

```
//
// FILENAME: eTDocInfoUI.m
// SUMMARY: Inspector/Controller for eText-based eTDocInfos
// SUPERCLASS: Object:eTDocInfoUI
// PROTOCOLS: <Inspectable>
// INTERFACE: eTDocInfoUI.nib
// AUTHOR: Rohit Khare
// COPYRIGHT: ©1993,94 California Institute of Technology, eText Project
//
// Implementation Comments
// This is a shared inspector (once copy of the nibs, can inspect any
// eText DocInfo). It is also used as an "implicit" annotation in eText docs.
//
// HISTORY
// 10/04/94: Revamped for eText5; renamed eTDocInfoUI
// 01/15/94: Created for eText4.0. Borrows from DocInspector in eText3.0
//
// Imported Interfaces
//
// #import "eTDocInfoUI.h"

@implementation eTDocInfoUI
//
// Class Management
//
+ new {
    static eTDocInfoUI *ui = nil;

    if (!ui) {
        ui = [[eTDocInfoUI alloc] init];
    }
    return ui;
}

- free {
    return self;
}

- init {
    char buf[MAXPATHLEN];
    NXBundle *bundle;

    [super init];
    bundle = [NXBundle bundleForClass:[eTDocInfoUI class]];
    if ([bundle getPath:buf forResource:"eTDocInfoUI" ofType:"nib"] ) {
        [NXApp loadNibFile:buf owner:self withNames:NO];
    } else {
        NXLogError("NIB not found: eTDocInfoUI");
    }
}
```

[illegible]

```

// <Inspectable> Obligations
//
- (const char *) inspectorTitle {return NXUniqueString("DocInfo");}
- (const NXAtom *) types {
    static NXAtom *typesForInspector=NULL;

    if (!typesForInspector) {
        typesForInspector = malloc(4 * sizeof(NXAtom));
        typesForInspector[0] = NXUniqueString(PROP);
        typesForInspector[1] = NXUniqueString(TAGS);
        typesForInspector[2] = NXUniqueString(INFO);
        typesForInspector[3] = NULL;
    }
    return typesForInspector;
}

- inspectorForType: (const char *) type {
    if (!strcmp(type, NXUniqueString(PROP)))
        return docInfoView;
    else if (!strcmp(type, NXUniqueString(TAGS)))
        return taggingView;
    else if (!strcmp(type, NXUniqueString(INFO)))
        return infoView;
    NXLogError("Massive Inspector Failure: Asked eTDocInfoUI for: %s", type);
    return docInfoView;
}

- resignInspector: (View *) oldInspector ofType: (const char *) type {
    if (oldInspector == docInfoView) {
        NXStream *stream;
        char *buffer, *data;
        int length, maxlen;

        [theDocInfo setDocTitle: [title stringValue]];
        [theDocInfo setAuthor: [author stringValue]];
        [theDocInfo setKeywords: [keywords stringValue]];
        [theDocInfo setParent: [parent stringValue]];
        [theDocInfo setPeers: [peers stringValue]];
        // Note that this call is in the eText category of theDocInfo; eDraw?
        [theDocInfo setEmitTrailer: (BOOL) [trailerSw state]];

        stream = NXOpenMemory(NULL, 0, NX_WRITEONLY);
        [comments writeRichText: stream];
        NXGetMemoryBuffer(stream, &buffer, &length, &maxlen);
        data = malloc((length+1) * sizeof(char));
        bcopy(buffer, data, length);
        data[length] = 0;
        NXCloseMemory(stream, NX_FREEBUFFER);
        [theDocInfo setRTFComments: NXUniqueString(data)];
    }
}

```

```

    free(data);

    [navigator touch];

} else if (oldInspector == taggingView) {
    [theDocInfo setTag:H1 to:[heading1Well font]];
    [theDocInfo setTag:H2 to:[heading2Well font]];
    [theDocInfo setTag:H3 to:[heading3Well font]];
    [theDocInfo setTag:H4 to:[heading4Well font]];
    [theDocInfo setTag:H5 to:[heading5Well font]];
    [theDocInfo setTag:H6 to:[heading6Well font]];
    [theDocInfo setTag:QT to:[quotationWell font]];
    [theDocInfo setTag:BODY to:[bodyWell font]];
}
return self;
}
- activateInspector: (View *) newInspector ofType: (const char *) type {
    if (newInspector == docInfoView) {
        NXStream *stream;
        char *buf;

        [docIDField setStringValue:[theDocInfo docIDStr]];
        [dateField setStringValue:[theDocInfo lastModifyDate]];
        [title setStringValue:[theDocInfo docTitle]];
        [keywords setStringValue:[theDocInfo keywords]];
        [author setStringValue:[theDocInfo author]];
        [parent setStringValue:[theDocInfo parent]];
        [peers setStringValue:[theDocInfo peers]];
        // Note that this call is in the eText category of theDocInfo; eDraw?
        [trailerSw setState:[theDocInfo emitTrailer]];
        buf = (char *)[theDocInfo rtfComments];
        if (buf) {
            stream = NXOpenMemory(buf, strlen(buf), NX_READONLY);
            [comments readRichText:stream];
            NXClose(stream);
        } else {
            [comments setSel:0 :[comments textLength]];
            [comments setText:""];
        }
    } else if (newInspector == taggingView) {
        [heading1Well setFont:[theDocInfo tagFont:H1]];
        [heading2Well setFont:[theDocInfo tagFont:H2]];
        [heading3Well setFont:[theDocInfo tagFont:H3]];
        [heading4Well setFont:[theDocInfo tagFont:H4]];
        [heading5Well setFont:[theDocInfo tagFont:H5]];
        [heading6Well setFont:[theDocInfo tagFont:H6]];
        [quotationWell setFont:[theDocInfo tagFont:QT]];
    }
}

```

[illegible]

```

- (Font *)heading2 {
    return [heading2Well font];}

- setHeading3:sender {
    if ([sender isKindOfClass:[Font class]])
        [heading3Well setFont:sender];
    return [self touch];}

- (Font *)heading3 {
    return [heading3Well font];}

- setHeading4:sender {
    if ([sender isKindOfClass:[Font class]])
        [heading4Well setFont:sender];
    return [self touch];}

- (Font *)heading4 {
    return [heading4Well font];}

- setHeading5:sender {
    if ([sender isKindOfClass:[Font class]])
        [heading5Well setFont:sender];
    return [self touch];}

- (Font *)heading5 {
    return [heading5Well font];}

- setHeading6:sender {
    if ([sender isKindOfClass:[Font class]])
        [heading6Well setFont:sender];
    return [self touch];}

- (Font *)heading6 {
    return [heading6Well font];}

- setQuotation:sender {
    if ([sender isKindOfClass:[Font class]])
        [quotationWell setFont:sender];
    return [self touch];}

- (Font *)quotation {
    return [quotationWell font];}

- setBody:sender {
    if ([sender isKindOfClass:[Font class]])
        [bodyWell setFont:sender];
    return [self touch];}

- (Font *)body {
    return [bodyWell font];}

```

@end