iOS - Lessons Learned

David LeBer
Align Software Inc.

A Technical Travelogue

Disclaimer

Oh, wait...

Itinerary

- Project Overview
- Requirements
- Our Choices
- Details
- Lessons Learned

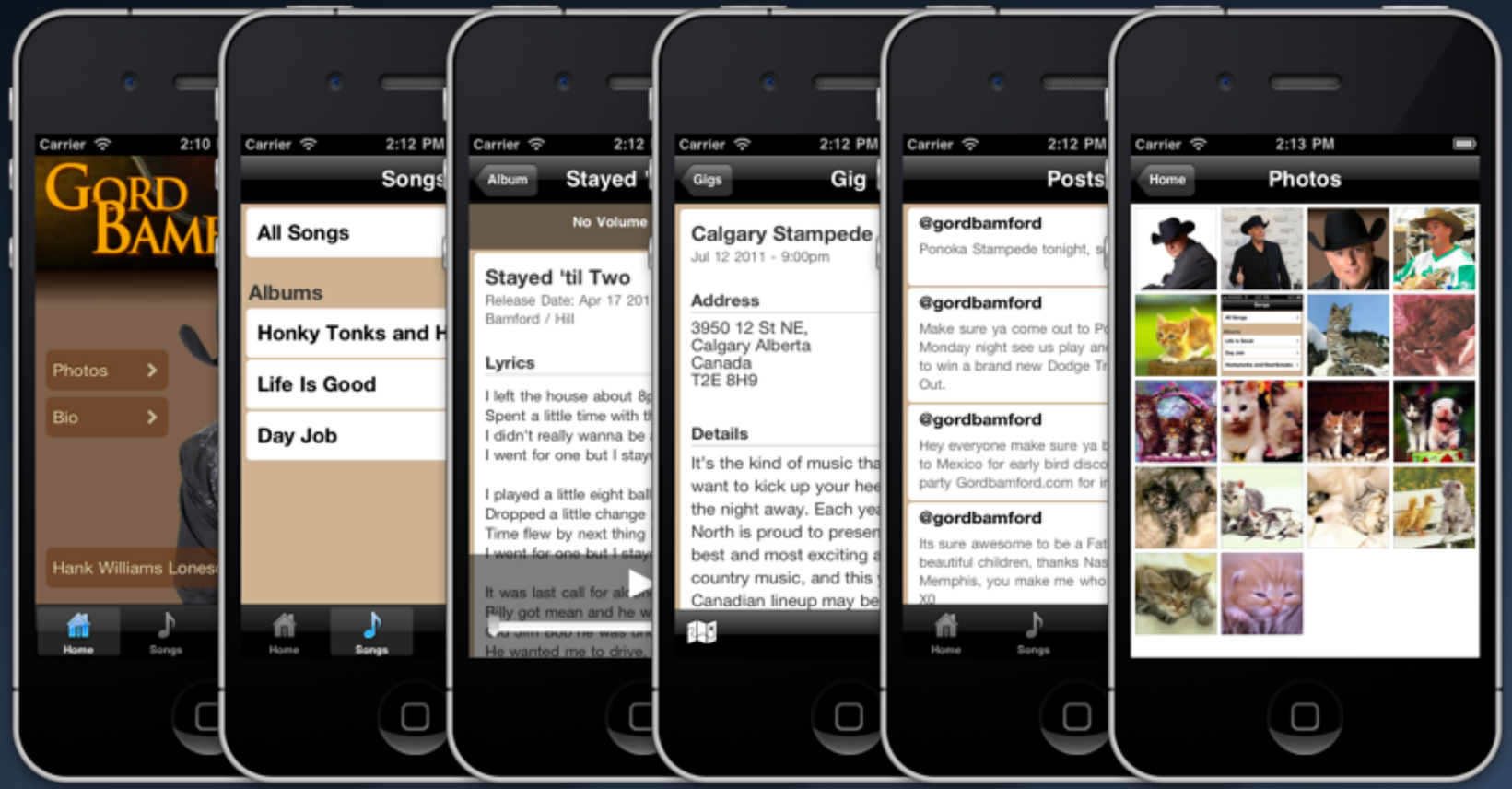
Landmarks

- iOS
- WebObjects
- Deployment

Project Overview

- “A vanity publishing platform for musicians”
- Native iOS based client
- WebObjects based RESTful backend

iOS Client Overview



Requirements

iOS Client Requirements

- Work offline
- Dynamic content / Skinnable / Customizable
- Streaming audio
- “Social”
- Responsive
- High functionality / Low cost

WO Backend Requirements

- User roles
- Agile
- “Cheap”

Deployment Requirements

- Cost effective
- Scalable
- Modular
- Flexible

Choices

iOS Client Choices

- Use CoreData
- Hybrid UI (Native / UIWebView)
- Client side templating
- Use as much 3rd party or open-source code as possible
- Do not re-invent the wheel

WO Choices

- D2W - ERModernLook
- ERRest
- Vend binary PLIST

Deployment Choices

- Collocation / Self hosting
- EC2/EBS/RDS
- Slicehost
- Linode

Details - iOS

Skinnable UI

- Using of UIWebview for UI
- Templated content
- On demand loading of resources

UIWebview Issues

- Slow
- Resource heavy

UIWebview Solution

- Preload UIWebView with empty page content.
- Load content using JavaScript
- Create a pool of “primed” UIWebViews

RDRWebViewPool

- A simple pool of ‘primed’ UIWebViews
- aquireWebview
- returnWebview
- preloadNextAvailableWebview

Mimicking iOS behaviour

- Bound ‘touchend’ events to JavaScript function with JQuery
- Event handler function performs highlighting and navigation
- Custom url scheme caught by UIWebview delegate
- Object identity extracted from DOM element ID
- Highlights generated using CSS gradient
- Chevrons drawn with CSSCanvasContext

Why not...

Loading Content

- UIWebView's stringByEvaluatingJavaScriptFromString
- JQuery: replaceWith(), html()

Templated Content

- Data is templated into HTML client-side
- HTML Cached locally
- MGTemplateEngine
  - Matt Legend Gemmell
  - http://mattgemmell.com/

Templated Content

```
<div class="wrapper items_wrapper" id="list_items_wrapper">
{% if itemsString.length != 0 %}
    {{ itemsString }}
{% else %}
    <div class="no_entries">
        <p>No {{ entityName | capitalized }}s available</p>
    </div>
{% /if %}
</div>
```

Templated Content

```
<div id="rss_{{ item.postIdentifier }}" class="clickable rss_clickable">
    <div class="item post_item rss_post_item">
        <div class="heading">{{ item.title }}</div>
        <div class="date">{{ item.relativeDate }} ago</div>
        <div class="details">{{ item.body | strip_and_crop: 140, ... }}</div>
        <div class="author">{{ item.authorName }}</div>
        <div class="badge rss_badge"></div>
    </div>
</div>
```

On Demand Resources

- Lazily load web resources (images, etc)
- ResourceHandler
  - override valueForKeyPath (“resource”, “label”)
  - return local path for resource
  - fetch missing resources

On Demand Resources

```
<div class="avatar-wrapper">
    <img class="avatar" src="{{ resource.author.avatar }}" />
</div>
```

On Demand Resources

```
<div id="listen_{{ song.primaryKey }}" class="clickable">
    <div class="listen-button">{{ label.listen }}</div>
</div>
```

Streaming Audio

- Apple's AudioFileStreamExample
  - Example only
  - Incomplete/buggy
- StreamingAudioPlayer
  - Matt Gallagher
  - http://cocoawithlove.com

Remote data sources

- NSURLRequest
- CFNetwork
- ASIHTTPRequest
  - http://allseeing-i.com/ASIHTTPRequest/

ASIHTTPRequest

- ASIHTTPRequest
- ASINetworkQueue
- ASIFormDataRequest
- ASIDownloadCache

Responsiveness

- Loading data, templating, fetching resources, etc
- Asynchronous behaviour is a must
- GCD / libdispatch
- NSOperation/NSOperationQueue

NSOperation

- Boilerplate code
  - Delegate
  - Success and failure callbacks
- NSOperation Xcode template
  - Jeff Lamarche
  - http://iphonedevelopment.blogspot.com

Coredata

- NSOperation/background threads
- Consume binary plist
- Simplifying fetching
- Keeping model/classes in sync

Coredata - Background

- Use separate NSManagedObjectContext
- Pass NSManagedObjectID between threads
- Merge changes when done

New MOC

```
NSPersistentStoreCoordinator *coordinator = //EXISTS
if ( coordinator != nil ) {
    moc = [[NSManagedObjectContext alloc] init];
    [moc setPersistentStoreCoordinator: coordinator];
}
```

NSManagedObjectID

```
NSManagedObjectID *objid = //EXISTS
NSManagedObject *obj = [moc objectWithID:objid];
```

Merge Changes

```
NSNotificationCenter *dnc = [NSNotificationCenter defaultCenter];
[dnc addObserver:self
        selector:@selector(handleManagedObjectContextChanges:)
            name:NSManagedObjectContextDidSaveNotification
          object:nil];

//
//
```

Merge Changes

```
- (void) handleManagedObjectContextChanges:(NSNotification *)note;
{
    if ( [[note object] isEqual:self.managedObjectContext] )
    {
        return;
    }
    // Merge changes into the main context on the main thread
    [self performSelectorOnMainThread:@selector(mergeChanges:)
                           withObject:note
                        waitUntilDone:YES];
}

- (void) mergeChanges:(NSNotification *)note;
{
    [self.managedObjectContext
                mergeChangesFromContextDidSaveNotification:note];
}
```

Consuming PLIST

- Created a category on NSManagedObject
- Attributes are easy
- Relationships not so much
- Use delete rules to dictate ownership
- Use userInfo for additional hints
  - illegalKeys, parent, etc.

Simplifying Fetching

- Fetching in Coredata is verbose
- Easy-Fetch is a category on NSManagedObjectContext
  - [moc fetchObjectsForEntityName:[Foo entityName]]
- Created by Austin Ziegler (a tacow member)
- https://github.com/halostatue/coredata-easyfetch

Keeping Classes in Sync

- mogenerator
- mogenerator is inspired by eogenerator
- Created by Jonathan “Wolf” Rentzsch
- http://rentzsch.github.com/mogenerator/

That “Social” Thing

- Sending content to social networks
- Dealing with multiple external APIs is a pain

ShareKit

- Send content to:
  - Twitter
  - Facebook
  - Email
  - Instapaper
- http://www.getsharekit.com/

ShareKit

- Add files
- Link to required frameworks
- Configure keys
- Write a button handler

ShareKit

```
- (void)myButtonHandlerAction
{
    // Create the item to share (in this example, a url)
    NSURL *url = [NSURL URLWithString:@"http://getsharekit.com"];
    SHKItem *item = [SHKItem URL:url title:@"ShareKit is Awesome!"];

    // Get the ShareKit action sheet
    SHKActionSheet *actionSheet = [SHKActionSheet actionSheetForItem:item];

    // Display the action sheet
    [actionSheet showFromToolbar:navigationController.toolbar];
}
```

ShareKit

Details - Model

WO Model vs CD Model

- WO model close to CD model - except:
  - Flattened attachment paths into parent
  - Content is dirty flags
  - Explicit primaryKey attribute
- WO model has Delete EO for tracking deletes
- All entities have a modification timestamp

Details - WO

WO - ERRest Details

- Request Validation
  - Embedded band specific token in request header
  - Controllers extend a subclass of ERXDefaultRouteController
  - See Pascal's ERRest talk for more
- Fetch qualifier
  - Every request included the Band id in a qualifier
  - Rest routes are generic, specifics fed as query parameters

WO - UI

- Needed Band site and Admin site
- Use ‘childrenChoices’ in ERXNavigationMenu

WO - UI

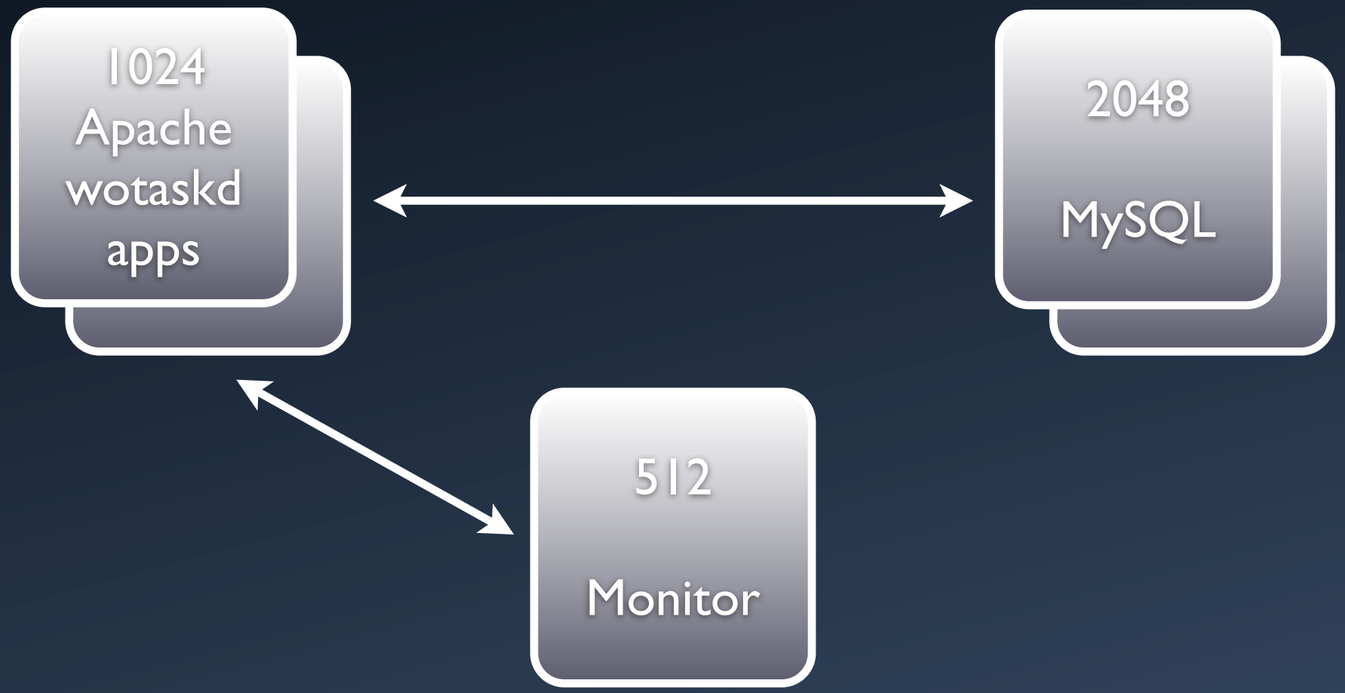
```
{
    name = "Root";
    children = "session.currentUser.isAdmin.toString";
    childrenChoices = {
        0 = ("Home","CurrentBand","Albums","Songs","Photos","Posts","Gigs");
        1 = ("Home","Bands","Members","Albums","Songs","Photos","Admin");
    };
},
```

Details - Deployment

Deployment - WO

- Deployment on Ubuntu (Linux)
- ImageMagic for image resize
- enhAacPlusEnc used to transcode HE AAC for streaming
- Conversions handled in background using ERXAsyncQueue

Deployment - WO



Lessons Learned

Lessons Learned

- When you load data is important
- Minimize startup times
- “Bake in” data for better first run experience
- All network access needs to be async
- Being pedantic about REST can get in the way of getting it done
- D2W is awesome

Q&A