Building a Server-Driven User Experience

Remote-controlled native UIs for fun and profit

Gregor Purdy
Engineer-at-Large

Fun and Profit

That’s some nice content you’ve got there…

“Hey, I know! Let’s build a mobile app…”

Translation key:

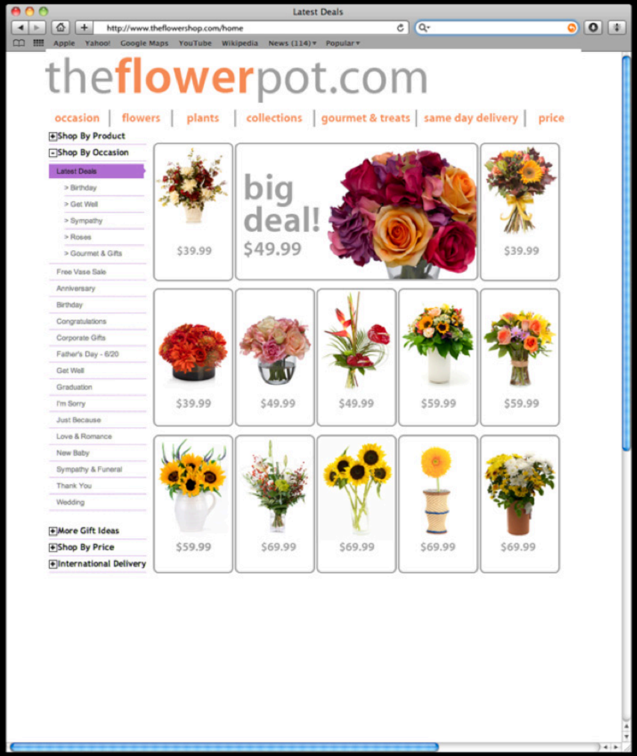
“Iterate the design”	Changing requirements
“Dedicated team”	Small team
“Window of opportunity”	Tight deadlines
“Lean”	You are paying for your own soda

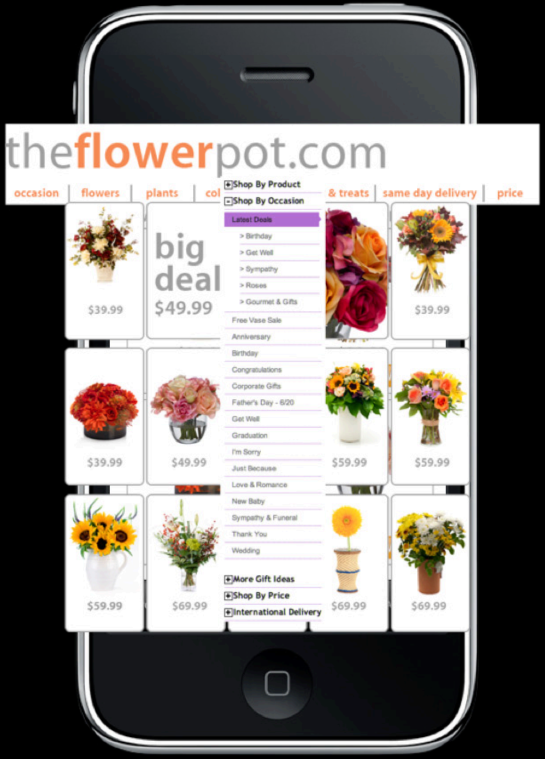
“…it’s going to be GREAT!”

Dream

Expand profit from existing content in a new context

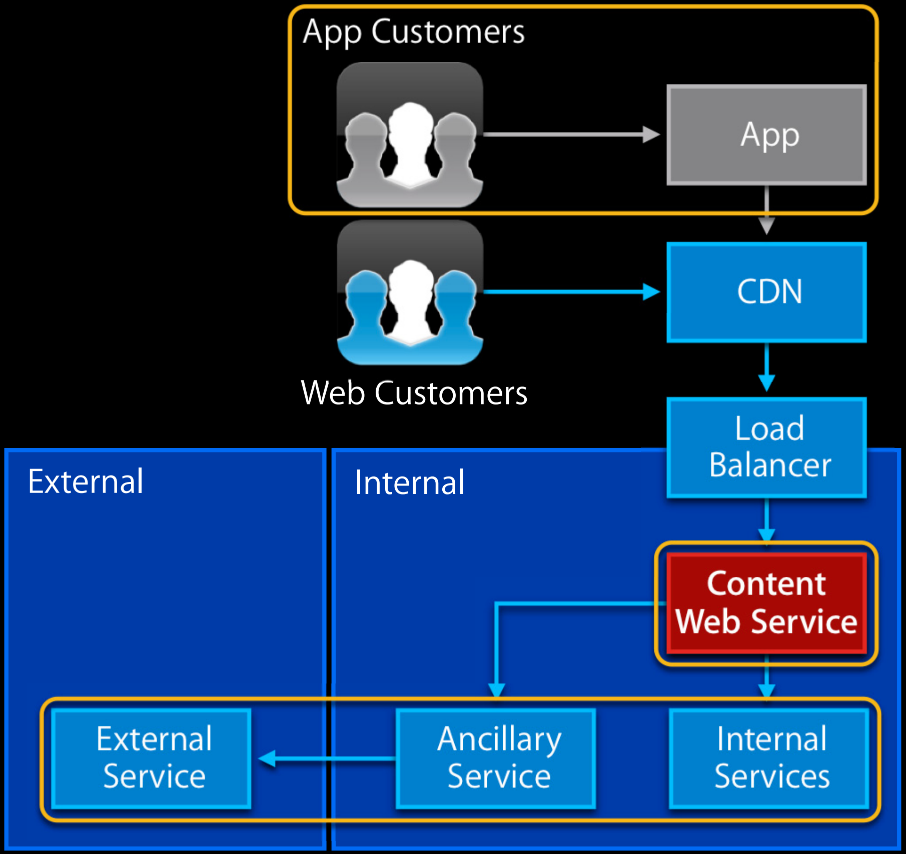
- Repurpose web content




theflowerpot.com
occasion | flowers | plants | col
& treats | same day delivery | price
big deal $49.99

Reality

Content Delivery Network



- Reduce cost and time-to-market by leveraging existing infrastructure
  - Services based architecture

Requirements

- Fresh content = Repeat use
    - Immediate content updates
    - Direct revenue
    - Ad revenue
- Agility
    - New content types
    - Don’t rev the app
    - Better for everyone

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

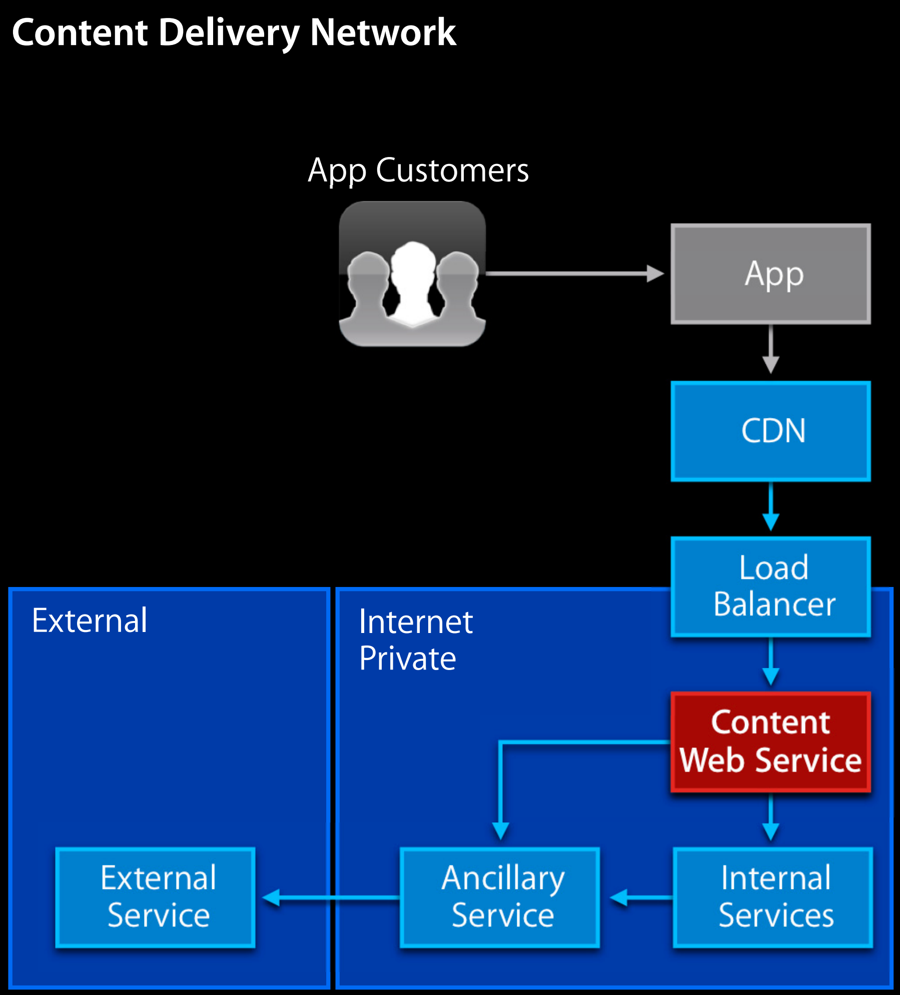
Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

Engineering a Solution

Design considerations

- Aspects of content
- Architectural options
- Responding to context
- Anticipating change



Design Considerations

Aspects of content

- User generated
- Curated
- Web service access
  - Just internal, or external too
- Variability by context
  - Translations
- Functions beyond read-only
  - Customer account management
  - Purchases

Design Considerations

Architectural options

- UIWebView—Fast time-to-market
  - Fancy shell around existing web content
  - Amazing HTML, JavaScript and CSS
- Native app—Better user experience
  - User expectations on navigation and “feel”
  - Stateful interactions
- Leverage native app capabilities
  - Coordinate calls to multiple web services
  - Custom rendering and animations
  - In-app purchases, camera, …

Design Considerations

Responding to context

- Selecting content
  - Device country and language
  - Core Location
  - Device type
  - App version
- One baked-in “config” URL
- Language fallback
- Add new translations
  - Try not to rev the app
  - Little “baked-in” content

Design Considerations

Anticipating change

- Freshness of content from service
- Caching app-side
  - HTTP headers
  - Custom
- Content Delivery Network (CDN)
  - Time to live in the cache
  - Standard HTTP cache control
- Emergency content updates
- Maintenance

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

Designing a Flexible Client

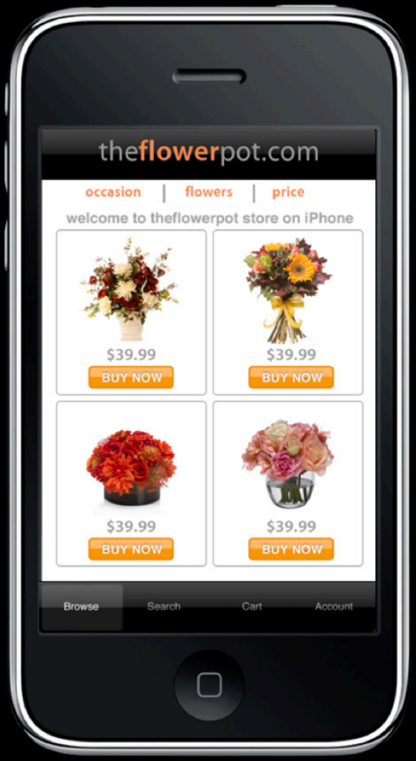
Scott Lopatin
Apple Store Engineering

Challenges

- Web is easy, universal updates to all
- Client is hard, can’t update everyone’s app
- Change data over time, don’t require updates
- We want our native UI

Solutions

- Property list control of UI
- Data that describes itself
- Handling dynamic data
- URL path generation
- Flexible categories
- Further optimizations



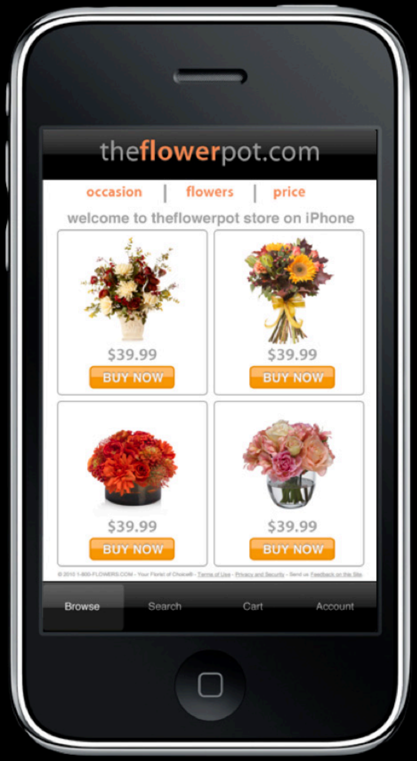
Solutions for Flexibility

Property list control

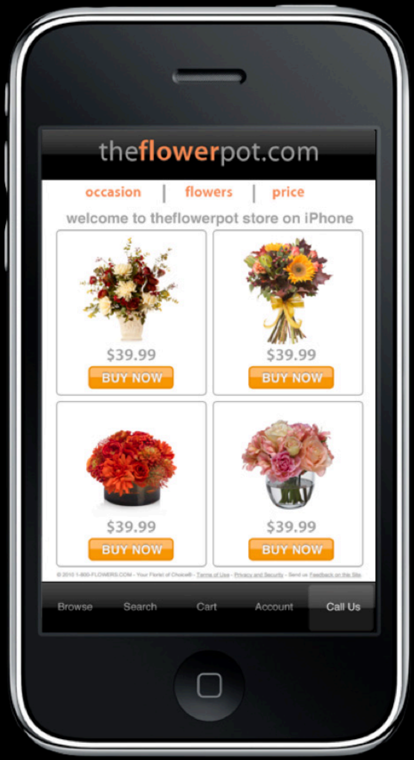
- Easy to generate from objects
- Works with many data types
- Platform independent
- One line to decode

Solutions for Flexibility

Property list control



```
...
    {
      title = "Terms & Conditions";
      action = viewcontroller;
      value = "WebViewController";
      path = “http://.../tsandcs.html
      type = link;
      fontStyle = bold;
      fontSize = 12;
      fontColor = "75:94:132:1.0";
      align = center;
      line = 3;
      height = 10;
    }
...
```


theflowerpot.com
occasion | flowers | price
welcome to theflowerpot store on iPhone
$39.99
BUY NOW
Browse
Search
Cart
Account
Call Us

Solutions for Flexibility

Property list control

```
...
      {
          device = "iphone";
          title = "Call Us";
          type = button;
          action = url;
          value = "tel:18005551212";
          fontStyle = bold;
          line = 1;
          height = 45;
      },
...
```

Solutions for Flexibility

Data that describes itself

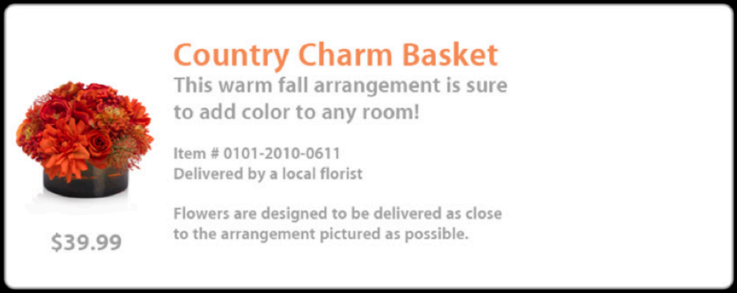
- Server side control of return paths
- Client routes to appropriate view controller
- Great for error handling

Solutions for Flexibility

Handling dynamic data

```
...
        content = {
            key = "value";
            key = "value";
            key = "value";
            key = "value";
        };
       datatype = “product”;
...
```

```
...
        content = {
            key = "value";
            key = "value";
            key = "value";
            key = "value";
        };
    datatype = “searchresults”;
...
```

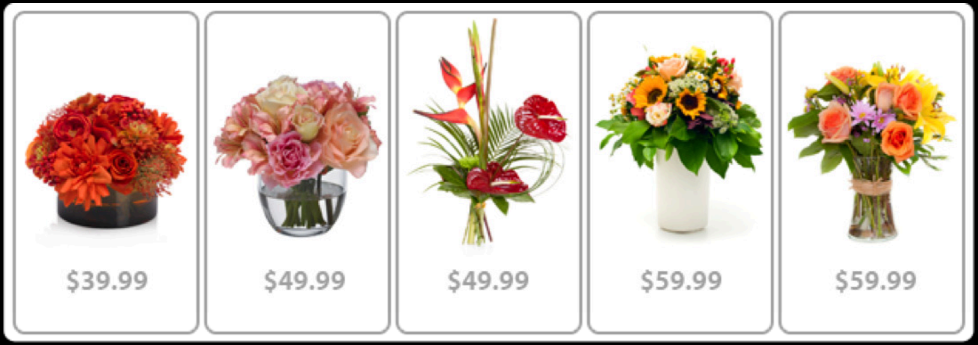


Country Charm Basket
This warm fall arrangement is sure to add color to any room!

Item # 0101-2010-0611
Delivered by a local florist

Flowers are designed to be delivered as close to the arrangement pictured as possible.

$39.99



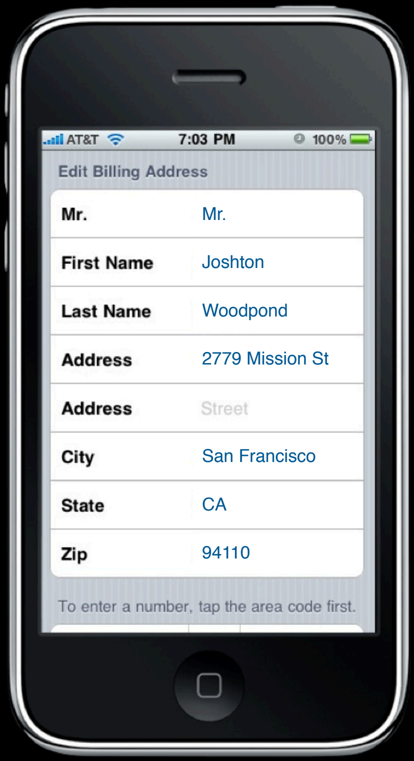
Solutions for Flexibility

Handling dynamic data

- Support datatypes with changing keys
- Write less code by iterating over results
- Still provide @dynamic ways to your data

Solutions for Flexibility

Handling dynamic data



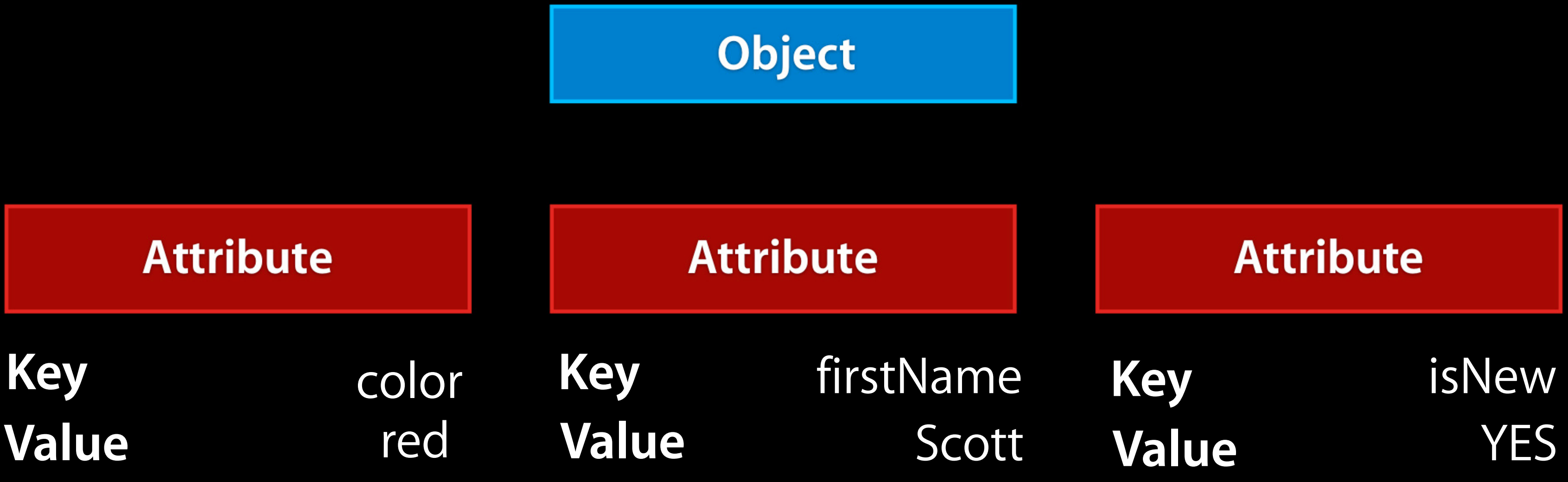
Edit Billing Address

Mr.	Mr.
First Name	Joshton
Last Name	Woodpond
Address	2779 Mission St
Address	Street
City	San Francisco
State	CA
Zip	94110

To enter a number, tap the area code first.

Solutions for Flexibility

Handling dynamic data



Key	color	**Key**	firstName	**Key**	isNew
Value	red	**Value**	Scott	**Value**	YES

Solutions for Flexibility

Handling dynamic data

- In methodSignatureForSelector

```
class_addMethod([self class], aSelector, (IMP)myGetImp, "@@:@");
```

- New method call

```
static id myGetImp(id self, SEL _cmd) {
     return [self valueForString:NSStringFromSelector(_cmd)];
}
```

- New method implementation

```
(id)valueForString:(NSString *)string {
for (Attribute *attribute in attributes) {
  if ([attribute.key isEqualToString:string]) {
    return attribute.value;
  } }
return nil;
}
```

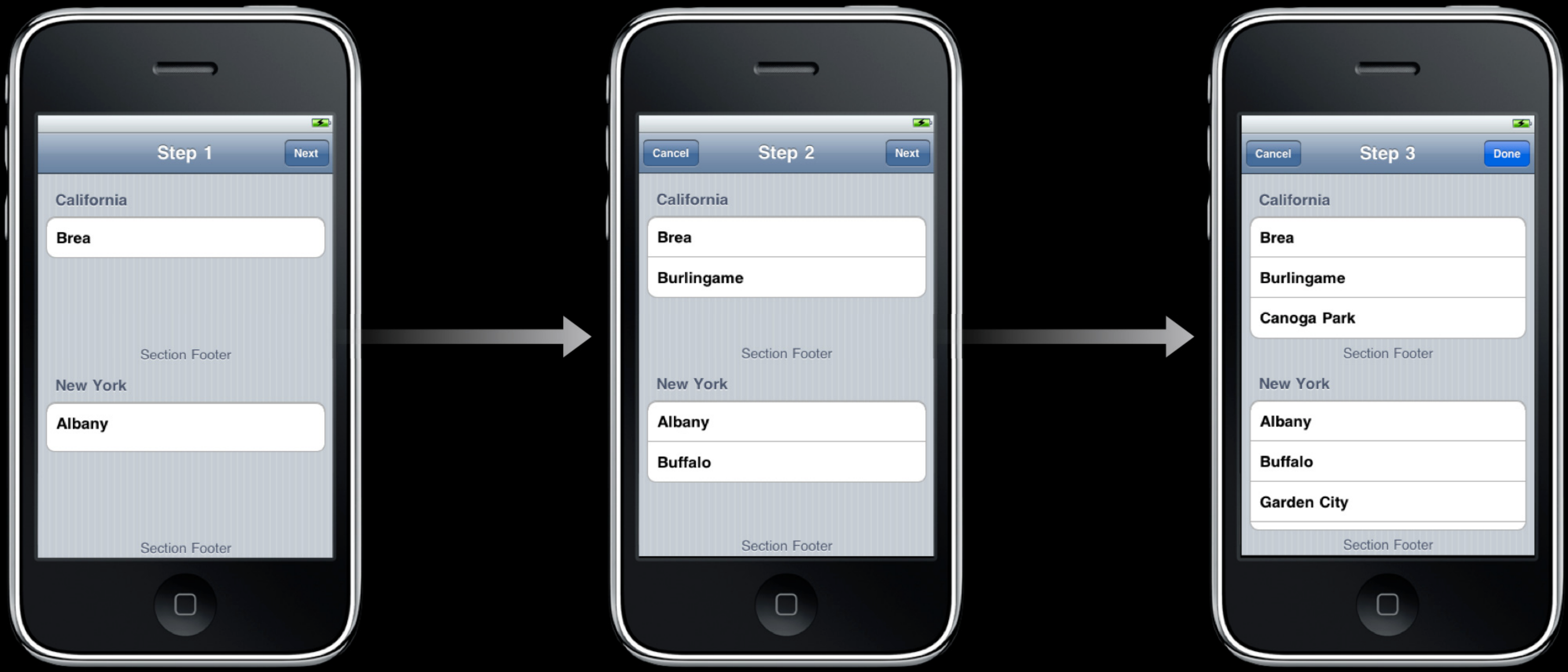
Solutions for Flexibility

URL path generation

- One URL to start a path
- Follow URL from response
- Add or remove steps from the server

Solutions for Flexibility

URL path generation



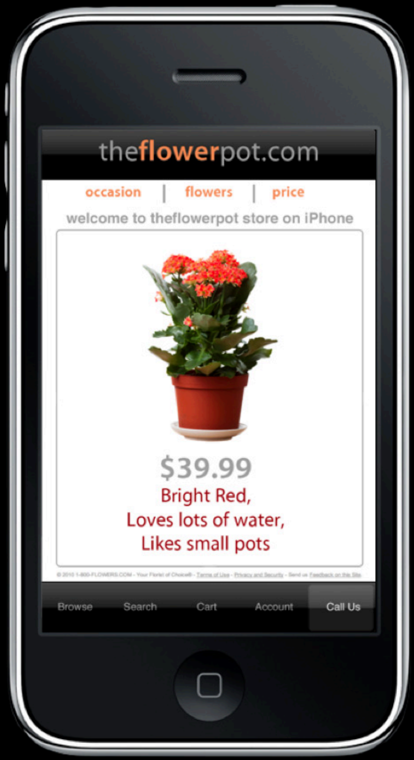
Solutions for Flexibility

URL path generation

```
...
       content = {
           key = "value";
           key = "value";
           key = "value";
       };
     continueURL = “http://www.apple.com/nextAction”;
     cancelURL = “http://www.apple.com/cancelAction”;
...
```

Solutions for Flexibility

Flexible UIKit categories



```
UIView-Extensions.h


-(void)layoutViews:(NSArray *)views
inRect:(CGRect)rect verticalPadding:
(CGFloat)padding shrinkToFit:(BOOL)
shrinkToFit;
```

Further Optimizations

Beyond remote controlled UI

- Optimized data types
- Server controlled expiration
- Persistence + HTTP cookies

Further Optimizations

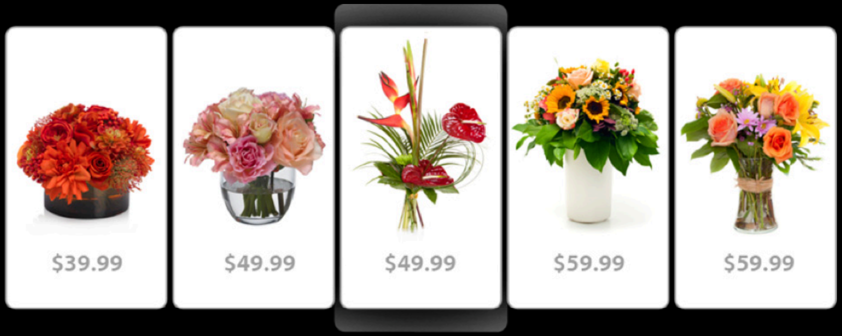
Optimized data types

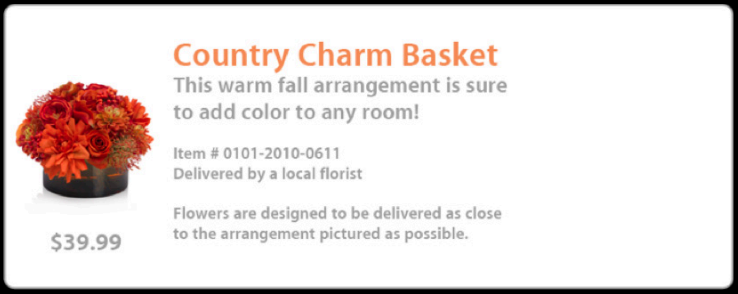
LightProduct

```
  {
name = “County Charm Basket”;
price = “$39.99”;
  },
```

HeavyProduct

```
  {
name = “County Charm Basket”;
price = “$39.99”;
rating = “5 Stars”;
description = “This warm fall
arrangement is sure to add color to
any room!”;
itemNo = “0101-2010-0611”;
  },
```





Country Charm Basket

This warm fall arrangement is sure to add color to any room!

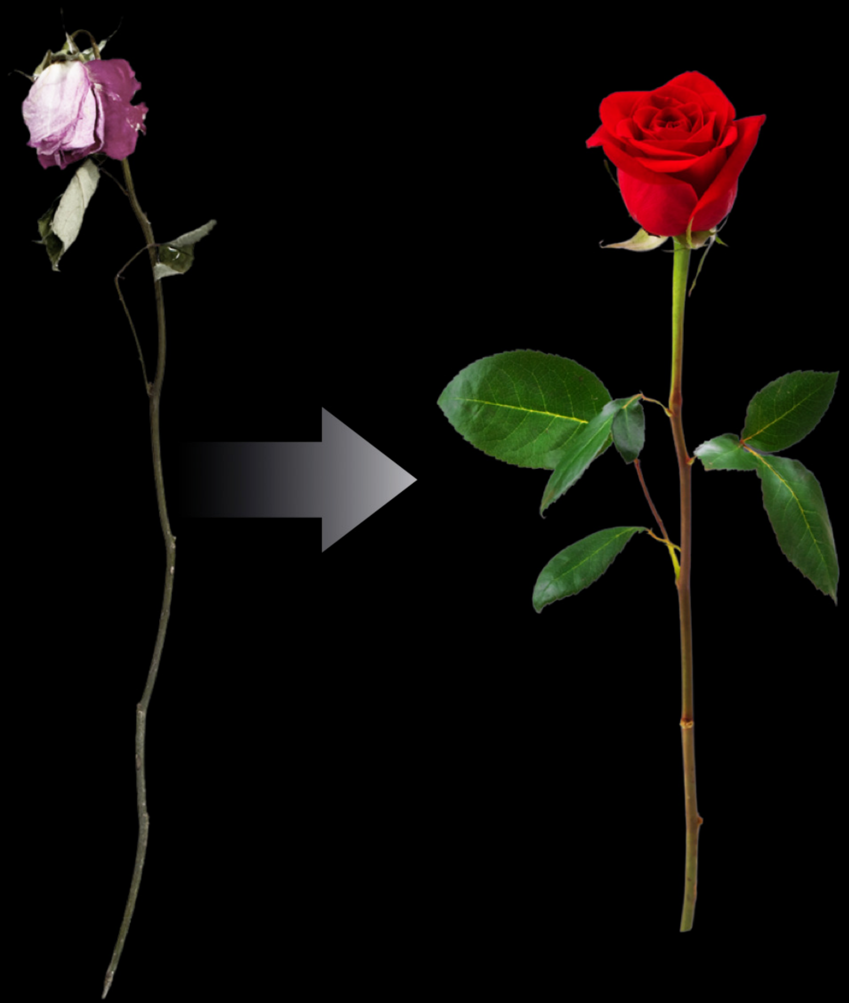
Item # 0101-2010-0611
Delivered by a local florist

Flowers are designed to be delivered as close to the arrangement pictured as possible.

$39.99

Further Optimizations

Server controlled expiration



```
{
  name = “County Charm Basket”;
  price = “$39.99”;
}
 _expire = “1274678220”;
```

Further Optimizations

Persistence + HTTP cookies

- Every request contains it
- No code to write
- Session vs. Persisted
- Automatically sandboxed

And Even After All That...

Sometimes you will need a binary update

Agenda

- Service oriented content delivery
    - Service orchestration
- Designing a flexible client
    - General enough to represent a variety of data types
    - “Remote controlled” native UI
- Core frameworks for data
    - Efficient server-client protocols
    - Remote data
- Lessons learned

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

Core Frameworks for Data

David den Boer
Engineering Manager, Retail Engineering

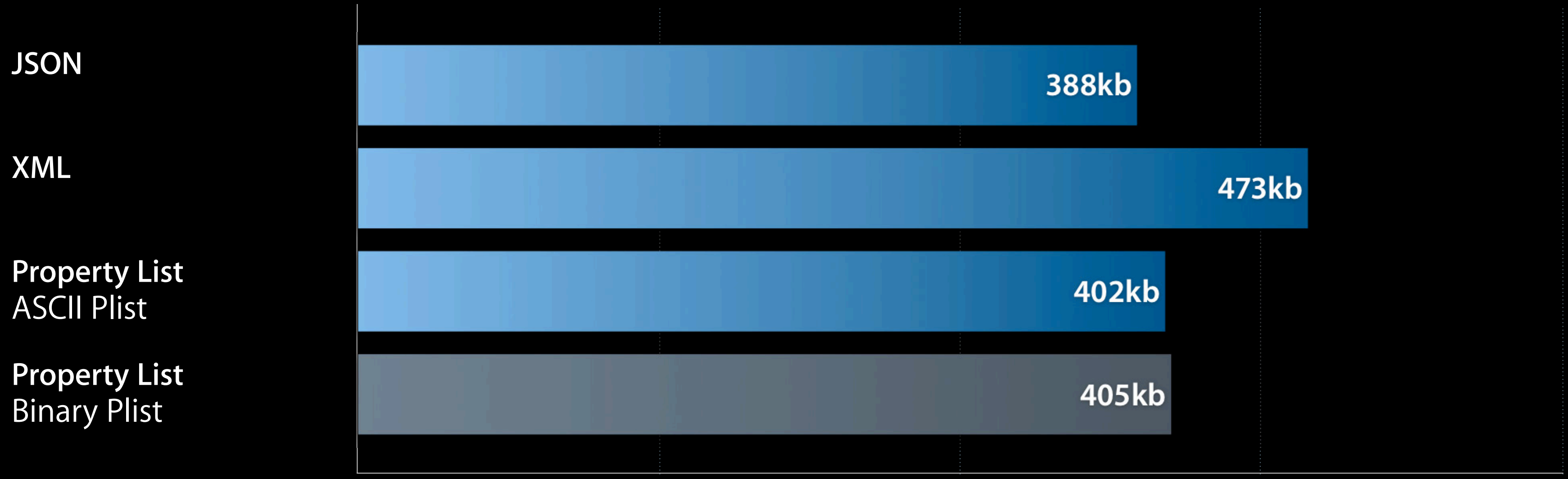
Core Frameworks for Data

Loading and utilizing remote data

- Remote data types
- Parsing data
- Client-side storage
- Benefits of Core Data
- Client/Server Data Store

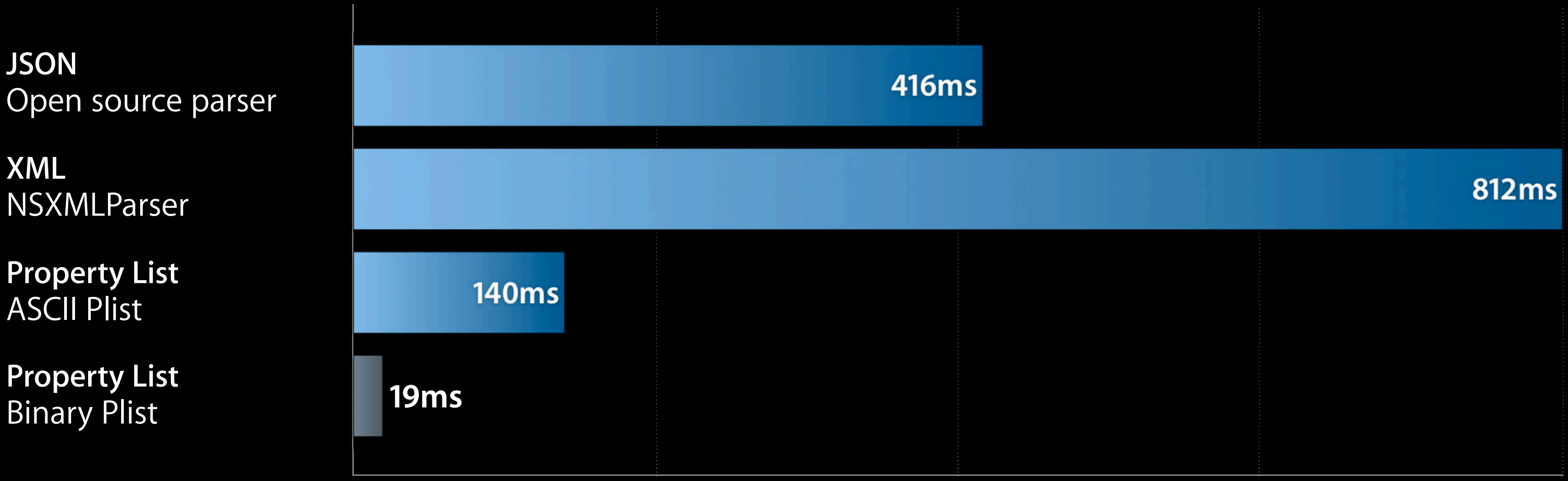
Remote Data Types

Data size



Remote Data Types

Parsing speed



Remote Data Types

Property lists

- Small data size
- Very fast parsing
- Easiest to create
  - With WebObjects, it is one line of code
  - CoreFoundation is open source
- Easiest to parse

```
+ (id)propertyListFromData:(NSData *)data
          mutabilityOption:(NSPropertyListMutabilityOptions)opt
                    format:(NSPropertyListFormat *)format
          errorDescription:(NSString **)errorString;
```

Client-side Storage

Available options

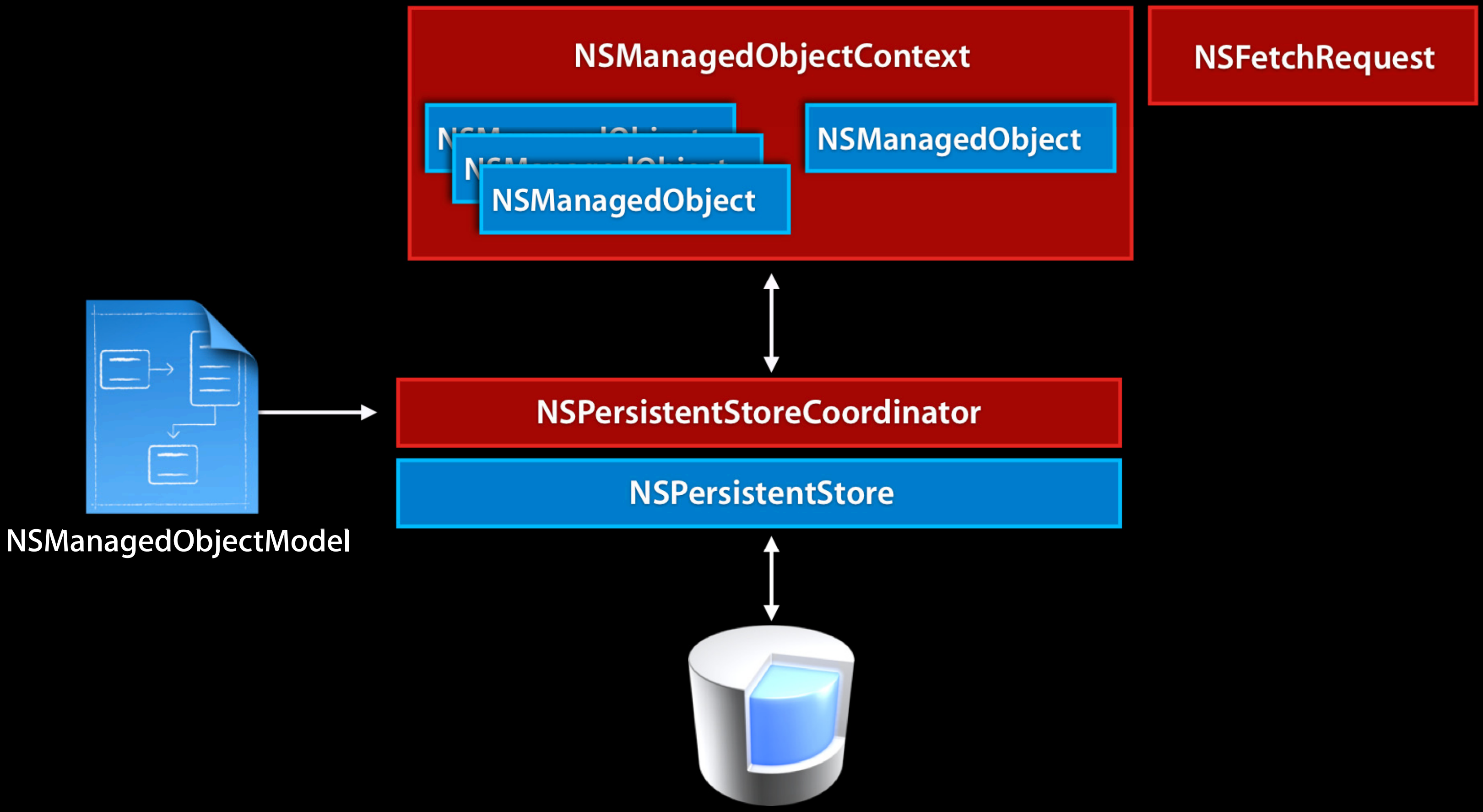
	NSDictionary	Data Objects	SQLite	CoreData
Pros	Simple	Simple Extensible	SQL Persistence	Simple Powerful Extensible Persistence
Cons	No persistence Unmanageable	No persistence	Complex	Not a database

Benefits of Core Data

Why use core data

- Persistence
    - Used for data that rarely changes
- Efficient fetching and saving
- Change tracking and Undo
- Object validation and relationship maintenance
- Supports KVC/KVO
- Performance

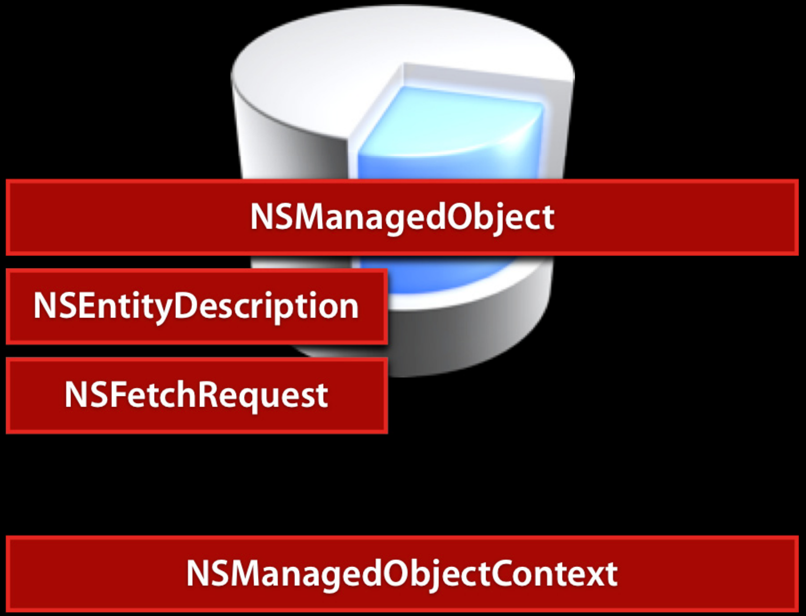
Core Data Architecture



Client/Server Data Store

Using Core Data for remote data storage

- Why?
  - Some data changes rarely
  - Ease of development
- What can it do?
  - Automatic fetching from server
  - Automatically propagates deletes to server
  - Automatically saves to server



Client/Server Data Store

How to…

- Start with a great foundation
- Subclass
- Add categories
- Update your model

Client/Server Data Store

Entities

- Entities can have server-side counterparts
- Entities can support up to four server-side operations
  - Fetch
  - Insert
  - Update
  - Delete

Client/Server Data Store

Entities

- Client/Server entities need helpers
  - operations
  - route
- NSEntityDescription category with methods for
  - route
  - shouldProcessInsert
  - shouldProcessUpdate
  - shouldProcessDelete

Client/Server Data Store

Managed objects

- NSManagedObjectContext subclass with override of:
  - save
  - executeFetchRequest:error:
- Category to create instance of object in different store:
  - localInstanceOfObject:
- NSManagedObject category which adds:
  - toDictionary
  - localInstanceInContext:

Client/Server Data Store

NSManagedObjectContext Category

```
- (NSManagedObject *)localInstanceOfObject:(NSManagedObject *)iObject {
    NSManagedObject *aReturnVal = nil;
    if (iObject && [iObject isKindOfClass:[NSManagedObject class]]) {
        if ([self objectRegisteredForID:[iObject objectID]]) {
            aReturnVal = iObject;
        } else {
            aReturnVal = [[NSManagedObject alloc] initWithEntity:[iObject entity]
insertIntoManagedObjectContext:self];
            for (NSString *aKey in [[[iObject entity] attributesByName] allKeys]) {
                id anObject = [iObject valueForKey:aKey];
                if (anObject) {
                    [aReturnVal setValue:anObject forKey:aKey];
                }
            }
        }
    }
    return [aReturnVal autorelease];
}
```

Client/Server Data Store

Fetching

- All fetches by default are client side
- Add endpoint binding to NSFetchRequest

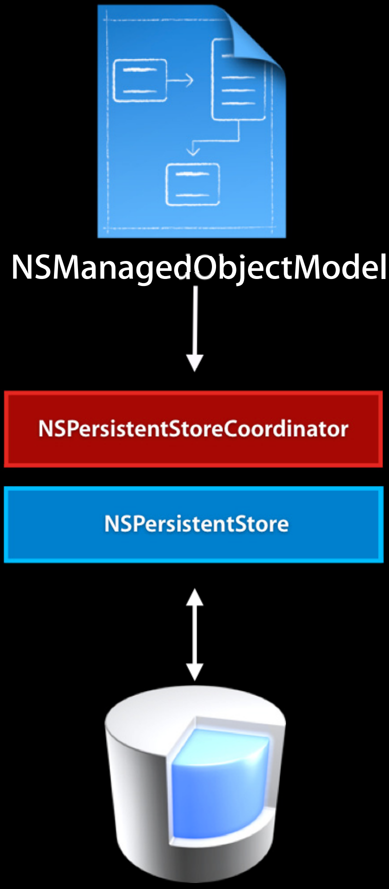
```
latitude == $LATITUDE AND longitude == $LONGITUDE AND endpoint == "nearbyStores"
```

- endpoint specifies method on this entity to execute
- Serialized NSFetchRequest properties include
    - expressions
    - sort orderings
    - fetch limits

Client/Server Data Store

Example fetch request

```
<dict>
    <key>predicate</key>
    <dict>
        <key>expression</key>
        <array>
            <dict>
                <key>key</key>
                <string>flowerID</string>
                <key>qualifier</key>
                <string>EQUALS</string>
                <key>value</key>
                <number>1231</number>
            </dict>
        </array>
    </dict>
```



Client/Server Core Data

Persistent stores

- Persistent or transient data?
- Multiple persistent stores
  - SQLite persistence
  - In-memory persistence
- Only one MOM required
  - Use localInstanceOfObject:

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

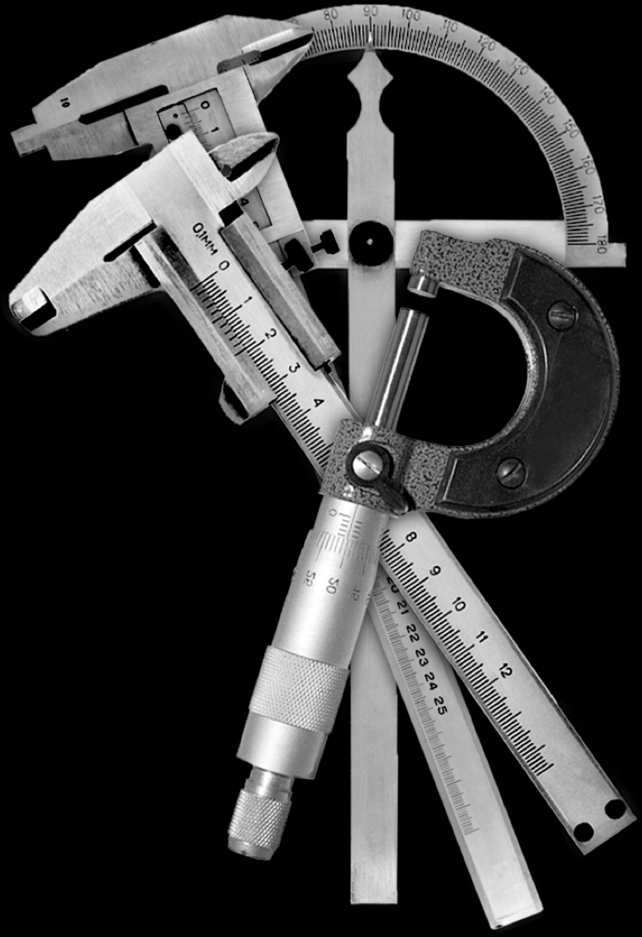
Lessons Learned

Gregor Purdy
Still Engineer-at-Large

Lessons Learned

Scaling beyond caching: Four “M’s”

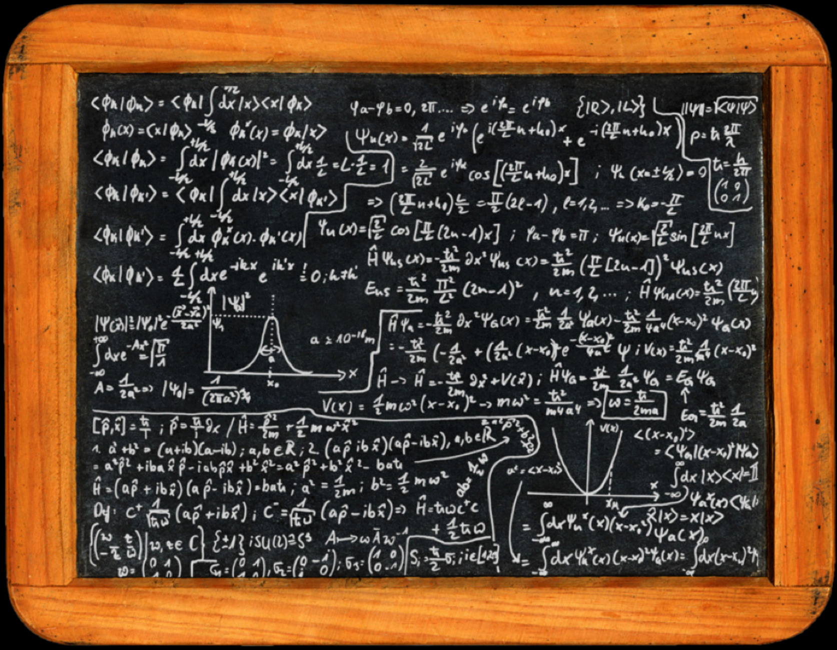
- Measure
- Model
- Monitor
- Message



Measure

First duty for performance

- Capture stats
- Allow introspection by internal caller



Model

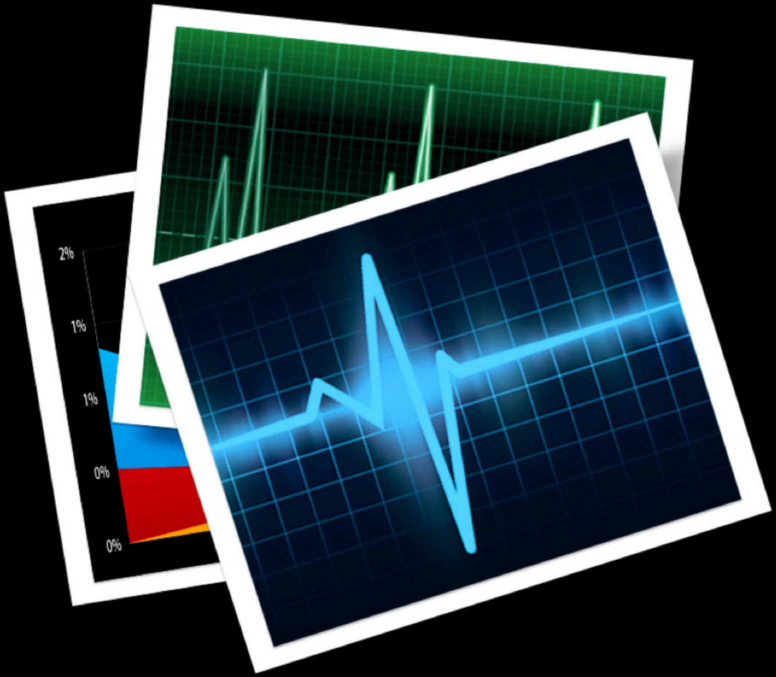
First duty for scale

- What is expected traffic, steady state and peak?
- What does that mean for different calls and their SLAs?

Monitor

First duty for operations

- Get stats and logs off the host into a monitoring system
- Capture history and show context in charts



Message

First duty for troubleshooting

```
2010-05-11 09:00:41:697|SyncServer|190|110f30|Server|Info| Cancelling all sync plans.
2010-05-11 09:00:41:714|SyncServer|190|110f30|Server|Info| Goodnight, Gracie.
2010-05-11 09:06:18:967|SyncServer|341|110f30|Logging|Info| Logging initialized, engine version 578.3 : log level 3
2010-05-11 09:06:19:006|SafariSyncClient|340|111ae0|ISyncManager|Info| unregister com.apple.Safari
2010-05-11 09:06:19:011|SyncServer|341|110f30|Server|Info| unregister client com.apple.Safari
2010-05-11 09:06:19:018|SafariSyncClient|340|111ae0|ISyncManager|Info| register client com.apple.Safari from /
2010-05-13 14:36:25:586|SyncServer|522|110f00|Logging|Info| Logging initialized, engine version 578.3 : log level 3
2010-05-13 14:36:29:435|AddressBookSync|520|111380|Stats|Stats| com.apple.AddressBook: refresh sync com.apple.contacts.
2010-05-13 14:36:29:435|AddressBookSync|520|111380|ISyncSession|Info| com.apple.AddressBook: prepare to refresh com.
2010-05-13 14:38:30:691|SyncServer|522|110f00|Server|Info| Cancelling all sync plans.
2010-05-13 14:38:30:697|SyncServer|522|110f00|Server|Info| Goodnight, Gracie.
2010-05-13 15:53:25:922|SyncServer|4566|110f00|Logging|Info| Logging initialized, engine version 578.3 : log level 3
2010-05-13 15:55:26:186|SyncServer|4566|110f00|Server|Info| Cancelling all sync plans.
2010-05-13 15:55:26:201|SyncServer|4566|110f00|Server|Info| Goodnight, Gracie.
2010-05-13 15:56:19:230|SyncServer|4576|110f00|Logging|Info| Logging initialized, engine version 578.3 : log level 3
2010-05-13 15:56:19:396|DockSyncClient|4577|110300|Stats|Stats| com.apple.Dock: slow sync com.apple.dock.item, com.apple.
2010-05-13 15:56:19:396|DockSyncClient|4577|110300|ISyncSession|Info| com.apple.Dock: prepare to slow sync com.apple.dock.
2010-05-13 15:57:47:955|DockSyncClient|4597|110300|Stats|Stats| com.apple.Dock: slow sync com.apple.dock.item, com.apple.
2010-05-13 15:57:47:955|DockSyncClient|4597|110300|ISyncSession|Info| com.apple.Dock: prepare to slow sync com.apple.dock.
2010-05-13 15:59:48:057|SyncServer|4576|110f00|Server|Info| Cancelling all sync plans.
2010-05-13 15:59:48:058|SyncServer|4576|110f00|Server|Info| Goodnight, Gracie.
2010-05-13 16:03:10:639|SyncServer|4721|110f00|Logging|Info| Logging initialized, engine version 578.3 : log level 3
2010-05-13 16:03:10:777|TransmitSync|4720|10a060|ISyncManager|Info| register client com.panic.TransmitSync.Favorites from
2010-05-13 16:05:10:861|SyncServer|4721|110f00|Server|Info| Cancelling all sync plans.
2010-05-13 16:05:10:862|SyncServer|4721|110f00|Server|Info| Goodnight, Gracie.
2010-05-16 09:27:27:607|SyncServer|980|6a209e0|Logging|Info| Logging initialized, engine version 578.3 : log level 3
2010-05-16 09:27:28:082|DockSyncClient|981|6a1fdd0|Stats|Stats| com.apple.Dock: slow sync com.apple.dock.item, com.apple.
2010-05-16 09:27:28:083|DockSyncClient|981|6a1fdd0|ISyncSession|Info| com.apple.Dock: prepare to slow sync com.apple.dock.
2010-05-16 09:27:37:711|DockSyncClient|988|6a1fdd0|Stats|Stats| com.apple.Dock: slow sync com.apple.dock.item, com..
2010-05-16 09:27:37:711|DockSyncClient|988|6a1fdd0|ISyncSession|Info| com.apple.Dock: prepare to slow sync com.apple.
2010-05-16 09:29:37:818|SyncServer|980|6a209e0|Server|Info| Cancelling all sync plans.
2010-05-16 09:29:37:832|SyncServer|980|6a209e0|Server|Info| Goodnight, Gracie.
2010-06-01 09:24:16:162|SyncServer|358|6a209e0|Logging|Info| Logging initialized, engine version 578.3 : log level 3
```

- The logs should contain session and other identifiers and actual values participating in the situation being logged

Agenda

- Service oriented content delivery
  - Service orchestration
- Designing a flexible client
  - General enough to represent a variety of data types
  - “Remote controlled” native UI
- Core frameworks for data
  - Efficient server-client protocols
  - Remote data
- Lessons learned

More Information

Mark Malone

Integration Technologies Evangelist
mgm@apple.com

iPhone Documentation
http://developer.apple.com/iphone

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Mastering Core Data	Russian Hill Wednesday 2:00PM
Network Apps for iPhone OS, Part 1	Pacific Heights Wednesday 2:00PM
Network Apps for iPhone OS, Part 2	Pacific Heights Wednesday 3:15PM
Crafting Custom Cocoa Views	Russian Hill Friday 10:15AM

Labs

Server-Driven User Experience Lab	Application Frameworks Lab A Wednesday 12:45 -1:45PM
Multitasking Lab	Application Frameworks Lab D Tuesday, 4:30PM – 6:30PM
Enterprise and In-House Development Lab	Application Frameworks Lab D Thursday, 11:30AM – 1:45PM

Q&A

WWDC10

[redacted]