MiscAnnouncer

Inherits From: Object

Conforms To: MiscDependency

Declared In: misckit/MiscAnnouncer.h

Class Description

MiscAnnouncer is a public version of the IKAnnouncer class found in IconKit. It provides a simple way to conform to the MiscDependency protocol and lets an object broadcast messages to its users and listeners. It also provides a polling mechanism to ensure that users approve of a pending action.

The simplest way for an object to use a MiscAnnouncer is to create one in an instance variable, and then forward all MiscDependency messages to it. Here is a skeleton example of a class *Foo* that uses MiscAnnouncer. A *Foo* sends out an announcement whenever its name changes.

```
@interface Foo : Object <MiscDepenendency>
{
```

```
    MiscAnnouncer
        * announcer;
    char
        * name;
    ...
}

@end


@implementation Foo


- init
{
    ...
    announcer = [[MiscAnnouncer  alloc]  initOwner: self];
    name = NULL;
    ...
}


- addUser: who        {   return [announcer  addUser: who];          }
- addListener: who    {   return [announcer  addListener: who];      }
- removeUser: who     {   return [announcer  removeListener: who];   }
- removeListener: who {   return [announcer  removeListener: who];   }


- setName: (const char *) theName
{
    if (name != NULL) free(name);
    name = theName ? NXCopyStringBuffer(theName) : NULL;
    [announcer  announce: @selector(didChangeName:)];

    return self;
```

```
}

@end
```

A class might do more than this example suggests. For example, a *Foo* could do reference counting garbage collection by checking in **removeUser:** whether any object still refers to it.

Instance Variables

id **owner**;
id **usersAndListeners**;
int **numUsers**;
BOOL **sendAnnouncements**;

owner	The object from which announcements and polling messages appear to originate.
usersAndListeners	The list of all users and listeners.
numUsers	The number of users.
sendAnnouncements	True if the MiscAnnouncer should send out announcements.

Method Types

Initialization and freeing	- initOwner: - free
Users and listeners	- addUser: - addListener: - removeUser: - removeListener: - numUsers - usersAndListeners
Announcements	- announce: - announce:with: - poll: - poll:with: - sendAnnouncements - setSendAnnouncements:

Instance Methods

addUser:

- **addUser:** *who*

Adds a new user to the MiscAnnouncer. Users will receive notification of changes to the MiscAnnouncer's owner via the **announce:** methods. If the program has been run with **-MiscAnnounceDebug** specified on the command line, then this method prints out an acknowledgement to stderr.

See also: - **removeUser:**, - **addUser:** (MiscDependency)

addListener:

- **addListener:** *who*

Adds a new listener to the MiscAnnouncer. Listeners will receive notification of changes to the MiscAnnouncer's owner via the **announce:** methods. If the program has been run with **-MiscAnnounceDebug** specified on the command line, then this method prints out an acknowledgement to stderr.

See also: - **removeListener:**, - **addListener:** (MiscDependency)

announce:

- **announce:**(SEL)*theMessage*

Broadcasts *theMessage* to all users and listeners of the MiscAnnouncer that can respond to it; users and listeners that do not implement *theMessage* do not receive it. The broadcast message should take exactly one argument, e.g. **didChangeName:**, which will contain the owner of the MiscAnnouncer. Receiving objects should register with either an **addUser:** or **addListener:** and implement the following:

```
- didChangeName: sender
{
    /* sender just changed its name */
    return self;
}
```

Announcements are only sent if the **sendAnnouncements** variable is True. Users only need to implement methods for the particular messages they care about.

See also: - **announce:with:**, - **poll:**, - **addUser:**, - **addListener:**

announce:with:

- **announce:**(SEL)*theMessage*
  with: *theArgument*

Like **announce:**, except that it provides an additional argument to go along with *theMessage*. For example, a list might send out the announcement **list:didAdd:** after it inserts an object. Receiving objects should implement something like,

```
- list: sender  didAdd: anObject
{
    /* sender just added anObject */
    return self;
}
```

Announcements are only sent out if the **sendAnnouncements** variable is True.

See also: - **announce:**, - **poll:with:**, - **addUser:**, - **addListener:**

free

- **free**

Frees the MiscAnnouncer. This method does not send out any announcements.

initOwner:

- **initOwner:** *theOwner*

Initializes a new MiscAnnouncer with the given owner. This method is the only way to set a MiscAnnouncer's owner.

numUsers

- (int)**numUsers**

Returns the current number of users registered with the MiscAnnouncer. This number will be less than the length of **usersAndListeners** if any objects have registered as listeners.

See also: - **usersAndListeners**

poll:

- (BOOL)**poll:**(SEL)*theMessage*

Conducts a poll of all users and listeners that can respond to *theMessage*. The result is the logical AND of all their responses. Just like C code, the AND evaluation terminates as soon as a single object responds NO, so some objects might never see the polling message.

Polling messages are always sent out, regardless of the current state of **sendAnnouncements**.

The syntax and usage is identical to **announce:**, with the one exception that *theMessage* should be declared to return a BOOL instead of an **id**.

See also: - **poll:with:**, - **announce:**

poll:with:

- (BOOL)**poll:**(SEL)*theMessage*
  with: *theArgument*

Like **poll:**, except that it provides an additional argument to go along with *theMessage*. The syntax and usage are identical to **announce:with:**, with the one exception that *theMessage* should be declared to return a BOOL instead of an **id**.

See also: - **poll:**, - **announce:with:**

removeUser:

- **removeUser:** *who*

Removes a user from the MiscAnnouncer. If the program has been run with **-MiscAnnounceDebug** specified on the command line, then this method prints out an acknowledgement to stderr. Classes that use a MiscAnnouncer can intercept this message to do reference counting garbage collection.

See also: - **addUser:**, - **removeUser:** (MiscDependency)

removeListener:

- **removeListener:** *who*

Removes a listener from the MiscAnnouncer. If the program has been run with **-MiscAnnounceDebug** specified on the command line, then this method prints out an acknowledgement to stderr.

See also: - **addListener:**, - **removeListener:** (MiscDependency)

sendAnnouncements

- (BOOL)**sendAnnouncements**

Returns YES if announcements are enabled, NO otherwise.

See also: - **setSendAnnouncements:**

setSendAnnouncements:

- **setSendAnnouncements:**(BOOL)*flag*

Enables or disables announcements. Returns **self**.

See also: - **sendAnnouncements**

usersAndListeners

- **usersAndListeners**

Returns the list of users and listeners for the MiscAnnouncer.

See also: - **numUsers**