SwiftRiver v1.0

NOTE: Before proceeding with this section, ensure your environment meets the system requirements

On this page

Get the code from GitHub

Get a copy of the latest (stable) code from our v1.0 tree on GitHub Repository using the following steps:

1. Create a `SwiftRiver` directory for the project and set up the necessary Git files

```
$ mkdir SwiftRiver && cd SwiftRiver && git init
$ git remote add origin git://github.com/ushahidi/SwiftRiver.git
$ git fetch
```

2. Checkout v1.0 of the application

```
$ git checkout -b v1.0 v1.0
$ git submodule init && git submodule update
```

These steps will pull in the code for the v1.0 tree into a newly created `SwiftRiver` directory. Copy this directory to your web server's document root or `public_html` directory.

Create application directories

Create following directories and ensure they are writable:

```
mkdir application/cache
mkdir application/logs
```

Create configuration files

Create a `.php` for each of the `.php.template` files in your `application/config` directory. This can be done via the command line as follows:

```
cp application/config/site.php.template application/config/site.php
cp application/config/database.php.template application/config/database.php
cp application/config/cache.php.template application/config/cache.php
cp application/config/auth.php.template application/config/auth.php
cp application/config/cookie.php.template application/config/cookie.php
```

Alternatively, you can use the following one-liner:

```
for config in application/config/*.php.template; do cp $config application/config/`basename ${config} .php.template`.php; done
```

Create the database

Log in to your MySQL server (via the command line) as follows:

```
mysql -u <username> -p
```

MySQL will prompt you for the password associated with `<username>`. Once logged in, run the following command at the MySQL prompt to create the database that shall host the data for your SwiftRiver installation.

```
create database <swiftriver-database>;
```

Where `<swiftriver-database>` is the name of your SwiftRiver database.
NOTE: `<username>` should be an account that has privileges to create a database on your MySQL server.

Next, run the following command (also at the MySQL prompt):

```
GRANT CREATE ROUTINE, CREATE VIEW, ALTER, SHOW VIEW, CREATE, ALTER ROUTINE, EVENT, INSERT, SELECT, DELETE, TRIGGER, GRANT OPTION,
REFERENCES, UPDATE, DROP, EXECUTE, LOCK TABLES, CREATE TEMPORARY TABLES, INDEX
ON <swiftriver-database>.* TO <swiftriver-user>@'localhost' IDENTIFIED BY <swiftriver-user-password>;
```

Where:

- `<swiftriver-database>` is the name of your SwiftRiver database
- `<swiftriver-user>` is the username to use when connecting to your SwiftRiver database
- `<swiftriver-user-password>` is the password associated with the user account to be used for connecting to your SwiftRiver database

Run the database schema setup script

The schema setup script is located at `/path/to/SwiftRiver/install/sql/swiftriver.sql`

```
mysql <swiftriver-database> -u <swiftriver-user> -p < /path/to/SwiftRiver/install/sql/swiftriver.sql
```

NOTE: If you get the following error,

```
ERROR 1418 (HY000) at line 801: This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you might want to use the less safe log_bin_trust_function_creators variable)
```

it's because MySQL gets paranoid when you attempt to create a function that is not deterministic and/or attempts to modify data. See http://dev.mysql.com/doc/refman/5.5/en/stored-programs-logging.html for a more detailed explanation. To circumvent this restriction, set the `log_bin_trust_function_creators` system variable to 1. You can do this from the MySQL prompt as follows:

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

Update the database configuration

Update your database configuration file (`application/config/database.php`) with the values you used for the `swiftriver-` parameters in the preceding steps.
The updated database configuration should read as follows:

```
return array
(
 'default' => array
 (
  'type'       => 'mysql',
  'connection' => array(
   'hostname'   => 'localhost',
   'database'   => '<swiftriver-database>',
   'username'   => '<swiftriver-user>',
   'password'   => '<swiftriver-user-password>',
   'persistent' => FALSE,
  ),
  'table_prefix' => '',
  'charset'      => 'utf8',
  'caching'      => TRUE,
  'profiling'    => TRUE,
 )
);
```

Configuring Background Jobs

Content Crawling and Post Processing

Add the following entries to your crontab to schedule content crawling every 30 minutes and post processing (semantic tagging, media extraction etc) every 15 minutes respectively:

```
0,30 * * * * cd /path/to/swiftriver; php5 index.php --task=crawler >> application/logs/crawl.log 2>&1
0,15,30,45 * * * * cd /path/to/swiftriver; php5 index.php --task=process >> application/logs/process.log 2>&1
```

River Expirty

River maintenance involves checking which rivers have expired and are about to expire and sending out notifications to their owners. To schedule maintenance to run every day at midnight, add the following entries to your crontab:

```
* 0 * * * cd <app home>; php5 index.php --task=river:expire >> application/logs/river_expiry.log 2>&1
```

Note: You may want to add cron jobs using the following command so that any files created by the cron jobs are writeable by the Apache user:

```
crontab -u <apache user> -e
```

Additional Configuration

Apache Configuration

SwiftRiver's is bundled with a configuration file (`.htaccess`) that is used to enable enable URL rewriting on an Apache Webserver. The objective of the URL rewrite is to produce clean URLs. In order for Apache to honor the directives in the `.htaccess`, you need to modify the `AllowOverride` directive in `<Directory>` section for your document root so that it reads as follows:

```
<Directory "/path/to/DocumentRoot">
   AllowOverride All
   ...
   ...
   ...
</Directory>
```

For more information on this directive, click here

Modify .htaccess and bootstrap.php

By default, the first few sections of the SwiftRiver `.htaccess` file reads as follows:

```
# Turn on URL rewriting
RewriteEngine On

# Installation directory
RewriteBase /
...
...
```

This is based on the premise that, the deployment URL is of the form http://www.swiftriver-deployment.com. However, if you are running the application on your local machine, chances are that the deployment URL shall be something like http://localhost/swiftriver. Therefore, you will need to modify the `RewriteBase` in the `.htaccess` so that the file (`.htaccess`) reads as follows:

```
# Turn on URL rewriting
RewriteEngine On

# Installation directory
RewriteBase /swiftriver
...
...
```

The next step is to edit `boostrap.php` - `/path/to/swiftriver-root/application/bootstrap.php` and modify the section that reads:

```
Kohana::init(array(
 'base_url'   => '/',
 'index_file' => '',
 'cache_dir' => APPPATH.'/cache',
 'caching' => Kohana::$environment === Kohana::PRODUCTION,
 'profiling' => Kohana::$environment !== Kohana::PRODUCTION,
 'errors' => TRUE
));
```

to:

```
Kohana::init(array(
 'base_url'   => 'http://localhost/swiftriver',
 'index_file' => '',
 'cache_dir' => APPPATH.'/cache',
 'caching' => Kohana::$environment === Kohana::PRODUCTION,
 'profiling' => Kohana::$environment !== Kohana::PRODUCTION,
 'errors' => TRUE
));
```

Cookie configuration

The cookie configuration file, `cookie.php` - `/path/to/swiftriver-root/application/config/cookie.php`, specifies the following:

- The domain the cookie is available to
- Whether to only serve cookies over secure connections
- Magic salt to add to the cookie - IMPORTANT!!: As a security precaution, it is advisable that you change the default value
- No. of seconds before cookie expires; default is `0` meaning that the cookie expires when the browser is closed.

By default, `cookie.php` reads as follows:

```
return array(

    // Restrict the domain the cookie is available to
   'domain' => 'example.com',

   // Only transmit cookies over secure connections
   'secure' => FALSE,

   // Magic salt to add to the cookie
   // NOTE: This is the default value and MUST be changed on your deployment
   'salt' => 'WqLHtxZ3X4iGu%<CceGZwR3dAd?3Z4BW',

   // Number of seconds before the cookie expires
   'expiration' => 0
);
```

NOTE: If you are serving your SwiftRiver installation via localhost (e.g.http://localhost/swiftriver), the `domain` parameter should be blank otherwise the application's CSRF protection mechanism shall prevent you from logging in and performing subsequent operations.

Accessing your SwiftRiver Installation

Point your browser to the URL of your SwiftRiver installation. At the login prompt, use `admin` and `password` for the username and password

respectively.

IMPORTANT!! Change the default password after the initial login

If you get the following error:

> Strict Standards: require() [function.require]: It is not safe to rely on the system's timezone settings. Please use the date.timezone setting, the TZ environment variable or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier.

it's because you haven't specified the timezone in your PHP configuration file - `php.ini`. Simply open your `php.ini` file for editing and look for the line with the following:

> date.timezone = <something here>

This is the configuration directive for the timezone used by all date/time functions. By default, SwiftRiver uses UTC for all date/time functions. Therefore, this line should read:

> date.timezone = 'UTC'

Once set, restart Apache in order for the changes to take effect.