Integrating the Google Feed API with 4DAF

By Robert Molina, Technical Services Team Member, 4D Inc.

Technical Note 08-05

Abstract

The flexibility of the 4D Ajax Framework allows for it to interact and be integrated alongside other frameworks and API's. This Technical Note will cover integrating the Google AJAX Feed API into a 4D Ajax Framework web application using the new Data Grid 2.0 object. An example database is included.

Introduction

Ajax is a staple in every web application project currently produced. The benefits of Ajax are that it allows creation of rich internet web applications. Web pages are more interactive with features such as live scrolling and data validation which results in an end-user experience compared to that of a desktop application. The 4DAF enables 4D Developers to take advantage of these benefits with minimal code needed. Since its creation, the 4D Ajax Framework has matured and evolved due to the feed back provided by 4D Developers. As a result, the upcoming version of the 4D Ajax Framework will provide a new version of the Data Grid. This updated object, Data Grid 2.0, has made it possible for a grid to interact with other frameworks and API's such as the Google AJAX Feed API.

This Technical Note shows how to integrate the Google AJAX feed API with Data Grid 2.0 within the 4D Ajax framework.

What is Needed?

The example consists of the following:

- 4D Database
- 4D Ajax Framework
- Google AJAX Feed API

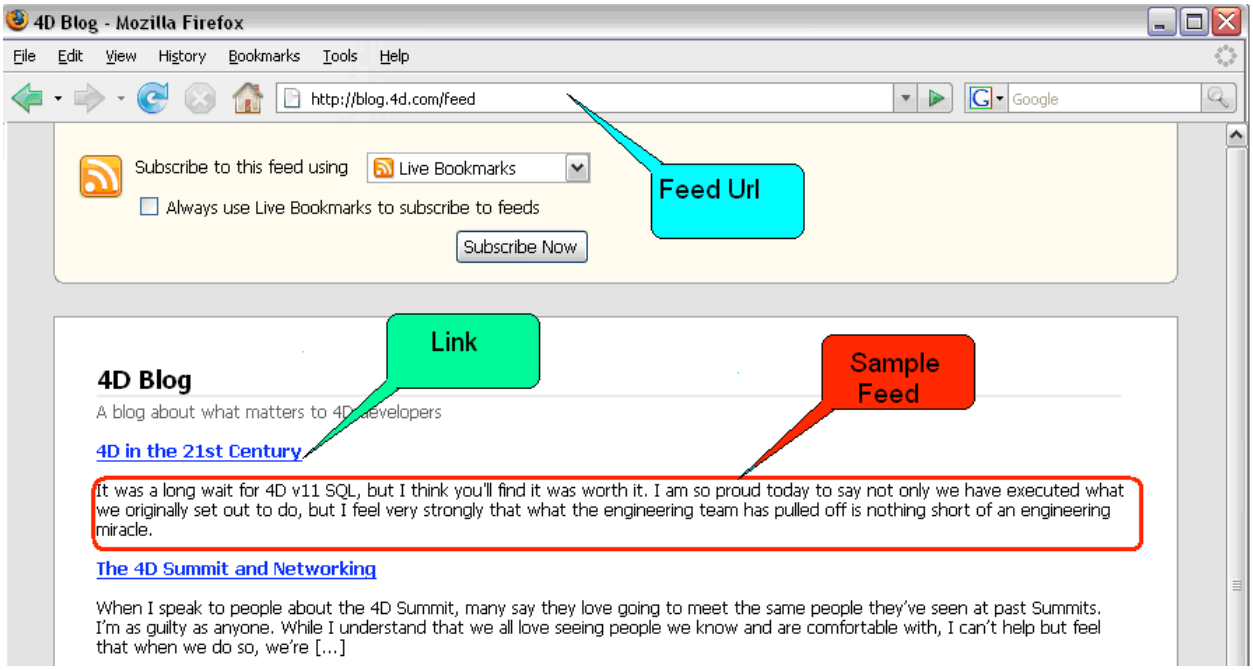
In order to combine all of the above components, knowledge of 4D and Javascript will be required. In addition, it is recommended to use Firebug add-on for Firefox to help aid in troubleshooting Javascript.

About the Google AJAX Feed API

One company that has embraced the Ajax technology is Google. Google uses Ajax within their web-based email client, feed reader, and their very popular map service. Out of these web-based applications, the Technical Note will focus on using the 4D Ajax Framework with the Google AJAX Feed API which uses the same feed fetching mechanism used in the feed reader service that Google provides.

What is a Feed?

A "feed", or web feed, is a data format that enables a web site to provide updated information to an end user. The end user would use an aggregator or feed reader to receive the feeds to view. Normally, a feed contains a small sample of an article or blog and if the user decides to view the entire content, a link is provided to view the content on the site. Below is a screenshot of Firefox's feed reader which lists feeds from 4D's blog site.



For more information regarding feeds visit:

http://en.wikipedia.org/wiki/Web_feed

What is the Google AJAX Feed API?

The Google AJAX Feed API allows the ability to access and download public feeds into an existing Ajax application using only JavaScript. Thus, this enables a developer to gather and use feed data from any public host. In addition, this same feed caching mechanism is used in Google's web-based feed reader, http://reader.google.com/. Some features provided by the API are the following:

- Supports the following feed formats (Atom 1.0, Atom 0.3, RSS 2.0, RSS 1.0, RSS 0.94, RSS 0.93, RSS 0.92, RSS 0.91, RSS 0.9)

- The data is returned in either two formats. One can return the data in JSON or XML format. The JSON format returns an abbreviated result format based

on the original feed. The XML format returns the actual XML content from the feed.

- Lines of Javascript code is reduced due to the use of the API.

***Note**: Feed data fetched from the Google AJAX Feed API comes from the results of Google's feed crawler ("Feedfetcher") which is normally updated every hour, thus some feed data might not be fully up to date.*

***Note**: By default, the API returns the feed in the JSON format.*

Why integrate the Google AJAX Feed API with 4DAF?

Integrating the Google AJAX Feed API into an existing 4D Ajax Framework application allows data from 4D to be mashed up with any public feed data. This is useful in that feed data can be pictures, news reports, and blogs. The Google AJAX Feed API centralizes this information and makes it easily accessible for the 4D Ajax Framework to access with a very small amount of Javascript code added. Another thing to note is 4D Ajax Framework is only communicating with Google.com servers, thus code that executes on the client browser does not directly communicate with the host feed site.

Terms of Use for Google AJAX Feed API

In order to use the Google AJAX Feed API, the developer must agree to make the service available to end users free of charge charge. For more information regarding the terms of use, go to the following:

http://code.google.com/apis/Ajaxfeeds/terms.html

Data Grid 2.0

The Technical Note example database uses the new Data Grid 2.0 object within the 4D Ajax Framework. This object replaces the previous object, Data Grid. In the context of this Technical Note the benefit of this object is that it now has a click event that provides a record reference of the row selected.

Example Database

The example database included with this document comes in three flavors:

- Interpreted Source Database for 4D 2004
- Interpreted Source Database for 4D v11 SQL
- Merged Database

The interpreted source databases allow you access to Developer Hooks and to perform further customization in the back end. Otherwise, feel free to connect to the merged version of this database.

The code and screenshots in this Technical Note will apply to both 2004 and 11, unless otherwise noted. Basically, the example database does the following:

1. The Data Grid 2.0 is loaded, fetches records from the Feeds Table, and displays the feed_name field.

2. When a user clicks on a record within the Data Grid 2.0, the feeds are then displayed on the page.

Note: *The example provided is based on a BETA version of the 4D Ajax Framework. In order to use the interpreted source databases you will need this version. As this version has not been released, for the purposes of testing the example application, please use the merged version of the example.*

For the latest information regarding 4D product releases please visit:

http://www.4d.com/corporate/releases.html

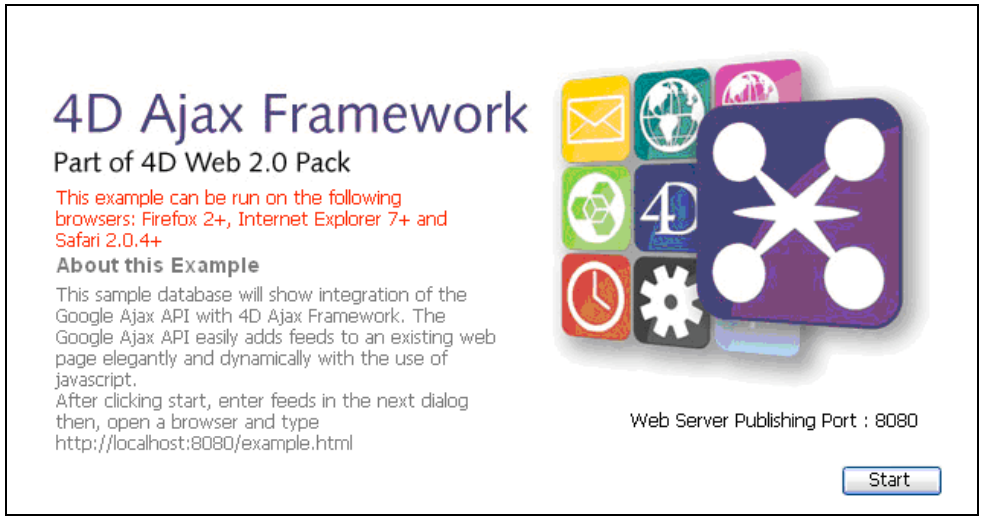
Database Structure

The example database utilizes one table to store the feed URL and feed name.

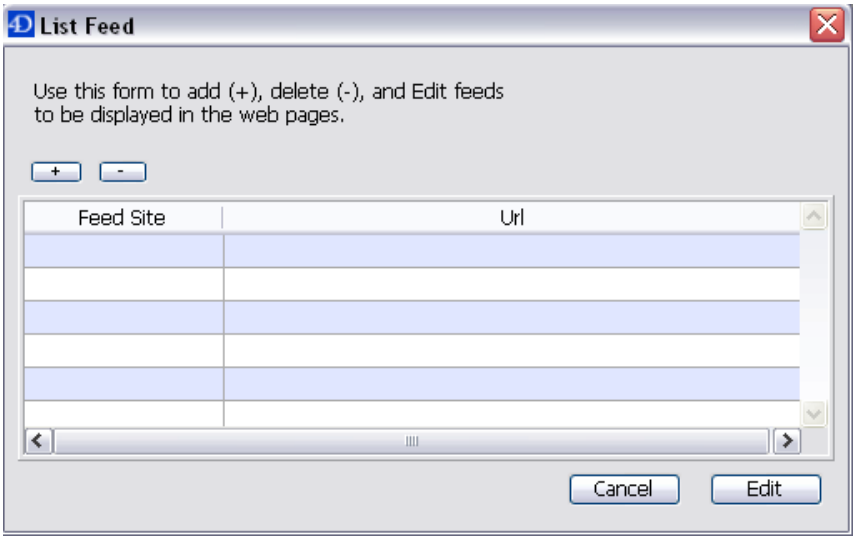
Note*: In the 4D 2004 version the [Dialog] table was created to associate Dialog forms. In 4D v11 SQL Project Forms are used.*

Using the Database

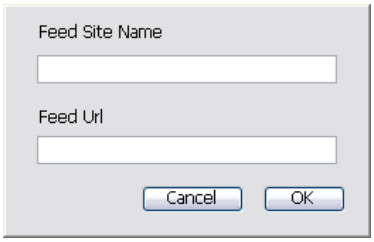
When launching the database the following screen is displayed:



Clicking on the "Start" button then displays the following form.

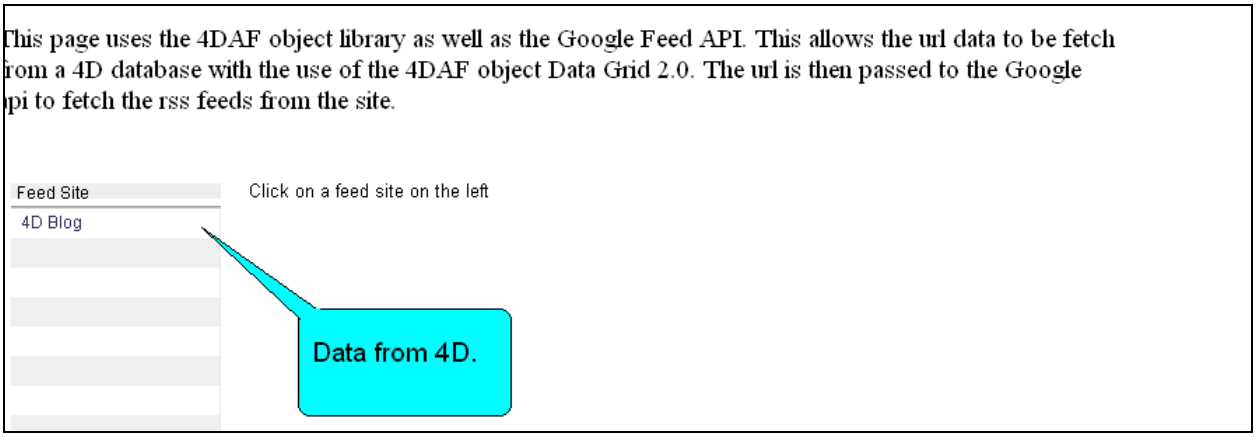


This form shows the current feeds stored within the database. In addition, this form provides the means to delete, add, or edit a feed. To add, click on + . After clicking that button the screenshot below is displayed.



Enter a feed name and associated URL. For example, for the Feed Site name enter "4D Blog". For the Feed URL enter http://blog.4d.com/feed.

Now open a browser and type in http://localhost:8080. The web page below should be displayed:



Clicking on the record 4D Blog will produce the page below:

Example Database Implementation

As mentioned previously, this example database incorporates Javascript as well as 4D code.

Initialization

Before any object provided by the 4D Ajax Framework or Google AJAX Feed API can be displayed on the page, initialization needs to be done. For the Google AJAX Feed API the following two scripts will need to be present:

Line12 of example.html

```
<script type="text/javascript"
src="http://www.google.com/jsapi?key=ABQIAAAAcR25AOG2Q54m1hJyBm22-
RQ16EPBx6Yn33_vZRCessV7V4PFpxT4qbAqfqSBXWTwZMBYtg5PhOENrQ"></script>
```

Line 19 of example.html

```
<script type="text/javascript">
  google.load("feeds", "1");//
</script>
```

In order to use the Google AJAX Feed API, a key is provided by Google to authenticate requests. To generate the key, use the following link to register.

http://code.google.com/apis/Ajaxfeeds/signup.html

Once receiving the key, place it in a script such as line 12 above. With that line present, the code in line 19 can then execute. This code asks Google to load version 1 of the feeds API.

Note: *The current version for the Google AJAX Feed API is version 1.*

On the 4D Ajax Framework side of things, the following code needs to be included.

```
<script language="javascript" type="text/javascript"
src="dax/dev/callbacks.js"></script>
<script language="javascript" type="text/javascript"
src="dax/js/framework.js"></script>
```

The above statements notify the browser that there will be calls made to these files. One of the calls is the following function.

Line 25 of example.html

```
function dax_load()
{
    Login("Guest", "");
}
```

When the page loads, the 4D Ajax Framework needs to authenticate with the connecting 4D database with user “Guest” and password “”. In order for this to

pass authentication, the *Dax_DevHook_Login* project method within the 4D structure needs to have the following code:

```
If ($userName_t="Guest")
   If ($password_t="")
         $loggedIn_b:=True
   End if
End if
```

The code above checks for user "Guest" and password "". If the parameters match, then allow access to the database. Once access granted, the onAfterInit() function is executed.

Displaying Data Grid 2.0

The onAfterInit() function contains the code to start displaying the objects:

```
function onAfterInit()//Guest user successfully authenticates, execute the code
{
//Create Data Grid 2.0
var myDataGrid = new dax_dataGrid('Feeds', $('griddiv'),1,0,false);
myDataGrid.go();//Initializes and shows the Grid
myDataGrid.setSelectionMode('single');//set the Grid to single selection mode
//When a row is clicked execute this function
myDataGrid.onDataRowClick = function (row, recordId) {
   ddwcontacturl="/dax/getddw?ddwid=100001&sessionid="+dax_bridge.sessionId+"&r
ecordid="+recordId
   callDDW(ddwcontacturl);

}
}
</script>
```

The first line in the function is:

```
var myDataGrid = new dax_dataGrid('Feeds', $('griddiv'),1,0,false);
```

This creates the myDataGrid object which uses the 'Feeds' selection. Other properties set for this object is 1 header row, 0 locked columns, hide control column, and lastly place the object in the div "griddiv".

```
myDataGrid.go();
```

The above line of code then displays the grid.

```
myDataGrid.setSelectionMode('single');
```

Once displayed, the selection mode for the object is set to single selection. At this point, the Data Grid 2.0 object is displayed on the page.

Displaying the feeds

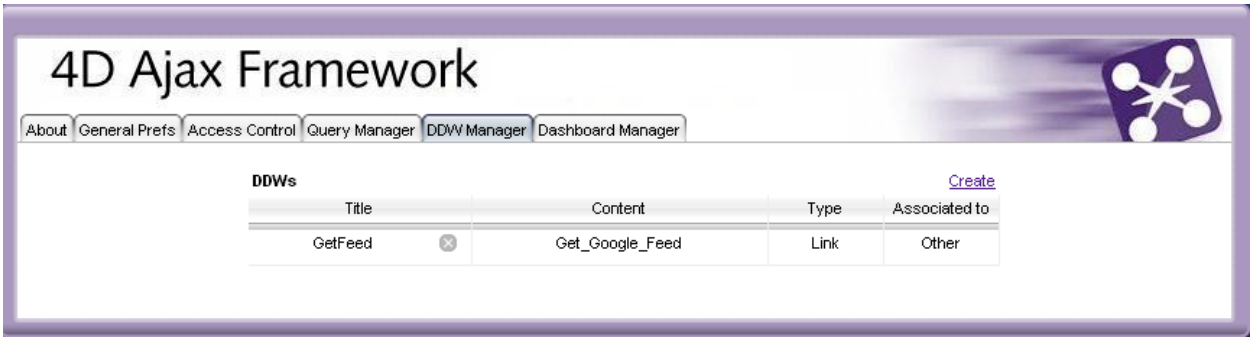
To display the feeds, a click event needs to occur on the Data Grid 2.0 object. When this occurs, the following function is executed:

```
myDataGrid.onDataRowClick = function (row, recordId) {
    ddwcontacturl="/dax/getddw?ddwid=100001&sessionid="+dax_bridge.sessionId+"&r
ecordid="+recordId
    callDDW(ddwcontacturl);
}
```

This function obtains the record ID of the selected item within the grid. The code then goes on to pass the ddwcontacturl variable to the function callDDW. The ddwcontacturl variable is a url string that will be sent to the 4D Web Server by the function makeCall.

```
function callDDW(aURL){
    makeCall(aURL, myDDWhandler);//This function calls the DDW at the backend
}
```

The URL string contains the ID of the DDW method, 100001. The DDW was created using the DDW manager:



The DDW method being called is *Get_Google_Feed*. Below is the code for the method.

```
C_TEXT($0;recordID)
ARRAY LONGINT($RecordNumbers_al;0)

DAX_Dev_DDW_GetAttributes ("RecordNumbers";->$RecordNumbers_al)
If (Size of array($RecordNumbers_al)>=1)
    GOTO RECORD([Feeds];$RecordNumbers_al{1})
    $0:=[Feeds]feed_url
Else
    $0:="No URL"
End if
```

This code receives the selected record numbers in the array $RecordNumbers_al. Since single selection mode was set for Data Grid 2.0, the array is expected to only have 1 element. That element is then used to load the record. Once the record is loaded, the method returns the contents of [Feeds]feed_url to the front end.

The front end waits for this response with the function myDDWhandler:

```
function myDDWhandler(http_response)//function that receives response and
passes
                                    //the response to the GetGoogleFeed
function.
{
    if (http_response.readyState != 4)
        return;

var response =
http_response.responseXML.getElementsByTagName("GetDDW").item(0);
if(response){
        // Get Returned URLfrom the XML response
        var responseUrl = response.getAttribute('url');
        if (responseUrl) {
            getGoogleFeed(responseUrl);
        }
    }
}
```

The response is received by this function and then passed onto the getGoogleFeed function. The function below then fetches the feed and draws it within the div "feedControl".

```
function getGoogleFeed(responseUrl{
        var feedControl = new google.feeds.FeedControl();
        feedControl.setNumEntries(7);
        feedControl.setLinkTarget(google.feeds.LINK_TARGET_BLANK);
        feedControl.addFeed(responseUrl);
        feedControl.draw(document.getElementById("feedControl"));
}
```

This line creates an object feedcontrol of type google.feeds.FeedControl():

```
var feedControl = new google.feeds.FeedControl();
```

This specifies the number of entries that will be fetched from the Google servers:

```
feedControl.setNumEntries(7);
```

This sets the property within the feeds to be opened in a new tab or window:

```
feedControl.setLinkTarget(google.feeds.LINK_TARGET_BLANK);
```

This adds the feed to the feedcontrol object.:

```
feedControl.addFeed(responseUrl);
```

Lastly, the object is then drawn in the "feedControl" div:

```
feedControl.draw(document.getElementById("feedControl"));
```

As a result, the feeds are displayed on the web page.

Conclusion

With a small amount of Javascript added to an existing 4D Ajax Framework web application, public feed data can be obtained through the Google AJAX Feed API. With the addition of the new Data Grid 2.0 object in the 4D Ajax Framework, 4D Developers can now use the grid object to interact with other API's and frameworks.

A Note about 4D Web 2.0 Pack

The products in 4D Web 2.0 Pack are a departure from most other 4D products. As 4D Web 2.0 pack is a subscription-based product it is expected that incremental releases will be made. Thus please note that this Technical Note is based on 4D Ajax Framework v11. As new features are implemented this Technical Note may become obsolete (faster than most other 4D products).

Related Resources

Google AJAX Feed API Developers Guide:
http://code.google.com/apis/Ajaxfeeds/documentation/

4D Ajax Framework Data Grid 2.0:
http://www.daxipedia.com/index.php/Data_Grid_2.0

Note: *Data Grid 2.0 is still in beta therefore information in the link provided is subject to change when released.*

For the latest information on the 4DAF consult the latest documentation and also check the 4D Web 2.0 Pack Wiki:
http://daxipedia.4d.com

For the latest news about 4D Web 2.0 Pack or to find out how to purchase it see:
http://www.4d.com/products/4dweb20pack.html