BLACKBIRD FOR TRAINING

Intermediate Python and Django Training Course: Backend, REST APIs and Performance

20 – 24 September 2027
London (UK)

Intermediate Python and Django Training Course: Backend, REST APIs and Performance

Ref: 31_361603 **Date:** 20 – 24 September 2027 **Location:** London (UK) **Fees:** 6100 GBP

Introduction:

Intermediate Python and Django training covers backend development with Django, the Python web framework, and Django REST Framework, its toolkit for building REST APIs, the web interfaces that exchange JSON data over HTTP. This 1-week course is for software developers, backend and web developers and Python programmers who build or maintain web applications, and its final deliverable is a tested, documented Django REST API with a performance tuning report. It is taught at intermediate practitioner level through hands-on labs built around real-world project scenarios, from data modelling and Celery background tasks to pytest testing and load testing.

Course Objectives:

- Apply Python functions, classes, virtual environments and HTTP conventions to build Django REST Framework endpoints that return correct status codes and JSON
- Distinguish primary keys from foreign keys and select the relationship types, SQL joins and nested serializers that fit a relational data model
- Apply business logic, Redis message queues and Celery background tasks, and distinguish Django signals from decorators when handling application events
- Compare unit tests with integration tests and apply pytest, Django testing tools and factory_boy to build a repeatable test suite
- Interpret Swagger/OpenAPI documentation and ORM query output to select caching and database optimisations for slow endpoints
- Justify performance tuning decisions with load testing results recorded in an API performance tuning report

Target Audience:

- Backend developers who build and extend Django services, deciding where business logic and background tasks belong
- Web developers who expose data to front-end and mobile clients, choosing how API endpoints, serializers and status codes are designed
- Python programmers moving into web development, judging which Django and Django REST Framework patterns fit a new project
- Software engineers who maintain existing web applications, deciding which slow queries, caches and endpoints to fix first
- IT professionals who support and test web applications and APIs, judging when test coverage and API documentation are ready for release

BLACKBIRD FOR TRAINING

Course Outline:

Day 1: Core Django and REST API Development

- Python Functions vs Classes: Structuring Reusable Backend Code for Django Projects
- Virtual Environments and Package Management: venv, pip and Requirements Files Explained
- HTTP Methods and Status Codes: GET, POST, PUT, PATCH and DELETE in APIs
- Request/Response Lifecycle and JSON: How Django Middleware and Views Process Calls
- Django REST Framework Fundamentals: APIView, ViewSets, Routers and First API Endpoints

Day 2: Database Modelling and Nested APIs

- Relational Database Design: Tables, Normalisation and Entity Relationships for Web Applications
- Primary Keys vs Foreign Keys: One-to-Many and Many-to-Many Relationship Modelling
- SQL Fundamentals: SELECT Queries, INNER JOIN and LEFT JOIN on Related Tables
- Django Models and ModelSerializer: Mapping Database Tables to Validated API Payloads
- Nested Serializers: Writable Nested Data and Related API Create and Update Operations

Day 3: Business Logic and Asynchronous Processing

- Business Logic in Django: Service Layers, Model Methods and Validation Rules
- Redis Fundamentals and Message Queues: Brokers, Keys and Data Structures for Tasks
- Background Task Processing with Celery: Workers, Retries and Periodic Tasks with Beat
- Django Signals vs Decorators: Decoupled Event Handling and Reusable View Behaviour
- Asynchronous Programming and Notification Systems: Async Views and Email or Push Alerts

Day 4: Testing and API Documentation

- Unit Tests vs Integration Tests: Scope, Isolation and Coverage in Django APIs
- pytest and pytest-django: Fixtures, Parametrisation and Database Access in Test Runs
- Django Testing Tools: TestCase, APIClient and APITestCase for Endpoint Checks
- Test Data Management with factory_boy: Factories, SubFactory and Realistic Fixture Data
- API Documentation with Swagger/OpenAPI: drf-spectacular Schemas and Interactive Endpoint Testing

Day 5: Lab: Performance Optimisation and Scalability

- select_related vs prefetch_related: Advanced Django ORM Usage Against N+1 Query Problems
- Query Optimisation with Django Debug Toolbar: Profiling, Annotations and Database Indexes
- Caching Strategies: Per-View, Template Fragment and Low-Level Caching with Redis
- Database Performance Considerations: Connection Pooling, Pagination and Bulk Operations
- Load Testing with Locust and API Performance Optimisation: Completing the Performance Tuning Report

Skills You Will Gain:

- REST API Design
- Relational Data Modelling
- Nested Serializer Implementation
- Asynchronous Task Processing
- Automated API Testing
- OpenAPI Documentation
- ORM Query Profiling
- Load Testing Analysis

Why Attend This Course:

- From writing ad hoc Django views to structuring Django REST Framework endpoints with consistent status codes, serializers and routers
- From running slow work inside web requests to moving it to Celery workers with Redis as the message broker
- From manual endpoint checks to an automated pytest suite with factory_boy data and Swagger/OpenAPI documentation
- From guessing at slow pages to a tested, documented Django REST API with a performance tuning report backed by load testing data

Conclusion:

Python and Django backend work comes down to turning data models into reliable, well-tested web services that stay fast as traffic grows. The course makes three distinctions clear: primary keys against foreign keys in relational modelling, Django signals against decorators when reacting to events, and unit tests against integration tests in a Django REST Framework project. It is the right choice for developers and Python programmers who already write basic Python and now need practitioner-level skill in building, testing and optimising Django APIs, rather than an introductory web course or an architecture-level course on distributed systems.

Frequently Asked Questions FAQ:

What should a delegate know before an intermediate Python and Django training course?

Delegates need working knowledge of basic Python syntax and some experience writing code for web applications. Python functions, classes and virtual environments are refreshed on the first day, so no previous Django REST Framework, Celery or pytest experience is assumed.

How does an intermediate Python and Django training course differ from an introductory web development course?

An introductory course teaches templates, forms and first pages. This intermediate course focuses on backend work: REST APIs with Django REST Framework, relational modelling, background tasks with Celery and Redis, automated testing and performance optimisation of existing applications.

What is the difference between select_related and prefetch_related in Python and Django performance work?

select_related follows foreign key and one-to-one relationships with an SQL JOIN in a single query, while prefetch_related runs a separate query for many-to-many and reverse relationships and joins the results in Python. Both remove the N+1 query problem on the right relationship type.

What does a delegate take back to work from an intermediate Python and Django training course?

Delegates take back a tested, documented Django REST API project with a pytest and factory_boy test suite, Swagger/OpenAPI documentation and a performance tuning report built from load testing results, ready to reuse as a reference for their own applications.

UK Traininig
PARTNER