Sons of Thunder Software
SOFTWARE DEVELOPMENT AND CONSULTING SERVICES

Products Services Developer Resources Contact STS About STS

livecode

Creating and Maintaining Data Storage Stacks

>> Keep your data separate from the UI. This way you can update the UI
>> independently of the data.
>
> Thanks for the fast reply, Scott.
>
> But this would require storing it in a text file and managing all the
> data manipulation stuff or alternatively using a database, right? If
> I have to do those things to keep data from being munged when I
> upgrade the app, I might just as well use a conventional programming
> environment! Transparency of data storage is crucial to stackware
> applications.

The UNIX and Windows restriction of executables being unable to write to themselves (Rev carries this forward to Mac OS as well for consistency) does require some re-thinking of data storage if you're used to working in Mac-specific xTalks.

Fortunately MC/Rev makes it easy:

You could store your data in an external text file, but then you'd have to parse it to have any useful structure to it.

If the nature of your data suggests a need for more structure than a single block of text, you could use a separate stack file for data storage.

The main benefit of using a stack file as opposed to a text file is the hierarchical structure afforded with objects and their custom properties.

Remember that every Rev object can have custom properties, and even multiple sets of custom properties. Accessing these is very fast — much faster than fields, and only a tiny bit slower than globals. You can use array notation if you like, and they can store binary data as well as text.

Since a new stack file contains one card in one mainstack, you instantly get this rich structure for organizing your data (for just 363 bytes of overhead):

```
                      stack
                        |
        ------------------------------------
        |               |              |
  custom property    cust. prop. set   |
                        |              |
                     custom property   |
                                       |
                                      card
                                       |
                        ---------------------------
                        |                          |
                  custom property          cust. prop. set
                                                   |
                                           custom property
```

Of course you can enrich this data hierarchy by adding more cards to the stack, adding objects to the cards, adding substacks, etc. With multiple property sets, every object inherently has a two-dimensional data space; with multiple objects in nested groups across multiple cards in mutiple substacks, you can effectively create an n-dimensional data space in one stack file. Using the substacks, cardNames, customKeys, and customPropertySets properties you can instantly obtain lists of any collection of members at any level of the hierarchy.

If you need a rich hierarchy with a lot of objects when the user creates a new data file, you can have a template data stack in your app's stack file, use the clone command to make a copy in memory, and set the filename propery of the newly-cloned data stack to assign it a file path; when you issue a save command for that stack, it'll save to the assigned path.

Because the storage space and access speed overhead are so low with custom properties, you can use stack files for data sets as large as memory allows and expect pretty snappy performance. In this sense, it rivals Panorama and other RAM-based databases, with all the ease and flexibility of an xTalk.

A simple example of using custom props for data storage:

```
on StoreData
  repeat with i = 1 to the number of flds
    set the UserData[i] of stack "MyDataStack.rev" to fld i
  end repeat
end StoreData

on DisplayData
  lock screen
  repeat with i = 1 to the number of flds
    put the UserData[i] of stack "MyDataStack.rev" into fld i
  end repeat
```

```
    unlock screen
  end DisplayData
```

Tip: if your data file needs to store images, just copy them to the card(s) of your data stack, and use buttons in your app's UI stacks to dispay those images by setting the button's icon. Unlike most other xTalks, Rev icons have no practical limit on size, and using this method allows a single image object to be displayed in multiple button objects with minimal RAM overhead.

Another tip: If you want to store data in a way that isn't obvious for someone to read if they open the stack file in a binary editor, you can use the compress and decompress functions on data you store in custom props, which will also reduce the size of the data file.

Richard Gaskin
Fourth World Media Corporation
Custom Software and Web Development for All Major Platforms
Developer of WebMerge 2.0: Publish any Database on Any Site

Ambassador@FourthWorld.com http://www.FourthWorld.com
Tel: 323-225-3717 AIM: FourthWorldInc

Posted 7/12/2002 to the Use-Revolution List *(See the complete thread)*

Print this tip

News and Rumors Products Services Developer Resources Contact STS About STS