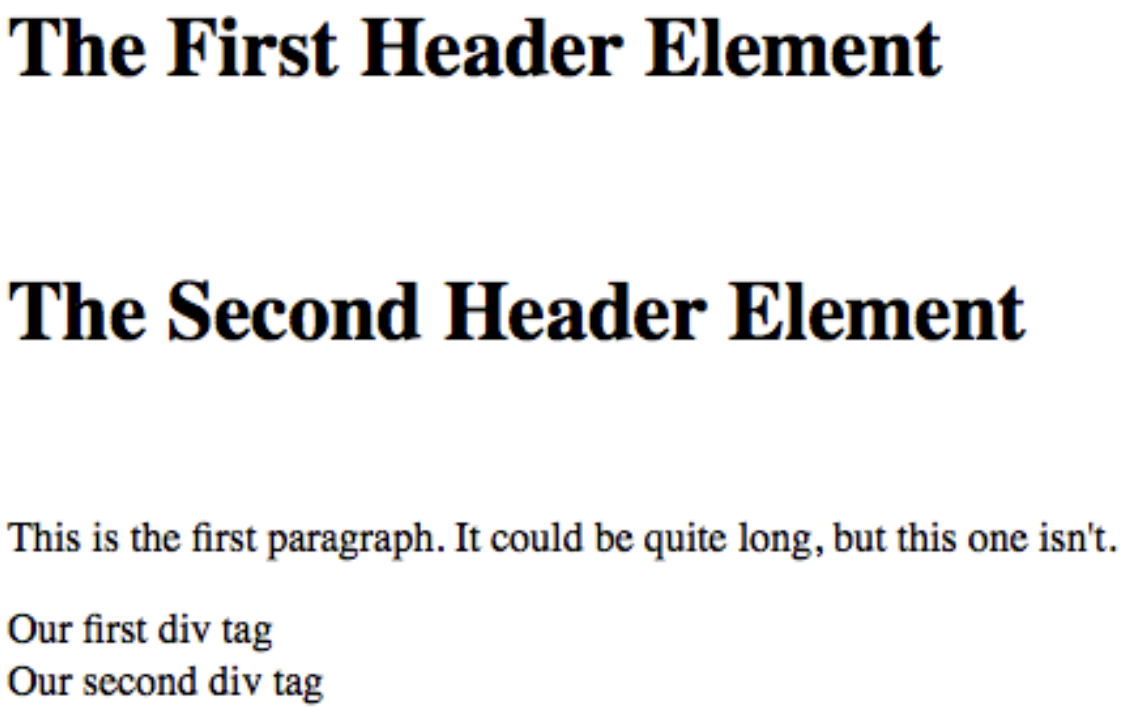
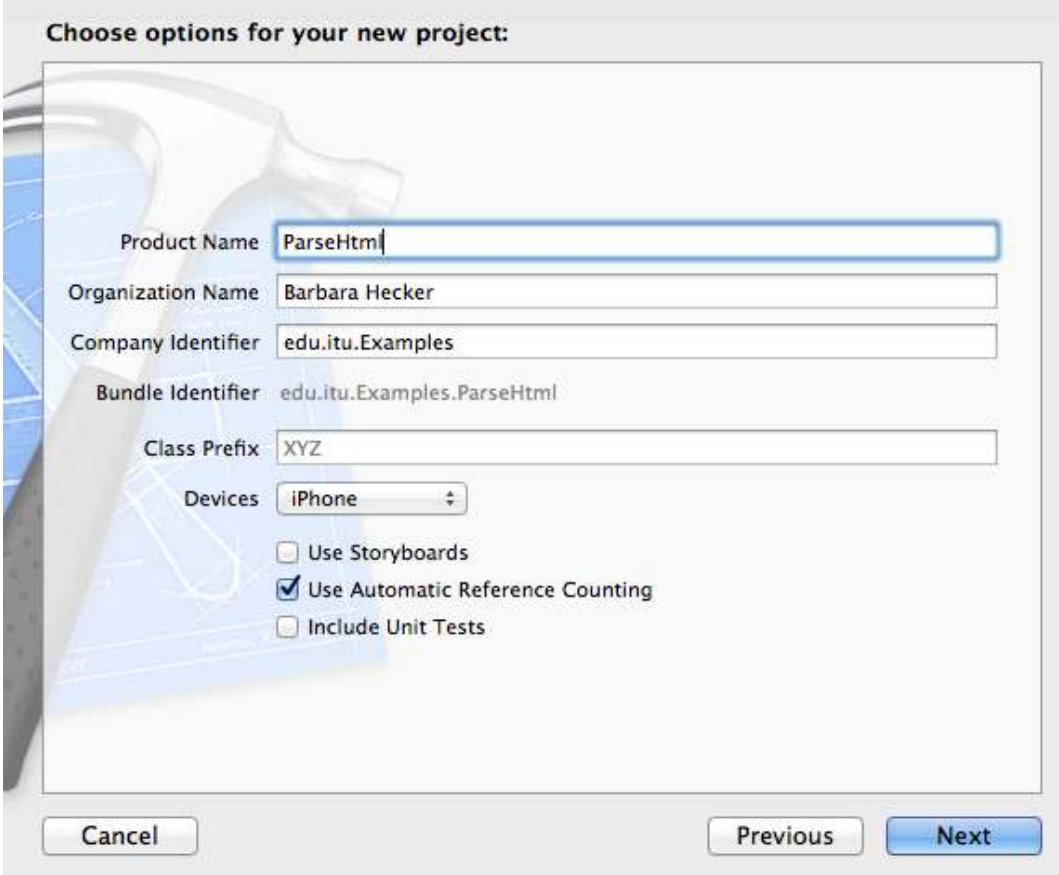
iPhone Tutorial – How to Parse HTML

In this tutorial we are going to parse a webpage and put the results into a list. The html page we are parsing is a very simple and you can view it at http://www.bhecker.com/testpage.html.



1. Start Xcode, Select "Create a new Xcode project," and choose the **Single View Application** template. Click next, name the project "**ParseHtml**," and select options as shown:



Click Next, choose a location to save the project and click Create.

2. We will first create an object to hold the html data. We will name this class **HtmlElement** and it will just be a simple Objective-C class that is a subclass of **NSObject**.

Select File->New->File and then Objective-C class.

3. Our new object will hold two NSStrings one for the name of the html tag and one for the value of the tag.

Open HtmlElement.h and add tow properties as follows:

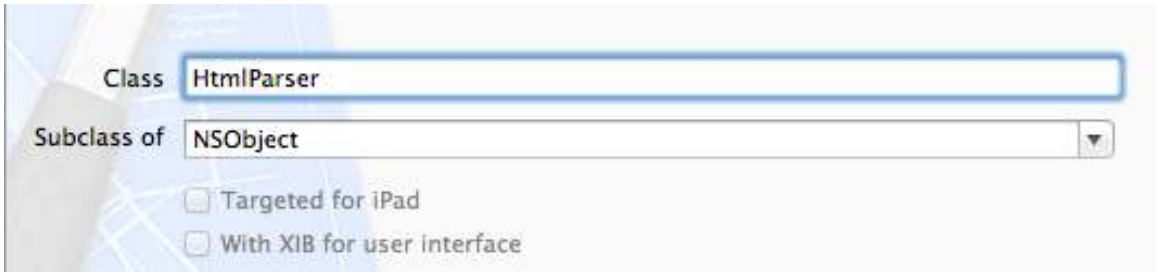
```
@property (nonatomic, retain) NSString *tag;
@property (nonatomic, retain) NSString *value;
```

4. The only thing we do in the implementation file is synthesize our variables. Open HtmlElement.m and add the following line of code.

```
@synthesize tag, value;
```

5. Next we need to create a parser class. Name it **HtmlParser** and it should also be a simple class that is a subclass of NSObject.

Select File->New->File and then Objective-C class. Name the class **HtmlParser**.



The parser object needs to import the **HtmlElement.h** class, hold four instance variables and declares one action. It also needs to implement **NSXMLParserDelegate** protocol.

Open **HtmlParser.h** and make the following changes:

Add the line to import the element:

```
#import "HtmlElement.h"
```

Add <NSXMLParserDelegate> to the interface line to use the protocol in the class implementation.

```
@interface HtmlParser : NSObject <NSXMLParserDelegate>
```

Add the following 4 properties and 1 method prototype:

```
@property NSMutableString *currentNodeContent;
@property NSMutableArray *elementArray;
@property NSXMLParser *parser;
@property HtmlElement *currentHTMLElement;

-(id) loadHtmlByURL:(NSString *)urlString;
```

5. Open HtmlParser.m and synthesize the properties we have created as follows:

```
@synthesize currentHTMLElement, elementArray, parser,
currentNodeContent;
```

6. We also need to add the implementation for the **loadHtmlByURL** method.

```
-(id) loadHtmlByURL:(NSString *)urlString
{
    NSURL *url = [NSURL URLWithString:urlString];
    NSData *nsData = [[NSData alloc]
initWithContentsOfURL:url];
    elementArray = [[NSMutableArray alloc] init];
    parser = [[NSXMLParser alloc] initWithData:nsData];
    parser.delegate = self;
    [parser parse];
    currentHTMLElement = [HtmlElement alloc];
    return self;

}
```

7. Next we need to implement 3 methods to follow the NSXMLParserDelegate protocol.

```
- (void) parser:(NSXMLParser *)parser
didStartElement:(NSString *)elementname
namespaceURI:(NSString *)namespaceURI
qualifiedName:(NSString *)qName attributes:(NSDictionary
*)attributeDict
{
    currentHTMLElement = [HtmlElement alloc];
}

- (void) parser:(NSXMLParser *)parser
didEndElement:(NSString *)elementname
namespaceURI:(NSString *)namespaceURI
qualifiedName:(NSString *)qName
{
    if ([elementname isEqualToString:@"title"])
    {
        currentHTMLElement.tag = elementname;
        currentHTMLElement.value = currentNodeContent;
        [elementArray addObject:currentHTMLElement];
        currentHTMLElement = nil;
        currentNodeContent = nil;
    }
    if ([elementname isEqualToString:@"h1"])
    {
        currentHTMLElement.tag = elementname;
        currentHTMLElement.value = currentNodeContent;
        [elementArray addObject:currentHTMLElement];
        currentHTMLElement = nil;
        currentNodeContent = nil;
    }
    if ([elementname isEqualToString:@"p"])
    {
        currentHTMLElement.tag = elementname;
        currentHTMLElement.value = currentNodeContent;
        [elementArray addObject:currentHTMLElement];
        currentHTMLElement = nil;
        currentNodeContent = nil;
    }
```

```
    if ([elementname isEqualToString:@"div"])
    {
        currentHTMLElement.tag = elementname;
        currentHTMLElement.value = currentNodeContent;
        [elementArray addObject:currentHTMLElement];
        currentHTMLElement = nil;
        currentNodeContent = nil;
    }
}

- (void) parser:(NSXMLParser *)parser
foundCharacters:(NSString *)string
{
    currentNodeContent = (NSMutableString *) [string
stringByTrimmingCharactersInSet:[NSCharacterSet
whitespaceAndNewlineCharacterSet]];
}
```

8. The rest of the work is done in the **ViewController**. The header file imports the HtmlParser object and declares it as an instance variable.

Open ViewController.h and add the line to import HtmlParsser.h as follows:

```
#import "HtmlParser.h"
```

Make the ViewContoller a subclass of UITableViewController instead of UIViewController by making this change:

```
@interface ViewController : UITableViewController
```

Make a new property for an instance of an HtmlParser as follows:

```
@property HtmlParser *htmlParser;
```

9. Now, open **ViewController.m** and add the line to synthesize as follows:

```
@synthesize htmlParser;
```

10. In the **viewDidLoad** method alloc the **htmlParser** and pass the url to the html page into the l**oadHtmlByUrl** action as follows:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
        htmlParser = [[HtmlParser alloc]
loadHtmlByURL:@"http://www.bhecker.com/testpage.html"]
;
    self.title = @"HTML Elements";
}
```

11. Add a new method, TableView and set the numberOfRowsInSection method to use the elementArray by adding the following:

```
// Customize the number of rows in the table view.
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
    return [[htmlParser elementArray] count];
}
```

12. Everything else gets done in the **cellForRowAtIndexPath** method. We set the current HtmlElement from the elementArray and then use it to set the labels in the cells.

```
// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";
    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
```

```
        cell = [[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:CellIdentifier];
    }
    // Configure the cell.
    HtmlElement *currentElement = [[htmlParser
elementArray] objectAtIndex:indexPath.row];
    cell.textLabel.text = [currentElement tag];
    cell.detailTextLabel.text = [currentElement value];
    return cell;

}
```

13. The last step is to make sure you are using the correct nib type for the project type. Select the **ViewController.xib** file and then view the properties in the attribute inspector. Change the class (3rd icon from the left) from UIView to UITableView. Don't put anything in the interface, it will be populated programmatically. You can change the background color if you wish.

13. Build and run and see the results: