IMAGE\_HANDLER

You may run into a project where you need to move an existing CRM into a new code base and need a way to archive the old site and keep track of requests related to those pages.

If you’re lucky, all the resources (images, videos, etc) are in a consistent location so you can use Web.Config or .htaccess to set up a route to handle that endpoint or handful of endpoints.

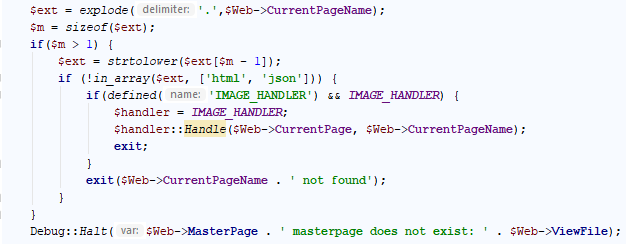
If you’re not so lucky, the old CRM will reference a complex folder structure that your new CRM doesn’t support and one that would be unprofitable to replicate in QuickDRY.

This is where IMAGE\_HANDLER comes in.

The default behavior of Web.Config and .htaccess is that anything that is not a file or a directory gets routed through index.php

And anything routed that is not explicitly defined in a php file under the pages folder with a masterpage results in an error

“masterpage does not exist”



This has been extended to first check if there is an extension on the requested url. If the extension is html or json it is assumed that a controller is expected and so the missing masterpage error will still be thrown.

However, if there is an extension that isn’t html or json, then it will check to see if IMAGE\_HANDLER is defined. If not, the browser will simply report that the file is missing. This avoids endless emails in a production environment from Halt alerts.

If you’d like to receive emails if a file is missing, you can certainly set that up in the IMAGE\_HANDLER

Since IMAGE\_HANDLER is not really different based on environment, the best place to define it is probably /settings.php



Note that this is the name of the class. It must contain the static function Handler that takes two parameters: the full url path of the page (excluding the host), and the name of the page.

You can put the ImageHandler class anywhere, but I put it into the QuickDRYInstance folder since it is an extension of the QuickDRY framework.



This implementation of an ImageHandler looks in the database for an entry with a matching URL to the one the framework is looking for. If it doesn’t find it, I still don’t want an email so I use “exit” instead of Halt to let the user know the file is missing.

If the entry in the database is found, the data is uncompressed and the ExitFile method of the HTTP class provided with QuickDRY makes it easy to have the browser show the file to the user rather than have it download automatically.

Notice that we are using base64\_decode on the data before uncompressing this. This is a preference of storing text data in a database. You can of course use binary data or whatever encoding you want. You could also load the file from the file system.

What this is for is to create routing for a mishmash of files from a CRM that was not well structured without a lot of work.

From here you can start to manage the transfer of your pages into the new database and filesystem structure.