Building a Mobile App with Delphi and FireMonkey for Experience Delphi & C++Builder developers.

Assumptions:

- Store images in DB Blob
- Create a report in HTML with images embedded using Data URI (see below)
- Use share sheet to share report

Agenda:

- Training Overview
    - Schedule
        - (Roughly 8 hours of training
    - Agenda
        - Goal
            - Help you get up to speed for mobile development with FireMonkey
            - This is a workshop - we are developing an app together
            - Expectations
                - Experienced with VCL & Delphi
                - Experience with Database development
                - Follow along with the exercises
            - Showing Delphi, but it will mostly work the same in C++Builder
    - Useful Information
        - There are many links to the DocWikis
            - http://docwiki.embarcadero.com/RADStudio/en/
            - http://docwiki.embarcadero.com/Libraries/en/
            - http://docwiki.embarcadero.com/CodeExamples/en/
        - Shortcuts on slides:
            - docwiki:RADStudio/FireMonkey_Platform_Services
            - Translates to http://docwiki.embarcadero.com/RADStudio/en/FireMonkey_Platform_Services
        - You have a copy of the slides and there are notes with more information and comments in the “speaker notes” section
    - App Specs
        - Project log collection application
        - Uses Embedded InterBase ToGo (or the free IBLite)
        - Database has projects with child log entries
            - Log entries include: DateTime, Picture, Geolocation, Orientation, Accelerometer, User notes
        - Screens
            - Edit project details
            - Add logs to project

            - Browse & edit projects
            - Browse & edit project log entries
            - Reporting
                - Export project with log entries as JSON or HTML
                - Save to file or share via email, etc.
- Introduction to FireMonkey
    - What is FireMonkey
        - FireMonkey is similar to VCL
            - Your VCL experience is applicable for FireMonkey
            - It is not a 1:1 mapping of the VCL
                - Eg: TLabel.Text instead of TLabel.Caption
            - Designed to be cross-platform:
                - iOS, Android, macOS, & Windows
                - Other platforms like Linux via 3rd parties
                - Cross platform is in its DNA
            - Still uses the RTL you know and love
            - FireMonkey also includes platform services and other non-visual components
            - Rendered by GPU
                - Uses DirectX on Windows
                - OpenGL on macOS
                - OpenGL ES on iOS & Android
            - Check out the Quick Start Guide
                - docwiki:RADStudio/en/FireMonkey_Quick_Start_Guide_-_Introduction
        - The FMX Form
            - Uses floating point numbers for positions, sizes, etc.
            - Supports animation and graphical effects
            - Very flexible controls
            - Many different layout options
            - All components are nestable
            - The FMX file is very similar to a VCL file
        - Understanding Platform Default Behavior
            - Many properties have an option of PlatformDefault value
                - This will change the value based on the platform
                - Tab Controls PlatformDefault property http://embt.co/tabtutorial
            - The Style can apply properties too based on platform
                - This is controlled with the StyledSettings property
                - Settings text parameters http://embt.co/SettingTextParameters
            - Change the ControlType property from Styled to Platform
                - Currently supporting iOS and Windows with Android coming soon

        - More information: http://embt.co/FMXNative
    - FMX Layouts
        -
    - FireMonkey Platform Services
        - A platform service is a FireMonkey interface that defines some functionality that might or might not be implemented on a particular run-time platform
            - Allows for different functionality and implementation per platform
        - FireMonkey implements many platform services
            - 52 services in 13 units
        - You can implement your own platform services
            - Use TPlatformServices.AddPlatformService and TPlatformServices.RemovePlatformService
            - For example, you can unregister one of the built-in platform services and replace it with a new implementation of the platform service that is tailored to fit your needs.
        - More information on Platform Services
            - http://embt.co/PlatformServices
- FireUI - Technology to Fine Tune Your UI
        - Device Views:
        - Multi-Device Preview: Gives you immediate preview of your UI on multiple platforms
        - FireUI LivePreview: View your UI on your physical device in real time
        - docwiki:RADStudio/en/FireUI_Live_Preview
    - Device Views
        - Allows you to add platform specific customized views to your layout
        - *Left Image: http://docwiki.embarcadero.com/images/RADStudio/Rio/e/0/06/ViewsDropDownMenu.png*
        - docwiki:RADStudio/en/Using_FireMonkey_Views
    - Multi-Device Preview
        - Accessible via: View > Tool Windows > Multi-Device Preview
        - *Image http://docwiki.embarcadero.com/images/RADStudio/Rio/e/7/7e/MDPreviewWindow1.png*
        - docwiki:/RADStudio/en/Multi-Device_Preview
- FMX Compared to VCL
    - Similarities
    - What are advantages?
    - What are limitations?
- Getting Started

        - Hello World on Windows
- Setup the Environment
    - General iOS vs. Android requirements
    - Downloading SDKs
    - Emulators
    - Introduction to Styles
    - Provisioning Apple devices
    - Basic architectures
        - Hello world on mobile
            - http://embt.co/Create1stApp
- App overview for the app we are building
    - Project log collection application
        - Uses Embedded InterBase ToGo (IBLite)
        - Database has projects with child log entries
            - Picture
            - Sensor information
                - Geolocation
                - Information about how the camera was facing when the photo was taken
                - http://docwiki.embarcadero.com/Libraries/en/System.Sensors.Components.TLocationSensor
                - http://docwiki.embarcadero.com/Libraries/en/System.Sensors.Components.TOrientationSensor
                - http://docwiki.embarcadero.com/Libraries/en/System.Sensors.Components.TMotionSensor
            - User notes
    - Screens
        - Edit project details
        - Add logs to project
        - Browse & edit projects
        - Browse & edit project log entries
    - Reporting
        - Export project with log entries as HTML file
            - Embedding
        - Export project as JSON
        - Share via Share Sheet
- Embedding InterBase
    - Creating InterBase database
        - [There are two tables: Projects and Log Entries, the latter has the image blob]
        - Also multiple users and the login will authenticate with InterBase
    - Using the deployment manager
    - LiveBindings
    - Just a simple example showing some data

- Setup Users and Login Screen [Reuse the Home Screen projects]
        - Home and Login Screens
            - Home Screen
            - Login Screen
        - Lab Exercise: Home and Login Screens
        - Multiple screens (Home Screen to Login Screen)
            - uHomeForm2.Form2Home.Hide; //Hide the Home Screen.
            - uLoginForm2.Form2Login.Show; //Show the Login Screen
        - Lab Exercise: Multiple screens
        - Authenticate user against InterBase
            - DataModule
            - FDConnectionIBLite.Params.Values['USER_NAME']
            - FDConnectionIBLite.Params.Values['Password']
            - FDConnectionIBLite.Connected := True;
        - Lab Exercise: Authenticate user against InterBase
    - Working with Styles
        - Working with Styles
        - Default Styles
        - Lab Exercise: Working with Default FMX Styles.
        - Resource Naming and Referencing
        - Style Resource Storage: Multi-Platform TStyleBook
        - Platform Styles
        - Custom Styles
        - Lab Exercise: StyleBook and Working with Custom Styles.
        - Nested Styles
        - Style-Resource Search Sequence
        - Form Style

    - App Navigation
        - TTabControl component
        - Lab Exercise: How to use Tab Components to Display Pages
        - .Show and .Hide methods
        - Glyph Buttons Arranged in a Grid Like Layout
        - Lab Exercise: Home Screen Navigation using Glyph Buttons
        - App Home Screen Navigation
- Build Data Capture Form
    - User input,
    - Keyboard,
    - Adding to database
- Sensors
    - Taking pictures
        - TCameraComponent
        - Taking a Picture with a Mobile Device Camera
        - Saving a Picture to the Device Photo Library

        - Using a Picture from the Mobile Device Photo Library
        - Sharing or Printing a Picture
    - Orientation
    - Location sensor
        - LocationSensor - Latitude and Longitude
        - Reverse Geocoding
        - Orientation Sensor - three-axis tilt, distance and heading,etc.
        - Accelerometer (Motion Sensor)-acceleration, angle, state, and speed of the device motion.
        - LAB Exercise: Create the Capture Data Form
    - Adding to database
- Build Data Output
    - Exporting and reporting
        - HTML with embedded images (see below)
        - JSON (with Base64 embedded images)
    - Share sheet
- Architecture Considerations
    - Android - Async dialogs, but don't bother getting into services.
    - iOS
    - How does Windows and macOS figure in?
        - Rapid prototype on Windows
- App Store Publishing
    - Google Play Store
    - Apple App Store

For reporting we will use the following to create a single HTML file with the image data embedded using Data URIs and Base64 encoding

………………………