

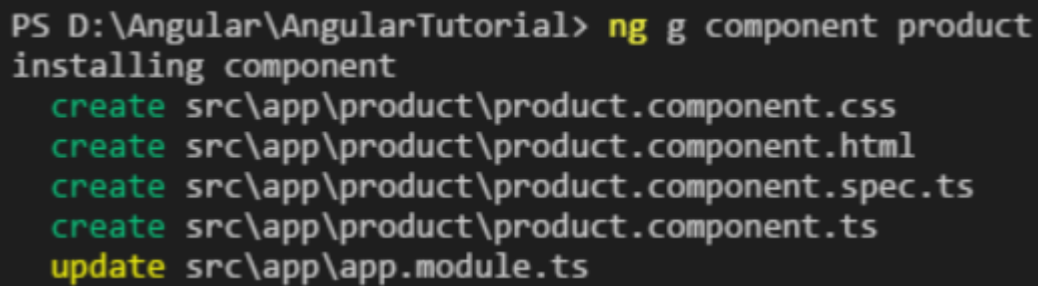
# Angular Tutorial

## Lekcja 2

### 1. Tworzenie nowych komponentów

Podczas poprzedniej lekcji zainstalowaliśmy angular cli służące m.in. do generacji nowych komponentów. Przyjrzyjmy się efektom działania następującej komendy:

```
ng g component product
```



```
PS D:\Angular\AngularTutorial> ng g component product
installing component
  create src\app\product\product.component.css
  create src\app\product\product.component.html
  create src\app\product\product.component.spec.ts
  create src\app\product\product.component.ts
  update src\app\app.module.ts
```

Jak widzimy utworzony został nowy folder zawierający komponent wraz z widokiem, stylami oraz plikiem testowym. Zajrzyjmy co zmieniło się w pliku app.module.ts:

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/http';

import { AppComponent } from './app.component';
import { ProductComponent } from './product/product.component';

@NgModule({
  declarations: [
    AppComponent,
    ProductComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Plik został automatycznie uzupełniony o definicję nowo powstałego komponentu, o czym musimy pamiętać tworząc go ręcznie. Warto o tym pamiętać, ponieważ brak deklaracji powoduje często długie poszukiwania gdzie leży błąd.

Komendę generate można rozszerzyć o dodatkowe parametry, co może okazać się przydatne przy generacji w odpowiednich folderach lub gdy mamy więcej niż 1 plik modułowy. Operacja wtedy wygląda tak:

Ng g component path nazwa\_komponentu (np. ng g component /products products-list)

Gdy dodatkowo określimy plik modułowy do aktualizacji:

Ng g component path nazwa\_komponentu -m plik\_modułowy

## 2. Funkcje, przypisywanie zdarzeń i modele

W poprzedniej lekcji mowa była o możliwości przypisania funkcji do zdarzeń. Tym razem przyjrzymy się temu ciut dokładniej. Zaczniemy od sposobu tworzenia funkcji w komponentach. Jedną zdążyliśmy już poznać – funkcję `ngOnInit` wykonującą się przy tworzeniu komponentu w przeglądarce. Zasada tworzenia funkcji jest prosta: `nazwa(argumenty) : typ zwrotny`. Reszta jest identyczna jak w Javie (no prawie 😊). Przykład:

```
increment(index : number) : number {  
  return index + 1;  
}
```

Funkcja ta przyjmuje jako argument liczbę (przyzwyczajcie się do notacji określającej typ argumentu) oraz zwraca tę liczbę zwiększoną o 1. Proste jak budowa cepa! W przypadku braku typu zwrótnego funkcja jest typem `void`.

Funkcję można przypisać do odpowiedniego elementu html (np. `Button`) i będzie ona wykonywana za każdym razem gdy użytkownik wywoła na nim akcję (np. naciśnięcie guzika). Sposób już znamy:

```
<button (click)="add()">Add</button>
```

Angular podobnie jak Java pozwala nam na definicję własnych klas i interfejsów, które mogą nam służyć jako modele. Wystarczy utworzyć plik `ts` i w identyczny sposób jak w Javie stworzyć klasę / interfejs (pamiętając tylko słowie kluczowym `export`!). Wygląda to mniej więcej tak:

```
export class Product {  
  name : string;  
  description : string;  
}
```

Zazwyczaj stosuje się konwencję w której pola są albo `public`, albo bez znacznika dostępu (druga opcja częściej spotykana). Taką klasę można stosować jako zmienną w aplikacji.

### 3. Znaczniki \*ngIf oraz \*ngFor

Na koniec teorii poznamy jeszcze dwie bardzo ważne komendy służące do pracy z plikami html. \*ngIf pozwala nam ukryć pewne elementy gdy nie spełniają one założonego warunku.

Założmy, że mamy w kontrolerze zdefiniowaną tablicę:

```
products : Product[];
```

I wyświetlamy ją w tabeli kodem html. Gdy tablica jest pusta, storna jednakże wygląda brzydko i pusto mając tylko nazwy kolumn, postanawiamy więc ukryć pustą tabelę. Z pomocą przyjdzie nam właśnie \*ngIf

```
<div *ngIf="products && products.length > 0">  
    //logika  
</div>
```

Można to traktować jako if'a wsadzonego do HTML 😊

Podobnie sprawa ma się z \*ngFor, które generuje nam kolejne rekordy na podstawie tablicy:

```
<div class="col-item" *ngFor="let product of products; let i = index" [attr.data-index]="i">
```

Tu dzieje się kilka ciekawszych rzeczy, zacznijmy od początku:

let product of products -> Dla każdego produktu w tabeli products.

Słowo let ma specjalne znaczenie w angularze, oznacza ono tworzenie zmiennej lokalnej, dostępnej tylko w zadanym obszarze (scope).

let i = index" [attr.data-index]="i" -> opcjonalne rozszerzenie ngFor, zapamiętujące w jego ciele index obecnego element w tabeli.

To by było na tyle jeśli chodzi o teorię!

## 4. Zadanie do wykonania:

W zadanym kodzie źródłowym stworzyć formularz dodający produkty do odpowiedniej tablicy w komponencie. Tablice produktów należy wyświetlić, dodając możliwość usunięcia produktu poprzez naciśnięcie przycisku przy wyświetlanym produkcie. Oczywiście nie wyświetlać pustej tablicy. W folderze Products wygenerować za pomocą cli nowy komponent o nazwie product-list i tam zawrzeć formularz dodawania, tablicę oraz wyświetlanie produktów. Całość oczywiście powinna ładnie wyglądać 😊

P.S. Pamiętać o selektorach! Wkrótce poznamy jeszcze inną metodę wyświetlania widoków!