五年後の
Control Enabler

成田丞

mindtools@mac.com

ControlEnabler？

- 5年程前に発表した
- 名前は成田が勝手に付けた
- UIパーツの更新をする
- Cocoa Bindingと同じ目的の別機構


選択箇所の更新
選択箇所の更新
アイコン画像
の更新
CurrencyConverter
90
0
80
90
100
Convert
Rate
Custom View
reset
Exchange Rate per $1:
0
Dollars to Convert:
0
80
90
100
選択箇
所の更新
名前の更新
数値をモデルと同期
Amount in Other Currency:
0.000000
reset
Convert

ここら辺のUIを更新

UIの更新の既存の手法

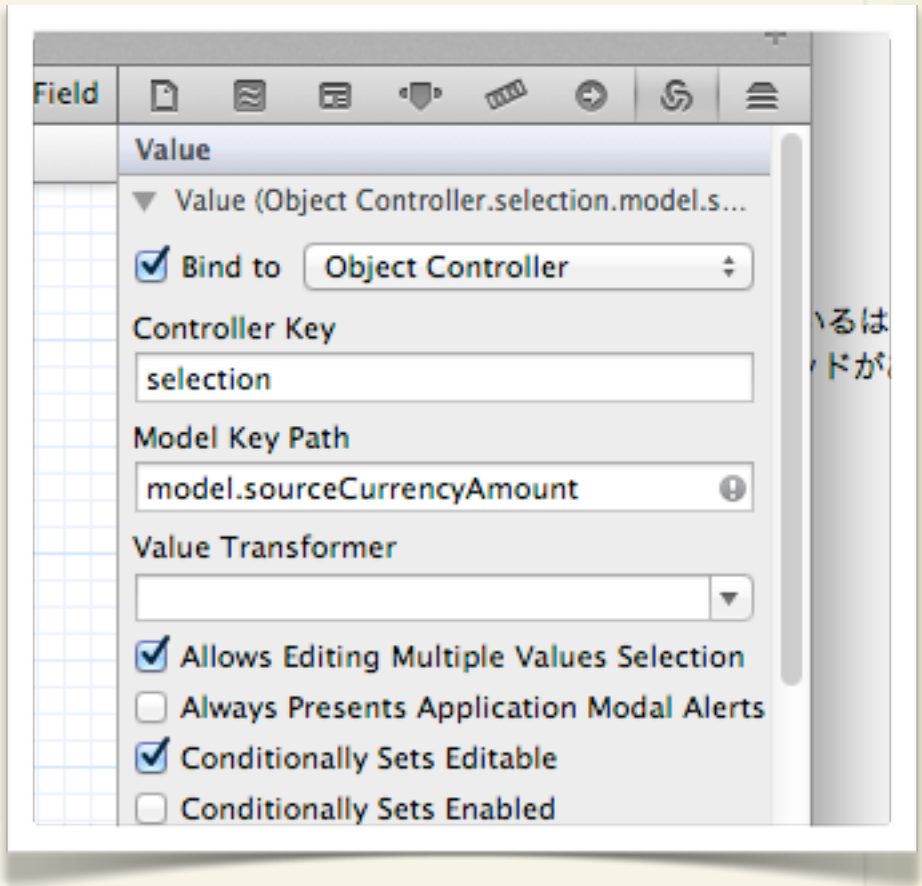
- 手書き
- Cocoa Binding
- NSUserInterfaceValidations

手書き

```
@interface CCAppDelegate : NSObject
   .
   .
@property (assign) IBOutlet NSTextField *rateField;
   .
   .
@end

@implementation CCAppDelegate
- (void) update
{
   .
   .
   [self.rateField setStringValue:@"this is Update."];
   .
   .
}
@end
```

Cocoa Binding

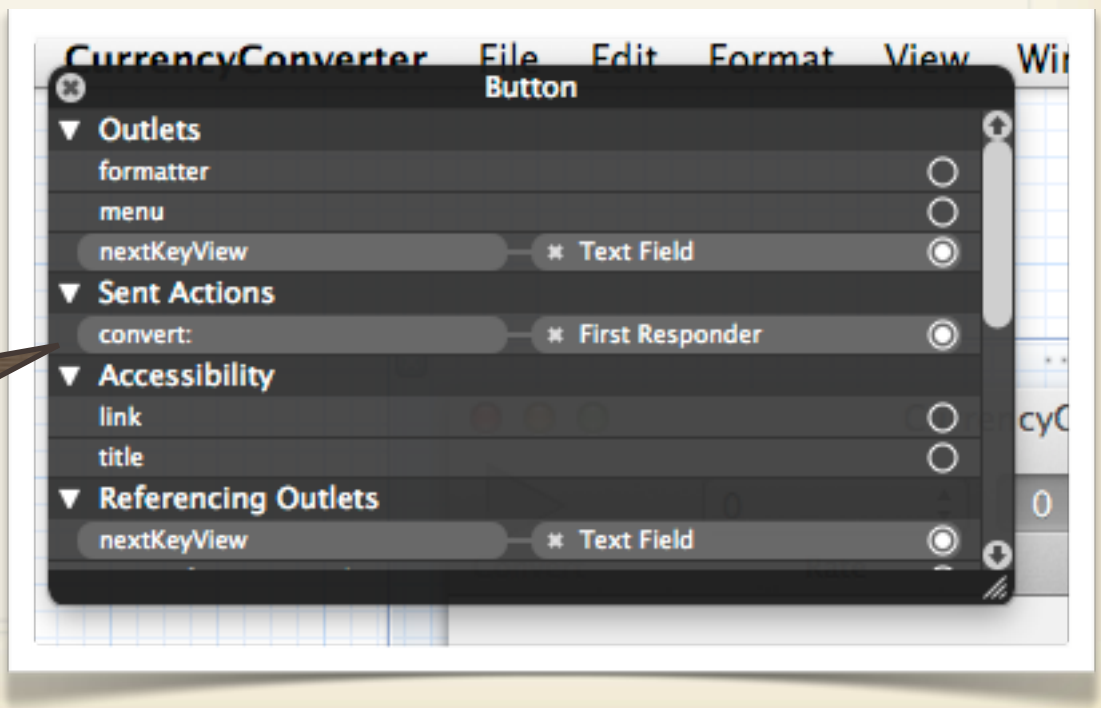
```
@interface CCAppDelegate : NSObject
   .
   .
@property (retain) CCModel *model;
   .
   .
@end

@implementation CCAppDelegate
   .
   .
@end
```



NSUserInterfaceValidation

```
@interface CCAppDelegate : NSObject
-(IBAction)convert:(id)sender;
@end

@implementation CCAppDelegate
-(BOOL)validateUserInterfaceItem(id<NSValidatedUserInterfaceItem>)anItem
{
   if( [anItem action] == @selector(convert:) )
   {
      if( 動作条件 ) return YES;
   }

   return NO;
}
@end
```

ターゲットを自動で探して
くれるのが利点



NSUserInterfaceValidationの注意事項

- NSMenuは自動でUpdate
- NSToolbarItemも自動でUpdate
- NSControlは手動で
  validateUserInterfaceItemを呼出す

ControlEnablerは？

特徴

- NSUserInterfaceValidationの方式を拡張
- MenuやToolBarだけでなくControlも自動で更新
- 設定でなくメソッド名の規約を定める

ControlEnabler

```
@interface CCAppDelegate : NSObject
-(IBAction)convert:(id)sender;
@end

@implementation CCAppDelegate
- (BOOL) canConvert:(id)sender
{
    return YES or NO;
}

- (NSString*) titleOfConvert:(id<NSValidatedUserInterfaceItem>)anItem
{
    if( [self canConvert:anItem] )
        return @"Convert";
    else
        return @"Don't convert";
}

@end
```

メソッド名を
“can<action名>:”とする

メソッド名を
“titleOf<action名>:”とする

メソッド名と機能

Actionメソッド名：“action:”

action:を呼出すコントロールのenable/disableは

- (BOOL) canAction:(id)sender

action:を呼出すコントロールのstate(On/Off等)は

- (NSIntger) stateOfAction:(id)sender

メソッド名と機能

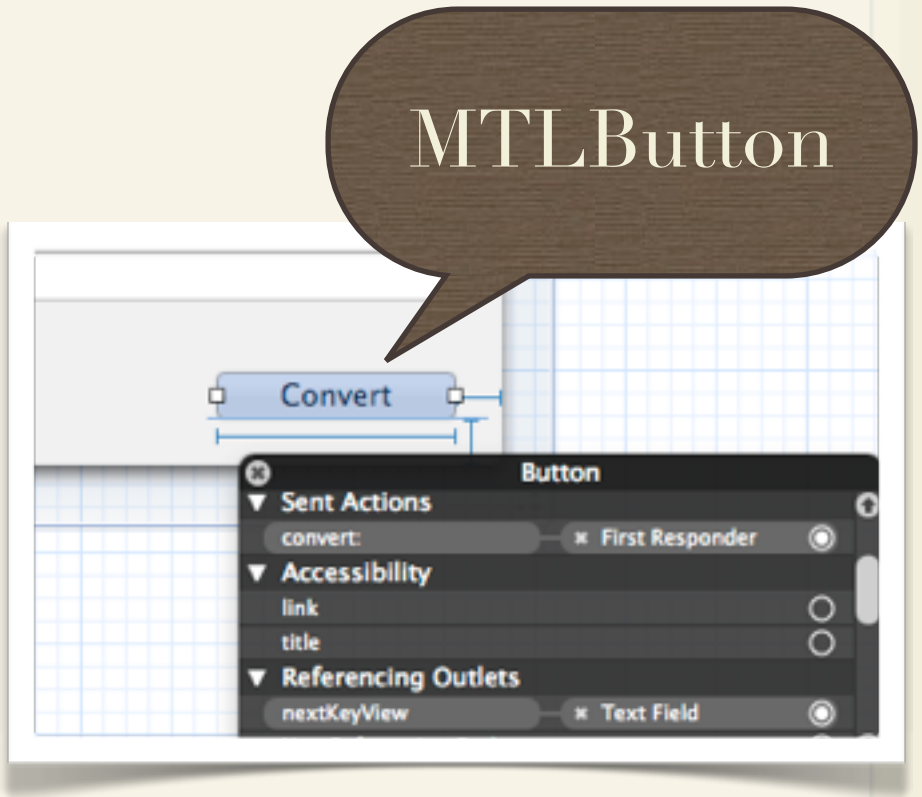
機能	メソッド名	返値の型
enable/disable	can<Action>:	BOOL
状態	stateOf<Action>:	NSCellStateValue
画像	imageOf<Action>:	NSImage
タイトル	titleOf<Action>:	NSString
物によって違う	stringValueOf<Action>:	NSString
toolTipの内容	toolTipOf<Action>:	NSString

コード例

- Buttonのenable/disable
- Toolbarのenable/disable
- Menuのenable/disable
- Menuの選択
- ラジオボタンの選択
- セグメントコントロールの選択

ButtonのEnable/disable

```
@interface CCAppDelegate : NSObject
-(IBAction)convert:(id)sender;
@end

@implementation CCAppDelegate
- (BOOL) canConvert:(id)sender
{
    return YES or NO;
}

- (BOOL)validateUserInterfaceItem:(id
<NSValidatedUserInterfaceItem>*)anItem
{
    return [self controlItemEnabler:anItem];
}
@end
```

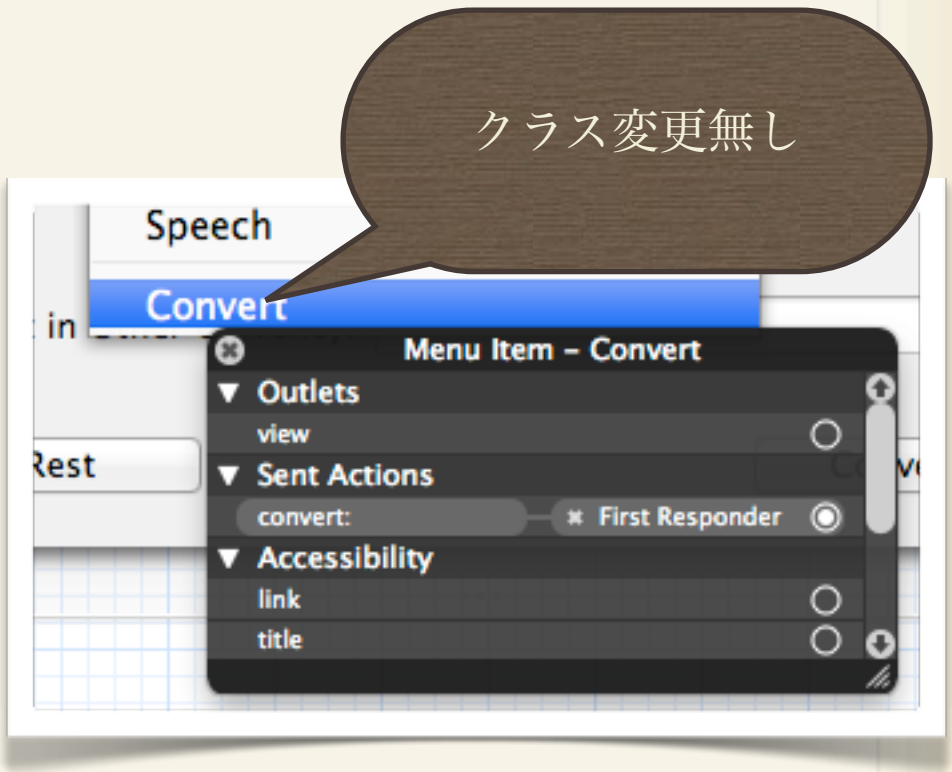


Toolbarのenable/disable

```
@interface CCAppDelegate : NSObject
-(IBAction)convert:(id)sender;
@end

@implementation CCAppDelegate
- (BOOL) canConvert:(id)sender
{
    return YES or NO;
}

- (BOOL)validateUserInterfaceItem:(id
<NSValidatedUserInterfaceItem>*)anItem
{
    return [self controlItemEnabler:anItem];
}
@end
```

Menuのenable/disable

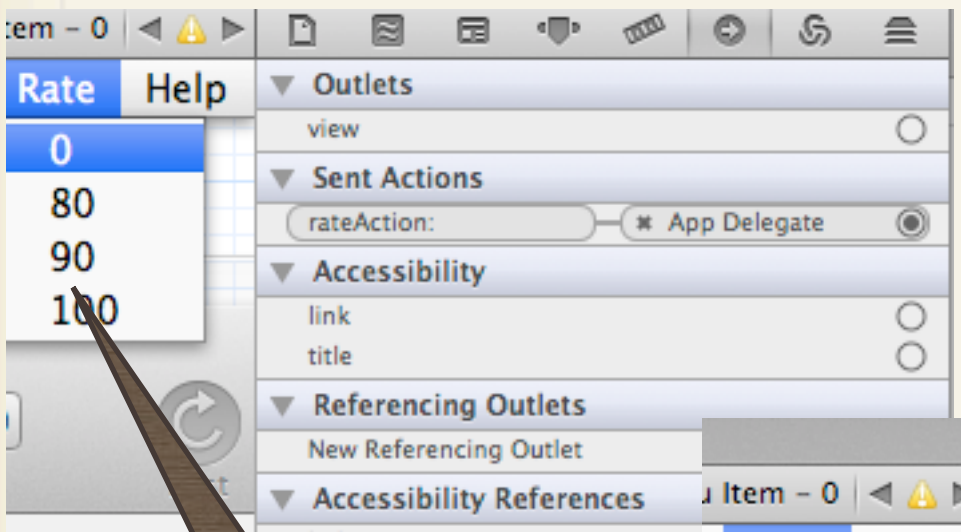
```
@interface CCAppDelegate : NSObject
-(IBAction)convert:(id)sender;
@end

@implementation CCAppDelegate
- (BOOL) canConvert:(id)sender
{
    return YES or NO;
}

- (BOOL)validateUserInterfaceItem:(id
<NSValidatedUserInterfaceItem>*)anItem
{
    return [self controlItemEnabler:anItem];
}
@end
```



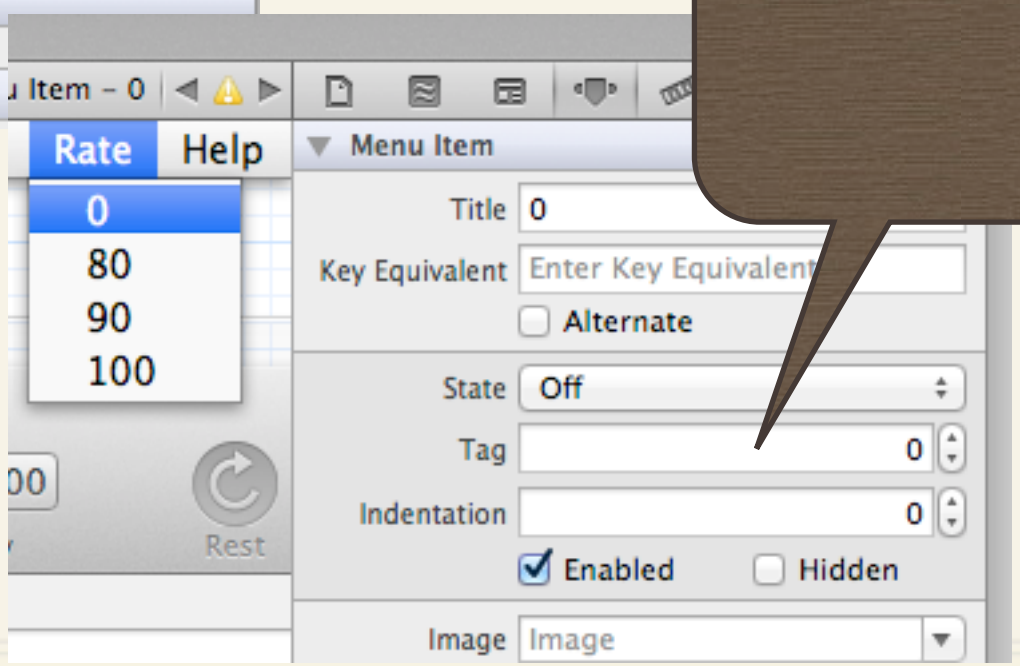
Menuの選択

```
@interface CCAppDelegate : NSObject
-(IBAction)rateAction:(id)sender;
@end

@implementation CCAppDelegate
-(IBAction)rateAction:(id)sender
{
    self.model.rate = [sender selectedTag];
}

- (NSInteger) stateOfRateAction:(id<NSValidatedUserInterfaceItem,
MTLControlItemEnablerProtocol>)anItem
{
    if( self.model.rate == [anItem selectedTag] )
      return NSOnState;
     else
      return NSOffState;
}

- (BOOL)validateUserInterfaceItem:....略

@end
```

Menuの選択



全部 rateAction:



タグは上から

0, 80, 90, 100を

割当てる

ラジオボタンの選択

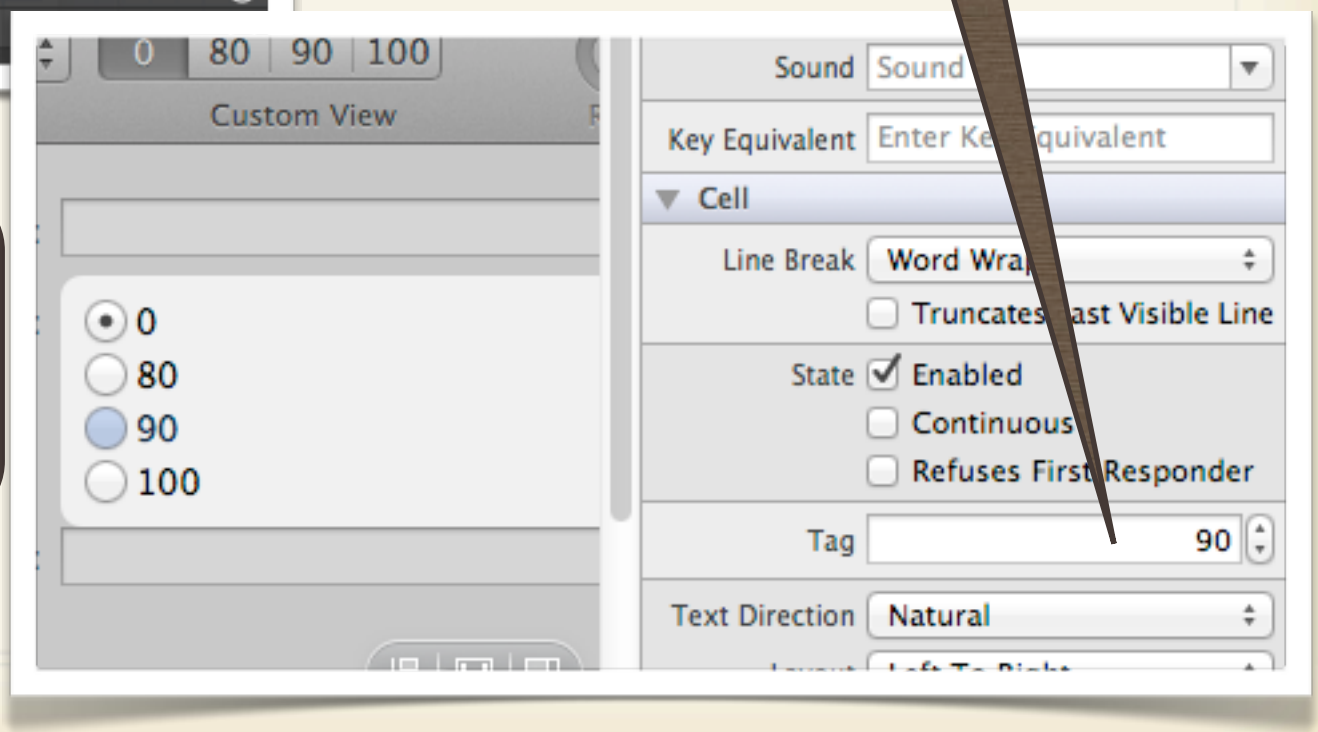
```
@interface CCAppDelegate : NSObject
-(IBAction)rateAction:(id)sender;
@end

@implementation CCAppDelegate
-(IBAction)rateAction:(id)sender
{
    self.model.rate = [sender selectedTag];
}

- (NSInteger) stateOfRateAction:(id<NSValidatedUserInterfaceItem,
MTLControlItemEnablerProtocol>)anItem
{
    if( self.model.rate == [anItem selectedTag] )
      return NSOnState;
    else
      return NSOffState;
}

- (BOOL)validateUserInterfaceItem:....略

@end
```

ラジオボタンの選択

MTLMatrixへ変更

タグは上から
0, 80, 90, 100を
割当てる

matrixは
rateAction:

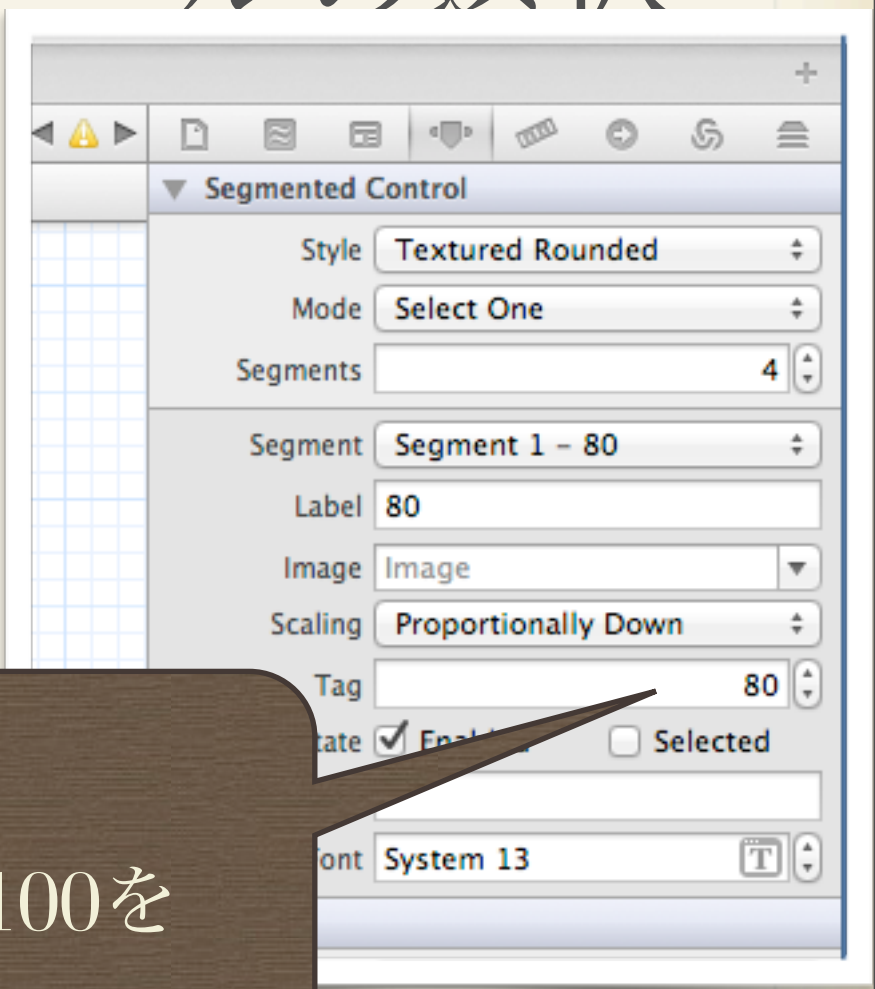


セグメントコントロールの選択

```
@interface CCAppDelegate : NSObject
-(IBAction)rateAction:(id)sender;
@end

@implementation CCAppDelegate
-(IBAction)rateAction:(id)sender
{
    self.model.rate = [sender selectedTag];
}

- (NSInteger) stateOfRateAction:(id<NSValidatedUserInterfaceItem,
MTLControlItemEnablerProtocol>)anItem
{
    if( self.model.rate == [anItem selectedTag] )
      return NSOnState;
    else
      return NSOffState;
}

- (BOOL)validateUserInterfaceItem:....略

@end
```

セグメントコントロールの選択

MTLSegmentControlへ変更



actionは

rateAction:

タグは

0, 80, 90, 100を

順に割当てる

Demo

利点と欠点

ControlEnablerの利点

- actionメソッドの近くに制御内容が書ける
  - 宣言的に表記出来る
- targetがnilでもOKなので疎結合
  - nibの分轄時に楽
- bindingよりIB上の操作が楽(かも)

ControlEnablerの欠点

- 規約に適合したメソッド名を正確に書かなければならない
- target-actionを持たないNSViewには対応出来ない
- bindingと共存出来ないかもしれない
- 成田が飽きるとライブラリ更新が止まる