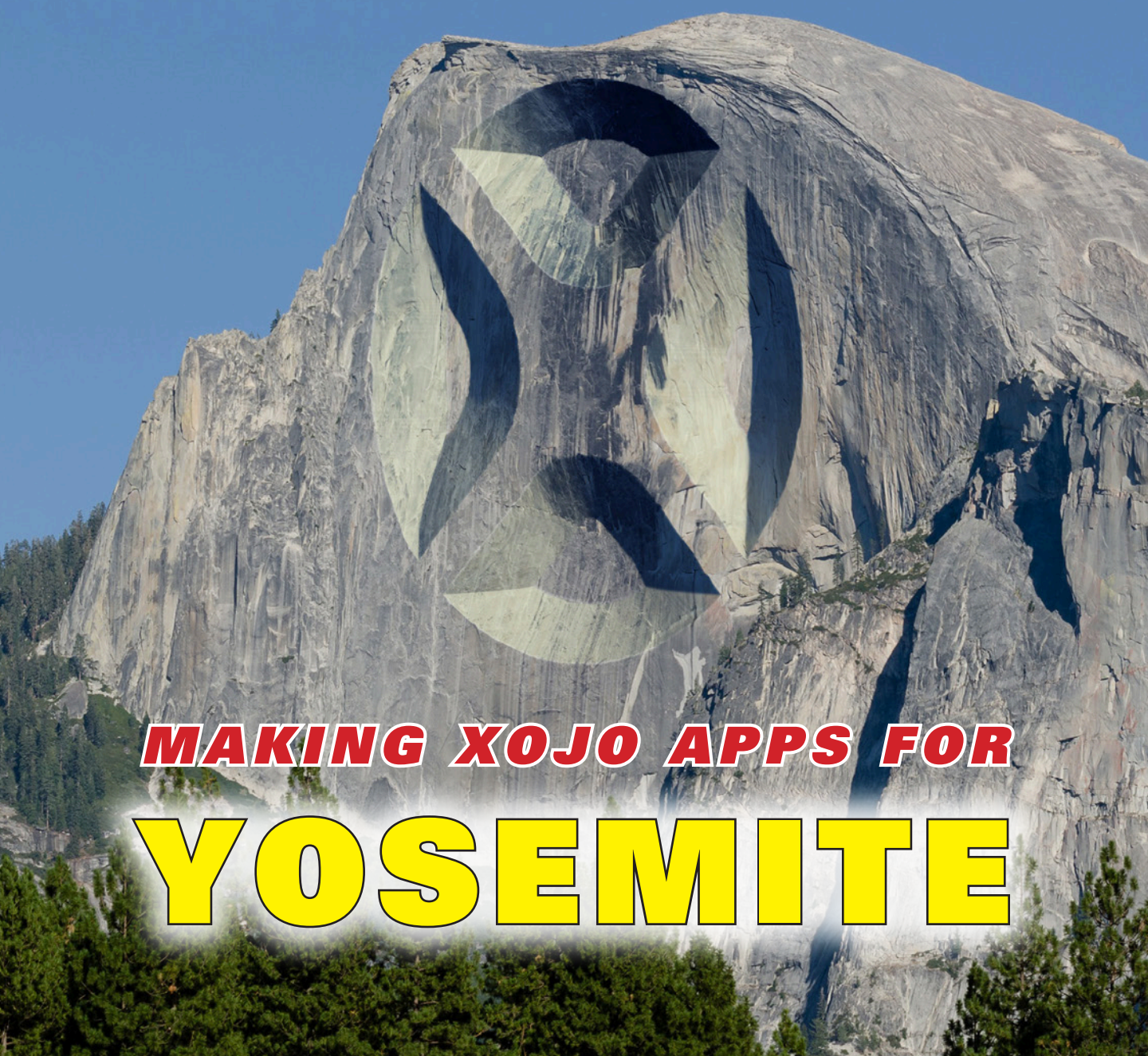
NOTIFICATIONS / INSIDE HandleSpecialURL / GAME CENTER / SIMPLEDRAW

SEPT/OCT2014
ISSUE 12.5

xdev

XDEVMAG.COM

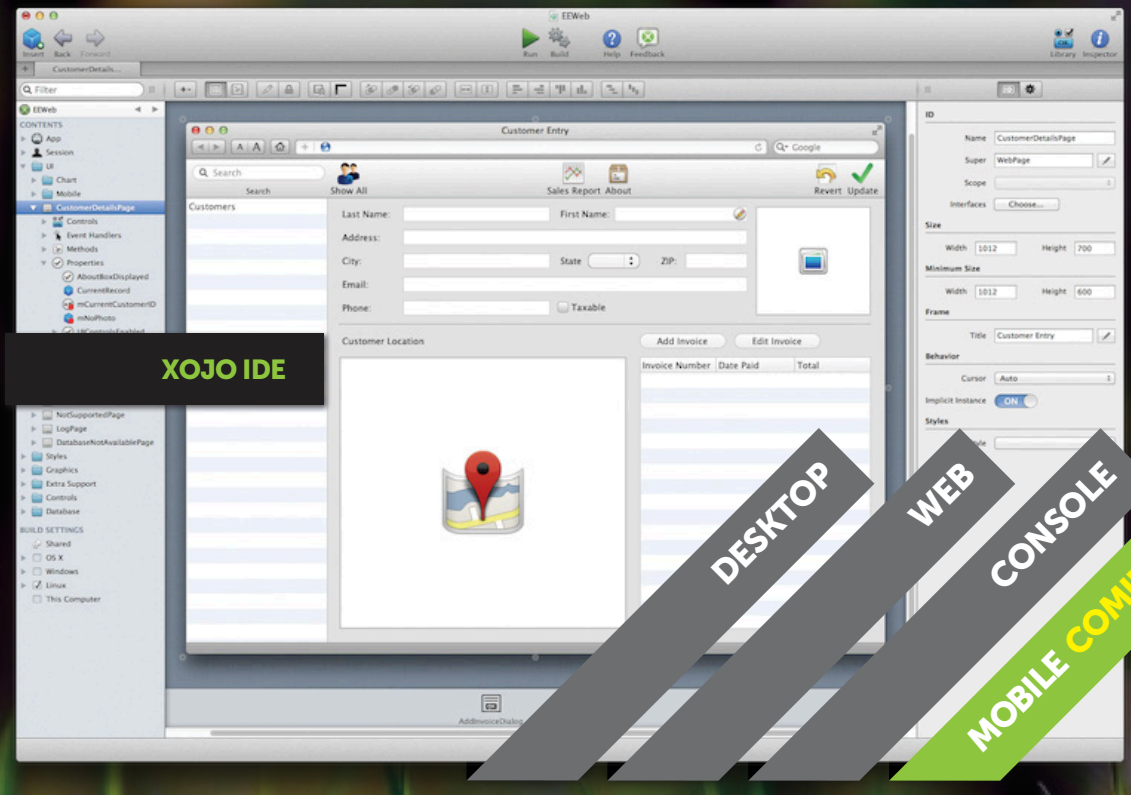
THE MAGAZINE FOR XOJO™ USERS



MAKING XOJO APPS FOR YOSEMITE

Contact us to upgrade today

Get Xojo Pro today and maximize your Xojo experience.



Xojo Pro is a special license package designed for professional developers. It is a single license for Desktop, Web, Database Access and Console, and includes priority support, guaranteed beta program access, a special Xojo Pro-only forum, 3x Feedback multiplier when ranking cases, consulting leads, and a license that will work on 3 machines. Contact custserv@xojo.com for a quote for your Xojo Pro upgrade.

XOJO™

www.xojo.com

Enabling ordinary people to create extraordinary apps™

SEPT/OCT2014
ISSUE 12.5

xdev

XDEVMAG.COM

THE MAGAZINE FOR XOJO™ USERS

FEATURES

xDev Magazine (ISSN 1540-3122) is the ultimate source for tutorials and advanced techniques for programming with the Xojo development environment. *xDev Magazine* is published bi-monthly (six times per year) by DesignWrite, PO Box 872, Lafayette, OR 97127. **Subscription Rate**: Annual subscriptions are $49.99 (http://www.xdevmag.comw/subscribe/). **Customer Service**: http://www.xdevmag.com/support.shtml (Address changes: http://www.xdevmag.com/addresschange.shtml). **Back Issues**: http://www.xdevmag.com/orders.shtml **Printed Issues**: Full color issues may be ordered at http://rbd.magcloud.com/ (Magcloud is a print-on-demand service).

SEPT/OCT2014
ISSUE 12.5

xdev

XDEVMAG.COM

THE MAGAZINE FOR XOJO™ USERS

COLUMNS

NEWS, REVIEWS, ETC.

ADVERTISER INDEX

by Marc Zeedar
publisher@xdevmag.com

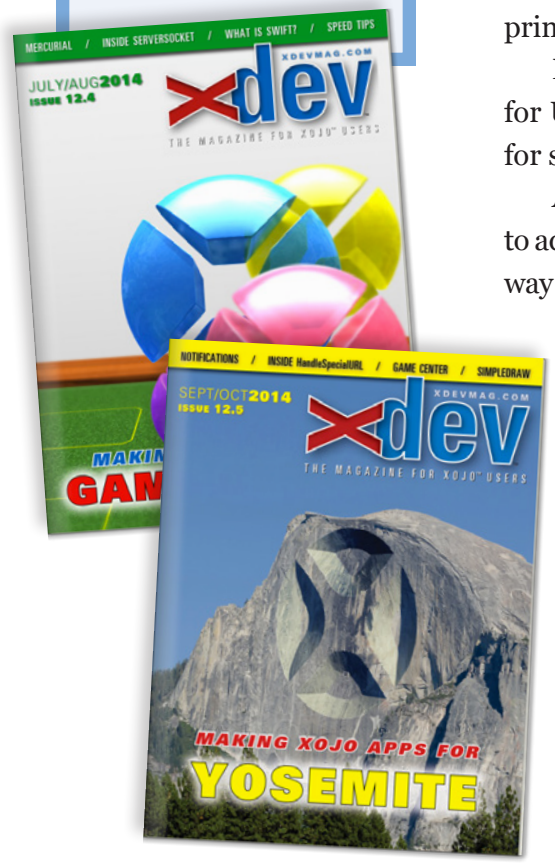
Print Is Back!

Thoughts from the Publisher

AT A GLANCE

XD#12500

About the Author:
When *xDev* publisher Marc Zeedar was a kid he used to create magazines just for fun. Now he's doing it for a living!



IT'S ironic, but right as people are getting used to digital magazines, I'm bringing back the print edition of *xDev*! Yet it is precisely because few people need print that this is economically feasible. In the old days I had to produce thousands of copies even if I didn't need that many. Print-on-demand is more expensive on a per-copy basis, but at least I can offer print subscriptions for those who prefer it. (Note that print subscribers will receive the PDF, too.)

I've tried to keep the cost as reasonable as possible: $100/year for U.S. and $125/year for international subscriptions. That's for six full-color 6"x8" booklets, including postage.

At the moment, I'm only offering annual subscriptions (I plan to add sales of individual issues soon). Also, there's no automatic way to upgrade digital subscriptions to print—if you're interested, email me and I'll let you know the cost and procedure.

In This Issue

Mac OS X Yosemite is coming, so Sam Rowlands has an article on making your Xojo apps work better with it. He's also got a terrific piece on creating an internal notification system for your apps. In our other features, Christian Schmitz explains Web Edition's HandleSpecialURL, and Tom Baumgartner is back with part 2 of his Game Center series.

Columns topics include my SimpleDraw project in *Beginner's Corner*, database design, coding standards, Regex conditionals, SQLite command line, a Xojo game developer, various tips, and so much more. Enjoy! xdev

XSCOPE FEEDBACK

[Regarding the XScope review in *xDev* 12.4]: I agree, it's expensive. But isn't the Text tool new? I cannot find it in my version 3.

Robert L

That's a good question. I left it ambiguous because I couldn't find info on their website. It made it sound like it was an update, which I thought was odd as I didn't remember it. (You'd think they'd promote it heavily as a new feature if it was.) I thought maybe they'd added Text via previous update and I had just never noticed (I don't use xScope that often and when I do, I usually just use the particular tool I need right then).

I do really like the Text tool, but I'm not sure it's worth the upgrade price on its own.

BITCOIN?

I'd love to purchase a few back issues of *xDev* and maybe subscribe. But I'd like to do that with Bitcoin. Do you accept it for payment?

Anthony P

Not at this time, but I'll certainly look into it. Are other readers interested in this form of payment? xdev

Send your Letters to the Editor to **publisher@xdevmag.com**. You must include your full name, but we will withhold publishing it on request. All letters may be edited for content or length, and become the property of *xDev Magazine*. If you don't wish your correspondance to be published, please let us know.

xdev
THE MAGAZINE FOR XOJO™ USERS
is published by DesignWrite,
P.O. Box 872, Lafayette, OR 97127-0872
and has no affiliation with Xojo, Inc.

PUBLISHER & EDITOR-IN-CHIEF
Marc Zeedar publisher@xdevmag.com

MANAGING EDITOR
Marc Zeedar editor@xdevmag.com

NEWS EDITOR
Marc Zeedar news@xdevmag.com

REVIEWS EDITOR
Dave Mancuso revieweditor@xdevmag.com

COPY EDITORS
Carol Van Wagner carolvw@xdevmag.com

ADVERTISING COORDINATOR
Marc Zeedar ads@xdevmag.com

WEBMASTER
Marc Zeedar webmaster@xdevmag.com

LAYOUT & DESIGN
Marc Zeedar mzeedar@xdevmag.com

COVER ART
Jeff Quan jquan@xdevmag.com
Yosemite photo:
Tuxyso / Wikimedia Commons / CC-BY-SA-3.0

ARTWORK
Dan Wilson (Caricatures) hacker@xdevmag.com

COLUMNISTS
Paul Lefebvre plefebvre@xdevmag.com
Craig Boyd cboyd@xdevmag.com
Seth Verrinder seth@xdevmag.com
Kem Tekinay kem@xdevmag.com
Markus Winter markus@xdevmag.com

How to Download Source Code

Article Resources: http://www.xdevmag.com/browse/

The "browse" website gives every *xDev* article a *permanent web page* where you can find links to downloads, updates and corrections, and more. Note that downloads include a modification date. Every article in *xDev Magazine* includes an "XD number" at the beginning presented like this: **XD#10234**

To retrieve an article's resources, follow these steps:

Step 1: Go to the *xDev* website (http://www.xdevmag.com/browse/search.shtml).

Step 2 (find article by number): In the **Get Article** search field type in the *XD number* of the article you need.

Alternate Step 2a (find article by issue): Find the issue you'd like to Browse and click on the link for that issue's Table of Contents.

Alternate Step 2b (download all resources): On an issue's Table of Contents page, there's a link to a file which includes *all the downloads* for a particular issue in one large archive.

HACKER by Dan Wilson <hacker@xdevmag.com>

XD#12501

XOJO EVENT IN BIRMINGHAM UK

The Xojo Developers UK User Group is pleased to announce a two-day event on Thursday 13th and Friday 14th of November 2014, at a new venue this year: the offices at 77 Paradise Circus, Queensway, Birmingham UK, B1 2DT.

On Thursday, there will be an iOS Beta Hands-On Workshop. Paul Lefebvre, the Xojo Evangelist, will give a demo via video chat and will go over the ins and outs of developing iOS apps using Xojo. That evening attendees are invited to Jimmy Spices, Regency Wharf, Broad Street, Birmingham B1 2DS (www.jimmyspices.co.uk). This is a great opportunity to meet other Xojo developers, to socialize, pick up tips, and find out what other developers do and how they do it.

On Friday there are Developer Presentations and Networking. Sessions will include Custom Control Creation, Security Via Obesity, Software Licensing, Version Control, JSON and a RESTful API, using ChartDirector and DynaPDF, and more. There will be a video chat with Xojo CEO Geoff Perlman -- get ready with your questions. At the end of the day there will be an open Q&A session to help you get the answers you are looking for. During the day there will be two Xojo T-shirts to be won in a free raffle!

Product: Xojo Event in Birmingham UK

Platforms: All

Price: £42 +VAT

Website: http://xojo-events-uk.eventbrite.co.uk

XOJO CLOUD NOW IN LONDON!

Xojo Cloud, the one-click app hosting service for Xojo web applications, is going global -- the London data center is now online. Xojo Cloud is secure, easy to set up and configure, and maintenance-free. Deploying your web app has never been so easy!

Product: Xojo Cloud

Platforms: All

Price: $49-$199/month

Website: http://xojo.com/cloud/

XOJO 2014 RELEASE 2.1 AVAILABLE

Xojo 2014 Release 2.1 is now available! This release has 31 improvements, including:

- Cocoa apps using RegEx are accepted by Mac App Store.
- Final version of Xojo with Carbon support.
- Fixes for Web regressions.

Product: Xojo 2014 Release 2

Platforms: All

Price: Free IDE (deployment support costs $100-$995)

Website: http://www.xojo.com/download

LAST CHANCE TO ATTEND MBS XOJO DEVELOPER CONFERENCE

There are just a couple of weeks left before the MBS Xojo Developer Conference September 17-20th in Koblenz, Germany. Hurry and register now before it's too late!

This is your chance to meet other Xojo developers from Europe. Get in touch, share ideas and learn new things. Already developers from 10 countries are registered. Sessions currently include:

- News about Xojo, including Xojo Cloud
- Custom Controls
- Helper Apps
- Advanced Xojo Features
- Extending the Framework

- Code Reuse
- CURL in detail
- In-App Purchases
- Client/Server Communication

Product: MBS Xojo Developer Conference

Platforms: All

Price: 299-399 Euro

Website:

http://www.monkeybreadsoftware.de/realbasic/events/koblenz-2014-event.shtml

MYSALESAGENT 2 RELEASED

mySalesAgent is a Mac-only App which makes sure you will never miss the Mac App Store sale you are waiting for again. It offers a clean, white iOS7-like interface in which the user can search for iOS Software, iPad-only Software, Mac Software, and eBooks available on Apples Stores. Just search for items which you want to purchase when they reach your targeted price. Set your price and forget about it. The mySalesAgent will inform you via OS X Notification Center Messages and/or via Email when an item reaches the targeted price.

Product: mySalesAgent 2

Platforms: OSX

Price: Free

Website: http://www.schneppi-software.de/

NEW PDF CLASSES FOR XOJO

BKeeney Software has introduced the first public beta release of BKS PDF Classes for Xojo. The PDF Classes are 100% native Xojo classes that allow developers to create PDF documents via code in their Xojo applications. The classes work for Mac OS X and Windows. Linux is currently unsupported.

The PDF Classes are for creating a PDF document. Others means must be used to display PDF documents from within a Xojo application.

Version 2.5.0 Beta 1 is the first public release of these classes. The classes cost $200 and contain the full source code. This introductory price is for a limited time only. Users that have a valid Fireye Software PDF Classes license can contact BKeeney Software at support at bkeeney dot com to receive a discount.

Product: Software

Platforms: OSX/Win

Price: $200 (introductory price)

Website: http://www.bkeeney.com/allproducts/pdf-classes/

GENERALLEDGER V1.0.5

GeneralLedger is an easy to use money management solution for home and small businesses from ToThePoint Software. Set up any number of Account Books. Create unlimited Expense and Income Accounts. Easily enter any type of transaction.

- Financial Tracking: simply track where the money goes.
- Budgeting: track income and expenses by accounts
- Accounting: allocate starting amounts for budget accounts

Product: GeneralLedger v1.0.5

Platforms: OSX

Price: $15

Website: http://ttpsoftware.com/thankgoodnessWP/generalledger/

NEW XOJO CANVAS BOOK PUBLISHED

Eugene Dakin has released a new Xojo book, *I Wish I Knew How To... Program the Canvas Control with Xojo Desktop*, that will provide you with the ability to learn how to modify pictures, graphics, make animation, and how to make two games.

The focus of this book is to work with topics related to the Canvas Control. All of these examples are created with native Xojo code and does not use any third party programs. The examples have been tested on Windows 7, 8.1, OS X 10.9, and Ubuntu 14.04 Operating Systems. By the time you finish the book you will be able to apply skills from these examples to create your own programs!

Topics included in the book:

Text, Chart Fundamentals, Objects, 2D Objects, Graphics, Blurring, Cropping, Gaussian Blur, Building basic controls, Animation, Sprites, Two games with step-by-step code explanations to help you build your own, and more...

The book includes twelve chapters and over 400 pages of code to build 50 example programs. Create realistic 2D game motion. It also includes example images and the ways to use them in programs.

Product: *I Wish I Knew How To... Program the Canvas Control with Xojo Desktop*

Platforms: All

Price: $8.99

Website: http://great-white-software.com/rblibrary/

BACKUP TO GO FROM OHANAWARE

Don't leave the safety of your important files to chance. What if your Mac gets damaged, or stolen? Hard drives fail, even backup drives can fail, especially when needed most.

Replacing a computer is one thing, but replacing your photos*, projects, web layouts, thesis, book drafts, artwork, graphic designs, app ideas and many other kinds of irreplaceable files and data, can be impossible. Unless you have adequate protection.

Everyone who owns a computer needs to think about multiple backup solutions. Before it's too late, and your digital belongings are lost forever.

Backup To Go is a must have utility, it adds the much needed extra level of protection. Making it ridiculously easy to take and keep your files safely with you, all the time.

Simply plug in a USB Stick, Thumbdrive, SD card or an external Hard Drive, and configure it with Backup To Go. Then Every time the disk is plugged into the Mac, Backup To Go will automatically copy the latest files and changes to the portable backup disk.

Many USB Sticks or Thumbdrives have keychain loops, SD cards are small enough to fit in a wallet, while external Hard Drives can fit in a purse or stay in the car. The important point is, there is an extra layer of protection in case the worst does happen.

Download Backup To Go today, and give your digital belongings the much needed protection they deserve.

Key Features of Backup To Go:

- Automatically backs up the latest changes and files, when the backup disk is plugged in.
- Automatically ejects the disk on completion, so it can safely be removed and taken with you*.
- Can automatically launch at the same time as your computer starts.
- Simple and straight forward Interface.
- Menubar icon for convenient access.

DATE	CITY	WHO	WHAT	MORE INFO
Sept. 2, 2014	1-2 PM ET	Getting to know Git	Webinar	http://www.xojo.com/support/webinar.php
Sept. 9, 2014	1-2 PM ET	Xojo Talk: with Bob Keeney	Webinar	http://www.xojo.com/support/webinar.php
Sept. 17-20, 2014	Germany	MBS Xojo Developer Conference	Conference	http://www.monkeybreadsoftware.de/realbasic/events/
Sept. 30, 2014	1-2 PM ET	Using Introspection	Webinar	http://www.xojo.com/support/webinar.php
Nov. 17-18	UK	Xojo Developers UK User Group	Conference	http://xojo-events-uk.eventbrite.co.uk

- Saves battery life, by using zero energy while idle.
- Retina ready.
- Supports multiple simultaneous backups, allowing different files to be copied to different disks, or the same files to different disks.
- Uses the fastest copy methods.

Product: Backup To Go 1.0

Platforms: OSX

Price: Free

Website: https://itunes.apple.com/us/app/backup-to-go-protect-your/id882272017?ls=1&mt=12

RAGE SOFTWARE RELEASES EVERWEB

RAGE Software announced an update to EverWeb, the easy to use website design and publishing tool for OS X. EverWeb aims to remove all the hurdles from designing, creating, and publishing professional websites with a complete drag and drop user interface for website creation combined with a one-click publishing solution. EverWeb users never have to write any code or focus on any of the technical details when designing a website. EverWeb 1.4 adds major enhancements to the Image Gallery, Navigation Menu, Image Slider, Facebook, and HTML Snippet widgets.

Starting from a professionally designed template, or from a blank canvas, users can design a website with modern features including; drop down menus, image galleries, e-commerce stores, social networking features and more. Under the hood, EverWeb creates fast loading, HTML5 and CSS3 websites that are compatible with all desktop and mobile web browsers.

Product: EverWeb 1.4

Platforms: OS X

Price: $80-$100

Website: http://everwebapp.com/

HOSTING A XOJO EVENT?

Let us know so we can add it to our events calendar!

news@xdevmag.com

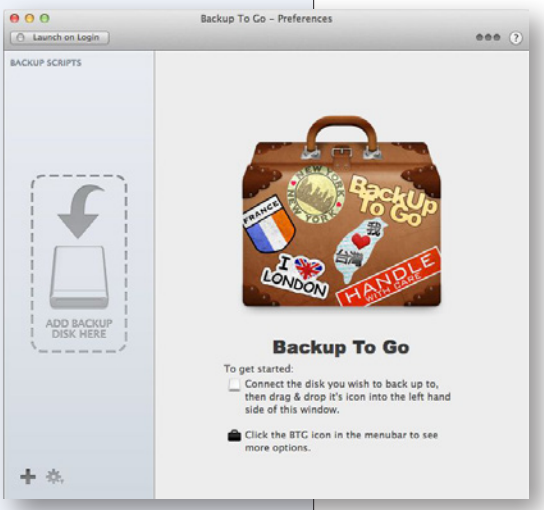
IN BRIEF

Product:
Backup To Go

Manufacturer:
Ohanaware

Price:
Free

Contact Info:
https://itunes.apple.com/us/app/backup-to-go-protect-your/id882272017?mt=12



XD#12502

Profile: Backup To Go

Marc Zeedar

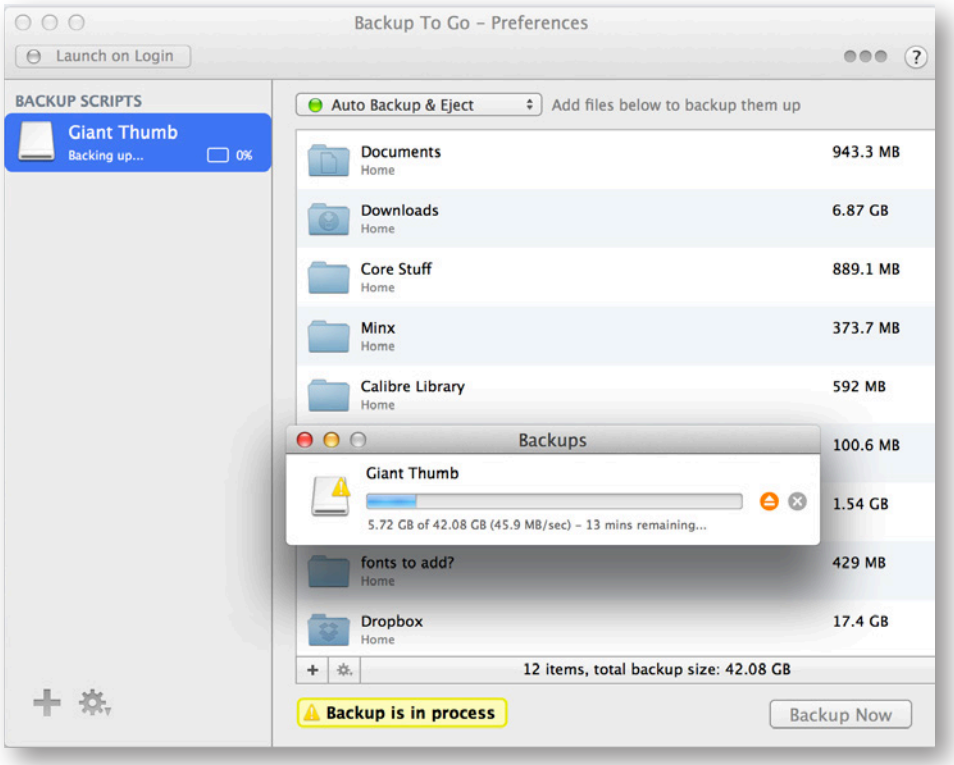
WE all know we should backup our files more often, but it's such a chore. There's a real fine line between making a backup program simple and easy to use, but still powerful enough to be useful. Ohanaware's done a neat job with their free Mac App Store offering, Backup To Go, which was, of course, created with Xojo.

Backup To Go is based on a simple premise of plugging in a drive of some kind (external drive, USB stick, SD card) and having a backup process run automatically. What's neat is you can specify exactly which files are backed up, so you don't have to backup your entire computer. That way you could create different backup strategies for different disks.

For example, one thumb drive could create a quick backup of all your most precious work files, while another might backup your applications. The work one you might plug in at the end of every day, for a daily backup, while the other you could do once a week or once a month (since applications aren't as critical as you can restore them in other ways).

Backup To Go runs as a menubar application (no icon on the Dock). Once launched, it watches for external drives that mount. You can select each drive from the list on the left of the preferences window and fill the right side with folders and files to back up (just drag them in or use the selection button). This works great for folders: I have a folder called Projects where I store most of my Xojo development, so each time I put that disk back into my computer, those items are backed up automatically.

The design of the app is extremely simple, and other than a few typos, I have few complaints. I really like the concept, especially the way I can set different disks to backup different parts of my Mac—I love



being able to use extra thumb drives and SD disks for backup and have each focus on a different kind of data. I'm not sure if the free price is a limited time offer, but it's certainly a great deal right now.

I did run into some errors during my backup tests—the app informed me that 479 files weren't copied. Finding the log of these errors wasn't the most intuitive (you click on a specific spot on the progress bar to open the log) and the log file seems overly technical for the uninitiated. In my case, all of the errors but one were due to permissions problems (it was trying to copy some funky low-level `/Library/` files), so it would have been nice to just see those grouped by error category ("Permissions errors"). The other file that didn't copy was the screenshot I made of the app working—I moved it from my desktop to this article's folder before the backup was finished and that generated a "file not found" error!

Beyond these minor quibbles, Backup To Go is impressive. Check it out and support your fellow Xojo developers. xdev

IN BRIEF

Product:
Grafio - Diagrams and Ideas

Manufacturer:
TenTouch Ltd.

Price:
$8.99

Contact Info:
https://itunes.apple.com/us/app/grafio-diagrams-ideas/id382418196?mt=8

Pros:
Nice interface; super easy to use; genuinely useful for much more than diagraming; excellent exporting capabilities; shape and symbol libraries are of great quality.

Cons:
Not as full-featured as a desktop app; a few bugs with the grouping feature; some shape and symbol libraries are locked unless you pay an additional fee.

Rating (1.0-5.0): 4.5

XD#12503

Grafio for iOS

Marc Zeedar

I'VE explored a number of diagraming apps for my iPad, but usually come away feeling frustrated. With all the limitations, it's more work to create such documents on an iPad than on my Mac. Grafio, however, is a pleasant surprise.

No, it's not as powerful as a desktop application, but it gives you 90% of the features you actually use and need. For example, I couldn't find a way to adjust a gradient fill—you can only use the pre-defined choices. But that's just fine, really.

Best of all, barring a few nitpicks, Grafio is so easy to use that it makes a chore fun. I can actually see myself choosing to use Grafio over my Mac which is right next to me simply because Grafio is more relaxing.

Another key feature is that Grafio isn't just for traditional diagrams (such as a flow chart). Since it includes enough graphical capabilities (as well as a large symbol library, mostly available via an in-app purchase), you can use it to create user interface designs, website mockups, presentation graphics, and much more. It's even really easy to use Grafio to create a quick logo. (Just grab a shape such as a shield or starburst, color it, add some text, and place an icon in the middle. Instant logo.)

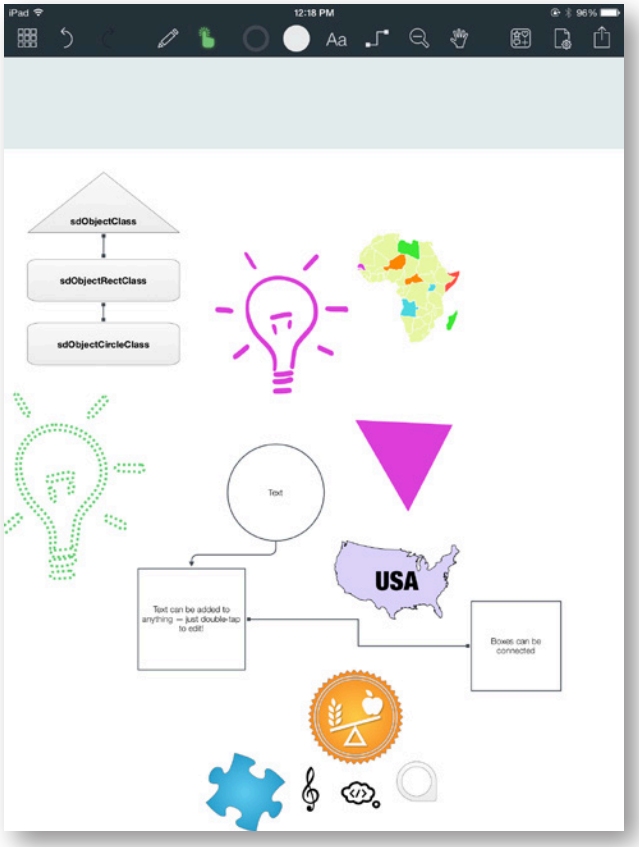
Grafio supports automatic shape recognition, so, supposedly, you can draw a rough box or circle and it will turn it into a perfect object. But either it's not that functional or I'm a terrible artist, because often when I drew a simple box it would stay a drawing and not convert it. Sometimes it worked and sometimes it didn't; I never could figure out why. But when it does work, the feature is really nice.

You can also add connecting lines and arrows between objects just by drawing a line with your finger between the two. At first I was frustrated because I couldn't figure out how to copy an existing line: I didn't like the way the default line would revert away from my custom-

ized version. But then I discovered that you can set your own defaults by simply tapping on the line settings at the top of the screen while no objects are selected and, after that, all new connecting lines you draw take on those characteristics. Nice. I am really pleased by all the choices for connecting, too: from angled lines to curves, simple to complex. It's your choice.

Despite its power, the app is amazingly simple. There are a few common gestures for enlarging, rotating, and zooming, and the simple menu at the top handles everything else. It's really nicely done.

Grafio comes with a library of a few shapes and drawings, but you can unlock more via in-app. Individual sets are 99 cents, but you can unlock all of them for $4.99. The app promises that if you do the latter, you'll get any new images added for free!



The quality of the libraries is excellent, however. (A lot of these kinds of apps use really cheap and cheesy clipart style graphics.) I was really glad to see that the maps can be ungrouped so you can break out individual countries or states. One flaw I noticed, however, is that if you group something, the entire group takes on a single characteristic. For example, when I selected a single country on the map of Africa and made it red, the red went away when I grouped the map again. Not a huge deal as you can just leave the object ungrouped to preserve individual characteristics, but hopefully that is a bug that will be fixed in the future. (Traditionally grouping should not change an object's appearance.)

When you're finished with your diagram, you can export it in several formats (PDF, PNG with transparency, JPG, etc.), either to your camera roll, email, or another app. Grafio also supports directly uploading to services such as Dropbox.

The bottom line is that Grafio is an excellent diagraming app. It's missing a few high-end features (such as the ability to customize gradients), but that's okay. I'd rather have the app simpler and easier to use. The less creating diagrams feels like real work, the more likely I am to use this app and do more work on my iPad. I suspect you'll see Grafio diagrams in *xDev* in the future! xdev

Feature

by Sam Rowlands

In-App Notifications

A system where your code can talk to itself

AT A GLANCE

XD#12504

Target Reader:
Intermediate

Source Code:
Yes

About the Author:
Sam is the codemancer behind Ohanaware Software.

TODAY I'm going to talk about in-app notifications. I don't mean user interface dialogs that pop-up to say something has been done, or the ones that appear when you've received a message. I'm going to explain an internal communication process that's used with Objective-C applications, and how this technique can be used to improve Xojo-made apps. It works wonderfully with threads!

In the diagram shown in Figure 1, you can see that I have three objects on a window: a thread, progress bar, and a timer. Now in order for the thread to update the progress bar, there has to be a timer which regularly polls the thread to find out its progress, and then it updates the progress bar. If the thread tries to directly update the progress bar, that will raise an exception.

How Do Notifications Help?

In Figure 2, you can see that the timer is gone, and instead we have a `NotificationManager` object. The notification manager acts as a bridge between the thread and the progress bar. The progress bar

registers itself with the NotificationManager as a listener, then any time the Thread sends a message to the NotificationManager, the message then gets forwarded to the Progress Bar.

The real power comes when you want to expand the process. For instance, you may have another window with another progress bar or even some code in the app that updates the dock icon. By also registering these objects for the messages, they'll automagically receive them when the thread sends the messages!

By removing the direct connection from the thread and the progress bar, we can easily get other objects to listen for the message, and if those objects no longer exist (say you close the second window) they no longer receive the messages—and you don't even have to write any extra code to handle any errors.

A real world example of this is my "Backup To Go" application. BTG uses threads to do the backing up. In the backup window there is a progress bar, and in the preferences window it displays if a backup is in progress or not (it will display the progress in a future version), and the same for the status item. Now the prefs window and the status item don't have any code written to "check" the thread and the thread has no idea about *any* of the interface elements. Instead, it simply sends messages to the notification center, which in turn forwards the messages to all registered objects.

Building the Notification System

We'll start by adding a class interface called owNotificationListener and we'll add one method. (I chose owNotificationListener, with ow meaning *Ohanaware*, but you can use what you like.)

```
notificationReceived(key as string, value as variant, sender as variant)
```

This method takes 3 variables:

- key - this is used to define the unique message that's being sent.
- value - what's the value (for the thread it was the progress).
- sender - the object that sent the message. Use this to determine where the message came from.

Next, add a module to the project called owNotificationCenter, and add the following properties:

```
Private dispatchQueue(-1) As Pair
Private dispatchTimer As timer
Private notificationSorter As dictionary
```

- dispatchQueue is where we'll hold messages until they're sent.
- dispatchTimer is what's responsible for sending the messages (and making it thread safe).
- notificationSorter is where we store which objects are registered to listen to which messages.

Now we'll add the register method to the module:

```
Protected Sub register(key as string, inObject as OwnotificationListener)
```

```
  if notificationSorter = nil then notificationSorter = new dictionary
  Dim c as collection = notificationsorter.lookup( key, new collection )
  Dim n,l as integer
  Dim w as new weakRef( inObject )
  n = c.count
  for l=1 to n
   if c.item( l ) = w then
    break
    return
   end if
  next
  c.add w
  notificationSorter.value( key ) = c
End Sub
```

If you look at the code, you can see the first thing it does is to check and make sure that our dictionary is not `nil`.

Then it tries to retrieve the storage for the key. If there is none, it creates one. The storage is a simple collection object, so that we easily store different objects in a single property.

Next, the code checks to make sure that this object isn't already registered (we wouldn't want it receiving multiple messages).

Lastly, we add our object to the collection and stuff it back into the dictionary.

> Note: we're using WeakRefs instead of the actual objects. This is so that our notification center doesn't keep objects in memory when other parts of the program have finished with them.

To handle messages, we'll add a `timerFired` method to the module. You see we're going to use a timer delegate (to cut down on classes and spaghetti code). When a message is sent, the timer will instead call this method in the module, rather than its own action event.

```
Private Sub timerFired(obj as timer)
  Dim receivers as collection
  Dim currentPost as Pair
  Dim n, l, deliveredCount as integer
  Dim w as weakRef
  Dim key as string
  Dim value, sender as variant
  while ubound( dispatchQueue ) > -1
   currentPost = dispatchQueue( 0 )
   key = currentPost.left
```

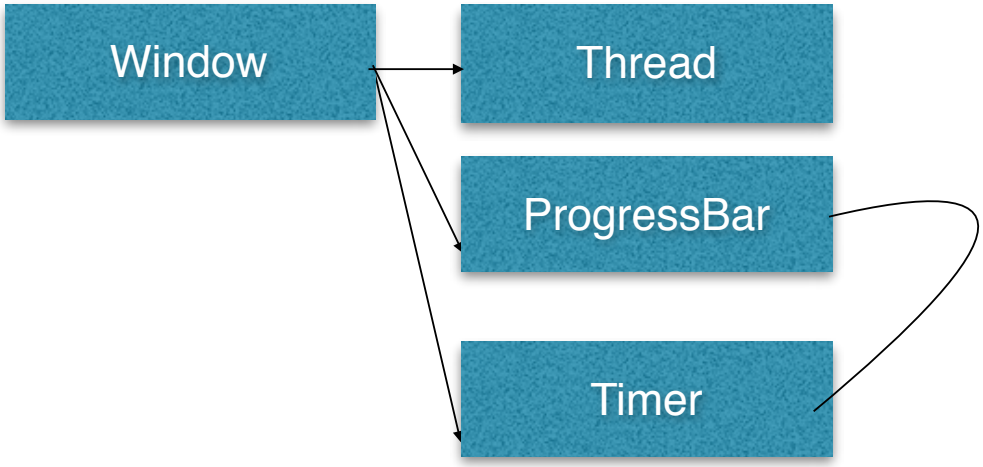


Figure 1: *Diagram of existing model*

```
value = pair( currentPost.right ).left
sender = pair( currentPost.right ).right
dispatchQueue.remove 0
deliveredCount = 0
receivers = notificationSorter.lookup( currentPost.left, nil )
if receivers = nil then
  #if debugBuild then
    system.DebugLog currentMethodName+" there are no receivers for the key:"+currentPost.left
  #endif
else
  n = receivers.count
  for l=1 to n
    w = receivers.item( l )
    if w <> nil and w.value <> nil then
      owNotificationListener( w.value ).notificationReceived( key, value, sender )
      deliveredCount = deliveredCount + 1
    end if
  next
  #if DebugBuild then
    if deliveredCount <> n then system.debugLog currentMethodName+" out of "+str( n )+" receivers, only "+str(
      deliveredCount )+" were valid"
```

```
    #endif
   end if
  wend
  obj.mode = 0
End Sub
```

This is probably the most complex that this code is going to get. The basic principle is we loop through the dispatchQueue, find out what is listening for that message, and then forward the message on to that object. Providing, of course, that object is still valid. You'll see I've added in some debugging code here. That will help track down any errors when you implement this into your application.

The last thing we do for this method is to make sure that the timer's mode is set back to zero (old habits).

Now we'll introduce the code that "sends" the message. In reality it simply queues it up and fires the timer (if it's not already underway).

```
Protected Sub send(key as string, value as variant = nil, sender as variant = nil)
  if notificationSorter <> nil then
   Dim p as pair = value : sender
   dispatchQueue.append key : p
   if dispatchTimer = nil then
```

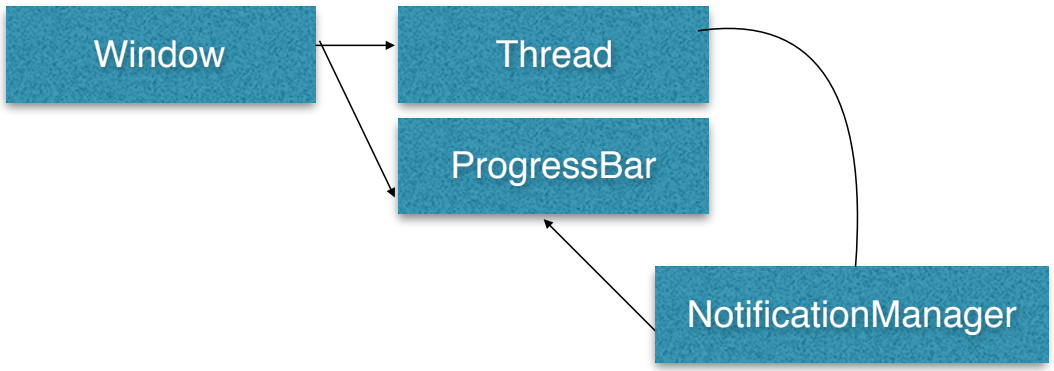


Figure 2: *Diagram of notification method*

```
    dispatchTimer = new Timer
    dispatchTimer.period = 1
    AddHandler dispatchTimer.action, AddressOf timerFired
  end if
  if dispatchTimer.mode <> 1 then dispatchTimer.mode = 1
 end if
End Sub
```

Example Application

On the default window of a new project, add a `Textfield` and a `Label`.

In the `TextChanged` event of the `Textfield`, add the following code to send a message:

```
owNotificationCenter.send "textFieldHasChanged", me.text, me
```

In this example, we're sending the message "textFieldHasChanged" with the value of the `Textfield` and the actual `Textfield` as the sender.

Select the window, and then in the window editor click the "Choose" button next to *interfaces*, and select `owNotificationListener`. After you click "OK", you'll see a new method has automagically been added to the window (see Figure 3).

In this new method, add the following code:

```
 select case key
 case "textFieldHasChanged"
  Label1.text = value.stringValue
```

```
end select
```

What the code does here is to compare the key and if the key is `textFieldHasChanged`, it then updates the label.

Select the window again and add the `Open` event. Insert the following line so that the Window registers itself to receive the `textFieldHasChanged` messages:

```
owNotificationCenter.register( "textFieldHasChanged", self )
```

Run the demo and start typing. I bet you're thinking, "What's to stop me from putting `label1.text = me.text` into the `textChanged` event of the textField?"

Nothing. However, doing it that way requires that objects know about each other. Using a notification system means that the objects don't have to know anything about each other and you don't need to write a bucket load of bridge code to handle that relationship. Just to prove this, duplicate the window in the project, and then add a button on the first `Window1` to show `Window11`.

Run the project, click the button and start typing in any text field... notice how both labels get updated? Pretty slick, huh?

Alright, now let's neaten up the functionality. First, we'll add a "stop listening" function. This way when an object is destroyed it can tell the notification center to clean up accordingly. In the `owNotificationCenter` module, add the following method:

```
Sub stopListening(extends onl as OwNotificationListener)
 if notificationSorter <> nil then
  Dim dCount, dPos as integer
  Dim currentKey as string
  Dim oCount, oPos as integer
  Dim receivers as collection
  Dim w as weakRef
  dcount = notificationSorter.count-1
  for dPos=dcount downto 0
   currentKey = notificationsorter.key( dPos )
   receivers = notificationSorter.value( currentKey )
   if receivers = nil then
    #if debugBuild then
     break
     system.DebugLog currentMethodName+" there are no receivers for "+currentKey
    #endif
    notificationSorter.remove currentKey
   else
    oCount = receivers.count
    for oPos = oCount downto 1
```

```
        w = receivers.item( opos )
        if w <> nil and w.value = onl then
          receivers.remove oPos
        elseif receivers.item( opos ) = nil then
          receivers.remove oPos
        end if
       next
       notificationSorter.value( currentKey ) = receivers
      end if
    next
  else
    #if debugBuild then
      break
      system.DebugLog currentMethodName+" notification center is not set-up"
    #endif
  end if
End Sub
```

Another complicated method, but all it does is step through all the registered notifications and objects until it finds the one we're removing. It also cleans up any objects that we didn't remove properly.

Last but not least, let's add a another neat function, `listen` as an extension of the class interface. We'll also make it so that it can take an array of things to listen to. This way you can simply register a control to start listening and pass in an array of messages to listen for. Again, we'll add this to the module.

```
Sub listen(extends onl as owNotificationListener, keys() as string)
  Dim n,l as integer
  n = ubound( keys )
  for l=0 to n
    register( keys( l ), onl )
  next
End Sub
```

Back into the `Window1` and `Window11`, change their `Open` events to read:

```
me.listen( array( "textFieldHasChanged" ) )
```

Add `Close` events with the following code:

```
me.stopListening
```

So, now when the window is closed, it will no longer be registered to listen to that event!

Taking it further

We've added code to a window where the window knows about the controls, but what about subclassed controls? With those there is no "governor" to direct messages—it's simply objects sending and receiving.

Insert a new class to the project. Name it `colorCheckbox`, and set its super to `Checkbox`. Add the `Action` event and insert the following line of code:

```
owNotificationCenter.send "colorCheckBox.valueChanged", true, me
```

When the checkbox is clicked, it will automatically send the `colorCheckBox.valueChanged` message!

Insert another new class, name it `colorPanel` and set its super to `Canvas`. Click on the "Choose..." button next to the "Interfaces" section and select `owNotificationListener`. In the auto-added `notificationReceived` method, enter the following code:

```
select case key
case "colorCheckBox.valueChanged"
  if checkbox( sender ).value then
    colorValue = &cFF000000
  else
    colorValue = &c0000FF00
```

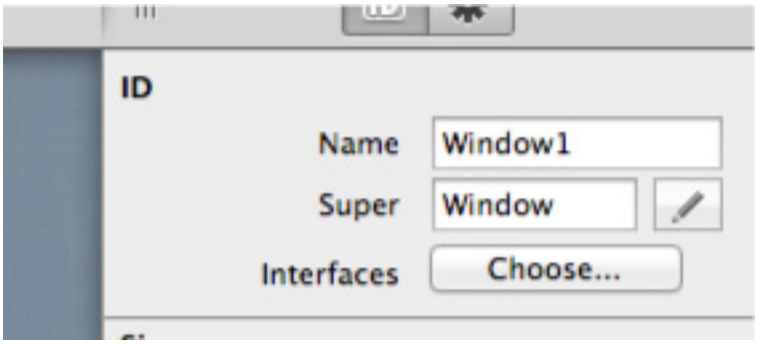


Figure 3: *Window Properties Panel*

```
  end if
  invalidate
end select
```

In this code, we'll take advantage of the `sender` property, to connect directly to the checkbox and then collect the checked status.

Now add a property `colorValue` of type `Color` to the colorPanel class. We'll use this to store the `colorValue` that we should be displaying.

Add the `Open` event to our `colorPanel` subclass and set the new class to listen for the checkbox's `colorCheckBox.valueChanged` notification:

```
me.listen array( "colorCheckBox.valueChanged" )
```

Lastly, for this class, add the "Paint" event, and get it to display something using the `colorValue`:

```
g.foreColor = colorValue
g.fillOval 0, 0, g.width, g.height
```

Now, drag the checkbox subclass onto `Window1`, and then drag the `colorPanel` subclass onto `Window11`.

When you run the project and click the box, nothing happens. Yet, if you click the "Button" to display the second window, then toggle the checkbox, the canvas subclass will update automagically.

This notification system is brilliant technology and one I'm sure you'll already be thinking of several uses for. We've been implementing it into everything we do, as it not only makes linking between objects neater, it makes the app so much more flexible. Now go to work!

by Tom Baumgartner

Writing OS X Apps for Game Center

Part 2: Requesting a Match

LAST issue described setting up your accounts with Apple so your game can interact with the Game Center (Sandboxed) testing facility. The steps are:

- Become a member of the Mac Developer Program
- Create and download the certificates from your member account
- In your member account register the Mac computers to be used in your testing
- Register your game application in iTunes Connect
- Create unique testing users in iTunes Connect
- Run Apple's Game Center on two test computers and request that your test users become friends
- I presented a simple program that talks to Game Center to authenticate the local user (localPlayer). The important steps are:
- Use Cocoa (note that Xojo has now announced the impending deprecation of Carbon)

AT A GLANCE

XD#12505

Target Reader:
Intermediate

Source Code:
Yes

RS Version Required:
2014r1+

Platform Supported:
Mac OS X

About the Author:
Tom is retired and enjoys programming in Xojo (at XDC 2014 he was one of the attendees recognized for 15 years of using RealBasic/Xojo). Along with curling, Xojo fills the Canadian winter when Tom can't golf, fish, or play tennis.

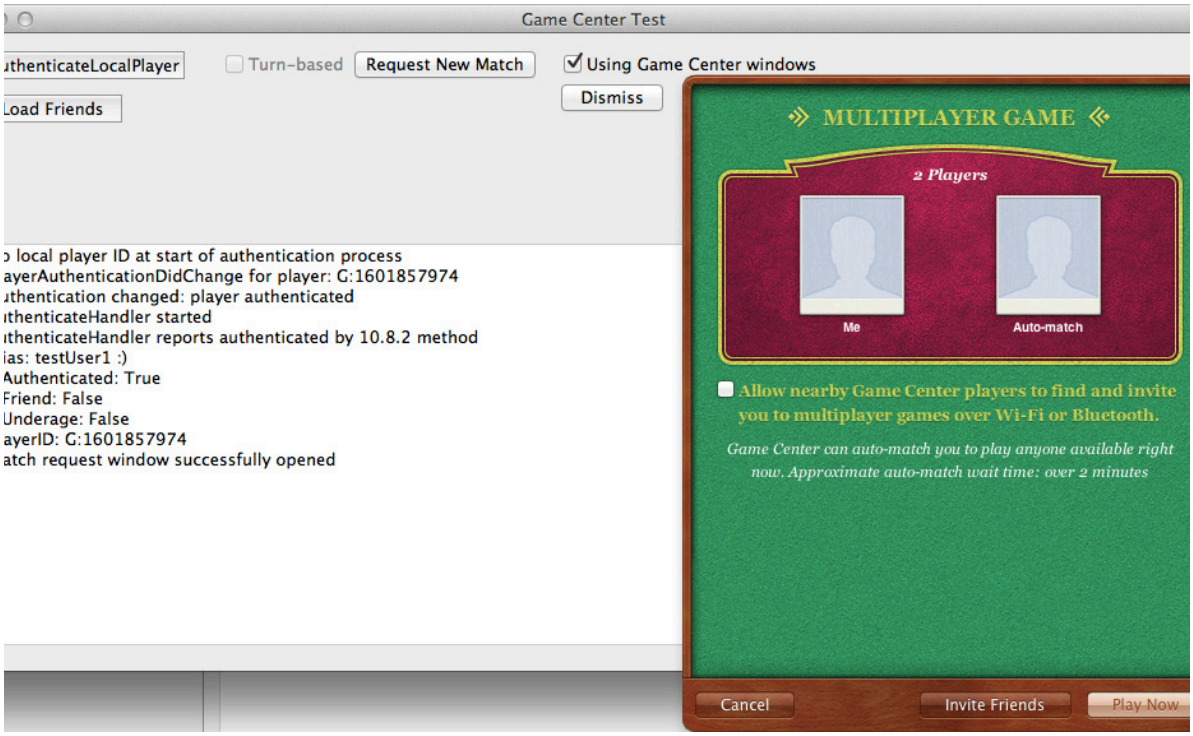


Figure 1: *Match request window opened*

- Use the exact same Bundle Identifier as entered into iTunes Connect
- Put the required Monkeybread plugins into the Plugins folder
- Create and import externally an App Wrapper Mini script to codesign both your debug and built applications (codesigning required to interact with Game Center)

Remember, when testing any app interaction with Game Center that there are communications between your computer and an Apple server somewhere in the world. The communications link can be unreliable and the server can be slow to respond. While working with the Game Center (Sandboxed) server I have experienced it working one hour and then not responding the next hour. I suspect the test facility is not as robust as the actual Game Center. Be patient and test repeatedly before you question your code.

That brings up one other point. I hope it is clear that the Game Center (Sandboxed) server has nothing to do with sandboxing of your app on your computer. The Game Center (Sandboxed) is a Game Center testing facility which isolates your game testing from the actual Game Center used by released applications. In a future article we will cover sandboxing of your game project which is required for the Mac App Store and is a recommended security feature for all Mac applications.

Figure 2: *Sending match request to a friend*

Requesting a Real-Time Match

This article will cover requesting a match (game). Although I like turn-based games, I have chosen to start with real-time games since that process seems more straightforward. For this reason, the example project has the turn-based checkbox disabled.

Note that we will not actually pass game information between computers this time because I still don't have that step working. Once again I want to thank Sam Rowlands of Ohanaware and Christian Schmitz of Monkeybread Software for their continuing help.

In all of your test runs, remember to first authenticate the local user by clicking the "AuthenticateLocalPlayer" button. In the example project, the "Request New Match" button will not be enabled until the user is authenticated.

You have two options when requesting a match—you can use the standard window provided by Game Center or create your own user interface. Let's start by using the standard Game Center window.

Request Match Using Game Center Window

Check the "Using Game Center windows" checkbox then click the "Request New Match" button. The pictures that follow are from the example project you can download. Figure 1 shows the Game Center window which results. In this window click the "Invite Friends" button which shows the window in Figure 2(a). Check the friend you want to invite and click the "Next" button which shows the window in Figure 2(b). Enter the text you want to send to your friend and click the "Send" button. This results in the "Starting Game" window of Figure 2(c).

The Action code for `pbRequestMatch` button is simple.

```
if NOT app.pBoolGameCenterIsAvailable then
```

```
  Log "Game Center is not available; your button push will be ignored"
  exit
end if
//request real-time or turn-based match based on cbMatchType checkbox
if cbMatchType.Value then
  TurnBasedMatchReq
else
  RealTimeMatchReq
end if
```

The `RealTimeMatchReq` method called above is shown in Code Listing 1. It creates a new GKMatchRequestMBS and sets its number of players to 2. If using the Game Center window, a `viewController` and a `matchmakerViewController` are created. Then the `viewController` presents the `matchmakerViewController`. The `viewController` object is saved as a property so the window can be closed later. From that point on, the user interacts with the Game Center window as described above.

Answer Match Request Using Game Center Windows

At the receiving end of the match request the player will get a banner notification. Clicking on the banner will start the game app if it is not currently running. The player will be asked to accept or decline the invitation as shown in Figure 3. This results in a "Starting Game" window similar to Figure 2(c).

The following code in MyGameKitMBS class `Invited` event implements this process.

```
Invited(MatchMaker as GKMatchmakerMBS, acceptedInvite as GKInviteMBS, playersToInvite() as string)
 Log "Got Invited event"
 //ask user if they want to play the game
 dim OK as Boolean = AcceptInvite
 If OK then
  if acceptedInvite <> nil then
   Log "received invitation from " + acceptedInvite.inviter
   if window1.cbInterface.Value then
    //user has selected to use the Game Center window
    dim viewController as new GKDialogControllerMBS
    dim mmvc as GKMatchmakerViewControllerMBS
    mmvc = new GKMatchmakerViewControllerMBS(acceptedInvite)
    dim OK1 as boolean = viewController.presentViewController(mmvc)
    if OK1 then
     Log "viewController successfully presented"
     //save object to so can dismiss it later
     Window1.pViewController = viewController
```

```
      //get a match for accepted invite through matchForInviteCompleted event
      MatchMaker.matchForInvite(acceptedInvite)
    else
      Log "Problem presenting viewController"
    end if
  else
    //else respond programmatically
    Log "MatchMaker responding programmatically to invite"
    MatchMaker.matchForInvite(acceptedInvite)
  end if
elseif playersToInvite() <> nil then
  //used when get invite in Game Center not in this app
  //this situation not covered here
  Log "received invitation with " + str(playersToInvite.Ubound+1) + " players invited"
end if
else
  //reject the invite with code to be added
end if
```

After the player has accepted the invite in the modal dialog, a `viewController` and `matchmakerViewController` are created and the Game Center window is presented. The `viewController` object

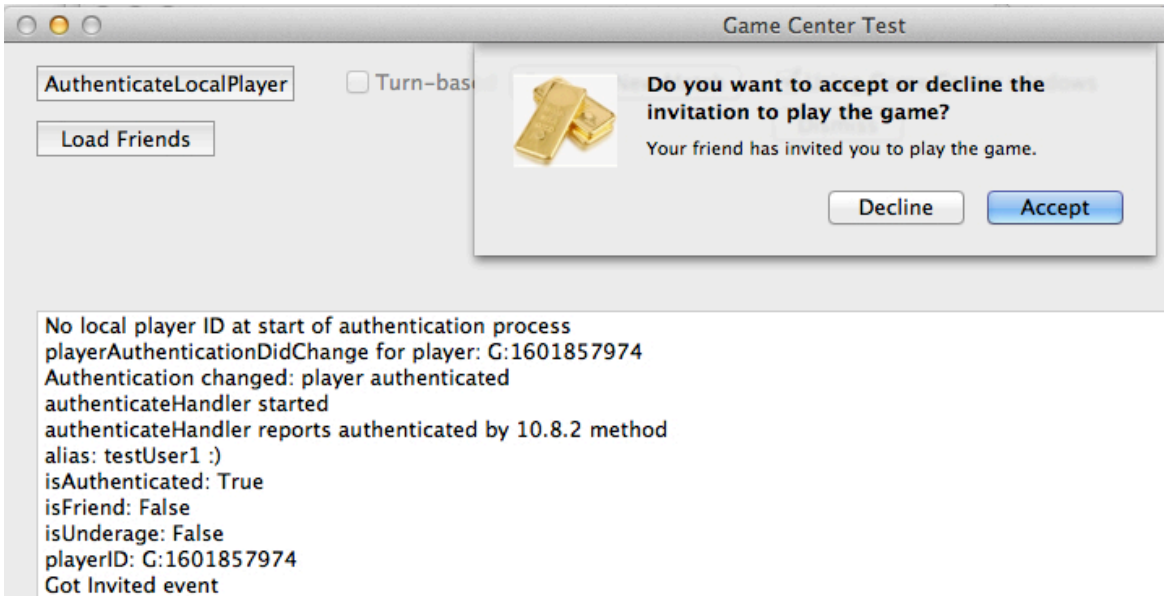


Figure 3: *Accept match invite dialog*

is saved as a property so it can be closed later. Now the code requests the GKMatchMBS object which is returned in the `matchForInviteCompleted` event and saved for future use as is shown in the following code. If the `expectedPlayerCount` is zero, the game can begin. Otherwise, the game needs to wait until all players have joined the match.

```
matchForInviteCompleted(Matchmaker as GKMatchmakerMBS, invite as GKInviteMBS, match as GKMatchMBS, error as
      NSErrorMBS, tag as variant)
 Log "matchForInviteCompleted"
 //save returned GKMatchMBS to use during this match
 pTheMatch = match
 dim count as integer = match.expectedPlayerCount
 if count = 0 then
  Log "Ready to start match!"
 else
  Log "Waiting for " + str(count) + " player(s) to join match"
 end if
 if error <> nil then
  Log "matchForInviteCompleted with error: " + error.LocalizedDescription
 end if
```

Finally, we have to close the Game Center windows that have been opened. On Mountain Lion the ESCape key will close the Game Center windows but on Mavericks it does not. Since we saved

the viewController as pViewController property when we displayed the Game Center window, the "Dismiss" button will close it with this simple code:

```
if pViewController <> nil then
  pViewController.dismiss
end if
```

That is as far as we will go in the match request process using a Game Center window in this article.

Programmatic Match Request and Response

You might want to use a programmatic match request when you don't care with whom the players are matched. Both players make a request and Game Center makes the match.

To request a match without using the Game Center window, you first need to uncheck the "Using Game Center windows" checkbox. After authenticating the local player, click the "Request New Match" button.

Code Listing 1 shows the simple code being executed during a programmatic match request. The Progress Wheel is made visible so the user knows something is happening.

If there is a friend identified in the popup menu, then that player ID is added to the match request and the match request's SetInviteeResponseHandler is called. (Note there is a "Load Friends" button which will allow you to download friends into a popup menu.)

In the Window1.Open event the pMatchMaker property was initialized with the shared instance of the GKMatchmakerMBS class with the following line of code.

```
pMatchMaker = MyMatchmakerMBS.sharedMatchmaker
```

Now this property is used to send the match request as seen at the bottom of Code Listing 1.

When both players have sent a match request, Game Center makes the match and each player is notified causing MyGameKitMBS to fire the findMatchForRequestCompleted event (see the resulting log messages in Figure 4). This event's code is:

```
findMatchForRequestCompleted(MatchMaker as GKMatchmakerMBS, request as GKMatchRequestMBS, match as
      GKMatchMBS, TurnBasedMatch as GKTurnBasedMatchMBS, error as NSErrorMBS, tag as variant)
  if error <> nil then
    Log "findMatchForRequestCompleted with error: " + error.LocalizedDescription
  else
    Log "findMatchForRequestCompleted"
    pTheMatch = match
    dim count as integer = match.expectedPlayerCount
    if count = 0 then
      Log "Ready to start game!"
```

Xojo Losing Build Script Reference

I am now using Xojo 2014 release 2. I spent a few hours looking for a problem which turned out to be Xojo losing track of the App Wrapper Mini script that codesigns the application after it is built. I had to locate the script and save the Xojo project more than once before it finally stuck. I think the reliable way is to quit Xojo after saving and restart the project.

Code Listing 1: RealTimeMatchReq Method

```
Sub RealTimeMatchReq()
  dim OK as Boolean
  dim request as new GKMatchRequestMBS

  //this is a 2 player game
  request.minPlayers = 2
  request.maxPlayers = 2
  request.defaultNumberOfPlayers = 2
  request.SetInviteeResponseHandler

  if cbInterface.Value then
    //use the Game Center provided window
    dim viewController as new GKDialogControllerMBS
    dim mmvc as GKMatchmakerViewControllerMBS
    mmvc = new GKMatchmakerViewControllerMBS(requ
        est)
    mmvc.DefaultInvitationMessage = "Please join my
        match"
    OK = viewController.presentViewController(mmvc)

    if OK then
      Log "Match request window successfully opened"
      //save object to so can dismiss it later
      pViewController = viewController
    else
      Log "Problem opening match request window"
    end if
  else
    //make a programmatic request
    //set spinning wheel while request is in progress
    ProgressWheel1.Visible = true
    //if there is an opponent playerID in the popup
        menu then setPlayersToInvite

    if popOpponentID.Text <> "" then
      dim players() as string
      players.Append(popOpponentID.Text)
      request.setPlayersToInvite(players)
      //setInviteeResponseHandler used when
        programmatically inviting specific players
      request.SetInviteeResponseHandler
    end if

    Log "No Game Center window will be opened as
        match request is sent"
    pMatchMaker.findMatchForRequest(request)
  end if
End Sub
```

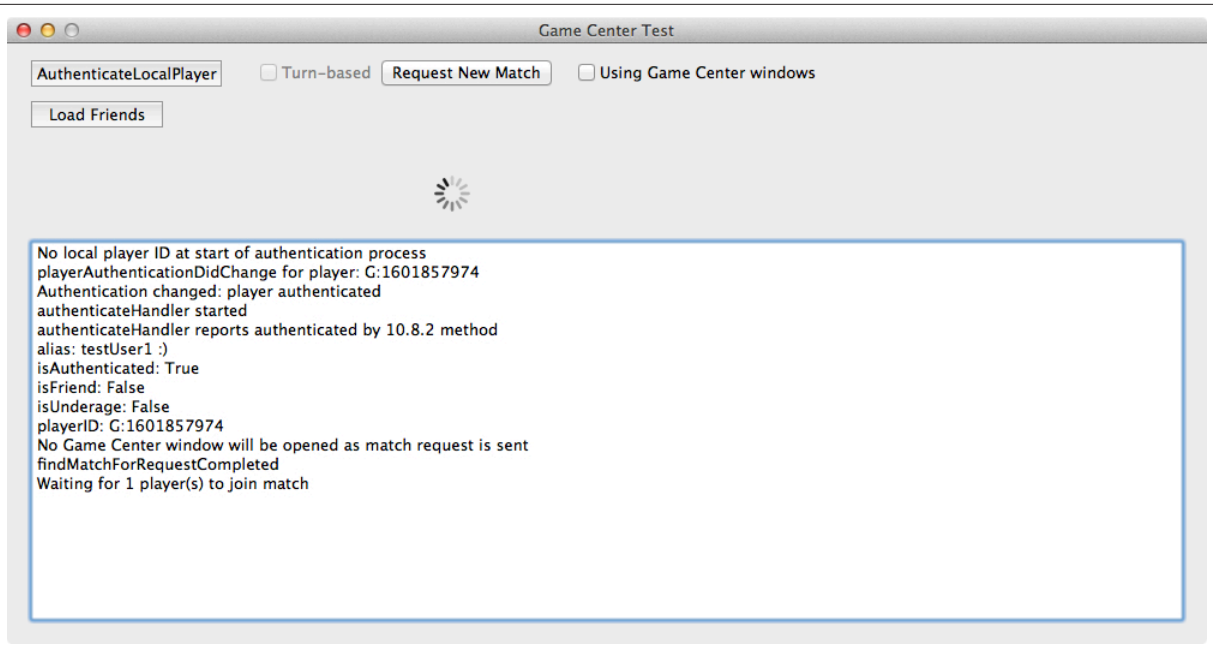


Figure 4: *Reporting a successful programmatic match request*

```
  else
    Log "Waiting for " + str(count) + " player(s) to join match"
  end if
end if
```

After logging the `findMatchForRequestCompleted` event, the GKMatchMBS instance is saved for future use by the game. Again the game can begin when the `expectedPlayerCount` is zero. This is as far as we will go in the match request process in this article.

Unfinished Business

Following is a list of some issues to be covered in future articles:

- After requesting a match with Game Center window it says "Starting Match" and the accepting app requests and receives a GKMatchMBS object actually needed to proceed, but the requesting app hasn't passed one—what is the next step?
- After Game Center passes to each app a GKMatchMBS, it says there is still one player to join match—why?
- When a player declines a match request, how does the app clean up?
- We need to cover turn-based matches

Next article we will solve these mysteries and play a simple real-time game.

by Christian Schmitz

Inside HandleSpecialURL

How to use Web Edition for your own REST API

AT A GLANCE

XD#12506

Target Reader:
Intermediate

Source Code:
Yes

About the Author:
Christian Schmitz is the creator of the Monkeybread Software Xojo/Real Studio Plugins.

With the web edition framework you get a full web server for handling web requests for free. You can use it with the `HandleSpecialURL` event in the app class. In our example, we use this event to handle requests with a REST-like API. We use JSON to package data for the transfer. This is different from a SOAP web service which uses XML for transfer.

Client

First, the client. We take a new project and name it `client`. On the main window we place a `listbox` and a `canvas`. The `listbox` will show the list of image file names and the `canvas` shows the image (see Figure 1).

We add an `HTTPSocket` to the window and name it `sock`. Now in the `open` event of the window you can start a query to the server to get the list of files:

```
sock.Get "http://127.0.0.1:8080/api/images"
```

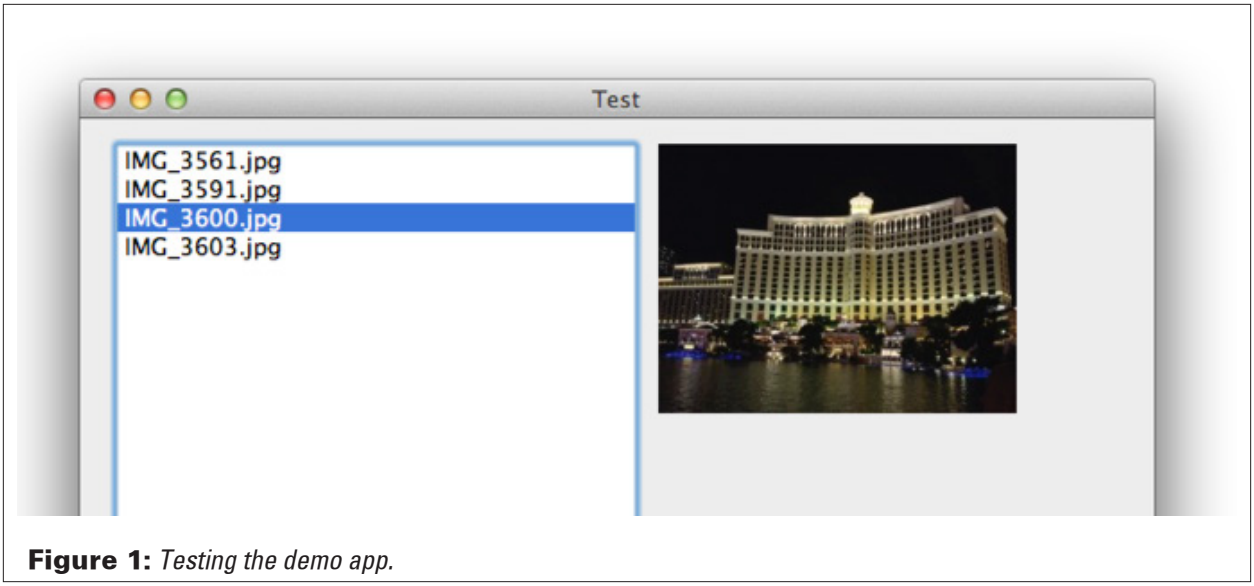


Figure 1: *Testing the demo app.*

As you see, the URL points to localhost as the server, for testing runs locally. Port is 8080 by default for debugging a web app. To trigger the `HandleSpecialURL` event, you can use a URL with `/special` inside or one with `/api`. For most cases, we prefer `/api` as other development environments also use it. The part with "images" is used in the server app to return the list of files.

Loading Image

Next, we add a `change` event to the `listbox`. There we will do a request to the server to query the current image selected. So, first we clear the canvas and check if something is selected. If something is selected, we get the name of the file from the listbox. Then we can send a request which ends with "image/" plus the file name. The server can search for this file and send it back to the client. In the example project it looks like this:

```
Sub Change()
  canvas1.Backdrop = nil
  if me.ListIndex >= 0 then
    dim name as string = me.Cell(me.ListIndex, 0)
    sock.Get "http://127.0.0.1:8080/api/image/" + name
  end if
End Sub
```

Handling responses

When we receive a response, the `PageReceived` event fires. There we can check the result. We can pick the mime type from the headers and check its value. If we got mime type `image/jpeg`, we have

a picture file! In order to show it in a canvas, we use `Picture.FromData(content)` for decompressing the JPEG file.

After that, we want to handle the case of receiving a JSON string. If mime type is `Application/json`, we can pass the content data to the `JSONItem` constructor to parse it. For our server, we will define that any json we return has a `Status` key set with either `OK` or `Error`. This way we can show an error message if an error is reported. If we got `OK`, than it must be our list of file names. So we can take the list returned in the JSON data and loop through the items to get all the file names and display them in the listbox. The event code looks like this:

```
Sub PageReceived(url as string, httpStatus as integer, headers as internetHeaders, content as string)
  dim mimeType as string = headers.Value("Content-Type")
  if mimeType = "image/jpeg" then
    dim p as Picture = Picture.FromData(content)
    canvas1.Backdrop = p
  elseif mimeType = "Application/json" then
    dim j as new JSONItem(content)
    dim status as string = j.Value("status")
    if status = "Error" then
      MsgBox j.Value("message")
    elseif status = "OK" then
      List.DeleteAllRows
      dim jlist as JSONItem = j.Value("list")
      dim u as integer = jlist.Count-1
      for i as integer = 0 to u
        dim name as string = jlist.Value(i)
        list.addrow name
```

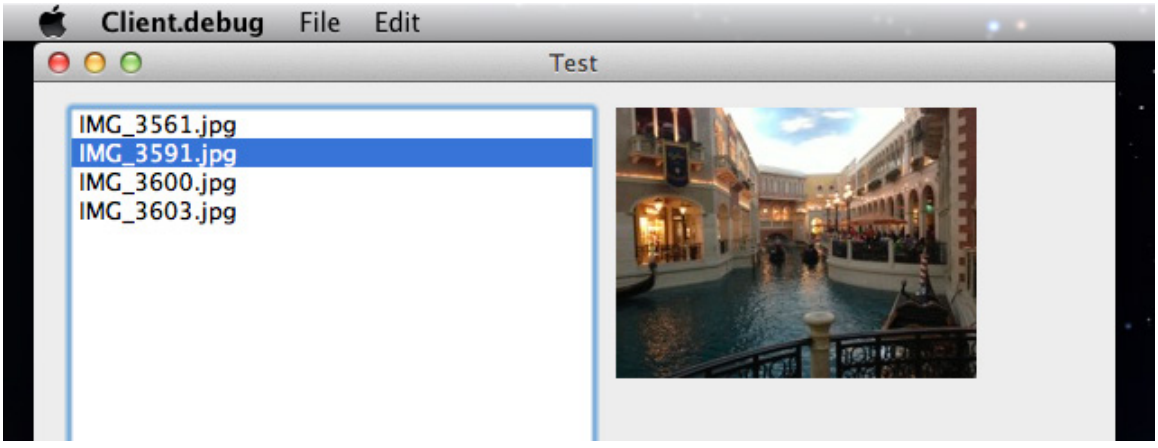


Figure 2: *Testing the client.*

```
    next
  end if
 else
  break // ?
 end if

End Sub
```

Server App

Next we create the server app with the REST web service. For the web project, we add first a property `ImageFolder` in the `app.open` event. Then, in the `app.open` event, we assign the correct `folderitem` to this `imageFolder` property so we can later access the image files.

Here we can add code to the `HandleSpecialURL` event. When we get a WebRequest object there, that's where we'll find properties like the path of the request. Also mime type, method, or query string are there, but we only use the path for this project. So if the path is "images", we know we've got a request for the list of images.

We loop through the image folder and get all the files there. If the file is visible, we add the file name to the json object we use as the list. Then we build a JSONItem for the response. There we put in a status item with the text `OK`. This way the client will know there was no error. So we add the list to this response and we can use the `Request.Print` method to send the string representation of the json response back. We set the mime type to `application/json`, so the client knows it's a json encoded object. Finally we return true here to tell the Xojo framework that we handled the request. The code so far looks like this:

```
if Request.Path = "images" then
  // build list of file names
  dim list as new JSONItem
  dim c as integer = ImagesFolder.Count
  for i as integer = 1 to c
    dim file as FolderItem = ImagesFolder.TrueItem(i)
    if file<>Nil and not file.Directory and file.Visible
       then
      list.Append file.name
    end if
  next
  // build result
  dim j as new JSONItem
  j.Value("status") = "OK"
  j.Value("list") = list
  Request.MIMEType = "application/json"
  Request.Print j.ToString
```

More HandleSpecialURL

Xojo engineer Greg O'Lone recently published a blog post regarding some bugs with HandleSpecialURL. While these aren't related to today's demo project, if you're interested in using HandleSpecialURL, it's a good idea to read Greg's post (http://www.xojo.com/blog/en/2014/08/handlespecialurl-changes-in-2014r21.php) and the follow-up discussion on the Xojo Forum (https://forum.xojo.com/14715-handlespecialurl-in-2014r2-1/0#p119787).

```
  Return true
end if
```

Send Image

Next in the HandleSpecialURL we want to add code to handle image requests. If the left part of the path is "image/", we expect the following path be the file name. We extract this name in a variable and check if an image file exists with this name. If the file exists, we assign this folderitem to the file property in the request. This way we tell the Xojo web framework to send out the file as the result.

Of course, we also set the mime type to image/jpeg, so the client knows an image is coming. In case the image file does not exist, we build a json object with status set to "Error" and add an error message. Then we send the json as string with correct mime type and return true. In our example the code looks like this:

```
if Request.Path.left(6) = "image/" then
  dim name as string = Request.Path.mid(7)
  dim file as FolderItem = ImagesFolder.Child(name)
  if file.Exists then
    // we return the file directly
    Request.file = file
    Request.MIMEType = "image/jpeg"
  else
    // report error
    dim j as new JSONItem
    j.Value("status") = "Error"
    j.Value("message") = "Invalid file name"
    Request.MIMEType = "application/json"
    Request.Print j.ToString
  end if
  Return true
end if
```

If the request was not for image list or a singe image, we return an error as the request was not valid. With that, our HandleSpecialURL event is done. Now you can try it. If things work right, the server app runs and the client can send requests, so it shows the list of images and you can click on an image and see the picture.

This is, of course, a simple implementation which does not do much error checking. Normally, you would query a file list from database and pass back names and IDs, so the request for the image would not pass the file name, but the ID. However, this demo should get you started. xdev

by Sam Rowlands

Yosemite Ready?

Tips for making your Xojo apps ready for Mac OS X Yosemite

MAC OS X 10.10, code name: "Yosemite." You know it's coming. You've probably played with a developer's preview or the public beta. How'd your application fare? "It works," is probably the answer. But is that enough?

For most cases, your Xojo-made application will continue to function just fine on Yosemite, but it won't by default take advantage of the new visual effects that Apple has added. So I'm going to show you how to get some of them at least. We'll cover a few things for this article:

- Checkboxes and Radio Buttons
- Combined Toolbar and Titlebar
- Transparent Titlebar
- Adding controls to the Titlebar

AT A GLANCE

XD#12507

Target Reader:
Intermediate

Source Code:
Yes

About the Author:
Sam is the codemancer behind Ohanaware Software.

- Status Item icons
- Translucent material

Checkboxes and Radio Buttons

If you're thinking they look and work fine, you're missing the tiny little animation that occurs when they're selected and unselected. This is easily fixed. It seems that Yosemite uses a heck of a lot more Core Animation (and why not), so all we need to do is switch on Core Animation for these objects. This is done by sending the object a "setWantsLayer:Yes" message. The code I've written below will cycle through all the controls on the window and set this for all Checkboxes and Radio Buttons.

```
#if TargetCocoa then
  // —- Enable Core Animation on Checkboxes & Radio Buttons, for 10.10 compatibility.
  declare sub setWantsLayer lib "AppKit" selector "setWantsLayer:" ( handle as integer, value as boolean )
  Dim n,l as integer
  n = me.controlCount-1
  for l=0 to n
    if me.control( l ) isa checkBox or me.control( l ) isa radioButton then
      setWantsLayer( me.control( l ).handle, true )
    end if
  next
#endif
```

Xojo has confirmed that this will automatically be fixed in a forthcoming update to Xojo, but until it's released you can use this code.

Combined Toolbar and Titlebar

An important functionality is where the Standard Xojo ToolBar gets embedded into the Titlebar (so the items are in line with window close, minimize, and maximize buttons). Add this code to a window's `Open` event:

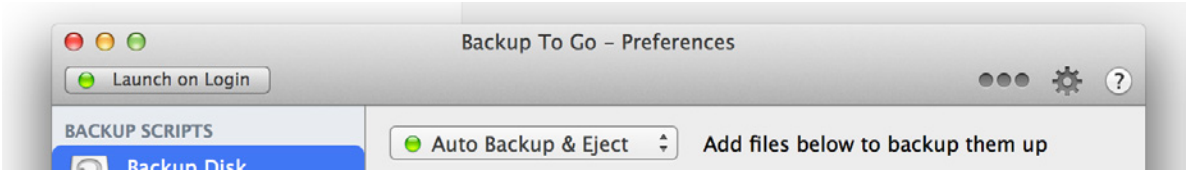


Figure 1: *Backup To Go Toolbar in Mavericks*

```
#if TargetCocoa then
  // —- Displays the toolbar and titlebar as one... 10.10 only.
  declare function NSClassFromString lib "Foundation" (aClassName as CFStringRef) as Ptr
  if NSClassFromString( "NSVisualEffectView" ) <> nil then
    declare sub titleVisibility lib "AppKit" selector "setTitleVisibility:" ( handle as integer, value as integer )
    const NSWindowTitleHidden = 1
    titleVisibility( me.handle, NSWindowTitleHidden )
  end if
#endif
```

Because the property `titleVisibility` is only available on 10.10, we need a way of testing for that OS. One way to do it is to see if the class `NSVisualEffectView` is valid. If it is, we simply set the `titleVisibility` property to `NSWindowTitleHidden`.

If you wish to make the toolbar show only icons, add this code:

```
#if TargetCocoa then
  // —- Set's the toolbar to be 'icon only' and small.
  const NSToolbarSizeModeSmall = 2
  const NSToolbarDisplayModeIconOnly = 2
  Declare Function getMyToolbar lib "AppKit" selector "toolbar" ( windHandle as integer ) as Ptr
  Dim NSToolBarRef as Ptr = getMyToolbar( self.handle )
  if NSToolBarRef <> nil then
    declare sub setDisplayMode lib "AppKit" selector "setDisplayMode:" ( toolbarHandle as Ptr, displayMode as
      integer )
    setDisplayMode( NSToolBarRef, 2 )
    declare sub setSizeMode lib "AppKit" selector "setSizeMode:" ( toolbarHandle as Ptr, sizeMode as integer )
    setSizeMode( NSToolBarRef, 2 )
  end if
#endif
```

Transparent Titlebar

Now before you get too excited, a Transparent Toolbar isn't as awesome as it sounds, but it can still be useful. The declares move the content area of your window up so that the Titlebar overlaps the content area, making the titlebar transparent. If you design it right, you can make a toolbar using the translucent material (described in a minute) so you'll end up with a real translucent titlebar.

> **Note**: Because your interface needs to be re-designed to accommodate this functionality, be warned that it will reduce backwards compatibility.

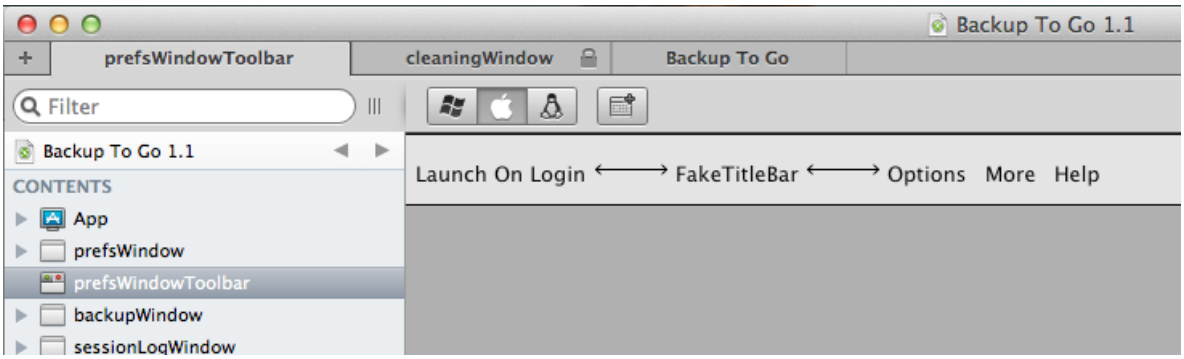


Figure 2: *Xojo Toolbar Designer*

```
#if TargetCocoa then
  const NSFullSizeContentViewWindowMask = 32768
  declare function NSClassFromString lib "Foundation" (aClassName as CFStringRef) as Ptr
  if NSClassFromString( "NSVisualEffectView" ) <> nil then
    declare sub setStyleMask lib "AppKit" selector "setStyleMask:" ( handle as integer, value as integer )
    declare function styleMask lib "AppKit" selector "styleMask" ( handle as integer ) as integer
    setStyleMask( me.handle, styleMask( me.handle ) + NSFullSizeContentViewWindowMask )
    declare sub titlebarAppearsTransparent lib "AppKit" selector "setTitlebarAppearsTransparent:" ( handle as
      integer, value as boolean )
    titleBarAppearsTransparent( me.handle, true )
  end if
#endif
```

Adding controls to the Titlebar

The best way to add controls (such as buttons) to the Titlebar is to actually add them to the toolbar. With the above functions, we can combine the toolbar into the Titlebar. Sounds odd, but this appears to be the correct way do it!

While I can't show you what I mean with Yosemite (screen shots of the beta aren't permitted), if you do this correctly with Mavericks and use the above functions to combine the toolbar with the titlebar and then reduce its size. You'll get the correct effect in Yosemite.

For this, I'll use the forth-coming update to Backup To Go to demonstrate for you. Figure 1 is how the toolbar looks in Mavericks.

Start by adding a toolbar to your project (Insert Toolbar), then populate it with simple buttons and spaces as required (see Figure 2).

Add the toolbar class to the window and arrange the controls off the window. We move them off the window so that we can designed the window without them being in the way (Figure 3).

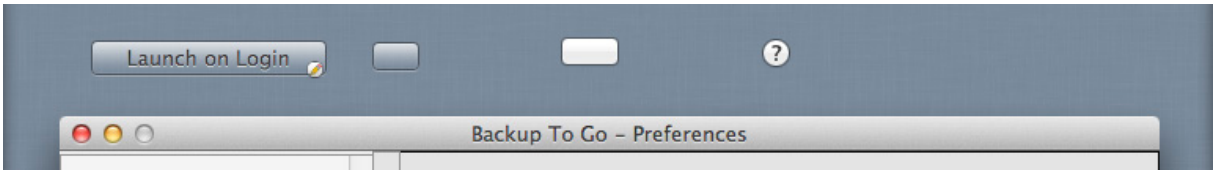


Figure 3: *Window contents layout*

If you have a copy of RetinaKit, there should be a module called `InterfaceTricks`. (If you can't find it, not to worry as I'll include the code needed below.) If you already have it on hand, you can skip the next section.

Add a module to the project and then insert a structure to the module. Name the structure `NSSize` and populate it with two singles as listed:

```
Width as single
Height as single
```

Next add the following method to the module. I'm using it in a module so that it can be easily re-used through my project or in other projects.

```
Sub setItemControl(extends t as toolbar, index as integer, value as rectControl)
  #if TargetCocoa then
    // We need to find the toolbar item
    // First we need to find the window that the toolbar is on.
    Dim w as window = t.Window

    Try
      Declare Function toolbar lib "AppKit" selector "toolbar" ( windHandle as integer ) as Ptr
      // - (NSArray *)items
      declare function items lib "AppKit" selector "items" ( toolbarRef as Ptr ) as Ptr
      declare function objectAtIndex lib "AppKit" selector "objectAtIndex:" ( ref as ptr, index as integer ) as Ptr
      Dim tempToolBarRef as Ptr = toolbar( w.handle )

      if tempToolBarRef <> nil then
        Dim itemArray as Ptr = items( tempToolBarRef )
        Dim itemRef as Ptr = objectAtIndex( itemArray, index )
        declare sub setView lib "AppKit" selector "setView:" ( ref as Ptr, viewHandle as integer )
        declare sub setMinSize lib "AppKit" selector "setMinSize:" ( ref as Ptr, size as NSSize )
        declare sub setMaxSize lib "AppKit" selector "setMaxSize:" ( ref as Ptr, size as NSSize )
        Dim minSize as NSSize
        Dim maxSize as NSSize
```

```
      minSize.width = value.width
      minSize.height = value.height + 2
      maxSize.width = value.width
      maxSize.Height = value.height
      setView( itemRef, value.handle )
      setMinSize( itemRef, minSize )
      setMaxSize( itemRef, maxSize )
      value.lockRight = false
      value.lockBottom = false
    end if
  End Try
 #endif
End Sub
```

If you read through the code, you can the see that the code goes and finds the `NSToolbar` ("toolbar" function) from the window. It then grabs the `items NSArray` from the toolbar and then the `NSToolbarItem` from the array. Once we have the items, we can then use the `setView` message to attach our button or any RectControl to the toolbar item.

We also do some cleaning up, such as altering the size of the toolbar item to match the control, and ensuring that the controls we're using do not auto resize when the window size is changed (otherwise they disappear).

Now; to add our controls to the toolbar, simply call this code in the Toolbar's `open` event:

```
me.setItemControl( 0, launchOnLoginButton )
me.setItemControl( 3, advancedOptionsButton )
me.setItemControl( 2, moreButton )
me.setItemControl( 4, helpButton )
```

Be aware that Toolbar Items start at zero and go left to right. If you compare the above code with the screenshots it should make sense.

Status Item Icons

If you're using a status item in your application, you're almost certainly going to need to change some code for it to display correctly for Dark Mode and Light Mode. Thankfully, it's easy to do. You need to get the `NSImage` and call `setTemplate:Yes`. Basically, once you've set this property, the OS handles the rest for you. Now `setTemplate` only works with grayscale images (non-grayscale images get converted to grayscale).

If you use the MBS plugin, you'll use the following code.

```
myNSImageMBS.isTemplate = true
```

If you use the MacOSLib

```
myNSImage.template = true
```

Translucent Material

> **Warning**: This is an ugly hack and messes with the Xojo hierarchy. Another drawback is that anything placed on top of this control during design time "disappears" when the window is shown.

What we're going to do is to make a canvas subclass called `NSVisualEffectView` (which reflects the name of the Cocoa object). I'm not implementing full functionality—that's for you to do if you want. However, I've found in my limited usage, the best way to implement this is to add a small `NSVisualEffectView` to the window, alter the control order so it's as far back as possible and has zero child controls. Then, in code, if we're running on 10.10, move and lock as you need it.

I'm hoping to obtain a better implementation, but if you really want the Translucent Material now, here you go.

Use the "Insert" menu to add a new class to your project, name this class `NSVisualEffectView` and change its super to `Canvas`. Add the `Open` event, and insert the following code:

```
#if targetCocoa then
// —- First let's check to see NSVisualEffectView exists.
declare function NSClassFromString lib "Foundation" (aClassName as CFStringRef) as Ptr
Dim NSVisualEffectViewClassRef as Ptr = NSClassFromString( "NSVisualEffectView" )

if NSVisualEffectViewClassRef <> nil then
  declare function alloc lib "Foundation" selector "alloc" (classRef as Ptr) as Ptr
  Dim NSVEffectRef as ptr = alloc( NSVisualEffectViewClassRef )
  declare function initWithFrame lib "AppKit" selector "initWithFrame:" (obj_id as Ptr, frameRect as NSRect) as Ptr
  dim frame as NSRect
  frame.X = 0.0
  frame.Y = 0.0
  frame.width = self.Width
  frame.height = self.Height
  Dim mySubClassRef as Ptr = initWithFrame( NSVEffectRef, frame )

  if mySubClassRef <> nil then
    declare sub addSubview lib "AppKit" selector "addSubview:" ( NSViewHandle as integer, subView as Ptr)
```

```
    declare sub setAutoresizingMask lib "AppKit" selector "setAutoresizingMask:" (id as Ptr, mask as Integer)
    declare function contentView lib "AppKit" selector "contentView" ( NSWindowHandle as integer ) as integer
    const NSViewWidthSizable = 2
    const NSViewHeightSizable = 16
    addSubview me.handle, mySubClassRef
    setAutoresizingMask mySubClassRef, NSViewWidthSizable or NSViewHeightSizable
    isUsingVisualEffect = true
  end if
end if
#endif
raiseEvent Open
```

Now add a property of type Boolean called `isUsingVisualEffect` to the class. Add an Event Definition called `Open` to the class also (so the `Open` event gets passed down the stack).

Finally, add a structure called `NSRect` to the class. You need to add the following four properties:

```
X as single
Y as single
Width as single
Height as single
```

To use this, simply drag the `NSVisualEffectView` onto your window. I would recommend following what I mentioned earlier about sending it to the back and only resizing it *if* you're running on 10.10.

Wrap Up

Even if you don't yet have a copy of Yosemite, start preparing your application now. Your app will still look and work great on Mavericks, while taking advantage of cool Yosemite features.

by Marc Zeedar
editor@xdevmag.com

Expanding Objects

OOP makes enhancing software easy

AT A GLANCE

XD#12508

Target Reader:
Beginner

Source Code:
Yes

Supported Platforms:
Mac OS X

About the Author:
Marc taught himself programming in high school when he bought his first computer but had no money for software. He's had fun learning ever since.

AT XDC 2014, I gave a talk on object-oriented programming (OOP). My idea was to encourage people to use more OOP. The analogy I used is that OOP is like getting enough exercise or eating enough healthy foods: we all do *some* of that, but we can all do more. Too much of the time OOP feels like it's too much work to set up, so I showed an example of how OOP can really save you time.

For the presentation, I created a bare bones drawing program called SimpleDraw. I then gave the audience a choice of three tools for me to add during the session: a rounded-rectangle, picture box, or text tool. Because the program was object-oriented, adding a whole new tool only requires a few lines of code: most of the functionality is inherited from the previous tools!

For today's column, I'll walk you through some of my code and show you what I did.

SimpleDraw

Let me say that while "simple" is in the name, SimpleDraw is actually a pretty sophisticated core of a drawing application (see Figure 1). It already supports things like the selection of and movement of objects, zooming in and out, colors, object opacity, and even moving drawn objects forward or backward (layering). Granted, there's tons more needed for a "real" (saleable) drawing application, but the basics are here (see Figure 2).

The cool thing is that with the basic template built, you can now easily expand SimpleDraw. It's really easy to add new types of drawing objects (which I'll show you how to do today), but since

all objects are built on a platform, you can even add new features to *all* the objects.

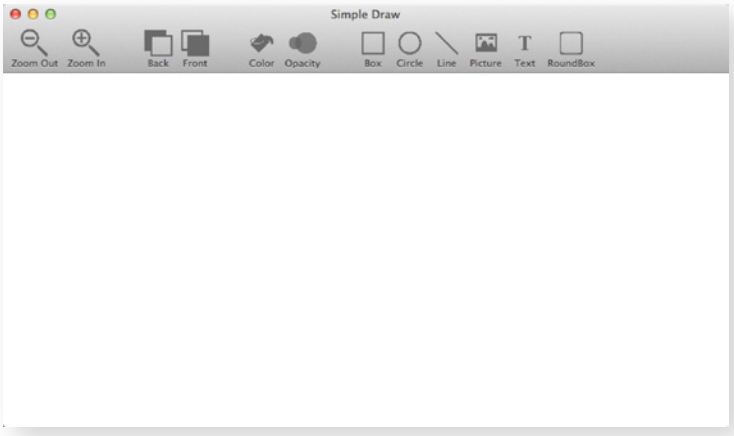


Figure 1: *SimpleDraw, in all its plain glory.*

For example, though the current version does not support named layers, adding such a feature wouldn't be too hard. You'd have to create some sort of data structure for storing the named layers, modify the graphics drawing routine to only draw the visible layers, and you'd need a basic property for all objects that would hold its layer id. Probably the most difficult part of this addition would be the user interface (possibly a floating palette) to allow the user to see and change the names and order of the layers.

The key is that by adding the basic layer property to the drawing object superclass, all objects would automatically support the layer knowledge—including saving that info when the document is saved!

Because the purpose of today's column is to show you how to add some new objects to SimpleDraw, I'm not going to spend a huge amount of time covering how the app shell itself is built. But some basics about it are important, so here's a quick overview of how SimpleDraw is structured.

An Overview

The first thing to note is that I built SimpleDraw as a ContainerControl (called `simpleDrawCC`). This helps keep everything contained, and makes it easy to add a "drawing area" to any window. SimpleDrawCC handles the basic user interface—toolbar, drawing, clicking, moving, and resizing objects, and so on (see Figure 3).

The actual drawing of the objects is a simple routine (called by the `Paint` event). As you can see, it simply loops through all the objects and tells each object to draw itself!

```
Private Sub draw(g As Graphics, printing as boolean = false, alternateX as integer = 0, alternateY as integer = 0)
  // Draw background and border.
  g.transparency = 0
  g.foreColor = &cFFFFFF // White
  g.fillRect(0, 0, g.width, g.height)
  dim x, y as integer
  if alternateX < 0 then x = -alternateX
```

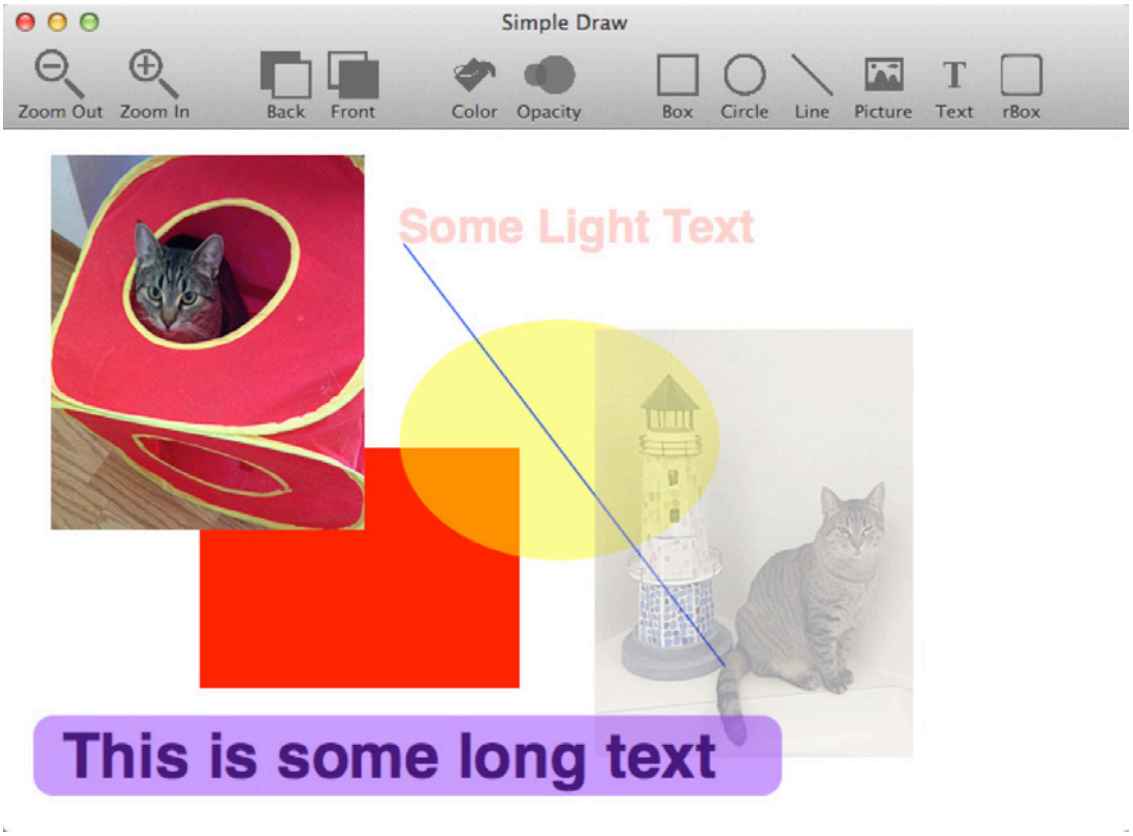


Figure 2: *SimpleDraw, with actual drawing elements. Notice the use of transparency, object coloring, text, basic shapes, and even pictures.*

```
  if alternateY < 0 then y = -alternateY
  // Draw.
  if myDrawing <> nil then
    dim objectNum as integer
    for i as integer = 0 to objectList.ubound
      objectNum = objectList(i)
      dim selected as boolean = (selection = objectNum)
      dim drawob as sdObjectType = myDrawing.lookup(objectNum, nil)
      if drawob <> nil then drawob.draw(g, x, y, myZoom, selected)
    next i
  end if // myDrawing = nil
End Sub
```

The complete drawing is stored in a dictionary object, `myDrawing`, and in an integer array, `objectList`. Every object in the drawing has a unique id which is used to look it up. If an object is selected, the object is drawn with the appropriate selection handles.

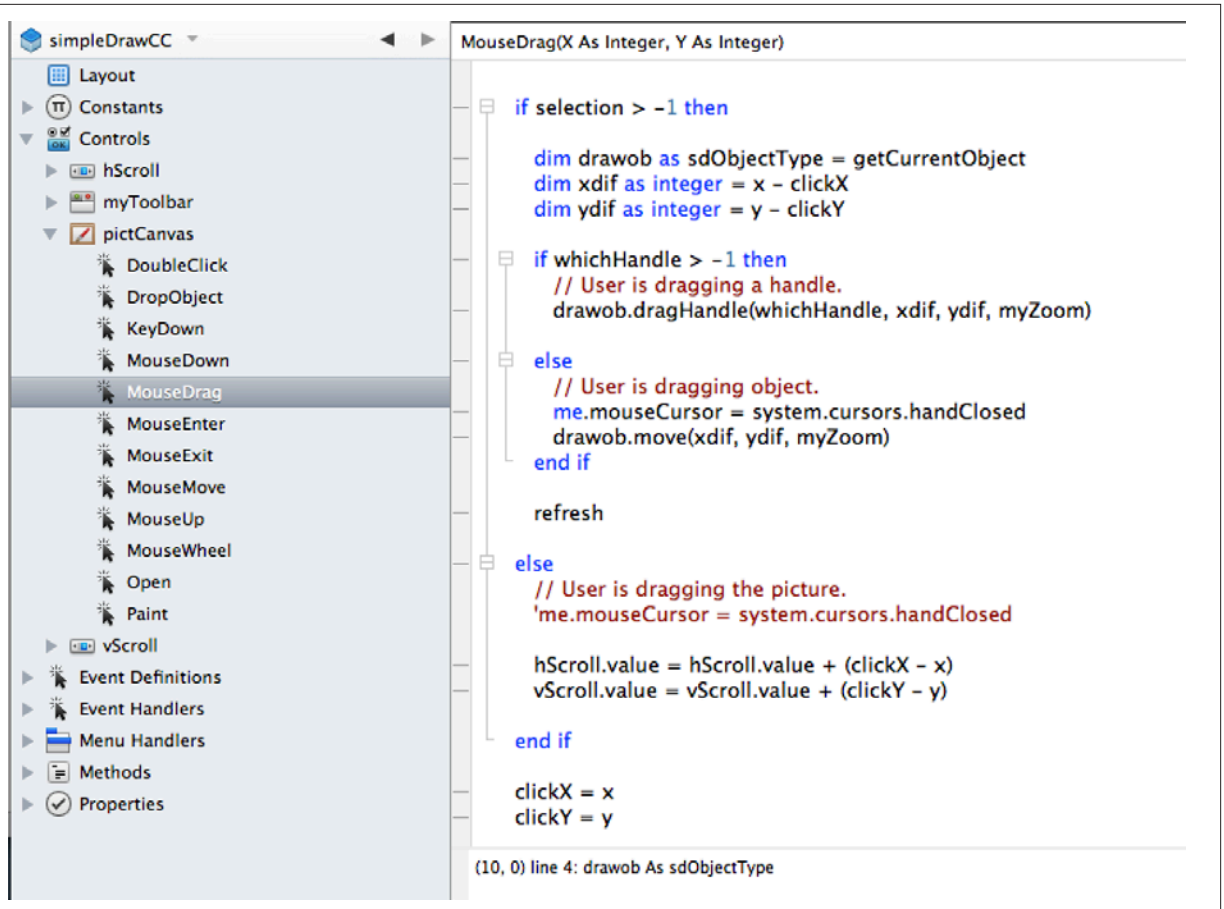


Figure 3: *A ContainerControl holds everything SimpleDraw needs to operate, including the main canvas object which handles all the graphics as well as selecting and moving individual drawing elements.*

Another critical part of SimpleDraw is a module called `sdLib`. It contains some critical functions for reading and writing XML, converting pictures and colors to text and back, and so on.

There's a folder called `simpledraw-graphics` which contains the icons for the toolbar items, and two file type objects (for imported pictures and the SimpleDraw file format).

But the most important folder is `SimpleDrawingObjects`, which contains all our interfaces and classes (see Figure 4). If you remember from previous columns, I use the metaphor of a *disguise* to describe an interface. *An interface lets one type of object pretend to be another*. This is a core part of what makes SimpleDraw work, so you need to understand this.

There are three interfaces used in SimpleDraw as well as several classes (one for each type of drawing object). Remember, I use a "type" suffix in my naming to indicate interfaces, and a "class" suffix for classes:

- sdObjectType
- sdObjectTextType

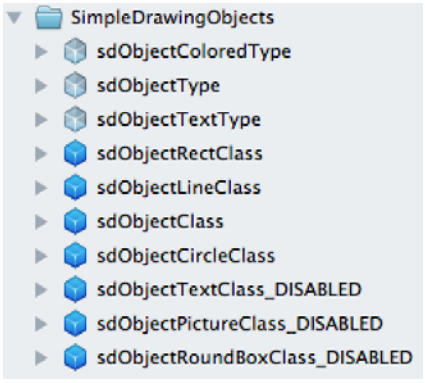


Figure 4: *The SimpleDraw objects, in the "raw" version of the project.*

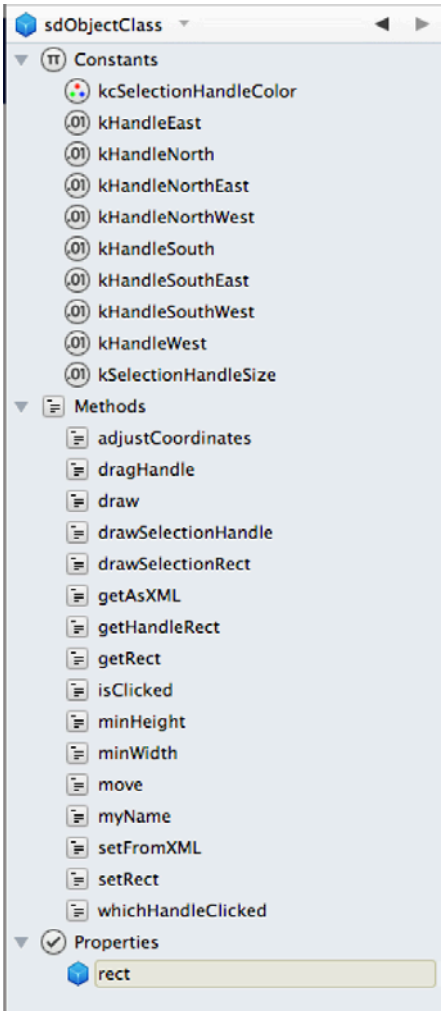


Figure 5: *The superclass of all our drawing objects, sdObjectClass.*

- sdObjectColoredType
- sdObjectClass
- sdObjectRectClass
- sdObjectLineClass
- sdObjectPictureClass
- etc.

One critical feature about interfaces is that a single object can have *multiple* interfaces. This is huge, since an object can only have one type (or have one superclass). This ability of interfaces allows us to create parts of the program that only depend on certain characteristics of drawing objects, making things much more generic.

The main interface, `sdObjectType`, just means *any* drawing objects. All drawing objects should be of this type. The other two interfaces represent different *abilities* of various object types.

For instance, `sdObjectColoredType` means that the object supports colors (fill, transparency, etc.) This means that when I have a routine that applies colors to an object, I only have to do it for objects of `sdObjectColoredType` type. That's *much* simpler than having to do a giant `select-case` for every type of object that supports color, and have to modify that code every time I add a new object. With this system, I can just create a new object and say it uses the `sdObjectColoredType` interface and *bam*, it supports being colored.

Likewise, `sdObjectTextType` means that the object supports text. That's obviously useful in a text drawing object, but by separating the feature as its own interface I can now add text capability to other

objects. For instance, how about a picture object that also happens to contain a built-in caption field? (I don't do that, but it would be easy to add.)

You could use this same system to add additional capabilities. (If it's for *all* objects, you'd add it to the base class, `sdObjectClass`, but if it's just for certain objects, you'd create a new interface to support that ability.)

The Base Class

All sdObjects are derived from `sdObjectClass`, so let's take a quick moment to see how that object is built. Look at Figure 5 and you'll see we have some constants (to represent the various selection handles), a single property (a rect), and some methods.

Most of these methods are things you'll never need to toy with as they perform basic internal functions such as adjusting coordinates to handle a zoomed drawing. Some methods deal with things you *may* want to override: by default the system draws dragable handles in the four corners of the object's rect, but you could create your own routine for doing this within your subclass if you needed a different behavior.

Some of the methods you'd *definitely* need to override, such as the `draw` method, which in the case of this do-nothing base class, does nothing. This is also true of `getAsXML` and `setFromXML`, which respectively return the object in XML format (for saving) and set the object's properties from XML (for loading an object from a saved XML file). More on that when we create our own object.

Note that because `sdObjectClass` is the most basic drawing class, it does very little. For certain types of new objects, you'll base it on this class, but for most I think you'll actually start with `sdObjectRectClass` which does a lot more. Looking at Figure 6 you'll see that `sdObjectRectClass` adds some basic properties (`fillColor` and `fillTransparency`), knows how to load and save those characteristics (via `getAsXML` and `setFromXML`), and can draw itself:

```
Sub draw(g as graphics, xDelta as integer, yDelta as integer, zoomLevel as double, selected as boolean)
  dim adjusted as realbasic.rect = adjustCoordinates(rect, xDelta, yDelta, zoomLevel)
  g.transparency = fillTransparency
  g.foreColor = fillColor
  g.fillRect(adjusted.left, adjusted.top, adjusted.width, adjusted.height)
  if selected then drawSelectionRect(g, adjusted)
End Sub
```

If you build your object upon `sdObjectRectClass`, you'll automatically have a rectangle which can be positioned, resized, colored, and made transparent. Of course, any new abilities your want to add will need to be added, but using `sdObjectRectClass` can give you a head start.

Adding New Drawing Objects

There are two project files in today's demo: `SimpleDraw Raw.xojo_binary_project`, which does *not* have the new drawing objects added, and `SimpleDraw Complete.xojo_binary_project`, which does. If you just want to try out or explore the final program, use the last one, but if you want to follow along in the tutorial, use the first one.

The "raw" one is what I used in my presentation. To make it easier and to save typing time, I actually included "disabled" versions of the classes I planned to add (rounded-rectangle, picture box, and text tool). Adding such tools from scratch isn't difficult, but I didn't want my audience growing bored while I stood there typing, so I used those versions as a scratchpad and copied and pasted their methods and properties into a brand new class.

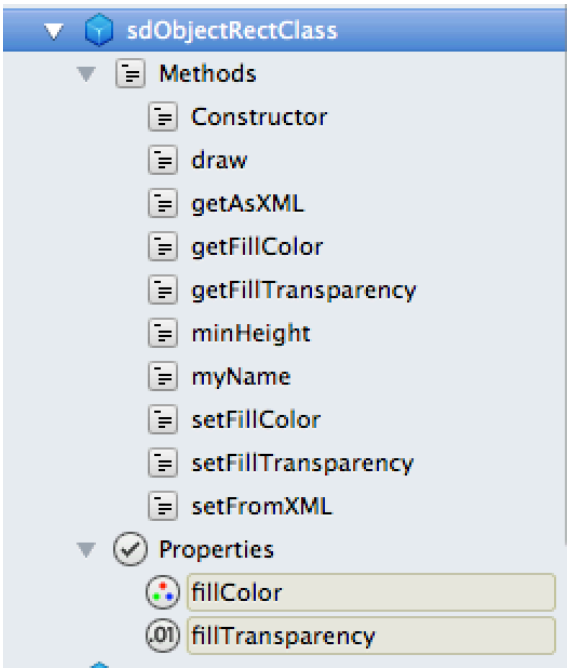


Figure 6: *The sdObjectRectClass class.*

A Rounded Corner Tool

As a simple example, let's add the rounded corner, which is really easy. Start by inserting a new class (`Insert > Class`). Rename it `sdObjectRoundBoxClass`. We then add two methods (you can copy these from `sdObjectRoundBoxClass_DISABLED` if you want):

```
Function myName() As string
  return "roundBoxObject"
End Function
```

```
case "roundBoxObject"
  ' dim r as new sdObjectRoundBoxClass(x, y, w, h)
  ' if node <> nil then r.setFromXML(node)
  ' myDrawing.value(id) = r...
```

Figure 7: *Uncommenting the "roundBoxObject" code with the newObject method.*

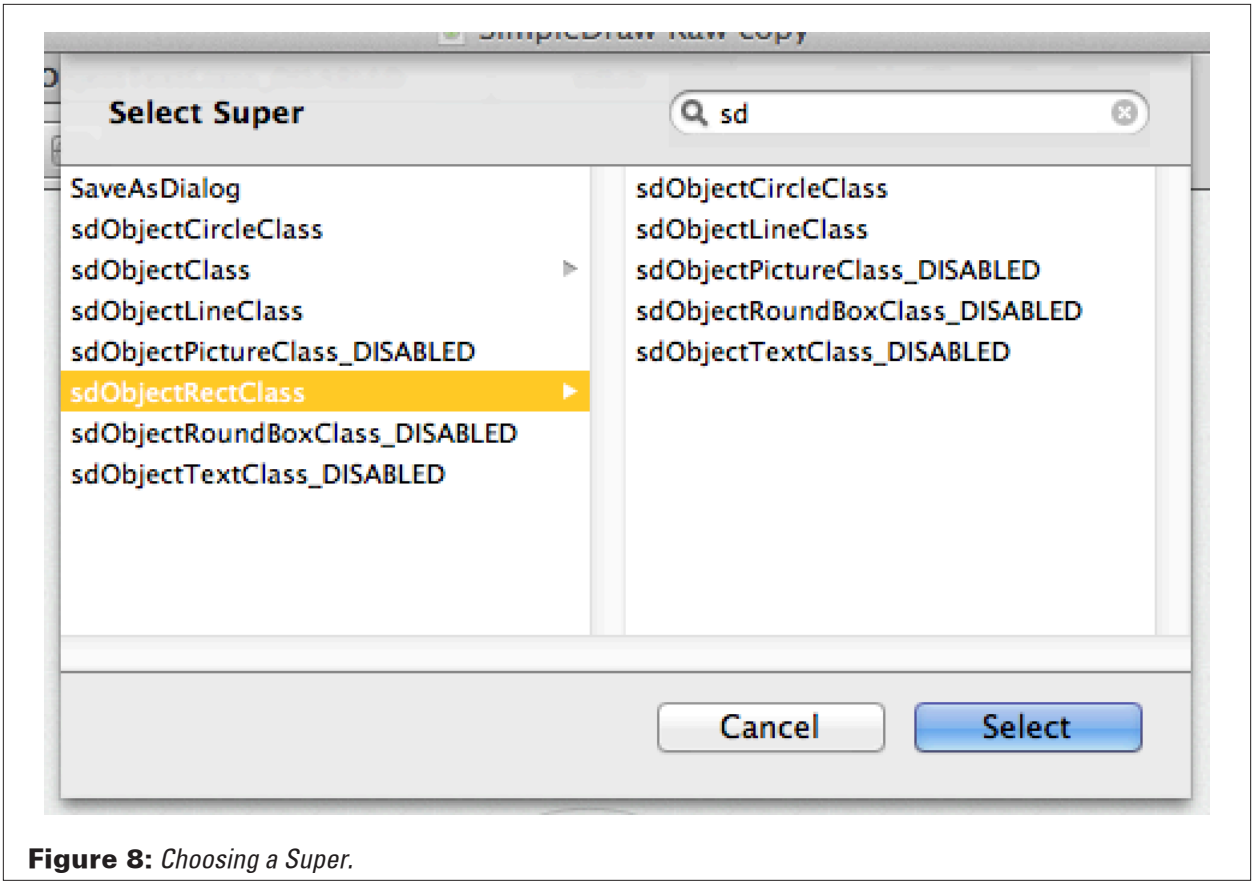


Figure 8: *Choosing a Super.*

```
Sub draw(g as graphics, xDelta as integer, yDelta as integer, zoomLevel as double, selected as boolean)
  dim adjusted as realbasic.rect = adjustCoordinates(rect, xDelta, yDelta, zoomLevel)
  g.transparency = fillTransparency
  g.foreColor = fillColor
  g.fillRoundRect(adjusted.left, adjusted.top, adjusted.width, adjusted.height, 25, 25)
  if selected then drawSelectionRect(g, adjusted)
End Sub
```

The first method just lets the object return a unique name for itself (this is the name for this *type* of object, not any specific object). In this case, we return `roundBoxObject` as the name. What's interesting about that is that in the XML that's generated when the file is saved, it will use "roundBoxObject" as the XML tag for this type of object (e.g. `<roundBoxObject></roundBoxObject>`).

The drawing routine for this graphic is one we could have copied from `sdObjectRectClass` as it only has one change: `g.fillRoundRect` becomes `g.fillRoundRect` and we add the roundness parameters to the call. (For simplicity, I used fixed values for the roundness of the corners, but you could use a property and add a way for the user to set the amount of roundness.)

Believe it or not, we're almost done adding our new object! We just need to make one more change: in `simpleDrawCC`, go to the `newObject` method and uncomment the lines under `case "roundBoxObject"`

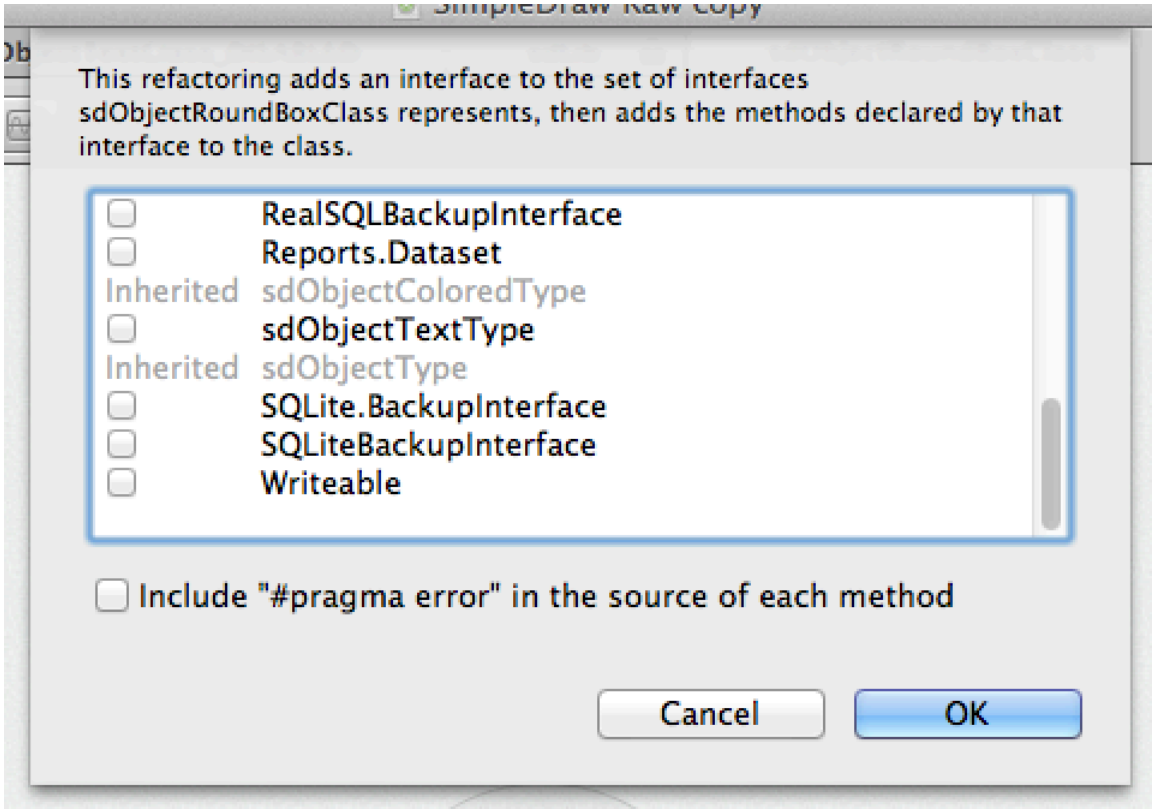


Figure 9: *Choosing an interface for sdObjectRoundBoxClass isn't necessary because it inherits the proper interfaces from sdObjectRectClass.*

(see Figure 7). This just lets the program create this new type of object. Other than user interface items, this is really the only base code that needs to be modified when you add new objects: everything else is code you add into the new objects themselves!

If you try and run SimpleDraw now, you'll get a bunch of errors. That's because we forgot one critical step: we never set the super and interfaces of our new object!

To do that, select `sdObjectRoundBoxClass` and click on the pencil icon next to the `Super` field in the Inspector. In the selection dialog (see Figure 8) you can type "sd" in the search box to restrict it to just "sd" objects, which is helpful. Choose `sdObjectRectClass` as the super.

Then, back on the Inspector panel, click the "Choose" button next to the Interfaces option (see Figure 9). Here you'll want to... do nothing, because our interfaces are already inherited from `sdObjectRectClass`!

Run the program now and you'll see it works: there's now a round box item on the toolbar and it lets us create round-cornered boxes which we can color, resize, and move around (see Figure 10). Wasn't that amazingly easy?

Granted, I did cheat a little, by pre-adding some code to the newObject method, and I'd already added the toolbar items earlier (they just don't work until we add the new objects). But the basic steps aren't too difficult:

- Add new class and set its super to sdObjectClass or sdObjectRectClass
- Add a `myName` method to return the new object type's name
- Add a `draw` method to override the existing one
- Modify the `newObject` method to support the new object type
- In the `myToolbar` control in `SimpleDrawCC`, add code in the `Open` and `Action` events for the new tool

For the simple case of `sdObjectRoundBoxClass`, which doesn't have any settings or properties, we don't have to do anything else—the default save routines work automatically. For most new tools, however, we'd probably need to also do a few other steps:

- Add properties to the object to store any settings
- Add `getAsXML` and `setFromXML` routines which return or set those properties

A good example of the latter is the picture tool: it lets us import a graphic onto our drawing canvas. Lets add it.

A Picture Tool

Insert a new class and name it `sdObjectPictureClass` and set its super to `sdObjectRectClass`. Add a property, `thePicture As picture`, to it. You can copy all the methods from `sdObjectPictureClass_DISABLED` if you want. I won't explain them all, but here's a look at a few of them.

The `draw` method simply draws the picture at the size of the rect of the object. It draws it with transparency. Easy!

```
Sub draw(g as graphics, xDelta as integer, yDelta as integer, zoomLevel as double, selected as boolean)
  dim adjusted as realbasic.rect = adjustCoordinates(rect, xDelta, yDelta, zoomLevel)
  dim g2 as graphics = g.clip(adjusted.left, adjusted.top, adjusted.width, adjusted.height)
  if thePicture <> nil then
    g2.transparency = fillTransparency
    g2.drawPicture(thePicture, 0, 0, g2.width, g2.height, 0, 0, thePicture.width, thePicture.height)
  end if
  if selected then drawSelectionRect(g, adjusted)
End Sub
```

More interesting are the two XML routines:

```
Function getAsXML() As string
  // Part of the sdObjectClass interface.
```

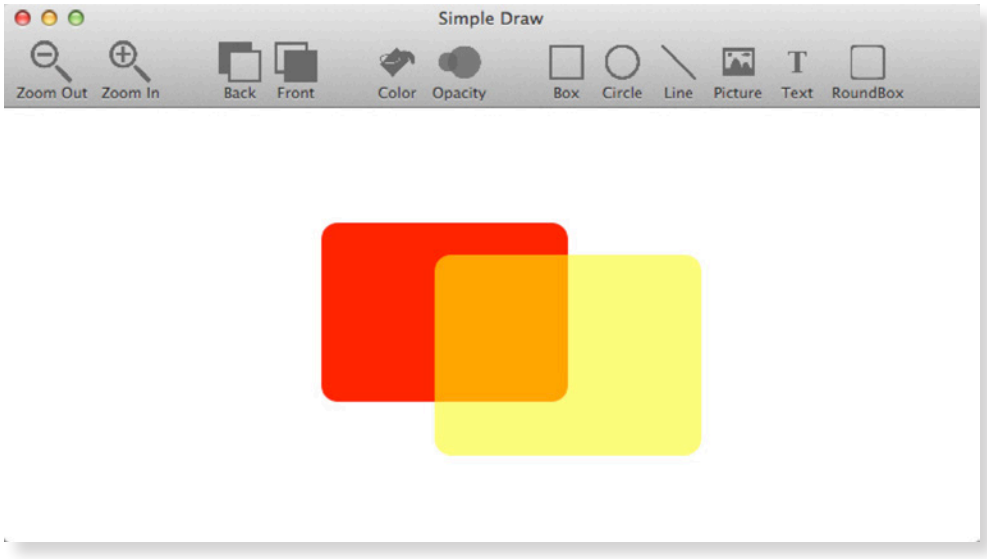


Figure 10: *SimpleDraw, now with a new round box tool!*

```
  dim xml as string
  xml = xml + sdLib.asXML(thePicture) + endOfLine
  return xml + super.getAsXML
End Function

Sub setFromXML(node as xmlNode)
  // Part of the sdObjectType interface.
  dim subNode as xmlNode = findSubNode(node, "picture")
  if subNode <> nil then me.thePicture = fromXML(subNode)
  super.setFromXML(node)
End Sub
```

The first one returns a properly formatted XML sub-node containing the data for this particular object. What's sweet is that since we call `super.getAsXML`, that adds in the XML from `sdObjectRectClass.getAsXML`—meaning the super does all the work for basic details like the rect shape, the fill color, and the transparency setting. For `sdObjectPictureClass`, all we need to support is the new `thePicture` property. To do that, we call our `sdLib` routine which converts a picture to text which can be saved inside the XML file.

The second method *receives* an XML node and extracts the picture from it and stores it inside our object's `thePicture` property. Could it be any easier?

All that's left is to uncomment the `case "pictureObject"` code in the `newObject` method. Bang, just like that, *we're done*! Run it and you'll see you can now place pictures, move them around, resize them, and even make them transparent. Amazing.

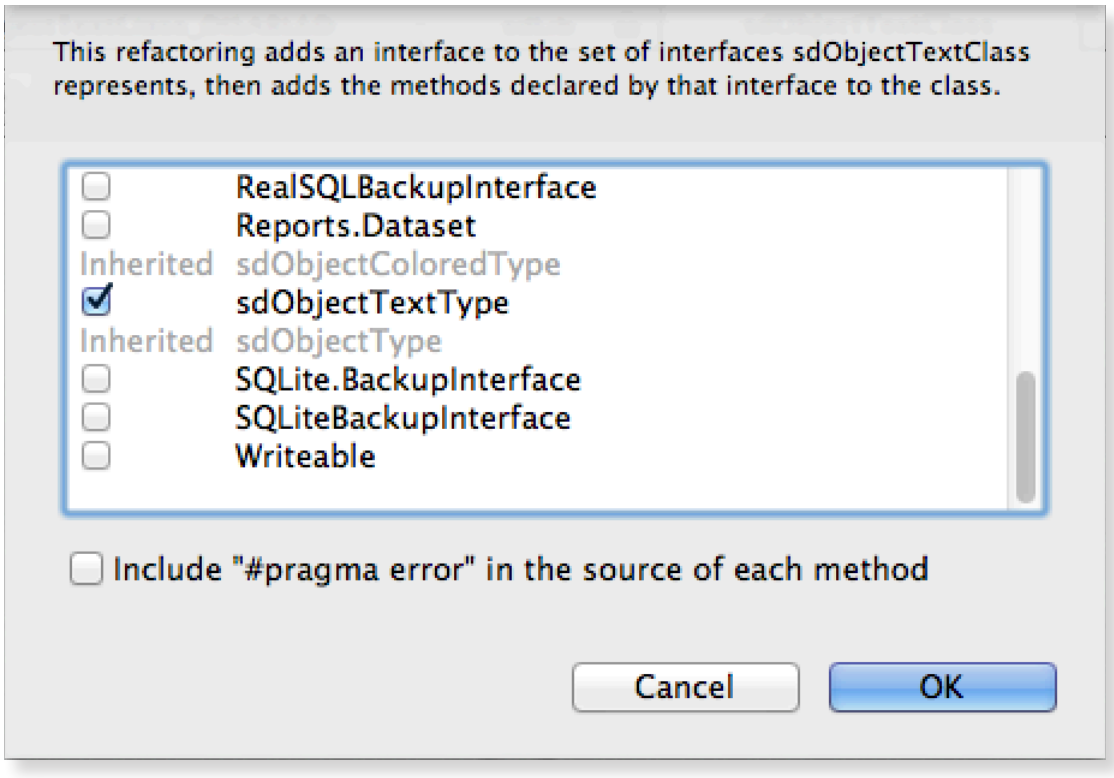


Figure 11: *Setting the sdObjectTextType interface.*

Adding a Text Tool

Adding a tool that lets us type text is definitely more complicated. I did a lot of the heavy lifting already, so I am cheating a little, but it's still pretty easy. As before, insert a new class and name it (`sdObjectTextClass`). Set its super to `sdObjectRectClass`. This time we *do* want to add an interface, so click the "Choose" button and select `sdObjectTextType` (see Figure 11).

Next, add a protected string property (`Protected text As string = "Some Text"`). You'll notice we already have some new methods added automatically when we added the `sdObjectTextType` interface. The text ones are really simple as they just set or get the property we just added:

```
Function getText() As string
  // Part of the sdObjectTextType interface.
  return me.text
End Function

Sub setText(text as string)
  // Part of the sdObjectTextType interface.
  me.text = text
End Sub
```

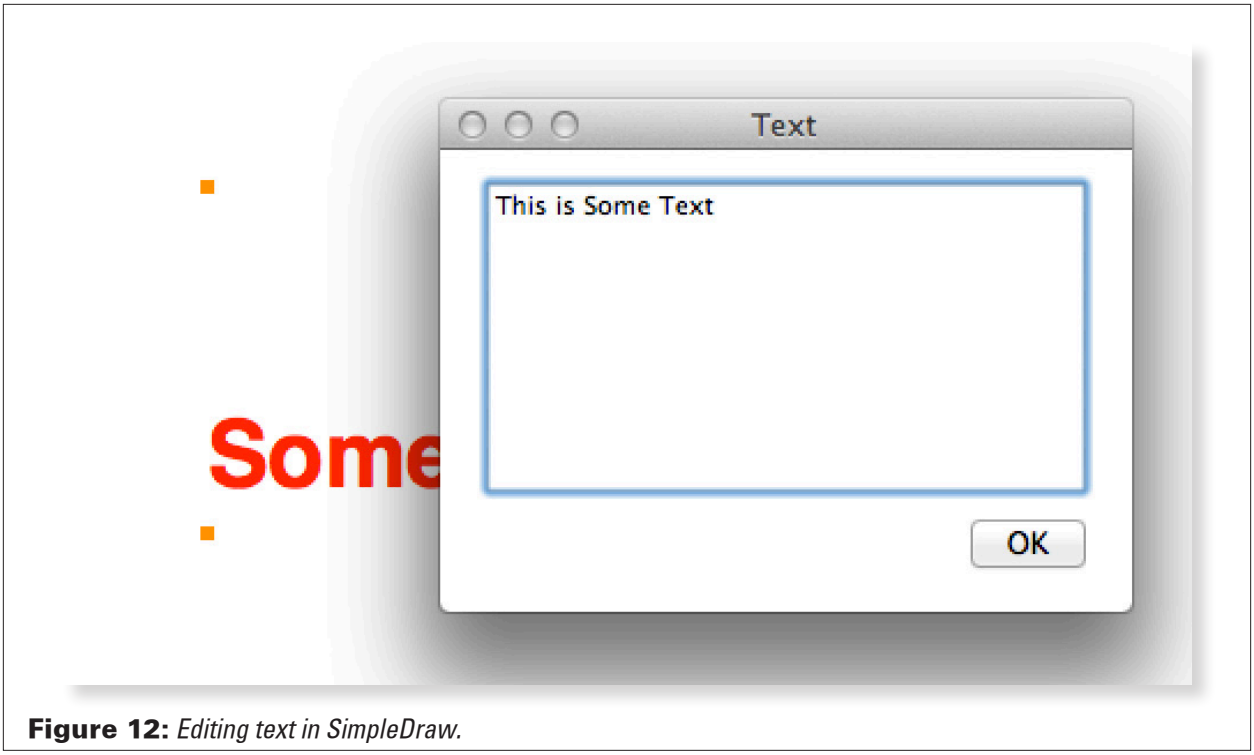


Figure 12: *Editing text in SimpleDraw.*

The doubleClick method is a little more complicated:

```
Sub doubleClick(x as integer, y as integer)
  // Part of the sdObjectTextType interface.
  popupTextWindow.left = x
  popupTextWindow.top = y
  popupTextWindow.taText.text = me.text
  popupTextWindow.showModal
  me.text = app.result
End Sub
```

Here I'm calling up a window I already created which gives the user a place to type their text. The edited text is returned via a property of the application's `app` class (`result`).

The other methods you can copy from `sdObjectTextClass_DISABLED` if you want. Briefly, I've added a custom method called `calcTextSize` that simply figures out the maximum size font for the text within the current size of the object's rectangle. (In other words, the text will always be as big as the box it is in. Make the box smaller and the text shrinks. Make it bigger and the text grows.)

The `calcTextSize` method is called by `draw`, when the text is told to draw itself. The XML routines are almost identical to the ones used in `sdObjectPictureClass`, except these set and get the `text` property of `sdObjectTextClass`.

One Last Fix

If you run the program, you'll notice that *almost* everything works. However, you can't double-click on text or pictures!

That's because we need to enable that. Go into `simpleDrawCC` and inside `pictCanvas`, find the `doubleClick` event. Uncomment the code there so it looks like this (if you've only added one type of object, either picture or text, you'll need to uncomment only the lines of code for that object):

```
Sub DoubleClick(X As Integer, Y As Integer)
  dim drawob as sdObjectType = getCurrentObject
  if drawob <> nil and drawob isa sdObjectTextClass then
    sdObjectTextClass(drawob).doubleClick(x, y)
  elseif drawob <> nil and drawob isa sdObjectPictureClass then
    sdObjectPictureClass(drawob).doubleClick(x, y)
  end if
  refresh
End Sub
```

This is actually a poor design decision on my part. Basically, not all objects support double-clicking, so here we're enabling it only for those types that do. However, I'm hard-coding those types in here, which I shouldn't do. To fix this, I'll create a `doubleClickableType` interface—then I can change this code to look for that type and call the object's `doubleClick` event then. But that's for the future. For now, this is what we have to do. Enable this code and we can edit our text (see Figure 12).

Finally, give saving your drawing a try. It should create an XML file something like this (picture data abbreviated for space):

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<drawing><textObject><rect><left><integer>123</integer></left><top><integer>216</integer></
      top><width><integer>253</integer></width><height><integer>61</integer></height></rect>
<text><string>VGhpcyBpcyBTb21lIFRleHQ=</string></text>
<fillColor><color>255 0 0</color></fillColor>
<fillTransparency><double>0.0</double></fillTransparency>
</textObject>
<circleObject><rect><left><integer>252</integer></left><top><integer>144</integer></top><width><integer>200</
      integer></width><height><integer>150</integer></height></rect>
<fillColor><color>255 255 0</color></fillColor>
<fillTransparency><double>80.0</double></fillTransparency>
</circleObject>
<pictureObject><rect><left><integer>74</integer></left><top><integer>11</integer></top><width><integer>238</
      integer></width><height><integer>204</integer></height></rect>
<picture>iVBORw0KGgoAAAANSUhEUgAAAWMAAAGQCAIAAAD0ppYBAAAEJGlDQ1BJQ0MgUHJvZmlsZQAAOBGF
...
```

```
a0E3IAOgEQYedihebtExHumNWhc0DGbMqZRAEsLtxomtny7/P0A9vVygNhByAAAAAElFTkSuQmCC
</picture>
<fillColor><color>255 0 0</color></fillColor>
<fillTransparency><double>0.0</double></fillTransparency>
</pictureObject>
</drawing>
```

If you reopen the saved file, you should see the previous drawing re-opened exactly as it was before.

Wrap Up

This column should get you started on exploring the benefits of object-oriented programming, particularly in how to build an OOP-based app that allows for flexible additions. Explore the project file and experiment by adding your own drawing tools.

By the way, as a backup for my XDC presentation, I created some little videos of me showing the steps for adding these three objects to SimpleDraw. There's no narration (as I assumed I would be there to narrate), but you may still find them interesting. You can download them from the *xDev* website (http://www.xdevmag.com/downloads/xdc2014movies.zip). Be aware it's 115MB file!

by Markus Winter

A Game Developer

From the capital of California

AT A GLANCE

XD#12509

Target Reader:
Beginner

Source Code:
No

About the Author:
Markus is a Molecular Biologist who taught himself REALbasic programming in 2003 to let the computer deal with some exceedingly tedious lab tasks. Some call it lazy, he thinks it smart. He still thinks of himself as an advanced beginner at best.

THANKS to *BayWatch* and those iconic red bathing suits, California is known the world over as the "Sunshine State" where friendly and easy-going people seem to relax at the beach all day long and life is a breeze. And, thanks to modern technology, people can actually get some work done at the beach, too. But this time we have a contribution, not from the beaches, but from California's Capital city: the "City of Trees," "The Big Tomato," and "The Camella Capital of the World." No, it's not San Francisco as everyone outside and many inside the U.S. seem to assume. It's Sacramento. A place where even university drop-outs like Forest Gump do quite well... and Xojo programmers.

Derek DiBenedetto, Age 43, Sacramento, Calif., USA

Software Developer and indie strategy game designer, simprosestudios.com

The blue glow of the Commodore 64 was my very first exposure to the world of programming. I remember sitting up nights, paging through a wholly inadequate instruction manual, and teaching myself the intricate dance of C64 BASIC. Those 38,911 bytes seemed like a gateway to a whole different world, and I was ready and willing to explore every bit.

Fast forward two years and I was entering a "Best Game" contest at *LoadStar*, a disk magazine devoted to the C64. I entered a dungeon exploration game where all of your adventures and rooms were randomly generated, with seven stats for every guy

Figure 1: *Main screen of Streets of Chaos, developed in Xojo*

tracked throughout the game, and won the contest! The rush was incredible, and my path was set. I was only 13 years old at the time.

Let us move ahead four years, and I was programming forms for my father's work to enable them to fill out accident reports and inventory requests more quickly using QBasic. (One of which they still use today!) Even then, I devoured any and all books about programming and theory, fascinated by the syntax and the various methods of writing better code.

As time went on, I moved on to Visual Basic 4, then VB5 and finally VB6, where I settled into a comfortable and productive relationship for a number of years, and started my own game company to create the strategy masterpieces I always knew I had in my head.

Over time, I began to feel constricted by the age and limited flexibility of VB6, and how apps tended to look "old" in a sense. Concepts like sliders, progress bars, and more modern controls seemed to be missing or required third-party add-ons, and even those were slowly disappearing as VB6 neared the end of its lifespan, at least according to Microsoft. After trying out VB.NET and VB 2013 and finding its syntax to be highly arcane and verbose, I knew I wouldn't be very productive with it, at least not for a long time. So my search continued.

I found Xojo in May 2014 while searching for an alternative to VB, and was immediately intrigued by the multi-platform ability, and the fact it could produce more modern looking and feeling apps.

Continued on page 71

by Seth Verrinder
seth@xdevmag.com

SQLite and the Command Line

Using the SQLite command shell

AT A GLANCE

XD#125010

Target Reader:
Intermediate

Source Code:
No

Xojo Version Required:
2013r1+

Platform(s) Supported:
Mac OS X, Windows, Linux

About the Author:
Seth is a software engineer who works on big data analytics. He worked with Xojo full-time for seven years as a consultant at BKeeney Software, Inc.

TODAY'S column is about the SQLite command line program. I like a good graphical interface but sometimes a command line is the fastest way to get things done and, apart from speed, there's a big advantage if you want to invoke a program from a script. That's why having a command line option for Xojo (for doing builds) has been on my wish list for some time. Build automation helps but it would be nice to be able to just write a script that invoked Xojo to do the build and then signed the resulting executable and built installers. But I digress.

SQLite does come with a command line shell. It comes pre-installed on MacOS so you can just open a terminal and type sqlite3. On Windows, you need to download a copy (http://sqlite.org). There are pre-built binaries on the download page.

Basics

The first thing you need to do is start a terminal. On MacOS, you can use the Terminal application in Applications->Utilities. On Windows, you can use PowerShell (my preference) or the cmd.exe program (go to the start menu and type it in).

Once the window opens, type the following command and press enter:

```
sqlite3
```

You should see something like the following:

```
SQLite version 3.8.5 2014-06-04 14:06:34
```

```
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

There are two types of commands that you can enter now: SQL statements and instructions in the SQLite shell. You can enter SQL statements directly, terminated by a semicolon (;) like so:

```
CREATE TABLE Person(name, age);
```

If you do a `SELECT` then the results of the query are returned right in the shell.

Commands for the shell start with a dot (.). The most useful one to know when you're starting out is the `.help` command which lists all of the available commands. The command to exit the shell is `.quit`.

The shell is a handy place to explore an existing database. Use the `.tables` command to list all of the tables in the database. The `.schema` command will show all of the `CREATE` statements used to define the database's schema. You can also show the schema for a single table using `.schema table-name`.

Note that when you don't specify a database file name on the command line, SQLite defaults to an in-memory database.

Data in, Data out

One of the most useful features of the shell is its support for working with flat files. If you have comma separated data, you can import that into an SQLite data as follows:

```
.mode csv
.headers on
.import file.csv new_table
```

This will create a table called `new_table` if it doesn't already exist. If that table already exists then `.import` will add the data from the file to the existing table.

The `.mode csv` command tells SQLite to expect comma separated values. `.headers on` indicates that the first row of the file contains the column names. If that isn't the case you can use `.headers off`. In that case you'll probably want to create the table ahead of time since SQLite won't know what the column names should be.

Setting the mode also changes how results are displayed in the shell. By default, results are in a human readable form that isn't suitable for parsing. For example:

```
insert into Person values ("Smith, John", 44);
select * from Person;
```

Here's the output with everything set to the default:

```
Smith, John,44
```

This is easy enough for a person to read but it would be hard to use this in a program. Here's the result after setting the mode to CSV and turning headers on:

```
name,age
"Smith, John",44
```

This is ready to import into another program like Excel or Numbers but it's still inside the shell. You could copy and paste the results but that's kind of tedious and only works if there are a handful of records.

Fortunately, there are a couple of ways to redirect the results to a file. The first way is to use the `.output` command. This redirects all output to a file until you change it again using another `.output` command.

For example:

```
.output results.csv
```

Now every time you run a query, the results will be added to results.csv. To switch back to showing results in the shell, use the output command with `stdout` as the file name.

Often you just want to output the results from a single query. There's a special command for that case: `.once results.csv`. This command switches the output for one query and then goes back to outputting results in the shell.

Another trick with csv mode is that, despite it's name, it can work with other separators as well. If you have a pipe separated file that you need to import, you can do that like this:

```
.mode csv
.separator |
```

Use the `.show` command (with no arguments) to see the current settings for all of the options.

Executing SQL Scripts

Another useful command is `.read`. This reads in a file and executes the commands inside it. This is particularly useful if you keep a file with all of the SQL needed to create your database. So to create a new copy of your database:

```
.read create-db.sql
```

Note that the file can also contain any of the dot commands. So you could write a script to create a table and import a CSV file into it, do some query and output the results to another file. Since SQL is such a high level language you can do some pretty sophisticated transformations without a lot of code.

Conclusion

Going back the beginning, I mentioned some of the benefits of a command line interface. Most of the column focused on entering commands into the SQLite shell, which is a command line but it's taking place inside of the sqlite3 executable. It's also possible to specify a command for SQLite to run at the terminal command line.

Let's say you have a script to create an empty database in a file named create-db.sql. You could run that as follows at the terminal:

```
sqlite3 test.db ".read create-db.sql"
```

This would open (or create) a database named test.db and then run the command in quotes (`.read create-db.sql`) and exit. The end result would be a freshly initialized database.

Hopefully that gives you a starting point and an overview for using the SQLite shell effectively. xdev

by Paul Lefebvre
plefebvre@xdevmag.com

Coding Standards

Develop a consistent style

AT A GLANCE

XD#12511

Target Reader:
All

About the Author:
Paul Lefebvre is the Xojo Developer Evangelist. He has developed software professionally for more than 20 years and been using Xojo and prior versions since 2001.

Xojo 2014 Release 2.1

In August, Xojo shipped Xojo 2014 Release 2.1. A follow-up to Release 2 from July, Release 2.1 fixed some lingering issues in the OS X Carbon and web frameworks.

The Carbon fixes are especially important because this is the last release of Xojo that will have OS X Carbon support. When Xojo 2014 Release 3 ships, the *only* options for OS X apps will be Cocoa.

Xojo Cloud

Also in August, we announced that Xojo Cloud has been expanded to a second data center in London. This was a common request and is particularly important for our European users due to hosting regulations there.

Xojo Cloud is the easiest way to deploy your Xojo web apps. With just one click of the Deploy button in the Xojo IDE, your web app is built and uploaded to your Xojo Cloud server for immediate availability!

Coding Standards

This topic has come up recently in the Xojo forum. I've briefly touched on coding standards in the Pragmatic Programmer webinar, but it warrants a longer discussion.

What are coding standards? Simply stated, they are a set of guidelines to make your development process more efficient.

These standards can be anything, but I'd like to focus on two:

- Source Code Management
- Coding Style

Source Code Management

As you are working on your projects, you're going to make lots of changes. Some changes will work well. Some will not work well. And some will be so horribly wrong that you'll want to completely undo what you just did. If you are just saving your project file after every change, how can you easily go back to prior versions?

I often see people come up with convoluted ways to accomplish this, such as repeatedly renaming project files (or folders), copying files to separate drives or folders or relying on DropBox or Time Machine versioning.

Although those processes might work, they are tedious and error-prone. A better idea is to use a solution that is actually designed for this sort of thing: source control (aka version control).

Subversion and Git are the two most commonly-used free source control systems.

Both work a bit differently, but they accomplish the same goal of allowing you to track changes made to all your source code. And don't think that you need to have multiple people working on your projects to benefit from a source control system. The ability to track changes is essential even for a single developer.

For best results you should use the Xojo Project file format, which saves each project item as its own separate file. I don't have space to go into how you would configure or set up Subversion or Git in this column, but luckily there is a lot of material that is already available on how to do just that:

- **Source Control with Xojo** webinar which gives you an overview of both Subversion and Git
- *User Guide Book 4: Development, Chapter 5: Code Management, Section 2: Using Source Control* covers both Subversion and Git
- **Getting to Know Git** webinar
- *Get Real with GIT*, in November/December 2010 of *xDev Magazine* (http://xdevmag.com/browse/9.1/9108/)

You can find the webinars at the webinar page in the Documentation wiki (http://docs.xojo.com/index.php/Videos). And of course the User Guide is easily available from the Help menu in Xojo.

Coding Style

A coding style is how to write your code, including:

- how it is formatted
- general programming techniques

Formatting of code is a problem as old as programming itself. You can see my formatting style when you look at any of the example projects that are included with Xojo (in addition to examples in the Language Reference and User Guides). Some of the formatting guidelines I follow include:

- All keywords are UpperCamelCase
- All local variables are lowerCamelCase
- Method names are UpperCamelCase
- Method parameters are lowerCamelCase
- Properties are UpperCamelCase
- Private properties have an "m" prefix: mPrivateProperty
- Control names are UpperCamelCase with a suffix indicating the type of control

And for programming techniques:

- Prefer `For..Each` loops over `For` with a counter variable
- Prefer `Select..Case` over `If..Then..ElseIf`
- Make methods and properties private by default
- Keep methods relatively short

There is no perfect coding style. The important thing with a coding style is not that we all use the same style or that one style is better than the other. The goal of the style is to make it easier for you to understand code when you look at it later. Of course, if you're working on a team, it is important to have the entire team using the same style so you can more easily understand each other's code.

Having a coding style (and following it) also demonstrates an attention to detail, which is an important programming trait. Don't let your code become a mish-mash of coding styles with all kinds of different formatting and coding techniques. Coding is a craft, so put care into it!

Until next time, keep on coding!

We Are Xojo

Continued from page 64

I knew I had to stretch my wings a bit, and immediately sat down and tried to learn it for my new game project, as a way to extend my reach to new customers and modernize my apps.

I've found the journey rather bumpy at times (the strong typing and required declaration of variables in particular took me a while to grow accustomed to), but I am now quite comfortable and productive as I create my new project for SimProse Studios (simprosestudios.com): "Streets of Chaos", a strategy/board game fully developed in Xojo (see Figure 1).

Xojo has its share of issues like any language, but overall it works (for me anyway) as the "next step" in what VB might have become had Microsoft not followed the .NET route.

Next Time

We would love to publish many more of your stories, so please keep sending them in.

by Craig Boyd
cboyd@xdevmag.com

Database Design Always Matters, Cont.

More time upfront or a lot more later

AT A GLANCE

XD#12512

Target Reader:
All

Source Code:
No

About the Author:
Craig Boyd is currently an Oracle DBA for a well-known national retailer. But in his 18 years of IT experience, he has been everything from a PC Technician, iSeries System Administrator, iSeries Programmer, Sr. Technical Lead, and Data Modeler. He lives in the great state of Texas with his wife and two kids.

IN this month's column we are continuing our discussion of refactoring an existing database design. I would strongly encourage you to review the last couple of columns before reading this one as we will be picking up where the last one left off.

As a quick review there are five steps we are going through:

1. Figure out the current state of the existing database
2. Identify the existing design flaws and determine how best to fix them
3. Gather requirements for the changes
4. Create the final design based on fixing the design flaws and the changes from the new requirements
5. If we have to spread the changes out over several agile iterations how do we do that?

Last month we started with step 2. This month we are going to continue that step. You may recall that there are three issues that we need to address:

1. Data Integrity
2. Inconsistent column definitions
3. Normal Form issues

As I walk you through this, observe how some of the things we do to fix the existing design will sometimes resolve more than one issue.

So, the first thing we are going to do is insulate the application as much as possible from the changes that we are going

to be doing by creating views for each table. This way the only code change that needs to be done in the application is the table name. All the column names will remain the same. There is another alternative approach that is even less invasive. We first create a new schema where the new objects will live.

Next, the new schema needs to be granted access (INSERT, UPDATE, DELETE) to the old schema objects. Now we can create views in the new schema with the same table name. This way the application only has to change the schema that they are pointing to. This approach is much more labor intensive for the DBA as they must make sure and create synonyms for all the objects in the old schema in the new one. A synonym is basically just a pointer in one schema to an object in another. This is handy for all kinds of things, but for our purposes that is the main thing you need to know. Creating a synonym for the tables would not be sufficient since every change to the table would potentially impact the application code whereas a view would insulate the application from a lot of changes since we can control what is exposed to the application. Which approach you take will depend on how your conversation goes with the application team. Also, you will want to very carefully test either approach. For the sake of this article we are going to assume my developers just want to change the table names and leave all the objects in the same schema. Remember, this is not a big application so doing a search and replace for table names and testing is much easier.

After creating all the views, our data model will look like Figure 1. The views are easy to identify as they have the "V_" prefix and each has a dotted outline. Notice that each view has a single line going to it from each table. This just shows which table is participating in the view. It does not have any meaning in a relational sense.

The first thing we are going to fix is the missing primary key on the PROVISIONING table. First, check to make sure that there are no duplicate values in the PROV_ID column by running the following two SQLs:

```
SELECT COUNT(PROV_ID) FROM PROVISIONING;
SELECT DISTINCT COUNT(PROV_ID) FROM PROVISIONING;
```

The first statement just counts the number of rows in the table. We could have just as easily said SELECT COUNT(*)... but I specified the column for clarity.

The second statement counts the distinct or unique values in the column PROV_ID. If the numbers are the same, then we know that the PROV_ID is unique for each row. As it turns out, we are in luck and that is the case this time. The next thing we must do is make sure that the column is NOT NULL. This is one of the conditions of a primary key column, it cannot be NULL. PROV_ID is set to NOT NULL so we move on to actually creating the primary key.

The syntax for Oracle is:

```
ALTER TABLE <SCHEMA.TABLE_NAME> ADD CONSTRAINT <CONSTRAINT_NAME> PRIMARY KEY (<COL1,COL2,...>);
```

```
ALTER TABLE PROVISIONING ADD CONSTRAINT PK_PROVISIONING PRIMARY KEY (PROV_ID);
```

This will do several things:

1. Check that the column is `NOT NULL` and make it `NOT NULL` if it is not. It will show an error if any of the column values are `NULL`.
2. It will create a unique index over the column `PROV_ID`. Remember that unique indexes will allow a single record to have a `NULL` value in the column, assuming we are talking about a single column being the unique index. The index is called `PK_PROVISIONING`.
3. It will create a primary key constraint that will use the unique index that was created and it will prevent `NULL` values from being used in the primary key. The primary key constraint is also called `PK_PROVISIONING`.

With a slight change of syntax, I could have omitted the name for the primary key constraint / unique index and Oracle would have created a system-generated name. I prefer to use explicit names that are formatted a particular way. In this manner, I am easily able to tell which objects belong to which.

Here is how I do my index naming:

Primary key indexes: `PK_<TABLE_NAME>`

foreign key indexes: `FXXX_<CHILD_TABLE_NAME>` where the last two Xs represent a two digit sequence number. For example, since the `APPLICATIONS` table has a foreign key to `PROVISIONING` on the `APPL_ID` column, the foreign key index is called `FX01_PROVISIONING`. So, when I go searching through the object list, all the foreign key indexes for the `PROVISIONING` table are easily grouped / sorted together. You can easily make the case for using the parent table name for figuring out where the foreign key is coming from too. This is just my personal preference and not a technical requirement.

For unique indexes other than the primary key: `AXXX_<CHILD_TABLE_NAME>` "A" is used to show me "Alternate Key" and the last two Xs represent a two digit sequence number.

For non-unique indexes: `IXXX_<CHILD_TABLE_NAME>` "I" is used to show that it is just a plain index and the last two Xs mean the same as in previous examples.

You have heard me and many others say this before: figure out what you like best and then be consistent!

Now that the primary key on `PROVISIONING` has been fixed, we are going to fix the `DATABASE` column. Really it should be the `DB_ID` that is brought over since that is the primary key in the `APP_DATABASES` table. So we add the column `DB_ID` as NULL. If we add the column as `NOT NULL`, then we have to provide a default value. I prefer the `NULL` approach since it is cleaner from a data quality perspective. If you go the default value route, then you have to create a temporary record in the `APP_DATABASES` table that these records can point to until the value can be set correctly. You have to provide that record since we will also be establishing a foreign key constraint after we add this column. With the first approach you are just setting a new

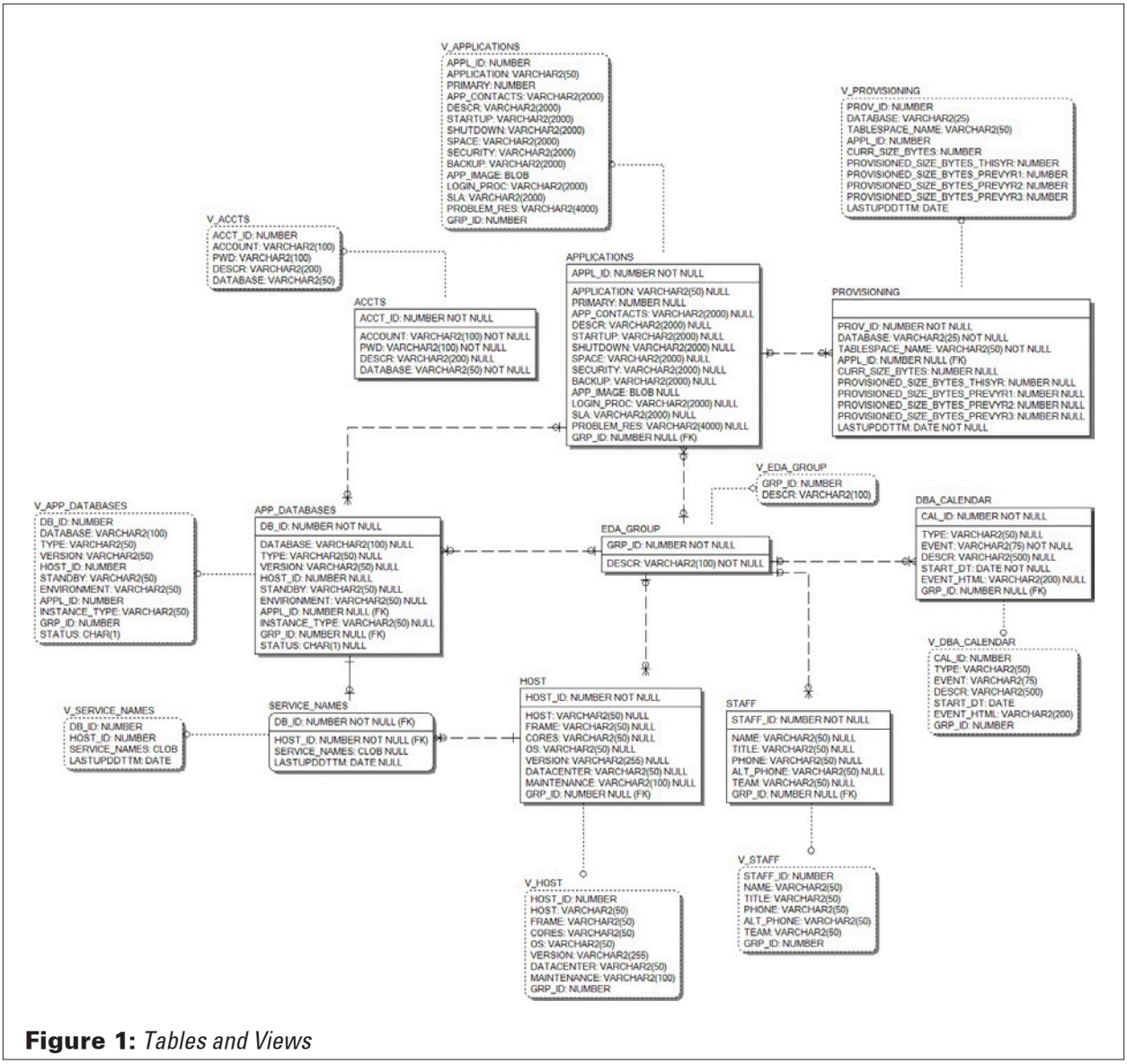


Figure 1: *Tables and Views*

column to NULL until you derive the values. With the second, you still have to derive the values in the child table *and* you have to go back and clean up the parent table.

Here are the steps in DDL:

1. `ALTER TABLE PROVISIONING ADD (DB_ID NUMBER);` (NULL is assumed, you have to explicitly state NOT NULL if that is what you want).
2. `ALTER TABLE PROVISIONING ADD (CONSTRAINT FK02_APP_DATABASES FOREIGN KEY (DB_ID) REFERENCES APP_DATABASES (DB_ID) ON DELETE SET NULL);` This is the foreign key constraint. For right now with the data integrity issues we have, we will leave the constraint for ON DELETE SET NULL. But once things are closer to being fixed, we will

change it to ON DELETE RESTRICT. That will prevent the parent record from being deleted if there are child records. Right now, if the parent is deleted, then the value is set to NULL.

3. CREATE INDEX FX02_PROVISIONING ON DAORADM.PROVISIONING (DB_ID ASC); When you create a foreign key constraint you almost always want to create a matching index to facilitate joins and referential integrity checks.

That sets us up now for actually fixing the data. At this point you either need to write some code (Xojo or SQL) to look in PROVISIONING.DATABASE and match it to APP_DATABASES.DATABASE. If you get a single hit, then update DB_ID. If you get multiple rows, then you are going to have to look at it and see which record it actually should belong to. And, in case you were wondering, none of these changes impact the view! Since we explicitly used each column name adding extra columns, changing one into a primary key has no bearing on the view.

Now that the data is fixed and we have the DB_ID in the PROVISIONING table, we need to do something to make the DATABASE column stay in sync and the underlying column needs to be renamed since the word DATABASE is a SQL keyword. You should never name any database object with a SQL reserve word because that creates all kinds of problems. Most databases will require that you either single or double quote the word to show that it is not actually the keyword you are invoking if it lets you use the word at all. It is just bad practice.

We can solve this in a couple of ways. Option 1 would be to do a join between the two tables in the view V_PROVISIONING and use the DATABASE column in APP_DATABASES which we will rename to DB_NM. Option 2 would require us to alter the size of the DATABASE column in PROVISIONING, rename it and keep the name in the view, and then add a trigger to populate the field for each new record or update it from the APP_DATABASES table to ensure that it is being populated with the correct value. Option 1 is much easier and more intuitive to understand so that is what I am going to do.

First we will rename the column DATABASE in APP_DATABASES to DB_NM. The SQL looks like this:

```
ALTER TABLE <SCHEMA.TABLE_NAME> RENAME COLUMN <CURRENT_COLUMN_NAME> TO <NEW_COLUMN_NAME>;
ALTER TABLE DAORADM.APP_DATABASES RENAME COLUMN DATABASE TO DB_NM;
```

Now we need to update V_APP_DATABASES immediately because this change breaks that view. The DDL for the updated view will look like this:

```
CREATE OR REPLACE VIEW V_APP_DATABASES
(
DB_ID,
DATABASE,
TYPE,
```

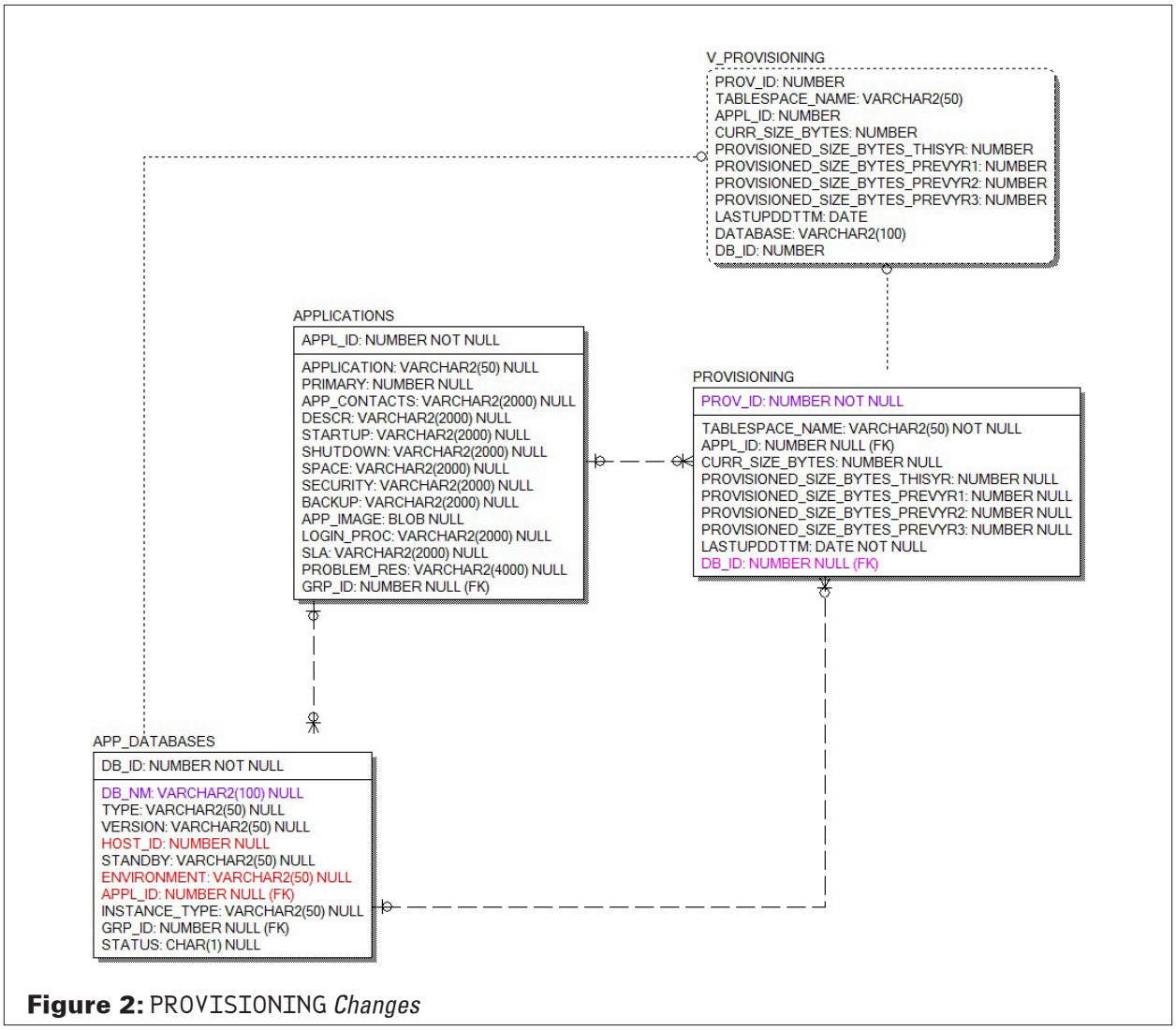


Figure 2: PROVISIONING *Changes*

```
VERSION,
HOST_ID,
STANDBY,
ENVIRONMENT,
APPL_ID,
INSTANCE_TYPE,
GRP_ID,
STATUS
)
AS
SELECT DB_ID,
    DB_NM,
    TYPE,
    VERSION,
    HOST_ID,
    STANDBY,
    ENVIRONMENT,
    APPL_ID,
    INSTANCE_TYPE,
    GRP_ID,
    STATUS
FROM APP_DATABASES;
```

As you can see the column rename is fairly straight forward. Depending on the

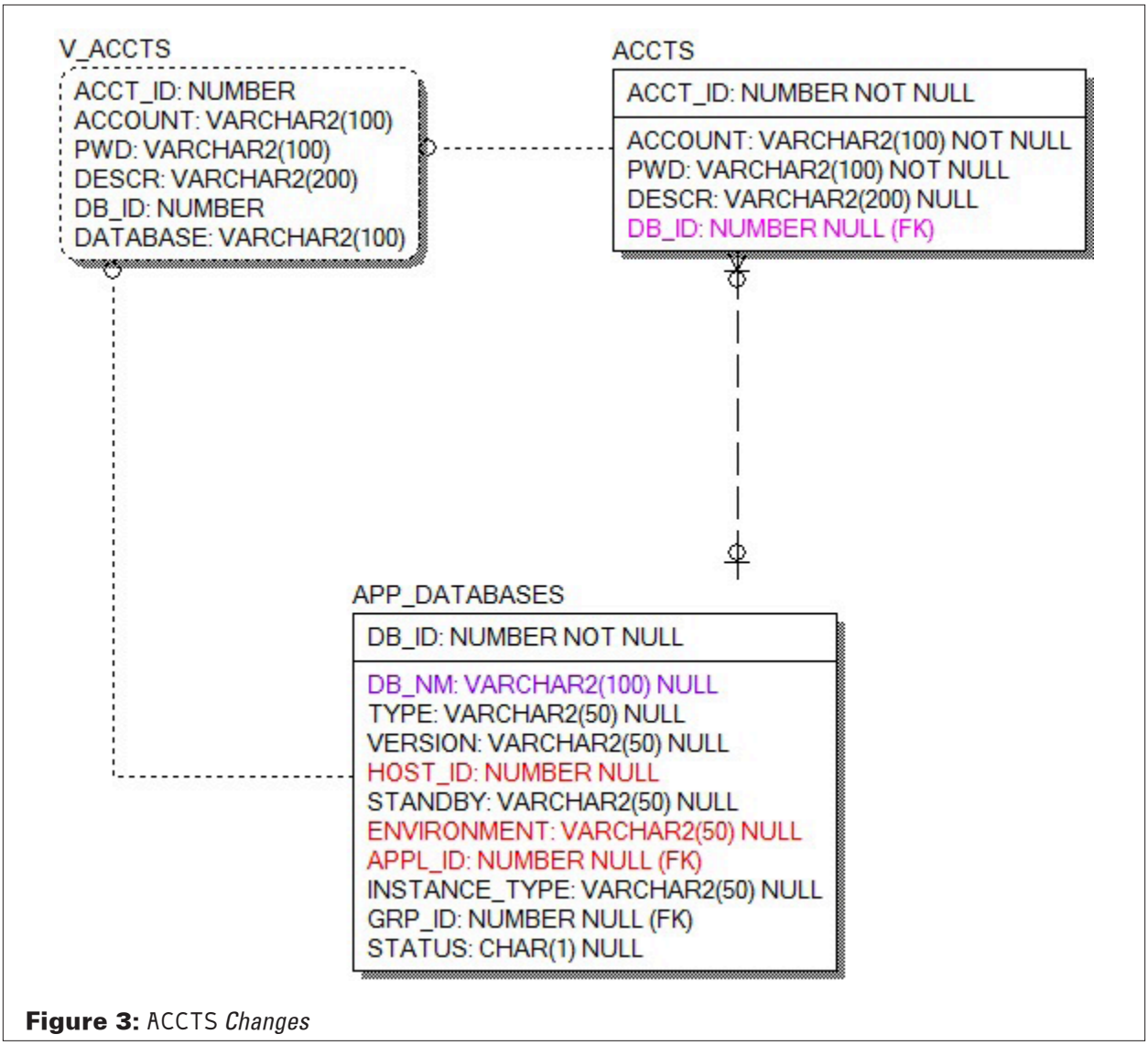


Figure 3: ACCTS *Changes*

DBMS you are using, there will likely be some variations.

Now we need to fix the view V_PROVISIONING by including a join between these two tables and bringing the DB_NM column from APP_DATABASES. The code will look something like this:

```
CREATE OR REPLACE VIEW V_PROVISIONING
(
PROV_ID,
TABLESPACE_NAME,
APPL_ID,
CURR_SIZE_BYTES,
PROVISIONED_SIZE_BYTES_THISYR,
PROVISIONED_SIZE_BYTES_PREVYR1,
PROVISIONED_SIZE_BYTES_PREVYR2,
PROVISIONED_SIZE_BYTES_PREVYR3,
LASTUPDDTTM,
DATABASE,
DB_ID
)
AS
```

```
SELECT P.PROV_ID,
    P.TABLESPACE_NAME,
    P.APPL_ID,
    P.CURR_SIZE_BYTES,
    P.PROVISIONED_SIZE_BYTES_THISYR,
    P.PROVISIONED_SIZE_BYTES_PREVYR1,
    P.PROVISIONED_SIZE_BYTES_PREVYR2,
    P.PROVISIONED_SIZE_BYTES_PREVYR3,
    P.LASTUPDDTTM,
    AD.DB_NM,
    P.DB_ID
FROM PROVISIONING P
INNER JOIN APP_DATABASES AD
ON P.DB_ID = AD.DB_ID;
```

Once you have tested the code and you are confident that it is presenting data that is consistent with the way the developers and end users are expecting it, you can now drop the PROVISIONING.DATABASE column.

The SQL looks like this:

```
ALTER TABLE <SCHEMA.TABLE_NAME> DROP COLUMN <COLUMN_NAME>;
ALTER TABLE DAORADM.PROVISIONING DROP COLUMN DATABASE;
```

For now we are going to leave the first normal form fields there in the table until after we have had a chance to find out what the real requirements are for those. That means that we are done with the PROVISIONING table. Now we are going to go through this same exact exercise with the ACCTS table.

Since it is all the same steps as with the PROVISIONING table, I will simply do it rather than walk you through it step by step to save a little time.

If you look at Figure 2 and Figure 3 you can see the changes up close. Note the color coding. That is one more way I communicate changes to developers. The red columns represent columns that I think have a high chance of changing. A purple column name means that an existing column was in fact changed. In the case of Figure 2 the column DATABASE was renamed to DB_NM and PROV_ID was made a primary key. Bright pink means that it is a new column for that table.

Usually, I provide a data model key or legend that explains the color coding so that they don't have to guess. Here I have simply omitted that for space considerations.

That is about all we have time to do for now. Next time we will continue this exercise with the APP_DATABASES table. I am looking forward to it and I hope you are, too! Until then, happy coding!

Eureka!

Tips and Tricks for the Xojo Developer

by Markus Winter

AT A GLANCE

XD#12513

Target Reader:
Beginner

Source Code:
Yes

About the Author:
Markus is a Molecular Biologist who taught himself REALbasic programming in 2003 to let the computer deal with some exceedingly tedious lab tasks. Some call it lazy, he thinks it smart. He still thinks of himself as an advanced beginner at best.

I HAVE been pretty busy for the last few months on a medium-sized project, and most of the tips this time are based on the problems I encountered and the solutions provided by kind fellow Xojans. Thanks to you all—I couldn't have done it without you!

Tip 1: Getting data out of the ListBox

Sometimes you need to get a whole row, or a column, or even the whole listbox contents out of a ListBox. Beginners usually iterate over every cell in a column like this:

```
Dim Data() as string
For row As Integer = 0 To LB.ListCount-1
 Data.append LB.cell( row, columnnumber ) // get column columnnumber
next
dim result as string = Join( Data, EndOfLine )
```

or every cell in a column like this:

```
Dim Data() as string
For col As Integer = 0 To LB.ColumnCount-1
  Data.append LB.cell( rownumber, col ) // get row rownumber
next
dim result as string = Join( Data, Tab )
// you do remember our Tab as often used constant, I hope
```

However, there is a much simpler way to do this, as has been pointed out by Alex von Siebenthal: simply call one of the following:

```
Dim Data() as string
Data = Listbox.cell( -1, columnnumber ) // gets column
Data = Listbox.cell( rownumber, -1 )  // gets row
Data = Listbox.cell( -1, -1 )  // gets everything
```

Tip 2: Combining a computed property with a weak reference

I love it when something keeps annoying me and I wonder "wouldn't it be possible to..." and then one of the great Xojo minds comes along and writes a tip or blog post about the question I thought about. It means I'm finally starting to ask the right questions!

There was an excellent tip on Christian's MBS Blog (http://www.mbsplugins.de/—if it isn't on your reading list yet, then it should really be as many tips are not specific to his MBS plugins): "Using computed property with weak reference."

Often we need to keep a property somewhere, but it needs to be a weak reference or we run the risk of making circular references and leaking memory in our app. A common example is when we need to keep a reference to a parent window. If the target (e.g. another window) has a reference to the parent, and the parent has a reference to the target, then trying to close the target will not release the reference! But we need the reference to become `Nil` when the target goes out of scope.

But using a WeakRef is also cumbersome and makes the code look ugly as we need to keep the WeakRef in mind when we access the reference: we can't simply say "ParentWindow =" or "= ParentWindow" but we need to say `WeakRef( ParentWindow)` and `ParentWindow.Value`.

It is much nicer to wrap it in a computed property. This way the compiler can check types when using the property. Internally, we map back to the weakRef object like this:

```
ComputedProperty ParentWindow As window1
  Sub Set(value as ParentWindow)
    if value<>Nil then
      mParentWindow = new weakref(value)
    else
      mParentWindow = nil
    end if
  End
  Function Get() as ParentWindow
    if mParentWindow<>nil then
      dim o as variant = mParentWindow.Value
      Return o
    end if
```

```
  End
End ComputedProperty
Property Private mParentWindow As WeakRef
```

As you see we have a private property with the actual weakRef. The computed property-getter checks if we have a WeakRef and provides the value of it. We use a variant to avoid the need of explicit casting. For the setter we either set our weakRef property to `nil` or a new weakRef pointing to the target value.

Please use weak references always for references in data structures where a leaf references back to the tree.

Tip 3: "Break the debugger"

When trying a just finished project with some actual user data, I ran into problems. Everything seemed to work, there were no errors, but the output was nonsense. Hunting down bugs that do *not* cause your code to throw up an exception can be an exercise in frustration. Setting breakpoints revealed that a value in a dictionary that should always be positive was negative—but I had no idea where that came from. And as that directory value was directly and indirectly changed in many different places, stepping through the code was not an appealing prospect. So I sprinkled my code with break statements that looked for "impossible" values like this:

```
if SomeValue < 0 then
  break
end if
```

It is basically a conditional breakpoint, and it quickly pointed me to the offending line. Of course, you can nicely wrap this up in a prettier one-line descriptive method like `BreakOnNegativeValue( ValueToCheck )`.

Which brings us to the next tip:

Tip 4: Be explicit about variants

The offending line was simply calculating the percentage from the amount stored in `DictB` and the sum of all values stored in `DictC`, and storing it in `DictA`:

```
DictA.Value( key ) = 100 * dictB.Value( key ) / DictC.Value( key )
```

On first look there is nothing wrong with this code. However, dictionaries use variants, and by not being explicit on how to convert the variant, the compiler uses the only information it has to determine what kind of math to use: in this case, `100`, which is an integer. So it's going to do the whole thing in integer arithmetic. Which meant that the number retrieved from the dictionary (23164713) was actually too big to fit into an integer after multiplication with 100, so it overflowed to -1978495996, and the result came out as a negative number: -3 instead of 4.7!

The lesson here then is: never ever use a value returned from a dictionary directly without:

- assigning it to another non-Variant variable,
- casting it, or
- using one of the properties of Variant like DoubleValue or IntegerValue.

So the above line should have been

```
DictA.Value( key ).DoubleValue = 100 * dictB.Value( key ).DoubleValue / DictC.Value( key ).DoubleValue
```

I could, of course, also have used 100.0 to force it to use floating point maths, but who remembers that after a few months?!

Thanks to Shao, Eli, Tim, Christian, and Norman for putting me right.

Tip 5: Kem makes the impossible possible...

...when it comes to RegEx, that is. I was looking for a way to get a list of unique entries from a list (as an example, think of a list of unique words in a text, or all the unique IP addresses in your website log). It is easy enough to do: use a RegEx to get the items you want from the text, then use a dictionary to get unique entries, and finally extract the list from the dictionary into an array.

However, I wondered: could RexEx do it in one go? The answer seemed to be "No" until Kem joined the discussion with this little beauty:

```
dim rx as new RegEx
rx.SearchPattern = "(?msi-U)\(([^)]+)\)(?!.*\(\g1\))"
dim  matches() as string
dim match as RegExMatch = rx.Search( sourceText )
while match <> nil
  matches.Append match.SubExpressionString( 1 )
  match = rx.Search()
wend
```

While I do have his excellent RegExRX (and it is a big help with my RegEx), I'm not sure I would have gotten that in my lifetime...

Next Time

That's it for this time. If you want to contribute any tips, then feel free to send me an email (tips@xdevmag.com).

by Kem Tekinay
(ktekinay@xdevmag.com)

Speaking On Condition

Conditionals In Regular Expressions

AT A GLANCE

XD#125014

Target Reader:
Beginner

Source Code:
No

Platform(s) Supported:
OSX, Windows, Linux

About the Author:
Kem Tekinay is a Macintosh consultant and programmer who started with Xojo when it was still REALbasic. He is the author of RegExRX (http://www.mactechnologies.com/index.php:i?page=downloads#regexrx), the popular regular expression editor for Mac and Windows.

COULD you write a method without using an `If` statement? Well, you could, if it had very limited scope, and often you do. But can you imagine writing an entire application like that? Well, you could do that too, if it were a one-trick-pony-type program. The classic "Hello World" needs to make no decisions of any kind, but something more complex must take conditions into account all the time.

And so it is with regular expressions. Usually your needs are pretty straightforward and you can create even complex patterns where the only conditions that need be considered are in the text itself. Regular expressions are one big conditional, after all, since every token will match—or not—depending on the source text.

But sometimes you will need to make more advanced decisions based on whether a section of the pattern matched, or even what was matched. Naturally, there are structures for this purpose.

Alternator

The simplest type of conditional is alternation, covered briefly in the previous column on Subgroups. Strictly speaking, this is not really a conditional, but it does provide a simple way to make a decision and is relevant to the usage of true conditionals.

When any one of multiple choices will fit your pattern, you use an alternator. It's an `Or` statement against your text where the this or that (or the other) will match. For example, suppose you are trying to match entries in a log file, but are only interested in certain months. If the dates are in the form of YYYY-MM-DD, and

you only want February and March entries, the pattern would use the alternation token, the vertical bar, to list the choices: `^\d{4}-(02|03)-\d{2}`

If you had more choices, you could keep listing them. To include November and December, you would use: `^\d{4}-(02|03|11|12)-\d{2}`

The parenthesis are required here because the alternation is embedded within your larger pattern. It applies back to the start of the pattern or the last subgroup, but they are not needed when your entire pattern is an alternation. For example, `cats|daogs are wild` would match the word "cats" or the phrase "dogs are wild", while `(cats|dogs) are wild` would match "cats are wild" or "dogs are wild".

As covered in the earlier column on Subgroups, you can choose to not capture subexpressions by using the `(?:` structure. In our earlier example, if the actual month is not important, we can rewrite the pattern like this: `^\d{4}-(?:02|03|11|12)-\d{2}`

Getting Iffy

A true conditional is when you test for a condition and react accordingly. In Xojo, this would be, among others, an `If condition Then...` or `If condition Then...Else...` statement, and there is an equivalent within regular expressions. Simply, the structure is `(?(condition)then)` or `(?(condition)then|else)` where `condition` is the circumstance for which you're testing, `then` the pattern to attempt if the condition is true, and, optionally, `else` the pattern when the condition is false.

There are two types of conditions to test. The first is a lookaround (covered in detail last issue) where you peek at the upcoming characters or those just passed to determine if they meet a criteria.

The second uses a reference to a past subgroup to determine if it made a match. That reference can be by absolute index (`1`, meaning "subgroup 1"), relative index (`-1`, meaning "the previous subgroup"), or name (`sg`, meaning "subgroup labeled sg"). The only restriction is that the subgroup must exist.

Let's use an example to illustrate each.

Testing With Lookaround

Suppose you want to test for a valid date in the form "m/d". That is, the first digit or digits will indicate the month and the final digit or digits the day. For the purpose of this example, we know that single-digits will never be prefixed with zero. Also, to keep it simple, we will only concern ourselves with the final two months of the year.

The difficulty here is that November can only have 30 days while December has 31, so the validation for each is slightly different. We will use alternation to ensure the digits don't exceed the permitted values. For November, that part of the pattern is: `\b11/(30|[12]\d|[1-9])\b`

The pattern looks for a word break (`\b`) followed by "11". Then, within a subgroup, it looks for "30", or a "1" or "2" followed by any digit, or any digit from "1" through "9".

To validate December, we use something similar but allow either a "0" or "1" after a "3" in the day: `\b12/(3[01]|[12]\d|[1-9])\b`

We could certainly combine these in a single pattern using alternation, but I'll leave that as an exercise for you. We want to use a conditional to test the upcoming month and pick the right pattern accordingly, so we will use a lookahead as the condition:

```
\b(?(?=11)11/(30|[12]\d|[1-9])|12/(3[01]|[12]\d|[1-9]))\b
```

The conditional is started by the `(?(` structure, and the condition determined by the positive lookahead `(?=11)`. If there is a match, it uses the first pattern, otherwise the second pattern. Depending on the month, the days are validated properly.

(Astute readers will note the assumption that the first number, if not "11", will be "12", but those are the terms stated at the outset. In the real world, this pattern would be insufficient.)

Notice all the parenthesis in that pattern. Parenthesis are normally used to designate subgroups, but they have a special purpose here of delineating the conditional. As such, the only subgroups that will be captured will be in `SubexpressionString(1)` if November, or in `SubexpressionString(2)` if December. Since we don't really need those groups captured here, we can use non-capturing groups for those sub-patterns.

```
\b(?(?=11)11/(?:30|[12]\d|[1-9])|12/(?:3[01]|[12]\d|[1-9]))\b
```

This pattern will not yield any subgroups.

Testing With Lookbehind

Since we know that we are looking for a number to lead off our string, we can easily rewrite this pattern by testing a lookbehind rather than a lookahead. In this case, we will match the first digits, then test their value to choose the appropriate pattern:

```
\b\d{2}/(?(?<=11/)(?:30|[12]\d|[1-9])|(?:3[01]|[12]\d|[1-9]))\b
```

This matches the first two digits after a word break followed by a slash, then uses a lookbehind to test that value. If it's "11/", it uses the first sub-pattern, otherwise the second.

The "then" and "else" patterns have to be enclosed in parenthesis or the pattern will not work. Remember, the `|` token serves double-duty. Normally it's an alternation, meaning "this pattern or that", but within the "if" construct, it means "else". Without the parenthesis, the regular expression engine would not be able to tell the difference and, in Xojo, would raise a RegExSearchPatternException.

Testing a Subgroup

Instead of a lookaround, you can test whether a subgroup was matched. The general form of this construct is `(subgroup)...(?(index)then|else)`. Rewriting our example, we would encapsulate the first digits in subgroups, then test to see which subgroup actually had a value.

We start off by looking for the two values of concern: `\b(?:(11)|(12))/`

The pattern is logically enclosed in a non-capturing subgroup so we can match the word break before and the slash after, but what's important are the subgroups within. If the date is in November, subgroup 1 will have a value, but with December, that will register "no match" and subgroup 2 will contain the month. We use this knowledge within the larger pattern:

```
\b(?:(11)|(12))/(?(1)(?:30|[12]\d|[1-9])|(?:3[01]|[12]\d|[1-9]))\b
```

In short, this looks for a word break followed by either "11" (subgroup 1) or "12" (subgroup 2), followed by a slash. The conditional checks to see if subgroup 1 has a value and uses either the first or second sub-pattern accordingly.

This approach has an additional benefit. While our original assumptions about limited choices of month are still in place, they are not applicable here. This pattern will look for either "11" or "12", and nothing else will match.

A Practical Example

Let's look at another usage that has more practical value like spotting a malformed URL.

A URL can use either double-quotes like `<a href="link">` or single-quotes like `<a href='link'>`, but it would be an error to mix them like `<a href="link'>`. To quickly scan for these, an *If* construct is invaluable. Basically, you'd want to match those cases where the first quotation mark was different from the second without knowing which comes first.

We start with the overall, and very simple, patten to match a URL: `<[^<>]+>`

This looks for the opening "<", then matches everything up to the matching ">". Since both "<" and ">" are excluded from the allowable matches, there is no chance that this pattern will start at the beginning of one URL and stop at the end of the next if the closing ">" is missing.

We now want to match the opening quotation, but don't know which it will be, so we enclose the first in its own subgroup. The start of the pattern is now: `<[^<>]+(?:("")|')`

Notice how only the double-quotes are enclosed in parenthesis since we only need it for testing purposes to know whether it matched or not.

Later in the pattern, we will test that subgroup and attempt to match the opposite. That part of the pattern looks like this: `(?(1)'|")`

This means, if subgroup 1 contained a match, we know it was a double-quote, so try to match a single-quote. Otherwise, it must have been a single-quote so look for a double-quote.

The complete pattern is: `<[^<>]+(?:("")|')[^<>]*(?(1)'|")[^<>]*>`

Using the "If" construct lets us match the opposite quotation mark towards the end of the string (instead of the one that was matched first). This will match `<a href="this'>` and `<a href='that">` (errors), but not `<a href="the other">` (valid HTML syntax).

Until Next Time

Conditionals give you a flexible, powerful way to craft patterns that take into account various conditions within the source text. Just as in a programming language like Xojo, you can use different subpatterns at various points within your larger pattern.

Next time I'll go over some miscellaneous features that make regular expressions more powerful and even easier to read.

New to Xojo?

Check out *xDev Magazine's* "Welcome to Xojo" Bundle!

Bundle includes:

- Year 11 of the magazine in printed book format (500 pages thick)
- Free shipping (up to a $25 value if you're not in the U.S.)
- Year 11 of the magazine in digital format (PDF)
- A one-year digital (PDF) subscription to xDev (beginning with issue 12.1)

That's a total of 12 issues of the magazine—two years worth of great learning!

Bought individually these items could cost as much as $130, but with this special bundle you can save up to 24%: the *Welcome to Xojo* bundle is **just $99** with *free shipping* of the book!

http://www.xdevmag.com/welcome2xojobundle.shtml