Mini HTTP Server

Requests папка

Сега е време да съберем всичко написано до момента в главните функциониращи класове.

"Requests" папката ще съдържа класове и интерфейси за съхранение и манипулиране данни за HTTP заявките.

IHttpRequest

Създайте интерфейс, който се казва "**IHttpRequest**", който ще описва поведението на **Request** обекта.

```
public interface IHttpRequest
{
    string Path { get; }

    string Url { get; }

    Dictionary<string, object> FormData { get; }

    Dictionary<string, object> QueryData { get; }

    IHttpHeaderCollection Headers { get; }

    HttpRequestMethod RequestMethod { get; }
}
```

HttpRequest

Създайте клас, който се казва "**HttpRequest**", който имеплементира **IHttpRequest** интерфейса. Класът трябва да имплементира и методите на интерфейса.

```
public HttpRequest(string requestString)
{
    CoreValidator.ThrowIfNullOrEmpty(text: requestString, name: nameof(requestString));

    this.FormData = new Dictionary<string, object>();
    this.QueryData = new Dictionary<string, object>();
    this.Headers = new HttpHeaderCollection();

    //TODO: Parse request data...
}

public string Path { get; private set; }

public string Url { get; private set; }

public Dictionary<string, object> FormData { get; }

public Dictionary<string, object> QueryData { get; }

public IHttpHeaderCollection Headers { get; }

public HttpRequestMethod RequestMethod { get; private set; }
```

As you can see the **HttpRequest** holds its **Path**, **Url**, **RequestMethod**, **Headers**, **Data** etc. Those things come from the requestString, which is passed to its constructor. That's how a HttpRequest should be instantiated.

Както виждате "**HttpRequest**", съдържа **Path**, **Url**, **RequestMethod**, **Headers**, **Data**. Тези данни идвам променлива "**requestString**", която се подава в констуктора. Това е начина, по които "**HttpRequest**" ще се инициализира.

"**requestString**" ще изглежда по този начин:

```
{method}                                {url}                                {protocol}
{header1key}:                                                                {header1value}
{header2key}:                                                                {header2value}
...
<CRLF>
{bodyparameter1key}={bodyparameter1value}&{bodyparameter2key}={bodyparameter2value}.
..
```

ВНИМАНИЕ: Както вече знаете, че **body parameters** не за задължителни.

Нека да разбием една нормална заявка и да видим как тя трябва да се мапне към нашите пропъртита.

GET заявка

Request Line	`GET /home/index?search=nissan&category=SUV#hashtag HTTP/1.1`

HTTP Request Headers	`Host: localhost:8000`
	`Accept: text/plain`
	`Authorization: Bearer POwJDsBzl5nrxDF4jah64RtAM022XBFypl8h6lcgi`
	`Cache-Control: no-cache`
	`User-Agent: Chrome/64.5`

Empty line (/r/n)	`<CRLF>`

Request Line:

- **The Request Method** – Името на метода е винаги с **главни** букви, което означа, че някак си трябва да бъде форматиран, когато се парсва към **Enum-а** "**HttpRequestMethod**". (Не използвайте **switch/case** или **if/else** констркции за преобразуването към **Enum**).
- **The Request URL** – Целият **URL**, съдържа "**Path**", "**Query String**" и "**Fragment**".
  - Вземете "Path" частта от URL-а, като го разаделите, форматирате и запишете стойността в "Path" пропъртито.
  - **Вземете "Query String" частта и добавете стойностите към "Query Data" речника.**
    Параметрите трябва да бъдат преобразувани по следният начин": parameterName = key, parameterValue = value.
  - **Fragments** are mostly used on the client side, so there is no need to store them in our class, thus there is no property for them.
  - **Fragments** предимно се използва в "client-side" частта, затова няма пропърти за тях.
- **The Request Protocol** – **Трябва** да бъде: "**HTTP/1.1**".

Те лесно могат да се преобразуват в следният формат: "**{key}: {value}**". Трябва да ги разделите и да създадете нова инстанция на "**HttpHeader**", и след това да се добави към "**Headers**" на "**Request**".

Empty Line –краят на "**Request Headers**"

POST заявка

```
POST /home/index HTTP/1.1
Host: localhost:8000
Accept: text/plain
Authorization: Bearer POwJDsBzl5nrxDF4jah6 RtAM022XBFypl8h6lcgi
Cache-Control: no-cache
User-Agent: Chrome/64.5
<CRLF>
username=pesho&password=12345
```

Request Body

"**POST Request**" е почти същят, освен неговото "**body**". "**Request Body**" съдържа параметри, които трябва да бъдат прехвърлени към "**Form Data**" речника, по същият начин, както "**Query Parameters**" бяха прехвърлени към "**Query Data**" речника.

Сега е време да имплементираме повече логика, кото ознчава много методи, ако искаме да спазваме принципите за "**High-Quality Code**". Имплементирайте следните методи.

```
private bool IsValidRequestLine(string[] requestLine)...

private bool IsValidRequestQueryString(string queryString, string[] queryParameters)...

private void ParseRequestMethod(string[] requestLine)...

private void ParseRequestUrl(string[] requestLine)...

private void ParseRequestPath()...

private void ParseHeaders(string[] requestContent)...

private void ParseCookies()...

private void ParseQueryParameters()...

private void ParseFormDataParameters(string formData)...

private void ParseRequestParameters(string formData)...

private void ParseRequest(string requestString)...
```

`ParseRequest()` е метода откъдето започва всичко:

```
public HttpRequest(string requestString)
{
    CoreValidator.ThrowIfNullOrEmpty(text: requestString, name: nameof(requestString));

    this.FormData = new Dictionary<string, object>();
    this.QueryData = new Dictionary<string, object>();
    this.Headers = new HttpHeaderCollection();

    this.ParseRequest(requestString);
}
```

Нека да видим как изглежда той:

```
private void ParseRequest(string requestString)
{
    string[] splitRequestContent = requestString
        .Split(separator: new[] {GlobalConstants.HttpNewLine}, StringSplitOptions.None);

    string[] requestLine = splitRequestContent[0].Trim().
        Split(separator: new [] {' '}, StringSplitOptions.RemoveEmptyEntries);

    if (!this.IsValidRequestLine(requestLine))
    {
        throw new BadRequestException();
    }

    this.ParseRequestMethod(requestLine);
    this.ParseRequestUrl(requestLine);
    this.ParseRequestPath();

    this.ParseHeaders(splitRequestContent.Skip(1).ToArray());
    this.ParseCookies();

    this.ParseRequestParameters(splitRequestContent[splitRequestContent.Length - 1]);
}
```

Както виждате "**requestString**" е разделен на нови редове в масив. Взимаме първият ред (The **Request Line**) и го разделяме. След това следват серия от проверки и присвояване не стойности към пропъртита.

Ще се наложи вие да имплементирате тези методи. Разбира се, ще ви бъдат дадени насоки, как да се справите с тях.

IsValidRequestLine() метод

Този метод проверя дали, разделеният "**requestLine**" съдържа точно 3 елемента и също така дали последният елемент е равен на "**HTTP/1.1**". Метода връща булев резултат.

IsValidRequestQueryString() метод

Този метод се използва в "**ParseQueryParameters()**" метода. Проверява дали "**Query**" низа е **NOT NULL** или **празен** и също така дали има поне **един** или много **queryParameters**.

ParseRequestMethod() метод

RequestMethod присвоя стойността, като преобразуваме пръвият елемент от разделеният "**requestLine**".

ParseRequestUrl() метод

Url присвоява стойността от вторият елемент на разделеният "**requestLine**".

ParseRequestPath() метод

Path присвоява стойността, като разделим **Url** и вземем само пътя от него.

ParseHeaders() метод

Пропускаме първият ред от "**requestLine**" и обхождаме всички останали редове, докато не стигнем празен ред. Всеки ред представлява "**header**", който трябва да бъде разделен и преобразуван към правилният тип. След това информацията от низа е прехвърлена към "**HttpHeader**" обкета и е добавен към "**Headers**" пропертито на "**Request**".

Хвърлете "**BadRequestException**", ако "**Host"** липсва след преобразуването.

ParseQueryParameters() метод

Извадете "**Query**" низа, като разделите "**Request's Url**" и вземете само "**query**" от него. След това разделете "**Query**" низа в различни параметри и го прехвърлете към "**Query Data Dictionary**".

Валидирайте "**Query**" низа, като извикате "**IsValidrequestQueryString()**"метода.

Ако в " **Request's Url**" липсва " **Query**" низа, не предприемайте действия.

Хвърлете "**BadRequestException**", ако "**Query**" не е валиден.

ParseFormDataParameters() метод

Разделете "**`Request`**'s **`Body`** " в различни параметри и го добавате към "**`Form Data Dictionary`**".

Не предприемайте действия, ако "**`Request`**" не съдържа тяло.

ParseRequestParameters() метод

Този метод извиква "**`ParseQueryParameters()`**" и " **`ParseFormDataParameters()`**" методите. Това е просто "**wrapping**" метод.

Ако сте имплементирали всички правилно, би трябвало да преобразувате дори и много сложни заявки без проблем.

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).