The Hotel Portal

Custom error handling on iOS

Michael Ochs, Mobile Developer, HRS

Agenda

- Responsibility
- NSError / NSErrorRecoveryAttempting
- Presentation
- UIResponder
- What’s next?

Error handling before

How not to do it

How not to do it

```
NSError *error = nil;
BOOL success = [self.data writeToFile:@"my/path"
                              options:NSDataWritingAtomic
                                error:&error];
if (!success) {
    // TODO: Implement error handling
}
```

How not to do it

```
NSError *error = nil;
BOOL success = [self.data writeToFile:@"my/path"
                              options:NSDataWritingAtomic
                                error:&error];
if (!success) {
    // TODO: Implement error handling
}
```

How not to do it

- Implemented in UIViewController subclasses
- Custom in each controller
- Various error messages for the same error
- The same error message for various errors

Responsibility

Responsibility

- Globally in the app
- Usable by every potential error receiver
- Easy to use
- Little setup effort
- Error recovery

Responsibility

Consumer of the error knows:

- what action led to the error
- is a retry possible
- how to retry

Creator of the error knows:

- what went wrong
- does a retry make sense
- how to recover

Responsibility

Responsibility

Consumer	Creator
Trigger action	
	Create error
Present error	
	Recover from error
Retry action	

Responsibility

Application

Controller

Data

Trigger action

Create error

Present error

Recover from error

Retry action

NSError

NSError

- NSLocalizedDescriptionKey
- NSLocalizedFailureReasonErrorKey
- NSLocalizedRecoverySuggestionErrorKey
- NSLocalizedRecoveryOptionsErrorKey
- NSRecoveryAttempterErrorKey

NSError

NSLocalizedDescriptionKey

The corresponding value is a localized string representation of the error that, if present, will be returned by localizedDescription.

NSError

NSLocalizedFailureReasonErrorKey

The corresponding value is a localized string representation containing the reason for the failure that, if present, will be returned by localizedFailureReason.

This string provides a more detailed explanation of the error than the description.

NSError

NSLocalizedRecoverySuggestionErrorKey

The corresponding value is a string containing the localized recovery suggestion for the error.

This string is suitable for displaying as the secondary message in an alert panel.

NSError

NSLocalizedRecoveryOptionsErrorKey

The corresponding value is an array containing the localized titles of buttons appropriate for displaying in an alert panel.

The first string is the title of the right-most and default button, the second the one to the left, and so on. The recovery options should be appropriate for the recovery suggestion returned by localizedRecoverySuggestion.

NSError

NSRecoveryAttempterErrorKey

The corresponding value is an object that conforms to the NSErrorRecoveryAttempting informal protocol.

The recovery attempter must be an object that can correctly interpret an index into the array returned by localizedRecoveryOptions.

NSError

- NSLocalizedDescriptionKey
- NSLocalizedFailureReasonErrorKey
- NSLocalizedRecoverySuggestionErrorKey
- NSLocalizedRecoveryOptionsErrorKey
- NSRecoveryAttempterErrorKey

NSError

Error creation

```
- (BOOL)doSaveOperation:(NSError **)error {

    NSError *error = [NSError errorWithDomain:MyErrorDomain
                                         code:MyErrorCode
                                     userInfo:userInfo];
    *error = error;

    return NO;
}
```

NSError

Error creation

```
NSDictionary *userInfo = @{
    NSLocalizedFailureReasonErrorKey:
                @"Something went wrong",

    NSLocalizedRecoverySuggestionErrorKey:
                @"Not successful. Try again!",

    NSLocalizedRecoveryOptionsErrorKey:
                @[ @"Cancel", @"Retry" ],

    NSRecoveryAttempterErrorKey:
                recoveryAttempter
};
```

NSError

Error creation

```
NSDictionary *userInfo = @{
    NSLocalizedFailureReasonErrorKey:
                @"Something went wrong",

    NSLocalizedRecoverySuggestionErrorKey:
                @"Not successful. Try again!",

    NSLocalizedRecoveryOptionsErrorKey:
                [recoveryAttempter localizedRecoveryOptions],

    NSRecoveryAttempterErrorKey:
                recoveryAttempter
};
```

NSErrorRecoveryAttempting

NSErrorRecoveryAttempting

- Informal protocol on NSObject
- Integrated with NSError
- Handles error recovery

NSErrorRecoveryAttempting

```
- (void)attemptRecoveryFromError:(NSError *)error
                     optionIndex:(NSUInteger)recoveryOptionIndex
                        delegate:(id)delegate
              didRecoverSelector:(SEL)didRecoverSelector
                     contextInfo:(void *)contextInfo;

- (BOOL)attemptRecoveryFromError:(NSError *)error
                     optionIndex:(NSUInteger)recoveryOptionIndex;
```

NSErrorRecoveryAttempting

```
- (void)attemptRecoveryFromError:(NSError *)error
                     optionIndex:(NSUInteger)recoveryOptionIndex
                        delegate:(id)delegate
              didRecoverSelector:(SEL)didRecoverSelector
                     contextInfo:(void *)contextInfo;

- (BOOL)attemptRecoveryFromError:(NSError *)error
                     optionIndex:(NSUInteger)recoveryOptionIndex;
```

NSErrorRecoveryAttempting

```
- (void)attemptRecoveryFromError:(NSError *)error
                     optionIndex:(NSUInteger)recoveryOptionIndex
                        delegate:(id)delegate
              didRecoverSelector:(SEL)didRecoverSelector
                     contextInfo:(void *)contextInfo;

- (BOOL)attemptRecoveryFromError:(NSError *)error
                     optionIndex:(NSUInteger)recoveryOptionIndex;
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
- (void)addRecoveryOptionWithTitle:(NSString *)title
                    recoveryAttempt:(BOOL(^)())recoveryBlock;

- (NSArray *)localizedRecoveryOptions;
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
- (void)addRecoveryOptionWithTitle:(NSString *)title
                   recoveryAttempt:(BOOL(^)())recoveryBlock;

- (NSArray *)localizedRecoveryOptions;
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
- (void)addRecoveryOptionWithTitle:(NSString *)title
                    recoveryAttempt:(BOOL(^)())recoveryBlock;

- (NSArray *)localizedRecoveryOptions;
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
HRSErrorRecoveryAttempter *recoveryAttempter =
                                    [HRSErrorRecoveryAttempter new];

[recoveryAttempter addCancelRecoveryOption];

[recoveryAttempter addRecoveryOptionWithTitle:@"Retry"
                                recoveryAttempt:^{
                                    return YES;
                                }];
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
HRSErrorRecoveryAttempter *recoveryAttempter =
                                    [HRSErrorRecoveryAttempter new];

[recoveryAttempter addCancelRecoveryOption];

[recoveryAttempter addRecoveryOptionWithTitle:@"Retry"
                                recoveryAttempt:^{
                                    return YES;
                                }];
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
HRSErrorRecoveryAttempter *recoveryAttempter =
                                    [HRSErrorRecoveryAttempter new];

[recoveryAttempter addCancelRecoveryOption];

[recoveryAttempter addRecoveryOptionWithTitle:@"Retry"
                                recoveryAttempt:^{
                                    return YES;
                                }];
```

NSErrorRecoveryAttempting

HRSErrorRecoveryAttempter

```
HRSErrorRecoveryAttempter *recoveryAttempter =
                                    [HRSErrorRecoveryAttempter new];

[recoveryAttempter addCancelRecoveryOption];

[recoveryAttempter addRecoveryOptionWithTitle:@"Retry"
                                recoveryAttempt:^{
                                    return YES;
                                }];
```

Presentation

Presentation

OS X

```
@interface NSResponder (NSErrorPresentation)

- (BOOL)presentError:(NSError *)anError;

- (void)presentError:(NSError *)error
      modalForWindow:(NSWindow *)aWindow
            delegate:(id)delegate
  didPresentSelector:(SEL)didPresentSelector
         contextInfo:(void *)contextInfo;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

OS X

```
@interface NSResponder (NSErrorPresentation)

- (BOOL)presentError:(NSError *)anError;

- (void)presentError:(NSError *)error
      modalForWindow:(NSWindow *)aWindow
            delegate:(id)delegate
  didPresentSelector:(SEL)didPresentSelector
         contextInfo:(void *)contextInfo;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

OS X

```
@interface NSResponder (NSErrorPresentation)

- (BOOL)presentError:(NSError *)anError;

- (void)presentError:(NSError *)error
      modalForWindow:(NSWindow *)aWindow
            delegate:(id)delegate
  didPresentSelector:(SEL)didPresentSelector
         contextInfo:(void *)contextInfo;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

OS X

```
@interface NSResponder (NSErrorPresentation)

- (BOOL)presentError:(NSError *)anError;

- (void)presentError:(NSError *)error
      modalForWindow:(NSWindow *)aWindow
            delegate:(id)delegate
  didPresentSelector:(SEL)didPresentSelector
         contextInfo:(void *)contextInfo;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

iOS

```
@interface UIResponder (HRSCustomErrorPresentation)

- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

iOS

```
@interface UIResponder (HRSCustomErrorPresentation)

- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

iOS

```
@interface UIResponder (HRSCustomErrorPresentation)

- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler;

- (NSError *)willPresentError:(NSError *)anError;

@end
```

Presentation

iOS

```
- (IBAction)save:(id)sender {
    NSError *error = nil;
    BOOL success = [self.data writeToFile:@"my/path"
                                  options:NSDataWritingAtomic
                                    error:&error];
    if (!success) {
        [self presentError:error
         completionHandler:^(BOOL didRecover) {
                             if (didRecover) {
                                 [self save:sender];
                             }
                         }];
    }
}
```

Presentation

iOS

```
- (IBAction)save:(id)sender {
    NSError *error = nil;
    BOOL success = [self.data writeToFile:@"my/path"
                                  options:NSDataWritingAtomic
                                    error:&error];
    if (!success) {
        [self presentError:error
         completionHandler:^(BOOL didRecover) {
                             if (didRecover) {
                                 [self save:sender];
                             }
                         }];
    }
}
```

Presentation

iOS

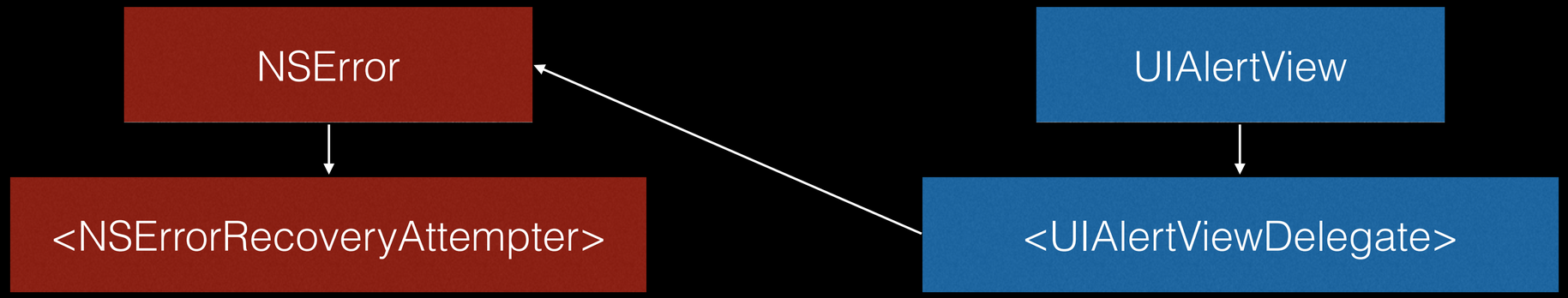
```
- (IBAction)save:(id)sender {
    NSError *error = nil;
    BOOL success = [self.data writeToFile:@"my/path"
                                  options:NSDataWritingAtomic
                                    error:&error];
    if (!success) {
        [self presentError:error
         completionHandler:^(BOOL didRecover) {
                             if (didRecover) {
                                 [self save:sender];
                             }
                         }];
    }
}
```

Architecture

Architecture

`- doSomething:`

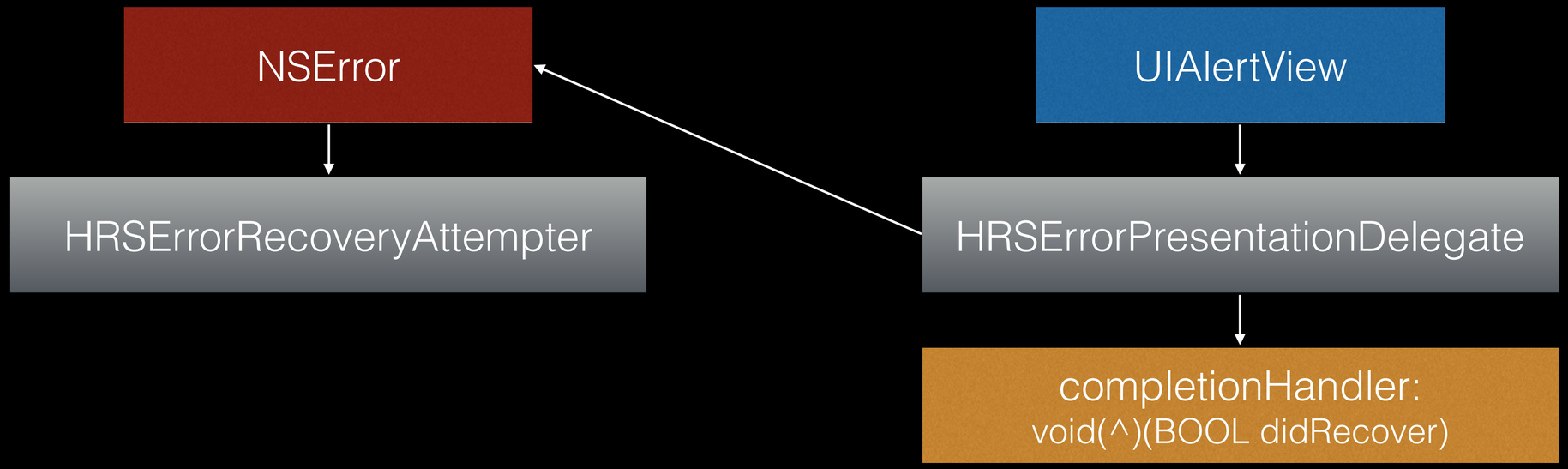
`- presentError:completionHandler:`



Architecture

`- doSomething:`

`- presentError:completionHandler:`



The path of an error

The path of an error

AppDelegate

UIViewController

Operation

```
-save:
```

```
- doSomething:
```

```
NSError
```

```
-presentError:completionHandler:
```

```
UIAlertView
```

```
-attemptRecoveryFromError:
        optionIndex:
```

```
void (^completionHandler)(BOOL didRecover)
```

UIResponder

UIResponder

`- presentError:completionHandler:`

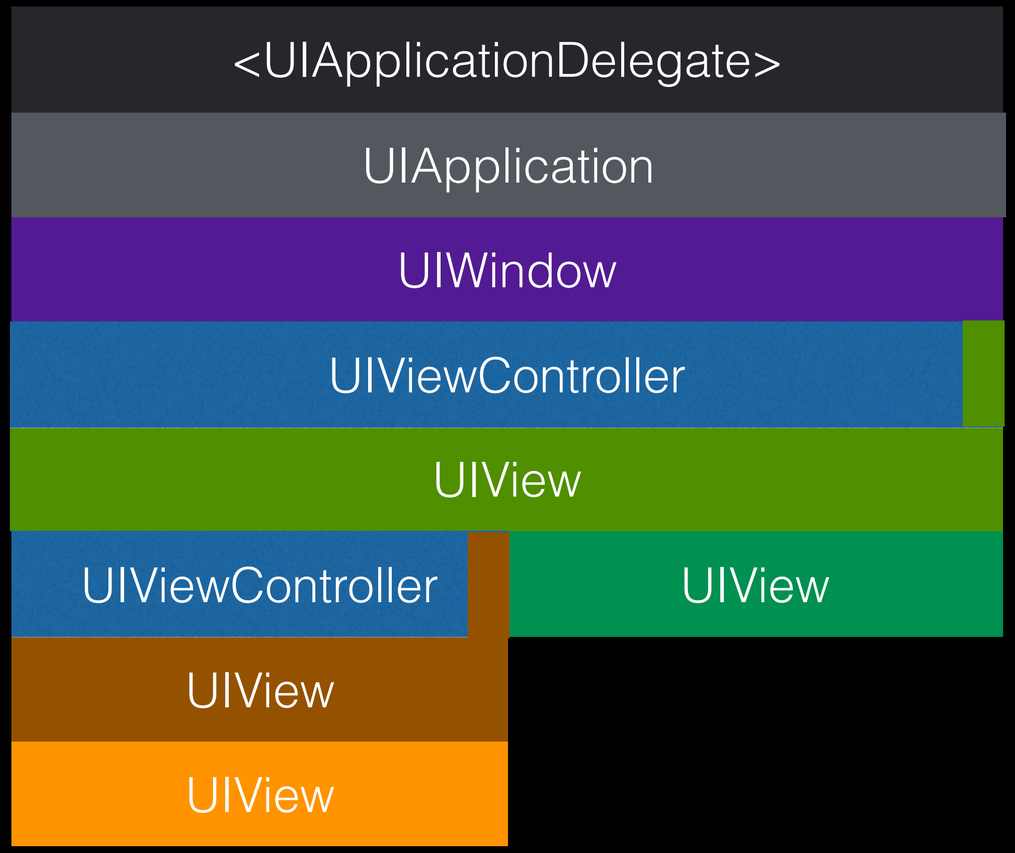
UIResponder

Subclasses

- UIView
- UIViewController
- UIApplication
- <UIApplicationDelegate>
- SKNode

UIResponder

```
- (UIResponder *)nextResponder;
```



UIResponder

```
@interface DataSource : UIResponder <UITableViewDataSource>
@property (nonatomic, weak) id<DataSourceDelegate> delegate;
@end

@implementation DataSource

- (UIResponder *)nextResponder {
    if ([self.delegate isKindOfClass:[UIResponder class]]) {
        return self.delegate;
    }
    return nil;
}

@end
```

UIResponder

```
@interface DataSource : UIResponder <UITableViewDataSource>
@property (nonatomic, weak) id<DataSourceDelegate> delegate;
@end

@implementation DataSource

- (UIResponder *)nextResponder {
    if ([self.delegate isKindOfClass:[UIResponder class]]) {
        return self.delegate;
    }
    return nil;
}

@end
```

UIResponder

```
@interface DataSource : UIResponder <UITableViewDataSource>
@property (nonatomic, weak) id<DataSourceDelegate> delegate;
@end

@implementation DataSource

- (UIResponder *)nextResponder {
    if ([self.delegate isKindOfClass:[UIResponder class]]) {
        return self.delegate;
    }
    return nil;
}

@end
```

UIResponder

UIResponder (HRSCustomErrorPresentation)

UIResponder

UIResponder (HRSCustomErrorPresentation)

```
- (NSError *)willPresentError:(NSError *)anError {
    return anError;
}
```

UIResponder

UIResponder (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }

    UIResponder *nextResponder = ([self nextResponder] ?:
                                  [UIApplication sharedApplication]);
    [nextResponder presentError:anError
              completionHandler:completionHandler];
}
```

UIResponder

UIResponder (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }

    UIResponder *nextResponder = ([self nextResponder] ?:
                                  [UIApplication sharedApplication]);
    [nextResponder presentError:anError
              completionHandler:completionHandler];
}
```

UIResponder

UIResponder (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }

    UIResponder *nextResponder = ([self nextResponder] ?:
                                  [UIApplication sharedApplication]);
    [nextResponder presentError:anError
              completionHandler:completionHandler];
}
```

UIResponder

UIResponder (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL didRecover))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }

    UIResponder *nextResponder = ([self nextResponder] ?:
                                  [UIApplication sharedApplication]);
    [nextResponder presentError:anError
              completionHandler:completionHandler];
}
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
- (NSError *)willPresentError:(NSError *)anError {
    if (anError.recoveryAttempter == nil) {
        NSDictionary *userInfo = @{ … };
        anError = [NSError errorWithDomain:anError.domain
                                      code:anError.code
                                  userInfo:userInfo];
    }
    return anError;
}
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
- (NSError *)willPresentError:(NSError *)anError {
    if (anError.recoveryAttempter == nil) {
        NSDictionary *userInfo = @{ … };
        anError = [NSError errorWithDomain:anError.domain
                                      code:anError.code
                                  userInfo:userInfo];
    }
    return anError;
}
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
- (NSError *)willPresentError:(NSError *)anError {
    if (anError.recoveryAttempter == nil) {
        NSDictionary *userInfo = @{ … };
        anError = [NSError errorWithDomain:anError.domain
                                      code:anError.code
                                  userInfo:userInfo];
    }
    return anError;
}
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
- (void)presentError:(NSError *)anError
   completionHandler:(void (^)(BOOL))completionHandler {

    anError = [self willPresentError:anError];
    if (anError == nil) {
        return;
    }
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
HRSErrorPresentationDelegate *delegate = …;

UIAlertView *alertView = [[UIAlertView alloc]
          initWithTitle:[anError localizedFailureReason]
                message:[anError localizedRecoverySuggestion]
               delegate:delegate
      cancelButtonTitle:nil
      otherButtonTitles:nil];

for (NSString *title in [anError localizedRecoveryOptions]) {
    [alertView addButtonWithTitle:title];
}

[alertView show];
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
HRSErrorPresentationDelegate *delegate = …;

UIAlertView *alertView = [[UIAlertView alloc]
          initWithTitle:[anError localizedFailureReason]
                message:[anError localizedRecoverySuggestion]
               delegate:delegate
      cancelButtonTitle:nil
      otherButtonTitles:nil];

for (NSString *title in [anError localizedRecoveryOptions]) {
    [alertView addButtonWithTitle:title];
}

[alertView show];
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
HRSErrorPresentationDelegate *delegate = …;

UIAlertView *alertView = [[UIAlertView alloc]
          initWithTitle:[anError localizedFailureReason]
                message:[anError localizedRecoverySuggestion]
               delegate:delegate
      cancelButtonTitle:nil
      otherButtonTitles:nil];

for (NSString *title in [anError localizedRecoveryOptions]) {
    [alertView addButtonWithTitle:title];
}

[alertView show];
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
HRSErrorPresentationDelegate *delegate = …;

UIAlertView *alertView = [[UIAlertView alloc]
          initWithTitle:[anError localizedFailureReason]
                message:[anError localizedRecoverySuggestion]
               delegate:delegate
      cancelButtonTitle:nil
      otherButtonTitles:nil];

for (NSString *title in [anError localizedRecoveryOptions]) {
    [alertView addButtonWithTitle:title];
}

[alertView show];
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
HRSErrorPresentationDelegate *delegate = …;

UIAlertView *alertView = [[UIAlertView alloc]
          initWithTitle:[anError localizedFailureReason]
                message:[anError localizedRecoverySuggestion]
               delegate:delegate
      cancelButtonTitle:nil
      otherButtonTitles:nil];

for (NSString *title in [anError localizedRecoveryOptions]) {
    [alertView addButtonWithTitle:title];
}

[alertView show];
```

UIResponder

AppDelegate (HRSCustomErrorPresentation)

```
    objc_setAssociatedObject(alertView,
                             DelegateAssociation,
                             delegate,
                             OBJC_ASSOCIATION_RETAIN);

}
```

What is next?

What’s next?

- Handle common errors
- Provide custom error presentation UI
- Add specific error recovery attempter
- This is all loosely coupled

What’s next?

- Available open source
- Currently considered beta
- Internal interaction might still change
- Feedback welcome

Related topics

- Realmac's "Cocoa error handling and recovery" http://realmacsoftware.com/blog/cocoa-error-handling-and-recovery

Feedback / Questions

@_mochs

michael.ochs@hrs.com

ios-coding.com

Thank you