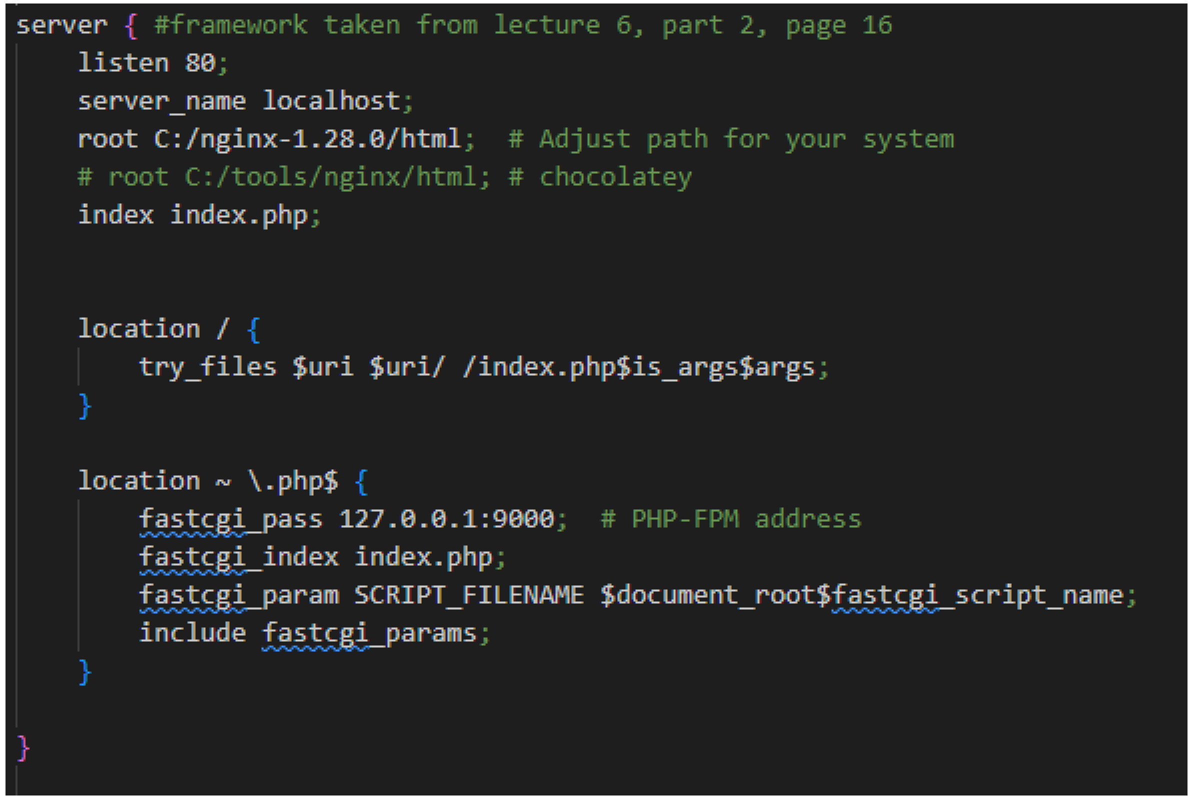
Setting up my APIs with NGINX, PHP, and MySQL

Eurydice Lunnemann

NGINX

- make sure you have NGINX installed in your C drive
- make sure your nginx.conf file server block matches the one shown below:



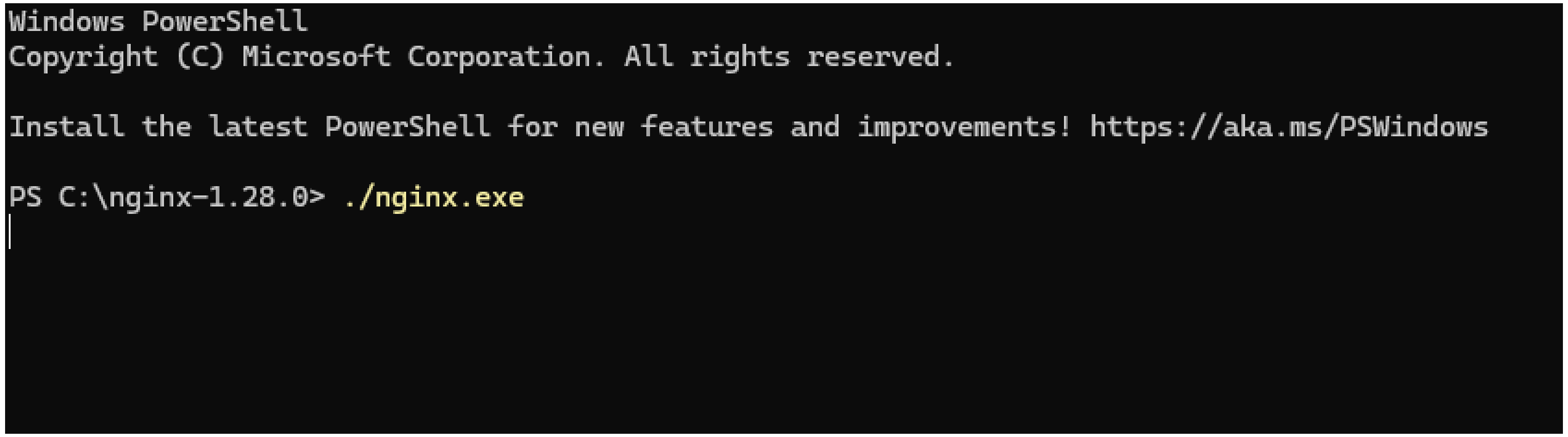
```
server { #framework taken from lecture 6, part 2, page 16
    listen 80;
    server_name localhost;
    root C:/nginx-1.28.0/html;  # Adjust path for your system
    # root C:/tools/nginx/html; # chocolatey
    index index.php;


    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ \.php$ {
        fastcgi_pass 127.0.0.1:9000;  # PHP-FPM address
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

}
```

- next, open a terminal in your NGINX folder
- run "./nginx.exe" to start your server



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\nginx-1.28.0> ./nginx.exe
```

PHP

- make sure you have php installed in your C drive
- run ".\php-cgi.exe -b 127.0.0.1:9000 -c C:\php\php.ini" to start your php server
- php traffic should automatically be routed to this server because of the NGINX config

MySQL

- make sure your MySQL service is running as MySQL80
- either through the GUI or CLI, add the following schemas and tables
    - public_apis
        - announcements (id, announcment, location, date_announced)
        - arts (id, name, size, artist)
        - audios (id, name, size, metadata)
        - candidates (id, name, votes, position)
        - dates (id, day, time, event)
        - grades (id, name, Math, English, Social_Studies)

- continued
    - public_apis continued
        - iot_devices (id, device_type, device_description, is_online)
        - stock_tickers (ticker_symbol, full_name, market_value, is_up)
    - private_apis
        - employees (id, name, tenure, department)
        - homeworks (id, name, questions, class)

Finishing and Acknowledgement

- assuming you set up all of your services correctly, you should be able to use my APIs seamlessly
- Acknowledgement: much of the content in these slides is a shortened version of content that can be found here

My APIs and their Endpoints

Eurydice Lunnemann

API 1 - IoT Device Manager

- Purpose: manage and view the status of different IoT devices
- Endpoint 1:
  - GET /iot-devices
  - Used to get all devices
  - Good for bulk auditing
- Endpoint 2:
  - GET /iot-devices/${id}
  - Used to get one device at a time
  - Saves on processing for the client if they only need to view one device

- Endpoint 3:
  - POST /iot-devices
  - Used to create a new device in the database
- Endpoint 4:
  - PUT /iot-devices/update/${id}
  - Used to update one device
- Endpoint 5:
  - PUT /iot-devices/toggle/${id}
  - Used to toggle the "is_online" value of one device
  - Good for quickly changing the state of a device without the overhead of an update

- Endpoint 6:
    - DELETE /iot-devices/{id}
    - Used to delete a device
- For more examples please go to code/tests/testdevi.html

API 2 - Announcement Manager

- Purpose: manage announcements that would go out to an organization
- Endpoint 1:
  - GET /announcements
  - Used to get all announcements
  - Good for a messaging board where all announcements need to be visible
- Endpoint 2:
  - GET /announcements/${id}
  - Used to get one announcement

- Endpoint 3:
  - POST /announcements
  - Used to create an announcement
  - The date parameter is set as the current date and time
- Endpoint 4:
  - PUT /announcements/{id}
  - Used to update an announcement
  - The date parameter is updated to the current time
- Endpoint 5:
  - DELETE /announcements/{id}
  - Used to delete an announcement
- For more examples please got to code/tests/testann.html

API 3 - Stock Ticker Manager

- Purpose: manage stock tickers that would go on a website dashboard or led display
- Endpoint 1:
  - GET /stock-tickers
  - Used to get all tickers at once
  - Good for a display where everything can be shown at the same time
- Endpoint 2:
  - GET /stock-tickers/{symbol}
  - Used to get one ticker at a time based on symbol (AAPL, etc)
  - Good for customizing a dashboard

- Endpoint 3:
  - POST /stock-tickers
  - Used to create a ticker
- Endpoint 4:
  - PUT /stock-tickers/{symbol}
  - Used to update a ticker
  - The "is_up" parameter is updated based on if the market value has gone up
- Endpoint 5:
  - DELETE /stock-tickers/{symbol}
  - Used to delete a ticker
- For more examples please got to code/tests/testtick.html

API 4 - Audio File Manager

- Purpose: manage data about audio files
- Endpoint 1:
  - GET /audios
  - Used to get all auidios at once
- Endpoint 2:
  - GET /audios/{id}
  - Used to get one audio at a time

- Endpoint 3:
    - POST /audios
    - Used to create an audio
- Endpoint 4:
    - PUT /audios/{id}
    - Used to update an audio
- Endpoint 5:
    - DELETE /audios/{id}
    - Used to delete an audio
- For more examples please got to code/tests/testaud.html

API 5 - Art File Manager

- Purpose: manage data about art files while crediting the artists who made them
- Endpoint 1:
  - GET /arts
  - Used to get all art files at once
  - good for an art hosting website
- Endpoint 2:
  - GET /arts/{id}
  - Used to get one art file at a time

- Endpoint 3:
    - POST /arts
    - Used to create an art file
- Endpoint 4:
    - PUT /arts/{id}
    - Used to update an art file
- Endpoint 5:
    - DELETE /arts/{id}
    - Used to delete an art file
- For more examples please got to code/tests/testart.html

API 6 - Political Candidate Manager

- Purpose: manage data about political candidates
- Endpoint 1:
  - GET /candidates
  - Used to get all candidates at once
  - good for an election website where votes need live updates
- Endpoint 2:
  - GET /candidates/{id}
  - Used to get one candidate at a time

- Endpoint 3:
  - POST /candidates
  - Used to create a candidate
- Endpoint 4:
  - PUT /candidates/{id}
  - Used to update a candidate
- Endpoint 5:
  - DELETE /candidates/{id}
  - Used to delete a candidate from the running
- For more examples please got to code/tests/testcand.html

API 7 - Calendar Manager

- Purpose: manage important dates
- Endpoint 1:
  - GET /dates
  - Used to get all events at once
  - good for a website like Google Calendar where a bunch of dates and events are visible
- Endpoint 2:
  - GET /dates/{id}
  - Used to get one event at a time

- Endpoint 3:
  - POST /dates
  - Used to create an event
- Endpoint 4:
  - PUT /dates/{id}
  - Used to update an event
- Endpoint 5:
  - DELETE /dates/{id}
  - Used to delete an event
- For more examples please got to code/tests/testdate.html

API 7 - Calendar Manager

- Purpose: manage important dates
- Endpoint 1:
  - GET /dates
  - Used to get all events at once
  - good for a website like Google Calendar where a bunch of dates and events are visible
- Endpoint 2:
  - GET /dates/{id}
  - Used to get one event at a time

- Endpoint 3:
  - POST /dates
  - Used to create an event
- Endpoint 4:
  - PUT /dates/{id}
  - Used to update an event
- Endpoint 5:
  - DELETE /dates/{id}
  - Used to delete an event
- For more examples please got to code/tests/testdate.html

API 8 - Student Grade Manager

- Purpose: manage student grades
- Endpoint 1:
  - GET /grades
  - Used to get all students' grades at once
  - Good for a teacher view
- Endpoint 2:
  - GET /grades/{id}
  - Used to get one student's grade at a time
  - Good for a student view

- Endpoint 3:
    - POST /grades
    - Used to create a student
- Endpoint 4:
    - PUT /grades/{id}
    - Used to update a student's grades
- Endpoint 5:
    - DELETE /grades/{id}
    - Used to delete a student
- For more examples please got to code/tests/testgrad.html

Bearer API 1 - Employee Manager

- Purpose: manage employee data as an admin
- credentials: admin, admin
- Endpoint 1:
  - GET /employees
  - Used to get all employees at once
- Endpoint 2:
  - GET /employees/{id}
  - Used to get one employee at a time

- Endpoint 3:
    - POST /employees
    - Used to create an employee
- Endpoint 4:
    - PUT /employees/{id}
    - Used to update an employee
- Endpoint 5:
    - DELETE /employees/{id}
    - Used to delete an employee
- For more examples please got to code/tests/testemp.html

Bearer API 2 - Student Homework Manager

- Purpose: manage student homework as a teacher
- credentials: admin, admin
- Endpoint 1:
  - GET /homeworks
  - Used to get all assigned homework at once
  - Good for course load planning
- Endpoint 2:
  - GET /homeworks/{id}
  - Used to get one piece of homework at a time

- Endpoint 3:
  - POST /homeworks
  - Used to create a piece of homework
- Endpoint 4:
  - PUT /homeworks/{id}
  - Used to update a piece of homework
- Endpoint 5:
  - DELETE /homeworks/{id}
  - Used to delete a piece of homework
- For more examples please got to code/tests/testhom.html

Acknowlegement

- A lot of the code in this project comes from this GitHub
- More specific references are in the individual code files