

Table of Contents

[Chapter 1 Introduction 4](#_Toc318377020)

[Chapter 2 The Mobile SDK Components 5](#_Toc318377021)

[2.1 The Server Side 6](#_Toc318377022)

[2.2 The Client Side 7](#_Toc318377023)

[Chapter 3 The Mobile SDK Installation 8](#_Toc318377024)

[3.1 Installing the Server Side 9](#_Toc318377025)

[3.1.1 Installing the Sitecore Web Service 9](#_Toc318377026)

[3.1.2 Installing the Sitecore Mobile SDK Components 12](#_Toc318377027)

[3.2 Installing the Client Side 15](#_Toc318377028)

[3.2.1 Installing Sitecore Mobile SDK to a project 15](#_Toc318377029)

[3.2.2 Using the Sitecore Mobile SDK Sample Application 17](#_Toc318377030)

[3.2.3 Installing and Using the NicamApp Example Project 17](#_Toc318377031)

[Chapter 4 Getting Started 20](#_Toc318377032)

[4.1 Getting Started with the Embedded Browser 21](#_Toc318377033)

[4.2 Getting started with the WebAPI service 22](#_Toc318377034)

[4.3 Combining the Use of the Embedded Browser and Web API Service 25](#_Toc318377035)

[Chapter 5 Using Enhanced Web View Reference 27](#_Toc318377036)

[5.1 The Left-Right Swipe Link elements 28](#_Toc318377037)

[5.1.1 The *Mobile Swipes* Component for Sitecore 28](#_Toc318377038)

[5.2 Getting a Snapshot from the Camera 30](#_Toc318377039)

[5.3 Getting the Accelerometer Data 31](#_Toc318377040)

[5.4 Getting the Device Info 32](#_Toc318377041)

[5.5 Managing the Contacts 33](#_Toc318377042)

[5.5.1 Creating a Contact 33](#_Toc318377043)

[5.5.2 Finding a Contact 34](#_Toc318377044)

[5.5.3 Removing a Contact 35](#_Toc318377045)

[5.5.4 The *Mobile Add Contact* Component for Sitecore 35](#_Toc318377046)

[5.6 Sending a Message to Twitter 39](#_Toc318377047)

[5.6.1 The *Mobile Tweet This* Component for Sitecore 40](#_Toc318377048)

[5.7 Send a Message to Email 43](#_Toc318377049)

[5.7.1 The *Email* Component for Sitecore 44](#_Toc318377050)

[5.8 Using Javascript Native Alert 48](#_Toc318377051)

[Chapter 6 Manipulating an Item Using the Framework’s API 49](#_Toc318377052)

[6.1 Content API – first steps 50](#_Toc318377053)

[6.1.1 Installing a Sitecore Mobile SDK Framework to the Project 50](#_Toc318377054)

[6.1.2 Establishing an Anonymous or Authenticated Session to a Website 50](#_Toc318377055)

[6.1.3 Recommendations and Best Practices 51](#_Toc318377056)

[6.1.4 Accessing an Item 52](#_Toc318377057)

[6.1.5 Requirements to Access an Item 54](#_Toc318377058)

[6.1.6 Accessing the Children of an Item 55](#_Toc318377059)

[6.1.7 Using the children of the particular item to populate a tab bar 56](#_Toc318377060)

[6.2 Accessing the Fields of an Item 59](#_Toc318377061)

[6.3 Accessing the Different types of a Field 60](#_Toc318377062)

[6.3.1 The Image Field 60](#_Toc318377063)

[6.3.2 The Checkbox Field 61](#_Toc318377064)

[6.3.3 The Date and Datetime Fields 61](#_Toc318377065)

[6.3.4 The Color Picker Field 61](#_Toc318377066)

[6.3.5 The Checklist, Multilist and Treelist Fields 62](#_Toc318377067)

[6.4 Accessing the Parent of an Item 63](#_Toc318377068)

[6.5 Accessing Items Using the Sitecore Query 64](#_Toc318377069)

[6.6 Reading Paged Items Effieciently 65](#_Toc318377070)

[6.7 Accessing the Different Languages of an Item 66](#_Toc318377071)

[6.8 Using the Cache 67](#_Toc318377072)

[6.8.1 Merging Requests 67](#_Toc318377073)

[6.8.2 Reading Cached Items 67](#_Toc318377074)

[6.9 Modifying the Design of the *WebView* Navigation Bar 69](#_Toc318377075)

# Introduction

Sitecore Mobile SDK is a framework that is designed to help the developer produce iOS based applications that use and serve content that is managed by Sitecore.

This guide describes the Sitecore Mobile SDK Framework. It is useful for developers who are looking for information about the framework’s API.

This document contains the following chapters:

‎Chapter 1 — Introduction

‎Chapter 2 — The Mobile SDK Components

‎Chapter 3 — The Mobile SDK Installation

‎Chapter 4 — Getting Started

‎Chapter 5 — Using Enhanced Web View Reference

‎Chapter 6 — Manipulating an Item Using the Framework’s API

# The Mobile SDK Components

The Mobile SDK consists of two components:

* A Sitecore package that you must install on the server side.
* An iOS project template and libraries that you must install on the client side.

## The Server Side

The Sitecore package contains the Sitecore Web API that is a new web service designed with REST principles to make Sitecore content accessible through the HTTP requests in the JSON format.

The Web API supports the accessibility for the content through either the item paths and item IDs or performing queries. The output produced by the service is highly customizable and optimized to reduce the number of requests.

## The Client Side

The client side is an iOS project that contains the Sitecore Mobile SDK that provides a set of Objective-C APIs. You can use these APIs to access the Sitecore content through the web service and using the content in your application. The APIs also include advanced behaviors such as preloading and caching.

The Mobile SDK also provides a way of displaying your existing HTML presentation while providing access to native iOS hardware and software abilities with minor additional code. You can reuse the elements of the website and reduce the implementation cost. You can also use native Objective-C when you need.

The list of native features available in HTML environment includes:

* + Left-Right swipe navigation
  + Access to the camera and photo library
  + Accelerometer data
  + Device information
  + Access to the contact list and easy creation of the new contacts
  + Creating tweets through the system-wide twitter accounts in iOS5
  + Sending email messages using the email account in iOS5
* Other features

To produce iOS apps with or without Sitecore Mobile SDK you need to have an Apple OS X machine with the Apple developer tools, such as XCode. The Mobile SDK supports iOS5 and above.

# The Mobile SDK Installation

## Installing the Server Side

### Installing the Sitecore Web Service

The following steps describe how to install the Sitecore web service:

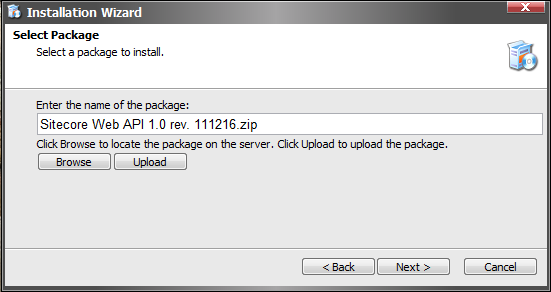
1. To install the Sitecore Web service on your site, use an Installation Wizard in the Sitecore backend. It can be accessed in two ways:
   * Launch Wizard button in the Install chunk:



* + Sitecore Development Tools Installation Wizard.

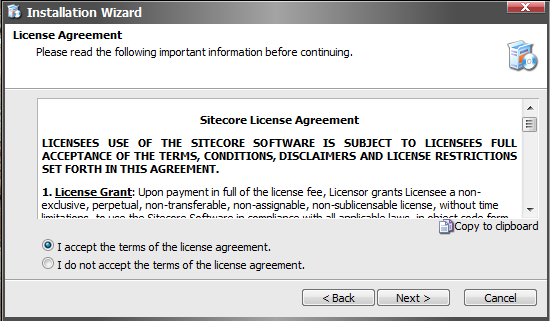
1. Follow the description of the wizard’s steps:

* Specify a package



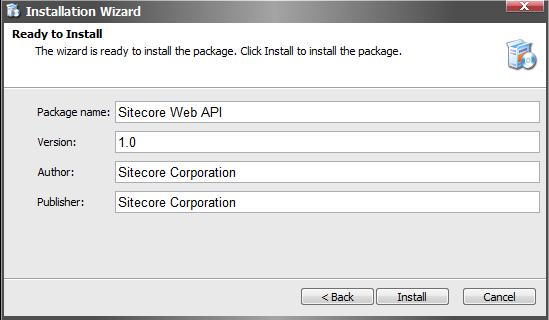
Click Browse to browse for an existing package or Upload to upload the new one. Packages are looked up in the folder specified in web.config.

* Accept the License Agreement:



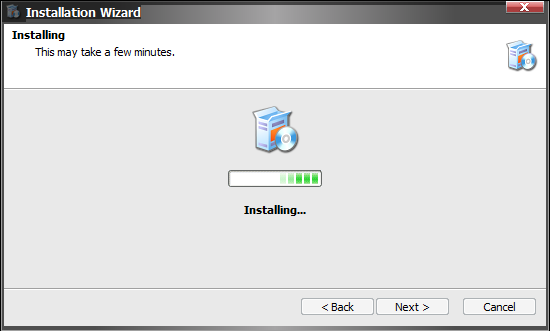
Review the License Agreement, accept it and click Next.

* Review the package information.



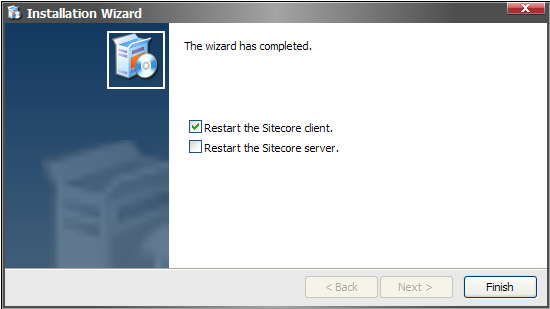
Review the package metadata and click Install.

* Installation process window



Wait while Installation Wizard continues the installation.

* Restart

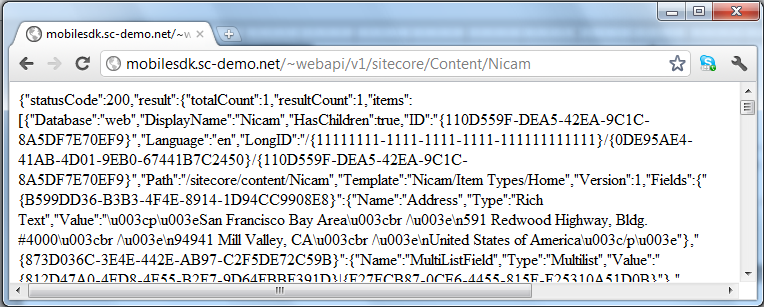


In the final window you can choose to restart Sitecore client or server.

* Make a simple request to the service using the browser to make sure it works:

Example:

http://yoursite/-/webapi/v1/sitecore/Content/Home

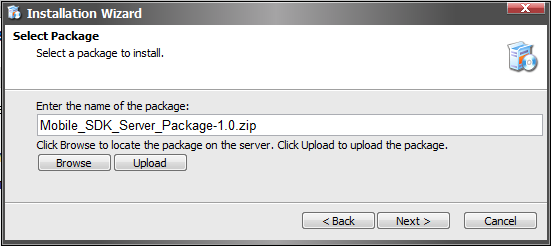


### Installing the Sitecore Mobile SDK Components

To install a Sitecore Mobile SDK components on you site use an Installation Wizard in the Sitecore backend.

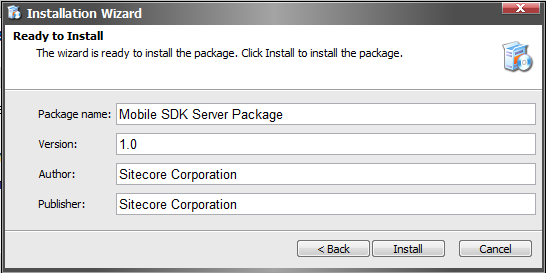
Follow description of the wizard’s steps to install the SDK components on the server side:

* Specify a package



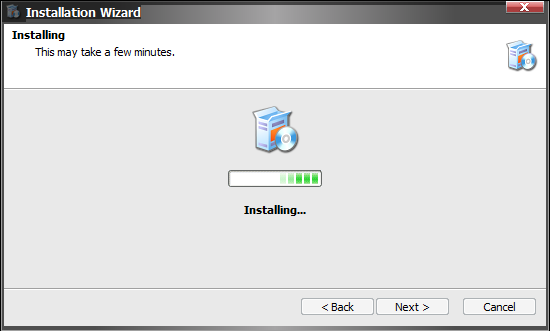
Click Browse to browse for an existing package or Upload to upload the new one.

* Review the package info



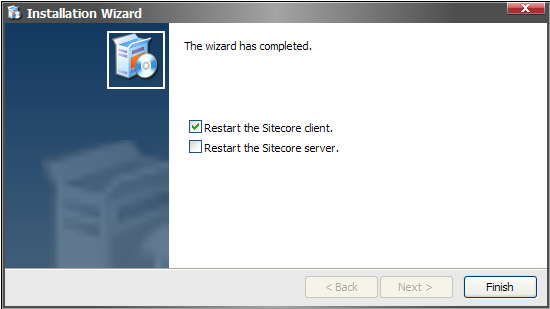
Review the package metadata and click Install.

* Installation process window



Wait while Installation Wizard completes the installation.

* Restart



In the final window you can choose to restart Sitecore client or server and then click Finish.

Now you can use the Sitecore Mobile SDK server side component.

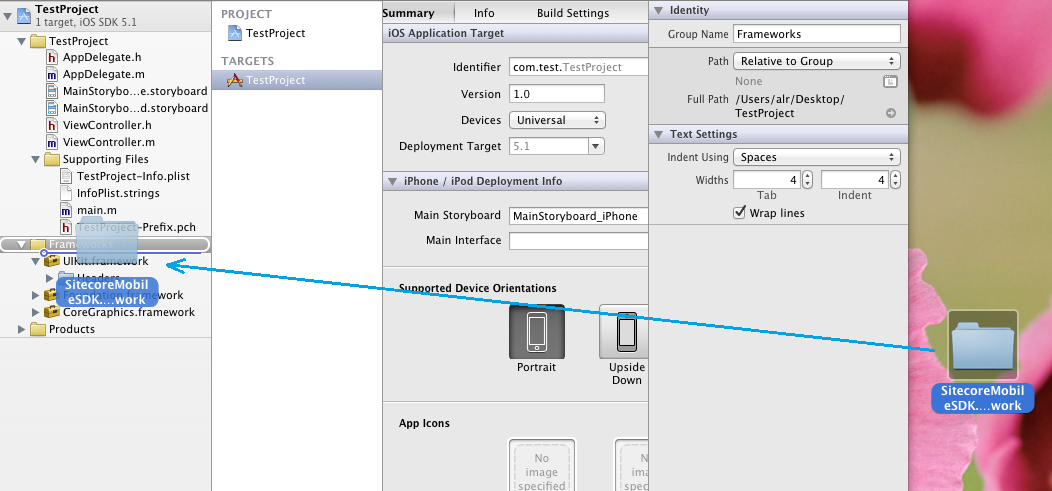
## Installing the Client Side

To start the client side installation, you must have the SitecoreMobileSDK.framework bundle.

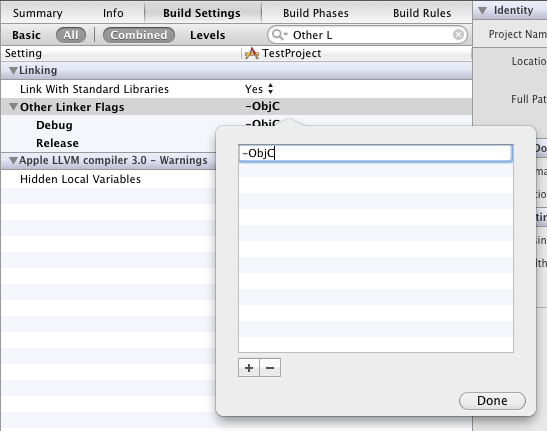
### Installing Sitecore Mobile SDK to a project

To use the Sitecore Mobile SDK framework in your application, use the following steps:

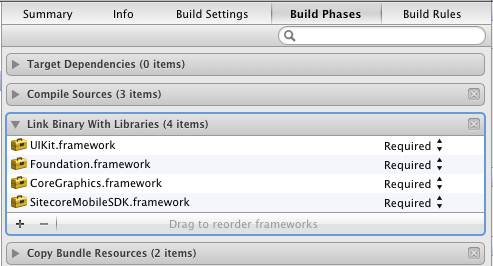
1. Create a simple XCode Single View Application project — for more details see the [Apple manual](http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iPhone101/Articles/01_CreatingProject.html). If you have an existing project, skip this step.
2. Drag-and-drop the SitecoreMobileSDK.framework bundle to the project's Frameworks source group.



1. Add an -ObjC flag to Other linker flag in the XCode Build Settings — for more details, see the [Apple manual](http://developer.apple.com/library/ios/#documentation/DeveloperTools/Reference/XcodeBuildSettingRef/1-Build_Setting_Reference/build_setting_ref.html).



1. Link the following frameworks: CFNetwork.framework, CoreMotion.framework, CoreLocation.framework, AddressBook.framework, AudioToolbox.framework, AddressBookUI.framework, Twitter.framework, MessageUI.framework to the project — for more details, see the [Apple manual](https://developer.apple.com/library/ios/#recipes/xcode_help-project_editor/Articles/AddingaLibrarytoaTarget.html).



1. Link following libraries: libxml2.dylib, libz.dylib, libsqlite3.dylib, libstdc++.dylib to the project — for more details, see the [Apple manual](https://developer.apple.com/library/ios/#recipes/xcode_help-project_editor/Articles/AddingaLibrarytoaTarget.html).
2. Add line - "#import <SitecoreMobileSDK/SitecoreMobileSDK.h>" to your project's \*.pch or GCC\_PREFIX\_HEADER file — for more details, see [Apple manual](http://developer.apple.com/library/mac/#documentation/DeveloperTools/Reference/XcodeBuildSettingRef/1-Build_Setting_Reference/build_setting_ref.html).

The following snippet is an example:

#ifdef \_\_OBJC\_\_

#import <SitecoreMobileSDK/SitecoreMobileSDK.h>

#import <UIKit/UIKit.h>

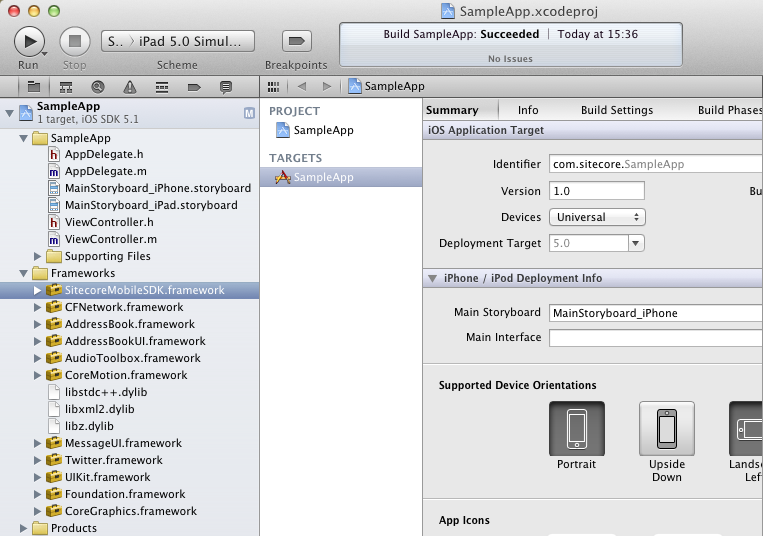
#import <Foundation/Foundation.h>

#endif

### Using the Sitecore Mobile SDK Sample Application

To build and use a sample application for the Sitecore Mobile SDK framework:

1. Open the SampleApp.xcodeproj project.
2. Drag-and-drop the SitecoreMobileSDK.framework bundle to the project's Frameworks source group.
3. Build the application.



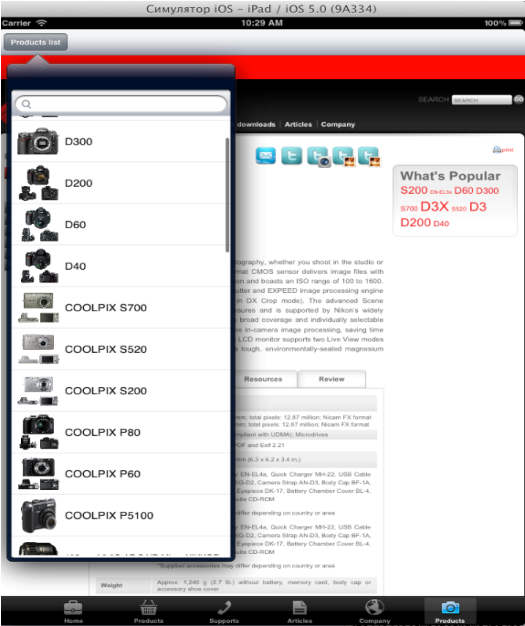
### Installing and Using the NicamApp Example Project

NicamApp project was created to demonstrate how to use the Sitecore Mobile SDK API and Embedded browser together to retrieve the Sitecore content and present it in an iOS application.

NicamApp contains TabBarController with site sections and embedded web browser for the site section in each tab.

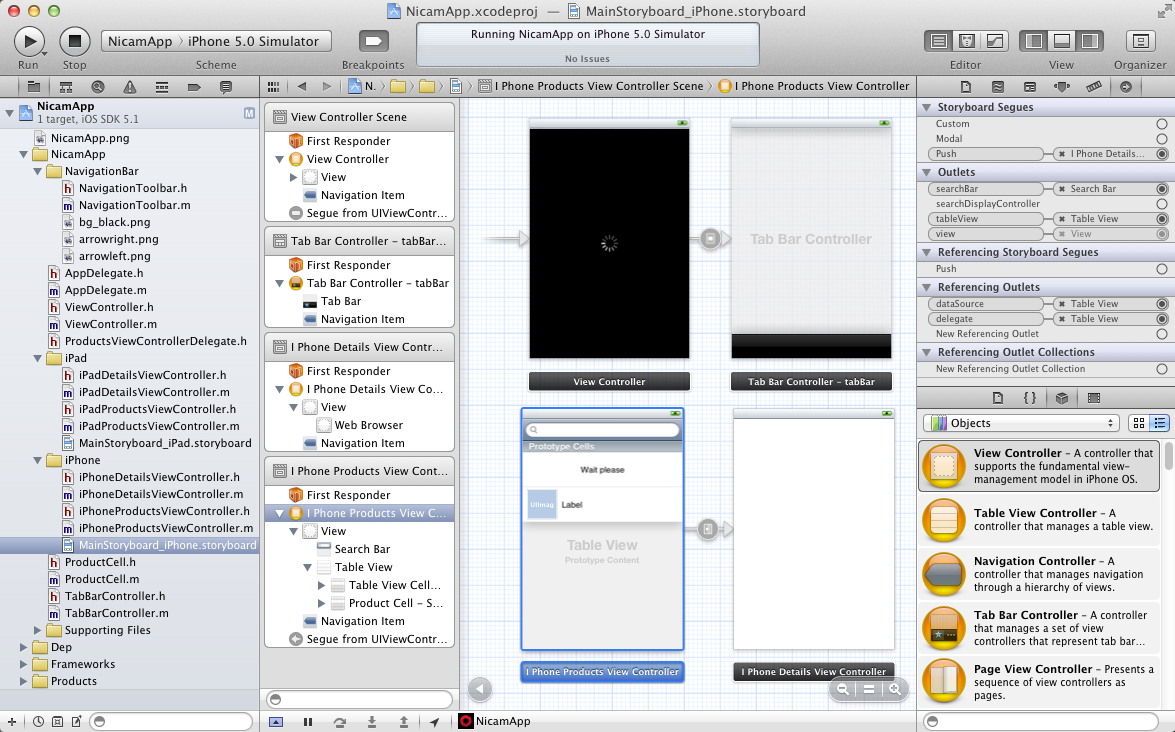
The last tab contains View Controller and Table View with all products of the site.

You can select one of them and view detailed information for this product in a web browser.

To install the NicamApp project:

1. Open the NicamApp.xcodeproj project.
2. Drag-and-drop the SitecoreMobileSDK.framework bundle to the project's Frameworks source group.
3. Build the application.



# Getting Started

There are two primary ways of making your applications with the Sitecore Mobile SDK. If you already have a website running on Sitecore and this website is or can be optimized for mobile, the simplest option to start with is to reuse the website presentation. You can achieve this by displaying an embedded browser window inside your app, while still being able to access native device features that are typically unavailable to a normal website through a set of html and javascript APIs. To get started using this approach, refer to the [Getting Started with the Embedded Browser](#_Getting_Started_with_2) section.

If you are familiar with Objective-C development then you might decide to code all or parts of your application using native code while still keeping the content in Sitecore to get all benefits of the Sitecore CMS. To learn more about Objective-C APIs, please refer to the [Getting Started with the Web API service](#_Getting_started_with_3) section.

And of course there’s plenty of room in between – you can use native code and UI elements for parts of the screen, such as standard navigational elements like tab bar and nav bar, while continuing to use the embedded browser to display your content. For an example of this approach, please refer to the [Combining the Use of the Embedded Browser and WebAPI Service](#_Combining_the_Use_1) section.

## Getting Started with the Embedded Browser

Sitecore Mobile SDK contains the SCWebView classes that extend WebView with additional futures like sharing via twitter or left-right swipes etc., see [Enhances WebView Reference](#_Enhanced_WebView_reference) for details. All methods of the SCWebView class are similar to UIWebView, so it can be used in the same way.

Example:

-(void)viewDidLoad

{

    [super viewDidLoad];

    SCWebView\* webView = [[SCWebView alloc] initWithFrame: self.view.bounds];

    webView.autoresizingMask = UIViewAutoresizingFlexibleWidth | UIViewAutoresizingFlexibleHeight;

    NSURL\* url = [NSURL URLWithString: @"<http://mobilesdk.sc-demo.net/>"];

    [webView loadURL: url];

    [self.view addSubview: webView];

}

That's all, it is ready to use. Any site displayed with SCWebView can use WebView Enhances like left-right swipes. For example if you want to open "http://ws-alr.dk.sitecore.net/Nicam.aspx" url on the right swipe and "http://ws-alr.dk.sitecore.net/Products.aspx" - on the left. All what you need to turn on swiping is add to such links to your web page:

<html>

<head>

    <link rel="scm-forward" href="<http://ws-alr.dk.sitecore.net/Nicam.aspx>" />

    <link rel="scm-back"    href="<http://ws-alr.dk.sitecore.net/Products.aspx>" />

Now swiping is ready to use. Also if you need “back” and “forward” browser's navigation controls, use SCWebBrowser instead of SCWebView, it has a similar API's interface, but contains additional navigation controls.

For the comprehensive list of the features available through the embedded WebView, please refer to the  [Enhanced WebView reference](#_Enhanced_WebView_reference) section.

## Getting started with the WebAPI service

For the quick start see the sections below (created an XCode project and installed a Sitecore Mobile SDK are expected):

Brief outline:

1. **How to establish an anonymous session to a website**

To establish the anonymous session to a website just create the SCApiContext instance with a web service host, example:

SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

Now the SCApiContext object ready to use to access required items and fields (Read [API Reference](#_How_to_establish) for details).

1. **How to get a single item**

You can use the -[SCApiContext itemReaderForItemPath:] and -[SCApiContext itemReaderForItemId:] methods to read items with item's path or ID, examples:

//Read an item with path

[context itemReaderForItemPath: @"/sitecore/content/nicam"](^(id result, NSError \*error)

{

SCItem\* item = result;

         NSLog(@"item display name: %@", item.displayName);

    } );

Read [API reference](#_How_to_Access_3) for details.

1. **What are the main things you might want to access the item: display name, fields etc.**

The Web API service provides items with following properties: “DisplayName”, “ID”, “LongID”, “Path” and “Template” and SCItem class has appropriate properties to access them:

-[SCItem displayName] - item's “DisplayName” property

-[SCItem itemId] - item's “ID” property

-[SCItem longID] - item's “LongID” property

-[SCItem path] - item's “Path” property

-[SCItem itemTemplate] - item's “Template” property

Output to console item's display name for example:

NSLog(@"item display name: %@", item.displayName);

To read more about properties of SCItem see [API reference](#_What_are_the_1).

Also SCItem object may content listed or all Sitecore item's fields according to -[SCItemsReaderRequest fieldNames] properties of request, see [“How to Access the Item's Fields”](#_How_to_Access_6) chapter for details.

1. **Getting the item’s children**

To load items with his children, we also can use -[SCApiContext itemsReaderWithRequest:] method.

First we need to create SCItemsReaderRequest object which contains the set of parameters to request items from the backend:

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

//The path of item

request.request = @"/sitecore/content/Nicam";

//Specifies the type of request option, now item path used

request.requestType = SCItemReaderRequestItemPath;

//Specifies the set of the items which will be loaded. Here item and its children will be loaded

request.scope = SCItemReaderSelfScope | SCItemReaderChildrenScope;

//The set of the field’s names which will be read with each item. Here no fields will be read.

request.fieldNames = [NSSet set];

Next load items with builded request object:

[context itemsReaderWithRequest: request](^(id result, NSError \*error)

{

    //result - is NSArray object where fist element is item and left items - its children

    SCItem \*item = [result objectAtIndex:0];

    NSLog(@"item display name: %@", item.displayName);

    for (NSUInteger index = 1; index < [result count]; ++index)

    {

       SCItem \*child = [result objectAtIndex:index];

       NSLog( @"child display name: %@", child.displayName);

    }

});

([Read API reference for details](#_How_to_Access_4))

1. **Using the children of the particular item to populate a tab bar**

To load field’s values of the item’s children you can use SCItemsReaderRequest with SCItemReaderRequestReadFieldsValues flag and SCItemReaderChildrenScope scope.

First of all you need to create an XCode Single View Application project, install a Sitecore MobileSDK framework (see [Installation Section](#_Install_Sitecore_Mobile) for details) and add a **Tab Bar Controller** to the project.

In implementation of ViewController write a code below:

- (void)viewDidLoad

{

    [super viewDidLoad];

NSMutableArray \*listOfViewControllers = [NSMutableArray new];

    SCApiContext \*session = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

    NSSet\* fieldNames = [NSSet setWithObjects: @"Menu title", @"Tab Icon", nil ];

    SCItemsReaderRequest\* request\_ = [SCItemsReaderRequest requestWithItemPath: @"/sitecore/content/Nicam/"

                                                                    fieldsNames: fieldNames];

    request\_.flags = SCItemReaderRequestReadFieldsValues; //to read field values

    request\_.scope = SCItemReaderChildrenScope; //to read children of the item

    [session itemsReaderWithRequest: request\_](^(id result, NSError \*errors)

    {

        for (SCItem\* item in result) {

            NSString\* title = [item fieldValueWithName: @"Menu title"];

            UIImage\* icon = [item fieldValueWithName: @"Tab Icon"];

            UIViewController\* viewController = [UIViewController new];

    viewController.title = title;

    viewController.tabBarItem.image = icon;

            [listOfViewControllers addObject: viewController];

        }

        [self performSegueWithIdentifier: @"showTabBar" sender: self];

        UITabBarController\* tabBar = (UITabBarController\*)self.modalViewController;

        [tabBar setViewControllers:listOfViewControllers animated:YES];

    } );

}

(Read [API reference](#_Using_the_children_2) for details).

For more features, such as establishing authenticated session, executing queries, using caching and many more please refer to the [Objective-C API Reference](#_Objective-C_API_Reference) section.

## Combining the Use of the Embedded Browser and Web API Service

There is an example to describe how you can retrieve a Sitecore data and use it with a creating an XCode Web View controls along with the using of the Embedded Browser. This example creates a Tab Bar menu with titles and icons from a Sitecore items and loads a web page in a web browser for each item.

First of all you need to create an XCode project, install Sitecore Mobile SDK framework and create a tab bar menu according to this [API Reference section](#_Objective-C_API_Reference).

The project is extends with the following code:

ViewController.m

@interface BrowserViewController : UIViewController

@property ( nonatomic, retain ) NSString\* urlString;

@end

@implementation BrowserViewController

@synthesize urlString;

-(SCWebBrowser\*)webView

{

    return (SCWebBrowser\*)self.view;

}

-(void)loadView

{

    self.view = [SCWebBrowser new];

}

//…

- (void)viewDidLoad

{

   //…

        for (SCItem\* item in result) {

            NSString\* title = [item fieldValueWithName: @"Menu title"];

            UIImage\* icon = [item fieldValueWithName: @"Tab Icon"];

            BrowserViewController\* viewController = [BrowserViewController new];

            viewController.title = title;

            viewController.tabBarItem.image = icon;

            viewController.urlString = [@"<http://mobilesdk.sc-demo.net/>" stringByAppendingString: item.path];

            [listOfViewControllers addObject: viewController];

        }

        [self performSegueWithIdentifier: @"showTabBar" sender: self];

        UITabBarController \*tabBar = (UITabBarController\*)self.modalViewController;

        [tabBar setViewControllers:listOfViewControllers animated:YES];

    } );

}

Build and run the application:



# Using Enhanced Web View Reference

## The Left-Right Swipe Link elements

To provide page changing with right (left) swiping you can use **left-right Swipe link** elements. You need to add the following type of link to your page:

For the right swipe:

<link rel="scm-forward" href="<http://mysite.com/some_page.aspx>" />

For the left swipe:

<link rel="scm-back" href="<http://mysite.com/some_page.aspx>" />

Note

By default, after swipe navigation, it's not to possible returns on previous page using opposite swipe gesture. So, if you need such behavior, please add link element (possible with empty href) like this:

<link rel="scm-back" />

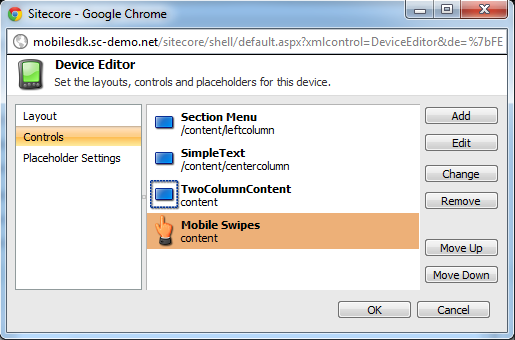
### The Mobile Swipes Component for Sitecore

You can use the Sitecore Mobile Swipes component instead of the javascript function described above.

First of all you need to have installed “Mobile SDK Server Package” component (see [Installation Section](#_SCMobile_Renderings_Installation) for details).

To use the Mobile Swipes component follow the next steps:

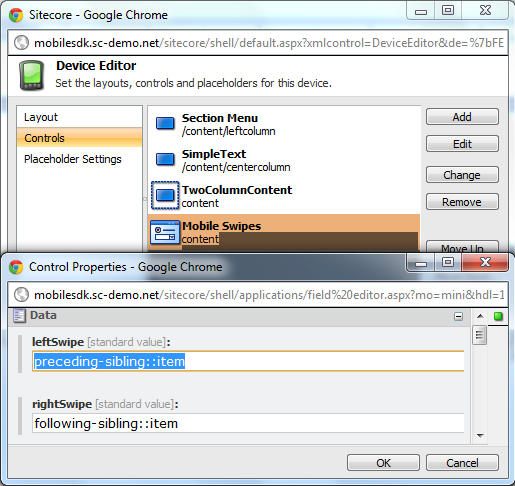
1. Add a “Mobile Swipes” component to some placeholder on a page (anywhere):



Note

You must add a swipe component to some placeholder on a page (not on layout or sublayout). It doesn’t matter where the placeholder is placed on the page. The Mobile Swipes component does not have any visualization and will not be displayed on the page.

1. Set the “leftSwipe” and (or) “rightSwipe” fields of the “Mobile Swipes” component. As a parameters for this fields you can use Sitecore Query related to the current Item.



If you left these parameters empty – it will be no swipes reaction at all.

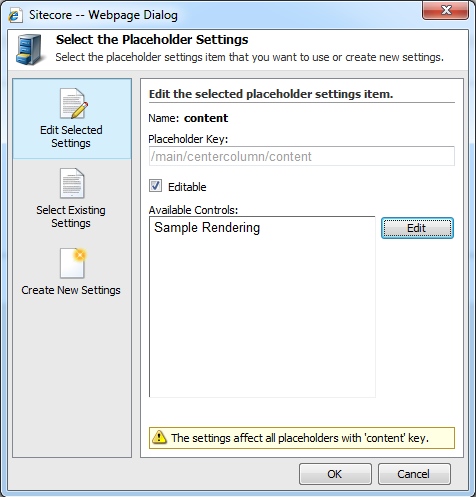
1. Publish your site.

### The Mobile Swipes components for Sitecore in the Page Editor

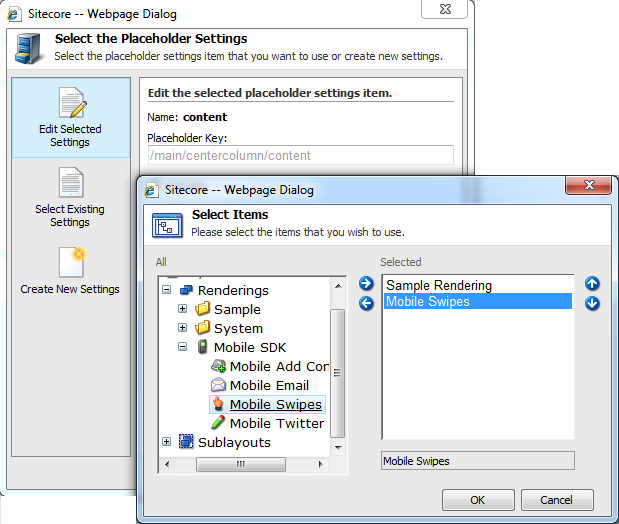
You can add *Mobile Swipe* component easy from Sitecore Page Editor.

Follow the next steps:

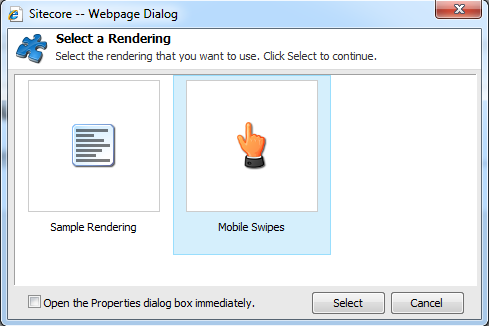
1. Open a Page Editor.
2. Select a placeholder you want to add a component. Open a placeholder settings:



1. Add a Mobile Swipes component to the Available Controls:



1. Select placeholder and click Add Component to the placeholder



1. Publish your site.

## Getting a Snapshot from the Camera

Initiates opening device camera application and gets snapshot from it.

scmobile.camera.getPicture( when\_ok, when\_error );

You can get a picture from this function with the ‘imageURI’ parameter.

Example:

function get\_image()

{

function when\_ok(imageURI)

{

    scmobile.console.log('getPicture OK: ' + imageURI);

    var image = document.getElementById('myImage');

    image.src = imageURI;

   }

  function when\_error() {}

  scmobile.camera.getPicture(when\_ok, when\_error);

}

## Getting the Accelerometer Data

Captures device motion in the x, y, and z directions.

var accelerometer = new scmobile.motion\_manager.Accelerometer();

You can get XYZ motion with accelerData parameter.

To stop accelerometer recognizing use following function:

accelerometer.stop();

Example:

function accelerometer()

{

  var accelerometer = new scmobile.motion\_manager.Accelerometer();

  accelerometer.onAcceleration = function(accelerData)

{

 scmobile.console.log( 'got acceleration: x: ' + accelerData.x

                                               + ' y: ' + accelerData.y

                                               + ' z: ' + accelerData.z

                                               + ' timestamp: ' +

accelerData.timestamp);

  accelerometer.stop();

   }

}

## Getting the Device Info

You can get device info (version, name, uuid) with following properties:

var device\_version = scmobile.device.version;

var device\_name = scmobile.device.name;

var device\_uuid = scmobile.device.uuid;

## Managing the Contacts

You can create, find, edit and remove contacts with the Sitecore Mobile SDK functions.

### Creating a Contact

var contact = scmobile.contacts.create({firstName: ‘FName’, lastName: ‘LName’});

contact.phones = ['333-555'];

contact.silentSave(onSuccess, onError);

A contact with specified fields creates in device contacts.

Note:

There is no required fields in create function. You can specify any fields or set none.

If such contact is already exists the function will create another contact (even if they absolutely the same).

silentSave function gets two callback functions – onSuccess and onError as parameters.

You can specify the following fields:

* firstName
* lastName
* company
* emails (array)
* phones (array)
* websites (array)
* photo (Image)
* birthday (Date)
* addresses (Array)

Example:

var contact = scmobile.contacts.create({firstName: ‘FName’, lastName: ‘LName’, company:

‘Company’});

contact.phones    = ['333-555', ‘+38(050)6663344”];

contact.emails    = [‘[email@mail.ru](mailto:email@mail.ru)’];

contact.websites  = [‘www.sitecore.net’];

contact\_.birthday = new Date(2001, 01, 10);

var address\_ = {};

address\_.street  = 'street';

address\_.city    = 'city';

address\_.state   = 'state';

address\_.zip     = 'zip';

address\_.country = 'country';

contact\_.addresses = [ address\_ ];

var onSuccess = function(contacts)

{

    window.alert("Contact  was successfully created.");

}

var onError = function(error)

{

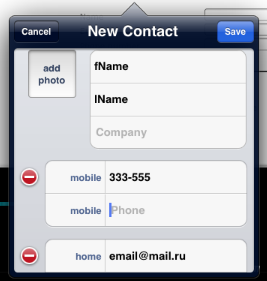
   scmobile.console.log('cannot create a contact');

}

contact.silentSave(onSuccess, onError);

Note:

You can use save function instead of silentSave. In this case you will see a native Contact’s window with filled predefined fields.



You can edit information and then click Save or Cancel.

### Finding a Contact

var predicate = function(contact)

{

return (contact.firstName == ‘FName’ && contact.lastName == ‘LName’)

};

scmobile.contacts.find(predicate, onSuccess, onError);

As a predicate expression you can use any logic expressions with operations ‘&&’ (and) and ‘||’ (or) and any contact’s fields.

The find function returns array of contacts which was found with specified predicate.

Example:

function find\_contact() {

    var onSuccess = function(contacts){

        contacts.forEach(

            function(contact) {

                window.alert("Contact: " + contact.firstName + " " + contact.lastName);

            } );

        if (contacts.length == 0)

        {

            window.alert("Contact doesn't exist.");

        }

    }

    var onError = function(error){

       scmobile.console.log('cannot find a contact’);

    }

    var predicate = function(contact){return (contact.firstName == ‘FName’ || contact.lastName == ‘LName’)};

   scmobile.contacts.find(predicate, onSuccess, onError);

}

### Removing a Contact

contact.remove(onSuccess, onError);

Remove function gets two callback functions – onSuccess and onError as parameters.

Example:

function remove\_contact(contact) {

    var onSuccess = function(){

        window.alert(“Contact was successfully removed”);

    }

    var onError = function(error){

       scmobile.console.log(‘cannot find a contact‘);

    }

    contact.remove(onSuccess, onError);

}

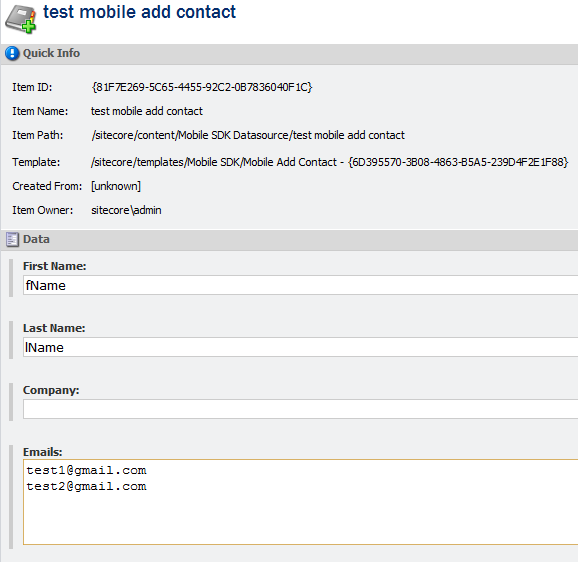
### The Mobile Add Contact Component for Sitecore

You can use the Sitecore Mobile Add Contact component instead of the javascript function described above.

First of all you need to have installed “Mobile SDK Server Package” package (see [Installation Section](#_SCMobile_Renderings_Installation) for details).

To use the Mobile Add Contact component follow the next steps:

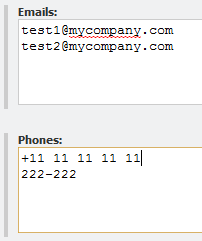
1. Create a source Item with “Mobile Contact” template for path: /sitecore/templates/SCMobile/Mobile Add Contact;



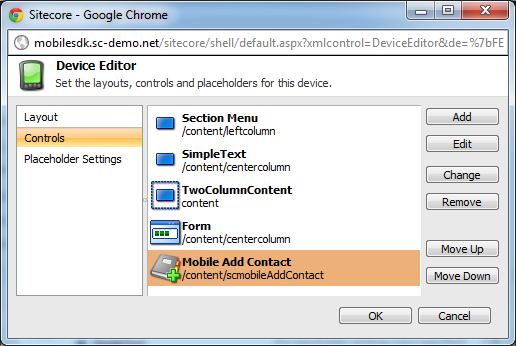
You can set the following fields:

* + - Icon (Image)
    - First Name (Single-Line Text)
    - Last Name (Single-Line Text)
    - Company (Single-Line Text)
    - Emails (Multi-Line Text)
    - Phones (Multi-Line Text)
    - Websites (Multi-Line Text)
    - Photo (Image)
    - Birthday(Date)
    - Addresses(Multi-Line Text)

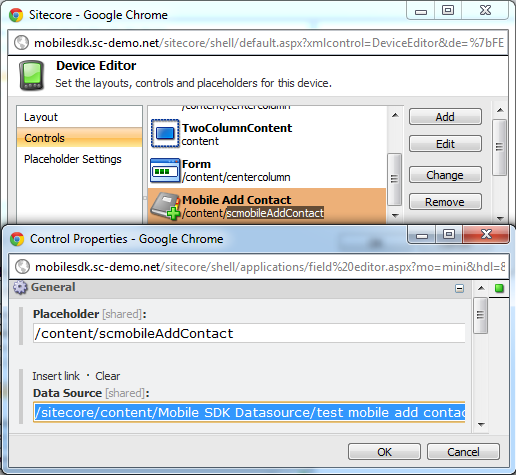
The following image shows the separate Emails and Phones:



1. Add a “Mobile Add Contact” component to the presentation you need:



1. Set the created item as a Data source to the “Mobile Add Contact” component:



1. Publish your site

After running your application on iOS device you will see a native Contact’s window with clicking “Add a contact” button which will be created with “Mobile Add Contact” component.

Note

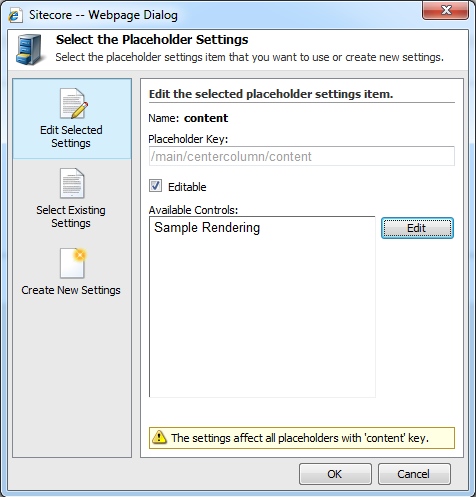
If such contact is already exists the function will create another contact (even if they absolutely the same).

### The Mobile Add Contact Component for Sitecore in the Page Editor

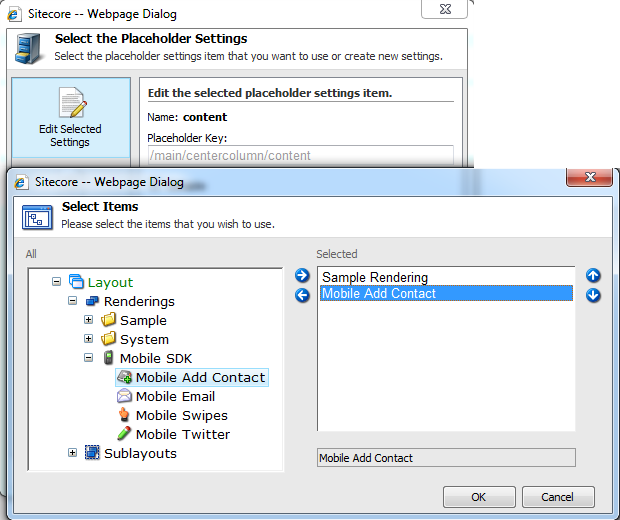
You can add *Mobile Add Contact* component easy from Sitecore Page Editor.

Follow the next steps:

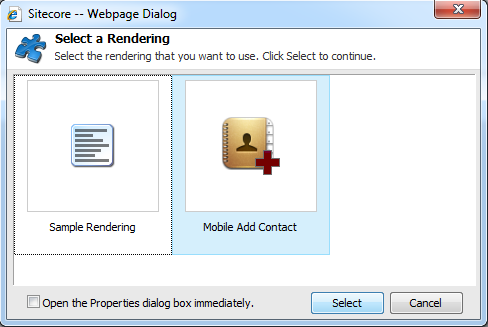
1. Open a Page Editor.
2. Select a placeholder you want to add a component. Open a placeholder settings:



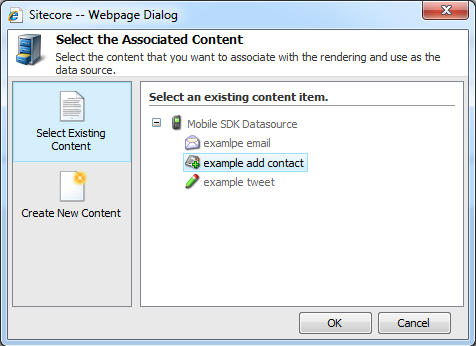
1. Add a *Mobile Add Contact* component to the Available Controls:



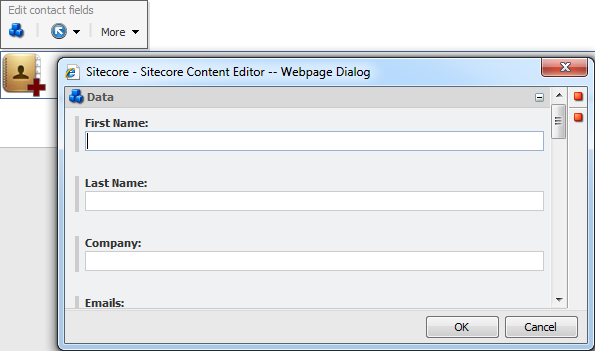
1. Select placeholder and click Add Component to the placeholder



1. You can choose *example add contact* or create a new datasource item as well



1. You can click on component icon and edit datasource item fields simply from a Page Editor:



1. Publish your site.

## Sending a Message to Twitter

You can send a message to Twitter using Mobile SDK functions. You can add an image and (or) a link to the message.

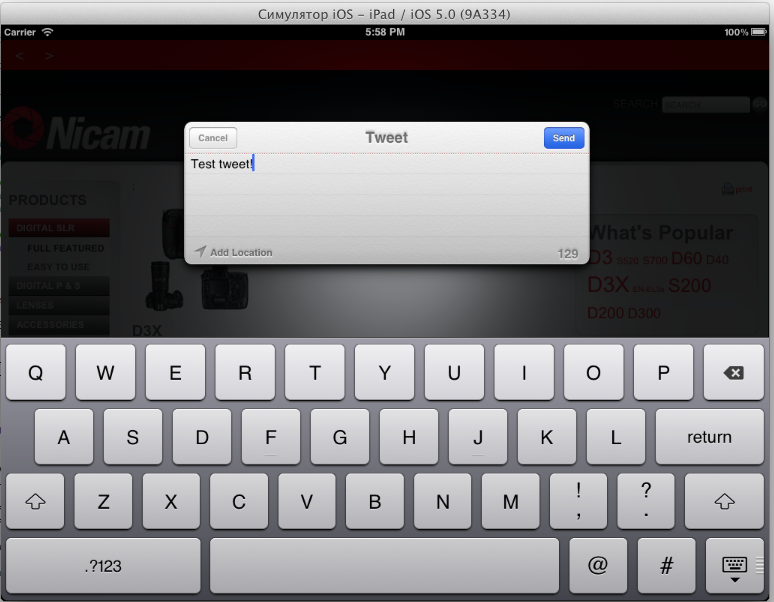
var tweet\_ = new scmobile.share.Tweet();

tweet\_.text = "Test tweet!";

tweet\_.send(onSuccess, onError);

Note

While calling this function you will see a Twitter window with predefined message and links.



You can edit this information, then click Send or Cancel.

Example:

function twitter() {

    var tweet\_ = new scmobile.share.Tweet();

    tweet\_.text = "Test tweet!";

    tweet\_.urls.push(“[www.sitecore.net](http://www.sitecore.net)”);

    tweet\_.imageUrls.push(myImageUrl);

    function onSuccess ()

    {}

    function onError (error)

    {

       scmobile.console.log('errorSend: ' + error);

    }

    tweet\_.send(onSuccess, onError);

}

### The Mobile Tweet This Component for Sitecore

You can use a Sitecore “Mobile Tweet This” component instead of javascript function described above.

First of all you need to have installed “Mobile SDK Server Package” component (see [Installation Section](#_SCMobile_Renderings_Installation) for details).

To use the Mobile Tweet This component follow the next steps:

1. Create a source Item with the Mobile Tweet This template for path: /sitecore/templates/SCMobile/Mobile Tweet This;



You can set the following fields:

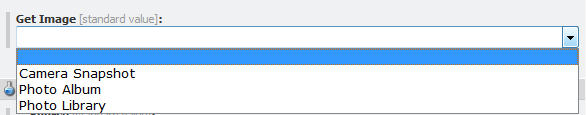
* + - Text (Single-Line Text)
    - Urls (Multi-Line Text)

Urls separation is a next line.

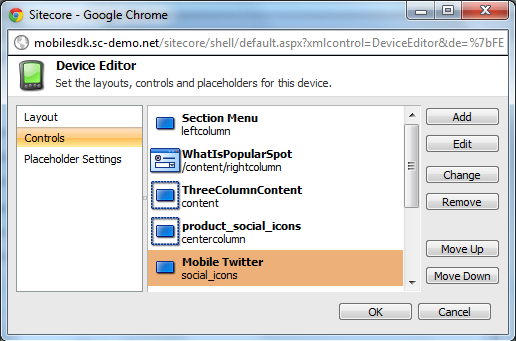
* + - Icon (Image)
    - Get Image (Droplist)

You can choose a following ways to get image from a device:

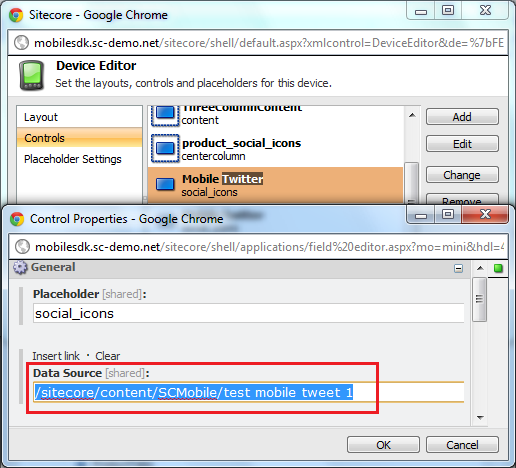
1. none – tweet message without image;
2. Photo Album – opens a camera Photo Album window to choose a photo;
3. Photo Library – opens a camera Photo Library window to choose a photo from some album;



1. Add a “Mobile Tweet This” component to the presentation you need:



1. Set the created item as a Data source to the Mobile Tweet This component:



1. Publish your site

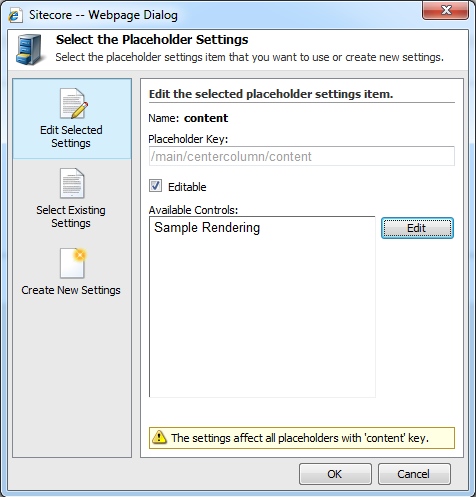
After running your application on iOS device and clicking “Tweet” button you will see a native Twitter window a predefined message, links (and snapshot if “Get Image” field is not empty).

### The Mobile Tweet this Component for Sitecore in the Page Editor

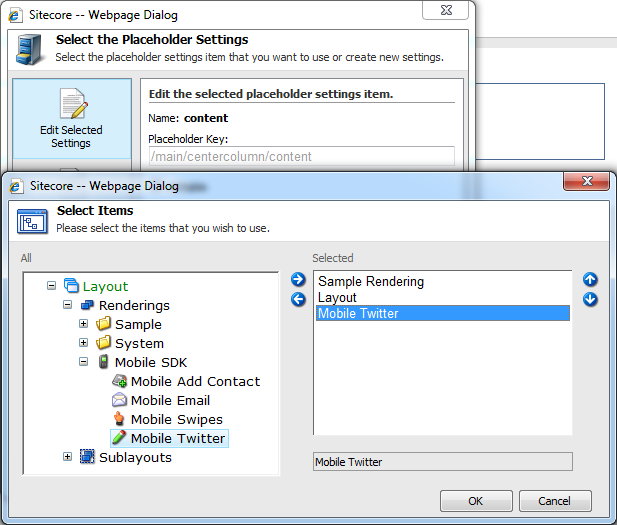
You can add *Mobile Tweet This* component easy from Sitecore Page Editor.

Follow the next steps:

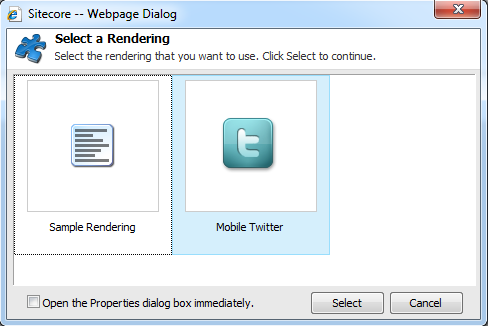
1. Open a Page Editor.
2. Select a placeholder you want to add a component. Open a placeholder settings:



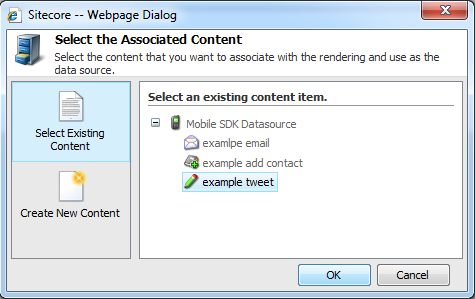
1. Add a *Mobile Tweet This* component to the Available Controls:



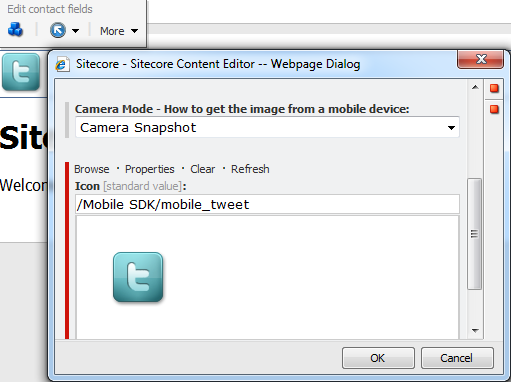
1. Select placeholder and click Add Component to the placeholder



1. You can choose *example tweet* or create a new datasource item as well



1. You can click on component icon and edit datasource item fields simply from a Page Editor:



1. Publish your site.

## Send a Message to Email

You can send a message to email using Mobile SDK functions.

Example:

var email = new scmobile.share.Email();

email.toRecipients = ['[email1@site.com](mailto:email1@site.com)', '[email2@site.com](mailto:email2@site.com)'];

email.ccRecipients = ['[email1@site.com](mailto:email1@site.com)', '[email2@site.com](mailto:email2@site.com)'];

email.bccRecipients = ['[email1@site.com](mailto:email1@site.com)', '[email2@site.com](mailto:email2@site.com)'];

email.subject = 'HI';

email.messageBody = 'Email Body';

function onSuccess(result)

{

  scmobile.console.log('onSuccess: ' + result.result);

}

function onError(error)

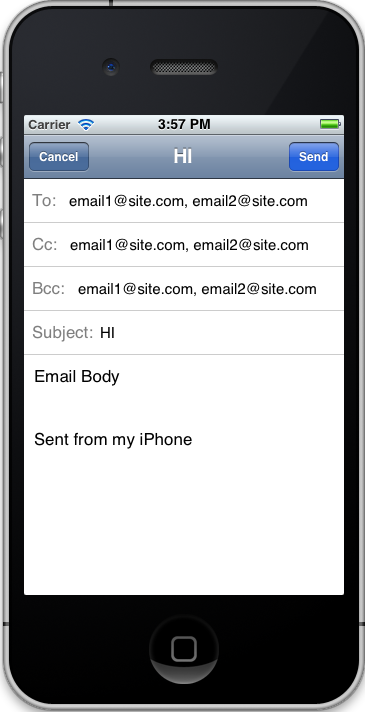
{

  scmobile.console.log('onError: ' + error.error);

}

email.send(onSuccess, onError);

This code should show Email Screen with specified recipients, subject and message body.



You can edit this information then click ‘Send’ or ‘Cancel’.

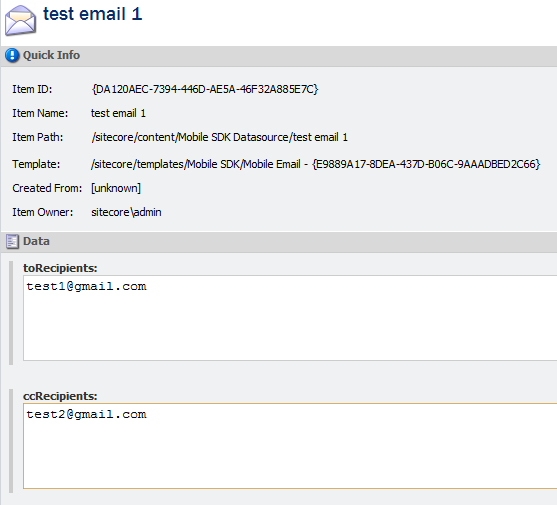
### The Email Component for Sitecore

You can use a Sitecore “Email” component instead of javascript function described above.

First of all you need to have installed “Mobile SDK Server Package” package (see [Installation Section](#_SCMobile_Renderings_Installation) for details).

To use the Email component follow the next steps:

1. Create a source Item with the Mobile Email template for path: /sitecore/templates/SCMobile/Mobile Email;

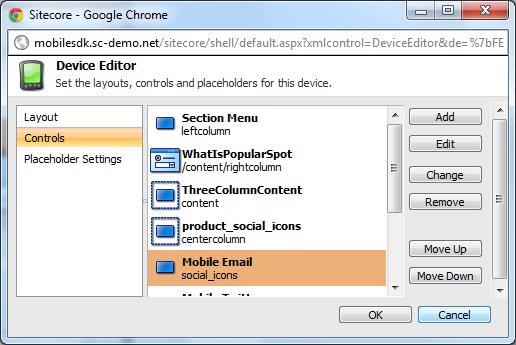


You can set the following fields:

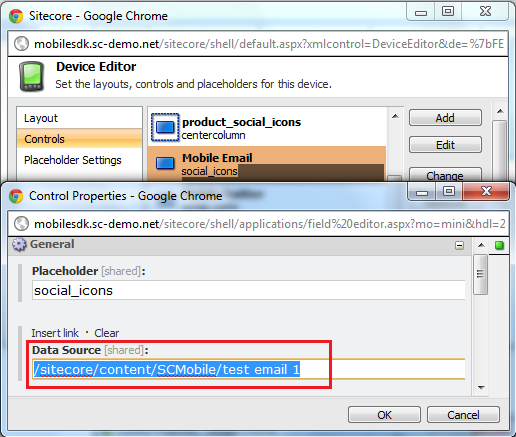
* + - toRecipients (Multi-Line Text)
    - ccRecipients (Multi-Line Text)
    - bccRecipients (Multi-Line Text)
    - Subject (Single-Line Text)
    - Message Body (Rich Text)
    - Icon (Image)

toRecipients, ccRecipients, bccRecipients separation is a next line.

1. Add a Mobile Email component to the presentation you need:



1. Set the created item as a Data source to the “Mobile Email” component:



1. Publish your site

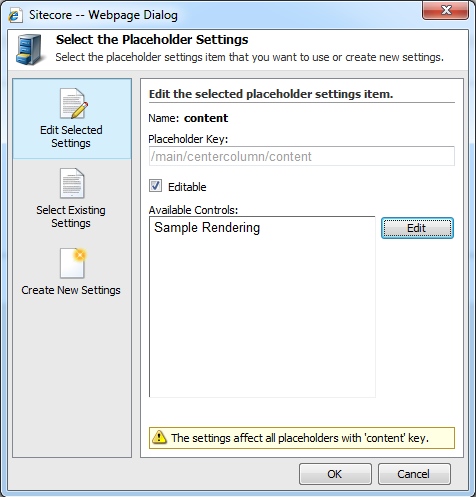
After running your application on iOS device and clicking “Email” button you will see Email window with specified recipients, subject and message body.

### The Mobile Email Component for Sitecore in the Page Editor

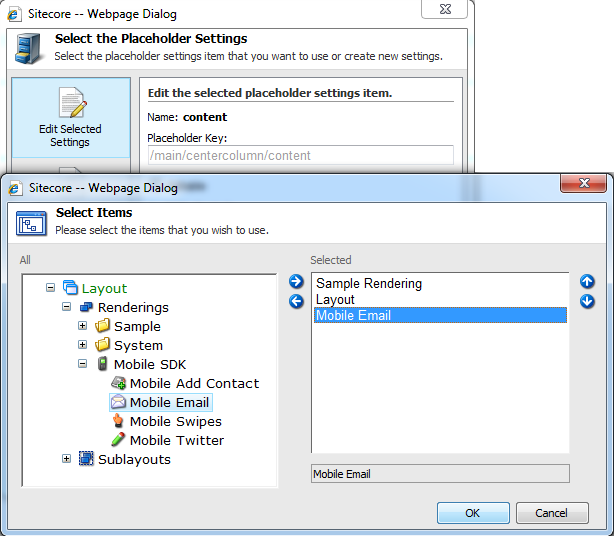
You can add *Mobile Email* component easy from Sitecore Page Editor.

Follow the next steps:

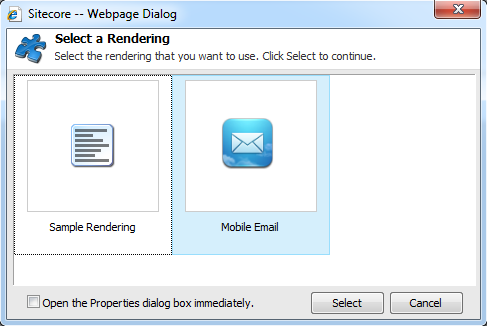
1. Open a Page Editor.
2. Select a placeholder you want to add a component. Open a placeholder settings:



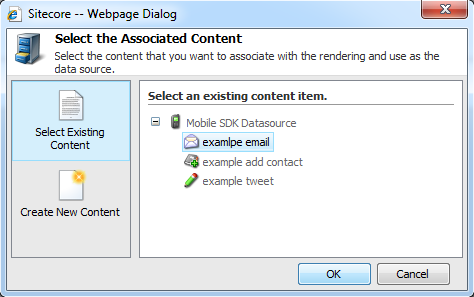
1. Add a *Mobile Email* component to the Available Controls:



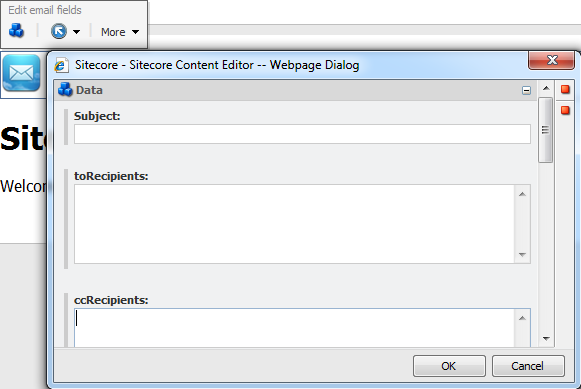
1. Select placeholder and click Add Component to the placeholder



1. You can choose *example tweet* or create a new datasource item as well



1. You can click on component icon and edit datasource item fields simply from a Page Editor:



1. Publish your site.

## Using Javascript Native Alert

Shows a custom alert or dialog box. Provide a possibility to use customize alerts with advanced parameters instead of the browser's alert function, which is typically less customizable.

scmobile.notification.alert(title, message, alertCallback, buttons);

title: Dialog title (String);

message: Dialog message (String);

alertCallback: Callback to invoke when alert dialog is dismissed. (Function);

buttons: Button name(s) (String);

Note

alertCallbackcan handle button pressed result: variable buttonIndexreturns index of pressed button (0… buttons count - 1).

Example:

device2web.notification.alert("Caption" ,

"Alert text",

function(buttonIndex) { //some code goes here; }                             ,

"Ok, Cancel");

# Manipulating an Item Using the Framework’s API

## Content API – first steps

Before going into item operations I think it is worth to spend time on the first steps you have to take before really using the API:

1. First, you need to add Sitecore Mobile SDK framework to your Xcode project (see [Installation Section](#_Install_Sitecore_Mobile) for details).
2. Initiating an anonymous or authenticated session (see [API Reference](#_How_to_establish) for details).
3. See recommendations and best practices, such as how to store the objects that you need to access the API once the session is established, whether the API is thread safe, etc.

### Installing a Sitecore Mobile SDK Framework to the Project

To install a Sitecore MobileSDK framework to an existed project see [Installation Section](#_Install_Sitecore_Mobile).

### Establishing an Anonymous or Authenticated Session to a Website

To establish an anonymous session to a web service, create SCApiContext instance using +[SCApiContext contextWithHost:] method.

Example:

SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/-/webapi)"];

To establish an authenticated session to a web service, create SCApiContext instance using +[SCApiContext contextWithHost:login:password:] method.

Example:

SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/-/webapi)"

                                                     login: @"test"

                                                  password: @"\*\*\*\*"];

Parameter – "mobilesdk.sc-demo.net/-/webapi" is a host of the Sitecore site plus Web API special path component: “-/webapi”. This method returns the existing SCApiContext object for given host or creates new one if it doesn’t exists.

Do not obligatorily to store a new created SCApiContext object in your application manually, all SCItem and SCField objects owns theirs SCApiContext instance and you can access it using [SCItem apiContext], [SCField apiContext] properties or calling [SCApiContext contextWithHost:] method with your web service host.

Note

SCApiContext object is not a “singleton”, if you have no any SCItem, SCField and SCReader object in a memory and any other entity does not retain SCApiContext object – it will be released.

After creating the SCApiContext object you can access required items and fields as anonymous user, using SCApiContext's methods like -[SCApiContext itemsReaderWithRequest:] etc., see [Api Reference](#_Objective-C_API_Reference) Guide for details.

### Recommendations and Best Practices

API Thread safety

The API is not thread safe - perform all calls to the API exactly from one thread. Main thread suggested.

Accessing the Sitecore content

All content API methods, that used to load data from backend, return a block with SCAsyncOp type. You can find it's definitions in SCAsyncOpDefinitions.h file:

typedef void (^SCAsyncOpResult)(id result, NSError \*error);

typedef void (^SCAsyncOp)(SCAsyncOpResult handler);

This block provides content asynchronously – it means that you should not care about threads. To load the Sitecore content - call this block with SCAsyncOpResult callback for getting an expected result.

SCAsyncOp block owns required context to load data from a backend. Imagine you need to read “Checkbox” field's items for item with path: "/sitecore/content/Nicam" hosted on "mobilesdk.sc-demo.net". First create SCAsyncOp block:

SCApiContext\* context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

SCItemsReaderRequest\* request = [SCItemsReaderRequest new];

request.requestType = SCItemReaderRequestItemPath;

 request.request = @"/sitecore/content/Nicam";

 request.flags = SCItemReaderRequestReadFieldsValues;

 request.fieldNames = [NSSet setWithObjects: @"CheckListField", nil];

 SCAsyncOp asyncOp = [context itemsReaderWithRequest: request];

Than just call a SCAsyncOp block to load items:

asyncOp(^(id result, NSError\* error)

{

         SCItem\* item = [result objectAtIndex: 0];

         SCChecklistField\* field = (SCChecklistField\*)[item fieldWithName: @"CheckListField"];

         NSLog(@"items1: %@", field.fieldValue);

 });

Here you should not care about owning of SCApiContext or SCItemsReaderRequest objects, SCAsyncOp block owns them itself. So, if you want to load described items next time, just call the same SCAsyncOp block again.

The SCAsyncOp block never owns of SCItem or SCField objects. Also if you implement own custom SCAsyncOp block try to avoid such behavior for preventing unacceptable accumulation of SCItems objects in the memory. It is assumed that you can keep the SCAsyncOp block all the application lifetime, but items should be released at the memory issues. See details below.

Storing the Loaded Items

SCApiContext object and SCAyncOp block does not owns SCItem instances, they can be owned only by their ancestors (for example SCItem objects owns his [SCItem allChildren] items). So for most cases you should take care about owning loaded items himself.

Based on the above owns rules we can provide such suggestions of using this items:

1. You can store SCApiContext objects and SCAyncOp blocks as long as you need, they does not require a lot of memory and once created can live all time.
2. SCItem instances are more heavy objects and you should be careful to storing it. It is assumed that you should load items lazy, only when they really required to displaying and release them if they are no needs or on IOS memory warnings. See brief example of these suggestions.

Example:

//create loader and show MyViewController

SCAsyncOp asyncOp = [self createSomeAsyncOpBlock];

MyViewController \*controller = [MyViewController new];

controller.asyncOp = asyncOp;

[self presentModalViewController:controller animated:YES];

@interface MyViewController : UIViewController

@property(nonatomic,copy) SCAsyncOp asyncOp;

@end

@interface MyViewController ()

@property(nonatomic,retain) NSArray\* items;

@end

@implementation MyViewController

//...

- (void)updateUIWithItems:(NSArray \*)items

{

    self.items = items;

    //display items here

}

- (void)viewDidLoad

{

    [super viewDidLoad];

    SCAsyncOp asyncOp = self.asyncOp;

    if (!asyncOp)

        return;

    //create weak pointer to avoid retains of MyViewController's instance

    \_\_weak MyViewController\* weakSelf = self;

    asyncOp(^(id result, NSError \*error)

    {

        if (asyncOp == weakSelf.asyncOp && weakSelf.isViewLoaded)

            [weakSelf updateUIWithItems: result];

    } );

}

- (void)viewDidUnload

{

    //release items

    self.items = nil;

    [super viewDidUnload];

}

### Accessing an Item

You can use the -[SCApiContext itemReaderForItemPath:] method to read any item with path and the -[SCApiContext itemReaderForItemId:] method to read any item with ID.

Examples:

//Read item with path

    SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

    [context itemReaderForItemPath: @"/sitecore/content/nicam"](^(id result, NSError \*error)

    {

         SCItem\* item = result;

         NSLog(@"item display name: %@", item.displayName);

    } );

    //Read item with ID

    SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

    [context itemReaderForItemId: @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}" ](^(id result, NSError\* error)

    {

         SCItem \*item = result;

         NSLog( @"item display name: %@", item.displayName);

    });

These methods return the asynchronous blocks — SCAsyncOp Objective-C blocks, which then called with SCAsyncOpResult block handler for getting an expected result.

A more general way to get the same result is using the -[SCApiContext itemsReaderWithRequest:] method

Examples:

//Read item with path

    SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

    SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

    request.scope = SCItemReaderSelfScope;

    request.request = @"/sitecore/content/nicam";

    request.requestType = SCItemReaderRequestItemPath;

    request.fieldNames = [NSSet set];//do not read item's fields

    [context itemsReaderWithRequest: request]( ^(id result, NSError \*error)

    {

        SCItem\* item = [context itemWithPath: @"/sitecore/content/nicam"];

        NSLog( @"item display name: %@", item.displayName );

    } );

    //Read item with ID

    SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

    SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

    request.scope = SCItemReaderSelfScope;

    request.request = @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}";

    request.requestType = SCItemReaderRequestItemId;

    request.fieldNames = [NSSet set];//do not read item's fields

    [context itemsReaderWithRequest: request]( ^(id result, NSError \*error)

    {

        SCItem\* item = [context itemWithId: @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"];

        NSLog(@"item display name: %@", item.displayName);

    });

If the item was already read and still exists in memory, you can access it using such methods:

SCItem \*item = [context itemWithPath: @"/sitecore/content/nicam"];

Or

SCItem \*item = [context itemWithId: @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"];

Note

Sitecore item path evaluating is not a case-sensitive.

### Requirements to Access an Item

SCItem class provides getters for Sitecore item's properties:

-[SCItem displayName] - item's “DisplayName” property

-[SCItem itemId] - item's “ID” property

-[SCItem longID] - item's “LongID” property

-[SCItem path] - item's “Path” property

-[SCItem itemTemplate] - item's “Template” property

Output to console item's display name for example:

NSLog(@"item display name: %@", item.displayName);

To access item's fields -[SCItem fieldsReaderForFieldsNames:] method can be used.

Example below shows how to read Nicam's item “Phone” and “Title” fields:

NSSet \*fieldsNames = [NSSet setWithObjects:@"Phone", @"Title", nil];

[item fieldsReaderForFieldsNames: fieldsNames](^(id result, NSError \*error)

{

     NSDictionary \*fields = result;

     SCField \*phoneField = [fields objectForKey: @"Phone"];

     NSLog(@"Phone field raw value: %@", phoneField.rawValue);

});

If you have no item, it is possible to load the fields with the item, example:

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/nicam";

request.requestType = SCItemReaderRequestItemPath;

request.scope = SCItemReaderSelfScope;

request.fieldNames = [NSSet setWithObjects: @"Phone", @"Title", nil];

[context itemsReaderWithRequest: request](^(id result, NSError \*error)

{

       SCItem\* item = [result objectAtIndex: 0];

       NSLog(@"Phone field raw value: %@", [item fieldValueWithName: @"Phone"]);

});

### Accessing the Children of an Item

You can use the -[SCItem childrenReader] method to read the children of an existing item.

Example:

SCItem \*item = [context itemWithPath: @"/sitecore/content/nicam"];

[item childrenReader](^(id result, NSError \*error)

 {

        NSArray \*children = result;

        NSLog(@"children count: %d", [children count]);

 } );

If you have no item, you can get the children with item ID or item path, example:

//read children of an item with item path: /sitecore/content/nicam

SCApiContext\* context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

[context childrenReaderWithItemPath: @"/sitecore/content/nicam"](^(id result, NSError \*error)

{

        NSArray\* children = result;

        NSLog(@"children count: %d", [children count]);

});

    //read children of an item with item ID

SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

[context childrenReaderWithItemId: @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"](^(id result, NSError \*error)

{

        NSArray\* children = result;

        NSLog(@"children count: %d", [children count]);

} );

A more general way to get the same result is using the -[SCApiContext itemsReaderWithRequest:] method.

Examples:

// read children of an item with path: /sitecore/content/nicam

SCApiContext\* context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/nicam";

request.requestType = SCItemReaderRequestItemPath;

request.scope = SCItemReaderChildrenScope;

request.fieldNames = [NSSet set];

[context itemsReaderWithRequest: request](^(id result, NSError\* error)

{

        NSArray\* children = result;

        NSLog(@"children count: %d", [children count]);

});

// read children of an item with ID

SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}";

request.requestType = SCItemReaderRequestItemId;

request.scope = SCItemReaderChildrenScope;

request.fieldNames = [NSSet set];

[context itemsReaderWithRequest: request](^(id result, NSError \*error)

{

        NSArray\* children = result;

        NSLog(@"children count: %d", [children count]);

});

If the children were already read, you can access it using -[SCItem allChildren] methods:

Example:

NSArray\* children = [item allChildren];

NSLog(@"children count: %d", [children count]);

### Using the children of the particular item to populate a tab bar

There is an example to describe how you can retrieve a Sitecore data and use it with a creating an XCode WebView controls. This example creates a Tab Bar menu with titles and icons from a Sitecore items using Mobile SDK API.

1. Create a simple XCode Single View Application project;
2. Install a Sitecore Mobile SDK framework (see [Installation Section](#_Install_Sitecore_Mobile) for details);
3. Add a **Tab Bar Controller** to the project. Assign the identifier to the Seque from a **View Controller** to the **Tab Bar Controller**:



1. In ViewController.h include a Sitecore Mobile SDK framework:

#import <SitecoreMobileSDK/SitecoreMobileSDK.h>

1. In implementation of ViewController write a code below:

- (void)viewDidLoad

{

    [super viewDidLoad];

NSMutableArray\* listOfViewControllers = [NSMutableArray new];

    SCApiContext\* \_session = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

    NSSet\* field\_names\_ = [NSSet setWithObjects: @"Menu title", @"Tab Icon", nil ];

    SCItemsReaderRequest\* request\_ = [SCItemsReaderRequest requestWithItemPath: @"/sitecore/content/Nicam/"

                                                                    fieldsNames: field\_names\_];

    request\_.flags = SCItemReaderRequestReadFieldsValues; //to read field values

    request\_.scope = SCItemReaderChildrenScope; //to read children of the item

    [\_session itemsReaderWithRequest: request\_](^(id result, NSError\* error\_)

    {

        for (SCItem\* item\_ in result) {

            NSString\* title\_ = [item\_ fieldValueWithName: @"Menu title"];

            UIImage\* icon\_ = [item\_ fieldValueWithName: @"Tab Icon"];

            UIViewController\* viewController\_ = [UIViewController new];

    viewController\_.title = title;

    viewController\_.tabBarItem.image = icon;

            [listOfViewControllers addObject: viewController\_];

        }

        [self performSegueWithIdentifier: @"showTabBar" sender: self];

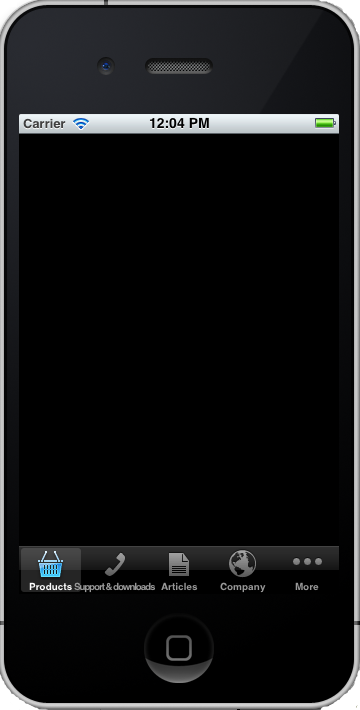
        UITabBarController\* tabBar\_ = (UITabBarController\*)self.modalViewController;

        [tabBar\_ setViewControllers:listOfViewControllers animated:YES];

    } );

}

1. Build and run the application:



## Accessing the Fields of an Item

If you have SCItems object and want to load its fields (“Phone” and “Title” for example) – -[SCItem fieldsReaderForFieldsNames:] method can be used for this purposes.

Example:

NSSet \*fieldsNames = [NSSet setWithObjects: @"Phone", @"Title", nil];

[item fieldsReaderForFieldsNames: fieldsNames](^(id result, NSError \*error)

{

        NSDictionary \*fields = result;

        SCField \*phoneField\_ = [fields objectForKey: @"Phone"];

        NSLog(@"Phone field raw value: %@", phoneField\_.rawValue);

});

If you have no item, it is possible to load the fields with the item, example:

// read Nicam item and its “Phone” and “Title” fields.

SCApiContext \*context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

 request.request = @"/sitecore/content/nicam";

 request.requestType = SCItemReaderRequestItemPath;

 request.scope = SCItemReaderSelfScope;

 request.fieldNames = [NSSet setWithObjects: @"Phone", @"Title", nil];

 [context itemsReaderWithRequest: request](^(id result, NSError \*error)

 {

        SCItem\* item = [result objectAtIndex: 0];

        NSLog(@"Phone field raw value: %@", [item fieldValueWithName: @"Phone"]);

 });

If you need to read all the item's fields – pass nil instead of fieldsNames set.

To access already loaded item's fields, call -[SCItem readFieldsByName] method. It returns NSDictionary objects with SCField objects and [SCField name] keys or nil if no fields was loaded for given item.

SCField class represents a Sitecore system item’s field. It provides getters for the field’s properties like “ID”, “Name” ,”Type” and “RawValue”.

-[SCField rawValue] property contains a simple text value.

Checkbox field stores the one character (“1”) when selected.

Rich Text Editor (RTE) fields contain XHTML.

Numerous field types contain the ID of a single item or multiple IDs separated by pipe characters (“|”).

Other field types contain XML or data in proprietary formats.

Some specific fields like “Image” and “Checklist” have separate classes SCImageField and SCChecklistField.

Please see Sitecore Mobile SDK doc HTML version for more details.

## Accessing the Different types of a Field

The Mobile content API provides a set of special classes to simplify work with "Simple" and "List" types of the Sitecore fields.

Special field’s types and corresponding content API classes:

* The "Checkbox" field will be represented with SCChecklistField class instance;
* "Color Picker" field with SCColorPickerField;
* "Date" field with SCDateField;
* "Datetime" field with SCDateTimeField;
* "Image" field with SCImageField;
* "Checklist" field with SCChecklistField;
* "Multilist" field with SCMultilistField;
* "Treelist" field with SCTreelistField.

All Sitecore field’s types not presented in list above will be represented by SCField class by default and its "fieldValue" property will contain the raw value of the field.

### The Image Field

If the type of the loaded Field is “Image”, the class of the field is SCImageField. The instance of this class has an additional property – [SCImageField imagePath] – the path to the Sitecore media item's image. It can be used to read an image using the [SCApiContext imageLoaderForSCImagePath:] or [SCImageField fieldValueReader] methods.

So if you need to load UIImage object for your SCImageField you can call [SCImageField fieldValueReader], example:

SCImageField\* field = [item fieldWithName: @"Tab Icon"];

[field fieldValueReader](^(id result, NSError \*error)

{

UIImage\* image = result;

NSLog( @"image size: %@", NSStringFromCGSize( image.size ) );

} );

Also may be useful to read images with fields together:

[item fieldValueReaderForFieldName: @"Tab Icon"](^(id result, NSError \*error)

{

UIImage\* image = result;

NSLog(@"image size: %@", NSStringFromCGSize(image.size));

});

Or to read the fields with the values at the item's loading:

SCApiContext\* context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest\* request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/nicam";

request.requestType = SCItemReaderRequestItemPath;

request.scope = SCItemReaderSelfScope;

request.fieldNames = [NSSet setWithObject: @"Tab Icon"];

request.flags = SCItemReaderRequestReadFieldsValues; //special flags which says to load field's values

[context itemsReaderWithRequest: request ](^(id result, NSError\* error)

{

SCItem\* item = [result objectAtIndex: 0];

UIImage\* image = [item fieldValueWithName: @"Tab Icon"];

NSLog( @"image size: %@", NSStringFromCGSize(image.size));

});

### The Checkbox Field

SCCheckboxField class represents the Sitecore "Checkbox" field.

-[ SCCheckboxField fieldValue] property of this class is NSNumber object.

[[SCCheckboxField fieldValue] boolValue] is NO if checkbox does not selected and YES otherwise.

Example:

    SCField\* field = [item fieldWithName: @"CheckBoxField"];

    NSLog( @"checkbox value: %@", [[field fieldValue] boolValue] );

### The Date and Datetime Fields

SCDateField and SCDateTimeField classes represent the Sitecore "Date" and "Datetime" fields accordingly.

"fieldValue" property of this classes is NSDate object which holds corresponding date value.

Example:

    SCField\* field = [item fieldWithName: @"DateField"];

    [field fieldValueReader](^(id result, NSError \*error)

    {

        NSDate\* date = result;

        NSLog(@"date value: %@", date );

    } );

### The Color Picker Field

SCColorPickerField class represents the Sitecore "Color Picker" field.

"fieldValue" property of this class is UIColor object which holds corresponding color value.

Example:

    SCField\* field = [item fieldWithName: @"ColorPickerField"];

    [field fieldValueReader](^(id result, NSError \*error)

    {

        UIColor\* color = result;

        NSLog(@"color value: %@", color);

    } );

### The Checklist, Multilist and Treelist Fields

SCChecklistField, SCMultilistField and SCTreelistField classes represent the "Checklist", "Multilist" and "Treelist" Sitecore fields accordingly.

"fieldValue" property of this classes is NSArray object which holds the list of the items specified with this fields.

Also in the "rawValue" property you can find raw value of field: the list item's IDs separated by "|" symbol.

"fieldValue" property is nil by default, so use -[SCField fieldValueReader] method to load NSArray of items. SCField object does not own loaded items to avoid retain cycles.

Example:

    [item fieldValueReaderForFieldName: @"Products List"](^(id result, NSError \*error)

    {

        NSArray \*items = result;

        NSLog(@"items count: %d", [items count]);

    });

Some other convenient cases of using -[SCField fieldValueReader] method you can find in [Image Field](#_Image_Field) section.

## Accessing the Parent of an Item

To read the parent of an existed item you can use -[SCItem itemsReaderWithRequest:] method with SCItemReaderParentScope scope:

// read the parent of an item with item Path

SCApiContext\* context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/nicam";

request.requestType = SCItemReaderRequestItemPath;

request.scope = SCItemReaderParentScope;

request.fieldNames = [NSSet set];

[context itemsReaderWithRequest: request](^(id result, NSError \*error)

{

SCItem\* parent = [result objectAtIndex: 0];

NSLog(@"parent display name: %@", parent.displayName);

} );

// read the parent of an item with item ID

SCApiContext\* context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}";

request.requestType = SCItemReaderRequestItemId;

request.scope = SCItemReaderParentScope;

request.fieldNames = [NSSet set];

[context itemsReaderWithRequest: request](^(id result, NSError \*error)

{

SCItem\* parent = [result objectAtIndex: 0];

NSLog(@"parent display name: %@", parent.displayName);

} );

If the parent item was already read and still exists in the memory, you can access it using SCItem property -[SCItem parent].

## Accessing Items Using the Sitecore Query

You can use the -[SCItem itemsReaderWithRequest:] method to retrieve items from a site that match a Sitecore query. Set the [SCItemsReaderRequest requestType] to the SCItemReaderRequestQuery.

It is important to remember that:

* Sitecore query is not always the most efficient way to locate items in repository with a large volume of a data. Consider using a paging where the system must frequently match items in a large branch.
* Sitecore query syntax is not the same as XPath syntax.
* Do not assume that Sitecore query returns items in document order or reverse document order.

Example:

SCApiContext \*context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/Nicam/Products/descendant::\*[@@templatename='Product Group']";

request.requestType = SCItemReaderRequestQuery;

request.fieldNames = [NSSet set]; //do not read

[context itemsReaderWithRequest: request ](^(id result, NSError\* error)

{

NSLog( @"result items count: %d", [result count]);

} );

The same example but with paging, this example reads only third and fourth items of expected result:

SCApiContext \*context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest\* request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/Nicam/Products/descendant::\*[@@templatename='Product Group']";

request.requestType = SCItemReaderRequestQuery;

request.fieldNames = [NSSet set];

request.page = 1; //the number of the page

request.pageSize = 2; //the page size

[context itemsReaderWithRequest: request ](^(id result, NSError\* error)

{

NSLog(@"result items count: %d", [result count]); // expected size is 2

});

## Reading Paged Items Efficiently

If it is expected a lot of items for the given request and the application try to save the memory and the time to read it, it is easy to use SCPagedItems object for paged items loading. The instance of SCPagedItems class allows loading any item at its index and maximum one page of the items will be loaded. Imagine potentially we have a numerous items for given query: /sitecore/content/Nicam/child::\*[@@templatename='Site Section']. Our task is to read only third and fourth items of an expected result for the given query.

Example:

SCApiContext \*context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest\* request = [SCItemsReaderRequest new];

request.requestType = SCItemReaderRequestQuery;

request.request = @"/sitecore/content/Nicam/child::\*[@@templatename='Site Section']";

request.flags = SCItemReaderRequestReadFieldsValues;

request.fieldNames = [ NSSet setWithObjects: @"Title", @"Tab Icon", nil ];

request.pageSize = 2;

SCPagedItems\* pagedItems = [SCPagedItems pagedItemsWithApiContext: context request: request];

[pagedItems itemReaderForIndex: 2 ](^(id result, NSError\* error)

{

SCItem\* item = result;

NSLog(@"item 3 display name: %@", item.displayName);

} );

[pagedItems itemReaderForIndex: 3](^(id result, NSError \*error)

{

SCItem\* item = result;

NSLog(@"item 4 display name: %@", item.displayName);

});

The main profit of using this class - you should not care about paging index calculation and cache logic for the merging requests when load items on the same page. Given example will load the page which contains third and fourth items only once. But if you do not need such behavior and want to read some page of items - just specify [SCItemsReaderRequest pageSize] and [SCItemsReaderRequest page] properties himself and call [SCApiContext itemsReaderWithRequest:] method with your SCItemsReaderRequest object.

Example:

SCApiContext\* context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest\* request = [ SCItemsReaderRequest new ];

request.requestType = SCItemReaderRequestQuery;

request.request = @"/sitecore/content/Nicam/child::\*[@@templatename='Site Section']";

request.flags = SCItemReaderRequestReadFieldsValues;

request.fieldNames = [ NSSet setWithObjects: @"Title", @"Tab Icon", nil ];

request.pageSize = 2;

request.page = 1;//start indexing from zero

[ context itemsReaderWithRequest: request ]( ^(id result, NSError\* error)

{

NSArray\* items = result;

NSLog( @"item count: %d", [items count] );

} );

## Accessing the Different Languages of an Item

You can use the -[SCApiContext defaultLanguage] property to specify a language to read the items with the -[SCApiContext itemWithId:], -[SCApiContext itemReaderForItemId:] methods etc.. The default value of this property is "en".

Example:

SCApiContext \*context = [SCApiContext contextWithHost:@"mobilesdk.sc-demo.net/-/webapi"];

context.defaultLanguage = @"da";

[context itemReaderForItemPath: @"/sitecore/content/nicam"](^(SCItem \*item, NSError \*error)

{

NSLog(@"item: %@", item.displayName);

});

Also you can specify language exactly for the separate request and do not change the SCApiContext default language. Use language property of the request.

Example:

SCApiContext \*context = [SCApiContext contextWithHost:@"mobilesdk.sc-demo.net/-/webapi"];

SCItemsReaderRequest \*request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/nicam";

request.requestType = SCItemReaderRequestItemPath;

request.language = @"da";

[context itemsReaderWithRequest:request](^(NSArray \*items, NSError \*error)

{

SCItem \*item = [items objectAtIndex:0];

NSLog(@"item: %@", item.displayName);

});

To access list of available languages call -[SCApiContext systemLanguagesReader].

Example:

SCApiContext \*context = [SCApiContext contextWithHost:@"mobilesdk.sc-demo.net/-/webapi"];

[context systemLanguagesReader](^(NSSet \*languages, NSError \*error)

{

NSLog(@"languages: %@", languages);

});

## Using the Cache

Sitecore Mobile SDK Content API tries to reduce a physical requests count to Sitecore web service using following cache solutions:

### Merging Requests

If you perform several equal requests to API to load data (at the same moment or while first request still executing), just one physical request to Sitecore web service will be performed but all SCAsynOp block's handlers will be processed.

Example:

Call some asyncOp block twice:

asyncOp(^(id result, NSError\* error)

{

    //result handler 1

});

asyncOp(^(id result, NSError\* error)

{

    //result handler 2

});

Only one physical request will be done here, so don't create owns request's merge logic above the API. “Merge requests” behavior is very useful when the different parts of UI try to load the same data at one moment. For example several UITable cells access the equal images or etc. All asynchronous operations of the API use this solution.

### Reading Cached Items

SCAyncOp block can return already loaded items not disturbing the Sitecore web service if the requested items are still in the memory. But you cannot rely on this behavior, for example Sitecore Query requests always loads items from a backend.

Note

Some item may be still live only for two reasons - your applications owns it or this item is a descendant for another item. In any other case, the SCItem object will be released and it will not be available for the cache.

Example how cache is working:

SCApiContext\* context = [SCApiContext contextWithHost: @"[mobilesdk.sc-demo.net/-/webapi](http://mobilesdk.sc-demo.net/~webapi)"];

SCItemsReaderRequest\* request = [SCItemsReaderRequest new];

request.request = @"/sitecore/content/Nicam";

request.requestType = SCItemReaderRequestItemPath;

request.fieldNames = [NSSet set];

[context itemsReaderWithRequest: request](^(id result, NSError\* error)

{

    SCItem\* item = [result objectAtIndex:0];

    SCAsyncOp childrenReader = [item childrenReader];

    //load children here

    childrenReader(^(id result, NSError \*error)

    {

        NSUInteger previousCount = [result count];

        NSLog( @"children Items count: %d", previousCount);

        //the flag to check that the block is already finish loading

        \_\_block BOOL wasLoadImmediately = NO;

        //load children again

        childrenReader(^(id result, NSError \*error)

        {

            NSLog(@"children Items count 2: %d", [result count]);

            wasLoadImmediately = (previousCount == [result count]);

        });

        //Check that the childrenReader block's handler was already called with a result

        NSAssert(wasLoadImmediately, @"wasLoadImmediately should be YES here");

        });

} );

In the last example NSAssert should not fail which means that the second call of childrenReader returns the result item's children immediately.

Note, cache is unavailable:

1. When a Sitecore query using. When -[SCItemsReaderRequest requestType] is equal to SCItemReaderRequestQuery, the loaded items is ignored and the API loads the data from a Sitecore web service directly.
2. When item was released. When items was released and removed from memory, cache also cannot be used. Now the API does not cache items into a file system.
3. When flag “ignore cache” is used. If you set [SCItemsReaderRequest flag] to SCItemReaderRequestIgnoreCache, the loaded items will be ignored also.

In all other cases API tries to reuse loaded item at requests.

## Modifying the Design of the WebView Navigation Bar

The SitecoreMobileSDK has two WebView components: SCWebView and SCWebBrowser. Both of them has interfaces similar to the standard UIWebView component but the difference between them is a navigation bar of the SCWebBrowser with back and forward buttons and activity indicator:

C:\Users\alr.DK\Desktop\Screen Shot 2012-01-24 at 14.51.50.png

If your application already provides its own controls for back and forward actions, as well as activity indication, and you want 100% control over UI – use the SCWebView control.

If you want to use the actions toolbar provided by the Mobile SDK, use the SCWebBrowser control. In this case you can customize the appearance of the toolbar with the method described below.

Presumably default navigation bar may not approach your application design requirements, and you may want to create your own navigation view. To do it you can use -[SCWebBrowser setCustomToollbarView:] method.

When your custom navigation view is conform a SCWebBrowserToolbar interface, you may use it to navigate WebView component onto back and next pages, draw activity indicator, provide browser navigation controls etc.

Example:

@interface MyNavigationHeader : UIView <SCWebBrowserToolbar>

@property(nonatomic,strong) UIButton \*backButton;

@property(nonatomic,strong) UIButton \*forwardButton;

@property(nonatomic,strong) UIActivityIndicatorView \*activityIndicator;

@end

@implementation MyNavigationHeader

- (id)initWithFrame:(CGRect)frame

{

self = [super initWithFrame:frame];

if(self)

{

// initialize your custom navigation view here

...

}

return self;

}

- (void)didStartLoadingWebBrowser:(SCWebBrowser\*)webBrowser

{

[self.activityIndicator startAnimating];

}

- (void)didStopLoadingWebBrowser:(SCWebBrowser\*)webBrowser

{

[self.activityIndicator stopAnimating];

}

#pragma mark Forward Actions

- (void)goBack:(id)sender

{

[self.delegate goBackWebBrowserNavigator:self];

}

- (void)goForward:(id)sender

{

[self.delegate goForwardWebBrowserNavigator:self];

}

@end

Now create an instance of your custom navigation view and set it to the WebView:

MyNavigationHeader \*view = [[MyNavigationHeader alloc]initWithFrame:CGRectMake(0.f, 0.f, 110.f, 80.f)];

view.autoresizingMask = UIViewAutoresizingFlexibleHeight;

[self.webBrowser setCustomToollbarView: view];