

memory (4). A HTML to WML converter (5) is used to convert said HTML content into WML content (6) and save it as a file in the server (7). Upon the conversion completion said background process WML page (8) is redirected to a final result WML page (9), which displays the content of said stored file in the server (7).

TECHNICAL FIELD

This invention relates in general to software engineering, and more particularly to method and system that browsing hypertext content from a WAP device.

BACKGROUND OF THE INVENTION

Wireless tele-communication has gained widespread popularity and has dramatically changed the style of our life. The Wireless Application Protocol (WAP) allows people to send email and surf the Internet with their mobile phone or PDA (Personal Digital Assistant). WAP service is expected to be accessible wherever mobile phone can be used in the near future.

However, recently one problem that persists is that the WAP users are not able to access most web sites, which is in the hypertext format. Only sites coded in wireless Markup Language (WML) can be accessed by WAP browsers, the rest bulk of the Internet, which is coded in Hyper Text Markup Language (HTML), remain inaccessible to WAP browser. To enable the WAP user to browser their web pages, companies have to develop two versions of web



pages at the same time: one is in the HTML format while another is in the WML format.

To solve this problem, some conversion logics have been suggested to convert HTML to WML. However till now (which means at the end of year 2000) no practical system has been reported to be developed to allow the WAP user to browser the HTML contents. The reasons why it is not practical for most existing methods to dynamically convert HTML to WML are as following:

1. As is known in the art, most web pages are the combination of scripting language (such as ASP, jSP, CGI and JavaScript, vbscript) and the markup language (such as HTML and XML), client to server request will no long work properly without the conversion of the scripting language.

2. The conversion of HTML to WML alone can only solve the problem of server to client response, which means that the conversion of HTML to WML alone can only be used to browser the hypertext content from a WAP device.

BRIEF DESCRIPTION OF THE PRIOR ART

An example of a method and system for the browsing hypertext pages by means of a mobile station in a telecommunication system is disclosed in Canada IPO No. CA2315653 or International Publication No. WO99/35595,



whereas a network infrastructure rather than the software, was described.

As for the software fields, a method and system for converting content of electronic data for wireless services is provided by International Publication No. WO 00/39666. However, it focuses on the logic and method for the content conversion rather than a complete system.

SUMMARY OF THE INVENTION

In accordance with preferred embodiments of the present invention, some of the problems associated with inaccessibility of most HTML web sites by WAP users are overcome.

A method and system for browsing hypertext content from a WAP device by means of a server-side application is provided. The method includes an Internet address Entrance WML page, a background processing WML page and a final result WML page. The Internet address Entrance WML page allows the user to enter the Internet address of hypertext content and submit to the background processing WML page in WAP protocol. The background processing WML page then load the hypertext content from remote INTERNET ADDRESS and convert the hypertext content into WML content with a pre-defined conversion logic by invoking a background process. Upon the conversion process completion, the converted contents are saved as a WML file on the server and the background processing WML page



is redirected to a final result WML page which displays the content of said converted WML file on the server.

The above as well as additional methods, features and advantages of the present invention will become apparent in the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

FIG. 1 depicts a hypertext browsing system in which a preferred embodiment of the present invention may be implemented.

FIG. 2 is a high level flowchart of the background process in accordance with a preferred embodiment of the present invention.

This Sheet Has Been Replaced



DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

With reference now to FIG. 1, an exemplary system for the browsing of hypertext documents from a WAP device by means of a server-side application in which a preferred embodiment of the present invention may be implemented is depicted. A WAP device (1) send a WAP request to the INTERNET ADDRESS Entrance WML page (2) where the user is allowed to enter the URL (Uniform Resource Locators, i.e. Internet address) of HTML (Hyper Text Markup Language) content. The URL Entrance WML page (2) then sends another WAP request to the background WML page (8). As a consequence, a background procedure will be triggered off to send a HTTP request to the Internet (3), receive a HTTP response from Internet and then stored the hypertext content in the memory. A HTML to WML converter (5) will then convert the HTML content into WML content (6) and save it as a file in the server (7). Upon the conversion process completion, the background processing WML page (8) will be redirected to the final result WML page (9) which displays the content of said converted WML file on the server (7).

In one exemplary preferred embodiment of the present invention, a practical application has been created with the Wireless Markup Language (WML) and Microsoft's Active Server Pages (ASP). However, other server scripting language such as the Common Gateway Interface (CGI), Java Server Pages (JSP), Java Servlets or Perl could also be used and the present invention is not limited to the Active Server Pages (ASP).



Table 1 shows an exemplary URL Entrance WML page (2), which comprises of the following elements:

i) A textbox allows user to enter URL of the remote HTML site address.

ii) A hyperlink which links to the background processing WML page (8).

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<?xml version="1.0"?>
  <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml_1.2.xml">
<wml>
  <card id="url_card" title="URL_Entrance">
    <do type="accept" label="GO!">
      <go href="conv.asp?url=${url}"/>
    </do>
    <p>
      URL: <input name="url" value=""/><br/>
    </p>
  </card>
```

Table 1. An exemplary URL Entrance WML page

FIG 2. Illustrates an exemplary flowchart of the background process comprising of the following steps:

At step 10, the background processing WML page gets the URL from the query string of request object of ASP (Active Server Pages).

To get the remote HTML content, the server need to and send a HTTP request to the Internet and then receive the HTML content and store the HTML content in the memory. In one exemplary preferred embodiment of the present invention, this is accomplished by a server component built with Microsoft's Active-X as is shown in Table 2:

This Sheet Has Been Replaced



The following is an exemplary steps to create the Active-X remote URL fetching server component with Visual Basic:

1. making a new "Active-X DLL" project
2. under the Project; References menu, activate the "Microsoft Internet Transfer Control" library(i.e. msinet.ocx) .
3. Suppose we set the project name as "HTTPCom"
4. Suppose we set the class name "GetHTTP"
5. Add in the Visual Basic source code for the component as following:

```
Public Url As String
Public Function Open_URL() As String
    Dim ConnObj As InetCtlsObjects.Inet
    Set ConnObj = New InetCtlsObjects.Inet
    ConnObj.AccessType = icUseDefault
    Open_URL = ConnObj.OpenURL(Url)
End Function
```

6. To register the server component , copy it to where the system DLLs are (e.g. \winnt\system32 for Windows NT), and then from a command prompt execute:
regsvr32 HTTPCom.dll

Table 2. Exemplary steps

to create the remote URL reading server component

Beside the Microsoft's Active-X, other server scripting language such as Java Applet could also be used to create the remote URL fetching server component and the present invention is not limited to the Microsoft's Active-X.

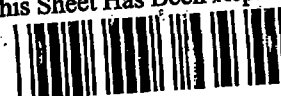
Then at step 11, the above server component we just created can be invoked on an ASP page with the following code as Table 3

The following is an exemplary steps to invoke the remote URL fetching server component within ASP:

```
<%
Set HttpObj = Server.CreateObject("HTTPCom.GetHTTP")
HttpObj.Url = URLstring
fetched_htm_text = HttpObj.Open_URL
set HttpObj=nothing
%>
```

Table 3. Exemplary steps to invoke

The remote URL fetching server component within ASP



At step 13, the fetched HTML content is temporarily stored in the memory.

At step 14, a HTML to WML converting process is invoked.

In one exemplary preferred embodiment of the present invention, the HTML is converted to WML by classifying the HTML tags into 3 categories: the deleting tag, the bypassing tag and the converting tag, as shown in Table 4. The deleting tag is the type of HTML tag whose open tag and close tag as well as all the text between its open tag and close tag will be deleted; whereas the bypassing tag is the type of HTML tag whose open tag and close tag will be deleted, but the text between its open tag and close tag will remain unchanged; the converting tag is the type of HTML tag whose open tag and close tag will be converted to the corresponding WML tag based on a pre-defined conversion logic as shown in Table 5 and the text between its open tag and close tag will remain unchanged.

	HTML tags listing
Deleting Tags	address, applet, area, blink, csobj, dd, del, dl, dt, embed, fig, form, iframe, image, input, isindex, link, map, media, option, param, range, script, select, select, span, style, textarea, textflow, title
Bypassing Tags	blockquote, dfn, dir, div, em, font, head, hr, ins, kbd, lang, lh, li, listing, marquee, math, menu, multicol, nobr, noframes, noscript, note, nowrap, ol, overlay, person, plaintext, pre, samp, small, spacer, spot, tab, table, tbody, tfoot, th, thead, tr, tt, ul, xmp, img
Converting Tags	body, center, meta, strong, wbr, var, sub, sup, strike, em, h1, h2, h3, h4, h5, h6, br, td, fn, b, p, i, u, a

Table 4 Three categories HTML tags listing

This Sheet Has Been Replaced.



Old HTML Tag	New WML Tag	
	Open Tag	Close Tag
body	card><p	/p></card
center	/p><p align="center"	/p><p
meta	head><meta http-equiv="cache-control" content="max-age=time"/	/head
strong	b	/b
wbr	br	N.A.
var	i	/i
sub	i	/i
sup	i	/i
struck	u	/u
em	b	/b
h1	b	/b
h2	b	/b
h3	i	/i
h4	i	i
h5	i	/i
h6	i	/i
br	br/	N.A.
td	br/	N.A.
fn	i	/i
b	b	/b
p	br	N.A.
i	i	/i
u	u	/u
a href=url	a href=conv.asp?url=url	/a

Table 5 HTML to WML tag conversion logic

In the above exemplary preferred embodiment of the present invention, the HTML to WML conversion logic includes following specifications:

1. The HTML hyperlink is replaced with the WML hyperlink, which is linked to the server application.

2. Due to the purpose of browsing only, it only converts text and links and disable the input tags.



Exemplary program logic used with exemplary pseudo-code of background process are illustrated below:

```

Program Start
Get the URL String
Create HTTP Connection object
Fetch the HTML contents
Store the content in the memory
For each character in the memory buffer
  If it's not end of buffer
    If it's not a special character such as "carriage return"
      If it's not the tag open token "<"
        If it's not a character between the tag open token "<"
          and tag close
            If it's not the tag close token "<"
              Get the tag name and Set tag_opened=0
              If it belongs to the deleting tag
                IF it belongs to the converting tag
                  Call subroutine to store the converted tag
                  Goto the next character
              Else
                Goto the next character
            Else
              Set del_tag_flag>0
              Goto the next character
          else
            IF the del_tag_flag>0
              Call subroutine to store the character
              Goto the next character
            Else
              Goto the next character
          Else
            Append the character into a string variable
          Else
            Initialization Set tag_opened=1
            Goto the next character
        Else
          Goto the next character
      Else
        Write into a WML file
        Redirect to the WML file
        Exit
    End of Loop
  End of Program

Function start
Get the text to be saved
If the current card buffer exceed the maximum size
  Open the next card buffer (i.e. card count increase 1)
  IF there are tags have not been closed
    Close all opening tags and append them into the current
    card buffer
  End if
End if
Append the text into the current card buffer
End of Function

```

Table 6. Exemplary pseudo-code of background process



Returning to Fig 2, the exemplary program flow of background process can be explained in the details as following:

At step 14, the program read each character from the HTML content in the memory buffer and bypass the following special characters: backspace, tab, linefeed, and carriage return characters at step 17.

At step 22, the program read out the string between a beginning tag token "<" and a ending tag token ">", get the tag indicator string, and match it with the tag list saved in the included file,

At step 28, if the tag belongs to the deleting tag, then erase the current tag string, as well as all of the content between said open tag and close tag of current tag at step 29;

At step 30, if the tag belongs to the converting tag, then replace the current tag string with the conversion tag as in the step 31, the content between open tag and close tag of current tag will remain as what it originally is. If the tag belongs to the hyperlink, point to the ASP file.

For all other tag which is neither the deleting tag nor the converting tag, erase the current tag string only, the content between open tag and close tag of current tag will remain as what it originally is;

In the converted tag saving subroutine at step 33, split the whole content into a multiple cards, each card contain a unique cards ID and has a maximum size of a deck (Most telephones won't accept a WML deck bigger than 2 Kilobytes) which is defined in the include file as of 2 kb. If the split point happened to be on the middle of a tag, the card before the splitting point must close the



opening tag and the card after the splitting point must open the same tag again. An array has to be built to save the opening tags.

The final step of background process is to save the converted cards into a file on the server (at step 16). The file name can use the URL string with the special characters like /,:,*,?,<,>,| removed. Every file must and a WML header like "<?xml version="1.0"?><!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">" . One HTML file converted to one WML file, which contains multiple cards.

After step 16 completed, the background processing WML page will be redirected to the saved WML file on the server as at step 18.

Since the stored files are growing bigger and bigger in size, it is necessary to have a scheduler to maintain the converted file on the server. In the above exemplary preferred embodiment of the present invention, a WINDOWS NT scheduler has been applied to delete the used files one day before as shown in the Table 7 as below:

```
@ECHO OFF
REM This batch file should be invoked once the window is loaded.
REM To delete all of .WML file in the conversion directory
at 02:00
/every:monday,tuesday,wednesday,thursday,friday,saturday,sunday
"del d:\Conv\Conv_data\*.wml"
```

Table 7. A WINDOWS NT scheduler to maintain the server



Beside the WINDOWS NT scheduler, other schedulers can also be used to maintain the server and the present invention is not limited to the WINDOWS NT scheduler.

It is also possible to combine some or all of above steps into a single server component and the present invention is not limited to the number of steps we have given.

This Sheet Has Been Replaced



The source code of one exemplary background process program is listed in Table 8 as below:

```
<!--#include file="array.asp"-->
<?xml version="1.0"?>
  <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml_1.2.xml">

<wml>
  <card id="background" title="Background Process">
<p>
  <b> Please Wait while processing...</b>
</p>
<%
'-----
'      Exemplary ASP program for background process
'-----

'-----
'      Variables Initialization
'-----

dim fetched_htm_text
' The HTML Content fetched from the remote URL
dim conv_tag_array(100) ,conv_tag_cnt
'array to store the currently opening tag
dim conv_wml_card(1000) ,conv_wml_cnt
'array to store the content of converted WML cards
dim loop_cnt,link_loop_cnt
dim link_url_str

'-----
'      Subroutine to save into buffer
' The "save_wml_text" is a subroutine to save a string into the
' buffer Since the buffer contain a plurities of cards, each tim
' before saving, this subroutine will check the size of the
' existing card buffer so that the size each card will not
exceed
' the maximum size allowed also this subroutine check all the
opened
' tag and close them before starting to write into a new card.
'-----

Sub save_wml_text(str_saving)
dim len_str,array_cnt
len_str=len(str_saving)
if len_str>0 and str_saving<>"<>" then
' filter out those str_saving is null
if len(conv_wml_card(conv_card_cnt)) > (max_card_size-
len_str) then
```



```

conv_wml_card(conv_card_cnt+1)="
if conv_tag_cnt > 0 then
  for array_cnt=conv_tag_cnt-1 to 0 step -1
    if conv_tag_array(array_cnt)=0 then
      ' when body encountered, need to add in card
link
      if conv_card_cnt >0 then
        ' link to last card

conv_wml_card(conv_card_cnt)=conv_wml_card(conv_card_cnt) & "<a
href=""#card" & cstr(conv_card_cnt) & "">To Card" &
cstr(conv_card_cnt) & "</a>"
      end if
      ' link to next card

conv_wml_card(conv_card_cnt)=conv_wml_card(conv_card_cnt) & "<a
href=""#card" & cstr(conv_card_cnt+2) & "">To Card" &
cstr(conv_card_cnt+2) & "</a>"
      end if

conv_wml_card(conv_card_cnt)=conv_wml_card(conv_card_cnt) & "<" &
convert_to_close_tag(conv_tag_array(array_cnt)) & ">"
next
  for array_cnt=0 to conv_tag_cnt-1
    if converting_tag(conv_tag_array(array_cnt))="body"
then ' when body encountered, need to add in card id

conv_wml_card(conv_card_cnt+1)=conv_wml_card(conv_card_cnt+1) &
"<card id=""card"&cstr(conv_card_cnt+2)&"" title=""page
"&cstr(conv_card_cnt+2)&""><p>"
      else

conv_wml_card(conv_card_cnt+1)=conv_wml_card(conv_card_cnt+1)&
"<" & convert_to_open_tag(conv_tag_array(array_cnt)) & ">"
      end if
      next
    end if
    conv_card_cnt =conv_card_cnt +1

conv_wml_card(conv_card_cnt)=conv_wml_card(conv_card_cnt)&str_sav
ing
  else

conv_wml_card(conv_card_cnt)=conv_wml_card(conv_card_cnt)&str_sav
ing
  end if
end if
End Sub
'-----
'   Program Start...
'-----

```

This Sheet Has Been Replaced



```

'-----
'      Create the server component "HTTPCom.GetHTTP"
'      Get the HTML contents from the remote URL
'-----

Server.ScriptTimeout = 240
URLstring = Request(url)      ' Get the URL string from Request

Set HttpObj = Server.CreateObject("HTTPCom.GetHTTP")
HttpObj.Url = URLstring
fetched_htm_text = lcase(HttpObj.Open_URL)
set HttpObj=nothing

'-----
'      Variable Initialazation
'-----

is_tag_opening=0      ' bool variable to indicate the status of
current tag : 1-is opening 0-closed
del_tag_ind=-1      ' integer variable to indicate whether the
content inside a tag should be totally deleted or not. (-1)- not
to delete (>=0) - to delete
belongs_to_tag=0      ' bool variable to indicate where the
current token belongs to an tag string or not 0-no 1-yes
full_tag_str=""      ' full text of tag string between "<" and
">"
tag_ind_str=""      ' Tag string like body or table
conv_tag_cnt=0      ' Converted tag counter
conv_card_cnt=0      ' Converted wml cards counter
conv_wml_card(0)=""
len_htm_text=len(fetched_htm_text)
del_tag=split(del_tag_str,",")
disable_tag=split(disable_tag_str,",")
converting_tag=split(converting_tag_str,",")
convert_to_open_tag=split(convert_to_open_tag_str,",")
convert_to_close_tag=split(convert_to_close_tag_str,",")

'-----
'      Start file processing
'-----

for pos=1 to len_htm_text
    in_char=mid(fetched_htm_text,pos,1)
    '8, 9, 10, and 13 convert to backspace, tab, linefeed, and
    carriage return characters, respectively.
    if in_char<>chr(8) and in_char<>chr(9) and in_char<>chr(10)
    and in_char<>chr(13) then

```

This Sheet Has Been Replaced



```

'-----
'      When a begining tag token "<" encountered
'-----
      if in_char= "<" then
        ' begin of a tag
        is_tag_opening=1
        belongs_to_tag=0
        full_tag_str="" ' set full tag string to null
        tag_ind_str="" 'set tag string to null
'-----
'      When inside "<" and ">"
'-----
      elseif is_tag_opening=1 and in_char<> ">" then 'if inside <
to >
        full_tag_str=full_tag_str+in_char
        if in_char= " " then
          ' end of getting the tag indicator str
          belongs_to_tag=1
        end if
        if belongs_to_tag=0 and in_char<> ">" then
          ' start to get the tag indicator str
          tag_ind_str=tag_ind_str+in_char
        end if
'-----
'      When a ending tag indicator ">" encountered
'-----
      elseif is_tag_opening=1 and in_char= ">" then
        ' end of a tag , start actual tag conversion
'-----
'      Actual Conversion Start
'-----
is_tag_opening=0
'-----
'      First loop thru all deleting tag
'-----
for loop_cnt=0 to UBound(del_tag)
  if trim(tag_ind_str)= del_tag(loop_cnt) then 'if it's
deleting open tag
    if del_tag_ind=-1 then
      del_tag_ind=loop_cnt ' for those content need to
be deleted,
    end if
  exit for
  elseif trim(tag_ind_str)= "/" & del_tag(loop_cnt)
then 'if it's deleting close tag
    if del_tag_ind=loop_cnt then
      del_tag_ind=-1

```



```

        end if
        exit for
    end if
next
'-----
'      If the current tag not belong to the deleting tags,
'      then loop thru all converting tag
'-----
    if del_tag_ind=(-1) then ' if the current tag not belong
to the deleting tags
        for loop_cnt=0 to UBound(converting_tag) 'loop thru all
converting tag
            if trim(tag_ind_str)= converting_tag(loop_cnt) then
                'if it's converting open tag, save the tag into a
array
                conv_tag_array(conv_tag_cnt)=loop_cnt
                conv_tag_cnt=conv_tag_cnt+1
                if trim(tag_ind_str)="body" then ' when body
encountered, need to add in card id
                    ' card number start from 1
                    save_wml_text("<card
id=""card"&cstr(conv_card_cnt+1)&"" title=""page
"&cstr(conv_card_cnt+1)&""><p>")
                elseif trim(tag_ind_str)="a" and full_tag_str<>">
then
                    ' in case of an hyperlink
                    link_url_str=""
                    link_tag_str=split(full_tag_str," ")
                    link_loop_cnt=1
                    do
                        if instr(link_tag_str(link_loop_cnt),"href")>0
then
link_url_str=replace(link_tag_str(link_loop_cnt),"href","")
                            link_url_str=replace(link_url_str,"=", "")
                            link_url_str=replace(link_url_str,"'", "")
                            link_url_str=replace(link_url_str,"'", "")
                            exit do
                        end if
                        link_loop_cnt=link_loop_cnt+1
                    loop while link_loop_cnt < UBound(link_tag_str)
                    if link_url_str <> "" then
                        save_wml_text("<a href='htm.asp?url=' &
trim(link_url_str) & '>")
                    else
                        save_wml_text("<a>")
                    end if
                else
                    save_wml_text("<" & convert_to_open_tag(loop_cnt)
& ">")

```

This Sheet Has Been Replaced



```

        end if
        exit for
    elseif trim(tag_ind_str)= "/" &
converting_tag(loop_cnt) then
'if it's converting close tag
        if conv_tag_array(conv_tag_cnt-1)=loop_cnt then
            ' if previous tag is the open tag of current tag,
then remove this tag from array

            conv_tag_cnt=conv_tag_cnt-1
            if trim(tag_ind_str)= "/body" then ' when body
encountered, need to add in to last card

conv_wml_card(conv_card_cnt)=conv_wml_card(conv_card_cnt) & "<a
href=""#card" & cstr(conv_card_cnt) & "">To Card" &
cstr(conv_card_cnt) & "</a>"
            end if
            save_wml_text("<" &
convert_to_close_tag(loop_cnt) & ">")
        else
            ' if previous tag failed to be closed, then force
it to be close
            do
                conv_tag_cnt=conv_tag_cnt-1
                save_wml_text("<" &
convert_to_close_tag(loop_cnt) & ">")
                loop while conv_tag_cnt>0 and
conv_tag_array(conv_tag_cnt-1)<>loop_cnt

            end if
            exit for
        end if
    next
end if

'-----
'           Do not save any thing between the open tage
'           and close tag of a deleting tag
'-----
elseif del_tag_ind>=0 then
'-----
'           Save all characters as usual
'           if they are between the open tage and close tag
'           of a non-deleting tag
'-----
else
    save_wml_text(in_char)
end if

```

This Sheet Has Been Replaced



```

end if
'-----
'      Conversion Complete
'-----
next

'-----
'      Write to file
'-----

conv_url=replace(lcase(URLstring),"http://","")
conv_url=replace(conv_url,"\"","_")
conv_url=replace(conv_url,"/","_")
conv_url=replace(conv_url,":","_")
conv_url=replace(conv_url,"*","_")
conv_url=replace(conv_url,"?","_")
conv_url=replace(conv_url,"<","_")
conv_url=replace(conv_url,">","_")
conv_url=replace(conv_url,"|","_")
conv_url=replace(conv_url,"=","_")
conv_url=replace(conv_url,".","_")
' save the buffer into a file with the filename as URL string in
which special characters have been removed

conv_filedir=Server.Mappath ("../")
conv_filename=conv_filedir & "\Conv\Conv_Data\" & conv_url &
".wml"

Set fs_conv = server.CreateObject("Scripting.FileSystemObject")
Set conv_wmlfile = fs_conv.CreateTextFile(conv_filename, true)

conv_wmlfile.WriteLine "<?xml version=\"1.0\"?><!DOCTYPE wml
PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">"
conv_wmlfile.WriteLine "<wml>"
for loop_cnt=0 to conv_card_cnt
    conv_wmlfile.WriteLine conv_wml_card(loop_cnt)
next
conv_wmlfile.WriteLine "</wml>"
conv_wmlfile.Close

set conv_wmlfile=nothing
set fs_conv=nothing
%>
</card>
</wml>

```

Table 8. An exemplary background process program
source code listing





Claims

What is claimed is:

1. Method of browsing hypertext content from a WAP device, comprising:

pulling out the hypertext content from remote Internet address;

converting HTML content into WML content;

storing converted WML file on the server

redirecting to said converted WML file.

2. The method of claim 1, wherein the method of hypertext browsing includes an application built with Hyper Text Markup Language (WML) and Server scripting language.

3. The method of claim 1, wherein the method of browsing hypertext content includes a WML page for Internet address Entrance, a WML page for background processing and a WML page for display the converted contents.

4. The method of claim 1, wherein pulling out the hypertext content from remote Internet address further comprises of sending a HTTP request to the Internet and then receiving the HTML content in HTTP response and storing said content in the memory.

This Sheet Has Been Replaced





5. The method of claim 1, wherein pulling out the hypertext content from remote Internet address includes using of a server component.

6. The method of claim 1, wherein converting HTML content into WML content includes the replacement of hyperlink of HTML content into the hyperlink of WML content and said hyperlink of WML content is linked to a server application.

7. The method of claim 1, wherein converting HTML content into WML content is a background procedure.

8. The method of claim 1, wherein converting HTML content into WML content includes classifying the HTML tags into the deleting tag, the bypassing tag and the converting tag.

9. The method of claim 1, wherein converting HTML content into WML content includes splitting the whole content into a plurality of WML cards;

10. The method of claim 1, wherein browsing hypertext content from a WAP device includes a scheduler to maintain the converted files on the server.

This Sheet Has Been Replaced

