Study Guide

embarcadero®

GETTING STARTED WITH WINDOWS AND MAC DEVELOPMENT

Creating your First Windows and Mac Desktop Application

E-Learning Series Course Book – Lesson 2

Embarcadero Technologies

Americas Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

May 10: Lesson 2 – Creating your First Windows and Mac Desktop Application

Version: 1.1
Last Updated: May 13, 2012
Presented: May 23, 2012

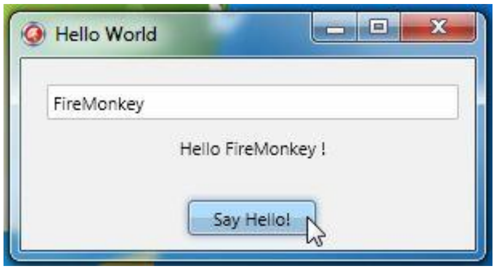
Prepared by: David Intersimone "David I", Embarcadero Technologies

davidi@embarcadero.com
http://blogs.embarcadero.com/davidi/

Contents

Introduction

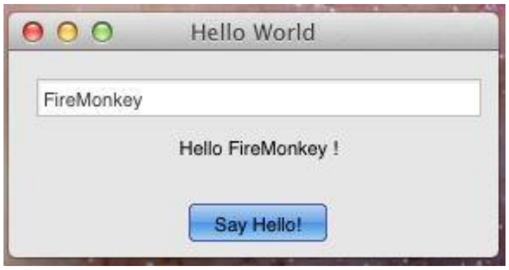
In lesson 1 you were introduced to the FireMonkey business application platform and how to install and set up the development environment. In lesson 2 you will build your first FireMonkey application. This lesson is designed to give you a basic understanding of the development workflow for building a Windows and Mac application with the FireMonkey business application platform. You'll build an easy-to-follow "Hello World" application that demonstrates the essential steps for creating your first FireMonkey application for the Windows and Mac OS X desktop platforms. Delphi developers will also build the same application for iOS. The basic development steps are the same between Windows, Mac, and iOS.



Hello World

FireMonkey

Hello FireMonkey !



Hello World

FireMonkey

Hello FireMonkey !

Creating your first Windows and Mac application (Delphi and C++)

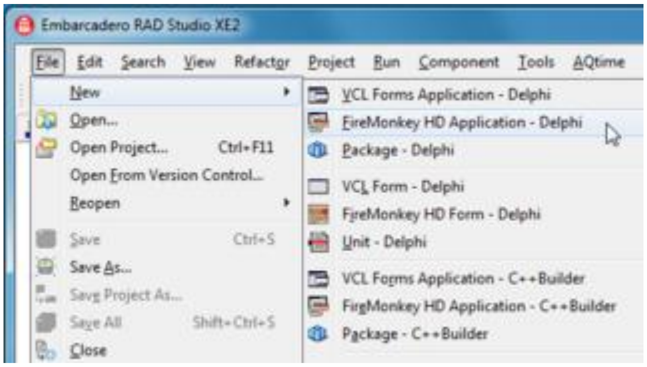
The "Hello World" application works on Windows and Mac OS X and uses the following user interface elements:

- A text field to enter a name.
- A button to click to say "Hello".
- A label to display the message "Hello + <your name>!"

There are six steps in the development workflow to design, program, compile, deploy and run the "Hello World" application on Windows and Mac. Let's get started!

Step 1: Create a new FireMonkey HD Application

Start the RAD Studio IDE. The File menu allows you to create a new FireMonkey application. To create a new FireMonkey HD application, select *File > New > FireMonkey HD Application - Delphi* or *File > New > FireMonkey HD Application - C++Builder*.

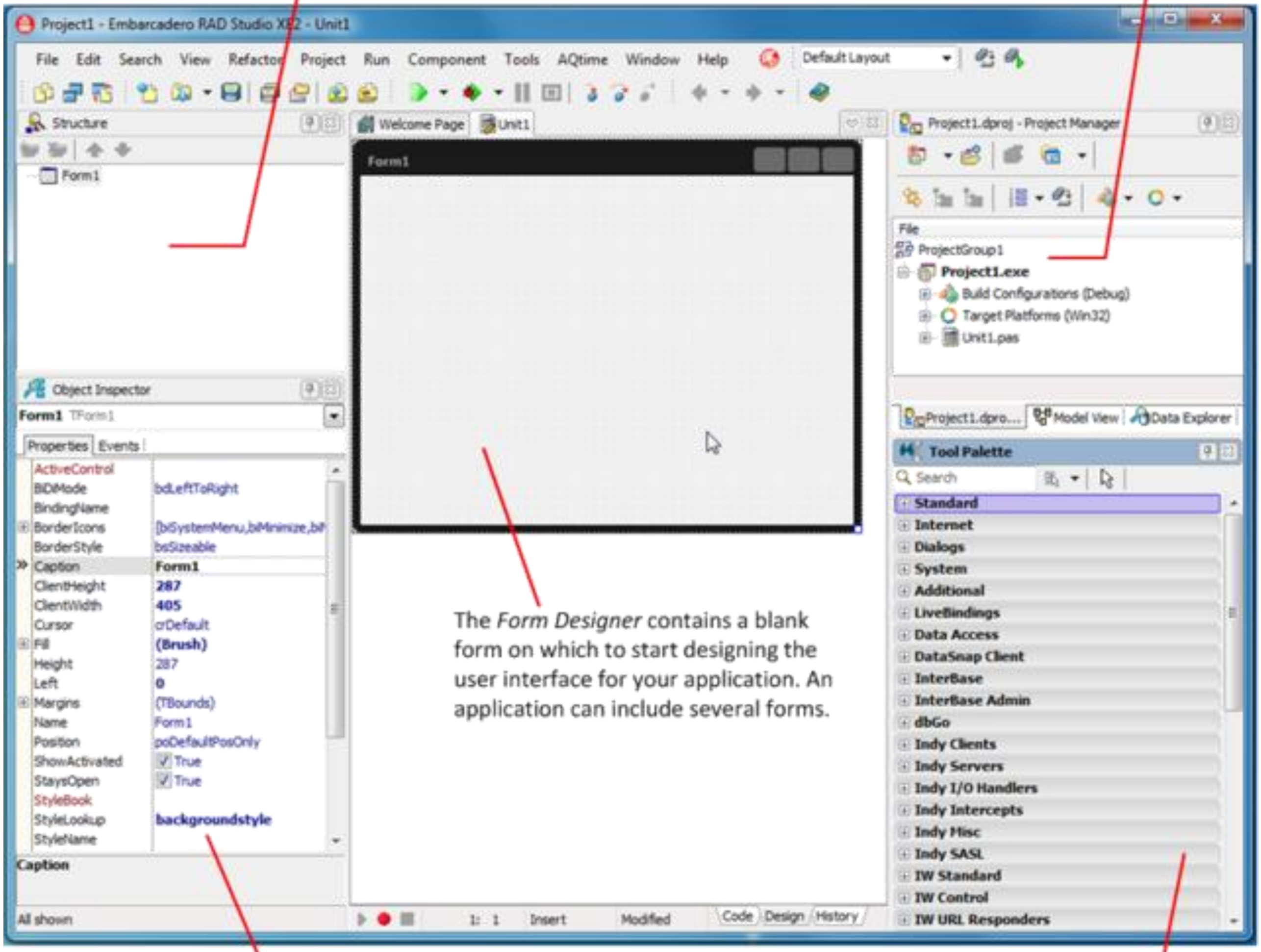


The IDE has 5 different views that you can use to help you design, compile, debug and run your applications. You can dock, undock, or close any of the views. You can always re-open any of the views using the **View** menu. Using the default layout you will see the following views when you start the IDE:

- Form Designer
- Structure View
- Object Inspector
- Project Manager
- Tool Palette

The *Structure View* displays a hierarchical view of your components' parent-child relationships.

The *Project Manager* displays a hierarchical view of the files inside the current project.



ActiveControl	
BiDiMode	bdLeftToRight
BindingName	
BorderIcons	[biSystemMenu,biMinimize,bi
BorderStyle	bsSizeable
Caption	Form1
ClientHeight	287
ClientWidth	405
Cursor	crDefault
Fill	(Brush)
Height	287
Left	0
Margins	(TBounds)
Name	Form1
Position	poDefaultPosOnly
ShowActivated	☑ True
StaysOpen	☑ True
StyleBook	
StyleLookup	backgroundstyle
StyleName	

The *Form Designer* contains a blank form on which to start designing the user interface for your application. An application can include several forms.

The *Object Inspector* is used to change objects' properties and create event handlers.

The *Tool Palette* contains ready-made components to add to your projects.

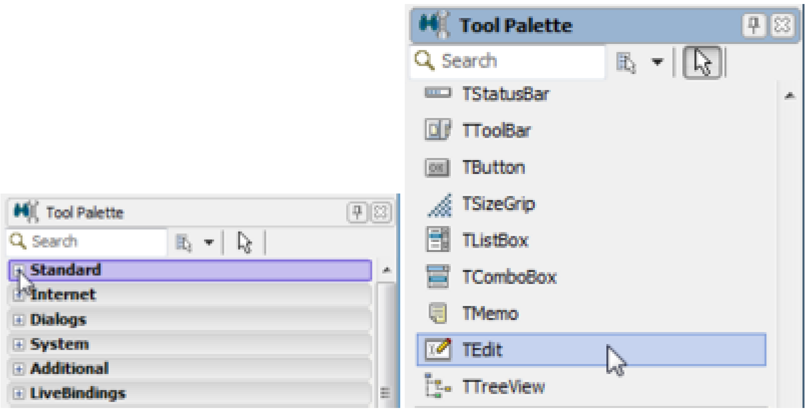
We will cover how you can create your own custom layouts in Lesson 3: The Integrated Development Environment.

Step 2: Place Components on the FireMonkey HD Form

After you've created the starting project, the second step is to design the user interface. There are hundreds of reusable components available in the IDE for creating user interfaces. There are three ways to find the components you want to use in your application.

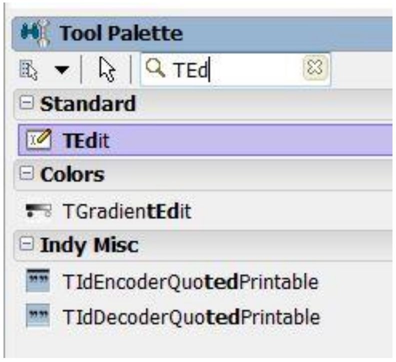
Finding Components using the Tool Palette

Move your cursor over the Tool Palette and expand the Standard category by clicking the plus (+) icon. Then select the **TEdit** component and drag and drop it onto the Form Designer (or double click on the TEdit component in the Tool Palette and it will be added to the form).



Finding Components using the Tool Palette’s Search Box

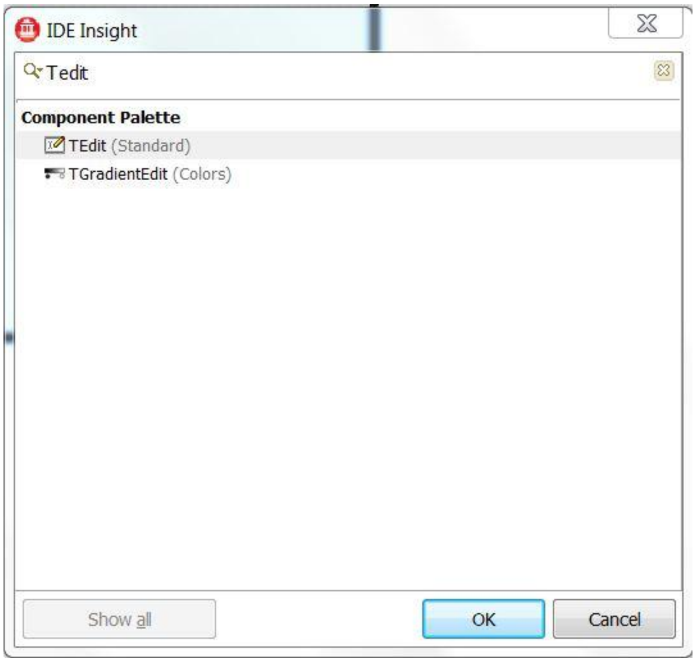
You can also use the Tool Palette’s search box to find a component. Click in the search box and start typing the letters “ted” and then select the **TEdit** component from the component choices presented.



Then select the **TEdit** component and drag and drop it onto the Form Designer (or double click on the TEdit component in the Tool Palette and it will be added to the form).

Finding Components using IDE Insight

The third way to find a component is to use **IDE Insight** by hitting the F6 key (or Control .) and start typing the word “TEdit” and you’ll see several choices (Components, Templates, Projects and other IDE capabilities that match the text you are typing) start to appear. Select the TEdit component from the list to add it to the form.



IDE Insight

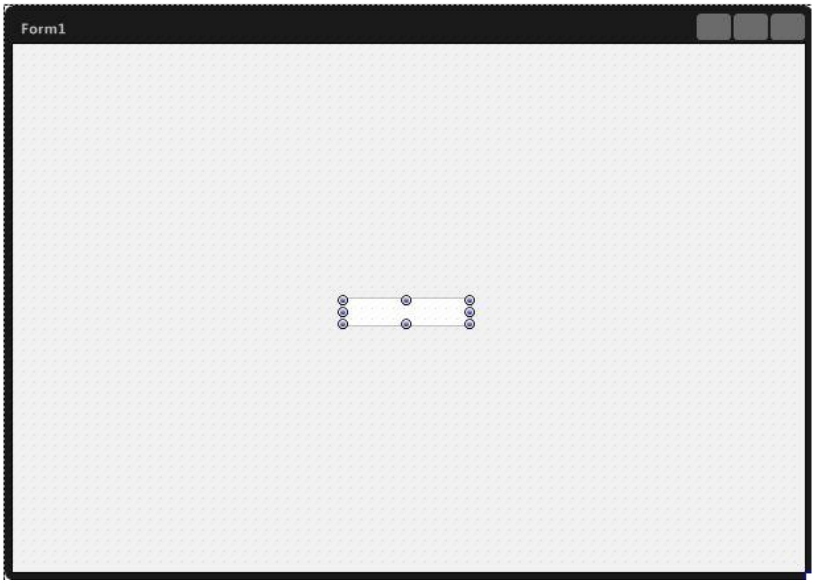
Tedit

Component Palette

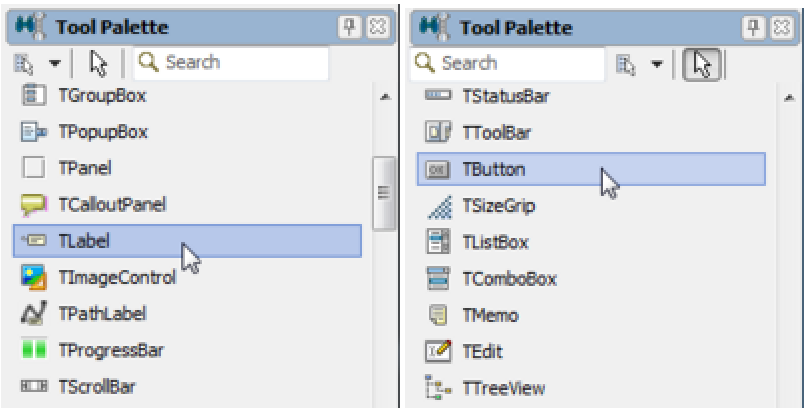
TEdit (Standard)

TGradientEdit (Colors)

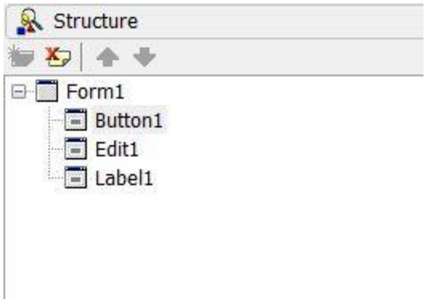
Regardless of which method you use to find the component, an instance of the **TEdit** component will be displayed on the form.



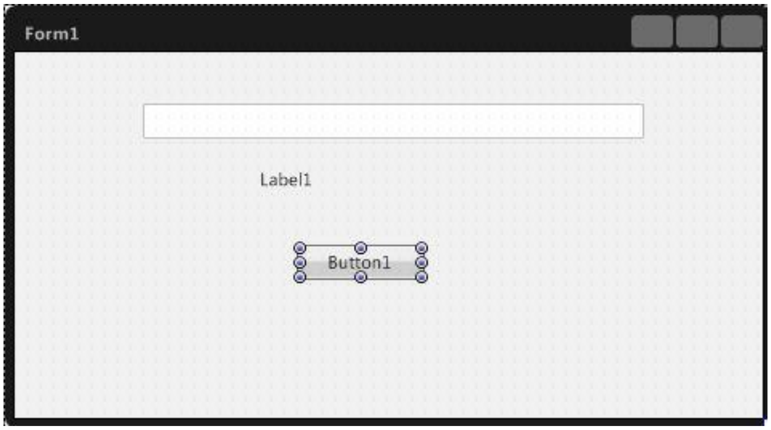
Use your favorite method to add the **TLabel** and **TButton** components onto the form.



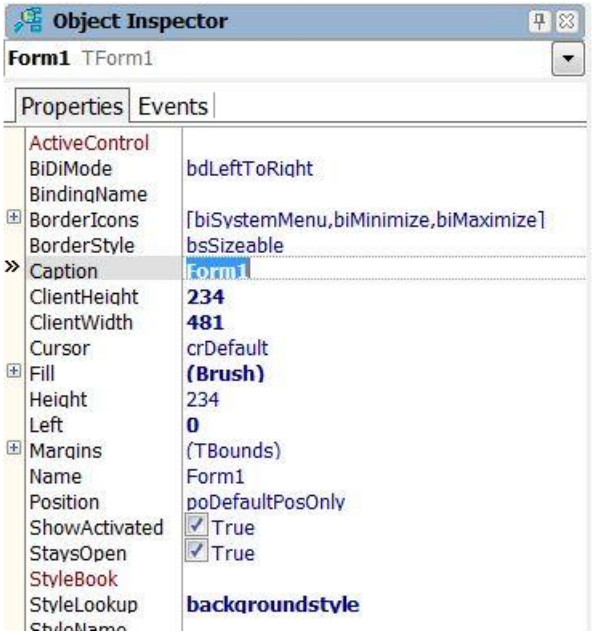
You should now see three components in the Form Designer all placed on top of each other. The three components are also listed in the Structure Pane in the upper left side of the IDE.



Use the mouse to resize the form (hold down the left mouse button on the lower right border of the form and drag the form up and to the left) and rearrange and resize the components on the Form Designer to look something like the following.



You can view and change a component's properties using the Object Inspector after selecting the component on the form. You can also select the form and change its properties. There are two additional ways to select an object to change their properties: 1) Select the form or a component in the Structure Pane and switch to the Object Inspector, 2) use the down arrow at the top of the Object Inspector to select the form or its components.



Object Inspector

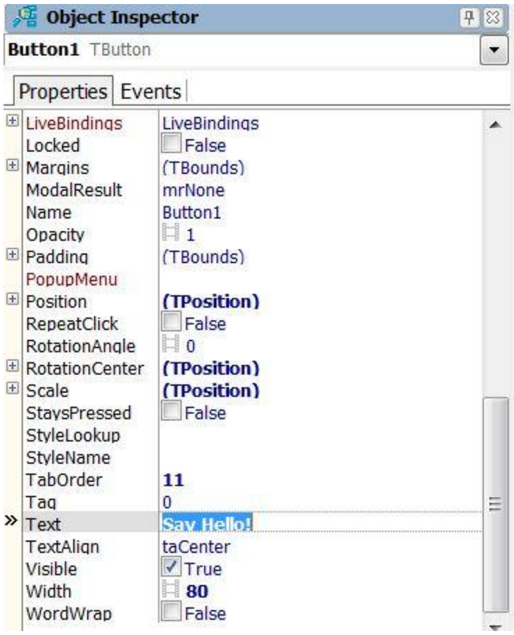
Form1 TForm1

Properties | Events

ActiveControl	
BiDiMode	bdLeftToRight
BindingName	
⊞ BorderIcons	[biSystemMenu,biMinimize,biMaximize]
BorderStyle	bsSizeable
» Caption	Form1
ClientHeight	234
ClientWidth	481
Cursor	crDefault
⊞ Fill	(Brush)
Height	234
Left	0
⊞ Margins	(TBounds)
Name	Form1
Position	poDefaultPosOnly
ShowActivated	☑ True
StaysOpen	☑ True
StyleBook	
StyleLookup	backgroundstyle
StyleName	

In the Object Inspector, change the form's **Caption** property to "Hello World" and hit the Enter key to apply the change.

Next, select the **TButton** component (on the form, in the Structure Pane or from the Object Inspector pull-down) and in Object Inspector select the Text property and type "Say Hello!" and hit the Enter key to apply the change to the button's caption. We will change the **TEdit** and **TLabel** properties programmatically by entering some code, as described in Step 3 below.

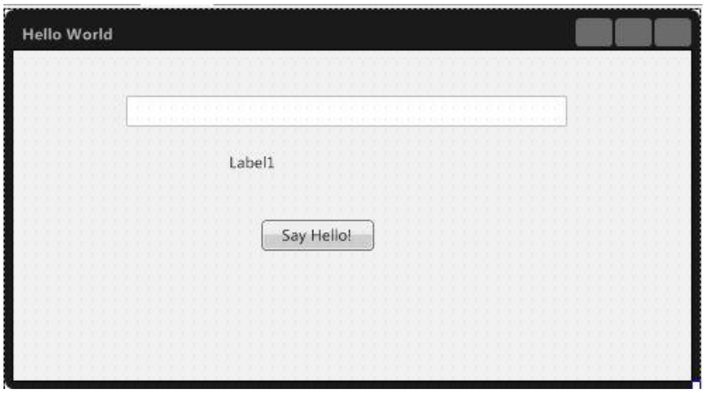


Object Inspector

Button1 TButton

Properties | Events

⊞ LiveBindings	LiveBindings
Locked	□ False
⊞ Margins	(TBounds)
ModalResult	mrNone
Name	Button1
Opacity	1
⊞ Padding	(TBounds)
PopupMenu	
⊞ Position	(TPosition)
RepeatClick	□ False
RotationAngle	0
⊞ RotationCenter	(TPosition)
⊞ Scale	(TPosition)
StaysPressed	□ False
StyleLookup	
StyleName	
TabOrder	11
Tag	0
» Text	Say Hello!
TextAlign	taCenter
Visible	☑ True
Width	80
WordWrap	□ False

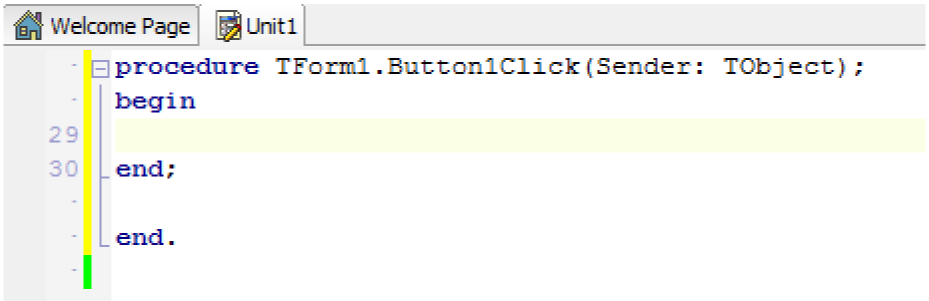


Step 3: Write Code for the TButton On-Click Event Handler

For GUI applications, most of the responses to users' actions such as button clicks and text field entries can be implemented as a response to an event. In RAD Studio, such responses are referred to as event handlers.

There are three ways to create the **OnClick** event handler skeleton code for the **TButton** component.

1) Double-click on the button in the Form Designer and RAD Studio will create the skeleton code to implement an event handler for the TButton's OnClick event.

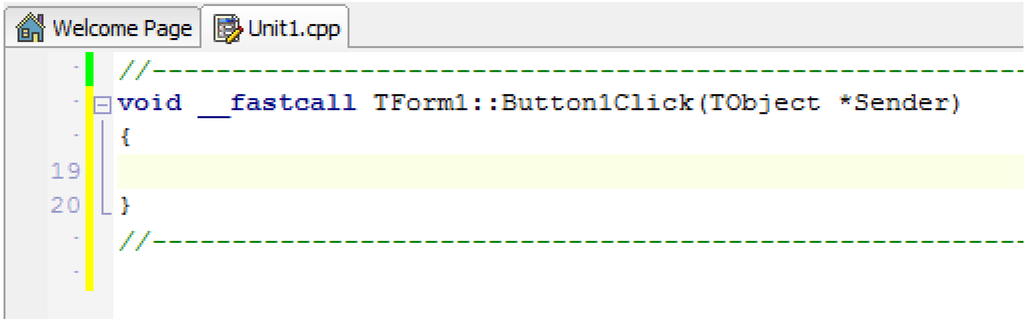


```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;

end.
```

(Delphi)

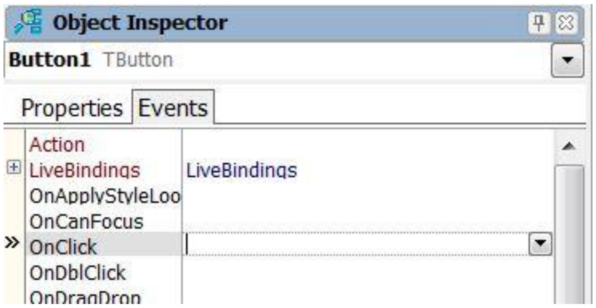


```
//---------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{

}
//---------------------------------------------------------------
```

(C++)

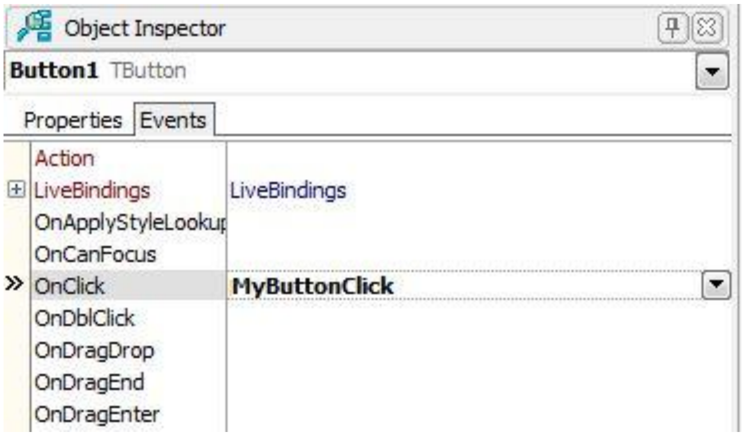
2) Use the Object Inspector's Events tab and double click on the **OnClick** entry for the **TButton** component.



Button1 TButton

Action	
LiveBindings	LiveBindings
OnApplyStyleLoo	
OnCanFocus	
OnClick	
OnDblClick	
OnDragDrop	

3) Select the OnClick event entry and type the name you want for your event handler.



Button1 TButton

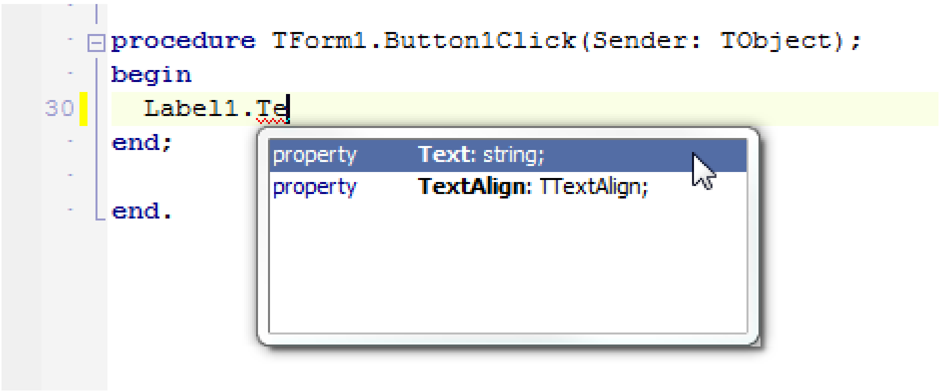
Action	
LiveBindings	LiveBindings
OnApplyStyleLookup	
OnCanFocus	
OnClick	MyButtonClick
OnDblClick	
OnDragDrop	
OnDragEnd	
OnDragEnter	

Next, you'll write the code for the TButton event handler. The code will take the contents of the edit box and change the label to display "Hello + <text entered into the edit box> + !".

Delphi Event Handling Code

```
Label1.Text := 'Hello ' + Edit1.Text + ' !';
```

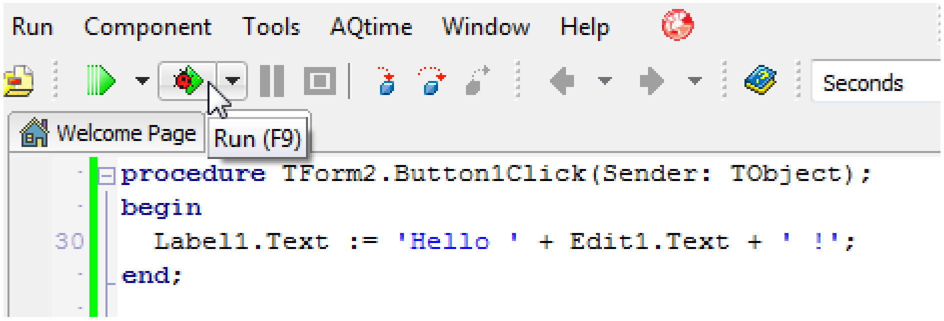
In Delphi, the quotation marks that must surround string literals are ' '. You can also use the plus (+) sign to concatenate strings. While you are typing code, some hints indicating the kind of parameter you need to specify will appear. Also, hints will indicate the kinds of members that are supported in a given class.



```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Te
end;

end.
```

```
property    Text: string;
property    TextAlign: TTextAlign;
```

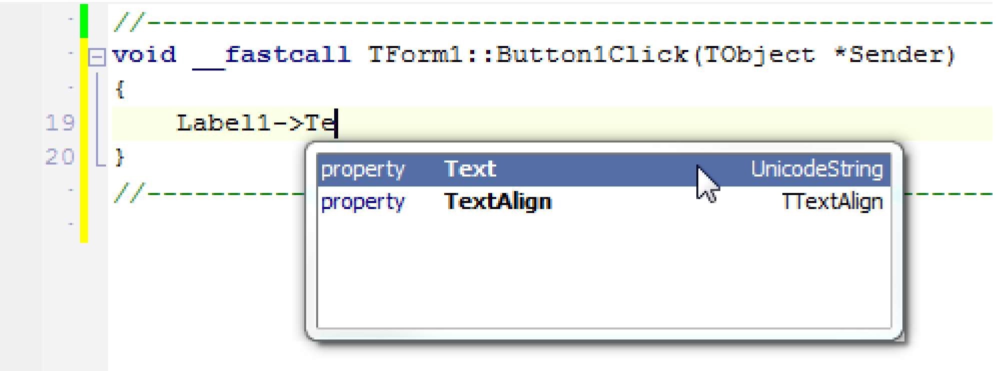


```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Label1.Text := 'Hello ' + Edit1.Text + ' !';
end;
```

C++ Event Handling Code

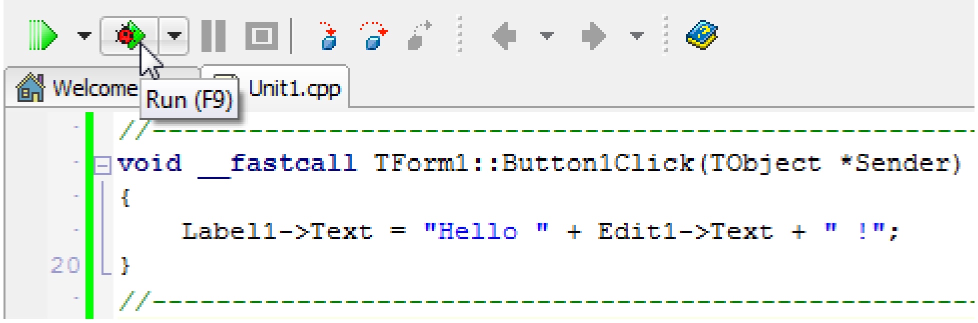
```
Label1->Text = "Hello " + Edit1->Text + " !";
```

In C++, the quotation marks that must surround string literals are " ". You can also use the plus (+) sign to concatenate strings. While you are typing code, some hints indicating the kind of parameter you need to specify will appear. Also, hints will indicate the kinds of members that are supported in a given class.



```
//--------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Label1->Te
}
//--------------------------------------------------------------
```

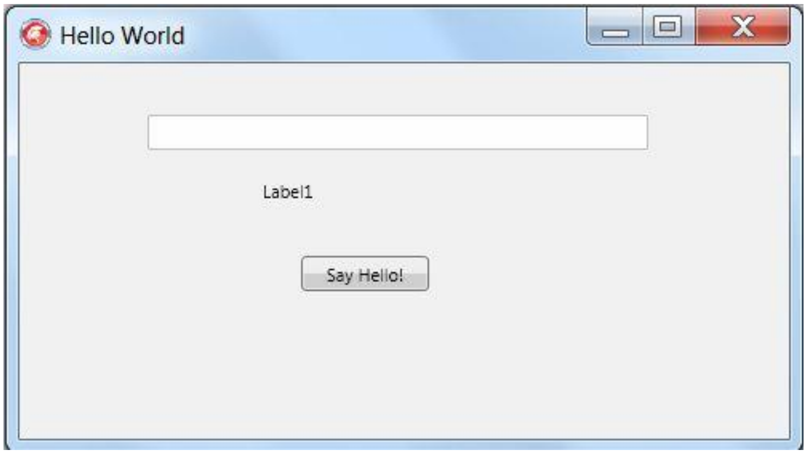
property	**Text**	UnicodeString
property	**TextAlign**	TTextAlign



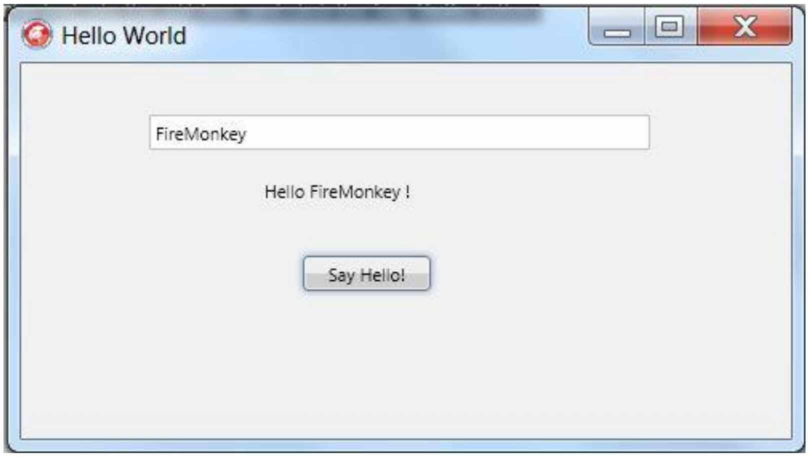
```
//--------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Label1->Text = "Hello " + Edit1->Text + " !";
}
//--------------------------------------------------------------
```

Step 4: Execute the Application

The implementation of this application is finished, so now you can run it. You can click the Run button in the IDE, press F9, or select Run > Run from the RAD Studio main menu. The project will be compiled and if there are no errors, the running application main window will appear.



Enter some text into the edit box (FireMonkey), and click the "Say Hello!" button. If your button click event handler is correct you will see the following display.



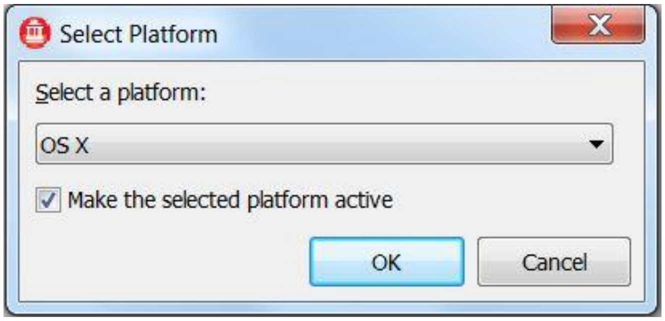
Congratulations! You have successfully designed, programmed, compiled and executed your first FireMonkey application for 32-bit Windows.

By default, RAD Studio creates applications that target 32-bit versions of the Windows operating system. For Delphi applications, you can also compile the project for 64-bit Windows. In the Project Manager, right-click on the Target Platforms node and select the ***Add Platform...*** choice from the popup menu. Select "64-bit Windows" from the list (For C++ projects "OS X" is the only choice). You can also choose to make the selected platform the new active platform using the check box. Click the OK button. Now your Delphi project can be built for both Win32 and Win64 platforms. Press F9 to execute your Win64 application.

Step 5: Supporting Mac OS X

By default, RAD Studio creates applications that target 32-bit versions of the Windows operating system. To compile and execute a Mac OS X version of the application you need to add Mac OS X as a target platform for your project.

In the Project Manager, right-click on the Target Platforms node and select the ***Add Platform...*** choice from the popup menu. Select “OS X” from the list (For C++ projects “OS X” is the only choice. You can also choose to make the selected platform the new active platform using the check box. Click the OK button.

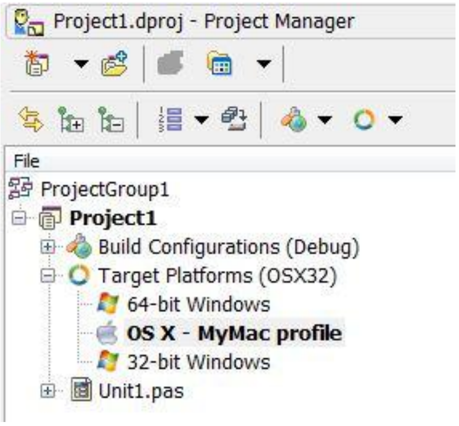

Select Platform

Select a platform: OS X

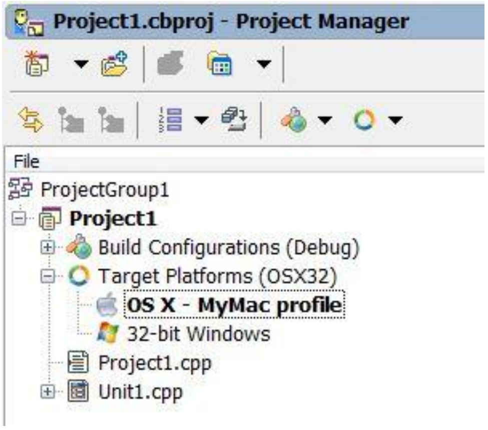
☑ Make the selected platform active

Now your project can be built for both Win32 and Mac OS X platforms. The RAD Studio development environment runs on Windows. You cannot run native OS X applications on your Windows machine. You need to connect your IDE to a Mac OS X machine. To deploy your application to Mac OS X and establish remote debug sessions automatically, RAD Studio provides a tool called **Platform Assistant** (PAServer). In Lesson 1 we showed you how to create a remote profile for your Macintosh computer.

In the Project Window, click the plus (+) sign to the left of the Target Platform node to expand the list of targets.


Project1.dproj - Project Manager

File
- ProjectGroup1
  - **Project1**
    - Build Configurations (Debug)
    - Target Platforms (OSX32)
      - 64-bit Windows
      - **OS X - MyMac profile**
      - 32-bit Windows
    - Unit1.pas

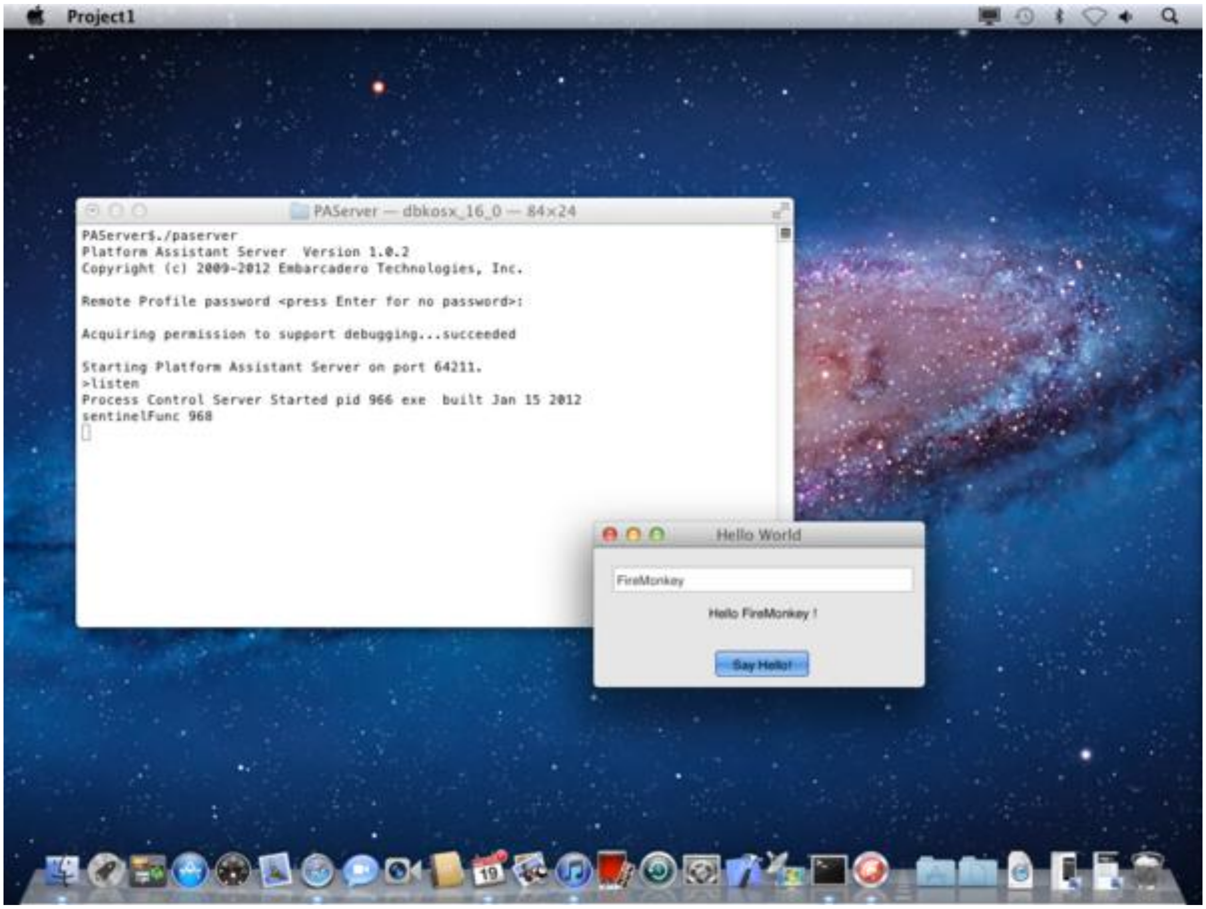

Project1.cbproj - Project Manager

File
- ProjectGroup1
  - **Project1**
    - Build Configurations (Debug)
    - Target Platforms (OSX32)
      - **OS X - MyMac profile**
      - 32-bit Windows
    - Project1.cpp
    - Unit1.cpp

The Target Platform list shows that OSX32 is the current target. Also notice that your OS X target node is active and “MyMac” remote profile created in Lesson 1 has been selected (you can designate one of your remote profiles as the default profile for each Target Platform).

Step 6: Run the Application on Mac OS X

Make sure you have started the PAServer on your target Macintosh. Press F9 to run the “Hello World” application on your Mac. Your application, running on the Mac, should look similar to the one shown in the following image.



```
PAServer$./paserver
Platform Assistant Server  Version 1.0.2
Copyright (c) 2009-2012 Embarcadero Technologies, Inc.

Remote Profile password <press Enter for no password>:

Acquiring permission to support debugging...succeeded

Starting Platform Assistant Server on port 64211.
>listen
Process Control Server Started pid 966 exe  built Jan 15 2012
sentinelFunc 968
```

Congratulations again! You have just designed, programmed, compiled, deployed and executed your first Macintosh application.

Creating your First iOS App (Delphi only in XE2)

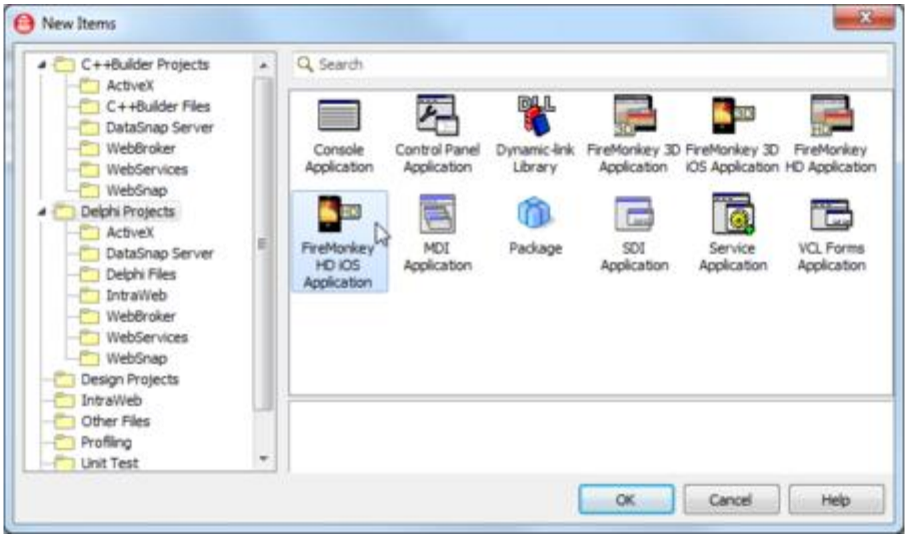
A FireMonkey application destined for iOS is created and tested initially as a native Win32 application in RAD Studio. Then you run the dpr2xcode tool to create an Xcode project for your iOS project. The second half of the FireMonkey iOS application development process occurs on the Mac using Xcode, the iOS simulator and on test iOS devices. That is, you do your production debugging on the Mac using the Xcode development platform, you compile the source files on the Mac using Free Pascal, and you run the iOS app using the FireMonkey and Free Pascal RTL.

In Lesson 1 you learned how to install the FireMonkey tools for iOS. Additional information about installing the FireMonkey-iOS tools on your Mac is available online at http://docwiki.embarcadero.com/RADStudio/XE2/en/Installing_FireMonkey-iOS_Tools_on_Mac.

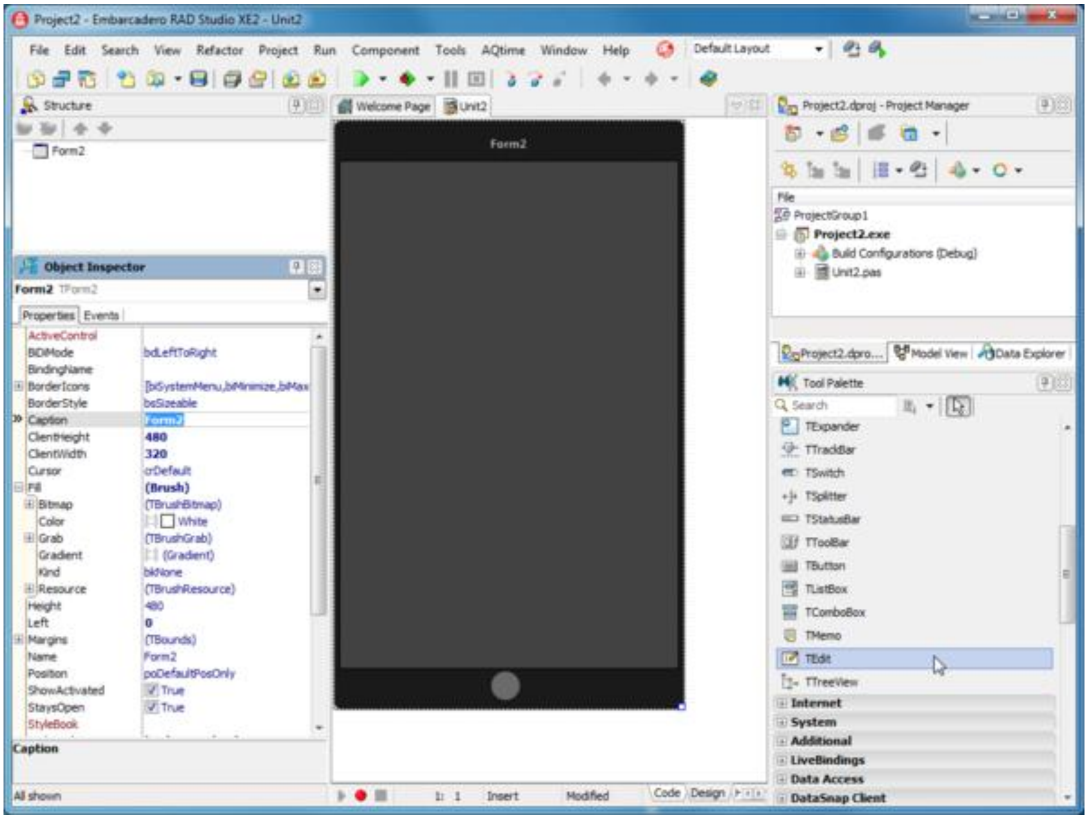
Create the FireMonkey HD iOS Application Project

To create a new FireMonkey iOS application, Select the ***File > New > Other…*** menu item to see the object gallery (New Items).

Select the FireMonkey HD iOS Application project wizard.



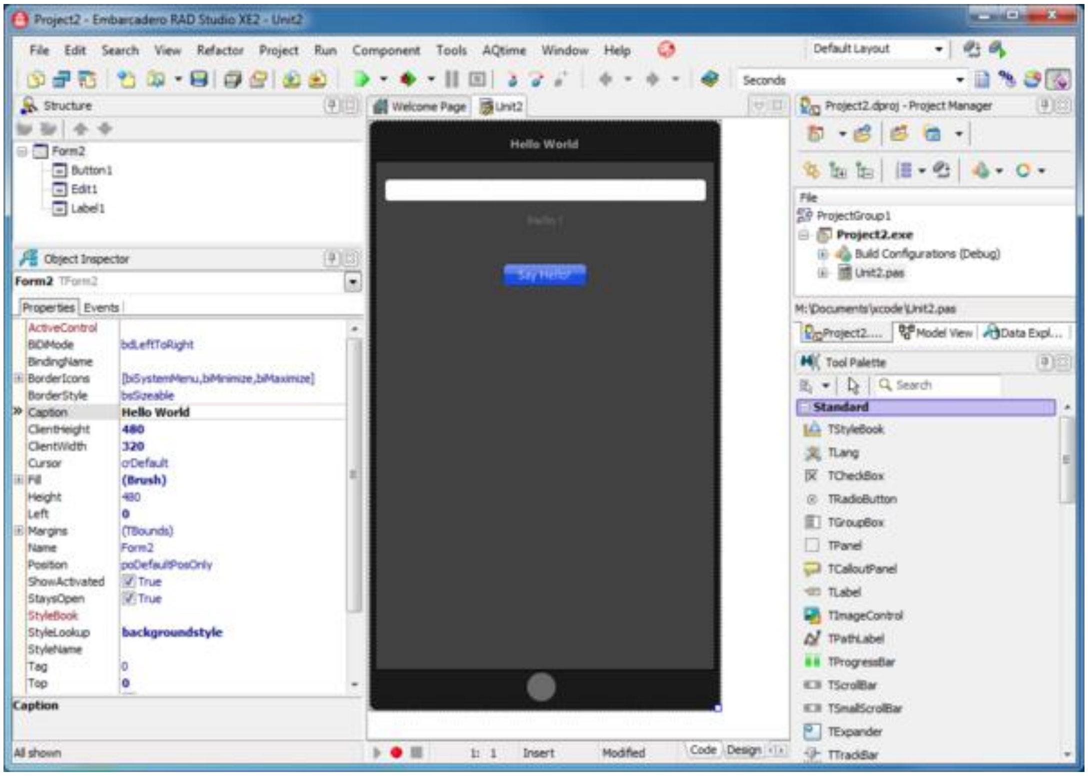
In the RAD Studio IDE you will see the Form Designer with the new form for your iOS Application.



ActiveControl	
BiDiMode	bdLeftToRight
BindingName	
BorderIcons	[biSystemMenu,biMinimize,biMax
BorderStyle	bsSizeable
Caption	Form2
ClientHeight	480
ClientWidth	320
Cursor	crDefault
Fill	(Brush)
Bitmap	(TBrushBitmap)
Color	□ White
Grab	(TBrushGrab)
Gradient	(Gradient)
Kind	bkNone
Resource	(TBrushResource)
Height	480
Left	0
Margins	(TBounds)
Name	Form2
Position	poDefaultPosOnly
ShowActivated	☑ True
StaysOpen	☑ True
StyleBook	

Design the UI for the iOS Application

Add the TButton, TEdit and TLabel components to your iOS form just like you did in Step 2 for the FireMonkey HD Desktop Application. Using the Object Inspector, set the Caption property for the form and the Text property for the TButton as you did in Step 2 for the FireMonkey HD Desktop Application.



Object Inspector

Form2 TForm2

Properties | Events

Property	Value
ActiveControl	
BiDiMode	bdLeftToRight
BindingName	
BorderIcons	[biSystemMenu,biMinimize,biMaximize]
BorderStyle	bsSizeable
Caption	Hello World
ClientHeight	480
ClientWidth	320
Cursor	crDefault
Fill	(Brush)
Height	480
Left	0
Margins	(TBounds)
Name	Form2
Position	poDefaultPosOnly
ShowActivated	☑ True
StaysOpen	☑ True
StyleBook	
StyleLookup	backgroundstyle
StyleName	
Tag	0
Top	0

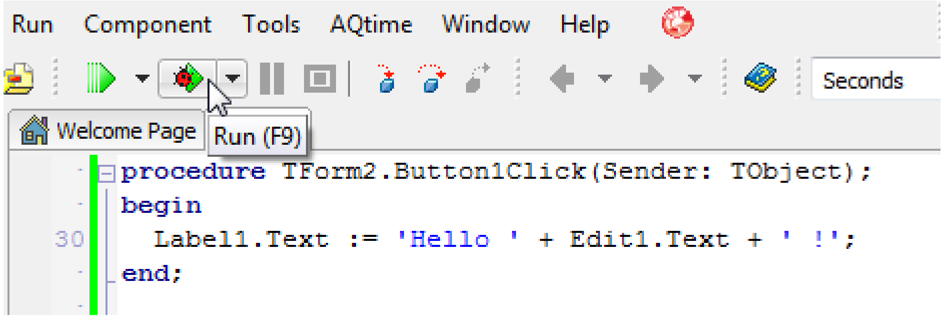
Caption

All shown

Write Delphi Code for the TButton On-Click Event Handler

Write the Delphi Code for the TButton On-Click Event Handler just like you did in Step 4 for the FireMonkey HD Desktop Application.

```
Label1.Text := 'Hello ' + Edit1.Text + ' !';
```

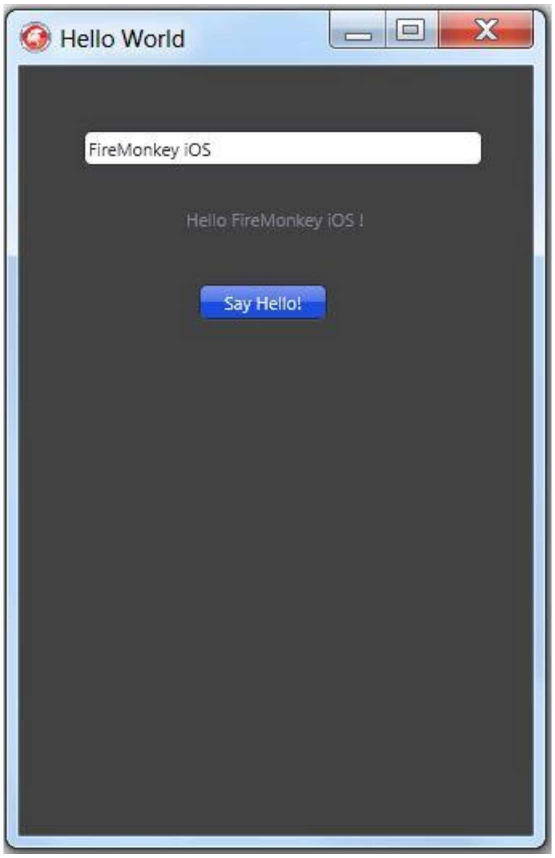


```
procedure TForm2.Button1Click(Sender: TObject);
begin
  Label1.Text := 'Hello ' + Edit1.Text + ' !';
end;
```

Save the project and project files using **File > Save All (Shift+Ctrl+S)** to a shared drive accessible to your Mac (Note: the path on the shared drive should NOT include any spaces as this will cause an Xcode compile time error).

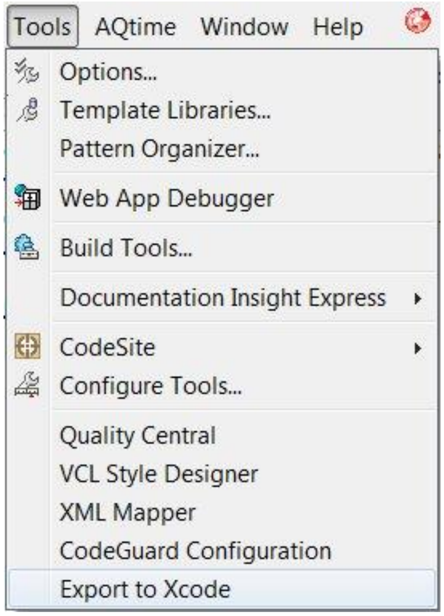
Test your iOS Application on Windows

The implementation of your iOS application is finished. You can compile and test it on Windows. Click the Run button in the IDE, press F9, or select Run > Run from the RAD Studio main menu. The application window will appear. Enter some text in the edit box and click the button.

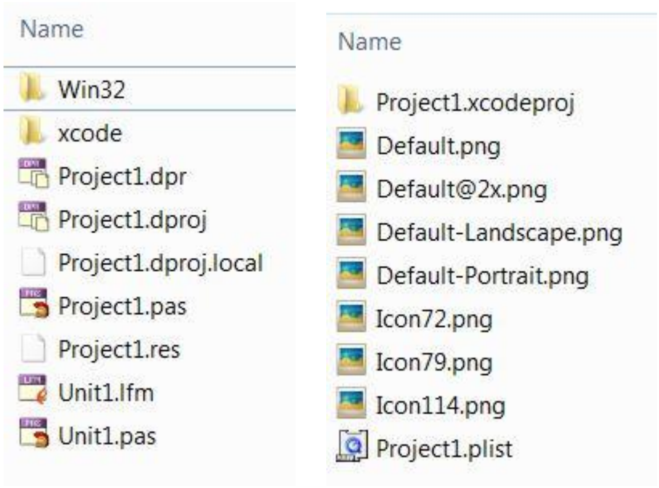


Run Export to Xcode (the dpr2xcode.exe tool)

The application must now be recompiled using Xcode on your Mac in order to run it with the iOS Simulator or on an iOS device. Run the “Export to Xcode” tool (in Lesson 1, we added the dpr2xcode utility to the Tools menu).



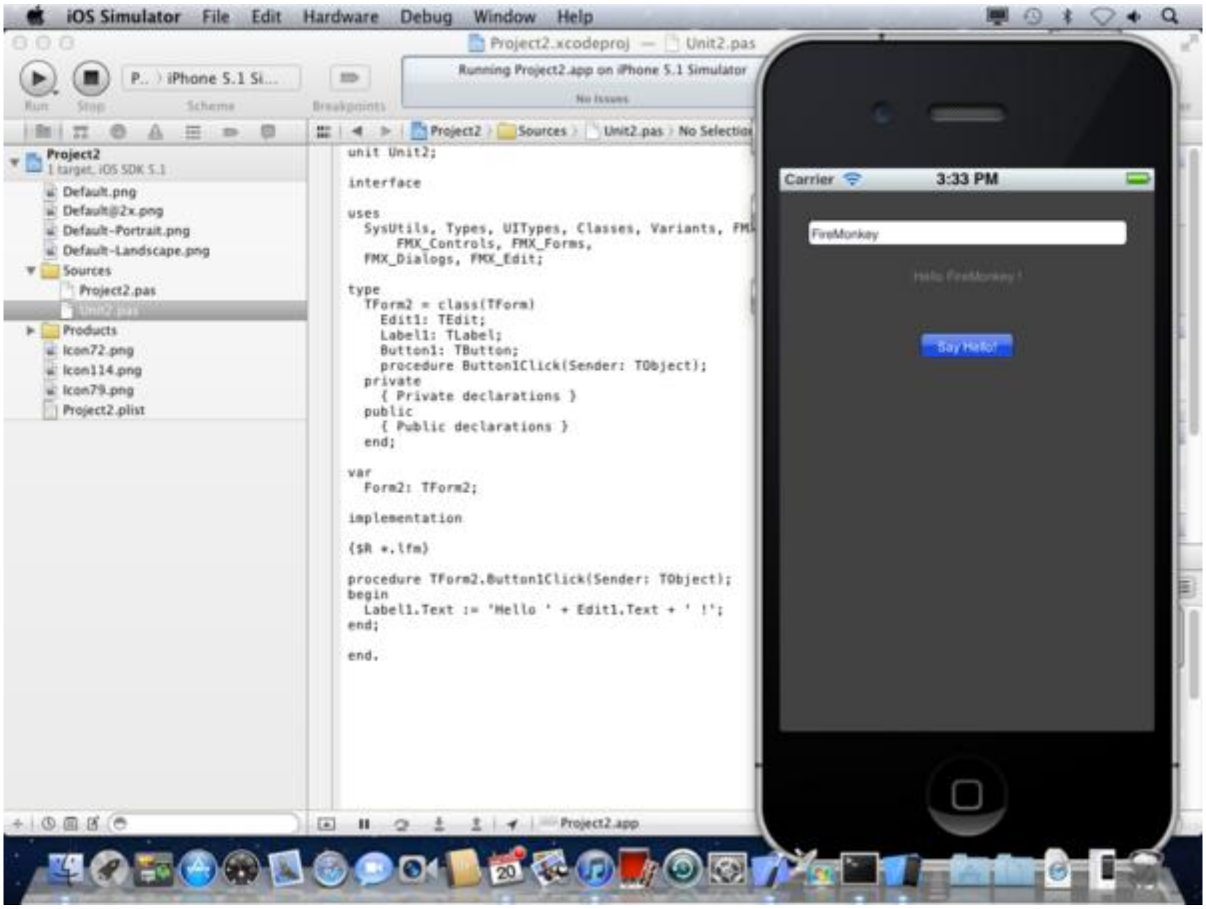
After dpr2xcode runs, in your iOS project directory you will see the following files and sub-folders:



The Win32 folder is present because you tested your iOS project in Windows. The Xcode folder contains the ".xcodeproject" file and additional files needed for your iOS application including the icon files and project file list.

Load the Project into Xcode, Compile It, and Test It

You are ready to execute your iOS application on your iOS Simulator. Start Xcode on your Mac. Open the Project1.xcodeproj project in Xcode using File > Open. Select the device (simulator or actual device) and click the Run button on the Xcode toolbar. Your FireMonkey application is now running on the iOS Simulator (or even on your iOS device).



```
unit Unit2;

interface

uses
  SysUtils, Types, UITypes, Classes, Variants, FM
      FMX_Controls, FMX_Forms,
  FMX_Dialogs, FMX_Edit;

type
  TForm2 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.lfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
  Label1.Text := 'Hello ' + Edit1.Text + ' !';
end;

end.
```

Congratulations! You have just designed, programmed, compiled, deployed and executed your first Apple iOS application.

Summary, Looking Forward, To Do Items, Resources, Q&A and the Quiz

In Lesson 2 you learned how to build your first Windows and Mac desktop application. Delphi developers also learned how to use the same steps to create your first Apple iOS application (don't worry C++ developers; we are working on a next generation C++ compiler that will support iOS).

In Lesson 3, you'll learn more about the capabilities of the RAD Studio Integrated Development Environment (IDE). The IDE helps you visually design user interfaces, set object properties, write code, and view and manage your application in various ways.

Until the next lesson, here are some things you can try, articles to read and videos to watch to enhance what you learned in Lesson 2 and to prepare you for lesson 3.

To Do Items

Use your paid or trial copy of Delphi XE2, C++Builder XE2 or RAD Studio XE2 to create the first Windows and Mac "Hello World" desktop application. For the form and each of the three components (TButton, TLabel and TEdit) use the Object Inspector to explore the different properties of each object. Use the Object Inspector to delete the Text for the TLabel component so that it is empty when you run the application. Put some default text in the TEdit text property so that the edit box is pre-filled when you run the application. Delphi developers, do the same for the iOS "Hello World" application.

Also, watch the Embarcadero YouTube video "Building your first Delphi and C++Builder FireMonkey HD Applications" (4 minutes and 13 seconds) - http://www.youtube.com/watch?v=WGQ2lXKdvvk.

Links to Additional Resources

- Getting Started Course landing page - http://www.embarcadero.com/firemonkey/firemonkey-e-learning-series
- FireMonkey platform prerequisites - http://docwiki.embarcadero.com/RADStudio/en/FireMonkey_Platform_Prerequisites
- Connecting your PC to a Mac - http://docwiki.embarcadero.com/RADStudio/en/Connecting_Your_PC_to_a_Mac
- Differences between Windows and other platforms - http://docwiki.embarcadero.com/RADStudio/en/Differences_between_Windows_and_Other_Platforms
- Watch the 90 second video "Multi-Platform Development with FireMonkey" - http://www.youtube.com/watch?v=NzLCGvpGGAg
- Read the Marco Cantu white paper, "Discover FireMonkey – The Next Generation Business Application Platform" - http://forms.embarcadero.com/forms/AMUSCA1110FireMonkey-MarcoCantuWhitepaper
- Download the FireMonkey architecture schematic poster that shows the major groupings of the platform (in PDF format) - http://www.embarcadero-info.com/firemonkey/firemonkey_chart_poster.pdf

Q&A:

Here are some of the answers for the questions I've received (so far) for this lesson. I will continue to update this Course Book during and after course. (To be added after the lesson webinars take place).

Q: In a FireMonkey iOS application is it possible to call the AppDelegate functions to implement Push Notifications?
A: Yes, but that requires custom code at this point. See FMX_Platform_iOS.pas to get at it.

Q: Thanks, that wasn't possible before as the function weren't implemented in the Free Pascal code. Has that been fixed?
A: I'm simply speculation that you can. I haven't tried it. I've done a bunch of things on the Objective-C/FPC level that can't be done from just Delphi. This will be addressed in the future, but for now if you want to do stuff we don't support, you can do so by getting at it from the lower levels.

Q: Can we use Delphi and FireMonkey to develop application on Windows Phone 7 and Blackberry?

A: Right now FireMonkey and Delphi support iOS. Other mobile platforms are being looked at for the future. Android will be the next major mobile platform for FireMonkey.

Q: Is there a spec sheet describing minimum MAC needed for development? I've only ever owned PCs.
A: The data sheets for RAD Studio XE2, Delphi XE2, and C++Builder XE2 have the system requirements. These requirements are also covered in "Getting Started with Windows and Mac Development Lesson 1.

Q: Does XE2 still have ALIGNMENT guide lines when moving components around?
A: For VCL, yes. For FireMonkey, not yet, but that is definitely on the roadmap

Q: When and how was the decision made to make the caption of a button now .text vs .caption as it was before?
A: VCL has the Caption for a TButton. In FireMonkey properties that are "text" have the Text property name (TLabel, TEdit, TButton, etc). Only TForm has a Caption, because the string is the caption of the Window.

Q: It looks like in some components we have .top and .left but in others we now have a new meaning to Position, X and Y?
A: In FireMonkey, **TForm** is special. **TForm** does have a Top and Left property. All other visual components on FireMonkey have a Position property - X and Y for HD components and X,Y and Z for 3D components.

Q: Hint: You can change the position of the items in the tool palette by drag and drop
A: Thank you

Q: I had to miss the first session. Is there any info on when it will be available?
A: All course book lessons and lesson video replays will be found on the landing page at http://www.embarcadero.com/firemonkey/firemonkey-e-learning-series

Q: Are more 3rd party vendors getting on the FM wagon?
A: Yes, FireMonkey components are appearing now. TMS Software has several FireMonkey components. FastReports has also released demos of their FastReport reporting components for FireMonkey.

Q: Is there a way to build the .app-application-file automatically? My DelphiXE2 only creates the binary-files so I have to ""create"" the .app-folder manually myself. It still would be awesome to have only one file: the .app-application-file.
A: The app bundle *is* built automatically. Grab it from ths scratch-dir in the PA server. To get the scratch-dir, simple type the character “s” in the PA server when running. Use a Terminal window or Finder to go there and find the app bundle.

Q: You made another project for iOS. Could you have created a new target for the first project to compile for iOS instead?
A: FireMonkey iOS projects use different versions of the FireMonkey libraries that work for iOS using the Free Pascal compiler.

Q: In response to the ""accomplished Delphi"" user question: I'm an accomplished Delphi user also, but I'm watching and participating in all of these lessons simply because I want to make sure I know

everything that ""I should"" know about FireMonkey. I can say that already there's at least 3 tips/tricks that I never knew existed.
A: Great feedback! Thanks!

Q: So the iOS form ""caption"" that is showing is really the Name property, right?
A: Well, the caption would actually show if the form wasn't full screen. It will show if you create a form and use it as a small dialog as opposed to a main form (that is always supposed to be full screen with no borders and no caption).

Q: Do you only need to do ""Export to Xcode"" if you want to open and run it from XCode? Not if your debugging remotely from the Delphi IDE?
A: You cannot debug and/or run an iOS from the IDE (as an iOS app). You can run it as a Win32 app on Windows, but that's only good for very simple testing of your app and tells you very little of on-device experience. You have to export to Xcode in order to build the app and run it on Simulator and/or device.

Q: Does Ampersand in button text in FireMonkey make a accelerator key like it does in VCL?
A: FireMonkey does not support Ampersand accelerator keys for TButton. FireMonkey does support ShortCut(s) for **TMenuBar** menu items. We'll cover designing menus in Lesson 5 – "Designing a FireMonkey HD User Interface".

Q: For those of us accomplished in Delphi, is lesson 5 the one to start watching if we want to learn FireMonkey and OS X / iOS x-platform programming?
A: That's almost impossible to answer. It's something you have to gauge based on your experience and your needs. If you missed anything by that time, you can always watch the replays of previous ones I guess.

Q: Is Xcode 4.3.2 fine? What other components should be installed to install Firemonkey from FireMonkey-iOS.dmg on Mac?
A: Right now you need 4.2 in order to install FPC 2.6.0 and FireMonkey (from the DMG file). You can then use 4.3 as well.

Q: Could you make PAServer to install on Mac without Java installed?
A: I never installed anything else on my Mac.

Q: FPC 2.6.0 needed Command Line Utilities for Xcode.
A: Ah, yes. Xcode 4.3 removed the command lines tools as a default install. You can simply install those from inside Xcode. This was covered in Lesson 1.

Q: I have some customers who I have been working to convince that they should get Delphi XE2. Can they sign up for these webinars?
A: Of course!

Q: FWIW, I've been around Delphi since D1. I've just learned three things I didn't know about the IDE. Every lesson has something of value.
A: That's awesome! I still learn stuff every day too.

Q: I updated my Xcode and can't install FireMonkey anymore. What do I have to do?

A: Xcode 4.3.2? You opted to delete 4.2.1? Download 4.2.1 for Lion - install it. Then install FPC 2.6.0 and FM.

Q: I don't use Macs, but many of my clients do, the concept of developing to both mac and windows is attractive, but in your experience is it realistic to expect Mac targets developed in windows to work without any Mac specific debugging? That sounds too good to be true to me and just wondering if I would really need some way to test on the Mac platform
A: If I understand your question correctly... No, it's not realistic to develop an app of any size without debugging on the target platform before you deploy to your users. In XE2 you develop and cross-compile to Mac OSX. You use the PA Server behind the scenes to deploy and debug to the Mac you're testing against. So except for installing the PA server on the Mac once and starting it each session, there's really nothing else that you need to do on the Mac. The rest is automatic. The app deploys, hooks into the debugger and runs on the Mac. Once running, you would of course use it as any other Mac app on the Mac itself and watch the debugger session on Windows.

Q: Is it possible to create custom components For FireMonkey in Delphi IDE?
A: Yes. You can use the **Components > New Component** menu to create FireMonkey components in the IDE. Also, check out the Embarcadero DocWiki for information about creating FireMonkey components for Delphi and C++:

- http://docwiki.embarcadero.com/RADStudio/en/Creating_a_FireMonkey_Component_(Delphi)
- http://docwiki.embarcadero.com/RADStudio/en/Creating_a_FireMonkey_Component_(C%2B%2B)

Q: I notice when David went to view processes, the 32-bit Hello World app is about 7.5 MB. Why are FireMonkey apps so large?
A: He was using the Debug build configuration instead of the Release build configuration. I deployed a FMX app yesterday that was 3.5MB and it's 10MB+ as debug build. It's just a guess that David's app would be less than 2MB without debug information. You can also select “build with runtime packages” to make even smaller executables (but you would need to deploy the runtime packages with your application).

Q: What does the HD in ""FireMonkey HD application"" stand for?
A: High Definition

Q: you can just drag the components, even to different groups in the palette.
A: True. But there's no ""sort"" option I think.

Q: If I want to deploy a Mac OSX app to a Mac, that I dont have PA server running on.. How do I do that? Which files do I need?
A: The app bundle. Typically FooBar.app (it's actually a directory).

Q: can we use DBGo for ADO components for Mac? (ADO)
A: Off the top of my head I will say no, as I'm pretty sure that they are COM based. No COM on Mac.

Q: Where can we download an updated copy of the Word document lesson file?
A: http://www.embarcadero.com/firemonkey/firemonkey-e-learning-series

Q: Would you need the PA installed on all the target machines you intend to deploy your ""hybrid"" application?
A: No. Only for deploy/debug from the IDE. Once you're done debugging and ready for end-user deployment you simply ship the app bundle somehow (email, ftp, appstore, whatever).

Q: if I code on Win32 or Win64 using dll's what effect does that have on porting to OSX or iOS?
A: You can create dynamic libraries (just like you do on Windows) for OSX - they are called dylibs. Just create a Dynamic Library project type and add your code. When the target is Windows, the compiler will create a DLL file. When the target is OSX, the compiler will create a dylib file.

Q: My program needs a little bit assemble code to be got involved. Will it still be supported under 64-bit system? This is very important to me.
A: Yes. But you have to write 64-bit assembler instead of 32-bit assembler... Also, a function that has assembler has to be entirely written in assembler. No inline here and there anymore.

Q: I have a question about the TThread Class. I can't setup a TThread object in a DLL file, because I found the thread procedure can not be triggered. But it used to fine in your older versions, even in 2006. Do you still support that feature in your XE version or above?
A: We still support TThread, yes.

Q: Are there any libraries etc that have to be deployed to Mac other than the bundle?
A: Not unless you add such requirements yourself. If you make a database client or a DataSnap client then you may need to add the relevant DbExpress driver dylibs and the DataSnap client dylib in the Deployment wizard. After that it's taken care of for you. These files get deployed into your app bundle.

Q: Could we see in the future how **TLang** works?
A: Yes, I can cover this in a later lesson. For now, check out the **TLang** component - http://docwiki.embarcadero.com/Libraries/en/FMX.Types.TLang.

Q: What is "AutoTranslate" property of the component TLabel?
A: The property tells FireMonkey runtime whether to look for translated text for the TLabel that is defined using the TLang component. See the DocWiki example at http://docwiki.embarcadero.com/CodeExamples/XE2/en/FMX.MultiLanguage_Sample

Q: When I create a Menu Item and add under it a sub-menu and try to add a sub-sub-menu , it does not display that menu. Any reason why?
A: I tried this using Delphi and C++Builder FireMonkey HD applications and I could create several levels of Child Menu items without any problem.

Q: Fire Monkey appears to parallel the VCL, simply a different set of components. Is this correct?
A: VCL is a component layer over the Windows API. FireMonkey is a business application platform that supports multiple operating systems and graphics rendering systems. While some of the components have the same names, the implementations are very different. VCL uses Windows to display your application UI. FireMonkey renders the application UI using scalable vector graphics in HD and 3D display modes. FireMonkey also takes advantage of the Graphics Processing Unit (GPU) when it can to render parts of your UI – especially for image effects and 3D.

Q: When i run my application in other computer, I get error for ""Missing something160.bpl file"". How i can import library files in my app?
A: You have the choice to either deploy the runtime packages you use, or turn off runtime packages.

Q: If I'm writing new applications, even if they are Windows only, should I do it as a FireMonkey app or stick with VCL?
A: FireMonkey is the business application platform of choice for building high performance, scalable vector graphics based application UIs that run on multiple platforms. VCL is tied directly to the Windows API. If you need Windows, Mac and mobile for the future of your application and you want to use one codebase, then FireMonkey is the right choice. Even if you only are targeting Windows today, FireMonkey's capabilities go way beyond the VCL.

Q: Is it possible so set up custom new projects groups so that one click creates the entire project group for custom components with the secondary Panel Test, Panel tester setup as shown in the wiki?
A: You can create your own projects and add them to the Object Gallery. Create your project and then in the project menu choose **Project > Add to Repository…** You can also build template libraries for custom projects - http://docwiki.embarcadero.com/RADStudio/en/Creating_Template_Libraries.

Q: I did a Hello World FireMonkey 3D app with a 3D label and ran it on Mac. The label text looks not anti-aliased and especially the text is distorted. Where can I find information on 3D so I can diagnose this problem?
A: We will cover building 3D applications using FireMonkey in Lesson 7. In the meantime you can look in the DocWiki for additional information
http://docwiki.embarcadero.com/RADStudio/en/Creating_the_User_Interface_(FireMonkey_3D_Tutorial)

If you have any additional questions – send me an email - davidi@embarcadero.com

Self Check Quiz

1) Which of the following are not views in the RAD Studio IDE?

a) Form Designer
b) Structure View
c) Window Washer
d) Object Inspector
e) Project Manager
f) Tool Palette

2) To change the caption of a TButton component in the Object Inspector, you change which property?

a) Caption
b) Text
c) Label

3) Which project node allows you to select the target operating system?

a) Build Configurations
b) Target Platform
c) Release
d) Operating System

Answers to the Self Check Quiz:

1c, 2b, 3b