edX Django REST Framework Extensions Documentation

Release 7.0.1

edX

Aug 11, 2021

Contents

This package provides extensions for Django REST Framework that are useful for developing on the edX platform.

CHAPTER 1

Requirements

- Python (2.7, 3.5, 3.6)
- Django (1.8, 1.9, 1.10, 1.11)
- Django REST Framework (3.2+)

CHAPTER 2

Installation

Install using `pip`:

```
$ pip install edx-drf-extensions
```

CHAPTER 3

Table of Contents

3.1 Settings

All settings for this package reside in a dict, *EDX_DRF_EXTENSIONS*. Within this dict, the following keys should be specified, depending on the functionality you are using.

3.1.1 BearerAuthentication

These settings are used by the `BearerAuthentication` class.

OAUTH2_USER_INFO_URL

Default: `None`

URL of an endpoint on the OAuth2 provider where `BearerAuthentication` can retrieve details about the user associated with the provided access token. This endpoint should return a JSON object with user details and `HTTP 200` if, and only if, the access token is valid. See `BearerAuthentication.process_user_info_response()` for an example of the expected data format.

3.1.2 JwtAuthentication

These settings are used by the `JwtAuthentication` class. Since this class is based on `JSONWebTokenAuthentication`, most of its settings can be found in the documentation for `rest_framework_jwt` at http://getblimp.github.io/django-rest-framework-jwt/#additional-settings.

JWT_AUTH['JWT_VERIFY_AUDIENCE']

Default: `True`

If you do *not* want to verify the JWT audience, set the `'JWT_VERIFY_AUDIENCE'` key in the `JWT_AUTH` setting to `False`.

JWT_PAYLOAD_USER_ATTRIBUTES

Default: `('email',)`

The list of user attributes in the JWT payload that `JwtAuthentication` will use to update the local `User` model. These payload attributes should exactly match the names the attributes on the local `User` model.

3.2 Authentication

Authentication classes are used to associate a request with a user. Unless otherwise noted, all of the classes below adhere to the Django REST Framework's API for authentication classes.

3.3 Middleware

This module contains middleware to ensure best practices of DRF and other endpoints..

class RequestCustomAttributesMiddleware(*get_response=None*)

Adds various request related custom attributes.

Possible custom attributes include:

request_authenticated_user_set_in_middleware: Example values: 'process_request', 'process_view', 'process_response', or 'process_exception'. Attribute won't exist if user is not authenticated.

request_auth_type_guess: Example values include: no-user, unauthenticated, jwt, bearer, other-token-type, jwt-cookie, or session-or-other Note: These are just guesses because if a token was expired, for example,

the user could have been authenticated by some other means.

request_client_name: The client name from edx-rest-api-client calls. request_referer request_user_agent: The user agent string from the request header. request_user_id: The user id of the request user. request_is_staff_or_superuser: *staff* or *superuser* depending on whether the

user in the request is a django staff or superuser.

This middleware is dependent on the RequestCacheMiddleware. You must include this middleware later. For example:

```
MIDDLEWARE = (
    'edx_django_utils.cache.middleware.RequestCacheMiddleware',
    'edx_rest_framework_extensions.middleware.RequestCustomAttributesMiddleware',
)
```

This middleware should also appear after any authentication middleware.

process_exception(*request*, *exception*)

Django middleware handler to process an exception

process_request(*request*)

Caches if authenticated user was found.

process_response(*request*, *response*)
Add custom attributes for various details of the request.

process_view(*request*, *view_func*, *view_args*, *view_kwargs*)
Caches if authenticated user was found.

class RequestMetricsMiddleware(**args*, ***kwargs*)
Deprecated class for handling middleware. Class has been renamed to RequestCustomAttributesMiddleware.

3.4 Permissions

Permissions determine whether a request should be granted or denied access. Unless otherwise noted, all of the classes below adhere to the Django REST Framework's API for permission classes.

class IsStaff
Allows access to "global" staff users..

has_permission(*request*, *view*)
Return *True* if permission is granted, *False* otherwise.

class IsSuperuser
Allows access only to superusers.

has_permission(*request*, *view*)
Return *True* if permission is granted, *False* otherwise.

class IsUserInUrl
Allows access if the requesting user matches the user in the URL.

has_permission(*request*, *view*)
Return *True* if permission is granted, *False* otherwise.

class JwtHasContentOrgFilterForRequestedCourse
The JWT used to authenticate contains the appropriate content provider filter for the requested course resource.

has_permission(*request*, *view*)
Ensure that the course_id kwarg provided to the view contains one of the organizations specified in the content provider filters in the JWT used to authenticate.

class JwtHasScope
The request is authenticated as a user and the token used has the right scope.

has_permission(*request*, *view*)
Return *True* if permission is granted, *False* otherwise.

class JwtHasUserFilterForRequestedUser
The JWT used to authenticate contains the appropriate user filter for the requested user resource.

has_permission(*request*, *view*)
If the JWT has a user filter, verify that the filtered user value matches the user in the URL.

class JwtRestrictedApplication
Allows access if the request was successfully authenticated with JwtAuthentication by a RestrictedApplication.

has_permission(*request*, *view*)
Return *True* if permission is granted, *False* otherwise.

class LoginRedirectIfUnauthenticated
A DRF permission class that will login redirect unauthorized users.

It can be used to convert a plain Django view that was using @login_required into a DRF APIView, which is useful to enable our DRF JwtAuthentication class.

Requires JwtRedirectToLoginIfUnauthenticatedMiddleware to work.

class NotJwtRestrictedApplication

Allows access if either the request was not authenticated with JwtAuthentication, or if it was successfully authenticated with JwtAuthentication and the Jwt was not flagged as restricted.

Note: Anonymous access will also pass this permission.

has_permission(*request*, *view*)

Return *True* if permission is granted, *False* otherwise.

3.5 Utility Functions

This module contains useful utility functions.

3.6 Change Log

3.6.1 Unreleased

3.6.2 [7.0.1] - 2021-08-10

Fixed

- Removed dropped `require_exp` and `require_iat` options from jwt.decode and instead used `require` option with both `exp` and `iat`. For more info visit this: https://pyjwt.readthedocs.io/en/stable/changelog.html#dropped-deprecated-require-options-in-jwt-decode
- This fixes an error in previous release which had a multiple breaking changes

3.6.3 [7.0.0] - 2021-08-03

Changed

- **BREAKING CHANGE:** `generate_jwt_token`: Now returns string (instead of bytes), and no longer requires decoding. This was to keep consistent with change to `jwt.encode` in *pyjwt* upgrade (see below).
- **BREAKING CHANGE:** Upgraded dependency `pyjwt[crypto]` to 2.1.0, which introduces its own breaking changes that may affect consumers of this library. Pay careful attention to the 2.0.0 breaking changes documented in https://pyjwt.readthedocs.io/en/stable/changelog.html#v2-0-0.

3.6.4 [6.6.0] - 2021-07-13

Added

- Added support for django3.1 and 3.2

3.6.5 [6.5.0] - 2021-02-12

Added

- Added a new custom attribute *jwt_auth_failed* to both monitor failures, and to help prepare for future refactors.

3.6.6 [6.4.0] - 2021-01-19

Added

- Added a new custom attribute *request_is_staff_or_superuser*

3.6.7 [6.3.0] - 2021-01-12

Removed

- Drop support for Python 3.5

3.6.8 [6.2.0] - 2020-08-24

Updated

- Renamed "custom metric" to "custom attribute" throughout the repo. This was based on a decision (ADR) captured in edx-django-utils.
  - Deprecated RequestMetricsMiddleware due to rename. Use RequestCustomAttributesMiddleware instead.

3.6.9 [6.1.2] - 2020-07-19

Fixed

- *_get_user_from_jwt* no longer throws an *UnsupportedMediaType* error for failing to parse "new user" requests.

3.6.10 [6.1.1] - 2020-07-19

Fixed

- Latest *drf-jwt* is throwing error in case of any other Authorization Header. Fixing that issue in *JwtAuthentication* class.

3.6.11 [6.1.0] - 2020-06-26

Changed

- Update *drf-jwt* to pull in new allow-list(they called it blacklist) feature.

Added

Fixed

3.6.12 [6.0.0] - 2020-05-05

Changed

- **BREAKING CHANGE**: Renamed 'request_auth_type' to 'request_auth_type_guess'. This makes it more clear that this metric could report the wrong value in certain cases. This could break dashboards or alerts that relied on this metric.
- **BREAKING CHANGE**: Renamed value *session-or-unknown* to *session-or-other*. This name makes it more clear that it is the method of authentication that is in question, not whether or not the user is authenticated. This could break dashboards or alerts that relied on this metric.

Added

- Added 'jwt-cookie' as new value for 'request_auth_type_guess'.
- Added new 'request_authenticated_user_found_in_middleware' metric. Helps identify for what middleware step the request user was set, if it was set. Example values: 'process_request', 'process_view', 'process_response', or 'process_exception'.

Fixed

- Fixed/Added setting of authentication metrics for exceptions as well.
- Fixed 'request_auth_type_guess' to be more accurate when recording values of 'unauthenticated' and 'no-user'.

Python Module Index

e

Index