Django Cheatsheet

Project Setup and Management

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

- Installation: `pip install django`.
- Create Project: `django-admin startproject myproject`.
- Create App: `python manage.py startapp myapp`.
- Run Development Server: `python manage.py runserver`.
- Management Commands:
  - `python manage.py createsuperuser`: Create admin user.
  - `python manage.py makemigrations`: Generate migration files for model changes.
  - `python manage.py migrate`: Apply migrations to database.
  - `python manage.py shell`: Enter Django-aware Python shell.

Models (Database Layer)

Define your data structure in `models.py`.

```
from django.db import models

class Article(models.Model):
    title =
models.CharField(max_length=200)
    content = models.TextField()
    pub_date =
models.DateTimeField('date published')

    def __str__(self):
        return self.title
```

Views and URLs

URL Configuration (urls.py)

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('article/<int:article_id>/',
views.detail, name='detail'),
]
```

Views (views.py)

```
from django.shortcuts import render,
get_object_or_404
from .models import Article

def detail(request, article_id):
    article = get_object_or_404(Article,
pk=article_id)
    return render(request, 'myapp/
detail.html', {'article': article})
```

Django Templates (DTL)

Dynamic HTML generation.

- Variables: `{{ article.title }}`.
- Tags: `{% if ... %}`, `{% for item in items %}`, `{% url 'detail' article.id %}`.
- Inheritance:
  - `{% extends 'base.html' %}`
  - `{% block content %} ... {% endblock %}`

Django Admin

- Register model:

```
from django.contrib import admin
from .models import Article

admin.site.register(Article)
```

- Access at: `/admin/` after running the server.

Forms

Handling user input safely.

```
from django import forms

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        fields = ['title', 'content']
```

REST Framework (DRF)

Standard toolkit for building Web APIs.

- Serializer: `class ArticleSerializer(serializers.ModelSerializer): ...`
- ViewSet: `class ArticleViewSet(viewsets.ModelViewSet): ...`

Middleware and Signals

- Middleware: Hook into Request/Response processing.
- Signals: Decouple apps by allowing certain senders to notify recipients when actions occur (e.g., `post_save`).

Pro Tips

Django Debug Toolbar

A configurable set of panels that display various debug information about the current request/response. Highly recommended for development.

QuerySet Optimization

- Use `select_related()` for foreign key relationships (joins).
- Use `prefetch_related()` for many-to-many or reverse foreign keys.
- Avoid N+1 query problems.

Security Settings

Always set `DEBUG = False` and configure `ALLOWED_HOSTS` in production settings. Use `python manage.py check --deploy`.