Contents

Chapter 1

Making Web Requests with HTTP

The Hypertext Transfer Protocol (HTTP) is the protocol used for transmitting hypertext over the World Wide Web. It is the foundation of any data exchange on the web, and it is a protocol used for transmitting hypertext requests from clients (like a user's browser) to servers, which respond with the requested resources.

1.1 HTTP Requests

An HTTP request is made up of several components:

- **Method**: The HTTP method, like GET (retrieve data), POST (send data), PUT (update data), DELETE (remove data), and so on.
- **URL**: The URL of the resource to retrieve, send data to, update or delete.
- **Headers**: Additional information about the request or response, like the content type of the body.
- **Body**: The body of the request, used when sending data in POST or PUT requests.

1.2 Using HTTP with Web-Based APIs

Software developers often use HTTP to interact with web-based APIs. An Application Programming Interface (API) is a set of rules that allows programs to talk to each other. The developer creates the API on the server and allows the client to talk to it.

When a developer makes a request to an API endpoint, they're asking the server to either send them some data or receive some data from them. The response from the server will often be in a format like JSON or XML, which the developer can then use in their own application.

For example, a developer might make a GET request to '`https://api.example.com/users`' to retrieve a list of all users. The server would respond with a list of users in a format like JSON.

Chapter 2

Using and Creating APIs

As a software engineer, you are likely familiar with building applications that interact with various external services and data sources. One of the most common methods for communication and integration is through HTTP APIs (Application Programming Interfaces). HTTP APIs provide a standardized way for applications to exchange data and functionality over the internet.

This chapter will introduce you to the world of HTTP APIs and explore how you can leverage them in your software development projects. We will cover the fundamental concepts, techniques, and best practices for effectively working with HTTP APIs.

An HTTP API allows two software systems to communicate and exchange information using the Hypertext Transfer Protocol (HTTP). It enables your application to make requests to an API server and receive responses in a structured format, such as JSON (JavaScript Object Notation) or XML (eXtensible Markup Language).

Using HTTP APIs offers a range of benefits for software engineers. It allows you to leverage external services and data sources, enabling your application to access functionality or retrieve valuable information from third-party systems. This opens up opportunities for integration with popular platforms, social media networks, payment gateways, geolocation services, and much more.

Throughout this chapter, we will explore various aspects of working with HTTP APIs, including:

- API endpoints and methods: Understanding how to interact with an API involves identifying the available endpoints and the supported methods, such as GET, POST, PUT, DELETE, and so on. We will discuss how to construct API requests and handle different response formats.
- Authentication and authorization: Many APIs require authentication to ensure secure access and protect sensitive data. We will delve into different authentication mechanisms, including API keys, tokens, OAuth, and other authentication protocols commonly used in API integrations.
- Request parameters and payloads: APIs often accept additional parameters or payloads to customize the request or send data for processing. We will explore how to pass query parameters, request headers, and request bodies when interacting with APIs.
- Error handling and status codes: Learning how to handle errors and interpret status

codes returned by APIs is crucial for building robust and resilient applications. We will discuss common status codes and best practices for handling various scenarios gracefully.

- Rate limiting and throttling: Many APIs impose restrictions on the number of requests you can make within a given timeframe to prevent abuse and ensure fair usage. We will cover techniques for handling rate limiting and implementing efficient strategies to manage API quotas.
- API documentation and testing: Proper documentation is essential for understanding an API's capabilities, endpoints, and expected behavior. We will explore how to read and interpret API documentation, as well as techniques for testing and validating API integrations.

By mastering the art of using HTTP APIs, you will expand your development toolkit and gain the ability to seamlessly integrate your applications with external services, leverage their functionalities, and build powerful, interconnected systems.

So, let's dive into the world of HTTP APIs and uncover the endless possibilities they offer for enhancing your software engineering projects.

FIXME talk about API keys, API requests, how it interacts with HTTP requests, why keys should not be shared

Chapter 3

Data Compression and Decompression

Data compression and decompression are fundamental techniques used in modern computing, enabling efficient storage and transmission of data. The concept of entropy, borrowed from the field of information theory, plays a crucial role in determining the compression rate.

3.1 Data Compression and Decompression

Data compression is the process of reducing the amount of data needed to represent a particular set of information. The two main types of data compression are lossless and lossy. Lossless compression ensures that the original data can be perfectly reconstructed from the compressed data, whereas lossy compression allows some loss of data for more significant compression rates. Decompression is the reverse process of compression, reconstructing the original data from the compressed format.

3.2 Entropy

In information theory, entropy measures the unpredictability or randomness of information content. More specifically, it quantifies the expected value of the information contained in a message. Lower entropy implies less randomness and more repetitiveness, which in turn means the data can be compressed more.

3.3 Entropy and Compression

The role of entropy in data compression is fundamental. The entropy of a source of data is the minimum number of bits required, on average, to encode symbols drawn from the source. It serves as a lower bound on the best possible lossless compression rate.

For a source X with probability distribution $p(x)$, the entropy $H(X)$ is defined as:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{3.1}$$

If the entropy of the data is high (i.e., the data is random and unpredictable), the potential for compression is low. On the other hand, if the entropy is low (the data is predictable), the data can be compressed to a smaller size.

Chapter 4

Dealing with JSON and XML

Appendix A

Answers to Exercises

Index