MTV Django Architecture

Tim Dosen PBP

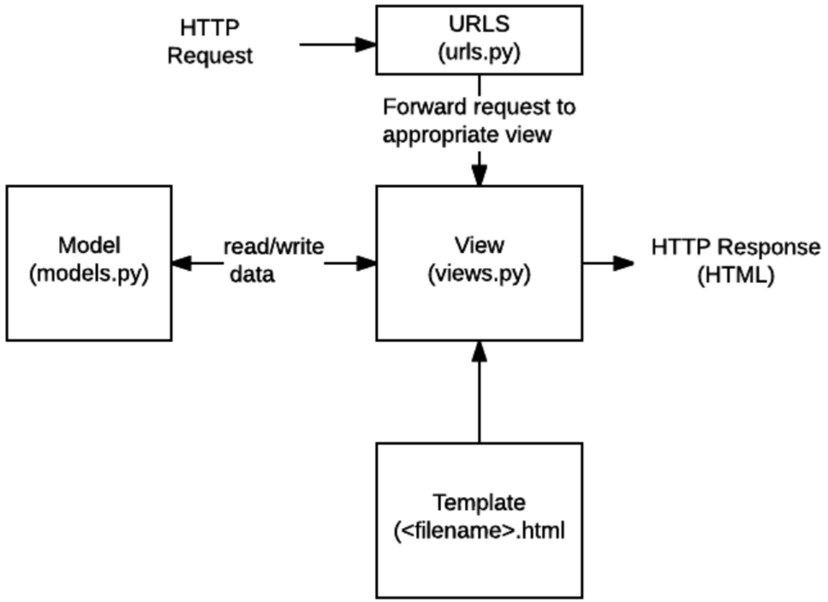
Question

Separation of Concern: Why we need it? What are the benefits of it?

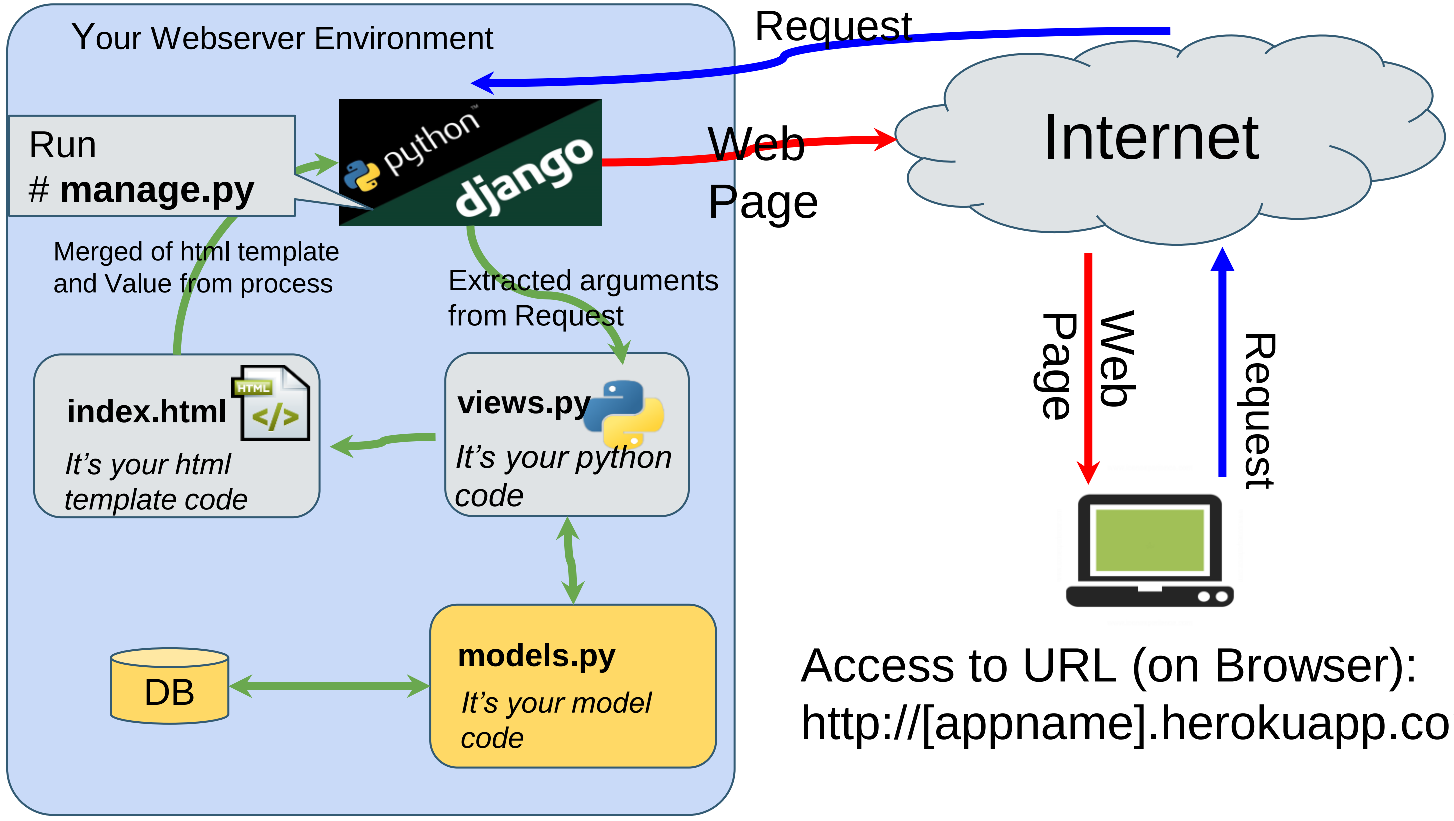
Django MTV Architecture

Django is often referred to as an MTV framework

- **M stands for “Model”**: the data access layer
- **T stands for “Template**”: the presentation layer
- **V stands for “View”:** the business logic layer, the bridge between models and templates.



How Django Framework Works



Access to URL (on Browser):
http://[appname].herokuapp.com/

Installing Django

- https://docs.djangoproject.com/en/4.1/intro/install/
- https://docs.python.org/3/library/venv.html

1. Install Python
   "Being a Python web framework, Django requires Python"
2. Install and activate a Python Virtual Environment

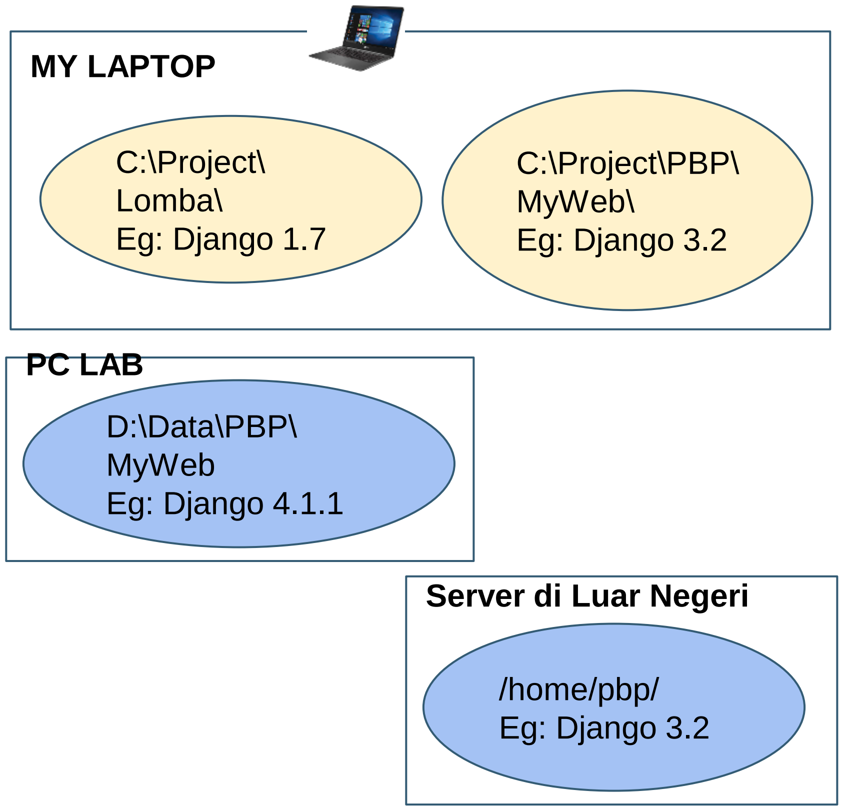
```
python -m venv env
env\Scripts\activate.bat
```

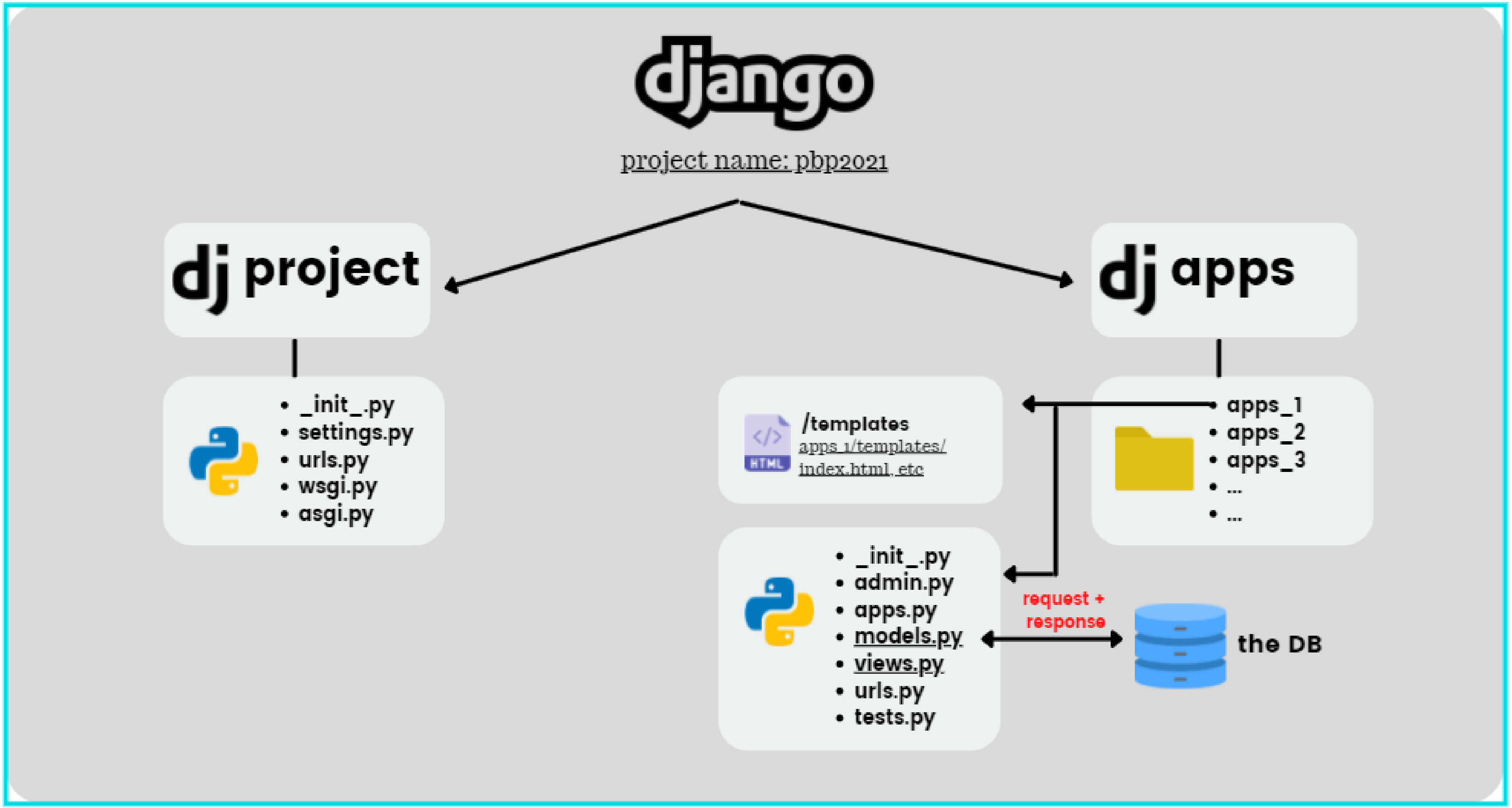
3. Install Django and/or required dependencies

```
pip install django
```

Or

```
pip install -r requirements.txt
```


MY LAPTOP
C:\Project\ Lomba\ Eg: Django 1.7
C:\Project\PBP\ MyWeb\ Eg: Django 3.2
PC LAB
D:\Data\PBP\ MyWeb Eg: Django 4.1.1
Server di Luar Negeri
/home/pbp/ Eg: Django 3.2


django
project name: pbp2021
dj project
init.py
settings.py
urls.py
wsgi.py
asgi.py
dj apps
apps_1
apps_2
apps_3
...
...
/templates
apps_1/templates/
index.html, etc
HTML
init.py
admin.py
apps.py
models.py
views.py
urls.py
tests.py
request +
response
the DB

Question
What is the difference between Django project and Django app?

Django Project vs Django App

- https://docs.djangoproject.com/en/4.1/intro/tutorial01/

- **Django Project**
  - A Django project – a collection of settings for an instance of Django, including database configuration, Django-specific options and application-specific settings.
  - A project is a collection of configuration and apps for a particular website.
  - A project can contain multiple apps.
- **Django App**
  - An app is a web application that does something – e.g., a blog system, a database of public records or a small poll app.
  - An app can be in multiple projects.

Django Project

- Start a project

```
django-admin startproject contohproject
```

- Auto-generated files:

```
contohproject/
    contohproject/
        __init__.py
        asgi.py
        settings.py      (configuration for the project)
        urls.py          (starting point to configure urls)
        wsgi.py
    manage.py            (command runner)
```

- Other django-admin command: https://docs.djangoproject.com/en/4.1/ref/django-admin/

Django Development Server

- Start up a development server

```
cd contohproject
python manage.py runserver
```

- The Django development server is meant to be used only during development so you can develop things rapidly, without having to deal with configuring a production server – such as Apache – until you’re ready for production.
- By default, the runserver command starts the development server on the internal IP at port 8000.
- After the server’s running, visit http://127.0.0.1:8000/ or http://localhost:8000/ with your web browser.

Django App

- Build an app

```
django-admin startapp contohapp
```

- The directory structure of the newly created `contohapp` application:

```
contohapp/
    migrations/
        __init__.py
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
```

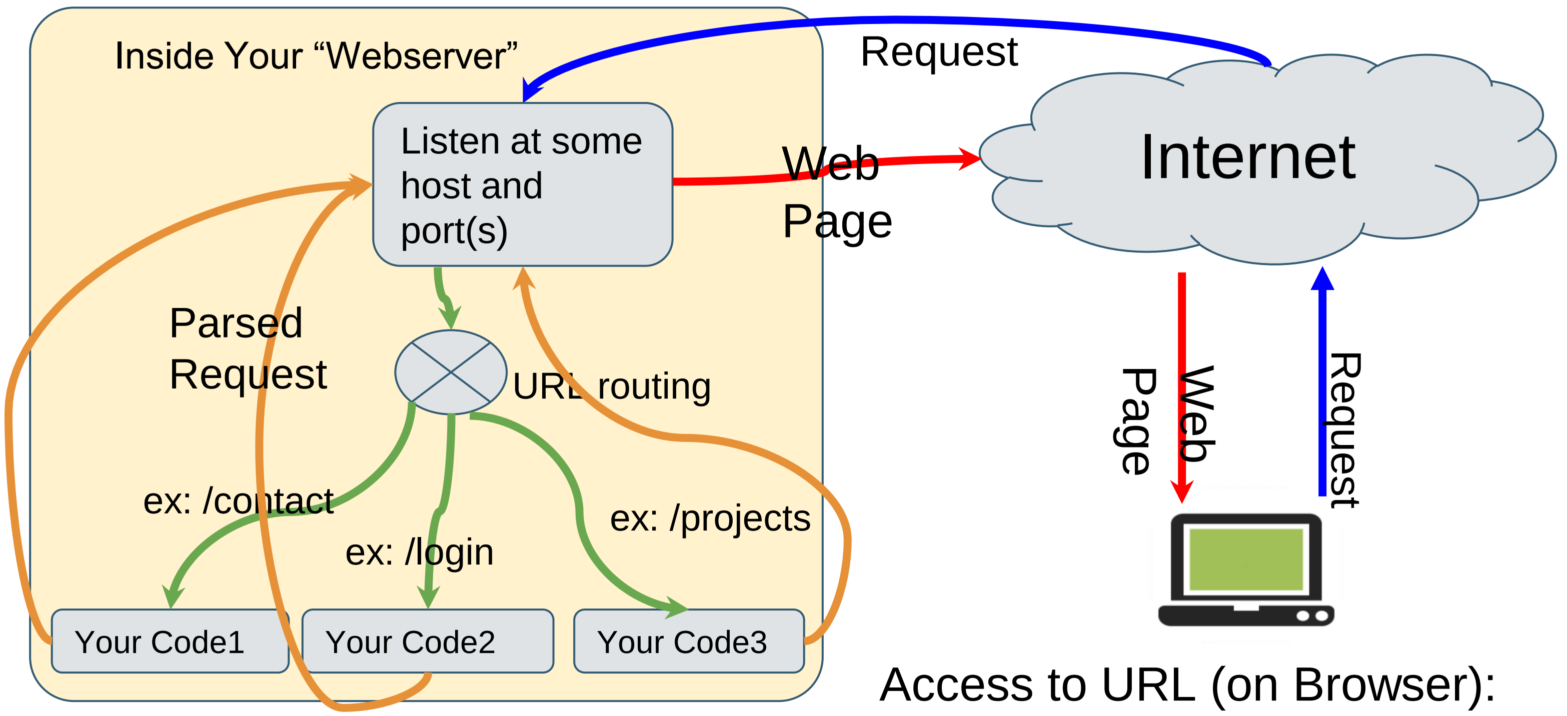
Django App

- We have to tell Django that it should use the `contohapp` application.
- Open `contohproject/settings.py` and add `contohapp` in the `INSTALLED_APPS` setting,

```
INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'contohapp',
]
```

Baca di Django documentation https://docs.djangoproject.com/en/4.1/intro/tutorial02/, apa saja aplikasi yang secara default sudah ada di INSTALLED_APPS setting dan apa fungsi dari masing-masing aplikasi tersebut.

Django URLs and Views



Access to URL (on Browser):
http://[appname].herokuapp.com/

Django URLs

- Django URLs forward request to appropriate view
- Django maps arguments from request or url to a view by using regular expression

- Contoh: supaya dapat mengakses contohapp di http://localhost:8000/contohapp

  1. Buatlah berkas urls.py di direktori aplikasi contohapp
  2. Tulis di dalam berkas tersebut:

```
from django.urls import include, path
from .views import index

urlpatterns = [
    path('', index, name='index'),
]
```

Django URLs

3. Di dalam berkas urls.py yang ada direktori project contohproject, import `django.urls.include` dan tambahkan modul `contohapp.urls` di dalam variabel `urlpatterns` sehingga isi berkas urls.py menjadi:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('contohapp', include('contohapp.urls')),
]
```

- https://docs.djangoproject.com/en/4.1/intro/tutorial01/
- https://docs.djangoproject.com/en/4.1/ref/urls/

Django Views

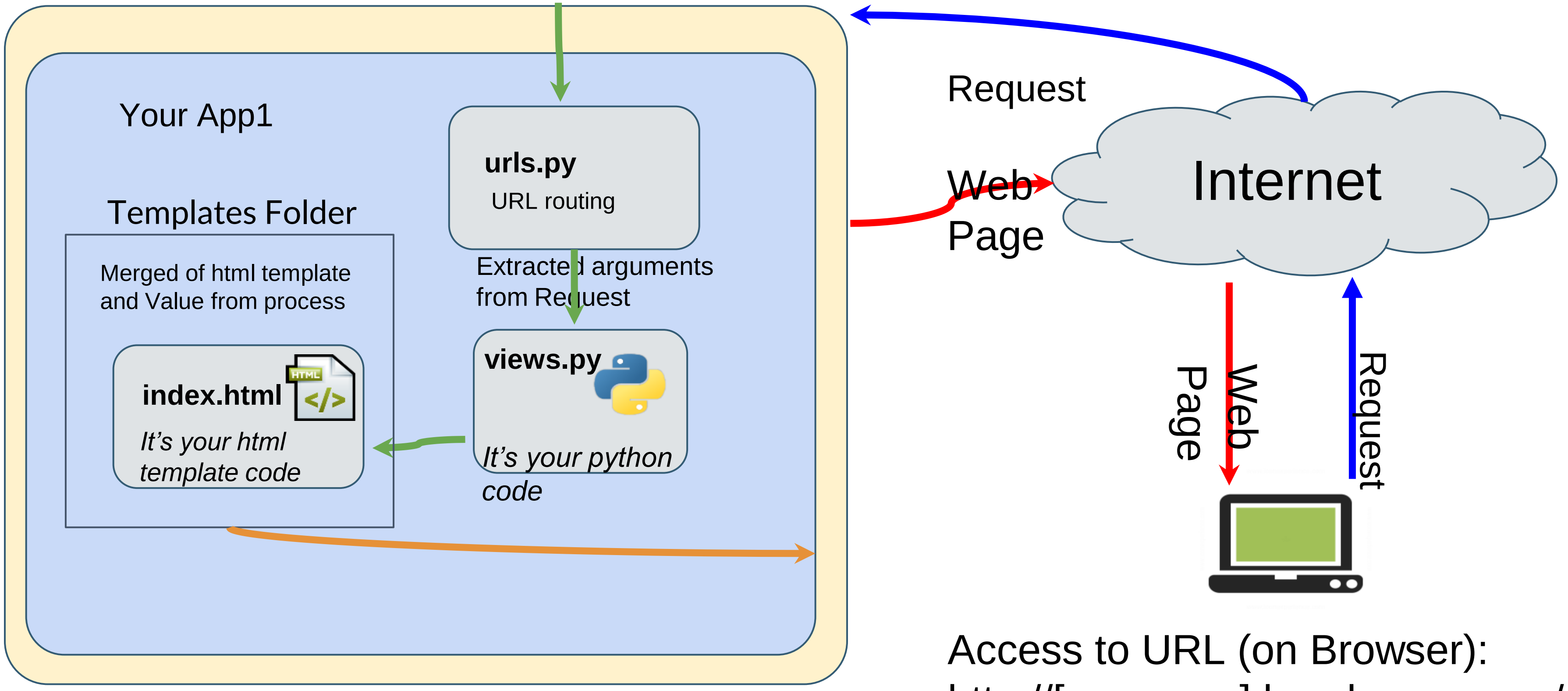
- A view function, or view for short, is a Python function that takes a web request and returns a web response. This response can be the HTML contents of a web page, a redirect, a 404 error, an XML document, an image, or anything.
- https://docs.djangoproject.com/en/4.1/intro/tutorial01/
- https://docs.djangoproject.com/en/4.1/topics/http/views/

- Untuk contohapp, setelah membuat konfigurasi urls, selanjutnya adalah buka berkas views.py yang ada di direktori contohapp dan tambahkan kode berikut ini:

```
from django.http import HttpResponse

def index(request):
    return HttpResponse ("Hello world!")
```

- Sekarang seharusnya http://localhost:8000/contohapp sudah dapat diakses di web browser.

Django Views and Templates



Access to URL (on Browser):
http://[appname].herokuapp.com/

Django Templates

- Most of frameworks include template engines
- Our apps likely has a lot of html that doesn't change (ex: header and footer) and some that does (ex: body)
- By convention DjangoTemplates looks for a "templates" subdirectory in each of the INSTALLED_APPS.

- Subdirektori "templates" harus kita buat sendiri.

- https://docs.djangoproject.com/en/4.1/intro/tutorial03/
- https://docs.djangoproject.com/en/4.1/ref/templates/builtins/

Untuk contohapp, buatlah folder bernama "templates" di dalam direktori aplikasi contohapp dan di dalam folder tersebut, buatlah berkas index.html berisi kode berikut ini:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>{{ name }}</title>
</head>
<body>
   <h1>Hello My Name is {{ name }}</h1>
</body>
</html>
```

Django Views and Templates

Views

Update contohapp/views.py menjadi:

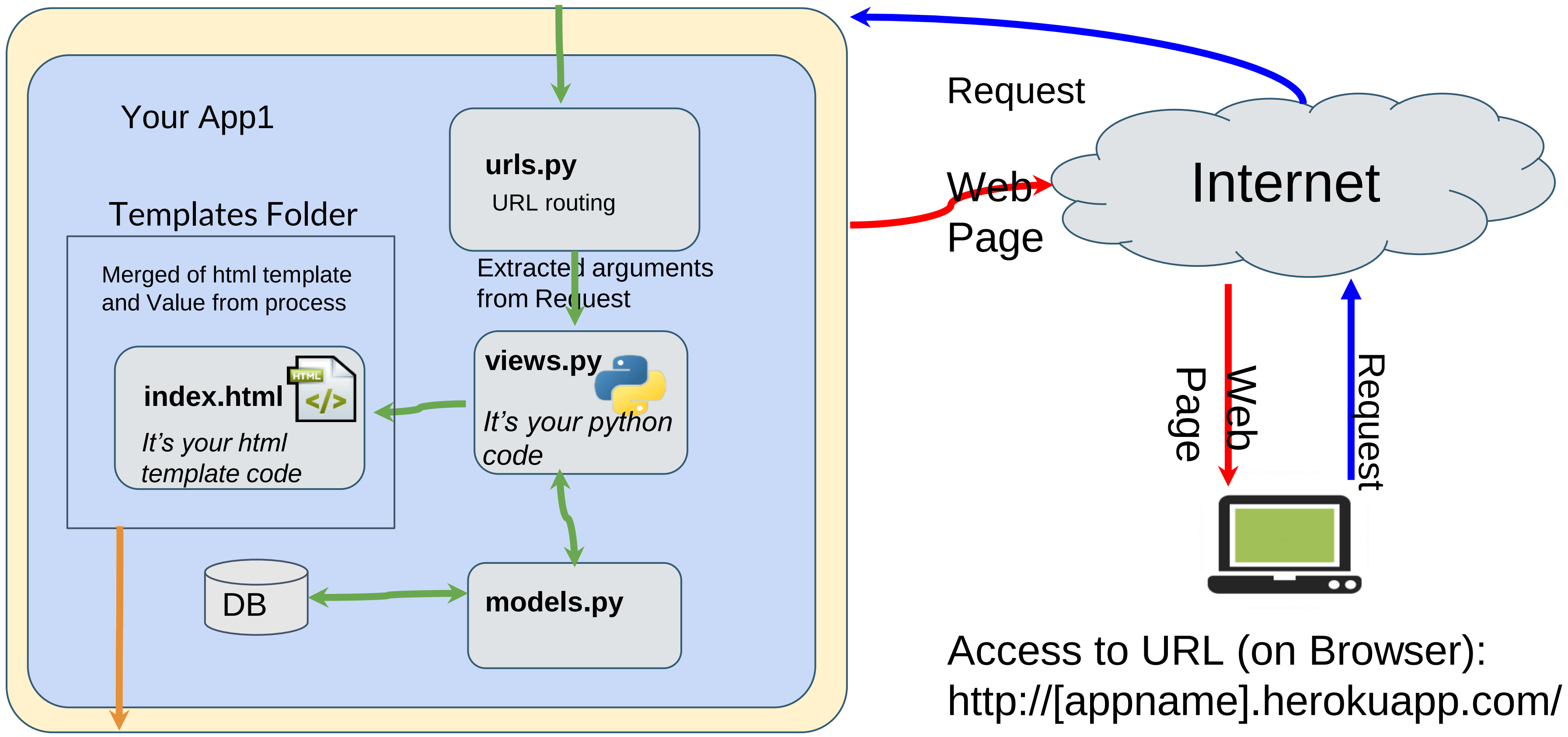
```
from django.shortcuts import render

mhs_name = 'Mahasiswa PBP'

def index(request):
    response = {'name' : mhs_name}
    return render(request,'index.html',response)
```

Templates

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>{{ name }}</title>
</head>
<body>
   <h1>Hello My Name is {{ name }}</h1>
</body>
</html>
```

Coba cek http://localhost:8000/contohapp di browser

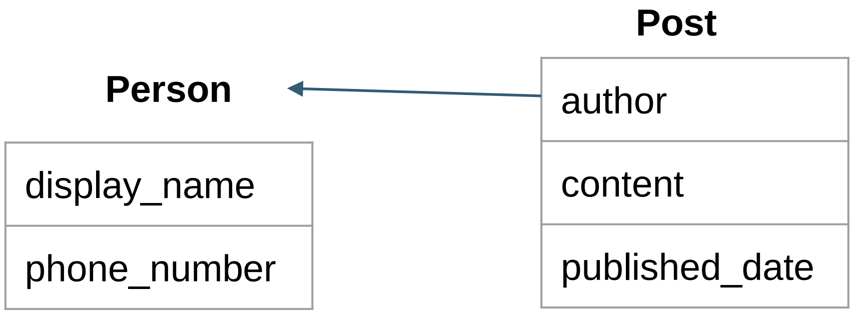
Django Views and Models



Access to URL (on Browser):
http://[appname].herokuapp.com/

Django Models

- Models are a set of data stored to be processed in our apps. We store them in form of tables that connect to one another. We can create, read, update, and delete (CRUD) data from the tables using several specific command instructions called SQL.
- Do you know or remember what is object?



- A model class == a database table
- A model instance == a database table row

Person

display_name	phone_number
Kak PeBePe	+628123456
Budi	+6281676732

- https://docs.djangoproject.com/en/4.1/topics/db/models/

Create Django Models

Update contohapp/models.py:

```
from django.db import models
from django.utils import timezone
from datetime import datetime, date

class Person(models.Model):
    display_name = models.CharField(max_length=30)
    phone_number = models.CharField(max_length=17)

class Post(models.Model):
    author = models.ForeignKey(Person, on_delete = models.CASCADE)
    content = models.CharField(max_length=125)
    published_date = models.DateTimeField(default=timezone.now)
```

Django Models - Field

Important part of database is to define the type of each attribute of a class

- models.CharField - define text with limited number of chars
- models.DateField - a date only
- models.DateTimeField - a date and time
- models.ForeignKey - a link to another model
- etc ..

https://docs.djangoproject.com/en/4.1/topics/db/models/#fields

Django Models

- Make sure your app is already installed! See settings.py again
- Run:

```
python manage.py makemigrations
python manage.py migrate
```

- https://docs.djangoproject.com/en/4.1/intro/tutorial02/
- The three-step guide to making model changes:

1. Change your models (in models.py)
2. Run `python manage.py makemigrations` to create migrations for those changes

   untuk mempersiapkan migrasi skema model ke dalam database Django lokal
3. Run `python manage.py migrate` to apply those changes to the database

   untuk menerapkan skema model yang telah dibuat ke dalam database Django lokal

Kenapa `makemigrations` dan `migrate` dipisah menjadi dua commands?
Apa yang terjadi jika kita hanya menjalankan `makemigrations` tanpa `migrate`?

Django Admin

- Django Admin is an internal application from Django to help developer/web admin for day-to-day operations, browsing data & support
- By default, the admin site url is http://localhost:8000/admin/
- To create a user to login with, use the `createsuperuser` command

```
python manage.py createsuperuser
```

(input username, email, and password)

Django Admin - Register

- To register models to Django Admin, update contohapp/admin.py:

```
from django.contrib import admin
from .models import Person, Post

admin.site.register(Person)
admin.site.register(Post)
```

- Open http://localhost:8000/admin/
- Add some `Person` objects in Django Admin

Django Models and Views

- Update contohapp/views.py:

```
from django.shortcuts import render
from .models import Person, Post

mhs_name = 'Kak PeBePe'

def index(request):
    persons = Person.objects.all().values()

    response = {'name' : mhs_name, 'persons' : persons}
    return render(request, 'index.html', response)
```

Django Models and Templates

- Update contohapp/templates/index.html:

```
...
<body>
    {% if persons %}
            {% for person in persons %}
                    {{ person.display_name }}
            <br/>
            {% endfor %}
    {% endif %}
</body>
...
```

Coba cek http://localhost:8000/contohapp di browser

Django Tests

- Automated testing is an extremely useful bug-killing tool for the modern Web developer. You can use a collection of tests – a test suite – to solve, or avoid, a number of problems:

  - When you're writing new code, you can use tests to validate your code works as expected.
  - When you're refactoring or modifying old code, you can use tests to ensure your changes haven't affected your application's behavior unexpectedly.

- The default `startapp` template creates a tests.py file in the new application.

https://docs.djangoproject.com/en/4.1/topics/testing/overview/

Django Tests Sample

- Update contohapp/tests.py:

```
from django.test import TestCase, Client
from django.urls import resolve
from .views import index

class ContohAppTest(TestCase):
    def test_contoh_app_url_is_exist(self):
        response = Client().get('/contohapp')
        self.assertEqual(response.status_code,200)

    def test_contoh_app_using_person_list_template(self):
        response = Client().get('/contohapp')
        self.assertTemplateUsed(response, 'index.html')
```

Django Tests & Code Coverage

- Run test: `python manage.py test`
- Code coverage
  - Code coverage describes how much source code has been tested. It shows which parts of your code are being exercised by tests and which are not. It's an important part of testing applications, so it's strongly recommended to check the coverage of your tests.
  - To install coverage package: `pip install coverage`
  - To run tests and collect coverage data of the executed files in the project:

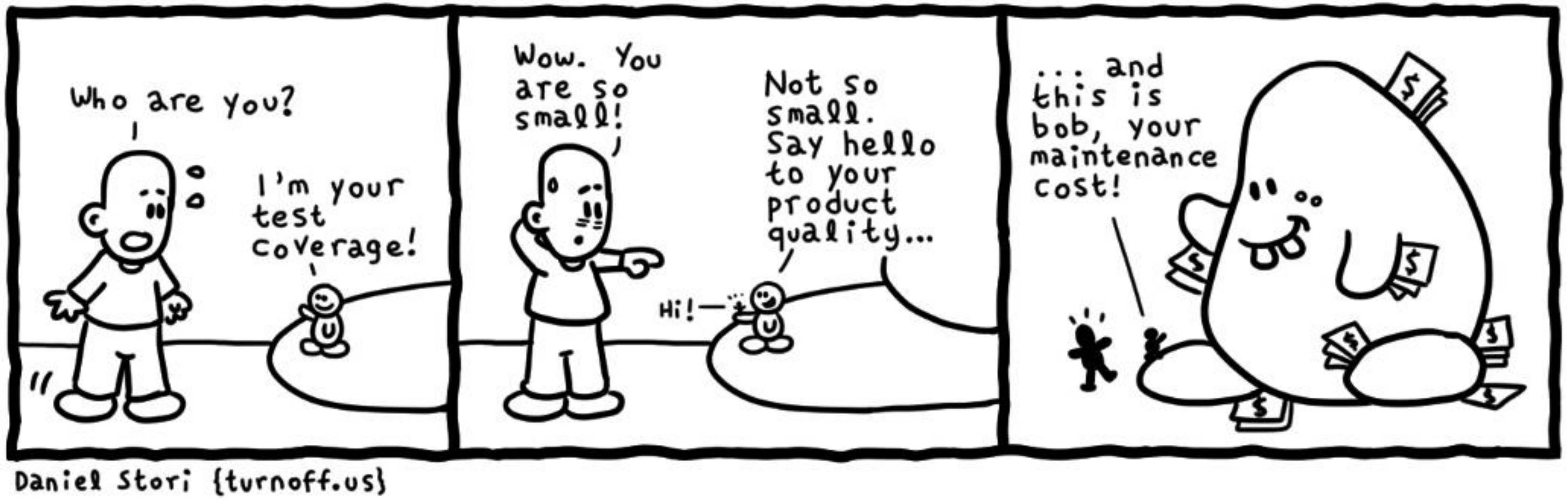
    ```
    coverage run --source='.' manage.py test
    ```

  - To see a report of coverage data:

    ```
    coverage report --show-missing
    ```

  - Note: Code coverage can only indicate that you've forgotten tests; it will not tell you whether your tests are good. Don't use good code coverage as an excuse to write lower quality tests.

Code Coverage


Who are you?
I'm your test coverage!
Wow. You are so small!
Not so small. Say hello to your product quality...
Hi!
... and this is bob, your maintenance cost!
Daniel Stori {turnoff.us}

Code coverage means what percentage of your codebase is covered by the tests or being tested. If you have no tests, then the code coverage is zero. (https://codeburst.io/10-reasons-why-code-coverage-matters-9a6272f224ae)