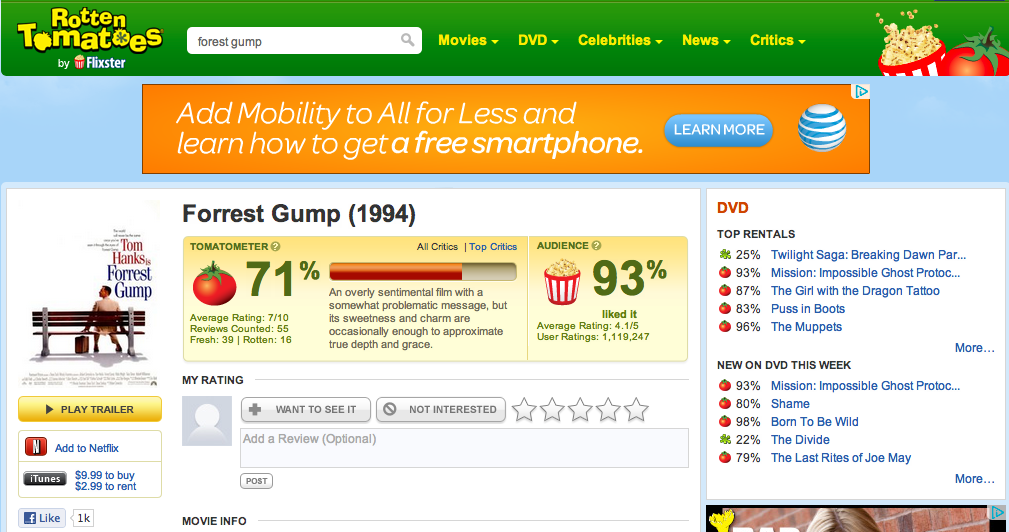
4.0: Evaluate and Parse a JSON Feed

JSON, pronounced [Jay Sawn], stands for JavaScript Object Notation. Just as news outlets and subscription services use XML RSS feeds, JSON is the preferred web-publishing notation used by developers. Many web sites and services create an Application Programming Interface (API) to customize, create a mobile app, or some other interface.

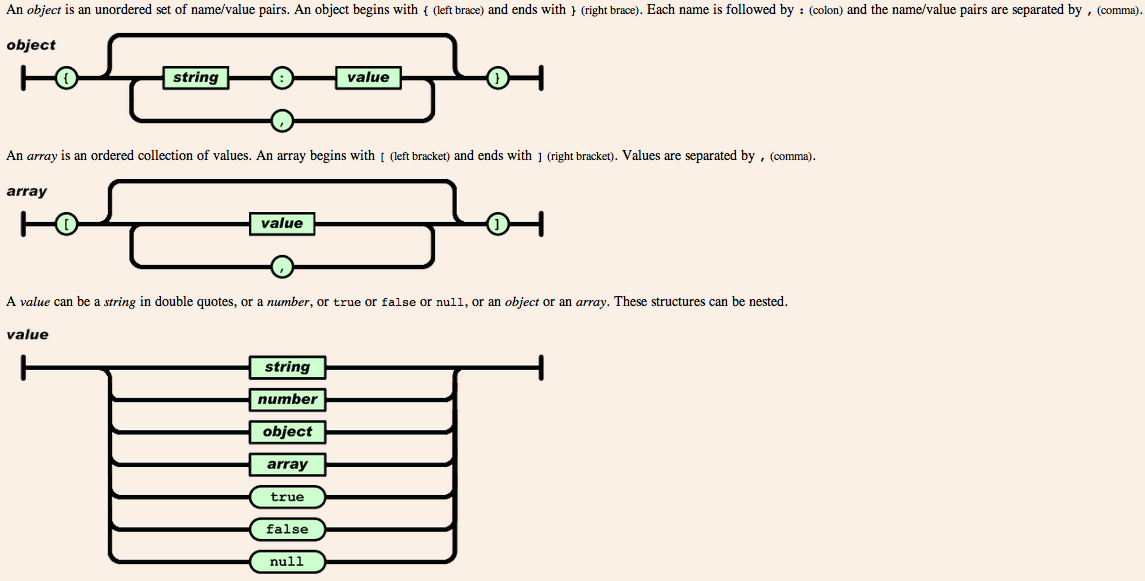
For example, [Rottentomatoes.com](http://rottentomatoes.com) is a web site that provides ratings, reviews, and other movie data. It is a great web site, but I may want to create a mobile application that quickly pulls the data based on the same search. The development team has created an API that allows us to interface with their data. Here is an example of using their web interface to search on the movie *Forrest Gump*.



Here is an example of the JSON created from that same search.



As illustrated on the json.org website, here are the different forms and structure of a JSON file.



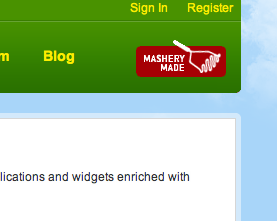
Here is a quick compare of the same data shown in XML and JSON from the json.org site.



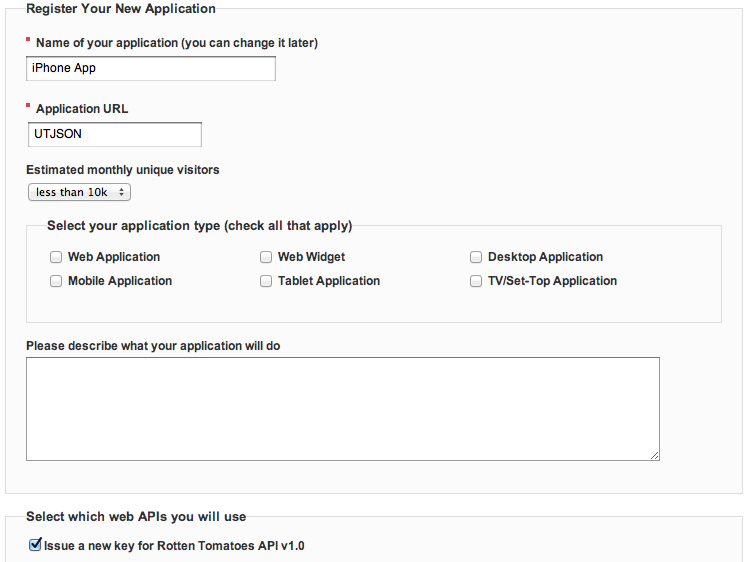
|  |  |
| --- | --- |
| Let’s get back to using the Rotten Tomatoes API and create a quick application that pulls data from the RottenTomatoes.com web site and display it on our iPhone/iPod/iPad. When we are done the application should look like the following. |  |

**Prerequisite:** Before we get started we need to get an API Key. This is usually a collection of characters that identify the user that is using the API.

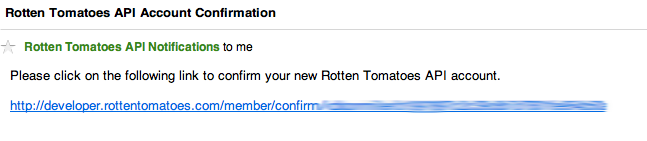
**Step P1:** Navigate to the web site <http://developer.rottentomatoes.com/> and click the register link in the top right hand corner.



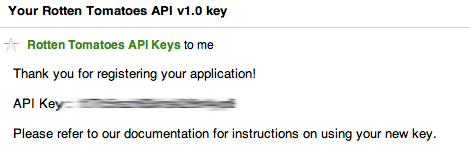
**Step P2:** Enter the required information. Here is a sample of information for the application.



**Step P3:** Validate email address with the email they send.



**Step P4:** Record your API key sent to you after you validated your email.

****

**Step 1:** Start a new Single View Application.

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

**Step 2:** Now in iOS 5, the JSON parser is built into the framework. We will be using the [NSJSONSerialization](http://developer.apple.com/library/ios/#documentation/Foundation/Reference/NSJSONSerialization_Class/Reference/Reference.html) class. You can review a pretty good example on Sonny Parlin’s blog at <http://sonnyparlin.com/2012/04/parsing-json-with-ios-5/>.

# NSJSONSerialization Class Reference

|  |  |
| --- | --- |
| **Inherits from** | [NSObject](http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSObject_Class/Reference/Reference.html#//apple_ref/occ/cl/NSObject) |
| **Conforms to** | [NSObject (NSObject)](http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Protocols/NSObject_Protocol/Reference/NSObject.html#//apple_ref/occ/intf/NSObject) |
| **Framework** | /System/Library/Frameworks/[Foundation.framework](http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/ObjC_classic/_index.html" \l "//apple_ref/doc/uid/20001091) |
| **Availability** | Available in iOS 5.0 and later. |
| **Declared in** | NSJSONSerialization.h |
| **Related sample code** | [Tweeting](http://developer.apple.com/library/ios/samplecode/Tweeting/Introduction/Intro.html#//apple_ref/doc/uid/DTS40011191) |

## Overview

You use the NSJSONSerialization class to convert JSON to Foundation objects and convert Foundation objects to JSON.

An object that may be converted to JSON must have the following properties:

* The top level object is an NSArray or NSDictionary.
* All objects are instances of NSString, NSNumber, NSArray, NSDictionary, or NSNull.
* All dictionary keys are instances of NSString.
* Numbers are not NaN or infinity.

Other rules may apply. Calling [isValidJSONObject:](http://developer.apple.com/library/ios/documentation/Foundation/Reference/NSJSONSerialization_Class/Reference/Reference.html#//apple_ref/occ/clm/NSJSONSerialization/isValidJSONObject:) or attempting a conversion are the definitive ways to tell if a given object can be converted to JSON data.

**Step 3:** Now let’s apply our design to the JSONRottenTomatoesViewController.xib file. Use the image below to include your textview, textfield, labels, and button.

**Step 4:** Now let’s create the IBOUTLETS and IBACTION in the header file

#import <UIKit/UIKit.h>

@interface JSONRottenTomatoesViewController : UIViewController

{

}

@property (nonatomic, strong) IBOutlet UITextField \*txtSearch;

@property (nonatomic, strong) IBOutlet UITextField \*txtTitle;

@property (nonatomic, strong) IBOutlet UITextField \*txtYear;

@property (nonatomic, strong) IBOutlet UITextField \*txtRating;

@property (nonatomic, strong) IBOutlet UITextField \*txtRuntime;

@property (nonatomic, strong) IBOutlet UITextView \*txtOutput;

@property (nonatomic, strong) IBOutlet UITextField \*txtTheater;

@property (nonatomic, strong) IBOutlet UITextField \*txtDVD;

-(IBAction)parseJSON:(id)sender;

@end

**Step 5:** Now let’s connect the outlets and action to the xib file.



**Step 6:** Synthesize the variables in the implementation file.

@synthesize txtTitle, txtYear, txtRating, txtRuntime;

@synthesize txtDVD, txtTheater, txtOutput, txtSearch;

**Step 7:** Let’s build the parseJSON method. This is the heart of the application.

-(IBAction)parseJSON:(id)sender

{

// Format search bar

NSString\* formattedSearch = [txtSearch.text stringByReplacingOccurrencesOfString:@" " withString:@"+"];

// Variable to store our API Key

NSString\* const RT\_API\_KEY = @"t7c5rszh63xhsk2r8n4yqrd8"; // This is where you put your API key

NSString\* searchURL = [NSString

stringWithFormat:@"http://api.rottentomatoes.com/api/public/v1.0/movies.json?apikey=%@&q=%@",

RT\_API\_KEY, formattedSearch];

NSLog(@"%@", searchURL);

NSError \*error = nil;

NSData \*jsonData = [NSData dataWithContentsOfURL:[NSURL URLWithString:searchURL]];

if (jsonData)

{

id jsonObjects = [NSJSONSerialization JSONObjectWithData:jsonData

options:NSJSONReadingMutableContainers

error:&error];

if (error)

{

NSLog(@"error is %@", [error localizedDescription]);

// Handle Error and return

return;

}

NSArray \*movieArray = [jsonObjects objectForKey:@"movies"];

// loop through movies

for (NSDictionary \*movie in movieArray)

{

NSLog(@"%@ is %@", movie, [jsonObjects objectForKey:movie]);

[txtOutput setText:[[NSString stringWithFormat:@"%@", movie]

stringByTrimmingCharactersInSet:

[NSCharacterSet whitespaceCharacterSet]]];

[txtTitle setText:[movie objectForKey:@"title"]];

[txtYear setText:[NSString stringWithFormat:@"%d",

[[movie objectForKey:@"year"] integerValue]]];

[txtRating setText:[movie objectForKey:@"mpaa\_rating"]];

[txtRuntime setText:[NSString stringWithFormat:@"%d minutes",

[[movie objectForKey:@"runtime"] integerValue]]];

// Get dvd and theater release

NSDictionary \*release\_dates = [movie objectForKey:@"release\_dates"];

NSString \*theaterRelease = [release\_dates objectForKey:@"theater"];

[txtTheater setText:theaterRelease];

NSString \*dvdRelease = [release\_dates objectForKey:@"dvd"];

[txtDVD setText:dvdRelease];

// Show the critics consensus; overwrites json output

// [txtOutput setText:[movie objectForKey:@"critics\_consensus"]];

}

[txtSearch resignFirstResponder];

}

else

{

// Handle Error

}

}

**Step 8:** Let’s just implement a method to hide the keyboard if we touch anywhere else on the screen.

-(void)touchesBegan:(NSSet \*)touches withEvent:(UIEvent \*)event

{

UITouch \*touch = [touches anyObject];

if (touch != nil)

{

[txtSearch resignFirstResponder];

}

}

Congrats!!!! We are done. It should look something like this:

