Bridging the Data Divide

Mapping JSON and XML to Objects in iOS

Requirement

- Access remote web services
- Download data in either JSON or XML
- Create/update objects in iOS apps

Test Case

Access information on routes and services for the Massachusetts Bay Transportation Authority (MBTA) using two different web services/ APIs:

- MBTA realtime v2 — *http://realtime.mbta.com/portal*
- NextBus Inc. — *https://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf*

Three choices examined

- **RestKit**

  https://github.com/RestKit/RestKit

- **RZImport**

  https://github.com/Raizlabs/RZImport

- **Mantle**

  https://github.com/Mantle/Mantle

Three implementations

- **RestKit**

  https://github.com/SteveCaine/MBTA-RestKit

- **RZImport**

  https://github.com/SteveCaine/MBTA-RZImport

- **Mantle**

  https://github.com/SteveCaine/NextBus-Mantle

RestKit

- First release — March 2011
- Latest release — April this year
- 159 contributors, 8,925 ★ on GitHub
- ~5,000 lines of code in 123 files

Pros

- All-in-one package: object-mapping support, networking (including MIME parts)
- CoreData support is built-in

Cons

- Older framework, requires older versions of components such as AFNetworking (uses 1.3.4, latest is 2.6.1)
- XML support is an “add-on” by the authors

RZImport

- First release — June 2014
- Latest release — August this year
- 8 contributors, 13 ★ on GitHub
- ~250 lines of code in 4 files

Pros

- Very lightweight (CoreData support added in separate framework)
- Will map JSON/XML to properties with “similar” names (including `CamelCase` and `snake_case`)

Cons

- No XML support by authors **or** third-parties

Mantle

- Created by GitHub for their own use
- First release — October 2012
- Latest release — September this year
- 57 contributors, 8,156 ★ on GitHub
- ~850 lines of code in 31 files

Pros

- Newer framework compared to RestKit
- Popular with my clients, often mentioned in job interviews
- Bills itself as a “lightweight alternative to CoreData” that can also serve as “convenient translation layer” if you later move to CoreData

Cons

- XML support is an “add-on” by unrelated author

Observations

- Custom data classes in RestKit are simple, because all object mapping is done in class methods on RestKit during setup.
- Custom classes in RZImport and Mantle are more comples as object mapping are defined in those classes (including overriding ‘default’ mappings).
- In Mantle, each custom data class stands alone (inheriting from MTLModel), while RZImport and RestKit allow custom data classes to inherit from each other and share code.

Code

RestKit implementation

```
@interface ApiTime : ApiData
      @property (   copy, nonatomic) NSNumber *server_dt;

[RKObjectMapping mappingForClass:[ApiTime class]]
      addAttributeMappingsFromArray:@[ @"server_dt" ]

[RKResponseDescriptor
      responseDescriptorWithMapping:
      method:<HTTP-GET>
      pathPattern:<request>
      keyPath:<response>
      statusCodes:<ex. 200 for OK>

RKObjectManager getObjectsAtPath:verb
   parameters:params
         success:^(RKObjectRequestOperation *operation, RKMappingResult *mappingResult) {...}
         failure:^(RKObjectRequestOperation *operation, NSError *error) {...}];
```

RZImport implementation

```
@interface ApiTime : ApiData
@property (   copy, nonatomic) NSNumber *server_dt; // MATCHES BY NAME
- (NSDate *)time;
@end

@implementation ApiTime
- (NSDate *)time { // after checking if property is nil
   return [NSDate dateWithTimeIntervalSince1970:[self.server_dt integerValue]];

// factory method to create different ApiData types based on request's verb+params
+ (ApiData *)itemForResponse:(NSDictionary *)response
                        verb:(NSString *)verb
                      params:(NSDictionary *)params {
// responses that return a single dictionary
   result = [ApiTime rzi_objectFromDictionary:response];
// responses that return an array of dictionaries under one key
   result = [[ApiXxx alloc] initWithResponse:response];
// responses that return an array of objects
   result = [ApiXxx rzi_objectsFromArray:response];
```

Mantle implementation

```
@interface TServerTime : MTLModel <MTLJSONSerializing>
@property (   copy, nonatomic, readonly) NSDate *servertime;
@end

@implementation TServerTime
+ (NSDictionary *)JSONKeyPathsByPropertyKey {
      return @{ @"servertime" : @"server_dt" };
}
+ (NSValueTransformer *)servertimeJSONTransformer {
      return [MTLValueTransformer transformerUsingForwardBlock:^id(NSString *value,
BOOL *success, NSError *__autoreleasing *error) {
            return [NSDate dateWithTimeIntervalSince1970:[value integerValue]];
      }];
}
@end
```

RZImport interface

```
//  RZImportable.h
@protocol RZImportable <NSObject>
@optional
+ (NSDictionary *)rzi_customMappings;
+ (NSArray *)rzi_ignoredKeys;
+ (NSArray *)rzi_nestedObjectKeys;
+ (NSString *)rzi_dateFormatForKey:(NSString *)key;
+ (id)rzi_existingObjectForDict:(NSDictionary *)dict;
- (BOOL)rzi_shouldImportValue:(id)value forKey:(NSString *)key;
@end

//  NSObject+RZImport.h
@interface NSObject (RZImport)
+ (instancetype)rzi_objectFromDictionary:(NSDictionary *)dict;
+ (instancetype)rzi_objectFromDictionary:(NSDictionary *)dict withMappings:(NSDictionary *)mappings;
+ (NSArray *)rzi_objectsFromArray:(NSArray *)array;
+ (NSArray *)rzi_objectsFromArray:(NSArray *)array withMappings:(NSDictionary *)mappings;
- (void)rzi_importValuesFromDict:(NSDictionary *)dict;
- (void)rzi_importValuesFromDict:(NSDictionary *)dict withMappings:(NSDictionary *)mappings;
@end
```

Demo

Questions?

on GitHub

- **RestKit**

  https://github.com/SteveCaine/MBTA-RestKit

- **RZImport**

  https://github.com/SteveCaine/MBTA-RZImport

- **Mantle**

  https://github.com/SteveCaine/NextBus-Mantle