Tworzenie aplikacji webowej w technologii ASP.NET Razor Pages

Podczas ćwiczenia utworzymy prostą aplikację webową w języku C#. Potrzebne będą:

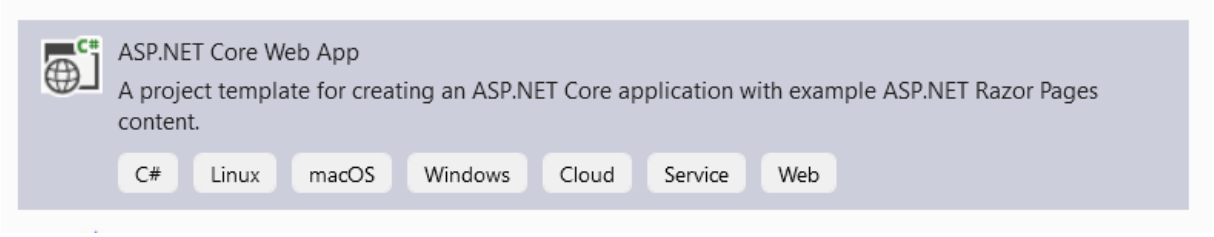
- Visual Studio 2022
- .NET SDK 6.0

W ramach zajęć użyte zostaną następujące technologie:

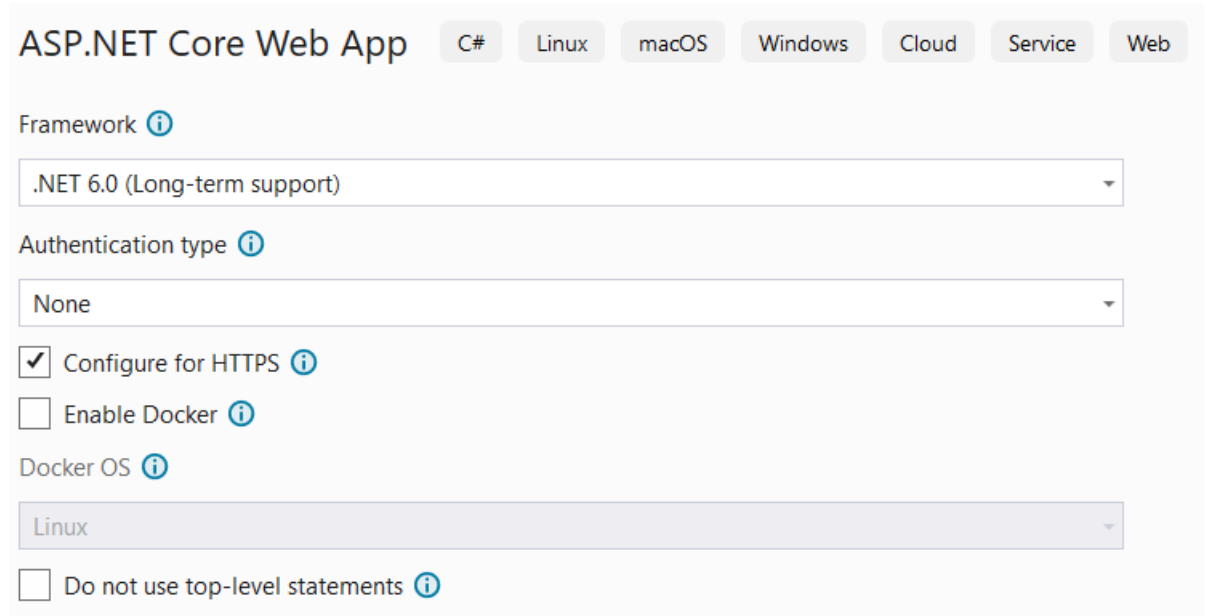
- Język C#
- .NET 6.0
- ASP.NET Core
- Razor Pages
- Entity Framework Core
- SQLite

Przebieg ćwiczenia:

1. Po uruchomieniu Visual Studio tworzymy nowy projekt. Jako szablon należy wybrać:



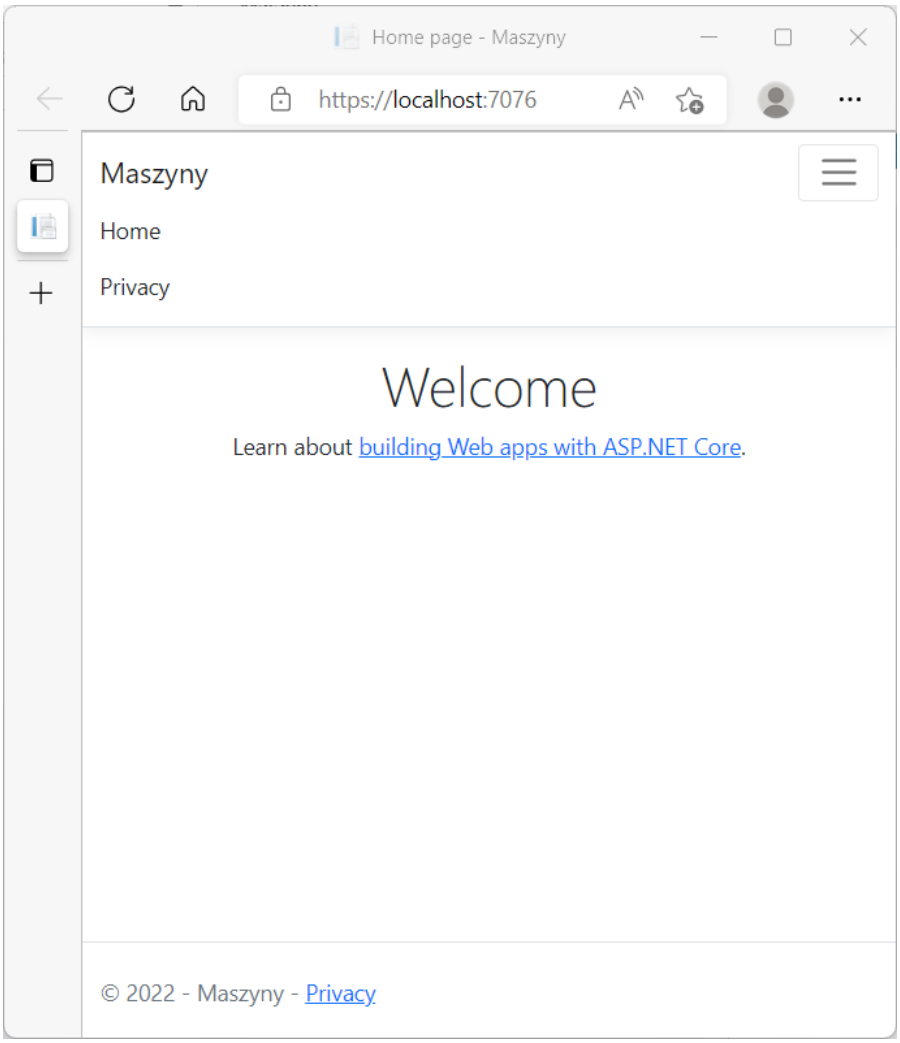
Wpisujemy nazwę aplikacji (tutaj: *Maszyny*), zaś opcje zostawiamy na domyślnych ustawieniach, jak niżej:



2. Po utworzeniu projektu uruchamiamy aplikację w trybie debugowania. Powinna się uruchomić także przeglądarka z adresem, pod którym aplikacja jest dostępna. Aplikacja domyślnie składa się z 2 stron:

- startowej (Home)
- polityki prywatności (Privacy).

Warto zauważyć, że witryna jest responsywna, jej układ dopasowuje się do rozmiarów ekranu. Bierze się to stąd, że jej styl oparto o bibliotekę Bootstrap.



3. Nasza aplikacja będzie umożliwiała zarządzanie maszynami. Jako pierwszy element utworzymy stronę do dodawania nowej maszyny. W tym celu po zatrzymaniu aplikacji wybieramy w drzewie projektu folder *Pages* i z menu kontekstowego wybieramy opcję *Add->Razor page*. W następnym oknie wybieramy opcję:

W kolejnym ekranie wpisujemy nazwę strony: *Create.cshtml*.

4. Nowa strona ma dwa oblicza: wygląd (w pliku Razor z rozszerzenie *.cshtml*) oraz logika (w pliku *.cs*). Najpierw dodajmy do strony kod formularza HTML do wprowadzania danych maszyny.

```
@page
@model Maszyny.Pages.CreateModel
@{
```

```
}

<form method="post">
    <fieldset>
        <label asp-for="Name">Machine name</label>
        <input asp-for="Name" placeholder="name" />
        <span asp-validation-for="Name" class="text-danger"></span>
        <br />
        <label asp-for="LaunchDate">Launch date</label>
        <input asp-for="LaunchDate" />
        <span asp-validation-for="LaunchDate" class="text-danger"></span>
        <br />
        <label asp-for="Operational">Operational</label>
        <input asp-for="Operational" />
        <span asp-validation-for="Operational" class="text-danger"></span>

        <br />

        <input type="submit" value="Create" id="submitButton" />

    </fieldset>
</form>
```

Dane z formularza będą przesyłane metodą POST w protokole HTTP. Oprócz elementów *<input>* w formularzy umieszczono etykiety *<label>* oraz miejsce na komunikaty błędów walidacji **.

5. Dane przesyłane z formularza będą przetwarzane za pomocą kodu C# z pliku *Create.cshtml.cs*. Jest tam zdefiniowana klasa *CreateModel* z domyślną funkcją *OnGet*. Ze względu na użycie metody POST nie skorzystamy z niej, lecz dodamy nową funkcję *OnPost*:

```
public IActionResult OnPost()
{
    if (string.IsNullOrWhiteSpace(Name))
    {
        ModelState.AddModelError("Name", "Enter machine name !");
        return Page();
    }

    return Page();
}
```

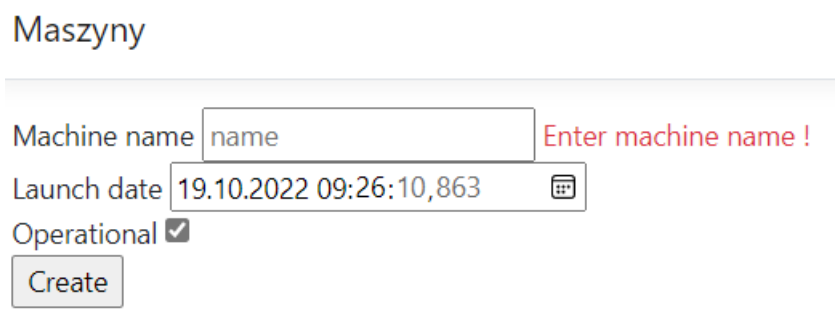
Zawiera ona prostą walidację nazwy maszyny, która zawsze musi być podana.

6. W celu przekazania danych z formularza do kodu C#, w klasie CreateModel należy utworzyć odpowiednie właściwości:

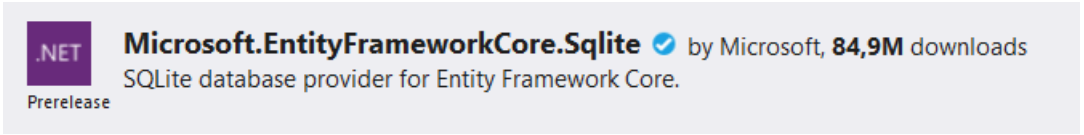
```
[BindProperty]
public string? Name { get; set; } = String.Empty;

[BindProperty]
public DateTime LaunchDate { get; set; } = DateTime.Now;

[BindProperty]
public bool Operational { get; set; } = true;
```

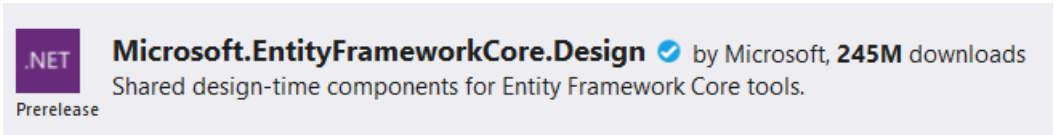
Mają one przypisane wartości domyślne, a atrybut *[BindProperty]* umożliwia ich powiązanie z odpowiednimi polami formularza (atrybut *asp-for*).

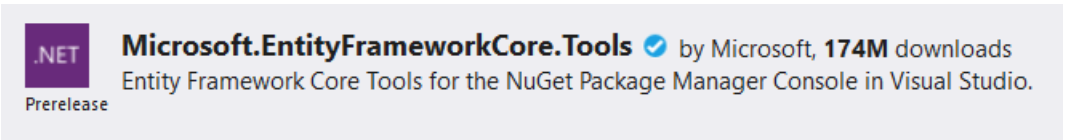
7. Uruchamiamy aplikację i w przeglądarce do adresu URL dopisujemy ręcznie */create*. Sprawdzamy, czy walidacja działa:



8. Teraz uzupełnimy aplikację, aby dane były na stałe zapisywane w bazie danych. W tym celu do projektu należy dodać odpowiednie pakiety. Z menu uruchamiamy menedżera pakietów NuGet, a w jego oknie wyszukujemy:

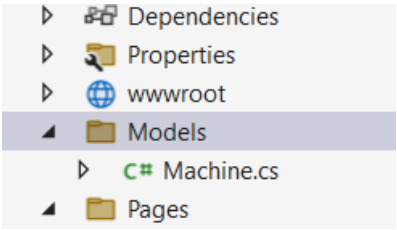






Instalujemy wersję stabilną *6.xx* tych pakietów.

9. Obsługa bazy danych wymaga utworzenia klasy modelowej. Tworzymy w projekcie nowy folder *Models*, a w nim nowy plik klasy C# o nazwie *Machine.cs*.



10. W pliku *Machine.cs* dodajemy definicję klasy *Machine*:

```
public class Machine
{
    public int Id { get; set; }
    public string? Name { get; set; } = String.Empty;
    public DateTime LaunchDate { get; set; } = DateTime.Now;
    public bool Operational { get; set; } = true;
}
```

11. Włączamy obsługę bazy danych w naszej aplikacji dodając w folderze *Models* nowy plik klasy kontkestu bazy danych *AppDbContext.cs* z następującą zawartością:

```
using Microsoft.EntityFrameworkCore;

namespace Maszyny.Models
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) :
base(options)
        {
        }

        public DbSet<Machine> Machines { get; set; }
    }
}
```

12. W pliku Program.cs dodajemy u góry:

```
using Microsoft.EntityFrameworkCore;
```

oraz usługę obsługującą bazę danych:

```
builder.Services.AddDbContext<AppDbContext>(o => o.UseSqlite($"Data
Source=app.db"));
```

13. W konsoli menedżera pakietów wpisujemy polecenia:

```
Add-Migration Init
```

a następnie:

```
Update-Database
```

Powinien wówczas utworzyć się plik bazy danych SQLite o nazwie *app.db*.

14. Teraz możemy już zapisywać dane maszyny do bazy. Uzupełnimy kod *Create.cshtml.cs* o odpowiednie instrukcje. Najpierw dodajmy konstruktor klasy, aby przekazywać mu kontekst bazy danych za pomocą tzw. wstrzykiwania zależności (*dependency injection*):

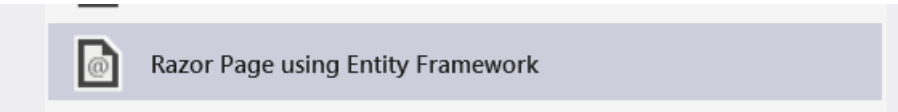
```
        public readonly AppDbContext db;

        public CreateModel(AppDbContext db)
        {
            this.db = db;
        }
```

Teraz dopiszmy do funkcji *OnPost* instrukcje zapisujące dane:

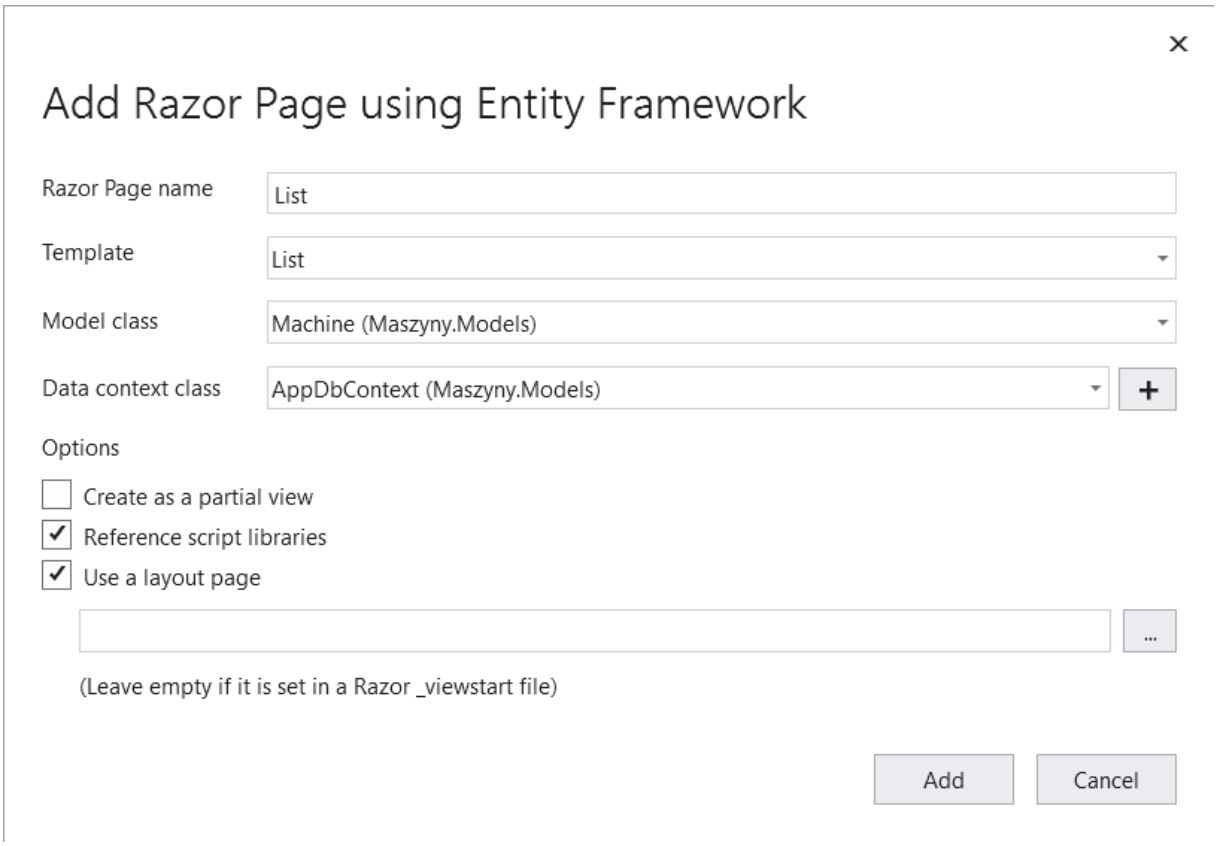
```
            db.Machines.Add(new Machine { Name = this.Name, LaunchDate =
this.LaunchDate, Operational = this.Operational });
            db.SaveChanges();

            return RedirectToPage("List");
```

Ostatnia linia przekierowuje nas do listy maszyn, którą utworzymy w dalszej części.

15. Do stworzenia listy maszyn użyjemy kreatora. Tym razem przy dodawaniu strony wybieramy:



W kolejnym oknie wybieramy opcje jak niżej:



16. Po uruchomieniu aplikacji możemy już dodawać maszyny i wyświetlać ich listę:

Zadania dodatkowe

1. Dodać walidację sprawdzającą, czy pierwszy znak nazwy maszyny jest literą.
2. Dodać obsługę usuwania i edycji maszyny.