



TASK

HTTP Requests and Responses

Visit our website

Introduction

WELCOME TO THE HTTP REQUESTS AND RESPONSES TASK!

This task will delve into front-end and back-end communication through the essential concepts of HTTP requests and responses. You will gain a comprehensive understanding of the underlying workings of HTTP (HyperText Transfer Protocol), the backbone of the World Wide Web, while exploring the intricacies of client-server interactions through HTTP messages. By grasping the structured format of HTTP messages, you will recognise the mechanisms responsible for the seamless interactions between clients and servers!

OVERVIEW OF HTTP

HTTP is an underlying protocol used by the World Wide Web. A protocol is a set of rules that are decided on and adopted to perform particular tasks consistently. 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.

CLIENT-SERVER COMMUNICATION AND HTTP MESSAGES

HTTP also refers to the communication between clients and servers. In this context, a **client** is a web browser. HTTP messages help exchange data between a client and the server. These messages are exchanged in a request-response format. In other words, an HTTP response message is the reply to an HTTP request message. These messages contain essential information for the client and server to communicate with each other effectively.

HTTP messages are written in several lines and contain information encoded in ASCII. They require a secure Transmission Control Protocol (TCP) connection. A TCP is a transport layer protocol used in computer networking that guarantees dependable and connected communication, ensuring precise delivery and

sequencing of data. The newer version of HTTP requires messages to be sent in frames with optimised performance.

The illustration below 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.

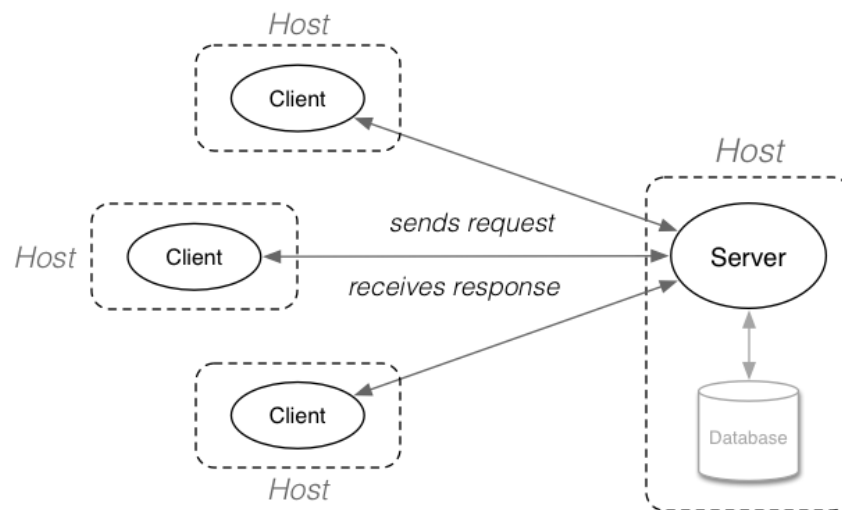


Figure 1: Client-Server Architecture

Image source:

<https://www.cse.unsw.edu.au/~cs1521/18s2/lectures/week08/slide065.html>

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 bootcamp, 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.

Next, we will dive into the composition of HTTP requests and responses and how they are structured.

HTTP REQUEST

The client sends an HTTP request message to the server to request a specific action, HTML page, or resource. For example, if a browser, i.e., the client requests an HTML page, then the server returns an HTML file. An HTTP request is composed of methods, URLs, headers, and a body. Let's look a little deeper into each of these elements.

1. Methods

The method of an HTTP request is present in the request line. There are several HTTP request methods, such as PUT, POST, GET, DELETE, HEAD, PATCH, etc. These methods target the URL of the resource or its path. The function of each method is unique and different from the others. The explanation of a few methods is as follows:

- **GET** retrieves resources from a server, such as web pages, images, documents, etc. It does not modify any resources on the server. It only reads the data. The data sent in with the GET request is typically limited in size and sent in the URL.
- **POST** methods can modify the data on the server. It can create, edit, and delete any object or subject on the site. It is commonly used to create new resources, update existing resources, or modify the server's state. It sends sensitive data not exposed in the URL, such as passwords. Servers also hold the power to control access to POST endpoints to enforce authentication.
- **DELETE** permanently removes a specific resource on the server. When a delete request is made, the client instructs the server to delete a resource identified by a URL. It contains no response body.

In the context of an API (Application Programming Interface), an endpoint refers to a specific URL (Uniform Resource Locator) or URI (Uniform Resource Identifier) that an application or client can use to interact with a particular resource or perform a specific action. Endpoints serve as the entry points to access various functionalities and data provided by the API.

Each endpoint is associated with a specific HTTP method (such as GET, POST, PUT, DELETE, etc.), which determines the type of operation that the client can perform on the resource. When a client makes an HTTP request to an API endpoint, the API server processes the request and responds with the appropriate data or performs the requested action.

2. URL

URL, or Uniform Resource Locator, is a specific web address used to locate resources on the web using the HTTP protocol. It consists of several components which combine to form a query string. It contains:

- **Protocol:** specified as “http://.”
- **Hostname:** identifies the server's domain or IP (Internet Protocol) address.
- **Port number (optional):** tells the port number to establish a connection with the server.
- **Path:** specifies the path or location of the resource on the server.
- **Query parameters (optional):** additional information, such as key-value pairs.
- **Fragment identifier (optional):** indicates a fragment inside the resource.

The overall format of the URL becomes:

```
http://<hostname>:<port>/<path>?<query_parameters>#<fragment_identifier>
```

3. Request Headers

HTTP request headers provide excess information regarding the request to the server and are found in key-value pairs. There are several kinds of headers, such as:

- **Host:** specifies the hostname and port number of the server.
- **User-agent:** used to identify the client who has made the request.
- **Accept:** indicates the type of media that the client accepts.
- **Content type:** indicates the type of media that can be included in the response body.
- **Cookie:** contains cookies to keep information about the session.
- **Referrer:** specifies the page URL that directs the client to the resource.
- **Accept-Language:** specifies how the client can accept a response.
- **Authorization:** gives credentials to access private resources on the server.

4. Request Body

The body of the request is an optional part of the request. It sends data with the request in several methods, such as **POST**. The request body consists of data in the form of objects, documents, etc.

There might be restrictions on the size of the server or network request. The body of the request might need to be encoded before transmission. Security concerns

need to be addressed when sending sensitive information over the server. Methods such as **GET** do not have a request body. The client often determines the structure of the body.

SPOT CHECK 1

Let us see what you can remember from this section.

1. How is the method of an HTTP request specified?
2. What is the purpose of an HTTP request message?
3. How does the DELETE method affect the server?
4. When is a request body used in an HTTP request message?

HTTP RESPONSE

An HTTP response is the response the server sends back to the client, i.e. the web browser, after a request has been executed. The components of an HTTP response include a status code, response header, and body.

1. Status code

The status code consists of a brief description of the status. It indicates the outcome of the request, whether the request was a failure or a success. These consist of 3-digit numeric codes. Several types of status codes are fixed for cases of success or failure.

Some types of HTTP status codes are:

Informational (1xx)

- **100 Continue:** shows that the server has received some part of the request, and the client should proceed with sending the rest.

Success (2xx)

- **200 OK:** shows the request was successful, and a response is being returned.
- **201 Created:** shows the request was a success, and the server is creating a new response as a response.
- **204 No Content:** The request was a success, but no content on the server can be returned as a response.

Redirection (3xx)

- **301 Moved Permanently:** The resource requested on the server has been moved permanently to another location.
- **302 Found:** The resource requested on the server is found in another location.
- **304 Not Modified:** The resource has not been modified since it was last accessed or requested.

Client error (4xx)

- **400 Bad requests:** There is an error on the client's side, so the server cannot comprehend the request.
- **401 Authorization:** The client needs authorisation to access the resources on the server.
- **404 Not Found:** The resource is not found on the server.

Server errors (5xx)

- **500 Internal Server error:** An unexpected error is encountered on the server side.
- **503 Service Unavailable:** The server is unavailable to process the request.



Take note:

Explore other types of [HTTP response status codes](#) and browser compatibility.

2. Response Headers

HTTP response headers are key-value pairs sent by the server as a response to an HTTP request. Some of the HTTP response headers include the following:

- **Content type:** provides information regarding the content type in the response body.
- **Content length:** gives information about the size of the content.
- **Set cookie:** sets a cookie on the browser of the client.
- **Last-Modified:** gives information about the last time the resource was modified.
- **Expires:** gives the time after which the response will expire.
- **Server:** indicates the version of the software.

- **Access-Control-Allow-Origin:** specifies the origins of accessing a resource in case of cross-origin requests.

3. Response Body

The body of the response is an optional part of the response. It can take various forms, such as text, XML, JSON objects, etc. The request body is separated from the headers by a blank line.

SPOT CHECK 2

Let us see what you can remember from this section.

1. What does a status code represent in an HTTP response?
2. How is a response body different from a response header?
3. Explain the difference between a 301 and 302 status code.

STRUCTURE OF HTTP REQUEST AND RESPONSE

The structures of HTTP requests and responses are almost identical. They both consist of the following:

Start line

The start line in an HTTP request is called the "Request Line." This line declares the action, method (like GET, POST, etc.), and the web page on which the action and method will occur.

In an HTTP response, the first line is the "Status Line." This tells us whether the action was successful or not, using a status code and a brief message.

Simply put, the Request Line asks for something, and the Status Line tells us what happened with that request.

Optional HTTP headers

HTTP headers are a list of strings that provide additional information about the request or describe the content included in the request's body. HTTP headers are optional.

An example of a header in an HTTP request is "**User-Agent: Mozilla/5.0**" and an example of a response header is "**Content-Type: text/html**".

Blank line

A blank line signifies the completion of sending all the requested information. It separates the header from the body.

Optional body

Adding information in the body is optional for a request. The body can contain data associated with the request, like the content of an HTML form or a document corresponding to the response. The start line and headers of the request determine the size of the body.



Take note:

Please read the Mozilla Developer Network documentation that breaks down the anatomy of an [HTTP message](#) with some examples.

Please note this is compulsory reading for this task.

CONCLUSION

In conclusion, HTTP requests and responses form a communication foundation between clients (front end) and servers (back end). They facilitate exchanging messages, images, files, etc., giving the user an interactive web experience.

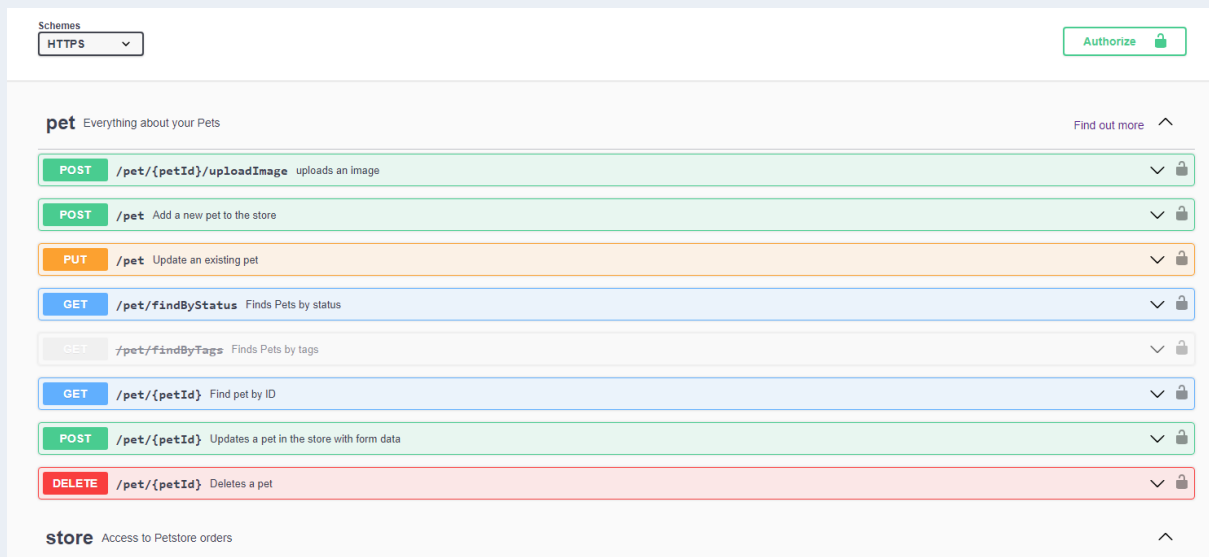
Compulsory Task 1

In this task, you will explore a sample server and its API endpoints. An API endpoint acts as the receiver of HTTP requests and dispatcher of HTTP responses. Then, you will answer questions related to HTTP messages and APIs.

Step 1:

Open your web browser and navigate to the Swagger UI:

- <https://petstore.swagger.io/>

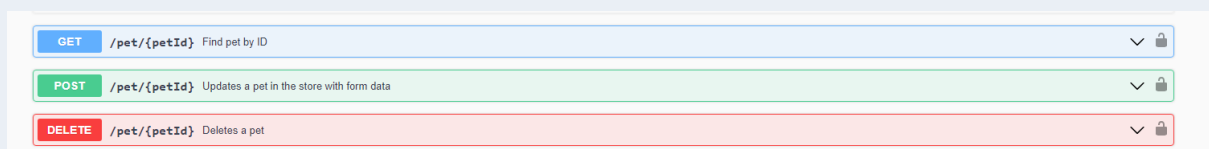


Step 2:

The Swagger UI is already populated with the API definitions of the Petstore API. Explore the API definitions and click on any endpoint (e.g. **GET /pet/{petId}**) to see more details, such as its parameters, responses, and the model of the data it works with.

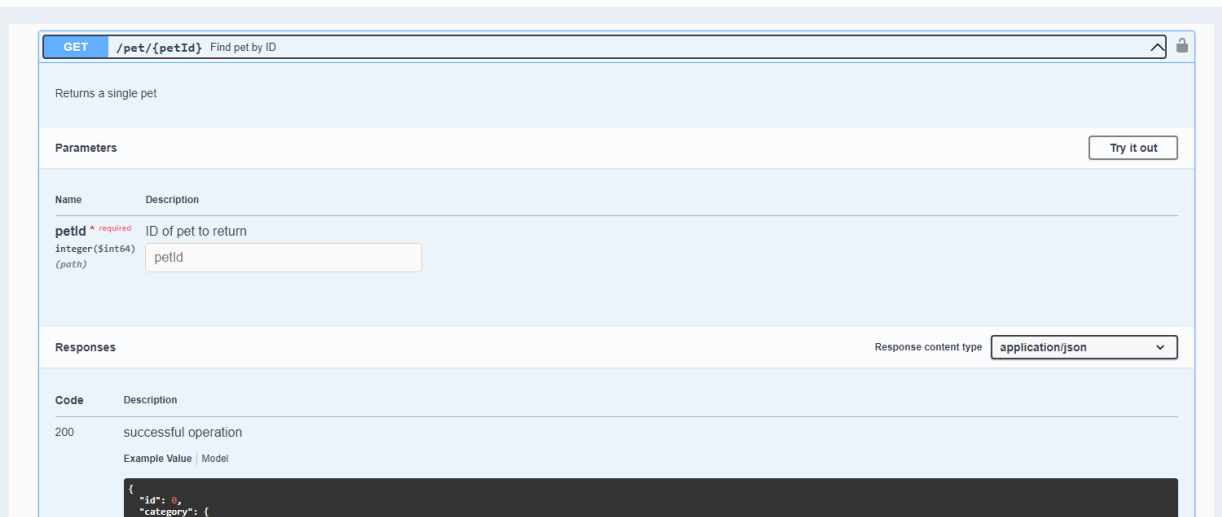
Step 3:

Try executing a request. Click on the "**GET /pet/{petId}**" endpoint to expand it. Then click the "Try it out" button.



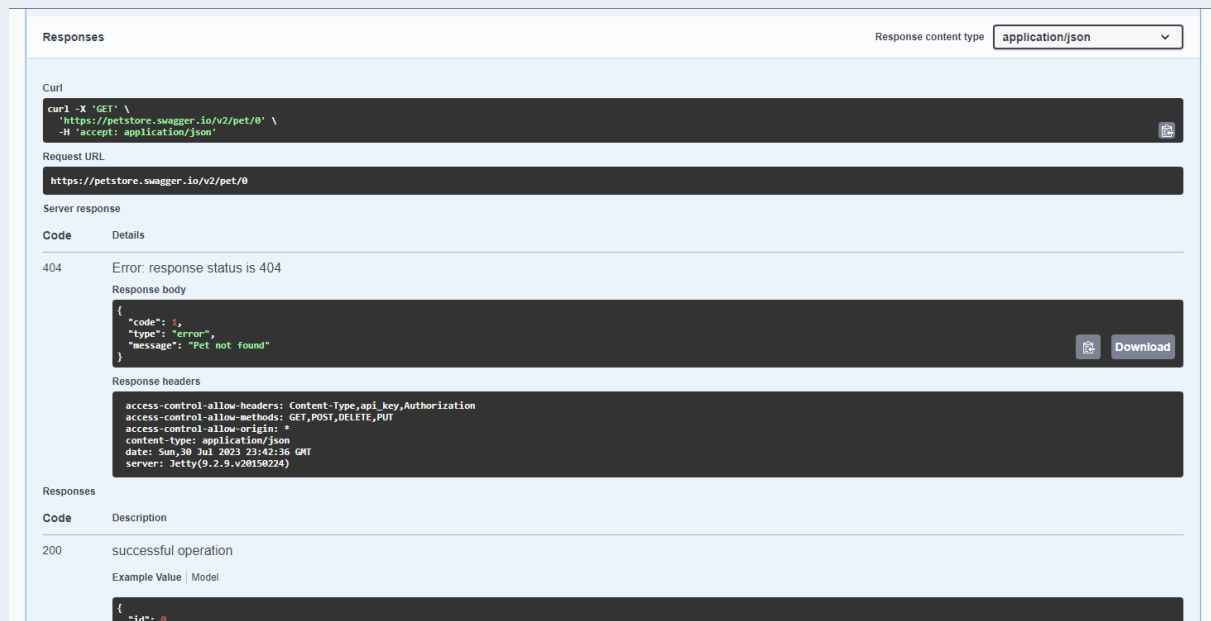
Step 4:

Enter "0" as the petId parameter and click "Execute".



Step 5:

Inspect the responses. Swagger UI will show you the curl command that it used to send the request, the URL of the request, the response body, and the response headers.

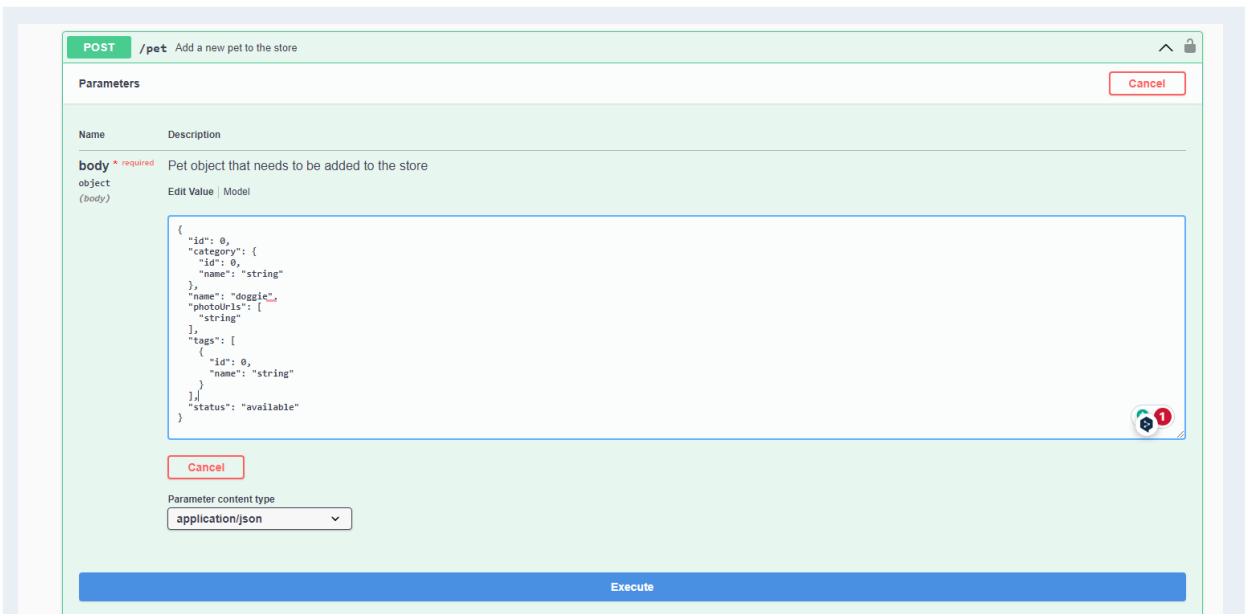


Step 6:

Try executing a different request. Find the "POST /pet" endpoint and click "Try it out".

Step 7:

Swagger UI provides an example body for the request. Modify the "name" and "status" fields and then click "Execute". Again, inspect the responses.



The image shows the Swagger UI interface for a POST request to the `/pet` endpoint. The title bar indicates the method is **POST** and the description is "Add a new pet to the store". There is a "Cancel" button in the top right corner.

Under the "Parameters" section, there is a table with two columns: "Name" and "Description". The first row shows a required parameter named **body** with the description "Pet object that needs to be added to the store". Below this, there are links for "object", "Edit Value", and "Model".

The "body" parameter is expanded, showing a JSON schema in a text editor. The schema defines a pet object with fields: `id` (integer), `category` (object with `id` and `name`), `name` (string), `photoUrls` (array of strings), `tags` (array of objects with `id` and `name`), and `status` (string, with a value of "available").

Below the text editor, there is another "Cancel" button and a "Parameter content type" dropdown menu set to `application/json`. At the bottom of the interface is a large blue "Execute" button.

For this task, submit a PDF document titled **SwaggerUI_Petstore**, answering the following questions:

1. What was the response when you sent a **GET** request to `/pet/1`?
2. What was the response code, and what does it mean?
3. What fields were in the response body, and what do they mean?
4. When you sent a **POST** request to `/pet`, what did you put in the request body?
5. What was the response to the **POST** request?
6. What was the response code of the **POST** request, and what does it mean?
7. Can you explain the purpose of the `/pet/{petId}` and `/pet` endpoints?
8. If the `/pet/{petId}` endpoint returned a 404 status code, what would that mean and what might cause it?
9. Choose one more endpoint from the Petstore API and explain its purpose and how you would use it.
10. **Optional:** What are the advantages of using a tool like Swagger UI when working with APIs?



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.



SPOT CHECK 1 ANSWERS

1. The method is specified in the first line of the HTTP request.
2. It initiates server actions like retrieving, submitting, or deleting data.
3. DELETE requests the removal of a specified resource.
4. A request body is used when additional data is sent to the server, typically with POST, PUT, or PATCH methods.

SPOT CHECK 2 ANSWERS

1. The status code in an HTTP response indicates the result of the request. It informs the client about the success, failure, or need for further action in relation to the request.
2. The response header contains metadata about the response, like content type, length, and server information. The response body contains the actual data (like HTML, JSON, etc.) that the server sends back in response to the request.
3. Both are HTTP status codes for redirection. 301 indicates a permanent redirect, meaning the resource has moved permanently to a new location. 302 indicates a temporary redirect, meaning the resource is temporarily available at a different location.