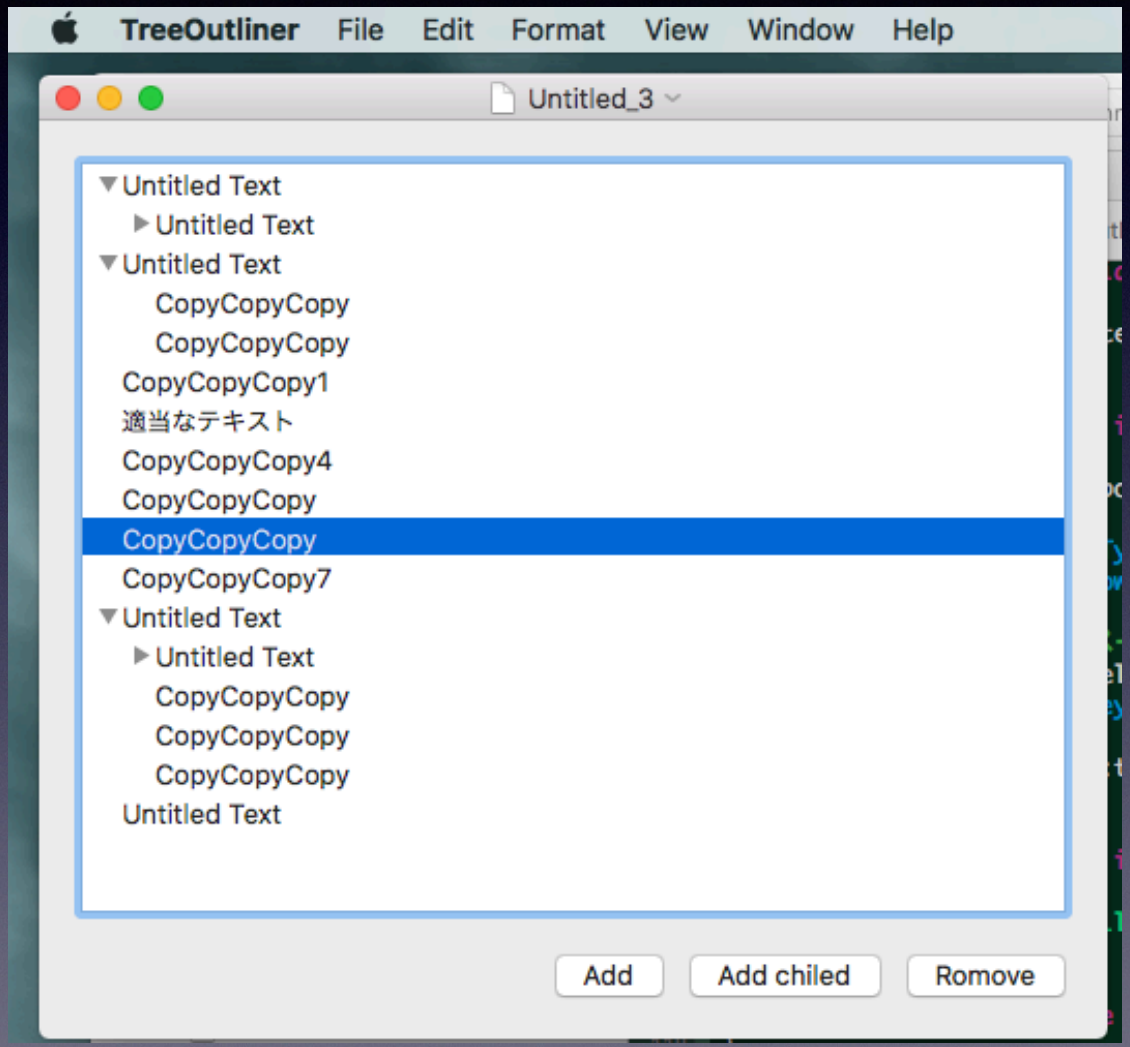
最小限アウトラインプロセッサの作り方

目的

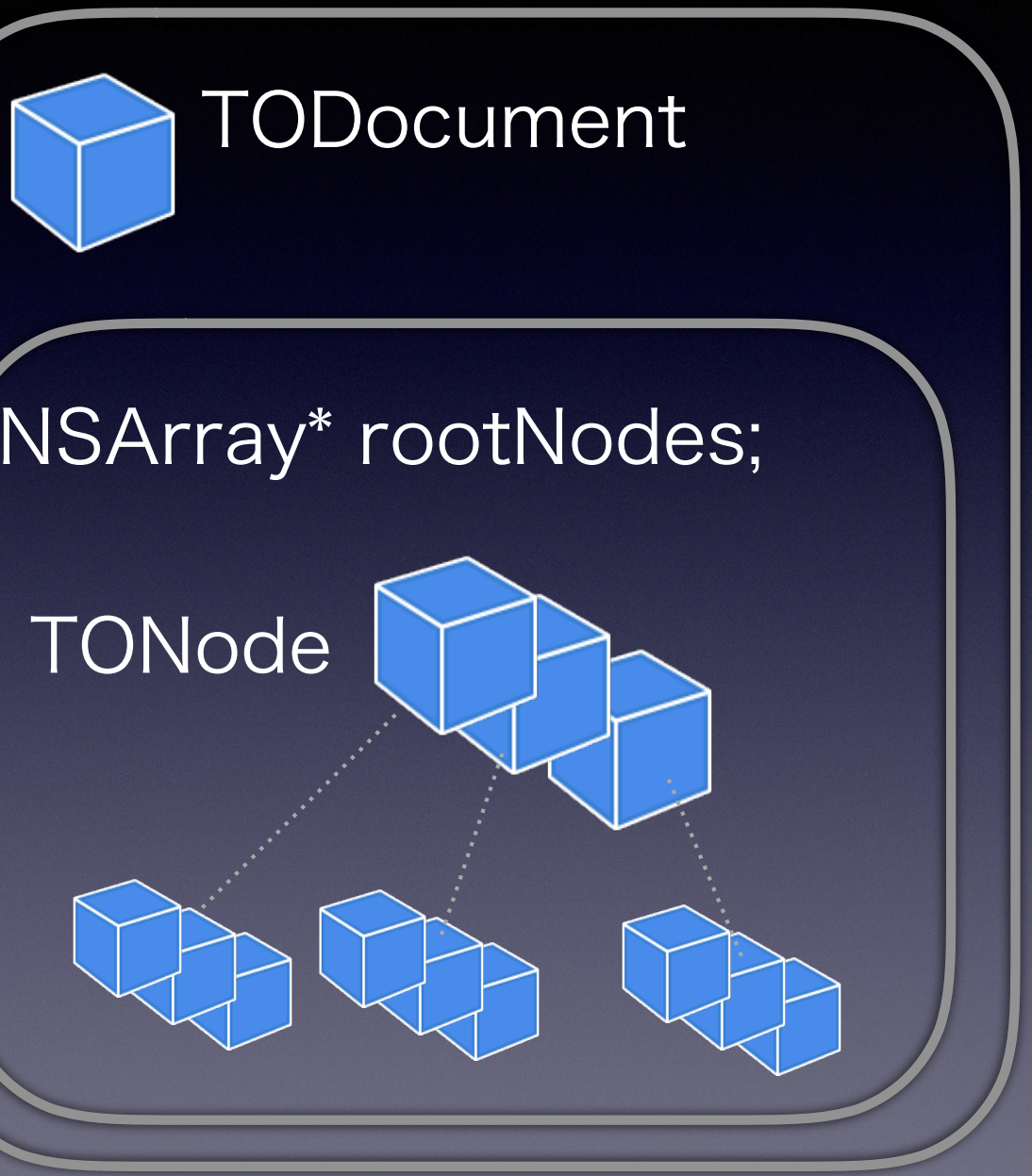
- アウトラインプロセッサ作成の最初の一歩にあたる部分をチュートリアル風にまとめた
- Drag&Dropについて古いAPIに基づくものが多いので、新しいAPIを使う資料の作成
- 今後同じ様なプロジェクトを開始する場合の備忘録
- 自分自身向けのNSOutlineView & NSTreeControllerの備忘録

アプリ概要説明

GUI


TreeOutliner File Edit Format View Window Help
Untitled_3
Untitled Text
Untitled Text
Untitled Text
CopyCopyCopy
CopyCopyCopy
CopyCopyCopy1
適当なテキスト
CopyCopyCopy4
CopyCopyCopy
CopyCopyCopy
CopyCopyCopy7
Untitled Text
Untitled Text
CopyCopyCopy
CopyCopyCopy
CopyCopyCopy
Untitled Text
Add
Add chiled
Romove

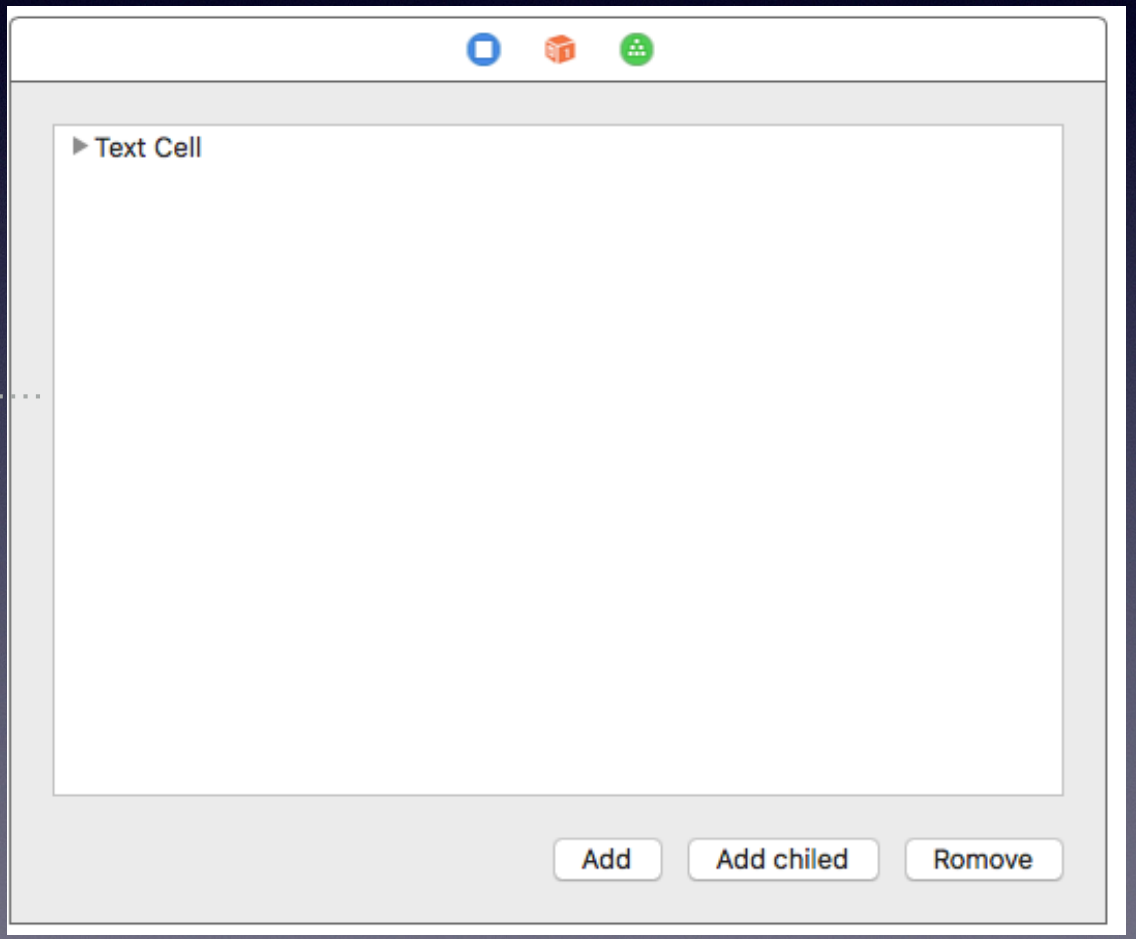
データモデル



NSTreeController

Viewモデル

NSOutlineView



NSTreeController

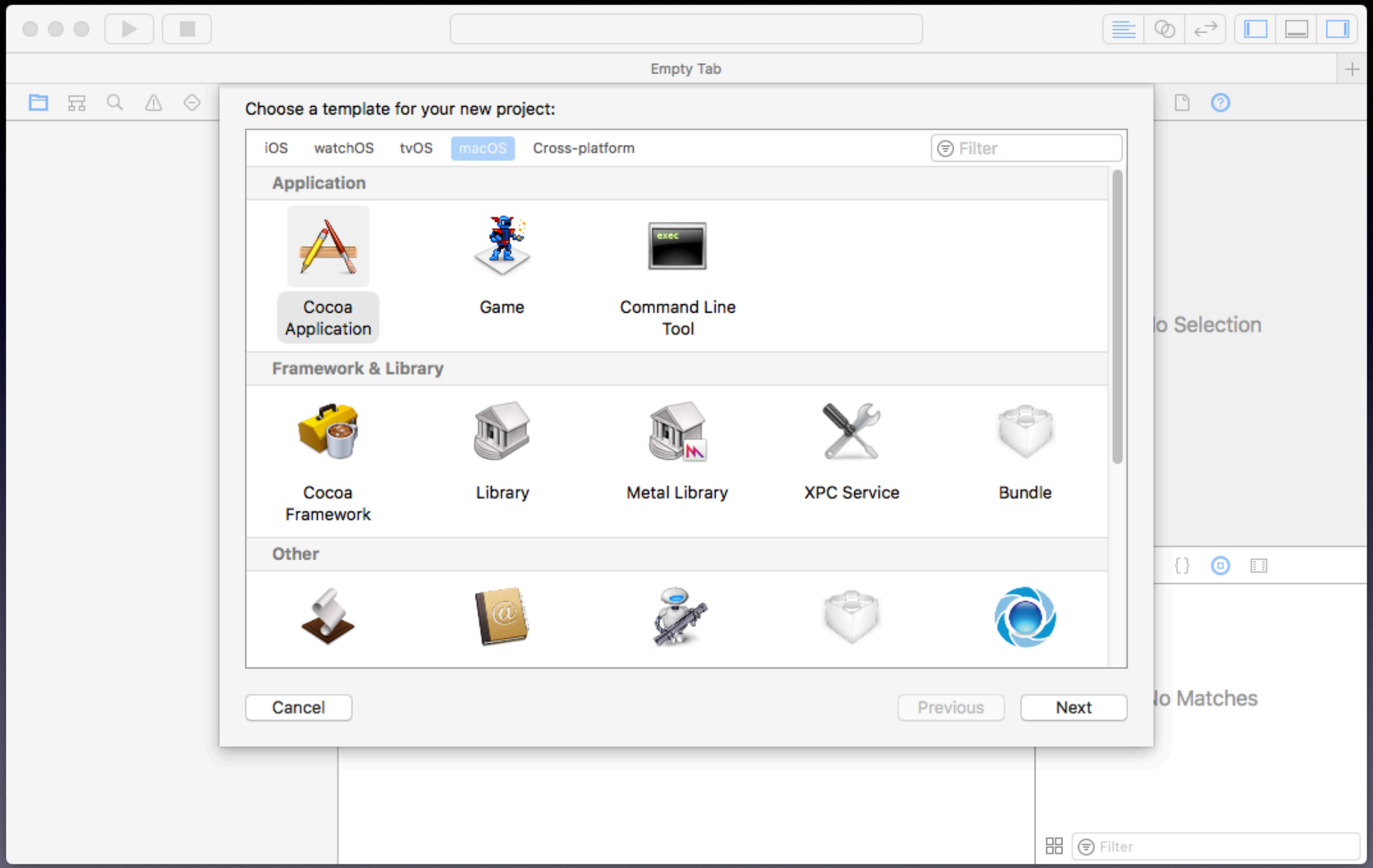
実装する機能

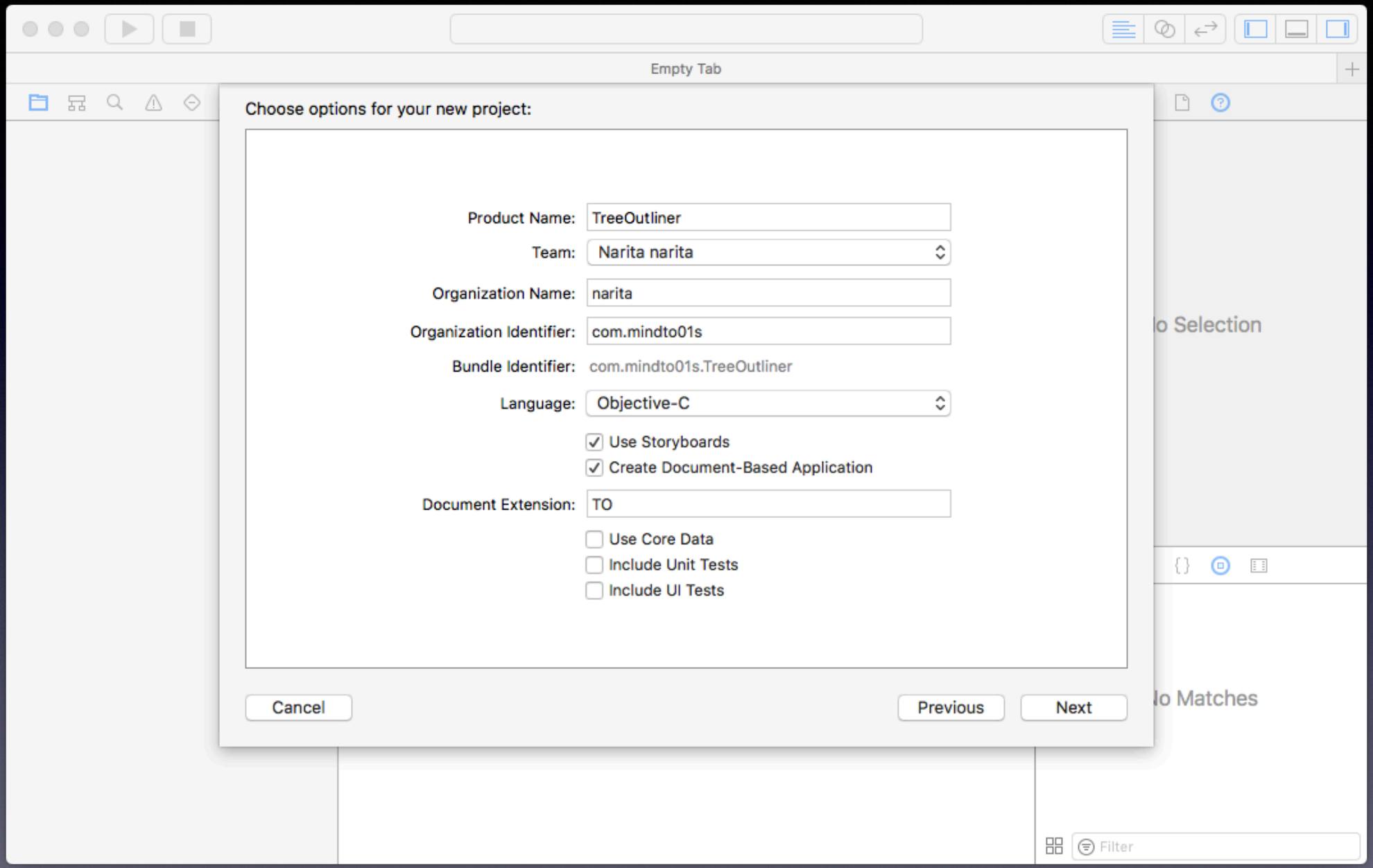
- ファイルの保存と読み込み
- Copy & Paste
- Drag & Drop

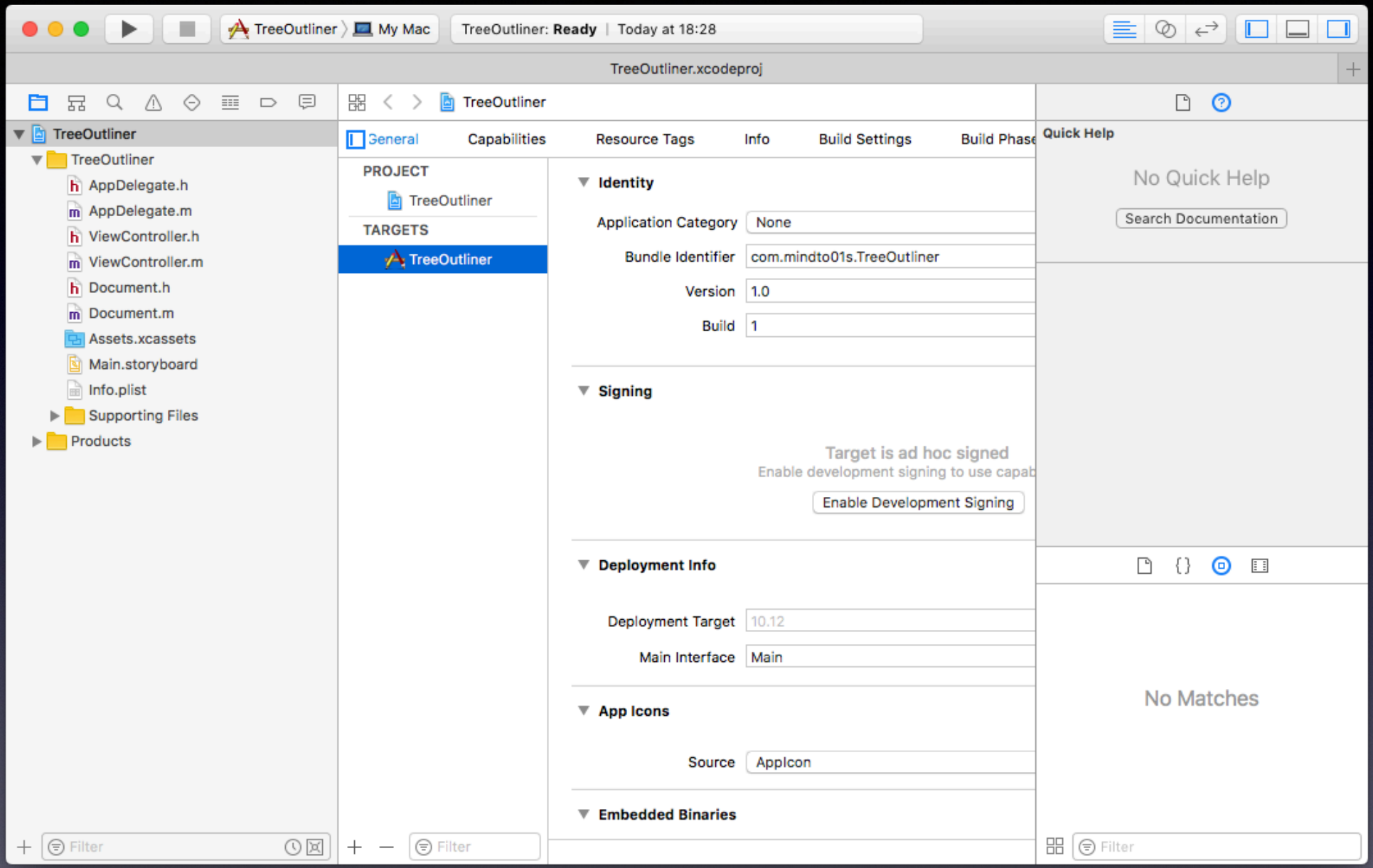
実装しない機能

- restore state (時間がなかった)
- Undo/Redo (忘れてた)
- 複数Window化 (単純化の為)
- Paneに分割 (単純化の為)
- CoreData (単純化の為)

プロジェクトの準備


Empty Tab
Choose a template for your new project:
iOS
watchOS
tvOS
macOS
Cross-platform
Filter
Application
Cocoa Application
Game
Command Line Tool
Framework & Library
Cocoa Framework
Library
Metal Library
XPC Service
Bundle
Other
Cancel
Previous
Next
o Selection
lo Matches
Filter


Empty Tab
Choose options for your new project:
Product Name: TreeOutliner
Team: Narita narita
Organization Name: narita
Organization Identifier: com.mindto01s
Bundle Identifier: com.mindto01s.TreeOutliner
Language: Objective-C
Use Storyboards
Create Document-Based Application
Document Extension: TO
Use Core Data
Include Unit Tests
Include UI Tests
Cancel
Previous
Next
o Selection
lo Matches
Filter


TreeOutliner 〉 My Mac
TreeOutliner: Ready | Today at 18:28
TreeOutliner.xcodeproj
TreeOutliner
TreeOutliner
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Document.h
Document.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Filter
TreeOutliner
General
Capabilities
Resource Tags
Info
Build Settings
Build Phase
PROJECT
TreeOutliner
TARGETS
TreeOutliner
Identity
Application Category
None
Bundle Identifier
com.mindto01s.TreeOutliner
Version
1.0
Build
1
Signing
Target is ad hoc signed
Enable development signing to use capab
Enable Development Signing
Deployment Info
Deployment Target
10.12
Main Interface
Main
App Icons
Source
AppIcon
Embedded Binaries
Filter
Quick Help
No Quick Help
Search Documentation
No Matches
Filter

クラスファイルの作成

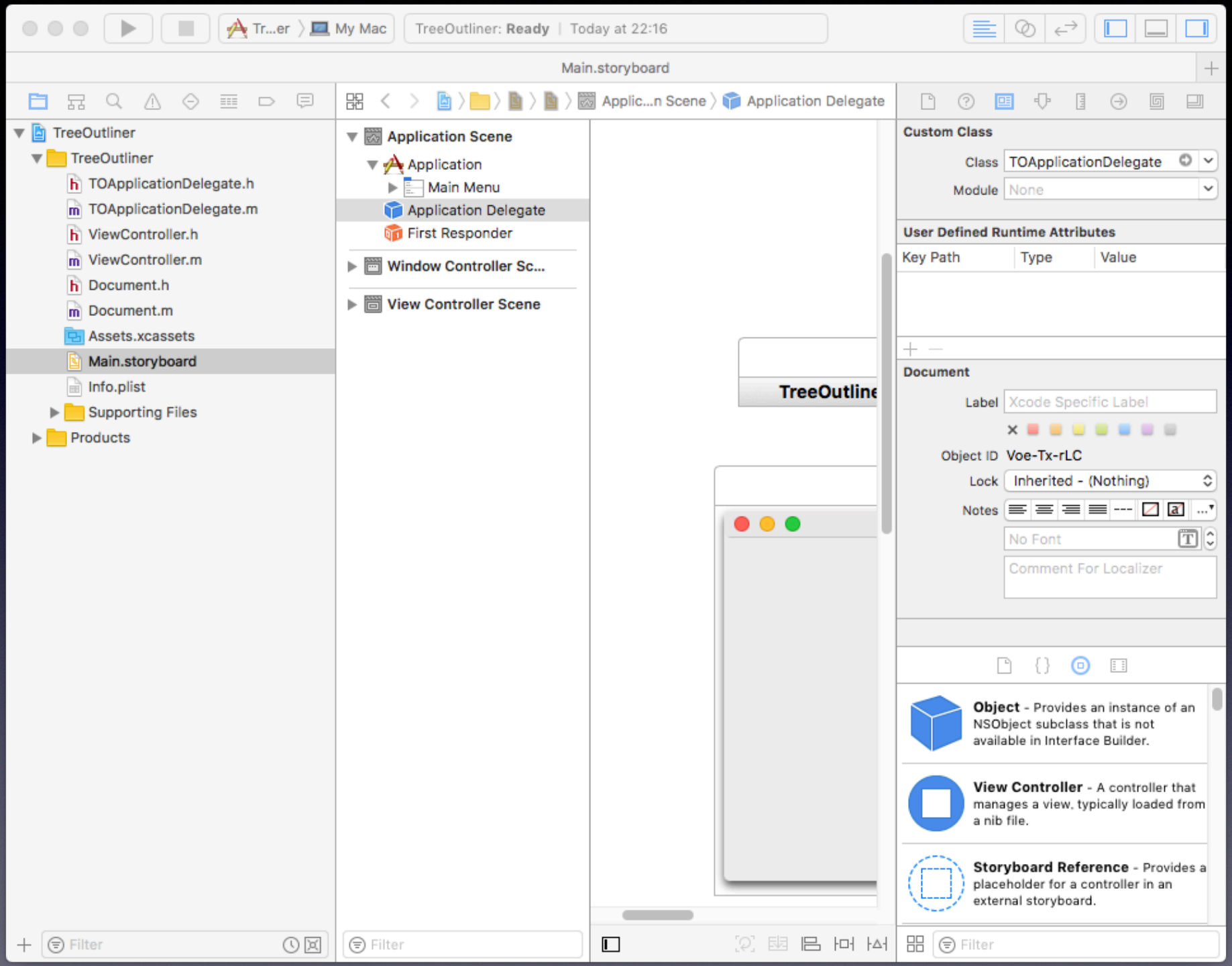
- ファイル名を以下に変更する
- AppDelegate => TOApplicationDelegate
- ViewController => TOViewController
- Document => TODocument

- TOApplicationDelegate

```
#import <Cocoa/Cocoa.h>

#define TOAppDelegate ((TOApplicationDelegate*)[NSApp delegate])

@interface TOApplicationDelegate : NSObject <NSApplicationDelegate>

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

#pragma mark - Utility methods


@end
```

TOApplicationDelegate

```
#import "TOApplicationDelegate.h"

@interface TOApplicationDelegate ()

@end

@implementation TOApplicationDelegate

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
    // Insert code here to initialize your application
}

- (void)applicationWillTerminate:(NSNotification *)aNotification
{
    // Insert code here to tear down your application
}

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

#pragma mark - Utility methods


@end
```

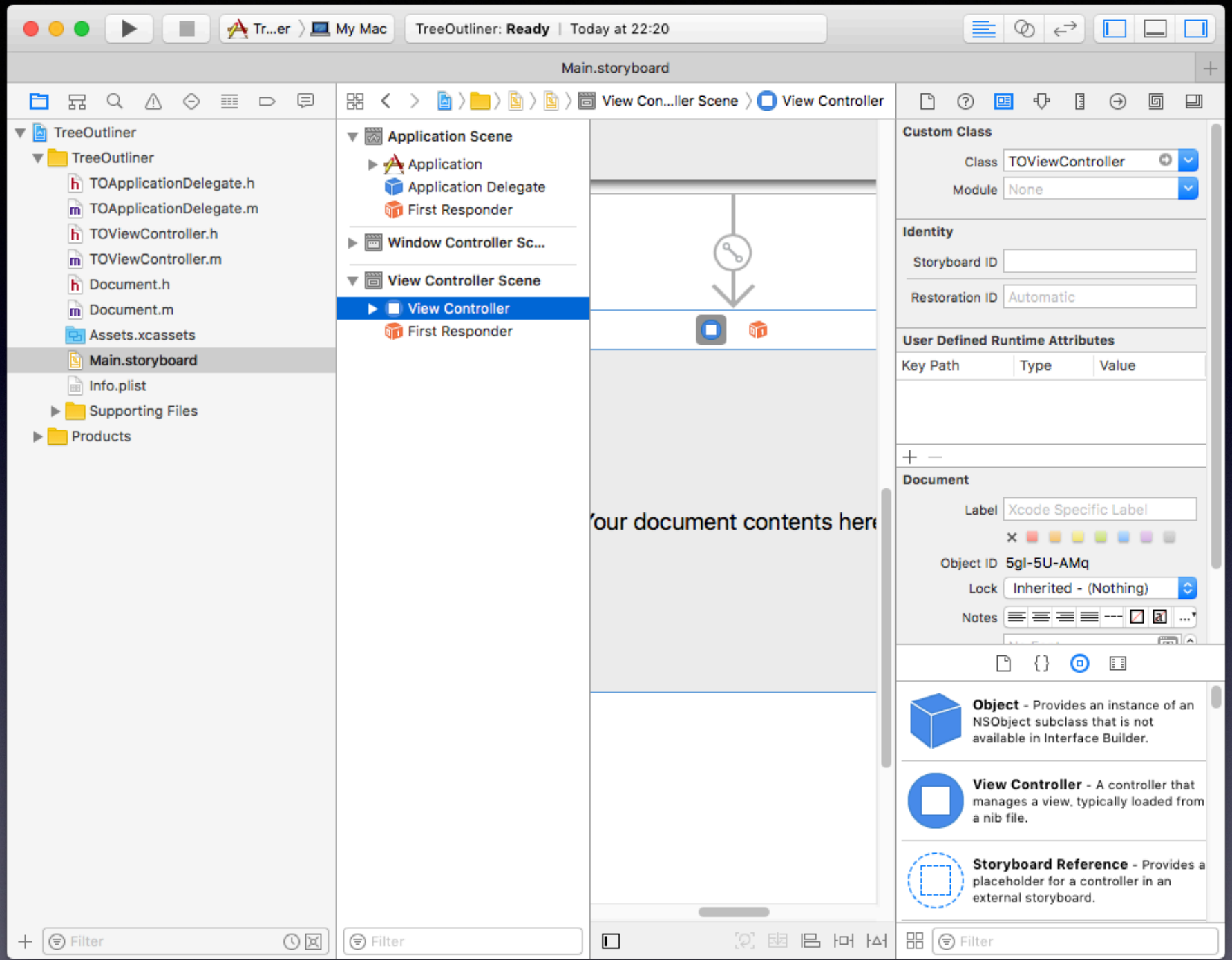

Tr...er
My Mac
TreeOutliner: Ready | Today at 22:16
Main.storyboard
Applic...n Scene
Application Delegate
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
ViewController.h
ViewController.m
Document.h
Document.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Application
Main Menu
Application Delegate
First Responder
Window Controller Sc...
View Controller Scene
TreeOutline
Custom Class
Class TOApplicationDelegate
Module None
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID Voe-Tx-rLC
Lock Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Object - Provides an instance of an NSObject subclass that is not available in Interface Builder.
View Controller - A controller that manages a view, typically loaded from a nib file.
Storyboard Reference - Provides a placeholder for a controller in an external storyboard.
Filter

- TOViewController

```
#import <Cocoa/Cocoa.h>

@interface TOViewController : NSViewController

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

#pragma mark - Utility methods

@end
```

TOViewController

```
#import "TOViewController.h"

@implementation TOViewController

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

- (void)viewDidLoad
{
    [super viewDidLoad];

    // Do any additional setup after loading the view.
}

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

- (void)setRepresentedObject:(id)representedObject
{
    [super setRepresentedObject:representedObject];

    // Update the view, if already loaded.
}

#pragma mark - Utility methods

@end
```


Tr...er
My Mac
TreeOutliner: Ready | Today at 22:20
Main.storyboard
View Con...ller Scene
View Controller
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
Document.h
Document.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Application
Application Delegate
First Responder
Window Controller Sc...
View Controller Scene
View Controller
First Responder
Filter
Filter
Filter
Custom Class
Class TOViewController
Module None
Identity
Storyboard ID
Restoration ID Automatic
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID 5gI-5U-AMq
Lock Inherited - (Nothing)
Notes
Object - Provides an instance of an NSObject subclass that is not available in Interface Builder.
View Controller - A controller that manages a view, typically loaded from a nib file.
Storyboard Reference - Provides a placeholder for a controller in an external storyboard.

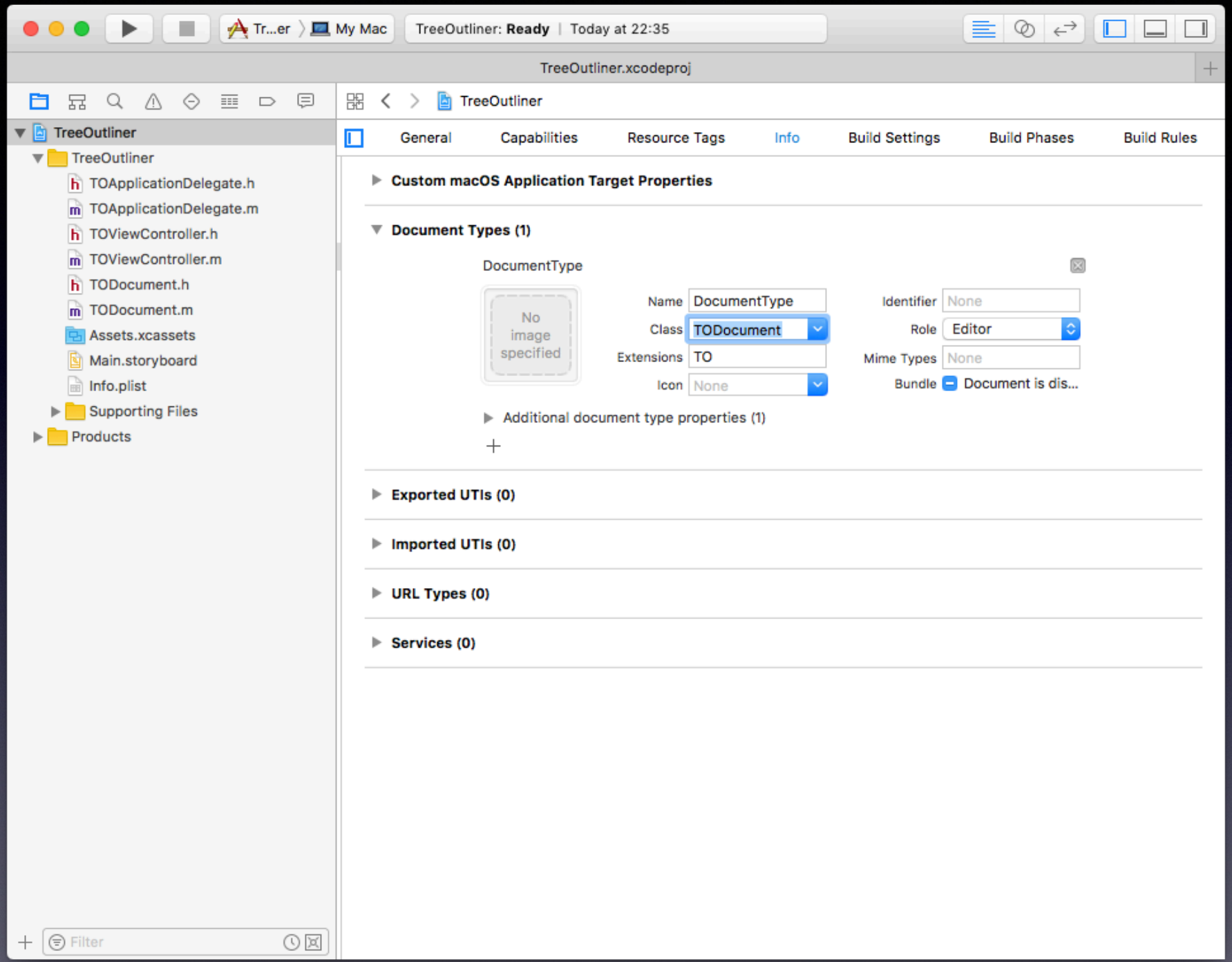
TODocument

```
#import <Cocoa/Cocoa.h>

@interface TODocument : NSDocument

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

#pragma mark - Utility methods

@end
```

• TODocument

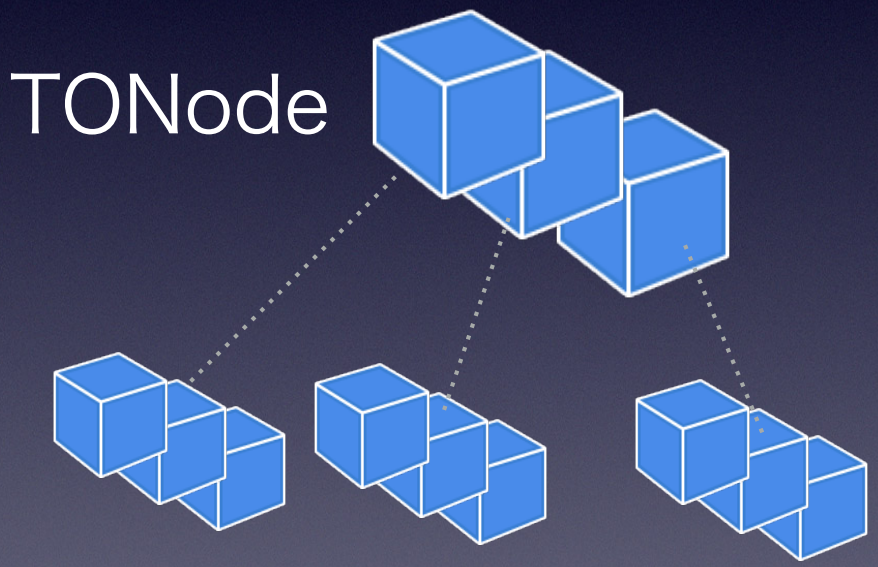
```
#import "TODocument.h"

@interface TODocument ()

@end

@implementation TODocument

#pragma mark - class methods

+ (BOOL)autosavesInPlace
{
    return YES;
}

#pragma mark - init methods

- (instancetype)init
{
    self = [super init];

    if (self)
    {
   }

    return self;
}

#pragma mark - dealloc

- (void)dealloc
{
}
```

```
#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - file handling methods

- (NSData *)dataOfType:(NSString *)typeName error:(NSError **)outError {
    [NSException raise:@"UnimplementedMethod" format:@"%@ is unimplemented",
NSStringFromSelector(_cmd)];
    return nil;
}


- (BOOL)readFromData:(NSData *)data ofType:(NSString *)typeName error:(NSError **)outError
{
    [NSException raise:@"UnimplementedMethod" format:@"%@ is unimplemented",
NSStringFromSelector(_cmd)];
    return YES;
}

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

#pragma mark - Utility methods

- (void)makeWindowControllers {
    // Override to return the Storyboard file name of the document.
    [self addWindowController:[[NSStoryboard storyboardWithName:@"Main" bundle:nil]
instantiateControllerWithIdentifier:@"Document Window Controller"]];
}


@end
```


Tr...er
My Mac
TreeOutliner: Ready | Today at 22:35
TreeOutliner.xcodeproj
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Filter
TreeOutliner
General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
Custom macOS Application Target Properties
Document Types (1)
DocumentType
No image specified
Name DocumentType
Class TODocument
Extensions TO
Icon None
Identifier None
Role Editor
Mime Types None
Bundle Document is dis...
Additional document type properties (1)
Exported UTIs (0)
Imported UTIs (0)
URL Types (0)
Services (0)

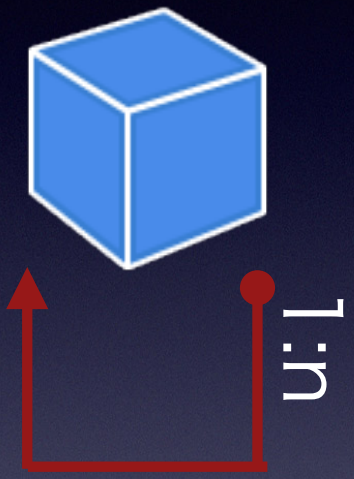
モデルクラスの作成

TONode



TONode

TONode



プロトコル沢山

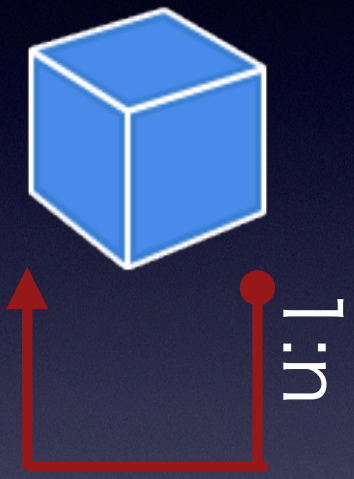
NSCoding,
NSCopying,
NSPasteboardWriting,
NSPasteboardReading

プロパティ少し

```
@property (strong  ) NSMutableArray<TONode*>* children;

@property (copy    ) NSString* text;
```

TONode

TONode



プロトコル沢山

NSCoding,
NSCopying,
NSPasteboardWriting,
NSPasteboardReading

プロパティ少し

```
@property (strong  ) NSMutableArray<TONode*>* children;

@property (copy    ) NSString* text;
```

TONode

```
#import <Cocoa/Cocoa.h>

extern NSString* TONodesPasteboardType;

@interface TONode : NSObject <NSCoding, NSCopying, NSPasteboardWriting, NSPasteboardReading>

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

@property (strong  ) NSMutableArray<TONode*>* children;

@property (copy    ) NSString* text;

#pragma mark - Utility methods

@end
```

TONode

```
#import "TONode.h"

NSString* TONodesPasteboardType = @"com.mindto01.TONodesPasteboardType";

@implementation TONode

#pragma mark - class methods

#pragma mark - init methods

- (id)init
{
    self = [super init];

    if( self != nil )
    {
        self.text     = @"Untitled Text";
        self.children = @[].mutableCopy;
    }

    return self;
}


#pragma mark - dealloc

- (void)dealloc
{
    self.text     = nil;
    self.children = nil;
}

#pragma mark - NSCopying, hash, isEqual:

- (instancetype)copyWithZone:(NSZone *)zone
{
    TONode* theNode = [[[self class] allocWithZone:zone] init];

    theNode.text = self.text;

    return theNode;
}

#pragma mark - NSCoding methods

- (instancetype)initWithCoder:(NSCoder *)coder
{
    // TONodeクラスの super class であるNSObjectはNSCodingに対応していないので、これでOK
    self = [self init];

    self.text     = [coder decodeObjectForKey:@"text"];
    self.children = [[coder decodeObjectForKey:@"children"] mutableCopy];

    return self;
}

- (void)encodeWithCoder:(NSCoder *)coder
{
    [coder encodeObject:self.text     forKey:@"text"];
```

・ TONode

```objc
#pragma mark - NSCoding methods

- (instancetype)initWithCoder:(NSCoder *)coder
{
    // TONodeクラスの super class であるNSObjectはNSCodingに対応していないので、これでOK
    self = [self init];

    self.text     = [coder decodeObjectForKey:@"text"];
    self.children = [[coder decodeObjectForKey:@"children"] mutableCopy];

    return self;
}

- (void)encodeWithCoder:(NSCoder *)coder
{
    [coder encodeObject:self.text     forKey:@"text"];
    [coder encodeObject:self.children forKey:@"children"];
}


// ペーストボードへ書き込む型の宣言
- (NSArray<NSString *> *)writableTypesForPasteboard:(NSPasteboard *)pasteboard
{
    return @[TONodesPasteboardType];
}

// ペーストボードに書き込むためにアーカイブする
- (id)pasteboardPropertyListForType:(NSString *)type
{
    id theResult = nil;

    if( [type isEqualToString:TONodesPasteboardType] )
    {
        theResult = [NSKeyedArchiver archivedDataWithRootObject:self];
    }

    return theResult;
}

// ペーストボードから読み込む型の宣言
+ (NSArray<NSString *> *)readableTypesForPasteboard:(NSPasteboard *)pasteboard
{
    return @[TONodesPasteboardType];
}

// ペーストボードからアンアーカイブする
- (nullable id)initWithPasteboardPropertyList:(id)propertyList ofType:(NSString *)type
{
    id theResult = nil;

    if( [type isEqualToString:TONodesPasteboardType] )
    {
        theResult = [NSKeyedUnarchiver unarchiveObjectWithData:propertyList];
    }

    return theResult;
}
```

TONodeのシリアライズ

TONodeと
PasteBoard

TODocumentの修正

TODocument

```
@interface TODocument : NSDocument

#pragma mark - class methods

#pragma mark - init methods

#pragma mark - dealloc

#pragma mark - NSCopying, hash, isEqual:

#pragma mark - NSCoding methods

#pragma mark - restorableState methods

#pragma mark - life cycle methods

#pragma mark - action methods

#pragma mark - event handling methods

#pragma mark - drawing methods

#pragma mark - delegate/datasource methods

#pragma mark - accessor methods (in pairs)

@property (strong) NSMutableArray* rootNodes; <— これを追加

#pragma mark - Utility methods

@end
```

・ TODocument

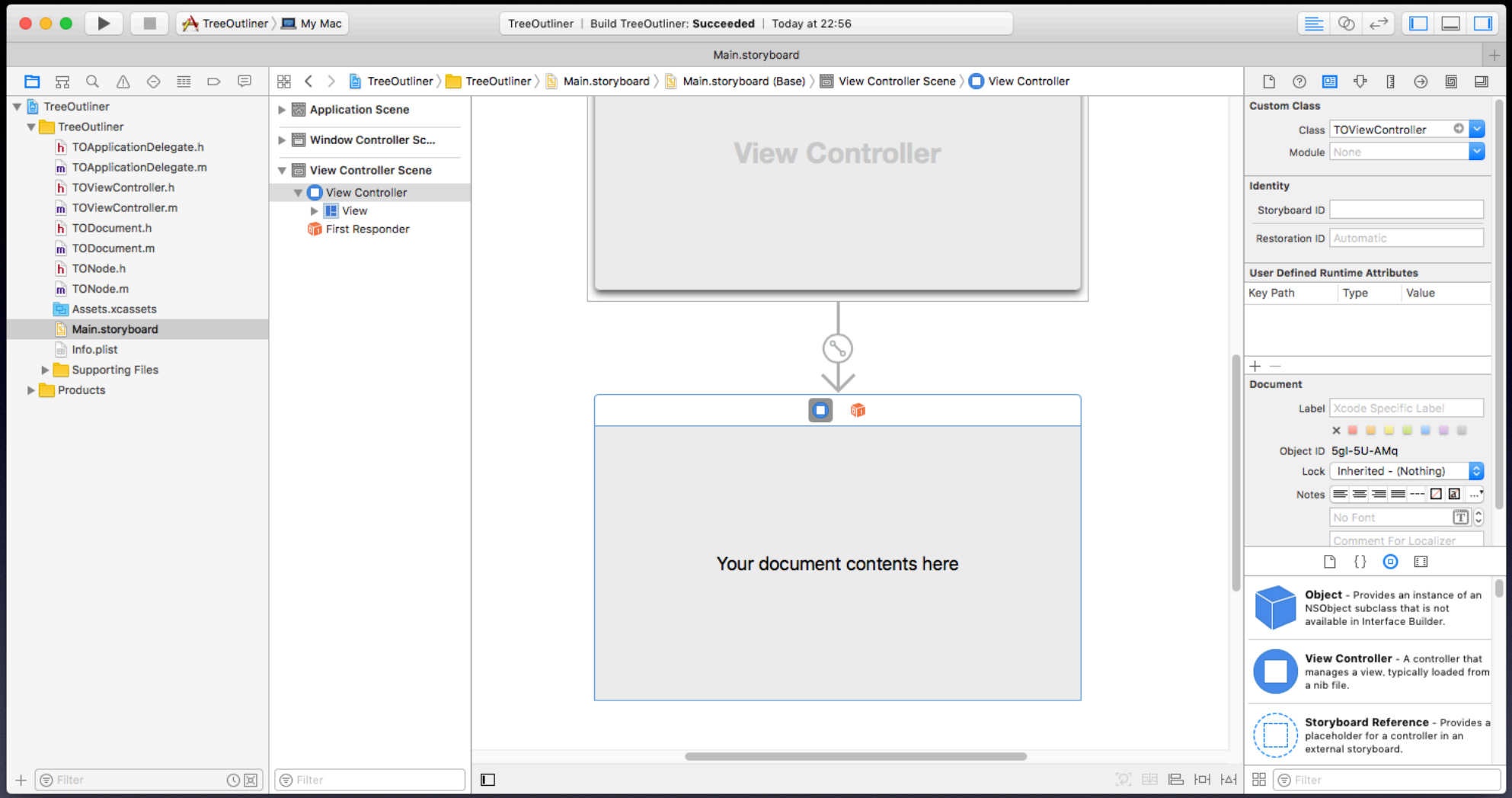
```
#pragma mark - class methods

+ (BOOL)autosavesInPlace
{
    return YES;
}

+ (BOOL)autosavesDrafts  ＜〜ーこれも追加するとデバッグが楽になるぞ
{
    return YES;
}


- (instancetype)init
{
    self = [super init];

    if (self)
    {
        self.rootNodes = @[].mutableCopy;  ＜〜ーこれを追加
    }

    return self;
}

#pragma mark - dealloc

- (void)dealloc
{
    self.rootNodes = nil;＜ーこれも追加。なくても良いけど、デバッグ時に楽
}
```

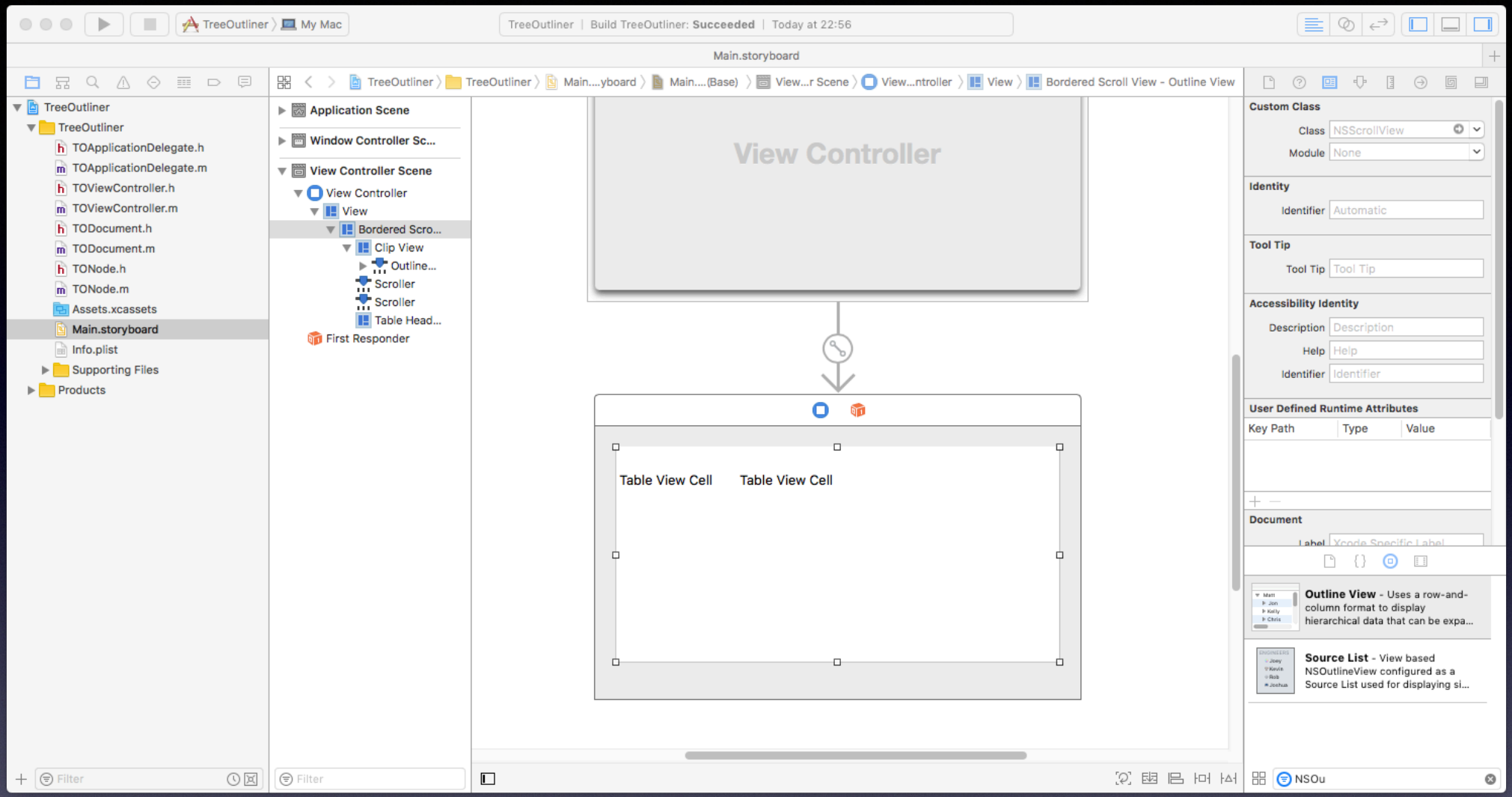
TODocument

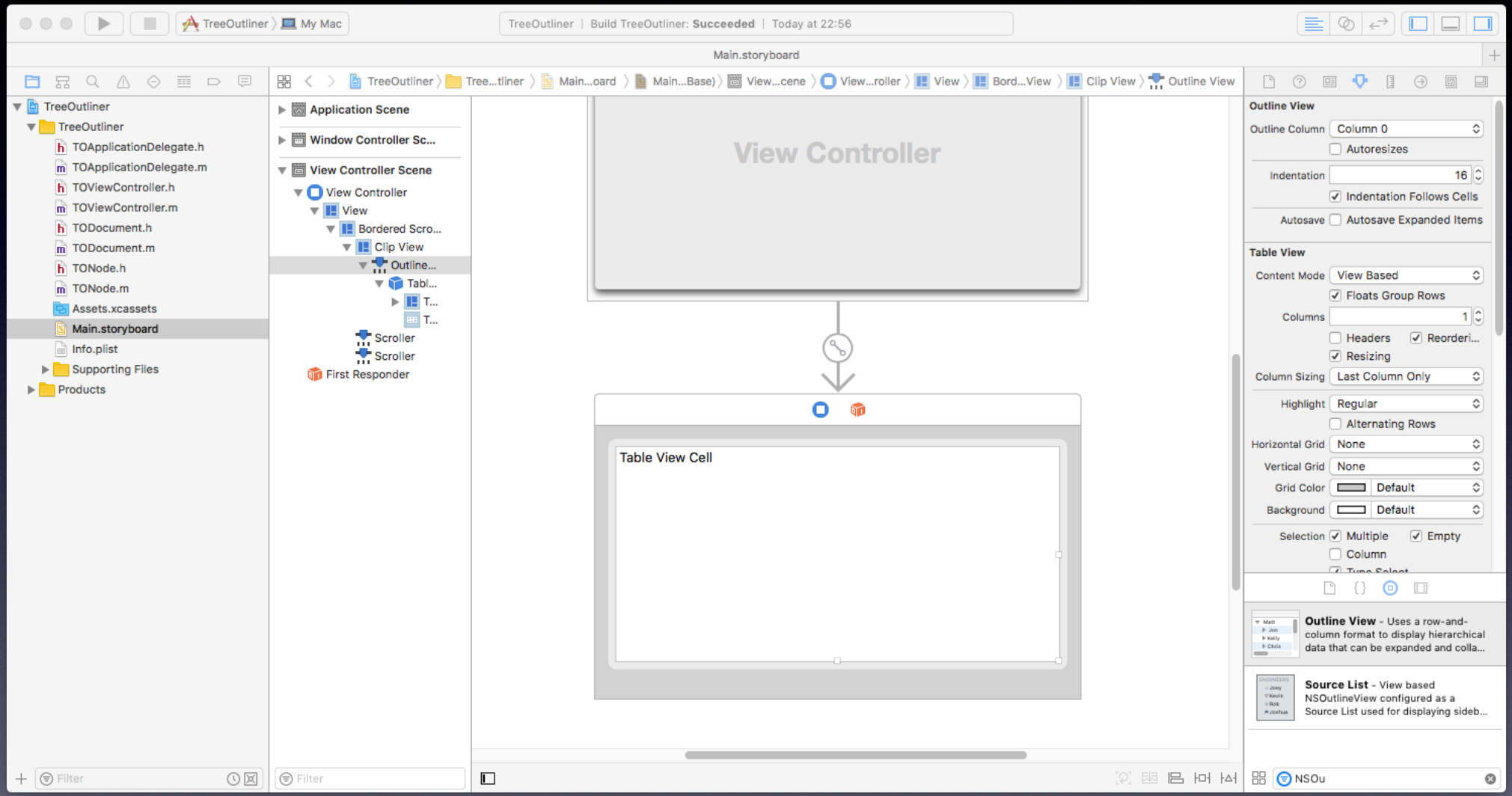
```
#pragma mark - drawing methods

#pragma mark - file handling methods

- (NSData *)dataOfType:(NSString *)typeName error:(NSError **)outError
{
    // アーカイブする
    NSData* theData = [NSKeyedArchiver archivedDataWithRootObject:self.rootNodes];

    return theData;
}

- (BOOL)readFromData:(NSData *)data ofType:(NSString *)typeName error:(NSError **)outError
{
    // アンアーカイブする
    self.rootNodes = [[NSKeyedUnarchiver unarchiveObjectWithData:data] mutableCopy];

    return YES;
}



#pragma mark - Utility methods

- (void)makeWindowControllers
{
    NSWindowController* theWindowController;

    // Override to return the Storyboard file name of the document.
    theWindowController = [[NSStoryboard storyboardWithName:@"Main" bundle:nil]
instantiateControllerWithIdentifier:@"Document Window Controller"]];

    [self addWindowController:theWindowController];

    // ここで、コンテンツデータをViewControllerへ渡す。
    theWindowController.contentViewController.representedObject = self;＜ーーこの1行がすごい大事
}
```

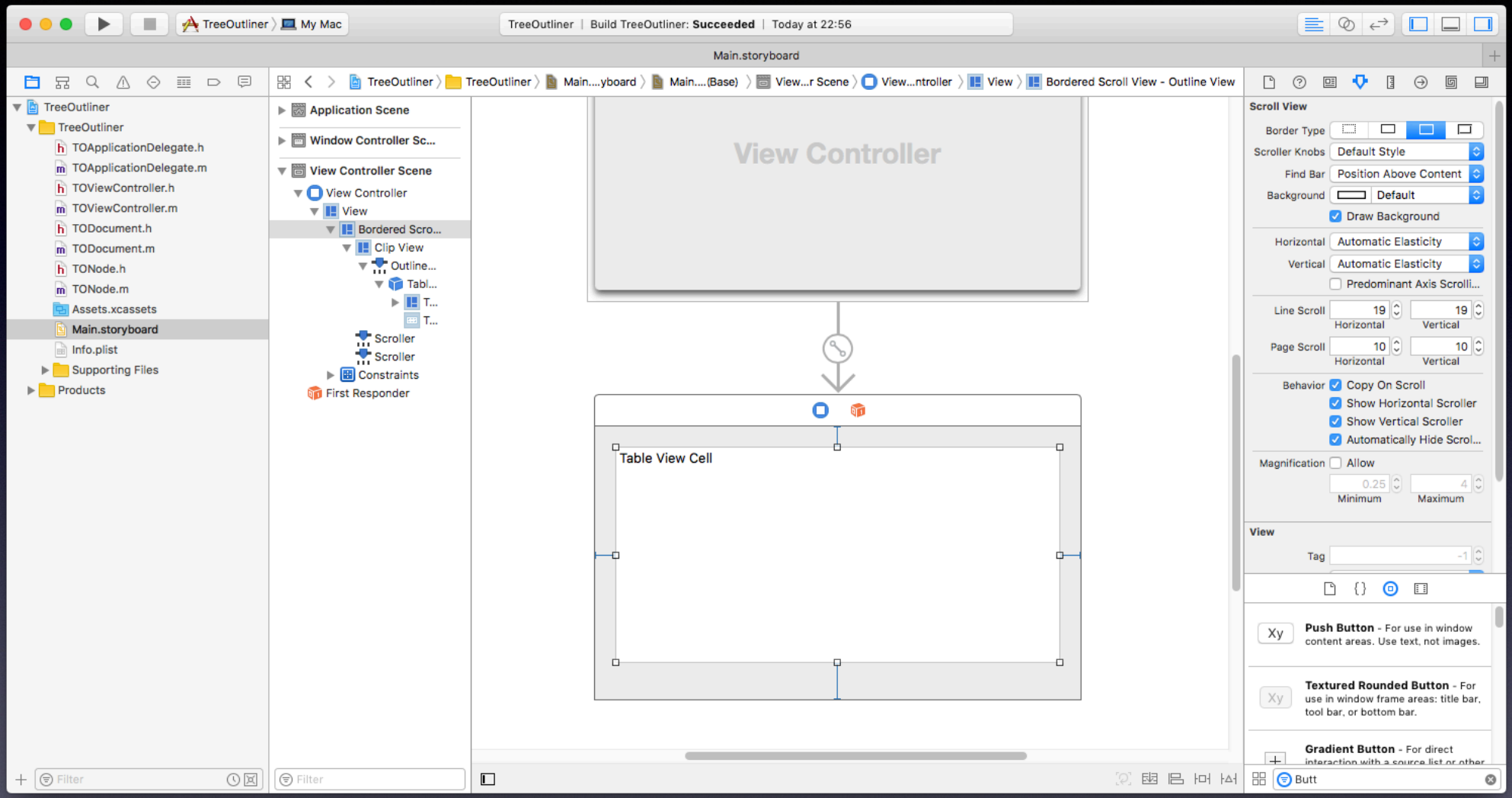
documentの保存と再生

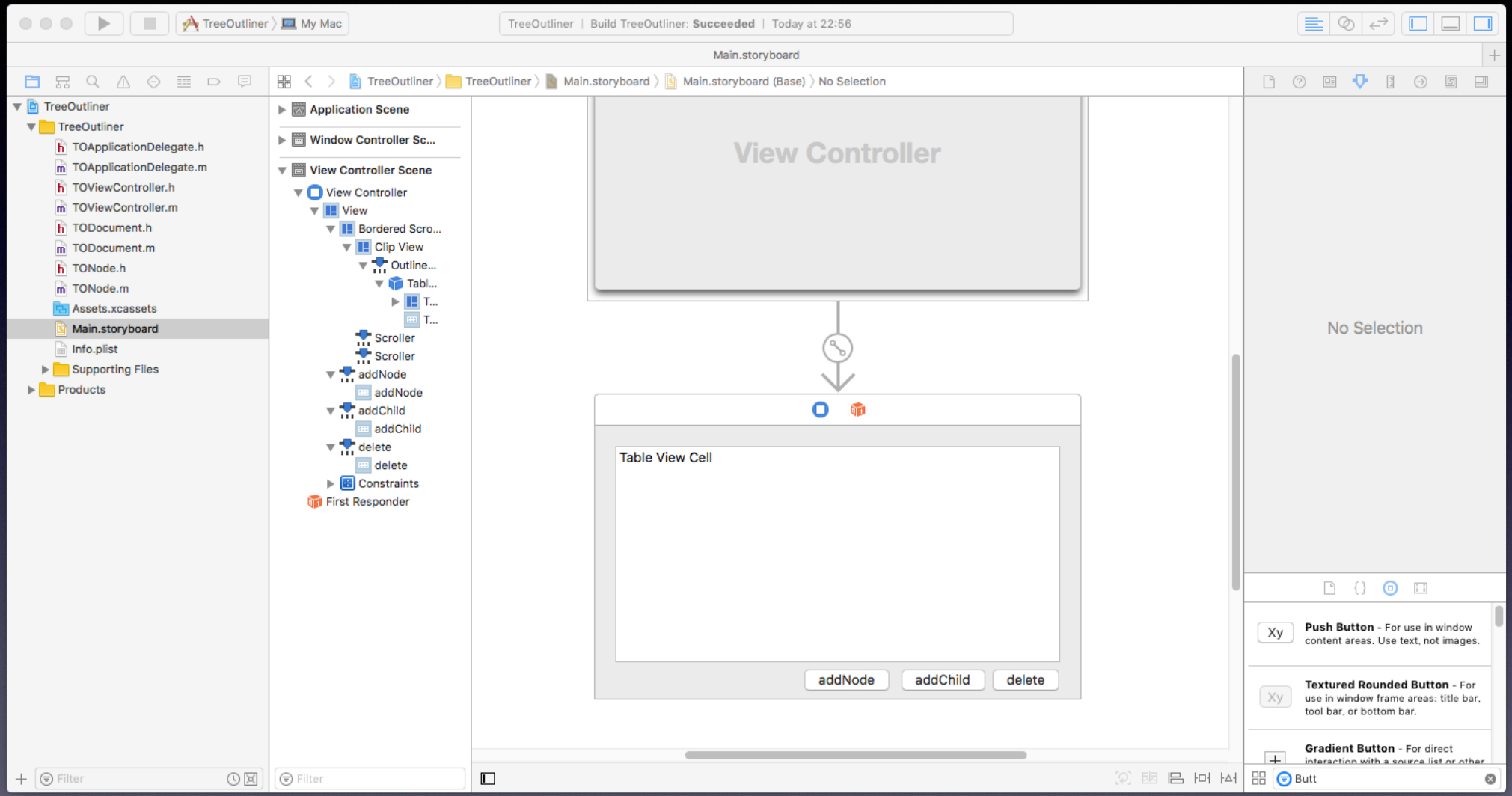
NSOutlineViewとTreeControllerの配置と設定

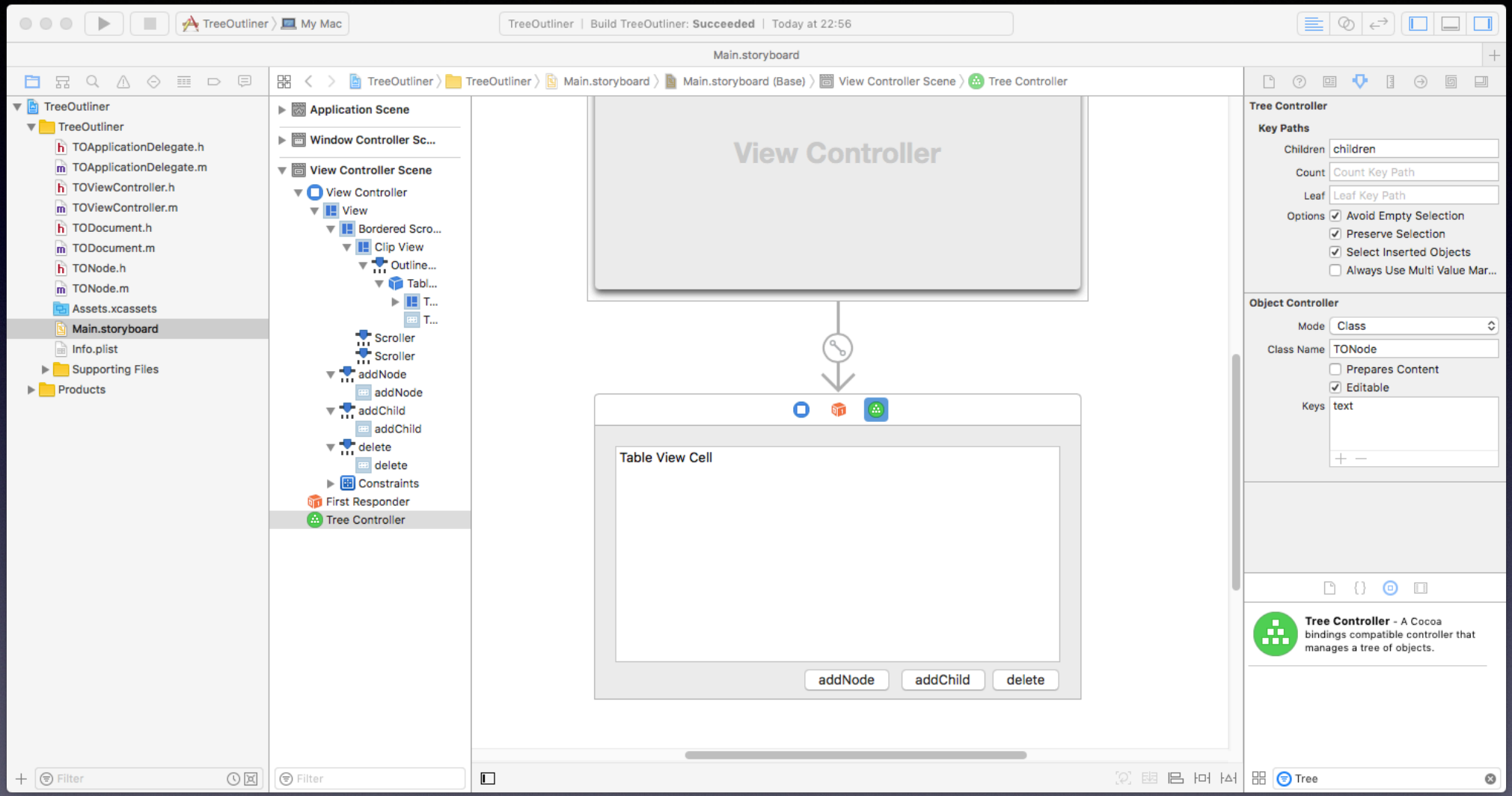

TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
First Responder
View Controller
Your document contents here
Custom Class
Class TOViewController
Module None
Identity
Storyboard ID
Restoration ID Automatic
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID 5gI-5U-AMq
Lock Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Object - Provides an instance of an NSObject subclass that is not available in Interface Builder.
View Controller - A controller that manages a view, typically loaded from a nib file.
Storyboard Reference - Provides a placeholder for a controller in an external storyboard.

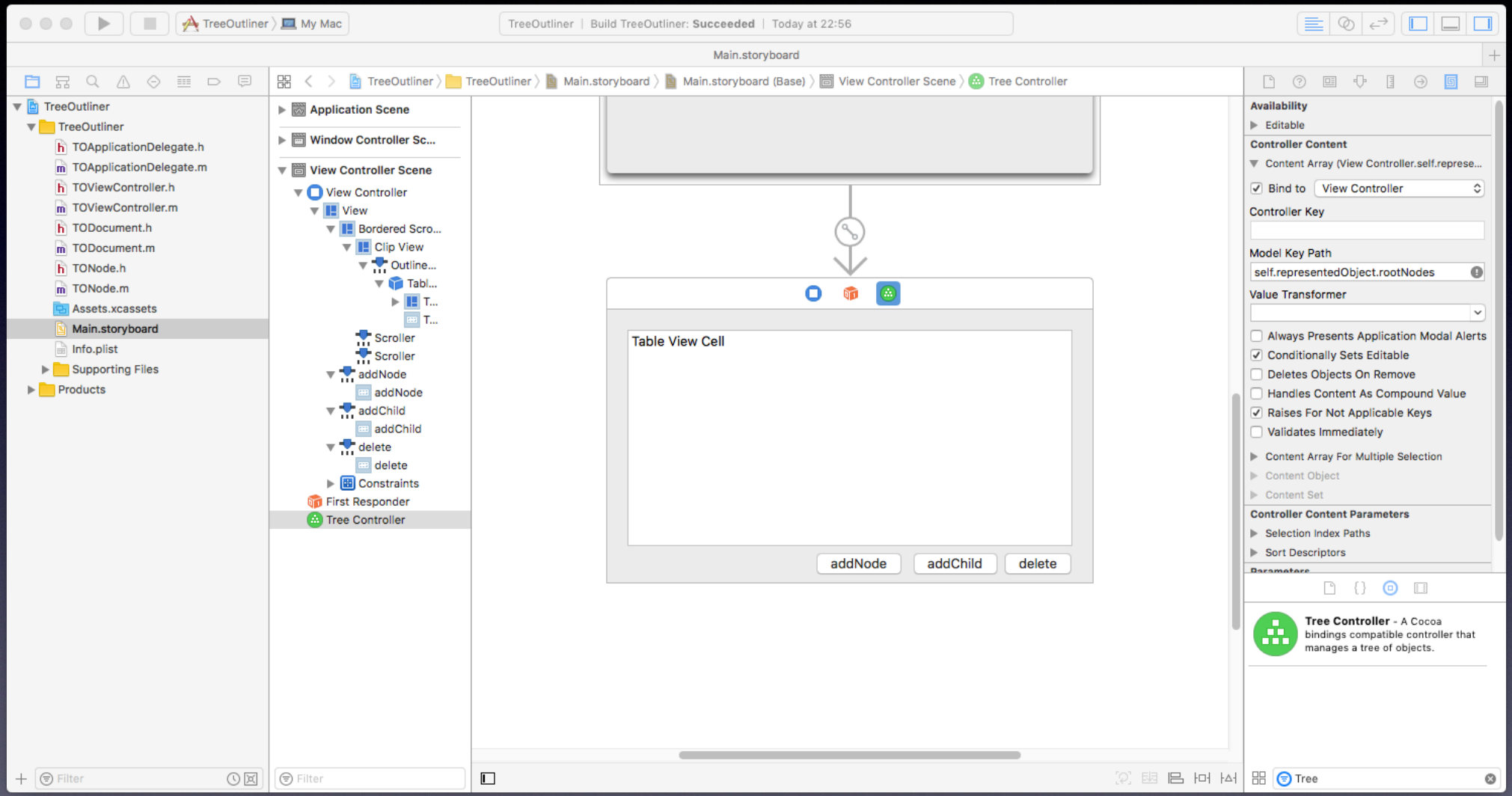

TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Scroller
Scroller
Table Head...
First Responder
View Controller
Table View Cell
Table View Cell
Custom Class
Class NSScrollView
Module None
Identity
Identifier Automatic
Tool Tip
Tool Tip Tool Tip
Accessibility Identity
Description Description
Help Help
Identifier Identifier
User Defined Runtime Attributes
Key Path
Type
Value
Document
Outline View - Uses a row-and-column format to display hierarchical data that can be expa...
Source List - View based NSOutlineView configured as a Source List used for displaying si...

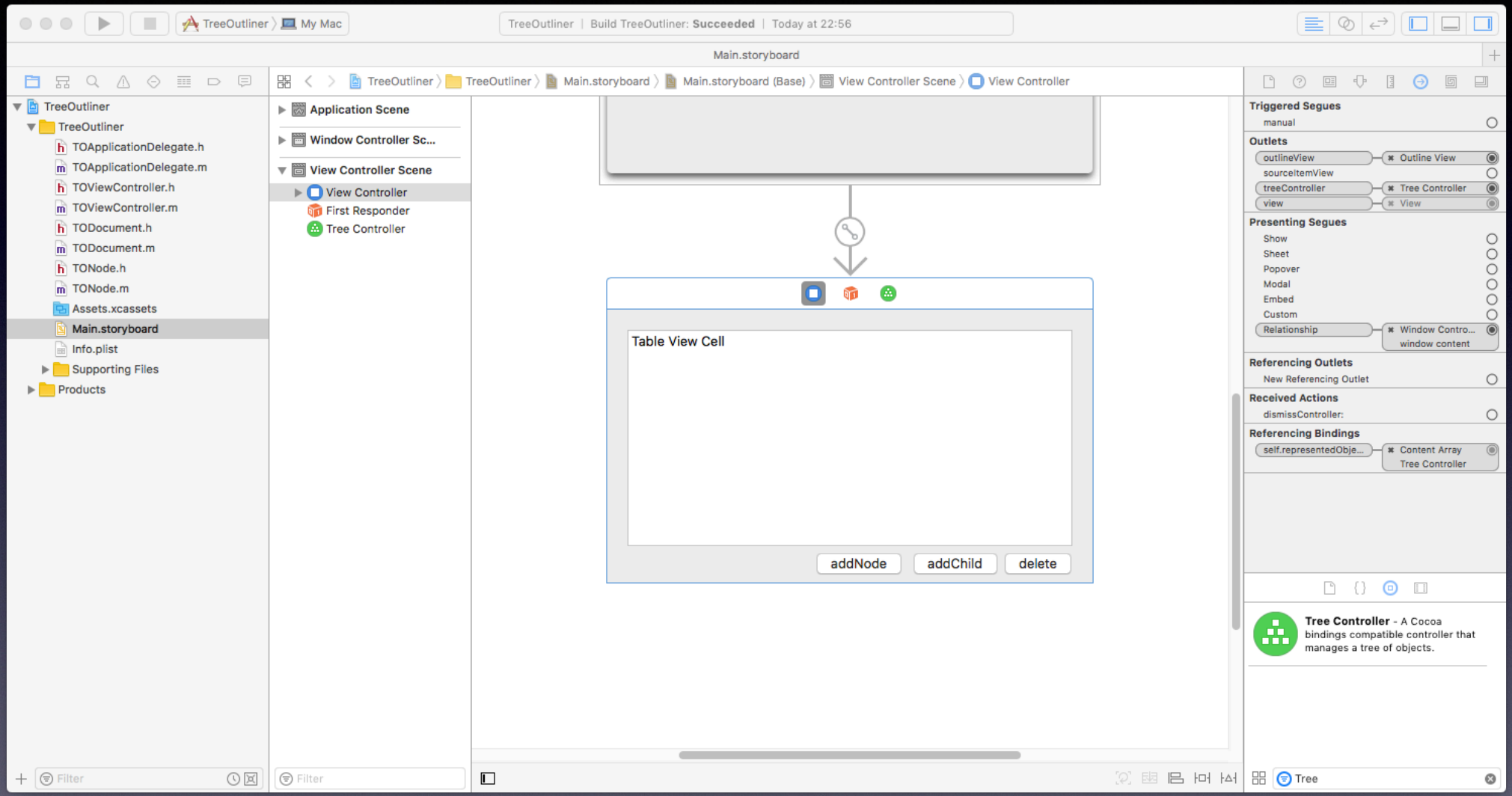

TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Tabl...
Scroller
Scroller
First Responder
View Controller
Table View Cell
Outline View
Outline Column
Column 0
Autoresizes
Indentation
16
Indentation Follows Cells
Autosave
Autosave Expanded Items
Table View
Content Mode
View Based
Floats Group Rows
Columns
1
Headers
Reorderi...
Resizing
Column Sizing
Last Column Only
Highlight
Regular
Alternating Rows
Horizontal Grid
None
Vertical Grid
None
Grid Color
Default
Background
Default
Selection
Multiple
Empty
Column
Outline View - Uses a row-and-column format to display hierarchical data that can be expanded and colla...
Source List - View based NSOutlineView configured as a Source List used for displaying sideb...

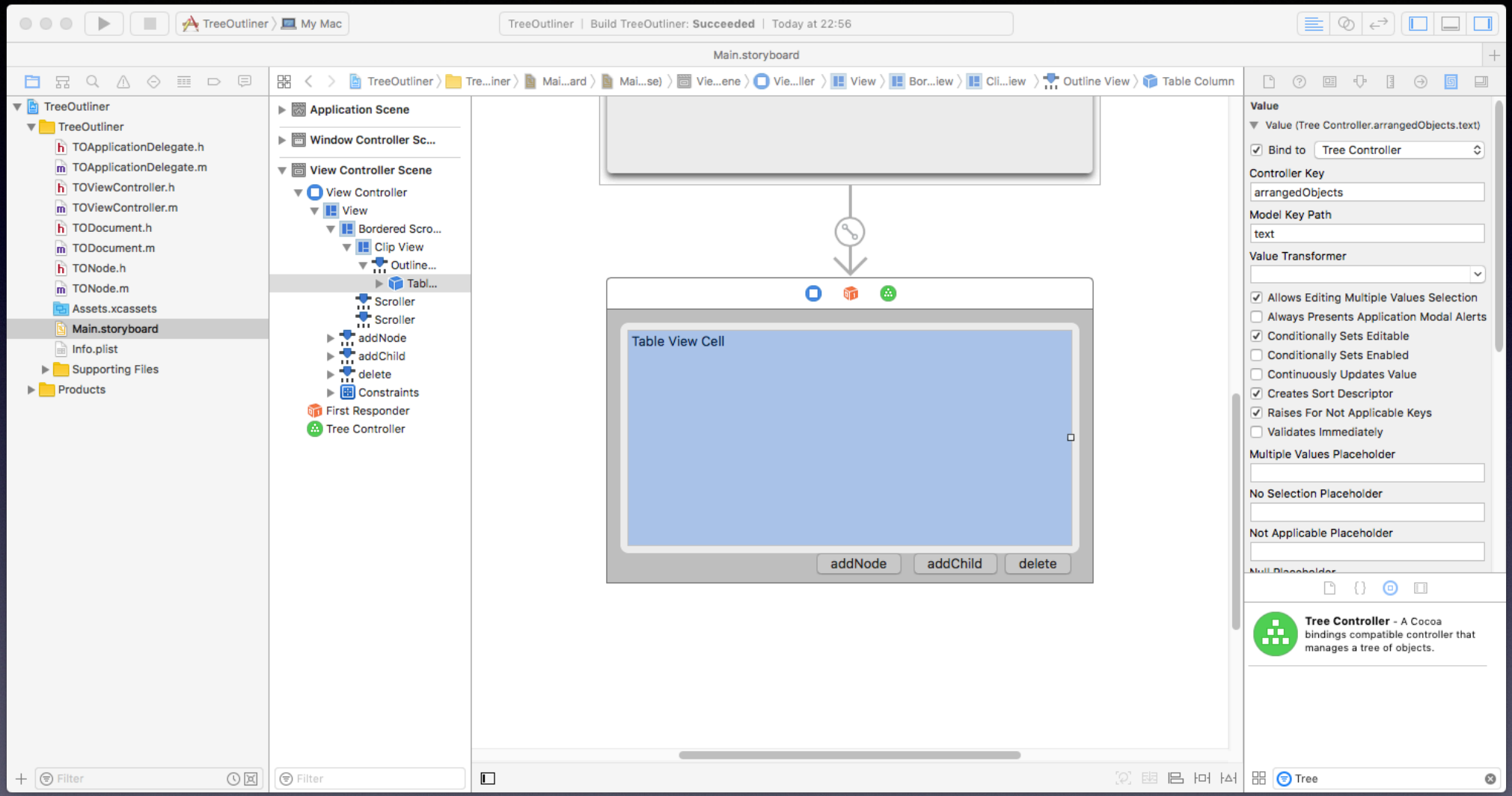

TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Tabl...
T...
T...
Scroller
Scroller
Constraints
First Responder
View Controller
Table View Cell
Scroll View
Border Type
Scroller Knobs
Default Style
Find Bar
Position Above Content
Background
Default
Draw Background
Horizontal
Automatic Elasticity
Vertical
Automatic Elasticity
Predominant Axis Scrolli...
Line Scroll
19
19
Horizontal
Vertical
Page Scroll
10
10
Horizontal
Vertical
Behavior
Copy On Scroll
Show Horizontal Scroller
Show Vertical Scroller
Automatically Hide Scrol...
Magnification
Allow
0.25
4
Minimum
Maximum
View
Tag
Push Button - For use in window content areas. Use text, not images.
Textured Rounded Button - For use in window frame areas: title bar, tool bar, or bottom bar.
Gradient Button - For direct
Butt


TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Tabl...
T...
T...
Scroller
Scroller
addNode
addNode
addChild
addChild
delete
delete
Constraints
First Responder
TreeOutliner
TreeOutliner
Main.storyboard
Main.storyboard (Base)
No Selection
View Controller
Table View Cell
addNode
addChild
delete
No Selection
Push Button - For use in window content areas. Use text, not images.
Textured Rounded Button - For use in window frame areas: title bar, tool bar, or bottom bar.
Gradient Button - For direct
Filter
Filter
Butt


TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Tabl...
T...
T...
Scroller
Scroller
addNode
addNode
addChild
addChild
delete
delete
Constraints
First Responder
Tree Controller
TreeOutliner
TreeOutliner
Main.storyboard
Main.storyboard (Base)
View Controller Scene
Tree Controller
View Controller
Table View Cell
addNode
addChild
delete
Tree Controller
Key Paths
Children
children
Count
Count Key Path
Leaf
Leaf Key Path
Options
Avoid Empty Selection
Preserve Selection
Select Inserted Objects
Always Use Multi Value Mar...
Object Controller
Mode
Class
Class Name
TONode
Prepares Content
Editable
Keys
text
Tree Controller - A Cocoa bindings compatible controller that manages a tree of objects.
Filter
Filter
Tree


TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Tabl...
T...
T...
Scroller
Scroller
addNode
addNode
addChild
addChild
delete
delete
Constraints
First Responder
Tree Controller
Table View Cell
addNode
addChild
delete
Availability
Editable
Controller Content
Content Array (View Controller.self.represe...
Bind to
View Controller
Controller Key
Model Key Path
self.representedObject.rootNodes
Value Transformer
Always Presents Application Modal Alerts
Conditionally Sets Editable
Deletes Objects On Remove
Handles Content As Compound Value
Raises For Not Applicable Keys
Validates Immediately
Content Array For Multiple Selection
Content Object
Content Set
Controller Content Parameters
Selection Index Paths
Sort Descriptors
Tree Controller - A Cocoa bindings compatible controller that manages a tree of objects.
Tree


TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
First Responder
Tree Controller
Table View Cell
addNode
addChild
delete
Triggered Segues
manual
Outlets
outlineView
Outline View
sourceItemView
treeController
Tree Controller
view
View
Presenting Segues
Show
Sheet
Popover
Modal
Embed
Custom
Relationship
Window Contro...
window content
Referencing Outlets
New Referencing Outlet
Received Actions
dismissController:
Referencing Bindings
self.representedObje...
Content Array
Tree Controller
Tree Controller - A Cocoa bindings compatible controller that manages a tree of objects.
Tree


TreeOutliner
My Mac
TreeOutliner | Build TreeOutliner: Succeeded | Today at 22:56
Main.storyboard
TreeOutliner
TreeOutliner
TOApplicationDelegate.h
TOApplicationDelegate.m
TOViewController.h
TOViewController.m
TODocument.h
TODocument.m
TONode.h
TONode.m
Assets.xcassets
Main.storyboard
Info.plist
Supporting Files
Products
Application Scene
Window Controller Sc...
View Controller Scene
View Controller
View
Bordered Scro...
Clip View
Outline...
Tabl...
Scroller
Scroller
addNode
addChild
delete
Constraints
First Responder
Tree Controller
Table View Cell
addNode
addChild
delete
Value
Value (Tree Controller.arrangedObjects.text)
Bind to
Tree Controller
Controller Key
arrangedObjects
Model Key Path
text
Value Transformer
Allows Editing Multiple Values Selection
Always Presents Application Modal Alerts
Conditionally Sets Editable
Conditionally Sets Enabled
Continuously Updates Value
Creates Sort Descriptor
Raises For Not Applicable Keys
Validates Immediately
Multiple Values Placeholder
No Selection Placeholder
Not Applicable Placeholder
Tree Controller - A Cocoa bindings compatible controller that manages a tree of objects.
Tree

ボタンは適当に接続してください。
面倒になってきた

Action

項目の追加と削除

TOViewController

```
#pragma mark - action methods

- (IBAction)add:(nullable id)sender;
- (BOOL) canAdd:(nullable
id<NSValidatedUserInterfaceItem>)item;

- (IBAction)remove:(nullable id)sender;
- (BOOL) canRemove:(nullable
id<NSValidatedUserInterfaceItem>)item;

- (IBAction)addChild:(nullable id)sender;
- (BOOL) canAddChild:(nullable
id<NSValidatedUserInterfaceItem>)item;
```

```
- (IBAction)add:(nullable id)sender
{
    [self.treeController add:sender];
}

- (BOOL) canAdd:(nullable
id<NSValidatedUserInterfaceItem>)item
{
    return [self.treeController canAdd];
}

- (IBAction)remove:(nullable id)sender
{
    [self.treeController remove:sender];
}

- (BOOL) canRemove:(nullable
id<NSValidatedUserInterfaceItem>)item
{
    return [self.treeController canRemove];
}

- (IBAction)addChild:(nullable id)sender
{
    [self.treeController addChild:sender];
}

- (BOOL) canAddChild:(nullable
id<NSValidatedUserInterfaceItem>)item
{
    return [self.treeController canAddChild];
}
```

Copy&Paste

```objc
- (IBAction)cut:(nullable id)sender
{
    [self copy:sender];
    [self remove:self];
}

- (BOOL) canCut:(nullable id<NSValidatedUserInterfaceItem>)item
{
    return [self canCopy:item] && [self canRemove:item];
}

- (IBAction)copy:(nullable id)sender
{
    NSPasteboard* thePasteboard = [NSPasteboard generalPasteboard];

    [thePasteboard declareTypes:@[TONodesPasteboardType]
                          owner:self];

    // 選択箇所のObjectをペーストボードにシリアライズ化してペーストボードに保存する。
    NSArray<TONode*>* theSelectedNodes = [self.treeController valueForKeyPath:@"selectedNodes.representedObject"];
    NSData* theData = [NSKeyedArchiver archivedDataWithRootObject:theSelectedNodes];

    [thePasteboard setData:theData forType:TONodesPasteboardType];
}

- (BOOL) canCopy:(nullable id<NSValidatedUserInterfaceItem>)item
{
    return self.treeController.selectedObjects.count != 0;
}

- (IBAction)paste:(nullable id)sender
{
    // クリップボードから、Nodeを再生。mutableにしているのは再生すると、コレクションはunmutableになってしまうため。
    NSPasteboard*   thePasteboard = [NSPasteboard generalPasteboard];
    NSData*         theData = [thePasteboard dataForType:TONodesPasteboardType];
    NSMutableArray* theNodeArray = [[NSKeyedUnarchiver unarchiveObjectWithData:theData] mutableCopy];

    // 選択の末尾を算定
    NSIndexPath* theInsertIndexPath = nil;

    if( self.treeController.selectionIndexPaths.count > 0 )
    {
        theInsertIndexPath = [self.treeController.selectionIndexPaths.lastObject indexPathByIncrementLastIndex];
```

Copy&Paste

```
    return self.treeController.selectedObjects.count != 0;
}

- (IBAction)paste:(nullable id)sender
{
    // クリップボードから、Nodeを再生。mutableにしているのは再生すると、コレクションはunmutableになってしまうため。
    NSPasteboard*   thePasteboard = [NSPasteboard generalPasteboard];
    NSData*         theData = [thePasteboard dataForType:TONodesPasteboardType];
    NSMutableArray* theNodeArray = [[NSKeyedUnarchiver unarchiveObjectWithData:theData] mutableCopy];

    // 選択の末尾を算定
    NSIndexPath* theInsertIndexPath = nil;

    if( self.treeController.selectionIndexPaths.count > 0 )
    {
        theInsertIndexPath = [self.treeController.selectionIndexPaths.lastObject indexPathByIncrementLastIndex];
    }
    else
    {
        theInsertIndexPath = [NSIndexPath indexPathWithIndex:0];
    }

    // theNodeArrayの個数に合わせてindexPathのarrayを作る
    NSArray<NSIndexPath*>* theIndexPaths = [theInsertIndexPath incrementalIndexPathsAtLength:theNodeArray.count];

    // 末尾に挿入
    [self.treeController insertObjects:theNodeArray atArrangedObjectIndexPaths:theIndexPaths];
}

- (BOOL) canPaste:(nullable id<NSValidatedUserInterfaceItem>)item
{
    // ペーストボードに入っているデータがTONodeをシリアライズ化したものであればOK
    NSPasteboard* thePasteboard = [NSPasteboard generalPasteboard];

    return [[thePasteboard types] containsObject:TONodesPasteboardType];

    return YES;
}

- (IBAction)delete:(nullable id)sender
{
    [self remove:sender];
}

- (BOOL) canDelete:(nullable id<NSValidatedUserInterfaceItem>)item
{
    return [self canRemove:item];
}
```

validateUserInterfaceItem:

TOViewController

```
// ツールバーやメニューバーには効くがNSButtonには有効ではない。
- (BOOL)validateUserInterfaceItem:(id<NSValidatedUserInterfaceItem>)item
{
    BOOL theResult = NO;
    SEL  theAction = [item action];

    if( [self respondsToSelector:theAction] )
    {
        NSString* theActionString = NSStringFromSelector(item.action);

        // "actionName:"を"canActionName:"にする。
        NSString* theCanActionString = [NSString stringWithFormat:@"can%@%@",
                                        [[theActionString substringToIndex:1] uppercaseString],// 頭文字は大文字に
                                        [theActionString substringFromIndex:1]]; // 頭文字以外はそのまま使う

        SEL theCanActon = NSSelectorFromString(theCanActionString);

        if( [self respondsToSelector:theCanActon] )
        {
            // やってることは[performSelector:withObjecy:]とほぼ同じ。返り値がBOOLのためにポインターを抜き出している
            typedef BOOL  (*methodFuncPtr)(id, SEL, id);
            methodFuncPtr theFunction = nil;

            theFunction = (methodFuncPtr)[ self methodForSelector:theCanActon];
            theResult   = (*theFunction)(self, theCanActon, item);
        }
        else
        {
            theResult = YES;
        }
    }
    else
    {
        // superにはvalidateUserInterfaceItem:がないので無視
    }

    return theResult;
}
```

Drag&Drop

書くべきメソッド

```
// DnD用のデータ保存
- (id <NSPasteboardWriting>)outlineView:(NSOutlineView *)outlineView pasteboardWriterForItem:(NSTreeNode *)item

// Dragセッション開始
- (void)        outlineView:(NSOutlineView *)outlineView
           draggingSession:(NSDraggingSession *)draggingSession
          willBeginAtPoint:(NSPoint)screenPoint
                  forItems:(NSArray *)draggedItems

// Dragセッション終了
- (void)     outlineView:(NSOutlineView *)outlineView
         draggingSession:(NSDraggingSession *)draggingSession
            endedAtPoint:(NSPoint)screenPoint
               operation:(NSDragOperation)operation


// Drop可否の判定
- (NSDragOperation)          outlineView:(NSOutlineView *)outlineView
                           validateDrop:(id <NSDraggingInfo>)draggingInfo
                           proposedItem:(NSTreeNode *)item
                     proposedChildIndex:(NSInteger)index

// Dropの処理
- (BOOL)outlineView:(NSOutlineView *)outlineView
         acceptDrop:(id <NSDraggingInfo>)draggingInfo
               item:(NSTreeNode *)item
         childIndex:(NSInteger)index
```

Drag開始

```
// Dragセッション開始
- (void)       outlineView:(NSOutlineView *)outlineView
          draggingSession:(NSDraggingSession *)draggingSession
         willBeginAtPoint:(NSPoint)screenPoint
                forItems:(NSArray *)draggedItems
{
    self.draggingItems   = [draggedItems copy];
}
```

Drag終了

```
// Dragセッション終了
- (void)     outlineView:(NSOutlineView *)outlineView
         draggingSession:(NSDraggingSession *)draggingSession
            endedAtPoint:(NSPoint)screenPoint
               operation:(NSDragOperation)operation
{
    // ゴミ箱にDragされたら
    if( operation == NSDragOperationDelete )
    {
        // Drag中のNodeを削除
        [self.treeController removeObjectsAtArrangedObjectIndexPaths:[self.draggingItems valueForKeyPath:@"indexPath"]];
    }

    self.draggingItems   = nil;
}
```

移動

```
case NSDragOperationMove:
{
    NSArray* theCopyDragItems = [self.draggingItems copy];

    // 同じoutlineViewからDragされてきた場合は、移動
    [self.treeController moveNodes:theCopyDragItems toIndexPath:theInsertPointIndexPath];

    theResult = YES;
}
    break;
```

コピー

```
        case NSDragOperationCopy:
        {
            // Drag中の個数に合わせて挿入位置のindexPathのarrayを作る
            NSArray<NSIndexPath*>* theIndexPaths = [[theInsertPointIndexPath indexPathByIncrementLastIndex]
                                                    incrementalIndexPathsAtLength:draggingInfo.numberOfValidItemsForDrop];

            NSMutableArray<TONode*>* theDragItems = [NSMutableArray arrayWithCapacity:theIndexPaths.count];

            // draggingInfoから順繰りにデータを解凍する
            [draggingInfo enumerateDraggingItemsWithOptions:NSDraggingItemEnumerationConcurrent
                                                    forView:outlineView
                                                    classes:@[[NSPasteboardItem class]]
                                              searchOptions:@{}
                                                 usingBlock:^(NSDraggingItem *draggingItem, NSInteger idx, BOOL *stop)
             {
                 NSPasteboardItem* thePasteboradItem = draggingItem.item;

                 if( ![[thePasteboradItem types] containsObject:TONodesPasteboardType] )
                 {
                     *stop = YES;
                 }
                 else
                 {
                     TONode* theNewObject;
                     theNewObject = [[TONode alloc ] initWithPasteboardPropertyList:[thePasteboradItem
dataForType:TONodesPasteboardType]
                                                                             ofType:TONodesPasteboardType];
                     [theDragItems addObject:theNewObject];
                 }
             }];

            [self.treeController insertObjects:theDragItems
                    atArrangedObjectIndexPaths:theIndexPaths];

            theResult = YES;
        }
            break;
```