```
//ÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐ
//  FILENAME:   eTLinkWell.m
//  SUMMARY:    Defines a well for drag-and-drop of .etfLink files.
//  SUPERCLASS:Object:Resonder:View:Control:Button:eTLinkWell
//  PROTOCOLS: None
//  INTERFACE: None. Instantiate as a customView in IB.
//  AUTHOR:     Rohit Khare
//  COPYRIGHT: ©1993,94 California Institure of Technology, eText Project
//ÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐ
//  Implementation Comments
//      This as gross a hack as they come. There, I said it. Ok?
//  This is a true perversion of target-action... but it works.
//ÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐ
//  HISTORY
//  10/04/94:  Revamped for eText5
//ÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐÐ
//  Imported Interfaces
//
    #import "eTLinkWell.h"

@implementation eTLinkWell
// Dragging URL-Squiggle support!//  For those fine guys at OmniGroup :)
- initOmniWell
{
    [self registerForDraggedTypes:&NXAsciiPboardType count:1];
    [self setEnabled:NO];
    [self setType:NX_PUSHONPUSHOFF];
    [self setBordered:NO];
    [self setIcon:EMPTY];
    [self setAltIcon: LIT];
    [self setTitle:"Drag an OmniWeb URL here:"];
    [self setIconPosition:NX_ICONRIGHT];
    omniMode = YES;
    return self;
}
- initLinkWell
{
    const char *fileType[] = {NXFilenamePboardType, NXFileContentsPboardType};
    [self registerForDraggedTypes:fileType count:2];
    currentLink.docID = NULL;
    [self setEnabled:NO];
    [self setIconPosition:NX_ICONONLY];
    [self setType:NX_PUSHONPUSHOFF];
    [self setBordered:NO];
    [self setIcon:EMPTY];
    [self setAltIcon: LIT];
    omniMode = NO;
```

```
    return self;
}

- setLink:(etfLink *)theLink
{
    if (theLink == NULL) {
        // clear out to the default image
        [self setIcon:EMPTY];
        currentLink.docID = NULL;
    } else {
        // display "occupied" image
        [self setIcon:FULL];
        currentLink = *theLink;
    }
    return self;
}

- (etfLink *) currentLink {return &currentLink;}

- (NXDragOperation) draggingEntered:sender
{
    NXAtom  types[3];
    id      thePB;

    if (![[self target] respondsTo:[self action]]) return NX_DragOperationNone;
    if ([sender draggingSource] == self) return NX_DragOperationNone;
    if (omniMode) {
        return NX_DragOperationGeneric;
    }
    thePB = [sender draggingPasteboard];
    types[0] = NXFilenamePboardType;
    types[1] = NXCreateFileContentsPboardType(LINK_EXT);
    types[2] = NULL;
    types[2] = [thePB findAvailableTypeFrom:types num:2];
    if (!types[2]) {
        return NX_DragOperationNone;
    } else if (types[2] == types[0]) {
        char    *data;
        int     len,i=0,j=0;
        BOOL    found=NO;

        [thePB readType:NXFilenamePboardType data:&data length:&len];
        while ((j < len) && !found){
            i = j;
            while (data[i] && (data[i] != '\t')) i++;
            data[i] = 0;
            if (!strcmp(LINK_EXT, rindex(data+j, '.')+1)) found = YES;
```

```
            else if (!strcmp(ETFD_EXT, rindex(data+j, '.')+1)) found = YES;
            j = i+1;
        }
        [thePB deallocatePasteboardData:data length:len];
        if (!found) return NX_DragOperationNone;
    }
    [self setState:1];
    return NX_DragOperationLink;
}
- draggingExited:sender {return [self setState:0];}
- (BOOL)prepareForDragOperation:sender {return YES;}
- (BOOL)performDragOperation:sender
{
    id theTable;
    NXAtom  types[3];
    id      thePB;
    BOOL    found=NO;

    thePB = [sender draggingPasteboard];
    if (omniMode) {
        [omniText setSel:0:[omniText textLength]];
        [omniText readSelectionFromPasteboard:thePB];
        [[self target] perform:[self action] with:self];
        return YES;
    }
    types[0] = NXFilenamePboardType;
    types[1] = NXCreateFileContentsPboardType(LINK_EXT);
    types[2] = [thePB findAvailableTypeFrom:types num:2];

    if (!types[2]) return NO;
    else if (types[2] == types[0]) {
        char    *data;
        int     len,i=0,j=0;

        [thePB readType:NXFilenamePboardType data:&data length:&len];
        while ((j < len) && !found){
            i = j;
            while (data[i] && (data[i] != '\t')) i++;
            data[i] = 0;
            theTable = nil;
            if (!strcmp(LINK_EXT, rindex(data+j, '.')+1)){
                // read this file in and issue the accept...:
                theTable = [[NXStringTable alloc] init];
                [theTable readFromFile:data+j];
            } else if (!strcmp(ETFD_EXT, rindex(data+j, '.')+1)) {
                // find a docID for this file, stash it away
                theTable=[[[eTDocInfo alloc] init] readComponentFromPath:data+j];
```

```
        }
        if (theTable) {
            currentLink.docID =
                NXUniqueString([theTable valueForStringKey:DOCID]);
            currentLink.anchorID =
                NXUniqueString([theTable valueForStringKey:ANCHORID]);
            currentLink.docTitle =
                NXUniqueString([theTable valueForStringKey:DOCTITLE]);
            currentLink.anchorTitle =
            NXUniqueString([theTable valueForStringKey:ANCHORTITLE]);
            if (currentLink.anchorID == NULL)
                currentLink.anchorID = NXUniqueString("0");
            if (currentLink.anchorTitle == NULL)
                currentLink.anchorTitle = NXUniqueString("Untitled");
            [[self target] perform:[self action] with:self];
            found = YES;
            if (!strcmp(LINK_EXT, rindex(data+j, '.')+1)){
                theTable = [[theTable empty] free];
                if (!strncmp(data+j, "/tmp", 4) &&
                   ((rindex(data+j, '/')-(data+j)) == 5))
                   unlink(data+j);
            }
        }
        j = i+1;
    }
    [thePB deallocatePasteboardData:data length:len];
} else if (types[2] == types[1]) {
    sprintf(path, "/tmp/%d.etfLink", [NXApp uniqueID]);
    free([thePB readFileContentsType:
        NXCreateFileContentsPboardType(LINK_EXT) toFile:path]);
    // "... returns an allocated string..."
    // read this file in and issue the accept...:
    theTable = [[NXStringTable alloc] init];
    if ([theTable readFromFile:path]) {
        currentLink.docID =
            NXUniqueString([theTable valueForStringKey:DOCID]);
        currentLink.anchorID =
            NXUniqueString([theTable valueForStringKey:ANCHORID]);
        currentLink.docTitle =
            NXUniqueString([theTable valueForStringKey:DOCTITLE]);
        currentLink.anchorTitle =
            NXUniqueString([theTable valueForStringKey:ANCHORTITLE]);
        if (currentLink.anchorID == NULL)
            currentLink.anchorID = NXUniqueString("0");
        if (currentLink.anchorTitle == NULL)
            currentLink.anchorTitle = NXUniqueString("Untitled");
        [[self target] perform:[self action] with:self];
```

```
            found = YES;
        }
        theTable = [[theTable empty] free];
        unlink(path);
    }
    [self setState:0];
    return found;
}

- (NXDragOperation)draggingSourceOperationMaskForLocal:(BOOL)isLocal
{
    return (isLocal ? NX_DragOperationLink : NX_DragOperationAll);
}
- (BOOL)acceptsFirstMouse {return YES;}
- (BOOL)shouldDelayWindowOrderingForEvent:(NXEvent *)e
{
    return (currentLink.docID ? YES : NO);
}
- mouseDown:(NXEvent *)e
{
    id      theTable;
    NXPoint org;
    NXPoint offset;
    id      dragPasteboard;
    NXAtom  types[3];

    if (currentLink.docID == NULL) return self;
    theTable = [[NXStringTable alloc] init];
    [theTable insertKey:DOCID value:NXCopyStringBuffer(currentLink.docID)];
    [theTable insertKey:ANCHORID value:NXCopyStringBuffer(currentLink.anchorID)];
    [theTable insertKey:DOCTITLE value:NXCopyStringBuffer(currentLink.docTitle)];
    [theTable insertKey:ANCHORTITLE value:
NXCopyStringBuffer(currentLink.anchorTitle)];
    sprintf(path, "/tmp/%s.etfLink", currentLink.anchorTitle);
    [theTable writeToFile:path];
    theTable = [[theTable empty] free];

    dragPasteboard = [Pasteboard newName: NXDragPboard];
    types[0] = NXFilenamePboardType;
    types[1] = NXFileContentsPboardType;
    types[2] = NULL;
    [dragPasteboard declareTypes:types num:2 owner:nil];
    [dragPasteboard writeType:types[0] data:path length:strlen(path)+1];
    [dragPasteboard writeFileContents:path];

    org = e->location;
    [self convertPoint:&org fromView:nil];
```

```
    org.x -= 5.0; org.y += 5.0;
    offset.x = offset.y = 0.0;
    [self dragImage:[NXImage findImageNamed:"NXLinkButton"]
        at:&org offset:&offset
        event:e pasteboard:dragPasteboard
        source:self slideBack: YES];
    return self;
}

- draggedImage:(NXImage *)image endedAt:(NXPoint *)screenPoint deposited:
(BOOL)didDeposit
{
// FIX THIS -- It litters /tmp with junk.
//  unlink(path);
    return self;
}
@end
```