Using the DataFlex for WordPress (DF4WP) plugin

Contents

Introduction

WordPress is an Open Source Content Management System and the most popular blogging software in use on the Web today. WordPress is based on PHP and MySQL, and is supported by a huge array of "plugin" components (over 40,000) which can be used to extend and tailor the functionality of the base software to such an extent that WordPress can be considered a platform in itself.

The DF4WP plugin allows DataFlex web applications or individual views to be embedded on WordPress pages, meaning that you can fully integrate your DataFlex applications within a WordPress site.

To use the DF4WP plugin involves setting up four different sets of software on the relevant server (in the event that they are not already present):

- Microsoft's Internet Information Services (IIS).
- WordPress (which also involves installing PHP and MySQL).
- The DataFlex WebApp Server.
- The DF4WP plugin.

We will deal with each of these in turn.

This document will assume that the target server environment is **Microsoft Windows Server 2012 R2**, although earlier Windows Server versions can also be used, or a workstation environment with **Microsoft Windows 7** (the procedure should be similar for Windows 8 or 10).

Before You Start

Google Chrome

We recommend that as a first step you download and install Google Chrome from https://www.google.com/chrome/.

Internet Explorer will make this process difficult by requesting that you "trust" each host that data is being accessed from, requiring a series of clicking "Add", "Add", "Close" buttons in pop-up windows before the page loads.

Clicking "Download now" followed by "Accept and Install" will trigger another of these sequences, causing the initial download to fail: click the browser's back button and repeat, then "Save" followed by "Run" to finish installing), making Chrome your default browser rather than Internet Explorer. This will make many processes much less painful.

You can then open this document in Chrome so that you can follow the required steps on your server as you work through them.

Notepad++

Also recommended is downloading and installing Notepad++ (from https://notepad-plus-plus.org/, using Chrome) which will make editing various configuration files (which are usually in Unix format) easier than using tools such as Windows Notepad.

Installing Internet Information Services (IIS)

In most cases Internet Information Services (IIS) will already be installed on the server, however in the event that it is not, go through the following steps to set it up:

1. Run Server Manager (there is usually an icon for it already on the taskbar, however if not then use "Start" (the Windows icon on the left of the taskbar) → "Administrative Tools" → "Server Manager".
2. Select "Local Server".
3. Scroll down to the "Roles and Features" pane.
4. In the "Tasks" combo (top-right of the pane), select "Add Roles and Features".
5. In the wizard, click "Next".
6. Select "Role-based or feature-based installation", then click "Next".
7. Select the local server and click "Next".
8. Scroll down to and check the "Web Server (IIS)" option.
9. Click "Add Features" in the pop-up.
10. Click "Next".
11. In "Features" accept the defaults and click "Next", then "Next" again.
12. In "Role Services" expand "Application Development" and check "ASP", "ISAPI Extensions" (the first may cause the second to be selected automatically) and "CGI" - click "Add Features" in the pop-up.
13. Expand "Management Tools".
14. Check "IIS Management Console", then expand "IIS 6 Management Compatibility".
15. Check "IIS 6 Metabase Compatibility".
16. Click "Next".
17. Click "Install" (the process may take a few minutes).
18. Click Close when the process is completed (it should say "Installation succeeded on {*machine-name*}" under the progress bar).
19. Open "Administrative Tools" (Task-bar → Start → Administrative Tools) and double-click "Internet Information Services (IIS) Manager".
20. Expand the local server and select "Application Pools".
21. Select the "DefaultAppPool" and click the "Advanced Settings" link in the right-hand pane; in the pop-up, ensure that "Enable 32-bit Applications" is "True" (double-clicking on "False" will change it to "True" and visa-vera"), then click the "OK" button.

Installing WordPress

The process of installing WordPress differs depending on whether you are installing on a server operating system or a workstation operating system.

On a server the process is made quite simple through the use of Microsoft's Web Platform Installer (WPI) which installs everything required, however on a workstation (Windows 7, 8 or 10) the WPI will install the WebMatrix program and IIS Express (a lightweight version of Internet Information Services) regardless of whether the full version of IIS is present on the machine, and will install WordPress under that. This is most probably not what you want.

Refer to the appropriate section below.

Server Install

In many cases where you wish to deploy the DF4WP plugin on a server, WordPress will already be installed and running, however if it is not, then use the following procedure:

1. Download and install the Microsoft Web Platform Installer (Web PI, or WPI) from http://www.microsoft.com/web/Downloads/platform.aspx (if you have not already done that).
2. Run WPI (it will most likely run automatically).
3. Type **WordPress** into the search box and press Enter.
4. Select WordPress (click "Add").
5. Click "Install".
6. Select "MySQL" as the database and give it a "root" password, then click "Continue".
7. Click "I Accept" in the pop-up (the install may take a few minutes).
8. In "Configure" use the Default Web Site.
9. Supply eight "unique phrases" (anything will do - random characters are fine so long as there are enough of them) and click "Continue".
10. Click the "Copy to clipboard" link to copy database credentials to the clipboard, then paste them somewhere you can find again.
11. Click "Finish" in the WordPress install dialog.
12. A WordPress setup page should now appear in your browser.
13. Select your preferred language.
14. Supply a Site Title, Username and E-Mail address - make sure you note down the password - then click "Install WordPress".
15. Click the "Log In" button.
16. Log in - you should be taken to the site dashboard.
17. In the left-hand pane, select "Settings", then "General".
18. In both "WordPress Address (URL)" and "Site Address (URL)" change "localhost" to the hostname or IP address of the server, then scroll down and click "Save Changes".
19. You might then need to log-in again.

Workstation Install

To install WordPress on a workstation, we strongly recommend that you do **not** follow the procedure for installing on a server above for the reasons previously stated.

Instead follow the steps below to separately install:

1. PHP - the language WordPress is written in
2. MySQL - the commonest (and free) database server used with WordPress
3. WordPress itself

Install PHP

1. From http://windows.php.net/download/ select and download the appropriate zipped version of PHP for your operating system. At time of writing (March 2016) these were:
    a. For 32-bit machines: "VC14 x86 Non Thread Safe" zip
    b. For 64-bit machines: "VC14 x64 Non Thread Safe" zip
2. Extract the downloaded file to a location on your machine (C:\PHP for instance).
3. In that directory, copy the file "**`php.ini-production`**" to "**`php.ini`**".
4. Open php.ini in a text editor and make the following changes to it (use Ctrl-F to find the relevant entries):
    a. Uncomment (remove the initial semi-colon) "**`fastcgi.impersonate = 1`**"
    b. Uncomment "**`fastcgi.logging = 0`**"
    c. Uncomment "**`cgi.fix_pathinfo=1`**"
    d. Uncomment "**`cgi.force_redirect = 1`**" and change the "1" to "0"
    e. Uncomment "**`extension=php_mysqli.dll`**"
    f. Uncomment "**`extension=php_openssl.dll`**"
5. Save and close the file.
6. Open IIS Manager (Start → Control Panel → Administrative Tools → Internet Information Services (IIS) Manager).
7. In the root (your local machine) in the "Features View" (bottom tabs) double-click "Handler Mappings".
8. In the right-hand pane click "Add Module Mapping" and fill in, browse to or select the values as follows:
    a. Request path: *.php
    b. Module: FastCgiModule
    c. Executable (optional): {*location you installed PHP*}\php-cgi.exe
    d. Name: PHP_via_FastCGI
9. Click the "Request Restrictions…" button.
10. Check the "Invoke handler only if request is mapped to:" checkbox.
11. Select "File or Folder" from the radio buttons.
12. Click "OK", "OK" then "Yes".
13. Open a text editor using "Run as Administrator".
14. In it place the text: "**`<?php phpinfo(); ?>`**".
15. Save the file as "**`phpinfo.php`**" in **`C:\Inetpub\wwwroot`**.
16. In a web browser, navigate to http://localhost/phpinfo.php.
17. You should ideally see a page containing a mass of information about your PHP installation.
18. If you do not, but instead get an HTTP Error 500.0 Internal Server Error saying "*The FastCGI process exited unexpectedly*", it may be because you do not have the required Visual C++ redistributable on your machine - to correct this, go to https://www.microsoft.com/en-gb/download/details.aspx?id=48145, click the "Download" button and on popup page select the appropriate redistributable for your machine:
    a. vc_redist.x86.exe for a 32-bit machine
    b. vc_redist.x64.exe for a 64-bit machine

19. Click "Next" to start the download, then execute the downloaded file.
20. Repeat step 16 above to ensure that the fix has worked.

Install the MySQL Database Server

1. From http://dev.mysql.com/downloads/mysql/ select the **Windows (x86, 32-bit), MySQL Installer MSI.**
2. On the next page scroll down and click the first of the two installers (1.6M) "Download" button.
3. On the next page (unless you wish to sign up), simply click the "*No thanks, just start my download*" link.
4. Once downloaded, execute the file and chose the "Run" option, allowing it permissions as required and clicking "Yes" if offered an upgraded version.
5. Accept the licence terms and click "Next".
6. Select the "Server Only" install and click "Next", then "Execute".
7. Click "Next" until the "Accounts and Roles" page of the installer appears - give it a root password (which you will need to remember) and click "Next", "Next" then "Execute".
8. When it completes, click "Finish" then "Next" then "Finish".
9. From the Windows "Start" menu, find and run the MySQL command line client, most probably (assuming the version you installed was 5.7) in Start → All Programs → MySQL → MySQL Server 5.7 → MySQL 5.7 Command Line Client.
10. Supply the password you set in step 7.
11. Enter the command: **`create database wordpress;`**.
12. Enter the command: **`exit.`**
13. Restart the workstation.

(**Note**: on a production server it is advisable to create a specific MySQL database user for WordPress and use their username and password in the following section, however our assumption here is that any workstation install will be for development, rather than deployment, purposes, so to keep things simple we have opted for using the database root user in that role.)

Install WordPress

1. Go to https://wordpress.org/download/ and click the "Download WordPress *x.x.x*" (4.4.2 at time of writing) button.
2. "Run as Administrator" Windows Explorer and open the downloaded zip file.
3. Copy the "wordpress" folder there to "**`C:\Interpub\wwwroot`**".
4. In that "wordpress" folder, rename the file "wp-config-sample.php" to "wp-config.php"
5. Open "wp-config.php" in a text editor run using "Run as Administrator".
6. Find the "**`define('DB_USER'...`**" entry and change "**`database_name_here`**" to "**`wordpress`**".
7. Find the "**`define('DB_USER'...`**" entry and change "**`username_here`**" to "**`root`**".
8. Find the "**`define('DB_PASSWORD'...`**" entry and change "**`password_here`**" to the root password you supplied to MySQL.

9. Find under the comment "**`Authentication Unique Keys and Salts`**" the list of "**`define(...`**" statements followed with "**`'put your unique phrase here'`**" and fill them in (alternatively, to save the effort of typing a lot of random characters, go to https://api.wordpress.org/secret-key/1.1/salt and copy the output, using it to replace that entire block of statements in the file).
10. Save the file and exit the text editor.
11. Open IIS Manager (Start → Control Panel → Administrative Tools → Internet Information Services (IIS) Manager) and expand the local machine, then expand Sites then expand Default Web Site.
12. Select the wordpress virtual directory.
13. In Features View (bottom tabs) double-click Default Document.
14. In the right-hand pane click "Add…".
15. Enter "index.php" and click "OK".
16. In a browser navigate to http://localhost/wordpress/wp-admin/install.php.
17. Set up the WordPress administrator's username and password (you can enter one rather than accepting WordPress's generated one) and click "Finish".
18. Log in with the username and password you entered.

Installing the DataFlex WebApp Server

1. Purchase a WebApp Server licence from your DataFlex supplier.
2. Download the WebApp Server from: ftp://ftp.dataaccess.com/pub/products/dataflex/Software/DataFlex2015-18.1.45.11.Server.exe (or whichever higher version is current at the time).
3. Open the downloaded .exe file and follow the installation instructions.

Installing the DF4WP Plugin in WordPress

1. On your development machine, open your new WordPress site in a browser.
2. Log in - you should be taken to the WordPress Dashboard page.
3. On the left-hand pane, select "Plugins", then "Add New".
4. On the "Add Plugins" page click the "Upload Plugin" button at the top.
5. Click the "Choose File" button and navigate to the directory you unzipped the DF4WP zip file to.
6. Select the **dataflex4wordpress.zip** file (**Note: you do not need to unzip this file** - WordPress expects plugins to be in zip-file format).
7. Click the "Install Now" button.
8. When that completes, click the "Activate plugin" link.
9. A "DataFlex" option should then appear in the left-hand navigation pane (below "Settings").
10. Select that, then enter the virtual directory name where you intend to install your DataFlex Web App in the "Path to DataFlex WebApp" form (**Note**: if the web application is to reside on the same server as WordPress then this need only be set to the local path to it, starting with a "**/**", however if it is to be hosted on a *different* server the entry should be of the form "***//serverNameOrIP/path/to/application***",

however such a deployment will require enabling Cross-Origin Resource Sharing - see that section below).
11. If required, you can select a DataFlex "Theme" to preload.
12. Enter a 32 character "Secret" which will be used to secure communications from WordPress to the web application (if you leave it blank a random 32 character string will be generated for you) - you will need this secret to set the psWPSecret property of your web application.
13. Click "Save Changes".

Modifying your web application, view(s), lookups and images to run under DF4WP

1. Extract the contents of the DF4WP.zip file into a directory on your development machine.
2. From that directory, copy the following files to your project's AppSrc directory:
    - cCryptoAES.pkg
    - cWPWebApp.pkg
    - cWPWebView.pkg
    - cWPWebModalDialog.pkg
    - cWPViewMixin.pkg
    - cWPWebImage
    - cWPWebColumnImage
3. Modify your WebApp.src to **`Use cWPWebApp.pkg`** rather than **`cWebApp.pkg`** and change the class of the **`oWebApp`** object to **`cWPWebApp`**.
4. For any view which is going to appear on a WordPress page (whether embedded stand-alone on a page or as part of an embedded web-app), change the **`Use cWebView`** statement to **`Use cWPWebView`** and change the class of the actual view object to be **`cWPWebView`** (for example **`oMyView is a cWebView`** would become **`oMyView is a cWPWebView`**).
5. If you intend to use the WordPress login mechanism rather than the DataFlex framework's login mechanism (see property **`pbUseWPLogin`** below), any look-up list modal dialogs which are going to be used by that view need to be modified to **`Use cWPWebModalDialog.pkg`** rather than **`cWebModalDialog.pkg`** and the class of the object changed to **`cWPWebModalDialog`**.
6. If you are using images in your application, you should change any **`Use`** statements to refer to **`cWPWebImage.pkg`** and/or **`cWPWebColumnImage.pkg`**, changing the class of the image objects appropriately.
7. Compile your application. Test that it works as normal.

Deploying your web application to the server

Upload your application from your development machine to somewhere on your WordPress server (you should only need to upload the AppHTML, Data and Programs sub-directories, plus any special directories your application uses).

Using the DataFlex Web Application Server Administrator, create your application as a new Web App (File → Configure New Web Application) and provide the configuration details.

Check that it works as normal, both on the server and remotely.

Embedding Applications and Views in WordPress

The procedure for embedding entire applications or individual views in WordPress pages is essentially the same for both approaches, with the major difference being that to embed a view, rather than an application, you specify the view to embed. If no view is specified then the entire application will be embedded.

Embedding an Application

1. On your WordPress site, log in and from the WordPress Dashboard, in the left-hand pane, select "Pages", then "Add New".
2. Enter a title for the new page.
3. In the content editor, add the "**shortcode**" text: **`[df4wp-webapp height=`*`999`*`]`** (including the square brackets). The "height" entry determines the number of pixels (*999*) which will be set aside for the application on the page - it is optional, but if not set, the default is a rather small 120px.
4. Optionally, within those square brackets, separated by a space you can add **`params="`*`param=value|param=value|param=value`*`"`**, where "*param*" can be any Web Property of the outer WebApp object (usually oWebpp), with the proviso that boolean values need to use 1 or 0 rather than True or False, so you might have: **`[df4wp-webapp app=TRUE height="610" params="psTextColor=Blue|psTheme=DF_Windows_Like"]`**
5. Another optional setting which can be made in the shortcode is "**apppath**", which allows you to specify the virtual directory to load the application from (if not specified, the application will be loaded from the default virtual directory you entered on the plugin's settings page).
6. If the application includes DataFlex Reports elements then the setting within the shortcode: **`dfr=TRUE`** should be used.
7. Once all that is done, you can use the "Publish" button to save your changes, then the "View Page" link to be taken to the page to see if it then works correctly. On the page you can subsequently click the "Edit" link to return to editing, where "Update" will then save your changes.
8. In order to make your page available to others, they will need a means to navigate to it; do this in the WordPress Dashboard by selecting "Appearance" → Menus from the left-hand pane.
9. In Menus, give your menu a name ("Main" or "Main Menu" are obvious choices) then click "Create Menu".
10. Switch to the "Manage Locations" tab and in the "Primary Menu" setting, select the menu you just created and click "Save Changes".

11. You can then choose which pages should be added to that menu using the checkboxes which will appear beside them and the "Add to Menu" button, followed by clicking the "Save Menu" button.
12. Next, click the "Manage Locations" tab at the top of the Menus page.
13. For "Primary Menu" use the combo box to select the menu you just created and click "Save Changes" (if you change WordPress themes, you may need to do these two steps again).

Embedding a view

Embedding a stand-alone view follows essentially the same procedure as for embedding an entire application, including the optional "**apppath**" (virtual directory) and "**params**" settings, although in this case entries in the "params" setting will refer to web properties of the view, rather than those of the web app. The difference is that within the shortcode you specify **view="oYourView"**, where **oYourView** is the actual object name of the view in your DataFlex code: **Object *oYourView* is a cWPWebView**.

In addition to the standard properties of a cWebView, the class cWPWebView defines an additional two web properties designed to control the appearance of embedded views on a WordPress page (intended to be set in the WordPress shortcode **params**):

- psTheme (String, default: "") allows a DataFlex CSS theme to be specified for the view
- pbShowToolBars (Boolean, default: True) determines whether the program's command bars (the Find and File toolbars) should be displayed above the view (in the shortcode, set the property using 1 for True, 0 for False)

As with the embedded application, you will usually want to put the page for your view on a WordPress menu to make it accessible to other users of the site.

List of Plugin Shortcode Settings

Within the **[df4wp_webapp** … **]** shortcode there are, as mentioned above, a number of optional settings which can be made. Each takes the form *setting*=*value* (value is sometimes in quotes - which kind does not matter).

view	Embeds a view; the value should be the view object name. If a view is not specified, the entire application will be embedded. The view name should be quoted.
apppath	The virtual directory (or path to it on another server - see step 10 in installing the plugin above) to load the web app from. If not present, the plugin's global setting will be used. The entry should be quoted.

params	A list of settings for web properties of the web app or view, of the form *property=value*, separated by pipe (|) characters. The entire list should be quoted.
height	An integer number of pixels to set the vertical space the app or view will have available - default: 120.
dfr	Set to TRUE (case insensitive and in this case not quoted) to include the DataFlex Reports Previewer components.
theme	Set to preload a theme. This does not set the theme for the embedded app or view, it simply loads it in advance of the app or view needing it, saving time. Theme name should be quoted.

Note

The plugin DataFlex code expects that your program will have a Command Bar which is referenced by the WebApp's **phoCommandBar** property, and that the Command Bar object will have a child object with the name **oMenu**. If this is not the case, some aspects of displaying these (or more accurately not displaying them, when not required) will probably not function correctly.

Making it look nice

At time of writing, WordPress initially installs with a choice of three themes, which control the layout of items on pages: "Twenty Sixteen" (the current default), "Twenty Fifteen" and "Twenty Fourteen".

Both Twenty Fifteen and Twenty Fourteen have wide side-bar menus, which restricts the screen width available on any page for your WebApp or views, so we recommend that, to start with at least, you use the Twenty Sixteen theme, although the older Twenty Thirteen theme, which has no side-bars at all, is also an option if it is available.

To do that (if Twenty Sixteen is not already the site's theme):

1. Go to the WordPress Dashboard.
2. In the left-hand pane, select "Appearance" → Themes.
3. Select theme "Twenty Sixteen" (or "Twenty Thirteen", if available).
4. Click the "Activate" button.

In the WordPress Twenty Sixteen theme, there are a couple of issues which manifest themselves with the DataFlex DF_Flat_Desktop theme (the pair work well together, visually, having similar color-schemes). Menu item text becomes underlined (in fact have a bottom box-shadow 1 pixel wide) and the date-prompt displays with a full window width. These can be fixed by inserting the following CSS at the bottom of the WebApp's application.css file (found in the AppHTML\CssStyle directory):

```
#OWEBAPP .WebDP{
      max-width: 250px;
}

#OWEBAPP a {
      box-shadow: 0 0 0 0;
}
```

Embedding an application vs. embedding views

Whether to embed one or more views from a WebApp into a WordPress site, or to embed an entire WebApp instead (or even as well as), is an issue which deserves consideration. There are arguments for using either approach and you should choose the one which best suits your requirements.

Embedding DataFlex views on pages in a WordPress site is a good way to add a limited amount of specific functionality to a site. You can have a page for each view, which you can then place in the WordPress menu system which will be familiar to WordPress users and easily understandable for new visitors.

However with this approach, each view will, in effect, become its own "Single Page Application". When navigated away from, it will lose all state, so, for instance, if a user were to be updating their customer records, but was interrupted by the need to put a new order on the system, on returning to the customer view, any data there would have been lost. (The user could, of course, open the Orders view page in a new browser tab or window, working around such an issue.)

Embedding an entire application would solve that state problem, as well as providing very fast and efficient switching between views, but at the price of introducing another menu on top of WordPress's own menu system.

If a significant amount of functionality is required to be embedded in a WordPress site, embedding an entire application may prove a better approach, requiring only a single WordPress page, rather than cluttering the WordPress menus with a large number of "view pages".

Login considerations

The majority of web applications (although not all) will require some form user identification for security and authentication.

In DataFlex web applications, the level of required identification is specified through setting the "**peLoginMode**" property of the web app object to one of: **lmLoginNone**, **lmLoginSupported** or **lmLoginRequired**.

However WordPress has its own user login mechanism and it may well be that adding an extra layer of login on top of that would not be desirable (another user-name and password for users to remember, another hurdle to jump before they can get on with whatever they are there for).

For that reason a new property of the web app: pbUseWPLogin (Boolean - default: False) has been added. If this is set to True, then it is the logged-in state of the user within WordPress which will determine whether they have access to the WebApp (whether an embedded app or embedded views), rather than that of the DataFlex framework. If a login is required and the user is not logged in they will be prompted to do so in order to access the feature.

The boolean function **`InWordPress`** of the cWPWebApp class can be called to determine if the application is running within WordPress.

To support this, a collection of information from WordPress is available in the DataFlex program through calls to the **`WPInfo`** function of the web app object. **`Get WPInfo of ghoWebApp "itemname" to var`** will return the desired value.

- sWPTimeSent (Number - the number of seconds since the start of the UNIX epoch, to four decimal places on Windows; used to ensure uniqueness in the security hash)
- bWPUserLoggedIn (Boolean - True if the user is logged into WordPress)
- sDF4WP_Parameters (String - the entire contents of the "params" setting in the "shortcode" which invoked the web app or view)
- bWPFullApp (Boolean - True if it is an embedded application, False if a view)
- sWPView (String - the object name of the view, if an embedded view)
- sWPSiteAddress (String - the base URL of the WordPress site)
- sWPLoginURL (String - the URL to use to login to the WordPress site)
- sWPHomeURL (String - the URL of the site's home page)

If the user is logged into WordPress, then the following items will contain information about them:

- iWPUserID (Integer - the WordPress internal ID of the user)
- iWPUserLevel (Integer - the WordPress "level" of the user - 10 is Administrator)
- sWPDispName (String - the display name of the user)
- sWPFirstName (String - the first name of the user)
- sWPLastName (String - the last name of the user)
- sWPUserLogin (String - the login of the user)
- sWPUserEMail (String - the e-mail address of the user)
- sWPUserURL (String - the user's web site, if one has been supplied)

In addition, this mechanism can be extended. If there is something in extra you would like WordPress to pass to your program, and you are familiar with WordPress's long list of functions, you can use WordPress's plugin editor to add it. From the WordPress administrator dashboard, on the left-menu, select "Plugins" then "Editor". In the top-right

combo-box select the DataFlex4WordPress plugin (click "Select"), then "DataFlex4WordPress/df4wp-page.php" from the list on the right of the page. The PHP code will be displayed in the editor. Scroll down until you reach the series of "**$info .= …**" lines and to the end add a new one of the same form:

```
$info .= chr(31) . 'yourItemName=' . someWordPressFunction();
```

Take care with the syntax - PHP is unforgiving of errors and the entire site may stop working if you make an error such as leaving off the terminating semicolon.

The result of the WordPress function (or it could be a variable) will then be available in your DataFlex program through calling:

```
Get WPInfo of ghoWebApp "yourItemName" to var
```

Cross-Origin Resource Sharing (CORS)

Embedding a DataFlex web application which is running on the same server as the WordPress site is relatively straightforward, however there may be many cases where this is, for one reason or another, not desirable, or even possible. The most common reason that it will not be possible is that the great majority of WordPress sites run on Linux (or Unix) servers, which DataFlex does not.

In such cases, the DataFlex web application will have to be hosted on a server other than the one WordPress is running on.

The DataFlex web app framework makes use of AJAX technology for the JavaScript application in the browser to communicate with the DataFlex application running on the server. This in turn uses the browser's - now thankfully standardised - XmlHttpRequest object (XHR for short). By deliberate design (although for reasons which can now be seen to be mistaken) the XHR is generally prohibited from making calls to a "host" (server) other than the one the page was originally loaded from (in this case, the WordPress server). In order to overcome this restriction a standard called Cross-Origin Resource Sharing (CORS) has been developed to allow such calls to be made. In order to use it you will need to make changes within Internet Information Services (IIS) on **the server your DataFlex application is hosted on**.

Using a text editor create a file containing the following XML, saving it as "**web.config**" in the AppHTML directory of your DataFlex web application. (If a web.config file already exists there, add the following to it, making sure that if the section(s) involved already exist you add the custom headers to them correctly.) The "***remoteServer***" entry should be replaced with the DNS hostname or IP address of the site you wish to embed the web app in (i.e. the WordPress server). If the connection is to be over HTTPS, that should replace "http" in the "**Access-Control-Allow-Origin**" entry.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Access-Control-Allow-Origin"
                          value="http://remoteServer" />
        <add name="Access-Control-Allow-Headers"
                          value="content-type" />
        <add name="Access-Control-Allow-Methods"
                          value="GET, POST" />
        <add name="Access-Control-Allow-Credentials"
                          value="true" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

The above has been formatted for readability and contains line-breaks which your XML ideally should not, so if you wish to copy-and-paste the code, use the unformatted version below:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Access-Control-Allow-Origin"
value="http://remoteServer" />
        <add name="Access-Control-Allow-Headers"
value="content-type" />
        <add name="Access-Control-Allow-Methods" value="GET, POST"
/>
        <add name="Access-Control-Allow-Credentials" value="true" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

Then:

1. From Start → Control Panel → Administrative Tools, launch Internet Information Services (IIS) Manager
2. In that, navigate to the virtual directory for your DataFlex web application
3. In "Features View" (bottom tabs) double-click on the "Handler Mappings" item

4. In right-hand pane click on "View Ordered List"
5. In the list, find the "OPTIONSVerbHandler" entry and select it (it is probably close to the bottom)
6. In the right-hand pane, click "Move Up" (click "Yes" when prompted with the warning about the list order being changed)
7. Keep clicking "Move Up" until it is at the top of the list

Note: As stated above, in the line "**`<add name="Access-Control-Allow-Origin" value="http://`*`remoteServer`*`" />`**" the "***`remoteServer`***" text should be replaced with the DNS hostname or IP address of the WordPress server, so if the WordPress server was running on a machine with a DNS name of "**`www.superblogsite.com`**" you would alter that line to read:

```
<add name="Access-Control-Allow-Origin"
                        value="http://www.superblogsite.com" />
```

Once this has all been done, you should be able to access your DataFlex application from WordPress pages.