

These slides are part of the downloadable resources of ***The Complete Guide to Django REST Framework and Vue JS*** Udemy course.


Let's keep in touch:

- ▶ YouTube Channel: https://www.youtube.com/channel/UCxPWtz5N--X3lyYJ13Zr99A?sub_confirmation=1
- ▶ Twitter: <https://www.twitter.com/pymike00>
- ▶ GitHub: <https://github.com/pymike00>



Django REST Framework

Level One - Objectives



DRF Level One - Objectives

- Make an introduction to the Package
- Learn how to integrate DRF into our projects
- Analyze all the Package's basic components with several practical examples

DRF Level One - Objectives

In the next sections we will then discuss increasingly advanced concepts such as authentication, automated testing and rapid development with Viewsets and Routers, up to the creation of a full fledged Single Page Application with Django REST Framework and Vue CLI in the final section of the course!

DRF Level One - What You Will Learn

- How to use Django REST Framework to create REST APIs
- Understand the concepts of Serialization and Deserialization
- How to use the Serializer and ModelSerializer classes
- How to create Function/Class Based API Views
- What theBrowsable API is and how to use it
- Define validation criteria for user input
- How to properly represent nested relationships between entities with DRF

Django REST Framework

Level One - Lessons

- DRF News API - Project Setup
 - Serializers
 - The @api_view Decorator
 - The APIView Class
 - Serializers Validation
 - The ModelSerializer Class
 - Nested Relationships
 - Competency Assessment and Project Solution
-



Django REST Framework

Introduction & Project Setup



DRF Level One - Introduction and Project Setup

What is Django REST Framework?

Django REST Framework is a powerful and flexible toolkit for building Web APIs with Python and Django

We can install it via pip and include it in any existing Django Project just by adding “rest_framework” to the INSTALLED_APPS list.

DRF Level One - Introduction and Project Setup

Django REST Framework allows us to create powerful production ready Web APIs in a very short time, with all the support and advantages of the Python and Django ecosystems and communities.

It's the number one package for creating REST APIs with Django, used by world class companies such as **Mozilla**, **Heroku**, **Red Hat** and **Eventbrite**.

DRF Level One - Introduction and Project Setup

Django REST Framework intentionally uses many of Django's programming conventions, making it a natural extension of the package that is easy to pick up and understand.

In this lesson we are going to create a new Django Project that we are then going to use throughout the section, learning how to use all of DRF's main components by creating a Web API for a news website.

Let's get started!

DRF Level One - Reference Links

<https://github.com/encode/django-rest-framework>

<https://www.django-rest-framework.org/>



Serializers



DRF Level One - Serializers

Serializers allow complex data such as querysets and model instances to be converted to native Python data types that can then be easily rendered into useful formats like JSON: this process is known as **Serialization** of Data.

Serializers are a very important component of DRF, that we can easily use by employing the **Serializer** and **ModelSerializer** classes.

Serializers also provide **deserialization**, allowing parsed data to be converted back into complex types, after first validating the incoming data.

DRF Level One - Serializers

In this lesson we are going to learn how to use the **Serializer** class to serialize/deserialize data from our Article model, that we've defined in the previous video.

We are also going to briefly talk about **Parsers** and **Renderers**.

Let's get started!

Serializers - Reference Links

<https://www.django-rest-framework.org/api-guide/serializers/>

<https://www.django-rest-framework.org/api-guide/fields/>

<https://www.django-rest-framework.org/api-guide/parsers/>

<https://www.django-rest-framework.org/api-guide/renderers/>



The @api_view decorator



DRF Level One - The `@api_view` decorator

Now that we have learned how to use Serializers to serialize and deserialize data, it's finally time to write our first **API View** to use them with!

Django REST Framework provides two *wrappers* we can use to write API Views:

- The **`@api_view`** decorator, for working with Function Based API Views
- The **`APIView`** class, for working with Class Based API Views

DRF Level One - The `@api_view` decorator

Because of these *wrappers*, our views will get for free all the code that is necessary to receive request instances, provide an appropriate response and handle exceptions such as the `ParseError` Exception that occurs when accessing `request.data` with malformed input.

In this lesson we are going to create two api views that will provide read & write functionalities for our articles, using the `@api_view` decorator.

DRF Level One - The @api_view decorator

We are also going to introduce one of Django REST Framework's most cherished and useful features, the **Browsable API**!

Let's get started!

The APIView Class

DRF Level One - The APIView Class

In the first lecture of this section we have mentioned that Django REST Framework uses many of Django's programming conventions.

One of the most striking examples is its use of Class Based Views.

As with “pure” Django, using this kind of pattern makes it easy and fast to create reliable Web Apps, and by reusing common functionalities it also helps us keep our code DRY (i.e. in line with the “Don't Repeat Yourself” principle of software development)

DRF Level One - The APIView Class

In this lesson we are going to start understanding how Class Based Views are used in the context of REST API development, by using DRF's **APIView** class.

Even though at a first glance the code we will write might look similar to the code we have written in the previous lecture, it is important to note that the APIView class is the class upon which all the Generic CBVs that we'll introduce later on are based, therefore it's important to make some examples of its use.

Let's get started!

The APIView Class - Reference Links

<https://www.django-rest-framework.org/api-guide/views/#class-based-views>

Serializers Validation

DRF Level One - Validation

So far we have seen how important it is to call the *is_valid()* method during the data deserialization step before accessing the *validated_data* dict or saving a new instance.

If any errors occur, the **.error** property will contain a dictionary with the corresponding error messages:

```
serializer = ContactSerializer(data={'body': 'some text', 'email': 'name#gmail.com'})
serializer.is_valid()
# False
serializer.errors
# {'email': [u'Enter a valid e-mail address.']}
```

DRF Level One - Validation

Learning about Validation, in this lesson we are going to see how to specify custom validation checks for our models and serializers using both Field-Level Validation and Object-Level Validation

Let's Get Started!

Validation - Reference Links

<https://www.django-rest-framework.org/api-guide/validators/>

<https://www.django-rest-framework.org/api-guide/serializers/#validation>



The ModelSerializer Class



DRF Level One - The ModelSerializer Class

In this lesson we are going to learn how to use the **ModelSerializer** class!

Similar to Django's ModelForm Class, ModelSerializer allows us to speed up the creation of Serializers based on models we defined, providing us all the code that is needed for the most common development use cases.

Let's Get Started!

The ModelSerializer Class - Reference Links

<https://www.django-rest-framework.org/api-guide/serializers/#modelserializer>

Nested Relationships

DRF Level One - Nested Relationships

In this lesson we are going to learn how to define **Nested Relationships** in our Serializer Classes!

As an example we are going to extend our News API project by creating a new Journalist model, that we are going to bind to our Article model with a ForeignKey Field, instead of the CharField that we are currently using!

DRF Level One - Nested Relationships

We are going to learn how to define, for each side of the relationship and for each instance, a list of primary keys or strings of the related objects, all the details of a specific instance or an automatically generated hyperlink to the corresponding endpoint.

We are also going to talk about when to actually make these relationships explicit by providing all the details of the instances involved, and when to opt for other design choices based on the development context instead.

Let's Get Started!

Nested Relationships - Reference Links

<https://www.django-rest-framework.org/api-guide/relations/>

<https://www.django-rest-framework.org/api-guide/relations/#api-reference>

<https://www.django-rest-framework.org/api-guide/relations/#nested-relationships>

<https://www.django-rest-framework.org/api-guide/relations/#hyperlinkedrelatedfield>



DRF Level One

Competency Assessment



DRF Level One - Competency Assessment

It's finally time to test your newly acquired skills with a new project!

Remember that you can download all the code written so far and the slides, from the first lesson of the section.

DRF Level One - Competency Assessment

The test consists in creating a Web API Project for a **JobBoard** website. Similar to indeed.com, companies will be able to create and publish new job offers that people will then be able to see.

Clients will be able to communicate with our Web API from 2 URL Endpoints:

1. **api/jobs/** - accepts GET and POST methods, allowing to create new instances and retrieve a list with all the available job offer instances
2. **api/jobs/<int:pk>/** - accepts GET, PUT and DELETE, allowing a user to retrieve, update or delete an object instance.

DRF Level One - Competency Assessment

You can choose if you want to write the project views using the `@api_view` decorator or the `APIView` class, and you can also choose to write the necessary `Serializer` using the homonym class or the `ModelSerializer` class instead.

What really matters is for you to have understood the respective advantages and disadvantages of the different options available... everything else is about your coding style and preferences!

You will not need to build an authentication system for this specific API, as that is a more advanced topic we will discuss together in the next sections of the course!

DRF Level One - Competency Assessment

Only one model is needed for the project, you can call it **JobOffer**.

It must have the following fields:

- company_name
- company_email
- job_title
- job_description
- salary
- city
- state
- created_at
- available

DRF Level One - Competency Assessment

In the next lesson we will see together in detail how to write one of the possible solutions for this test.

Happy Coding!



DRF Level One

Competency Assessment



DRF Level One - Project Solution

You can download all the code for the project from the downloadable resources of this section, and from my GitHub Profile:

<https://github.com/pymike00>