Why use Authentication & Permission?

Currently our API doesn't have any restrictions on who can edit or delete Data.
We'd like to have some more advanced behavior in order to make sure that:

- Data is always associated with a creator.
- Only authenticated users may create Data.
- Only the creator of a Data may update or delete it.
- Unauthenticated requests should have full read-only access.

Authentication

Authentication is the mechanism of associating an incoming request with a set of identifying credentials, such as the user the request came from, or the token that it was signed with. The permission and throttling policies can then use those credentials to determine if the request should be permitted.

Authentication is always run at the very start of the view, before the permission and throttling checks occur, and before any other code is allowed to proceed.

Authentication

REST framework provides a number of authentication schemes out of the box, and also allows you to implement custom schemes.

- BasicAuthentication
- SessionAuthentication
- TokenAuthentication
- RemoteUserAuthentication
- Custom authentication

BasicAuthentication

This authentication scheme uses HTTP Basic Authentication, signed against a user's username and password.

Basic authentication is generally only appropriate for testing.

If successfully authenticated, BasicAuthentication provides the following credentials.

- *request.user* will be a Django User instance.
- *request.auth* will be None.

Unauthenticated responses that are denied permission will result in an HTTP 401 Unauthorized response with an appropriate WWW-Authenticate header. For example:

WWW-Authenticate: Basic realm="api"

BasicAuthentication

Note: If you use BasicAuthentication in production you must ensure that your API is only available over https.

You should also ensure that your API clients will always re-request the username and password at login, and will never store those details to persistent storage.

Permission

Permissions are used to grant or deny access for different classes of users to different parts of the API.

Permission checks are always run at the very start of the view, before any other code is allowed to proceed.

Permission checks will typically use the authentication information in the *request.user* and *request.auth* properties to determine if the incoming request should be permitted.

Permission Classes

Permissions in REST framework are always defined as a list of permission classes.

- AllowAny
- IsAuthenticated
- IsAdminUser
- IsAuthenticatedOrReadOnly
- DjangoModelPermissions
- DjangoModelPermissionsOrAnonReadOnly
- DjangoObjectPermissions
- Custom Permissions

AllowAny

The AllowAny permission class will allow unrestricted access, regardless of if the request was authenticated or unauthenticated.

This permission is not strictly required, since you can achieve the same result by using an empty list or tuple for the permissions setting, but you may find it useful to specify this class because it makes the intention explicit.

IsAuthenticated

The IsAuthenticated permission class will deny permission to any unauthenticated user, and allow permission otherwise.

This permission is suitable if you want your API to only be accessible to registered users.

IsAdminUser

The IsAdminUser permission class will deny permission to any user, unless user.is_staff is True in which case permission will be allowed.

This permission is suitable if you want your API to only be accessible to a subset of trusted administrators.