# **Introduction to iPhone Programming**

## using Dashcode and Xcode







All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of the author.

Trademarked names may appear in this document. Rather than use a trademark symbol with every occurrence of a trademarked name, the names are used only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this document is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this document, the author shall not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this document.

## About the Author



Wei-Meng Lee (**weimenglee@learn2develop.net**) is a technologist and founder of Developer Learning Solutions (<u>http://www.learn2develop.net</u>), a technology company specializing in hands-on training on the latest Microsoft and Mac OS X technologies. He is also an established author with Wrox and O'Reilly.

Wei-Meng first started the iPhone programming course in Singapore and it has since received many positive feedbacks. His hands-on approach to iPhone programming makes understanding the subject much easier than reading books,

tutorials, and documentations from Apple. Wei-Meng is currently working with Wrox on "*Beginning iPhone SDK Programming with Objective-C*", due to be published in Jan 2010.

## Lab 1 – Getting Started with Dashcode

In this lab, you will learn how to get started with developing Web applications for the iPhone. Using the Dashcode tool (part of the iPhone SDK), you will create a currency convertor application that makes use of a client side database to store currency rates.

1. Launch Dashcode and create a new **Custom** project. Notice that by default, Dashcode has created a **content** and a **footer** parts for you. Parts are the various views that you see on your web applications, such as buttons, text, etc. For this lab, you will create a simple currency convertor web application for the iPhone.

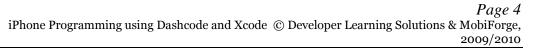
Run	Pause View		-
	content		-
Ŀ	l footer		
×	Application Attributes		
	Web Clip Icon		-
	Share	16.	100
			-

- 2. Select each of these parts and press the delete key. You shall delete these two parts and add your own parts manually.
- 3. Using the Library (Window→Show Library), drag-and-drop a Stack Layout part to the design surface:

$\odot \bigcirc \bigcirc$	Untitled 3	C	$\supset$	
	🚺 🚺 🔍 Sea			
Run Pause View	Inspector Library 1200 - 1200 - 13000 - 1300 - 1300 - 13000 - 1300 - 1	Search		
Vntitled		00	Library	
► Hight stackLayout		Pi	arts Code Pl	notos
Application Attributes	1	🔮 Parts Libr	ary	\$
Application Attributes			_	- n
Application Attributes Web Clip Icon		Push Button	Left Rounded	Right Rounde
Share 🔊	·			
		Vertical Level	Bottom Recta	Ellipse Shape
Workflow Steps		Rectangle Shape	Rounded Rec	Top Rectangl
Place an item from the library on the canvas, then change its properties with the inspector.			Text Mag 3 Dec 3 Units 3	
		Box	Browser	Column Layout
C Library C Canvas		Miss 3 Tim 3 Orient 3 Trin 3		Mix > Origin > Origin > Garde >
7 Mark as Done		Edge-to-Edg	Rounded Box	Rounded Rec
Add handlers & code Required		<b>\$</b>		Q
♦- :=	E >E	Stack Layout	Canvas	Search Field
				Text
		Text Area	Text Field	Text 🕴
		Stack	Layout	
		views. Use the in	s an area that cor nspector to add a t the transitions	tains swappable nd remove between them.
		* Q	Filter	

Expand the stackLayout part and you should see that it contains two subviews - view1 and view2. Select view1 and change its size to 320px by 356px via the Inspector window (Window→Show Inspector). Do the same for view2.

000	CurrencyConvertor	$\bigcirc$
▶,		() (I) Q Search
Run Pause View		Inspector Library Search
CurrencyConvertor		1 1 1 200 1 1 1 1 1 300 1 1 1 1 400 1 1 1
🔻 🖬 stackLayout		
📰 view1 🔤		
view2	O O O Metrics (view1)	2P
Application Attributes	Layout Document Flow	
Web Clip Icon	Size 320px († 356px (†	
Share 🔊	Width Height	
	Margin 0px 🗘 0px 🗘	
	Top Left	
Workflow Steps	0px 🗘 0px 🗘	
Lay out interface	Bottom Right	
Place an item from the library	Autoresize	
on the canvas, then change its properties with the inspector.	~	
	S.	
Library		
Canvas Inspector		
		·
(?) Mark as Done	Constraints none (* none *	•
Add handlers & code Required	Max Width Max Height	5
	0px 🗘 356px 🗘	9
¢- :∃ 🗹 🛛 🛛 🖓	Min Width Min Height	1.



5. Double-click on **view1** and rename it as **mainScreen**. Do the same for **view2** and rename it as **settings**:



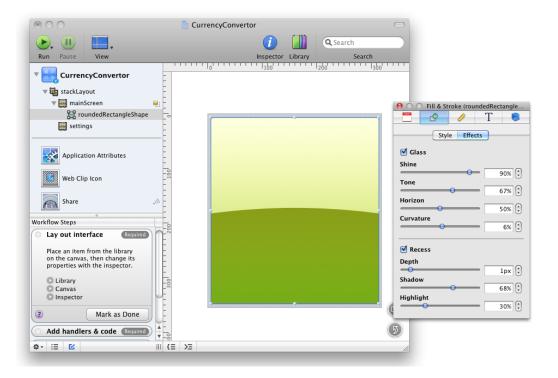
6. In the Library, drag-and-drop the Rounded Rectangle Shape part onto the mainScreen view:

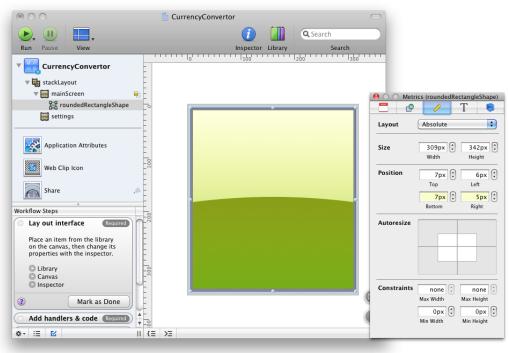
		CO C Library
00	CurrencyConvertor	Parts Code Photos
	G Search	🔮 Parts Library 🛟
Run Pause View	Inspector Library Search	
CurrencyConvertor	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Push Button Left Rounded Right Rounde
v 🖏 stackLayout		
🔻 🧮 mainScreen 📃 🗧		
🔡 roundedRectangleShape 🔤		Vertical Level Bottom Recta Ellipse Shape
settings		
Application Attributes		Rectangle Shape Rounded Rec Top Rectangl
Web Clip Icon		
Share 🔊		Box Browser Column Layout
Workflow Steps	i i l	Intrift         2           1984         2           1984         2           1984         2           1984         2           1984         2           1984         2           1984         2           1984         2
Lay out interface		Edge-to-Edg Rounded Box Rounded Rec
Place an item from the library on the canvas, then change its properties with the inspector.		
C Library		Stack Layout Canvas Search Field
Canvas Inspector		Text
(? Mark as Done		Text Area Text Field Text
Add handlers & code Required	5	Rounded Rectangle Shape
		Rounded Rectangle Shape
		A shape is an area with a custom fill and stroke. The rounded rectangle shape features four rounded corners. Adjust the roundness of the corners and the fill and stroke in the Fill & Stroke inspector.
		C Filter

7. In its **Inspector window**, select the **Fill & Stroke** tab and in the **Style** tab select **Gradient** fill and select two colors:

$\odot \odot \odot$	CurrencyConvertor	
▶	() A Search	
Run Pause View	Inspector Library Search	
CurrencyConvertor	10, 10, 10, 10, 10, 10, 10, 10, 10, 10,	
v 🖬 stackLayout		
🔻 🧱 mainScreen 📃		😝 🔿 🔿 Fill & Stroke (roundedRectangle
🔡 roundedRectangleShape	¢ • • •	T 8
settings –		
Application Attributes		Style Effects
Share M		
Lay out interface Required Place an item from the library on the canvas, then change its properties with the inspector.		Corner Roundness Opacity 00% (;)
Canvas Inspector		Stroke Solid : Spx :
Add handlers & code (Required)		5
** := 🗳	= /=	11

8. Select the Effects tab and check the Glass and Recess checkboxes:



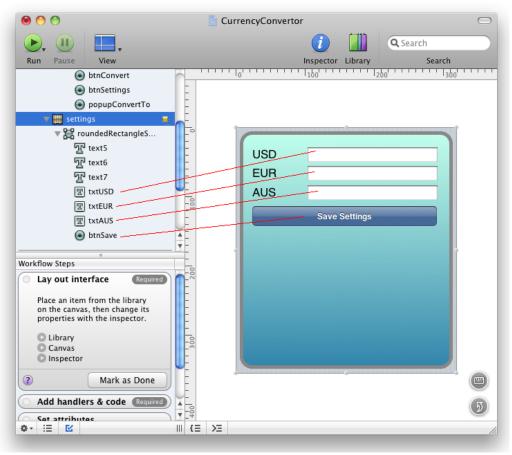


9. Select the Metrics tab and select the Absolute layout:

- 10. Add the following parts to the **Rounded Rectangle Shape** part and name them as shown:
  - Text
  - TextField
  - Pop-up Menu
  - Push Button

▶, Ⅲ ■,	G Search
	Search
Run Pause View	Inspector Library Search
CurrencyConvertor	10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
▼ 🙀 stackLayout	
🔻 🧱 mainScreen 📃 🖡	
🔻 🧮 roundedRectangleS	· · · · · · · · · · · · · · · · · · ·
T text	Enter Amount
T text4	
T txtAmount	Convert to Item 1 \$
btnConvert	
btnSettings	Convert!
popupConvertTo	
v i settings	
Workflow Steps	
Lay out interface Required Place an item from the library on the canvas, then change its properties with the inspector.     Library     Canvas     Inspector     Mark as Done	Settings
	<b>(</b>
Add handlers & code Required	
Set attributes	
\$- :∃ 🗹 III (∃ >	

11. Select the **settings** subview and repeat the same steps you have performed above. The following figure shows the parts added to the **settings** subview.



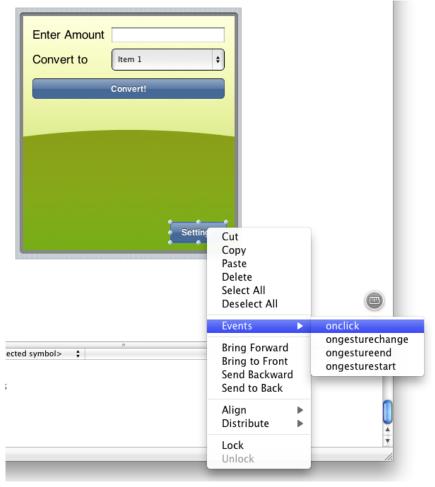
12. You are now ready to view the application on the iPhone Simulator. Press Command-r to view the application on the iPhone Simulator. Notice that the application is displayed by mobile Safari on the iPhone.

		-	-
Il Carrier 🔶	5:46 PM		
	encyConve	rtor	
file:///var/folders/l	5/151 🖒	Goog	le
Enter Amount	t t		
Convert to	Item 1		
Convent to	Liem		
	Convert!	_	
		Set	tings
	+	Ĥ	G

Notice that you can only see the **mainScreen** subview. To see the **settings** subview, you need to write some code to navigate to it from the **mainScreen** subview.

## **Coding the Application**

1. So you are now ready to write some code. With the **mainScreen** subview selected, right-click on the **Settings** button and select **Events**-**>onclick**:



2. You will be asked to name the event handler for this event. Name it as shown below:



	Inspector Library Search	
-100	100 100 100 100 100 100 100 100 100 100	500
<b>M</b> -		
-		
, –°		
	Enter Amount	
	Convert to Item 1 +	
-8	Convert!	
<b>M</b> -		
F		
-		
2001 1 1 2001		
E		
-		
<b>A</b>		
<b>T</b> -		
-00	Settings	
d -		
		(5)
Imain.js	lickHandler 🛊	
8 // Function: Load()		121
	's onload event when the web application is ready to start	
10 //		
11 function load()		
12 {		
] 13 dashcode.setupParts();		
14 3		
d 15		
16 function btnSettings_ClickHand	ller(event)	
	are (event)	
17 { 18 // Insert Code Here		
18 // Insert Code Here		•
		<b>A</b>
20		v
Ⅲ (Ξ >Ξ		1
		111

3. Notice that the code editor now appears at the bottom of the designer:

#### 4. Enter the following code:

function btnSettings ClickHandler(event)

```
var views = document.getElementById('stackLayout');
var settings = document.getElementById('settings');
if (views && views.object && settings) {
    views.object.setCurrentView(settings);
}
```

 Select the settings subview and right-click on the Save Settings button and select Events→onclick. Name the handler as btnSave\_ClickHandler. Enter the following code:

```
function btnSave ClickHandler(event)
```

```
var views = document.getElementById('stackLayout');
var front = document.getElementById('mainScreen');
if (views && views.object && front) {
    views.object.setCurrentView(front, true);
}
```

{

}

{

6. Test the application again by pressing Command-r. This time, you will be able to navigate to the **settings** view by tapping on the **Settings** button in the **mainScreen** subview:

Il Carrier 🗢 5:58 PM 👄	🔐 Carrier 奈 5:58 PM 🛁
CurrencyConvertor	CurrencyConvertor
file:///var/folders/15/151 🖒 Google	file:///var/folders/15/151 C Google
Enter Amount	USD
Convert to Item 1	EUR
Convert!	AUS
	Save Settings
Settings	
C	

## **Database Access**

1. In the **main.js** file, add the following lines of code for performing database operations:

```
// The client-side database
var database = null;
var DB_tableName = "CurrencyKeyValueTable";
                                                // database name
// Function: initDB() - Init and create the local database, if possible
function initDB()
ſ
    try {
        if (window.openDatabase) {
            database = openDatabase("ExchangeRatesDB", "1.0",
                                    "Exchange Rates Database", 1000);
            if (database) {
                database.transaction(function(tx) {
                    tx.executeSql("SELECT COUNT(*) FROM " +
                    DB_tableName, [],
                    function(tx, result) {
                        loadRates();
                    },
                    function(tx, error) {
                        // Database doesn't exist. Let's create one.
                        tx.executeSql("CREATE TABLE " + DB tableName +
```

Page 13 iPhone Programming using Dashcode and Xcode © Developer Learning Solutions & MobiForge, 2009/2010

```
" (id INTEGER PRIMARY KEY," +
                            key TEXT," +
                         " value TEXT)", [], function(tx, result) {
                             initRates();
                             loadRates ();
                         });
                    });
                });
            }
        }
    } catch(e) {
        database = null;
    }
}
// Function: initRates() - Initialize the default exchange rates
function initRates()
Ł
    if (database) {
        database.transaction(function (tx) {
            tx.executeSql("INSERT INTO " + DB tableName +
                 " (id, key, value) VALUES (?, ?, ?)", [0, 'USD', 1.44]);
            tx.executeSql("INSERT INTO " + DB_tableName +
            " (id, key, value) VALUES (?, ?, ?)", [1, 'EUR', 2.05]);
tx.executeSql("INSERT INTO " + DB_tableName +
                 " (id, key, value) VALUES (?, ?, ?)", [2, 'AUS', 1.19]);
        });
    }
}
// Function: loadRates() - Load the currency exchange rates from DB
function loadRates()
ł
    var element;
    var popUpElement = document.getElementById('popupConvertTo');
    if (database) {
        database.transaction(function(tx) {
            tx.executeSql("SELECT key, value FROM " + DB_tableName, [],
            function(tx, result) {
                 for (var i = 0; i < result.rows.length; ++i) {</pre>
                     var row = result.rows.item(i);
                     var key = row['key'];
                    var value = row['value'];
                     //---populate the pop-up menu part---
                     popUpElement.options[i].text = key;
                     popUpElement.options[i].value = value;
                     if (key == 'USD') {
                         element = document.getElementById('txtUSD');
                     }
                     else {
                         if (key == 'EUR') {
                             element = document.getElementById('txtEUR');
                         }
                         else if (key == 'AUS') {
                             element = document.getElementById('txtAUS');
                         1
                     }
                     element.value = value;
                }
            },
            function(tx, error) {
                 showError('Failed to retrieve stored information from
database - ' +
                     error.message);
```

Page 14 iPhone Programming using Dashcode and Xcode © Developer Learning Solutions & MobiForge, 2009/2010

```
});
        });
    }
    else {
        loadDefaultRates();
    }
}
// Function: saveRates() - Save the currency exchange rates into DB
function saveRates()
ł
    if (database) {
        var elementUSD = document.getElementById('txtUSD');
        var elementEUR = document.getElementById('txtEUR');
        var elementAUS = document.getElementById('txtAUS');
        database.transaction(function (tx) {
            tx.executeSql("UPDATE " + DB_tableName + " SET key = 'USD',
                value = ? WHERE id = 0", [elementUSD.value]);
            tx.executeSql("UPDATE " + DB tableName + " SET key = 'EUR',
                value = ? WHERE id = 1", [elementEUR.value]);
            tx.executeSql("UPDATE " + DB_tableName + " SET key = 'AUS',
                value = ? WHERE id = 2", [elementAUS.value]);
        });
    loadRates();
}
// Function: deleteTable() - Delete currency exchange table from DB
function deleteTable()
ł
    try {
        if (window.openDatabase) {
            database = openDatabase("ExchangeRatesDB", "1.0",
                                     "Exchange Rates Database");
            if (database) {
                database.transaction(function(tx) {
                    tx.executeSql("DROP TABLE " + DB tableName, []);
                });
            }
        }
    } catch(e) {
    }
}
// Function: loadDefaultRates() - Load the default exchange rates
function loadDefaultRates()
    var popUpElement = document.getElementById('popupConvertTo');
    var element = document.getElementById('txtUSD');
    element.value = "1.44";
    popUpElement.options[0].text = "USD";
    popUpElement.options[0].value = element.value;
    element = document.getElementById('txtEUR');
    element.value = "2.05";
    popUpElement.options[1].text = "EUR";
    popUpElement.options[1].value = element.value;
    element = document.getElementById('txtAUS');
    element.value = "1.19";
    popUpElement.options[2].text = "AUS";
    popUpElement.options[2].value = element.value;
3
```

The database code above is pretty straightforward - store the exchange rates inside the database and populate the pop-up menu part when the rates are retrieved.

2. Modify the load() function as follows:

```
//
// Function: load()
// Called by HTML body element's onload event when the web application is
ready to
// start
//
function load()
{
    dashcode.setupParts();
```

```
initDB();
if (!database) {
    loadDefaultRates();
}
}
```

## 3. Modify the btnSave\_ClickHandler() function as follows:

```
function btnSave_ClickHandler(event)
```

```
{
    saveRates();
```

}

```
var views = document.getElementById('stackLayout');
var front = document.getElementById('mainScreen');
if (views && views.object && front) {
    views.object.setCurrentView(front, true);
}
```

4. Press Command-R to test the application. When the application is loaded, the popup menu will now display the three different currencies:

F	-
Il Carrier 🔶 9:26 AM	
ount (	
USD	
Convert!	
Previous Next AutoFill	Done
✓ USD	
EUR	
AUS	
AUS	
AUS	

5. When you tap on the **Settings** button, the exchange rates would also be displayed in the **settings** subview:

Il Carrier 🗢	9:26 AM	
0	CurrencyConvertor	
Tile:///var/fold	ers/15/151 Č	Google
C		
USD	1.44	
EUR	2.05	
AUS	1.19	
	Save Settings	
	+ _0	
	+ ជ	
	+ A	- C
	+ a	7 6

## Performing the Conversion

- You are now ready to perform the actual conversion of the currencies. In Dashcode, select the mainScreen subview and right-click on the Convert! Button and select Events→onclick.
- 2. Name the event handler as btnConvert\_ClickHandler and code it as follows:

```
function btnConvert_ClickHandler(event)
```

```
var amount = document.getElementById("txtAmount").value;
var rates = document.getElementById("popupConvertTo").value;
var result = amount * rates;
alert(result);
```

}

3. Press Command-r to test the application. Enter an amount and select the currency to convert. Tapping on the **Convert!** button will now display the amount converted:

9:47 AM rencyConve 15/151 C	0	
	Goog	ile
t 25 USD		
il <b>e://(null</b> 36	))	
ОК	-	
	Set	lings
+	Ê	Ū
	ile://(null 36 OK	ile://(null) 36 OK Set

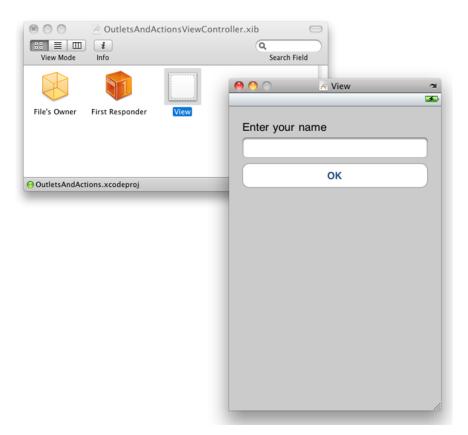
## Lab 2 – Getting Started with Xcode

In this lab, you will learn about the fundamentals of iPhone native application programming - understanding the concepts of actions and outlets. You will then build a simple application that displays the current time.

1. Using Xcode, create a View-based Application project and name it as OutletsAndActions.

iPhone OS			<b>3</b>	
Application Navigation-based Applicati			CLUB (S)	
Window-based Application	Navigation-based Application	Navigation-based Core Data Application	OpenGL ES Application	
User Templates				
MacFUSE				- 1
Mac OS X	* …			- 1
-	Tab Bar Application	Utility Application	View-based Application	
Application Audio Units			Application	
Automator Action Bundle Command Line Utility Dynamic Library Framework Java Kernel Extension Standard Apple Plug-ins	a single vi	late provides a starting point ew. It provides a view controll hat contains the view.		
Static Library				
Other				

 Edit the OutletsAndActionsViewController.xib file by double-clicking on it to open it in Interface Builder. When Interface Builder is loaded, double-click on the View item in the OutletsAndActionsViewController.xib window to visually display the View. Populate the three views onto the View window - Label, TextField, and Button.



## **Creating Actions and Outlets**

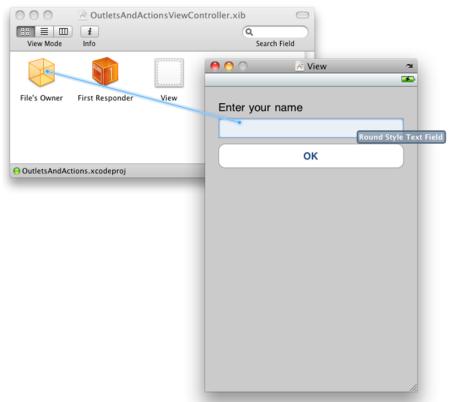
1. In the **OutletsAndActionsViewController.xib** window, select the **File's Owner** item and view its **Identity Inspector** window. Observe that there are two sections here - **Class Actions** and **Class Outlets**.

OutletsAndAd	ctionsViewCo	ontroller.xib
View Mode Info		Search Field
		O O Outlets And Actions View Controller Id.
File's Owner First Responder	View	🗢 🔿 🥖 📵
		V Class Identity
		Class OutletsAndActionsViewContrc
		▼ Class Actions
		Action A Type
OutletsAndActions.xcodeproj		OutletsAndActionsViewController.xib
		btnClicked: id
		+ -
		Class Outlets
		Outlet 🔺 Type
		OutletsAndActionsViewController.xib
		txtName UITextField
		► Interface Builder Identity

- 2. To add an action, click on the "+" button under the **Class Actions** section and name the action as **btnClicked:** (take note of the :). This action will be used to handle the event that will be raised when the button is pressed.
- 3. Likewise, for the outlet, click on the "+" button and name the outlet as **txtName**. For the outlet, you need to specify the type of view you are referring to. In this case, you will use this outlet to connect to the TextField view programmatically. Hence, specify the type as **UITextField**.

## **Connecting Actions and Outlets**

1. In the **OutletsAndActionsViewController.xib** window, control-click and drag the File's Owner item to the TextField view. A popup will appear; select the outlet named **txtName**.



2. To connect an action, you control-click and drag a view to the File's Owner item. Hence, for the OK Button view, control-click and drag the OK Button view to the File's Owner item. Select the action named **btnClicked:**.

000 🖂 Ou	tletsAndActionsViewController.xib		0 0	💉 View	2
	٩				<u>*</u>
View Mode Info	Si	earch Field	Enter your r	name	
	1				
File's Owner First Re	esponder View			ОК	).
					,
OutletsAndActions.xco	deproj	11.			
					1.

## Defining the Actions and Outlets in the View Controller class

1. In the OutletsAndActionsViewController.h file, define the following: #import <UIKit/UIKit.h> @interface OutletsAndActionsViewController : UIViewController {

```
//---declaring the outlet---
```

IBOutlet UITextField \*txtName;

}

```
//---expose the outlet as a property---
@property (nonatomic, retain) UITextField *txtName;
//---declaring the action---
-(IBAction) btnClicked: (id) sender;
```

0end

2. In the OutletsAndActionsViewController.m file, define the following:

```
#import "OutletsAndActionsViewController.h"
```

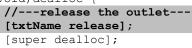
@implementation OutletsAndActionsViewController

```
//---synthesize the property---
@synthesize txtName;
//---displays an alert view when the button is clicked---
-(IBAction) btnClicked:(id) sender {
```

```
NSString *str = [[NSString alloc] initWithFormat:@"Hello, %@",
```

Page 24 iPhone Programming using Dashcode and Xcode © Developer Learning Solutions & MobiForge, 2009/2010

```
txtName.text];
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Hello!"
                             message:str delegate:self
                             cancelButtonTitle:@"Done"
                             otherButtonTitles:nil];
    [alert show];
    [str release];
    [alert release];
- (void) dealloc {
```



}

}

3. Press Command-R to test the application on the iPhone Simulator. When the application is loaded, tap on the TextField view to bring up the keyboard. Enter your name and then press the OK button. You will see the message as shown below:

ſ		
.atil Carrier 🤝	11:02 AM	0
Enter your	name	
Wei-Meng	Lee	
	ОК	
	Hello!	
He	llo, Wei-Meng Le	e
	Done	TIOP
ASD	FGH、	JKL
☆ Z X	СVВМ	N M 💌
.?123	00000	return
.: 120	space	Tetum

## A More Sophisticated Example

Now that you had a good understanding of outlets and actions, let's work on a slightly more sophisticated example so that the application does something useful. You will now modify the application so that you can display the current time continuously, updating it every second. You also have the option to stop and resume the clock if you wish.

1. In the View window, add two additional views - Button and Label:

0 0	🗡 View	2
		<b>S</b>
Enter your	name	
Enter your	hame	
	ок	
1	Start Clock	Ŷ
Label		
Labor		
		11.

2. Change the font of the Label view by selecting it and then pressing Command-t. Change its font as shown below.

		3		0	1	0
			▼ Label			
Enter you	ir name		Text	Label		
			Baseline	Align Centers		•
			Line Breaks	Tail Truncation		•
	ок		Layout	Aignment	# Lines	1 🗘
			Font		t Thin, 48.0	
	Start Clock		Font Size	🗹 Adjust to fit	Minimur	10 🕄
			Color			
	Labe		Shadow	Text Fonts	Highlig	ght
	Labe				Highlig	ght
	Labe	Collection	I T	Fonts T D Typeface		ght
	Labe	Collection English	<u>I</u> • <del>T</del>	T Typeface		
	Labe	Collection English Favorites	I T Family Cocros sans	Fonts		
	Labe	Collection English	T • T Family Lucida Sans Ty Marker Felt Matura MT Scri	T T Typeface Dew Wide		Siz
	Labe	Collection English Favorites Recently Used	Eamily Family Cocross sams Lucida Sams Ty Marker Felt Matura MT Scri Menlo	Typeface Thin Wide		Siz 36
	Labe	Collection English Favorites Recently Used Fixed Width	T • F Family Luciuda Sans Ty Marker Felt Matura MT Scr Menlo Microsoft Sans	Typeface Thin Wide		Siz 36 48 64 72
	Labe	Collection English Favorites Recently Used Fixed Width Fun Modern PDF	T • F Family Lucida Sans Ty Marker Felt Matura MT Scr Menlo Microsoft Sans Mistral	Fonts Typefact Thin Pew pt C Seri		Siz 36 48 64 72 96
	Labe	Collection English Favorites Recently Used Fixed Width Fun Modern PDF Traditional	Family Lucida Sans T Lucida Sans T Marker Felt Matura MT Scr Menlo Microsoft Sans Mistral Modern No. 20	Typefact Typefact Dew pt C Seri		Siz 36 43 64 72 96 14
	Labe	Collection English Favorites Recently Used Fixed Width Fun Modern PDF	T • F Family Lucida Sans Ty Marker Felt Matura MT Scr Menlo Microsoft Sans Mistral	Typefact Typefact Dew pt C Seri		Si 3 6 7 9

3. Save the file in Interface Builder and go back to Xcode.

Page 26 iPhone Programming using Dashcode and Xcode © Developer Learning Solutions & MobiForge, 2009/2010 4. In the **OutletsAndActionsViewController.h** file, add the following declarations:

```
#import <UIKit/UIKit.h>
@interface OutletsAndActionsViewController : UIViewController {
    IBOutlet UITextField *txtName;
    //---add the following declarations---
    IBOutlet UIButton *btnStartStop;
   IBOutlet UILabel *lblClock;
   NSDateFormatter *formatter;
   NSDate *date;
   NSTimer *timer;
}
@property (nonatomic, retain) UITextField *txtName;
//---add the following properties declarations---
@property (nonatomic, retain) UIButton *btnStartStop;
@property (nonatomic, retain) UILabel *lblClock;
-(IBAction) btnClicked: (id) sender;
//---add the following declaration---
```

@end

5. Perform the following connections:

-(IBAction) btnStartStopClicked: (id) sender;

- Connect the IbIClock outlet to the Label view (that one you just added)
- Connect the **btnStartStop** outlet to the Button view (that one you just added)
- Connect the Start Clock button to the btnStartStopClicked: action.
- 6. Save the file in Interface Builder.

#### Coding the Application

1. Back in Xcode, define the following in the **OutletsAndActionsViewController.m** file:

#import "OutletsAndActionsViewController.h"

@implementation OutletsAndActionsViewController

@synthesize txtName;

```
//---synthesize all the properties---
@synthesize btnStartStop;
@synthesize lblClock;
```

```
- (void)viewDidLoad {
    //---initialize the NSDateFormatter object---
    formatter =[[NSDateFormatter alloc] init];
    [super viewDidLoad];
}
-(IBAction) btnStartStopClicked: (id) sender {
    if ([[btnStartStop titleForState:UIControlStateNormal]
    isEqualToString:@"Start Clock"])
    {
        //---start the timer---
        timer = [NSTimer scheduledTimerWithTimeInterval:(1)
            target:self
            selector:@selector(updateTime)
            userInfo:nil
```

Page 27 iPhone Programming using Dashcode and Xcode © Developer Learning Solutions & MobiForge, 2009/2010

```
repeats:YES];
//---change the caption to "Stop Clock"---
[btnStartStop setTitle:@"Stop Clock" forState:UIControlStateNormal];
}
else
{
    //---stop the timer---
    [timer invalidate];
    //---change the caption back to "Start Clock"---
    [btnStartStop setTitle:@"Start Clock"
        forState:UIControlStateNormal];
}
```

```
//---called every second---
-(void)updateTime{
    //---display the time---
    date = [NSDate date];
    [formatter setTimeStyle:NSDateFormatterMediumStyle];
    lblClock.text = [formatter stringFromDate:date];
```

```
- (void) dealloc {
    [txtName release];
    //---release all the outlets and objects---
    [btnStartStop release];
    [lblClock release];
    [formatter release];
    [date release];
    [super dealloc];
```

2. To test the application, press Command-r in Xcode. Tapping the **Start Clock** button will start the clock, which updates itself every one second.

}

}

