Using Compendium-TA to generate News for a website and rss feeds

Alan Richardson <alan@compendiumdev.co.uk>

2004-04-05

Foreword

This article will describe the use of Compendium-TA and the internal scripts and processes we use at Compendium Developments for creating all the news information at compendiumdev.co.uk [http://www.compendiumdev.co.uk] .

The news at Compendium Developments is divided into a number of categories. The most recent items from the WebWatch category are shown on the front page, all other items are stored in category archives and an rss feed is generated for all the most recent items across all categories.

This document is supported by the entity definition files, used, and all mentioned scripts. You should feel free to customise any of these for your own use, or use the supplied files directly.

Table of Contents

Supporting Files for this Case Study 1
General Details 2
- Compendium Developments News Sections 2
- Design of the process 3
- Entity Definition Stage 5
- write news items 6
- Output files with a script 7
- Synchronize Directories 9
- PHP Scripts 10
- Alternative Designs and possible future amendments 11

Technical Details 12
- VB Script Macro 12

Supporting Files for this Case Study

All supporting files for this case study are available on the compendiumdev.co.uk website case studies page [http://www.compendiumdev.co.uk/compendium-ta/casestudies]

There is a zip archive containing the following directory structure which should be unzipped into your Compendium-TA install folder. A self extracting zip is available for download which will do this automatically.

- \entities\

    - newsEntry.txt - the newsEntry entity definition for importing into a model.

- \scripts\

    - webnews.txt - the VB script associated with the newsEntry entity and documented here, edit the file to customise it to your needs

- scriptDefns

    - webnewsDefns.txt - the script definitions file to associate webnews.txt with the newsEntry in Compendium-TA
    - \documents\ - documentation for the script

        - \webNewsDefns\ - documentation associated with this script

            - newsEntryExample.mdb - an example database using the script and entity
            - archivelist.php - a version of the php script used to display the archive list
            - category.php - a version of the php script used to display the categories
            - webAndRss.pdf - this document

The intention of this document is to provide you with information on some of the capabilities of Compendium-TA, and to provide you with a usable set of extensions to Compendium-TA and to document their use and customisation.

Feel free to use this case study and the scripts and entities associated with it either in their current form or customised for your own requirements.

General Details

Compendium Developments News Sections

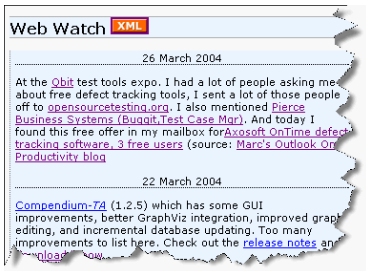
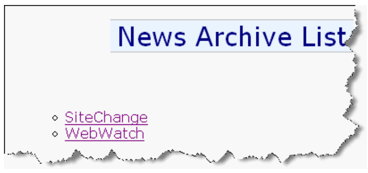
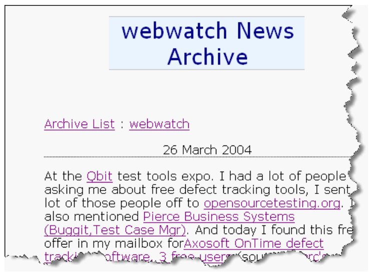
compendiumdev.co.uk [http://www.compendiumdev.co.uk] has a 'WebWatch' section on the front page. Which we use to draw visitors attention to interesting items that we have found on the web. Over time this has become used as a general news section as well as just web information. We needed a way of making sure that the WebWatch section didn't deviate too far from the original purpose of being up to date web pointers, but we still wanted to be able to have news items.

After looking around for a variety of solutions: forums, blogs and wikis, we decided that we really just needed a fairly simple solution with a small subset of functionality. Basically just having as many categories as we wanted, being able to display subsets of those categories for the front page and various other index pages, and to be able to generate an rss feed for changes.

And since most of our updating is done on the desktop, we didn't need a lot of online security and updating.

So a combination of Compendium-TA, PHP and a dash of VB Script seemed like a way of prototyping a solution. And it turned out that the prototype became the system that we stuck with.

Table 1. News on the Compendium Developments Web Site

WebWatch on front page	News Sections Index	category.php output
 Web Watch XML 26 March 2004 At the Qbit test tools expo. I had a lot of people asking me about free defect tracking tools, I sent a lot of those people off to opensourcetesting.org. I also mentioned Pierce Business Systems (Buggit,Test Case Mgr). And today I found this free offer in my mailbox forAxosoft OnTime defect tracking software, 3 free users (source: Marc's Outlook On Productivity blog 22 March 2004 Compendium-TA (1.2.5) which has some GUI improvements, better GraphViz integration, improved grap editing, and incremental database updating. Too many improvements to list here. Check out the release notes an but very easy to install and get working. Archives :News : Web Watch	 News Archive List ◦ SiteChange ◦ WebWatch	 webwatch News Archive Archive List : webwatch 26 March 2004 At the Qbit test tools expo. I had a lot of people asking me about free defect tracking tools, I sent lot of those people off to opensourcetesting.org. also mentioned Pierce Business Systems (Buggit,Test Case Mgr). And today I found this fre offer in my mailbox forAxosoft OnTime defect

In addition to the WebWatch section on the front page, we have a list of sections [http://www.compendiumdev.co.uk/news/archivelist.php]

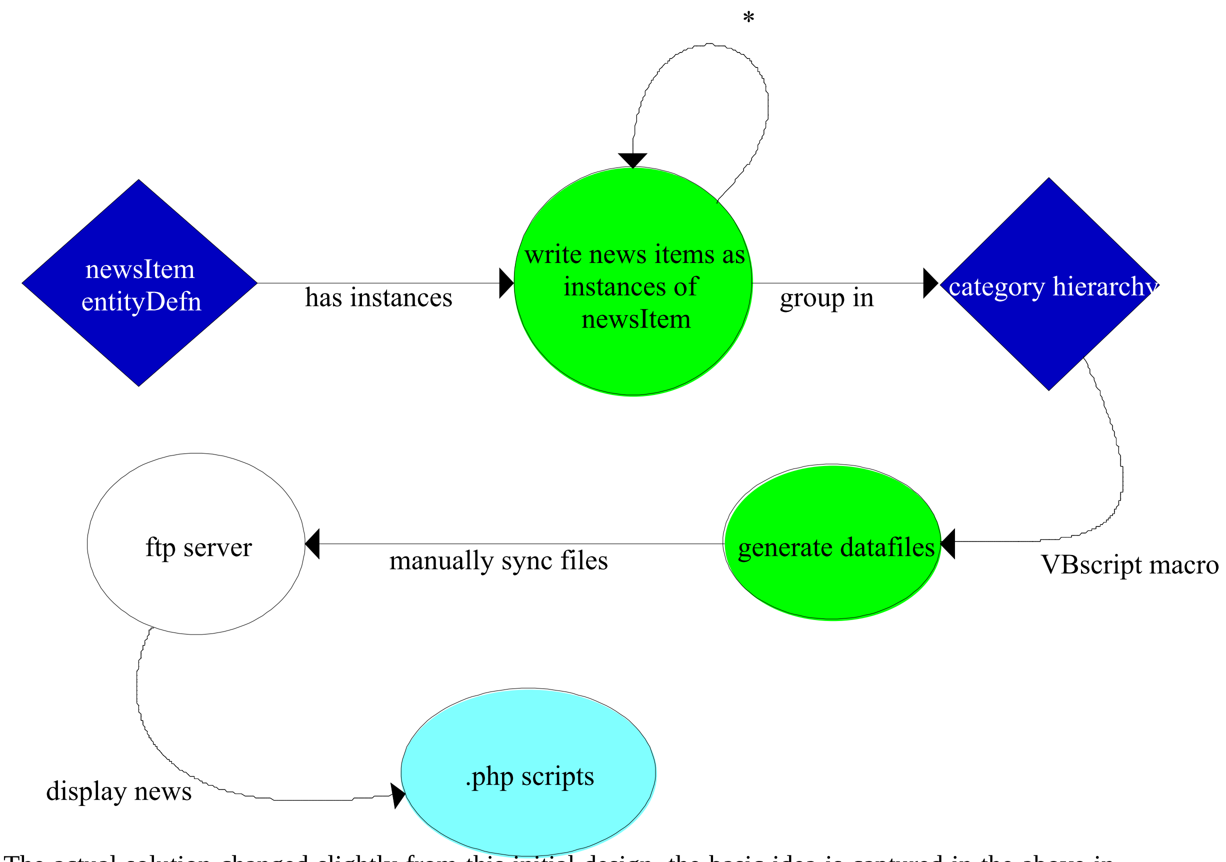
And each section has a number of pages listing the news items in that section on separate pages controlled by the category.php script.

Design of the process

The basic design for the process was that we would:

- create a Compendium-TA entity called newsItem to represent our news items,
- write all our news items in Compendium-TA as instances of entityDefn newsItem
- use a Hierarchy for grouping the items into categories,
- have a simple script associated with the hierarchy which would kick out all the required data files into a directory,
- synchronise this directory with our web server,
- have a number of PHP scripts on the web server that would display these data files

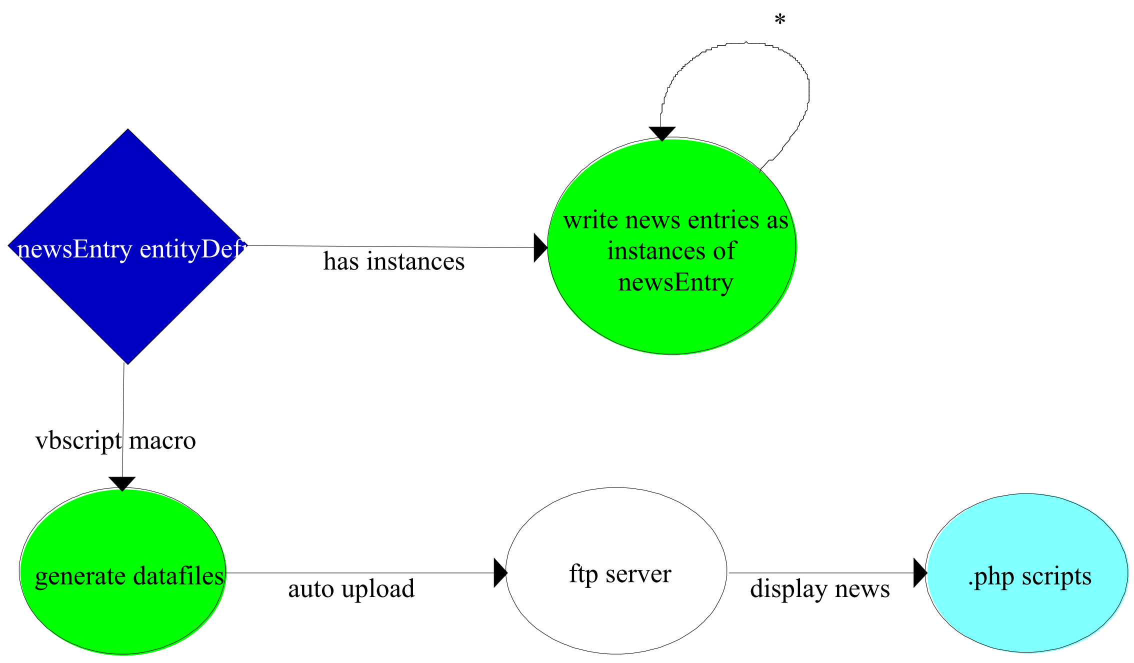
Figure 1. Initial Process Design



The actual solution changed slightly from this initial design, the basic idea is captured in the above information and as this document progresses, I'll explain why we changed our design to the current solution below:

- create a Compendium-TA entity called newsEntry to represent our entries,
- write all our news items in Compendium-TA as instances of entityDefn newsItem, with the category the entry is in defined by the value of the `type` attribute
- have a simple script associated with the hierarchy which generates the data files into a directory, and automatically uploads them via FTP to the web server
- have a number of PHP scripts on the web server that will display these data files

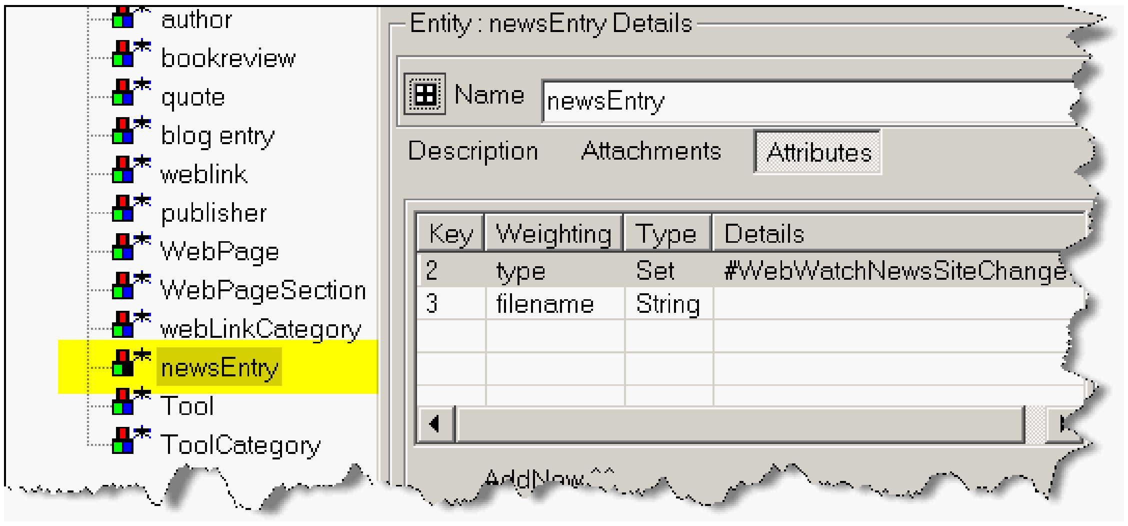
Figure 2. Current Process Design



Entity Definition Stage

The first thing we do is to create an entity definition. I called it `newsEntry` and added two attributes: `type` & `filename`

Figure 3. Definition of the `newsEntry`



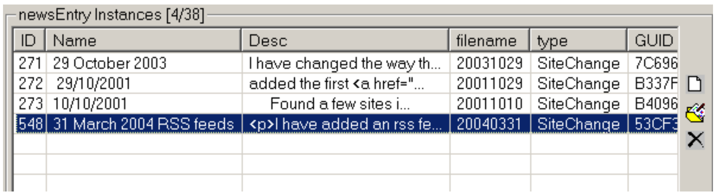
Name: newsEntry

Key	Weighting	Type	Details
2	type	Set	#WebWatchNewsSiteChange
3	filename	String	

`type`	a set with the values {#WebWatch, News, SiteChange}. WebWatch was made the default value by prefixing it with a #. This was going to be controlled by the position in a hierarchy, but by using a *Set* attribute it is easier to prototype the process and write the scripts. And with the new filtering functionality it is easy to

restrict the list to show the required entries. Adding new categories is a simple as adding a new value to the set.

Figure 4. Filtered Instance list to show SiteChange items only



newsEntry Instances [4/38]

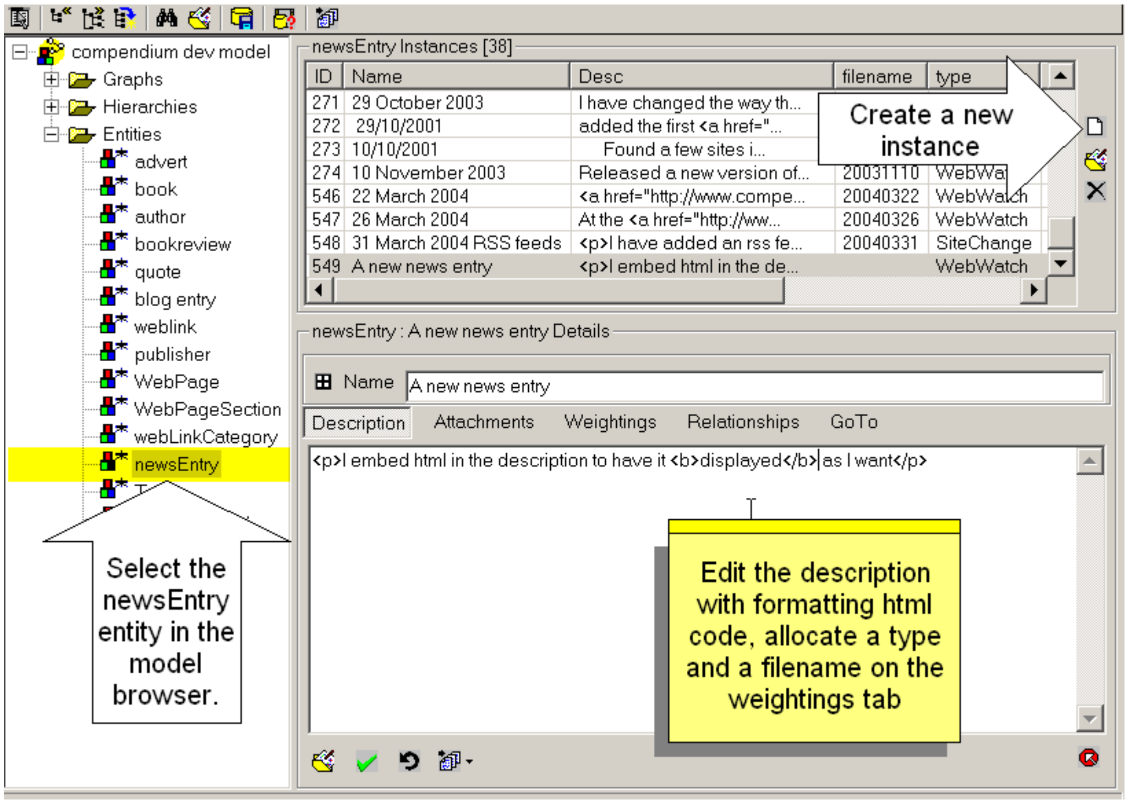
ID	Name	Desc	filename	type	GUID
271	29 October 2003	I have changed the way th...	20031029	SiteChange	7C696
272	29/10/2001	added the first <a href="...	20011029	SiteChange	B337F
273	10/10/2001	Found a few sites i...	20011010	SiteChange	B4096
548	31 March 2004 RSS feeds	<p>I have added an rss fe...	20040331	SiteChange	53CF3

`filename` is a string to be used as the unique filename to store the item in. We could have chosen to use the GUID as that is always unique and would have required no maintenance from the user, but, just as a fairly arbitrary design choice to make the filenames more readable and any subsequent URLs more typeable I chose to manually enter a date format as the filename using the format *YYYYMMDD* e.g. *20040326* representing the 26th of March 2004. The actual filename written to the drive will have the *type* value prefixed and *.dat* prepended giving an actual filename for a WebWatch entry as `webwatch20040326.dat`

write news items

All news items are input to the system as instances of `newsEntry`, so it is easy to create entries

Figure 5. Create an instance of `newsEntry`



newsEntry Instances [38]

ID	Name	Desc	filename	type
271	29 October 2003	I have changed the way th...		
272	29/10/2001	added the first <a href="...		
273	10/10/2001	Found a few sites i...		
274	10 November 2003	Released a new version of...	20031110	WebWa
546	22 March 2004	<a href="http://www.compe...	20040322	WebWatch
547	26 March 2004	At the <a href="http://ww...	20040326	WebWatch
548	31 March 2004 RSS feeds	<p>I have added an rss fe...	20040331	SiteChange
549	A new news entry	<p>I embed html in the de...		WebWatch

newsEntry : A new news entry Details

Name: A new news entry

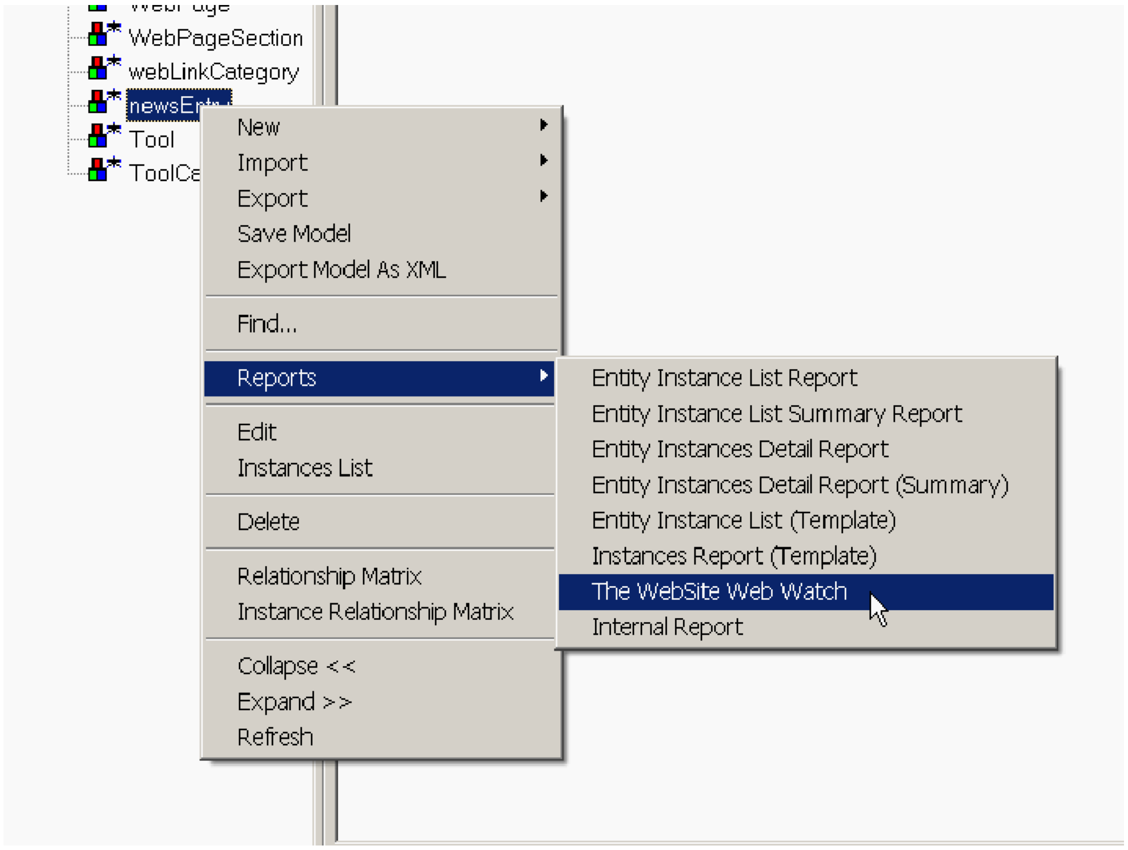
```
<p>I embed html in the description to have it <b>displayed</b> as I want</p>
```

Output files with a script

A simple script was written in VB Script to output all the files required to generate the news. The basic script details are described here and in a later section we will describe the script fully.

The script will be defined as being applicable to the entity definition so when we right click on the entity definition we will choose the macro from the popup context menu.

Figure 6. Context popup menu

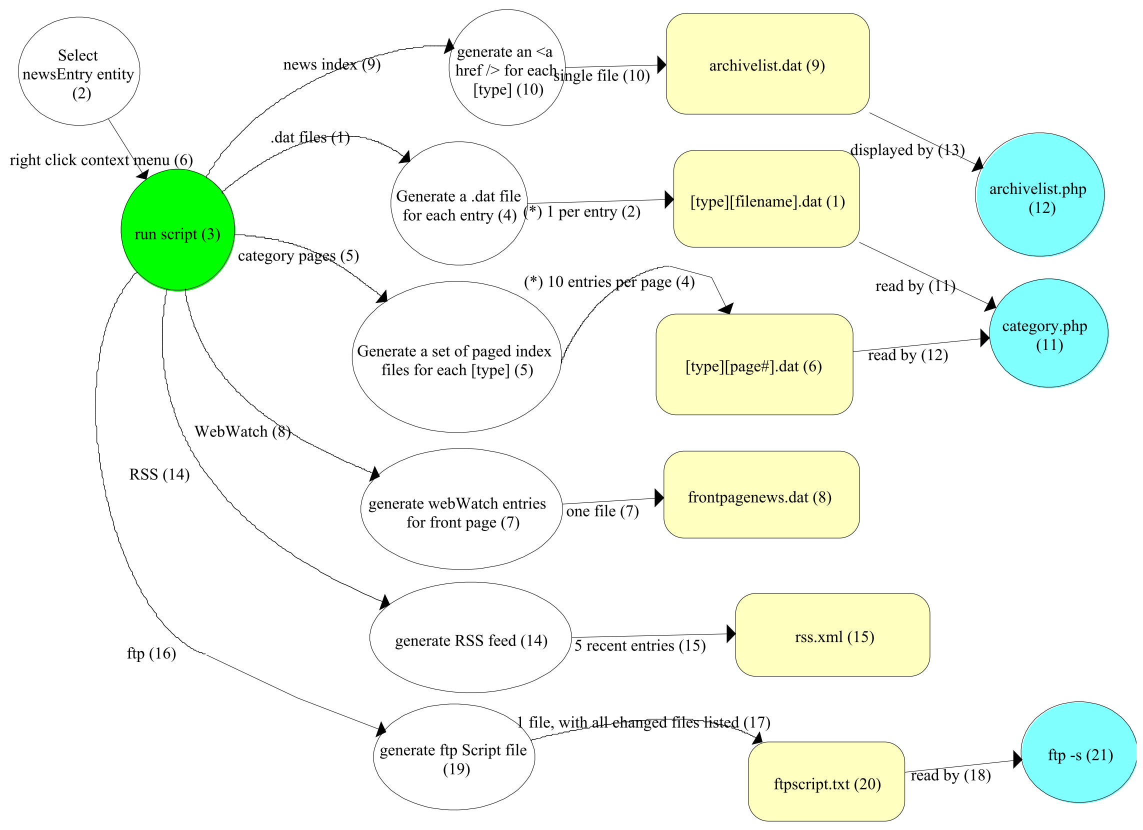


The script generates a number of files:

- 1 file for each `newsEntry`, with the filename format *[type][filename].dat*, only those entries which are new, or have been amended since the .dat file was last written, are output. These files are read by **category.php** when called with the following syntax **category.php?pageid=[type][filename]** *e.g.* ***category.php?pageid=webwatch20040312***
- `archivelist.dat`, a single file which contains <a> tags for each category of the form **category.php?catid=[type]?pageid=1** which will display the first page of entries in the category
- a set of files for each category of the form `[type][page#].dat`, where *[type]* is the category name and *[page#]* is a number starting at 1 and increasing by 1 every 10 entries. The entries are listed in descending date order so that the most recent entry in any category is listed on page 1 e.g. `webwatch1.dat`
- `frontpagenews.dat` is a single file containing the 10 most recent entries in the webwatch category which is displayed on the front page of the compendiumdev.co.uk website
- `rss.xml` is a single file containing an rss 2.0 feed of the 5 most recent entries across all categories.
- `ftpScript.txt` is written with all the FTP commands required to upload the changed files to the remote directory.

The script runs silently as the location of the files is hard coded into the script.

Figure 7. Architecture diagram for the output script



Synchronize Directories

When the script has finished, all the files are stored in a single data directory.

The initial requirement for the design of the process was to manually sync the directory across to the FTP server using ftpVoyager [http://www.rhinosoft.com] or FileZilla [http://filezilla.sourceforge.net/].

The script has been amended since it was first created and now automatically generates an FTP command file which is suitable for passing to the Microsoft FTP command utility [http://www.microsoft.com/WINDOWSXP/home/using/productdoc/en/ftp.asp].

The Microsoft FTP command is run automatically and passed the script as an argument.

Other FTP command line utilities are available if your version of windows does not come with the FTP command built in, e.g. The Kermit FTP Client [http://www.columbia.edu/kermit/ftpclient.html], or ncFTP [http://www.columbia.edu/kermit/ftpclient.html]

Figure 8. Example FTP Script generated

```
open ftp.compendiumdev.co.uk
[username]
[password]
cd public_html/news/dat
put "E:\news\dat\frontpagenews.dat"
put "E:\news\dat\archivelist.dat"
put "E:\news\dat\rss.xml"
put "E:\news\dat\sitechange1.dat"
put "E:\news\dat\webwatch1.dat"
put "E:\news\dat\webwatch2.dat"
put "E:\news\dat\webwatch3.dat"
put "E:\news\dat\webwatch4.dat"
quit
```

`[username] and [password] have been omitted for security reasons.`

A useful reference to the Microsoft FTP command line is at nsftools.com [http://www.nsftools.com/tips/MSFTP.htm]

Sometimes I leave out the **quit** command so that I can review the output of the script and ensure that the FTP transfer has completed successfully.

PHP Scripts

The compendiumdev.co.uk website is PHP driven. There are 3 main scripts that use the data files generated by the VB Script macro. There is nothing at all complicated about these files as all the work is done by Compendium-TA when generating the data files. These PHP scripts simply load and display the generated data files.

`default.php`

This is the script used on the main page at compendiumdev.co.uk and all this does is display the `frontpagenews.dat` file.

```
...
function displayFile($filename){
    $outputFromPHP = fopen($filename,'r');
    while ($line = fgets($outputFromPHP, 1024)){
        print($line);
    }
    fclose($outputFromPHP);
}
...
displayFile("frontpagenews.dat");
```

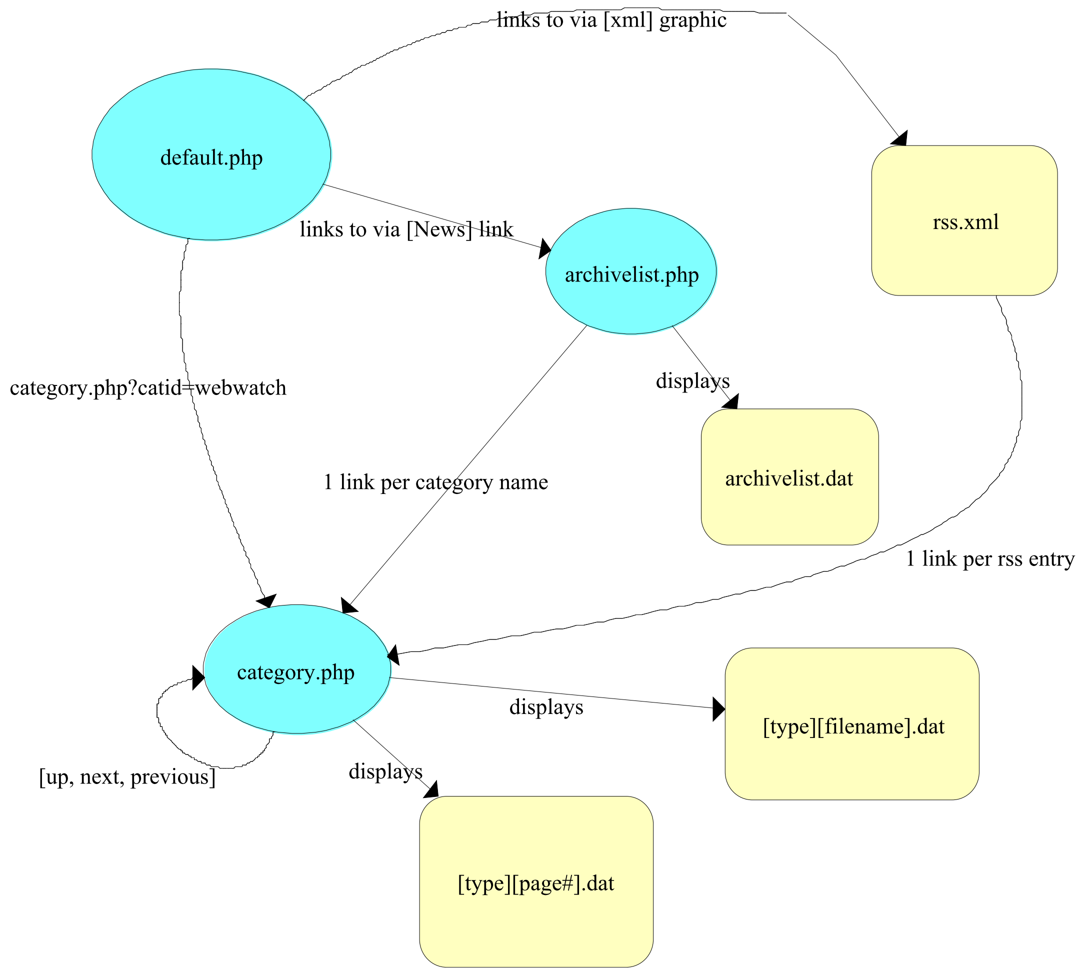
`category.php`

This file is used in two ways, either to display an index file e.g. `webwatch1.dat` or to display an entry e.g. `webwatch20040324.dat`. It is actually very simple and uses a displayFile function to display the external file, it knows which file to use by concatenating the two arguments supplied **catid** and **pageid** e.g. **category.php?catid=webwatch&pageid=1** would display `webwatch1.dat` and **category.php?pageid=webwatch20040324** would display `webwatch20040324.dat`

`archivelist.php`

This is a very simple PHP script that loads in and displays the `archivelist.dat` file

Figure 9. Website Scripts Architecture Summary



Alternative Designs and possible future amendments

The decision to control the categories by using a Set attribute rather than a position in a hierarchy has a number of effects.

- It made the macro easier to write as I just run through the entity instances instead of traversing the hierarchy
- It means that each `newsEntry` instance can only be in one category

At a future date I may wish to add the entries into multiple categories as the rss specification allows this. And when I do that I have the option of using the original hierarchy design because the same entity Instance can be in multiple hierarchy branches. Or I may choose to make an `entityDefn` for the category and xref this to the `newsEntry` instances.

Using the xrefs would mean that the VBScript would change less structurally as we would still be traversing the instances.

Another option for the future is to use the `auditdetails` on the `newsEntry` to define the filename. Specifically the createdDate as this should not be subject to change by the system. This would have the advantage of less manual work by the user when creating `newsEntry` instances.

Technical Details

You can view all the script files in the accompanying archive for this document available on the Compendium-TA Case Studies [http://www.compendiumdev.co.uk/compendium-ta/casestudies] page

VB Script Macro

Compendium-TA supports the Microsoft Windows Scripting Host in order to provide extensibility to the user.

Script Definition

This file is tab delimited and stored as `webnewsDefns.txt` in the `scriptDefns` directory of the Compendium-TA installation path.

```
Report Entity.newsEntry The WebSite Web Watch webNews.txt allWebNews VBScript
```

It is configured to only apply to `Entity.newsEntry` so only appears in the right click context menu of entity definitions named "newsEntry", you can see a screenshot of this earlier in the document.

The report appears named "The WebSite Web Watch" in the context menu and when selected it runs the VBScript contained in `webNews.txt` located in the `scripts` directory of the Compendium-TA installation path. And the function that is run is called "allWebNews"

The tab delimited entries are also documented in the Compendium-TA help system:

Type	At the time of writing, version 1.2.5, only `Report` is supported.
appliesTo	This value controls when the `Display Name` appears in the context menu. In this case, when the user right clicks on an `EntityDefn` called 'newsEntry': `Entity.newsEntry`
Display Name	The text that is displayed in the context menu: `The WebSite Web Watch`
Script File Name	The name of the file stored in the `\scripts\` subdirectory of the Compendium-TA installation directory: `webNews.txt`
Procedure	The subroutine in the `Script File Name` which is called to start the macro, no arguments are ever passed to this subroutine: `allWebNews`
Language	The wsh compatible language that is the script is written in: `VBScript`
libraryFiles	No library files are used by this script.

Script

The actual script is very simple although it might not appear so to beginners, but I have tried to make it as easy to customise as possible so that you can re-use it on your own sites, even if you don't know how to program, by putting many of the important customisable options at the top of the script as `const` statements.

The script has many comments in it when you read the actual VBScript. The code sections here are excerpts to help explain the code and provide an overview so that you find it easier to read and understand the actual code of the script. Not all the code is documented in this document.

The Const Statements

Const statements are constants that are used throughout the script. These are like variables but are read-only when the script is executing

The vbQuote constant is one that I use to make string concatenation easier:

```
 Const vbQuote = """"
```

If you want to customise the script then the following constants are important for general processing:

```
Const maxWebWatchEntries = 10
Const maxRSSEntries = 5
const outputFolder = "E:\My Webs\Compendium\news\dat\"
const itemsPerPage = 10
```

maxWebWatchEntries	the number of items to add to the `frontpagenews.dat` file
maxRSSEntries	the number of items to add to the rss file: `rss.xml`
outputFolder	the full local path determining where all output files are written
itemsPerPage	the number of items to add in the index files for each category

The following constants configure the ftp processing done by the script:

```
const autoFTP = 1
const autoUpload = 1
const ftpScriptPath = "E:\My Webs\Compendium\news\ftpscript.txt"
const ftpUserName = "[type your ftp username here]"
const ftpPassword = "[type your ftp password here]"
const ftpOpen = "ftp.compendiumdev.co.uk"
const ftpDirectory = "public_html/news/dat"
const autoQuit = 0
```

autoFTP	Set this to '1' to enable the generation of an ftp script file
autoUpload	Set this '1' to have the macro automatically run the generated ftp script file and upload the changed files to the ftp server
ftpScriptPath	The full path including filename of where to write the ftp script
ftpUserName	The user name that you use to log in to the ftp server

ftpPassword	The password that you use to log in to the ftp server
ftpOpen	The address of your ftp server
ftpDirectory	the directory that you want the ftp script to upload the files to
autoQuit	Set this to '1' if you want the macro to write a 'quit' statement to the ftp script. If you write a quit statement then the script window will close automatically after the upload. I like to check the results of the upload in the ftp window and manually type 'quit' into the ftp window to close the session.

The Main Script Functions

These functions are the main processing control of the script.

```
Public Sub allWebNews()
Public Sub webNewsReport (reportID)
public function sortByWeightingDesc(aCollection,aWeightingName)
```

allWebNews	This function is the external entry point to the script. Since we enter the script with no parameters I create entry point functions that call a main function and pass any required parameters. In this case the main function `webNewsReport` only performs one function and so is rather redundant but this way of organising the functions gives greater scope for expansion.
webNewsReport	This is the main control function but, as with `allWebNews` only one function is really called and this mechanism is mainly for future expansion. The function sets up the file system access and calls `outputAllWebWatchDataFiles`
sortByWeightingDesc	Given a collection object (`aCollection`) and the name of a weighting (`aWeightingName`) the function will return a `Dictionary` of all the objects in `aCollection` sorted by the value of the objects' in the `aWeightingName`.

News Item File Functions

Functions used to write the News Item files

```
public sub outputAllWebWatchDataFiles(aPath)
public sub outputNewsItemDataFile( aPath, theItem)
public function returnNewsItemOutput(theItem)
```

outputAllWebWatchDataFiles	Although this is called 2 tiers down the call stack, this is the main function in the script. This works through all the entities in the model to find the `news-`

	`Entry` entity, creates the frontpage file, the rss file, and the main index file. The files are created in the supplied (`aPath` folder). It then sorts the entity instances by "*filename*" (which is used to track the date of the entries). For each of the instances, the script writes out the individual news file for the instance (if it has changed or is new), and adds it to the relevant index files and rss file.
outputNewsItemDataFile	Given a filename path (`aPath`), and a `newsEntry` instance (`theItem`) If the `newsEntry` instance is new, or has been amended since the newsEntry Instance data file (*{aPath}\[type][filename].dat*) was last created then we write the file again.
returnNewsItemOutput	Given a `newsEntry` instance (`theItem`) this function returns a string representation of the instance which can be written to the index files and to the *[type][filename].dat* files.

Archive Index File Functions

```
public sub writeThisToTheArchive(  aPath, _
                                   theItemCount, _
                                   categoryName, _
                                   writeThisText)

public sub addReferencesToEndOfArchiveFiles(    aPath, _
                                                theItemCount, _
                                                categoryName)
```

writeThisToTheArchive	Write some preformatted text to a category's archive file. Given a filename path (`aPath`), the current count of items in this category (`theItemCount`), the name of the category to write this information (`categoryName`), and the formatted `news-Entry` instance as a string (`writeThisText`). The function calculates which page to add the item to.
addReferencesToEndOfArchive-Files	Every archive file has a set of links at the bottom of the page to jump to any page in the category, and to jump to the next page. *[1] [2] [3] [next page]* This function adds those links to the archive file defined by the parameters: `aPath`, the file path of the archive file; `theItem-Count`, the current numer of items in the category; `category-Name`, the name of the category.

FTP Functions

Functions relating to the FTP file generation and calling the FTP program. The functions are only executed if the `autoFtp` const is set to '1'.

```
public sub writeFTPFileHeader()
public sub writeFTPPutFile(localFile)
public sub writeFTPFileTrailer()
public sub runFTP()
```

writeFTPFileHeader	A set of string output functions to open the FTP connection and log in.
writeFTPPutFile	Write an FTP *put* command for the given file path (`localFile`) to the FTP script file.
writeFTPFileTrailer	Write the *quit* statement to the FTP file if the `autoQuit` const is set to '1'.
runFTP	If `autoUpload` const is set to '1' then this function runs the *FTP* program with the written FTP script file.

RSS Functions

Each of the rss functions is passed a reference to an open file system `File` object (`aFile`).

These functions write the rss details so that they conform to the rss v2.0 specification at harvard.edu [http://blogs.law.harvard.edu/tech/rss].

```
public sub writeRSSFileHeader(aFile)
public sub writeRSSItem(aFile, anItem)
public sub writeRSSFileFooter(aFile)
```

writeRSSFileHeader	A set of string output functions to create the rss file header. You should edit the details in here to match the email, web and user details that you want to appear in your rss file.
writeRSSItem	Write out the `newsEntry` instance (`anItem`) in a form suitable for rss syndication. This function demonstrates the use of the `auditdetails` object.
writeRSSFileFooter	Close the xml tags in the rss file

3rd Party Utility Functions

Functions written by others found on internet sites, but used by this script:

```
public function formatDate(format, intTimeStamp)
```

formatDate	VBScript does not have an alternative for the VB format$ function, so I am using this one that I found at www.ilovejackdaniels.com [http://www.ilovejackdaniels.com/ASP/VBScript_Date_Format_Functions] to format the dates into the specified rss dat format i.e. *"%D, %d %M %Y %h:%i:%s"* - *"Mon, 05 Apr 2004 13:05:32"*.