MiscTeePalette

An Overview Of Things

Have you ever noticed that while you can show off IB's power by connecting a TextField and a Slider and having them coordinate without writing code, you can't do that **and** have the data also be sent to a controlling object? Well that can now be taken care of by using a MiscSliderField **or** by connecting all three objects using one MiscTee (which provides a far more general solution than the MiscSliderField).

This palette provides one object class (MiscTee) for dragging into your File windows in IB. It is then available for general spreading of messages. It can be used in two ways.

As a multiplier of target/action messaging, you simple connect the MiscTee as the target with the **ping:** action. You then connect the MiscTee to as many other objects as you like giving each of them their own action. When **ping:** is sent, all the other objects will get their respective messages from the original sender as if they were the target.

As a simpler splitter, you can just connect objects without any action. These objects will be forwarded any messages that the MiscTee doesn't understand (which is the Object methods and the very few MiscTee methods). This can be used to set up several objects as delegates to the same object, each providing different functionality. The message is only forwarded to the first object the MiscTee finds that responds to the message. Unfortunately, I don't think there is an actual definition of ªfirstº when connections are made with IB, so don't count on certain behavior if two delegates respond to the same message. If you set up the connections with code, the ªfirstº object it finds is the first one connected.

Unfortunately, the above problem with connection order causes a slightly bigger problem. Since connection order is indeterminate it is entirely possible (and likely) that the connection from the host object to the MiscTee will be made before the MiscTee is connected to the delegates. This causes problems when the host object wants to talk to its delegate before all the connections are made. For example, it isn't reliably possible to connect an NXBrowser to a MiscTee for delegation splitting because somewhere before all the connections are made the NXBrowser asks its delegate (which it thinks is the MiscTee) to fill the cells. This can't happen because the MiscTee isn't connected to anything it can pass the messages on to for handling. My suggestion is to code the connections someplace where you can control the order of events more specifically. If you come up with a way to solve this problem, or a consistent way to deal with it easily, let me know so I can implement it or at least mention an easy work around.

Some Other Details

The connection inspector tries its hardest to work just like the regular IB inspector while also allowing an arbitrary number of connections. Since it can only find out what actions are available by searching the runtime structures for methods that have one argument of type 'id' it will find some methods that aren't real actions. There is a list of these in the NonActions.txt file kept in the .palette wrapper. Feel free to add to it as you need to, it's parsed anew every time IB starts up. There is a related file, ValidActions.txt, which are methods that should not be put on the list as they are actual actions for an object somewhere. This is merely a reference for any editing of the other file.

I use a StringList object in the project. I have set it up in the .palette so that it only loads the class if it doesn't already exist. This allows you to load the MiscTeePalette with or without the StringList palette loaded with only one condition ± if you load the StringList palette, you must load it before the MiscTeePalette because StringList doesn't do the same checking.

There is an NXBrowserCell subclass that people might be interested in. It allows the cells to have the various states that the IB connection cells have. A leaf can be blank or have a dimple (or other image) that sits in the right end of the cell aligned with the branch arrow. It is, for that reason, named MiscDimpleCell.

I tried terribly hard to get the actual names of the objects (like what shows under the icons in the file window) in IB for the connection display but was unable to find any combination of -name and related methods that would give me anything but the class. If you figure out a way to find out that information, please tell me so I can incorporate it ± I can imagine it being pretty frustrating to have to figure out which TextField you've connected the MiscTee

to when there's a window full of them.