source: settings.py

Settings

> Namespaces are one honking great idea - let's do more of those!
>
> — The Zen of Python

Configuration for REST framework is all namespaced inside a single Django setting, named `REST_FRAMEWORK`.

For example your project's `settings.py` file might include something like this:

```
REST_FRAMEWORK = {
    'DEFAULT_RENDERER_CLASSES': (
        'rest_framework.renderers.JSONRenderer',
    ),
    'DEFAULT_PARSER_CLASSES': (
        'rest_framework.parsers.JSONParser',
    )
}
```

Accessing settings

If you need to access the values of REST framework's API settings in your project, you should use the `api_settings` object. For example.

```
from rest_framework.settings import api_settings

print api_settings.DEFAULT_AUTHENTICATION_CLASSES
```

The `api_settings` object will check for any user-defined settings, and otherwise fall back to the default values. Any setting that uses string import paths to refer to a class will automatically import and return the referenced class, instead of the string literal.

API Reference

API policy settings

The following settings control the basic API policies, and are applied to every `APIView` class based view, or `@api_view` function based view.

DEFAULT_RENDERER_CLASSES

A list or tuple of renderer classes, that determines the default set of renderers that may be used when returning a `Response` object.

Default:

```
(
    'rest_framework.renderers.JSONRenderer',
    'rest_framework.renderers.BrowsableAPIRenderer',
)
```

DEFAULT_PARSER_CLASSES

A list or tuple of parser classes, that determines the default set of parsers used when accessing the `request.data` property.

Default:

```
(
    'rest_framework.parsers.JSONParser',
    'rest_framework.parsers.FormParser',
    'rest_framework.parsers.MultiPartParser'
)
```

DEFAULT_AUTHENTICATION_CLASSES

A list or tuple of authentication classes, that determines the default set of authenticators used when accessing the `request.user` or `request.auth` properties.

Default:

```
(
    'rest_framework.authentication.SessionAuthentication',
    'rest_framework.authentication.BasicAuthentication'
)
```

DEFAULT_PERMISSION_CLASSES

A list or tuple of permission classes, that determines the default set of permissions checked at the start of a view. Permission must be granted by every class in the list.

Default:

```
(
    'rest_framework.permissions.AllowAny',
)
```

DEFAULT_THROTTLE_CLASSES

A list or tuple of throttle classes, that determines the default set of throttles checked at the start of a view.

Default: `()`

DEFAULT_CONTENT_NEGOTIATION_CLASS

A content negotiation class, that determines how a renderer is selected for the response, given an incoming request.

Default: `'rest_framework.negotiation.DefaultContentNegotiation'`

Generic view settings

The following settings control the behavior of the generic class based views.

DEFAULT_PAGINATION_SERIALIZER_CLASS

A class the determines the default serialization style for paginated responses.

Default: `rest_framework.pagination.PaginationSerializer`

DEFAULT_FILTER_BACKENDS

A list of filter backend classes that should be used for generic filtering. If set to `None` then generic filtering is disabled.

PAGINATE_BY

The default page size to use for pagination. If set to `None`, pagination is disabled by default.

Default: `None`

PAGINATE_BY_PARAM

This setting is pending deprecation.

See the pagination documentation for further guidance on setting the pagination style.

The name of a query parameter, which can be used by the client to override the default page size to

use for pagination. If set to `None`, clients may not override the default page size.

For example, given the following settings:

```
REST_FRAMEWORK = {
    'PAGINATE_BY': 10,
    'PAGINATE_BY_PARAM': 'page_size',
}
```

A client would be able to modify the pagination size by using the `page_size` query parameter. For example:

```
GET http://example.com/api/accounts?page_size=25
```

Default: `None`

MAX_PAGINATE_BY

This setting is pending deprecation.

See the pagination documentation for further guidance on setting the pagination style.

The maximum page size to allow when the page size is specified by the client. If set to `None`, then no maximum limit is applied.

For example, given the following settings:

```
REST_FRAMEWORK = {
    'PAGINATE_BY': 10,
    'PAGINATE_BY_PARAM': 'page_size',
    'MAX_PAGINATE_BY': 100
}
```

A client request like the following would return a paginated list of up to 100 items.

```
GET http://example.com/api/accounts?page_size=999
```

Default: `None`

SEARCH_PARAM

The name of a query parameter, which can be used to specify the search term used by

`SearchFilter`.

Default: `search`

ORDERING_PARAM

The name of a query parameter, which can be used to specify the ordering of results returned by `OrderingFilter`.

Default: `ordering`

Versioning settings

DEFAULT_VERSION

The value that should be used for `request.version` when no versioning information is present.

Default: `None`

ALLOWED_VERSIONS

If set, this value will restrict the set of versions that may be returned by the versioning scheme, and will raise an error if the provided version if not in this set.

Default: `None`

VERSION_PARAMETER

The string that should used for any versioning parameters, such as in the media type or URL query parameters.

Default: `'version'`

Authentication settings

The following settings control the behavior of unauthenticated requests.

UNAUTHENTICATED_USER

The class that should be used to initialize `request.user` for unauthenticated requests.

Default: `django.contrib.auth.models.AnonymousUser`

UNAUTHENTICATED_TOKEN

The class that should be used to initialize `request.auth` for unauthenticated requests.

Default: `None`

Test settings

The following settings control the behavior of APIRequestFactory and APIClient

TEST_REQUEST_DEFAULT_FORMAT

The default format that should be used when making test requests.

This should match up with the format of one of the renderer classes in the `TEST_REQUEST_RENDERER_CLASSES` setting.

Default: `'multipart'`

TEST_REQUEST_RENDERER_CLASSES

The renderer classes that are supported when building test requests.

The format of any of these renderer classes may be used when constructing a test request, for example: `client.post('/users', {'username': 'jamie'}, format='json')`

Default:

```
(
    'rest_framework.renderers.MultiPartRenderer',
    'rest_framework.renderers.JSONRenderer'
)
```

Browser overrides

The following settings provide URL or form-based overrides of the default browser behavior.

FORM_METHOD_OVERRIDE

The name of a form field that may be used to override the HTTP method of the form.

If the value of this setting is `None` then form method overloading will be disabled.

Default: `'_method'`

FORM_CONTENT_OVERRIDE

The name of a form field that may be used to override the content of the form payload. Must be used together with `FORM_CONTENTTYPE_OVERRIDE`.

If either setting is `None` then form content overloading will be disabled.

Default: `'_content'`

FORM_CONTENTTYPE_OVERRIDE

The name of a form field that may be used to override the content type of the form payload. Must be used together with `FORM_CONTENT_OVERRIDE`.

If either setting is `None` then form content overloading will be disabled.

Default: `'_content_type'`

URL_ACCEPT_OVERRIDE

The name of a URL parameter that may be used to override the HTTP `Accept` header.

If the value of this setting is `None` then URL accept overloading will be disabled.

Default: `'accept'`

URL_FORMAT_OVERRIDE

The name of a URL parameter that may be used to override the default `Accept` header based content negotiation.

If the value of this setting is `None` then URL format overloading will be disabled.

Default: `'format'`

Date and time formatting

The following settings are used to control how date and time representations may be parsed and rendered.

DATETIME_FORMAT

A format string that should be used by default for rendering the output of `DateTimeField` serializer fields. If `None`, then `DateTimeField` serializer fields will return Python `datetime` objects, and the datetime encoding will be determined by the renderer.

May be any of `None`, `'iso-8601'` or a Python strftime format string.

Default: `'iso-8601'`

DATETIME_INPUT_FORMATS

A list of format strings that should be used by default for parsing inputs to `DateTimeField` serializer

fields.

May be a list including the string `'iso-8601'` or Python strftime format strings.

Default: `['iso-8601']`

DATE_FORMAT

A format string that should be used by default for rendering the output of `DateField` serializer fields. If `None`, then `DateField` serializer fields will return Python `date` objects, and the date encoding will be determined by the renderer.

May be any of `None`, `'iso-8601'` or a Python strftime format string.

Default: `'iso-8601'`

DATE_INPUT_FORMATS

A list of format strings that should be used by default for parsing inputs to `DateField` serializer fields.

May be a list including the string `'iso-8601'` or Python strftime format strings.

Default: `['iso-8601']`

TIME_FORMAT

A format string that should be used by default for rendering the output of `TimeField` serializer fields. If `None`, then `TimeField` serializer fields will return Python `time` objects, and the time encoding will be determined by the renderer.

May be any of `None`, `'iso-8601'` or a Python strftime format string.

Default: `'iso-8601'`

TIME_INPUT_FORMATS

A list of format strings that should be used by default for parsing inputs to `TimeField` serializer fields.

May be a list including the string `'iso-8601'` or Python strftime format strings.

Default: `['iso-8601']`

Encodings

UNICODE_JSON

When set to `True`, JSON responses will allow unicode characters in responses. For example:

```
{"unicode black star":"□"}
```

When set to `False`, JSON responses will escape non-ascii characters, like so:

```
{"unicode black star":"\u2605"}
```

Both styles conform to RFC 4627, and are syntactically valid JSON. The unicode style is preferred as being more user-friendly when inspecting API responses.

Default: `True`

COMPACT_JSON

When set to `True`, JSON responses will return compact representations, with no spacing after `':'` and `','` characters. For example:

```
{"is_admin":false,"email":"jane@example"}
```

When set to `False`, JSON responses will return slightly more verbose representations, like so:

```
{"is_admin": false, "email": "jane@example"}
```

The default style is to return minified responses, in line with Heroku's API design guidelines.

Default: `True`

COERCE_DECIMAL_TO_STRING

When returning decimal objects in API representations that do not support a native decimal type, it is normally best to return the value as a string. This avoids the loss of precision that occurs with binary floating point implementations.

When set to `True`, the serializer `DecimalField` class will return strings instead of `Decimal` objects. When set to `False`, serializers will return `Decimal` objects, which the default JSON encoder will return as floats.

Default: `True`

View names and descriptions

The following settings are used to generate the view names and descriptions, as used in responses to `OPTIONS` requests, and as used in the browsable API.

VIEW_NAME_FUNCTION

A string representing the function that should be used when generating view names.

This should be a function with the following signature:

```
view_name(cls, suffix=None)
```

- `cls`: The view class. Typically the name function would inspect the name of the class when generating a descriptive name, by accessing `cls.__name__`.
- `suffix`: The optional suffix used when differentiating individual views in a viewset.

Default: `'rest_framework.views.get_view_name'`

VIEW_DESCRIPTION_FUNCTION

A string representing the function that should be used when generating view descriptions.

This setting can be changed to support markup styles other than the default markdown. For example, you can use it to support `rst` markup in your view docstrings being output in the browsable API.

This should be a function with the following signature:

```
view_description(cls, html=False)
```

- `cls`: The view class. Typically the description function would inspect the docstring of the class when generating a description, by accessing `cls.__doc__`
- `html`: A boolean indicating if HTML output is required. `True` when used in the browsable API, and `False` when used in generating `OPTIONS` responses.

Default: `'rest_framework.views.get_view_description'`

Miscellaneous settings

EXCEPTION_HANDLER

A string representing the function that should be used when returning a response for any given exception. If the function returns `None`, a 500 error will be raised.

This setting can be changed to support error responses other than the default `{"detail": "Failure..."}` responses. For example, you can use it to provide API responses like `{"errors":`

`[{"message": "Failure...", "code": ""} ...]}`.

This should be a function with the following signature:

```
exception_handler(exc, context)
```

- `exc`: The exception.

Default: `'rest_framework.views.exception_handler'`

NON_FIELD_ERRORS_KEY

A string representing the key that should be used for serializer errors that do not refer to a specific field, but are instead general errors.

Default: `'non_field_errors'`

URL_FIELD_NAME

A string representing the key that should be used for the URL fields generated by `HyperlinkedModelSerializer`.

Default: `'url'`

FORMAT_SUFFIX_KWARG

The name of a parameter in the URL conf that may be used to provide a format suffix.

Default: `'format'`

NUM_PROXIES

An integer of 0 or more, that may be used to specify the number of application proxies that the API runs behind. This allows throttling to more accurately identify client IP addresses. If set to `None` then less strict IP matching will be used by the throttle classes.

Default: `None`