

Composant Serveur API REST pour WinDev®

Léger.

Rapide.

Gratuit & Open Source.

Sans serveur IIS/Apache/...

Sans licence Serveur WebDev®.

Pas de SAAS, pas d'abonnement.

Windows 32/64 & Linux

Présentation

NEW : LightREST V3

La Documentation



Téléchargements

Un coup de main ?

Classe IrRoute



Role

Décrit une route REST : chemin, méthode REST, callback, timeout, options (OPTIONS/CORS), custom data et hooks (niveau route).

Objet passé en paramètre à la méthode *IrServer::AddRoute()*



Membres et propriétés

Nom	Type	Usage
Route	chaîne	Chemin REST (ex: /customers/{id}).
Méthode	chaîne	Méthode HTTP (GET/POST/PUT/DELETE/...).
Handler	procédure	Handler REST appelée par le moteur LightREST lorsque la route est appelée. La procédure doit être prototypée ainsi : <pre> <i>procedure</i> <i>procRoute(poRequest</i> <i>objet IrRequest,</i> <i>poResponse objet</i> <i>IrResponse)</i> </pre> Prototype antérieur à la V3 : <pre> <i>procedure</i> <i>procRoute(poRequest</i> <i>objet IrRequest) : objet</i> <i>IrResponse</i> </pre> L'objet IrResponse (ou l'objet poResponse selon le cas ci-dessus) retourné doit affecter :

		- à <i>minima</i> le membre :Status en cas d'erreur (éventuellement complété par des détails dans :Body) - :Body, :ContentType et :Status en cas d'envoi d'une réponse valide
Tag	ch aî ne	Etiquette pour identifier facilement la route dans le code (dans un HOOK par exemple).  Par défaut, le Tag est initialisé à la valeur Route@Méthode (exemple : "/ping@GET"), mais le développeur LightREST peut affecter toute valeur qui lui sera utile.
Timeout	Durée	Timeout spécifique à la route. Ce membre est prioritaire sur le timeout défini au niveau du serveur LightREST (IrServer:RequestTimeout).

		En cas de dépassement, l'exécution du Handler REST WinDev en cours est annulée et une erreur HTTP 408 (StatusRequestTimeou t) est envoyée au client.
Use Global Option Headers	boolean	Si vrai, ajoute automatiquement une route OPTIONS associée. Les headers OPTIONS globaux du serveur sont alors utilisés.
Custom Data	Variable	Données métier ou techniques persistantes attachées à la route. Seront intégrées dans le membre <u>IrRequest:RouteCustomData</u> reçu en paramètre de la procédure REST.
NoRegex	boolean	Désactive la validation par expression régulière de la route.

A la création de la route, la validité de sa syntaxe est vérifiée avec l'expression RegEx définie dans la constante
IrServer::ROUTE_REGEX
 .

Si :NoRegex=vrai, aucune vérification ne sera effectuée (dans l'éventualité où l'expression RegEx proposée par LightREST ne correspondrait pas au besoin).

OptionsHeaders	ta ble au as so cia tif (c cS an sC as se +A ve cD ou bl	Headers OPTIONS spécifiques à cette route. Ils seront ajoutés systématiquement aux réponses de cette route avec la méthode OPTIONS. Utilisé habituellement pour gérer le cross domain avec les clients de type navigateur.
----------------	---	--

```
on  
)  
de  
ch  
aî  
ne  
s
```

▼ C Constantes

Contrôle du serveur

Nom	Type	Usage
Sta rt		

Méthodes HTTP

Nom	Type	Usage
Met hod GET	ch aî ne	Méthode REST <u>GET</u>
Met hod POS	ch aî ne	Méthode REST <u>POST</u>

T		
Met hod PUT	ch aî ne	Méthode REST <u>PUT</u>
Met hod DEL ETE	ch aî ne	Méthode REST <u>DELETE</u>
Met hod HEA D	ch aî ne	Méthode REST <u>HEAD</u>
Met hod PAT CH	ch aî ne	Méthode REST <u>PATCH</u>
Met hod OPT ION S	ch aî ne	Méthode REST <u>OPTIONS</u>
Met hod TRA CE	ch aî ne	Méthode REST <u>TRACE</u>

Méthodes

AddHook

Présentation

Ajoute un Hook au niveau de la route.

Le(s) hook(s) ajouté(s) ici ne s'applique(nt) qu'à cette route et permet(tent) :

- d'exécuter un traitement avant le Handler REST (auth, validation, rate-limit, etc.)
- d'exécuter un traitement après le Handler REST (audit, métriques, etc.)
- de modifier ou remplacer la réponse juste avant l'envoi au client (selon les évènements définis dans IrHook)

Si plusieurs hooks sont passés en paramètre, ils sont exécutés dans cet ordre (jusqu'à éventuelle annulation par EVE_ABORT).

 Pour installer un hook global pour toutes les routes, utiliser IrServer.AddHook().

Prototypage

```
AddHook(poHook objet IrHook [, poF
```

**Paramètres**

No m	T y p e	Usage
po Ho ok 1 [, po Ho ok 2, po Ho ok 3, .. .] &n bs p; &n bs p; &n bs p; &n	O bj et (s) <i>lr</i> <i>H</i> <i>o</i> <i>o</i> <i>k</i>	Hook(s) à ajouter au niveau de la route.

`bs``p;`

Valeurs retournées

(aucune)

Exemple

- Route publique */ping* sans authentification
- Route */private* avec hook d'authentification au niveau route

```
oServer      est objet
lrServer
oRoutePing   est objet
lrRoute
oRoutePriv   est objet
lrRoute
oAuthHook    est objet lrHook
bOK          est booléen
sErr         est chaîne
```

[Copier](#)

```
// ---- Hook d'auth ----
oAuthHook.Evants =
lrHook::EVE_BEFORE_METHOD
oAuthHook.Handler =
fnAuthHook

// ---- Route publique ----
oRoutePing.Method =
lrRoute::MethodGET
oRoutePing.Route =
"/ping"
oRoutePing.RESTFunction =
procPing
```

```
// ---- Route privée ----
oRoutePriv.Method      =
lrRoute::MethodGET
oRoutePriv.Route       =
"/private"
oRoutePriv.RESTFunction =
procPrivate
oRoutePriv.AddHook(oAuthHook
)

(bOK, sErr) =
oServer::AddRoute(oRoutePing)
SI PAS bOK ALORS
Erreur(sErr) ; RETOUR ; FIN

(bOK, sErr) =
oServer::AddRoute(oRoutePriv)
SI PAS bOK ALORS
Erreur(sErr) ; RETOUR ; FIN

// ---- Handlers ----
PROCEDURE INTERNE
fnAuthHook(pInfo
lrHook:lrEventInfo) :
lrHook:lrEventReturn
    // pInfo dépend de lrHook
    (request/response/message...
    )
    // Vérifie un header,
    sinon prépare une réponse
    401 selon l'implémentation
    lrHook

    SI
    pInfo.Request.GetHeaderValue
    ("token")="xxxxxxxxxxxxx"
    ALORS
        RENVOYER lrHook::EVE_OK
    SINON
        //Auth KO, on annule
        l'exécution de la route avec
        une erreur 401
        pInfo.Response.Status =
        lrResponse::StatusForbidden
        RENVOYER
        lrHook::EVE_ABORT
```

```
FIN
FIN

PROCEDURE INTERNE
procPing(poRequest objet
lrRequest) : objet
lrResponse
    oResp est objet lrResponse

    oResp.Status      =
lrResponse::StatusOK
    oResp.ContentType =
lrResponse::ContentTXT
    oResp.Body         = "pong"
    RENVOYER oResp
FIN

PROCEDURE INTERNE
procPrivate(poRequest objet
lrRequest) : objet
lrResponse
    oResp est objet lrResponse

    oResp.Status      =
lrResponse::StatusOK
    oResp.ContentType =
lrResponse::ContentTXT
    oResp.Body         =
"private data"
    RENVOYER oResp
FIN
```