DataAccess WORLDWIDE

Getting the Most from REST and JSON

John Tuohy
DISD 2018
March 2018

Character encoding review

- Single byte character encoding
  - ASCII
  - OEM
  - ANSI
- Multi-byte character encoding (Unicode)
  - UTF-16
  - UTF-8

- DataFlex does most of its character (string) *processing* in OEM

- The standard character *exchange* over the Internet is UTF-8

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

The DataFlex HTTP transfer classes

- cHttpTransfer
  - Low level
  - Supports the most common HTTP transfer verbs: Get, Post (and Put)
    - Get HttpGetRequest
    - Get HttpPostRequest
    - Get HttpPostAddrRequest
  - As of 18.2, it supports any http verb
    - Get HttpVerbAddressRequest
  - Returns the data in an event, its your job to process it
    - Procedure OnDataReceived
  - You can send and receive in whatever encoding format you want (i.e., it's your problem)

- cXmlHttpTransfer
  - Sub-class of cHttpTransfer
  - Used to transfer XML data across requests
  - Passes and receives XML objects, if invalid XML, it fails
  - Support HTTP Post and Get
    - Get HttpGetXmlNode
    - Get HttpPostXmlNode
  - Uses UTF8 encoding
  - Used by client web-services (cClientWebService), which is why you never see it.

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