6 – 7 . APRIL

ATLANTA, GA, USA

SYNERGY 2017

Building for the future. Better, faster, everywhere.

DataFlex for WordPress (DF4WP)

Mike Peat
Unicorn InterGlobal

WordPress

Originally just an open source blogging tool, it has become the most widely deployed content management software on the Web:

- Of all the sites on the Web, 45% use *some* form of content management system
- Of those, almost 60% use WordPress (source: w3techs)
- Hence **over 25% of all (public) web sites use WordPress** (~75 **million** sites - source: ManageWP)
- By comparison, the next two most popular content management systems - Joomla and Drupal - are used by 2.7% and 2.2% of sites respectively

WordPress as a platform

- WordPress has **44,000+** plugins available for it, allowing it to be customised for all sorts of uses, from eCommerce, to appointment scheduling, to image galleries… almost anything
- Together they make WordPress effectively a platform in its own right
- That is a platform we can now deliver applications for… DataFlex *everywhere*!

DF4WP Plugin

- The DF4WP plugin allows you to integrate your DataFlex web applications in a WordPress site (requires DF 18.2+)
- Existing vertical or horizontal apps can be quickly modified for integration (without compromising their existing functionality)
- New applications can be written to order

What is it exactly?

DF4WP has two parts to it:

1. The actual WordPress plugin, made up of a small set of PHP, JavaScript and CSS files
2. A DataFlex library workspace containing classes modified to work better under WordPress

The Plugin

The plugin itself is delivered as a zip file containing all that is required on the WordPress side (dataflex4wordpress.zip)

WordPress expects plugins to be in zip format, so you should not extract those files, but upload the zip file as-is to the target WordPress site

Installing the Plugin

You must have a login for the site with Administrator (or "Super Admin") rights

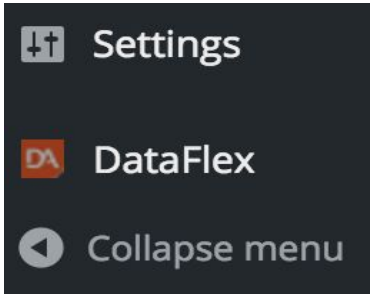
On the Administrator dashboard, in the left-hand menu, go to "Plugins" → "Add New"

Click "Upload Plugin", then "Choose File", selecting the dataflex4wordpress.zip file

Installing the Plugin

Click "Install Now" then on the resulting page, click "Activate Plugin"

You should now see a "DataFlex" entry appearing at the bottom of the dashboard left-menu:



Install Demo

Configuring the Plugin

Hovering over, or clicking, the "DataFlex" entry on that left menu will show two sub-items:

- Dashboard
- Help

The Dashboard is where configuration entries are made

Configuring the Plugin

There are only three items to configure:

- Path to the default DataFlex WebApp
- Default theme (to preload)
- Secret (32 char key for data passed to the WebApp)

The Path and Theme are simply defaults - they can be overridden - but the Secret is important; if left blank one will be generated

Configuring the Plugin

Clicking the "Save Changes" button will update the plugin's properties and you are ready to go!

Configure Demo

Embedding a WebApp (or View)

Create a new WordPress page: Dashboard → Pages → Add New

Give it a title

Enter a "shortcode": [df4wp-webapp]

Add settings to the shortcode

Click the "Publish" button, then the "View page" link

Shortcode settings

- **view**="*oViewObjectName*" (embed a view)
- **height**=*999* (in pixels for the app/view)
- **apppath**="*/path/to/the/webapp*"
- **theme**="*ThemeToPreload*"
- **params**="*prop=val|prop=val|prop=val*"
- **dfr**=TRUE (if you need the DataFlex Reports previewer web components loaded)

The "params" setting

Allows you to dynamically set Web properties of the embedded object (app or view)

Pipe "|" separated list of property-value pairs:

"property=*value*|property=*value*"

Two new web properties added for views:

psTheme - allow views to be themed

pbShowToolBars - show the command bars

Embedding Demo

Customising for WordPress

Use the DF4WP library in your Workspace

Several classes should then be changed:

- cWebApp ⇒ cWPWebApp
- cWebView ⇒ cWPWebView
- cWebModalDialog ⇒ cWPWebModalDialog
- cWebImage ⇒ cWPWebImage
- cWebColumnImage ⇒ cWPWebColumnImage

The appropriate Use statements should also be changed

Customising for WordPress

These modifications should **not** change the behaviour of the application outside WordPress

You can maintain a single code-base for your app for stand-alone deployment or embedding within a WordPress site

Customising for WordPress

- A new WebApp property - pbUseWPLogin - can be set to allow your application to use WordPress's logged-in state for the user rather than the DataFlex framework login
- This is a developer choice: should the user have to log in at all? If so, should it be to WordPress or specifically your application?

Customising for WordPress

The 32 character "secret" we created in the plugin configuration needs to be known to the application

Set psWPEncryptSecret to its value

Either in the source code or from a DB field

New boolean WebApp function:
InWordPress

Data passed from WordPress

To support this, there is also a new function of the WebApp: GetWPInfo

Pass it the (string) name of the data you want and it will return the value

Data is passed AES-256 encrypted from WordPress with checking on an MD5 hash

Data passed from WordPress

- sWPTimeSent
- bWPUserLoggedIn
- sDF4WP_Parameters
- sWPView
- sWPSiteAddress
- iWPUserID
- sWPUserDispName
- sWPUserEMail
- sWPUserLogin
- iWPUserLevel
- sWPUserFirstName
- sWPUserLastName
- sWPLoginURL
- sWPHomeURL

Data passed from WordPress

To access those values we can code, for instance:

Get WPInfo of ghoWebApp "iWPUserLevel" to *var*

Extending that information

If you have an understanding of WordPress's functionality

And you are not afraid of a little PHP code

You can freely add to that passed data

Extending that information

In the WordPress Administrator Dashboard

Under "Plugins" on the left-menu

There is an "Editor" option

Select the DataFlex4Wordpress plugin

Then the df4wp-page.php file

Extending that information

Scroll down to around line 60:

```
...
$info .= chr(31) . 'sWPUserLastName=' . $current_user->user_lastname;
$info .= chr(31) . 'sWPLoginURL='     . wp_login_url();
$info .= chr(31) . 'sWPHomeURL='      . home_url();
```

Add lines to that, for instance:

```
$info .= chr(31) . 'sWPTheme='        . wp_get_theme()['Name'];
$info .= chr(31) . 'sWPThemeVer='     . wp_get_theme()['Version'];
```

The apppath setting

The path to your DataFlex web app can be set either in the plugin's settings (a default) or specifically in the shortcode via the apppath setting

That path should start with a slash - / - but if it doesn't one will be added

That path is to a virtual directory on the WordPress server

However if your application is on a different server - which it would need to be if the WordPress server is a Unix or Linux machine for instance - you can use the full path: "//DNSName.Or.IP.Address/path/to/webapp"

Cross-Origin Resource Sharing

If you do require to load your web app from a different server, then that server will have to be set up for Cross-Origin Resource Sharing (CORS)

See:
http://www.unicorninterglobal.com/Company-White-Papers-Cross-Origin-Resource-Sharing-CORS-816

Cross-Origin Resource Sharing

The first thing to know is that CORS is done on **the WebApp machine** - not the WordPress machine

You have to configure it to respond to a "*preflight*" HTTP OPTIONS (as opposed to GET or POST) request from the user's browser

That will ask if it is prepared to allow access for a page loaded from the WordPress server

It has to respond with a series of "Access-Control-Allow-..." HTTP Headers that will permit access

Cross-Origin Resource Sharing

In **IIS Manager**, in the "Features View" for your WebApp virtual directory, you use the "**HTTP Response Headers**" option to "**Add**" four new headers:

- **Access-Control-Allow-Origin** - set to the WordPress hostname or IP address
- **Access-Control-Allow-Methods** - set to GET, POST
- **Access-Control-Allow-Headers** - set to content-type
- **Access-Control-Allow-Credentials** - set to true

Cross-Origin Resource Sharing

Then you should rearrange the Handler Mappings to ensure the Options Handler grabs the request first:

Use the IIS "**Handler Mappings**" feature and in that click the "**View Ordered List**" link

Locate and select the **OPTIONSVerbHandler** (probably near the bottom), then click the "**Move Up**" link (it will ask if you really want to reorder the list - say Yes) until it is at the top

Cross-Origin Resource Sharing

You should then be able to embed your web app in WordPress on the target machine

To embed a given web app in WordPress sites on more than one server, it gets trickier, because using a wildcard ("*") in Allow-Origin is incompatible with also using Allow-Credentials - which WebApp needs for its cookies, but it can be done

Cross Site Demo

Thank you!

Any Questions?