

## Contents

Contents	1
Introduction to MainSpace	2
MainSpace Getting Started Guide	3
Example #1 – Display a Comment	3
Example #2 – List the Contents	5
Example #3 – Return Responses	8
Example #4 – Simple Table Display	11
Example #5 – Simple Formatted Table Display	13
Example #6 – Table Display with 3 <sup>rd</sup> Party Library	15
Where to Next?	18
MainSpace SAMPLEs	19
DB2 View Sample App	19

## Introduction to MainSpace

MainSpace is Mainframe Cloud's agile DevOps solution for IBM's System Z.

MainSpace allows web app developers to build applications for the mainframe in web languages such as JavaScript and HTML5. Users have the flexibility to utilise third party charting libraries to create whatever visuals are required.

This release of MainSpace supports a DB2 Interface.

The MainSpace Getting Started section will guide you through creating simple Apps.

## MainSpace Getting Started Guide

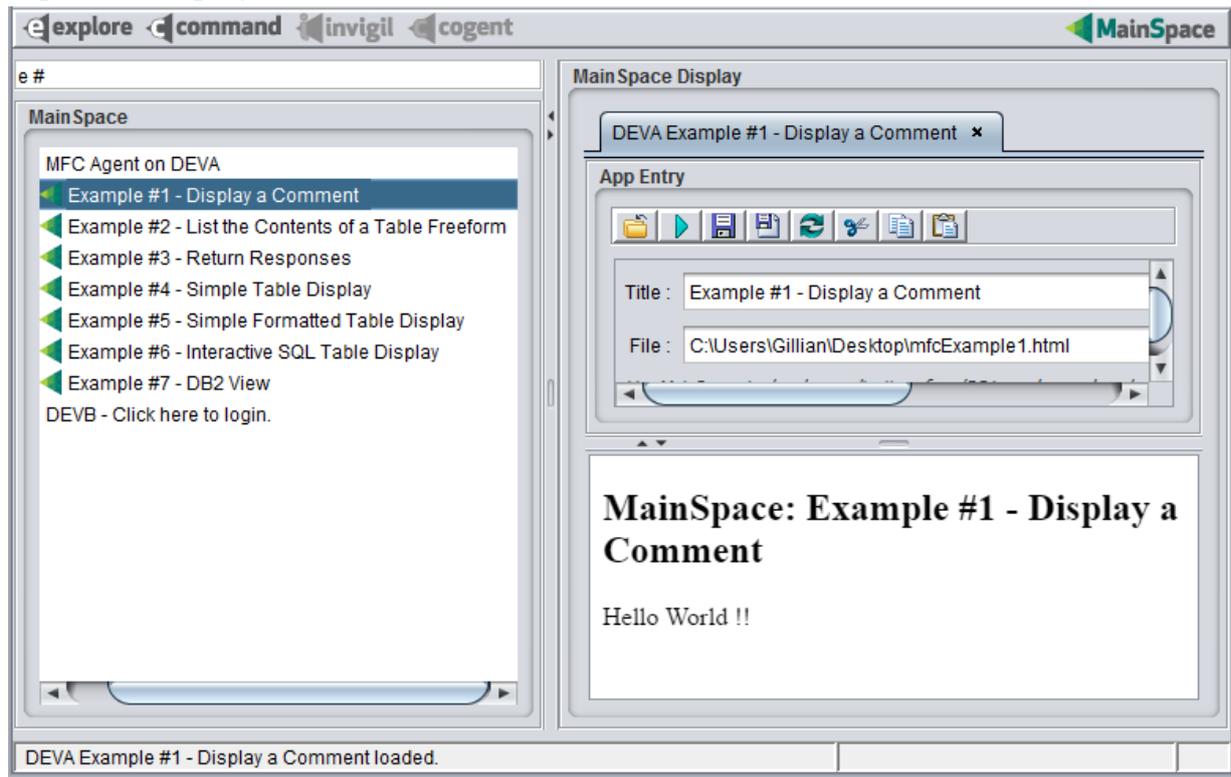
The examples below guide you step by step through the process of writing MainSpace Apps. The Examples are located on the Mainframe Cloud website at the links provided in each example below.

### Example #1 – Display a Comment

We begin with a simple MainSpace App.

1. Open this **example html File** [here](#) and copy the contents.
2. In Notepad create a new file and paste the contents of the above example.
3. Save the file with **.html** extension. (The example is saved to the Desktop as “mfcExample1.html” )
4. Login to Mainframe Cloud.
5. Click **MainSpace** product.
6. Create a NEW App (CTRL+N)
7. Specify a Title: Example #1 - Display a Comment
8. Specify a File Location: right click on the field & drill down to where you saved the “mfcExample1.html” file on your desktop.
9. Click Save Icon (CTRL+S)
10. Click Execute Icon (CTRL+F5)
11. You should see a similar display to below in [Expected Display](#).

## Expected Display after the Execute action



## Example #2 – List the Contents

This example MainSpace App executes a simple SQL statement and displays the return response in a raw freeform dump. Before you start you must know:

### DB2 SSID

The up to 4-character z/OS DB2 SSID you wish to interrogate. Specified in this statement: `obj1.setDB2SSID( "DBBG" );`

### DB2 Table

The z/OS DB2 Table you wish to interrogate. Specified in this statement: `obj1.runSQL( "SELECT * FROM DSN81110.EMP" );`

This example will list the contents of a DB2 table to show the field names and the data. Ensure its not too large a table.

1. Open this **example html File** [here](#) and copy the contents.
2. In Notepad create a new file and paste the contents of the above example.
3. Change the DB2 SSID and the DB2 Table reference to your own values.
4. Save the file with **.html** extension. (The example is saved to the Desktop as "mfcExample2.html" )
5. Login to Mainframe Cloud.
6. Click **MainSpace** product.
7. Create a NEW App (CTRL+N)
8. Specify a Title: Example #2 – List the Contents of a Table Freeform
9. Specify a File Location: right click on the field & drill down to where you saved the "mfcExample2.html" file on your desktop.
10. Click Save Icon (CTRL+S)
11. Click Execute Icon (CTRL+F5)
12. You should see a similar display to below in [Expected Display](#).

## mfcExample2.html – API Call Explained

Referring to the mfcExample2.html sample code let's examine the API Call.

**MFCAgentRequest** is the class used to perform MainSpace API calls:

```
MFCAgentRequest.newObject( "obj1" );  
obj1.callbackFunction = "callBack( obj1.response );";  
obj1.setDB2SSID( "DBBG" );  
obj1.runSQL( "SELECT * FROM DSN81110.EMP" );
```

First, create a new object instance to work with:

```
MFCAgentRequest.newObject( "obj1" );
```

Second, nominate the user function that is to be executed when the API call is complete.

```
obj1.callbackFunction = "callBack( obj1.response );";
```

Third, set the API call specific required input fields. We are executing a DB2 SQL query, for this you need to specify the DB2 SSID and the SQL string. The SSID (subsystem identifier) identifies which DB2 region you wish to query.

```
obj1.setDB2SSID( "DBBG" );  
obj1.runSQL( "SELECT * FROM DSN81110.EMP" );
```

The **MFCAgentRequest** API call will return response data in the 'response' object in JSON format.

The callback function definition specifies a function name "callBack", and the input object to feed to that function.

In this example, the API call returns the "obj1.response" JSON object which is passed to the callBack function to process/display the result of the API call.

## mfcExample2.html – callbackFunction Explained

Referring to the above mfcExample2.html sample code we examine the callbackFunction statement. When the MainSpace API call is complete it will call the specified JavaScript function. The JavaScript function name and input parm(s) were specified in the **MFCAgentRequest callbackFunction** field:

```
obj1.callbackFunction = "callBack( obj1.response );";
```

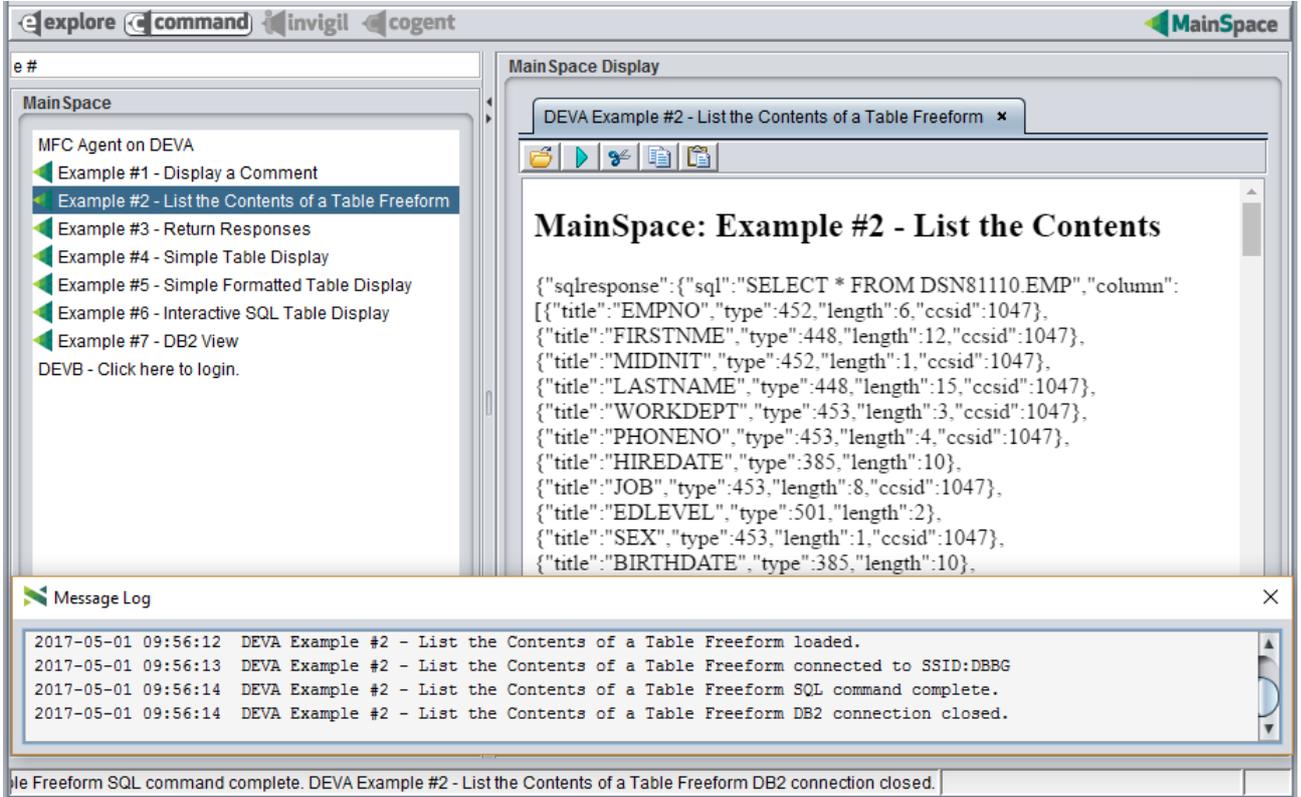
This is the JavaScript function:

```
function callBack( response ) {  
    var myJSON = JSON.stringify( response );  
    document.getElementById("sample").innerHTML = myJSON;  
}
```

This is a very basic function to begin with. It defines local variable 'response', which will be populated by the API call return data. The local variable 'response' can be any valid variable name.

This example converts the JSON object into a raw string, and then populates the web page content with this data. This example app will execute the API call as soon as the App is loaded.

### Expected Display after the Execute action



The screenshot displays the MainSpace application interface. The main window is titled "MainSpace Display" and shows a tab for "DEVA Example #2 - List the Contents of a Table Freeform". The content area displays the following JSON response:

```
{"sqlresponse":{"sql":"SELECT * FROM DSN81110.EMP","column": [{"title":"EMPNO","type":452,"length":6,"ccsid":1047}, {"title":"FIRSTNME","type":448,"length":12,"ccsid":1047}, {"title":"MIDINIT","type":452,"length":1,"ccsid":1047}, {"title":"LASTNAME","type":448,"length":15,"ccsid":1047}, {"title":"WORKDEPT","type":453,"length":3,"ccsid":1047}, {"title":"PHONENO","type":453,"length":4,"ccsid":1047}, {"title":"HIREDATE","type":385,"length":10}, {"title":"JOB","type":453,"length":8,"ccsid":1047}, {"title":"EDLEVEL","type":501,"length":2}, {"title":"SEX","type":453,"length":1,"ccsid":1047}, {"title":"BIRTHDATE","type":385,"length":10},
```

Below the main display is a "Message Log" window showing the following entries:

```
2017-05-01 09:56:12 DEVA Example #2 - List the Contents of a Table Freeform loaded.
2017-05-01 09:56:13 DEVA Example #2 - List the Contents of a Table Freeform connected to SSID:DBBG
2017-05-01 09:56:14 DEVA Example #2 - List the Contents of a Table Freeform SQL command complete.
2017-05-01 09:56:14 DEVA Example #2 - List the Contents of a Table Freeform DB2 connection closed.
```

At the bottom of the interface, a status bar displays the message: "File Freeform SQL command complete. DEVA Example #2 - List the Contents of a Table Freeform DB2 connection closed."

Note: The Message Log displays the actions executed.

## Example #3 – Return Responses

This example MainSpace App executes a simple SQL statement and displays a response. Before you start you must know:

### DB2 SSID

The up to 4-character z/OS DB2 SSID you wish to interrogate. Specified in this statement: `obj1.setDB2SSID( "DBBG" );`

### DB2 Table

The z/OS DB2 Table you wish to interrogate. Specified in this statement: `obj1.runSQL( "SELECT * FROM DSN81110.EMP" );`

This example targets some of the fields returned in the response object to provide a more meaningful display. This example allows you to experiment with changing values.

1. Open this **example html File** [here](#) and copy the contents.
2. In Notepad create a new file and paste the contents of the above example.
3. Change the DB2 SSID and the DB2 Table reference to your own values.
4. Save the file with **.html** extension. (The example is saved to the Desktop as "mfcExample3.html" )
5. Login to Mainframe Cloud.
6. Click **MainSpace** product.
7. Create a NEW App (CTRL+N)
8. Specify a Title: Example #3 – Return Responses
9. Specify a File Location: right click on the field & drill down to where you saved the "mfcExample3.html" file on your desktop.
10. Click Save Icon (CTRL+S)
11. Click Execute Icon (CTRL+F5)
12. You should see a similar display below in [Expected Display](#).
13. To observe another message, in your mfcExample3.html, change the value of the SSID to an invalid value. In this example it was changed to "DBBGz".
14. Save your mfcExample3.html and re-execute the App in MainSpace.
15. You should see a similar display below in the second screen shot.

## mfcExample3.html – callbackFunction Explained

Referring to the above mfcExample3.html sample code let's examine the callback function. The only change from Example 2 to Example 3 is a more complex callback function to display information about the SQL query issued.

First, construct the message variable if the API call returns a message:

```
// Construct message.
var sqlmsg = '';
if ( typeof response.sqlresponse.message != "undefined" ) {
  for ( var i = 0 ; i < response.sqlresponse.message.length ; i++ ) {
    sqlmsg = sqlmsg + response.sqlresponse.message[i] + '<BR>';
  }
}
```

Second, set the row total:

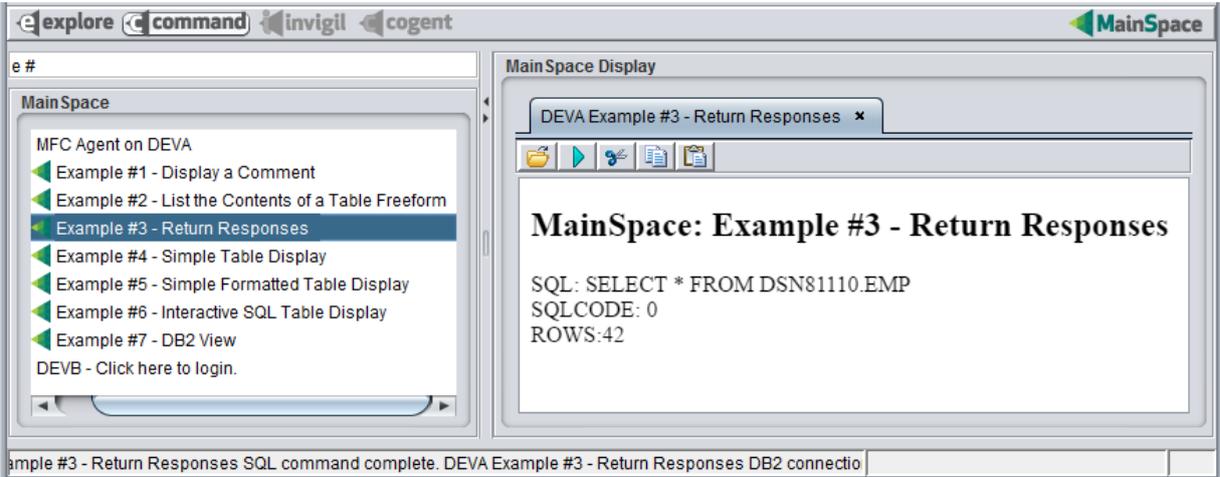
```
// Get row total
var rows = 0;
if ( typeof response.sqlresponse.row != "undefined" ) {
  rows = response.sqlresponse.row.length;
}
```

Third, display the results:

```
// Display results.
document.getElementById("sample").innerHTML = 'SQL: ' + response.sqlresponse.sql +
'<BR>SQLCODE: ' + response.sqlresponse.sqlcode +
( ( sqlmsg == '' )
? ''
: '<BR><BR>MESSAGE<BR>' + sqlmsg
) +
'<BR>ROWS:' + rows;
```

## Expected Display after the Execute action

Expected display when all values are correct.

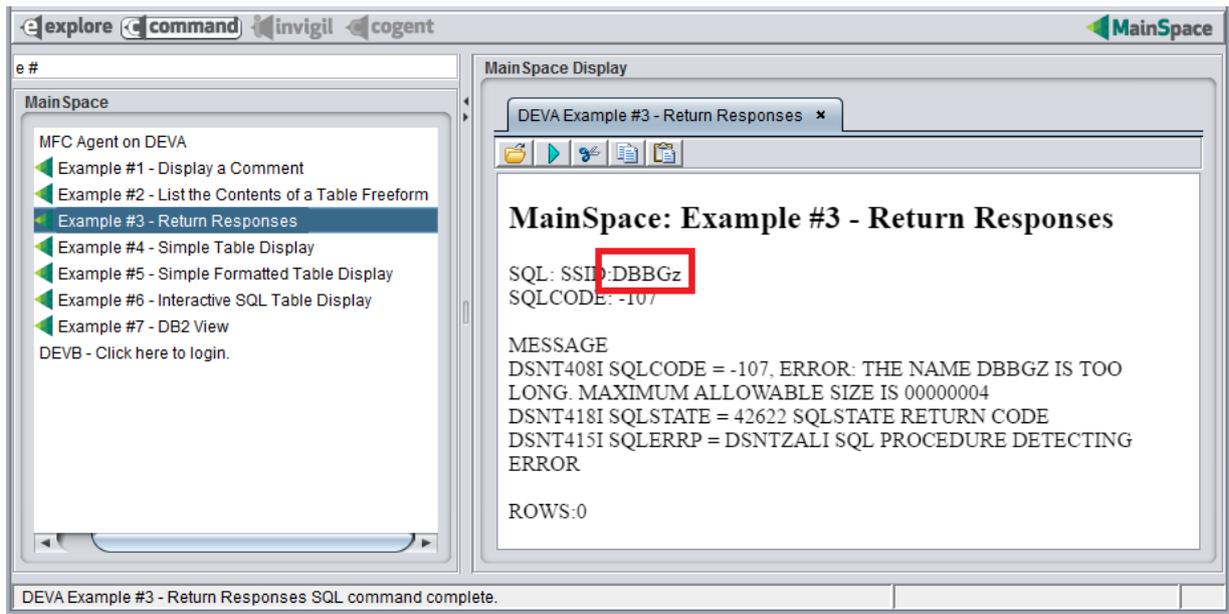


The screenshot shows the MainSpace application interface. On the left, a navigation pane lists several examples, with "Example #3 - Return Responses" selected. The main display area shows the results of an SQL query:

```
SQL: SELECT * FROM DSN81110.EMP
SQLCODE: 0
ROWS:42
```

The status bar at the bottom indicates: "Example #3 - Return Responses SQL command complete. DEVA Example #3 - Return Responses DB2 connectio".

Expected display when the SSID value is incorrect.



## Example #4 – Simple Table Display

This example MainSpace App executes a simple SQL statement and displays the output in a table. Before you start you must know:

### DB2 SSID

The up to 4-character z/OS DB2 SSID you wish to interrogate. Specified in this statement: `obj1.setDB2SSID( "DBBG" );`

### DB2 Table

The z/OS DB2 Table you wish to interrogate. Specified in this statement: `obj1.runSQL( "SELECT * FROM DSN81110.EMP" );`

This example displays the SQL query result in a Table:

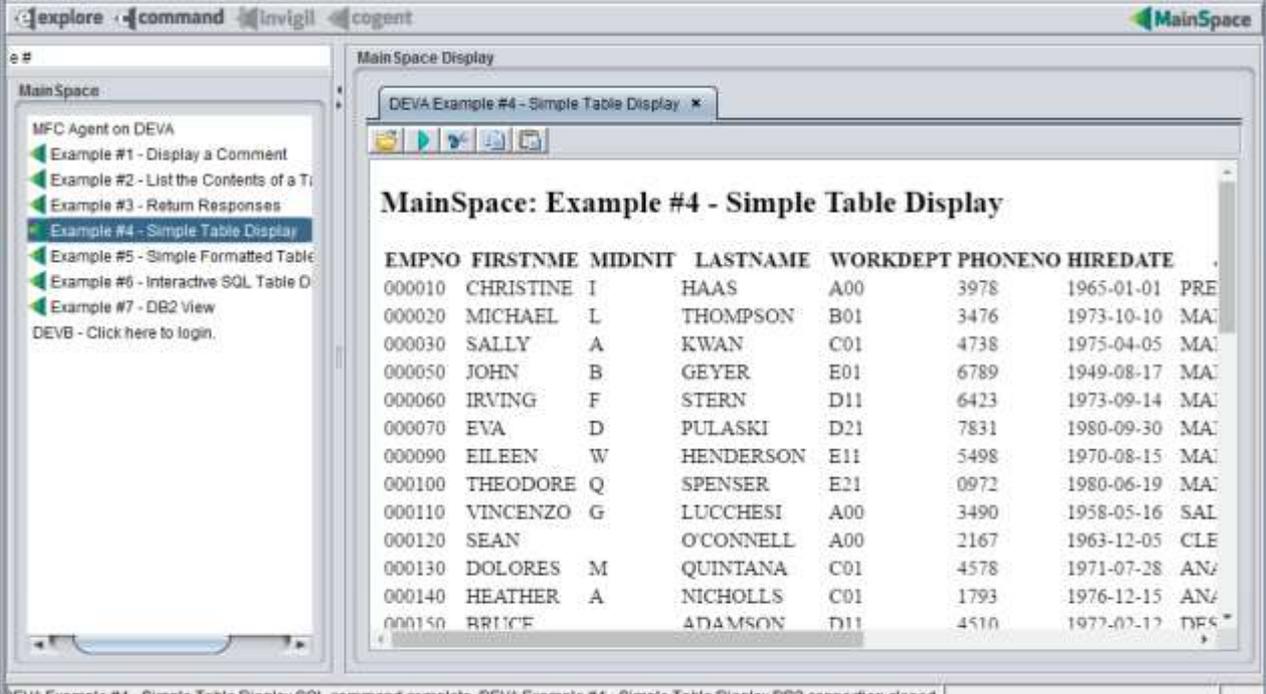
1. Open this **example html File** [here](#) and copy the contents.
2. In Notepad create a new file and paste the contents of the above example.
3. Change the DB2 SSID and the DB2 Table reference to your own values.
4. Save the file with **.html** extension. (The example below is saved to the Desktop as "mfcExample4.html" )
5. Login to Mainframe Cloud.
6. Click **MainSpace** product.
7. Create a NEW App (CTRL+N)
8. Specify a Title: Example #4 – Simple Table Display
9. Specify a File Location: right click on the field & drill down to where you saved the "mfcExample4.html" file on your desktop.
10. Click Save Icon (CTRL+S)
11. Click Execute Icon (CTRL+F5)
12. You should see a similar display below in [Expected Display](#).

## mfcExample4.html – callbackFunction Explained

Referring to the mfcExample4.html sample code the only change from Example 3 to Example 4 is an enhanced callback function with some basic JavaScript. The JavaScript function will display an html table populated with the API response data.

```
function callBack( response ) {
  var mfcTable = '<table><tr>';
  for ( headI = 0; headI < response.sqlresponse.column.length; headI++ ) {
    mfcTable += '<th>' + response.sqlresponse.column[headI].title + '</th>';
  }
  mfcTable += '</tr>';
  for ( rowI = 0; rowI < response.sqlresponse.row.length; rowI++ ) {
    mfcTable += '<tr>';
    for ( headI = 0; headI < response.sqlresponse.column.length; headI++ ) {
      mfcTable += '<td>' +
        (response.sqlresponse.row[rowI])[response.sqlresponse.column[headI].title] +
        '</td>';
    }
    mfcTable += '</tr>';
  }
  mfcTable += '</table>';
  document.getElementById("sample").innerHTML = mfcTable;
}
```

## Expected Display after the Execute action



The screenshot shows the MainSpace interface with a table titled "MainSpace: Example #4 - Simple Table Display". The table contains the following data:

EMPNO	FIRSTNAME	MIDDLEINITIAL	LASTNAME	WORKDEPT	PHONENO	HIREDATE	DEPARTMENT
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRE
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MA
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MA
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MA
000060	IRVING	F	STERN	D11	6423	1973-09-14	MA
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MA
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MA
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MA
000110	VINCENZO	G	LUCCHESI	A00	3490	1958-05-16	SAL
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLE
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	AN
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	AN
000150	BRYCE		ADAMSON	D11	4510	1977-07-17	DEP

The status bar at the bottom of the window displays: "DEVA Example #4 - Simple Table Display SQL command complete. DEVA Example #4 - Simple Table Display DB2 connection closed."

## Example #5 – Simple Formatted Table Display

This example MainSpace App executes a simple SQL statement and displays the output in a formatted table. This example introduces some basic CSS styling into the html head section of our web page to make our table more readable. Before you start you must know:

### DB2 SSID

The up to 4-character z/OS DB2 SSID you wish to interrogate. Specified in this statement: `obj1.setDB2SSID( "DBBG" );`

### DB2 Table

The z/OS DB2 Table you wish to interrogate. Specified in this statement: `obj1.runSQL( "SELECT * FROM DSN81110.EMP" );`

This example displays the SQL query result in a formatted Table:

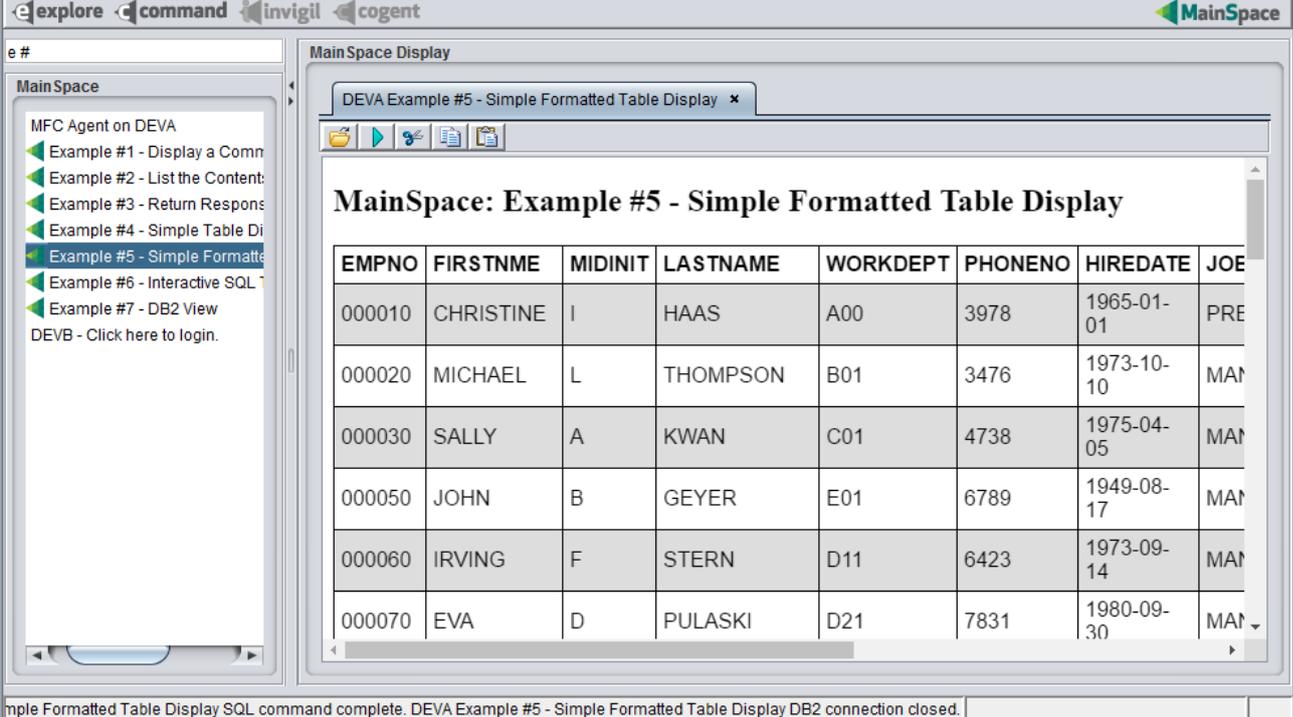
1. Open this **example html File** [here](#) and copy the contents.
2. In Notepad create a new file and paste the contents of the above example.
3. Change the DB2 SSID and the DB2 Table reference to your own values.
4. Save the file with **.html** extension. (The example below is saved to the Desktop as "mfcExample5.html" )
5. Login to Mainframe Cloud.
6. Click **MainSpace** product.
7. Create a NEW App (CTRL+N)
8. Specify a Title: Example #5 – Simple Formatted Table Display
9. Specify a File Location: right click on the field & drill down to where you saved the "mfcExample5.html" file on your desktop.
10. Click Save Icon (CTRL+S)
11. Click Execute Icon (CTRL+F5)
12. You should see a similar display below in [Expected Display](#).

## mfcExample5.html - <head> Section Explained

Referring to the mfcExample5.html sample code the only change from Example 4 to Example 5 is the inclusion of some formatting code in the head section of the MainSpace App web file. CSS Styling code has been used as follows:

```
<head>
  <!-- ----->
  <!-- CSS Style Section ----->
  <!-- ----->
  <style>
    table {
      font-family: arial, sans-serif;
      border-collapse: collapse;
      width: 100%;
    }
    td, th {
      border: 1px solid #000000;
      text-align: left;
      padding: 5px;
    }
    tr:nth-child(even) {
      background-color: #dddddd;
    }
  </style>
</head>
```

## Expected Display after the Execute action



The screenshot shows the MainSpace application interface. On the left, a navigation pane lists several examples, with 'Example #5 - Simple Formatted Table Display' selected. The main display area shows a table titled 'MainSpace: Example #5 - Simple Formatted Table Display'. The table has 8 columns: EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE, and JOE. The data is as follows:

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOE
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRE
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MAN
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MAN
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MAN
000060	IRVING	F	STERN	D11	6423	1973-09-14	MAN
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MAN

At the bottom of the window, a status bar indicates: 'Simple Formatted Table Display SQL command complete. DEVA Example #5 - Simple Formatted Table Display DB2 connection closed.'

## Example #6 – Table Display with 3<sup>rd</sup> Party Library

This example MainSpace App uses a third-party graphics library to display the output. We therefore do not need the CSS Styling code or the JavaScript table code that was used in Example 5. Before you start you must know:

### DB2 SSID

The up to 4-character z/OS DB2 SSID you wish to interrogate. Specified in this statement: "DB2SSID":"*DBBG*",

### DB2 Table

The z/OS DB2 Table you wish to interrogate. Specified in this statement:

```
"SQL": [ "SELECT * FROM DSN81110.EMP" ]
```

This example displays the SQL query result in a formatted 3<sup>rd</sup> party Table:

1. Open this **example html File** [here](#) and copy the contents.
2. In Notepad create a new file and paste the contents of the above example.
3. Change the DB2 SSID and the DB2 Table reference to your own values.
4. Save the file with **.html** extension. (The example below is saved to the Desktop as "mfcExample6.html" )
5. Login to Mainframe Cloud.
6. Click **MainSpace** product.
7. Create a NEW App (CTRL+N)
8. Specify a Title: Example #6 – Third Party GUI Table Display
9. Specify a File Location: right click on the field & drill down to where you saved the "mfcExample6.html" file on your desktop.
10. Click Save Icon (CTRL+S)
11. Click Execute Icon (CTRL+F5)
12. You should see a similar display below in [Expected Display](#).

## mfcExample6.html – <head> Section Explained

Referring to the mfcExample6.html sample code the head section now references the 3<sup>rd</sup> party graphics libraries as follows:

```
<head>
<!-- ----->
<!-- IMPORT THIRD PARTY GUI LIBRARIES. -->
<!-- ----->
<link href="http://www.mainframecloudplatform.com/applibs/fgrid/fancy.min.css" rel="stylesheet">
<script type="text/javascript" src="http://www.mainframecloudplatform.com/applibs/fgrid/fancy.full.min.js"></script>
</head>
```

## mfcExample6.html – callbackFunction Explained

Referring to the mfcExample6.html sample code the callback function utilises the 3<sup>rd</sup> party libraries as follows.

First, the column data object returned from the API call must be massaged:

```
// Create column headings object with extracts from response data.
var headings = [];
for ( i = 0 ; i < response.sqlresponse.column.length ; i++ ) {
    headings.push( { index: response.sqlresponse.column[i].title,
                    title: response.sqlresponse.column[i].title } );
}
```

Second, we code a call to display the data in a table:

- Pass the new heading object created above - **headings**
- Pass the table data object (**response.sqlresponse.row**) from the API call.

```
/*
 * Call third party GUI library to display table with the JSON data.
 */
new FancyGrid({
    renderTo: 'container',
    width: 'fit',
    height: 400,
    data: response.sqlresponse.row,
    columns: headings
});
```

Notes:

- The table **data** returned from the API call does not require changing as the 3<sup>rd</sup> party solution accepts the response row data object as is.
- Many 3<sup>rd</sup> party libraries require different approaches to utilize them, but the majority accept data in JSON format. In this example we did not need to change the row data object.
- For more information on how to use the 3<sup>rd</sup> party library in this example refer to <https://fancygrid.com/> and click the documentation link.

## mfcExample6.html – API Call Explained

Referring to the mfcExample6.html sample code let's examine the JSON input to the API Call. A noticeable difference in this example is the format of the input data to the API call. This example formats the input data as a JSON object as opposed to the previous examples which were standard variables.

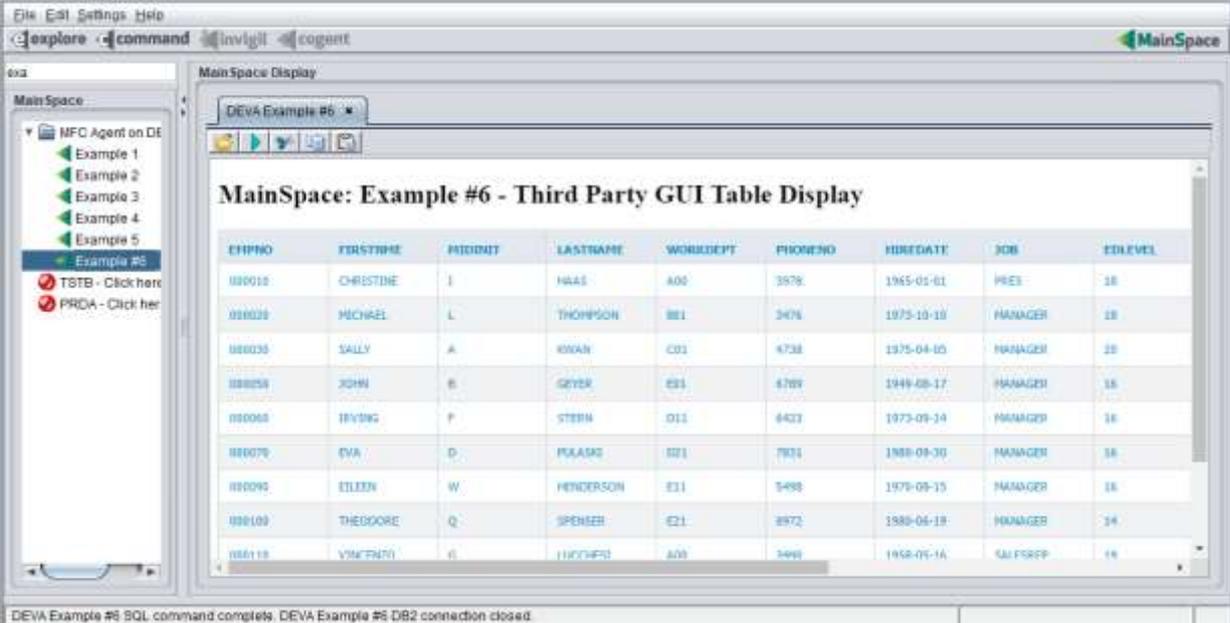
We create the JSON object (**json\_inputargs**) with the required fields (DB2SSID, callbackFunction, SQL), and we pass this object to the **runSQL** API method call as follows:

```

/*
 * MainSpace API call.
 * Perform an SQL query and call nominated user function when SQL query complete.
 */
MFCAgentRequest.newObject( "obj1" );
var json_inputargs = {
  "DB2SSID":"DBBG",
  "callbackFunction":"callBack( obj1.response );",
  "SQL":[ "SELECT * FROM DSN81110.EMP"
        ]
}
obj1.runSQL( json_inputargs );

```

## Expected Display after the Execute action



DEVA Example #6 SQL command complete. DEVA Example #6 DB2 connection closed.

## Where to Next?

Write your own Apps using your IDE of choice & explore the art of the possible.

Use the SAMPLE Apps in the Mainframe Cloud platform as inspiration.

Share your App ideas and interact with the MainSpace Github Community [here](#).

## MainSpace SAMPLEs

Mainframe Cloud will continue to provide SAMPLE MainSpace Apps to give you an idea of how MainSpace can be used. Only those MainSpace Apps that require some guidance will be listed below.

### DB2 View Sample App

This sample MainSpace App – DB2 View – allows you to interrogate DB2 subsystem IDs, schema's and tables that you have authority to view.

The following input fields are available:

#### DB2 SSID

The up to 4-character z/OS DB2 SSID you wish to interrogate.

#### Schema Filter

Optional: provide a string to filter the available Schema's for this DB2 SSID.

#### Table Filter

Optional: provide a string to filter the available Tables for this DB2 SSID and Schema Filter specified.

#### Maximum Rows

Specify a value otherwise the default will be used.

#### Maximum Columns

Specify a value otherwise the default will be used.

To run the DB2 View App:

1. Login to Mainframe Cloud.
2. Click **MainSpace** product
3. Click on the **SAMPLE – DB2 View** app.
4. Double click on the DB2 View TAB to open the iFrame & maximise the window.  
Note: this allows a full window view of the output.
5. Enter the values for each field above.
6. Click the **Load Tables** button.
7. Click on schema's/tables to interrogate the tables.
8. If necessary refresh the app by clicking the **Execute** icon or click on another Table.

Note: Some columns may be omitted if they contain values that are non-displayable. This DB2 View app will list these columns at the bottom of the output.