Angular Training

Services

Peter Bouda, hey@peterbouda.eu

ng-lisbon.com

Why Services?

- They are the central instance to process data
- Services are TypeScript classes
- It's a best practice to decorate them as `@Injectable`
- We will inject them to components or other services
- We need to provide them to be a part of the Dependency Injection Tree
- Angular will create a singleton for each service
- Logic that is independant of the view will be implemented in services

Service

```
import { Injectable } from '@angular/core';

@Injectable()
export class PizzaService {
  private toppings = [];

  constructor() { }

  addTopping(topping) {
    this.toppings.push(topping);
  }

  getToppings() {
    return this.toppings;
  }
}
```

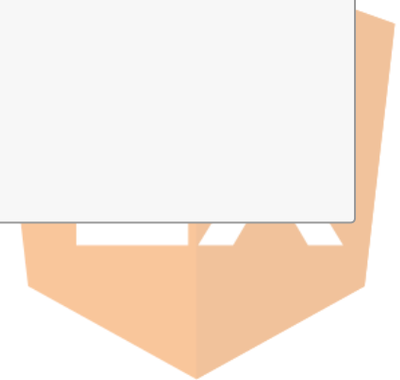
Dependency Injection

- A design pattern to manage dependencies between parts of an app
- Individual parts of an application become more independant of other parts
- Ease testing, as we can mock dependencies and focus our tests on only one part (e.g. one component)

ng-lisbon.com

Dependency Injection

```
class Hamburger {
  private bun: Bun;
  private patty: Patty;
  private toppings: Toppings;
  constructor() {
    this.bun = new Bun('withSesameSeeds');
    this.patty = new Patty('beef');
    this.toppings = new Toppings(['lettuce', 'pickle', 'tomato']);
  }
}
```



Dependency Injection

```
class Hamburger {
  private bun: Bun;
  private patty: Patty;
  private toppings: Toppings;
  constructor(bunType: string, pattyType: string, toppings: string[]) {
    this.bun = new Bun(bunType);
    this.patty = new Patty(pattyType);
    this.toppings = new Toppings(toppings);
  }
}
```

Dependency Injection

```
class Hamburger {
  private bun: Bun;
  private patty: Patty;
  private toppings: Toppings;
  constructor(bun: Bun, patty: Patty, toppings: Toppings) {
    this.bun = bun;
    this.patty = patty;
    this.toppings = toppings;
  }
}
```

In TypeScript:

```
class Hamburger {
  constructor(private bun: Bun, private patty: Patty,
    private toppings: Toppings) {}
}
```

Provide and Inject a Service

```
import { Component, OnInit } from '@angular/core';

import { PizzaService } from './pizza.service';

@Component({
  selector: 'app-adder',
  templateUrl: './adder.component.html',
  styles: [],
  providers: [PizzaService]
})
export class AdderComponent implements OnInit {

  constructor(private pizzaService: PizzaService) { }

  onAddTopping() {
    this.pizzaService.addTopping('Salat');
  }

  ngOnInit() {
  }
}
```

Providers

- Providers make the dependencies available for injection via the Dependency Tree
- The dependencies are then available for all child components
- Providers in `app.module` are available for injection in the whole app
- With this strategy we make the data processing available throughout the app

ng-lisbon.com

Provider in app.module

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { HeaderComponent } from './header.component';
import { AdderComponent } from './adder.component';
import { PizzaService } from './pizza.service';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    AdderComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [PizzaService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Inject a Service in Another Service

```
import { Injectable } from '@angular/core';

import { ToppingService } from './topping.service';

@Injectable()
export class PizzaService {
  private toppings = [];

  constructor(private toppingService: ToppingService) {
    this.toppings.push(toppingService.getToppings());
  }

  addTopping(topping) {
    this.toppings.push(topping);
  }

  getToppings() {
    return this.toppings;
  }
}
```