




EMBARCADERO CONFERENCE ONLINE

Código, código e mais código
Conectando pessoas através do desenvolvimento

EMBARCADERO CONFERENCE ONLINE



O que você não sabe sobre
JSON no Delphi

Gustavo Mena Barreto

Palestrante

Gustavo Mena Barreto

Analista de Sistemas na Aquasoft

Formado em Análise e Desenvolvimento pela Unisinos/RS

Mais de 10 anos de experiência com Delphi

Agenda

- Opções para tratamento e processamento de JSON no Delphi.
- Forma mais produtiva de realizar um parser.
- Opções que automatizam tratamentos padrões no processamento de JSON.

Por que falar sobre JSON no Delphi?



Opções para tratamento de JSON

- Modo mais simples
- JsonToDelphi
- SuperObject
- Componente RESTResponseDataSetAdapter
- Classe TJJSONMarshal

JSON

```
{
  "block_hash": "0000000000000000c504bdea36e531d8089d324f2d936c86e3274f97f8a44328",
  "inputs": [
    {
      "prev_hash": "583910b7bf90ab802e22e5c25a89b59862b20c8c1aeb24dfb94e7a508a70f121",
      "output_index": 1,
      "script": "48304502210086de855e03008ab",
      "output_value": 16450000,
      "sequence": 4294967295,
      "addresses": [
        "1GbMfYui17L5m6sAy3L3WXAAtf1P32bxJXq"
      ],
      "script_type": "pay-to-pubkey-hash",
      "age": 292997
    },
    {
      "prev_hash": "f6966bb3d3ba0eb97fd11b223fb13c793c0b4a1b3deb575e8ae38d666c1d00d9",
      "output_index": 36,
      "script": "48304502210086de855e03008ab",
      "output_value": 10061545,
      "sequence": 4294967295,
      "addresses": [
        "19YtzZdcfs1V2ZCgyRwo8i2wLT8ND1Tu4L"
      ],
      "script_type": "pay-to-pubkey-hash",
      "age": 292998
    }
  ]
}
```

Antes de tudo...

uses

```
System.JSON,  
REST.Json;
```

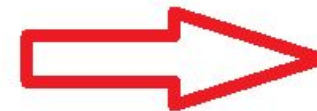

Modo mais simples

```
var
  objeto : TJSONObject;
  ObjetoArray: TJSONArray;
  I: Integer;
begin
  //Leitura
  objeto := TJsonObject.ParseJSONValue(MemoJson.Lines.Text) as TJSONObject;

  ShowMessage(objeto.Get('block_hash').ToString);
  ShowMessage(objeto.GetValue('block_hash').Value);

  ObjetoArray := objeto.Get('inputs').JsonValue as TJSONArray;

  for I := 0 to ObjetoArray.Size - 1 do
  begin
    MemoResultado.Lines.Add(
      ObjetoArray.Get(i).ToString + ': '
      + ObjetoArray.Get(i).Value);
  end;
end;
```



```
{
  "prev_hash": "583910b7bf90ab802e22e5c25a89b59862b20c8c1aeb24dfb94e7a508a70f12\r\n1",
  "output_index": 1,
  "script": "48304502210086de855e03008ab",
  "output_value": 16450000,
  "sequence": 4294967295,
  "addresses": [
    "1GbMfYui17L5m6sAy3L3WXAtf1P32bxJXq"
  ],
  "script_type": "pay-to-pubkey-hash",
  "age": 292997
},
{
  "prev_hash": "f6966bb3d3ba0eb97fd11b223fb13c793c0b4a1b3deb575e8ae38d666c1d00d\r\n9",
  "output_index": 36,
  "script": "48304502210086de855e03008ab",
  "output_value": 10061545,
  "sequence": 4294967295,
  "addresses": [
    "19YtzZdcfs1V2ZCgyRwo8i2wLT8ND1Tu4L"
  ],
  "script_type": "pay-to-pubkey-hash",
  "age": 292998
}
```

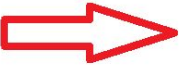
Modo mais simples

```
var
  objeto : TJSONObject;
  ObjetoArray: TJSONArray;
  ObjetoInputs: TJSONObject;
begin
  //Escrita
  objeto := TJSONObject.Create;
  ObjetoArray := TJSONArray.Create;

  objeto.AddPair('block_hash', '123456789');
  Objeto.AddPair('Inputs', ObjetoArray);

  ObjetoInputs := TJSONObject.Create;
  ObjetoInputs.AddPair('prev_hash', '987654321');
  ObjetoInputs.AddPair('script', '1');

  ObjetoArray.Add(ObjetoInputs);
  MemoResultado.Lines.Text := objeto.ToString;
end;
```



```
{"block_hash": "123456789", "Inputs":
[{"prev_hash": "987654321", "script": "1"}]}
```

JsonToDelphi

- jsontodelphi.com
- Transforma Json em Classe do Delphi

JsonToDelphi

```
var
  TesteJson: TRoot;
  Inputs: TInputs;
begin
  TesteJson := TJson.JsonToObject<TRoot>(MemoJson.Lines.Text);

  ShowMessage('BlockHash - '+TesteJson.BlockHash);

  for Inputs in TesteJson.Inputs do
  begin
    Showmessage('ScriptType - '+Inputs.ScriptType);
  end;
end;
```


JsonToDelphi

```
var
  TesteJson: TRoot;
  Inputs: TInputs;
  I: Integer;
begin
  TesteJson := TRoot.Create;
  TesteJson.BlockHash := 'www.aquasoft.com.br';

  for I := 0 to 2 do
  begin
    Inputs := TInputs.Create;
    Inputs.Age := 20;
    Inputs.Script := 'teste script';

    TesteJson.Inputs.Add(Inputs);
  end;

  MemoResultado.Lines.Text :=
    TJson.ObjectToJsonString(TesteJson, [joIgnoreEmptyStrings]);
end;
```



```
{
  "addresses": [],
  "block_hash": "www.aquasoft.com.br",
  "block_height": 0,
  "block_index": 0,
  "confidence": 0,
  "confirmations": 0,
  "confirmed": "1899-12-30T00:00:00.000-02:00",
  "double_spend": false,
  "fees": 0,
  "inputs": [
    {
      "addresses": [],
      "age": 20,
      "output_index": 0,
      "output_value": 0,
      "script": "teste script",
      "sequence": 0
    },
    {
      "addresses": [],
      "age": 20,
      "output_index": 0,
      "output_value": 0,
      "script": "teste script",
      "sequence": 0
    },
    {
      "addresses": [],
      "age": 20,
      "output_index": 0,
      "output_value": 0,
      "script": "teste script",
      "sequence": 0
    }
  ],
  "outputs": [
    {
      "received": "1899-12-30T00:00:00.000-02:00",
      "size": 0,
      "total": 0,
      "ver": 0,
      "vin_sz": 0,
      "vout_sz": 0,
      "vsize": 0
    }
  ]
}
```

Tratamentos padrões de JSON

- docwiki.embarcadero.com/Libraries/Alexandria/en/REST.Json.TJsonOption

REST.Json.TJsonOption

Up to Parent: REST.Json

Delphi

```
TJsonOption = (joIgnoreEmptyStrings, joIgnoreEmptyArrays,  
joDateIsUTC, joDateFormatUnix, joDateFormatISO8601,  
joDateFormatMongo, joDateFormatParse,  
joBytesFormatArray, joBytesFormatBase64,  
joIndentCaseCamel, joIndentCaseLower, joIndentCaseUpper,  
joIndentCasePreserve);
```


SuperObject

- Biblioteca gratuita para tratar JSON
- Utilizada em versões do Delphi que não possuem suporte nativo a JSON
- github.com/hgourvest/superobject

SuperObject

```
var
  Objeto: ISuperObject;
  Inputs: ISuperObject;
begin
  Objeto := SO(MemoJson.Lines.Text);
  ShowMessage(Objeto.S['block_hash']);
  ShowMessage(Objeto['block_hash'].AsString);

  for Inputs in objeto['inputs'] do
  begin
    MemoResultado.Lines.Add(Inputs.AsString);
  end;
end;
```

```
{ "age": 292997, "script_type": "pay-to-pubkey-hash", "script": "48304502210086de855e03008ab", "addresses": [ "1GbMfYui17L5m6sAy3L3WXAtf1P32bxJXq" ], "output_index": 1, "sequence": 4294967295, "output_value": 16450000, "prev_hash": "583910b7bf90ab802e22e5c25a89b59862b20c8c1aeb24dfb94e7a508a70f121" }
{ "age": 292998, "script_type": "pay-to-pubkey-hash", "script": "48304502210086de855e03008ab", "addresses": [ "19YtzZdcfs1V2ZCgyRWo8i2wLT8ND1Tu4L" ], "output_index": 36, "sequence": 4294967295, "output_value": 10061545, "prev_hash": "f6966bb3d3ba0eb97fd11b223fb13c793c0b4a1b3deb575e8ae38d666c1d00d9" }
```

SuperObject

```
var
  Objeto: ISuperObject;
  Inputs: ISuperObject;
begin
  Objeto := SO;
  Objeto.S['block_hash'] := '123456789';
  Objeto.O['inputs'] := SA([]);

  Inputs := SO;
  Inputs.S['script'] := 'Teste script 1';
  Inputs.I['age'] := 30;
  Objeto.A['inputs'].O[0] := Inputs;

  Inputs := SO;
  Inputs.S['script'] := 'Teste script 2';
  Inputs.I['age'] := 60;

  Objeto.A['inputs'].O[1] := Inputs;

  MemoResultado.Lines.Text := Objeto.AsString;
end;
```



```
{"block_hash":"123456789","inputs":
[{"age":30,"script":"Teste script 1"},
{"age":60,"script":"Teste script 2"}]}
```


Componente `RESTResponseDataSetAdapter`

- Componente da Paleta REST Client
- Pode ser utilizado em conjunto com a ferramenta REST Debugger

Componente RESTResponseDataSetAdapter

REST Debugger 10.4.2
Embarcadero Technologies

Request

Request Parameters Authentication Connection

Method: GET URL: https://api.blockcypher.com/v1/btc/main/txs/f854aebae95150b379cc1187d848d58225f3c4157fe992bcd166f58bd5063449

Content-Type:

Custom body:

Information

The following components have been copied to the clipboard: TRESTClient, TRESTRequest, TRESTResponse, TRESTResponseDataSetAdapter, TFDMemTable

OK

Response

https://api.blockcypher.com/v1/btc/main/txs/f854aebae95150b379cc1187d848d58225f3c4157fe992bcd166f58bd5063449
200 : OK - 3316 bytes of data returned. Timing: Pre: 0ms - Exec: 984ms - Post: 79ms - Total: 1063ms

Headers Body **Tabular Data**

Content is valid JSON JSON Root Element: Nested Use Types: Rich Apply

block_hash	block_heig...	block_index	hash	addre
00	293000,00	19,00	f854aebae95150b379cc1187d848d58225f3c4157fe992bcd166f58bd5063449	["13X

Proxy-server disabled

Componente RESTResponseDataSetAdapter

```
begin  
    RESTRequest1.Execute;  
    RESTResponseDataSetAdapter1.Active := True;  
end;
```


Classe TJSONMarshal

- Classe que baseia nas informações adicionadas via RTTI(Run Time Type Information) para exportar automaticamente todos os campos públicos do objeto de uma classe
- docwiki.embarcadero.com/Libraries/Alexandria/en/Data.DBXJSONReflect.TJSONMarshal
- docwiki.embarcadero.com/RADStudio/Alexandria/en/Serializing_User_Objects

Classe TJSONMarshal

```
var
  lMarshal : TJSONMarshal;
  TesteJson: TRoot;
  Inputs: TInputs;
  I: Integer;
begin
  TesteJson := TRoot.Create;
  TesteJson.BlockHash := 'www.aquasoft.com.br';

  for I := 0 to 2 do
  begin
    Inputs := TInputs.Create;
    Inputs.Age := 20;
    Inputs.Script := 'teste script';

    TesteJson.Inputs.Add(Inputs);
  end;

  lMarshal := TJSONMarshal.Create (TJSONConverter.Create);
  MemoResultado.Lines.Text := lMarshal.Marshal(TesteJson).ToString();

  lMarshal.Free;
end;
```

Classe TJSONMarshal

```
{Marca FScript para não ser incluído na serialização }  
[JSONMarshaled(false)]  
FScript: string;
```


Classe TJSONMarshal

- Existe a possibilidade de registrar uma função anônima para realizar conversões personalizadas, sendo possível converter uma classe inteira para um outro tipo de dado e também converter um único campo e tratá-lo conforme a necessidade.
- procedure RegisterConverter

EMBARCADERO
CONFERENCE ONLINE

DÚVIDAS PERGUNTAS





gustavo.barreto@aquasoft.com.br



<https://br.linkedin.com/in/gustavo-mena-barreto-59670512a>

**EMBARCADERO
CONFERENCE ONLINE**

OBRIGADO



EMBARCADERO CONFERENCE ONLINE

Código, código e mais código
Conectando pessoas através do desenvolvimento