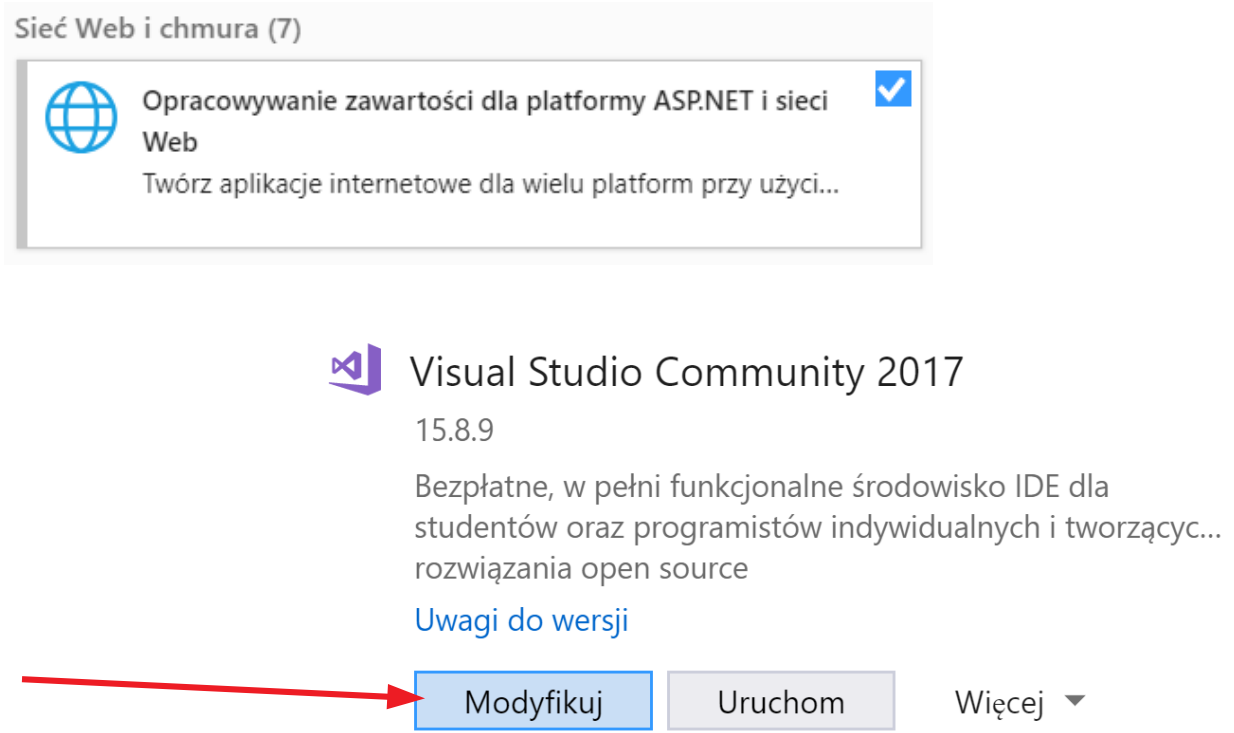
Advanced Programming C#

Lecture 4

dr hab. inż. Małgorzata Janik

malgorzata.janik@pw.edu.pl

Today you will need:



PROJECTS!

This week (or next Monday the latest) I would like to receive proposals for the Projects. Please send 1-2 sentences via Teams.

Project I

- Few summary slides about the projects.
- Presentation must include:
  - Idea, description & specification of the project
    - used technologies
  - Screenshots of the prototype of the application

- Presentation must be put in the github and then the link sent to malgorzata.janik@pw.edu.pl

Classes #6: Project I

- 10 min presentation / project
- Presentation must include:
  - Idea, description & specification of the project
    - used technologies
  - Screenshots of the prototype of the application
  - Interesting knowledge /skills obtained during the realization of the project (at least 1 example)
    - Should be presented in such a way that it would be interesting for other students
- Presentation must be sent to malgorzata.janik@pw.edu.pl latest 13.11, 9:00
- Prototype of the project should available for further checks and discussion

ASP .NET Core (Razor Pages)

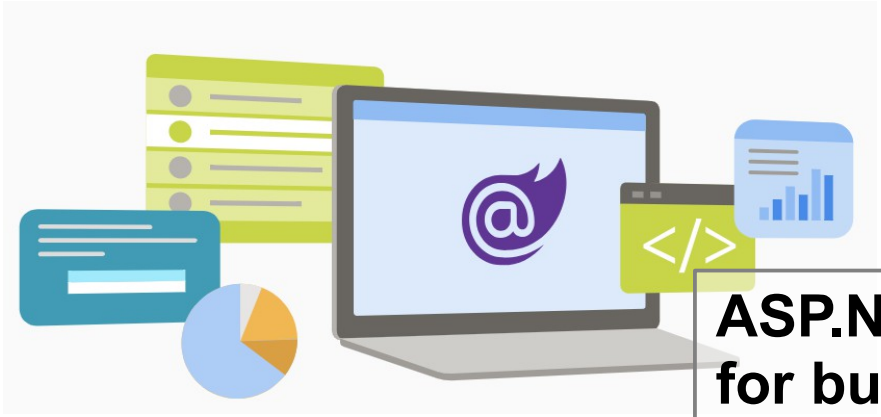
Web developement with .NET

ASP.NET Core

Free. Cross-platform. Open source.
A framework for building web apps and services with .NET and C#.

Get started

Supported on Windows, Linux, and macOS



ASP stands for **A**ctive **S**erver **P**ages

ASP.NET is an open source **web framework for building modern web apps and services** with .NET. ASP.NET creates websites based on HTML5, CSS, and JavaScript that are simple, fast, and can scale to millions of users.

https://www.asp.net/

ASP .NET Core Razor Pages

ASP.NET Core Razor Pages is a part of the ASP.NET web application framework and is included in Visual Studio.

What is Razor Pages

A framework based on Razor – a syntax that combines HTML with C#.

Each page consists of two files (encouraging separation of concerns):

- **Page.cshtml (HTML + Razor)**
- **Page.cshtml.cs (logic in C#)**

Logic and view are kept close together → easier to maintain.

Ideal for simple and medium-sized web applications.

Razor Pages makes use of the popular C# programming language for server-side programming, and the easy-to-learn Razor templating syntax for embedding C# in HTML mark-up to generate content for browsers dynamically.

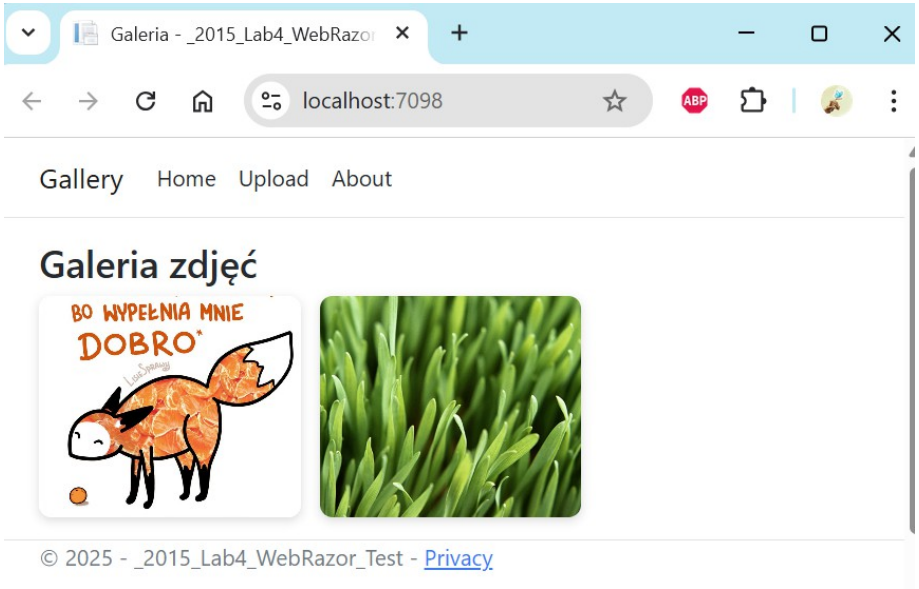
The Razor Pages framework is lightweight and very flexible. It provides the developer with full control over rendered HTML. Razor Pages is the recommended framework for cross-platform server-side HTML generation.

https://www.asp.net/web-forms/what-is-web-forms

Tasks

Demonstration

- Intro: create new ASP.NET Core Web App (Razor Pages), check it out, make some simple modifications



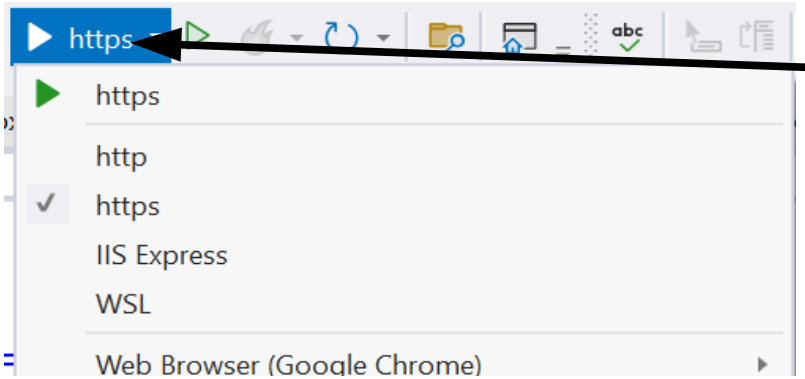
- File upload: allow users to upload their pictures into appropriate directory on the server
- Gallery: display uploaded pictures side by side
- If picture is clicked, bigger version should be displayed below

Tasks: Intro

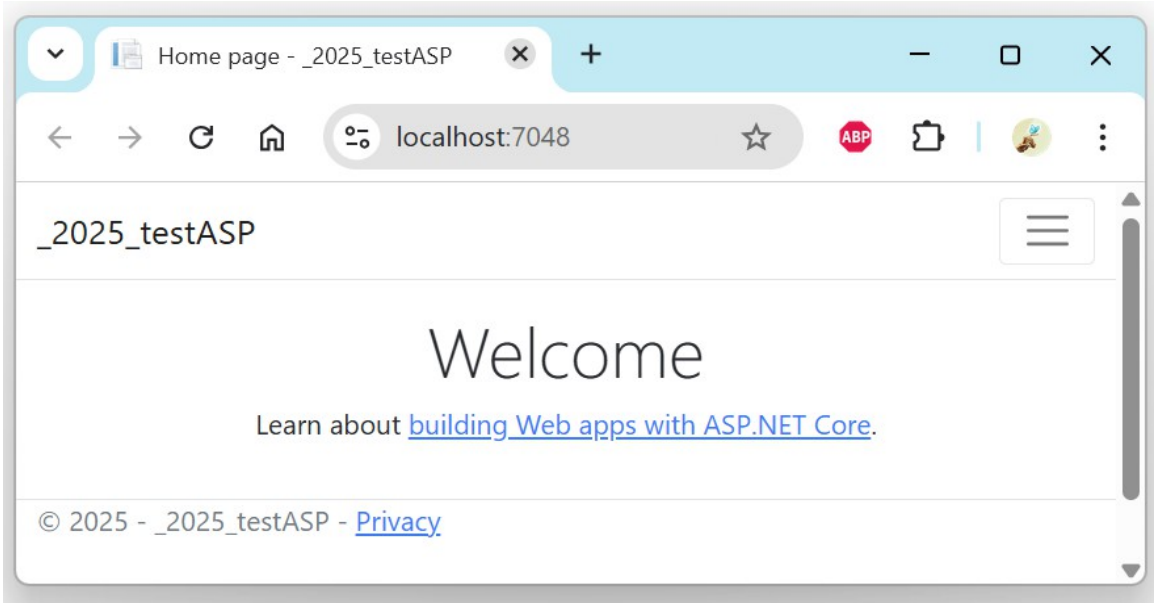
- **Create new ASP.NET Core Web App (Razor Pages): „PhotoGallery”**
- **Run the project in the browser**

- Modify the „Privacy” web page
- Add new Razor Page → „Upload”. Add it to the main menu (at the top)
  - Project → Add new... → Razor Page...
- Prepare Photo Gallery service

Run default project



in Lab228 use **http**



Modification of „Privacy” webpage

- Modify Privacy.cshtml page:

```
@page
@model PrivacyModel
@{
    ViewData["Title"] = "Privacy Policy";
}
<h2>Privacy Policy</h2>
<p>This photo gallery is a demo web application created for educational purposes.</p>


<h3>1. Uploaded Photos</h3>
<p> Photos uploaded through this website are stored locally on the server in the <code>wwwroot/uploads</code> folder.
They are not shared, analyzed, or transmitted to any third parties.
</p>


<h3>2. Personal Data</h3>
<p>The application does not collect or process personal data such as names, emails, or IP addresses beyond what is
technically necessary for hosting the website.</p>


<h3>3. Cookies and Logs</h3>
<p>No cookies or tracking technologies are used in this demo project.
    Basic server logs may record requests for diagnostic purposes.
</p>
<h3>4. Disclaimer</h3>
<p>This site is a student project.
    Uploaded content may be deleted at any time and should not include sensitive or copyrighted material.
</p>
```

- Run again

Adding new page

- Add new Razor Page → „Upload”. Add it to the main menu (at the top)
  - Project → Add new... → Razor Page...

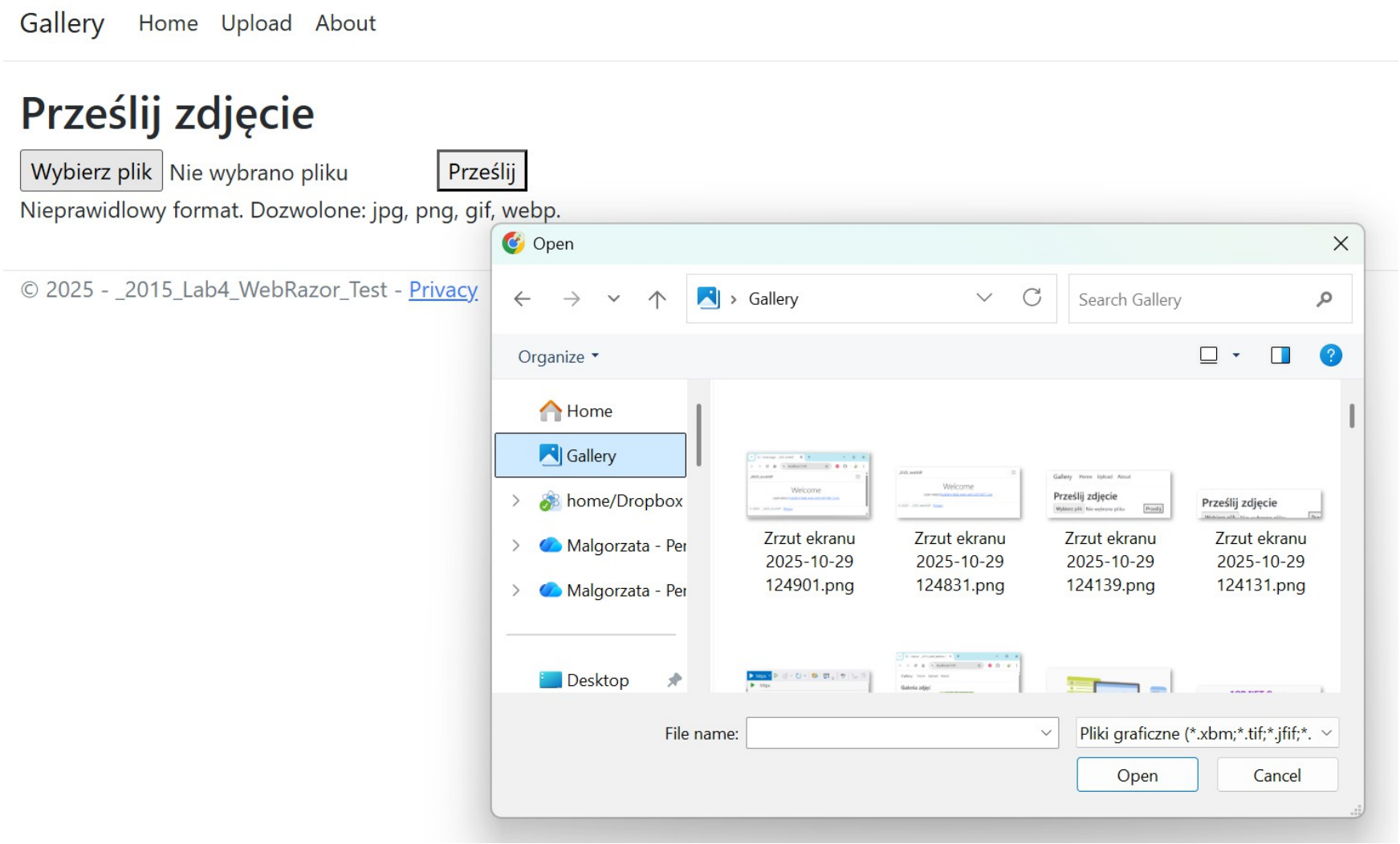
Add a link to the Upload page into the layout (Pages/Shared/_Layout.cshtml):

```
<li><a class="nav-link text-dark" asp-page="/Upload">Upload</a></li>
```

- Run again

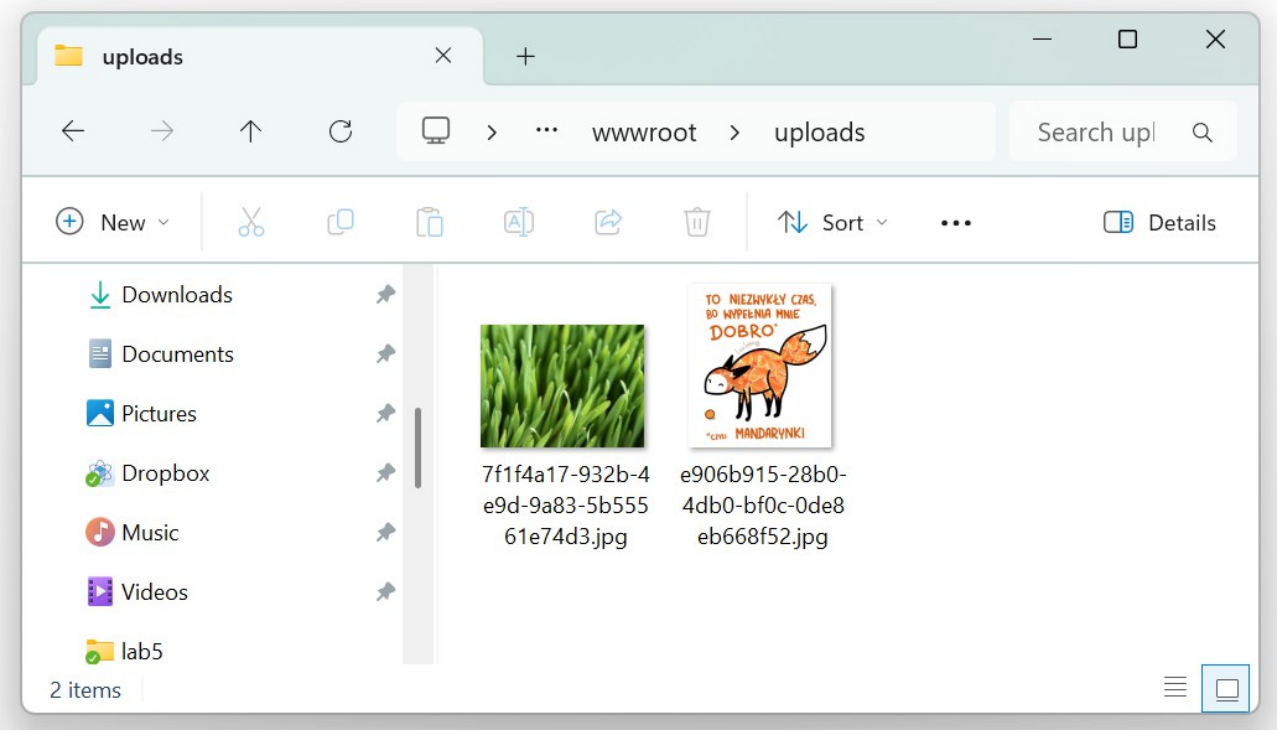
Tasks: Photo Gallery – File upload

- Idea:



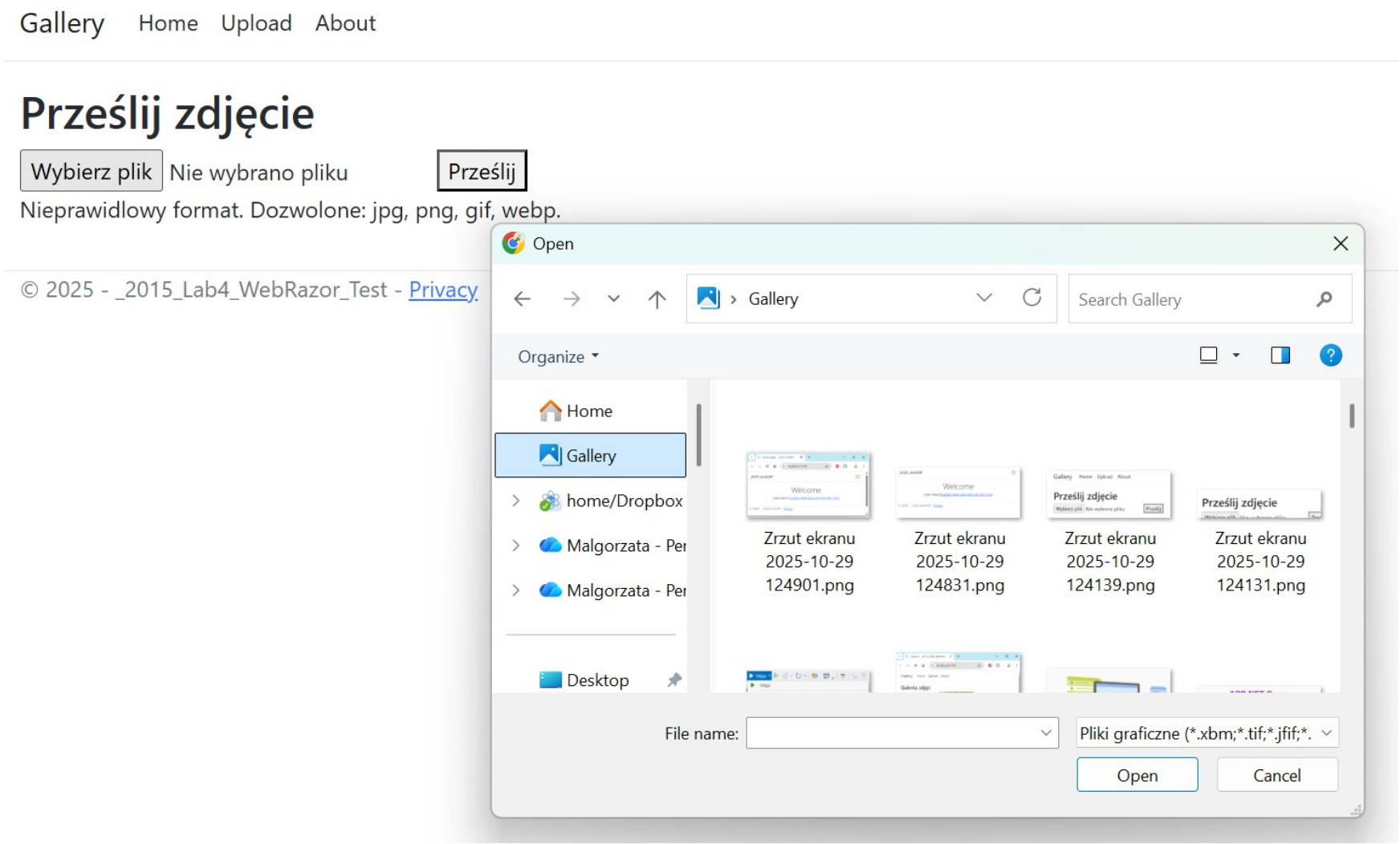
Tasks: Photo Gallery – File upload

- Uploaded files should be saved in „uploads” folder (located in wwwroot the project home folder)



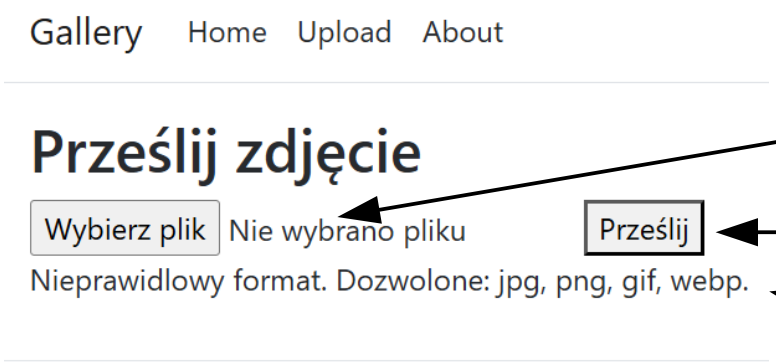
Tasks: Photo Gallery – File upload

- Idea:



Tasks: Photo Gallery – File upload

- Idea:



File upload (opens window with file browser)

Button

Status which displays information in case of problems.

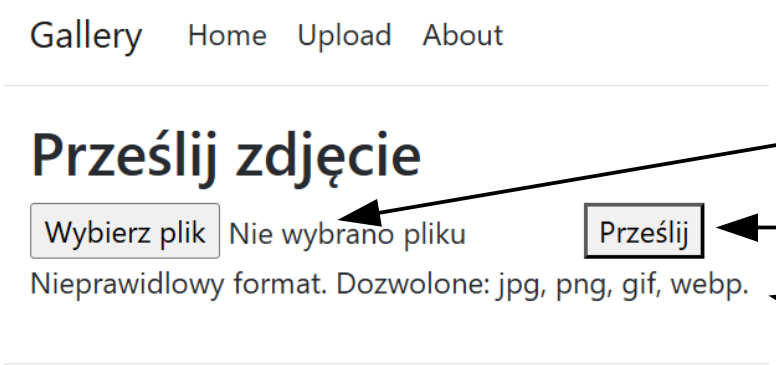
- Add controls:

```
<input type="file" name="ImageFile" accept="image/*" />
```

  - Creates a file input field.
  - type="file" — opens the file picker window.
  - name="ImageFile" — the field name used by ASP.NET Core to bind the uploaded file to the corresponding property (public IFormFile? ImageFile { get; set; }).
  - accept="image/*" — restricts file selection to images (jpg, png, gif, etc.).

Tasks: Photo Gallery – File upload

- Idea:



File upload (opens window with file browser)

Button

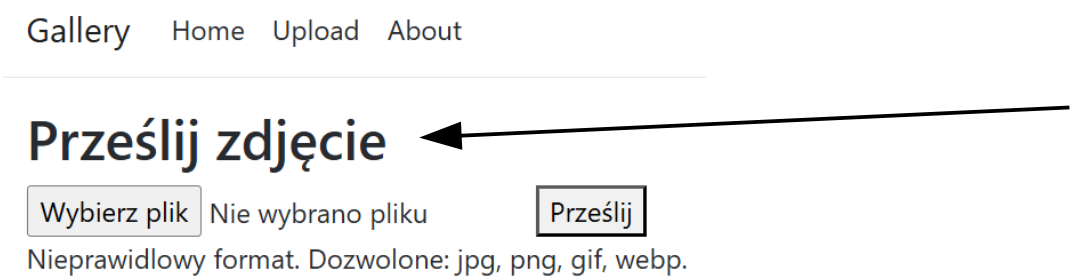
Status which displays information in case of problems.

- Add controls:

```
<button type="submit">Prześlij</button>
```

- Submits the form to the server when clicked.
- The browser sends an HTTP POST request to the same page (/Upload).
- On the server side, the OnPostAsync() method of the UploadModel class is executed.

Tasks: Photo Gallery – File upload



<h2> header with some info</h2>

Connect both controls in one form:

```
<form method="post" enctype="multipart/form-data">
    <input ...
    <button ...
</form>
```

Test how it looks!

- `<form>...</form>` - This tag wraps all the input fields. It tells the browser: “this is a set of data to be sent to the server.”
- `method="post"` - Specifies how the data is sent to the server. POST means: send the data in the body of the HTTP request (not in the URL). It’s typically used when sending files or modifying data. In Razor Pages, this triggers the OnPost or OnPostAsync() method in the corresponding .cshtml.cs file (in this case, Upload.cshtml.cs).
- `enctype="multipart/form-data"` - This defines the encoding type used when sending data. It’s required when uploading files. It tells the browser to send the file as binary data in separate parts of the HTTP request. If you used the default application/x-www-form-urlencoded, only the file name would be sent — not the file itself..

Upload file: model (C#)

- Model page adds functonality to the GUI we defined.
- **OnPostAsync()** will be called when button is clicked.
- **ImageFile** and **Message** properties allow for exchange of data with the page.

```
public class UploadModel : PageModel
{
        [BindProperty]
        public IFormFile? ImageFile { get; set; }

        public string? Message { get; set; }

        public void OnGet() { }

        public async Task<IActionResult> OnPostAsync()
        {



            return Page();
        }
}
```

Upload file: model (C#)

- Modify OnPostAsync method
  - Check:
    - If folder exists
      - Get folder path+name:

```
var uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(),
                                 "wwwroot", "uploads");
```

      - Check if folder exists:

```
Directory.Exists(uploadsFolder)
```

      - If not, create folder:

```
Directory.CreateDirectory(uploadsFolder);
```

Upload file: model (C#)

- IModify OnPostAsync method
  - Create new name for the uploaded file:
    - We want the name to be unique (if client uploades two files with the same name, we want to be able to store both)
    - We have to change default file name: insert Guid
      - NameOfFile.jpg → f47ac10b-58cc-4372-a567NameOfFile.jpg

Comes from Guid

    - Useful methods:

```
ImageFile.FileName;
```
→ returns filename

```
var uniqeID = $"{Guid.NewGuid()}";
```
→ creates a new unique identifier (a Globally Unique Identifier)

```
Path.Combine(uploadsFolder, uniqeID + fileName)
```
→ „Path.Combine()" method allows to create strings representing properly defined paths

You need to set correctly `string filePath` for the new (copied) file

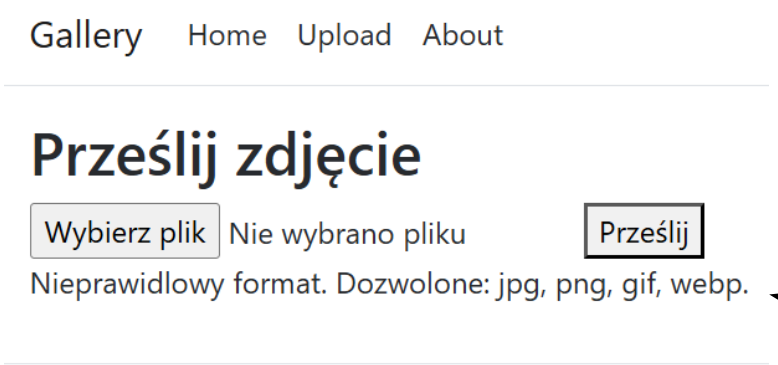
Upload file: model (C#)

- If all is set: copy the file!

```
public class UploadModel : PageModel
{
        [BindProperty]
        public IFormFile? ImageFile { get; set; }

        public string? Message { get; set; }

        public void OnGet() { }

        public async Task<IActionResult> OnPostAsync()
        {
            ……….

            using (var stream = System.IO.File.Create(filePath))
            {
                await ImageFile.CopyToAsync(stream);
            }

            Message = "Uploaded succsefully.";
            return Page();
        }
}
```

Test if copying works!
(check the folder)

Tasks: Photo Gallery – File upload



Status which displays information in case of problems.

- Add to the Razor page (after the form):

```
@if (Model.Message != null) {
    <p>@Model.Message</p>
}
```

- This is a Razor expression — it conditionally displays a message returned from the model, e.g.:
- “Upload successful” or “Error – please select a file.”

Tasks: Photo Gallery – File upload

- Modify OnPostAsync method
  - Check:
    - Is image uploaded?
      - `ImageFile` is not null and it's length is not 0
      - If not, change Message to appropriate text.
      - Remember to `return Page();` after Message is set.
    - If file type is one of the following: jpg / gif / png / webp?

```
var allowed = new[] { ".jpg", ".jpeg", ".png", ".gif", ".webp" };
var ext = Path.GetExtension(ImageFile.FileName).ToLowerInvariant();
if (!allowed.Contains(ext)) { … }
```

      - If not, write appropriate text in Message
    - If file size is < 5 MB?

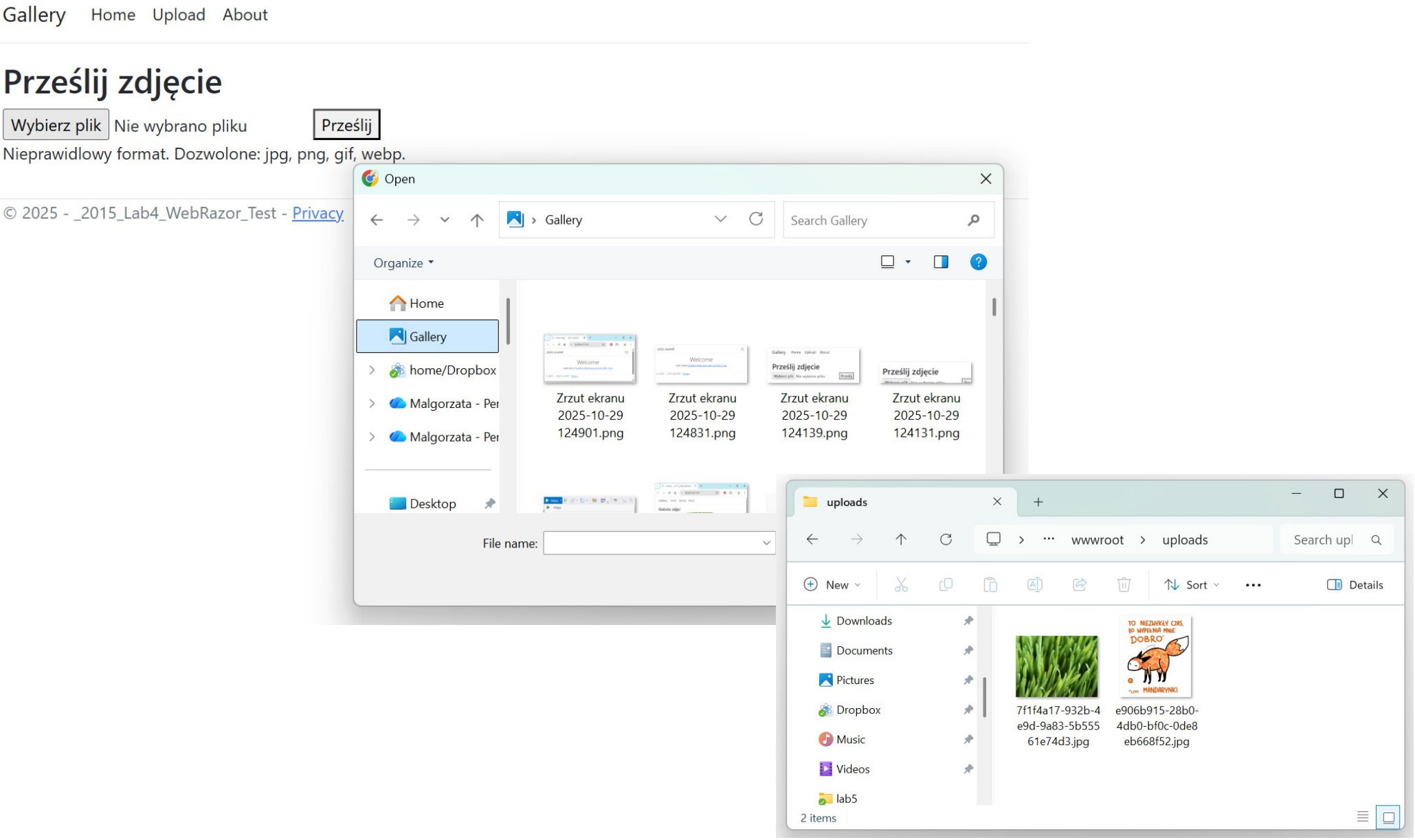
```
ImageFile.Length > 5 * 1024 * 1024
```

      - If not, write appropriate text in Message

Check if it works!

Tasks: Photo Gallery – File upload

- Your stage now, but nothing displayed yet on the page:



Tasks: Photo Gallery – Display Photos

- Change Index.cshtml.cs file:

```
public List<string> Images { get; set; } = new();
public void OnGet()
{
    var uploads = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "uploads");
    if (!Directory.Exists(uploads)) return;

    var files = Directory.GetFiles(uploads)
        .Select(Path.GetFileName)
        .Where(f => !string.IsNullOrEmpty(f))
        .OrderByDescending(f => f)
        .ToList();

    Images = files;
}
```

Tasks: Photo Gallery – Display Photos

- Change Index.cshtml file: add thumbnails of images for all photos

```
<div class="grid">
@foreach(var img in Model.Images)
{
    <div class="thumb">
            <img src="~/uploads/@img" alt="photo"" />
    </div>
}
</div>
```

1. @foreach (var img in Model.Images)
   - Loops through a collection called Images in your Razor page's model.
2.
   - Creates a container (usually styled with CSS) for a single image thumbnail.
3.
   - Generates an <img> tag whose src points to your wwwroot/uploads folder.

Test – photos should be displayed.

Tasks: Photo Gallery – Display Photos

- For now all photos are displayed in their original size - we want to create small tumbnails of the same size. We will use css for that:

  Create gallery.css file in the wwwroot/css/gallery.css.

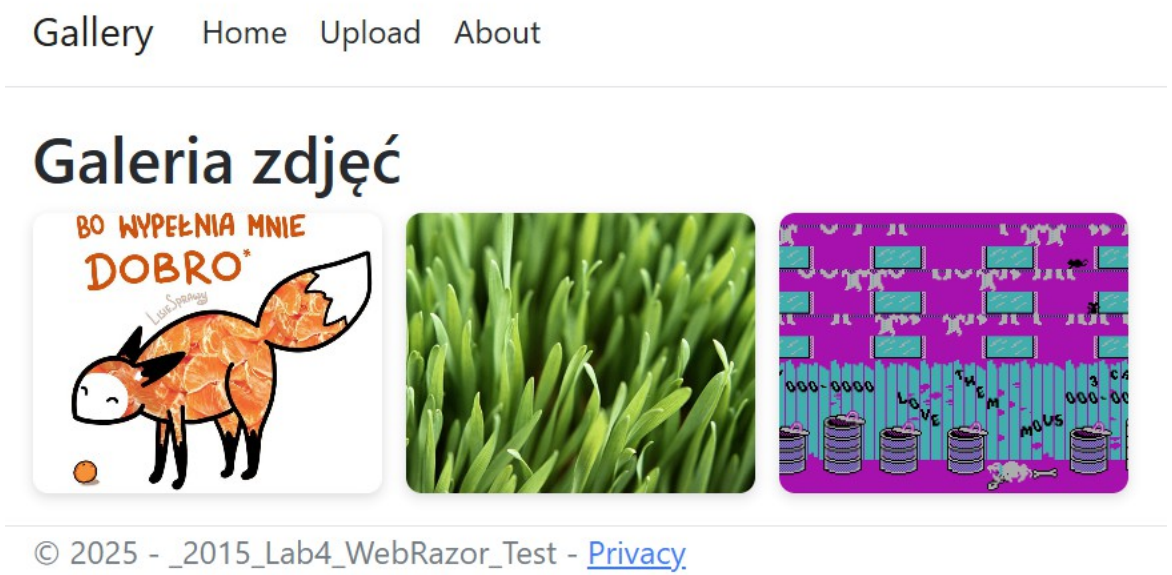
  Copy content of this simple css file:

  http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/gallery.css

- Add this line to the Index.cshtml

```
<link rel="stylesheet" href="~/css/gallery.css" />
```

Try it!



Tasks: Photo Gallery

- **After clicking a photo we would like it to be enlarged.**
  - Create a javascript for this task in wwwroot/js/lightbox.js. You can copy the contents: https://www.if.pw.edu.pl/~majanik/data/Csharp/Files/lightbox.js
  - Add an lightbox control at the bottom of Index.cshtml

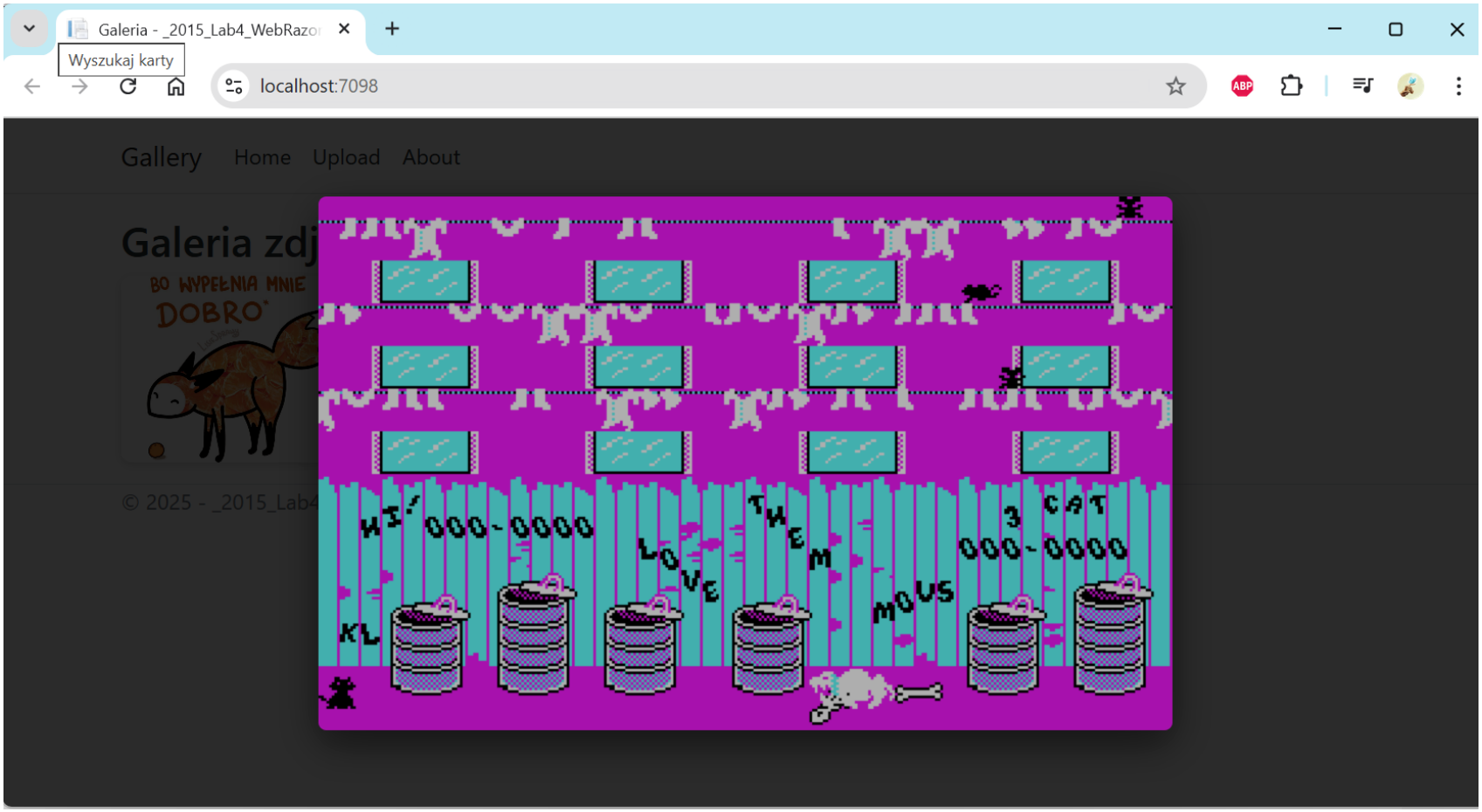
```
<!-- Lightbox -->
<div id="lightbox" onclick="closeLightbox()" style="display:none;">
    <img id="lightbox-img" src="" alt="preview" />
</div>
<script src="~/js/lightbox.js"></script>
```

  - Add attribute to the img controls in foreach:

```
onclick="openLightbox('@Url.Content("~/uploads/" + img)')
```

Tasks: Photo Gallery

- **After clicking a photo we would like it to be enlarged.**
  - `Effect:`



KONIEC

dr hab. inż. Małgorzata Janik
malgorzata.janik@pw.edu.pl