Db2 for z/OS Modern Application: Watson, ML, REST, Spark and more

Jane Man

IBM

Session code: G12

Wednesday, 04.10.2017 13:40-14:40

Platform: Db2 for z/OS

Objectives

- To introduce how to build a cognitive Db2 solution using Waston
- To illustrate how to do Machine Learning on Db2 data without a GUI
- To discuss how to do Spark analytics
- To show how to use Db2 as REST provider using Db2 native REST service
- To demonstrate how to use Db2 as REST consumer using JSON feature in Db2

Please note

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM's sole discretion.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

- Db2 as REST provider using Db2 native REST service
- Db2 as REST consumer using JSON feature in Db2
- Cognitive Db2 solution using Waston
- Perform Machine Learning on Db2 data without a GUI
- Spark analytics

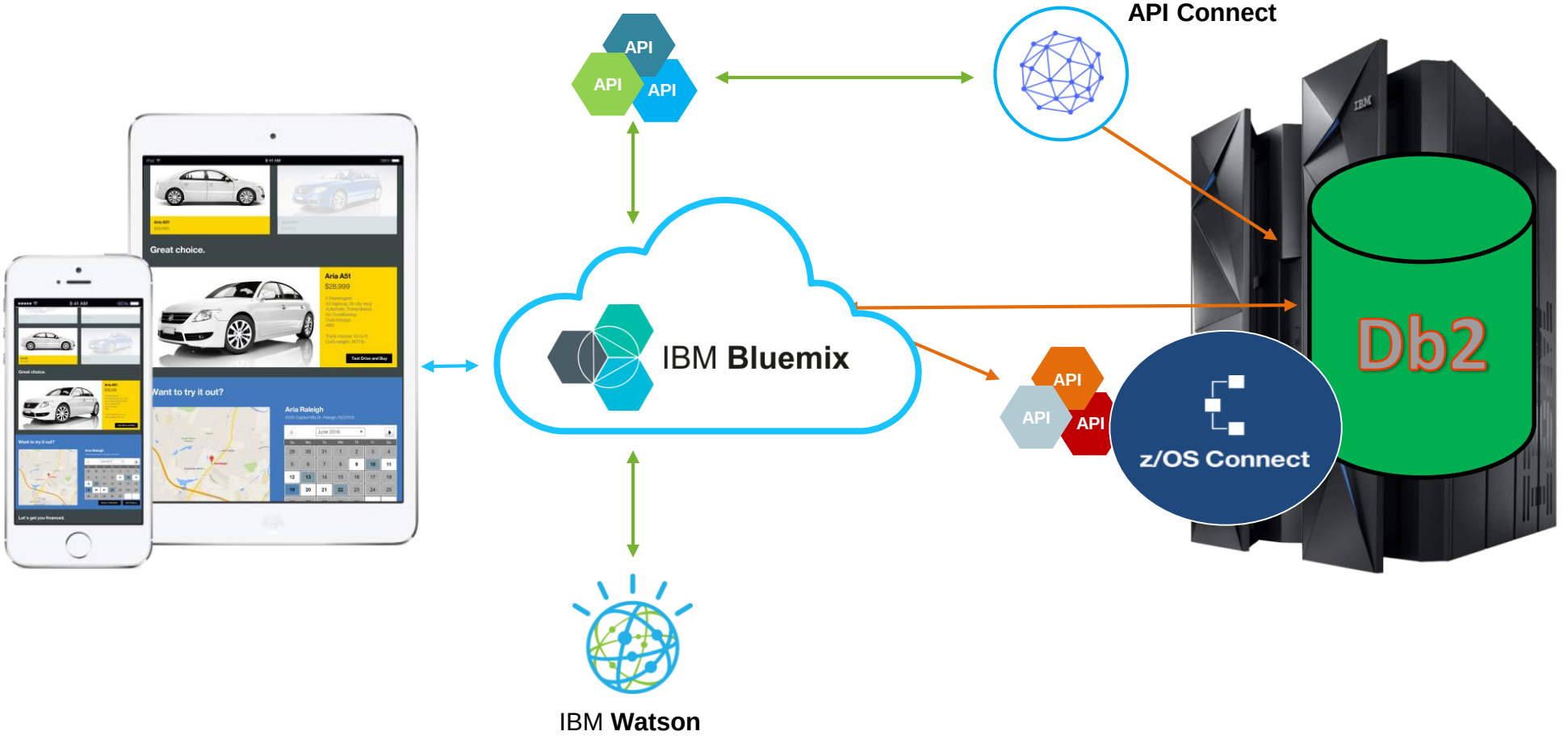
Agenda

- Db2 as REST provider using Db2 native REST service
- Db2 as REST consumer using JSON feature in Db2
- Cognitive Db2 solution using Waston
- Perform Machine Learning on Db2 data without a GUI
- Spark analytics

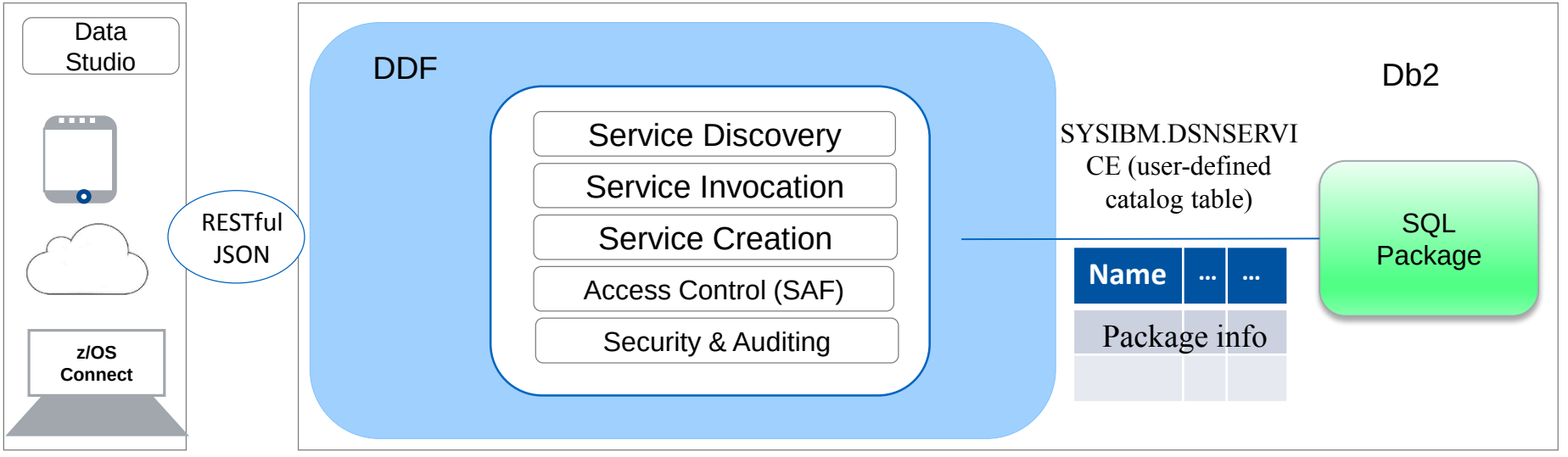
POST https://*<host>*:*<port>*/services/simpleSelect1
{ "LOCATION": "*<location>*" }

540 million RESTful requests per hour

- Db2 is *now* a RESTful service provider (V11 PI66828)
- Enabling new business value for your enterprise data
- Modernizing using the power of SQL
- Unleashing Db2 data for the API Economy

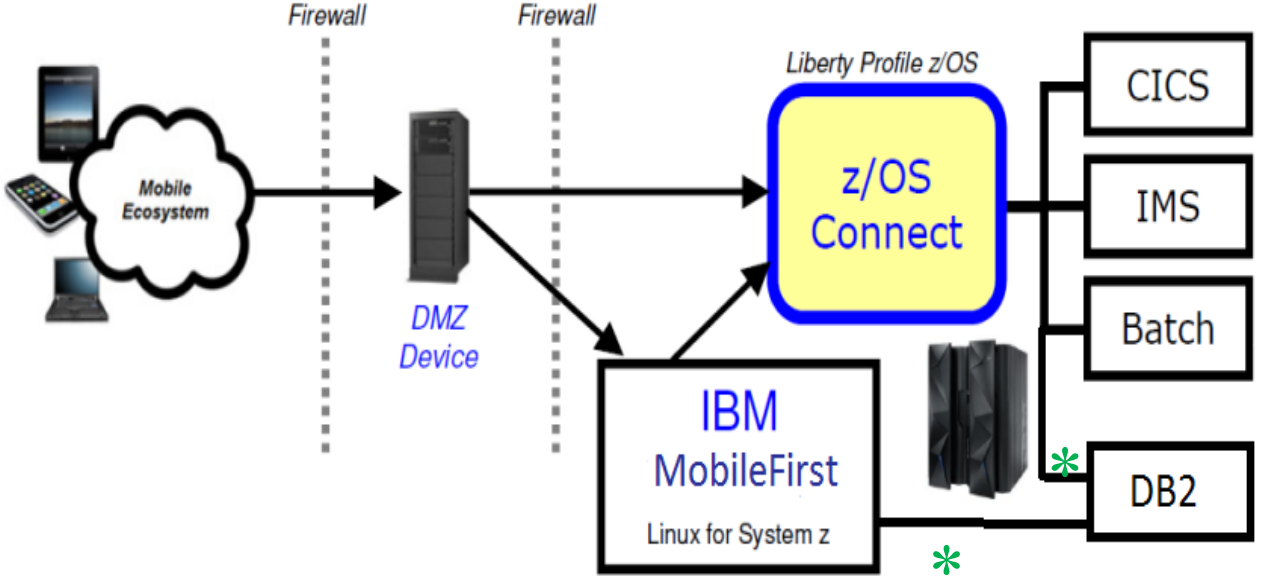

API Connect
API
API
API
Great choice.
Aria A51
$28,999
Test Drive and Buy
Want to try it out?
Aria Raleigh
IBM Bluemix
API
API
API
z/OS Connect
Db2
IBM Watson

Db2 as a Service Provider – Solution Overview



Db2 User Created Service – A single SQL statement (INSERT, UPDATE, DELETE, SELECT, CALL) bound into a package.
Db2 Management Services – Create, discover and invoke a DB2 user created service using a REST API.
DDF Interface – Processes HTTP requests and replies, performs REST/JSON message formatting invokes as a static SQL statement
Availability – Utilizes existing DDF capability including thread pooling, profiling and enclave classification.
Leverages existing package lifecycle – Services are in the catalog tables and saved in the directory.
Security, accounting, statistics, and auditing – Uses existing DDF infrastructure. Supporting classification, profiling, and thread pooling
z/OS Connect EE Integration – Uses the z/OS Connect EE REST Client to allow exporting of DB2 svs into z/OS Connect rest client

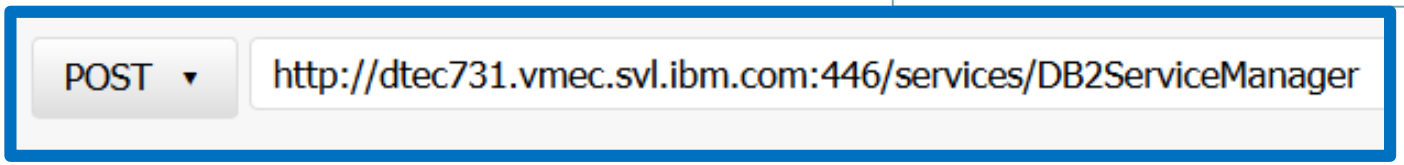
Simplified End-to-End Architecture for Mobile and Cloud Application invoking z Services using APIs



Create a Db2 REST service

1. Customize and Run DSNTIJRS to create Db2 RESTful service database and table
2. Authorize user to access services
   Access to Db2 REST is protected by a SAF DSNR class <ssid>.REST profile
3. Create a Db2 REST service

```
//RESTDSNR JOB CLASS=A,MSGCLASS=H,REGION=0M,
//      USER=RACF000,PASSWORD=PASSWORD
//*************************************************
//* RACF COMMANDS TO ADD DSNR CLASS PROFILES FOR REST
//*************************************************/
/STEP1  EXEC PGM=IKJEFT01
//SYSRACF   DD DSN=SYS1.RACF,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN DD *
RDEFINE DSNR (DB2A.REST) UACC(READ)
```



Headers

The headers specified below will be added to any sent by the browser or specified elsewhere.

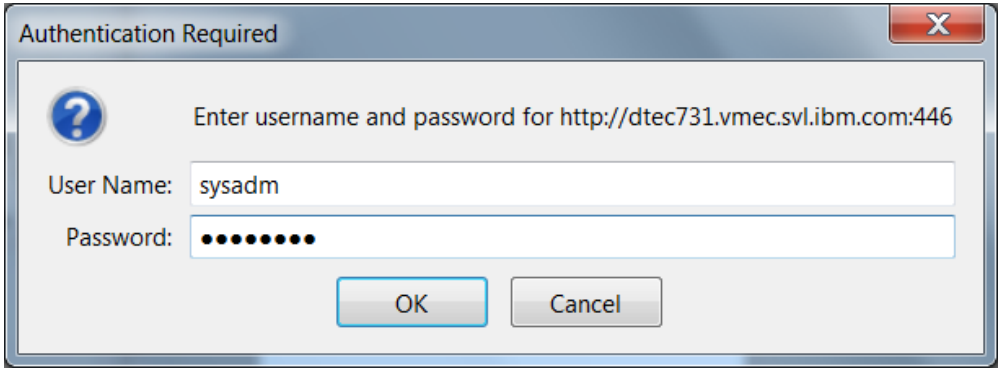
Name	Value
Accept	application/json
Content-Type	application/json
Accept-Charset	UTF-8

Enter the data and its corresponding MIME type below.

application/json

```
{
  "requestType": "createService",
  "sqlStmt": "SELECT SUBSTR(STRING,1,60) as STRING from
SYSIBM.SYSXMLSTRINGS WHERE STRINGID= ?",
  "serviceName": "selectSYSXMLStrings",
  "description": "Select string from XML string id"
}
```

Backend details : create a Db2 REST service



Output of creating Db2 REST service

```
{
 "StatusCode": 201,
  "StatusDescription": "DB2 Rest Service selectSYSXMLStrings was created
successfully.",
  "URL":
"http://dtec731.vmec.svl.ibm.com:446/services/SYSIBMSERVICE/selectSYSXMLStrings"
}
```

Backend details : Discover all Db2 REST service (optional)

GET http://dtec731.vmec.svl.ibm.com:446/services


```
{       "DB2Services": [{
                "ServiceName": "DB2ServiceDiscover",
                "ServiceCollectionID": null,
                "ServiceDescription": "DB2 service to list all available services.",
                "ServiceProvider": "db2service-1.0",
                "ServiceURL": "http:\/\/dtec731.vmec.svl.ibm.com:446\/services\/DB2ServiceDiscover"
        }, {
                "ServiceName": "DB2ServiceManager",
                "ServiceCollectionID": null,
                "ServiceDescription": "DB2 service to create, drop, or alter a user defined
service.",
                "ServiceProvider": "db2service-1.0",
                "ServiceURL": "http:\/\/dtec731.vmec.svl.ibm.com:446\/services\/DB2ServiceManager"
        }, {
                "ServiceName": "selectSYSXMLStrings",
                "ServiceCollectionID": "SYSIBMSERVICE",
                "ServiceDescription": "Select string from XML string id",
                "ServiceProvider": "db2service-1.0",
                "ServiceURL":
"http:\/\/dtec731.vmec.svl.ibm.com:446\/services\/SYSIBMSERVICE\/selectSYSXMLStrings"
        }]
}
```

Backend details : create a Db2 REST svs (cont'd)

4. Get details for the service

Output of creating Db2 REST service

```
{
 "StatusCode": 201,
 "StatusDescription": "DB2 Rest Service selectSYSXMLStrings was created successfully.",
 "URL":
"http:\/\/dtec731.vmec.svl.ibm.com:446\/services\/SYSIBMSERVICE\/selectSYSXMLStrings"
}
```

GET http://dtec731.vmec.svl.ibm.com:446/services/SYSIBMSERVICE/selectSYSXMLStrings

INPUT

```
{"selectSYSXMLStrings": {
                    "serviceName": "selectSYSXMLStrings",
                    "serviceCollectionID": "SYSIBMSERVICE",
                    …..
                    "serviceStatus": "started",
                    "RequestSchema": {…
                              "properties": {
                                        "P1": {
                                                  "type": ["null", "integer"],
                                                  "multipleOf": 1,
                                                  "minimum": -2147483648,
                                                  "maximum": 2147483647,
                                                  "description": "Nullable INTEGER"
                                        }
                              },
                              "required": ["P1"],….
```

Backend details : create a Db2 REST service (cont'd)

Details on `selectSYSXMLStrings` service

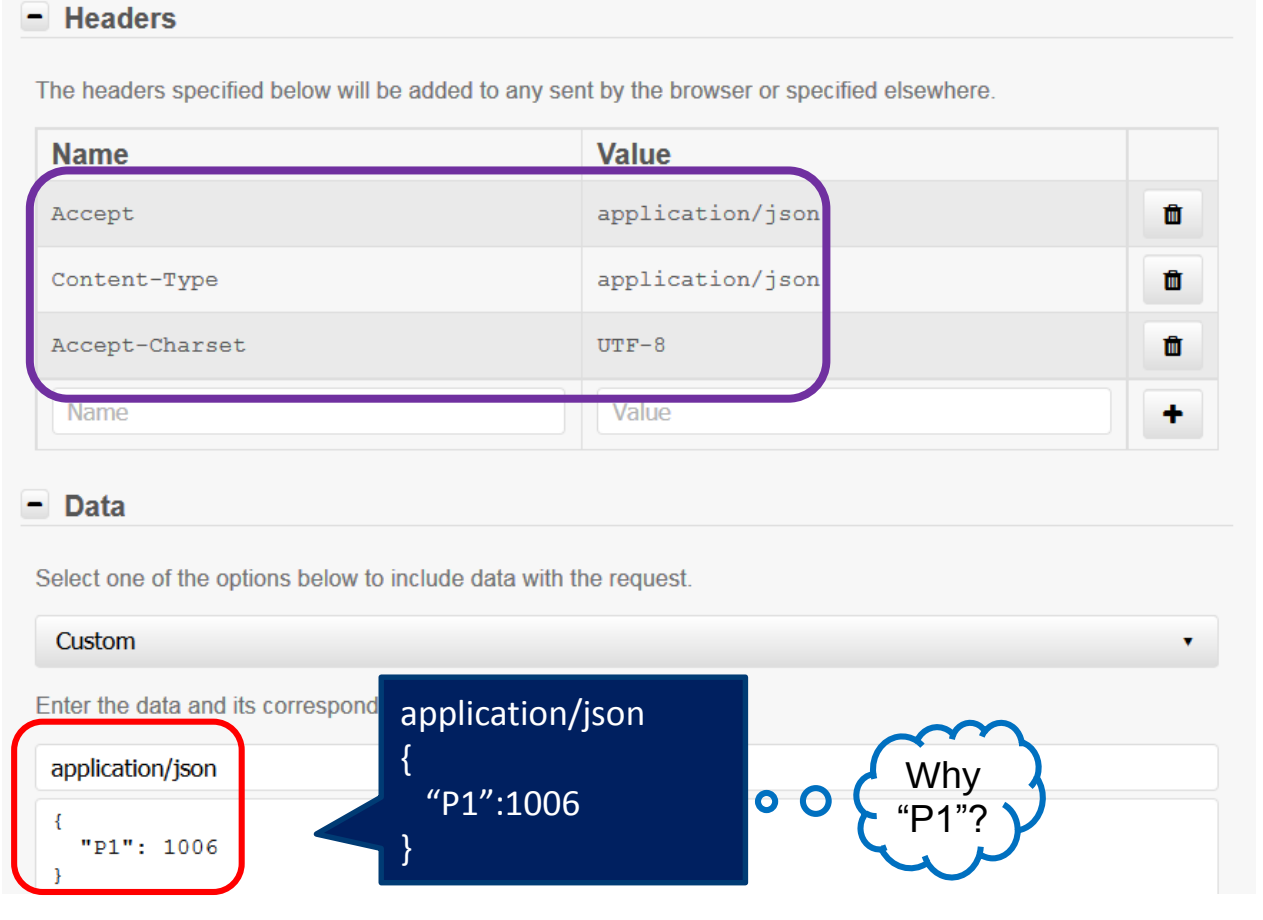
```
"ResponseSchema": {
        "$schema": "http:\/\/json-schema.org\/draft-04\/schema#",
        "type": "object",
        "properties": {
            "ResultSet Output": {
                "type": "array",
                "items": {
                    "type": "object",
                    "properties": {
                        "STRING": {
                            "type": "string",
                            "description": "CHAR(60)"
                        }
                    },
                    "required": ["STRING"],….
```

OUTPUT

Backend details : create a Db2 REST service (cont'd)

5. Invoke a service

POST http://dtec731.vmec.svl.ibm.com:446/services/SYSIBMSERVICE/selectSYSXMLStrings



OUTPUT

```
{
"ResultSet Output": [{
 "STRING": "space"}],
"StatusCode": 200,
"StatusDescription":
"Execution Successful"
}
```

Agenda

- Db2 as REST provider using Db2 native REST service
- Db2 as REST consumer using JSON feature in Db2
- Cognitive Db2 solution using Waston
- Perform Machine Learning on Db2 data without a GUI
- Spark analytics

JSON_VAL Built-in function

To extract and retrieve JSON data into SQL data types from BSON.

>>-JSON_VAL—(--*json-value*--,--*search-string*--,--*result-type*--)--------><

The JSON_VAL function returns an element of a JSON document identified by the JSON field name specified in *search-string*. The value of the JSON element is returned in the data type and length specified in *result-type*

Example:

```
JSON_VAL(DATA,'PO.customer.@cid', 'i:na')
```

Result-type	Function return type / length
'n'	DECFLOAT(34)
'i'	INTEGER
'l'	BIGINT
'f'	DOUBLE
'd'	DATE
'ts'	TIMESTAMP
't'	TIME
's:n'	VARCHAR (n)
'b:n'	VARCHAR(n) FOR BIT DATA
'u'	INTEGER / 4

PI39003 remove the requirement that 1st parameter has to be a BLOB column

SYSTOOLS.JSON_TABLE

This function returns array of elements in JSON data.

Example:

```
SELECT X.* FROM JSONPO,
TABLE(SYSTOOLS.JSON_TABLE(DATA,
'PO.items.item', 's:200')) X
```

```
'{"PO":{"@id": 101,
        "@orderDate": "2014-11-18",
        "customer": {"@cid": 999},
        "items": {
          "item": [{"@partNum": "872-AA",
                    "productName": "Lawnmower",
                    "quantity": 1,
                    "USPrice": 149.99,
                    "shipDate": "2014-11-20"
                   },
                   {"@partNum": "945-ZG",
                    "productName": "Sapphire Bracelet",
                    "quantity": 2,
                    "USPrice": 178.99,
                    "comment": "Not shipped"
                   }
                  ]
                }
      }
```

Output:

```
TYPE    VALUE
3       {@partNum:"872-
AA",productName:"Lawnmower",quantity:1,USPrice:149.990000,shipDate:"2014-11-20"}
3       {@partNum:"945-ZG",productName:"Sapphire
Bracelet",quantity:2,USPrice:178.990000,comment:"Not shipped"}
```

JSON_TABLE + JSON_VAL : convert JSON to Relational

Example:

```
SELECT
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), '@partNum', 's:10') as "@partNum",
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'productName', 's:20') as  "productName",
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'quantity', 'i') as "quantity",
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'USPrice', 'f') as "USPrice",
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'shipDate', 's:20') as "shipDate"
FROM JSONPO,
    TABLE(SYSTOOLS.JSON_TABLE(DATA, 'PO.items.item', 's:200')) X
```

VALUE

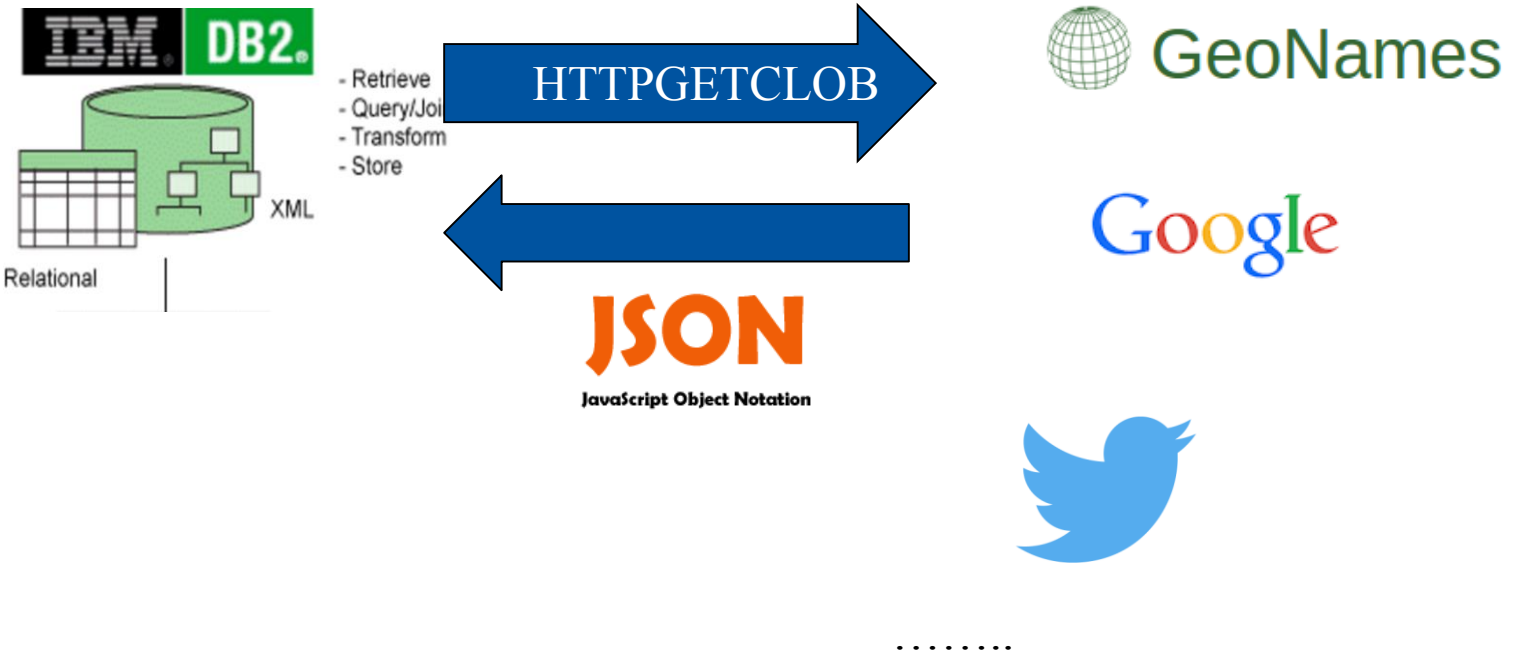
{@partNum:"872-AA",productName:"Lawnmower",quantity:1,USPrice:149.990000,shipDate:"2014-11-20"}
@partNum:"945-ZG",productName:"Sapphire Bracelet",quantity:2,USPrice:178.990000,comment:"Not shipped"}

Output:

```
@partNum    productName          quantity    USPrice          shipDate
872-AA      Lawnmower            1           149.99           2014-11-20
945-ZG      Sapphire Bracelet    2           178.99           <null>
  2 record(s) selected
```

Require PI39003

Db2 for z/OS as REST service consumer



Db2 for z/OS as REST service consumer

GeoNames

```
SELECT DB2XML.HTTPGETCLOB(
  CAST ('http://ws.geonames.org/countryInfo?lang=' ||
          DB2XML.URLENCODE('en','') ||
        '&country=' ||
          DB2XML.URLENCODE('us','') ||
        '&type=JSON' AS VARCHAR(255)),
  CAST(NULL AS CLOB(1K)))
FROM SYSIBM.SYSDUMMY1
```

Output (in JSON)

```
{"status":{
   "message":"…..",
   "value":10}}
```

```
with tablea as (
SELECT DB2XML.HTTPGETCLOB(
  CAST ('http://ws.geonames.org/countryInfo?lang=' ||
          DB2XML.URLENCODE('en','') ||
        '&country=' ||
          DB2XML.URLENCODE('us','') ||
        '&type=JSON' AS VARCHAR(255)),
  CAST(NULL AS CLOB(1K))) as DATA
FROM SYSIBM.SYSDUMMY1)
select JSON_VAL(SYSTOOLS.JSON2BSON(DATA),
'status.value', 'i')
from tablea
```

```
10
  1 record(s) selected
```

Db2 for z/OS as REST service consumer - Yahoo REST APIs

YAHOO!

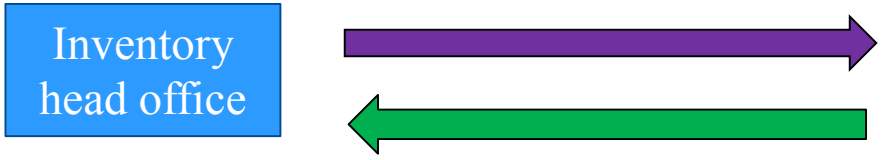



```
With TABLEA as (
SELECT DB2XML.HTTPGETCLOB( CAST
('https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20weather.fore
cast%20where%20woeid%20in%20%28select%20woeid%20from%20geo.places%281%29%20whe
re%20text%3D%22fremont%2C%20ca%22%29&format=json&env=store%3A%2F%2Fdatatables.
org%2Falltableswithkeys' AS VARCHAR(255)),
  CAST(NULL AS CLOB(1K))) AS DATA
FROM SYSIBM.SYSDUMMY1)
select
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'date', 's:20') as date,
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'high', 'i') as temperature_high,
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'low', 'i') as temperature_low,
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'text', 's:20') as description
from tablea, TABLE(SYSTOOLS.JSON_TABLE(SYSTOOLS.JSON2BSON(DATA),
'query.results.item.forecast', 's:200')) X
```

Array output, use JSON_TABLE

```
DATE                 TEMPERATURE_HIGH TEMPERATURE_LOW DESCRIPTION
06 Aug 2016          75               54              Partly Cloudy
07 Aug 2016          76               53              Mostly Sunny
08 Aug 2016          78               53              Mostly Sunny
09 Aug 2016          80               56              Mostly Sunny
10 Aug 2016          79               57              Mostly Sunny
11 Aug 2016          77               56              Mostly Sunny
…
  10 record(s) selected
```

Db2 for z/OS as REST service consumer – inventory count




```
{
  "Location": 123,
  "Date": "2014-11-18",
  "inventory":
   [{ "itemid": "833",
       "available_qty": 23
    },
    { "itemid": "999",
      "available_qty": 44
    },…..
   ]
}
```

San Francisco branch

Cupertino branch

Newark branch

Los Angeles branch

San Jose branch

Atlanta branch

New York city branch

.....

Db2 for z/OS as REST service consumer – inventory count

Inventory head office

```
with tablea as (
SELECT DB2XML.HTTPGETCLOB(
'http://…..',…
) as DATA
FROM SYSIBM.SYSDUMMY1)
SELECT
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'itemid', 's:10') as
    "itemid",
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'available_qty', 's:20') as
    "available_qty"
FROM  tablea,
  TABLE(SYSTOOLS.JSON_TABLE(SYSTOOLS.JSON2BSON(DATA), 'inventory',
's:200')) X
```

```
{
  "Location": 123,
  "Date": "2014-11-18",
  "inventory":
   [{ "itemid": "833",
      "available_qty": 23
    },
    { "itemid": "999",
      "available_qty": 44
    },…..
   ]
}
```

```
itemid      available_qty
833         23
999         44
  2 record(s) selected
```

Db2 for z/OS as REST consumer – inventory count - JOIN

ITEMTABLE

PART	DESCRIPTION
833	Lawnmower
999	Sapphire Bracelet

JOIN

```
{
  "Location": 123,
  "Date": "2014-11-18",
  "inventory":
   [{ "itemid": "833",
      "available_qty": 23
    },
    { "itemid": "999",
      "available_qty": 44
    },…..
   ]
}
```

```
with tablea as (
SELECT DB2XML.HTTPGETCLOB(
'http://…..',…
) as DATA
FROM SYSIBM.SYSDUMMY1),
breakdown (itemid, available_qty) as
(
SELECT
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'itemid', 's:10') as "itemid",
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'available_qty', 's:20') as
"available_qty"
FROM  tablea,
     TABLE(SYSTOOLS.JSON_TABLE(SYSTOOLS.JSON2BSON(DATA), 'inventory',
's:200')) X)
SELECT B.itemid, B.available_qty, I.description
FROM ITEMTABLE I, breakdown B
WHERE B.itemid = I.part
```

```
itemid      available_qty
833         23
999         44
  2 record(s) selected
```

Db2 for z/OS as REST service consumer – inventory count - JOIN

```
ITEMTABLE
PART        DESCRIPTION
833         Lawnmower
999         Sapphire Bracelet
```

From ITEMTABLE

```
itemid      available_qty
833         23
999         44
  2 record(s) selected
```

From HTTPGETCLOB

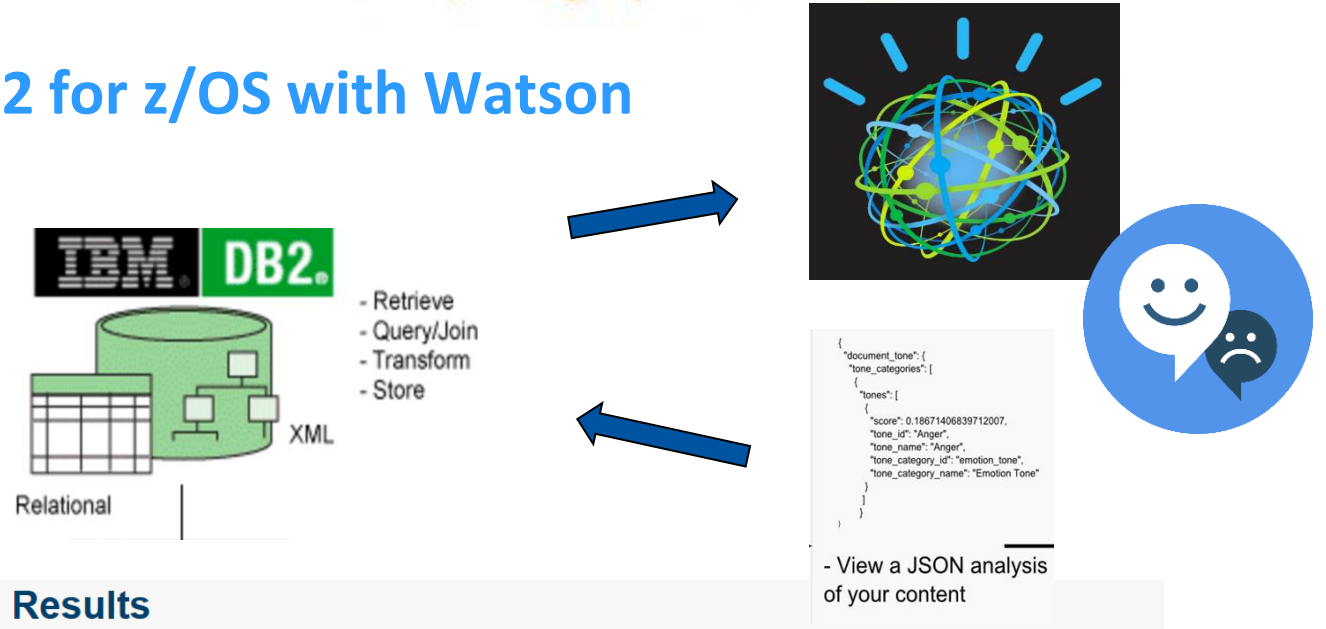
```
ITEMID      AVAILABLE_QTY          DESCRIPTION
833         23                     Lawnmower
999         44                     Sapphire Bracelet
  2 record(s) selected
```

RESULT

Agenda

- Db2 as REST provider using Db2 native REST service
- Db2 as REST consumer using JSON feature in Db2
- Cognitive Db2 solution using Waston
- Perform Machine Learning on Db2 data without a GUI
- Spark analytics

Db2 for z/OS with Watson



http://www.ibm.com/watson/developercloud/doc/tone-analyzer/case-study.shtml

What tone is it?

I am so happy today!

I have worked days and nights for three years without a salary raise!

I thought I can sleep in this session, but that crazy speaker keep asking us questions. Now she is asking us to read this long text and tell her what tone it is. How do I know? Ask a machine to do this….

???

Db2 for z/OS and Waston Tone Analyzer

```
WITH Authorization AS (
  SELECT  DB2XML.BASE64ENCODE(CAST('<userid>:<password>' AS
VARCHAR(64) CCSID 1208)) AS Authorization_Identity
FROM SYSIBM.SYSDUMMY1
)
SELECT DB2XML.HTTPGETCLOB(
  CAST ('https://gateway.watsonplatform.net/tone-
analyzer/api/v3/tone?version=2016-05-19&text=' ||
DB2XML.URLENCODE('I thought I can sleep in this session, but that
crazy speaker keep asking us questions. Now she is asking us to
read this long text and tell her what tone it is. How do I know?
Ask a machine to do this','') AS VARCHAR(255)),
  CAST('<httpHeader><header name="Authorization" value="Basic
'||Authorization_Identity||'" /></httpHeader>' AS CLOB(1K)))
FROM Authorization#
```


```
{"document_tone":{"tone_categories":[{"tones":[
{"score":0.206372,"tone_id":"anger","tone_name":"Anger"},
{"score":0.364932,"tone_id":"disgust","tone_name":"Disgust"},
{"score":0.451429,"tone_id":"fear","tone_name":"Fear"},
{"score":0.097665,"tone_id":"joy","tone_name":"Joy"},
{"score":0.439061,"tone_id":"sadness","tone_name":"Sadness"}]…
```


Hmm…What JSON function should I used?

Db2 for z/OS and Waston Tone Analyzer

```
WITH Authorization AS (
 SELECT  DB2XML.BASE64ENCODE(CAST('<userid>:<password>' AS VARCHAR(64) CCSID
 1208)) AS Authorization_Identity
 FROM SYSIBM.SYSDUMMY1
),
TONETABLE AS(
  SELECT DB2XML.HTTPGETCLOB(
    CAST ('https://gateway.watsonplatform.net/tone-
analyzer/api/v3/tone?version=2016-05-19&text=' || DB2XML.URLENCODE('I thought
I can sleep in this session, but that crazy speaker keep asking us questions.
Now she is asking us to read this long text and tell her what tone it is. How
do I know? Ask a machine to do this','') AS VARCHAR(255)),
    CAST('<httpHeader><header name="Authorization" value="Basic
'||Authorization_Identity||'" /></httpHeader>' AS CLOB(1K))) TONERESULT
  FROM Authorization)
select
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'score', 'f') as score,
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'tone_name', 's:20') as tone
from TONETABLE,
TABLE(SYSTOOLS.JSON_TABLE(SYSTOOLS.JSON2BSON(TONERESULT),
'document_tone.tone_categories.tones', 's:200')) X
order by score DESC
```

Db2 for z/OS and Waston Tone Analyzer

```
WITH Authorization AS (
 SELECT  DB2XML.BASE64ENCODE
 1208)) AS Authorization_Ide
 FROM SYSIBM.SYSDUMMY1
),
TONETABLE AS(
  SELECT DB2XML.HTTPGETCLOB(
    CAST ('https://gateway.w
analyzer/api/v3/tone?version
I can sleep in this session,
Now she is asking us to read
do I know? Ask a machine to
    CAST('<httpHeader><heade
'||Authorization_Identity||'
  FROM Authorization)
```

SCORE	TONE
0.99	**Extraversion**
0.987	Emotional Range
0.97	Agreeableness
0.97	**Analytical**
0.699	Tentative
0.451429	**Fear**
0.439061	Sadness
0.364932	Disgust
0.206372	Anger
0.097665	Joy
0.007	Openness
0.004	Conscientiousness
0.0	Confident

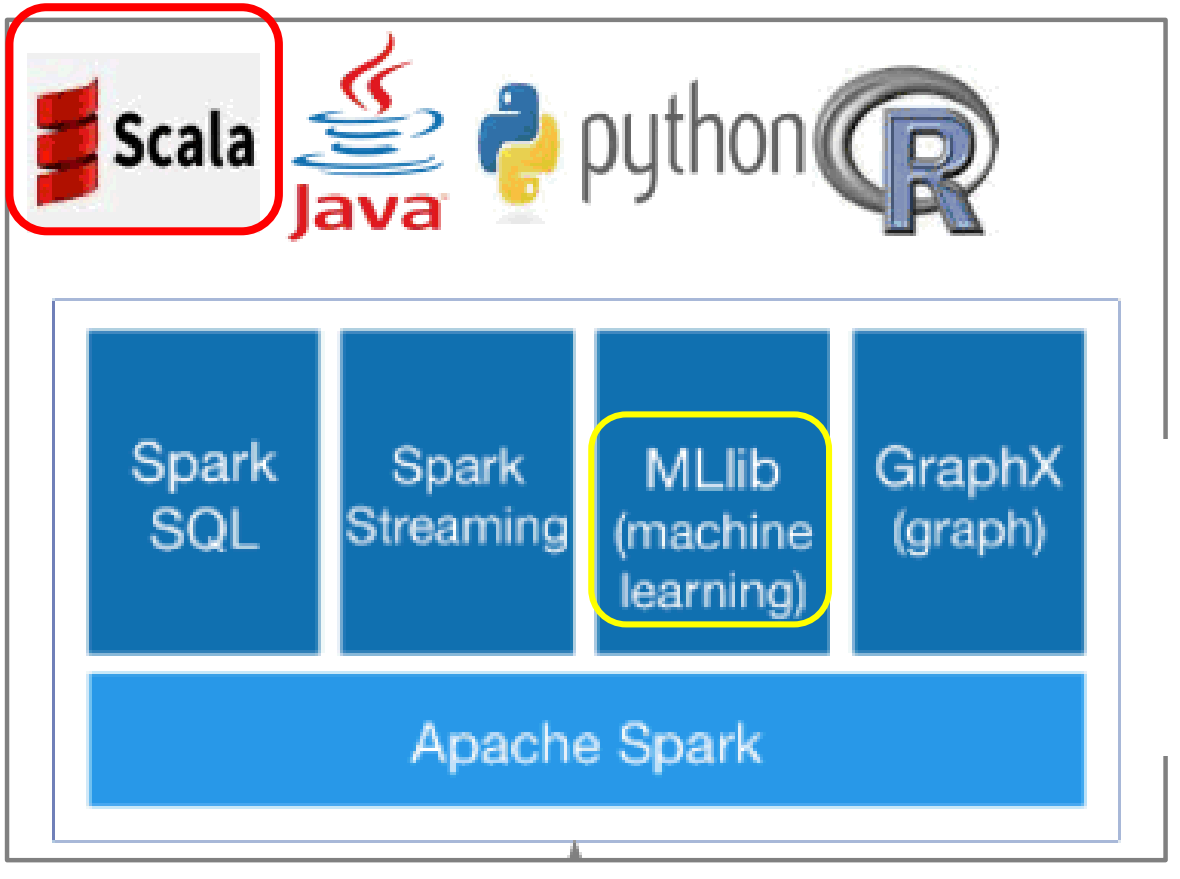
13 record(s) selected

```
select
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'score', 'f') as score,
  JSON_VAL(SYSTOOLS.JSON2BSON(X.VALUE), 'tone_name', 's:20') as tone
from TONETABLE,
TABLE(SYSTOOLS.JSON_TABLE(SYSTOOLS.JSON2BSON(TONERESULT),
'document_tone.tone_categories.tones', 's:200')) X
order by score DESC
```

Agenda

- Db2 as REST provider using Db2 native REST service
- Db2 as REST consumer using JSON feature in Db2
- Cognitive Db2 solution using Waston
- Perform Machine Learning on Db2 data without a GUI
- Spark analytics

Spark Overview



Installation/Pre-req

1. Db2 JDBC driver jar : db2jcc4.ar, db2jcc_license_cisuz.jar (the license file is only needed if the Db2 for z/OS server is not activated)
2. Java 1.7 or above
3. Spark (I use Spark 2.1.0). You can download the pre-built version from https://spark.apache.org/downloads.html

Our Scenarios

AGE	GENDER	BP_TOP	CHOLESTEROL_RATIO	SODIUM	POTASSIUM	DRUG
23	F	139.0	5.1	0.793	0.031	drugY
47	M	90.0	5.5	0.739	0.056	drugC
47	M	90.0	4.9	0.697	0.069	drugC
28	F	120.0	4.8	0.564	0.072	drugX
61	F	90.0	5.2	0.559	0.031	drugY
22	F	115.0	4.3	0.677	0.079	drugX
49	F	119.0	4.3	0.790	0.049	drugY
41	M	90.0	4.8	0.767	0.069	drugC
60	M	120.0	4.7	0.777	0.051	drugY
43	M	90.0	2.2	0.526	0.027	drugY

Input (features) — Output

All string need to be converted to double!

Steps Overview

1. Create and populate table
2. Launch Spark and connect to Db2
3. Preprocess the inputs(features) to our model
4. Find a suitable machine learning algorithms
5. Train and test the model
6. Save the model

For Step 2 to 6, we are following the steps in https://spark.apache.org/docs/2.0.2/ml-classification-regression.html#decision-tree-classifier

Step 1: Create and Populate table

```
drop table t1#
create table t1(age int, gender char(1), BP_top double,
cholesterol_ratio double, sodium double, potassium double, drug
varchar(5))#
insert into t1 values(23, 'F', 139, 5.1, 0.793, 0.031, 'drugY')#
insert into t1 values(47, 'M', 90,  5.5, 0.739, 0.056, 'drugC')#
insert into t1 values(47, 'M', 90,  4.9, 0.697, 0.069, 'drugC')#
insert into t1 values(28, 'F', 120, 4.8, 0.564, 0.072, 'drugX')#
insert into t1 values(61, 'F', 90,  5.2, 0.559, 0.031, 'drugY')#
insert into t1 values(22, 'F', 115, 4.3, 0.677, 0.079, 'drugX')#
insert into t1 values(49, 'F', 119, 4.3, 0.790, 0.049, 'drugY')#
insert into t1 values(41, 'M', 90,  4.8, 0.767, 0.069, 'drugC')#
insert into t1 values(60, 'M', 120, 4.7, 0.777, 0.051, 'drugY')#
insert into t1 values(43, 'M', 90,  2.2, 0.526, 0.027, 'drugY')#
```

Step 2: Launch Spark and connect to Db2 for z/OS

```
Set HADOOP_HOME=C:\Hadoop
set SPARKBIN=C:\Spark210\spark-2.1.0-bin-hadoop2.7\bin
SET PATH=C:\Program Files\IBM\Java70\bin;%SPARKBIN%;%PATH%
```

Run as Administrator

```
C:\Spark210\spark-2.1.0-bin-hadoop2.7\bin\spark-shell.cmd --driver-class-
path
C:\jcc_home\jccbuilds\jcc411\db2jcc4.jar;C:\jcc_home\jccbuilds\jcc411\db2jc
c_license_cisuz.jar
```

```
import org.apache.spark.ml.classification.DecisionTreeClassifier
import org.apache.spark.ml.feature.{IndexToString, StringIndexer,
VectorIndexer, VectorAssembler,OneHotEncoder,SQLTransformer}
import org.apache.spark.ml.{Pipeline, PipelineModel}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.classification.DecisionTreeClassificationModel
import org.apache.spark.sql.SparkSession
```

```
//Create a Spark session (Since Spark 2.0, use SparkSession instead of sqlContext)
 val spark =
 SparkSession.builder().appName("JaneMLExample").getOrCreate()
```

Step 2: Launch Spark and connect to Db2 for z/OS

//Get data from Db2 (customize according to your setting)

```
val DB2Data = spark.read.format("jdbc").options(Map("url" ->
"jdbc:db2://hostname:port/location:currentSchema=schema;user=user;passwo
rd=password;","driver" -> "com.ibm.db2.jcc.DB2Driver", "dbtable" ->
"SYSADM.T1")).load()
```

Step 3: Preprocess the inputs(features) to our model

//Convert all string to double

```
val labelIndexer = new
StringIndexer().setInputCol("DRUG").setOutputCol("drugLabel").fit(DB2Data)

val genderIndexer = new
StringIndexer().setInputCol("GENDER").setOutputCol("genderIndex")

val genderEncoder = new
OneHotEncoder().setInputCol("genderIndex").setOutputCol("genderVec")
```

Step 3: Preprocess the inputs(features) to our model – cont'd

//Create features vector as input to our model

```
val assembler = new VectorAssembler().setInputCols(Array("AGE", "genderVec",
"BP_TOP", "CHOLESTEROL_RATIO", "SODIUM", "POTASSIUM")).setOutputCol("features")
```

//Split the data into training and test sets (30% held out for testing).

```
val Array(trainingData, testData) = DB2Data.randomSplit(Array(0.7, 0.3))
```

Step 4: Find a suitable machine learning algorithms

//Train a DecisionTree model.

```
val dt = new
DecisionTreeClassifier().setLabelCol("drugLabel").setFeaturesCol("features")
```

//Convert indexed label (prediction in double) back to original labels like drugX, drugY, drugC (output in predictedLabel)

```
val labelConverter = new
IndexToString().setInputCol("prediction").setOutputCol("predictedLabel").
setLabels(labelIndexer.labels)
```

Step 5: Train and test the model

//Chain all the steps (like indexers, tree) in a Pipeline

```
val pipeline = new Pipeline().setStages(Array(labelIndexer,
genderIndexer, genderEncoder, assembler, dt, labelConverter))
```

//Train model using trainingData

```
val model = pipeline.fit(trainingData)
```

//Make predictions using testData .

```
val predictions = model.transform(testData)
```

//Display predication

```
predictions.select("DRUG",
"predictedLabel",
"drugLabel",
"features").show()
```

```
+-----+--------------+---------+--------------------+
| DRUG|predictedLabel|drugLabel|            features|
+-----+--------------+---------+--------------------+
|drugX|         drugX|      2.0|[22.0,0.0,115.0,4...|
|drugC|         drugC|      1.0|[47.0,1.0,90.0,4....|
|drugY|         drugY|      0.0|[49.0,0.0,119.0,4...|
+-----+--------------+---------+--------------------+
```

Actual

Predicted

Step 6: Save the model

//save the fitted pipeline to disk

```
model.write.overwrite().save("C:/Jane/Work/DB2/socialMedia/machineLearning/DB2Model1")
```

Spark 1.6.2 will NOT work

//load it back in during production

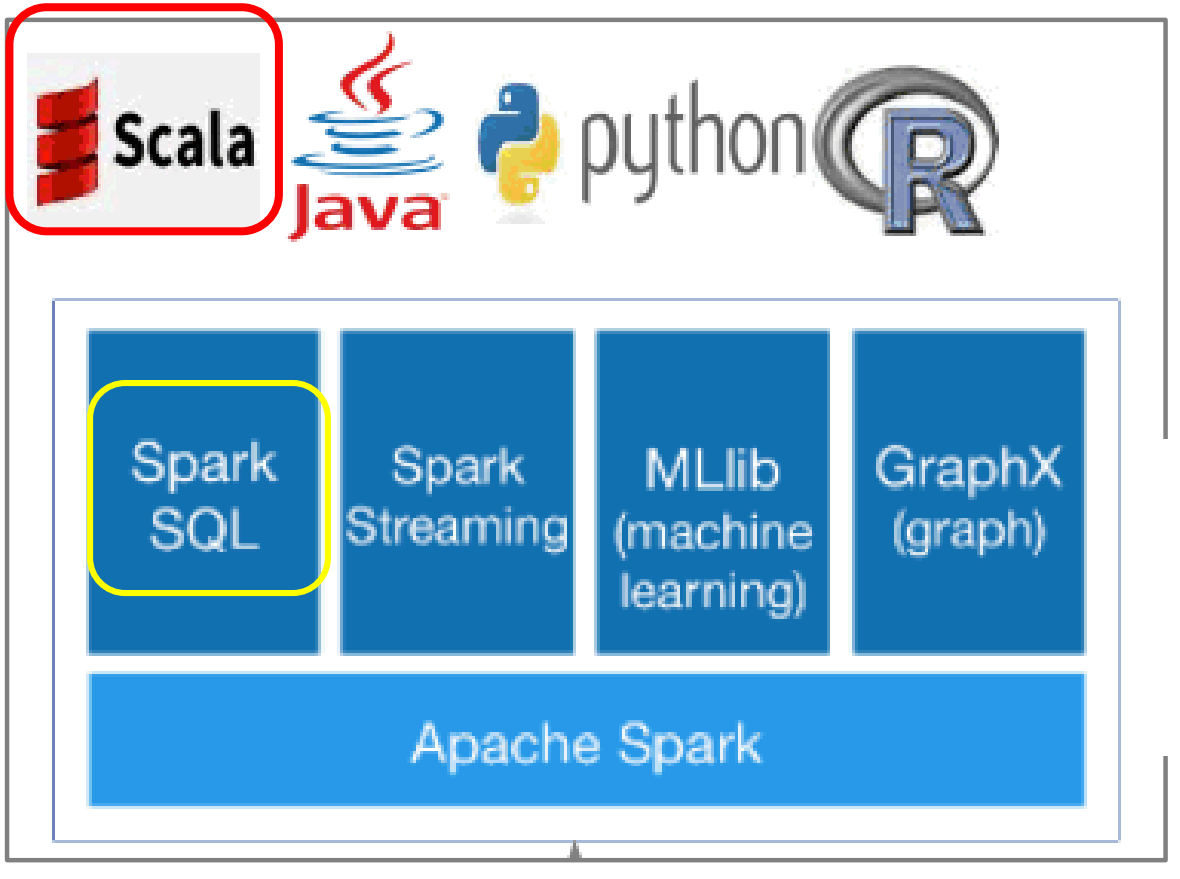
```
val sameModel = PipelineModel.load("C:/Jane/Work/DB2/socialMedia/machineLearning/DB2Model1")

val predictionAgain = sameModel.transform(testData)
```

Agenda

- Db2 as REST provider using Db2 native REST service
- Db2 as REST consumer using JSON feature in Db2
- Cognitive Db2 solution using Waston
- Perform Machine Learning on Db2 data without a GUI
- Spark analytics

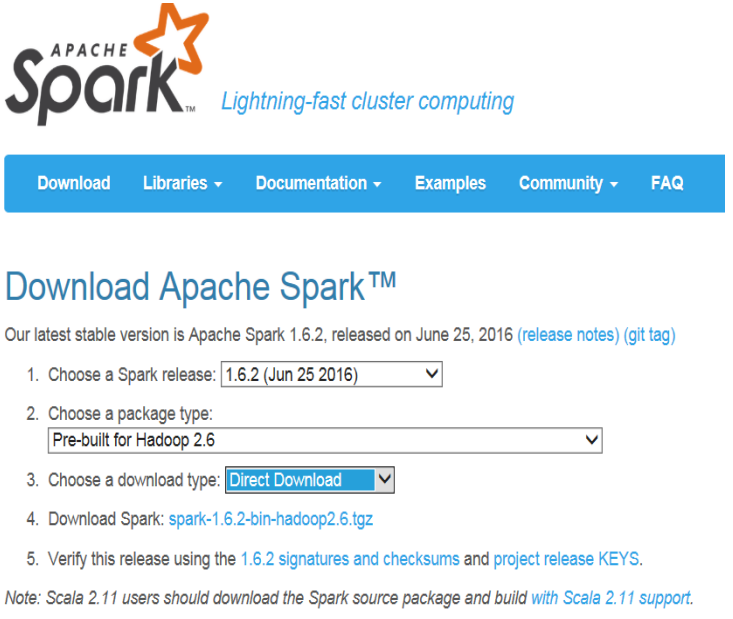
Spark Overview



Spark

Installation/Pre-req

1. JDK 1.7 or 1.8
2. Download spark* from https://spark.apache.org/downloads.html
   (I download the pre-built binaries to my Window 7 machine)
   *Apache Spark for z/OS (see resources section)
3. JCC jars



Ref: http://www.worldofdb2.com/profiles/blogs/using-spark-s-interactive-scala-shell-for-accessing-db2-data

Running Scala : Environment Setting

1. :: MUST start cmd prompt as administrator!!

I put this is a .bat file

```
Set HADOOP_HOME=C:\Hadoop
set SPARKBIN=C:\Spark162\spark-1.6.2-bin-hadoop2.6\bin
::add JCC to classpath
set JCCJAR=C:\jcc_home\jccbuilds\jcc411
set
CLASSPATH=%JCCJAR%\db2jcc4.jar;%JCCJAR%\db2jcc_license_ci
suz.jar;%JCCJAR%\db2jcc_license_cu.jar;%CLASSPATH%

::set path for jdk 7
SET PATH=C:\Program
Files\IBM\Java70\bin;%SPARKBIN%;%PATH%
```

2. Run spark-shell.cmd under bin directory

```
C:\Spark162\spark-1.6.2-bin-hadoop2.6\bin\spark-shell.cmd
```

Basic Scala

3. Create a dataframe to a table inside Db2 for z/OS

When a SQL context is created, you should see a scala prompt:

For example, the command to create a dataframe to SYSIBM. SYSXMLSTRINGS catalog table:

```
val catalogDF = sqlContext.load("jdbc", Map("url" ->
"jdbc:db2://dtec222.vmec.svl.ibm.com:446/STLEC1:currentSchema=s
ysadm;user=sysadm;password=password;","driver" ->
"com.ibm.db2.jcc.DB2Driver","dbtable" ->
"SYSIBM.SYSXMLSTRINGS"))
```

Basic Scala Command

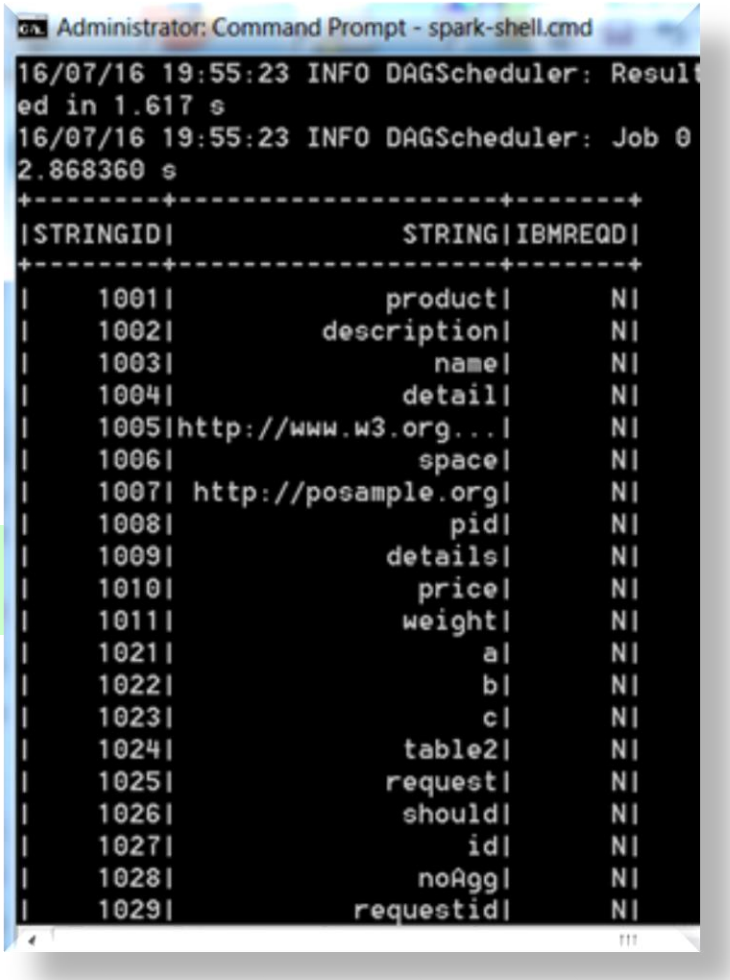
4. Enter following scala command to show the content of dataframe (SYSIBM. SYSXMLSTRINGS)

```
catalogDF.show()
```

```
catalogDF.registerTempTable("xmldf")
```

```
sqlContext.sql("select STRINGID, STRING from xmldf
WHERE STRINGID < 1005").show
```

```
+--------+-----------+
|STRINGID|     STRING|
+--------+-----------+
|    1001|    product|
|    1002|description|
|    1003|       name|
|    1004|     detail|
+--------+-----------+
```



Spark Analytics on XML

You can use either one or both

2 major packages:

1. scala.xml.XML
   (already include in the pre-built binaries, no need to download extra package)

Good for basic searching

2. com.databricks.spark.xml
   Use the following command to pull in the databricks XML package when launching the scala shell:

```
bin\spark-shell --packages com.databricks:spark-xml_2.10:0.3.3
```

Good for XML document with array structure

* this command is for scala 2.10

See my blogs in Resources section for more details

Our Scenarios

```
create table t1(id int, xmlcol xml);
```

2 XML documents are inserted in T1

```
<PO id="123" orderDate="2013-11-18">
 <customer cid="123"/>
 <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>149.99</USPrice>
      <shipDate>2013-11-20</shipDate>
    </item>
    <item partNum="945-ZG">
      <productName>Sapphire
Bracelet</productName>
      <quantity>2</quantity>
      <USPrice>178.99</USPrice>
      <comment>Not shipped</comment>
    </item>
  </items>
</PO>
```

Hubert

```
<PO id="999" orderDate="2013-11-18">
 <customer cid="999"/>
 <items>
    <item partNum="900-AA">
      <productName>pen</productName>
      <quantity>1</quantity>
      <USPrice>5.0</USPrice>
      <shipDate>2013-11-20</shipDate>
    </item>
    <item partNum="145-ZG">
      <productName>Johnson lotion</productName>
      <quantity>2</quantity>
      <USPrice>10.0</USPrice>
      <comment>Not shipped</comment>
    </item>
      <item partNum="945-ZG">
      <productName>Sapphire Bracelet</productName>
      <quantity>3</quantity>
      <USPrice>178.99</USPrice>
      <comment>Not shipped</comment>
    </item>
  </items>
</PO>
```

Jorn

scala.xml.XML

//1. import the required package

```
scala> import scala.xml.XML
```

//2. create a dataframe for our T1 table

```
scala> val xmlDF = sqlContext.load("jdbc", Map("url" ->
"jdbc:db2://dtec222.vmec.svl.ibm.com:446/STLEC1:currentSchema=sysad
m;user=sysadm;password=password;","driver" ->
"com.ibm.db2.jcc.DB2Driver", "dbtable" -> "SYSADM.T1"))
```

//3. create a string array to hold the XML document

```
scala> var xmlStringArray = (xmlDF.select(xmlDF("XMLCOL"))).map(
_.getString( 0 ) ).map( scala.xml.XML.loadString( _ ) ).collect()
```

```
xmlStringArray: Array[scala.xml.Elem] = Array(<PO
orderDate="2013-11-18" id="123"><customer cid="123"/><items>
<item partNum="872-AA"><productName>Lawnmower
</productName><quantity>1</quantity><USPrice>149.99</USPrice><sh
ipDate>2013-11-20</shipDate>…….
```

<PO ..id="123">…

<PO ..id="999">…

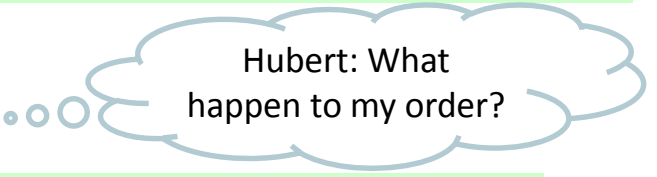
scala.xml.XML

```
<PO id="123" orderDate="2013-11-18"
  <customer cid="123"/>
  <items>
     <item partNum="872-AA">
       <productName>Lawnmower</pro
....
```

```
<PO id="999" orderDate="2013-11-18">
  <customer cid="999"/>
  <items>
     <item partNum="900-AA">
       <productName>pen</productName>
....
```

//find all cid

```
scala> var cid = xmlStringArray.map( _ \\ "PO" \ "customer" \ "@cid" )
cid: Array[scala.xml.NodeSeq] = Array(123, 999)
```



//create a function to check for a particular cid

```
scala> def cidEquals(value: String)(node: scala.xml.Node) = {
     ( node \\ "PO" \ "customer" \ "@cid" ).exists(_.text == value)
}
```

//find all productName for cid = 123

```
scala> xmlStringArray.map( _ \\ "_" filter cidEquals("123") ).map(
_ \\ "PO" \ "items" \ "item" \ "productName" ).foreach(println(_))
<productName>Lawnmower</productName><productName>Sapphire
Bracelet</productName>
```

scala.xml.XML

Hubert: How much I need to pay?

//create a function to find total price

```
scala> def totalPrice(node: scala.xml.Node) :
    Double = {
        var USPrice :Double = 0
        var quantity :Int = 0
        var totalPrice :Double = 0
        var items  = (node \\ "PO" \ "items")
        for (item <- items \\ "item")
        {
           USPrice = (item \\ "USPrice").text.toDouble
           quantity = (item \\ "quantity").text.toInt
           totalPrice = totalPrice + (USPrice * quantity)
        }

        return totalPrice
}
```

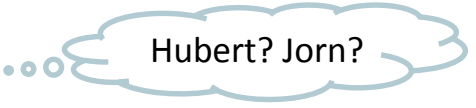
```
<PO id="123" orderDate="2013-11-18">
  <customer cid="123"/>
  <items>
    <item partNum="872-AA">
<productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>149.99</USPrice>
      <shipDate>2013-11-20</shipDate>
    </item>
    <item partNum="945-ZG">
      <productName>Sapphire
Bracelet</productName>
      <quantity>2</quantity>
      <USPrice>178.99</USPrice>
      <comment>Not shipped</comment>
    </item>
  </items>
</PO>
```

scala.xml.XML

```
<PO id="999" orderDate="2013-11-18">
  <customer cid="999"/>
  <items>….
```

//find total price for each XML document

```
scala> xmlStringArray.map(totalPrice(_)).foreach(println(_))
507.97
561.97
```



//associate cid with total price

```
scala> def cidNTotalPriceSimple(node: scala.xml.Node) = {
  ((node \\ "PO" \ "customer" \ "@cid").text.toInt,totalPrice(node))
}
```

//create a dataframe for cid and corresponding total price

```
scala> val df =
sc.parallelize(xmlStringArray.map(cidNTotalPriceSimple(_))).toDF("c
id", "totalPrice")
```

```
scala> df.printSchema()
root
 |-- cid: integer (nullable = false)
 |-- totalPrice: double (nullable = false)
```

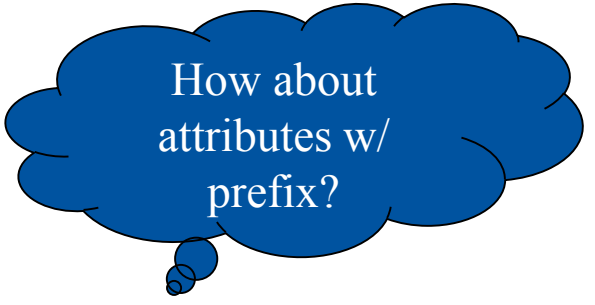
cid	totalPrice
123	507.97
999	561.97

scala.xml.XML

//find all cid and total price and sort total price ascendingly

```
scala> df.select("cid", "totalPrice").sort(asc("totalPrice")).show
```

```
+---+----------+
|cid|totalPrice|
+---+----------+
|123|    507.97|
|999|    561.97|
+---+----------+
```



//find cid with total price > 550

```
df.filter($"totalPrice" > 550).show
```

```
+---+----------+
|cid|totalPrice|
+---+----------+
|999|    561.97|
+---+----------+
```

```
scala> val xmldoc = <a xmlns:bar="http://bar.com" bar:b="bAttributes"></a>
```

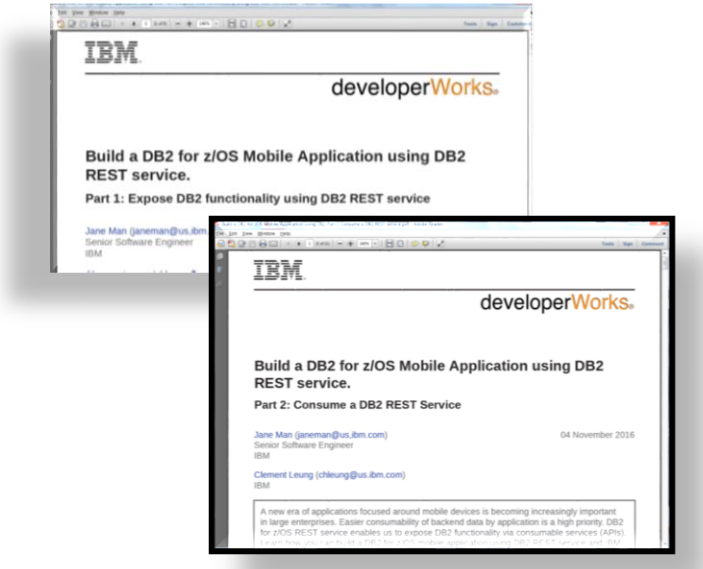
```
scala> xmldoc \ "@{http://bar.com}b"
```

```
res0: scala.xml.NodeSeq = bAttributes
```

Summary

- **Db2 + Watson:** Congitive Solution on your enterprise data
- Use Db2 for a z/OS as REST consumer to utilize data outside Db2
- Db2 for z/OS is a **REST service provider** now!
- Apache **Spark** provides big data analytics capabilities on Db2 data.
- Write your own **Machine Learning** application using Db2 for z/OS data

Resources



- Db2 for z/OS Native DDF REST services - 2 whitepaper by Jane Man
  https://www.ibm.com/developerworks/community/blogs/e429a8a2-b27f-48f3-aa73-ca13d5b69759/entry/DB2_for_z_OS_Native_DDF_REST_services?lang=en
- Db2 REST services
  http://www.ibm.com/support/knowledgecenter/en/SSEPEK_11.0.0/restserv/src/tpc/db2z_restservices.html
- IBM MobileFirst 7 Developer Edition
  https://developer.ibm.com/mobilefirstplatform/install/#studio

Resources – Spark Analytics for XML

- IBM Packages for Apache Spark
  https://www.ibm.com/developerworks/java/jdk/spark/
- Installing IBM z/OS Platform for Apache Spark
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102609
- Spark Analytics for XML data in DB2 for z/OS – Part 1 : scala.xml.XML by Jane Man
  http://ibm.co/2frZw2y
- Spark Analytics for XML data in DB2 for z/OS – Part 2 : databricks XML package – on a single XML document by Jane Man http://ibm.co/2frZE22
- Spark Analytics for XML data in DB2 for z/OS – Part 3 : databricks XML package – on XML data in database by Jane Man http://ibm.co/2frZF6a
- Spark Blog 1 - Using Spark's interactive Scala shell for accessing DB2 data using JDBC driver and Spark's new DataFrames API
  http://www.worldofdb2.com/profiles/blogs/using-spark-s-interactive-scala-shell-for-accessing-db2-data
- Instantiating a case class from a list of parameters http://stackoverflow.com/questions/4290955/instantiating-a-case-class-from-a-list-of-parameters

Resources – for Machine Learning and Python

- Machine Learning using DB2 for z/OS data and Spark Part 1 by Jane Man
  https://www.ibm.com/developerworks/community/blogs/e429a8a2-b27f-48f3-aa73-ca13d5b69759/entry/Machine_Learning_using_DB2_for_z_OS_data_and_Spark_Part_1?lang=en

- Machine Learning using DB2 for z/OS data and Spark Part 2 by Jane Man
  https://www.ibm.com/developerworks/community/blogs/e429a8a2-b27f-48f3-aa73-ca13d5b69759/entry/Machine_Learning_using_DB2_for_z_OS_data_and_Spark_Part_2?lang=en

- Quick Start in accessing DB2 for z/OS from Python applications by Jane Man
  https://www.ibm.com/developerworks/community/blogs/e429a8a2-b27f-48f3-aa73-ca13d5b69759/entry/Quick_Start_in_accessing_DB2_for_z_OS_from_Python_applications?lang=en


Questions?

Jane Man
IBM
janeman@us.ibm.com

Db2 for z/OS Modern Application: Watson, ML, REST, Spark and more

Session code: G12