

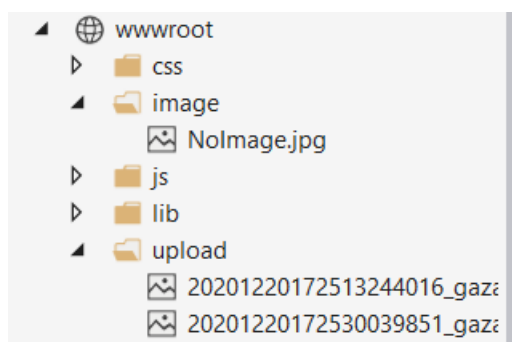
Wstęp

W ramach formularza na stronie WWW użytkownik może podać jako jego element plik (pliki). Aby było to możliwe należy zmienić sposób kodowania żądania POST poprzez dopisanie atrybutu `enctype`. Domyślnie ma on wartość „application/x-www-form-urlencoded”, dla plików najlepiej użyć „multi/form-data”:

```
<form enctype="multipart/form-data" asp-action="Create" >
```

W ASP .net istnieje interfejs przydatny do odbierania w ten sposób wysłanych plików - `IFormFile`. Posiada kilka metod i właściwości, z czego co najmniej dwie można użyć po otrzymaniu pliku w formularzu: właściwości `Filename` (nazwa pliku wysłanego przez użytkownika) oraz metoda `CopyTo(Stream target)` do przekopiowania binarnego pliku do wybranego strumienia (najczęściej pliku na serwerze WWW). Podczas tworzenia nazwy pliku na serwerze WWW należy wziąć pod uwagę możliwość, że nazwy plików użytkownika mogą się powtarzać, więc najczęściej dodaje się do nich (lub wręcz zastępuje) losowym ciągiem (np. za pomocą metody `Guid.NewGuid()`) lub stemplem czasu (`DateTime.Now.ToString(<formatowanie>)`).

Folder, w którym należy przechowywać pliki powinien być w ramach folderu `wwwroot` projektu. W poniższym przypadku `image` to folder na obrazy stałe, natomiast `upload` – na obrazy użytkownika.



W celu poprawnego zapisania ścieżki dostępu do danej kartoteki warto wstrzyknąć w konstruktor kontrolera parametr typu `IWebHostEnvironment` celem pobrania go pomocą właściwości `WebRootPath`:

```
string uploadFolder = Path.Combine(_hostingEnvironment.WebRootPath, "upload");
```

W widoku z formularzem, w którym chcemy pozwolić na dołączanie pliku należy stworzyć odpowiednią kontrolkę.

Jeśli dodatkowo chcemy, aby użytkownik miał podgląd jaki obrazek wybrał należy dopisać skrypt w języku JavaScript. Przykład kontrolki oraz miejsca na podgląd znajduje się poniżej (zakładając model, w którym jest właściwość `FormFile` o typie `IFormFile`):

```

<div class="form-group row">
  <label asp-for="FormFile" class="col-form-label"></label>
  <div class="col-sm-10">
    <input asp-for="FormFile" class="form-control custom-file-input" onchange="previewFile(event)" />
    <label class="custom-file-label">choose a file...</label>
  </div>
  <img id="preview" />
</div>

```

Kod w zielonych ramkach potrzebny jest dla skryptu w JavaScriptcie. Najlepiej dodać go w odpowiedniej sekcji jak poniżej:

```

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}

    <script>
        var previewFile = function (event) {
            var reader = new FileReader();
            reader.onload = function () {
                var output = document.getElementById('preview');
                output.src = reader.result;
            };
            reader.readAsDataURL(event.target.files[0]);
        };
    </script>
}

```

Lista zadań

Do realizacji listy należy użyć **EntityFramework**. Zademonstrować działanie aplikacji ASP po wykonaniu wszystkich poniższych kroków.

1. Stworzyć klasę **Category**, która posiada tylko identyfikator i nazwę. Stworzyć klasę dla pamiętania danych o towarze (**Article**) w sklepie. Powinien posiadać co najmniej nazwę cenę, obrazek i kategorię (jako klucz obcy). Jeden artykuł musi należeć do jednej kategorii, każda kategoria może mieć dowolną liczbę towarów. Obrazek może być dodany przez użytkownika, ale nie musi (w bazie danych będzie wartość NULL).
2. Stworzyć graficzny interfejs użytkownika do manipulowania danymi operacje CRUD dla kolekcji **kategorii** i **towarów**. Obrazy podane przez użytkownika podczas tworzenia instancji towaru pamiętać w wybranym folderze **wwwroot**. Dla uproszczenia można założyć, że obrazek można ustawić tylko podczas tworzenia towaru (bez możliwości zmiany podczas akcji **Edit**).
3. Akcja **Index** powinna pokazywać **miniatury** obrazków, natomiast **Edit**, **Details** i **Delete** obraz w **oryginalnych wymiarach**.
 - a. Wersja obowiązkowa: w **Index** widok klasyczny tabelaryczny
 - b. Wersja dla chętnych: z wykorzystaniem **Bootstrapa** stworzenie widoku kafelkowego (3-4 kafelki w liniach).
4. Ponieważ użytkownik może nie podać obrazka, użyć jakiegoś jednego **obrazka zastępczego**.
5. Operacja **Delete** dla towaru powinna **usuwać** plik z obrazem z **folderu**. Jak powinna się wpływać na towary operacja **Delete** dla kategorii?
6. Stworzyć kontroler **ShopController** do pokazania towarów na podstawie wybranej kategorii: użytkownik wybiera (menu z lewej strony lub lista rozwijalna)





kategorię, a po wybraniu pojawia się lista towarów z **danej kategorii** (z wszystkimi informacjami, w tym obrazkami). Użytkownik nie może wykonać żadnej operacji modyfikacji danych (w przyszłości będzie mógł tylko dodać towar do koszyka)

Przykład możliwego widoku akcji Index dla towarów:

WebAppLab12 [Categories](#) [Articles](#) [Shop](#)

Index

[Create New](#)

Name	Price	Category	Photo	
Barbie Doll	120	Toys		Edit Details Delete
Hammer	122	Tools		Edit Details Delete
Gazania	12	Flowers		Edit Details Delete
Gazania 2	13	Flowers		Edit Details Delete

Przykład widoku tworzenia towaru (przed i po wybraniu pliku obrazka):

Create

Article

Name

Price

Select category

Photo choose a file...

Browse

Create

[Back to List](#)

Create

Article

Name

Price

Select category

Photo: choose a file...

Browse



Create

[Back to List](#)

Data I: Spotkanie 11 (100 punktów)

Data II: Spotkanie 12 (80 punktów)

Data III: Spotkanie 13 (50 punktów)