Cheat Sheet for comprensive Django

Django Cheat Sheet

1. Getting Started

Installation

- **Install Django**: `pip install django`
- **Create a Project**: `django-admin startproject projectname`
- **Run Server**: `python manage.py runserver`

Project Structure

```
projectname/
    manage.py
    projectname/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

---

2. Project Setup

Settings (`settings.py`)

- **INSTALLED_APPS**: Add your apps here.
- **DATABASES**: Configure your database.
- **TEMPLATES**: Define template directories.
- **STATIC_URL**: URL for static files.
- **MEDIA_URL**: URL for user-uploaded files.

Environment Variables

- Use `django-environ` for managing environment variables.

---

3. Models

Defining Models

```
from django.db import models

class MyModel(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
```

Field Types

- **CharField**: `models.CharField(max_length=100)`
- **TextField**: `models.TextField()`
- **IntegerField**: `models.IntegerField()`
- **DateTimeField**: `models.DateTimeField(auto_now_add=True)`
- **ForeignKey**: `models.ForeignKey(OtherModel, on_delete=models.CASCADE)`
- **ManyToManyField**: `models.ManyToManyField(OtherModel)`

Model Methods

- **`__str__`**: Define the string representation.
- **`get_absolute_url`**: Define the URL for the model instance.

Meta Options

```
class Meta:
    ordering = ['-created_at']
    verbose_name_plural = 'My Models'
```

---

4. Views

Function-Based Views

```
from django.http import HttpResponse

def my_view(request):
    return HttpResponse("Hello, World!")
```

Class-Based Views

```
from django.views.generic import ListView, DetailView
from .models import MyModel

class MyModelListView(ListView):
    model = MyModel
    template_name = 'mymodel_list.html'

class MyModelDetailView(DetailView):
    model = MyModel
    template_name = 'mymodel_detail.html'
```

View Decorators

- **@login_required**: `from django.contrib.auth.decorators import login_required`
- **@csrf_exempt**: `from django.views.decorators.csrf import csrf_exempt`

---

5. Templates

Template Structure

```
{% extends "base.html" %}

{% block content %}
    <h1>{{ object.name }}</h1>
    <p>{{ object.description }}</p>
{% endblock %}
```

Template Tags

- **`{% for %}`**: Loop through a list.
- **`{% if %}`**: Conditional statement.
- **`{% url %}`**: Generate URLs.
- **`{% include %}`**: Include another template.

Template Filters

- **`{{ value|default:"nothing" }}`**: Set a default value.
- **`{{ value|length }}`**: Get the length of a list.
- **`{{ value|date:"F j, Y" }}`**: Format dates.

---

6. Forms

Defining Forms

```
from django import forms

class MyForm(forms.Form):
    name = forms.CharField(max_length=100)
    email = forms.EmailField()
```

Model Forms

```
from django.forms import ModelForm
from .models import MyModel

class MyModelForm(ModelForm):
    class Meta:
        model = MyModel
        fields = ['name', 'description']
```

Form Rendering

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
</form>
```

---

7. Admin

Registering Models

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

admin.site.register(MyModel)
```

Custom Admin Classes

```
class MyModelAdmin(admin.ModelAdmin):
    list_display = ('name', 'created_at')
    search_fields = ('name',)
```

---

8. URLs

URL Patterns

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

urlpatterns = [
    path('my-view/', my_view, name='my_view'),
]
```

URL Namespaces

```
app_name = 'myapp'

urlpatterns = [
    path('my-view/', my_view, name='my_view'),
]
```

Including URLs

```
from django.urls import include, path

urlpatterns = [
    path('myapp/', include('myapp.urls')),
]
```

---

9. Migrations

Creating Migrations

- **Create**: `python manage.py makemigrations`
- **Apply**: `python manage.py migrate`

Custom Migration Operations

```
from django.db import migrations

def forwards_func(apps, schema_editor):
    # Migration code
    pass
```

```
class Migration(migrations.Migration):
    operations = [
        migrations.RunPython(forwards_func),
    ]
```

---

10. Testing

Writing Tests

```
from django.test import TestCase
from .models import MyModel

class MyModelTests(TestCase):
    def test_model_creation(self):
        obj = MyModel.objects.create(name="Test", description="Test
Description")
        self.assertEqual(obj.name, "Test")
```

Running Tests

- **Run All Tests**: `python manage.py test`
- **Run Specific Test**: `python manage.py test myapp.MyModelTests`

---

11. Static Files

Serving Static Files

- **Collect Static Files**: `python manage.py collectstatic`
- **Template Tag**: `{% load static %}`
- **Usage**: `<img src="{% static 'myapp/image.jpg' %}" alt="My image">`

Media Files

- **Upload to Media**: `MEDIA_URL` and `MEDIA_ROOT` in `settings.py`
- **Template Tag**: `{{ MEDIA_URL }}`

---

12. Sessions and Authentication

User Authentication

- **Login**: `from django.contrib.auth import authenticate, login`
- **Logout**: `from django.contrib.auth import logout`

Permissions

- **Check Permissions**: `user.has_perm('myapp.can_edit')`

Custom User Model

- **Extend AbstractUser**: `from django.contrib.auth.models import AbstractUser`

---

13. Middleware

Custom Middleware

```
class MyMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response

    def __call__(self, request):
        # Code to be executed before the view
        response = self.get_response(request)
        # Code to be executed after the view
        return response
```

Adding Middleware

- **Add to `MIDDLEWARE` in `settings.py`**: ``'myapp.middleware.MyMiddleware'``

---

14. Signals

Defining Signals

```
from django.db.models.signals import post_save
from django.dispatch import receiver
from .models import MyModel

@receiver(post_save, sender=MyModel)
def my_handler(sender, instance, created, **kwargs):
    if created:
        # Do something
        pass
```

Connecting Signals

- **In `apps.py`**: `from django.apps import AppConfig`
- **Override `ready` method**: `def ready(self): import myapp.signals`

---

15. Management Commands

Creating Commands

```
from django.core.management.base import BaseCommand

class Command(BaseCommand):
    help = 'My custom command'

    def handle(self, *args, **kwargs):
        self.stdout.write('Command executed successfully')
```

Running Commands

- **Run Command**: `python manage.py my_custom_command`

---

16. Deployment

WSGI/ASGI

- **WSGI**: `wsgi.py`
- **ASGI**: `asgi.py`

Deployment Checklist

- **Set Debug to False**: `DEBUG = False`
- **Set Allowed Hosts**: `ALLOWED_HOSTS = ['yourdomain.com']`
- **Set Static and Media URLs**: `STATIC_URL` and `MEDIA_URL`

Deployment Tools

- **Gunicorn**: WSGI HTTP Server
- **Daphne**: ASGI HTTP Server
- **Nginx**: Reverse Proxy

---

17. Advanced Topics

Custom Managers

```
class MyModelManager(models.Manager):
    def get_queryset(self):
        return super().get_queryset().filter(active=True)

class MyModel(models.Model):
    objects = MyModelManager()
```

QuerySets

- **Filter**: `MyModel.objects.filter(name="Test")`
- **Exclude**: `MyModel.objects.exclude(name="Test")`
- **Order By**: `MyModel.objects.order_by('-created_at')`

Caching

- **Simple Cache**: `from django.core.cache import cache`
- **Cache Decorator**: `from django.views.decorators.cache import cache_page`

Internationalization (i18n)

- **Enable i18n**: `USE_I18N = True`
- **Translate Strings**: `{% trans "Hello" %}`

This cheat sheet provides a comprehensive overview of Django, covering essential features, shortcuts, tips, and tricks. Use it as a quick reference guide for your Django projects.

By Ahmed Baheeg Khorshid

ver 1.0