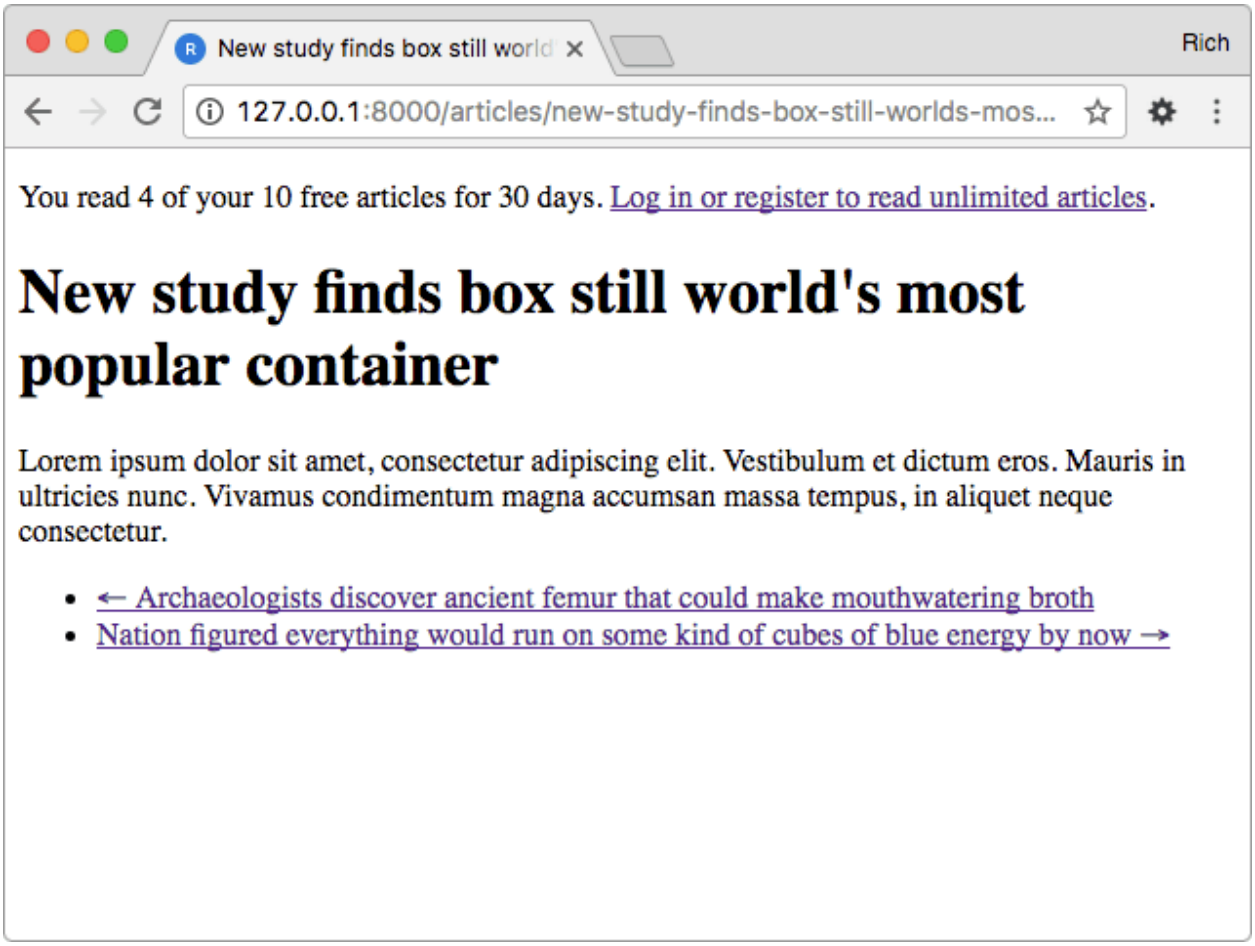
django-registrationwall Documentation

Release 0.1

Richard Cornish

Aug 30, 2017

Contents



Django Registration Wall is a Django mixin application that limits an anonymous user's access to content, after which the user is redirected to the login URL. The behavior is modeled after the common paywall scenario.

Fake news articles credit goes to The Onion.

- Package distribution
- Code repository
- Documentation
- Tests

CHAPTER 1

Install

```
$ pip install django-registrationwall
```

Add to `settings.py`.

```
INSTALLED_APPS = [
    # ...
    'regwall',
]
```

Add to one of your `views.py`.

```
from django.views.generic import DetailView

from regwall.mixins import RaiseRegWallMixin

from .models import Article


class ArticleDetailView(RaiseRegWallMixin, DetailView):
    model = Article
```

CHAPTER 2

Contents

Install

Install with the pip package manager.

```
$ mkvirtualenv myvenv -p python3
$ pip install django
$ pip install django-registrationwall
```

After creating a project, add `regwall` to `INSTALLED_APPS` in `settings.py`.

```
INSTALLED_APPS = [
    # ...
    'regwall',
]
```

Remember to update your `requirements.txt` file. In your project directory:

```
$ pip freeze > requirements.txt
```

Usage

Views

The app is almost entirely a mixin that subclasses Django's `AccessMixin`. Import the mixin and subclass it in a `DetailView` or a view that uses `SingleObjectMixin`.

```
from django.views.generic import DetailView

from regwall.mixins import RaiseRegWallMixin

```

```
from .models import Article


class ArticleDetailView(RaiseRegWallMixin, DetailView):
    model = Article
```

On each request, the mixin increments the number of consumed resources and checks its count against the `REGWALL_LIMIT` setting. Resources contain information about each view's main object, which means the mixin expects to be added to views that focus on a single object such as `DetailView`, although technically any view that incorporates `SingleObjectMixin` is valid.

The app stores the visited resources into the browser session, whose session ID is stored in a cookie in the user's web browser. The app does not employ more sophisticated user tracking such as IP detection and storage.

Template tags

The bulk of the app's logic is in the mixin, but template tags allow for display of the list of resources consumed, the limit, and the days of expiration.

Load the template tags.

```
{% load regwall_tags %}
```

Assign any of the tags' output to a variable with the `as` syntax.

```
{% load regwall_tags %}

{% get_regwall_limit as regwall_limit %}
{% get_regwall_expire as regwall_expire %}
{% get_regwall_attempts as regwall_attempts %}
{% get_regwall_successes as regwall_successes %}
```

Use the assigned variables as you like.

{% get_regwall_limit %}

Gets the number of resources a user can consume before the registration wall is raised.

```
{% load regwall_tags %}

{% get_regwall_limit as regwall_limit %}

<p>The limit is {{ regwall_limit }} articles.</p>
```

{% get_regwall_expire %}

Gets the number of days until the consumed resources count is reset to zero.

```
{% load regwall_tags %}

{% get_regwall_limit as regwall_limit %}
{% get_regwall_expire as regwall_expire %}

<p>The limit is {{ regwall_limit }} articles for {{ regwall_expire }} days.</p>
```

{% get_regwall_attempts %}

Gets a list of the attempted consumed resources. The mixin logs each attempt a user makes for a request, which is not necessarily the same as a successful request. Each item of the list is a dictionary, which contains the `app_label`, `id`, `headline`, and `url` of a single resource. You will probably use the `length` filter on `get_regwall_attempts` to get the number of attempted consumed resources.

```
{% get_regwall_attempts as regwall_attempts %}

<p>You tried to read {{ regwall_attempts|length }} free articles.</p>
```

Use `get_regwall_attempts` to check against the result of `get_regwall_limit`.

```
{% get_regwall_limit as regwall_limit %}
{% get_regwall_expire as regwall_expire %}
{% get_regwall_attempts as regwall_attempts %}

{% if regwall_attempts|length >= regwall_limit %}
<p>You read all of your {{ regwall_limit }} articles for {{ regwall_expire }} days.</
↪p>
{% endif %}
```

{% get_regwall_successes %}

Similar to `get_regwall_attempts`, but `get_regwall_successes` gets a list of the resources that were successful delivered to the user.

```
{% load regwall_tags %}

{% get_regwall_limit as regwall_limit %}
{% get_regwall_expire as regwall_expire %}
{% get_regwall_attempts as regwall_attempts %}
{% get_regwall_successes as regwall_successes %}

{% if regwall_attempts|length >= regwall_limit %}
<p>You read all {{ regwall_successes|length }} of your {{ regwall_limit }} articles␣
↪for {{ regwall_expire }} days.</p>
<ol>
    {% for article in regwall_successes %}
    <li><a href="{{ article.url }}">{{ article.headline }}</a></li>
    {% endfor %}
</ol>
{% endif %}
```

Note that because different models can use different conventions for what constitutes a "headline," the template tag checks against these model attributes in this order: `headline`, `title`, `name`, and finally empty string.

Includes

To ease the creation of probable messages displayed to users, use (or be inspired by) the app's template includes in the `regwall` directory.

regwall/detail.html

Usage in a template, intended for a "detail" template whose view probably uses a `DetailView` of your own creation:

```
{% include 'regwall/detail.html' %}
```

The result:

```
{% load regwall_tags %}

{% get_regwall_attempts as regwall_attempts %}
{% get_regwall_successes as regwall_successes %}
{% get_regwall_limit as regwall_limit %}
{% get_regwall_expire as regwall_expire %}

{% if regwall_successes|length > 0 %}
<p>You read {{ regwall_successes|length }} of your {{ regwall_limit }} free article{{␣
↪regwall_limit|pluralize }} for {{ regwall_expire }} day{{ regwall_expire|pluralize }
↪}. <a href="{% url 'login' %}">Log in or register to read unlimited articles</a>.</
↪p>
{% endif %}
```

regwall/login.html

Usage in a template, intended for `registration/login.html`:

```
{% include 'regwall/login.html' %}
```

The result:

```
{% load regwall_tags %}

{% get_regwall_attempts as regwall_attempts %}
{% get_regwall_successes as regwall_successes %}
{% get_regwall_limit as regwall_limit %}
{% get_regwall_expire as regwall_expire %}

{% if regwall_attempts|length >= regwall_limit %}
<p>You read {{ regwall_successes|length }} of your {{ regwall_limit }} free article{{␣
↪regwall_limit|pluralize }} for {{ regwall_expire }} day{{ regwall_expire|pluralize }
↪}. Log in or register to read unlimited articles.</p>
{% endif %}
```

regwall/history.html

Usage in a template, intended for `registration/login.html`:

```
{% include 'regwall/history.html' %}
```

The result:

```
{% load i18n regwall_tags %}

{% get_regwall_attempts as regwall_attempts %}
{% get_regwall_successes as regwall_successes %}
```

```
{% get_regwall_limit as regwall_limit %}

{% if regwall_attempts|length >= regwall_limit %}
<h2>{% trans 'You read these articles' %}</h2>
<ol>
    {% for article in regwall_successes %}
    <li><a href="{{ article.url }}">{{ article.headline }}</a></li>
    {% endfor %}
</ol>
{% endif %}
```

Demo

The repo contains a sample Django project that shows how a typical intergration might occur with the template tags and includes. A fixture with sample data is also included to quickly test.

```
$ mkvirtualenv -p python3 demo
(demo)$ git clone git@github.com:richardcornish/django-registrationwall.git
(demo)$ cd django-registrationwall/demo/
(demo)$ pip install -r requirements.txt
(demo)$ cd demo/
(demo)$ python manage.py migrate
(demo)$ python manage.py loaddata articles_article.json
(demo)$ python manage.py runserver
```

Open http://127.0.0.1:8000/articles/.

Settings

The mixin tag offers three settings. By default, they are:

```
REGWALL_LIMIT = 10

REGWALL_EXPIRE = 30

REGWALL_SOCIAL = [
    'google',
    'facebook',
    'twitter',
]
```

REGWALL_LIMIT

An integer indicating the number of resources to display before the registration wall appears.

The mixin displays `10` resources by default.

REGWALL_EXPIRE

An integer indicating the number of days before the consumed resources count is reset to zero.

The mixin resets after `30` days by default.

REGWALL_SOCIAL

A list of strings of domains whose referral does not increment the consumed resources count. In other words, visitors coming from these domains are not penalized. Previously, the app used a rudimentary method of domain checking with the `urlparse`/`urllib.parse` modules. Because URLs vary so widely in construction, the app now uses the tldextract package to accurately extract the domain. Therefore, this setting should contain only domains and not top-level domains, e.g. `['google', 'facebook', 'twitter']` and *not* `['google.com', 'facebook.com', 'twitter.com']`.

The mixin allows referrals from Google, Facebook, and Twitter by default.

Documentation

Full documentation is available online.

However, you can also build the documentation from source. Enter your virtual environment.

```
$ workon myvenv
```

Clone the code repository.

```
$ git clone git@github.com:richardcornish/django-registrationwall.git
$ cd django-registrationwall/
```

Install Sphinx, `sphinx-autobuild`, and `sphinx_rtd_theme`.

```
$ pip install sphinx sphinx-autobuild sphinx_rtd_theme
```

Create an HTML build.

```
$ (cd docs/ && make html)
```

Or use `sphinx-autobuild` to watch for live changes.

```
$ sphinx-autobuild docs/ docs/_build_html
```

Open 127.0.0.1:8000.

Tests

Continuous integration test results are available online.

However, you can also test the source code.

```
$ workon myvenv
$ django-admin test regwall.tests --settings="regwall.tests.settings"

Creating test database for alias 'default'...
..
----------------------------------------------------------------------
Ran 2 tests in 0.229s

OK
Destroying test database for alias 'default'...
```

A bundled settings file allows you to test the code without even creating a Django project.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

R