

## Re: <resource> elements in WSCs

**Source:** <http://www.tech-archive.net/Archive/Scripting/microsoft.public.scripting.wsh/2004-11/0259.html>

---

**From:** Al Dunbar [MS-MVP] ([alan-no-drub-spam\\_at\\_hotmail.com](mailto:alan-no-drub-spam_at_hotmail.com))

**Date:** 11/13/04

Date: Sat, 13 Nov 2004 10:29:06 -0700

"Al Dunbar [MS-MVP]" <[alan-no-drub-spam@hotmail.com](mailto:alan-no-drub-spam@hotmail.com)> wrote in message  
news:%23unObHayEHA.1300@TK2MSFTNGP14.phx.gbl...

>

> "Alex K. Angelopoulos [MVP]" <[aka-at-mvps-dot-org](mailto:aka-at-mvps-dot-org)> wrote in message  
> news:u%23JtYcIyEHA.4060@TK2MSFTNGP10.phx.gbl...

>> Al Dunbar [MS-MVP] wrote:

>>> "Alex K. Angelopoulos [MVP]" <[aka-at-mvps-dot-org](mailto:aka-at-mvps-dot-org)> wrote in message  
>>> news:euxP4b7xEHA.3844@TK2MSFTNGP09.phx.gbl...

>>

>>

>>>> *But it is. A resource element can hold any text whatsoever, but I had  
>>>> never considered using it for localization.*

>>>

>>> *That's interesting, as both references (the docs "Script56.CHM" and  
Dino*

>>> *Esposito's WSH Programmers Guide) give localization as their only  
>>> practical example of its use.*

>>

>> *That's hilarious.*

>

> *What's hilarious? That the above is said about localization and you  
haven't*

> *used it for that purpose, or that the books have failed to mention some of  
> your uses below?*

>

> *Reminds me of the dictionary object a bit in that regard...*

>

>> *I've used resources for:*

>> + *storing literal html or code fragments I wanted to use somewhere  
without*

>> + *mangling them inline;*

>> + *as "targets" for dynamic reconfiguration of the above;*

>> + *storing special extended help information;*

>> + *and storing initialization/state data.*

>>

>> *When I saw the original post, I thought it was a brilliant new idea. :)*

>>

>>  
>>> *My organization is bilingual, and adjusting internally developed*  
>>> *applications written in one language to run as effectively in the*  
*other*  
>>> *language can be quite a chore...*  
>>  
>> *You mentioned people who are language, not programming subject experts*  
*doing*  
>> *the modifications (and I accidentally snipped it). One irritating aspect*  
*of*  
>> *this is that it puts script code at risk for accidental breakage since*  
*we*  
>> *don't have a simple string table externalization mechanism.*  
>  
> *Presumably, they do not work on the code, per se, but supply their*  
> *translation text by hard copy or in a document. Then people less likely to*  
> *break the code do the actual editing.*  
>  
>>> *1) assigning meaningful ID's to trivial strings, especially when there*  
>>> *are many, and then being able to work back and forth between the*  
>>> *content of the string and its context in the code.*  
>>  
>> *In theory the way it "should" work is that you use any symbolic name you*  
>> *want since it doesn't need to be localized –*  
> *Aunt\_pen\_wood\_furniture\_example*  
> *could be a decent resource name.*  
>  
> *Understood. My problem is with managing the creation of a whole bunch of*  
> *these things in such a way that the code is still more or less reasonable.*  
> *For this, the symbolic names would need to be detailed enough, but not too*  
> *detailed...*  
>  
>>> *2) dealing with strings that combine literal text with variable*  
*values:*  
>>> *and then use a function to return the string in the correct language*  
*and*  
>>> *with all tokens replaced. This, of course, creates yet another issue*  
*of*  
>>> *creating a global set of tokens that are consistent with variable*  
>>> *values in the code.*  
>>  
>> *Nothing that works in WSFs/WSCs, unfortunately. With 'real' XML, you can*  
>> *predefine entities in your XML (or in an external file), then use them*  
>> *throughout your text and expect proper substitution. If we were talking*  
>> *absolute names – e.g. a brand name of some kind – your examples would*  
*work*  
>> *with something like*  
>>  
>>> *"My Aunt's [[writingdevice]] is on the [[woodtype]]*  
>>> *[[furnituredevice]]"*  
>>>

```
> > > "La [[writingdevice]] de ma tant est dans le [[furnituredevice]]  
> > > [[woodtype]]"  
> > <!ENTITY writingdevice "Pilot UniBall">  
> >  
> > and then having literal text elements with correct locale tags that look  
> > like this:  
> > My Aunt's &writingdevice; is on the...  
> > La &writingdevice; de ma tant est dans le ...
```

Sorry, had to leave in a hurry and thought I had cancelled my post. Anyway, continuing on....

```
> Close. What I have come up with is a class with an overloaded property:  
>  
> thepen = "ballpoint pen"  
    thetable = "kitchen table"  
> XXX.text = array( _  
> "writing device", thepen, _  
> "furniture device", thetable )
```

```
XXX.text = "Aunt pen wood furniture example"
```

```
substitutedvalue = XXX.text
```

Two resources are defined:

"EN Aunt pen wood furniture example" contains: "My aunt's <writing device> is on the <furniture device>"

"FR Aunt pen wood furniture example" contains: "Le <writing device> de ma tant est sur le <furniture device>".

The text property code would use getlocale to determine which resource to use, the English or the French. The technique of substituting tokenized values is of great help in getting around the fact that different languages have different sentence structure, such that these variable values may not always appear in the same order.

Of course, this is a poor example for the management of bilingual code because the substitution values are given in English. More likely the substitution values would be the name of an account, the date, or some other information that would be the same in either language.

Since generating this class (and thanks to the OP for getting me to consider this again) I have found other reasons to justify the use of getresource. But the overriding one is that it is much easier to generate (and maintain!) a block of text literally in a resource field than with the "equivalent" vbscript code:

```
msgtext = "My aunt's " & writingDevice _  
    & " is on the " & furnitureDevice
```

microsoft.public.scripting.wsh: Re: <resource> elements in WSCs

or:

```
msgtext = " My Aunt's "  
msgtext = msgtext & writingDevice  
msgtext = msgtext & " is on the "  
msgtext = msgtext & furnitureDevice
```

/Al