

# Installing the SwiftRiver stack on Ubuntu/Debian

The SwiftRiver engine has a few moving parts. This guide covers what you need to do to get a basic SwiftRiver installation going. Mix and match these instructions to match what you know you need.

## On this page

- Requirements
  - Handy guides
- Installing the API
  - Set up API war file in Tomcat
  - Setting up the database
  - Setting up Solr
  - Get it going!
- SwiftRiver UI Client
- Indexing / Crawling / Messaging services
- Installing SwiftRiver as a service
  - Logs to watch
  - Issues with RabbitMQ
  - Example swiftriver-api.xml
  - swiftriver-api.xml properties

## Recommended Platforms

- Ubuntu 10.04 TLS +
- Debian 6.0 (Squeeze)

This guide does not cover installation under Windows systems. If you want to do that, please use the [SwiftRiver windows instructions](#) and update them with anything you learn.

## Requirements

- Java JDK 1.5 or greater: preferably Java 1.6 (also known as Java 6)
- MySQL Server version 5.1 or greater
- Apache Solr
- Apache Tomcat (version 6.0 or greater) or other servlet container
- Apache HTTP server with mod\_rewrite enabled
- PHP version 5.3 or greater
- Python version 2.6 or version 2.7
- RabbitMQ version 3.0 or greater
- Sendmail or other Mail Transfer Agent
- Git
- Maven

## Handy guides

- [Install Solr 4.6 with Tomcat 7 on Debian 7](#): gives you Java, Tomcat, Solr (but don't secure the admin page as suggested). This guide works for Ubuntu too. Note that this script doesn't install the Java JDK required to install SwiftRiver.

## Notes

- These instructions assume that you're installing SwiftRiver on localhost. Please change the steps below if you're using a different hostname for your machine.
- We'll use /opt/swiftriver as the base directory to install all the SwiftRiver files. If you want to use a different base directory, make sure that you use the correct path consistently as you follow these instructions.
- These instructions are based on a [script written by Emmanuel Kala](#), which is a more compact version of this guide.
- Always check the `/var/log/tomcat7/catalina.out` log file during installation, to watch for any weird exceptions that might stall SwiftRiver's initialisation.
- Some of the paths used in this guide might differ from your environment: please adapt accordingly. We recommend adding these paths to

your bash, as a few services might benefit from that:

```
$TOMCAT_HOME # Tomcat directory (e.g. /etc/tomcat7/)
$JAVA_HOME # Java directory (e.g. /usr/lib/jvm/default-java)
```

## Installing the API

### Set up API war file in Tomcat

#### 1. Checkout API

```
# Create directories to hold code and war file
mkdir -p /opt/swiftriver/src
mkdir -p /opt/swiftriver/api
cd /opt/swiftriver/src

# Clone API code
git clone git://github.com/ushahidi/SwiftRiver-API.git
```

#### 2. Compile API and copy resources to the correct directories

```
# Go into code directory
cd /opt/swiftriver/src/SwiftRiver-API

# Package file
mvn clean package

# Move war file to api folder
cp target/swiftriver-api.war /opt/swiftriver/api

# Copy configuration files to correct api places
cp target/classes/config/swiftriver-api.xml $TOMCAT_HOME/Catalina/localhost
cp target/classes/indexer.properties /opt/swiftriver/api
chmod 666 /opt/swiftriver/api/indexer.properties
```

#### 3. Update the entries below in file swiftriver-api.xml. You shouldn't need to change any other values (More details are in the file below)

```

<!-- SET DOCBASE TO /opt/swifriver/api/swifriver-api.war -->
<Context docBase="/opt/swifriver/api/swifriver-api.war" path="/swifriver-api">

  <!-- UPDATE THE DATABASE INFO BELOW TO REFLECT THE CREDENTIALS CREATED ON THE SECTION
  BELOW (db: swifriver, user: swifriver, password:swifriver -->
    <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver"
      maxActive="8" maxIdle="4" name="jdbc/SwiftRiverDB" type="javax.sql.DataSource"
      url="jdbc:mysql://localhost/swifriver?zeroDateTimeBehavior=convertToNull"
      username="swifriver" password="swifriver" />
  ...
  <!-- UPDATE solr/serverURL TO http://localhost:8080/solr/swifriver -->
    <Environment name="solr/serverURL" type="java.lang.String" value="http://localhost:8080/solr/swifriver"/>

  <!-- UPDATE solr/indexerProperties TO /opt/swifriver/api/indexer.properties -->
    <Environment name="solr/indexerProperties" type="java.lang.String"
value="/opt/swifriver/api/indexer.properties" />
  ...

```

## Setting up the database

Log into your MySQL database and execute the following commands:

```

-- Creates `swifriver` database
CREATE DATABASE swifriver CHARACTER SET utf8 COLLATE utf8_unicode_ci;

-- Adds user with password 'swifriver'
GRANT ALL PRIVILEGES ON swifriver.* TO swifriver@'localhost' IDENTIFIED BY 'swifriver';

```

Then exit the MySQL prompt and import the mysql schema into the database:

```
mysql -uswifriver -p swifriver < /opt/swifriver/src/SwiftRiver-API/src/main/resources/config/sql/schema.sql
```

## Tomcat, Java and JDBC

If your application fails to start later on, it might be because it's not connecting to the MySQL database. To solve that, you'll need to make sure that the JDBC driver is installed on Tomcat.

```

# Install JDBC
sudo apt-get install libmysql-java

# Copy driver to Tomcat's lib (make sure to double check these paths make sense on your system!)
cp /usr/share/java/mysql.jar /usr/share/tomcat7/lib

```

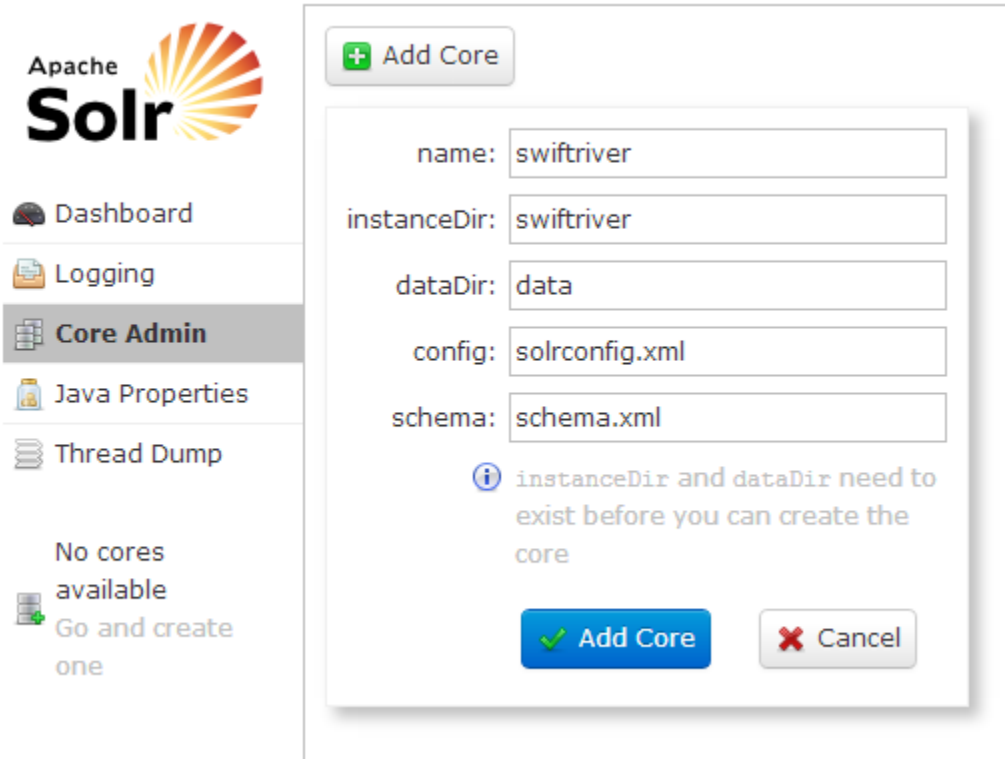
## Setting up Solr

To get Solr going, we need to move some files around:

```
# Copy configuration files to solr directory
cp -r /opt/swiftriver/src/SwiftRiver-API/solr/swiftriver/conf /var/lib/tomcat7/solr/swiftriver

# Make sure Tomcat can write on that dir
chown -R tomcat7:tomcat7 /var/lib/tomcat7/solr/swiftriver
```

Once that's done, navigate to <http://localhost:8080/solr>, Go to Core Admin > Add core and fill in the fields as shown below.



The screenshot shows the Apache Solr Core Admin interface. On the left is a navigation menu with options: Dashboard, Logging, Core Admin (selected), Java Properties, and Thread Dump. Below the menu, it says "No cores available" and "Go and create one". The main content area shows a "Add Core" dialog box with the following fields:

- name: swiftriver
- instanceDir: swiftriver
- dataDir: data
- config: solrconfig.xml
- schema: schema.xml

Below the fields, there is an information icon and the text: "instanceDir and dataDir need to exist before you can create the core". At the bottom of the dialog are two buttons: "Add Core" (with a green checkmark) and "Cancel" (with a red X).

## Get it going!

With all that in place, all you have to do is:

```
# Restart tomcat7
service tomcat7 restart

# Test it!
curl http://localhost:8080/swiftriver-api/

# Expected answer:
{"error":"unauthorized","error_description":"An Authentication object was not found in the SecurityContext"}
```

If that's what you got, do a little dance! You're half way there!

(If not, check your logs, and battle it kid!)

## SwiftRiver UI Client

As you can see from the [architecture](#), the UI client is separate from the API. If you're feeling Rambo, just jump straight into interacting with the API.

If not, here's what you have to do to get the UI going.

(Assuming that your DocumentRoot is on /var/www)

```
# Make sure you have all the libraries required by the front end
sudo apt-get install php5-curl php5-mcrypt php5-mysql php5-gd

# Go into your webroot
cd /var/www

# Set up git for directoy
git init
git remote add origin git://github.com/ushahidi/SwiftRiver.git
git fetch

# Checkout version 1.0 of code and grab all submodules
git checkout master
git submodule init && git submodule update

# Create necessary dirs
mkdir application/cache
chmod 777 application/cache
mkdir application/logs
chmod 777 application/logs

# Init all config files
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
```

Once that's done, update the following files:

```
application/config/auth.php

return array(
    'driver' => 'SwiftRiver',

    // OAuth Parameters
    'token_endpoint' => 'http://localhost:8080/swiftriver-api/oauth/token', // UPDATE THIS TO POINT TO YOUR API URL
    'client_id' => 'trusted-client',
    'client_secret' => 'somesecret',
    'grant_type' => 'password',

    // No authentication for these controllers
    'ignore_controllers' => array('login', 'error_handler', 'welcome')
);
```

```
application/config/site.php
```

```
...
```

```
/* Default maximum number of rivers a user can create */  
'default_river_quota' => 1,
```

```
/* Default maximum number of drops a river can hold */  
'default_river_drop_quota' => 10000,
```

```
/* Site url */  
'site_url' => 'http://localhost', // UPDATE THIS TO THE ADDRESS WHERE YOUR UI IS BEING DEPLOYED
```

```
// IF YOU ARE SETTING UP PROPER EMAIL SERVERS, UPDATE THE VALUES BELOW TO MAKE EVERYTHING NEAT  
/* Default domain for outgoing emails */  
'email_domain' => 'example.com',
```

```
/* Default domain for outgoing comment notification emails */  
'comments_email_domain' => 'example.com'
```

```
...
```

```
modules/SwiftRiver_API/config/swiftriver.php
```

```
return array(  
'base_url' => 'http://localhost:8080/swiftriver-api/v1', // UPDATE THIS TO POINT TO YOUR API URL  
);
```

You shouldn't need to change any other files.

Once that's done, do an `apache2ctl restart` for good measure and you should be good to go. Visit `http://localhost` and do another dance.

## mod\_rewrite

If you get a "Not Found" response when the system redirects you to `/login`, maybe `mod_rewrite` is getting in the way. If that's the case:

- Check that `mod_rewrite` is installed and enabled. (run `a2enmod rewrite` in case it isn't)
- Make sure you have `AllowOverride All` (as opposed to `None`) for your directory on the apache conf file (restart Apache after that!)

## Indexing / Crawling / Messaging services

If you've done everything above, you should be able to log into the UI, but nothing much else. Let's change that! There's quite a bit to be done, so get a cup of coffee and hold tight.

Please make sure the RabbitMQ server is installed before proceeding - you shouldn't need to change any configurations for it.

1. First, install all the requirements for the services below

### Services requirements

```
apt-get -y install python2.7 python-pip python-mysqldb python-imaging python-lxml python-httplib2
pip install tweepy
pip install pika
pip install feedparser
pip install python-cloudfiles
```

## 2. Check out the Swiftriver Core

```
# Create services directory
mkdir /opt/swiftriver/services/python

# Create logs directory
mkdir /opt/swiftriver/services/python/logs

# Get back to src directory
cd /opt/swiftriver/src

# Clone core
git clone git://github.com/ushahidi/SwiftRiver-Core.git

# Copy the rss, semanticsqueue and mediaextractor apps to the service directory
cp -rf SwiftRiver-Core/rss /opt/swiftriver/services/python
cp -rf SwiftRiver-Core/twitter /opt/swiftriver/services/python
cp -rf SwiftRiver-Core/semanticsqueue /opt/swiftriver/services/python
cp -rf SwiftRiver-Core/mediaextractor /opt/swiftriver/services/python
```

## 3. Configure services

## RSS Service

```
# Configure RSS
cd /opt/swiftriver/services/python/rss/config

# Create log dir
mkdir /opt/swiftriver/services/python/rss/logs

# Copy templates
cp rss_fetcher.cfg.template rss_fetcher.cfg
cp rss_scheduler.cfg.template rss_scheduler.cfg

# For both files
# Update "/path/to/log" to "/opt/swiftriver/services/python/rss/logs"
# On pid_file, update "/path/to" to "/opt/swiftriver/services/python/rss"
# Update DB details too
host=localhost
port=3306
user=swiftriver
pass=swiftriver
database=swiftriver

# Run SQL install
cd /opt/swiftriver/services/python/install
mysql -uswiftriver -p swiftriver < rss.sql
```

## Media Extractor service

```
# Create log dir
mkdir /opt/swiftriver/services/python/mediaextractor/logs

# Go to config directory
cd /opt/swiftriver/services/python/mediaextractor/config

# Copy template
cp mediaextractor.cfg.template mediaextractor.cfg

# Edit file and Update "/path/to/log" to "/opt/swiftriver/services/python/mediaextractor/logs"
# On pid_file, update "/path/to" to "/opt/swiftriver/services/python/mediaextractor"
```



### Semantics Queue service

```
# Create log dir
mkdir /opt/swiftriver/services/python/semanticsqueue/logs

# Go to config directory
cd /opt/swiftriver/services/python/semanticsqueue/config

# Copy template
cp semanticsqueue.cfg.template semanticsqueue.cfg

# Update "/path/to/log" to "/opt/swiftriver/services/python/semanticsqueue/logs"
# On pid_file, update "/path/to" to "/opt/swiftriver/services/python/semanticsqueue"
```

### Twitter service

```
# Create log dir
mkdir /opt/swiftriver/services/python/twitter/logs

# Create Twitter cache file
touch /var/cache/twitter.cache

# Go to config directory
cd /opt/swiftriver/services/python/twitter/config

# Copy templates
cp firehose.cfg.template firehose.cfg
cp manager.cfg.template manager.cfg

# For both files
# Update "/path/to/log" to "/opt/swiftriver/services/python/twitter/logs"
# On pid_file, update "/path/to" to "/opt/swiftriver/services/python/twitter"

# Update all Twitter credentials on firehose.cfg (you'll need to have an app registered on twitter for that)
consumer_key=TWITTER_API_KEY
consumer_secret=TWITTER_API_SECRET
token_key=TWITTER_ACCESS_TOKEN
token_secret=TWITTER_ACCESS_TOKEN_SECRET

# FOR manager.cfg
# Update Twitter cache file
cache_file=/var/cache/twitter.cache

# Update database credentials
host=localhost
port=3306
user=swiftriver
pass=swiftriver
database=swiftriver
```

4. Copy lib folder to services folder

```
# Back to source directory
cd /opt/swiftriver/src/

# Copy lib folder
cp -rf SwiftRiver-Core/lib /opt/swiftriver/services/python
```

5. Build and install the Swiftriver API Client

#### API client config

```
# Go to source directory
cd /opt/swiftriver/src/

# Clone repo
git clone git://github.com/ushahidi/swiftriver-api-java.git

# Build it
cd /opt/swiftriver/src/swiftriver-api-java
mvn clean install
```

6. Install the dropqueue processor

## Dropqueue config

```
# Source dir
cd /opt/swiftriver/src

# Clone repo
git clone git://github.com/ushahidi/swiftriver-core-dropqueue-processor.git dropqueue-processor

# Before building, it's worth noting that the dropqueue-processor/pom.xml file asks for a huge amount of
memory for the war file.
# If you have less than 4GB of RAM on your machine, make sure to edit the pom.xml file accordingly or else
everything will explode.
# You need to change the "jvmSettings > initialMemorySize" and "jvmSettings > maxMemorySize".
# I would recommend setting half the system memory to maxMemorySize, and half of that to initialMemorySize.
# Also note that this might play on performance, so be smart about your system hardware requirements!

# Build
cd dropqueue-processor
mvn clean package

# Move build and config files to services folder
cp -rf target/generated-resources/appassembler/jsr/dropqueue-processor /opt/swiftriver/services
cp config/* /opt/swiftriver/services/dropqueue-processor/conf

# Create wrapper conf file and make wrappers executable
cd /opt/swiftriver/services
perl -p -i -e 's/-Dext\.prop\.dir/-Dext\.prop\.dir=Vopt\swiftriver\services\dropqueue-processor\conf/g'
dropqueue-processor/conf/wrapper.conf
chmod +x dropqueue-processor/bin/dropqueue-processor
chmod +x dropqueue-processor/bin/wrapper-linux*
```

### 7. Install rules processor

## Rules processor

```
# Source dir
cd /opt/swiftriver/src

# Clone repo
git clone git://github.com/ushahidi/swiftriver-core-rules-processor.git rules-processor

# Before building, it's worth noting that the rules-processor/pom.xml file asks for a huge amount of memory for
the war file.
# If you have less than 4GB of RAM on your machine, make sure to edit the pom.xml file accordingly or else
everything will explode.
# You need to change the "jvmSettings > initialMemorySize" and "jvmSettings > maxMemorySize".
# I would recommend setting half the system memory to maxMemorySize, and half of that to initialMemorySize.
# Also note that this might play on performance, so be smart about your system hardware requirements!

# Build
cd rules-processor
mvn clean package

# Move build and config files to services folder
cp -rf target/generated-resources/appassembler/jsr/rules-processor /opt/swiftriver/services
cp config/* /opt/swiftriver/services/rules-processor/conf

# Update the rules /opt/swiftriver/services/rules-processor/conf/rules-processor.properties file with
db.driverClassName=com.mysql.jdbc.Driver
db.url=jdbc:mysql://localhost/swiftriver?zeroDateTimeBehavior=convertToNull
db.username=swiftriver
db.password=swiftriver

# Create wrapper conf file and make wrappers executable
cd /opt/swiftriver/services
perl -p -i -e 's/-Dext\.prop\.dir/-Dext\.prop\.dir=Vopt\swiftriver\services\rules-processor\conf/g'
rules-processor/conf/wrapper.conf
chmod +x rules-processor/bin/rules-processor
chmod +x rules-processor/bin/wrapper-linux*
```

8. This is everything to configure for now. Well done for making it to here! You are a champion!

## Installing Swiftriver as a service

Now that we configured everything, all you have to do is copy and paste the lines below on your command line so that Swiftriver runs as a service

### Creating Swiftriver service

```
# Run the remaining ops from the home directory
cd ~/

# Create the swiftriver script
cat <<EOF > swiftriver
#!/bin/bash
### BEGIN INIT INFO
# Provides: SwiftRiver service stack bootstrap
# Required-Start:
```

```
# Required-Stop:
# Default-Start:
# Default-Stop:
# Short-Description: Automatically start the background crawlers for SwiftRiver
# Description
### END INIT INFO

# Set the PYTHONPATH
export PYTHONPATH=$PYTHONPATH:/opt/swiftriver/services/python/lib

# cd into the directory with the services dir
cd /opt/swiftriver/services

start_services() {
# RSS Scheduler and Fetcher
python python/rss/rss_scheduler.py start
python python/rss/rss_fetcher.py start

# Semantics queue and media extraction
python python/semanticsqueue/semanticsqueue.py start
python python/mediaextractor/mediaextractor.py start

# Twitter
python python/twitter/manager.py start
python python/twitter/firehose.py start

# DropQueue processor and rules processor
dropqueue-processor/bin/dropqueue-processor start
rules-processor/bin/rules-processor start
}

# Stops the content services
stop_services() {
# Twitter
python python/twitter/firehose.py stop
python python/twitter/manager.py stop

# RSS
python python/rss/rss_fetcher.py stop
python python/rss/rss_scheduler.py stop

# Queues
python python/semanticsqueue/semanticsqueue.py stop
python python/mediaextractor/mediaextractor.py stop

# DropQueue and rules processor
dropqueue-processor/bin/dropqueue-processor stop
rules-processor/bin/rules-processor stop
}

case "$1" in
start)
echo "Starting SwiftRiver content services"
start_services
;;
stop)
echo "Stopping SwiftRiver content services"
stop_services
;;

```

```
restart)
echo "Restarting SwiftRiver content services"
stop_services
start_services
;;
*)
echo "Usage: /etc/init.d/swiftriver (start|stop|restart)"
exit 1
;;
esac
EOF

cp swiftriver /etc/init.d
chmod a+x /etc/init.d/swiftriver
```

```
# Add the swiftriver service
update-rc.d swiftriver defaults 95 05
```

Now, all you have to do is run `service swiftriver start`, and enjoy your success!

## More info

### Logs to watch

If during installation, anything doesn't behave as expected, here are the logs you should check (double points if you paste log dumps into bug reports!):

- Apache logs (/var/logs/apache2, mainly error.log)
- Tomcat logs (/var/logs/tomcat7, mainly catalina.out)
- Application logs (i.e.: /var/www/application/logs)
- MySQL logs - enable the general log if you want to really get to the thick of what's going on
- Service logs - each service will have logs being written on their respective logs directory. Keep an eye on those!

We can't stress enough how helpful these are. 😊

### Issues with RabbitMQ

If you run `rabbitmqctl status` and you see something like `Error: unable to connect to node 'rabbit@swiftriver-dev': nodedown` as a response, RabbitMQ is having problems starting.

Run `sudo service rabbitmq-server restart` and RabbitMQ should be good again.

### Example swiftriver-api.xml

```

<Context docBase="/opt/swiftriver/api/swiftriver-api.war" path="/swiftriver-api" >
  <!-- SwiftRiver Database configuration -->
  <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver"
    maxActive="8" maxIdle="4"
    name="jdbc/SwiftRiverDB"
    type="javax.sql.DataSource"
    url="jdbc:mysql://localhost/swiftriver?zeroDateTimeBehavior=convertToNull"
    username="swiftriver"
    password="swiftriver"/>

  <!-- Encryption Key -->
  <Environment name="encryptionKey" type="java.lang.String" value="2344228477#97{7&amp;6&gt;82}"/>

  <!-- MQ Properties -->
  <Environment name="mqHost" type="java.lang.String" value="localhost"/>
  <Environment name="mqUser" type="java.lang.String" value="guest"/>
  <Environment name="mqPass" type="java.lang.String" value="guest"/>

  <!-- HTTP Solr Server -->
  <Environment name="solr/serverURL" type="java.lang.String" value="http://localhost:8080/solr/swiftriver"/>

  <!-- Location of Solr indexing properties file -->
  <Environment name="solr/indexerProperties" type="java.lang.String" value="/opt/swiftriver/api/indexer.properties"
/>

  <!-- Keys for the indexer properties file -->
  <Environment name="indexer/lastDropIdPropKey" type="java.lang.String" value="indexer.lastDropId" />
  <Environment name="indexer/batchSizePropKey" type="java.lang.String" value="indexer.batchSize" />
  <Environment name="indexer/runInterval" type="java.lang.String" value="30000"/>

  <!-- Default authentication scheme. Possible values are:
    database
    crowdmapid

    'database' is the default
  -->
  <Environment name="authSchemeName" type="java.lang.String" value="database"/>

  <!-- CrowdmapID API URL e.g. https://example.com/ -->
  <Environment name="crowdmapid/serverURL" type="java.lang.String" value="https://crowdmapid.com/api"/>
  <Environment name="crowdmapid/apiKey" type="java.lang.String" value="" />
  <Environment name="crowdmapid/apiKeyParamName" type="java.lang.String" value="api_secret"/>

  <!-- Mail configuration -->
  <Environment name="mail/host" type="java.lang.String" value="localhost" />
  <Environment name="mail/senderAddress" type="java.lang.String" value="no-reply@swiftriver.dev"/>
  <Environment name="mail/resetPasswordUrl" type="java.lang.String"
value="http://swiftriver.dev/login/reset_password"/>
  <Environment name="mail/activateAccountUrl" type="java.lang.String" value="http://swiftriver.dev/login/activate"/>

</Context>

```

## swiftriver-api.xml properties

Parameter	Description
-----------	-------------



mqHost	The host running the RabbitMQ server
mqUser	User to connect to RabbitMQ
mqPassword	Password for the user used to connect to RabbitMQ
solr/serverURL	URL of your Solr server
solr/indexerProperties	Location of the properties file for the indexer - a background process that periodically updates Solr with the new drops
indexer/lastDropIDPropKey	The property key that specifies the ID of the last drop to be posted to Solr. This value serves as the reference point for fetching new drops
indexer/batchSizePropKey	The property key that specifies the maximum number of drops to post to Solr during each run
indexer/runInterval	The property key that specifies how often (in milliseconds) the indexer should check for new drops and update Solr
authSchemeName	Name of the authentication scheme. The possible values are <code>database</code> and <code>crowdmapid</code>
crowdmapid/serverURL	URL of the CrowdmapID deployment