Files Web Service
Cpt S 422 Homework Assignment
by Evan Olds

Assignment Instructions:

Read all instructions *carefully* before you write any code.

In this assignment you will implement a web service that can be used with your server core from the previous assignments. This service will allow you to "share" a folder (and all contained files and subfolders) from your hard drive. The web service produces a listing of files in the browser and allows them to be downloaded.

Implement the FilesWebService Class

Create a class named **FilesWebService** in the **FilesWebService.cs** file. You will reference files from your web server core, so remember to include these in your submission zip. The same goes for the file used to implement the file systems from the previous homework.

Have the FilesWebService class inherit from the WebService abstract base class. Implement the following functions and properties in this class:

- FilesWebService(FileSys422 fs)
    - Constructor for the service that takes a reference to the file system to share. All content from this file system will be shared by your service. Recall from the previous homework that you can construct a StandardFileSystem (which inherits from FileSys422) from any folder on your computer and it will not let content above that directory be shared.
- string ServiceURI
    - For the ServiceURI property, return the hard-coded string "/files". This means you will need you navigate to localhost:4220/files/something in order to download files from this service in the browser.
- void Handler(WebRequest req)
    - First note that the web server core will not send any requests to this function that don't start with the string "/files" (provided you implemented the server core correctly).
    - Parse the URI and determine which of the 3 is the case:
        - The URI maps to an existing directory in the file system
        - The URI maps to an existing file in the file system
        - The URI maps to something that doesn't exist in the file system
    - Handle the request appropriately based on the case
        - File listing in HTML (further described below) for a directory URI

        - File contents (remember: as the response body!) for a file URI
        - 404 for any other URI
    - Provide support for partial content responses (i.e. support the Range header)
        - https://tools.ietf.org/html/rfc7233
- `string BuildDirHTML(Dir422 directory)`
    - This utility function builds the HTML string for a web page that provides a list of directory contents.
    - Since learning HTML is not really a major goal of the class, you are given the generic HTML format below. You can add additional formatting if you are familiar with HTML (and perhaps CSS). The only major requirements are:
        - Have valid, working links for all files and folders in the directory, in list (one link per "line") on the page. It should be easy for anyone to navigate the page.
        - The list of folder and file links are separated on the page such that it's easy to distinguish them visually. Since there is no requirement for icons by the links or any special formatting on the links, it might be difficult for the user to distinguish a link as going to another folder vs downloading a file if all the file and folder links were in one list and not separated sections.
        - The text of the link is the file or folder name only, not the full URI.

Listing Format String (example for what would be the root of the shared content)

```
<html>
  <h1>Folders</h1>
  <a href="/files/Documents">Documents</a>
  <br>
  <a href="/files/Desktop">Desktop</a>
  <br>
  <a href="/files/Downloads">Downloads</a>
  <br>
  <a href="files/Pictures">Pictures</a>
  <br>
  <h1>Files</h1>
  <a href="files/someFileInHomeFolder.txt">someFileInHomeFolder.txt</a>
  <br>
  <a href="files/another%20file.txt">another file.txt</a>
  <br>
</html>
```

Additional Details

- Have proper Content-Type headers for JPEG, PNG, PDF, MP4, TXT, HTML and XML files
- Have proper Content-Length headers for all responses
- Always stream file data for responses. Do NOT try to load the entire file into memory. Read it in chunks and send it in chunks. A chunk size of anywhere from 0.5 to 8 kb should be fine.