SHELLEXECUTE: A TOOL FOR LAUNCHING EXTERNAL PROGRAMS

ERWIN KALVELAGEN

ABSTRACT. SHELLEXECUTE is a small wrapper program for the `shellexecute` Windows API call. It allows you to spawn an external program based on the file type of the document to open.

1. Overview

GAMS provides the `execute` statement to execute external programs. In some cases we want to let the system figure out what application to start for a given document. This can be accomplished with SHELLEXECUTE. For instance, when we call:

```
> shellexecute demo.html
```

Windows will launch the web browser and show `demo.html`. This works correctly, irrelevant whether the user installed Microsoft Internet Explorer or Netscape's web browser.

The program is a wrapper around the `shellexecute` API call. For more information see: `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/functions/shellexecute.asp`

2. Usage

The command line for SHELLEXECUTE looks like:

SHELLEXECUTE [**/verb=vvv**] [**/showcmd=flag**] [**/dir=ddd**] **filename args**

The filename is either a document (e.g. `demo.html` or `book1.xls`) or an application (e.g. `winword.exe` or `notepad`). If a document is provided, the associated application will be launched. If a directory name is provided the windows explorer will be launched.

If additional parameters are specified they are considered as command line parameters for the process to be spawned.

The following options are recognized:

/verb=vvv:
Specifies the action to be performed. The allowed actions are application dependent. Some commonly available verbs include:

Date: March 25, 2004.

verb	**Description**
edit	Launches an editor and opens the document for editing.
find	Initiates a search starting from the specified directory.
open	Launches an application. If this file is not an executable file, its associated application is launched.
print	Prints the document file.
properties	Displays the object's properties.

If no verb is specified the default command for the file class will be used (in many cases this is "**open**").

/showcmd=flag:

Flag that specifies how an application is to be displayed when it is opened. The actual behavior is up to the launched program. The possible values are:

showcmd	**Description**
`SW_HIDE`	Hides the window and activates another window.
`SW_MAXIMIZE`	Maximizes the specified window.
`SW_MINIMIZE`	Minimizes the specified window and activates the next top-level window in the z-order.
`SW_RESTORE`	Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.
`SW_SHOW`	Activates the window and displays it in its current size and position.
`SW_SHOWMAXIMIZED`	Activates the window and displays it as a maximized window.
`SW_SHOWMINIMIZED`	Activates the window and displays it as a minimized window.
`SW_SHOWMINNOACTIVE`	Displays the window as a minimized window. The active window remains active.
`SW_SHOWNA`	Displays the window in its current state. The active window remains active.
`SW_SHOWNOACTIVATE`	Displays a window in its most recent size and position. The active window remains active.
`SW_SHOWNORMAL`	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

The default is `SW_SHOWNORMAL`.

/dir=ddd:

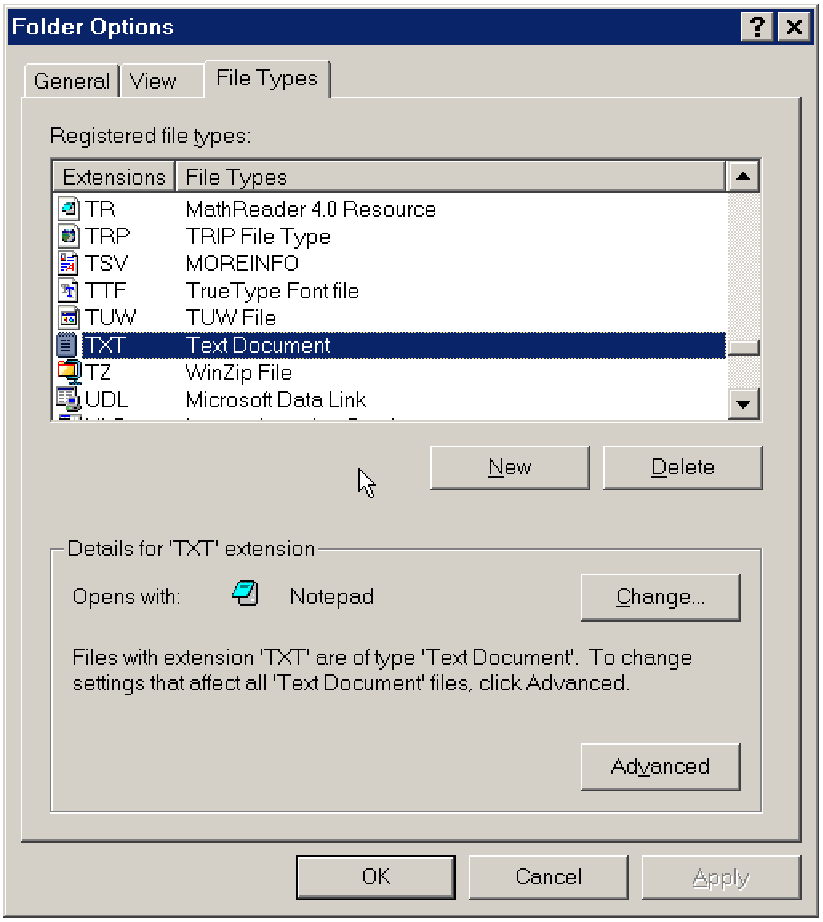
the default directory for the sub-process.

In many cases you will not need to use any options.

3. Examples

Below are some examples of ShellExecute. It is noted that much of the behavior is depending on the file associations that are installed on your machine. These file associations can be inspected and changed with the Windows Explorer, see figure 1.

3.1. Spawning a web browser.



Registered file types:

Extensions	File Types
TR	MathReader 4.0 Resource
TRP	TRIP File Type
TSV	MOREINFO
TTF	TrueType Font file
TUW	TUW File
TXT	Text Document
TZ	WinZip File
UDL	Microsoft Data Link

Details for 'TXT' extension

Opens with: Notepad

Files with extension 'TXT' are of type 'Text Document'. To change settings that affect all 'Text Document' files, click Advanced.

FIGURE 1. File associations in the Windows Explorer

```
>shellexecute turkey.html
```

3.2. Spawning notepad.

```
>shellexecute trnsport.txt
```

If a different program is associated with a .TXT file, a different program will be launched. In my case notepad will be launched due to the file association shown in figure 1.

3.3. Spawning the GAMS IDE to view a GDX file.

```
>shellexecute trnsport.gdx
```

3.4. Spawning Excel.

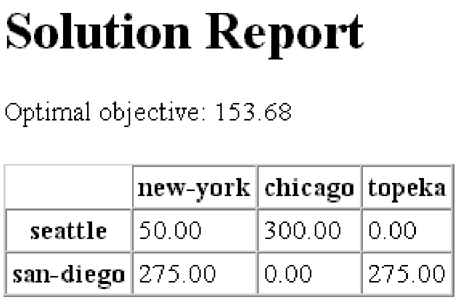
```
>shellexecute test.xls
```

3.5. Calling shellexecute from GAMS.

```
execute '=shellexecute trnsport.html';
```

3.6. A complete GAMS example.

The following model is the `trnsport` model from the model library. We write the solution both to an HTML file and a CSV file. HTML is standard ASCII and is easily written using the PUT facility. CSV is even easier, as setting the FILE suffix `csv.pc=5;` will automatically generate comma separated values. The web browser is launched to view the HTML file, while Excel will be spawned to view the CSV file.

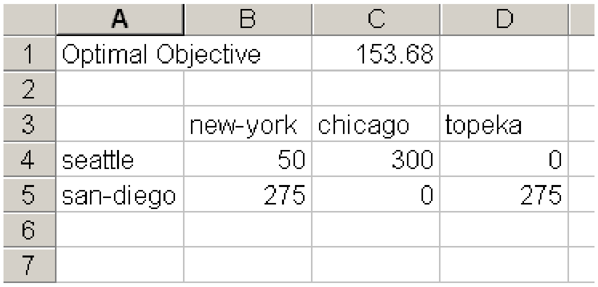


Solution Report

Optimal objective: 153.68

	new-york	chicago	topeka
seattle	50.00	300.00	0.00
san-diego	275.00	0.00	275.00

Figure 2. HTML report rendered by web browser



	A	B	C	D
1	Optimal Objective		153.68	
2				
3		new-york	chicago	topeka
4	seattle	50	300	0
5	san-diego	275	0	275
6				
7				

Figure 3. CSV file as displayed by Excel

```
$Title  A Transportation Problem (TRNSPORT,SEQ=1)
$Ontext

  Write a solution report in HTML and CSV and spawn
  a browser and Excel to view the results.

  Erwin Kalvelagen, May 2004

$Offtext


  Sets
       i   canning plants   / seattle, san-diego /
       j   markets          / new-york, chicago, topeka / ;

  Parameters
```

```
       a(i)  capacity of plant i in cases
         /    seattle     350
              san-diego   600  /

       b(j)  demand at market j in cases
         /    new-york    325
              chicago     300
              topeka      275  / ;

  Table d(i,j)  distance in thousands of miles
                    new-york       chicago      topeka
      seattle          2.5           1.7          1.8
      san-diego        2.5           1.8          1.4  ;

  Scalar f  freight in dollars per case per thousand miles  /90/ ;

  Parameter c(i,j)  transport cost in thousands of dollars per case ;

            c(i,j) = f * d(i,j) / 1000 ;

  Variables
       x(i,j)  shipment quantities in cases
       z       total transportation costs in thousands of dollars ;

  Positive Variable x ;

  Equations
       cost        define objective function
       supply(i)   observe supply limit at plant i
       demand(j)   satisfy demand at market j ;

  cost ..        z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

  supply(i) ..   sum(j, x(i,j))  =l=  a(i) ;

  demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;

  Model transport /all/ ;

  Solve transport using lp minimizing z ;

  Display x.l, x.m ;


*---------------------------------------------------------------------------
* write a solution report in HTML and launch the browser
*---------------------------------------------------------------------------

file html /’results.html’/;
put html;

put ’<H1>Solution Report</H1>’/;
put ’Optimal objective:’,z.l/;
put ’<p>’/;

put ’<table border="1" cellpadding="3" cellspacing="0">’/;

put ’<tr><td></td>’;
loop(j, put ’<th>’,j.tl,’</th>’ );
put ’</tr>’/;

loop(i,
   put ’<tr><th>’,i.tl,’</th>’;
   loop(j, put ’<td>’,x.l(i,j),’</td>’ );
   put ’</tr>’/;
);

put ’</table>’/;

putclose;

execute ’=shellexecute results.html’;
```

```
*------------------------------------------------------------------------
* write a solution report in CSV and spawn Excel to view it
*------------------------------------------------------------------------


file csv /results.csv/;
csv.pc=5;
put csv;
put "Optimal Objective","",z.l/;
put /;
put ’’;
loop(j, put j.tl);
put /;

loop(i,
   put i.tl;
   loop(j, put x.l(i,j));
   put /;
);

putclose;
execute ’=shellexecute results.csv’;
```

3.7. **Exporting to MS Access.** This example will launch MS Access after exporting a parameter to a MDB file. For more information see [1].

```
$ontext

 Test of GDX2ACCESS. Dumps a symbol to an Access Database,
 and launches MS ACCESS.

$offtext

set i /i1*i100/;
alias (i,j);
parameter p(i,j);
p(i,j) = uniform(-100,100);
execute_unload ’test.gdx’,p;

execute ’=gdx2access test.gdx’;
execute ’=ShellExecute test.mdb’;
```

4. Common error messages

4.1. File does not exist.

```
E:\wtools\ver000\examples>shellexecute xxxx.html
ShellExecute Version 1.0
The system cannot find the file specified
```

4.2. No file association.

```
E:\wtools\ver000\examples>shellexecute table02.dbf
ShellExecute Version 1.0
No application is associated with the specified file for this operation
```

References

1. Erwin Kalvelagen, *GDX2ACCESS: A Tool to Convert GDX Data to MS ACCESS Tables*, http://www.gams.com/~erwin/interface/gdx2access.pdf.

GAMS Development Corp.
E-mail address: erwin@gams.com