G00001

DOCUMENT MANAGEMENT METHOD AND TOOL

The present invention relates to a method of management of documents stored in a computer file system. The present invention also relates to a tool for the management of documents stored in a computer file system.

The storing and retrieving of information by computers is well known to be facilitated through the use of computer file storage systems. See for example 'Operating Systems, Design and Implementation' by Tanenbaum and Woodhull, Prentice Hall 1997. When an application program such as a word processor is used on a computer implementing such a file storage system, the computer can, for example, be used to store and retrieve electronic documents.

As will be further well known the need for users of different computers to utilise a shared computer file storage system so as to be able to share access to electronic documents may be met through the linking of the computers to suitable networks. Shared computer file storage systems may then be used to implement collaborative document stores where a number of collaborating authors may share access to their stored collective works.

The variety of purpose of electronic documents, as with traditional printed documents, has naturally led to a diversity of styles in which these documents appear. Word processing applications range in sophistication from very simple text editors, handling only ASCII (American Standard Code for Information Interchange)

150150

characters, to complex desktop publishing packages, allowing freedom of choice as to the style of presentation of the document.

It has often proved desirable to control the appearance of a document through the use of a so-called 'template'. At the time of creation of a document a template will be chosen and the content of the document, for example text, will then be created within that template. A default template might be provided by the word processing system. If however, in accordance with the preferences or needs of the user, a different template is desired then this will have to be manually chosen. In either case the template features will be specified by instructions within the document to which they are intended to apply and hence will be inextricably linked with the document itself.

In consequence, one problem facing the user of a word processing system is that, if another template is subsequently desired to be applied, then a new document will have to be opened with the new template and then the content from the old document will have to be copied and pasted into the new document. Moreover, if such a change were required for a batch of documents then each document would have to be changed individually.

A related problem facing the manager of a computer file system implementing a collaborative document store is that each of the collaborating authors may have a different word processing system or a different personal preference for the style of documents. In this way, different documents offered for storage may have different appearances, in which case the collaborative document store will lack a consistent

'look and feel'. To implement such a consistent 'look and feel' would, as indicated above, require the manual application of the desired template or templates to each document.

At the present time such a collaborative document store might very well be presented as a 'Web site'. It will be well known that the World Wide Web (WWW or Web hereinafter), has a wide variety of associated concepts and standards. A well known example of a rich source of information relating to these concepts and standards is the *World Wide Web Consortium (http://www.w3c.org), a body responsible for the* development of such standards. The World Wide Web Consortium is hosted by the Laboratory for Computer Science at the Massachusetts Institute of Technology (MIT). Concepts such as a 'Web site', a 'Web page', a 'Web browser' and a 'hyperlink' and standards such as HyperText Markup Language (HTML hereinafter) and eXtensible Markup Language (XML hereinafter) as examples of Standard Generalised Markup Languages (SGMLs) will be well understood.

The problem of the creation of a consistent Web site 'look and feel' has for example been addressed through the development of so-called 'Cascading Style Sheets' (CSS). CSS provide a degree of so-called 'inheritance' of style features to allow for such features to 'cascade' through related documents. As with the present day examples discussed above however, the style features are specified or called by instructions within the document to which they are intended to apply and hence again, they are inextricably linked with the document itself.

Quite apart from the problems of hierarchical control of appearance of pages in a Web site, more general problems of Web site management are known to include the management of content represented in the Web pages, including the archiving of content older than a predetermined number of days, without the consequent possibility of 'broken hyperlinks'.

According to one aspect of the present invention there is provided a method of managing information-bearing content files stored in a computer file system; the computer file system being divided into directories; the method comprising:

locating one or more content files; each content file being stored in a directory of the computer file system;

associating one or more template files with each directory in which at least one content file is stored; each template file being effective, when applied to a content file, to carry out a respective pre-determined operation on the content file; and

applying the or each template file associated with a given directory to each content file stored in that directory.

Advantageously, in this way template files are associated not with particular content files as in the prior art, but with the directory in which the content file is stored. Consequently the template chosen to be applied to a given content file may be selected by associating the chosen template with the directory in which the content file is stored or by moving the content file into a directory with which that chosen template is associated. Should a different template be desired to be applied then, instead of the manual re-editing of the 'templated' file produced by the method which would be necessitated by the prior art, quite simply either the different template is associated with the directory in which the content file is stored, or the content file is

moved to a different directory with which the new template is associated, and in either case the method is re-executed.

It will be appreciated that this method will be even more advantageous when a batch of files is to have a new template applied. Either the new template can be associated with the directory in which the batch of files is stored or the batch of files can be stored in a directory with which the new template is associated. In either case the necessity of the laborious manual re-editing of each and every content file is again avoided.

Preferably the computer file system is divided into a hierarchical arrangement of directories and the one or more templates associated with each directory located in the direct hierarchical path from a directory in which a content file is stored up to and including the uppermost directory in the hierarchical arrangement are also associated with the directory in which the content file is stored.

Advantageously this will allow the hierarchical application of one or more templates to a given content file in such a way as to allow a cascade of template styles. In this way a consistent 'look and feel' can be brought to the presentation of the content files stored in the hierarchical file system. Again, the templates applied to a content file to achieve this advantage are so applied simply by virtue of their association with the directory in which the content file is stored. Changes in the cascading style can be effected, not through laborious manual re-editing of each and every content file, but quite simply through, either the introduction of the association of the desired template or templates into a given path of the hierarchy or the movement of desired files into a directory in an appropriate path of the hierarchy.

Further preferably the association of a template with a directory is made on the basis of the template file being stored in that directory.

Advantageously, not only does this provide a particularly simple and elegant means of associating a template with a directory but it renders further specification of the

association needless. All the association information needed by the invention is provided through the mere choice of location of the template.

Yet further preferably the method further comprises:

associating metadata with each content file; and

carrying out the respective pre-determined operation on each content file upon the application of an associated template file on the basis of the respective associated metadata .

Advantageously this will permit the templates to carry out a wide variety of file management operations on the content files in accordance with the metadata information about the content files. One simple example might relate to the elapsed time (an item of metadata) since the content file was stored in the directory; a template might be effective, when applied, to cause a content file to be archived if the metadata indicated that more than a pre-determined amount of time had elapsed since the content file was stored in the directory.

Apparatus to perform the method is also provided.

Embodiments of the present invention are now described, by way of example, with reference to the accompanying drawings in which:

Figure 1 illustrates a general purpose computer system;

Figure 2 illustrates a first source file structure and associated content and template files stored therein;

Figure 3 illustrates a first process flowchart for use with respect to the first source file structure and associated content and template files stored therein;

Figure 4 illustrates a second source file structure and a separate directory and associated content and template files stored therein;

Figure 5 illustrates a second process flowchart for use with respect to the second source file structure and separate directory and associated content and template files stored therein;

Figure 6 illustrates a third process flowchart for setting up a third source file structure with a separate directory and storing associated content and template files therein;

Figure 7 illustrates the third source file structure and separate directory with associated content and template files stored therein set up in accordance with the preceding process flowchart;

Figure 8 illustrates a target file structure generated from the source file structure and associated content and template files as illustrated in Figure 7.

A first embodiment of a method and tool according to the invention will now be discussed with reference to Figures 1,2 and 3.

Figure 1 illustrates a conventional general purpose computer 100. Such a computer 100 will typically have at least a central processing unit (CPU) (not shown), read-only memory (ROM) (not shown), random-access memory (RAM) (not shown), a storage device such as a hard disk (not shown), a device for reading from and writing to storage media such as a floppy disk drive 102 for reading from and writing to a floppy disk 104, input devices such as keyboard 106 and a mouse 108, a display device such as a monitor 110 and input and output ports (not shown) for connection to other devices or communications networks. The computer 100 is illustrated as

connected to a communications network 112 which in turn allows communication with other computers 114,116,118 similarly connected.

The computer 100 may utilise any suitable operating system, a well known example being Microsoft Windows™. Application programs may be written in any suitable language. Such an operating system and application programs may be loaded onto the storage device (not shown) of the computer 100. An application may, for example, be written in Java™ in which case it is required that a Java Virtual Machine be present on the computer 100. As will be well known the Java Virtual Machine is available for downloading from the Sun Corporation™ Website (http://www.sun.com). There are many well known alternatives of suitable languages in which to write application programs, one example of which is C++.

The tool disclosed in accordance with the first embodiment of the invention may be implemented as a software application program to be executed by the computer 100. The tool implemented as such a software application program may then be stored in any suitable computer readable storage media form, for example on floppy disk 104, for loading into the computer 100, via the floppy disk drive 102, for execution. A well known alternative would be to store the software application on a CD-ROM (not shown) for loading into the computer 100 via a CD-ROM drive (not shown) for execution. A further well known alternative would be to download the software application program over the network 112, for execution by the computer 100.

Before the tool is to be invoked, the file system and files upon which the tool will act must be provided. Having regard to Figure 2, a two-tier hierarchical file structure 200

is indicated, with a first-tier root directory 202 and first, second and third second-tier sub-directories 204,206,208. A first text document 210 has been stored in the first-tier root directory 202 and second, third and fourth text documents 212,214,216 have been stored in the first, second and third second-tier sub-directories 204,206,208 respectively. A first template file 218 has been stored in the first-tier root directory 202. A second template file 220 has been stored in the first second-tier sub-directory 204.

When invoked, the tool will first perform a conventional 'tree traversing' procedure on the hierarchical file structure 200, in a top down fashion. 'Tree traversing' is also known as 'tree creeping' or 'tree crawling'. As will be well known, in the course of this first procedure the branching form of the file structure 200 in terms of the root directory 202 and the sub-directories 204,206,208 will be detected and stored by the tool. Typically the description of the branching form stored by the tool may include information as to the name of a given directory, the relationship and name of the parent directory of the given directory, the relationships and names of any sub-directories of the given directory, the relationships and names of any other directories which share the same parent directory as the given directory and so on.

The tool next performs a tree-traversing procedure in a bottom-up fashion. This second procedure will be discussed having regard to the process flowchart indicated in Figure 3. This process will be executed in respect of each content file stored in the file structure 200.

Step 300 provides for the selection of one particular content file stored in a given sub-directory. Step 302 provides that the tool will first search the sub-directory in which the content file is itself located for a template file. If such a template file is found then according to step 304 the tool will apply that template to the content file and the process for that particular content file will terminate.

Typically such an application of a template to a content file will first involve the opening of a new file including the template. The body of the content file is then automatically copied and pasted into this new file. In this way the template is automatically applied to the body of the content file on the basis of the location in which the content file was stored. The template is, in effect, associated with the (sub) directory in which the content file is located rather than just with the content file itself as in the prior art. The new file may of course be stored anywhere but, by way of example, newly 'templated' files 222,224,226,228 are indicated in Figure 2 (in dashed lines) as being stored in the same (sub) directory as the content file upon which the templated file is based.

Should such a template not however be found in that sub-directory then, in accordance with step 306, so long as that (sub) directory is not the root directory, the tool will next consider the parent (sub) directory of the sub-directory in which the content file is located, in accordance with step 308. Should such a template still not be found upon repeat of step 302 then, in accordance with further repetition of steps 306, 308 and 302, the tool will continue the search up the hierarchical structure until a template is found. That template would then be applied. If no template is found in the root directory then the process for that particular content file will terminate.

It will be appreciated that the process flowchart depicted in Figure 3 indicates an exclusive application of a template. Once a single template has been found, that template is applied. It will further be appreciated however that templates may be applied in conjunction with one another, either complementing one another or overriding one another to the extent provided for. In this case the bottom-up tree traversing would traverse the tree all the way from the lowest sub-directory under consideration up to the root directory to determine the location of all relevant template files such that the appropriate application of all the relevant templates could be performed.

In particular, having regard to Figures 2 and 3, in this alternative once the template 220 in the first second-tier sub-directory 204 was located, then the tree-traversing would not terminate in accordance with step 304, but would record the location of that template 220 and carry on up the hierarchy to record the location of the first template 218 in the root directory 200. The appropriate application of both the first template 218 and the second template 220 could then be performed.

In this way, in this example it will be clearly seen that the templates are 'inherited' down the hierarchy form parent directory to child directory, providing a 'cascading' of template style. Having regard to Figure 2, the first template 218 would be applied to the first, third and fourth text documents 210,214,216. Both, however, the first template 218 and the second template 220 would be applied to the second text document 212.

By way of an example, the first template 218 might define a general meetings template, suitable for use with the documents in the first-tier root directory 202 and the second and third second-tier root directories 206,208, whereas the second template 220 might define a template for particular meetings, for example board meetings, which require, for example, that the general meetings template be complemented or overridden to the extent provided for by the board meetings template.

It will be immediately apparent that this first embodiment according to the invention demonstrates notable advantages over the techniques according to the prior art.

If, for example, a further particular meeting type arose, requiring a third template, and related documents were to be stored in the second second-tier subdirectory 206 then all that would be required to apply the third template to all the relevant documents would be to store the third template in the second second-tier sub directory 206 and re-execute the method through re-running the tool.

By way of another example, if a batch of documents such as the fourth text document 216 stored in the third second-tier sub-directory 208 were to be required to be changed into the templated form suitable for use in board meetings, all that need be done is the moving of all the relevant document files from the third second-tier sub-directory 208 to the, in this example, first second-tier sub-directory 204, followed by re-execution of the method.

A second embodiment according to the invention will now be discussed with reference to Figures 1,4 and 5.

Having regard to Figure 1, the tool disclosed in accordance with the second embodiment of the invention may again be implemented as a software application program to be executed by the computer 100. The tool implemented as such a software application program may then be stored in any suitable computer readable storage media form, for example on floppy disk 104 or on a CD-ROM (not shown), for loading into a computer 100 for execution. Again a well known alternative would be to download the software application program over the network 112, for execution by the computer 100.

Again by way of example a two-tier hierarchical file structure 400 is indicated, with a first-tier root directory 402 and first, second and third second-tier sub-directories 404,406,408. Again a first text document 410 has been placed in the first-tier root directory 402 and second, third and fourth text documents 412,414,416 have been placed in the first, second and third second-tier sub-directories 404,406,408 respectively.

In a separate directory 418, outside the hierarchical file structure 400, a look-up table 420 is stored. First and second template files 422,424 are also stored in the separate directory 418. The look-up table 420 associates (sub) directories with templates. In this example the look-up table might, for example, indicate that the first template file 422 is associated with the first-tier root directory 402 and the second template file 424 is associated with the first second-tier sub-directory 404. It will be appreciated

that a look-up table is only one example of providing a correspondence between a (sub) directory and one or more templates. Other examples such as a linked-list will be well known.

When invoked, as with the first embodiment, the tool will perform a 'tree traversing' procedure, in a top down fashion. Again, in the course of this first procedure the branching form of the file structure 400 in terms of the root directory 402 and the sub-directories 404,406,408 will be detected and stored by the tool. Yet again, typically the description of the branching form stored by the tool may include information as to the name of a given directory, the relationship and name of the parent directory of the given directory, the relationships and names of any sub-directories of the given directory, the relationships and names of any other directories which share the same parent directory as the given directory, the names of any files stored within the given directory and so on.

The tool next performs a tree-traversing procedure in a bottom-up fashion. This second procedure will now be discussed having regard to the process flowchart indicated in Figure 5.

Step 500 provides for the selection of a particular content file stored in a given sub-directory. Upon invocation of the tool, step 502 provides that the tool will first consult the lookup table 420 to find out if there is a template file associated with the sub-directory in which the content file is itself located. If such a template is found then according to step 504 the tool will apply that template to the content file and the process for that particular content file will terminate.

Again, typically, such an application of a template to a content file will involve the opening of a new file including the template. The body of the content file will be automatically copied and pasted into this new file. In this way the template is automatically applied to the body of the content file. The new file may of course be stored anywhere but, again by way of example, newly 'templated' files 426,428,430,432 are indicated in Figure 4 (in dashed lines) as being stored in the same (sub) directory as the content file upon which the templated file is based.

Should such a template not however be found to be associated with that sub-directory then, so long as that (sub) directory is not the root directory in accordance with step 506, then in accordance with step 508 the tool will consult the lookup table 420 to find out if there is a template associated with the parent (sub) directory of the sub-directory in which the content file is located. Should such a template still not be found upon repeat of step 502, then, in accordance with a repetition of steps 506, 508 and 502, the tool will continue the search up the hierarchical structure until such a template is found. That template would then be applied. If no such template is found to be associated with the root directory 402 then the process for that particular content file will terminate.

Again, as with the first embodiment, a non-exclusive application of templates may be considered. In this case, having regard to Figures 4 and 5, in this alternative once the template associated with the first second-tier sub-directory 404 was located, then the tree-traversing would not terminate in accordance with step 504, but would record the association of that template 424 and carry on up the hierarchy to record the

association of the first template 422 in the root directory 402. The appropriate application of both the first template 422 and the second template 424 could then be performed.

In this way it will again be clearly seen that the templates are 'inherited' down the hierarchy form parent directory to child directory, providing a 'cascading' of template style and the associated advantages discussed with respect to the first embodiment.

A third embodiment according to the invention will now be described with reference to Figures 1,3,6,7 and 8.

Having regard to Figure 1, the tool disclosed in accordance with the third embodiment of the invention may again be implemented as a software application program to be executed by the computer 100. The tool as implemented as such a software application program may then again be stored in any suitable computer readable storage media form, for example on a floppy disk 104 or on a CD-ROM (not shown), for loading into a computer 100 for execution. Again a well known alternative would be to download the software application program over the network 112, for execution by the computer 100.

The subject matter of the Applicant's co-pending international patent application no. PCT/GB98/00804 is hereby incorporated by reference.

The method disclosed in the co-pending application is such that simple documents, for example text documents, can be used to derive, for example, HTML documents representing the content of the simple documents and suitable for use in a Web site. The files containing the 'content' desired to be made available in the Web pages are placed in the directories and sub-directories of a file structure. When the method is invoked this file structure is traversed and the content files stored in the file structure are then used to generate Web pages representing the associated 'content'.

The method and tool here described as the third embodiment according to the present invention enhance the functionality of the method as disclosed in the co-pending application. Files containing template information are placed in a separate directory outside the file structure. Yet further files containing template information, however, are placed in the directories and sub-directories of the file structure. When the 'enhanced' tool according to the third embodiment of the present invention (hereinafter referred to as 'the tool') is invoked, the appropriate templates are assembled and applied to the respective content containing files, for example the text documents, to create Web pages 'managed' through the application of these templates.

Figure 6 illustrates a flow chart indicating a process by which such a 'managed' Website may be created using the tool. The respective steps 600 to 612 will be illustrated with further reference to Figures 7. It will be appreciated that since a number of these steps involve the independent setting in place of elements upon which the tool acts when invoked, the order of some of the steps indicated in Figure 6 is interchangeable.

Step 600 provides for the creation of a 'source' file structure upon which the tool may subsequently act. Figure 7 illustrates an example of a source file structure 700 with a three tier hierarchy. A first-tier root directory 702 is created with first, second and third sub-directories 704,706,708 at the second tier level. Created as sub-directories of the third second-tier sub-directory 708, first and second sub-directories 710,712 are provided at the third tier level. It will be well known that this may be done using, for example, Windows File Manager™ or Windows Explorer™ when using a Windows™ operating system.

Step 602 provides for the creation of, in this exemplary embodiment, a set of HTML templates. It will be appreciated however that any other suitable form of template could be used.

As will be well known Web 'pages' may be created through the use of a particular markup language, HyperText Markup Language (HTML). It is through the use of HTML that, for example, the appearance of the text or graphics of a given Web page may be controlled along with the linking of one such Web page to another. The HTML instructions are also known as 'tags'. The format of a Web page may thus be defined, for example, through the insertion of HTML tags in an ASCII text document. A wide variety of simple text editors, word processors and more sophisticated HTML editors are available for this purpose. One example of a suitable application program is Microsoft Word™.

In defining the format of a Web page, instructions may be given, for example, as to the appearance of headers, toolbars and footer sections. The instructions will define templates for these headers, toolbars and footer sections and these templates may then be used, in conjunction if desired, to control the appearance of any one or more Web pages.

In performing Step 602, a separate directory 714 is used to store a set of HTML templates 716 so created.

Step 604 provides for the insertion of user-defined tags into the HTML templates 716.

The standard tags recognised in HTML may be augmented by user-defined tags (user-defined in the sense that they are defined by the computer user creating the tags) for use with a given application. In this case the tags are intended for use with the tool and will be referred to hereinafter as 'tool tags'. These tool tags will be included in the HTML defining a given template, in the form, ' !!TAG_NAME!! '. The double exclamation marks surrounding the TAG_NAME indicate to the tool that a pre-defined association, typically the insertion of text, is to be made when the tool is invoked.

This pre-defined association may be made on the basis of the information gathered about the file structure when the tree traversing is performed. As in respect of the first and second embodiments, typically the description of the branching form stored by the tool will include information as to the name of a given directory, the relationship and name of the parent directory of the given directory, the relationships

and names of any sub-directories of the given directory, the relationships and names of any other directories which share the same parent directory as the given directory, the names of any files stored within the given directory and so on.

A list of examples of such tool tags is provided in Table 1.

Examples of tool tags

!!Title!!	Displays the title of the current file (without the extension) as text.
!!Child_Links_1!!	Displays the names, as links, of all the files contained within the directory. It does not display sub-directories. The list is sorted alphabetically and each shows the date and time when it was created.
!!Child_Links_2!!	Displays the names, as links, of all the subdirectories leading from the current directory. The list is sorted alphabetically.
!!Child_Links_3!!	Displays the entire contents of all text files in the current directory.
!!Child_Links_4!!	Displays the names, as links, of all the directories which have the same parent directory as the current directory. The list is sorted alphabetically.
!!Archive_Link!!	If files have been archived, a link to them is displayed. If no files have been archived, nothing is shown.
!!All_Child_1!!	Displays a list of articles which have been added within the number of days specified by the archive
!!Parent!!	Creates a link to the directory directly above the current directory
!!Home!!	Creates a link to the home page of the site
!!Feedback!!	Creates a link to the email address of whichever person who has been designated to receive such feedback.
!!Comment!!	Inserts text from a .CMT file in the sub-directory.
!!File_Text!!	Displays the full contents of a text file.

!!Sibling_Links_2!!	Displays, as links, all the other text files that are in the same sub-directory as the current text file. This is for use only within file directories.

Table 1

Step 606 provides for the creation of, in this exemplary embodiment, a set of eXtensible Markup Language (XML) templates for storage within the source file structure 700. It will again be appreciated however that any other suitable form of template could also be used.

The function of these XML templates is to assemble the HTML templates 716 to create the page format as desired. By way of example, a given XML template might call an HTML header template, an HTML title template, an HTML text template and an HTML footer template. An example of such an XML template is as follows:

```
<PAGE>
<COMPONENT_NAME1>
    <FILE> C:\xxx\yyy.aaa <FILE>
    <TAG1>YES</TAG1>
    <TAG2>YES</TAG2>
</COMPONENT_NAME1>
<COMPONENT_NAME2>
...
...
</COMPONENT_NAME2>
...
...
</PAGE>
```

As will be seen, the XML file begins with < PAGE > and ends with < /PAGE >. The COMPONENT_NAME is the name assigned to that portion of the page being defined. Examples of COMPONENT_NAME might be HEADER, TITLE or FOOTER. The < FILE > tag is used to indicate a location for the HTML template in question. < TAG1 > and <TAG2 > are the tags used in this component. As noted above, a list of such tags, by way of example, is provided in Table 1. < TAG1 >YES</TAG1> indicates that TAG1 will be used within the template being called. All tags used within the template are declared in this manner.

Functionality of a different type may be provided by, for example, an <ARCHIVE> component:

```
<ARCHIVE>
    <TYPE> R no of days </TYPE>
</ARCHIVE>
```

The inclusion of this component causes the amount of time that has elapsed since content files were stored in their respective directories to be tested. The date and time of storage of a file within a computer file system is typically available from the operating system as an associated property of the file concerned. If it is found, upon application of a template including such an archive component to a content file, that this time period is greater than a pre-determined period of time then the content file is archived. The archiving process will be discussed further below.

Having regard to Figure 7, these XML templates may be stored throughout the source file structure 700. By way of illustration of the operation of this embodiment however, the placing of such XML templates in only a single sub-directory will be here discussed in detail. Similar XML templates are however to be understood to be seeded throughout the source file structure 700 as desired. The third second-tier sub-directory 708 is shown as storing first and second XML templates 718,720. The function of the first XML template 718 is to define the appearance of content files as Web pages through the assembly of HTML templates in accordance with the discussion above. The somewhat different function of the second XML template 720 is to allow the creation of directory pages containing links to other pages, as will be discussed further below.

In the sense discussed with respect to the first and second embodiments, the first XML template 718 is said to be 'inherited' down the hierarchical source file structure 700. The effect of this application of 'inheritance' will be discussed below in greater detail.

Step 608 provides for the possibility of the creation of a file containing global information which may be useful for the operation of the tool. Accordingly control file 722, which may be written in XML, is stored in the separate directory 714.

An archive template 726, for example written in XML, may also be created and stored in the separate directory 714, the function of which is to determine the appearance of an archive page. An example of the global information stored in the

control file 722 might, for example, be the location of the storage of this archive template 726.

Step 610 provides for the introduction of the content files throughout the hierarchical source file structure 700. Again, by way of example, a text file (file extension, ' .txt') 724 is shown as being stored in the third second-tier sub-directory 708. Yet again, such files are to be understood as stored throughout the source file structure 700 as desired. Again, it will be well known that this may be done using, for example, Windows File Manager™ or Windows Explorer™ when using a Windows™ operating system.

Step 612 provides for the invocation of the tool.

The operation of the tool will now be explained with further reference to Figures 3,7 and 8.

When invoked, the tool may typically be passed at least three parameters: the location in the computer file system of a Control file (in this example in the separate directory 714), the location in the computer file system of a Source file (in this case the location of the source file structure 700) and the location of a 'target' directory into which the 'templated' Web pages may be output.

Given the location of the source file structure 700, as with the first and second embodiments, when invoked the tool will first perform a so-called 'tree traversing' procedure, in a top down fashion. Yet again in the course of this first procedure the

branching form of the file structure in terms of the root directory and the sub-directories will be detected and stored by the tool. Again, as indicated above, typically the description of the branching form stored by the tool includes information as to the name of a given directory, the relationship and name of the parent directory of the given directory, the relationships and names of any sub-directories of the given directory, the relationships and names of any other directories which share the same parent directory as the given directory and so on.

As with the first embodiment the tool next performs a tree-traversing procedure in a bottom-up fashion. This second procedure is similar to that discussed in respect of the first embodiment and will now be discussed having regard to the process flowchart indicated in Figure 3.

As will be clear from the above description, the templates used in this third embodiment differ from those discussed with respect to the first and second embodiments in that they are themselves assembled from fragment templates. At the equivalent of step 302 therefore the tool would search not for the template in the sense of the first embodiment but for the first XML template 718. It is this template which, when applied by the tool in the equivalent of step 304, would call the relevant HTML templates 716 to assemble the specified constructed template to be applied to the content file 724.

In accordance with the method disclosed in the applicant's co-pending application, the text-bearing content file 724 is used to generate an HTML file. However, in doing so, in accordance with this third embodiment according to the invention, the XML

template assembled HTML templates are also applied such that the new 'templated' file is not merely in an HTML form suitable for use as a Web page, but has, for example, the appearance desired in accordance with the templates. In this way the content file has been converted into a 'templated' Web page.

It is to be noted that were the text-bearing content file 724 as submitted by an external author to contain HTML statements, then these HTML statements could be honoured to the extent that they do not conflict with the templating HTML according to this embodiment of the invention. One example might be the inclusion of a hyperlink within the body of text of the content file. As with the first two embodiments the new file could of course be stored anywhere.

Instead, however, in this third embodiment the tool creates a 'target' file structure at the location specified to the tool upon invocation, into which the newly templated files created by the tool are output. The structure of the target file directory will mirror the source file structure. Using the example of the source file structure illustrated in Figure 7 and having regard to Figure 8, a mirrored target file structure 800 is created. A first-tier root directory 802 with first, second and third second-tier subdirectories 804,806,808 and first and second third-tier subdirectories 810,812 corresponding to those of Figure 7 is illustrated. A newly 'templated' file 814 corresponding to the content file 724 in the third second-tier subdirectory 708 of the source file structure 700 is stored in the third second-tier subdirectory 808 of the target file structure 800.

As indicated above, templates of the form of both the first and second XML templates 718,720 and content files as desired were to be understood as seeded throughout the source file structure 700.

The above method will be performed in respect of each content file stored in the source file structure 700. The relevant template or templates, the equivalents of the first XML template 718, will be applied to each content file and a corresponding templated HTML file, suitable for use as a Web page, will be output to the appropriate (sub) directory of the target file structure.

The second XML template 720 and its equivalents will be applied by the tool, utilising the description of the branching form of the source file structure 700 stored by the tool, to generate a directory document in respect of each (sub) directory of the target file structure. Having regard to Figure 8, a directory document 816 will be generated in the form of an HTML file, suitable for use as a Web page and is shown, by way of example, as being stored in the third second-tier sub-directory 808. HTML instructions included in this directory page 816 will cause links to be provided, for example, not only to allow access to all the 'templated' content-bearing Web pages in that (sub) directory, but also to navigate to, for example, the home or root directory, the parent directory, child (sub) directories and sibling directories (sharing the same parent directory) or, by way of further example, to an archive page.

An 'archive' link in the directory page will lead to an archive page, which may, for example, be created and stored in the same (sub) directory as the relevant directory page. In accordance with the above description of the archiving process, the inclusion

of an archive tag in an XML template will, upon the application of that template to a content file stored longer ago than a pre-determined period of time, cause that content file to be deemed to be an 'archived' file. The content file may still be used to generate a 'templated' Web page but no entry will appear in the 'current' directory page for the relevant (sub) directory, rather the directory page will contain a link to the above mentioned archive page, which will then provide links to the 'archived' 'templated' Web pages. Having regard to Figure 8, an example archive page 818 is illustrated, again stored in the third second-tier sub-directory 808.

As will be well known, to render the Web pages stored in the target directory available as a Web site, the location of the target directory in the computer file system can simply be provided to a Web server application program. One example of a suitable Web server application program is Microsoft Internet Information Server™. The Web server may then render the Web site accessible to external Web browsers upon suitable connection.

It will apparent from the above that the third embodiment of the method and tool according to the invention provides an advantageous means of Web site creation.

It will be apparent that a 'managed' Web site can be created, not with the attendant difficulties associated with the prior art, as discussed at length in the applicant's copending international patent application, but merely through the creation of a source file structure of appropriate form (for example relating to different types of meetings), the seeding throughout the structure of templates (in this example relating to the different types of meetings and 'cascading' down to allow for more general

meeting templates to be complemented or overridden to the extent provided for by more specific ones), the storing in the appropriate directories of the relevant documents (in this example associated with each different type of meeting) and finally, the execution of the method (the running of the tool).

Furthermore however, it will be appreciated that the third embodiment of the method and tool according to the invention also provides an advantageous means of Web site maintenance through the simple expedient of re-running the method and tool on a pre-existing source file structure.

If the 'look and feel' of the existing Web site is to be changed then, quite simply, the HTML templates stored in the separate directory could be edited and the tool re-run. The Web site would then be re-created but with the new HTML templates defining a new 'look and feel'. Yet further, either existing XML templates could be moved to different directories or new XML templates could be introduced into the appropriate directories of the source file hierarchy to alter the cascade of templates in the desired path and the tool re-run.

New content files from a wide variety of contributing authors could of course be introduced into the appropriate directories of the source file structure and the tool re-run to render the new content files as Web pages with the desired 'look and feel', as to achieve consistency with the existing pages.

In the prior art, if a re-structuring of a Web site is to be effected a manual amendment of the interconnecting hyperlinks is typically required. As will be well known, this

manual process can often lead to so-called 'broken links' where the document to which the hyperlink points no longer exists at that place.

If a re-structuring is required with the third embodiment according to the invention however, despite the source file structure being amended and documents and templates being moved about to effect the desired restructuring, since each time the tool is re-run the resulting Web site is re-created anew, no such breaking of links can occur.

Further, the archiving process may take place each time the tool is run, automatically ensuring that older content is archived and yet still reliably accessible.

As will be well known, such re-running of the tool can easily be provided as a scheduled event on, for example, a daily basis to provide ongoing advantageous management of the Web site according to the invention.

CLAIMS

1. A method of managing information-bearing content files stored in a computer file system; the computer file system being divided into directories; the method comprising:

locating one or more content files; each content file being stored in a directory of the computer file system;

associating one or more template files with each directory in which at least one content file is stored; each template file being effective, when applied to a content file, to carry out a respective pre-determined operation on the content file; and

applying the or each template file associated with a given directory to each content file stored in that directory.

2. A method as claimed in claim 1 in which the computer file system is divided into a hierarchical arrangement of directories and in which the one or more templates associated with each directory located in the direct hierarchical path from a directory in which a content file is stored up to and including the uppermost directory in the hierarchical arrangement are also associated with the directory in which the content file is stored.

3. A method as claimed in claim 1 or claim 2 in which the association of a template with a directory is made on the basis of the template file being stored in that directory.

4. A method as claimed in any of claims 1 to 3 further comprising:

associating metadata with each content file; and

carrying out the respective pre-determined operation on each content file upon the application of an associated template file on the basis of the respective associated metadata .

5. Apparatus for managing information-bearing content files stored in a computer file system; the computer file system being divided into directories; comprising:

means for locating one or more content files; each content file being stored in a directory of the computer file system;

means for associating one or more template files with each directory in which at least one content file is stored; each template file being effective, when applied to a content file, to carry out a respective pre-determined operation on the content file; and

means for applying the or each template file associated with a given directory to each content file stored in that directory.

6. Apparatus as claimed in claim 5 wherein the computer file system is divided into a hierarchical arrangement of directories, in which the means for associating one or more templates with each directory also associates with the directory in which the content file is stored, the one or more templates associated with each directory located in the direct hierarchical path from a directory in which a content file is stored up to and including the uppermost directory in the hierarchical arrangement.

7. Apparatus as claimed in claim 5 or claim 6 in which the the means for associating one or more templates with each directory makes the association of a template with a directory on the basis of the template file being stored in that directory.

8. Apparatus as claimed in any of claims 5 to 7 further comprising:

means for associating metadata with each content file; wherein

the respective pre-determined operation on each content file upon the application of an associated template file is carried out on the basis of the respective associated metadata .

9. A computer program storage device readable by a computer, said device embodying computer readable code executable by the computer to perform the method according to any one of claims 1 to 4.

10. A signal embodying computer executable code for loading into a computer for the performance of the method according to any one of claims 1 to 4.