DataAccess WORLDWIDE

Introduction to JSON in DataFlex

Harm Wibier
Data Access Worldwide

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, Phyton, JavaScript.

The JSON Format

- Objects
- Arrays

- Strings
- Booleans
- Numbers

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

DataAccess WORLDWIDE

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>
```

DataAccess WORLDWIDE

JSON

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

- <u>Lighter and native to JavaScript</u>

XML

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

- <u>Heavier but wider supported</u>

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
  - 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

Handle parsing errors

- ParseString returns false indicating error
- ReportParseError
  - Generates a DataFlex error
- psParseError
  - Provides the error text

```
Get ParseString of hoJson sData to bSuccess
If (bSuccess) Begin
   // Do your thing!
End
Else Begin
   // Handle errors how you prefer
   Send ReportParseError of hoJson
End

Send Destroy of hoJson
```

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
  - pbRequireAllMembers
    - If set to false missing members will just be ignored and are left empty
- **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
```

peWhitespace

- jpWhitespace_Plain

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

- jpWhitespace_Spaced

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

- jpWhitespace_Pretty

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

cJsonObject Instances

- Multiple instances of the same JSON object are possible
- Internally reference counting is used
  - Once a handle is obtained it will point to that piece of JSON until it is destroyed or reinitialized
    - Obtaining as in Create, Member, MemberByIndex
    - Destroying as in Destroy, RemoveMember
    - Reinitializing means calling Parse, ParseUtf8, DataTypeToJson or InitializeJsonType
- One piece of JSON could part of two structures
  - AddMember and SetMember work by reference (they do not clone the objects)
- <u>Every JSON object needs to be destroyed individually</u>

cJsonHttpTransfer

- Easily communicate with Restfull JSON Services
- HTTP Verbs:
  - POST, GET, DELETE, PUT, PATCH
- Directly parses into cJsonObject

<u>John Tuohy will discuss this tomorrow in depth…</u>

Thank you for your time

Have a nice day!