d4',936,false);
?>
```

第三个参数为布尔型，为真则表示对字符串内容进行编码操作，为假则表示对字符串进行解码操作；第二个参数则是编码类型，一般65001表示UTF-8，936表示GBK，一般也就使用UTF-8为主，但针对一些比较早期的接口，还是需要GBK支持的。

RTC脚本模型课堂 - Base64相关

2017-01-04 11:11:19 36 0

Base64编码，我想大家应该很熟悉了，早期的电子邮件编码，现在的网页图片编码等，都采用了base64编码，来对字符串或者二进制数据流进行转码，以便于传输，在脚本模型中，分别提供了以下几个内置函数，以支持这些操作：

```
base64('123456',true/false);              |<?base64('123456',true);'|'base64('MTIzNDU2',false);?>
<?
$f:=FnBase64toFile('iVBORw0KGgoAAAANSUhEUgAAABMAAAARCAYAAAA/mJfHAAACz0lEQVQ4jXVUz2sTQRh9s7M/kiba'
+'1KZSCq0HCQgteLMexINUL6KCIngQpFLBg/+A3kTPIiheFJGi4klPKl6sF7G11Wqp1FKkbWKsSW2y'
+'TTfNZrM74+xOzKatHfKYH+R7+77vfTOkWq1ybBo/F6cwOfM8M8zNvYRWyqJSKYDEDUPED/CDXT3HseZ'
+'SzfR2Z1qxKyWqiCO4zTIOOf4/GEYlrmAzPc3yGemARcBSfO8vs4Rj7RgYOgu9h85F8QWijZU27Yb'
+'7F8/DiOqE2Smn8AuFbDT0AG1TkTrswJEwRGjHibuXwb1XPQdPQ/PZSC5XC5Ql14j4r5DempR3DK'
+'pgzcBopQGKdKsHZFYodvjMCN7JHKGPOw9OMVUJ4FdcuIUk3kjG3hi0wYFIpYcMIx9/g6egYfSLKV'
+'pS9QaxlU1+YRoSIvJtPZAiJBFYKdqgq1vrdn3qG4lIVaqVRQ+j0GpZKDHhQoDArQPIj0ICZS9Ml0'
+'Rf6BR2NIT45IMtfKIOJY4uO0bmsTGlZLcM93UhNkGgxG5LmhYjE9J8k8uwziKfKzPhjCddMZF4gI'
+'NSpT0aoaUF0i0xfOUtsRparV4LEdcKs+mbLROQ8b9sT1U6SiGBQaMUL1jjAx2QlF13VoiT5oXg06'
+'0SWUOshGaAK2Q5EUNQITPcj1YGarZbB9/VC7urrgJk+gmn+KWEtcqqj9p7/qZy2EoiNiyDMmla1F'
+'dyO2txdqKiXvV8EcQlv6hci/LQzeDm5oDlv+BX7hmtiSoHzBaO2/gqKbFPl7MoV/4JuApvVaBWbq'
+'ABIDp0BVJSRTRLPGTz+EWRa9ZllhANuGaLmAwq4eJK7eEs4IhwUZYYwFgv0XI5AtLu7q69vQJ54j'
+'FmsXvdAeumwL1fk8rKIF9+QgEmcvNtpwdCwL8mdFvCd8y5MmrpaJ9fGXILOfQMyiICTg8Q6g9yBi'
+'h45B8x2tD/8tu3NvHH8B9EZlK6ozLtkAAAAASUVORK5CYII=',formatdatetime('yyyyMMddhhmmsszzz".jpg"',now));
if $f<>'' then begin
'文件保存成功，文件名为：'$f;' <img src="'$f'">'
end else begin
'文件保存失败。';
end;
?>
```

FnBase64toFile的应用场景，比如手机app端提交一张图片上来，服务端接收并保存成图片，然后返回可访问的路径，比如app中上传自定义人物头像，上传照片等，当然也可以上传其它类型的文件。关于文件上传这块，脚本模型还提供了单独的接口，请参考upfile.html这个演示。

RTC脚本模型课堂 - 同名函数

2017-01-04 11:18:48 43 0

脚本模型中提供了一些基本函数，和delphi中的一些函数是同名的，用法也一样，比如copy、trim等，部分如下，其它请参考基本语法那篇文章和文档目录下的部分说明.txt

```
formatdatetime('yyyy-MM-DD hh:mm:ss.zzz',now);
|<?formatdatetime('yyyy-MM-DD hh:mm:ss.zzz',now);?>
formatfloat('#,##0.#0',555.55);
|<?formatfloat('#,##0.#0',555.55);?>
copy('abcdefg',pos('e','abcdefg'),2);
|<?copy('abcdefg',pos('e','abcdefg'),2);?>
length('12345');
|<?length('12345');?>
trim(' abc ');
|<?trim(' abc ');?>
md5('123456');
|<?md5('123456');?>
base64('123456',true/false);
|<?base64('123456',true);'|'base64('MTIzNDU2',false);?>
StrToFloat('55.55');
|<?StrToFloat('55.55');?>
Trunc(55.55);
|<?Trunc(55.55);?>
```

base64在前一篇文章有专门介绍，base64+md5+copy函数，可以拿来生成一些加密内容，比如网站后台的用户登录的密码，这样就会比较安全，从而避免密码明码保存，泄漏了产生损失的尴尬。

RTC脚本模型课堂 - 数据库相关

2017-01-04 11:26:18 52 0

不管是网站，还是接口，都离不开数据展示或处理的操作，脚本模型采用的是Delphi中的FireDAC组件来连接和操作数据库的，目前脚本模型中只添加了msaccess、mssql和oracle的支持，当然，添加其它数据库的支持，也是比较简单的事，不过要注意的是，FireDAC并不会帮你处理数据库连接所需的驱动，所以你连接对应的数据库时，需要自己安装驱动，比如在win64位环境下，连接msaccess需要安装对应的驱动程序，比如连接oracle，也需要自己安装oracle的客户端或者带上那几个dll动态库文件。

目前提供了以下操作数据库的函数，分别为FnRunSQL、FnRunSQLtoJSON、FnExecSQL和FnRunSQLtoFile，定义及示例如下：

```
<?//执行SQL语句并返回结果（JSON）
//会根据参数个数，读取和设置参数，比如3个参数，第一个为sql语句，2为p1，3为p2
FnRunSQLtoJSON('select * from tuser where fuser<>:p1','你好');
?>
<?//执行SQL语句并返回结果（JSON）需要参数sql
FnRunSQLtoJSON(FnUrlencode(REQUEST.QUERY('sql'),65001,false));
?>
<?//执行SQL语句并返回结果（DataSet）
//会根据参数个数，读取和设置参数，比如3个参数，第一个为sql语句，2为p1，3为p2
$x:=fnrunsql('select * from tuser where fuser<>:p1','你好');
//fid fuser fpwd fname
//1 admin admin 管理员
//2 test test 测试
if $x.EMPTY then
'没有可用的数据'
else
begin
$x.first;
while not $x.eof do
  begin
  '第' $x.row '行:'
  for $i:=0 to $x.fieldcount-1 do
    begin
    $x($i);
    if $i<$x.fieldcount-1 then ', ';
    end;
   '<br>'
  $x.next;
  end;
end;
?>
<?//执行SQL语句并返回结果（保存成文件，并返回下载链接）
//会根据参数个数，读取和设置参数，比如3个参数，第一个为sql语句，2为p1，3为p2
FnRunSQLtoFile('select * from tuser where fuser<>:p1','你好');
?>

原始数据:
<?//执行SQL语句并返回结果（DataSet）
$x:=fnrunsql('select * from tuser');
//fid fuser fpwd fname fage
//1 admin admin 管理员 18
//2 test test 测试 3
if $x.EMPTY then
'没有可用的数据'
else
begin
$x.first;
while not $x.eof do
  begin
  '第' $x.row '行:'
  for $i:=0 to $x.fieldcount-1 do
    begin
    $x($i);
    if $i<$x.fieldcount-1 then ', ';
    end;
   '<br>'
  $x.next;
  end;
```

```
end;
?>
处理结果:
<?//执行SQL语句并返回处理结果（不返回数据集）
//会根据参数个数，读取和设置参数，比如3个参数，第一个为sql语句，2为p1，3为p2
fnexecsql('update tuser set fpwd=:p1,fage=:p3 where fuser=:p2','xxoo','test',88);
?>
<br>修改后的数据：<br>
<?//执行SQL语句并返回结果（DataSet）
$x:=fnrunsql('select * from tuser');
//fid fuser fpwd fname
//1 admin admin 管理员
//2 test test 测试
if $x.EMPTY then
'没有可用的数据'
else
begin
$x.first;
while not $x.eof do
  begin
  '第' $x.row '行:'
  for $i:=0 to $x.fieldcount-1 do
    begin
    $x($i);
    if $i<$x.fieldcount-1 then ', ';
    end;
   '<br>'
  $x.next;
  end;
end;
?>
处理结果:
<?//执行SQL语句并返回处理结果（不返回数据集）
//会根据参数个数，读取和设置参数，比如3个参数，第一个为sql语句，2为p1，3为p2
fnexecsql('update tuser set fpwd=:p1,fage=:p3 where fuser=:p2','test','test',3);
?>
修改后的数据:
<?//执行SQL语句并返回结果（DataSet）
$x:=fnrunsql('select * from tuser');
//fid fuser fpwd fname fage
//1 admin admin 管理员 18
//2 test test 测试 3
if $x.EMPTY then
'没有可用的数据'
else
begin
$x.first;
while not $x.eof do
  begin
  '第' $x.row '行:'
  for $i:=0 to $x.fieldcount-1 do
    begin
    $x($i);
    if $i<$x.fieldcount-1 then ', ';
    end;
   '<br>'
  $x.next;
  end;
end;
?>
```

在使用前，您需要先对SrvConfig.ini进行配置，以连接对应的数据库，才可以正常使用以上的函数。

比如连接msaccess

```
;数据库:MSACCESS/MSSQL/MSSQL2000/ORACLE
DBDriver=MSACCESS
DBFile=.\db.mdb
```

比如连接mssql2000，注意，连接mssql2000以上版本，应该是mssql

```
;数据库:MSACCESS/MSSQL/MSSQL2000/ORACLE
DBDriver=MSSQL2000
DBUser=sa
DBPwd=密码
DBAddr=127.0.0.1,1433
DBFile=master
```

RTC脚本模型课堂 - session的使用

2017-01-04 11:55:26 58 0

Session:在计算机中，尤其是在网络应用中，称为“会话控制”。Session 对象存储特定用户会话所需的属性及配置信息。这样，当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。当用户请求来自应用程序的 Web 页时，如果该用户还没有会话，则 Web 服务器将自动创建一个 Session 对象。当会话过期或被放弃后，服务器将终止该会话。Session 对象最常见的一个用法就是存储用户的首选项。例如，如果用户指明不喜欢查看图形，就可以将该信息存储在 Session 对象中。在脚本模型中，您需要建立session对象，比如在用户登录成功后，新建这个用户的session对象，然后把一些参数绑定上去，比如：

```
session.open;
session.keepalive:=1800;//半小时
session('ip'):=client.ip;//绑定ip
```

获取sessionid则是session.id，脚本中提供了session.find和session.have两个操作函数，find相当于查找这个sessionid，如果找到就重置它的超时时间计时，如果找不到则返回找不到；have则仅仅是查找这个session对象有没有被释放，而不会去重置它的超时时间计时。

session对象可以绑定您所需要缓存的数据，比如绑定ip啊，绑定用户名密码啊，绑定用户的一些属性啊什么的，或者仅仅是缓存一下全局的变量等。

注意：session对象一但超时被释放，缓存的数据也就丢失了。

RTC脚本模型课堂 - json对象

2017-01-04 12:56:46 52 0

在网络通信中，json是比较普及的数据交换格式了，delphi中有很多这方面的支持，比如QDAC的QJSON和常见的superobject等，rtc自己也有json解析的支持，虽然qjson很好，但我还是继续用rtc的json，这个也可以说是习惯问题吧。

```
FnJsonToRecord('{"name": "姓名","sex": "性别"}');
|<?$x:=FnJsonToRecord('{"name": "姓名","sex": "性别"}');'all:'$x;?>
|<?'count:'$x.count;?>
|<?'name:'$x('name');?>
|<?'sex:'$x('sex');?>

FnRecordToJson(FnJsonToRecord('{"name": "姓名","sex": "性别"}'));
|<?FnUnicodeToAnsi(FnRecordToJson(FnJsonToRecord('{"name": "姓名","sex": "性别"}')));?>

ps:这里引用了FnUnicodeToAnsi来显示中文
```

脚本模型中我提供了两个函数，分别是将json转换成记录集和记录集转换成json，比如FnJsonToRecord转换以后，就可以直接通过$x.count的形式去访问json中的对象了，是不是比较方便？

RTC脚本模型课堂 - 套用html5模板

2017-01-05 08:48:53 44 0

脚本模型除了开发API接口外，更适合制作网站，对于我们delphi开发人员而言，网站开发这块明显是弱项；那么，我们如何快速的进入网站开发模式呢？首先，我们应该从网页模板开始着手！

百度一下“网页模板”或者“html5模板”，会有很多免费的模板可以下载使用，我们可以下载一个html5模板；这里要说一下一个概念，就是响应式web设计：页面的设计与开发应当根据用户行为以及设备环境(系统平台、屏幕尺寸、屏幕定向等)进行相应的响应和调整。具体的实践方式由多方面组成，包括弹性网格和布局、图片、CSS media query的使用等。无论用户正在使用笔记本还是iPad，我们的页面都应该能够自动切换分辨率、图片尺寸及相关脚本功能等，以适应不同设备；换句话说，页面应该有能力去自动响应用户的设备环境。响应式网页设计就是一个网站能够兼容多个终端——而不是为每个终端做一个特定的版本。

在脚本模型的包里，已经有一个html5响应式网页的模板，做为演示，它具备的功能是根据用户访问时填写的参数，切换不同的语言，分别是简体中文和英文。这里我们又该说明一下目录设定，在bin目录下，分别有script和www目录，script目录下保存的是我们的脚本文件，而www目录下则放的是一些静态页或者资源文件，也就是可以直接通过浏览器访问到的资源。比如一个静态页index.html，你把它全部复制到www目录下，就可以通过http://localhost:8833/index.html 访问到了；而脚本目录下的文件，我们则需要通过http://localhost:8833/www 或者http://localhost:8833/ask 访问到。

用文本编辑器打开bin\script\index.html文件，找到第39行，你会看到这样一句：

```
<?if request.query('lang')='chs' then '首页' else 'Home';?>
```

这是什么意思呢？request.query('lang')就是指获取用户提交的参数lang，如果参数值等于chs，则输出“首页”，否则输出“Home”，这样能看明白吧？是不是很简单就实现了根据参数输出不同的内容了？其实，不管是自行制作网站，或者套用网站模板，道理是一样的，只要在页面中嵌入脚本，处理和展示数据就可以了。

来，赶紧去下载一个html5网页模板来自己套用试试，你会发现，制作一个网站，也并不是那么的难。

当然，这只是一个开始，后续深入，还是需要学习一下bootstrap、jquery和css的，如果贵公司有相应的web前端开发人员，那你只要搞明白如何嵌入脚本就行了，会更加简单省力。

RTC脚本模型课堂 - AJAX介绍

2017-01-05 09:11:45 66 0

什么是AJAX？AJAX即“Asynchronous Javascript And XML”（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术。AJAX = 异步 JavaScript和XML（标准通用标记语言的子集）。AJAX 是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

比如，我们制作一个登录页面，当用户输入错误的密码时，在当前页面就要进行提醒，而不是整个页面切换后再提示再返回原来的登录页面，这样的用户体验是完全不同的。

AJAX基本用法，可以查看一下Bin\Script\AJAX案例 目录下的AJAX.api和getip.api两个文件，AJAX.api负责调用getip.api并返回数据，在当前页。咱们来分析一下它的代码：

```
<html>
<head>
<script type="text/javascript">
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","getIP.api",true);
xmlhttp.send();
}
</script>
</head>
<body>

<button type="button" onclick="loadXMLDoc()">请求数据</button>
<div id="myDiv"></div>

</body>
</html>
```

代码中<div id="myDiv">是表示这个div用来显示返回值，上面js函数loadXMLDoc()中的document.getElementById("myDiv").innerHTML就是找到这个div并设置其内容的处理；而xmlhttp.open("GET","getIP.api",true)则是进行http get操作，访问getIP.api，这样是不是大致能看明白了？

这个是AJAX的基本用法，但我们可以使用jquery来更加简单的使用它，在jquery中封装了几个方法，分别是$.get $.post 和 $.ajax，比如$.get，这样用：

```
$.get("网址?参数名=参数值&参数名2=参数值2",
    function(result){
      alert(result);
    }
  );
```

是不是很简单？但要注意，ajax是有跨域访问的限制的，如果你想访问非本域的地址，就需要在被访问者那里定义一些属性，当然也可以使用脚本中的fnHTTPS函数，那就没这个限制了。

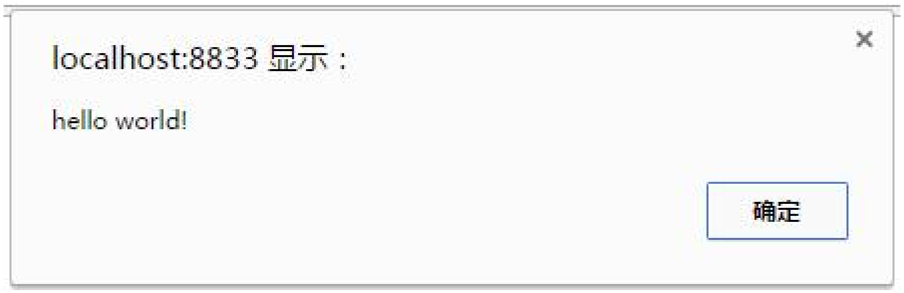
RTC脚本模型课堂 - ShowMessage

2017-01-07 10:26:51 99+ 0

ShowMessage对delphi开发人员而言，是个非常熟悉的玩意，常常需要在软件上做一些合适的提醒，以达到更好的用户体验。今天我们来介绍一下网站里的提示框，也就是JavaSciprt中的alert，同时也介绍一下如何使用三方的js插件，来达到更好的效果。

首先，我们来看一下JavaScript的alert代码和效果，如下：

```
<script type="text/javascript">
  alert('hello world!');
</script>
```


localhost:8833 显示：

hello world!

这就是js基本的提示框样式了，比较简单，我们一般在调试时用来输出一些数据用用，正式场合一般不用，所以就这样简单的盖过吧。

推荐一个三方的js插件，名字叫sweetalert，官网地址是 http://t4t5.github.io/sweetalert/ (http://t4t5.github.io/sweetalert/) github地址是 https://github.com/t4t5/sweetalert (https://github.com/t4t5/sweetalert) ，简单介绍一下它的用法和效果，如下：

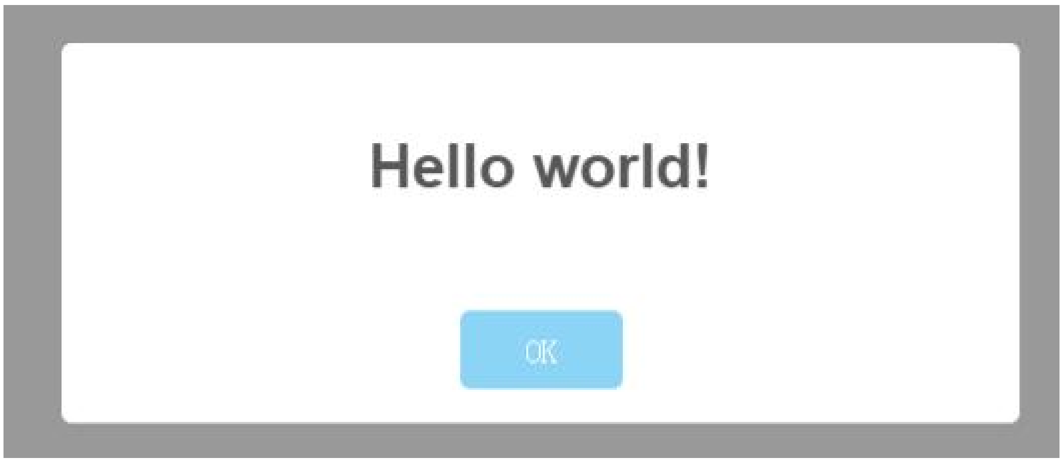
首先，讲下如何安装sweetalert，非常简单，只要在页中引用它的js和css就可以了，代码如下：

```
<script src="//cdn.bootcss.com/sweetalert/1.1.3/sweetalert-dev.min.js"></script>
<link href="//cdn.bootcss.com/sweetalert/1.1.3/sweetalert.min.css" rel="stylesheet">
```

我这里是引用的cdn上的链接，你也可以去官网下载到本地，引用本地文件。然后就可以开始调用它了，代码如下：

```
<script type="text/javascript">
   swal("Hello world!");
</script>
```

也就是把alert直接替换成swal就行了，是不是很简单？效果如下：

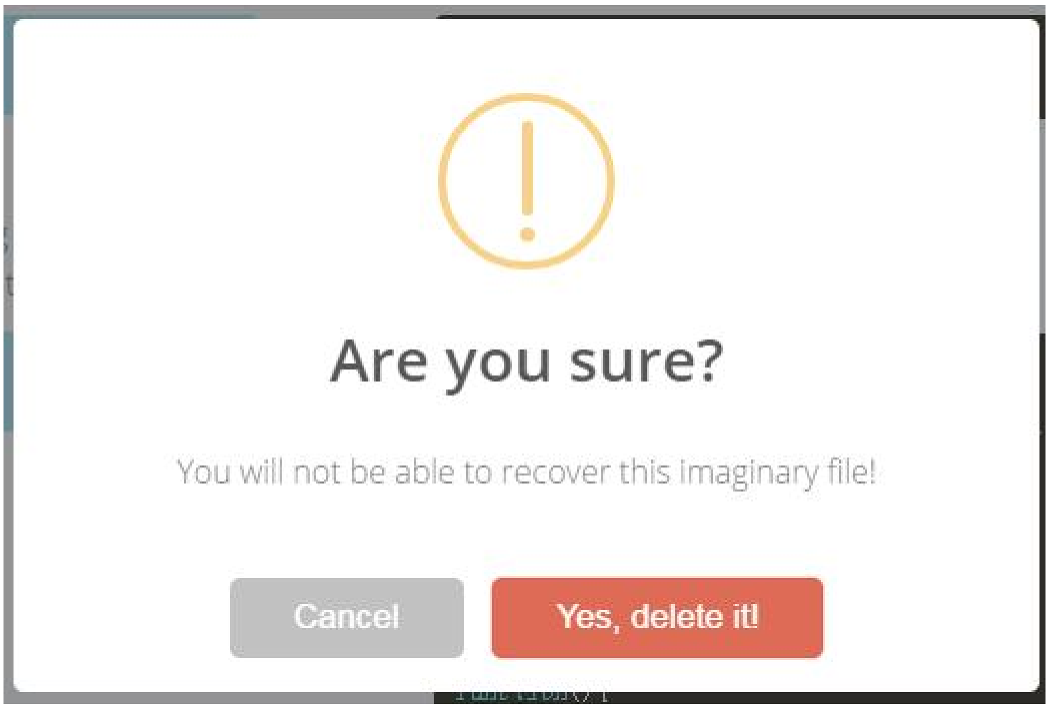

Hello world!

我们再来介绍一下它的其它几种用法，效果图就不贴了，只贴代码：

```
//用于提示出错信息
swal({
  <a href="title:"","这是个出错提示">title:"","这是个出错提示</a>! ",
  type:"error",
  confirmButtonColor:"#428bca",
  confirmButtonText:"关闭"
});
//用于提示完成信息，注意timer参数，是指这个时间后自动关闭提示框
swal({
  <a href="title:"","恭喜！操作完成！",timer: 5000,type:"success"">title:"",
  "恭喜！操作完成！",
  ti</a><a href="mer: 5000,t">mer: 5000,
  t</a><a href="title:"","恭喜！操作完成！",timer: 5000,type:"success"">ype:"success"</a>,
  confirmButtonColor:"#428bca",
  confirmButtonText:"关闭"
});
```

当然还有询问提示框等应用方式，比如询问是否进行删除操作，代码和效果如下：

```
swal({
  title: "Are you sure?",
  text: "You will not be able to recover this imaginary file!",
  type: "warning",
  showCancelButton: true,
  confirmButtonColor: "#DD6B55",
  confirmButtonText: "Yes, delete it!",
  closeOnConfirm: false
},
function(){
  swal("Deleted!", "Your imaginary file has been deleted.", "success");
});
```



其它的一些用法，请参考官网的介绍，今天是否学会如何为您的网站或者应用页增加提示框了呢？网上还有很多类似的js插件，可以像delphi的vcl一样帮您解决很多问题，增加开发效率。

RTC脚本模型课堂 - 打印网页元素

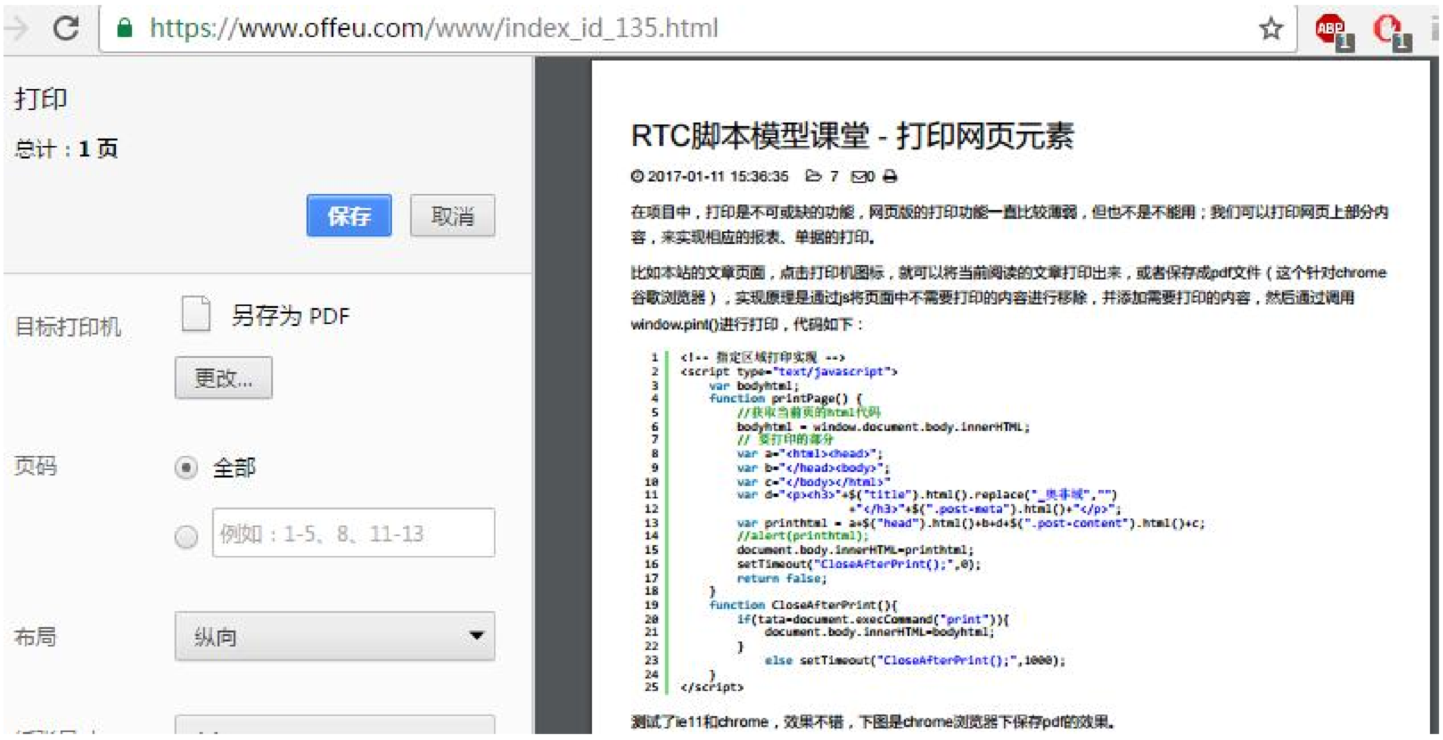
2017-01-11 15:36:35 8 0

在项目中，打印是不可或缺的功能，网页版的打印功能一直比较薄弱，但也不是不能用；我们可以打印网页上部分内容，来实现相应的报表、单据的打印。

比如本站的文章页面，点击打印机图标，就可以将当前阅读的文章打印出来，或者保存成pdf文件（这个针对chrome谷歌浏览器），实现原理是通过js将页面中不需要打印的内容进行移除，并添加需要打印的内容，然后通过调用window.pint()进行打印，代码如下：

```
<!-- 指定区域打印实现 -->
<script type="text/javascript">
    var bodyhtml;
    function printPage() {
        //获取当前页的html代码
        bodyhtml = window.document.body.innerHTML;
        // 要打印的部分
        var a="<html><head>";
        var b="</head><body>";
        var c="</body></html>"
        var d="<p><h3>"+$("title").html().replace("_奥非域","")
                +"</h3>"+$(".post-meta").html()+"</p>";
        var printhtml = a+$("head").html()+b+d+$(".post-content").html()+c;
        //alert(printhtml);
        document.body.innerHTML=printhtml;
        setTimeout("CloseAfterPrint();",0);
        return false;
    }
    function CloseAfterPrint(){
        if(tata=document.execCommand("print")){
            document.body.innerHTML=bodyhtml;
        }
            else setTimeout("CloseAfterPrint();",1000);
    }
</script>
```

测试了ie11和chrome，效果不错，下图是chrome浏览器下保存pdf的效果。



https://www.offeu.com/www/index_id_135.html

打印

总计：1 页

目标打印机 另存为 PDF

页码 ● 全部

例如：1-5、8、11-13

布局 纵向

RTC脚本模型课堂 - 打印网页元素

2017-01-11 15:36:35 7 0

在项目中，打印是不可或缺的功能，网页版的打印功能一直比较薄弱，但也不是不能用；我们可以打印网页上部分内容，来实现相应的报表、单据的打印。

比如本站的文章页面，点击打印机图标，就可以将当前阅读的文章打印出来，或者保存成pdf文件（这个针对chrome谷歌浏览器），实现原理是通过js将页面中不需要打印的内容进行移除，并添加需要打印的内容，然后通过调用window.pint()进行打印，代码如下：

```
<!-- 指定区域打印实现 -->
<script type="text/javascript">
    var bodyhtml;
    function printPage() {
        //获取当前页的html代码
        bodyhtml = window.document.body.innerHTML;
        // 要打印的部分
        var a="<html><head>";
        var b="</head><body>";
        var c="</body></html>"
        var d="<p><h3>"+$("title").html().replace("_奥非域","")
                +"</h3>"+$(".post-meta").html()+"</p>";
        var printhtml = a+$("head").html()+b+d+$(".post-content").html()+c;
        //alert(printhtml);
        document.body.innerHTML=printhtml;
        setTimeout("CloseAfterPrint();",0);
        return false;
    }
    function CloseAfterPrint(){
        if(tata=document.execCommand("print")){
            document.body.innerHTML=bodyhtml;
        }
            else setTimeout("CloseAfterPrint();",1000);
    }
</script>
```

测试了ie11和chrome，效果不错，下图是chrome浏览器下保存pdf的效果。