JSON プロバイダ

HTTP / HTTPS 対応

Version 1.0.0

ユーザーズ ガイド

December 14, 2017

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2016-10-5	初版.
	2017-12-14	誤植修正.

目次

1. はじめに

JSON プロバイダは, JSON 形式でデータをサーバへ送信するプロバイダです.

このプロバイダを利用することで, 任意のサーバに JSON データを送信することができます.

なお, 送信対象となる Web サーバは, REST の原則に基づいて API が設計されている必要があります.

本ドキュメントでは, JSON プロバイダの概要と, 実装されている CAO インタフェース(関数仕様)について説明しています.

2. プロバイダの概要

2.1. インストール

JSONプロバイダモジュールは，下記のDLLで構成されています．ORiN2 SDKのインストーラでインストールした場合は，インストール作業は不要です．手動でインストールする場合は，表 2-1 のように実行してください．

表 2-1 JSONプロバイダ

ファイル名	CaoProvJSON.dll
ProgID	CaoProv.JSON
レジストリ登録	regsvr32 CaoProvJSON.dll
レジストリ登録の抹消	regsvr32 /u CaoProvJSON.dll

2.2. 概要

JSON プロバイダは，登録されたデータを JSON 形式に変換し，変換したデータを HTTP/HTTPS のプロトコルに従って Web サーバへ送信します．

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

書式　AddController (<bstrCtrlName:BSTR>, <bstrProvName:BSTR>, <bstrPcName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName > : [in] コントローラ名
< bstrProvName > : [in] プロバイダ名. 固定値 ="CaoProv.JSON"
<bstrPcName> : [in] プロバイダの実行マシン名(未使用)
<bstrOption> : [in] オプション文字列

- Web サーバのベース URI(BaseURI)
- 無効な SSL 証明書の許可(AllowInvalidCert)

以下にオプション文字列(bstrOption)の詳細を示します.

表 2-2 CaoWorksapce::AddController メソッドのオプション文字列

設定項目	設定内容	必須	備考
BaseURI	Web サーバの URI 文字列	○	
AllowInvalidCert	無効な SSL 証明書の許可 "YES" : 許可する "NO" : 許可しない	×	Https の場合のみ有効です. 省略時は "NO" として扱います. 社内の Web サーバなど, 接続先が信頼できる場合にのみ設定してください.

使用例　使用例として,「123.168.1.3:443」に無効なSSL証明書で許可して接続するコードを示します.

```
Private caoEng As CaoEngine      ' Engine オブジェクト
Private caoWs As CaoWorkspace    ' WorkSpace オブジェクト
Private caoCtrl As CaoController ' Controller オブジェクト

Set caoEng = New CaoEngine
Set caoWS = caoEng.CaoWorkspaces.Item(0)
Set caoCtrl = CaoWS.AddController("JSON", "CaoProv.JSON", "",
"BaseURI=https://192.168.1.3:443,AllowInvalidCert=YES")
```

2.3.2. CaoController::AddFile メソッド

書式　AddFile (<bstrName:BSTR>, <bstrOption:BSTR>)

bstrOption には，送信先 Web サーバのリソースのエンドポイント URI を設定することができます.

< bstrName > : [in] ファイル名

<bstrOption> : [in] オプション文字列

- Web サーバリソースのエンドポイント URI

表 2-3 CaoController::AddFile メソッドのオプション文字列

設定項目	設定内容	必須	備考
EndpointURI	Web サーバのリソースのエンドポイント URI 文字列	×	エンドポイントを指定しない場合，CaoController::AddFile メソッドで指定したベース URI に対して POST 送信を実行します.

使用例　使用例として，2.3.1の設定例の場合https://192.168.1.3:443/users/ へPOSTするコードを示します.

```
Private m_caoFile as CaoFile     ' Fileオブジェクト
set m_caoFile = caoCtrl.AddFile("point", " EndpointURI=users/")
```

3. コマンドリファレンス

3.1. File クラス

3.1.1. CaoFile::AddVariable メソッド

この AddVariable メソッドを使用してファイルへプロパティを追加します. このメソッドの引数(bstrName)は, JSON を出力する際のオブジェクトのキーとして扱われます.

書式　AddVariable(<bstrName:BSTR>, <bstrOption:BSTR>)

< bstrName > : [in] 変数名
< bstrOption > : [in] オプション文字列 (未使用)

追加された変数に対して値を設定することで, JSON オブジェクトへ値を設定します.
以下に, 設定できる値の型の一覧を示します.

表 3-1 Variable に設定可能な型一覧

種類	JSON 型	設定する型
null 値	null	VT_NULL
真偽値	Boolean	VT_BOOL
整数	Number	VT_I4
小数	Number	VT_R8
文字列	String	VT_BSTR
配列(真偽値)	Boolean[]	VT_ARRAY | VT_BOOL
配列(整数)	Number[]	VT_ARRAY | VT_I4
配列(小数)	Number[]	VT_ARRAY | VT_R8
配列(文字列)	String[]	VT_ARRAY | VT_BSTR

使用例　CaoVariable への値設定

```
caoVar = m_caoFile.AddVarialbe("name", "")
caoVar.Value = "File1"
```

3.1.2. CaoFile::AddFile メソッド

CaoFile クラスに File クラスを追加します.
オプション文字列として, 追加先の File クラスと追加する File クラスとを紐づけるキー(BSTR 型)を指定する必要があります. また, 追加先の File クラスでこの File クラスをどのように保持するかも指定します.
設定方法は, CaoController::AddFile メソッドと同様です.

書式　AddFile(<bstrName:BSTR>, <bstrOption:BSTR>)

< bstrName > : [in] ファイル名
< bstrOption > : [in] オプション文字列

- 親 File とこの File を紐づけるキー(Key)
- 親 File でこの File をどのような型で保持するか(Type)

以下に, オプション文字列(bstrOption)の詳細を示します.

表 3-2 CaoFile::AddFile メソッドのオプション文字列

設定項目	設定内容	必須	備考
Key	キーの文字列	○	
Type	“Array”: 配列として保持 “Object”: オブジェクトとして保持	×	省略時は “Object” として扱います.

使用例

以下に, オプション文字列の指定と結果として出力される JSON の例を示します.

表 3-3 AddFile のオプション指定と出力される JSON の例

コマンド例	出力される JSON

caoFile1 = caoContoller.addFile("Parent", "EndpointURI=xxxx")
parentName = caoFile1.addVariable("name", "")
parentName.Value = "Taro"
parentAge = caoFile1.addVariable("age", "")
parentage.Value = 34
 	
caoFile2 = caoFile1.AddFile("child", "Key=child,Type=Object")
childName = caoFile2.addVariable("name", "")
childName.Value = "Hanako"
	{
	“name”: “Taro”,
	“age”: 34,
	“child”: {
	“name: “Hanako”
	}
	}

caoFile1 = caoContoller.addFile("Parent", "EndpointURI=xxxx")
parentName = caoFile1.addVariable("name", "")
parentName.Value = "Jiro"
 	
caoChildFile1 = caoFile1.AddFile("child1", "Key=children,Type=Array")
childName1 = caoChildFile1.addVariable("name", "")
childName1.Value = "Yoshiko"
 	
caoChildFile2 = caoFile.AddFile("child2", "Key=children,Type=Array")
childName2 = caoChildFile2.addVariable("name", "")
childName2.Value = "Masako"
	{
	“name”: “Jiro”,
	“children”: [{
	“name: “Yoshiko”
	},{
	“name: “Masako”
	}]
	}

3.1.3. CaoFile::Execute メソッド

CaoFile クラスに属するプロバイダ固有の拡張コマンドを実行します.

CaoController::AddFile メソッドで追加された File クラスのみ実行できます.

CaoFile::AddFile メソッドで追加された File クラスで実行した場合, このメソッドは失敗します.

書式　Execute(<bstrCmd:BSTR>, <vntParam:VARIANT>)

< bstrCmd > : [in] コマンド名

< vntParam > : [in] パラメータ

以下に, 現在使用可能なコマンドの一覧を示します.

表 3-4 CaoFile::Execute コマンド一覧

コマンド	機能	ページ
SetRequestHeader	Http リクエストヘッダを追加します	12
SetTimeout	Http 要求のタイムアウト時間を設定します	12
CreateAndPost	データを JSON 形式でサーバへ POST します	13
CreateJSON	データを JSON 文字列として出力します	13
Post	JSON 文字列を Web サーバへ POST します	13

3.1.3.1. CaoFile::Execute(“SetRequestHeader”) コマンド

Web サーバへリクエストを送信する際のリクエストヘッダを追加します.

引数は, ”{リクエストヘッダのキー},{リクエストヘッダの値}”のように, 配列の形式で指定する必要があります.

また, 同一のリクエストキーを指定した場合, 以前の値を上書きするので注意してください.

なお, Content-Type および Content-Length については, プロバイダが自動で設定するため設定の必要はありません.

設定例: X-MyCompanyToken,0123456789ABCD

書式 SetRequestHeader (<Data>)

< Data > : [in] 追加するリクエストヘッダのキーおよびその値
(VT_BSTR | VT_ARRAY)

使用例

```
result = m_caoFile.Execute("SetRequestHeader", "X-MyCompanyToken,0123456789ABCD")
```

3.1.3.2. CaoFile::Execute(“SetTimeout”) コマンド

Web サーバへリクエストを送信する際のタイムアウト時間を追加します. 単位はミリ秒です. 0 より大きな値を指定してください.

省略した場合は 5000(5 秒)が設定されます.

書式 SetRequestHeader (<Data>)

< Data > : [in] Http 要求のタイムアウト時間(ミリ秒)
(VT_I4)

使用例

```
result = m_caoFile.Execute("SetTimeout", 1000)
```

3.1.3.3. CaoFile::Execute("CreateAndPost") コマンド

File の保持するデータを, JSON 形式で Web サーバへ POST 送信します.
戻り値には, POST したサーバからのレスポンスボディ文字列が設定されます.

書式　CreateAndPost ()

使用例

```
result = m_caoFile.Execute("CreateAndPost")
```

3.1.3.4. CaoFile::Execute("CreateJSON") コマンド

File の保持するデータを, JSON 形式の文字列として返却します.

書式　CreateJSON ()

使用例

```
result = m_caoFile.Execute("CreateJSON")
```

3.1.3.5. CaoFile::Execute("Post") コマンド

任意の文字列を Web サーバへ POST します.
文字列の形式が JSON でない場合, 3.1.3.1 に従って, Request-Header の Content-Type を適当な値で上書きすることができます.
戻り値には, POST したサーバからのレスポンスボディ文字列が設定されます.

書式　Post (<Data>)

< Data >	:	[in] 送信する文字列 (VT_BSTR)

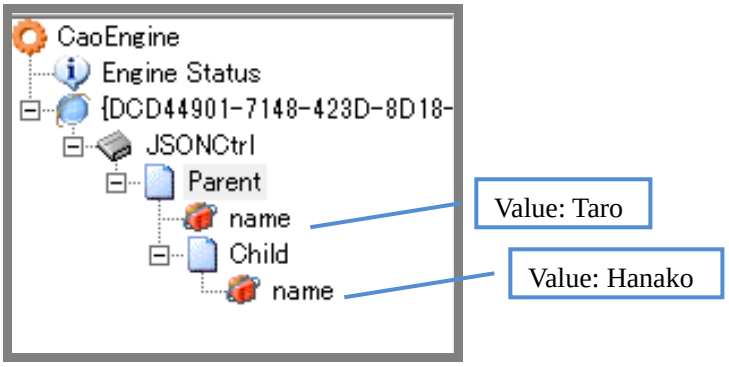
使用例

```
result = m_caoFile.Execute("Post", "{""name"":""Taro""}")
```

4. CAO 構成例

以下に，本プロバイダを用いて JSON オブジェクトを表現する際の CAO 構成例を示します.

JSON オブジェクト	CAO 構成

{
“name”: “Taro”,
“child”: {
“name: “Hanako”
}
}
	

付録A. 2-clause BSD license For picojson

【picojson】

https://github.com/kazuho/picojson

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.