

2013

# Tài Liệu Hướng Dẫn Xây Dựng Ứng Dụng iPhone



Nguyễn Anh Tiệp - Cao Thanh Vàng  
Đại Học Lạc Hồng  
20/11/2013

## MỤC LỤC

<b>CHƯƠNG I CHUẨN BỊ TRƯỚC KHI BẮT ĐẦU XÂY DỰNG ỨNG DỤNG .....</b>	<b>1</b>
<b>1.1 CHUẨN BỊ HỆ ĐIỀU HÀNH MAC OS .....</b>	<b>2</b>
1.1.1 Lập Trình Ứng Dụng Iphone Trên Windows .....	2
1.1.2 Sử Dụng Sản Phẩm Chính Hãng Apple.....	3
1.1.3 Chạy Hệ Điều Hành Mac Os Trên Pc/Laptop Intel/Amd – Tại Sao Không? .....	5
<b>1.2 PHẦN MỀM XCODE .....</b>	<b>7</b>
1.2.1 Cài Đặt Thông Qua Bản Tải Về Từ Trang Dành Cho Developer.....	8
1.2.2 Cài Đặt Thông Qua Apple Mac Store.....	10
1.2.3 Cài Đặt Từ Bản Xcode Được Chia Sẻ Trên Internet .....	11
<b>CHƯƠNG II TÌM HIỂU XCODE VÀ IOS SIMULATOR .....</b>	<b>12</b>
<b>2.1 TÌM HIỂU XCODE 5 .....</b>	<b>13</b>
2.1.1 Giới Thiệu Về Xcode 5 .....	13
2.1.2 Thao Tác Tạo Ứng Dụng Mới .....	16
2.1.3 Tìm Hiểu Giao Diện Xcode Và Một Số Tính Năng .....	20
2.1.4 Thiết Kế Giao Diện .....	26
2.1.5 Viết Code .....	29
2.1.6 Thực Thi Và Kiểm Tra Lỗi Của Ứng Dụng.....	34
<b>2.2 TÌM HIỂU IOS SIMULATOR .....</b>	<b>38</b>
2.2.1 Giới Thiệu iOS Simulator .....	38
2.2.2 Tìm Hiểu iOS Simulator .....	38
<b>CHƯƠNG III NGÔN NGỮ OBJECTIVE-C .....</b>	<b>48</b>
<b>3.1 GIỚI THIỆU NGÔN NGỮ OBJECTIVED-C .....</b>	<b>49</b>
<b>3.2 KHAI BÁO BIẾN - CÁCH SỬ DỤNG.....</b>	<b>49</b>
3.2.1 Biến.....	49
3.2.2 Quy Tắc Đặt Tên.....	50
<b>3.3 KIỂU DỮ LIỆU .....</b>	<b>50</b>

3.4 PHÉP TOÁN .....	51
3.5 CHÚ THÍCH CODE .....	51
3.6 XUẤT DỮ LIỆU RA MÀN HÌNH .....	52
3.7 FUNCTION .....	54
3.7.1 Định Nghĩa .....	54
3.7.2 Phương Thức Không Có Tham Số Truyền Vào .....	54
3.7.3 Phương Thức Có 1 Tham Số Truyền Vào .....	55
3.7.4 Phương Thức Có Nhiều Tham Số Truyền Vào .....	55
3.8 CẤU TRÚC ĐIỀU KIỆN .....	57
3.8.1 Câu Lệnh If .....	57
3.8.2 Câu Lệnh If – Else .....	57
3.8.3 Câu Lệnh Switch - Case .....	58
3.9 Cấu Trúc Lặp .....	59
3.9.1 Vòng Lặp For .....	59
3.9.2 Vòng Lặp While .....	59
3.9.3 Vòng Lặp Do-While .....	60
3.10 MẢNG .....	61
3.10.1 Định Nghĩa .....	61
3.10.2 Mảng Nsarray .....	61
3.11 CHUỖI – ĐỐI TƯỢNG NSSTRING .....	62
3.11.1 Khởi Tạo Chuỗi .....	62
3.11.2 Đối Tượng NSString .....	63
3.11.3 Tìm Kiếm Bên Trong Chuỗi .....	63
3.11.4 Tìm Chuỗi Và Thay Nó Thành Chuỗi Khác .....	64
3.11.5 Xoá Nội Dung Bên Trong Chuỗi .....	64
3.11.6 Cắt Chuỗi .....	65
3.11.7 Chèn Ký Tự Vào Trong Chuỗi .....	66
3.11.8 Chèn Ký Tự Vào Cuối Chuỗi .....	66
3.11.8 So Sánh Chuỗi .....	67
3.11.9 So Sánh Chuỗi Với Ký Tự Đầu Và Cuối Chuỗi .....	67

3.11.10 Chuyển Đổi Hình Dạng Của Chữ.....	68
3.11.11 Chuyển Chuỗi Thành Dạng Số.....	70
CHƯƠNG IV MỘT SỐ THAO TÁC CƠ BẢN.....	72
4.1 APP ICON – LOADING SCREEN .....	73
4.1.1 App Icon.....	73
4.1.2 Loading Screen.....	75
4.2 THAY ĐỔI APP NAME .....	76
4.3 ẨN STATUS BAR .....	78
4.4 BACKGROUND.....	79
4.4.1 Background Image.....	79
4.4.2 Background Color .....	83
4.5 THÊM FRAMEWORK .....	84
CHƯƠNG V MỘT SỐ ĐỔI TUỢNG CƠ BẢN.....	86
5.1 ĐỔI TUỢNG LABEL – BUTTON – TEXT FIELD .....	87
5.1.1 Giới Thiệu.....	87
5.1.2 Ví Dụ .....	90
5.2 KẾT NỐI CƠ SỞ DỮ LIỆU VỚI SQLITE .....	94
5.2.1 Giới Thiệu.....	94
5.2.2 Cài Đặt Sqlite Manager Cho Firefox .....	94
5.2.3 Cấu Hình Ứng Dụng Để Tương Tác Với Sqlite .....	95
5.2.4 Các Hàm Trong Sqlite.....	96
5.2.5 Ví Dụ .....	99
5.3 SỬ DỤNG CAMERA IPHONE .....	109
5.3.1 Giới Thiệu.....	109
5.3.2 Ví Dụ .....	109
5.4 UIIMAGE.....	113
5.4.1 Giới Thiệu.....	113
5.4.2 Các Định Dạng Ảnh Hỗ Trợ Trên Iphone .....	114
5.4.3 Ví Dụ .....	114
5.5 UIALERT VIEW.....	121

<b>5.5.1 Giới Thiệu.....</b>	<b>121</b>
<b>5.5.2 Đặc Điểm.....</b>	<b>121</b>
<b>5.5.3 Ví Dụ .....</b>	<b>122</b>
<b>5.6 UISLIDER .....</b>	<b>124</b>
<b>5.6.1 Giới Thiệu.....</b>	<b>124</b>
<b>5.6.2 Đặc Điểm.....</b>	<b>125</b>
<b>5.6.3 Ví Dụ .....</b>	<b>125</b>
<b>5.7 UIWEBVIEW .....</b>	<b>128</b>
<b>5.7.1 Giới Thiệu.....</b>	<b>128</b>
<b>5.7.2 Ví Dụ .....</b>	<b>129</b>
<b>5.8 ACTIVITY INDICATOR VIEW.....</b>	<b>132</b>
<b>5.8.1 Giới Thiệu.....</b>	<b>132</b>
<b>5.8.2 Ví Dụ .....</b>	<b>132</b>
<b>5.9 ACTIONSCHEET .....</b>	<b>136</b>
<b>5.9.1 Giới Thiệu.....</b>	<b>136</b>
<b>5.9.2 Đặc Điểm.....</b>	<b>136</b>
<b>5.9.3 Ví Dụ .....</b>	<b>137</b>
<b>5.10 MK MAP VIEW .....</b>	<b>138</b>
<b>5.10.1 Giới Thiệu.....</b>	<b>138</b>
<b>5.10.2 Ví Dụ .....</b>	<b>139</b>
<b>5.11 TABLE VIEW CONTROLLER.....</b>	<b>143</b>
<b>5.11.1 Giới Thiệu.....</b>	<b>143</b>
<b>5.11.2 Ví Dụ .....</b>	<b>143</b>
<b>5.12 SEARCH BAR .....</b>	<b>147</b>
<b>5.12.1 Giới Thiệu.....</b>	<b>147</b>
<b>5.12.2 Ví Dụ .....</b>	<b>148</b>
<b>5.13 TRUYỀN DỮ LIỆU GIỮA CÁC VIEW.....</b>	<b>152</b>
<b>5.13.1 Giới Thiệu.....</b>	<b>152</b>
<b>5.13.2 Ví Dụ .....</b>	<b>152</b>
<b>CHƯƠNG VI HƯỚNG DẪN XÂY DỰNG PHẦN MỀM.....</b>	<b>157</b>

<b>6.1 PHẦN MỀM KIỂM TRA MÃ VIN .....</b>	<b>158</b>
<b>6.1.1 Giới Thiệu .....</b>	<b>158</b>
<b>6.1.2 Chuẩn Bị .....</b>	<b>158</b>
<b>6.1.3 Cấu Trúc Phần Mềm .....</b>	<b>158</b>
<b>6.1.4 Cơ Chế Vận Hành.....</b>	<b>159</b>
<b>6.1.5 Tính Năng .....</b>	<b>160</b>
<b>6.1.6 Tiến Hành .....</b>	<b>163</b>
<b>6.2 PHẦN MỀM TÌM KIẾM ĐỊA ĐIỂM XUNG QUANH (PLACESNEARME) ..</b>	<b>220</b>
<b>6.2.1 Giới Thiệu .....</b>	<b>220</b>
<b>6.2.2 Chuẩn Bị .....</b>	<b>220</b>
<b>6.2.3 Cấu Trúc Phần Mềm .....</b>	<b>220</b>
<b>6.2.4 Cơ Chế Vận Hành Của Placesnearme .....</b>	<b>221</b>
<b>6.2.5 Tính Năng .....</b>	<b>222</b>
<b>6.2.6 Tiến Hành .....</b>	<b>226</b>
<b>CHƯƠNG VII ĐUÀ ỦNG DỤNG LÊN IPHONE .....</b>	<b>385</b>
<b>7.1 GIỚI THIỆU .....</b>	<b>386</b>
<b>7.2 QUÁ TRÌNH CHUẨN BỊ .....</b>	<b>386</b>
<b>7.3 TIẾN HÀNH .....</b>	<b>388</b>
<b>CHƯƠNG VIII MỘT SỐ VẤN ĐỀ KHÁC .....</b>	<b>396</b>
<b>8.1 XÂY DỰNG ỦNG DỤNG CHO IOS 6 - IOS 6.1 TRÊN XCODE 5 .....</b>	<b>397</b>
<b>8.2 XÂY DỰNG ỦNG DỤNG HỖ TRỢ NHIỀU VERSION IOS .....</b>	<b>403</b>
<b>CÂU HỎI THƯỜNG GẶP .....</b>	<b>407</b>
<b>PHỤ LỤC .....</b>	<b>411</b>

## LỜI MỞ ĐẦU

Ngày nay xu hướng sử dụng Smartphone và máy tính bảng đang gia tăng nhanh chóng trên thế giới nói chung và Việt Nam nói riêng, trong đó Việt Nam hiện đang đứng thứ hai thế giới về tốc độ tăng trưởng smartphone & máy tính bảng với tốc độ tăng trưởng 266%. Android, iOS, Windows Phone là những hệ điều hành chạy trên Smartphone và máy tính bảng phổ biến nhất thế giới: Android 75%, iOS 17,3%, Windows Phone 3,2%. Tại Việt Nam, theo nghiên cứu của IDC, vào thời điểm quý 2/2013, iOS đang chiếm tỉ lệ 1.6% trên tổng số thiết bị phân phối tại Việt Nam, đứng thứ ba sau Android và Windows Phone.

Cùng với sự tăng trưởng của Smartphone và các hệ điều hành chạy trên Smartphone, số lượng ứng dụng cho các hệ điều hành ngày càng tăng, tính cho đến hết năm 2012, số lượng ứng dụng iOS trên Apple App Store đã hơn 775.000 ứng dụng và Google Play đã có hơn 700.000 ứng dụng. Với sự phát triển quy mô lớn của ứng dụng, nhu cầu tìm hiểu về lập trình ứng dụng cho các hệ điều hành cũng tăng dần.

Tuy nhiên, thực tiễn cho thấy, việc tìm hiểu cũng như tham gia các lớp học về lập trình ứng dụng iPhone ở Việt Nam còn nhiều hạn chế và khó khăn. Các lớp dạy lập trình ứng dụng iPhone chỉ mới xuất hiện nhiều trong thời gian gần đây, do đó số lượng vẫn còn hạn chế.

Bên cạnh đó nguồn tài liệu tiếng Việt còn ít, việc tìm hiểu và sử dụng công cụ lập trình cũng như tham khảo tài liệu tiếng Anh về lập trình ứng dụng iPhone đòi hỏi người tìm hiểu phải tiêu tốn một khoảng thời gian dài cũng như có một ít hiểu biết về lập trình và khả năng đọc hiểu tiếng Anh tốt. Hơn nữa các tài liệu tiếng Việt do các trung tâm giảng dạy lập trình iPhone biên soạn chỉ lưu hành nội bộ, người tìm hiểu buộc phải chi một khoản tiền để tham dự lớp học mới có thể có được những tài liệu này.

Với mong muốn tìm hiểu cách xây dựng ứng dụng iPhone để có thêm kiến thức mới, giúp ích cho quá trình làm việc sau khi ra trường cũng như giảm bớt những khó khăn cho người mới bắt đầu tìm hiểu về lập trình ứng dụng trên iPhone, nhóm nghiên cứu đã thực

hiện nghiên cứu, xây dựng một số ứng dụng trên iPhone dựa trên kiến thức tìm hiểu được, từ đó tổng hợp và xây dựng thành tài liệu Hướng dẫn xây dựng ứng dụng trên iPhone. Với những ví dụ riêng cho từng đối tượng, người đọc sẽ dễ dàng nắm bắt và hiểu rõ cách sử dụng, chức năng của từng đối tượng khác nhau. Bên cạnh đó tài liệu còn kèm theo hướng dẫn chi tiết từng bước để xây dựng một vài ứng dụng thực tế mà nhóm nghiên cứu đã thực hiện được trong quá trình nghiên cứu. Hi vọng rằng nội dung của tài liệu này sẽ giúp ích phần nào cho mọi người khi bắt đầu tìm hiểu về lập trình iPhone, từ đó có thể tiết kiệm bớt thời gian cho quá trình tìm hiểu.

Mọi ý kiến đóng góp xin liên lạc qua email [caothienmokimlong@gmail.com](mailto:caothienmokimlong@gmail.com) hoặc [anhtiep20@gmail.com](mailto:anhtiep20@gmail.com). Rất mong nhận được sự góp ý chân thành từ mọi người để tài liệu hướng dẫn ngày càng hoàn thiện hơn.

**Nhóm nghiên cứu**

**Nguyễn Anh Tiệp – Cao Thanh Vàng**

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## **CHƯƠNG I**

### **CHUẨN BỊ TRƯỚC KHI BẮT ĐẦU XÂY DỰNG ỨNG DỤNG**

Quá trình chuẩn bị trước khi bắt đầu lập trình ứng dụng trên iPhone là quá trình cơ bản mà bạn phải chuẩn bị cho thật kỹ lưỡng. Quá trình này sẽ chuẩn bị cho bạn các điều kiện cần thiết để có thể thuận lợi bắt đầu tìm hiểu về lập trình ứng dụng trên iPhone. Trong chương này, bạn sẽ được giới thiệu sơ lược về quá trình chuẩn bị, các cách thức để bạn có được hệ điều hành Mac OS và bộ công cụ Xcode. Hai điều kiện này là điều kiện cần thiết để bắt đầu lập trình iOS.

Tuy nhiên nội dung phần chuẩn bị hệ điều hành Mac OS chỉ được giới thiệu sơ lược với bạn các cách thức để có được một hệ điều hành Mac OS ổn định (cách thức cài đặt Mac OS trên thiết bị Intel/AMD) cho việc lập trình chứ không đi sâu vào hướng dẫn cụ thể bởi điều đó thuộc về một lĩnh vực kiến thức khác đòi hỏi phải đào sâu tìm hiểu.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## 1.1 CHUẨN BỊ HỆ ĐIỀU HÀNH MAC OS

Việc lập trình ứng dụng cho iPhone đòi hỏi bạn phải sử dụng bộ công cụ do chính Apple cung cấp gọi là Xcode, và bộ công cụ này chạy trên nền tảng của hệ điều hành Mac OS. Nếu bạn muốn bắt đầu cho việc xây dựng ứng dụng cho iPhone thì bạn nên bắt đầu ngay với việc chuẩn bị cho mình một chiếc máy chạy hệ điều hành Mac OS. Sau đây chúng tôi sẽ giới thiệu cho bạn một số cách thức để bạn có thể sở hữu cho mình một hệ điều hành Mac OS phù hợp.

### 1.1.1 Lập Trình Ứng Dụng Iphone Trên Windows

Nếu bạn muốn lập trình ứng dụng cho iPhone, nhưng lại muốn thực hiện trên môi trường Windows để có thể thao tác thêm các công việc khác trên môi trường này, lựa chọn tốt nhất cho bạn lúc này chính là sử dụng một máy ảo chạy hệ điều hành Mac OS để lập trình. Tuy nhiên nếu bạn lựa chọn phương án này, cấu hình máy của bạn phải mạnh, CPU xử lý tốt, và bạn nên dành ít nhất 2G RAM cho máy ảo hoạt động tốt hơn.

#### *Làm cách nào để tôi có thể sở hữu một máy ảo chạy Mac OS cho riêng mình?*

Bạn có hai cách để có thể sở hữu một máy ảo chạy Mac OS cho mình.

Cách một là bạn chuẩn bị một đĩa cài Mac OS và phần mềm hỗ trợ tạo máy ảo như VMWare , Virtualbox ... sau đó bạn tiến hành cài đặt bình thường bằng tay. Đĩa cài đặt Mac OS bạn có thể mua ở các tiệm bán đĩa phần mềm. Hoặc bạn có thể tải về trên mạng một đĩa cài Mac OS đuôi .iso ( Thường thì trên mạng có đĩa cài đuôi .dmg, bạn phải chuyển đổi qua đuôi .iso để được hỗ trợ tốt nhất từ phần mềm tạo máy ảo).

Cách thứ hai là bạn tải về một máy ảo hoàn chỉnh đã được cài đặt sẵn Mac OS, sau đó mở lên bằng phần mềm chạy máy ảo là bạn đã có thể có một hệ điều hành Mac OS.

Bạn nên truy cập vào

Diễn đàn Tinh Tế: [www.tinhte.vn/forums/chuyen-de-hackintosh.361/](http://www.tinhte.vn/forums/chuyen-de-hackintosh.361/) .

Cộng đồng Hackintosh: [www.facebook.com/groups/hackintoshvietnam/](http://www.facebook.com/groups/hackintoshvietnam/).

dễ tìm hiểu thêm tiến trình cài đặt máy ảo, các vấn đề xảy ra trong quá trình cài.

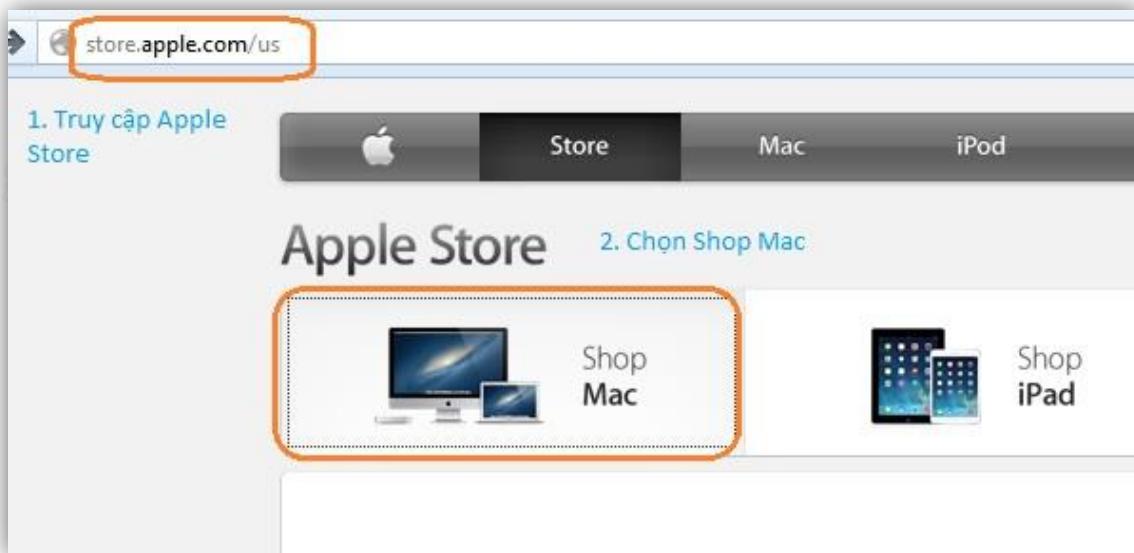
Ngoài ra bạn cũng có thể tìm các máy ảo chạy Mac OS cài đặt sẵn tại

Soul Dev Team: <http://www.souldevteam.net/> .

### 1.1.2 Sử Dụng Sản Phẩm Chính Hãng Apple

Nếu gia đình bạn có điều kiện, bạn có thể mua ngay cho mình một chiếc máy Apple chính hãng, sử dụng trọn vẹn tính năng cũng như sự hỗ trợ tối đa từ Apple.

Bạn truy cập vào [www.apple.com](http://store.apple.com/us) , chọn Store, sau đó lựa chọn Shop Mac để đến chuyên mục bán các sản phẩm chạy Mac OS của Apple. Tại đây bạn có thể lựa chọn nhiều loại sản phẩm khác nhau như Macbook Air, Macbook Pro, iMac, Mac mini....

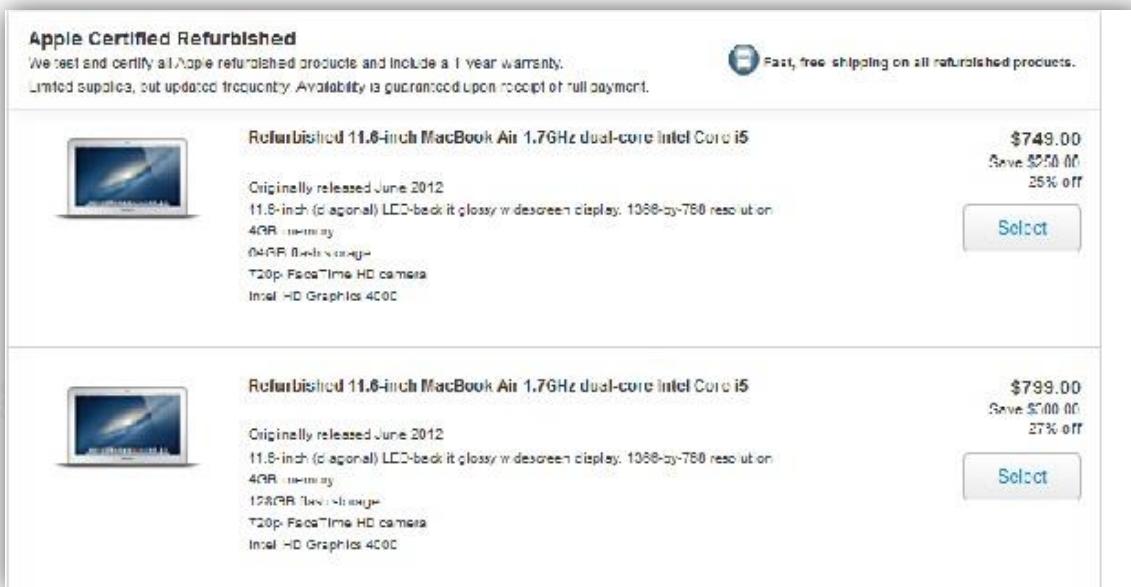


**Hình 1.1 Truy cập Store Mac OS**

Tùy theo sở thích và nhu cầu, cũng như khả năng tài chính mà bạn lựa chọn cho mình một chiếc máy thích hợp. Bạn có thể thanh toán cho Apple và đợi hàng được chuyển về, hoặc tìm đến các trung tâm bán hàng của Apple ở gần nhà để mua. Hoặc bạn cũng có thể truy cập vào địa chỉ [www.apple.com/asia/reseller/](http://www.apple.com/asia/reseller/) để tìm kiếm các địa điểm bán hàng của Apple gần nhất. Ở Việt Nam, bạn hãy truy cập vào trang

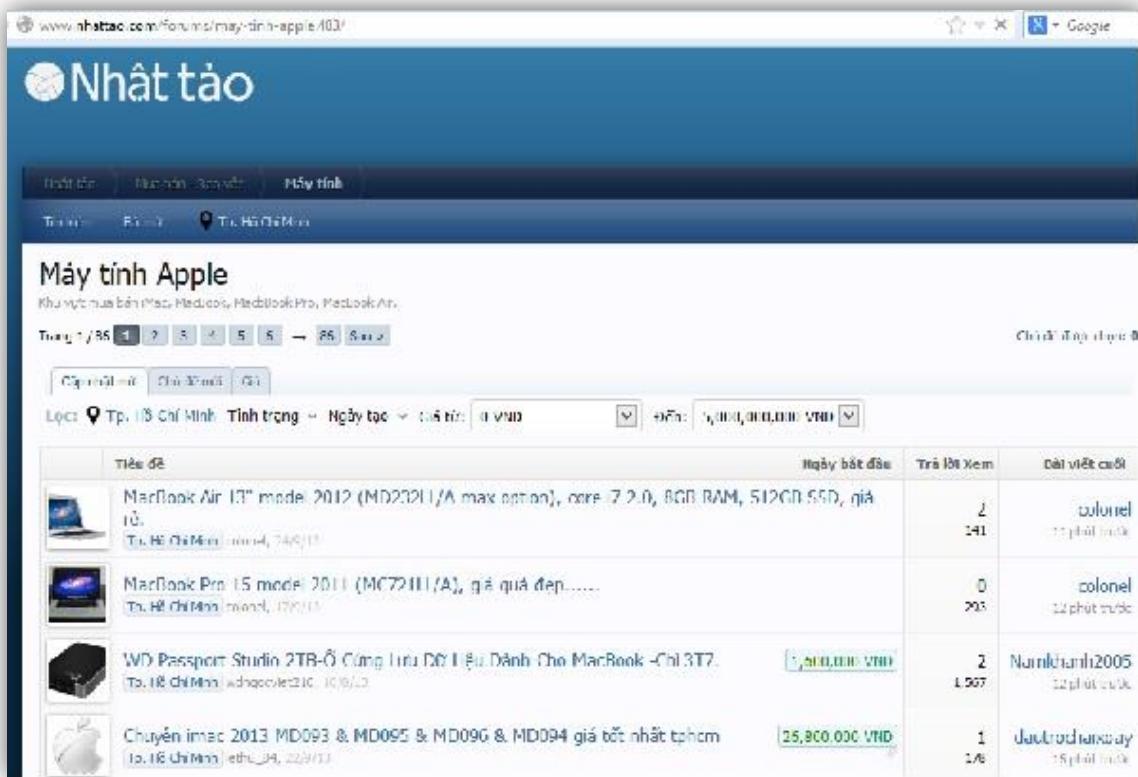
[www.icenter.com.vn](http://www.icenter.com.vn) để có thể nhanh chóng tìm được cửa hàng bán sản phẩm của Apple gần nhất nơi bạn sống. Việc này giúp bạn có thể nhanh chóng có trong tay một sản phẩm chính hãng Apple, thanh toán tiện lợi mà không phải tốn thời gian cho việc chuyển khoản thanh toán cũng như chờ đợi vận chuyển sản phẩm từ nước ngoài về.

Nếu muốn tiết kiệm thêm chi phí bạn có thể truy cập [www.store.apple.com/us/browse/home/specialdeals/mac](http://www.store.apple.com/us/browse/home/specialdeals/mac) để tìm mua các giảm phẩm Refurbished của Apple với mức giá thấp hơn.



## Hình 1.2 Store Refurbished

Nếu bạn muốn mua máy chính hãng Apple với giá rẻ hơn nữa, thì việc mua lại sản phẩm đã qua sử dụng là một gợi ý đáng cho bạn cân nhắc. Đối với việc mua lại máy đã qua sử dụng bạn có thể vào một số trang mua bán có chất lượng để tìm mua sản phẩm phù hợp với túi tiền. Chẳng hạn như các trang [www.5giay.vn](http://www.5giay.vn), [www.nhattao.com](http://www.nhattao.com) ..... là các trang với rất nhiều sự chọn, các dòng máy đa dạng cũng như mức tiền khác nhau, tùy theo nhu cầu và điều kiện cho phép mà bạn có thể tìm được một chiếc máy đúng ý mình.



Nguyễn Anh Tiệp Cao Thành Vàng © 2013

### Hình 1.3 Mua lại máy cũ

Nếu bạn lựa chọn mua máy cũ, phải lựa chọn thật kỹ lưỡng từ đời máy đến cấu hình, tốt nhất nên chọn mua những máy sản xuất trong 1-2 năm gần nhất, đặc biệt bạn phải kiểm tra xem máy mình muốn mua có hỗ trợ những phiên bản Mac OS nào. Vì sao phải như vậy ? Bởi vì Xcode 5 chạy trên Mac OS 8 trở lên, nếu bạn chọn mua máy chỉ hỗ trợ Mac OS 7 trở về trước, thì bạn không thể cài được Xcode 5 mà chỉ có thể sử dụng các phiên bản thấp hơn.

### 1.1.3 Chạy Hệ Điều Hành Mac Os Trên Pc/Laptop Intel/Amd – Tại Sao Không?

Việc cài Mac OS lên chiếc PC hay laptop của bạn cũng là một phương pháp tốt để vừa có được hệ điều hành Mac OS, vừa có thêm kinh nghiệm trong quá trình cài đặt, và hơn nữa là tiết kiệm được chi phí. Có hai dạng cài đặt hackintosh là cài đặt từ một bản đã được chỉnh sửa sẵn như iAtkos (hỗ trợ dòng máy intel) hay Niresh (hỗ trợ thêm dòng máy AMD) và cài đặt từ đĩa gốc của Apple (Mac OS Retail). Dạng cài đặt từ đĩa gốc của Apple đòi hỏi bạn sau khi cài đặt phải tiến hành thêm nhiều thao tác khác để có

được một hệ điều hành Mac OS hoàn thiện, còn đối với bản đã chỉnh sửa sẵn thì gần như không điều chỉnh thêm nhiều.



**Hình 1.4 iATKOS và Niresh**

Bạn có hai phương án để lựa chọn nếu muốn chạy Mac OS lên máy của mình: tự cài thủ công và thuê người cài đặt. Nếu bạn thuê người cài đặt, chi phí thường khoảng từ 100.000 vnđ đến 200.000 vnđ tùy theo yêu cầu cài đặt như thế nào.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

The screenshot shows a search results page from a search engine. The search query "cài đặt mac os tại nhà" is entered in the search bar. Below the search bar, there are filters for "K/V Research", "AVV Sandbox", "Traffic Estimator", "Trends", and "CSV". The search results are categorized by "Web", "Hình ảnh", "Video", "Thảo luận", "Thêm", and "Các công cụ tìm kiếm". The results section displays approximately 502,000 results found in 0.34 seconds. The first result is a link titled "Giới thiệu dịch vụ cài macbook tại nhà giá rẻ-chất lượng HCM ...". Below the link, it says "www.tinhte.vn > ... > Tin tức - Giới thiệu - Đánh giá". The snippet of the page content includes: "14-08-2013 - Tư vấn miễn phí ALO 0909.167.388 (24/24) Cài đặt Macbook tại nhà (tận nơi) Ré nhá: TpHCM Cài Mac Os tại nhà | Cài win cho Mac tại nhà ...". Other snippets mention "Sử dụng Uniphox toàn tập Windows", "lường dẫn cài đặt hệ điều hành MAC trên PC toàn tập.", "có ai nhận cài mac OS X tại nhà không ạ", and "Các kết quả khác từ www.tinhte.vn".

**Hình 1.5 Tìm kiếm “Cài đặt Mac OS cho PC/Laptop”**

Với việc thuê người cài, bạn chỉ gần báo cáo hình máy cho người cài đặt để họ quyết định và cho bạn lời khuyên. Bởi việc cài đặt Hackintosh phụ thuộc rất nhiều vào độ tương thích phần cứng của thiết bị với hệ điều hành Mac OS, do đó không phải máy nào cũng có thể cài được. Tuy nhiên phương án này giúp bạn đỡ vất vả hơn, cũng như tiết kiệm được thời gian hơn.

Một phương án khác đó là bạn tự cài. Với cách lựa chọn này, bạn phải tìm hiểu về Hackintosh, cũng như các lưu ý khi cài đặt, các lỗi xảy ra và cách khắc phục ... Nếu bạn lựa chọn cách này, bạn nên tìm hiểu một số diễn đàn, hội nhóm chuyên về Hackintosh để có thêm kiến thức, kinh nghiệm và sự trợ giúp từ cộng đồng. Ở đây chúng tôi giới thiệu cho bạn hai địa chỉ uy tín về Hackintosh:

- Chuyên Đề Hackintosh trên Tinh Tế ở địa chỉ: [www.tinhte.vn/forums/chuyen-de-hackintosh.361/](http://www.tinhte.vn/forums/chuyen-de-hackintosh.361/)
- Group Hackintosh – We Love Mac ở địa chỉ:  
[www.facebook.com/groups/hackintoshvietnam/](http://www.facebook.com/groups/hackintoshvietnam/)

Tại đây bạn sẽ tìm được nhiều tài liệu hướng dẫn cũng như sự giúp đỡ tận tình của mọi người, hi vọng bạn sẽ có một hệ điều hành Mac OS hoàn chỉnh.

#### Bạn có biết:

- Phần cứng của máy tính rất quan trọng khi quyết định cài Mac OS lên máy tính thông thường vì không phải phần cứng nào cũng có thể tìm được Kext (Có thể hiểu giống như Driver cho thiết bị trên hệ điều hành Windows).
- Card VGA Intel Graphic HD 4000 và Intel Graphic HD 3000 được Mac OS hỗ trợ Kext rất tốt.
- Đa phần dòng HP Probook hỗ trợ tốt Hackintosh.

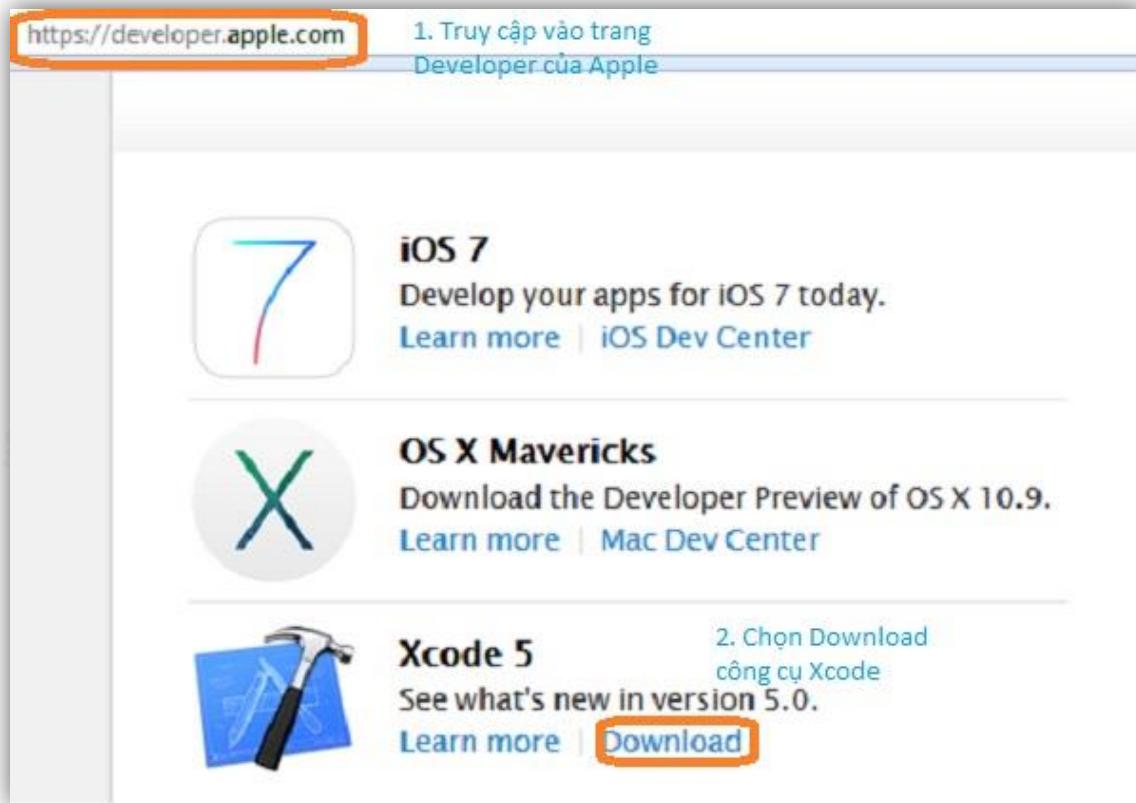
## 1.2 PHẦN MỀM XCODE

Sau khi bạn đã có được hệ điều hành Mac OS, việc tiếp theo bạn phải làm trước khi có thể bắt đầu lập trình ứng dụng iPhone là cài bộ công cụ lập trình Xcode do Apple cung

cấp cho các nhà lập trình ứng dụng, để các nhà lập trình có thể phát triển ứng dụng cho cả iOS lẫn Mac OS. Việc cài đặt Xcode có nhiều cách, tùy theo bạn chọn lựa cách nào phù hợp với bản thân.

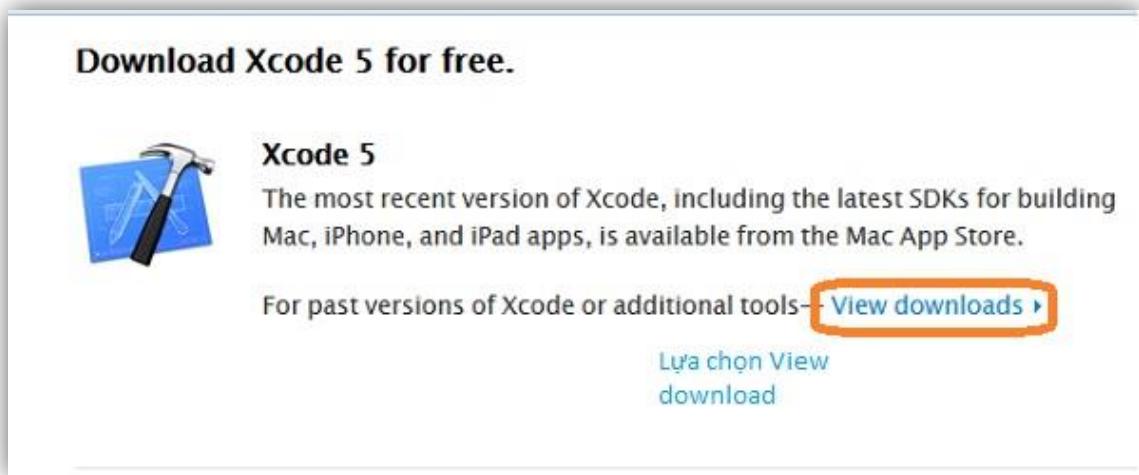
### 1.2.1 Cài Đặt Thông Qua Bản Tải Về Từ Trang Dành Cho Developer

Bạn truy cập vào trang [www.developer.apple.com](https://developer.apple.com) để tiến hành tải phiên bản Xcode mới nhất ( hoặc các phiên bản khác tùy theo nhu cầu của bạn).



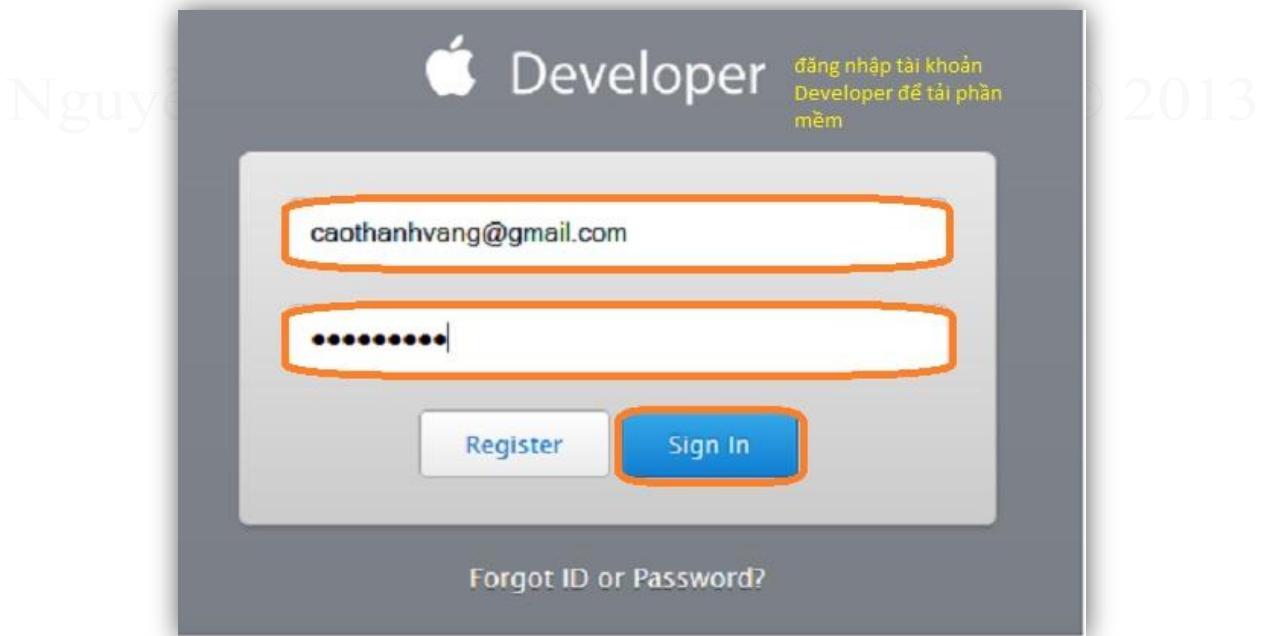
**Hình 1.6 Tải Xcode từ trang Developer**

Tiếp theo bạn lựa chọn **View Download**.



**Hình 1.7 Chọn View Download**

Apple sẽ yêu cầu bạn đăng nhập tài khoản Developer ID để tiếp tục.



**Hình 1.8 Đăng nhập Developer ID**

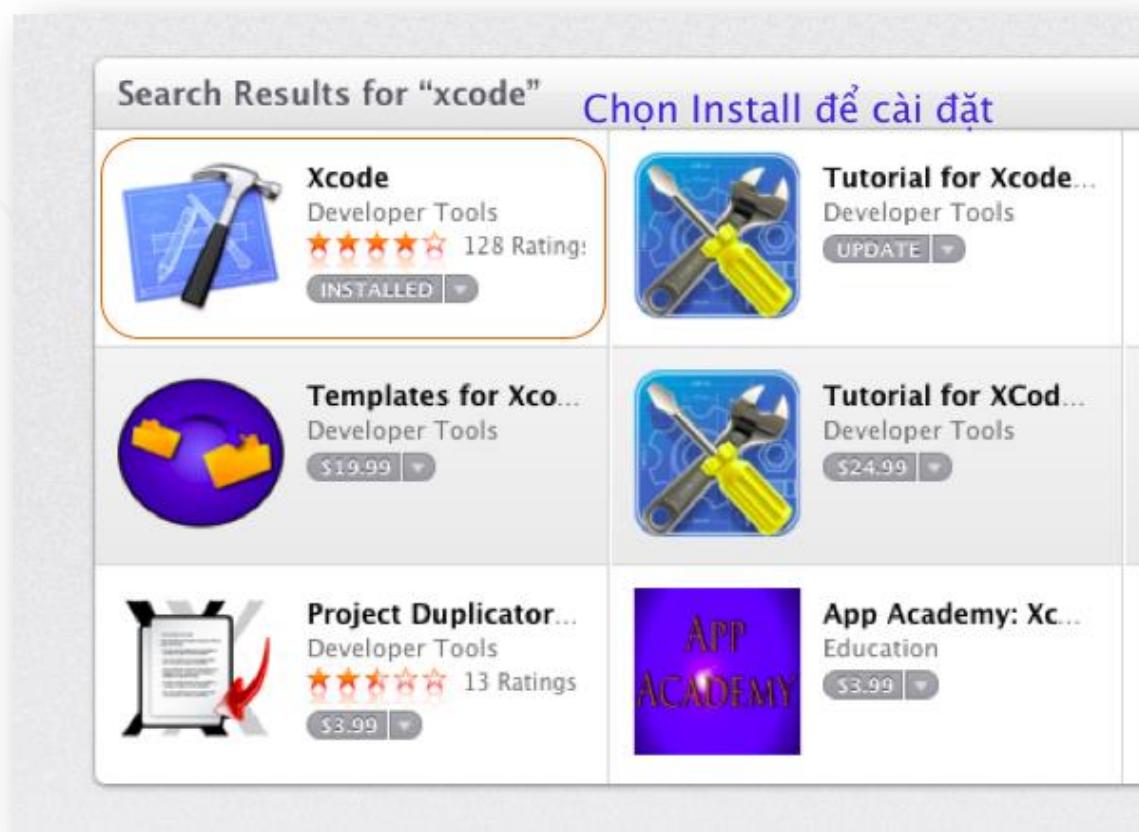
Sau khi bạn đăng nhập bằng tài khoản Developer xong, thực hiện theo hướng dẫn, bạn sẽ tải về được phần mềm Xcode và tiến hành cài đặt trên máy.

Với tài khoản Developer ID với giá 99\$ / năm, bạn sẽ luôn được Apple cập nhật thông tin công nghệ mới, cũng như hỗ trợ, sử dụng trước các phiên bản mới của Apple như các bản preview iOS, Mac OS, Xcode...

### 1.2.2 Cài Đặt Thông Qua Apple Mac Store

Đây là cách phổ biến nhất, vì việc tải Xcode trên Apple Store là miễn phí, và chỉ cần bạn có một tài khoản Apple ID là được, không yêu cầu phải là Developer ID.

Bạn chỉ cần truy cập vào Apple Mac Store và tìm kiếm Xcode, bạn sẽ thấy kết quả là phần mềm Xcode Free, công việc bây giờ là bạn chỉ cần Install và chờ đợi hoàn tất.



Hình 1.9 Cài đặt Xcode qua Apple Mac Store

Trong App Store bạn cũng có thể tìm được nhiều giáo trình, bài giảng về lập trình ứng dụng bằng Xcode.

### **1.2.3 Cài Đặt Từ Bản Xcode Được Chia Sẻ Trên Internet**

Đối với cách cài đặt này, bạn chỉ cần truy cập internet và tìm bản cài đặt Xcode được chia sẻ trên mạng và tải về cài đặt trên máy của bạn. Tuy nhiên bạn sẽ mất thời gian để tìm kiếm trên internet để tìm được bản cài đặt vừa ý, tốc độ tải tốt nhất.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## **CHƯƠNG II**

### **TÌM HIỂU XCODE VÀ IOS SIMULATOR**

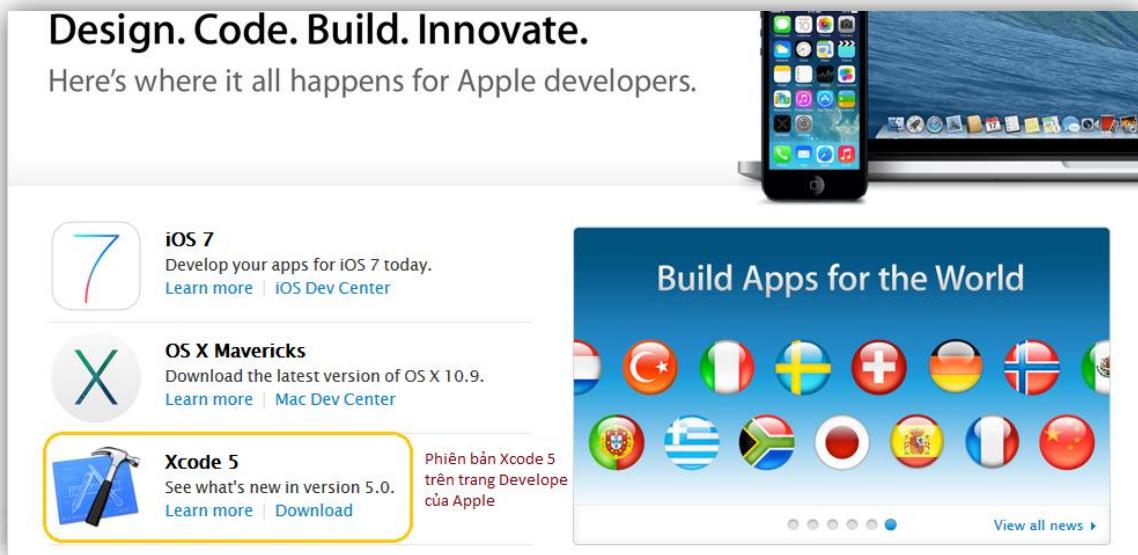
Chương này sẽ mang đến cho bạn kiến thức về bộ công cụ Xcode cũng như công cụ giả lập hệ điều hành iOS là iOS Simulator. Qua những kiến thức được cung cấp, bạn sẽ nắm rõ về giao diện, một số tính năng, các button và công dụng của nó trên Xcode và iOS Simulator. Ngoài ra, bạn sẽ được hướng dẫn một số thao tác cơ bản khi sử dụng Xcode, iOS Simulator từ đó bạn sẽ dễ dàng hơn trong việc sử dụng bộ công cụ này trong quá trình lập trình ứng dụng iPhone về sau.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## 2.1 TÌM HIỂU XCODE 5

### 2.1.1 Giới Thiệu Về Xcode 5

Phần mềm Xcode là bộ công cụ do Apple cung cấp cho các lập trình viên để lập trình ứng dụng cho các thiết bị chạy hệ điều hành của Apple. Phiên bản mới nhất hiện nay của Xcode là bản Xcode 5 trên trang Developer của Apple.



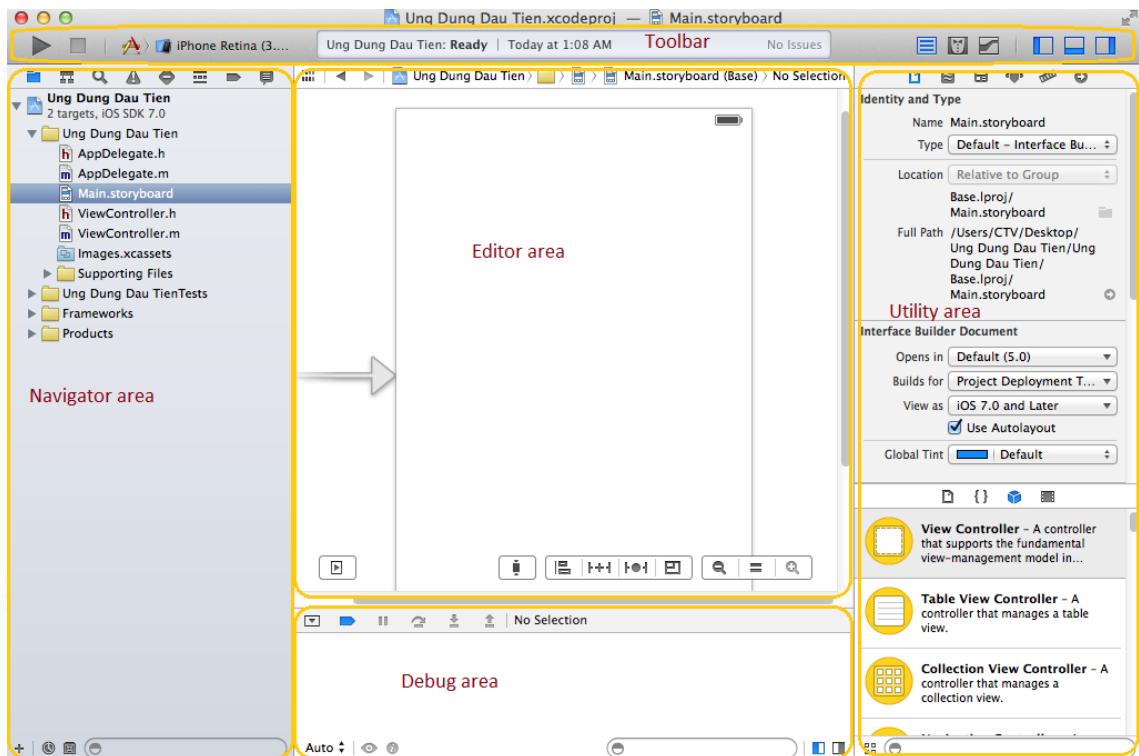
Hình 2.1 Phiên bản Xcode 5 trên trang Apple

Giao diện làm việc của Xcode gồm có 5 phần chính : **Toolbar**, **Editor area**, **Navigator area**, **Debug area**, **Utility area**.

- **Debug area** : đây là vùng hỗ trợ bạn trong quá trình debug lỗi của chương trình.
- **Toolbar area**: vùng chứa các công cụ tiện ích giúp bạn có thể đơn giản trong việc chạy, debug ứng dụng, lựa chọn iOS Simulator, đóng mở các vùng khác...
- **Editor area**: vùng để bạn thiết kế giao diện, viết và chỉnh sửa code của chương trình.
- **Utility area**: vùng này cho phép bạn tùy chỉnh các tham số, giá trị của các đối tượng trên giao diện, cũng như cho phép bạn kéo thả và sử dụng các đối tượng

có sẵn của Xcode như Button, Label, Slider... hay các đoạn code mẫu ( If, Switch...).

- **Navigator area:** cung cấp cho bạn một cách nhìn trực quan, tiện lợi cho việc quản lý ứng dụng, xem thông báo lỗi, tìm kiếm một đoạn code trong chương trình hay kiểm tra mức độ hoạt động của RAM, CPU khi chạy ứng dụng...



**Hình 2.2 Giao diện Xcode**

Xcode cũng cung cấp cho bạn một chế độ gỡ lỗi thông minh hỗ trợ bạn trong việc phát hiện lỗi, cảnh báo lỗi và gợi ý thay thế khắc phục

The screenshot shows the Xcode interface with the file `ViewController.m` open. A yellow box highlights the status bar at the top right, which displays a warning icon with the number '1'. Another yellow box highlights a tooltip in the bottom right corner of the code editor, which reads: 'Nhận diện và thông báo các lỗi, các cảnh báo.'

```

// ViewController.m
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.
//

#import "ViewController.h"

@interface ViewController : UIViewController

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically
    // from a nib.
}

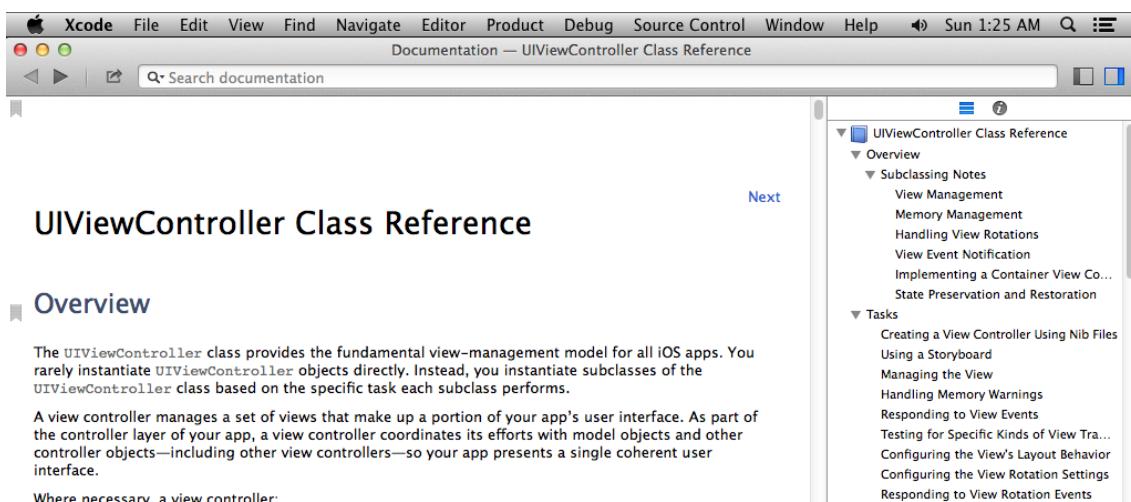
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    int a; // Unused variable 'a'
    // Dispose of any resources that can be recreated.
}

@end

```

**Hình 2.3 Chế độ gỡ lỗi**

Hơn thế nữa, kèm theo Xcode là một bộ tài liệu hướng dẫn từng bước, chi tiết và tiện lợi nhằm hỗ trợ người dùng trong việc lập trình. Trong quá trình viết ứng dụng, nếu bạn muốn tìm hiểu thêm một đối tượng, bạn có thể sử dụng tới bộ tài liệu này để có được hướng dẫn, ví dụ minh họa dễ hiểu.



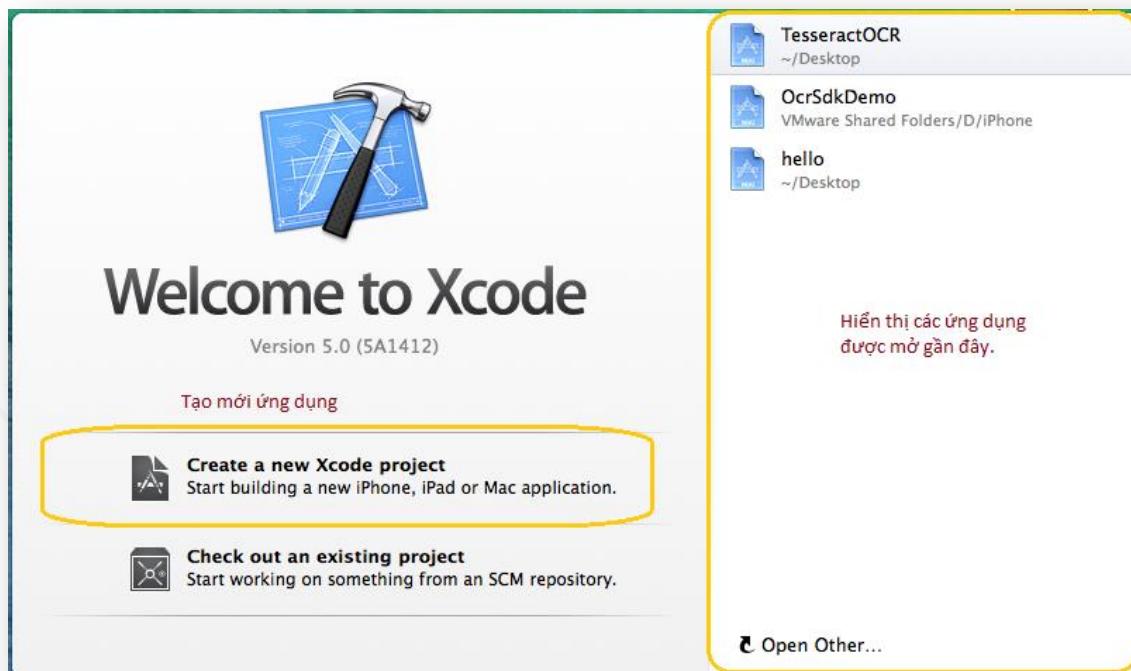
**Hình 2.4 Tài liệu hướng dẫn**

Bạn có thể xem thêm tài liệu về Xcode do Apple cung cấp tại:

[https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode\\_Overview/About\\_Xcode/about.html](https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/About_Xcode/about.html)

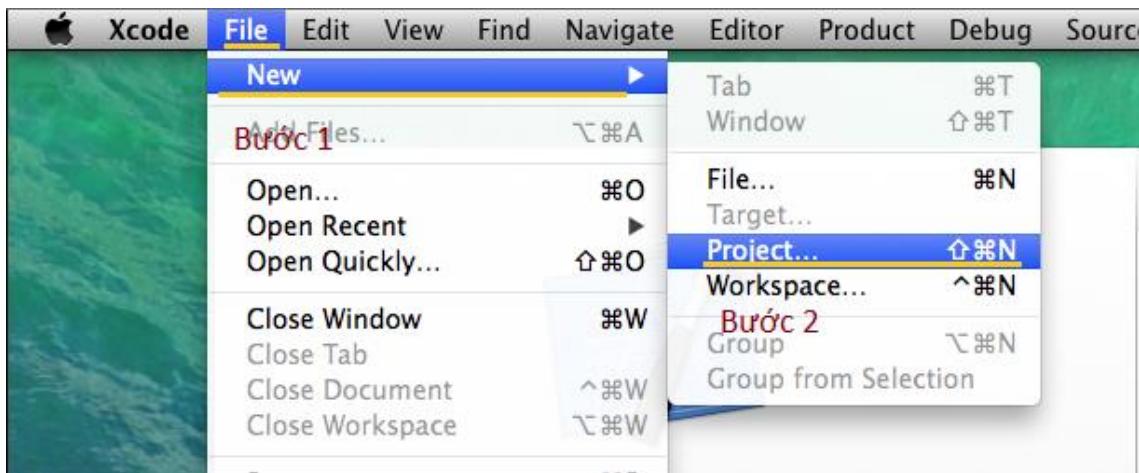
### 2.1.2 Thao Tác Tạo Ứng Dụng Mới

Khi khởi động Xcode lên, giao diện hiện ra cho phép bạn tạo một project mới, hoặc mở lại các project gần đây.



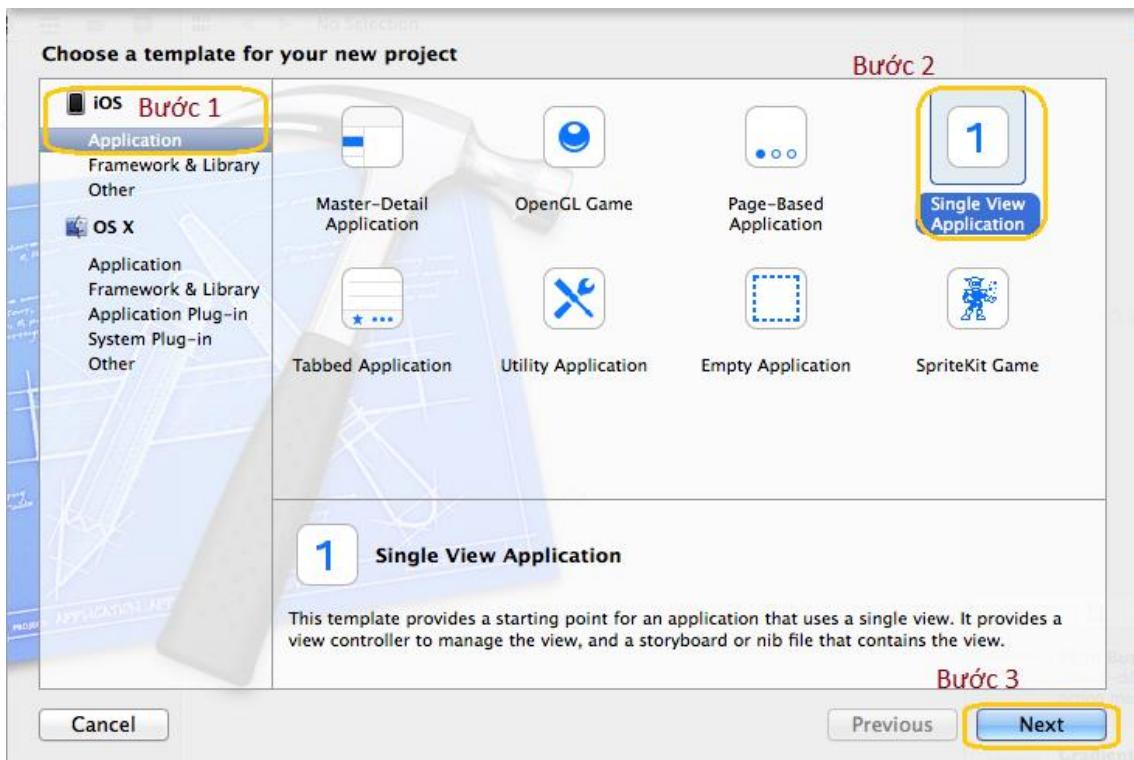
**Hình 2.5 Giao diện Xcode khi mở lên**

Tại giao diện Xcode, bạn có thể tạo mới một project bằng cách chọn **Create new project**. Ngoài ra bạn có thể tạo project mới bằng cách chọn **File > New > Project**.



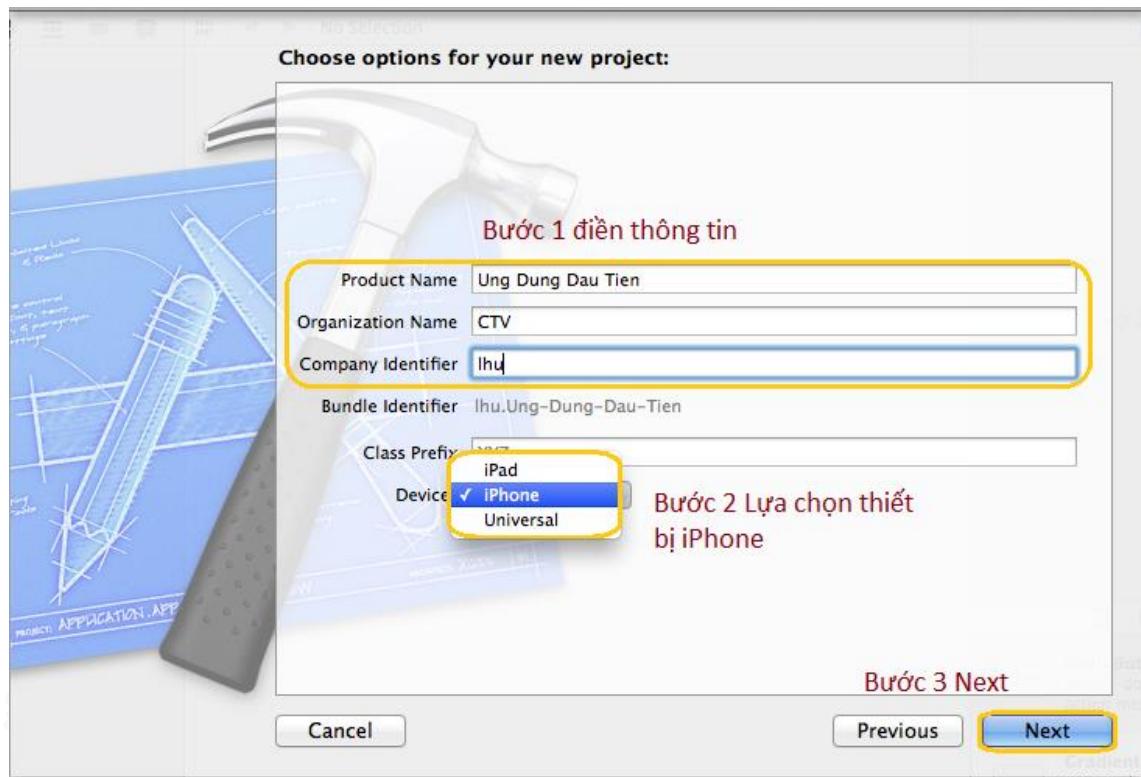
Hình 2.6 Tạo mới Project bằng Menu

Sau khi chọn New Project, Xcode sẽ yêu cầu bạn lựa chọn một hình thức cho Project này (ứng dụng cho iPhone hay Mac OS, Single View hay Empty View...). Cách đơn giản nhất là bạn chọn **Single View**.



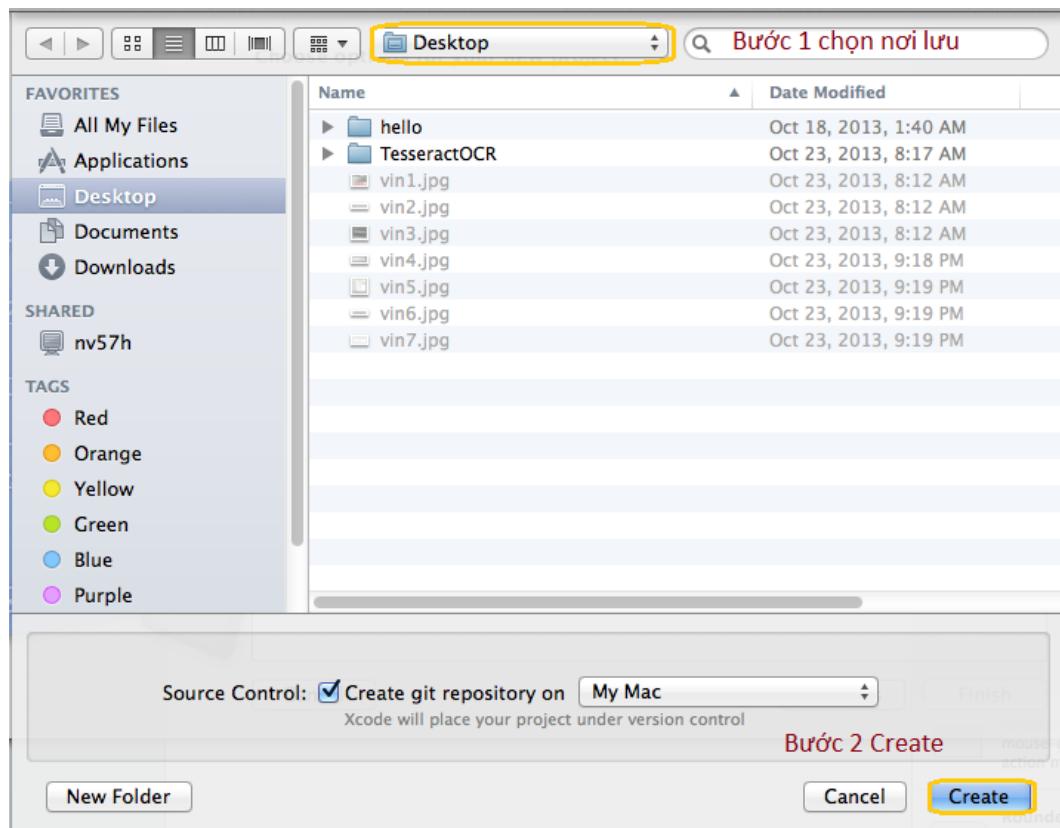
Hình 2.7 Chọn lựa mẫu cho project

Tiếp theo đó bạn điền thêm một vài thuộc tính của Project như **Product Name**, **Organization Name**, **Company Identifier**. Sau đó bạn tiến hành lựa chọn Devices cho Project (iPhone, iPad hay Universal để viết ứng dụng cho cả hai).



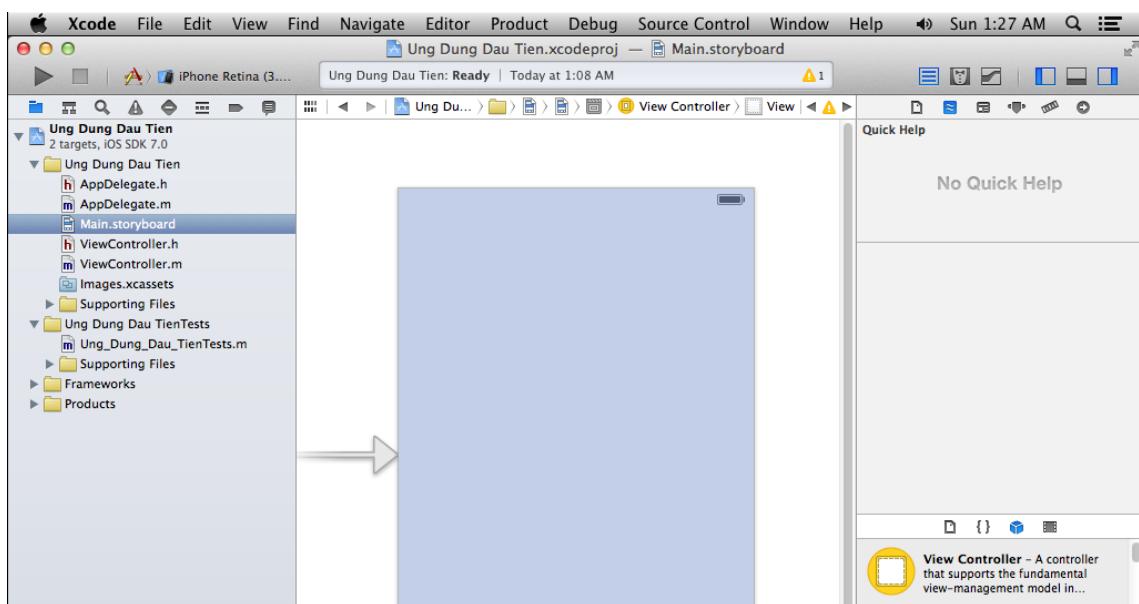
**Hình 2.8** **Điền thông tin cho project**

Tiếp theo bạn chọn nơi lưu trữ Project trên máy tính để lưu Project và chọn **Create**.



Hình 2.9 Chọn nơi lưu Project

Như vậy bạn đã tạo xong một Project mới.

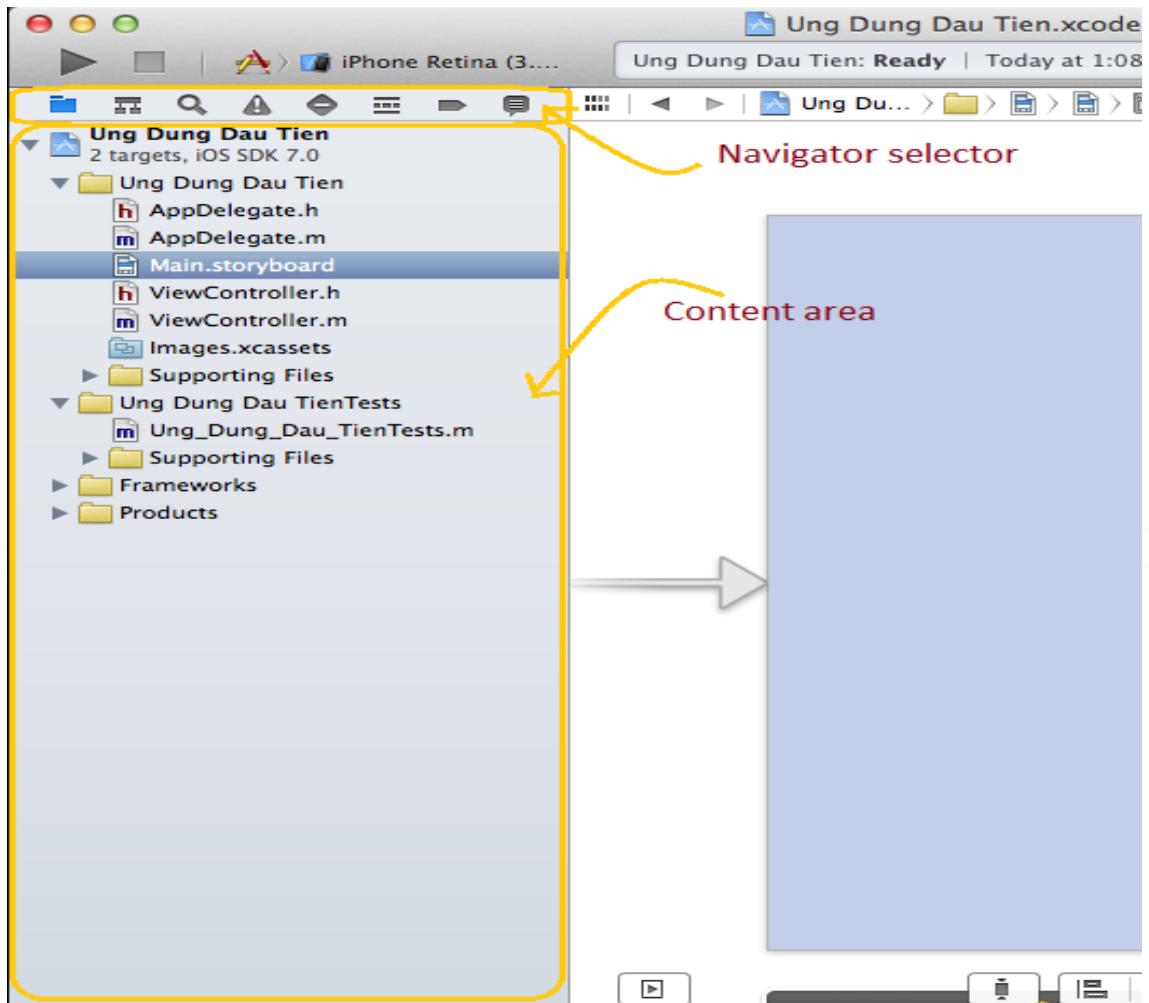


Hình 2.10 Giao diện project mới tạo

## 2.1.3 Tìm Hiểu Giao Diện Xcode Và Một Số Tính Năng

### 2.1.3.1 Navigator Area

**Navigator area** cho phép bạn quản lý ứng dụng hiệu quả như quản lý các tập tin, thư mục, quản lý các thông báo lỗi và cảnh báo, quản lý việc debug... Có thể chia Navigator area thành hai phần chính là **Navigator selector bar** và **Content area**.



Hình 2.11 Giao diện Navigator

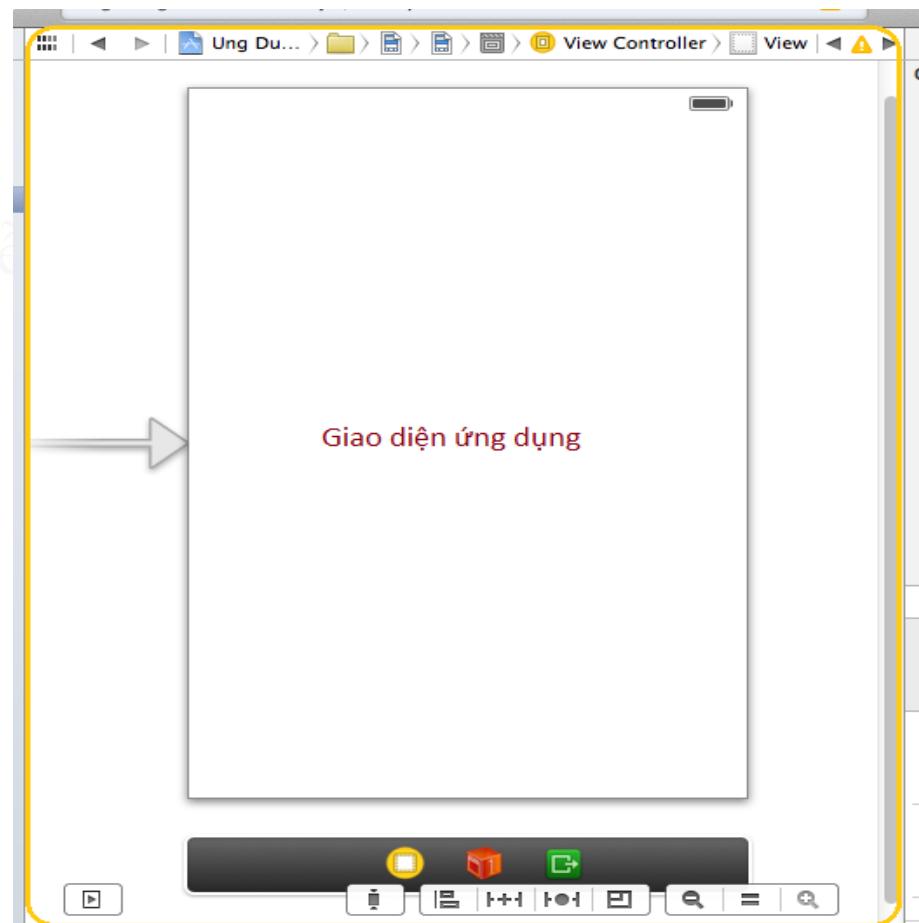
Trong Navigator selector bar gồm một số button chính sau:

- **Project Navigator** (📁): dùng để quản lý các tập tin của ứng dụng như thêm, xóa, gom nhóm... Các tập tin quản lý sẽ được thể hiện trong Content area.

- **Find Navigator** (🔍): sử dụng để tìm kiếm một cách nhanh chóng các string trong ứng dụng, tìm kiếm nội dung mở rộng.
- **Issue Navigator** (⚠️): quản lý các thông báo lỗi, cảnh báo của ứng dụng.
- **Debug Navigator** (⠇): theo dõi quá trình debug ứng dụng.

### 2.1.3.2 Editor Area

**Editor area** cho phép bạn thiết kế giao diện, viết và sửa code cho ứng dụng. Khi bạn chọn tập tin **Storyboard** bên Content area thì Editor area sẽ hiển thị giao diện **Interface Builder** cho bạn thiết kế giao diện. Tương tự với tập tin .m và .h thì Editor area sẽ hiển thị nội dung code của tập tin.



Hình 2.12 Giao diện ứng dụng



```
// ViewController.m
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.
//

#import "ViewController.h"

@interface ViewController : UIViewController

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically
    // from a nib.
}

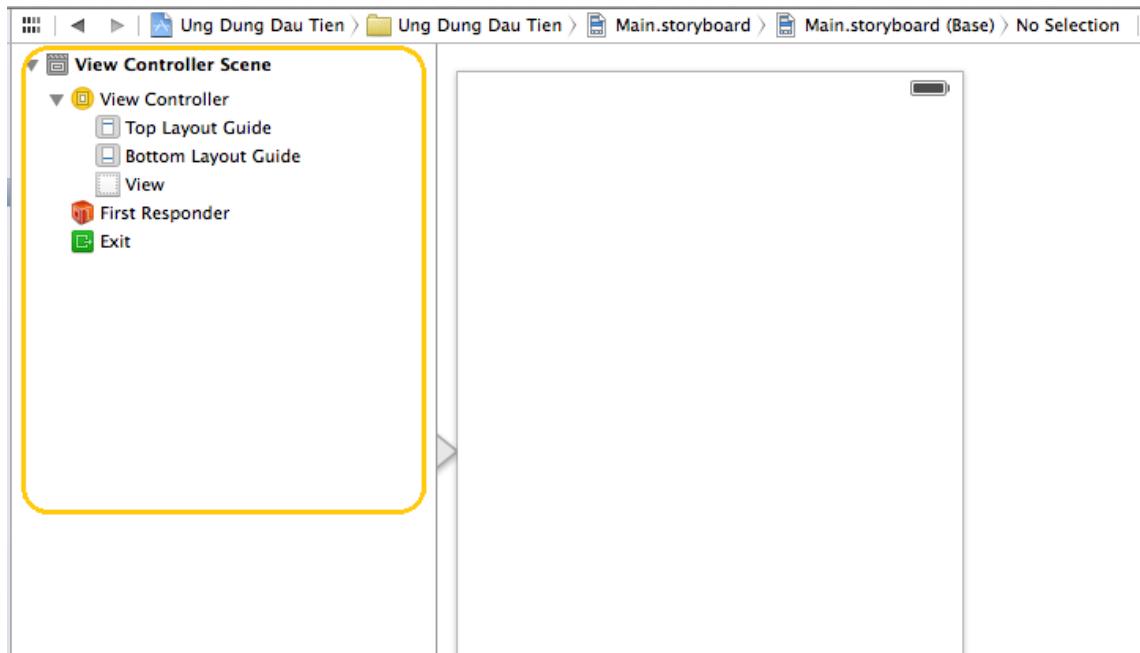
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    int a;
    // Dispose of any resources that can be recreated.
}

@end
```

Giao diện code

**Hình 2.13 Giao diện code**

**Editor area** còn cho phép bạn quản lý các đối tượng trong giao diện một cách chi tiết hơn. Trong phần Interface Builder, bạn chọn button **Show Document Outline** (  ) bên góc trái màn hình, bạn sẽ thấy được một vùng quản lý phân cấp các đối tượng.



**Hình 2.14 Giao diện quản lý chi tiết**

### 2.1.3.3 Toolbar Area

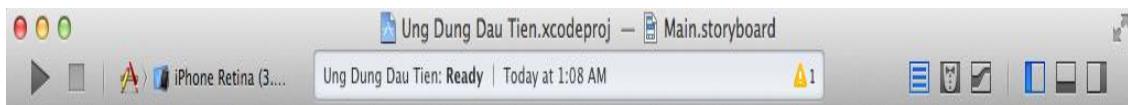
Toolbar cho phép thực hiện một số thao tác một cách nhanh chóng thông qua các Button mà không cần phải dùng tới Menu. Toolbar gồm một số thành phần sau:

- **Run button** (▶): dùng để chạy thử ứng dụng.
- **Stop button** (□): dùng để dừng việc chạy thử ứng dụng.
- **Scheme menu** (A > iPhone Retina (3....)): dùng để lựa chọn iOS Simulator thích hợp để chạy ứng dụng.
- **Activity viewer**: thông báo trạng thái của ứng dụng, cũng như hiện các trạng thái lỗi, cảnh báo của chương trình (nếu có).



**Hình 2.15 Activity viewer**

- **Editor selector** ( ): gồm các button dùng để điều chỉnh Editor area (cho phép chia đôi Editor area ra làm hai hay chỉ là một vùng duy nhất...).
- **View selector** ( ): dùng để ẩn/hiện các vùng Navigator area, Utility area, Debug area.

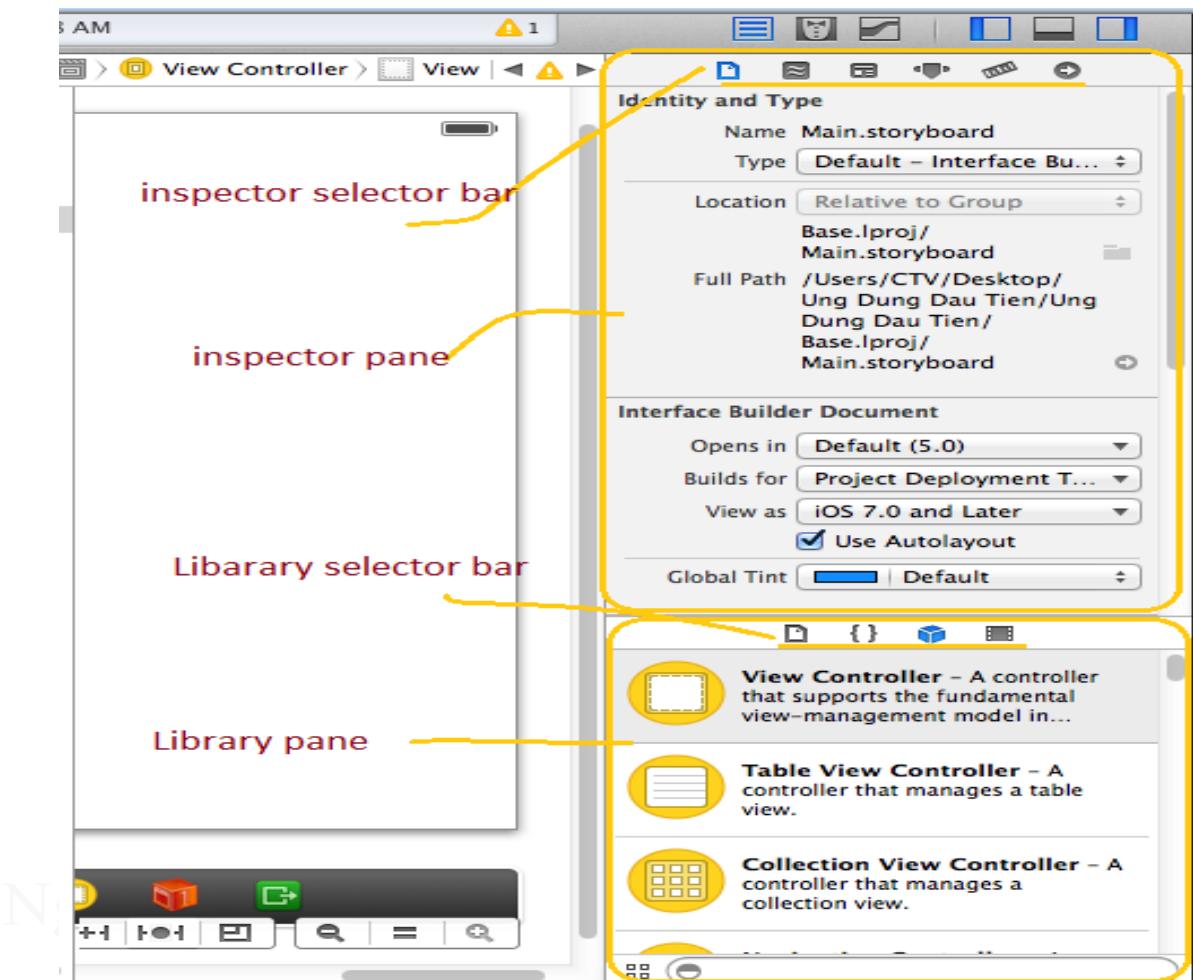


**Hình 2.16 Toolbar area**

#### 2.1.3.4 Utility Area

**Utility area** được sử dụng để thay đổi các thuộc tính của đối tượng bên Interface Builder, ngoài ra còn được sử dụng để lựa chọn và kéo thả các đối tượng, đoạn code mẫu vào Interface Buider và Editor.

Utility area được chia làm 2 vùng chính là **Inspector** và **Library**. **Inspector pane** là vùng cho phép bạn có sự thay đổi thuộc tính của đối tượng. Trên đầu của Inspector pane là **Inspector selector bar** bao gồm các button hỗ trợ bạn trong việc điều chỉnh thuộc tính. Trên đầu của **Libarary pane** là Libaray selector bar bao gồm các button để bạn có thể chọn lựa phù hợp trong việc sử dụng các đoạn code mẫu, các đối tượng.



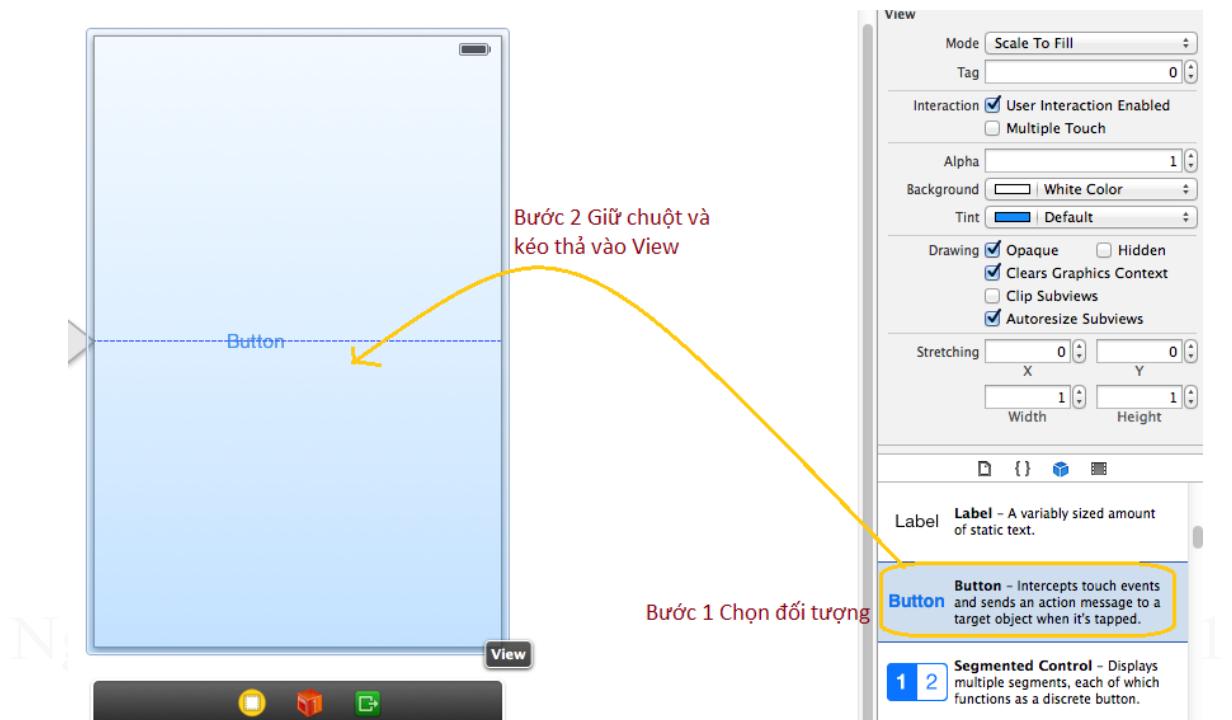
Hình 2.17 Utility area

Trong Inspector selector bar, có hai button bạn cần lưu ý là **Attribute** (Attribute icon) và **Quick Help** (Quick Help icon). Quick Help cho phép bạn tra cứu một cách nhanh chóng các đối tượng, hàm trong tài liệu kèm theo của Xcode. Attribute cho phép bạn thay đổi các thuộc tính của đối tượng.

Trong Library selector bar, có một số button quan trọng là **Code snippets** ({ icon}), **Objects** (Object icon). Code snippets hiển thị cho bạn danh sách các đoạn code mẫu để bạn lựa chọn sử dụng trong quá trình viết ứng dụng. Objects hiển thị các đối tượng của Xcode cho bạn sử dụng thiết kế giao diện ứng dụng.

## 2.1.4 Thiết Kế Giao Diện

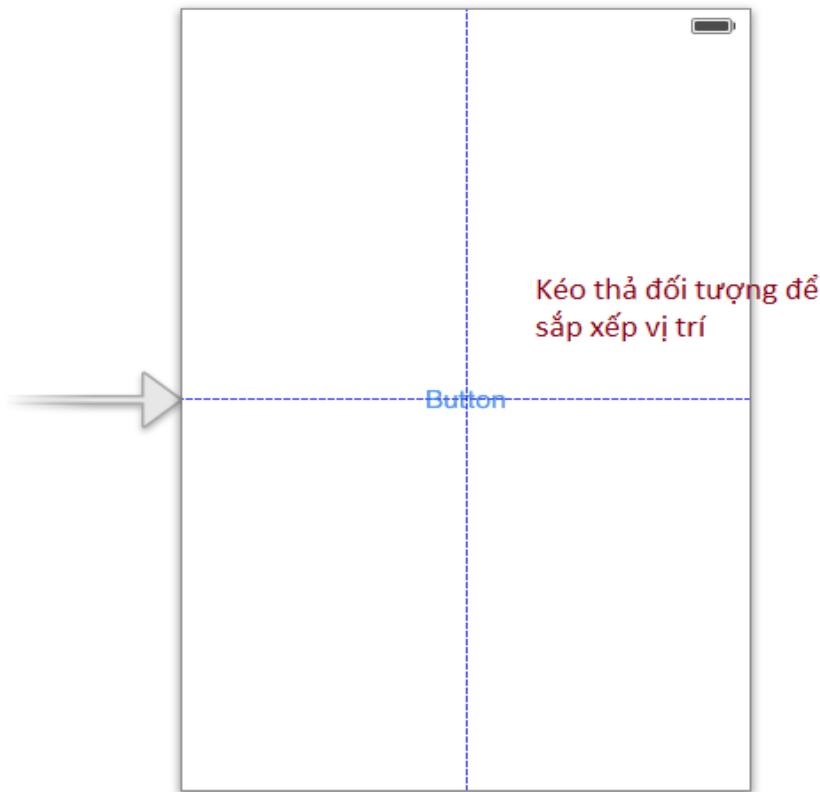
Giao diện ứng dụng trong Xcode được thiết kế thông qua **Interface Builder**, các đối tượng của Interface Builder được cung cấp trong **Utility area**. Để tiến hành thiết kế giao diện, bạn kéo thả đối tượng trong Utility area vào Interface Builder.



Hình 2.18 Kéo thả các đối tượng vào giao diện

Tại đây bạn có thể điều chỉnh, sắp xếp vị trí của các đối tượng theo ý tưởng thiết kế của bạn. Trong Xcode, tập tin bạn dùng để thiết kế giao diện là tập tin **.storyboard**.

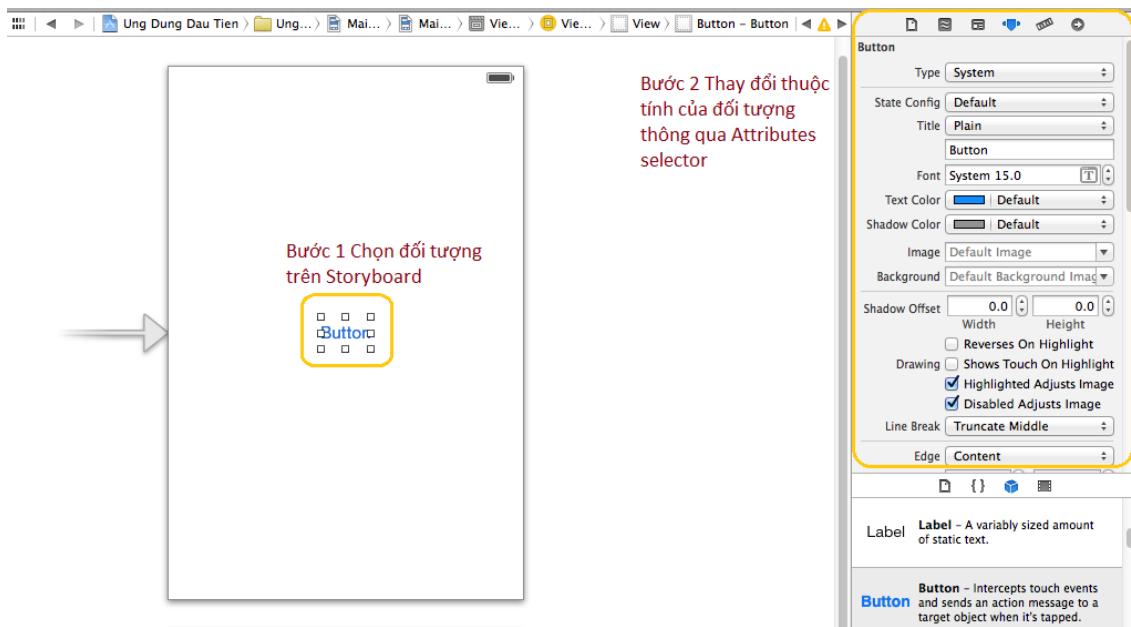
Khi kéo thả các đối tượng vào Interface Builder, bạn sẽ dễ dàng xác định vị trí đặt đối tượng sao cho giao diện cân đối, không bị hiện tượng lệch khi Run ứng dụng. Đó là nhờ vào tính năng hỗ trợ canh chỉnh của Xcode thông qua các đường kẻ đứt nét màu xanh.



Nguyễn Anh Tú | 2013

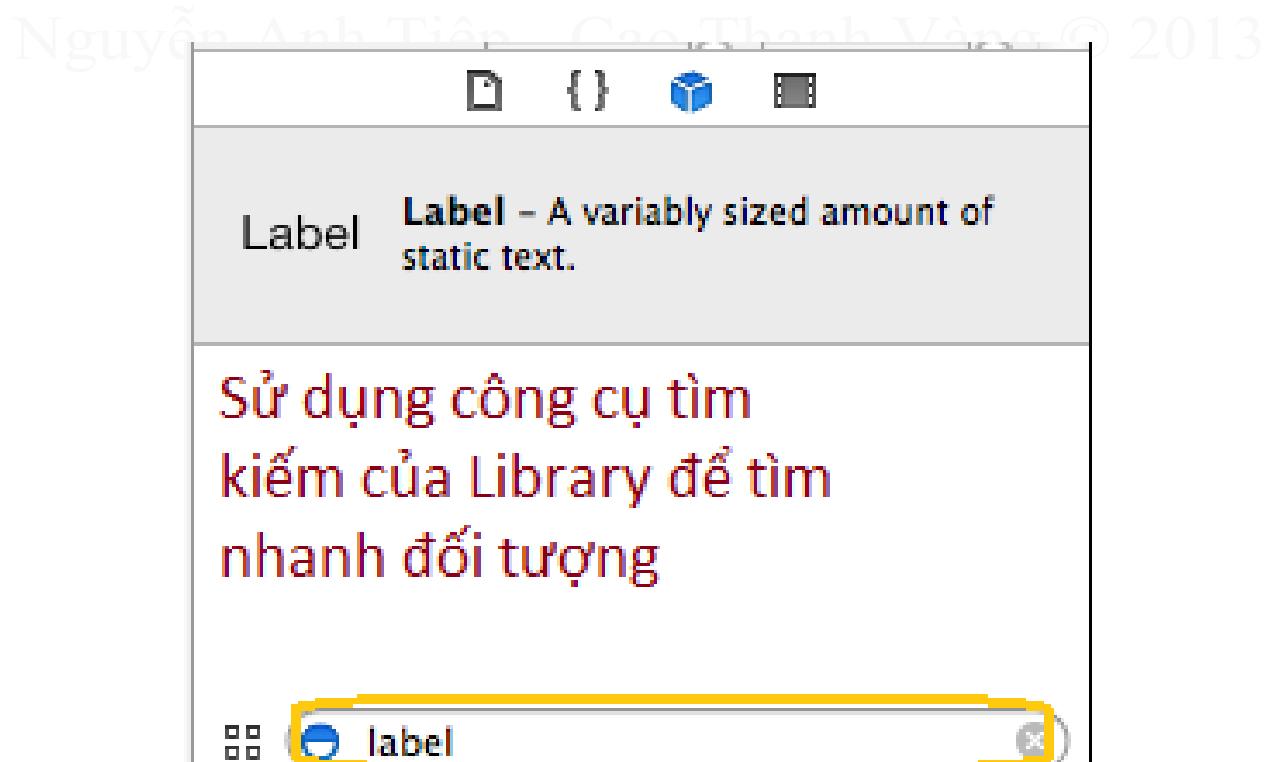
**Hình 2.19 Canh chỉnh vị trí theo đường kẻ xanh**

Bạn có thể điều chỉnh các thuộc tính của các đối tượng trong Interface Builder thông qua vùng Inspector pane.



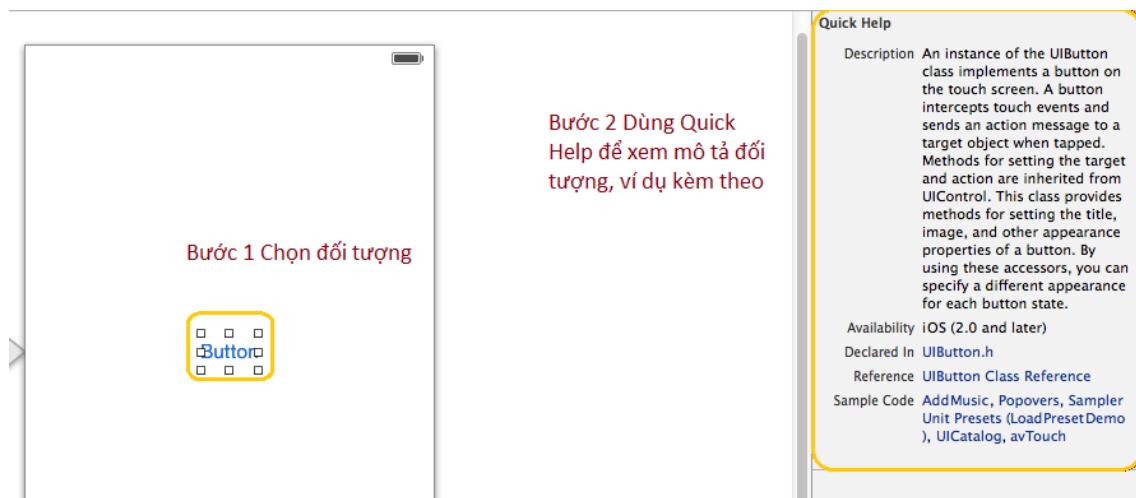
**Hình 2.20 Thay đổi thuộc tính của đối tượng**

Để tìm kiếm đối tượng một cách nhanh chóng, bạn sử dụng công cụ tìm kiếm trong Library pane.



**Hình 2.21 Tìm kiếm trong Library**

Hơn nữa, nếu bạn muốn tìm hiểu rõ hơn về đối tượng đó, xem ví dụ cụ thể minh họa, bạn có thể sử dụng đến bộ tài liệu hỗ trợ của Xcode.



**Hình 2.22 Xem tài liệu hỗ trợ của đối tượng**

### 2.1.5 Viết Code

Phần code của ứng dụng thường được viết trong hai tập tin là **.h** và **.m**. Tập tin **.h** thường được sử dụng để kết nối và khai báo các đối tượng của Interface Builder trước khi muốn sử dụng các đối tượng này để lập trình. Ngoài ra tập tin **.h** còn dùng để khai báo các hàm sự kiện trước khi sử dụng trong tập tin **.m**. Tập tin **.m** dùng để triển khai các hàm sự kiện mà bạn đã khai báo bên tập tin **.h**.

Tập tin dùng để khai báo các đối tượng, các hàm

```

// ViewController.h
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface ViewController : UIViewController
@end

```

Tập tin dùng để viết chương trình, triển khai các hàm

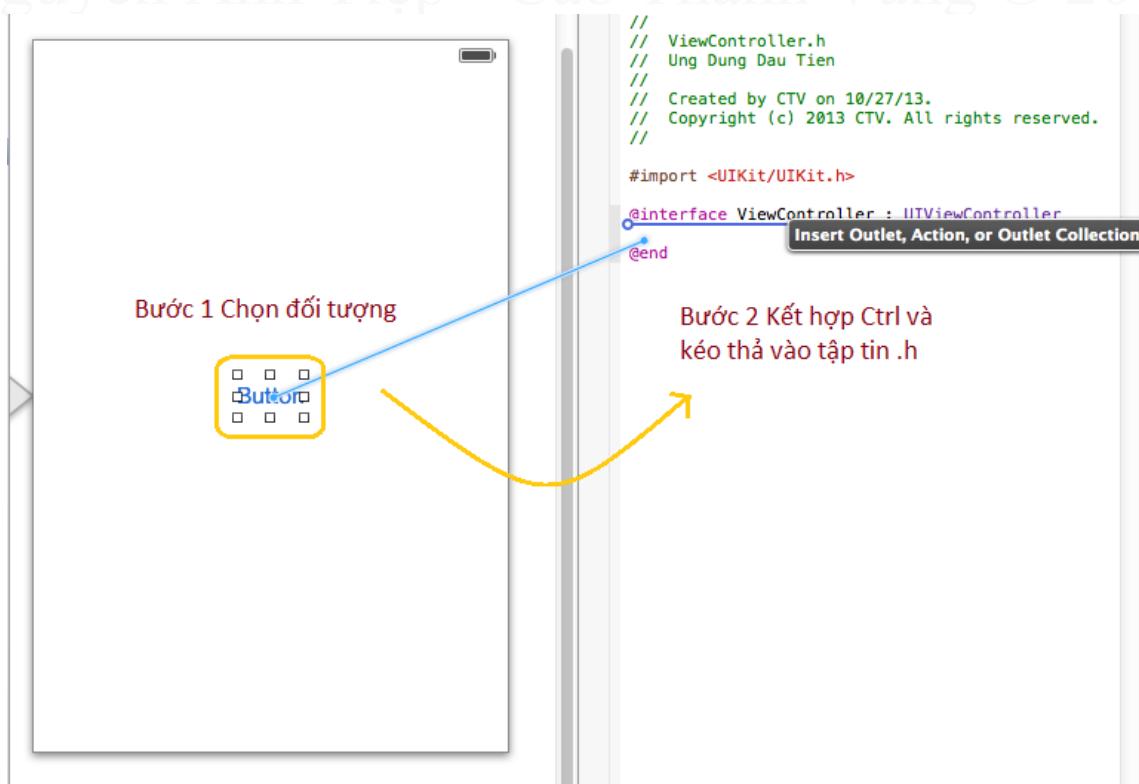
```

// ViewController.m
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.
//
#import "ViewController.h"
@interface ViewController ()
@end
@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    int a;
    // Dispose of any resources that can be recreated.
}
@end

```

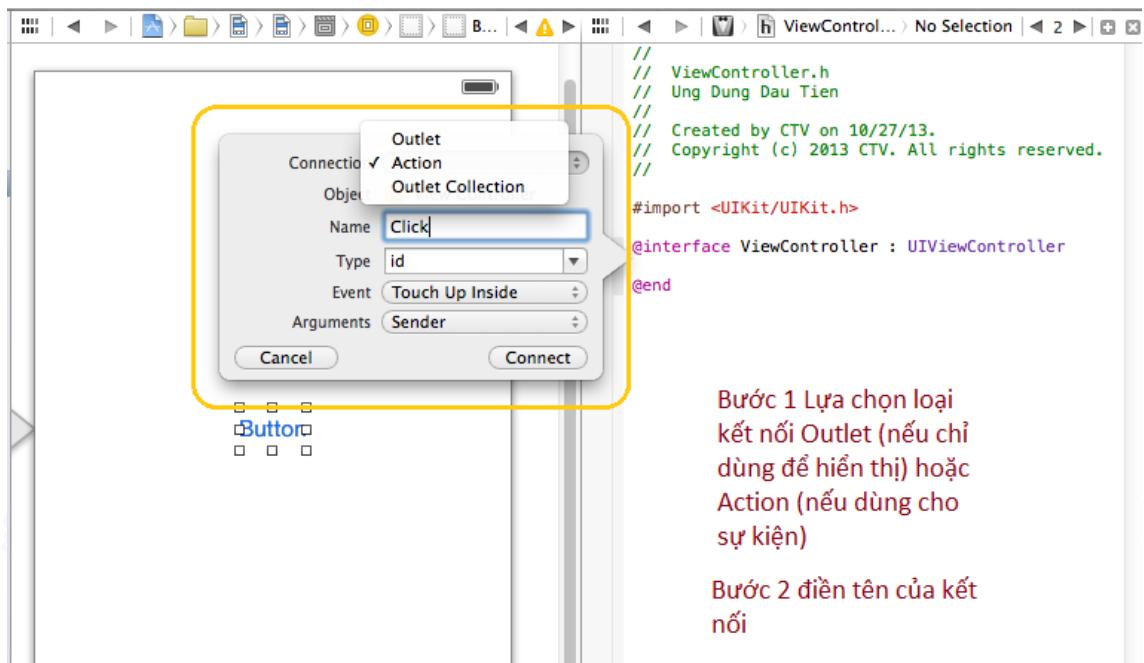
**Hình 2.23 Hai tập tin sử dụng để viết code cho ứng dụng**

Để kết nối đối tượng trong Interface Builder vào tập tin .h (ảnh xạ), bạn nhấn kết hợp **Ctrl + nhấp chuột vào đối tượng và kéo thả vào tập tin .h**.



**Hình 2.24 Ánh xạ đối tượng vào tập tin .h**

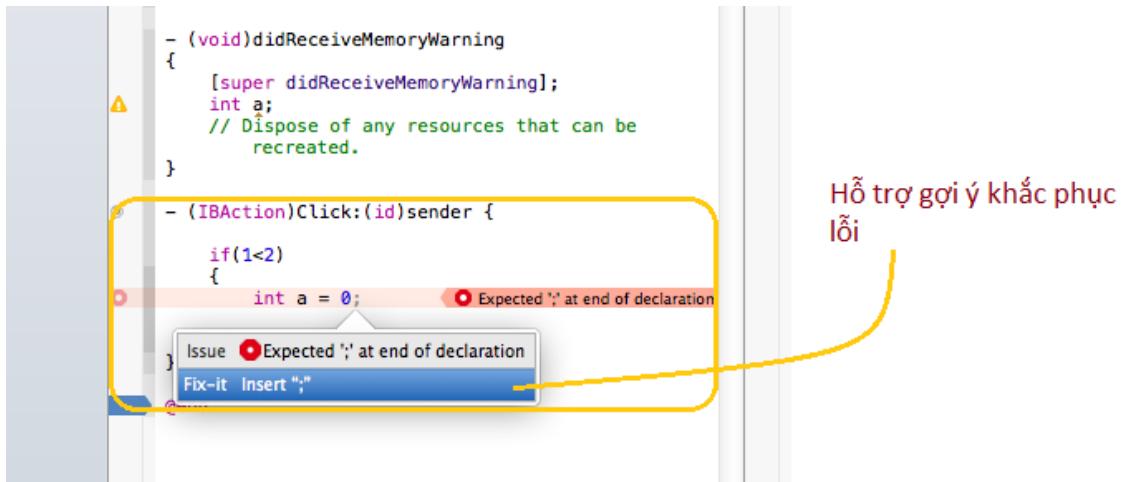
Khi hộp thoại hiện ra, bạn lựa chọn loại kết nối cho đối tượng là **Action** hoặc **Outlet**. Hiểu một cách đơn giản, đối tượng nào mà bạn sử dụng để hiển thị thông tin ra bên ngoài thì thuộc loại Outlet. Đối tượng nào mà bạn muốn viết code để khi tương tác với đối tượng đó sẽ cho ra kết quả mà bạn muốn ( ví dụ bạn muốn nhấn vào Button sẽ hiện “Hello World” ) thì bạn sẽ chọn loại là Action. Một đối tượng có thể vừa là Action, vừa là Outlet tùy vào người viết ứng dụng quy định.



**Hình 2.25 Lựa chọn kiểu ánh xạ**

Khi bạn viết code, Xcode hỗ trợ bạn trong việc phát hiện lỗi và thông báo cho bạn bằng hình tròn màu đỏ chứa dấu chấm than tại vị trí phát hiện lỗi. Ngoài ra Xcode cũng có chế độ cảnh báo bằng tam giác màu vàng cho đoạn code mà Xcode cho rằng cần cải thiện để bạn kịp thời kiểm tra lại và chỉnh sửa cho hiệu quả ( nếu cần ).

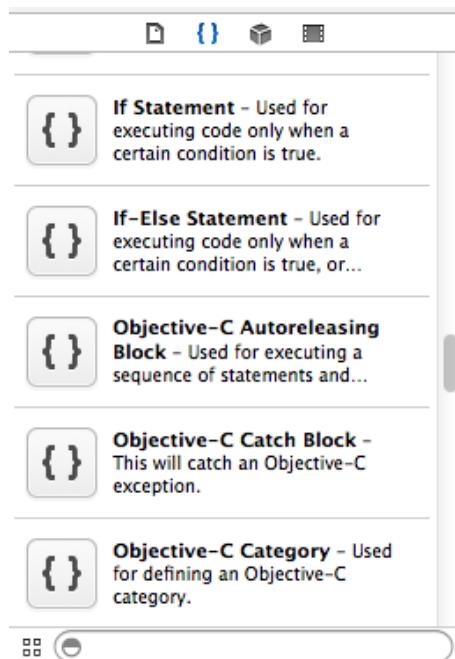
Xcode còn hỗ trợ bạn một chế độ gợi ý sửa lỗi trong quá trình viết code. Ví dụ bạn viết code nhưng quên dấu “;” , Xcode sẽ gợi ý cho bạn có dấu “;” để hoàn tất đoạn code hoàn chỉnh.



Hình 2.26 Gợi ý khắc phục lỗi

Xcode chỉ hỗ trợ khắc phục một số lỗi cơ bản về cú pháp, các lỗi liên quan khác trong quá trình viết code các bạn phải tự giải quyết

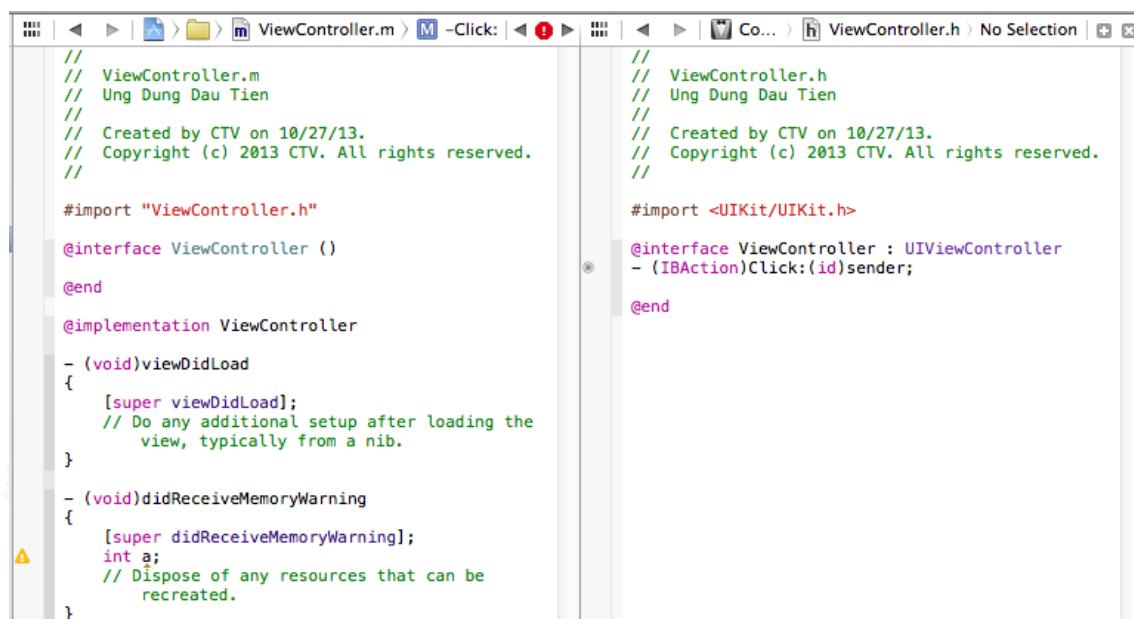
Trong Library pane, Xcode hỗ trợ sẵn cho bạn một số đoạn code mẫu trong thư viện, nếu bạn cần xài đoạn code nào, bạn chỉ cần kéo thả đoạn code đó từ Library pane sang Editor area.



Hình 2.27 Một số code mẫu để sử dụng

Nếu bạn có một đoạn code, bạn muốn lưu nó lại để lần sau sử dụng thì bạn chỉ cần chọn đoạn code đó và kéo thả vào Library pane, đặt tên cho đoạn code đó và lưu lại. Như vậy lần sau bạn muốn sử dụng lại, chỉ việc kéo thả từ Library sang là được.

Để thuận tiện cho việc viết code nhanh chóng, bạn có thể bật chế độ **chia đôi vùng Editor area làm hai**, lúc này bạn có thể xem cùng lúc cả tập tin .h lẫn tập tin .m. Muốn thực hiện điều đó, trong Toolbar, bạn nhấp chuột vào nút **Assistant Editor Button** (  ).



```
// ViewController.m
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.

#import "ViewController.h"

@interface ViewController : UIViewController
- (IBAction)Click:(id)sender;
@end

@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    int a;
    // Dispose of any resources that can be recreated.
}
}

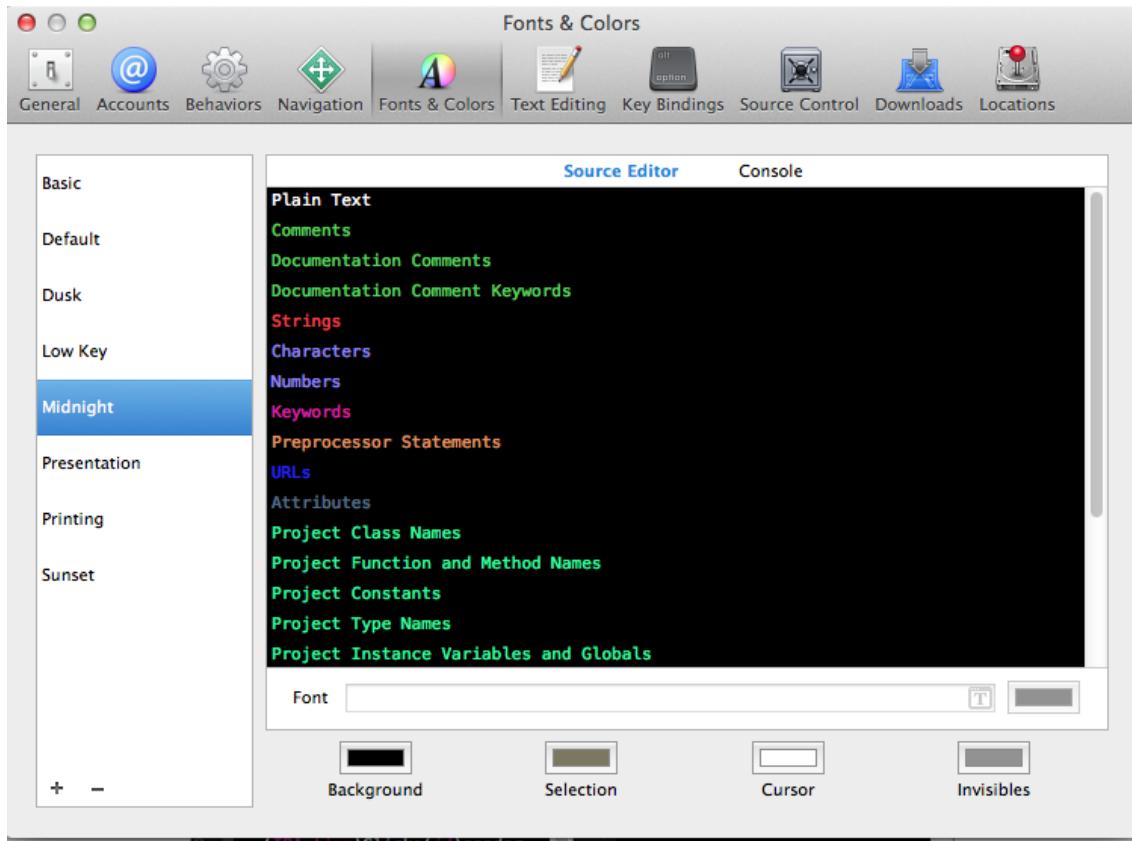
// ViewController.h
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
- (IBAction)Click:(id)sender;
@end
```

Hình 2.28 Chia đôi màn hình Editor để tiện làm việc

Bạn có thể tùy chỉnh lại font và màu chữ để tạo nên một sự thay đổi, tạo thêm cảm hứng mới. Bạn chọn **Xcode** > chọn **Preferences** > chọn **Fonts & Colors**, sau đó bạn tùy chọn một định dạng mình thích.



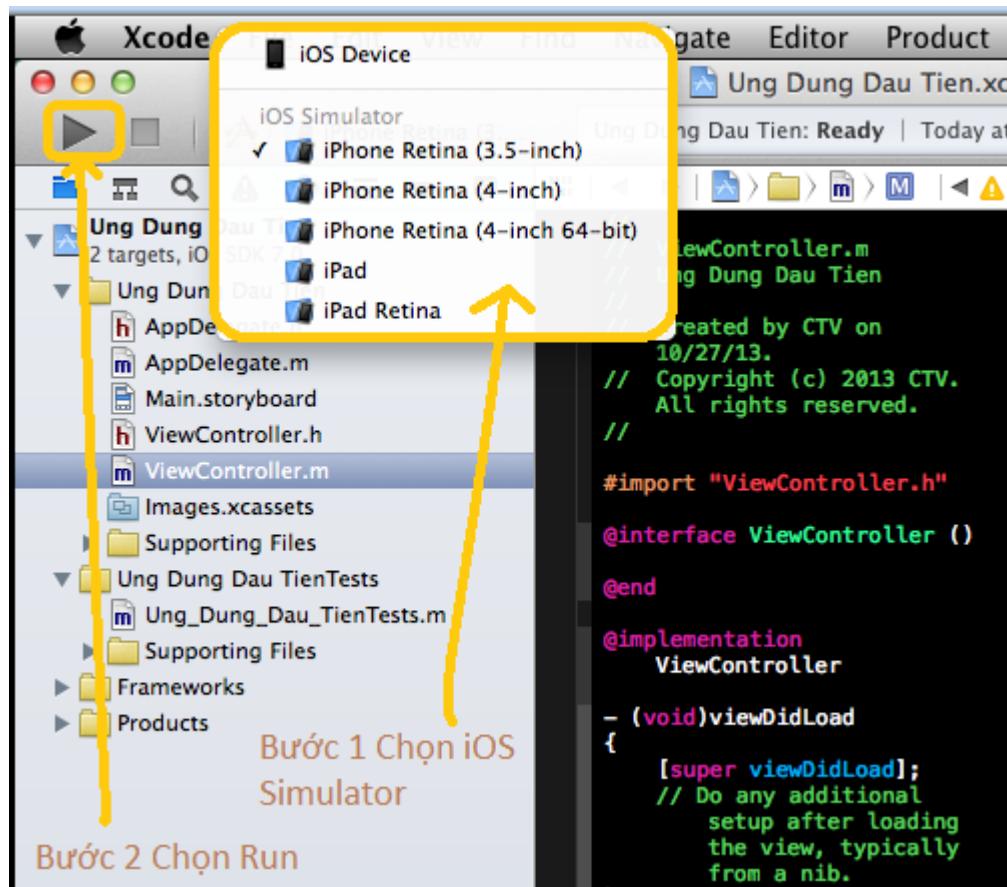
Hình 2.29 Tùy chỉnh Font & Color cho giao diện lập trình code của Xcode

Trong quá trình viết chương trình, bạn cũng có thể sử dụng nút Quick Help trong Utility area để mở tài liệu tham khảo và tra cứu các hàm cũng như xem ví dụ minh họa cho các hàm.

## 2.1.6 Thực Thi Và Kiểm Tra Lỗi Của Ứng Dụng

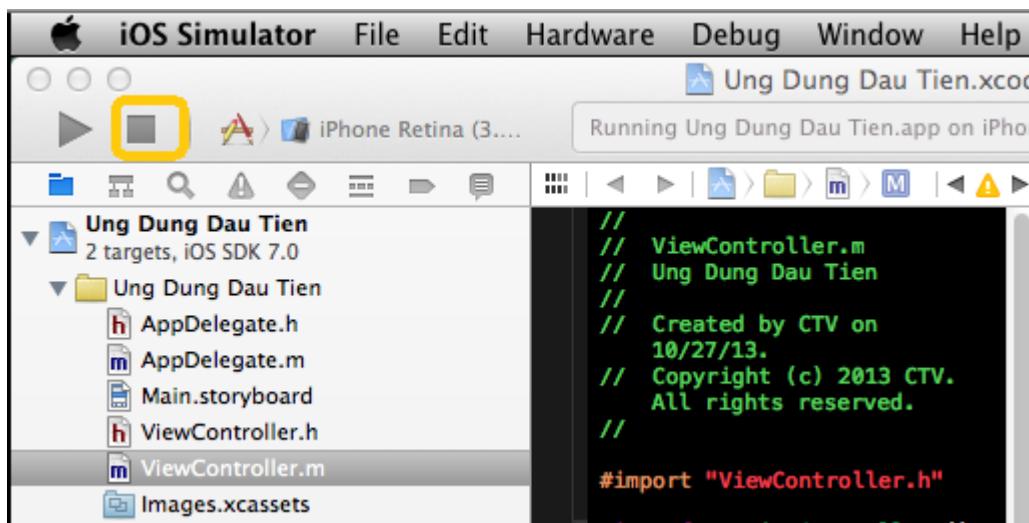
### 2.1.6.1 Thực Thi Ứng Dụng

Khi bạn muốn chạy thử và đưa ứng dụng lên iOS Simulator, bạn sử dụng các button chuyên dụng trên Toolbar. Để chạy ứng dụng, trước tiên bạn phải chọn lựa thiết bị mà bạn muốn chạy ứng dụng lên đó. Bạn có thể chọn chạy lên thiết bị thật hoặc trên iOS Simulator. Trong iOS Simulator, tùy theo ứng dụng của bạn viết cho thiết bị nào mà lựa chọn thiết bị đó, ví dụ iPad, iPhone, iPhone Retina...



**Hình 2.30 Lựa chọn thiết bị iOS Simulator**

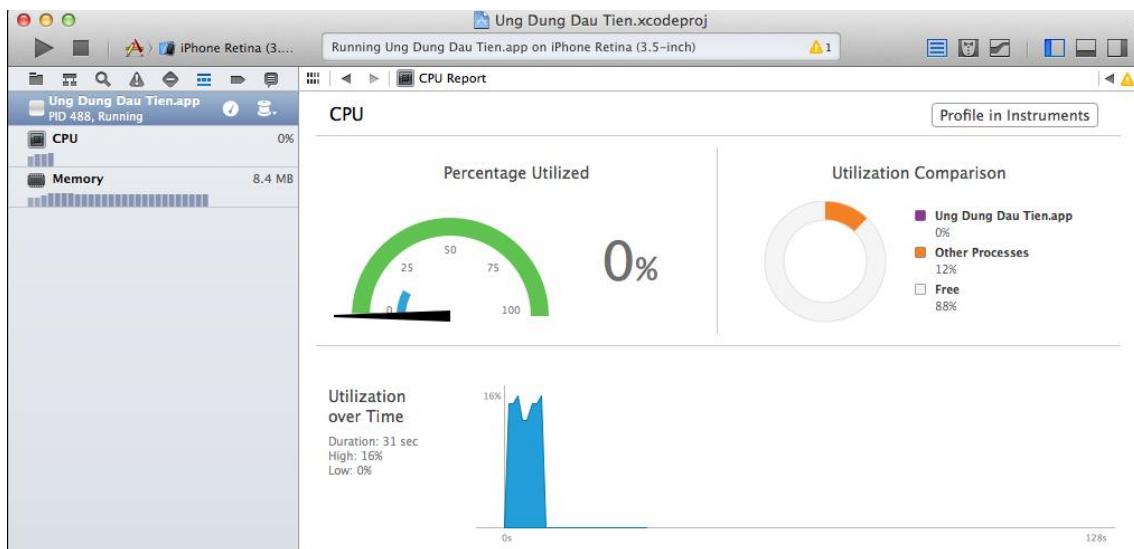
Sau khi chọn lựa xong phần thiết bị, bạn nhấn **Run** để chạy chương trình. Nếu muốn dừng chương trình, bạn nhấn **Stop**.



**Hình 2.31 Nhấn Stop để dừng chương trình**

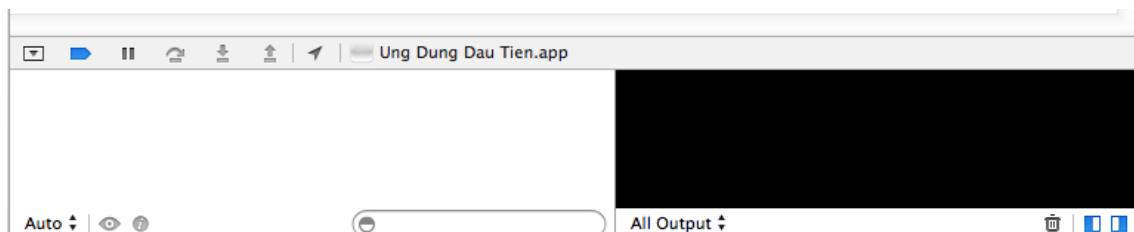
### 2.1.6.2 Kiểm Tra Lỗi Của Ứng Dụng

Vùng Debug area hỗ trợ bạn debug ứng dụng để kiểm tra từng bước, ngoài ra còn có vùng Debug Navigator cho phép bạn theo dõi các tiến trình, cũng như việc sử dụng RAM, CPU.



Hình 2.32 Vùng Debug Navigator

Trong Debug area có các nút hỗ trợ như bật tắt các Breakpoint, nhóm nút hỗ trợ thực thi ứng dụng từng bước hỗ trợ bạn debug một cách chính xác hơn.



Hình 2.33 Vùng Debug area

Để đánh dấu breakpoint, bạn chọn vị trí cần đánh dấu, và nhấp chuột vào Breakpoint gutter tương đương với vị trí đó.

Chọn vị trí cần Debug và đặt Breakpoint

```

@interface ViewController ()
```

```

@end
```

```

@implementation ViewController
```

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}
```

```

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    int a=0;
    // Dispose of any resources that can be recreated.
}
```

```

- (IBAction)Click:(id)sender {
}
```

```

@end

```

**Hình 2.34 Đánh dấu Breakpoint**

Ngoài ra nếu bạn muốn thu gọn 1 đoạn code nào đó, bạn có thể sử dụng thanh đứng hỗ trợ bên cạnh thanh đứng đặt Breakpoint.

Chọn đoạn code cần thu gọn rồi nhấp chuột vào vị trí trên thanh đứng

Kết quả cho thấy đoạn code được thu gọn

```

// ViewController.m
// Ung Dung Dau Tien
//
// Created by CTV on 10/27/13.
// Copyright (c) 2013 CTV. All rights reserved.
//

#import "ViewController.h"

@interface ViewController ()
```

```

@end
```

```

@implementation ViewController
```

```

- (void)viewDidLoad
{ ... }
```

```

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    int a=0;
    // Dispose of any resources that can be recreated.
}
```

```

- (IBAction)Click:(id)sender {
}
```

```

@end

```

**Hình 2.35 Thu gọn đoạn code cho dễ nhìn**

## 2.2 TÌM HIỂU IOS SIMULATOR

### 2.2.1 Giới Thiệu iOS Simulator

**iOS Simulator** là một phần trong bộ công cụ kèm theo của phần mềm Xcode. iOS Simulator chứa **iOS SDK** cho phép bạn chạy trên Mac OS để giả lập môi trường iPhone, iPad nhằm phục vụ cho việc kiểm thử ứng dụng được viết ra trước khi kiểm thử ứng dụng trên thiết bị thật.

**iOS Simulator** cho phép bạn cài đặt nhiều thiết bị iOS khác nhau như iPhone, iPhone Retina, iPad, iPad Retina.. với nhiều phiên bản iOS khác nhau như 6.0, 6.1, 7.0.... Do đó bạn có thể dễ dàng xây dựng ứng dụng của mình dành cho phiên bản iOS mới hoặc dùng cho cả phiên bản iOS cũ.

Với **iOS Simulator**, bạn có thể kiểm thử ứng dụng của bạn về thiết kế giao diện, về tính năng của ứng dụng, từ đó có thể khắc phục các lỗi phát sinh, tối ưu hóa ứng dụng trước khi bạn đem ứng dụng lên thiết bị thật.

Bạn có thể đọc thêm tài liệu iOS Simulator User Guide trên iOS Developer Library để tìm hiểu thêm, cũng như có thêm kinh nghiệm sử dụng iOS Simulator.

Xem tại :

[https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS\\_Simulator\\_Guide/Introduction/Introduction.html](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/Introduction/Introduction.html)

### 2.2.2 Tìm Hiểu iOS Simulator

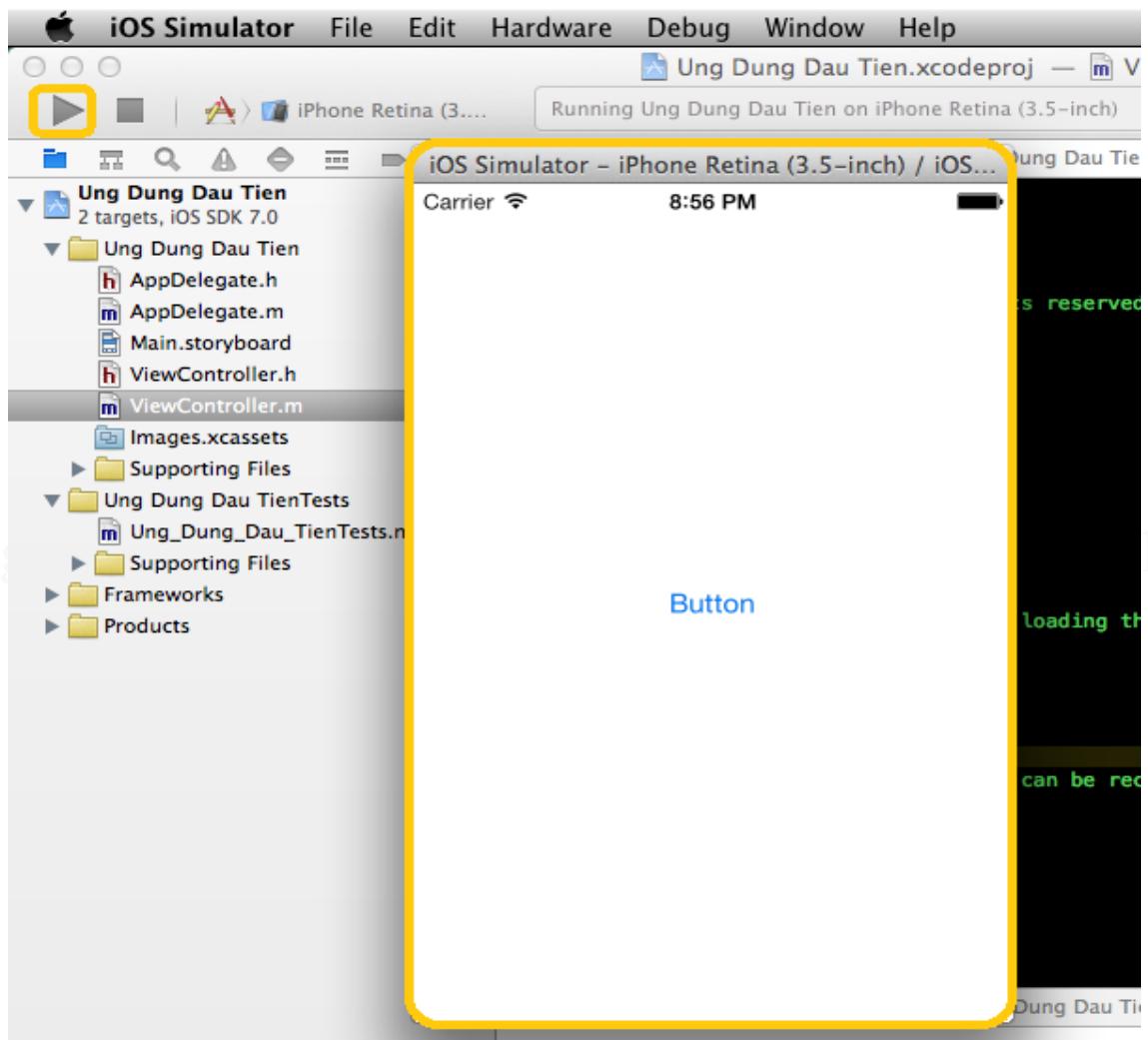
Ứng dụng iOS Simulator có thể chạy chung với phần mềm Xcode hoặc chạy độc lập đều được. Bạn có thể tương tác với iOS Simulator thông qua bàn phím, chuột để nhập dữ liệu cũng như điều khiển các sự kiện của người dùng.

Phần này sẽ giúp bạn tìm hiểu một số điểm cơ bản của iOS Simulator để hỗ trợ bạn tốt hơn trong quá trình viết ứng dụng cho iOS.

### 2.2.2.1 Thao Tác Cơ Bản Với IOS Simulator

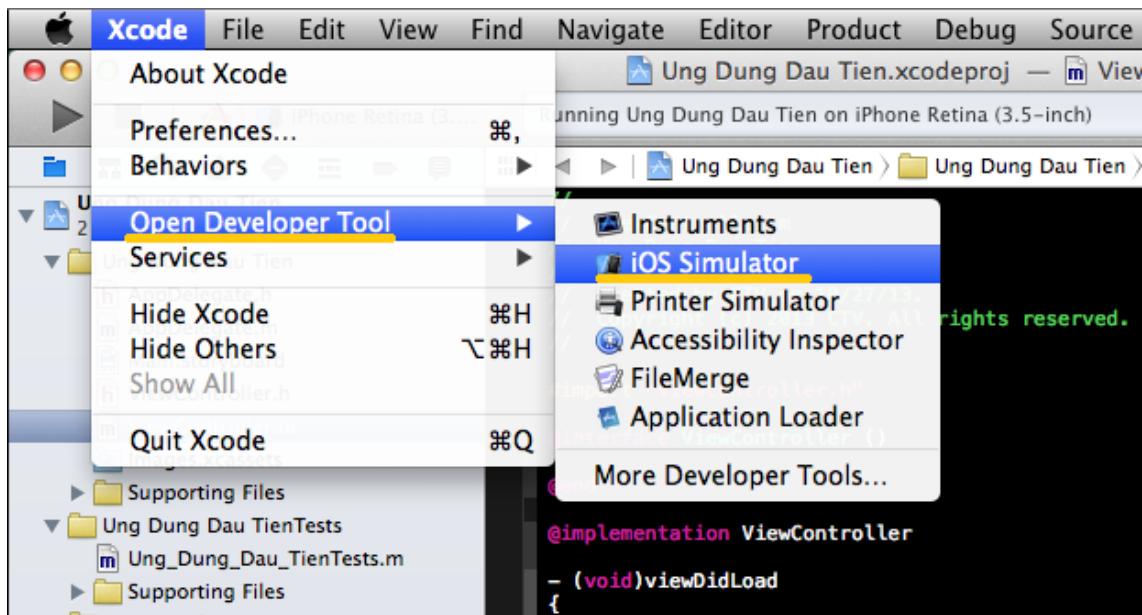
#### Thao tác mở và thoát ios simulator

Để mở iOS Simulator bạn có hai cách. Một là bạn **chạy ứng dụng trong Xcode để khởi động iOS Simulator**. Với cách này bạn chỉ cần chọn iOS Simulator phù hợp rồi chọn Run.



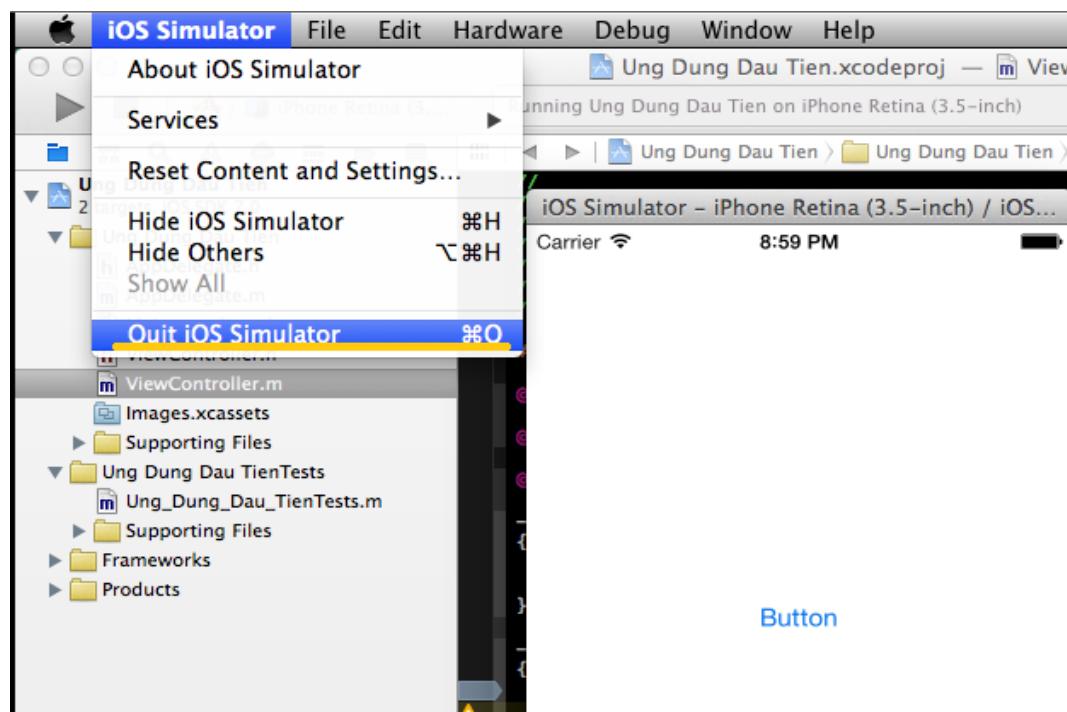
Hình 2.36 Chọn Run để mở iOS Simulator

Cách thứ hai là bạn chọn menu **Xcode > chọn Open Develop Tool > iOS Simulator**. Khi ấy iOS Simulator sẽ được khởi động.



Hình 2.37 Mở iOS Simulator trong Menu

Mặc dù là một phần trong bộ công cụ của Xcode, nhưng iOS Simulator vẫn có thể tiếp tục hoạt động dù Xcode có bị đóng chương trình. Do đó nếu bạn muốn thoát hẳn iOS Simulator, bạn chọn menu iOS Simulator > chọn **Quit iOS Simulator**.

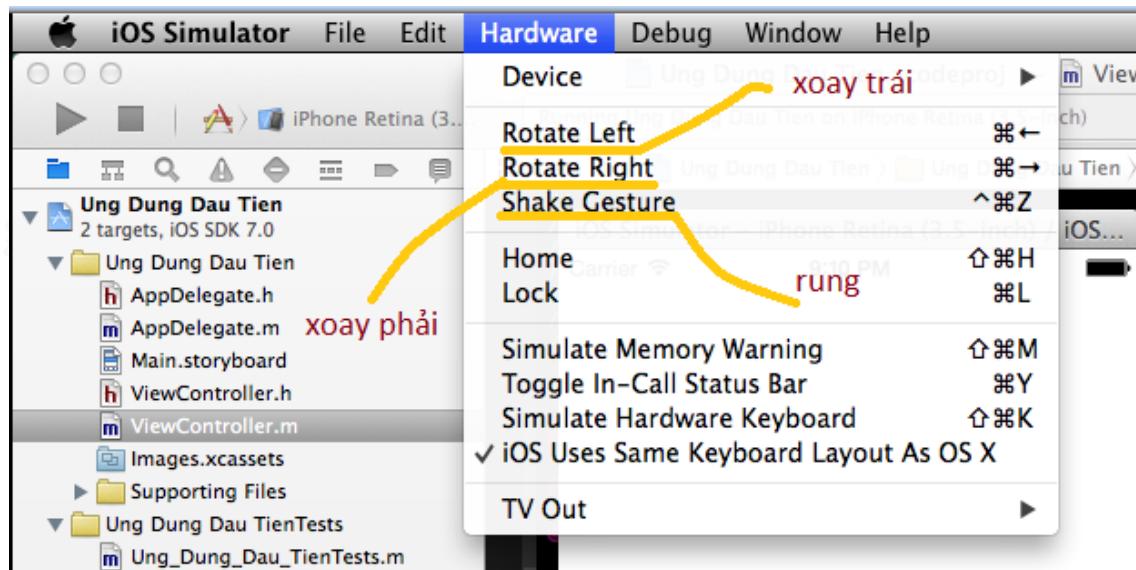


Hình 2.38 Trong menu chọn **Quit iOS Simulator**

## Xoay màn hình iOS Simulator

Trong quá trình chạy ứng dụng trên iOS Simulator để kiểm thử, đôi lúc bạn cần sử dụng tới chức năng xoay màn hình để có thể kiểm tra tính tương thích của ứng dụng với từng kiểu màn hình ( ngang, đứng...) hoặc để phù hợp với ứng dụng của bạn ( chẳng hạn viết ứng dụng sử dụng màn hình ngang). Nếu là thiết bị thật, thật dễ dàng để bạn có thể xoay màn hình cho phù hợp. Tuy nhiên với iOS Simulator, bạn cần phải sử dụng đến chức năng xoay màn hình được hỗ trợ sẵn để có thể xoay màn hình theo ý muốn.

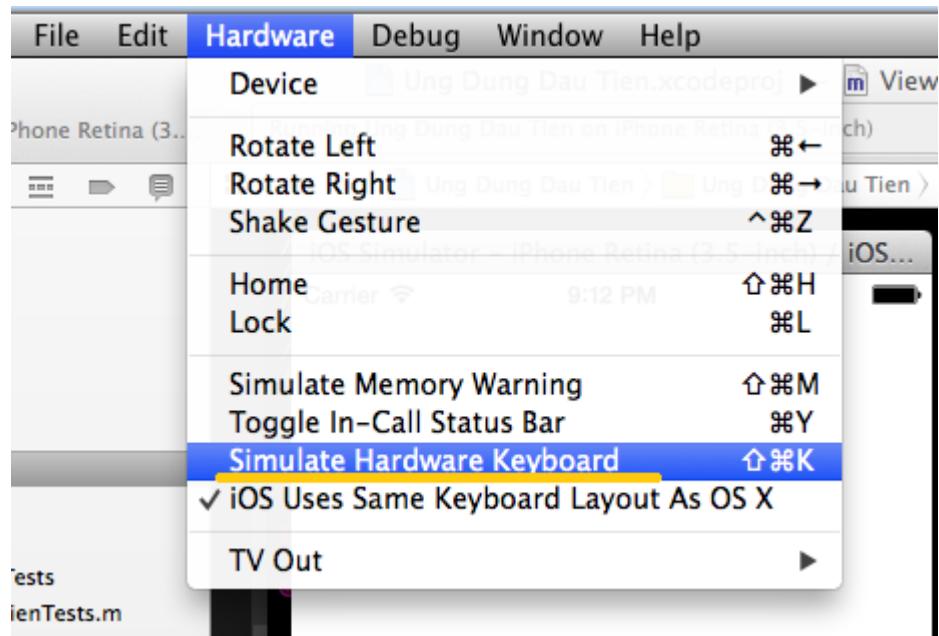
Bạn có thể vào menu **Hardware** > chọn **Rotate Left** nếu bạn muốn xoay qua trái ; chọn **Rotate Right** nếu bạn muốn xoay qua phải ; chọn **Shake Gesture** nếu bạn muốn rung nhẹ.



Hình 2.39 Trong menu chọn xoay màn hình iOS Simulator

## Ẩn/hiện keyboard

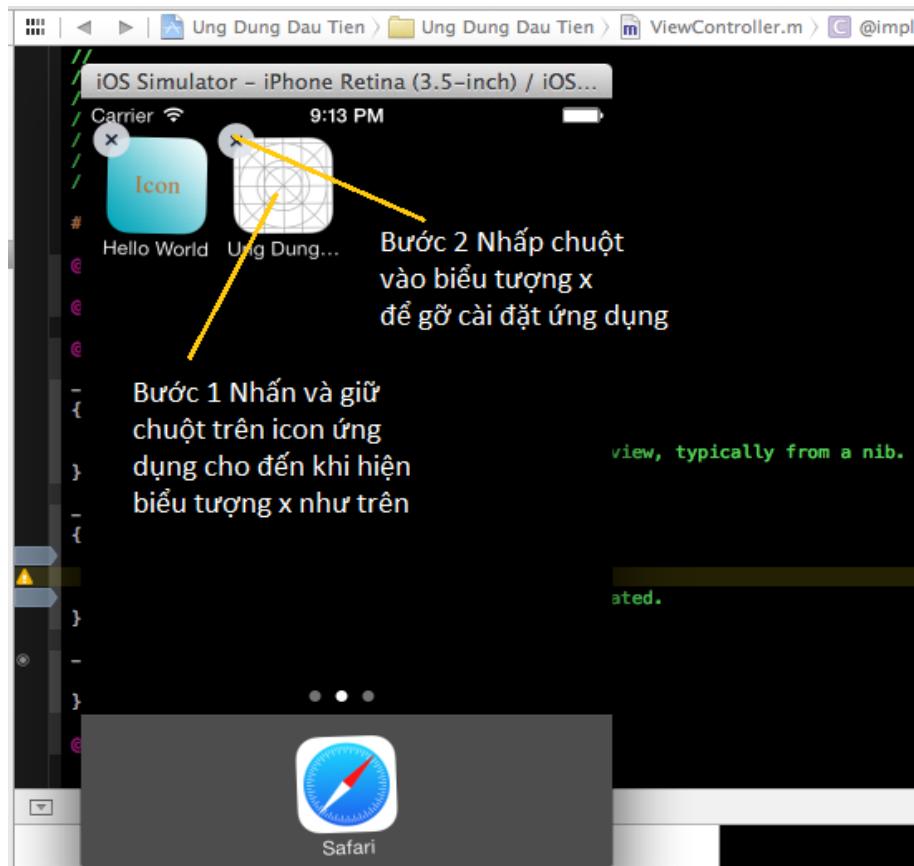
Trong quá trình kiểm thử ứng dụng, nhiều trường hợp bạn cần sử dụng đến bàn phím của iOS, hoặc sau khi nhập liệu xong trong TextField nhưng ứng dụng của bạn chưa có chức năng ẩn bàn phím đi, lúc đó bạn cần sử dụng đến tính năng Keyboard của iOS Simulator để ẩn/hiện bàn phím. Bạn có thể vào menu **Hardware** > chọn **Simulate Hardware Keyboard**.



**Hình 2.40** Ẩn/Hiện keyboard trong iOS Simulator

### Cài đặt và gỡ bỏ ứng dụng trên iOS Simulator

Ứng dụng trong iOS Simulator được cài đặt thông qua Xcode. Khi bạn chạy ứng dụng bằng Xcode thì Xcode sẽ cài đặt ứng dụng đó vào iOS Simulator. Cách thức gỡ bỏ ứng dụng cũng giống như trên thiết bị iOS thật. Bạn chỉ cần nhấp và giữ chuột (hoặc trackpad) trên biểu tượng của ứng dụng cho đến khi xuất hiện biểu tượng dấu x, bạn chỉ cần nhấp vào dấu x để gỡ bỏ ứng dụng. Sau khi hoàn tất chỉ cần ấn Home để trở lại ban đầu.



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

**Hình 2.41 Gỡ ứng dụng trong iOS Simulator**

### Bổ sung thêm các phiên bản iOS và các thiết bị iOS

iOS Simulator cho phép bạn có thể chạy ứng dụng trên nhiều loại thiết bị như iPhone, iPhone Retina, iPad, iPad Retina. Đồng thời, iOS Simulator cũng cho phép bạn sử dụng nhiều phiên bản khác nhau của iOS như iOS 6.0, iOS 6.1, iOS 7.0.

Mặc định sau khi cài Xcode 5, iOS Simulator kèm theo đã được cài đặt để hỗ trợ các thiết bị iPhone Retina, iPad Retina và iOS 7.0. Nếu bạn muốn iOS Simulator chạy các thiết bị iPhone, iPad thông thường và các phiên bản iOS thấp hơn như iOS 6.0, iOS 6.1 thì bạn cần phải tải và cài đặt thêm. Bạn vào **Xcode > chọn Preferences > chọn mục Download**. Tại đây bạn lựa chọn phiên bản iOS cần cài đặt thêm và tải về.



**Hình 2.42 Tải thêm các iOS Simulator phiên bản cũ hoặc tài liệu**

### Chụp ảnh màn hình iOS Simulator

Nếu bạn muốn chụp ảnh màn hình của iOS Simulator, bạn có thể lưu lại ảnh chụp màn hình của iOS Simulator lên màn hình của Mac OS. Để làm việc đó, bạn chọn **File** > chọn **Save Screen Shot**, khi đó ảnh chụp màn hình sẽ được lưu trên màn hình Mac OS.



**Hình 2.43 Chụp ảnh màn hình iOS Simulator**

## Copy - Paste trong iOS Simulator

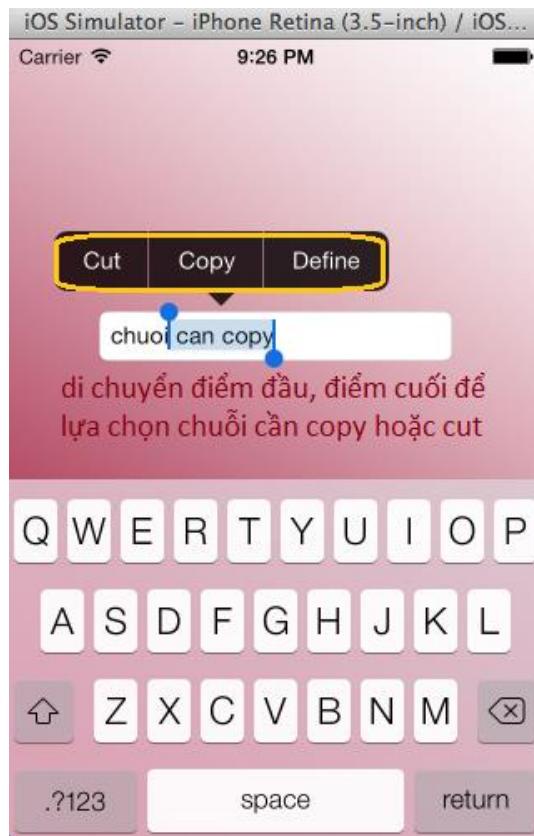
Trong iOS Simulator cũng hỗ trợ bạn Copy và Paste một chuỗi.

Để Copy một chuỗi, bạn **nhấp chuột vào chuỗi** để hiển thị nút **Select** và **Select All**. Chọn Select nếu bạn muốn lựa chọn một từ nào đó, hoặc Select All nếu muốn chọn tất cả.



**Hình 2.44 Chọn Select hoặc Select All**

Di chuyển điểm đầu và điểm cuối để đánh dấu lại chuỗi cần chọn > chọn **Copy**



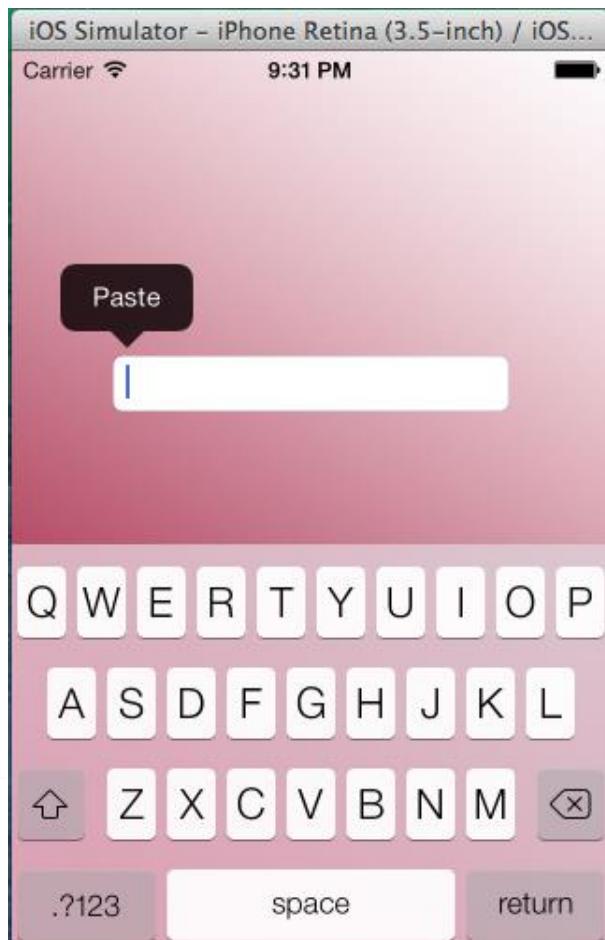
Nguyễn / **Hình 2.45 Lựa chọn chuỗi cần copy và chọn Copy** © 2013

Để Paste một chuỗi vào iOS Simulator, trước tiên bạn chọn **Edit > Paste** để chuyển chuỗi được copy từ **Mac vào iOS Simulator**.



**Hình 2.46 Paste từ Mac OS vào iOS Simulator**

Sau đó **chọn vị trí muốn Paste** chuỗi trong iOS Simulator > **Double-click** vào vị trí đó để hiện ra nút Paste > chọn **Paste**.



**Hình 2.47 Chọn Paste**

#### 2.2.2.2 Một Số Hạn Chế Của iOS Simulator

Mặc dù iOS Simulator rất hữu ích cho bạn kiểm thử ứng dụng trước khi đưa lên thiết bị thật, tuy nhiên bản thân iOS Simulator vẫn còn một số hạn chế nhất định. Đối với phần cứng, iOS Simulator vẫn còn khiếm khuyết ở một số điểm như **không có camera, không có microphone...** Ngoài ra còn **một số framework không được hỗ trợ** như Media player, Messenger UI ... Nếu như ở các phiên bản trước của Xcode, iOS Simulator còn hỗ trợ được với các phiên bản của iOS thấp hơn như iOS thì trong phiên bản này, iOS Simulator chỉ hỗ trợ từ phiên bản iOS 6.0 trở lên.

### **CHƯƠNG III**

### **NGÔN NGỮ OBJECTIVE-C**

Mặc dù Xcode hỗ trợ nhiều ngôn ngữ trong việc lập trình ứng dụng trong iPhone, nhưng đóng vai trò chủ yếu nhất vẫn là ngôn ngữ Objective-C bởi sự thân thiện, dễ sử dụng của nó. Chương này sẽ hướng dẫn bạn một số nét cơ bản của ngôn ngữ lập trình Objective-C với hi vọng bạn sẽ nắm được sơ lược cách sử dụng, cú pháp của ngôn ngữ này để thuận tiện hơn trong việc xây dựng ứng dụng. Nội dung chương sẽ trình bày sơ lược một số vấn đề sau của ngôn ngữ Objective-C:

- ❖ Khai báo biến
- ❖ Kiểu dữ liệu
- ❖ Các phép toán
- ❖ Hàm (Function)
- ❖ Cấu trúc điều kiện
- ❖ Cấu trúc lặp
- ❖ Mảng
- ❖ Chuỗi

### **3.1 GIỚI THIỆU NGÔN NGỮ OBJECTIVED-C**

Ngôn ngữ Objective-C được tạo ra bởi Brad Cox và Tom Love vào năm 1980 tại công ty Stepstone. Từ năm 1988, công ty NeXT Sofware nắm giữ bản quyền của ngôn ngữ Objective-C. Họ đã phát triển các bộ thư viện và cả môi trường phát triển cho nó có tên là NEXTSTEP.

Đến cuối tháng 12 năm 1996, hãng Apple đã mua lại công ty NeXT Software, môi trường NEXTSTEP/OPENSTEP đã trở thành phần cốt lõi của hệ điều hành OS X mà Apple giới thiệu sau này. Phiên bản chính thức của môi trường phát triển này do Apple giới thiệu ban đầu có tên là Cocoa.

Bằng việc hỗ trợ sẵn ngôn ngữ Objective-C, đồng thời tích hợp một số công cụ phát triển khác như Project Builder (đây là tiền thân của Xcode) và Interface Builder, Apple đã tạo ra một môi trường mạnh mẽ để phát triển ứng dụng trên Mac OS X. Đến năm 2007, Apple tung ra bản nâng cấp cho ngôn ngữ Objective-C và gọi đó là Objective-C 2.0.

Ngôn ngữ lập trình Objective-C dựa trên nền tảng ngôn ngữ C nhưng bổ sung thêm hỗ trợ lập trình hướng đối tượng. Objective-C là ngôn ngữ lập trình sử dụng để viết ứng dụng cho Apple's iOS và hệ điều hành Mac OS.

### **3.2 KHAI BÁO BIẾN - CÁCH SỬ DỤNG**

#### **3.2.1 Biến**

Biến được sử dụng để lưu trữ các giá trị của ứng dụng. Biến gồm có: kiểu dữ liệu, tên biến và giá trị của biến. Cú pháp:

Kiểu\_dữ\_liệu tên\_biến ;

hoặc

Kiểu\_dữ\_liệu tên\_biến = giá\_trị\_của\_bien;

Trong đó “=”: lệnh gán giá trị cho biến

VD: int x ; hoặc int x = 10;

### 3.2.2 Quy Tắc Đặt Tên

Quy tắc đặt tên biến:

- Ngôn ngữ Objective-C có phân biệt hoa thường.
- Tên biến không có dấu tiếng việt.
- Tên biến không có khoảng trắng.
- Tên biến không được bắt đầu bằng số.
- Tên biến không được có các ký tự đặc biệt (ngoại trừ dấu gạch dưới \_)
- Tên biến không được đặt trùng với các từ khoá của ngôn ngữ objective-C. VD:  
void, if, static, ...

### 3.3 KIỂU DỮ LIỆU

Kiểu dữ liệu sẽ giúp trình biên dịch xác định được loại dữ liệu (số nguyên, số thực, chuỗi,...) mà chúng ta muốn lưu trữ là gì từ đó sẽ cấp phát lượng bộ nhớ tương ứng với loại dữ liệu mà chúng ta cần lưu trữ. Objective-C hỗ trợ các kiểu dữ liệu cơ bản như sau:

Loại dữ liệu	Tên kiểu	Số ô nhớ	Miền giá trị
Kí tự	char	1 byte	-128 .. 127
	unsigned char	1 byte	0 .. 255
Số nguyên	int	4 bytes	- 2147483648 .. 2147483647
	unsigned int	4 bytes	0 .. 4294967295
	short	2 bytes	-32768 .. 32767
	unsigned short	2 bytes	0 .. 65535
	long	4 bytes	-2147483648 .. 2147483647
	unsigned long	4 bytes	0 .. 4294967295
	long long	8 bytes	-9,223,372,036,854,775,808 ..

			9,223,372,036,854,775,807
	unsigned long long	8 bytes	0 .. 18,446,744,073,709,551,615
Số thực	float	4 bytes	0 .. 3.4028235e+38
	double	8 bytes	0 .. 1.7976931E+308
	Long double	16 bytes	0 .. 1.1897315E+509
Logic	BOOL	1 bytes	0, 1; True, False; Yes, No

**Bảng 3.1 Kiểu dữ liệu trong Objectived-C**

### 3.4 PHÉP TOÁN

Phép toán	Ký hiệu	Ví dụ
Cộng	+	A + B
Trừ	-	A - B
Nhân	*	A * B
Chia	/	A / B
Lấy phần dư	%	M % N

**Bảng 3.2 Các phép toán trong Objectived-C**

Lưu ý: phép lấy phần dư chỉ được dùng trên 2 toán hạng kiểu số nguyên (nếu không sẽ sinh lỗi cú pháp)

Ví dụ:  $9 \% 5 = 4$

### 3.5 CHÚ THÍCH CODE

Chú thích một dòng code

//ghi chú chỉ trên 1 dòng

Chú thích một đoạn code

```
/*
Ghi chú trên 1
hay nhiều dòng
*/
```

### 3.6 XUẤT DỮ LIỆU RA MÀN HÌNH

Hàm NSLog là hàm đặc biệt của hệ thống, hàm này được thiết kế dùng cho việc hiển thị các thông báo lỗi.

Cú pháp:

`NSLog("Nội dung in" [, các biểu thức]);`

Nội dung in	Ví dụ
Các ký tự cần in	LacHong University
Các ký tự đặc biệt bắt đầu bởi dấu \	\n : xuống dòng; \t : ký tự Tab; \\" : ký tự \"; \; : ký tự ; \” : dấu “
Các mã định dạng giá trị của biểu thức	%mã_kiểu_dữ_liệu (kiểu float là f; kiểu int là d hay i)

Bảng 3.3 Ví dụ các kiểu xuất dữ liệu ra màn hình

VD: Dùng hàm NSLog để hiển thị chuỗi “LacHong University”

```
int main(int argc, const char * argv[])
{
    // insert code here...
    NSLog(@"LacHong University");

    return 0;
}
```

**Hình 3.1** Code sử dụng hàm NSLog()

Kết quả hiển thị:

```
2013-10-10 20:27:49.812 Demo1[369:303] LacHong University
Program ended with exit code: 0
```

**Hình 3.2** Kết quả in ra màn hình

**Ghi chú:** @”...”: bên trong cặp nháy đôi là chuỗi cần lưu log để kiểm tra.

```
int main(int argc, const char * argv[])
{
    // insert code here...
    int nam = 2013;
    NSLog(@"%@", @"Truong Dai Hoc \n \t Lac Hong %d", nam);

    return 0;
}
```

**Hình 3.3** Dùng NSLog() xuất ra màn hình một số.

Kết quả hiển thị:

```
2013-10-10 21:34:31.875 Demo1[336:303]
Truong Dai Hoc
    Lac Hong 2013
Program ended with exit code: 0
```

**Hình 3.4** Kết quả xuất ra chữ và số

## 3.7 FUNCTION

### 3.7.1 Định Nghĩa

Function Là tập hợp các dòng code, gom thành 1 khối. Khối code này được đặt tên.+ Khối code này chỉ được thực thi khi tên khối code được gọi.

### 3.7.2 Phương Thức Không Có Tham Số Truyền Vào

Khai báo:

```
- (kiểu_hàm) tên_hàm  
{  
//các câu lệnh  
[return [biểu_thức];]  
}
```

Trong đó: Anh Tiệp - Cao Thành Vàng © 2013

- Dấu “-“ đại diện cho loại phương thức mà ta phải gọi nó thông qua đối tượng của lớp chứa đựng phương thức đó.
- Kiểu hàm: là kiểu của <biểu thức> trong lệnh return.
- Tên hàm: do chúng ta tự nghĩ ra

VD:

```
- (char *)helloWorld {  
    return "Hello world!";  
}
```

Hình 3.5 Hàm HelloWorld

Gọi hàm:

```
NSLog(@"%@", [self helloWorld]);
```

### Hình 3.6 Xuất ra màn hình HelloWorld

Trong đó **self** là lớp hiện tại có chứa phương thức “helloWorld”.

#### 3.7.3 Phương Thức Có 1 Tham Số Truyền Vào

Khai báo:

```
- (kiểu_hàm) tên_hàm : (kiểu_dữ_liệu)tên_tham_số
{
    //các câu lệnh
    [return [biểu_thức];
}
```

VD:

```
- (char *)helloWorld:(char *)str {
    return str;
}
```

### Hình 3.7 Hàm HelloWorld với tham số truyền vào

Gọi hàm:

```
NSLog(@"%@", [self helloWorld:@"Hello World!"]);
```

### Hình 3.8 Gọi hàm HelloWorld

#### 3.7.4 Phương Thức Có Nhiều Tham Số Truyền Vào

Khai báo:

```

- (kiểu_hàm) tên_hàm : (kiểu_dữ_liệu)tham_số_1 [mô_tả]: (kiểu_dữ_liệu)tham_số_2
{
    //các câu lệnh

    [return [biểu_thúc];]
}

```

### Trong đó

- dấu hai chấm “:” được dùng để ngăn cách giữa các “tham\_số”.
- [mô\_tả]: có thể có hoặc không, dùng để mô tả cho tên tham số.

VD:

```

- (int) congHaiSo:(int)a :(int)b {
    return a + b;
}

```

Nguyễn Anh Tú - Cao Thanh Vàng © 2013

**Hình 3.9 Hàm cộng hai số**

### Trong đó

- a: là tham số thứ 1.
- b: là tham số thứ 2.

Gọi hàm:

```

NSLog(@"%@", [self congHaiSo:1 :2]);

```

**Hình 3.10 In kết quả hàm cộng**

Trong đó 1 và 2 là các đối số được truyền vào hàm “congHaiSo”.

## 3.8 CÂU TRÚC ĐIỀU KIỆN

### 3.8.1 Câu Lệnh If

Cú pháp

```
if( điều_kiện )  
{  
    //Công việc 1  
}
```

**Lưu ý:** điều kiện là một câu hỏi mà câu trả lời là YES hoặc NO, “Công việc 1” sẽ được thực hiện nếu điều kiện là YES.

Ví dụ:

- if (a > b) a có lớn hơn b không?
- if (a >= b) a có lớn hơn hoặc bằng b không?
- if (a < b) a có nhỏ hơn b không?
- if (a <= b) a có nhỏ hơn hoặc bằng b không?
- if (a == b) a có bằng b không?
- if (a != b) a có khác b không?

### 3.8.2 Câu Lệnh If – Else

Cú pháp

```
if( điều_kiện )  
{  
    //Công việc 1  
}  
else  
{
```

```
//Công việc 2
```

```
}
```

### 3.8.3 Câu Lệnh Switch - Case

Cú pháp

```
switch (biểu_thức)
{
    case hằng_1: [công_việc_1]
    case hằng_2: [công_việc_2]
    ...
    case hằng_n: [công_việc_n]
    default: [công_việc_n+1]
}
```

Trong đó

- Biểu thức và hằng kiểu số nguyên
- Lệnh “break” thoát khỏi “switch”

VD:

```
int chon = 1;

switch (chon) {
    case 1:
        NSLog(@"Chon 1");
        break;
    case 2:
        NSLog(@"Chon 2");
        break;

    default:
        NSLog(@"Chon 3");
        break;
}
```

Hình 3.11 Cấu trúc Switch

### 3.9 Cấu Trúc Lặp

#### 3.9.1 Vòng Lặp For

Cú pháp

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

```
for (biểu_thức1; biểu_thức2; biểu_thức3)
{
    //Công việc
}
```

VD:

```
for (int i=0; i < 10; i++) {
    NSLog(@"%@",i);
}
```

Hình 3.12 Vòng lặp For

#### 3.9.2 Vòng Lặp While

Cú pháp:

```
while (<điều_kiện_lặp>)
```

```
{
```

```
//Công việc
```

```
}
```

Trong đó:

- Khi <điều\_kiện\_lặp> còn đúng thì còn thực hiện <công\_việc>. Vòng lặp kết thúc khi <điều\_kiện\_lặp> sai.

VD:

```
int i=0;
while (i < 10) {
    NSLog(@"%@",i);
    i++;
}
```

Nguyễn Anh Tú - Coder Đam Mê Vàng © 2013

### 3.9.3 Vòng Lặp Do-While

Cú pháp:

```
do {
```

```
//Công việc lặp
```

```
} while(<điều_kiện_lặp>)
```

Trong đó:

- Thực hiện công việc ít nhất 1 lần, và lặp lại công việc khi <điều\_kiện\_lặp> là đúng.

VD:

```
int i=0;
do {
    NSLog(@"%@",i);
    i++;
} while (i < 10);
```

Hình 3.14 Vòng lặp Do - While

## 3.10 MẢNG

### 3.10.1 Định Nghĩa

Mảng là 1 tập hợp nhiều biến nhớ có cùng kiểu, gọi là kiểu phần tử của mảng, được cấp phát ở những vị trí liên tục trong bộ nhớ.

Mỗi phần tử (biến nhớ) trong mảng được xác định dựa trên tên mảng và các chỉ số xác định vị trí của phần tử trong mảng.

Chỉ số phần tử của mảng là các số nguyên không âm.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

### 3.10.2 Mảng Nsarray

Sử dụng đối tượng NSArray để khởi tạo mảng.

Cú pháp:

```
NSArray *tên_mảng;
```

VD:

```
NSArray *mangTen;
```

Để khởi tạo giá trị cho các phần tử trong mảng ta sử dụng hàm `<initWithObjects>`

VD:

```

NSArray *mangTen;
mangTen = [[NSArray alloc] initWithObjects:@"Angela Phuong Trinh", @"Elly Tran", @"Ba Tung", nil];
int count = [mangTen count];
NSLog(@"So phan tu trong mang %i", count);
NSLog(@"Phan tu vi tri 1 %@", [mangTen objectAtIndex:1]);

```

**Hình 3.15 Sử dụng NSArray**

Trong đó

- hàm <count>: đếm số phần tử trong mảng.
- alloc: cấp phát vùng nhớ cho mảng “mangTen”.
- %@: Mã kiểu dữ liệu đối với kiểu chuỗi “NSString” trong Objective-C.
- objectAtIndex: lấy ra phần tử ở vị trí index.

Kết quả

```

2013-10-11 22:29:40.779 Demo[1073:70b] So phan tu trong mang 3
2013-10-11 22:29:40.787 Demo[1073:70b] Phan tu vi tri 1 [Elly Tran]

```

**Hình 3.16 Kết quả sử dụng NSArray**

## 3.11 CHUỖI – ĐỐI TƯỢNG NSSTRING

### 3.11.1 Khởi Tạo Chuỗi

Đối với chuỗi trong Objective-C phải bắt đầu bằng @

```

@@"LacHong University"

```

**Hình 3.17 Khởi tạo chuỗi**

Để kiểm tra giá trị của chuỗi ta dùng hàm NSLog

```
 NSLog(@"%@", @"LacHong University");
```

Hình 3.18 Kiểm tra giá trị chuỗi

Đếm số lượng ký tự (chiều dài của chuỗi)

```
int len =[@"LacHong University" length];
NSLog(@"Chieu dai cua chuoi la %i", len);
```

Hình 3.19 Đếm số lượng ký tự

### 3.11.2 Đối Tượng NSString

Tạo đối tượng có kiểu là NSString để có thể sử dụng các thao tác liên quan đến chuỗi do Class NSString cung cấp (ghép chuỗi, cắt chuỗi...)

```
NSString *str = @"LacHong University";
```

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

Hình 3.20 Đối tượng NSString

### 3.11.3 Tìm Kiếm Bên Trong Chuỗi

Dùng hàm rangeOfString để tìm chuỗi “str1” có xuất hiện trong chuỗi “str2” không.

```
NSString *str1 = @"University";
NSString *str2 = @"LacHong University 2013";

NSRange match;
match = [str2 rangeOfString:str1];

if (match.location == NSNotFound) {
    NSLog(@"Khong tim thay");
} else {
    NSLog(@"Tim thay tai vi tri thu %i", match.location);
}
```

Hình 3.21 Tìm kiếm bên trong chuỗi

Kết quả

```
2013-10-11 23:20:12.266 Demo[1231:70b] Tim thay tai vi tri thu 8
```

### Hình 3.22 Kết quả tìm kiếm

#### 3.11.4 Tìm Chuỗi Và Thay Nó Thành Chuỗi Khác

Dùng hàm replaceCharactersInRange để thay đổi chuỗi thành chuỗi khác.

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong University"];
[str1 replaceCharactersInRange:NSMakeRange(8, 10) withString:@"Dai Hoc"];
NSLog(@"%@", str1);
```

### Hình 3.23 Thay đổi ký tự trong chuỗi

Kết quả:

```
2013-10-11 23:34:01.829 Demo[1358:70b] str1: LacHong Dai Hoc
```

### Hình 3.24 Kết quả thay đổi ký tự

#### 3.11.5 Xoá Nội Dung Bên Trong Chuỗi

Dùng hàm deleteCharactersInRange để xóa một nội dung bên trong chuỗi.

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong University"];
[str1 deleteCharactersInRange:[str1 rangeOfString:@"University"]];
NSLog(@"%@", str1);
```

### Hình 3.25 Xóa nội dung trong một chuỗi

Kết quả

```
2013-10-11 23:40:57.375 Demo[1407:70b] str1: LacHong
```

Hình 3.26 Kết quả sau khi xóa nội dung

### 3.11.6 Cắt Chuỗi

Cắt chuỗi có giới hạn số lượng ký tự, lấy từ ký tự thứ n.

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong University"];
NSString *str2;
str2 = [str1 substringWithRange:NSMakeRange(0, 8)];
NSLog(@"%@", str2);
```

Hình 3.27 Cắt chuỗi từ vị trí n

Kết quả

```
2013-10-11 23:45:39.594 Demo[1444:70b] str2: LacHong
```

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

Hình 3.28 Kết quả cắt chuỗi từ vị trí n

Lấy tất cả ký tự còn lại, tính từ ký tự thứ n.

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong University"];
NSString *str2;
str2 = [str1 substringFromIndex:8];
NSLog(@"%@", str2);
```

Hình 3.29 Lấy các ký tự còn lại từ vị trí n

Kết quả

```
2013-10-11 23:48:44.435 Demo[1469:70b] str2: University
```

Hình 3.30 Kết quả lấy các ký tự còn lại từ vị trí n

Tách chuỗi thành các phần nhỏ

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong-University-2013"];
NSArray *mang = [str1 componentsSeparatedByString:@"-"];
for (NSString *str in mang) {
    NSLog(@"%@", str);
}
```

Hình 3.31 Tách chuỗi thành các phần nhỏ

Kết quả

```
2013-10-11 23:53:05.002 Demo[1508:70b] LacHong
2013-10-11 23:53:05.008 Demo[1508:70b] University
2013-10-11 23:53:05.011 Demo[1508:70b] 2013
```

Hình 3.32 Kết quả tách chuỗi

### 3.11.7 Chèn Ký Tự Vào Trong Chuỗi

Dùng hàm insertString để chèn ký tự hoặc một chuỗi vào chuỗi.

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong 2013"];
[str1 insertString:@"University" atIndex:8];
NSLog(@"str1: %@", str1);
```

Hình 3.33 Chèn ký tự vào chuỗi

Kết quả

```
2013-10-11 23:59:09.045 Demo[1652:70b] str1: LacHong University 2013
```

Hình 3.34 Kết quả chèn ký tự vào chuỗi

### 3.11.8 Chèn Ký Tự Vào Cuối Chuỗi

Dùng hàm appendString để chèn ký tự vào cuối chuỗi.

```
NSMutableString *str1 = [NSMutableString stringWithString:@"LacHong University"];
[str1 appendString:@" 2013"];
NSLog(@"%@", str1);
```

**Hình 3.35 Chèn ký tự vào cuối chuỗi**

Kết quả

```
2013-10-12 00:01:09.118 Demo[1673:70b] str1: LacHong University 2013
```

**Hình 3.36 Kết quả chèn ký tự vào cuối chuỗi**

### 3.11.8 So Sánh Chuỗi

Dùng hàm isEqualToString để so sánh hai chuỗi.

```
NSString *str1 = @"LacHong 2013";
NSString *str2 = @"LacHong 2014";

if ([str1 isEqualToString:str2]) {
    NSLog(@"str1 va str2 giong nhau");
} else {
    NSLog(@"str1 va str2 khac nhau");
}
```

**Hình 3.37 So sánh hai chuỗi**

Kết quả

```
2013-10-12 00:05:31.137 Demo[1720:70b] str1 va str2 khac nhau
```

**Hình 3.38 Kết quả so sánh hai chuỗi**

### 3.11.9 So Sánh Chuỗi Với Ký Tự Đầu Và Cuối Chuỗi

Dùng hàm hasPrefix và hasSuffix để so sánh với ký tự đầu chuỗi, ký tự cuối chuỗi.

```

NSString *str1 = @"LacHong University";
BOOL result;

result = [str1 hasPrefix:@"LacHong"];
if (result) {
    NSLog(@"Chuoi bat dau bang chu \"LacHong\"");
}

result = [str1 hasSuffix:@"University"];
if (result) {
    NSLog(@"Chuoi ket thuc bang chu \"University\"");
}

```

Hình 3.39 So sánh với ký tự đầu, ký tự cuối

Kết quả

```

2013-10-12 00:11:55.600 Demo[1778:70b] Chuoi bat dau bang chu
"LacHong"
2013-10-12 00:11:55.617 Demo[1778:70b] Chuoi ket thuc bang chu
"University"

```

Hình 3.40 Kết quả so sánh với ký tự đầu chuỗi, ký tự cuối chuỗi

### 3.11.10 Chuyển Đổi Hình Dạng Của Chữ

Viết hoa chữ đầu.

```

NSString *str1 = @"lAc h0Ng uNiversity";
NSString *str2 = [str1 capitalizedString];
NSLog(@"%@", str2);

```

Hình 3.41 Viết hoa ký tự đầu

Kết quả

**2013-10-12 00:19:37.874 Demo[1878:70b] str2: Lac Hong University**

### Hình 3.42 Kết quả viết hoa ký tự đầu

Tất cả viết thường.

```
NSString *str1 = @"LacHong University";
NSString *str2 = [str1 lowercaseString];
NSLog(@"%@", str2);
```

### Hình 3.43 Viết thường tất cả ký tự

Kết quả

**2013-10-12 00:21:25.141 Demo[1895:70b] str2: lachong university**

Nguyễn Anh Tài © Nguyễn Văn Hùng © 2013

Tất cả viết hoa.

```
NSString *str1 = @"LacHong University";
NSString *str2 = [str1 uppercaseString];
NSLog(@"%@", str2);
```

### Hình 3.45 Viết hoa tất cả ký tự

Kết quả

**2013-10-12 00:22:22.760 Demo[1913:70b] str2: LACHONG UNIVERSITY**

### Hình 3.46 Kết quả viết hoa các ký tự

### 3.11.11 Chuyển Chuỗi Thành Dạng Số

Chuyển thành số nguyên

```
NSString *str1 = @"10";
int i = [str1 intValue];
NSLog(@"%@", i);
```

Hình 3.47 Chuyển thành số nguyên

Kết quả

```
2013-10-12 00:24:37.872 Demo[1935:70b] 10
```

Hình 3.48 Kết quả chuyển thành số nguyên

Chuyển thành NSInteger

```
NSString *str1 = @"10";
NSInteger i = [str1 integerValue];
NSLog(@"%@", i);
```

Hình 3.49 Chuyển thành NSInteger

Kết quả

```
2013-10-12 00:35:21.132 Demo[2054:70b] 10
```

Hình 3.50 Kết quả chuyển thành NSInteger

Chuyển thành float

```
NSString *str1 = @"10.242";
float i = [str1 floatValue];
NSLog(@"%@", i);
```

Hình 3.51 Chuyển thành float

Kết quả

```
2013-10-12 00:36:30.301 Demo[2074:70b] 10.242
```

Hình 3.52 Kết quả chuyển thành float

Chuyển thành số double

```
NSString *str1 = @"10.242";
double i = [str1 doubleValue];
NSLog(@"%@", i);
```

Hình 3.53 Chuyển thành double

Kết quả

```
2013-10-12 00:27:01.350 Demo[1997:70b] 10.242
```

Hình 3.54 Kết quả chuyển thành double

## CHƯƠNG IV

### MỘT SỐ THAO TÁC CƠ BẢN

Trước khi bắt đầu lập trình ứng dụng trên iPhone, ngoài việc bạn tìm hiểu về Xcode và iOS Simulator, bạn còn phải tìm hiểu thêm một số thao tác cơ bản trong lập trình iOS. Hiểu biết về các thao tác cơ bản này sẽ hỗ trợ cho bạn trong quá trình tìm hiểu, cũng như là bài học vỡ lòng trước khi đi sâu vào tìm hiểu các đối tượng cơ bản trong Xcode ở chương sau. Nội dung chương này gồm một số thao tác cơ bản như:

- ❖ Thay đổi icon của ứng dụng
- ❖ Thay đổi màn hình khi ứng dụng vừa được mở lên
- ❖ Thay đổi tên ứng dụng khi tên cũ quá dài
- ❖ Tùy chỉnh background
- ❖ Thêm mới một framework
- ❖ Ân thanh trạng thái status bar

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

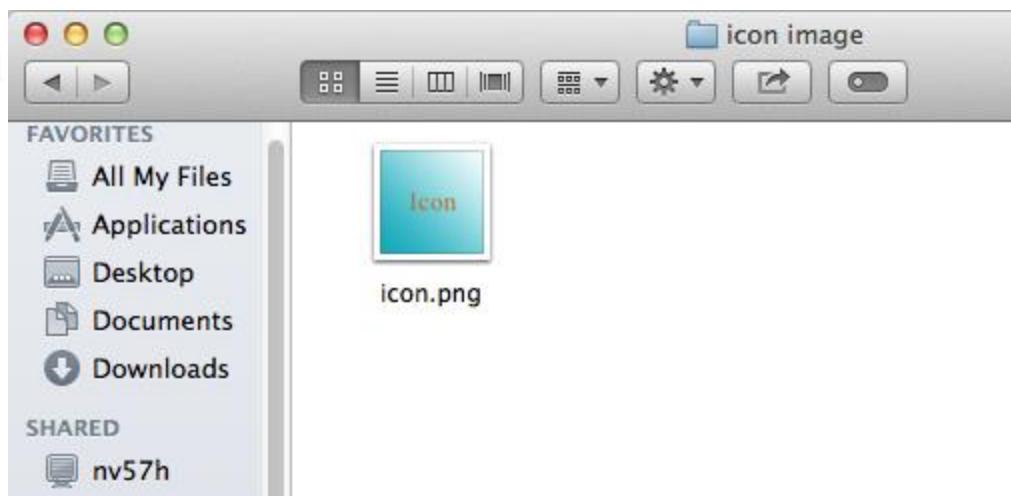
## 4.1 APP ICON – LOADING SCREEN

Phần này sẽ hướng dẫn bạn cách hiển thị cũng như tạo icon, background image, loading screen cho ứng dụng của bạn. App icon, background image, loading screen cho ứng dụng của bạn cần dựa theo kích thước quy định của Apple và nên ở định dạng PNG. Bạn có thể tham khảo thêm kích thước chuẩn ở đây

[https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#/apple\\_ref/doc/uid/TP40006556-CH27-SW1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html#/apple_ref/doc/uid/TP40006556-CH27-SW1).

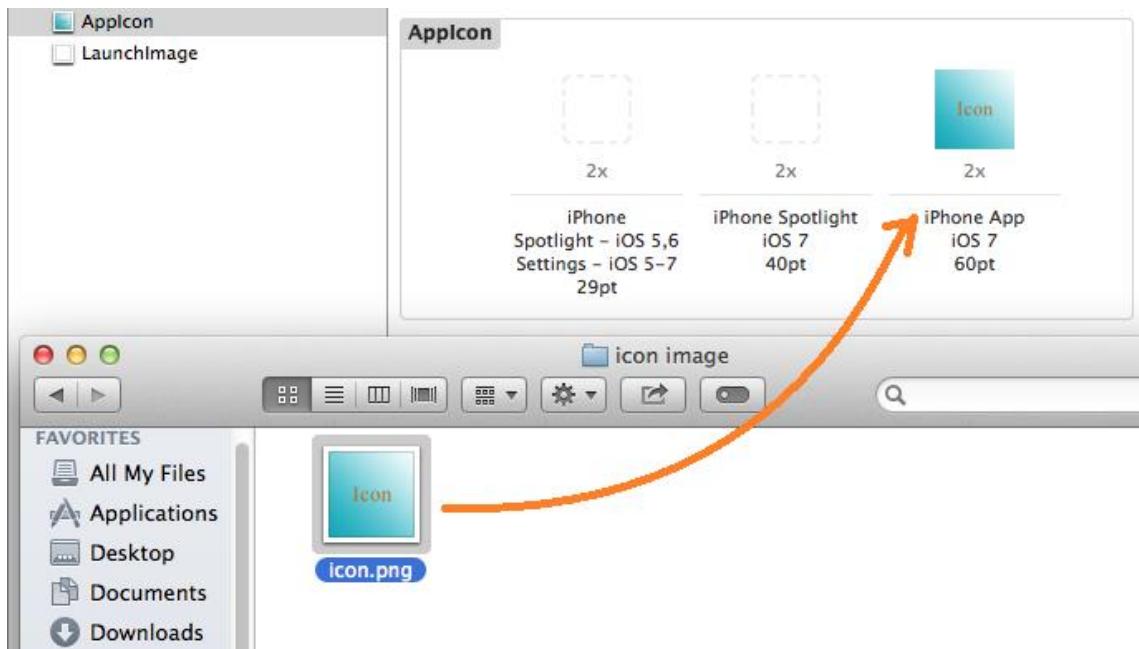
### 4.1.1 App Icon

App icon là biểu tượng của ứng dụng sẽ hiển thị ra màn hình iPhone sau khi ứng dụng được cài đặt, đồng thời icon của ứng dụng cũng hiển thị lên App Store khi bạn đem ứng dụng của mình lên App Store. Có hai loại icon là icon dành cho ứng dụng trên iPhone và icon dành cho ứng dụng trên App Store. App icon cho iPhone Retina là 120x120, App icon cho App Store là 1024x1024.



Hình 4.1 Icon

Sau khi chuẩn bị được hình ảnh với kích thước phù hợp, trong phần thông tin của **Project**, bạn kéo xuống phần **App icon** và kéo các icon chuẩn bị sẵn vào đúng vị trí của icon.



**Hình 4.2 Kéo thả icon vào vị trí**

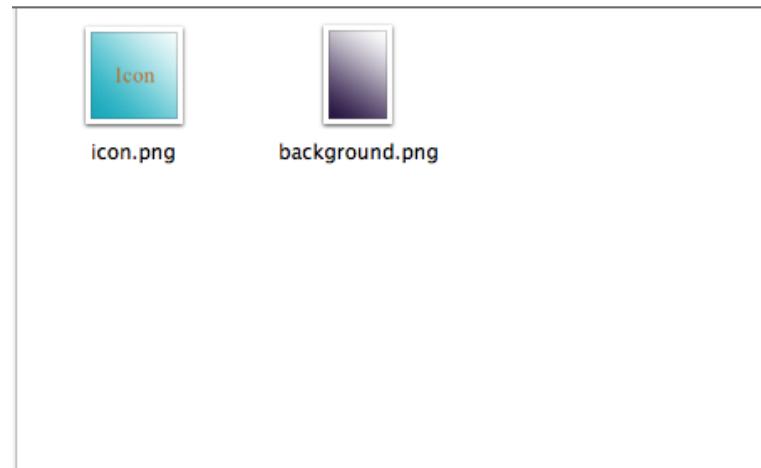
Bạn tiến hành chạy thử ứng dụng trên iOS Simulator sẽ thấy icon của ứng dụng.



**Hình 4.3 Icon trên iOS simulator**

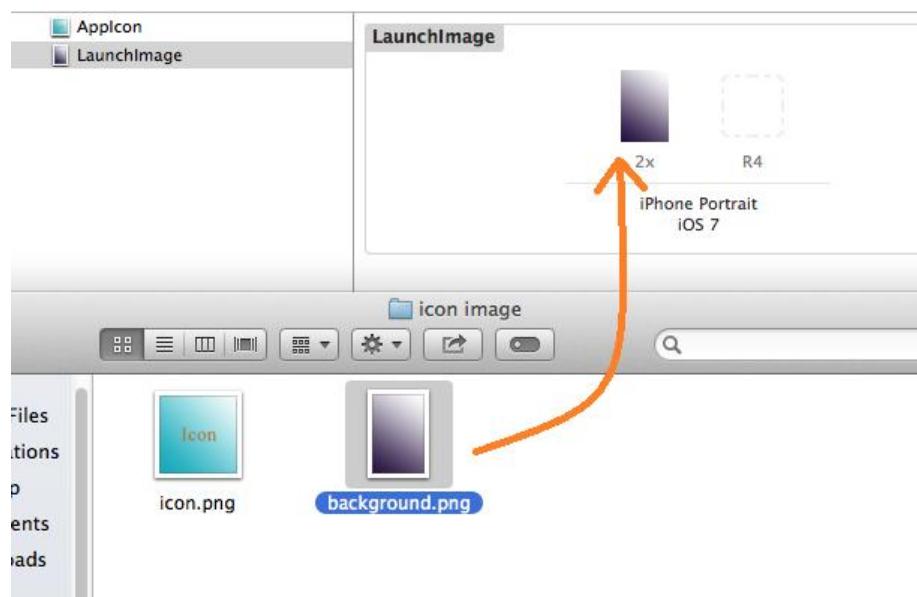
#### 4.1.2 Loading Screen

Loading Screen là hình ảnh mà khi chạy ứng dụng lên bạn sẽ thấy nó, tùy theo tốc độ của ứng dụng mà thời gian hiển thị Loading Screen nhanh hoặc chậm khác nhau. Để chuẩn bị cho Loading Screen bạn cần chuẩn bị trước hình ảnh dạng PNG, kích thước 640x1136 (iPhone 5 trở lên), 640x960 (iPhone 4).



Hình 4.4 Chuẩn bị Background

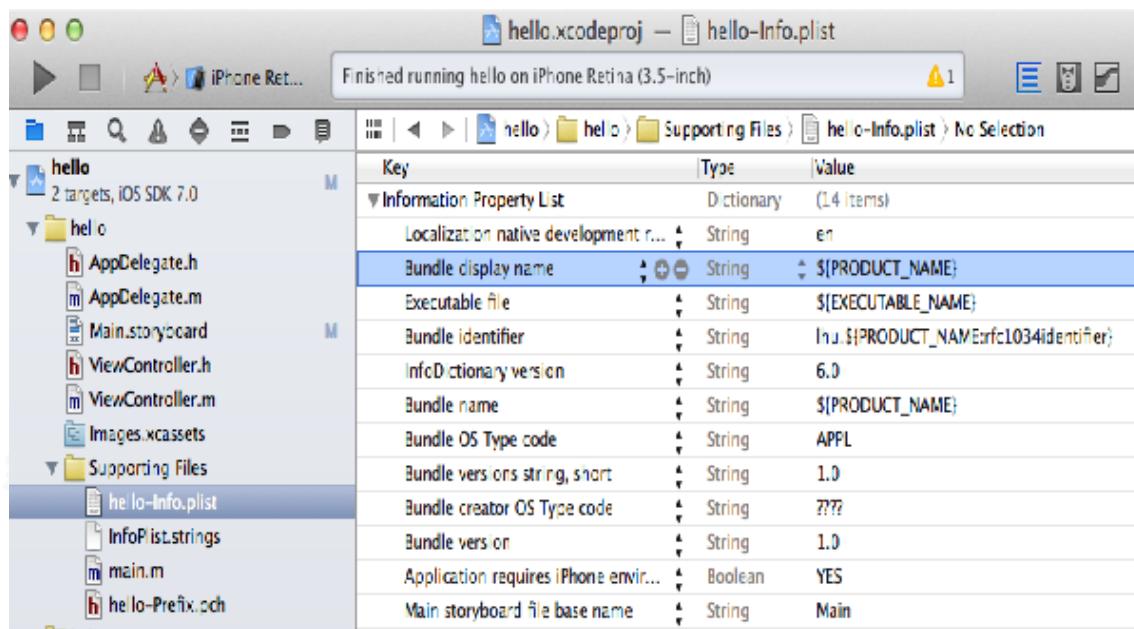
Sau khi chuẩn bị xong hình ảnh, trong phần **Launch Image**, bạn kéo thả hình ảnh vào Xcode cho đúng vị trí.



Hình 4.5 Kéo thả background image vào Lauch Image

## 4.2 THAY ĐỔI APP NAME

App Name là tên của ứng dụng hiển thị bên dưới App icon trên màn hình iPhone. Nhiều trường hợp vì tên ứng dụng quá dài nên không thể hiển thị hết trên màn hình iPhone, do đó bạn cần phải đổi tên ứng dụng lại cho phù hợp, ngắn gọn để có thể hiện được tên ứng dụng bên dưới icon, như vậy ứng dụng sẽ có tính thẩm mỹ hơn. Để khắc phục trường hợp này, bạn cần phải thay đổi lại App Name. Bạn truy cập tập tin .plist và sửa đổi Bundle Display Name thành tên mới của ứng dụng.

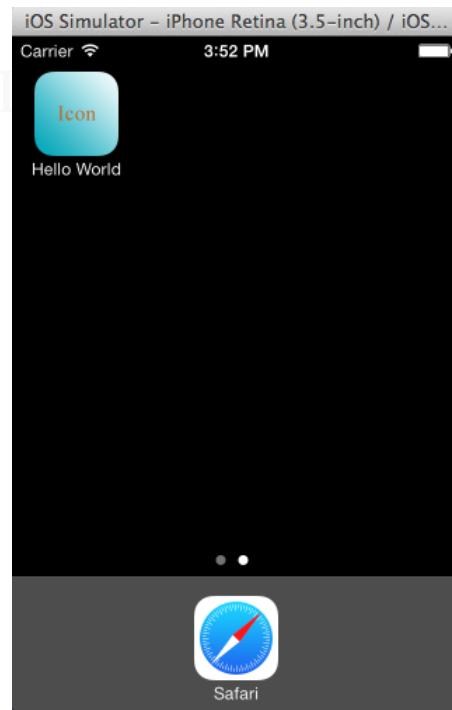


Hình 4.6 Truy cập tập tin .plist

hello.xcodeproj — hello-Info.plist		
Finished running hello on iPhone Retina (3.5-inch)		
Key   Type   Value		
▼ Information Property List	Dictionary	(14 items)
Localization native development r...	String	en
Bundle display name	String	Hello World
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	lhu.\$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone envir...	Boolean	YES
Main storyboard file base name	String	Main
► Required device capabilities	Array	(1 item)
► Supported interface orientations	Array	(3 items)

**Hình 4.7 Sửa đổi Bundle display name**

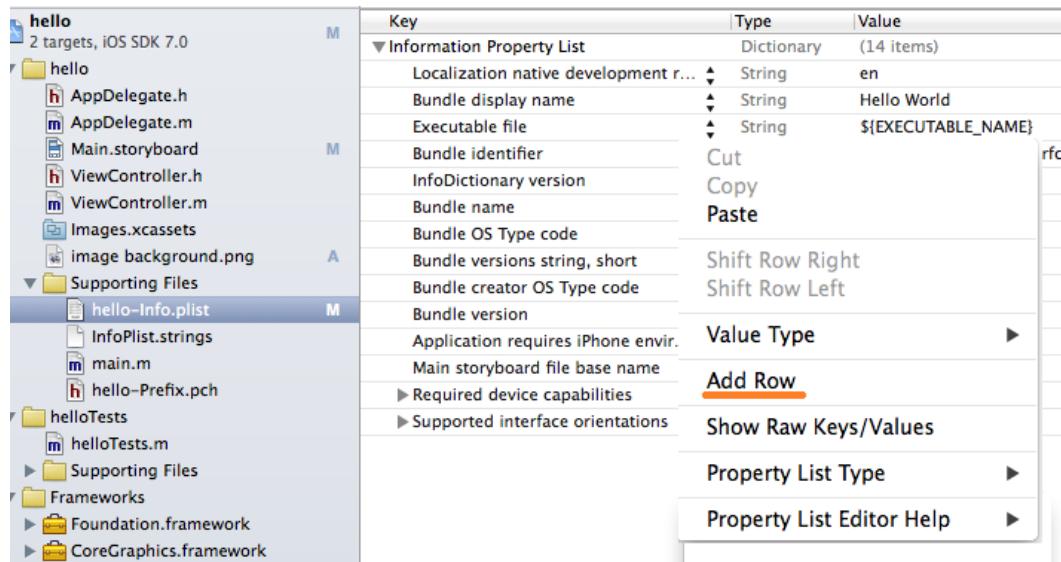
Sau khi hoàn tất, bạn chạy ứng dụng trên iOS Simulator để xem kết quả.



**Hình 4.8 Thay đổi tên trên iOS Simulator**

## 4.3 ẨN STATUS BAR

Việc ẩn status bar cho phép bạn chạy ứng dụng của mình toàn màn hình mà không phải thu nhỏ một phần giao diện dành chõ cho status bar. Trước tiên bạn chọn tập tin .plist trong project, sau đó bạn thêm một dòng mới.



Nguyễn Anh Tú | Hình 4.9 Thêm dòng mới

Trong danh sách lựa chọn của dòng mới thêm vào, bạn lựa chọn Status bar is initially hidden và trả về giá trị là yes.

Localization native development r...	String	en
Bundle display name	String	Hello World
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	lhu.\$(PRODUCT_NAME):rfc1034identifier
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone envir...	Boolean	YES
Main storyboard file base name	String	Main
► Required device capabilities	Array	(1 item)
Status bar is initially hidden	Boolean	YES
► Supported interface orientations	Array	(3 items)

Nguyễn Anh Tú | Hình 4.10 Chọn giá trị Yes

Chạy ứng dụng trên iOS Simulator để xem kết quả.



**Hình 4.11** Thanh trạng thái Status bar đã mất

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

## 4.4 BACKGROUND

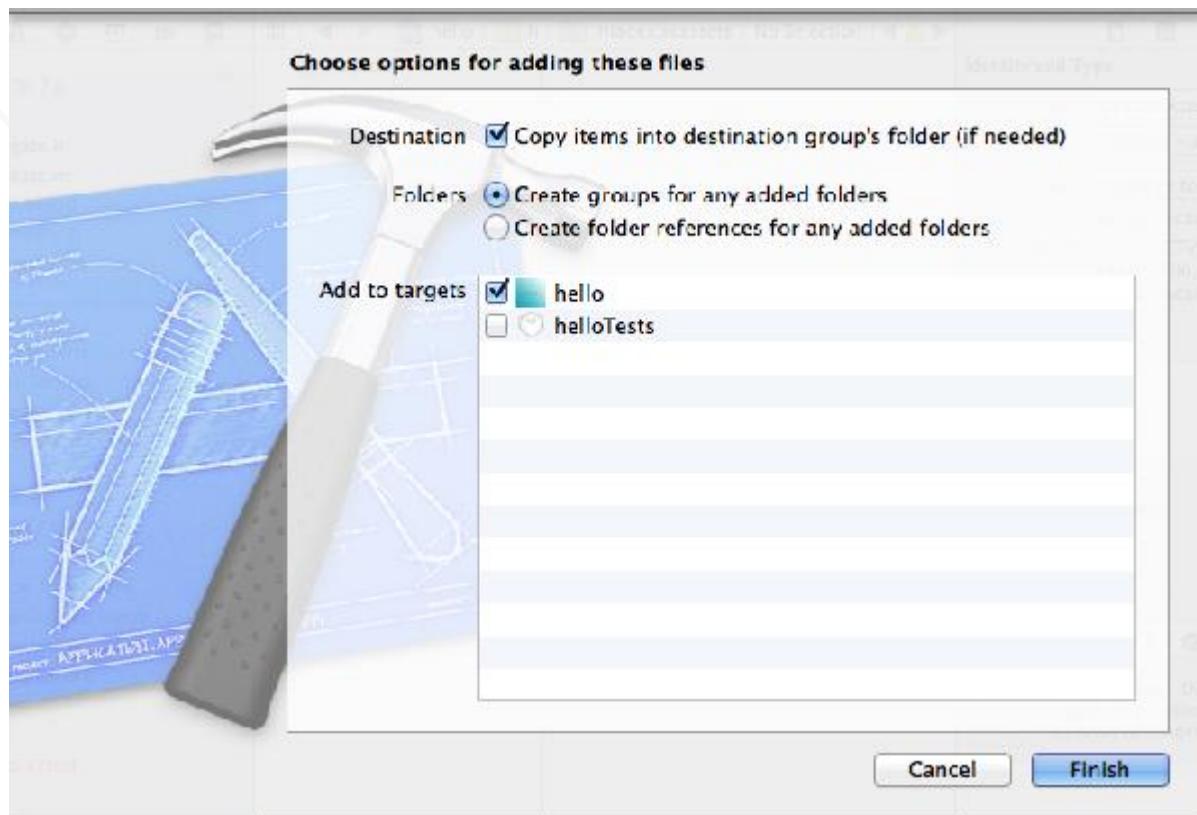
### 4.4.1 Background Image

Hướng dẫn này sẽ hướng dẫn bạn tạo và sử dụng một hình ảnh làm hình nền cho ứng dụng của bạn. Trước tiên bạn cần phải chuẩn bị một tấm ảnh để làm hình nền. Kích thước của tấm ảnh này giống như hình ảnh trong Loading Screen: định dạng PNG, kích thước 640x1136 (iPhone 5 trở lên), 640x960 (iPhone 4).



**Hình 4.12 Background Image**

Sau khi chuẩn bị xong hình ảnh để làm background cho ứng dụng, bạn chép hình ảnh vào trong project.



**Hình 4.13 Chép hình ảnh vào Project**

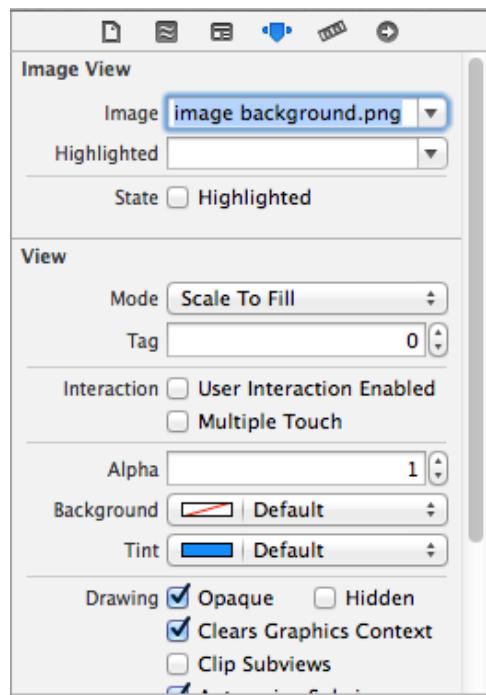
Tiếp theo bạn từ Utility area, bạn kéo thả đối tượng UIImageView vào Interface Builder, điều chỉnh kích thước UIImageView cho vừa với màn hình.



Nguyễn Anh Tú | Vàng © 2013

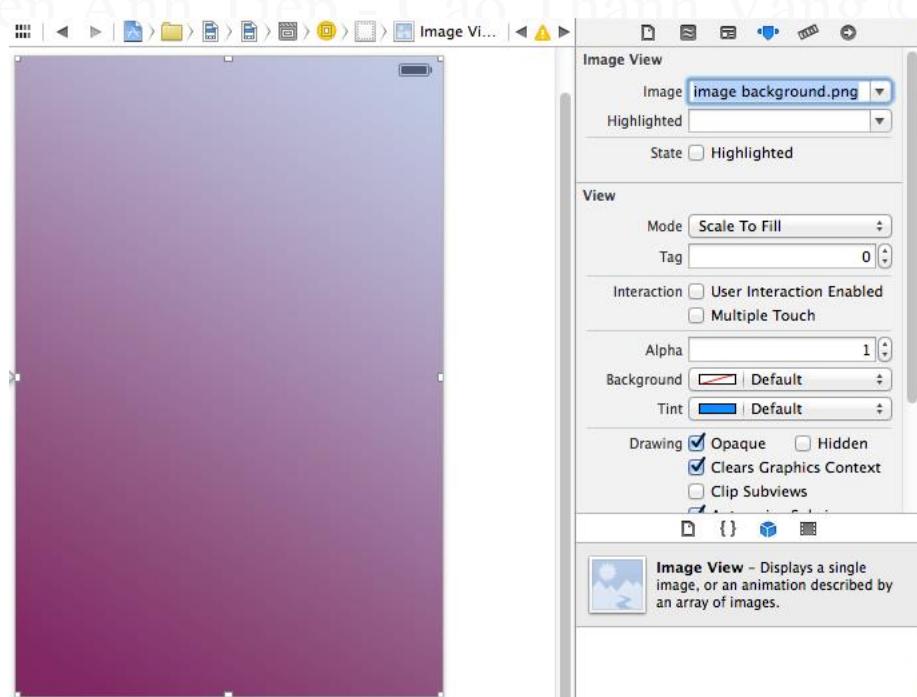
**Hình 4.14 Kéo thả UIImageView vào Project**

Trong **Inspector selector pane > Attributes inspector**, bạn tìm đến mục Image View > chọn Image > lựa chọn hình ảnh bạn muốn làm background.



Hình 4.15 Lựa chọn hình ảnh làm background

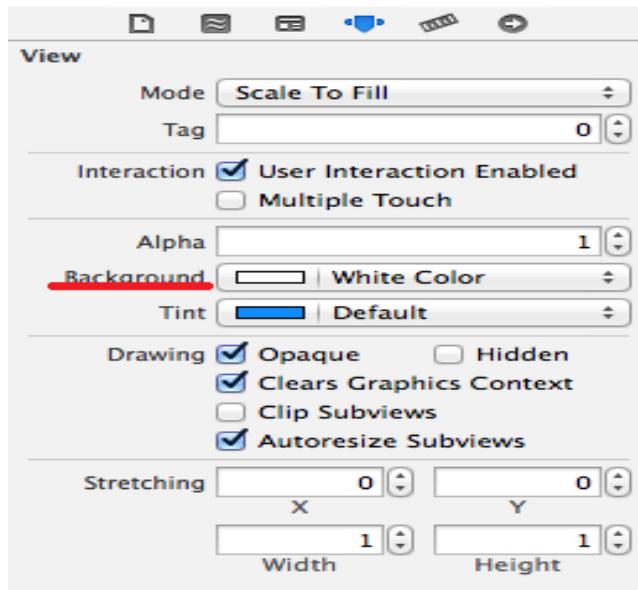
Kết quả.



Hình 4.16 Thay đổi background bằng image

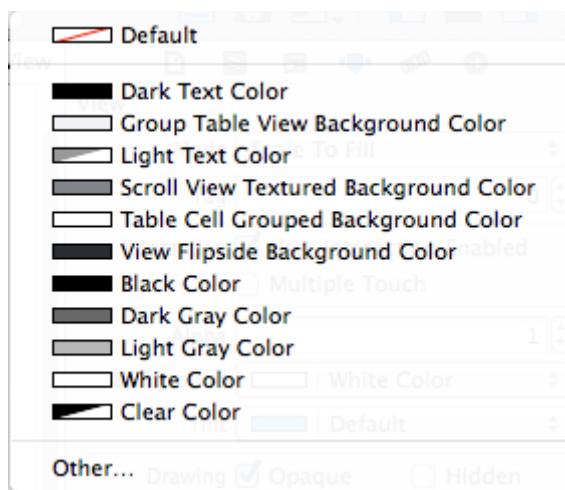
#### 4.4.2 Background Color

Ngoài việc sử dụng hình ảnh làm background cho ứng dụng, nếu bạn yêu thích sự đơn giản, bạn có thể tạo background bằng cách dùng một màu mà bạn ưa thích. Trong Inspector selector pane, bạn chọn **Attributes inspector**, tìm đến phần **Background**.



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
**Hình 4.17 Attributes inspector**

Tại đây bạn cho hiện ra bảng màu để chọn lựa với nhiều màu khác nhau, hãy lựa chọn cho mình một màu phù hợp, nếu muốn nhiều màu hơn có thể chọn **Other**.



**Hình 4.18 Chọn màu**

Sau khi chọn xong màu thì background sẽ đổi màu với màu bạn đã lựa chọn.



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

Hình 4.19 Background thay đổi

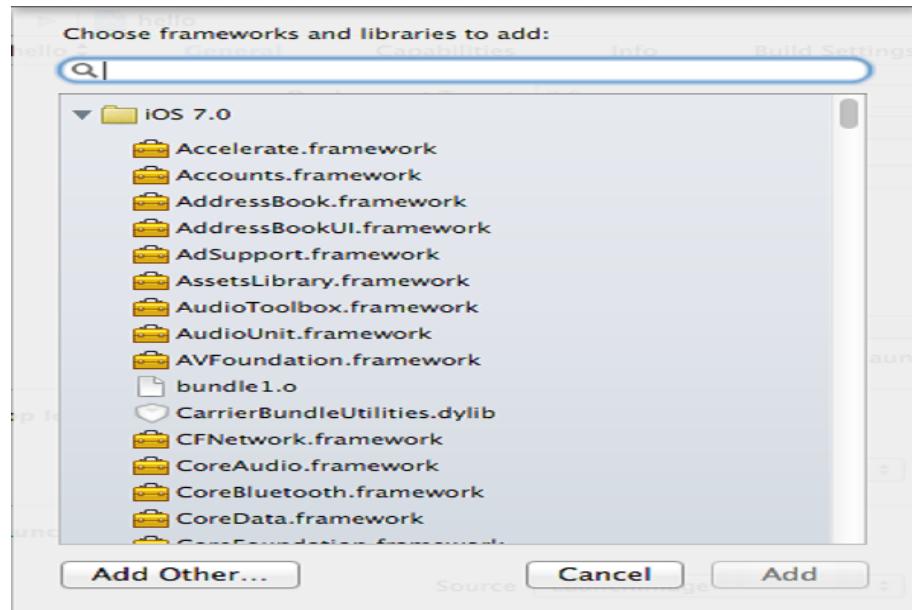
## 4.5 THÊM FRAMEWORK

Mặc dù Xcode hỗ trợ nhiều Framework hỗ trợ cho các nhà phát triển ứng dụng, tuy nhiên chỉ một số framework cơ bản được thêm vào khi tạo project, các framework còn lại thì khi viết ứng dụng, bạn phải tự thêm vào để có thể sử dụng framework đó. Để tiến hành thêm một framework, trong giao diện **General** của **Project**, bạn kéo xuống tới mục **Link Frameworks and Libraries**.

▼ Linked Frameworks and Libraries	
Name	Status
CoreGraphics.framework	Required ▲
UIKit.framework	Required ▲
Foundation.framework	Required ▲
+	-

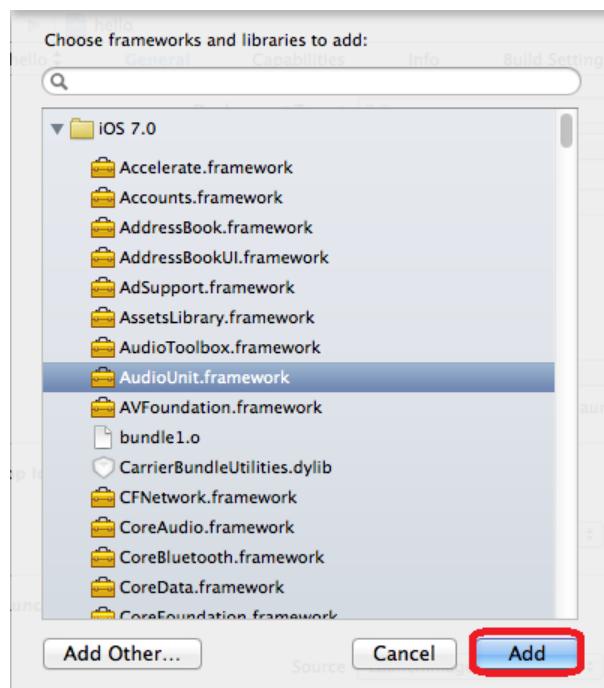
Hình 4.20 Link Frameworks and Libraries

Bạn nhấp chuột vào biểu tượng dấu cộng để mở ra bảng danh mục các framework mà Xcode hỗ trợ.



**Hình 4.21 Chọn Framework**

Sau đó bạn lựa chọn framework mà bạn muốn thêm vào project rồi nhấn **Add**.



**Hình 4.22 chọn Add**

## CHƯƠNG V MỘT SỐ ĐỐI TƯỢNG CƠ BẢN

Chương này trình bày một số đối tượng cơ bản thường được sử dụng để viết ứng dụng cho iOS. Qua chương này, bạn có thể nắm được chức năng cũng như cách sử dụng một số đối tượng cơ bản, từ đó có thể vận dụng vào viết ứng dụng. Do khả năng của người viết còn giới hạn nên số đối tượng được giới thiệu trong chương này vẫn còn hạn chế. Bạn có thể tìm hiểu thêm nhiều đối tượng khác, tìm hiểu sâu hơn từng đối tượng bằng cách truy cập vào hướng dẫn của Apple cho người lập trình ở địa chỉ sau:

<https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/index.html>

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## 5.1 ĐỐI TƯỢNG LABEL – BUTTON – TEXT FIELD

### 5.1.1 Giới Thiệu

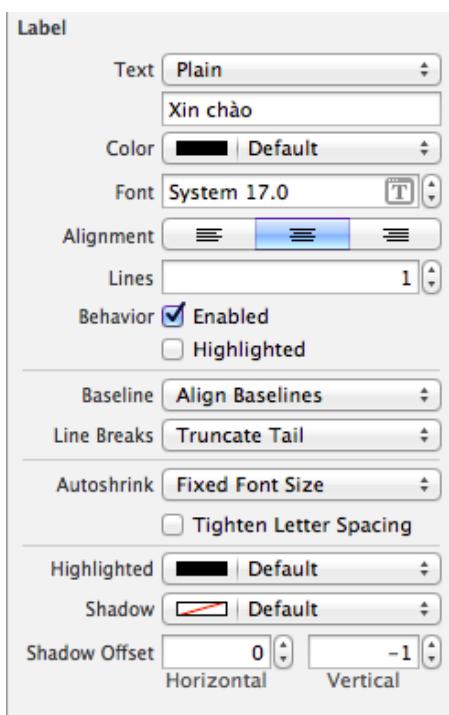
#### 5.1.1.1 Đối Tượng Label

Label dùng để hiển thị một nội dung là chữ/câu/đoạn văn có tính cố định ít thay đổi.



**Hình 5.1 Label**

Label cho phép người dùng thay đổi một số thuộc tính để hiển thị nội dung ra giao diện cho phù hợp như tùy chỉnh màu chữ, font và kích thước, canh trái/giữa/phải hay cho nội dung của label xuống dòng.



**Hình 5.2 Thuộc tính của Label**

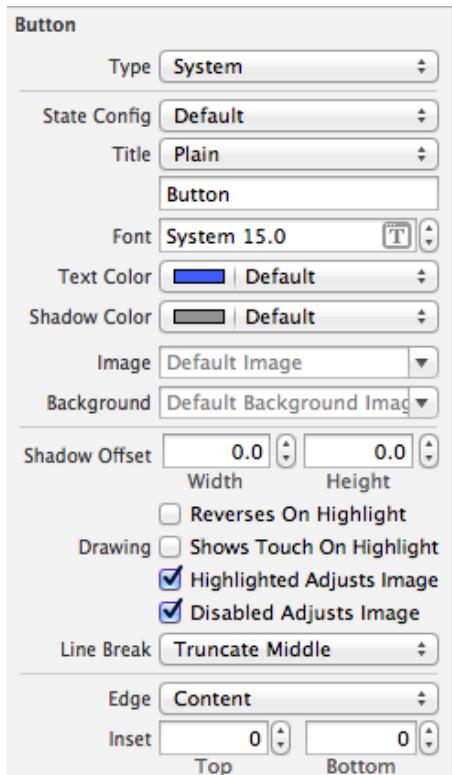
### **5.1.1.2 Đối Tượng Button**

Button có thể được thiết kế hiển thị chữ hoặc hiển thị theo hình ảnh, thường được sử dụng để người dùng tương tác với ứng dụng nhằm tạo ra một sự kiện nào đó của ứng dụng. Chẳng hạn bạn nhập tên của bạn vào text field, bạn muốn khi người dùng chạm vào button sẽ hiện tên bạn ra màn hình. Lúc này button sẽ được thiết kế để nắm bắt sự kiện khi người dùng chạm vào, và thực hiện chức năng hiển thị tên bạn ra màn hình.



**Hình 5.3 Button**

Với button, bạn có thể lựa chọn một số định dạng button có sẵn, hoặc tùy chọn một dạng button khác theo ý bạn. Không những thế bạn còn có thể tùy chỉnh font chữ, màu chữ, màu nền của button hoặc thay thế button theo một hình ảnh button được thiết kế trước.



**Hình 5.4 Thuộc tính của Button**

## Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

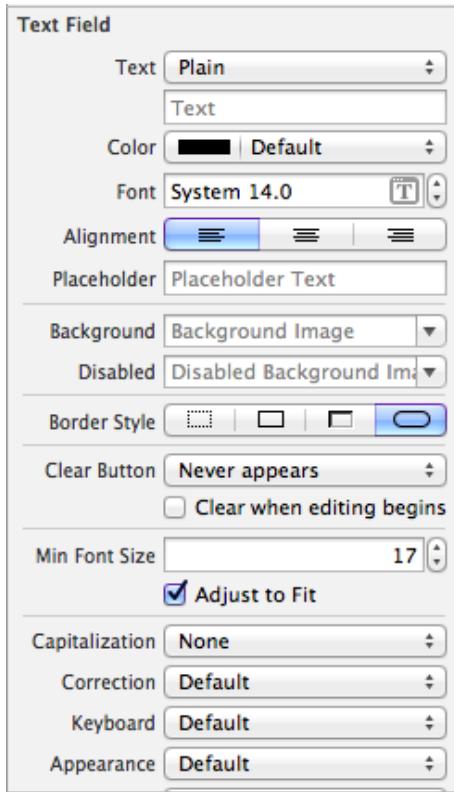
### 5.1.1.3 Đối Tượng Text Field

Đối tượng Text field thường được sử dụng để người dùng nhập dữ liệu đầu vào cho ứng dụng. Chẳng hạn như bạn viết chương trình tính tổng hai số, thì bạn sẽ dùng Text field để người dùng nhập vào hai số cần tính tổng, và bạn sẽ sử dụng giá trị nhập vào của Text field để tính toán và hiển thị kết quả cho người dùng.



**Hình 5.5 Text Field**

Text field cho phép tùy chỉnh canh lề trái/giữa/phải, tùy chỉnh font chữ và kích thước chữ trong Text field. Hơn nữa, bạn có thể tùy chỉnh hình dạng của Text field, cũng như định dạng nội dung gợi ý cho người dùng (placeholder).



**Hình 5.6 Thuộc tính của Text Field**

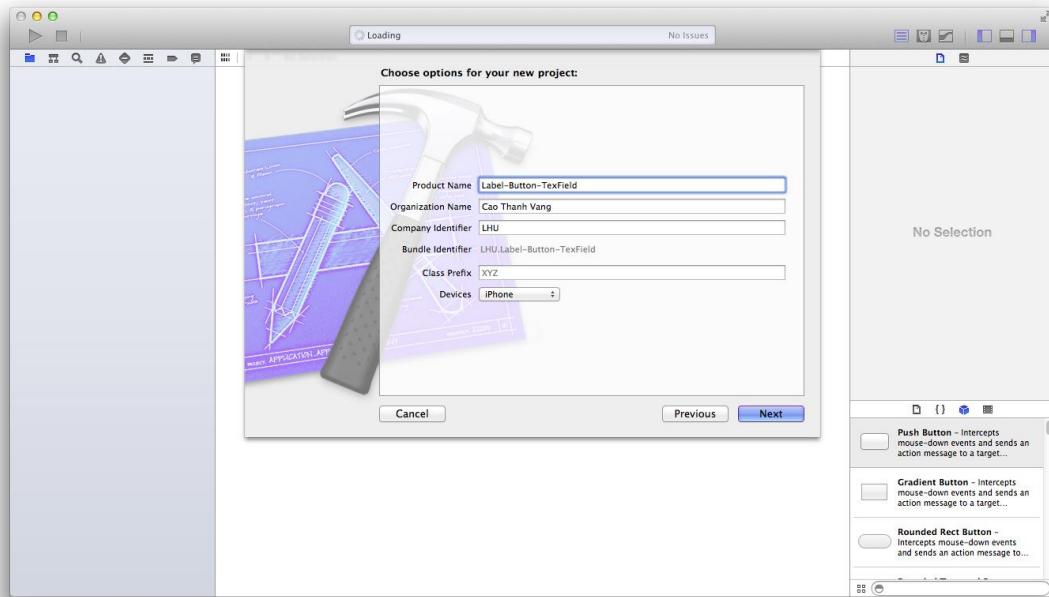
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

### 5.1.2 Ví Dụ

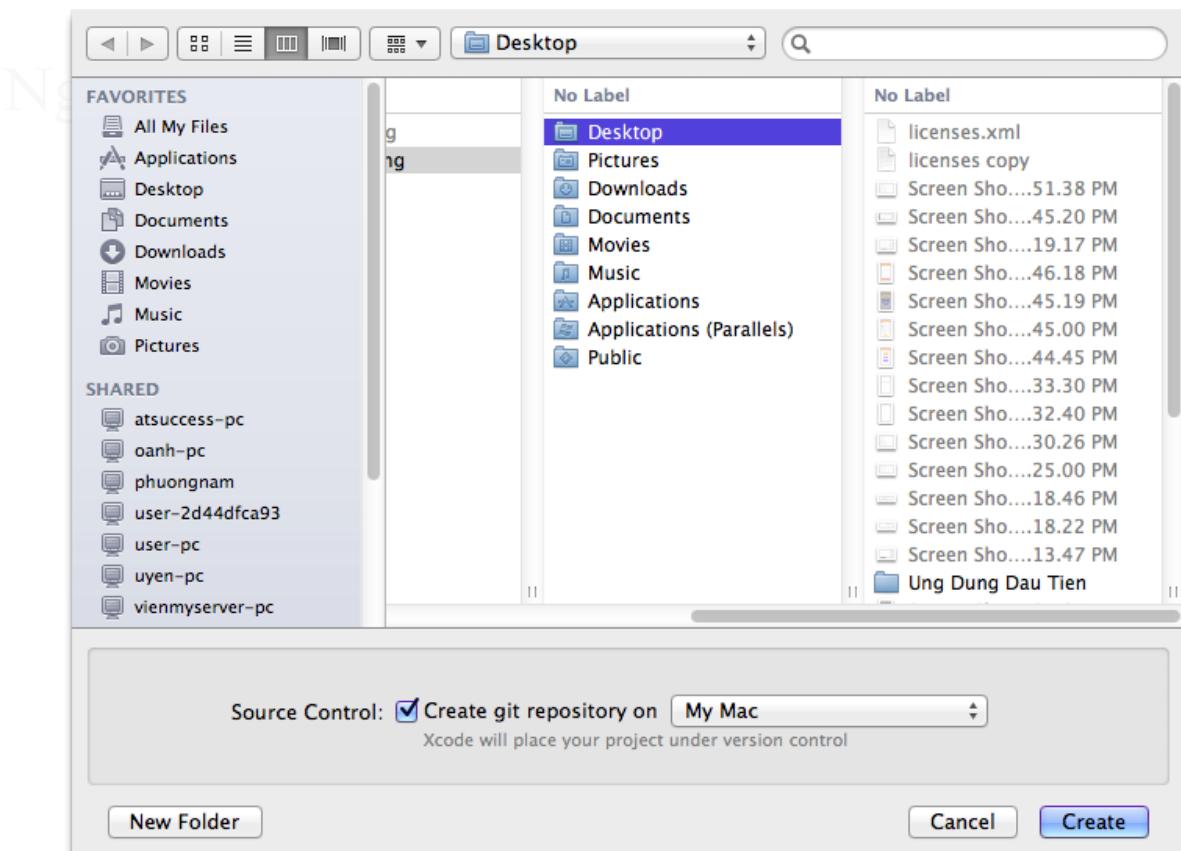
Trong phần ví dụ này sẽ hướng dẫn các bạn viết một ứng dụng nhỏ để hiển thị tên của bạn ra màn hình. Ứng dụng sẽ có một label, một button, một text field. Bạn sẽ nhập tên bạn vào text field, sau đó chạm vào button, label sẽ hiển thị nội dung mà bạn đã nhập vào text field trước đó.

- Tạo ứng dụng có tên là Label-Button-TextField.
- Thiết kế giao diện cho ứng dụng gồm một label, một button, một text field.
- Ánh xạ các đối tượng vào tập tin .h.
- Viết code cho sự kiện khi người dùng chạm vào button.

**Bước 1:** Tạo project mới.

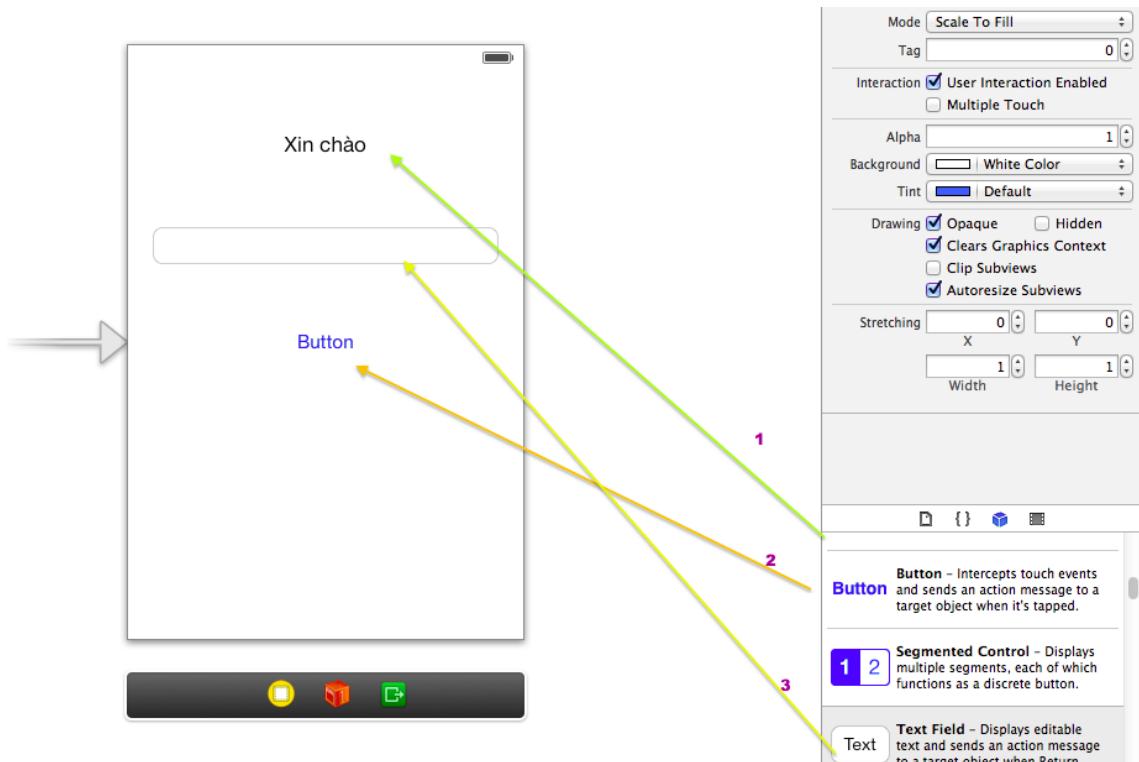


Hình 5.7 Tạo New Project



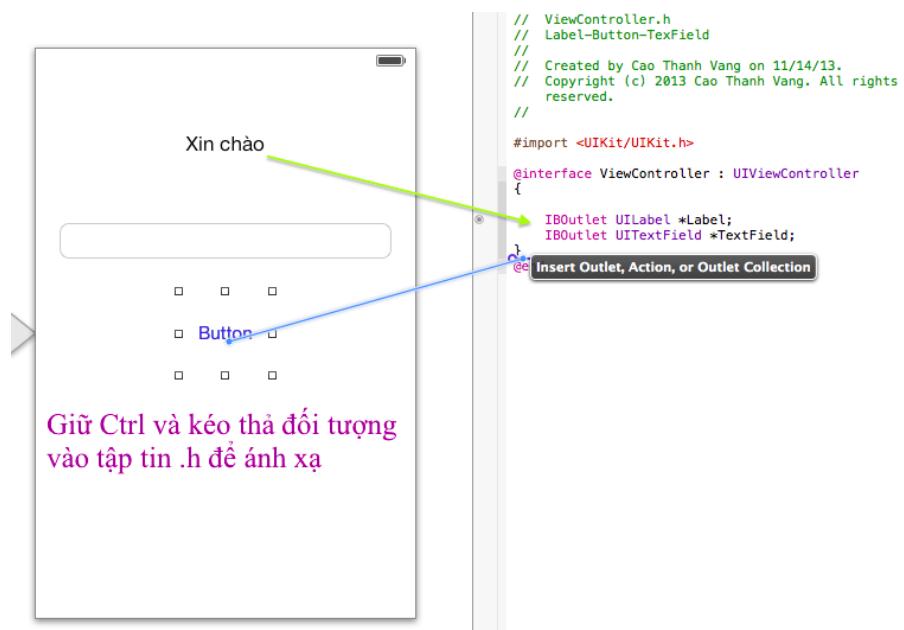
Hình 5.8 Chọn nơi lưu project

## Bước 2: thiết kế giao diện

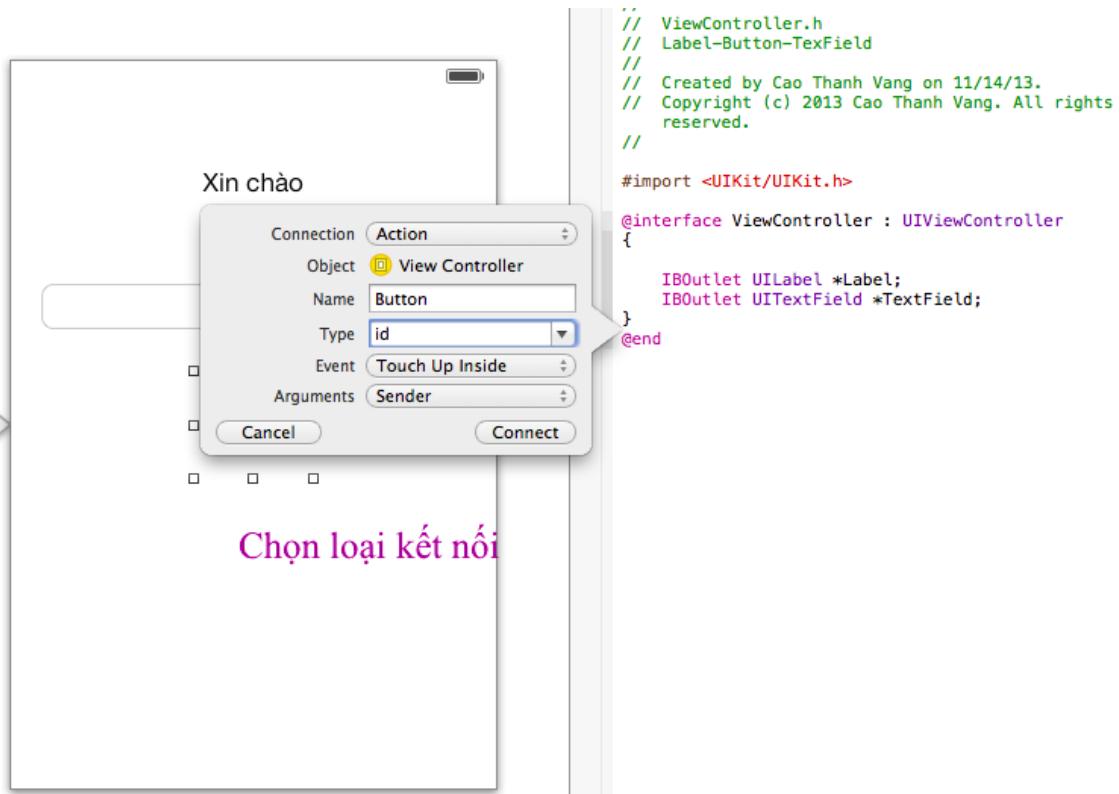


Nguyễn Anh Tín © Cao Thành Vàng © 2013

## Bước 3: ánh xạ đối tượng.



Hình 5.10 ánh xạ đối tượng



**Hình 5.11 ánh xạ đối tượng (tt)**

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

**Bước 4:** viết code cho button.

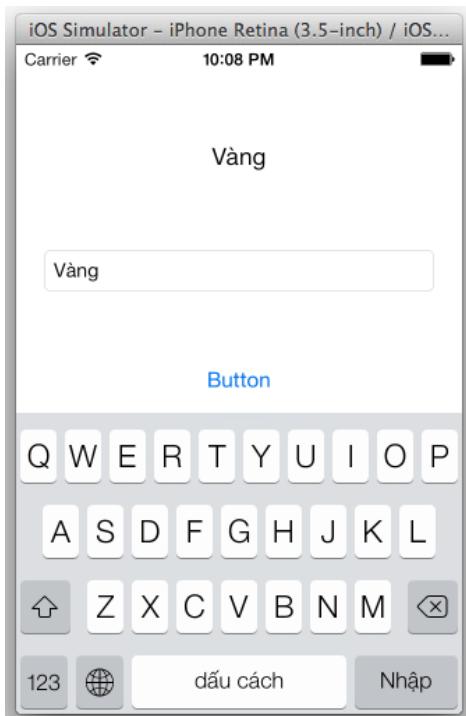
```

@IBAction Button:(id)sender {
    Label.text = TextField.text;
}
@end
|

```

**Hình 5.12** viết code cho button

**Bước 5:** chạy thử



**Hình 5.13 chạy thử ứng dụng**

## 5.2 KẾT NỐI CƠ SỞ DỮ LIỆU VỚI SQLITE

### 5.2.1 Giới Thiệu

SQLite là một hệ quản trị cơ sở dữ liệu có thể chạy hoàn toàn độc lập mà không cần đến server. SQLite thường được người lập trình sử dụng để lưu trữ cơ sở dữ liệu khi viết ứng dụng cho các thiết bị của Apple. Các lệnh truy vấn trên SQLite sử dụng các lệnh truy vấn của SQL (ví dụ SELECT, UPDATE, CREATE...). Việc quản lý SQLite rất đơn giản, bạn chỉ cần quản lý thông qua một plugin của FireFox là SQLite Manager. Bạn có thể tìm hiểu thêm tại <http://www.sqlite.org>.

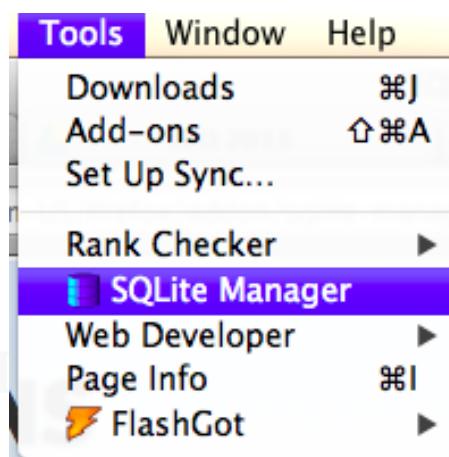
### 5.2.2 Cài Đặt Sqlite Manager Cho Firefox

Để cài đặt plugin quản lý SQLite cho Firefox, bạn vào phần Addon của trình duyệt Firefox và tìm plugin SQLite Manager rồi Add to Firefox.



**Hình 5.14 Addon SQLite**

Sau khi cài đặt xong, trong phần Tool bạn sẽ thấy như hình.



**Hình 5.15 SQLite Manager trong Tool**

### 5.2.3 Cấu Hình Ứng Dụng Để Tương Tác Với Sqlite

Để ứng dụng có thể thao tác với cơ sở dữ liệu của SQLite, bạn cần bổ sung thêm thư viện hỗ trợ vào project. Trong phần Build Phase, mục Link to Library, bạn thêm vào thư viện libsqlite3.dylib vào project.

Name	Status
libsqlite3.dylib	Required
CoreGraphics.framework	Required
UIKit.framework	Required
Foundation.framework	Required

**Hình 5.16 Thêm framework hỗ trợ SQLite**

#### 5.2.4 Các Hàm Trong Sqlite

Trong SQLite có một số hàm cơ bản cho phép bạn tương tác dễ dàng với cơ sở dữ liệu.

- **sqlite3\_open()**: mở một kết nối đến tập tin sqlite. Nếu tập tin này chưa có, hệ thống sẽ tự động tạo ra.
- **sqlite3\_close()**: đóng kết nối đến sqlite.
- **sqlite3\_prepare\_v2()**: khởi tạo câu lệnh truy vấn SQL để thực thi.
- **sqlite3\_step()**: thực thi lệnh truy vấn được tạo bởi hàm sqlite3\_prepare\_v2().
- **sqlite3\_column\_<type>()**: trả về kết quả dữ liệu từ câu truy vấn SQL, với kiểu dữ liệu được khai báo trong <type> (thường là text, bytes, int, int16).
- **sqlite3\_finalize()**: xoá câu lệnh truy vấn SQL được khởi tạo bởi hàm sqlite3\_prepare\_v2() trong bộ nhớ.

Bạn có thể tham khảo thêm một số hàm khác tại: [www.sqlite.org/c3ref/funclist.html](http://www.sqlite.org/c3ref/funclist.html)

##### 5.2.4.1 Khởi Tạo Đối Tượng Sqlite – Cao Thành Vàng © 2013

Trước khi tương tác với cơ sở dữ liệu SQLite, bạn cần phải tạo đối tượng dạng này bằng cách khai báo một biến có kiểu sqlite3 trong tập tin .h.

```
|  
| sqlite3 *contactDB;
```

Hình 5.17 Khai báo sqlite3

##### 5.2.4.2 Kết Nối Hoặc Tạo Database

Dùng hàm sqlite3\_open() để mở kết nối đến cơ sở dữ liệu sqlite.

```
-----  
if(sqlite3_open(dbpath, &contactDB)==SQLITE_OK)  
{  
    ...  
}
```

Hình 5.18 Hàm sqlite3\_open

Nếu tập tin này chưa có thì sẽ tự động được tạo.

**Filename:** là đường dẫn đến tập tin sqlite. Nếu tên database có dấu tiếng việt thì cần chuẩn hoá lại theo dạng UTF-8 trước khi truyền vào.

Sử dụng biến SQLITE\_OK để kiểm tra trạng thái trả về của việc mở kết nối đến database có thành công hay không.

#### 5.2.4.3 Khởi Tạo Và Thực Thi Lệnh Truy Vấn

Câu lệnh truy vấn được khởi tạo và lưu trữ vào đối tượng sqlite\_stmt() và truyền vào hàm sqlite\_prepare\_v2() để thực thi.

```
sqlite3_stmt *statement;
```

**Hình 5.19 khai báo đối tượng sqlite\_stmt**

```
//-----
if (sqlite3_prepare_v2(contactDB,
                        query_stmt, -1, &statement, nil) == SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        kq = [[NSString alloc]initWithUTF8String:(const char *) sqlite3_column_text(statement, 1)];
        UIAlertView *notice = [[UIAlertView alloc]initWithTitle:@"Kết Quả" message:kq delegate:self cancelButtonTitle:@"OK" otherButtonTitles:
                                nil, nil];
        [notice show];
    }
    else
    {
        NSLog(@"No data");
    }
    sqlite3_finalize(statement);
}
```

**Hình 5.20 Thực thi đối tượng sqlite\_stmt trong sqlite3\_prepare\_v2**

Câu lệnh SQL được thực hiện bởi các lệnh sau:

```
if (sqlite3_step(statement) == SQLITE_ROW)
{
    kq = [[NSString alloc]initWithUTF8String:(const char *) sqlite3_column_text(statement, 1)];
    UIAlertView *notice = [[UIAlertView alloc]initWithTitle:@"Kết Quả" message:kq delegate:self cancelButtonTitle:@"OK" otherButtonTitles:
                                nil, nil];
    [notice show];
}
else
{
    NSLog(@"No data");
}
```

**Hình 5.21 Thực hiện câu lệnh SQL lấy kết quả trả về**

Trong đó câu lệnh sqlite3\_step() sẽ trả về các dạng như sau:

- Nếu câu lệnh sql dạng INSERT, DELETE, UPDATE, CREATE... thì trả về kết quả có dạng là SQLITE\_OK để báo tình trạng có thực thi được hay không.
- Nếu câu lệnh slq dạng SELECT sẽ trả về kết quả có dạng là SQLITE\_ROW, là tập hợp các hàng dữ liệu được lấy từ database.

#### 5.2.4.4 Truy Xuất Dữ Liệu Database

Khởi tạo câu lệnh truy vấn.

```
NSMutableString *query = [NSMutableString stringWithString:@"SELECT * FROM DemSo WHERE So='1'"];
[query appendString:[Input text]];
[query appendString:@"']";
const char *query_stmt = [query UTF8String];
```

**Hình 5.22 Khởi tạo câu lệnh truy vấn**

Bóc tách dữ liệu lấy được.

```
N;
if (sqlite3_step(statement) == SQLITE_ROW)
{
    kq = [[NSString alloc]initWithUTF8String:(const char *) sqlite3_column_text(statement, 1)];
    UIAlertView *notice = [[UIAlertView alloc]initWithTitle:@"Kết Quả" message:kq delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
    [notice show];
}
else
{
    NSLog(@"No data");
}
```

**Hình 5.23 Tách dữ liệu lấy được**

#### 5.2.4.5 Đóng Kết Nối Database

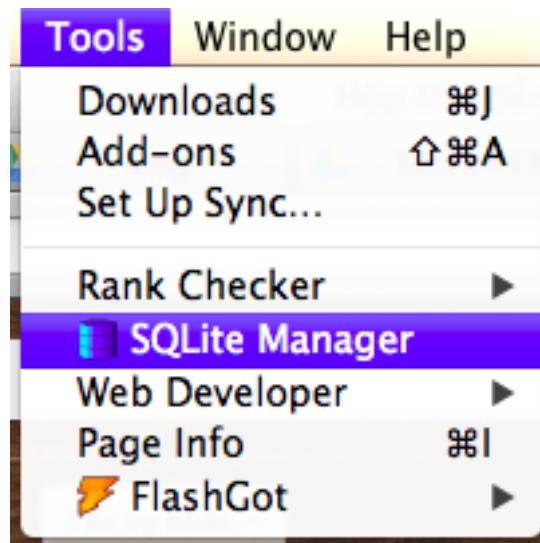
Sau khi hoàn tất quá trình tương tác cơ sở dữ liệu, bạn nên đóng lại kết nối cơ sở dữ liệu để có

```
sqlite3_finalize(statement);
```

**Hình 5.24 đóng kết nối cơ sở dữ liệu**

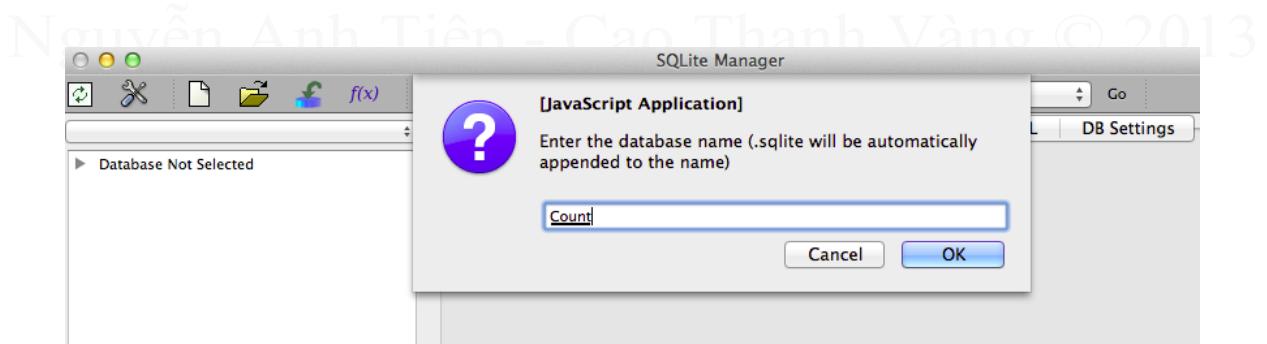
### 5.2.5 Ví Dụ

Bước 1: khởi động firefox, vào SQLite Manager



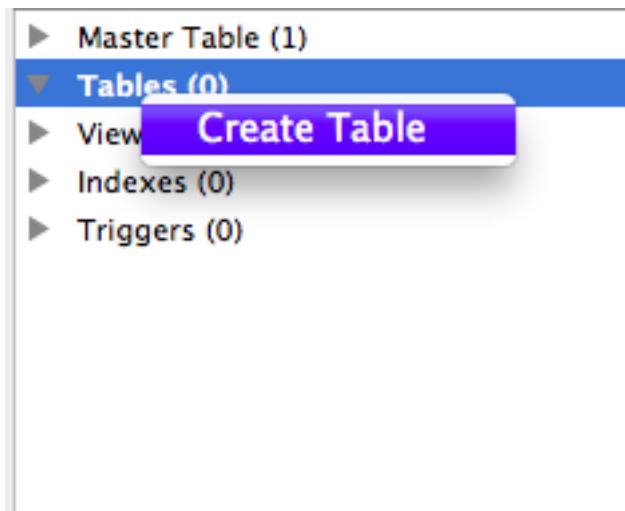
Hình 5.25 Truy cập SQLite Manager

Bước 2: tạo một cơ sở dữ liệu mới tên là “Count”.

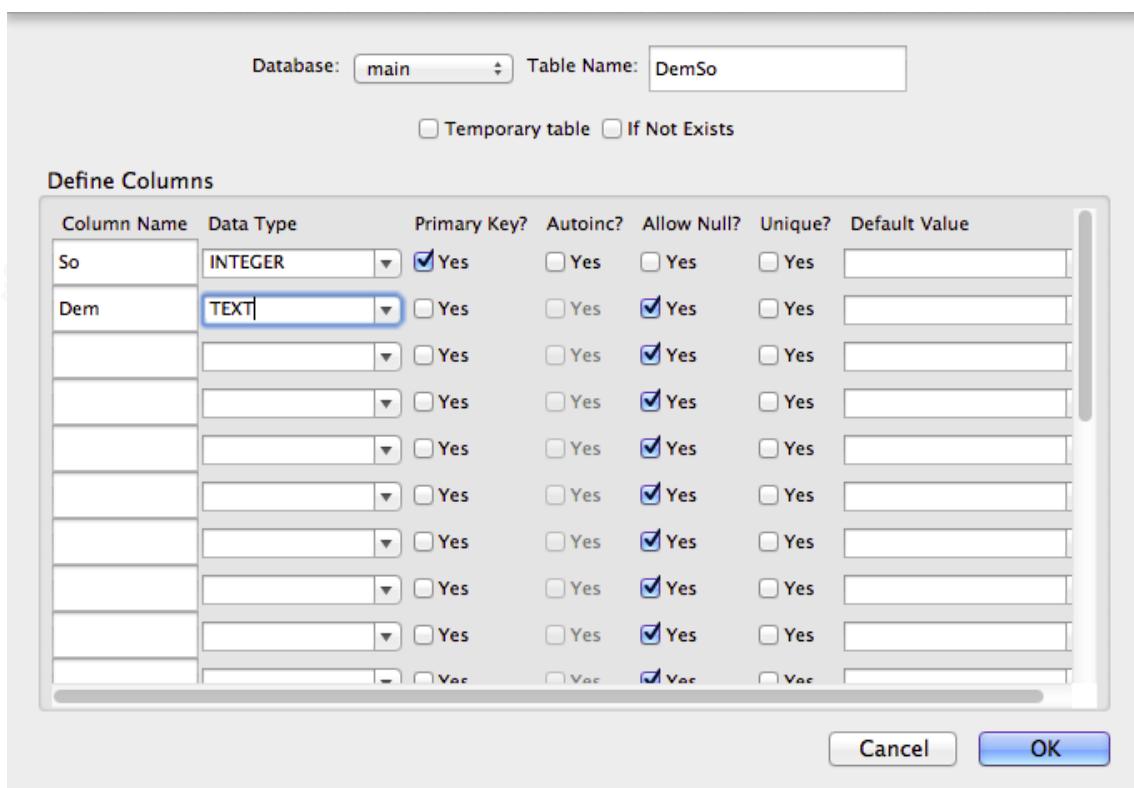


Hình 5.26 Tạo CSDL mới

Bước 3: Tạo một table mới

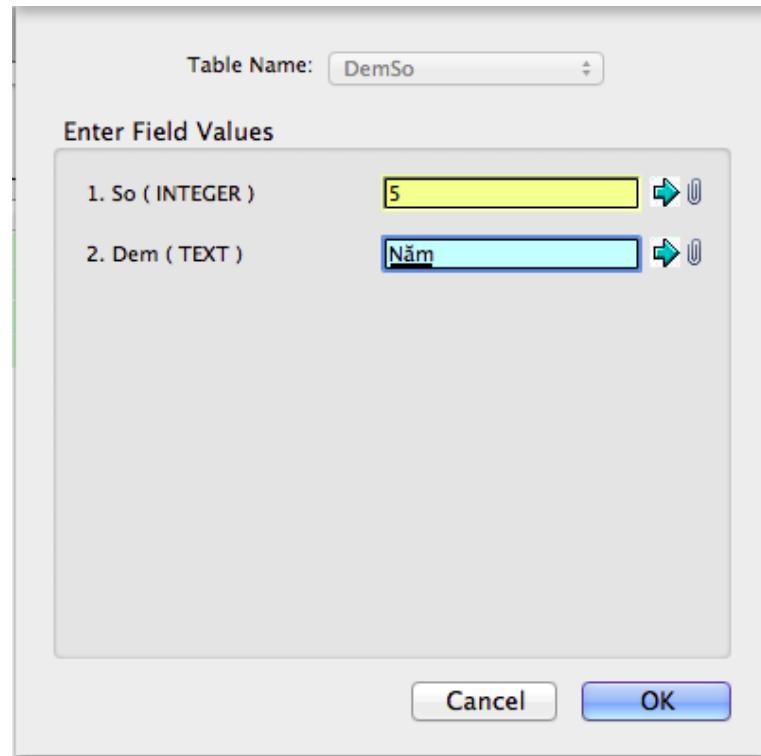


**Hình 5.27 Tạo table**



**Hình 5.28 Khai báo thuộc tính cho Table**

## Bước 4: nhập dữ liệu

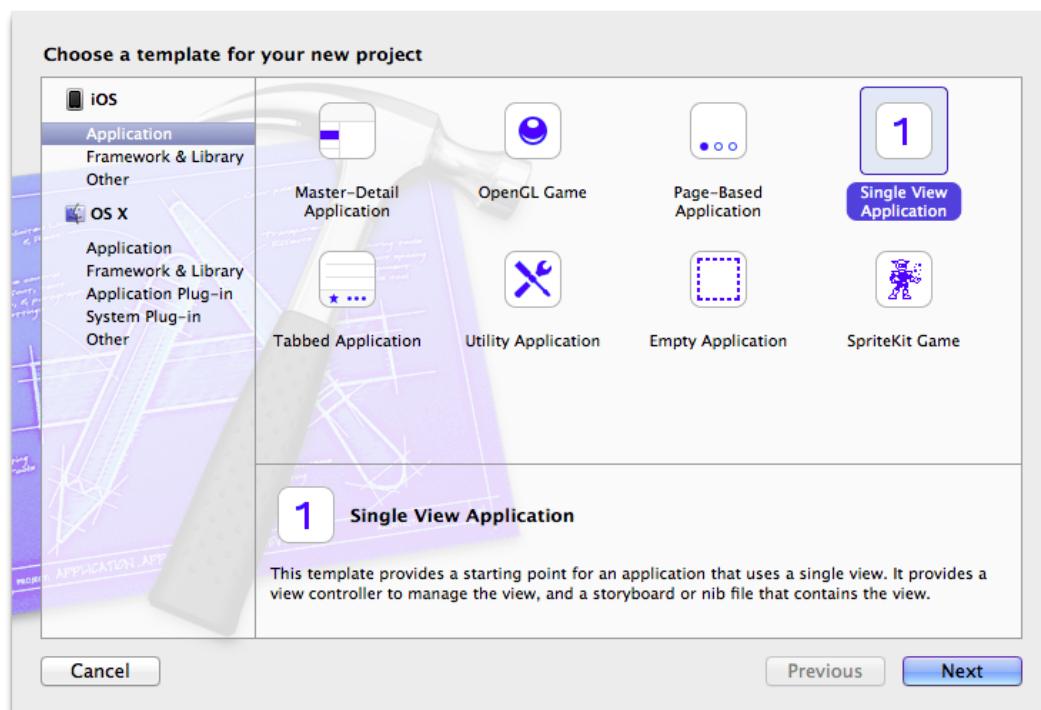


Hình 5.29 Nhập dữ liệu

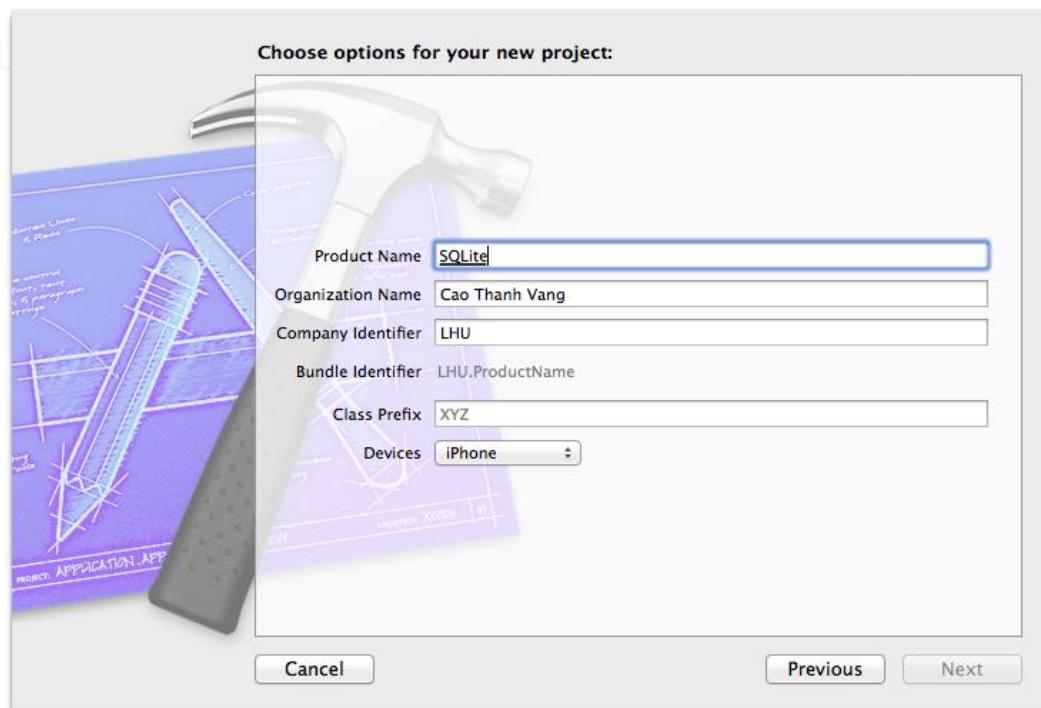
TABLE DemSo		Search	Show All	Add	Duplicate	Edit	Delete
So	Dem						
1	Một						
2	Hai						
3	Ba						
4	Bốn						
5	Năm						
6	Sáu						
7	Bảy						
8	Tám						
9	Chín						
10	Mười						

Hình 5.30 Kết quả

Bước 5: tạo project mới “sqlite”

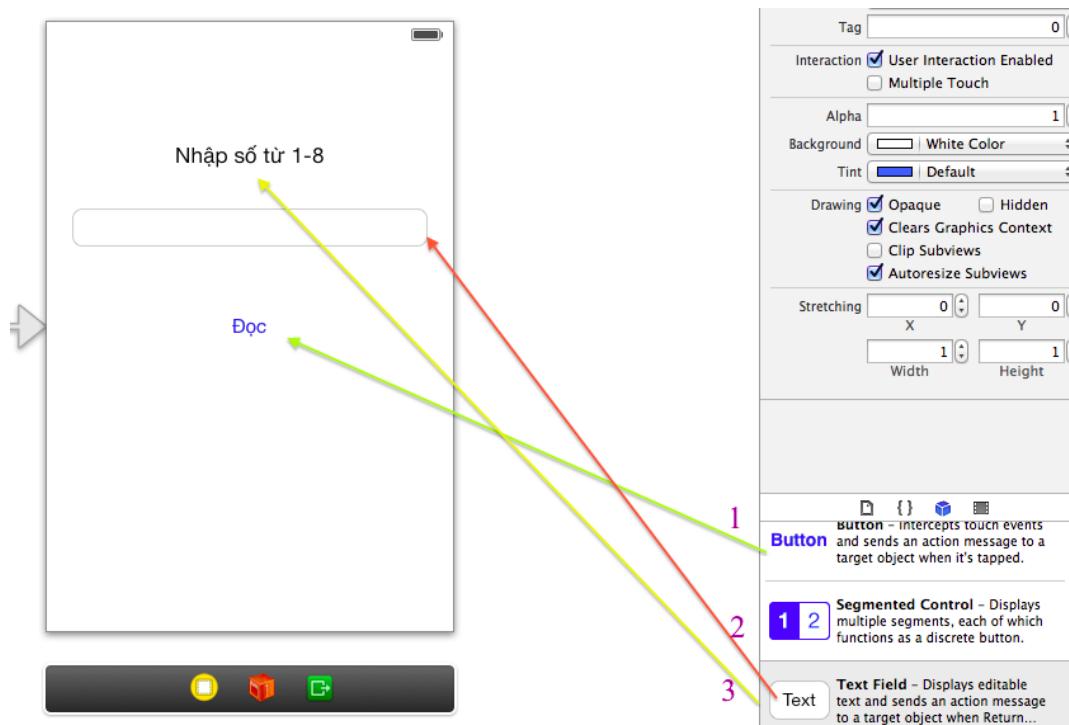


Hình 5.31 Tạo Project mới



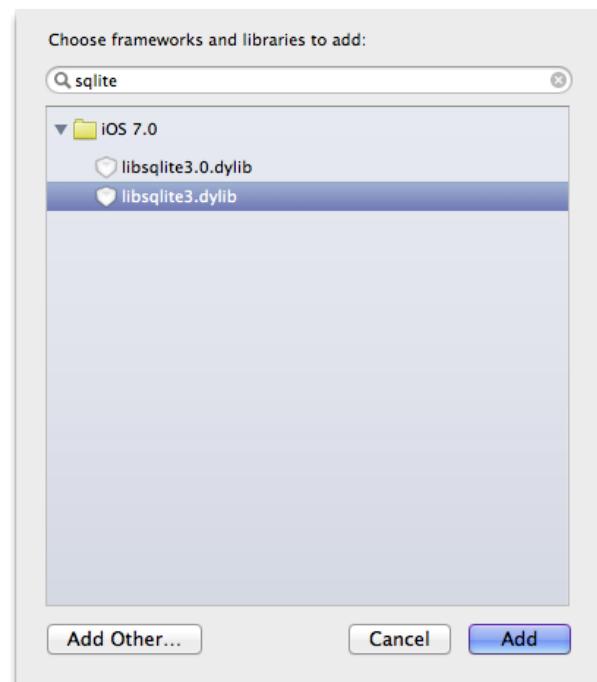
Hình 5.32 Điền thông tin Project

Bước 6: thiết kế giao diện



**Hình 5.33 Thiết kế giao diện**

**Bước 7:** thêm thư viện vào project.

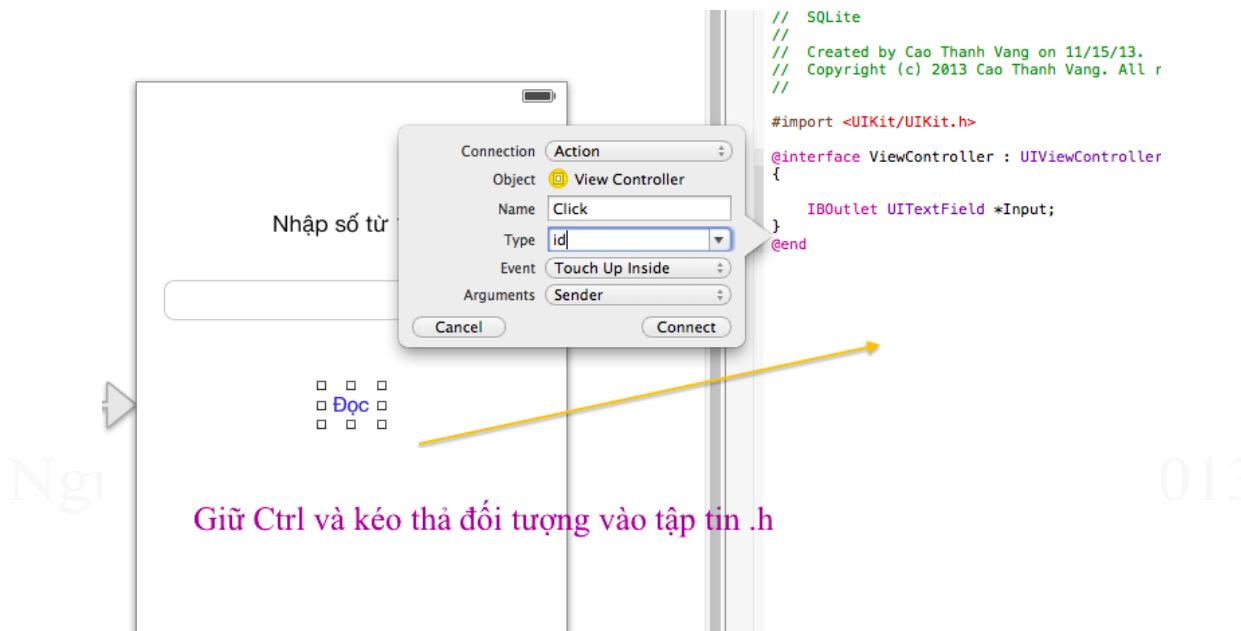


**Hình 5.34 Thêm thư viện**

Name	Status
libssqlite3.dylib	Required ▲
CoreGraphics.framework	Required ▲
UIKit.framework	Required ▲
Foundation.framework	Required ▲

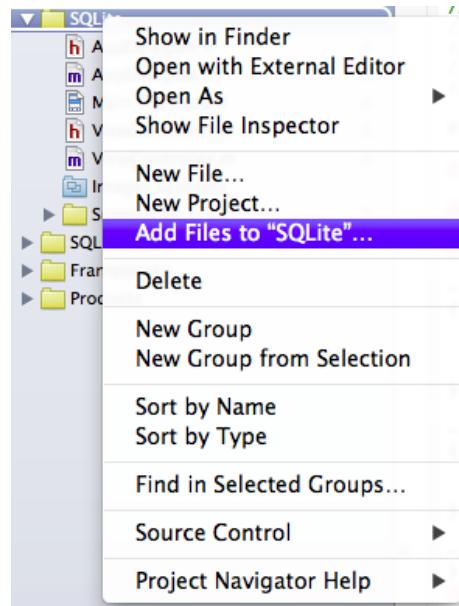
Hình 5.35 Sau khi thêm thư viện

**Bước 8:** Ánh xạ đối tượng

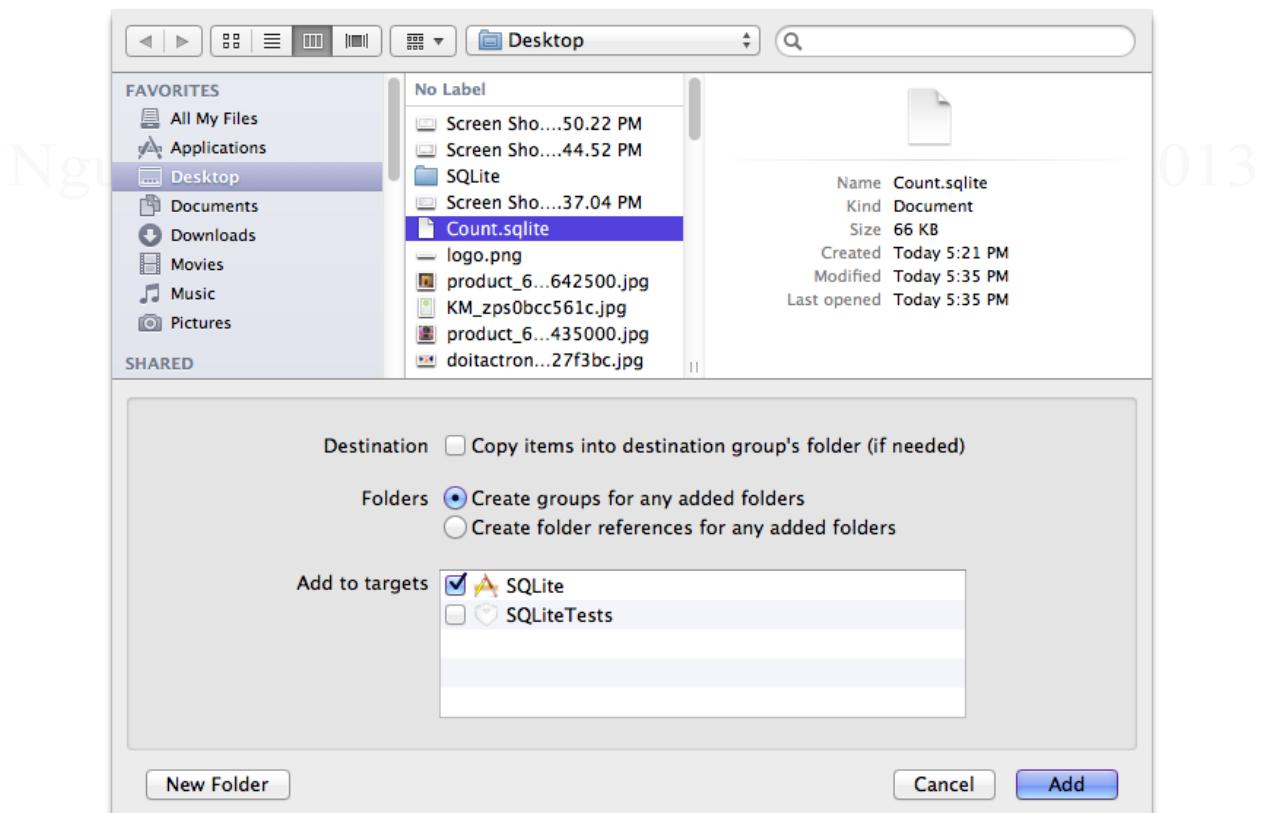


Hình 5.36 Ánh xạ đối tượng

**Bước 9:** Thêm tập tin sqlite vào project



**Hình 5.37 Thêm file mới**



**Hình 5.38 Add**

**Bước 10:** thêm thư viện sqlite3.h vào project, khai báo đối tượng sqlite3 và nsstring

```

#import <UIKit/UIKit.h>
#import <sqlite3.h>

@interface ViewController : UIViewController
{
    IBOutlet UITextField *Input;
    sqlite3 *contactDB;
    NSString *dataPath;
}
- (IBAction)Click:(id)sender;
@end

```

**Hình 5.39 Kết quả ánh xạ**

**Bước 11:** viết code trong hàm View Didload để lấy đường dẫn tới tập tin sqlite.

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    NSString *docdir;
    docdir = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                NSUserDomainMask, YES)[0];
    dataPath = [[NSString alloc] initWithString:[docdir
        stringByAppendingPathComponent:@"Count.sqlite"]];

    NSFileManager *filemanager = [NSFileManager defaultManager];
    if([filemanager fileExistsAtPath:dataPath]==NO)
    {
        NSString *databasePathFromApp = [[[NSBundle mainBundle]
            resourcePath] stringByAppendingPathComponent:@"Count.sqlite"];
        [filemanager copyItemAtPath:databasePathFromApp toPath:dataPath
            error:nil];
    }
    // Do any additional setup after loading the view, typically from a
    // nib.
}

```

**Hình 5.40 Viết code View Didload**

**Bước 12:** viết code cho sự kiện Click

Khai báo đối tượng lưu đường dẫn database.

```
    @interface ViewController ()  
    @end  
    const char *dbpath;
```

**Hình 5.41 Khai báo đối tượng lưu đường dẫn**

```
//khai bao doi tuong luu lai duong dan database  
dbpath =[dataPath UTF8String];
```

**Hình 5.42 Lưu đường dẫn vào đối tượng**

Khai báo đối tượng statement

```
    @interface ViewController ()  
    @end  
    sqlite3_stmt *statement;
```

Nguyễn Anh Tú - CSE - 2013 - Vàng © 2013

**Hình 5.43 Khai báo đối tượng sqlite3\_stmt**

Viết lệnh truy xuất dữ liệu

```
//ham truy xuat  
NSMutableString *query = [NSMutableString stringWithString:@"SELECT  
    * FROM DemSo WHERE So='";  
[query appendString:[Input text]];  
[query appendString:@"']";
```

**Hình 5.44 Lệnh truy xuất dữ liệu**

Khai báo đối tượng lưu trữ câu truy vấn

```
|  
const char *query_stmt = [query UTF8String];
```

**Hình 5.45 Khai báo đối tượng lưu câu truy vấn**

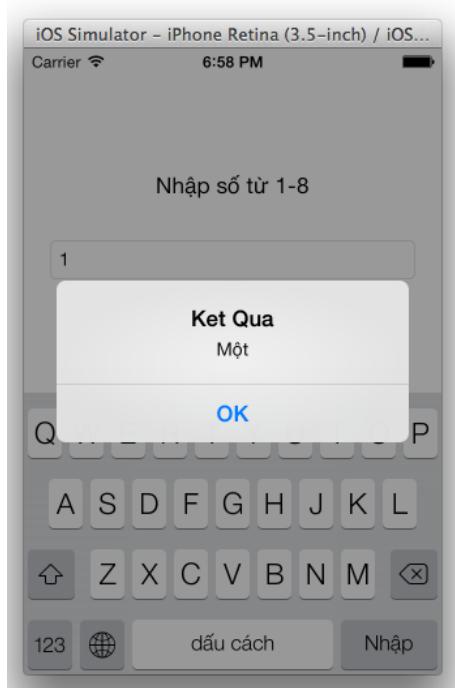
Viết code truy vấn

```
NSString *kq = [[NSString alloc] init];
if(sqlite3_open(dbpath, &contactDB)==SQLITE_OK)
{
    //-----
    if (sqlite3_prepare_v2(contactDB,
                           query_stmt, -1, &statement, nil) ==
        SQLITE_OK)
    {
        if (sqlite3_step(statement) == SQLITE_ROW)
        {
            kq = [[NSString alloc] initWithUTF8String:(const char *)
                sqlite3_column_text(statement, 1)];
            UIAlertView *notice = [[UIAlertView
                alloc]initWithTitle:@"Kết Quả" message:kq delegate:
                self cancelButtonTitle:@"OK" otherButtonTitles:nil,
                nil];
            [notice show];
        }
        else
        {
            NSLog(@"No data");
        }
        sqlite3_finalize(statement);
    }
}
```

Nguyễn Văn Hùng - Sinh viên năm 2013

Hình 5.46 Viết code truy vấn dữ liệu

Kết quả.



**Hình 5.47 Kết quả**

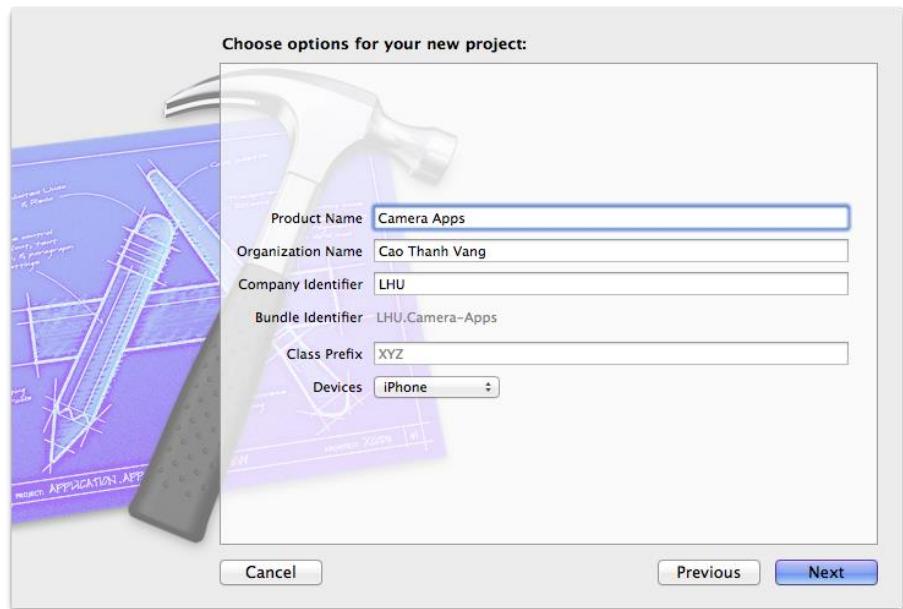
### 5.3 SỬ DỤNG CAMERA IPHONE

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013  
**5.3.1 Giới Thiệu**

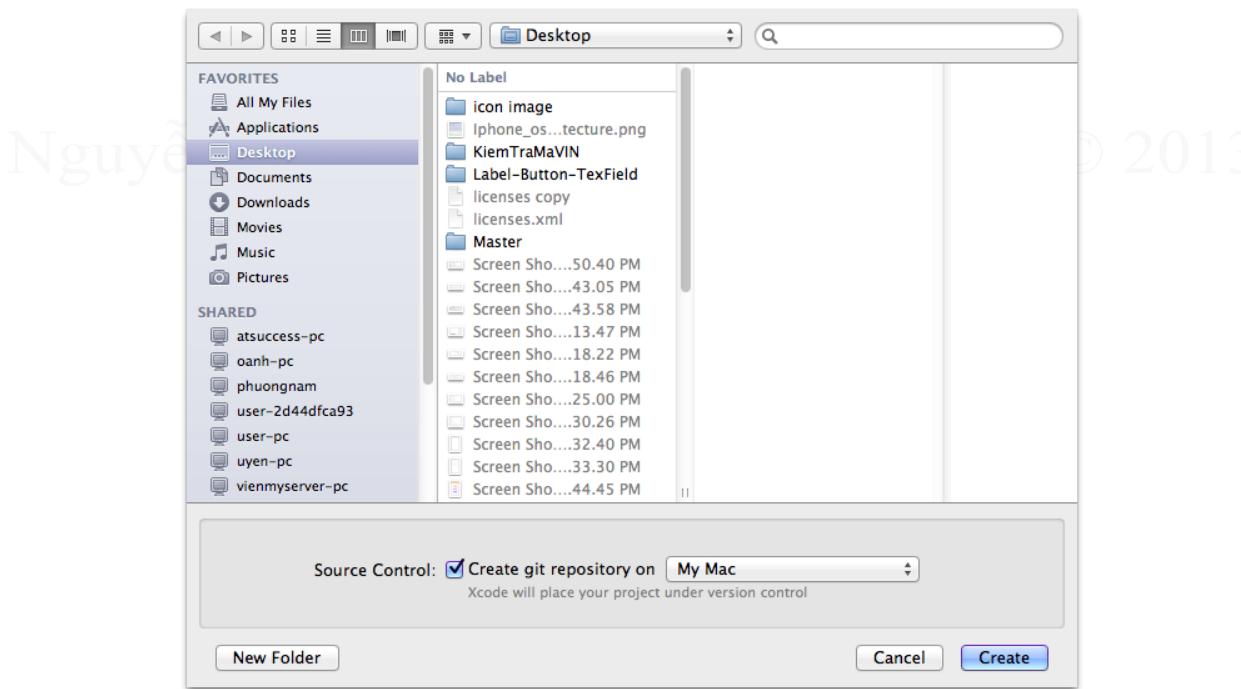
Phần này sẽ hướng dẫn các bạn cách sử dụng thư viện hỗ trợ của iOS để dùng camera iPhone chụp ảnh. Thư viện iOS cung cấp một lớp là UIImagePickerController dùng để quản lý việc tương tác với camera hoặc photo library. Tuy nhiên UIImagePickerController không thể sử dụng trực tiếp mà cần tới một đối tượng khác delegate (thừa kế) lại nó. Sau đây chúng ta sẽ xây dựng một ứng dụng đơn giản dùng camera để chụp ảnh. Giao diện của ứng dụng này gồm một UIImageView để hiển thị hình ảnh, một button tên là “Chụp Ảnh”.

#### 5.3.2 Ví Dụ

Tạo ứng dụng mới đặt tên là **CameraApp**.

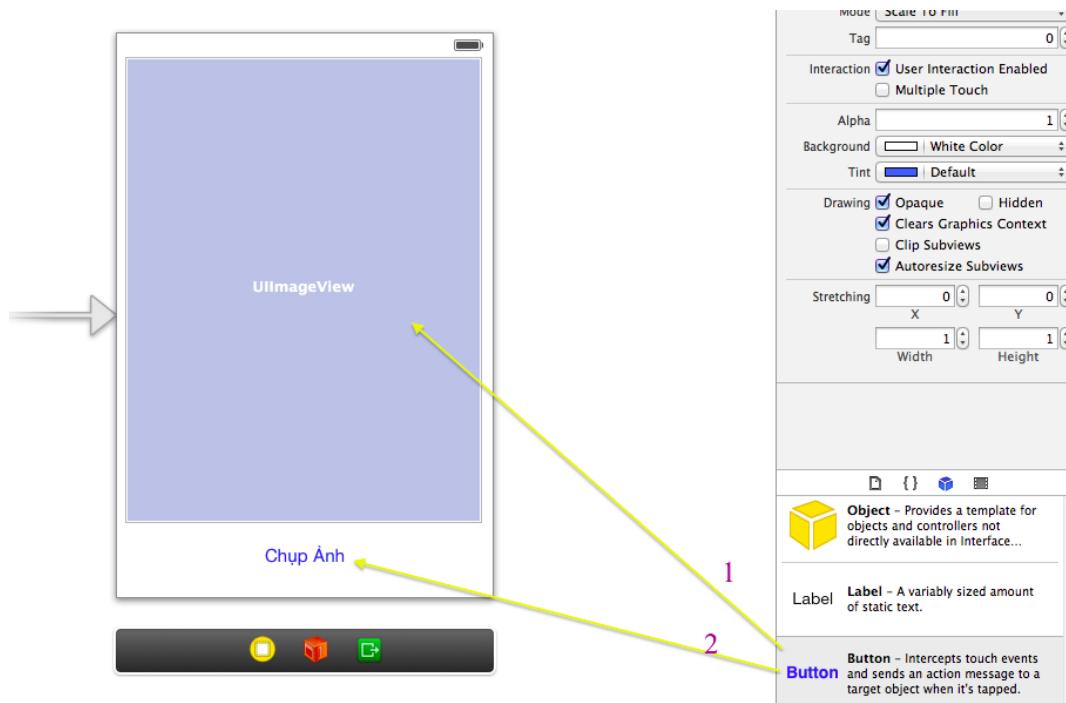


**Hình 5.48 Tạo project mới**



**Hình 5.49 Chọn nơi lưu project**

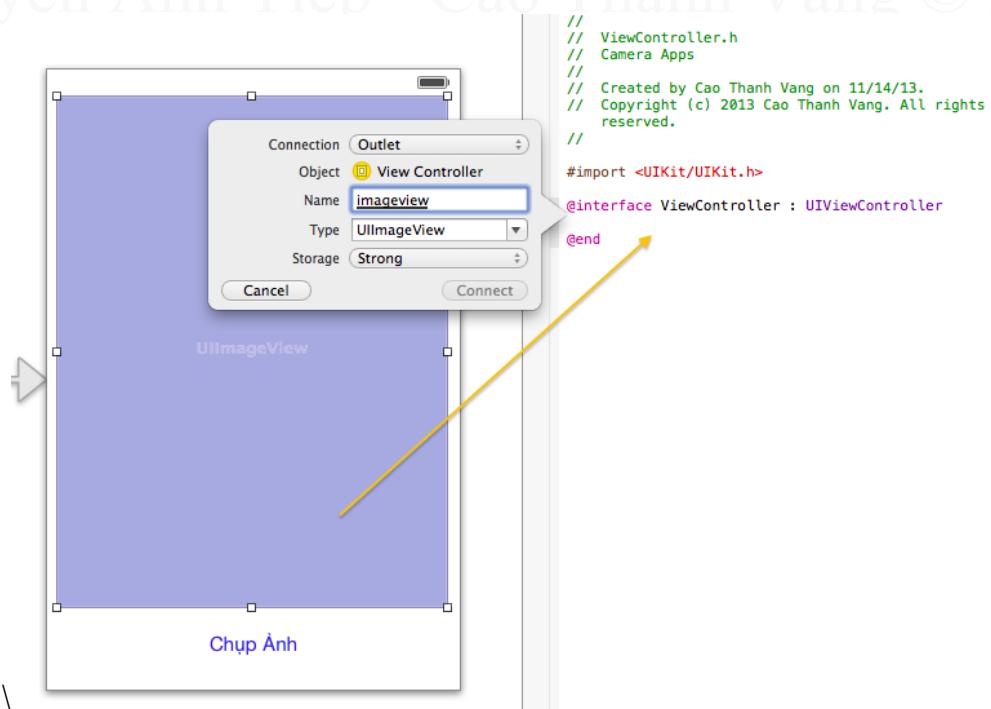
Thiết kế giao diện ứng dụng.



**Hình 5.50 Thiết kế giao diện**

Ánh xạ các đối tượng vào tập tin **ViewController.h**.

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013



**Hình 5.51 Ánh xạ đối tượng**

```

// ViewController.h
// Camera Apps
//
// Created by Cao Thanh Vang on 11/14/13.
// Copyright (c) 2013 Cao Thanh Vang. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
@property (strong, nonatomic) IBOutlet UIImageView *imageview;
- (IBAction)takephoto:(id)sender;

@end

```

### Hình 5.52 Kết quả

Trong ViewController, thêm đoạn code sau để thừa kế lại UIImagePickerController.

```

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<
UIImagePickerControllerDelegate,
UINavigationControllerDelegate>

```

### Hình 5.53 Thừa kế UIImagePickerController

Viết code cho button Chụp Ảnh.

```

- (IBAction)takephoto:(id)sender {
    UIImagePickerController *picker = [[UIImagePickerController alloc] init];
    picker.delegate = self;
    picker.allowsEditing = YES;
    picker.sourceType = UIImagePickerControllerSourceTypeCamera;
    [self presentViewController:picker animated:YES completion:NULL];
}

```

### Hình 5.54 Viết code cho button Chụp ảnh

Viết code hai phương thức của UIImagePickerController.

```

-(void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    UIImage *chonlua = info[UIImagePickerControllerEditedImage];
    self.imageview.image = chonlua;
    [picker dismissViewControllerAnimated:YES completion:NULL];
}

-(void)imagePickerControllerDidCancel:(UIImagePickerController *)picker
{
    [picker dismissViewControllerAnimated:YES completion:NULL];
}

```

**Hình 5.55** Viết code cho hai phương thức của UIImagePickerController

Viết code cho ViewDidLoad.

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    if (![[UIImagePickerController isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera]) {
        UIAlertView *myalertView = [[UIAlertView alloc] initWithTitle:@"Error"
                                                               message:@"Device has no camera"
                                                               delegate:nil
                                                               cancelButtonTitle:@"OK"
                                                               otherButtonTitles: nil];
        [myalertView show];
    }
    // Do any additional setup after loading the view, typically from a nib.
}

```

Nguyễn Anh Tài - Cao Thắng Vàng © 2013

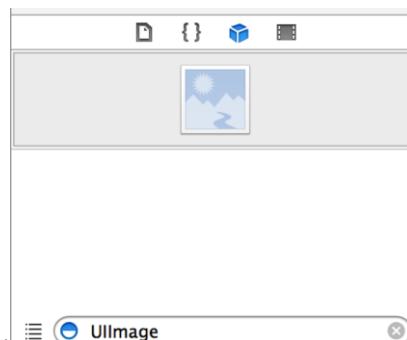
**Hình 5.56** Viết code cho hàm ViewDidLoad

Chạy ứng dụng lên thiết bị thật để kiểm tra kết quả.

## 5.4 UIIMAGE

### 5.4.1 Giới Thiệu

**UIImage** dùng để hiển thị hình ảnh, có thể sử dụng các hiệu ứng animation để hiển thị hình ảnh.



**Hình 5.57** Đối tượng UIImage



Hình 5.58 UIImageView khi kéo ra thiết kế giao diện

#### 5.4.2 Các Định Dạng Ảnh Hỗ Trợ Trên iPhone

Các dạng file ảnh được hỗ trợ tốt trên iPhone:

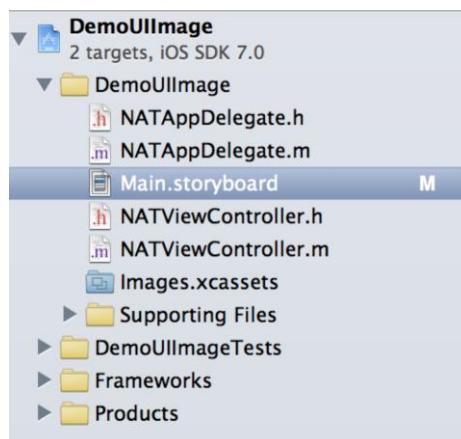
Tên định dạng	Tên file mở rộng
Tagged Image File Format (TIFF)	.tiff, .tif
Join Photographic Experts Group (JPEG)	.jpg, .jpeg
Graphic Interchange Format (GIF)	.gif
Portable Network Graphic (PNG)	.png
Windows Bitmap Format (DIB)	.bmp, .bmfpf
Windows Icon Format	.ico
Windows Cursor	.cur
Xwindow bitmap	.xbm

#### 5.4.3 Ví Dụ

Hiển thị ảnh từ một file có sẵn trên máy.

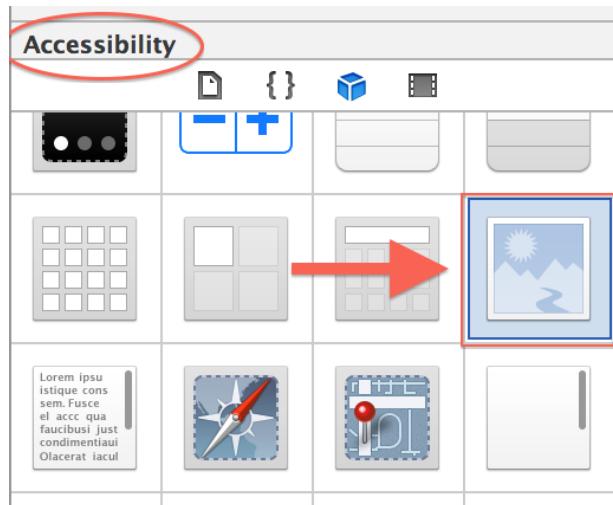
Bước 1: Add UIImageView vào “Storyboard”.

→ Chọn “Main.storyboard”

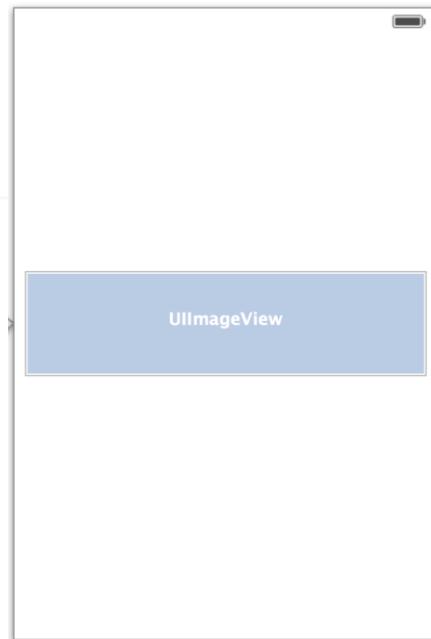


Hình 5.59 Chọn storyboard

→ Kéo thả UIImageView từ hộp thoại “Accessibility” vào “Storyboard”.



Hình 5.60 Đối tượng UIImageView



Hình 5.61 Kết quả sau khi kéo vào storyboard

Bước 2: Ánh xạ UIImageView vào file “NATViewController.h” dạng outlet với tên “imgHinhAnh”.

```

#import <UIKit/UIKit.h>

@interface NATViewController : UIViewController {
    IBOutlet UIImageView *imgHinhAnh;
}

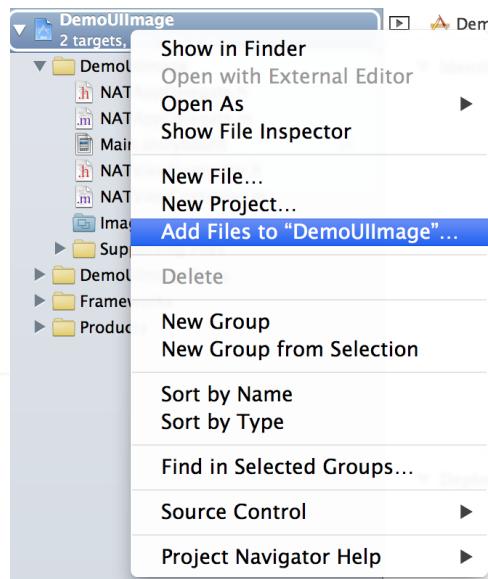
@end

```

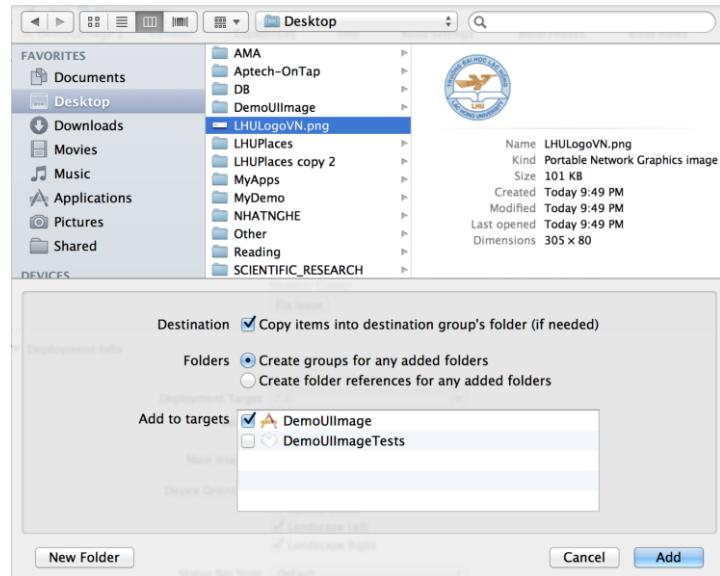
**Hình 5.62 Kết quả sau khi ánh xạ**

**Bước 3:** Add file hình vào project đang làm.

→ Click phải chuột vào project → chọn “Add Files to...”



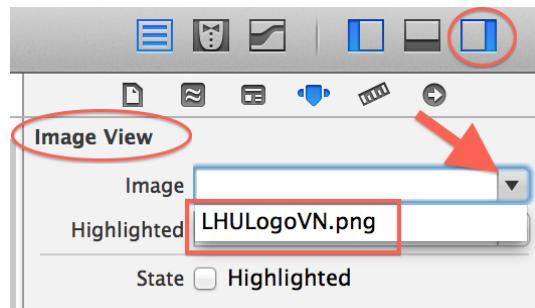
**Hình 5.63 Thêm file vào project**



**Hình 5.64 Chọn file cần thêm vào**

**Bước 4:** Load file hình vào UIImageView.

Cách 1: Chọn hình từ hộp thoại “Image View”.



**Hình 5.65 Cách chọn từ Image View**



**Hình 5.66 Kết quả sau khi chọn**

Cách 2: Gọi phương thức sau để load file hình vào “UIImageView”.

```
imgHinhAnh.image = [UIImage imageNamed:@"LHULogoVN.png"];
```

**Hình 5.67 Cách sử dụng code**

→ Mở file “NATViewController.m”, tại phương thức “viewDidLoad”.



Hình 5.68 Chọn NATViewController

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    imgHinhAnh.image = [UIImage imageNamed:@"LHULogoVN.png"];
}
```

Hình 5.69 Viết code hiện thị hình ảnh

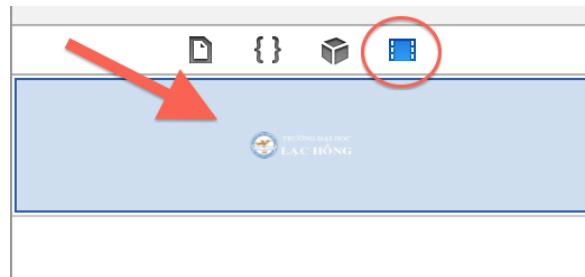
Kết quả.



Hình 5.70 Kết quả

Lưu ý: phương thức “viewDidLoad” sẽ được gọi mỗi khi chúng ta mở ứng dụng.

Cách 3: Kéo thả trực tiếp đối tượng Image vào “Storyboard”:



Hình 5.71 Kéo thả Image vào Storyboard



Hình 5.72 Kết quả

Hiển thị ảnh từ một URL:

Sử dụng phương thức “**initWithData**” để load nội dung từ url.

```
NSString *strURL= @"http://lhu.edu.vn/Page/LHUVN/_default/image/LHULogoVN.png";
NSURL *url = [NSURL URLWithString:strURL];
NSData *imageData = [[NSData alloc] initWithContentsOfURL:url];
imgHinhAnh.image = [[UIImage alloc] initWithData:imageData];
```

Hình 5.73 Code load nội dung từ url

→ Mở file “**NATViewController.m**”, tại phương thức “**viewDidLoad**”.



**Hình 5.74 Mở file NATViewController.m**

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    NSURL *url = [NSURL URLWithString:@"http://lhu.edu.vn/ViewPage/LHUVN/_default/image/LHULogoVN.png"];
    NSData *imageData = [[NSData alloc] initWithContentsOfURL:url];
    imgHinhAnh.image = [[UIImage alloc] initWithData:imageData];
}
```

**Hình 5.75 Viết code cho hàm ViewDidLoad**

Kết quả.

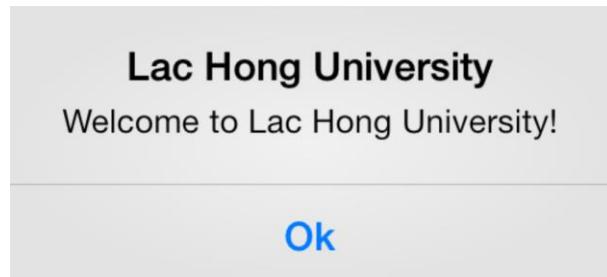


**Hình 5.76 Kết quả**

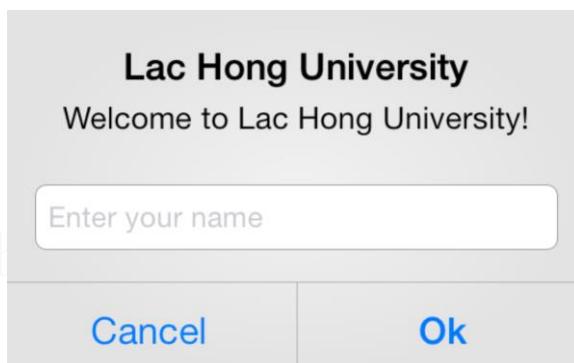
## 5.5 UIALERT VIEW

### 5.5.1 Giới Thiệu

**UIalertView** là đối tượng được sử dụng để hiển thị nội dung thông báo cho người dùng, hoặc để người dùng nhập liệu.



Hình 5.77 UIAlertview



Hình 5.78 UIAlertview cho phép nhập liệu

### 5.5.2 Đặc Điểm

Khởi tạo đối tượng UIAlertview

```
initWithTitle: message: delegate: cancelButtonTitle: otherButtonTitles
```

Các thuộc tính của UIAlertview

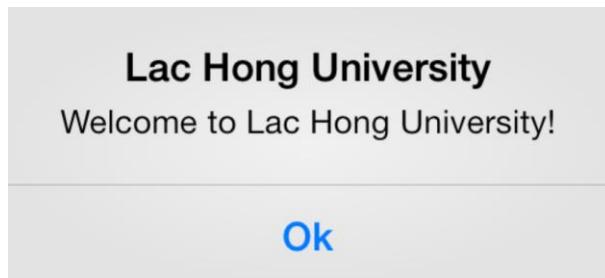
Thuộc tính	Điễn giải
title	Tiêu đề của Alert View.
message	Nội dung thông báo
delegate	Đối tượng nhận giá trị từ UIAlertview.
alertViewStyle	Kiểu của 1 Alert View khi hiển thị trên ứng dụng
numberOfButtons	Trả về số lượng buttons có trên Alert View
cancelButtonIndex	Vị trí index của button Cancel trên UIAlertview
firstOtherButtonIndex	Giá trị của nút đầu tiên trong UIAlertview, mặc định là 0.

## Các phương thức của UIAlertView

Phương thức	Điễn giải
addButtonWithTitle	Gán tiêu đề cho Alert View
buttonTitleAtIndex	Lấy tiêu đề của button tại vị trí index
textAtIndex	Lấy nội dung của textField tại vị trí index
show	Gọi 1 Alert View
dismissWithClickedButtonIndex	Huỷ 1 UIAlertView

### 5.5.3 Ví Dụ

Hiển thị AlerView dạng thông báo.



**Hình 5.79 Hiển thị dạng thông báo**

**Bước 1:** Tại phương thức “viewDidLoad” khởi tạo đối tượng “UIAlertView” như sau:

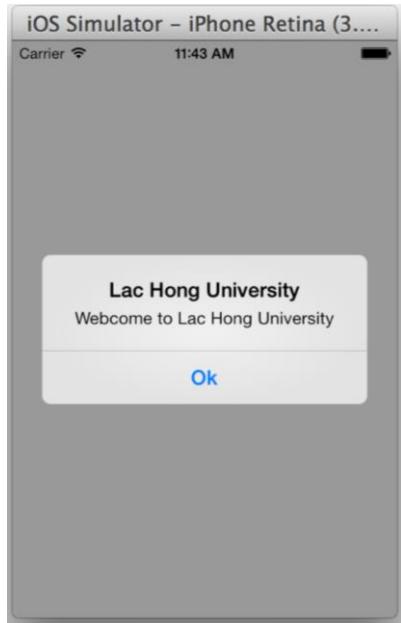
```
thongBao = [[UIAlertView alloc]
            initWithTitle:@"Lac Hong University"
            message:@"Webcome to Lac Hong University"
            delegate:self
            cancelButtonTitle:@"Ok"
            otherButtonTitles:nil, nil];
```

**Bước 2:** Hiển thị “UIAlertView” bằng phương thức “show”

→ Thêm đoạn code sau vào cuối phương thức “viewDidLoad”:

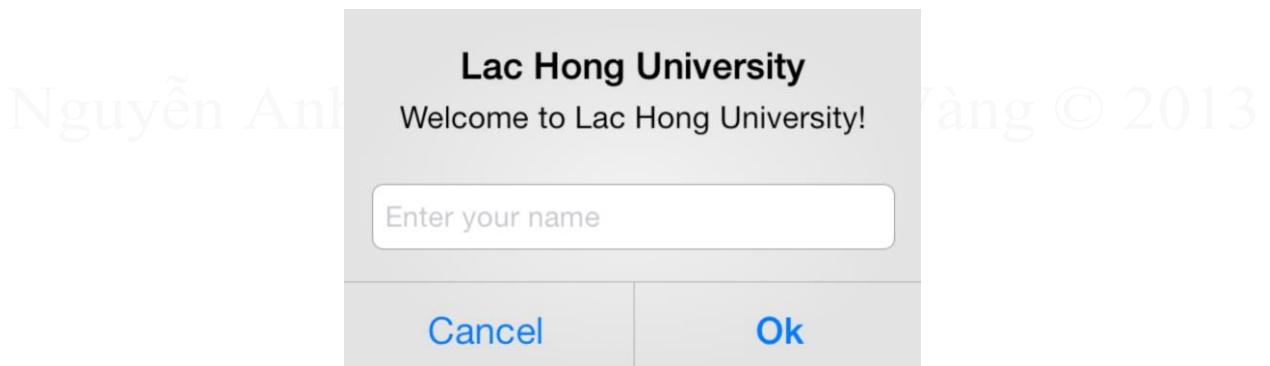
```
[thongBao show];
```

**Bước 3: Cmd + R chạy thử, kết quả:**



**Hình 5.80 Kết quả hiển thị**

Hiển thị AlertView dạng Text Field



**Hình 5.81 Hiển thị dạng Text Field**

**Bước 1:** Khai báo “AlertView”

→ Tại “viewDidLoad” khai báo “AlertView” như sau:

```
thongBao = [[UIAlertView alloc] initWithTitle:@"Lac Hong University"
                                         message:@"Webcome to Lac Hong University"
                                         delegate:self
                                         cancelButtonTitle:@"Cancel"
                                         otherButtonTitles:@"Ok", nil];
```

**Hình 5.82 Khai báo AlertView**

**Bước 2:** Thiết lập thuộc tính “alertViewStyle” cho “AlertView” là kiểu “Text”.

```
thongBao.alertViewStyle = UIAlertViewStylePlainTextInput;
```

### Hình 5.83 Thiết lập thuộc tính

**Bước 3:** Thiết lập thuộc tính cho “Text”.

```
UITextField *txtTen = [thongBao textFieldAtIndex:0];
txtTen.placeholder = @"Enter your name";
```

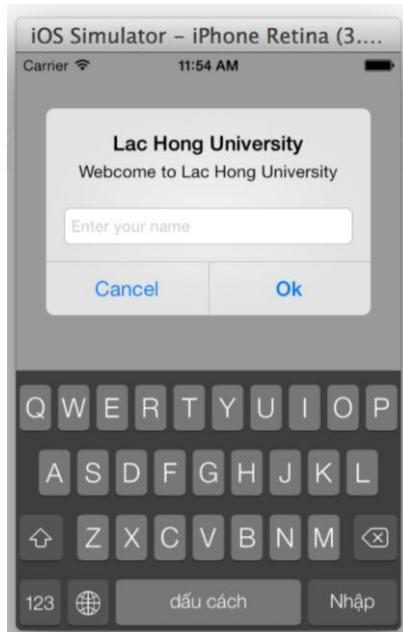
### Hình 5.84 Thiết lập thuộc tính

**Bước 4:** Hiển thị “AlertView”.

```
[thongBao show];
```

### Hình 5.85 Hiển thị AlertView

**Bước 5:** Cmd + R chạy thử, kết quả:



### Hình 5.86 Hiển thị kết quả

## 5.6 UISLIDER

### 5.6.1 Giới Thiệu

UISlider là một đối tượng cho phép người dùng thay đổi giá trị bằng cách di chuyển thanh trượt.



Hình 5.87 UISLIDER

### 5.6.2 Đặc Điểm

Lấy giá trị từ Slider:

Cách 1: dùng property “value”

value property

Cách 2: dùng phương thức “setValue”

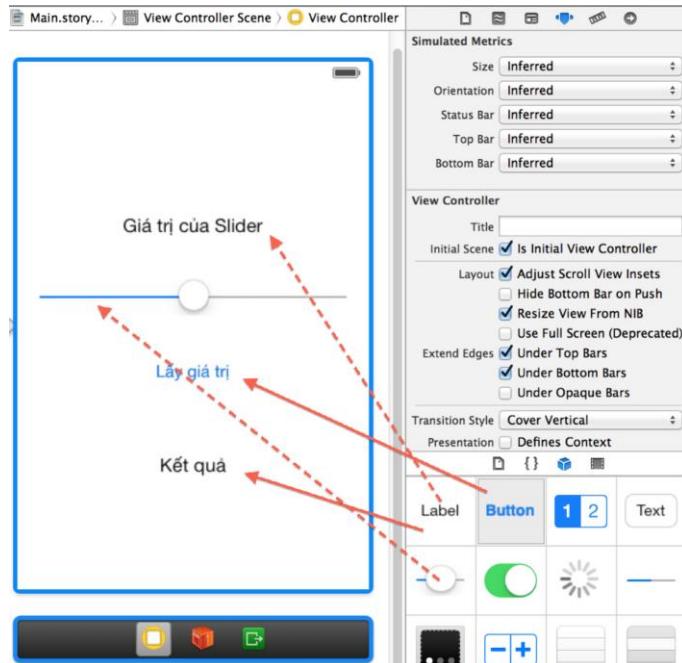
– setValue:animated:

Thiết lập giá trị nhỏ nhất và lớn nhất cho Slider:

- Thiết lập giá trị nhỏ nhất: **minimumValue**
- Thiết lập giá trị lớn nhất: **maximumValue**

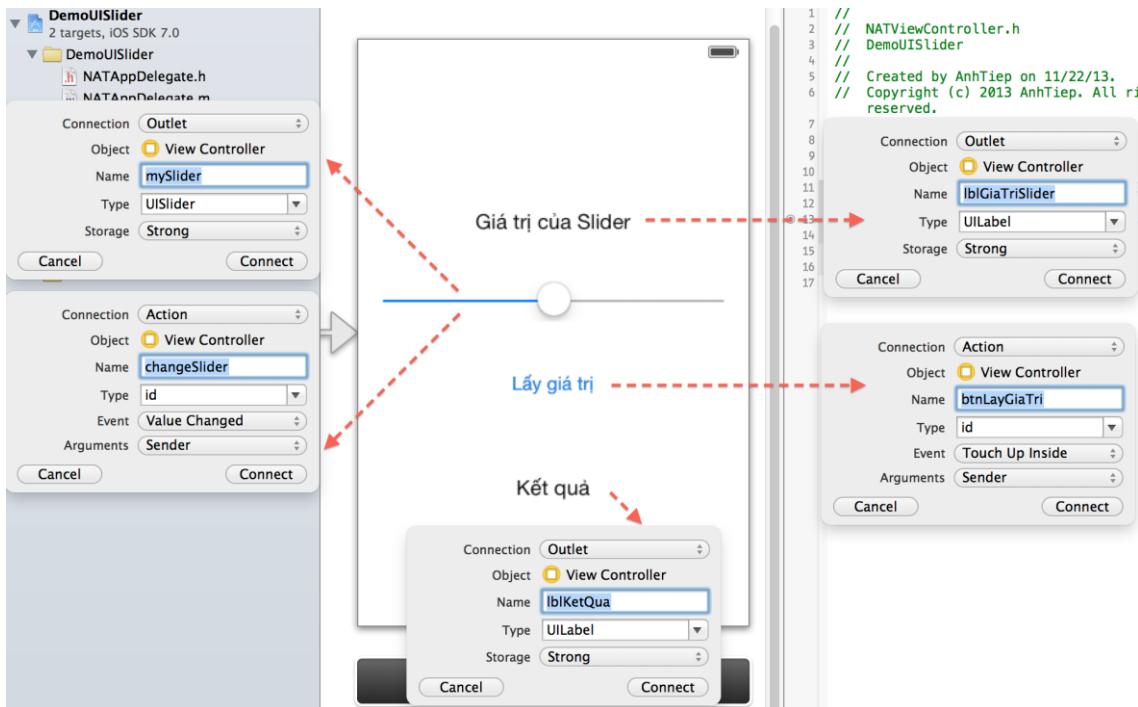
### 5.6.3 Ví Dụ

Bước 1: Thiết kế giao diện như sau.



Hình 5.88 Thiết kế giao diện

## Bước 2: Map các đối tượng như sau:



Hình 5.89 Ánh xạ các đối tượng

→ Kết quả sau khi Map

```

8
9 #import <UIKit/UIKit.h>
10
11 @interface NATViewController : UIViewController {
12
13     IBOutlet UILabel *lblGiaTriSlider;
14     IBOutlet UISlider *mySlider;
15     IBOutlet UILabel *lblKetQua;
16 }
17
18 - (IBAction)btnLayGiaTri:(id)sender;
19 - (IBAction)changeSlider:(id)sender;
20 @end
21

```

Hình 5.90 Kết quả sau khi ánh xạ

Bước 3: Thiết lập giá trị nhỏ nhất và lớn nhất cho “Slider” trong khoảng “5 → 50”.

→ Mở “ViewController.m” → thêm đoạn code sau vào cuối phương thức “viewDidLoad”.

```
mySlider.minimumValue = 5;  
mySlider.maximumValue = 50;
```

### Hình 5.91 Thiết lập giá trị

→ Thiết lập giá trị chọn mặc định là “30” → thêm đoạn code sau vào cuối phương thức “**viewDidLoad**”.

```
[mySlider setValue:30.0 animated:YES];
```

### Hình 5.92 Thêm code vào cuối viewDidLoad

**Bước 4:** Gán kết quả của Slider vào “lblGiaTriSlider” khi người dùng thay đổi thanh trượt.

→ Thêm đoạn code sau vào phương thức “**changeSlider**”.

```
lblGiaTriSlider.text = [NSString stringWithFormat:@"Bạn chọn %.0f", mySlider.value];
```

### Hình 5.93 Thêm code vào phương thức changeSlider

**Bước 5:** Lấy giá trị từ Slider gán vào “lblKetQua” khi người dùng click vào button “**btnLayGiaTri**”.

```
lblKetQua.text = [NSString stringWithFormat:@"%.0f", mySlider.value];
```

### Hình 5.94 Gán giá trị cho label

**Bước 6:** Cmd + r chạy thử, kết quả.



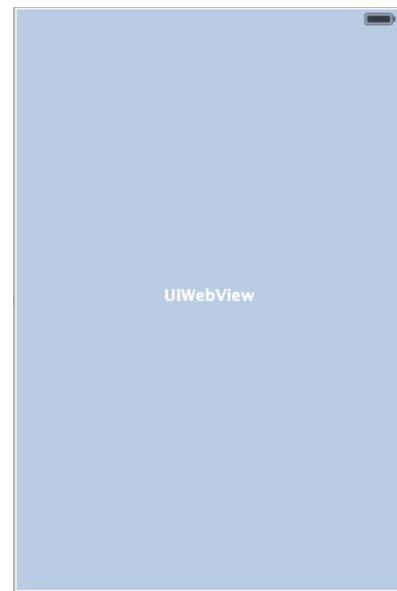
**Hình 5.95 Kết quả hiển thị**

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

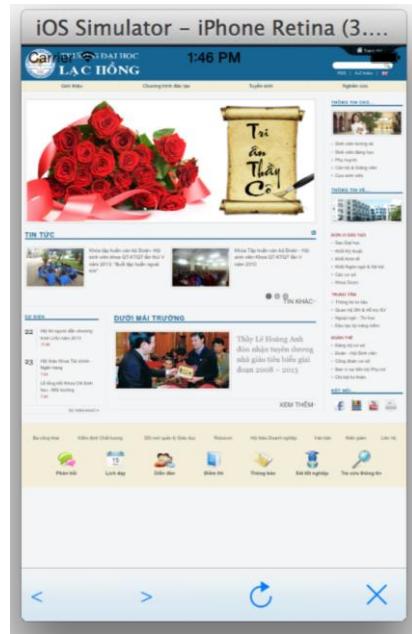
## 5.7 UIWEBVIEW

### 5.7.1 Giới Thiệu

**UIWebView** là đối tượng giúp hiển thị nội dung của một url nào đó.



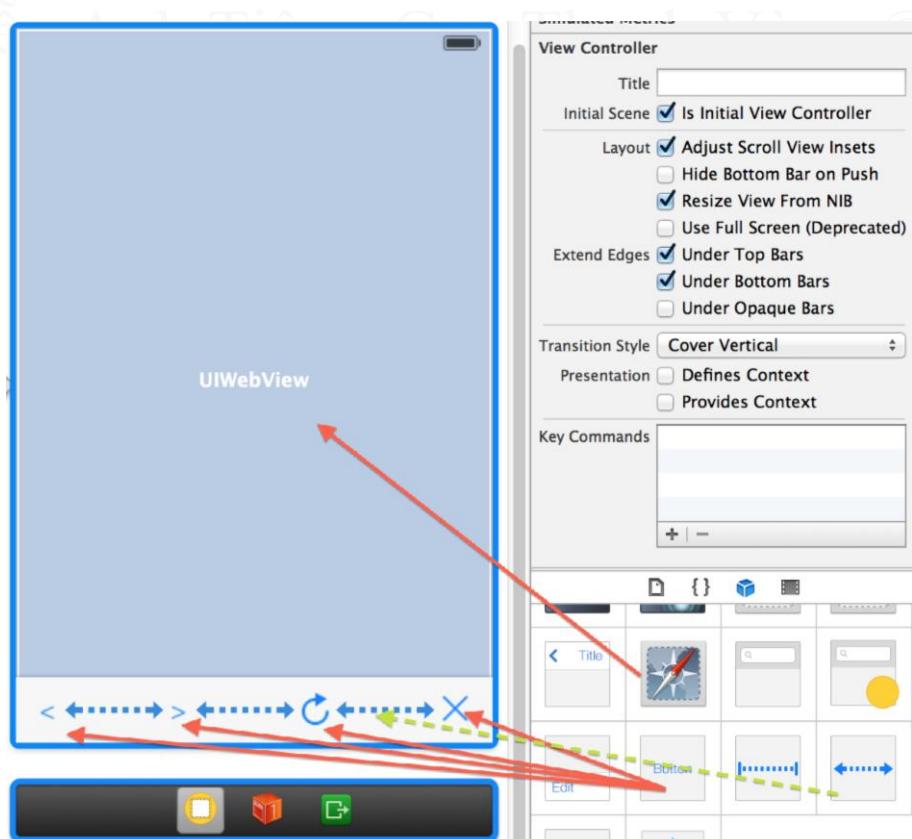
**Hình 5.96 UIWebView**



**Hình 5.97 Hiển thị UIWebView**

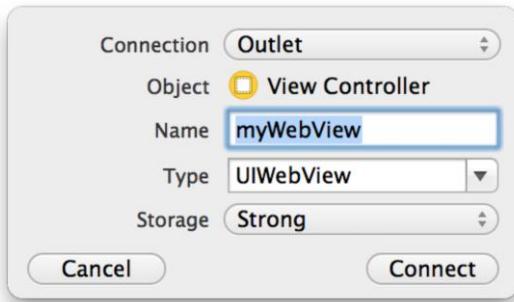
### 5.7.2 Ví Dụ

Bước 1: Thiết kế giao diện như sau.



**Hình 5.98 Thiết kế giao diện**

**Bước 2:** Map đối tượng “UIWebView” với tên “myWebView”.



**Hình 5.99** Ánh xạ đối tượng

**Bước 3:** Tải một url.

→ Mở file “ViewController.m” thêm đoạn code sau vào cuối phương thức “viewDidLoad”.

```
NSURL *url = [NSURL URLWithString:@"http://lhu.edu.vn"];
NSURLRequest *myRequest = [NSURLRequest requestWithURL:url];
[myWebView loadRequest:myRequest];
```

**Hình 5.100** Thêm code vào viewDidLoad

→ Chạy thử, kết quả.



**Hình 5.101** Kết quả chạy thử

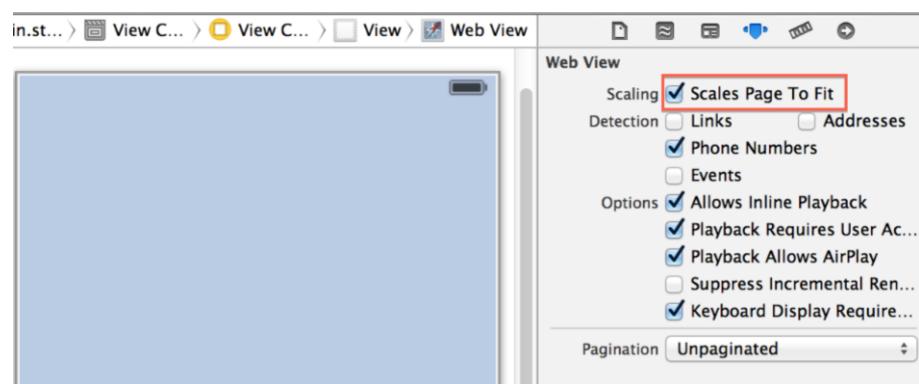
**Bước 4:** Thực hiện các chức năng: back, next, refrest, stop.

→ Qua giao diện lần lượt nhấn giữ Ctrl các đối tượng (back, next, refrest,top) và kéo thả vào “UIWebView”.



Hình 5.102 Tạo kết các cho các chức năng của button.

→ Thiết lập thuộc tính cho “UIWebView” là “Scales Page To Fit”.



Hình 5.103 Chọn Scales Page To Fit

Bước 5: Chạy thử, kết quả.



**Hình 5.104 Kết quả**

## **5.8 ACTIVITY INDICATOR VIEW**

### **5.8.1 Giới Thiệu**

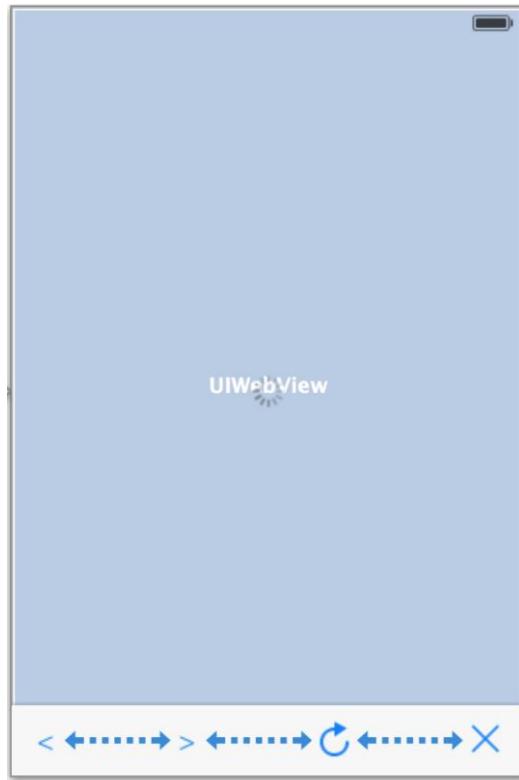
Activity Indicator View là một đối tượng thông báo chờ đợi cho người dùng.



**Hình 5.105 Activity Indicator View**

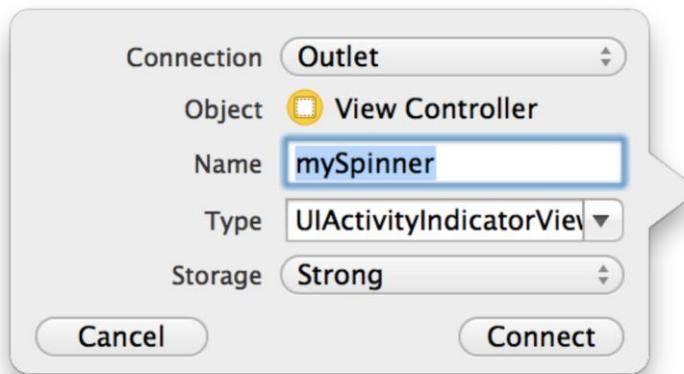
### **5.8.2 Ví Dụ**

**Bước1:** Tiếp tục với ví dụ của “UIWebView”, kéo thả đối tượng “Activity Indicator View”.



**Hình 5.106 Kéo thả Activity Indicator vào giao diện**

**Bước 2:** Map đối tượng “Activity Indicator View” với tên “mySpinner”.



**Hình 5.107 Ánh xạ đối tượng**

**Bước 3:** Thêm protocol <UIWebViewDelegate> trong file “ViewController.h”

```

9  #import <UIKit/UIKit.h>
10
11 @interface NATViewController : UIViewController
12 <UIWebViewDelegate>
13 {
14     IBOutlet UIWebView *myWebView;
15     IBOutlet UIActivityIndicatorView *mySpinner;
16 }
17
18 @end

```

### Hình 5.108 Thêm Protocol UIWebViewDelegate

**Bước 4:** Mở file “ViewController.m” thêm 2 phương thức sau.

```

- (void)webViewDidStartLoad:(UIWebView *)webView {
    //Spinner bắt đầu quay
    [mySpinner startAnimating];
}

```

**Giải thích:** phương thức này sẽ được gọi khi đối tượng “UIWebView” bắt đầu tải dữ liệu từ URL.

```

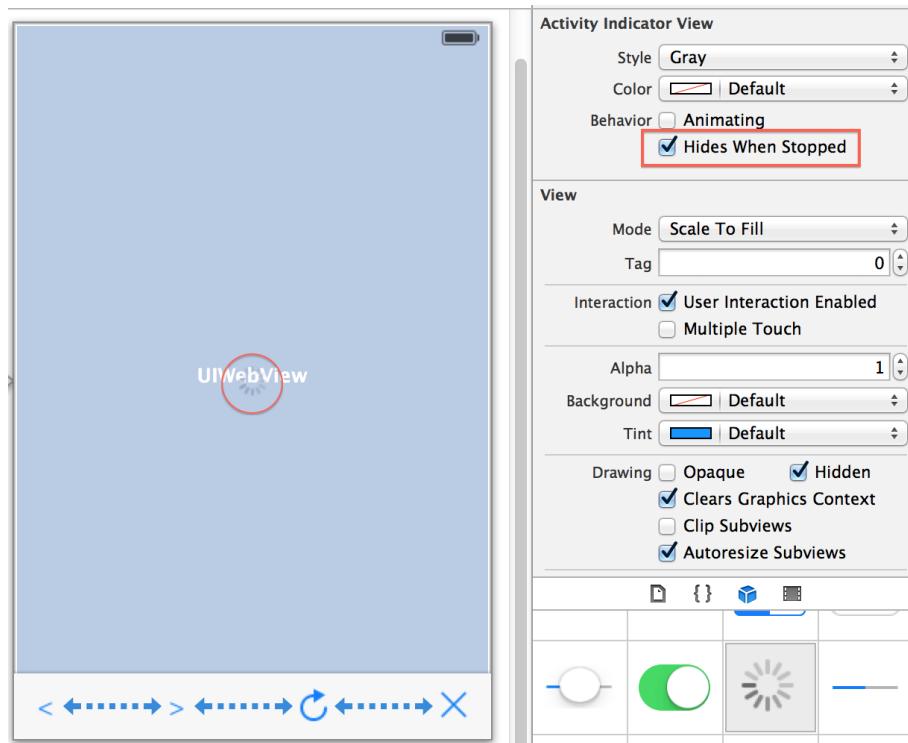
- (void)webViewDidFinishLoad:(UIWebView *)webView
{
    //Ngưng spinner
    [mySpinner stopAnimating];
    //Gán tiêu đề vào "UIWebViewController"
    NSString* title = [webView stringByEvaluatingJavaScriptFromString:
    @"document.title"];
    self.navigationItem.title = title;
}

```

**Giải thích:** phương thức này sẽ được gọi khi đối tượng “UIWebView” tải dữ liệu từ URL xong.

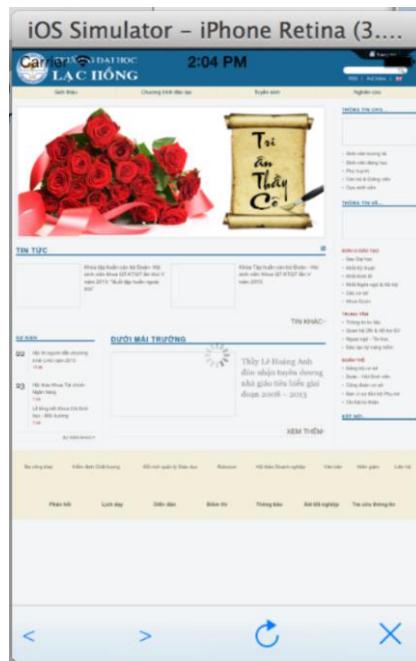
→ Thêm đoạn code sau vào cuối phương thức “myWebView.delegate = self;” để cho phép 2 phương thức trên đối tượng “UIWebView” quản lý và tự gọi.

**Bước 5:** Qua giao diện thiết lập thuộc tính “Hides When Stopped” cho đối tượng “Activity Indicator View”.



Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

**Bước 6:** chạy thử, kết quả:

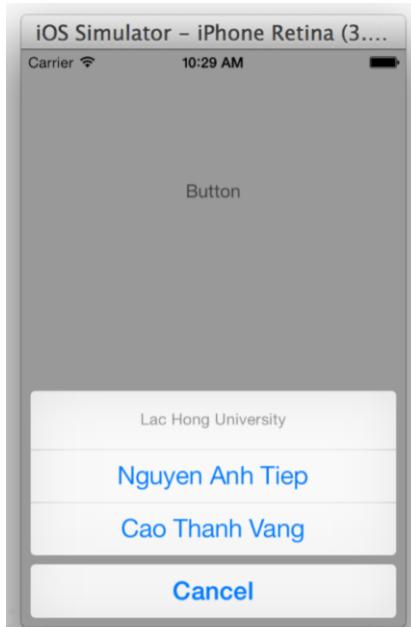


Hình 5.110 Kết quả

## 5.9 ACTIONSCHEET

### 5.9.1 Giới Thiệu

**Action Sheet** Cho phép người dùng xác nhận các hành động trên “ViewController”.



Hình 5.111 Action Sheet

### 5.9.2 Đặc Điểm

Khởi tạo Action Sheet:

```
initWithTitle:delegate:cancelButtonTitle:destructiveButtonTitle:otherButtonTitles
```

Thiết lập thuộc tính cho Action Sheet:

- **Delegate** : có giá trị là “nil” hoặc một đối tượng nào đó.
- **title**: Tiêu đề của Action Sheets.
- **visible**: ẩn hiện Action Sheet.
- **actionSheetStyle**: kiểu Action Sheet.

Cấu hình cho các button trên Action Sheet:

- **Thêm button**: – addButtonWithTitle:
- **Lấy tổng số button**: numberOfButtons
- **Lấy tiêu đề của button**: – buttonTitleAtIndex:
- **Lấy vị trí của button cancel**: cancelButtonIndex

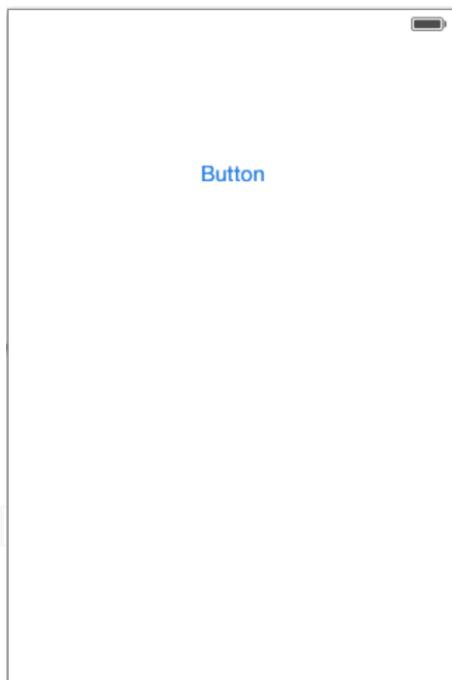
- **Lấy vị trí của button đầu tiên:** firstOtherButtonIndex

Hiển thị một Action Sheet:

- showInView:

### 5.9.3 Ví Dụ

**Bước 1:** Thiết kế giao diện như sau:



**Hình 5.112 Thiết kế giao diện**

**Bước 2:** Thêm protocol <UIActionSheetDelegate> và map (action) button với tên “touchBtn” → kết quả file “ViewController.h”

```
#import <UIKit/UIKit.h>

@interface NATViewController : UIViewController

<UIActionSheetDelegate> {

}

- (IBAction)touchBtn:(id)sender;

@end
```

**Bước 3:** Tại phương thức “touchBtn” khởi tạo đối tượng “Action Sheet”:

```
UIActionSheet *actionSheet = [[UIActionSheet alloc]
    initWithTitle:@"Lac Hong University"
    delegate:self
    cancelButtonTitle:@"Cancel"
    destructiveButtonTitle:nil
    otherButtonTitles:@"Nguyen Anh Tiep", @"Cao Thanh Vang", nil];
NSLog(@"%@",actionSheet.numberOfButtons);
```

### Hình 5.113 Khởi tạo Action Sheet

**Bước 4:** Hiển thị Action Sheet bằng cách thêm đoạn code bên dưới vào cuối phương thức “touchBtn”.

```
[actionSheet showInView:self.view];
```

### Hình 5.114 Hiển thị Action Sheet

**Bước 5:** Thêm phương thức “clickButtonAtIndex” để in ra button mà người dùng chọn.

```
- (void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex {
    NSLog(@"%@", [actionSheet buttonTitleAtIndex:buttonIndex]);
}
```

### Hình 5.115 Phương thức in button

**Bước 6:** Chạy thử, kết quả Output:

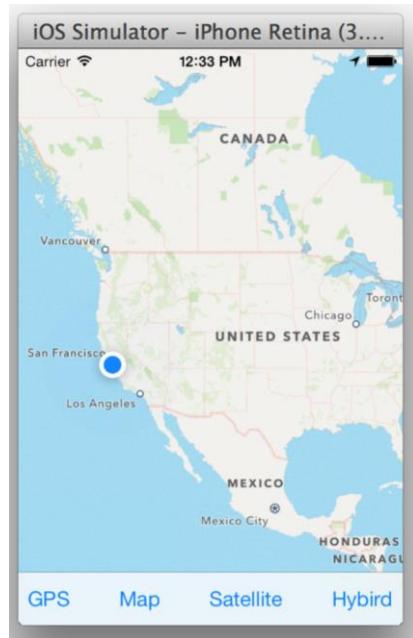
```
2013-11-23 11:16:13.845 DemoUIActionSheet[764:a0b]
Nguyen Anh Tiep
```

### Hình 5.116 Kết quả chạy thử

## 5.10 MK MAP VIEW

### 5.10.1 Giới Thiệu

**MK Map View** là một đối tượng cung cấp giao diện bản đồ.

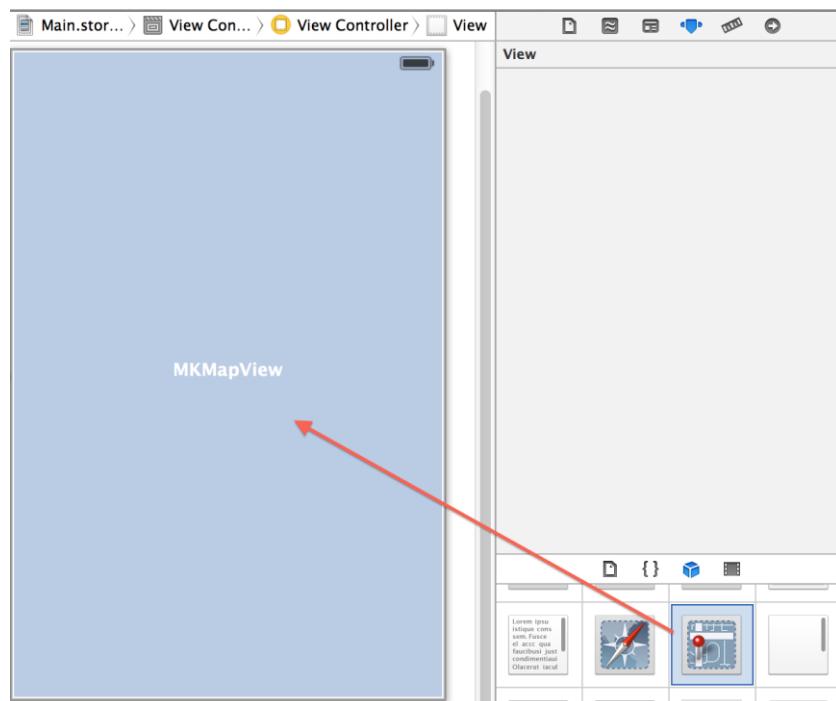


**Hình 5.117 Map View**

### 5.10.2 Ví Dụ

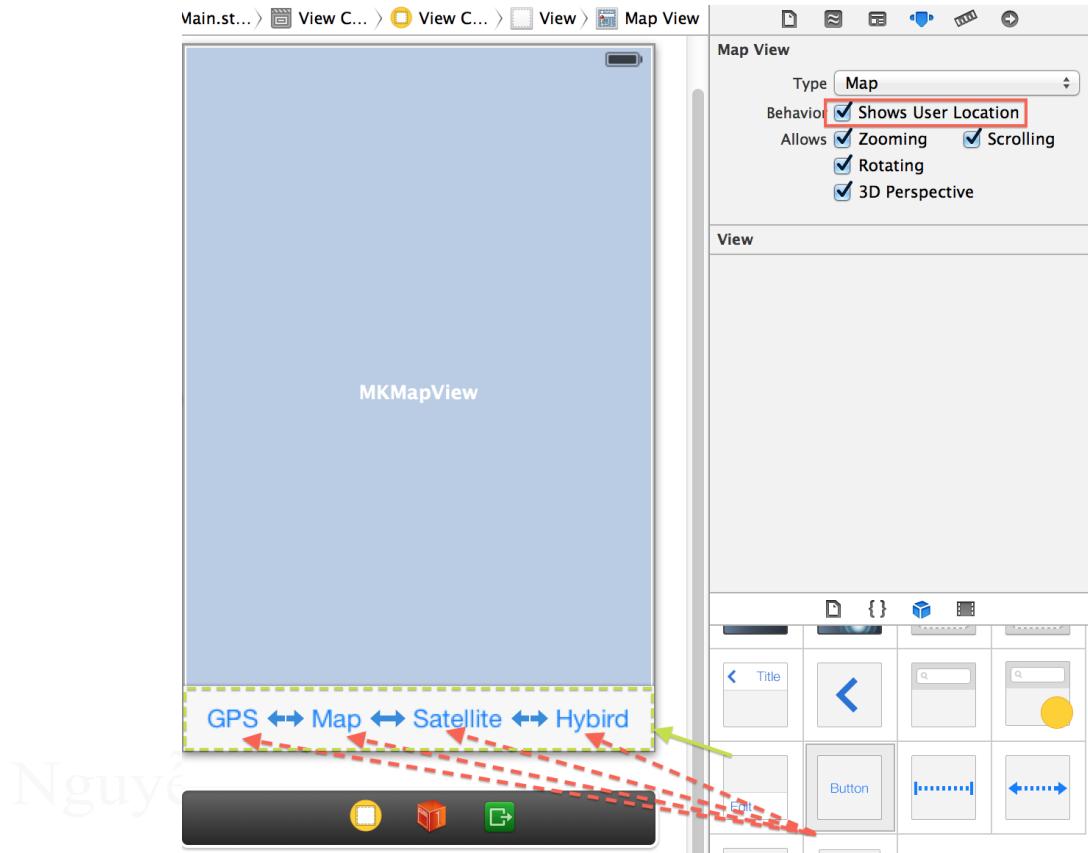
**Bước 1:** Thiết kế giao diện như sau:

→ Kéo đồi tượng “Map View” vào “View Controller”



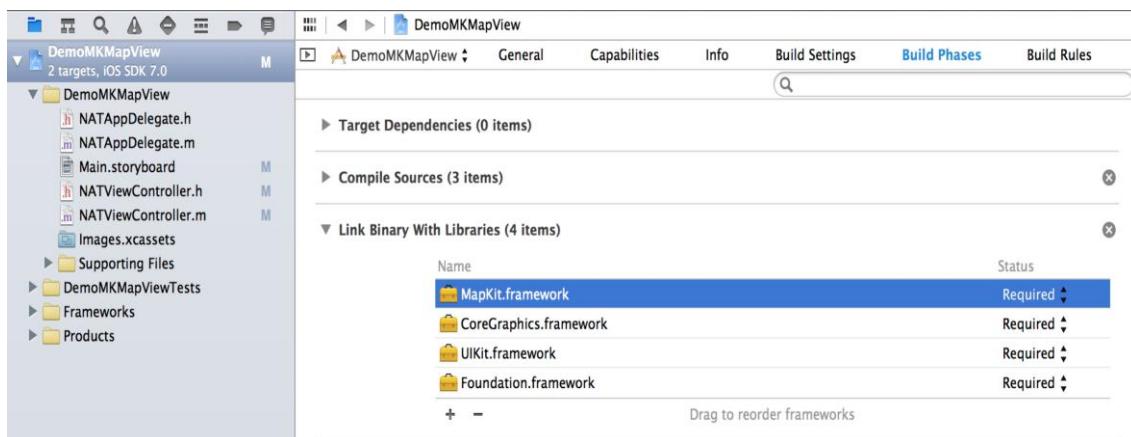
**Hình 5.118 Kéo thả Map View vào**

→ Kéo “Toolbar” và các “Bar button” vào “View Controller” → check thuộc tính “Show User Location” cho đối tượng “MKMapView”.



**Hình 5.119 Thiết kế giao diện**

→ Thêm thư viện “MapKit.framework” vào project:

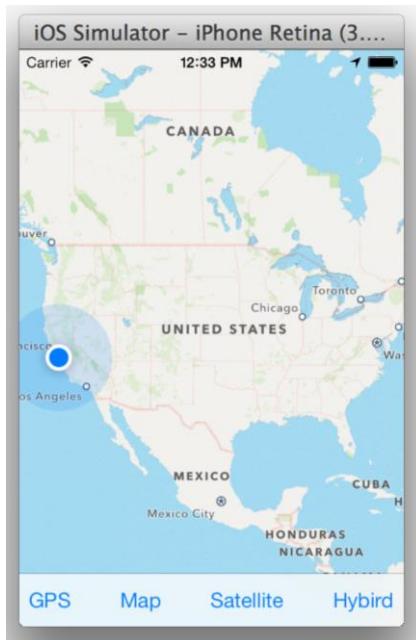


**Hình 5.120 Thêm thư viện**

→ Mở file “ViewController.h” import “MapKit.h”

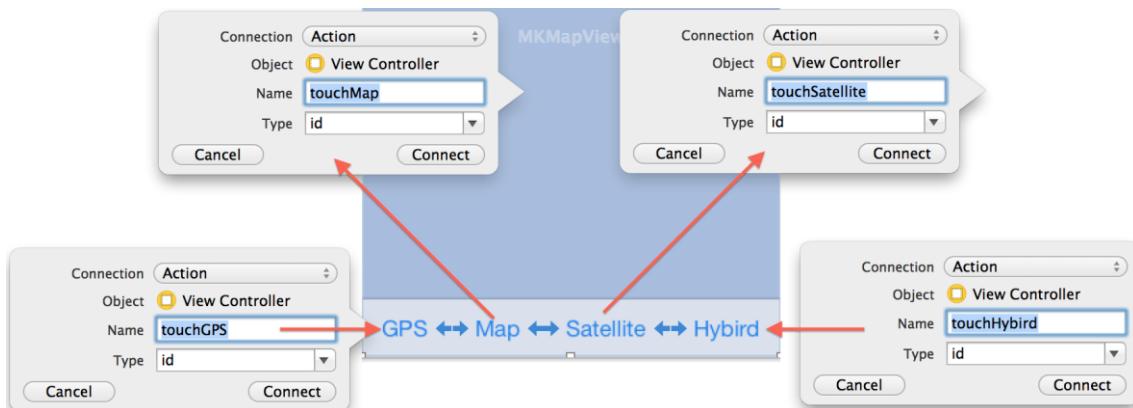
```
#import <MapKit/MapKit.h>
```

→ Chạy thử, kết quả:



Hình 5.121 Chạy thử

**Bước 2:** Map (action) các đối tượng: GPS (touchGPS), Map (touchMap), Satellite (touchSatellite), Hybird (touchHybird) vào “ViewController.h”.



Hình 5.122 Ánh xạ các button

**Bước 3:** Hiện thực phương thức “touchGPS”

→ Mở “ViewController.m” thêm đoạn code sau vào phương thức “touchGPS”:

```
myMapView.showsUserLocation = NO;  
myMapView.showsUserLocation = YES;
```

### Hình 5.123 Thêm code vào touchGPS

**Bước 4:** Hiện thực lại phương thức “touchMap”

→ Mở “ViewController.m” thêm đoạn code sau vào phương thức “touchMap”:

```
myMapView.mapType = MKMapTypeStandard;
```

**Bước 5:** Hiện thực lại phương thức “touchSatellite”

→ Mở “ViewController.m” thêm đoạn code sau vào phương thức “touchSatellite”:

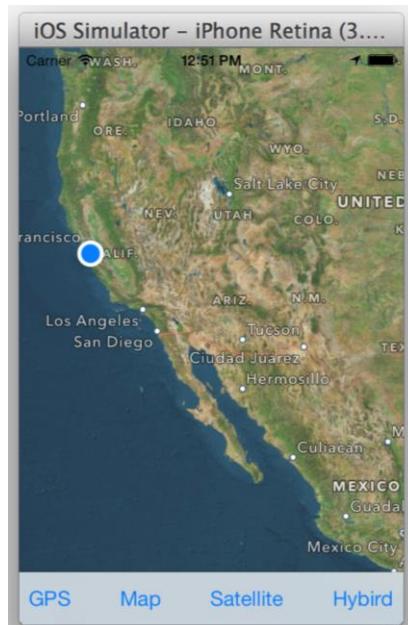
```
myMapView.mapType = MKMapTypeSatellite;
```

**Bước 6:** Hiện thực lại phương thức “touchHybird”

→ Mở “ViewController.m” thêm đoạn code sau vào phương thức “touchHybird”:

```
myMapView.mapType = MKMapTypeHybrid;
```

**Bước 7:** Chạy thử, kết quả:



Hình 5.124 Kết quả

## 5.11 TABLE VIEW CONTROLLER

### 5.11.1 Giới Thiệu

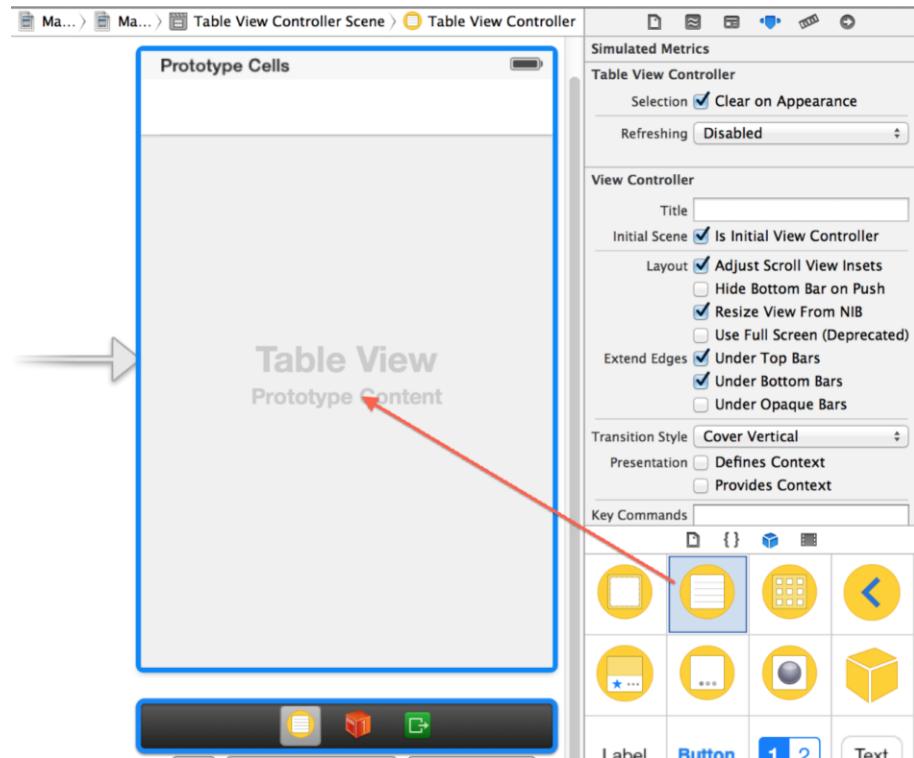
Table View Controller được sử dụng để hiển thị dữ liệu theo dạng bảng.



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

### 5.11.2 Ví Dụ

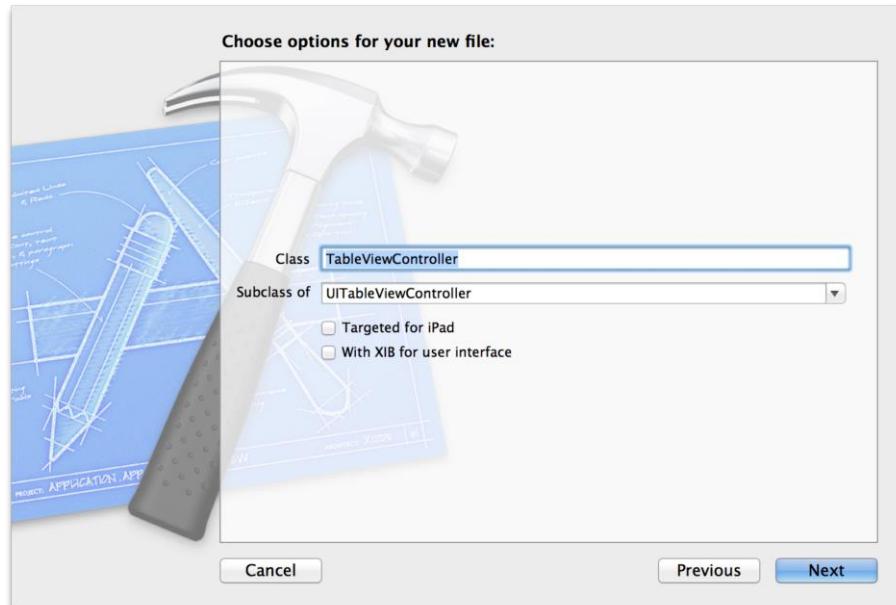
Bước 1: Kéo thả một “Table View Controller” sang giao diện thiết kế.



**Hình 5.126 Kéo thả Table View vào giao diện**

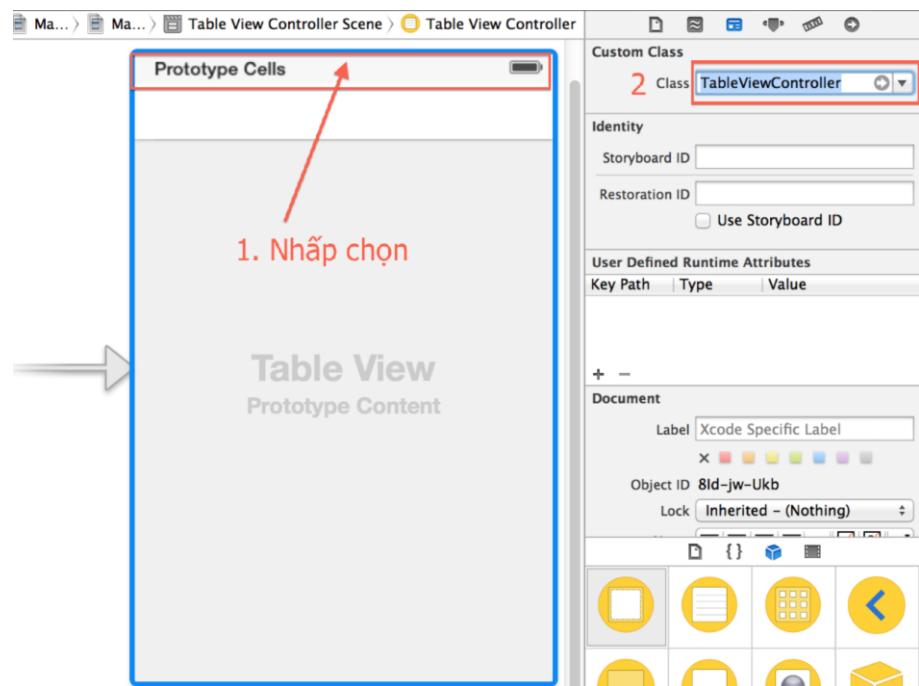
**Bước 2:** Tạo một lớp tên “Table View Controller” để gắn vào “Table View Controller” vừa kéo thả.

→ Nhấn Ctr + N để tạo 1 lớp → Chọn Objective – C class → Class: TableViewController → Subclass of: UITableViewController.



**Hình 5.127 Tạo class mới**

→ Qua giao diện gắn lớp vừa tạo vào “TableViewController”



Hình 5.128 Gắn class với Table View

**Bước 3:** Mở “TableViewController.m” bên dưới “@implementation...” khai báo 1 mảng “mangSinhVien”:

```
@implementation TableViewController  
NSMutableArray *mangSinhVien;
```

**Bước 4:** Khởi tạo mảng sinh viên trong phương thức “viewDidLoad”.

```
mangSinhVien = [[NSMutableArray alloc] initWithObjects:
```

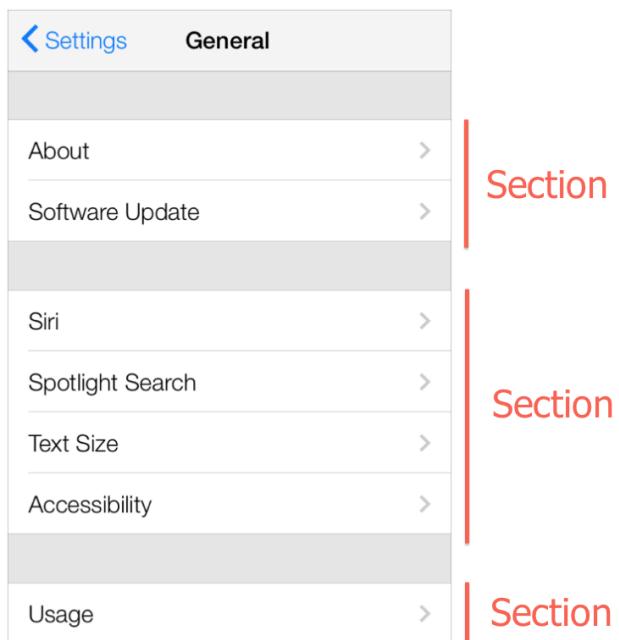
```
    @"Nguyen Anh Tiep",  
    @"Cao Thanh Vang",  
    @"Nguyen Xuan Thanh",  
    @"Nguyen Xuan Thao",  
    @"Nguyen Xuan Hoa",  
    @"Nguyen Hoai Nam",  
    @"Le Thi Bich Ha",
```

```
@ "Pham Thi Anh Nguyet",  
@ "Nguyen Tien Trung", nil];
```

**Bước 5:** Tại phương thức “numberOfSectionsInTableView” sửa lại như sau:

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView  
{  
    return 1;  
}
```

**Giải thích:** phương thức này cho phép chúng ta thiết lập số “Section” trong “Table View Controller”. Section có dạng như sau:



**Hình 5.129 Section**

**Bước 6:** Sửa lại phương thức “numberOfRowsInSection” như sau:

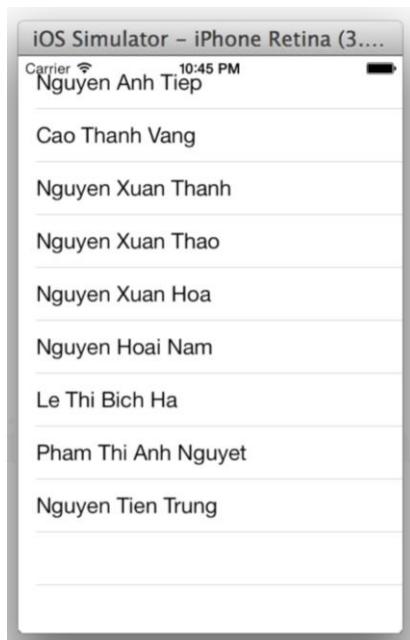
```
- (NSInteger)tableView:(UITableView *)tableView  
 numberOfRowsInSection:(NSInteger)section  
{  
    return mangSinhVien.count;  
}
```

**Giải thích:** phương thức này cho phép chúng ta thiết lập số dòng trong 1 Section, ở đây chính là số phần tử của mảng

**Bước 7:** tại phương thức “cellForRowAtIndexPath” bên dưới “// Configure the cell...” thêm đoạn code sau để hiển thị tên sinh viên lên “Table View Controller”:

```
NSInteger row = indexPath.row;  
cell.textLabel.text = mangSinhVien[row];
```

**Bước 8:** Cmd + r chạy thử, kết quả:



**Hình 5.130 Kết quả**

## 5.12 SEARCH BAR

### 5.12.1 Giới Thiệu

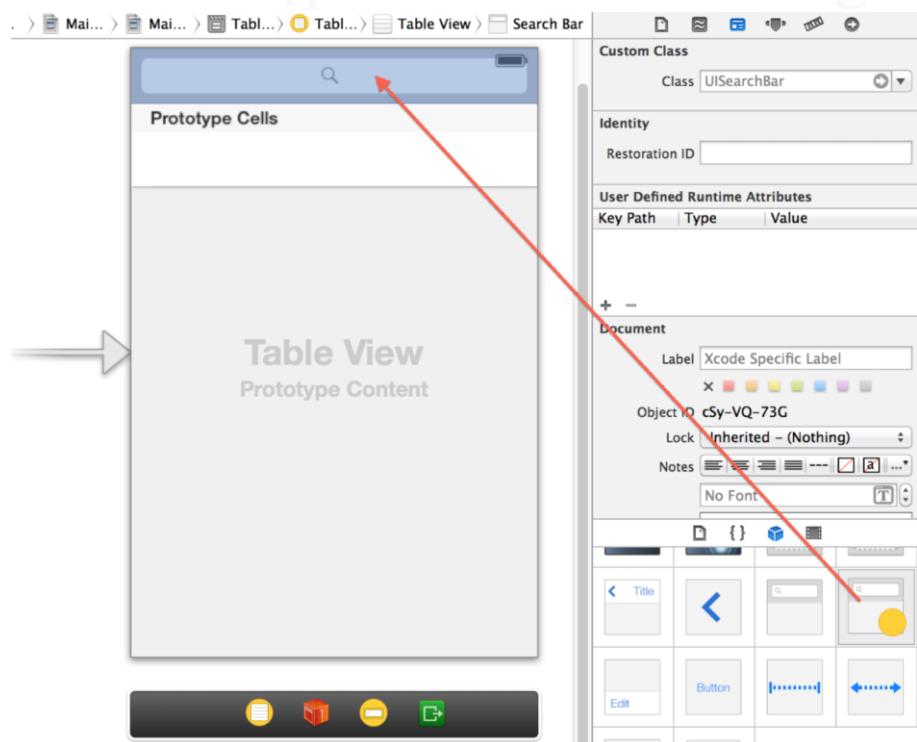
**Search Bar** được sử dụng để tìm kiếm trong “Table View Controller”.



**Hình 5.131 Search Bar**

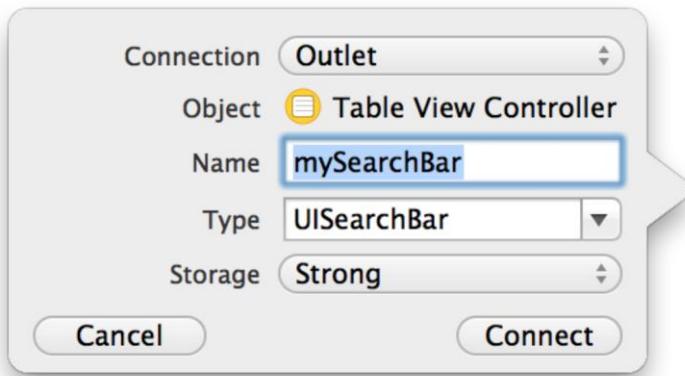
### 5.12.2 Ví Dụ

**Bước 1:** Tiếp tục với ví dụ của “Table View Controller”, thêm đối tượng “Search Bar and Search Display Controller” vào “Table View Controller”.



**Hình 5.132 Kéo thả Search Bar vào**

**Bước 2:** Map đối tượng Search Bar với tên “myearchBar”.



**Hình 5.133** Ánh xạ đối tượng

**Bước 3:** Bổ sung protocol <UISearchBarDelegate> để sử dụng được các phương thức của đối tượng Search Bar.

→ Mở “TableViewController.h” → bên dưới “@interface...” thêm vào như sau:

```
#import <UIKit/UIKit.h>

@interface TableViewController : UITableViewController

<UISearchBarDelegate>

{
    IBOutlet UISearchBar *myearchBar;
}

@end
```

**Bước 4:** Mở “TableViewController.m” → thêm vào phương thức “timKiemDuLieu” để tìm kiếm dữ liệu trong mảng “mangSinhVien”.

```
- (void)timKiemDuLieu {
    NSPredicate *resultsPredicate = [NSPredicate predicateWithFormat:@"SELF contains
[search] %@", myearchBar.text];
    mangKetQua = [[mangSinhVien filteredArrayUsingPredicate:resultsPredicate]
mutableCopy];
}
```

**Giải thích:** phương thức “filteredArrayUsingPredicate” được hỗ trợ sẵn trong mảng và dùng để tìm kiếm dữ liệu trong một mảng nào đó.

**Bước 5:** Thêm vào phương thức “textDidChange” để thực hiện tìm kiếm mỗi khi người dùng nhập từ khoá vào Search Bar.

```
- (void)searchBar:(UISearchBar *)searchBar textDidChange:(NSString *)searchText {  
    [self timKiemDuLieu];  
}
```

**Bước 6:** Sửa lại phương thức “numberOfRowsInSection” như sau:

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section  
{  
    if (tableView == self.tableView) {  
        return mangSinhVien.count;  
    } else {  
        [self timKiemDuLieu];  
        return mangKetQua.count;  
    }  
}
```

**Giải thích:** mặc định khi người dùng không tìm kiếm thì số dòng trong một “Section” chính là số phần tử của mảng “mangSinhVien”, nhưng khi người dùng sử dụng “Search Bar” để tìm kiếm thì số dòng lúc này chính là số phần tử trong mảng kết quả tìm kiếm được “mangKetQua”.

**Bước 7:** Sửa lại phương thức “cellForRowIndexPath” như sau:

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath  
{  
    static NSString *CellIdentifier = @"Cell";
```

```

UITableViewController *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];

cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];

// Configure the cell...

NSInteger row = indexPath.row;

if (tableView == self.tableView) {

    cell.textLabel.text = mangSinhVien[row];

} else {

    cell.textLabel.text = mangKetQua[row];

}

return cell;
}

```

**Giải thích:** mặc định khi mảng được sử dụng để hiển thị lên “Table View Controller” là mảng Sinh Viên “mangSinhVien” nhưng khi người dùng sử dụng “Search Bar” để tìm kiếm thì mảng hiển thị dữ liệu lên “Table View Controller” lúc này chính là mảng kết quả tìm kiếm “mangKetQua”.

**Bước 8:** Cmd + r chạy thử, kết quả:



**Hình 5.134 Kết quả**

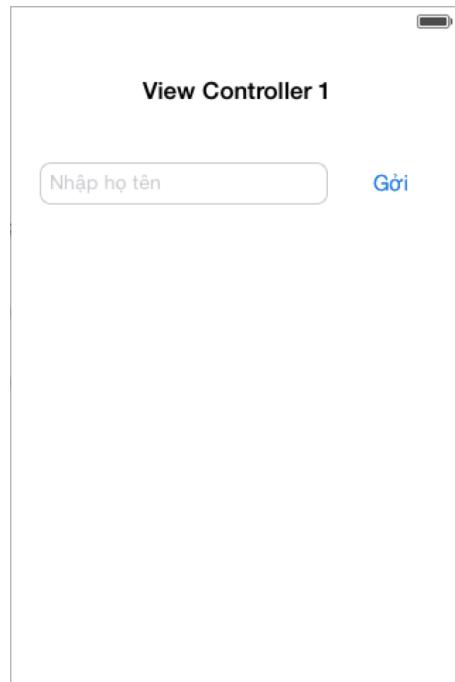
## **5.13 TRUYỀN DỮ LIỆU GIỮA CÁC VIEW**

### **5.13.1 Giới Thiệu**

Để truyền dữ liệu qua lại giữa các “View Controller” trong một ứng dụng, NSUserDefaults là một lớp có các phương thức hỗ trợ làm điều này.

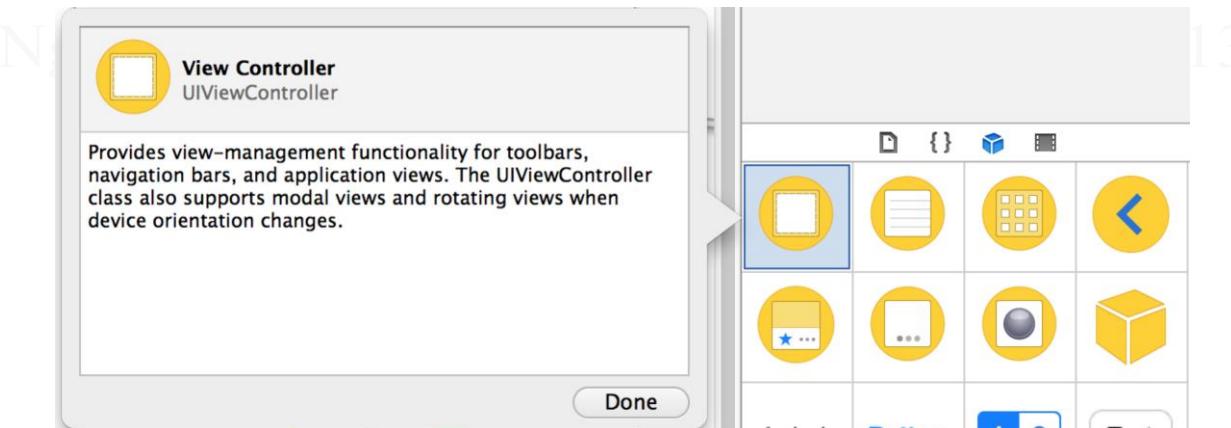
### **5.13.2 Ví Dụ**

**Bước 1:** Thiết kế giao diện cho như sau và Map (Outlet) đối tượng textField với tên “txtHoTen”.



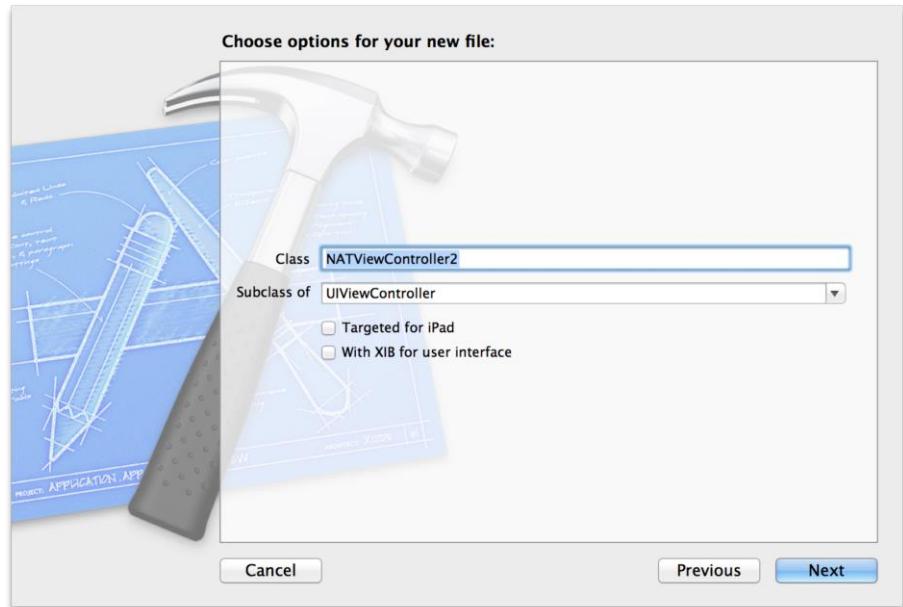
**Hình 5.135 Thiết kế giao diện**

**Bước 2:** Thêm một “View Controller” vào giao diện thiết kế.



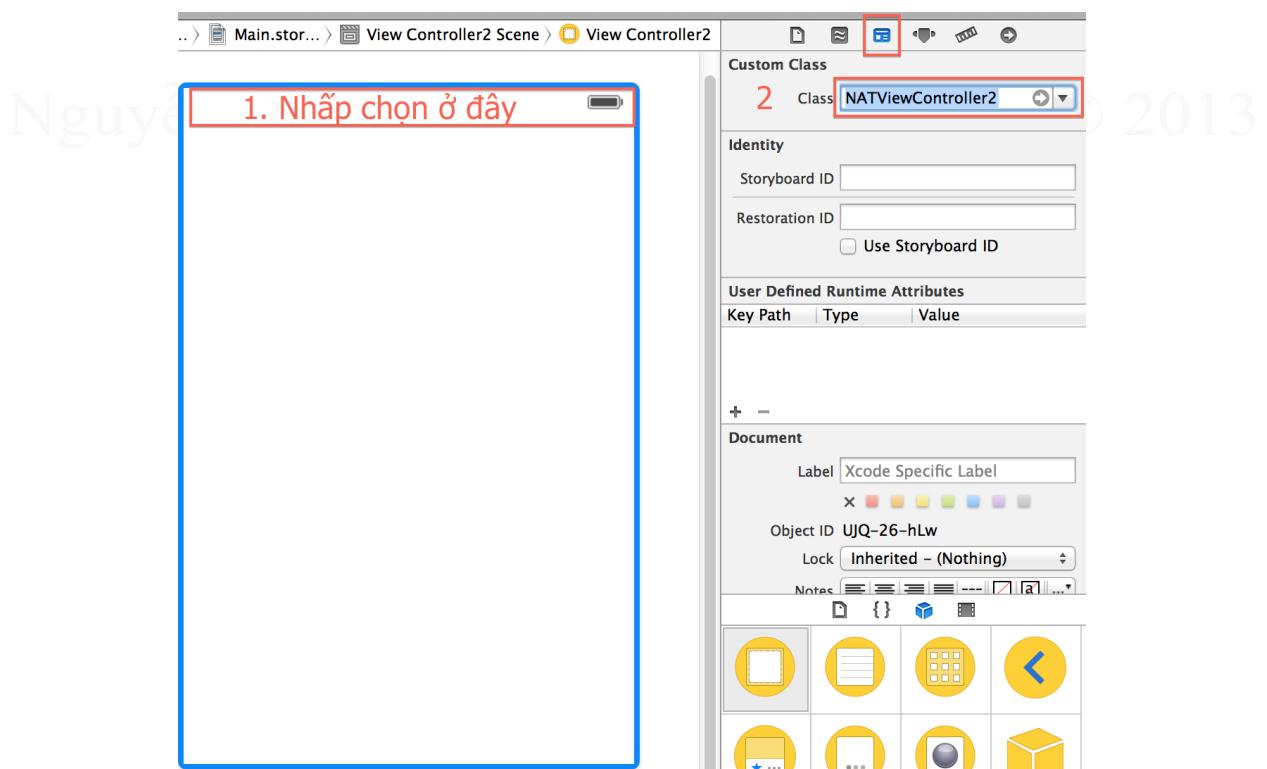
**Hình 5.136 Thêm View Controller vào**

**Bước 3:** Thêm một lớp có tên “NATViewController2” → nhấn Ctrl + N để tạo một lớp → Objective-C class → Subclass of: UIViewController.



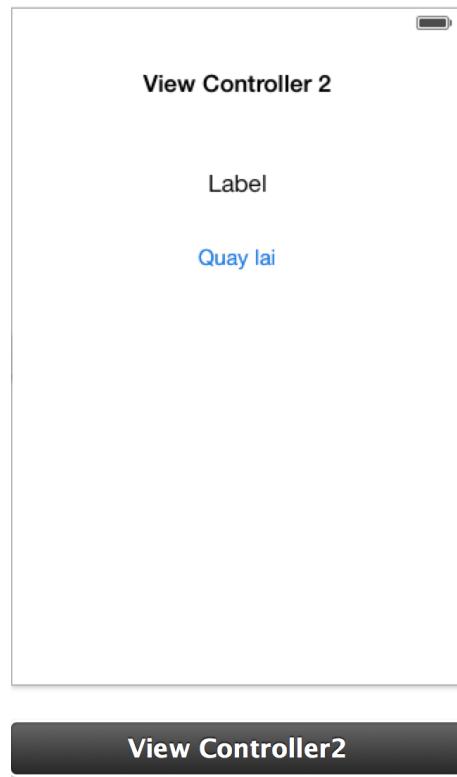
**Hình 5.137 Thêm class mới**

**Bước 4:** trỏ vào giao diện “View Controller” đã tạo ở bước 2.



**Hình 5.138 Gán class vào giao diện**

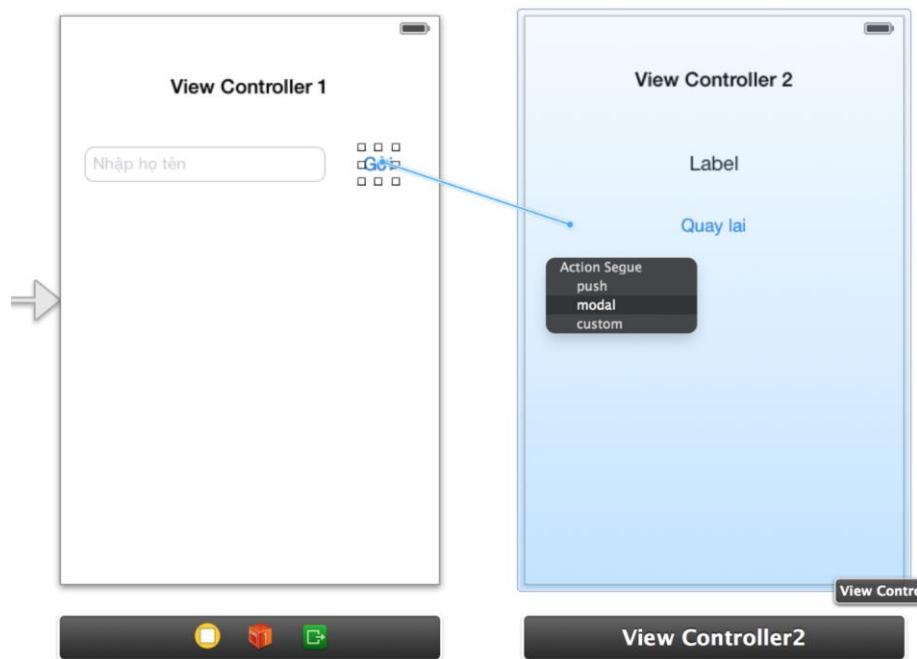
**Bước 5:** Thiết kế giao diện cho “View Controller 2” như sau và đổi tượng “Label” với tên “lblKetQua”.



View Controller2

**Hình 5.139 Thiết kế giao diện**

**Bước 6:** Tạo liên kết giữa 2 “View Controller” bằng cách nhấn giữ phím Ctrl sau đó kéo thả từ button “Gởi” trên “View Controller” thứ nhất sao “View Controller” thứ 2 → chọn modal.



**Hình 5.140 Tạo liên kết giữa 2 View Controller**

**Bước 7:** Làm tương tự bước 6 đối với button “Quay lai” trên “View Controller2”.

**Bước 8:** Thực hiện truyền dữ liệu từ “View Controller” thứ nhất.

→ Mở “ViewController.m” → thêm phương thức “prepareForSegue”

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {  
    [[NSUserDefaults standardUserDefaults] setObject:txtHoTen.text forKey:@"HoTen"];  
}
```

**Giải thích:** Phương thức “prepareForSegue” được gọi tự động trước khi chuyển sang một “View Controller” khác.

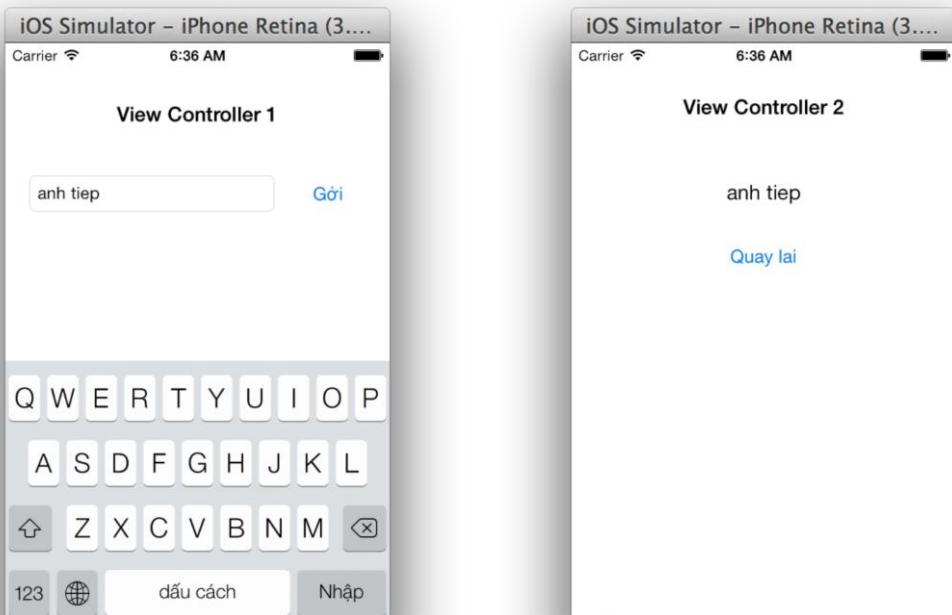
**Bước 9:** Thực hiện nhận dữ liệu từ “View Controller” thứ nhất.

→ Mở “NATViewController2.m” → trong phương thức “viewDidLoad” bổ sung đoạn code sau:

```
lblKetQua.text= [[NSUserDefaults standardUserDefaults] objectForKey:@"HoTen"];
```

**Bước 10:** Chạy thử, kết quả:

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013



**Hình 5.141 Kết quả**

## **CHƯƠNG VI**

### **HƯỚNG DẪN XÂY DỰNG PHẦN MỀM**

Qua các nội dung ở các chương trước, chắc hẳn bạn đã nắm được cách thức để xây dựng một ứng dụng, cũng như hiểu và sử dụng được một số đối tượng trong Xcode vào thiết kế và xây dựng ứng dụng trên iPhone. Tuy nhiên để bổ sung, cũng cố lại kiến thức cho bạn, cũng như giúp bạn có thể tự làm được một ứng dụng hoàn chỉnh từ cơ bản đến nâng cao, nhóm tác giả sẽ giới thiệu đến bạn quá trình thiết kế và xây dựng hai ứng dụng hoàn chỉnh dựa trên các đối tượng trong Xcode đã được giới thiệu để hệ thống lại kiến thức mà tài liệu đã cung cấp, cũng như cho bạn làm quen với việc xây dựng ứng dụng hoàn chỉnh.

Hai ứng dụng trong chương này sẽ được giới thiệu và hướng dẫn một cách chi tiết quá trình thiết kế và xây dựng để bạn có thể thực hiện theo một cách dễ dàng. Hi vọng rằng sau khi thực hiện xong hai ứng dụng này, bạn đã có thể tự tìm ý tưởng và xây dựng nên ứng dụng riêng cho mình.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## 6.1 PHẦN MỀM KIỂM TRA MÃ VIN

### 6.1.1 Giới Thiệu

Phần mềm Kiểm Tra Mã VIN là phần mềm giúp người dùng có thể kiểm tra mã VIN của xe hơi một cách dễ dàng. Từ đó người dùng có thể biết thêm thông tin về xe hơi theo mã VIN như hãng xe, năm sản xuất, nơi sản xuất, nhiên liệu...

### 6.1.2 Chuẩn Bị

- Ứng dụng sẽ được viết trên phiên bản XCode 4 (Xcode 4.6.3) hoặc Xcode 5 (với điều kiện đưa về dạng hỗ trợ xây dựng ứng dụng chạy cho các phiên bản iOS cũ, nội dung này sẽ được trình bày sau).
- Một tài khoản Google hoặc Facebook để đăng ký sử dụng dịch vụ OCR SDK API và bộ thư viện OCR (xem folder source đính kèm).
- Một số hình ảnh sử dụng cho phần mềm (xem folder source đính kèm).
- Cơ sở dữ liệu mã VIN (xem trong folder source).

### 6.1.3 Cấu Trúc Phần Mềm

Phần mềm Kiểm tra mã VIN có cấu trúc gồm 3 phần: cơ sở dữ liệu về mã VIN được lưu trữ bằng SQLite, giao diện nhập dữ liệu và hiển thị dữ liệu (sử dụng các đối tượng trong Xcode) và xử lý nhận diện mã VIN trong hình ảnh thông qua Cloud OCR SDK API của hãng ABBYY.

Cơ sở dữ liệu mã VIN sử dụng cơ sở dữ liệu mã VIN năm 2013 của hãng Ford. Cơ sở dữ liệu này sẽ được lưu trữ thông qua hệ quản trị cơ sở dữ liệu SQLite. Với việc sử dụng SQLite, cơ sở dữ liệu sẽ dễ dàng truy cập bằng các câu lệnh truy vấn của SQL như SELECT, CREATE, UPDATE ... Việc quản lý SQLite dễ dàng và đơn giản thông qua plugin SQLite Manager của Firefox và có thể chạy hoàn toàn độc lập mà không cần đến server.

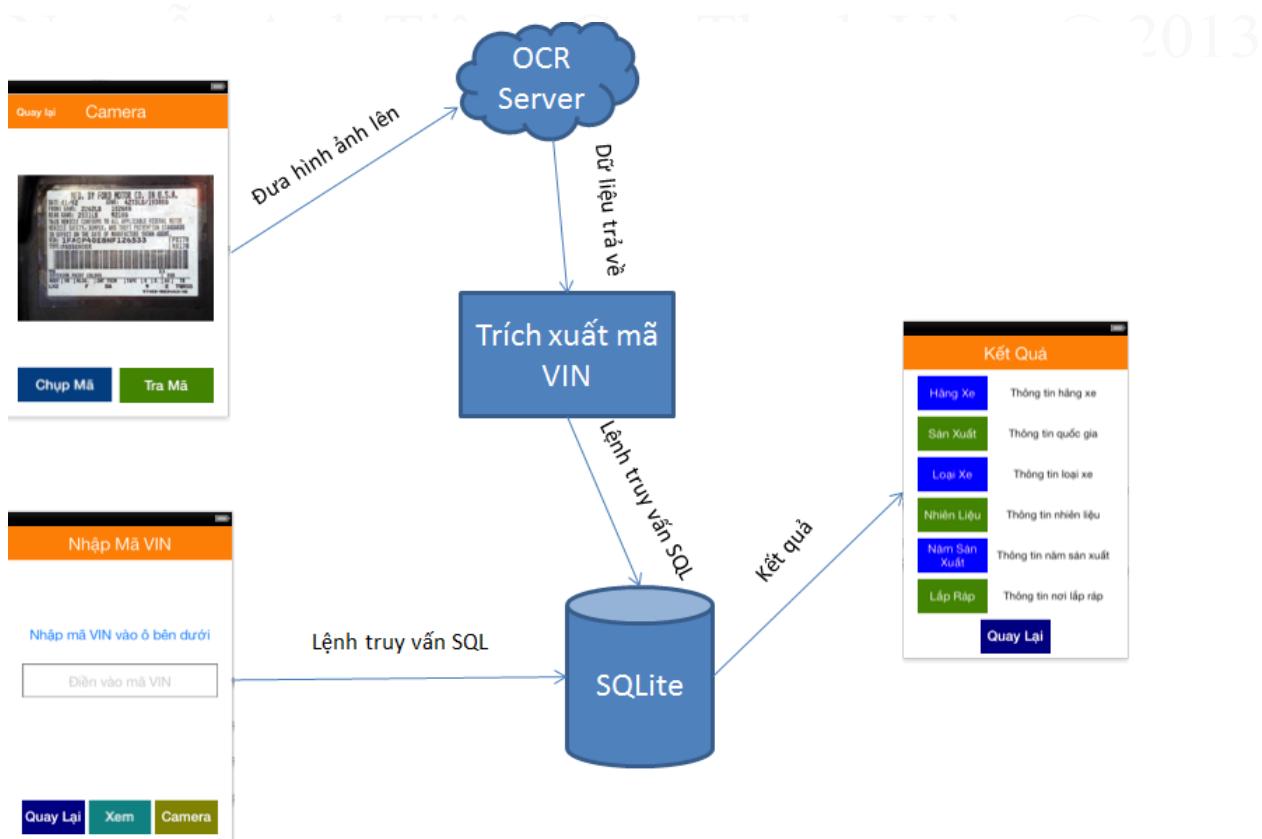
Giao diện nhập dữ liệu và hiển thị dữ liệu sử dụng các đối tượng của Xcode như Button, Label, Text Field, View Controller ... cũng như Camera của iPhone để nhận dữ liệu do người dùng nhập vào, đồng thời hiển thị dữ liệu đã được truy xuất từ SQLite.

Xử lý nhận diện mã VIN trong hình ảnh sử dụng OCR SDK API của hãng ABBYY. Hình ảnh sẽ được đưa lên server, server sẽ xử lý nhận diện và trả về một đoạn text nhận diện được từ hình ảnh. Phần mềm sẽ xử lý đoạn text trả về từ server để tách mã VIN ra và đưa vào SQLite để tra cứu dữ liệu.

#### 6.1.4 Cơ Chế Vận Hành

Phần mềm hoạt động theo hai cơ chế chính. Cơ chế thứ nhất là người dùng sẽ nhập vào mã VIN bằng tay, chương trình sẽ khởi tạo các câu lệnh truy vấn SQL và gửi đến SQLite để truy xuất dữ liệu và đem kết quả hiển thị ra giao diện.

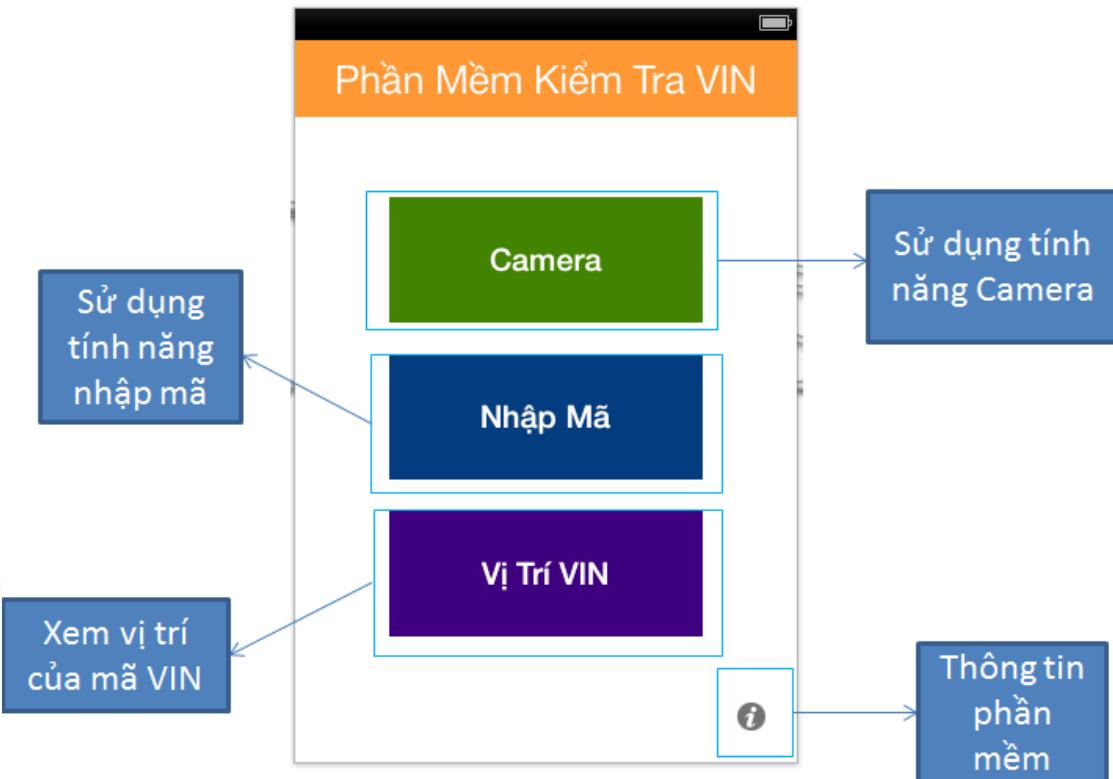
Cơ chế thứ hai sẽ cho phép người dùng sử dụng camera của iPhone chụp lại mã VIN và đưa hình ảnh đó lên OCR Server. Server sẽ xử lý và trả về một đoạn text kết quả sau khi nhận diện ký tự trong hình ảnh. Phần mềm sẽ phải xử lý đoạn text trả về để lọc ra được mã VIN. Nếu hình ảnh chính xác, mã VIN sẽ được lấy ra và tạo câu lệnh truy vấn gửi đến SQLite. SQLite sẽ truy xuất dữ liệu và trả về kết quả. Kết quả sẽ hiển thị ra giao diện.



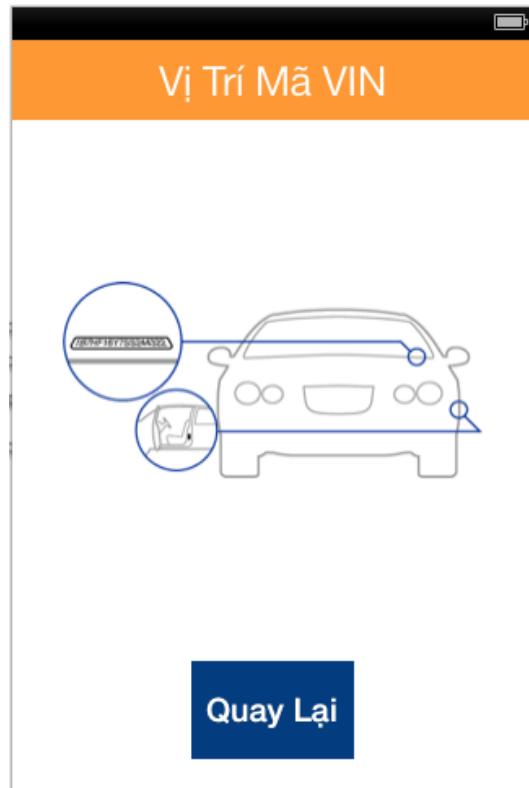
Hình 6.1 Cơ chế hoạt động của phần mềm

### 6.1.5 Tính Năng

Phần mềm Kiểm tra mã VIN cho phép người dùng có thể kiểm tra mã VIN của xe, từ đó có được một số thông tin về xe nhưng hãng xe, loại xe, nơi sản xuất, năm sản xuất... Ngoài ra phần mềm cũng hỗ trợ người dùng tìm kiếm theo các phương án như nhập vào số VIN hoặc sử dụng camera của iPhone để chụp lại mã VIN rồi tra cứu.

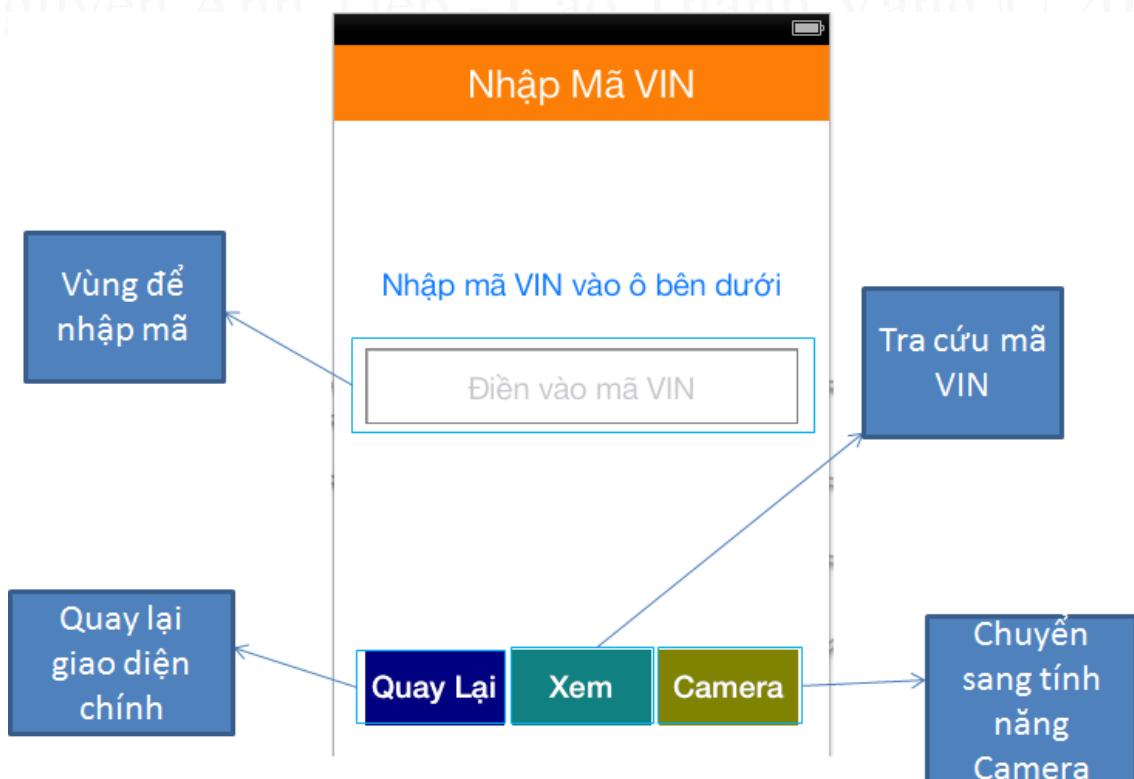


Hình 6.2 Giao diện tổng quan của phần mềm

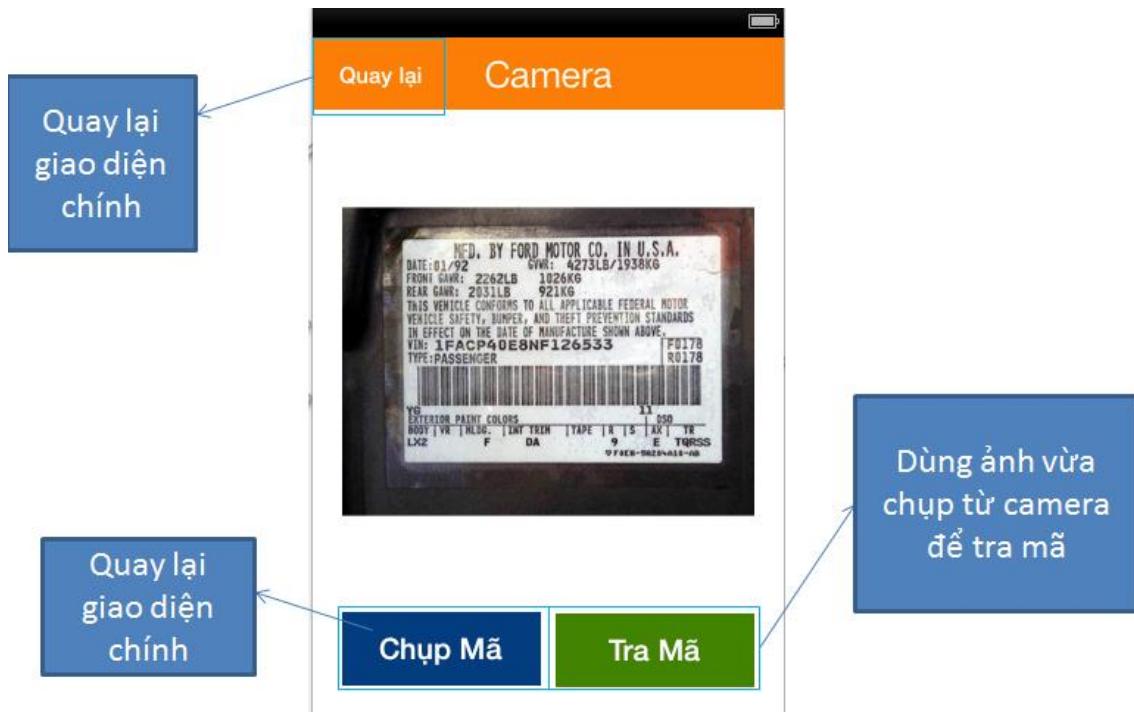


Hình 6.3 Vị trí mã VIN

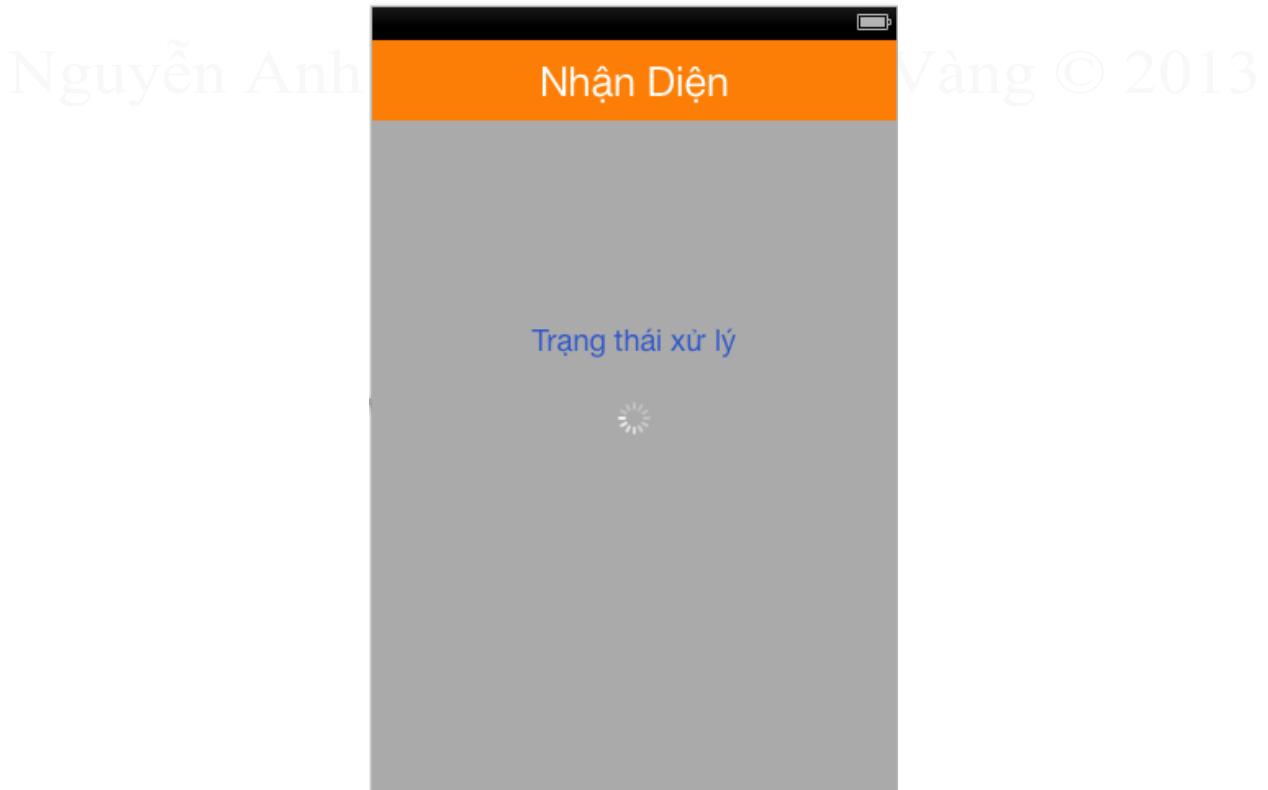
Nguyễn Anh Tiết - Cao Thanh Văn © 2013



Hình 6.4 Tính năng nhập mã VIN để tra cứu



**Hình 6.5 Giao diện camera**



**Hình 6.6 Giao diện xử lý hình ảnh**



**Hình 6.7 Giao diện kết quả**

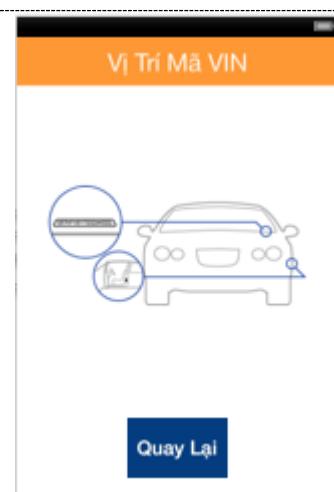
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

### 6.1.6 Tiến Hành

Ứng dụng bao gồm các giao diện độc lập được liên kết lại với nhau thành một khối thống nhất. Sau đây là các giao diện của phần mềm.



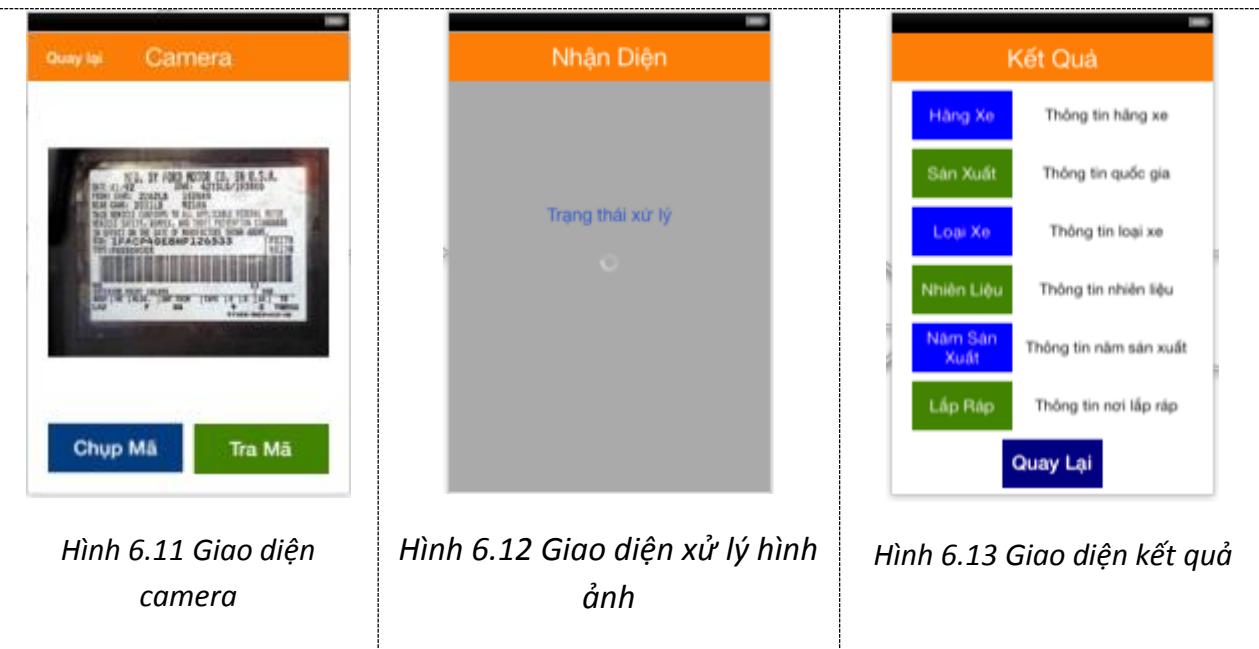
**Hình 6.8 Giao diện chính**



**Hình 6.9 Giao diện vị trí mã VIN**



**Hình 6.10 Giao diện nhập mã VIN**

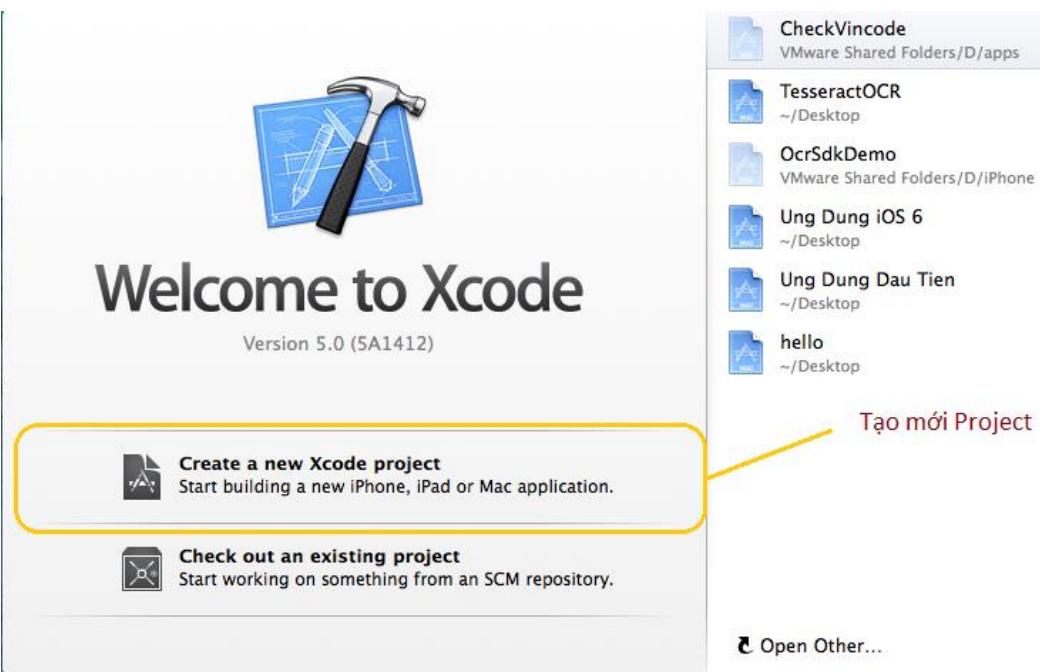


Hình 6.11 Giao diện camera

Hình 6.12 Giao diện xử lý hình ảnh

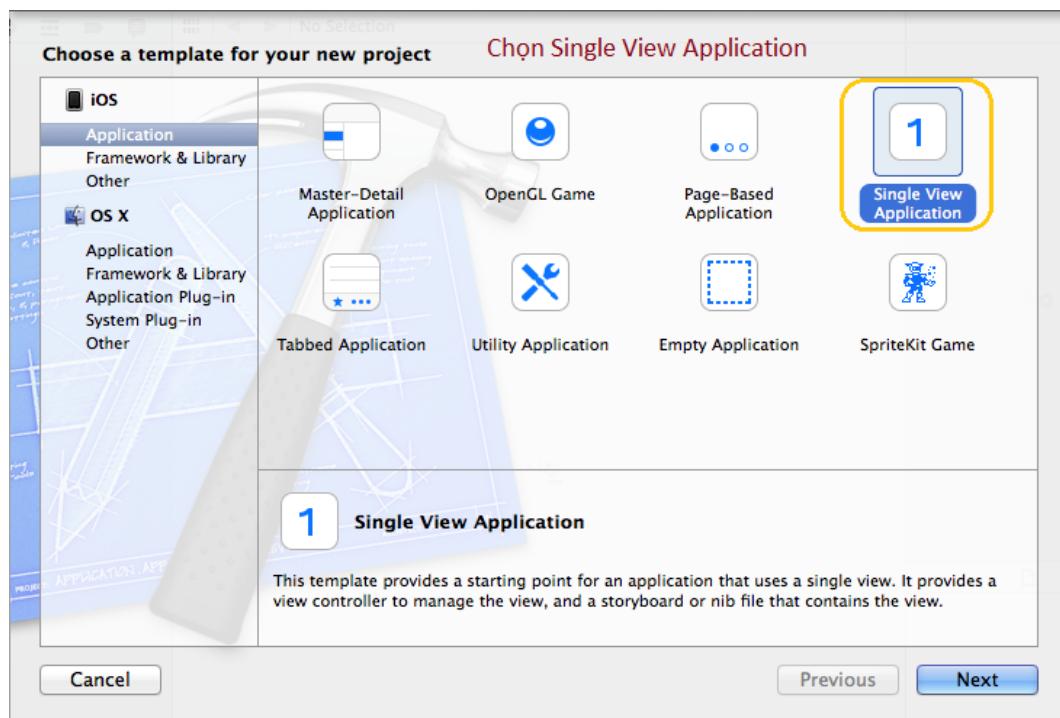
Hình 6.13 Giao diện kết quả

### Bước 1: Tạo project mới.



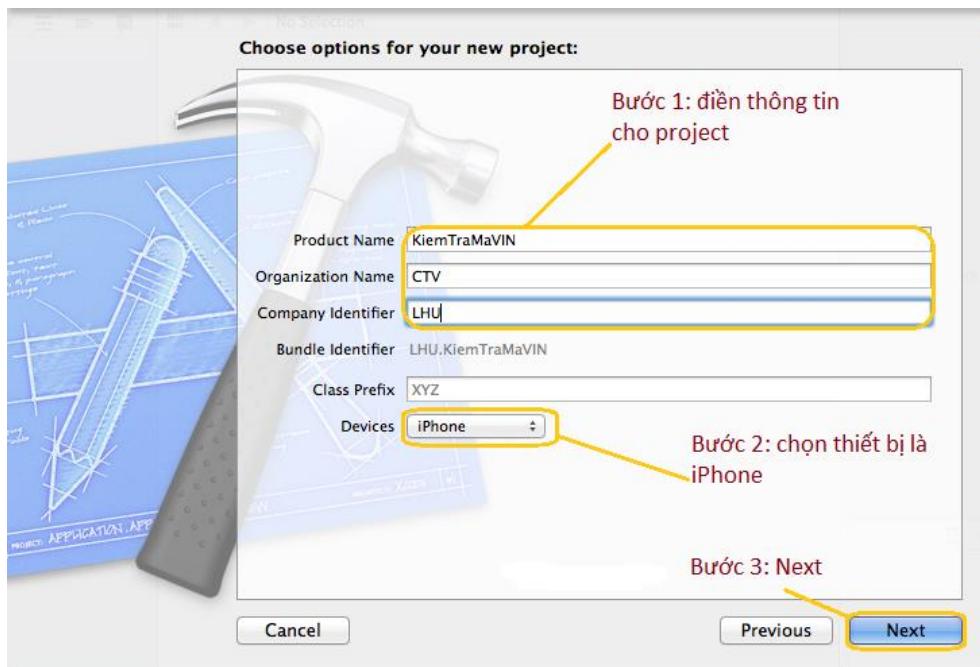
Hình 6.14 Tạo project

→ Chọn Single View Application.



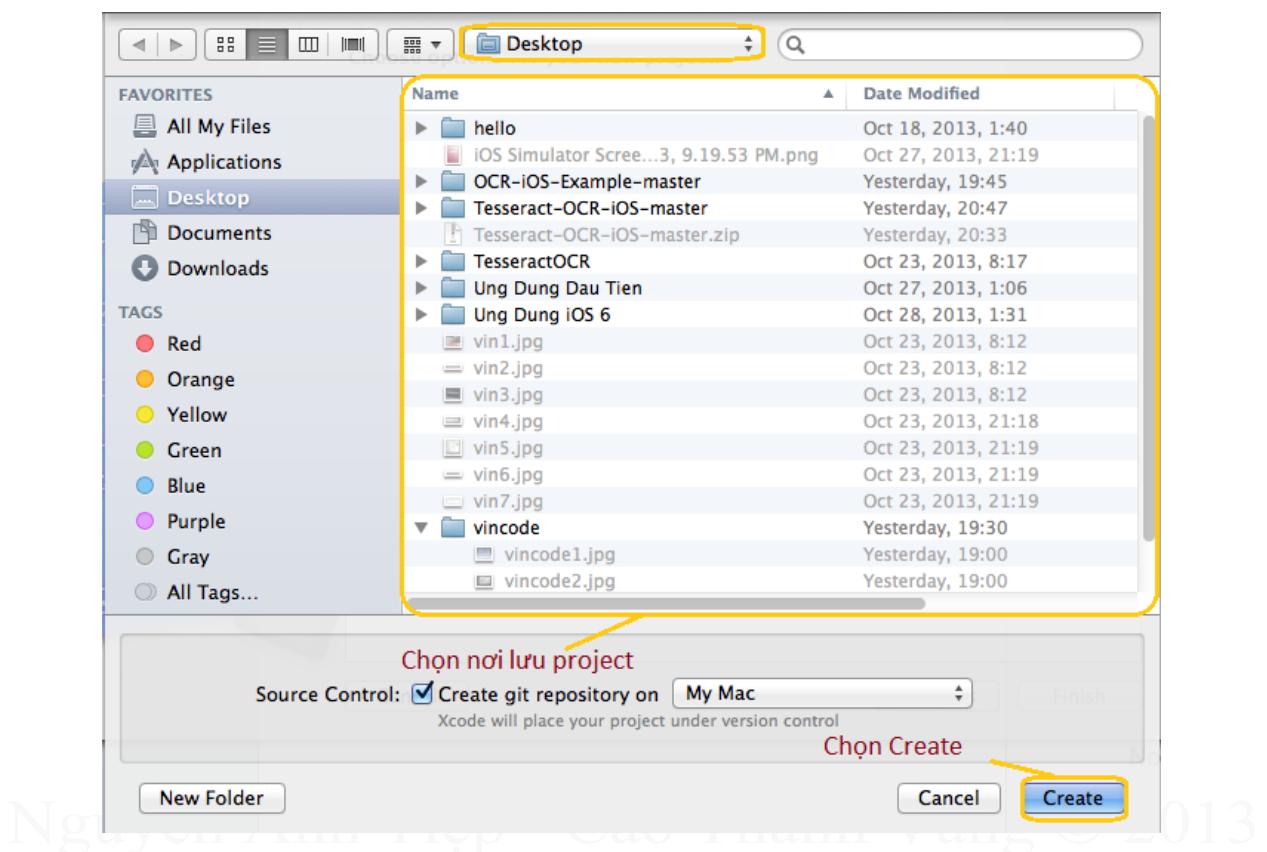
**Hình 6.15 Single View Application**

→Đặt tên project là KiemTraMaVIN. Phần Organization Name và Company Identifier các bạn điền theo ý mình. Device chọn là iPhone.



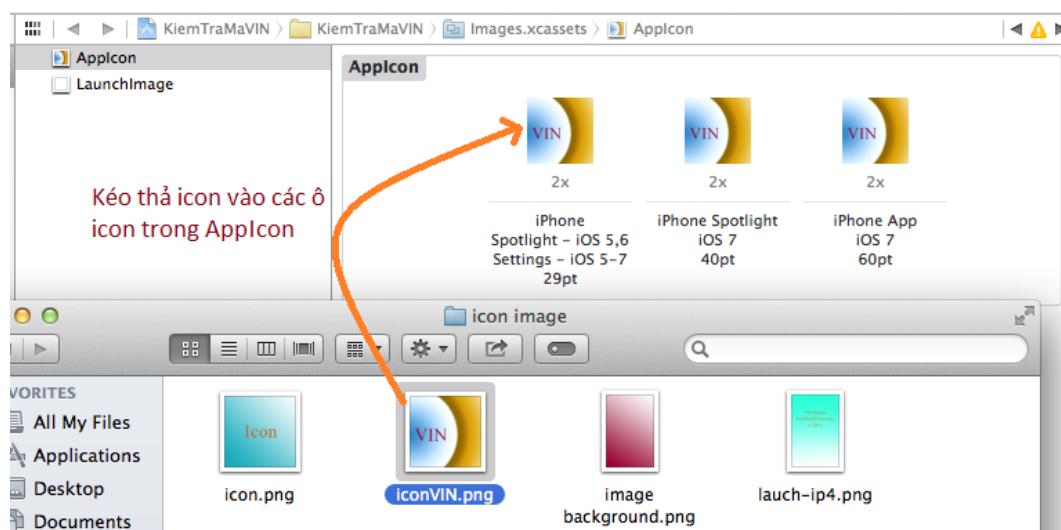
**Hình 6.16 Tùy chọn thông số cho project**

→ Chọn vị trí lưu project.

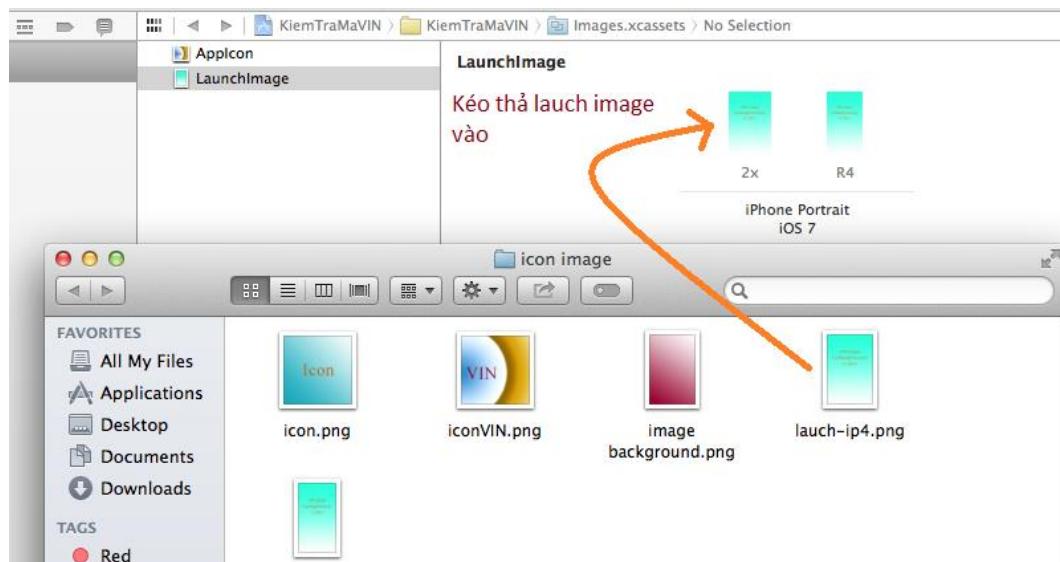


Hình 6.17 Chọn vị trí lưu

→ Chèn icon, launch image cho ứng dụng.



Hình 6.18 Bổ sung icon

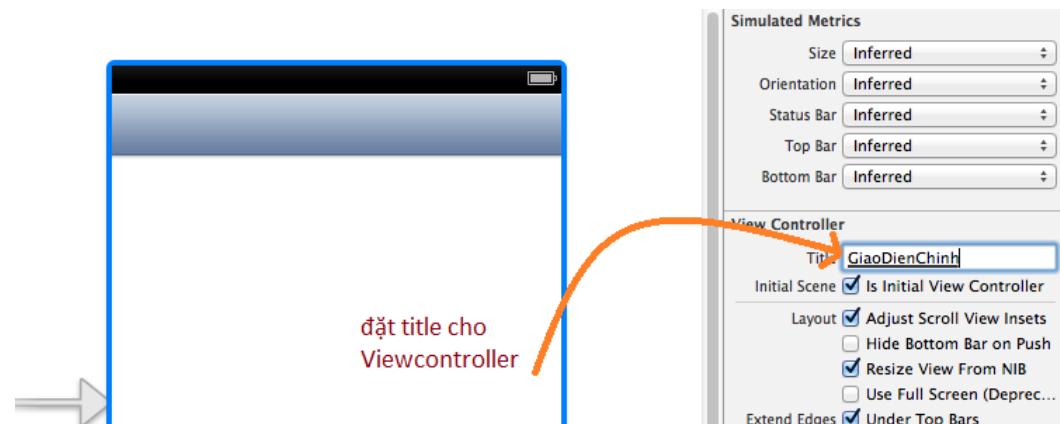


**Hình 6.19 Bổ sung Launch image**

→ Tùy chỉnh project để có thể chạy được trên các iOS version khác nhau (Việc này làm trước hay sau khi hoàn thành project đều được, xem thêm nội dung chương VIII).

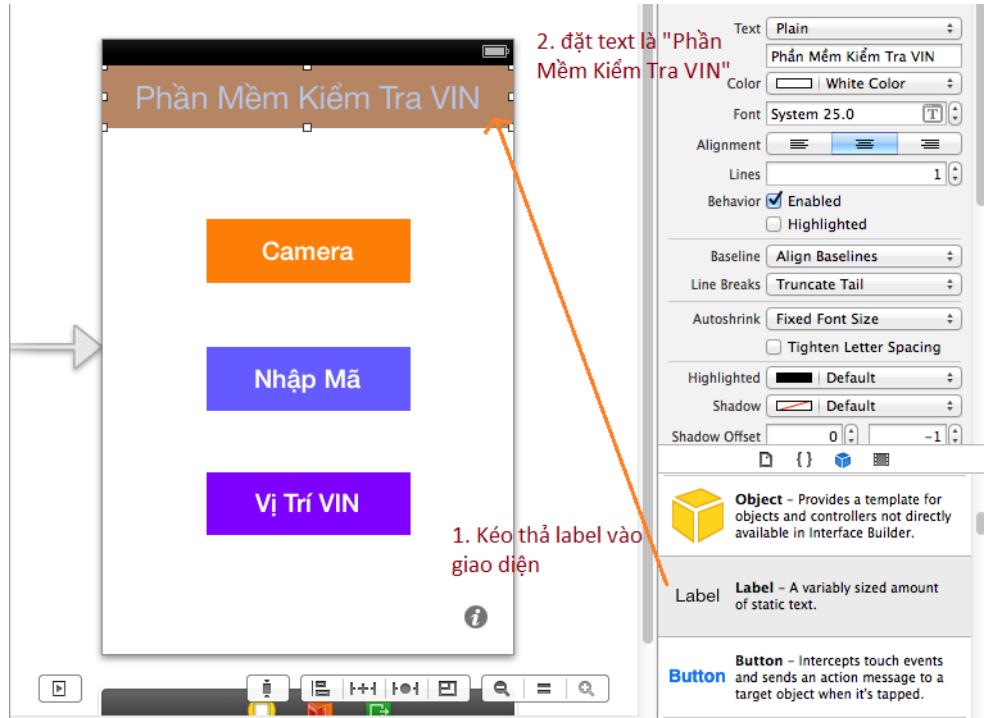
**Bước 2:** Thiết kế giao diện chính.

→ Đặt title cho Viewcontroller giao diện là GiaoDienChinh.



**Hình 6.20 Đặt Title cho Viewcontroller**

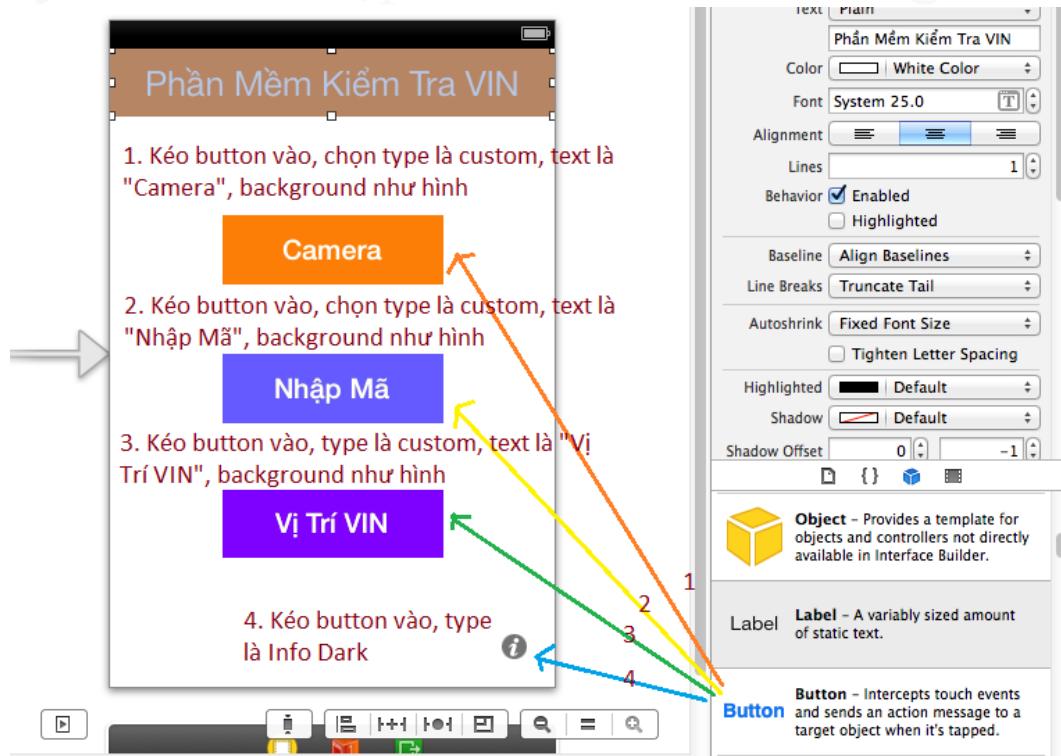
→ Kéo thả label vào giao diện và đặt text là “Phần Mềm Kiểm Tra Mã VIN”.



**Hình 6.21 Kéo Label vào giao diện**

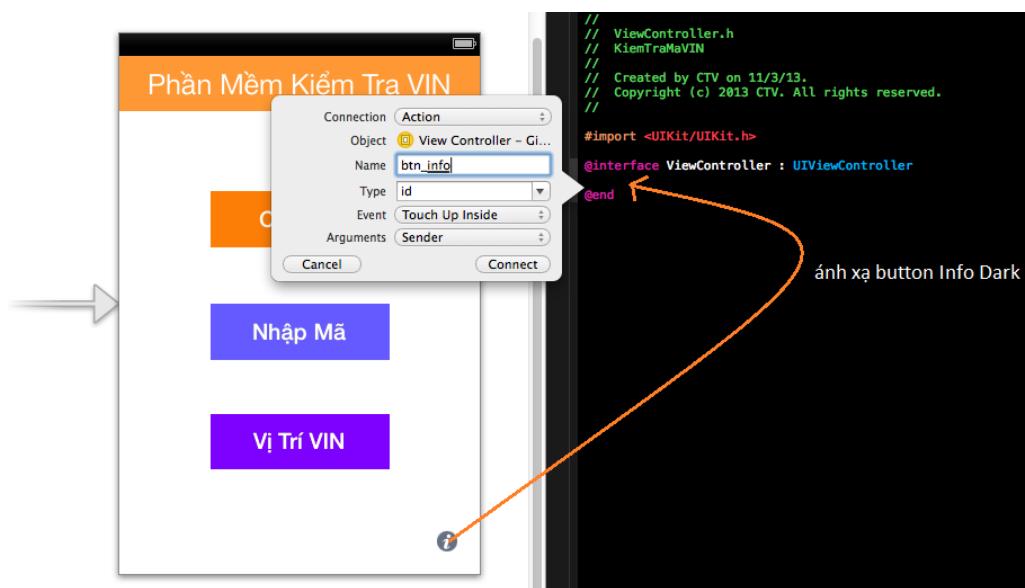
→Kéo button vào giao diện

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013



**Hình 6.22 Kéo Button vào giao diện**

→ Ánh xạ button Info Dark.



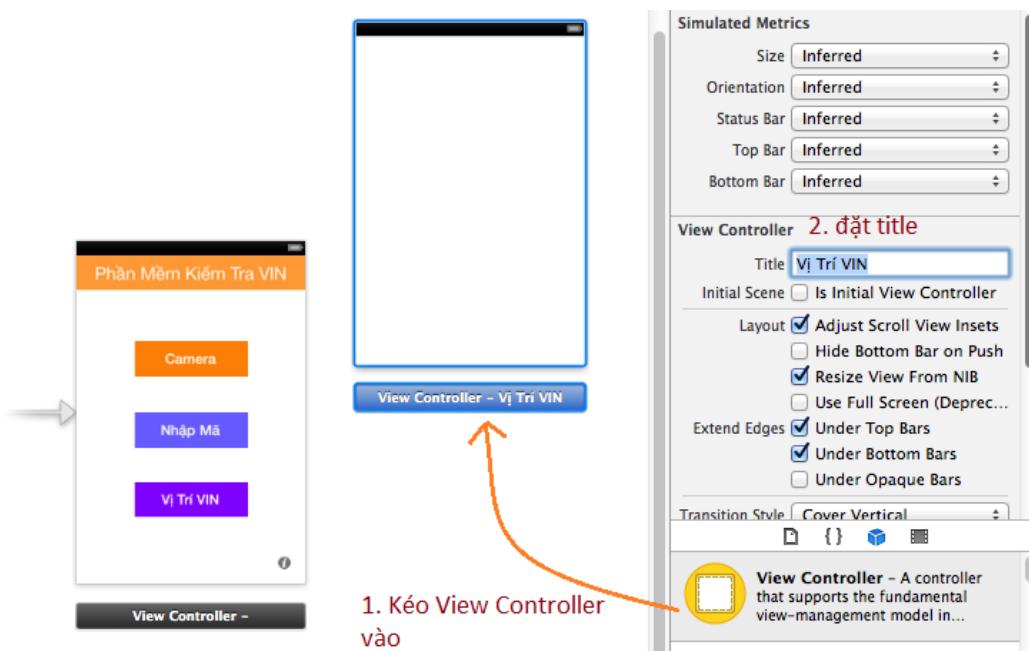
Hình 6.23 Ánh xạ button Info

→ Viết code cho button Info Dark

```
- (IBAction)btn_info:(id)sender {
    UIAlertView *InfoNotice = [[UIAlertView alloc]initWithTitle:@"Thông Tin Phần Mềm" message:@"Phần mềm tra cứu mã VIN \n Lac Hong University © 2013" delegate:self cancelButtonTitle:@"Đồng Ý" otherButtonTitles:nil, nil];
    [InfoNotice show];
}
@end
```

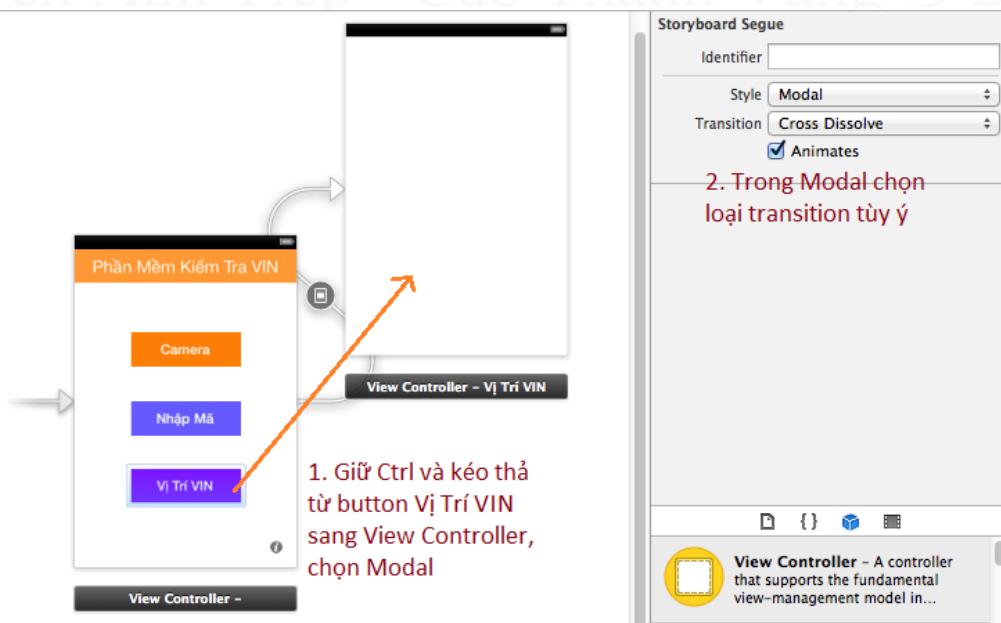
Hình 6.24 Viết code cho button

Bước 3: Kéo thả thêm một Viewcontroller đặt title là “Vị Trí VIN”.



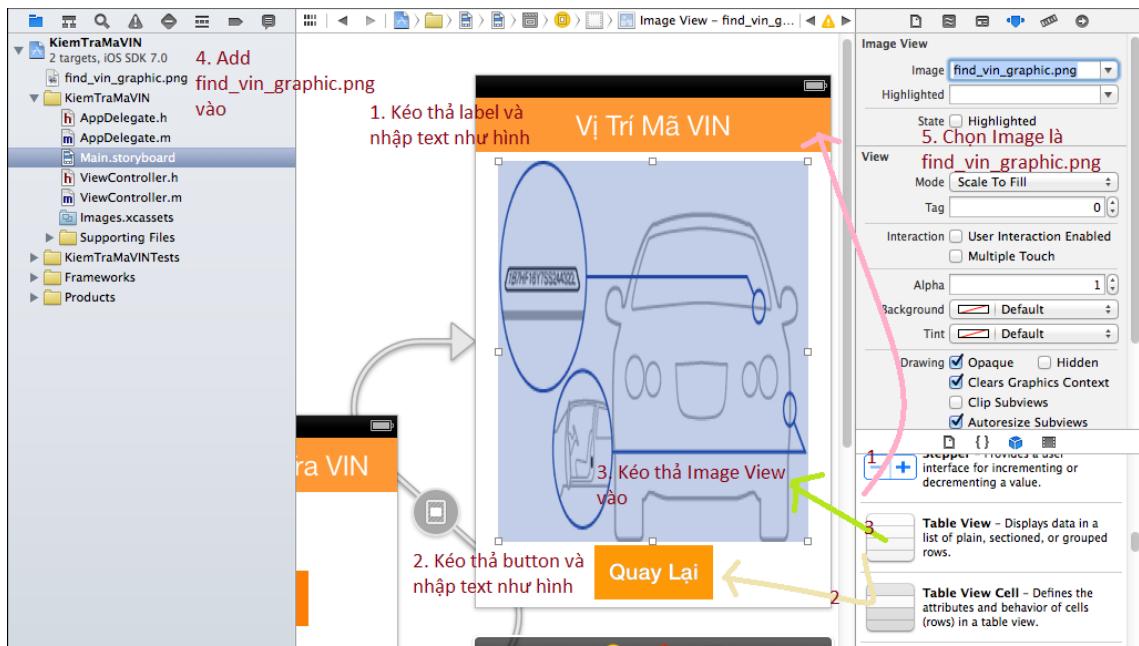
**Hình 6.25 View Controller Vị Trí VIN**

→Tạo liên kết chuyển đổi view sao cho khi click vào button Vị Trí VIN trên View Controller - Giao Diện Chính sẽ chuyển sang View Controller-Vị Trí VIN.



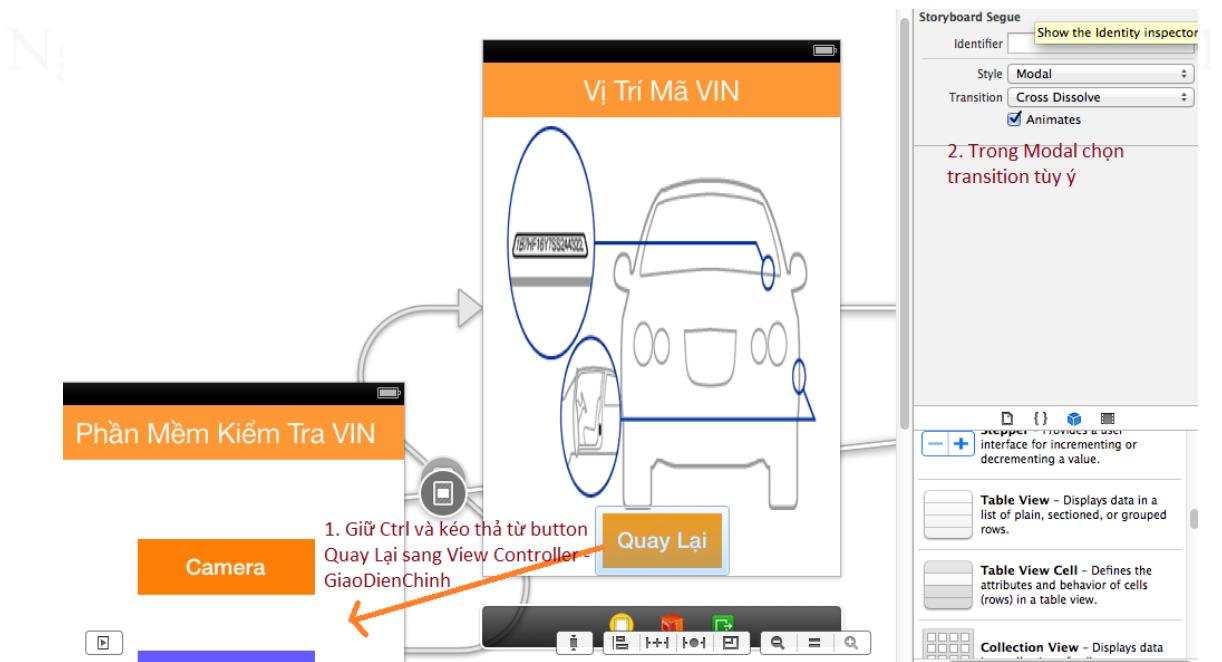
**Hình 6.26 Tạo liên kết giữ hai view**

→Thiết kế giao diện cho View Controller – Vị Trí VIN



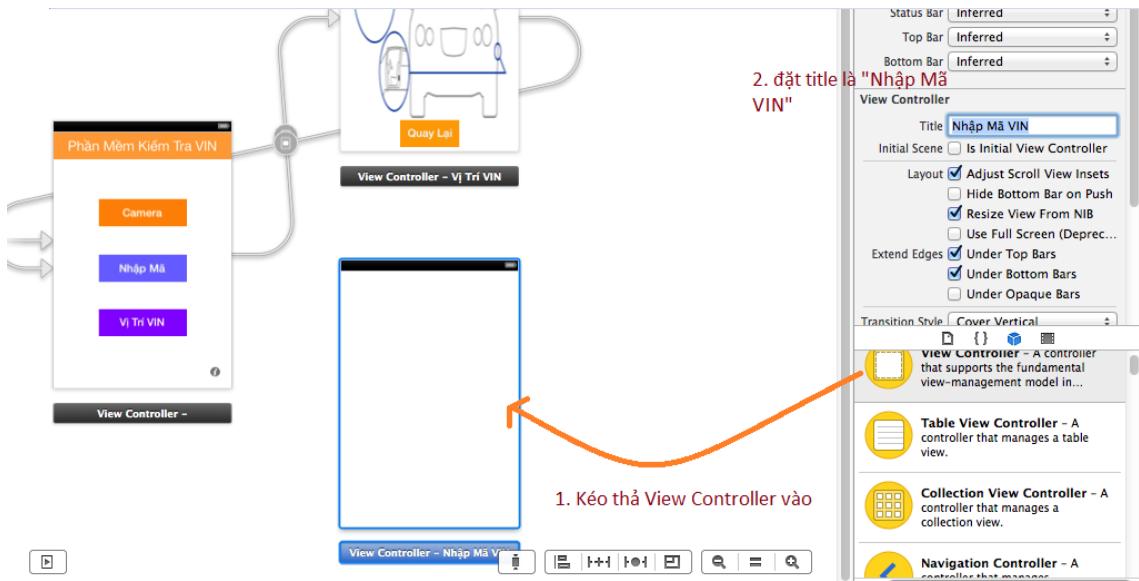
**Hình 6.27 Thiết kế giao diện**

→ Tạo kết nối chuyển về giao diện chính khi click vào button Quay Lại



**Hình 6.28 Tạo kết nối cho button Quay Lại**

**Bước 4:** Kéo thả một View Controller khác vào , đặt title là “Nhập Mã VIN”.



**Hình 6.29 Thêm View Controller mới tên là Nhập Mã VIN**

→ Thiết kế giao diện cho View Controller – Nhập Mã VIN.

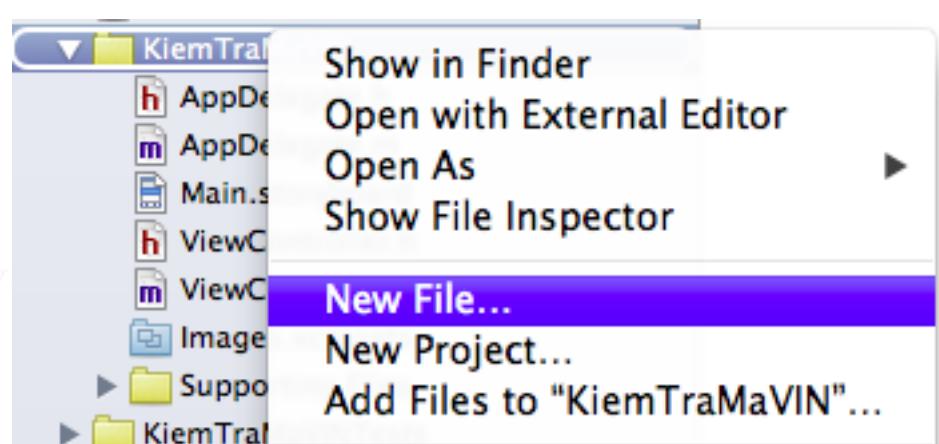


**Hình 6.30 Thiết kế giao diện**

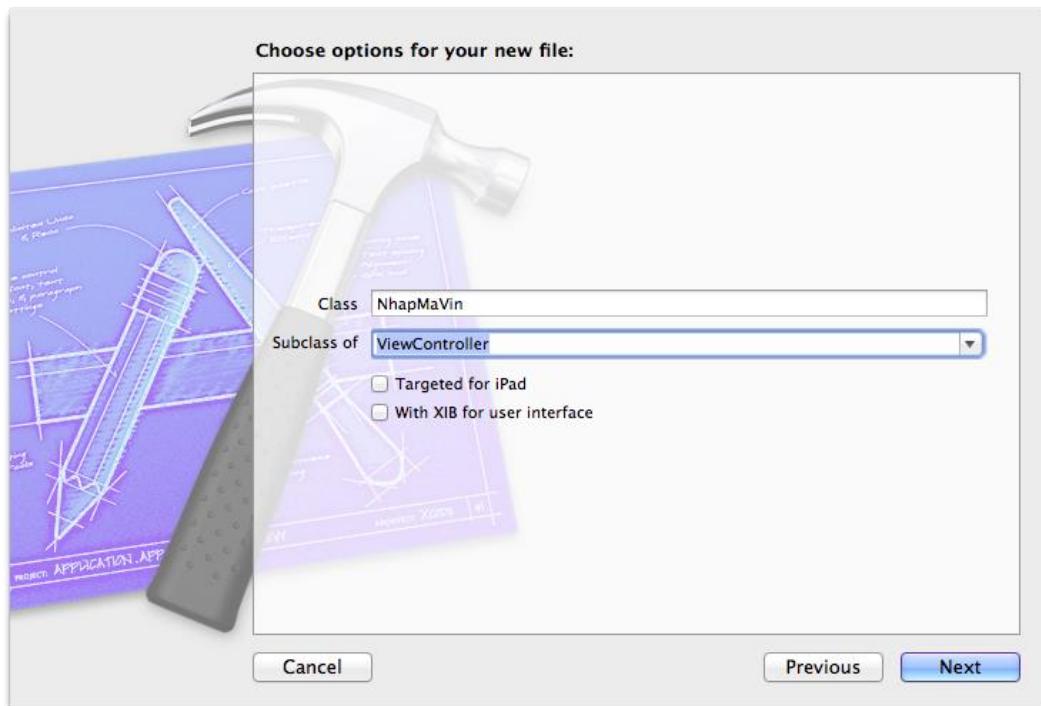
Trong đó

- [1] Kéo một label vào và nhập nội dung như hình.
- [2] Kéo một text field để người dùng nhập mã VIN vào
- [3] Kéo button vào sao cho khi click vào button Xem sẽ tiến hành kiểm tra mã VIN được nhập vào và đưa đến View Controller Kết Quả. Nếu click vào Camera sẽ chuyển đến view Camera, click vào Quay lại sẽ về giao diện chính.

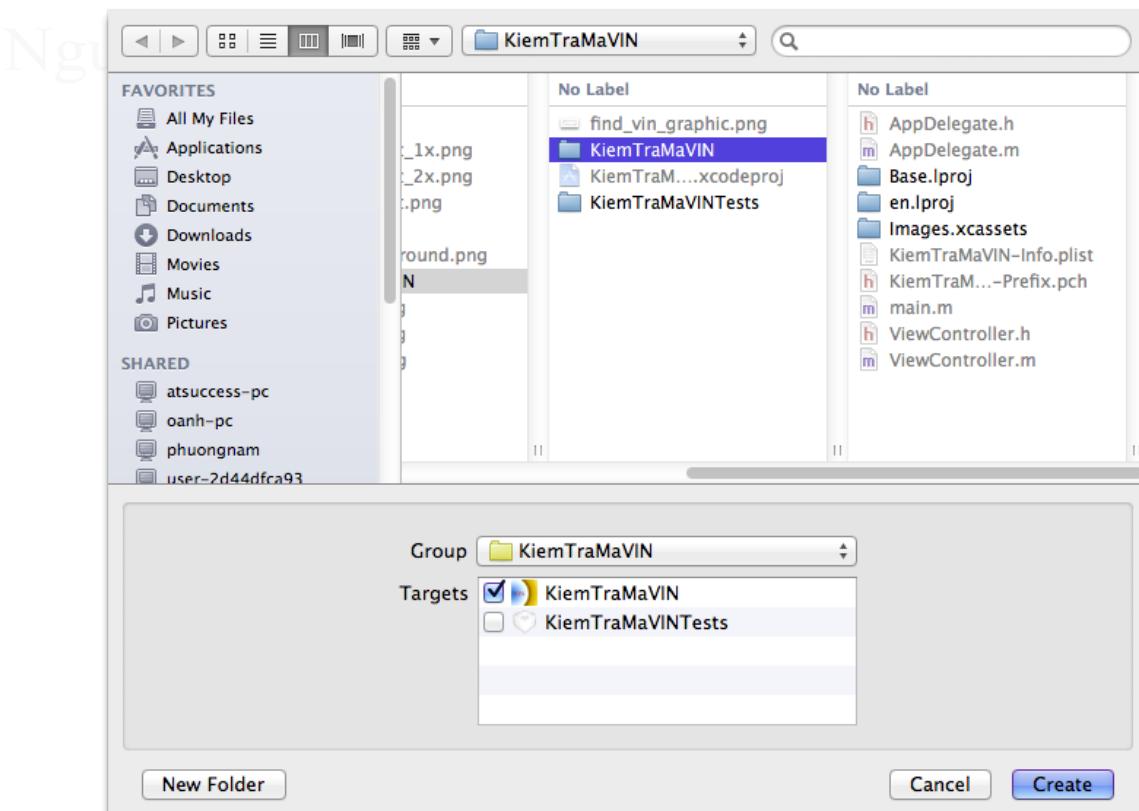
→Thêm một class mới để quản lý Nhập Mã Vin



Hình 6.31 New File



Hình 6.32 Thêm một class mới



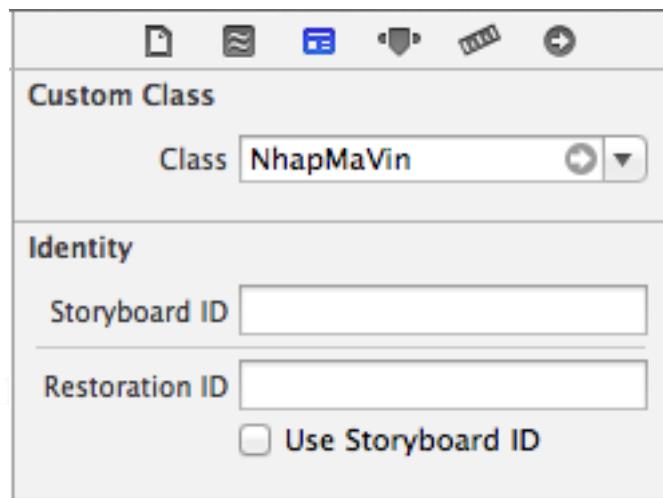
Hình 6.33 Chọn Create

→ Kết quả sau khi tạo



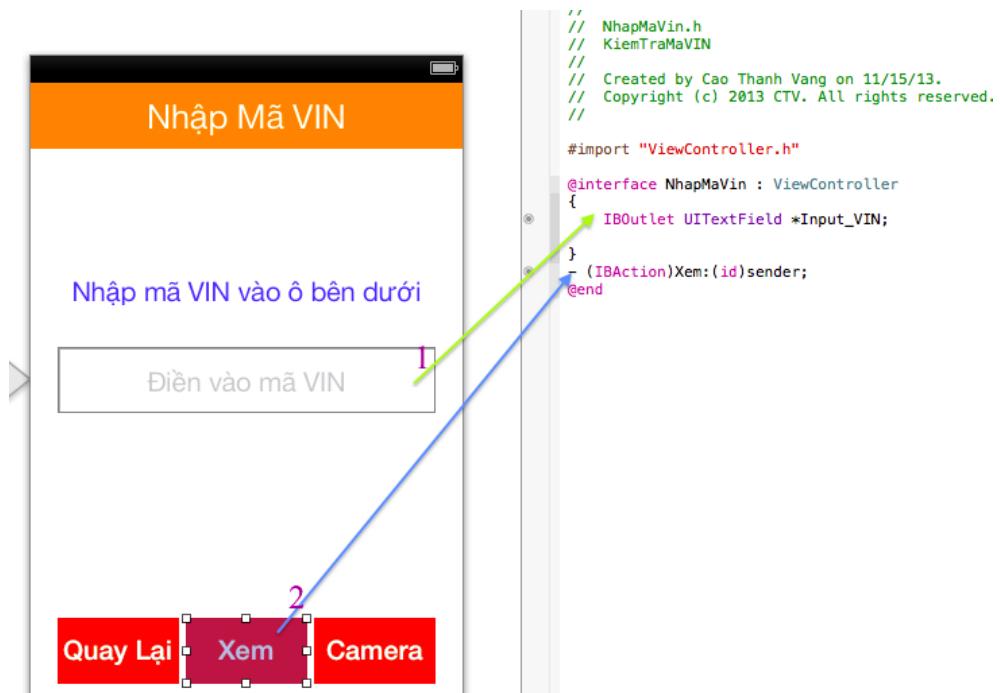
Hình 6.33 Kết quả

→ Chọn View Controller – Nhập Mã VIN, bên khung Inspector pane, chọn tab Identity Inspector.



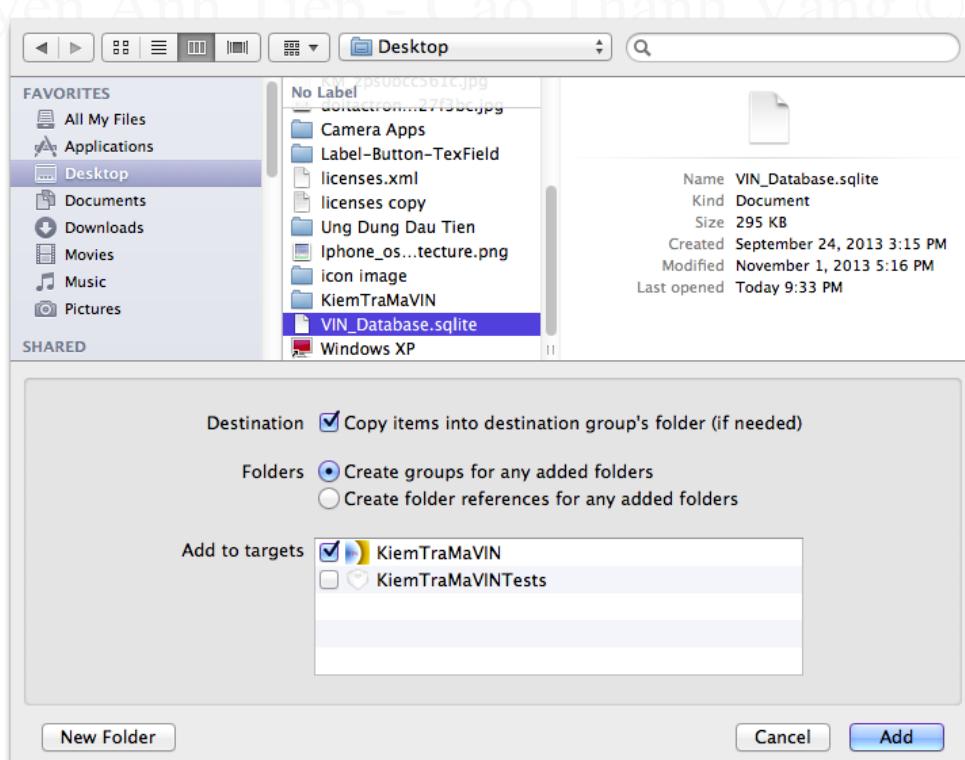
Hình 6.34 Gán class vào View Controller

→ Ánh xạ button Xem vào Textfield vào.



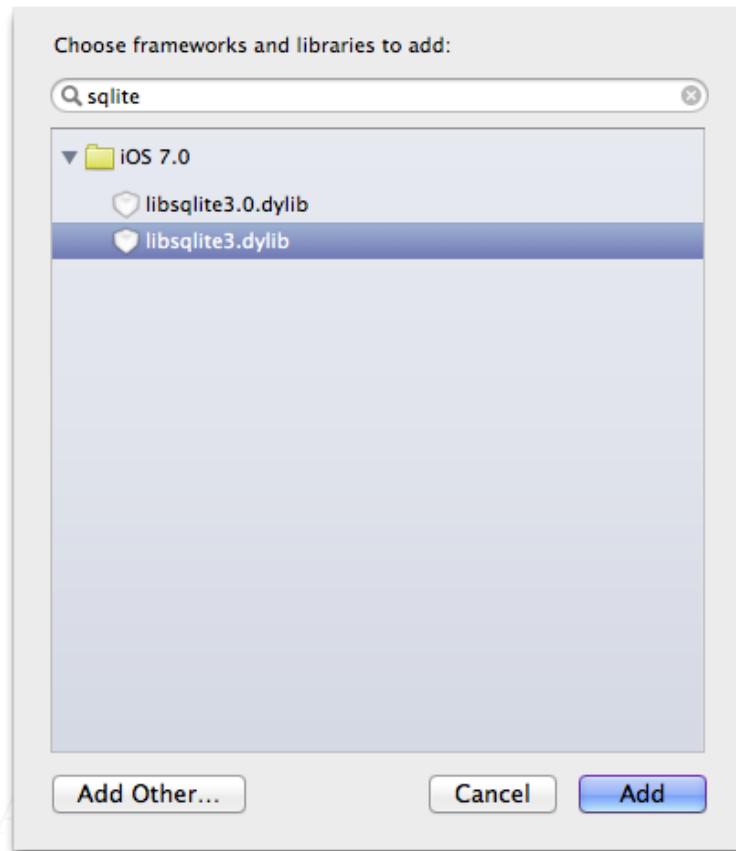
**Hình 6.35 Ánh xạ đối tượng**

→Thêm tập tin cơ sở dữ liệu VIN vào project.



**Hình 6.36 Thêm CSDL vào project**

→ Thêm thư viện sqlite vào project



Hình 6.37 Thêm thư viện hỗ trợ sqlite

Name	Status
libsqLite3.dylib	Required ▲
CoreGraphics.framework	Required ▲
UIKit.framework	Required ▲
Foundation.framework	Required ▲

Hình 6.38 Kết quả sau khi thêm thư viện

→ Thêm thư viện sqlite3.h vào tập tin .h và khai báo đối tượng **contactDB** dạng sqlite3, đối tượng **databasePath** dạng NSString để lưu giữ đường dẫn đến database.

```

#import "ViewController.h"
#import "sqlite3.h"

@interface NhapMaVin : ViewController
{
    IBOutlet UITextField *Input_VIN;

    sqlite3 *contactDB;
    NSString *databasePath;
}

- (IBAction)Xem:(id)sender;
@end

```

**Hình 6.39 Thêm thư viện và khai báo contactDB,databasePath**

→ Viết code cho hàm **viewDidLoad** để lấy đường dẫn đến tập tin sqlite. Bên trong hàm ViewDidLoad, bạn viết các đoạn code sau:

```

//khai bao doi tuong luu duong dan thu muc
NSString *docsDir;

//luu duong dan thu muc
docsDir = NSSearchPathForDirectoriesInDomains
    (NSDocumentDirectory, NSUserDomainMask, YES)[0];

//lay duong dan tap tin sqlite
databasePath = [[NSString alloc] initWithString:[docsDir
    stringByAppendingPathComponent:@"VIN_Database.sqlite"]]
];

```

**Hình 6.40 Khai báo các đối tượng**

```

//khai bao doi tuong quan ly tap tin de kiem tra tap tin
//sqlite co ton tai hay khong

NSFileManager *filemgr = [NSFileManager defaultManager];

if([filemgr fileExistsAtPath:databasePath]==NO)
{
    NSString *databasePathFromApp = [[[NSBundle mainBundle
        ] resourcePath]
        stringByAppendingPathComponent:@"VIN_Database.sqlite"];
    [filemgr copyItemAtPath:databasePathFromApp toPath:
        databasePath error:nil];
}

```

**Hình 6.41 Khai báo đối tượng kiểm tra tập tin**

→ Viết hàm để bàn phím ẩn đi khi người dùng nhấn vào Return. Trước hết thừa kế lại **UITextFieldDelegate** cho view Controller trong “**NhapMaVIN.h**”.

```
#import "ViewController.h"
#import "sqlite3.h"

@interface NhapMaVin : ViewController<UITextFieldDelegate>
{
    IBOutlet UITextField *Input_VIN;
    sqlite3 *contactDB;
    NSString *databasePath;
}
```

**Hình 6.42 Ké thừa UITextFieldDelegate**

→ Khai báo thêm @property cho Text field

```
@property (strong,nonatomic)UITextField *Input_VIN;
```

**Hình 6.43 Khai báo property cho Textfield**

→ Viết code ẩn bàn phím trong “**NhapMaVIN.m**”.

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField
{
    [textField resignFirstResponder];
    return YES;
}
```

**Hình 6.44 Viết code ẩn bàn phím**

→ Trong ViewDidLoad viết thêm

```
- (void)viewDidLoad
{
    self.Input_VIN.delegate = self;
    [super viewDidLoad];
    ...
}
```

**Hình 6.45 Viết thêm cho ViewDidLoad**

→ Trong NhapMaVIN.h, ngoài phương thức Xem do button ánh xạ, bạn khai báo thêm hai phương thức để kiểm tra độ dài mã VIN và phương thức kiểm tra vị trí thứ 9 của mã VIN.

```
- (IBAction)Xem:(id)sender;
- (BOOL)KiemTraDoDai: (NSString*)
    inputstring;
- (BOOL)KiemTraDigit: (char) inputpoint9;
```

Hình 6.46 Khai báo thêm hai phương thức

→ Viết code cho hai phương thức mới khai báo

```
- (BOOL)KiemTraDoDai: (NSString*) inputstring
{
    if (inputstring.length==17) {
        return TRUE;
    }
    else
        return FALSE;
}
```

Hình 6.47 Phương thức KiemTraDoDai

```
- (BOOL)KiemTraDigit:(char)inputpoint9
{
    if (inputpoint9=='X') {

        return TRUE;
    }
    else
    {   if (inputpoint9>='0' && inputpoint9<='9') {
        return TRUE;
    }
    else
        return FALSE;
}
}
```

Hình 6.48 Phương thức KiemTraDigit

→ Khai báo biến **dppath** lưu trữ đường dẫn đến tập tin sqlite đã lấy được từ hàm viewDidLoad

```
#import "NhapMaVin.h"

@interface NhapMaVin ()

@end

const char *dbpath;
|
@implementation NhapMaVin
```

**Hình 6.49** Khai báo biến lưu đường dẫn đến tập tin sqlite từ hàm viewDidLoad

→ Khai báo biến kiểu **sqlite3\_stmt** để chứa dữ liệu trả về từ câu lệnh truy vấn.

```
#import "NhapMaVin.h"

@interface NhapMaVin ()

@end

const char *dbpath;
sqlite3_stmt *statement;
|
```

**Hình 6.50** Khai báo biến kiểu sqlite3\_stmt để lưu kết quả truy vấn

→ Khai báo biến để lưu các giá trị được trích xuất từ dữ liệu trả về.

```
@property (strong, nonatomic) NSString *  
    manufacturer;  
@property (strong, nonatomic) NSString *  
    make;  
@property (strong, nonatomic) NSString *  
    type;  
@property (strong, nonatomic) NSString *  
    year;  
@property (strong, nonatomic) NSString *  
    factory;  
@property (strong, nonatomic) NSString *  
    fuel;
```

Hình 6.51 Khai báo trong tập tin NhapMaVIN.h

```
@implementation NhapMaVin  
  
@synthesize manufacturer;  
@synthesize make;  
@synthesize type;  
@synthesize year;  
@synthesize factory;  
@synthesize fuel;
```

Nguyễn A © 2013

Hình 6.52 Khai báo trong tập tin NhapMaVIN.m

→ Viết code cho phương thức Xem của button như sau

→ Gán giá trị cho biến **dbpath** lưu trữ đường dẫn sqlite đã chuẩn hóa UTF-8 và biến **vin\_upcase** lưu giá trị VIN sau khi đã chuyển sang dạng viết Hoa.

```
//luu lai duong dan database  
dbpath = [databasePath UTF8String];  
  
//khai bao luu tru vincode  
NSString *vin_upcase= [[NSString alloc] init];
```

Hình 6.53 Khai báo biến

→ Kiểm tra mã VIN có NULL hay không, nếu có thì thông báo mã rỗng, nếu không thì triển khai tiếp code trong hàm else.

```

//kiem tra ma vin null
if (Input_VIN.text==NULL) {
    UIAlertView *notice = [[UIAlertView alloc]
        initWithTitle:@"Thông Báo" message:@"Mã VIN rỗng!"
        delegate:self cancelButtonTitle:@"Đồng ý"
        otherButtonTitles:nil, nil];
    [notice show];
}
else
{

```

### Hình 6.54 Nếu mã VIN là NULL thì xuất thông báo

→ Trong hàm else (nếu mã VIN không NULL sẽ thực hiện code trong hàm else) gán giá trị cho vin\_upcase là mã VIN đã viết hoa toàn bộ, sau đó viết tiếp các đoạn code tiếp theo.

```

//chuan hoa thanh chu in hoa
vin_upcase = [Input_VIN.text uppercaseString];

```

### Hình 6.55 Viết hoa mã VIN và gán cho vin\_upcase

→ Tạo biến lưu kết quả trả về từ hàm kiểm tra độ dài VIN và gọi hàm kiểm tra xem độ dài VIN có phù hợp hay không.

```

//kiem tra do dai chuoi
BOOL valuelenght=FALSE;
valuelenght = [self KiemTraDoDai:vin_upcase];

```

### Hình 6.56 Lưu kết quả kiểm tra độ dài

→ Tạo biến lưu ký tự vị trí thứ 9 của mã VIN, kiểm tra xem ký tự có hợp lệ hay không.

```

//kiem tra vi tri thu 9
char point9;
point9 = [vin_upcase characterAtIndex:8];
BOOL valuedigit=FALSE;
valuedigit = [self KiemTraDigit:point9];

```

### Hình 6.57 Lưu kết quả kiểm tra vị trí thứ 9

→ Bắt đầu đi sâu vào viết hàm **if** để xem kiểm tra độ dài và ký tự vị trí 9 có hợp lệ hay không → Nếu không thì xuất thông báo, nếu hợp lệ tiến hành thực thi trong hàm **else**.

```
//begin code
if (valuedigit==FALSE || valuelenght==FALSE) {
    UIAlertView *notice = [[UIAlertView alloc] initWithTitle:@"Thông
        Báo" message:@"Mã VIN sai!" delegate:self
        cancelButtonTitle:@"Đồng ý" otherButtonTitles:nil, nil];
    [notice show];
}
else
{
    ...
}
```

Hình 6.58 Viết code cho điều kiện If

→ Tiến hành viết code cho hàm else để xử lý nếu mã VIN thoả mãn yêu cầu  
→ Tạo câu lệnh truy vấn cho mã vin vị trí 1-3, 8,10,11

```
//kiem tra
//check point 1-3
NSMutableString *query13 = [NSMutableString
    stringWithString:@"SELECT * FROM Pos_123 WHERE
    vincode=''";
NSString *vincode13 = [[NSString alloc]init];
vincode13 = [vin_upcase substringWithRange:
    NSMakeRange(0, 3)];
[query13 appendString:vincode13];
[query13 appendString:@""];

```

Hình 6.59 Tạo câu lệnh truy vấn vị trí 1-3

```
//check point 8
NSMutableString *query8 = [NSMutableString
    stringWithString:@"SELECT * FROM Pos_8 WHERE
    vincode=''";
NSString *vincode8 = [[NSString alloc]init];
vincode8 = [vin_upcase substringWithRange:
    NSMakeRange(7, 1)];
[query8 appendString:vincode8];
[query8 appendString:@""];
```

Hình 6.60 Tạo câu lệnh truy vấn vị trí 8

```

//check point 10
NSMutableString *query10 = [NSMutableString
    stringWithString:@"SELECT * FROM Pos_10 WHERE
    vincode=''];
NSString *vincode10 = [[NSString alloc]init];
vincode10 = [vin_upcase substringWithRange:
    NSMakeRange(9, 1)];
[query10 appendString:vincode10];
[query10 appendString:@""];

```

**Hình 6.61 Tạo câu lệnh truy vấn vị trí 10**

```

//check point 11
NSMutableString *query11 = [NSMutableString
    stringWithString:@"SELECT * FROM Pos_11 WHERE
    vincode=''];
NSString *vincode11 = [[NSString alloc]init];
vincode11 = [vin_upcase substringWithRange:
    NSMakeRange(10, 1)];
[query11 appendString:vincode11];
[query11 appendString:@""];

```

Nguyễn Văn Hùng – 2013

**Hình 6.62 Tạo câu lệnh truy vấn vị trí 11**

→ Khai báo biến để chứa giá trị thu được sau khi mã hoá các lệnh truy vấn về dạng hỗ trợ UTF-8.

```

const char *query_stmt13 = [query13 UTF8String];
const char *query_stmt8 = [query8 UTF8String];
const char *query_stmt10 = [query10 UTF8String];
const char *query_stmt11 = [query11 UTF8String];

```

**Hình 6.63 Tạo biến lưu giá trị đã mã hóa**

→ Viết code truy vấn dữ liệu với các lệnh truy vấn tương ứng. Trước tiên kiểm tra điều kiện kết nối dữ liệu, nếu kết nối được thì triển khai truy vấn dữ liệu trong hàm **if**, ngược lại báo không kết nối được.

```
if(sqlite3_open(dbpath, &contactDB)==SQLITE_OK)
{
```

### Hình 6.64 Kiểm tra mở kết nối với sqlite

→ Viết code truy vấn dữ liệu cho mã vin 1-3

```
//-----
if (sqlite3_prepare_v2(contactDB,
                        query_stmt13, -1, &
                        statement, nil) ==
    SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        manufactor = [[NSString alloc]
                      initWithUTF8String:(const char *)
                      sqlite3_column_text(statement, 1)];
        make = [[NSString alloc]
                 initWithUTF8String:(const char *)
                 sqlite3_column_text(statement, 2)];
        type = [[NSString alloc]
                 initWithUTF8String:(const char *)
                 sqlite3_column_text(statement, 3)];
        NSLog(@"%@", @"%@\n%@ \n%@", manufactor, make,
               type);
    }
    else
    {
        NSLog(@"No data");
    }
    sqlite3_finalize(statement);
}
```

### Hình 6.65 Code truy vấn vị trí 1-3

→ Code truy vấn dữ liệu cho mã vin 8

```

//-----
if(sqlite3_prepare_v2(contactDB,
                      query_stmt8, -1, &
                      statement, nil) ==
   SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        fuel = [[NSString alloc]
                 initWithUTF8String:(const char *)
                 sqlite3_column_text(statement, 1)];

        NSLog(@"%@",fuel);
    }
    else
    {
        NSLog(@"No data");
    }
    sqlite3_finalize(statement);
}

```

**Hình 6.66** Code truy vấn vị trí 8

→Code truy vấn dữ liệu cho mã vin vị trí 10

Nguyễn Anh Tiễn – Cao Thành Vàng ⓒ 2013

```

//-----
if (sqlite3_prepare_v2(contactDB,
                      query_stmt10, -1, &
                      statement, nil) ==
   SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        year = [[NSString alloc]
                  initWithUTF8String:(const char *)
                  sqlite3_column_text(statement, 1)];
        NSLog(@"%@",year);

    }
    else
    {
        NSLog(@"No data");
    }
    sqlite3_finalize(statement);
}

```

**Hình 6.67** Code truy vấn cho vị trí 10

→Viết code truy vấn cho mã vin vị trí 11

```

//-----
if(sqlite3_prepare_v2(contactDB,
                      query_stmt11, -1, &
                      statement, nil) ==
   SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        factory = [[NSString alloc]
                    initWithUTF8String:(const char *)
                    sqlite3_column_text(statement, 1)];
        NSLog(@"%@",factory);
    }
    else
    {
        NSLog(@"No data");
    }
    sqlite3_finalize(statement);
}

```

### Hình 6.68 Code truy vấn cho vị trí 11

→ Sau khi viết xong code truy vấn dữ liệu cho mã VIN, tiến hành đóng kết nối truy xuất dữ liệu sử dụng sqlite3\_close. Đồng thời viết code cho trường hợp else của câu lệnh if dùng mở kết nối với sqlite.

```

        sqlite3_close(contactDB);

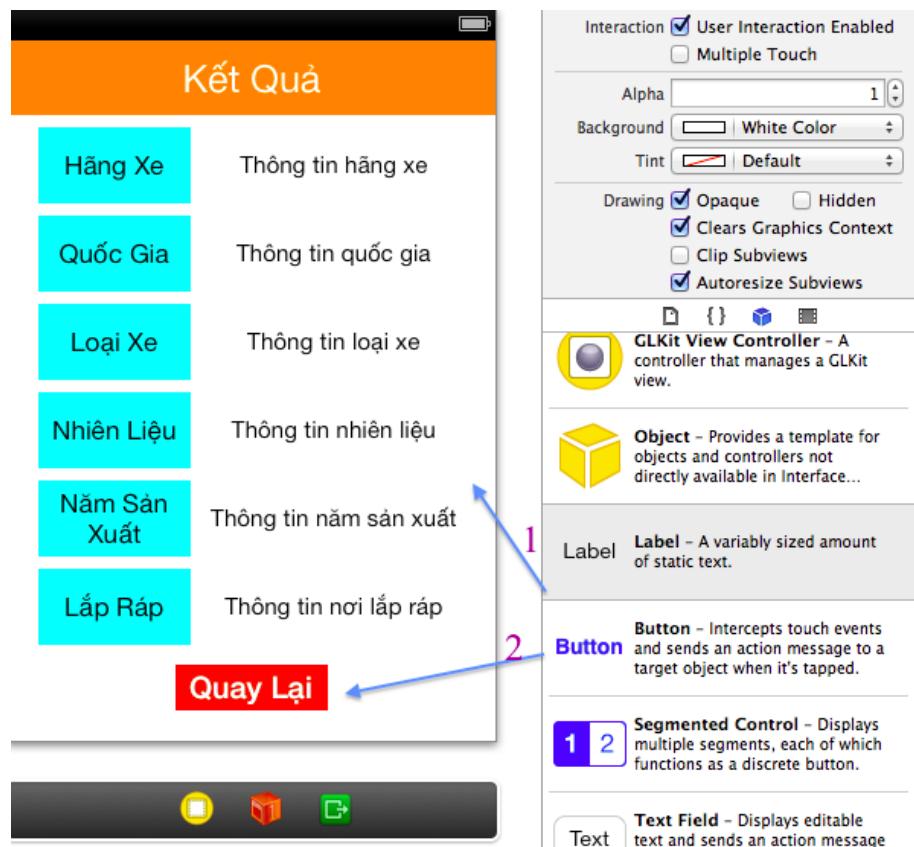
    }
else
{
    NSLog(@"Can't Open Database");
}

}

```

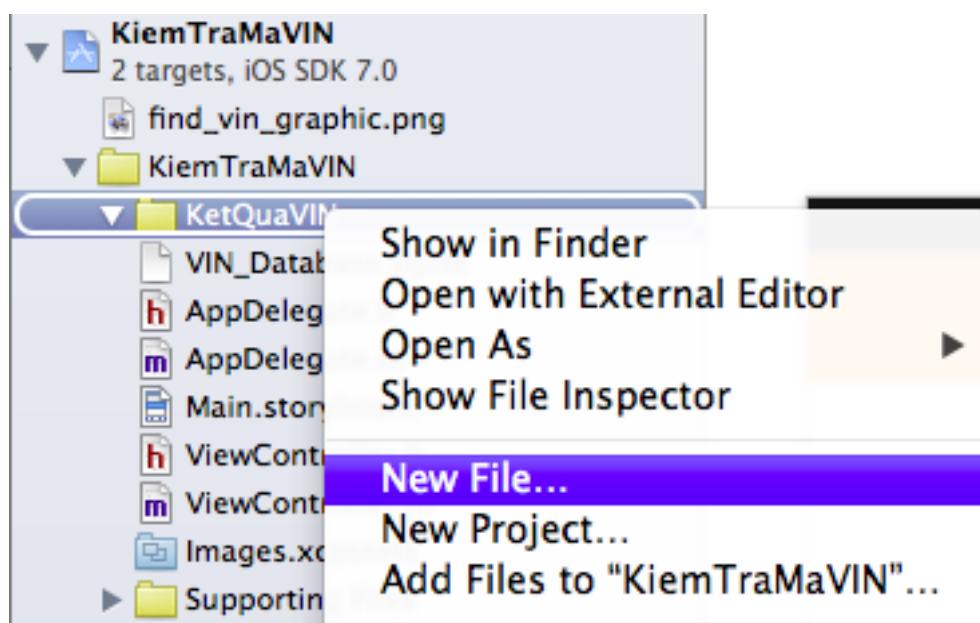
### Hình 6.69 Đóng kết nối sqlite và viết lệnh else

**Bước 5:** Thêm một View Controller mới vào ứng dụng, đặt tên là **KQVIN**. Thiết kế theo giao diện với một button là **Quay Lại**, còn lại là các label để hiển thị.

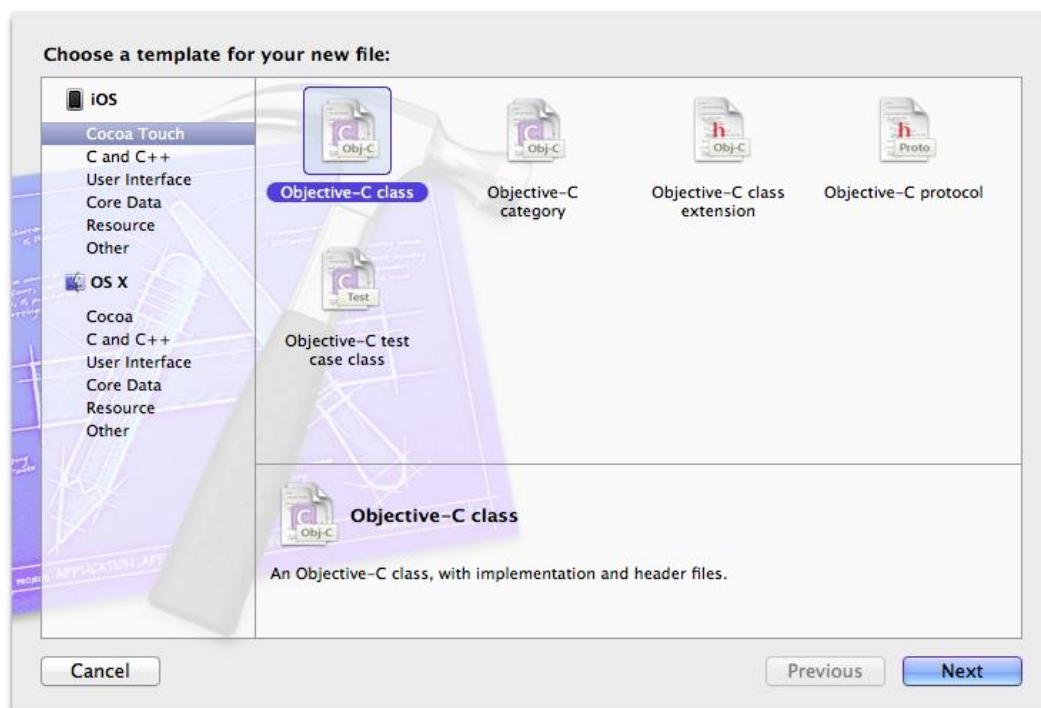


Nguyễn Anh Tài - Code Together © 2013

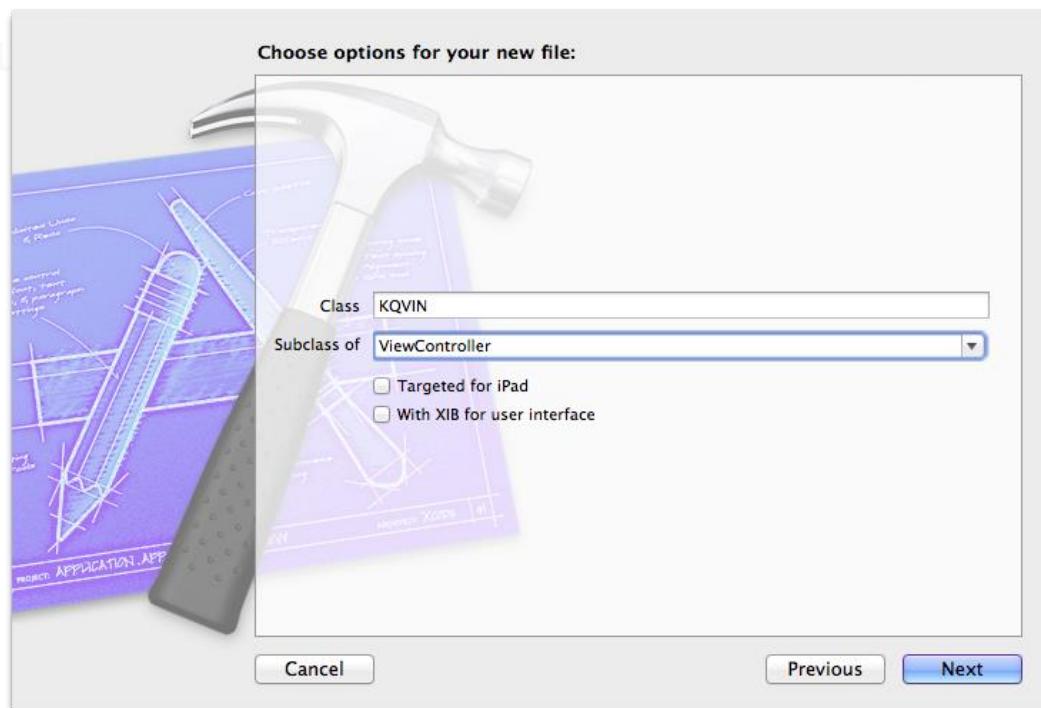
→ Thêm một class là **KetQuaVIN** dạng **ViewController**.



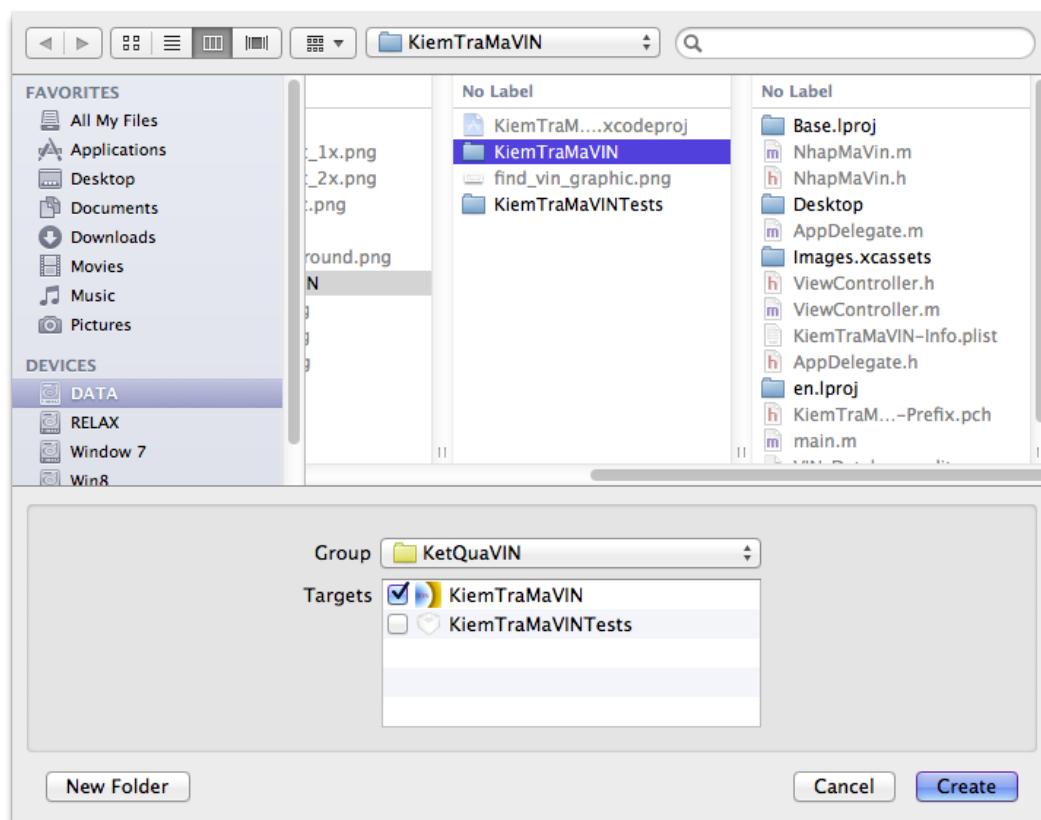
Hình 6.71 New File



Hình 6.72 Tạo class mới

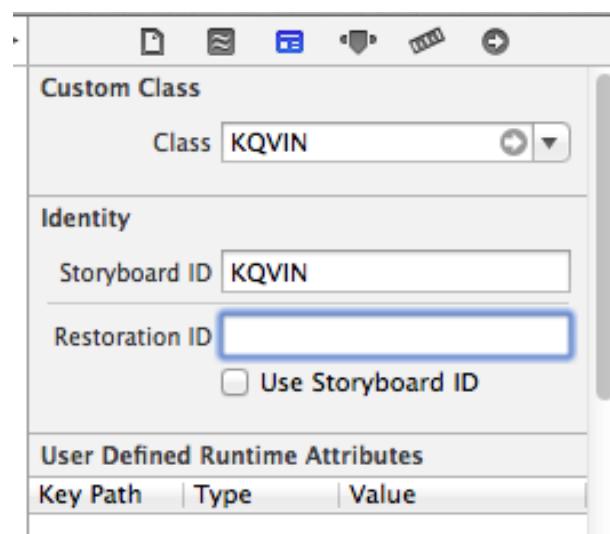


Hình 6.73 Class KQVIN



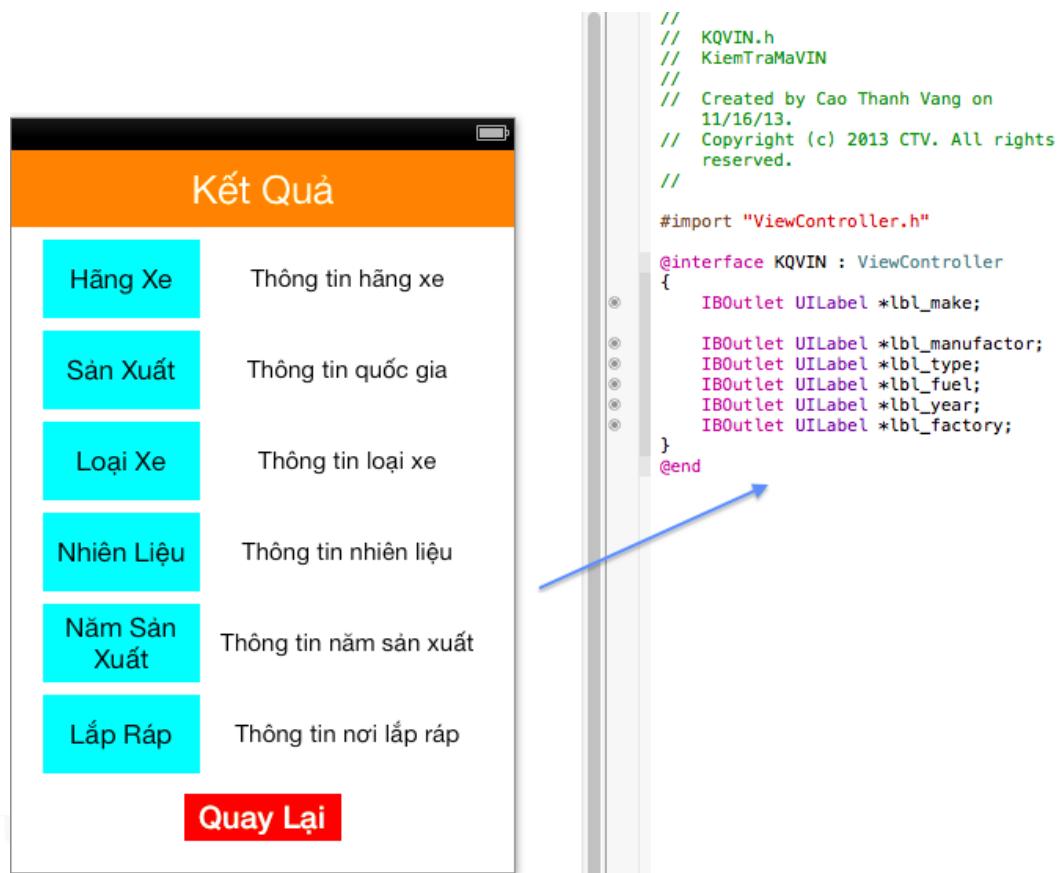
Nguyễn Anh Tiệp **Hình 6.74 Create** © 2013

→Đặt lớp mới thêm vào làm lớp quản lý cho View Controller KQVIN. Đồng thời đặt **Identity → Storyboard ID** là tên lớp quản lý.



**Hình 6.75 Gán class vào Identity**

→ Ánh xạ các đối tượng vào KQVIN



Hình 6.76 Ánh xạ đối tượng

→ Khai báo biến để lưu thông tin từ Nhập Mã VIN chuyển qua.

```
@property (strong, nonatomic) NSString *  
    manufacturer;  
@property (strong, nonatomic) NSString *  
    make;  
@property (strong, nonatomic) NSString *  
    type;  
@property (strong, nonatomic) NSString *  
    year;  
@property (strong, nonatomic) NSString *  
    factory;  
@property (strong, nonatomic) NSString *  
    fuel;|
```

Hình 6.77 Khai báo biến để lưu dữ liệu

→ Viết code để hiển thị kết quả ra giao diện KQVIN

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    lbl_make.text = _make;
    lbl_manufactor.text = _manufactor;
    lbl_type.text = _type;
    lbl_fuel.text = _fuel;
    lbl_year.text = _year;
    lbl_factory.text = _factory;

    // Do any additional setup after
    // loading the view.
}
```

**Hình 6.78** Viết code hiển thị dữ liệu ra màn hình

→ import class NhậpMãVIN vào KQVIN

```
//
#import "KQVIN.h"
#import "NhậpMãVIN.h"

@interface KQVIN ()
```

**Hình 6.79** Import class NhậpMãVIN

→ Trở lại NhậpMãVIN.m, import lớp quản lý của KQVIN vào Nhập MÃ VIN Controller

```
#import "NhậpMãVIN.h"
#import "KQVIN.h"

@interface NhậpMãVIN ()
```

**Hình 6.80** Import class KQVIN

→ Viết code để chuyển dữ liệu từ NhậpMãVIN sang KQVIN

```

//transfer data
KQVIN *kq = [self.storyboard
    instantiateViewControllerWithIdentifier:@"KQVIN"];

kq.manufactor = manufactor;
kq.make = make;
kq.type = type;
kq.fuel = fuel;
kq.year = year;
kq.factory = factory;

[self presentViewController:kq animated:YES completion:
nil];

```

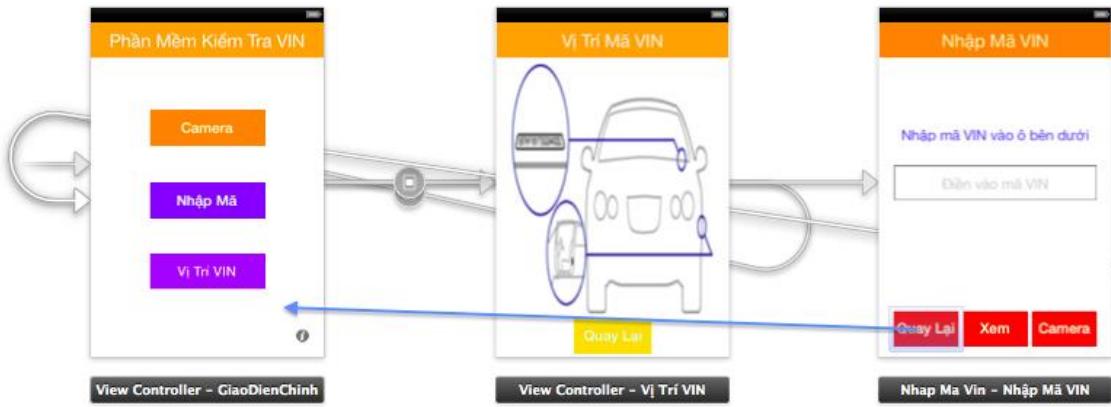
**Hình 6.81** Viết code chuyển dữ liệu từ Nhập Mã VIN sang KQVIN

→ Tạo kết nối cho button Quay Lại của KQVIN về Nhập Mã VIN



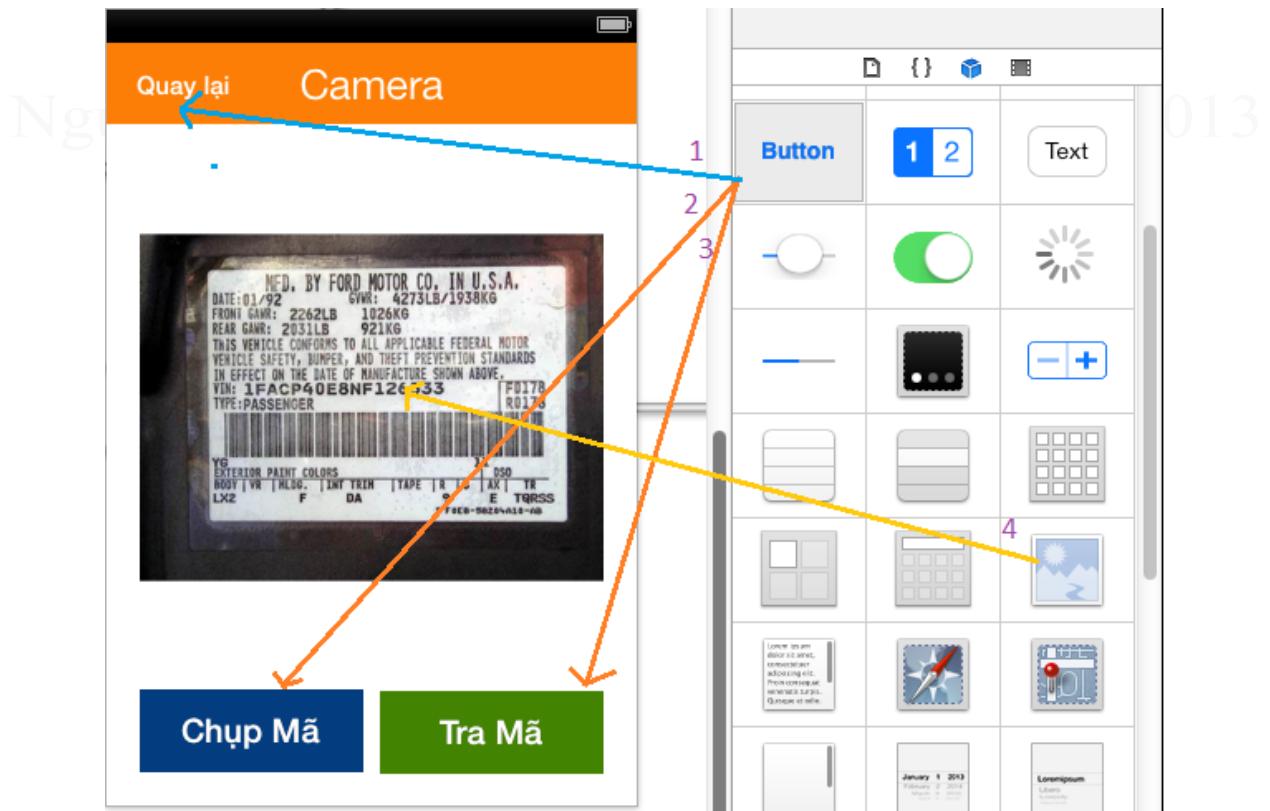
**Hình 6.82** Tạo kết nối cho button Quay Lại của KQVIN

→ Tạo kết nối cho button Quay Lại của Nhập Mã VIN về giao diện chính.



**Hình 6.83 Tạo kết nối cho button Quay Lại của Nhập Mã VIN**

**Bước 6:** Thêm một view controller mới tên là **Camera**. Thiết kế như giao diện với một button để truy cập camera, một button để đưa ảnh lên server xử lý OCR.

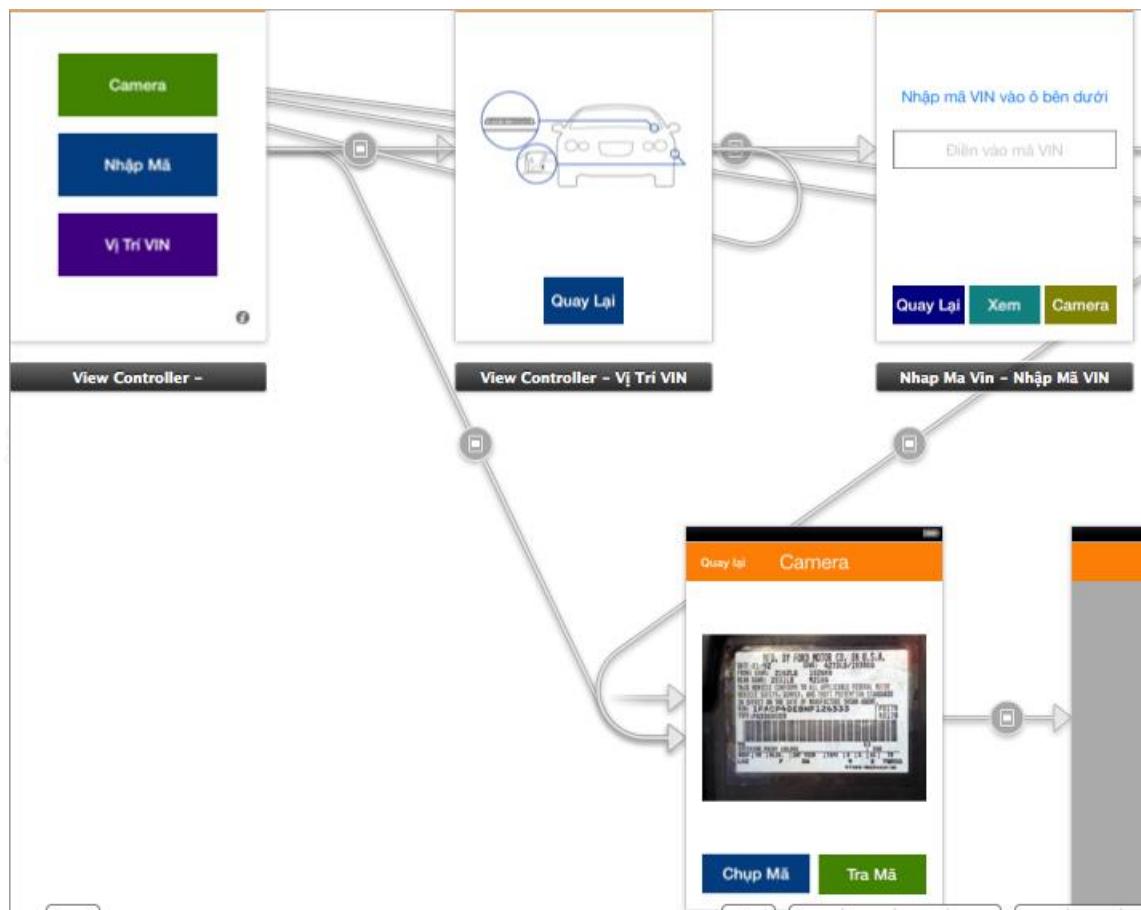


Hình 6.84 Thiết kế giao diện

Trong đó

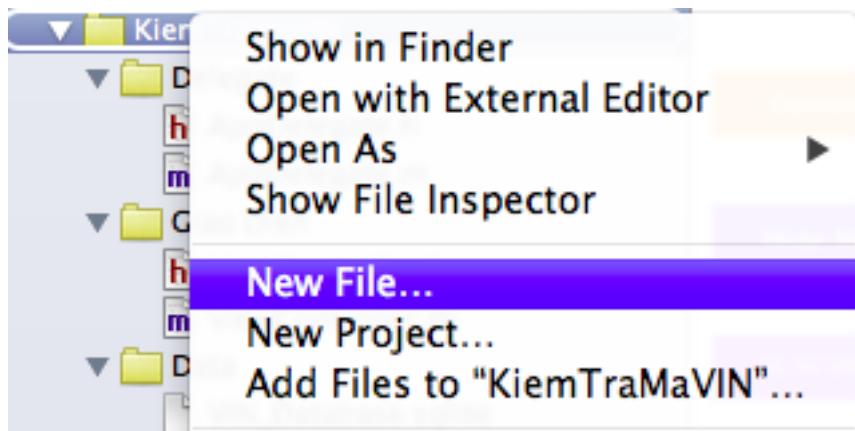
- [1] Button **Quay Lại** dùng để trở lại giao diện chính.
- [2] Button **Chụp Mã** dùng để truy cập vào camera của iPhone.
- [3] Button **Tra Mã** để upload hình ảnh chụp được lên server.
- [4] **UIImageView** để hiển thị hình ảnh. Với hình ảnh mặc định là **sample.png** (hình ảnh có trong folder source).

→ Kết nối button **Camera** trên Nhập Mã VIN, button **Camera** trên giao diện chính với **view controller Camera**.

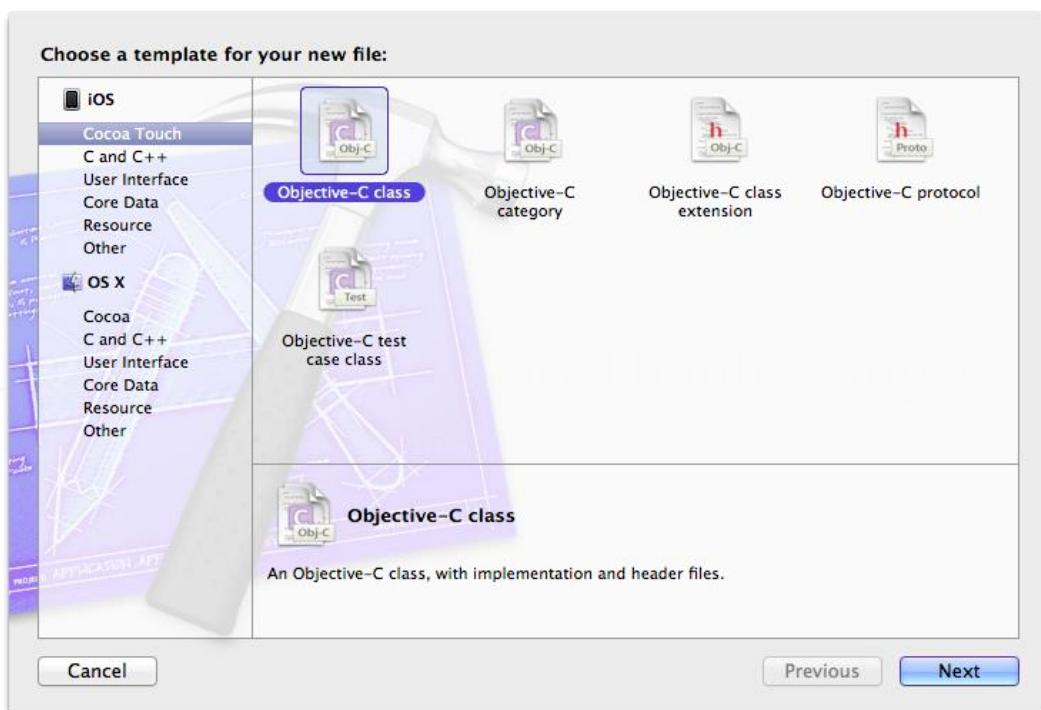


**Hình 6.85** Kết nối các button Camera với View Controller Camera

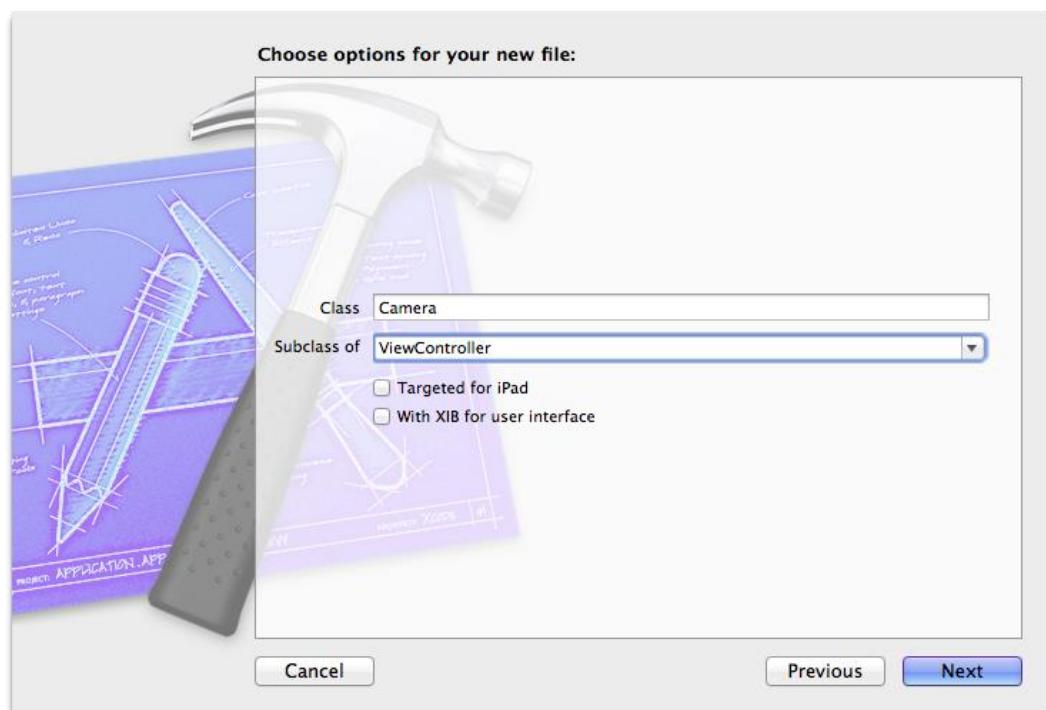
→ Thêm class mới vào project để quản lý ViewController Camera.



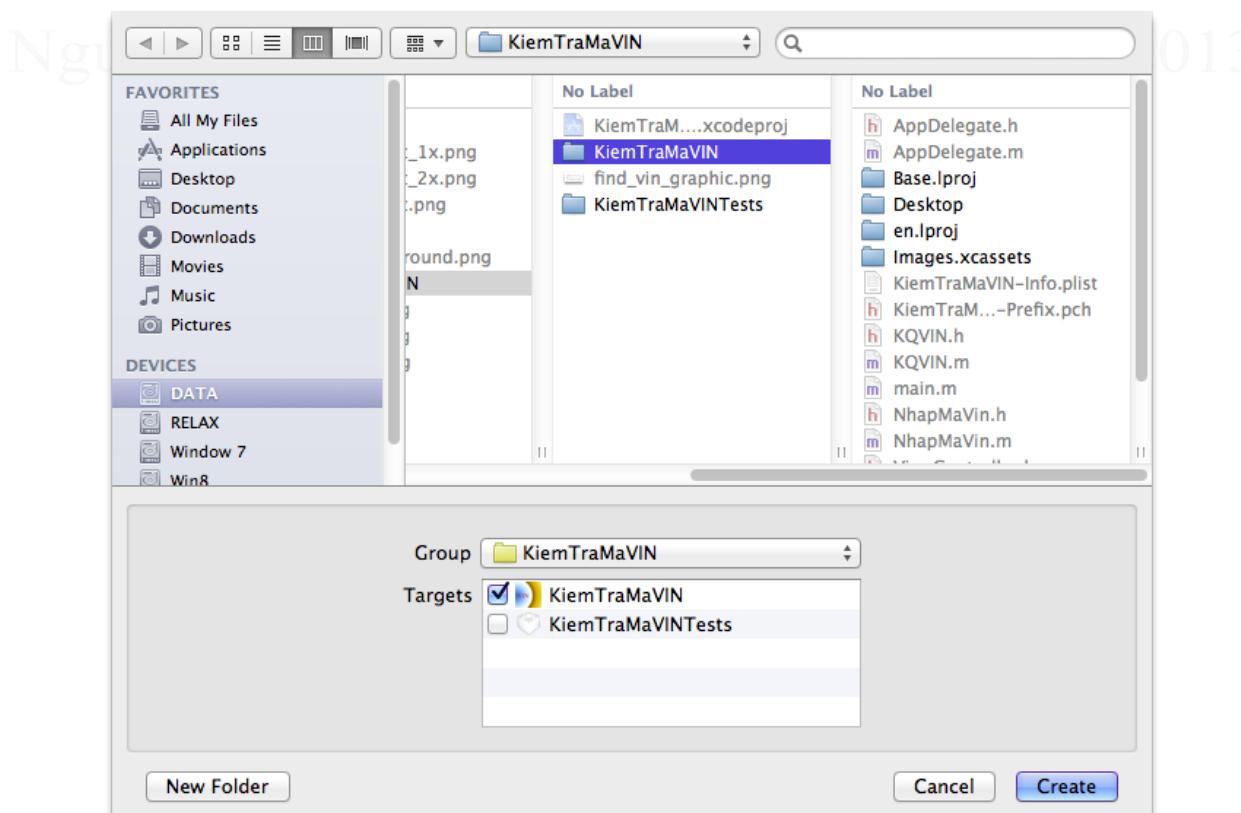
Hình 6.86 New File



Hình 6.87 Tạo class mới

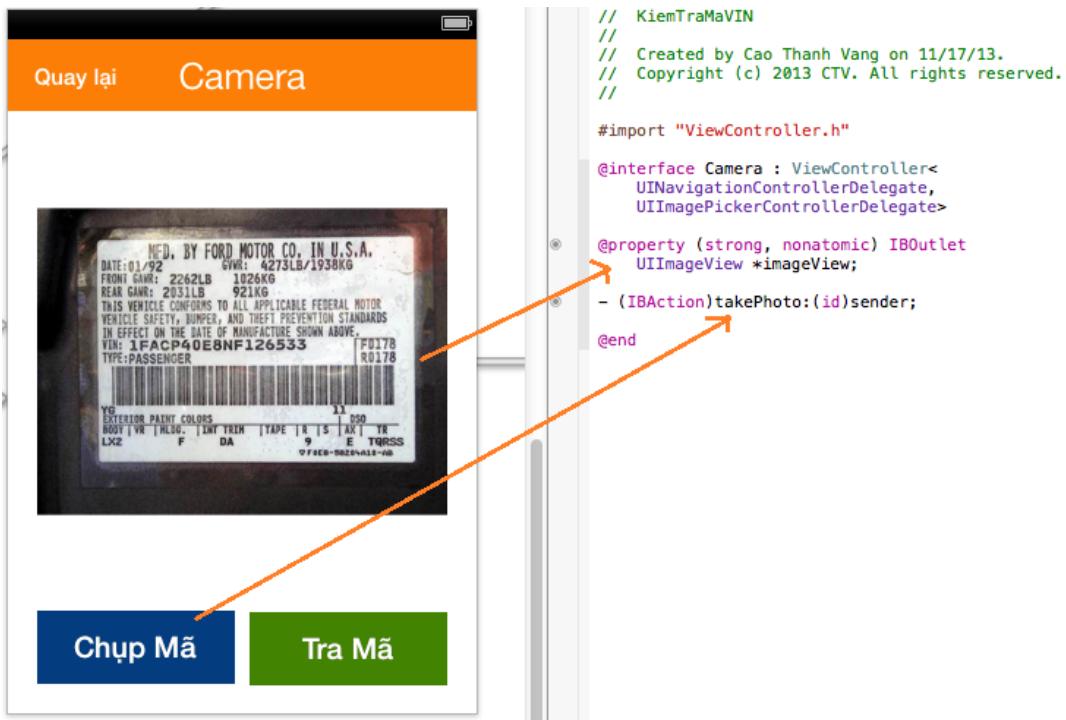


Hình 6.88 Class Camera



Hình 6.89 Create

→ Ánh xạ các đối tượng vào tập tin Camera.h



Hình 6.90 Ánh xạ đối tượng

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

→ Trong tập tin AppDelegate.h khai báo đối tượng UIImage để lưu ảnh chụp được từ camera.

```
//
// AppDelegate.h
// KiemTraMaVIN
//
// Created by CTV on 11/3/13.
// Copyright (c) 2013 CTV. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <
    UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) UIImage*
    imageToProcess;
|
@end
```

Hình 6.91 Khai báo UIImage

→ Trong tập tin **AppDelegate.m** viết code sau để các View khác có thể truy xuất **imageToProcess** dễ dàng.

```
@implementation AppDelegate  
@synthesize imageToProcess;
```

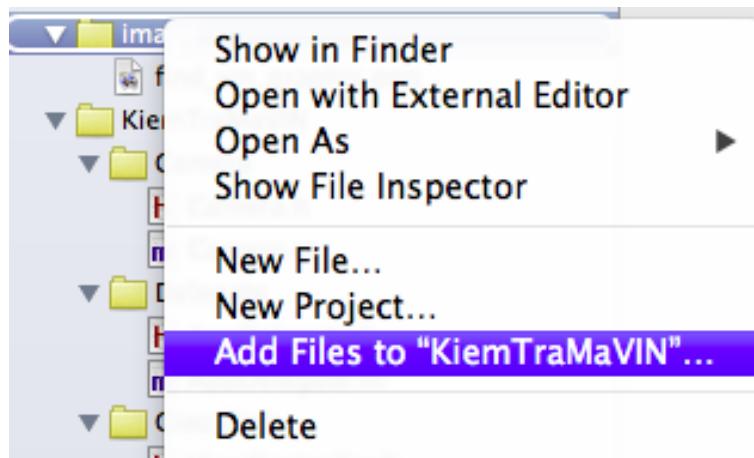
**Hình 6.92** Viết code Trong **AppDelegate.m**

→ Viết thêm vào hàm **didFinishLaunching** để gán hình ảnh mặc định cho **imageToProcess**.

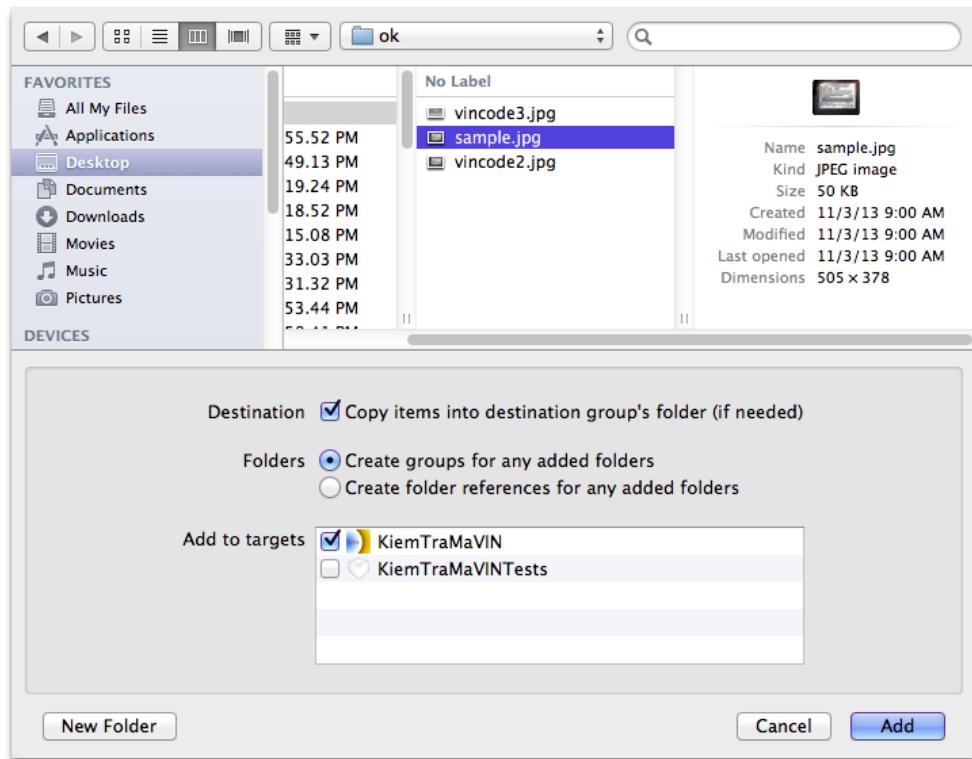
```
- (BOOL)application:(UIApplication *)application  
didFinishLaunchingWithOptions:(NSDictionary *)  
launchOptions  
{  
  
    imageToProcess = [UIImage  
        imageNamed:@"sample.jpg"];  
  
    // Override point for customization after  
    // application launch.  
    sleep(3);  
    return YES;  
}
```

**Hình 6.93** Viết code trong **didFinishLaunching**

→ Thêm tập tin ảnh **sample.jpg** vào project.



**Hình 6.94** Thêm tập tin



**Hình 6.95 Chọn tập tin sample.jpg và Add**

→ Trong tập tin **camera.h**, thừa kế lại **delegate** của **UIImagePickerController**.

```
#import "ViewController.h"

@interface Camera : ViewController<UINavigationControllerDelegate,UIImagePickerControllerDelegate>
```

**Hình 6.96 Thừa kế Delegate của UIImagePickerController**

→ Trong tập tin **camera.m**, import thư viện **AppDelegate.h** và viết code sau.

```
#import "Camera.h"
#import "AppDelegate.h"
@interface Camera ()
```

```
@end
```

```
@implementation Camera
```

```
@synthesize imageView;
```

**Hình 6.97 Import thư viện AppDelegate.h**

→ Trong hàm viewDidLoad, viết code để lấy ảnh được khai báo trong AppDelegate.h gán vào imageView.

```
- (void)viewDidLoad
{
    self.imageView.image = [(AppDelegate*)[[UIApplication sharedApplication] delegate] imageToProcess];
    [super viewDidLoad];
    // Do any additional setup after loading the view.
}
```

### Hình 6.98 Gán ảnh trong AppDelegate vào imageView

→ Viết thêm hàm viewDidUnload

```
-(void)viewDidUnload
{
    [self setImageView:nil];
    [super viewDidUnload];
}
```

### Hình 6.99 Viết thêm hàm viewDidLoad

→ Viết code chụp ảnh. Khai báo đối tượng UIImagePickerController là imagePicker, sourceType là UIImagePickerControllerSourceTypeCamera để chọn là sử dụng camera.

```
- (IBAction)takePhoto:(id)sender {
    UIImagePickerController *imagePicker = [[UIImagePickerController alloc] init];
    imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
    imagePicker.delegate = self;
    [self presentViewController:imagePicker animated:YES completion:NULL];
}
```

### Hình 6.100 Viết code chụp ảnh

→ Viết code để lấy ảnh chụp được vào imageView, đồng thời lưu vào biến của AppDelegate.

```

- (void) imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    UIImage *image = [info objectForKey:@"UIImagePickerControllerOriginalImage"];

    [picker dismissViewControllerAnimated:YES completion:NULL];
    self.imageView.image = image;
    [(AppDelegate*)[[UIApplication sharedApplication] delegate] setImageToProcess:image];
}

```

**Hình 6.101** Viết code gán hình ảnh từ camera vào imageView và AppDelegate

→ Viết code cho sự kiện huỷ chụp ảnh

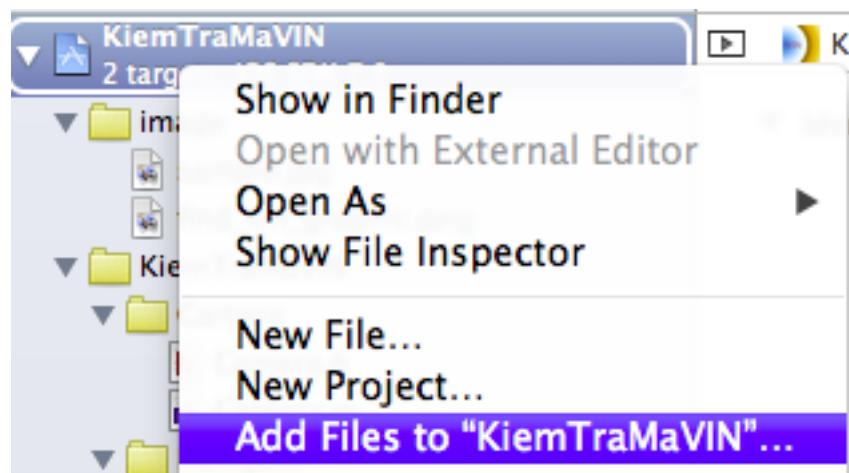
```

-(void) imagePickerControllerDidCancel:(UIImagePickerController *)picker
{
    [picker dismissViewControllerAnimated:YES completion:NULL];
}

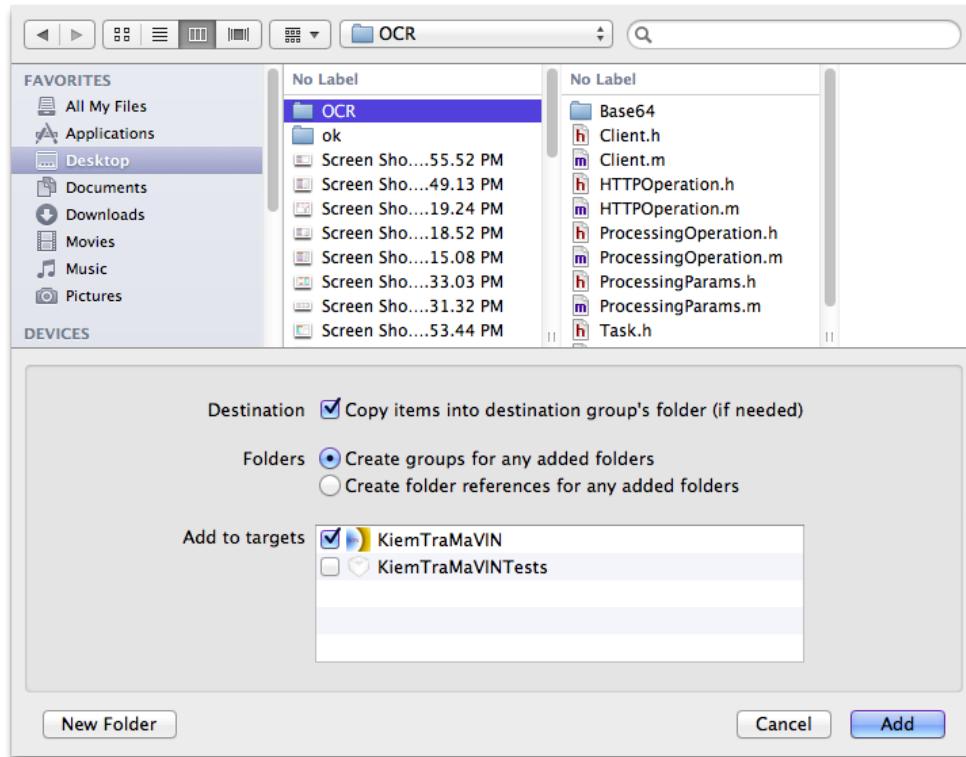
```

**Hình 6.102** Viết code cho sự kiện hủy chụp ảnh

→ Thêm bộ thư viện OCR vào project.

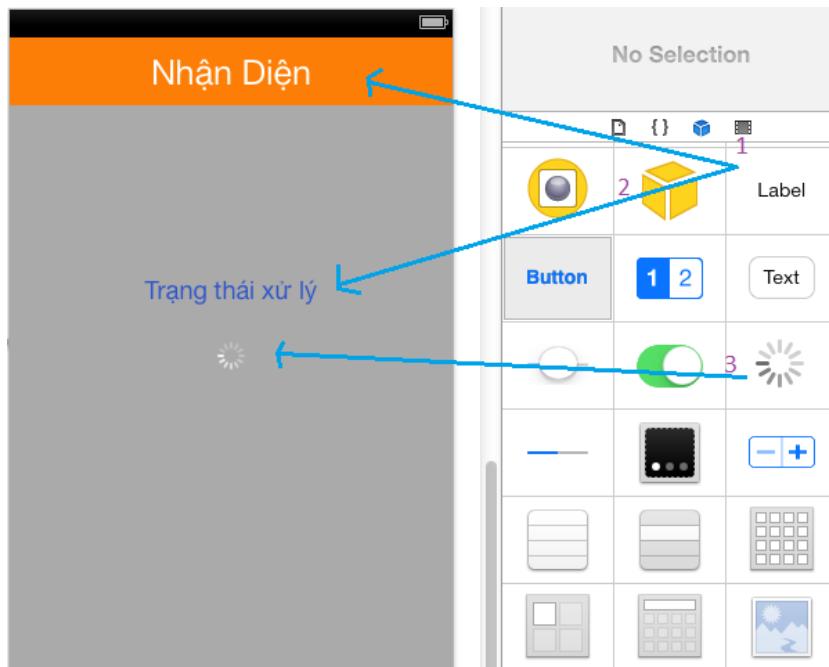


**Hình 6.103** Add Files vào project



**Hình 6.104 Chọn thư viện OCR và Add**

**Bước 7:** Thêm một viewcontroller mới đặt tên là Nhận Diện, thiết kế như hình.

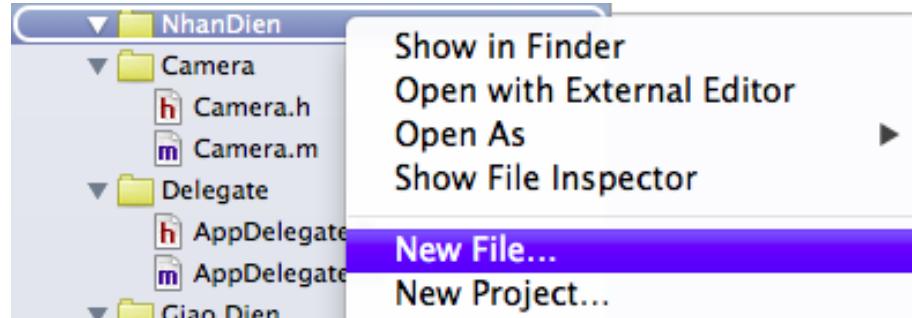


**Hình 6.105 Thiết kế giao diện Nhận Diện**

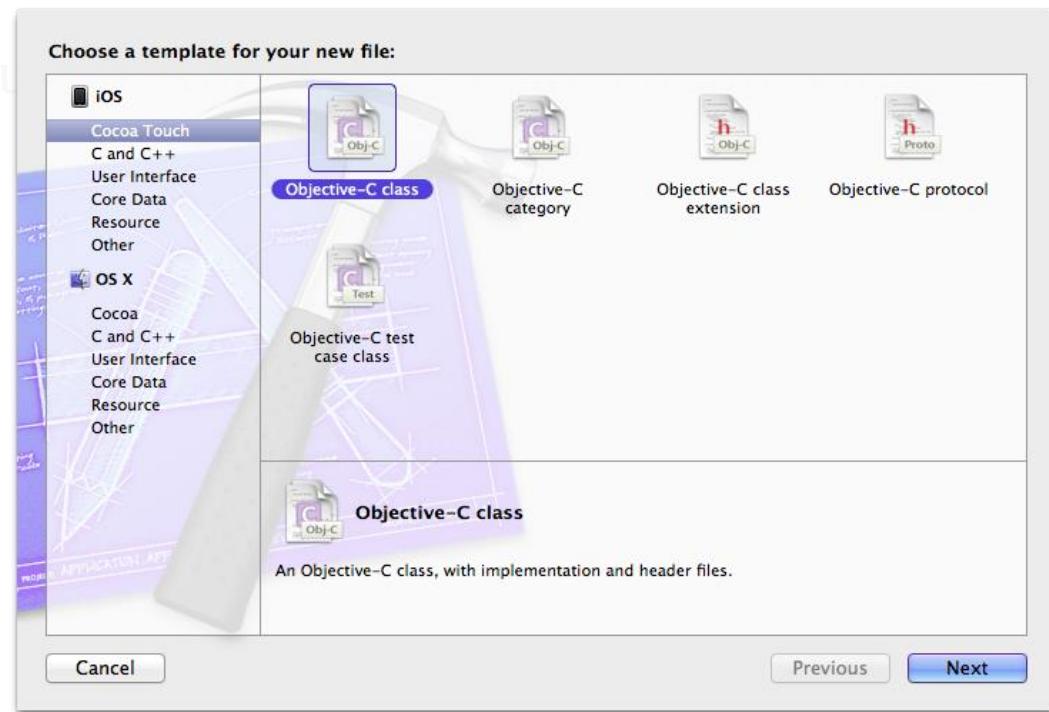
Trong đó

- [1] Label hiển tên của viewController Nhận Điện.
- [2] Label hiển thị trạng thái xử lý của viewController.
- [3] Một Activity Indicator để hiện trạng thái đang xử lý.

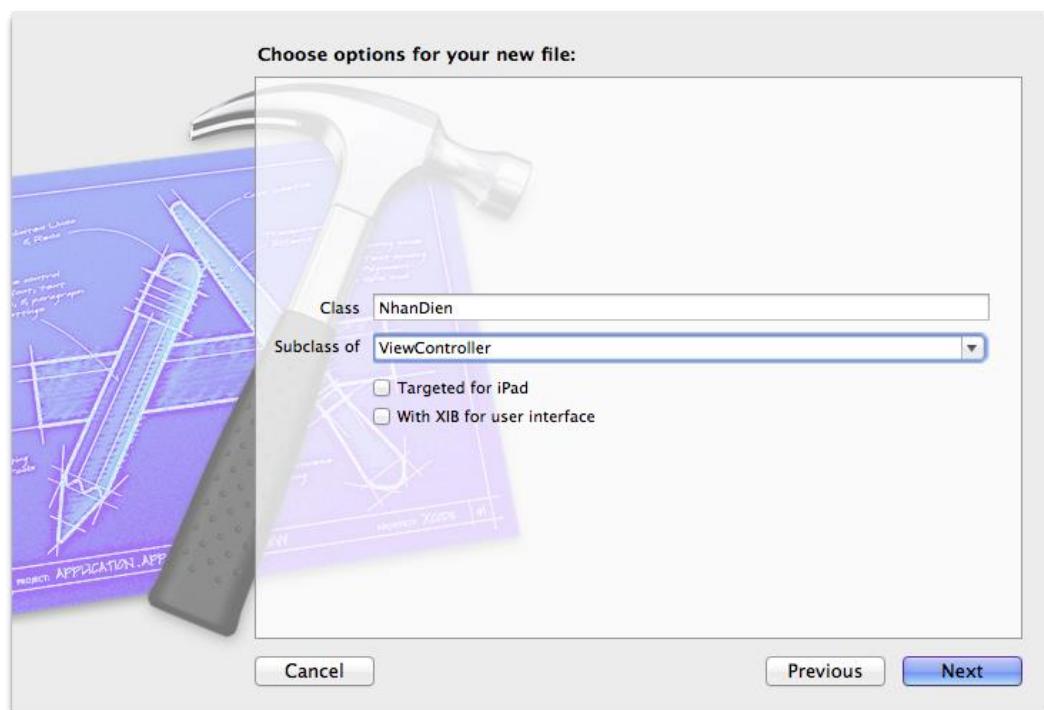
→ Thêm một class mới là NhanDien để quản lý viewcontroller mới.



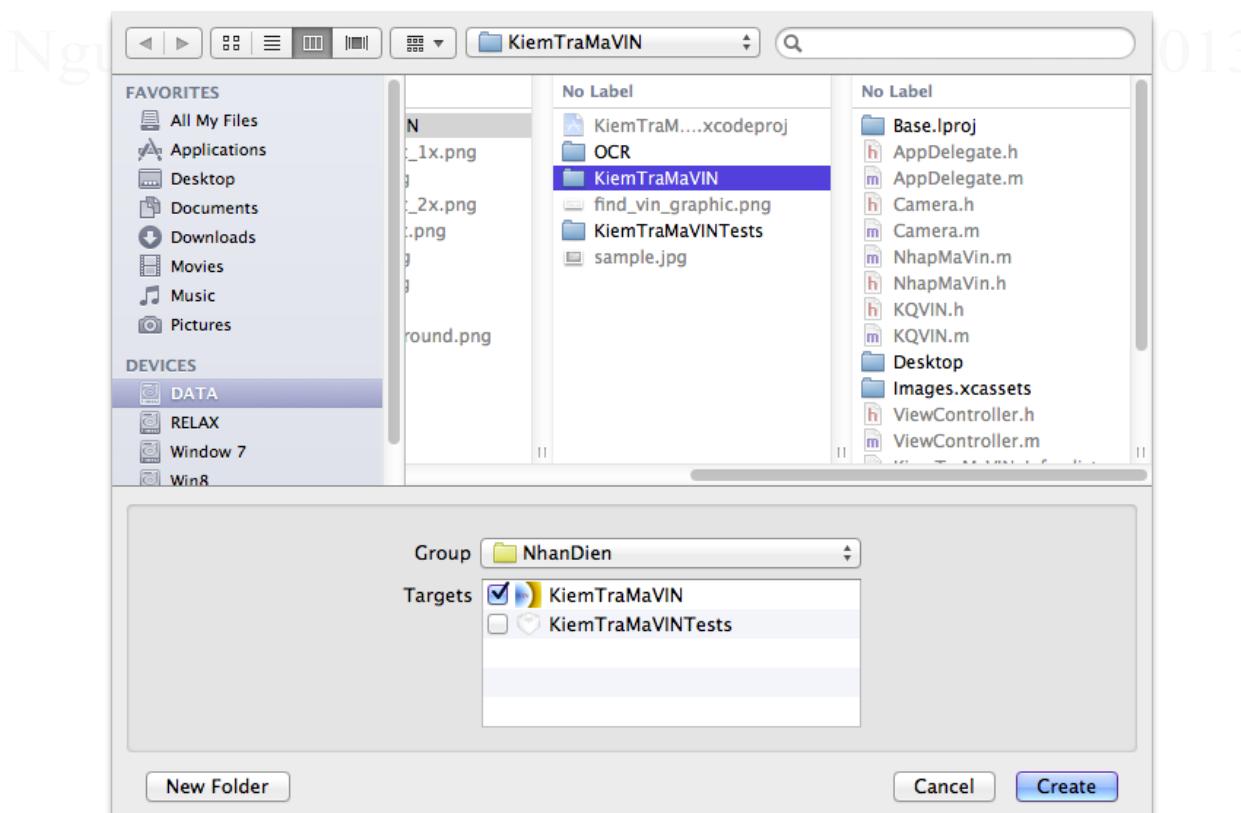
Hình 6.106 New File



Hình 6.107 Tạo class mới

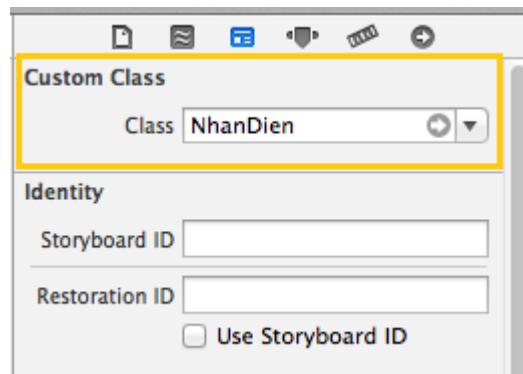


Hình 6.108 Class Nhan Dien



Hình 6.109 Create

→ Thêm class NhanDien làm class quản lý cho ViewController Nhận Diện.



**Hình 6.110 Đặt class Nhan Dien làm class quản lý**

→ Ánh xạ đối tượng vào tập tin NhanDien.h

```
//khai bao bien cho giao dien
@property (strong, nonatomic) IBOutlet UILabel *statusLabel;
@property (strong, nonatomic) IBOutlet UIActivityIndicatorView *statusIndicator;
```

Nguyễn Anh Tú - Khoa Công nghệ Thông tin - Đại học Vàng © 2013

→ Import client.h vào thư viện và thừa kế lại lớp đó.

```
...
#import "ViewController.h"
#import "Client.h"

@interface NhanDien : ViewController<ClientDelegate>
```

**Hình 6.112 Import Client.h và thừa kế lại ClientDelegate**

→ Trong tập tin NhanDien.m, import thư viện AppDelegate.h

```
...
#import "NhanDien.h"
#import "AppDelegate.h"
```

**Hình 6.113 Import AppDelegate.h**

→ Truy cập vào địa chỉ [www.ocrsdk.com](http://www.ocrsdk.com) → Register Free Trail



Hình 6.114 Start free trial now



Hình 6.115 Register OCR SDK

→ Tại trang đăng ký, bạn có thể chọn lựa đăng nhập với tài khoản Google, Facebook hoặc đăng ký theo email mới. Bạn nên chọn tài khoản Google để rút ngắn thời gian.



Hình 6.116 Lựa chọn cách đăng ký

→ Sau khi đăng ký và logon thành công. Chọn Add New Application.



**Hình 6.117 Add New Application**

→ Diền Application Name rồi chọn Create Application

## ABBYY® Cloud OCR SDK *Console*

### Create new Application

Enter a descriptive name for your new Application

Application Name:

Example: AndroidBusinessCardReader

[Create Application](#)

**Hình 6.118 Create Application**

→ Mật khẩu sẽ được gửi về email, bạn kiểm tra email để lấy mật khẩu.

## ABBYY® Cloud OCR SDK *Console*

Application created successfully

Password for newly created Application **CheckVIN** has been sent to your email  
**caothienmokimlong@gmail.com**.

[Proceed](#)

**Hình 6.119 Mật khẩu gửi về email**

→ Sau khi có mật khẩu, trong **NhanDien.m**, khai báo tên application và mật khẩu đã đăng ký với ABBYY.

```
// ten Application  
static NSString* MyApplicationID = @"VIN Decoder";  
// mat khau  
static NSString* MyPassword = @"CXRKdYzHTfaVdwPrt  
+JD9A9v";
```

**Hình 6.120 Khai báo tên application và mật khẩu**

→ Viết thêm code sau đây để dễ dàng sử dụng các biến đã khai báo bên NhanDien.h

```
@implementation NhanDien  
  
@synthesize statusLabel;  
@synthesize statusIndicator;
```

**Hình 6.121 Viết code cho các biến đã khai báo trong NhanDien.h**

Ngôn Ngữ Apple Tiếng Việt - Cao Thành Vàng © 2013

```
- (void)viewDidUnload  
{  
    [self setStatusLabel:nil];  
    [self setStatusIndicator:nil];  
    [super viewDidUnload];  
    // Release any retained subviews of the main view.  
    // e.g. self.myOutlet = nil;  
}
```

**Hình 6.122 Viết hàm viewDidUnload**

→ Viết hàm viewWillAppear

```
- (void)viewWillAppear:(BOOL)animated  
{  
    statusLabel.hidden = NO;  
    statusIndicator.hidden = NO;  
    [super viewWillAppear:animated];  
}
```

**Hình 6.123 Viết hàm viewWillAppear**

→ Viết hàm `viewDidAppear`

```
- (void)viewDidAppear:(BOOL)animated
{
    statusLabel.text = @"Loading image...";

    UIImage* image = [[AppDelegate*] [[UIApplication sharedApplication] delegate] imageToProcess];

    Client *client = [[Client alloc] initWithApplicationID:MyApplicationID password:MyPassword];

    [client setDelegate:self];

    ProcessingParams* params = [[ProcessingParams alloc] init];

    [client processImage:image withParams:params];

    statusLabel.text = @"Uploading image...";

    [super viewDidAppear:animated];
}
```

**Hình 6.124** Viết hàm `viewDidAppear`

→ Viết hàm `clientDidFinishUpload`

```
- (void)clientDidFinishUpload:(Client *)sender
{
    statusLabel.text = @"Processing image...";
}
```

**Hình 6.125** `clientDidFinishUpload`

→ Viết hàm `clientDidFinishProcessing`

```
- (void)clientDidFinishProcessing:(Client *)sender
{
    statusLabel.text = @"Downloading result...";
}
```

**Hình 6.126** `clientDidFinishProcessing`

→ Viết hàm `didFailedWithError` xử lý nếu lỗi xảy ra.

```

- (void)client:(Client *)sender didFailedWithError:(NSError *)error
{
    UIAlertView* alert = [[UIAlertView alloc] initWithTitle:@"Error"
                                                    message:[error localizedDescription]
                                                    delegate:nil
                                                    cancelButtonTitle:@"Cancel"
                                                    otherButtonTitles:nil, nil];

    [alert show];

    statusLabel.text = [error localizedDescription];
    statusIndicator.hidden = YES;
    Back.hidden = NO;
}

```

**Hình 6.127 didFailedWithError**

→ Trong NhanDien.h, khai báo contactDB dạng sqlite3 và databasePath dạng NSString để sử dụng tra cứu cơ sở dữ liệu.

```

@interface NhanDien : ViewController<ClientDelegate>
{
    //khai bao doi tuong quan ly sqlite
    sqlite3 *contactDB;
    NSString *databasePath;
}

```

Nguyễn Anh Tú - Tài liệu Vàng © 2013

**Hình 6.128 Khai báo đối tượng**

→ Import thư viện “sqlite3.h”.

```

#import "ViewController.h"
#import "Client.h"
#import "sqlite3.h"

```

**Hình 6.129 Import sqlite3.h**

→ Trong NhanDien.h, khai báo các đối tượng lưu trữ dữ liệu từ sqlite.

```

//khai bao bien luu ket qua VIN
@property (strong,nonatomic) NSString *manufactor;
@property (strong,nonatomic) NSString *make;
@property (strong,nonatomic) NSString *type;
@property (strong,nonatomic) NSString *year;
@property (strong,nonatomic) NSString *factory;
@property (strong,nonatomic) NSString *fuel;

```

**Hình 6.130 Khai báo đối tượng lưu kết quả**

→ Tiếp tục khai báo thêm các hàm cần sử dụng.

```
//khai bao ham su dung  
- (BOOL)KiemTraDoDai: (NSString*) inputstring;  
- (BOOL)KiemTraDigit: (char) inputpoint9;  
- (NSString*)ChuanHoa: (NSString *)result;
```

### Hình 6.131 Khai báo các hàm

→ Trong NhanDien.m, import “KQVIN.h” và “AppDelegate.h”.

```
#import "NhanDien.h"  
#import "AppDelegate.h"  
#import "KQVIN.h"
```

### Hình 6.132 Import thư viện

→ Khai báo các biến lưu đường dẫn, biến dạng sqlite3\_stmt để lưu kết quả từ sqlite, các biến bên NhanDien.h để truy xuất dễ dàng.

```
//khai bao duong dan + bien truy xuat sqlite  
const char *dbpath;  
sqlite3_stmt *statement;  
  
@implementation NhanDien  
  
//khai bao de truy xuat toan cuc cac thuoc tinh cua VIN  
@synthesize manufactor;  
@synthesize make;  
@synthesize type;  
@synthesize year;  
@synthesize factory;  
@synthesize fuel;
```

### Hình 6.133 Khai báo biến

→ Viết code cho hàm viewDidLoad như NhapMaVIN.m.

```

- (void)viewDidLoad
{
    [super viewDidLoad];

    //khai báo doi tuong luu duong dan thu muc
    NSString *docsDir;

    //luu duong dan thu muc
    docsDir = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)[0];

    //lay duong dan tap tin sqlite
    databasePath = [[NSString alloc] initWithString:[docsDir
        stringByAppendingPathComponent:@"VIN_Database.sqlite"]];

    //khai báo doi tuong quan ly tap tin de kiem tra tap tin sqlite co ton tai hay khong
    NSFileManager *filemgr = [NSFileManager defaultManager];

    if([filemgr fileExistsAtPath:databasePath]==NO)
    {
        NSString *databasePathFromApp = [[[NSBundle mainBundle] resourcePath]
            stringByAppendingPathComponent:@"VIN_Database.sqlite"];
        [filemgr copyItemAtPath:databasePathFromApp toPath:databasePath error:nil];
    }

    // Do any additional setup after loading the view.
}

```

### Hình 6.134 Viết code hàm viewDidLoad

→ Viết code cho hàm KiemTraDoDai và KiemTraDigit đã khai báo bên NhanDien.h

```

Nguyễn A © 2013
- (BOOL)KiemTraDoDai: (NSString*) inputstring
{
    if (inputstring.length==17) {
        return TRUE;
    }
    else
        return FALSE;
}

- (BOOL)KiemTraDigit:(char)inputpoint9
{
    if (inputpoint9=='X') {
        return TRUE;
    }
    else
    {   if (inputpoint9>='0' && inputpoint9<='9') {
        return TRUE;
    }
    else
        return FALSE;
}

```

### Hình 6.135 Viết code cho hàm đã khai báo

→ Viết code cho hàm ChuanHoa để bóc tách số VIN từ dữ liệu trả về. Do dữ liệu từ server sẽ trả về một chuỗi các ký tự mà server nhận diện được, do đó cần phải chuẩn hóa

các ký tự đặc biệt thành khoảng trắng, sau đó tiến hành bóc tách mã VIN từ chuỗi đã chuẩn hóa.

```
- (NSString*)ChuanHoa: (NSString *)result
{
    //thay the ki tu dac biet
    NSRegularExpression *expression = [NSRegularExpression regularExpressionWithPattern:@"[`:,|\\t\\n.*!'-]" options:0 error:NULL];

    NSString *clearstring = [expression stringByReplacingMatchesInString:result options:0 range:NSMakeRange
    (0, [result length]) withTemplate:@" "];
    NSLog(@"clear \n %@",clearstring);

    NSMutableString *str = [NSMutableString stringWithString:clearstring];
    NSArray *arrstring = [str componentsSeparatedByString:@" "];
    NSString *callback;

    for(NSString *step in arrstring)
    {
        if([step length]>=17)
        {
            NSLog(@"%@",step);
            NSLog(@"%@",[step length]);
            callback = step;
        }
    }
    return callback;
}
```

### Hình 6.136 Hàm ChuanHoa

→ Viết hàm **didFinishDownloadData** xử lý dữ liệu trả về giống như code đã viết cho button Xem của view **Nhập Mã VIN**.

Trước tiên, từ chuỗi trả về sau khi chạy hàm ChuanHoa, tiến hành bỏ đi ký tự xuống dòng ở cuối chuỗi.

```
- (void)client:(Client *)sender didFinishDownloadData:(NSData *)downloadedData
{
    //lay du lieu tai ve
    statusLabel.hidden = YES;
    statusIndicator.hidden = YES;
    NSString* result = [[NSString alloc] initWithData:downloadedData encoding:NSUTF8StringEncoding];

    NSString *result2 = [self ChuanHoa:result];
    result2 = [result2 substringToIndex:17];
    NSLog(@"%@",result2);
    NSLog(@"%@",[result2 length]);
```

### Hình 6.137 Bỏ đi ký tự xuống dòng ở cuối chuỗi

Tiếp theo viết code để lấy đường dẫn của tập tin sqlite và kiểm tra độ dài, ký tự vị trí 9 của mã VIN giống như đã làm trong NhapMaVIN.m.

```
//luu lai duong dan database
dbpath = [databasePath UTF8String];

//khai bao luu tru vincode
NSString *vin_upcase= [[NSString alloc]init];

//chuan hoa thanh chu in hoa
vin_upcase = [result2 uppercaseString];
NSLog(@"%@",vin_upcase);

//kiem tra do dai chuoi
BOOL valuelenght=FALSE;
valuelenght = [self KiemTraDoDai:vin_upcase];
NSLog(@"%@",valuelenght);

//kiem tra vi tri thu 9
char point9;
point9 = [vin_upcase characterAtIndex:8];
BOOL valuedigit=FALSE;
valuedigit = [self KiemTraDigit:point9];
NSLog(@"%@",valuedigit);
```

### Hình 6.138 Viết code lấy đường dẫn database và kiểm tra vị trí thứ 9

Nếu kết quả trả về của các hàm kiểm tra là sai, thì xuất thông báo, nếu đúng sẽ triển khai tiếp trong hàm else.

```
//begin code
if (valuedigit==FALSE || valuelenght==FALSE) {
    UIAlertView *notice = [[UIAlertView alloc]initWithTitle:@"Thông Báo" message:@"Mã VIN sai!"
        delegate:self cancelButtonTitle:@"Đồng ý" otherButtonTitles:nil, nil];
    [notice show];
}
else
{
```

### Hình 6.139 Viết code xử lý các giá trị trả về từ hàm kiểm tra

Nếu kết quả trả về là đúng, thì tiến hành thực hiện code trong hàm else. Viết code trong hàm else tương tự NhapMaVIN.h. Trước hết tạo các câu lệnh truy vấn.

```

//kiem tra
//check point 1-3
NSMutableString *query13 = [NSMutableString stringWithString:@"SELECT * FROM Pos_123 WHERE
    vincode=''];
NSString *vincode13 = [[NSString alloc]init];
vincode13 = [vin_upcase substringWithRange:NSMakeRange(0, 3)];
[query13 appendString:vincode13];
[query13 appendString:@""];

//check point 8
NSMutableString *query8 = [NSMutableString stringWithString:@"SELECT * FROM Pos_8 WHERE vincode='"
    ];
NSString *vincode8 = [[NSString alloc]init];
vincode8 = [vin_upcase substringWithRange:NSMakeRange(7, 1)];
[query8 appendString:vincode8];
[query8 appendString:@""];

//check point 10
NSMutableString *query10 = [NSMutableString stringWithString:@"SELECT * FROM Pos_10 WHERE
    vincode='"];
NSString *vincode10 = [[NSString alloc]init];
vincode10 = [vin_upcase substringWithRange:NSMakeRange(9, 1)];
[query10 appendString:vincode10];
[query10 appendString:@""];

//check point 11
NSMutableString *query11 = [NSMutableString stringWithString:@"SELECT * FROM Pos_11 WHERE
    vincode='"];
NSString *vincode11 = [[NSString alloc]init];
vincode11 = [vin_upcase substringWithRange:NSMakeRange(10, 1)];
[query11 appendString:vincode11];
[query11 appendString:@""];

```

N

13

### Hình 6.140 Viết câu lệnh truy vấn

Khai báo các biến để lưu lại câu lệnh truy vấn sau khi đã chuẩn hóa UTF-8.

```

const char *query_stmt13 = [query13 UTF8String];
const char *query_stmt8 = [query8 UTF8String];
const char *query_stmt10 = [query10 UTF8String];
const char *query_stmt11 = [query11 UTF8String];

```

### Hình 6.141 Khai báo biến lưu câu truy vấn đã chuẩn hóa

Mở kết nối tới cơ sở dữ liệu.

```

if(sqlite3_open(dbpath, &contactDB)==SQLITE_OK)
{

```

### Hình 6.142 Mở kết nối tới cơ sở dữ liệu

Viết code truy xuất dữ liệu cho các vị trí mã VIN 1-3, 8, 10, 11.

```

if (sqlite3_prepare_v2(contactDB,
                      query_stmt13, -1, &statement, nil) == SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        manufactor = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text
                                                (statement, 1)];
        make = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement
                                         , 2)];
        type = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement
                                         , 3)];
        NSLog(@"%@", @"\n %@", @"\n %@", manufactor, make, type);
    }
    else
    {
        NSLog(@"Không có dữ liệu");
    }
    sqlite3_finalize(statement);
}

```

**Hình 6.143 Truy xuất dữ liệu vị trí 1-3**

```

//-----
if(sqlite3_prepare_v2(contactDB,
                      query_stmt8, -1, &statement, nil) == SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        fuel = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement
                                         , 1)];
        NSLog(@"%@", fuel);
    }
    else
    {
        NSLog(@"No Fuel data");
        fuel = @"Không có dữ liệu";
    }
    sqlite3_finalize(statement);
}

```

**Hình 6.144 Truy xuất dữ liệu vị trí 8**

```

//-----
if (sqlite3_prepare_v2(contactDB,
                      query_stmt10, -1, &statement, nil) == SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        year = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text(statement
                                         , 1)];
        NSLog(@"%@", year);

    }
    else
    {
        NSLog(@"No Year data");
        year = @"Không có dữ liệu";
    }
    sqlite3_finalize(statement);
}

```

**Hình 6.145 Truy xuất dữ liệu vị trí 10**

```

//-----
if(sqlite3_prepare_v2(contactDB,
                      query_stmt11, -1, &statement, nil) == SQLITE_OK)
{
    if (sqlite3_step(statement) == SQLITE_ROW)
    {
        factory = [[NSString alloc] initWithUTF8String:(const char *) sqlite3_column_text
                                                 (statement, 1)];
        NSLog(@"%@",factory);
    }
    else
    {
        NSLog(@"No Factory data");
        factory = @"Không có dữ liệu";
    }
    sqlite3_finalize(statement);
}

```

**Hình 6.146 Truy xuất dữ liệu vị trí 11**

```

        sqlite3_close(contactDB);

    }
    else
    {
        NSLog(@"Can't Open Database");
    }
}

```

**Hình 6.147 Đóng kết nối và viết code trửng hợp không kết nối được sqlite**

→ Viết thêm code chuyển dữ liệu sang KQVIN

```

//tranfer data
KQVIN *kq = [self.storyboard instantiateViewControllerWithIdentifier:@"KQVIN"];

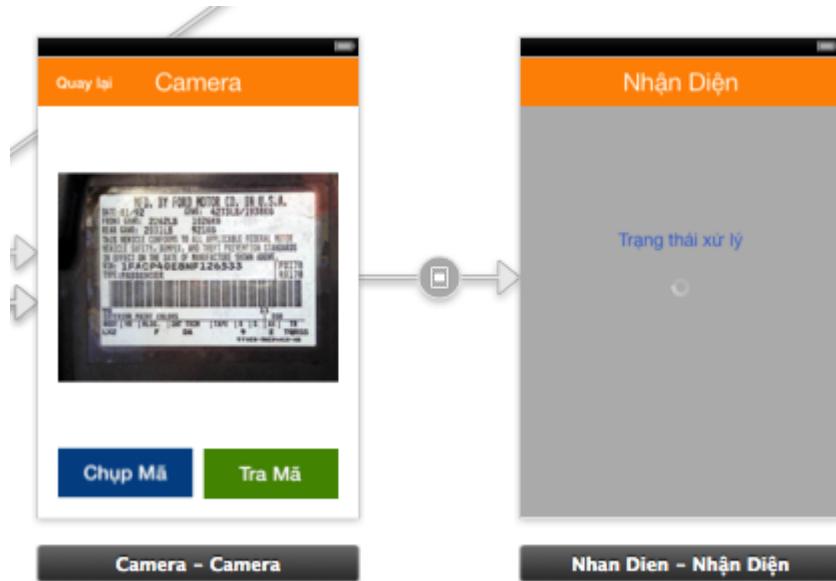
kq.manufactor = manufactor;
kq.make = make;
kq.type = type;
kq.fuel = fuel;
kq.year = year;
kq.factory = factory;

[self presentViewController:kq animated:YES completion:nil];

```

**Hình 6.148 Viết code chuyển dữ liệu sang KQVIN**

→ Tạo liên kết từ button Tra Mã của viewCamera sang viewNhanDien



**Hình 6.149 Tạo liên kết từ button Tra Mã sang view Nhận Diện**

## 6.2 PHẦN MỀM TÌM KIẾM ĐỊA ĐIỂM XUNG QUANH (PLACESNEARME)

### 6.2.1 Giới Thiệu

Khi đi du lịch, đi chơi đâu đó hay đến một nơi xa lạ nào đó hầu hết chúng ta đều có nhu cầu tìm kiếm những địa điểm xung quanh chúng ta trong 1 khoảng cách nào đó (100, 200mét,...). Chẳng hạn: ATM, khách sạn, quán ăn, cafe, khu du lịch, sân bay...

PlacesNearMe cho phép chúng ta làm được điều này.

### 6.2.2 Chuẩn Bị

- Để thuận tiện cho việc build ứng dụng lên iPhone, ứng dụng sẽ được viết trên phiên bản XCode 4 (Xcode 4.6.3).
- Một tài khoản Google để sử dụng dịch vụ Places API.
- Một bộ hình ảnh sử dụng cho PlacesNearMe đã lọc sẵn từ Google (xem folder source).

### 6.2.3 Cấu Trúc Phần Mềm

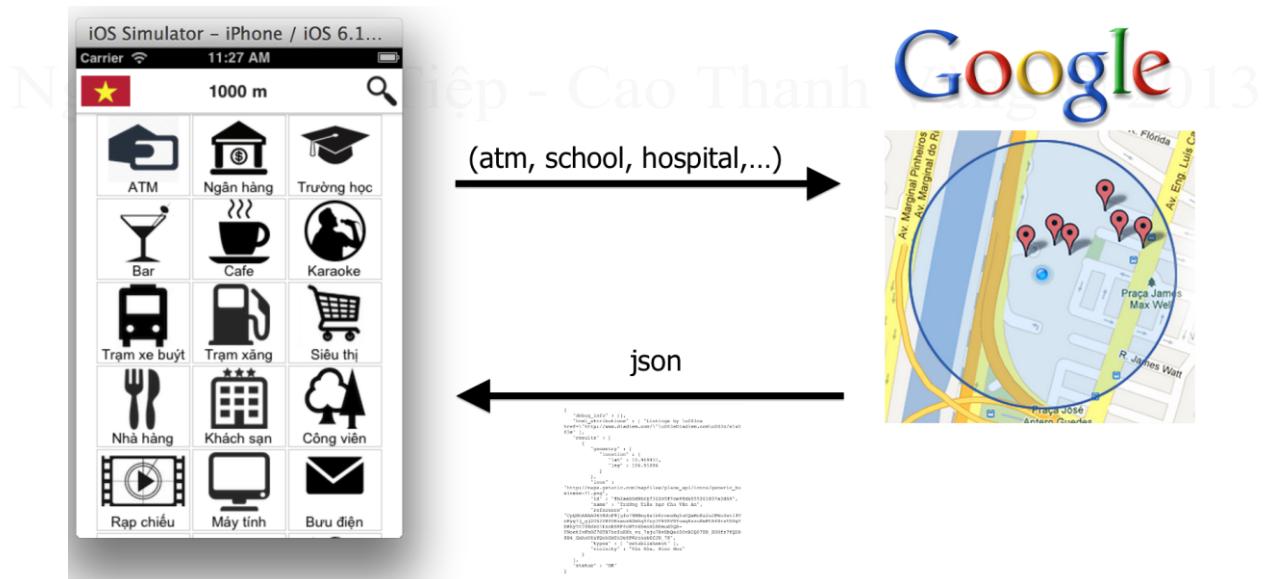
PlacesNearMe có cấu trúc gồm hai phần: lấy dữ liệu (Places API) và hiển thị dữ liệu (Các đối tượng trong Xcode).

Phần lấy dữ liệu sử dụng một dịch vụ do Google cung cấp cho phép tìm kiếm các địa điểm xung quanh một vị trí nào đó, dịch vụ này được gọi là Places API. Places API sẽ nhận vào các từ khóa tìm kiếm như: tọa độ, địa điểm, khoảng cách cần tìm ... và trả về một dạng văn bản có cấu trúc được gọi là json.

Phần hiển thị dữ liệu dựa vào các đối tượng hỗ trợ trong công cụ lập trình Xcode, lập trình viên sẽ xử lý và hiển thị dữ liệu json lên giao diện. Các đối tượng điển hình trong Xcode mà PlacesNearMe sử dụng: Map View, Web View, Alert View, Image View, Table View Controller...

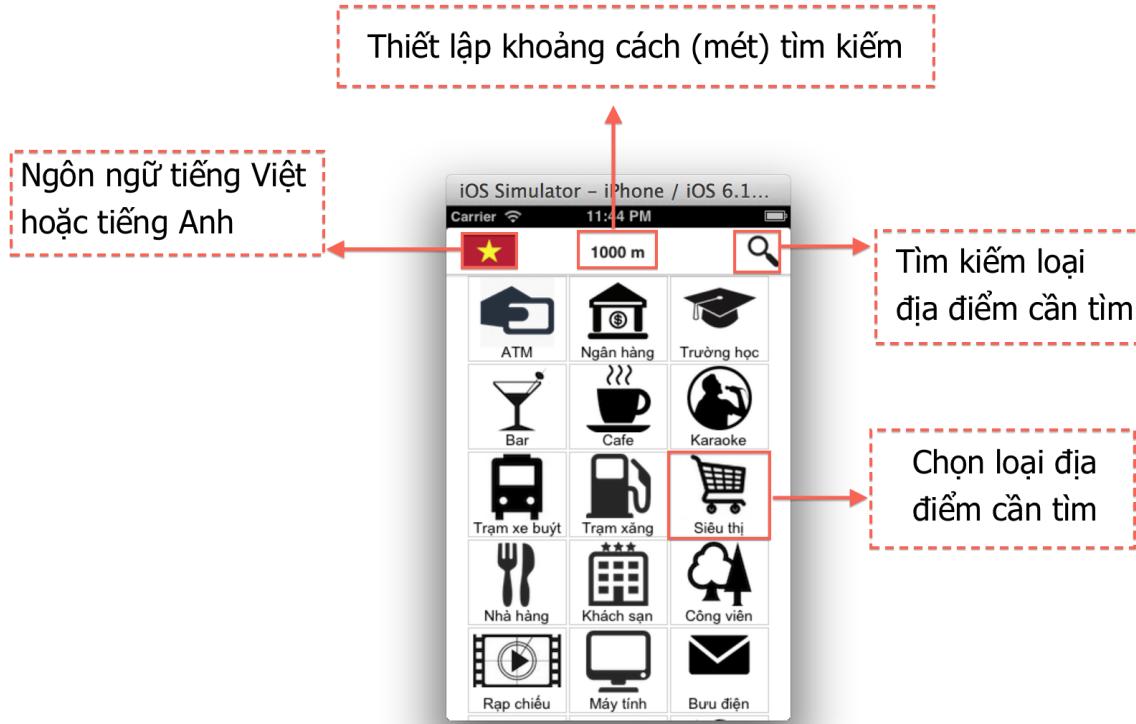
#### 6.2.4 Cơ Chế Vận Hành Của Placesnearme

PlacesNearMe sẽ gửi Google các từ khoá tìm kiếm (ATM, School, Hospital,...) Google sẽ trả về một dạng text có cấu trúc (json), dựa vào nguồn dữ liệu đó ứng dụng sẽ xử lý và thể hiện lên iPhone.



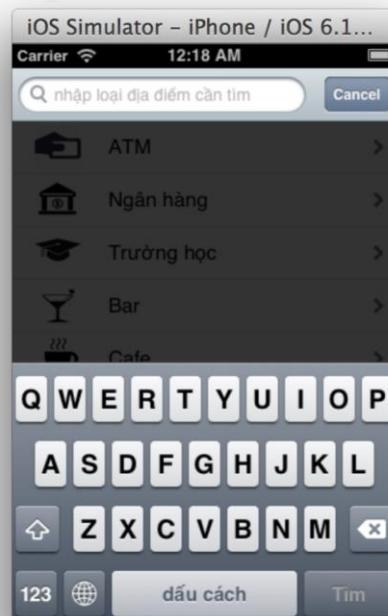
Hình 6.150 Cơ chế hoạt động của ứng dụng

### 6.2.5 Tính Năng



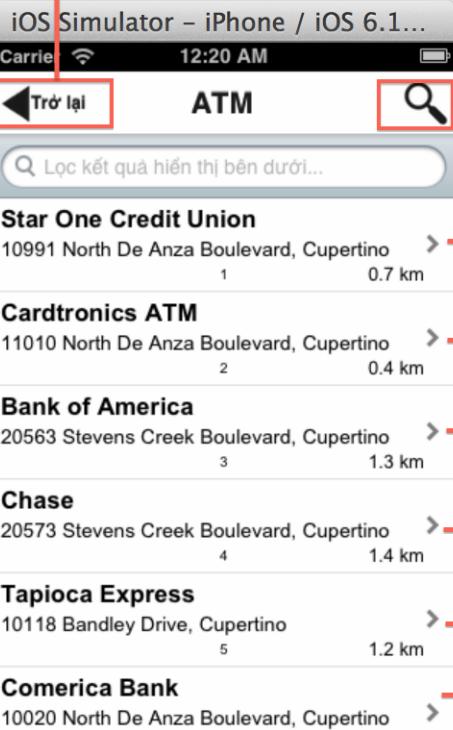
Hình 6.151 Các tính năng của phần mềm trên giao diện chính

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013



Hình 6.152 Giao diện của tính năng “tìm kiếm loại địa điểm cần tìm”

Trở lại giao diện chính

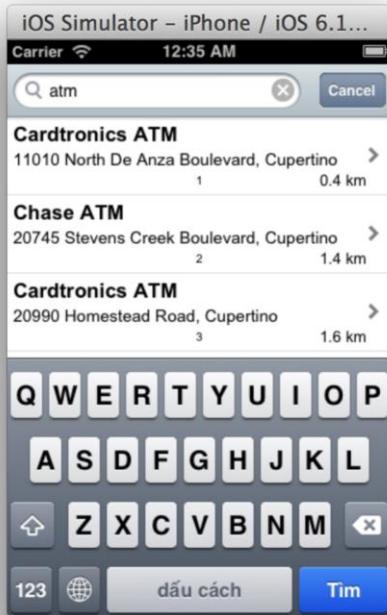


Lọc kết quả sau khi  
chọn loại địa điểm

Hiển thị kết quả tìm kiếm  
cho loại địa điểm ATM

Nguồn: Cao Thành Vàng © 2013

Hình 6.153 Các tính năng trên giao diện kết quả tìm kiếm

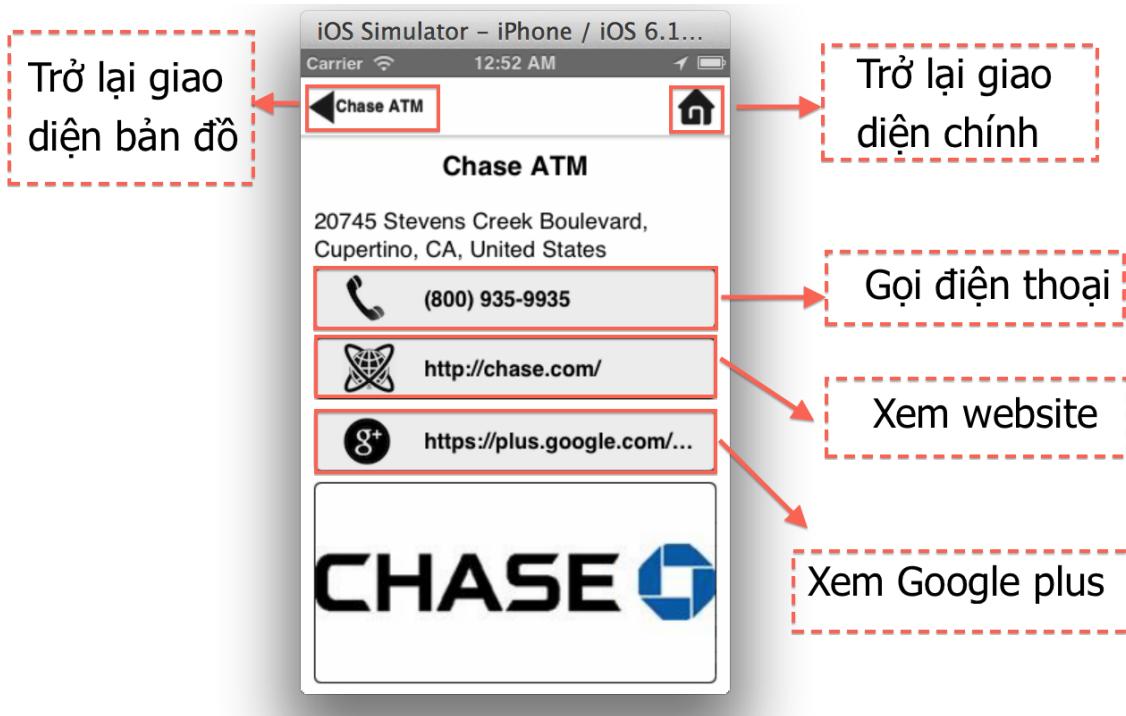


**Hình 6.154 Giao diện của tính năng “lọc kết quả sau khi đã chọn địa điểm”**

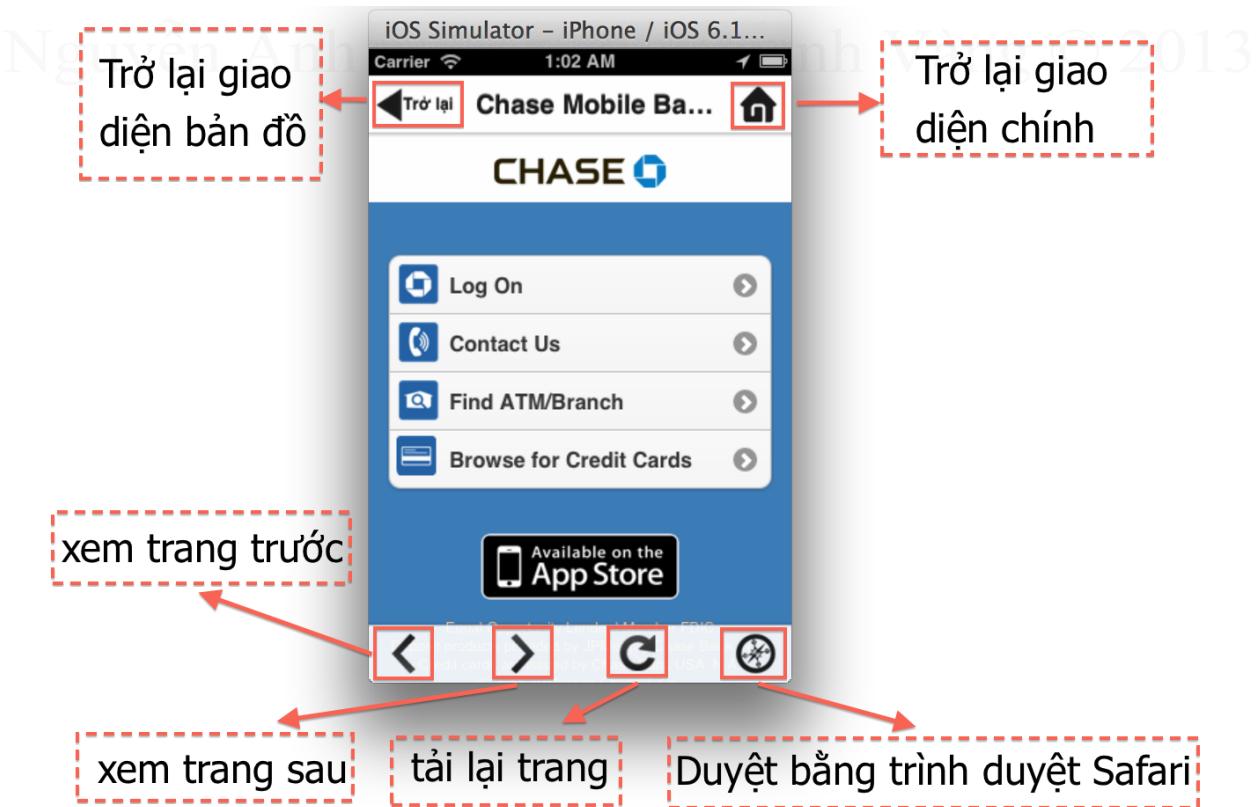
Nguyễn Anh Tiết - Cao Thành Vàng © 2013



**Hình 6.155 Các tính năng trên giao diện bản đồ**



**Hình 6.156 Các tính năng trên giao diện thông tin chi tiết của nơi tìm kiếm**



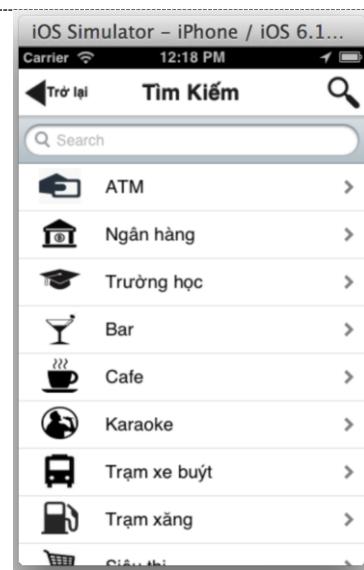
**Hình 6.157 Các tính năng trên giao diện duyệt web**

### **6.2.6 Tiến Hành**

Ứng dụng sẽ được tách thành nhiều ví dụ nhỏ chạy độc lập với nhau, sau khi hoàn thiện các ví dụ, bạn sẽ ghép các ví dụ nhỏ này lại tạo thành 1 ứng dụng như sau:



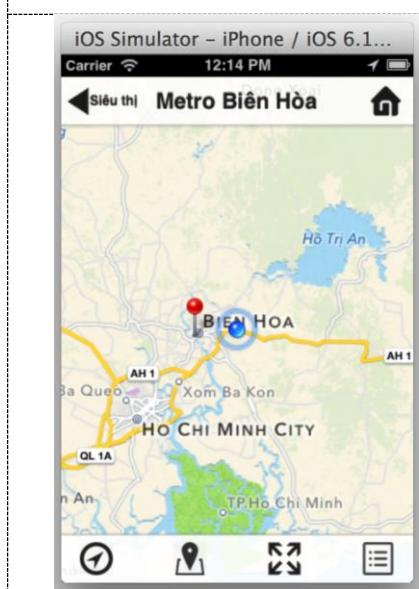
Hình 6.158 Giao diện chính



Hình 6.159 Giao diện tìm kiếm



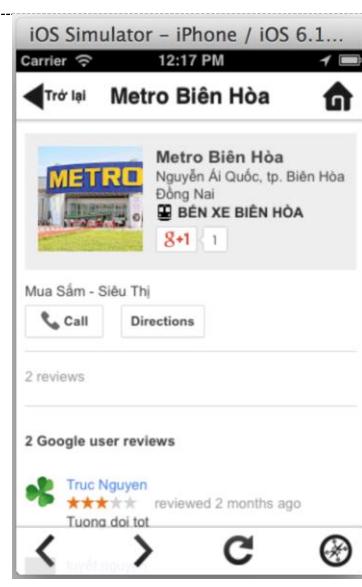
## Hình 6.160 Giao diện kết quả



Hình 6.161 Giao diện bản đồ.



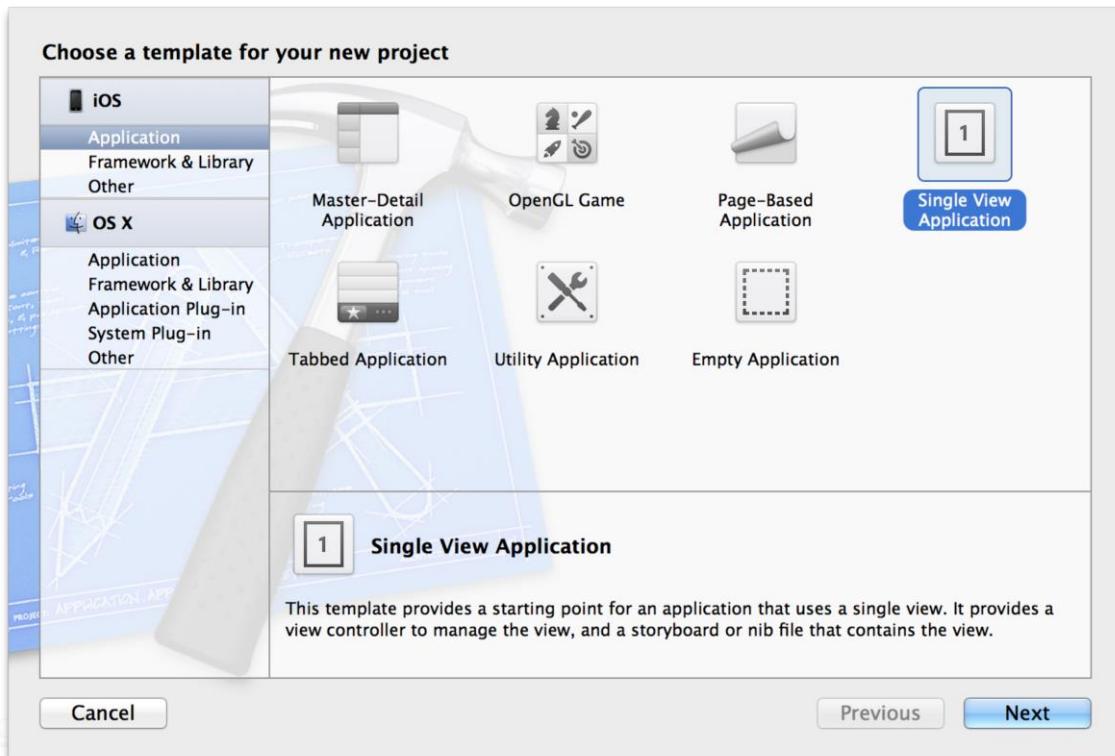
*Hình 6.162 Giao diện thông tin chi tiết của nơi tìm kiếm*



### Hình 6.163 Giao diện duyệt web.

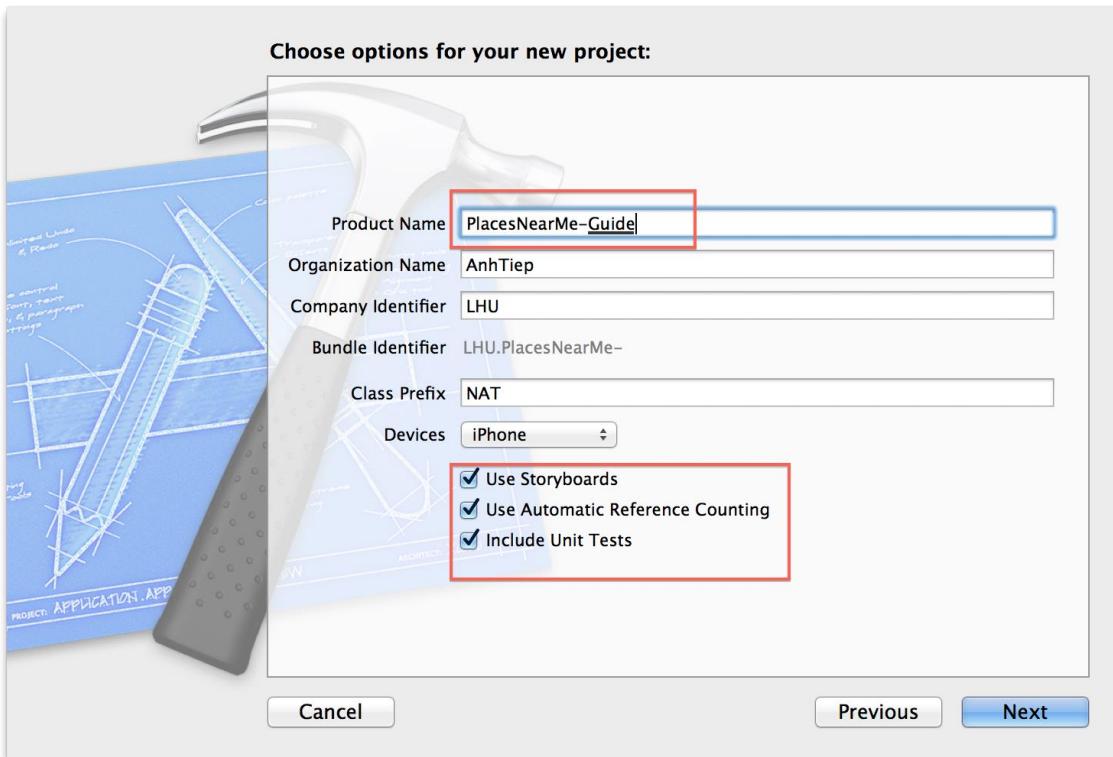
## Bước 1: Tạo project

Mở XCode → chọn “Single View Application”



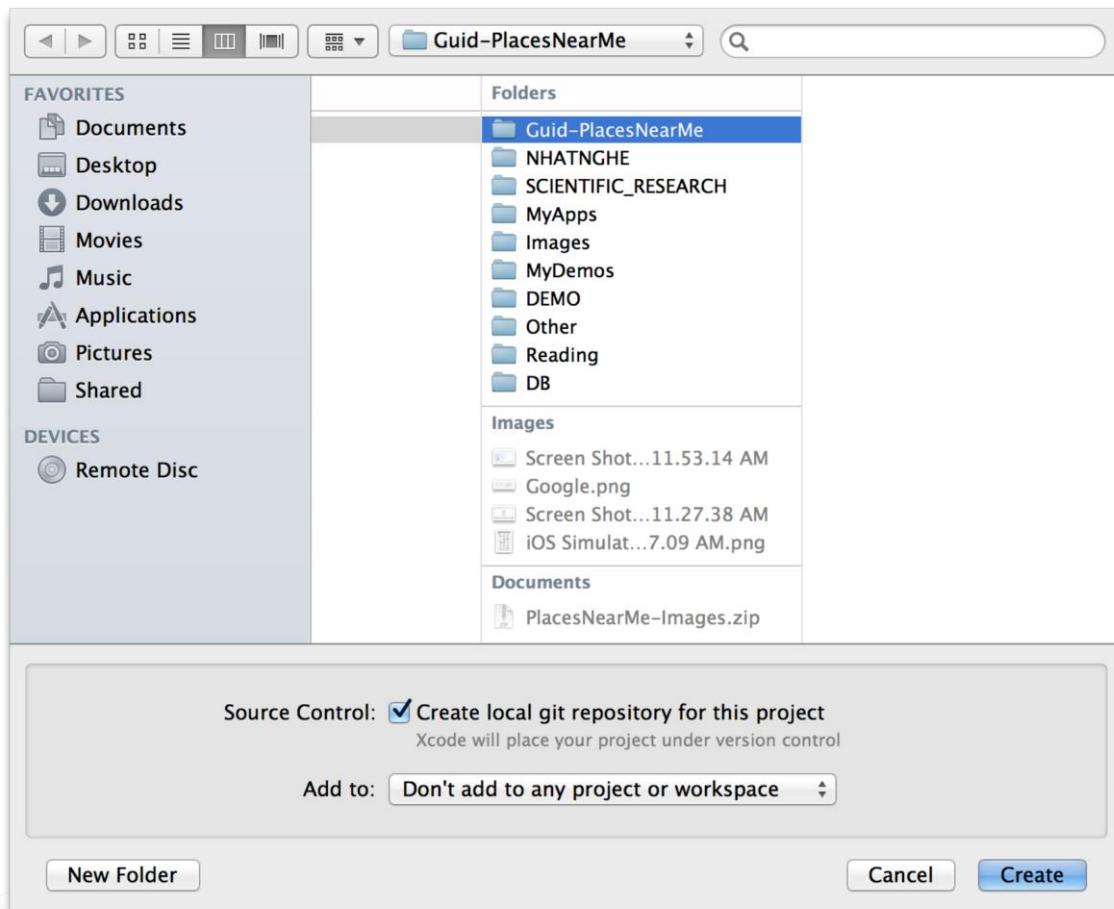
Hình 6.164 Tạo Single View Application

Click **Next** và nhập tên ứng dụng, check vào các đối tượng bên dưới để sử dụng Storyboard (dùng khi thiết kế ứng dụng có nhiều View Controller).



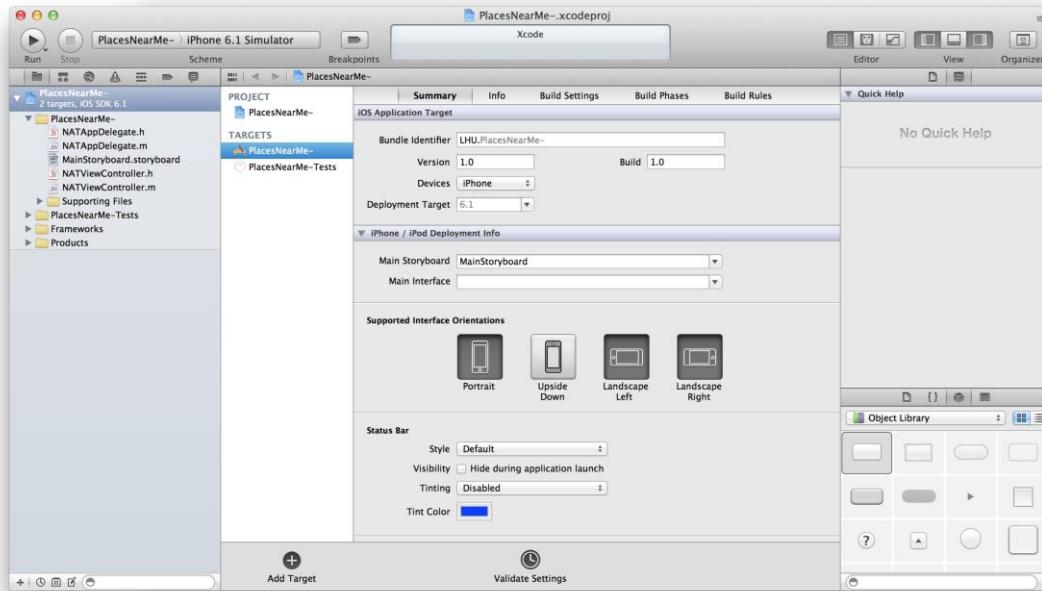
Hình 6.165 Check các đối tượng và chọn Next

Chọn vị trí cần lưu → click Ok



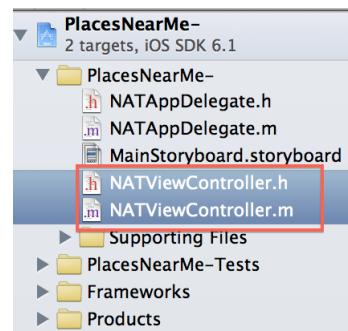
**Hình 6.166 Chọn nơi lưu project**

Giao diện sau khi tạo xong



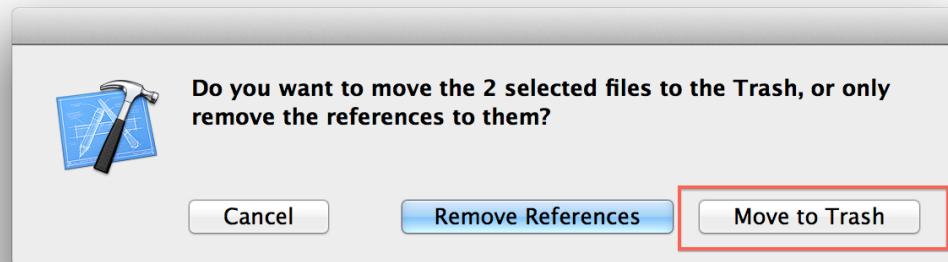
**Hình 6.167** Ứng dụng sau khi tạo xong

**Bước 2:** Xoá 2 file “**NATViewController.h**” và “**NATViewController.m**” vì hai file này sẽ tạo mới hoàn toàn.



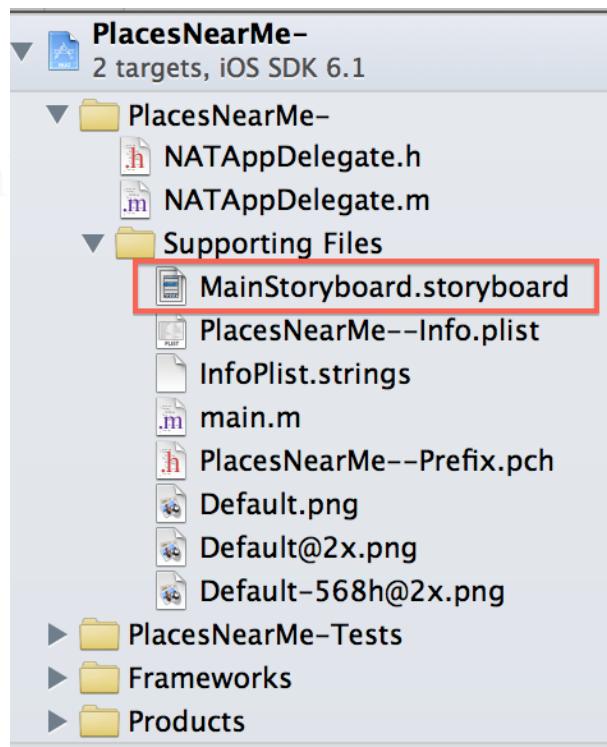
**Hình 6.168** Xóa hai file NATViewController

Chọn “**Move to Trash**” để xoá hoàn toàn



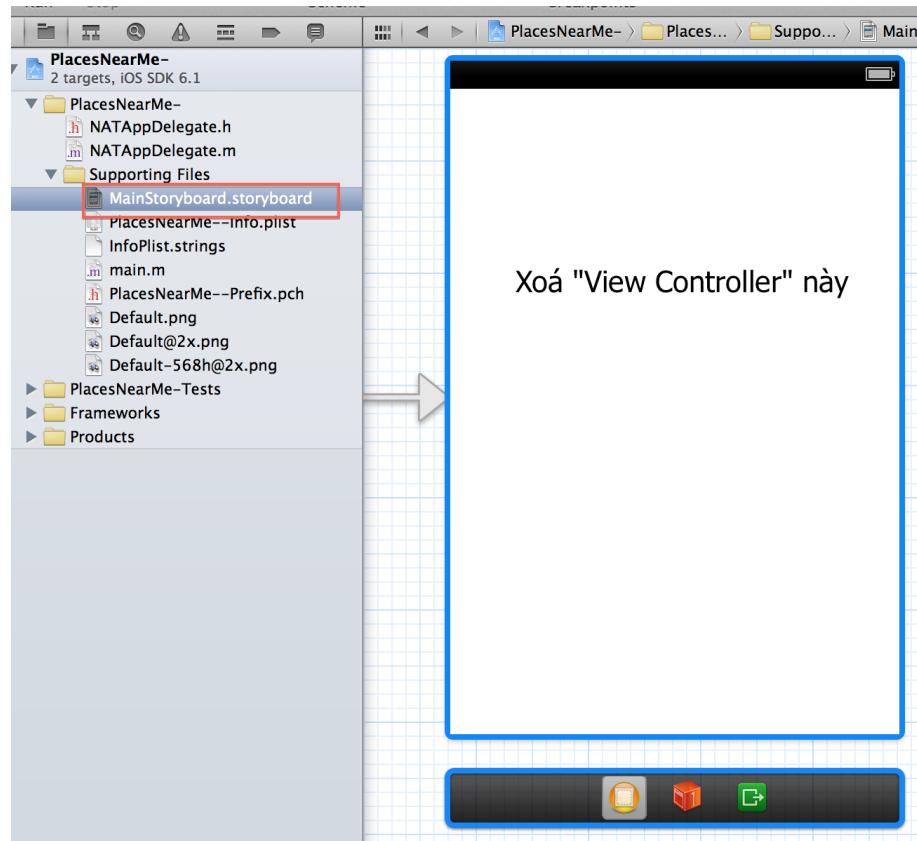
Hình 6.169 Move to Trash

Kéo thả file “**MainStoryboard.storyboard**” vào “**Supporting Files**” để tiện cho việc quản lý.



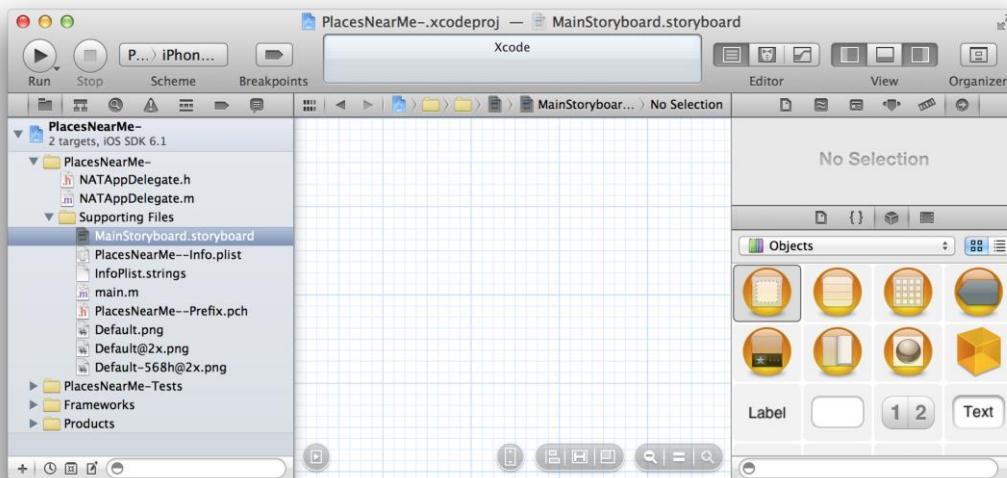
Hình 6.170 Di chuyển MainStoryboard

**Bước 3:** Click chọn “**MainStoryboard.storyboard**” và xóa giao diện “**View Controller**”



Nguyễn Anh Tô - Cao Thắng Vàng © 2013

Giao diện sau khi hoàn thành:



Hình 6.172 Giao diện sau khi hoàn thành

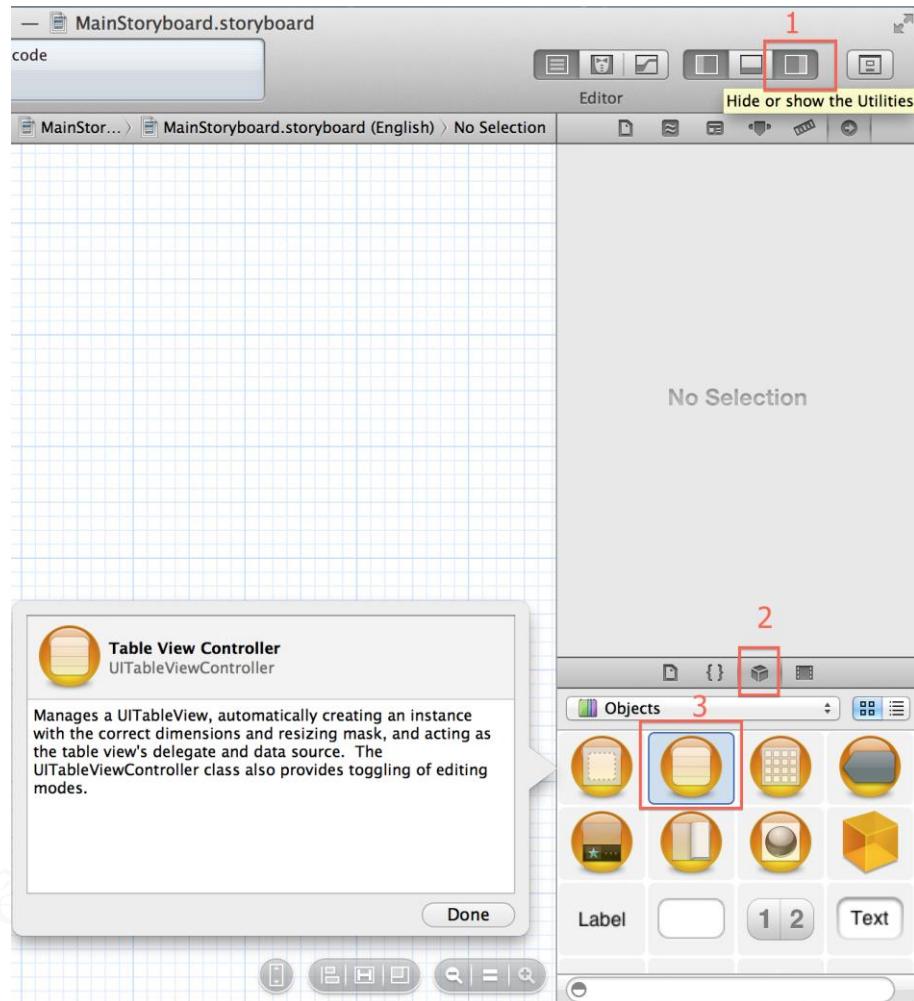
#### Bước 4: Xây dựng View Controller “HomeViewController”



Hình 6.173 Giao diện HomeViewController

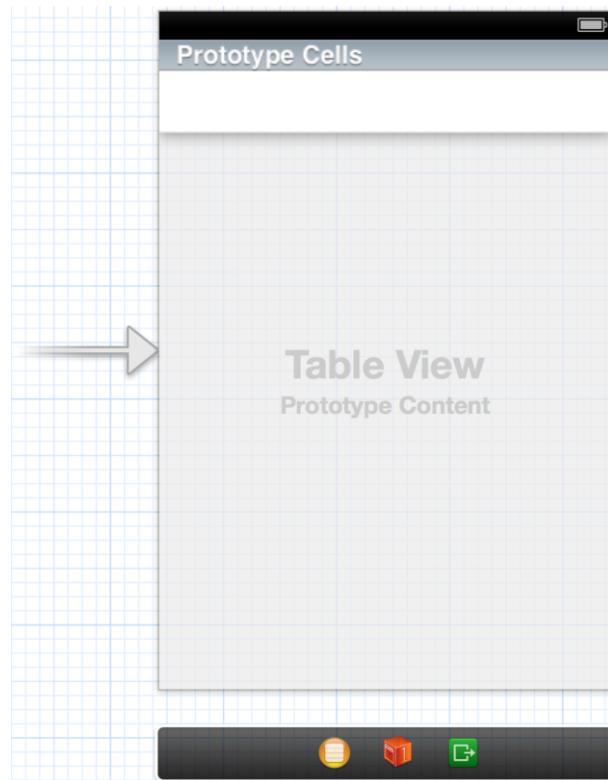
Kéo thả “Table View Controller” bằng cách:

→ Click chọn “MainStoryboard.storyboard” → nhấn phím tắt (**Cmd + option + 0**) hoặc nhìn vào góc phải phía trên chọn hình vuông kẽ cuối để show cửa sổ Utilities.



**Hình 6.174 Chọn đối tượng Table View Controller vào**

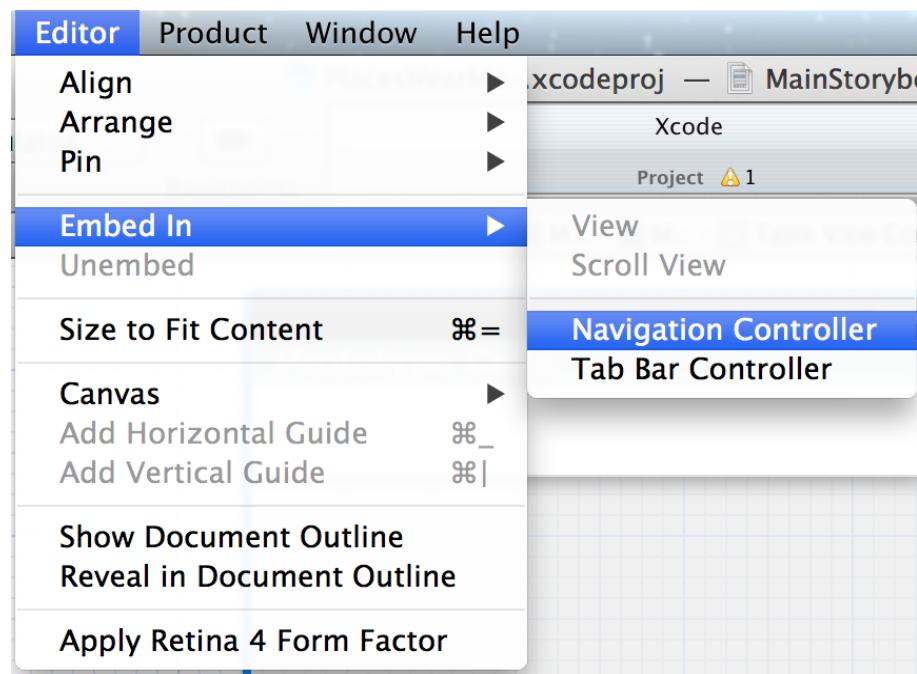
→ Kéo thả “**Table View Controller**” vào màn hình thiết kế (**Storyboard**), kết quả như sau.



**Hình 6.175 Giao diện Table View Controller**

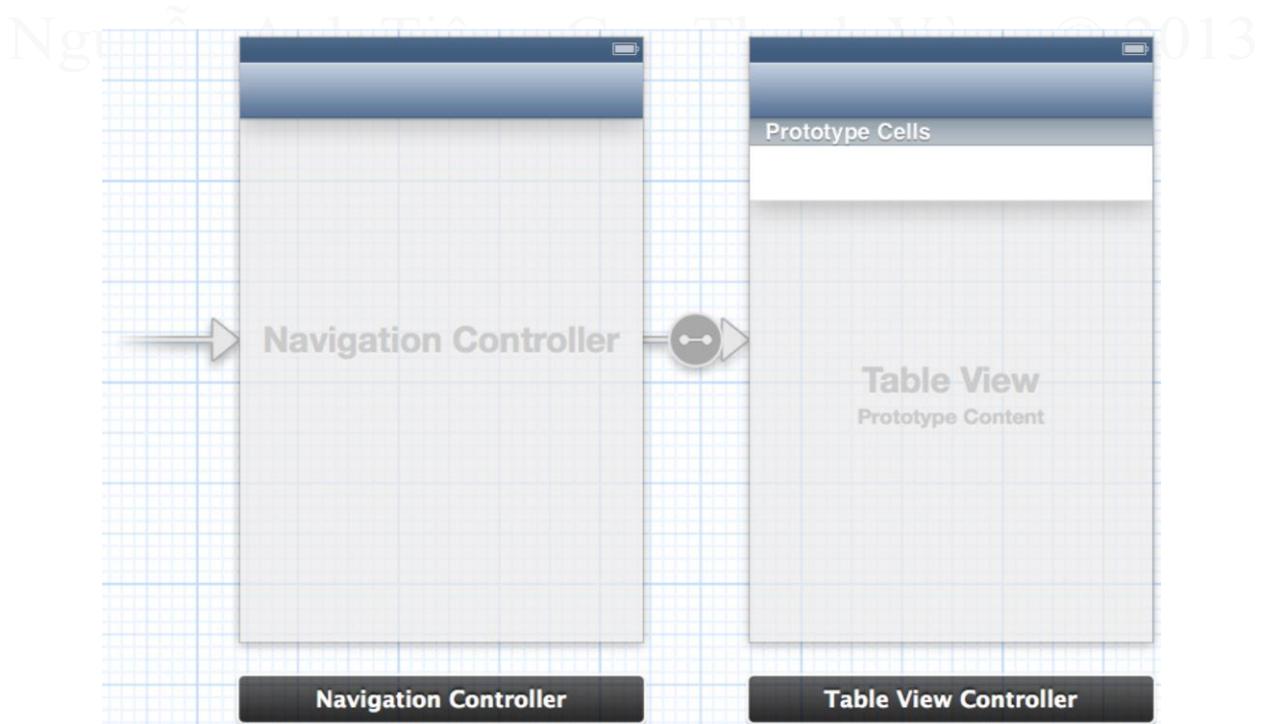
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
Tạo “Navigation Controller”:

→ Click chọn **Table View Controller** → click menu **Editor** → **Embed In** → **Navigation Controller**.



**Hình 6.176 Chọn Navigation Controller**

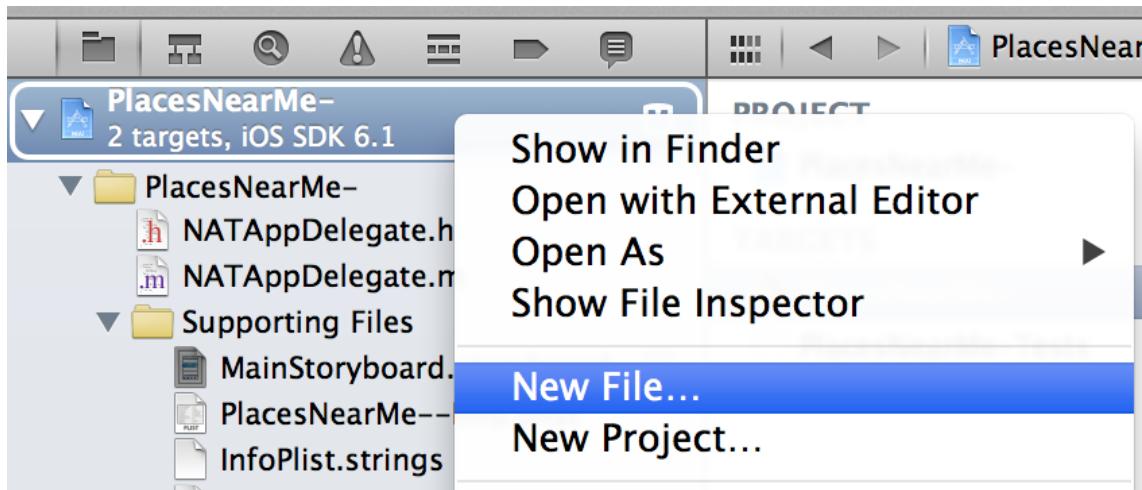
Kết quả



**Hình 6.177 Kết quả sau khi tạo Navigation Controller**

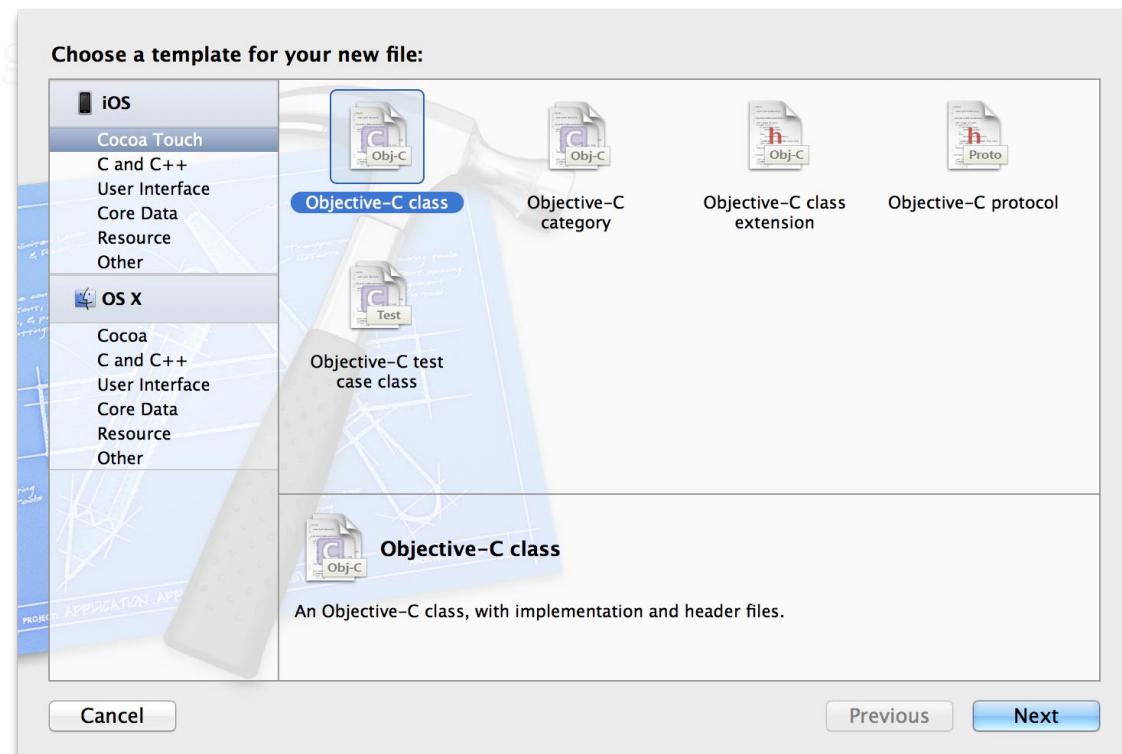
Tạo lớp đối tượng để viết code cho “**Table View Controller**”.

→ Khung bên trái góc trên click phải chuột vào Project → “New file” (phím tắt Cmd + N).



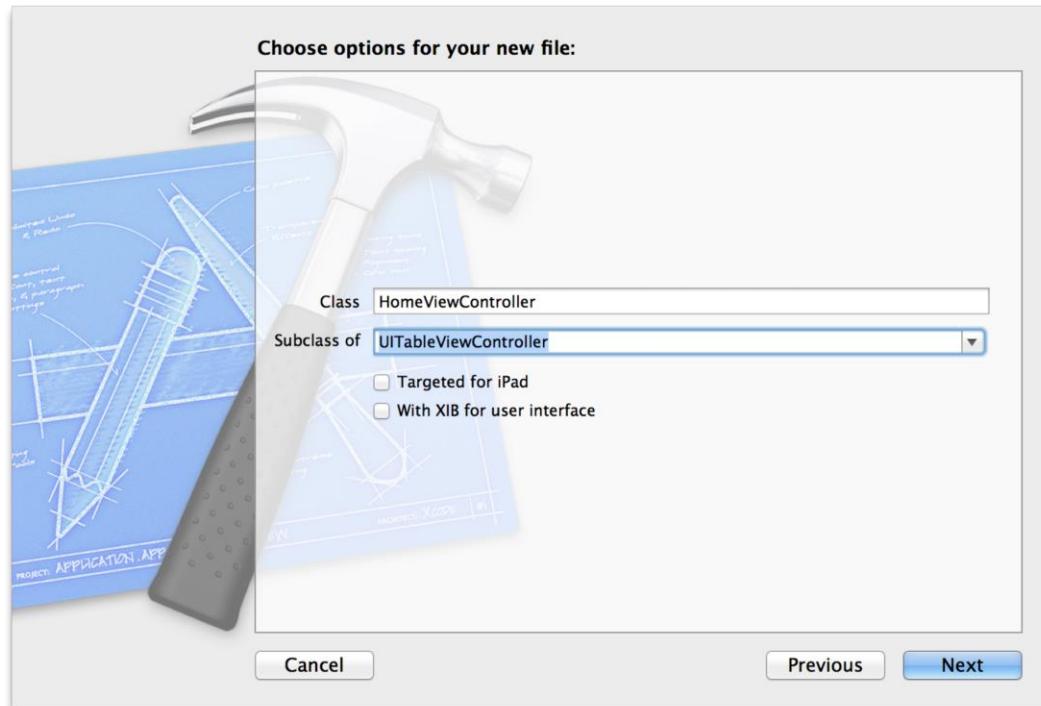
Hình 6.178 New File

→ Chọn “Objective-C Class” → Next



Hình 6.179 Objective-C Class

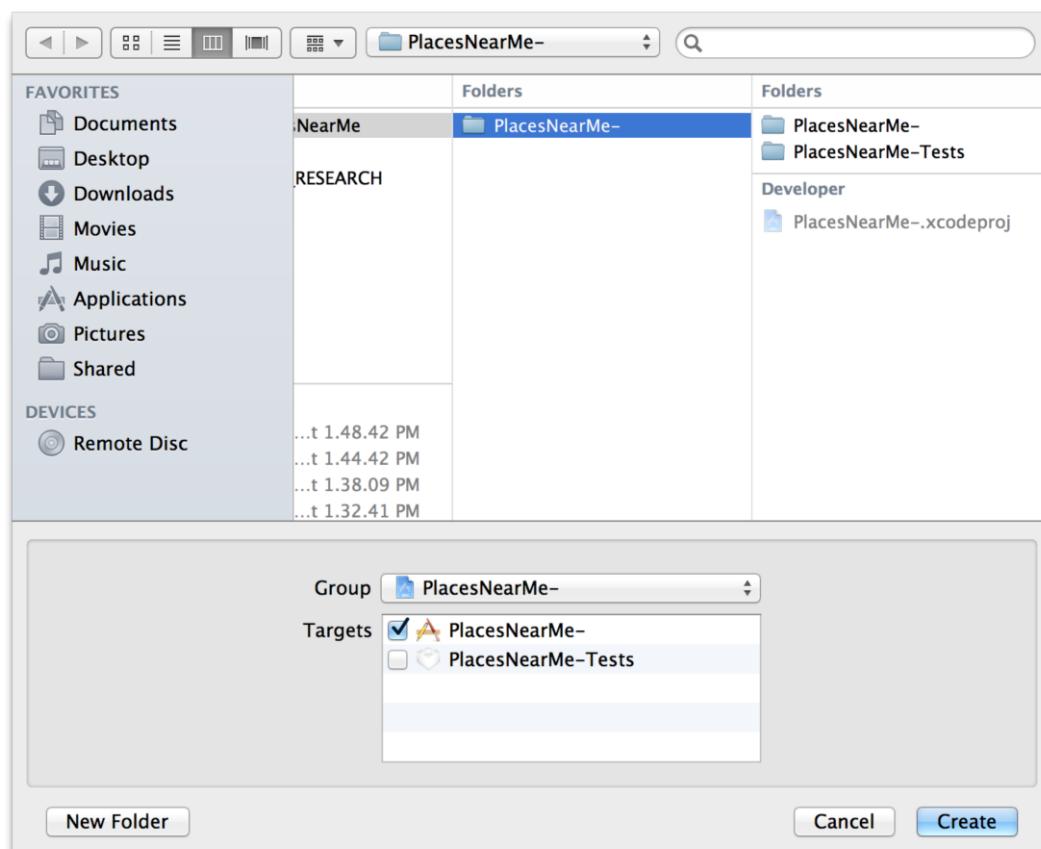
→ Subclass of: chọn “UITableViewController” → Class: đặt tên “HomeViewController” → “Next”.



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

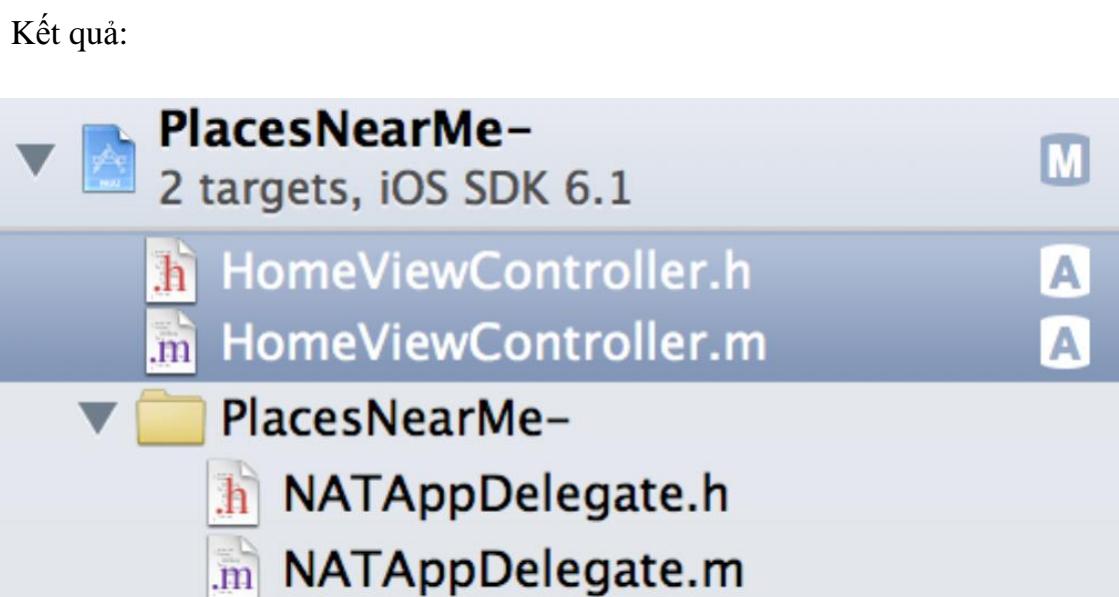
**Hình 6.180 Next**

→ Vị trí lưu để mặc định → Click “Create”



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

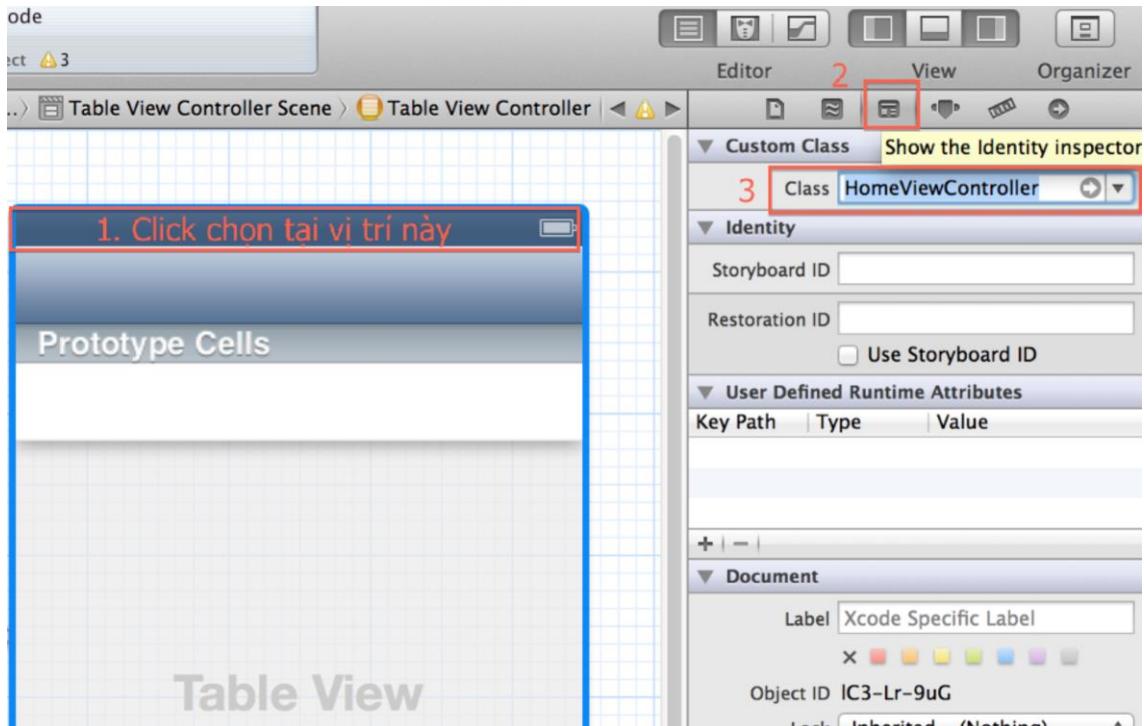
**Hình 6.181 Create**



**Hình 6.182 Kết quả**

Add lớp vừa tạo “HomeViewController” vào “TableViewCellController”.

→ Chọn “**TableViewController**” → “**Identity inspector**” → trong Class chọn hoặc nhập “**HomeViewController**”

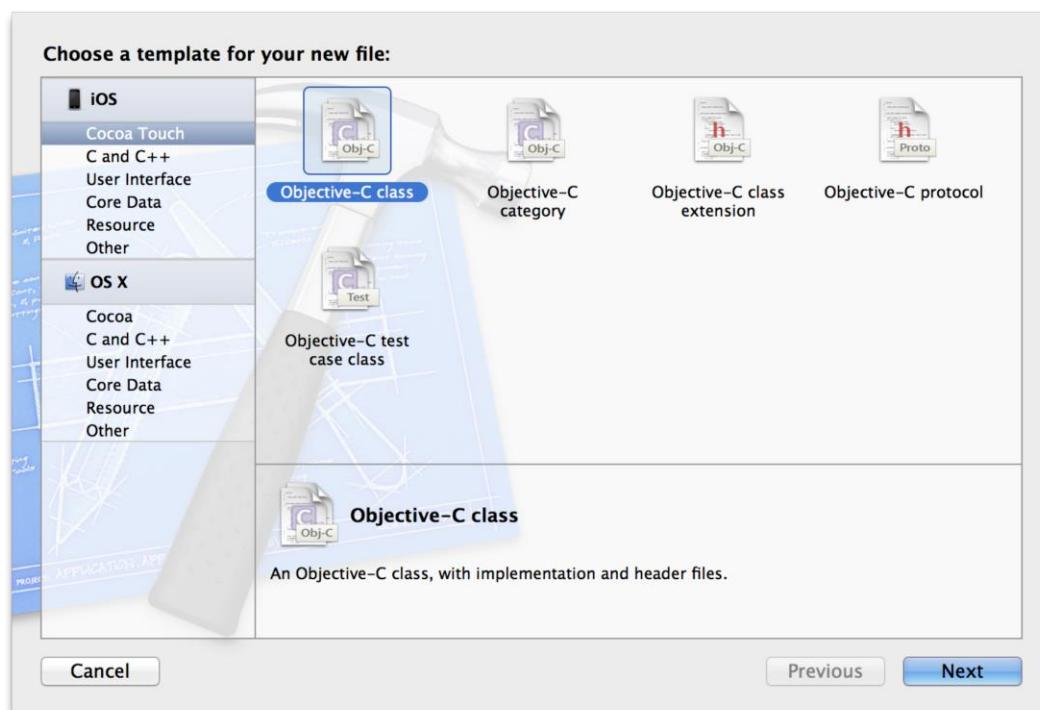


Hình 6.183 Thêm class mới tạo cho Table View Controller

Tạo đối tượng “**Place**” với các thuộc tính:

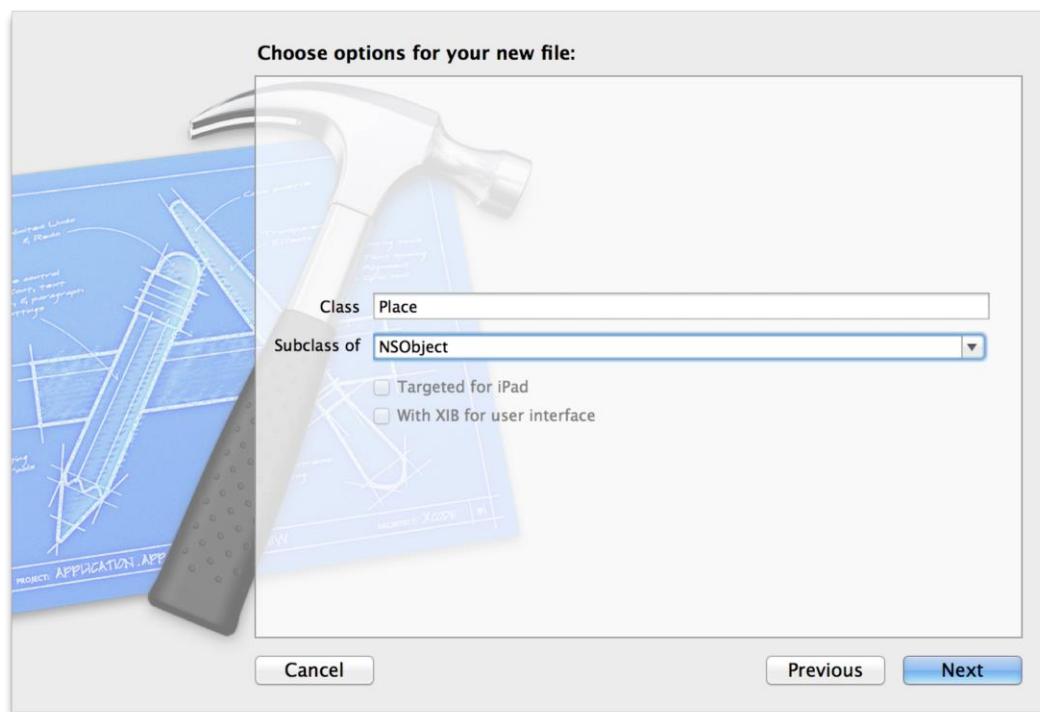
- **imageName**: tên hình;
- **titleEn**: tên hiển thị bằng tiếng anh
- **titleVi**: tên hiển thị bằng tiếng việt
- **placeType**: loại nơi tìm kiếm
- **keyWord**: từ khoá tìm kiếm

Tạo lớp “**Place**” bằng phím tắt “**Cmd + N**” → “**Objective-C Class**”.



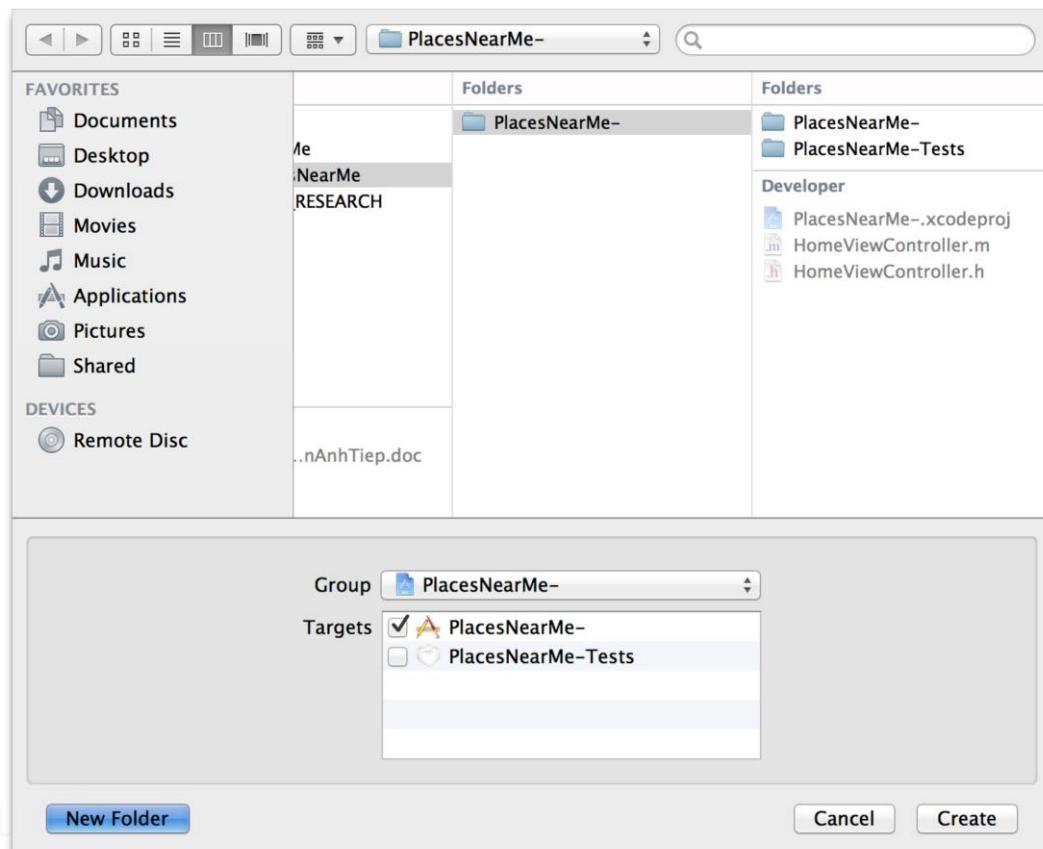
Hình 6.184 Tạo lớp Place

Class: “Place” → Subclass of: “NSObject” → “Next”  
Nguyễn Anh Hiệp - Cao Thanh Vàng © 2013



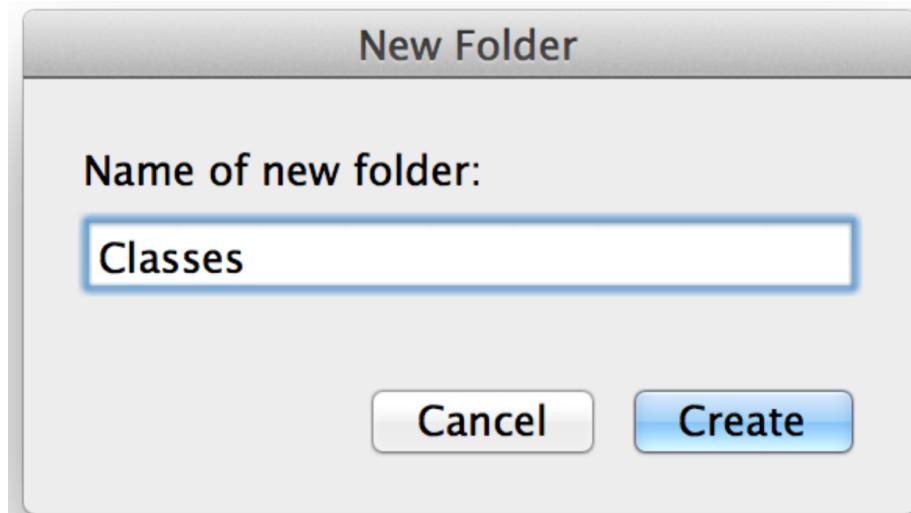
Hình 6.185 Next

Tạo 1 folder tên “**Classes**” để dễ quản lý bằng cách click “**New Folder**”



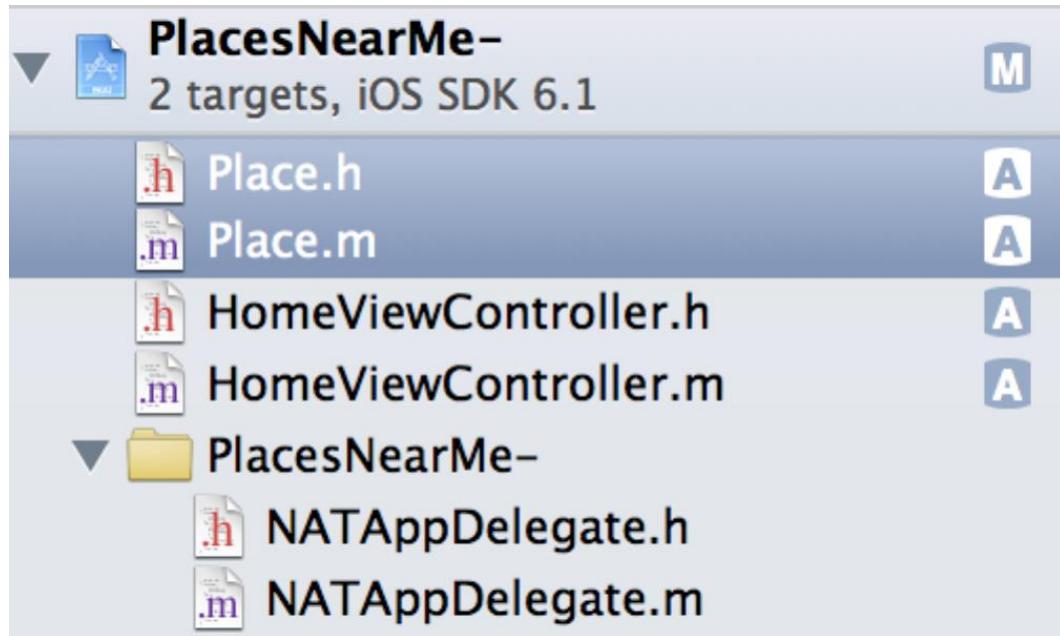
**Hình 6.186 Tạo New Folder để lưu class**

Nhập tên “**Classes**” → click “**Create**” → “**Create**”



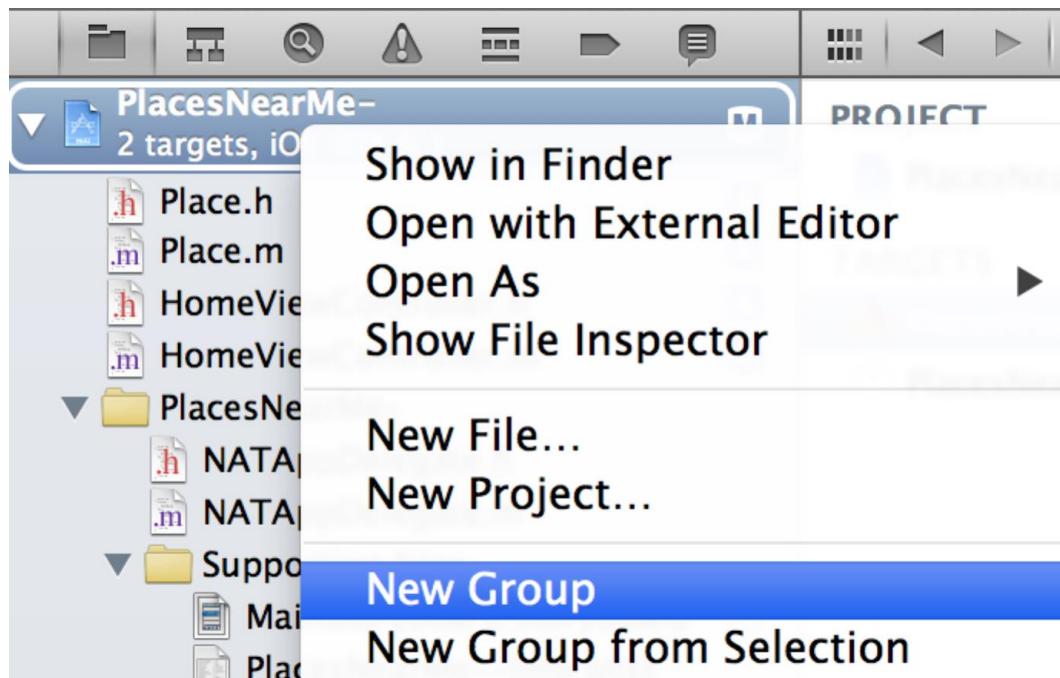
**Hình 6.187 Create**

Kết quả:



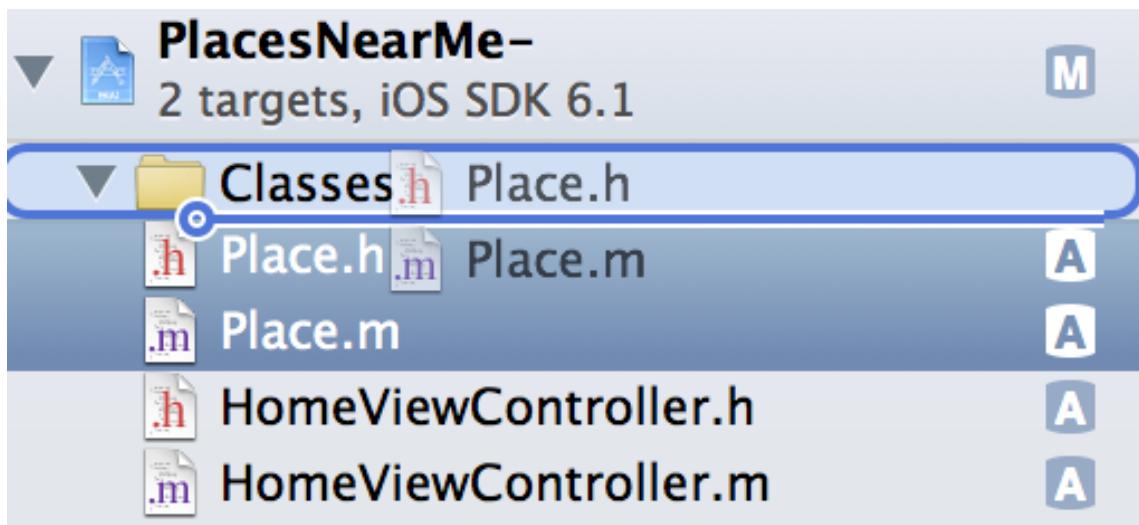
Hình 6.188 Kết quả

Tạo một Group tên “**Classes**” để tiện quản lý, phải chuột vào project → “**New Group**” → nhập “**Classes**”.



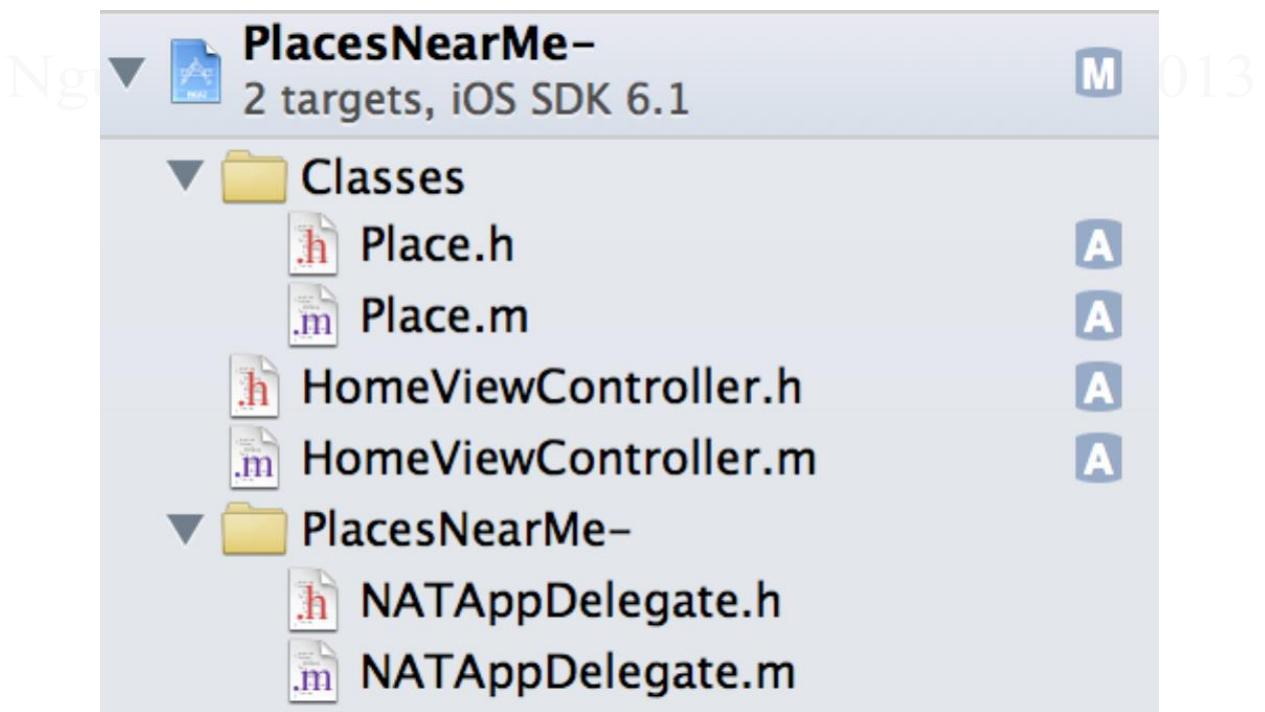
Hình 6.189 New Group

Kéo thả 2 file “Place.h” và “Place.m” vào Group “Classes”



Hình 6.190 Kéo thả 2 file vào Classes

Kết quả:



Hình 6.191 Kết quả

Mở “Place.h” để khai báo các thuộc tính đã liệt kê phía trên(Cách khai báo các thuộc tính này tương tự như **getter/ setter** của C# hoặc java).

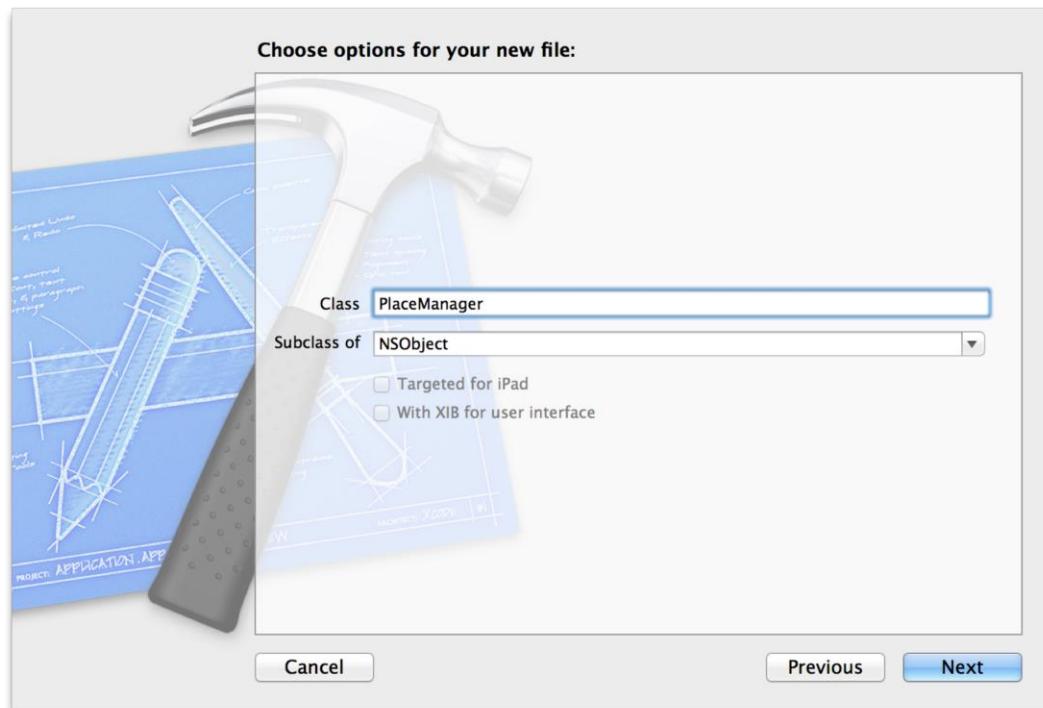
```
@interface Place : NSObject  
  
@property (nonatomic, strong) NSString *imageName;  
  
@property (nonatomic, strong) NSString *titleEn;  
  
@property (nonatomic, strong) NSString *titleVi;  
  
@property (nonatomic, strong) NSString *placeType;  
  
@property (nonatomic, strong) NSString *keyWord;  
  
@end
```

Tạo lớp “**PlaceManager**” để quản lý các đối tượng “**Place**”, lớp “**PlaceManager**” sẽ có 2 phương thức chính:

- **getArrPlaces**: phương thức này trả về một mảng gồm các “Place”
- **setPlace**: dùng để khởi tạo 1 đối tượng “Place”

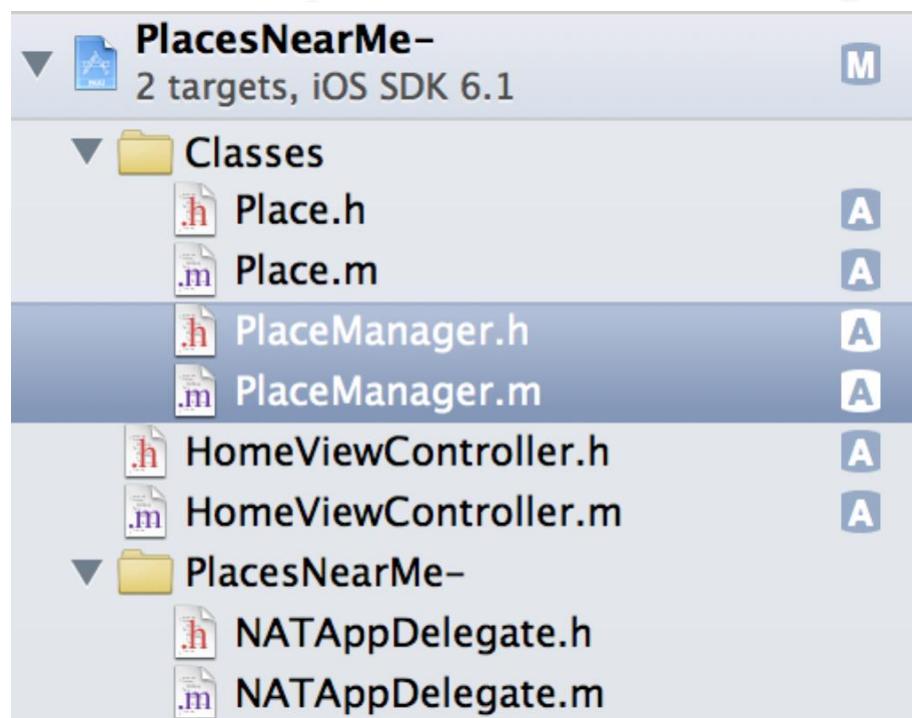
Tương tự như trên, tạo 1 class với :

Class: “**PlaceManager**” → Subclass of: “**NSObject**” → lưu trong “**Classes**”.



**Hình 6.192 Tạo mới class**

Kết quả: Nguyễn Anh Tiệp - Cao Thành Vàng © 2013



**Hình 6.193 Kết quả**

Mở file “**PlaceManager.h**” để khai báo phương thức “**getArrPlaces**”, việc khai báo này sẽ giúp chúng ta gọi trực tiếp phương thức “**getArrPlaces**” thông qua tên lớp “**PlaceManager**” và để làm được điều này chúng ta sẽ sử dụng dấu “+” thay cho dấu “-“ trước phương thức (tương tự phương thức static bên C# hoặc java).

```
@interface PlaceManager : NSObject  
+ (NSMutableArray *)getArrPlaces;  
@end
```

Mở file “**PlaceManager.m**” , bên dưới dưới **#import "PlaceManager.h"**, ta import thêm lớp “**Place.h**” để sử dụng được để sử dụng được các phương thức **getter/setter** của lớp này như sau:

```
#import "PlaceManager.h"  
#import "Place.h"
```

Thêm phương thức “**setPlace**” để tạo đối tượng “**Place**” như sau:

```
+ (Place *)setPlace:(NSString *)image  
    titleEn:(NSString *)titleEn  
    titleVi:(NSString *)titleVi  
    placeType:(NSString *)placeType  
    keyWord:(NSString *)keyWord {  
  
    Place *pt = [[Place alloc] init];  
    pt.imageName = image;  
    pt.titleEn = titleEn;  
    pt.titleVi = titleVi;  
    pt.placeType = placeType;  
    pt.keyWord = keyWord;  
  
    return pt;  
}
```

**Giải thích:** phương thức “**setPlace**” trả về 1 đối tượng “**Place**” với các thuộc tính: **image**, **titleEn**, **titleVi**, **placeType**, **keyWord**.

Cũng file “**PlaceManager.m**” hiện thực lại phương thức “**getArrPlaces**” đã khai báo ở file “**PlaceManager.h**” trước đó như sau:

```
+ (NSMutableArray *)getArrPlaces {  
    //Lưu ý: placeType & keyWord phải viết thường vì chúng là các từ khoá của Google (google
```

yêu cầu viết thường)

```
return [[NSMutableArray alloc] initWithObjects:  
    [self setPlace:@"atm.png" titleEn:@"ATM" titleVi:@"ATM" placeType:@"establishment"  
keyWord:@[@"atm"],  
    [self setPlace:@"bank.png" titleEn:@"Bank" titleVi:@"Ngân hàng"  
placeType:@"establishment" keyWord:@[@"bank"],  
    [self setPlace:@"school.png" titleEn:@"School" titleVi:@"Trường học"  
placeType:@"establishment" keyWord:@[@"school"],  
  
    [self setPlace:@"bar.png" titleEn:@"Bar" titleVi:@"Bar" placeType:@"establishment"  
keyWord:@[@"bar"],  
    [self setPlace:@"coffee_shops.png" titleEn:@"Cafe" titleVi:@"Cafe"  
placeType:@"establishment" keyWord:@[@"cafe"],  
    [self setPlace:@"karaoke.png" titleEn:@"Karaoke" titleVi:@"Karaoke"  
placeType:@"establishment" keyWord:@[@"karaoke"],  
  
    [self setPlace:@"bus_station.png" titleEn:@"Bus station" titleVi:@"Trạm xe buýt"  
placeType:@"bus_station" keyWord:@""],  
    [self setPlace:@"gas_station.png" titleEn:@"Gas station" titleVi:@"Trạm xăng"  
placeType:@"gas_station" keyWord:@[@"station"],  
    [self setPlace:@"supermarket.png" titleEn:@"Supermarket" titleVi:@"Siêu thị"  
placeType:@"establishment" keyWord:@[@"supermarket"],  
  
    [self setPlace:@"restaurant.png" titleEn:@"Restaurant" titleVi:@"Nhà hàng"  
placeType:@"establishment" keyWord:@[@"restaurant"],  
    [self setPlace:@"lodging.png" titleEn:@"Hotel" titleVi:@"Khách sạn"  
placeType:@"establishment" keyWord:@[@"hotel"],  
    [self setPlace:@"park.png" titleEn:@"Park" titleVi:@"Công viên"  
placeType:@"establishment" keyWord:@[@"park"],  
  
    [self setPlace:@"movie_theater.png" titleEn:@"Movie theater" titleVi:@"Rạp chiếu"  
placeType:@"establishment" keyWord:@[@"movie theater"],  
    [self setPlace:@"computer.png" titleEn:@"Computer store" titleVi:@"Máy tính"  
placeType:@"establishment" keyWord:@[@"computer"],  
    [self setPlace:@"post_office.png" titleEn:@"Post office" titleVi:@"Bưu điện"  
placeType:@"establishment" keyWord:@[@"post_office"],  
  
    [self setPlace:@"train_station.png" titleEn:@"Train station" titleVi:@"Ga xe lửa"  
placeType:@"train_station" keyWord:@""],  
    [self setPlace:@"airport.png" titleEn:@"Airport" titleVi:@"Sân bay" placeType:@"airport"  
keyWord:@""],  
    [self setPlace:@"bakery.png" titleEn:@"bakery" titleVi:@"Tiệm bánh"  
placeType:@"establishment" keyWord:@[@"bakery"],  
  
    [self setPlace:@"beauty_salon.png" titleEn:@"Beauty salon" titleVi:@"Làm đẹp"  
placeType:@"establishment" keyWord:@[@"beauty_salon"],  
    [self setPlace:@"spa.png" titleEn:@"Spa" titleVi:@"Spa" placeType:@"establishment"  
keyWord:@[@"spa"],  
    [self setPlace:@"hair.png" titleEn:@"Hair care" titleVi:@"Tiệm uốn tóc"  
placeType:@"establishment" keyWord:@[@"hair"],  
  
    [self setPlace:@"book_store.png" titleEn:@"Book Store" titleVi:@"Nhà sách"]
```

```

placeType:@ "establishment" keyWord:@ "book_store"],
    [self setPlace:@ "cemetery.png" titleEn:@ "Cemetery" titleVi:@ "Nghĩa trang"
placeType:@ "establishment" keyWord:@ "cemetery"],
    [self setPlace:@ "church.png" titleEn:@ "Church" titleVi:@ "Nhà thờ"
placeType:@ "establishment" keyWord:@ "church"],

    [self setPlace:@ "clothing_store.png" titleEn:@ "Clothing store" titleVi:@ "Quần áo"
placeType:@ "establishment" keyWord:@ "clothing_store"],
    [self setPlace:@ "shoe_store.png" titleEn:@ "Shoe store" titleVi:@ "Dày dép"
placeType:@ "establishment" keyWord:@ "shoe_store"],
    [self setPlace:@ "convenience_store.png" titleEn:@ "Convenience store" titleVi:@ "Tiệm
tạp hoá" placeType:@ "establishment" keyWord:@ "convenience_store"],

    [self setPlace:@ "electronics_store.png" titleEn:@ "Electronics store" titleVi:@ "Điện tử"
placeType:@ "electronics_store" keyWord:@ ""],
    [self setPlace:@ "furniture_store.png" titleEn:@ "Furniture store" titleVi:@ "Nội thất"
placeType:@ "establishment" keyWord:@ "furniture_store"],
    [self setPlace:@ "sport.png" titleEn:@ "Sport store" titleVi:@ "Thể thao"
placeType:@ "establishment" keyWord:@ "sport"],

    [self setPlace:@ "hospital.png" titleEn:@ "Hospital" titleVi:@ "Bệnh viện"
placeType:@ "establishment" keyWord:@ "hospital"],
    [self setPlace:@ "pharmacy.png" titleEn:@ "Pharmacy" titleVi:@ "Tiệm thuốc"
placeType:@ "establishment" keyWord:@ "pharmacy"],
    [self setPlace:@ "library.png" titleEn:@ "Library" titleVi:@ "Thư viện"
placeType:@ "establishment" keyWord:@ "library"],

    [self setPlace:@ "museum.png" titleEn:@ "Museum" titleVi:@ "Bảo tàng"
placeType:@ "establishment" keyWord:@ "museum"],
    [self setPlace:@ "parking.png" titleEn:@ "Parking" titleVi:@ "Bãi đỗ xe"
placeType:@ "establishment" keyWord:@ "parking"],
    [self setPlace:@ "police.png" titleEn:@ "Police" titleVi:@ "Cảnh sát" placeType:@ "police"
keyWord:@ ""],

    [self setPlace:@ "real_estate_agency.png" titleEn:@ "Real estate agency" titleVi:@ "Bất
động sản" placeType:@ "establishment" keyWord:@ "real_estate_agency"],
    [self setPlace:@ "stadium.png" titleEn:@ "Stadium" titleVi:@ "Sân vận động"
placeType:@ "establishment" keyWord:@ "stadium"],
    [self setPlace:@ "travel_agency.png" titleEn:@ "Travel agency" titleVi:@ "Vé máy bay"
placeType:@ "establishment" keyWord:@ "travel agency"],

    [self setPlace:@ "veterinary_care.png" titleEn:@ "Veterinary care" titleVi:@ "Thú y"
placeType:@ "establishment" keyWord:@ "veterinary care"],
    [self setPlace:@ "zoo.png" titleEn:@ "Zoo" titleVi:@ "Sở thú" placeType:@ "zoo"
keyWord:@ ""],
    [self setPlace:@ "car_dealer.png" titleEn:@ "Car dealer" titleVi:@ "Đại lý xe hơi"
placeType:@ "car_dealer" keyWord:@ ""],
    nil];
}

```

### Giải thích:

- Phương thức “**getArrPlaces**”: trả về 1 mảng các đối tượng “**Place**”
  - **[NSMutableArray alloc] initWithObjects**: sẽ giúp chúng ta khởi tạo 1 mảng các đối tượng.
  - Để gọi phương thức “**setPlace**” ta dùng “**self <ten\_phuong\_thuc>**”.
- VD:** `[self setPlace:<tham_so_1> <tham_so_2> <tham_so_n>];`

Gọi thử phương thức “**getArrPlaces**” vừa hiện thực để đảm bảo chúng ta đã có 1 mảng các đối tượng “**Place**”.

→ Mở file “**HomeViewController.m**” → tìm và xoá các dòng sau để loại bỏ cảnh báo:

```
#pragma mark - Table view data source
#warning Potentially incomplete method implementation.
#warning Incomplete method implementation.
```

**Kết quả:**

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 0;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return 0;
}
```

**Hình 6.194 Kết quả**

Khai báo 1 mảng “**arrPlaces**” bên dưới “**@implementation HomeViewController**”:

```
@implementation HomeViewController
NSMutableArray *arrPlaces;
```

Bên dưới `#import "HomeViewController.h"` import thêm “`Place.h`” và “`PlaceManager.h`”.

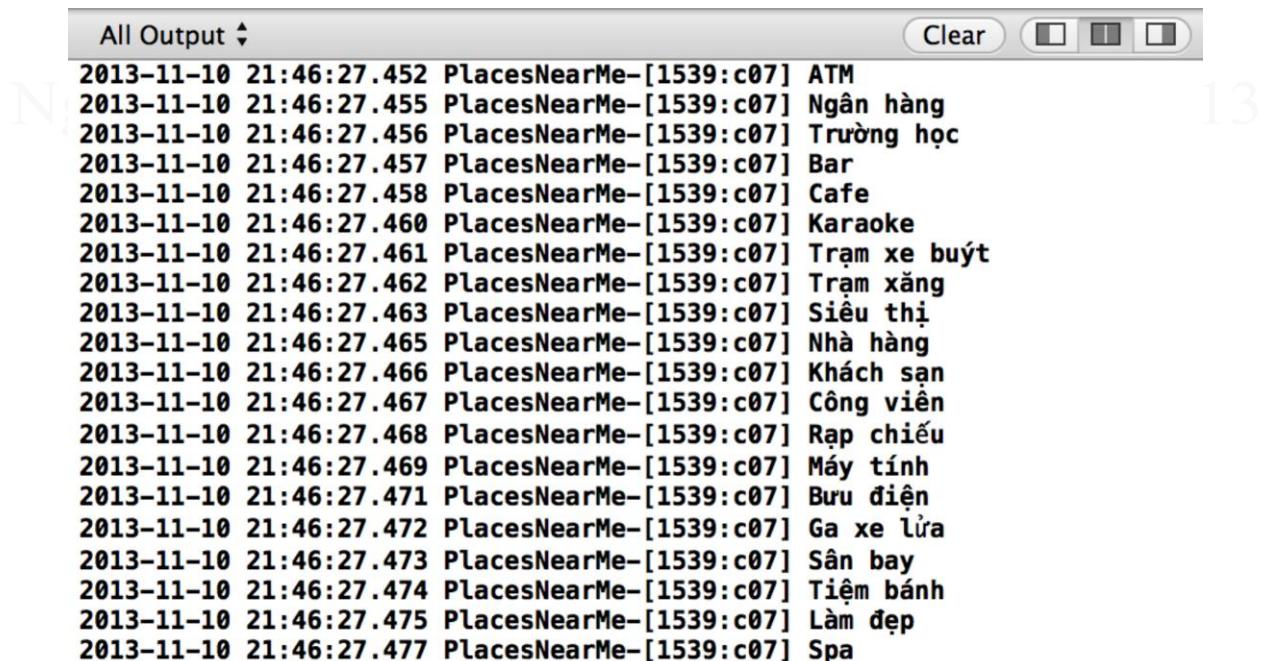
```
#import "HomeViewController.h"
#import "Place.h"
#import "PlaceManager.h"
```

Tại hàm “`viewDidLoad`” ta gọi và in thử phương thức “`getArrPlaces`” của lớp “`PlaceManager`”:

```
//Gọi phương thức getArrPlaces và gán kết quả vào mảng arrPlaces
arrPlaces = [PlaceManager getArrPlaces];

//In thử mảng arrPlaces
for (Place *p in arrPlaces) {
    NSLog(@"%@", p.titleVi);
}
```

Kết quả:



The screenshot shows the Xcode Output window titled "All Output". It contains a list of log messages from the console. Each message consists of a timestamp, a log level (2013-11-10 21:46:27.452), a file name (PlacesNearMe-[1539:c07]), a method name (ATM, Ngân hàng, Trường học, Bar, Cafe, Karaoke, Trạm xe buýt, Trạm xăng, Siêu thị, Nhà hàng, Khách sạn, Công viên, Rạp chiếu, Máy tính, Büro điện, Ga xe lửa, Sân bay, Tiệm bánh, Làm đẹp, Spa), and a place name in Vietnamese. The log entries are as follows:

```
2013-11-10 21:46:27.452 PlacesNearMe-[1539:c07] ATM
2013-11-10 21:46:27.455 PlacesNearMe-[1539:c07] Ngân hàng
2013-11-10 21:46:27.456 PlacesNearMe-[1539:c07] Trường học
2013-11-10 21:46:27.457 PlacesNearMe-[1539:c07] Bar
2013-11-10 21:46:27.458 PlacesNearMe-[1539:c07] Cafe
2013-11-10 21:46:27.460 PlacesNearMe-[1539:c07] Karaoke
2013-11-10 21:46:27.461 PlacesNearMe-[1539:c07] Trạm xe buýt
2013-11-10 21:46:27.462 PlacesNearMe-[1539:c07] Trạm xăng
2013-11-10 21:46:27.463 PlacesNearMe-[1539:c07] Siêu thị
2013-11-10 21:46:27.465 PlacesNearMe-[1539:c07] Nhà hàng
2013-11-10 21:46:27.466 PlacesNearMe-[1539:c07] Khách sạn
2013-11-10 21:46:27.467 PlacesNearMe-[1539:c07] Công viên
2013-11-10 21:46:27.468 PlacesNearMe-[1539:c07] Rạp chiếu
2013-11-10 21:46:27.469 PlacesNearMe-[1539:c07] Máy tính
2013-11-10 21:46:27.471 PlacesNearMe-[1539:c07] Büro điện
2013-11-10 21:46:27.472 PlacesNearMe-[1539:c07] Ga xe lửa
2013-11-10 21:46:27.473 PlacesNearMe-[1539:c07] Sân bay
2013-11-10 21:46:27.474 PlacesNearMe-[1539:c07] Tiệm bánh
2013-11-10 21:46:27.475 PlacesNearMe-[1539:c07] Làm đẹp
2013-11-10 21:46:27.477 PlacesNearMe-[1539:c07] Spa
```

Hình 6.195 Kết quả

Sau khi in thử mà ra được kết quả như trên, đến đây tạm ôn, xoá hàm in thử vì chúng ta không cần nữa, kết quả hàm “`viewDidLoad`”:

```
- (void)viewDidLoad
```

```

{
    [super viewDidLoad];
    //Gọi phương thức getArrPlaces và gán kết quả vào mảng arrPlaces
    arrPlaces = [PlaceManager getArrPlaces];
}

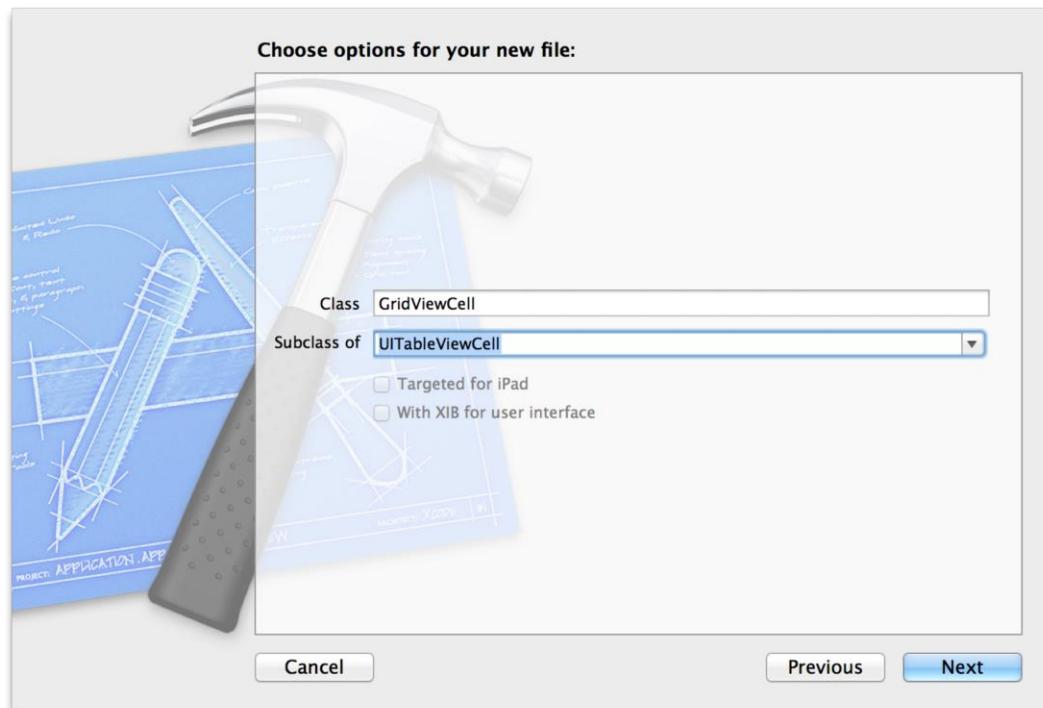
```

Hiển thị dữ liệu lên giao diện, kết quả đạt được như sau:



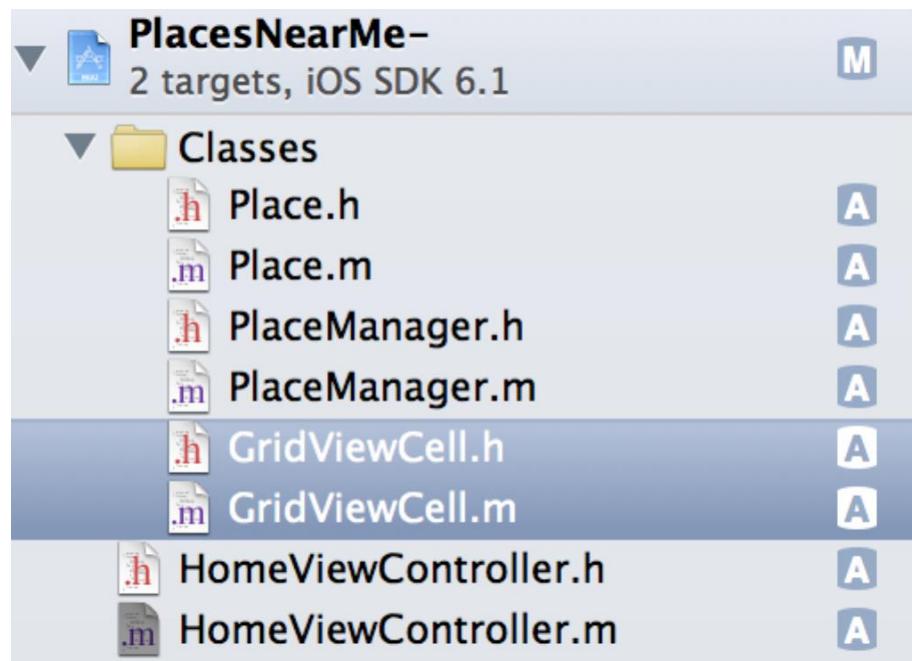
Hình 6.196 Hiển thị dữ liệu

Để hiển thị hình ảnh dạng lưới như trên, chúng ta cần tạo 1 lớp “**GridViewCell**”, “**Cmd + N**” để tạo 1 đối tượng → Class: **GridViewCell** → Subclass of: **UITableViewCell** → **Next**.



Hình 6.197 Tạo mới một class

Kết quả: Nguyễn Anh Tiệp - Cao Thành Vàng © 2013



Hình 6.198 Class mới tạo

Mở file “GridViewCell.h” thêm các thuộc tính như sau:

```

@interface GridViewCell : UITableViewCell

@property (nonatomic, strong) UIButton *column1;
@property (nonatomic, strong) UIButton *column2;
@property (nonatomic, strong) UIButton *column3;

@end

```

Bên trên import <UIKit/UIKit.h> định nghĩa các thông số sau cho Cell đồng thời import QuartzCore.h để sử dụng các hàm cấu hình border trong lớp này:

```

#define CELL_WIDTH 92 //Độ rộng của ô
#define CELL_HEIGHT 80 //Chiều cao của ô
#define CELL_MARGIN_LEFT 20 //Cách lề trái
#define CELL_MARGIN_TOP 1 //Cách lề trên

#import <UIKit/UIKit.h>
#import <QuartzCore/QuartzCore.h>

```

Mở file “GridViewCell.m” bên dưới @implement GridViewCell, Synthesize để sử dụng được các columns đã khai báo bên “GridViewCell.h”

```

@implementation GridViewCell

@synthesize column1, column2, column3;

```

Tạo hàm “configColumn” để cấu hình giao diện cho column.

**VD:** Vị trí text trong column, font chữ, kích thước chữ, border,...

```

- (void)configColumn:(UIButton *)cln {
    //Đưa text xuống vị trí cuối cùng
    cln.contentVerticalAlignment = UIControlContentVerticalAlignmentBottom;

    //Thiết lập màu chữ font và kích thước
    [cln setTitleColor:[UIColor blackColor] forState:UIControlStateNormal];
    [cln.titleLabel setFont:[UIFont fontWithName:@"Arial" size:14.0]];
    [cln.titleLabel setAdjustsFontSizeToFitWidth:YES];

    //Tạo border
    [cln.layer setBorderColor:[[UIColor colorWithRed:0.8 green:0.8 blue:0.8 alpha:1] CGColor]];
    cln.layer.borderWidth = 1;

    //Thêm cell vào view
    [self addSubview:cln];
}

```

Tại hàm “**initWithStyle**” (đây có thể xem như 1 Constructor bên C# hoặc java), khởi tạo dữ liệu cho từng cột như sau:

```
- (id)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString *)reuseIdentifier
{
    self = [super initWithStyle:style reuseIdentifier:reuseIdentifier];

    if (self) {
        column1 = [[UIButton alloc] initWithFrame:
                    CGRectMake(CELL_MARGIN_LEFT, CELL_MARGIN_TOP, CELL_WIDTH,
                               CELL_HEIGHT)];
        [self configColumn:column1];

        column2 = [[UIButton alloc] initWithFrame:
                    CGRectMake(CELL_WIDTH + CELL_MARGIN_LEFT + 2, CELL_MARGIN_TOP,
                               CELL_WIDTH, CELL_HEIGHT)];
        [self configColumn:column2];

        column3 = [[UIButton alloc] initWithFrame:
                    CGRectMake(CELL_WIDTH + CELL_WIDTH + CELL_MARGIN_LEFT + 4,
                               CELL_MARGIN_TOP, CELL_WIDTH, CELL_HEIGHT)];
        [self configColumn:column3];
    }

    return self;
}
```

### Giải thích:

- Mỗi Column của chúng ta sẽ là 1 button do đó chúng ta cần sử dụng **[[UIButton alloc] initWithFrame]** để khởi tạo 1 button.
- **CGRectMake**: hàm này sẽ xác định vị trí, kích thước của một button gồm các tham số (lề trái, lề trên, độ rộng, chiều cao).

Mở file “**HomeViewController.m**”, hàm “**numberOfSectionsInTableView**” sửa 0 thành 1:

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 0;
}
```

→ Hàm “**numberOfRowsInSection**” sửa lại như sau:

```

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    return arrPlaces.count / 3;
}

```

→ Thêm hàm **configColumn** để thiết lập hình nền, tiêu đề, hành động cho mỗi column (button)

```

- (void)configColumn:(UIButton *)column index:(int)index {
    NSString *image = [[arrPlaces objectAtIndex:index] imageName];
    NSString *title = [[arrPlaces objectAtIndex:index] titleVi];
    //Gán background cho column (button)
    [column setBackgroundColor:[UIColor colorWithPatternImage:[UIImage imageNamed:image]]];
    //Gán tiêu đề cho column
    [column setTitle:title forState:UIControlStateNormal];
    //Thêm hành động cho column
    [column addTarget:self action:@selector(clickButtons:) forControlEvents:UIControlEventTouchUpInside];
}

```

→ import thêm "**GridViewCell.h**"

→ Hàm “**cellForRowIndexPath**” sửa lại như sau:

```

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";
    GridViewCell *cell = (GridViewCell*)[tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    cell = [[GridViewCell alloc] initWithStyle:UITableViewCellStyleDefault
        reuseIdentifier:CellIdentifier];
    cell.selectionStyle = UITableViewCellAccessoryNone;
    //Dismiss line of cells
    self.tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
    // Configure the cell...
    int row = indexPath.row;
    int firstColumn = row * 3;
    [self configColumn:cell.column1 index:firstColumn];
    [self configColumn:cell.column2 index:firstColumn + 1];
    [self configColumn:cell.column3 index:firstColumn + 2];
    return cell;
}

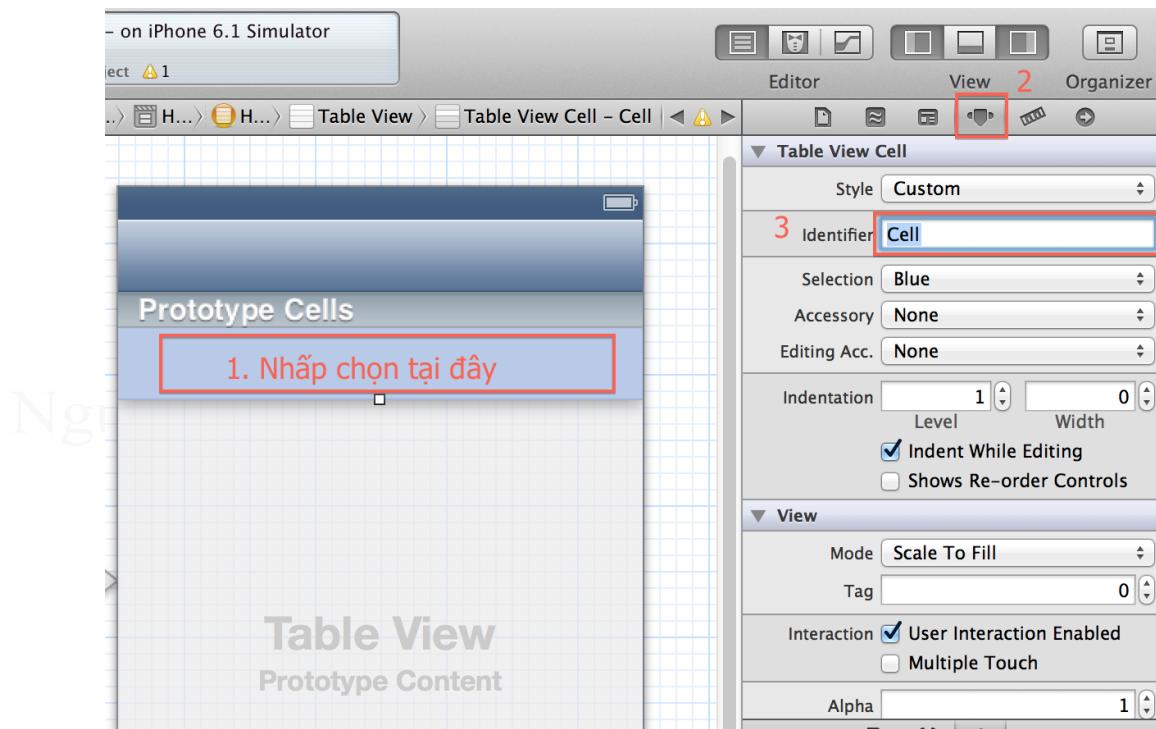
```

```
}
```

→ Thêm hàm “**heightForRowAtIndexPath**” để cấu hình chiều cao cho Row

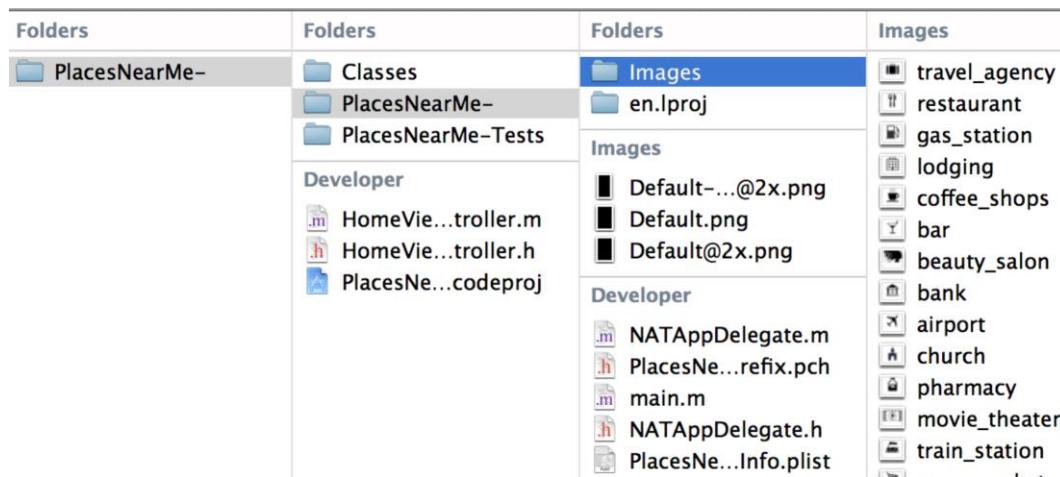
```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath {
    return 82;
}
```

→ Qua “**MainStoryboard.storyboard**” → cấu hình cho cell của **TableViewController** như sau:



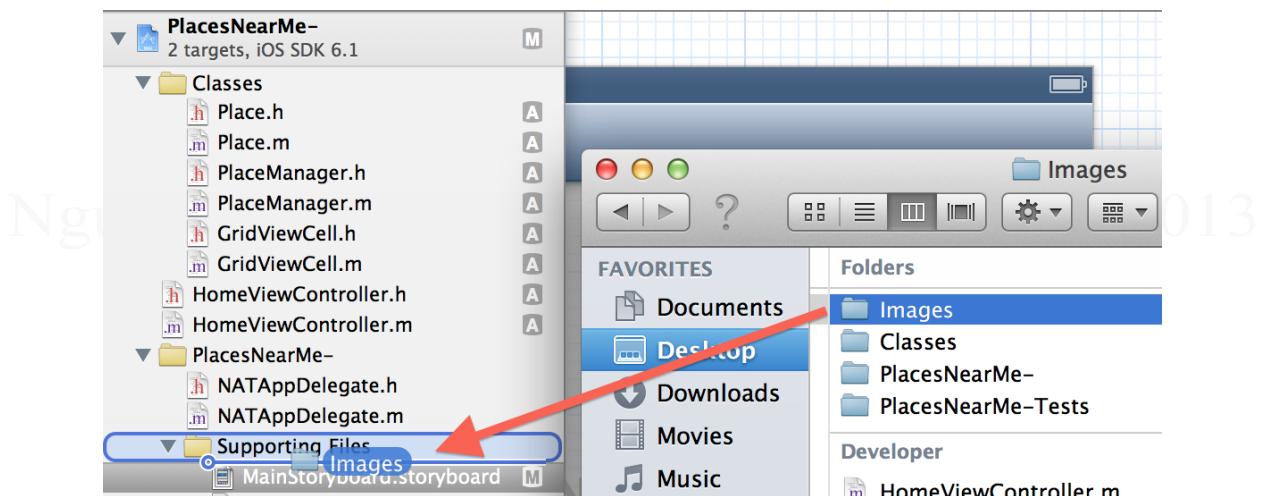
**Hình 6.199 Cấu hình cho Cell**

→ Chép thư mục **Images** trong source và bỏ vào bên trong project như sau:



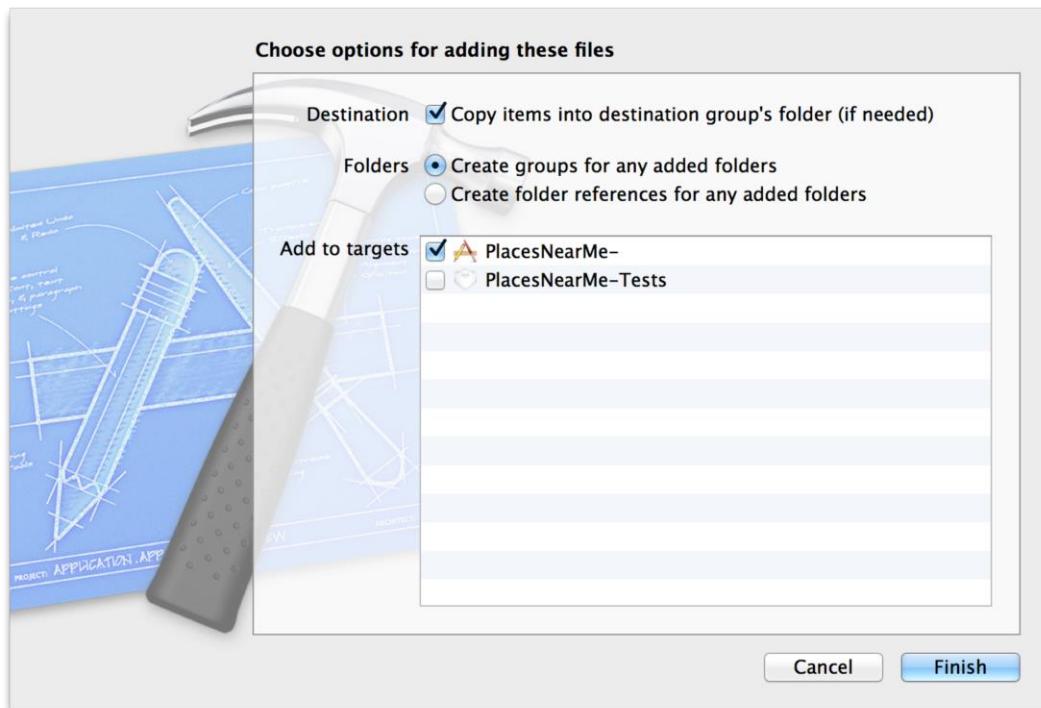
**Hình 6.200 Thu mục tải về**

→ Kéo thả thư mục images bên trong project vào XCode:



**Hình 6.201 Kéo thả thư mục hình ảnh vào project**

→ Check **Copy items...** → Click **Finish**



**Hình 6.202 Chọn Copy items**

→ Chạy thử (**cmd + r**) → kết quả:



**Hình 6.203 Giao diện kết quả**

→ Khi click vào các hình (button) sẽ bị lỗi do chúng ta chưa có hàm “**clickButton**” đã add trước đó (xem hàm configColumn), thêm hàm “**selectedIndex** “ và “**clickButton**” như sau:

```
- (int)selectedIndex:(id)sender {
    NSString *selectedTitle = [sender currentTitle];
    int i=0;

    for (Place *p in arrPlaces) {
        if ([selectedTitle isEqualToString:p.titleEn] || [selectedTitle isEqualToString:p.titleVi]) {
            return i;
        }
        i++;
    }
    return i;
}
```

```

- (void)clickButtons:(id)sender {
    int selected = [self selectedIndex:sender];
    NSLog(@"%@",selected, [arrPlaces[selected] titleVi]);
}

```

**Giải thích:** hàm **selectedIndex** sẽ giúp chúng ta xác định vị trí (index) mà người dùng click vào hình ảnh

→ Chạy ứng dụng → kết quả sau khi click vào các hình

```

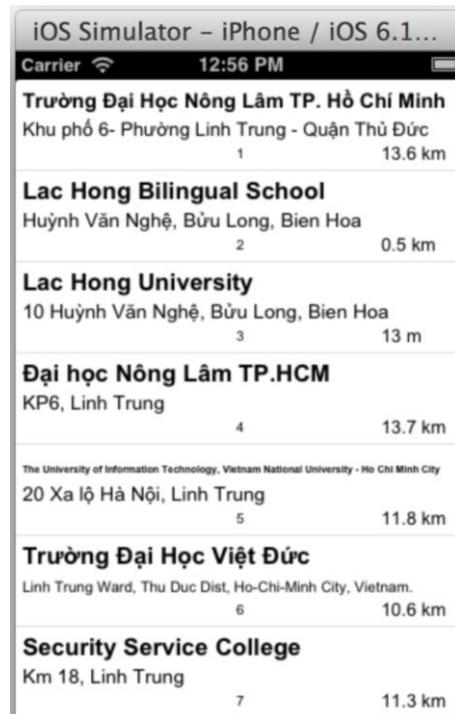
2013-11-10 23:55:13.523 PlacesNearMe-[7133:c07] 0.ATM
2013-11-10 23:55:14.505 PlacesNearMe-[7133:c07] 1.Ngân hàng
2013-11-10 23:55:15.851 PlacesNearMe-[7133:c07] 2.Trường học
2013-11-10 23:55:17.505 PlacesNearMe-[7133:c07] 5.Karaoke
2013-11-10 23:55:19.055 PlacesNearMe-[7133:c07] 4.Cafe
2013-11-10 23:55:20.068 PlacesNearMe-[7133:c07] 3.Bar

```

### Hình 6.204 Kết quả

**Bước 5:** Xây dựng View Controller “**FindingPlaceViewController**”

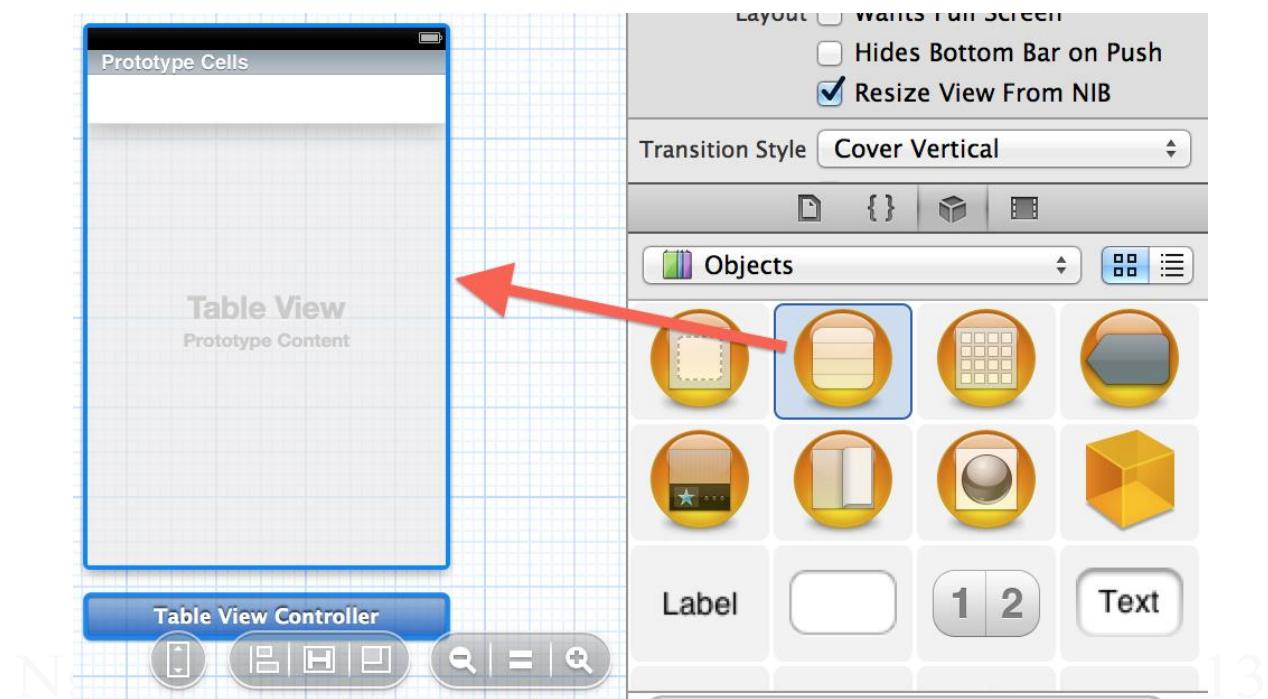
“**FindingPlaceViewController**” sẽ hiển thị kết quả tìm kiếm được từ Google như sau:



Hình 6.205 Giao diện FindingPlace

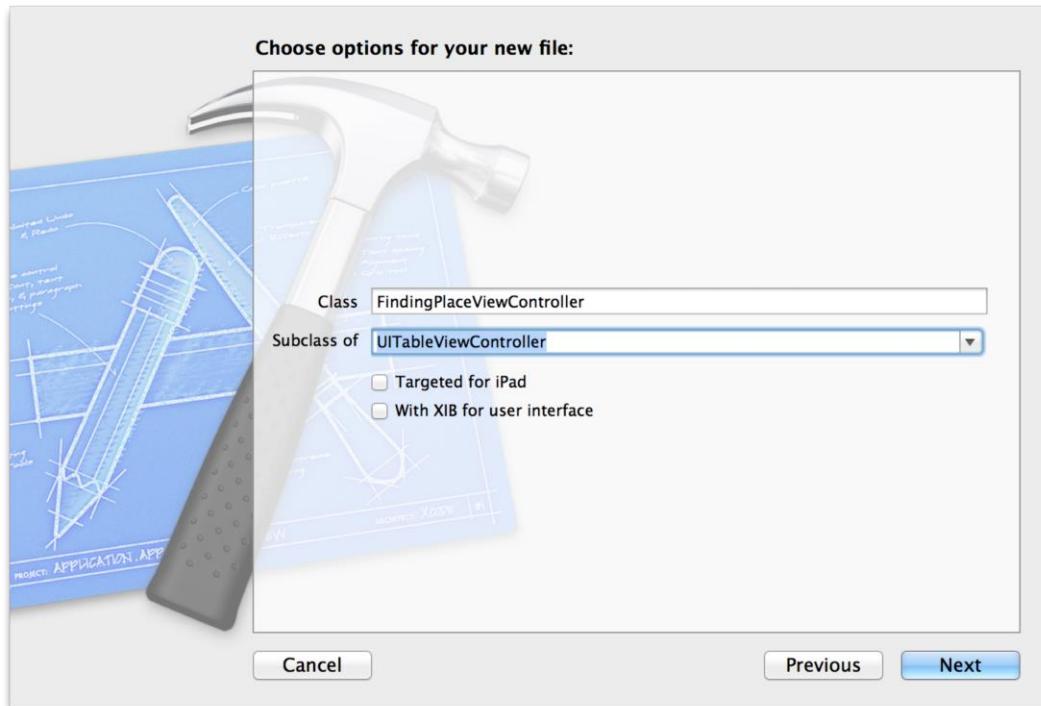
Tạo thêm 1 “**TableViewController**”, 1 lớp “**FindingPlaceViewController**” và đặt tên cho “**TableViewCell**” là “**Cell**”.

→ Kéo thả thêm 1 “**TableViewController**” vào giao diện thiết kế



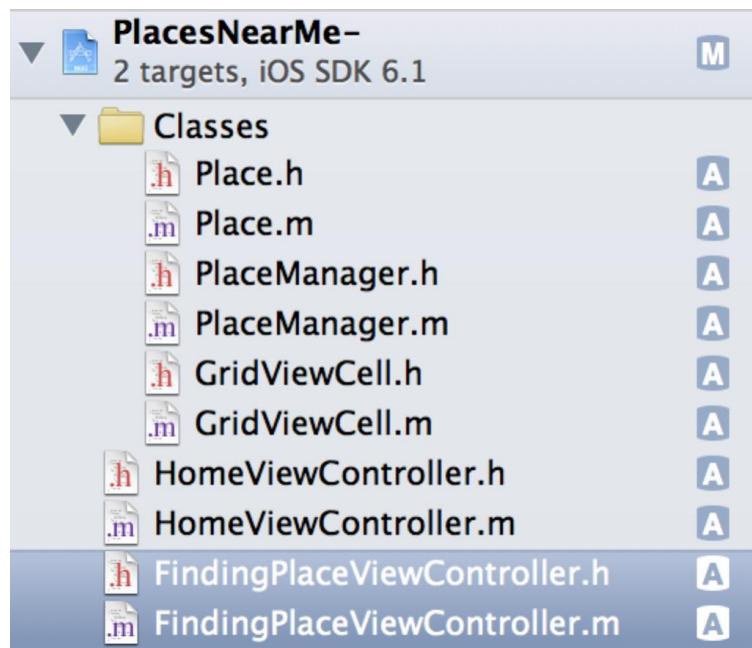
Hình 6.206 Tạo mới TableView

→ **Cmd + N** để tạo 1 lớp với Class: **FindingPlaceViewController** → Subclass of: **UITableViewController**.



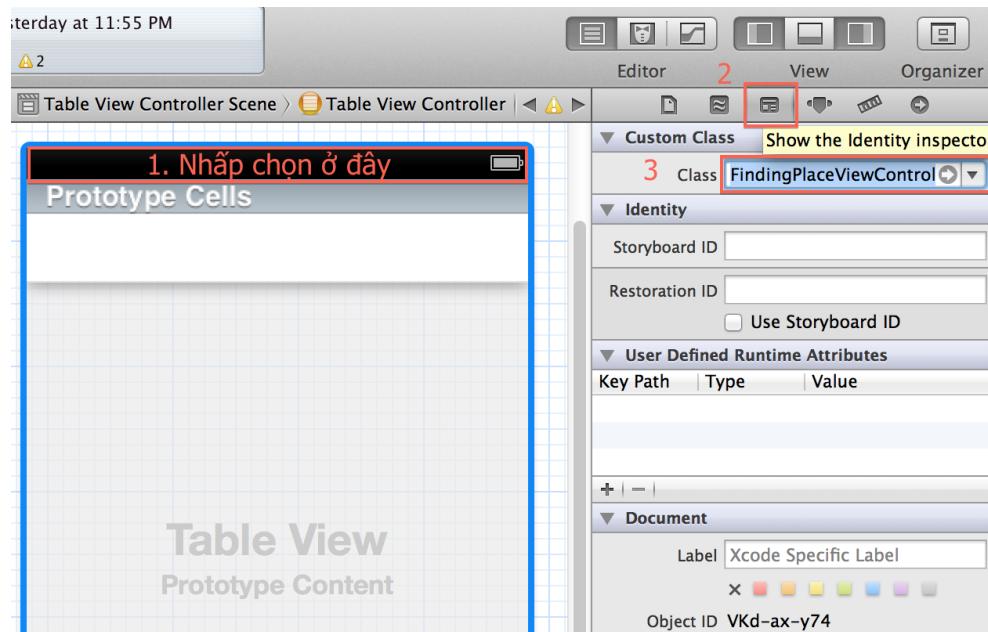
**Hình 6.207 Tạo mới Class**

→ Kết quả  
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013



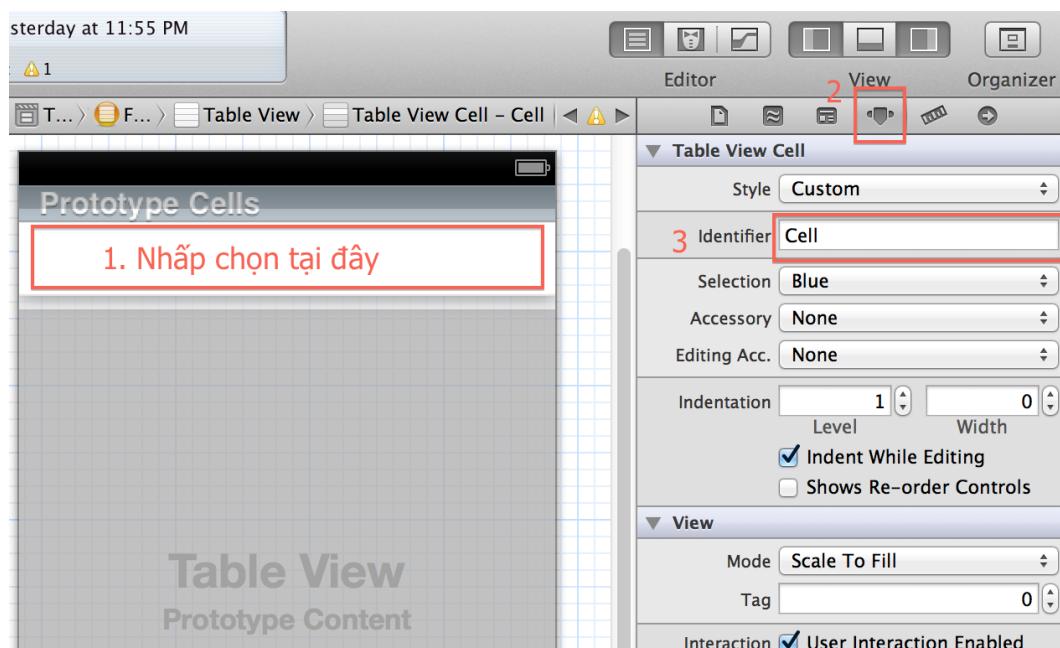
**Hình 6.208 Kết quả tạo class mới**

- Add lớp vừa tạo “**FindingPlaceViewController**” vào “**TableViewController**”
- Chọn “**TableViewController**” → “**Identity inspector**” → trong Class chọn hoặc nhập “**FindingPlaceViewController**”



Hình 6.209 Thêm class mới vào Table View Controller

- Đặt tên “Cell” cho “TableViewController”



Hình 6.210 Đặt tên Cell

Vào <https://code.google.com/apis/console/> để đăng ký và sử dụng dịch vụ **Places API** của Google:

The screenshot shows the Google Cloud Console interface for a project named "LHUPlaces". On the left, there's a sidebar with links: Overview, APIs & auth (with APIs selected), Registered apps (highlighted with a red box and labeled 1), Consent screen, Notification endpoints. The main area has a table with columns "NAME" and "STATUS". It lists five APIs: Contacts API (ON), Google Maps SDK for iOS (ON), Places API (ON, highlighted with a red box and labeled 2), Ad Exchange Buyer API (OFF), and Ad Exchange Seller API (OFF). The "Places API" row is highlighted with a yellow background.

NAME	STATUS
Contacts API	ON
Google Maps SDK for iOS	ON
Places API	ON
Ad Exchange Buyer API	OFF
Ad Exchange Seller API	OFF

**Hình 6.211 Đăng ký dịch vụ Google**

→ Vào “Registered apps” → “BrowserKey...” để lấy key:

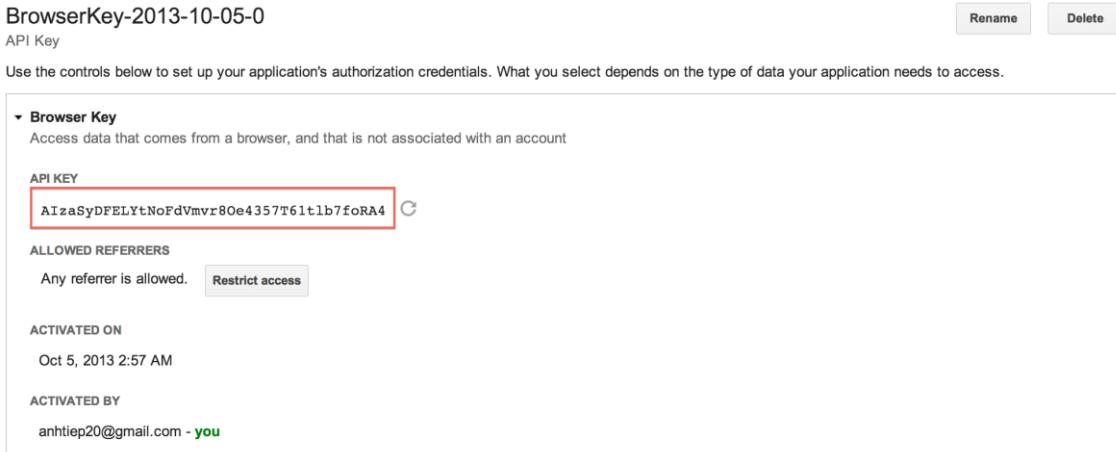
The screenshot shows the Google Cloud Console interface for a user named "NguyenAnhTiep". On the left, there's a sidebar with links: Overview, APIs & auth (with APIs selected), Registered apps (highlighted with a red box and labeled 1), Consent screen, Notification endpoints. The main area has a "REGISTER APP" button and a form for entering app details. The "NAME" field contains "Service Account-project". In the "REGISTER APP" section, the "NAME" field is labeled 2 and contains "BrowserKey-2013-10-05-0".

REGISTER APP

NAME	Service Account-project
2	BrowserKey-2013-10-05-0

**Hình 6.212 Lấy key API**

→ Key



### Hình 6.213 Key Google API

Request thử HTTP URL của Google, tham khảo tại [ĐÂY](#).

Google HTTP URL:

```
https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=<vĩ_độ>,<kinh_độ>&radius=<khoảng_cách_tìm>&types=<loại_nơi_tìm>&keyword=<API_KEY>&sensor=true&key=<tù_khoá_tìm>
```

VD:

[https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=10.970396,106.915351&radius=400&types=establishment%7Cestablishment&sensor=true&key=AIzaSyBKBekY4nxGHKh6wGqCtNGtMJRbl7FmTKM&keyword=school.](https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=10.970396,106.915351&radius=400&types=establishment%7Cestablishment&sensor=true&key=AIzaSyBKBekY4nxGHKh6wGqCtNGtMJRbl7FmTKM&keyword=school)

→ Kết quả Google trả về tài liệu dạng json:

```
{
  "debug_info" : [],
  "html_attributions" : [ "Listings by \u003ca href=\"http://www.diadiem.com/\u003eDiadiem.com\u003c/a\u003e\" " ],
  "results" : [
    {
      "geometry" : {
        "location" : {
          "lat" : 10.969931,
          "lng" : 106.91096
        }
      },
      "icon" : "http://maps.gstatic.com/mapfiles/place_api/icons/generic_business-71.png",
      "id" : "fb2ae0d8b67322c5f7ce9fdb555261837a3d69",
      "name" : "Trường Tiểu học Chu Văn An",
      "reference" : "CpQBhAAAAPN4g1z-89QluxJCBaCE4ZZcE2Hu1H1352wAInTiRFi_n6xIBqCPYEuZ3-ElK6s8c-3-gBVjifFN4amYpPwhHxxz1HpLcsVJ77gcV2Mug3PlKgcVsag_VTIqaGU7jyvanUwOx5KHSyCAjk25BRGfw16qQKS4-RiM2hiV3tJI-CMJQBrdjME1YKQEzcpHfsqhIQTdzz9VJO-Jvt3OTWYduQxoUccLBkxecDpvgezriB8gaw-0mmRIs",
      "types" : [ "establishment" ],
      "vicinity" : "Tân Hòa, Biên Hòa"
    }
  ],
  "status" : "OK"
}
```

### Hình 6.214 Kết quả Google trả về dạng Json

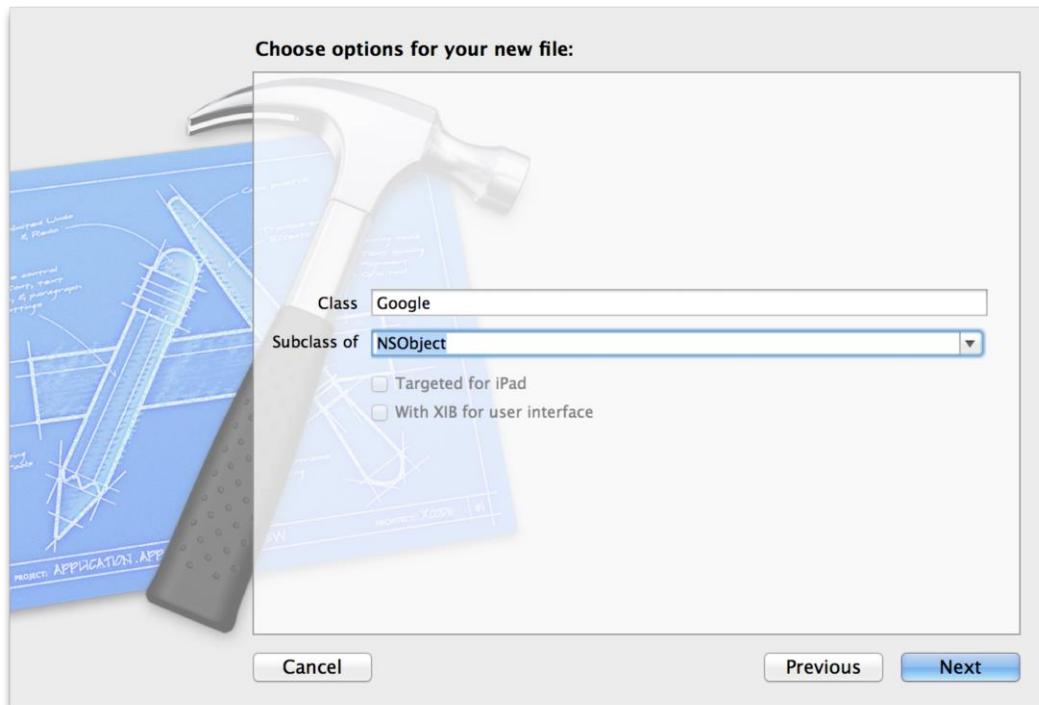
## Giải thích:

- **location**: bắt đầu tìm kiếm tại vị trí nào. VD: 10.953212,106.802378
- **Radius**: tìm kiếm trong phạm vi khoảng cách là bao nhiêu (meters). VD: 400 met
- **Types**: loại nơi cần tìm, những loại nơi này do Google định nghĩa. VD: establishment, school, hospital, atm,... Tham khảo tại [https://developers.google.com/places/documentation/supported\\_types](https://developers.google.com/places/documentation/supported_types).
- **Keyword**: từ khoá cần tìm, cho chúng ta tự định nghĩa. VD: school, university,...
- **Key**: API Key đã đăng ký ở trên.

**VD:** AIzaSyBKBekY4nxGHKh6wGqCtNGtMJRbl7FmTKM

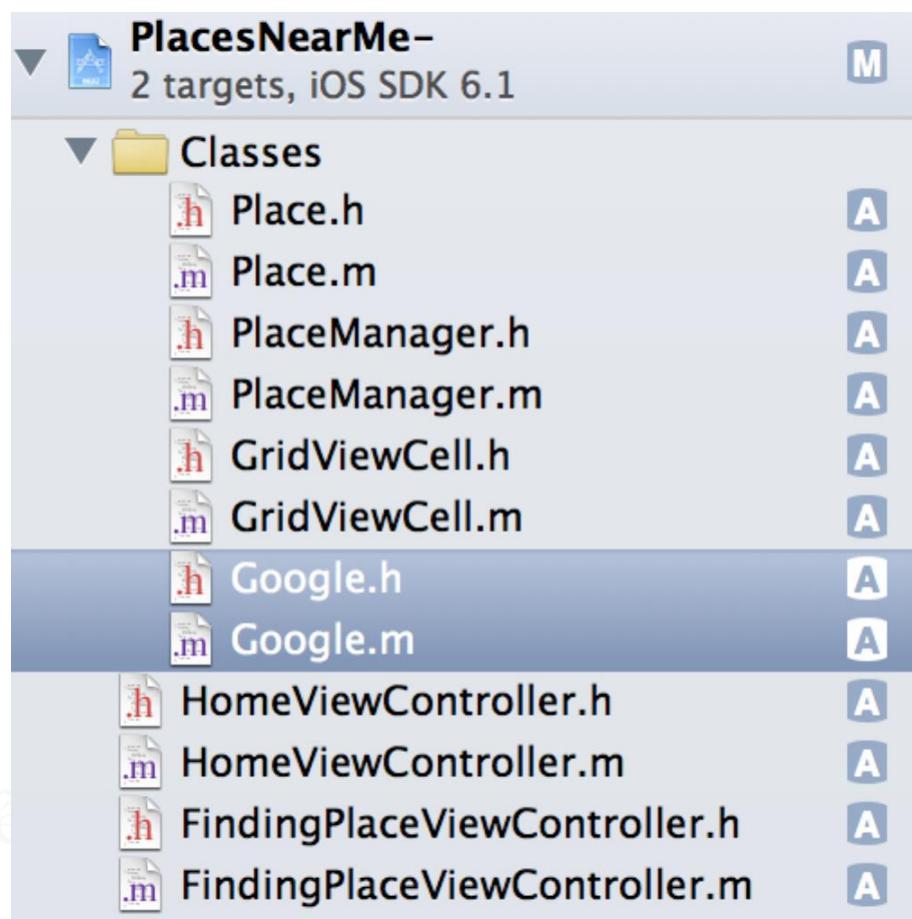
→ Tạo lớp “**Google**” để request các HTTP URL của Google. Lớp “**Google**” sẽ có các phương thức trả về tài liệu dạng json.

→ Cmd + N để tạo lớp mới → Class: **Google** → Subclass of: **NSObject** → Next



Hình 6.215 Tạo lớp Google

→ Kết quả:



Hình 6.216 Kết quả

→ Mở “Google.h” để khai báo các phương thức sau:

```
//Định nghĩa 1 API Key
#define GOOGLE_API_KEY @"AlzaSyBKBekY4nxGHKh6wGqCtNGtMJRbl7FmTKM"

#import <Foundation/Foundation.h>
//Thêm thư viện <CoreLocation/CoreLocation.h> để sử dụng CLLocationCoordinate2D
//(đây là một kiểu dữ liệu strut gồm vĩ độ (latitude) và kinh độ (longitude))
#import <CoreLocation/CoreLocation.h>

@interface Google : NSObject

//phương thức searchPlaces tìm các địa điểm xung quanh
+ (NSDictionary *)searchPlaces:(CLLocationCoordinate2D )coordinate
    radius:(NSString *)radius
    placeType:(NSString *)placeType
    keyword:(NSString *)keyword;
```

```

//khi có địa điểm rồi, chúng ta sẽ cần tìm khoảng cách từ vị trí hiện tại tới địa điểm đó,
//phương thức searchDistances sẽ làm điều này.
+ (NSDictionary *)searchDistances:(NSString *)origins destinations:(NSString *)destinations;

//searchDetail sẽ lấy thông tin chi tiết về một nơi nào đó dựa vào tham số reference từ
//searchPlaces.
+ (NSDictionary *)searchDetail:(NSString *)reference;

//searchPlacePhoto lấy url của một tấm hình dựa vào photoReference từ searchPlaces
+ (NSData *)searchPlacePhoto:(NSString *)photoReference;

@end

```

→ Mở file “**Google.m**” để hiện thực lại các phương thức đã khai báo trên:

```

#import "Google.h"

@implementation Google

+ (NSDictionary *)searchPlaces:(CLLocationCoordinate2D )coordinate
    radius:(NSString *)radius
    placeType:(NSString *)placeType
    keyword:(NSString *)keyword {
    NSString *url = [NSString stringWithFormat:@"https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=%f,%f&radius=%@&types=%@&keyword=%@&sensor=true&key=%@", coordinate.latitude, coordinate.longitude, radius, placeType, keyword, GOOGLE_API_KEY];

    return [self queryGooglePlaces:url];
}

+ (NSDictionary *)searchDistances:(NSString *)origins destinations:(NSString *)destinations {
    NSString *url = [NSString stringWithFormat:@"http://maps.googleapis.com/maps/api/distancematrix/json?origins=%@&destinations=%@&mode=walking&sensor=true", origins, destinations];
    return [self queryGooglePlaces:url];
}

+ (NSDictionary *)searchDetail:(NSString *)reference {
    NSString *url = [NSString stringWithFormat:@"https://maps.googleapis.com/maps/api/place/details/json?reference=%@&sensor=true&key=%@", reference, GOOGLE_API_KEY];
    return [self queryGooglePlaces:url];
}

+ (NSData *)searchPlacePhoto:(NSString *)photoReference {
    NSString *strURL = [NSString stringWithFormat:@"https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photoreference=%@&sensor=true&key=%@", photoReference, GOOGLE_API_KEY];
    NSURL *url = [NSURL URLWithString:strURL];
}

```

```

NSData *imgData = [[NSData alloc] initWithContentsOfURL:url];
return imgData;
}

/*queryGooglePlaces sẽ thực hiện request một url lên Google và trả về json, các phương thức khác
(searchPlaces, searchDistances, searchDetail, searchPlacePhoto) chỉ cần truyền vào 1 chuỗi url
và nhận kết quả json trả về.*/
+ (NSDictionary *)queryGooglePlaces:(NSString *)urlGoogle {
NSURL *googleRequestURL = [NSURL URLWithString:[urlGoogle
stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding]];
NSData *data = [NSData dataWithContentsOfURL:googleRequestURL];
NSDictionary *json = [NSJSONSerialization JSONObjectWithData:data options:kNilOptions
error:nil];
return json;
}
@end

```

Hiển thị dữ liệu lên “**FindingPlaceViewController**”

→ Mở “**FindingPlaceViewController.m**” → tìm và xoá các dòng sau để loại bỏ cảnh báo:

```

#pragma mark - Table view data source
#warning Potentially incomplete method implementation.
#warning Incomplete method implementation.

```

→ import thêm “**Google.h**”:

```

#import "FindingPlaceViewController.h"
#import "Google.h"

```

→ Dưới “**@implementation FindingPlaceViewController**” khai báo mảng toàn cục “**arrPlaceSearching**” để hứng kết quả trả về sau khi request tới Google và 1 biến “**coodinate**” để lưu tọa độ hiện :

```

@implementation FindingPlaceViewController
NSMutableArray *arrPlaceSearching;
CLLocationCoordinate2D coordinate;

```

→ Thêm hàm “**searchPlacesFromGoogle**” để nhận kết quả json từ Google sau đó chuyển về Mảng.

```

- (NSMutableArray *)searchPlacesFromGoogle:(NSString *)radius placeType:(NSString *)
placeType keyword:(NSString *)keyword {
//Tìm kiếm các địa điểm từ google.
NSDictionary *json = [Google searchPlaces:coordinate radius:radius placeType:placeType
keyword:keyword];
return [json objectForKey:@"results"];
}

```

→ Tại “**viewDidLoad**” request thử Google:

```

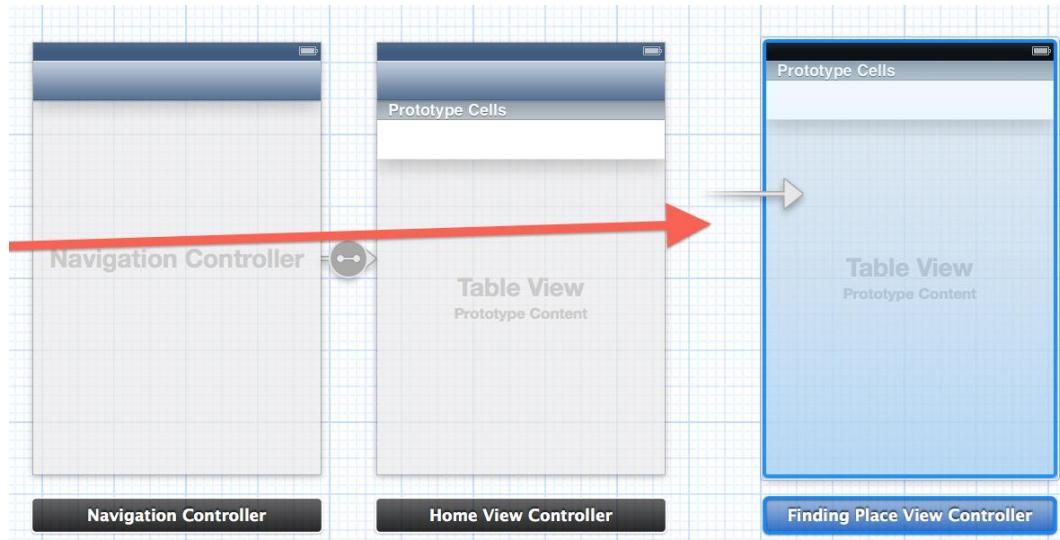
- (void)viewDidLoad
{
[super viewDidLoad];

coordinate.latitude = 10.953212;
coordinate.longitude = 106.802378;
NSString *radius = @"10000";
NSString *placeType = @"establishment";
NSString *keyWord = @"school";
arrPlaceSearching = [self searchPlacesFromGoogle:radius placeType:placeType
keyword:keyWord];

 NSLog(@"%@",arrPlaceSearching);
}

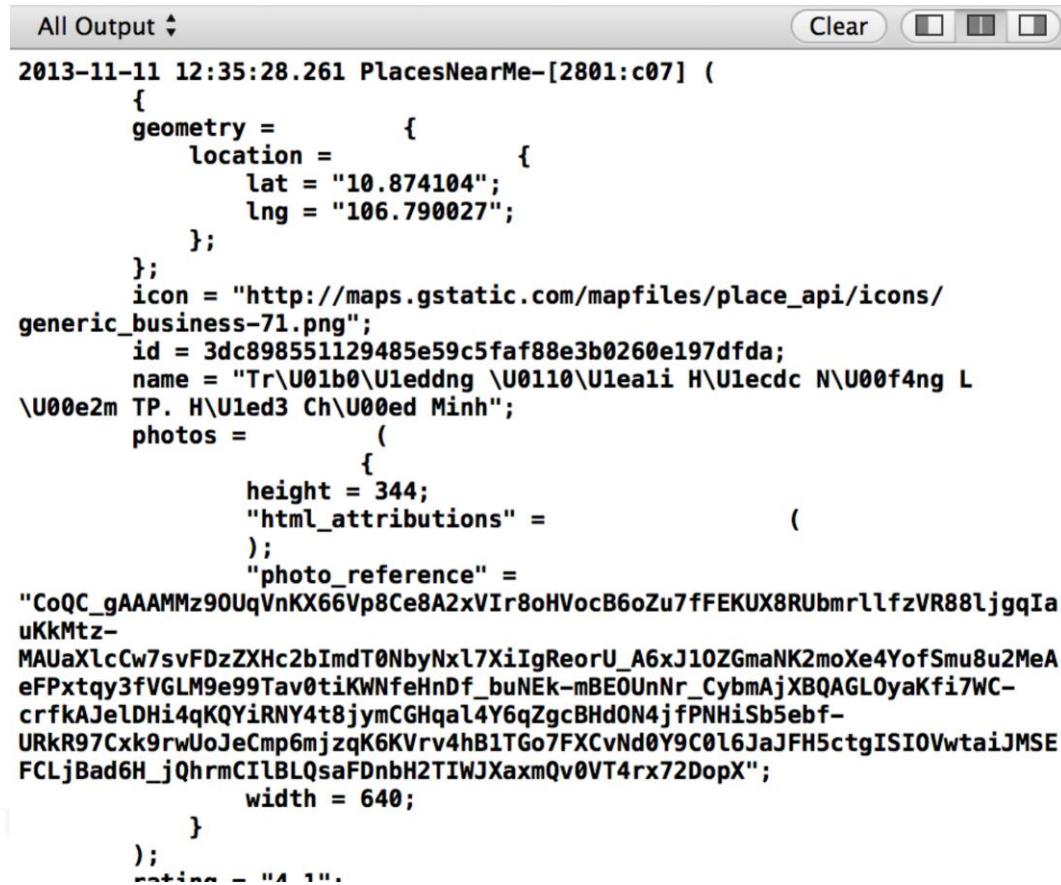
```

→ Chọn “**MainStoryboard.storyboard**” kéo thả mũi tên từ “**NavigationController**” qua “**FindingPlaceViewController**” để chạy thử “**FindingPlaceViewController**”:



**Hình 6.217 Đặt chế độ ưu tiên chạy thử FindingPlace View Controller**

→ Đảm bảo có kết nối internet trước khi chạy, “**Cmd + R**” để chạy thử, kết quả:



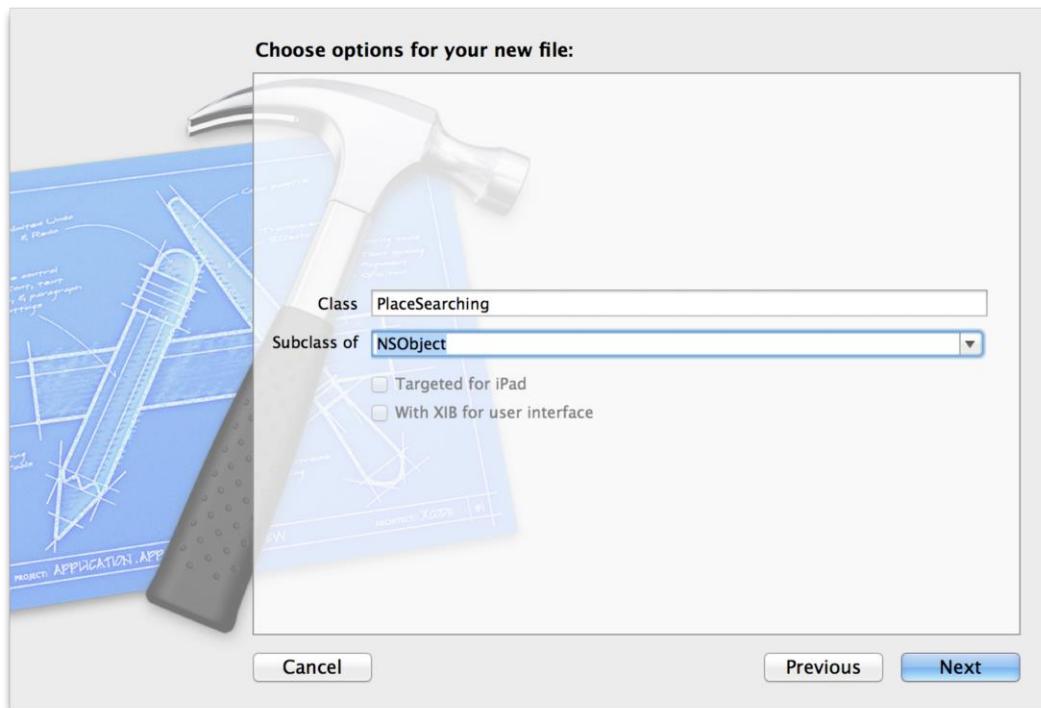
```
All Output ▾
2013-11-11 12:35:28.261 PlacesNearMe-[2801:c07] (
{
    geometry = {
        location = {
            lat = "10.874104";
            lng = "106.790027";
        };
    };
    icon = "http://maps.gstatic.com/mapfiles/place_api/icons/generic_business-71.png";
    id = 3dc898551129485e59c5faf88e3b0260e197dfda;
    name = "Trường Lederding \U0110\U1eadli H\U1ecdc N\U00f4ng L\U00e2m TP. H\U1ed3 Ch\U00ed Minh";
    photos = (
        {
            height = 344;
            "html_attributions" =
            (
            );
            "photo_reference" =
            "CoQC_gAAAMMz90UqVnKX66Vp8CeA2xVlr8oHVocB6oZu7fFEKUX8RUbmrllfzVR88ljgqIauKkMtz-
            MAUaXlcCw7svFDzZXHc2bImdT0NbyNx17XiIgReorU_A6xJ10ZGmaNK2moXe4YofSmu8u2MeA
            eFPxtqy3fVGLM9e99Tav0tiKWNfeHnDf_buNEk-mBE0UnNr_CybmaAjXBQAGL0yaKfi7WC-
            crfkAJelDHi4qKQYiRNY4t8jymCGHqal4Y6qZgcBHd0N4jfPNHiSb5ebf-
            URkR97Cxk9rwUoJeCmp6mjzqK6KVrv4hB1TGo7FXCvNd0Y9C0l6JaJFH5ctgISI0VwtaiJMSE
            FCLjBad6H_jQhrmCILBLQsaFDnbH2TIWJXaxmQv0VT4rx72DopX";
            width = 640;
        }
    );
    rating = "4.1";
}
```

Hình 6.218 Kết quả chạy thử

→ Dựa vào mảng dữ liệu **NSDictionary** như trên, chúng ta sẽ trích các thuộc tính:

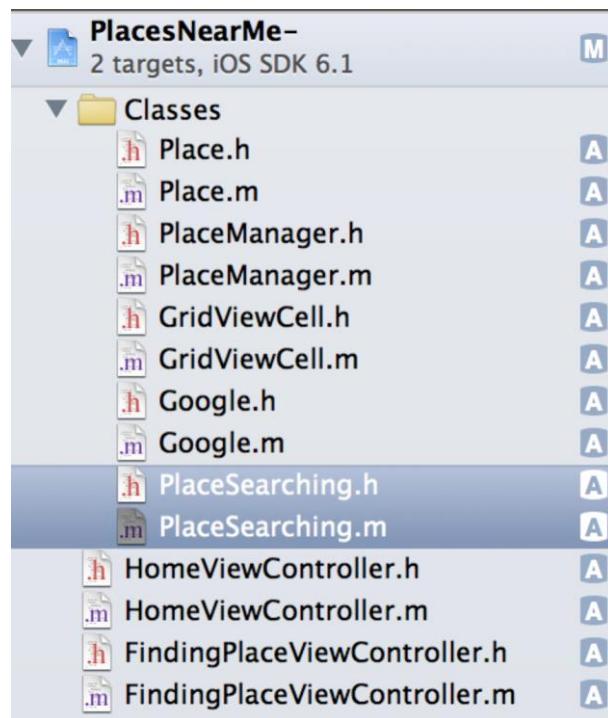
- **name**: tên địa điểm
- **vicinity**: địa chỉ
- **units**: số km
- **latitude**: vĩ độ
- **longitude**: kinh độ
- **reference**: để lấy thông tin chi tiết của địa điểm

→ Sau khi trích ra các thuộc tính trên chúng ta sẽ đẩy dữ liệu vào đối tượng “**PlaceSearching**”, **Cmd + N** để tạo đối tượng này → Class: **PlaceSearching** → Subclass of: **NSObject**.



**Hình 6.219 Tạo mới class**

→ Kết quả:



**Hình 6.220 Kết quả tạo mới class**

→ Mở file “**PlaceSearching.h**” khai báo các thuộc tính như sau:

```
#import <Foundation/Foundation.h>

@interface PlaceSearching : NSObject

@property (nonatomic, strong) NSString *name;
@property (nonatomic, strong) NSString *vicinity;
@property (nonatomic, strong) NSString *units;
@property (nonatomic, strong) NSString *latitude;
@property (nonatomic, strong) NSString *longitude;
@property (nonatomic, strong) NSString *reference;

@end
```

→ Quay lại file “**FindingPlaceViewController.m**” → import thêm “**PlaceSearching.h**”

```
#import "FindingPlaceViewController.h"
#import "Google.h"
#import "PlaceSearching.h"
```

→ Thêm phương thức “**searchDistancesFromGoogle**”, phương thức này sẽ trả về 1 mảng các khoảng cách (số km):

```
- (NSMutableArray *)searchDistancesFromGoogle:(NSMutableArray *)arrPlaces {
    //Lấy các toạ độ(latitude,longitude) của mảng địa điểm vừa tìm được (arrPlaces) để tìm
    //khoảng cách (số Km. VD: 2.5 Km)
    //và đổ kết quả tìm được vào mảng arrDistances.
    NSString *origins= [NSString stringWithFormat:@"%@,%f", coordinate.latitude,
    coordinate.longitude];
    NSString *destinations = [self getDestinations:arrPlaces];

    //dựa vào toạ độ gốc và toạ độ đích để lấy số Km.
    NSDictionary *json = [Google searchDistances:origins destinations:destinations];
    //(*) VD: NSLog(@"%@",[json objectForKey:@"rows"] objectAtIndex:0]
    objectForKey:@"elements"]);
    return [[[json objectForKey:@"rows"] objectAtIndex:0] objectForKey:@"elements"];
}

- (NSString *)getDestinations:(NSMutableArray *)arrPlaces {
    NSString *strDestinations;

    if ([arrPlaces count] > 0) {
        NSDictionary *geometry = [[arrPlaces objectAtIndex:0] objectForKey:@"geometry"];
        NSDictionary *location = [geometry objectForKey:@"location"];
        NSString *lat = [location objectForKey:@"lat"];
        NSString *lng = [location objectForKey:@"lng"];
```

```

int count = [arrPlaces count];
//Gán toạ độ đầu tiên vào chuỗi strDestinations
strDestinations = [NSString stringWithFormat:@"%@,%@", lat, lng];

//Ghép các toạ độ còn lại vào chuỗi strDestinations phân cách bởi dấu "|"
for (int i=1; i<count; i++) {
    geometry = [[arrPlaces objectAtIndex:i] objectForKey:@"geometry"];
    location = [geometry objectForKey:@"location"];
    lat = [location objectForKey:@"lat"];
    lng = [location objectForKey:@"lng"];

    strDestinations = [NSString stringWithFormat:@"%@|%@", strDestinations, lat, lng];
}

return strDestinations;
}

```

### Giải thích:

- Phương thức “**getDestinations**” sẽ lấy ra tất cả các toạ độ trong mảng mà “**searchPlacesFromGoogle**” trả về, sau đó ghép các toạ độ này lại thành 1 chuỗi các toạ độ phân cách nhau bởi dấu “|”.

**VD:** Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

10.970549,106.915440|10.980949,106.915440|10.970549,106.915440|10.980949,106.915440 hoặc NULL

- Dựa vào toạ độ gốc và chuỗi toạ độ như trên, hàm “**searchDistancesFromGoogle**” sẽ request tới Google và nhận về tài liệu json chứa các khoảng cách, sau đó “**searchDistancesFromGoogle**” sẽ chuyển json về thành mảng và trả mảng đó về. Cụ thể hàm “**searchDistancesFromGoogle**” sẽ thực thi HTTP URL sau của Google:

<http://maps.googleapis.com/maps/api/distancematrix/json?origins=10.953212,106.802378&destinations=10.970549,106.915440|10.980949,106.915440|10.970549,106.915440|10.980949,106.915440&mode=walking&units=imperial&sensor=true>

Kết quả:

```
{  
    "destination_addresses" : [  
        "8 Quốc Lộ 1, Tân Hòa, Biên Hòa, Đồng Nai, Việt Nam",  
        "Lê Ngô Cát, Tân Hòa, Biên Hòa, Đồng Nai, Việt Nam",  
        "8 Quốc Lộ 1, Tân Hòa, Biên Hòa, Đồng Nai, Việt Nam",  
        "Lê Ngô Cát, Tân Hòa, Biên Hòa, Đồng Nai, Việt Nam"  
    ],  
    "origin_addresses" : [ "4 Huỳnh Văn Nghệ, Bửu Long, Biên Hòa, Đồng Nai, Việt Nam" ],  
    "rows" : [  
        {  
            "elements" : [  
                {  
                    "distance" : {  
                        "text" : "8.3 mi",  
                        "value" : 13392  
                    },  
                    "duration" : {  
                        "text" : "2 hours 47 mins",  
                        "value" : 10030  
                    },  
                    "status" : "OK"  
                },  
                {  
                    "distance" : {  
                        "text" : "8.7 mi",  
                        "value" : 14070  
                    },  
                    "duration" : {  
                        "text" : "2 hours 54 mins",  
                        "value" : 10428  
                    },  
                    "status" : "OK"  
                },  
                {  
                    "distance" : {  
                        "text" : "8.3 mi",  
                        "value" : 13392  
                    },  
                    "duration" : {  
                        "text" : "2 hours 47 mins",  
                        "value" : 10030  
                    },  
                    "status" : "OK"  
                }  
            ]  
        }  
    ]  
}
```

Nguyễn Anh Tiết Vàng © 2013

→ Chạy thử hàm “**searchDistanceFromGoogle**” bằng cách sửa lại hàm “**viewDidLoad**” như sau:

```
- (void)viewDidLoad  
{  
    [super viewDidLoad];  
  
    coordinate.latitude = 10.953212;  
    coordinate.longitude = 106.802378;  
    NSString *radius = @"10000";  
    NSString *placeType = @"establishment";  
    NSString *keyWord = @"school";  
    arrPlaceSearching = [self searchPlacesFromGoogle:radius placeType:placeType  
    keyword:keyWord];  
  
    NSLog(@"%@",[self searchDistancesFromGoogle:arrPlaceSearching]);  
}
```

→ Kết quả:

```
All Output ▾
Clear
2013-11-11 14:44:43.231 PlacesNearMe-[4583:c07] (
{
    distance = {
        text = "13.6 km";
        value = 13586;
    };
    duration = {
        text = "2 hours 49 mins";
        value = 10136;
    };
    status = OK;
},
{
    distance = {
        text = "0.5 km";
        value = 498;
    };
    duration = {
        text = "6 mins";
        value = 359;
    };
    status = OK;
},
-
```

Hình 6.222 Kết quả chạy thử

→ Tổng kết: sau 2 lần chạy thử, chúng ta đã có 2 mảng mà phần tử của mỗi mảng chính là 1 **NSDictionary** (kiểu dữ liệu gồm có “key” và “value”):

Mảng đầu tiên được trả về từ hàm “searchPlacesFromGoogle”:

All Output    

```

2013-11-11 14:55:47.303 PlacesNearMe-[4787:c07] (
{
    geometry = {
        location = {
            lat = "10.874104";
            lng = "106.790027";
        };
        icon = "http://maps.gstatic.com/mapfiles/place_api/icons/
generic_business-71.png";
        id = "3dc898551129485e59c5faf88e3b0260e197dfda";
        name = "Tr\U01b0\U1eddng \U0110\U1ea1i H\U1ecdc N\U00f4ng L
\U00e2m TP. H\U1ed3 Ch\U00ed Minh";
        photos = (
{
            height = 344;
            "html_attributions" =
(
);
            "photo_reference" = "CoQC_gAAAFCRk5j6ds_Z9lQ0jnFrwhkK6_-
hfAnV4yrnuh-
nf0Dv0Is48qafwIteDRxodFvn4kRQT1ngKhxxsfjj2IJlK3ThlXbA3_3PxmgirbmUJYl0npN-
IvfUDGL7XZFAzHtma2M9gm9Rka5hkr1YiJFnL-sRib8dwE-X9C0dcf_3wzJDyDzG13mT-
seF2oEJvgstE36hMRR8_deq17MCvqqD37ZPik6ySor0CJzBVQCYcq8fyN5WsXhW8PsP-
eawj284RCqimfwXYq1_2zKQ0GRUPBFGGrHjq_hj4s1Btml8rJ2HuVcQDndTSztqZG1Cbh-
rB69Rmg3_sg8TIK-Y2CsivoSEJq51bWNxx3PV46zm04KG0YaFD-
UanB84pJVR0nx05Ji3XQ-1bXB";
            width = 640;
}
);
        rating = "4.1";
        reference =
"CqQBlwAAAF4Z4ZX2xPVhVNmnHGR4Vcd7eJe7nwtge36pn8Q_BkQNoEV0TcnJVI3P2MSp1cwx
NSJX6n4i_hhwkYJd1f3tVW-
NXTVTrLPkRwm47k5yqx0t5A2APe9aOn5eFAkq4eG0UniYSvgayKWhgp0UM1cqNh5iJ6lKPmm
_1LyWAinp_YPU4NqrgkZRrLYcXBV5yEPQKACsZu6rF9wCdmP1kWNMRESEHrgsxv80MAA53PeK
Y0cc44aFHJaeYLQBpSC_rXFsrro3kyL8e-";
        types = (
            establishment
);
        vicinity = "Khu ph\U1ed1 6- Ph\U01b0\U1eddng Linh Trung - Qu
\U1eadn Th\U1ee7 \U0110\U1ee9c";
},
{
    geometry =

```

Hình 6.223 Mảng thứ nhất trả về

Mảng thứ hai được trả về từ hàm “**searchDistancesFromGoogle**”

All Output Clear

```
2013-11-11 15:02:56.002 PlacesNearMe-[4943:c07] (
    {
        distance =
            {
                text = "13.6 km";
                value = 13586;
            };
        duration =
            {
                text = "2 hours 49 mins";
                value = 10136;
            };
        status = OK;
    },
    {
        distance =
            {
                text = "0.5 km";
                value = 498;
            };
        duration =
            {
                text = "6 mins";
                value = 359;
            };
        status = OK;
    },
    {
        distance =
            {
                text = "13 m";
                value = 13;
            };
    }
)
```

**Hình 6.224 Mảng thứ hai trả về**

→ Chú ý những ô vuông khoanh đỏ phía trên chính là những thuộc tính của lớp “**PlaceSearching**” mà tôi cần lấy ra. Để lấy ra các thuộc tính từ 2 mảng phía trên, tôi sẽ sử dụng 1 hàm “**“getPlaceSearching”**” có nhiệm vụ lấy các thuộc tính tôi đã khoanh đỏ phía trên và đưa vào 1 mảng các đối tượng ”**PlaceSearching**”, hàm này như sau:

```
//Trả về: mảng arrPlaceSearching từ 2 mảng arrPlaces và arrDistances
-(NSMutableArray *)getPlaceSearching:(NSMutableArray *)arrPlaces {
    NSMutableArray *arrPlaceSearching = [[NSMutableArray alloc] init];

    if ([arrPlaces count] > 0) {
        NSMutableArray *arrDistances = [self searchDistancesFromGoogle:arrPlaces];
        int count = [arrPlaces count];

        for (int i=0; i<count; i++) {
            NSDictionary *geometry = [arrPlaces[i] objectForKey:@"geometry"];
            NSDictionary *location = [geometry objectForKey:@"location"];

            PlaceSearching *ps = [[PlaceSearching alloc] init];
            ps.name = [[arrPlaces objectAtIndex:i] objectForKey:@"name"];
            ps.vicinity = [[arrPlaces objectAtIndex:i] objectForKey:@"vicinity"];
            ps.latitude = [location objectForKey:@"lat"];
            ps.longitude = [location objectForKey:@"lng"];
            ps.reference = [arrPlaces[i] objectForKey:@"reference"];

            ps.units = [[[arrDistances objectAtIndex:i] objectForKey:@"distance"]

```

```

objectForKey:@"text"];
    [arrPlaceSearching addObject:ps];
}
return arrPlaceSearching;
}

```

→ Chạy thử hàm “**getPlaceSearching**” bằng cách sửa lại “**viewDidLoad**” như sau:

```

- (void)viewDidLoad
{
    [super viewDidLoad];

coordinate.latitude = 10.953212;
coordinate.longitude = 106.802378;
NSString *radius = @"10000";
NSString *placeType = @"establishment";
NSString *keyWord = @"school";

NSMutableArray *arrPlaces = [self searchPlacesFromGoogle:radius placeType:placeType
keyword:keyWord];
arrPlaceSearching = [self getPlaceSearching:arrPlaces];

for (PlaceSearching *ps in arrPlaceSearching) {
    NSLog(@"%@", ps.name);
}
}

```

→ Kết quả:

```

2013-11-11 15:45:43.813 PlacesNearMe-[5603:c07] Trường Đại Học Nông Lâm
TP. Hồ Chí Minh
2013-11-11 15:45:43.815 PlacesNearMe-[5603:c07] Lac Hong Bilingual School
2013-11-11 15:45:43.815 PlacesNearMe-[5603:c07] Lac Hong University
2013-11-11 15:45:43.815 PlacesNearMe-[5603:c07] Đại học Nông Lâm TP.HCM
2013-11-11 15:45:43.815 PlacesNearMe-[5603:c07] The University of
Information Technology, Vietnam National University - Ho Chi Minh City
2013-11-11 15:45:43.816 PlacesNearMe-[5603:c07] Trường Đại Học Việt Đức
2013-11-11 15:45:43.816 PlacesNearMe-[5603:c07] Security Service College
2013-11-11 15:45:43.816 PlacesNearMe-[5603:c07] Trường Tiểu học Xuân Hiệp
2013-11-11 15:45:43.816 PlacesNearMe-[5603:c07] Dong Nai University of
Technology
2013-11-11 15:45:43.817 PlacesNearMe-[5603:c07] Trường THCS Tân Đông Hiệp
2013-11-11 15:45:43.817 PlacesNearMe-[5603:c07] Nguyen Du Primary School
2013-11-11 15:45:43.817 PlacesNearMe-[5603:c07] Tran Bien High School
2013-11-11 15:45:43.817 PlacesNearMe-[5603:c07] Trường THPT Dĩ An
2013-11-11 15:45:43.818 PlacesNearMe-[5603:c07] Ngo Quyen High School
2013-11-11 15:45:43.818 PlacesNearMe-[5603:c07] Trường Chính trị tỉnh
Đồng Nai

```

**Hình 6.225 Kết quả trả về**

→ Sửa lại 2 hàm “**numberOfSectionsInTableView**” và “**numberOfRowsInSection**” như sau:

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return arrPlaceSearching.count;
}
```

→ Thêm hàm “**createLabel**” để tạo Label cho Cell

```
- (UILabel *)createLabel:(CGRect)rect
    font:(NSString *)font
    size:(int)size
    text:(NSString *)text {
    UILabel *lbl = [[UILabel alloc] initWithFrame:rect];
    [lbl setFont:[UIFont fontWithName:font size:size]];
    [lbl setAdjustsFontSizeToFitWidth:YES];
    lbl.text = text;

    return lbl;
}
```

→ Tại hàm “**cellForRowAtIndexPath**” sửa lại như sau:

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier
forIndexPath:indexPath];

    int row = indexPath.row;

    PlaceSearching *ps = [[PlaceSearching alloc] init];
    ps = arrPlaceSearching[row];

    // Configure the cell...
    //Set name
    UILabel *lblName = [self createLabel:CGRectMake(5, 5, 310, 20) font:@"Arial-BoldMT"
size:17 text:ps.name];
    [cell.contentView addSubview:lblName];

    //Set vicinity
    UILabel *lblVicinity = [self createLabel:CGRectMake(5, 25, 290, 20) font:@"Arial" size:14
text:ps.vicinity];
```

```

[cell.contentView addSubview:lblVicinity];

//Set numbers
UILabel *lblNumbers = [self createLabel:CGRectMake(0, 48, 320, 10) font:@"Arial" size:10
text:[NSString stringWithFormat:@"%@",row+1]];
lblNumbers.textAlignment = NSTextAlignmentCenter;
[cell.contentView addSubview:lblNumbers];

//Set Distance (kilometers)
UILabel *lblDistance = [self createLabel:CGRectMake(260, 46, 60, 15) font:@"Arial" size:13
text:ps.units];
[cell.contentView addSubview:lblDistance];

return cell;
}

```

→ Set chiều cao cho từng dòng trong “**TableViewController**”

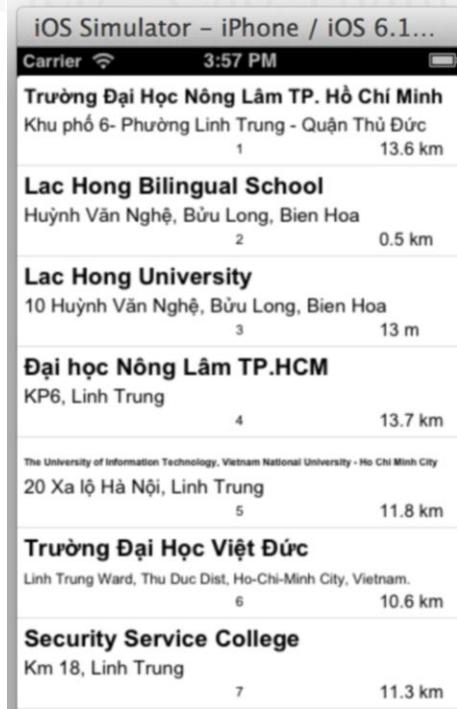
```

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)
*)indexPath {
    return 65;
}

```

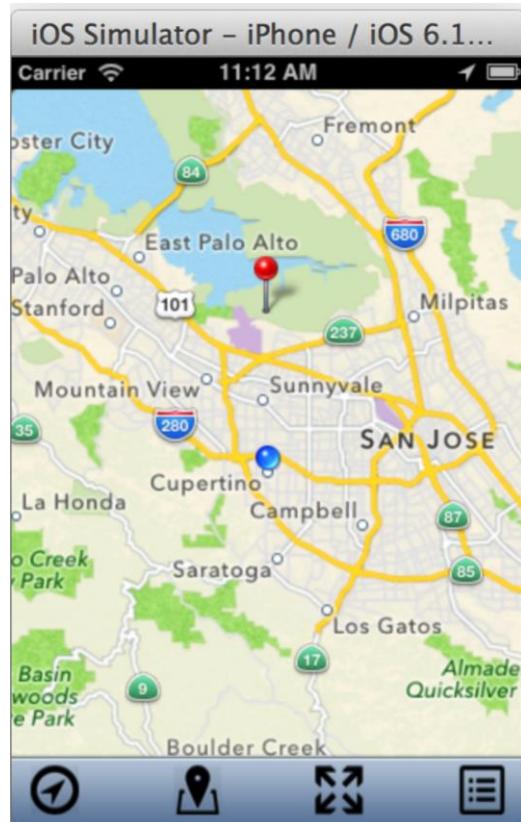
→ Cmd + R chạy thử → kết quả:

Nguyễn Anh Tiễn - Cao Thành Vàng © 2013



**Hình 6.226 Kết quả chạy thử**

## Bước 6: Xây dựng ViewController “MapViewController”

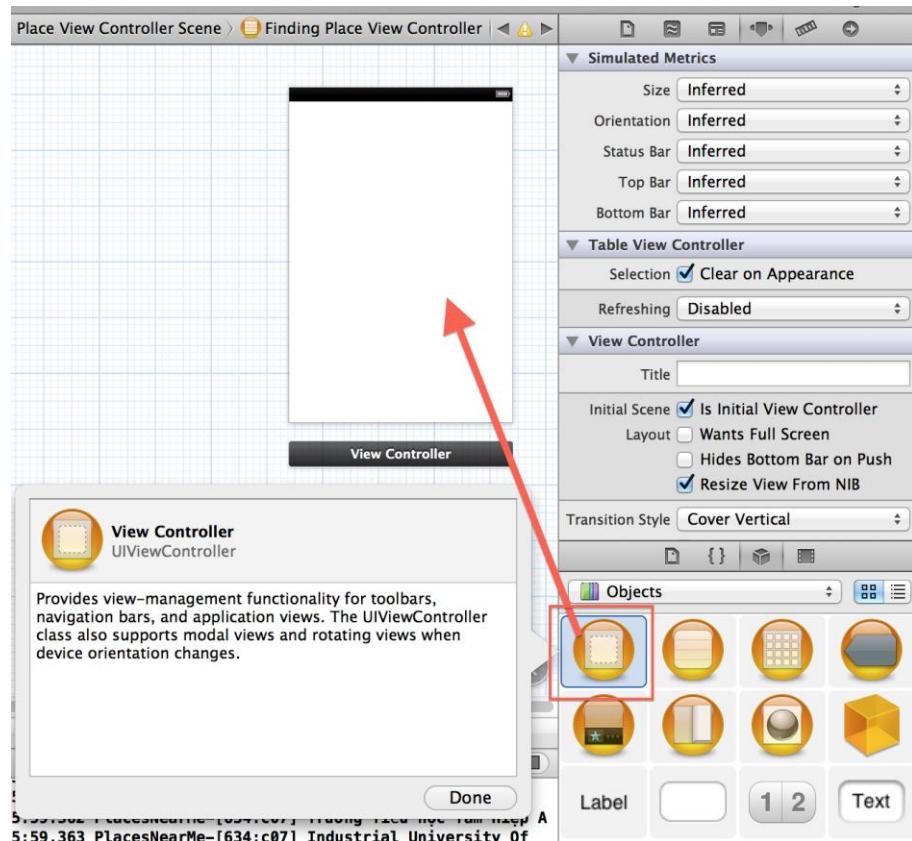


Nguyễn Anh Vàng © 2013

**Hình 6.227 Map View Controller**

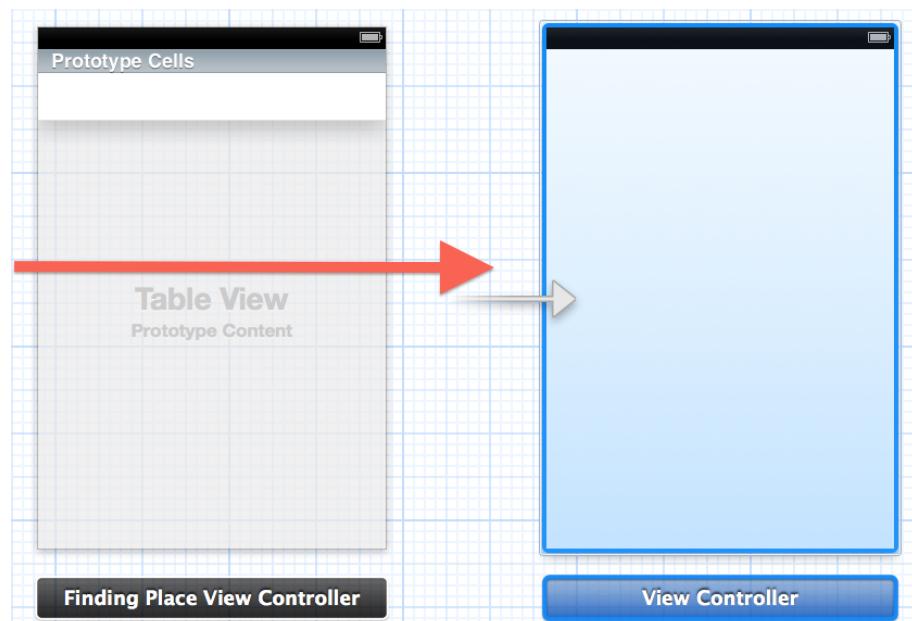
Thiết kế giao diện:

→ Click “MainStoryboard.storyboard” → Kéo thả “View Controller” vào storyboard:



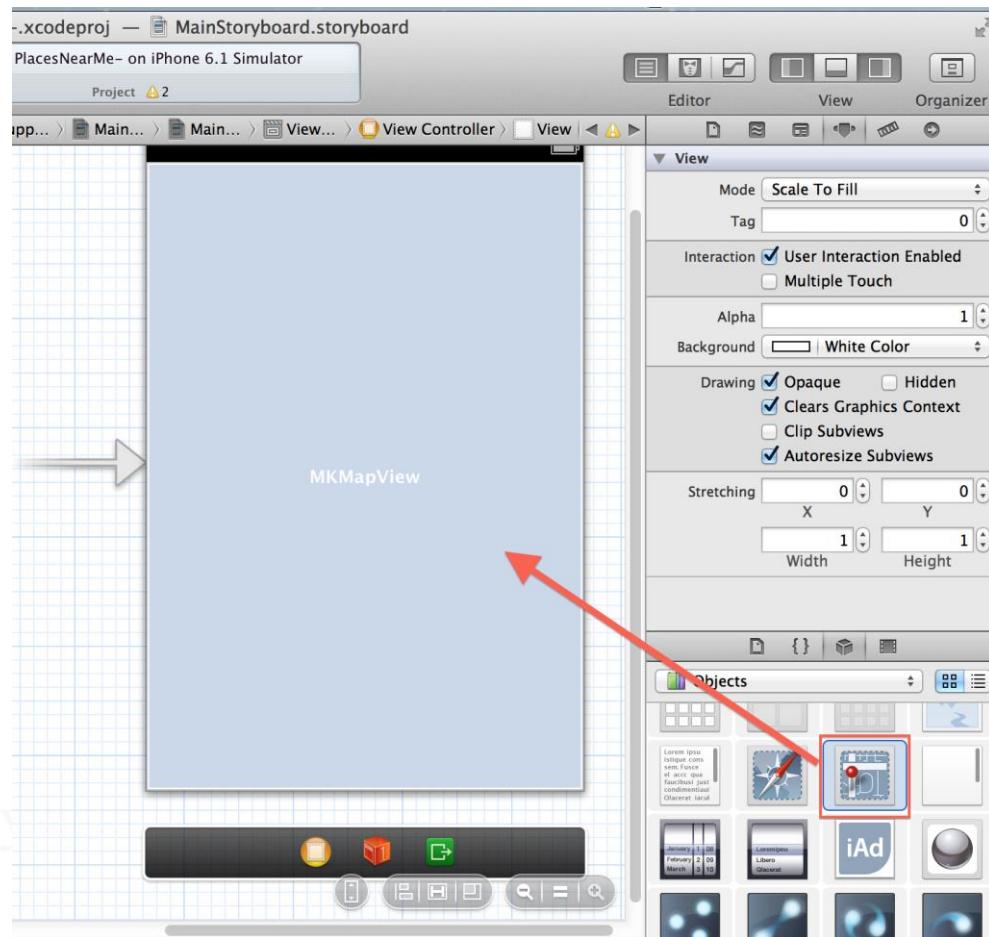
Hình 6.228 Kéo thả View Controller

→ Kéo thả mũi tên từ “Finding Place View Controller” sang “View Controller”.



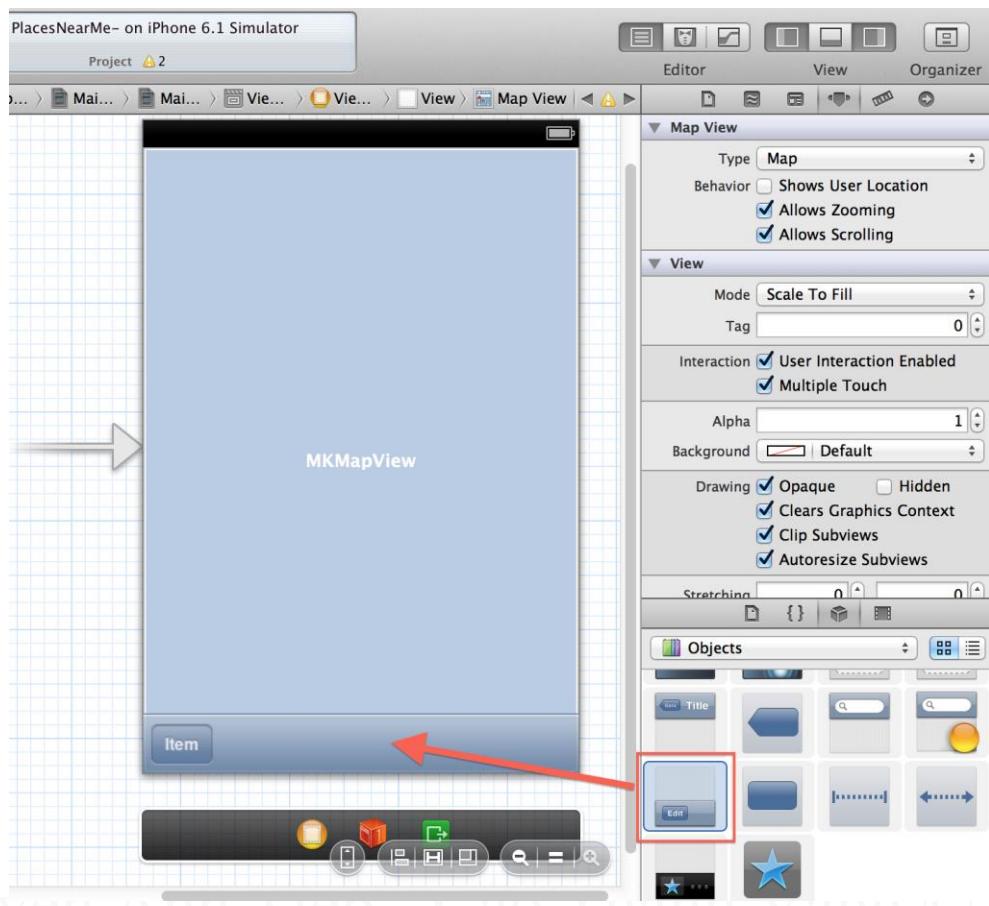
Hình 6.229 Đặt cho View Controller chạy thử trước

→ Kéo thả đối tượng “Map View” vào “View Controller”



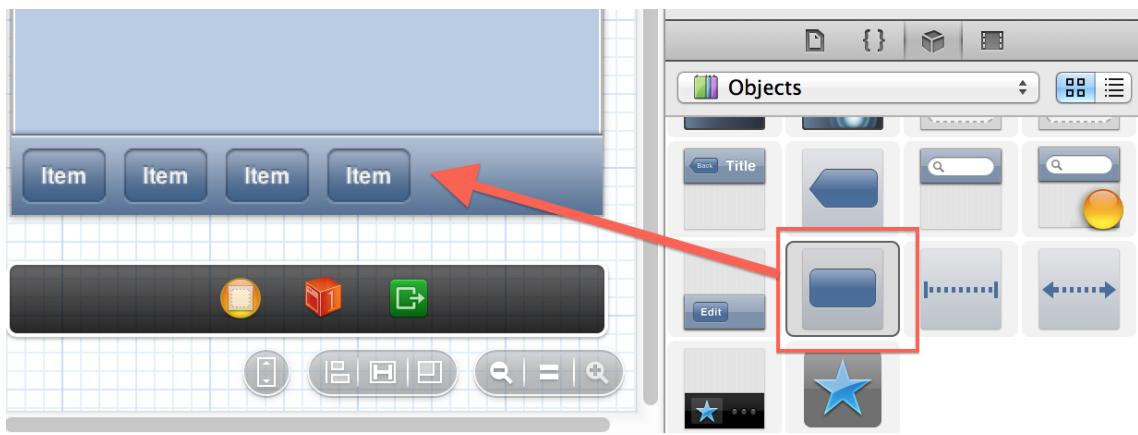
Hình 6.230 Kéo thả Map View vào View Controller

→ Kéo thả “Toolbar” vào “View Controller”



**Hình 6.231 Kéo thả Toolbar vào giao diện**

→ Kéo thả thêm 3 “Bar button item” vào “Toolbar”:



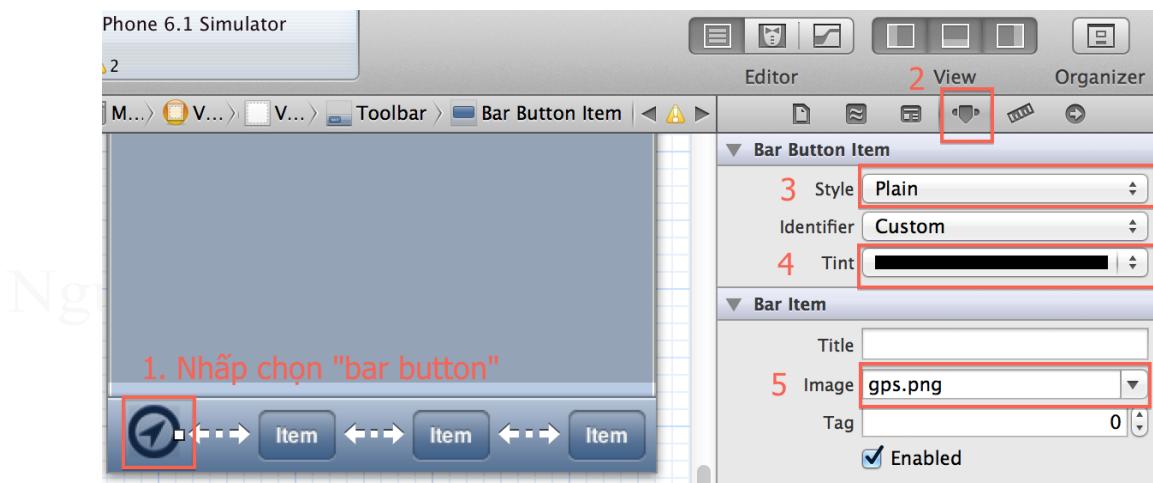
**Hình 6.232 Kéo thả Bar button item vào Toolbar**

→ Giữa các “Bar button” kéo thả “Flexible Space Bar Button Item”



**Hình 6.233 Kéo thả Flexible Space Bar Button Item vào**

→ Cấu hình cho “Bar button item” đầu tiên:



**Hình 6.234 Cấu hình Bar button item**

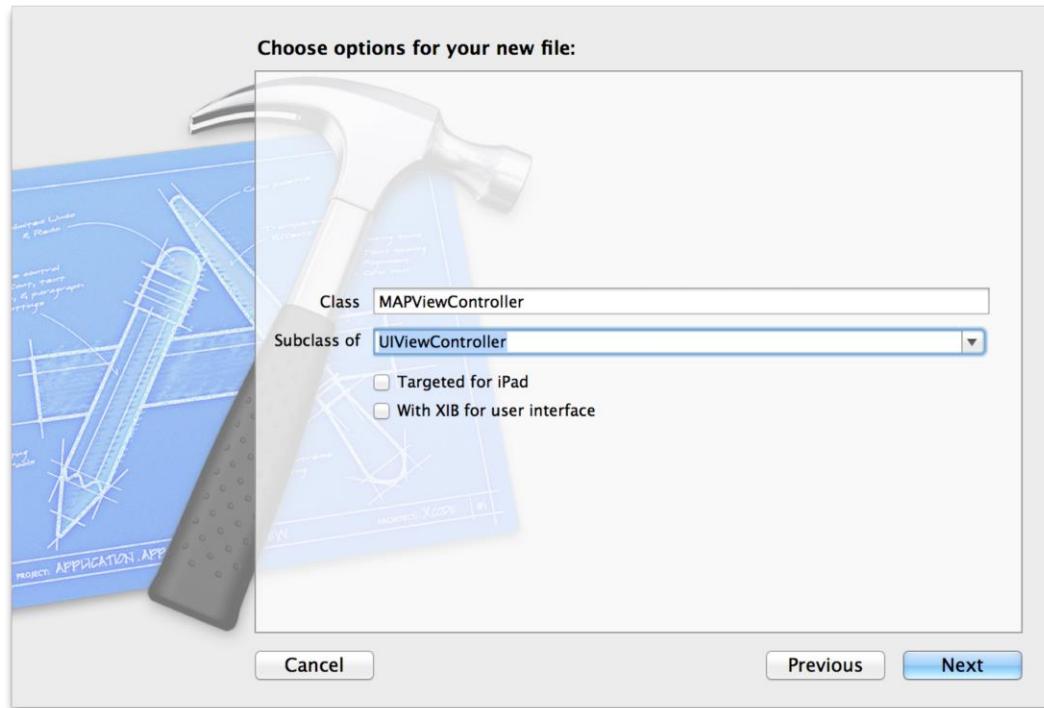
→ Các bar button khác làm tương tự với image: map.png, full\_screen.png, detail.png



**Hình 6.235 Các button khác tương tự**

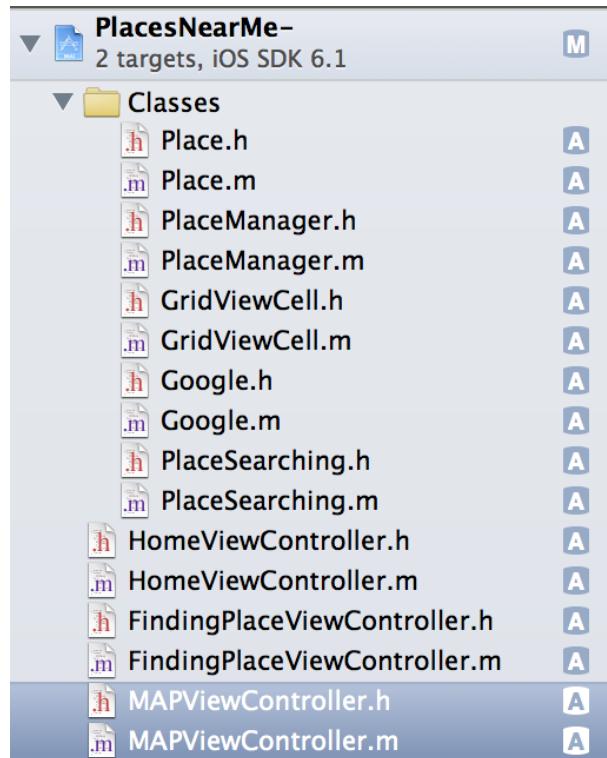
Tạo 1 Class “**MapViewController**” và add vào “**ViewController**” vừa tạo.

- Cmd + N để tạo Object → Objective-C Class → Class: MapViewController  
→ Subclass of: UIViewController.



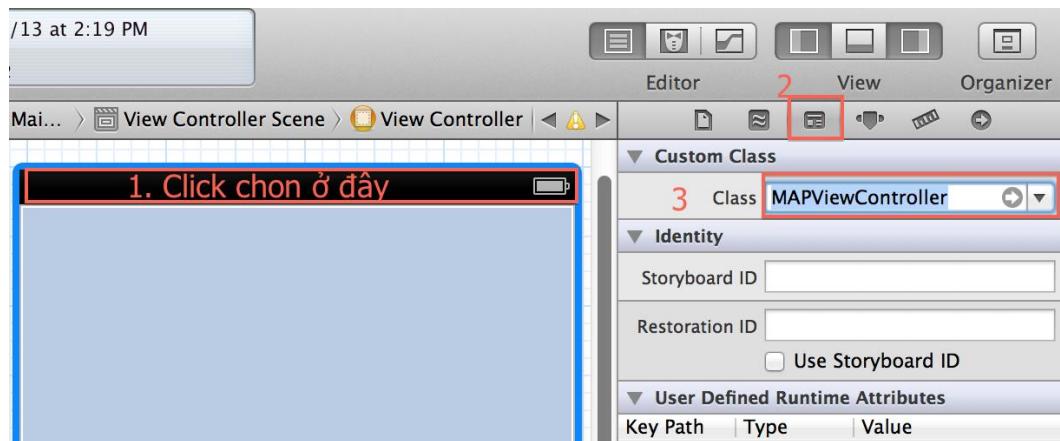
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
**Hình 6.236 Tạo mới class**

→ Kết quả:



Hình 6.237 Class mới tạo

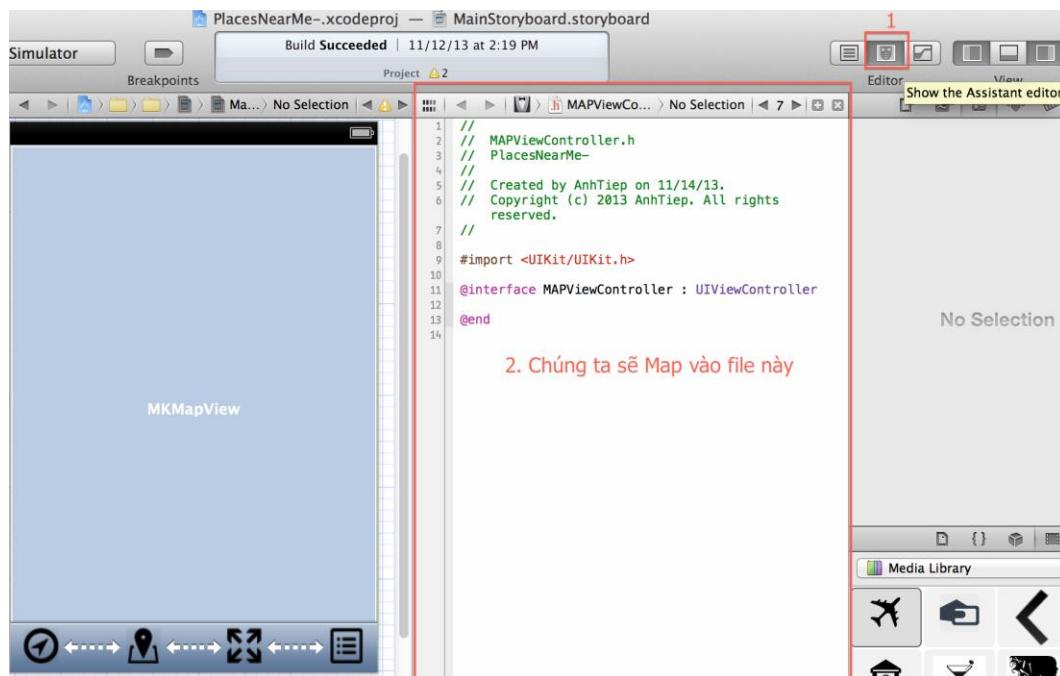
→ Click “MainStoryboard.storyboard” → Click “ViewController” → Class: MapViewController.



Hình 6.238 Thêm class cho MapViewController

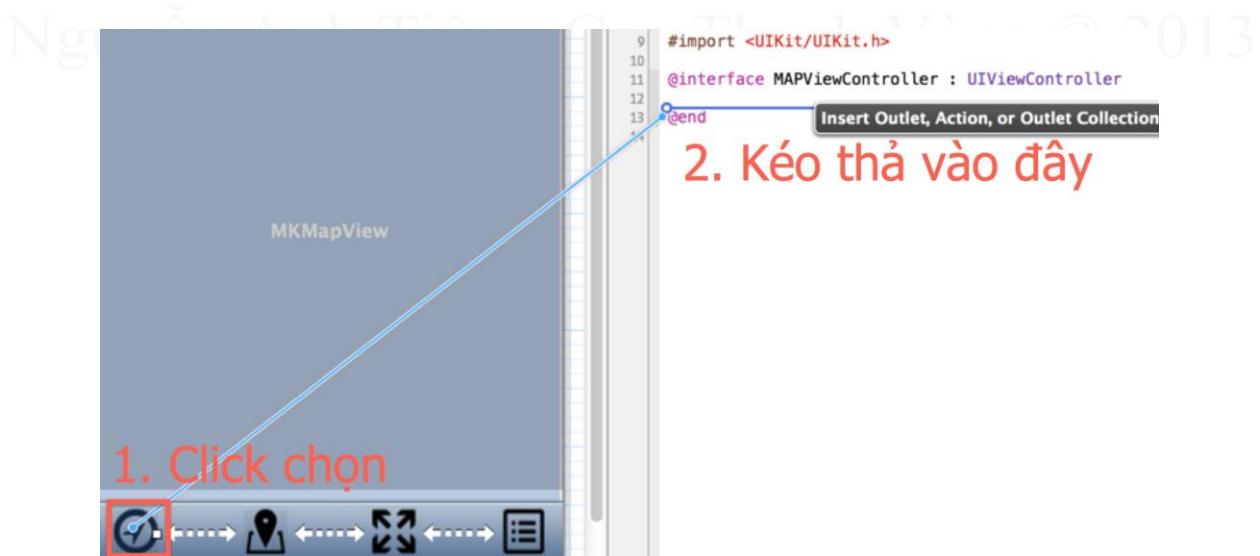
Map các đối tượng vào file “MapViewController.h”

→ Click “MainStoryboard.storyboard” → click Assistant editor



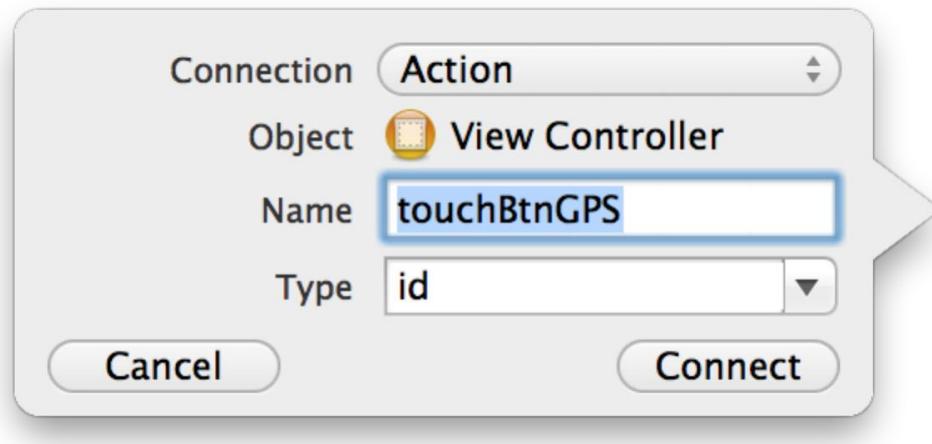
**Hình 6.239 Ánh xạ đối tượng**

→ Click chọn đối tượng cần map → kéo thả vào giữa “@interface” và “@end”:



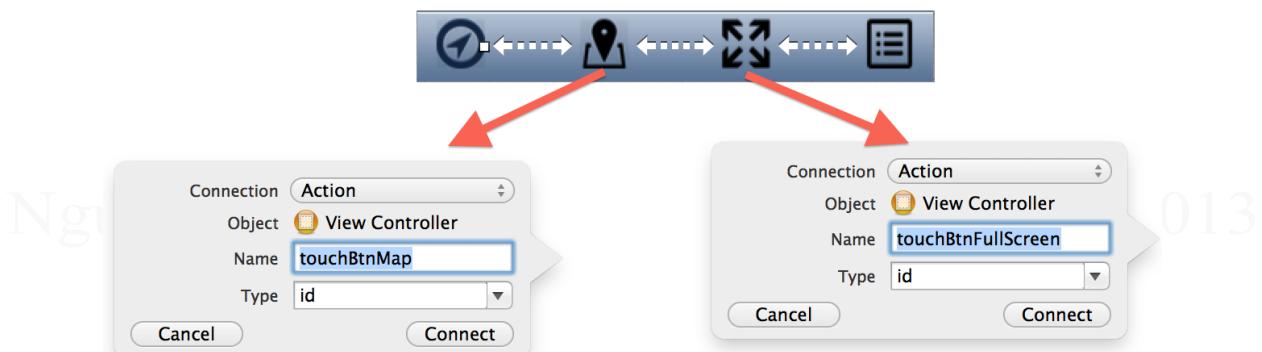
**Hình 6.240 Kéo thả đối tượng**

→ Connection: Action → Name: touchBtnGPS



**Hình 6.241 Chọn loại kết nối**

→ Map tương tự với 2 đối tượng như sau:



**Hình 6.242 Thực hiện tương tự cho các đối tượng còn lại**

→ Kết quả:

```
@interface MAPViewController : UIViewController
{
    - (IBAction)touchBtnGPS:(id)sender;
    - (IBAction)touchBtnMap:(id)sender;
    - (IBAction)touchBtnFullScreen:(id)sender;
}
@end
```

**Hình 6.243 Kết quả**

→ Qua file “**MapViewController.m**” chúng ta sẽ thấy có 3 phương thức mới xuất hiện:

```

- (IBAction)touchBtnGPS:(id)sender {
}

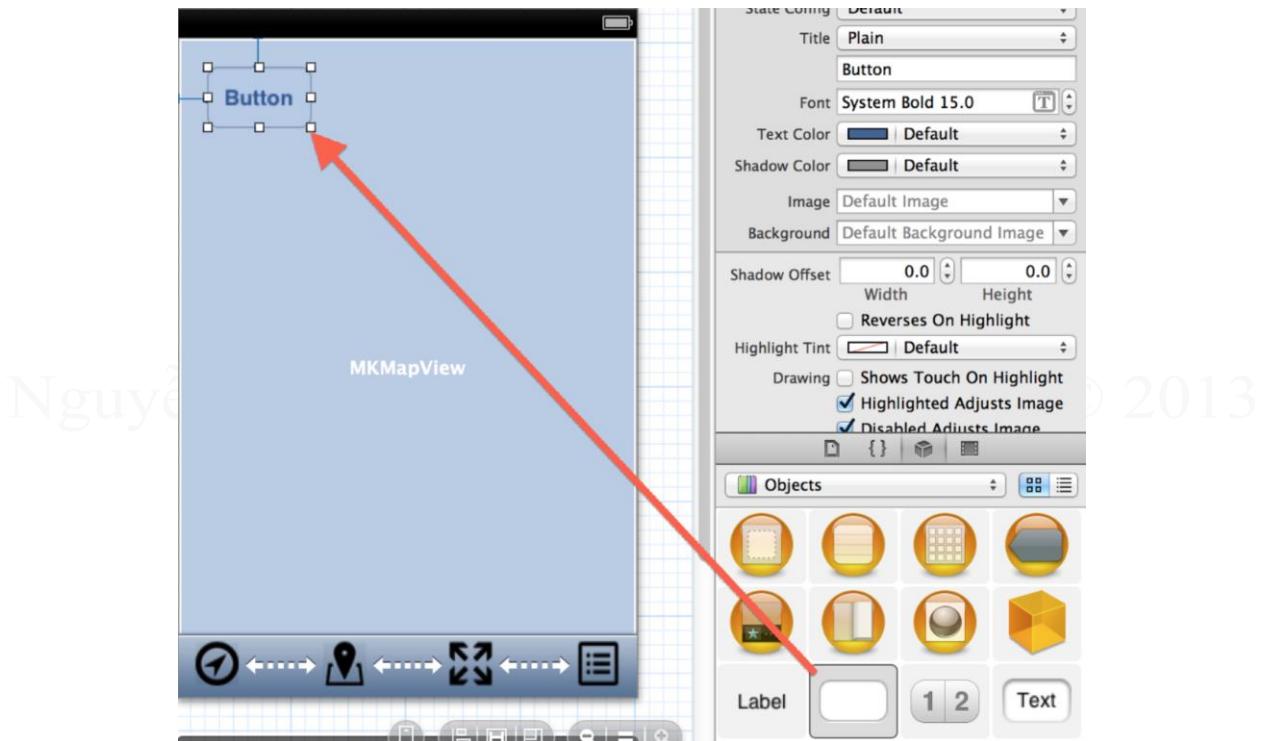
- (IBAction)touchBtnMap:(id)sender {
}

- (IBAction)touchBtnFullScreen:(id)sender {
}

```

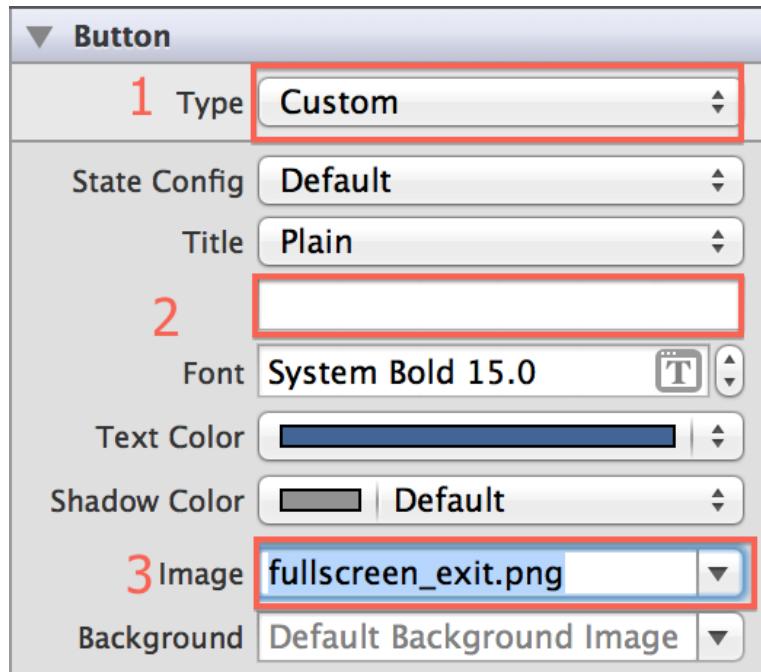
**Hình 6.244 Ba phương thức mới**

→ Quay trở lại giao diện thiết kế kéo thêm 1 đối tượng “Round Rect Button”



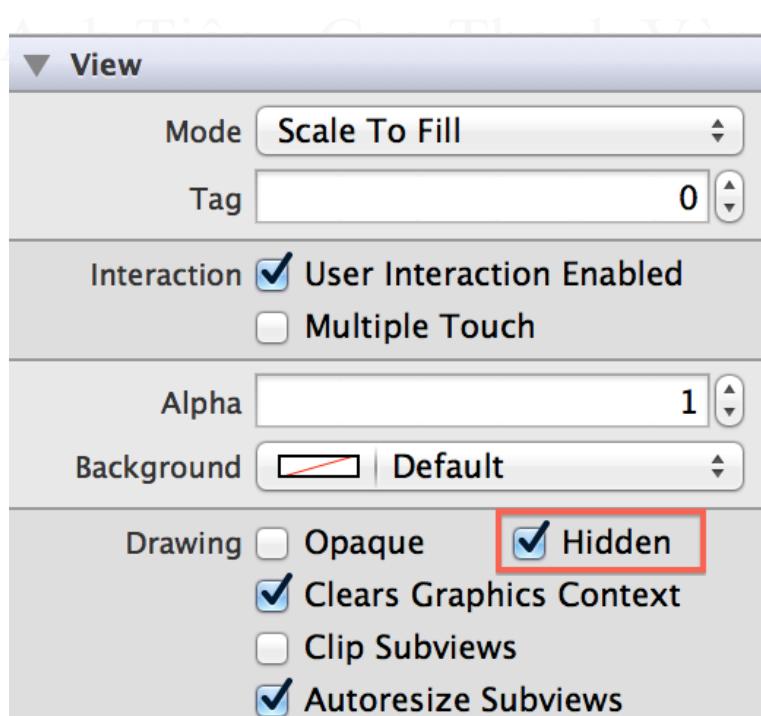
**Hình 6.245 Thêm đối tượng button**

→ Cấu hình các thuộc tính cho **Button** → Type: **Custom** → Phần nội dung để trống  
 → image: **fullscreen\_exit.png**.



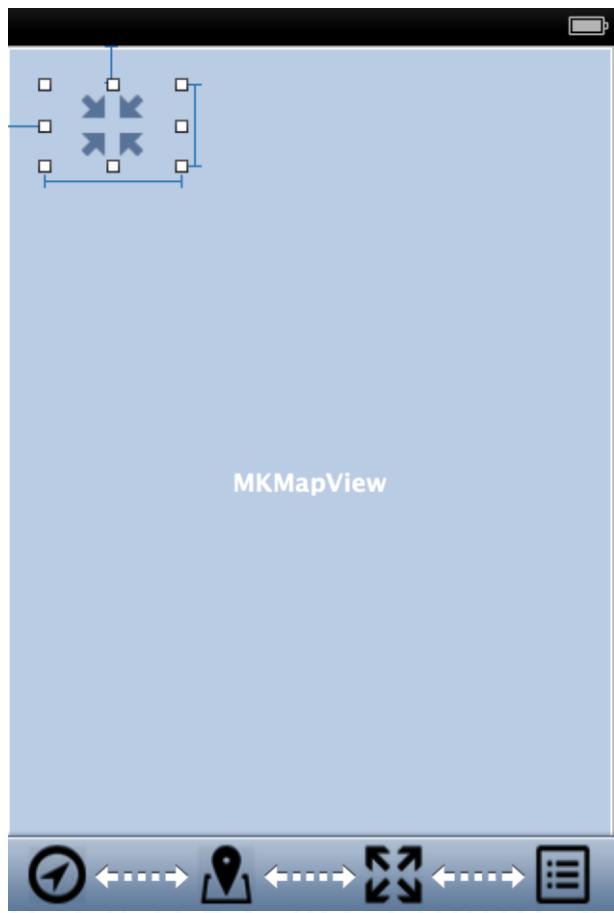
**Hình 6.246** **Chỉnh thuộc tính button**

→ Check “**hidden**” để mặc định button này ẩn



**Hình 6.247** **Chọn Hidden**

→ Kết quả:



Nguyễn Anh Tú - Cao Thanh Vàng © 2013

Hình 6.248 Kết quả MapView

→ Mở file “**MapViewController.h**” thêm khối ngoặc “{ }” sau “**@interface MAPViewController : UIViewController**” để Map các thuộc tính:

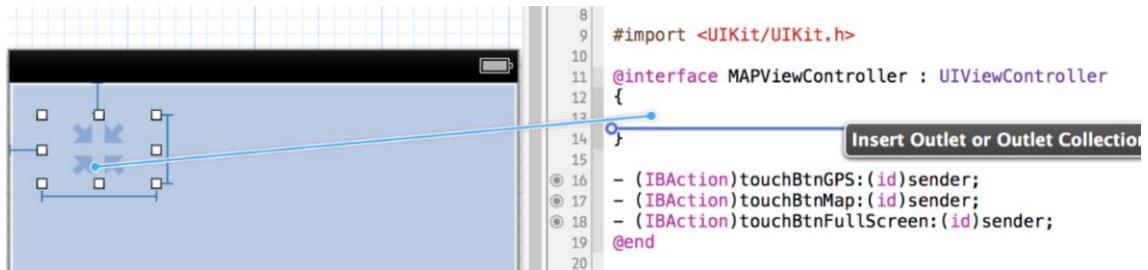
```
#import <UIKit/UIKit.h>

@interface MAPViewController : UIViewController
{
}

- (IBAction)touchBtnGPS:(id)sender;
- (IBAction)touchBtnMap:(id)sender;
- (IBAction)touchBtnFullScreen:(id)sender;
@end
```

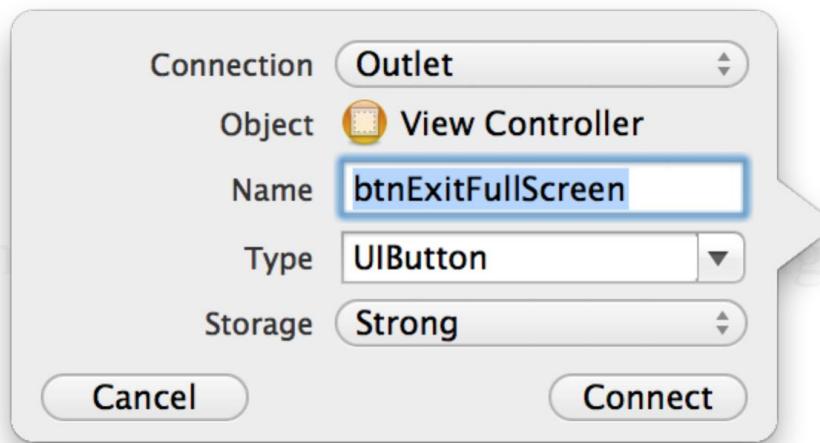
Hình 6.249 Bổ sung dấu ngoặc

→ Map Tương tự cho đối tượng mới thêm vào (Round Rect Button) nhưng kéo thả vào trong khái “{ }” →



Hình 6.250 Ánh xạ button

→ Connection: **Outlet** → Name: **btnExitFullScreen**.

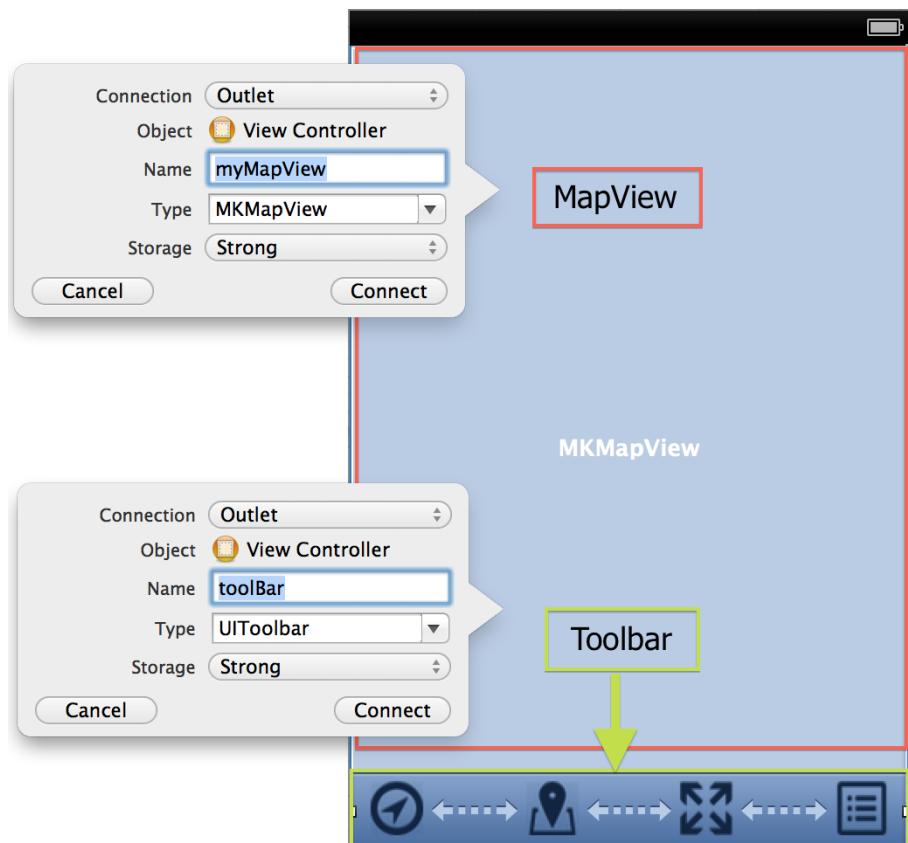


Hình 6.251 Chọn loại kết nối

→ Tương tự Map 2 đối tượng “MapView” và “Toolbar”

**MapView:** Connection: outlet → Name: myMapView

**Toolbar:** Connection: outlet → Name: toolBar



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
**Hình 6.252 Ánh xạ các đối tượng khác**

→ Kết quả (Sẽ có 1 lỗi nhỏ do chưa importMapKit):

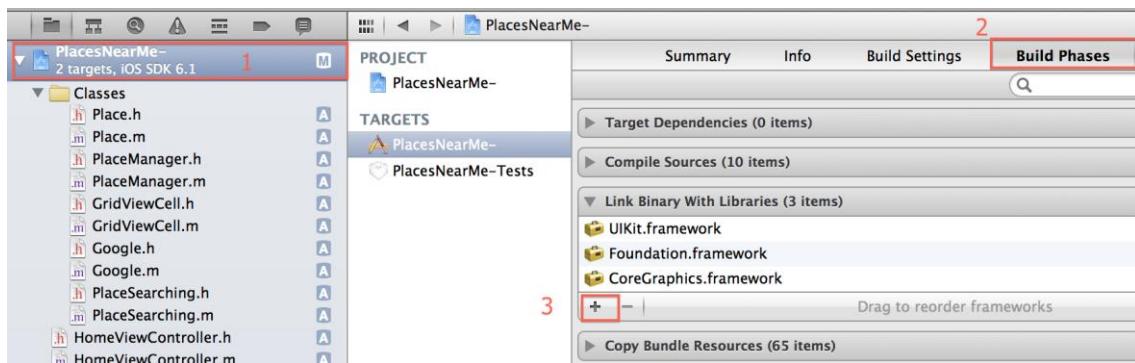
```

8
9 #import <UIKit/UIKit.h>
10
11 @interface MAPViewController : UIViewController
12 {
13
14     IBOutlet MKMapView *myMapView;
15     IBOutlet UIButton *btnExitFullScreen;
16     IBOutlet UIToolbar *toolBar;
17 }
18
19 - (IBAction)touchBtnGPS:(id)sender;
20 - (IBAction)touchBtnMap:(id)sender;
21 - (IBAction)touchBtnFullScreen:(id)sender;
22 @end

```

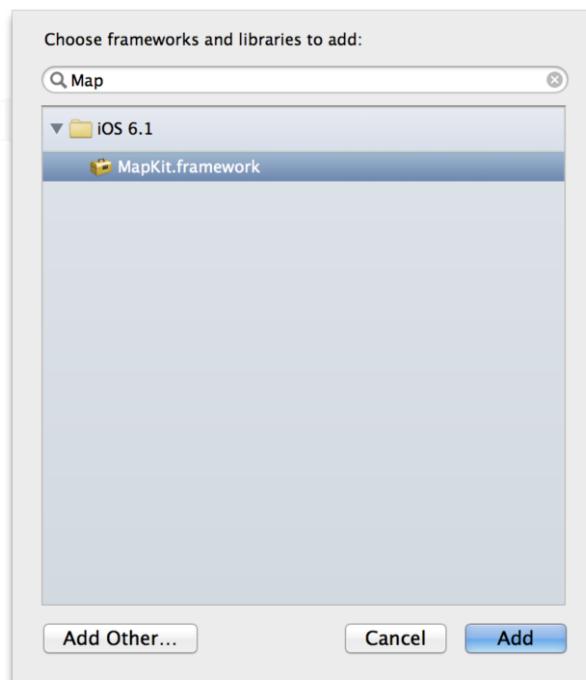
**Hình 6.253 Kết quả**

- Tiến hành import “**MapKit.h**” bằng cách click vào project → Tab “**Buil Phases**”  
 → Tab “**Link Binary With Libraries**”



**Hình 6.254 Thêm Framework**

- Click dấu “+” xuất hiện cửa sổ cho phép thêm frameworks và libraries → thêm framework “**MapKit.framework**”



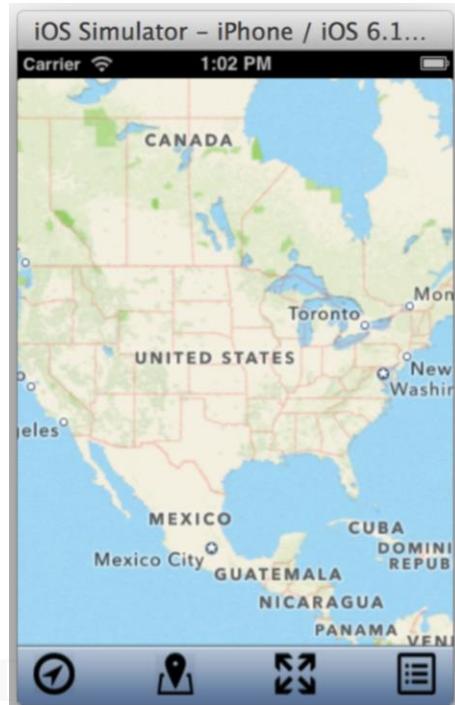
**Hình 6.255 Thêm MapKit**

- import thư viện “**MapKit.h**” vào file “**MapViewController**” sẽ không còn lỗi đó.

```
#import <UIKit/UIKit.h>
```

```
#import <MapKit/MapKit.h>
```

→ Hoàn tất quá trình chuẩn bị giao diện và Map các đối tượng, **Cmd + R** để Run thử:

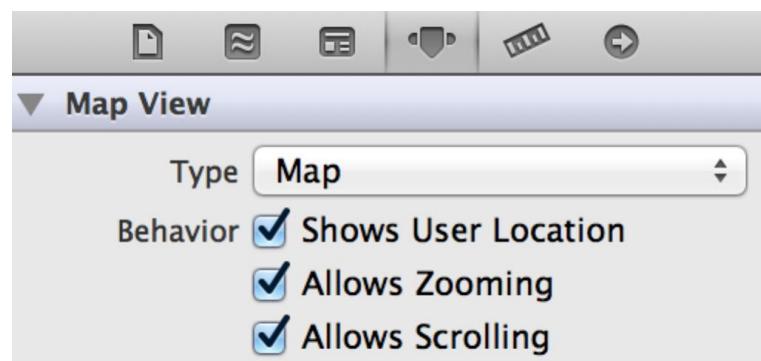


Nguyễn Anh Tú Vàng © 2013

**Hình 6.256 Kết quả chạy thử**

Hiển thị “Annotation” lên **MapView**.

→ Qua giao diện → Click chọn vào **MapView** → Check chọn thuộc tính “Shows User Location”.



**Hình 6.257 Shows User Location**

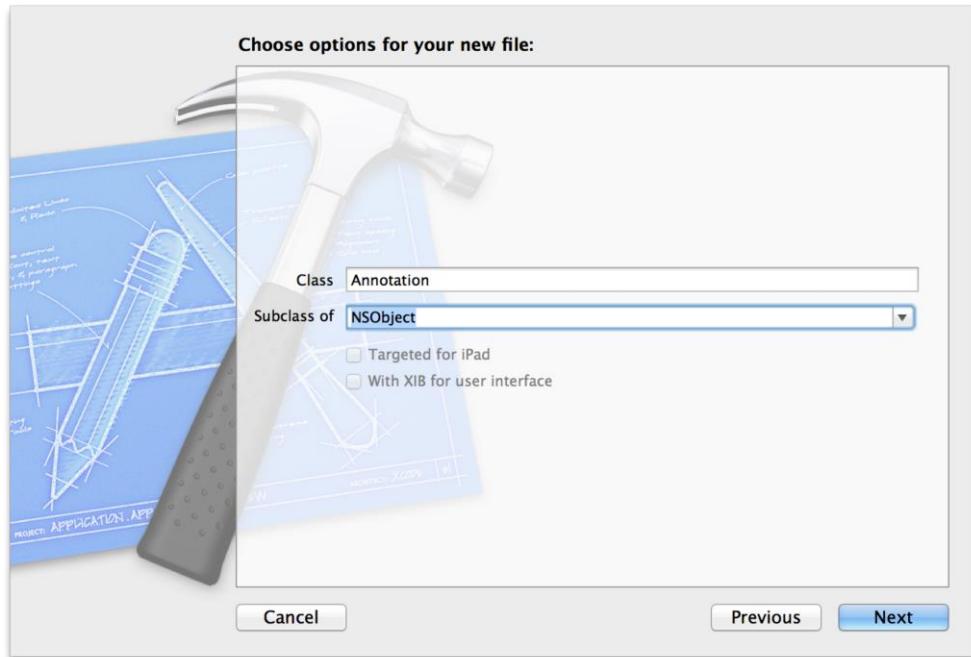
→ Chạy thử sẽ thấy xuất hiện “Annotation” màu xanh (chính là vị trí hiện tại của chúng ta nhưng do máy ảo chưa có định vị GPS nên vị trí mặc định là vị trí của Apple)



Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

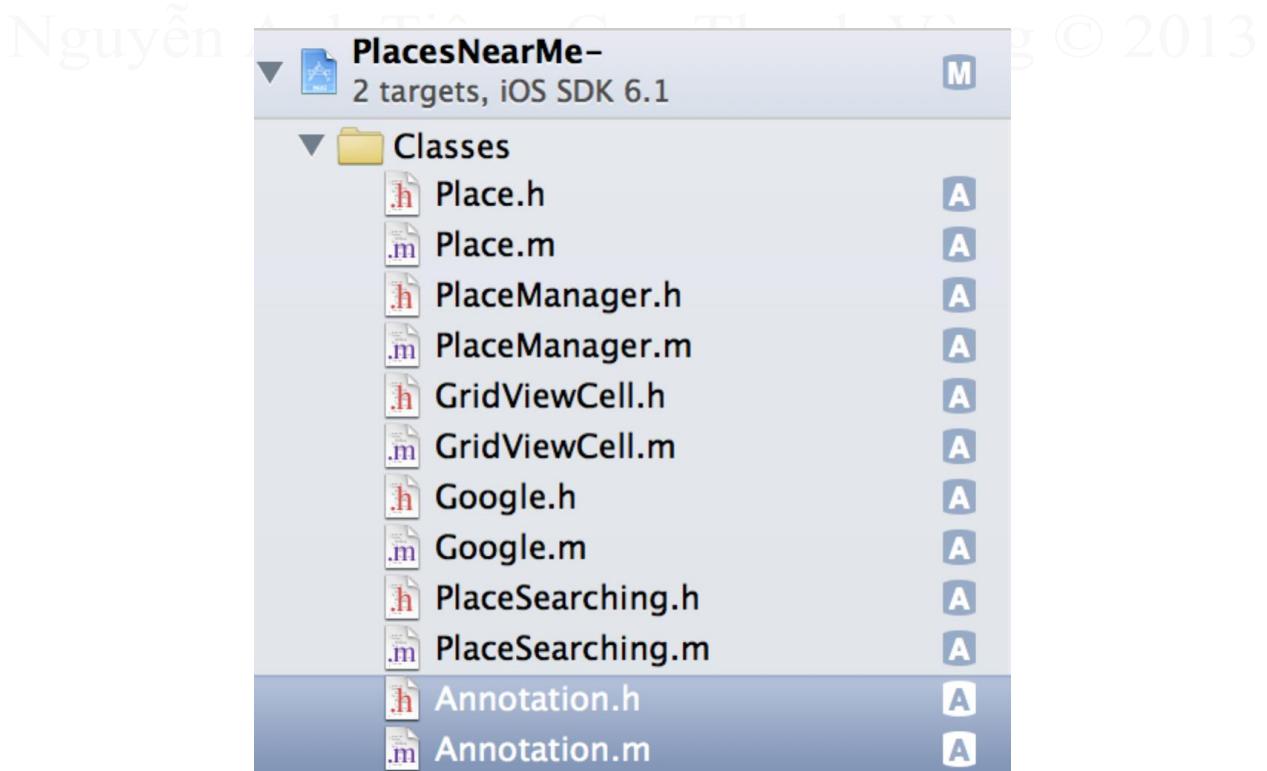
**Hình 6.258 Kết quả chạy thử**

→ Bây giờ chúng ta sẽ in (Plot) Annotation lên MapView, để làm được điều này chúng ta cần tạo 1 lớp “Annotation” → Cmd + N để tạo Object → Objective-C Class → Class: Annotation → Subclass of: NSObject.



Hình 6.259 Tạo class mới

→ Kết quả:



Hình 6.260 Kết quả tạo class

→ Mở “Annotation.h” → Khai báo các thuộc tính như sau

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface Annotation : NSObject
<MKAnnotation>

@property (nonatomic, assign) CLLocationCoordinate2D coordinate;
@property (nonatomic, copy) NSString *title;
@property (nonatomic, copy) NSString *subtitle;

@end
```

**Giải thích:**

- **Coordinate:** giữ toạ độ của Annotation
- **Title:** Tiêu đề cho Annotation
- **Subtitle:** Nội dung cho Annotation

→ Để lấy ra toạ độ hiện tại chúng ta khai báo thêm biến “locationManager” trong file “MapViewController.h” bên dưới “IBOutlet UIToolbar \*toolBar;”

```
CLLocationManager *locationManager;
```

→ Bên dưới “@interface NATMapViewController : UIViewController” thêm vào các protocol sau:

```
<MKMapViewDelegate,
CLLocationManagerDelegate,
UIActionSheetDelegate>
```

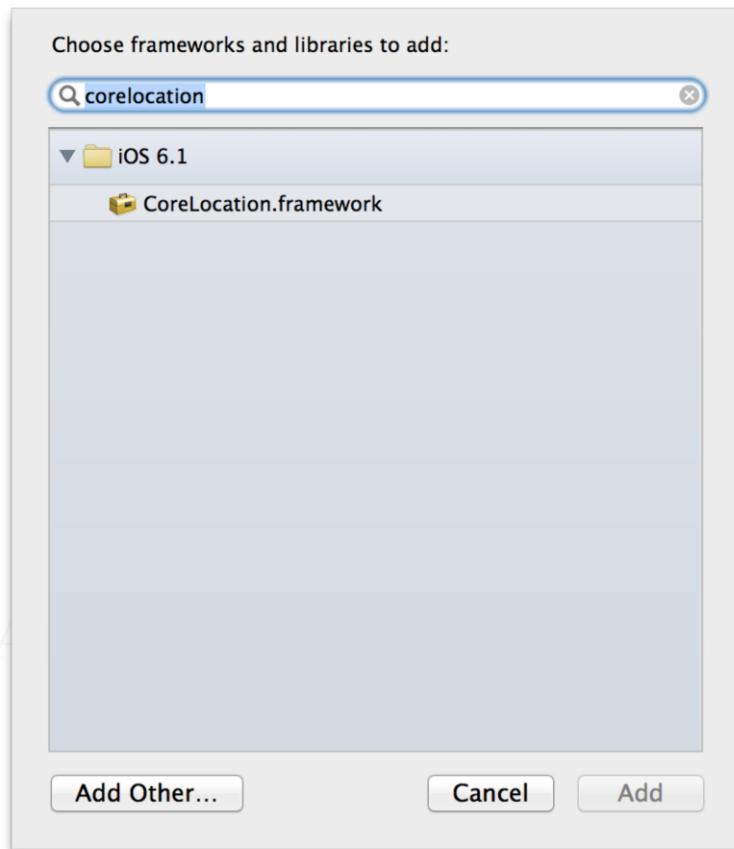
→ Kết quả file “MapViewController.h”:

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface MAPViewController : UIViewController
<CLLocationManagerDelegate>
{
    IBOutlet MKMapView *myMapView;
    IBOutlet UIButton *btnExitFullScreen;
    IBOutlet UIToolbar *toolBar;
    CLLocationManager *locationManager;
}
```

```
- (IBAction)touchBtnGPS:(id)sender;
- (IBAction)touchBtnMap:(id)sender;
- (IBAction)touchBtnFullScreen:(id)sender;
@end
```

→ Thêm framework “CoreLocation.framework”



Hình 6.261 Thêm CoreLocation Framework

→ Mở file “MapViewController.m” thêm hàm sau để xác định toạ độ hiện tại:

```
- (void)startLocation {
    if (locationManager == nil) {
        locationManager = [[CLLocationManager alloc] init];
    }

    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyBest;
}
```

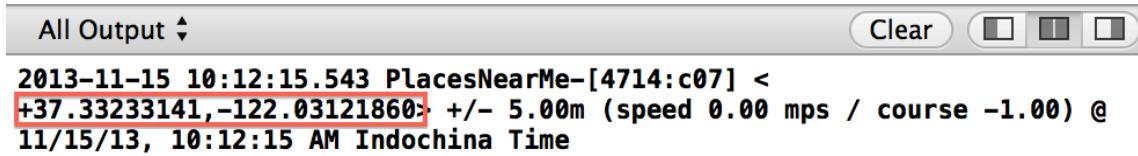
→ Tại “viewDidLoad” in thử vị trí như sau:

```
- (void)viewDidLoad
```

```
{
    [super viewDidLoad];

    [self startLocation];
    NSLog(@"%@",locationManager.location);
}
```

→ Kết quả:



The screenshot shows the Xcode Output window with the title "All Output". The log output is as follows:

```
2013-11-15 10:12:15.543 PlacesNearMe-[4714:c07] <
+37.33233141,-122.03121860> +/- 5.00m (speed 0.00 mps / course -1.00) @
11/15/13, 10:12:15 AM Indochina Time
```

**Hình 6.262 Kết quả in thử**

→ Dựa vào tọa độ hiện tại “**37.33233141, -122.03121860**”, in ra 1 Annotation ở một vị trí nào đó gần với tọa độ hiện tại.

**VD:** 37.43233141, -122.13121860. Mở file “**MapViewController.m**” Import lớp “**Annotation**” đã tạo trước đó:

```
#import "Annotation.h"
```

→ Thêm hàm sau để để in Annotation lên MapView:

```
- (void)addAnnotation:(NSString *)latitude
    longitude:(NSString *)longitude
    title:(NSString *)title
    subtitle:(NSString *)subtitle {

    //Create a coordinate
    CLLocationCoordinate2D otherCurrent;
    otherCurrent.latitude = [latitude floatValue];
    otherCurrent.longitude = [longitude floatValue];

    //Create an annotation
    Annotation *myAnnotation = [[Annotation alloc] init];
    myAnnotation.coordinate = otherCurrent;
    myAnnotation.title = title;
    myAnnotation.subtitle = subtitle;

    //add an annotation
    [myMapView addAnnotation:myAnnotation];
}
```

→ Chạy thử bằng cách sửa lại hàm “**viewDidLoad**” như sau:

```

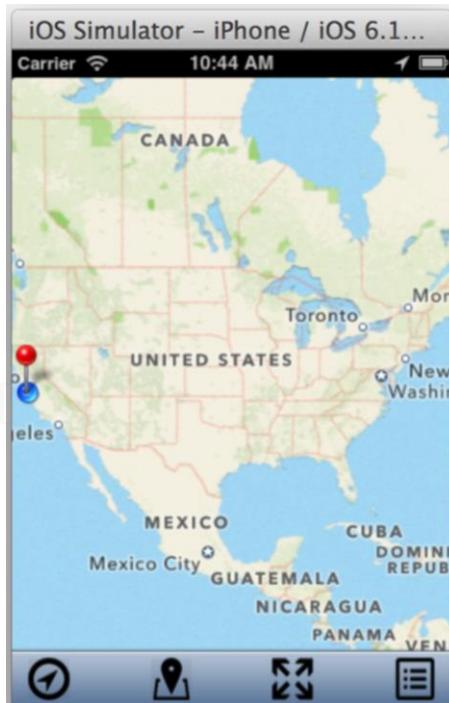
- (void)viewDidLoad
{
    [super viewDidLoad];

    [self startLocation];

    [self addAnnotation:@"37.43233141"
        longitude:@"-122.03121860"
        title:@"Nguyen Anh Tiep"
        subtitle:@"47 E/10 KP.9, P.Tan Hoa, BH-DN"];
}

```

→ Kết quả:



**Hình 6.263 Kết quả chạy thử**

→ Để phóng to bản đồ tại vị trí hiện tại, thêm protocol “**MKMapViewDelegate**”.  
Mở file “**MapViewController.h**” sau protocol ”**CLLocationManagerDelegate**” thêm vào “**MKMapViewDelegate**”:

<**CLLocationManagerDelegate**, **MKMapViewDelegate**>

**Mục đích:** để gọi hàm “**didAddAnnotationViews**”, hàm này sẽ tự gọi khi một “**Annotation**” nào đó được thêm vào MapView.

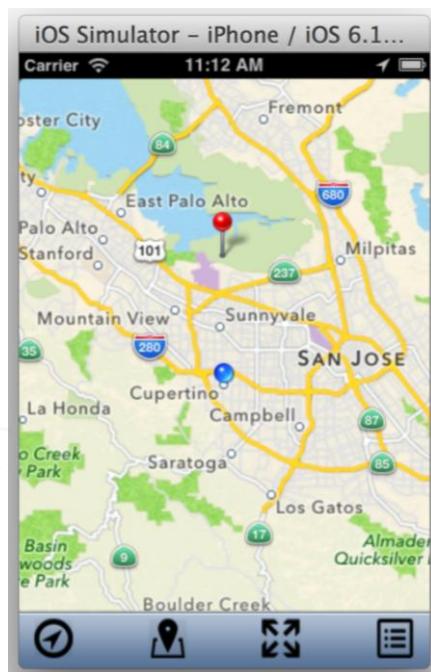
→ Mở file “**MapViewController.m**” thêm hàm “**didAddAnnotationViews**”:

```

- (void)mapView:(MKMapView *)mapView didAddAnnotationViews:(NSArray *)views {
    MKCoordinateRegion region;
    region = MKCoordinateRegionMakeWithDistance(locationManager.location.coordinate,
                                                20000, 20000);
    [mapView setRegion:region animated:YES];
}

```

→ Tại “**viewDidLoad**” thêm vào “**myMapView.delegate = self;**” → Cmd + R chạy thử, kết quả:



Nguyễn Anh Tú - Vàng © 2013

**Hình 6.264 Kết quả chạy thử**

Thực hiện chức năng “**GPS**”, chức năng này sẽ xác định vị trí hiện tại của người dùng trên MapView:

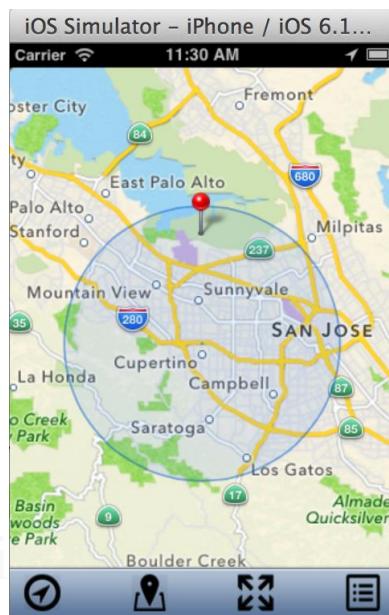


**Hình 6.265 Chức năng GPS**

→ Mở “**MapViewController.m**” → tại hàm “**touchBtnGPS**” sửa lại như sau:

```
- (IBAction)touchBtnGPS:(id)sender {  
    myMapView.showsUserLocation = NO;  
    myMapView.showsUserLocation = YES;  
}
```

→ **Cmd + R** chạy thử, kết quả:



**Hình 6.266 Chạy thử**

Thực hiện chức năng “**Map Type**”, chức năng này cho phép chúng ta chọn loại bản đồ. VD: Map, Satellite, Hybrid



**Hình 6.267 Chức năng Map Type**

→ Mở “**MapViewController.h**” bổ sung protocol “**UIActionSheetDelegate**”:

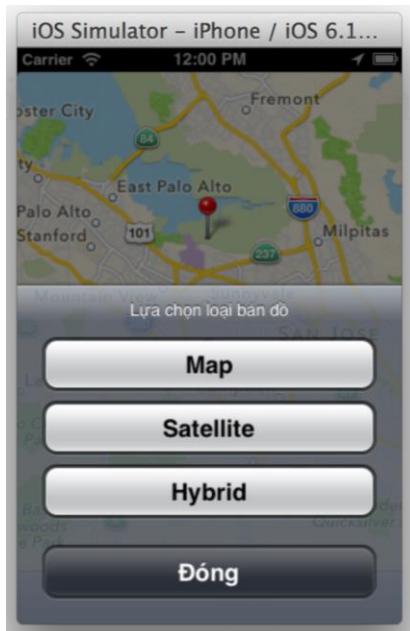
```
<CLLocationManagerDelegate, MKMapViewDelegate, UIActionSheetDelegate>
```

→ Mở “**MapViewController.m**” tại hàm “**touchBtnMap**” sửa lại như sau:

```
- (IBAction)touchBtnMap:(id)sender {
    NSString *title, *cancel;
    title = @"Lựa chọn loại bản đồ";
    cancel = @"Đóng";

    //Khởi tạo 1 actionSheet
    UIActionSheet *actionSheet = [[UIActionSheet alloc]
        initWithTitle:title
        delegate:self
        cancelButtonTitle:cancel
        destructiveButtonTitle:nil
        otherButtonTitles:@"Map", @"Satellite", @"Hybrid", nil];
    //Hiển thị 1 actionSheets
    [actionSheet showInView:self.view];
}
```

→ **Cmd + R** chạy thử, kết quả:



**Hình 6.268 Kết quả chạy thử**

→ Nhưng chưa có gì xảy ra khi click vào button, giờ chúng ta sẽ thêm hàm sau:

```
- (void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex {
    switch (buttonIndex) {
        case 0:
            myMapView.mapType = MKMapTypeStandard;
```

```

        break;
    case 1:
        myMapView.mapType = MKMapTypeSatellite;
        break;
    case 2:
        myMapView.mapType = MKMapTypeHybrid;
        break;
    default:
        break;
    }
}

```

→ Cmd + R chạy thử, kết quả:



Hình 6.269 Kết quả chạy thử

Thực hiện chức năng xem toàn màn hình bằng cách ấn thanh “Toolbar”

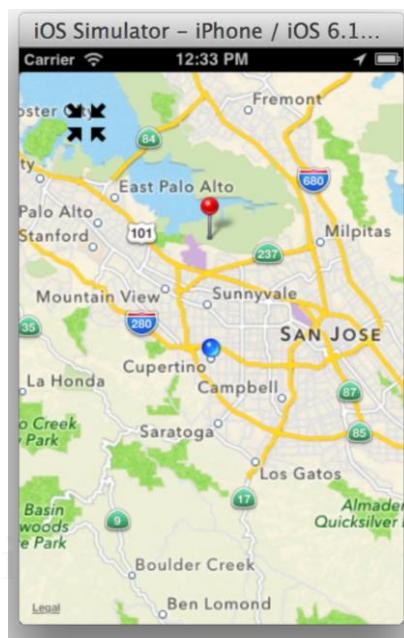


Hình 6.270 Chức năng xem toàn màn hình

→ Mở “**MapViewController.m**” tại hàm “**touchBtnFullScreen**”:

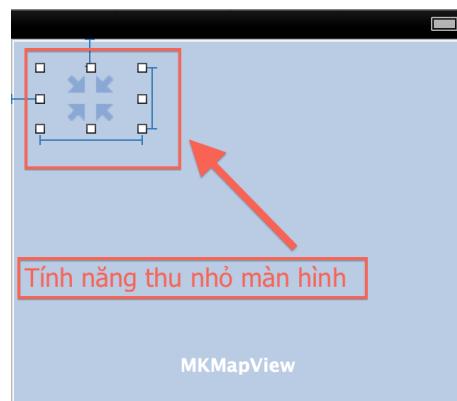
```
- (IBAction)touchBtnFullScreen:(id)sender {
    btnExitFullScreen.hidden = NO;
    toolBar.hidden = YES;
}
```

→ Chạy thử, kết quả:



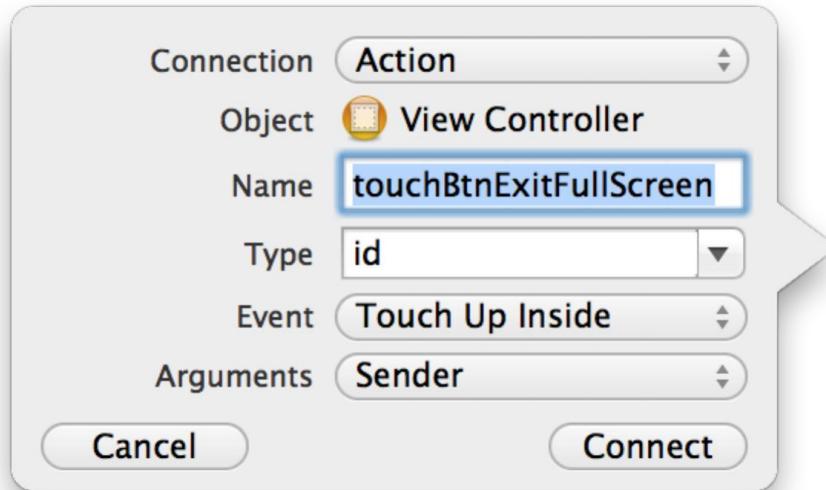
**Hình 6.271 Kết quả chạy thử**

Thực hiện tính năng thu nhỏ màn hình.



**Hình 6.272 chức năng thu nhỏ màn hình**

→ Map đối tượng trên (**btnExitFullScreen**) với các thuộc tính:



**Hình 6.273 Ánh xạ đối tượng**

→ Kết quả “MapViewController.h”:

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface MAPViewController : UIViewController
<CLLocationManagerDelegate, MKMapViewDelegate, UIActionSheetDelegate>
{
    IBOutlet MKMapView *myMapView;
    IBOutlet UIButton *btnExitFullScreen;
    IBOutlet UIToolbar *toolBar;
    CLLocationManager *locationManager;
}

- (IBAction)touchBtnGPS:(id)sender;
- (IBAction)touchBtnMap:(id)sender;
- (IBAction)touchBtnFullScreen:(id)sender;
- (IBAction)touchBtnExitFullScreen:(id)sender;

@end
```

→ Mở “MapViewController.m” tại hàm “touchBtnExitFullScreen”:

```
- (IBAction)touchBtnExitFullScreen:(id)sender {
    btnExitFullScreen.hidden = YES;
    toolBar.hidden = NO;
}
```

→ Chạy thử, kết quả:



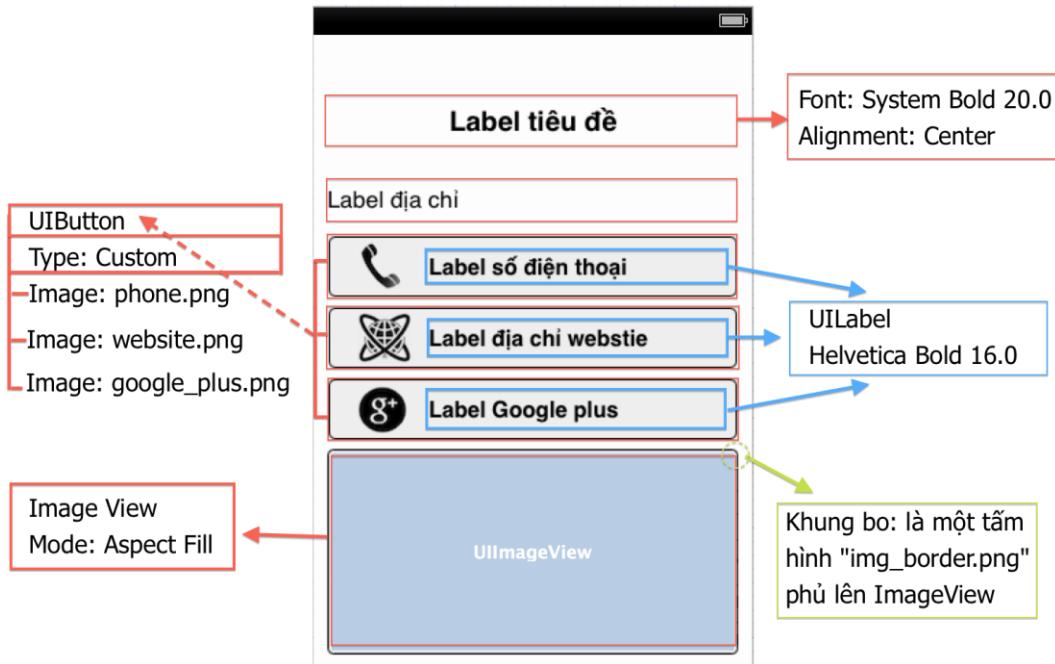
**Hình 6.274 Kết quả chạy thử**

**Bước 7 Xây dựng “PlaceDetailViewController”.**



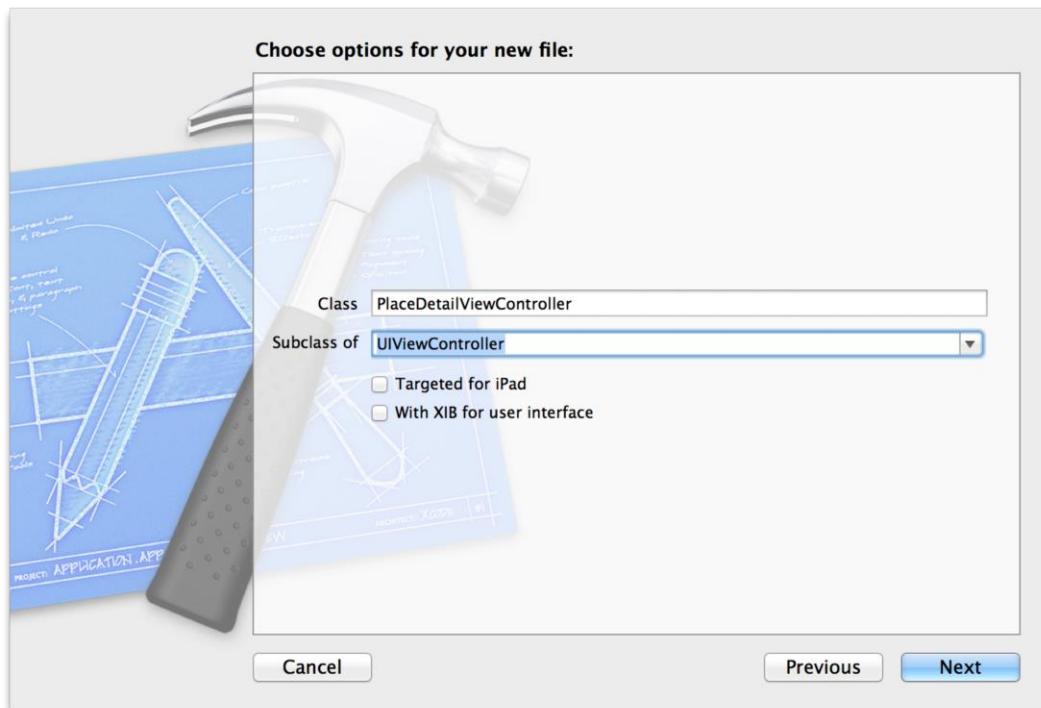
**Hình 6.275 PlaceDetail**

Kéo thả một “ViewController” mới và thiết kế giao diện như sau:



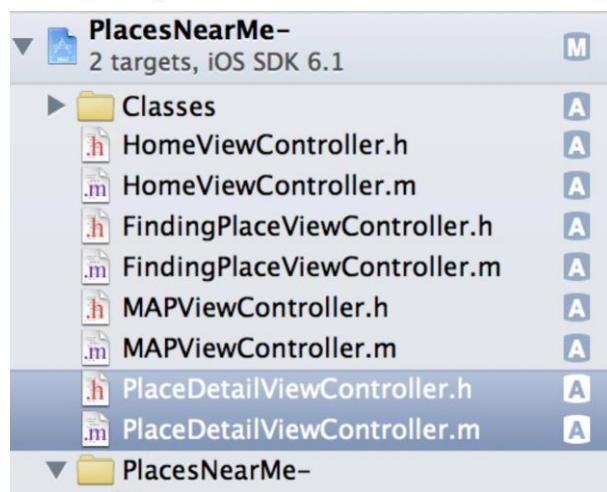
Hình 6.276 Kéo thả View Controller mới vào

Tạo một Class tên “**PlaceDetailViewController**” và trả vào “**ViewController**” vừa tạo.  
 → Cmd + N để tạo Object → Class: **PlaceDetailViewController** → Subclass of: **UIViewController**.



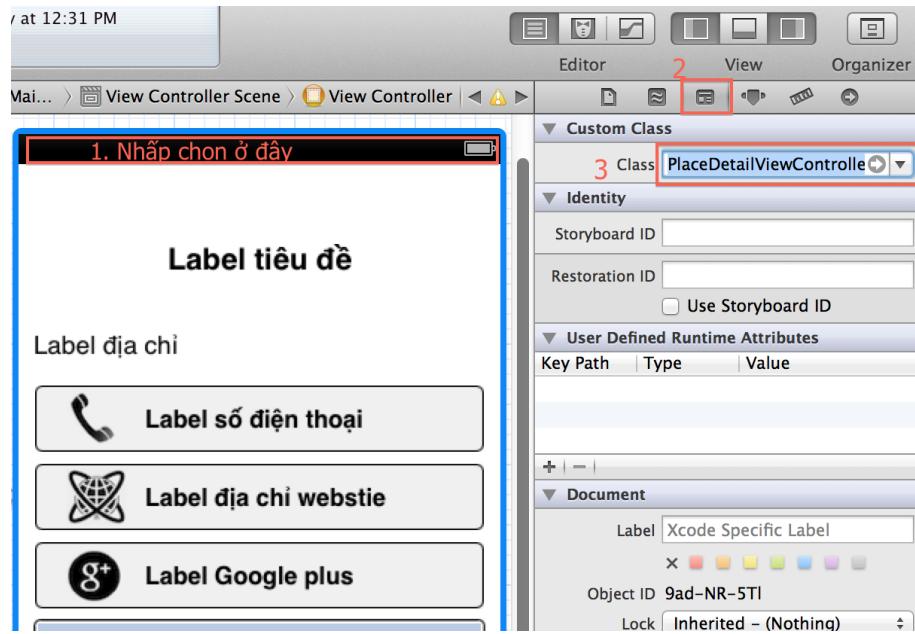
**Hình 6.277 Tạo class mới**

→ Kết quả: Nguyen Anh Tiep - Cao Thanh Vàng © 2013



**Hình 6.278 Kết quả mới tạo**

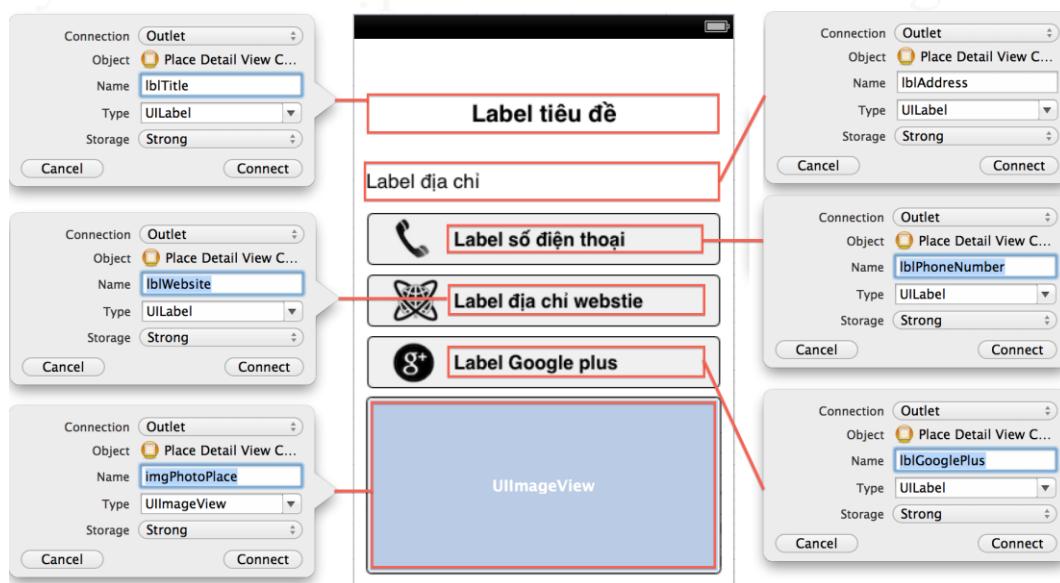
→ Qua giao diện trở vào “**ViewController**” vừa tạo



**Hình 6.279 Thêm class cho View Controller**

Map các đối tượng vào “**PlaceDetailViewController.h**”:

→ Map (Outlet) các đối tượng theo hình sau:



**Hình 6.280 Ánh xạ đối tượng**

→ Map (Action) cho đối tượng **PhoneNumber**:



**Hình 6.281 Ánh xạ PhoneNumber**

→ Kết quả “PlaceDetailViewController.h”:

```
#import <UIKit/UIKit.h>

@interface PlaceDetailViewController : UIViewController {

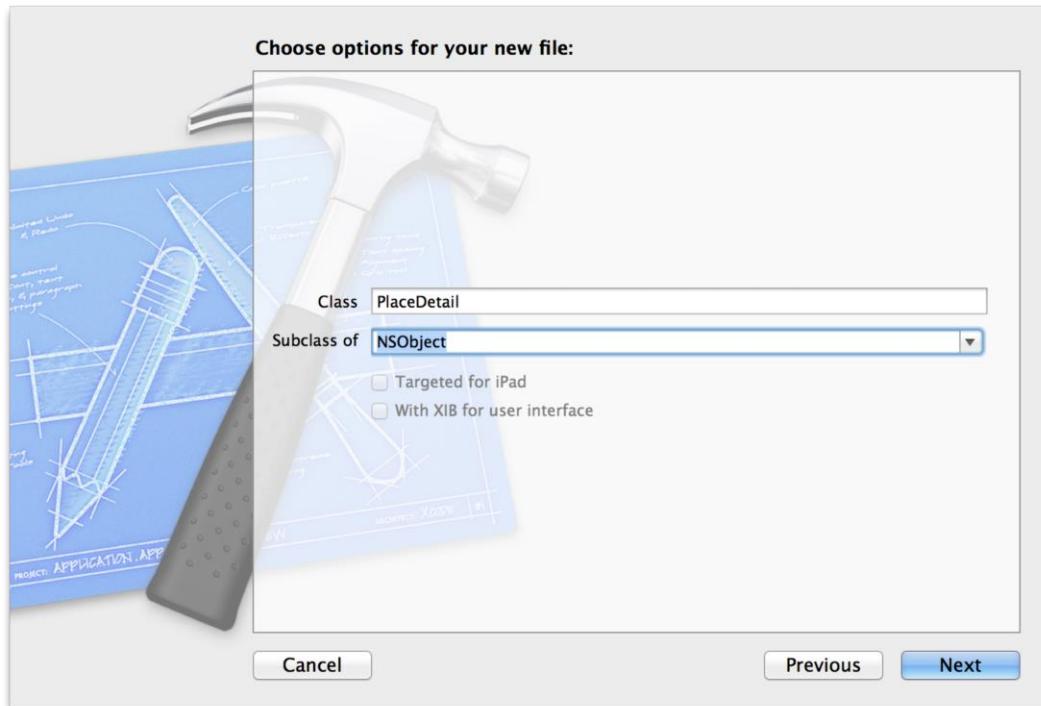
IBOutlet UILabel *lblTitle;
IBOutlet UILabel *lblAddress;
IBOutlet UILabel *lblPhoneNumber;
IBOutlet UILabel *lblWebsite;
IBOutlet UILabel *lblGooglePlus;
IBOutlet UIImageView *imgPhotoPlace;
}

- (IBAction)touchBtnPhone:(id)sender;

@end
```

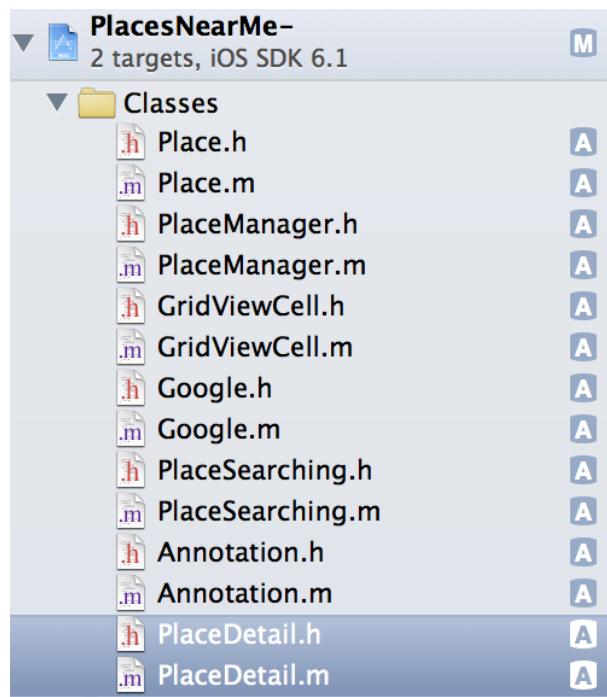
Thực hiện chức năng hiển thị text lên giao diện:

→ Cmd + N để tạo lớp “PlaceDetail” → Objective-C class → Class: **PlaceDetail** → Subclass of: **NSObject**.



**Hình 6.282 Tạo class mới**

→ Kết quả:



**Hình 6.283 Kết quả tạo class**

→ Mở “**PlaceDetail.h**” tạo các thuộc tính như sau:

```
#import <Foundation/Foundation.h>

@interface PlaceDetail : NSObject

@property (nonatomic, strong) NSString *name;
@property (nonatomic, strong) NSString *address;
@property (nonatomic, strong) NSString *phoneNumber;
@property (nonatomic, strong) NSString *website;
@property (nonatomic, strong) NSString *googlePlus;
@property (nonatomic, strong) NSArray *photos;

@end
```

→ Mở “**PlaceDetailViewController.m**” import “**PlaceDetail.h**”

```
#import "PlaceDetail.h"
```

→ Thêm phương thức “**displayPlaceDetail**” như sau:

```
- (void)displayPlaceDetail:(PlaceDetail *)pd {
    lblTitle.text = pd.name;
    lblAddress.text = pd.address;
    lblPhoneNumber.text = pd.phoneNumber;
    lblWebsite.text = pd.website;
    lblGooglePlus.text = pd.googlePlus;
    //photo sẽ thêm sau
}
```

**Giải thích:** phương thức này sẽ nhận vào 1 đối tượng “**PlaceDetail**” và hiển thị các thuộc tính trong đối tượng này lên giao diện.

→ Tại “**viewDidLoad**” tạo 1 đối tượng “**PlaceDetail**” và chạy thử.



Hình 6.284 Giao diện chạy thử

Thực hiện chức năng gọi điện thoại.

→ Mở “PlaceDetailViewController.m” tại hàm “touchBtnPhone” thêm vào như sau:

```
- (IBAction)touchBtnPhone:(id)sender {
    if ([lblPhoneNumber.text] != nil) {
        NSString *temp = [lblPhoneNumber.text stringByReplacingOccurrencesOfString:@" "
withString:@""];
        NSString *phoneStr = [NSString stringWithFormat:@"tel:prompt://%@",temp];
        [[UIApplication sharedApplication] openURL:[NSURL URLWithString:phoneStr]];
    }
}
```

**Giải thích:** phương thức này lấy số điện thoại từ “**lblPhoneNumber**” loại bỏ khoảng trắng sau đó thực hiện tính năng gọi bằng phương thức “**sharedApplication**”. Chức năng gọi chúng ta sẽ kiểm tra sau khi build ứng dụng lên iPhone thật đơn giản vì Simulator không thể gọi điện.

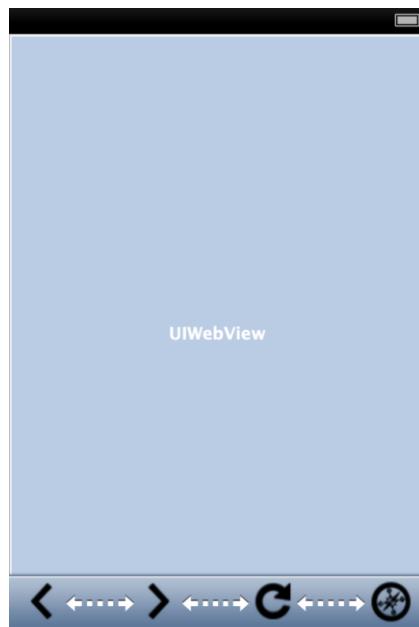
**Bước 8** Xây dựng “**WebViewController**”.



Hình 6.285 Giao diện webview

**Giới thiệu:** “**WebViewController**” có chức năng tương tự như một trình duyệt web: lùi lại (back), tiếp thô (next), làm tươi (refrest).

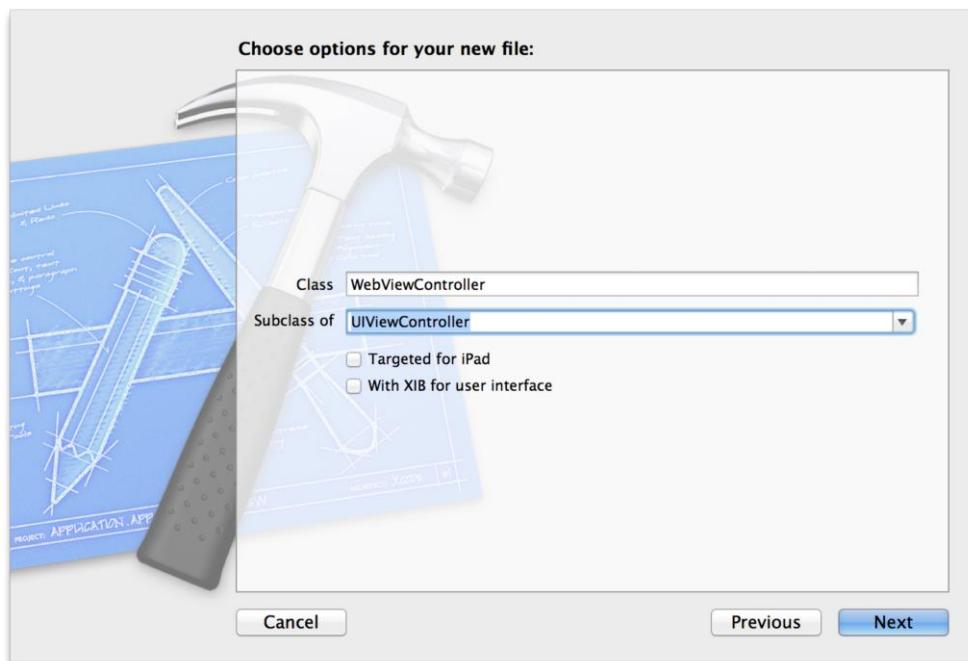
Thiết kế giao diện như sau (tương tự cách thiết kế giao diện của “**MapViewController**”).



Hình 6.286 Giao diện Webview

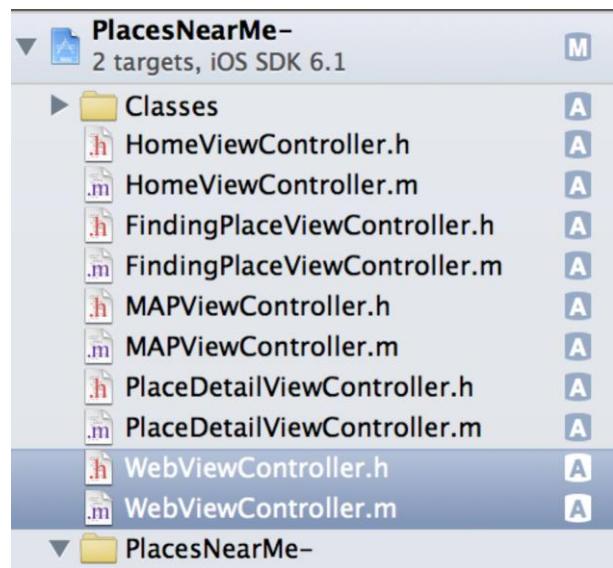
Tạo một Class “**WebViewController**” và trả vào “**ViewController**” vừa tạo:

- **Cmd + N** để tạo một object → Objective-C class → Class: **WebViewController**
- Subclass of: **UIViewController**.



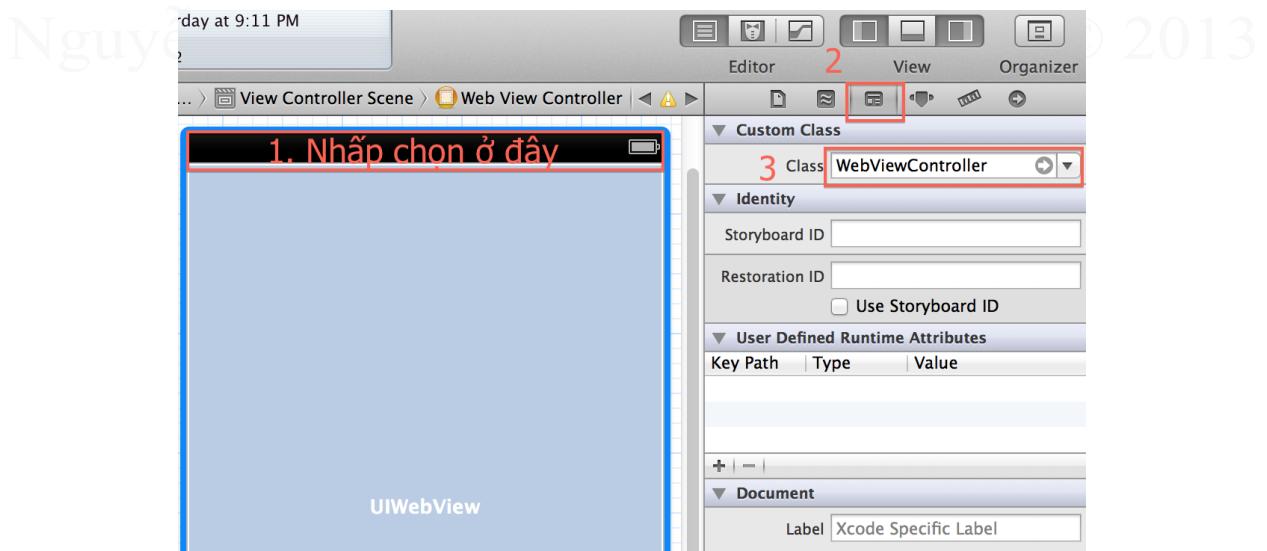
Hình 6.287 Tạo class mới

→ Kết quả:



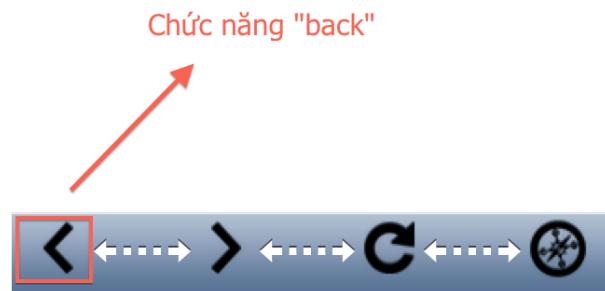
Hình 6.288 Kết quả tạo class

→ Thực hiện trỏ lớp vừa tạo vào “ViewController”.



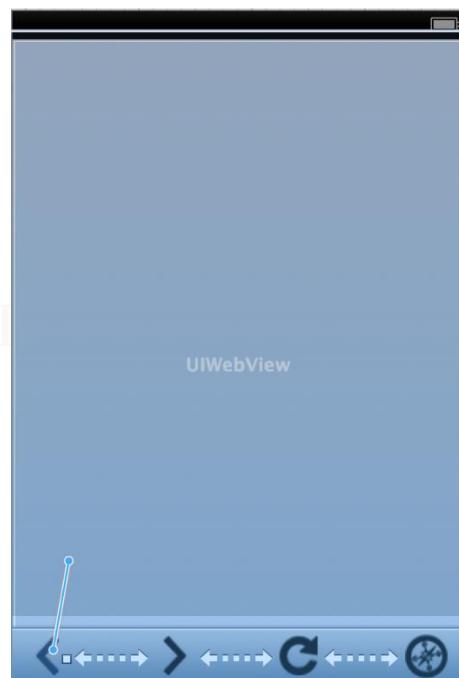
Hình 6.289 Trỏ class mới vào ViewController

Thực hiện chức năng “back”.



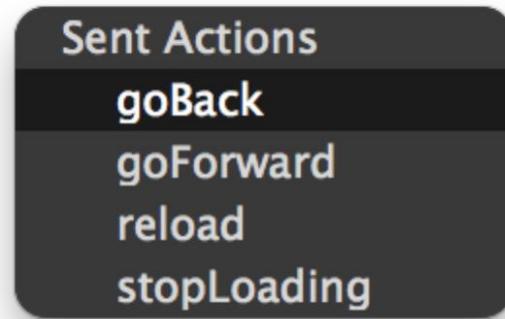
**Hình 6.290 Chức năng Back**

→ Nhấn giữ phím Ctrl và kéo thả đúi tượng bar button “Back” vào “UIWebView” như sau.



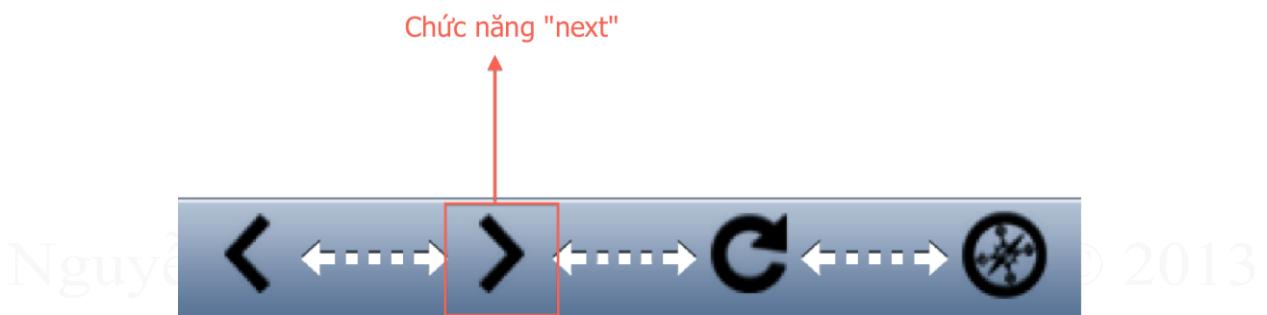
**Hình 6.291 Kéo thả button Back vào UIWebView**

→ Xuất hiện hộp thoại “Send Actions” → chọn “goBack”



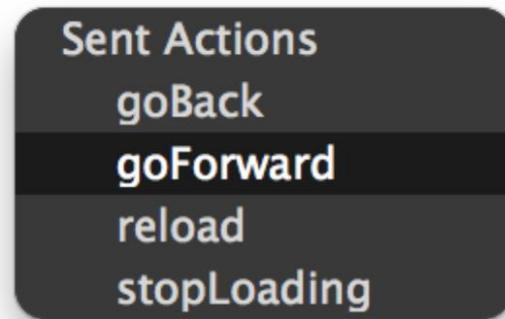
**Hình 6.292 Chọn GoBack**

Thực hiện chức năng “Next”



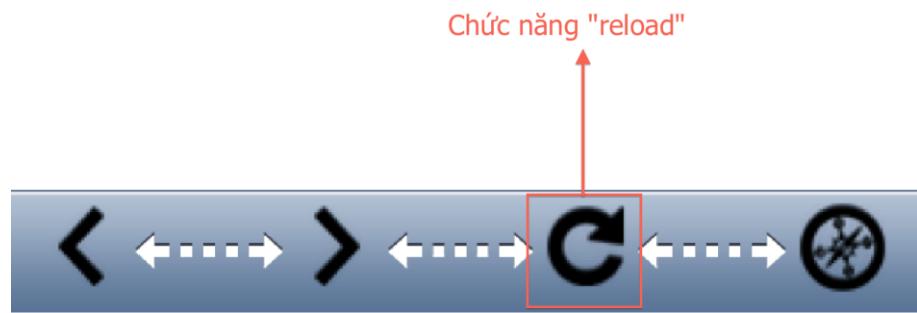
**Hình 6.293 Chức năng Next**

→ Thực hiện tương tự như vừa rồi → chọn “goForward”



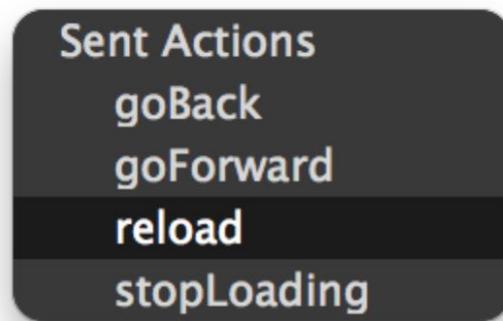
**Hình 6.294 Chọn goForward**

Thực hiện chức năng “Reload”.



**Hình 6.295 Chức năng Reload**

→ Thực hiện tương tự như trên → chọn “Reload”.



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

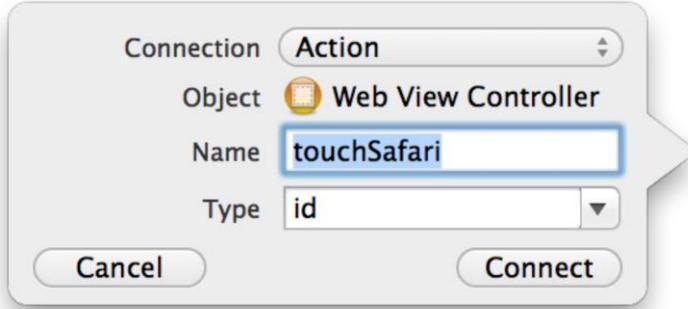
**Hình 6.296 Chọn Reload**

Thực hiện chức năng mở web với “Safari”



**Hình 6.297 Chức năng mở Web**

→ Mở file “**WebViewController**” Map (Action) cho đối tượng “**Safari**”.



**Hình 6.298 Ánh xạ đối tượng**

→ Kết quả “**WebViewController.h**”

```
#import <UIKit/UIKit.h>

@interface WebViewController : UIViewController
- (IBAction)touchSafari:(id)sender;
@end
```

→ Mở “**WebViewController.m**” bên dưới “**@implementation...**” khai báo 1 biến “**url**”  
dạng chuỗi:

```
@implementation WebViewController
NSString *url;
```

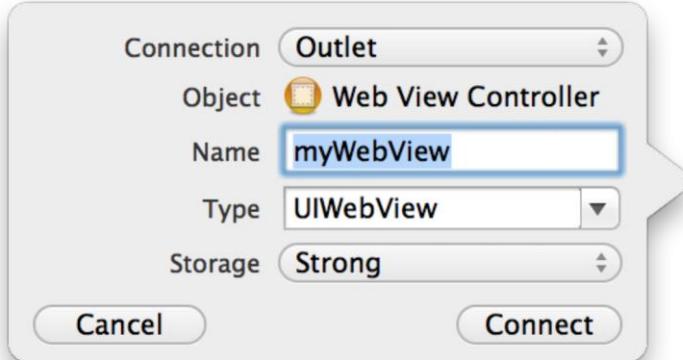
→ sửa lại phương thức “**touchSafari**” như sau:

```
- (IBAction)touchSafari:(id)sender {
    [[UIApplication sharedApplication] openURL:[NSURL URLWithString:url]];
}
```

**Giải thích:** phương thức sharedApplication được dùng để mở một chuỗi url nào đó  
như: “địa chỉ website, số điện thoại”

Thực hiện mở một “**url**” cho “**UIWebView**”:

→ Map (Outlet) cho đối tượng “**UIWebView**”



**Hình 6.299 Ánh xạ đối tượng**

→ Kết quả “WebViewController.h”

```
#import <UIKit/UIKit.h>

@interface WebViewController : UIViewController {
    IBOutlet UIWebView *myWebView;
}

- (IBAction)touchSafari:(id)sender;
@end
```

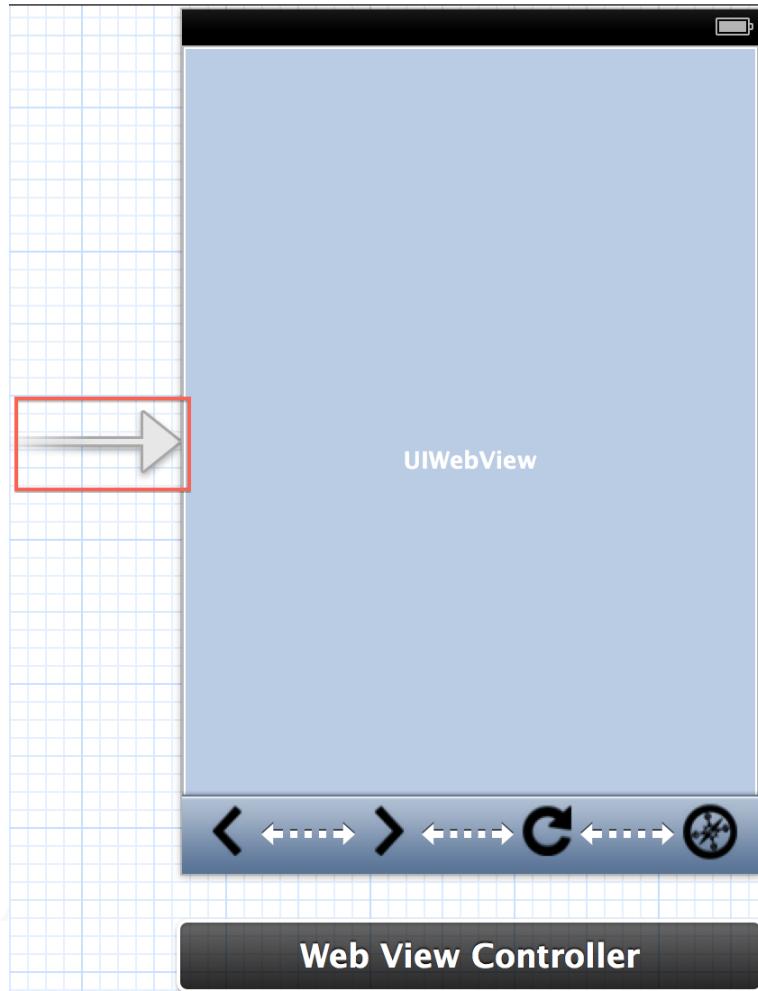
→ Mở “**WebViewController.m**” thêm vào phương thức:

```
- (void)loadURL:(NSString *)urlStr {
    NSURL *url = [NSURL URLWithString:urlStr];
    NSURLRequest *myRequest = [NSURLRequest requestWithURL:url];
    [myWebView loadRequest:myRequest];
}
```

→ Tại hàm “**viewDidLoad**” sửa lại như sau:

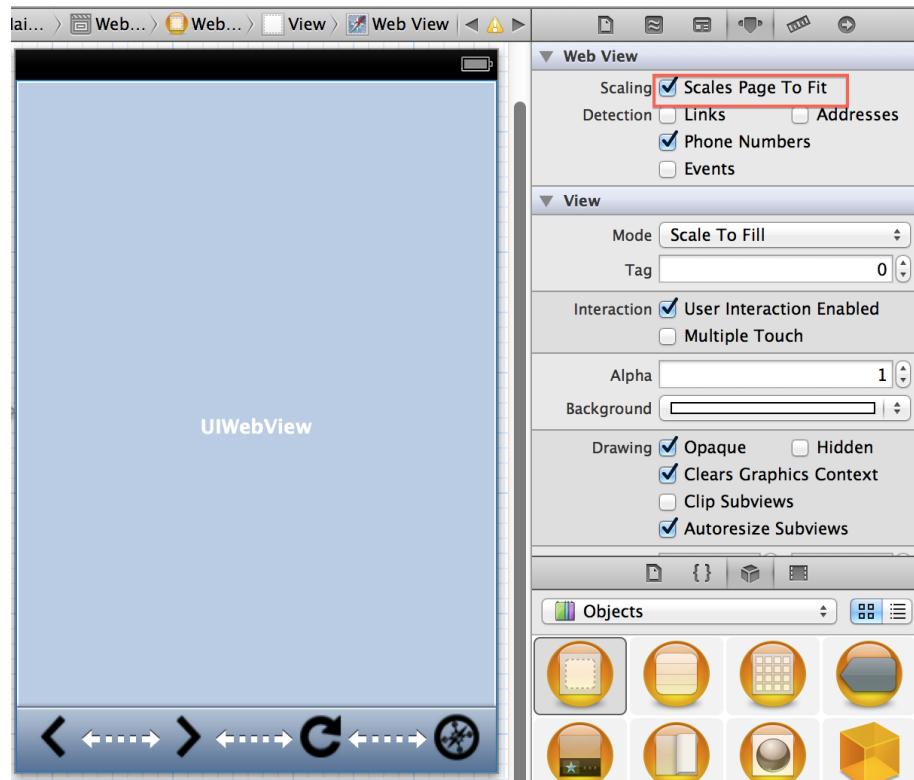
```
- (void)viewDidLoad
{
    [super viewDidLoad];
    url = @"http://lhu.edu.vn";
    [self loadURL:url];
}
```

→ Qua giao diện kéo thả mũi tên từ “**Place Detail View Controller**” sang “**Web View Controller**” để chạy bằng “**Web View Controller**”.



**Hình 6.300 Đặt chạy thử Web View Controller**

→ Chỉnh lại thuộc tính “Scaling” cho “UIWebView” là “Scales Page To Fit”.



**Hình 6.301 Tùy chỉnh thuộc tính Scale**

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
 → Cmd + R chạy thử, kết quả:



**Hình 6.302 Kết quả chạy thử**

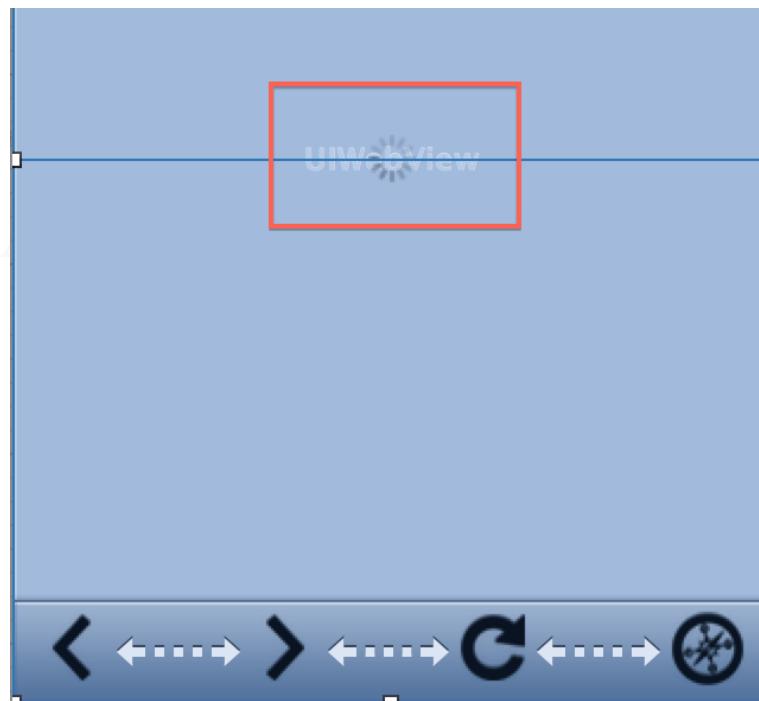
Thực hiện “Activity Indicator View” hay đơn giản là một Spinner



**Hình 6.303 Activity Indicator View**

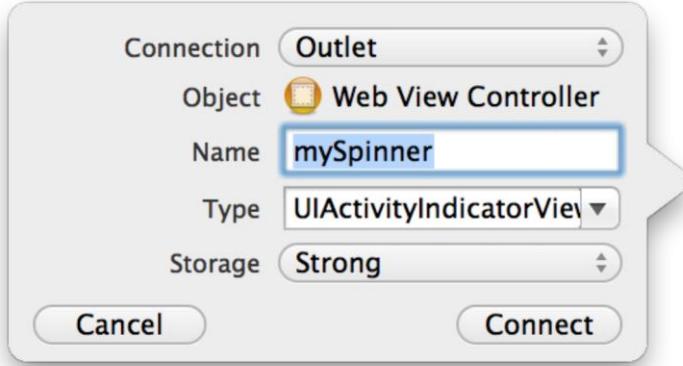
**Giới thiệu:** Spinner dùng để thông báo cho người dùng biết chương trình đó vẫn đang chạy.

→ Kéo thả “Activity Indicator View” vào “UIWebView Controller”.



**Hình 6.304 Kéo thả Activity Indicator**

→ Map (Outlet) “Activity Indicator View” vào file “MapViewController.h”



**Hình 6.305 Chọn loại kết nối**

→ Mở “**MapViewController.h**” → Thêm protocol “**UIWebViewDelegate**”, kết quả:

```
#import <UIKit/UIKit.h>

@interface WebViewController : UIViewController
<UIWebViewDelegate>
{
    IBOutlet UIWebView *myWebView;
    IBOutlet UIActivityIndicatorView *mySpinner;
}
- (IBAction)touchSafari:(id)sender;
@end
```

→ Mở “**MapViewController.m**” thêm các phương thức sau

```
- (void)webViewDidStartLoad:(UIWebView *)webView {
    //Spinner bắt đầu quay
    [mySpinner startAnimating];
}

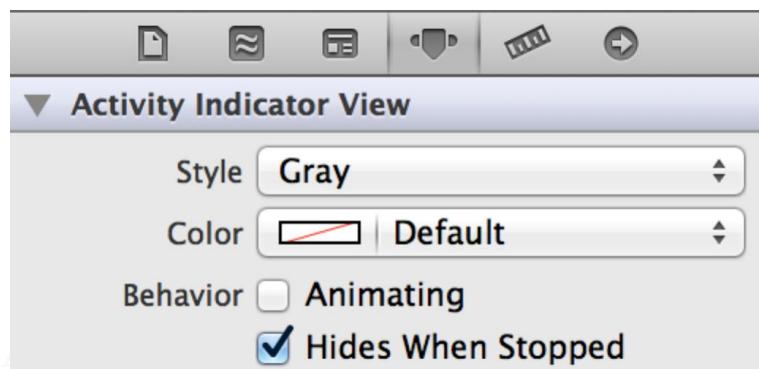
- (void)webViewDidFinishLoad:(UIWebView *)webView
{
    //Ngưng spinner
    [mySpinner stopAnimating];
    //Gán tiêu đề vào "UIWebViewController"
    NSString* title = [webView stringByEvaluatingJavaScriptFromString: @"document.title"];
    self.navigationItem.title = title;
};

- (void)webView:(UIWebView *)webView didFailLoadWithError:(NSError *)error {
    [mySpinner stopAnimating];
}
```

**Giải thích:** tất cả các phương thức vừa thêm vào đều được “UIWebView” tự động gọi khi:

- **webViewDidStartLoad**: gọi khi UIWebView bắt đầu load một url nào đó.
- **webViewDidFinishLoad**: gọi khi UIWebView đã load xong một url nào đó.
- **didFailLoadWithError**: gọi khi UIWebView load không thành công một url nào đó.

→ Để “Spinner” tự ẩn hiện chúng ta cần thiết lập thuộc tính “**Hides When Stopped**” cho Spinner.



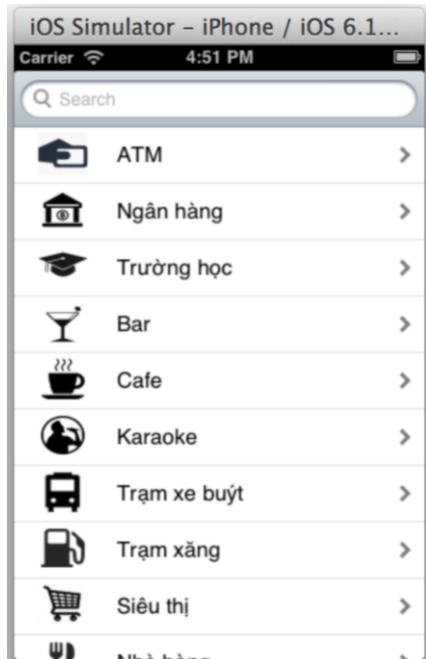
**Hình 6.306 Hide When Stopped**

→ Mở “**WebViewController.m**” bổ sung thêm “**myWebView.delegate = self;**” và chạy thử, kết quả:



**Hình 6.307 Chạy thử ứng dụng**

**Bước 9:** Xây dựng “**SearchingHomeViewController**”.

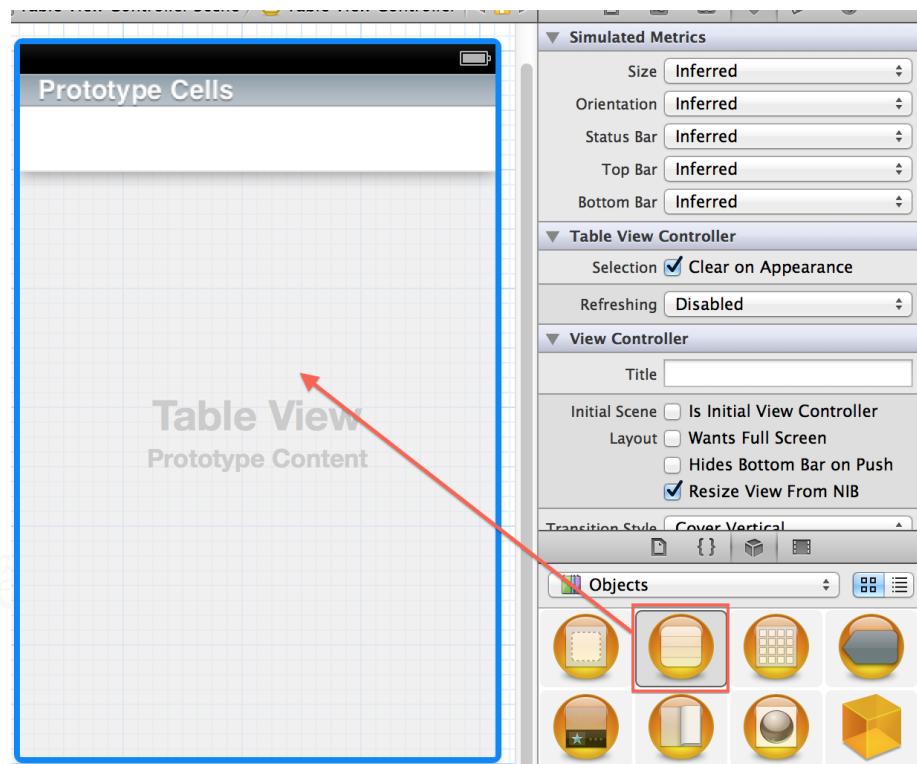


**Hình 6.308 Searching Home View Controller**

**Giới thiệu:** “**SearchingHomeViewController**” giúp người dùng tìm kiếm các địa điểm nhanh hơn.

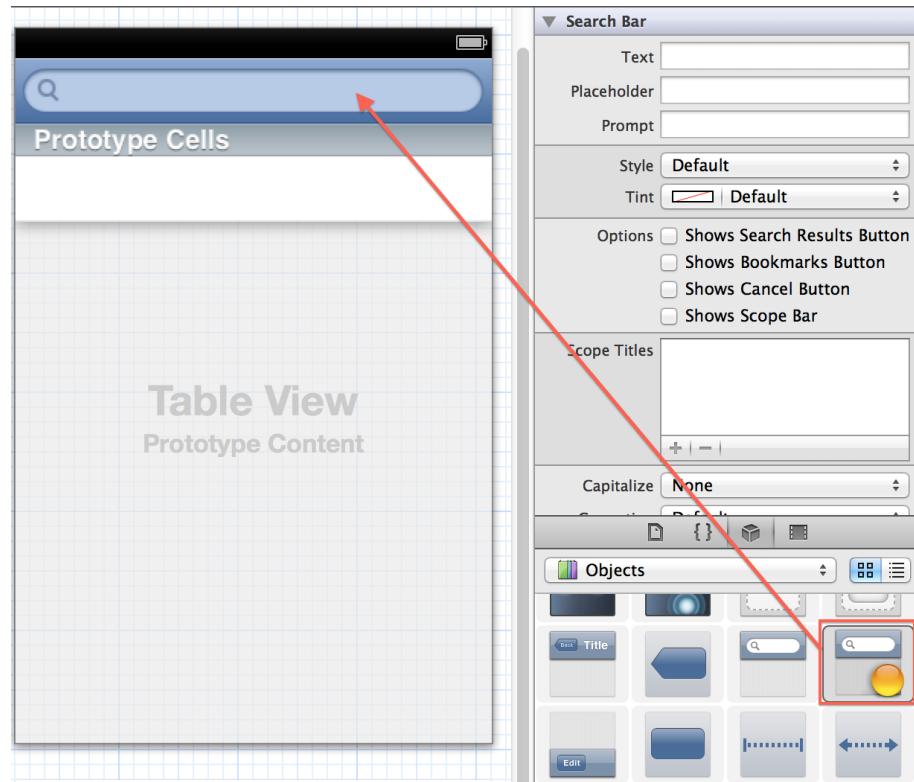
Thiết kế giao diện:

→ Kéo thả “**TableViewController**”



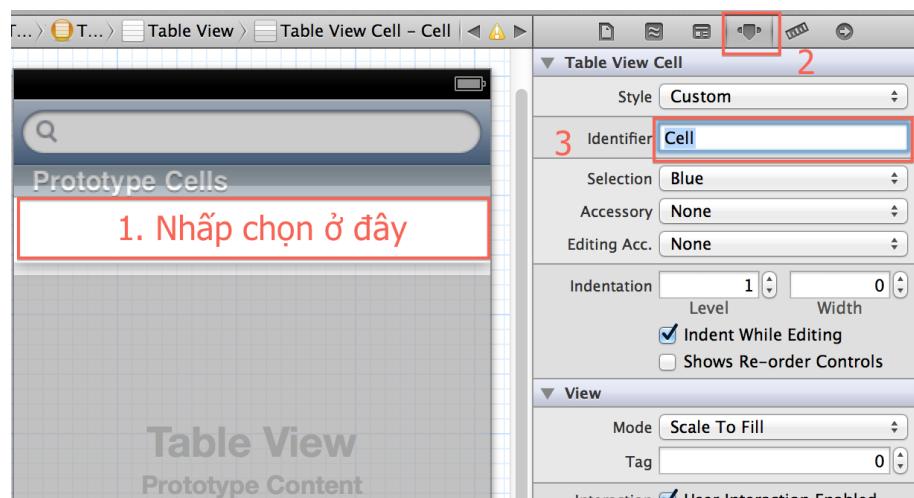
**Hình 6.309 Kéo thả Table View**

→ Thêm “**Search Bar and Search Diplay Controller**”.



**Hình 6.310 Kéo thả Search bar**

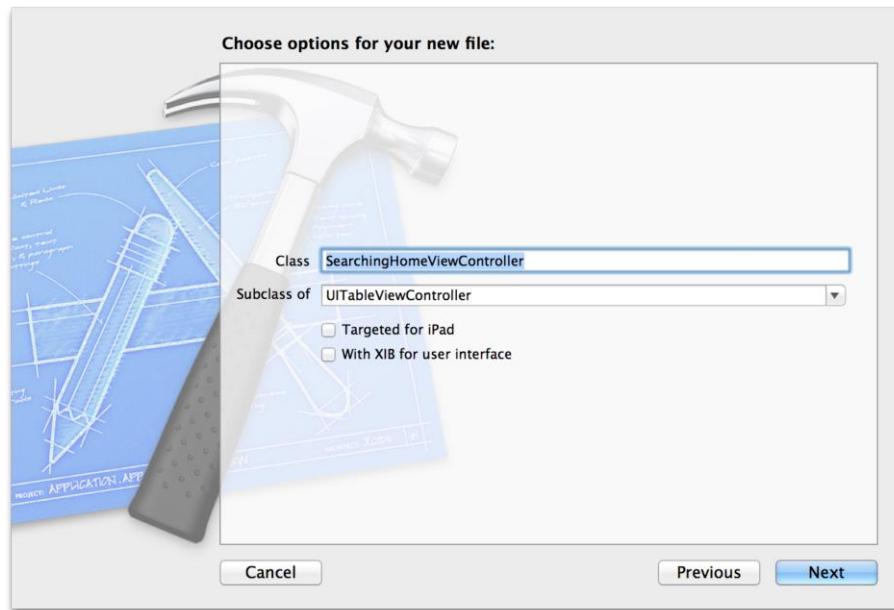
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
→ Đặt tên “Cell”



**Hình 6.311 Đặt tên Cell**

Tạo lớp “**SearchingHoveViewController**” và trả vào “**Table View Controller**”.

→ Cmd + N để tạo đối tượng → Objective-C class → Class:  
**SearchingHomeViewController** → Subclass of: **UITableViewController**.



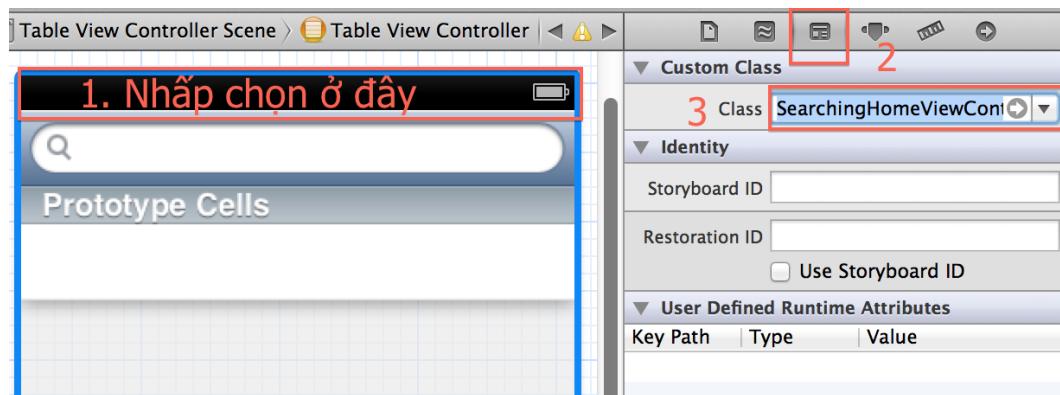
**Hình 6.312 Tạo class mới**

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
→ Kết quả:



**Hình 6.313 Kết quả sau khi tạo class**

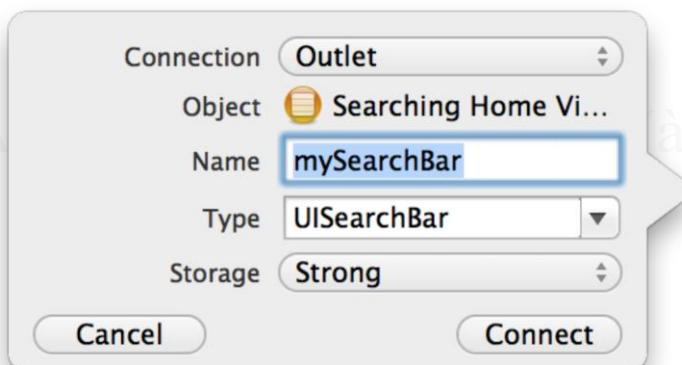
→ Qua giao diện trở vào “Table View Controller”.



**Hình 6.314 Trỏ class vào Table View**

Hiển thị dữ liệu lên giao diện:

→ Map (Outlet) đối tượng “Search Bar and Search Diplay Controller” với tên “myearchBar”



**Hình 6.315 Ánh xạ đối tượng**

→ Kết quả.

```
#import <UIKit/UIKit.h>

@interface SearchingHomeViewController : UITableViewController {
    IBOutlet UISearchBar *myearchBar;
}

@end
```

→ Mở “**SearchingHomeViewController.m**” dưới “**@implementation...**” khai báo 2 mảng.

```
@implementation SearchingHomeViewController
NSMutableArray *arrPlaces;
NSMutableArray *arrSearchingResults;
```

→ Import thêm 2 lớp:

```
#import "Place.h"
#import "PlaceManager.h"
```

→ Sửa lại 2 phương thức “**numberOfSectionsInTableView**” và “**numberOfRowsInSection**”

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return arrPlaces.count;
}
```

→ Sửa lại phương thức “**cellForRowAtIndexPath**” như sau:

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

    cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
    cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
    cell.selectionStyle = UITableViewCellAccessoryNone;

    cell.clearsContextBeforeDrawing = YES;
    int row = indexPath.row;
    Place *place;
    if (tableView == self.tableView) {
        place = arrPlaces[row];
    } else {
        place = arrSearchingResults[row];
    }

    // Configure the cell...
    UILabel *lblText = [[UILabel alloc] initWithFrame:CGRectMake(80, 11, 230, 20)];

    lblText.text = [place titleVi];
    [cell.contentView addSubview:lblText];
```

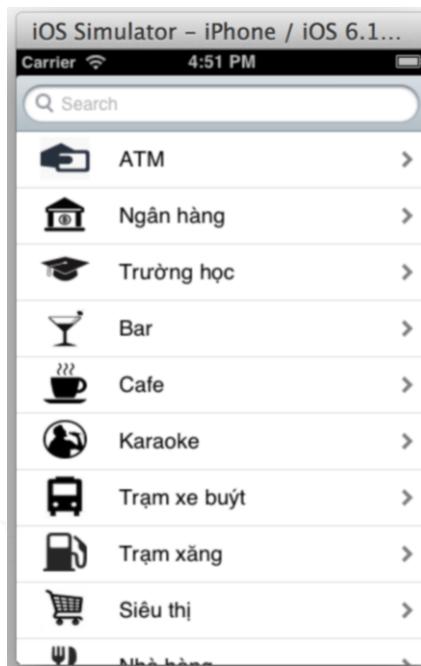
```

UIImageView *imgImage = [[UIImageView alloc] initWithFrame:CGRectMake(12, 4, 52, 40)];
imgImage.image = [UIImage imageNamed:[place imageName]];
[cell.contentView addSubview:imgImage];

return cell;
}

```

→ Chạy thử:



**Hình 6.316 Kết quả chạy thử**

Thực hiện tìm kiếm:

→ Mở “**SearchingHomeViewController.m**” Thêm phương thức  
“**searchThroughData**”

```

- (void)searchThroughData {
    arrSearchingResults = nil;
    NSPredicate *resultsPredicate;

    resultsPredicate = [NSPredicate predicateWithFormat:@"SELF.titleVi contains [search] %@", mySearchBar.text];

    arrSearchingResults = [[arrPlaces filteredArrayUsingPredicate:resultsPredicate] mutableCopy];
}

```

**Giải thích:** phương thức này sử dụng “**filteredArrayUsingPredicate**” để tìm kiếm nội dung trong 1 mảng (**arrPlaces**). Sau khi tìm xong sẽ trả kết quả về cho mảng **arrSearchingResults**.

→ Thêm phương thức “**textDidChange**”

```
- (void)searchBar:(UISearchBar *)searchBar textDidChange:(NSString *)searchText {
    [self searchThroughData];
}
```

**Giải thích:** đây là phương thức có sẵn của đối tượng “**searchBar**”, phương thức này được gọi khi nội dung trong “**searchBar**” thay đổi và mỗi khi thay đổi như vậy chúng ta sẽ gọi tới phương thức “**searchThroughData**” để tìm kiếm kết quả.

→ Sửa lại phương thức “**numberOfRowsInSection**”

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    if (tableView == self.tableView) {
        return arrPlaces.count;
    } else {
        [self searchThroughData];
        return arrSearchingResults.count;
    }
}
```

**Giải thích:** phương thức này cho biết số dòng trong một table, chúng ta sẽ phân ra 2 trường hợp, trường hợp mặc định tức khi chưa sử dụng “**searchBar**” để tìm kiếm thì số dòng trong table chính là số phần tử trong mảng “**arrSearchingResults**”, trường hợp còn lại khi sử dụng tìm kiếm (searchBar) thì số dòng trong table chính là số phần tử trong mảng “**arrSearchingResults**” cũng chính là số kết quả tìm được.

→ Chạy thử, kết quả:

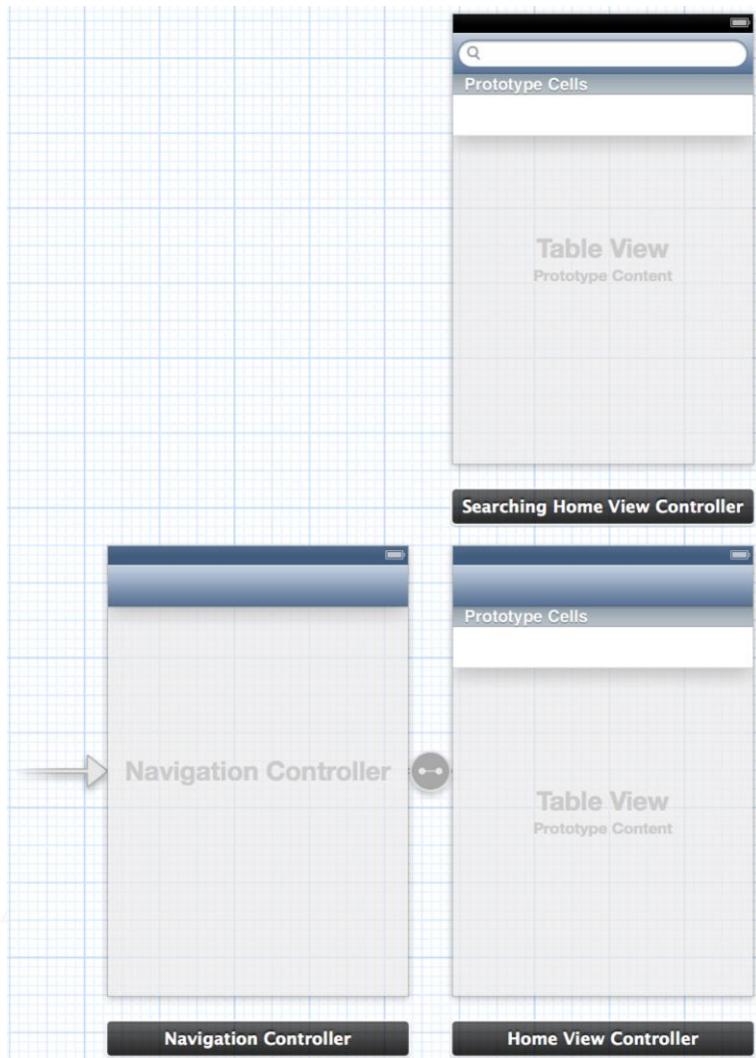


**Hình 6.317** Kết quả chạy thử

**Bước 10:** Liên kết các “ViewController”:

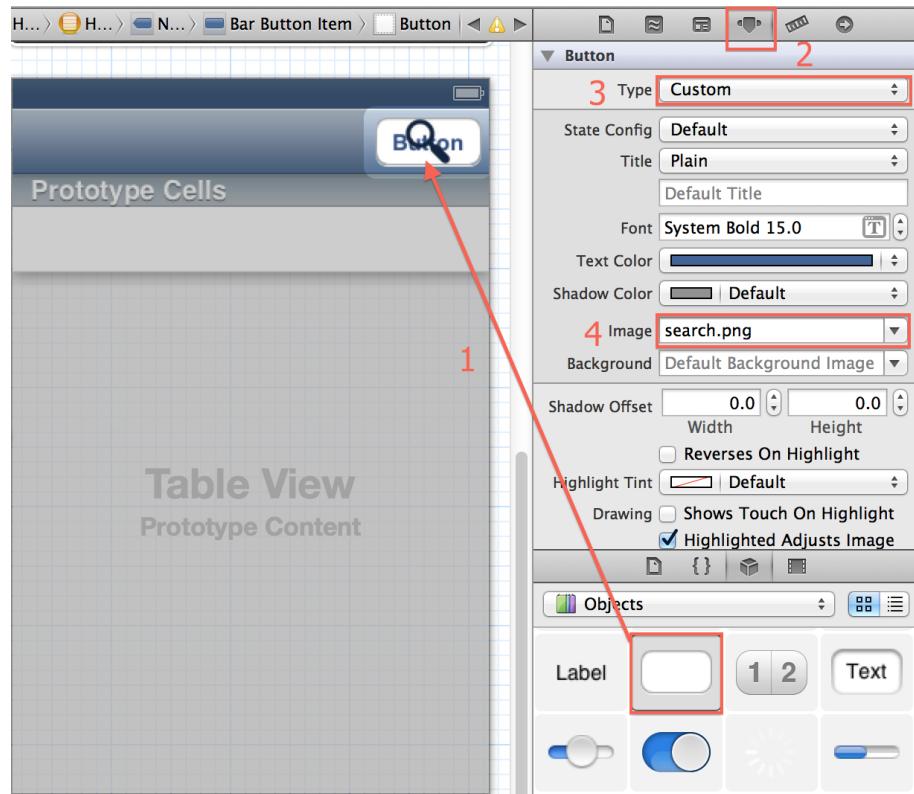
Ở các phần trước bạn đã thiết kế các “ViewController” như: **HomeViewController**, **SearchingHomeViewController**, **FindingPlaceViewController**, **MapViewController**, **PlaceDetailViewController**, **WebViewController**. Các “ViewController” này hoàn toàn có thể chạy độc lập và trong phần này nhiệm vụ của bạn là liên kết tất cả các “ViewController” lại với nhau.

Liên kết giữa “**Home View Controller**” và “**Searching Home View Controller**”:



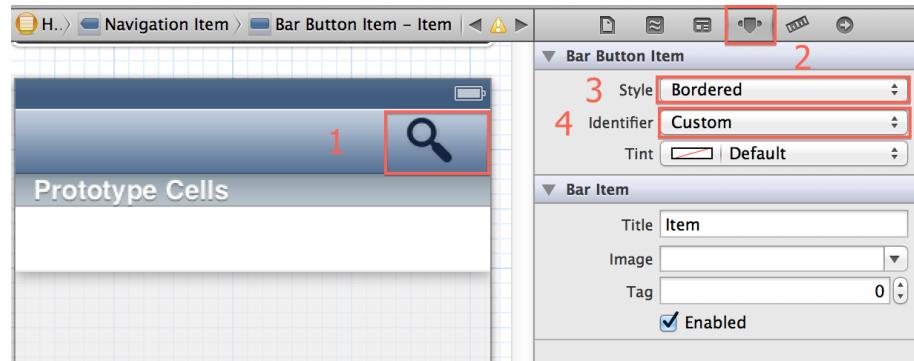
**Hình 6.318 Liên kết Home và Searching Home**

→ Tại “**Home View Controller**” kéo thả “**Round Rect Button**” vào “**Navigation Bar**” và thiết lập các thuộc tính cho button như sau:



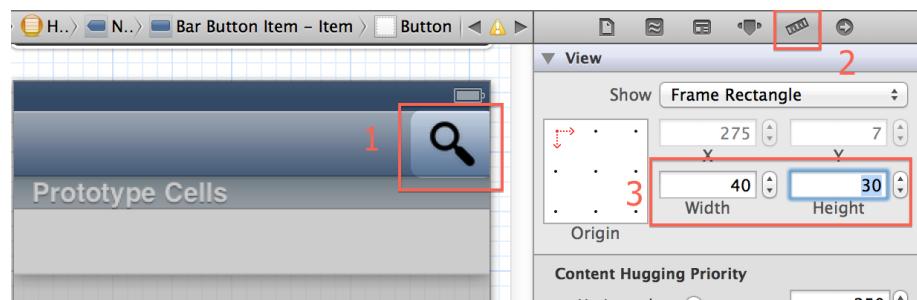
Hình 6.319 Kéo thả button vào

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
 → Nhấp chuột ra ngoài và click chọn lại button → identifier: **Custom**.



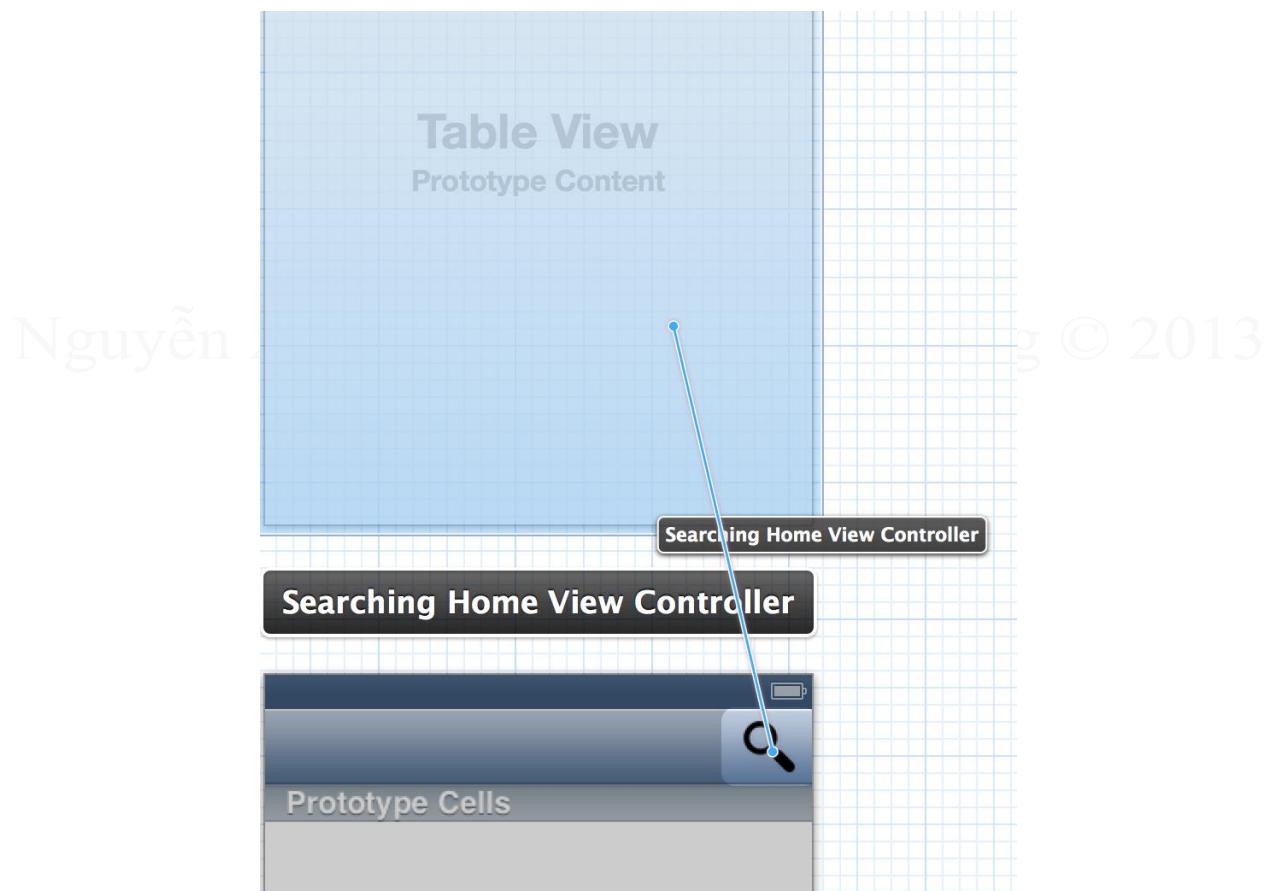
Hình 6.320 Tùy chỉnh button

→ Thiết lập độ rộng và chiều cao cho button:



**Hình 6.321 Thiết lập chiều rộng – cao cho button**

→ Giữ Ctrl và kéo thả button search từ “Home View Controller” sang “Searching Home View Controller”.



**Hình 6.322 Tạo liên kết**

→ Chọn “Push”

## Action Segue

### push

### modal

### custom

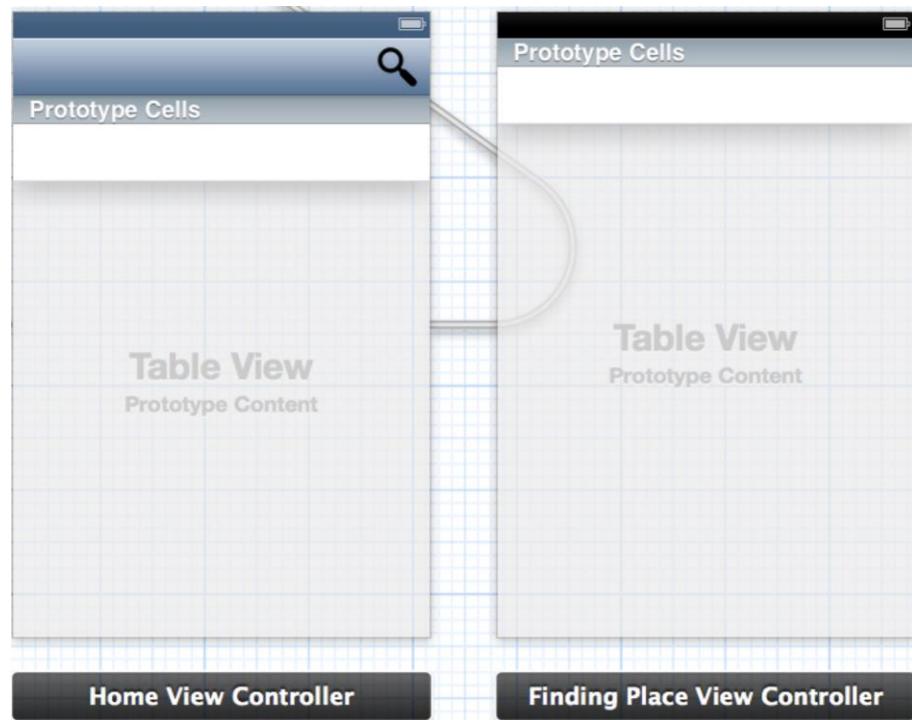
Hình 6.323 Chọn loại liên kết

→ Chạy thử, kết quả:



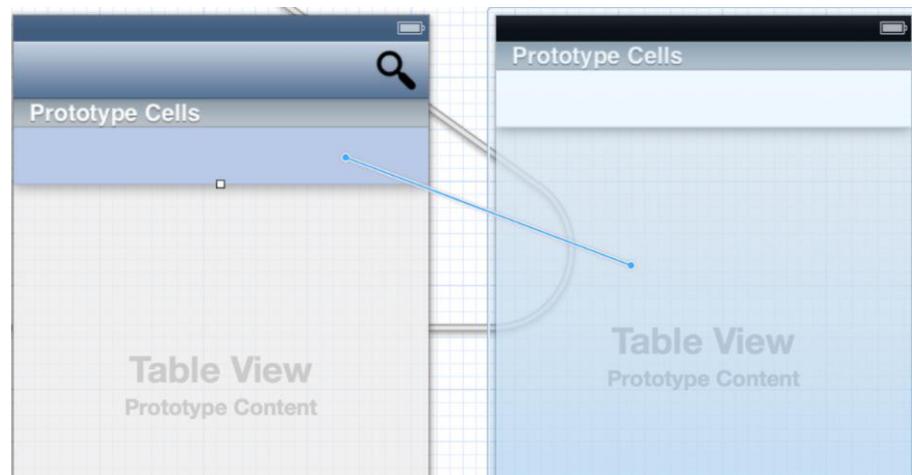
Hình 6.324 Kết quả chạy thử

Liên kết giữa “**Home View Controller**” và “**Finding Place View Controller**”.



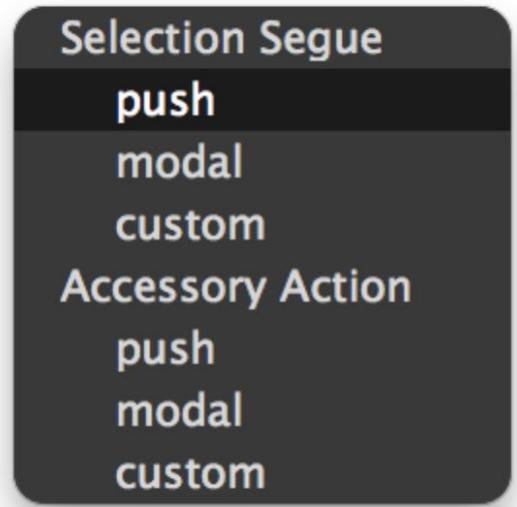
**Hình 6.325** Liên kết Home với Finding

→ Tại “Home View Controller” nhấp chọn Cell → giữ Ctrl và kéo thả sang “Finding Place View Controller”.



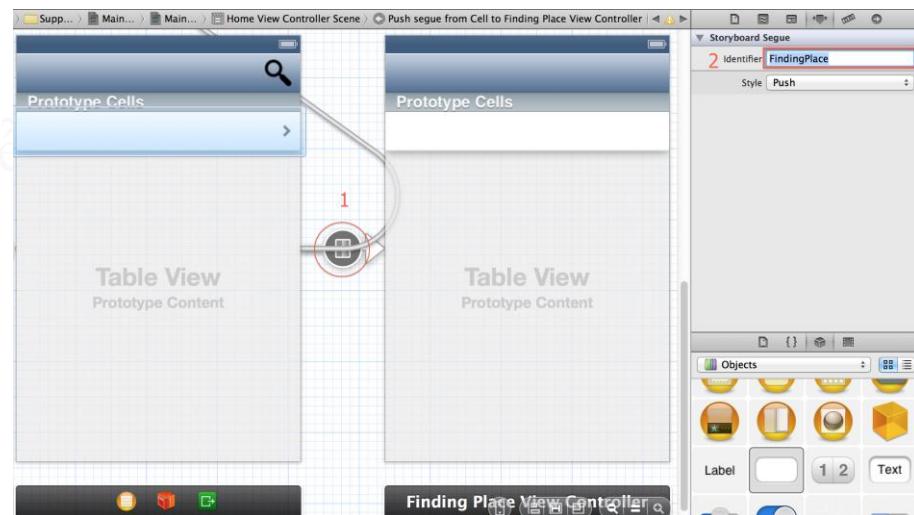
**Hình 6.326** Tạo liên kết

→ Chọn “Push”



**Hình 6.327 Chọn Push**

→ Đặt thuộc tính “identifier” là “**FindingPlace**”



**Hình 6.328 Đặt thuộc tính identifier**

→ Mở “**HomeViewController.m**” → thêm đoạn code sau vào cuối hàm “**clickButtons**”.

```
[self performSegueWithIdentifier:@"FindingPlace" sender:nil];
```

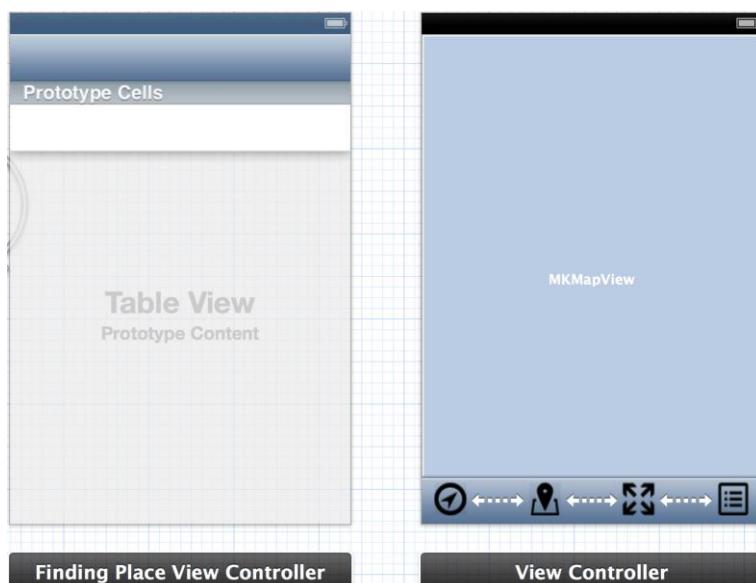
**Giải thích:** “**performSegueWithIdentifier**” cho phép chúng ta chuyển “View Controller” bằng code.

→ Chạy thử, kết quả:



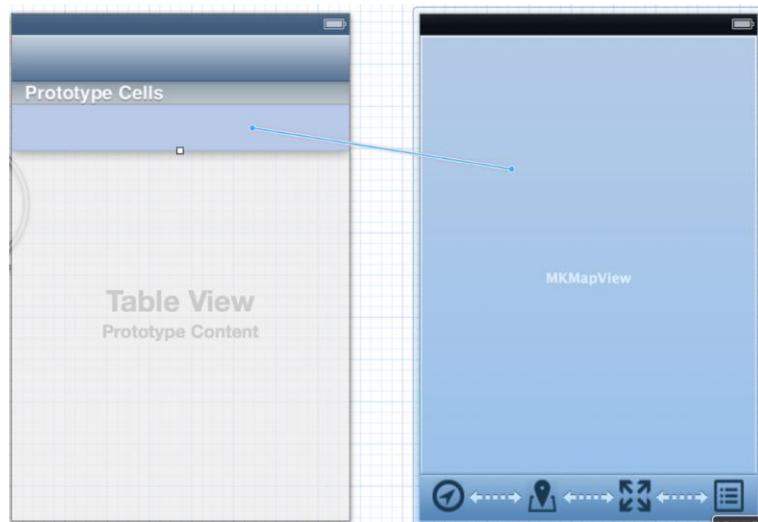
Nguyễn Anh Tiệp - Cao Thành Vàng © 2013  
Hình 6.329 Kết quả chạy thử

Liên kết giữa “**Finding Place View Controller**” và “**Map View Controller**”.



Hình 6.330 Liên kết Finding và Mapview

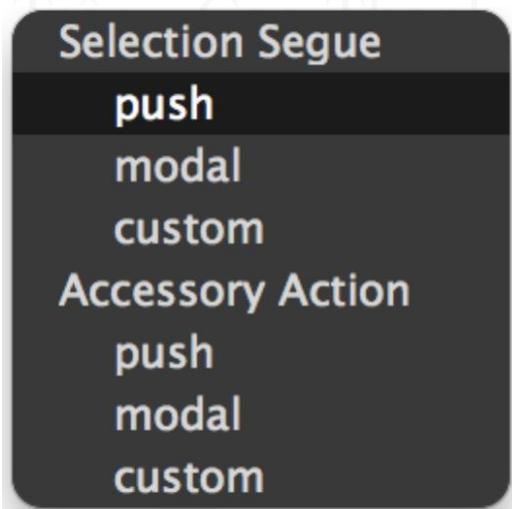
→ Tại “**Finding Place View Controller**” nhập chọn **Cell** → giữ Ctrl và kéo thả sang “**Map View Controller**”.



**Hình 6.321 Tạo liên kết**

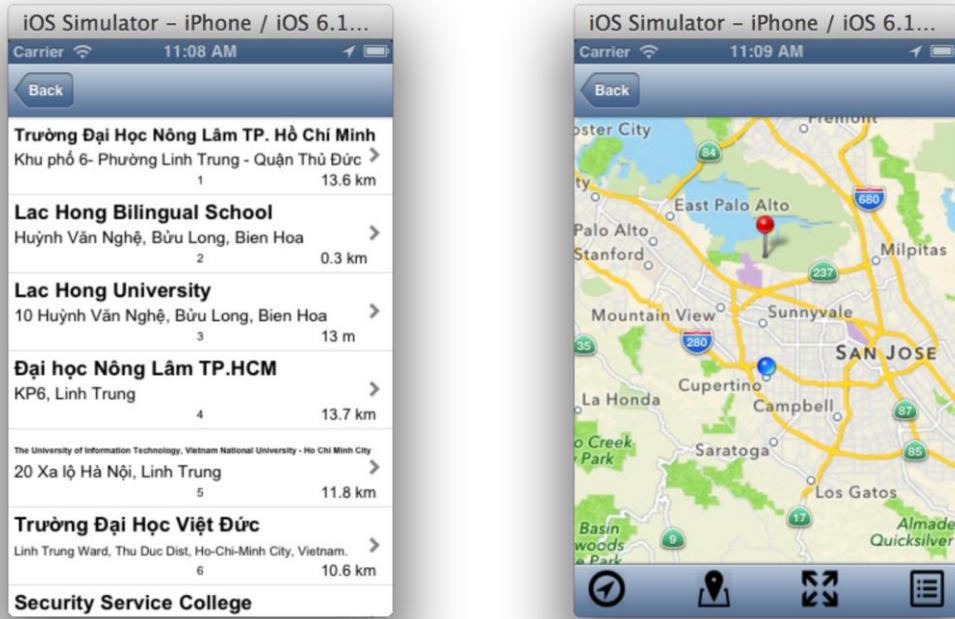
→ Chọn “**Push**”.

Nguyễn Anh Tài - iOS Development Vàng © 2013



**Hình 6.322 Chọn Push**

→ Chạy thử, kết quả.



**Hình 6.323 Kết quả chạy thử**  
**Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013**  
 Liên kết giữa “Map View Controller” và “Place Detail View Controller”.



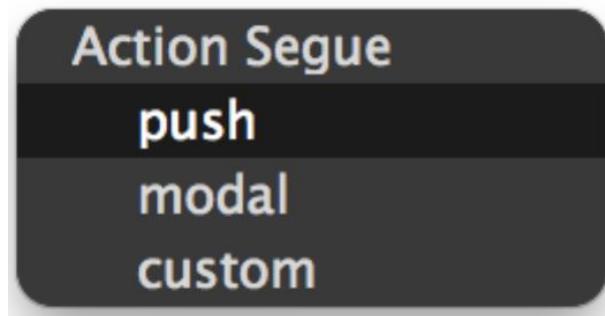
**Hình 6.324 Liên kết Map View và Place Detail**

→ Tại “**Map View Controller**” nhấp chọn button detail (góc phải dưới cùng) → giữ Ctrl và kéo thả sang “**Place Detail View Controller**”.



**Hình 6.235 Tạo liên kết**

Nguyễn Minh Tiệp - Cao Thành Vàng © 2013  
→ Chọn “**Push**”



**Hình 6.326 Chọn Push**

→ Có thể giao diện bên “**Place Detail View Controller**” sẽ bị lệch đi, bạn tự canh chỉnh lại. Chạy thử, kết quả.



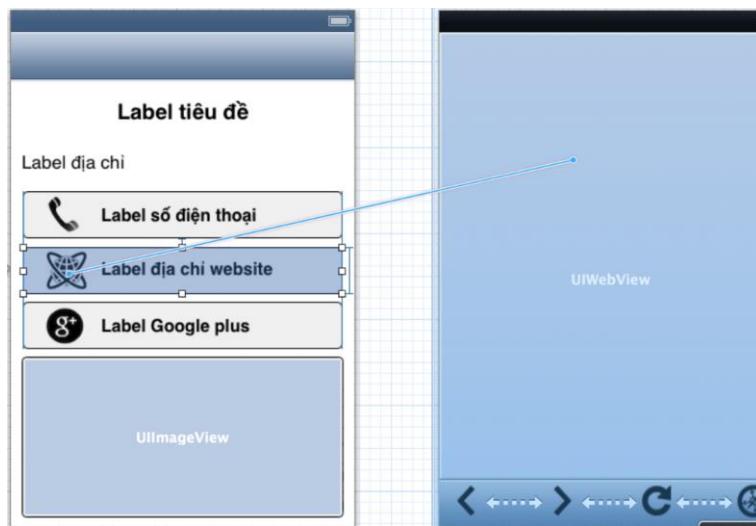
**Hình 6.327 Kết quả chạy thử**

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013  
Liên kết giữa “Place Detail View Controller” và “Web View Controller”.



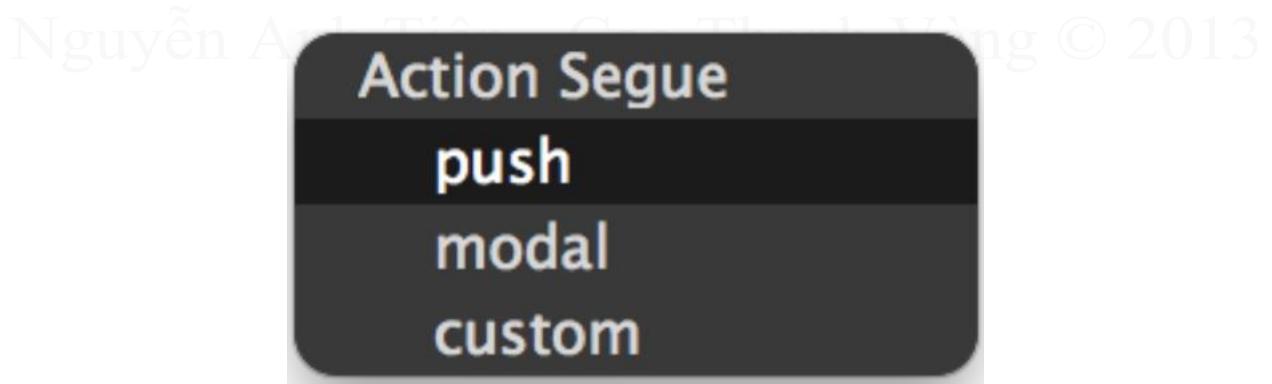
**Hình 6.328 Liên kết Place Detail và Web View**

- Tại “Place Detail View Controller” nhấp chọn button website (hình trái địa cầu)
- giữ Ctrl và kéo thả sang “Web View Controller”.



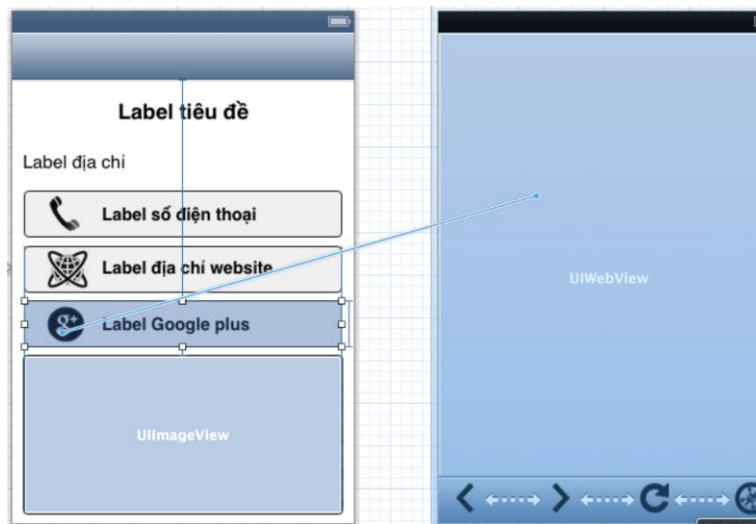
Hình 6.329 Tạo liên kết

- Chọn “Push”



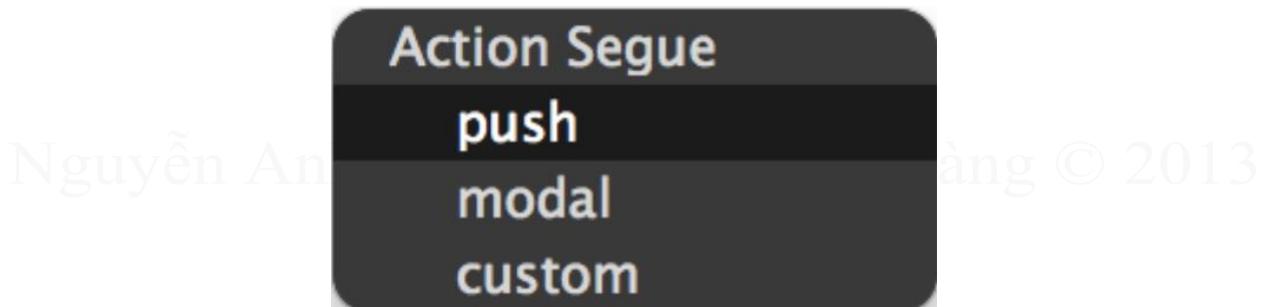
Hình 6.330 Chọn Push

- Tương tự với button Google Plus.



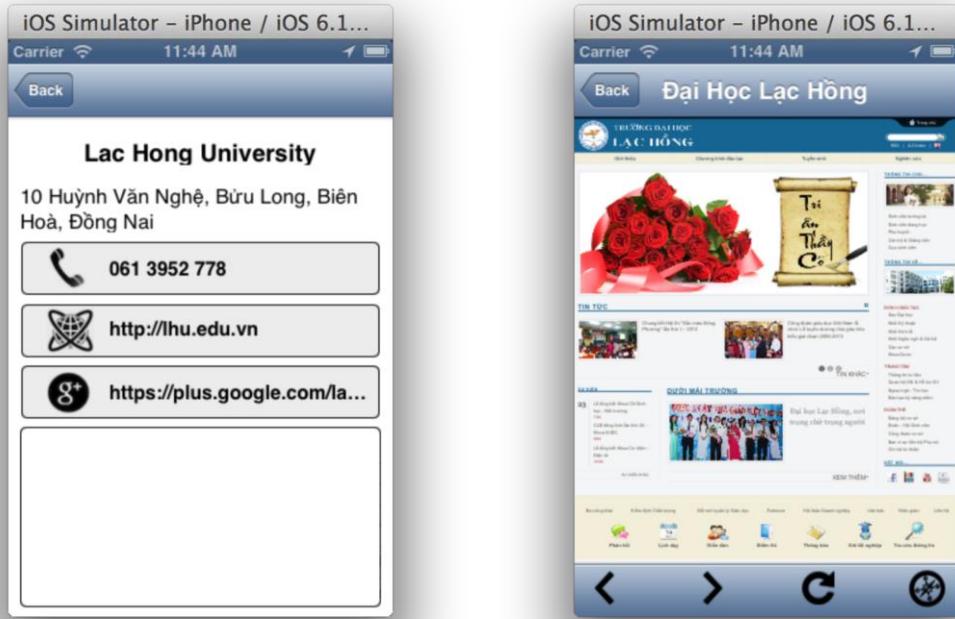
**Hình 6.331 Thực hiện với button G+**

→ Chọn “Push”



**Hình 6.332 chọn Push**

→ Chạy thử, kết quả.



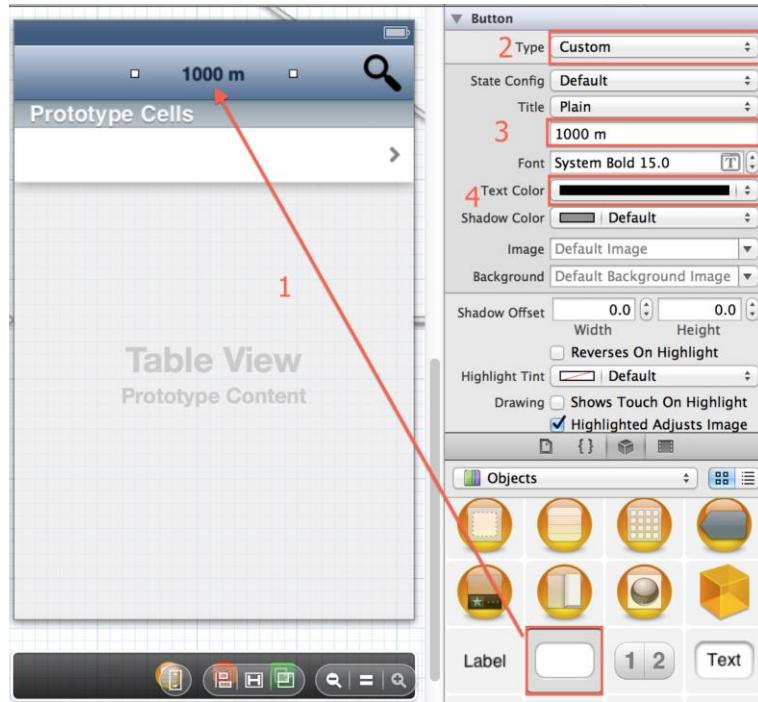
**Hình 6.333 Kết quả chạy thử**

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

**Bước 11:** Hoàn thiện ứng dụng.

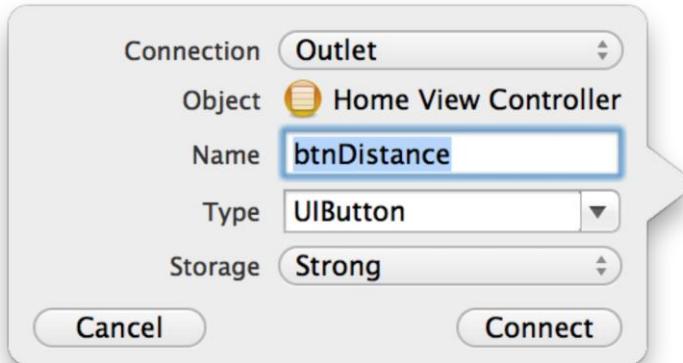
Hoàn thiện “**Home View Controller**”.

- Thêm chức năng thiết lập khoảng cách tìm kiếm.
- Tại “**Home View Controller**” nhất giữ Ctr và kéo thả đối tượng button và “**Navigation Bar**” → thiết lập các thuộc tính như hình.



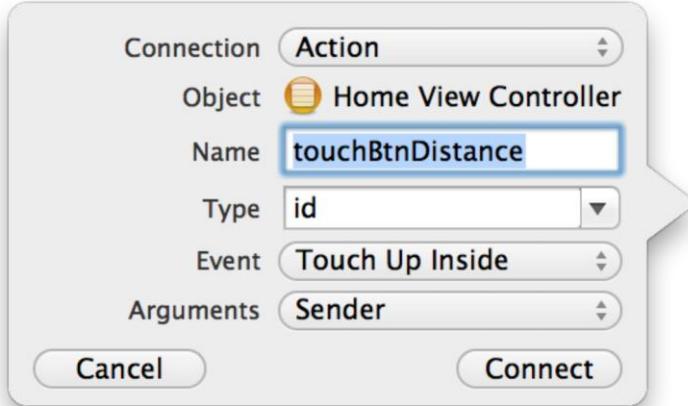
**Hình 6.334 Thiết lập thuộc tính**

→ Map (Outlet) đối tượng button vào file “**HomeViewController.h**” với tên”**btnDistance**”.



**Hình 6.335 Ánh xạ đối tượng**

→ Map (Action) đối tượng button vào file “**HomeViewController.h**” với tên “**touchBtnDistance**”



**Hình 6.336 Ánh xạ đối tượng**

→ Thêm protocol <UIActionSheetDelegate>, kết quả “MapViewController.h”:

```
#import <UIKit/UIKit.h>

@interface HomeViewController : UITableViewController
<UIActionSheetDelegate>
{
    IBOutlet UIButton *btnDistance;
}
-(IBAction)touchBtnDistance:(id)sender;
@end
```

→ Mở “MapViewController.m” thêm hàm:

```
- (void)changeSlider:(id)sender {
    UISlider *slider = (UISlider *)sender;
    NSString *value = [NSString stringWithFormat:@"%0.0f m", slider.value];
    [btnDistance setTitle:value forState:UIControlStateNormal];
    [btnDistance setTitleColor:[UIColor redColor] forState:UIControlStateNormal];
}
```

**Giải thích:** phương thức **changeSlider** được gọi khi người dùng thay đổi khoảng cách tìm kiếm. changeSlider sẽ thực hiện thay đổi và lưu lại giá trị khoảng cách vào button “**btnDistance**”.

```
- (NSString *)getDistance {
    NSString *strDistance = btnDistance.titleLabel.text;
    return [strDistance substringToIndex:strDistance.length-2];
}
```

**Giải thích:** phương thức **getDisstance** được sử dụng để loại bỏ “m” của khoảng cách tìm kiếm. VD: “1000 m” → “1000”.

```
- (void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex {  
    [btnDistance setTitleColor:[UIColor blackColor] forState:UIControlStateNormal];  
}
```

**Giải thích:** phương thức actionSheet được gọi tự động khi người dùng bắt đầu thay đổi slider, phương thức này sẽ thực hiện đổi màu chữ.

→ Sửa lại hàm “**touchBtnDistance**” như sau:

```
- (IBAction)touchBtnDistance:(id)sender {  
    NSString *title;  
    NSString *cancel;  
  
    title = @"Chọn khoảng cách tìm kiếm từ vị trí của bạn (meters)";  
    cancel = @"Đóng";  
  
    UIActionSheet *actionSheet = [[UIActionSheet alloc]  
        initWithTitle:nil  
        delegate:self  
        cancelButtonTitle:cancel  
        destructiveButtonTitle:nil  
        otherButtonTitles:nil];  
  
    //Thêm Slider vào ActionSheet  
    UISlider *slider = [[UISlider alloc] initWithFrame:CGRectMake(30, -50, 260, 70)];  
    slider.minimumValue = 100;  
    slider.maximumValue = 50000;  
    slider.value = [[self getDistance] floatValue];  
    [slider addTarget:self action:@selector(changeSlider:)  
    forControlEvents:UIControlEventValueChanged];  
    [actionSheet addSubview:slider];  
  
    //Thêm Title vào ActionSheet  
    UILabel *lblTitle = [[UILabel alloc] initWithFrame:CGRectMake(0, -80, 300, 40)];  
    lblTitle.text = title;  
    lblTitle.numberOfLines = 0;  
    lblTitle.textAlignment = NSTextAlignmentCenter;  
    [lblTitle setBackgroundColor:[UIColor clearColor]];  
    lblTitle.textColor = [UIColor whiteColor];  
    [actionSheet addSubview:lblTitle];  
  
    //Chỉnh lại vị trí của ActionSheet  
    [actionSheet showInView:self.view];  
    [actionSheet setBounds:CGRectMake(0,-90, 320, 280)];  
}
```

**Giải thích:** khi người dùng chạm vào **btnDistance**, phương thức “**touchBtnDistance**” sẽ được gọi, phương thức này sẽ hiển thị giao diện “ActionSheet” để người dùng thay đổi khoảng cách.

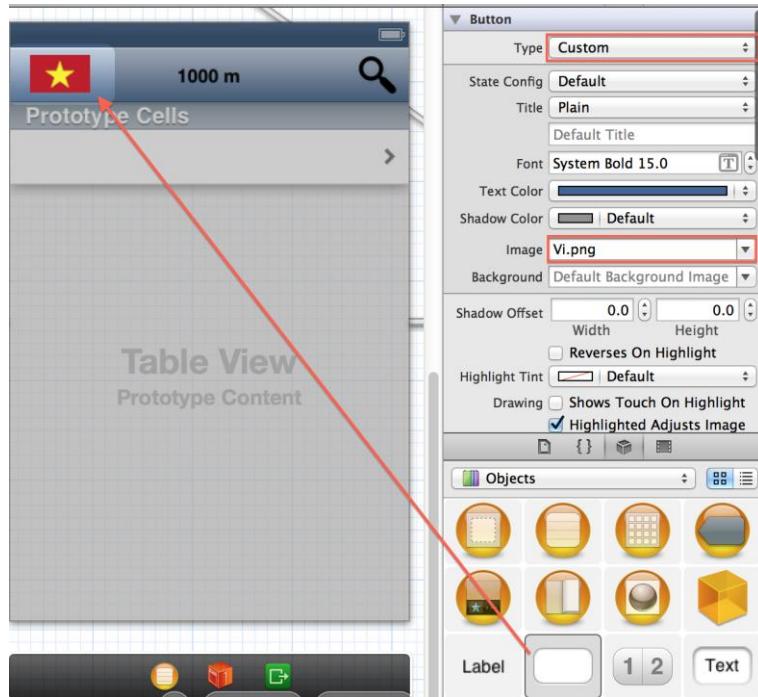
→ Chạy thử, kết quả:



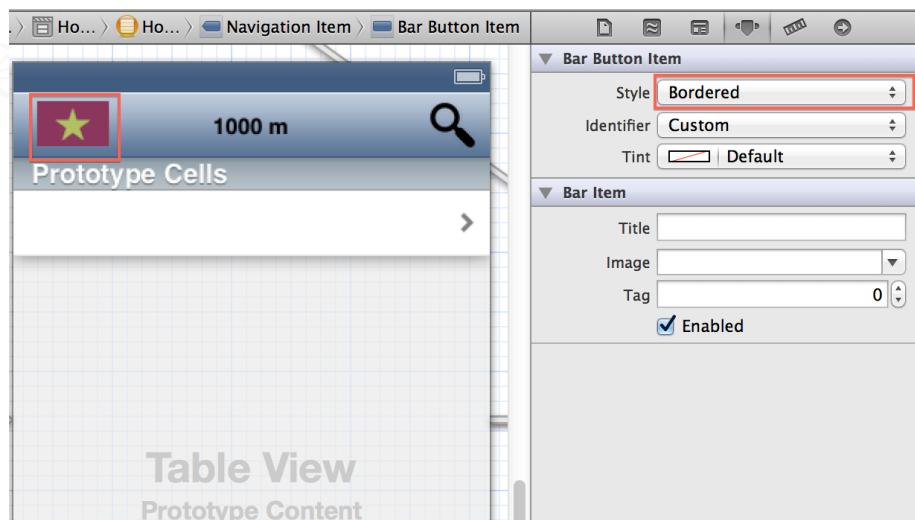
**Hình 6.337 Kết quả chạy thử**

Thêm chức năng thiết lập ngôn ngữ:

→ Tại “**Home View Controller**” thêm button như hình vẽ:



Hình 6.338 Thêm button



Hình 6.339 Cấu hình button

→ Map (Outlet) cho button với tên “**btnLanguage**” và Map (Action) với tên “**touchBtnLanguage**”, kết quả “**HomeViewController.h**”:

```
#import <UIKit/UIKit.h>

@interface HomeViewController : UITableViewController
<UIActionSheetDelegate>
```

```
{
    IBOutlet UIButton *btnDistance;
    IBOutlet UIButton *btnLanguage;
}
- (IBAction)touchBtnDistance:(id)sender;
- (IBAction)touchBtnLanguage:(id)sender;
@end
```

→ Mở “**HomeViewController.m**” → tại hàm “**configColumn**” sửa lại như sau:

```
- (void)configColumn:(UIButton *)column index:(int)index {
    NSString *image = [[arrPlaces objectAtIndex:index] imageName];

    NSString *title;
    if ([btnLanguage currentImage] == [UIImage imageNamed:@"Vi.png"]) {
        title = [arrPlaces[index] titleVi];
    } else {
        title = [arrPlaces[index] titleEn];
    }

    //Gán background cho column (button)
    [column setBackgroundColor:[UIColor colorWithPatternImage:[UIImage imageNamed:image]]];
    //Gán tiêu đề cho column
    [column setTitle:title forState:UIControlStateNormal];
    //Thêm hành động cho column
    [column addTarget:self action:@selector(clickButtons:) forControlEvents:UIControlEventTouchUpInside];
}
```

→ Mở “**HomeViewController.m**” → tại hàm “**touchBtnLanguage**” sửa lại như sau:

```
- (IBAction)touchBtnLanguage:(id)sender {
    UIImage *img=[(UIButton *) sender currentImage];

    if (img == [UIImage imageNamed:@"Vi.png"]) {
        [sender setImage:[UIImage imageNamed:@"En.png"] forState:UIControlStateNormal];
    } else {
        [sender setImage:[UIImage imageNamed:@"Vi.png"] forState:UIControlStateNormal];
    }

    [self.tableView reloadData];
}
```

→ Chạy thử, kết quả:



**Hình 6.340 Kết quả chạy thử**

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013  
 Lấy toạ độ hiện tại của người dùng để chuẩn bị truyền sang “**Finding Place View Controller**”.

→ Mở “**HomeViewController.h**”, import “**CoreLocation**” → thêm protocol “**CLLocationManagerDelegate**” và biến “**locationManager**”:

```
#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>

@interface HomeViewController : UITableViewController
<UIActionSheetDelegate, CLLocationManagerDelegate>
{
    IBOutlet UIButton *btnDistance;
    IBOutlet UIButton *btnLanguage;
    CLLocationManager *locationManager;
}
- (IBAction)touchBtnDistance:(id)sender;
- (IBAction)touchBtnLanguage:(id)sender;
@end
```

→ Mở “**HomeViewController.m**” thêm 2 phương thức sau để lấy toạ độ người dùng:

```

- (void)startLocation {
    if (locationManager == nil) {
        locationManager = [[CLLocationManager alloc] init];
    }

    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyBest;

    [locationManager startUpdatingLocation];
}

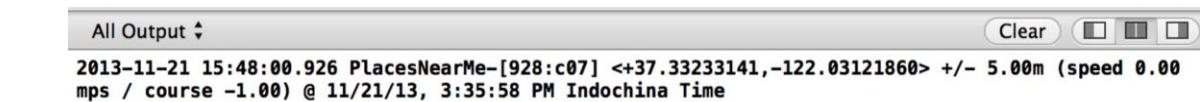
- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations {
    [locationManager stopUpdatingLocation];
}

```

→ Thêm “**NSLog(@"%@",locationManager.location);**” vào cuối hàm “**touchBtnLanguage**” để in ra thử toạ độ người dùng.

```
NSLog(@"%@",locationManager.location);
```

→ Thêm “[self startLocation];” vào cuối phương thức “**viewDidLoad**”, chạy thử và click vào button language, kết quả:



**Hình 6.341 Kết quả chạy thử**

Truyền dữ liệu qua “**Finding Place View Controller**”

→ Để truyền dữ liệu giữa các “ViewController” ta sử dụng “**NSUserDefaults**” và để truyền một đối tượng với “**NSUserDefaults**” thì đối tượng đó phải có 2 phương thức decoder và encoder → Mở “**Place.h**” phía trên “#import” thêm các key như sau:

```

#define ImageNameKey  @"ImageNameKey"
#define TitleEnKey    @"TitleEnKey"
#define TitleViKey    @"TitleViKey"
#define PlaceTypeKey  @"PlaceTypeKey"
#define KeyWordKey    @"KeyWordKey"

```

→ Qua file “**Place.m**” sửa lại như sau:

```

#import "Place.h"

@implementation Place

@synthesize imageName, titleEn, titleVi, placeType, keyWord;

- (id)initWithCoder:(NSCoder *)decoder {
    self = [super init];
    if(self) {
        imageName = [decoder decodeObjectForKey:ImageNameKey];
        titleEn = [decoder decodeObjectForKey:TitleEnKey];
        titleVi = [decoder decodeObjectForKey:TitleViKey];
        placeType = [decoder decodeObjectForKey:PlaceTypeKey];
        keyWord = [decoder decodeObjectForKey:KeyWordKey];
    }
    return self;
}

- (void)encodeWithCoder:(NSCoder *)encoder {
    [encoder encodeObject:imageName forKey:ImageNameKey];
    [encoder encodeObject:titleEn forKey:TitleEnKey];
    [encoder encodeObject:titleVi forKey:TitleViKey];
    [encoder encodeObject:placeType forKey:PlaceTypeKey];
    [encoder encodeObject:keyWord forKey:KeyWordKey];
}

@end

```

→ Mở “**HomeViewController.m**” sửa hàm “**clickButton**” như sau:

```

- (void)clickButtons:(id)sender {
    int selected = [self selectedIndex:sender];

    //Truyền dữ liệu qua ViewController kế tiếp (FindingPlaceViewController)
    Place *p = arrPlaces[selected];
    NSData *data = [NSKeyedArchiver archivedDataWithRootObject:p];
    [[NSUserDefaults standardUserDefaults] setObject:data forKey:@"Place"];

    //Kiểm tra nếu đã lấy được vị trí hiện tại của user lúc đó mới thực hiện chuyển VIEW
    if (locationManager.location) {
        [self performSegueWithIdentifier:@"FindingPlace" sender:nil];
    } else {
        UIAlertView *notifyAlert = [[UIAlertView alloc]
            initWithTitle:@"Lỗi"
            message:@"Kiểm tra \"Wifi\" hoặc \"Location Services\""
            delegate:self
            cancelButtonTitle:@"Đóng"
            otherButtonTitles:nil, nil];
        [notifyAlert show];
    }
}

```

```
}
```

**Giải thích:** hàm này sẽ gửi đối tượng “Place” sang các “ViewController” khác và kiểm tra đã tồn tại wifi hoặc cho phép sử dụng “**Location Services**” hay chưa.

→ Thêm hàm “**prepareForSegue**” như sau:

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    CLLocationCoordinate2D coordinate = locationManager.location.coordinate;
    NSUserDefaults *passingValue = [NSUserDefaults standardUserDefaults];
    [passingValue setFloat:coordinate.latitude forKey:@"currentLatitude"];
    [passingValue setFloat:coordinate.longitude forKey:@"currentLongitude"];
    [passingValue setObject:[self getDistance] forKey:@"distance"];
}
```

**Giải thích:** hàm này sẽ sử dụng “**NSUserDefaults**” để truyền dữ liệu(toạ độ, khoảng cách) qua các “**ViewController**” khác.

Hoàn thiện “**Finding Place View Controller**”.

→ Hiển thị đúng địa điểm người dùng cần tìm từ “Home View Controller”

→ Mở “FindingPlaceViewController.m” import “Place.h”:

```
#import "Place.h"
```

→ Thêm phương thức “**passingPlace**” để nhận đối tượng “place” mà “**Home View Controller**” đã truyền đi.

```
- (Place *)passingPlace {
    NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"Place"];
    return [NSKeyedUnarchiver unarchiveObjectWithData:data];
}
```

→ Tại “**viewDidLoad**” sửa lại như sau:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    NSUserDefaults *passingValue = [NSUserDefaults standardUserDefaults];
    coordinate.latitude = [[passingValue objectForKey:@"currentLatitude"] floatValue];
    coordinate.longitude = [[passingValue objectForKey:@"currentLongitude"] floatValue];
    NSString *radius = [passingValue objectForKey:@"distance"];
```

```

Place *place = [[Place alloc] init];
place = [self passingPlace];

NSString *placeType = place.placeType;
NSString *keyWord = place.keyWord;

NSMutableArray *arrPlaces = [self searchPlacesFromGoogle:radius placeType:placeType
keyword:keyWord];
arrPlaceSearching = [self getPlaceSearching:arrPlaces];

self.navigationItem.title = place.titleVi;
}

```

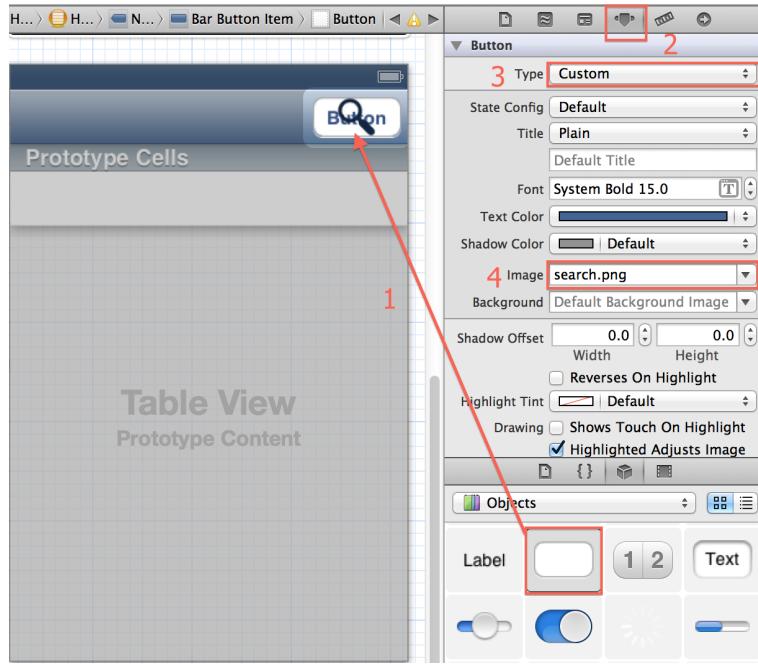
→ Chạy thử → chọn nhà hàng → kết quả:



**Hình 6.342 Kết quả chạy thử**

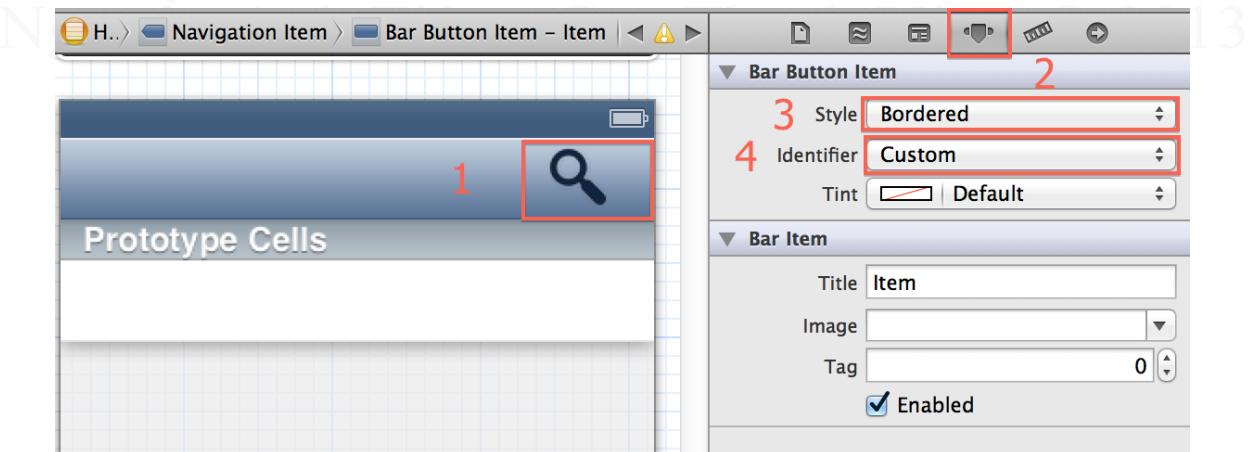
Thêm tính năng tìm kiếm:

→ Tại “**Finding Place View Controller**” kéo thả “**Round Rect Button**” vào “**Navigation Bar**” và thiết lập các thuộc tính cho button như sau:



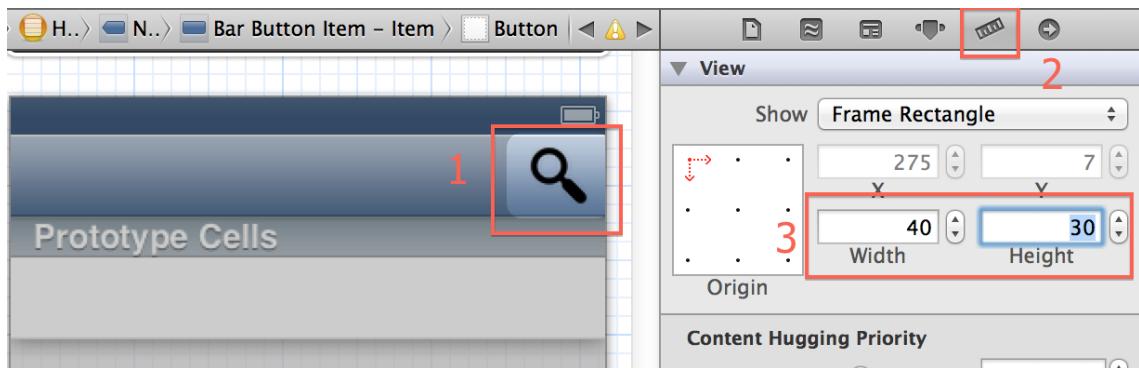
**Hình 6.343 Thêm Button vào giao diện**

→ Nhấp chuột ra ngoài và click chọn lại button → identifier: **Custom**.



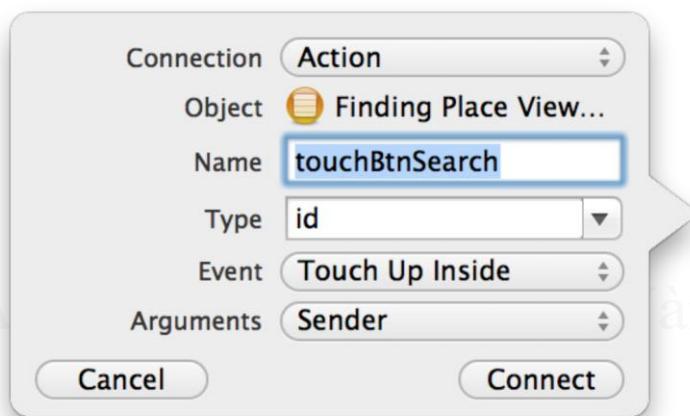
**Hình 6.344 Bổ sung Identifier**

→ Thiết lập độ rộng và chiều cao cho button:



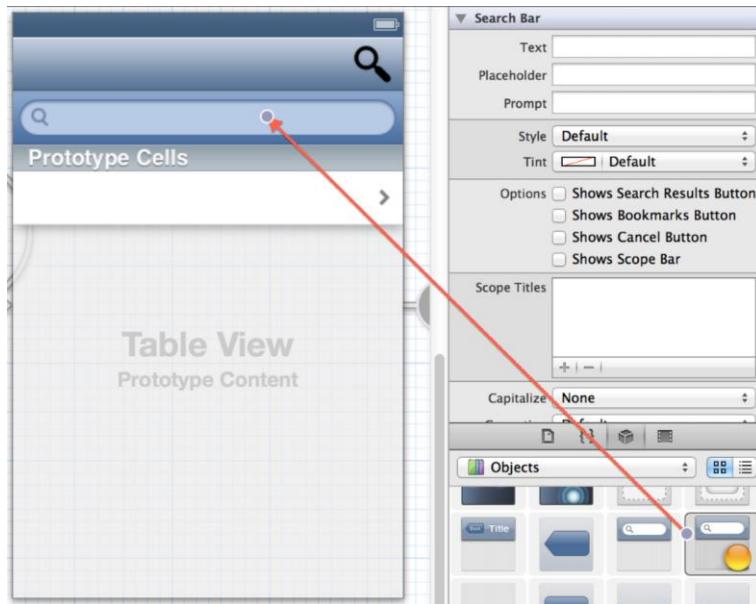
**Hình 6.345 Thiết lập chiều cao – rộng của button**

→ Map (Action) button search với tên “**touchBtnSearch**”:



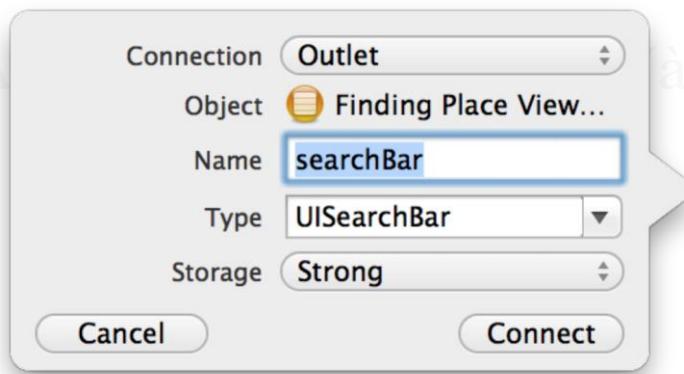
**Hình 6.346 Ánh xạ đối tượng**

→ Qua giao diện kéo thêm “**Search Bar and Search Diplay Controller**” vào “**Finding Place View Controller**”.



**Hình 6.347 Thêm Search bar**

→ Map (Outlet) đối tượng “Search Bar” với tên “searchBar”



**Hình 6.348 Ánh xạ đối tượng**

→ Thêm protocol “**UISearchBarDelegate**”, kết quả  
“**FindingPlaceViewController.h**”:

```
#import <UIKit/UIKit.h>

@interface FindingPlaceViewController : UITableViewController
<UISearchBarDelegate>
{
    IBOutlet UISearchBar *searchBar;
}
```

```
- (IBAction)touchBtnSearch:(id)sender;
@end
```

→ Mở “**FindingPlaceViewController.m**” tại hàm “**touchBtnSearch**” thêm đoạn code sau:

```
[self.searchDisplayController.searchBar becomeFirstResponder];
```

→ Bên dưới “**@implementation...**” khai báo mảng “**arrFilteringResults**”

```
NSMutableArray *arrFilteringResults;
```

→ Thêm 2 phương thức sau:

```
- (void)searchThroughData {
    arrFilteringResults = nil;

    NSPredicate *resultsPredicate = [NSPredicate predicateWithFormat:@"SELF.name contains
[search] %@", searchBar.text];
    arrFilteringResults = [[arrPlaceSearching filteredArrayUsingPredicate:resultsPredicate]
mutableCopy];
}

- (void)searchBar:(UISearchBar *)searchBar textDidChange:(NSString *)searchText {
    [self searchThroughData];
}
```

→ Sửa lại hàm “**numberOfRowsInSection**” như sau:

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    if (tableView == self.tableView) {
        return arrPlaceSearching.count;
    } else {
        [self searchThroughData];
        return arrFilteringResults.count;
    }
}
```

→ Tại hàm “**cellForRowAtIndexPath**”, thay thế đoạn code sau vào giữa “**static NSString \*CellIdentifier = @"Cell";**” và “**// Configure the cell...**”:

```
static NSString *CellIdentifier = @"Cell";

UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
```

```

cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
cell.selectionStyle = UITableViewCellAccessoryNone;

int row = indexPath.row;
PlaceSearching *ps;
if (tableView == self.tableView) {
    ps = [arrPlaceSearching objectAtIndex:row];
} else {
    ps = [arrFilteringResults objectAtIndex:row];
}

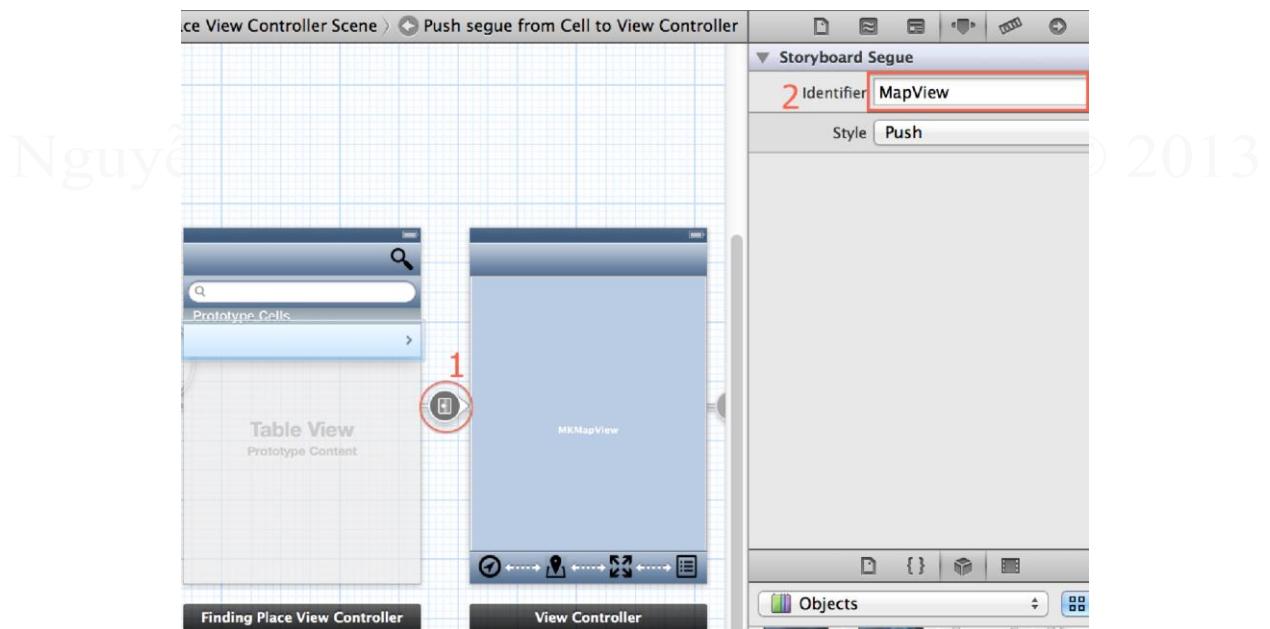
// Configure the cell...

```

→ Thêm đoạn code sau vào hàm “**didSelectRowAtIndexPath**”:

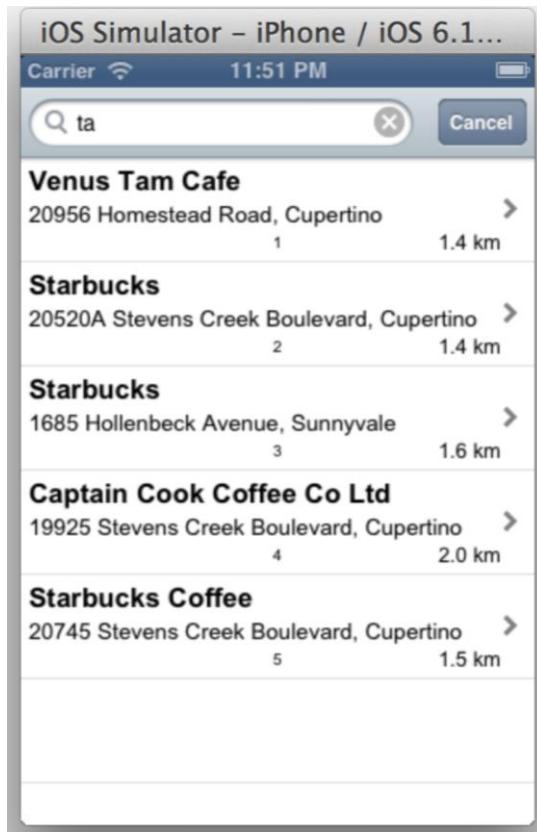
```
[self performSegueWithIdentifier:@"MapView" sender:nil];
```

→ Qua giao diện đặt tên cho liên kết giữa “**Finding Place View Controller**” và “**Map View Controller**” là “**MapView**”.



**Hình 6.349 Đặt tên liên kết**

→ Chạy thử, kết quả:



Nguyễn Anh Tú - Tài liệu lập trình iOS - Phần Vàng © 2013

Truyền dữ liệu qua các “ViewController” khác

→ Tương tự như truyền đối tượng “Place”, mở “**PlaceManager.h**” thêm vào các key trước “#import...”

```
#define NameKey      @"NameKey"
#define VicinityKey   @"VicinityKey"
#define UnitsKey       @"UnitsKey"
#define LatitudeKey    @"LatitudeKey"
#define LongitudeKey   @"LongitudeKey"
#define ReferenceKey   @"ReferenceKey"
```

→ Mở “**PlaceManager.m**” thêm 2 phương thức sau:

```
#import "PlaceSearching.h"

@implementation PlaceSearching
@synthesize name, vicinity, units, latitude, longitude, reference;

- (id)initWithCoder:(NSCoder *)decoder {
    self = [super init];
```

```

if(self) {
    name = [decoder decodeObjectForKey:NameKey];
    vicinity = [decoder decodeObjectForKey:VicinityKey];
    units = [decoder decodeObjectForKey:UnitsKey];
    latitude = [decoder decodeObjectForKey:LatitudeKey];
    longitude = [decoder decodeObjectForKey:LongitudeKey];
    reference = [decoder decodeObjectForKey:ReferenceKey];
}
return self;
}

- (void)encodeWithCoder:(NSCoder *)encoder {
[encoder encodeObject:name forKey:NameKey];
[encoder encodeObject:vicinity forKey:VicinityKey];
[encoder encodeObject:units forKey:UnitsKey];
[encoder encodeObject:latitude forKey:LatitudeKey];
[encoder encodeObject:longitude forKey:LongitudeKey];
[encoder encodeObject:reference forKey:ReferenceKey];
}
@end

```

→ Mở “**FindingPlaceViewController.m**” thêm hàm **“prepareForSegue”**

```

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
//  int selected = self.tableView.indexPathForSelectedRow.row;
NSIndexPath *indexPath;

PlaceSearching *ps;

if (self.searchDisplayController.active) {
    indexPath = [[self.searchDisplayController searchResultsTableView]
indexPathForSelectedRow];
    ps = arrFilteringResults[indexPath.row];
} else {
    indexPath = self.tableView.indexPathForSelectedRow;
    ps = arrPlaceSearching[indexPath.row];
}

NSData *data = [NSKeyedArchiver archivedDataWithRootObject:ps];
[[NSUserDefaults standardUserDefaults] setObject:data forKey:@"PlaceSearching"];
}

```

Hoàn thiện “**Map View Controller**”.

→ Mở “**MapViewController.m**” import “**PlaceSearching.h**”

```
#import "PlaceSearching.h"
```

→ Sửa lại “**viewDidLoad**” như sau:

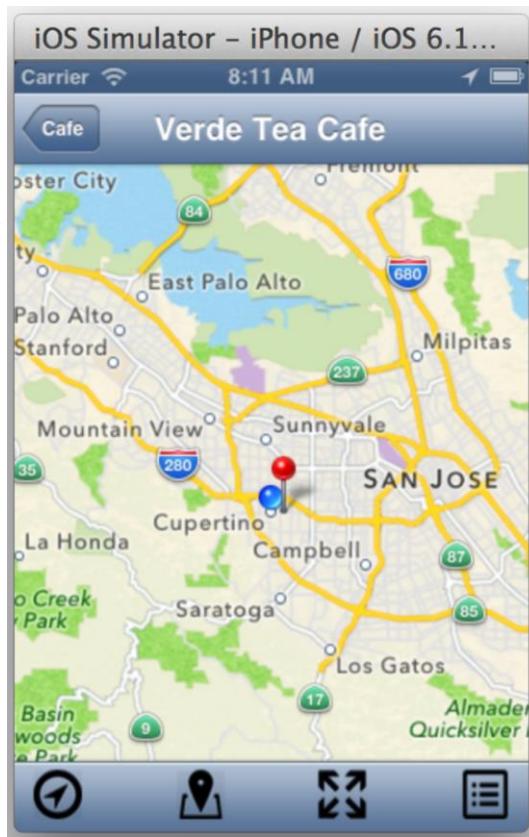
```
- (void)viewDidLoad
```

```

{
    [super viewDidLoad];
    [self startLocation];
    myMapView.delegate = self;
    NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"PlaceSearching"];
    PlaceSearching *ps = [NSKeyedUnarchiver unarchiveObjectWithData:data];
    [self addAnnotation:ps.latitude
                  longitude:ps.longitude
                     title:ps.name
                    subtitle:ps.vicinity];
    self.navigationItem.title = ps.name;
}

```

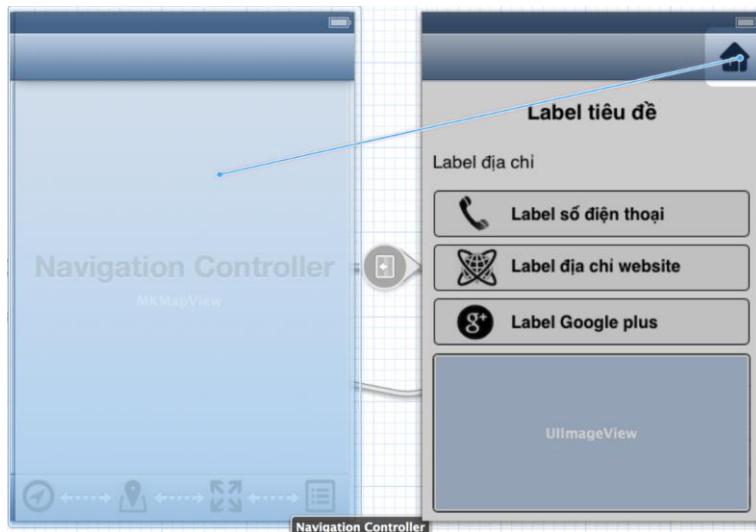
→ Chạy thử, kết quả:



**Hình 6.351 Kết quả chạy thử**

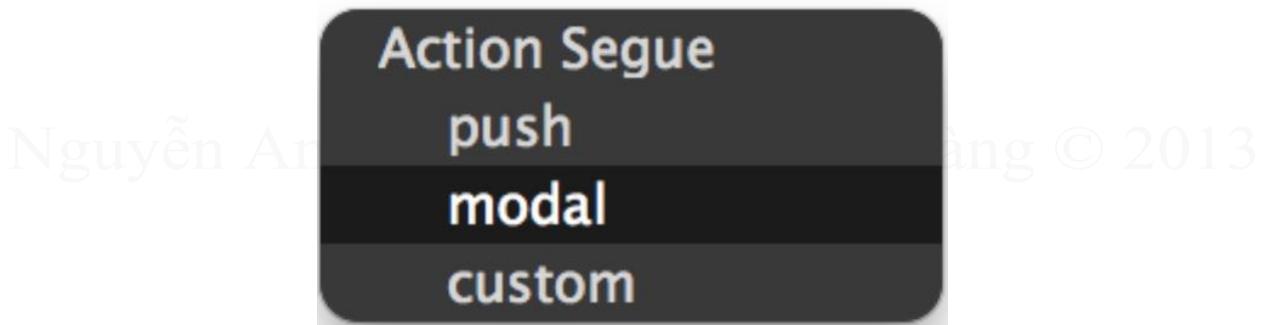
Thêm button home

→ Trên button Home nhấn giữ Ctrl và kéo thả về “Navigation Controller”



**Hình 6.352 Thêm button Home**

→ Chọn “modal”



**Hình 6.353 Ánh xạ đối tượng**

Hoàn thiện “Place Detail View Controller”

Thực hiện load dữ liệu từ Google

→ Mở “PlaceDetailViewController.m” import “Google.h” và “PlaceSearching.h”

```
#import "Google.h"
#import "PlaceSearching.h"
```

→ Thêm các phương thức sau:

```
- (PlaceDetail *)searchPlaceDetailFromGoogle:(NSString *)reference {
    NSDictionary *json = [Google searchDetail:reference];
    NSDictionary *result = [json objectForKey:@"result"];
```

```

PlaceDetail *pd = [[PlaceDetail alloc] init];

pd.name = [result objectForKey:@"name"];
pd.address = [result objectForKey:@"formatted_address"];
pd.phoneNumber = [result objectForKey:@"formatted_phone_number"];
pd.website = [result objectForKey:@"website"];
pd.googlePlus = [result objectForKey:@"url"];
pd.photos = [result objectForKey:@"photos"];

return pd;
}

- (NSData *)searchPlacePhotoFromGoogle:(NSString *)photoReference {
    return [Google searchPlacePhoto:photoReference];
}

```

→ Sửa lại “viewDidLoad” như sau:

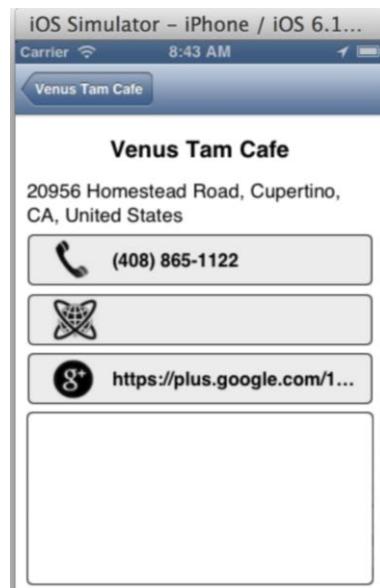
```

- (void)viewDidLoad
{
    [super viewDidLoad];

NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"PlaceSearching"];
PlaceSearching *ps = [NSKeyedUnarchiver unarchiveObjectWithData:data];

PlaceDetail *pd = [self searchPlaceDetailFromGoogle:ps.reference];
[self displayPlaceDetail:pd];
}

```



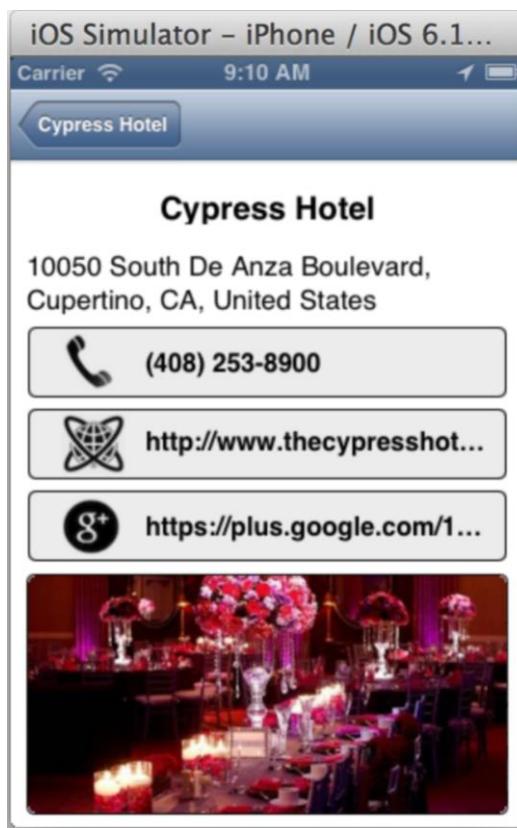
Hình 6.354 Kết quả chạy thử

Load hình ảnh từ Google:

→ Mở “**PlaceDetailViewController.m**” thêm đoạn code sau vào cuối hàm “**displayPlaceDetail**”

```
NSString *photoReference = [pd.photos[0] objectForKey:@"photo_reference"];
NSData *data = [self searchPlacePhotoFromGoogle:photoReference];
imgPhotoPlace.image = [[UIImage alloc] initWithData:data];
```

→ Chạy thử, kết quả:



**Hình 6.355 Kết quả chạy thử**

Thêm button home.

→ Thêm button home như giao diện sau.



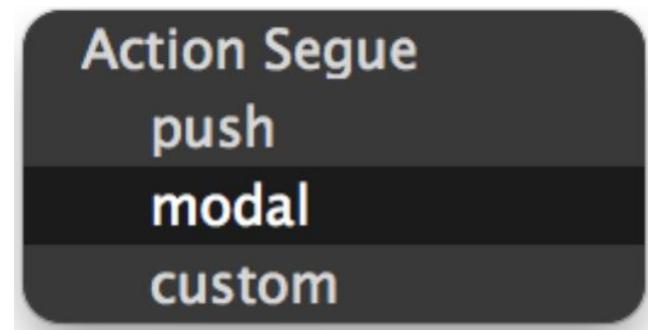
**Hình 6.356 Thêm button Home**

→ Trên button Home nhấn giữ Ctrl và kéo thả về “Navigation Controller”.



**Hình 6.357 Tạo kết nối**

→ Chọn “modal”



**Hình 6.358 Chọn Modal**

Truyền dữ liệu sang “**Web View Controller**”.

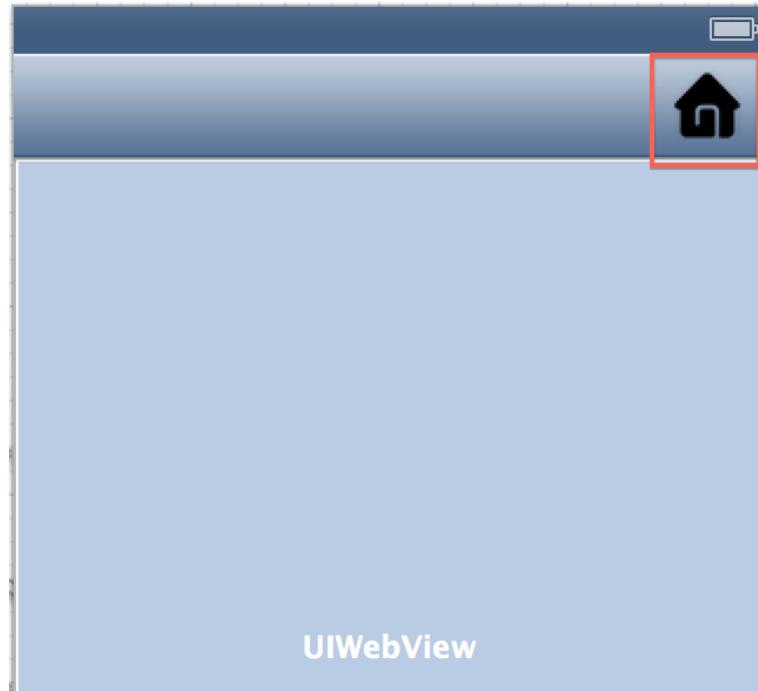
→ Thêm phương thức sau:

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    NSString *url;
    if ([sender currentImage] == [UIImage imageNamed:@"website.png"]) {
        url = lblWebsite.text;
    } else {
        url = lblGooglePlus.text;
    }
    [[NSUserDefaults standardUserDefaults] setObject:url forKey:@"url"];
}
```

Hoàn thiện “**Web View Controller**”

Thêm button home

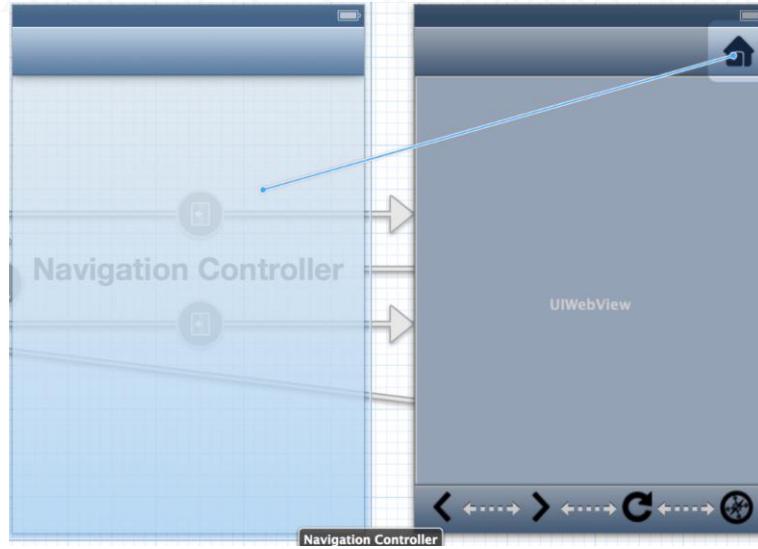
→ Thêm button home như giao diện sau:



**Hình 6.359 Thêm button Home**

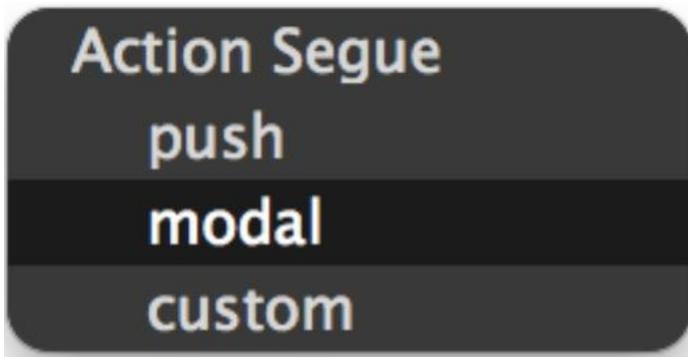
→ Trên button Home nhấn giữ Ctrl và kéo thả về “Navigation Controller”.

Nguyễn Anh Tiên - Cao Thành Vàng © 2013



**Hình 6.360 Tạo kết nối**

→ Chọn “modal”



Hình 6.361 Chọn modal

Nhận Url từ “Place Detail View Controller”.

→ Mở “WebViewController.m” sửa lại “viewDidLoad” như sau:

```
- (void)viewDidLoad
{
    [super viewDidLoad];

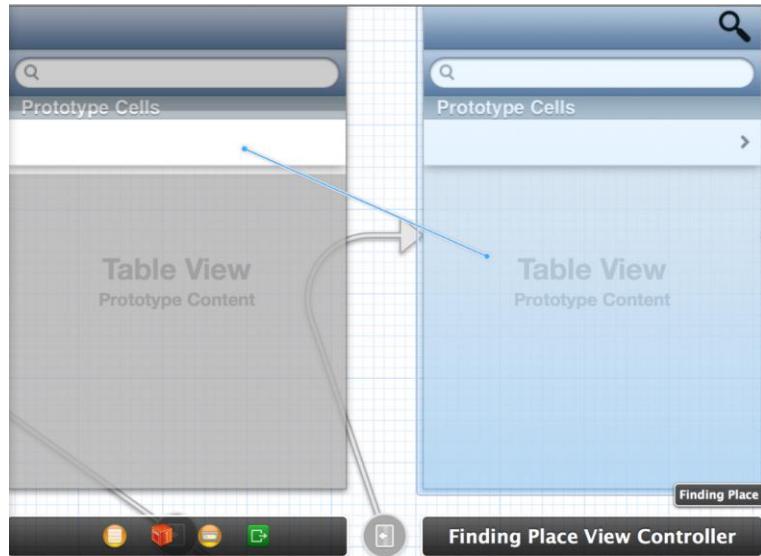
    url = [[NSUserDefaults standardUserDefaults] objectForKey:@"url"];
    if (!url) {
        url = @"https://www.google.com";
    }

    [self loadURL:url];
    myWebView.delegate = self;
}
```

Hoàn thiện “SearchingHomeViewController”.

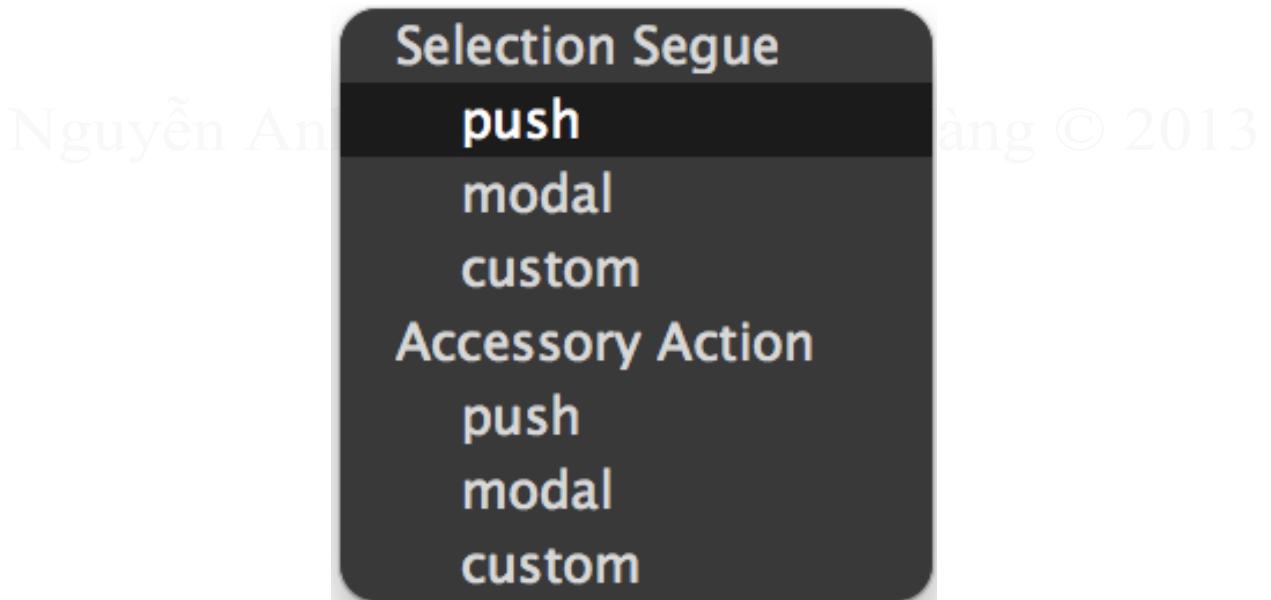
Liên kết giữa “Searching Home View Controller” và “Finding Place View Controller”.

→ Qua giao diện kéo thả Cell từ “Searching Home View Controller” sang “Finding Place View Controller”.



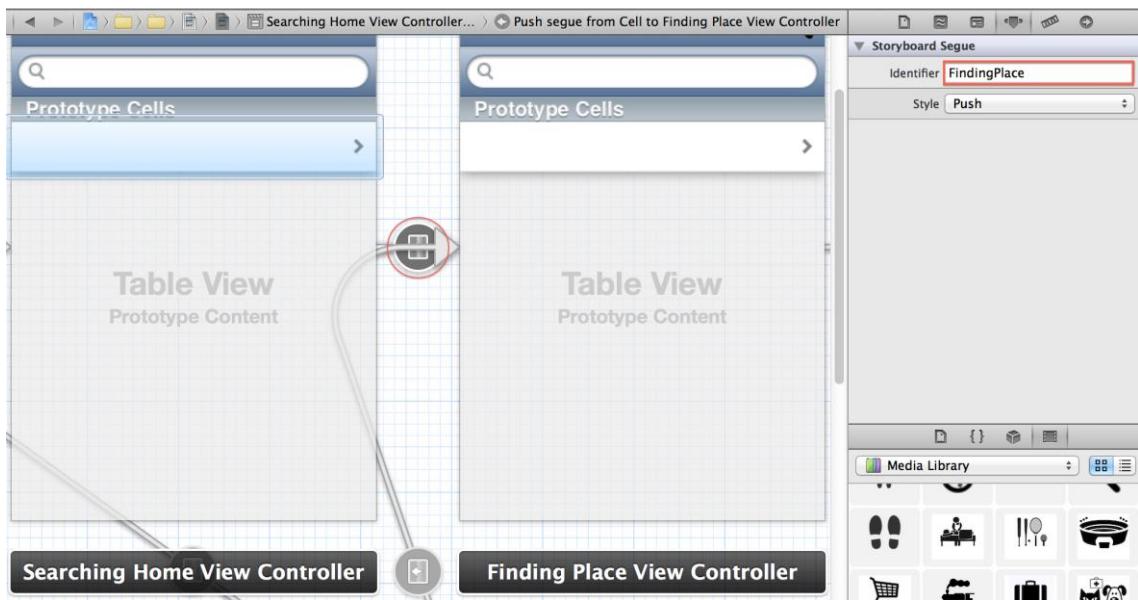
Hình 6.362 Tạo liên kết

→ Chọn “Push”.



Hình 6.363 Chọn Push

→ Đặt tên cho liên kết là “**Finding Place**”



**Hình 6.364 Đặt tên liên kết**

→ Mở “**SearchingHomeViewController.m**” tại hàm  
“**didSelectRowAtIndexPath**” thêm đoạn code sau:

```
[self performSegueWithIdentifier:@"FindingPlace" sender:nil];
```

→ Thêm phương thức “**prepareForSegue**”

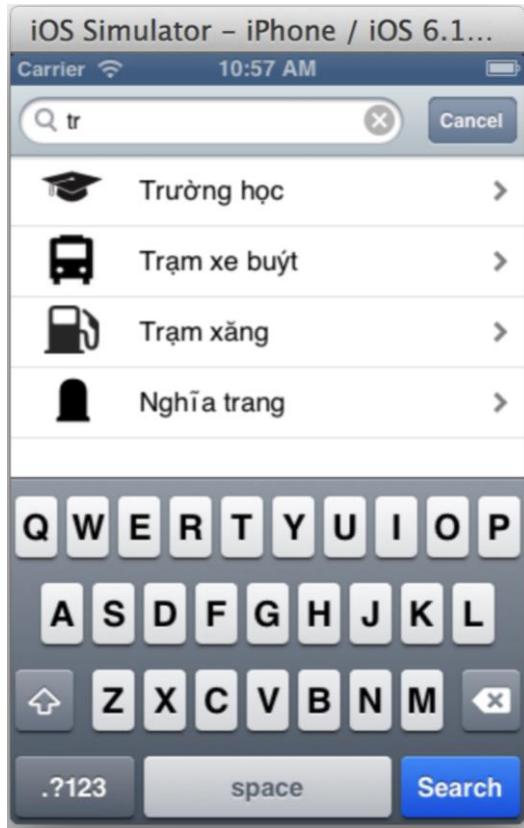
```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    NSIndexPath *indexPath;
    Place *p;

    if (self.searchDisplayController.isActive) {
        indexPath = [[self.searchDisplayController searchResultsTableView]
                     indexPathForSelectedRow];
        p = arrSearchingResults[indexPath.row];
    } else {

        indexPath = [self.tableView indexPathForSelectedRow];
        p = arrPlaces[indexPath.row];
    }

    //Truyền dữ liệu qua ViewController kế tiếp (NATFindingPlaceTVC)
    NSData *data = [NSKeyedArchiver archivedDataWithRootObject:p];
    [[NSUserDefaults standardUserDefaults] setObject:data forKey:@"Place"];
}
```

→ Chạy thử, kết quả:



Hình 6.365 Kết quả chạy thử

Đổi màu “Navigation Bar”.

→ Mở “**HomeViewController.m**” thêm phương thức sau:

```
- (void)configNavigationBar:(UIColor *)bgColor textColor:(UIColor *)fontColor{
    //set bar color
    self.navigationController.navigationBar.tintColor = bgColor;
    self.navigationController.navigationBar.alpha = 0.9f;
    self.navigationController.navigationBar.translucent = YES;
    [self.navigationController.navigationBar setTitleTextAttributes:[NSDictionary
        dictionaryWithObject:fontColor forKey:UITextAttributeTextColor]];
    //set back button color
    [[UIBarButtonItem appearanceWhenContainedIn:[UINavigationBar class], nil]
        setTitleTextAttributes:[NSDictionary dictionaryWithObjectsAndKeys:fontColor,
        UITextAttributeTextColor,nil] forState:UIControlStateNormal];
}
```

→ Thêm đoạn code sau vào cuối phương thức “**viewDidLoad**”

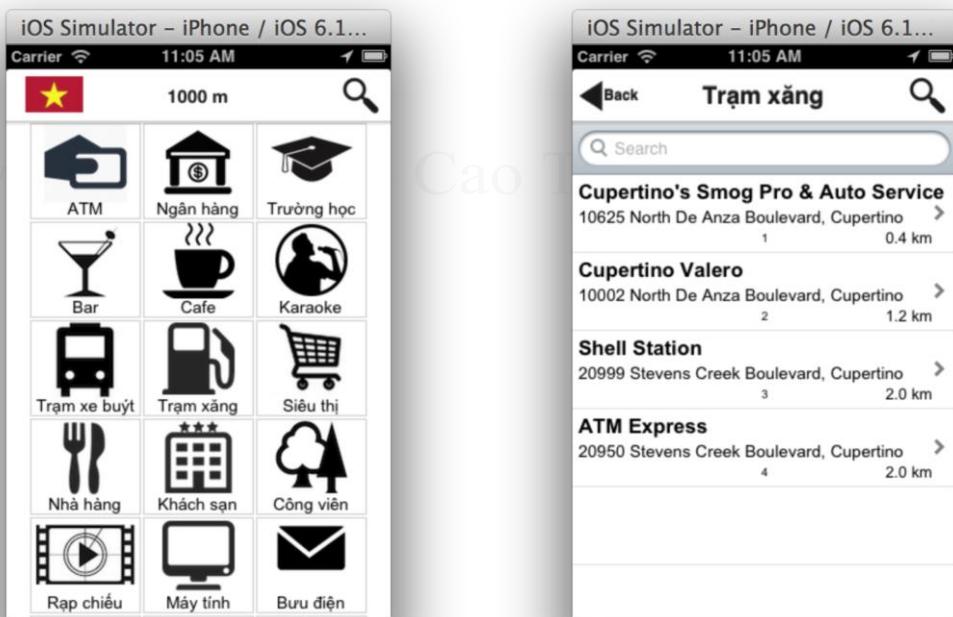
```
[self configNavigationBarBackground:[UIColor whiteColor] textColor:[UIColor blackColor]];
```

→ Mở “**NAppDelegate.m**” → hàm “**“didFinishLaunchingWithOptions”**” sửa lại:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    UIImage * backButtonImage = [UIImage imageNamed: @"go_back.png"];
    backButtonImage = [backButtonImage stretchableImageWithLeftCapWidth: 21.0
topCapHeight: 30.0];
    [[UIBarButtonItem appearance] setBackButtonBackgroundImage: backButtonImage forState: UIControlStateNormal barMetrics: UIBarMetricsDefault];

    // Override point for customization after application launch.
    return YES;
}
```

→ Chạy thử, kết quả:



Hình 6.366 Kết quả chạy thử

## **CHƯƠNG VII**

### **ĐƯA ỦNG DỤNG LÊN IPHONE**

Sau khi xây dựng hoàn chỉnh ứng dụng cho iPhone, bạn cần phải đưa ứng dụng lên chạy trên thiết bị iPhone thật để kiểm tra lại hoạt động của ứng dụng đó trên iPhone. Ngay cả khi bạn còn đăng xây dựng ứng dụng trên Xcode, nhiều ứng dụng bạn cũng phải đưa lên thiết bị thật mới kiểm tra được tính năng, do iOS Simulator tuy mô phỏng iPhone nhưng vẫn còn một số điểm hạn chế như không có camera, không có microphone ... Hơn nữa bạn sẽ có được một cảm thấy hào hứng, phấn khởi khi nhìn thấy ứng dụng hoàn thiện của bạn chạy ổn định trên iPhone của bạn hay trên iPhone của bạn bè.

Do điều kiện cho phép còn nhiều hạn chế nên nội dung chương này chỉ giới thiệu đến bạn cách thức đưa ứng dụng lên iPhone đã jailbreak bằng công cụ JailCoder. Sau khi đọc xong chương này, bạn sẽ nắm được quá trình chuẩn bị trước khi đưa ứng dụng lên iPhone, cũng như cách thức từng bước để đưa ứng dụng lên iPhone bằng công cụ JailCoder.

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## 7.1 GIỚI THIỆU

Để build ứng dụng lên iPhone (máy thật), Apple yêu cầu lập trình viên phải có tài khoản Apple Developer và chi phí cho 1 tài khoản như vậy là 99\$/năm, vậy thì nếu chúng ta chưa có đủ điều kiện để sắm 1 tài khoản 99\$ liệu có cách nào để build ứng dụng lên iPhone. Jail Coder giúp chúng ta làm được điều này mà không phải mất tài khoản 99\$ và đương nhiên đây là một cách không chính thống do đó vẫn còn nhiều nhược điểm khi sử dụng Jail Coder. Hơn nữa để sử dụng được Jail Coder hiệu quả đòi hỏi iPhone phải được Jailbreak sẵn.

Bạn có thể tìm hiểu thêm về Jailbreak iPhone tại:

Tinhte: <http://www.tinhte.vn/forums/thay-doi-nang-cap-firmware.163/>

GSM: <http://gsm.vn/forums/firmware-jailbreak-unlock.502/>

Heaveniphone: <http://heaveniphone.com/forums/22-iphone-hoi-dap-thac-mac-phan-mem.html>

## 7.2 QUÁ TRÌNH CHUẨN BỊ

**Bước 1:** Download công cụ **Jail Coder** tại <http://oneiros.altervista.org/jailcoder/>. Jail Coder được cài đặt trên các hệ điều hành OSX như Mac OSX Leopard, Snow Leopard, Lion, Lion Mountain...

**Bước 2:** Thoả các yêu cầu sau:

- Sử dụng Xcode 3 đến Xcode 4.
- Đã cài đặt **AppSync**, vào **Cydia** để tìm và cài đặt phần mềm AppSync. Lưu ý nên cài đặt AppSync tương đương với phiên bản iOS chúng ta đang sử dụng.

**Ví dụ:** iPhone đang dùng iOS 6.x.x thì sẽ cài “*AppSync for iOS6*”.



Hình 7.1 AppSync

- iPhone dùng để đưa ứng dụng lên phải là iPhone đã được Jailbreak, những iPhone đã Jailbreak hầu hết đều có icon Cydia trên giao diện home của iPhone.



Hình 7.2 Biểu tượng Cydia trên màn hình

- Thoát hoàn toàn XCode và các iOS Simulator đang chạy trước khi tiến hành Jail Coder.
- Phiên bản iOS đang sử dụng trên iPhone phải lớn hơn hoặc bằng phiên bản iOS mà ứng dụng chúng ta đang sử dụng.

**Ví dụ:** iPhone đang sử dụng iOS 6, bạn muốn build ứng dụng nào đó lên iPhone thì ứng dụng đó phải sử dụng iOS SDK 6 trở xuống.

### 7.3 TIẾN HÀNH

**Bước 1:** mở Jail Coder → Click Guided Path.



Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013  
**Hình 7.3 Guided Patch**

Click “Got it”

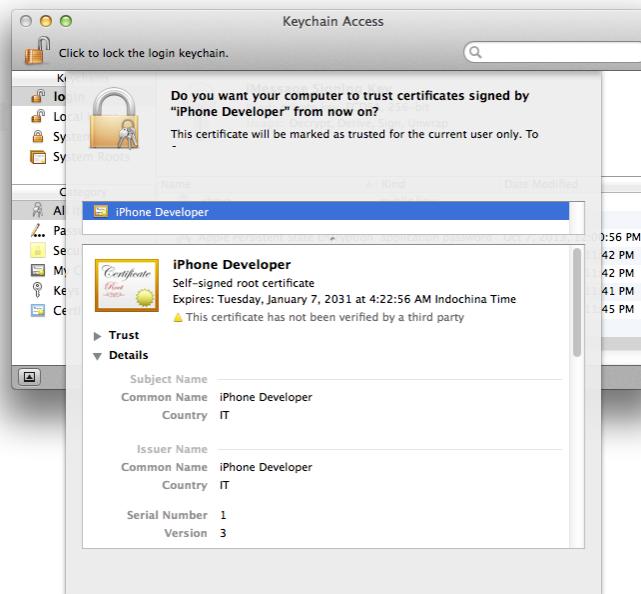


**Hình 7.4 Got it**

Click “Certificate Root” → xuất hiện cửa sổ “Keychain Access”



Hình 7.5 Certificate Root



Hình 7.6 Keychain Access

Click “Always Trust” để trust Certificate “iPhone Developer”, cửa sổ yêu cầu nhập password xuất hiện, nhập password hiện tại của máy Mac → chọn “Update eSetting”.



**Hình 7.7 Nhập mật khẩu**

Quay lại cửa sổ “Jail Coder” click “**Certificate Private**”, tương tự nhập password (3 lần), sẽ hiển thị thông báo lỗi nhưng không sao → click **Ok**.



**Hình 7.8 Certificate Private**



**Hình 7.9 Nhập mật khẩu**



**Hình 7.10 Thông báo lỗi**

Nếu thành công chúng ta sẽ thấy Certificate “iPhone Developer” đã được thêm vào Keychain Access như sau:



Nguyễn Anh Tiệp - Cao Thành Vàng © 2013

**Hình 7.11 Certificate iPhone Developer**

Quay lại cửa sổ “Jail Coder” → click “Next”



**Hình 7.12 Tại cửa sổ Jailcoder click Next**



**Hình 7.13 Patch my Xcode**

Click “**Patch my Xcode**” xuất hiện cửa sổ yêu cầu nhập password.



Nguyễn Anh Tiên - Cao Thành Vàng © 2013  
**Hình 7.14 Nhập mật khẩu**

Nếu thành công sẽ xuất hiện giao diện như sau



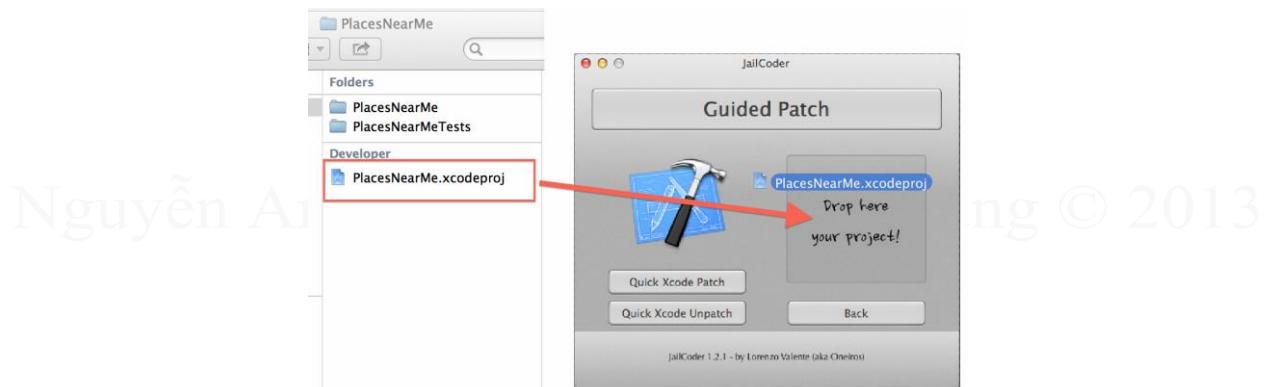
**Hình 7.15 Giao diện Patch thành công**

Click “**Back to Main Menu**” để patch project mà chúng ta cần build lên iPhone.



**Hình 7.16 Tại giao diện chính chọn Patch my Project**

Click “Patch My Project” sau đó kéo thả file “\*.xcodeproj” vào khung “Drop here your project!”



**Hình 7.16 Kéo thả project vào**

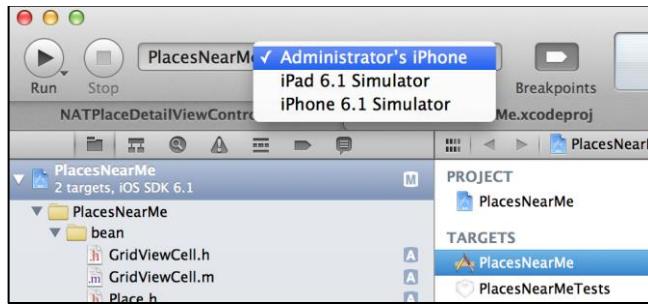
Nếu thành công chúng ta sẽ thấy JailCoder thông báo “**Patched!**”



**Hình 7.17 Patch thành công**

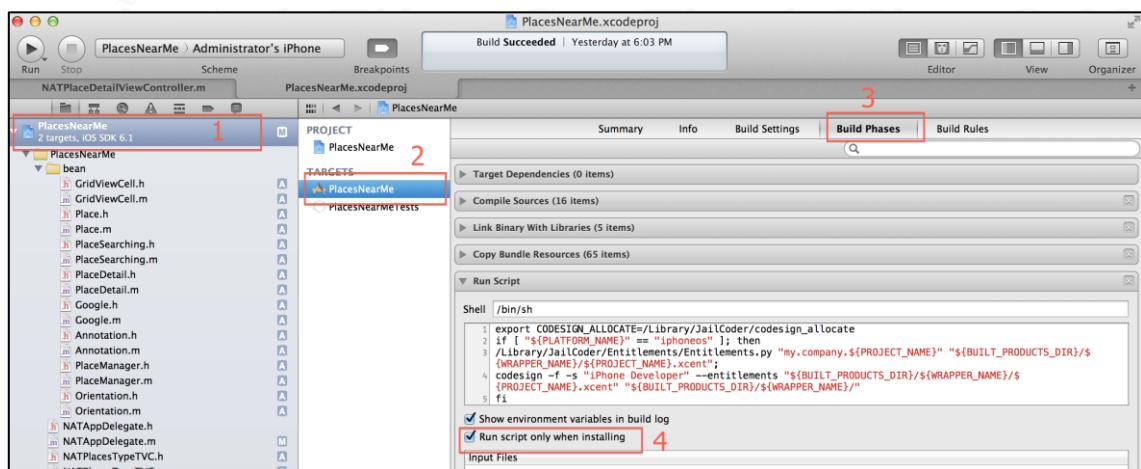
**Bước 2:** Kết nối iPhone vào máy tính, nếu iPhone đặt password thì phải nhập password để mở iPhone hoặc tốt hơn nên bỏ password của iPhone.

**Bước 3:** Mở Project mà chúng ta đã path trước đó, chờ khoảng 1 phút (để check ID và những thứ linh tinh) sẽ thấy xuất hiện thiết bị của chúng ta.



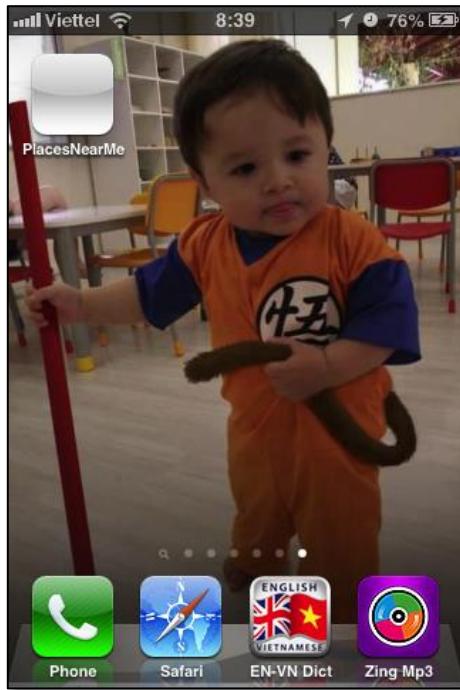
Hình 7.18 Thiết bị iPhone xuất hiện trong Xcode

**Bước 4:** Mở “Build Phases” → tab “Run Script” → check vào “Run script only when installing”



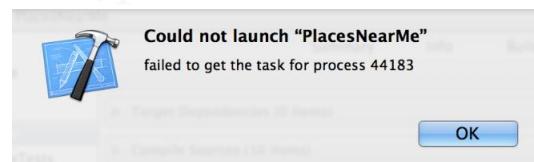
Hình 7.19 Chọn Run script trong Build Phases

**Bước 5:** Run ứng dụng, nếu thành công sẽ thấy ứng dụng của chúng ta trên iPhone.



**Hình 7.20** Ứng dụng Run thành công sẽ có iCon trên màn hình

Lưu ý: nếu xuất hiện thông báo “Launch” như sau thì không sao hết.  
Nguyễn Anh Tiệp - Cao Thành Vang © 2013



**Hình 7.21** Thông báo xuất hiện nhưng không ảnh hưởng

## CHƯƠNG VIII

### MỘT SỐ VẤN ĐỀ KHÁC

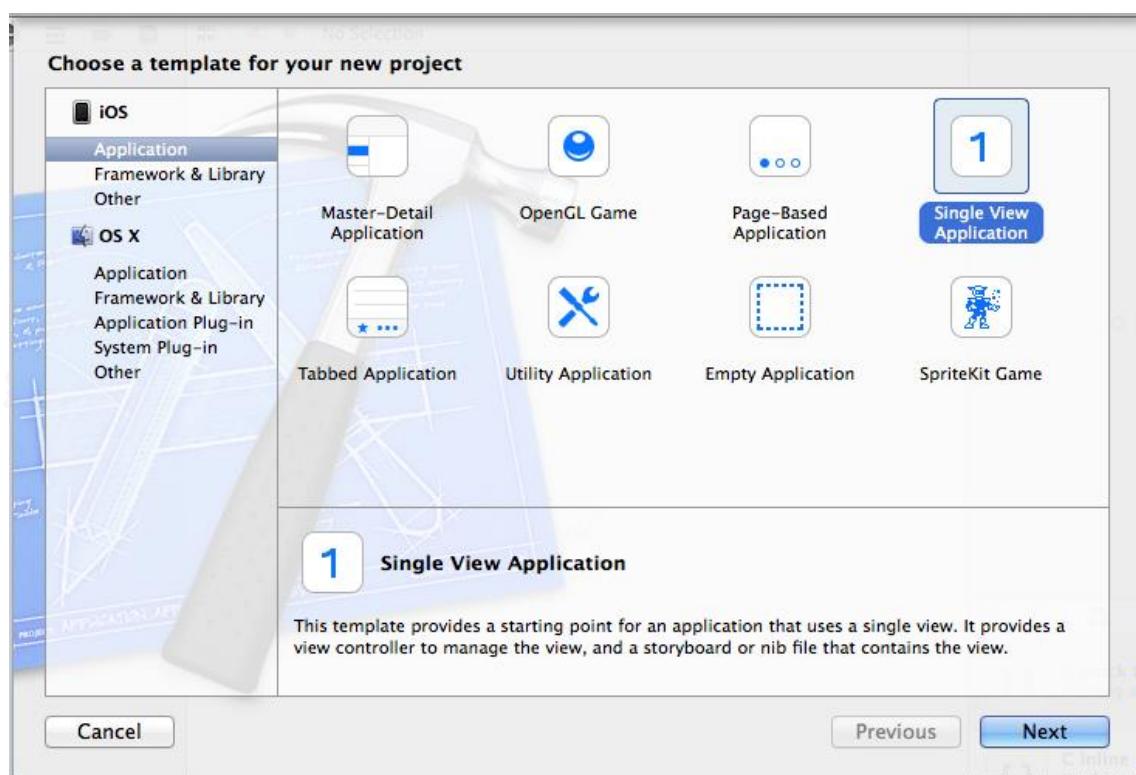
Mặc dù iOS 7 đã chính thức được đưa vào sử dụng và cho phép nâng cấp lên từ các phiên bản iOS trước đó. Tuy nhiên vì một số lí do như điều kiện kinh tế, iOS 7 còn lỗi, chưa có jailbreak ... nên người dùng vẫn còn dành sự ưu ái cho phiên bản iOS cũ hơn. Do đó khi bạn tiến hành xây dựng một ứng dụng cho iPhone, bạn phải suy nghĩ xem ứng dụng của bạn sẽ hỗ trợ cho phiên bản iOS nào hay sẽ hỗ trợ cho cả hai. Nếu ứng dụng của bạn hỗ trợ được cả iOS 7 lẫn các phiên bản thì đó là một điều tuyệt vời. Ứng dụng có thể tương thích được nhiều đời máy, nhiều phiên bản iOS, tiếp cận được nhiều người dùng hơn.

Tuy nhiên bạn sẽ đặt câu hỏi làm sao để ứng dụng của bạn có thể tương thích được như vậy? Bạn đừng quá lo lắng về việc này, Xcode đã hỗ trợ sẵn cho bạn. Các vấn đề được trình bày trong chương này sẽ giúp bạn hiểu hơn về cách thức để viết một ứng dụng cho phiên bản iOS cũ bằng công cụ Xcode 5, đồng thời nội dung chương cũng giúp bạn khám phá tính năng hỗ trợ xây dựng ứng dụng có thể chạy song song iOS 7 và các phiên bản iOS cũ hơn mà Xcode 5 mang đến.

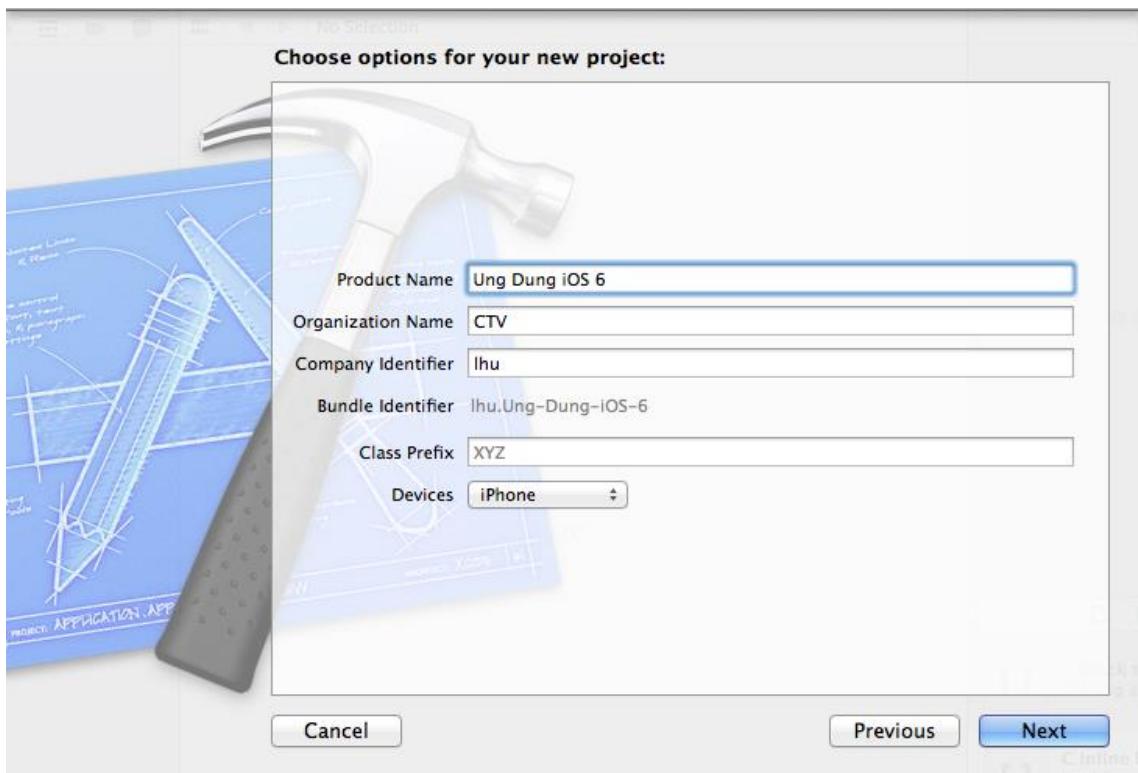
## 8.1 XÂY DỰNG ỨNG DỤNG CHO IOS 6 - IOS 6.1 TRÊN XCODE 5

Xcode 5 được sử dụng để viết ứng dụng cho các nền tảng iOS 6 và mới hơn (iOS 7). Tuy nhiên sau khi cài được iOS 6 SDK cho iOS Simulator thì bạn vẫn chưa thể dùng Xcode 5 để viết ứng dụng cho iOS 6 cũng như chạy thử chương trình trên iOS Simulator với iOS 6 SDK. Bạn cần phải thông qua một số thao tác tinh chỉnh cho project ứng dụng trước khi có thể viết chương trình cho iOS 6 và chạy được trên iOS Simulator với iOS 6 SDK. Sau đây là ví dụ minh họa việc xây dựng ứng dụng cho iOS 6 bằng Xcode 5.

Trước tiên các bạn **tạo mới một project**, ở đây project có tên là “*Ung dung iOS 6*”.

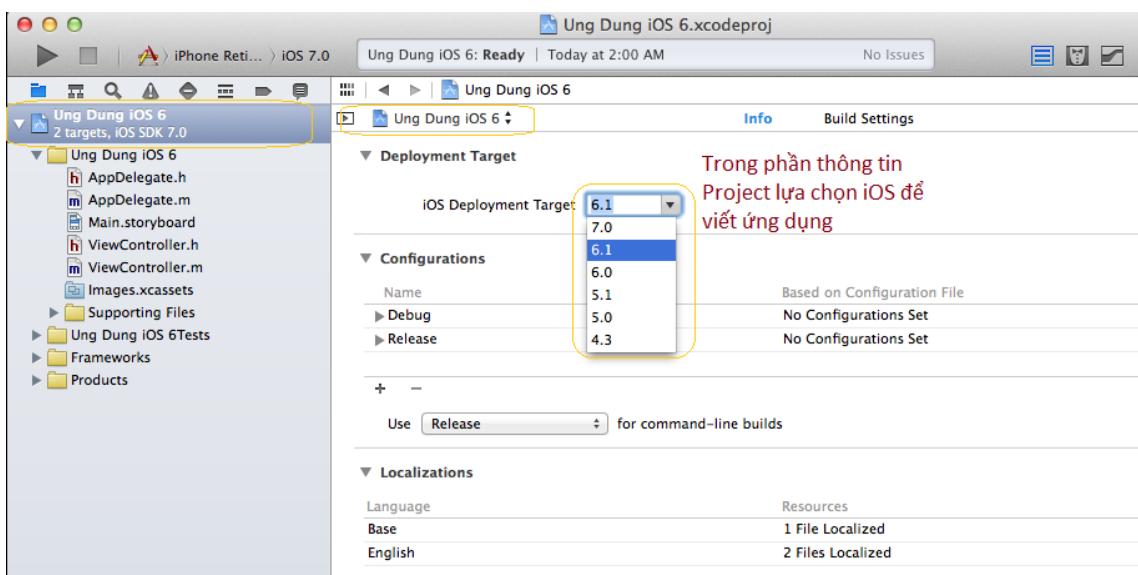


**Hình 8.1 Tạo Project mới**



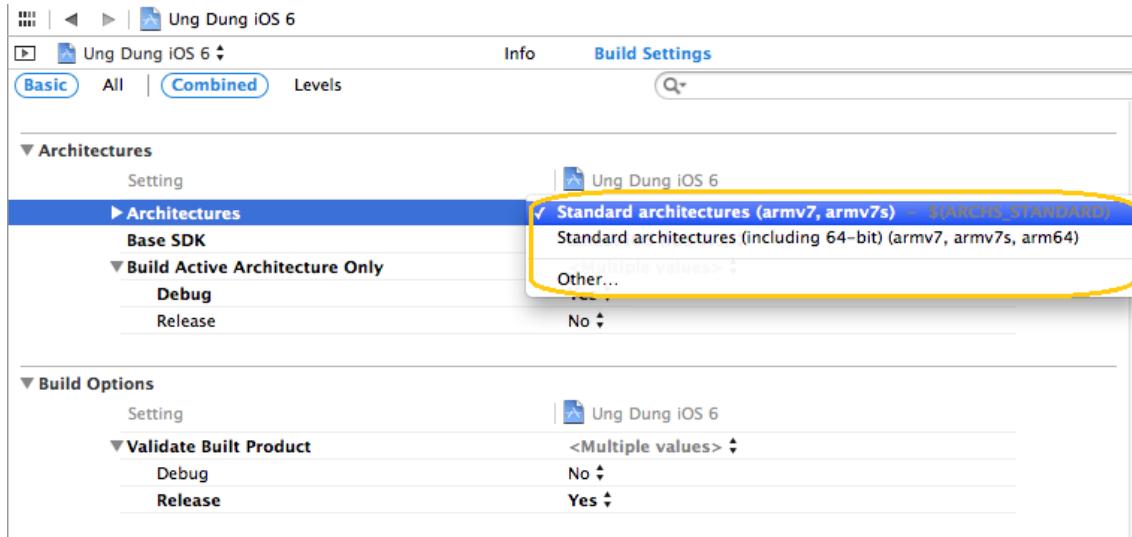
**Hình 8.2 Điện thông tin Project**

Trong phần thông tin của Project, tại mục **iOS Deployment Target**, lựa chọn phiên bản iOS cho ứng dụng.



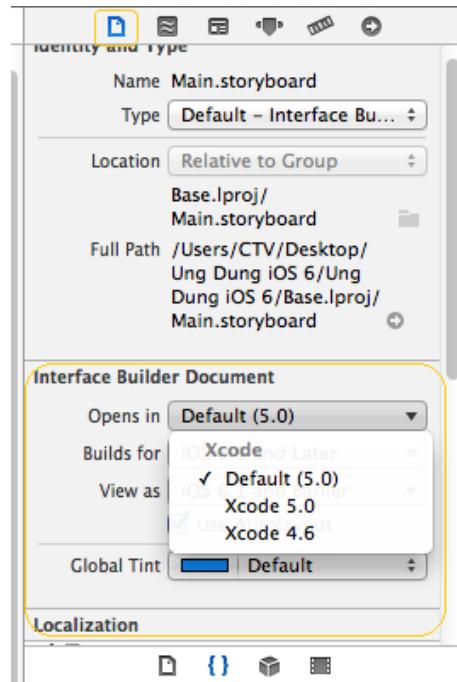
**Hình 8.3 Lựa chọn phiên bản iOS cho project**

Trong phần **Build Settings**, mục **Architectures**, bạn chọn **Standard architectures (armv7, armv7s)** để có thể chạy được ứng dụng trên iOS Simulator với iOS 6. Vì **mặc định** của Xcode 5 sẽ là **Standard architectures (including 64-bit)** để hỗ trợ bộ xử lý mới của Apple cho iPhone với công nghệ 64bit.



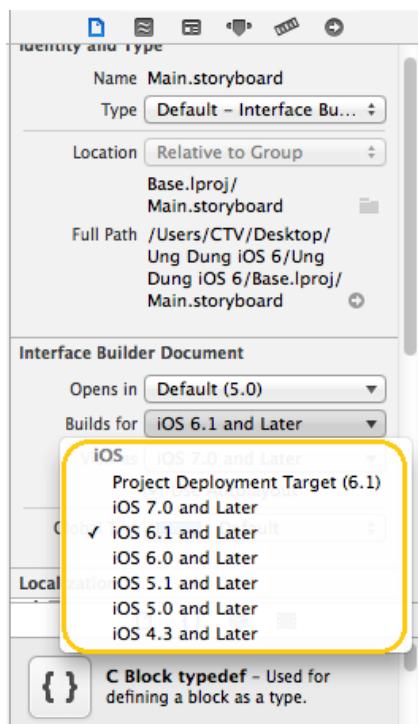
**Hình 8.4 Chọn Standard architecture (armv7, armv7s)**

Chọn tập tin Storyboard, bên Utility area, chọn **File inspector**. Tại mục **Interface Builder Document**, mặc định của project là được mở bằng Xcode 5, nếu bạn muốn mở lên bằng Xcode 4 thì bạn lựa chọn Xcode 4 trong mục **Opens in**.



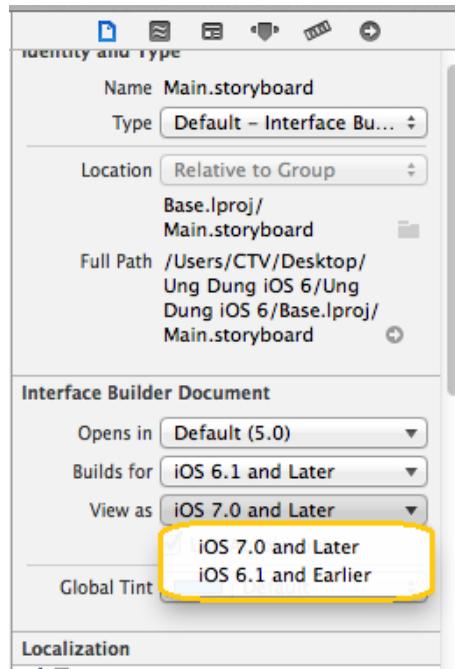
Hình 8.5 Lựa chọn phiên bản Xcode để mở project

Tại mục **Build for**, bạn chọn **iOS 6.1 and Later** để build ứng dụng cho iOS 6 hoặc mới hơn.



Hình 8.6 Chọn iOS 6.1 and Later

Tại mục **View as**, bạn lựa chọn chế độ xem của Storyboard khi thiết kế giao diện.



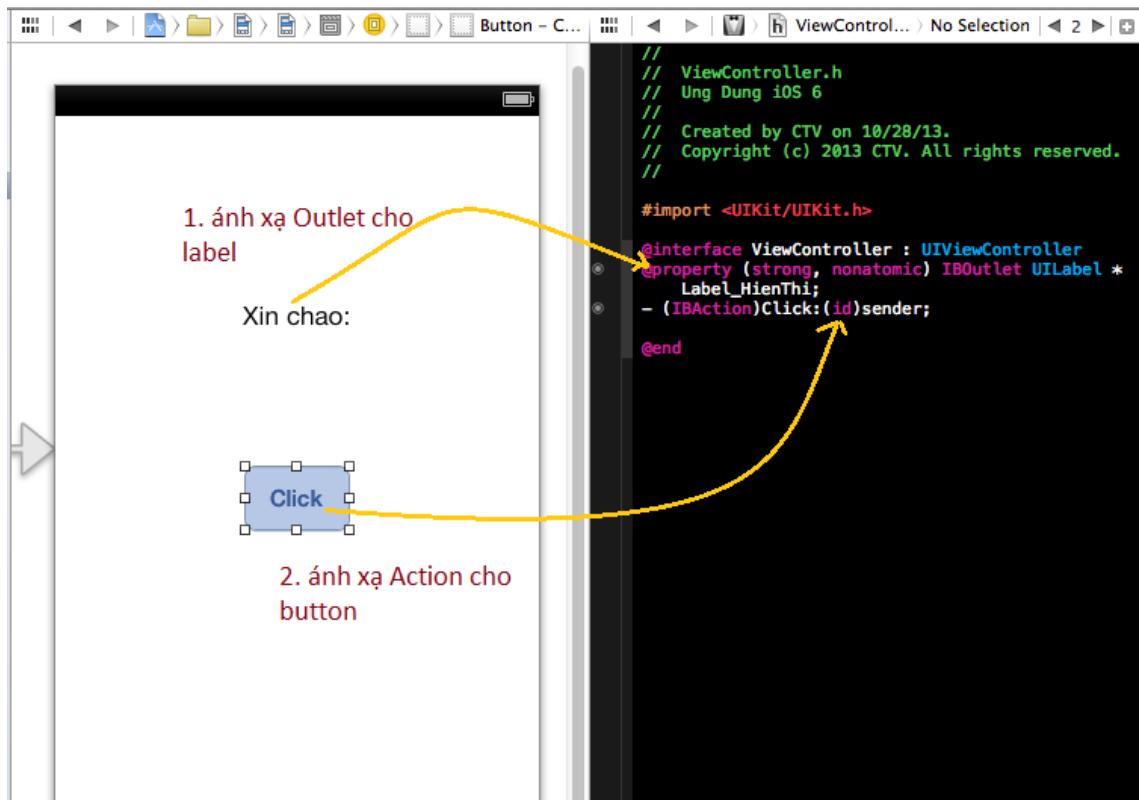
**Hình 8.7 Lựa chọn cách hiển thị của giao diện khi thiết kế**

Sau khi thiết lập xong cho project thì các bạn có thể thiết kế giao diện và viết code cho ứng dụng một cách bình thường. Các bạn thiết kế giao diện gồm một Button, một Label.



**Hình 8.8 Thiết kế giao diện ứng dụng**  
401

Tiếp theo tiến hành ánh xạ các đối tượng vào chương trình.



Nguyen Anh Tiep - Cao Thanh Vang © 2013

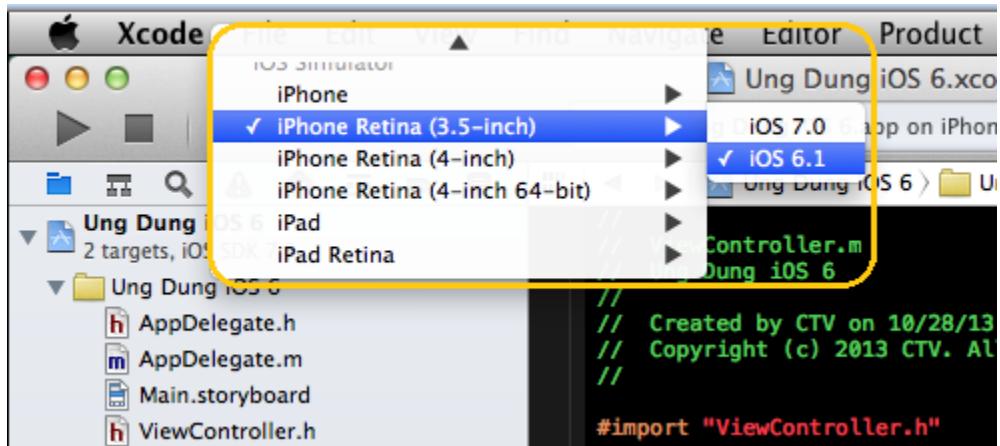
### Hình 8.9 Ánh xạ các đối tượng

Sau khi ánh xạ xong, tiến hành viết code cho sự kiện của button khi người dùng chạm vào. Khi người dùng chạm vào button thì label sẽ hiển thị nội dung “Xin chao ban da den voi ung dung tren iOS 6 viet bang Xcode 5”.

```
- (IBAction)Click:(id)sender {
    _Label_HienThi.text = @"Xin chao ban da den voi ung dung tren iOS 6 viet bang Xcode 5";
}
```

### Hình 8.10 Viết code cho sự kiện

Chọn phiên bản iOS 6 cho iOS Simulator rồi chọn Run.



**Hình 8.11 Chạy ứng dụng trên iOS Simulator iOS 6.1**

Kết quả khi chạm vào button.



**Hình 8.12 Giao diện iOS Simulator khi chạy ứng dụng**

## 8.2 XÂY DỰNG ỦNG DỤNG HỖ TRỢ NHIỀU VERSION IOS

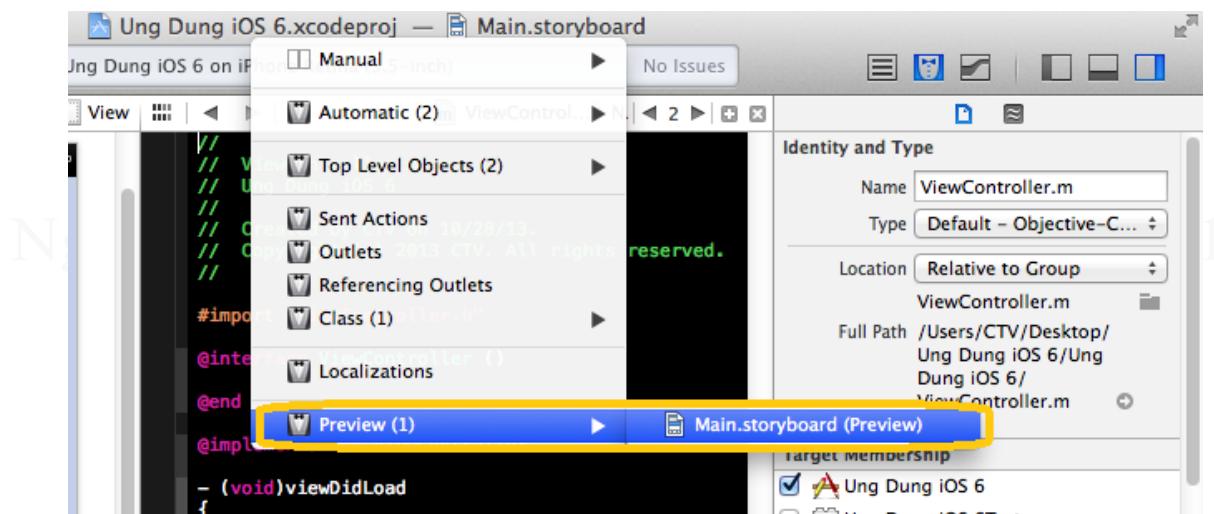
Mặc dù iOS 7 đã chính thức ra mắt, tuy nhiên số lượng lớn người dùng vẫn đang còn ở iOS 6 chứ chưa hoàn toàn nâng cấp lên hết. Do đó khi viết ứng dụng, điều đặt ra là bạn viết ứng dụng cho phiên bản iOS nào, nếu bạn viết cho iOS 7 thì ứng dụng chỉ chạy được trên iOS 7, nếu bạn viết cho iOS 6 thì ứng dụng chỉ chạy được trên iOS 6. Câu hỏi

đặt ra là liệu có thể viết ứng dụng có thể chạy trên iOS 7 lẫn iOS 6 hay không, chỉ có thể mới tiếp cận được nhiều người dùng ? **Câu trả lời là có.**

*Bạn có thể tham khảo thêm về việc hỗ trợ iOS 6 và iOS 7 tại đây.*

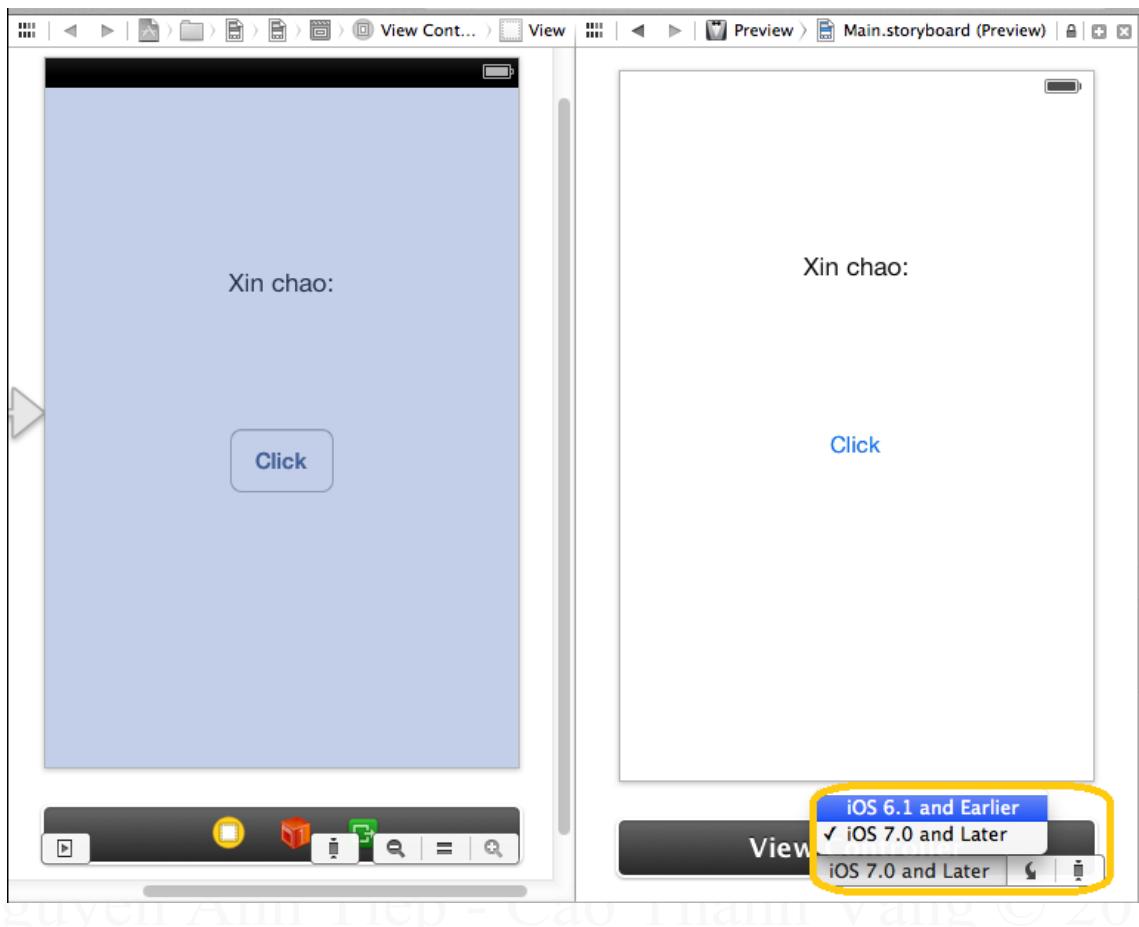
[https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/TransitionGuide/index.html#/apple\\_ref/doc/uid/TP40013174](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/TransitionGuide/index.html#/apple_ref/doc/uid/TP40013174)

Cũng với ứng dụng minh họa trong phần trên, bạn chọn **Assistant Editor** (  ) để chia đôi Editor area cho **dễ quan sát**. Trong menu của Assistant Editor, bạn chọn **Preview > chọn Storyboard** để xem giao diện ứng dụng trên khung Editor vừa mới mở ra.



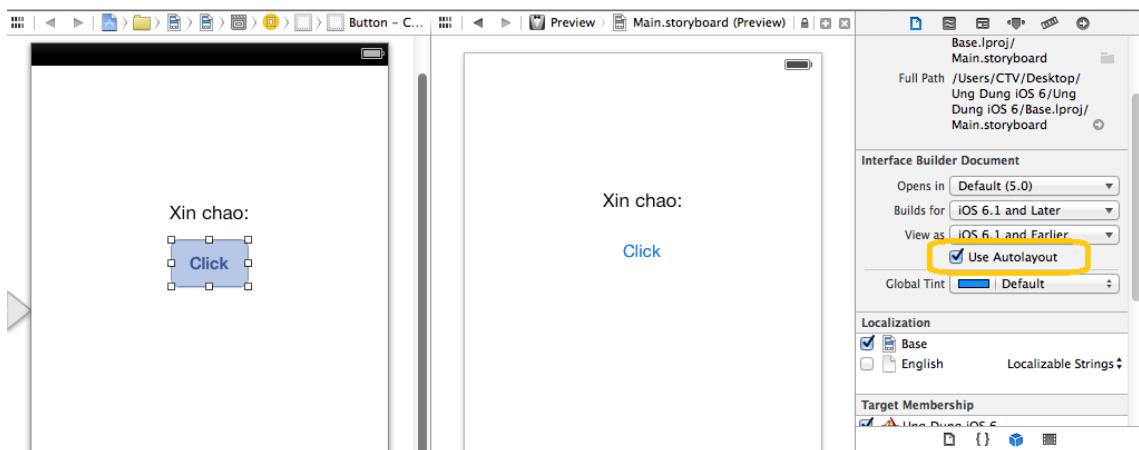
**Hình 8.13 Chọn Preview > Storyboard**

Tại cửa sổ giao diện mới này, bạn chọn xem với chế độ **iOS 7 and Later**. Như vậy là bạn có thể xem được giao diện của ứng dụng ở chế độ iOS 6 và iOS 7 cùng lúc. Lúc này bạn có thể thiết kế giao diện và tinh chỉnh sao cho ổn định về giao diện của iOS 6 lẫn iOS 7.



**Hình 8.14 Xem giao diện với chế độ iOS 7.0 and Later**

Trong project, bạn đánh dấu vào **Use Autolayout** để ứng dụng **tự động nhận diện** phiên bản iOS và sẽ cho ra **giao diện tương thích** với từng phiên bản.



**Hình 8.15 Chọn Use Autolayout**

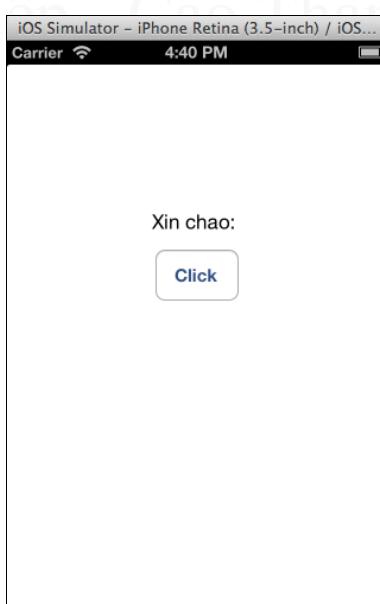
Giao diện khi chạy bằng iOS Simulator với iOS 7.



**Hình 8.16 Giao diện khi chạy bằng iOS Simulator với iOS 7**

Giao diện khi chạy bằng iOS Simulator với iOS 6.1

Nguyễn Anh Tiệp - Cao Thắng Vàng © 2013



**Hình 8.17 Giao diện khi chạy bằng iOS Simulator với iOS 6.1**

## CÂU HỎI THƯỜNG GẶP

### ✚ **Tự học lập trình iPhone có được không?**

Bạn có thể tự học lập trình iPhone, tuy nhiên thời gian sẽ lâu hơn và đòi hỏi bạn phải kiên nhẫn, tìm hiểu nhiều từ các tài liệu, video, internet.

### ✚ **Người chưa biết gì về lập trình có học viết ứng dụng iPhone được không?**

Bạn chưa biết gì về lập trình vẫn có thể học viết ứng dụng cho iPhone như bao người khác chỉ cần bạn có đam mê, siêng năng học và thực hành, tìm hiểu tài liệu. Tuy nhiên nếu có kiến thức cơ bản về lập trình bạn sẽ tiếp thu nội dung nhanh hơn

### ✚ **Không có iPhone có học lập trình iPhone được không?**

Không có iPhone thì bạn vẫn có thể học lập trình iPhone bình thường, ứng dụng bạn xây dựng được có thể chạy thử trên công cụ giả lập iPhone mà Apple đã cung cấp kèm theo Xcode là iOS Simulator, tuy nhiên iOS Simulator vẫn còn một số hạn chế mà chỉ có iPhone thật mới có như camera. Tuy nhiên bạn có thể tìm mua iPhone cũ chạy iOS 6 để học lập trình, giá tương đối rẻ.

### ✚ **Không có máy Mac có thể lập trình iPhone hay không?**

Bạn có thể lập trình iPhone mà không cần tới máy Mac. Bạn có thể xem [Chương 1, phần 1.1](#).

### ✚ **Sử dụng hệ điều hành Windows, làm sao cài được Xcode để lập trình iPhone?**

Bạn có thể cài Mac OS lên máy ảo rồi cài đặt Xcode bình thường. Như vậy bạn có thể lập trình iPhone trên Window rồi. Bạn có thể xem thêm tại [Chương 1, phần 1.1](#).

### ✚ **Ngôn ngữ nào được dùng để viết ứng dụng iPhone?**

Ứng dụng iPhone thường được viết bằng các ngôn ngữ như Objective-C, C++, C hoặc Java tuy nhiên ngôn ngữ Objective-C vẫn là ngôn ngữ chính được sử dụng nhiều nhất.

## **Sử dụng công cụ gì để viết ứng dụng iPhone?**

Để viết ứng dụng iPhone, bạn sử dụng công cụ Xcode do chính Apple cung cấp. Ứng dụng này được tải miễn phí từ App Store của Mac OS. Bạn xem thêm tại [Chương 1, phần 1.2](#).

## **Muốn học về lập trình iOS phải bắt đầu từ đâu?**

Muốn học lập trình iOS, trước hết bạn phải chuẩn bị được hệ điều hành Mac OS, phần mềm Xcode (xem [Chương 1](#)). Tiếp theo bạn tìm hiểu cách sử dụng Xcode (xem [Chương 2](#)), tìm hiểu Objective-C (xem [Chương 3](#)). Sau khi đã chuẩn bị xong, bạn tìm hiểu các đối tượng trong Xcode (xem [Chương 4](#), [Chương 5](#)) và tìm hiểu thêm các tài liệu, video, website được cung cấp ở [Phu lục](#). Sau khi trải qua quá trình này, bạn đã có được lượng kiến thức tương đối để tiếp tục tìm hiểu sâu hơn.

## **Nên đọc tài liệu gì khi bắt đầu tìm hiểu lập trình iOS?**

Khi bắt đầu tìm hiểu lập trình iOS, bạn nên đọc trước tài liệu do Apple cung cấp tại <https://developer.apple.com/library/ios/documentation/> để nắm được một số kiến thức cơ bản. Ngoài ra bạn cũng nên tìm hiểu các tài liệu, video, website được cung cấp trong phần [Phu Lục](#).

## **Xcode là gì?**

Xcode là phần mềm Apple cung cấp dùng để lập trình ứng dụng iPhone. Bạn xem tại [Chương 1, phần 1.2](#).

## **Objective-C là gì?**

Objective-C là ngôn ngữ được phát triển dựa trên ngôn ngữ C. Ngôn ngữ Objective-C được sử dụng là ngôn ngữ chính trong lập trình ứng dụng iPhone. Bạn xem thêm tại [Chương 3](#).

## **Có thể sử dụng Xcode 5 để viết ứng dụng cho các phiên bản iOS cũ hay không?**

Bạn vẫn có thể sử dụng Xcode 5 để viết ứng dụng cho các phiên bản iOS cũ. Xem tại [Chương 8, phần 8.1](#).

#### **Làm sao để ứng dụng có thể vừa chạy được iOS 7 vừa chạy được iOS cũ?**

Xcode 5 đã hỗ trợ cơ chế tự động hiển thị giao diện theo phiên bản iOS cho ứng dụng, nếu ứng dụng chạy trên iOS 7 sẽ có giao diện khác, chạy trên iOS 6 sẽ có giao diện khác. Xem tại [Chương 8, phần 8.2](#).

#### **Làm thế nào để đưa ứng dụng mới lập trình lên iPhone?**

Nếu iPhone của bạn đã jailbreak, bạn có thể đưa ứng dụng lên iPhone dùng JailCoder, xem [Chương 7](#). Nếu iPhone chưa jailbreak, bạn cần có tài khoản Apple Developer, xem thêm tại:

[https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html#/apple\\_ref/doc/uid/TP40012582-CH1-SW1](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html#/apple_ref/doc/uid/TP40012582-CH1-SW1) .

#### **Làm sao để chia sẻ ứng dụng mới lập trình cho người khác?**

Nếu người bạn muốn chia sẻ ứng dụng có iPhone đã jailbreak, bạn có thể dùng JailCoder để đưa ứng dụng lên iPhone. Nếu bạn có tài khoản Developer ID bạn có thể chuyển ứng dụng sang dạng .ipa rồi chia sẻ.

#### **Làm thế nào để đưa ứng dụng lên App Store?**

Để đưa ứng dụng lên App Store, bạn cần có một tài khoản Apple Developer với mức phí 99\$/năm. Bạn có thể xem thêm hướng dẫn tại

[https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html#/apple\\_ref/doc/uid/TP40012582-CH1-SW1](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html#/apple_ref/doc/uid/TP40012582-CH1-SW1)

#### **Developer ID là gì? Có những loại Developer ID nào?**

Apple Developer ID là tài khoản dành cho các lập trình viên của Apple. Với tài khoản này, bạn có thể đưa ứng dụng lên App Store, chạy ứng dụng lên iPhone mà không cần JailCoder, hơn nữa bạn sẽ được cung cấp trước các thông tin từ Apple như các bản

thử nghiệm Xcode, thử nghiệm iOS mới... Apple Developer ID dành cho iOS có 3 loại chính: iOS Developer Program dành cho các lập trình viên với 99\$/năm, iOS Developer Enterprise Program dành cho các doanh nghiệp với 299\$/năm, iOS Developer University Program dành cho các trường đưa lập trình iOS vào giảng dạy và được miễn phí. Bạn có thể tìm hiểu thêm tại <https://developer.apple.com/programs/>

Nguyễn Anh Tiệp - Cao Thanh Vàng © 2013

## PHỤ LỤC

### Source Kèm Theo Tài Liệu

<https://www.mediafire.com/folder/pulecvu1ic48t/>

### Sách Bạn Nên Đọc

1. Apress, *Learn Objective-C on the Mac For OS X and iOS 2<sup>nd</sup>*.
2. Aaron Hillegass , *Objective-C Programming: The Big Nerd Ranch Guide.*
3. Apress, *iPhone and iPad Apps for Absolute Beginners 3<sup>rd</sup>*.
4. O'Reilly , *Head First iPhone & iPad Development.*
5. O'Reilly , *iPhone App Development: The missing manual.*
6. Neal Goldstein & Dave Wilson, *iOS 6 Application Development For Dummies.*
7. Neal Goldstein , *iPhone Application Development For Dummies.*

### Video Bạn Nên Xem

1. <http://www.youtube.com/user/MilmersXcode>
2. <http://www.youtube.com/playlist?list=PLA138EFCAFA592E7E>
3. <http://www.youtube.com/user/ChupaMobile>
4. <http://www.youtube.com/user/CarnegieMellonU?feature=watch>
5. <http://www.youtube.com/user/CodeWithChris>
6. <http://www.youtube.com/user/iThanhVN>
7. <https://itunes.apple.com/vn/course/developing-ios-7-apps-for/id733644550>
8. <https://itunes.apple.com/vn/course/iphone-application-programming/id727587146>

### Website Tiện Ích

1. <http://geekylemon.com/>

2. <http://www.appcoda.com/tutorials/>
3. <https://www.udemy.com/blog/how-to-build-an-iphone-app-from-scratch-for-non-technical-people/>
4. <http://www.idev101.com/learn/>
5. <http://www.mobioneer.com/6-iphone-app-development-tutorial.html/>
6. <http://nhatnghe.com/forum/forumdisplay.php?f=154>
7. <http://codewithchris.com/>
8. <http://www.chupamobile.com/tutorial-ios>
9. <http://www.iphoneapptuts.com/>
10. <http://www.facebook.com/LapTrinhiOS>
11. <http://iosclass.blogspot.in/>

Nguyễn Anh Tiệp - Cao Thành Vàng © 2013