

CM3035 Advanced Web Development

Lesson 7

Created by Ben Gay

CRUD

CRUD is an acronym for
CREATE, READ, UPDATE, DELETE

REST Framework

REST = representational state transfer

REST Api

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services

REST Api Example

localhost:8000/students/

OEM USB Drivers... Good book for be... 127.0.0.1:8000/pay/ Move WordPress f... http://192.168.0.1... hdb www.facebooklike... D-Link real-estate DlinkHomeCam

Django REST framework ben

Students List

OPTIONS GET

GET /students/

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

```
[
  {
    "id": 1,
    "name": "John",
    "alias": "cool guy"
  },
  {
    "id": 2,
    "name": "Mary",
    "alias": "Playful"
  }
]
```

Raw data HTML form

Name

Alias

POST

REST Api

Let's Begin to build REST Api with Django

REST Api

1. Run Virtual Environment
2. Create a new project - **L7proj**
3. Create a new Django app - **L7app**

Connect Django to Postgre database

```
5 # Database
6 # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
7
8 DATABASES = {
9     'default': {
10         # 'ENGINE': 'django.db.backends.sqlite3',
11         # 'NAME': BASE_DIR / 'db.sqlite3',
12         'ENGINE': 'django.db.backends.postgresql',
13         'NAME': 'simDatabase',
14         'USER': 'ben',
15         'PASSWORD': 'password',
16         'HOST': 'localhost',
17         'PORT': '5433',
18     }
19 }
```


Settings.py

Python3 manage.py migrate

```
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'L7app.apps.L7AppConfig',
41     #'rest_framework',
42 ]
43
```

Create super user

`python manage.py createsuperuser`

Leave username blank

Enter your email

Enter simple password


And

Hit **Y** to confirm

‘Superuser Created Successfully’

Login in as super user

```
python manage.py runserver  
localhost:8000/admin
```



A screenshot of a login form for the Django REST framework. The form is titled "Django REST framework" and is set against a light gray grid background. It contains two input fields: "Username:" with the value "ben" and "Password:" with masked characters ".....". Below the fields is a blue "Log in" button.

Django REST framework

Username:

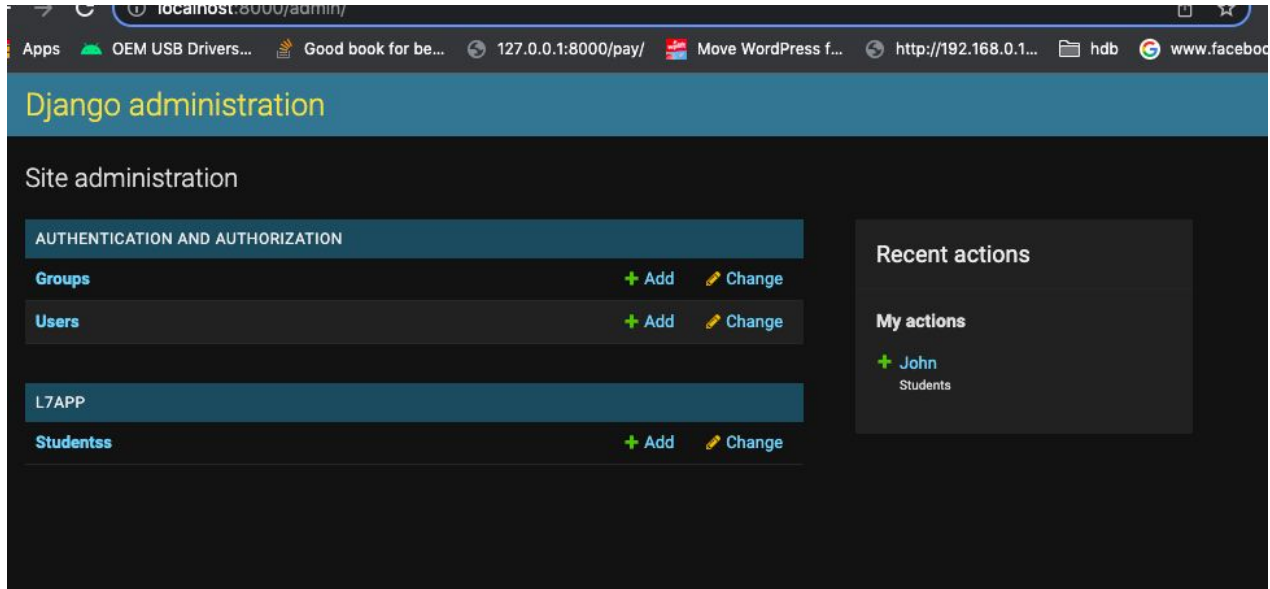
ben

Password:

.....

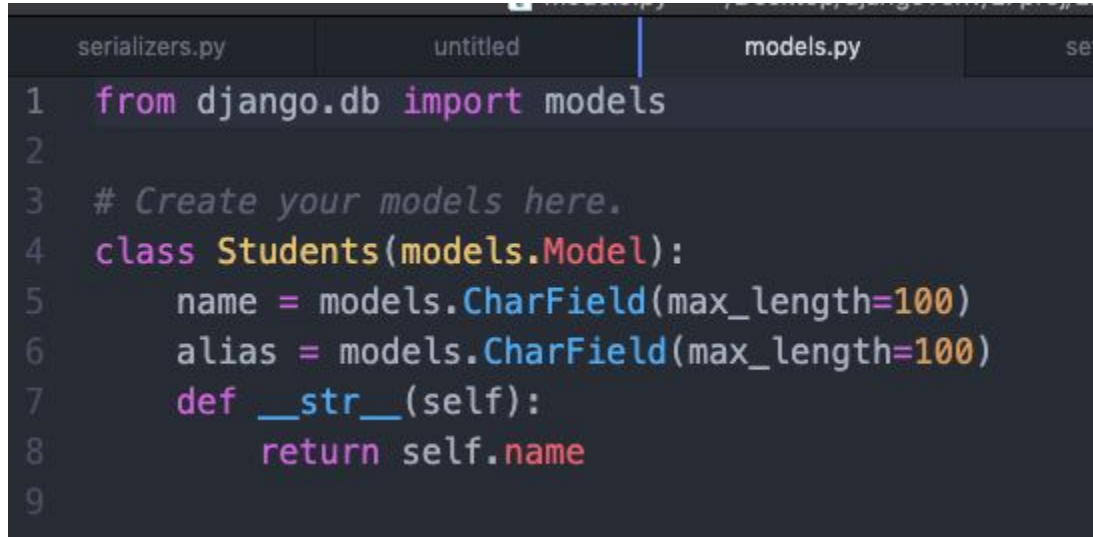
Log in

admin dashboard



models.py

Python3 manage.py makemigrations



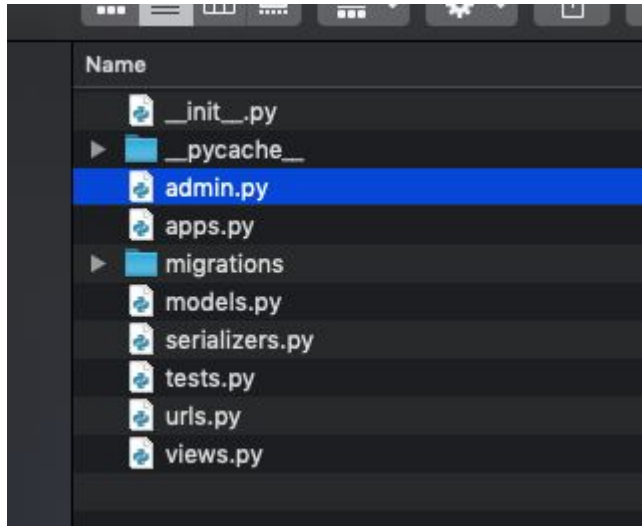
```
1 from django.db import models
2
3 # Create your models here.
4 class Students(models.Model):
5     name = models.CharField(max_length=100)
6     alias = models.CharField(max_length=100)
7     def __str__(self):
8         return self.name
9
```

Successfully created table

> L3app_topping	
> L7app_students	
> auth_group	
> auth_group_permissions	
> auth_permission	
> auth_user	
> auth_user_groups	
> auth_user_user_permissions	
> django_admin_log	
> django_content_type	
> django_migrations	

Data Output	Explain	Messages	Notifications
id [PK] bigint	name character varying (100)	alias character varying (100)	
1	1 John	cool guy	
2	2 Mary	Playful	

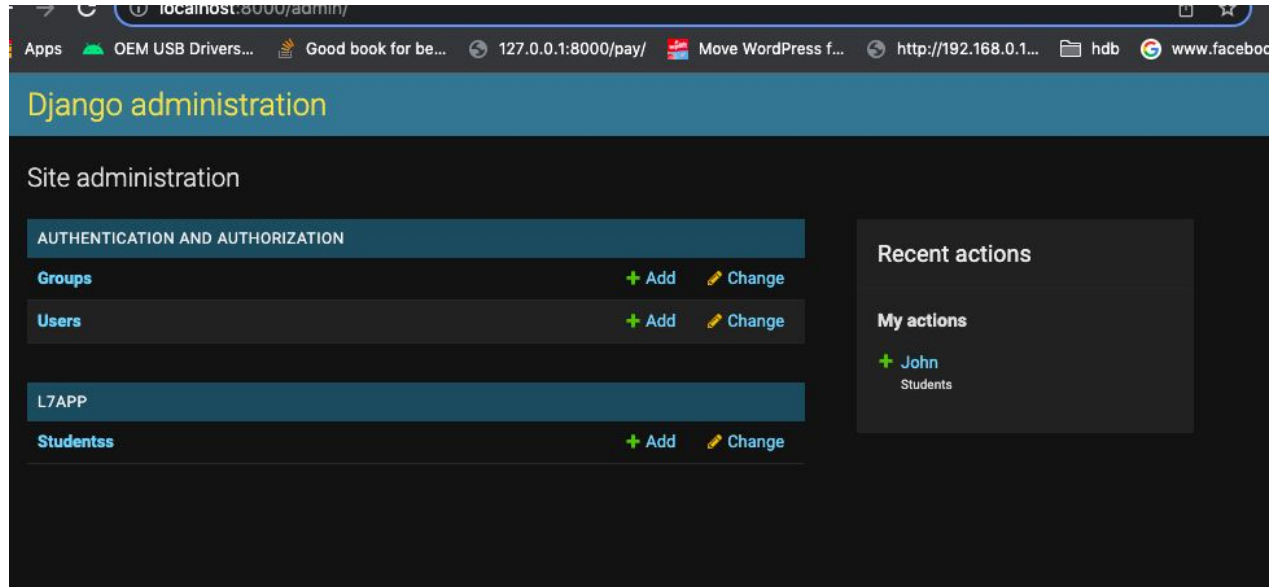
Register Students table to admin site



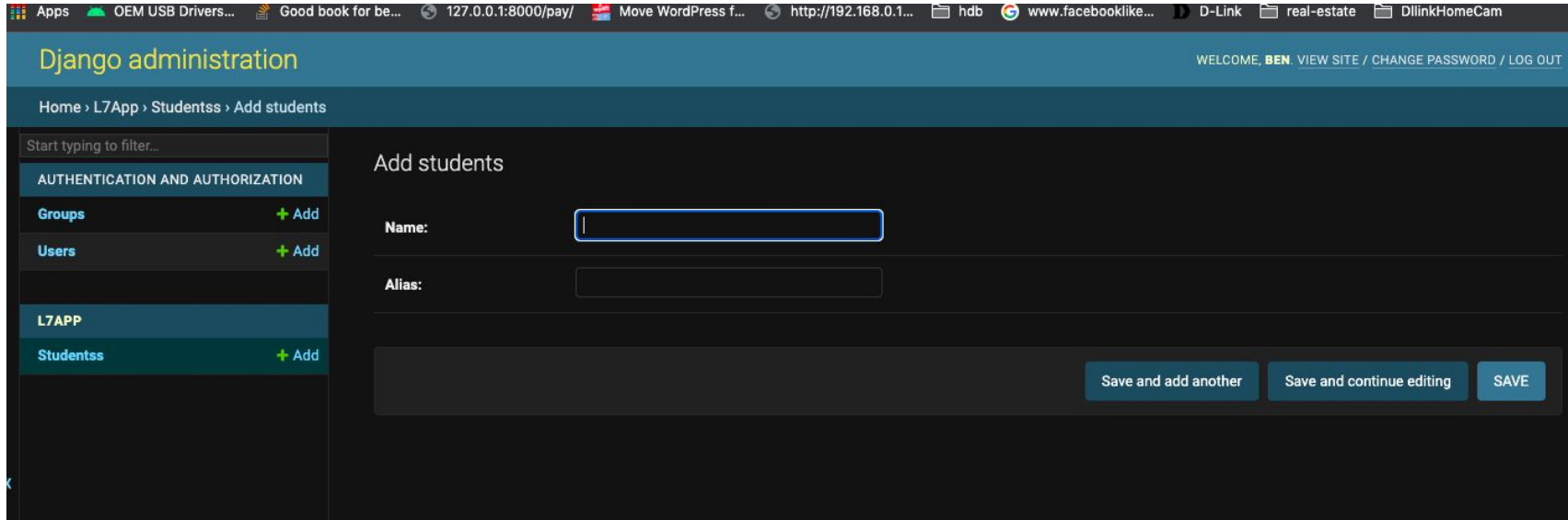
```
serializers.py  admin.py  untitled
1  from django.contrib import admin
2  from .models import Students
3
4  admin.site.register(Students)
5  # Register your models here.
6
```

Python3 manage.py runserver

localhost:8000/admin



Add a new student into database



The screenshot shows the Django administration interface. The top navigation bar is dark blue with the text "Django administration" on the left and "WELCOME BEN VIEW SITE / CHANGE PASSWORD / LOG OUT" on the right. Below the navigation bar is a breadcrumb trail: "Home > L7App > Studentss > Add students". On the left side, there is a sidebar with a search bar "Start typing to filter..." and a list of menu items. The "L7APP" section is expanded, showing "Studentss" with a "+ Add" link. The main content area is titled "Add students" and contains two form fields: "Name:" and "Alias:". The "Name:" field is currently empty and has a blue border. The "Alias:" field is also empty. At the bottom right of the form, there are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

Apps OEM USB Drivers... Good book for be... 127.0.0.1:8000/pay/ Move WordPress f... http://192.168.0.1... hdb www.facebooklike... D-Link real-estate DlinkHomeCam

Django administration WELCOME BEN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > L7App > Studentss > Add students

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

L7APP

Studentss + Add

Add students

Name:

Alias:

Save and add another Save and continue editing SAVE

Check to ensure entry successful

5

Data Output				Explain	Messages	Notifications
		id [PK] bigint 	name character varying (100) 	alias character varying (100) 		
1		1	John			cool guy
2		2	Mary			Playful

Setup REST Framework

Let's do it

Install REST Framework

Pip3 install djangorestframework

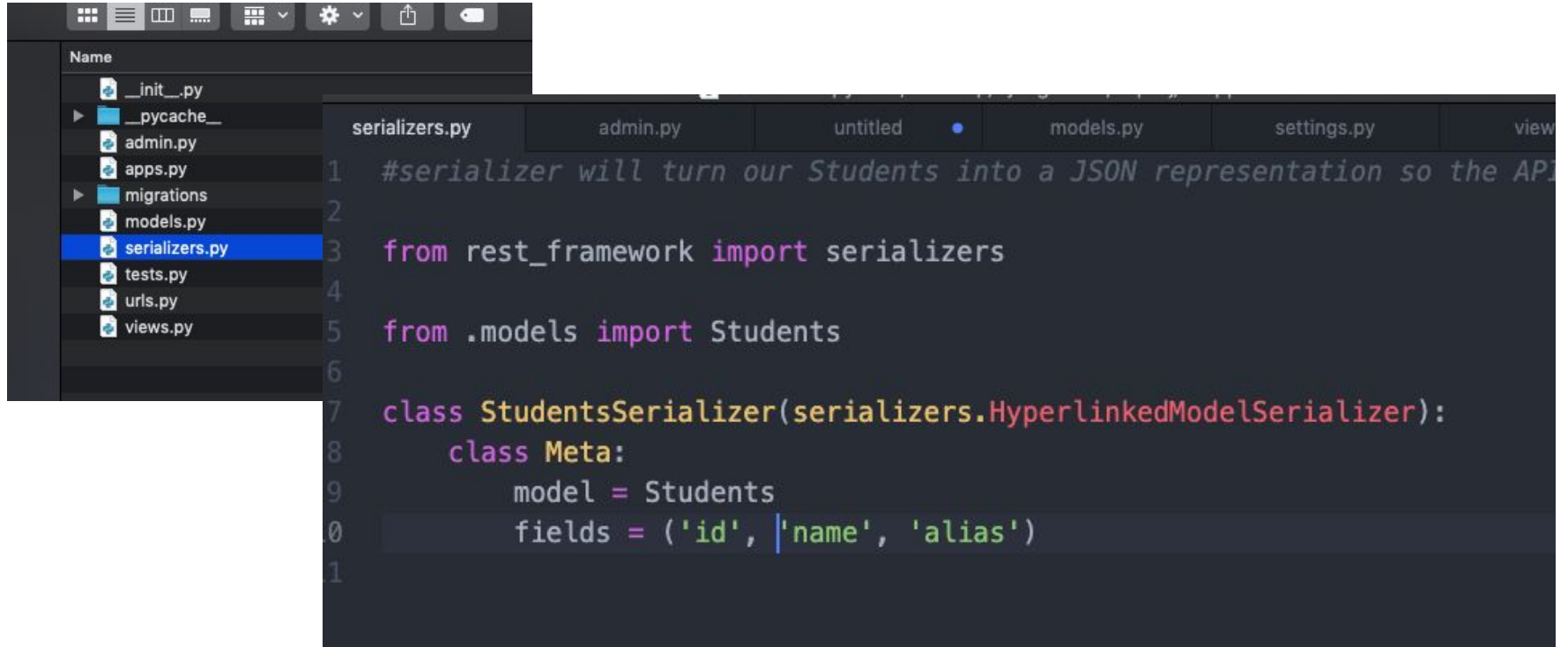
Settings.py

```
0
1  # Application definition
2
3  INSTALLED_APPS = [
4      'django.contrib.admin',
5      'django.contrib.auth',
6      'django.contrib.contenttypes',
7      'django.contrib.sessions',
8      'django.contrib.messages',
9      'django.contrib.staticfiles',
0      'L7app.apps.L7AppConfig',
1      'rest_framework',
2  ]
```

Serialization

Serialization is the process of converting a Model to JSON

L7app/serializers.py



The image shows a code editor interface with a file explorer on the left and a code editor on the right. The file explorer lists the following files: `__init__.py`, `__pycache__`, `admin.py`, `apps.py`, `migrations`, `models.py`, `serializers.py` (selected), `tests.py`, `urls.py`, and `views.py`. The code editor displays the content of `serializers.py` with the following code:

```
1  #serializer will turn our Students into a JSON representation so the API
2
3  from rest_framework import serializers
4
5  from .models import Students
6
7  class StudentsSerializer(serializers.HyperlinkedModelSerializer):
8      class Meta:
9          model = Students
10         fields = ('id', 'name', 'alias')
11
```

Display data

views.py

```
from django.shortcuts import render
from rest_framework import viewsets
from .serializers import StudentsSerializer
from .models import Students

# Create your views here.

class StudentsViewSet(viewsets.ModelViewSet):
    queryset = Students.objects.all().order_by('name')
    serializer_class = StudentsSerializer
```


L7proj/urls.py

```
13     1. Import the include() function: from dja
14     2. Add a URL to urlpatterns: path('blog/'
15     """
16     from django.contrib import admin
17     from django.urls import path, include
18
19     urlpatterns = [
20         path('admin/', admin.site.urls),
21         path('', include('L7app.urls')),
22     ]
23
```

L7app/urls.py

```
urls.py — L7app | serializers.py | admin.py | untitled | models.py | settings.py | views.py
from django.urls import include, path
from rest_framework import routers
from . import views

router = routers.DefaultRouter()
router.register(r'students', views.StudentsViewSet)

# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework'))
]
```

Python3 manage.py runserver

Go to localhost:8000

The image displays two screenshots of the Django REST framework API browser interface. The top screenshot shows the 'Api Root' endpoint, which is the default basic root view for the DefaultRouter. It displays the HTTP status '200 OK', allowed methods (GET, HEAD, OPTIONS), content type (application/json), and a JSON response with a 'students' field pointing to 'http://localhost:8000/students/'. The bottom screenshot shows the 'Students List' endpoint, which returns a list of two students: John (id: 1, alias: 'cool guy') and Mary (id: 2, alias: 'Playful'). Below the JSON response, there is a form with input fields for 'Name' and 'Alias', and a 'POST' button. The interface also includes tabs for 'Raw data' and 'HTML form'.

Django REST framework

ben ▾

Api Root

Options GET ▾

GET /

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

```
{
  "students": "http://localhost:8000/students/"
}
```

Django REST framework

ben ▾

Api Root / Students List

Options GET ▾

Students List

GET /students/

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

```
[
  {
    "id": 1,
    "name": "John",
    "alias": "cool guy"
  },
  {
    "id": 2,
    "name": "Mary",
    "alias": "Playful"
  }
]
```

Raw data HTML form

Name

Alias

POST

Get student in id = 1

http://localhost:8000/students/1/

The screenshot shows a web browser window with the address bar set to `localhost:8000/students/1/`. The browser's taskbar at the top includes icons for various applications like a file explorer, a terminal, and social media. The page itself is the Django REST framework interface, displaying the 'Students Instance' detail view. The breadcrumb trail at the top reads 'Api Root / Students List / Students Instance'. On the right side of the page title, there are three buttons: 'DELETE' (red), 'OPTIONS' (blue), and 'GET' (blue with a dropdown arrow). Below the title, the HTTP method 'GET /students/1/' is shown. The main content area displays the response details: 'HTTP 200 OK', 'Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The JSON response body is shown in a light blue box:

```
{  "id": 1,  "name": "John",  "alias": "cool guy"}
```

. At the bottom of the page, there are two tabs: 'Raw data' (selected) and 'HTML form'. The 'HTML form' tab is active, showing a form with two input fields: 'Name' with the value 'John' and 'Alias' with the value 'cool guy'. A blue 'PUT' button is located at the bottom right of the form.

Django REST framework

ben ▾

Api Root / Students List / Students Instance

Students Instance

DELETE OPTIONS GET ▾

GET /students/1/

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

```
{  "id": 1,  "name": "John",  "alias": "cool guy"}
```

Raw data HTML form

Name John

Alias cool guy

PUT

Post a New student in database

Try it!

Try to view in Json format

Try to patch existing student

End of Lesson 7