

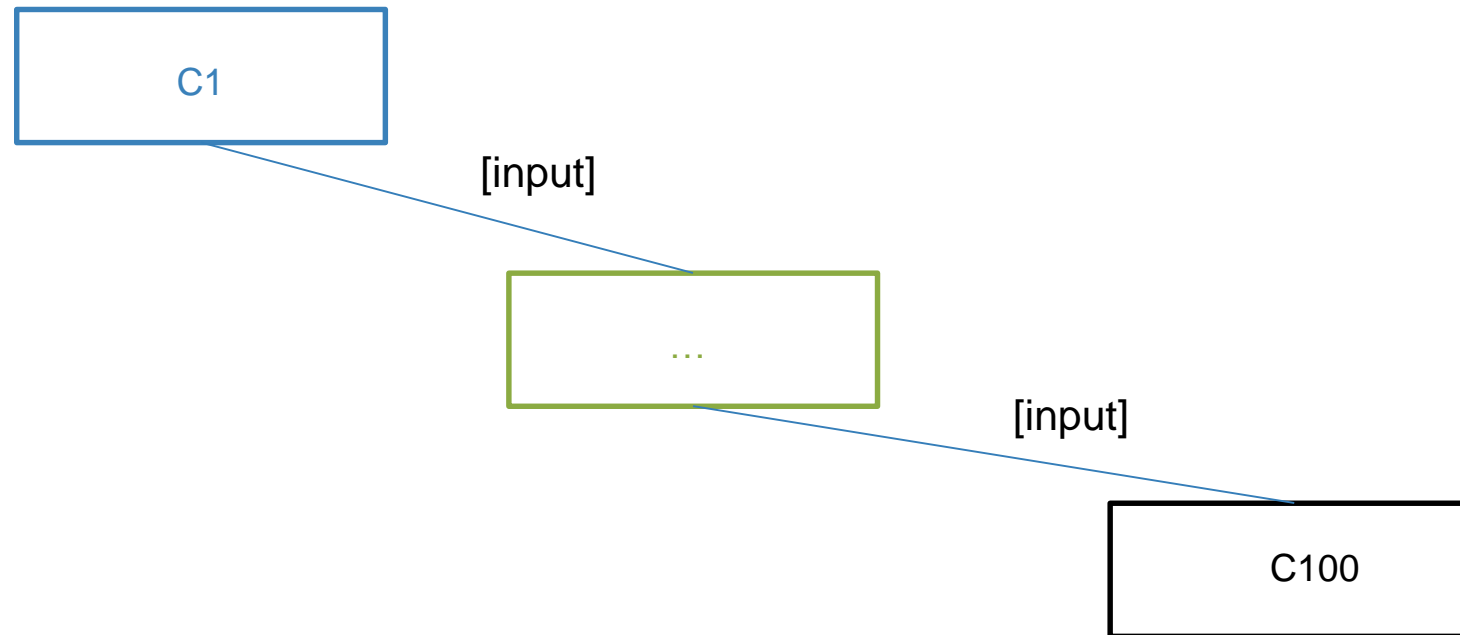


Injection de dépendances

Animé par Mazen Gharbi

Comment faire?

- Vous avez une application Angular complexe, il faut passer une information du component n°1 au component n°100..



Dependency Injection

- ▷ La "dependency injection" était l'une des principales fonctionnalités d'AngularJS et on la retrouve dans Angular.
- ▷ Les avantages :
 - › Facilite la gestion des dépendances
 - › Améliore l'extensibilité de nos applications
 - › Facilite les tests unitaires
 - › Persiste la donnée à travers notre application
 - › Factorisation du code
- ▷ Angular va se charger d'instancier les Dependency Injection et nous fournir les instances

Sans injection de dépendances

```
class UserStore {  
  
    getUser(userId: string): Observable<User> {  
  
        let restApi = new RestApi(new ConnectionBackend(), new RequestOptions({headers: ...}));  
  
        return restApi.users.get(userId);  
    }  
}
```

Avec

```
class UserStore {  
    private _restApiInstance: RestApi = null;  
  
    constructor(restApi: RestApi) {  
        this._restApiInstance = restApi;  
    }  
  
    getUser(userId: string): Observable<User> {  
        return this._restApiInstance.users.get(userId);  
    }  
}
```

Plus concis

```
class UserStore {  
    constructor(private _restApi: RestApi) {}  
  
    getUser(userId: string): Observable<User> {  
        return this._restApi.users.get(userId);  
    }  
}
```

TypeScript



▷ Super, comment créer une dependency injection?


<https://stackblitz.com/edit/angular-services-pokemon>

 *Pensez à déclarer la D.I. dans votre module. Cela est fait automatiquement avec ng generate service « nom »*

Qui se charge de créer les instances?

Création d'une D.I.

pokemon.service.ts

```
@Injectable({  
  providedIn: 'root'  Depuis Angular6, voici la manière préconisée de référencer les services  
  // dans votre application  
})  
export class PokemonsService {  
  private listPokemons: Pokemon[];  
  
  constructor() {  
    this.listPokemons = [  
      new Pokemon('Bulbizarre', Type.Plante, 5),  
      new Pokemon('Mr. Mime', Type.Psy, 17)  
    ];  
  }  
  
  getPokemons(): Pokemon[] {  
    return this.listPokemons;  
  }  
}
```

 Dans un Service, « `ngOnInit` » n'existe pas, la logique d'initialisation doit être ajoutée dans le constructeur !

Appel d'une D.I.

app.component.ts

```
constructor(private pokemonsService: PokemonsService) {  
}  
  
public ngOnInit() {  
    this.pokemons = this.pokemonsService.getPokemons();  
}
```

Injectors

Depuis Angular 8



```
const injector: Injector = Injector.create({ providers: [TestService] });  
  
let instanceA = injector.get(TestService);  
let instanceB = injector.get(TestService);  
  
console.log(instanceA === instanceB); // true
```

Un injecteur est créé par composant

- Pourquoi ?

@Inject

```
export class AppComponent implements OnInit {  
    constructor(@Inject(TestService) test) {}  
}
```

=

```
export class AppComponent implements OnInit {  
    constructor(test: TestService) {}  
}
```

```
@NgModule({
  bootstrap: [ UserNameEditorComponent ],
  declarations: [
    UserNameEditorComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [

    /* Using another class. */
    {
      provide: ApiRest,
      useClass: ApiRestv2
    },

    /* Using a custom instance. */
    {
      deps: [Config]
      provide: ApiRest,
      useFactory: (config: Config) => new ApiRest(config)
    },

    /* Using a custom value. */
    {
      provide: ApiRest,
      useValue: new ApiRest()
    }

  ]
})
export class AppModule {
}
```

Questions