Data

IAP 2010 ❄

iphonedev.csail.mit.edu

edward benson / eob@csail.mit.edu

Today

- Property Lists
- User Defaults
- Settings Panels
- CoreData

Property Lists

Today

Add persistence.

1. Using Property Lists in the filesystem

2.0 Using NSUserDefaults

2.5 Creating a System Preference Panel

CD

3. Using the SQLite database

Property Lists

Property Lists are a simple way to serialize basic data to disk.

`NSData`	`NSDate`
`NSString`	`NSArray`
`NSNumber`	`NSDictionary`

PLists are to Apple what YAML is to the Ruby/Rails community

Property Lists

For example

`info.plist`

contains your application settings.

Key	Value
▼ Information Property List	(11 items)
Localization native development re	en
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Bundle identifier	com.yourcompany.${PRODUCT_NAME:identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.2
LSRequiresIPhoneOS	☑
Main nib file base name	MainWindow

Property Lists

These are a great way to store small collections of data.

But first we need to know what the iPhone Application directory structure looks like.

The On-phone Filesystem Structure

/<Application Home>


/Documents ***application-specific data files***

/Library


/Preferences ***stores system settings***

/Caches ***cache data between runs***

/tmp ***thrown out after each run***


Backed up during sync with iTunes

Property Lists

So if we want RW access to our PLists,

$HOME/Documents

It we just want RO access:

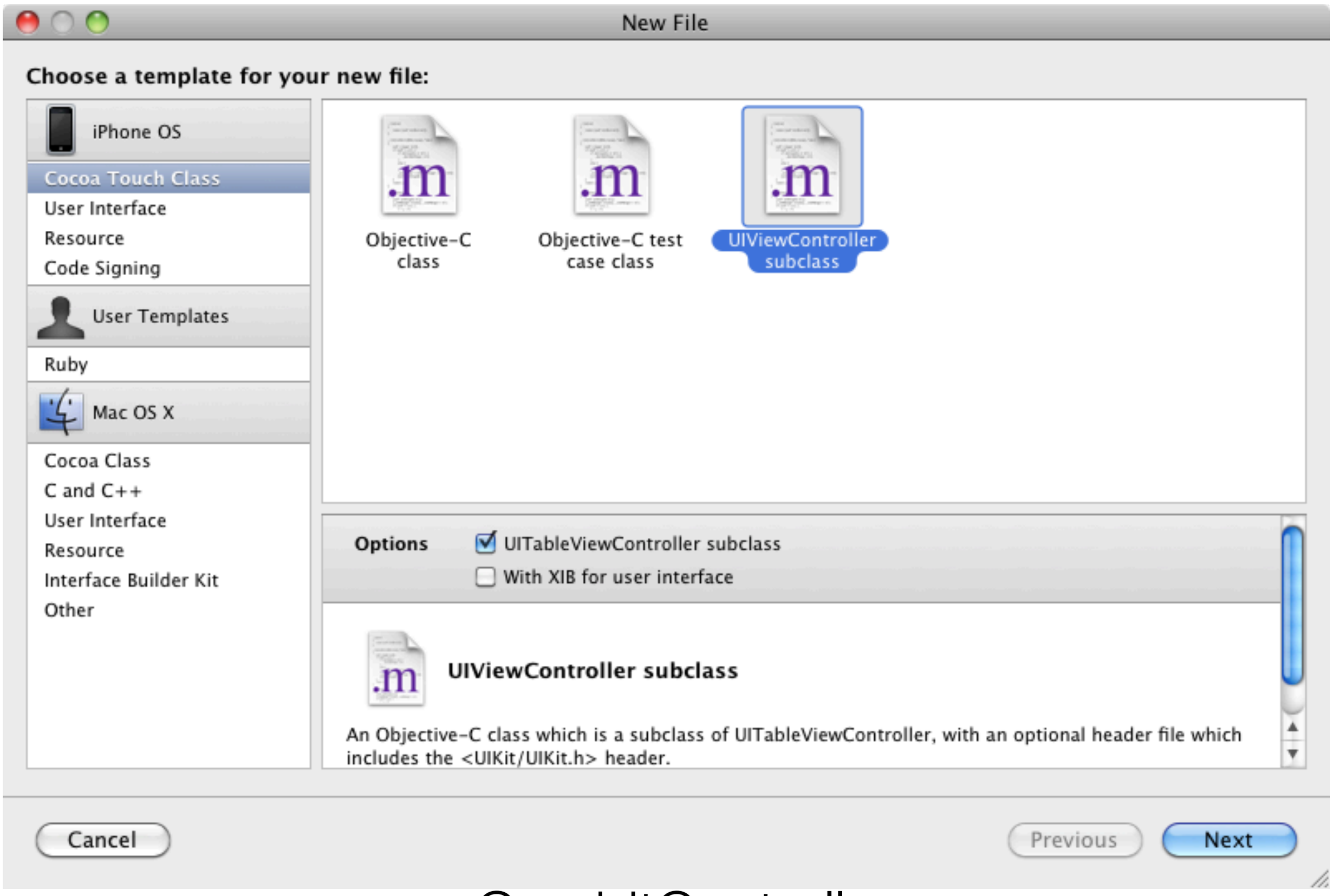
Anywhere is cool

We'll just do RO in this example, but you'll see how to use $HOME/Documents later today..

Gambit.plist

Key	Type	Value
▼ Root	Array	(2 items)
▼ Item 0	Dictionary	(2 items)
Name	String	The Avalance
Sequence	String	ROCK ROCK ROCK
▼ Item 1	Dictionary	(2 items)
Name	String	The Diplomat
Sequence	String	PAPER PAPER ROCK

We'll show these gambits in a table



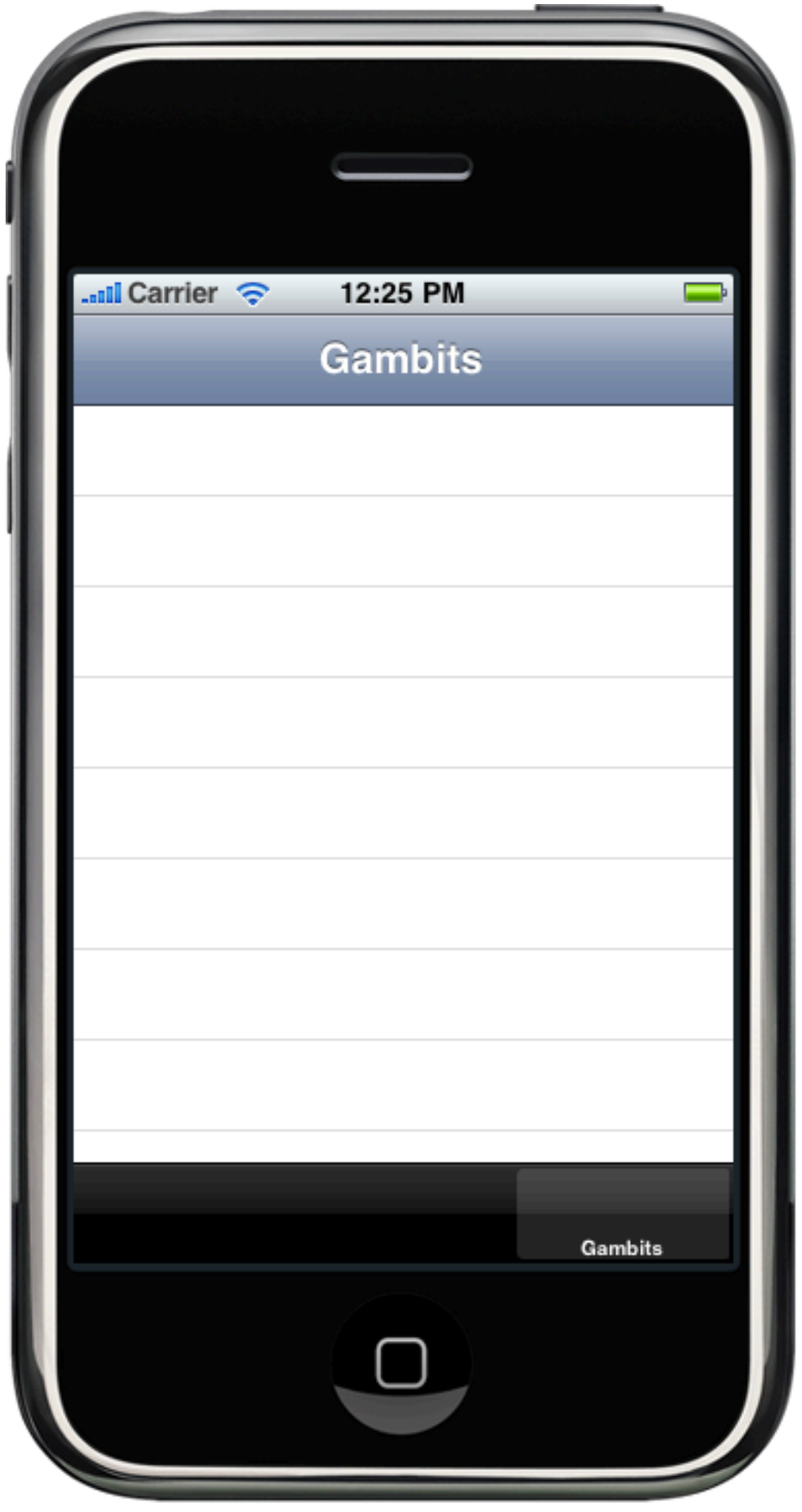
GambitController

We'll add this to our tabs at the bottom

This is the same as when we added the history controller

App Delegate

```
GameHistoryController *gambit = [[[GambitController alloc] init]
                                                    autorelease];

UINavigationController *gambitWrapper = [[[UINavigationController alloc]
                        initWithRootViewController:gambit] autorelease];



tabs.viewControllers = [NSArray arrayWithObjects:gameViewController,
                                                 historyWrapper,
                                                 gambitWrapper,
                                                 nil];
```


Carrier
12:25 PM
Gambits
Gambits

Create our gambits instance variable

Interface

```
@interface GambitController : UITableViewController {
  NSArray *gambits;
}

@end
```

Implementation

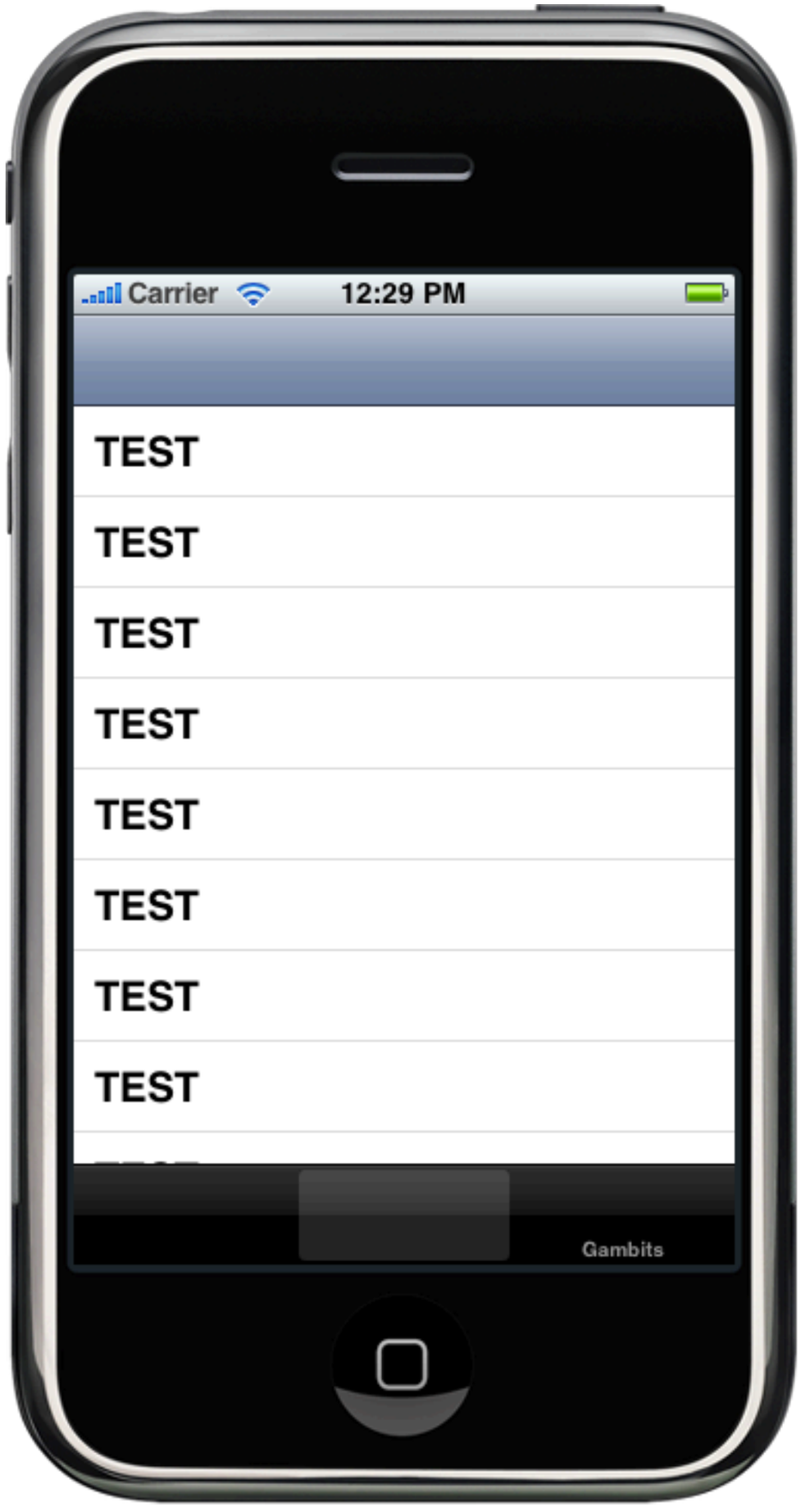
```
- (void)dealloc {
    [gambits release];
     [super dealloc];
}
```

Initialize our Gambits by reading in the PList

```
-(id)init {
    if (self = [super init]) {
        self.title = @"Gambits";
        NSString *bundlePath = [[NSBundle mainBundle] bundlePath];
        NSString *dataPath = [bundlePath stringByAppendingPathComponent:@"Gambits.plist"];
        gambits = [[[NSArray alloc] initWithContentsOfFile:dataPath] retain];
    }
    return self;
}
```

Seriously.. this is it -- a one liner

Reading in PLists is very easy.

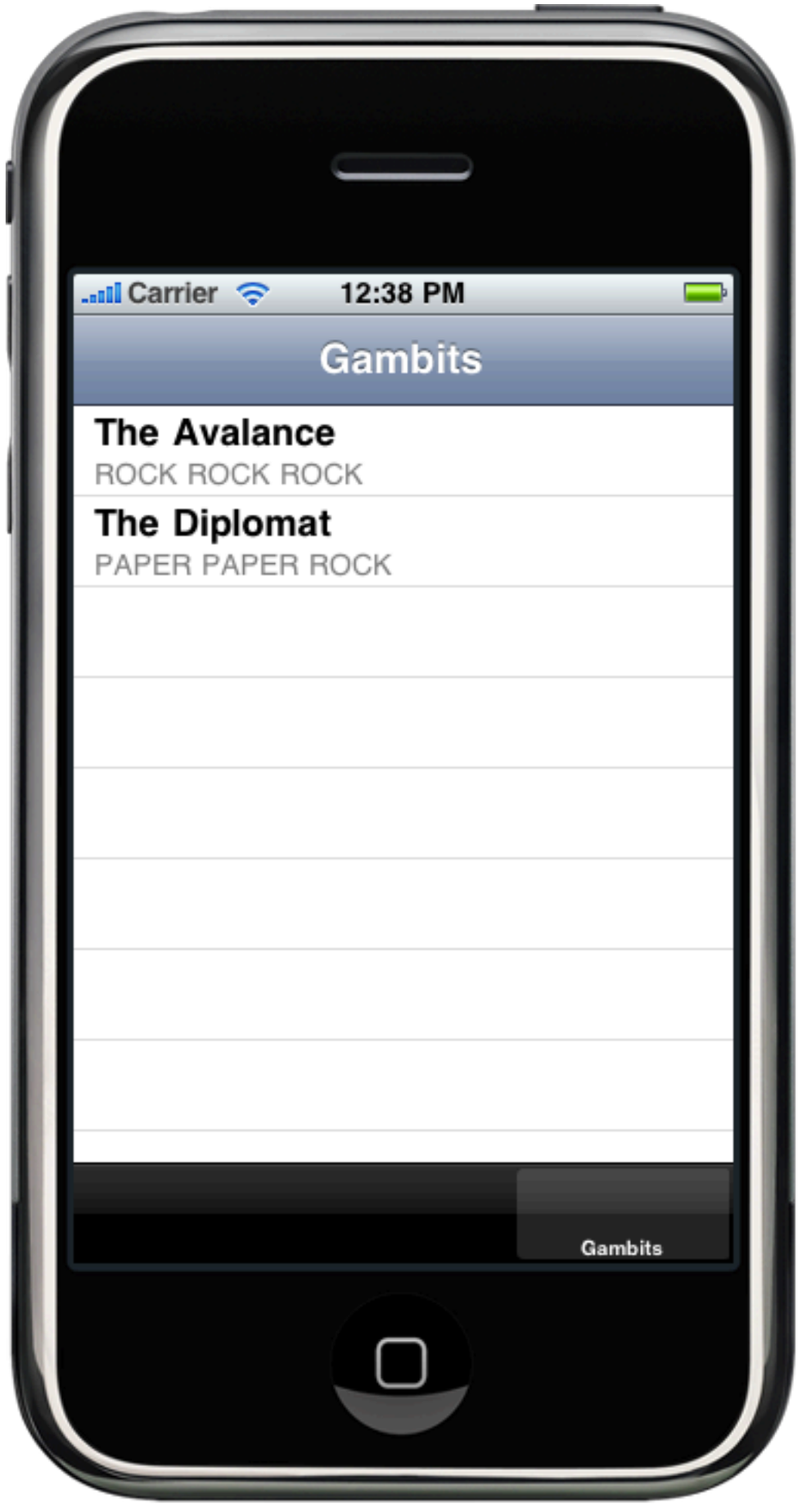

Carrier
12:29 PM
TEST
TEST
TEST
TEST
TEST
TEST
TEST
Gambits

Fill it in with real data

```
// Customize the number of rows in the table view.
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return [gambits count];
}
```

cellForRowAtIndexPath

```
// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath *)indexPath {

    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:CellIdentifier] autorelease];
    }

    // Set up the cell...
  NSDictionary *gambit = [gambits objectAtIndex:indexPath.row];
  cell.textLabel.text = [gambit objectForKey:@"Name"];
  cell.detailTextLabel.text = [gambit objectForKey:@"Sequence"];

  return cell;
}
```


Carrier
12:38 PM
Gambits
The Avalance
ROCK ROCK ROCK
The Diplomat
PAPER PAPER ROCK
Gambits

User Defaults

User Defaults

Apple provides a way to persist a “User Defaults” property list *for you*.

NSUserDefaults

```
+ standardUserDefaults
+ resetStandardUserDefaults
```

It acts similarly to a Dictionary object

```
NSUserDefaults *userDefaults = [NSUserDefaults standardUserDefaults];
String *nickName = [userDefaults stringForKey:@"nickName"];
```

NSUserDefaults

Anything you set on this object is automatically synced to a database.

To force a sync with the database, just:

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

In our App Delegate

```
-(void)spy {
}
```

In our App Delegate

```
-(void)spy {
  NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
}
```

In our App Delegate

```
-(void)spy {
  NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
  NSDate *lastRun = [defaults objectForKey:@"lastRun"];
}
```

In our App Delegate

```
-(void)spy {
  NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
  NSDate *lastRun = [defaults objectForKey:@"lastRun"];
  NSLog(@"The last run of this application was: %@", lastRun);

}
```

In our App Delegate

```
-(void)spy {
  NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
  NSDate *lastRun = [defaults objectForKey:@"lastRun"];
  NSLog(@"The last run of this application was: %@", lastRun);

  [defaults setObject:[NSDate date] forKey:@"lastRun"];
}
```

In our App Delegate

```
-(void)spy {
  NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
  NSDate *lastRun = [defaults objectForKey:@"lastRun"];
  NSLog(@"The last run of this application was: %@", lastRun);

  [defaults setObject:[NSDate date] forKey:@"lastRun"];
  [defaults synchronize];
  // Make point about lazy saving
}
```

applicationDidFinishLaunching

```
[self spy];
```

Settings

NSUserDefaults

The data gets stashed in the NSUserDefaults object

Using **system settings** to store data

Apple user interface guidelines: put every application setting in the system-wide settings panel.

Your choice what to do, but it seems like at least the important stuff belongs there (login info, etc)

In general, adherence to interface guidelines make your application easier for users to learn to use: the fewer new/unexpected metaphors, the better.

Creating defaults for the first time

in app delegate

```
- (void)checkForDefaults {
  if ( // some USER DEFAULT you expect doesn't exist ) {
     // Respond somehow
  }
}
```

What should the response be?

- Nag your users to fill in their information
- Fill in the information yourself with the defaults

To Nag, or to Launch?

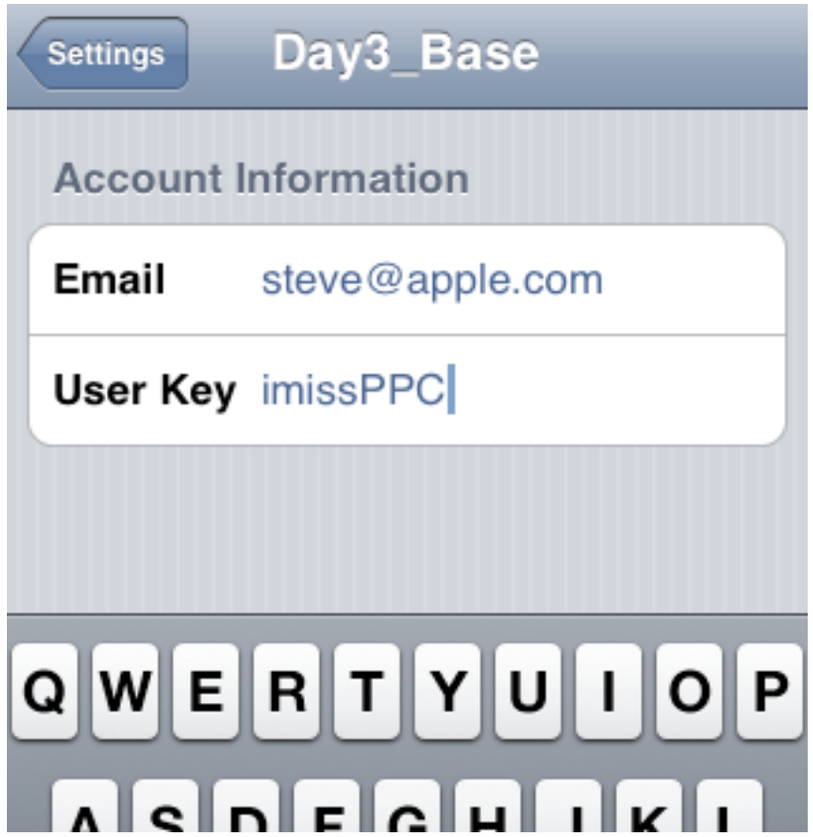
```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    NSString *emailAddress = [[NSUserDefaults standardUserDefaults] stringForKey:kEmailKey];
    NSString *password = [[NSUserDefaults standardUserDefaults] stringForKey:kPasswordKey];

    if ((emailAddress == nil) && (password == nil)) {
        [self loadNagScreen:application];
    }
    else {
        [self continueLoadingApp:application];
    }

    [window makeKeyAndVisible];
}
```

Using **system settings** to store data

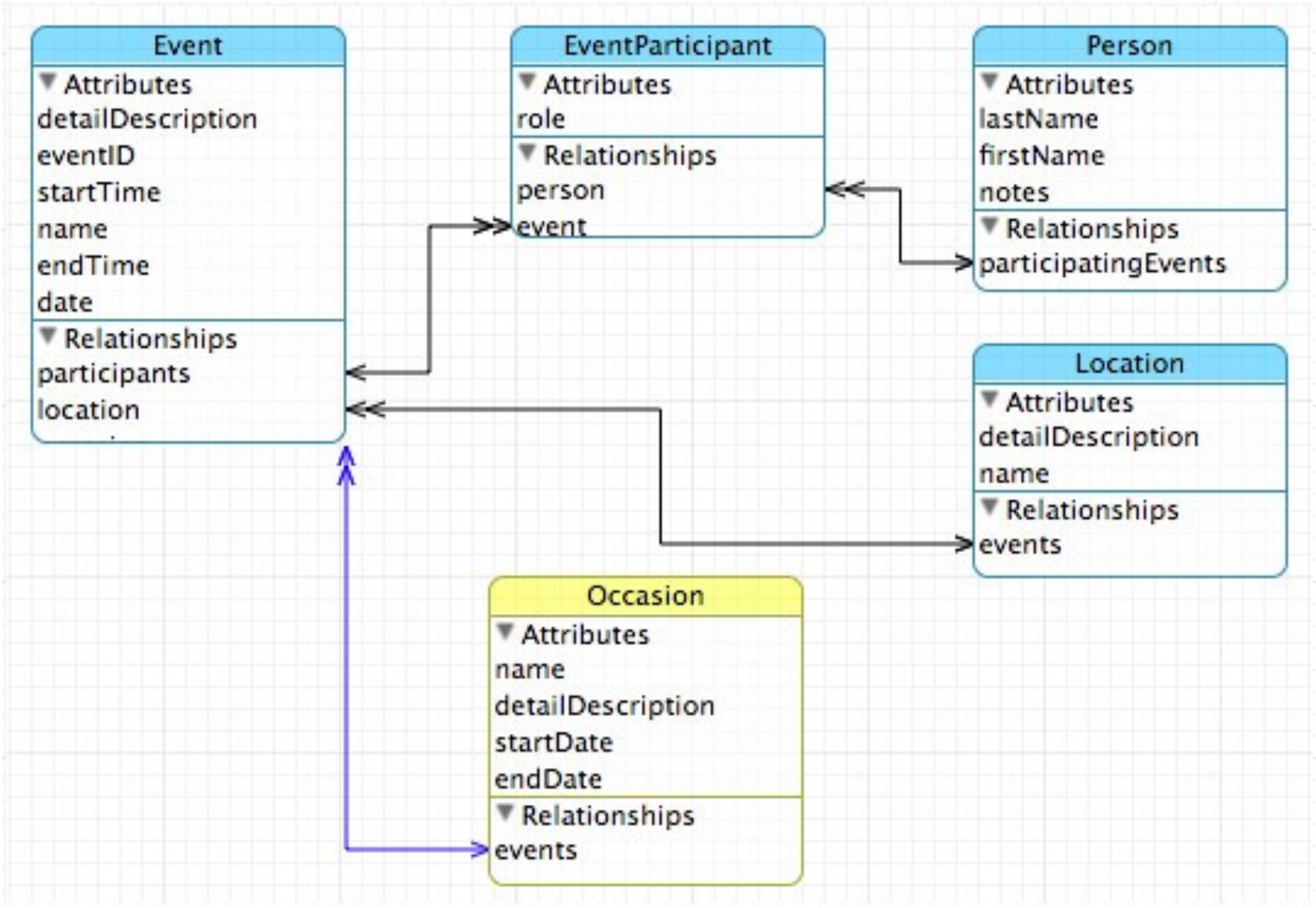
Key	Type	Value
▼ Root	Dictionary	(2 items)
Title	String	GroupJot
▼ PreferenceSpecifiers	Array	(3 items)
▼ Item 1	Dictionary	(2 items)
Title	String	Account Information
Type	String	PSGroupSpecifier
▼ Item 2	Dictionary	(4 items)
DefaultValue	String	me@gmail.com
Key	String	emailKey
Title	String	Email
Type	String	PSTextFieldSpecifier
▼ Item 3	Dictionary	(4 items)
DefaultValue	String	a4fb9
Key	String	passwordKey
Title	String	User Key
Type	String	PSTextFieldSpecifier

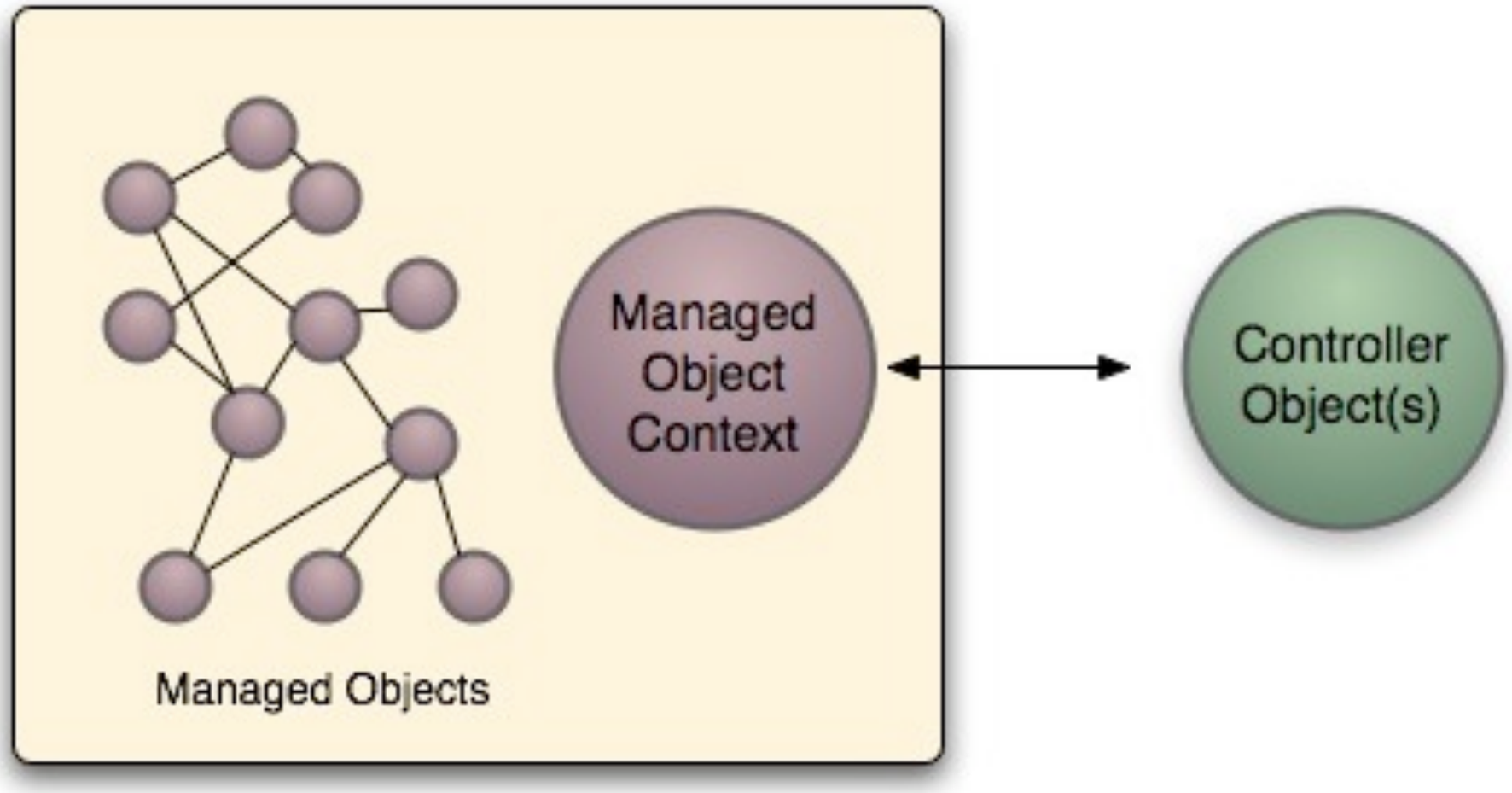
And Then...


Settings
General
Safari
Photos
Day3_Base

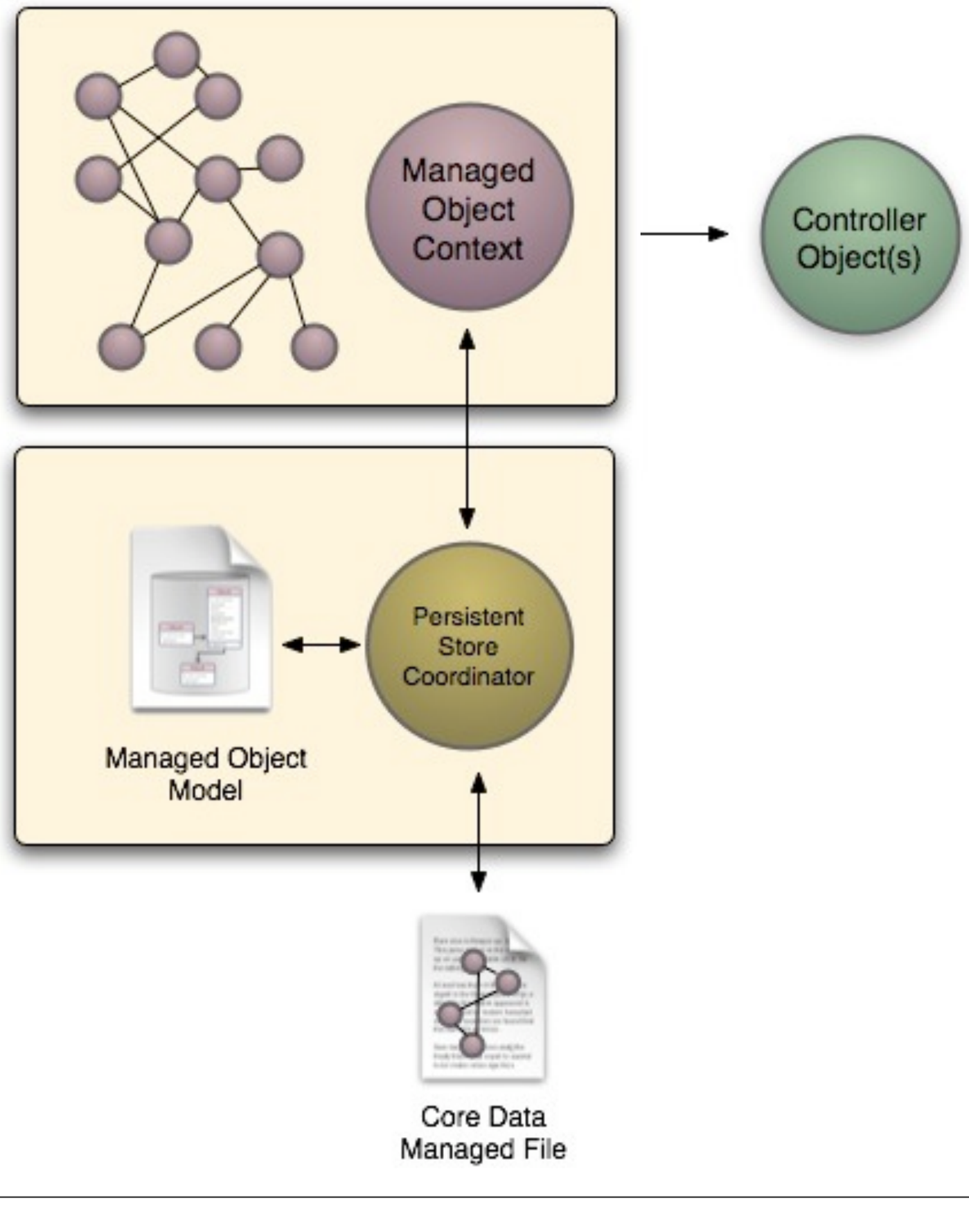

Settings
Day3_Base
Account Information
Email
steve@apple.com
User Key
imissPPC
Q W E R T Y U I O P

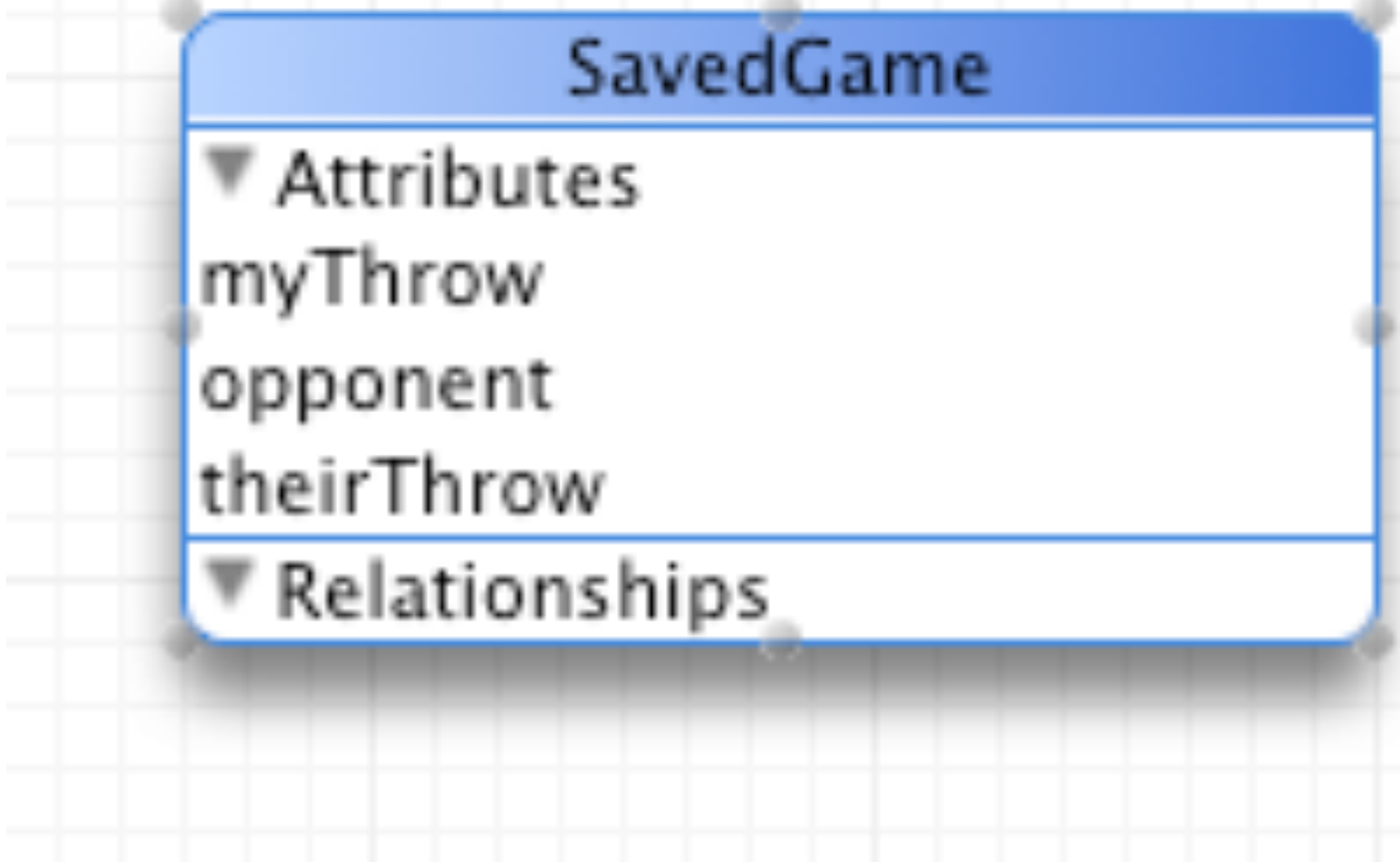
Core Data


Event
Attributes
detailDescription
eventID
startTime
name
endTime
date
Relationships
participants
location
EventParticipant
Attributes
role
Relationships
person
event
Person
Attributes
lastName
firstName
notes
Relationships
participatingEvents
Location
Attributes
detailDescription
name
Relationships
events
Occasion
Attributes
name
detailDescription
startDate
endDate
Relationships
events




```
[[NSApp delegate] managedObjectContext];
```


Managed
Object
Context
Controller
Object(s)
Persistent
Store
Coordinator
Managed Object
Model
Core Data
Managed File


SavedGame
Attributes
myThrow
opponent
theirThrow
Relationships

RPSGameViewController.m

```
-(void)completeGame {
    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }
}
```

RPSGameViewController.m

```
-(void)completeGame {

    // CREATE A NEW SavedGame AND SAVE IT


    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }
}
```

RPSGameViewController.m

```
-(void)completeGame {
    NSManagedObjectContext *context =
                [[[UIApplication sharedApplication] delegate] managedObjectContext];

    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }

}
```

RPSGameViewController.m

```
-(void)completeGame {
    NSManagedObjectContext *context =
                [[[UIApplication sharedApplication] delegate] managedObjectContext];
    NSManagedObject *savedGame =
                [NSEntityDescription insertNewObjectForEntityForName:@"SavedGame"
                                              inManagedObjectContext:context];

    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }

}
```

RPSGameViewController.m

```
-(void)completeGame {
    NSManagedObjectContext *context =
                [[[UIApplication sharedApplication] delegate] managedObjectContext];
    NSManagedObject *savedGame =
                [NSEntityDescription insertNewObjectForEntityForName:@"SavedGame"
                                              inManagedObjectContext:context];
    [savedGame setValue:@"Player 2" forKey:@"opponent"];

    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }

}
```

RPSGameViewController.m

```
-(void)completeGame {
   NSManagedObjectContext *context =
              [[[UIApplication sharedApplication] delegate] managedObjectContext];
   NSManagedObject *savedGame =
              [NSEntityDescription insertNewObjectForEntityForName:@"SavedGame"
                                            inManagedObjectContext:context];
    [savedGame setValue:@"Player 2" forKey:@"opponent"];
    [savedGame setValue:selfThrow forKey:@"myThrow"];

    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }

}
```

RPSGameViewController.m

```
-(void)completeGame {
   NSManagedObjectContext *context =
              [[[UIApplication sharedApplication] delegate] managedObjectContext];
   NSManagedObject *savedGame =
              [NSEntityDescription insertNewObjectForEntityForName:@"SavedGame"
                                            inManagedObjectContext:context];
    [savedGame setValue:@"Player 2" forKey:@"opponent"];
    [savedGame setValue:selfThrow forKey:@"myThrow"];
    [savedGame setValue:otherThrow forKey:@"theirThrow"];

    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }

}
```

RPSGameViewController.m

```
-(void)completeGame {
    NSManagedObjectContext *context =
                [[[UIApplication sharedApplication] delegate] managedObjectContext];
    NSManagedObject *savedGame =
                [NSEntityDescription insertNewObjectForEntityForName:@"SavedGame"
                                              inManagedObjectContext:context];
    [savedGame setValue:@"Player 2" forKey:@"opponent"];
    [savedGame setValue:selfThrow forKey:@"myThrow"];
    [savedGame setValue:otherThrow forKey:@"theirThrow"];

    if ([selfThrow rpsBeats:otherThrow]) {
        outcomeLabel.text = @"Player 1 Wins!";
    }
    else {
        outcomeLabel.text = @"Player 2 Wins!";
    }

    [context save:nil];
}
```

SQLite

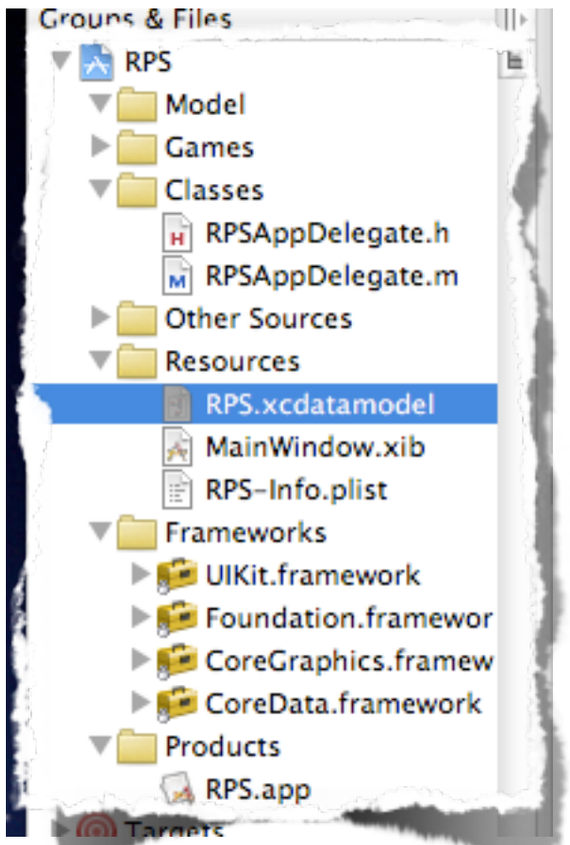
```
sqlite> .tables
ZSAVEDGAME    Z_METADATA
Z_PRIMARYKEY
sqlite> select * from ZSAVEDGAME;
1|1|1|Player 2|paper|rock
2|1|1|Player 2|paper|scissors
sqlite>
```

But is this a safe way to code?

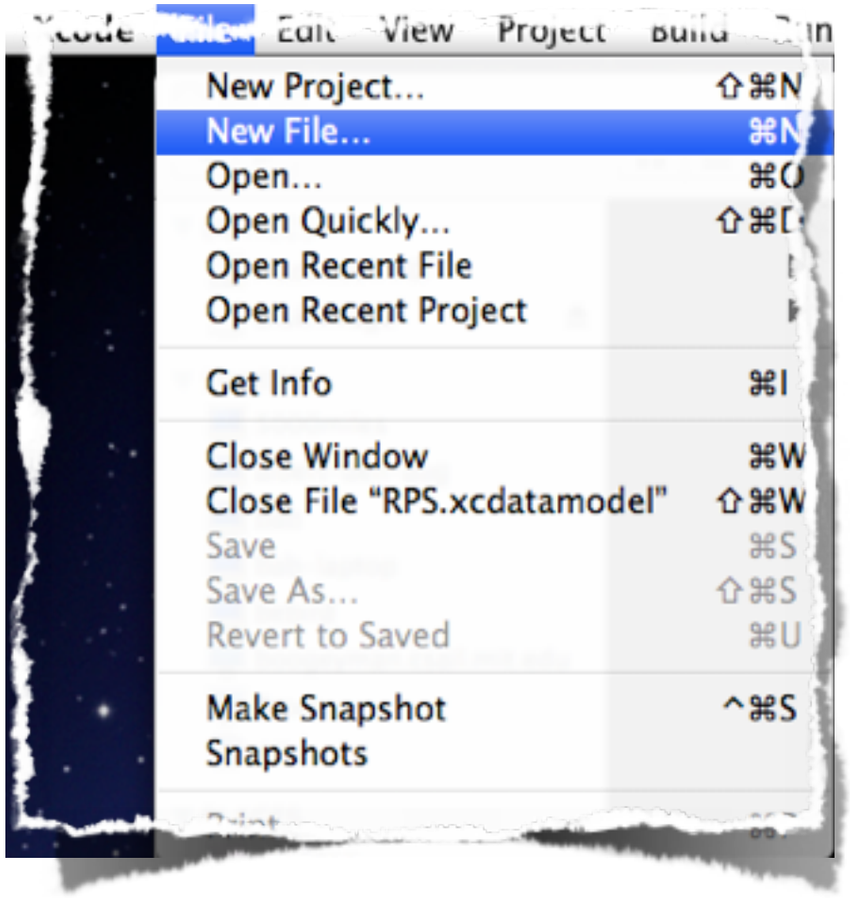
```
NSManagedObject *savedGame;
 [savedGame setValue:@"Player 2" forKey:@"opponent"];
```

Auto-generate Stubs for your Models

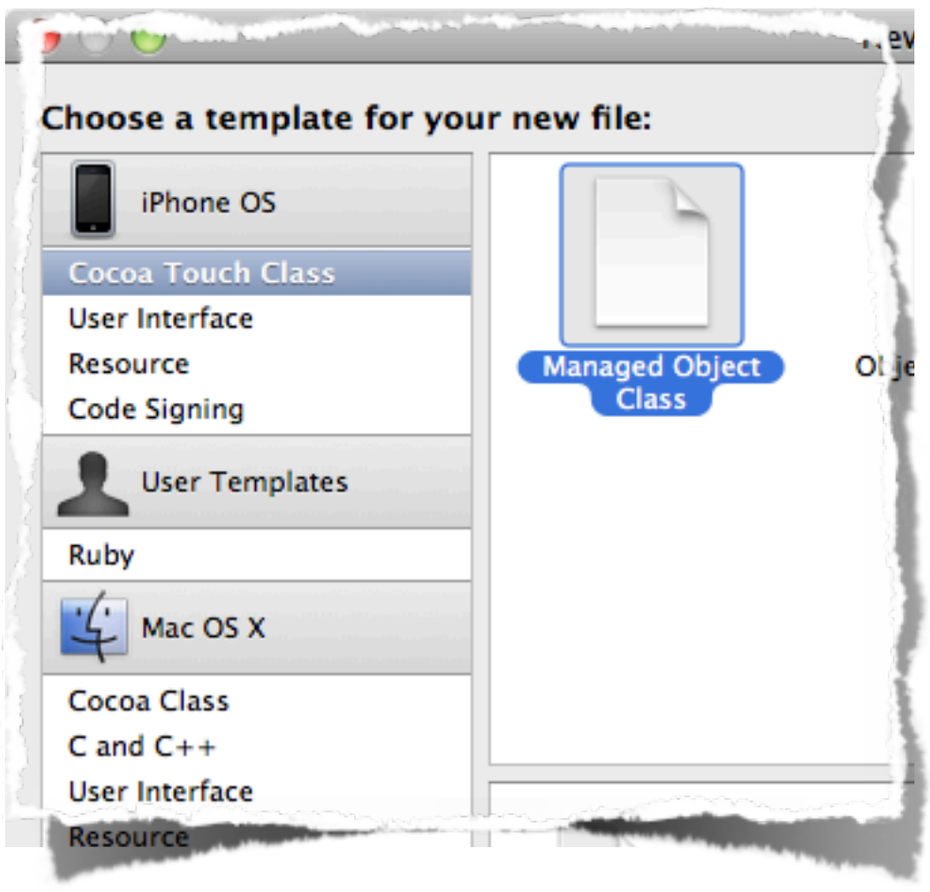
Guarantees static type safety



Save & Select Data Model



File -> New File



Managed Object Class

Core Data

Using the Data

Entity Fetch Requests

GameHistoryController.h

```
@interface GameHistoryController : UITableViewController {
  NSMutableArray *savedGames;
}
```

GameHistoryController.m

```
- (void)dealloc {
  [savedGames release];
    [super dealloc];
}
```

```objc
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];
        NSEntityDescription *entity = [NSEntityDescription entityForName:@"SavedGame"
                                                    inManagedObjectContext:context];
        [request setEntity:entity];

        NSError *error;
        savedGames = [[[self managedObjectContext] executeFetchRequest:request
                                                                error:&error] mutableCopy];
        [request release];

    }
    return self;
}
```

```
-(id)init {
    if (self = [super init]) {
        self.title = @"History";

    }
    return self;
}
```

Goal: Fetch saved games from the object store

```
-(id)init {
   if (self = [super init]) {
       self.title = @"History";
       NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                    managedObjectContext];

   }
   return self;
}
```

```objc
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];

    }
    return self;
}
```

```
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];
        NSEntityDescription *entity = [NSEntityDescription entityForName:@"SavedGame"
                                                    inManagedObjectContext:context];

    }
    return self;
}
```

```objc
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];
        NSEntityDescription *entity = [NSEntityDescription entityForName:@"SavedGame"
                                                    inManagedObjectContext:context];
        [request setEntity:entity];


    }
    return self;
}
```

```
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];
        NSEntityDescription *entity = [NSEntityDescription entityForName:@"SavedGame"
                                                    inManagedObjectContext:context];
        [request setEntity:entity];

        NSError *error;

    }
    return self;
}
```

```
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];
        NSEntityDescription *entity = [NSEntityDescription entityForName:@"SavedGame"
                                                    inManagedObjectContext:context];
        [request setEntity:entity];

        NSError *error;
        savedGames = [[context executeFetchRequest:request
                                                        error:&error] mutableCopy];

    }
    return self;
}
```

```objc
-(id)init {
    if (self = [super init]) {
        self.title = @"History";
        NSManagedObjectContext *context = [[[UIApplication sharedApplication] delegate]
                                                                managedObjectContext];
        NSFetchRequest *request = [[NSFetchRequest alloc] init];
        NSEntityDescription *entity = [NSEntityDescription entityForName:@"SavedGame"
                                                  inManagedObjectContext:context];
        [request setEntity:entity];

        NSError *error;
        savedGames = [[context executeFetchRequest:request
                                                        error:&error] mutableCopy];
        [request release];

    }
    return self;
}
```

You can guess what will come next

```
// Customize the number of rows in the table view.
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)
section {
    return [savedGames count];
}
```

```
- (UITableViewCell *)tableView:(UITableView *)
tableView cellForRowAtIndexPath:(NSIndexPath *)
indexPath {
```

UITableViewCellStyleDefault → UITableViewCellStyleSubtitle

```
NSManagedObject *savedGame = [savedGames objectAtIndex:indexPath.row];
cell.textLabel.text = [savedGame valueForKey:@"opponent"];
cell.detailTextLabel.text = [NSString stringWithFormat:@"My %@ to their %@",
                             [savedGame valueForKey:@"myThrow"],
                             [savedGame valueForKey:@"theirThrow"]];
```


Carrier
1:35 PM
History
Player 2
My rock to their paper
Player 2
My scissors to their paper
Player 2
My rock to their paper
Player 2
My scissors to their paper
Player 2
My rock to their rock
History
Gambits

Of course, we're leaving out many things

Updating the table when new data arrives

Sorting by date

Drilling down on detail views