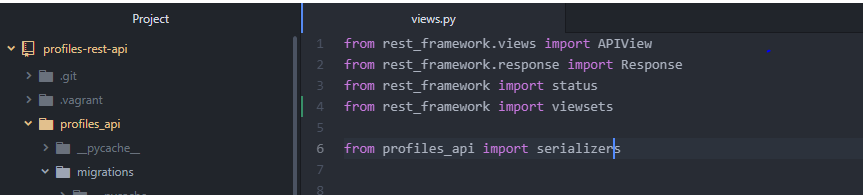
**Create a ViewSet:**

1. In Atom editor, under profiles\_api app, open views.py

2. Perform Imports

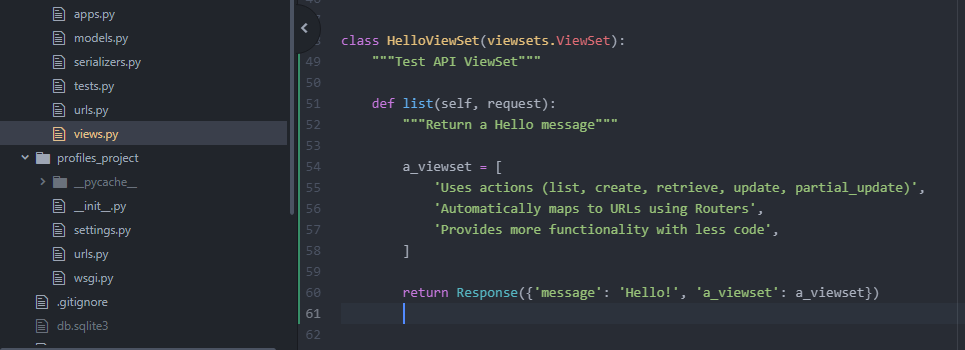


3. Create a new class

Here we add functions that represent actions that we would perform on an API.

list – HTTP GET to the root of the endpoint linked to our viewset.

Lists a set of object that the viewset represents.



request – passed in by DRF when the request is made.

So when the request is made to the root url of the viewset, it will call the list function which will create the list object and it will return it as a response to our API call.

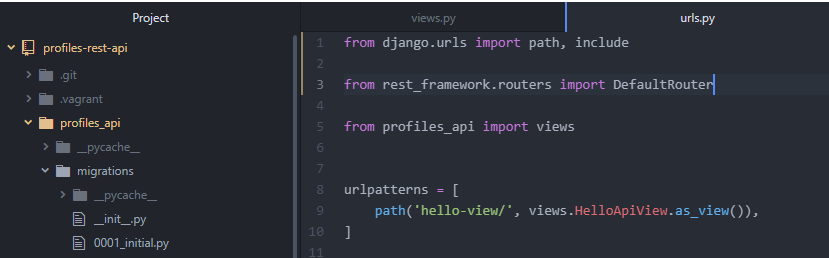
**Add URL Router:**

Configure a URL to point to our viewset.

We need to register our viewset with a URL to make it accessible through our API.

router – class provided by DRF to generate different routes that are available for our viewsets.

1. Perform Imports.

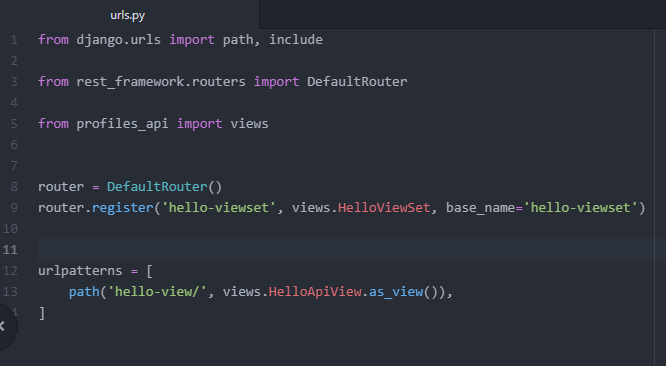


Imports function called ‘include’

include – used for including list of URLs in the URL pattern and assigning the lists to a specific URL.

Import DefaultRouter from rest\_framework.

2. Create default router and register the viewset with the default router.



Assign the DefaultRouter to a variable.

‘router.register’ – register specific viewsets with our router.

‘hello-viewset’ – name of the URL that we want to create.

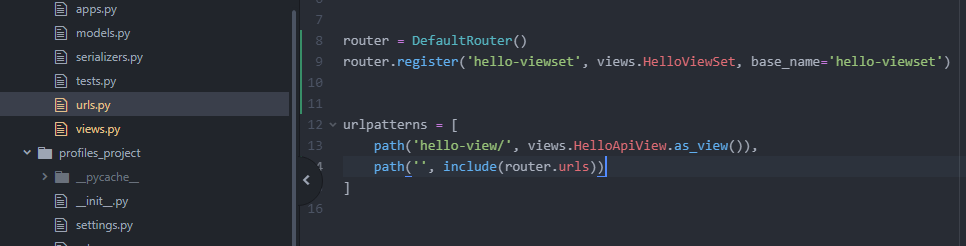
{Here the router will create all 4 URLs for us, so we don’t need to specify a ‘/’ when we define our viewset URL name}

‘views.HelloViewSet’ – viewset that we wish to register to this URL.

“base\_name=’hello-viewset’” - specify a base name for our viewset.

{Used for retrieving the URLs in our router}

3. Pass the URLs into urlpatterns (in urls.py in profiles\_api)



As we register new routes with our router it generates a list of URLs that are associated with our viewset. It figures out the URLs that are required for all of the functions that we add to our viewset and then it generates the URLs list which we can pass using the path function and include functions to the urlpatterns.

‘ ’ – The reason to specify a blank string here is that we don’t want to put a prefix to the URL. We want to include all of the URLs in the base of this URLs file.

**Testing our ViewSet:**

1. In Gitbash, connect to the vagrant server.

-> vagrant up

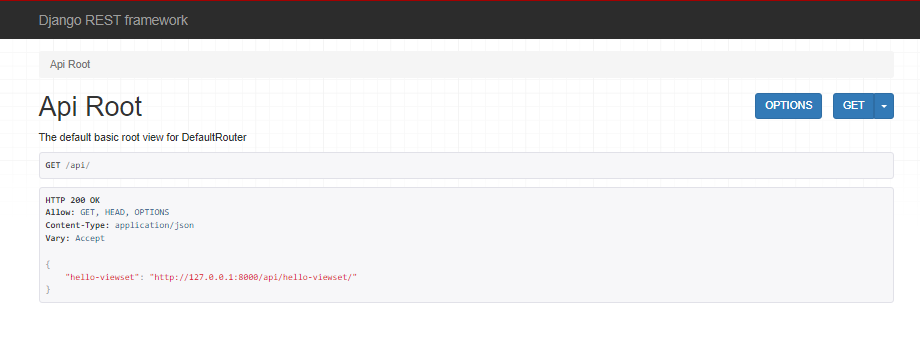
-> vagrant ssh

-> cd /vagrant

-> source ~/env/bin/activate

-> python manage.py runserver 0.0.0.0:8000

2. Go to your browser and open <http://127.0.0.1:8000/api/>



3. Click on the http link

