

Django and REST Framework with Introductory Machine Learning

Build a small web/API application and integrate a supplied OCR example

FA-0728 · 5 days

Software Development · Data & Analytics · DevOps, Cloud & Infrastructure

Introduction

Use basic Python skills to develop a small Django website and API, then integrate a supplied OCR or image-processing example. Follow the application from environment setup and database-backed pages through REST interfaces and a controlled deployment exercise.

The focus is connecting existing components and understanding their constraints, not training an advanced vision model or mastering production operations. Accounts, infrastructure and network permissions for any remote lab are arranged separately and must match the supplied environment.

Learning outcomes

- Create a Django project with templates, models, database access and a tailored admin interface.
- Handle users, errors, static/media data and selected application interactions.
- Integrate and evaluate a supplied image/OCR component and manage compatible dependencies.
- Expose selected functionality through Django REST Framework serializers and endpoints.
- Distinguish authentication from permissions and apply appropriate session/token controls.
- Explain WSGI/ASGI and deployment requirements, including a reverse proxy, HTTPS and operational settings.

Prerequisites

Basic programming knowledge in python. If the learner doesn't know python 3, an additional day for a python crash course can be added.

Participants should also be comfortable with basic HTML, files and command-line use. An optional preparatory Python day assumes existing general programming knowledge.

A compatible supported Django/Python/REST Framework stack, database and supplied OCR dependencies; approved local or isolated remote lab access.

Training outline

1. Day 1 — Django environment and request flow

- Installing Django
- Managing environments
- Introductory developments
- Passing data
- Templates

Create a small project, trace URLs/views/templates and pass sample data through a request.

2. Day 2 — Models, admin and presentation

- Build on the supplied project structure and a modest front-end template.
- Models, database configuration and migrations; tailor the Django admin without treating it as the complete public application.
- Manage static files and distinguish them from uploaded media; verify a sample database-backed page.

3. Day 3 — Users, errors and an OCR integration

- Handle selected errors and validate inputs; use appropriate user and model permissions.
- Connect a supplied OCR/image-processing component, parse uploaded binary/image data and check the output against sample evidence.
- Manage separate runtime requirements or dependency constraints where needed; compare an in-process call with a separately hosted inference service.
- Review image limits, data handling, runtime cost and model errors; no custom-model training or accuracy guarantee is implied.

4. Day 4 — REST APIs and access controls

- Expose selected application functionality through serializers, views and routes; distinguish a headless API from removal of the application's data/security model.
- Use curl or an approved REST client to send/receive sample requests and consume a supplied API.
- Apply authentication and permission policies separately; inspect session/CSRF versus token-based behaviour.
- Validate inputs and inspect success, unauthenticated and forbidden cases.
- Timestamped or refreshed data does not by itself create a real-time streaming architecture; discuss event/async requirements where relevant.

5. Day 5 — Controlled deployment and operations review

- Resolve compatible dependencies, protect secrets and configure environment-specific settings.
- Review a production-suitable WSGI or ASGI server; the development server is not the production service.
- Inspect a supplied reverse-proxy and HTTPS deployment, static/media handling and process lifecycle.
- Use Django deployment checks, logs and selected restart/failure observations in the isolated lab.
- Document remaining reliability, performance, security and model-monitoring work before any operational use.