source: reverse.py

Returning URLs

> The central feature that distinguishes the REST architectural style from other network-based styles is its emphasis on a uniform interface between components.
>
> — Roy Fielding, Architectural Styles and the Design of Network-based Software Architectures

As a rule, it's probably better practice to return absolute URIs from your Web APIs, such as `http://example.com/foobar`, rather than returning relative URIs, such as `/foobar`.

The advantages of doing so are:

- It's more explicit.
- It leaves less work for your API clients.
- There's no ambiguity about the meaning of the string when it's found in representations such as JSON that do not have a native URI type.
- It makes it easy to do things like markup HTML representations with hyperlinks.

REST framework provides two utility functions to make it more simple to return absolute URIs from your Web API.

There's no requirement for you to use them, but if you do then the self-describing API will be able to automatically hyperlink its output for you, which makes browsing the API much easier.

reverse

Signature: `reverse(viewname, *args, **kwargs)`

Has the same behavior as `django.core.urlresolvers.reverse`, except that it returns a fully qualified URL, using the request to determine the host and port.

You should **include the request as a keyword argument** to the function, for example:

```
from rest_framework.reverse import reverse
from rest_framework.views import APIView
from django.utils.timezone import now

class APIRootView(APIView):
    def get(self, request):
        year = now().year
        data = {
            ...
            'year-summary-url': reverse('year-summary', args=[year],
```

```
request=request)
        }
        return Response(data)
```

reverse_lazy

Signature: `reverse_lazy(viewname, *args, **kwargs)`

Has the same behavior as `django.core.urlresolvers.reverse_lazy`, except that it returns a fully qualified URL, using the request to determine the host and port.

As with the `reverse` function, you should **include the request as a keyword argument** to the function, for example:

```
api_root = reverse_lazy('api-root', request=request)
```