WorkFlows

Introduction

In the WorkLife FrameWork™ `WorkFlows` are a way to create a set of *Mathematica* functions that are executed when the WorkFlow itself is executed.

A WorkFlow can be executed programmatically or from the WorkFlows Palette, and a given workflow can be executed in a variety of ways.

WorkFlows are useful in wide a variety of ways. One simple application is to create custom setups for the WorkLife FrameWork™'s Palettes. In the following section we will show an example of this and, in doing so, give a description of how a `WorkFlow` functions.

Example

For the buttons and executable commands that are described n this section to work it is assumed that you have installed the WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

A `WorkFlow` is created with the function `CreateWorkFlow`.

CreateWorkFlow[name, {"flowname1":→flow1, "flowname2":→flow2,... }] creates a WorkFlow with
given rules. The rules must be in the form of Delayed Rules. If a previous WorkFlow was created
replaced with the new one.

Usage Message for `CreateWorkFlow`

WorkFlows that are created in this way are automatically saved to a file and are reloaded whenever you load the WorkLife FrameWork™ in a *Mathematica* session.

When a `WorkFlow` has been created it can be executed with the function `WorkFlow`.

WorkFlow[name,{"flownameA","flownameB",...}] executes the WorkFlow with the given name with
order. WorkFlow[name] gives the list of delayed rules corresponding to the WorkFlow. WorkFlow
workflow in the default order (the order of the rules given in the WorkFlow's original definition: i.e
WorkFlow[name]. WorkFlow[name,Names] gives the list of the Names of the WorkFlow elements

Usage Message for `WorkFlow`

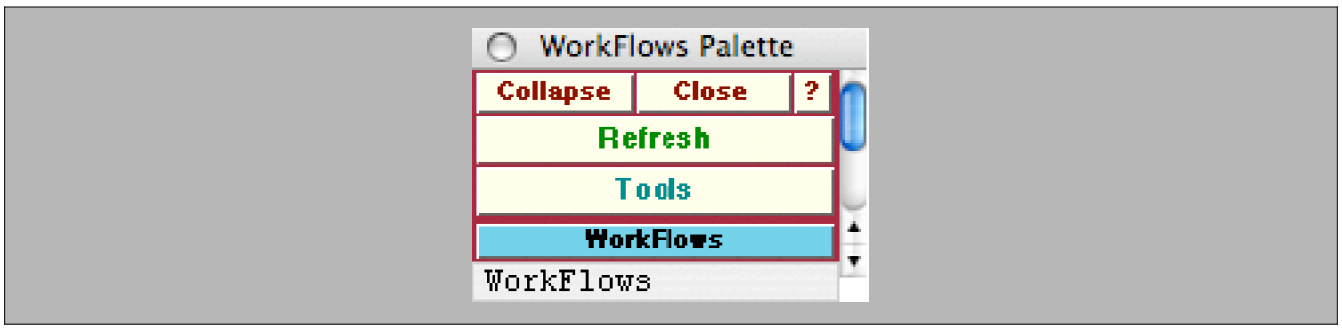
What a `CreateWorkFlow` does is associate a set of executable *Mathematica* statements with a set of strings. Then, with the function `WorkFlow` you can evaluate those statements in the order in which their associated strings appear in a list.

As a simple example we create a WorkFlow that closes all of the open palettes and then opens the Notebooks Palette, the Blog Tools Palette, and the Formatting Palette. The WorkFlow will be assigned to the parameter `paletteSetup` as in the following:

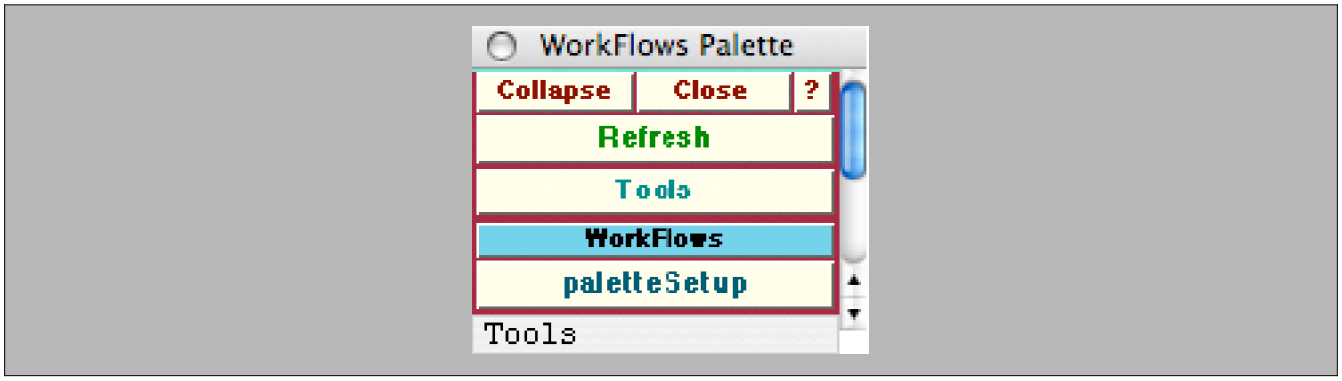
```
CreateWorkFlow[paletteSetup,
 {"ClosePalettes" :> CloseAllPalettes[LeaveOpen → None],
  "Notebooks" :> NotebooksPalette[],
  "BlogTools" :> BlogPalette[],
  "Formatting" :> FormattingPalette[]}]
```

```
{ClosePalettes :> CloseAllPalettes[LeaveOpen → None], Notebooks :> NotebooksPalette[],
 BlogTools :> BlogPalette[], Formatting :> FormattingPalette[]}
```

Before this was executed the WorkFlows Palette looked like (assuming that there were no previous WorkFlows defined):



And after the `CreateWorkFlow` expression is executed it looks like:



To execute this `WorkFlow` all that is needed is to click on the paletteSetup button in this Palette. This will then close all of the Palettes that you currently have open and replace them with the ones that this `WorkFlow` opens.

Another way to do the same thing by executing a function is to execute,

```
WorkFlow[paletteSetup, Default];
```

Let us say that you only wanted to open the Blog Tools Palette and the Formatting Palette after closing all of the open Palettes. Then you could execute a custom version of this WorkFlow like this:

```
WorkFlow[paletteSetup, {"ClosePalettes", "BlogTools", "Formatting"}];
```

💡 Note that the WorkFlow executes the expressions corresponding to the strings from left to right.

So, for example, executing,

```
WorkFlow[paletteSetup, {"BlogTools", "ClosePalettes", "Formatting"}];
```

will leave only the Formatting Palette open when the execution is completed.

Other WorkFlow Functions

> For the buttons and executable commands that are described n this section to work it is assumed that you have installed the WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

A list of the currently available WorkFlows is given by the function WorkFlows:

```
WorkFlows[]
```

```
{paletteSetup}
```

Note that the elements of this list are Strings, even though the parameter that a WorkFlow is attached to is not a string.

```
WorkFlows[] // FullForm
```

```
List["paletteSetup"]
```

To see the list of strings—tags—that you assigned to the *Mathematica* expressions in your WorkFlow you execute

```
WorkFlow[paletteSetup, Names]
```

```
{ClosePalettes, Notebooks, BlogTools, Formatting}
```

And to see the set of Rules that you specified for these, execute the single argument form of WorkFlow for the WorkFlow that you are interested in:

```
WorkFlow[paletteSetup]
```

```
{ClosePalettes :↦ CloseAllPalettes[LeaveOpen → None], Notebooks :↦ NotebooksPalette[],
 BlogTools :↦ BlogPalette[], Formatting :↦ FormattingPalette[]}
```

To remove a WorkFlow that you have previously designed and implemented with CreateWorkFlow, use the function ClearWorkFlow:

```
NotebookWrite[EvaluationNotebook[], UsageCellList[ClearWorkFlow]];
```

> ClearWorkFlow[name] clears the WorkFlow with the given name.
>
> Usage Message for `ClearWorkFlow`

So, to clear the WorkFlow that we defined in the previous section (and also remove it from the WorkFlows Palette) execute

```
ClearWorkFlow[paletteSetup]
```

So, the WorkFlow that we created earlier is gone:

```
WorkFlows[]
```

```
{}
```