Welcome

Fundamentals of API Design Day 1

Hello...

About me...

- Senior Software Engineer
- Work in Aerospace
- UCI Instructor
- Dad, Beards, Video Games, Reading, Vikings

Prerequisites

This course assumes you

- Have been introduced to how to program in Python
- Have never developed an API in Python and are new to the terminology and paradigms used in API development

Please Note

If you have never programmed before or if you've only used Bash, Powershell, SQL or similar, this is probably not the right class for you.

Please chat me so we can get you to the right resource.

Why study this subject?

- Level up your knowledge in Python and RESTful API’s
- Familiarize yourself with API vocabulary and best practices
- Contribute to Starbucks API Development

We teach over 400 technology topics

You experience our impact on a daily basis!

My pledge to you

I will...

- Make this interactive
- Ask you questions
- Ensure everyone can speak
- Create an inclusive learning environment
- Use an on-screen timer for breaks

…also, if you have an accessibility need, please let me know

Objectives

At the end of this course you will be able to:

- Explain the purpose of APIs
- Define the key terms in API development
- Differentiate between different types of APIs and API protocols
- Build contracts into your APIs
- Describe RESTful API and its standards
- Discuss the API lifecycle
- Understand how security is important for APIs

Agenda

- APIs: Purpose and Nomenclature
- API Types
- Introduction to Flask, OpenAPI, Swagger and REST
- API Contracts
- Documenting APIs
    - Intro to OpenAPI/Swagger
    - Creating Swagger UI and CRUD Operations

How we’re going to work together

- You'll have a copy of all the course materials – we will be using Flask and OpenAPI (explained shortly)
- You will be following along and…
  - doing coding exercises/labs alongside the material presented
  - Using the cloud based environment provided with all installs required

Student Introductions

- Job title?
- Where are you based?
- Experience with Python (or other programming languages)?
- Why learn about API's?
- Fun fact?

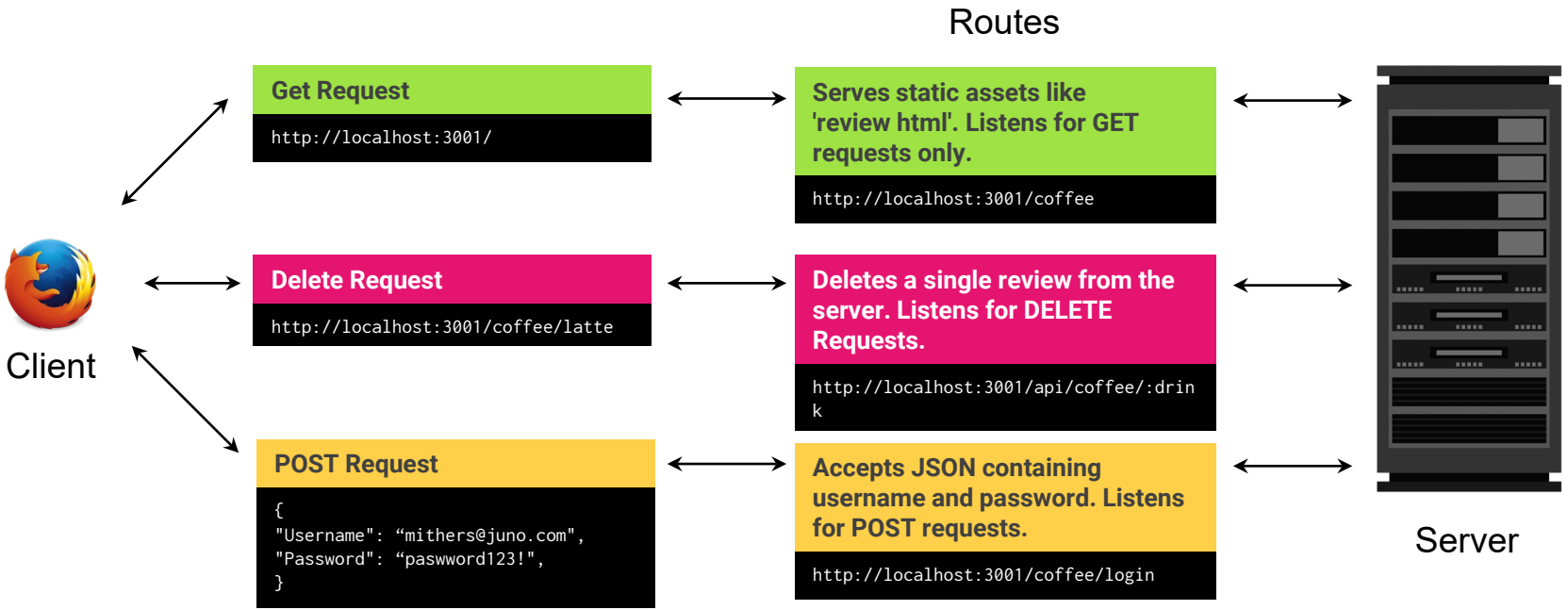
Setup Check

- Log in to Virtual Environment
- Clone Repo
- Check Versions of software

What is an API?

- API stands for Application Programming Interface.
- Application refers to any software with a distinct function
- Interface can be thought of as a contract between two applications
- This contract defines how the two applications communicate with each other using requests and responses.

What is an API?



What are the benefits of APIs?

- Efficiency
- Automation
- Integration
- Future Ready

What are the disadvantages of APIs?

- Security
- Implementation
- Maintennance

Key Terms

- API
- API Call
- API Endpoint
- API Gateway
- API Keys
- API Layer
- API Portal
- API Request
- Parameters
- API Security
- CRUD
- POST
- GET
- DELETE
- PUT
- REST
- OpenAPI
- JSON
- SOAP

API Types

- Open or Public
- Partner
- Internal or Private

Intro to Flask

Let’s Code

REST – What is it?

- **Stateless:** The server won't maintain any state between requests from the client.
- **Client-server:** The client and server must be decoupled from each other, allowing each to develop independently.
- **Cacheable:** The data retrieved from the server should be cacheable either by the client or by the server.
- **Uniform interface:** The server will provide a uniform interface for accessing resources without defining their representation.
- **Layered system:** The client may access the resources on the server indirectly through other layers such as a proxy or load balancer.
- **Code on demand (optional):** The server may transfer code to the client that it can run, such as JavaScript for a single-page application.

CRUD

- Create
- Read
- Update
- Delete

- POST
- GET
- PUT
- DELETE

Thank you!

DevelopIntelligence
A PLURALSIGHT COMPANY

Welcome

Fundamentals of API Design Day 2

Agenda

- Git Pull Pick up where we left off
- API Lifecycle
  - API Versioning
- API Security
  - API Keys
  - OAuth
- API Ecosystem
  - API First
  - DevOps
  - TDD
  - Scalability

API Versioning

- **When Should You Change the Version Number?**
- REST doesn't provide for any specific versioning guidelines, but the more commonly used approaches fall into three categories:
  - URI Versioning
  - Custom Request Headers
  - "Accept" Header

API Lifecycle

- Planning and Designing
- Developing
- Testing
- Deploying
- Retiring

API Security

- Keep It Simple
- HTTPS
- Password Hash
- Timestamps in Requests
- Input Parameter Validation
- API Tokens
- Basic Auth

DevOps/Scalability

- Contract First Approach
- Semantic Versioning
- API-Management-as-Code
- Ensure Testability
    - Unit Tests (Our Focus Today)
    - Integration Tests
    - Acceptance Tests
    - End to End Tests
    - Performance Tests