TASK

Network Protocols and System Architecture

Visit our website

Introduction

WELCOME TO THE NETWORK PROTOCOLS AND SYSTEM ARCHITECTURE TASK!

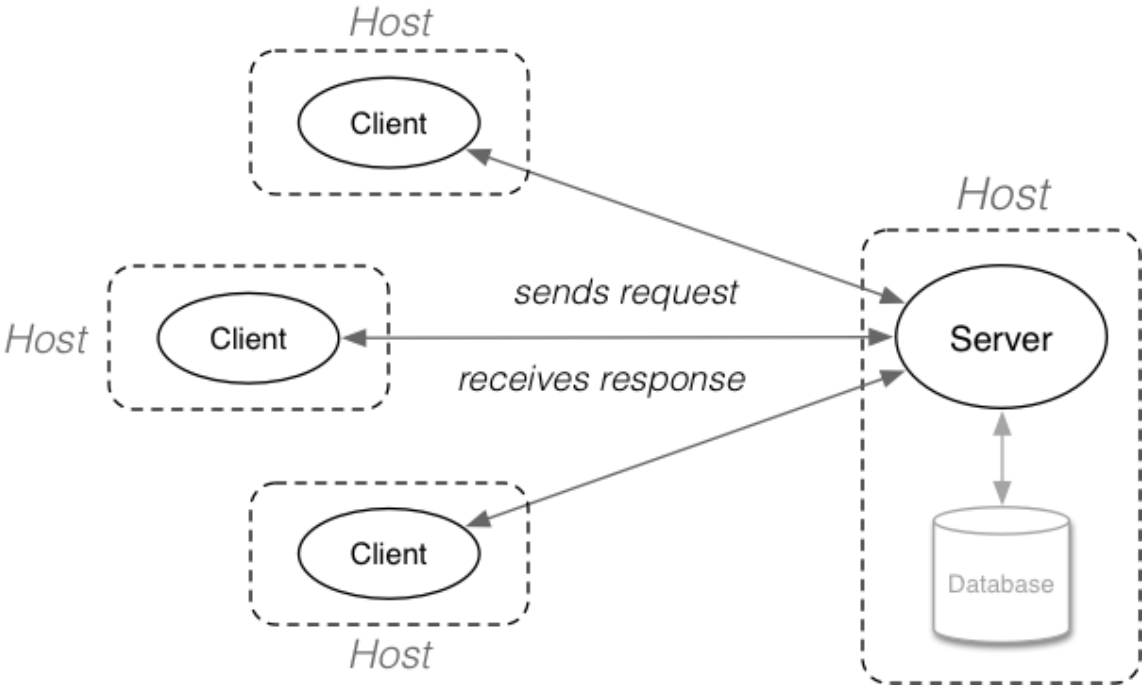
In this task, we will be exploring how computers communicate with each other over networks such as the Internet using the Hypertext Transfer Protocol (HTTP) protocol and the client-server architecture. This is a very important concept for web development. Enjoy!

WHAT IS CLIENT-SERVER ARCHITECTURE?

Client-server architecture, also known as a client-server model, is a network architecture that breaks down tasks and workloads between clients and servers that reside on the same system or are linked by a computer network such as an intranet or the Internet.

Client-server architecture typically features multiple workstations, PCs, or other devices belonging to users, connected to a central server via an Internet connection, or other network connection. The client sends a request for data, and the server accepts and processes the request, sending the requested data back to the user who needs it on the client-side.

For the purpose of this bootcamp we will be focusing on how this model relates to web development in particular.



The above illustration shows how clients interact with a server and how that server interacts with the database so that data is processed by a server and served back to the clients.

It is not entirely accurate to say that a server is a computer that clients make requests to. A computer needs to have appropriate server software running on it in order to make it a server. Examples of server software are **Apache**, **Tomcat**, and **Nginx**. In this course, you will learn to use Node.js and Express.js for setting up a web server.

A client also does not just refer to a device that makes the request to a server, but as with a server, a client needs the appropriate software to make requests. The most common client that you will have come across in your everyday life is your web browser! However, there are other clients as well, such as your Netflix application when you are streaming videos. Here the application is the client and there is a server that is serving the video data to the client.

A client can therefore be any device that has the software necessary to communicate with the server.

WHAT IS HTTP?

HTTP is an underlying protocol used by the World Wide Web. A protocol is basically a set of rules that are decided on and adopted so that there is consistency in how to perform particular tasks. HTTP defines how messages are formed and transmitted between clients and servers, and what actions web servers and clients should take in response to various commands.

A simple example of how HTTP is implemented is when a URL is entered into the browser, and the browser sends an HTTP command to the web server directing it to search for and transmit the requested web page. The response would in this case be an HTML file that the browser can interpret and display to the user.

The HTTP protocol is a ***stateless*** one. This means that every HTTP request the server receives is independent and does not relate to requests that came prior to it. For example, imagine the following scenario: a request is made for the first ten user records from a database, and then another request is made for the next ten records. These requests would be unrelated to each other.

With a ***stateful*** protocol, the server remembers each client position inside the result-set, and therefore the requests will be similar to the following:

- Give me the first ten user records
- Give me the next ten records

With a stateless protocol, however, the server doesn't hold the state of its client, and therefore the client's position in the result-set needs to be sent as part of the requests, like this:

- Give me user records from user 1 to user 10
- Give me user records from user 11 to user 20

THE REQUEST-RESPONSE CYCLE

As we start to build out web applications, it is essential to be able to visualise the way information flows through the system, typically called the Request/Response Cycle.

First, a user gives a client a URL, and the client builds a request for information (or resources) to be generated by a server. The request is sent from the client to the server. When the server receives that request, it uses the information included in the request to build a response that contains the requested information. Once created, that response is sent back to the client in the requested format, to be rendered to the user.

It is our job as web developers to build out and maintain servers that can successfully build responses based on standardised requests that will be received from clients. But, what does a standard request look like? We need to know that before we can start building servers that will respond successfully!

THE HTTP MESSAGE

HTTP messages are used for the requests and responses. HTTP messages are composed of textual information encoded in ASCII, and span multiple lines.

Web developers, or webmasters, rarely craft these textual HTTP messages themselves (although this can be done): software, a Web browser, proxy, or Web server perform this action.

Take note:

The Mozilla Developer Network documentation breaks down the anatomy of an **HTTP message** very well.

THIS IS COMPULSORY READING FOR THIS TASK.

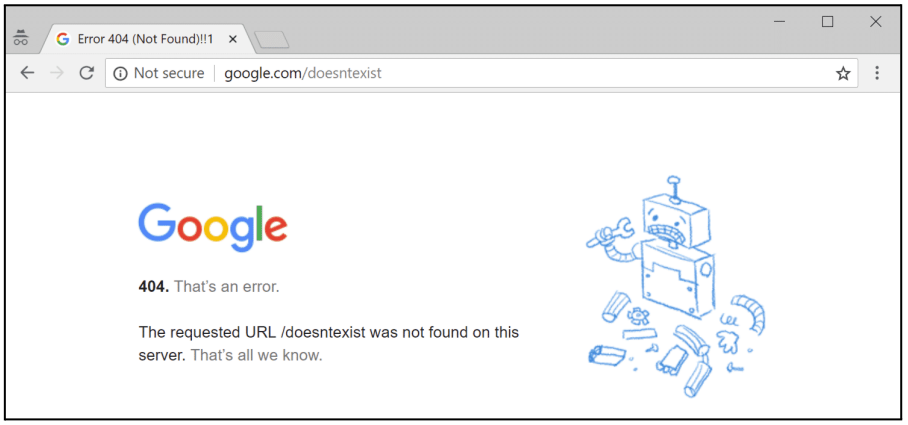
Please read the resource before moving on!

STATUS CODES

HTTP status codes are like short notes from a server that get tacked onto a web page. They're not actually part of the site's content. Instead, they're messages from the server letting you know how things went when it received the request to view a certain page.

These kinds of messages are returned every time your browser interacts with a server, even if you don't see them. If you're a website owner or developer, understanding HTTP status codes is critical. When they do show up, HTTP status codes are an invaluable tool for diagnosing and fixing website configuration errors.

HTTP status codes are delivered to your browser in the HTTP header. They are returned with an HTTP response message every time a request is made.
It's usually only when something goes wrong that you might see one displayed in your browser. A common status code that you may have discovered in your daily dealings in a browser is the **404 page not found** error.



HTTP Status codes fall into 5 classes:

- 100s: Informational codes indicating that the request initiated by the browser is continuing.
- 200s: Success codes returned when the browser request was received, understood, and processed by the server.
- 300s: Redirection codes returned when a new resource has been substituted for the requested resource.
- 400s: Client error codes indicating that there was a problem with the request.
- 500s: Server error codes indicating that the request was accepted, but that an error on the server prevented the fulfilment of the request.

Take note:

Familiarise yourself with the types of **HTTP response status codes**.

THIS IS A COMPULSORY READING FOR THIS TASK.

Please read the resource before moving on.

MIME TYPES

A multipurpose internet mail extension, or MIME type, is an Internet standard that describes the contents of Internet files based on their natures and formats. This cataloging helps the browser open the file with the appropriate extension or plugin and also helps a server identify how to process a specific file if it is sent from client to server. Although the term includes the word "mail" for electronic mail, it's used for web pages, too.

MIME types contain two parts: a type and a sub-type.

The type describes the categorisation of MIME types linked to each other.
In contrast, a subtype is unique to a specific file type that is part of the larger type category.

The syntax of the structure is : type/subtype

A MIME type is insensitive to text case, but traditionally is written all in lower case.

Discrete types indicate the category of the document; this can be one of the following:

Type	Description	Example of typical subtypes
`text`	Represents any document that contains text and is theoretically human readable	`text/plain`, `text/html`, `text/css`, `text/javascript`
`image`	Represents any kind of images. Videos are not included, though animated images (like animated gif) are describes with an image type.	`image/gif`, `image/png`, `image/jpeg`, `image/bmp`, `image/webp`
`audio`	Represents any kind of audio files	`audio/midi`, `audio/mpeg`, `audio/webm`, `audio/ogg`, `audio/wav`
`video`	Represents any kind of video files	`video/webm`, `video/ogg`
`application`	Represents any kind of binary data.	`application/octet-stream`, `application/pkcs12`, `application/vnd.mspowerpoint`, `application/xhtml+xml`, `application/xml`, `application/pdf`

Here is a **list of all MIME types**.

Compulsory Task 1

Create a file named **answers.txt** and answer the following questions. Please note that some research may need to be done to answer these questions, and you will be required to read through the resources linked in this document as well. This is a vital part of the learning process, not something aside from it, and if done properly will significantly improve your understanding of the topic.

1. The "P" in HTTP means protocol. Explain in your own words what a protocol is.
2. HTTP is built on top of which protocol?
3. If a request is successful, what will the status code of the response be?
4. What is a stateless protocol?
5. Which of the following are valid MIME types?

a. text/time
b. image/jpeg
c. text/javascript
d. text/jsx
e. image/psd
f. text/calendar

6. Which status message can you expect if a server does not allow you permission to request a specific resource?

Completed the task(s)?

Ask an expert code reviewer to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.