source: negotiation.py

Content negotiation

> HTTP has provisions for several mechanisms for "content negotiation" - the process of selecting the best representation for a given response when there are multiple representations available.
>
> — RFC 2616, Fielding et al.

Content negotiation is the process of selecting one of multiple possible representations to return to a client, based on client or server preferences.

Determining the accepted renderer

REST framework uses a simple style of content negotiation to determine which media type should be returned to a client, based on the available renderers, the priorities of each of those renderers, and the client's `Accept:` header. The style used is partly client-driven, and partly server-driven.

1. More specific media types are given preference to less specific media types.
2. If multiple media types have the same specificity, then preference is given to based on the ordering of the renderers configured for the given view.

For example, given the following `Accept` header:

```
application/json; indent=4, application/json, application/yaml, text/html, */*
```

The priorities for each of the given media types would be:

- `application/json; indent=4`
- `application/json`, `application/yaml` and `text/html`
- `*/*`

If the requested view was only configured with renderers for `YAML` and `HTML`, then REST framework would select whichever renderer was listed first in the `renderer_classes` list or `DEFAULT_RENDERER_CLASSES` setting.

For more information on the `HTTP Accept` header, see RFC 2616

Note: "q" values are not taken into account by REST framework when determining preference. The use of "q" values negatively impacts caching, and in the author's opinion they are an unnecessary and overcomplicated approach to content negotiation.

This is a valid approach as the HTTP spec deliberately underspecifies how a server should weight server-based preferences against client-based preferences.

Custom content negotiation

It's unlikely that you'll want to provide a custom content negotiation scheme for REST framework, but you can do so if needed. To implement a custom content negotiation scheme override `BaseContentNegotiation`.

REST framework's content negotiation classes handle selection of both the appropriate parser for the request, and the appropriate renderer for the response, so you should implement both the `.select_parser(request, parsers)` and `.select_renderer(request, renderers, format_suffix)` methods.

The `select_parser()` method should return one of the parser instances from the list of available parsers, or `None` if none of the parsers can handle the incoming request.

The `select_renderer()` method should return a two-tuple of (renderer instance, media type), or raise a `NotAcceptable` exception.

Example

The following is a custom content negotiation class which ignores the client request when selecting the appropriate parser or renderer.

```
from rest_framework.negotiation import BaseContentNegotiation

class IgnoreClientContentNegotiation(BaseContentNegotiation):
    def select_parser(self, request, parsers):
        """
        Select the first parser in the `.parser_classes` list.
        """
        return parsers[0]

    def select_renderer(self, request, renderers, format_suffix):
        """
        Select the first renderer in the `.renderer_classes` list.
        """
        return (renderers[0], renderers[0].media_type)
```

Setting the content negotiation

The default content negotiation class may be set globally, using the `DEFAULT_CONTENT_NEGOTIATION_CLASS` setting. For example, the following settings would use our example `IgnoreClientContentNegotiation` class.

```
REST_FRAMEWORK = {
    'DEFAULT_CONTENT_NEGOTIATION_CLASS':
'myapp.negotiation.IgnoreClientContentNegotiation',
}
```

You can also set the content negotiation used for an individual view, or viewset, using the `APIView` class based views.

```
from myapp.negotiation import IgnoreClientContentNegotiation
from rest_framework.response import Response
from rest_framework.views import APIView

class NoNegotiationView(APIView):
    """
    An example view that does not perform content negotiation.
    """
    content_negotiation_class = IgnoreClientContentNegotiation

    def get(self, request, format=None):
        return Response({
            'accepted media type': request.accepted_renderer.media_type
        })
```