IBM Db2

Open Data RESTful Support

George Baklarz

December 6th, 2017

simplify coding

```
{
  "store"   : "json",
  "call"    : "RESTful",
  "code"    : "OData",
  "exploit" : "relational",
  "get"     : "results"
}
```

Data Without Borders

- **Customers want to access Data from non traditional platforms like Mobile platforms, from mobile apps (Android, Windows, iOS, etc.)**
  - And in some cases want to publish access to data as well to 3rd parties
- **Mobile and Cloud developers expect direct HTTP and JSON access to data without requiring Database Drivers**
- **OData is standards based way for our customers to leverage Db2 modern Cloud and Mobile Application Models**

OData (Open Data Protocol) is an ISO/IEC approved, OASIS standard that defines a set of best practices for building and consuming RESTful APIs.

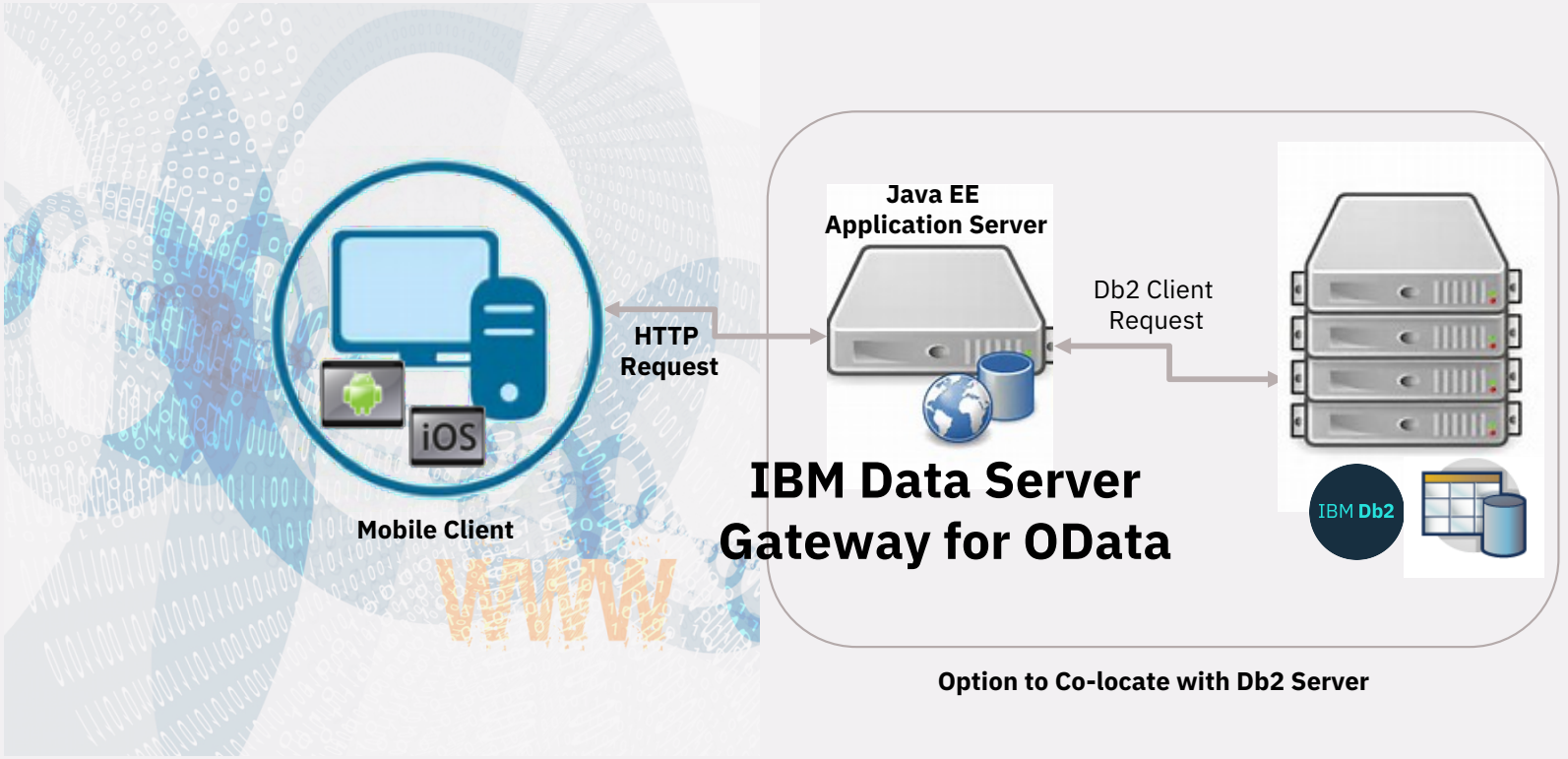
OData helps you focus on your business logic while building RESTful APIs without having to worry about the various approaches to define request and response headers, status codes, HTTP methods, URL conventions, media types, payload formats, query options, etc.

OData also provides guidance for tracking changes, defining functions/actions for reusable procedures, and sending asynchronous/batch requests.

http://www.odata.org/

Open Data Protocol (OData) Gateway

Removing the Need for Database Drivers and APIs



IBM DS Gateway for OData Code

- ***IBM Data Server Gateway for OData*** **is released in two formats**

1. Integrated with WebSphere Liberty profile

- IBM Data Server Gateway for OData release is integrated with WebSphere Liberty Profile Application server
- Configured with a default server that enables you to quickly start the server and be productive

2. Web Application Resource (WAR) file

- IBM Data Server Gateway for OData is also released as a WAR file that can be deployed to any Java servlet container (Application server) of your choice

IBM DS Gateway for OData Download

- **Download location:**
  - https://www-945.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~Information%2BManagement&product=ibm/Information+Management/IBM+Data+Server+Client+Packages&release=11.1.*&platform=Linux&function=fixId&fixids=*odata*FP001*&includeSupersedes=0&source=fc
  - Or use the quick link! ibm.biz/Db2OData

☐	1	fix pack: DSClients--ibm_gateway_server_for_odata_war-1.0.0-FP001 IBM Data Server Gateway for ODATA WAR File V1.0.0 Fix Pack 1	2016/12/13
☐	2	fix pack: DSClients--ibm_gateway_server_for_odata-1.0.0-FP001 IBM Data Server Gateway for ODATA V1.0.0	2016/12/13

Installation Instructions

- **Create a Directory to hold the IBM Data Server Gateway for OData file**
  - `mkdir db2odata` or `md db2odata`
- **Copy the zip file into the newly created directory and unzip it**
  - `pkunzip` or `tar -xvf` the file
  - This will create a directory called v1.0.0_ibm_gateway_server_for_odata
- **Change the Permission bits on the following files (UNIX, Linux)**
  - `chmod 555 start_ibm_gateway_server_odata`
  - `chmod 555 stop_ibm_gateway_server_odata`
  - `chmod 555 server/bin/server`

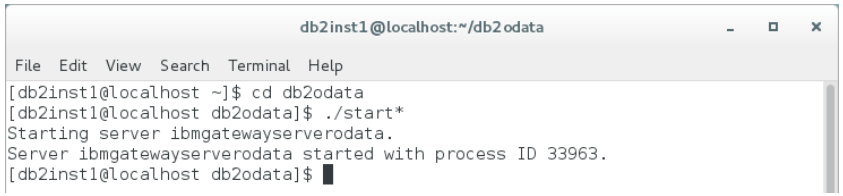
Starting and Stopping the Application Server

- **To Start the server:**
  - In Windows run `start_ibm_gateway_server_odata.bat`
  - In Linux, AIX or Mac run `./start_ibm_gateway_server_odata`
- **Make sure the permission bits have been updated or otherwise you will get a Permission denied error!**
- **To Stop the server:**
  - In Windows run `stop_ibm_gateway_server_odata.bat`
  - In Linux, AIX or Mac run `./stop_ibm_gateway_server_odata`
- **Note – The service requires that port 9080 be open on your machine**
  - The following command are examples only and will be different depending on your O/S and firewall standards

```
sudo iptables -I INPUT -p tcp --dport 9080 -j ACCEPT
sudo service iptables save
```

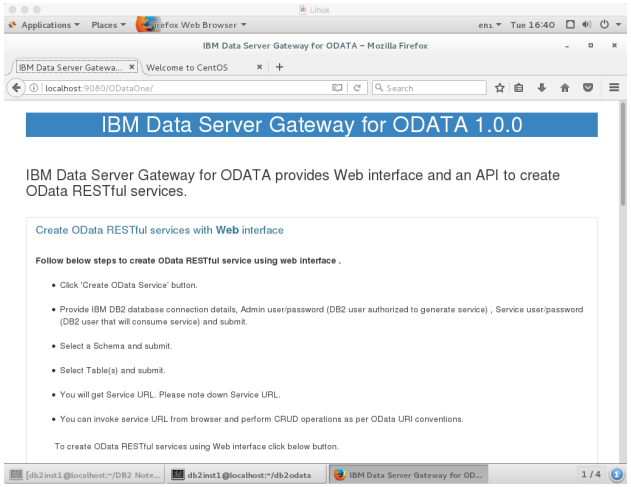
Validating Installation

- **When starting the service, you should see the following output:**



- **After the service has started, use a web browser to view the service:**

  `http://localhost:9080/ODataOne/`



IBM DS Gateway for OData Support

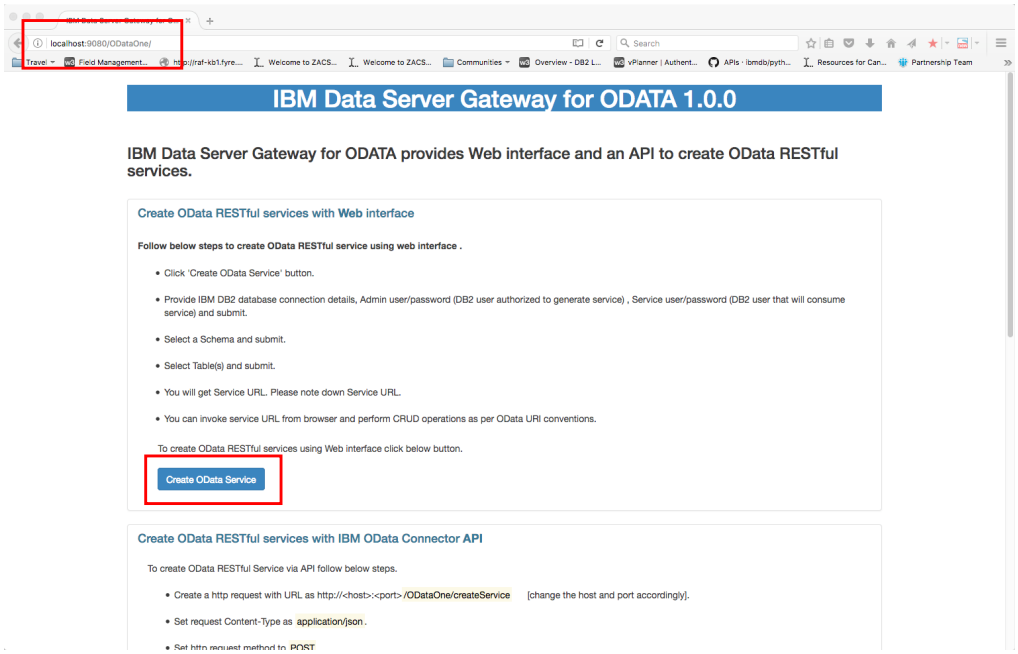
- **Persistence of REST API endpoints/services are kept in an Apache Derby database**
  - Small footprint database used for mapping REST API calls to resources in the Db2 database
- **Authentication and Authorization are used during OData REST API endpoints and service generation**
  - Client needs to authenticate mobile user using whatever local protocol is applicable
    - Password
    - Fingerprint
    - Two-factor authentication
  - Authentication is not handled as part of the OData implementation
- **SSL/TLS supported**
  - Secure transmission of requests and retrieval of data

Generating a Service Using Web Console(1)

- **Use the following URL to access the service console:**

  http://localhost:9080/ODataOne/

- **Click on the Create OData Service button**

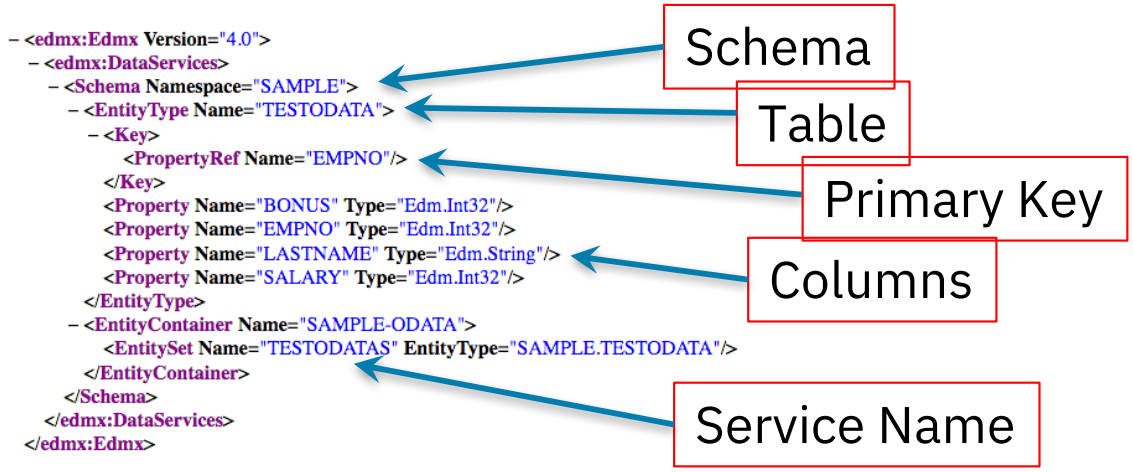


Metadata Information

- **The service root URL can be used to view metadata information about the tables defined to it:**

  http://localhost:9080/ODataOne/ODataService/SAMPLE-db2ab6eeb40d4a75a525333ee610fb50/$metadata

- **Detailed information returned about tables, primary keys, and individual column attributes are returned in an XML document**(PIA)



Metadata Data Types

- **The OData data types do not map exactly to Db2 data types**
- **The following table gives the OData keyword value (Edm.type) and the data type it represents**

Data Type	Contents
Binary	Binary data
Boolean	Binary-valued logic
Byte	Unsigned 8-bit integer
Date	Date without a time-zone offset
Decimal	Numeric values with fixed precision and scale
Double	IEEE 754 binary64 floating-point number (15-17 decimal digits)
Duration	Signed duration in days, hours, minutes, and (sub)seconds
Guid	16-byte (128-bit) unique identifier
Int16	Signed 16-bit integer
Int32	Signed 32-bit integer
Int64	Signed 64-bit integer
SByte	Signed 8-bit integer
Single	IEEE 754 binary32 floating-point number (6-9 decimal digits)
String	Sequence of UTF-8 characters
TimeOfDay	Clock time 00:00-23:59:59.999999999999

Generating a Service Using RESTful Calls(1)

- **Create a JSON string with the following content**

```
{
 "database":"<database_name>",
 "host":"<database_host>", "port":"<database_port>",
 "db2AdminUser":"<db2_admin_user>",
 "db2AdminPassword":"<db2_admin_password>",
 "db2ServiceUser":"<db2_service_user>",
 "db2ServicePassword": "<db2_service_password>",
 "ssl": false,
 "schema": "<schema_name>",
 "tablenames": ["table1", "table2",…]
}
```

- **Send a POST request with the above string to:**

```
http://<host>:<port>/ODataOne/createService
```

- **Notes:**
  - Content-Type must be set to application/json
  - The JSON string must be in the request header

Generating a Service Using RESTful Calls(2)

- **OData service will return the following information**

```
Your service is created successfully !

Service Root URL: http://<server_host>:<port>/ODataOne/ODataService/sample-
a1085cdc671c4b5dbd969c3be3e411e8

Service Metadata URL: http://<server_host>:<port>/ODataOne/ODataService/sample-
a1085cdc671c4b5dbd969c3be3e411e8/$metadata
```

- **The Service Root URL is used for subsequent OData calls**

RESTful Support

Simplifying Access to Data Sources from Mobile Devices

- **The IBM OData Gateway provides a RESTful service that provides for the following operations:**
  - GET (SELECT)
  - POST (INSERT)
  - PUT (UPDATE a full row)
  - PATCH (UPDATE selected columns in a row)
  - DELETE (Delete a row)

```
url = http://localhost:9080/ODataOne/ODataService/SAMPLE-db2ab6eeb40d4a75a525333ee610fb50

header = {
         'Content-Type':'application/json',
         'Accept':'application/json'
         }

OData = "/EMPLOYEES?$select=LASTNAME,SALARY&$filter=SALARY gt 50000"

RESTful.get(url + OData, headers=header)
```

OData Sample


```
SELECT LASTNAME, SALARY
FROM   EMPLOYEE
WHERE  SALARY > 50000
```

Service Address

```
url = http://localhost:9080/ODataOne/ODataService/SAMPLE-db2ab6eeb40d4a75a525333ee610fb50
```

```
header = {
         'Content-Type':'application/json',
         'Accept':'application/json'
         }
```

Handshake

```
Odata = "/EMPLOYEES?$select=LASTNAME,SALARY&$filter=SALARY gt 50000"
```

OData Request

```
RESTful.get(url + OData, headers=header)
```

OData RESTful Syntax

- **Header file**
  - The POST, GET, PATCH, PUT, and DELETE RESTful all use a common header that tells the service how the content is formatted (JSON)
    `{'Content-Type':'application/json','Accept':'application/json'}`
- **Service URL**
  - URL refers to the service URL that allows access to the table(s) that you have registered in the IBM OData Gateway
- **Service Name**
  - Db2 table name is appended with the letter 'S' becomes the service name(PIA)
  - i.e. `EMPLOYEE -> EMPLOYEES`
- **OData command is always appended to the end of the Service URL**
  - `RESTful.get(url + OData, headers=header)`
- **A ? is added to the end of the service name if there are optional parameters in the OData command**

OData GET (SELECT) Command

- **The OData GET command will return data from one table or view**
- **The select is accomplished through the RESTful GET request:**
  - `RESTful.get(url + OData, headers=header)`
- **The OData request contains the name of the table service and the optional parameters used to filter and format the rows**

```
/EMPLOYEES?$select=LASTNAME,SALARY&$filter=SALARY gt 50000
```

OData GET Syntax

- **The GET command will return data from one table**
  - There is no ability to join tables with the current implementation of OData
  - You can use views in Db2 as a way to join tables and use that in your selects
  - The table name in the OData command always has an "S" appended to it since it refers to the service name
- **Syntax:**
  - `/<table>S(primary key)[?options]`
  - `/<table>S[?options]`
- **Options:**
  - `$select=column1, columns, …`
  - `$filter=logic`
  - `$top=x [number of rows]`
  - `$format=json`
  - `$count=true`

OData GET Options(1)

- **The Options in a GET clause can be used in any order but must be separated by an ampersand (&) character**

```
$select=NAME&$top=1
```

- **Options are case and whitespace sensitive**
  - You must place spaces between arguments in the $filter option
- **$select**
  - This clause specifies which columns you want retrieved from the table
  - You must use the UPPERCASE name of the column if the table was created using the traditional Db2 naming rules (i.e. the name is folded to uppercase)
  - The order makes no difference in how they are returned
  - Columns are always returned in alphabetical order, regardless of how you specify them in the command(PIA)
  - Commas separate the column names with no extra spaces
  - Example: `$select=LASTNAME,FIRSTNME`

OData GET Options(2)

- **$top=x**
  - Limits the result set to the first "x" rows from the top
  - Example: `$top=5`
- **$format=json**
  - Return the result as JSON records (default)
- **$count=true**
  - Return the count of records found
  - This also returns the actual rows, so this isn't the same as a Db2 COUNT(*) clause(PIA)
  - Best practice would be to limit the output to one column in the answer set to minimize the data returned if all you really wanted was the count
  - Using $top will result in $count reflecting ONLY the records returned, not the true count of the table(PIA)

OData GET Options(3)

- **$filter**
  - Used to apply filtering logic to the rows being returned
- **Comparison operators**
  - `lt` (<), `gt`(>), `le` (<=), `ge` (>=), `ne` (<>), `eq` (=)
- **Logical Operators**
  - `and`, `or`, `(`, `)`
  - No "not" available(PIA)
- **Mathematical Operators**
  - No math operators are allowed, including the unary minus operator(PIA)
  - Use a VIEW to do these calculations i.e. `COMM + BONUS > SALARY * .20`
- **String Operator**
  - `contains(field,pattern)`
  - Similar to the LIKE clause except that it searches for the pattern anywhere

OData GET Examples(4)

- Select employee '000010' with all columns returned (using primary key)

```
/EMPLOYEES('000010')
```

- Select employee '000010' with their first and last name retrieved

```
/EMPLOYEES?$select FIRSTNME,LASTNAME&$filter=EMPNO eq '000010'
```

- Return the count of all employees (which includes their records too!)

```
/EMPLOYEES?$count=true
```

- Select the LASTNAME and SALARY of the employees whose salary is greater than 50000 and limit the output to the first 5

```
/EMPLOYEES?$select=LASTNAME,SALARY&$filter=SALARY gt 50000&$top=5
```

- Select the LASTNAME and BONUS of the employees who last name has the letters 'AA' in it, and whose bonus is equal to 1000

```
/EMPLOYEES?$select=LASTNAME,BONUS&$filter=contains(LASTNAME,'AA') and BONUS eq 1000
```

- Select the employees who work in department A00 or work in D01 and have a salary > 60000

```
/EMPLOYESS?$filter=WORKDEPT eq 'A00' or (WORKDEPT eq 'D01' and
SALARY gt 60000)
```

OData POST Format(1)

- **The OData POST command will INSERT a single row into a table**
- **An INSERT is accomplished through the RESTful POST request:**
  - `RESTful.post(url + OData, headers=header, parameters)`
- **The OData request contains the name of the table service**

  `/EMPLOYEES`
- **The OData request is appended to the end of the URL:**

  `http://localhost:9080/ODataOne/ODataService/SAMPLE-db2ab6eeb40d4a75a525333ee610fb50/EMPLOYEES`

OData POST Format(2)

- **The parameter field is a JSON record that has the names and contents of all of the columns**

```
{
  'EMPNO': 1,
  'LASTNAME': 'Fred',
  'SALARY': 10000,
  'BONUS': 1000
}
```

- **The data is checked by Db2 to make sure it has the proper format**
  - If a column name is not provided, the default value will be NULL
  - If the column is defined as NOT NULL (without a default), then an error will be returned

```
SQL0407N Assignment of a NULL value to a NOT NULL column "TBSPACEID=2, TABLEID=22, COLNO=2" is not allowed. SQLSTATE=23502
SQLCODE=-407
```

OData PATCH Format(1)

- **The OData PATCH command will UPDATE fields in a single row in a table**
- **An UPDATE is accomplished through the RESTful PATCH request:**

```
RESTful.patch(url + OData, headers=header, parameters)
```

- **The OData request contains the name of the table service and the primary key of the record to be updated**

```
/EMPLOYEES('000010')
```

- **You must provide a unique key for the record to update**
  - Multi-row updates are not supported with this syntax
- **You can replace all of the values in a record by using the PUT command and providing all of the fields and values in the parameter list**

OData PATCH Format(2)

- **The parameter field is a JSON record that has the names and contents of all of the columns that you want to change**

```
{
  'SALARY': 10000,
  'BONUS': null
}
```

- **The data is checked by Db2 to make sure it has the proper format**
  - If the column is defined as NOT NULL (without a default), then an error will be returned if you try to set it to NULL

SQL0407N Assignment of a NULL value to a NOT NULL column "TBSPACEID=2, TABLEID=22, COLNO=2" is not allowed. SQLSTATE=23502 SQLCODE=-407

OData DELETE Format

- **The OData DELETE command will delete a single row in a table**
- **An DELETE is accomplished through the RESTful DELETE request:**

```
RESTful.delete(url + OData, headers=header, parameters)
```

- **The OData request contains the name of the table service and the primary key of the record to be deleted**

```
/EMPLOYEES('000010')
```

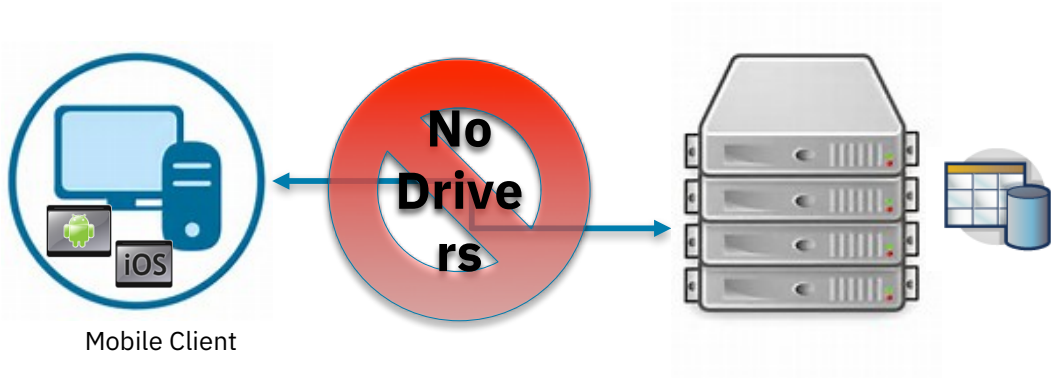
- **You must provide a unique key for the record to delete**
  - Multi-row deletes are not supported with this syntax

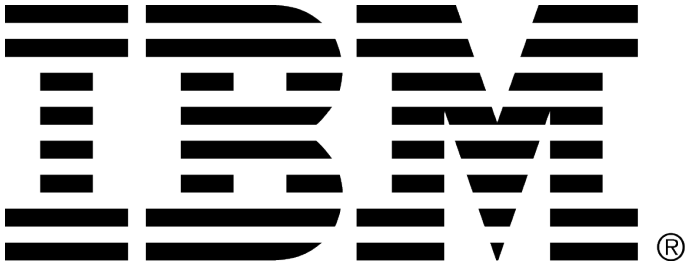
Sample Db2 OData Notebooks

- **Jupyter notebooks are available to try OData functions against Db2 11**
  - Includes a Jupyter notebook extension (%odata) to generate a valid service URL for requests to a Db2 Database
  - Translates SQL syntax for INSERT, UPDATE, DELETE, and SELECT commands into equivalent OData commands
  - https://github.com/DB2-Samples/db2odata.git
- **An introduction to the OData gateway is found in the following developerWorks article:**
  - http://ibm.biz/Db2ODataArticle
- **The code can be obtained through the following link:**
  - ibm.biz/Db2OData
- **Videos**
  - https://www.youtube.com/watch?v=z06bl_K6Ckc
  - https://www.youtube.com/watch?v=dHVBiTphoCM

Summary

- The *IBM Data Server Gateway for OData* (OData Gateway) is a standards-based way to leverage DB2 in Cloud and Mobile applications
- No requirement for a database driver with data access available from non-traditional platforms like Mobile platforms (Android, Windows, iOS etc.) using REST API endpoints
- Reduces the overall application complexity by eliminating the need to write middle-tier components that do nothing but transform data into JSON or XML




IBM

Please note