CS193P - Lecture 9

iPhone Application Development

Data in Your iPhone App

Chris Marcellino

Today's Topics

- Data in Your iPhone App
  - Saving & loading **local data**
  - Accessing **remote data** over the Internet

Today's Topics

- Property Lists, NSUserDefaults and Settings
- iPhone's File System
- Archiving Objects
- The Joy of SQLite
- JSON
- Apple Push Notification Service

Property Lists

Property Lists

- Convenient way to store a **small amount of data**
  - Arrays, dictionaries, strings, numbers, dates, raw data
  - Human-readable XML or binary format
- NSUserDefaults class uses property lists under the hood

When Not to Use Property Lists

- More than a few hundred KB of data
  - Loading a property list is all-or-nothing
- Complex object graphs
- Custom object types
- Multiple writers (e.g. not ACID)

Reading & Writing Property Lists

- NSArray and NSDictionary convenience methods
- Operate recursively

```
// Writing
- (BOOL)writeToFile:(NSString *)aPath atomically:(BOOL)flag;
- (BOOL)writeToURL:(NSURL *)aURL atomically:(BOOL)flag;

// Reading
- (id)initWithContentsOfFile:(NSString *)aPath;
- (id)initWithContentsOfURL:(NSURL *)aURL;
```

Writing an Array to Disk

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
```

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
                  [NSNumber numberWithBool:YES],
```

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
                  [NSNumber numberWithBool:YES],
                  [NSDate dateWithTimeIntervalSinceNow:60],
```

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
                  [NSNumber numberWithBool:YES],
                  [NSDate dateWithTimeIntervalSinceNow:60],
                  nil];
```

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
                  [NSNumber numberWithBool:YES],
                  [NSDate dateWithTimeIntervalSinceNow:60],
                  nil];
[array writeToFile:@“MyArray.plist” atomically:YES];
```

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
                  [NSNumber numberWithBool:YES],
                  [NSDate dateWithTimeIntervalSinceNow:60],
                  nil];
[array writeToFile:@“MyArray.plist” atomically:YES];
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
     <string>Foo</string>
     <true/>
     <date>2010-02-02T09:26:18Z</date>
</array>
</plist>
```

Writing a Dictionary to Disk

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
```

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
                     @“Bob”, @“Name”,
```

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
                      @“Bob”, @“Name”,
                      [NSNumber numberWithInt:9], @“Lecture”,
```

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
                      @“Bob”, @“Name”,
                      [NSNumber numberWithInt:9], @“Lecture”,
                      nil];
```

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
                      @“Bob”, @“Name”,
                      [NSNumber numberWithInt:9], @“Lecture”,
                      nil];
[dict writeToFile:@“MyDict.plist” atomically:YES];
```

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
                        @“Bob”, @“Name”,
                        [NSNumber numberWithInt:9], @“Lecture”,
                        nil];
[dict writeToFile:@“MyDict.plist” atomically:YES];
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
     <key>Name</key>
     <string>Bob</string>
     <key>Lecture</key>
     <integer>10</integer>
</dict>
</plist>
```

NSPropertyListSerialization

- Allows finer-grained control
  - File format
  - More descriptive errors
  - Mutability

NSPropertyListSerialization

- Allows finer-grained control
  - File format
  - More descriptive errors
  - Mutability

```
// Property list to NSData
+ (NSData *)dataFromPropertyList:(id)plist
                          format:(NSPropertyListFormat)format
                errorDescription:(NSString **)errorString;

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

More on Property Lists

- “Property List Programming Guide for Cocoa” http://developer.apple.com/documentation/Cocoa/Conceptual/PropertyLists/

iPhone’s File System

Keeping Applications Separate

Image (cc) by davidsilver on Flickr

Why Keep Applications Separate?

- Security
- Privacy
- Cleanup after deleting an app

Home Directory Layout

Home Directory Layout

- Each app has its **own set of directories**

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
    - MyApp.app

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
    - MyApp.app
        - MyApp

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
    - MyApp.app
        - MyApp
        - MainWindow.nib

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
  - MyApp.app
    - MyApp
    - MainWindow.nib
    - SomeImage.png

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
    - MyApp.app
        - MyApp
        - MainWindow.nib
        - SomeImage.png
    - Documents

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
    - MyApp.app
        - MyApp
        - MainWindow.nib
        - SomeImage.png
    - Documents
    - Library

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
  - MyApp.app
    - MyApp
    - MainWindow.nib
    - SomeImage.png
  - Documents
  - Library
    - Caches

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
    - MyApp.app
        - MyApp
        - MainWindow.nib
        - SomeImage.png
    - Documents
    - Library
        - Caches
        - Preferences

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
  - MyApp.app
    - MyApp
    - MainWindow.nib
    - SomeImage.png
  - Documents
  - Library
    - Caches
    - Preferences
- Applications only read and write within their home directory

Home Directory Layout

- Each app has its **own set of directories**
- <Application Home>
  - MyApp.app
    - MyApp
    - MainWindow.nib
    - SomeImage.png
  - Documents
  - Library
    - Caches
    - Preferences
- Applications only read and write within their home directory
- Backed up by iTunes during sync (mostly)

File Paths in Your Application

File Paths in Your Application

```
// Basic directories
NSString *homePath = NSHomeDirectory();
NSString *tmpPath = NSTemporaryDirectory();
```

File Paths in Your Application

```
// Basic directories
NSString *homePath = NSHomeDirectory();
NSString *tmpPath = NSTemporaryDirectory();

// Documents directory
NSArray *paths = NSSearchPathForDirectoriesInDomains
                 (NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsPath = [paths objectAtIndex:0];
```

File Paths in Your Application

```
// Basic directories
NSString *homePath = NSHomeDirectory();
NSString *tmpPath = NSTemporaryDirectory();

// Documents directory
NSArray *paths = NSSearchPathForDirectoriesInDomains
                 (NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsPath = [paths objectAtIndex:0];


// <Application Home>/Documents/foo.plist
NSString *fooPath =
[documentsPath stringByAppendingPathComponent:@“foo.plist”];
```

Including Writable Files with Your App

- Many applications want to include some starter data
- But application bundles are code signed
  - You can’t modify the contents of your app bundle
- To include a writable data file with your app...
  - Build it as part of your app bundle
  - On first launch, **copy it to your Documents directory**

Archiving Objects

Archiving Objects

- Next logical step from property lists
  - Include arbitrary classes
  - Complex object graphs
- Used by Interface Builder for NIBs

Making Objects Archivable

- Conform to the <NSCoding> protocol

Making Objects Archivable

- Conform to the <NSCoding> protocol

```
// Encode an object for an archive
- (void)encodeWithCoder:(NSCoder *)coder
{
  [super encodeWithCoder:coder];
  [coder encodeObject:name forKey:@“Name”];
  [coder encodeInteger:numberOfSides forKey:@“Sides”];
}

// Decode an object from an archive
- (id)initWithCoder:(NSCoder *)coder
{
  self = [super initWithCoder:coder];
  name = [[coder decodeObjectForKey:@“Name”] retain];
  numberOfSides = [coder decodeIntegerForKey:@“Side”];
}
```

Archiving & Unarchiving Object Graphs

Archiving & Unarchiving Object Graphs

- Creating an archive

```
NSArray *polygons = ...;
NSString *path = ...;
BOOL result = [NSKeyedArchiver archiveRootObject:polygons
                                          toFile:path];
```

Archiving & Unarchiving Object Graphs

- Creating an archive

```
NSArray *polygons = ...;
NSString *path = ...;
BOOL result = [NSKeyedArchiver archiveRootObject:polygons
                                           toFile:path];
```

- Decoding an archive

```
NSArray *polygons = nil;
NSString *path = ...;
polygons = [NSKeyedUnarchiver unarchiveObjectWithFile:path];
```

More on Archiving Objects

- “Archives and Serializations Programming Guide for Cocoa” http://developer.apple.com/documentation/Cocoa/Conceptual/Archiving/

The Joy of SQLite

SQLite

- Complete SQL database in an ordinary file
- Simple, compact, fast, reliable
- No server
- Free/Open Source Software
- Great for embedded devices
  - Included on the iPhone platform

When Not to Use SQLite

- Multi-gigabyte databases
- High concurrency (multiple writers)
- Client-server applications
- “Appropriate Uses for SQLite”
  http://www.sqlite.org/whentouse.html

SQLite C API Basics

SQLite C API Basics

- Open the database

```
int sqlite3_open(const char *filename, sqlite3 **db);
```

SQLite C API Basics

- Open the database

```
int sqlite3_open(const char *filename, sqlite3 **db);
```

- Execute a SQL statement

```
int sqlite3_exec(sqlite3 *db, const char *sql,
                 int (*callback)(void*,int,char**,char**),
                 void *context, char **error);
```

SQLite C API Basics

- Open the database

```
int sqlite3_open(const char *filename, sqlite3 **db);
```

- Execute a SQL statement

```
int sqlite3_exec(sqlite3 *db, const char *sql,
                 int (*callback)(void*,int,char**,char**),
                 void *context, char **error);

// Your callback
int callback(void *context, int count,
             char **values, char **columns);
```

SQLite C API Basics

- Open the database

```
int sqlite3_open(const char *filename, sqlite3 **db);
```

- Execute a SQL statement

```
int sqlite3_exec(sqlite3 *db, const char *sql,
                 int (*callback)(void*,int,char**,char**),
                 void *context, char **error);

// Your callback
int callback(void *context, int count,
             char **values, char **columns);
```

- Close the database

```
int sqlite3_close(sqlite3 *db);
```

Demo:
Simple SQLite

More on SQLite

- "SQLite in 5 Minutes Or Less"
  http://www.sqlite.org/quickstart.html
- "Intro to the SQLite C Interface"
  http://www.sqlite.org/cintro.html

Core Data

- Object-graph management and persistence framework
  - Makes it easy to save & load model objects
    - Properties
    - Relationships
  - Higher-level abstraction than SQLite or property lists
- Available on the Mac OS X desktop
- Now available on iPhone OS 3.0

Two classes you should know about...

Two classes you should know about...

- NSPredicate

Two classes you should know about...

- NSPredicate
    - "Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering."

Two classes you should know about...

- NSPredicate
  - "Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering."
  - -[NSPredicate predicateWithFormat:]

Two classes you should know about...

- NSPredicate
  - "Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering."
  - -[NSPredicate predicateWithFormat:]
  - Simple comparisons:

Two classes you should know about...

- NSPredicate
  - “Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering.”
  - -[NSPredicate predicateWithFormat:]
  - Simple comparisons:
    - *grade == “7”*

Two classes you should know about...

- NSPredicate
  - “Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering.”
  - -[NSPredicate predicateWithFormat:]
  - Simple comparisons:
    - *grade == “7”*
    - *user.firstName like "Tom"*

Two classes you should know about...

- NSPredicate
  - “Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering.”
  - -[NSPredicate predicateWithFormat:]
  - Simple comparisons:
    - *grade == “7”*
    - *user.firstName like "Tom"*
    - *“first contains [c]”chris”*

Two classes you should know about...

- NSPredicate
  - “Used to define logical conditions used to constrain a search either for a fetch or for in-memory filtering.”
  - -[NSPredicate predicateWithFormat:]
  - Simple comparisons:
    - *grade == “7”*
    - *user.firstName like "Tom"*
    - *“first contains [c]”chris”*
  - Many, many options: http://developer.apple.com/mac/library/documentation/cocoa/Conceptual/Predicates/Articles/pSyntax.html

Two classes you should know about...

Two classes you should know about...

- NSEntityDescription

Two classes you should know about...

- NSEntityDescription
  - Used for inserting a new object into a Core Data Managed Object context

Two classes you should know about...

- NSEntityDescription
  - Used for inserting a new object into a Core Data Managed Object context
  - *- [NSEntityDescription insertNewObjectForEntityForName:inManagedObjectContext:]*

Two classes you should know about...

- NSEntityDescription
  - Used for inserting a new object into a Core Data Managed Object context
  - *- [NSEntityDescription insertNewObjectForEntityForName:inManagedObjectContext:]*
  - See the documentation!

Two classes you should know about...

- NSEntityDescription
  - Used for inserting a new object into a Core Data Managed Object context
  - *- [NSEntityDescription insertNewObjectForEntityForName:inManagedObjectContext:]*
  - See the documentation!
    - http://developer.apple.com/mac/library/documentation/cocoa/reference/CoreDataFramework/Classes/NSEntityDescription_Class/NSEntityDescription.html

Web Services

Your Application & The Cloud

- Store & access remote data
- May be under your control or someone else's
- Many Web 2.0 apps/sites provide developer API

“I made a location-based user-generated video blogging mashup... for pets!”

Integrating with Web Services

- **Non-goal** of this class: teach you all about web services
  - Plenty of tutorials accessible, search on Google
- Many are exposed via RESTful interfaces with XML or JSON
  - **RE**presentational **S**tate **T**ransfer
    - Stateless interactions
    - Well defined client/server roles & interfaces
    - e.g. HTTP
- High level overview of parsing these types of data

XML

Options for Parsing XML

- libxml2
    - Tree-based: easy to parse, entire tree in memory
    - Event-driven: less memory, more complex to manage state
    - Text reader: fast, easy to write, efficient
- NSXMLParser
    - Event-driven API: simpler but less powerful than libxml2

More on Parsing XML

- Brent Simmons, “libxml2 + xmlTextReader on Macs” http://inessential.com/?comments=1&postid=3489
  - Includes example of parsing Twitter XML!
- Big Nerd Ranch, “Parsing XML in Cocoa” http://weblog.bignerdranch.com/?p=48
  - Covers the basics of NSXMLReader

JSON

JavaScript Object Notation

- More lightweight than XML
- Looks a lot like a property list
    - Arrays, dictionaries, strings, numbers
- Open source json-framework wrapper for Objective-C

What does a JSON object look like?

What does a JSON object look like?

```
{
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
  “itunes-u” : true,
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
  “itunes-u” : true,
  “midterm-exam” : null,
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
  “itunes-u” : true,
  “midterm-exam” : null,
  “assignments” : [ “WhatATool”,
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
  “itunes-u” : true,
  “midterm-exam” : null,
  “assignments” : [ “WhatATool”,
                    “HelloPoly”,
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
  “itunes-u” : true,
  “midterm-exam” : null,
  “assignments” : [ “WhatATool”,
                    “HelloPoly”,
                    “Presence” ]
```

What does a JSON object look like?

```
{
  “instructor” : “Josh Shaffer”,
  “students” : 60,
  “itunes-u” : true,
  “midterm-exam” : null,
  “assignments” : [ “WhatATool”,
                    “HelloPoly”,
                    “Presence” ]
}
```

Using json-framework

- Reading a JSON string into Foundation objects

Using json-framework

- Reading a JSON string into Foundation objects

```
#import <JSON/JSON.h>
```

Using json-framework

- Reading a JSON string into Foundation objects

```
#import <JSON/JSON.h>

// Get a JSON string from the cloud
NSString *jsonString = ...;
```

Using json-framework

- Reading a JSON string into Foundation objects

```
#import <JSON/JSON.h>

// Get a JSON string from the cloud
NSString *jsonString = ...;

// Parsing will result in Foundation objects
// Top level may be an NSDictionary or an NSArray
id object = [jsonString JSONValue];
```

Using json-framework

- Writing a JSON string from Foundation objects

Using json-framework

- Writing a JSON string from Foundation objects

```
// Create some data in your app
```

Using json-framework

- Writing a JSON string from Foundation objects

```
// Create some data in your app
NSDictionary *dictionary = ...;

// Convert into a JSON string before sending to the cloud
```

Using json-framework

- Writing a JSON string from Foundation objects

```
// Create some data in your app
NSDictionary *dictionary = ...;

// Convert into a JSON string before sending to the cloud
jsonString = [dictionary JSONRepresentation];
```

Demo:
Flickr API with JSON

More on JSON

- "JSON Parser/Generator for Objective-C"
  http://code.google.com/p/json-framework/
- "Introducing JSON"
  http://www.json.org/

Apple Push Notification Service

Overview

Overview

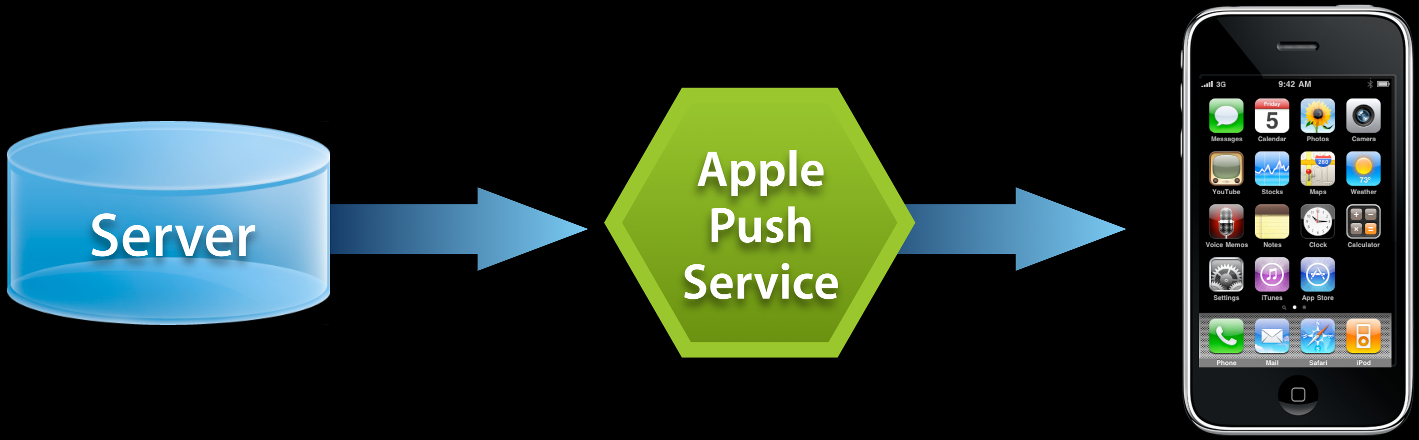
- Show badges, alerts and play sounds without app running

Overview

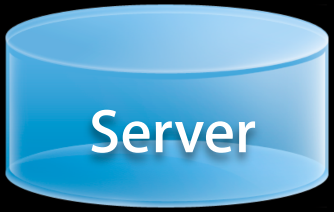
- Show badges, alerts and play sounds without app running
- Minimal server infrastructure needed

Overview

- Show badges, alerts and play sounds without app running
- Minimal server infrastructure needed
- Preserves battery life: 1 versus *n* TCP/IP connections

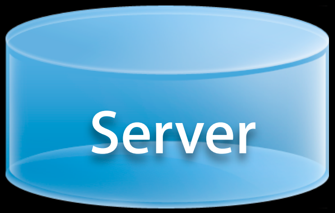


Using the Service



Using the Service

What you need



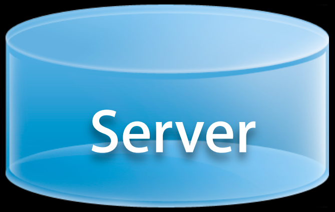
Using the Service

What you need

Server
Certificate
Standard

Using the Service

What you need



`edu.stanford.cs193.app`

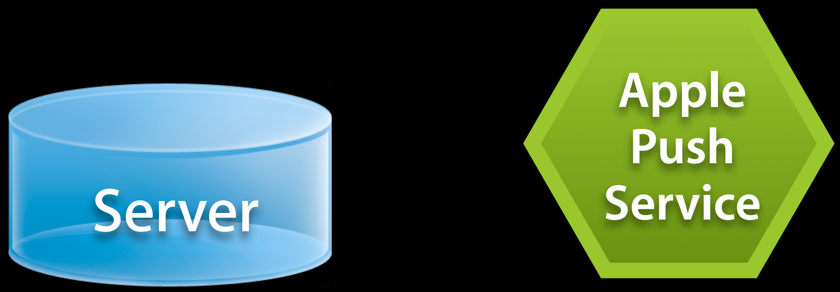
Using the Service

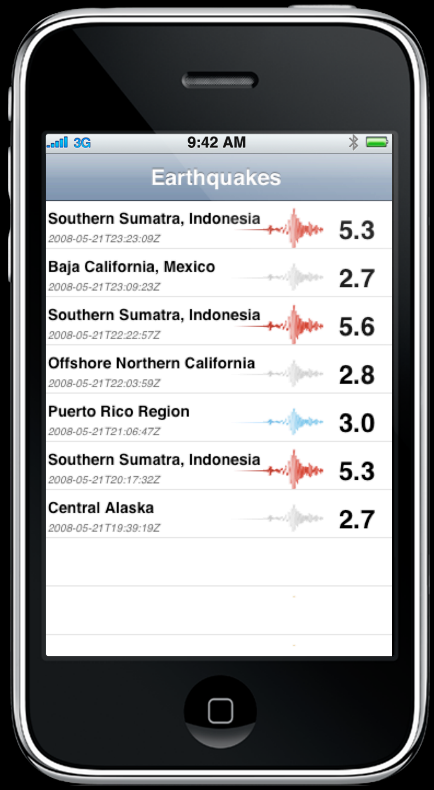
What you need

`edu.stanford.cs193.app`

Using the Service

What you do



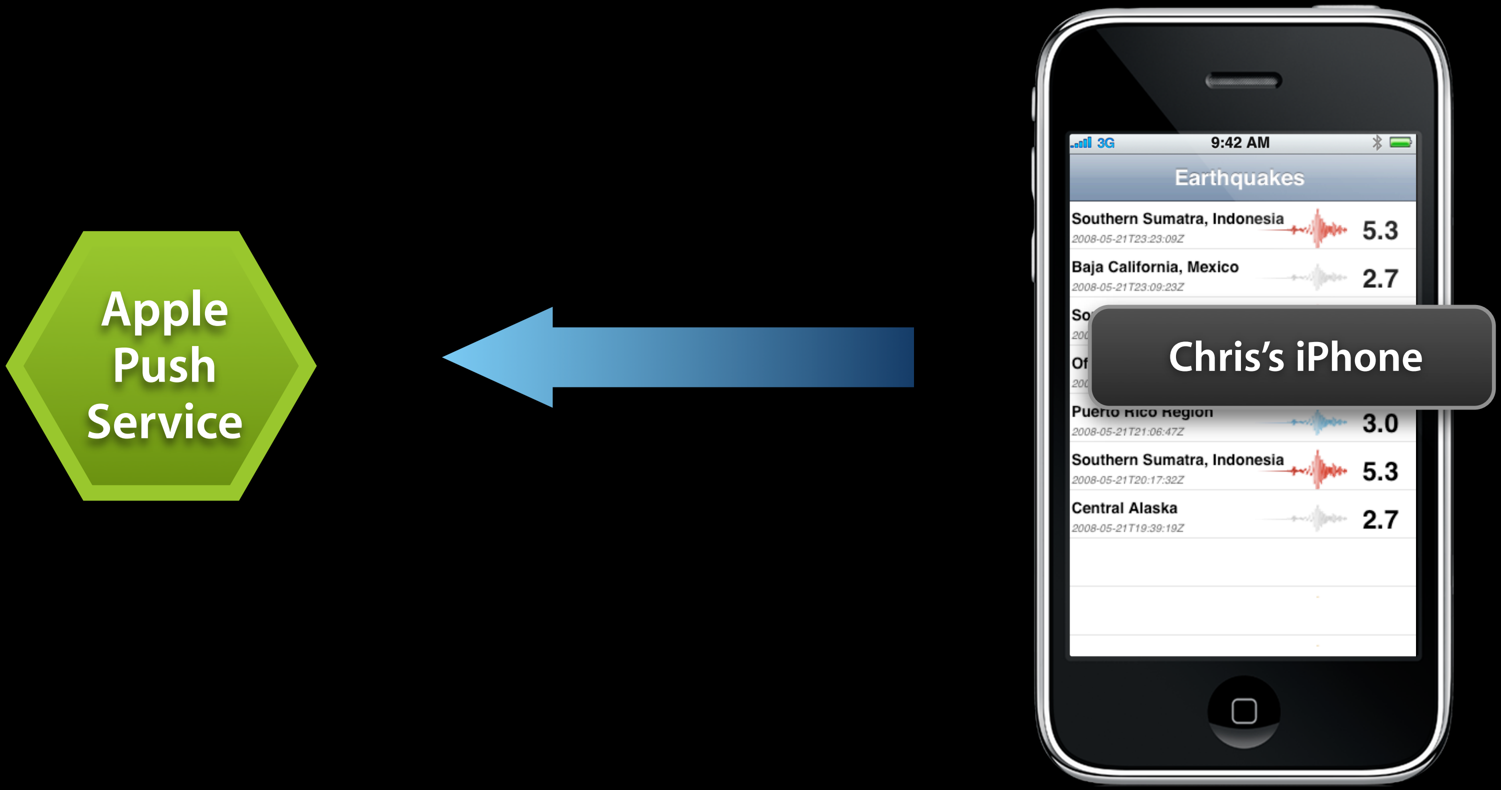


Earthquakes

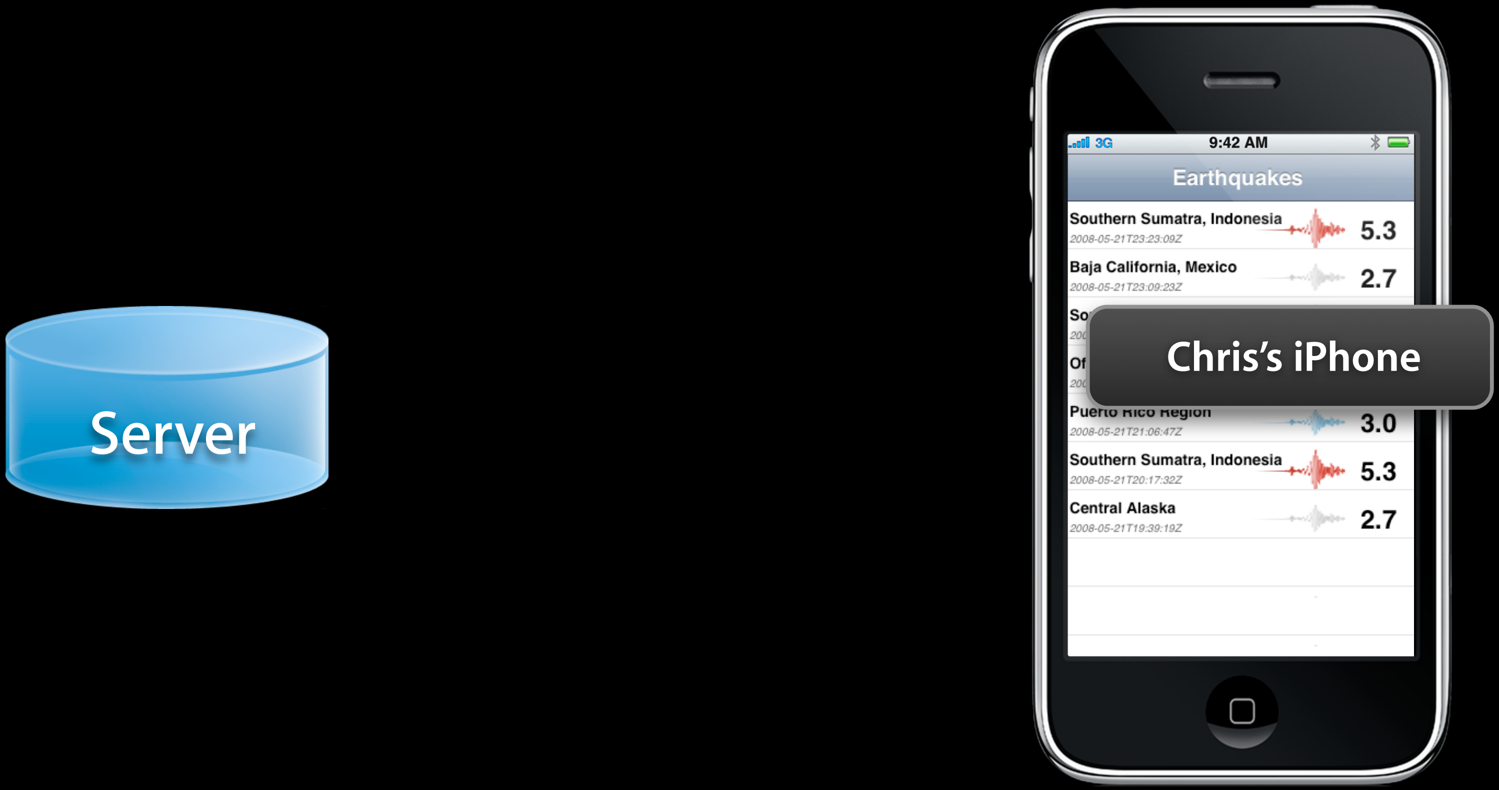
Southern Sumatra, Indonesia 2008-05-21T23:23:09Z	5.3
Baja California, Mexico 2008-05-21T23:09:23Z	2.7
Southern Sumatra, Indonesia 2008-05-21T22:22:57Z	5.6
Offshore Northern California 2008-05-21T22:03:59Z	2.8
Puerto Rico Region 2008-05-21T21:06:47Z	3.0
Southern Sumatra, Indonesia 2008-05-21T20:17:32Z	5.3
Central Alaska 2008-05-21T19:39:19Z	2.7

Using the Service

1. Register with the service

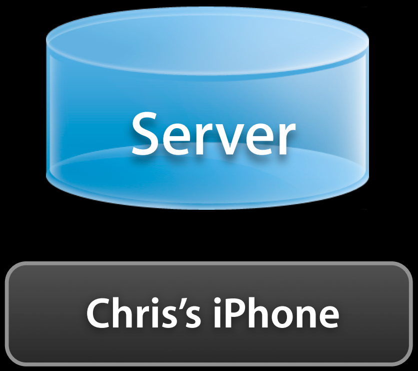


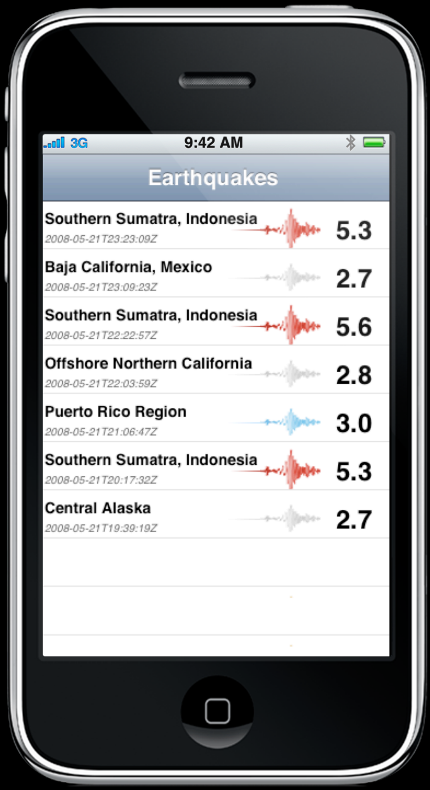
Using the Service


Server
Chris's iPhone
9:42 AM
Earthquakes
Southern Sumatra, Indonesia 5.3
Baja California, Mexico 2.7
Puerto Rico Region 3.0
Southern Sumatra, Indonesia 5.3
Central Alaska 2.7

Using the Service

2. Send token to your server

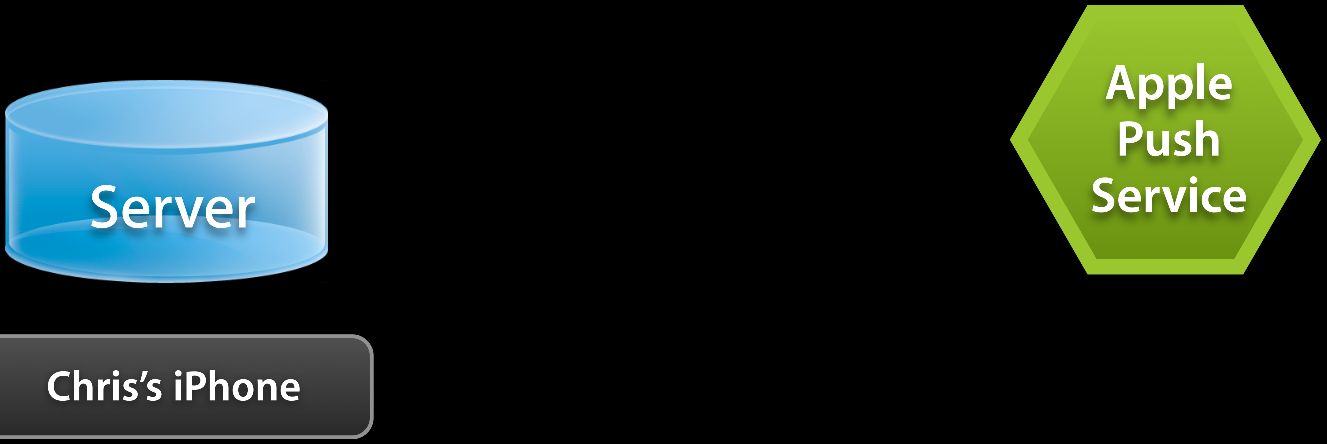




Southern Sumatra, Indonesia 2008-05-21T23:23:09Z	5.3
Baja California, Mexico 2008-05-21T23:09:23Z	2.7
Southern Sumatra, Indonesia 2008-05-21T22:22:57Z	5.6
Offshore Northern California 2008-05-21T22:03:59Z	2.8
Puerto Rico Region 2008-05-21T21:06:47Z	3.0
Southern Sumatra, Indonesia 2008-05-21T20:17:32Z	5.3
Central Alaska 2008-05-21T19:39:19Z	2.7

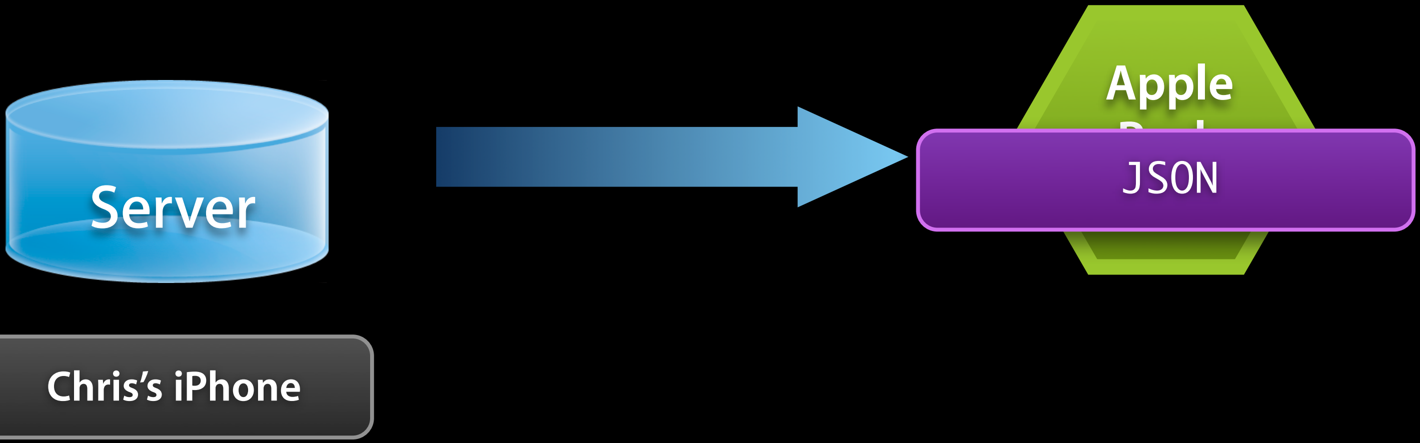
Using the Service

3. Send notifications



Using the Service

3. Send notifications



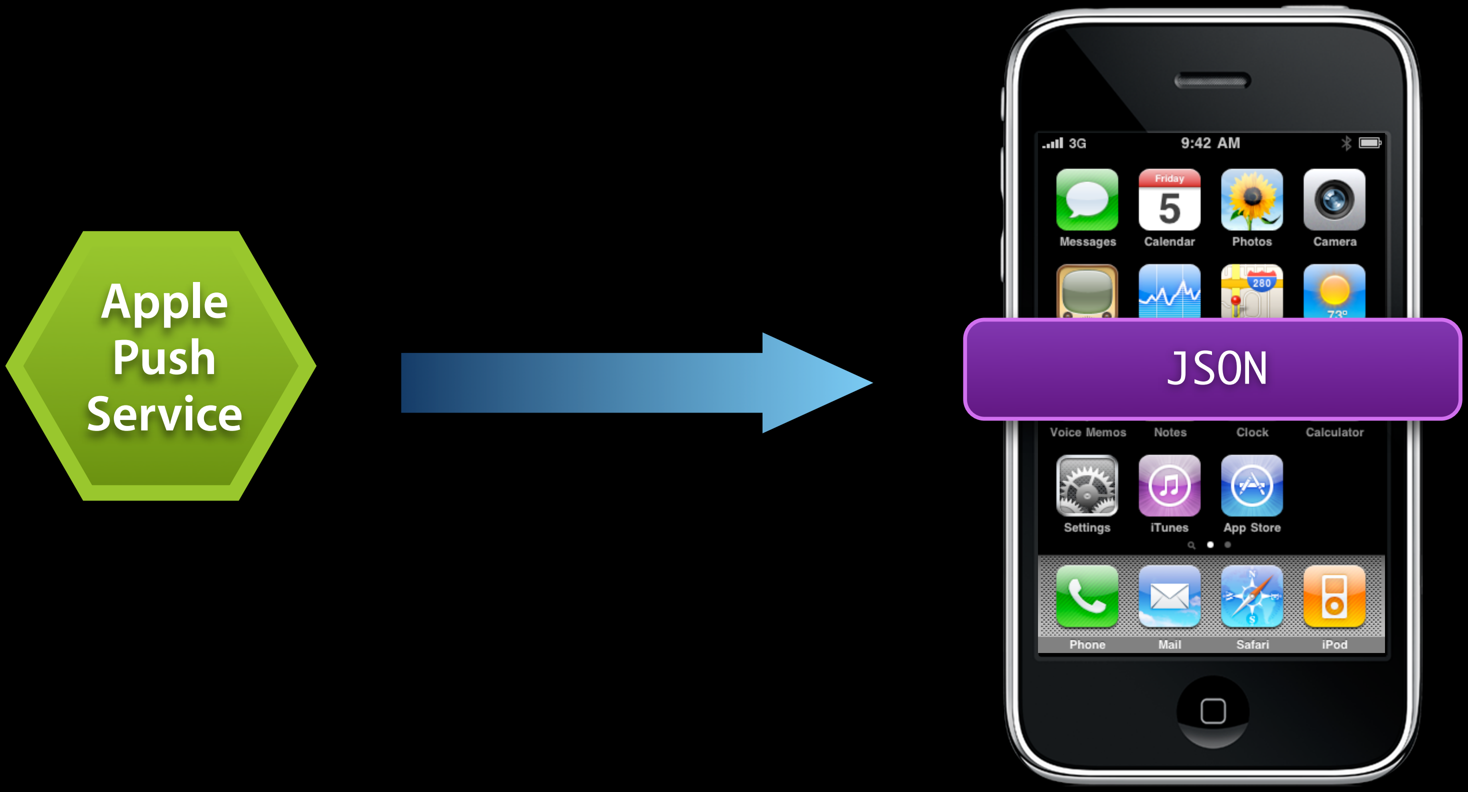
Using the Service

4. Receive notifications

Apple
Push
Service

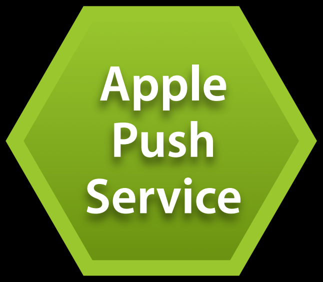
Using the Service

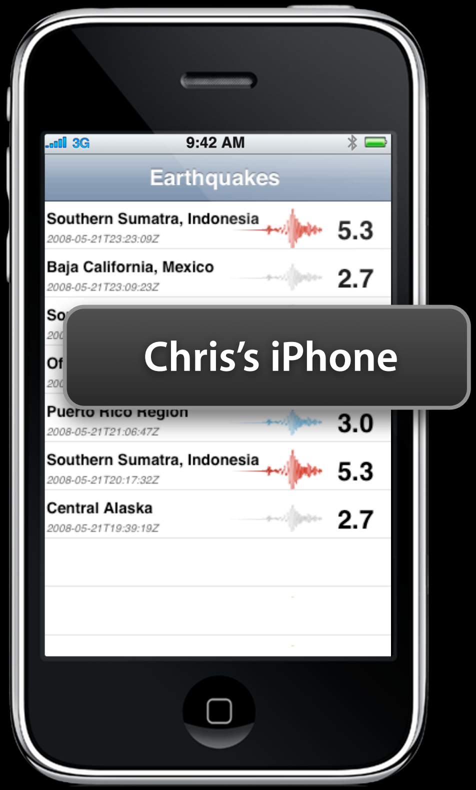
4. Receive notifications



Using the Service

1. Register with the service

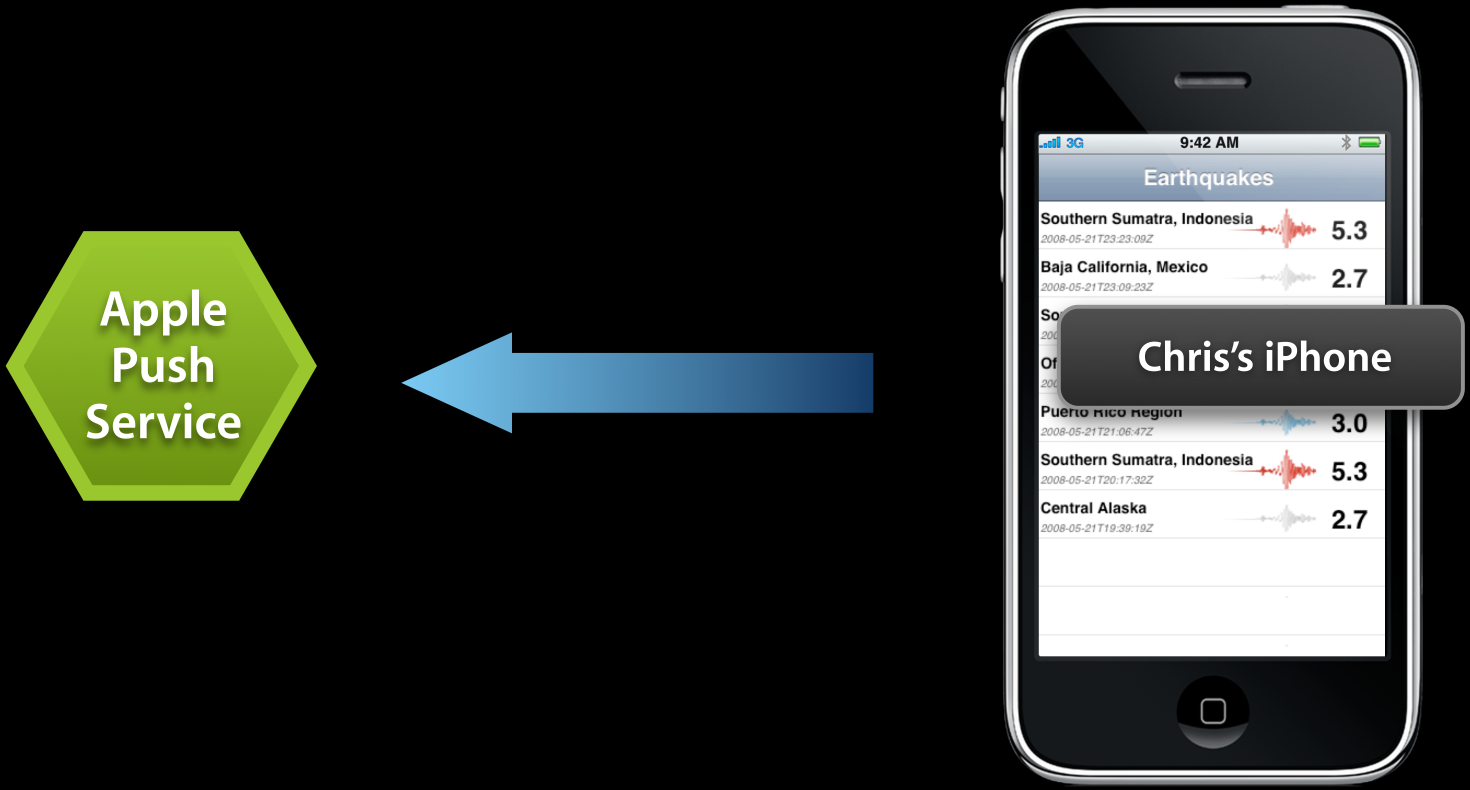




Southern Sumatra, Indonesia	2008-05-21T23:23:09Z	5.3
Baja California, Mexico	2008-05-21T23:09:23Z	2.7
Puerto Rico Region	2008-05-21T21:06:47Z	3.0
Southern Sumatra, Indonesia	2008-05-21T20:17:32Z	5.3
Central Alaska	2008-05-21T19:39:19Z	2.7

Using the Service

1. Register with the service



Registering with the Service

Application launch

- UIKit API in UIApplication.h to register
    - Pass the types you want to receive

```
 -(void)application:(UIApplication *)application
          didFinishLaunchingWithOptions:(NSDictionary *)options
{
     // Register this app on this device
     UIRemoteNotificationType myTypes = UIRemoteNotificationTypeSounds |
                                        UIRemoteNotificationTypeBadges;
     [application registerForRemoteNotificationTypes:myTypes];
}
```

Registering with the Service

Delegate callbacks

Registering with the Service

Delegate callbacks

```
 - (void)application:(UIApplication *)application
    didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)token
{
    // Phone home with device token
}
```

Registering with the Service

Delegate callbacks

```
 - (void)application:(UIApplication *)application
    didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)token
{
    // Phone home with device token
}
```

```
 - (void)application:(UIApplication *)application
    didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
{
    // Oh noes! Check your Provisioning Profile on device and in Xcode
}
```

Registering with the Service

```
96385da767191121a851963983fdac9bbdf74dcf6219ae14ed8d08228
```

Registering with the Service

The device token

```
96385da767191121a851963983fdac9bbdf74dcf6219ae14ed8d08228
```

Registering with the Service

The device token

- Uniquely identifies device

```
96385da767191121a851963983fdac9bbdf74dcf6219ae14ed8d08228
```

Registering with the Service

The device token

- Uniquely identifies device
    - Distinct from -[UIDevice deviceIdentifier]

```
96385da767191121a851963983fdac9bbdf74dcf6219ae14ed8d08228
```

Registering with the Service

The device token

- Uniquely identifies device
    - Distinct from -[UIDevice deviceIdentifier]
- Just call registration API again if token is needed

```
96385da767191121a851963983fdac9bbdf74dcf6219ae14ed8d08228
```

Registering for Notifications

Optional callbacks and methods

- UIApplicationDelegate

```
- (void)application:(UIApplication *)application
     didReceiveRemoteNotification:(NSDictionary *)userInfo
```

Registering for Notifications

Optional callbacks and methods

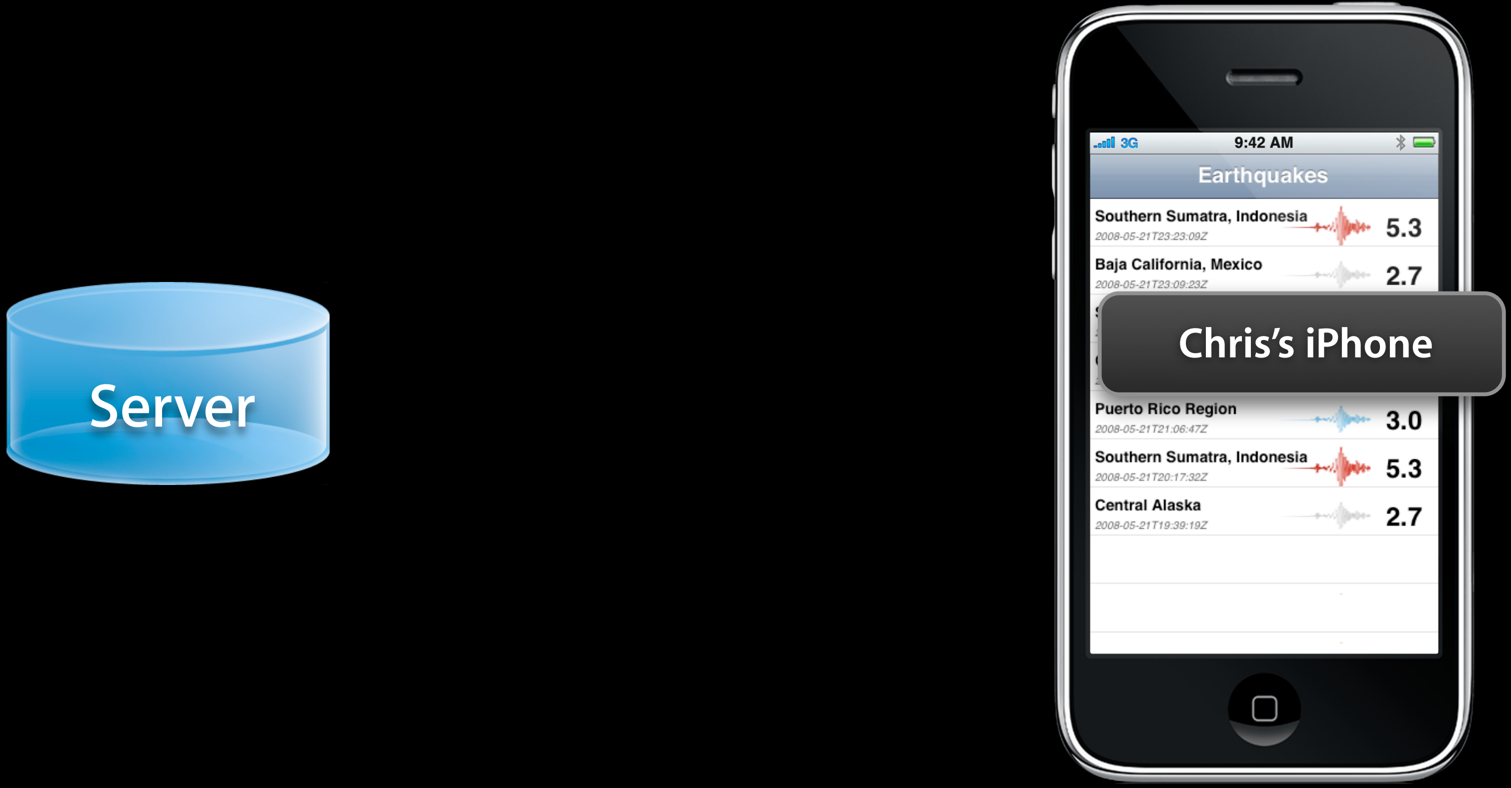
- UIApplicationDelegate

```
- (void)application:(UIApplication *)application
     didReceiveRemoteNotification:(NSDictionary *)userInfo
```

- UIApplication

```
- (UIRemoteNotificationType)enabledRemoteNotificationTypes
```

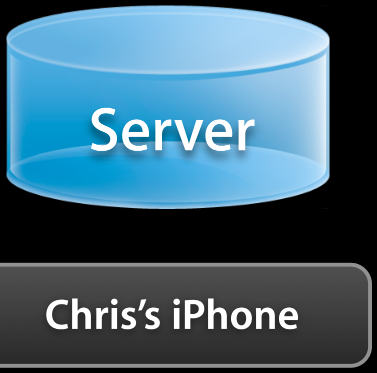
Using the Service

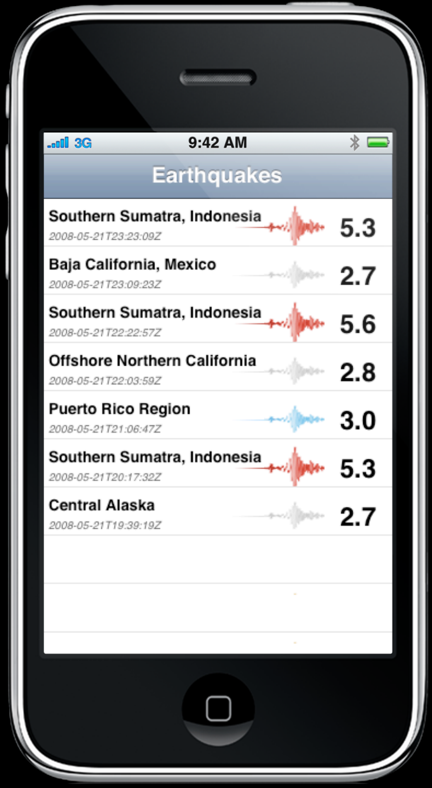


Southern Sumatra, Indonesia 2008-05-21T23:23:09Z 5.3
Baja California, Mexico 2008-05-21T23:09:23Z 2.7
Puerto Rico Region 2008-05-21T21:06:47Z 3.0
Southern Sumatra, Indonesia 2008-05-21T20:17:32Z 5.3
Central Alaska 2008-05-21T19:39:19Z 2.7

Using the Service

2. Send token to your server

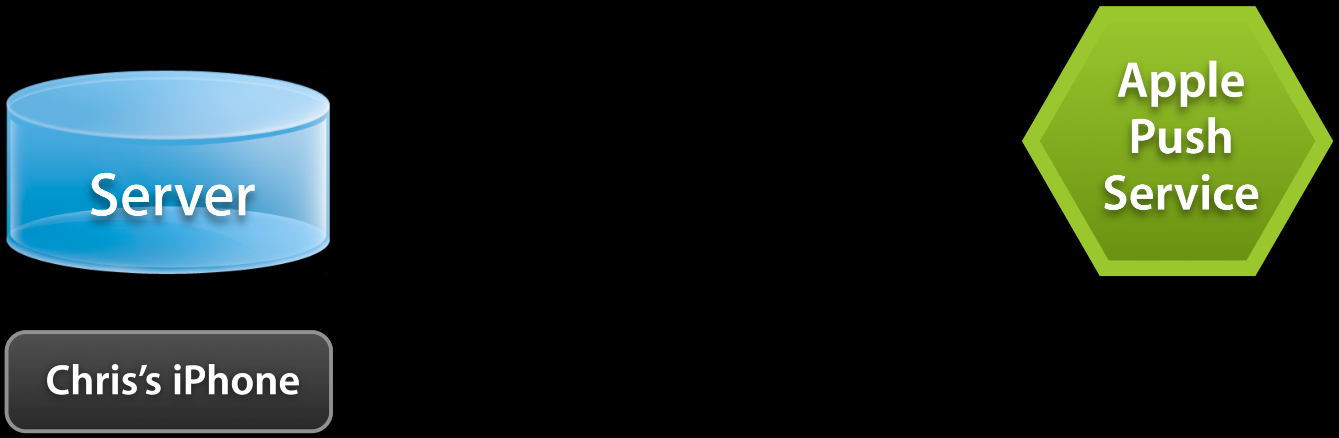




Southern Sumatra, Indonesia	2008-05-21T23:23:09Z	5.3
Baja California, Mexico	2008-05-21T23:09:23Z	2.7
Southern Sumatra, Indonesia	2008-05-21T22:22:57Z	5.6
Offshore Northern California	2008-05-21T22:03:59Z	2.8
Puerto Rico Region	2008-05-21T21:06:47Z	3.0
Southern Sumatra, Indonesia	2008-05-21T20:17:32Z	5.3
Central Alaska	2008-05-21T19:39:19Z	2.7

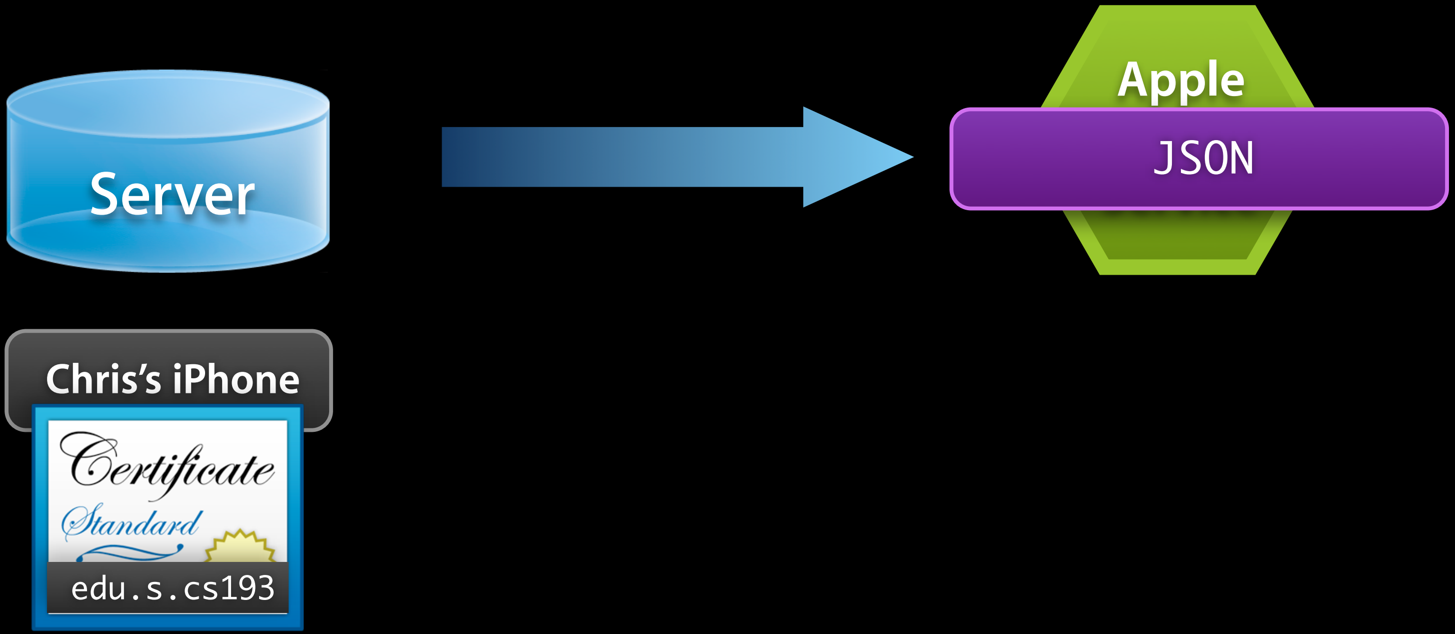
Using the Service

3. Send notifications



Using the Service

3. Send notifications



Sending Notifications

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle.aiff"
    },
    "acme1" : "conversation9964"
}
```

Sending Notifications

Message payload

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle.aiff"
    },
    "acme1" : "conversation9964"
}
```

Sending Notifications
Message payload

- Strict RFC 4627 JSON

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle.aiff"
    },
    "acme1" : "conversation9964"
}
```

Sending Notifications

Message payload

- Strict RFC 4627 JSON
- 256 byte maximum

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle.aiff"
    },
    "acme1" : "conversation9964"
}
```

Sending Notifications

Message payload

- *aps* dictionary reserved for the sound, badge or alert keys
    - All keys optional

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle.aiff"
    },
    "acme1" : "conversation9964"
}
```

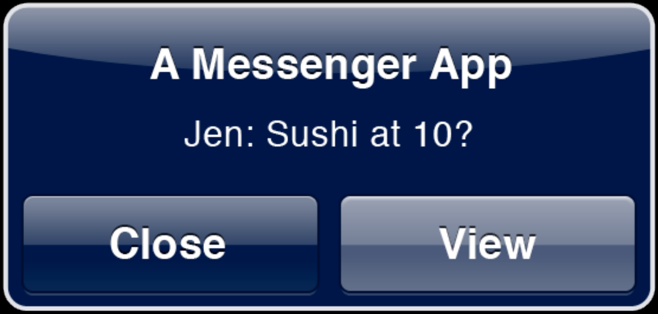
- Rest of payload is for your app

Sending Notifications

Message payload

- *aps* dictionary reserved for the sound, badge or alert keys
  - All keys optional

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle.aiff"
    },
    "acme1" : "conversation9964"
}
```



- Rest of payload is for your app

Badges

badge key, integer value

- Positive integer
  - Or omit to remove

```
{
    "aps" : {
        "badge" : 1
    }
}
```

Badges

badge key, integer value

- Positive integer
    - Or omit to remove

```
{
    "aps" : {
        "badge" : 1
    }
}
```

Sounds

sound key, string value

- Either a filename in app bundle
  - linear PCM
  - MA4
  - µLaw
  - aLaw
- Or “default”
- Vibration is automatic

```
{
    "aps" : {
        "sound" : "Jingle.aiff"
    }
}
```

Alerts

alert key, string or dictionary value

- Simplest form is just a string value

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?"
    }
}
```

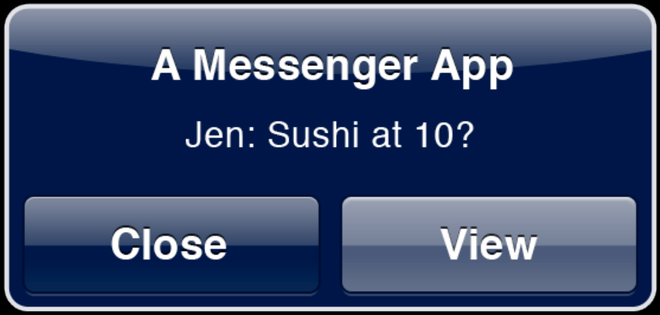
- Can be localized (see documentation)
- Can also customize the text on the view button
    - or omit it

Alerts

alert key, string or dictionary value

- Simplest form is just a string value

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?"
    }
}
```



- Can be localized (see documentation)
- Can also customize the text on the view button
  - or omit it

Sending the Payload

Send JSON that is stripped of whitespace

```
{
    "aps" : {
        "alert" : "Jen: Sushi at 10?",
        "badge" : 1,
        "sound" : "Jingle1.aiff"
    },
    "acme1" : "conversation9964"
```

150 bytes

Sending the Payload

Send JSON that is stripped of whitespace

```
{"aps":{"alert":"Jen: Sushi at 10?","badge":
1,
"sound":"Jingle.aiff"},"acme1":"conversation
9964"}
```

96 bytes

Demo:
Pushing to the Flickr app

NSUserDefaults recap

(time permitting)

NSUserDefaults

- Convenient way to store settings and lightweight state
  - Arrays, dictionaries, strings, numbers, dates, raw data
  - Settings bundles can be created so that user defaults can be set from Settings app
  - Internally stored as property lists

Reading & Writing User Defaults

- Key-value store
- Base methods accept and return objects for values

```
+ (NSUserDefaults *)standardUserDefaults;

- (id)objectForKey:(NSString *)defaultName;
- (void)setObject:(id)value forKey:(NSString *)defaultName;
- (void)removeObjectForKey:(NSString *)defaultName;

- (BOOL)synchronize;
```

Reading & Writing User Defaults

- Many convenience methods that ‘box’ and ‘unbox’ the object
    - and perform type checking

```
- (NSString *)stringForKey:(NSString *)defaultName;
- (NSArray *)arrayForKey:(NSString *)defaultName;
- (NSDictionary *)dictionaryForKey:(NSString *)defaultName;
- (NSData *)dataForKey:(NSString *)defaultName;
- (NSArray *)stringArrayForKey:(NSString *)defaultName;
- (NSInteger)integerForKey:(NSString *)defaultName;
- (float)floatForKey:(NSString *)defaultName;
- (double)doubleForKey:(NSString *)defaultName;
- (BOOL)boolForKey:(NSString *)defaultName;

- (void)setInteger:(NSInteger)value forKey:(NSString *)
defaultName;
- (void)setFloat:(float)value forKey:(NSString *)defaultName;
- (void)setDouble:(double)value forKey:(NSString *)defaultName;
- (void)setBool:(BOOL)value forKey:(NSString *)defaultName;
```

-[NSUserDefaults synchronize]

- Call [[NSUserDefaults standardUserDefaults] synchronize] to write changes to disk
- Also loads external changes from disk (useful on Mac OS X)

More on NSUserDefaults

- “User Defaults Programming Topics for Cocoa”
  http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/UserDefaults/Tasks/UsingDefaults.html

Demo:
NSUserDefaults and Settings

Recap

- Property lists, NSUserDefaults
  - Quick & easy, but limited
- Archived objects
  - More flexible, but require writing a lot of code
- SQLite
  - Elegant solution for many types of problems
- XML and JSON
  - Low-overhead options for talking to “the cloud”
  - Apple Push Notification Service pushes JSON from your server to devices

Questions?