Sitecore CMS 6.5-6.6

Sitecore Mobile SDK Developer's Guide

A developer's guide to configuring and working with the Sitecore Mobile SDK

Table of Contents

Chapter 1

Introduction

Sitecore Mobile SDK is a framework that is designed to help developers create iOS based applications that use and serve the content in Sitecore.

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

This document contains the following chapters:

- **Chapter 1 — Introduction**
  This chapter contains an introduction for this guide.
- **Chapter 2 —The Mobile SDK Installation**
  This chapter describes how to install the Mobile SDK on the client and the server sides.
- **Chapter 3 — Getting Started**
  This chapter describes requirements and procedures to start using the Mobile SDK.
- **Chapter 4 — Using the Enhanced Web View Reference**
  This chapter describes how to use the Enhanced Web View Reference components.
- **Chapter 5 — Using the API to Manipulate an Item**
  This chapter describes how to use the API to manipulate an item.

1.1 The Mobile SDK Components

This section contains brief descriptions for the components of the Sitecore Mobile SDK.

1.1.1 The Server

The Sitecore Mobile SDK server package contains components that allow you to use the iOS features in your website.

The Sitecore Mobile SDK server package requires:

- .Net Framework version 2.0 or later for the Application Pool of your website.
- Sitecore CMS version 6.5 or later.

The Mobile SDK API is called the Sitecore Item Web API and is web service based. Developers use the Sitecore Item Web API to manipulate Sitecore content items through HTTP requests. The API gives access to the content through items paths, IDs, and Sitecore queries. This service produces output in JSON format and is both highly customizable and optimized to support a minimum number of requests.

1.1.2 The Client

The client is an iOS project that contains the `SitecoreMobileSDK.framework` bundle that provides an Objective-C API. You can use this API to access Sitecore content through the web service and use this content in your application. The API also supports advanced features such as *caching*.

You can use the Mobile SDK to display the existing HTML presentation while having access to the native iOS hardware and software abilities. You can reuse the elements of the website and reduce the implementation cost. You can also use native Objective-C.

The list of native features available in HTML environment includes:

- Left-Right swipe navigation
- Access to the camera and the photo library
- Accelerometer
- Device information
- Access to the built-in address book and easy creation of new contacts
- Creating tweets through the system-wide twitter accounts in iOS5
- Sending e-mails using the e-mail account in iOS5

Sitecore Mobile SDK does not support the following iOS features:

- Accessibility
- PassBook — introduced in iOS 6.0
- Facebook integration — introduced in iOS 6.0
- iCloud — introduced in iOS iOS 4.3.3
- AirPlay — introduced in iOS 5.0
- AirPrint — introduced in iOS 5.0

- QuickLook — introduced in iOS 5.0
- Push Notifications — introduced in iOS3.0

Note
If you are using the Sitecore Mobile SDK or want to develop an iOS application on your own, you must enroll in Apple's iOS development program https://developer.apple.com/programs/ios/. To run Xcode or any other developer tools produced by Apple, you also need a workstation with the latest Mac OS. Sitecore Mobile SDK supports iOS 5.0 or later and Xcode 4.4.0 or later.

Chapter 2

The Mobile SDK Installation

This chapter describes how to install the Mobile SDK components.

This chapter contains the following sections:

- Installing the Server Side Components
- Installing the Client Side Components

2.1 Installing the Server Side Components

You must install the Sitecore Item Web API service and the Mobile SDK components before you can access Sitecore content.

2.1.1 Installing the Sitecore Item Web API Service

Use the Sitecore Installation Wizard to install the Sitecore Item Web API service:

1. Log in to the **Sitecore Desktop**.
2. Click **Sitecore**, **Development Tools**, and then **Installation Wizard**.

   The wizard will guide you through the installation process.
3. In the **Select Package** dialog box, specify the package that you want to install.
4. Click **Browse** to locate the package or click **Upload** to upload the package.

   The folder, to which the packages are uploaded, is specified in the `web.config` file.
5. In the **License Agreement** dialog box, accept the license agreement.
6. In the **Ready to Install** dialog box, you can review the package information and then click **Install**.
7. When the installation is complete, you can choose to restart the Sitecore client or the Sitecore server and then click **Finish**.

   To test that the Sitecore Item Web API Service is working, make a simple request to the service.
8. In the browser, enter the URL: `http://yoursite/-/item/v1/sitecore/Content/Home`

To retrieve content from the website, add the following element to `Sitecore.ItemWebApi.config`:

```
<sites>
<!--Other site elements could be here.-->
<site name="website">
   <patch:attribute name="itemwebapi.mode"> AdvancedSecurity</patch:attribute>
   <patch:attribute name="itemwebapi.access">ReadWrite</patch:attribute>
   <patch:attribute name="itemwebapi.allowanonymousaccess">true</patch:attribute>
 </site>
</sites>
<?xml version="1.0" encoding="utf-8"?>
```

2.1.2 Installing the Sitecore Mobile SDK Components

Use the Sitecore Installation Wizard to install the Sitecore Mobile SDK components on the server side:

1. Log in to the **Sitecore Desktop**.
2. Click **Sitecore**, **Development Tools**, and then **Installation Wizard**.

   The wizard will guide you through the installation process.
3. In the **Select Package** dialog box, specify the package that you want to install.
4. Click **Browse** to browse for an existing package or **Upload** to upload a new one.
5. In the **Ready to Install** dialog box, you can review the package information and then click **Install**.
6. When the installation is complete, you can choose to restart the Sitecore client or the Sitecore server and then click **Finish**.

7. In the `<xslExtensions>` section of the `web.config` file add the following:

```
<extension mode="on" type="Sitecore.XslHelpers.MobileExtensions, Sitecore.Mobile"
 namespace="http://www.sitecore.net/scmobile" />
```

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

Note
The installation of Sitecore Mobile SDK Components package is optional. You can implement your own custom mobile renderings that use the native features of the device.

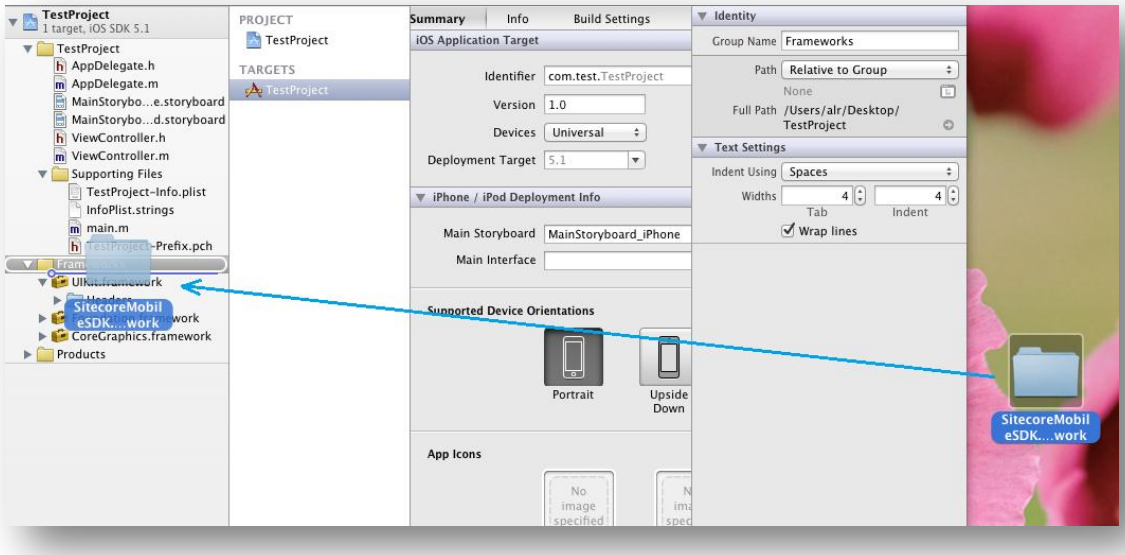
2.2 Installing the Client Side Components

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

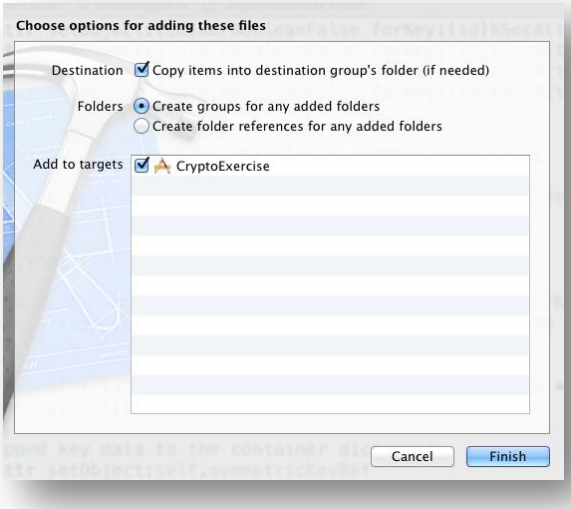
2.2.1 Installing the Sitecore Mobile SDK in a Project

To use the Sitecore Mobile SDK framework in your application:

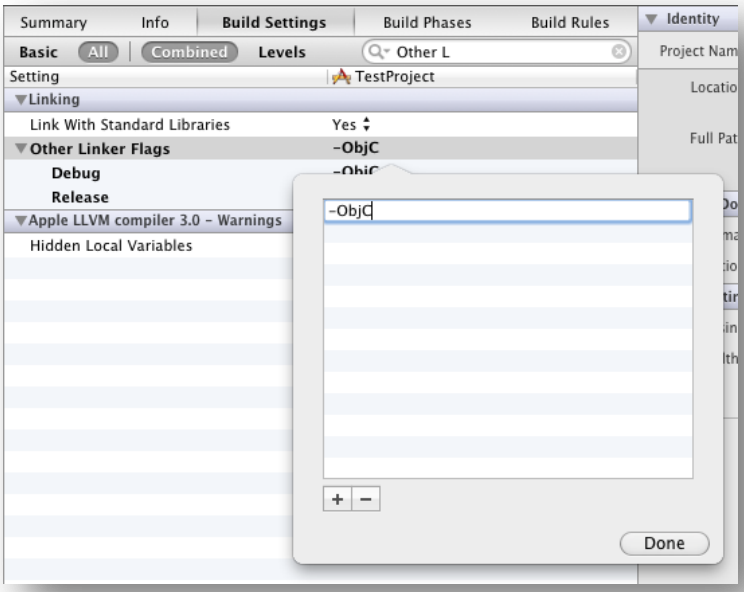
1. Create a simple *Xcode Single View Application* project. For more information, see the *Getting Started* section in the *Apple Developer Manual*. If you have already created an Xcode Single View Application project, you can skip this step.
2. Drag the `SitecoreMobileSDK.framework` bundle and drop it into the project's Frameworks source group.



3. Check **Copy items into destination group's folder** unless you intentionally want to add a framework reference.



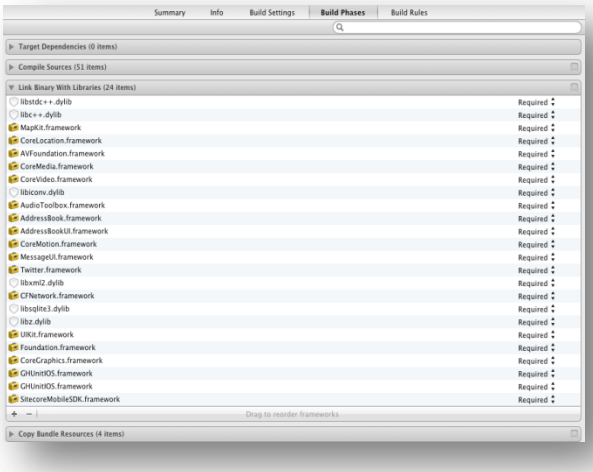
4. Add the `-ObjC` flag to the `Other linker` flag in the Xcode Build Settings. For more information, see the section *Build Setting Reference* in the *Apple Developer Manual*.



Note
If you are still getting linker errors after adding the `-ObjC` flag, try to replace it with `-all_load`. However, this is not a common case.

5. Link the following frameworks to the project:
   - CFNetwork.framework
   - CoreMotion.framework
   - CoreLocation.framework
   - CoreMedia.framework
   - CoreVideo.framework
   - AddressBook.framework
   - AudioToolbox.framework
   - AddressBookUI.framework
   - Twitter.framework
   - MessageUI.framework
   - MapKit.framework
   - AVFoundation.framework

   For more information, see the section *Linking to Library or Framework* in the *Project Editor Help* in the *Apple Developer Manual.*



6. Link the following libraries to the project:
   - libxml2.dylib
   - libz.dylib
   - libsqlite3.dylib
   - libstdc++.dylib
   - libiconv.dylib
   - libc++.dylib

   For more information, see the section *Linking to Library or Framework* in the *Project Editor Help* in the Apple Developer Manual.

7. Add the statement "`#import <SitecoreMobileSDK/SitecoreMobileSDK.h>`" to your project's `*.pch` or `GCC_PREFIX_HEADER` file. For more information, see the section *Build Setting Reference* in the *Apple Developer Manual*.

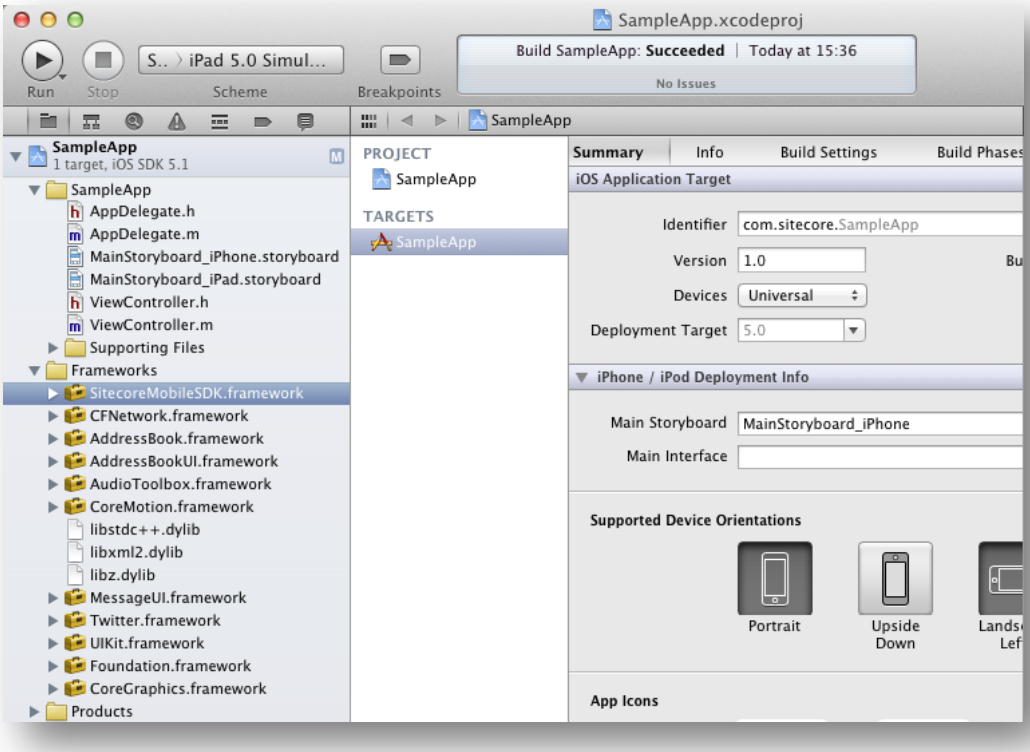
   Example:

```
#ifdef __OBJC__
#import <SitecoreMobileSDK/SitecoreMobileSDK.h>
#import <UIKit/UIKit.h>
#import <Foundation/Foundation.h>
#endif
```

2.2.2 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 the `SitecoreMobileSDK.framework` bundle and drop it into the project's *Frameworks* source group.
3. Build the application.

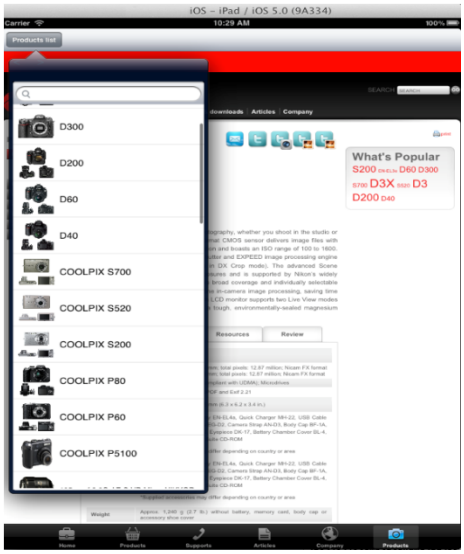


2.2.3 Installing and Using the NicamApp Sample Project

The *NicamApp* project has been created to illustrate how to use the Sitecore Mobile SDK API and the embedded browser to retrieve Sitecore content and present it in an iOS application.

The *NicamApp* contains a **TabBarController** that lists website sections and an embedded web browser for the website section in each tab.

The last tab contains the **View Controller** and a **Table View** that lists all the products on the website.

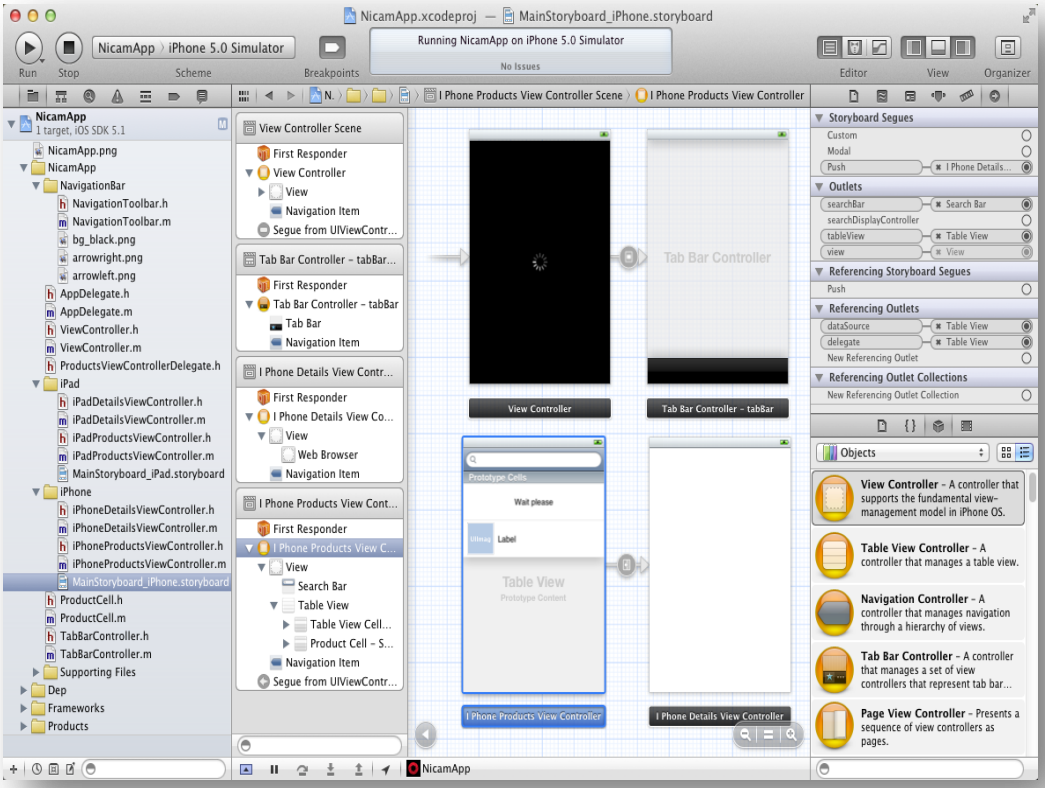


You can select one of the products and view its detailed information in a web browser.

To install the *NicamApp* project:

1. Open the `NicamApp.xcodeproj` project.
2. Drag the `SitecoreMobileSDK.framework` bundle and drop it into the project's *Frameworks* source group.

3. Build the application.



Chapter 3

Getting Started

This chapter presents the different models of working with the Mobile SDK.

This chapter contains the following sections:

- Models of Working with the Mobile SDK
- Getting Started with the Embedded Browser
- Getting Started with the Item Web API Service
- Combining the Embedded Browser and the Item Web API Service

3.1 Models of Working with the Mobile SDK

There are three models that you can apply to work with the Sitecore Mobile SDK:

- Embedded Browser — If you already have a website that is running on Sitecore and optimized for the mobile application, you must use the website presentation. Then, you can display the embedded browser window inside your application. The native device features are typically unavailable in a normal website but you can access them through a set of HTML and JavaScript APIs. For more information about this approach, see the section *Getting Started with the Embedded Browser*.
- Item Web API — If you are familiar with Objective-C development, you can use the native code to develop all or part of your application and access the content in Sitecore to get all the benefits of the Sitecore CMS. For more information about Objective-C API, see the section *Getting Started with the Item Web API Service*.
- Creating a Hybrid of the Embedded Browser and the Item Web API — For example, you can use the native code and UI elements for parts of the screen, such as the standard navigation elements like the tab bar and the navigation bar, and use the embedded browser to display your content. For more information about this approach, see the section *Combining the Embedded Browser and the Item Web API Service*.

3.2 Getting Started with the Embedded Browser

The Sitecore Mobile SDK contains the `SCWebView` classes that extend the `WebView` class with additional features such as sharing on Twitter and left-right swiping. For more information, see the chapter *Using the Enhanced Web View* Reference.

You can use `SCWebView` in the same way as `UIWebView`, as all of their methods are similar.

The following example illustrates how to use the `SCWebView` class:

```
-(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];
}
```

You are now ready to use the `SCWebView` class. You can use the `WebView` enhancements, such as, left-right swipes with any website that is displayed with `SCWebView`.

For example, if you want to browse to `http://mobilesdk.sc-demo.net/Nicam.aspx` on the right swipe and to `http://mobilesdk.sc-demo.net/Products.aspx` on the left swipe, you must turn on swiping and add the links to the web page as follows:

```
<html>
<head>
    <link rel="scm-forward" href="http://mobilesdk.sc-demo.net/Nicam.aspx" />
    <link rel="scm-back"    href="http://mobilesdk.sc-demo.net/Products.aspx" />
```

Swiping is now enabled.

If you also want to use the browser's *Back* and *Forward* navigation controls, use `SCWebBrowser` instead of `SCWebView`. This is because `SCWebBrowser` inherits methods and properties of `SCWebView` and adds navigation controls.

For a complete list of the features that are available in the Embedded Web View, see the chapter *Using the Enhanced Web View Reference*.

3.3 Getting Started with the Item Web API Service

To start working with the Item Web API service, you must create an Xcode project and install the Sitecore Mobile SDK.

The following list is an overview of how to use the Item Web API service:

Establish an anonymous session for a website

Create an object of type `SCApiContext` with a web service host:

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

Now, use this object to access the required items and fields. For more information, see the section *Installing the Client Side Components*.

Get a single item

Use the `[SCApiContext itemReaderForItemPath:]` and `[SCApiContext itemReaderForItemId:]` methods to read the item using the item's path and ID, for example:

```
//Reading an item using its path
 [context itemReaderForItemPath: @"/sitecore/content/nicam"](^(id result, NSError
 *error)
 {
    SCItem* item = result;
    NSLog(@"item display name: %@", item.displayName);
 } );
```

For more information, see the section *Accessing an Item*.

Get the main properties of the item

Use the Item Web API service to get the items and the following properties:

- `DisplayName`
- `ID`
- `LongID`
- `Path`
- `Template`

Use the `SCItem` class to access these properties:

Property	Class
`DisplayName`	`[SCItem displayName]`
`ID`	`[SCItem itemId]`
`LongID`	`[SCItem longID]`
`Path`	`[SCItem path]`
`Template`	`[SCItem itemTemplate]`

For example, the following method shows the item's display name in the console:

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

For more information about the properties of the `SCItem` class, see the section *Accessing an Item*.

The `SCItem` object may contain some or all Sitecore item's fields according to the type value of the `[SCItemsReaderRequest fieldNames]` property of the request.

The `fieldNames` property of the request can be of type: `nil`, `empty set`, or `set of strings`.

- If the type is `nil`, all the fields are read for the item except for the standard fields.
- If the type is `empty set`, only the item is read without the fields.
- If the type is `set of strings`, only the fields in the set are read.

Note
To read a field, its *Read* property must be set to *Allow* in the *Field Remote Read* security settings of Sitecore.

For more information, see the section *Accessing the Fields of an Item*.
To retrieve the item's fields for an extranet user, you should set permission to `Allow` for `Language Read` security of these fields or turn off the `CheckSecurityOnLanguages` property in the `Web.config` file:

```
        <setting name="CheckSecurityOnLanguages" value="false" />
```

Read the item's children

You must use the `[SCApiContext itemsReaderWithRequest:]` method to read an item and its children:

1. Create an `SCItemsReaderRequest` object that contains the set of parameters:

```
        SCItemsReaderRequest *request = [SCItemsReaderRequest new];
        //The path of the item
        request.request = @"/sitecore/content/Nicam";
        //Specifies the type of request - now, the item path used
        request.requestType = SCItemReaderRequestItemPath;
        //Specifies the set of the items that are loaded. Here, the item and its children are
        loaded
        request.scope = SCItemReaderSelfScope | SCItemReaderChildrenScope;
        //The set of the field's names that are read with each item. Here, no fields are read.
        request.fieldNames = [NSSet set];
```

2. Load the items using the created request object:

```
        [context itemsReaderWithRequest: request](^(NSArray* result, NSError *error)
            {
                if ( ![result objectAtIndex:0])
                {
                    NSLog( @"Item not received. Error : %@", error );
                    return;
                }

                //result - is an NSArray object where the first element is the item
                SCItem *item = [result objectAtIndex:0];
                // and other elements are its children
                NSArray* children = [result subarrayWithRange: NSMakeRange(1,[result count]-1)];
                NSLog(@"item display name: %@", item.displayName);
                for (SCItem* child in children)
                {
                    NSLog(@"child display name: %@", child.displayName);
                }
            });
```

For more information, see the section *Accessing the Children of an Item*.

Populate the tab bar

You must use the `SCItemsReaderRequest` class with the `SCItemReaderRequestReadFieldsValues` flag and the `SCItemReaderChildrenScope` scope to load the field values of the item's children.

To populate the tab bar:

1. Create an Xcode Single View Application project.
2. Install the Sitecore Mobile SDK.

   For more information about installing the Mobile SDK, see the chapter *The Mobile SDK Installation*.
3. Add a *Tab Bar Controller* to the project — in the implementation of `ViewController`, add the following code:

```
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];


    NSMutableArray *listOfViewControllers = [NSMutableArray new];
    SCApiContext *session = [SCApiContext contextWithHost:
    @"mobilesdk.sc-demo.net/-/item"];
    NSSet* fieldNames = [NSSet setWithObjects: @"Menu title", @"Tab Icon", nil];
    SCItemsReaderRequest* request = [SCItemsReaderRequest requestWithItemPath:
    @"/sitecore/content/Nicam/" fieldsNames: fieldNames];
    request.flags = SCItemReaderRequestReadFieldsValues; //to read the field values
    request.scope = SCItemReaderChildrenScope; //to read the 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];
    } );

}
```

For more information, see the section *Populating the Tab Bar with the Children of an Item*.

3.4 Combining the Embedded Browser and the Item Web API Service

You can retrieve Sitecore content, and use it in both the Xcode Web View controls and the embedded browser. This example describes how to create a tab bar menu that contains the Sitecore items, titles, and icons and then load the web page in a web browser for each item.

The following procedure describes this scenario:

1. Install the Sitecore Mobile SDK.
2. Create an Xcode project.
3. Create a tab bar menu. For more information about creating a tab bar menu, see the section *Populating the Tab Bar with the Children of an Item*.
4. Extend the project with the following code:

```
#import "ViewController.h"

@interface BrowserViewController : UIViewController

@property ( nonatomic, strong ) NSString* urlString;

@end

@implementation BrowserViewController

@synthesize urlString;

-(SCWebBrowser*)webView
{
    return (SCWebBrowser*)self.view;
}

-(void)loadView
{
    self.view = [SCWebBrowser new];
}

-(void)viewDidLoad
{
    [ super viewDidLoad ];
    NSURL* url = [ NSURL URLWithString: self.urlString ];
    self.webView.scalesPageToFit = YES;
    [ self.webView loadRequest: [ NSURLRequest requestWithURL: url ] ];
}

@end

@implementation ViewController
{
    SCApiContext* _context;
    NSString* rootPath;
}

-(void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear: animated];

    rootPath = @"http://mobilesdk.sc-demo.net";

    if (_context)
        return;

     context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/item"];
```

```
        SCItemsReaderRequest *request = [SCItemsReaderRequest new];
        NSMutableArray* listOfViewControllers = [NSMutableArray new];
        request.request = @"/sitecore/content/Nicam/*[@@templatename='Site Section']";
        request.requestType = SCItemReaderRequestQuery;
        request.fieldNames = [[NSSet alloc] initWithObjects: @"Menu title", @"Tab Icon",
        nil];
        request.flags = SCItemReaderRequestReadFieldsValues;
        [_context itemsReaderWithRequest: request](^(id result, NSError* error_)
        {
            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 = [rootPath stringByAppendingString: item.path];
                [listOfViewControllers addObject: viewController];
            }
            [self performSegueWithIdentifier: @"showTabBar" sender: self];
            UITabBarController *tabBar = (UITabBarController*)self.modalViewController;
            [tabBar setViewControllers:listOfViewControllers animated:YES];
        });
    }

    @end
```

5. Build and run the application.

Chapter 4

Using the Enhanced Web View Reference

This chapter describes the main features of the Enhanced Web View Reference. Each section in this chapter presents a feature, its usability, and an example.

This chapter contains the following sections:

- The Left-Right Swipe Link Elements
- Getting an Image from the Device
- Getting the Accelerometer Data
- Getting the Device Information
- Managing Contacts
- Sending Tweets
- Sending E-Mails
- Using JavaScript Native Alert
- Reading QR Codes Using Objective-C API
- Map Navigation

4.1 The Left-Right Swipe Link Elements

You can use the *left-right swipe link* elements to support page changing with right-left swiping.

You must add the following links 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 is not possible to return to the previous page using the opposite swipe gesture. However, you can add this feature by adding the link element, possibly with an empty `href`, as follows:
`<link rel="scm-back"/>`

You can also use the `swipeForward()` and `swipeBack()` functions to initiate swipe navigation event directly from JavaScript code.

For example:

```
<script language="javascript" type="text/javascript">
function next page()
{
    scmobile.console.log('swipeForward');
    scmobile.navigations.swipeForward();
}
function previous_page()
{
    scmobile.console.log('swipeBack');
    scmobile.navigations.swipeBack();
}
</script>
<a href="#" onclick="previous_page()">Previous</a>
<a href="#" onclick="next_page()">Next</a>
```

Note
You can use the Mobile SDK functions such as `scmobile.console.log` only after the `scmobileReady` event is handled. Otherwise, the script crashes and throws an exception.

You can also add a listener to the event:
`document.addEventListener('scmobileReady', onDeviceReady, false);`

4.1.1 The Mobile Swipes Component for Sitecore

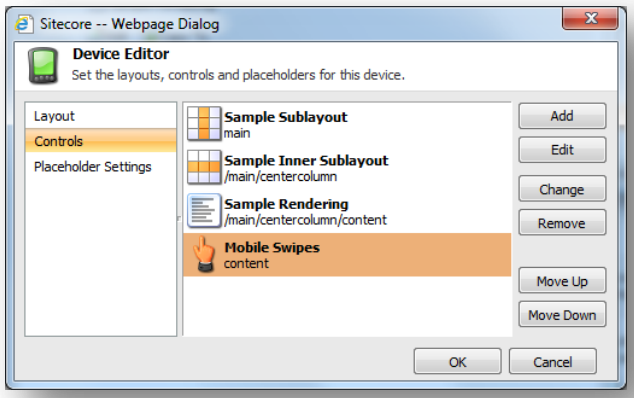
You can use the Sitecore *Mobile Swipes* component instead of the `scm-forward` and `scm-back` functions.

To use the *Mobile Swipes* component:

1. Install the *Mobile SDK Server Package* component.

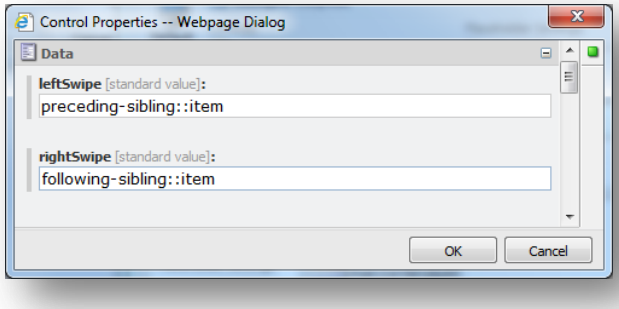
For more information, see the section *Installing the Server Side Components*.

2. Add the *Mobile Swipes* component to the desired placeholder on the page:



Note
You must add the swipe component to a placeholder on a page and not to a layout or sublayout. It does not matter where the placeholder is placed on the page. The *Mobile Swipes* component is not displayed on the page.

3. Set the *leftSwipe* and *rightSwipe* fields of the *Mobile Swipes* component. For the current item, you can use Sitecore queries as parameters of these fields.



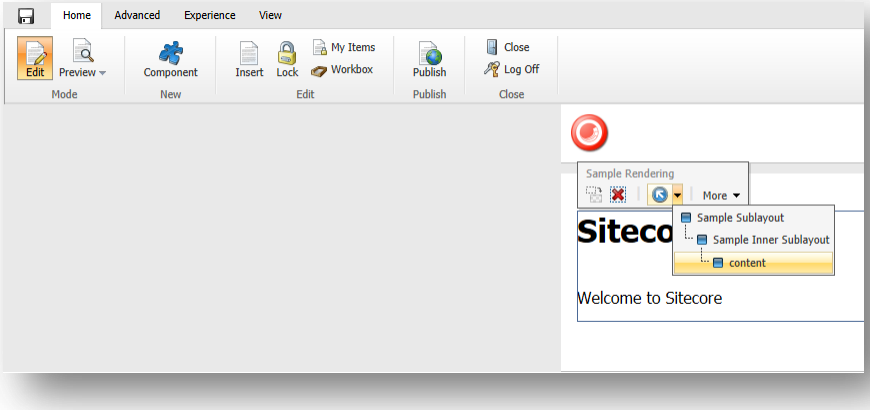
4. Publish your website.

4.1.2 Adding the Mobile Swipes Components in the Page Editor

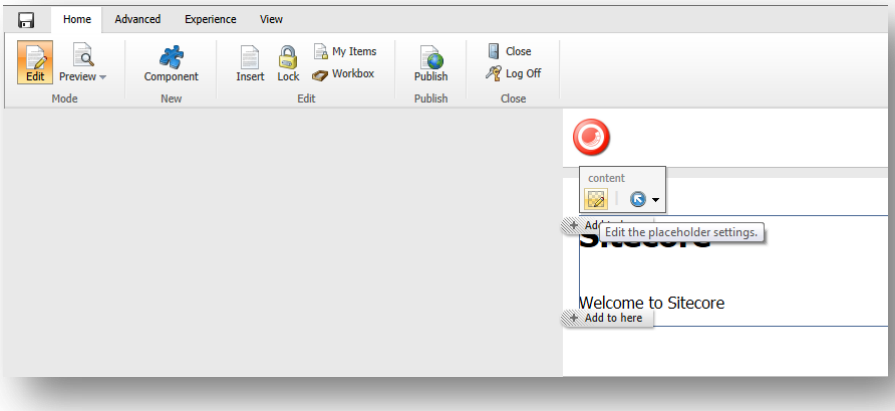
To add the *Mobile Swipe* components in the **Page Editor**:

1. Open the **Page Editor**.
2. Navigate to the page that you want to add the *Mobile Swipes* component to.
3. Select the placeholder that you want to add the *Mobile Swipes* component to and the floating toolbar appears.

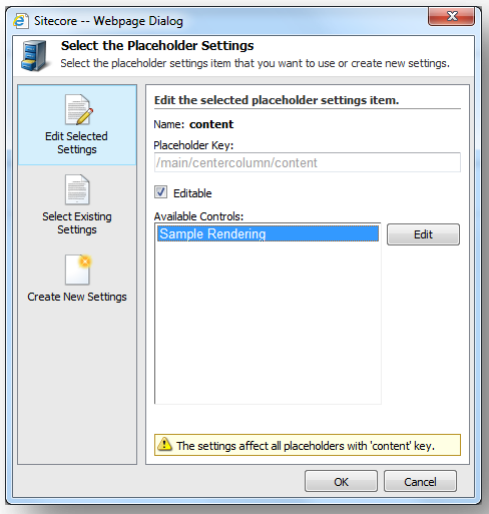
4. In the floating toolbar, click the **Show ancestors** drop-down arrow and then click **content**.



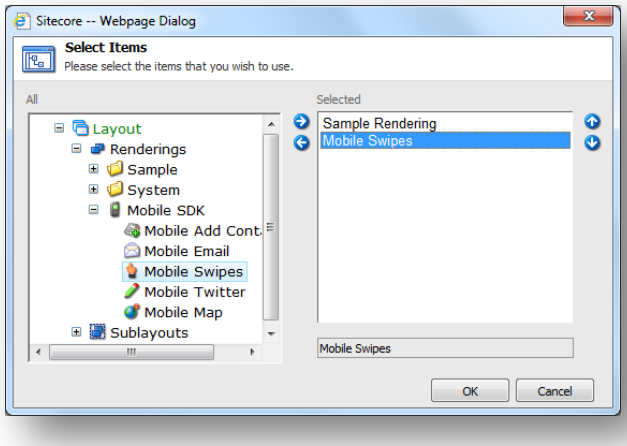
5. Edit the item place holder settings of **Content**.



6. Click **Edit**.

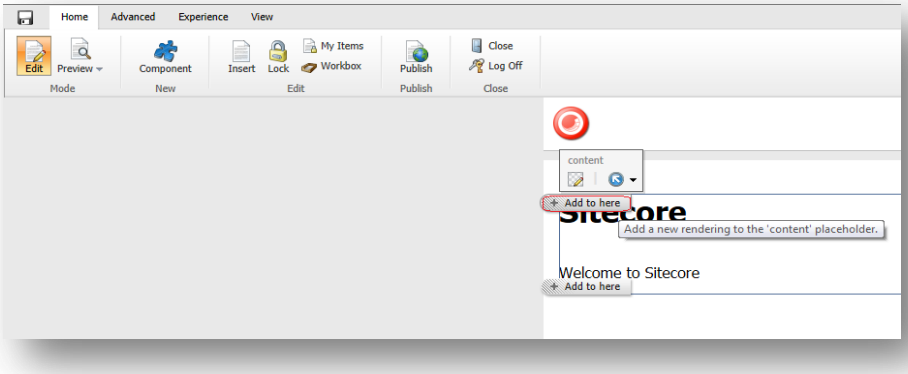


7. Add the **Mobile Swipes** component to the Available Controls.

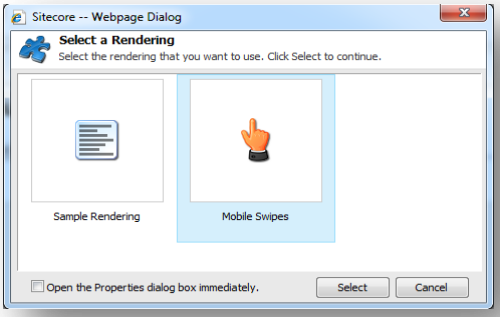


8. Select the placeholder and click **OK** to add the component to the placeholder.

9. Add a new rendering to **Content** placeholder.



10. Choose the **Mobile Swipes** rendering component and click **Select**.



11. Publish your website.

4.2 Getting an Image from the Device

You can get a snapshot from the device's photo camera or get one of the images that are saved in the photo library.

To get a snapshot from the camera of the mobile device, use the following function:

```
scmobile.camera.getPicture( when_ok, when_error, options );
```

The following example illustrates how to use this function to get an image using the image URI as a parameter:

```
function get_image()
{
  function when_ok(imageURI)
  {
    scmobile.console.log('getPicture OK: ' + imageURI);
    var image = document.getElementById('myImage');
    image.src = imageURI;
   }
  function when_error() {}
  var options = {};
  options.sourceType = scmobile.camera.PictureSourceType.CAMERA;
  scmobile.camera.getPicture(when_ok, when_error, options);
}
```

You can get an image from the device's photo album or photo library:

- To take a picture, use the `scmobile.camera.PictureSourceType.CAMERA` source type.
- To use the image from the photo album, use the `scmobile.camera.PictureSourceType.SAVEDPHOTOALBUM` source type.
- To use the image from the photo library, use the `scmobile.camera.PictureSourceType.PHOTOLIBRARY` source type.

4.3 Getting the Accelerometer Data

To capture the device motion in the x, y, and z dimensions of the accelerometer, use the following function:

```
var accelerometer = new scmobile.motion_manager.Accelerometer();
```

You can use the `accelerData` parameter to get the XYZ motion.

To stop accelerometer, use the following function:

```
accelerometer.stop();
```

The following example illustrates how to use the XYZ dimensions and the `stop` function:

```
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();
   }
}
```

4.4 Getting the Device Information

To get the device information, such as the version, the name, and the universal unique identifier (UUID), use the following properties:

```
var device_version = scmobile.device.version;
var device_name = scmobile.device.name;
var device_uuid = scmobile.device.uuid;
```

4.5 Managing Contacts

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

4.5.1 Creating a Contact

To create a contact, use the following JavaScript function:

```
var contact = scmobile.contacts.create({firstName: 'FName', lastName: 'LName'});
contact.phones = ['333-555'];
contact.silentSave(onSuccess, onError);
```

The contact with the specified fields is created and is added to the device list of contacts.

The `create` function does not contain required fields. You can specify a value for each field or you can leave them empty.

Note
If the contact already exists, the function will create another contact *even if they are identical*.

The `silentSave` function takes the two callback functions — `onSuccess` and `onError` as parameters.

You can assign values to the following fields:

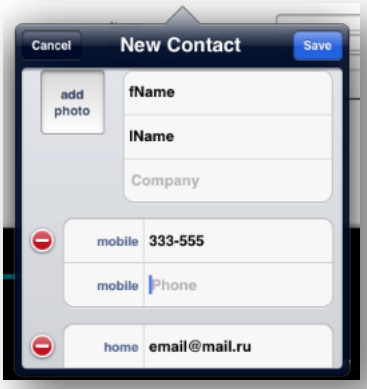
- `firstName` of type `string`
- `lastName` of type `string`
- `company` of type `string`
- `emails` of type `array`
- `phones` of type `array`
- `websites` of type `array`
- `photo` of type `Image`
- `birthday` of type `Date`
- `addresses` of type `Array`

Example:

```
var contact = scmobile.contacts.create({firstName: 'FName', lastName: 'LName',
company: 'Company'});
contact.phones = ['333-555', '+38(050)6663344'];
contact.emails = ['email@mail.ru'];
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(contact)
{
    window.alert("Contact  was successfully created.");
}
var onError = function(error)
```

```
        {
           window.alert("Cannot create a contact.");
        }
        contact.silentSave(onSuccess, onError);
```

If you use the `save` function instead of the `silentSave` function, a standard Apple Contact's window is displayed.



You can edit the information in this window and then click **Save** or **Cancel**.

4.5.2 Finding a Contact

To find a contact, use the following function:

```
        var predicate = function(contact)
        {
        return (contact.firstName == 'FName' && contact.lastName == 'LName')
        };
        scmobile.contacts.silentSelect(predicate, onSuccess, onError);
```

In the predicate expression, you can use any logical expression. You must use the `&&` operator for *AND*, the `||` operator for *OR*. You can also search with any of the fields in a contact.

The `find` function returns an array of contacts according to the predicate expression.

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.silentSelect(predicate, onSuccess, onError);
    }
```

Note
The function `scmobile.contacts.silentSelect` retrieves all available contacts from the address book and executes the predicate on the JavaScript side. Overusing this function results in performance issues.

4.5.3 Removing a Contact

To remove a contact, use the following function:

```
    contact.remove(onSuccess, onError);
```

The `remove` function has the 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);
    }
```

4.5.4 Using the Mobile Add Contact Component in Sitecore

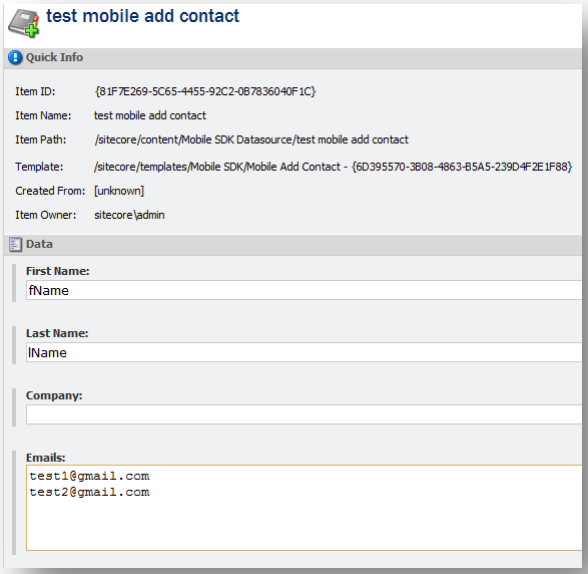
You can use the Sitecore *Mobile Add Contact* component instead of the JavaScript function that is described in the *Creating a Contact* section.

To use the *Mobile Add Contact* component:

1. Install the Mobile SDK Server Package.

   For more info about the installation, see the section *Installing the Server Side Components*.

2. Create an item based on the *Mobile Add Contact* template — `/sitecore/templates/SCMobile/Mobile Add Contact`



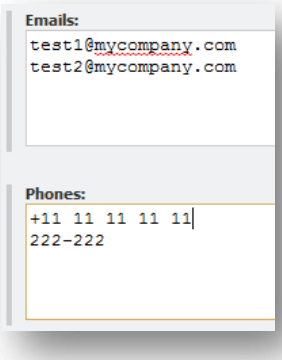
The Data section contains the following fields:

Field	Type
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
Icon	Image

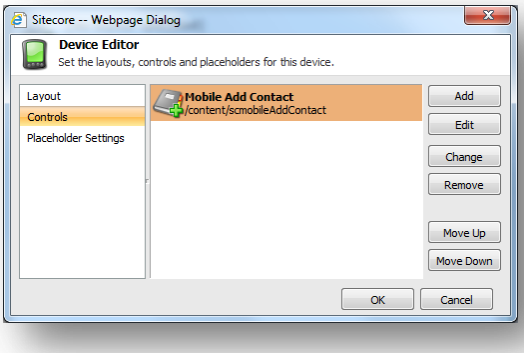
The Address fields are:

Field	Type
Street	Single-Line Text
City	Single-Line Text
State	Single-Line Text
Zip	Single-Line Text
Country	Multi-Line Text

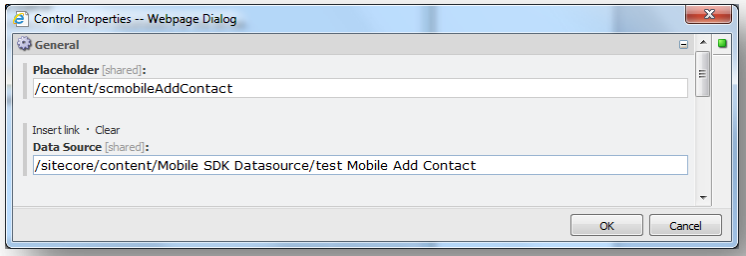
The following image shows the e-mail addresses and phone numbers:



3. In the **Template Manager** or the **Content Editor**, edit the standard values item or the individual item.
4. Click the Presentation tab.
5. On the Presentation tab, in the Layout group, click the Details command. The **Layout Details** dialog appears.
6. In the Layout Details dialog, below the device for which you want to configure layout details, click **Edit**. The **Device Editor** appears.
7. In the **Device Editor**, click the Controls tab, and then click Add. The **Select a Rendering** dialog appears.
8. In the **Select a Rendering** dialog, select the *Mobile Add Contact* rendering.
9. In the **Select a Rendering** dialog, click the *Mobile Add Contact* rendering, enter the placeholder path, and then click **Select**.
10. In the **Device Editor**, in the Controls tab, select *Mobile Add Contact*, and click **Edit**



11. Set the created item as a Data source to the *Mobile Add Contact* component and then click **OK**.



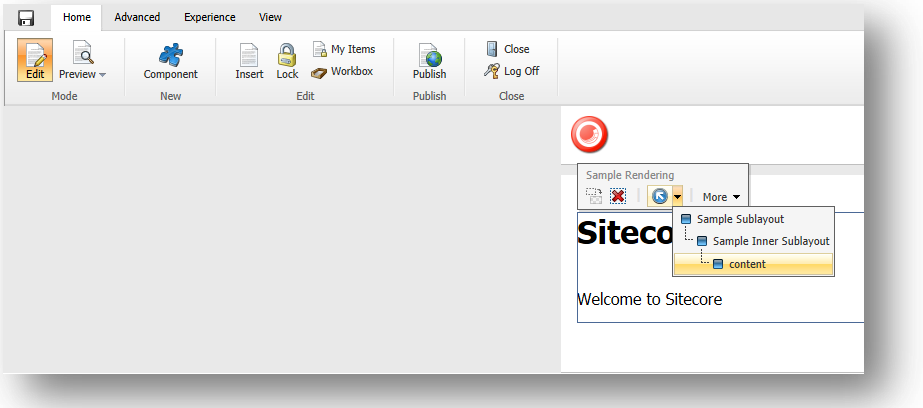
12. Publish your website.
13. Run the application in an iOS device, click the **Add a Contact** button and then you should see a standard Apple Contact's window.

Note
If the contact already exists, the function will create another contact *even though they are exactly the same*.

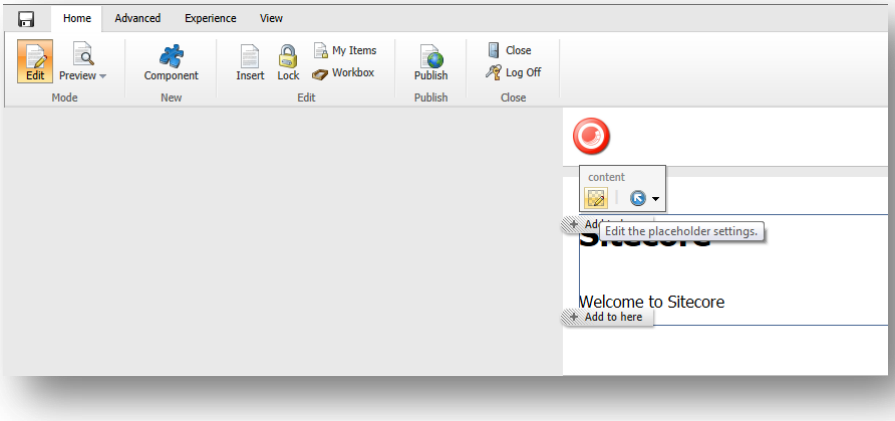
4.5.5 Using the Mobile Add Contact Component in the Page Editor

To add the *Mobile Add Contact* component: in the **Page Editor**:

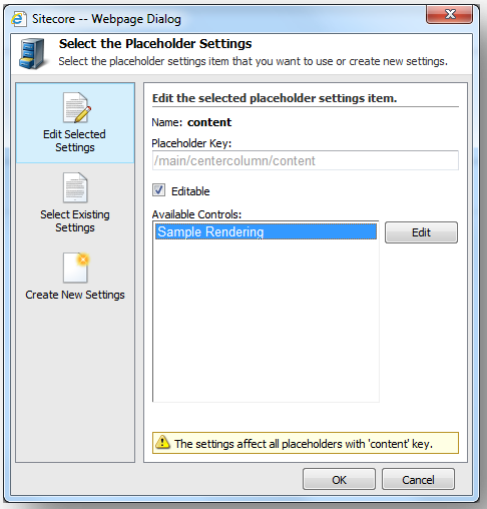
1. Open the **Page Editor**.
2. Navigate to the page that you want to add the *Mobile Add Contact* component to.
3. Select the placeholder that you want to add the *Mobile Add Contact* component to and the floating toolbar appears.
4. In the floating toolbar, click the **Show ancestors** drop-down arrow and then click **content**.



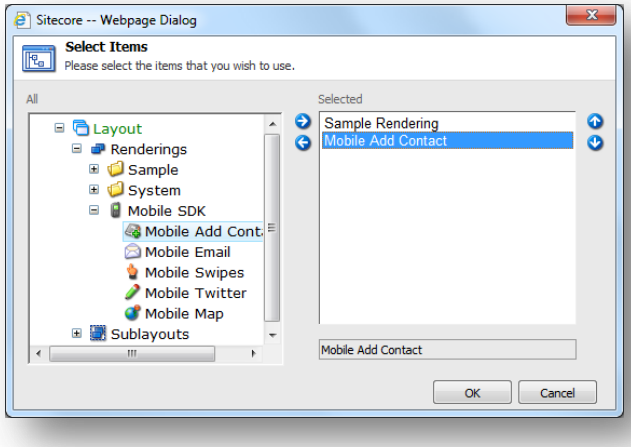
5. Edit the item place holder settings of **Content**.



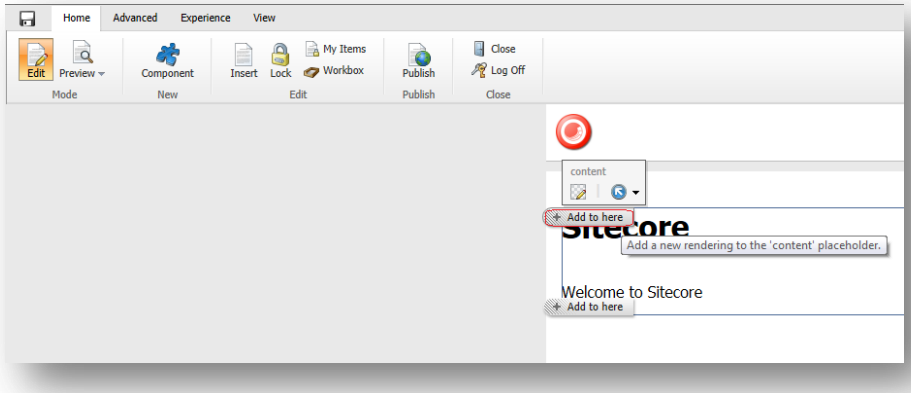
6. Click **Edit**.



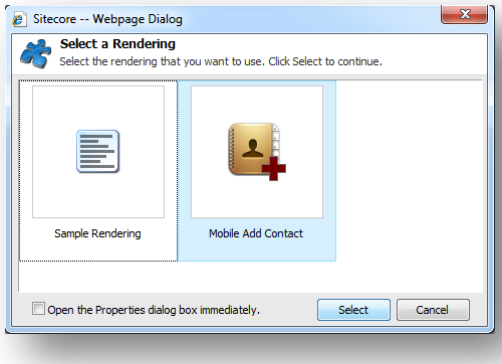
7. Add the *Mobile Add Contact* component to the Available Controls.



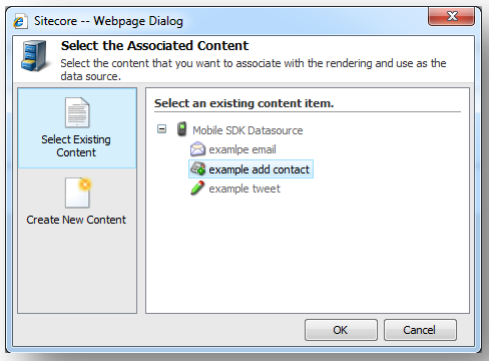
8. Select the placeholder and click **OK** to add the component to the placeholder.
9. Add a new rendering to **Content** placeholder.



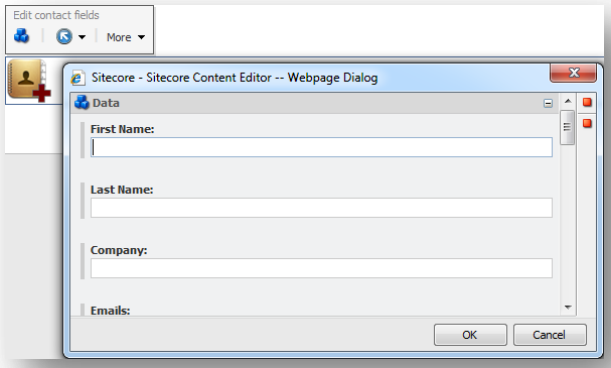
10. Choose the **Mobile Add Contact** rendering component and click **Select**.



11. You can choose **example add contact** or create a new data source item.



12. You can click the component icon and edit the data source item fields in the **Page Editor**.



13. Publish your website.

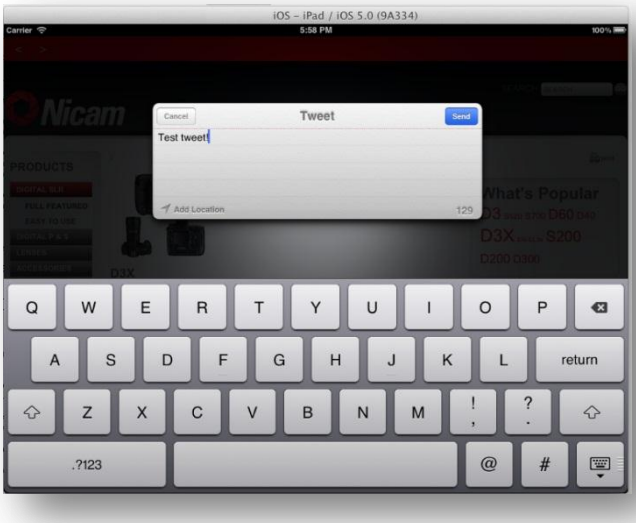
4.6 Sending Tweets

You can send tweets using the Mobile SDK functions. You can add an image and a link to the message.

To send a tweet, use the following function:

```
var tweet_ = new scmobile.share.Tweet();
tweet_.text = "Test tweet!";
tweet_.send(onSuccess, onError);
```

When you call the `Send` function, you see a **Twitter** window that contains the predefined message.



You can edit this information, and then tap **Send** or **Cancel**.

Example:

```
function twitter()
{
    var tweet_ = new scmobile.share.Tweet();
    tweet_.text = "Test tweet!";
    tweet_.urls.push("www.sitecore.net");
    tweet_.imageUrls.push(myImageUrl);

    function onSuccess ()
    {}
    function onError (error)
    {
       scmobile.console.log('errorSend: ' + error);
    }
    tweet_.send(onSuccess, onError);
}
```

4.6.1 Using the Mobile Tweet This Component in Sitecore

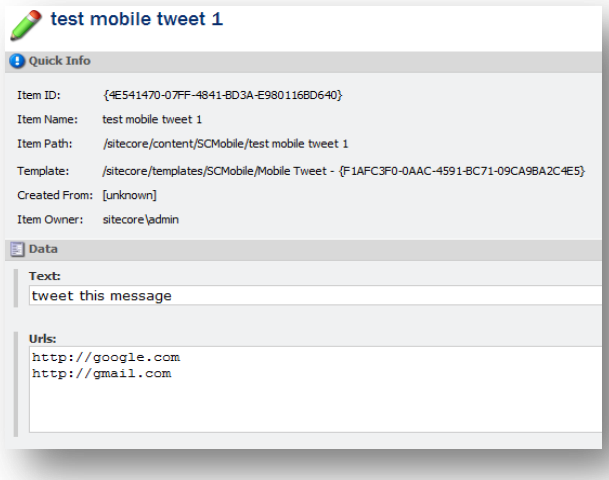
You can use the Sitecore *Mobile Tweet This* component instead of the JavaScript `Tweet` function that is mentioned in the introduction of this section.

To use the Sitecore *Mobile Tweet This* component:

1. Install the *Mobile SDK Server Package* component

   For more information about installing the server package, see the section *Installing the Server Side Components*.

2. Create an item based on the *Mobile Tweet This* template:
   `/sitecore/templates/SCMobile/Mobile Tweet`



The following table contains the fields types of the item that you are creating:

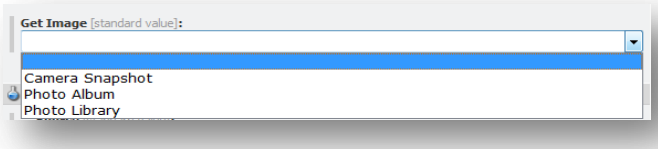
Field	Type
Text	Single-Line Text
Urls	Multi-Line Text **Note** Each URL must be entered on a separate line.
Icon	Image
GetImage	Droplist

3. In the *GetImage* field, specify the method that you want to use to get an image.

   You can get an image from the device using the one of the following options:

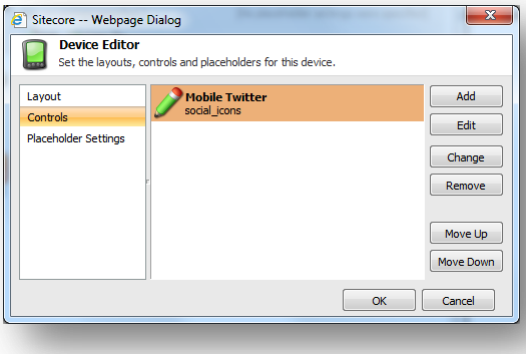
   - None — tweets the message without image
   - Camera Snapshot —tweets the message with a camera snapshot.
   - Photo Album — opens a Photo Album window from which you can select a photo.

     Photo Library — opens a Photo Library window from which you can select a photo.

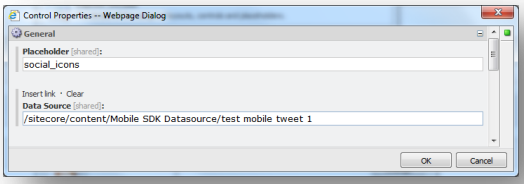


4. In the **Template Manager** or the **Content Editor**, edit the standard values item or the individual item.
5. Click the Presentation tab.
6. On the Presentation tab, in the Layout group, click the Details command. The **Layout Details** dialog appears.
7. In the Layout Details dialog, below the device for which you want to configure layout details, click **Edit**. The **Device Editor** appears.
8. In the **Device Editor**, click the Controls tab, and then click **Add**. The **Select a Rendering** dialog appears.
9. In the **Select a Rendering** dialog, select the *Mobile Twitter* Contact rendering.
10. In the **Select a Rendering** dialog, click the *Mobile Twitter* rendering, enter the placeholder path, and then click **Select**.

    In the **Device Editor**, in the Controls tab, select *Mobile Twitter*, and click **Edit**.



11. Set the created item as a Data source to the *Mobile Twitter* component and click **OK**.



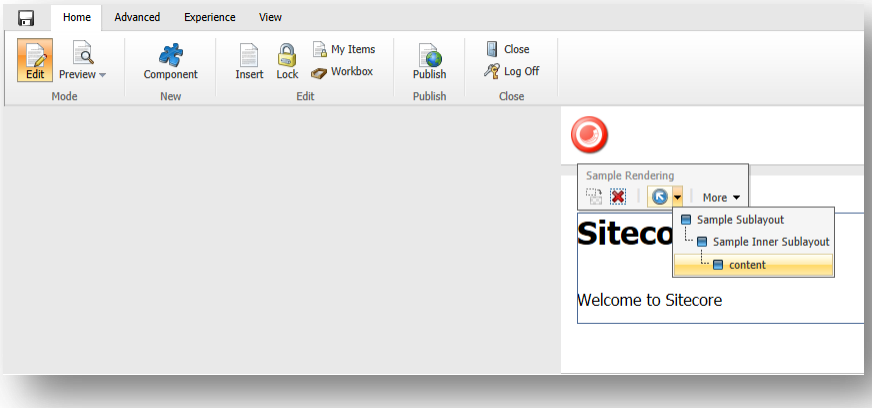
12. Publish your site.
13. Run the application in an iOS device, click the **Tweet** button, and the Twitter window with the predefined message and links appears. You should also see a snapshot if the `Get Image` field is not empty.

4.6.2 Adding the Sitecore Mobile Twitter Component in the Page Editor

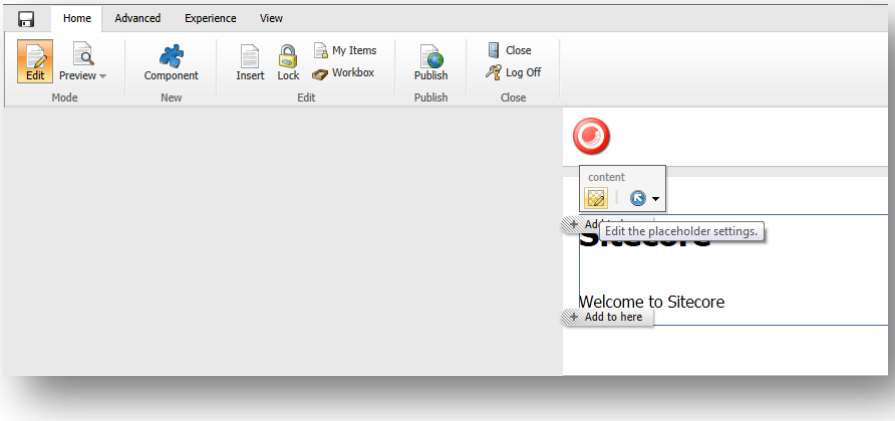
You can also use the **Page Editor** to add the *Mobile Twitter This* component.

To use the **Page Editor** to add the *Mobile Twitter* component:

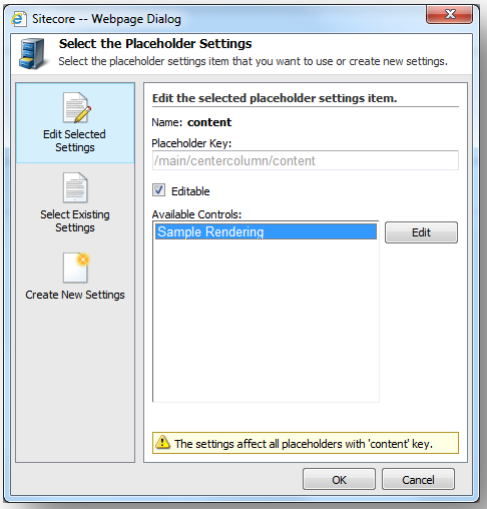
1. Open the **Page Editor**.
2. Navigate to the page that you want to add the *Mobile Twitter* component to.
3. Select the placeholder that you want to add the *Mobile Twitter* component to and the floating toolbar appears.
4. In the floating toolbar, click the **Show ancestors** drop-down arrow and then click **content**.



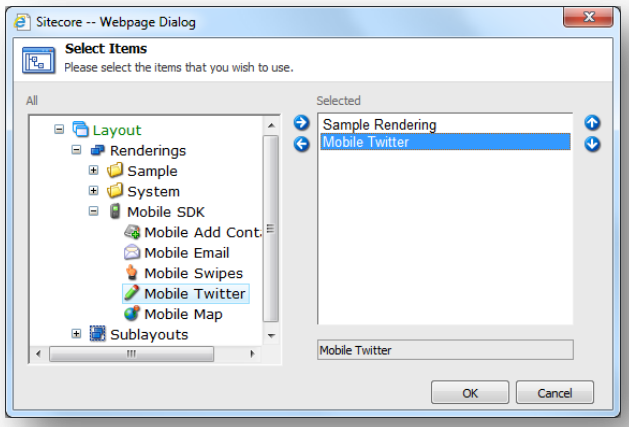
5. Edit the item place holder settings of **Content**.



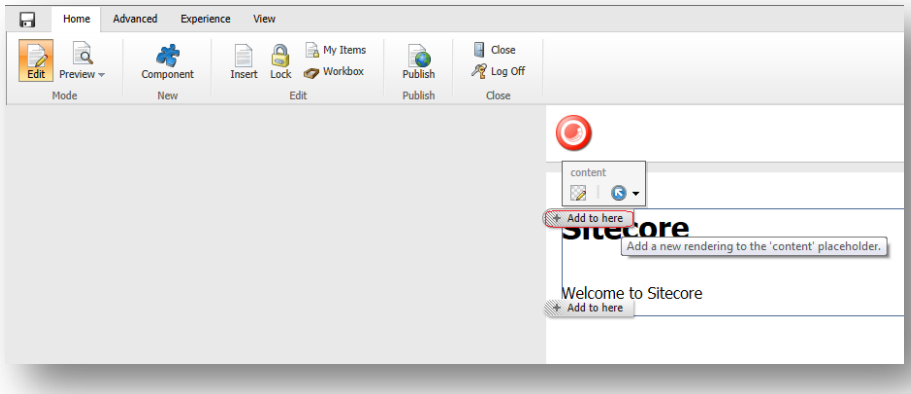
6. Click **Edit**.



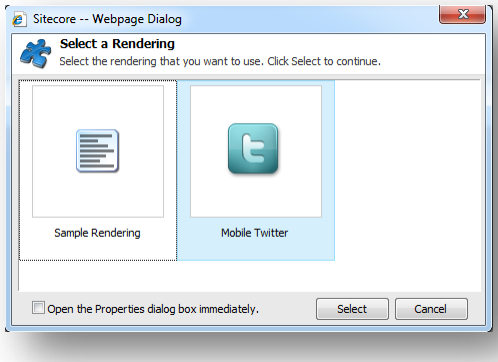
7. Add the *Mobile Twitter* component to the Available Controls.



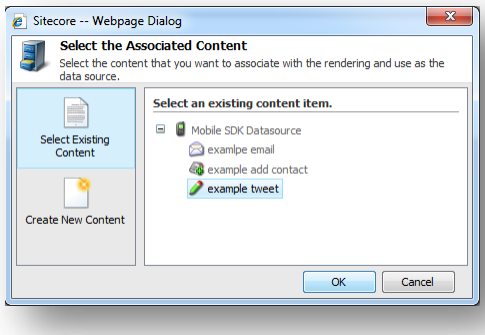
8. Select the placeholder and click **OK** to add the component to the placeholder.
9. Add a new rendering to **Content** placeholder.



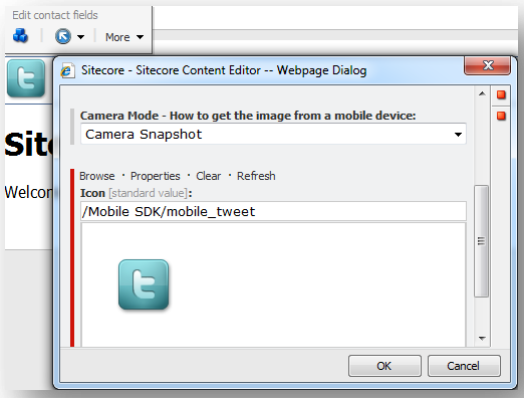
10. Choose the **Mobile Twitter** rendering component and click **Select**.



11. You can choose **example tweet** or create a new data source item.



12. You can click the component icon and edit the fields of the data source item in the **Page Editor**.



13. Publish your website.

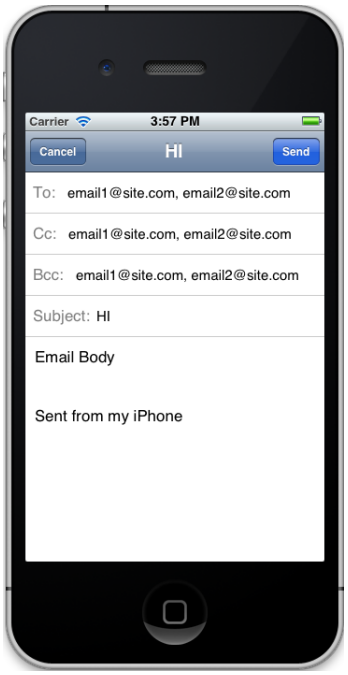
4.7 Sending E-Mails

You can use the Mobile SDK functions to send e-mails.

The following example illustrates how to display an e-mail screen with the specified recipients, subject, and message body:

```
var email = new scmobile.share.Email();
email.toRecipients = ['email1@site.com', 'email2@site.com'];
email.ccRecipients = ['email3@site.com', 'email4@site.com'];
email.bccRecipients = ['email5@site.com', 'email6@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);
```

The following image illustrates a sample output:



You can edit this information and then tap **Send** or **Cancel**.

To send e-mails in HTML format, you must add the `email.isHTML = true;` statement to the previous example.

Note
You must also use the absolute media URLs instead of the relative ones.

Incorrect:
```
email.messageBody = ' <img width="205" height="73"
src="~/media/Images/Nicam/Home/header_logo.ashx" alt="Nicam Logo" /> ';
```

Correct:
```
email.messageBody = ' <img width="205" height="73" src="http://mobilesdk.sc-
demo.net/~/media/Images/Nicam/Home/header_logo.ashx" alt="Nicam Logo" /> ';
```
You must write `"http://"` since the Sitecore Mobile SDK does not support URL autocomplete in the `Email` class.

4.7.1 Using the E-Mail Component in Sitecore

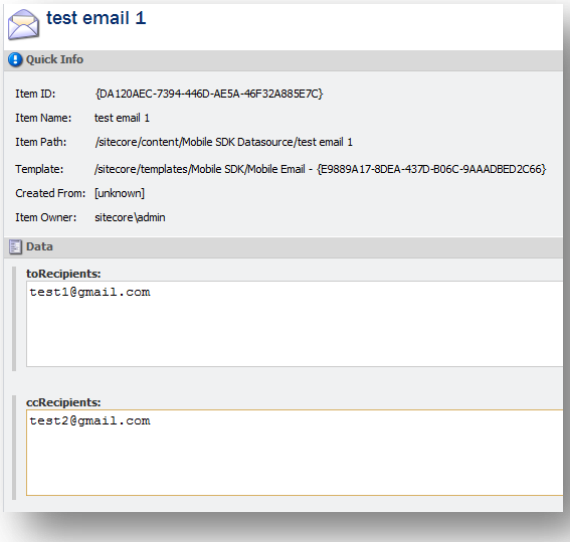
You can use the Sitecore the *Email* component instead of the `Email` JavaScript function that is mentioned in the introduction of this section.

To use the *Email* component:

1. Install the Mobile SDK Server package.

   For more information, see the section *Installing the Server Side Components*.

2. Create an item based on the *Mobile Email* template:
   `/sitecore/templates/SCMobile/Mobile Email`;

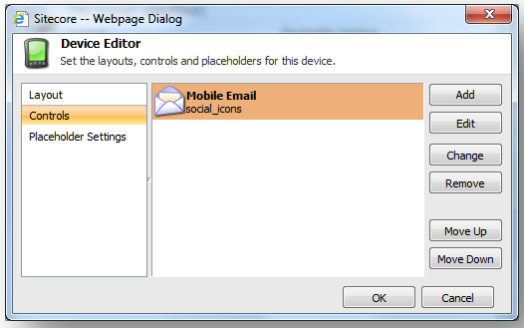


3. You can use the following fields:

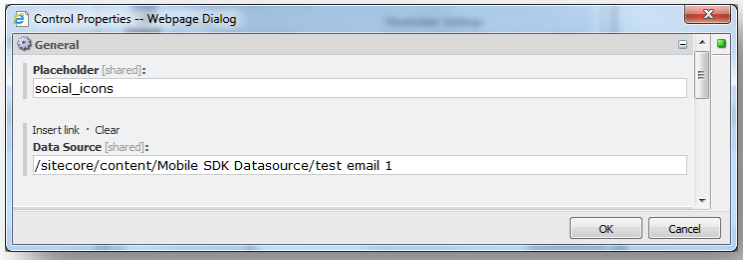
Field	Type
toRecipients	Multi-Line Text **Note** You must enter each recipient on a separate line.
ccRecipients	Multi-Line Text **Note** You must enter each recipient on a separate line.
bccRecipients	Multi-Line Text **Note** You must enter each recipient on a separate line.
Subject	Single-Line Text
Message Body	Rich Text
Icon	Image

4. In the Template Manager or the **Content Editor**, edit the standard values item or the individual item.
5. Click the Presentation tab.
6. On the Presentation tab, in the Layout group, click the Details command. The **Layout Details** dialog appears.
7. In the **Layout Details** dialog, below the device for which you want to configure layout details, click Edit. The **Device Editor** appears.
8. In the **Device Editor**, click the Controls tab, and then click **Add**. The **Select a Rendering** dialog appears.
9. In the **Select a Rendering** dialog, select the *Mobile Email* Contact rendering.
10. In the **Select a Rendering** dialog, click the *Mobile Email* rendering, enter the placeholder path, and then click **Select**.

11. In the **Device Editor**, in the Controls tab, select *Mobile Email*, and click **Edit**.



12. Set the created item as a data source for the *Mobile Email* component.

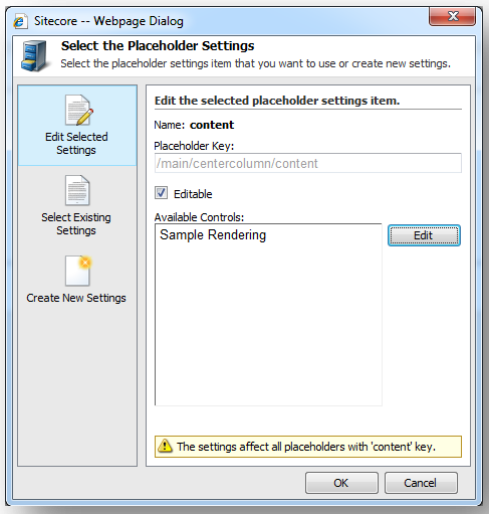


13. Publish your website
14. Run your application on an iOS device, tap **Email**, and then you must you see the **Email** window with the list of recipients, the subject, and the message body.

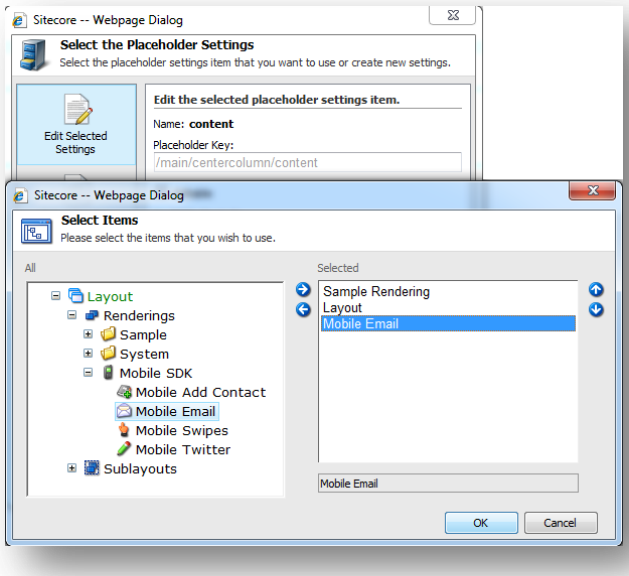
4.7.2 Adding the Mobile Email Component in the Page Editor

To add the *Mobile Email* component in the **Page Editor**:

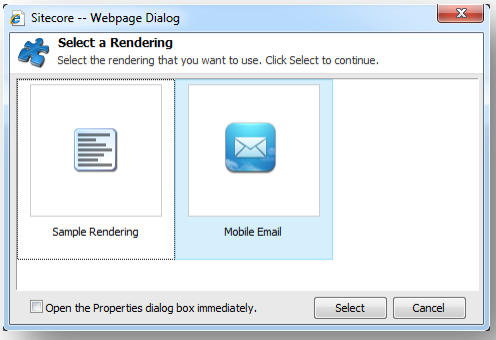
1. Open the **Page Editor**.
2. Select a placeholder to which you want to add the component and open the **placeholder settings**.



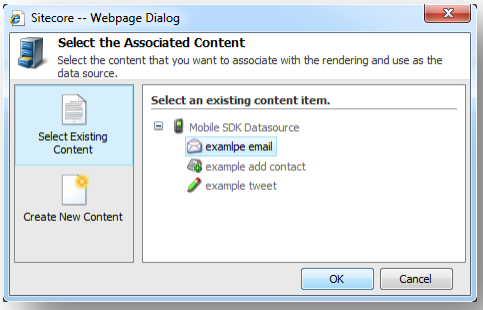
3. Add the *Mobile Email* component to the available controls.



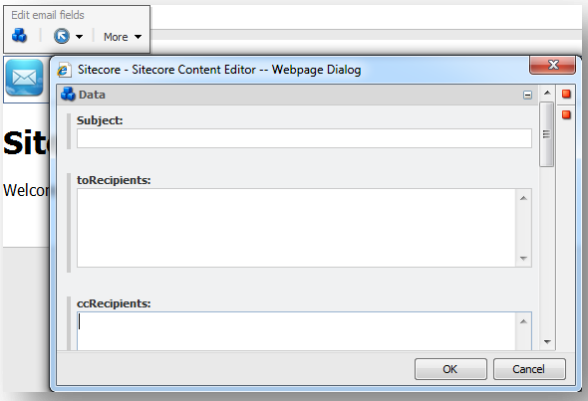
4. Select the placeholder and click **OK** to add the component to the placeholder.



5. You can choose *example tweet* or create a new data source item.



6. You can click the component icon and edit the data source item fields in the **Page Editor**.



7. Publish your website.

4.8 Using JavaScript Native Alert

JavaScript Native Alert is a custom alert or dialog box that is used to customize the alerts. This is instead of the browser's alert function which is less customizable.

The following code snippet illustrates the `alert` function:

```
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);
```

The `alertCallback` function handles the Button Pressed event: The `buttonIndex` variable returns the index of the pressed button. The index of the first pressed button is `0` — zero.

Example:

```
scmobile.notification.alert("Caption" ,"Alert text", function(buttonIndex)
{ /*add your code here; */ }, "Ok, Cancel");
```

4.9 Reading QR Codes Using Objective-C API

To read QR codes, use the `IBOutlet` property of type `SCQRCodeReaderView`:

1. In the `ViewController.h` file , create the `IBOutlet` property:

```
@property (nonatomic, weak) IBOutlet SCQRCodeReaderView* codeReaderView;
```

2. Specify that your view controller responds to the `SCQRCodeReaderViewDelegate` protocol:

```
@interface ViewController : UIViewController <SCQRCodeReaderViewDelegate>
```

3. Synthesize the `codeReaderView` property:

```
@synthesize codeReaderView;
```

4. Implement the method in the `SCQRCodeReaderViewDelegate` protocol.

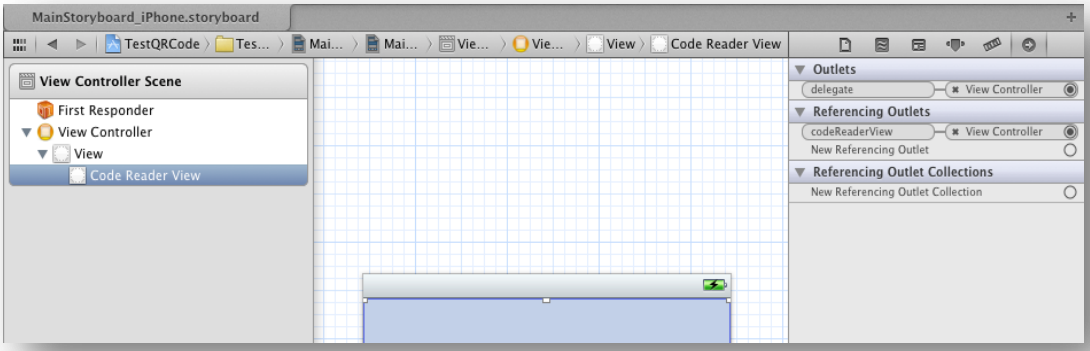
```
- (void)qrCodeReaderView:(SCQRCodeReaderView*)readerView
didGetScanResult:(NSString*)resultString
{
 NSLog(@"Scan result %@", resultString);
}
```

This method is called when the scanner reads the result.

5. Call the `startCapture` and `stopCapture` methods in the `SCQRCodeReaderView` property to start and stop the Capture process by calling the and methods

```
-(void)viewWillAppear:(BOOL)animated
{
 [super viewWillAppear: animated];
 [self.codeReaderView startCapture];
}
```

6. In the *Storyboard* file, add a new sub-view to the `ViewController` view and set its type to `SCQRCodeReaderView`. Link it with your outlet property and set the delegate property to the *View* controller:



In the Storyboard file you can specify the size of `SCQRCodeReaderView`.

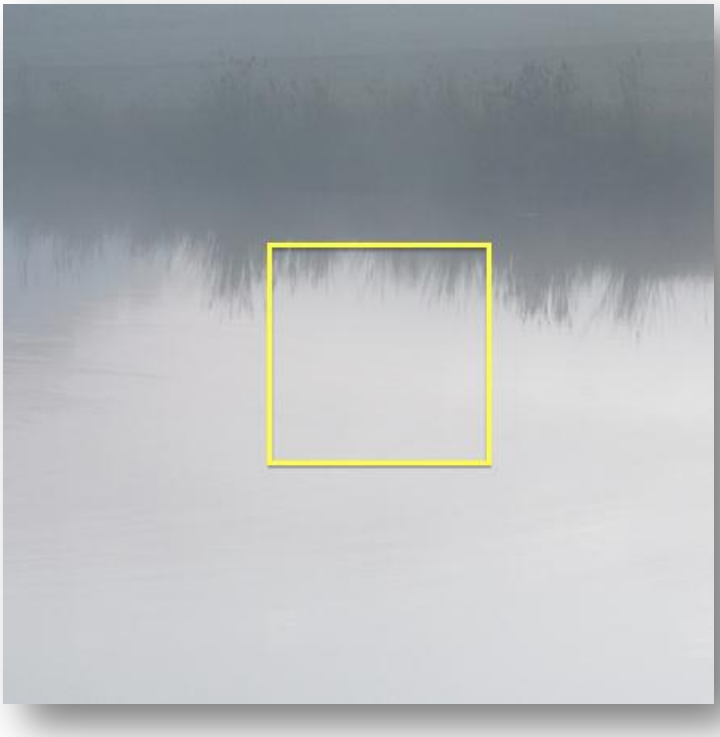
4.9.1 Additional Functionality

The `SCQRCodeReaderView` contains a `captureRect` property:

```
@property (nonatomic) CGRect captureRect;
```

The `captureRect` property specifies the rectangular space in `SCQRCodeReaderView` from which you can scan the picture.

This property is useful for the iPad screen because you can create `SCQRCodeReaderView` with a larger frame for a better view. In this case, you must define `captureRect` with a rectangle to only scan the `SCQRCodeReaderView` area.



Use the `isCaptureInProgress` property to check whether the capturing process is running or not:

```
- (BOOL)isCaptureInProgress;
```

To create `SCQRCodeReaderView` manually, without using the outlet in the Storyboard file, call the appropriate constructor:

```
SCQRCodeReaderView* codeReaderView = [SCQRCodeReaderView viewWithDelegate: self
captureRect: self.view.bounds];
codeReaderView.frame = self.view.bounds;
[self.view addSubview: codeReaderView];
```

4.10 Map Navigation

You can use the Map Navigation functionality in the Sitecore Mobile SDK to locate addresses in a map. You can also draw a route from the user's current location to the nearest address in the addresses list.

4.10.1 The Map Navigation Objective-C API

You can use the Sitecore Mobile SDK to create a Map View in your iOS application and retrieve Sitecore content from the Map View data source.

To use the Map Navigation Objective-C API:

1. In the `ViewController.h` file, create the `IBOutlet` property:

```
@property (nonatomic, weak) IBOutlet SCMapView* mapView;
```

2. Define that your view controller supports the `SCMapViewDelegate` protocol:

```
@interface ViewController : UIViewController <SCMapViewDelegate>
```

3. Synthesize the `mapView` property:

```
@synthesize mapView;
```

4. To show the address items on the map, use the `-[SCMapView addItemsAnnotationsForQuery: apiContext: handler]` method or the `-[SCMapView addItemsAnnotationsWithPath: apiContext: handler]` method.

5. In the Storyboard file, add a new sub-view to the `ViewController` view, set its type to `SCMapView`, link it with your outlet property, and then set the delegate property to the View controller.

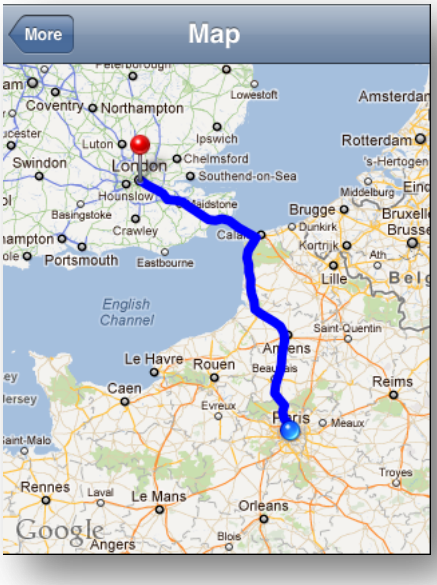
Example:

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

//get the content using a query
NSString* query = @"/sitecore/content/Mobile SDK
Datasource/descendant::*[@@templatename='Mobile Address']";
[mapView addItemsAnnotationsForQuery: query
apiContext: context handler: ^(NSError* error) {NSLog(@"error: %@", error); }];

//get the content using its path
NSString* path = @"/sitecore/content/Mobile SDK Datasource";
[mapView addItemsAnnotationsWithPath: path apiContext: context handler: ^(NSError*
error)
{NSLog(@"error: %@", error);}];
```

Use the `drawRouteToNearestItemAddress` property to specify whether or not you want to draw a route between the user's current location and nearest address in the list of addresses.

```
mapView.drawRouteToNearestAddress = true;
```

By default, the value of the `drawRouteToNearestItemAddress` property is `true`. This means that the map finds the nearest address to the user's current location, resizes to show these two points, and draws the route between them.



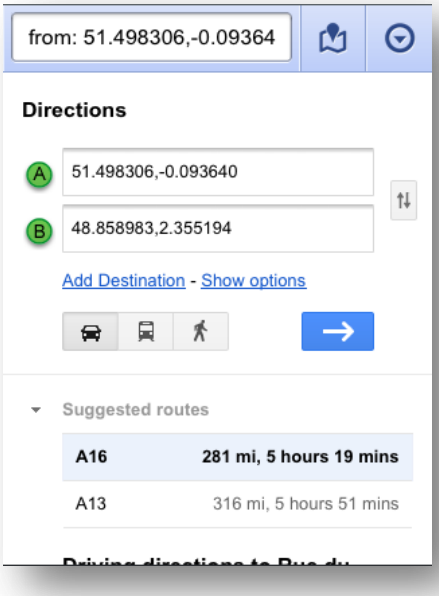
If you set the value of the `drawRouteToNearestItemAddress` property to `false`, the map displays the user's current location at the center of the screen and zooms with the value that is specified in the `regionRadius` property:

```
        mapView.regionRadius = 10000;
```

By default, the `regionRadius` property is set in meters and is set to `0`. If you leave it set to `0`, the map resizes to the maximum resolution and the map of the whole world is displayed.

Note
If the user does not allow the system to use his current location, the map resizes to show all the addresses at once.

If you tap one of the pins on the map, you can see the address tooltip. Tap **Go to Google Maps** and you can work with the address in Google Maps application.



Note
The Sitecore Mobile SDK uses the Google Maps API to draw routes on the map. However, the Google Maps API has a limitation, see the *Google Directions API* for details. You can draw your own overlays and place marks on the map using the `standart MapKit` framework methods**.**

4.10.2 The Map Navigation JavaScript API

In the Sitecore Mobile SDK, you can use JavaScript functions on a website page to open a map.

To use the Map Navigation JavaScript API:

1. Create a map object:

```
var maps_ = new scmobile.google_maps.GoogleMaps();
```

2. In the address, set the following fields:
   - `icon` — the image that is displayed in the address tooltip
   - `title` — the address title that is displayed in the address tooltip
   - `street` — the street name that is displayed in the address tooltip.
   - `city` — the city name that is displayed in the address tooltip.
   - `State` — the state name that is displayed in the address tooltip.
   - `Zip` — the zip code that is displayed in the address tooltip.
   - `Country` — the country name that is displayed in the address tooltip.

Example:

```
var maps_ = new scmobile.google_maps.GoogleMaps();
maps_.drawRoute = true;
maps_.regionRadius = 100000;
try {
    var address_ = {};
    address_.icon = image URL;
    address_.title = "Address 1";
    address_.street = "45 Avenue Jean Zay";
    address_.city = "Paris";
    address_.state = "Livry-Gargan";
    address_.zip = "93190";
    address_.country = "France";
    maps_.addresses.push(address_);
}
maps_.show();
```

You can also set the `drawRoute` and `regionRadius` properties. For more information, see the section *The Map Navigation Objective-C API*.

4.10.3 The Sitecore Mobile Map Component

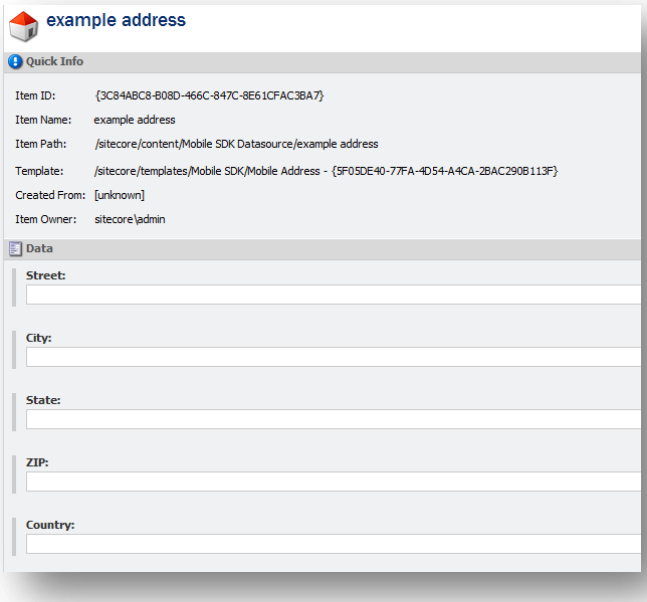
You can use the Sitecore *Mobile Map* component to show the map feature on your website.

To use the Sitecore *Mobile Map* component:

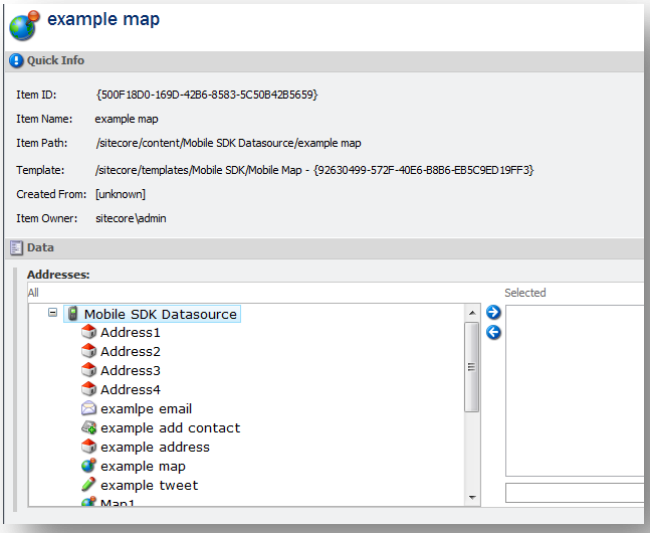
1. Install the Mobile SDK Server Package component.

   For more information, see the section *Installing the Server Side Components*.

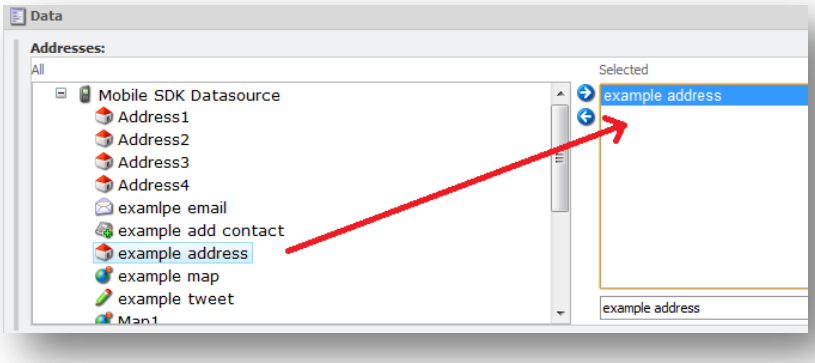
2. Create an item based on the `Mobile Address` template `/sitecore/templates/SCMobile/Mobile Address`.



3. Assign values to the following fields:
   - Street — Single-Line Text
   - City — Single-Line Text
   - State — Single-Line Text
   - ZIP — Single-Line Text
   - Country — Single-Line Text
   - Icon — Image
4. Create an item based on the *Mobile Map* template `/sitecore/templates/SCMobile/Mobile Map`.

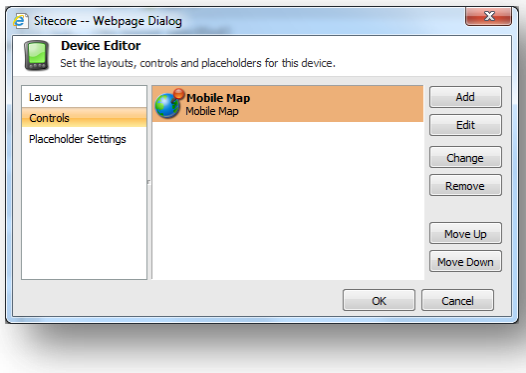


5. Add the *Address* item to the **Addresses** in the *Map* item.

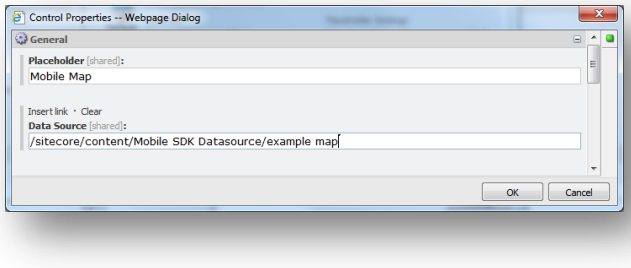


6. You can also assign values to the following fields:
   - Icon — Image
   - Region Radius — Single-Line Text
   - Show Route — Checkbox

   For more information about these parameters, see the section *The Map Navigation Objective-C* API.
7. In the Template Manager or the **Content Editor**, edit the standard values item or the individual item.
8. Click the Presentation tab.
9. On the Presentation tab, in the Layout group, click the Details command. The **Layout Details** dialog appears.
10. In the Layout Details dialog, below the device for which you want to configure layout details, click **Edit**. The **Device Editor** appears.
11. In the **Device Editor**, click the Controls tab, and then click **Add**. The **Select a Rendering** dialog appears.
12. In the **Select a Rendering** dialog, select the *Mobile Map* Contact rendering.
13. In the **Select a Rendering** dialog, click the *Mobile Map* rendering, enter the placeholder path, and then click **Select**.
14. In the **Device Editor**, in the Controls tab, select *Mobile Map*, and click **Edit**



15. Set the created item as a data source for the *Mobile Map* component.



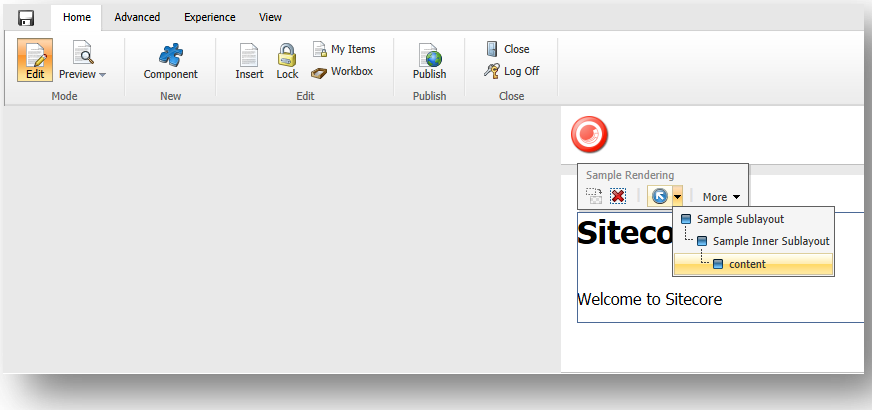
16. Publish your website

17. Run the application in an iOS device, tap **Map** and the **Map** window with the predefined addresses and route parameters is displayed.

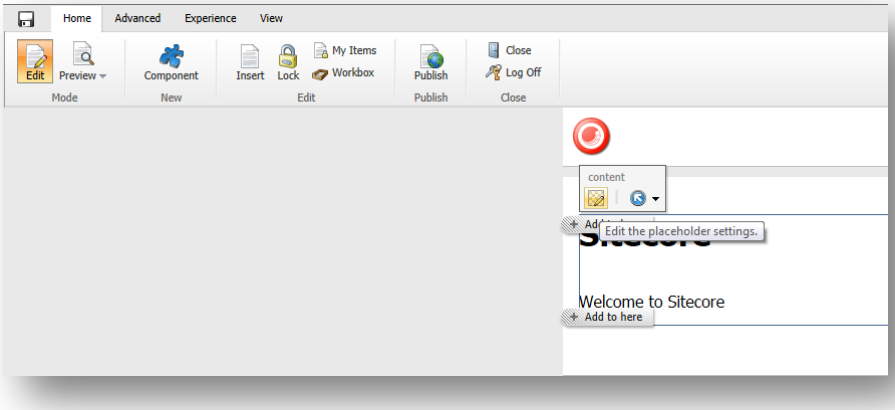
4.10.4 Adding the Mobile Map Component in the Page Editor

To add the *Mobile Map* Component in the **Page Editor**:

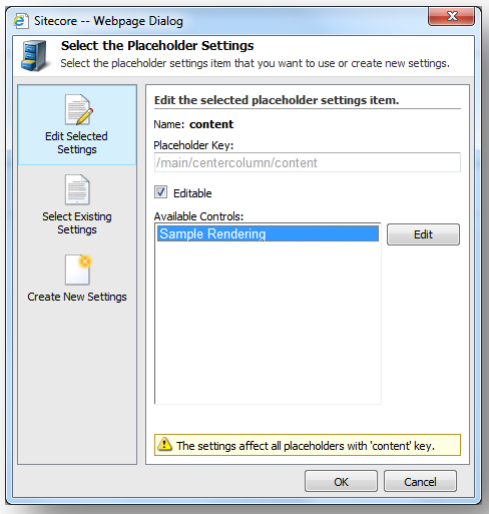
1. Open the **Page Editor**.
2. Navigate to the page that you want to add the *Mobile Map* component to.
3. Select the placeholder that you want to add the *Mobile Map* component to and the floating toolbar appears.
4. In the floating toolbar, click the **Show ancestors** drop-down arrow and then click **content**.



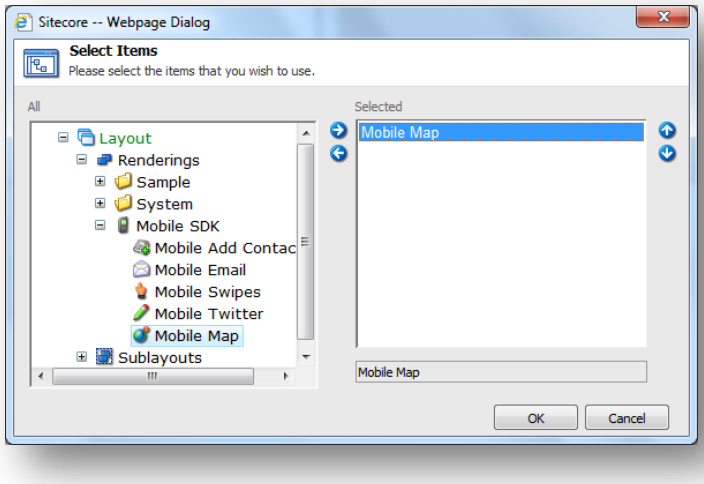
5. Edit the item place holder settings of **Content**.



6. Click **Edit**.

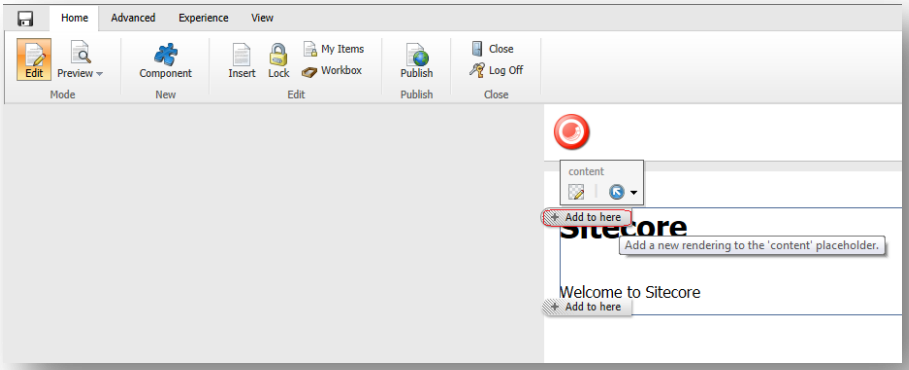


7. Add the *Mobile Map* component to the available controls.

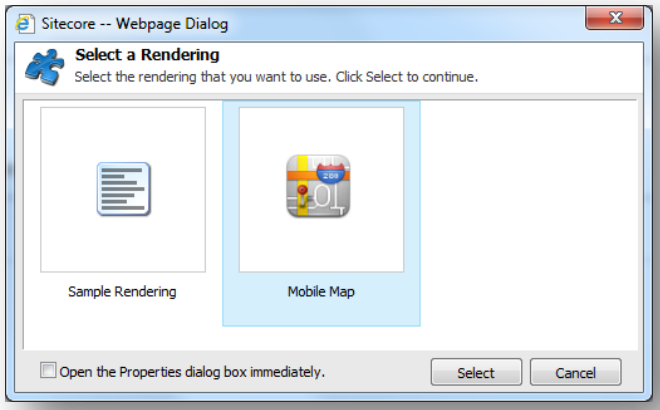


8. Select the placeholder and click **OK** to add the component to the placeholder.

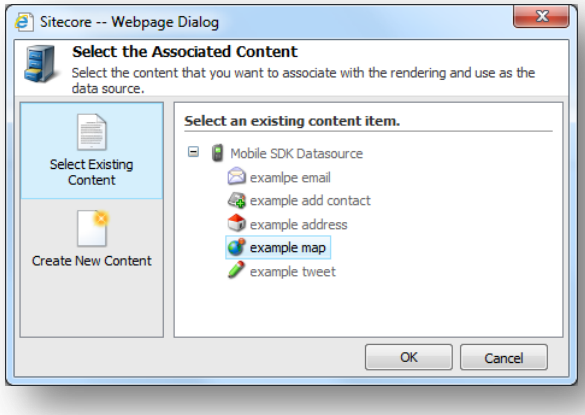
9. Add a new rendering to **Content** placeholder.



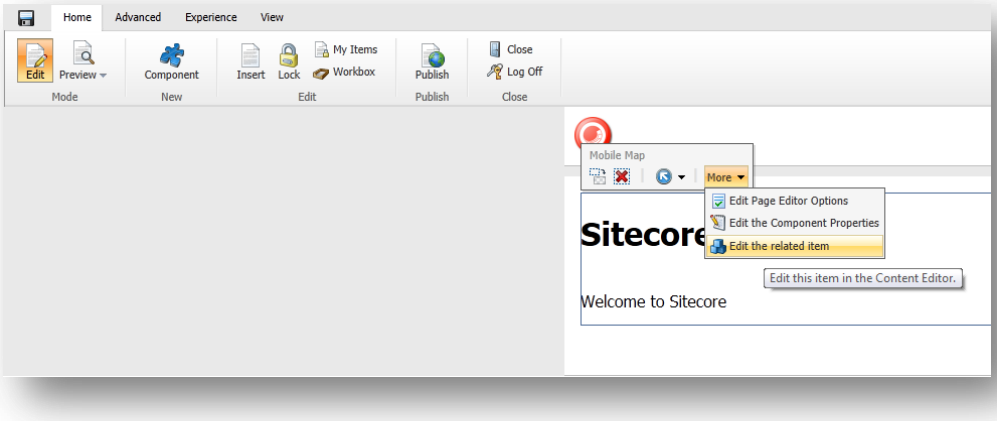
10. Choose the **Mobile Map** rendering component and click **Select**.



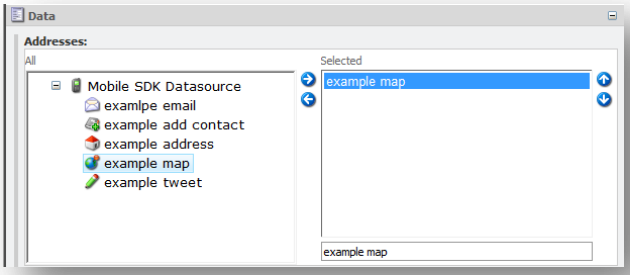
11. You can choose *example map* or create a new data source item.



12. You can the fields of the data source item in the **Content Editor**.



13. Choose the data source that you want to edit.



14. Publish your website.

Chapter 5

Using the API to Manipulate an Item

This chapter describes the most frequently used features in the Sitecore Mobile SDK. Each section in this chapter describes a feature, its relevant classes, and how to use it.

This chapter contains the following sections:

5.1 Content API Requirements

Before you can use the API to manipulate the content, you must:

- Add the Sitecore Mobile SDK framework to the Xcode project.

  For more information, see the section. *Installing the Sitecore Mobile SDK in a Project*.
- Initiate an anonymous or authenticated session.
- Understand the recommendations and best practices, such as how to store the objects that you need to access the API once the session is established and whether the API is thread safe or not.

The following sections describe these requirements.

5.1.1 Establishing an Anonymous or Authenticated Session on the Website

To establish an anonymous session on the web service, create an instance from the `SCApiContext` class using the `[SCApiContext contextWithHost:]` method.

Example:

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

To establish an authenticated session on the web service, create an instance from the `SCApiContext` class using the `[SCApiContext contextWithHost:login:password:]` method.

Example:

```
SCApiContext *context = [SCApiContext contextWithHost:
@"mobilesdk.sc-demo.net/-/item" login: @"domain\\test" password: @"****"];
```

The `mobilesdk.sc-demo.net/-/item` parameter is the host of the Sitecore website and the Item Web API extension of the path is `/item`. This method returns the existing `SCApiContext` object for a given host or creates a new one if it does not exist.

You must enter the domain name and the user name — domainname\\username

You do not have to manually store the new `SCApiContext` object in your application. All the `SCItem` and `SCField` objects own their `SCApiContext` instances. You can use the `[SCItem apiContext]` and `[SCField apiContext]` properties or call the `[SCApiContext contextWithHost:]` method to access the `SCApiContext` instances.

Note
`SCApiContext` is not a singleton object. If you don't have any `SCItem`, `SCField`, and `SCReader` objects that retain the `SCApiContext` object, this object is removed from memory.

After creating the `SCApiContext` object, you can use the `SCApiContext` methods to access the required items and fields as an anonymous user, such as `[SCApiContext itemsReaderWithRequest]`.

5.1.2 Recommendations and Best Practices

This section describes the recommendations and best practices that you should consider when you work with the API.

API Thread Safety

The API is not thread safe. You must include all the API calls in one thread, preferably the main thread.

Accessing the Sitecore Content

All the content API methods, that are used to load data from the backend, return a block of `SCAsyncOp` type. The definitions of this type are in the `SCAsyncOpDefinitions.h` file:

```
typedef void (^SCAsyncOpResult)(id result, NSError *error);
typedef void (^SCAsyncOp)(SCAsyncOpResult handler);
```

This block gives you the content asynchronously. This means that you do not need to worry about threads. To load the Sitecore content, use the `SCAsyncOpResult` callback to call this block. The `SCAsyncOp` block owns the corresponding context while loading data from the backend. This means that the corresponding context is not disposed while it is used.

To load the `Checkbox` field's items for the `/sitecore/content/Nicam` item that is hosted on `mobilesdk.sc-demo.net`:

1. Create the `SCAsyncOp` block:

```
SCApiContext* context = [SCApiContext contextWithHost: @"mobilesdk.sc-demo.net/-/item"
login: @"domain\\username"  password: @"password"];
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];
```

2. Call the `SCAsyncOp` block to load the items:

```
asyncOp(^(id result, NSError* error)
{
    SCItem* item = [result lastObject];
    SCChecklistField* field = (SCChecklistField*)[item fieldWithName:
    @"CheckListField"];
    NSLog(@"items1: %@", field.fieldValue);
});
```

The `SCApiContext` and `SCItemsReaderRequest` objects exist in your application as long as they are used by the operations. You should not worry about their lifetime.

The following section contains more details.

To retrieve content from the website, add this site in the `Sitecore.ItemWebApi.config` file:

```
<sites>
<!--Other site elements could be here.-->
  <site name="website">
    <patch:attribute name="itemwebapi.mode">AdvancedSecurity</patch:attribute>
    <patch:attribute name="itemwebapi.access">ReadWrite</patch:attribute>
    <patch:attribute name="itemwebapi.allowanonymousaccess">true</patch:attribute>
  </site>
</sites>
```

Storing the Loaded Items

The lifetimes of `SCItem` and `SCField` are decoupled from `SCApiContext`. They should be disposed as soon as you stop using them. Otherwise, your application will be subject to memory leaks or will even be stopped by iOS. You must also know that `SCItem` and `SCField` are owned by their parent objects. For example, the `SCItem` objects owns the `[SCItem allChildren]` items.

According to these ownership rules:

- The `SCApiContext` objects and the `SCAyncOp` blocks do not consume large amounts of memory. You can use any amount of `SCApiContext` objects and `SCAyncOp` blocks without destroying them during the whole application lifetime. However, you are limited with the RAM of your iDevice.

- `SCItem` objects contain some user data. That is why they require more memory and you should not load too many of them. In the memory, you must only keep the items that you want to display. You can save the other items in a persistent storage device — for example, on the flash disk of your iDevice.

The following example illustrates these suggestions:

```
//Creating the loader and showing  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,strong) NSArray* items;
@end

@implementation MyViewController

- (void)updateUIWithItems:(NSArray *)items
{
    self.items = items;
    //Displaying items here
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    SCAsyncOp asyncOp = self.asyncOp;
    if (!asyncOp)
        return;

    //Creating weak pointer to avoid retains of the MyViewController's instance
    __weak MyViewController* weakSelf = self;
    asyncOp(^(id result, NSError *error)
    {
        if (asyncOp == weakSelf.asyncOp && weakSelf.isViewLoaded)
            [weakSelf updateUIWithItems: result];
    } );
}

- (void)viewDidUnload
```

```
{
    //Releasing the items
    self.items = nil;

    [super viewDidUnload];
}
```

5.1.3 Accessing an Item

You can use the `[SCApiContext itemReaderForItemPath:]` method to read an item using its path and the `[SCApiContext itemReaderForItemId:]` method to read an item using its ID.

Example:

```
//Reading the item using its path
SCApiContext *context = [SCApiContext contextWithHost:
@"mobilesdk.sc-demo.net/-/webapi"];
[context itemReaderForItemPath: @"/sitecore/content/nicam"](^(id result, NSError
*error)
{
    SCItem* item = result;
    NSLog(@"item display name: %@", item.displayName);
} );
//Reading the item using its ID
SCApiContext *context = [SCApiContext contextWithHost:
@"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 `SCAsyncOp` Objective-C blocks that use the `SCAsyncOpResult` block handler to get an item.

A more general approach to get the same result is to use the `[SCApiContext itemsReaderWithRequest:]` method.

Example:

```
//Reading the item using its path
SCApiContext *context = [SCApiContext contextWithHost:
@"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 );
} );

//Reading it item using its ID
SCApiContext *context = [SCApiContext contextWithHost:
@"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 has already been read from the backend and still exists in the memory, you can access it by invoking:

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

Or

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

Note
The path to the Sitecore item is not case sensitive.

5.1.4 Accessing the Item Properties

You can use the `SCItem` class to get the properties of a Sitecore item:

- `[SCItem displayName]` to get the *DisplayName* property of the item.
- `[SCItem itemId]` to get the *ID* property of the item.
- `[SCItem longID]` to get the item's *LongID* property.
- `[SCItem path]` to get the item's *Path* property.
- `[SCItem itemTemplate]` to get the item's *Template* property.

For example, to show the display name of the item as output in the console:

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

The `SCItem` object always contain its properties. Unlike the fields of an item, you should not call any additional download functions to read the properties.

To access the item's field, use the `[SCItem fieldsReaderForFieldsNames:]` method.

The following example illustrates how to read the Nicam item's `Phone` and `Title` fields:

```
NSSet *fieldsNames = [NSSet setWithObjects: @"Menu title", nil];
[item fieldsReaderForFieldsNames: fieldsNames](^(id result, NSError *error)
{
   NSDictionary *fields = result;
   SCField *phoneField = [fields objectForKey: @"Menu title"];
   NSLog(@"Menu title field raw value: %@", phoneField.rawValue);
});
```

The following example illustrates how to read the item and its fields:

```
SCItemsReaderRequest *request = [SCItemsReaderRequest new];
request.request = @"/sitecore/content/nicam";
request.requestType = SCItemReaderRequestItemPath;
request.scope = SCItemReaderSelfScope;
request.fieldNames = [NSSet setWithObjects: @"Menu title", nil];
[context itemsReaderWithRequest: request](^(id result, NSError *error)
{
   SCItem* item = [result lastObject];
   NSLog(@"Menu title field raw value: %@", [item fieldValueWithName: @"Menu title"]);
```

```
});
```

5.1.5 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]);
} );
```

You can read the item's children using the item's ID or path, if you have not already read the item.

Example:

```
//Reading the children of an item using the item's path: /sitecore/content/nicam
SCApiContext* context = [SCApiContext contextWithHost:
@"mobilesdk.sc-demo.net/-/webapi"];
[context childrenReaderWithItemPath: @"/sitecore/content/nicam"](^(id result, NSError
*error)
{
    NSArray* children = result;
    NSLog(@"children count: %d", [children count]);
});

//Reading the children of an item using the item's ID
SCApiContext *context = [SCApiContext contextWithHost:
@"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 approach to get the same result is to use the `[SCApiContext itemsReaderWithRequest:]` method.

Example:

```
//Reading the children of an item using its path: /sitecore/content/nicam
SCApiContext* context = [SCApiContext contextWithHost:
@"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]);
});

//Reading the children of an item using the item's 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 = SCItemReaderChildrenScope;
request.fieldNames = [NSSet set];
[context itemsReaderWithRequest: request](^(id result, NSError *error)
```

```
        {
                NSArray* children = result;
                NSLog(@"children count: %d", [children count]);
        });
```

If the children has already been received from the backend and still exist in memory, you can access them using the `[SCItem allChildren]` method.

Example:

```
        NSArray* children = [item allChildren];
        NSLog(@"children count: %d", [children count]);
```

5.1.6 Populating the Tab Bar with the Children of an Item

To populate the tab bar with the children of an item using the Xcode Web View controls:

1. Use the Mobile SDK API to create a tab bar menu with titles and icons that come from the corresponding Sitecore items.
2. Create a simple Xcode Single View Application project.
3. Install the Sitecore Mobile SDK framework.

   For more information, see the section *Installing the Sitecore Mobile SDK in a Project*.
4. Add a tab bar controller to the project.
5. Assign the identifier to the **Seque** from **UIView Controller** to the **Tab Bar Controller**.

6. In the `ViewController.h` file, include the Sitecore Mobile SDK framework:

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

7. In the implementation of the **View Controller**, add the following code snippet.

```
        - (void)viewDidAppear:(BOOL)animated
        {
            [super viewDidAppear:animated];
            NSMutableArray *listOfViewControllers = [NSMutableArray new];
            SCApiContext *session = [SCApiContext contextWithHost:
            @"mobilesdk.sc-demo.net/-/item"];
```

```
            NSSet* fieldNames = [NSSet setWithObjects: @"Menu title", @"Tab Icon", nil];
            SCItemsReaderRequest* request = [SCItemsReaderRequest requestWithItemPath:
            @"/sitecore/content/Nicam/"
            fieldsNames: fieldNames];
            request.flags = SCItemReaderRequestReadFieldsValues; //to read the field values
            request.scope = SCItemReaderChildrenScope; //to read the 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];
            } );
        }
```

8. Build and run the application.

5.2 Accessing the Fields of an Item

If you want to read the `Phone` and the `Title` fields of the `SCItems` object, you must use the `[SCItem fieldsReaderForFieldsNames:]` method.

Example:

```
NSSet *fieldsNames = [NSSet setWithObjects: @"Menu title", nil];
[item fieldsReaderForFieldsNames: fieldsNames](^(id result, NSError *error)
{
    NSDictionary *fields = result;
    SCField *field_ = [fields objectForKey: @"Menu title"];
    NSLog(@"field raw value: %@", field_.rawValue);
});
```

The following example illustrates how to read the fields of the item, if you have not already read the item:

```
//Reading the Nicam item and its fields: Menu Title
SCApiContext *context = [SCApiContext contextWithHost:
@"mobilesdk.sc-demo.net/-/item"];
SCItemsReaderRequest *request = [SCItemsReaderRequest new];
request.request = @"/sitecore/content/nicam";
request.requestType = SCItemReaderRequestItemPath;
request.scope = SCItemReaderSelfScope;
request.fieldNames = [NSSet setWithObjects: @"Menu title", nil];
[context itemsReaderWithRequest: request](^(id result, NSError *error)
{
    SCItem* item = [result lastObject];
    NSLog(@"Menu title field raw value: %@", [item fieldValueWithName: @"Menu
    title"]);
});
```

If you want to read all of the item's fields, pass `nil` to the `fieldsNames` attribute.

To access the fields of an item that has already been loaded, call the `[SCItem readFieldsByName]` method. This method returns the `NSDictionary` objects with the `SCField` objects and the `[SCField name]` keys. It returns `nil` if no field has been read for the given item.

The `SCField` class represents a field of the Sitecore system item. You can use it to get the field's properties such as `ID`, `Name`, `Type` and `RawValue`.
To retrieve the item's fields for an extranet user, you must set the permission to *Allow* for the *Language Read* security of these fields or turn off the `CheckSecurityOnLanguages` property in the `Web.config` file:

```
<setting name="CheckSecurityOnLanguages" value="false" />
```

5.3 Accessing the Different Types of Fields

The Mobile content API contains a set of special classes that are designed to make it easier for you to work with the simple and list types of the Sitecore fields.

The following table lists the field types and the corresponding content API classes:

Field Type	API Class
Checkbox	`SCChecklistField`
Color	`SCColorPickerField`
Date	`SCDateField`
Datetime	`SCDateTimeField`
Image	`SCImageField`
Checklist	`SCChecklistField`
Multilist	`SCMultilistField`
Treelist	`SCTreelistField`

Note
All the Sitecore field types that are not in the table are represented by the `SCField` class. The `fieldValue` property of each field type contains the raw value of the field.

5.3.1 The Image Field

If the field is of type *Image*, the corresponding class of the field is `SCImageField`. The instance of this class has an additional property: `[SCImageField imagePath]` that contains the path to the image — the Sitecore media item. To read an image, use the `[SCApiContext imageLoaderForSCImagePath:]` method or the `[SCImageField fieldValueReader]` method.

To load the `UIImage` object for your `SCImageField` class, call the `[SCImageField fieldValueReader]` method.

Example:

```
SCImageField* field = [item fieldWithName: @"Tab Icon"];
[field fieldValueReader](^(id result, NSError *error)
{
    UIImage* image = result;
    NSLog(@"image size: %@", NSStringFromCGSize(image.size));
} );
```

We recommend that you:

- Read the images with the fields:

```
[item fieldValueReaderForFieldName: @"Tab Icon"](^(id result, NSError *error)
{
    UIImage* image = result;
    NSLog(@"image size: %@", NSStringFromCGSize(image.size));
});
```

- Read the fields with the values when reading the item:

```
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 lastObject];
    UIImage* image = [item fieldValueWithName: @"Tab Icon"];
    NSLog( @"image size: %@", NSStringFromCGSize(image.size));
});
```

5.3.2 The Checkbox Field

The `SCCheckboxField` class represents the Sitecore *Checkbox* field. The `[SCCheckboxField fieldValue]` property of this class is of type `NSNumber`. The value of the `[[SCCheckboxField fieldValue] boolValue]` is `Yes` if the checkbox is selected and `No` if the checkbox is not selected.

Example:

```
SCField* field = [item fieldWithName: @"CheckBoxField"];
NSLog( @"checkbox value: %@", [[field fieldValue] boolValue] );
```

5.3.3 The Date and Datetime Fields

The `SCDateField` and `SCDateTimeField` classes represent the Sitecore `Date` and `Datetime` fields respectively.

The `fieldValue` property of these classes is of type `NSDate` object and holds the corresponding value of the date.

Example:

```
SCField* field = [item fieldWithName: @"DateField"];
[field fieldValueReader](^(id result, NSError *error)
{
    NSDate* date = result;
    NSLog(@"date value: %@", date );
} );
```

5.3.4 The Color Picker Field

The `SCColorPickerField` class represents the Sitecore `Color Picker` field. The `fieldValue` property of this class is an object of type `UIColor` and holds the corresponding value of the color.

Example:

```
SCField* field = [item fieldWithName: @"ColorPickerField"];
[field fieldValueReader](^(id result, NSError *error)
{
    UIColor* color = result;
    NSLog(@"color value: %@", color);
} );
```

5.3.5 The Checklist, Multilist and Treelist Fields

The `SCChecklistField`, `SCMultilistField` and `SCTreelistField` classes represent the `Checklist`, `Multilist` and `Treelist` Sitecore fields, respectively.

The `fieldValue` property of these classes is an object of type `NSArray` object and holds the list of the items specified with these fields.

In the `rawValue` property, you can find the raw value of the field — the list item's IDs separated by the "`|`" symbol.

By default, the `fieldValue` property is set to `nil`. Therefore, you must use the `[SCField fieldValueReader]` method to load the list of items. To avoid retain cycles, the `SCField` object does not own the read items.

Example:

```
[item fieldValueReaderForFieldName: @"Products List"](^(id result, NSError *error)
{
    NSArray *items = result;
    NSLog(@"items count: %d", [items count]);
});
```

You can find more convenient cases that use the `[SCField fieldValueReader]` method in *The Image Field* section.

5.3.6 The Droplink and Droptree Fields

The `SCDroplinkField` and `SCDroptreeField` classes represent the Sitecore `Droplink` and `Droptree` fields, respectively.

The `SCDroplinkField` and `SCDroptreeField` classes are similar. The `fieldValue` property of these classes contains an object of type `SCItem` that holds the linked item and the `[SCField fieldValueReader]` method to read it.

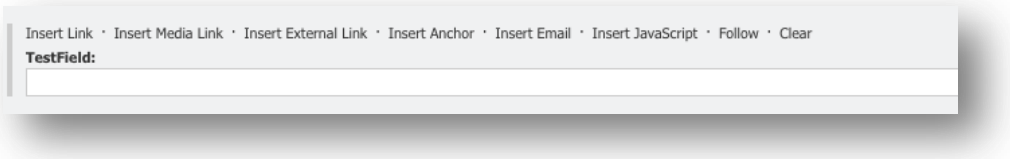
The `rawValue` property contains the linked item's ID.

The following example illustrates how to read the linked item:

```
[field fieldValueReader](^(id result, NSError *error)
{
    SCItem* item = result;
    NSLog(@"linked item: %@", item);
});
```

5.3.7 The General Link Field

The `SCGeneralLinkField` class represents the Sitecore General Link field. The General Link field type can link different entities such as `item`, `media item`, `email` and `external link`.



The `SCGeneralLinkField` class has the `linkData` property that holds the polymorphic linked entity data. Each type of the linked entity has its corresponding class:

Linked Entity Type	Class
Internal link	`SCInternalFieldLinkData`
Media link	`SCMediaFieldLinkData`
External link	`SCExternalFieldLinkData`

Linked Entity Type	Class
Anchor link	`SCAnchorFieldLinkData`
Email link	`SCEmailFieldLinkData`
JavaScript link	`SCJavascriptFieldLinkData`

The properties of these classes contain meta information about the link — `Link Description`, `Link type`, `Url`, and `Alternate Text`.

Example:

```
SCGeneralLinkField* field;
NSLog(@"general link description: %@", field.linkData.linkDescription);
NSLog(@"general link url: %@", field.linkData.url);
```

The `SCInternalFieldLinkData` class has:

- The `itemReader` method to load the linked item.
- The `SCMediaFieldLinkData` class and the `[SCMediaFieldLinkData imageReader]` method to load the linked image.

To find the description of all link types entities and their properties, see the `SCFieldLinkData.h` file of the Mobile SDK framework.

5.4 Accessing the Parent of an Item

To read the parent of an existing item, you can use the `[SCItem itemsReaderWithRequest:]` method with the `SCItemReaderParentScope` scope:

```
//Reading the parent of an item using the item's 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 lastObject];
    NSLog(@"parent display name: %@", parent.displayName);
} );

//Reading the parent of an item using the item's 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 lastObject];
    NSLog(@"parent display name: %@", parent.displayName);
} );
```

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

5.5 Accessing Items Using Sitecore Query

You can use the `[SCItem itemsReaderWithRequest:]` method to retrieve the items that match a Sitecore query from a website. You must then set the value of `[SCItemsReaderRequest requestType]` attribute to the query of type `SCItemReaderRequestQuery`.

Remember that:

- The Sitecore query is not always the most efficient way to locate items in a repository that contains a large amount of data. You must use paging where the system must frequently match the items in a large bunch.
- The Sitecore query syntax is different from the XPath syntax.
- You must not assume that the Sitecore query returns the items in the document order or the 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 following example is the same as the previous one but with paging. Only the
third and the fourth items of the result array are read:
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 page number
request.pageSize = 2; //the page size
[context itemsReaderWithRequest: request ](^(id result, NSError* error)
{
    NSLog(@"result items count: %d", [result count]); //the expected size is 2
});
```

5.6 Reading Paged Items Efficiently

When you read a bulky data objects such as many items in one request, we can have a problem with memory management. You can use the `SCPagedItems` object to load only one paged item using its index.

If you run the query: `/sitecore/content/Nicam/child::*[@@templatename='Site Section'], the` result can contain many items. If you only want to read the third and the fourth items, use the following code:

```
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);
});
```

If you use this class, you do not have to worry about paging index calculation and the cache logic for merging requests when it reads the items on the same page. In the previous example, the page that contains third and fourth items loads only once. However, if you do not need this behavior and want to read a page of items, just specify the `[SCItemsReaderRequest pageSize]` and `[SCItemsReaderRequest page]` properties and call the `[SCApiContext itemsReaderWithRequest:]` method in 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; //it starts indexing from zero.
[context itemsReaderWithRequest: request]( ^(id result, NSError* error)
{
    NSArray* items = result;
    NSLog( @"item count: %d", [items count] );
} );
```

5.7 Accessing the Different Languages of an Item

To set the language, use the `[SCApiContext defaultLanguage]` property. The default value of this property is `"en"`.

To read the items with the language that you specified, use these methods:

- `-[SCApiContext itemWithId:]`
- `-[SCApiContext itemReaderForItemId:]`

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);
});
```

You can also use the `language` property of the request to specify the languages for separate requests without changing the `SCApiContext` default language.

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 lastObject];
    NSLog(@"item: %@", item.displayName);
});
```

To access the list of available languages, call the `[SCApiContext systemLanguagesReader]` method.

Example:

```
SCApiContext *context = [SCApiContext contextWithHost:
@"mobilesdk.sc-demo.net/-/webapi"];
[context systemLanguagesReader](^(NSSet *languages, NSError *error)
{
    NSLog(@"languages: %@", languages);
});
```

5.8 Using the Cache

The Sitecore Mobile SDK Content API reduces the number of requests to the Sitecore web service by:

- Merging the requests
- Reading the cached Items

5.8.1 Merging Requests

If you perform several API requests to load data, while the first request is being executed, only one request to Sitecore web service is handled while all of the `SCAsynOp` block's handlers are being processed.

For example, call the `asyncOp` block twice:

```
asyncOp(^(id result, NSError* error)
{
    //result handler 1
});
asyncOp(^(id result, NSError* error)
{
    //result handler 2
});
```

Here, only one HTTP request is executed. Therefore, you shouldn't create *Merge Requests* logic above the API. The *Merge Requests* feature is useful when different parts of UI try to load the same data at the same time — for example, several `UITable` cells using the same image.

Note
All asynchronous API operations use this solution.

5.8.2 Reading the Cached Items

If the requested items are still in the memory, the `SCAyncOp` block returns the loaded items without disturbing the Sitecore web service. However, you cannot rely on this feature. For example, Sitecore query requests always load the items from the backend.

An item may still be in memory because:

- Your application owns this item.
- This item is a descendant of another item.

An example of how the cache works:

```
SCApiContext* context = [SCApiContext contextWithHost:
@"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 lastObject];
    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 if the block finished loading
```

```
                __block BOOL wasLoadImmediately = NO;
                //Loading the children again
                childrenReader(^(id result, NSError *error)
                {
                    NSLog(@"children Items count 2: %d", [result count]);
                    wasLoadImmediately = (previousCount == [result count]);
                });

                //Checking that the childrenReader block's handler is already called with the
                result
                NSAssert(wasLoadImmediately, @"wasLoadImmediately should be YES here");
                });
        } );
```

In the previous example, `NSAssert` does not fail. The second call of the `childrenReader` method immediately returns the resulted item's children.

The cache is unavailable when:

- In the Sitecore query, the `[SCItemsReaderRequest requestType]` is equal to `SCItemReaderRequestQuery`. In this case, the read items are ignored and the API directly reads the data from a Sitecore web service.
- The item is released and removed from memory. In this case, the API cannot cache the item in the file system.
- The `ignore cache` flag is used. If you set `[SCItemsReaderRequest flag]` to `SCItemReaderRequestIgnoreCache`, the loaded items are also ignored.

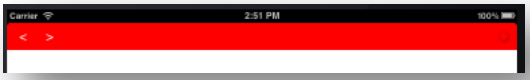
In all other cases the API tries to reuse read items in the requests.

5.9 Modifying the Design of the Web View Navigation Bar

The Sitecore Mobile SDK contains two Web View components:

- `SCWebView`
- `SCWebBrowser`

Both of them have interfaces that are similar to the standard `UIWebView` component. The difference between them is the navigation bar of the `SCWebBrowser` that contains back and forward buttons and an activity indicator:



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

If you want to use the actions toolbar in the Mobile SDK, use the `SCWebBrowser` control. In this case, use the `[SCWebBrowser setCustomToollbarView:]` method to customize the appearance of the toolbar. You can also use the `[SCWebBrowser setCustomToollbarView:]` method if the default navigation bar does not match your application design requirements and you may want to create your own navigation view.

If your custom navigation view conforms to the `SCWebBrowserToolbar` interface, you can use it to customize the Web View Components such as back and forward navigation, the activity indicator, and other browser navigation controls.

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 *Web View*:

```
MyNavigationHeader *view = [[MyNavigationHeader alloc]initWithFrame:CGRectMake(0.f,
0.f, 110.f, 80.f)];
view.autoresizingMask = UIViewAutoresizingFlexibleHeight;
[self.webBrowser setCustomToollbarView: view];
```

5.10 Creating Items Using the Mobile SDK

To use the Mobile SDK API to create an item in the **Content Tree**:

1. Login to the `SCApiContext` as a user who has permission to create items:

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

2. To create the item, use the `[SCApiContext itemCreatorWithRequest:]` method.

Example:

```
-(void)createItem
{
    //Creating the API's context
    SCApiContext *context = [SCApiContext contextWithHost:
    @"mobilesdk.sc-demo.net/-/webapi" login: @"domain\\login" password: @"password"];

    //Setting the database that is used to store the item
    context.defaultDatabase = @"web";

    //The path where the new item is created (as subitem for the given path)
    NSString* itemPath  = @"/sitecore/content/Test Data";
    SCCreateItemRequest *request = [SCCreateItemRequest requestWithItemPath:itemPath];

    //Assigning the new item's name, template and fields values
    request.itemName = @"apiItem test";
    request.itemTemplate = @"Sample/Sample Item";
    request.fieldsRawValuesByName = [[NSDictionary alloc] initWithObjectsAndKeys:
    @"Title 2", @"Title", @"Text 2", @"Text", nil ];

    //Assigning the item's fields that are returned in the callback after the item
    creation
    request.fieldNames = [NSSet setWithObjects: @"Title", @"Text", nil];

    //Creating the item
    [context itemCreatorWithRequest:request](^(SCItem *item, NSError *error)
    {
        NSLog(@"result item: %@", item);
        NSLog(@"items fields: %@", item.readFieldsByName);
    });
}
```

The `[SCApiContext itemCreatorWithRequest:]` method returns the `SCAsyncOp` block which is called to create an item. The `SCAsyncOpResult` handler returns an object of type `SCItem`. if the item is successfully created or of type `SCError` if it is not created.

5.11 Modifying the Item's Fields Using the Mobile SDK

To use the Mobile SDK API to modify the items in the **Content Tree**:

1. Login to the `SCApiContext` as a user who has permission to modify the items:

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

2. Read the item.
3. Change the `[SCField rawValue]` property of its fields.
4. Call the `[SCItem saveItem]` method to save the changes.

Example:

```
-(void)readAndEditExistedItem
{
    //Creating the API's context
    SCApiContext *context = [SCApiContext contextWithHost:
    @"mobilesdk.sc-demo.net/-/webapi" login: @"domain\\login" password: @"password"];

    //Setting the database from which the items are read and edited
    context.defaultDatabase = @"web";

    //The item's path to edit
    NSString* itemPath = @"/sitecore/content/Test Data/test1";
    NSSet* fieldNames = [NSSet setWithObjects: @"Text", nil];
    SCItemsReaderRequest* request = [SCItemsReaderRequest requestWithItemPath:
    itemPath   fieldsNames: fieldNames];
    [context itemsReaderWithRequest:request](^(NSArray *items, NSError *error)
    {
        SCItem *item = [items count] == 0 ? nil : [items lastObject];
        SCField *field = [item.readFieldsByName objectForKey: @"Text"];
        field.rawValue = @"New Text";
        if(item)
        {
            [item saveItem](^(SCItem *editedItem, NSError *error)
            {
                NSLog(@"readFieldsByName: %@", editedItem.readFieldsByName);
            });
        }
    } );
}
```

This example prints the values of the edited fields to the console.

If you want to edit several items at same time, set the `[SCItemReaderRequest scope]` property to `SCItemReaderChildrenScope` or the `[SCItemReaderRequest requestType]` property to `SCItemReaderRequestQuery`.

5.12 Deleting an Item Using the Mobile SDK

To use the Mobile SDK API to delete an item in the **Content Tree**:

1. Login to the `SCApiContext` as a user, who has the appropriate permissions:

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

2. Read this item
3. Delete the item using the `[SCItem removeItem]` method.

Example:

```
-(void)readAndRemoveOneItem
{
    //Creating the API's context
    SCApiContext *context = [SCApiContext contextWithHost:
    @"mobilesdk.sc-demo.net/-/webapi" login: @"domain\\login" password: @"password"];

    //Setting the database from which the item is read and removed
    context.defaultDatabase = @"web";

    //Finding and removing the test1 item
    NSString *itemPath = @"/sitecore/content/Test Data/test1";
    SCItemsReaderRequest *request = [SCItemsReaderRequest
    requestWithItemPath:itemPath];
    [context itemsReaderWithRequest:request](^(NSArray *items, NSError *error)
    {
        SCItem *item = [items count] == 0 ? nil : [items lastObject];
        if(item)
        {
            //Removing the item found
            [item removeItem](^(SCItem *removedItem, NSError *error)
            {
                NSLog(@"removedItem.itemId: %@", removedItem.itemId);
                NSLog(@"removedItem: %@", removedItem);
            });
        }
    });
}
```

4. You can also use the `[SCApiContext removeItemsWithRequest:]` method to remove several items at once.

Example:

```
-(void)removeSeveralItems
{
    //Creating the API's context
    SCApiContext *context = [SCApiContext contextWithHost:
    @"mobilesdk.sc-demo.net/-/webapi" login: @"domain\\login" password: @"password"];

    //Setting the database from which the item is read and removed
    context.defaultDatabase = @"web";

    //Finding and removing all children of the item with path: "/sitecore/Test Data"
      and inherits from the 'Sample Item' template
    NSString *itemPath = @"/sitecore/content/Test Data/child::*[@@templatename='Sample
    Item']";
    SCItemsReaderRequest* request = [SCItemsReaderRequest new];
    request.request = itemPath;
    request.requestType = SCItemReaderRequestQuery;
    [context removeItemsWithRequest:request](^(id response, NSError *error)
    {
```

```
            NSLog(@"deleted items count: %@", response);
        });
    }
```

This example writes a list of the deleted items IDs to the console.

5.13 Uploading Media Files to Sitecore Media Library

To upload media files to the **Media library**, use the `-[SCApiContext mediaItemCreatorWithRequest:]` method:

Example:

```
-(void)createMediaItem
{
    //Creating the API's context
    SCApiContext *context = [SCApiContext contextWithHost:
    @"mobilesdk.sc-demo.net/-/webapi" login: @"domain\\login"  password: @"password"];

    //Setting the database which will used to create media item
    context.defaultDatabase = @"web";

    //Assigning media items fields as item's name, item's template etc.
    SCCreateMediaItemRequest *request = [SCCreateMediaItemRequest new];

    //fileName is presented in "Alt" field.
    request.fileName = @"ScreenShot.png";
    request.itemName = @"mediaItem";
    request.itemTemplate = @"System/Media/Unversioned/Image";

    //Media file to store
    request.mediaItemData = UIImagePNGRepresentation([UIImage
    imageNamed:request.fileName]);
    request.fieldNames = [NSSet new];
    request.contentType  = @"image/png";

    //The folder in which the media item is created - This folder can be empty
    request.folder = @"/Media Folder";
    [context mediaItemCreatorWithRequest:request](^(SCItem *item, NSError *error)
    { NSLog(@"result media item: %@", item); } );
}
```

The `[SCApiContext mediaItemCreatorWithRequest:]` method returns the `SCAsyncOp` block that is called to create the media item. The `SCAsyncOpResult` handler returns an object of type `SCItem` if the item is successfully created or of type `SCError` if the item is not created.

5.13.1 Uploading Media Files in JavaScript

To create media items in JavaScript you must use the `scmobile.contentapi.createMediaItem` method.

To create media item, you must specify the following obligatory parameters:

- `imageUrl` — the URL of the image you want to download.
- `login` — Sitecore user's login which is permitted to create media items
- `password` — Sitecore user's password
- `itemName` — the name of the media Item
- `path` — the path of the media Item in the Media Library
- `database` — the database name on which you want to upload the media.
- `fields` — the fields of the media item

Example:

```
var createItemInfo = {};
createItemInfo.imageUrl = 'http://allimages.com/best picture.jpg';
createItemInfo.login = 'login';
createItemInfo.password = 'password';

//The database used to create media item, default is web
createItemInfo.database = 'web';

//The relative path in "Media Library" folder, can be empty.
createItemInfo.path = 'images';
createItemInfo.itemName = 'media Item name';

//Compression quality used at compressing image into jpg format, default and max is 1.
createItemInfo.compressionQuality = .5;

//Media items fields raw values by names
createItemInfo.fields = {Alt: 'some alt'};

function onSuccess()
{
    scmobile.console.log('media item was created');
}

function onError(msg)
{
    scmobile.console.log('create media item error: ' + msg);
}

//Creating the media item with a set of fields
scmobile.contentapi.createMediaItem(createItemInfo, onSuccess, onError);
```

5.14 Getting the HTML Markup of Isolated Rendering Using Mobile SDK

To get the string of the rendered HTML, use the `[SCApiContext renderingHTMLLoaderForRenderingWithId:sourceId:]` method.

This method returns the `SCAsyncOp` loader that you can use to get the HTML that is rendered as an object of type `NSString`.

Example:

```
SCApiContext* apiContext = [SCApiContext contextWithHost:
@"mobilesdk.sc-demo.net/-/item"];
SCAsyncOp loader = [apiContext renderingHTMLLoaderForRenderingWithId: @"{E036028E-
                    9CB4-4B41-A945-E4D6FF4FA549}"
                    sourceId: @"{55672F84-181A-4B04-A87F-AEF9A2612179}"];
loader(^(id result, NSError * error)
{
    if ( nil != result_ )
    {
        NSLog(@"result %@",  result_ );
    }
    else
    {
        NSLog(@"error %@",  error_.localizedDescription );
    }
});
```

If an error occurs, this example returns an HTML markup or a string that contains an error message.