monkehTweet v1.3.1

Thank you for downloading monkehTweet, a ColdFusion package developed to interact with the Twitter APIs.

License and Credits

Authors

Developed by Matt Gifford AKA coldfumonkeh
http://www.mattgifford.co.uk

Got a lot out of this package? Saved you time and money?
Share the love and visit Matt's wishlist:
http://www.amazon.co.uk/wishlist/B9PFNDZNH4PY

Requirements

monkehTweet requires ColdFusion 8+

Installation

Unzip the package archive to the desired location (typically in the web root). After installation, you will see the following directories:

- **installation** (contains this installation guide, no more, no less)
- **com** (contains all of the ColdFusion Components required for the application to interact with the Twitter API)

The application does not interact with any database, and as such needs no data sources.

Getting Started

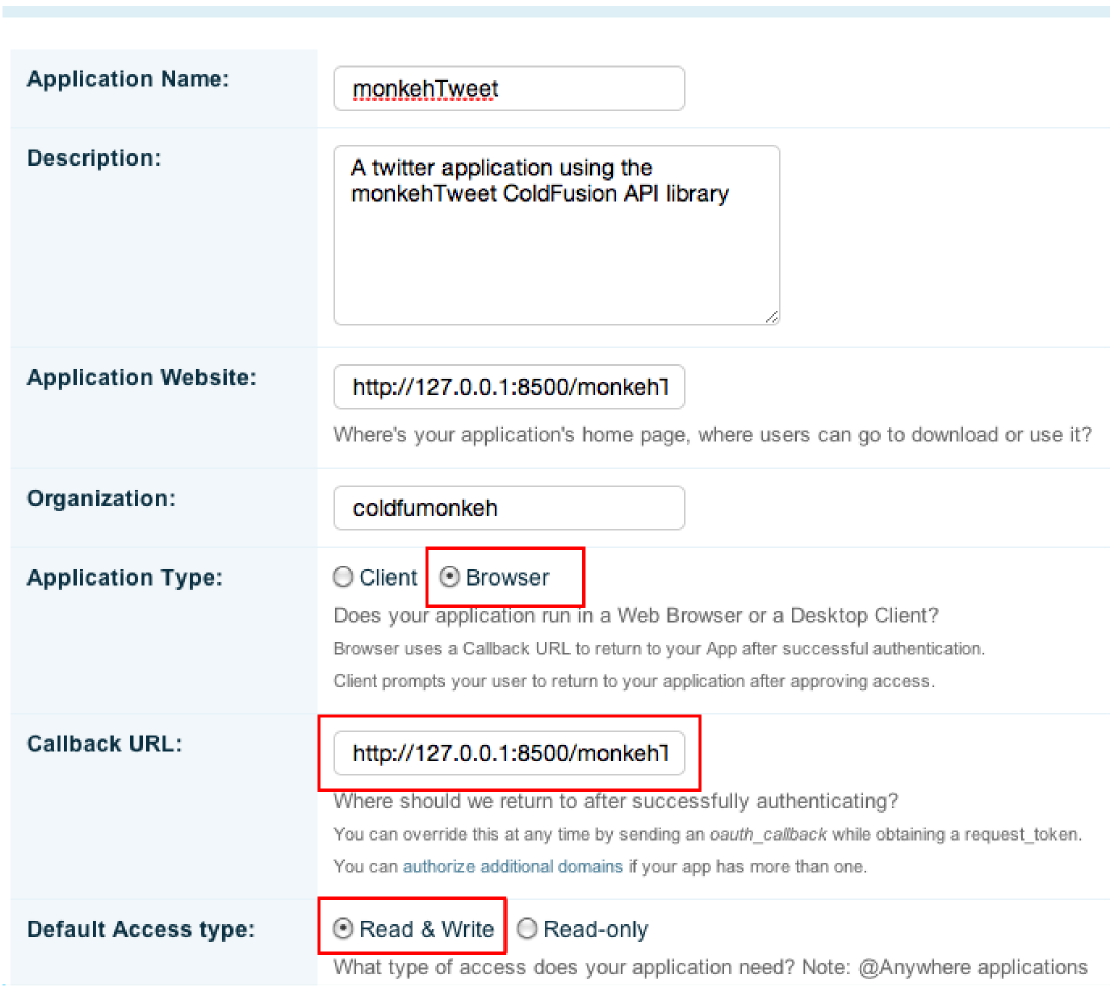
Version 1.2.4+ of monkehTweet was released to include the OAuth authentication protocol, which now completely replaces the basic authentication method previously offered by Twitter.

To interact with the Twitter API, you will need to create an application with Twitter, which generates the consumer key and secret values that are required to instantiate and work with monkehTweet (and indeed the API itself).

Visit http://dev.twitter.com/apps to register a new application.

Items to note in here:

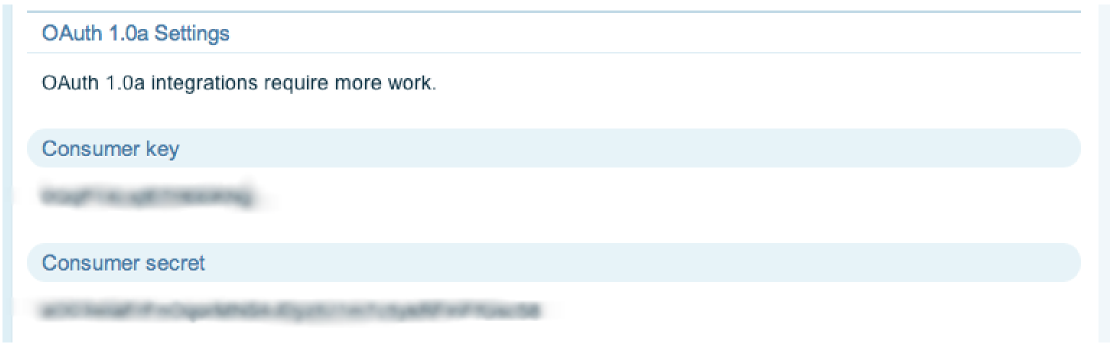
- Application type – set to browser
- Callback URL – you can set your initial callback URL here (e.g. http://localhost:8500/authorize.cfm) but you can overwrite this when making the actual authorization call from monkehTweet
- Access type – set to Read & Write to allow posts



Application Name:	monkehTweet
Description:	A twitter application using the monkehTweet ColdFusion API library
Application Website:	http://127.0.0.1:8500/monkehT Where's your application's home page, where users can go to download or use it?
Organization:	coldfumonkeh
Application Type:	Client [✓ Browser] Does your application run in a Web Browser or a Desktop Client? Browser uses a Callback URL to return to your App after successful authentication. Client prompts your user to return to your application after approving access.
Callback URL:	http://127.0.0.1:8500/monkehT Where should we return to after successfully authenticating? You can override this at any time by sending an *oauth_callback* while obtaining a request_token. You can authorize additional domains if your app has more than one.
Default Access type:	[✓ Read & Write] Read-only What type of access does your application need? Note: @Anywhere applications

Figure 1 – The initial application page

Having created your application, you need to make a note of the consumer details (the consumer key and consumer secret). These are required when instantiating the monkehTweet component.



OAuth 1.0a Settings

OAuth 1.0a integrations require more work.

Consumer key

[redacted]

Consumer secret

[redacted]

Figure 2 – The required consumer details from the application page

If you are using the application for numerous users (a number of accounts using your application to interact with Twitter) then your job here is done.

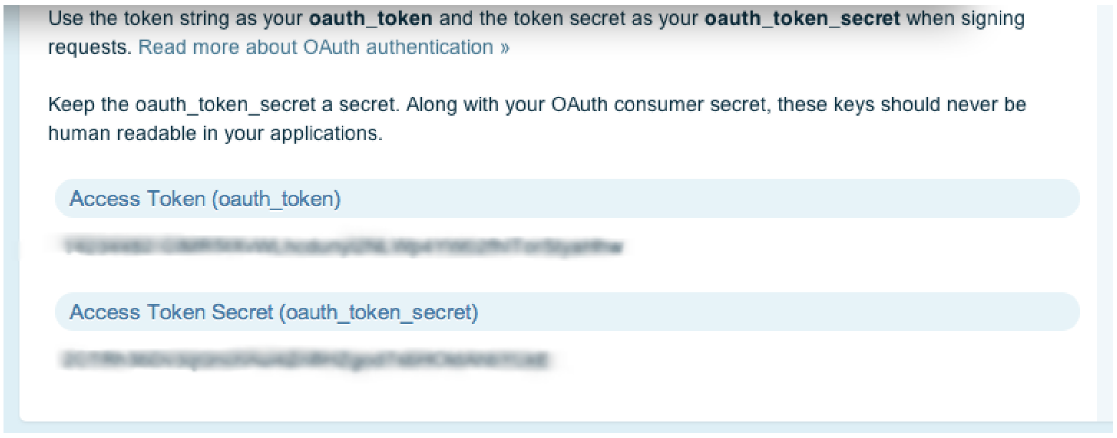
Simply instantiate the component with the consumer details and set up your code to request authentication and access for each user, as per the code samples within the monkehTweet download.

Instantiation is incredibly easy, as seen in the below example:

```
<cffunction name="OnApplicationStart"
             access="public"
             returntype="boolean"
             output="false">
       <cfscript>
       application.objMonkehTweet =
              createObject('component',
              'com.coldfumonkeh.monkehTweet')
              .init(
              consumerKey          = '< enter your consumer key >',
              consumerSecret       = '< enter your consumer secret >',
              parseResults         = true
              );
              return true;
       </cfscript>
</cffunction>
```

If you are intending to be the sole user (only your Twitter account will access the API) then you can bypass the requirement for authentication.

Within the application details page for your new application on http://dev.twitter.com, select the ‘My Access Token’ option from the right-hand menu.



Use the token string as your **oauth_token** and the token secret as your **oauth_token_secret** when signing requests. Read more about OAuth authentication »

Keep the oauth_token_secret a secret. Along with your OAuth consumer secret, these keys should never be human readable in your applications.

Access Token (oauth_token)

Access Token Secret (oauth_token_secret)

Figure 3 – Obtaining the access details from your application page

Copy the access token and access token secret (the OAuth tokens) from this screen, and use these along with your Twitter account screen name in the init() method when instantiating the monkehTweet component, as seen below:

```
<cffunction name="OnApplicationStart"
             access="public"
             returntype="boolean"
             output="false">
       <cfscript>
       application.objMonkehTweet =
              createObject('component',
              'com.coldfumonkeh.monkehTweet')
              .init(
              consumerKey          = '< enter your consumer key >',
              consumerSecret       = '< enter your consumer secret >',
              oauthToken           = '< enter the access token >',
              oauthTokenSecret     = '< enter the access secret >',
              userAccountName      = '< enter your twitter screen name >',
              parseResults         = true
              );
              return true;
       </cfscript>
</cffunction>
```

This essentially bypasses the need for OAuth requests and redirects as we now have the access details for your account. You are ready to use monkehTweet instantly.

Lucky you. ☺

@Anywhere

Twitter @Anywhere is an easy-to-deploy solution for bringing the Twitter communication platform to your site. @Anywhere promotes a more engaged user base for your site. Use @Anywhere to add Follow Buttons, Hovercards, linkify Twitter usernames, and build deeper integrations with "Connect to Twitter."

monkehTweet now comes with the @Anywhere functionality built in to help you easily add these functions to your websites and applications.

For more information on the @Anywhere methods, visit the official Twitter docs: https://dev.twitter.com/docs/anywhere/

Linkify Users

@Anywhere provides a convenient way to link Twitter usernames found in your web site or application back to a user's profile page on Twitter.com. A Twitter screen name is an '@' symbol followed by 1 to 20 alphanumeric characters, including underscores ("_"). *@ev* or *@biz* and two examples of Twitter username.

Here's how you can link usernames within your page content directly to their Twitter profile pages:

```
<cfoutput>#application.objMonkehTweet.linkifyUsers()#</cfoutput>
```

This will convert ALL Twitter names into a link.
There are two optional parameters that can be used with this method:

- **htmlSection** - the ID of the document element to apply the linkifyUsers to. If left blank, all Twitter names within the document will be linkified. This allows you to choose only a section of your site to apply the method to.

- **className** - by default, linkifying usernames will wrap matched names in an anchor element with a class of 'twitter-anywhere-user'. Here you can specify an alternate class name to adjust to suit your CSS.

```
<cfoutput>
	#application.objMonkehTweet.linkifyUsers(
								htmlSection='testsection',
								className='myClass'
							)#
</cfoutput>
```

In the above example, only names within the DOM element with the *id* attribute set to *testsection* will be linkified, and we are applying a CSS class of *myClass* to style them.

Hovercards

Hovercards are a feature that can be seen on Twitter.com and are now available to developers through @Anywhere. A Hovercard is a small, context-aware tooltip that provides access to data about a particular Twitter user. Hovercards also allows a user to take action upon a Twitter user such as following and un-following, as well as toggling device updates.

Here's how you can apply Hovercards to Twitter usernames within your page content:

```
<cfoutput>#application.objMonkehTweet.linkifyUsers()#</cfoutput>
```

This will apply a Hovercard to ALL Twitter names within your page.
There are four optional parameters that can be used with this method:

- **htmlSection** - the ID of the document element to apply the Hovercard to. If left blank, all Twitter names within the document will have a Hovercard . This allows you to choose only a section of your site to apply the method to.

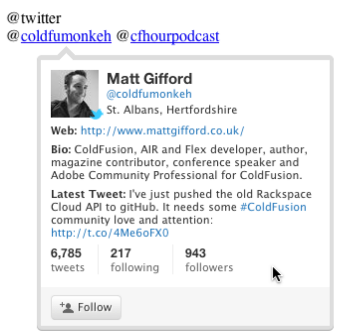
- **linkify** - If Twitter names have already been linkified elsewhere, set to false.

- **infer** - Use the infer option to trigger Hovercards on elements whose text contains a Twitter username. When the infer option is used, the Hovercards method will not call the linkifyUsers method. This is useful when Twitter usernames have already been linkified by some other means. For example: <a ...>Follow @coldfumonkeh on Twitter!</a>.

- **expanded** - Set to true to render the Hovercards in expanded state by default.

```
<cfoutput>
	#application.objMonkehTweet.hovercards(
								htmlSection='testsection',
								expanded=true
							)#
</cfoutput>
```

In the above example, only names within the DOM element with the *id* attribute set to *testsection* will have Hovercards applied to them, and we are setting them to be expanded by default. The result would look something like this:



@twitter
@coldfumonkeh @cfhourpodcast

Matt Gifford
@coldfumonkeh
St. Albans, Hertfordshire

Web: http://www.mattgifford.co.uk/

Bio: ColdFusion, AIR and Flex developer, author, magazine contributor, conference speaker and Adobe Community Professional for ColdFusion.

Latest Tweet: I've just pushed the old Rackspace Cloud API to gitHub. It needs some #ColdFusion community love and attention: http://t.co/4Me6oFX0

6,785 tweets | 217 following | 943 followers

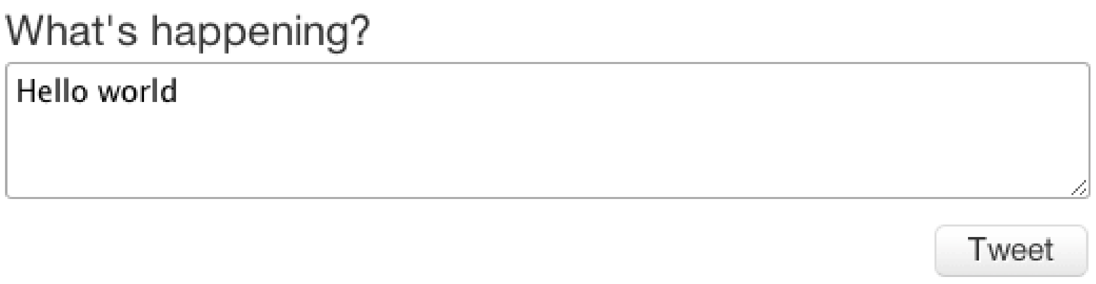
Tweet Boxes

The Tweet Box allows Twitter users to tweet directly from within your web site or web application.

The following example places the Tweet Box in the element with the *id* attribute of *tweetbox*:

```
<cfoutput>
	#application.objMonkehTweet.tweetBox(
				htmlSection='tweetbox',
				defaultContent='Hello world'
			)#
</cfoutput>
```

The default implementation of the Tweet Box will produce the following:



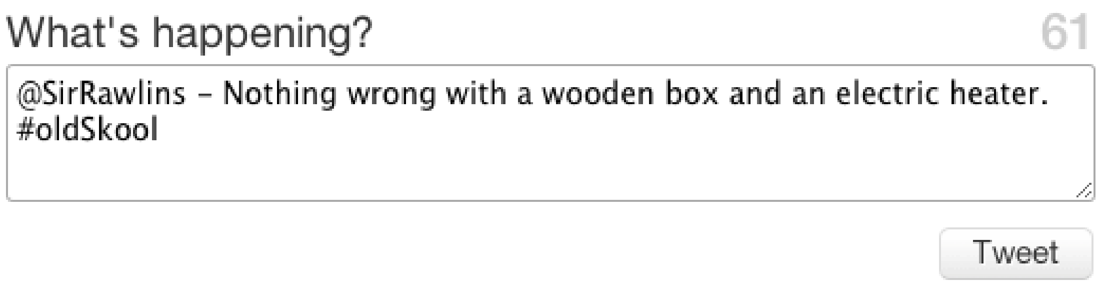
There are a number of configuration options that you can apply to customise this feature:

- **counter** - display a counter in the Tweet Box for counting characters. True or false.
- **height** - The height of the Tweet Box in pixels.
- **width** - The width of the Tweet Box in pixels.
- **label** - The text above the Tweet Box, a call to action.
- **defaultContent** - Pre-populated text in the Tweet Box. Useful for an @mention, a #hashtag, a link, etc.
- **onTweet** - Specify a listener for when a tweet is sent from the Tweet Box. The listener receives two arguments: a plaintext tweet and an HTML tweet.
- **data** - Key + value pairs representing any of the additional metadata that can be set when updating a user's status. See the REST API documentation for a complete list of the possible options.

Here's another example with extra options:

```
<cfoutput>
	#application.objMonkehTweet.tweetBox(
		htmlSection		=	'tweetbox',
		counter			=	true,
		defaultContent		=	'@SirRawlins - Nothing wrong
						with a wooden box and an electric
						heater. ##oldSkool',
		data			=	{
			in_reply_to_status_id	=	'154863490811699200'
		}
	)#
</cfoutput>
```

In the above code we have provided the default content text for the Tweet Box, and as this is a reply to a particular status, we can use the data argument and create a struct to send in that references the original status ID.



@SirRawlins – Nothing wrong with a wooden box and an electric heater. #oldSkool

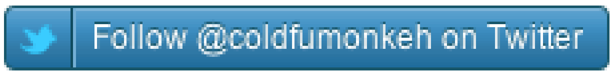
Follow Button

Follow buttons make it easy to provide users of your site or application with a way to follow users on Twitter.

Adding a follow button to your web site or web application using monkehTweet is easy:

```
<cfoutput>
	#application.objMonkehTweet.followButton(
						htmlSection='followbutton',
						twittername='coldfumonkeh'
					)#
</cfoutput>
```

This will apply the follow button for the supplied *twittername* value into the *htmlSection* with the *id* attribute set to *followbutton*. The result of this would look like so:



User Authentication

The "Connect with Twitter" button provides a method for users to authenticate securely with Twitter, yielding your application with an access token for use in API calls.

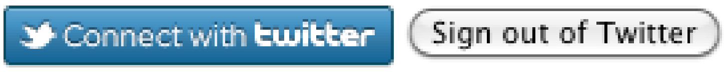
Adding "Connect with Twitter" buttons to your application using monkehTweet is easy:

```
<cfoutput>

	#application.objMonkehTweet.login(htmlSection='login')#

</cfoutput>
```

In the above code, we are applying the login button to the element with the *id* attribute set to *login*.

You can sign out of Twitter by setting an HTML element within the page that references the twttr JavaScript object:

```
<button type="button"

	onclick="twttr.anywhere.signOut();">Sign out of Twitter</button>
```



Multiple Methods At Once

Each of the above @Anywhere methods will generate their own JavaScript methods onto your page and include the required external JS file from Twitter.

What if you want to use more than one of these functions in your ColdFusion application? monkehTweet has you covered. You can use a single method called twitAnywhere. This accepts one argument called *params*, which is a structure containing all data required to instantiate and display the other methods.

Here's a simple example in which we want to create Hovercards, a Follow Button and a Tweet Box for our page:

```
<cfset stuParams = {
	hovercards		=	{
		htmlSection		=	'testsection',
		expanded		=	true
	},
	followButton =	{
		htmlSection		=	'followbutton',
		twittername		=	'coldfumonkeh'
	},
	tweetBox		=	{
		htmlSection		=	'tweetbox',
		counter			=	true
	}

} />

<cfoutput>#application.objMonkehTweet.twitAnywhere(stuParams)#</cfoutput>
```

As you can see, the params variable contains a structure for each method we wish to call, and this structure holds the arguments for each method.

monkehTweet takes care of building the single JavaScript function for you, combining all of your requested methods into the one request.

Limitations

Remaining functionality

monkehTweet is currently up to date with the Twitter API.

Testing

monkehTweet has been tested on the Adobe ColdFusion platform.
It has yet to be tested fully on Railo or OpenBlueDragon.