DJANGO!

SOME STARTING QUESTIONS

- What is Django?
- What can it do?
- Why do we use it?
- How do I get started?

WHAT IS DJANGO?

"Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source."

What does this even say?

- Web framework
  - Django provides you with an abundance of tools that do the tedium for you. These tools come in the form of Python modules and classes
- Rapid development and clean, pragmatic design
  - The tools you're given expect you to conform to a standard way of taking user input, generating web pages, saving data, etc.

Django vs. Flask?

Django

- full stack framework
- many built in features such as admin panel, authentication, etc.
- has designated template and model formats for URL routing
- provides more database integrations support

Flask

- micro framework with minimalist goal
- allows for full control and customization
- uses Jinja2 as its template engine, no formalized model scheme; it’s up to you!
- preferred for smaller projects

WHAT CAN IT DO?

Django aids in:

- Creating dynamic web pages
- Sanitizing user input
- Creating and saving database entries
- Authenticating and authorizing users

WHY DO WE USE DJANGO?

- Free
- Open source
- Written in Python
- Widespread

HOW DO I GET STARTED?

- Django provides a variety of different tutorials and documentation
- https://www.djangoproject.com/start/
- Start fiddling around with it!
- Find examples online!

CREATING A NEW DJANGO PROJECT

- Install Django on your local machine
  - Install Django Instructions
- Create a new folder and install the django project there:

```
mkdir “foo”

django-admin startproject “bar” “foo”
```

```
my_project/
├── manage.py
├── my_project/
├── first_app/
├── second_app/
```

```
my_project/
├── __init__.py
├── asgi.py
├── settings.py
├── urls.py
├── wsgi.py
```

```
second_app/
├── __init__.py
├── urls.py
├── views.py
├── models.py
├── forms.py
```

```
first_app/
├── __init__.py
├── urls.py
├── views.py
├── models.py
├── forms.py
```

Django Project Structure

HOW DOES DJANGO WORK?

Webpages, databases, and everything in between

FRONT END

URLS

Listed in a file named “urls.py” or “urlconf.py”

We map URLs to Views

VIEWS

The main way that users interact with any Django site

Views process data passed in and display information based on templates

Different kinds: function-based vs. class-based

Templates

HTML that is rendered from a view to the user

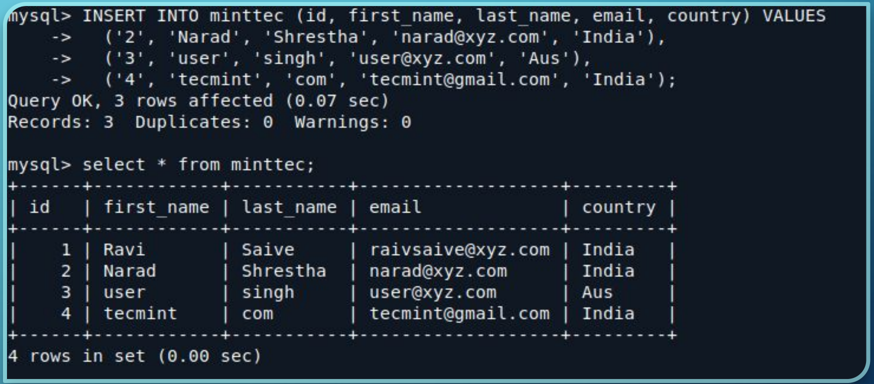
Django has its own template language

Provides ability for dynamic webpages

BACK END

MODELS

- How Django stores and processes data
- Written like Python classes
- Can use model fields to store and group data
- Can define functions to perform actions on a model object



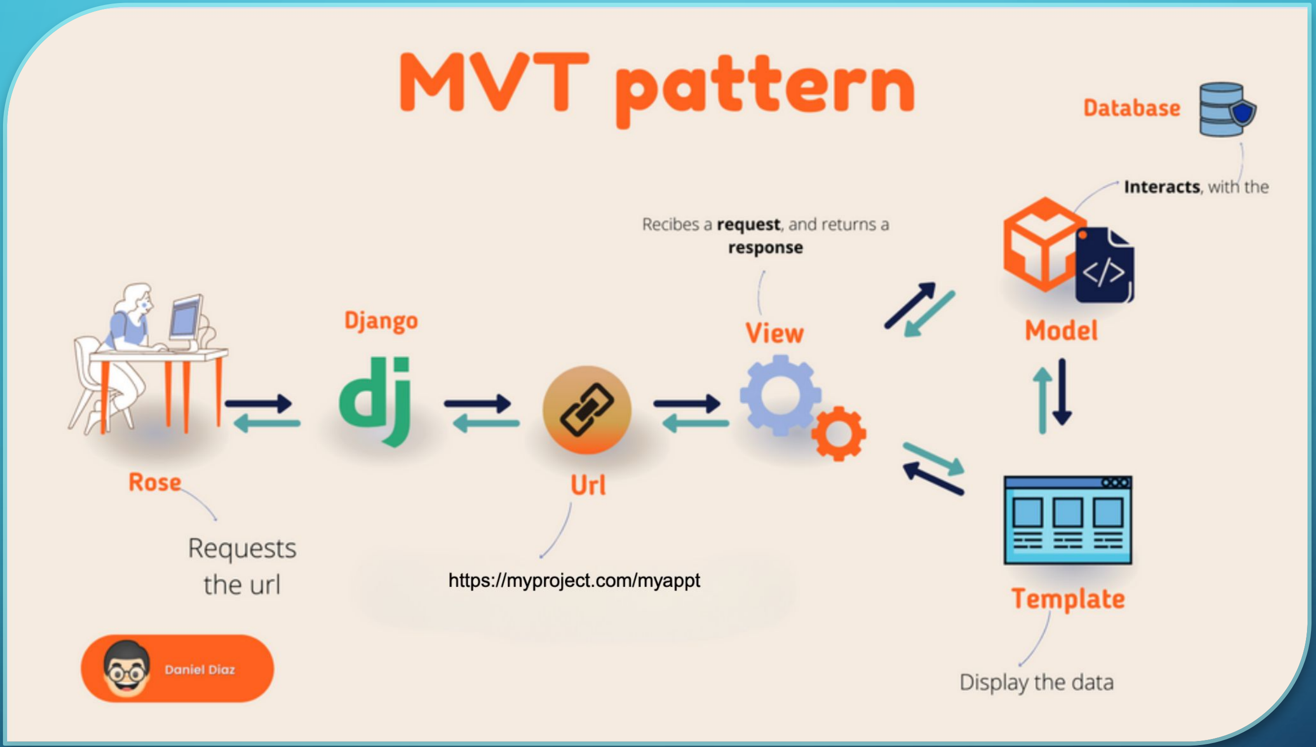
```
mysql> INSERT INTO minttec (id, first_name, last_name, email, country) VALUES
    ->    ('2', 'Narad', 'Shrestha', 'narad@xyz.com', 'India'),
    ->    ('3', 'user', 'singh', 'user@xyz.com', 'Aus'),
    ->    ('4', 'tecmint', 'com', 'tecmint@gmail.com', 'India');
Query OK, 3 rows affected (0.07 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from minttec;
+------+------------+-----------+-------------------+---------+
| id   | first_name | last_name | email             | country |
+------+------------+-----------+-------------------+---------+
|    1 | Ravi       | Saive     | raivsaive@xyz.com | India   |
|    2 | Narad      | Shrestha  | narad@xyz.com     | India   |
|    3 | user       | singh     | user@xyz.com      | Aus     |
|    4 | tecmint    | com       | tecmint@gmail.com | India   |
+------+------------+-----------+-------------------+---------+
4 rows in set (0.00 sec)
```

FORMS

Rendered by Views (via templates)

Takes input from a user, verifies that it isn't malicious, and then cleans it



Putting it all together

THE ADMIN PAGE

VIEW, CHANGE,
AND DELETE
DATABASE ENTRIES