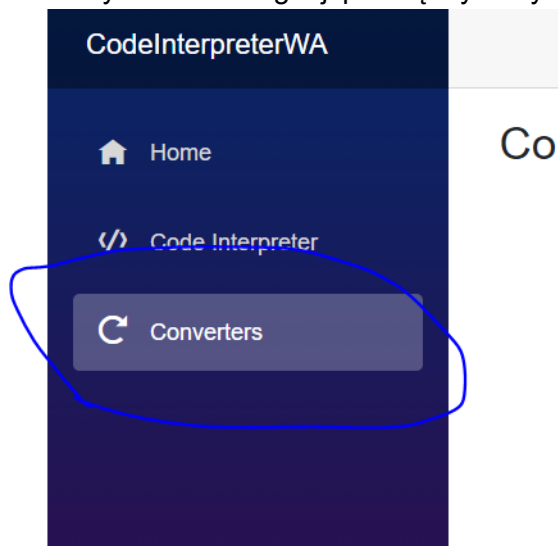


1. Proszę utworzyć nowy folder **Pages/Converters** oraz dodać nowy komponent o nazwie **Converters**, następnie proszę zdefiniować routing dla nowego komponentu. **@page "/converters"**. Komponent ten będzie swoistym kontenerem dla pozostałych komponentów (child components)
2. Teraz aby umożliwić szybki i prosty dostęp do nowego komponentu proszę dodać go do nawigacji. Jako styl ikonki nawigacji proszę wykorzystać te klasy: *oi oi-reload*



3. Proszę dodać nowy komponent o nazwie **TimeConverter**. Zadaniem tego komponentu będzie konwertowanie godzin na sekundy.
4. W tym kroku proszę dodać nowo utworzony komponent **TimeConverter** do głównego komponentu przeliczników **Converters**.

```
1  @page "/converters"
2
3  <h3>Converters</h3>
4
5  <TimeConverter></TimeConverter>
6
7  @code {
8
9  }
10
```

5. Proszę zmodyfikować kod komponentu **TimeConverter** aby wyglądał następująco:

```
<div>
  <div class="col-md-4">
    <div class="form-group">
      <label>Hours</label>
      <input type="number" class="form-control" placeholder="Hours..." @bind="_hours"/>
    </div>
    <button class="btn btn-primary" @onclick="ConvertTime">
      <span class="oi oi-clock mr-2" aria-hidden="true"></span>Convert
    </button>
  </div>

  @if(_seconds != null)
  {
    <div class="row blockquote mt-5">
      <span>In @_hours hours we have @_seconds seconds ! </span>
    </div>
  }
</div>

@code {
  int? _seconds = null;

  int? _hours { get; set; }

  private void ConvertTime()
  {
    _seconds = _hours * 3600;
  }
}
```

W tym fragmencie kodu pierwszy raz obsługujemy zdarzenie kliknięcia przycisku. Jak widać jest to zrealizowane za pomocą składni **@onclick="nazwaMetody"**. Po kliknięciu przycisku *Convert* godziny są konwertowane na sekundy w metodzie *ConvertTime*.

Warto też zwrócić uwagę na instrukcję warunkową **if** w kodzie html. W naszym przypadku pozwala na ukrycie fragmentu kodu html dopóki zmienna *\_seconds* nie będzie różna od null. Inaczej mówiąc dopóki w zmiennej nie będziemy mieli obliczonej wartości sekund.

6. Proszę sprawdzić czy przelicznik działa poprawnie

CodeInterpreterWA

Home

Code Interpreter

Converters

Converters

Converters

Hours

Convert

In 2 hours we have 7200 seconds !

7. Proszę dodać nowy komponent o nazwie **DistanceConverter**
8. W tym kroku należy dodać nowo utworzony komponent (DistanceConverter) do głównego komponentu **Converters**
9. Proszę zmodyfikować kod komponentu **DistanceConverter** w następujący sposób:

```
<div>
  <div class="col-md-4">
    <div class="form-group">
      <label>Meters</label>
      <input type="number" class="form-control" placeholder="Meters..." @bind="Meters" @bind:event="oninput" />
    </div>
  </div>
</div>

@if (_inches != null)
{
  <div class="row blockquote mt-5">
    <span>In @Meters m equal to @_inches inches !</span>
  </div>
}
</div>

@code {
  double? _inches = null;
  double? _meters = null;

  private double? Meters
  {
    get => _meters;
    set
    {
      _meters = value;
      _inches = ConvertDistance();
    }
  }

  private double? ConvertDistance()
  {
    return Meters / 0.0254d;
  }
}
```

Jak widzicie w tym przypadku nie korzystamy z przycisku, wpisane metry są zamieniane na cale od razu po wpisaniu wartości. Proszę przebudować aplikację i sprawdzić poprawność działania przelicznika.

10. Aktualnie w głównym komponencie **Converters** wyświetlamy dwa przeliczniki naraz. Teraz dodamy selektor który pozwoli nam wybrać z którego przelicznika chcemy korzystać. W pierwszym kroku proszę dodać model który wykorzystamy w naszym selektorze. Proszę dodać klasę w ciele @code jak poniżej:

```
@code {  
  
    class ConverterItem  
    {  
        public ConverterItem(int id, string nameOfConverter)  
        {  
            Id = id;  
            NameOfConverter = nameOfConverter;  
        }  
  
        public int Id { get; set; }  
        public string NameOfConverter { get; set; }  
    }  
}
```

11. Teraz mając już model możemy utworzyć listę dostępnych przeliczników. Proszę zdefiniować listę typu ConverterItem i dodać do niej dwa elementy definiujące dwa przeliczniki.  
Przelicznik nr 1: Id = 1, NameOfConverter= "Time converter"  
Przelicznik nr 2: Id = 2, NameOfConverter= "Distance converter"

```
readonly List<ConverterItem> _availableConverters = new List<ConverterItem>  
{  
    new ConverterItem (1, "Time converter"),  
    new ConverterItem (2, "Distance converter")  
};
```


12. Proszę utworzyć zmienną typu int o nazwie `_selectedConverterId`. Zmienną tą będzie przetrzymywała identyfikator przelicznika wybranego przez użytkownika.

```
int _selectedConverterId;
```

13. Mając już model oraz zmienną w której będziemy przechowywać id wybranego przelicznika, możemy stworzyć nasz selektor. W głównym komponencie przeliczników proszę usunąć wcześniej dodane przeliczniki a w ich miejsce proszę dodać selektor wykorzystując wcześniej zdefiniowaną listę oraz

zmienną `selectedConverterId`

```
<TimeConverter></TimeConverter>  
<DistanceConverter></DistanceConverter>
```



```
<select @bind="_selectedConverterId" class="form-control col-md-4 mb-4" >  
  @foreach (var converter in _availableConverters)  
  {  
    <option value="@converter.Id">@converter.NameOfConverter</option>  
  }  
</select>
```

14. Teraz wykorzystując zmienną `_selectedConverterId` oraz instrukcję `Switch` proszę dodać kod który w zależności od wyboru użytkownika będzie wyświetlał odpowiedni przelicznik. W przypadku gdy użytkownik nic nie wybrał proszę wyświetlić tekst *"Please select converter"*.

```
@switch (_selectedConverterId)  
{  
  case 1:  
    <TimeConverter></TimeConverter>  
    break;  
  case 2:  
    <DistanceConverter></DistanceConverter>  
    break;  
  default:  
    <span>Please select converter...</span>  
    break;  
}
```

15. Proszę uruchomić aplikację i sprawdzić czy aplikacja działa poprawnie.

### Converters

Time converter

Hours

Hours ....

⌂

Convert

### Converters

Distance Converter

Meters

Meters ....

16. Przelicznik odległości korzysta z stałej wartości współczynnika (**0.0254**), dodajmy możliwość przekazania tej wartości jako wymagany parametr komponentu DistanceConverter. W tym celu proszę dodać nowe property typu double z atrybutem [Parameter], a następnie podmienić w metodzie ConvertDistance stałą wartość 0.0254 na nasz nowy parametr.

```
[Parameter]
public double DistanceFactor { get; set; }

private double? ConvertDistance()
{
    return Meters / DistanceFactor;
}
```

17. Teraz musimy przekazać wartość współczynnika w miejscu wywołania komponentu DistanceConverter

```
case 2:
    <DistanceConverter DistanceFactor="0.0254"></DistanceConverter>
```

Taki zabieg pozwala na tworzenie bardziej dynamicznych komponentów które możemy dostosować do aktualnych potrzeb. Proszę uruchomić aplikację i sprawdzić czy w dalszym ciągu otrzymujemy poprawne wartości. 1 metr => ~39.37 cali