Building REST APIs with Python

ด้วยความต้องการที่เพิ่มขึ้นสำหรับนักพัฒนาแบบ Full-Stack มีความจำเป็นที่จะต้องเรียนรู้การผสมผสานข้อดีของภาษา python กับเหล่า library ยอดนิยมทั้งหลายมาสรรค์สร้างแอพพลิเคชั่นให้น่าสนใจยิ่งขึ้น

Django เป็นหนึ่งใน framework ที่สนับสนุนการพัฒนาส่วน front-end และรองรับการทำงานฝั่ง backend ได้ด้วย ในหลักสูตรนี้จะนำเสนอการใช้งาน Python ออกแบบและพัฒนา Rest API ด้วยภาษา Python ด้วยทำงานร่วมกับ ReactJS

ในหลักสูตรเลือกใช้ฐานข้อมูล Postgres มาอธิบายการกำหนดค่าระหว่าง Server และ Client เพื่อให้เห็นภาพการทำงานของ API และนำไปประยุกต์กับฐานข้อมูลอื่น ๆ ของผู้เรียนได้ต่อไป

วัตถุประสงค์:

- Set up your own application project in Django
- Organize your Django project's development and productions environment
- Write Django views and use routes to handle incoming requests
- Create your own Django templates for your Python web API and learn to use template filters and tags
- Work with RESTful APIs in Django
- Quickly build clean APIs with the Django REST Framework

กลุ่มเป้าหมาย:

- นักเรียนนักศึกษา
- ครู อาจารย์ วิทยากรที่สนใจ
- นักวิชาการ นักไอที หรือผู้ดูและระบบ
- ตลอดจนผู้สนใจทั่วไปในการพัฒนา REST APIs with Python

ความรู้พื้นฐาน:

- พื้นฐานภาษา Python
- พื้นฐานการใช้งาน Django framework
- พื้นฐานการใช้งานฐานข้อมูล PostgreSQL
- พื้นฐานการใช้งาน Windows and MacOS

ระยะเวลาในการอบรม:

- 12 ชั่วโมง (2 วัน)

ราคาคอร์สอบรม:

- 6,500 บาท / คน (ราคานี้ยังไม่ได้รวมภาษีมูลค่าเพิ่ม)

วิทยากรผู้สอน:

- อาจารย์สามิตร โกยม

คอร์สที่ควรอบรมก่อนหน้า:

- หลักสูตร Python Basic

เนื้อหาการอบรม:

Module 1: Getting Started

The Bigger Picture

- Look at HTML response
- Look at REST API server

Your Development Environment

- Install Python 3 based on the Operating System of our computer. Create a Virual Environment
- Use Pip to install Django and the Django REST framework
- Use pip requirements.txt to identify installed python components in a development environment

Installing PostgreSQL

- Install PostgreSQL into our development environment
- Use Psql to check our installation and create a new database
- Use pip to install PostgreSQL support into Python

Django Projects and Apps

- Create a project directory and use django-admin startproject
- Use manage.py startapp to create a Django application
- Review Django external application publishing guidelines

Using the Django Development Environment

- Introduce settings.py and configuring our database
- Highlight the DEBUG flag and placing sensitive information in settings.py
- Use Django's development webserver via the manage.py runserver command and seeing the Django initial default page

Module 2: The MVT Framework Approach

MVC and MVT Framework

- Look at the concept of MVC and MVT Framework
- Look at Django models
- Look at Django template

Creating and Working with Models

- Introduce Django model and how relationships between Models can be used
- Create a Django model
- Create methods on Django models and test them

Migrations and Database Queries

- Consider what Django migrations are and have Django create schema migrations

- Consider what data migrations are and create one, and then apply all migrations created so far in this video
- Perform some simple queries against our database

Writing Our First View

- Discuss Django class based views and use view to create a simple view that returns a hard coded response
- Perform URL configuration or routing at both the Project and App level to make our new view accessible. Access the view in a browser
- Make use of URL querystring and kwarg parameters. Discuss the pros and cons of each approach

Routing and HTTP Methods

- Discuss how Django handles requests from the request in to middleware, then URL config, then view and then the reverse for the response
- Review URL configuration from previous video and discuss use of path() and re_path()
- Cover methods (verbs) of Http Requests including Get and Post requests. How Http methods then translate to class methods

Using Templates

- Explain how hardcoded responses are undesirable. Introduce Django's TemplateView
- Learn about using base templates and view specific templates that extend base templates. Learn how to make data available within templates.
- Review template settings in settings.py and see our first set of templates in action
- Explain how templates should not perform core logic, however we can use template tags and filters to perform presentation logic

Testing

- Learn how we need to use automated testing tools and how we can write test cases that place our code under test

- Make use of pqsl to configure PostgreSQL for testing
- Develop Integration tests that make use of the Django test client
- Develop unit tests that test specific view functions using a RequestFactory as well as class attributes

Module 3: Building Your Django RESTful API

Exploring RESTful APIs

- Explain the difference between traditional web responses and API responses. Learn how REST provides an API on HTTP
- Learn what Resources, HTTP Methods and CRUD operations are
- Explain how a result of an API request is communicated via a status code

Writing a Simple Hello World API

- Make use of a URL pattern configuration to give access to our API view
- Define a View using the base View class that uses JsonResponse to return an API response
- Include in our View definition validation of a name parameter and return an error if this is equal to fred. Show results in a browser

Exploring the DRF

- Learn how the DRF helps us in providing code that we would otherwise have to provide ourselves many times over
- Learn what the key parts of the DRF are, including serializers, DRF views, routers and authentication and permissioning. Learn how these relate to CRUD operations
- Explore who uses the DRF and see the web browsable feature of the DRF

Serializing Models

- Review our existing models with focus on the Bookmark Model, then we will define a serializer with create and update methods
- Using the Python console, we will create some initial data in the database and then demonstrate what our serializers do with this data along with simulated received data

- We will define a serializer based on Bookmark using a ModelSerializer
- We will then demonstrate how this serializer can be used with a list of Bookmarks in the Python console

Refactoring Our API with the DRF

- Learn about basic DRF view URL configuration, including format suffixes
- Define a Bookmark List view that can build a list and create new Bookmarks
- Learn about basic DRF detail view URL configuration. Define a Bookmark detail view that can get detail, update a Bookmark and delete a Bookmark
- Watch a demonstration of all of the functionality previously developed in this video, including listing existing Bookmarks, creating new Bookmarks, updating existing Bookmarks and deleting them

Generic Django REST Framework Views

- Make use of ListCreate-APIView for Bookmark listing and new Bookmark creation
- Make use of RetriveUpdate-APIView for Bookmark detail queries, updating a Bookmark and deleting a Bookmark
- Take advantage of the ModelViewset that combines all the functionality seen earlier in this video in one convenient package

Extra Viewset Actions and Routers

- Learn about Hyperlinked-Model-Serializers and how they can create related links in our API. Significantly simplify our URL configuration with a DRF router
- Create a specialised viewset action that can add a like to a Bookmark. Learn how to define this in the viewset and ensure the DRF router is aware of it. See how it is accessed from the browser
- Consider each of the view strategies covered in this and previous videos and scenarios where each would be the best fit

Testing the API

- Learn how to write an integration test of a DRF based viewset and place different actions under test
- Learn how to write a unit test of specific methods of a DRF based viewset, how to isolate dependencies via patching and using mocks to test each operation of the code
- See how the Django test runner is used in combination with DRF test tools