Model hoá dữ liệu

Huy Phạm

About me

- 2011 freelancer.
- 2012 RoR developer at Zigexn Ventura.
- 2013 CTO at Lozi Singapore (gọi vốn thành công series A).
- 2016 Consultant cho Pixai project (Pixel Artificial Intelligent).
- Skills: Back-end, Front-end, Mobile, Analytics, Team leader…
- Tools: Golang, Nodejs, Ruby, RabbitMQ, Kafka, Storm, ElasticSearch, PostgresSQL, MongoDB, Redis…

Table of content

1. Model hoá dữ liệu là gì

Model hoá

- Wiki: "A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real world."

- Custom class trong Objective-C:

```
#import "SModel.h"
#import "Carrot.h"

@interface Vegetable : SModel

@property (nonatomic, copy) NSString *_id;
@property (nonatomic, strong) Carrot *carrot;

@end
```

2. Tại sao cần phải model hoá dữ liệu

Làm việc với JSON

- Làm với với JSON trong Object-C và Swift thông qua kiểu dữ liệu NSDictionary và Dictionary.
- Các NSDictionary và Dictionary là các hash.

```
{
    name: "Huy",
    age: 23,
    city: "Saigon",
    country: "Vietnam",
    bio: "Milk Carrot"
}
```

Vấn đề khi sử dụng hash

- Không biết trước các fields.
- Build-in của Foundation.
- Truy cập thông qua hash.
- Bị crash nếu truy cập vào child nếu parent bị nil.
- Có thể save xuống đĩa.

Sử dụng custom class

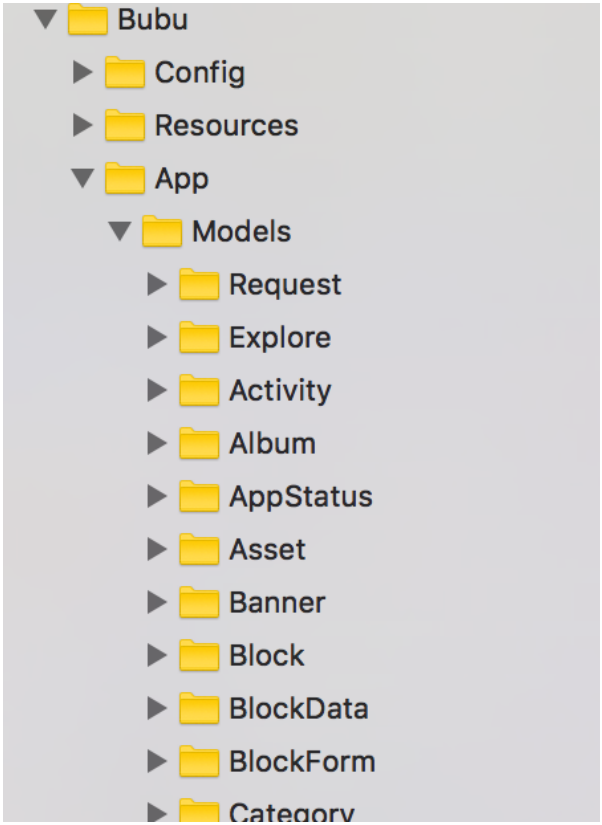
- Biết trước các fields.
- Tự xây dựng.
- Truy cập thông qua con trỏ.
- Có thể handle các trường hợp lỗi dữ liệu.
- Không thể save xuống đĩa dưới dạng thông thường.

Data consistency

- Đảm bảo sự nhất quán dữ liệu trong ứng dụng là điều khó khăn.
- Ví dụ: khi người dùng đổi tên trong trang cá nhân => tất cả các trang liên quan và các phần liên quan mà có hiển thị tên người dùng cũng phải được đổi theo.
- Dữ liệu càng phân tán càng khó để nhất quán.
- Custom class để model hoá dữ liệu sẽ dễ dàng để quản lý và nhất quán được dữ liệu.
- Tránh lỗi typo.

3. Cách xây dựng một model

Custom class




```
@interface BlockModel : Model

@property (nonatomic, copy) NSString *__id;
@property (nonatomic, copy) NSString *slug;
@property (nonatomic, copy) NSString *caption;
@property (nonatomic, copy) NSString *image;
@property (nonatomic, strong) UserModel *createdBy;
@property (nonatomic, strong) UserModel *uploadedBy;
@property (nonatomic, strong) DishModel *dish;
@property (nonatomic, strong) PinModel *pin;
@property (nonatomic, strong) AspectRatioModel *aspectRatio;
@property (nonatomic, strong) ColorModel *color;
@property (nonatomic, strong) BlockCountModel *count;
@property (nonatomic, copy) NSString *createdAt;
@property (nonatomic) NSInteger buyable;
@property (nonatomic) BOOL isLiked;
@property (nonatomic) BOOL isPinned;
@property (nonatomic) BOOL soldOut;
@property (nonatomic) BOOL isSubscribe;
@property (nonatomic, strong) SubCategoryModel *itemCategory;

@end
```

Hình 1

Các vấn đề

- Làm sao khai báo một model:
  Swift: let block = Block()
  Objective-C: var block = [[Block alloc] init]

- Làm sao khởi tạo dữ liệu
  block.id =1
  block.name = "Milk carrot"

- Sử dụng như một class instance thông thường, nhưng như vậy thì nó quá chán?

4. Auto mapping từ Dictionary

Objective-C runtime

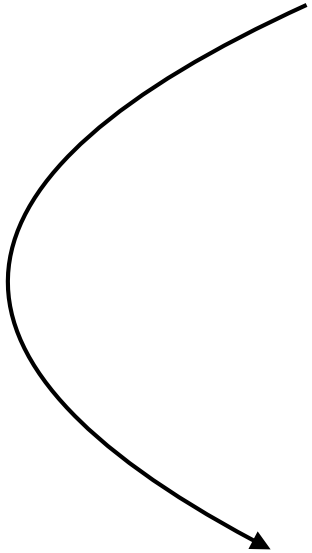
- Là một thư viện runtime chứa các hàm tiện ích để làm việc với object.
- Làm việc với class: Load class từ chuỗi string, lấy super/sub class, lấy name của class.
- Làm việc với đối tượng: lấy tên class của đối tượng, gọi hàm, kiểm tra xem đối tượng có hồi đáp 1 lời gọi function không.
- Tạo ra một lời gọi function bằng cách gửi message.
- Làm việc với các thuộc tính của đối tượng(*).
- Associated một đối tượng với một đối tượng khác (giúp cho việc thêm bớt một thuộc tính vào một class đã có sẵn mà ta không có source code của class đó).
- Và các tính năng khác nữa...

Ý tưởng

- biết được các thuộc tính của đối tượng (mục *) trong thời gian thực thi thì mình hoàn toàn có thể set được giá trị cho các thuộc tính đó trong lúc khởi tạo với dữ liệu là một Dictionary. Và để sử dụng thư viện Objective-C runtime xây dựng nên một model có khả năng tự init data của mình bằng Dictionary mà không cần phải viết hàm init.

Dictionary

```
{
    name: "Huy",
    age: 23,
    city: "Saigon",
    country: "Vietnam",
    bio: "Milk Carrot"
}
```



Model

```
{
    name: String,
    age: float,
    bio: String
}
```

Lấy danh sách thuộc tính

```
# hàm lấy danh sách thuộc tính của một class trong objc-runtime

objc_property_t *properties = class_copyPropertyList([self class],
&outCount);

# lặp qua lần lượt các thuộc tính để lấy ra kiểu dữ liệu
for(int i = 0; i < outCount; i++) {
    objc_property_t property = properties[i];
    objc_property_t property = properties[i];
    const char *propName = property_getName(property);
....
}
```

Chỗ này viết lộn, anh em coi source code đỡ nha :D

Gán giá trị cho thuộc tính

```
# lấy giá trị ứng với thuộc tính trong Dictionary
id value = [dictionary valueForKey:propertyName];

# gán giá trị cho thuộc tính đó trong model.
[self setValue:value forKey:propertyName];
```

Các vấn đề

- Chú ý handle các trường hợp dữ liệu lỗi, không khớp kiểu.
- Chú ý các kiểu dữ liệu đặc biệt như: Boolean, NSDictionary, Array…
- Các key từ khoá trùng với từ khoá của ngôn ngữ: id, description…

Cải tiến

- Phải lấy danh sách thuộc tính mỗi lần init một object init 1000 object phải chạy runtime 1000 lần! cần xây dựng cơ chế cached lại các properties khi khai báo lần đầu.
- Hỗ trợ nested mapping và array mapping.
- Hỗ trợ xử lý các trường hợp key dữ liệu trùng với từ khoá ngôn ngữ.
- Các hàm tiện ích.

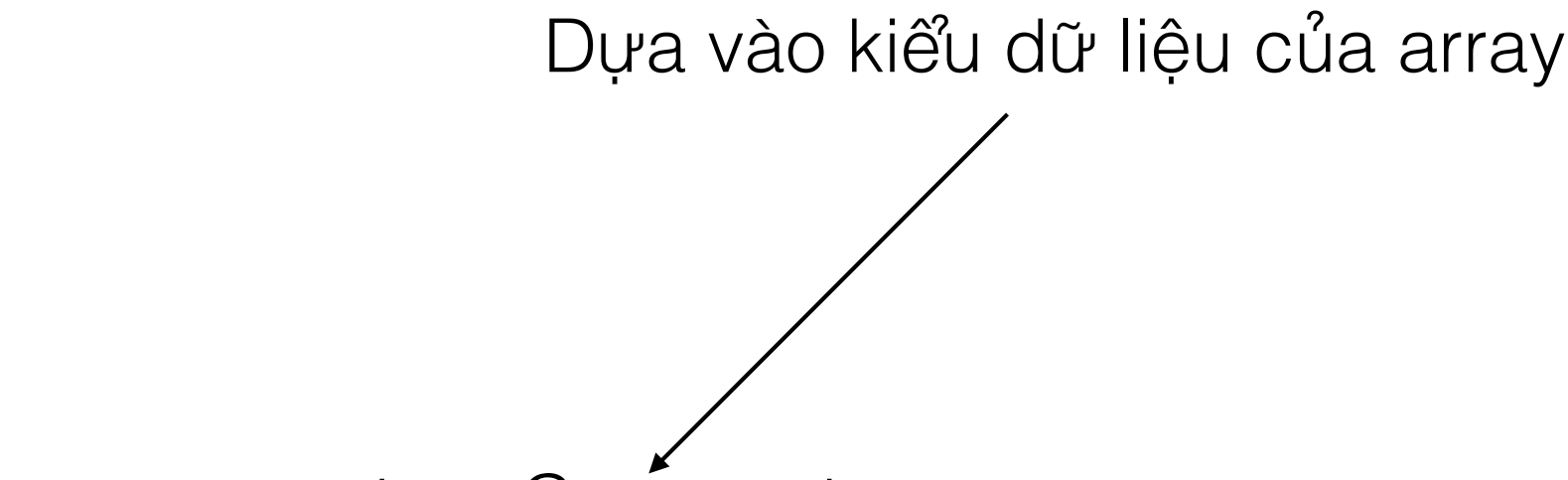
5. Nested mapping

Init model lồng nhau

- Trở lại với hình 1.
- Nếu một model chứa một model => nested model
- Để xử lý trường hợp dữ liệu lồng nhau ta có thể dựa vào hàm: isKindOfClass để init model lồng nhau:

- [propertyType isKindOfClass:[Model class]] =>
  Nested model

6. Array mapping



- comments: <Comment>

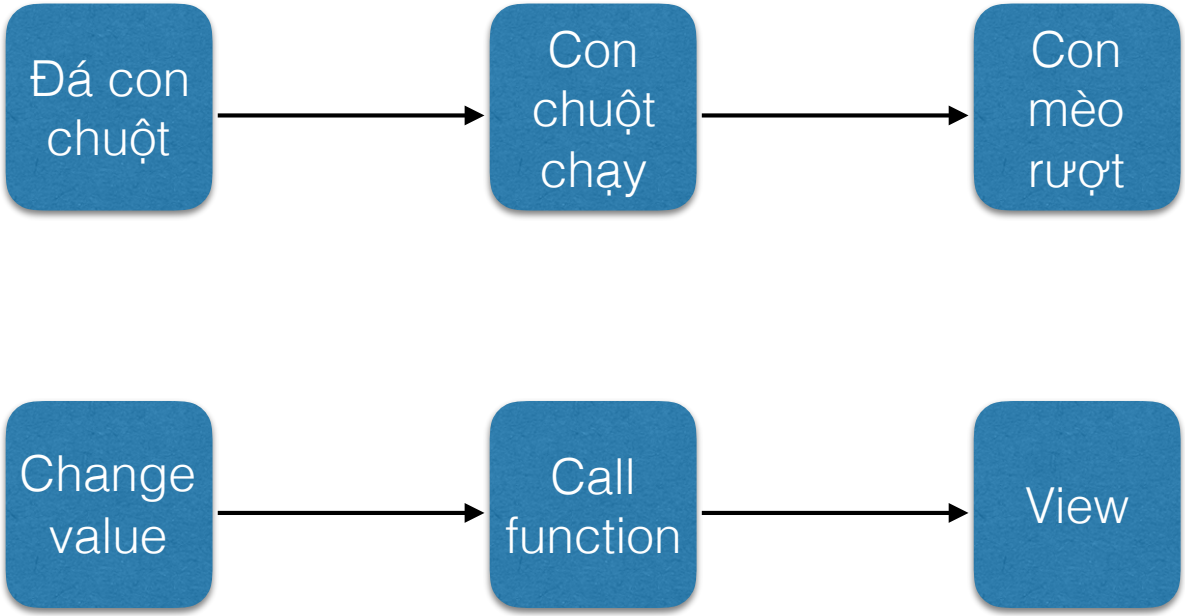
6. Initial value và error handler

Initial value và error handler

- Vì là một custom object nên hoàn toàn có thể tạo các giá trị mặc định khi khởi tạo.
- Mặc định các giá trị default là các giá trị dễ sử dụng: String -> rỗng, int -> số 0, con trỏ -> object rỗng…
- Việc mặc định các giá trị sẽ giúp handle được các trường hợp crash khi truy cập hoặc so sánh, gán dữ liệu…
- Xây dựng các tiện ích init đối tượng qua networking: model.fetchInBackground()

7. Reactive

Reactive


Đá con chuột
Con chuột chạy
Con mèo rượt
Change value
Call function
View