Introduction to RestKit

Blake Watters

Technology Overview

- Integrated HTTP stack
  - NSURLConnection based
  - Simplifies common tasks
- Object Mapping
  - Build domain objects for REST resources
- Core Data Integration
  - Active Record Pattern Implementation
  - Extends Object Mapping
- Table UI Toolkit
  - Map objects to table cells

Project Overview

- Apache Licensed
- Built on Core Apple Technologies
- Production Ready
- Well supported
- Fast Moving
- Large, active community
- Over 1100 Github Watchers
- ~200 Forks
- 500+ Mailing List Members

Network Layer

RestKit

Initializing RKClient

```
- (void)initRKClient {
    // Initialize with a Base URL
    RKClient* client = [RKClient clientWithBaseURL:@"http://restkit.org"];

    // Setup HTTP AUTH
    client.username = @"restkit";
    client.password = @"rocks";

    // Set an app-wide API key HTTP header
    [client setValue:@"123456" forHTTPHeaderField:@"X-RESTKIT-API-KEY"];

    // The first initialized RKClient becomes
    // the sharedClient instance
    [[RKClient sharedClient] isNetworkAvailable];
}
```

RestKit

Sending Requests

```
- (void)sendRequest {
    // Send an HTTP GET request to 'http://restkit.org/contacts'
    [[RKClient sharedClient] get:@"/contacts" delegate:self];
}

// RKRequestDelegate methods

- (void)request:(RKRequest *)request didLoadResponse:(RKResponse *)response {
    RKLogInfo(@"Yay! We Got a response");
}

- (void)request:(RKRequest*)request didFailLoadWithError:(NSError *)error {
    RKLogInfo(@"Oh no! We encountered an error: %@", [error
localizedDescription]);
}
```

RestKit

Processing Responses

```
- (void)sendRequest {
    // Send an HTTP GET request to 'http://restkit.org/contacts'
    [[RKClient sharedClient] get:@"/contacts" delegate:self];
}

- (void)request:(RKRequest *)request didLoadResponse:(RKResponse *)response {
    RKLogInfo(@"Request Headers: %@", [response allHeaderFields]);
    RKLogInfo(@"Cookies: %@", [response cookies])

    if ([response isSuccessful]) {
        // Response status was 200..299
        if ([response isCreated] && [response isJSON]) {
            // Looks like we have a 201 response of type 'application/json'
            RKLogInfo(@"The JSON is %@", [response bodyAsJSON]);
        }
    } else if ([response isError]) {
        // Response status was either 400..499 or 500..599
        RKLogInfo(@"Ouch! We have an HTTP error. Status Code description: %@",
[response localizedStatusCodeString]);
    }
}
```

RestKit

Requests with Parameters

```
- (void)sendRequestWithParams {
    // Simple params
    NSDictionary* paramsDictionary = [NSDictionary dictionaryWithObjectsAndKeys:
                                                    @"Joe Blow", @"name",
                                                    @"Acme, Inc", @"company", nil];
    [[RKClient sharedClient] post:@"/contacts" params:paramsDictionary
delegate:self];

    // Multi-part params via RKParams
    RKParams* params = [RKParams paramsWithDictionary:paramsDictionary];
    NSData* imageData = UIImagePNGRepresentation([UIImage
imageNamed:@"picture.jpg"]);
    [params setData:imageData MIMEType:@"image/png" forParam:@"photo"];
    [params setFile:@"bio.txt" forParam:@"attachment"];
    [[RKClient sharedClient] post:@"/contacts" params:params delegate:self];
}
```

RestKit

Reachability

```
- (void)demoReachability {
    // Check if the network is available
    [[RKClient sharedClient] isNetworkAvailable];

    // Register for changes in network availability
    NSNotificationCenter* center = [NSNotificationCenter defaultCenter];
    [center addObserver:self selector:@selector(reachabilityDidChange:)
name:RKReachabilityStateChangedNotification object:nil];
}

- (void)reachabilityDidChange:(NSNotification *)notification {
    RKReachabilityObserver* observer = (RKReachabilityObserver *) [notification
object];
    RKReachabilityNetworkStatus status = [observer networkStatus];
    if (RKReachabilityNotReachable == status) {
        RKLogInfo(@"No network access!");
    } else if (RKReachabilityReachableViaWiFi == status) {
        RKLogInfo(@"Online via WiFi!");
    } else if (RKReachabilityReachableViaWWAN == status) {
        RKLogInfo(@"Online via Edge or 3G!");
    }
}
```

RestKit

Request Queue

```
- (IBAction)queueRequests {
    RKRequestQueue* queue = [RKRequestQueue new];
    queue.delegate = self;
    queue.concurrentRequestsLimit = 1;
    queue.showsNetworkActivityIndicatorWhenBusy = YES;

    // Queue up 4 requests
    [queue addRequest:[RKRequest requestWithURL:[NSURL URLWithString:@"http://
restkit.org"] delegate:nil]];
    [queue addRequest:[RKRequest requestWithURL:[NSURL URLWithString:@"http://
restkit.org"] delegate:nil]];
    [queue addRequest:[RKRequest requestWithURL:[NSURL URLWithString:@"http://
restkit.org"] delegate:nil]];
    [queue addRequest:[RKRequest requestWithURL:[NSURL URLWithString:@"http://
restkit.org"] delegate:nil]];

    // Start processing!
    [queue start];

    [queue cancelAllRequests];
}
```

Object Mapping

Initializing the Object Manager

```
- (void)initObjectManager {
    RKObjectManager *manager = [RKObjectManager objectManagerWithBaseURL:@"http://
restkit.org"];

    // Ask for & generate JSON
    manager.acceptMIMEType = RKMIMETypeJSON;
    manager.serializationMIMEType = RKMIMETypeJSON;

    // Object manager has a client
    [manager.client setValue:@"123456" forHTTPHeaderField:@"X-RESTKIT-API-KEY"];
}
```

RestKit

Modeling a RESTful Service

/contacts

```
[{
   'contact': {
      'id': 1234,
      'name': 'Blake Watters',
      'email': 'blake@restkit.org',
      'company': 'GateGuru',
      'birth_date': '11/27/1982'
   }
},
{
   'contact': {
      'id': 3456,
      'name': 'John Doe',
      'email': 'john@doe.com',
      'company': 'Acme, Inc'
   }
}]
```

@interface Contact

```
@interface Contact : NSObject

 @property (retain) NSNumber* contactID;
 @property (retain) NSString* name;
 @property (retain) NSString* email;
 @property (retain) NSString* company;
 @property (retain) NSDate* birthDate;
@end
```

RestKit

Configuring an Object Mapping

```
- (void)setupContactMapping {
    RKObjectMapping *contactMapping = [RKObjectMapping mappingForClass:[Contact class]];
    [contactMapping mapKeyPath:@"id" toAttribute:@"contactID"];
    [contactMapping mapKeyPath:@"birth_date" toAttribute:@"birthDate"];
    [contactMapping mapAttributes:@"name", @"email", @"company", nil];

    NSDateFormatter *dateFormatter = [NSDateFormatter new];
    [dateFormatter setDateFormat:@"MM/dd/yyyy"]; // 11/27/1982
    dateFormatter.timeZone = [NSTimeZone timeZoneWithAbbreviation:@"UTC"];
    contactMapping.dateFormatters = [NSArray arrayWithObject:dateFormatter];

    [[RKObjectManager sharedManager].mappingProvider setMapping:contactMapping
forKeyPath:@"contact"];
}
```

RestKit

Loading Remote Objects

```
- (void)loadRemoteObjects {
    [[RKObjectManager sharedManager] loadObjectsAtResourcePath:@"/contacts" delegate:self];
}

- (void)objectLoader:(RKObjectLoader *)objectLoader didLoadObjects:(NSArray *)objects {
    if ([objectLoader wasSentToResourcePath:@"/contacts"]) {
          // Introspect the resource path
          NSLog(@"Nice! We loaded the following contacts: %@", objects);
    }
}

- (void)objectLoader:(RKObjectLoader *)objectLoader didFailWithError:(NSError *)error {
    // Note that failures here can be at the _application_ level in addition to transport
    NSLog(@"Rats! Failed to load objects: %@", [error localizedDescription]);
}
```

RestKit

What just happened?

- RKObjectManager configured to map ‘contact’ dictionaries to Contact class
- Asynchronous GET sent to ‘/contacts’ via RKObjectLoader
- 200 response returned, with JSON body
- RKObjectMapper parsed payload and mapped JSON data to Contact objects
- Callback invoked with array of Contacts

Configuring the Router

```
- (void)configureRouter {
    RKObjectRouter* router = [RKObjectManager sharedManager].router;

    // Configure a default route
    [router routeClass:[Contact class] toResourcePath:@"/contacts/:contactID"];
    [router routeClass:[Contact class] toResourcePath:@"/contacts" forMethod:RKRequestMethodPOST];
}
```

RestKit

RESTful Object Manipulation

```
// Create a new Contact
- (void)createObject {
    Contact* contact = [Contact new];
    contact.name = @"RestKit User";
    contact.email = @"user@restkit.org";

    // POST to /contacts
    [[RKObjectManager sharedManager] postObject:contact delegate:self];
}

// Edit Contact with ID 12345
- (void)editObject {
    Contact* contact = [Contact new];
    contact.contactID = [NSNumber numberWithInt:12345];
    contact.name = @"New Name";

    // POST to /contacts/12345
    [[RKObjectManager sharedManager] putObject:contact delegate:self];
}

// Delete Contact with ID 321
- (void)deleteObject {
    Contact* contact = [Contact new];
    contact.contactID = [NSNumber numberWithInt:321];

    // DELETE to /contacts/321
    [[RKObjectManager sharedManager] deleteObject:contact delegate:self];
}
```

Core Data

Configuring RKManagedObjectStore

```
- (void)configureObjectStore {
  // Initialize the RestKit Object Manager
  RKObjectManager* objectManager = [RKObjectManager objectManagerWithBaseURL:@"http://restkit.org"];

  // Initialize object store
  // We are using the Core Data support, so we have initialized a managed object store backed
  // with a SQLite database.
  objectManager.objectStore = [RKManagedObjectStore objectStoreWithStoreFilename:@"Contacts.sqlite"];
}
```

Making Contact Persistent

```
#import <RestKit/CoreData/CoreData.h>

@interface Contact : NSManagedObject

@end

@implementation Contact

@dynamic name;
@dynamic email;
@dynamic company;
@dynamic contactID;

@end

- (void)coreDataMapping {
    RKManagedObjectMapping *mapping = [RKManagedObjectMapping mappingForClass:
[Contact class] inManagedObjectStore:store];
    mapping.primaryKeyAttribute = @"contactID";
}
```

Working with Core Data

```
- (void)workWithCoreData {
    // Get all the Contacts
    NSArray* contacts = [Contact findAll];

    // Count the Contacts
    NSError* error = nil;
    NSUInteger count = [Contact count:&error];

    // Find Contact by primary key
    NSNumber* contactID = [NSNumber numberWithInt:12345];
    Article* somebody = [Contact findFirstByAttribute:@"contactID"
withValue:contactID];

    // Find Contacts with criteria
    NSPredicate* predicate = [NSPredicate predicateWithFormat:@"name contains[cd]
'restkit'"];
    NSArray* matches = [Contact findAllWithPredicate:predicate];

    // Find first 10 Contacts, sorted by name
    NSSortDescriptor* sortDescriptor = [NSSortDescriptor
sortDescriptorWithKey:@"name" ascending:YES];
    NSFetchRequest* fetchRequest = [Contact fetchRequest];
    [fetchRequest setFetchLimit:10];
    [fetchRequest setSortDescriptors:[NSArray arrayWithObject:sortDescriptor]];
    NSArray* sortedContacts = [Contact executeFetchRequest:fetchRequest];

}
```

RestKit

Database Seeding

```
// This is typically configured as a secondary target on your project
// Dump your seed data out of your backend system in JSON format
// Add to the project as resources
// Run the secondary target in the Simulator
- (void)seedTheDatabase {
    // Setup the object manager
    RKObjectManager* objectManager = [RKObjectManager objectManagerWithBaseURL:@"http://
restkit.org"];
    objectManager.objectStore = [RKManagedObjectStore
objectManagerWithStoreFilename:@"ContactsSeed.sqlite"];

    // Load all the data from the file contacts.json into a seed database
    // The seeder will print instructions for how to copy the data to your app
    RKObjectSeeder* seeder = [RKObjectSeeder objectSeederWithObjectManager:objectManager];
    [seeder seedObjectsFromFiles:@"contacts.json", nil];
    [seeder finalizeSeedingAndExit];
}
```

RestKit UI

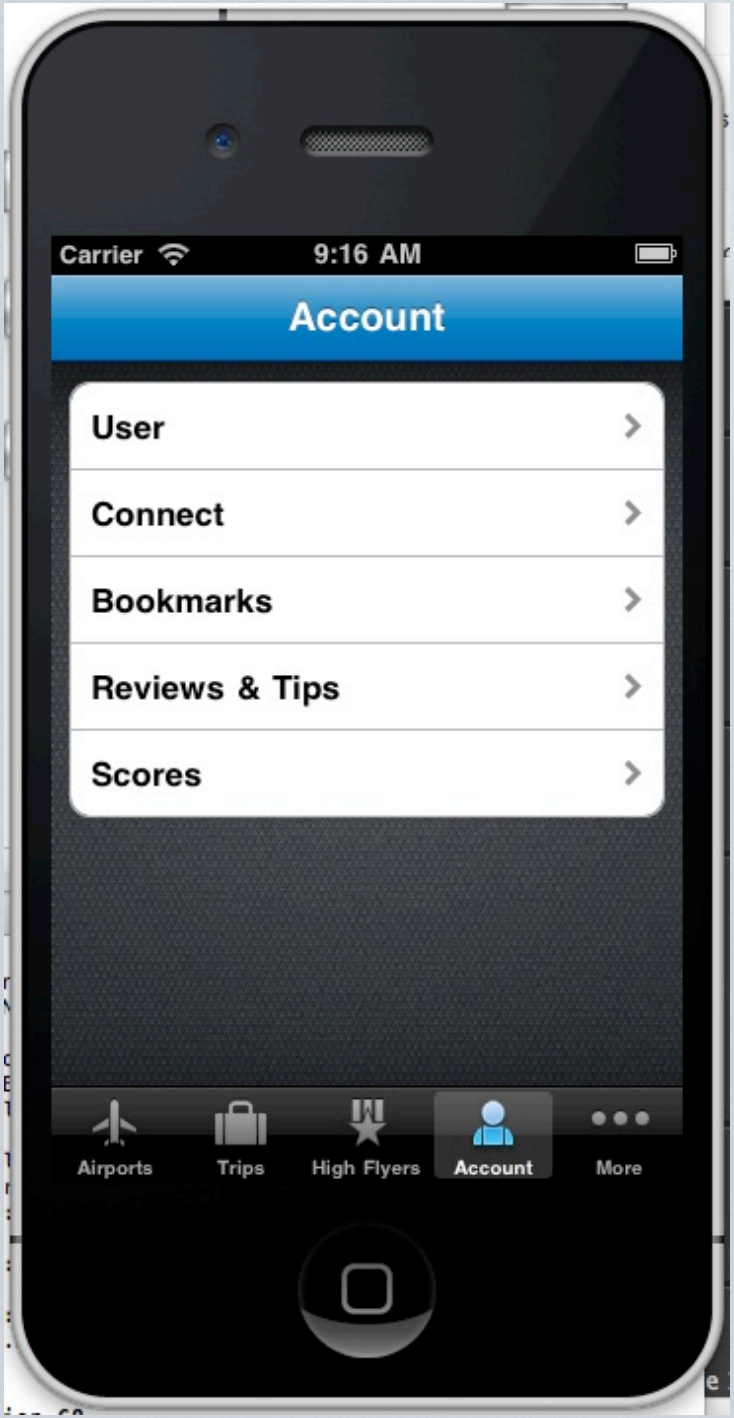
RestKit

Static Table

```
- (void)loadStaticTable {
   RKTableController *tableController = [RKTableController
tableControllerWithTableView:self.tableView];
    NSArray* tableItems = [NSArray arrayWithObjects:
       [RKTableItem tableItemWithText:@"User" URL:@"gg://user"],
       [RKTableItem tableItemWithText:@"Connect" URL:@"gg://user/connect"],
       [RKTableItem tableItemWithText:@"Bookmarks" URL:@"gg://user/bookmarks"],
       [RKTableItem tableItemWithText:@"Reviews & Tips" URL:@"gg://user/reviews"],
       [RKTableItem tableItemWithText:@"Scores" URL:@"gg://user/scores"],
       nil];

    // Load the table view
    [self.tableController loadTableItems:tableItems
withMappingBlock:^(RKTableViewCellMapping* cellMapping) {
        cellMapping.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
        cellMapping.onSelectCellForObjectAtIndexPath = ^(UITableViewCell* cell, id
object, NSIndexPath* indexPath) {
            TTOpenURL([object valueForKey:@"URL"]);
        };
    }];
}
```

RestKit


Carrier
9:16 AM
Account
User
Connect
Bookmarks
Reviews & Tips
Scores
Airports
Trips
High Flyers
Account
More

Network Table

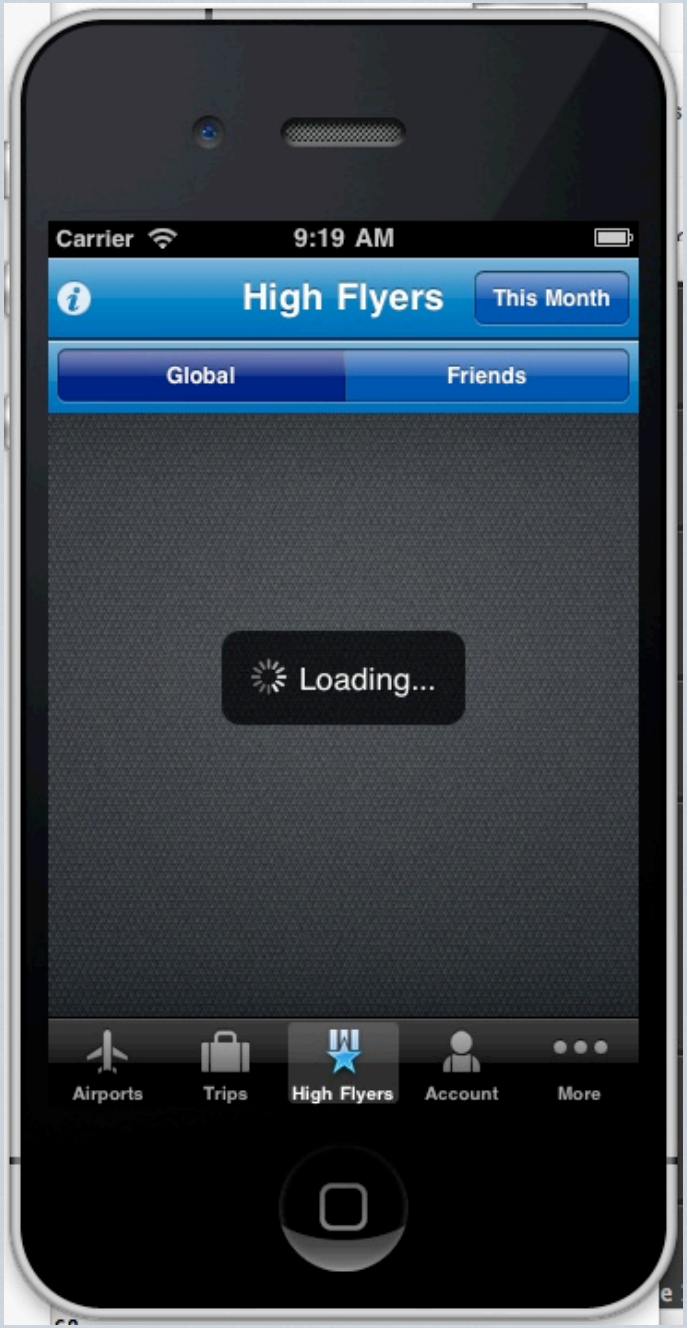
RestKit

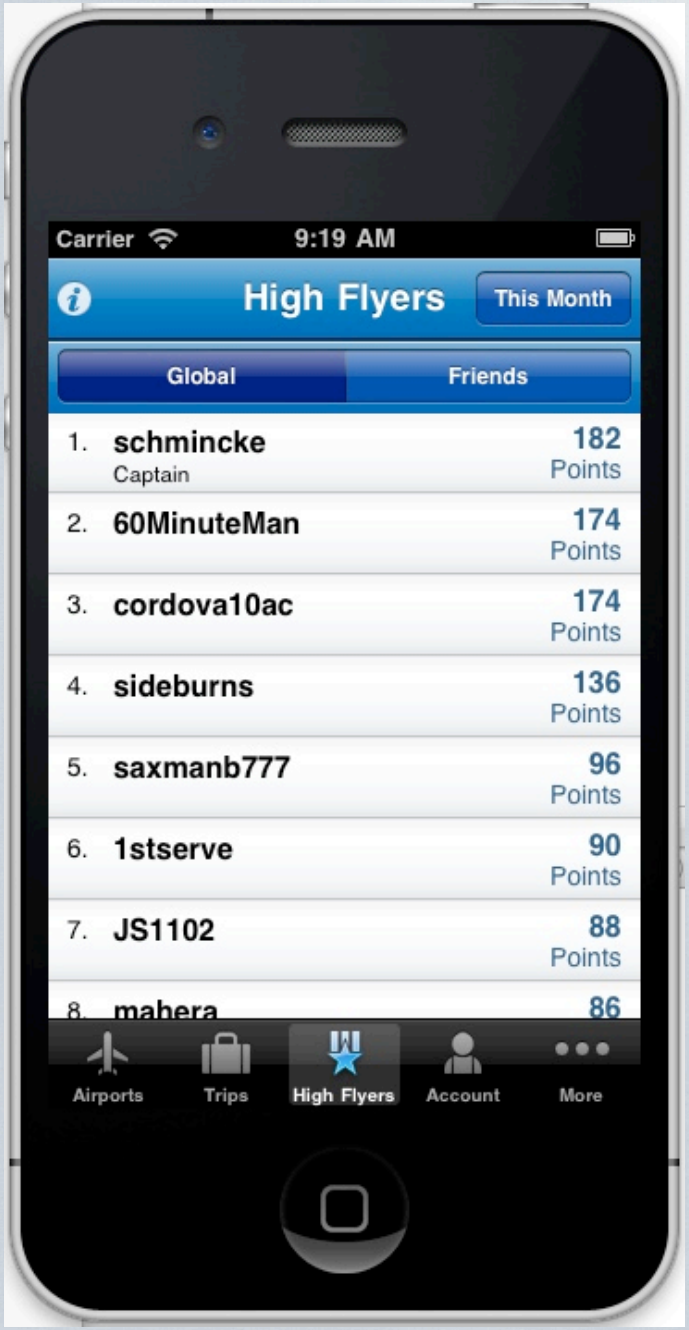
```
- (void)loadTableView {
    // Setup the Table View Model
    self.tableController = [RKTableController tableControllerWithTableView:self.tableView];
    self.tableController.delegate = self;
    self.tableController.imageForEmpty = [UIImage imageNamed:@"no_high_flyers.png"];
    self.tableController.imageForOffline = [UIImage imageNamed:@"offline.png"];

    [self.tableController mapObjectsWithClass:[GGHighFlyer class] toTableCellsWithMapping:
[RKTableViewCellMapping cellMappingWithBlock:^(RKTableViewCellMapping* cellMapping) {
        cellMapping.cellClass = [GGHighFlyerTableViewCell class];
        cellMapping.selectionStyle = UITableViewCellSelectionStyleNone;
        cellMapping.rowHeight = 44;
        [cellMapping mapAttributes:@"points", @"login", nil];
        cellMapping.onCellWillAppearForObjectAtIndexPath = ^(UITableViewCell* cell, id object,
NSIndexPath* indexPath) {
            GGHighFlyerTableViewCell* highFlyerCell = (GGHighFlyerTableViewCell*) cell;
            highFlyerCell.captain = (indexPath.row == 0);
            highFlyerCell.rank = indexPath.row + 1;
        };
    }]];

    NSString* resourcePath = [NSString stringWithFormat:@"/airports/%@/high_flyers.json", [_airport
airportId]];
    NSString* resourcePathWithQueryString = [self addQueryStringToResourcePath:resourcePath];
    [self.tableController loadTableFromResourcePath:resourcePathWithQueryString];
}
```

RestKit


Carrier
9:19 AM
High Flyers
This Month
Global
Friends
Loading...
Airports
Trips
High Flyers
Account
More


Carrier
9:19 AM
High Flyers
This Month
Global
Friends
1. schmincke
Captain
182
Points
2. 60MinuteMan
174
Points
3. cordova10ac
174
Points
4. sideburns
136
Points
5. saxmanb777
96
Points
6. 1stserve
90
Points
7. JS1102
88
Points
8. mabera
86
Airports
Trips
High Flyers
Account
More

RestKit

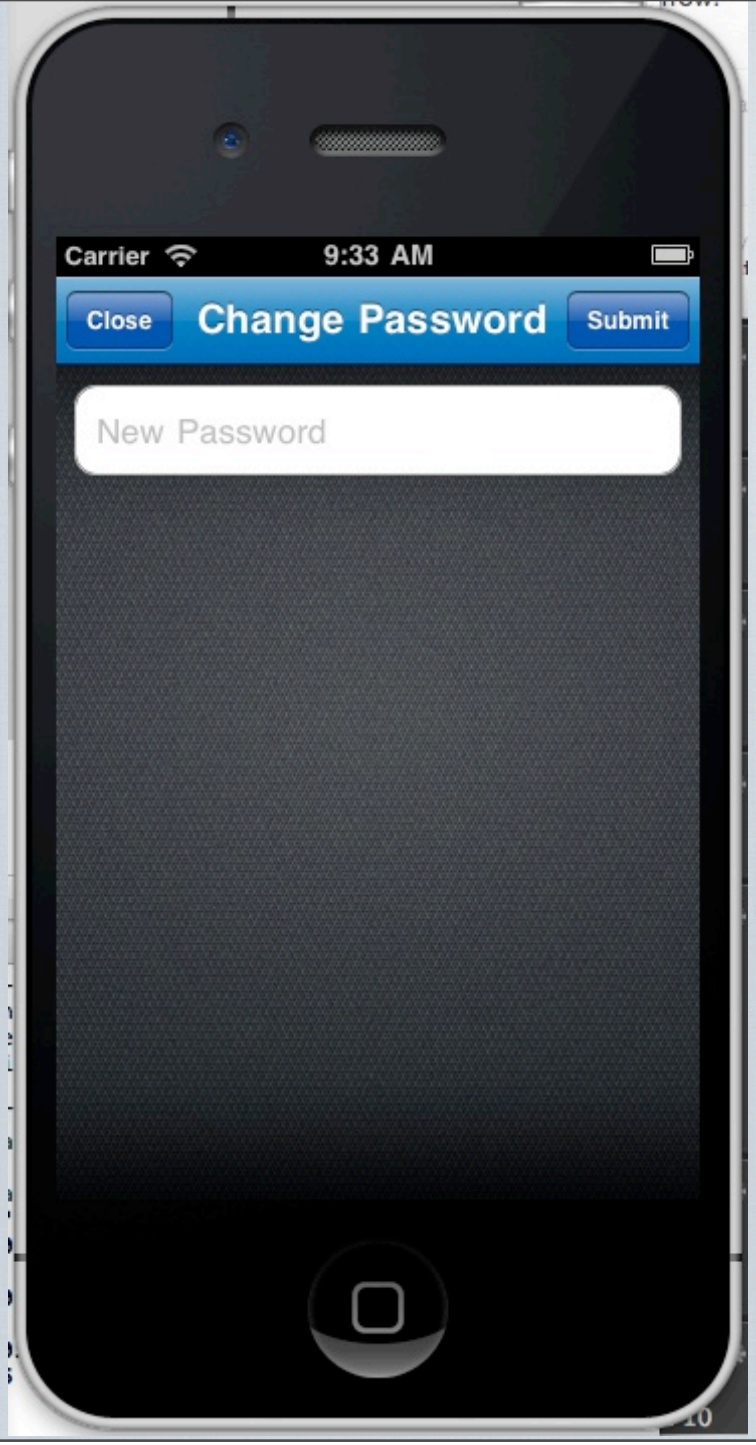
Building a Form

```
- (void)changePasswordForm {
    GGUser *user = [GGUser user];
    RKForm *form = [RKForm formForObject:user withBlock:^(RKForm *form) {
        [form addRowForAttribute:@"password"
withControlType:RKFormControlTypeTextFieldSecure block:^(RKControlTableItem
*tableItem) {
            tableItem.textField.placeholder = @"New Password";
            tableItem.textField.returnKeyType = UIReturnKeyDone;
            tableItem.textField.keyboardAppearance = UIKeyboardAppearanceAlert;
        }];

        form.onSubmit = ^{
            NSError* error = nil;
            GGUser *user = (GGUser *) form.object;
            if (! [[GGUser user] changePassword:user.password error:&error]) {
                GGAlertWithTitleAndMessage(@"Invalid Password", [error
localizedDescription]);
            }
        };
    }];

    [self.tableController loadForm:form];
}
```

RestKit



RestKit

Thank You!

http://restkit.org/

http://twitter.com/restkit

http://github.com/RestKit/RestKit