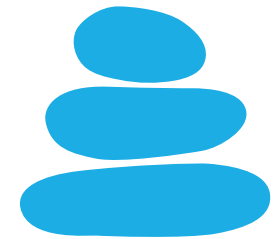


Topic 1: The Web Stack

CM3035 Advanced Web Development

Mr Kan Hwa Heng



Learning Objectives

1. Components of a full stack web application
2. Introduction to Django
3. Main components of Django

Background: Introducing the Full Stack Web Server

Lesson 1.1



Overview of TCP/IP

TCP/IP is a networking protocol used by the Internet and most networks that we use.

Each physical device communicating on a network is assigned an IP address.

- Example of an IP address: 172.27.177.57

Software applications on the same device listens for messages on specific port numbers

Examples of an IP address with port numbers:

- 172.27.177.57:20
- 172.27.177.57:80

Overview of TCP/IP

Some common application ports

- Port 20 and 21: FTP data and FTP control
- Port 22: Remote login secure shell (SSH)
- Port 53: DNS
- Port 80: Hypertext Transfer Protocol (HTTP)
- Port 143: E-mail Internet Message Access Protocol (IMAP)
- Port 443: Secure Hypertext Transfer Protocol (HTTPS)

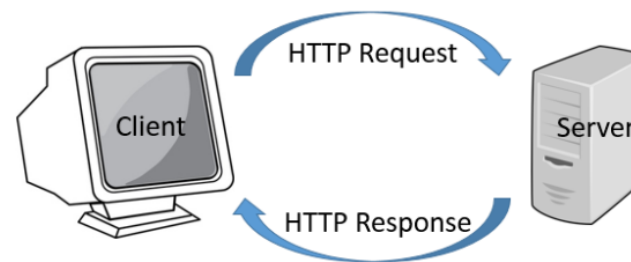
Overview of HTTP

HTTP – Hypertext Transfer Protocol

HTTP is a client/server protocol for fetching resources such as HTML documents. It is the protocol used to communicate with web servers.

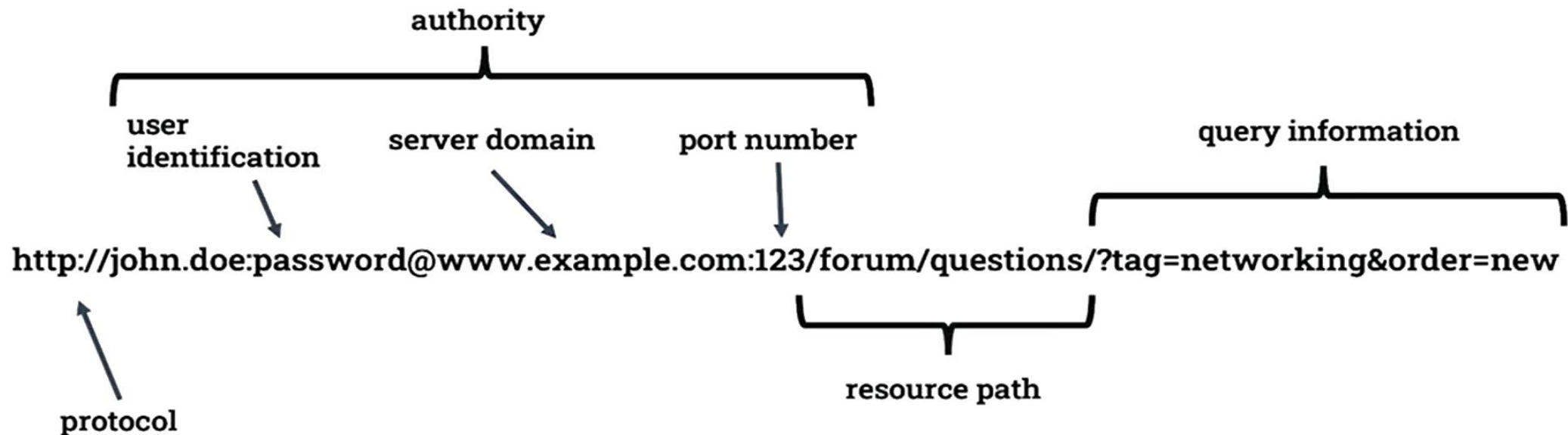
HTTP is a stateless request-response protocol

- Client application requests data
- Server application sends data



Overview of HTTP

Client requests use **Uniform Resource Identifiers (URIs)** to specify required resources



Overview of HTTP

HTTP requests and responses

Line	HTTP REQUEST	HTTP RESPONSE
1	request line	status line
2	requests meta data	response meta data
3	blank line	blank line
4+	request body	response body

Overview of HTTP

HTTP request methods

HTTP REQUEST	MEANING
GET	Request a resource
POST	Request a method
PUT	Send data to the server
DELETE	Delete a remote resource

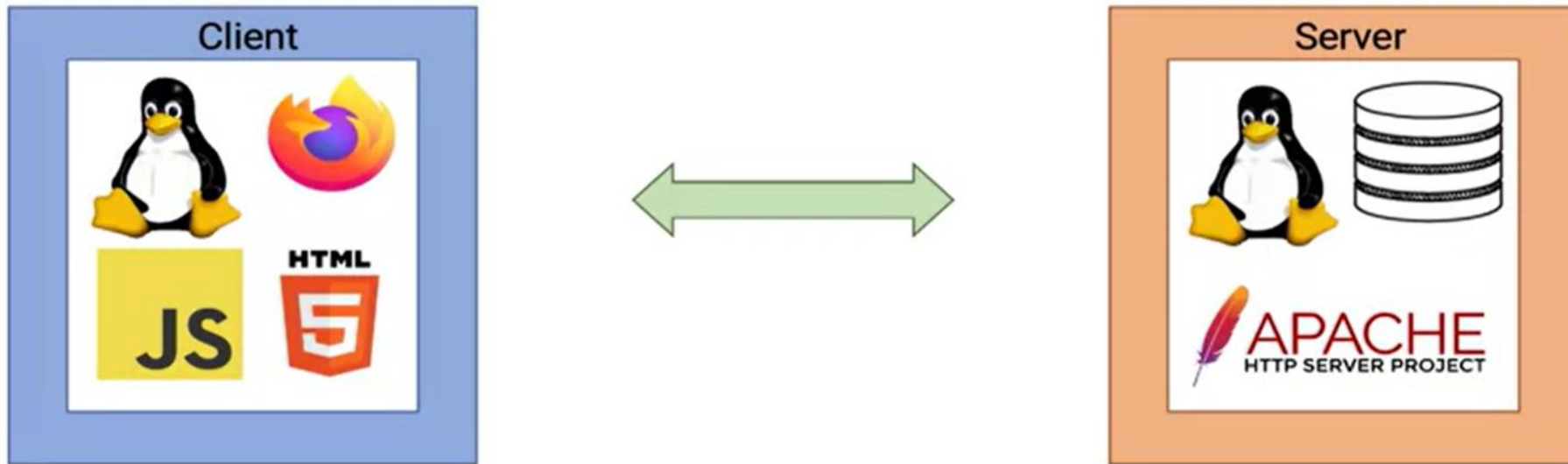
Overview of HTTP

HTTP status codes

Response class	Response type	Meaning
1xx	informational	request received correctly, processing is ongoing
2xx	success	request received correctly, responded correctly
3xx	redirection	requests received, resource moved location
4xx	client error codes	request malformed
5xx	server error codes	server is unable to fulfil request

Full Stack Web Applications

A **web stack** is a suite of technologies required to deliver a website or web application.



Common Web Application Stacks

	LAMP	WISA	MEAN
Operating system	Linux	Windows	Linux
Web server	Apache	Windows Server IIS	Express.js
Data store	MySQL	Microsoft SQL Server	MongoDB
Server-side programming language	PHP	ASP.net	Javascript
Frontend framework	unspecified	unspecified	Angular

Alternative Applications

	Common	Alternatives
Web server	Apache	Nginx, Express.js
Data store	MySQL	MongoDB, PostgreSQL
Server-side programming language	PHP	Python, Ruby, JavaScript, Java
Frontend framework	Angular, REACT,	

Elements of Web Development (1 of 2)

Browsers - Google Chrome, Safari, Firefox, Internet Explorer

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

Programming languages such as JavaScript, Python, Ruby, PHP, Objective-C, Swift, Java.

Frameworks such as Meteor, Node.js, Ruby on rails, Django, Ionic, Bootstrap, .NET, Content Management Systems (CMS), Angular.js

Libraries – groups of code snippets

Database such as MongoDB, Redis, PostgreSQL, MySQL, Oracle, SQL Server, Neo4j

Elements of Web Development (2 of 2)

Client (or client-side)

Server (or server-side)

Front-end – HTML, CSS, JavaScript

Back-end – servers, databases

Protocols – HTTP, DDP, REST

API – Application Programming Interface

Data formats – JSON, XML, CSV

Git Hub

Web Hosting

Client-side scripting

Server-side scripting

Modern Web Servers

Apache

- Free and open source
- Available since 1995

Nginx

- Free and open source
- Available since 2004
- Outperforms Apache
- Runs principally on Unix/Linux

Web Application Frameworks

A software library or framework for the easy development of web applications.

Provides:

- Routing user data requests
- Data store access
- Automated web page generation

Framework 'weight'

Lightweight	Fully featured
Less rigid application layout	Highly structured layout
Flexible application design	Specific application design pattern
Functionality via plugins or user coding	Lots of functionality included

* Django is a fully featured framework

MVC

- **Model**

**Store, generate,
retrieve data**

- **View**

**Controls how to
build the user
interface**

- **Controller**

**Application
logic. Connects
the user to the
model and views**

Data Storage

Relational (SQL) Databases	Non-Relational (NoSQL) Databases
Structured and require schemas to be defined before adding any data.	Document-oriented with dynamic schema. Information with different structure can be added on-the-fly.
Data model: <ul style="list-style-type: none">• Tables with rows and columns.• Associations between tables	There are NoSQL databases with different data models: <ul style="list-style-type: none">• Key-value model• Columnar• Document database• Graph database
Widely common type of database used in business organisations to store structured data.	Use to store semi-structure or unstructured data.

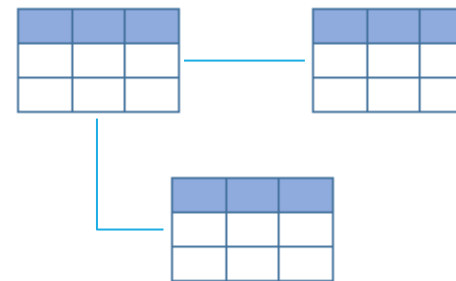
Data Storage

Relational Database:

- MySQL – smaller memory footprint
- PostgreSQL – better concurrent performance
- Two most popular free and open-source RDBMS
- Use SQL

NoSQL

- Do not use relational data modelling
- Common types
 - Document stores
 - Key-value stores
 - Graph databases



Relational Model

Introduction to Django



Introduction to Django

Official website: <https://www.djangoproject.com/>

Django is an open-source web framework written in Python. It was named after the jazz guitarist [Django Reinhardt](#). It is maintained by the [Django Software Foundation](#) (DSF) since 2008.

Django is still in active development, with [new versions](#) being released regularly.

Django has built-in support for database querying, URL mapping, and template rendering.

Django vs Flask

Django

Django is a **fully-featured** framework based on the Model-View-Template (MVT) architecture.

Django is **feature-rich** and is suitable for large-scale projects with its built-in features such as ORM and authentication system.

Django has a **steeper learning curve** due to its comprehensive nature and pre-defined conventions.

Django is well-suited for **building complex web applications** and APIs, providing a structured and scalable framework for rapid development

Flask

Flash is a **lightweight** framework designed to be simple and easy to use

Flask is **minimalistic and flexible**, allowing developers to choose components and libraries as needed.

Flask has a **simpler and more straightforward** design, allowing developers to quickly grasp the fundamentals and get started with web development.

Flask is **ideal for prototyping, smaller projects**, and applications requiring customisation and flexibility.

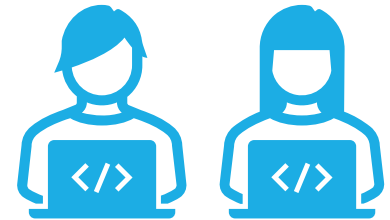
Django vs Flask

Flask vs Django: A Comparison

<https://kinsta.com/blog/flask-vs-django/>

OVERVIEW OF THE MAIN COMPONENTS OF DJANGO

Lesson 1.3



Main Components of Django

Request and Response Objects

- <https://docs.djangoproject.com/en/4.2/ref/request-response/>

Django Views

Django Models

Django Templates

Django URLs

Activity

Exercise 1.01: Set up environment

Exercise 1.02: Create First Application

Note

Install Django 4.2: `pip install Django==4.2.11`

If you install Django version 3.0.3 as shown video 1.202, you may hit an error like because because it is not compatible with the latest version of Python:

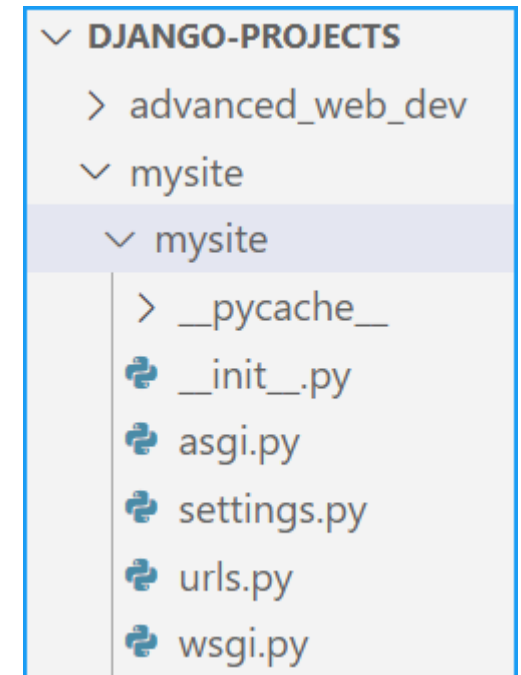
```
Installing collected packages: django
Successfully installed django-3.0.3

(advanced_web_dev) C:\PROJECT\topic1>django-admin startproject simplesite
Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "C:\PROJECT\advanced_web_dev\Scripts\django-admin.exe\__main__.py", line 4, in <module>
  File "C:\PROJECT\advanced_web_dev\Lib\site-packages\django\__init__.py", line 1, in <module>
    from django.utils.version import get_version
  File "C:\PROJECT\advanced_web_dev\Lib\site-packages\django\utils\version.py", line 6, in <module>
    from distutils.version import LooseVersion
ModuleNotFoundError: No module named 'distutils'

(advanced_web_dev) C:\PROJECT\topic1>
```

Django Auto-generated Project Files

<code>__init__.py</code>	Empty file that tells Python to treat this directory like a Python package
<code>asgi.py</code>	Configurations for ASGI web servers to communicate with Django application
<code>settings.py</code>	Settings for the Django project
<code>urls.py</code>	Global URL mappings to views and other child URL mappings
<code>wsgi.py</code>	Configurations for WSGI web servers to communicate with Django application
<code>manage.py</code>	Special file that acts as a command line tool to perform administrative tasks



settings.py

Setting	Description
BASE_DIR	Project base path
SECRET_KEY	Automatically generated value used for hashing, tokens and other cryptographic functions
DEBUG	When set to True, Django displays exceptions to the browser
INSTALLED_APPS	List of applications installed in the project
ROOT_URLCONF	Top-level URL mappings
TEMPLATES	Settings for the location of templates
DATABASES	Database configurations

Default Apps

django.contrib.admin – Administration site

django.contrib.auth – An authentication system

django.contrib.contenttypes – A framework for content types

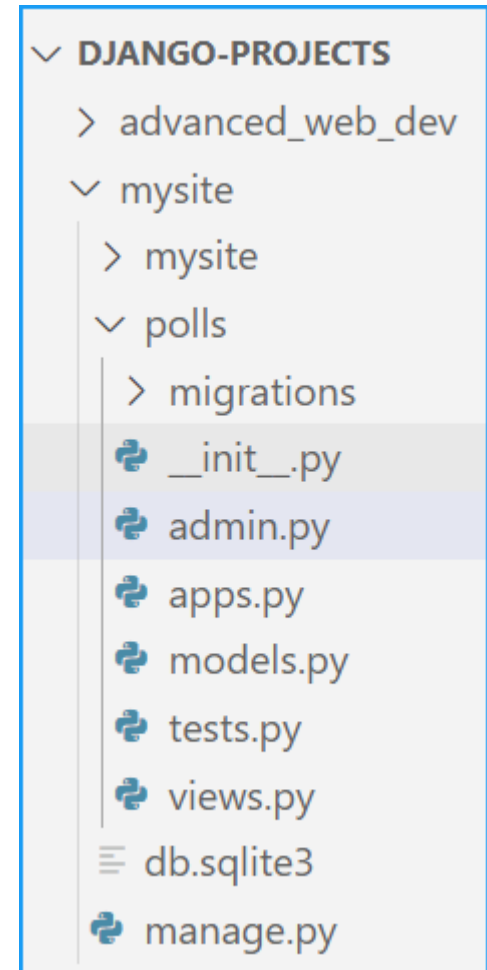
django.contrib.sessions – A session framework

django.contrib.messages – A messaging framework

django.contrib.staticfiles – A framework for managing static files

Django Auto-Generated App Files

<code>__init__.py</code>	A special file that tells Python to treat this directory like a Python package
<code>admin.py</code>	Configurations for the Django admin application
<code>apps.py</code>	Application configurations
<code>migrations</code>	Directory that contains instructions to create and modify database
<code>models.py</code>	Relational database tables defined as classes
<code>test.py</code>	Tests scripts
<code>views.py</code>	Views



Poll Application – URL Routing

How URL routing works

http://localhost:8000/polls

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

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

mysite.urls

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
    path("", views.index, name="index"),
]
```

polls.urls

```
from django.http import HttpResponse
```

```
def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

polls.views

Database

Django uses SQLite database by default

- `BASE_DIR/db.sqlite3`
- `mysite.settings.DATABASES`

Activity

Installed DB Browser for SQLite. We will use it to view the database

- <https://sqlitebrowser.org/dl/>

Exercise 1.03: Create Database

Exercise 1.04: Explore the Django Admin Site

Exercise 1.05: Create a page using Django template

Django Templates

Django has two built-in template engines:

- Django Templating Language (DTL)
- Jinja2

Django templates:

<https://docs.djangoproject.com/en/4.2/topics/templates/>

Django template language:

<https://docs.djangoproject.com/en/4.2/ref/templates/language/>

Render() Function

1

2

3

```
return render(request, 'helloworld/simple.html', {'address':first_address,'name':resident_name})
```

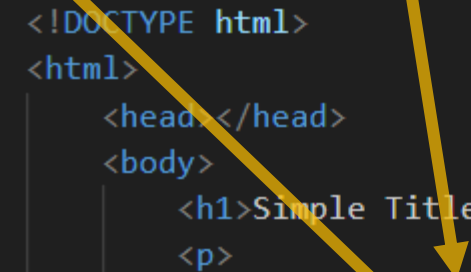
-
- 1 Requested object
 - 2 Template for creating the response
 - 3 Context object
-

Django shortcut functions:

<https://docs.djangoproject.com/en/4.2/topics/http/shortcuts/>

Render() Function

```
return render(request, 'helloworld/simple.html', {'address':first_address,'name':resident_name})
```



```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>Simple Title For This Page</h1>
    <p>
      Name: {{ name }}<br />
      Address: {{ address }}
    </p>
  </body>
</html>
```

Helloworld/simple.html

URL Dispatcher

The URL dispatcher maps URLs to Python functions in your views.

- <https://docs.djangoproject.com/en/4.2/topics/http/urls/>

Path converters

- `path('user/<int:id>', helloworld.id_lookup)` -- match 'user/29', 'user/1', etc
- `path('user/<str:name>', helloworld.name_lookup)` -- match 'user/john', 'user/ron', etc

Matching path using regular expressions.

- `re_path(r'user/(?P<id>\d+)$', helloworld.id_lookup)`
- `re_path(r'user/(?P<name>\w+)$', helloworld.name_lookup)`

Building a Lightweight Application

The point of lesson 1.402 is to show that it is not necessary to create the advanced layout auto-generated by Django (models, views, urls, etc). It is possible to create a lightweight web application in just one file. The [helloworld.py](#) file is in the folder 'lightweightsite'.

Start the site with the command: `py helloworld.py runserver`