# Exercises: Handmade HTTP Server Part 2

Problems for exercises and homework for the [“C# Web Basics” course @ SoftUni](https://softuni.bg/courses/csharp-web-development-basics).

In the following exercises, you will be required to use the HTTP Server you created. (No more Apache ☺).

## File Repository

Create a web application that **shows files and folders in given directory on your local hard drive** (for example “C:\Repo”). When you try to reach the main route of the website show the directory and show the list of files inside that directory. Look at the examples for better understanding of how the application should look like.

### Examples

|  |  |
| --- | --- |
| **Initial state** | **Result** |
| C:\Users\Valio\AppData\Local\Microsoft\Windows\INetCacheContent.Word\repo.png |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Hints

* You need to **convert the request URL to local path**. For example, if your local directory is located in “C:\Repo” the following transformations should apply:
  + From localhost:8081 to C:\Repo
  + From localhost:8081/folder to C:\Repo\folder
  + From localhost:8081/test-text.txt to C:\Repo\test-text.txt
* Search in internet how to **obtain files and folders** by given directory path using C#
* To be able browser to open files he needs to know their [**Media Type**](https://en.wikipedia.org/wiki/Media_type) (or MIME Type). For example, if you try to return response containing **.jpeg** image, the response’s header content type must be set to “**image/jpeg**”. Consider that when returning response with content. Use the provided QuickMimeTypeMapper.cs class to help you picking up the right MIME type based on the file extension.

### \*Bonus Tasks

* \*\*Add functionality to **create new folder**
* \*\*\*\*Add functionality to **upload file**

## HTTP Server Unit Tests

Finally, we should write some unit tests to be sure our server works as expected. **Write unit tests** to make sure CookieCollection and HttpProcessor classes work as expected. In the tests, you need to cover at least the following scenarios:

* CookieCollection class
* **Add cookie** to the collection
* **Get cookie** by given name (index)
* **Add cookie** by given name and cookie itself (index)
* **Change existing cookie** in collection by given name (index)
* Check if **Count** property works as expected
* Check if **Contains** function works as expected
* **Loop** through **the collection** of cookies
* Check if cookies are **properly printed** (separated with “; “) when printed via ToString() method
* HttpProcessor class
  + Process **GET** request and return response with code **200 (OK)**
  + Process **GET request** **throwing Exception** and return response with code **500 (Internal Server Error)**
  + Process **GET** request and return response with code **404 (Not Found)**
  + Process **GET** request with query string and return response with code **200 (OK)**
  + Process **POST** request and return response with code **200 (OK)**
  + Process **POST** request and return response with code **404 (Not Found)**
  + Process **POST** request and return response with code **405 (Method Not Allowed)**
  + Process **GET** request and return response with code **405 (Method Not Allowed)**

### Hints

Refactor the HttpProcessor class so it can be easier for testing.