



Global Knowledge®

Angular + TypeScript

Module 3 - Services

Peter Kassenaar –
info@kassenaar.com

WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA



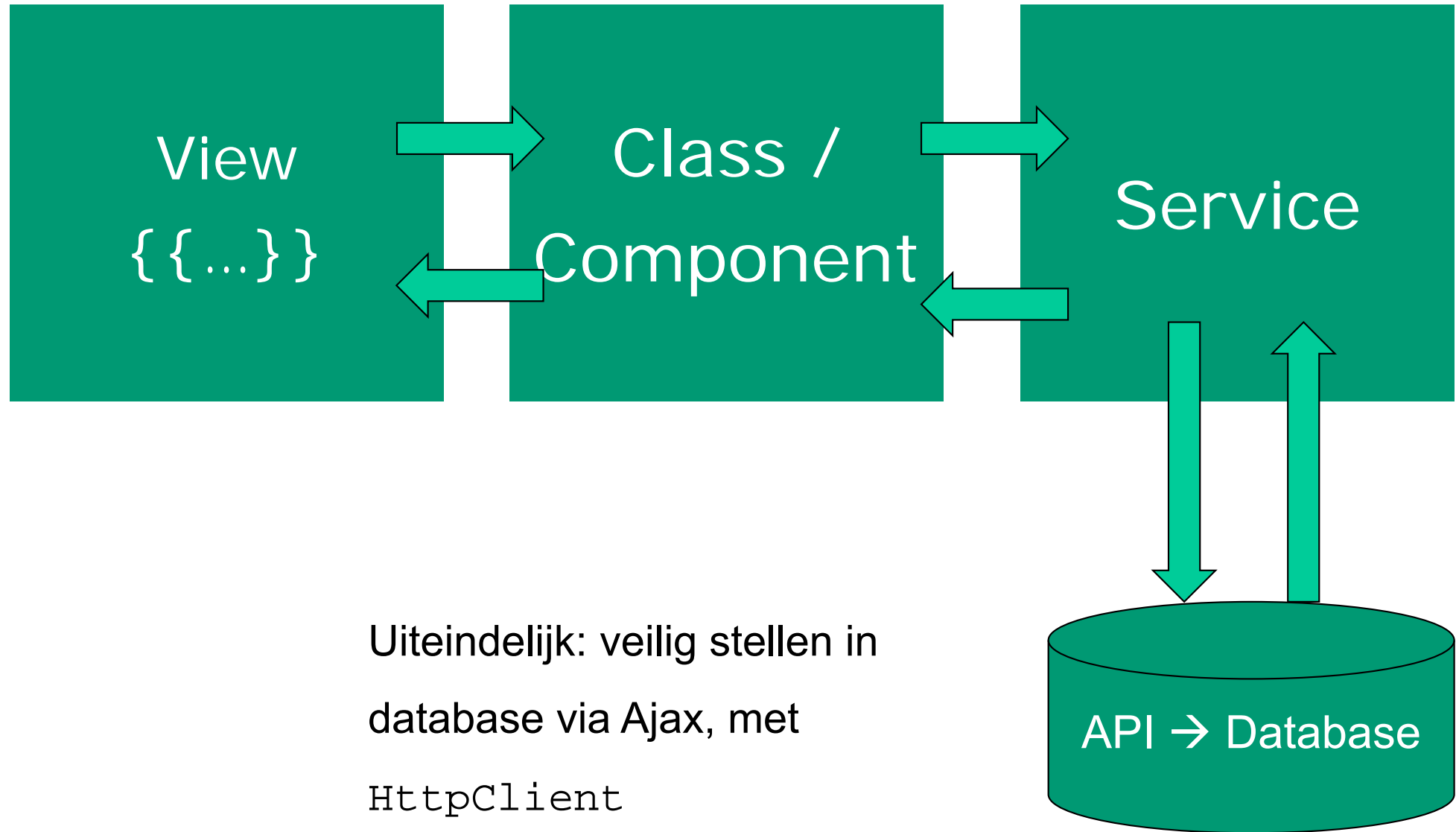
Hoofdstuk 5
p. 121 en verder

Services

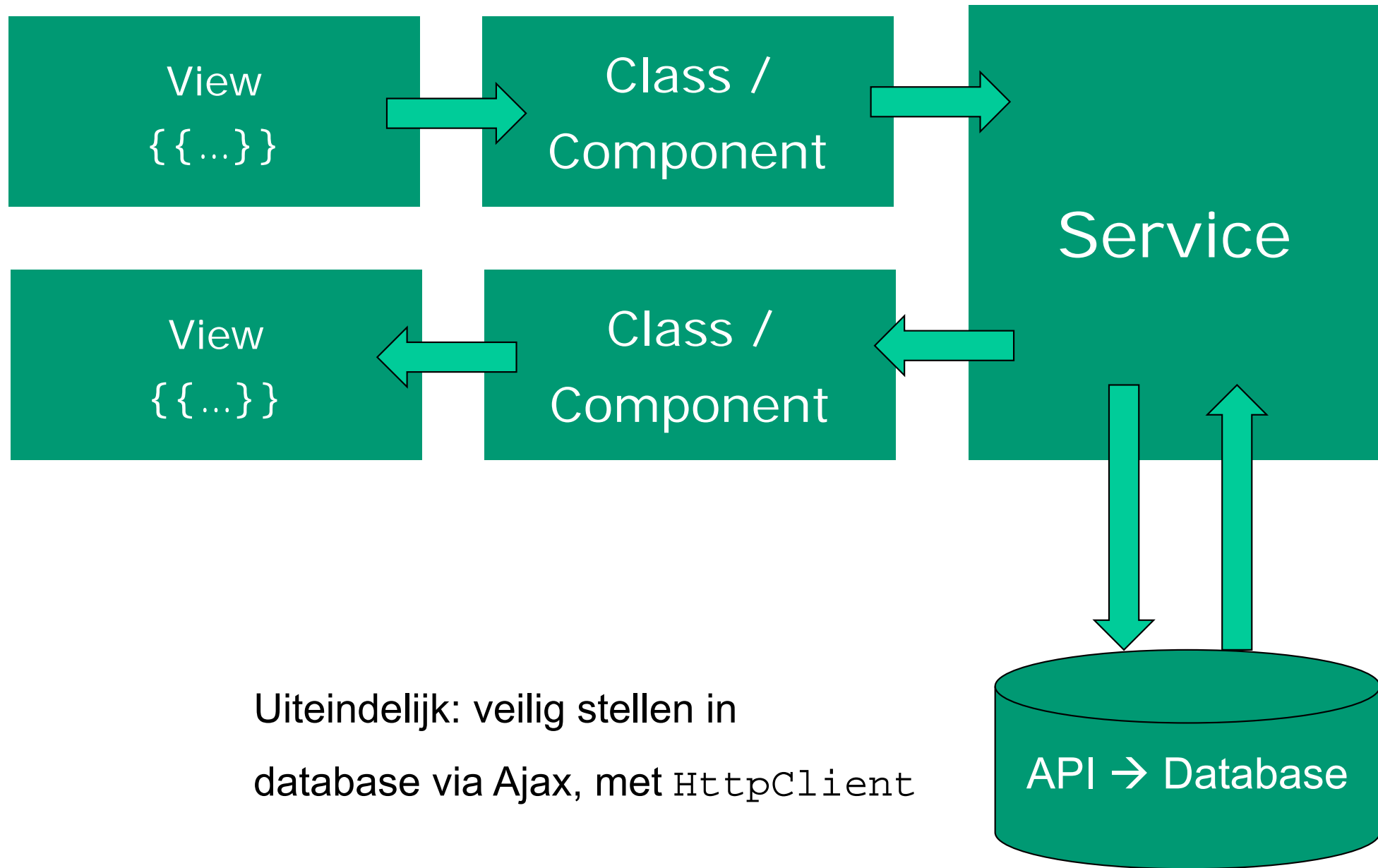
Doel – datafunctionality herbruikbaar maken voor verschillende componenten

- Data retrieval
 - Data caching
 - Data Storage,
 - ...
-
- Angular: één optie
 - `export class myDataService { ... }`

Data flow



Data flow



Services in Angular

Data services in Angular 1:

```
angular.module( 'myApp' )  
  .service(...)  
  .factory(...)  
  .provider(...)
```

Data services in Angular 2+:

```
import {Injectable} from '@angular/core';  
  
@Injectable()  
export class CityService{  
  //...  
}
```

De rol van @Injectable

Why? – Dependency Injection (DI) en metadata!

“TypeScript sees the @Injectable() decorator and emits metadata about our service, metadata that Angular may need to inject other dependencies into this service.”

<https://angular.io/docs/ts/latest/tutorial/toh-pt4.html>

"Our service doesn't have any dependencies at the moment. Add the decorator anyway."

*It is a best practice to apply the
@Injectable() decorator **from the start** both
for consistency and for future-proofing"*



Creating a service

Creating a service in 3 steps

Creating a service

```
ng generate service [name]
```

Stap 1 – service maken (static data)

```
import { Injectable } from '@angular/core';
import { City } from './city.model'

@Injectable()
export class CityService {
  private cities:City[] = [
    new City(1, 'Groningen', 'Groningen'),
    ...
  ];

  // retourner alle cities
  getCities() {
    return this.cities
  }

  // retourner city op basis van ID
  getCity(id:number) {
    return this.cities.find(c => c.id === id);
  }
}
```

Stap 2 – Service consumeren/injecten

```
...
import {CityService} from "../city.service";

@Component({
  selector    : 'hello-world',
  templateUrl: 'app/app.component.html',
})

export class AppComponent implements OnInit {
  // Properties voor de component/class
  currentCity: City;
  cities: City[];
  cityPhoto: string;

  constructor(private cityService: CityService) {

  }

  ngOnInit() {
    this.cities = this.cityService.getCities();
  }

  getCity(city: City) {
    this.currentCity = this.cityService.getCity(city.id);
    this.cityPhoto   = `img/${this.currentCity.name}.jpg`;
    console.log('City opgehaald:', this.currentCity);
  }
}
```

local
variables

Constructor: DI + shorthand
voor nieuwe private variable +
instantiëring!

Detailgegevens voor
city bij (click) event

Instantiation?

- Let op: geen `new()` instantie van de Service!
 - Services zijn Singletons
 - Worden opgehaald uit de Module en/of geïnstantieerd in een `constructor()`

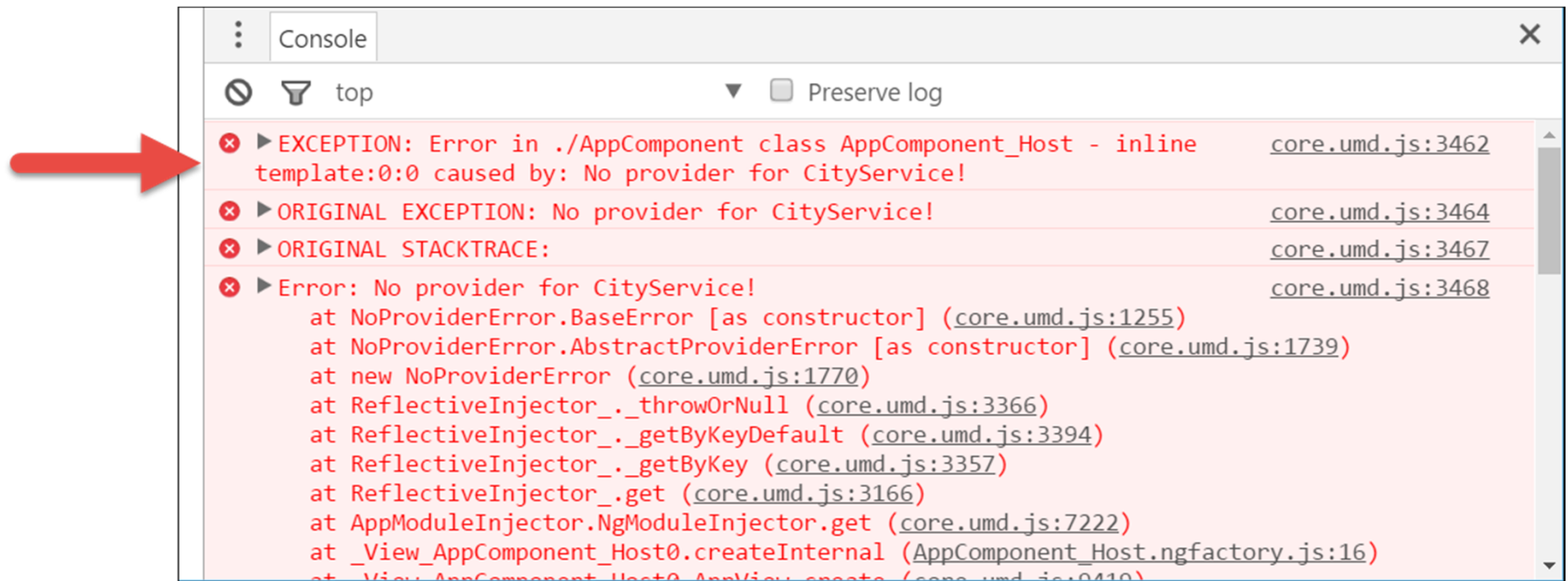
```
constructor(private cityService:CityService) { ... }
```

“The constructor itself does nothing.

*The parameter simultaneously defines a
private cityService property and identifies it
as a CityService injection service.”*

“No provider for CityService”

- Solution: inject in `app.module.ts`



Optie 1: Service injecteren in Module

- Alleen de *referentie* naar CityService is niet voldoende.
- Angular moet de service *injecteren* in de module
- Gebruik de annotatie `providers: [...]`

```
// Module declaration
@NgModule({
  imports      : [BrowserModule],
  declarations: [AppComponent],
  bootstrap    : [AppComponent],
  providers    : [CityService] // DI voor service
})
export class AppModule {
}
```



Array met
Service-
dependencies

Optie 2 : Angular 6+ , gebruik providedIn

- “Tree shakeable providers”
- Niet meer opgeven welke services in een Module worden gebruikt, maar andersom:
- In de service opgeven in welke modules deze wordt gebruikt.

```
@Injectable({  
  providedIn: 'root'  
})  
export class CityService {  
  ...  
}
```

```
@NgModule({  
  imports      : [BrowserModule],  
  declarations: [AppComponent],  
  bootstrap    : [AppComponent],  
  // providers  : [CityService]  
})
```


Singleton?

- Services zijn (in principe) singletons
 - Maar: afhankelijk van de plek waar ze geïnstantieerd worden!
 - Ze zijn een singleton voor de module en alle child components.
 - Module/Site-wide gebruiken? Instantieer service in `app.module.ts` of gebruik `providedIn: 'root'`

Checkpoint

- Elke service in Angular 2 is een `class`
- Service importeren in de component waarin je hem injecteert
- Dependency Injection in constructor van Component
- Vergeet niet: Instantiëren in `ngModule` OF: `providedIn: 'root'` gebruiken
- Functies van de service gebruiken in de componenten
- Voorbeeld: `/200-services-static`
- Oefening 5a)

Workshop....

