3PixelRule_Gray.tiff ¬

Jim Million   uunet!wiltel!jmillion   October 1992

WrapperInspector (v1.1)

WrapperInspector is a custom WorkspaceManager Contents Inspector.   It registers with WM to manage inspection of 'wrapped' (also refered to as 'bundled') WM files.   Bundles appear as files in WM when in fact, they are actually directories.   For example, .app, .bundle, .rtfd are bundled files.   Normally, you have to 'Open As Folder' to see into these wrappers.

WrapperInspector displays a bundle's contents in a browser.   You may select entries in this browser to 'sub' inspect (for example, you may view .tiff images, or play .snd files).   You may also 'Open' the file with its default application, and/or 'Open As Folder' if it is a directory.   You may extend WrapperInspector by registering other wrapper types with WM, creating new subinspectors to handle other file types, and registering additional file types with existing subinspectors.

Revision History

v1.1

The following (Noted in 1.0 Readme) has been partially fixed. The 'Open' and 'Open As Folder' methods produce inconsistent performance behavior.   These operations takes either a few seconds (usually true), or at least 45 seconds (seems to be some time-out value).   This is understandable as it seems WM is contending for its own resources.   WrapperInspector uses the 3.0 object 'workspace' (WorkspaceRequest   protocol) in performing these.   Optimization, delayed performs, and 2.1 Speaker/Listener style were tried, but did not help (and were taken out to keep code simple).   'Open' has been fixed with a workaround.   There is no complete fix for the 'Open As Folder' operation.

876954_PixelRule_Gray.tiff ¬

Important Files...

paste.tiff ¬ bundle.registy contains registration info (tells WM which files this inspector will inspect).

74015_paste.tiff ¬ Makefile.preamble loads bundle.registry info into the _ICON section of the Mach-o file.

391187_paste.tiff ¬ WrapperInspector.[hm] is the owner of the bundle nib file (WrapperInspector.nib). It

    manages inspection and coordinates 'sub' inspection of the registered files.

391017_paste.tiff ¬ DefaultSubInspector.[hm] is an abstract superclass provided for all subinspectors.

    Subinspectors provide simple inspection for specified file types found in the wrapper.

188764_paste.tiff ¬ ImageSubInspector.[hm] inspects image files.

225814_paste.tiff ¬ TextSubInspector.[hm] inspects specified text files.

424989_paste.tiff ¬ SoundSubInspector.[hm] inspects sound files.

972586_paste.tiff ¬ WrapperInspector.nib is the interface file for the main and subinspectors.

317905_paste.tiff ¬ Inspector.strings contains key/value pairs where key equal file extension and value

    equal the class name of the subinspector class responsible for managing subinspection.

Installation

WrapperInspector overrides WM's default contents inspection which allows you to 'sort' the contents of the selected wrapper.

To install, copy WrapperInspector.bundle into ~Apps (or anywhere in WM's search path - NextApps, LocalApps, etc.).   WM will look into the bundle when you do the copy, and register the specified file types (extensions) at that time (first load takes a moment).

To unload, remove the bundle from the search path, log out.   Upon next login, bundle will not be loaded.

To reload, replace the existing bundle, log out. New bundle will be loaded upon next login.

Registered Wrappers

The following extensions are currently registered for inspection:

app	palette	font	nib
bundle	addresses clr		mbox
preferences	pkg	service	dbmodel

To add an extension:
1] Append the following line to bundle.registry:
   {type=InspectorCommand; mode=contents; extension=**YourExtension**;
   selp=selectionOneOnly; class=WrapperInspector}
2] Rebuild project (PB.project found in the WrapperInspector folder).
3] Install WrapperInspector.bundle.

To delete an extension:
1] Remove the line containing the unwanted extension from bundle.registry.
2] Rebuild project (PB.project found in the WrapperInspector folder).
3] Install WrapperInspector.bundle.

Adding A New SubInspector

1] Create a subclass of DefaultSubInspector (Suggest naming as [Type]SubInspector. I
   originally used ImageInspector and SoundInspector and found these already existed
   within WM, hence use of 'Sub'Inspector.).   Your subinspector should override
   DefaultSubInspector where specific behavior is required (see existing for example).
2] Add [hm] to ProjectBuilder (project is in WrapperInspector folder - PB.project).
3] Open WrapperInspector.nib (in English.lproj), parse interface file into IB, and
   'instantiate' an instance of your subclass.
4] Create an interface contained within a Box or Window (see existing for size, etc.).
5] Connect your subinspector's 'inspectorView' instance variable to the Box or Window
   instance containing your interface components.   The 'contentView' of this instance
   variable is what is returned to WrapperInspector in the method 'inspectorView' and
   subsequently displayed.
6] Register the file types (extensions) that your subinspector will inspect.   Do this in IB by
   inspecting the 'Attributes' of the NXStringTable instance (InspectorStrings).   The key is
   is the file extension.   The value is the classname of your subinspector.   Add as many

entries as you wish.   Save these changes to the file Inspector.strings (in English.lproj).
(Answer Yes to 'Do you want to Replace').
7]   Rebuild Project and install.

To add or delete filetypes registered for subinspection, perform step 6 and 7.

How WrapperInspector interacts with subinspectors...
When a file is selected in WrapperInspector's browser, it tries to find a match for the file's extension in the NXStringTable (containing extensions/classnames).   If a match is found, a look-up is performed in the run-time for the classname.   If found, the class method 'new' is sent to the Class id.   If an instance is returned, the instance is sent the message 'inspectorView' which returns the View that WrapperInspector displays.   The instance is then sent the message 'inspect:' with the full pathname of the selection as the argument.   Your subinspector then performs the appropriate inspection.

Notes...

222474_paste.tiff ¬ The TextSubInspector display behaves incorrectly at times. This seems to occur only
occasionally when scrolling large (usually rtf[d]) documents.   Changing window
buffering did not seem to help.

797099_paste.tiff ¬ It is unfortunate that you cannot register with WM for files that do not have extensions
(e.g. Makefile).   It is also unfortunate that I perpetuated this with the subinspectors.   If
you desire other behavior, alter the method 'fileSelected:' in WrapperInspector.m.

64463_paste.tiff ¬ RTFD bundles were not registered.

667372_paste.tiff ¬ The WrapperInspector browser only allows single selection.

Documentation

NeXT documentation on custom WorkspaceManager Contents Inspectors...
GeneralRef/19_WorkspaceManager/IntroWorkspace.rtf
GeneralRef/19_WorkspaceManager/Classes/WMInspector.rtf