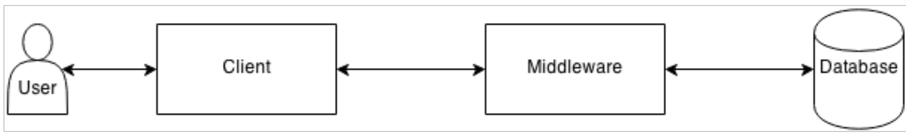
Django
Web Application Framework

Web Application Development 2

As architects we need to see through the complexities, abstract away, and design a useful solution.

Where should we start?

High Level System Architecture



- We need to work out what we are going to build
- And we need to decide what technologies will be used in each box
- For the middleware, we will be using Django as the Web Application Framework (WAF) for building the application server

Django

- Pronounced JANG-oh
- *“Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design”*
- Claimed to be the web framework for perfectionists with deadlines

Django

- History:
  - Created by developers at a newspaper in 2003 who were keen on using Python to efficiently develop rapidly-produced content
  - Open Sourced in 2005, first major release in 2008
- Primary Focus:
  - Dynamic and database driven websites
  - Content based websites
  - Many large sites have used it:
    - Instagram, Spotify, The Washington Post, Dropbox, Quora, Pinterest…

Why Django for Web Dev

- Lets you divide a site or code module into logical components
  - Underpinned by the MVC/MTV design pattern
  - Providing flexibility and easier to change
- Provides automatically generated web administration
  - Easier to manage the database
- Provides many pre-packaged APIs for common tasks

Why Django for Web Dev

- Provides a template system to define HTML templates
  - Avoids code duplications
  - Subscribes to DRY principle
- Allows you extra control to define what the URL will be for a given view
  - URL requested by browser not directly accessing html file
  - Loose Coupling Principle
- Allows you to separate business logic from the presentation
  - Separation of Concerns

Overall Design Philosophy

- Loose Coupling
  - various layers of the framework shouldn't "know" about details of each other
- Don't Repeat Yourself (DRY)
  - "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system"
- Less Code
- Quick Development
- Explicit is better than implicit
  - A core Python principle
- Consistency
- See https://docs.djangoproject.com/en/2.2/misc/design-philosophies/ for more details

Django Modules

- Administration Interface (CRUD)
  - Create, Read, Update and Delete
- Authentication Systems
- Form Handling
- Session Handling
- Syndication Frameworks
  - RSS and Atom feeds
- Caching
- Internationalization and Localization
- And much, much more…

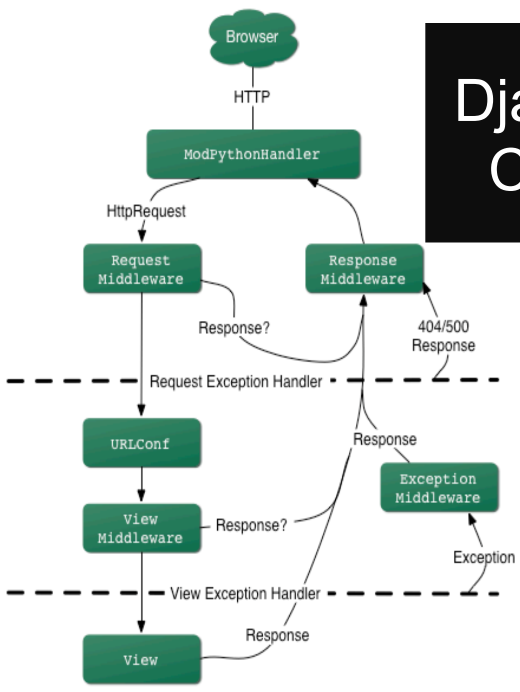
Model View Controller

- Software architectural design pattern
  - Divides software into three interconnected parts
  - Separation of concerns
- **Models** describe your internal data representation
- **Views** determines what the user sees
- **Controller** binds it all together, handles logic, interactions
- Not firm definitions; individual 'MVC' designs vary significantly

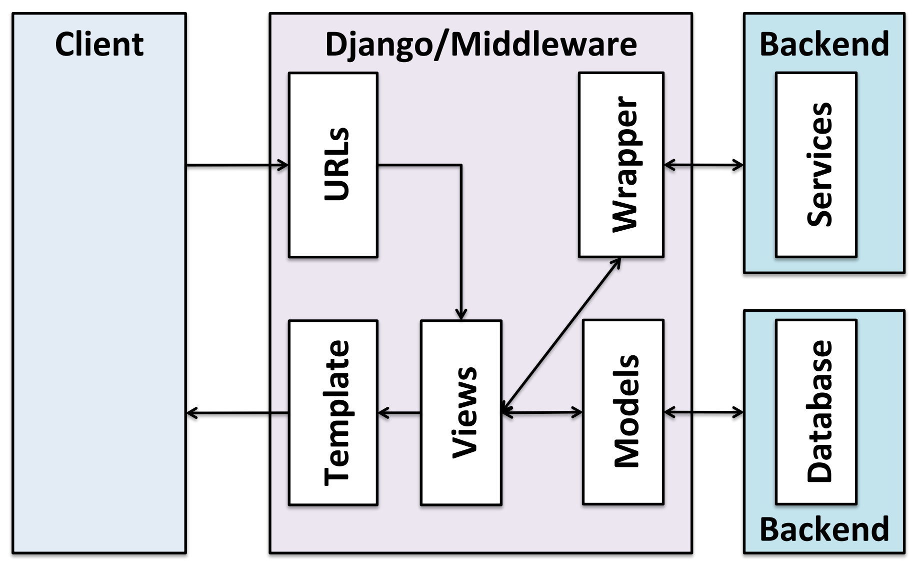
Model View Controller in Django

- **Model** describes database entities and relationships, specified as Python classes
- **Views** specified using web templating
- **Controller** is handled by
  - the Django Framework
  - URL parser maps urls to views
- Again, definition not concrete
  - Slightly complicated because Django has objects called 'views' that query/process data, and might be considered part of the controller role in MVC
  - Some people include templates in the describing Django's structure, so MVCT, or just MTV

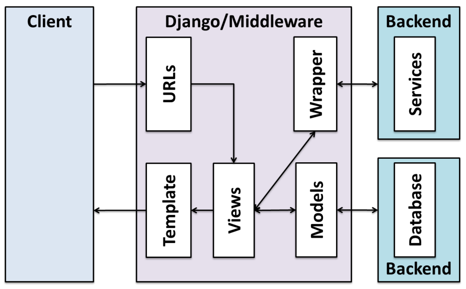
Django Internal Control Flow



Simplified Internal Flow

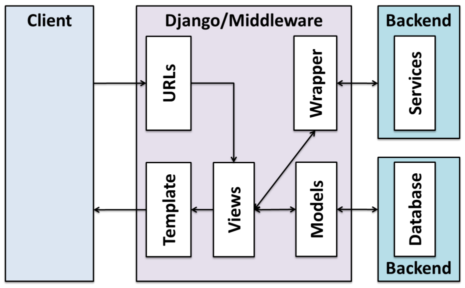


Internal Sections/Components



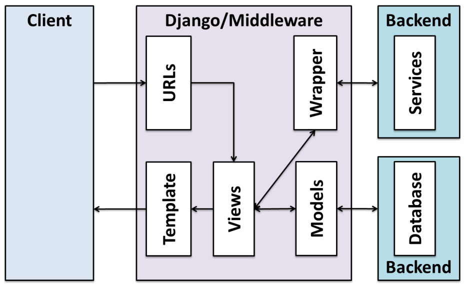
- **Building Data Models (usually in models.py):**
  - The models specify the entities and relationships in the database – these provide an Object Relational Mapping (ORM) to the actual database tables
  - The framework constructs the database given the models defined

Internal Sections/Components



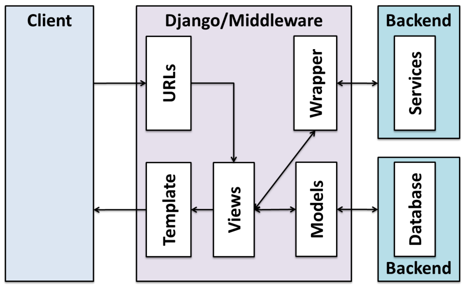
- **Defining Views (usually in views.py):**
  - Views are responsible for handling and processing the specific request, collating the data from databases/external services, then selecting the template, for the response to be generated

Internal Sections/Components



- **Controlling flow (usually in urls.py):**
  - To specify what view function should handle a particular URL (or part thereof), URL patterns are used to find matches with the URL, and to route this request to the appropriate view
  - The use of pattern matching means that different instantiations can be handled by a common pattern

Internal Sections/Components



- **Providing Templates**
  - The templates mean the response format (html,xml,etc) is decoupled from the data to be presented

Django Beginner’s Tutorial

Parts 1 and 2

```
https://docs.djangoproject.com/en/2.2/intro/tutorial01/
https://docs.djangoproject.com/en/2.2/intro/tutorial02/
```

•Making ‘Polls’ web application in Django

•Using PythonAnywhere
•Creating projects
•Creating views
•Mapping URLs
•Creating models
•Querying and modifying models
•The Django admin site

PythonAnywhere

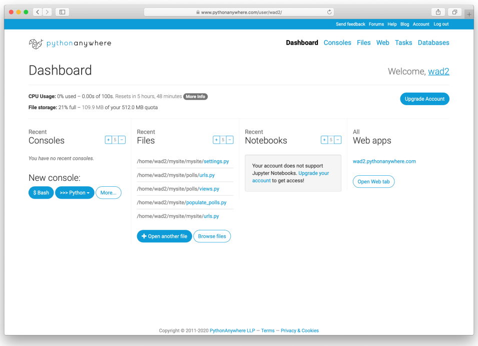
- Allows you to host your web applications
- Sign up for a Beginner account – host 1 web app

  `https://www.pythonanywhere.com`

- Application will then be deployed at

  `https://<username>.pythonanywhere.com`

- Tabs:
  - Consoles
  - Files
  - Web
  - Tasks
  - Databases



Dashboard

Welcome, wad2

CPU Usage: 0% used – 0.00s of 100s. Resets in 5 hours, 48 minutes

File storage: 21% full – 109.9 MB of your 512.0 MB quota

Recent Consoles

You have no recent consoles.

New console:

Recent Files

/home/wad2/mysite/mysite/settings.py
/home/wad2/mysite/polls/urls.py
/home/wad2/mysite/polls/views.py
/home/wad2/mysite/populate_polls.py
/home/wad2/mysite/mysite/urls.py

Recent Notebooks

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

All Web apps

wad2.pythonanywhere.com

Copyright © 2011-2020 PythonAnywhere LLP — Terms — Privacy & Cookies

Creating a virtual environment

- virtualenv - alternative to Anaconda using in labs

- Open up a Bash console via the Consoles tab
  - `mkvirtualenv -p python3.8 polls`

- Check out your virtual environments at `/home/<username>/.virtualenvs` *or with* `lsvirtualenv`

- Install packages:
  - `pip install django==2.2.17`
  - `pip install pillow`

- Switch to the virtual env when you open a new console:
  - `workon polls`

- If you wish to leave a virtual env:
  - `deactivate`

Configuring PythonAnywhere

- On the "Web tab", select "Add a new web app"
  - select `Manual configuration` as web framework
  - select `Python 3.8` as Python version

- Under "Virtualenv" enter `polls`

- Configure your WSGI file (delete the code from the existing file and replace it with the code from the sample WSGI on the WAD2 Moodle page), changing "wad2" to your PythonAnywhere username

- Assign static paths
  - `/static/` should be set to `/home/<username>/mysite/static`
  - `/media/` should be set to `/home/<username>/mysite/media`
  - `/static/admin` should be set to `/home/<username>/.virtualenvs/polls/lib/python3.8/site-packages/django/contrib/admin/static/admin`

Creating the project and application

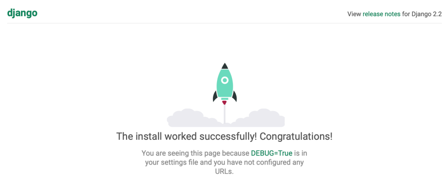
- Create the `mysite` project. In a console:
  - `django-admin startproject mysite`

Directory Structure

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Creating the project and application

- Create the `mysite` project:
  - In console: `django-admin startproject mysite`
- Inform PythonAnywhere of the source code location
  - `/home/<username>/mysite`
- Configure `ALLOWED_HOSTS` in `settings.py`
  - Add `'<username>.pythonanywhere.com'`
- Press green “Reload” button on web tab and navigate to `https://<username>.pythonanywhere.com/`



django

View release notes for Django 2.2

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

- Create the `polls` application:
  - `python manage.py startapp polls`

Directory Structure

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
    polls/
        __init__.py
        admin.py
        apps.py
        migrations/
            __init__.py
        models.py
        tests.py
        views.py
```