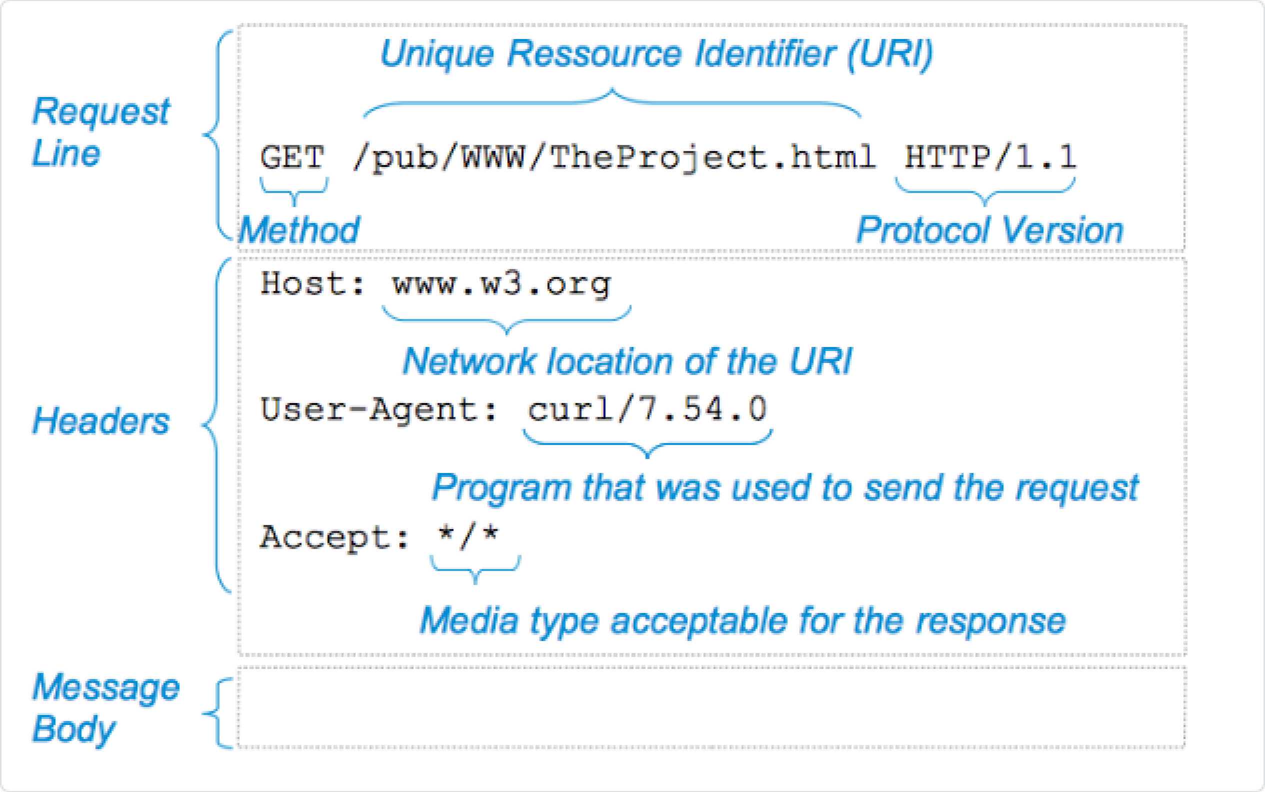
HTTP Request

An HTTP request consists of a request line, headers, a blank line, and an optional body. Each part has a specific role in telling the server what the client wants and providing any necessary data.


Request Line
Unique Ressource Identifier (URI)
GET /pub/WWW/TheProject.html HTTP/1.1
Method
Protocol Version
Headers
Host: www.w3.org
Network location of the URI
User-Agent: curl/7.54.0
Program that was used to send the request
Accept: */*
Media type acceptable for the response
Message Body

Anatomy of an HTTP request[fig:Anatomy-of-HTTP-request]

1. Request Line

- **Format:** `METHOD PATH HTTP/VERSION`
  - **METHOD:** The HTTP method (e.g., GET, POST, PUT, DELETE).
  - **PATH:** The URL path of the resource (e.g., /api/users/1).
  - **HTTP/VERSION:** The HTTP protocol version (e.g., HTTP/1.1).
- **Example:**
  `GET /api/products HTTP/1.1`

2. Headers

- **Purpose:** Provide metadata about the request.
- **Some headers examples:**
  - `Host`: The domain name of the server (e.g., `Host: example.com`)

  - `Content-Type`: The format of the request body (e.g., `application/json`)
  - `Authorization`: Credentials for authentication (e.g., Bearer token)
  - `User-Agent`: Info about the client (browser, app, etc.)
- **Example:**

```
Content-Type: application/json
Authorization: Bearer <token>
```

3. Blank Line

- Separates headers from the body.
- Always present, even if the body is empty.

4. Body (Optional)

- **Purpose:** Contains data sent to the server (mainly in POST, PUT, PATCH requests).
- **Example (JSON):**

```
{
  "name": "Jane Doe",
  "email": "jane@example.com"
}
```

Full Example of an HTTP Request

```
POST /api/users HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer abc123

{
  "name": "Jane Doe",
  "email": "jane@example.com"
}
```

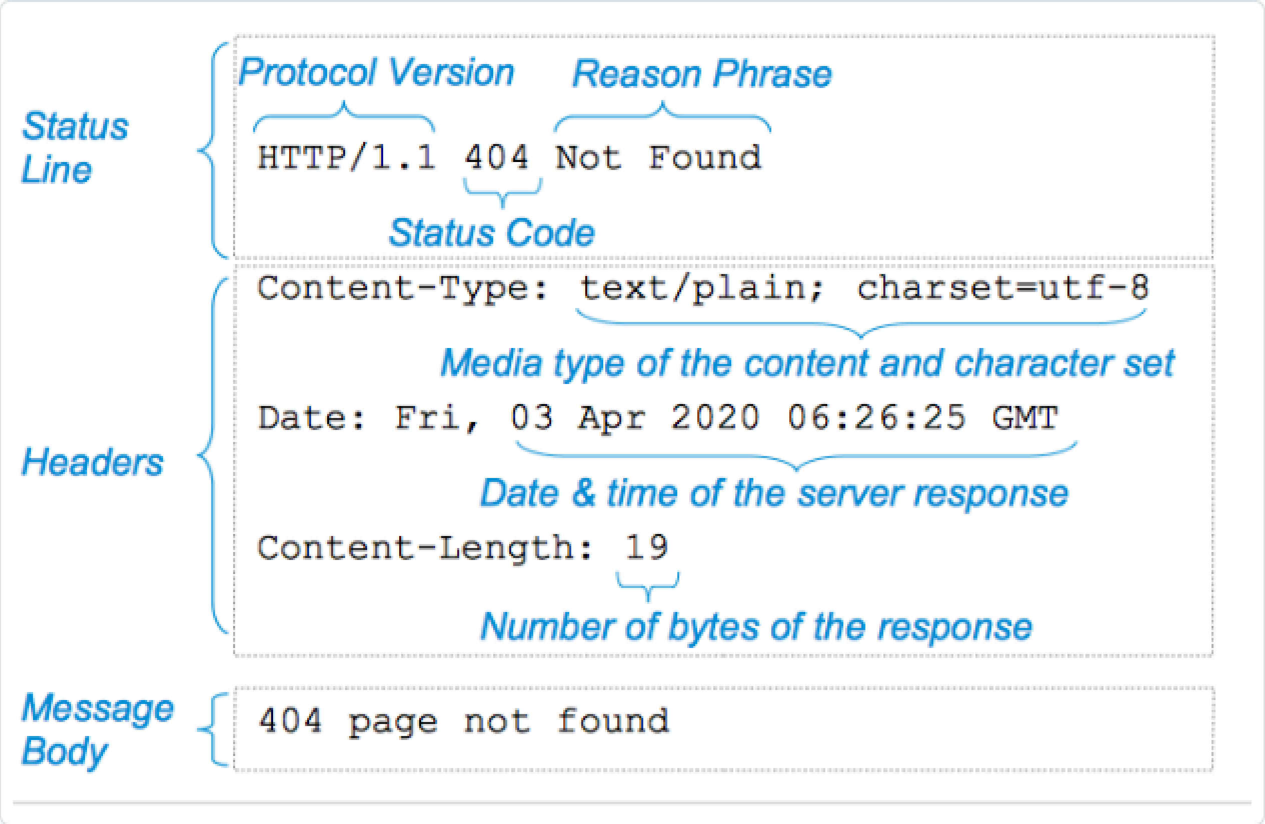
Summary Table

Part	Description
Request Line	Method, path, and HTTP version

Part	Description
Headers	Metadata (content type, authentication, etc.)
Blank Line	Marks end of headers, before body
Body	Data sent to server (optional, often for POST/PUT/PATCH)

HTTP Response

An HTTP response consists of a status line, headers, a blank line, and an optional body. Each part serves to inform the client about the result of the request and provide any requested data.



```
HTTP/1.1 404 Not Found
Content-Type: text/plain; charset=utf-8
Date: Fri, 03 Apr 2020 06:26:25 GMT
Content-Length: 19

404 page not found
```

Status Line
Protocol Version
Reason Phrase
Status Code
Headers
Media type of the content and character set
Date & time of the server response
Number of bytes of the response
Message Body

Anatomy of an HTTP response[fig:Anatomy-of-a-http-response]

1. Status Line

- **Format:** `HTTP/VERSION STATUS_CODE REASON_PHRASE`
  - **HTTP/VERSION:** The HTTP protocol version (e.g., HTTP/1.1).
  - **STATUS_CODE:** A numeric code indicating the result (e.g., 200, 404).
  - **REASON_PHRASE:** Textual description of the status (e.g., OK, Not Found).
- **Example:**
  `HTTP/1.1 200 OK`

2. Headers

- **Purpose:** Provide metadata about the response.
- **Examples:**
  - `Content-Type`: The format of the response body (e.g., `application/json`)
  - `Content-Length`: Size of the response body in bytes
  - `Set-Cookie`: Sets cookies for the client
  - `Cache-Control`: Instructs caching behavior
- **Example:**

```
    Content-Type: application/json
    Content-Length: 123
```

3. Blank Line

- Separates headers from the body.
- Always present, even if the body is empty.

4. Body (Optional)

- **Purpose:** Contains the data sent back to the client (HTML, JSON, images, etc.).
- **Example (JSON):**

```
    {
      "id": 1,
      "name": "Jane Doe",
      "email": "jane@example.com"
    }
```

Full Example of an HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
  "id": 1,
  "name": "Jane Doe",
  "email": "jane@example.com"
}
```

Summary Table

Part	Description
Status Line	Protocol, status code, and reason phrase
Headers	Metadata (content type, length, cookies, etc.)
Blank Line	Marks end of headers, before body
Body	Data sent to client (optional, often for GET/POST/PUT)

** https://www.practical-go-lessons.com/chap-26-basic-http-server