Product: Database Accelerator

Title: Implement static and dynamic queries using QuerySharePoint

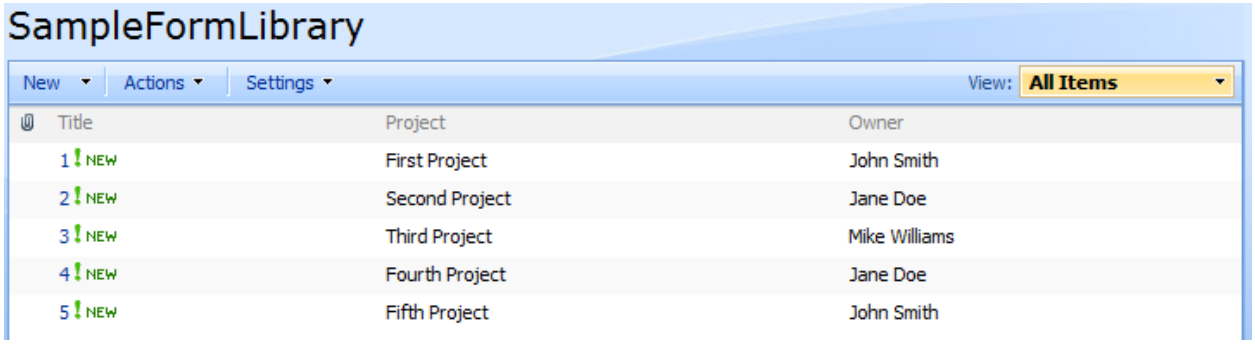
Qdabra's Database Accelerator (DBXL) allows you to obtain data from a SharePoint list by using the QuerySharePoint web method. This tutorial is a step-by-step guide to explain how to use this data in a dropdown control in your form.

Below you will find an outline of this document's contents.

PRE-REQUISITES

The steps below assume that you have already created a document library (or list) in SharePoint called SampleFormLibrary.

This document library or list should contain at least 5 submitted documents populated with Project and Owner values, as shown in the sample data below.

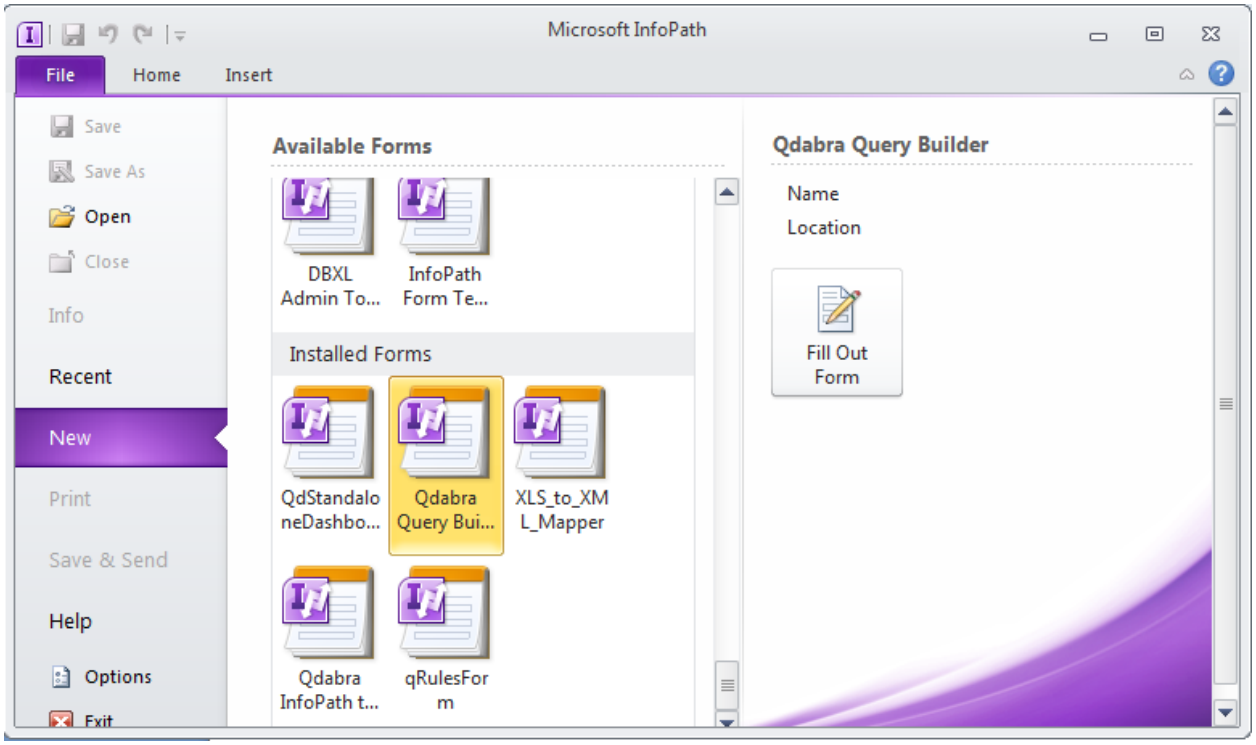


Before executing this tutorial, you must have installed Query Builder. For more information on installing Query Builder, please see: http://www.infopathdev.com/files/folders/other_subjects/entry81626.aspx

STATIC QUERY

GENERATE QUERYXML

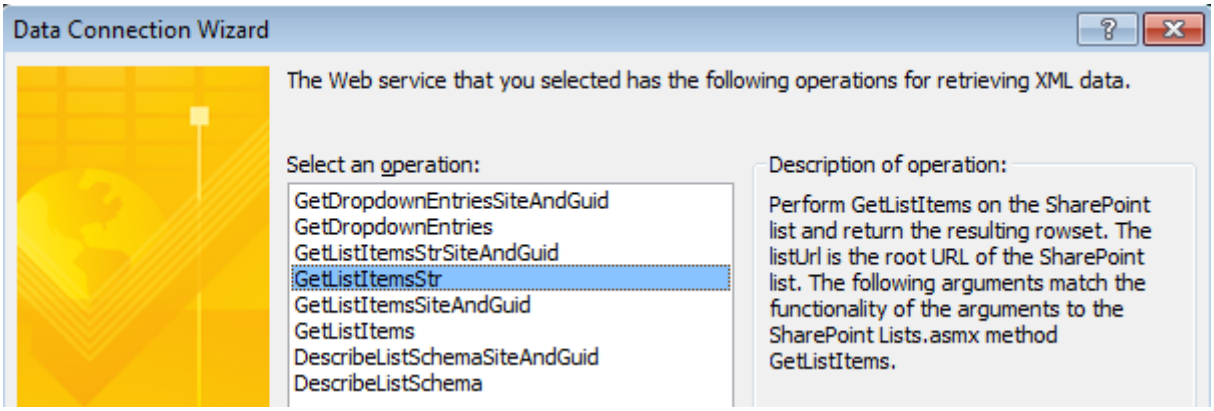
1. Open InfoPath 2010 Filler, select **Qdabra Query Builder** and click on **Fill out this form**.



2. In the Data Sources table, enter the root URL to your installation of DBXL in the **Web Service URL Prefix** field. (e.g. *http://<yourservername>/*)
3. The **Web Service Name** field is automatically populated to the default value, which is *QdabraWebService*. If you have chosen a different value, modify the content of the field.
4. For the **InfoPath Form Template** field, you only need to attach an InfoPath form if you are going to be using a field from the form to build the query. We will do this in a future step, but for now this can be left blank.
5. Enter the SharePoint List URL, for example, http://<yourservername>/Lists/SampleFormLibrary/.
6. Under **QueryColumns**, click on **Insert return column**.
7. Use the dropdown to select the column you wish yo return as part of your query.
8. Repeat step 7 to load as many columns as desired.
9. Click on **Build Now** and copy the first two **Query Strings** into Notepad.
10. Minimize QueryBuilder since we will use it later in this tutorial.

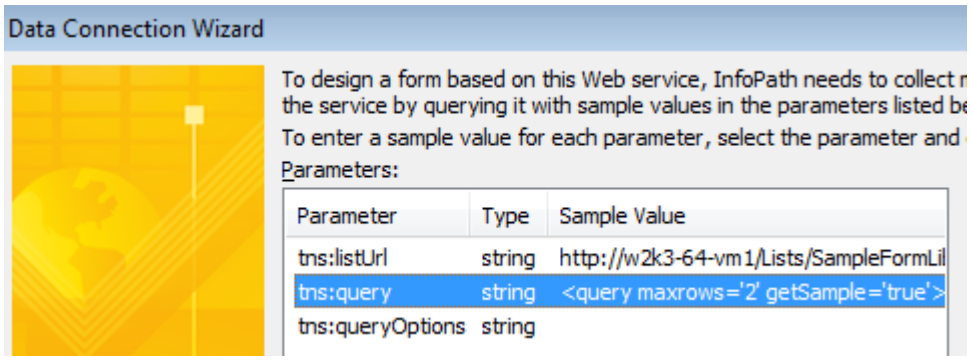
POPULATE A DROPDOWN USING QUERYSHAREPOINT

11. Launch the InfoPath 2010 designer. Select blank form and click **Design**.
12. In the Data tab, click on **Data Connections**.
13. Click on **Add**.
14. Select **Create a New Connection to Receive data**. Click **Next**.
15. From the following screen, select **SOAP Web Service** and click **Next**.
16. Type in the URL for the QuerySharePoint service, http://<yourservername>/QdabraWebService/QuerySharePoint.asmx and click **Next**.
17. In the following screen, select **GetListItemsStr** and click **Next**.



18. For the **listUrl** parameter, enter the URL to the SharePoint list.

19. For the **query** parameter, enter the first query generated by QueryBuilder.



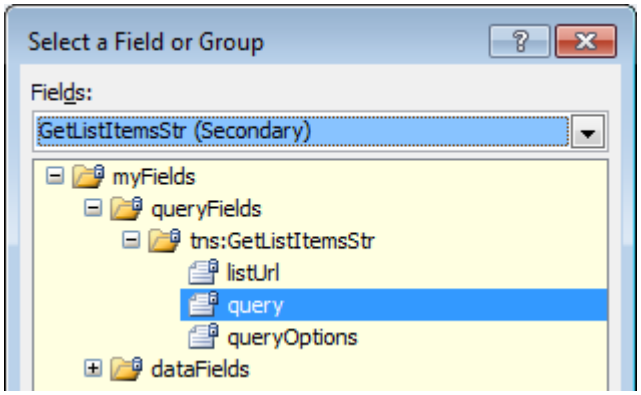
20. Click **Next**, enter the same values in the following screen, and click **Next** again.

21. Click **Next**, uncheck the checkbox for **Automatically retrieve data when form is opened**, and click **Finish**.

22. Click **Close** in the data connections dialog.

23. In the **Data** tab, click on **Form Load**.
24. Add a new **Action** and name it **QuerySharePoint**.

25. Next to "Run these actions" click **Add** and select **Set a field's value**.
26. For the **Field**, select the **query** node in the GetListItemsStr secondary data connection.

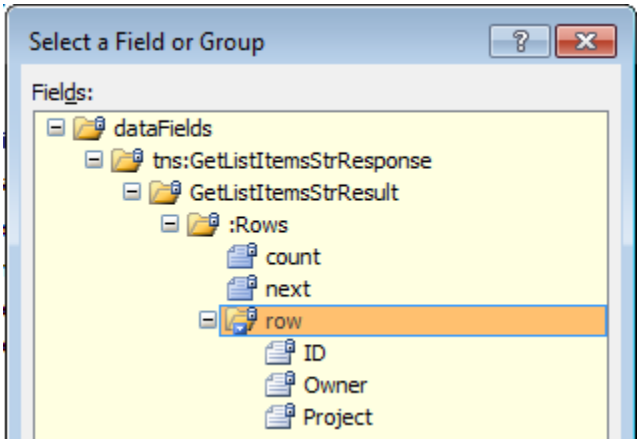


27. For Value, enter the second Query String generated by QueryBuilder. Click **OK**.
28. Add a second action, which queries for data using the GetListItemsStr data connection.

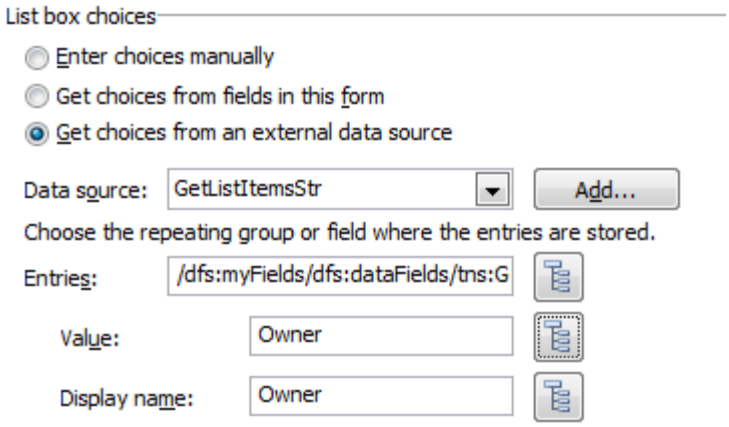
Now we just need controls to display the data in our form.

29. Add a dropdown list control to your canvas and set its name to *owner*.

30. Configure the dropdown list properties
    - Right click on your dropdown list and select **Properties**.
    - Under **List box choices**, select "Get choices from an external data source".
    - Make sure the GetListItemsStr is selected for the **Data Source**.
    - Click **Select XPath** button for **Entries**, drill down to the lowest folder in the schema (row).



    - Select this node and click **OK**.
    - For **Value**, select the field you want to store in the XML.
    - For **Display**, select the field you want to display in your dropdown list. Click **OK** twice.
    - We use Owner for both Value and Display because this is necessary for the filtering we will do in the next section.



31. Test your form.
    - Preview your form. The form will show you the display and not the actual value. The value gets stored in the XML (and stored in SQL, if you set up a database mapping).

ADD A DYNAMIC QUERY

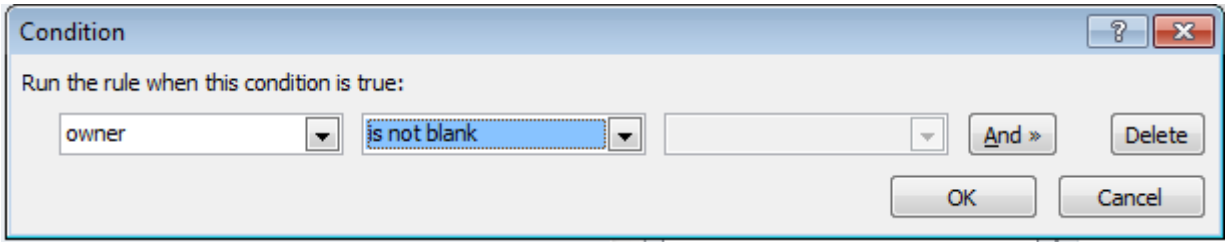
With a few additional steps we can implement a dynamic query that will call QuerySharePoint after a user provides some missing piece of data. First we need to modify the form, then modify the query generated by Query Builder.

UPDATE THE FORM

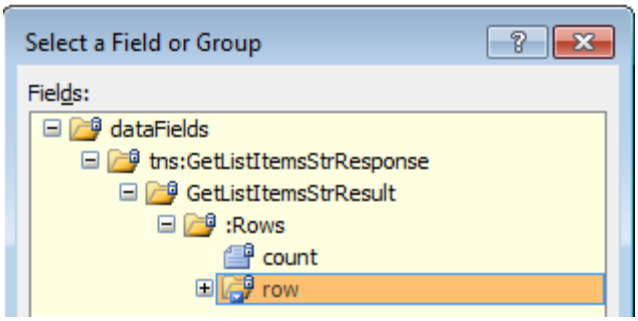
1. Add a second dropdown control to your canvas.
2. Follow steps 12-22 above to add a second data connection, and call it **GetProjects.**
3. Right click on the new dropdown control and select **Properties**. Change the field name to *projects*. Click **OK** to close the properties window. **Save** the form.

RETURN TO QUERY BUILDER.

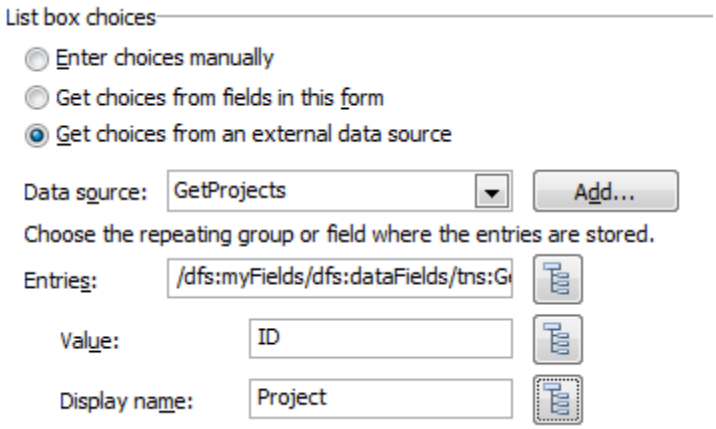
4. Attach your InfoPath form to the **InfoPath Form Template** file attachment control.
5. Click on **Insert comparison** and select the **Owner** column for the **Column Name**. Leave the Comparison operator set to **Equal**.
6. For **Value or Column Name**, click on the **Select Schema Node** icon  and select the *owner* node in your InfoPath form. What this means is that the QuerySharePoint data connection will only return the data that matches the value entered by the form user in the owner node.
7. Click on **Build Now** and copy queries #1 and #3 into Notepad. (You will notice that the first and second queries are the same.)
8. Back in the InfoPath designer, select the owner dropdown and click **Manage Rules**.
9. Click **New** and select **Action**.
10. Add a condition which will execute this rule only when **owner** is not blank.



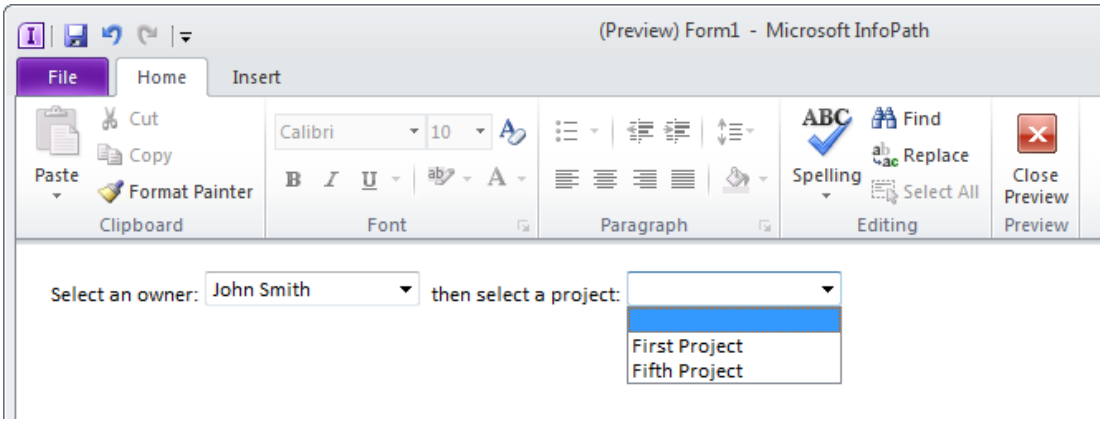
11. Under **Run these actions**, click **Add** and select **Set a field's value**.
12. Select the query node from the GetProjects secondary data connection. Then click **OK**.
13. Click on the icon  to the right of the **Value** textbox and paste the third query generated by Query Builder. Then click **OK** twice.
14. Click on **Add** and select **Query for data**.
15. Make sure that the data connection to GetProjects is selected from the **Data connection** dropdown, and then click **OK**.
16. Right click on your dropdown list and select **Properties**.
17. Under **List box choices**, select "Get choices from an external data source".
18. Make sure the GetProjects is selected for the **Data Source**.
19. Click **Select XPath** button for **Entries**, drill down to the lowest folder in the schema (row).



20. Select this node and click **OK**.
21. For **Value**, select the field you want to store in the XML.
22. For **Display**, select the field you want to display in your dropdown list. Click **OK** twice.



23. Save the form and click **Preview**.
24. Make a selection in the first dropdown, and watch the second dropdown get populated with the corresponding values!



HINTS AND TIPS

INSTALLING QUERYBUILDER

Sometimes installing QueryBuilder will generate the following error: "***Cannot create InfoPath.ExternalApplication object. Error code = 0x80080005***." This error is usually avoided if you select the "Just Me" option when installing QueryBuilder (instead of "Everyone").

INVALID XML

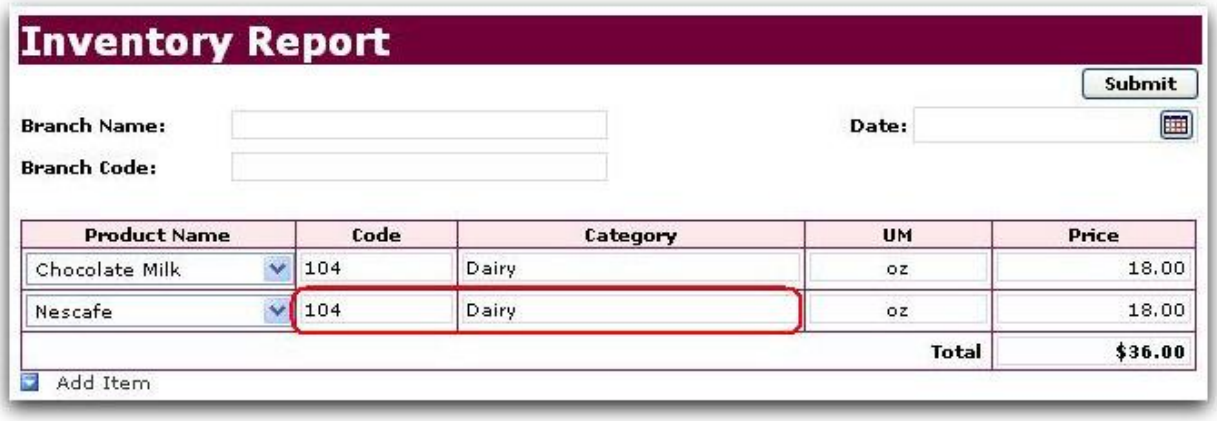
When working with QuerySharePoint or QueryDB, you might encounter an error that says: "Data at the root level is invalid. Line 1, position 1". This error indicates a problem with the <query> node. You could try using QueryBuilder to re-generate the value for <query>, but sometimes this indicates user error.

For example:

1. In a dynamic query, you should check that the <query> that begins with concat() is used in rules and not in the data connection wizard.
2. When entering the dynamic query into the rule, you must first click the function button (fx) before pasting the concat() query.

USING CURRENT() IN REPEATING TABLES

You might encounter a problem when using QuerySharePoint or QueryDB to dynamically retrieve values into a repeating table. To illustrate this scenario, imagine that a user selects a ***Product Name*** from a dropdown, and the goal is to set the ***Code*** and the rest of the repeating table fields with the corresponding values for the selected product. A second QuerySharePoint or QueryDB connection is created to dynamically query. The problem? The query (as generated by QueryBuilder) will not get updated when inserting new rows. The query always used the ***Product Name*** from the first row and returns the same value, as seen below.



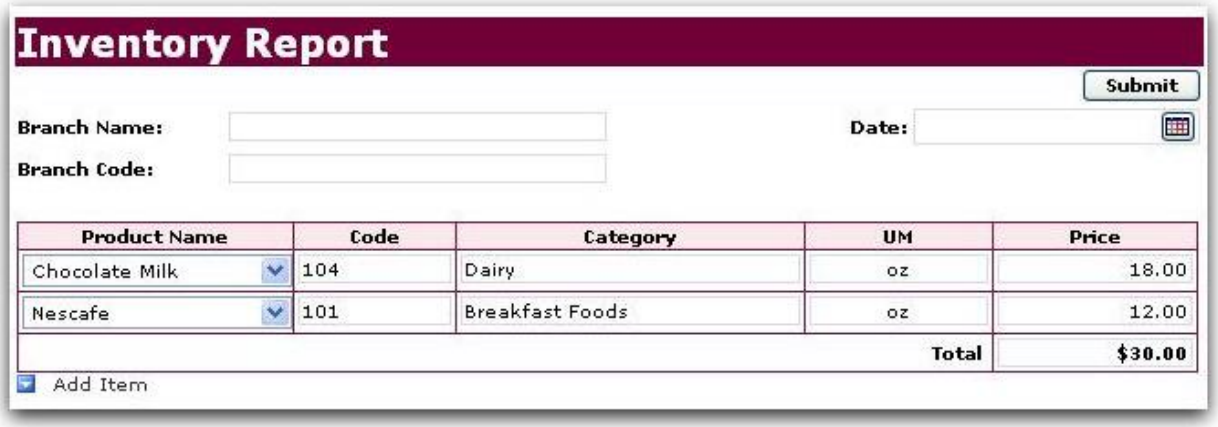
To identify the issue, we first need to look at the <query>that was created using Query Builder:

> *concat("<query><columns><column name='Product_x0020_Code'/><column name='Category'/><column name='Unit_x0020_Of_x0020_Measure'/><column name='Price'/></columns><filter><eq><column name='Product_x0020_Name'/><value>",* ***ProductName****, "</value></eq></filter></query>")*

To fix this, simple replace ***ProductName*** with current():

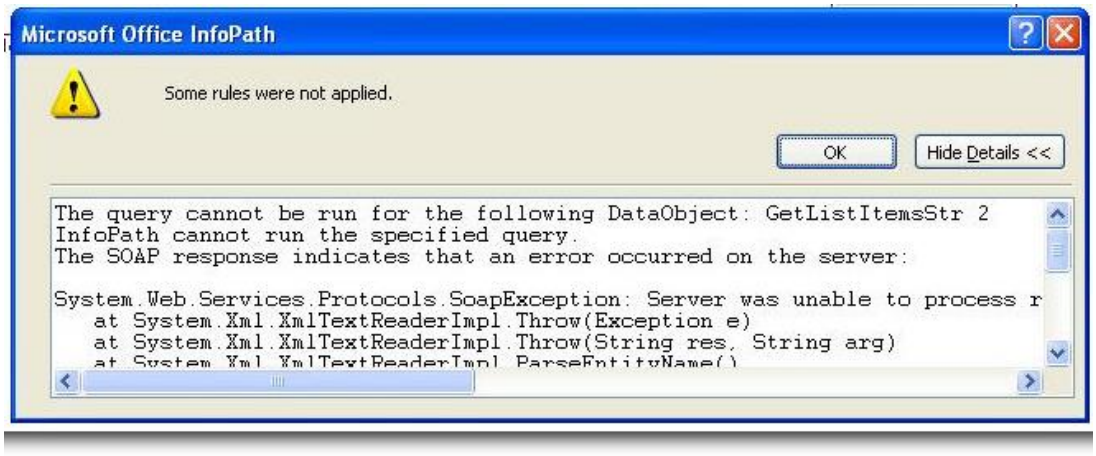
> *concat("<query><columns><column name='Product_x0020_Code'/><column name='Category'/><column name='Unit_x0020_Of_x0020_Measure'/><column name='Price'/></columns><filter><eq><column name='Product_x0020_Name'/><value>",* ***current()****, "</value></eq></filter></query>")*

Now, when the user inserts a new row and selects a different ***Product Name***, we get the correct data:



USING CDATA FOR SPECIAL CHARACTERS

When SharePoint or database queries contain special characters, you might encounter an error such as the one seen in the screenshot below:



This means that our database or SharePoint data contains special characters, and we must edit the value of <query> to use CDATA.

> *concat("<query><columns><column name='Product_x0020_Code'/><column name='Category'/><column name='Unit_x0020_Of_x0020_Measure'/><column name='Price'/></columns><filter><eq><column name='Product_x0020_Name'/><value>", current(), "</value></eq></filter></query>")*

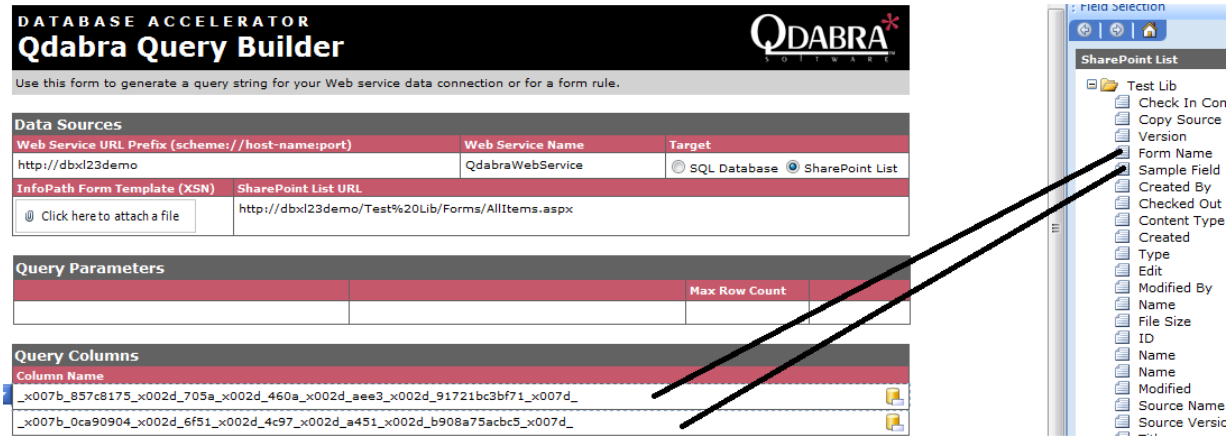
What we need to do is just add a CDATA inside the query <value> </value>:

> *concat("<query><columns><column name='Product_x0020_Code'/><column name='Category'/><column name='Unit_x0020_Of_x0020_Measure'/><column name='Price'/></columns><filter><eq><column name='Product_x0020_Name'/><value>**<![CDATA["**, current(), "**]]>**</value></eq></filter></query>")*

Note that CDATA should always start with the sequence: **<![CDATA[** and ends with **]]>**.

QUERYBUILDER DOES NOT SHOW USER-FRIENDLY COLUMN NAMES

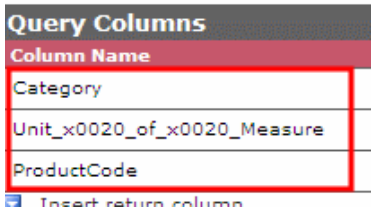
When using older versions of QueryBuilder to build SharePoint queries, you might notice that QueryBuilder might display column names that are not user-friendly, as in the screenshot below.



The ***<query>*** generated by QueryBuilder will still work, in spite of the column names.

QUERYSHAREPOINT CANNOT DERIVE A SCHEMA WITH BLANK FIELDS

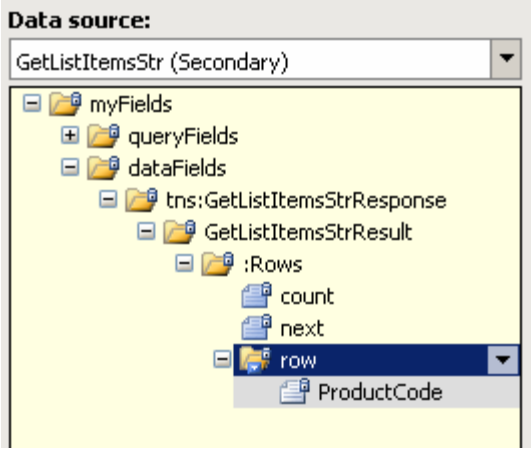
QueryBuilder allows you to specify the columns you wish to retrieve.



The screenshot above shows three columns being queried by QueryBuilder. The <query> that is generated looks like this:

> *<query maxrows='2'><columns><column name=**'Category'**/><column name=**'Unit_x0020_of_x0020_Measure'**/><column name=**'ProductCode'**/></columns></query>*

However, the data connection may not return all three columns, as seen below:



This happens when the source data contains blank values, which means that the web service was unable to derive the correct schema. The workaround is simple: remove the maxrows parameter:

> *<query><columns><column name=**'Category'**/><column name=**'Unit_x0020_of_x0020_Measure'**/><column name=**'ProductCode'**/></columns></query>*

USING COUNTONLY

The countOnly parameter allows you to return only the number of items, instead of returning the entire data set. To set the parameter to true, use a query like this:

> *<query **countOnly='true'** database="MyDB" table="MyTable"></query>*

If the countOnly paremeter is equal to anything other than 'true' (or if it is not present), then it is considered to be equal to false, and all records are returned.

ENCODED SPACES IN THE URL PRODUCE AN ERROR

When using QueryBuilder and QuerySharePoint, you need to enter the URL to your SharePoint list. Internet Explorer encodes spaces as %20, but DBXL produces an error if you use a URL containing a %20.

The workaround is simply to replace %20 in the URL with an actual space. This works in QueryBuilder as well as in the InfoPath data connection wizard.

SUPPORT

If you have questions about the information in this document, please contact us for assistance.

Licensed customers contact us via **Support@Qdabra.com**. Alternatively, please use the InfoPathDev.com Qdabra Product support forums to request help.