El objeto oCGI

El objeto oCGI sera el encargado de realizar las comunicaciones entre Apache y nuestra Aplicación, al instanciarlo con `oCgi := TCgi():New()` inicializa la conexión con Apache y se trae a nuestra aplicación todos los parámetros de la llamada realizada desde el navegador (por POST, GET o como URL Amigable), los cookies activos en la sesión y las variables de Apache.

Como el oCGI es un MEMVAR que esta definido en Nefele.ch puede hacerse referencia a el desde cualquier parte de nuestra aplicación.

Propiedades

Nombre	Init	Descripción	Versión
aCookies		Array de las cookies recibidas.	0.1
aParams		Array de los parámetros recibidos.	0.1
aSharedParams	{}	aParams que serán añadidos a todos los controles que realicen llamas al CGI	0.3
aRoutes		Array con las rutas envidas como URL amigable.	0.1
cFunction	“FUNCTION”	Nombre de la propiedad que recibimos que contendrá la función a llamar de nuestro CGI, todos controles que generan llamadas le asignaran este nombre a la propiedad.	0.1
cFunctEndCookie		Función a ejecutar cuando se recibe un cookie caducado.	0.1
cMainFunction	“MainFunction”	Método del oCGI que sera llamada por defecto si no se indica otra	0.1
cPathTmp	“.\tmp”	Carpeta temporal que utilizara nuestro CGI, por ejemplo para guardar los ficheros adjuntos	0.1
cSameSitePolicy	“lax”	Protección de cookies (info)	0.3
lGetData	.t.	Admitimos parámetros pasados por GET	0.1
lGetFunction	.t.	Admitimos funciones pasadas por GET	0.1
lCheckNavigator	.t.	Comprobamos compatibilidad del navegador cliente	0.1
lSendTTFB	.t.	Adjunta al final del HTML, enviado como respuesta, los milisegundos desde la llamada como un comentario, solo funciona en modo Debug	0.1
lMsgSavePage	.f.	Nos graba en el fichero “SendPage.html” dentro de la carpeta cgi-bin de Apache, el código Html que se va ha enviar en SendPage(), muy útil para depurar el código que se enviara al navegador, solo valida en modo Debuger. Cuando enviamos con SendScript() se guarda en SendScript.html y con SendJSon() como SendJSon.html	0.3
nDuracionCookie		Duración de los cookies por defecto.	0.1
nRequestMode		Modo de llamada recibida desde el navegador ver valores posibles en Constantes	0.1

Métodos

Método	Descripción	Versión
New	Instanciación del objeto, lo habitual es asignarlo en este momento a oCGI.	0.1
Run	Este método comprobara si la propiedad indicada en ::cFunction contiene algún método declarado de la Clase TCgi, y en caso afirmativo lo ejecuta, en caso de no existir realizara una llamada al método Pagina404, lo que devolverá al navegador la indicación de que la página no existe.	0.1
Tratamiento de parámetros recibidos		
GetCgiValue	Recuperación del valor de un parámetro recibido.	0.1
GetCgiArray	Recuperación del array de selecciones de un parámetro recibido.	0.1
GetCgiType	Recuperación del tipo de valor de un parámetro recibido.	0.1
GetCgiFile	Recuperación del fichero temporal recibido en un parámetro recibido.	0.1
GetCgiMethod	Recuperación el metodo de recepción del Param (rmGET o rmPOST).	0.4
aParamsToHash	Nos retorna un Hash con todos los parámetros recibidos.	0.1
Tratamiento de Cookies		
SendHardCookie	Enviar cookie con fecha de caducidad.	0.1
SendSoftCookie	Enviar una cookie de sesión.	0.1
SendCodefCookie	Enviar cookie codificada.	0.1
GetCodefCookie	Recuperar valor de un cookie encriptado.	0.1
GetCookieValue	Recuperar valor de un cookie.	0.1
DeleteCookie	Borrar cookies.	0.1
Métodos de Envió		
SendPage	Enviar código HTML al navegador.	0.1
SendCachePage	Cachear código HTML y Enviar al navegador.	0.1
SendPageNoCache	Enviar código HTML al navegador pero sin guardarlo en la caché.	0.1
SendFile	Enviar un fichero al navegador.	0.1
SendJSon	Enviar un JSon al navegador.	0.1
SendScript	Enviar un Script JS al navegador.	0.1
Tratamiento de aSharedParams		
AddSharedParam	Añade a aSharedParams, si existe lo actualiza.	0.3
GetSharedParam	Recupera el valor de un aSharedParams.	0.3
DelSharedParam	Elimina un aSharedParams.	0.3
Otros Métodos		
Console	Podemos enviar mensajes a Néfele Console para facilitar la depuración.	0.1
Pagina404	Método que se ejecutara cuando sea llamada una función que no exista o no sea accesible.	0.1
MainFunction	Función a ejecutar si no se indica ninguna en la llamada.	0.1
Tiempos	Utilidad para poner puntos de control de tiempo de ejecución en nuestro CGI.	0.1
AddDeprecated	Añade un mensaje a la lista de “Deprecated”.	0.1

El Enrutador

Como ya comentamos antes solo son admitidas como llamadas a funciones de nuestro programa desde el navegador, las que correspondan con métodos de la clase TCgi, por lo que tendremos que crearnos nuestro enrutador sobrecargando la clase TCgi.

Podemos aprovechar dicha sobrecarga para definir también alguna de las propiedades de oCGI.

```
CLASS TCgi FROM XCgi
  PROPERTY lGetData INIT .f.
  METHOD MainFunction()         INLINE Menu()
  METHOD inicio()               INLINE Menu()
  METHOD pruebaspedro()         INLINE PruebasPedro()
  METHOD visparam()             INLINE VisParam()
  METHOD visparam2()            INLINE VisParam( .T. )
  METHOD apachevars()           INLINE ApacheVars()
  METHOD holamundo()
END CLASS
```

En este ejemplo utilizamos la técnica de tener nuestras PROCEDURE y mediante la clausula INLINE realizar el enrutamiento de las llamadas hacia dichas PROCEDURE, incluso haciendo dos llamadas al mismo PROCEDURE con parametros distintos dependiendo del método del oCGI llamado.

También podemos desarrollar los métodos de la clase TCgi.

```
METHOD holamundo() CLASS TCgi
  WITH OBJECT TWebPage():New()
     .
     .
     .
    oCGI:SendPage(:Create())
  END WITH
RETURN Nil
```

Si utilizamos el NéfeleWizard para montar la estructura inicial de nuestra aplicación, este nos creara un enrutador básico en Main.prg, justo debajo del **PROCEDURE CGI_Init()** que es donde se instancia la clase TCgi asignándola a oCGI y se inicia con oCGI:Run().

From:
https://nefele.dev/wiki/ - **Nefele Project**

Permanent link:
https://nefele.dev/wiki/nefele/ocgi?rev=1621781627

Last update: **23/05/2021 14:53**