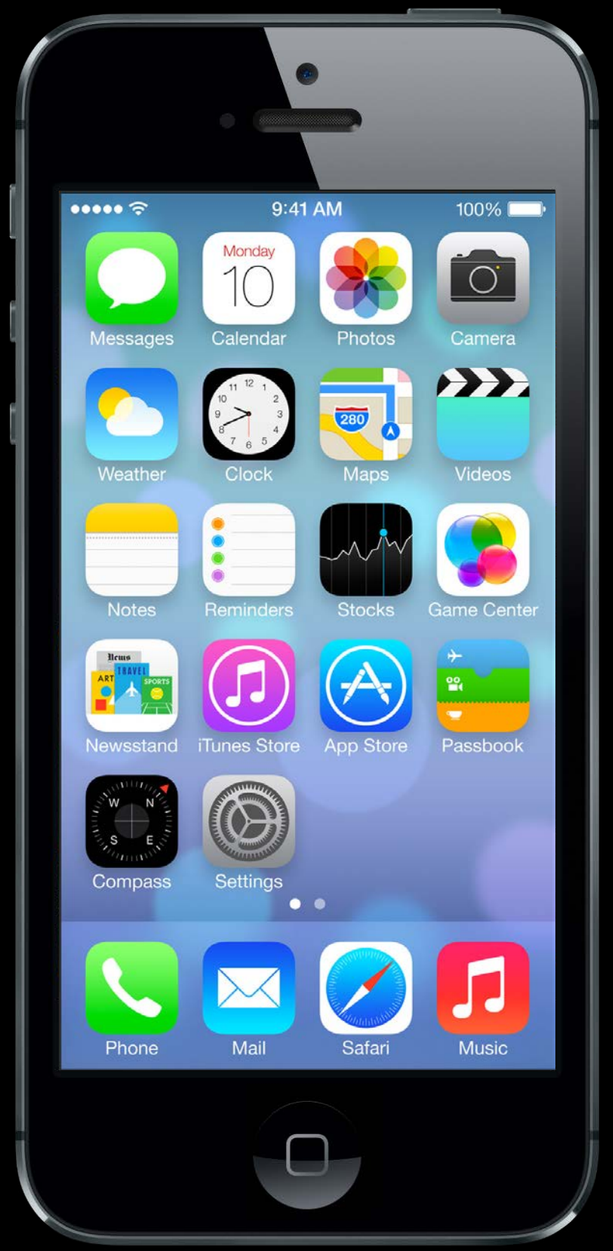
Stanford CS193p

Developing Applications for iOS

Fall 2013-14



Core Data and UITableView

- How to hook these up
  - As you can imagine, they were (probably literally) made for each other!
  - The magic to doing this? `NSFetchedResultsController` …

Core Data and UITableView

- NSFetchedResultsController

  Simply hooks an NSFetchRequest up to a UITableViewController

  Usually you'll have an NSFetchedResultsController @property in your UITableViewController.

  It will be hooked up to an NSFetchRequest that returns the data you want to show in your table.

  Then use it to answer all your UITableViewDataSource protocol's questions!

- For example …

```
- (NSUInteger)numberOfSectionsInTableView:(UITableView *)sender
{
    return [[self.fetchedResultsController sections] count];
}

- (NSUInteger)tableView:(UITableView *)sender numberOfRowsInSection:(NSUInteger)section
{
    return [[[self.fetchedResultsController sections] objectAtIndex:section] numberOfObjects];
}
```

NSFetchedResultsController

- Very important method … objectAtIndexPath:

NSFetchedResultsController method …

```
- (NSManagedObject *)objectAtIndexPath:(NSIndexPath *)indexPath;
```

Here's how you would use it in, for example, tableView:cellForRowAtIndexPath: …

```
- (UITableViewCell *)tableView:(UITableView *)sender
         cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = ...;
    NSManagedObject *managedObject =  // or, e.g., Photo *photo = (Photo *) …
        [self.fetchedResultsController objectAtIndexPath:indexPath];
   // load up the cell based on the properties of the managedObject
   // of course, if you had a custom subclass, you'd be using dot notation to get them
   return cell;
}
```

NSFetchedResultsController

- How do you create an NSFetchedResultsController?

  Just need the NSFetchRequest to drive it (and a NSManagedObjectContext to fetch from).
  Let's say we want to show all photos taken by someone with the name photogName in our table:

```
NSFetchRequest *request = [NSFetchRequest fetchRequestWithEntityName:@“Photo”];
request.sortDescriptors = @[[NSSortDescriptor sortDescriptorWithKey:@“title” ...]];
request.predicate = [NSPredicate predicateWithFormat:@“whoTook.name = %@”, photogName];

NSFetchedResultsController *frc = [[NSFetchedResultsController alloc]
    initWithFetchRequest:(NSFetchRequest *)request
    managedObjectContext:(NSManagedObjectContext *)context
      sectionNameKeyPath:(NSString *)keyThatSaysWhichSectionEachManagedObjectIsIn
               cacheName:@“MyPhotoCache”];  // careful!
```

  Be sure that any cacheName you use is always associated with exactly the same request.
  It's okay to specify nil for the cacheName (no cacheing of fetch results in that case).

  It is critical that the sortDescriptor matches up with the keyThatSaysWhichSection...
  The results must sort such that all objects in the first section come first, second second, etc.

NSFetchedResultsController

- NSFRC also "watches" changes in Core Data and auto-updates table

  Uses a key-value observing mechanism.

  When it notices a change, it sends message like this to its `delegate` …

```
- (void)controller:(NSFetchedResultsController *)controller
   didChangeObject:(id)anObject
       atIndexPath:(NSIndexPath *)indexPath
     forChangeType:(NSFetchedResultsChangeType)type
      newIndexPath:(NSIndexPath *)newIndexPath
{
   // here you are supposed call appropriate UITableView methods to update rows
   // but don't worry, we're going to make it easy on you …
}
```

Demo

- Photomania
  - Gets recent photos from Flickr.
  - Shows a list of photographers who took all the photos.
  - Select a photographer -> shows a list of all the photos that photographer took.
  - Core Data Entities: Photographer and Photo.
- Watch for ...
  - How we define our database schema graphically in Xcode.
  - How we create `NSManagedObject` subclasses and then add categories to them.
  - Especially how we use categories to create "factory" methods to create/initialize database objects.
  - The Application Delegate (finally!)
  - `NSManagedObjectContext`
  - Background Fetching
  - Background URL Sessions
  - `NSNotification` posting and listening
  - How we use `CoreDataTableViewController` to hook the table views up to the database.

Coming Up

- Next Week
  - More Multitasking
  - Advanced Segueing
  - Map Kit?