UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Angular DI

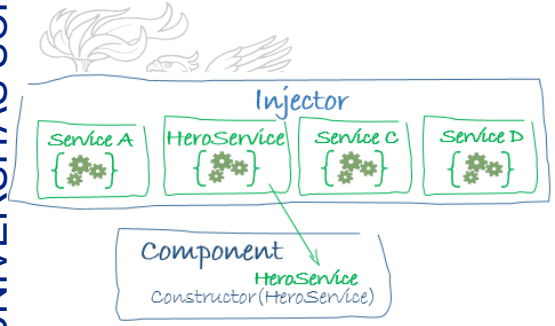
Dr. Bilicki Vilmos
Szoftverfejlesztés Tanszék

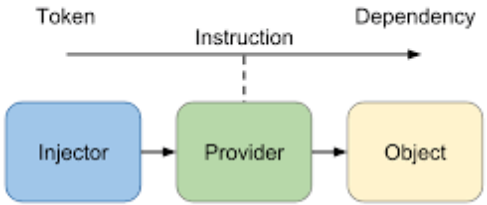
Az előző előadás összefoglalója

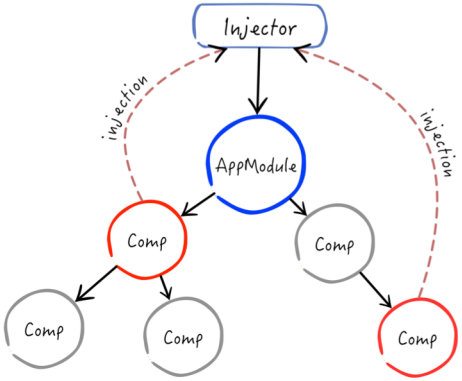
- Bevezetés: NgModules
- JS Modulok vs. NgModule-ok
- Gyakran használt NgModule-ok
- Képesség modulok
- A képesség modulok típusai
- Belépő komponensek
- Szolgáltatások
- Singleton Szolgáltatások
- Képesség modulok laza betöltése
- NgModulok megosztása
- NgModule API

Áttekintés

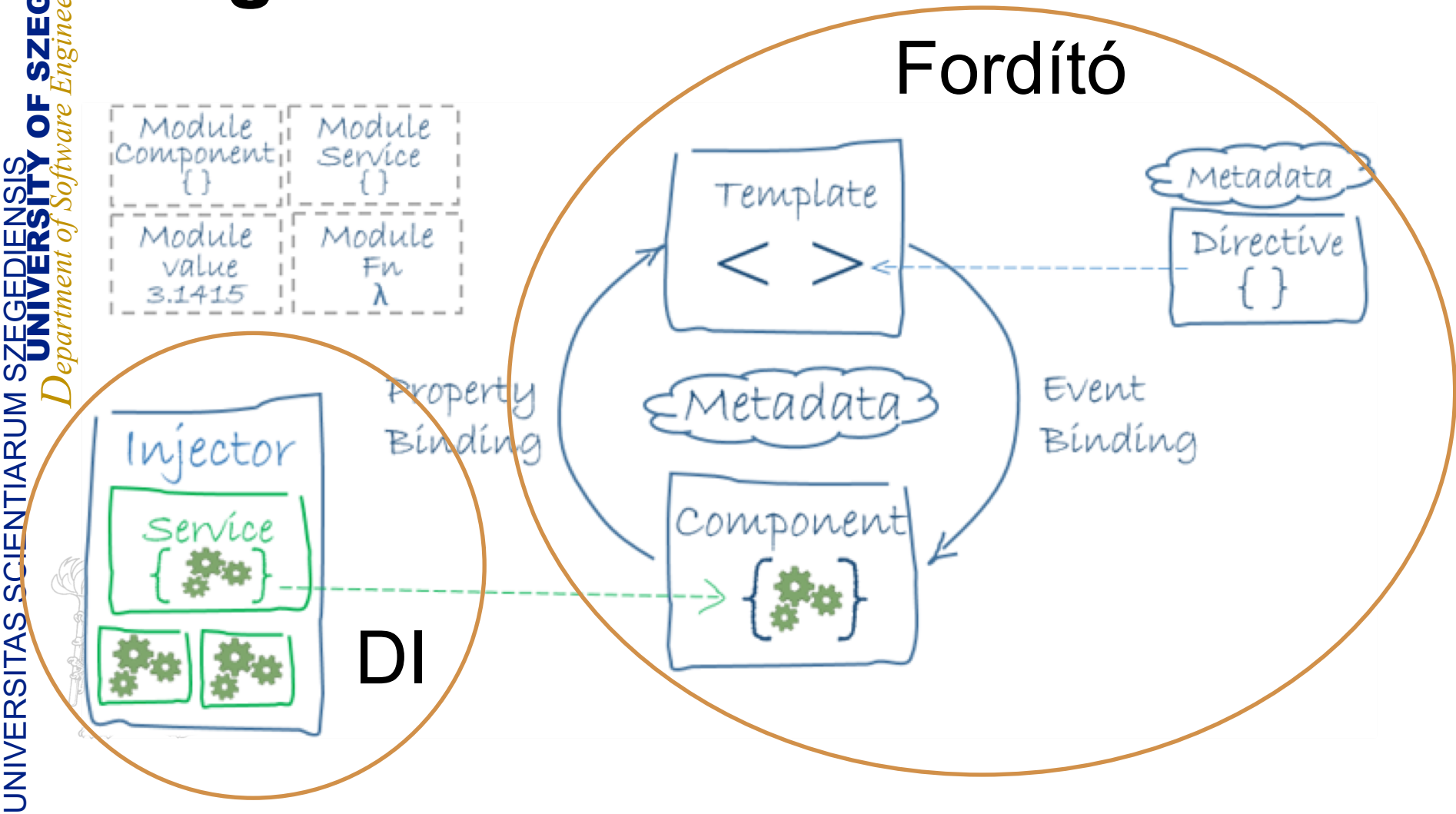
- DI
- Angular Dependency Injection
- Hierarchikus Injectorok
- DI Szolgáltatók
- DI működés közben
- Navigáció a komponens fán







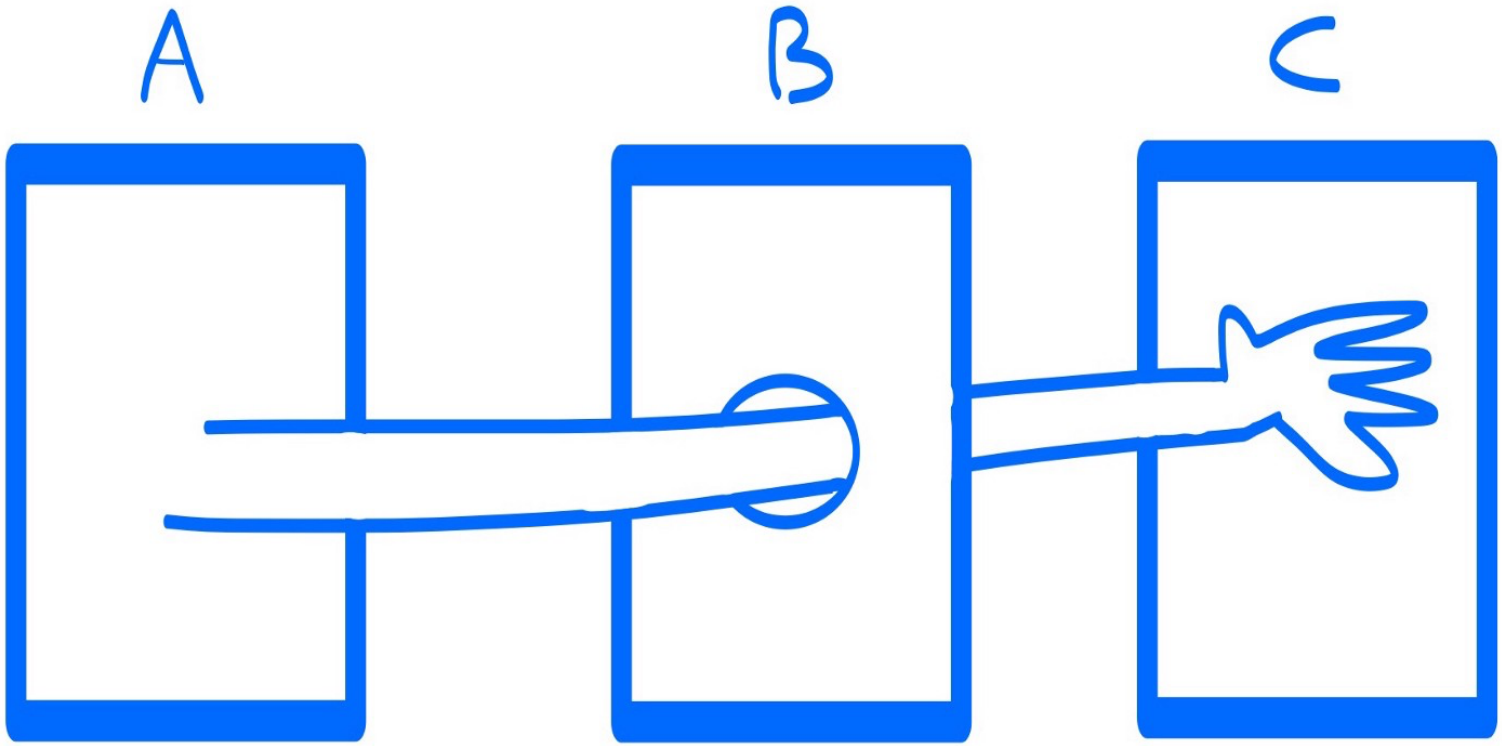
Angular architektúra



Moduláris szoftver

- Információ egységbezárása
- Modul
  - Adott interfészekkel szeparált kódrész
- Hogyan?
  - SOF (Separation of Concerns) – a belső működés elrejtése
  - SRP (Single Responsibility Principle)
  - Demeter törvénye
  - Butább komponens – Karbantarthatóbb kód
  - Függőségek

Demeter törvénye



Modulok - függőségek

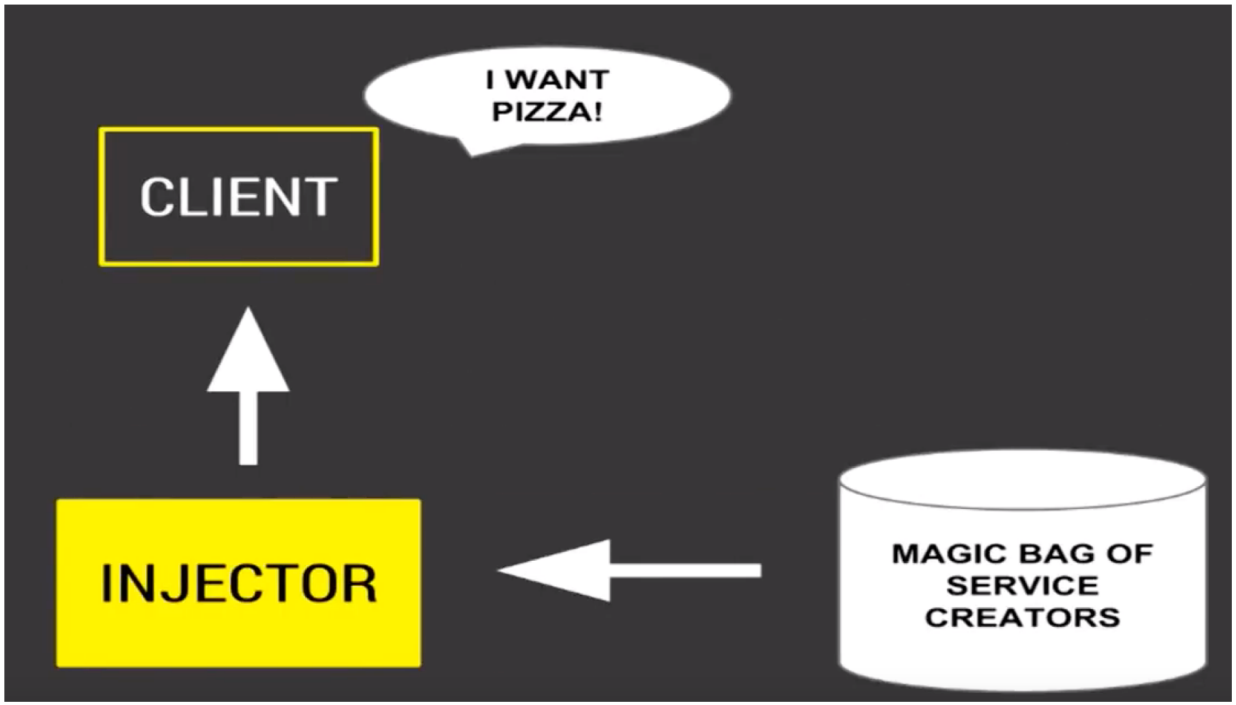
```
function Meetup(){
   var pizza = new Pizza();
   pizza.tasteGood();
}
```

```
function Meetup(){
   var pizza = locator.get("pizza");
   pizza.tasteGood();
}
```

```
function Meetup(pizza){
   pizza.tasteGood();
}
```

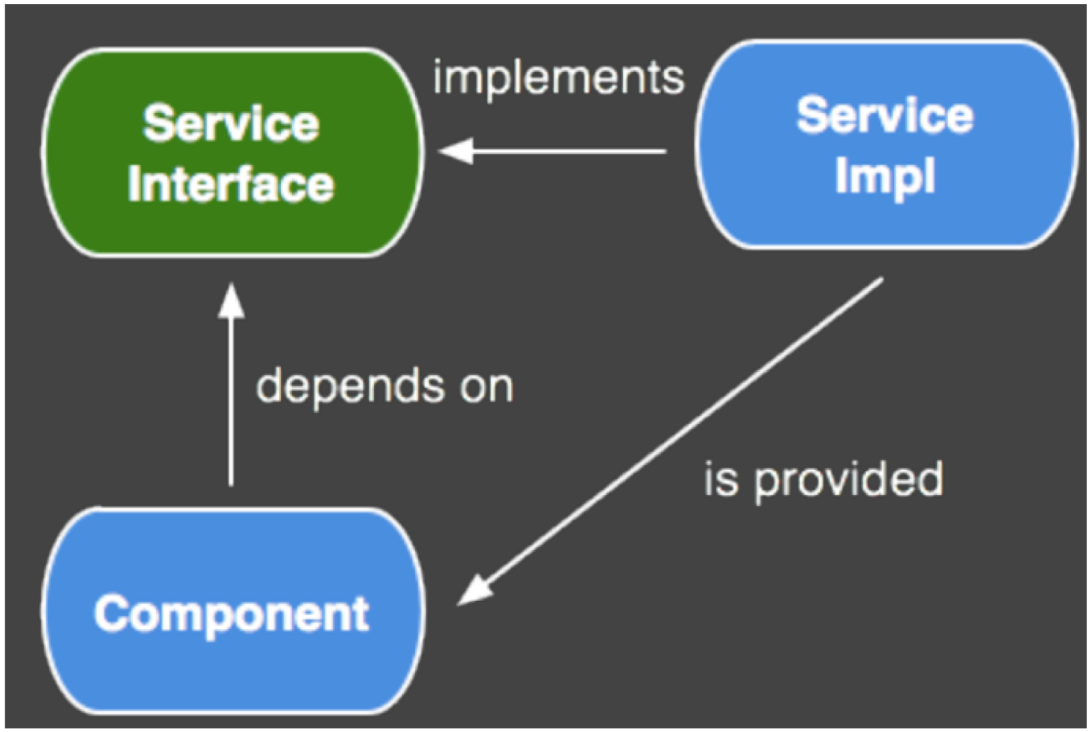
```
function main(){
   var meetup = new Meetup(
      new Pizza(new Flour(
      new Wheat(new Plow() ) ) ) );
}
```

DI

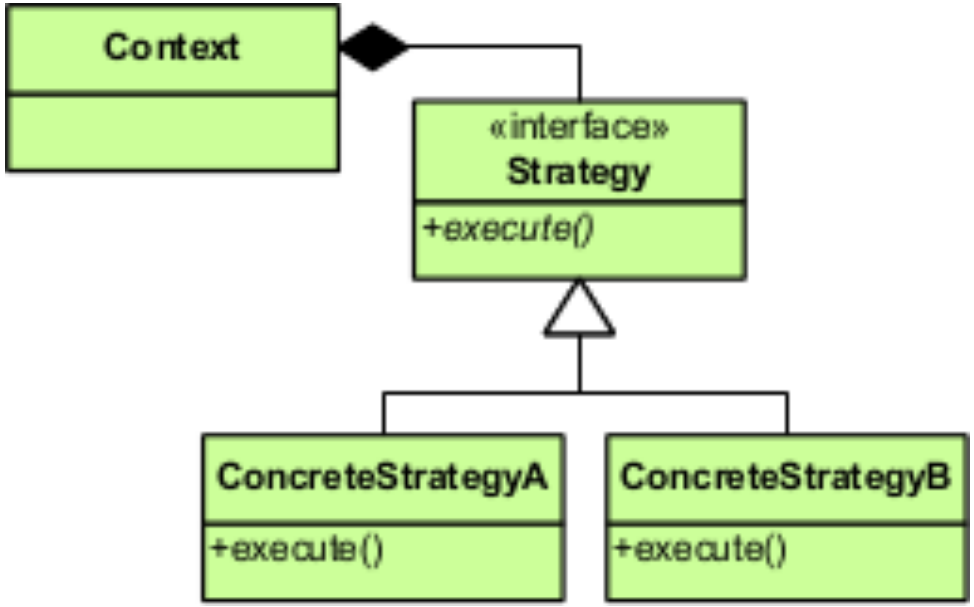


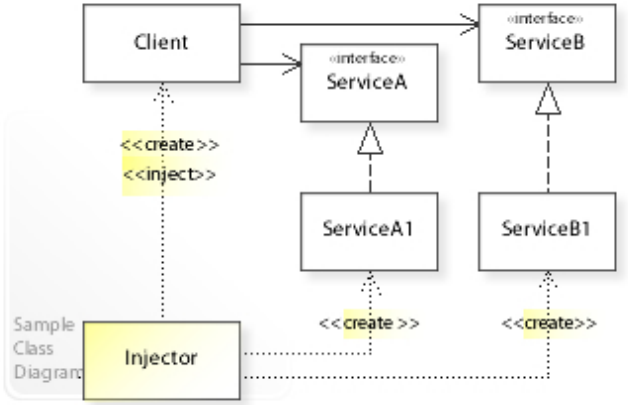
DI

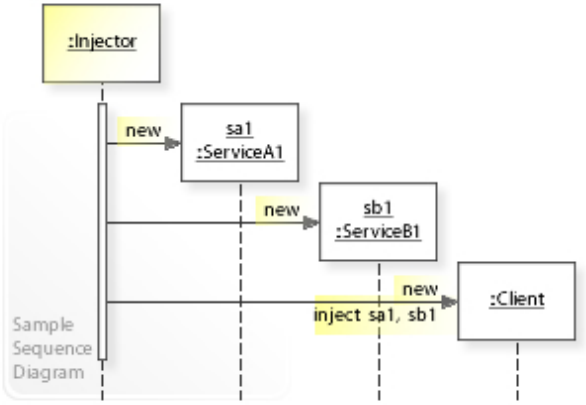
- konstruktor
- setter
- interfész



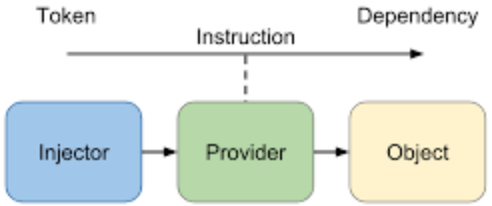
DI vs. Stratégia minta







Angular DI



- Saját DI keretrendszer
- Konstruktor alapú
- Függőségek
  - Szolgáltatások
  - Egyéb objektumok (annotált)

- Injektor
  - Hogyan injektáljuk a szolgáltatást? (Angular)
- Szolgáltató
  - Hogyan hozzuk létre. Szolgáltatást?
- @Injectable()
  - providedIn – az injektálás gyökere
- @NgModule(), @Component()
  - providers – komponens vagy modul szintű

src/app/heroes/hero.service.ts

```
import { Injectable } from '@angular/core';
import { HEROES } from './mock-heroes';

@Injectable({
  // we declare that this service should be create
  // by the root application injector.
  providedIn: 'root',
})
export class HeroService {
  getHeroes() { return HEROES; }
}
```

```
import { Component }   from '@angular/core';
import { Hero }        from './hero';
import { HeroService } from './hero.service';

@Component({
  selector: 'app-hero-list',
  template: `
    <div *ngFor="let hero of heroes">
      {{hero.id}} - {{hero.name}}
    </div>
  `
})
export class HeroListComponent {
  heroes: Hero[];

  constructor(heroService: HeroService) {
    this.heroes = heroService.getHeroes();
  }
}
```

Angular Komponensek

- Controller
- Directive
- Pipe

Nem Injektálható

- Factory
- Service
- Pipe
- Provider
- Value
- Constant

Injektálható

Hierarchikus DI

- A szolgáltatások adott injektor szkópon belül singelton-ok
- Egy gyökér injektor
- A hierarchia mentén gyermek injektorok
  - Függetlenek egymástól
  - Saját példányok
  - Élettartama az adott komponens/modul élettartamához van kötve
  - Az ősök szkópját látja, használhatja
- Szolgáltatások függőségei, hasonló mint más komponensnél

DI függőség

- Kötelező
- Opcionális (@Optional())


```
import { Injectable } from '@angular/core';
import { HEROES }     from './mock-heroes';
import { Logger }     from '../logger.service';

@Injectable({
  providedIn: 'root',
})
export class HeroService {

  constructor(private logger: Logger) {  }

  getHeroes() {
    this.logger.log('Getting heroes ...');
    return HEROES;
  }
}
```



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

@Injectable({
  providedIn: 'root'
})
export class Logger {
  logs: string[] = []; // capture logs for testing

  log(message: string) {
    this.logs.push(message);
    console.log(message);
  }
}
```

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

```
constructor(@Optional() private logger: Logger) {
  if (this.logger) {
    this.logger.log(some_message);
  }
}
```

Angular DI címkék

- Karakterlánc
- Típus
- Injektor

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

class MandrillService {};
class SendGridService {};

let injector = ReflectiveInjector.resolveAndCreate([
  { provide: "EmailService", useClass: MandrillService }  (1)
]);

let emailService = injector.get("EmailService");
console.log(emailService); // new MandrillService()
```

```
class EmailService {};
class MandrillService extends EmailService {};
class SendGridService extends EmailService {};

let injector = ReflectiveInjector.resolveAndCreate([
  { provide: EmailService, useClass: SendGridService }
]);

let emailService = injector.get(EmailService);
console.log(emailService);
```

```
import { ReflectiveInjector } from '@angular/core';
import { InjectionToken } from '@angular/core';

class MandrillService {};
class SendGridService {};
let EmailService = new InjectionToken<string>("EmailService"); (1)

let injector = ReflectiveInjector.resolveAndCreate([
  { provide: EmailService, useClass: SendGridService } (2)
]);

let emailService = injector.get(EmailService);
console.log(emailService);
```

DI címkék

- Injektor -> szolgáltatás kapcsolat – token
  - token-szolgáltató map
- Egyszerű esteben
  - Példány – Típus
- Beépített konstansok
  - PLATFORM_INITIALIZER

src/app/injector.component.ts

```
heroService: HeroService;
```

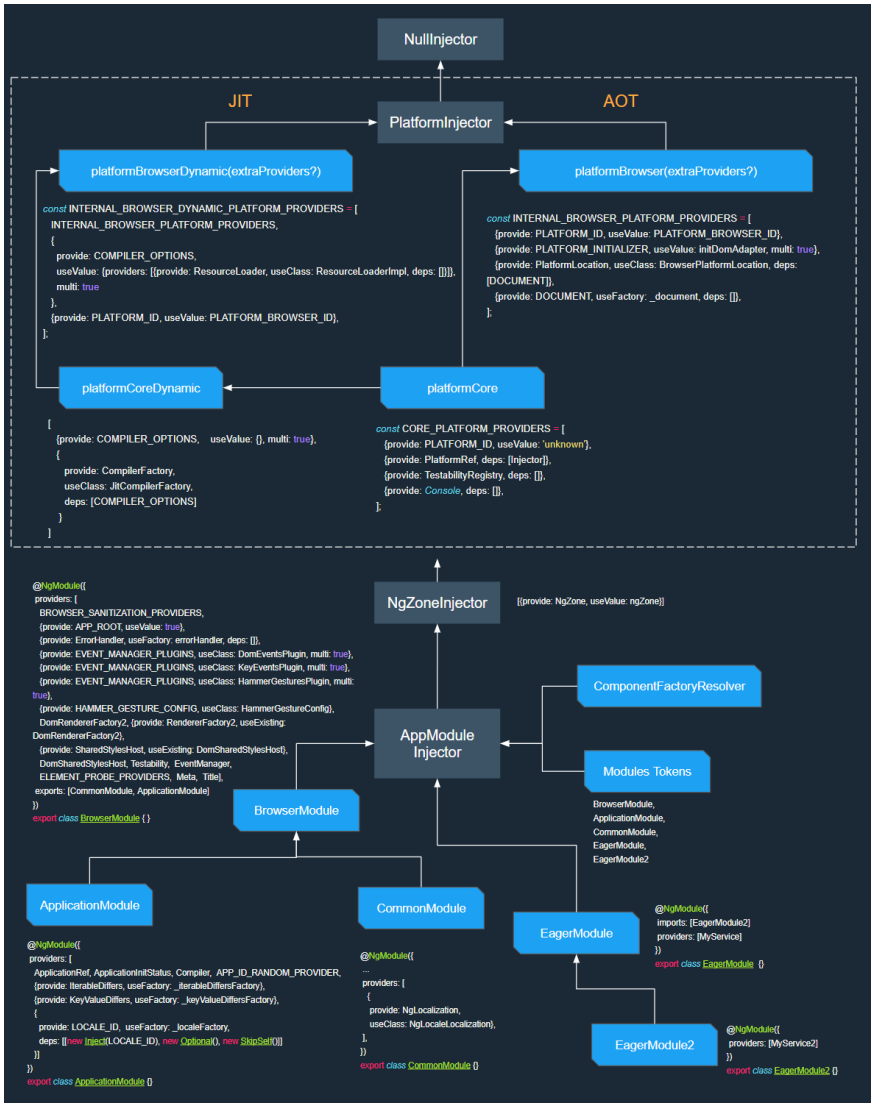
src/app/heroes/hero-list.component.ts

```
constructor(heroService: HeroService)
```

```
export const APP_TOKENS = [
 { provide: PLATFORM_INITIALIZER, useFactory: platformInitialized, multi: true    },
 { provide: APP_INITIALIZER, useFactory: delayBootstrapping, multi: true },
 { provide: APP_BOOTSTRAP_LISTENER, useFactory: appBootstrapped, multi: true },
];
```

DI helyszínek

- Elem
  - @Directive
- Komponens
  - @Component
- Modul
  - @NgModule
- Platform

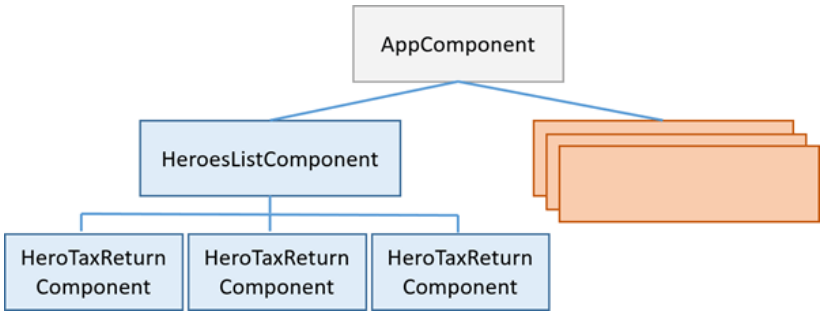


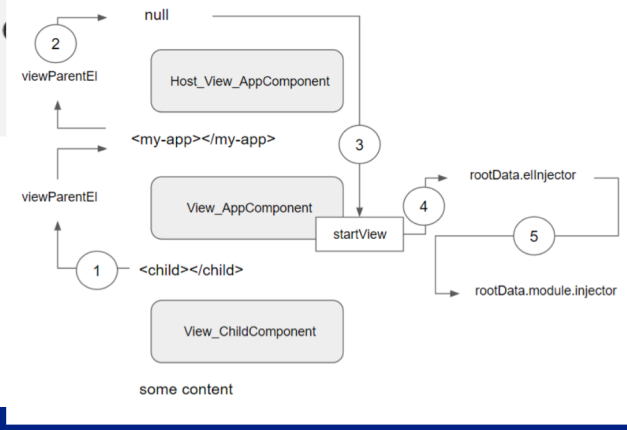
DI feloldás

- Szkóp, Élettartam (Tree Shaking)
- @Host – adott szintre szűkíti a keresést

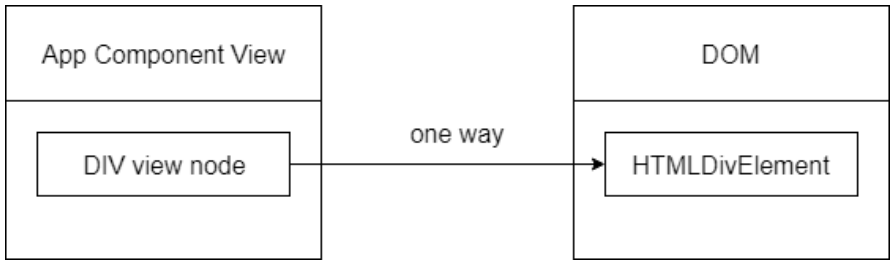
```
@Directive({
    selector: '[ngModel]',
})
export class NgModel {
    constructor(@Optional() @Host() parent: ControlContainer) {}
    private _setUpControl(): void {
        ...
        this.parent.formDirective.addControl(this);
    }
}
```

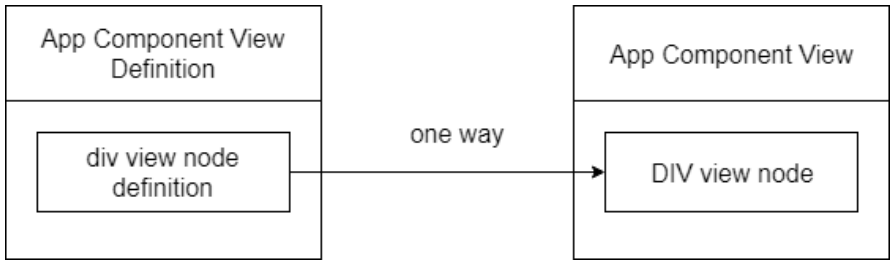
```
HostView_AppComponent
    <my-app></my-app>

View_AppComponent
    <child></child>

View_ChildCompon
    some content
```





Elem DI





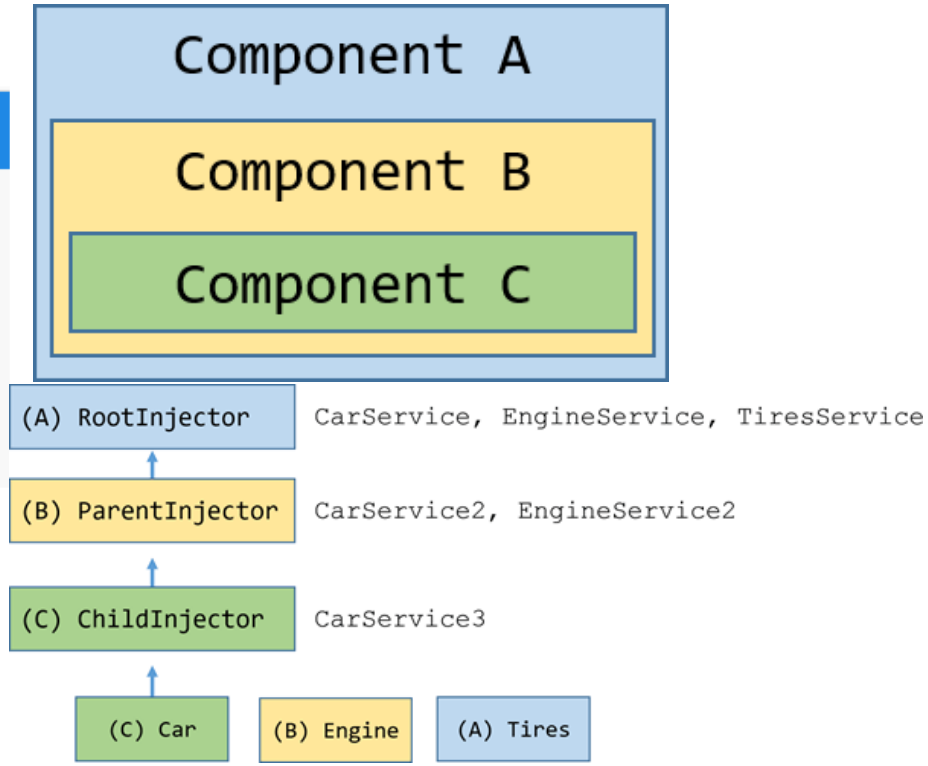

```
@Directive({
    selector: '[a]',
    providers: [ADirService]
})
export class ADirective {}
```

Komponens DI

- Párhuzamos hierarchia az alkalmazás komponens fájával
  - Szolgáltatás izoláció
  - Specializáció

src/app/villains-list.component.ts (metadata)

```
@Component({
  selector: 'app-villains-list',
  templateUrl: './villains-list.component.html',
  providers: [ VillainsService ]
})
```

DI szolgáltatók

- A DI szolgáltató egy token segítségével konfigurált injetorral biztosítja a futásidejű függőség elemet
- Hazsnálhatunk alternatív szolgáltató által létrehozott injektort is:
  - provide
  - useClass
  - useExisting
  - useValue
  - useFactory

Alternatív osztály szolgáltatók

- Ugyanazt a szolgáltatást más osztály is adhatja

```
[{ provide: Logger, useClass: BetterLogger }]
```

```
@Injectable()
export class EvenBetterLogger extends Logger {
  constructor(private userService: UserService) { super(); }

  log(message: string) {
    let name = this.userService.user.name;
    super.log(`Message to ${name}: ${message}`);
  }
}
```

```
[ UserService,
  { provide: Logger, useClass: EvenBetterLogger }]
```

Helyettesítő osztály (Alias)

- Más néven is elérhető adott szolgáltatás

```
[ NewLogger,
  // Not aliased! Creates two instances of `NewLogger`
  { provide: OldLogger, useClass: NewLogger}]
```

```
[ NewLogger,
  // Alias OldLogger w/ reference to NewLogger
  { provide: OldLogger, useExisting: NewLogger}]
```

Érték biztosítók (Value provider)

- Kész objektumot szeretnénk átadni

```
// An object in the shape of the logger service
export function SilentLoggerFn() {}

const silentLogger = {
  logs: ['Silent logger says "Shhhhh!". Provided via "useValue"'],
  log: SilentLoggerFn
};
```

```
[{ provide: Logger, useValue: silentLogger }]
```

Factory biztosítók

- A függő változó csak futási időben derül ki
- Nem szolgáltatás beburkolása szolgáltatásba

src/app/heroes/hero.service.provider.ts (excerpt)

```
let heroServiceFactory = (logger: Logger, userService: UserService) => {
  return new HeroService(logger, userService.user.isAuthorized);
};
```

src/app/heroes/hero.service.provider.ts (excerpt)

```
export let heroServiceProvider =
  { provide: HeroService,
    useFactory: heroServiceFactory,
    deps: [Logger, UserService]
  };
```

Farázás képes biztosítók

- A nem hivatkozott kódot a fordító kiveszi a csomagból
- A Farázás képes biztosítókat a fordító képes kivenni, ha nincs rájuk szükség
- NgModul és Komponens szintű injektáló biztosítók nem Farázás képesek

src/app/tree-shaking/service-and-modules.ts

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

@Injectable()
export class Service {
  doSomething(): void {
  }
}

@NgModule({
  providers: [Service],
})
export class ServiceModule {
}
```

src/app/tree-shaking/app.modules.ts

```
@NgModule({
  imports: [
    BrowserModule,
    RouterModule.forRoot([]),
    ServiceModule,
  ],
})
export class AppModule {
}
```

Farázás képes biztosító

src/app/tree-shaking/service.ts

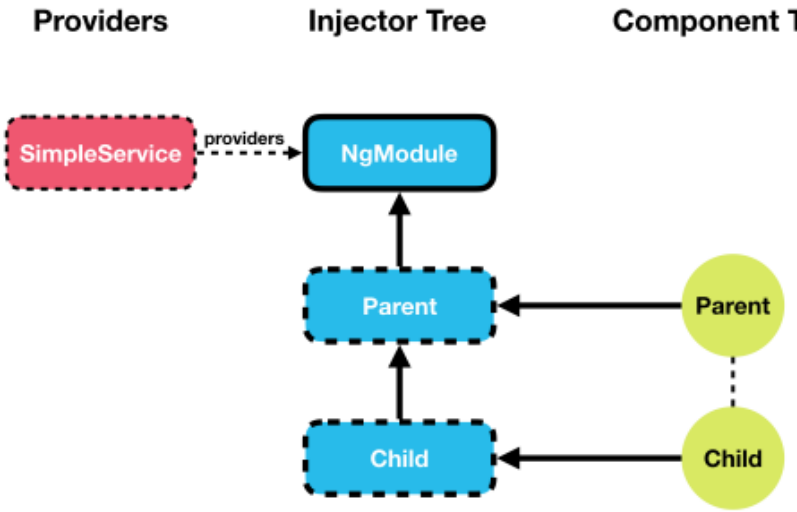
```
@Injectable({
  providedIn: 'root',
})
export class Service {
}
```

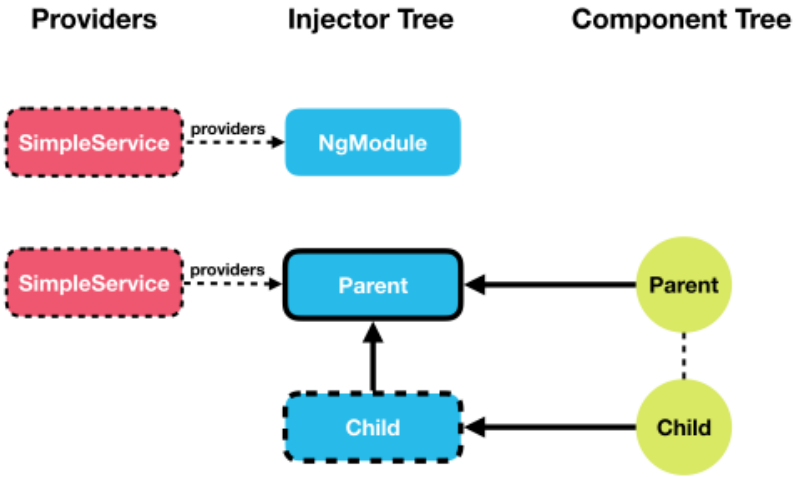

src/app/tree-shaking/service.0.ts

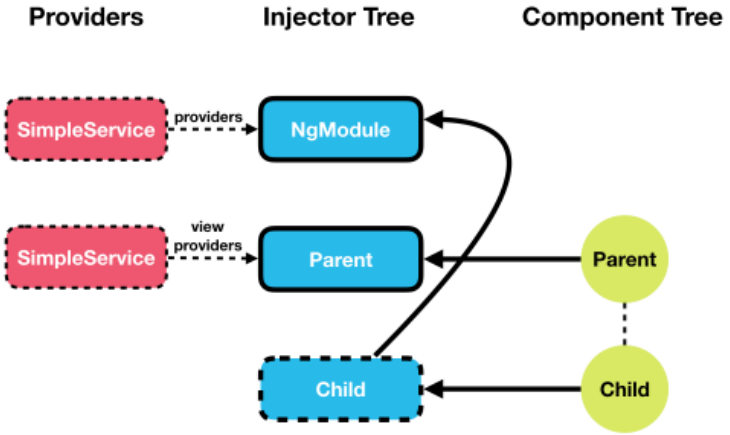
```
@Injectable({
  providedIn: 'root',
  useFactory: () => new Service('dependency'),
})
export class Service {
  constructor(private dep: string) {
  }
}
```

Összefoglaló

- NgModule
- Component
- viewChild

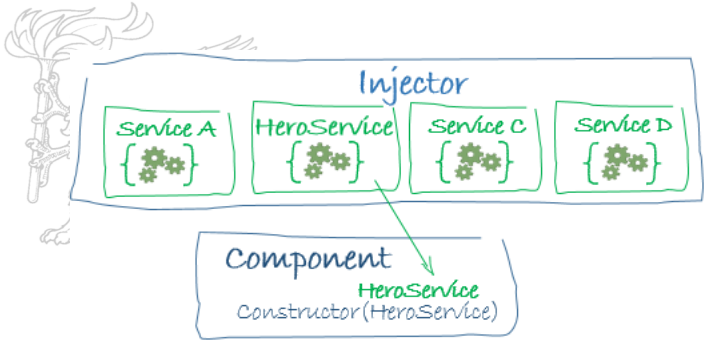


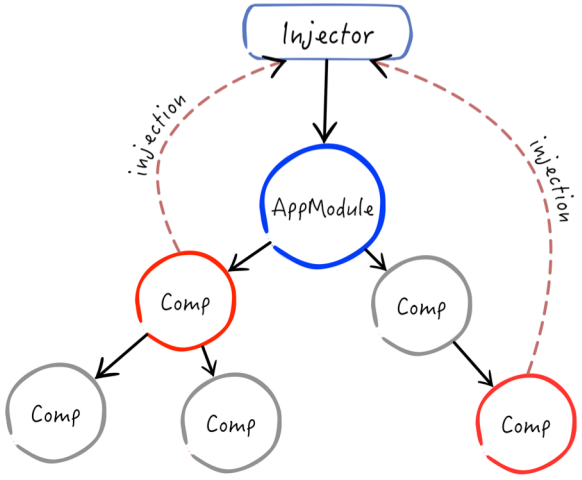




Áttekintés

- DI
- Angular Dependency Injection
- Hierarchikus Injectorok
- DI Szolgáltatók
- DI működés közben





A következő előadás összefoglalója

- Angular Routing, Navigation
  - Router import
  - Konfiguráció
  - Router outlet
  - Router link
  - Aktív router link
  - Router állapot
  - Aktivált útvonal
  - Router események