# Twilex Documentation

***Release 0.1***

**Erika Heidi**

July 21, 2014

Contents

Twilex provides a simple way for bootstrapping Twitter applications, using Silex, Flint and TTools. If you are used to Symfony conventions, it will be really easy for you to get started.

CHAPTER 1

# Getting Started Guide

Twilex provides a simple way for bootstrapping Twitter applications, using Silex, Flint and TTools.

This skeleton project features a functional Twitter authentication (sign-in with Twitter) so you can start developing your application without worrying about the complicated process of tokens exchange and session handling for users authorization. The application follows Symfony conventions.

If you are new to Silex, please have a look at their documentation.

For more info about the TTools library and its features / docs, check its repository: https://github.com/ttools/ttools

## 1.1 1. Creating Twilex skeleton

You'll need Composer. If you don't have Composer yet, install it by running:

```
$ curl -sS https://getcomposer.org/installer | php
```

Now run the *create-project* command:

```
$ php composer.phar create-project twilex/twilex-standard [directory-name] [version]
```

Where *version* should be the latest stable release of twilex-standard.

Example:

```
$ php composer.phar create-project twilex/twilex-standard myTwitterClient 0.1.3
```

## 1.2 2. Configuring Twilex

Now you need to create a configuration file for Twilex:

```
$ cd MyTwitterApp
$ cp app/config/config.yml.dist app/config/config.yml
```

Edit the *config.yml* file to configure its settings. This is how the default *config.yml.dist* file looks like:

```
config:
    name: Twilex
    author: Erika Heidi
    description: Simple Twitter App micro framework
    tags: twitter, silex, micro
```

```
twitter.keys:
    api_key: REPLACE_WITH_YOUR_APP_KEY
    api_secret: REPLACE_WITH_YOUR_APP_SECRET

locale: en_GB

bundles.path: "%root_dir%/src"

active_bundles:
    - "%root_dir%/src/ExampleBundle"

http_cache.cache_dir: "%root_dir%/app/cache/http"
routing.resource: "%root_dir%/app/config/routing.yml"
```

Edit the APP info and provide the Twitter API keys from your Twitter App.

## 1.3 3. Running the Application

### 1.3.1 Using Vagrant (recommended)

To get started even more quickly, you can use the included Vagrant setup.

Dependencies:

- Vagrant
- VirtualBox
- Ansible

Having these dependencies installed, you can just run:

```
$ vagrant up
```

And the application will be available at *http://192.168.13.37*. Then you can use *http://192.168.13.37/connect* as the callback url of your Twitter App

*note: Windows is not officially supported by Ansible. If you're using a Windows machine for developing, you'll need to use a local web server.*

### 1.3.2 Running it locally

If you prefer using a local web server, it's recommended that you create a **vhost** using a **local domain**, such as "myapp.local". It makes easier for the callback URL configuration. In this scenario, you would use the URL *http://myapp.local/connect*.

CHAPTER 2

# Developing your Application

## 2.1 Creating Bundles

It's recommended that you create bundles to organize your application. The basic app comes with an example bundle to serve as model for your custom bundles.

There's a console command to help creating the basic Bundle structure. Run:

```
$ php app/console bundle-create MyBundleName
```

A structure like this will be created:

```
src/MyBundleName
-- Controller
|   -- DefaultController.php
-- Resources
    -- config
    |   -- routing.yml
    -- views
        -- default
            -- index.html.twig
```

The name provided, in this case *MyBundleName*, will be the namespace of your bundle.

### 2.1.1 Activating a Bundle

You'll need to add the new bundle path to the file *app/config/config.yml*, under the item "active_bundles", as shown below:

```
active_bundles:
    - "%root_dir%/src/ExampleBundle"
    - "%root_dir%/src/MyCustomBundle1"
    - "%root_dir%/src/MyCustomBundle2"
```

You'll also need to include the routing file for the bundle inside the main *app/config/routing.yml*. Edit this file and add the relevant import statements:

```
mycustombundle1_main:
    resource: "src/MyCustomBundle1/Resources/config/routing.yml"

mycustombundle2_main:
    resource: "src/MyCustomBundle2/Resources/config/routing.yml"
```

## 2.2 Overwriting Templates

You can overwrite the default templates that come with the core Twilex bundle. To overwrite a template, you just need to create a template under the same structure as the default ones. The default templates are located at the main twilex/twilex bundle, at *vendor/twilex/twilex/src/Twilex/Resources/views*.

If you check the *ExampleBundle* views folder, you'll notice that the template *default/index.html* overwrites the default *index* template from the twilex/twilex bundle.

*Note: when overwriting templates, the highest priority is for the first bundle included in the list of 'active_bundles'.*

### 2.2.1 Assets

For now, all assets are located in the *web* directory, in the root of the application. For future versions, an integration with Assetic for assets management is planned.

## 2.3 Default Routes

Twilex comes with two main routes that you'll need for connecting your users to Twitter:

### 2.3.1 /connect

This is the entrypoint for authorization on Twitter. After authorizing, the user should be redirected here in order to finish the authentication process. You need to use this route in the callback url of your Twitter App.

### 2.3.2 /logout

This is the route for logging out the user.

### 2.3.3 Customizing

You can also create custom routes / controllers for the authorization process. Check the ConnectController (from the main twilex/twilex bundle) to have an idea of how we send a user for authorization.

## 2.4 Creating Console Commands

The ExampleBundle included in the installation has an example of a Console Command. You can create your own console commands, and to make them available in the included *app/console* you just need to place them inside a "Command" folder in your bundle, following the standard nomenclature "SomethingCommand.php" for the class name.

Check *ExampleBundle/Command/ExampleCommand.php* to see how you can create Commands, or have a look at the Symfony documentation about the Console component.

CHAPTER 3

# Twitter APP Creation

In order to use Twilex and have access to the Twitter API, you need to create an application on the Twitter Developers page. If this is the first time you do this, here you can find a step-by-step guide.

## 3.1 Sign-up / Sig-in

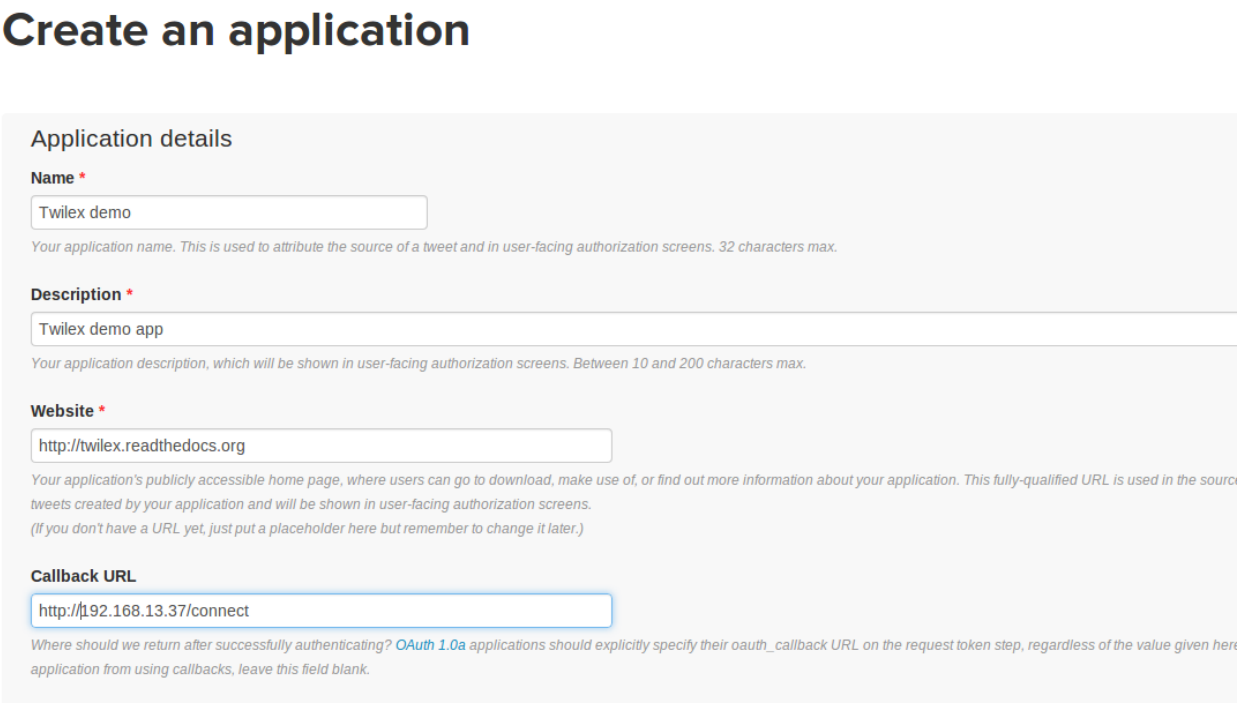
Go to http://dev.twitter.com and sign in.

## 3.2 Access your applications

Access the item "My Applications" from the menu.

## 3.3 Create the application

Now click on "Create New App". You'll fill a form like this:




### 3.3.1 Name

The application name that will be used in the authorization screen.

### 3.3.2 Description

A comprehensive description that will be used in the authorization screen.

### 3.3.3 Website

A website were people can find more information about the application.

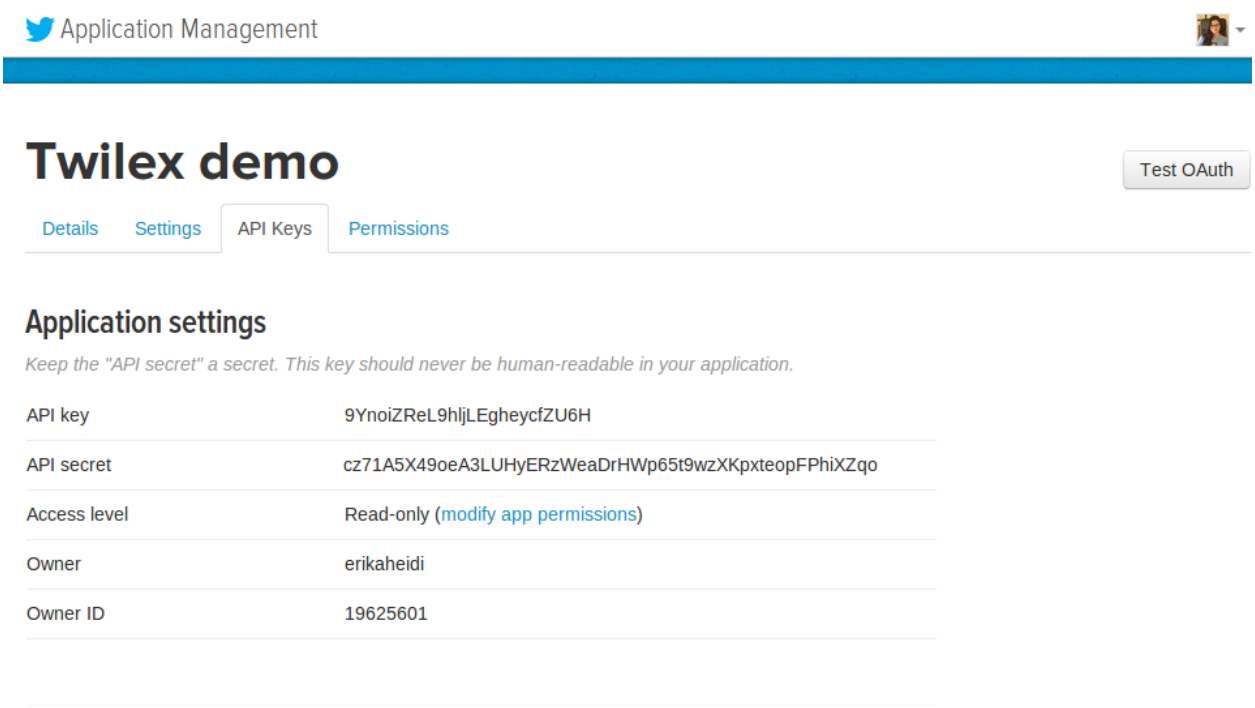
### 3.3.4 Callback URL

This is **important** when you want to create an application that authenticates users, which is most probably the case.

The callback URL is a URL to where the user will be redirected right after authorizing the application. Twilex handles the authorization process at the default route */connect* in your application. In this example, I'll be using the Vagrant setup included, which will create a virtual machine in my local network in the specified IP address (default from the *Vagrantfile* provided with Twilex, but you can change it). After running *vagrant up* the application will be found at http://192.168.13.37, and users will be able to connect to Twitter accessing http://192.168.13.37/connect .

## 3.4 Now get your Keys

You'll need the **API key** (consumer_key) and **API secret** (consumer_secret) in order to configure Twilex. These keys are available at the tab "API Keys" in your application settings:




## 3.5 Note about app permissions

The default permission for new apps is **read only**. With this permission you won't be able to create tweets or change any data. If you think you will need write permission, you can edit your application settings to make the change.

CHAPTER 4

# Crash-course

## 4.1 1. Install the skeleton

First, create the application skeleton using *composer create-project* (you will need Composer):

```
$ php composer.phar create-project twilex/twilex-standard [project-folder] [version]
```

Where the version should be the latest stable release. Check the repository in order to get this information: https://github.com/twilex/twilex-standard

Example:

```
$ php composer.phar create-project twilex/twilex-standard MyTwitterApp 0.1.3
```

## 4.2 2. Configure the App settings

```
$ cd MyTwitterApp
$ cp app/config/config.yml.dist app/config/config.yml
```

Now edit the *config.yml* file to configure its settings. You'll need to provide the API keys from your Twitter App.

## 4.3 3. Run the App

### 4.3.1 Using Vagrant

Twilex includes a Vagrant setup built with Ansible.

Dependencies:

- Vagrant
- VirtualBox
- Ansible

Having these dependencies installed, you can just run:

```
$ vagrant up
```

And the application will be available at *http://192.168.13.37*. Then you can use *http://192.168.13.37/connect* as the callback url of your Twitter App

### 4.3.2 Using a local web server

If you prefer using a local web server, it's recommended that you create a **vhost** using a **local domain**, such as "myapp.local". It makes easier for the callback URL configuration. In this scenario, you would use the URL *http://myapp.local/connect*.