# Lab: HTTP Protocol

This document defines the **in-class exercises** assignments for the ["C# Web Development Basics" course @ Software University](https://softuni.bg/courses/csharp-web-development-basics). You can submit your code in the course web site.

For these exercises there will be no actual web development. These are just simple console application, which aim to train your newly learned knowledge of the HTTP Protocol and its essentials.

1. **URL Decode**

You have been tasked to write a simple Console URL Decoder. You will receive an encoded URL. Your job is to decode the URL and print its decoded version on the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| http://www.google.bg/search?q=C%23 | http://www.google.bg/search?q=C# |
| https://mysite.com/show?n%40m3= p3%24h0 | https://mysite/show?n@m3= p3$h0 |
| http://url-decoder.com/i%23de%25?id=23 | http://url-decoder.com/i#de%?id=23 |

### Hints

Use one of the methods of the WebUtility class.

1. **Validate URL**

You have been tasked to create a simple Console URL Validator. The validator works with custom rules though.

You will receive **encoded URL**. **Decode** the **URL** and **validate** it.

* If the URL is **valid**, you must **break it** into its **composite** **particles** and **print** them.
* If the URL is **invalid**, you must **print** an **error**.

### URL Validation

A URL is validated by the following rules:

|  |  |
| --- | --- |
| Protocol | **Required** |
| Host | **Required** |
| Port | **Required** (default value for http - **80**, for https - **443**) |
| Path | **Required** (default value: /) |
| Query Strings | Optional (multiple query strings are separated by &) |
| Fragment | Optional |

### Example URLs

#### Valid URLs

* http://mysite.com:80/demo/index.aspx
* https://my-site.bg
* https://mysite.bg/demo/search?id=22o#go

#### Invalid URLs

* https://mysite:80/demo/index.aspx
* somesite.com:80/search?
* https/mysite.bg?id=2

### Output

#### Valid URL

A valid URL’s parts should be printed in the following format:

"Protocol: {protocol}"

"Host: {host}"

"Port: {port}"

"Path: {path}"

"Query: {query string}" (if any)

"Fragment: {fragment}" (if any)

#### Invalid URL

An invalid URL input should produce the following result:

"Invalid URL".

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| http://softuni.bg/ | Protocol: http  Host: softuni.bg  Port: 80  Path: / |
| https://softuni.bg:447/search?Query=pesho&Users=true#go | Protocol: https  Host: softuni.bg  Port: 447  Path: /search  Query: Query=pesho&Users=true  Fragment: go |
| http://google.com:443/ | Invalid URL |

### Hints

URL value can be encoded, so it's a good idea to use WebUtility.UrlDecode to decode it.

## Request Parser

You have been tasked to write a **console application** that simulates an **HTTP Server**’s behavior. You will be receiving **HTTP requests** and printing **HTTP responses** for them in a custom format.

You will receive **several lines** with **valid paths**. The **last part** of the **path** will be the **allowed method**. Keep reading paths until you receive "END".

After that, you will receive a **HTTP request**. You will have to **parse** the **request** and **return** a response based on that request.

If the **path** of the **given** request **CANNOT** be **found** in the **received paths** or the **request method** is **NOT allowed** for the **path**, the result will be:

* Status: 404 Not Found
* Response Text: NotFound

In all other cases the result will be:

* Status: 200 OK
* Response Text: OK

### Input

The input comes in the format "{path}/{method}".

### Output

Write the result on the console in the following format:

"HTTP/1.1 {status code}"

"Content-Length: {length of response text}"

"Content-Type: text/plain"

*"\n"*

"{response text}"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| /register/get  /register/post  END  GET /register HTTP/1.1 | HTTP/1.1 200 OK  Content-Length: 2  Content-Type: text/plain  OK |
| /login/get  /register/get  /login/post  END  POST /register HTTP/1.1 | HTTP/1.1 404 NotFound  Content-Length: 9  Content-Type: text/plain  NotFound |
| /index/get  /index/post  END  POST /login HTTP/1.1 | HTTP/1.1 404 NotFound  Content-Length: 9  Content-Type: text/plain  NotFound |