# TeraBrowser

*Version 1.0*

*Author: Alvaro J. Gene*

*Email: Gene001@teraexe.com*

*Website: www.teraexe.com*

# TERABROWSER

TeraBrowser is an iOS application that a user can use to store different kinds of web pages; then, after storing a group of websites, a user can use this application to load his/her webs while he/she is not connected to a computer network. In this PDF, you are going to find a source code about a file that is called the *AGene_Browser.swift* file. If you want to run the TeraBrowser in one of your devices or iPhones, you have to use the *AGene_Browser.swift* file and follow those instructions:

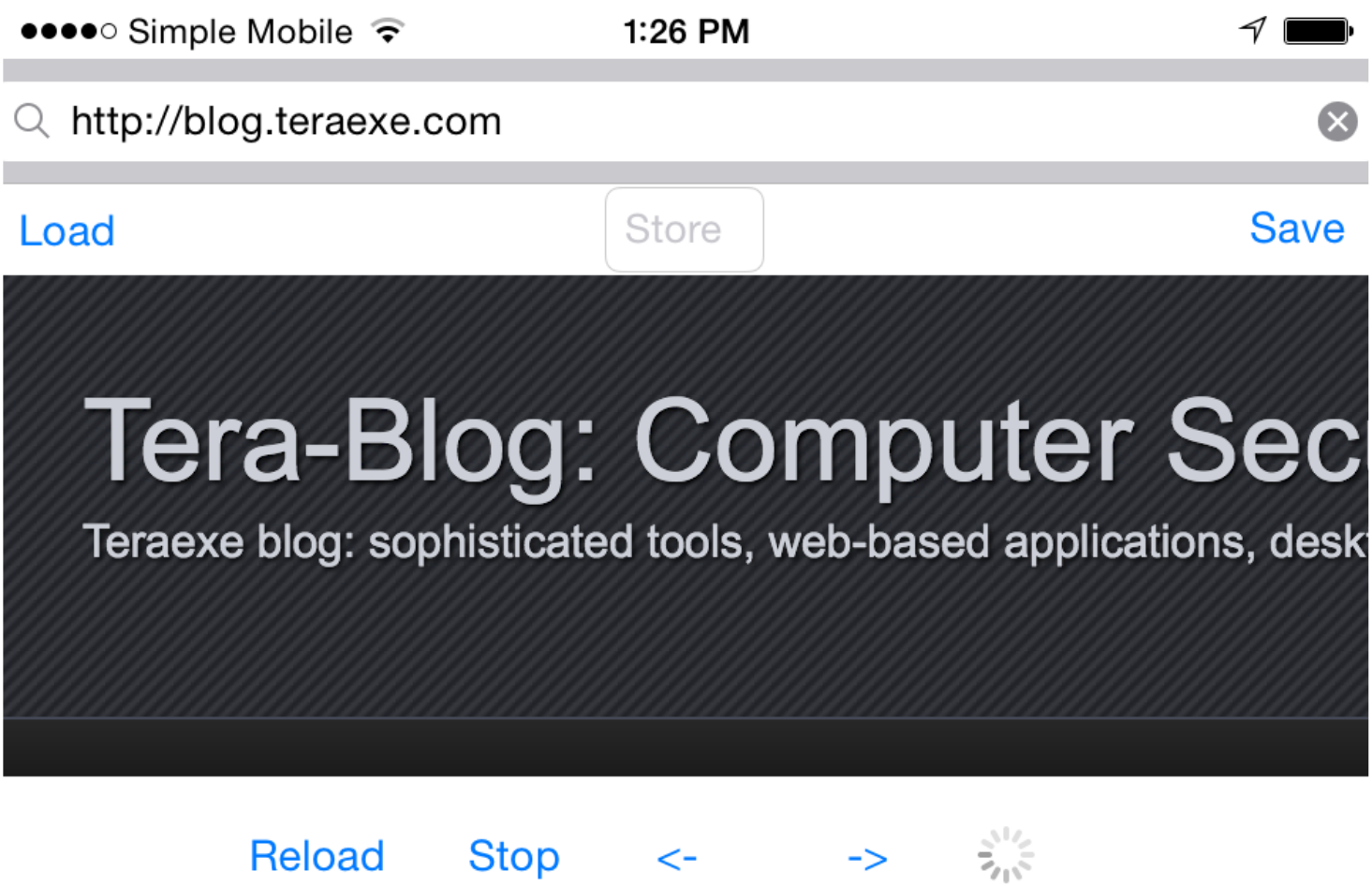
First, you have to use Xcode and configure it for two different kinds of frameworks, which are known as Core-Data and Webkit. Therefore, in addition to Xcode and Swift, you will need basic knowledge about Core-Data and Webkit to be able to run the TeraBrowser application.

Second, in Core-Data, you have to create the AGene_CoreData entity; then, you have to add three string attributes that will be called agene_web_a, agene_web_b, and agene_web_c.

Third, you have to use the Xcode canvas to add two buttons: the Save button that will use the btnSave_Data function and will save some websites, and the Load button that will use the btnLoad_Data function and will load a web page.

Finally, you have to use the Xcode canvas to add a group of objects (Main.storyboard). If you see the outlets in the next source code, you will know what kind of objects you have to add to the Xcode canvas.

## VISUAL ASPECT



## FILE 1: AGENE_BROWSER.SWIFT

```
//
//  AGene_Browser.swift
//  TeraBrowser
//
//  Created by Alvaro J. Gene on 8/7/16.
//  Copyright © 2016 Alvaro J. Gene. All rights reserved.
//

/*
 Author: Alvaro J. Gene (Socket_0x03)
 Name: TeraBrowser
 Date: August 7, 2016.
 Language: Swift
 Description: TeraBrowser is an iOS application that a user can use to store different kinds of
 web pages. To put it another way, thanks to the TeraBrowser, an individual is going to be able
 to store a website; then, after storing the content of a web page, a user is going to be able
 to see a website while he/she is not connected to a computer network. In this version, a
 user can use this program to store textual information, including the content of an HTML
 page; moreover, in a future version, this application may incorporate other features, such as
 storing images/audio/videos/PDFs. As you can see, those are some of the most important features
 that a user can taste after installing this application on his/her iPhone:
 1. Store and retrieve websites through a keyword.
 2. Activity indicator to perform an animation when a web is loading its content.
 3. Reload the content of a web page.
 4. Stop loading the content of a web page.
 5. Regular browser features to navigate through pages, such as go back/forward.
 */

//Importing the UIKit framework to be able to build an iOS application:
import UIKit
//Importing the CoreData framework to store and retrieve data:
import CoreData
//Importing the WebKit framework to build a browser:
import WebKit


//The AGene_Browser Class:
class AGene_Browser: UIViewController, UIWebViewDelegate, UISearchBarDelegate {

    //******************************** [ Outlet Variables ] ********************************
```

```
//The Search Bar field where a user has to type a URL:
@IBOutlet weak var AGene_SearchBar: UISearchBar!

//The activity indicator to show the loading process of a website:
@IBOutlet weak var AGene_ActivityIndicator: UIActivityIndicatorView!

//Retrieve TextField: A textfield to store the content of a web (for test purposes):
@IBOutlet weak var AGene_TextField_Retrieve: UITextField!

//Store TextField: A textfield to store/retrieve websites through a keyword:
@IBOutlet weak var AGene_TextField_Store: UITextField!

//URL TextField: A textfield to store a URL string:
@IBOutlet weak var AGene_TextField_URL: UITextField!

//The webView to display the content of a website:
@IBOutlet weak var webView: UIWebView!

//****************************** [ Global String Variables ] ******************************

/*After using the NSData function to save data (content of URL), we will use the
 AGene_URL_ContentofURL variable to store the content of a URL.*/
var AGene_URL_ContentofURL = "Teraexe";

/*After using the absoluteString function to get a URL string,
 we will store the URL on the next variable:*/
var AGene_URL_Get_URL = "Socket_0x03";

/*After using the HTTPMethod function to get an HTTPMethod (GET, POST...),
 we will store its value on the AGene_req_as_string variable: */
var AGene_req_as_string = "1986";

//******************************** [ Save Data Button ] ********************************

//Declaring a function to save data (the content of a web page):
@IBAction func btnSave_Data() {

    //Declaring a variable for the delegate of the application object:
    let AGene_AD: AppDelegate = (UIApplication.sharedApplication().delegate as! AppDelegate)
```

```
//Using the NSManagedObjectContext class to create a context for managed-objects:
let AGene_MO_Context: NSManagedObjectContext = AGene_AD.managedObjectContext

/*Using the insertNewObjectForEntityForName method to insert some values
 on a specific entity that is known as AGene_CoreData:*/
let AGene_CDEntity = NSEntityDescription.insertNewObjectForEntityForName("AGene_CoreData",
                inManagedObjectContext: AGene_MO_Context) as NSManagedObject

//**********Web Name (name to save/load a website):
/*We will use the setValue method to add some values on
 the agene_web_a attribute of the AGene_CoreData entity:*/
AGene_CDEntity.setValue(AGene_TextField_Store.text!,
                forKey: "agene_web_a")

//**********Web Content:
/*We will use the setValue method to add some values on
 the agene_web_b attribute of the AGene_CoreData entity:*/
AGene_CDEntity.setValue(AGene_URL_ContentofURL + // URL Content.
                //AGene_req_as_string +  // HTTP Method.
                AGene_TextField_Retrieve.text!,
                forKey: "agene_web_b")

//**********Web URL:
/*We will use the setValue method to add some values on
 the agene_web_c attribute of the AGene_CoreData entity:*/
AGene_CDEntity.setValue(AGene_URL_Get_URL + // URL String.
                AGene_TextField_URL.text!,
                forKey: "agene_web_c")


/*After using some methods like setValue to perform some operations on
 the AGene_MO_Context variable, a coder can use the save method to save
 his/her results. As you can see, we are using error handling techniques
 with do-try-catch while we are using the save method.*/
do {

   //Using the save method to save the content of a web page:
   try AGene_MO_Context.save()

   /*Using the print statement to display the content of a
    web page on the app terminal:*/
   print(AGene_URL_ContentofURL)
```

```
    }

    /*If the save method is not able to perform its task,
     the next block of code will display some errors.*/
    catch let error {

        /* Using the print statement to show information about an error; an error that
         states that the save button was not able to save the content of a web page.*/
        print(error)
    }

    //Using the print statement to show the data that we are storing in our application:
    print(AGene_CDEntity)

    //Showing a text message after saving or storing some information:
    print("The application was able to store the content of a web page.")

//Closing the btnSave_Data function:
}

//******************************** [ Load Data Button ] ********************************

//Declaring a function to load data (the content of a web page):
@IBAction func btnLoad_Data() {

    //Declaring a variable for the delegate of the application object:
    let AGene_AD: AppDelegate = (UIApplication.sharedApplication().delegate as! AppDelegate)

    //Using the NSManagedObjectContext class to create a context for managed-objects:
    let AGene_MO_Context: NSManagedObjectContext = AGene_AD.managedObjectContext

    /*Using the the NSFetchRequest class to retrieve data from a persistent store.
     While a coder is using this class, he/she has to specify an entity; in this case,
     it will be the AGene_CoreData entity.*/
    let AGene_CDRequest = NSFetchRequest(entityName: "AGene_CoreData")

    //Setting returnsObjectsAsFaults as false:
    AGene_CDRequest.returnsObjectsAsFaults = false;

    /*As you should know, a coder can use predicates to represent logical conditions.
     In the next code, we will use the NSPredicate class to define a logical condition,
     which will be used to fletch some results (from a persistent store to an app).*/
```

```
AGene_CDRequest.predicate = NSPredicate(format: "agene_web_a = %@",
                                AGene_TextField_Store.text!)

/*Using error-handling techniques with do-try-catch to be able to fetch some
 information. While we are using error-handling techniques, we are using
 the executeFetchRequest to fetch data from the AGene_CoreData entity.*/
do { let AG_Result: NSArray = try AGene_MO_Context.executeFetchRequest(AGene_CDRequest)

   /*The if-statement will execute a block of code if the AG_Result variable
    has a value that is more than zero; in this case, we will use the count
    statement and the > operator to check the values of the AG_Result variable.*/
   if(AG_Result.count > 0) {

      //Declaring a variable to store the values of the AG_Result variable:
      let AGene_ReReq = AG_Result[0] as! NSManagedObject


      //Using the valueForKey to return the values of the agene_web_a property:
      AGene_TextField_Store.text = AGene_ReReq.valueForKey("agene_web_a") as? String

      //Using the valueForKey to return the values of the agene_web_b property:
      AGene_TextField_Retrieve.text = AGene_ReReq.valueForKey("agene_web_b") as? String

      //Using the valueForKey to return the values of the agene_web_c property:
      AGene_TextField_URL.text = AGene_ReReq.valueForKey("agene_web_c") as? String


      //Storing the values (content of a web page) on the AG_Show_Field variable:
      let AG_Show_Field = AGene_ReReq.valueForKey("agene_web_b") as! String

      //Storing the values (URL of a web page) on the AG_Show_URL variable:
      let AG_Show_URL = AGene_ReReq.valueForKey("agene_web_c") as! String

      /*For test purposes, we will use the next three lines of code to display
       information on a terminal and test our iPhone application.*/

      //Showing the content of a website on a terminal:
      print(AG_Show_Field)

      //Displaying the URL of a website on a terminal:
      print(AG_Show_URL)
```

```
            //Displaying a text message if CoreData is able to load information:
            print("The application was able to load the content of a web page.")


            /*Using the NSURL object to represent a URL and store its value
             on the AGene_LD_URL variable:*/
            let AGene_LD_URL = NSURL(string: AG_Show_URL)

            /*Loading a website on the webView; in this case, we will use the
             loadHTMLString method to load the HTML content of a web page:*/
            self.webView!.loadHTMLString(AG_Show_Field, baseURL: AGene_LD_URL)

            //Closing the if-statement:
        }

        //Closing the do-statement:
    }

    /*If the previous block of code is not able to perform its task,
     the catch method will display some errors.*/
    catch let error {

        //Using the print statement to show some errors:
        print(error)
    }

//Closing the btnLoad_Data function:
}

//**************************** [ The viewDidLoad Function ] ****************************

override func viewDidLoad() {

    /*As a rule of Apple, while a coder is using the viewDidLoad method, he/she has
     to use the "super.viewDidLoad()". In some versions of Xcode, a coder will see
     some warnings if he/she doesn't use it.*/
    super.viewDidLoad()

    /*We have to set the UIWebView delegate before using
     the instances or objects of the UIWebView Class:*/
    webView.delegate = self;
```

```
    /*We have to set the UISearchBar delegate before using
    the instances or objects of the UISearchBar class:*/
    AGene_SearchBar.delegate = self;

    /*Setting the keyboardDisplayRequiresUserAction value as true. When a coder sets
     this value as true, the application can display a keyword and/or other input views
     when a user is interacting with an element, such as the content of a web-page.
     For example, if a user is using a search engine website, he/she can see the keyword
     when he/she touches a text field to type a word.*/
    webView.keyboardDisplayRequiresUserAction = true

    webView.frame = self.view.frame;
}

//******************** [ The searchBarSearchButtonClicked Function ] ********************

/*The searchBarSearchButtonClicked Method: This method will communicate with
 the UISearchBarDelegate protocol when a user clicks the search button
 (or a user types return to show the content of a web).*/
func searchBarSearchButtonClicked(searchBar: UISearchBar) {

    /* UIKit Framework > UIResponder Class > resignFirstResponder Method.
    (a) The UIResponder Class: It responds to events, such as touch/motion events.

    (b) The resignFirstResponder Method:
     If a user clicks the text-field of a search-bar, the iPhone will show the keyboard
     and the text-field will become "the first responder". Then, if a user performs a
     different task (like typing return or clicking outside of the text-field of
     a search-bar), the keyboard is going to disappear and the text-field will not be
     "the first responder" thanks to the resignFirstResponder method.
     To put it another way, if you don't use the resignFirstResponder method, the
     text-field of a search-bar will still be "the first responder" (the keyword will
     not disappear) if a user clicks outside the text-field or types return.*/
    searchBar.resignFirstResponder()

    //Declaring the AGene_SearchBar_Text variable to search text:
    let AGene_SearchBar_Text = searchBar.text

    /* Inherits Form: NSObject Class > NSURL Class
     Note: A coder can use the NSURL object when he/she wants to represent a URL.
     Using the NSURL object to declare the AGene_NSURL variable: */
    let AGene_NSURL = NSURL(string: AGene_SearchBar_Text!)
```

```
    /* Inherits Form: NSObject Class > NSURLRequest Class
     Note: A coder can use the NSURLRequest object when he/she wants to represent a
     URL request. An important point that a coder should know about the NSURLRequest
     class is that it only encapsulate data from a URL request; therefore, a coder has
     to use other classes to load the content of a web or send a request to a server.
     Using the NSURLRequest object to declare the AGene_URLRequest variable: */
    let AGene_URLRequest = NSURLRequest(URL: AGene_NSURL!)


    //*************************The Three Variables:

    //*****1. Content of URL:
    //Using the NSData function to store the content of a URL on a variable:
    let AGene_URL_as_NSData: NSData = NSData(contentsOfURL: AGene_NSURL!)!
    //the AG_URLDStr variable will store the unicode String value:
    let AG_URLDStr = String(data: AGene_URL_as_NSData, encoding: NSUTF8StringEncoding)
    //Storing the AG_URLDStr value on the AGene_URL_ContentofURL variable
    AGene_URL_ContentofURL = AG_URLDStr!;

    //*****2. URL String:
    //The AGene_URL_absoluteString variable will store a URL string:
    let AGene_URL_absoluteString : String = AGene_NSURL!.absoluteString
    //Storing the previous value on the AGene_URL_Get_URL variable:
    AGene_URL_Get_URL = AGene_URL_absoluteString;

    //*****3. HTTPMethod:
    //Using the HTTPMethod function to store the HTTPMethod on a variable:
    let AGene_HTTPMethod : String = AGene_URLRequest.HTTPMethod!
    //Storing the HTTPMethod on the AGene_req_as_string variable:
    AGene_req_as_string = AGene_HTTPMethod;


    /* UIKit Framework > UIWebView Class > Information about loadRequest
     The loadRequest Method: A coder can use this method to start loading the
     content of a web page. After storing a URL on the AGene_URLRequest variable,
     we will use the loadRequest method to start loading the content of a website.*/
    self.webView!.loadRequest(AGene_URLRequest)

//Closing the searchBarSearchButtonClicked function:
}
```

```
    //************************ [ The webViewDidStartLoad Function ] ************************

    /* Using the webViewDidStartLoad method (of the UIWebViewDelegate protocol) on
    the webView variable (of the UIWebView class) to be able to perform an animation
    when a website is starting to load its content.*/
    func webViewDidStartLoad(webView : UIWebView) {

        /*Using the startAnimating method on the AGene_ActivityIndicator variable to
         start animating "the activity indicator" when a website is loading its content.*/
        AGene_ActivityIndicator.startAnimating()
    }

    //************************ [ The webViewDidFinishLoad Function ] ************************

    /* Using the webViewDidFinishLoad method (of the UIWebViewDelegate protocol) on
    the webView variable (of the UIWebView class) to be able to perform an animation
    when a website is not loading its content.*/
    func webViewDidFinishLoad(webView : UIWebView) {

        /*Using the stopAnimating method on the AGene_ActivityIndicator variable to stop
         animating "the activity indicator" when a website finished loading its content.*/
        AGene_ActivityIndicator.stopAnimating()
    }

    //********************** [ The didReceiveMemoryWarning Function ] **********************

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()

    }

//Closing the AGene_Browser Class:
}
```