# Django - Practice Sheet

10 Q&A on MTV, ORM, views, authentication, and REST framework



**Q1. What is Django's MTV architecture and how does it map to MVC?**

**A:** Model handles data and business logic. Template renders HTML. View acts as the controller  it processes requests, queries the Model, and returns a Template response. The URL dispatcher maps URL patterns to View functions or class-based views.

**Q2. How does Django's ORM work and how do you create a migration?**

**A:** Define model classes with fields (CharField, IntegerField, ForeignKey). Run python manage.py makemigrations to generate migration files from model changes, then migrate to apply them. Django tracks applied migrations in the django_migrations table.

**Q3. What is the difference between an FBV and a CBV in Django?**

**A:** A Function-Based View is a plain Python function taking a request and returning a response  simple and explicit. A Class-Based View inherits from View (or ListView, CreateView etc.) and overrides get()/post() methods, enabling reuse via mixins.

**Q4. How does Django URL routing resolve a request to a view?**

**A:** Django iterates urlpatterns in the root urls.py, testing each path() or re_path() pattern. include() delegates to a nested urls.py. Named groups or path converters (<int:pk>) capture parts of the URL and pass them as kwargs to the view.

**Q5. What is Django REST Framework (DRF) and what does a serializer do?**

**A:** DRF adds API tools on top of Django. A Serializer converts model instances to Python dicts (for JSON responses) and validates incoming dicts back to model instances. ModelSerializer auto-generates fields from the model definition.

**Q6. How does Django's permission system work?**

**A:** Django auto-creates add, change, delete, view permissions for every model. @login_required or LoginRequiredMixin enforces authentication. @permission_required('app.can_publish') or PermissionRequiredMixin checks named permissions stored in the auth_permission table.

**Q7. What is Django middleware and how does the request/response cycle work?**

**A:** Middleware classes wrap the view. process_request runs before the view (returning a response short-circuits the rest). process_response runs after. Built-in middleware handles sessions, CSRF tokens, authentication, and security headers.

**Q8. What is template inheritance in Django and how does it work?**

**A:** A base template defines named {% block %} areas. Child templates extend it with {% extends 'base.html' %} and override blocks with their own content. The template engine replaces blocks at render time, enabling DRY layouts across pages.

**Q9. How does Django caching work and what backends are available?**

**A:** Django's cache framework uses a unified API (cache.get/set/delete). Backends include in-memory (LocMemCache), Redis (django-redis), Memcached, and database. @cache_page(60*15) caches entire view responses; template fragment caching and low-level API give finer

**Q10. How do you write tests in Django?**

**A:** Subclass django.test.TestCase (wraps each test in a transaction that rolls back). Use self.client.get('/url/') to simulate HTTP requests. APIClient (DRF) sends JSON requests. Use factories or fixtures to create test data. Run with python manage.py test.