



Angular Fundamentals

Module 3 – Services



A CANON COMPANY

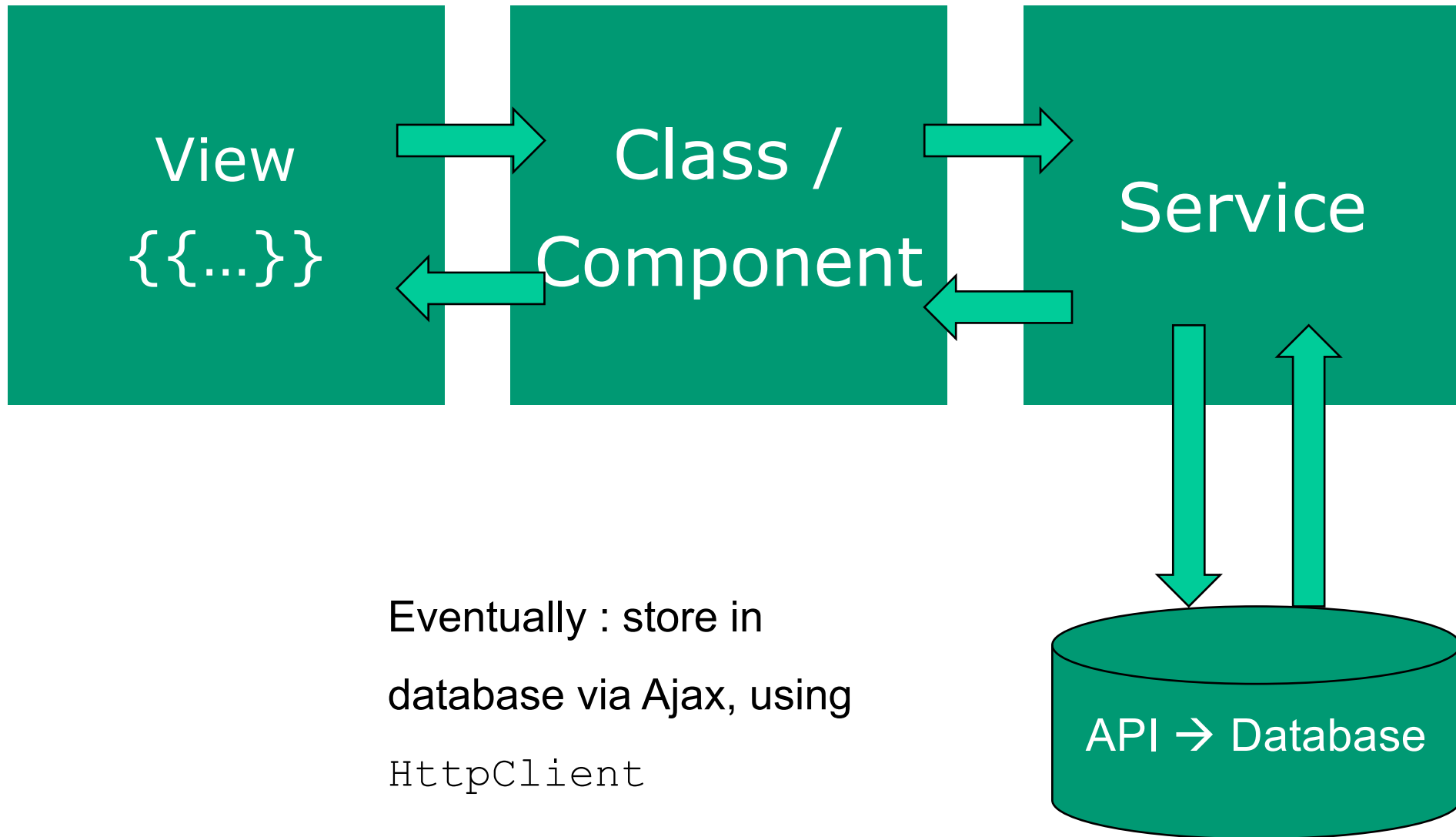


Peter Kassenaar
info@kassenaar.com

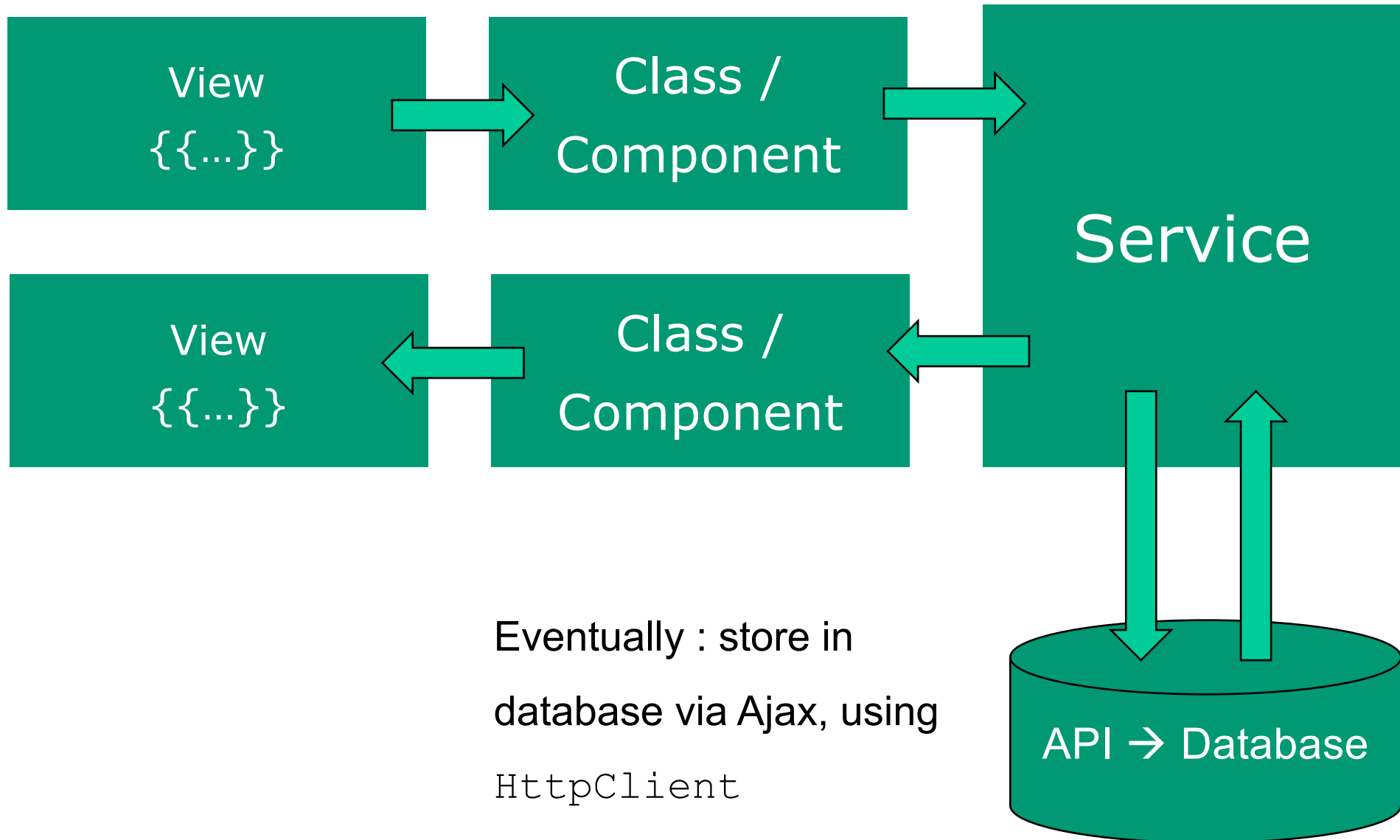
Services

- Goal – *reuse* data functionality over different components
 - Data retrieval
 - Data caching
 - Data Storage,
 - ...
- Angular: one option
 - `export class myDataService { ... }`

Data flow



...and with multiple components



Services in Angular 2

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{  
  //....  
}
```

Make sure to use @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"*

Step 1 – create service (static data)

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

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

  // return all cities
  getCities() {
    return this.cities
  }

  // return city based on id
  getCity(id:number) {
    return this.cities.find(c => c.id === id);
  }
}
```


Step 2 – Inject/consume service

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

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

export class AppComponent implements OnInit {
  // Properties for 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);
    .....
  }
}
```

local
variables

Constructor: shorthand to
instantiate private variable

Details for city on
(click) event

Instantiation?

- Pay attention: no `new()` instance of Service!
 - Services are –mostly- Singletons
 - Are fetched from the Module and/or instantiated in

`constructor()`

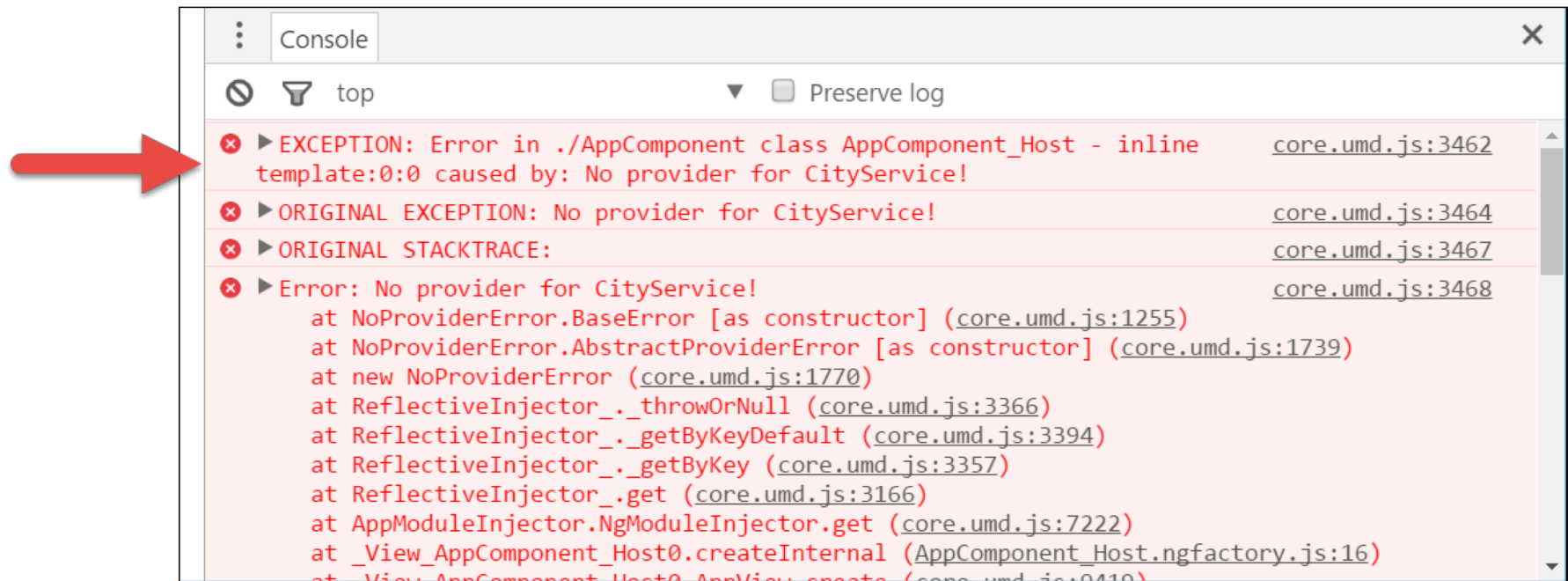
constructor(**private** cityService:CityService) { ... }

"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`



Step 3, option 1 – Inject service in Module

Only an import/reference to CityService is not sufficient.

Angular has to *inject* the service in the module

Use the annotation `providers: [...]`

// Module declaration

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



Array with Service-
dependencies

Step 3, option 2 : Angular 6/7, use providedIn

- “Tree shakeable providers”
- Don’t tell the Module which services to use, the other way around:
- tell the service in which module it is used

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

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

<https://blog.angular.io/version-6-of-angular-now-available-cc56b0efa7a4>

Singleton?

- Services are (usually) singletons
 - But: it depends where the service is provided/instantiated!
 - Services are singleton for Component/Module and all child components.
 - Using Module/Site-wide? (recommended)
 - Instantiate service in `app.module.ts`

Checkpoint

- Every service in Angular 2 is a `class`
- Import and inject in the module that uses it
- Import and instantiate in `constructor()` of the class that needs access to the service methods
- Add service to `providers: []` or use `providedIn`
- Exercise 5a) + 5b)
- Example: `\200-services-static`

Exercise....

