第4回

「 第4回.メッセージングについて」

toru.inoue@kissaki.tv

タグは#TechHUBJP

Agenda 1/2

・今日やる事：オブジェクト間の通信と、テストの実践

・効能：NSNotificationを使った疎結合がわかる
　iOSでのテストの書き方が判る

・今日やらない事：
　Notification以外の疎結合のレクチャー
　(他にもいろいろな手法や手段が有るよ)

　テストの有用性の証明
　(テストは、意味と価値と効果を意識しないと意味ない。)

Agenda2/2

1.オブジェクト間の情報通信

2.iOSでのテスト

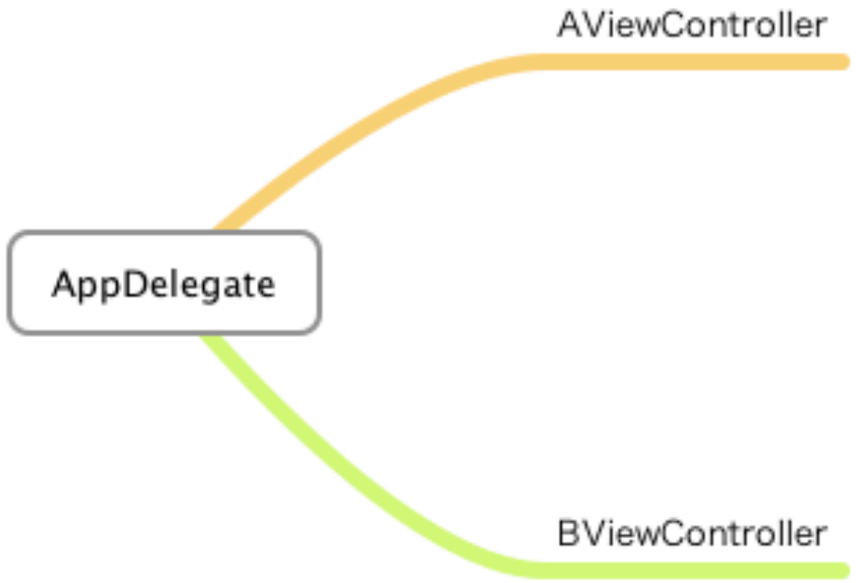
Drive1/19

1.オブジェクト間の情報通信

Drive2/19

1-0-0. オブジェクト間の情報通信とは
・オブジェクトの間で、**お互いに何か影響**を与える事。

「お互いに何か影響」？
詳しく言うと、
AppDelegateがあり、そいつが
クラスAViewControllerのインスタンスaVCont
と
BViewControllerのインスタンスbViewCont
を持ってるとしよう。



この時、AppDelegate内では、[a AViewContのメソッド]とか書ける。
例えばAppDelegate内で、[a.view `removeFromSuperview`]とかしたら、
きっとaで扱ってたビューは画面から消えるんだろう。

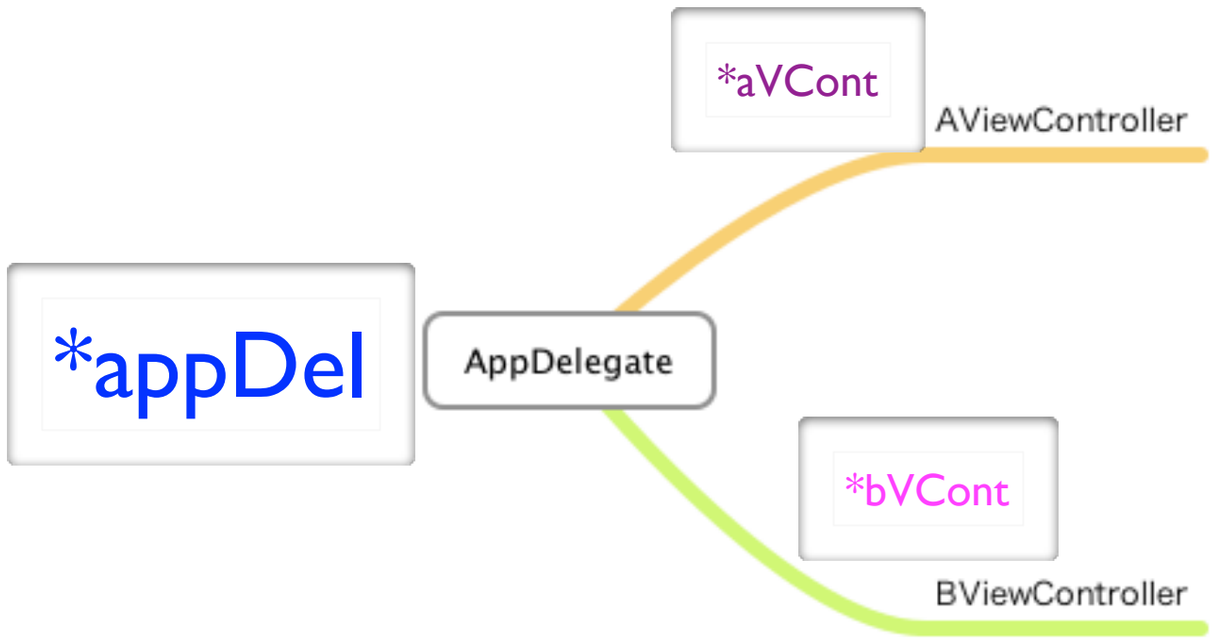
この時、AppDelegateからはaに影響を与えられてる。

Drive3/19

1-0-1.具体例として、2つの画面の切り換え

2画面あるアプリ：AppDelegate　-　ViewControllerA
　　L　-　ViewControllerB

☆ViewControllerをA,B２つ持ってるプロジェクトを作っちゃおう。



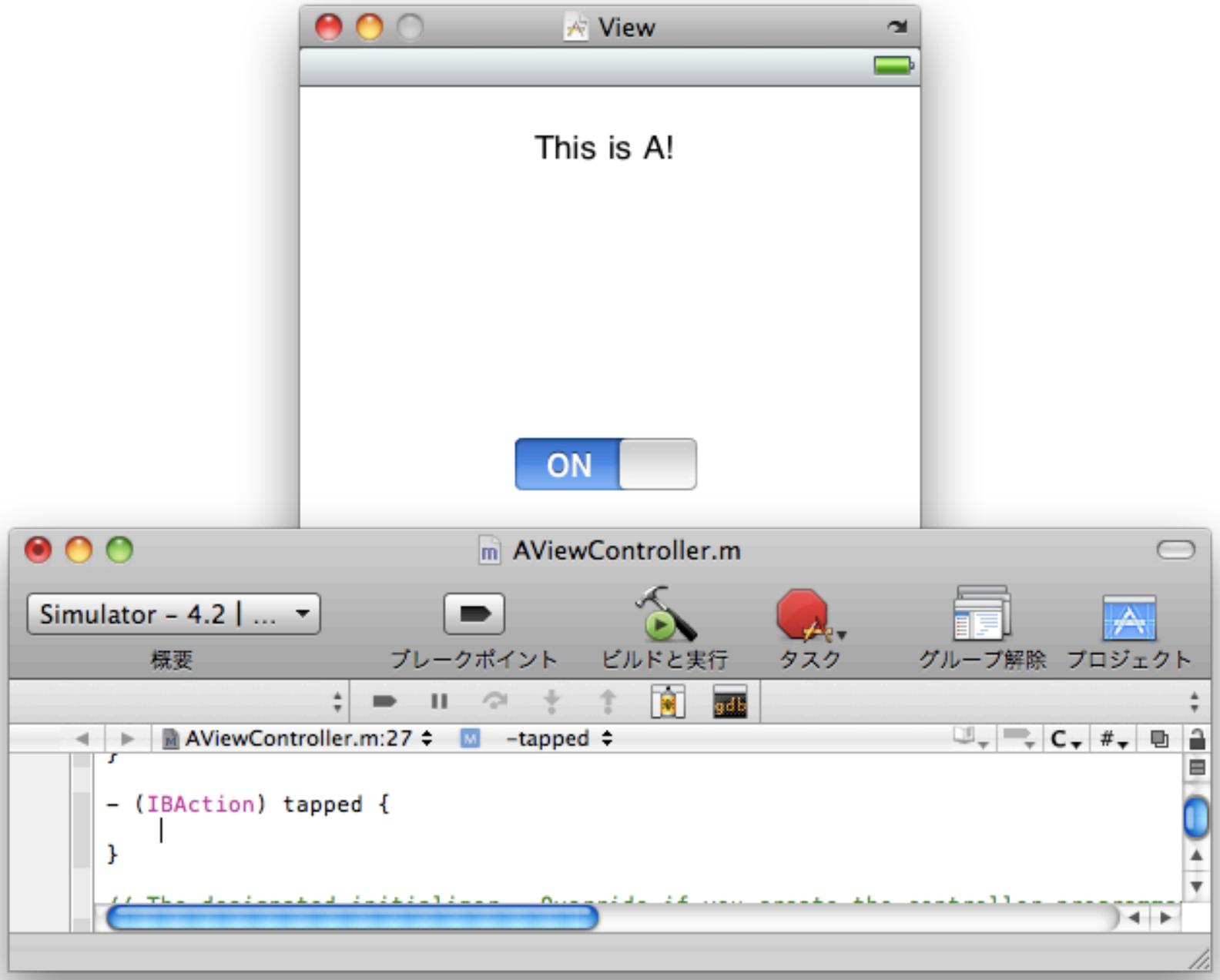
→ここで、
　起動時にAの画面が表示されてて、
　ViewControllerAからなんかインプットがあり、画面を入れ替えたい、としよう。

Drive4/19

1-0-2.画面の切り換えがしたい

画面切り換えの引き金は、Aの画面上のスイッチを押したら、にしよう。

スイッチ付けてIBActionも書く。IBで繋げる。



Drive5/19

1-0-3.画面切り換えの動作

ここで、AppDelegateは`window`を持ってる。

aはaのviewを持ってる。

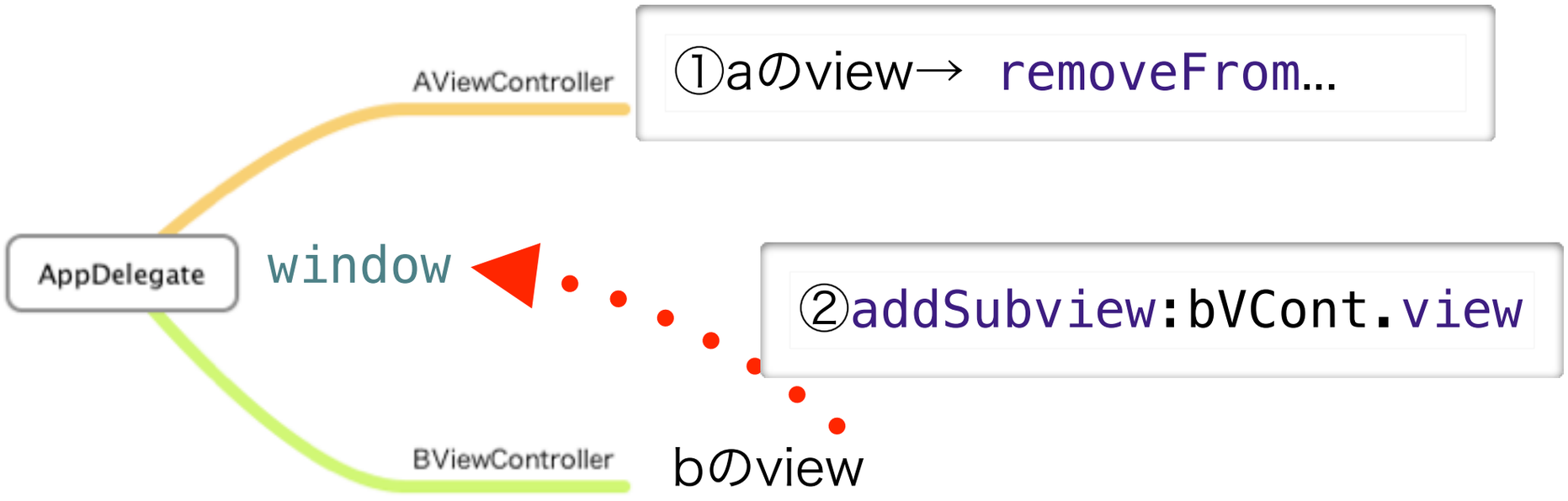
bはbのviewを持ってる。

そうすると、画面を切り替えるには、

Aに対して、`removeFromSuperview`

`window`に対して、bのviewを`addSubview:bVCont.view`

とかできればいい気がする。



Drive6/19

1-0-4.Aのスイッチから何が起こるか

①の、Aに対してremoveFrom..は、簡単。

スイッチ押されたところに[self.view remove..]とかやればいい。

②、Bのviewを window にaddSubview:bVCont.view

これが、大変。

Aは window やBの事を知らない

☆実際には、windowはaViewCont.view.superview=windowなので触れる。

superviewの意味についてはドキュメント見ろ。

であれば、どうするか。ここで無理せず、

bVCont.view と window に触れるオブジェクトに頼もう。

＝AppDelegate。

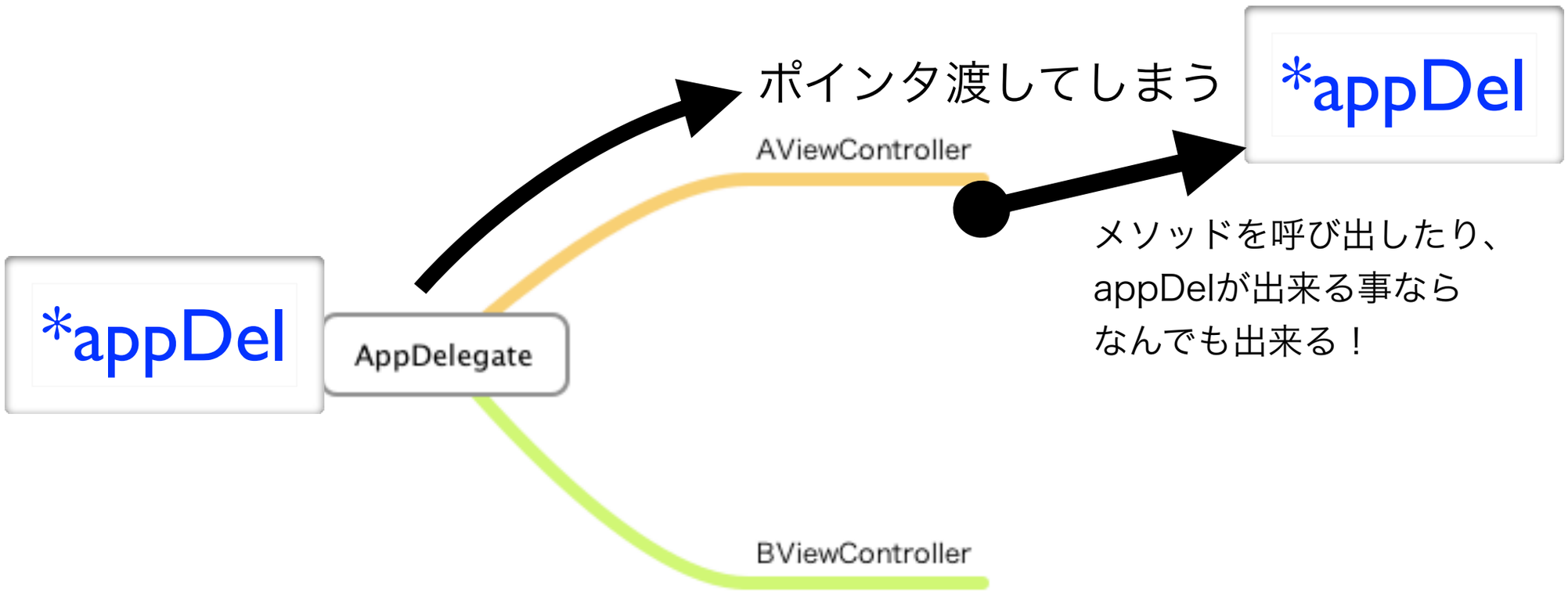
スイッチが押されたら、AppDelegateに伝えよう。

Drive7/19

1-0-5. Aから親(AppDelegate)へと、どうやってイベントを伝えるか

→C言語的な発想として、ポインタを渡す。

ポインタを渡して、
Aから親のポインタ→親のメソッドを呼んで、そのメソッド内でAの `removeFromSuperview` 内とか、Bのviewを `window` に`addSubview:bVCont.view` とか、できる。できるけど、、、、

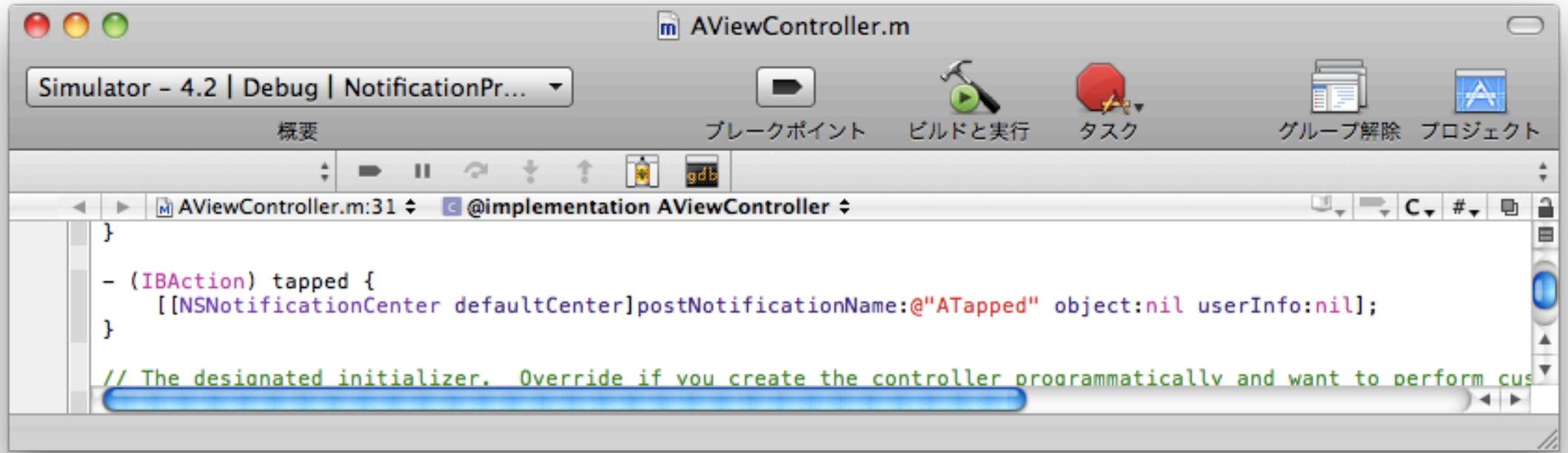


だが、ここでは、この手法は使わない。

Drive8/19

1-0-6.iOSなら、Notification(通知)が使える
　NSNotificationを使うと、オブジェクト間の通信が出来る。

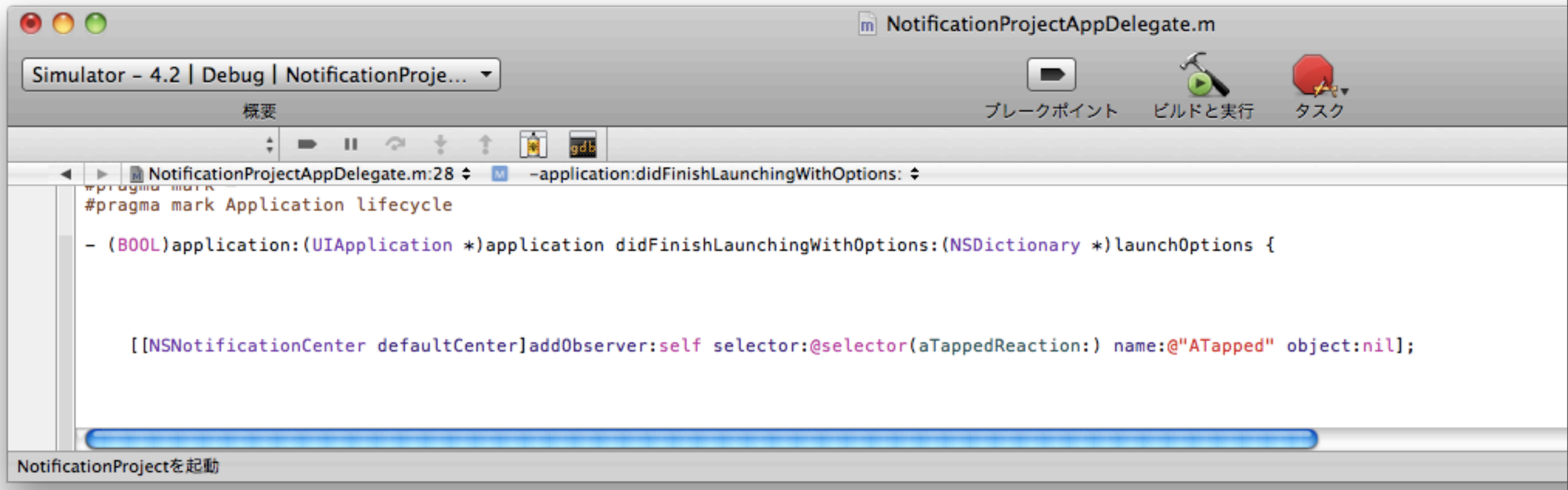
　下記はNSNotificationを使って、
　スイッチが押された事を伝えるコード。



Drive9/19

1-0-7.iOSなら、Notification(通知)が使える(2)

Delegateに、これを受け取るコードを書こう



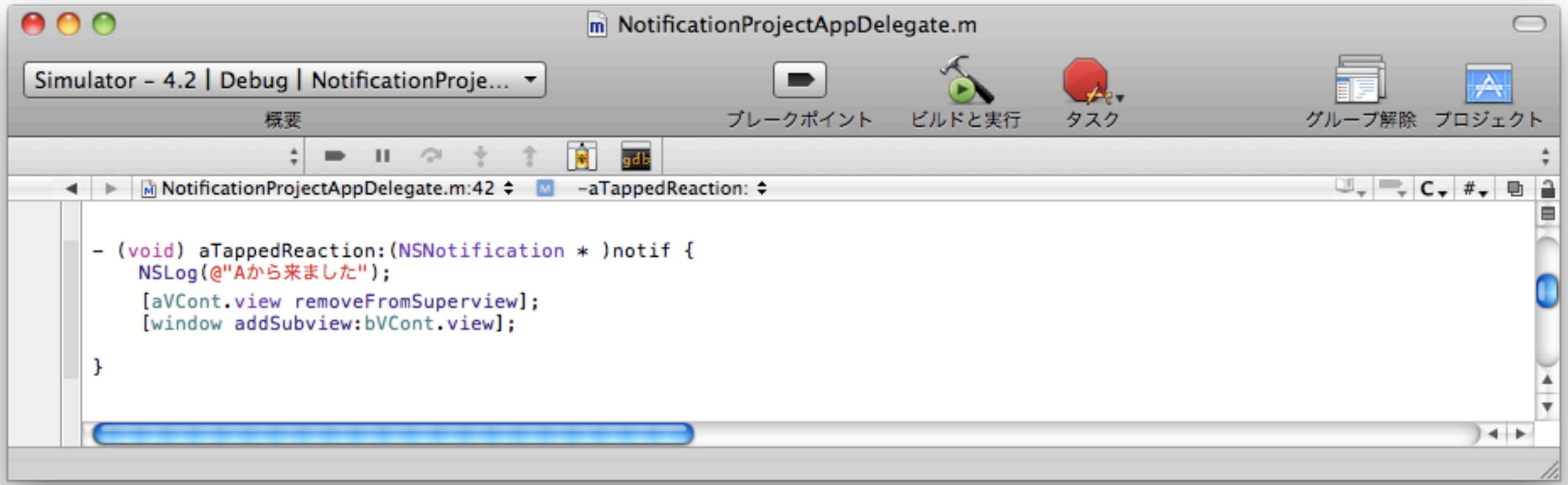
1-0-8. これで、A→AppDelegate間の通信

スイッチを押すと、AppDelegateにあるメソッドが動く。

そしたら、目的を果たそう。

・Aを`removeFromSuperview`

・Bを`[window addSubview:bVCont.view]`



Drive11/19

1-0-9.Notificationの仕組み

ざっくり：

```
[[NSNotificationCenter defaultCenter]addObserver:self
selector:@selector(aTappedReaction:) name:@"ATapped"
object:nil];
```

ATappedという名前で、observerを

[NSNotificationCenter defaultCenter]に追加している。

Drive12/19

1-0-10.Notificationの仕組み

しっとり：

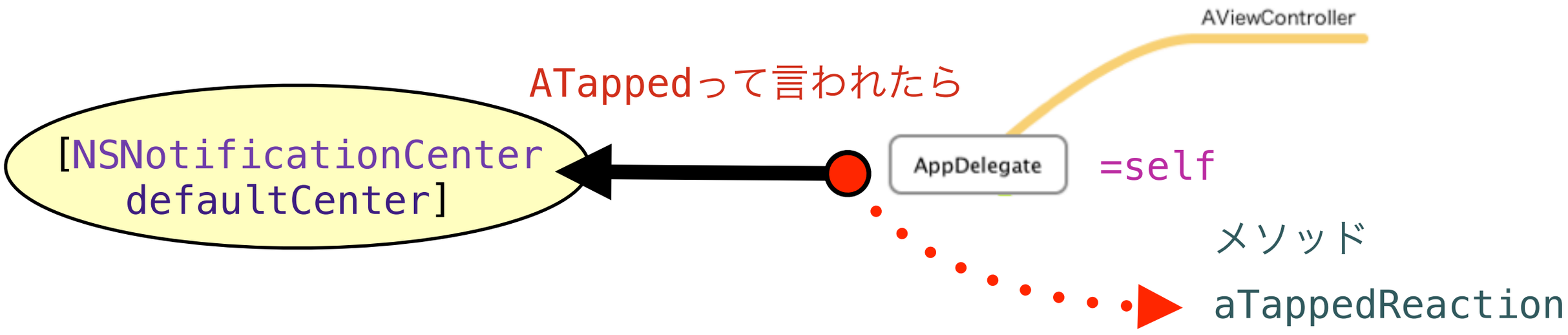
[NSNotificationCenter defaultCenter]

というものが、Objective-CのRuntimeの背景にある。こいつに、

「ATappedって言われた」ら、

「self aTappedReaction メソッドを動かすぜ」、

っていう、Observerをaddしてる。



Drive13/19

1-0-11.Aから、ATappedって言う

```
[[NSNotificationCenter defaultCenter]
postNotificationName:@"ATapped" object:nil userInfo:nil];
```

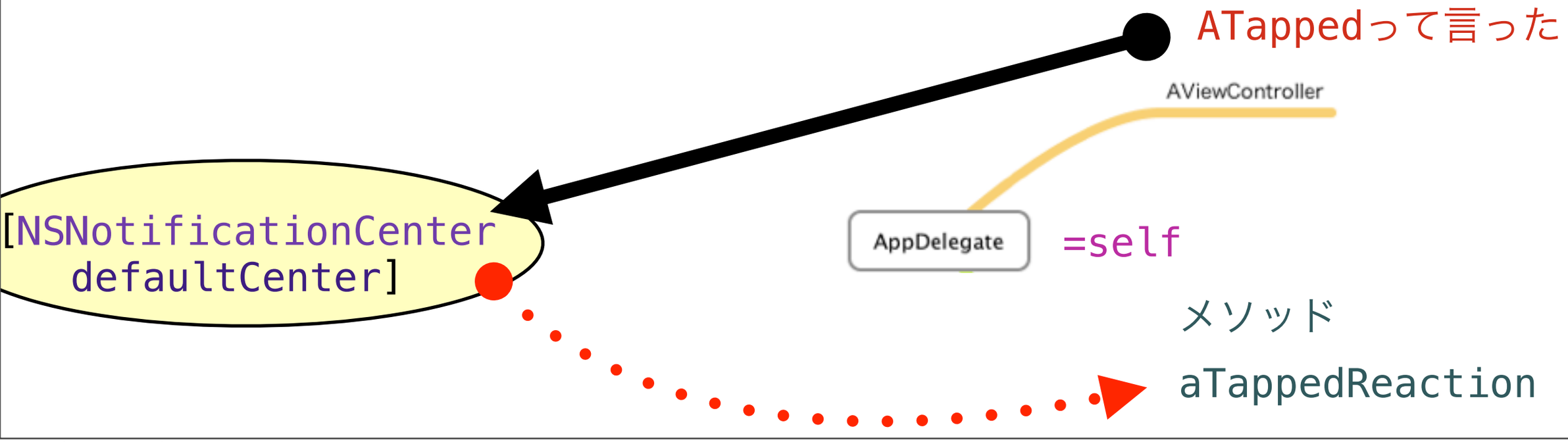
これは、ATappedって名前のNotification(通知)を

[NSNotificationCenter defaultCenter]に対して

post(投函)する、って意味。

するとまあ、先ほどセットしておいた「ATappedって言った」って事に成る。

で、まあ、動く。



Drive14/19

1-0-12.ここにポインタの関係性は無い

A⇄AppDelegate が発生している訳だが、
Aにも、AppDelegateにも、
お互いの事は書いてない。

AppDelegateは`ATapped`って言われたらメソッドを動かすよ、て
設定しただけだし、
Aは`ATapped`って**言っただけ**だ。

Objective-Cでは、こんな風に簡単に、特にお互いを書かずにオブジェクト間の通信ができる。
特に、オブジェクト渡したりしない通信の手法を、**疎結合**って言ったりする。

逆に、ポインタ渡したりガッチガチにオブジェクトを共有する手法を、
密結合って言ったりする。
あくまで加減の問題なので、あんま気にすんな。　(e.g. Notificationに比べてポインタは密結合)

どっちが良いとか無い。ケースバイケース。

Drive15/19

1-0-13.　BからAに戻るのも作ってみようか

手法は判ったと思うので、

今、スイッチ押したらAからAppDelegateに行き、

Aを外し、Bをセットした。

Bにもなんか置いて、何かしたらこんどはBが消えてAが出るようにしようぜ。

そしたらAの画面⇄Bの画面、の遷移が出来る。

Drive16/19

1-0-14.NSNotificationの良いとこと悪いとこ

恩恵としては、

・誰向けとか、相手の事とか、**受取手の事を全く書かずに**コードが書ける
・構造的に、**受取手を追加すればプログラムで出来る事が広がって行く**、
という事が出来る。

機能を足したら、ソレが何時起こって欲しいか、引き金を何処かに書けば良い。

デメリットとしては、

・**誰が受け取るのか判らない**から何が起こるかパッと見判らない

最悪、書いた人すら、全体像を見ないと何してるか判らないコードが出来上がる。

(というか、誰が受け取って何をするか、など、類推するキーが足りなすぎる)
コードとしてポインタから解放され、もの凄く自由になるが、仕様書や、ルール付けが必須になる。

Drive17/19

1-0-15.結論

いろいろなオブジェクト間通信が有るが、はっきり言って好きずき。

ポインタ渡し：

循環参照に気をつけないといけない。

ポインタを渡す事で、関係性を絶対的に明示できる。

反面、コード間の構造がガッチガチになる。変更orz、　追加orz

NSNotification：

プログラムの全体を見ないと、何が起こるか判らない

発信する情報は自由に足せる

受け取る設定を追記すれば幾らでも機能を足せる

Drive18/19

Drive19/19

2-0-0.iOSでのテスト

　　ぶっちゃけ某社のブログが一番詳しい。

http://kissaki-blog.blogspot.com/2011/02/ios.html

LookBack

1.オブジェクト間の情報通信
2.iOSでのテスト

プログラムの質問、提出はこっちな！！
自分の名前と宛名忘れんなよ！！
↓

toru.inoue@kissaki.tv

No credit..

ぶっちゃけ質問コーナーだべ。
あと、第3回時点での、CoreDataのバージョンアップの話とか、出来たらする。