



The Web framework for perfectionists  
with deadlines

A gentle introduction by  
Steven

## What's for today?

- Introduction
- Web application at a glance
- Django
  - What? Why? Django
  - Key Concepts in Django
- Live coding session building a web-app with Django
- Deploying Django application (Last hour)

## Some rules

- Please silence your phone
- Go nuts with the examples, play around with it, break it, and most importantly, have fun
- Ask questions. Please raise your hand first and I'll let you speak when I'm ready for your question :)
- All of this slides are available online at  
[http://bit.do/gentle\\_django](http://bit.do/gentle_django)

## Who am I?

- Steven
- Graduated from The University of Edinburgh in 2016\* and Binus International in 2012\*\*
- Github: @SeiryuZ
- Email: [steven057@binus.ac.id](mailto:steven057@binus.ac.id)

\* Sponsored by Indonesian Endowment Fund for Education (LPDP)

\*\* Sponsored by Mom and Dad Foundation (UMP)

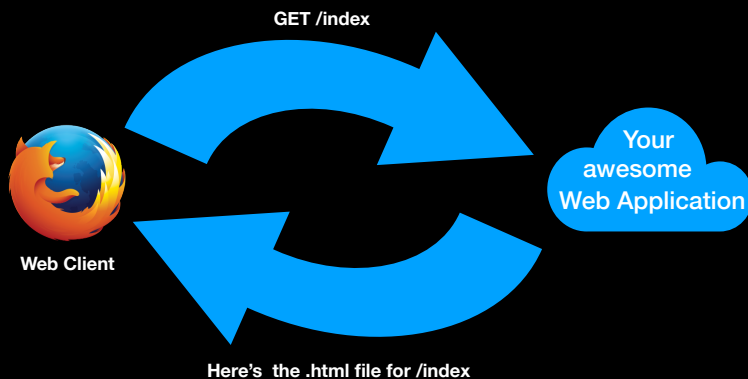
# Who am I?

User Inspired, not University of Indonesia

- I worked in a Django shop for almost 5 years now. PT UI ↗
- We worked on some really cool stuff:
  - [stamps.co.id](http://stamps.co.id)
  - [salon.co.id](http://salon.co.id) ← Acquired by Lunch Actually
  - Setipe (iOS) ←
  - GO-JEK (Initial iOS version)
  - Zap Laundry (secret project)

# Web application at a glance

## Web Application at a glance



## Web Application at a glance

```
➤ curl http://www.google.co.id/ -v
* Trying 216.58.199.195...
* TCP_NODELAY set
* Connected to www.google.co.id (216.58.199.195) port 80 (#0)
> GET / HTTP/1.1
> Host: www.google.co.id
> User-Agent: curl/7.54.0
> Accept: */*

< HTTP/1.1 200 OK
< Date: Sat, 21 Oct 2017 06:12:50 GMT
< Expires: -1
< Cache-Control: private, max-age=0
< Content-Type: text/html; charset=ISO-8859-1
< P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
< Server: gws
< X-XSS-Protection: 1; mode=block
< X-Frame-Options: SAMEORIGIN
< Set-Cookie: IP_JAR=2017-10-21-06; expires=Sat, 28-Oct-2017 06:12:50 GMT; path=/; domain=.google.co.id
< Set-Cookie: NID=114=bKEJpi-c52qd39-2eznbRlykacy:IMWgMWF0v6C13zNb7QrjX8CE-uQ68-g0R80F-R4vaGrtLxucm3giU916V8jv_WTid9JQ0gfJLA
Oy908ltr1tvkyGfUnRZq; expires=Sun, 22-Apr-2018 06:12:50 GMT; path=/; domain=.google.co.id; HttpOnly
< Accept-Ranges: none
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
<

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="id"><head><meta content="text/html; charset=UTF-8" ht
tp-equiv="Content-Type"><meta content="/images/branding/google/1x/googleleg_standard_color_128dp.png" itemprop="image"><title>Googl
e</title><script>(function(){window.google={kE1:'YUxqWc37D4X5vgTjokaACA',kEXP1:'1352261,1352821,1352961,1354277,1354481,1354915,13
55004,1355324,1355568,1355735,1355820,1355892,1356833,3700754,3700440,3700476,4029385,4031109,4041302,4041899,4043492,4044347,4045
841,4048347,4063220,4072776,4076999,4078430,4081039,4081165,4090894,4092183,4093169,4095910,4097153,4097922,4097929,4098721,409872
8,4098752,4102237,4103475,4103845,4104242,4104288,4104414,4106455,4108895,4109489,4109596,4110656,4111590,4112931,4113217,4115218,
4115404,4116349,4116724,4116731,4117328,4117980,4118227,4118798,4119272,4119283,4119740,4119811,4119820,4120415,4120660,4121835,41
```

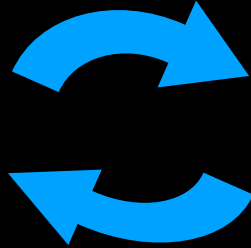
**Request** (indicated by a blue arrow pointing to the request lines)

**Response** (indicated by a blue arrow pointing to the response lines)

## Web Application at a glance



I need coffee,  
Toraja single origin  
manual brew please



Here's your coffeeh, sir

**moz://a**

\* Reality check, It is unrealistic to think you will be able to create instagram-scale Webapp after this workshop



**Battle  
Tested**

This is where I showed off some secret numbers

## What is Django

- Originally, an internal framework for a web team who maintain newspaper websites between 2003 - 2005
- Evolved into a generic web development framework that was open-sourced in July 2005
- First milestone release (1.0) in September 2008

# What is Django

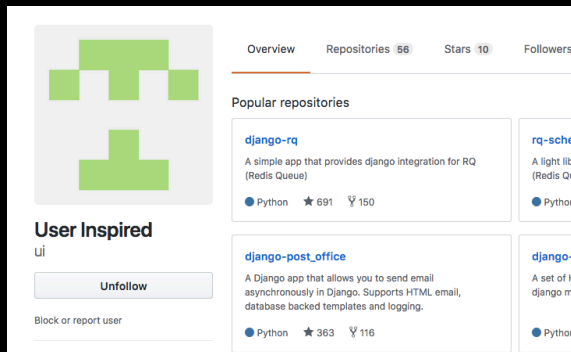
- A high-level Python Web framework
  - “Complete”™ - Batteries included
  - Flexible
  - Secure
  - Scalable

# Why Django/Python?

- Easy to teach (Really important in a startup)
- Clean codes
- Open Source
- Huge extensive libraries that extends Django's functionalities
- Fantastic communities

# Huge Libraries?

- <https://pypi.python.org/pypi?%3Aaction=search&term=django&submit=search>



So let's get our hands dirty

We'll create a simple web app together called that will allow you to track your expenses

## Some note about this live-coding session

- This might not follow best practices due to time constraint
- Some settings are taken care off by the skeleton project
- I am going to skip Cascading Style Sheet (CSS) and Javascript(JS) entirely

## Pre-requisite

- python 3 is ready (let me explain why python3)
- git is ready
- heroku-cli is ready
- Internet connection (I suggest tethering to your phone / mobile modem)

## Pre-requisite

- Open up [http://bit.do/gentle\\_django](http://bit.do/gentle_django)
- There's a file called "scratchpad.md", all of the command you see are there for copy paste

## Pre-requisite

- Open up terminal / cmd
- Navigate to the correct folder you want
- git clone [https://github.com/SeiryuZ/my\\_expenses.git](https://github.com/SeiryuZ/my_expenses.git)

## Pre-requisite

- pip install virtualenvwrapper (virtualenvwrapper-win for windows)
- mkvirtualenv my\_expenses -p <python path>
- workon my\_expenses
- cd my\_expenses
- pip install django==1.11.6

## What Django Project looks

```
-- (up a dir)
/Users/steven/dev/my_expenses/
- my_expenses/
  - __pycache__/
  - apps/
    - __pycache__/
    - expenses/
      - __pycache__/
      - migrations/
      - __init__.py
      - admin.py
      - apps.py
      - models.py
      - tests.py
      - views.py
      - __init__.py
    - __init__.py
    - settings.py
    - urls.py
    - views.py
    - wsgi.py
  - templates/
    - index.html
    - db.sqlite3
    - manage.py
    - Procfile
    - requirements.txt
```

```
3 The urlpatterns
4 https://
5 Examples:
6 Function vi
7 1. Add
8 2. Add
9 Class-based
10 1. Add
11 2. Add
12 Including a
13 1. Impo
14 2. Add
15 """
16 from django
17 from django
18
19 from .views
20
21 urlpatterns
22 #url(r
23 url(r'^
24 ]
```

## Django Components

- settings.py
- Handling HTTP Requests
  - urls
  - views
- forms
- models + database
- templates

## Remember the coffee shop analogy?



# Django Components

- settings.py - **Boss**
- Handling HTTP Requests
  - urls - **Cashier**
  - views - **Barista**
- forms - **Point of Sales System**
- models + database - **Coffee Bean + Storeroom**
- templates - **Paper Cup**

# settings.py

- the owner of the coffee shop
- dictates what got sold and who will serve the customer, who is going to prepare drinks / coffee
- all the shop's secret is with him

# urls.py

- The cashier of the coffee shop
- Routes all customer requests to the correct persons
- It will try to find the correct person that is able to handle the request, if none is found, it will apologize to the customer that their request cannot be fulfilled.

# Views (function based)

- The Barista of the shop
- The one actually producing the order, taking the coffee beans, grinding it, and prepare it into delicious coffeeh
- And finally delivering the coffee to the customer

# Forms

- The cashier machine
- It validates that when requesting resources the customer give appropriate resource to work on (Money)
- If it's not enough, wrong currency, or not paying at all, reject the request

# Model + Database

- In our Analogy, Model is like the beans, and database is the warehouse where you store your coffee beans
- It is what get processed by the barista to produce whatever the customer wants

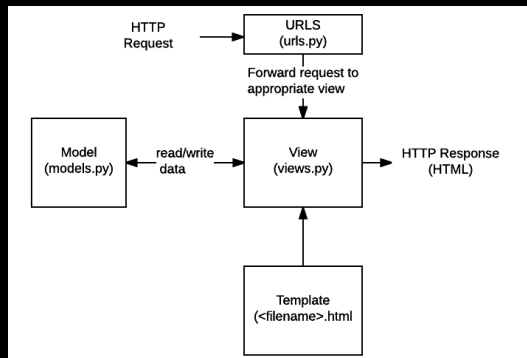
# Object Relational Mappers

- There is one component which handles the operation to the database in high-level
- Warehouse guy
- It translates the request from the Barista, "Get me the beans from papua that was shipped 3 days ago" and gives the correct bean back as requested
- It is much better than the barista going to the warehouse themselves and retrieving it (You can do it, though)

# Templates

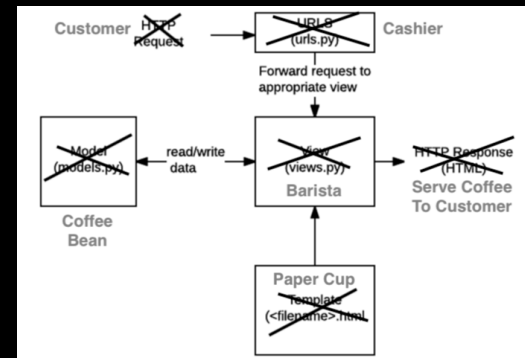
- Is the cup where you served the coffee
- You know how to make a coffee, even if all the requests have different content, you have a template on how to serve it
- And you can customize this a lot or even replace the cup entirely





## Django MVT in a nutshell

taken from <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>



## Django MVT with our Analogy

## Ok you got the concept

- Now, we are going to build the web app together
- Live coding session (\*Praying to demo God\*)
  - Our goal for this session:
    - Create expenses model
    - List all expenses ordered by created time
    - Add form to allow user to enter expenses
    - Report on expenses based on type

## Deploying our Web App



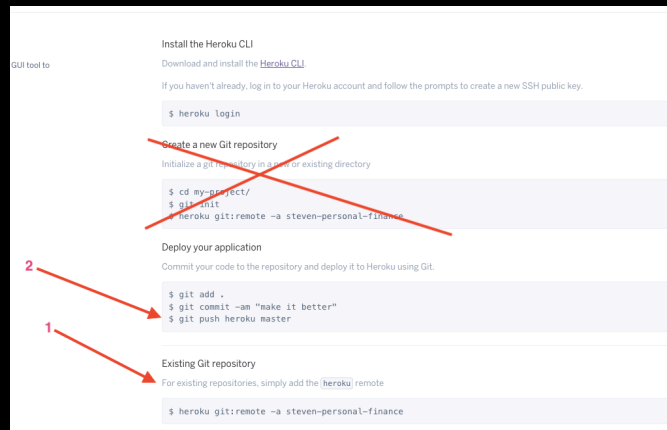
# Heroku?

- Platform As A Service(PAAS)
- Provide really generous free tier (550 hours monthly)
- We're going to use their free tier to deploy our web app

# How to deploy with heroku?

- Login on heroku.com
- Create new App
- App name must be unique ([app-name.herokuapp.com](https://app-name.herokuapp.com)) or leave empty and Heroku will generate one for you
- Choose USA, standard
- Click create app

# How to deploy with heroku?



The screenshot shows the 'Install the Heroku CLI' section of the Heroku documentation. It includes instructions to download and install the Heroku CLI, log in, and create a new Git repository. A red 'X' is drawn over the 'Create a new Git repository' section. Two red arrows point to the 'Deploy your application' section: arrow '1' points to the 'Existing Git repository' subsection, and arrow '2' points to the 'Deploy your application' subsection. The 'Existing Git repository' subsection shows the command: `$ heroku git:remote -a steven-personal-finance`. The 'Deploy your application' subsection shows the commands: `$ git add .`, `$ git commit -m "make it better"`, and `$ git push heroku master`.

Drum roll pleaseeee



heroku open

## So what's next?

- Feel free to play around with the web app / break it
- A good feature to add are:
  - Authentication (You don't want everyone to be able to change your data right?)
  - Add another model, Income
  - Add more types of report (Net gain report), add charts

## More resources

- <https://tutorial.djangogirls.org/en/>
- [https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial\\_local\\_library\\_website](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website)
- <https://docs.djangoproject.com/en/1.11/intro/tutorial01/>

## Last words

- Questions? [steven057@binus.ac.id](mailto:steven057@binus.ac.id)
- Interested in Python + Django? Talk to me, people are looking for Django developers
- Thanks for having me :)