9) HTTP & Web Services Lesson

HTTP Requests: Methods, Headers, and Bodies

9 min to complete · By Ryan Desmond, Jared Larsen

Contents

The previous lesson provided a general explanation of HTTP (HyperText Transfer Protocol), and its crucial role in enabling web communication. You learned that HTTP is the backbone of data transfer on the internet, facilitating interactions across a multitude of varying systems. Now it's time so see how information is actually transmitted over this protocol.

HTTP requests are the primary means through which data is exchanged over the internet. In this lesson, you will learn how these requests work, the various types (methods) of requests (such as GET, POST, PUT, DELETE), and the structure of an HTTP request.

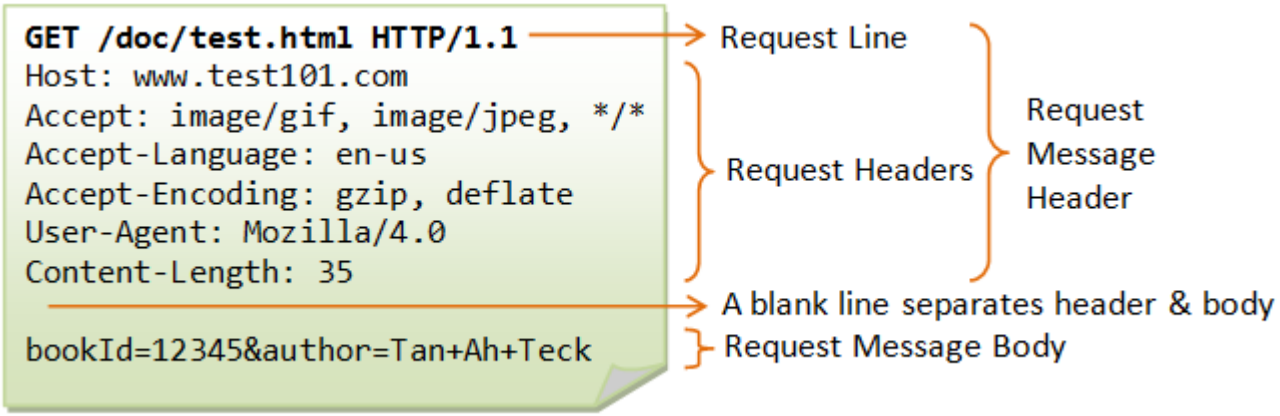
HTTP Request Structure

HTTP requests are very flexible in how they allow information exchange. There are three locations where you can store and send information within a request, each with its own

abilities and restrictions. Here is a list of possible locations:

- HTTP Headers
- HTTP Methods
- HTTP Bodies

The image below shows you how these three components create a full HTTP message:



Let's start with HTTP headers.

HTTP Headers

HTTP headers are not mandatory, but they are typically included in every HTTP request. They are formatted to have a key (you can think of this as the name of a variable) and a value. The headers essentially make up a Hash Map that provides important information about the request or the client sending the request.

Normally, headers are used to negotiate the HTTP business between client and server. Information like which language (e.g., English) is being used, the date, the type of browser, and the type of content sent back can all be found in headers. However, it is possible to add your own custom headers which can contain any data you want.

HTTP Methods

Each HTTP request is sent across the web with a method. This 'method' consists of a single verb in all caps. It instructs the server what action you want it to take. The most commonly used methods include GET, PUT, POST, DELETE and PATCH, and here is a quick summary of each:

- **GET:** Retrieve data from a specified resource.

- **POST:** Submit data to be processed. Usually to create a new resource.
- **PUT:** Update a specified resource.
- **DELETE:** Delete a specified resource.
- **PATCH:** Partially update resources.
- **HEAD:** Returns only the HTTP headers for a request. Similar to GET, but the HTTP body is not returned.
- **OPTIONS:** Returns the supported HTTP methods available for the entered URL.

HTTP Body

The HTTP body is slightly more complicated than the aforementioned information storage locations. This is true for two reasons:

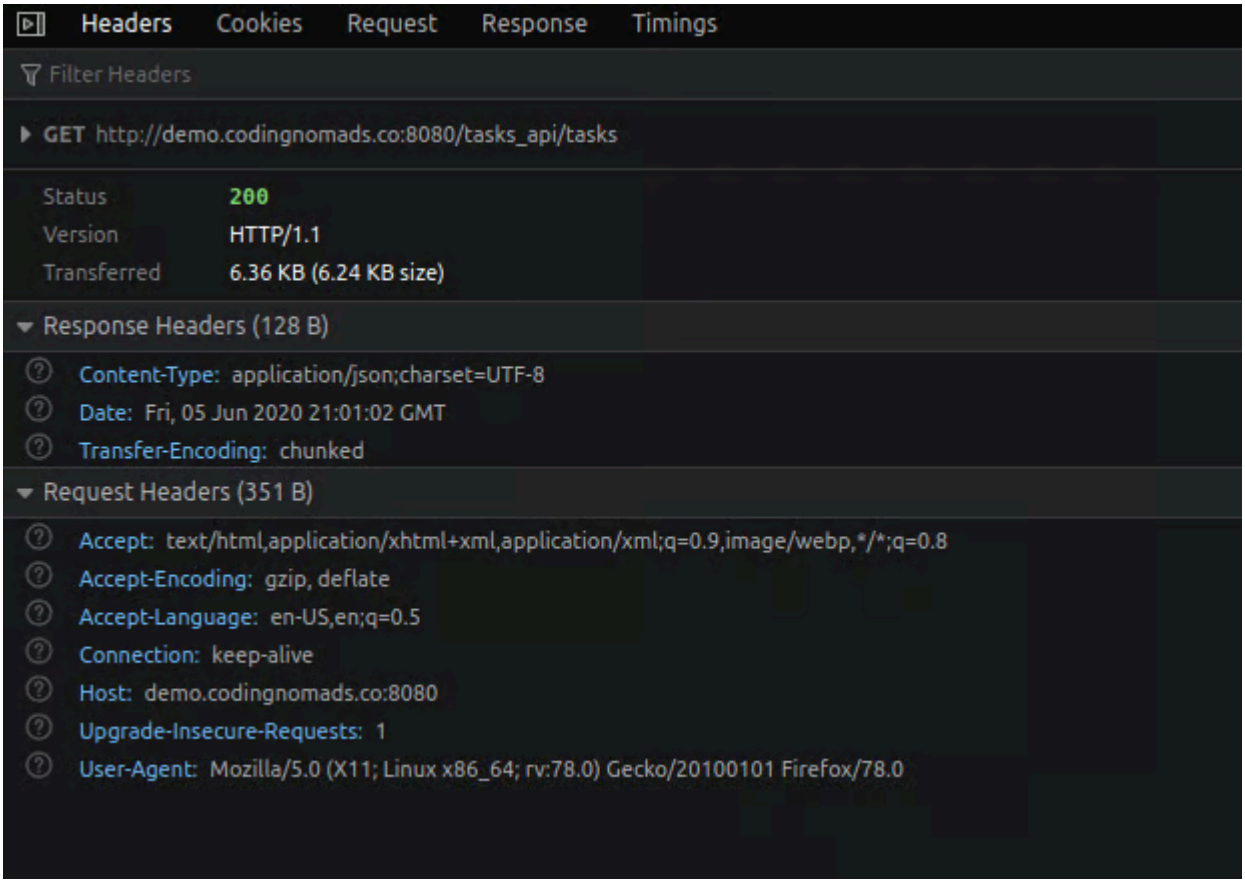
1. When sending requests (not including responses), systems only accept HTTP bodies for certain HTTP methods. For example, generally if you send a GET request to the server with some data stored in the body, this will get completely overlooked. However, send a POST request, and the server will automatically look for information in the body. The response that is sent back from the server may or may not contain a body, but you're in control so you can manage this however you want.

2. The HTTP body can transport any data format you like. You can send an image, plain text, JSON, binary and more. The sky is the limit. While this freedom is useful, it also has a downside. Consider the following example: You have set up an automated program that makes a request for the daily weather from your local weather station. It manipulates the data and stores it in a database. One day, the weather station updates its service, and the HTTP body which used to contain plain text now contains HTML. You did not design your system to interpret and pull apart HTML information and so it crashes. Not great.

Luckily, there is a way to combat these types of failures. The client and the server can tell each other what type of content they are able to understand. This is done using an HTTP header named "Accept". If in the above example, your system had specified that it was only able to understand plain text, the weather station (if it wanted to facilitate backward compatibility) would have sent the data in the old format. Here is some more information on the Accept header.

Take a look at the image below. This is an HTTP response body in a JSON format. It may look daunting but do not worry. We'll go over this more later, but some extra exposure is

always good.



This response was the data returned from `http://demo.codingnomads.co:8080/tasks_api/tasks`

Learn by Doing

Location: `http://demo.codingnomads.co:8080/tasks_api/tasks`

- Visit this link in your browser.
- Open dev tools (right-click, then "Inspect").
- Navigate to the Network tab, then refresh the page.
- Take some time to inspect the data returned in the Preview and Header tabs (inside the Network tab).

Summary: HTTP Requests

This section covered three important components of HTTP requests. The use of each is critical when it comes to utilizing the HTTP protocol to its fullest extent. Let's recap:

- HTTP headers store information using a key/value system.
- HTTP methods instruct the server what the intent of the request is.
- HTTP bodies allow you to transmit data in almost any format.

Some common HTTP methods include:

- `GET` : Retrieve data from a specified resource.
- `POST` : Submit data to be processed. Usually to create a new resource.
- `PUT` : Update a specified resource.
- `DELETE` : Delete a specified resource.
- `PATCH` : Partially update resources.

Previous Next → URL Structure in Web Development