

HTTP Basics

HTTP Request & HTTP Response



SoftUni Team

Technical Trainers

 Software
University

The logo consists of a yellow hexagon with a blue 'S' shape inside, followed by the text "Software University".

SoftUni



Software University

<https://softuni.bg>

Table of Contents

1. The HTTP Protocol – Basic Concepts
2. HTTP Developer Tools
3. HTML Forms
4. HTTP Request
5. HTTP Response
6. URLs and URL Structure

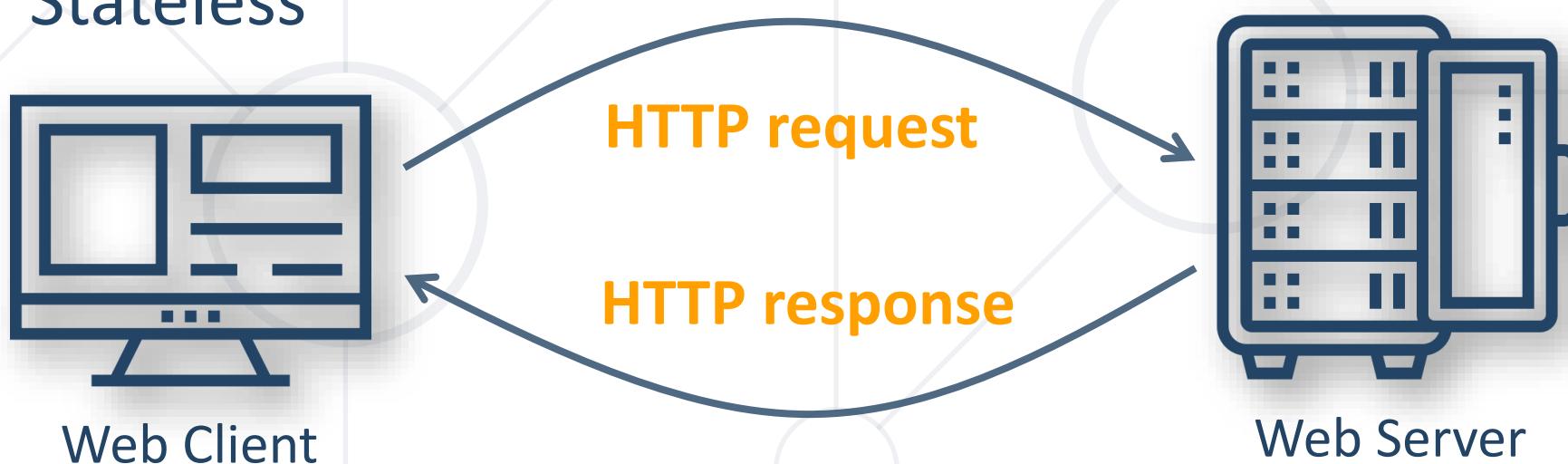




`http://`

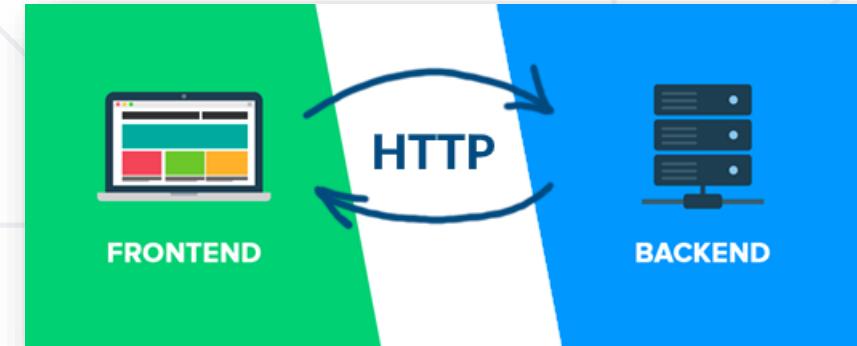
HTTP Protocol – Basics

- **HTTP (HyperText Transfer Protocol)**
 - Text-based client-server protocol for the Internet
 - For transferring Web resources (HTML files, images, styles, etc.)
 - Request-response based, relies on URLs (like <https://softuni.org>)
 - Stateless



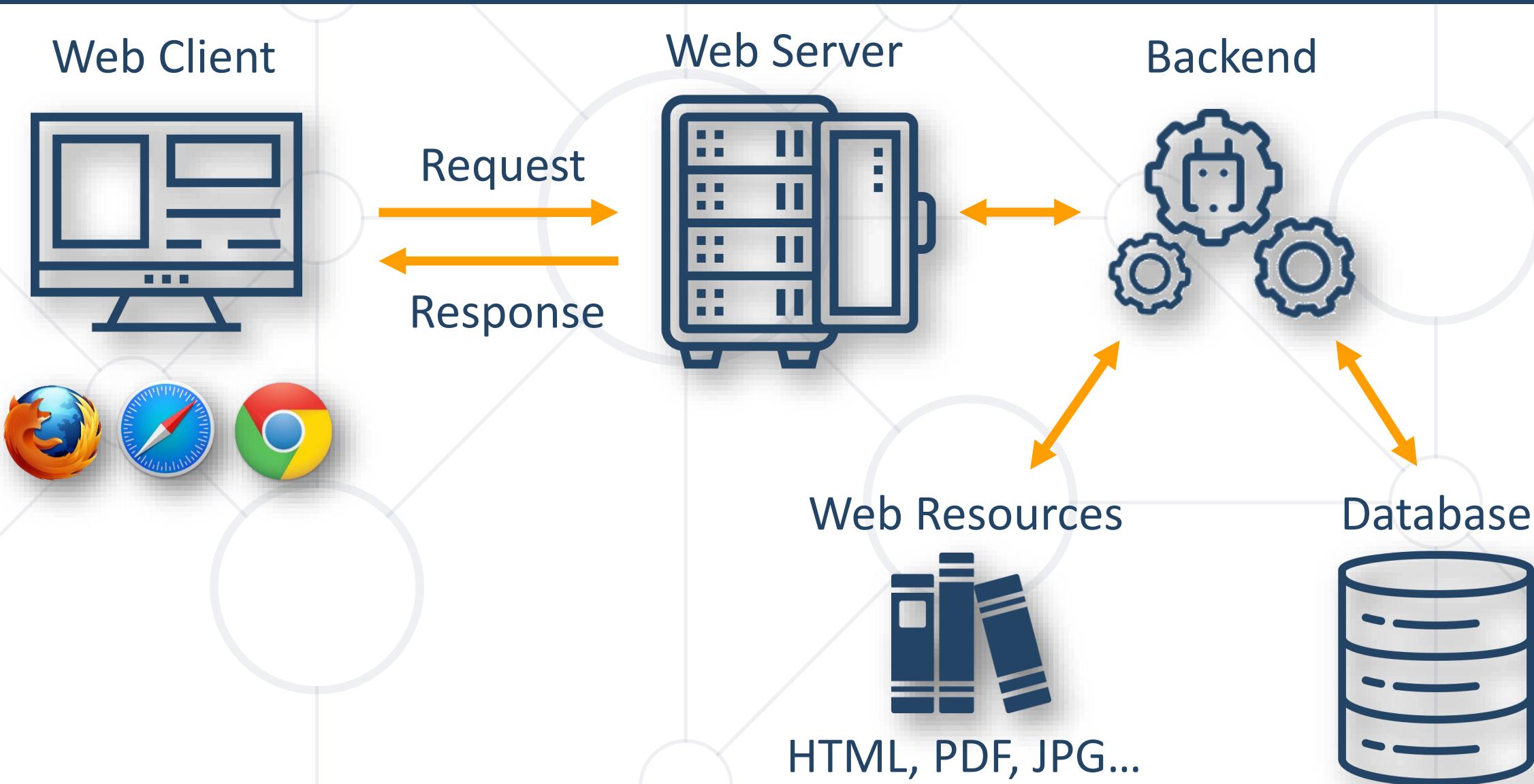
Front-End and Back-End

- **Front-end** and **back-end** separates the modern apps into **client-side** (UI) and **server-side** (data) components
- **Front-end** == client-side components (presentation layer), e.g., React app
 - Implement the **user interface** (UI)
- **Back-end** == server-side components (business logic APIs), e.g., ASP.NET Core
 - Provide **data storage and processing**

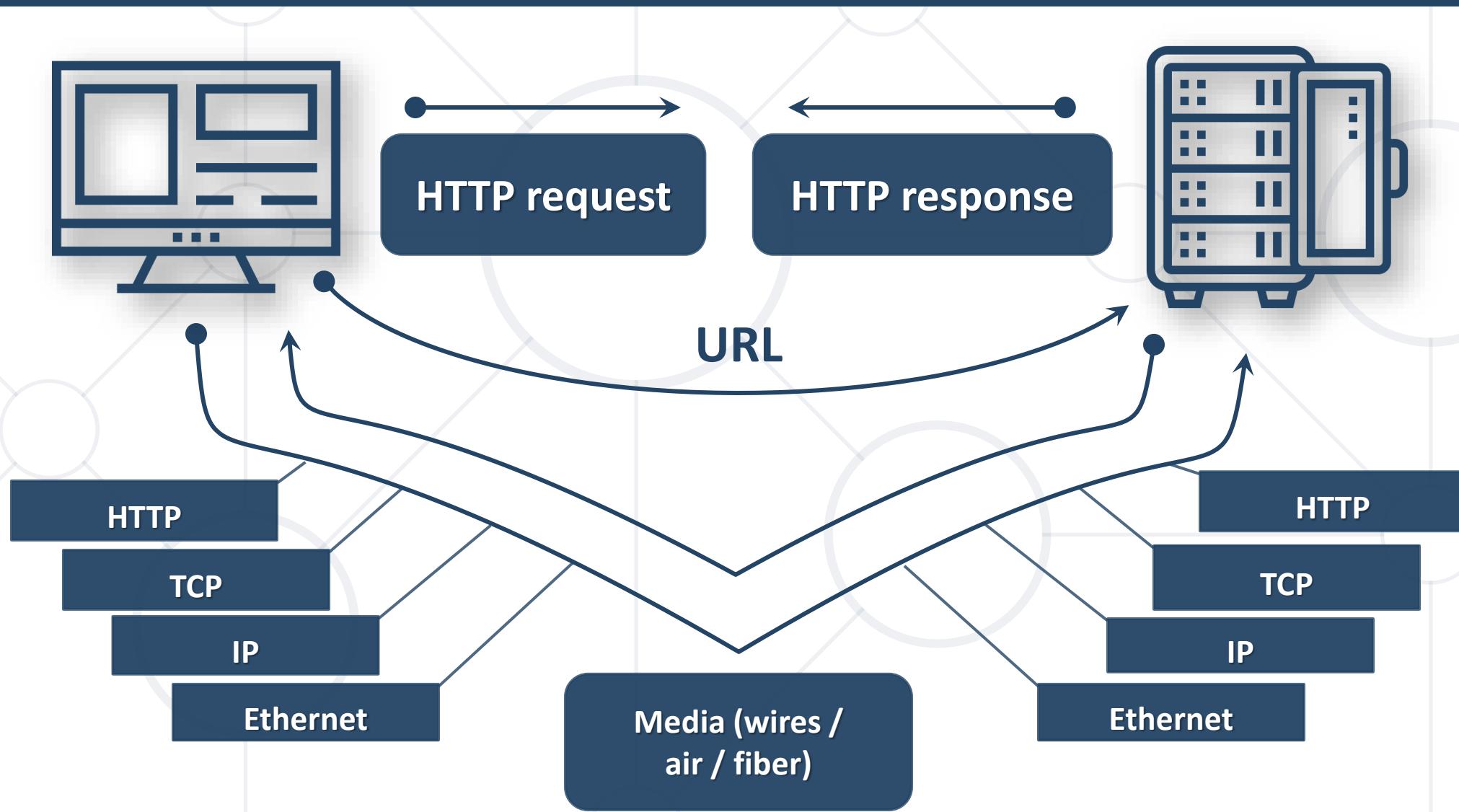


- **HTTP** connects front-end with back-end

The Client-Server Model in Web Apps



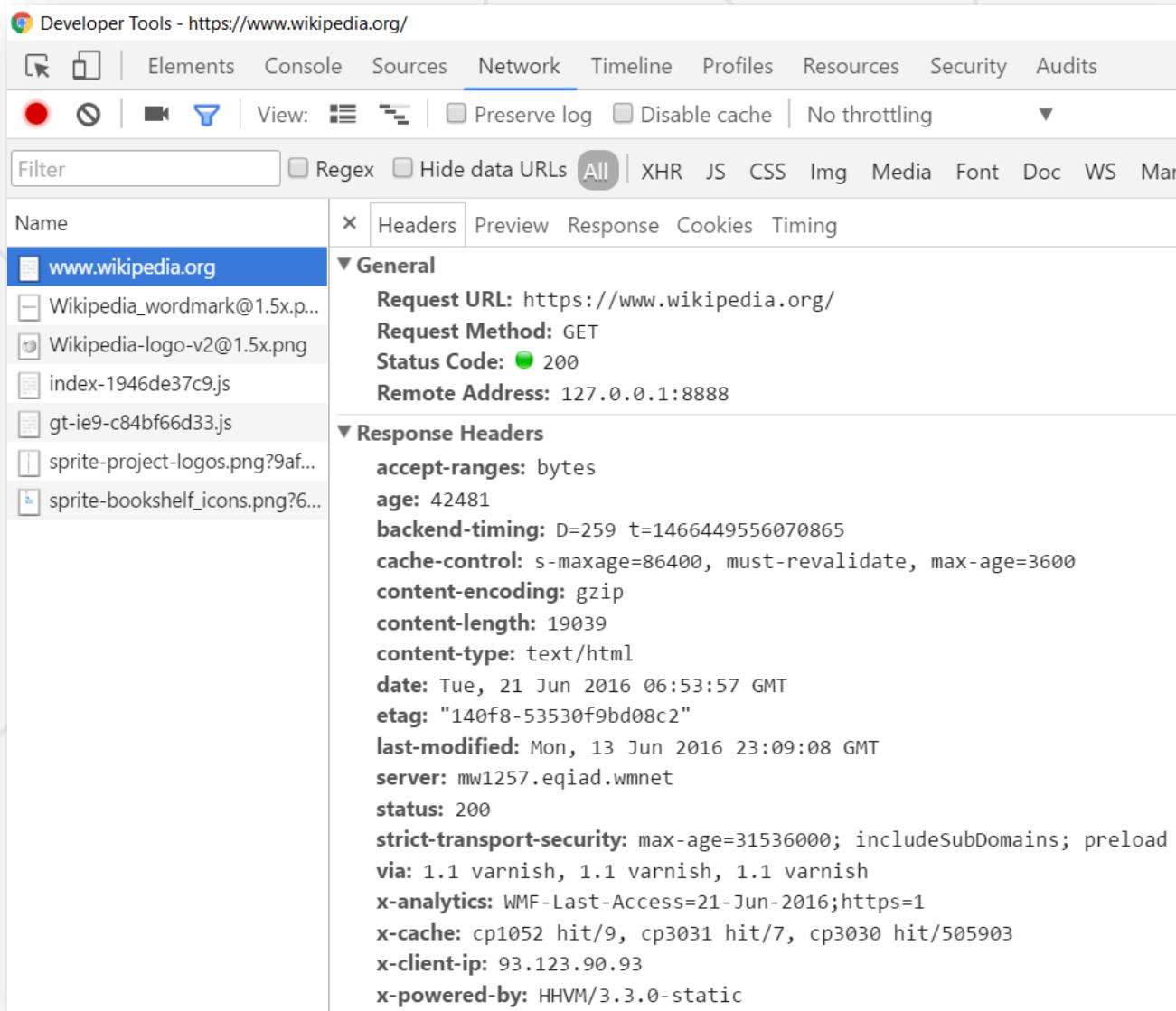
Network Layers and HTTP





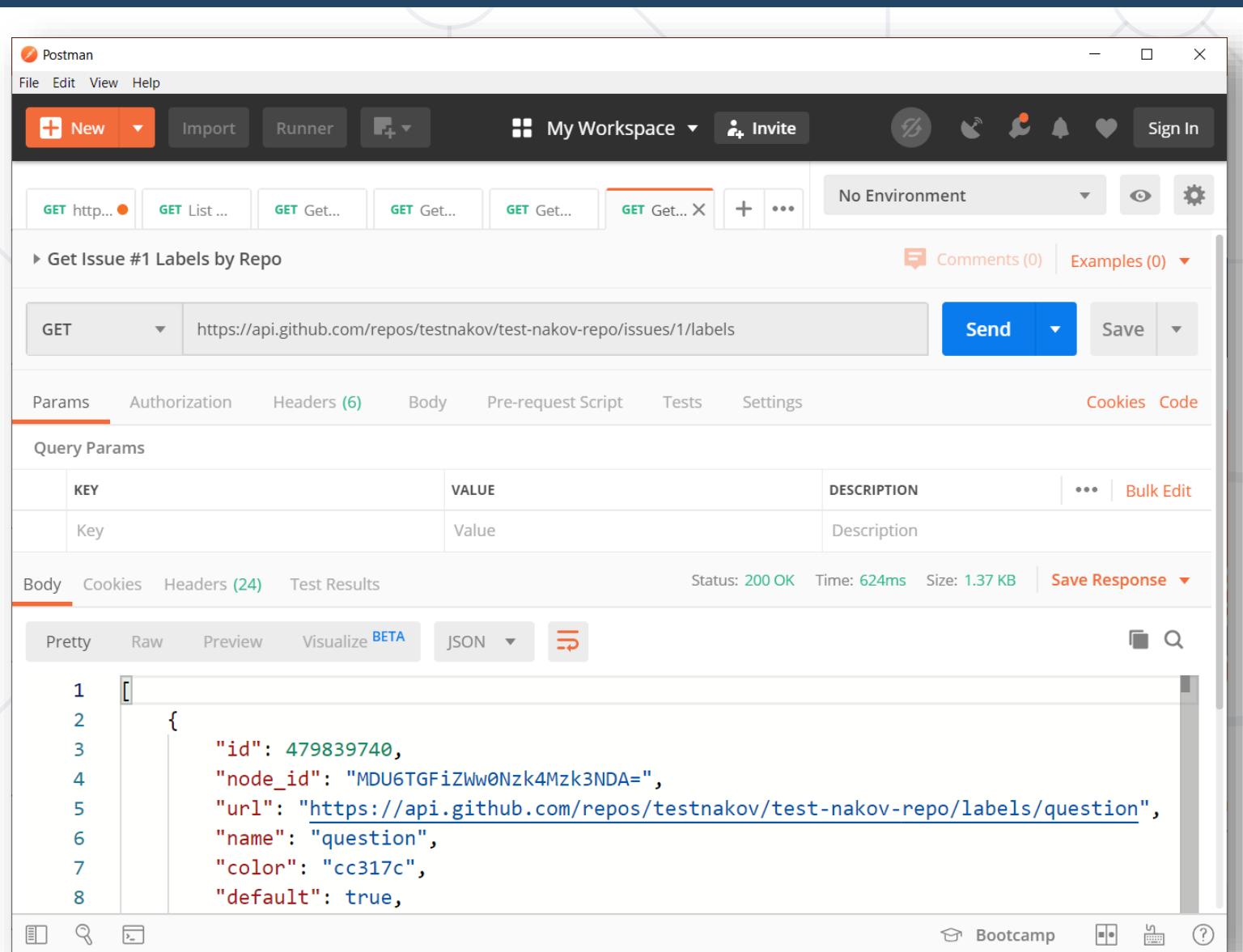
HTTP Dev Tools

HTTP Developer Tools: Network Inspector



- Chrome Developer Tools
 - Press **[F12]** in Chrome
 - Open the **[Network]** tab
 - Inspect the HTTP traffic

HTTP Developer Tools: HTTP Client Tools



The screenshot shows the Postman application interface. At the top, there's a toolbar with 'File', 'Edit', 'View', 'Help' menus, and buttons for 'New', 'Import', 'Runner', and 'Invite'. Below the toolbar, a navigation bar includes 'My Workspace', 'Invite', and icons for profile, notifications, and sign in. A status bar at the bottom says 'Bootcamp'.

In the main workspace, a 'Get Issue #1 Labels by Repo' request is listed. The request type is 'GET' and the URL is 'https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/labels'. Below the URL, there are buttons for 'Send' and 'Save'. The 'Params' tab is selected, showing a table with one row: 'Key' (Value) and 'Value' (Description). Other tabs include 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', 'Settings', 'Cookies', and 'Code'. The 'Body' tab is active, displaying a JSON response:

```
1 [ { 2   "id": 479839740, 3   "node_id": "MDU6TGFiZWw0Nzk4Mzk3NDA=", 4   "url": "https://api.github.com/repos/testnakov/test-nakov-repo/labels/question", 5   "name": "question", 6   "color": "cc317c", 7   "default": true, 8 } ]
```



Postman

- HTTP client tool for developers
- Compose and send HTTP requests
- Insomnia Core
- Hoppscotch



HTML Forms

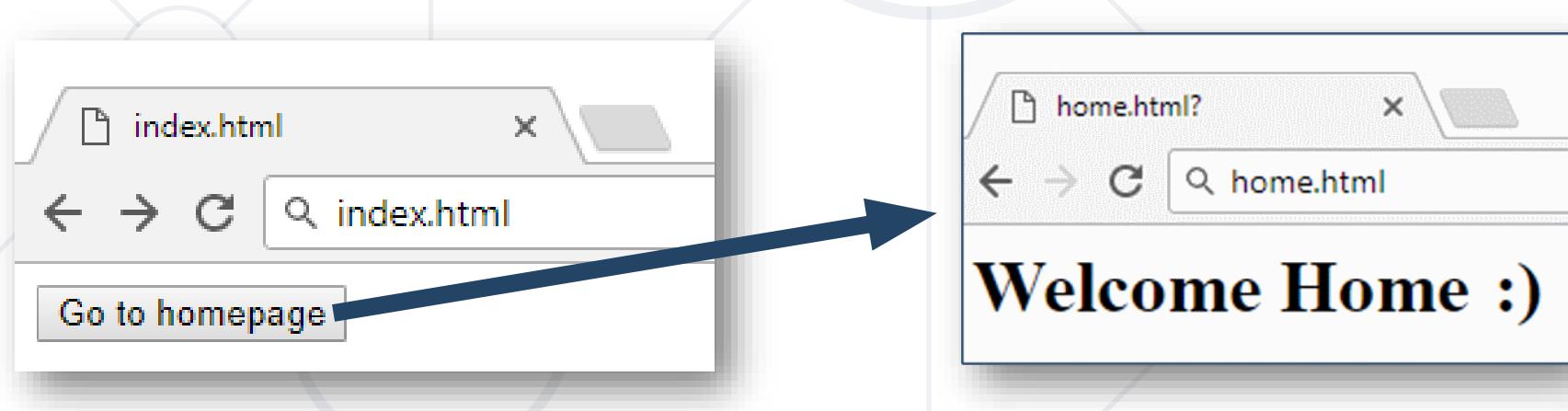
Form Submission: GET and POST

HTML Forms: Action

- The "action" attribute defines where to submit the form data

```
<form action="home.html">  
  <input type="submit" value="Go to homepage"/>  
</form>
```

Relative or full URL

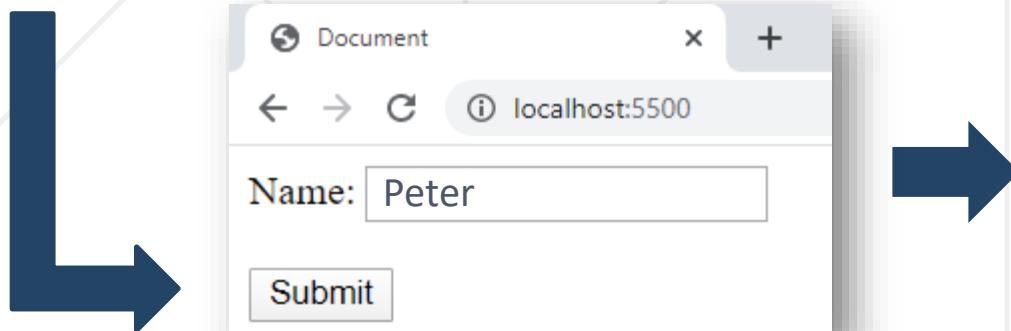


Example: <https://repl.it/@nakov/http-form-example>

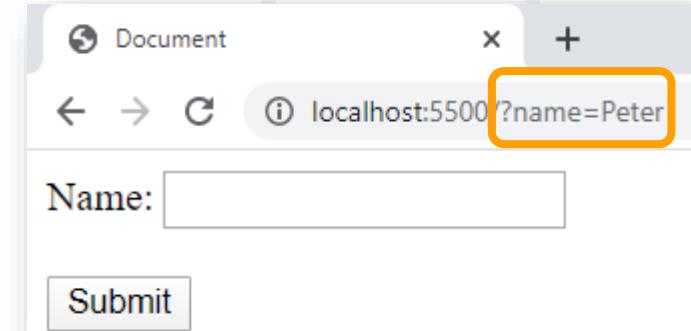
HTML Forms: Method GET

- Forms can specify the **HTTP method** for sending the form data

```
<form method="get">  
  Name: <input type="text" name="name">  
  <br /><br />  
  <input type="submit" value="Submit">  
</form>
```



A screenshot of a browser window titled "Document". The address bar shows "localhost:5500". The page contains a form with a text input field labeled "Name:" containing the value "Peter", and a "Submit" button below it.



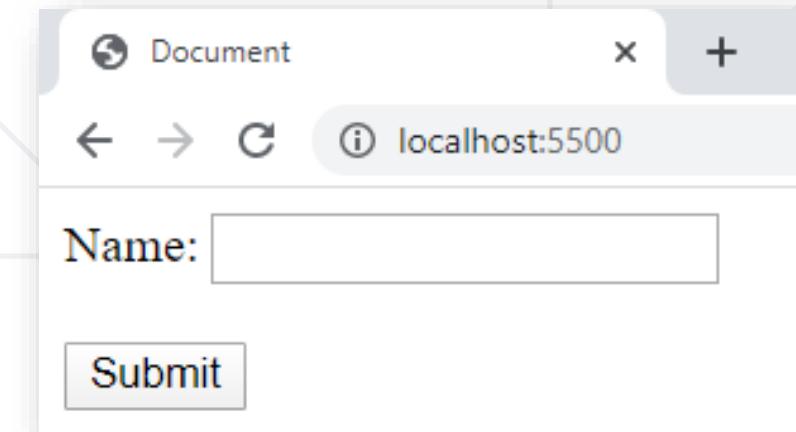
A screenshot of a browser window titled "Document". The address bar shows "localhost:5500?name=Peter". The page contains a form with a text input field labeled "Name:" and a "Submit" button below it.

The form data is
in the URL

Example: <https://repl.it/@nakov/http-get-example>

HTML Forms: Method POST

```
<form method="post">  
  Name: <input type="text" name="name">  
  <br /><br />  
  <input type="submit" value="Submit">  
</form>
```



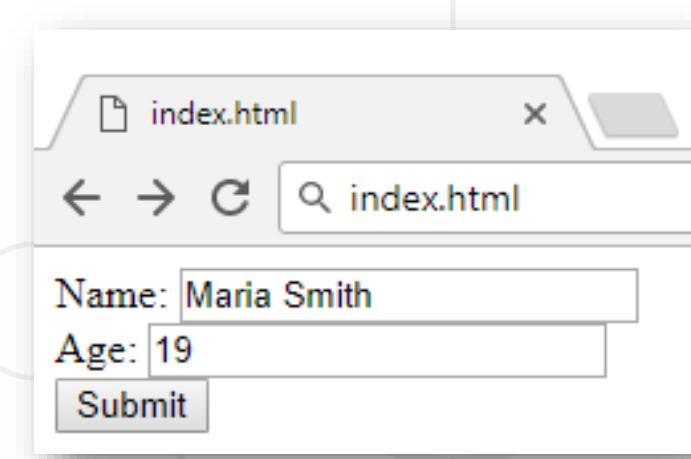
POST /index.html HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 10
name=Peter

The HTTP request body holds
the submitted form data

Example: <https://repl.it/@nakov/http-post-example>

URL Encoded Form Data – Example

```
<form method="post">  
  Name: <input type="text" name="name"/> <br/>  
  Age: <input type="text" name="age"/> <br/>  
  <input type="submit" />  
</form>
```



POST /index.html HTTP/1.1

Host: localhost

Content-Type: application/x-www-form-urlencoded

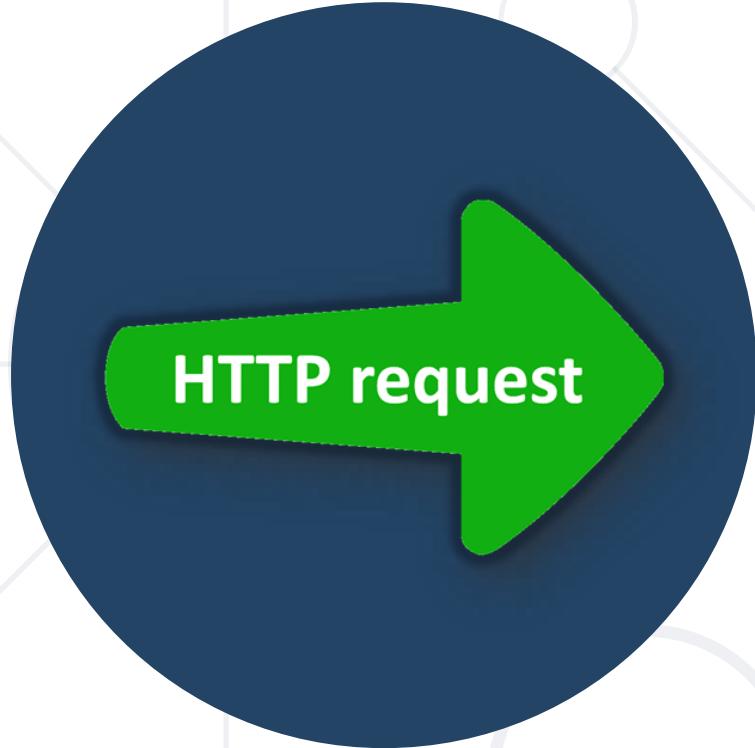
Content-Length: 23

name=Maria+Smith&age=19

URL-encoded form data

File upload fields
are not supported
(unless multipart
encoding is set)

Example: <https://repl.it/@nakov/http-post-example-name-age>



HTTP Request

Request Method, Headers, Body

HTTP Request Methods

- HTTP defines **methods** to indicate the desired action to be performed on the identified resource

| Method | Description | Method |
|--------|--|---------|
| GET |  Retrieve a resource | CONNECT |
| POST |  Create / store a resource | OPTIONS |
| PUT |  Update (replace) a resource | TRACE |
| DELETE |  Delete (remove) a resource | |
| PATCH |  Update resource partially (modify) | |
| HEAD |  Retrieve the resource's headers | |

CRUD == the four main functions of persistent storage

HTTP GET Request – Example

GET /users/SoftUni-Tech-Module/repos HTTP/1.1

Host: api.github.com

Accept: */*

Accept-Language: en

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)

AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/54.0.2840.71 Safari/537.36

Connection: keep-alive

Cache-Control: no-cache

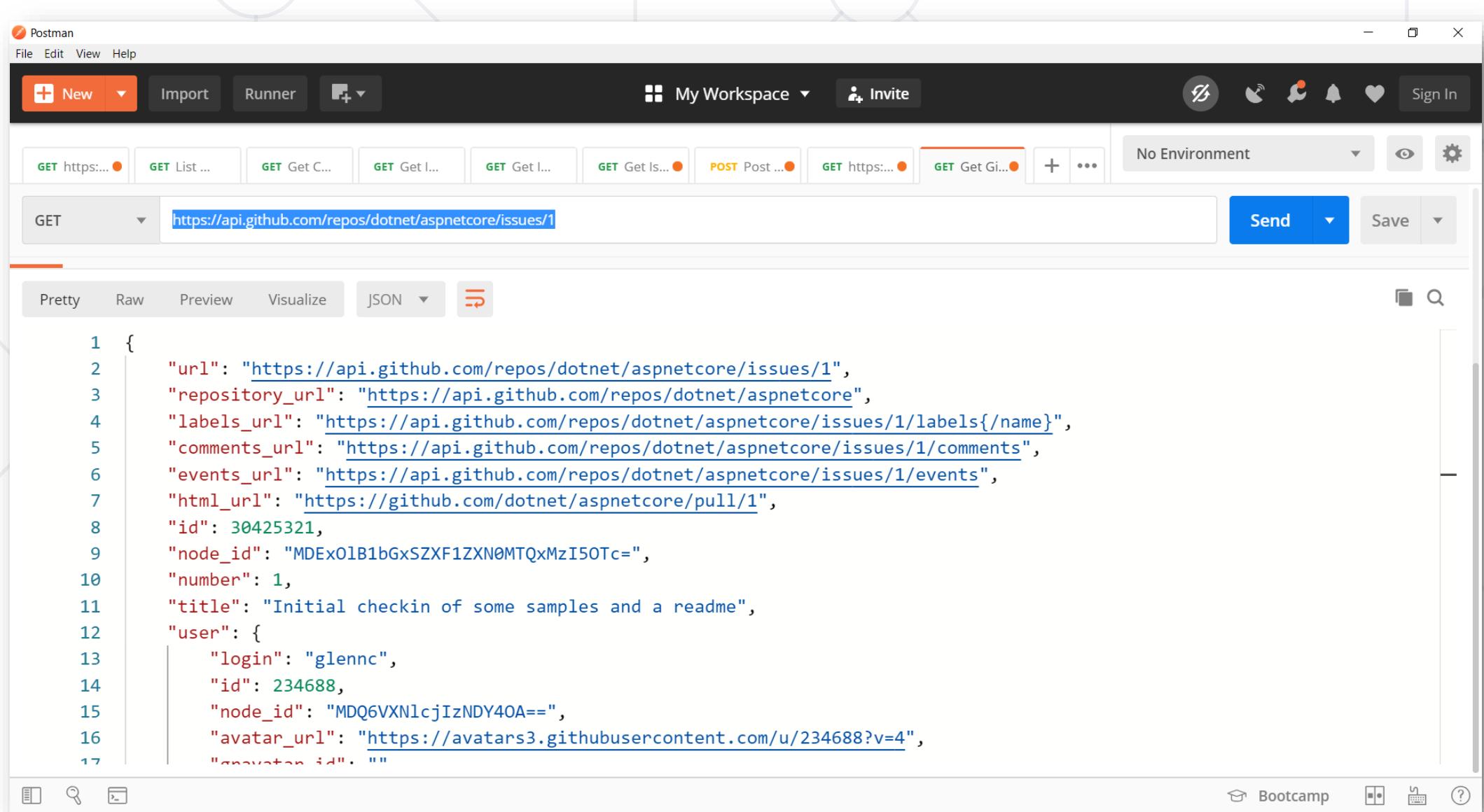
<CRLF> The request body is empty

Realtive URI,
not full URL

HTTP request line

HTTP headers

HTTP GET – Example with Postman



The screenshot shows the Postman application interface. The top navigation bar includes 'File', 'Edit', 'View', 'Help', 'Import', 'Runner', and workspace-related buttons ('My Workspace', 'Invite'). The toolbar below has several items, including a 'GET https:...' item which is currently selected. The main request panel shows a 'GET' method being used to access the URL <https://api.github.com/repos/dotnet/aspnetcore/issues/1>. The response pane displays the JSON data for the first issue of the 'dotnet/aspnetcore' repository. The JSON output is formatted with line numbers and collapsible sections.

```
1 {
2   "url": "https://api.github.com/repos/dotnet/aspnetcore/issues/1",
3   "repository_url": "https://api.github.com/repos/dotnet/aspnetcore",
4   "labels_url": "https://api.github.com/repos/dotnet/aspnetcore/issues/1/labels{/name}",
5   "comments_url": "https://api.github.com/repos/dotnet/aspnetcore/issues/1/comments",
6   "events_url": "https://api.github.com/repos/dotnet/aspnetcore/issues/1/events",
7   "html_url": "https://github.com/dotnet/aspnetcore/pull/1",
8   "id": 30425321,
9   "node_id": "MDExOlB1bGxSZXF1ZXN0MTQzMzI5OTc=",
10  "number": 1,
11  "title": "Initial checkin of some samples and a readme",
12  "user": {
13    "login": "glennc",
14    "id": 234688,
15    "node_id": "MDQ6VXNlcjIzNDY40A==",
16    "avatar_url": "https://avatars3.githubusercontent.com/u/234688?v=4",
17    "gravatar_id": ""
}
```

HTTP POST Request – Example

POST /post HTTP/1.1

HTTP request line

URL: <https://postman-echo.com/post>

Host: postman-echo.com

Accept: */*

Accept-Encoding: gzip, deflate

Content-Type: application/json

Connection: keep-alive

Content-Length: 95

<CRLF>

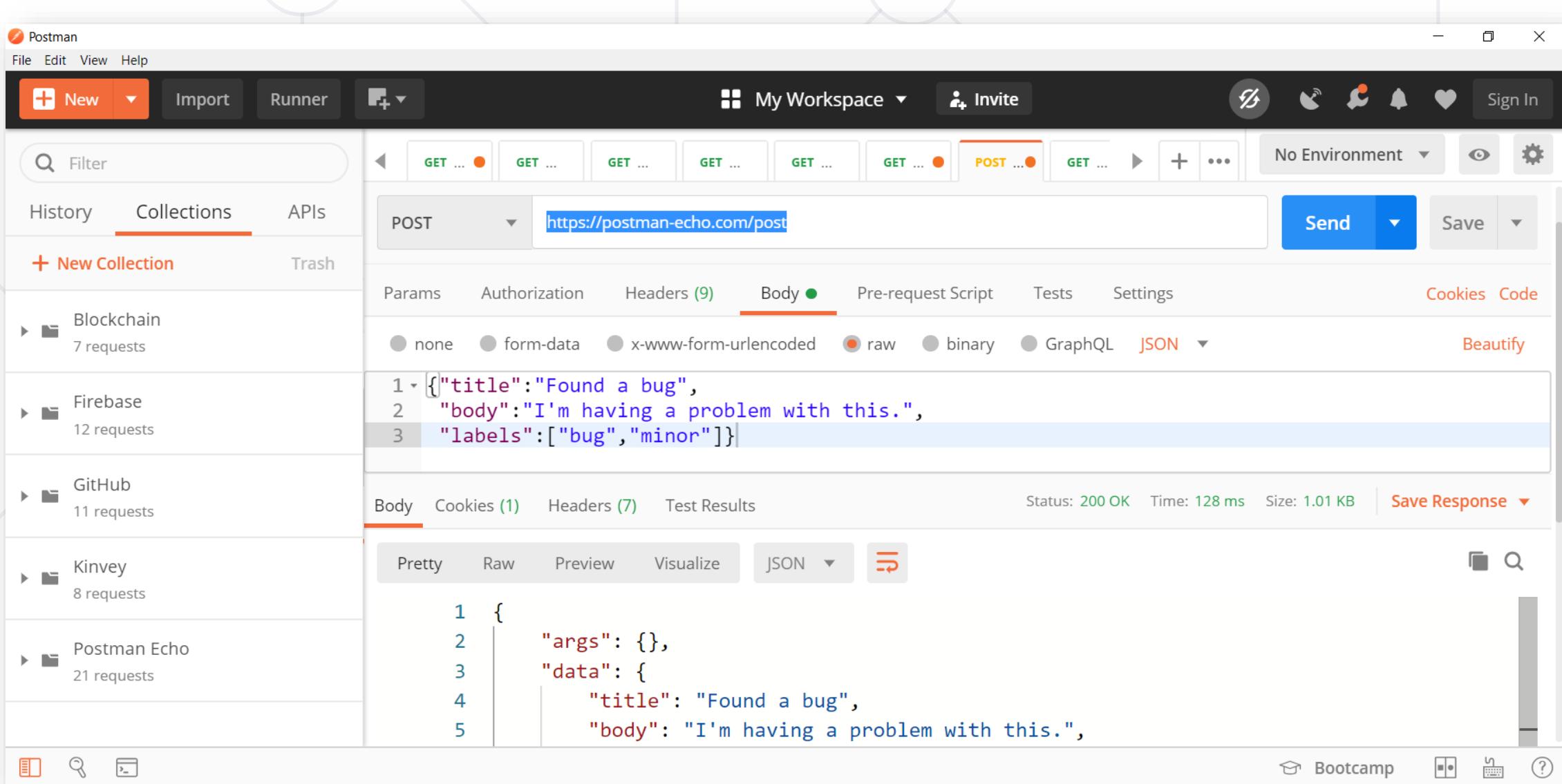
{"title": "Found a bug",
 "body": "I'm having a problem with this.",
 "labels": ["bug", "minor"]}

<CRLF>

HTTP headers

The request body holds
the submitted data

HTTP POST – Example with Postman



The screenshot shows the Postman application interface. In the top navigation bar, the 'POST' method and the URL <https://postman-echo.com/post> are selected. The 'Body' tab is active, showing a JSON payload:

```
1 {"title": "Found a bug",
2  "body": "I'm having a problem with this.",
3  "labels": ["bug", "minor"]}
```

The response status is 200 OK, with a time of 128 ms and a size of 1.01 KB. The JSON response body is displayed below:

```
1 {
2   "args": {},
3   "data": {
4     "title": "Found a bug",
5     "body": "I'm having a problem with this.",
```



The diagram features a central dark blue circle containing a green arrow pointing left, with the text "HTTP response" written in white. This central element is surrounded by a network of light gray circles connected by thin gray lines, forming a mesh-like pattern that suggests a complex system or data flow.

HTTP response

HTTP Response

Response Status, Headers, Body

HTTP Response – Example

HTTP/1.1 200 OK

HTTP response status line

Date: Fri, 11 Nov 2016 16:09:18 GMT+2

Server: Apache/2.2.14 (Linux)

Accept-Ranges: bytes

HTTP response headers

Content-Length: 84

Content-Type: text/html

<CRLF>

<html>

HTTP response body

 <head><title>Test</title></head>

 <body>Test HTML page.</body>

</html>

HTTP Response Status Codes

| Status Code | Action | Description |
|-------------|--------------|--|
| 200 | OK | Successfully retrieved resource |
| 201 | Created | A new resource was created |
| 204 | No Content | Request has nothing to return |
| 301 / 302 | Moved | Moved to another location (redirect) |
| 400 | Bad Request | Invalid request / syntax error |
| 401 / 403 | Unauthorized | Authentication failed / Access denied |
| 404 | Not Found | Invalid resource was requested |
| 409 | Conflict | Conflict was detected, e.g. duplicated email |
| 500 / 503 | Server Error | Internal server error / Service unavailable |

Content-Type and Disposition

- The **Content-Type / Content-Disposition** headers specify how to process the HTTP request / response body

Content-Type: application/json

JSON-encoded data

Content-Type: text/html; charset=utf-8

UTF-8 encoded
HTML page

Content-Type: application/pdf

Download a PDF file

Content-Disposition: attachment;
filename="Financial-Report-2020.pdf"

- Standard media types: <https://iana.org/assignments/media-types>

HTTP Conversation: Example

GET /trainings/courses HTTP/1.1

Host: softuni.org

User-Agent: Mozilla/5.0

<CRLF>

HTTP Request

HTTP/1.1 200 OK

Date: Tue, 16 May 2020 15:13:41 GMT

Server: Microsoft-HTTPAPI/2.0

Last-Modified: Tue, 16 Jan 2018 15:13:42 GMT

Content-Length: 18586

<CRLF>

<html><title>Get a Tech Degree from...</title>

HTTP Response



URL

Protocol, Host, Path, Query String

Uniform Resource Locator (URL)



- **Network protocol** (`http`, `ftp`, `https`...) – HTTP in most cases
- **Host or IP address** (`softuni.org`, `gmail.com`, `127.0.0.1`, `web`)
- **Port** (the default port is `80`) – integer in the range [0...65535]
- **Path** (`/forum`, `/path/index.php`)
- **Query string** (`?id=27&lang=en`)
- **Fragment** (`#slides`) – navigate to some section in the page

Query String

- Query string contains data that is **not part** of the path structure

```
http://example.com/path/to/page?name=tom&color=purple
```
- Commonly used in searches and dynamic pages
- It is the part of the URL after the question mark (?) symbol
- Parameters have **name=value** format
- Multiple parameters are separated by the **&** delimiter

URL Encoding

- URLs are encoded according to [RFC 1738](#)
 - Normal URL characters – have no special meaning
[0-9a-zA-Z]
 - Reserved URL characters – have a **special meaning**
! * ' () ; : @ & = + \$ / , ? # []
 - Reserved characters are **escaped by percent encoding**
%[character hex code]
 - **Space** is encoded as "+" or "%20"

URL Encoding – Examples

- All other characters are escaped by **% hex code**, e.g.,

| Char | URL Encoding |
|-------|--------------|
| space | %20 |
| " | %22 |
| # | %23 |
| \$ | %24 |

| Char | URL Encoding |
|------|--------------|
| % | %25 |
| & | %26 |
| Щ | %D1%89 |
| 爰 | %E7%88%B1 |

- Example

Наков-爰-SoftUni

Each char is converted to its UTF-8 bytes, represented as hex digits

%D0%9D%D0%B0%D0%BA%D0%BE%D0%B2-%E7%88%B1-SoftUni

Valid and Invalid URLs – Examples

■ Some valid URLs

http://www.google.bg/search?sourceid=navclient&ie=UTF-8&rlz=1T4GGGL_enBG369BG369&q=http+get+vs+post

<http://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%BA%D0%BA%D0%BD%D0%B4%D0%B5%D0%BC%D0%B8%D1%8F>

■ Some invalid URLs

Should be: C%23+.NET+4.0

[http://google.com/search?q=C# .NET 4.0](http://google.com/search?q=C%23 .NET 4.0)

<http://google.com/search?q=код>

Should be: %D0%BA%D0%BE%D0%B4

- **HyperText Transfer Protocol**
 - Text-based client-server protocol for the Internet
 - Works with message pairs
 - Request: method + headers + body
 - Response: status + headers + body
 - The URL parts: **protocol, host, port, path, query string and fragment**

