# Assignment Task:
# Integrate AI/ML in your Django Application

## Setting Up The REST API Project

So let's start from the very beginning. Install Django and DRF:

```
pip install django
pip install djangorestframework
```

Create a new Django project:

```
django-admin.py startproject myapi .
```

Navigate to the **myapi** folder:

```
cd myapi
```

Start a new app. I will call my app **core**:

```
django-admin.py startapp core
```

Here is what your project structure should look like:

```
myapi/
 |-- core/
 |    |-- migrations/
 |    |-- __init__.py
 |    |-- admin.py
 |    |-- apps.py
 |    |-- models.py
 |    |-- tests.py
 |    +-- views.py
 |-- __init__.py
 |-- settings.py
 |-- urls.py
 +-- wsgi.py
manage.py
```

Add the **core** app (you created) and the **rest_framework** app (you installed) to the `INSTALLED_APPS`, inside the **settings.py** module:

**myapi/settings.py**

```
INSTALLED_APPS = [
    # Django Apps
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third-Party Apps
    'rest_framework',

    # Local Apps (Your project's apps)
    'myapi.core',
]
```

Return to the project root (the folder where the **manage.py** script is), and migrate the database:

```
python manage.py migrate
```

Let’s create our first API view just to test things out:

**myapi/core/views.py**

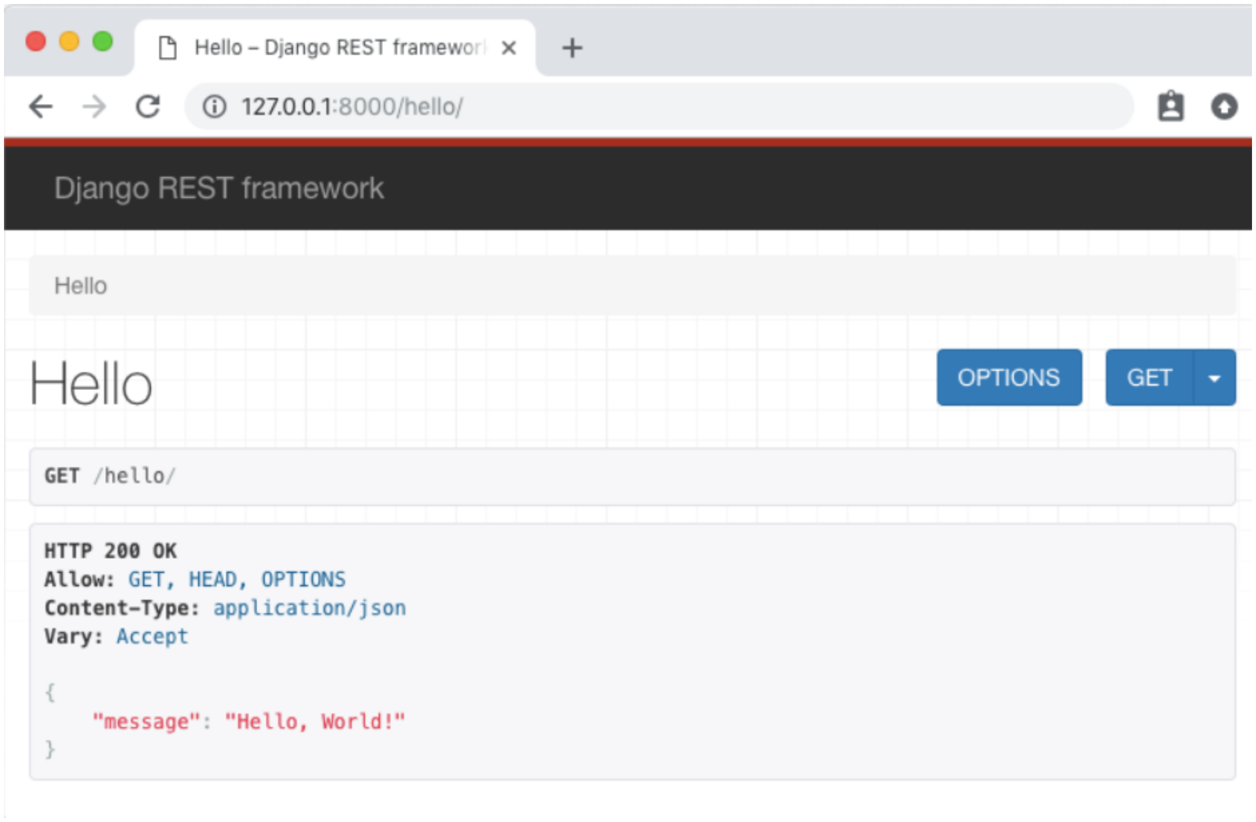
```
from rest_framework.views import APIView
from rest_framework.response import Response

class HelloView(APIView):
    def get(self, request):
        content = {'message': 'Hello, World!'}
        return Response(content)
```

Now register a path in the **urls.py** module:

**myapi/urls.py**

```
from django.urls import path
from myapi.core import views

urlpatterns = [
    path('hello/', views.HelloView.as_view(), name='hello'),
]
```

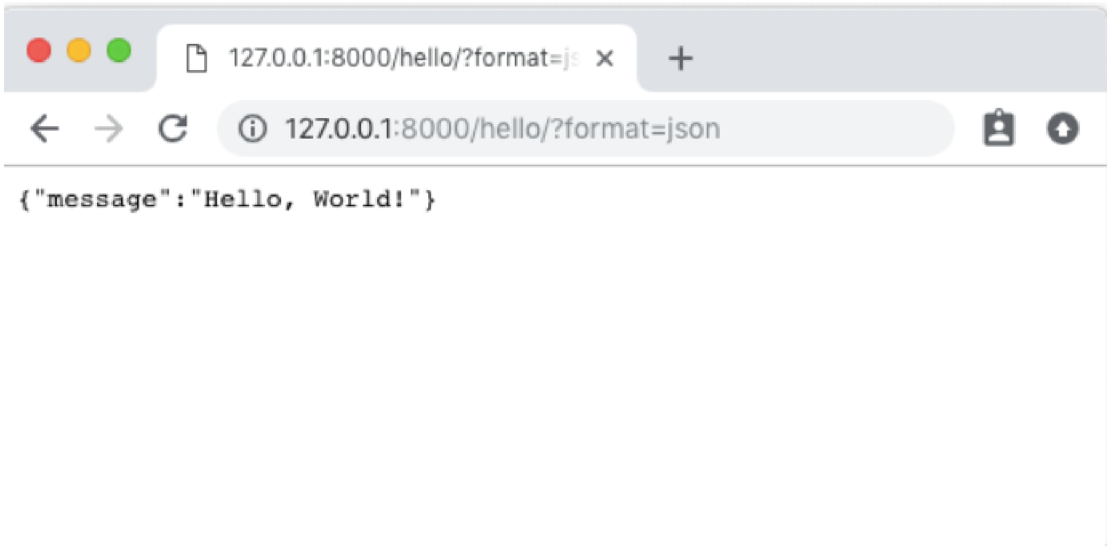
So now we have an API with just one endpoint `/hello/` that we can perform `GET` requests. We can use the browser to consume this endpoint, just by accessing the URL [http://127.0.0.1:8000/hello/](http://127.0.0.1:8000/hello/):



```
GET /hello/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "message": "Hello, World!"
}
```

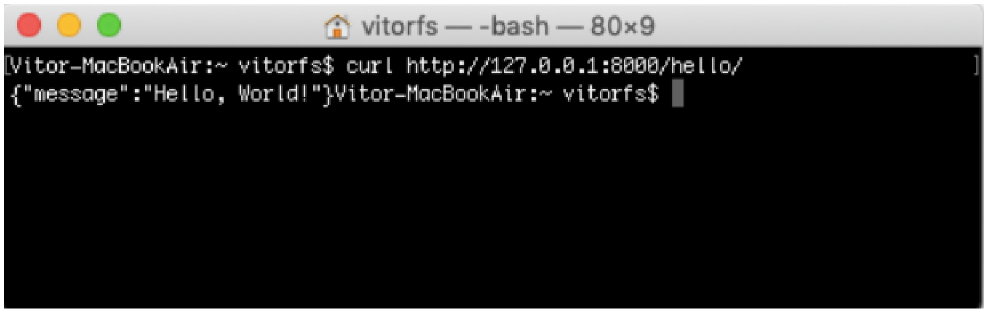
We can also ask to receive the response as plain JSON data by passing the `format` parameter in the querystring like `http://127.0.0.1:8000/hello/?format=json`:


127.0.0.1:8000/hello/?format=js
127.0.0.1:8000/hello/?format=json

```
{"message":"Hello, World!"}
```

Both methods are fine to try out a DRF API, but sometimes a command line tool is more handy as we can play more easily with the requests headers. You can use cURL, which is widely available on all major Linux/macOS distributions:

```
curl http://127.0.0.1:8000/hello/
```



```
Vitor-MacBookAir:~ vitorfs$ curl http://127.0.0.1:8000/hello/
{"message":"Hello, World!"}Vitor-MacBookAir:~ vitorfs$
```

## Implementing the Token Authentication

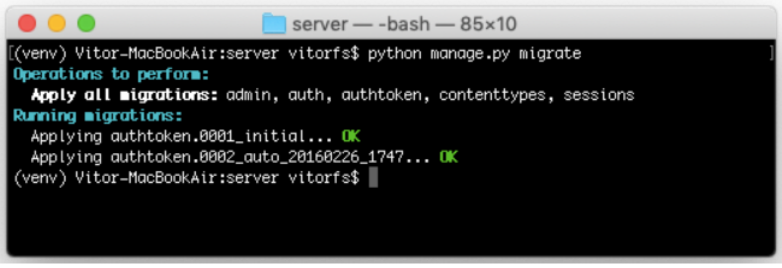
We need to add two pieces of information in our **settings.py** module. First include **rest_framework.authtoken** to your `INSTALLED_APPS` and include the `TokenAuthentication` to `REST_FRAMEWORK`:

### myapi/settings.py

```
INSTALLED_APPS = [
    # Django Apps
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third-Party Apps
    'rest_framework',
    'rest_framework.authtoken',  # <-- Here

    # Local Apps (Your project's apps)
    'myapi.core',
]

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',  # <-- And here
    ],
}
```

Migrate the database to create the table that will store the authentication tokens:

```
python manage.py migrate
```



```
(venv) Vitor-MacBookAir:server vitorfs$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, authtoken, contenttypes, sessions
Running migrations:
  Applying authtoken.0001_initial... OK
  Applying authtoken.0002_auto_20160226_1747... OK
(venv) Vitor-MacBookAir:server vitorfs$
```