

# Wstęp do tworzenia aplikacji WWW w .NET Core Razor - formularze

*Programowanie aplikacji WWW w technologii .NET, 2021/2022  
Przygotowała: I. Kartowicz-Stolarska*

## CEL

1. Wykonanie szablonu poniższej strony WWW.

FizzBuzzWeb   Home   Privacy

---

## Welcome User

Twój szczęśliwy numer

Zatwierdź

Fizz

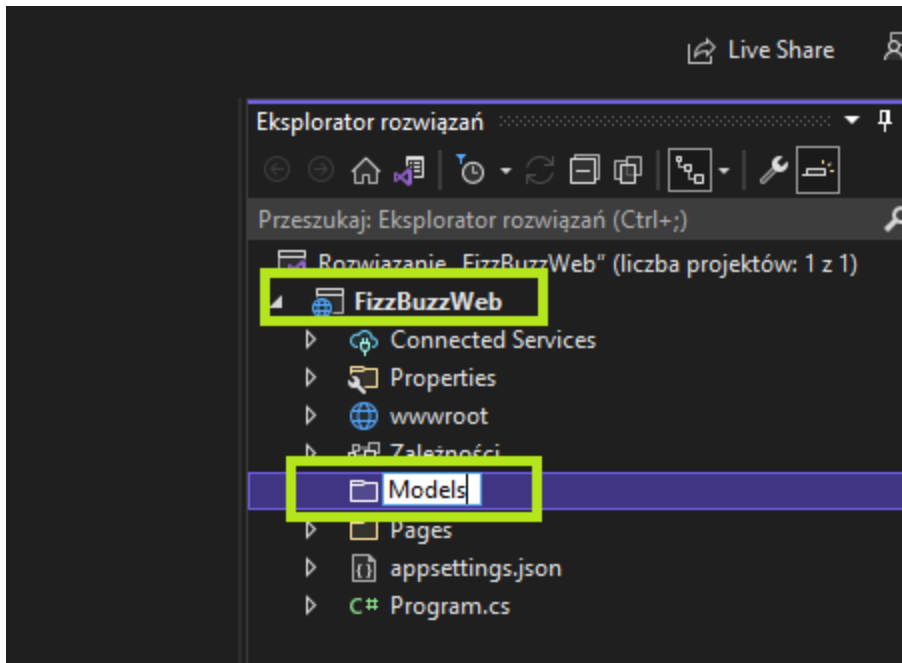
2. Podstawowa obsługa formularzy i RazorTagHelpers.

## WYMAGANIA

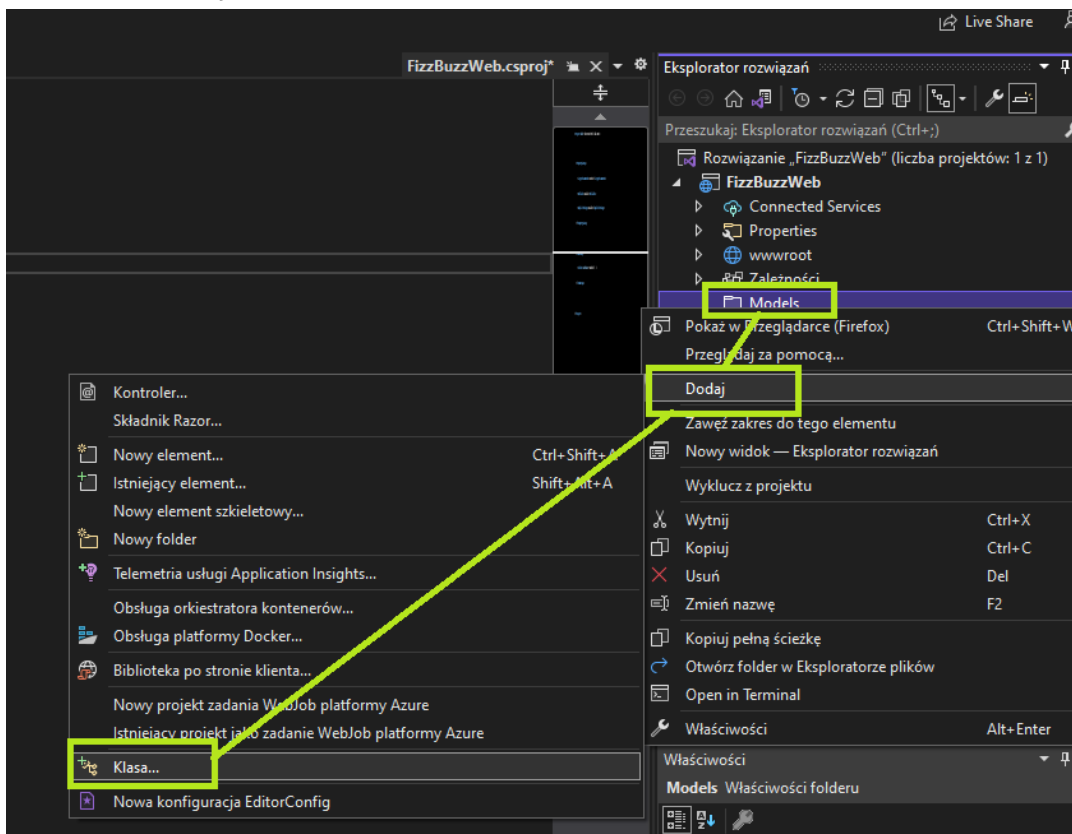
1. Visual Studio 2022
2. .NET Core 6

## TUTORIAL

1. Pobierz projekt FizzBuzzWeb z poprzednich zajęć.
2. Do projektu dodaj katalog Models. W Eksplorator rozwiązań kliknij w nazwę projektu np. FizzBuzzWeb. Z menu kontekstowego wybierz Dodaj -> Nowy folder. W celu zachowania konwencji nowy folder nazwij Models.



3. W katalogu Models utwórz nową klasę FizzBuzz.cs (prawy myszy na katalogu Models -> Dodaj -> Klasa).



4. Klasę FizzBuzz uzupełnij o właściwości zgodnie z przykładem:

```
public class FizzBuzz
{
    public int Number {get; set;}
}
```

5. W pliku *Pages/Index.cshtml/Index.cshtml.cs* przygotuj dane:
  - a. utwórz właściwość *FizzBuzz* typu *FizzBuzz* (UWAGA: dodaj odpowiednie odwołanie do przestrzeni nazw).

```
Index.cshtml.cs
FizzBuzzWeb
1 using FizzBuzzWeb.Models;
2 using Microsoft.AspNetCore.Mvc;
3 using Microsoft.AspNetCore.Mvc.RazorPages;
4
5 namespace FizzBuzzWeb.Pages
6 {
7     public class IndexModel : PageModel
8     {
9         private readonly ILogger<IndexModel> _logger;
10        public FizzBuzz FizzBuzz { get; set; }
11        public IndexModel(ILogger<IndexModel> logger)
12        {
13            _logger = logger;
14        }
15    }
16 }
```

- b. utwórz właściwość *Name* typu *string*

```
namespace FizzBuzzWeb.Pages
{
    public class IndexModel : PageModel
    {
        private readonly ILogger<IndexModel> _logger;
        public FizzBuzz FizzBuzz { get; set; }
        public string Name { get; set; }
    }
}
```

6. Zmień szablon *Pages/Index.cshtml* w taki sposób, aby wyświetlał zmienną *Name*:
7. W pliku *Pages/Index.cshtml/Pages/Index.cshtml.cs* uzupełnij metodę *onGet()* o poniższy kod:

```
<h1 class="display-4">Welcome @Model.Name</h1>
```

```
if (string.IsNullOrEmpty(Name)) {
    Name = "User";
}
```

```
}
```

```
public void OnGet()
{
    if (string.IsNullOrEmpty(Name))
    {
        Name = "User";
    }
}
```

Przetestuj działanie adresów URL z parametrem `?Name=abc` np. <https://localhost:44389/?Name=abc> i bez tego parametru w adresie URL. Czy jest różnica?

8. W pliku `Pages/Index.cshtml/Pages/Index.cshtml.cs` parametr `Name` uzupełnij o opcję:

```
[BindProperty(SupportsGet = true)]
```

```
namespace FizzBuzzWeb.Pages
{
    Odwolań: 8
    public class IndexModel : PageModel
    {
        Odwolań: 0
        private readonly ILogger<IndexModel> _logger;
        public FizzBuzz FizzBuzz { get; set; }
        Odwolań: 3
        [BindProperty(SupportsGet = true)]
        public string Name { get; set; }
    }
}
```

Przetestuj działanie adresów URL z parametrem `?Name=abc` i bez tego parametru.

9. Przygotuj formularz umożliwiający wczytanie danych. Przejdź do pliku `Pages/Index.cshtml` i zawartość strony zamień na:

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Fizz Buzz";
}
<div class="text-center">
    <h1 class="display-4">Welcome @Model.Name</h1>
    <form method="post">
        <label for="Number">Numer:</label>
        <input name="Number" /><br />
        <button type="submit">Zatwierdź</button>
    </form>
</div>
```

10. Przetestuj działanie aplikacji. Sprawdź jak wygląda kod źródłowy w przeglądarce.
11. Następnie zmień kod formularza na formularz z RazorTagHelpers (plik *Pages/Index.cshtml*):

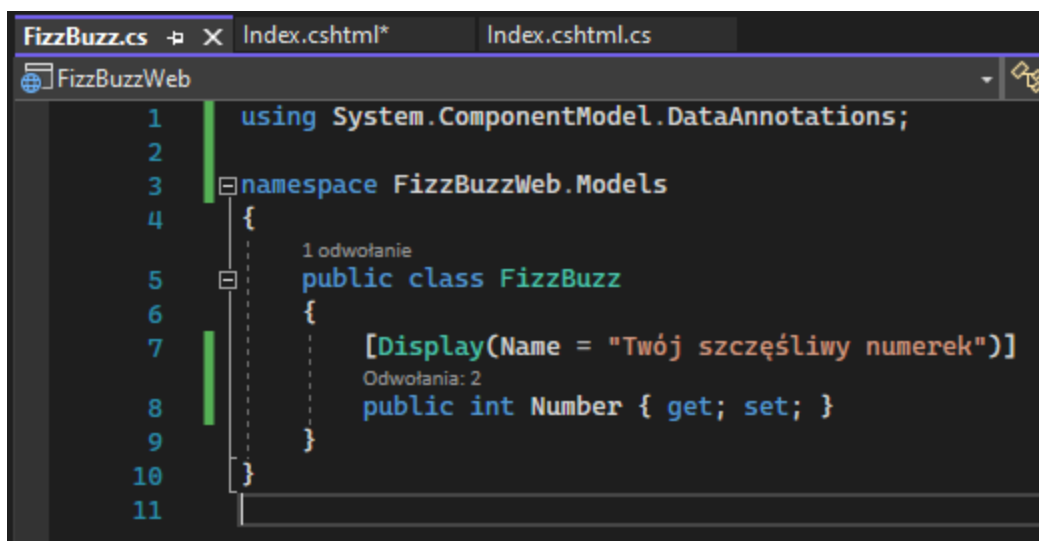
```
<form method="post">
    <label asp-for="FizzBuzz.Number"></label>
    <input asp-for="FizzBuzz.Number" /><br />
    <button type="submit">Zatwierdź</button>
</form>
```

12. W modelu *Models/FizzBuzz.cs* dodaj etykiety do pól np.

```
[Display(Name = "Twój szczęśliwy numer")]
public int Number { get; set; }
```

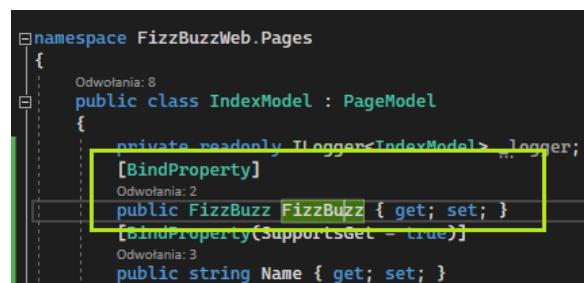
Uwaga: Wymagana jest przestrzeń nazw:

```
using System.ComponentModel.DataAnnotations;
```



13. W pliku *Pages/Index.cshtml/Index.cshtml.cs* dodaj obsługę metody POST.
- a. do atrybutu FizzBuzz w *Index.cshtml/Index.cshtml.cs* dodaj opcję:

```
[BindProperty]
```



b. W pliku *Index.cshtml/Index.cshtml.cs* dodaj metodę *onPost*:

```
public IActionResult OnPost()
{
    if (!ModelState.IsValid)
    {
        return Page();
    }
    return RedirectToPage("./Privacy");
}
```

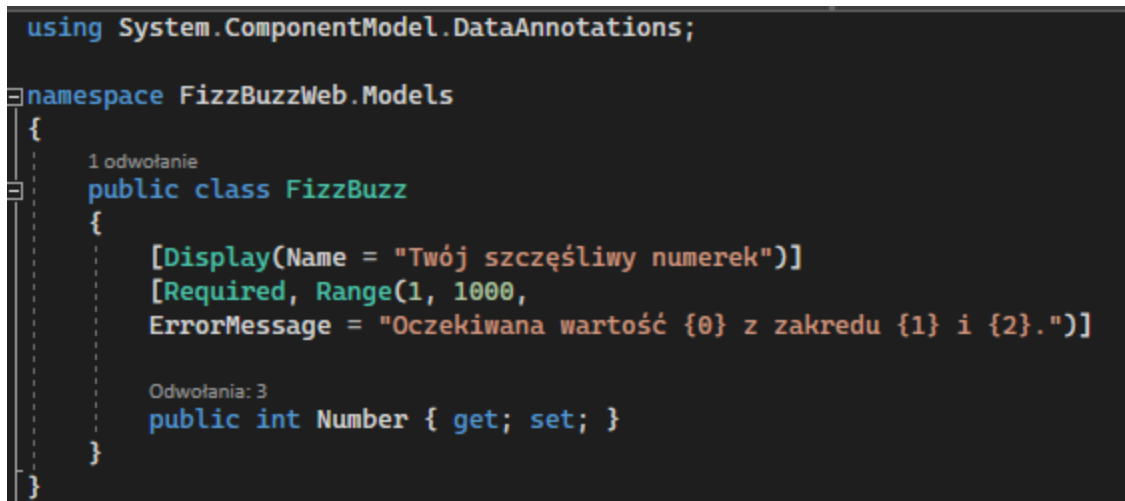
14. Włącz wyświetlanie walidacji w szablonie aplikacji ( w pliku *Pages/Index.cshtml*) np.

```
<span asp-validation-for="FizzBuzz.Number" class="text-danger"></span>
```

15. Dodaj walidację formularza danych po stronie klasy.

Do atrybutu *Number* klasy *FizzBuzz* dodaj różne walidatory (*DataAnnotation*) np.

```
[Required], [Range(1, 1000, ErrorMessage = "Oczekiwana wartość {0} z zakresu {1} i {2}.")], [Required(ErrorMessage="Pole jest obowiązkowe")]
```



```
using System.ComponentModel.DataAnnotations;

namespace FizzBuzzWeb.Models
{
    1 odwołanie
    public class FizzBuzz
    {
        [Display(Name = "Twój szczęśliwy numer")]
        [Required, Range(1, 1000,
            ErrorMessage = "Oczekiwana wartość {0} z zakresu {1} i {2}.")]

        Odwołania: 3
        public int Number { get; set; }
    }
}
```

Przetestuj działanie aplikacji:

- z liczbą z zakresu 1-1000,
- liczbą spoza podanego zakresu
- z pustą wartością. Jaki błąd zwraca formularz, gdy nie podasz żadnej wartości?

16. Zmień deklarację atrybutu *Number* na:

```
public int? Number { get; set; }
```

```

namespace FizzBuzzWeb.Models
{
    1 odwołanie
    public class FizzBuzz
    {
        [Display(Name = "Twój szczęśliwy numererek")]
        [Required, Range(1, 1000,
        ErrorMessage = "Oczekiwana wartość {0} z zakresu {1} i {2}.")]

        2 odwołanie
        public int? Number { get; set; }
    }
}

```

Ponownie przetestuj aplikację. Jaki komunikat pojawia się na stronie, gdy w formularzu przesyłana jest pusta wartość?

17. Rozbuduj utworzoną aplikację w taki sposób, aby aplikacja sprawdzała, czy podana przez użytkownika liczba jest podzielna przez 3, 5, przez 3 i 5. W ramach odpowiedzi na wysłaną liczbę, pod formularzem powinien pojawić się komunikat:

- “Fizz”, jeśli podana liczba jest podzielna przez 3
- “Buzz”, jeśli podana liczba jest podzielna przez 5
- “FizzBuzz”, jeśli liczba jest podzielna przez 3 i 5
- “Liczba: <podana przez użytkownika liczba> nie spełnia kryteriów FizzBuzz” w pozostałych przypadkach

Dozwolone są jedynie liczby całkowite. Aplikacja powinna wyświetlać błąd w przypadku podania nieprawidłowych danych.

Spróbuj ostylować formularz według załączonego przykładu:

FizzBuzzWeb Home Privacy

# Welcome User

Twój szczęśliwy numererek

Zatwierdź

Fizz

Skorzystaj z dokumentacji Bootstrap:

<https://getbootstrap.com/docs/4.0/components/alerts/>

<https://getbootstrap.com/docs/4.0/components/forms/#form-grid>