# Cocoaで
# マルチウインドウ

成田　丞

mindtools@mac.com

http://www.mindto01s.com

# 複数のWindowを扱う

NSDocumentと

NSDocumentControllerのお話ではない。

データの持ち方や表示の仕方のお話。

成田は本当はなんて用語かを判ってない。

何をしているかも判ってない。

毎度のように勝手に飛ばして行きます。

# 最もシンプルなGUIアプリケーション構造

アプリケーション　　データモデル　　ウィンドウ

一つのアプリ

一つのモデル

一つのウィンドウ

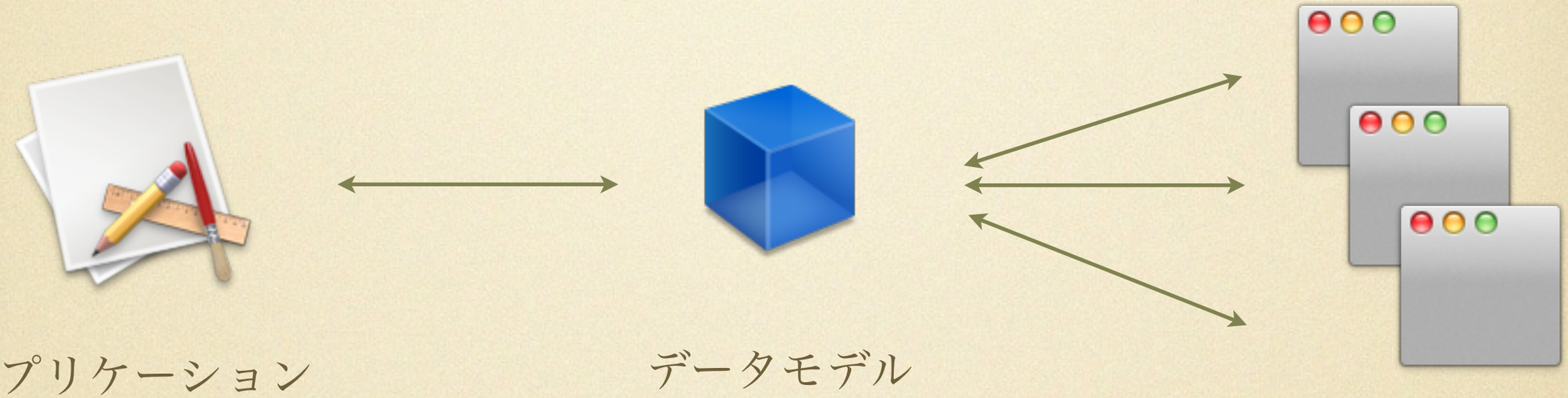
# 例えば以下のようなアプリ


計算機

AppStore

Photo Booth

# 最もシンプルなマルチウインドウGUIアプリケーション構造




同一のModelを複数のViewで表示している

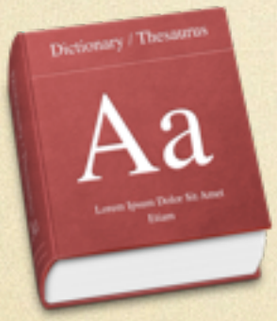
一つのアプリ

一つのモデル

複数のウィンドウ

# 例えば以下のようなアプリ

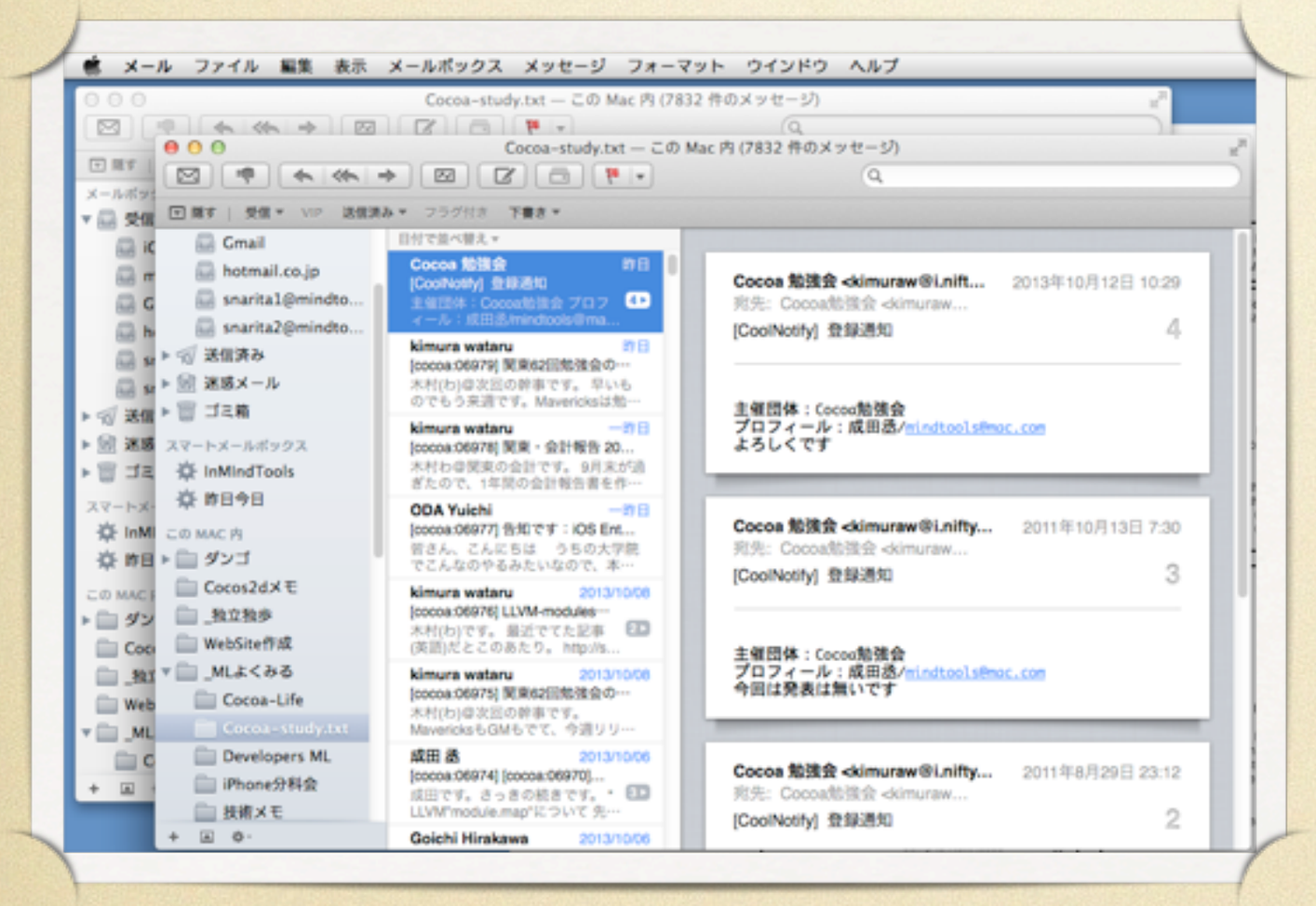
Cocoa Browser




辞書

Mail

# 例えば以下のような表示

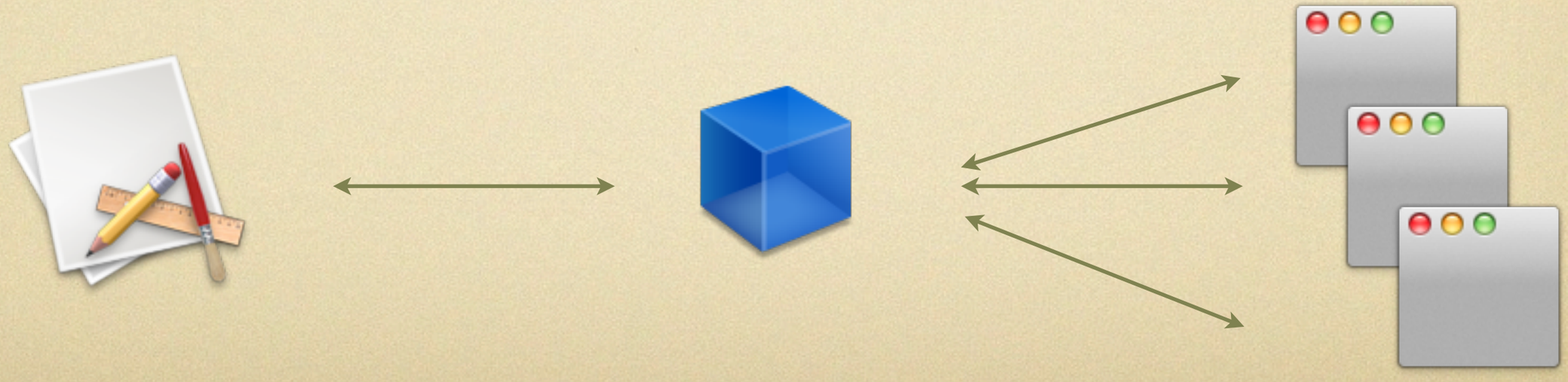



同じデータを別Windowで表示している

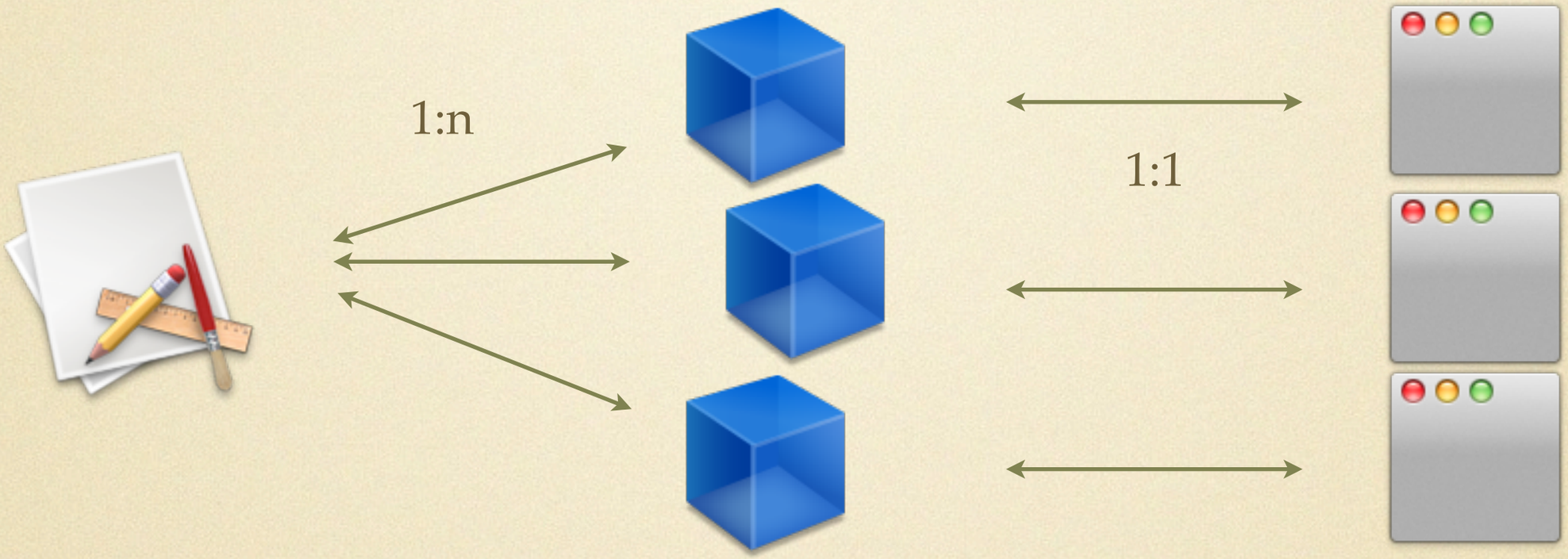
# 命名

## 1M1W型

## 1MnW型



# ドキュメントベースアプリケーション構造




アプリケーション　　データモデル　　ウィンドウ

一つのアプリ

n個のモデル

n個のウィンドウ

Modelと Viewは 1:1

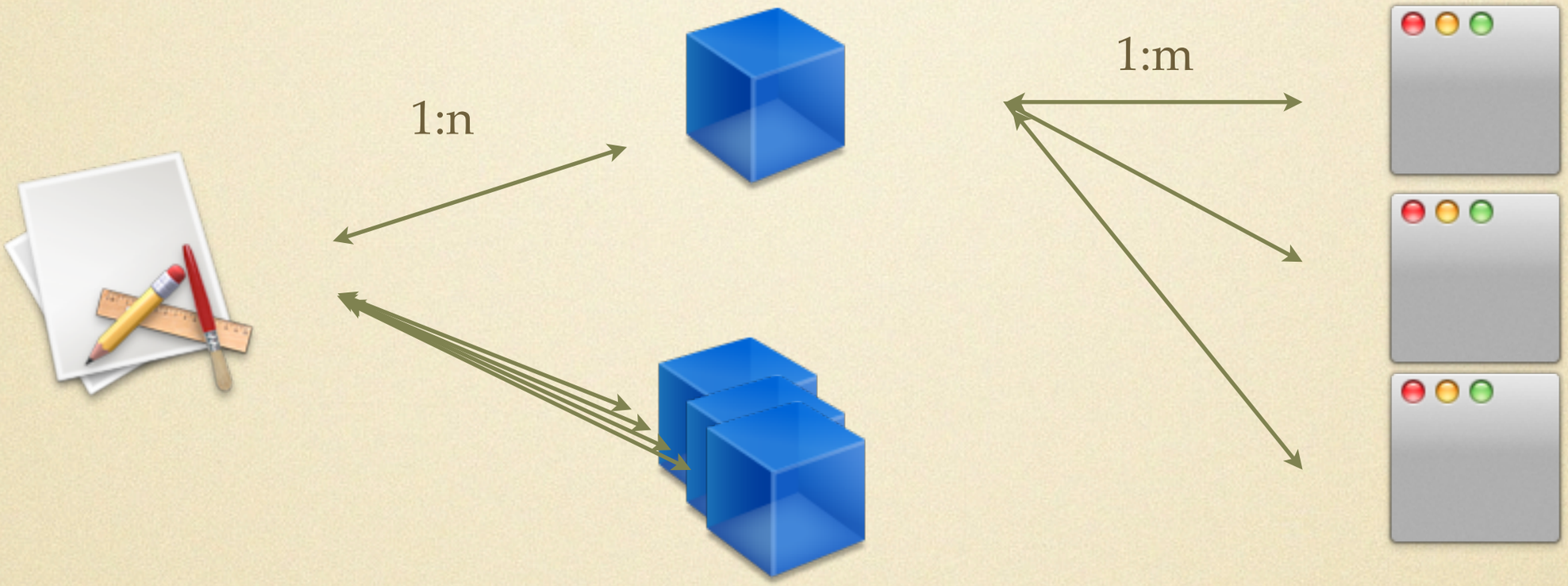
# 例えば以下のようなアプリ

Pages

Keynote


Numbers

# ドキュメントベース マルチウインドウ アプリケーション構造




アプリケーション　　データモデル　　ウィンドウ

- 一つのアプリ
- n個のモデル
- m個のウィンドウ

同一のModelを複数のViewで表示している

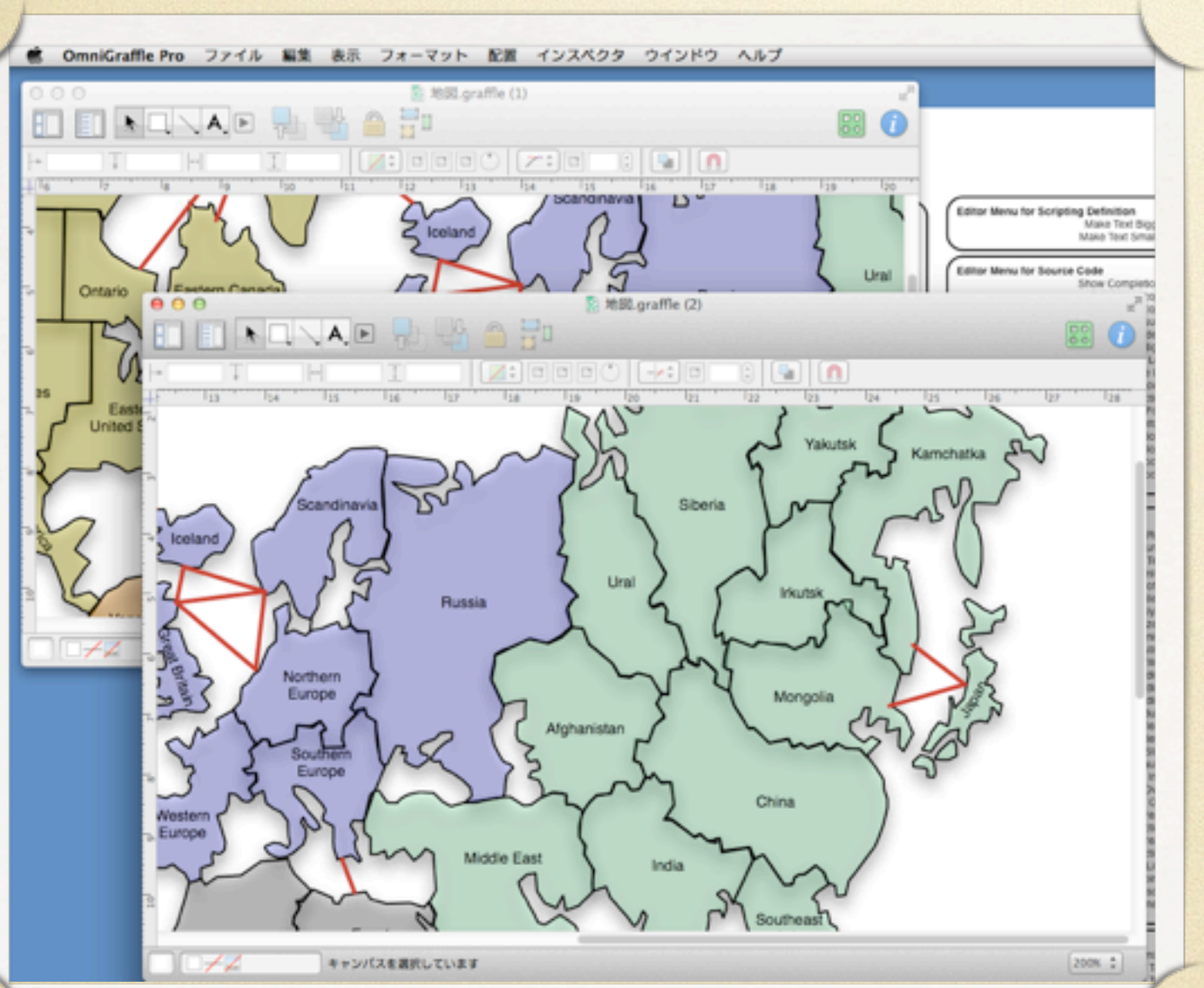
# 例えば以下のようなアプリ

OmniGraffle Pro

PhotoShop CS

MS-Word

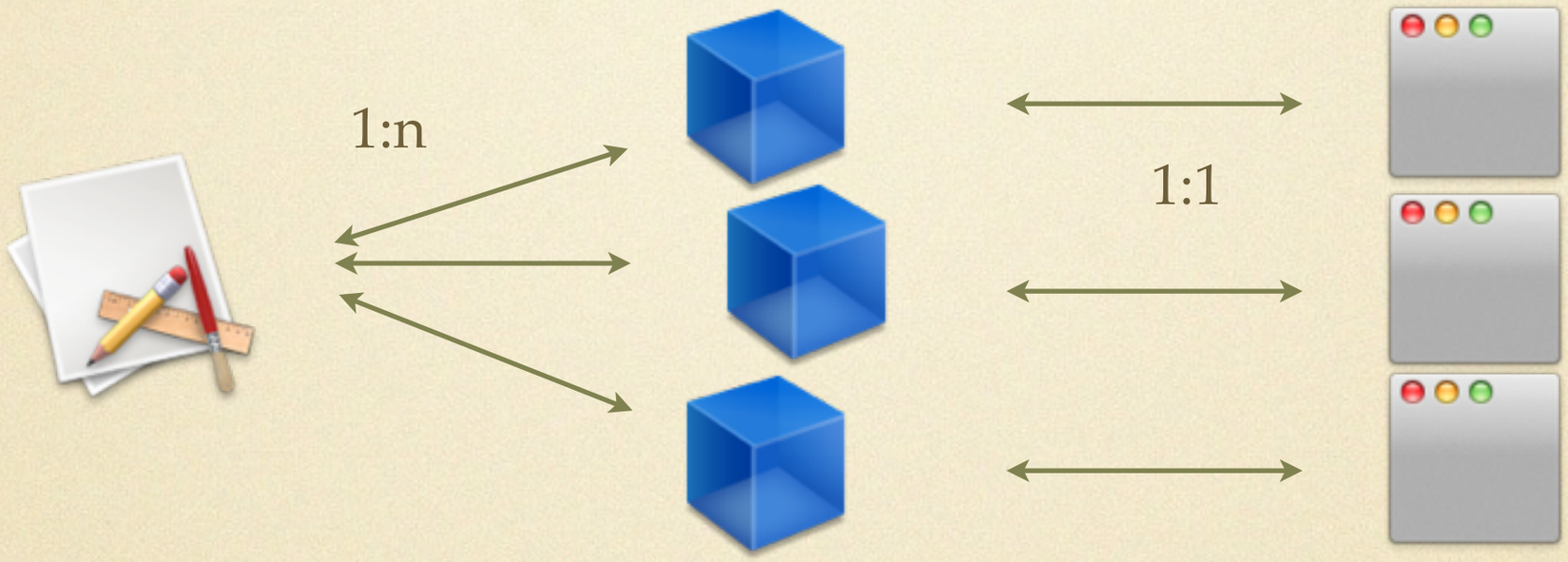
# 例えば以下のような表示



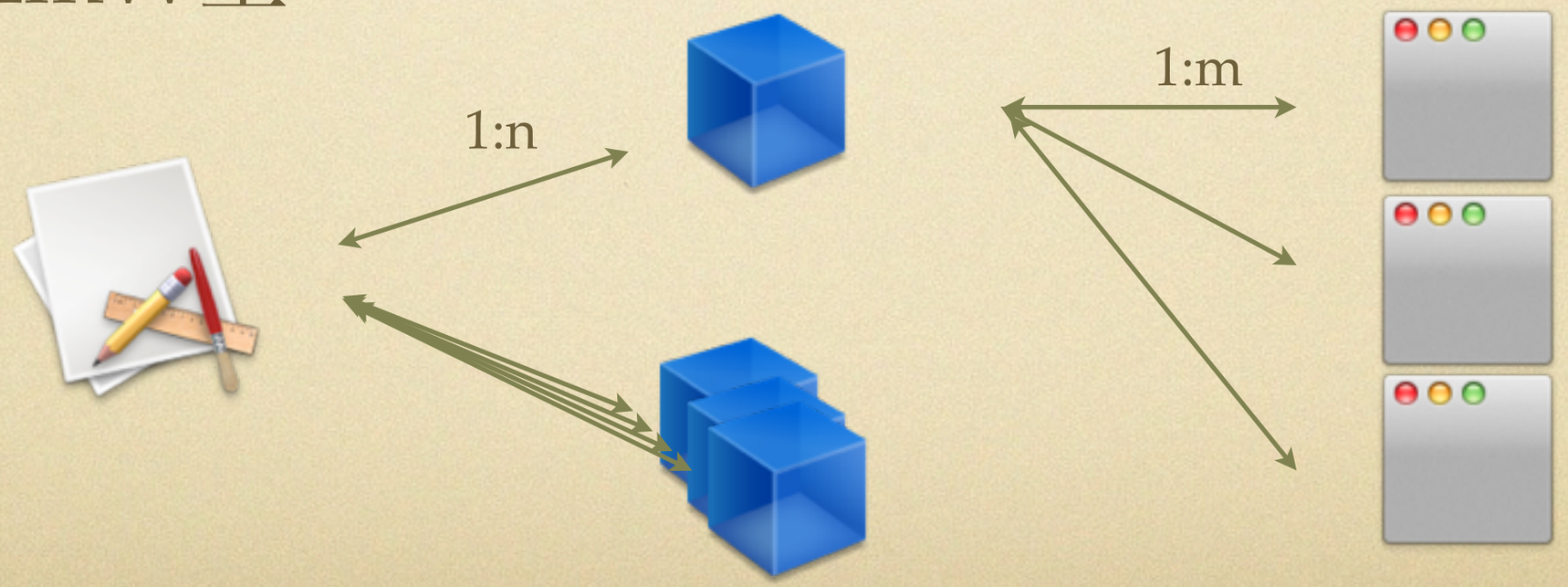

同じファイルを別Windowで表示している

# 命名

## nM1W型




## nMmW型



# 考えを整理する

# 分類してみる

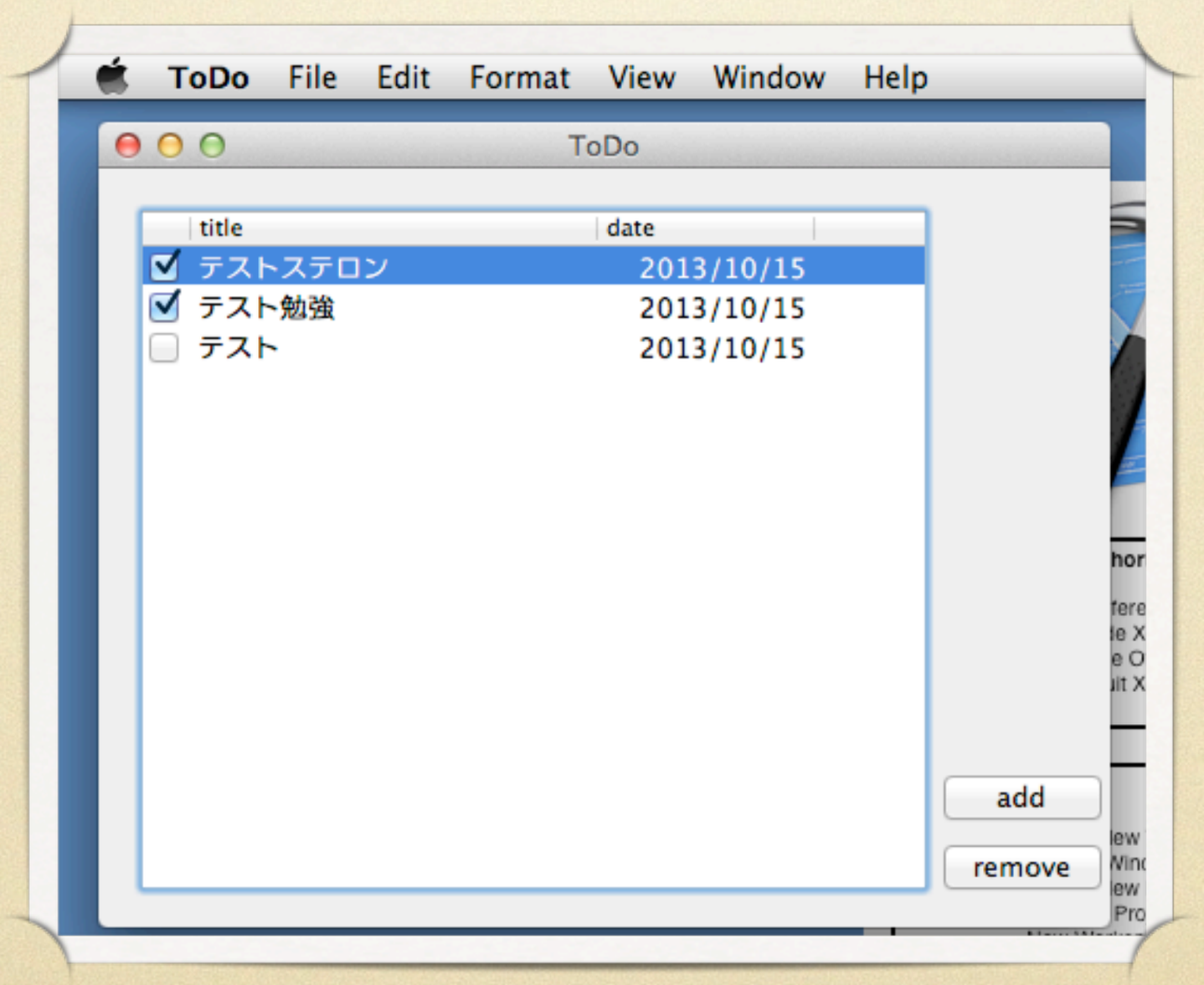
| | |
|---|---|
| 1M1W型 | 単機能なユーティリティー |
| 1MnW型 | ファイルを意識させない非ドキュメントモデル |
| nM1W型 | 標準的なドキュメントモデル |
| nMmW型 | 大型のグラフィックソフト的な何か |

# ライブラリのサポート

| | |
|---|---|
| 1M1W型 | なし。無くても平気 |
| 1MnW型 | 特になし。でもコレ本命じゃねーの？ |
| nM1W型 | NSDocumentとNSDocumentController |
| nMmW型 | メソッドのオーバーライドで対応 |

# 実装しながら考える

# ToDoアプリを作る


ToDo File Edit Format View Window Help
ToDo
title
date
テストステロン 2013/10/15
テスト勉強 2013/10/15
テスト 2013/10/15
add
remove

# 仕様

- 項目の追加と削除
- Undo/Redo
- モデルデータの保存と読込み
- window状態の保存と再生

ほとんどCoreDataとbindingに御任せです。

# 1M1W型

アプリケーション

データモデル

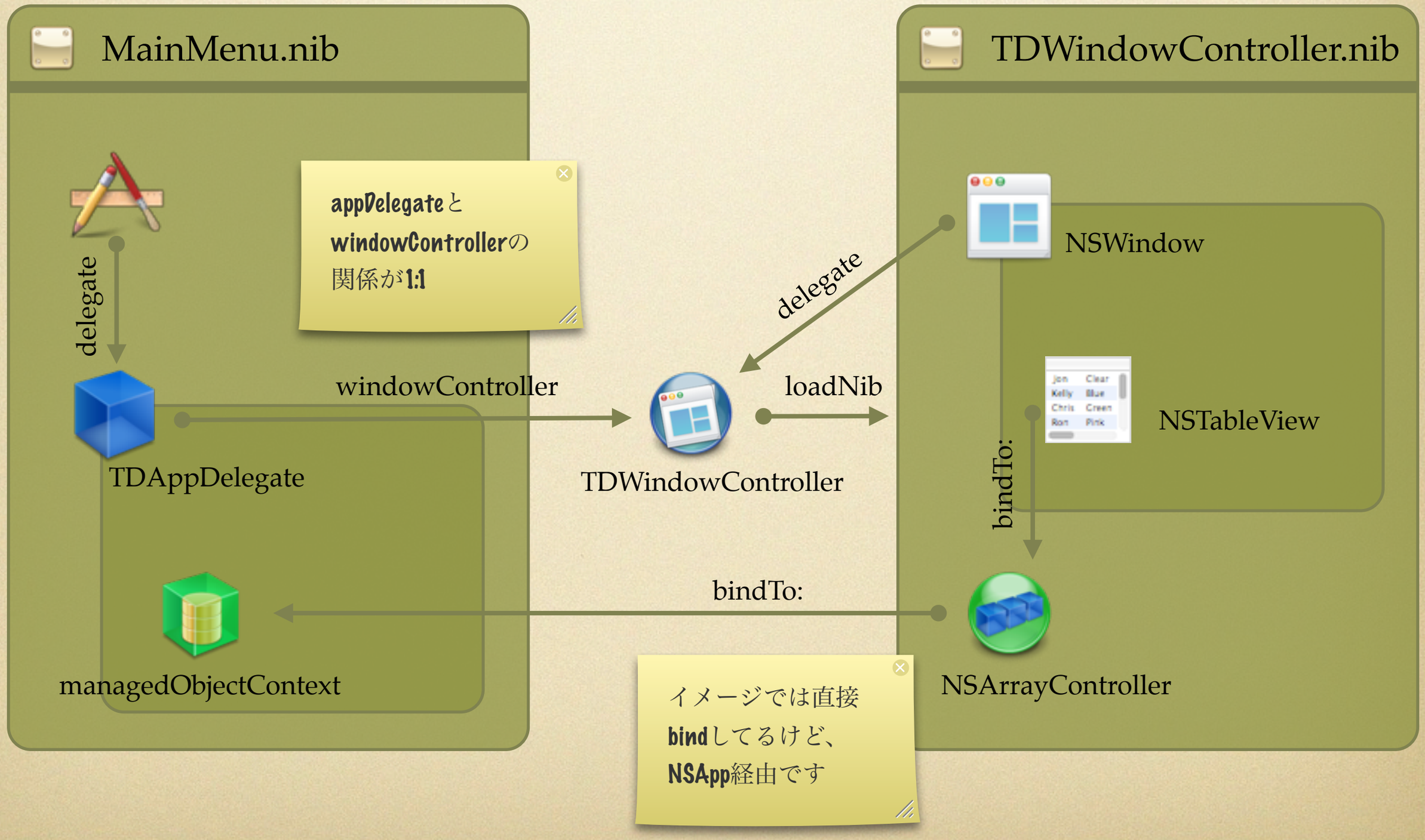
ウィンドウ

サンプルコードはToDo_1M1Wです

# コード & Nib

| | |
|---|---|
| **TDAppDelegate** | Application delegate。<br>起動と同時にTDWindowControllerのインスタンス化でWindowを開く以外は、CoreDataのメンテナンスコード。 |
| **MainMenu.xib** | メニューの作成と、TDAppDelegateのインスタンス化をしてます。<br>WindowをインスタンスはTDWindowController.xibで行ってます。 |
| **TDWindowController.h.m** | TDWindowController.nibの読込みと、ソート項目と選択項目の状態保存をやってます。選択項目の状態保存出来ないバグあり。 |
| **TDWindowController.xib** | Windowのインスタンス化とGUIパーツをCoreDataへのBindingを行ってます。 |
| **ToDoItem.h.m** | CoreDataのクラス。初期値に作成日を入れる為に作った。 |

# Objectグラフ


MainMenu.nib
appDelegateと
windowControllerの
関係が1:1
delegate
windowController
TDAppDelegate
managedObjectContext
TDWindowController
loadNib
TDWindowController.nib
NSWindow
delegate
NSTableView
bindTo:
bindTo:
NSArrayController
イメージでは直接
bindしてるけど、
NSApp経由です

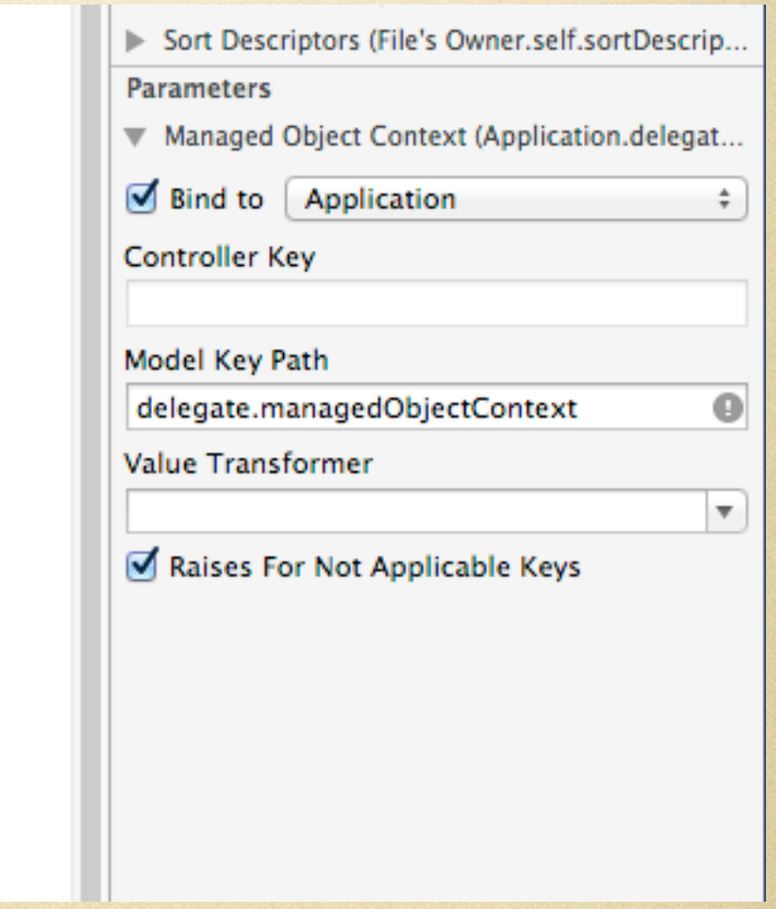
# 見所

## TDWindowController.xib

Applicationは何処でもアクセス可能なので、

delegate.managedObjectContext

と手繰れる。



# 1MnW型

アプリケーション

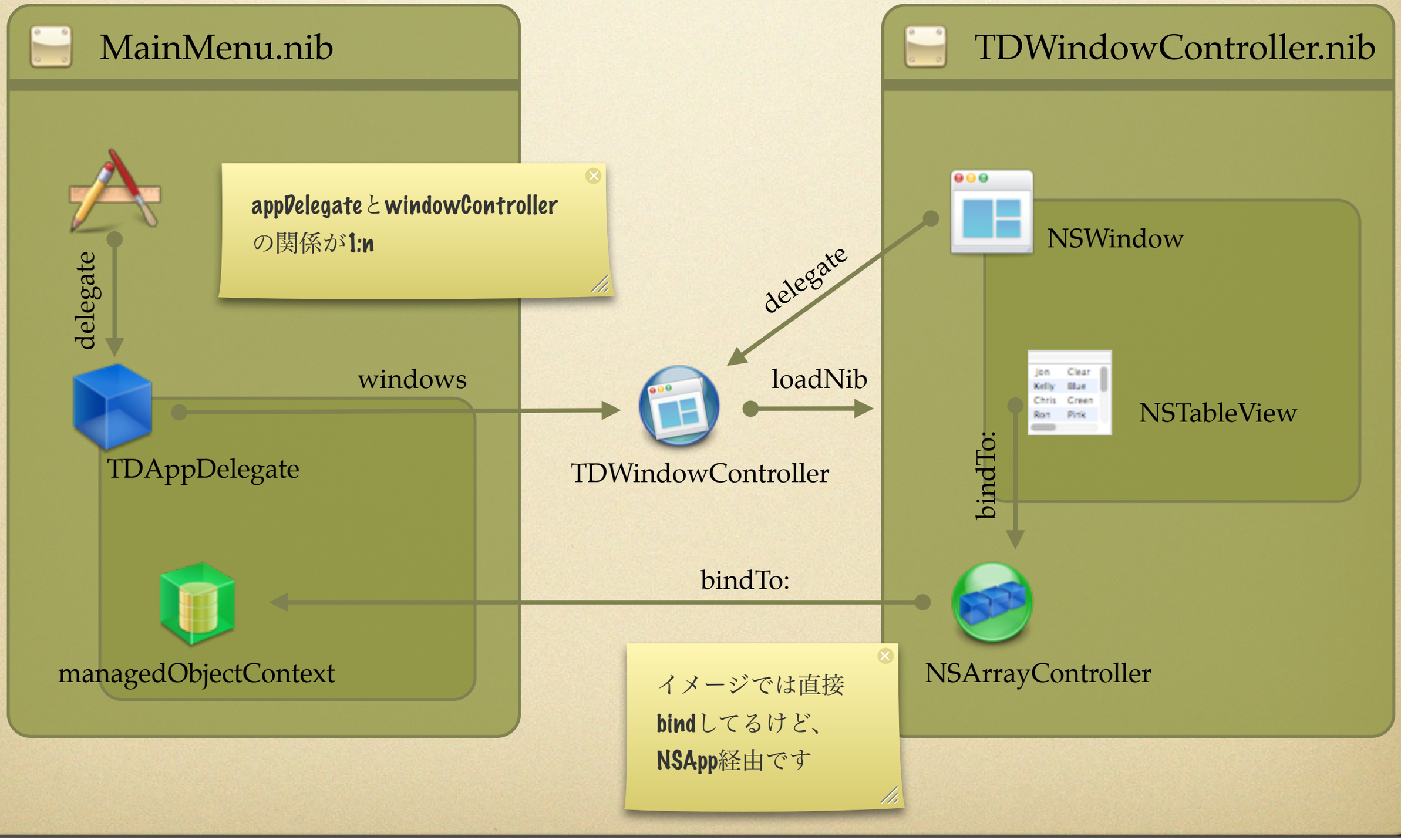
データモデル

ウィンドウ

# コード & Nib

| | |
|---|---|
| **TDAppDelegate** | Application delegate。 |
| **MainMenu.xib** | メニューの作成と、TDAppDelegateのインスタンス化をしてます。 |
| **TDWindowController.h.m** | TDWindowController.nibの読込みと、ソート項目と選択項目の状態保存をやってます。選択項目の状態保存出来ないバグあり。 |
| **TDWindowController.xib** | Windowのインスタンス化とGUIパーツをCoreDataへのBindingを行ってます。 |
| **ToDoItem.h.m** | CoreDataのクラス。初期値に作成日を入れる為に作った。 |

# Objectグラフ


MainMenu.nib
appDelegateとwindowControllerの関係が1:n
delegate
windows
TDAppDelegate
managedObjectContext
TDWindowController
loadNib
bindTo:
TDWindowController.nib
NSWindow
delegate
NSTableView
bindTo:
NSArrayController
イメージでは直接bindしてるけど、NSApp経由です

# 見所

## [TDAppDelegate synchronizeWindowIdentifier]

windowの増減の毎に呼出して、windowのidentifierを付け直してます。

```
#define kTDWindowIdentifierLabel @"TDWindowController:"
- (void) synchronizeWindowIdentifier
{
    for( TDWindowController* i in self.windows )
    {
        i.window.identifier = [NSString stringWithFormat:(kTDWindowIdentifierLabel @"%lu"),
                               [self.windows indexOfObject:i]];
    }
}
```

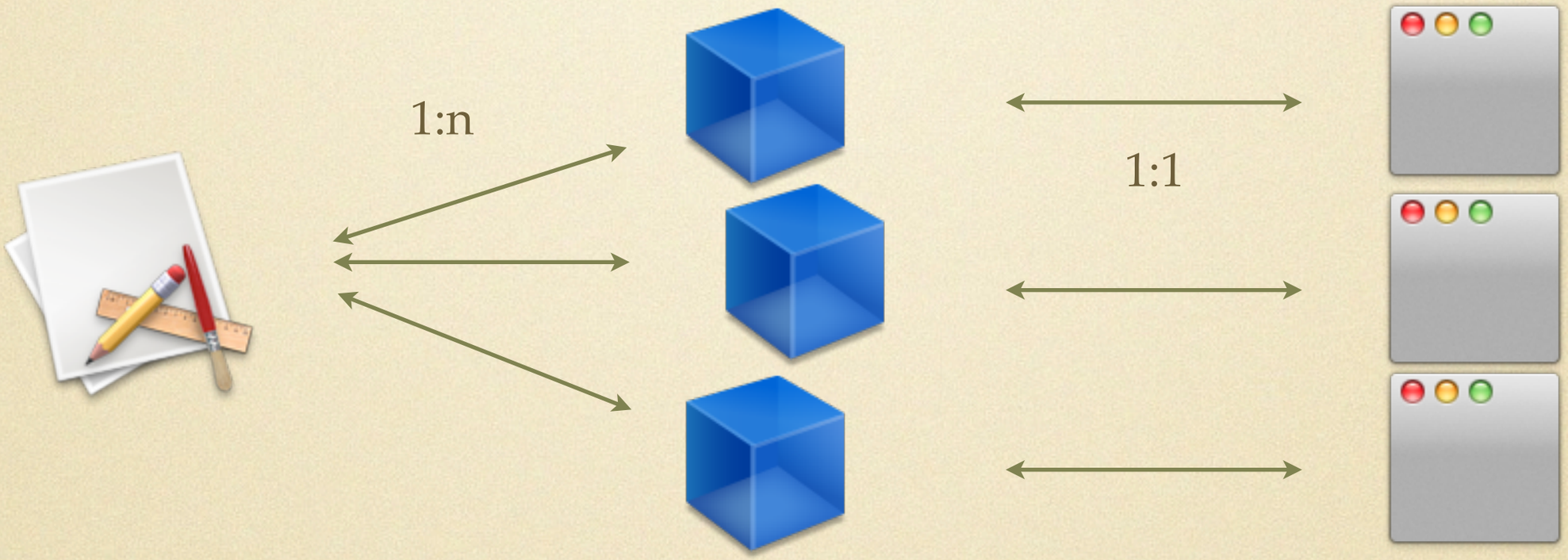
# 見所

## [TDAppDelegate restoreWindowWithIdent.....

```
+ (void)restoreWindowWithIdentifier:(NSString *)identifier
                              state:(NSCoder *)state
                  completionHandler:(void (^)(NSWindow *, NSError *))completionHandler
{
    // 詳細はコードを見てもらうとして、処理は
    // identifierを調べてを、”クラス名:連番”のフォーマットで分轄。
        // windowの数が少なければ新規にWindowControllerを作る
        // 数が十分であれば連番の番号のwindowの状態を復帰させる
}
```

この復帰作業やidentefierの名前付けをwindow毎に行う事でWindowoを動的に持っても自然に復帰出来るようにしている

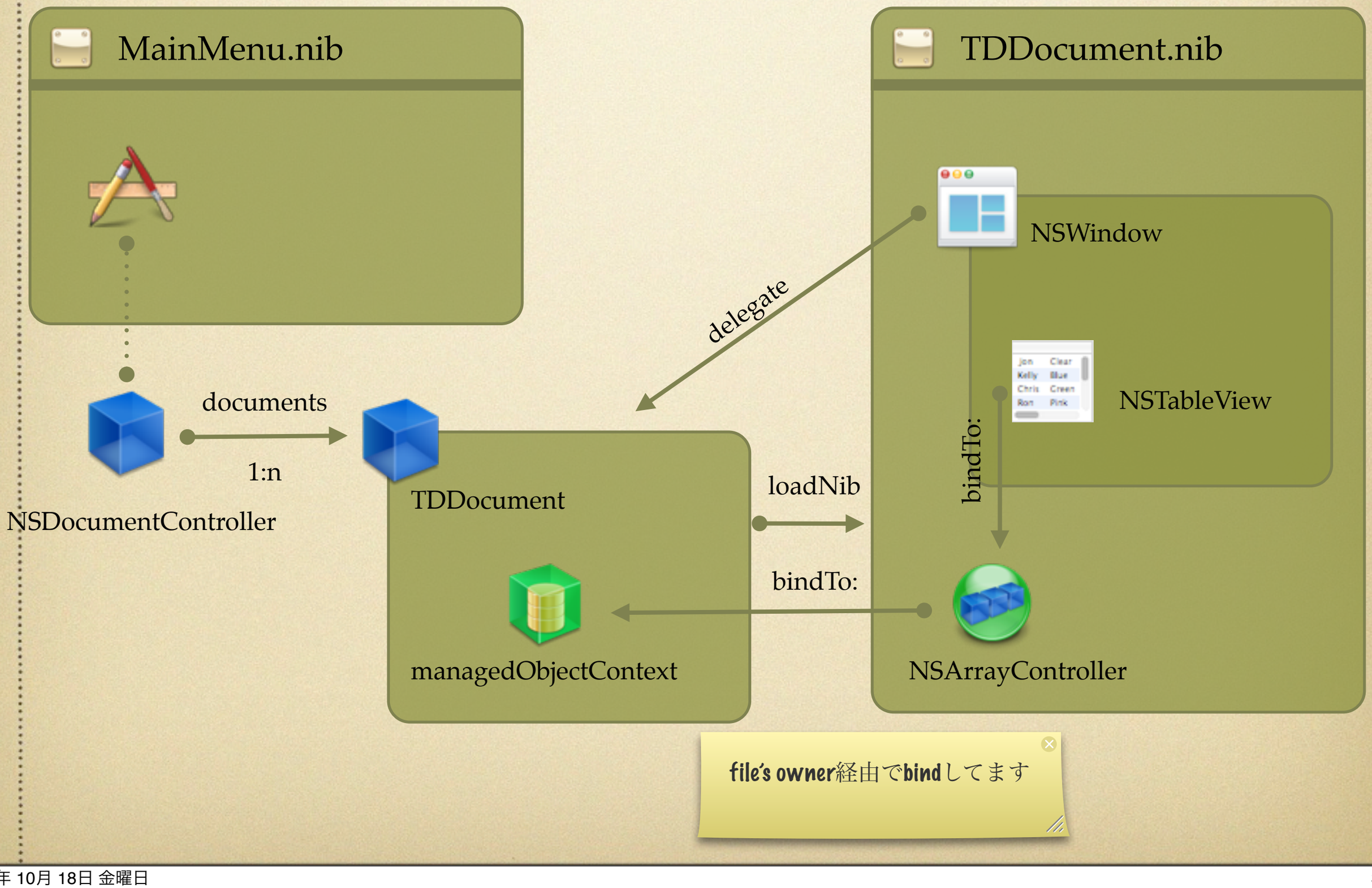
# nM1W型



# コード & Nib

| | |
|---|---|
| **MainMenu.xib** | メニューの作成だけ |
| **TDDocument.h.m** | TDDocument.xib の読込みと、ソート項目と選択項目の状態保存をやってます。選択項目の状態保存出来ないバグあり。 |
| **TDDocument.xib** | Windowのインスタンス化とGUIパーツをCoreDataへのBindingを行ってます。 |
| **ToDoItem.h.m** | CoreDataのクラス。初期値に作成日を入れる為に作った。 |

# Objectグラフ


MainMenu.nib
TDDocument.nib
NSWindow
delegate
NSTableView
documents
1:n
bindTo:
NSDocumentController
TDDocument
loadNib
bindTo:
managedObjectContext
NSArrayController
file's owner経由でbindしてます

# 見所

## TDDocument.m
## 多分一番シンプル

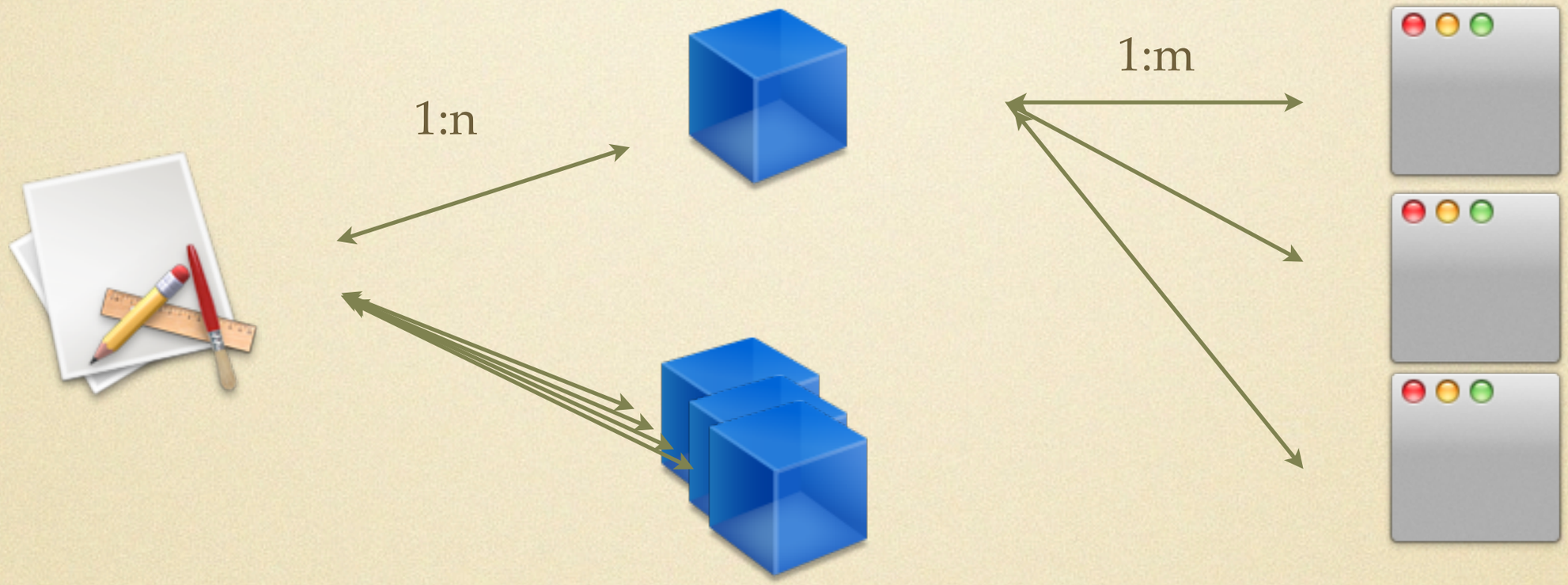
状態の保存と復帰を行っている

```
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder;
- (void)restoreStateWithCoder:(NSCoder *)coder
```

状態が変化した事を知らせるためだけに、” invalidateRestorableState”を呼出してます。

```
- (void) setSortDescriptorArray:(NSArray *)sortDescriptorArray
{
    _sortDescriptorArray = sortDescriptorArray;
    [self invalidateRestorableState];
}
```

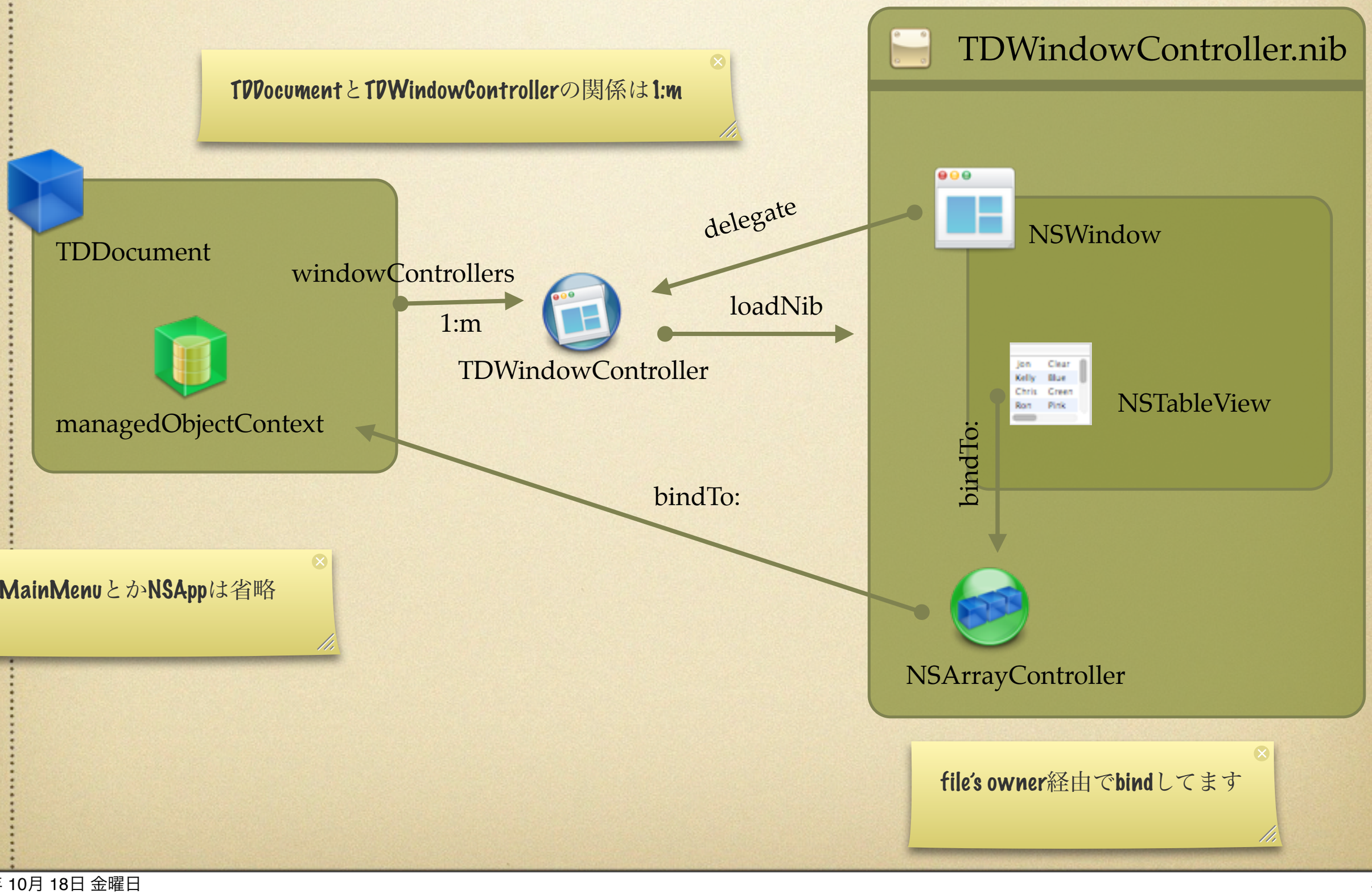
# nMmW型



# コード & Nib

| | |
|---|---|
| **MainMenu.xib** | メニューの作成だけ |
| **TDDocument** | WindowControllerの作成/削除を行う事で複数Windowへの対応。 |
| **TDWindowController.h.m** | TDWindowController.nibの読込みと、ソート項目と選択項目の状態保存をやってます。選択項目の状態保存出来ないバグあり。 |
| **TDWindowController.xib** | Windowのインスタンス化とGUIパーツをCoreDataへのBindingを行ってます。 |
| **ToDoItem.h.m** | CoreDataのクラス。初期値に作成日を入れる為に作った。 |

# Objectグラフ


TDDocumentとTDWindowControllerの関係は1:m
TDWindowController.nib
TDDocument
windowControllers
1:m
delegate
loadNib
NSWindow
NSTableView
bindTo:
TDWindowController
managedObjectContext
bindTo:
NSArrayController
MainMenuとかNSAppは省略
file's owner経由でbindしてます

# 見所

## TDDocument.m

1MnW型と同じようにここで、Identifierの更新と、windowの作成を行ってます。

- (void) synchronizeWindowIdentifier

- (void)restoreDocumentWindowWithIdentifier:(NSString *)identifier .....

windowが追加/削除される毎に、Identiferとタイトルの更新を行ってます

- (void)addWindowController: (NSWindowController *)windowController
- (void)removeWindowController: (NSWindowController *)windowController

# 見所

## TDWindowController

```
// クラス名と同じnibファイルを読み込ませる
- (NSString *)windowNibName
{
    NSString* theSuperWindowNibName = [super windowNibName];

    if(theSuperWindowNibName != nil && ![theSuperWindowNibName isEqualToString:@""])
    {
        return theSuperWindowNibName;
    }
    else
    {
        NSString* theClassName = NSStringFromClass([self class]);

        return theClassName;
    }
}
```

# 見所

```
// window毎に”ドキュメント名:(番号)”なタイトルを付ける
- (NSString *) windowTitleForDocumentDisplayName: (NSString *)displayName
{
    if ([self document] != nil){
                NSString*   theDisplayName;
                NSArray*   theWinCtrls;

                theDisplayName = [[self document] displayName];
                theWinCtrls = [[self document] windowControllers];

                if([theWinCtrls count] > 1){
                        NSUInteger        theIndex;
                        NSString*   theResult;

                        theIndex = [theWinCtrls indexOfObject:self] + 1;
                        theResult = [NSString stringWithFormat:@"%@:(%lu)", theDisplayName, theIndex];
                        return theResult;
                }
                else
                {return [super windowTitleForDocumentDisplayName:displayName];}
    }
    else
    {return [super windowTitleForDocumentDisplayName:displayName];}
}
```

# 感想はこんな感じ

| | |
|---|---|
| 1M1W型 | nibの分轄以外はふつー |
| 1MnW型 | 少しのコード追加でも資料が少ない |
| nM1W型 | 多分一番シンプル |
| nMmW型 | nibの分轄と少しのコード追加 |

# まとめ

アプリケーション

データモデル

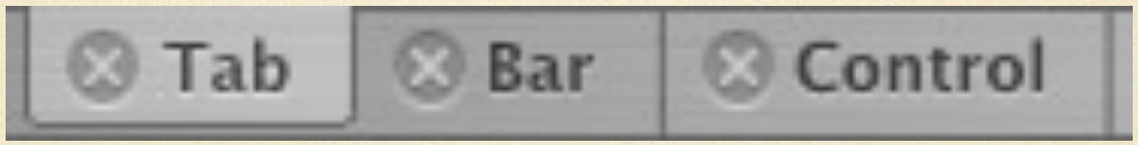
ウィンドウ

これらの関係を1:nの関係にする手法の説明でした。

でも問題があります。

# Tab interface

タブ




windowController経由でmodelにアクセスしてはイケナイ。

NSDocumentとNSWindowControllerの関係も見直す必要あり

# Pane interface

ペイン

NSSplitterViewじゃないよ。

分轄した画面がそれぞれ同じモデルの同じ表示方法の、別の部分を表示している。