Django Global Login Required Middleware Documentation

Release 1.1.2

M.A. Heshmatkhah

Mar 25, 2020

Contents:

Django Global Login Required Middleware (django-glrm) is a Django middleware that make all views and URLs login required.

It's common in Django that most of the site's pages are protected, with just a few exceptions of pages that remain public (e.g. login page, etc.). It can be quite tedious to decorate all of the views with `@login_required`, and it can be easy to forget to decorate some of them.

So, you can use **Django Global Login Required Middleware** to make all page login required excep some of them.

CHAPTER 1

Django Global Login Required Middleware

This module is a Django middleware that make all views and URLs login required.

Contents

1.1 Documentation

1.1.1 Installation

you can install Django Global Login Required Middleware using `pip`:

```
$ pip install django-glrm
```

1.1.2 Usage

To install this app, you should add `'global_login_required.LoginRequiredMiddleware'` to `settings.MIDDLEWARE`

```
MIDDLEWARE = [
    # default contents
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    ...

    'global_login_required.GlobalLoginRequiredMiddleware',

    ...
]
```

then all routes in your sile will be login required.

there is 4 ways to exclude a url or view from being login required:

- Add a `@login_not_required` decorator for view (**function based** or **class based**)
- List the public view (not login required views) in settings.py at *PUBLIC_VIEWS*
- List the public url's regex is settings.py at *PUBLIC_PATHS*
- Add `LOGIN_NOT_REQUIRED` property to view class

1.1.3 Decorator

if you want to use `login_not_required` decorator for a **class based view**, it should be in one of this formats:

1. Use as a normal decorator for class

```
from global_login_required import login_not_required
from django.views.generic import ListView

@login_not_required
class test_ClassBasedView_decorator(ListView):
    ...
```

2. Decorating in URLconf:

```
from global_login_required import login_not_required

urlpatterns = [
...
    path(r'^cbv_decorator/', login_not_required(test_ClassBasedView_decorator.as_
↪view())),
...
]
```

3. Decorating the class:

```
from global_login_required import login_not_required
from django.utils.decorators import method_decorator
```

(continues on next page)

(continued from previous page)

```
from django.views.generic import ListView

@method_decorator(login_not_required, name='dispatch')
class test_ClassBasedView_method_decorator(ListView):
    ...
```

Danger: If you combine `login_not_required` decorator with a `login_required` decorator, your view will be login required.

1.1.4 Class Property

also you can a `LOGIN_NOT_REQUIRED` to your class based views and your class will be publicly available:

```
from django.views.generic import ListView

class test_ClassBasedView_property_public(ListView):
    LOGIN_NOT_REQUIRED = True # Makes the view publicly available

    def get(self, request, *args, **kwargs):
        return HttpResponse("Response from view.")
```

If you set `LOGIN_NOT_REQUIRED` to `False` your view still login required:

```
from django.views.generic import ListView

class test_ClassBasedView_property(ListView):
    LOGIN_NOT_REQUIRED = False # The view still login required

    def get(self, request, *args, **kwargs):
        return HttpResponse("Response from view.")
```

1.1.5 Settings

There is 2 settings available

PUBLIC_VIEWS

This setting is a **python list** that contains string path to any view that you want to make it publicly available:

```
PUBLIC_VIEWS = [
    'django.contrib.auth.views.login',
    'myapp.views.the_view',
]
```

The middleware will check every request and if responsible view of the request was listed at this setting, it will ignore checking for authentication.

Note: The view listed here can be **function based** or **class based**.

PUBLIC_PATHS

This setting is a **python list** that contains regex strings of URIs that you to make them publicly available:

```
PUBLIC_PATHS = [
    '^%s.*' % MEDIA_URL, # allow public access to any media on your application
    r'^/accounts/.*', # allow public access to all django-allauth views
]
```

the `r` letter before the regular expression is **optional** and tells python that this is a regex not a normal python string, but python `re` package can handel this itself.

also you can list exact URL in here.

The middleware will check every request and if URI of the request match with any of listed regular expressions, it will ignore checking for authentication.

Warning: It's important to handel authentication of urls that are private but match with some of listed patterns.

For example user profile page (`/accounts/profile/`) in above example should be login required:

- You can use `login_required` decorator for such views.
- You can write more complex regex that ensures correct access rights.

Note: If you manually add a `login_required` decorator to view, and then list that view in settings, the final final result will be **login required**.

CHAPTER 2

Code Description

class `global_login_required.`**`GlobalLoginRequiredMiddleware`**(*get_response=None*)

Bases: `django.utils.deprecation.MiddlewareMixin`

Main idea from Julien Phalip

Note: Compatible with django 1.11+

`check_decorator`(*view*)

Checks is the view function decorated or not

Parameters **`view`** – view function

Return type bool

`get_view`(*view_path*)

returns callable instance of view from view path :param view_path: view path, like *'django.contrib.auth.views.login'* :return: a callable instance of view, in this example *login*

`matches_public_path`(*path*)

Checks is the request path is listed in *PUBLIC_PATHS*, so it is not login required.

Parameters **`path`** (*`string`*) – the request path

Returns true if view was listed in *PUBLIC_PATHS*.

Return type bool

`matches_public_view`(*view*)

Checks is the view is listed in *PUBLIC_VIEWS*, so it is not login required.

Parameters **`view`** (*`callable`*) – the view

Returns true if view was listed in *PUBLIC_VIEWS*.

Return type bool

process_view(*request*, *view_func*, *view_args*, *view_kwargs*)
checks if this view is not in excluded views (by *PUBLIC_VIEWS*, *PUBLIC_PATHS* or *login_not_required* decorator), adds *login_required* decorator for view, otherwise run the view it self.

Note: read the documentation at djangoproject

Parameters

- **request** (*HttpRequest*) – this will be passed to next process_view and finally will be passed to view.
- **view_func** (*callable*) – the view function
- **view_args** – the arguments that will be passed to view
- **view_kwargs** – the keyword arguments that will be passed to view

Returns *None* in order to continue processing **process_view** in middleware chain or ‘ HttpResponse‘ for braking middleware chain.

`global_login_required.`**login_not_required**(*the_view*)
Decorator which marks the given view as public (not login required).

This decorator adds a read only property ‘‘ LOGIN_NOT_REQUIRED‘‘ to the view, and if it’s value be `True`, the middleware will not force for login.

If you combine this with a `login_required` decorator, your view will be login required.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index