CoGrammar

Django

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: **Feedback on Lectures**

Lecture Objectives

1. Define client-server architecture and HTTP.
2. Define the MVT Architecture and how we use it to create web applications with Django.
3. Define ORMs and how they are used to connect objects with relational databases.
4. Utilise Django's MVT architecture to build your own web applications.

Client-Server Architecture

- Network architecture that breaks down task and workloads between clients and server
- Can reside on same system or linked by a computer network
- Typically consists of multiple workstations, PCs or other devices belonging to users connected to a central server
- Connect through internet connection or other network connection

Client-Server Architecture

- Basic steps
  - Client sends request for data
  - Server accepts request
  - Server processes request
  - Send requested data back to user

Servers and Clients

- Servers
  - Not just a computer clients make requests to
  - Requires appropriate server software running to be a server

    E.g. Apache, Tomcat, Nginx
- Client
  - Not just any device making requests
  - Requires correct software to make requests
  - Most common client - Web browser
  - Your social media application is also a client

HTTP

- HyperText Transfer Protocol
- Underlying protocol of WWW
- Defines how messages are formed and transmitted between clients and server
- Defines actions clients and server must take in response to various commands

HTTP

- Basic example of HTTP implementation
  - Urls gets entered in a browser
  - Browser send HTTP command to server
  - Command directs server to search for and transmit requested page
  - Response can be HTML in this instance

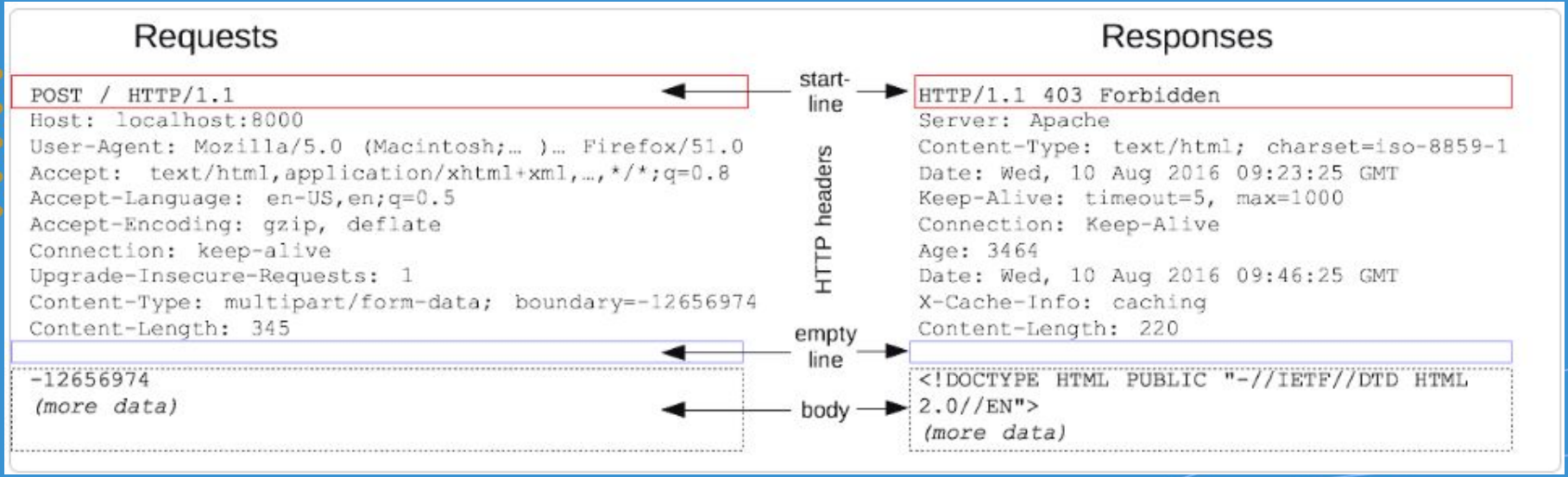
HTTP

- HTTP is a stateless protocol
- Each request is independent from the previous request
- E.g. a request is made for the first ten records in a database and then another request is made for the next ten records
- Stateful protocol
  - Give me the first 10 records
  - Give me the next 10 records
- Stateless protocol
  - Give me records 1-10
  - Give me records 11-20

HTTP messages

- Used for requests and responses
- Composed of textual information encoded in ASCII and spans multiple lines
- Consists of
  - Start line
  - Headers
    - General
    - Request
    - Representational
  - Body

HTTP messages



Status Codes

- Short notes tacked onto a webpage
- Not part of the site's content but messages telling us how things went
- Returned every time your browser interacts with a server
- Helps diagnose and fix website configuration

Status Codes

5 Classes of status codes

- 100s
  - Informational code
  - Indicates request initiated in continuing
- 200s
  - Success code
  - Indicates request was received, understood and processed

Status Codes

- 300s
  - Redirection codes
  - When a new resource in substituted for the requested resource
- 400s
  - Client Error
  - Problem with request
- 500s
  - Server error
  - Request was accepted but a server error has occurred

What is Django?

- Open-source web framework
- Used for developing secure and scalable websites and web applications
- Platforms using Django: Instagram, Spotify, Youtube and many more

Why Use Django?

- Has a large list of libraries and tools
- Allows for the creation of robust data driven applications.
- Code is fast to implement and is very clean and pragmatic

Model-View-Template (MVT) Architecture

- Variation of Model-View-controller architecture
- Three main components
  - **Model**: Represents the business logic and data structure of the application.
  - **View**: Handles the interaction between the user and the application, managing the presentation logic.
  - **Template**: Deals with the presentation layer, defining the structure and appearance of the HTML content.

Object Relational Mapping(ORM)

- Used to connect OOP to relational databases.
- Allows for the implementation of CRUD operations in your objects
- CRUD: Create, Read, Update, Delete

Model

- Models represent the structure of the application's data.
- Define models as Python classes, each representing a table in the database.
- Use Django's ORM to interact with the database, abstracting SQL queries.
- Model classes are created in models.py.
- When changes are made to model classes you will have to synchronise your changes with the database using database migration.

Views

- Views are Python functions we create in the views.py file.
- Views define the behaviour of our URL patterns.
- Views handle user requests and define the logic for processing them.
- They interact with models to retrieve or update data.
- Views return appropriate HTTP responses, such as rendering templates or redirecting.

Templates

- Templates define the structure of the HTML pages.
- They incorporate dynamic data using template tags.
- They receive data from views through context dictionaries.
- Templates are stored in the templates directory.
- HTML pages are constructed using template tags for data integration.

Component Interaction

- Model-View interaction:
  - The View interacts with the Model to retrieve data needed for presentation.
  - The View can modify data in the Model based on user interactions.
- View-Template interaction:
  - The View updates the Template with data from the Model.
  - The Template handles user input and triggers actions in the View.
- Template-Model interaction:
  - The Template can update the Model indirectly through user interactions.
  - The Template reflects changes in the Model by dynamically updating the HTML content.

Poll:

Assessment

Wrapping Up

Django

Django is an open-source web framework that is used for developing secure and scalable websites and web applications

MVT-Architecture

Software design pattern with 3 components Model, view and template.

Object Relational Models

Used to connect OOP to relational databases and allows for the implementation of CRUD operations in your objects

CoGrammar

Questions

CoGrammar

Thank you for joining