

# HTTP Basics

## HTTP Request & HTTP Response



SoftUni Team  
Technical Trainers



**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



[sli.do](https://sli.do)

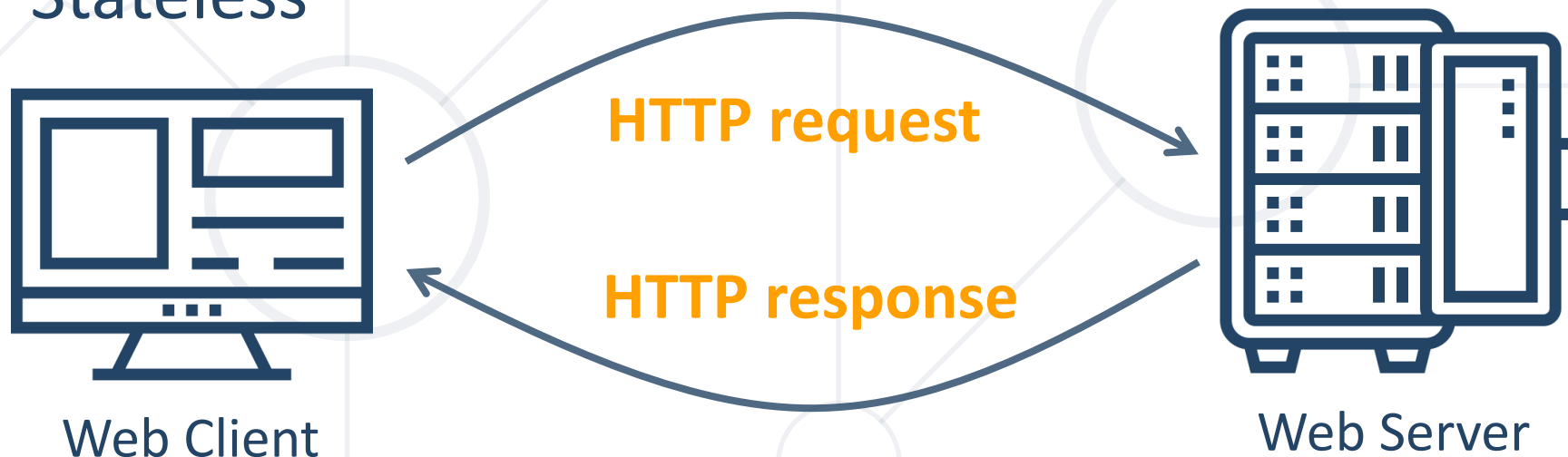
**#fund-common**

A background network diagram consisting of a grid of light gray lines intersecting at various points. At these intersections, there are several circles of different sizes, some solid light gray and some hollow, representing nodes in a network.

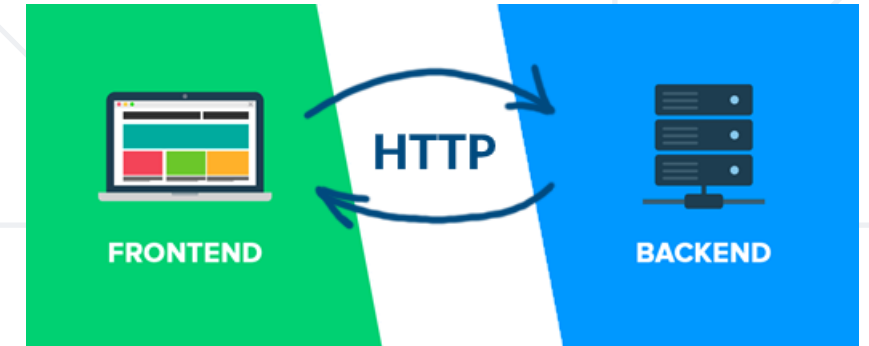
**http://**

# **HTTP Protocol – Basics**

- **HTTP** (**H**yperText **T**ransfer **P**rotocol)
  - 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

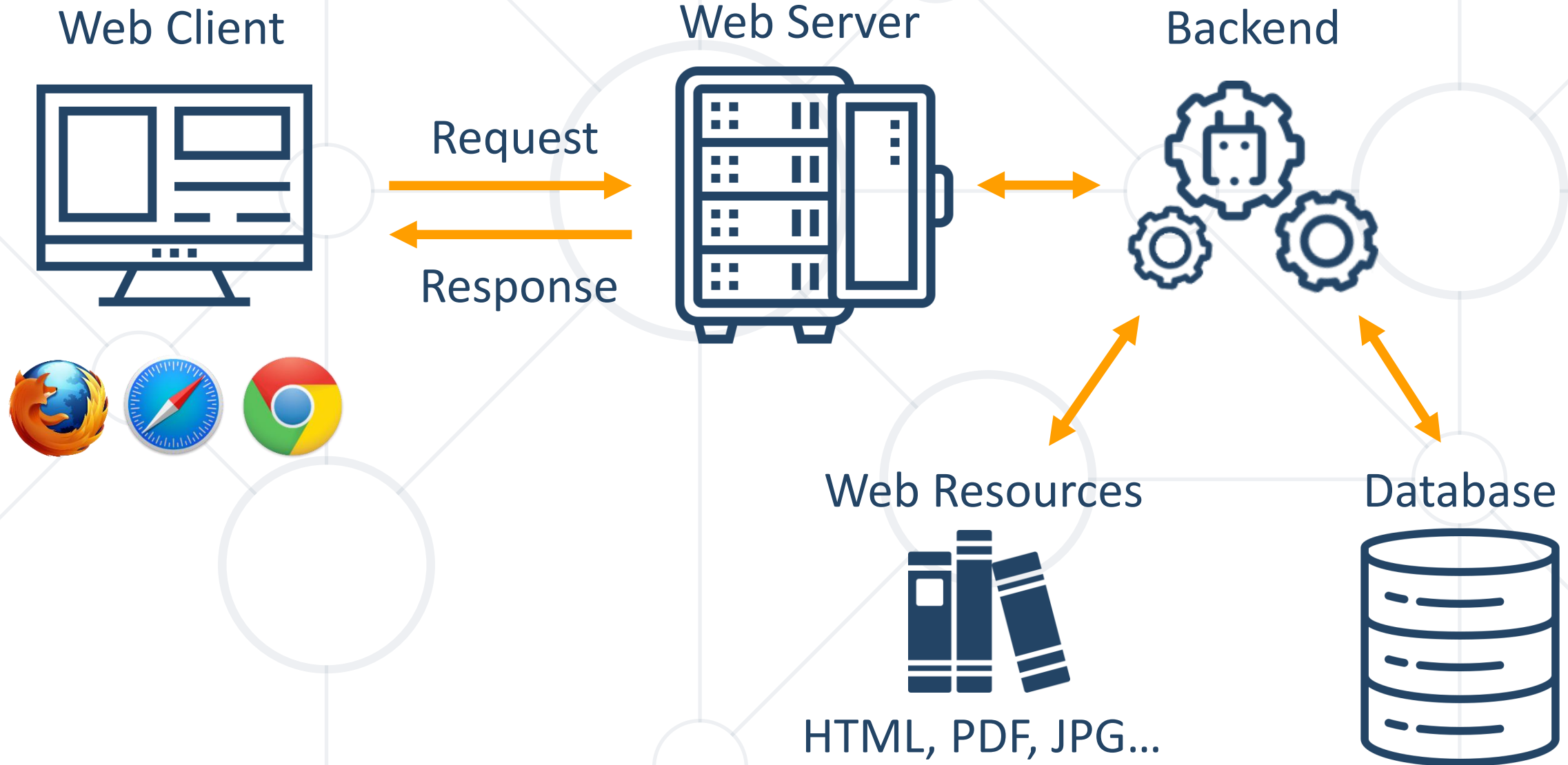


- **Front-end** and **back-end** separates the modern apps into **client-side** (UI) and **server-side** (data) components
- **Front-end** == client-side components (presentation layer), e.g. React app
  - Implement the **user interface** (UI)
- **Back-end** == server-side components (business logic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

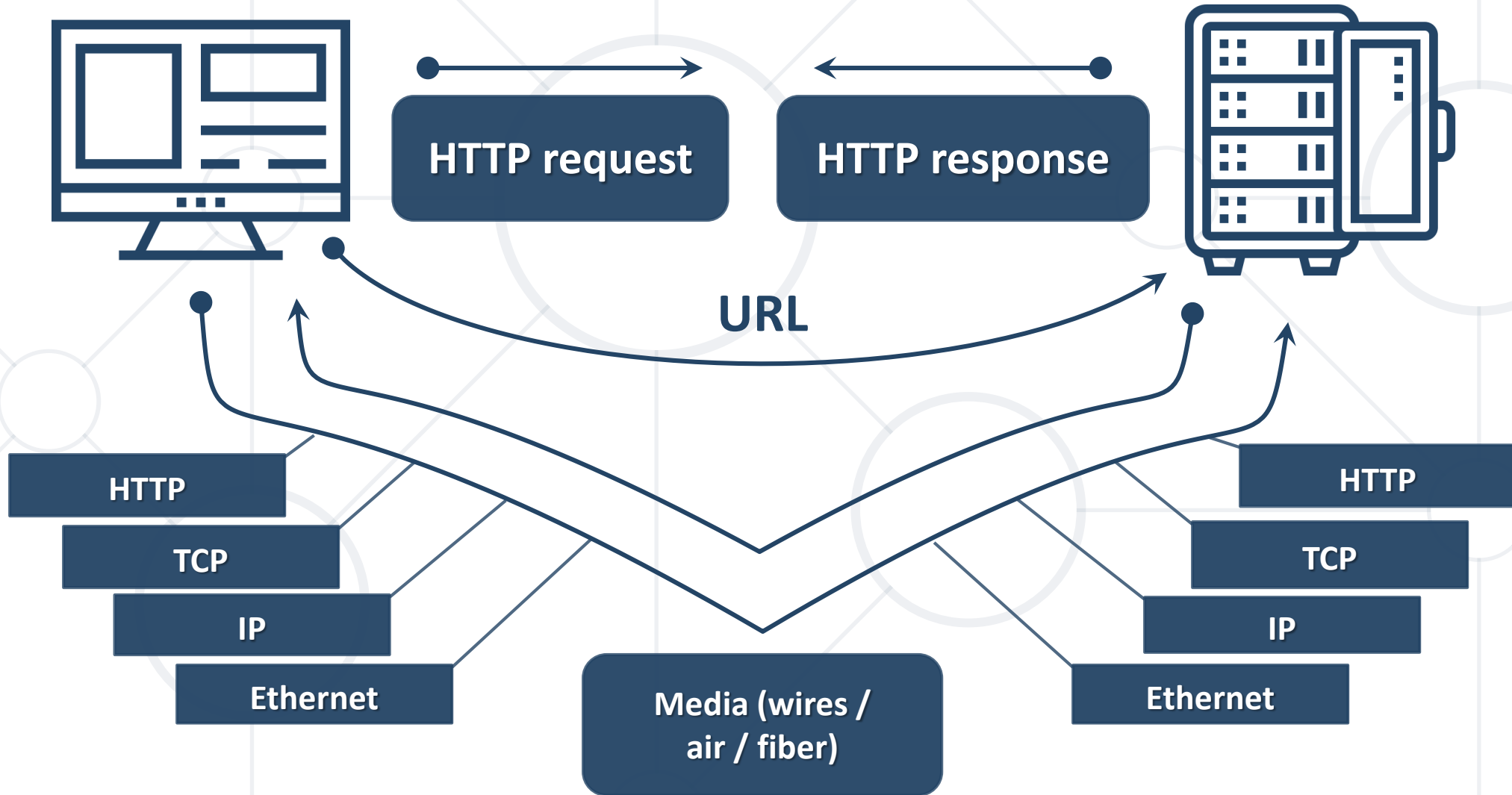


- **HTTP** connects front-end with back-end

# The Client-Server Model in Web Apps



# Network Layers and HTTP

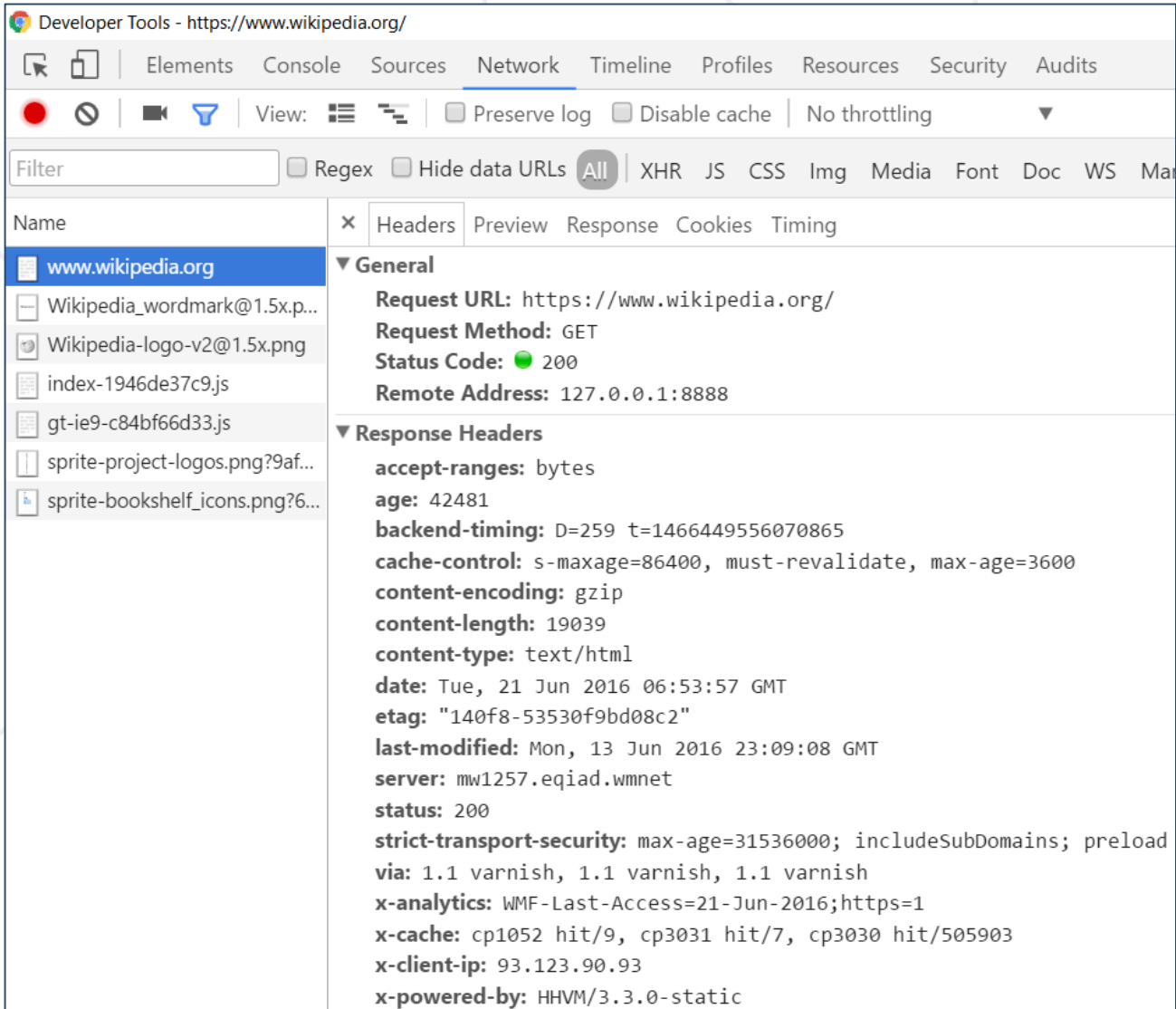






**HTTP Dev Tools**

# HTTP Developer Tools: Network Inspector



The screenshot shows the Chrome Developer Tools interface with the Network tab selected. The left sidebar lists several resources, with 'www.wikipedia.org' highlighted. The main panel displays the details for this request:

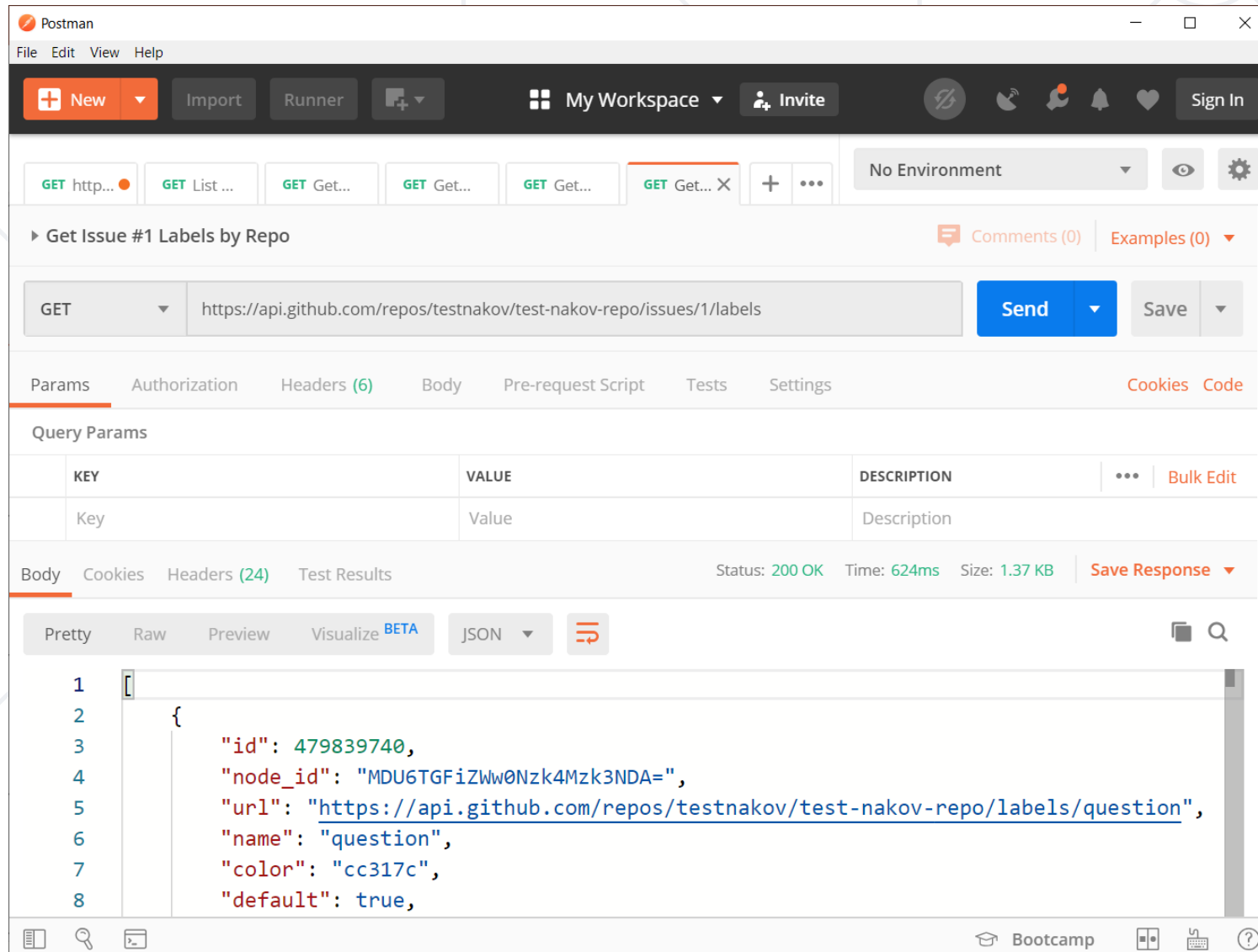
- Request URL:** https://www.wikipedia.org/
- Request Method:** GET
- Status Code:** 200
- Remote Address:** 127.0.0.1:8888

Below the request details, the 'Response Headers' section is expanded, showing various headers:

- accept-ranges:** bytes
- age:** 42481
- backend-timing:** D=259 t=1466449556070865
- cache-control:** s-maxage=86400, must-revalidate, max-age=3600
- content-encoding:** gzip
- content-length:** 19039
- content-type:** text/html
- date:** Tue, 21 Jun 2016 06:53:57 GMT
- etag:** "140f8-53530f9bd08c2"
- last-modified:** Mon, 13 Jun 2016 23:09:08 GMT
- server:** mw1257.eqiad.wmnet
- status:** 200
- strict-transport-security:** max-age=31536000; includeSubDomains; preload
- via:** 1.1 varnish, 1.1 varnish, 1.1 varnish
- x-analytics:** WMF-Last-Access=21-Jun-2016;https=1
- x-cache:** cp1052 hit/9, cp3031 hit/7, cp3030 hit/505903
- x-client-ip:** 93.123.90.93
- x-powered-by:** HHVM/3.3.0-static

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

# HTTP Developer Tools: HTTP Client Tools



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



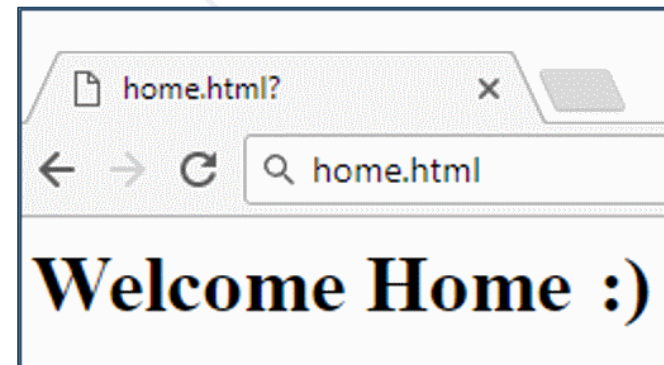
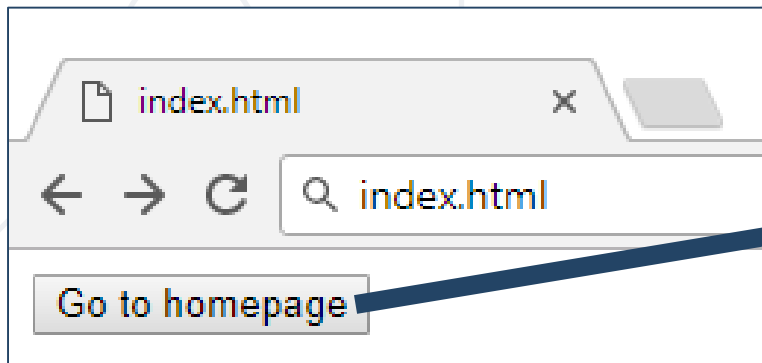
# HTML Forms

Form Submission: GET and POST

- 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



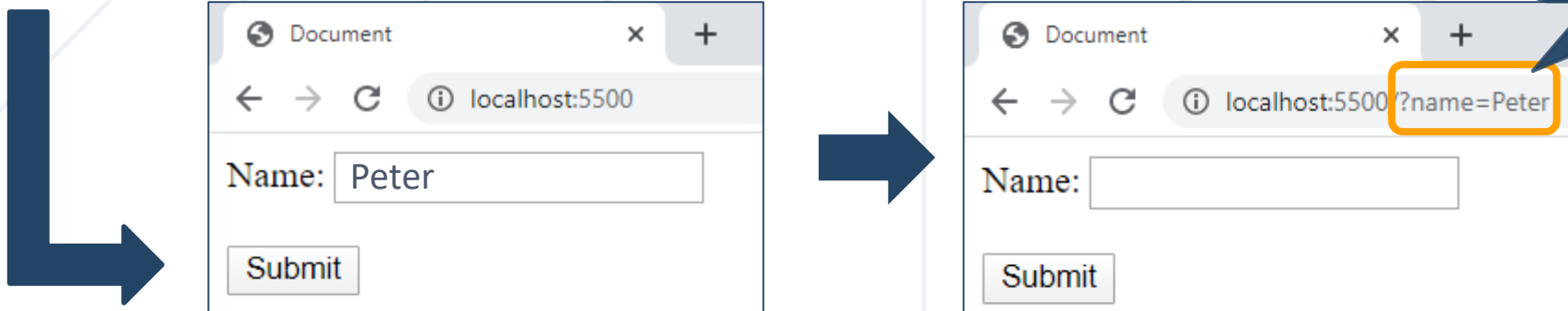
Example: <https://repl.it/@nakov/http-form-example>

# 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>
```

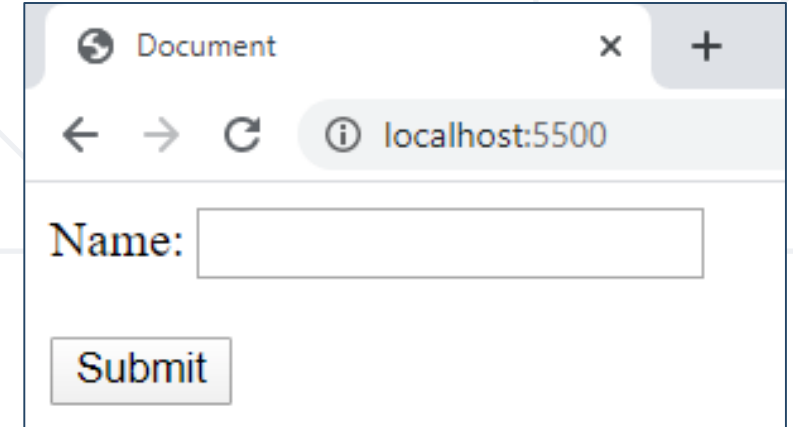
The form data  
is in the URL



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>
```



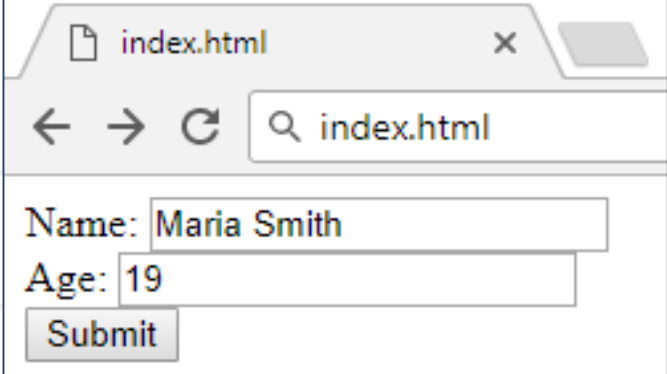
```
POST /index.html HTTP/1.1  
Host: localhost  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 10  
name=Peter
```

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>
```



A screenshot of a web browser window. The address bar shows 'index.html'. The page content includes a form with two text input fields: 'Name: Maria Smith' and 'Age: 19'. Below these fields is a 'Submit' button.

POST /index.html HTTP/1.1

Host: localhost

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

Content-Length: 23

name=Maria+Smith&age=19

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

URL-encoded form data

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	CRUD == the four main functions of persistent storage	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			

# 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>**

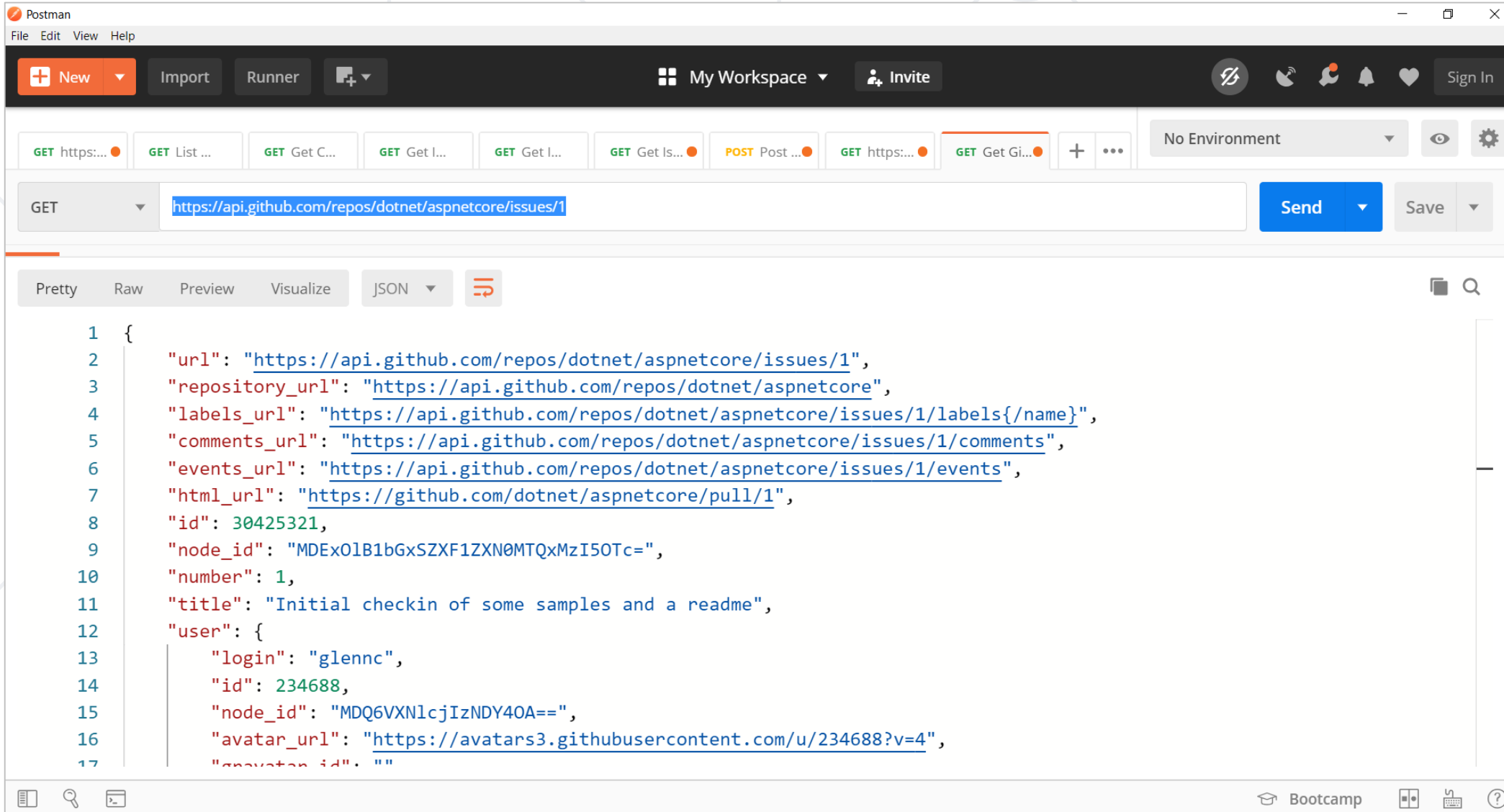
The request body is empty

Relative URI,  
not full URL

HTTP request line

HTTP headers

# HTTP GET – Example with Postman



The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', and 'My Workspace'. The main workspace shows a list of requests, with the selected one being a GET request to `https://api.github.com/repos/dotnet/aspnetcore/issues/1`. The request details are visible, showing the method 'GET' and the URL. The response is displayed in the 'Pretty' view, showing a JSON object with various fields including 'url', 'repository\_url', 'labels\_url', 'comments\_url', 'events\_url', 'html\_url', 'id', 'node\_id', 'number', 'title', and 'user'.

```
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": "MDExO1B1bGxSZXF1ZXN0MTQxMzI5OTc=",
10  "number": 1,
11  "title": "Initial checkin of some samples and a readme",
12  "user": {
13    "login": "glenn",
14    "id": 234688,
15    "node_id": "MDQ6VXNlcjIzNDY4OA==",
16    "avatar_url": "https://avatars3.githubusercontent.com/u/234688?v=4",
17    "avatar_id": ""
```

# HTTP POST Request – Example

**POST** /post **HTTP/1.1**

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

Host: postman-echo.com

HTTP request line

Accept: \*/\*

Accept-Encoding: gzip, deflate

Content-Type: application/json

Connection: keep-alive

Content-Length: 95

HTTP headers

<CRLF>

The request body holds  
the submitted data

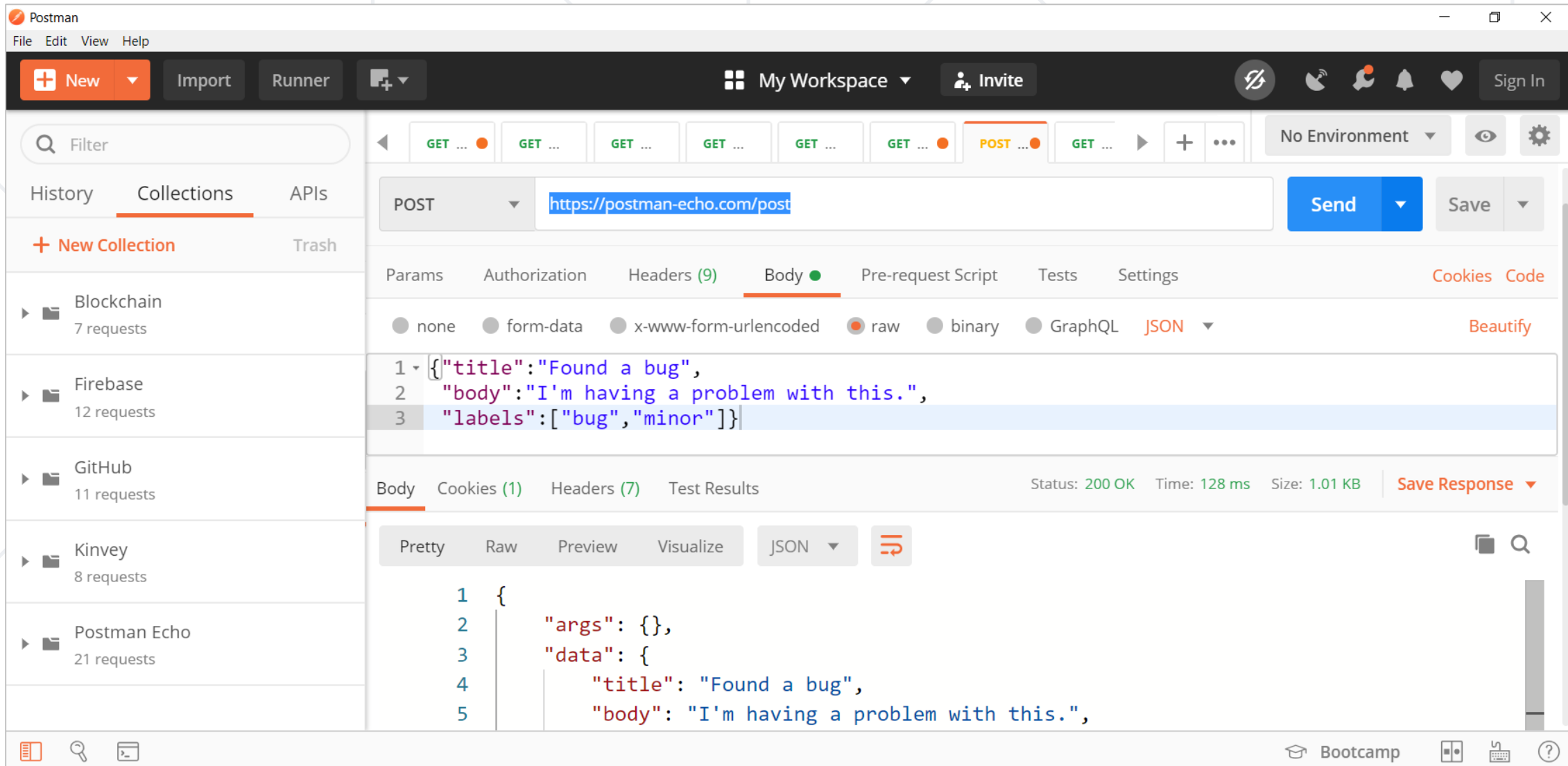
{"title": "Found a bug",

"body": "I'm having a problem with this.",

"labels": ["bug", "minor"]}

<CRLF>

# HTTP POST – Example with Postman





# 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

Content-Length: 84

Content-Type: text/html

HTTP response headers

<CRLF>

HTTP response body

<html>

<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

- 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)

`http://mysite.com:8080/demo/index.php?id=27&lang=en#lectures`

Protocol      Host      Port      Path      Query string      Fragment

- **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 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

- 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=1T4GGLL_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%B0_%D0%B0%D0%BA%D0%B0%D0%B4%D0%B5%D0%BC%D0%B8%D1%8F`

- Some invalid URLs

`http://google.com/search?&q=C# .NET 4.0`

Should be: `C%23+ .NET+4.0`

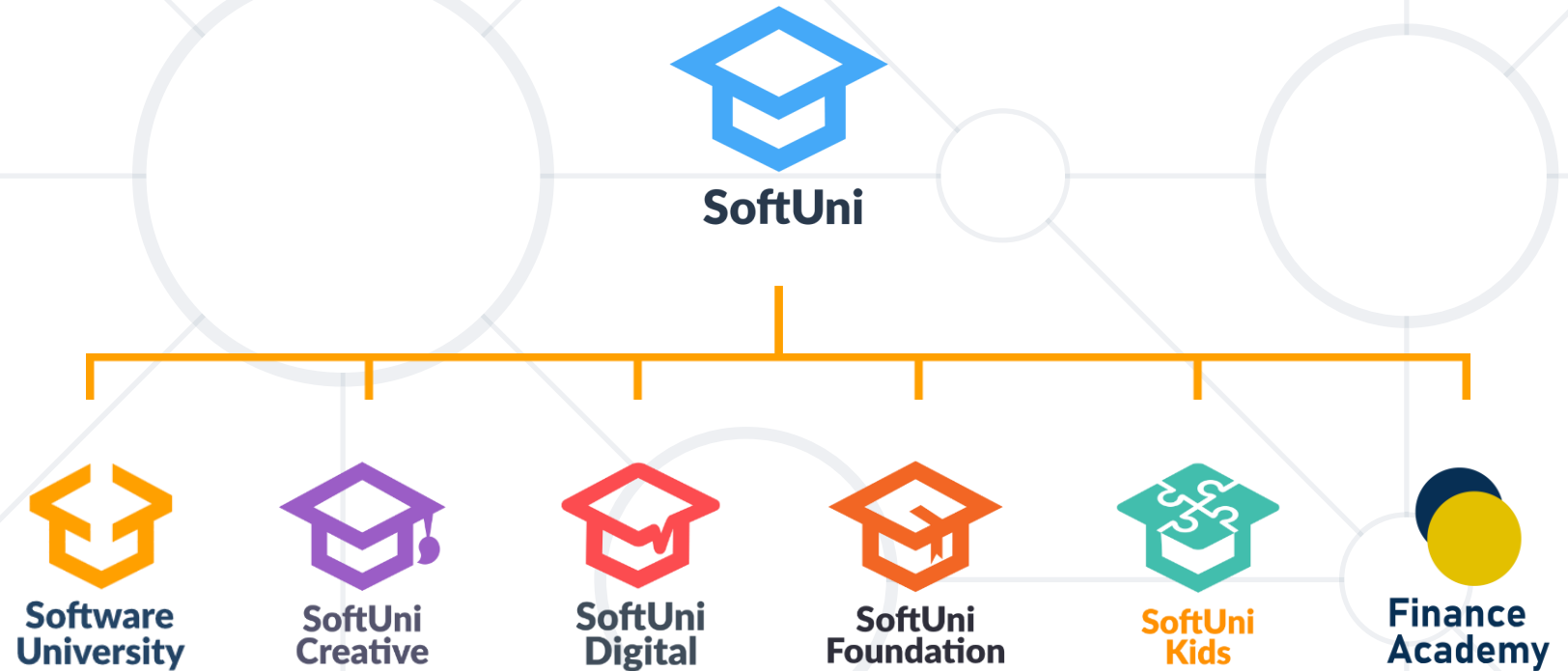
`http://google.com/search?&q=код`

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

- **H**yper**T**ext **T**ransfer **P**rotocol
  - 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**



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**



**POKERSTARS**  
POKER | CASINO | SPORTS  
a Flutter International brand

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

  
**DRAFT  
KINGS**



**SOFTWARE  
GROUP**

createX



**Postbank**  
Решения за твоето утре

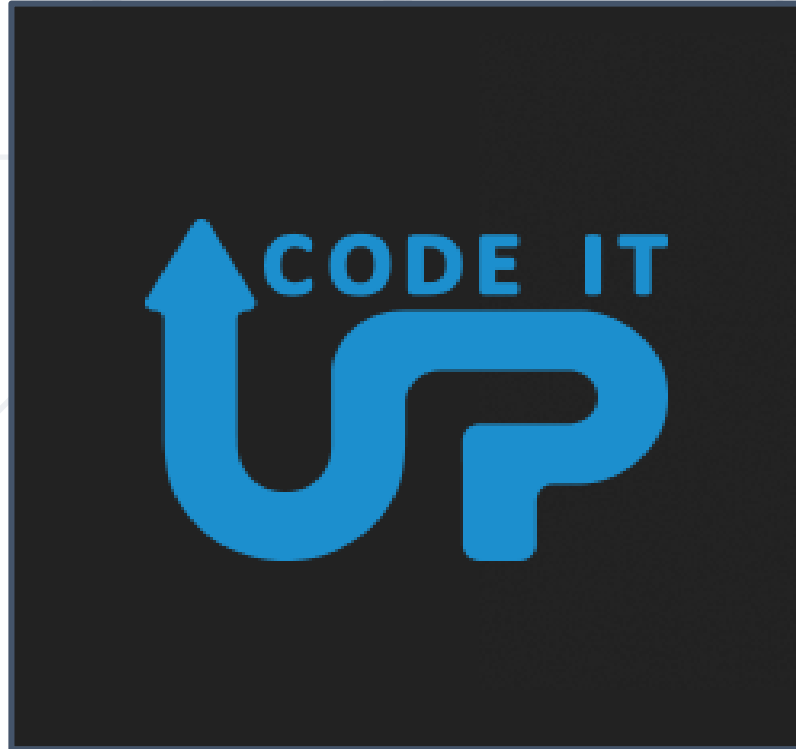


**BOSCH**

**DXC**  
TECHNOLOGY



**SmartIT**



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>

