# django-single-session

***Release 0.2***

**Willem Van Onsem**

**Feb 24, 2023**

# USER GUIDE

django-single-session is a Django library for ensuring that a user can be logged in to only one session at a time.

**Features:**

- ensure that a user is logged in at at most one session/browser/device
- ensure that a user logs out from all sessions if they log out
- two actions for the *ModelAdmin* to log out (admin) users

CHAPTER

ONE

# INSTALLATION

The package can be fetched as *django-single-session*, so for example with *pip* with:

```
pip3 install django-single-session
```

One can install the app by adding the *single_session* app to the *INSTALLED_APPS* setting:

```
# settings.py

# ...

INSTALLED_APPS = [
    # ...,
    'django.contrib.sessions',
    # ...,
    'single_session'
    # ...
]

MIDDLEWARE = [
    # ...,
    'django.contrib.sessions.middleware.SessionMiddleware',
    # ...,
    'django.contrib.auth.middleware.AuthenticationMiddlware',
    # ...
]
```

In order to work properly, the *SessionMiddleware* and *AuthenticationMiddleware* is be necessary, or another middleware class that will add a *.session* and *.user* attribute on the request object and will trigger the *user_logged_in* and *user_logged_out* signals with the proper session and user.

You also need to be using the database session backend

and running *migrate* to migrate the database properly:

```
python3 manage.py migrate single_session
```

CHAPTER

TWO

# GETTING STARTED

Once the package is installed, this will by default enforce that a user will only have *one* logged in session. This will *not* proactively logout existing sessions: only if the user logs in with another browser or device, the old session(s) will be closed.

You can disable the single session behavior by specifying the *SINGLE_USER_SESSION* setting in *settings.py* and thus setting this value to *False* (or any other value with truthiness *False*).

You can customise this behaviour by making the *SINGLE_USER_SESSION* setting be a string representing the name of a function which takes a user object as an argument. If this function returns *True* then the user will be logged out. If it returns *False* then the user will not be logged out.

The tool will also clean up *all* sessions of a user in case that user logs out. This thus means that if a user logs out on one browser/device, they will log out on all other browsers/devices as well. This functionality is still enabled if *SINGLE_USER_SESSION* is set to *False*. You can disable this by setting the *LOGOUT_ALL_SESSIONS* setting in *settings.py* to *False* (or any other value with truthiness *False*).

CHAPTER

THREE

# ADMIN ACTIONS

If one uses the admin site(s), and there is a *ModelAdmin* for the user model, then the package adds two extra actions to that admin. These actions can log out (normal) users, and all (including admin) users.

For users to work with these actions, they should be a super user (administrator), or have the *single_session.logout* and *single_session.logout_all* permissions respectively. We strongly advise *not* to give a user the *single_session.logout_all* permission, since that would mean that that user can log out administrator users, and by keeping these logged out, thus prevent administrators to do their job properly.

CHAPTER

FOUR

# MODELS

class single_session.models.**UserSession**(**args*, ***kwargs*)

A model used to store the relation between the session ids and the user model. This is used to determine efficiently what session(s) belong to what user.

The model also defines two extra permissions that can be used to log out all users, and all users except the admin users.

**exception DoesNotExist**

**exception MultipleObjectsReturned**

CHAPTER

FIVE

# SIGNALS

single_session.signals.**change_settings**(*sender*, *setting*, *value*, *enter*, ***kwargs*)

A signal handler that is attached to the setting_changed handler that will subscribe and unsubscribe the signal handlers to the proper signals.

single_session.signals.**remove_all_sessions**(*sender*, *user*, *request*, ***kwargs*)

A signal handler attached to the user_logged_out signal that will remove all sessions associated to that user. This will run if the LOGOUT_ALL_SESSIONS setting is enabled.

single_session.signals.**remove_other_sessions**(*sender*, *user*, *request*, ***kwargs*)

A signal handler attached to the user_logged_in signal that will create a UserSession that associates the session with the user that has logged in. If the SINGLE_USER_SESSION setting is enabled (by default), it will remove all the old sessions associated to that user.

CHAPTER

SIX

# ADMIN

single_session.admin.**has_single_session_logout_all_permission**(*request*, *instance=None*)

Checks if the user has a single_session.logout_all permission.

single_session.admin.**has_single_session_logout_permission**(*request*, *instance=None*)

Checks if the user has a single_session.logout permission.

single_session.admin.**logout_all_users_on_all_sessions**(*modeladmin*, *request*, *queryset*)

An action to log out all the users including admin users. This requires a single_session.logout_all permission.

single_session.admin.**logout_user_on_all_sessions**(*modeladmin*, *request*, *queryset*)

An action to log out all the users that are not admin users. This requires a single_session.logout permission.

CHAPTER

# SEVEN

# APPS

**class** single_session.apps.**SingleSessionConfig**(*app_name*, *app_module*)

The app config for the single_session app.

**ready**()

This ready() method will load the signals that will be triggered if a user has logged in or logged out, and will populate the *ModelAdmin* for the user model, given there is such ModelAdmin.

# PYTHON MODULE INDEX

## S

# INDEX