

HTTP Basics

HTTP Request & HTTP Response



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#fund-common

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

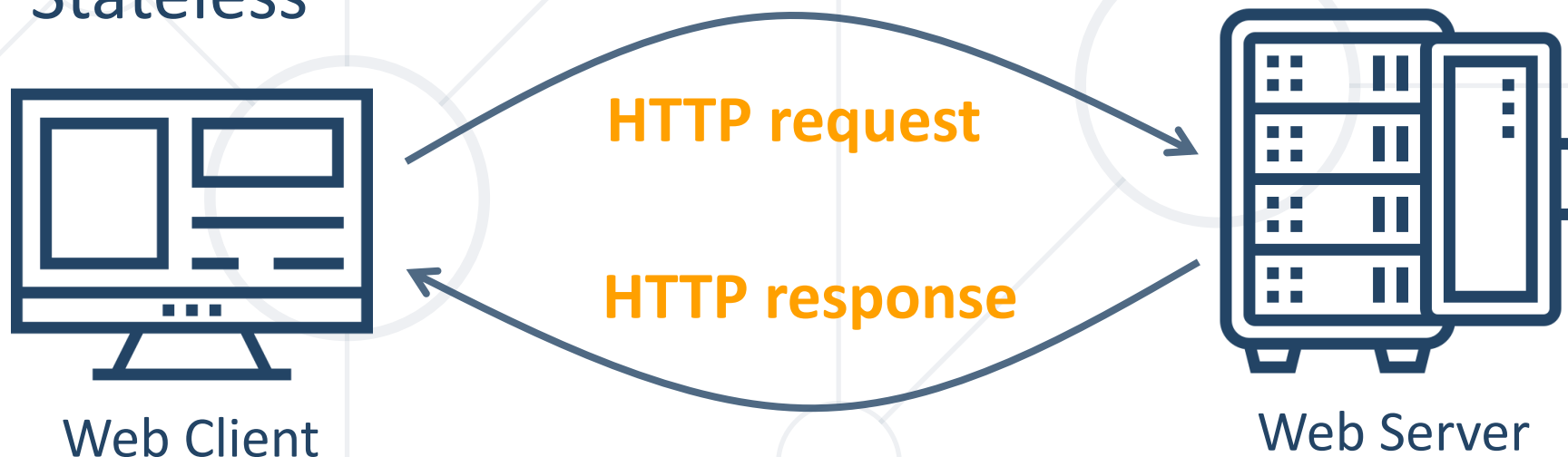


A background network diagram consisting of a grid of light gray lines intersecting at various points. At these intersections, there are several circles of different sizes, some solid light gray and some hollow, representing nodes in a network.

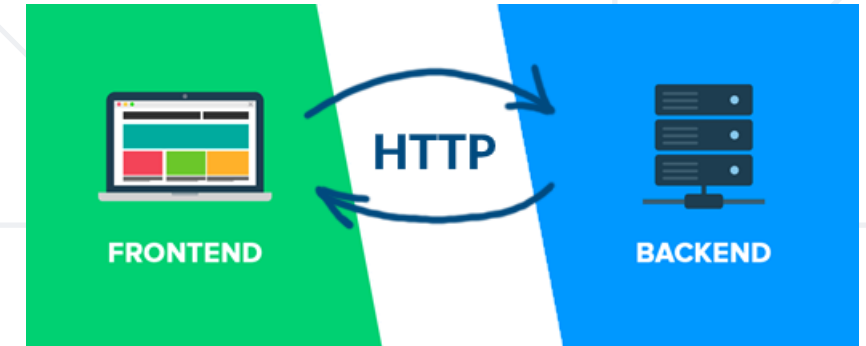
http://

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

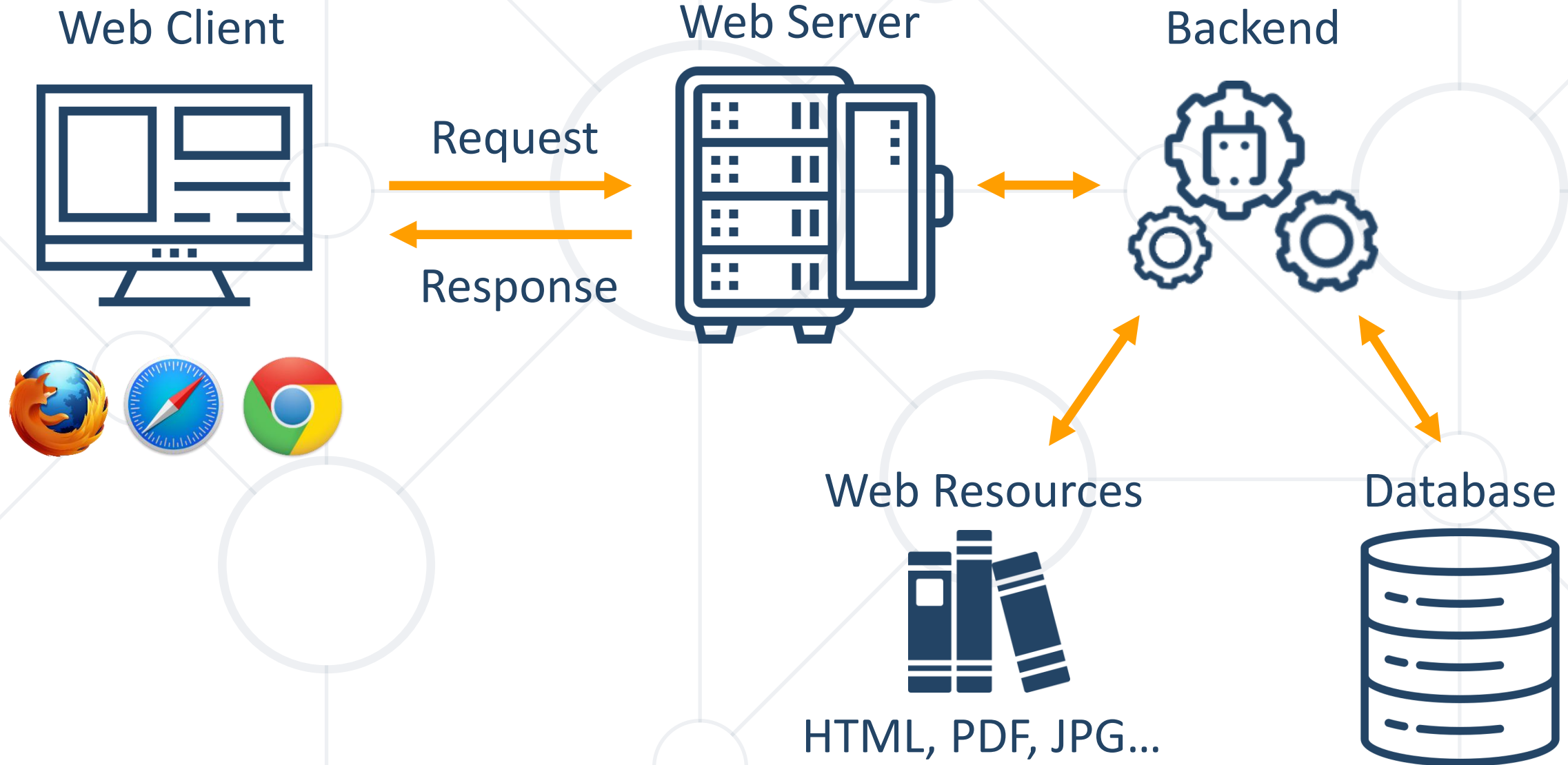


- **Front-end** and **back-end** separates the modern apps into **client-side** (UI) and **server-side** (data) components
- **Front-end** == client-side components (presentation layer), e.g., React app
 - Implement the **user interface** (UI)
- **Back-end** == server-side components (business logic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

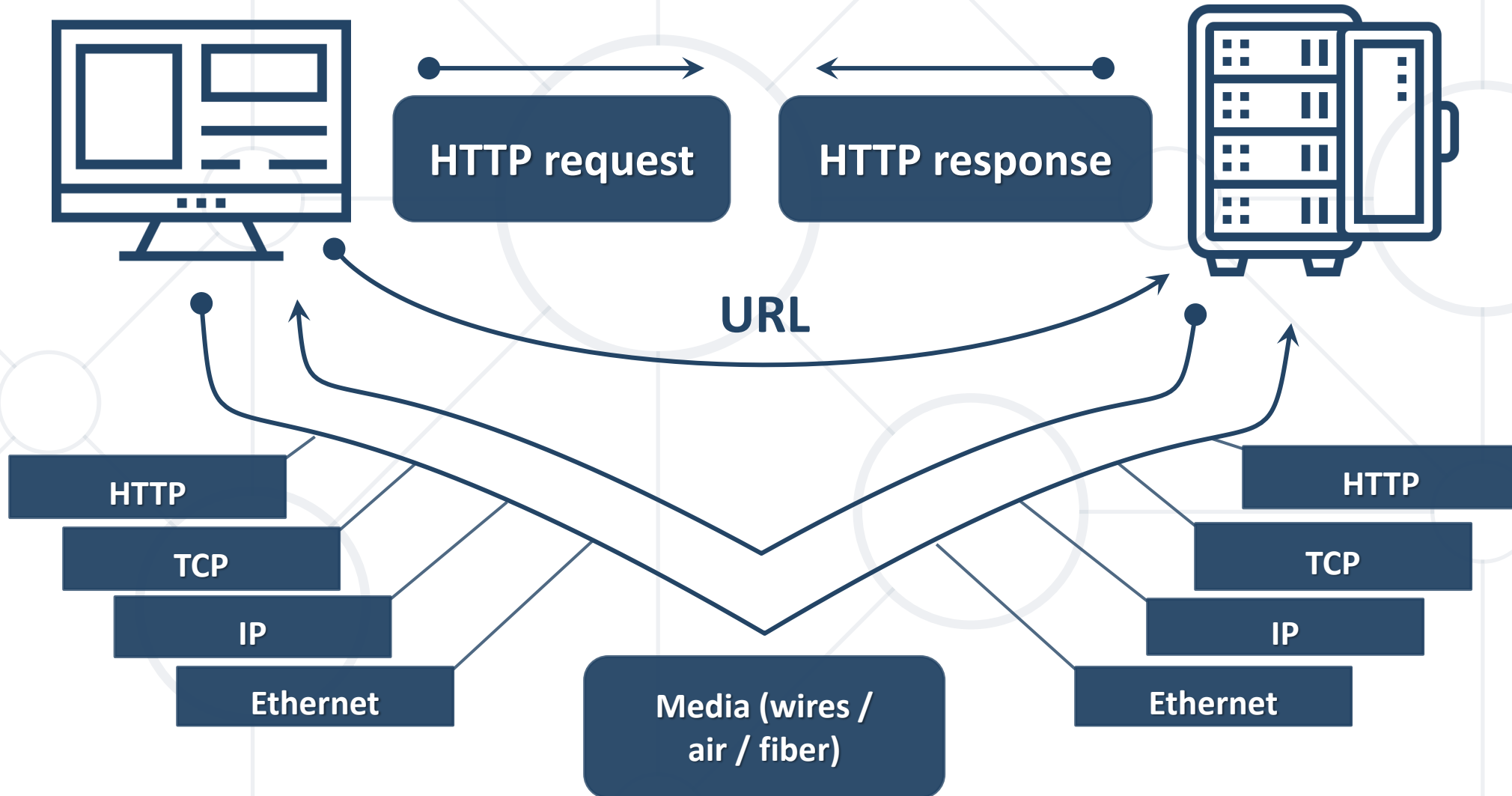


- **HTTP** connects front-end with back-end

The Client-Server Model in Web Apps



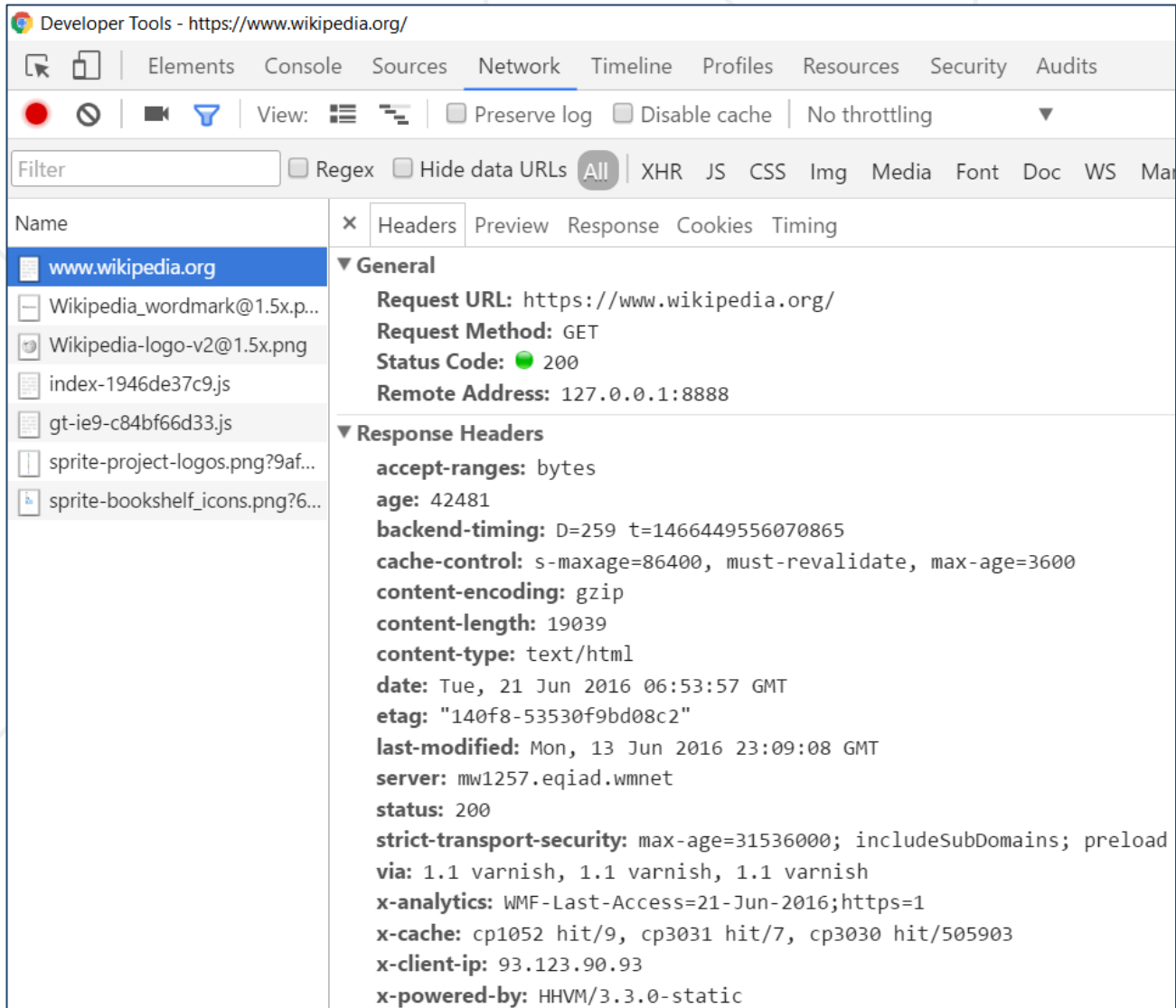
Network Layers and HTTP





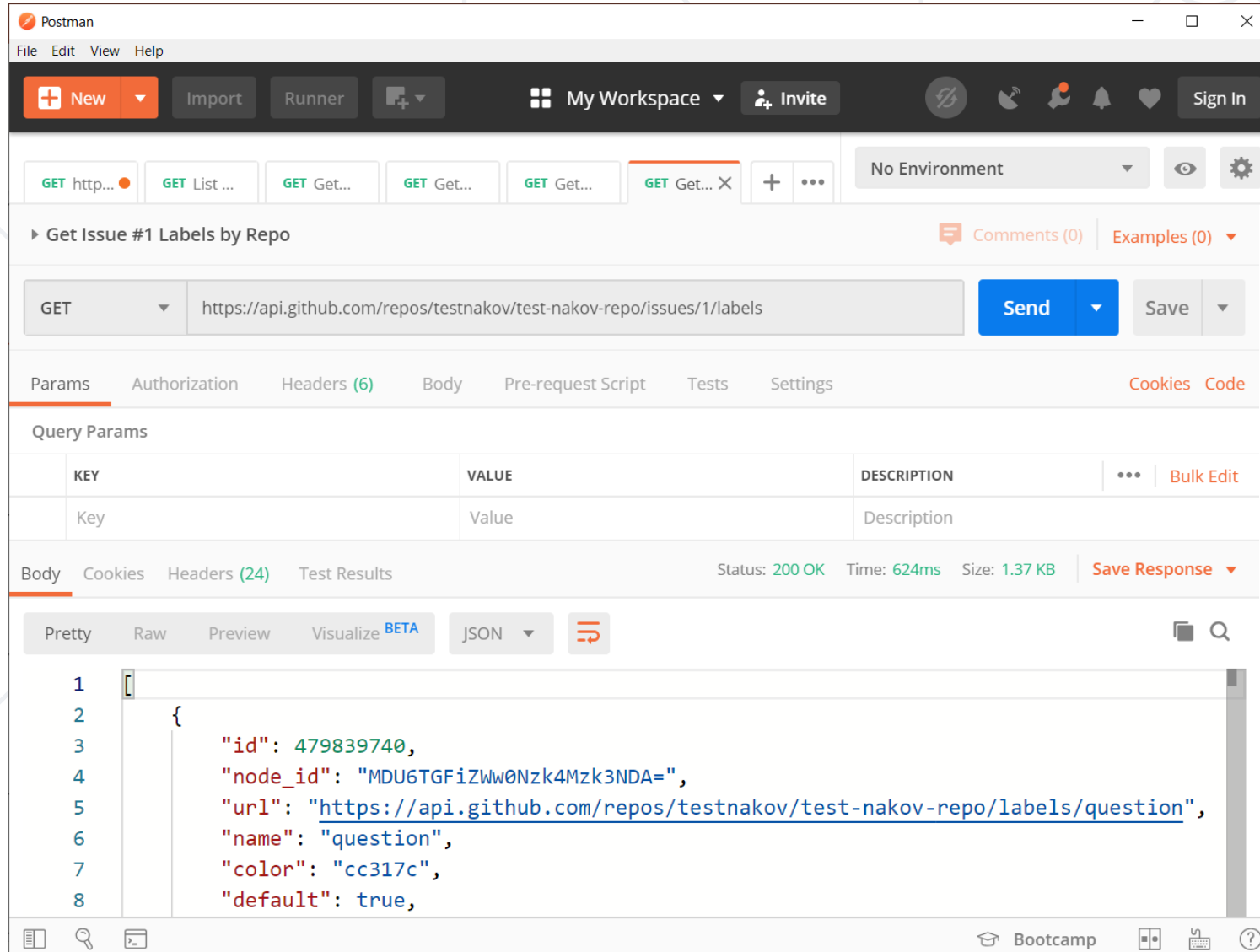
HTTP Dev Tools

HTTP Developer Tools: Network Inspector



- 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
- Hoppscotch



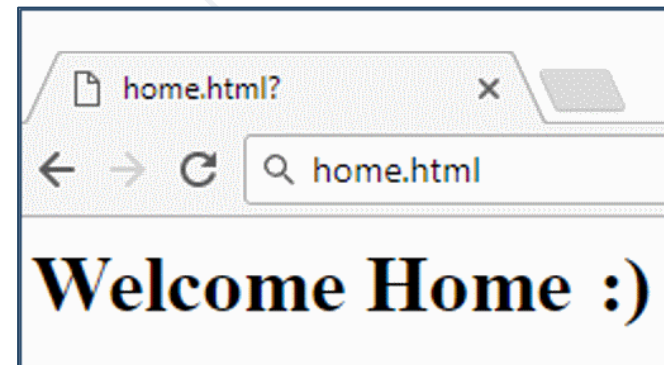
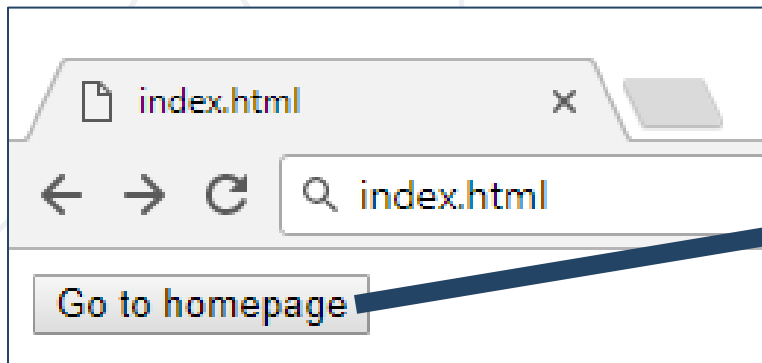
HTML Forms

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

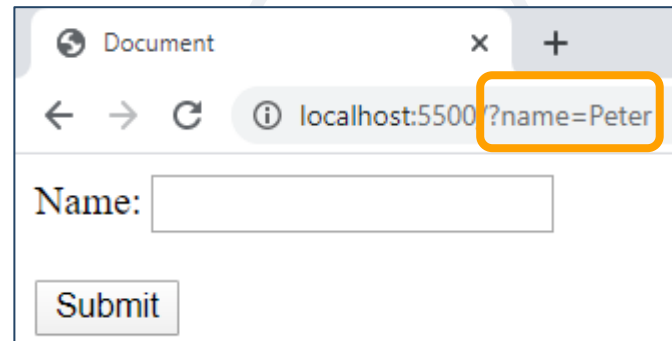
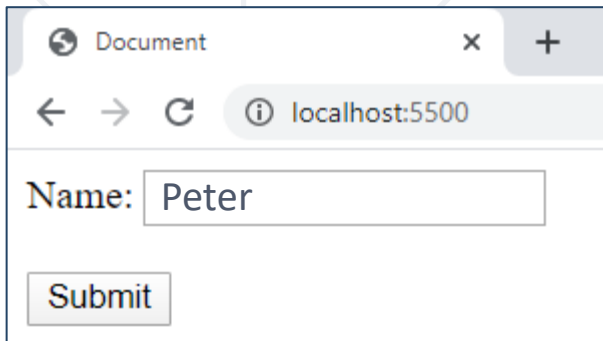


Example: <https://repl.it/@nakov/http-form-example>

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>
```

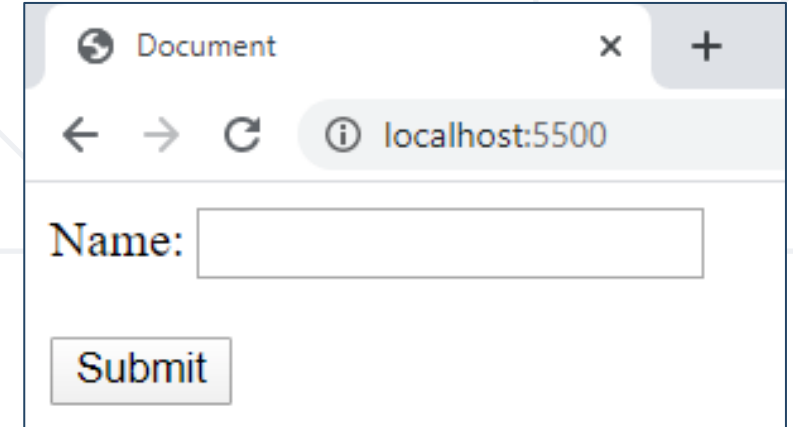


The form data is
in the URL

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
```

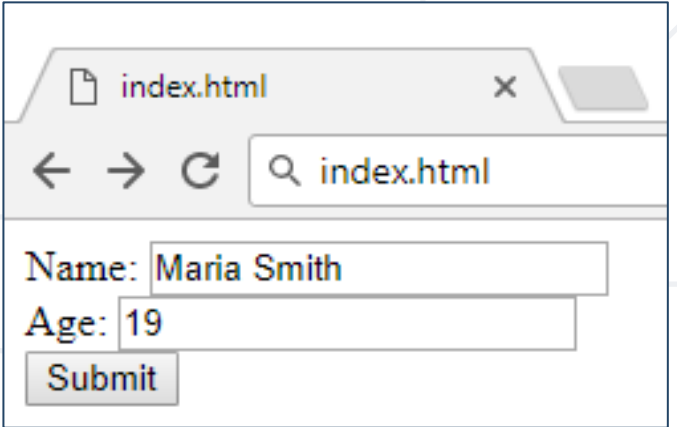
name=Peter

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>
```



index.html

← → ↻ 🔍 index.html

Name:

Age:

```
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	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>

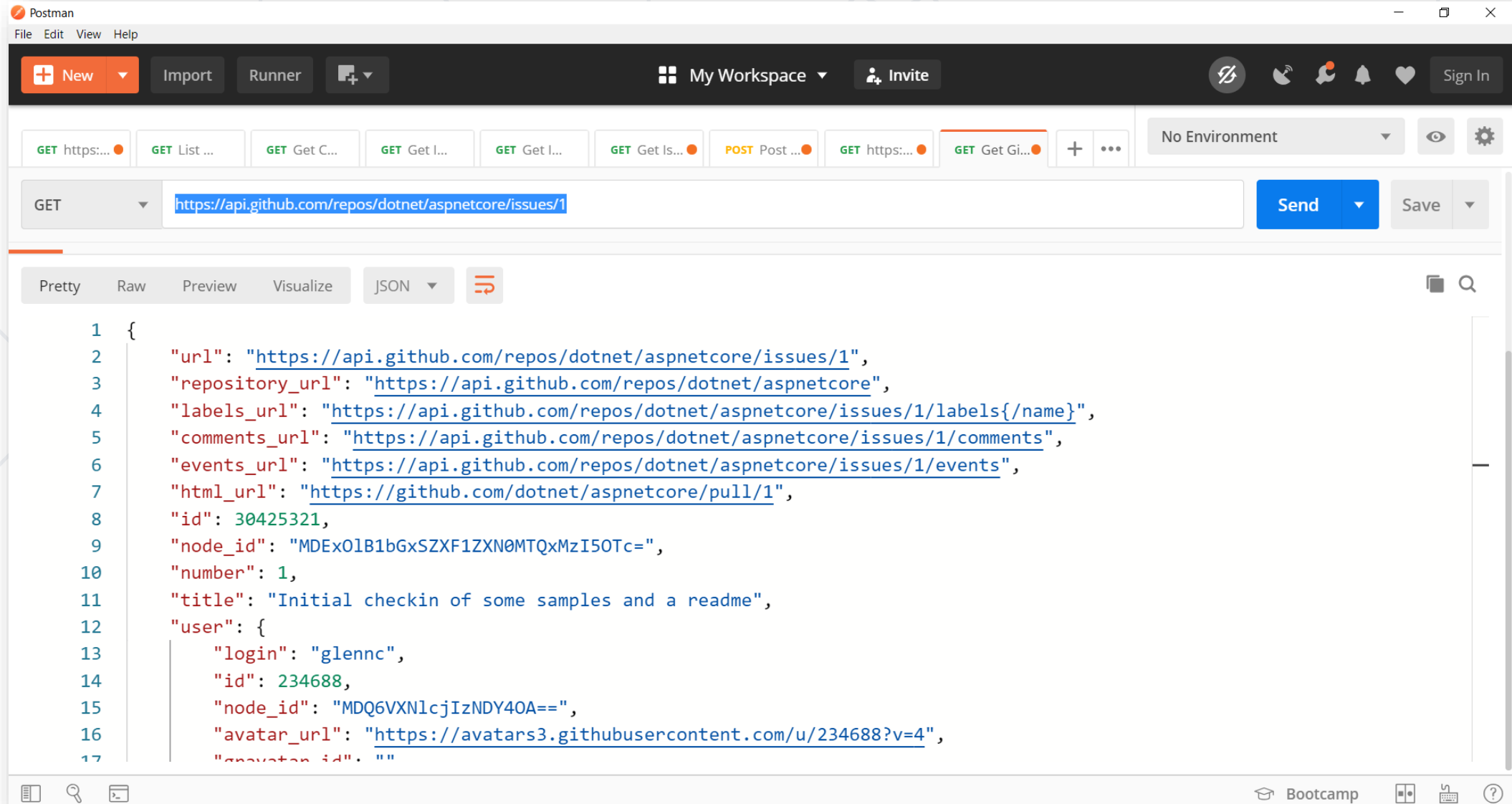
Relative URI,
not full URL

HTTP request line

HTTP headers

The request body is empty

HTTP GET – Example with Postman



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

HTTP headers

<CRLF>

{"title": "Found a bug",

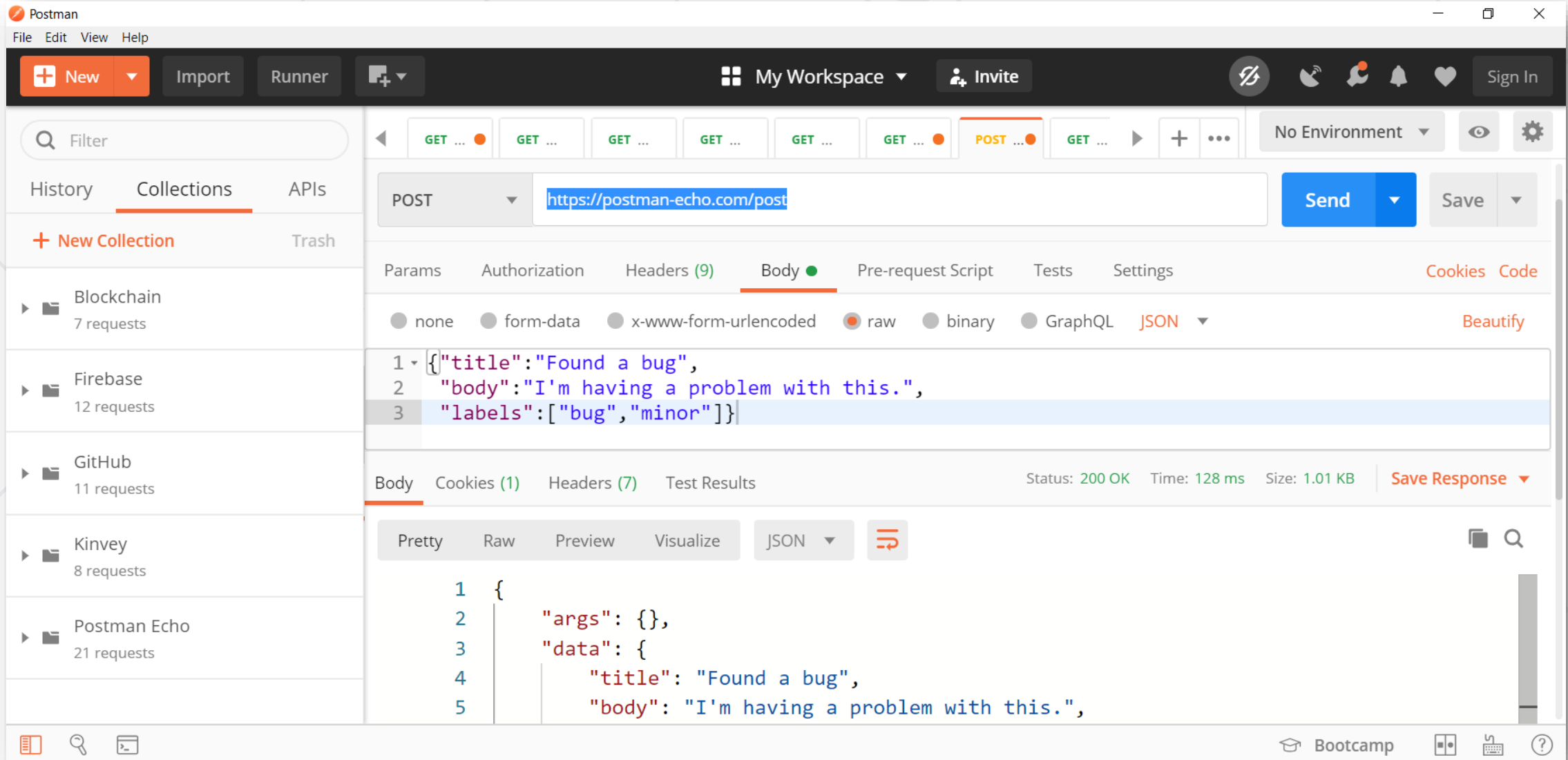
"body": "I'm having a problem with this.",

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

The request body holds
the submitted data

<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>

<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

- 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**

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

Download a PDF file

- 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`

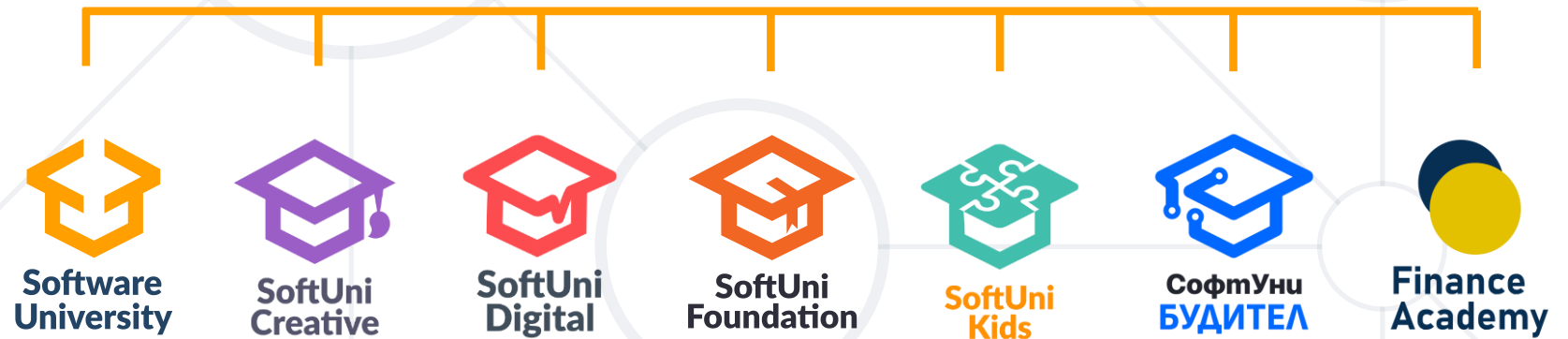
`http://google.com/search?&q=код`

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



- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, softuni.org
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity



- 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>

