
European DataFlex users' conference - Edinburgh, Scotland
EDUC
2018
DataFlex to New Heights
Introducing JSON in DataFlex
John Tuohy


JSON

JavaScript Object Notation

- JSON is a lightweight data-interchange format.
- JSON is a text format.
- JSON is completely language independent.
- JSON is based on a subset of JavaScript.
- JSON is easy for humans tor read and write.
- JSON is easy for machines parse and generate.
- JSON is familiar to programmers of C-family languages like C++, Java, Python, JavaScript.

The JSON Format

- Objects
- Arrays
- Strings
- Booleans
- Numbers

```
{
   "name" : "John",
   "details" : {
      "age" : 31,
      "male" : true
   },
   "ratings" : [ 8, 7.5, 8, 5.5 ]
}
```

JSON

```
{
   "name" : "John",
   "details" : {
      "age" : 31,
      "male" : true,
   },
   "ratings" : [
      8,
      7.5,
      8,
      5.5
   ]
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<student>
   <name>John</name>
   <details>
      <age>31</age>
      <male>true</male>
   </details>
   <ratings>
      <rate>8</rate>
      <rate>7.5</rate>
      <rate>8</rate>
      <rate>5.5</rate>
   </ratings>
</student>
```

JSON

- Human readable
- Hierachical
- Quicker
  - Shorter
    - No end tags
  - Easier to parse
    - Can be evaluated in some languges
- Has arrays

- Lighter and native to JavaScript

XML

- Human readable
- Hierachical
- Better standardized
- More extensive
  - Attributes
  - Namespaces
  - XML Schema
  - XSL
  - XPath

- Heavier but wider supported

Usage

- Interchange data
  - REST JSON API's
  - JavaScript WebApps
    - Like the WebApp Framework
- Store data
  - Serialize data from memory

JSON in DataFlex

http://localhost/WebOrder_19/CustomerAndOrderInfo.wso

- Web Services

  http://localhost/WebOrder_19/CustomerAndOrderInfo.wso

  - Built in JSON support
  - Every DataFlex Web-Service can be called using JSON
  - JSON parsing & generation happens in ISAPI handler
- cJsonObject
  - Manually parse & generate JSON
  - Serialize / deserialize structs and arrays
- cJsonHttpTransfer
  - Communicate with JSON Services

cJsonObject

- Parse JSON
- Generate JSON
- Manipulate JSON like a DOM structure

- Convert DataFlex structs and arrays into JSON
- Convert JSON into DataFlex structs and arrays

Parsing Examples

JSON <> Struct

```
{
 "name": "Troy Umstead",
 "details": {
  "age": 20,
  "male": true
 },
 "ratings": [
  2.3,
  5.2,
  4.0,
  9.4
 ]
}
```

```
Struct tStudentDetail
    Integer age
    Boolean male
End_Struct

Struct tStudent
    String name
    tStudentDetail details
    Number[] ratings
End_Struct
```

Structs with JSON

- **Parse JSON into Struct**
  - Parse JSON using cJsonObject
  - Use JsonToDataType function
    - Fills a struct using JSON data
- **Serialize Struct to JSON**
  - Use DataTypeToJson procedure
    - Generates JSON objects based on the struct data
  - Stringify into a JSON string

Struct Examples

API Overview

- cJsonObject
  - ParseString, ParseUtf8
    - Parses a JSON string into the JSON DOM
  - Stringify, StringifyUtf8
    - Generates the JSON string from the JSON DOM
  - DataTypeToJson, JsonToDataType
    - Convert JSON DOM to structs / arrays
  - Member, MemberByIndex, MemberCount, MemberNameByIndex, HasMember, JsonType
    - Traverse JSON DOM
  - AddMember, SetMember, SetMemberValue, InitializeJsonType
    - Manipulate JSON

ParseUtf8 and StringifyUtf8

- UChar array as parameter / argument
  - No argument-size limitations
  - Supported by Read_Block, Write_Block, Set_Field_Value, Get_Field_Value, Field_Current_UCAValue
- Expected encoding is UTF-8
  - Default format when transmitting JSON object the web
- Convert manually if needed
  - ConvertUCharArray of cChartTranslate

```
UChar[] uData

Direct_Input "FileWithJsonData.json"
Read_Block uData -1
Close_Input

Get Create (RefClass(cJsonObject)) to hoJsonDom
Get ParseUtf8 of hoJsonDom uData to bParsed
```


JSON and REST

The HTTP transfer protocol

- Different types of HTTP requests use different to protocols
  - The most common are Post and Get
  - There are others such as Put, Delete and Patch - these are referred to as verbs
  - The low level http interface actually has you send these verbs as:
    - GET, POST, PATCH, PUT, DELETE
- Basically a request sends a verb and data with some header information
- Until early 2000s the HTTP verbs were considered low-level arcane knowledge

REST

- Then REST and RESTful web-services were created

REST: Representational state transfer (**REST**) or **RESTful** Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. (Wikipedia)

REST web services

- REST uses three parts of an HTTP request to create web-service
  - Verb
  - URL
  - Data
- Verb – the verb to define the operation
  - GET – used to retrieve data
  - DELETE – use to delete data
  - POST – Used to Create data
  - PUT – Used to Replace data
  - PATCH – is used to …. whatever
- URL - It uses the URL to determine what the operation is acting on
  //api.example.com/customers/
  //api.example.com/customers/1
  //api.example.com/customers/1/orders
- Data – is passed and returned as HTTP data

REST – web services (continued)

- To make this all a bit more concrete:
  - REST uses the standard HTTP verbs and URLs to determine what to do
  - The data is exchanged as HTTP character data
  - REST often exchanges data in JSON
  - REST transfers the JSON data using UTF-8
  - The content of the data is whatever the service defines (and you need to figure this out)

- SOAP web service vs. Rest web service
  - Not at all the same thing

cJsonHttpTransfer class

- Sub-class of cHttpTransfer
- Used to transfer JSON data across requests
- Passes and receives JSON objects, if invalid JSON data, it fails
- Supports all RESTful verbs Get, Post, Put, Delete, Patch and "verb" just in case
  - Get HttpGetJson
  - Get HttpPutJson
  - Get HttpPostJson
  - Get HttpPatchJson
  - Get HttpDeleteJson
  - Get HttpVerbJson
- Uses UTF8 encoding
- Used to transfer JSON via Http - you have to write the code yourself

Using cJsonHttpTransfer objects

```
Object oJsonHttp is a cJsonHttpTransfer
End_Object

Get HttpGetJson of oJsonHttp "api.example.com" "customers" (&bOk) to hoJsonOut

Get HttpGetJson of oJsonHttp "api.example.com" "customers/1" (&bOk) to hoJsonOut

Get HttpGetJson of oJsonHttp "api.example.com" "customers/1/Orders" (&bOk) to hoJsonOut

Get HttpPostJson of oJsonHttp "api.example.com" "customers" hoJSONIn (&bOk) to hoJsonOut

Get HttpPatchJson of oJsonHttp "api.example.com" "customers/1" hoJSONIn (&bOk) to hoJsonOut

Get HttpDeleteJson of oJsonHttp "api.example.com" "customers/1" 0 (&bOk) to hoJsonOut
```

Examples

- For the samples we use:

  http://jsonplaceholder.typicode.com


European DataFlex users' conference - Edinburgh, Scotland
EDUC
2018
DataFlex to New Heights
Thank you!
Are there any questions?