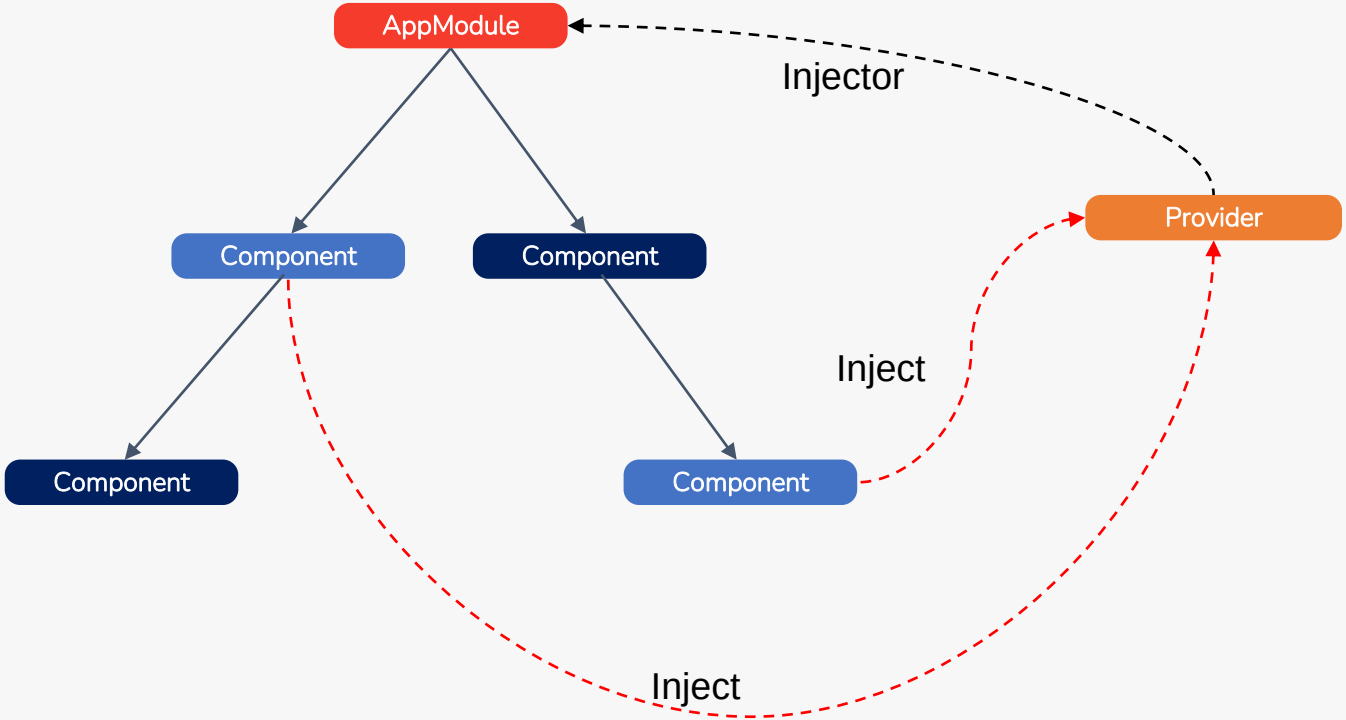
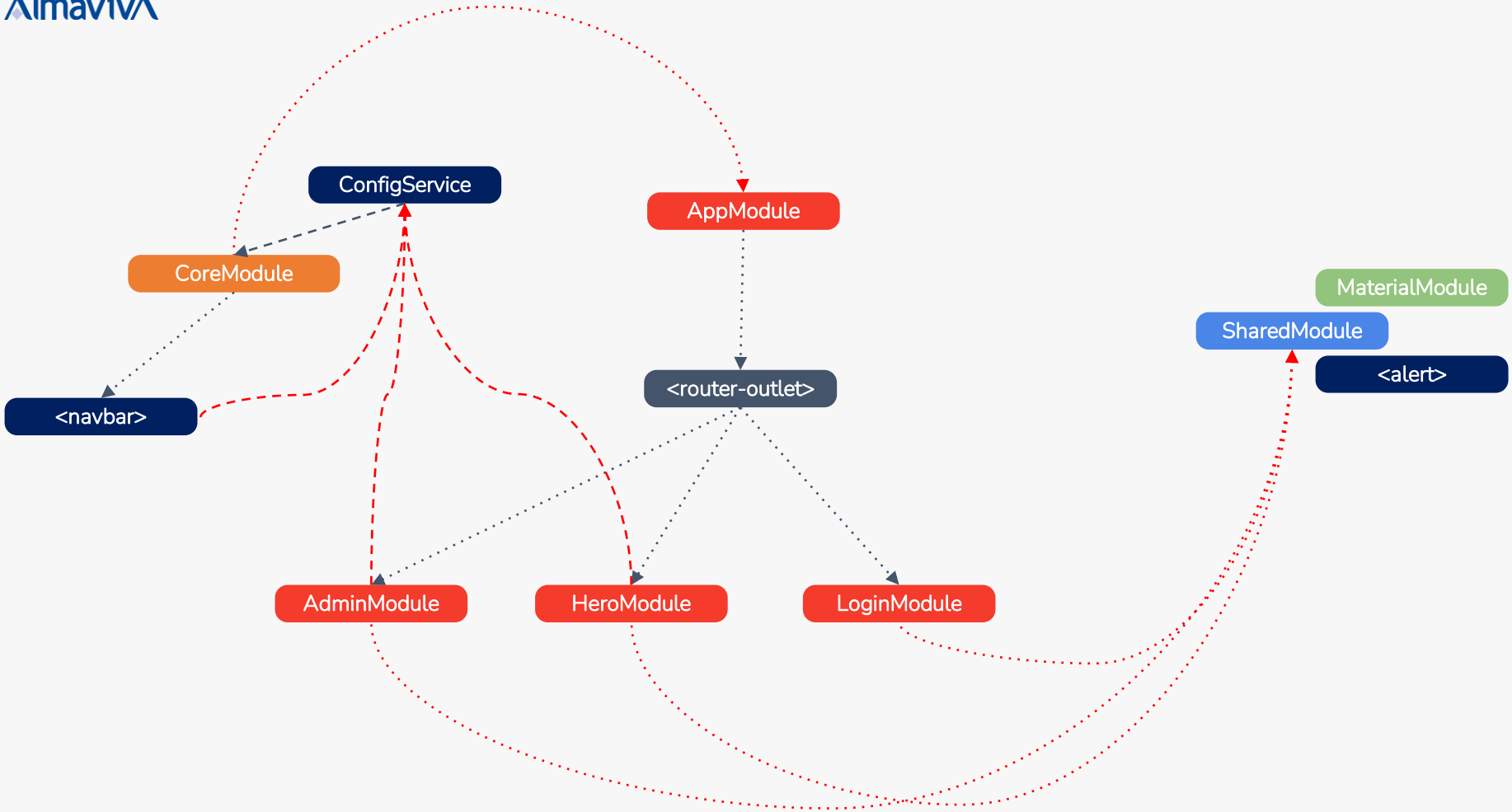
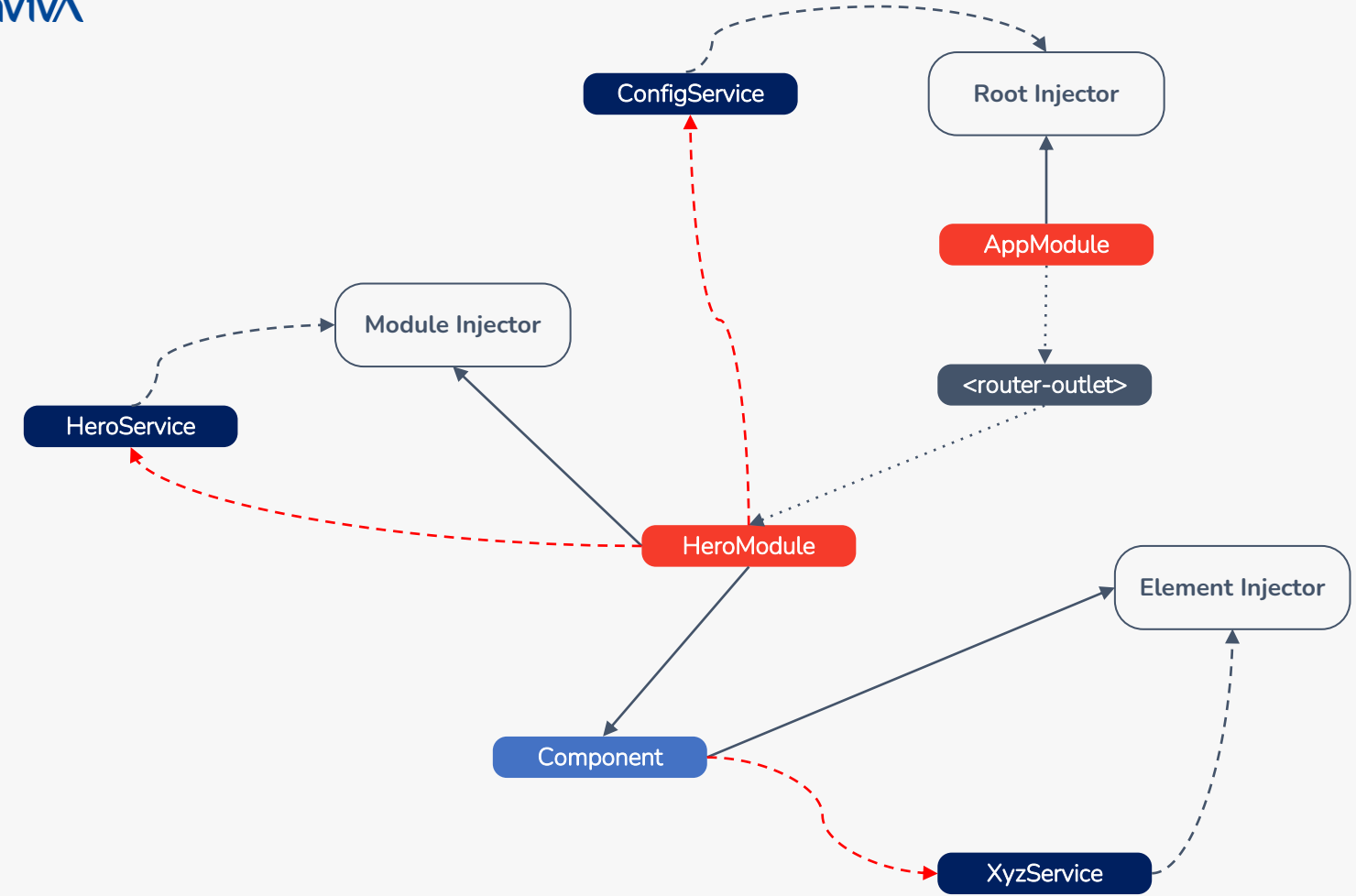

# Services & DI

*Fravezzi Mattia*
*m.fravezzi@almaviva.it*

# Dependency Injection




ConfigService
AppModule
CoreModule
MaterialModule
SharedModule
<alert>
<router-outlet>
<navbar>
AdminModule
HeroModule
LoginModule


ConfigService
Root Injector
AppModule
Module Injector
<router-outlet>
HeroService
HeroModule
Element Injector
Component
XyzService

# Injector Gerarchici

- Module Injector
- Element Injector
- Platform Module Injector
- Platform Null Injector

AlmavivA

# Module Injector

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

```
@NgModule({
    providers: [ HeroService ]
})
export class HeroPageModule { }
```

AlmavivA

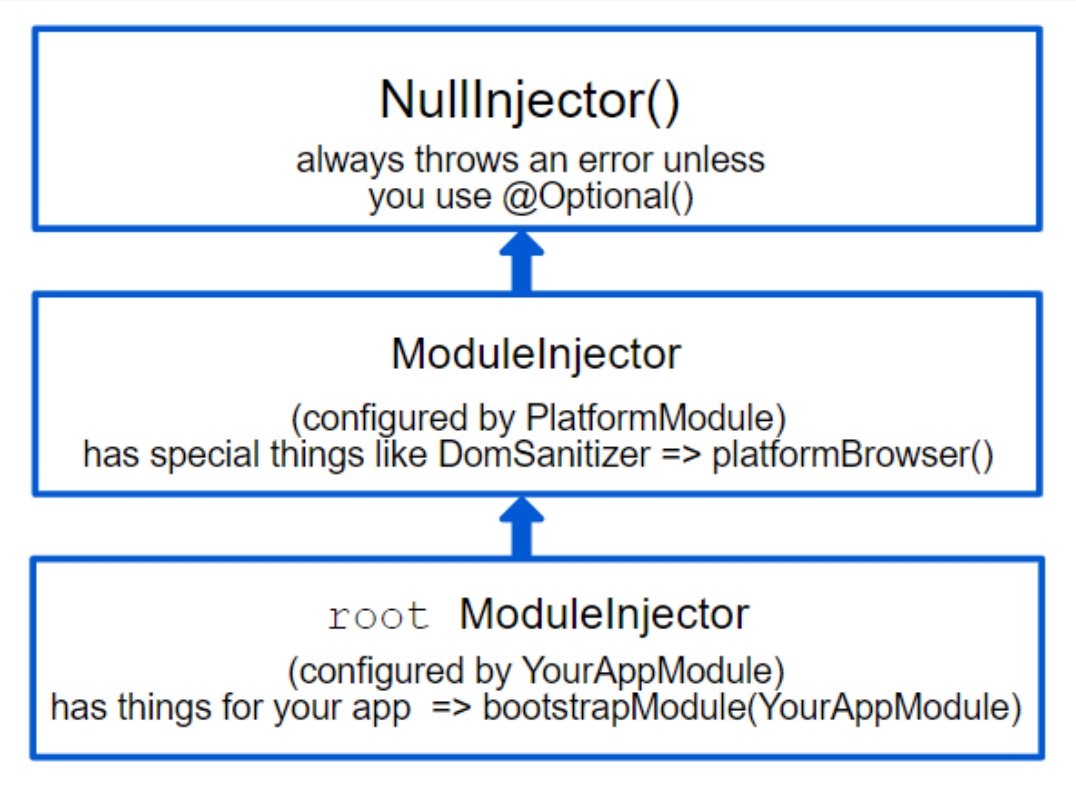
# Element Injector

```
@Component({
    selector: 'app-hero-add',
    templateUrl: './hero-add.component.html',
    styleUrls: ['./hero-add.component.scss'],
    providers: [{ provide: ItemService, useValue: { name: 'lamp' } }]
})
```

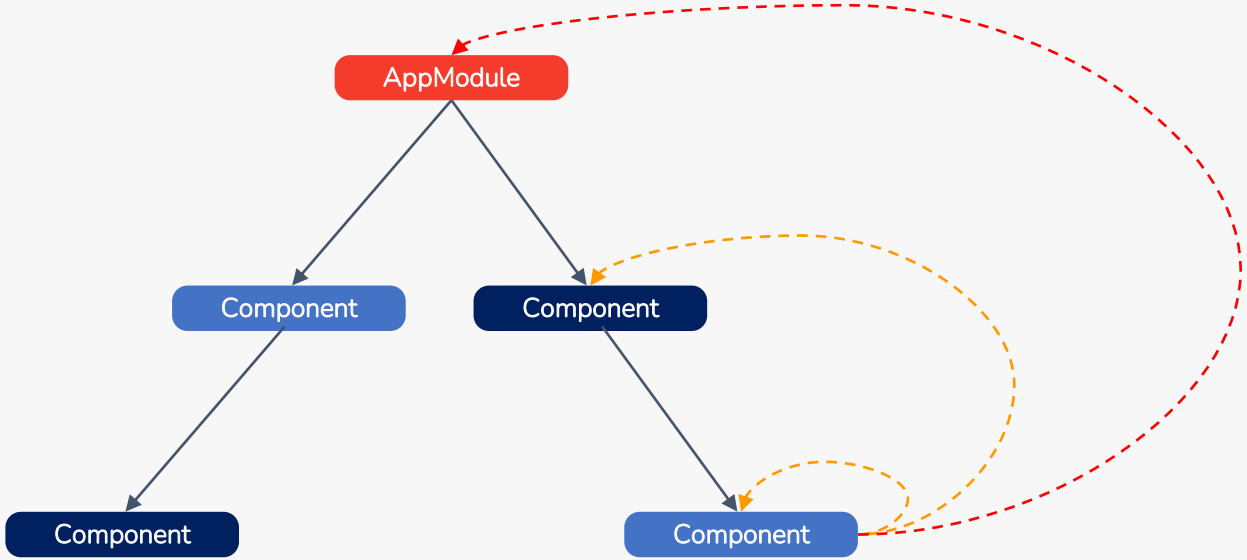
# Platform Module Injector

```
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

# Gerarchia app-platform



# Regole di risoluzione



# Modificatori di Risoluzione

- `@Optional()`
- `@SkipSelf()`
- `@Self()`
- `@Host()`

# Services

```
@Injectable({
    providedIn: 'root',
    useClass: HeroServiceTest,
    useExisting: HeroServiceTest,
    useFactory: heroServiceFactory,
    useValue: {
        getHeroes: function () {
            return [... ]
        }
    },
    deps: [HttpClient]
})
export class HeroService {
    constructor( ... ) { }
    public getHeroes() { ... }
}
```

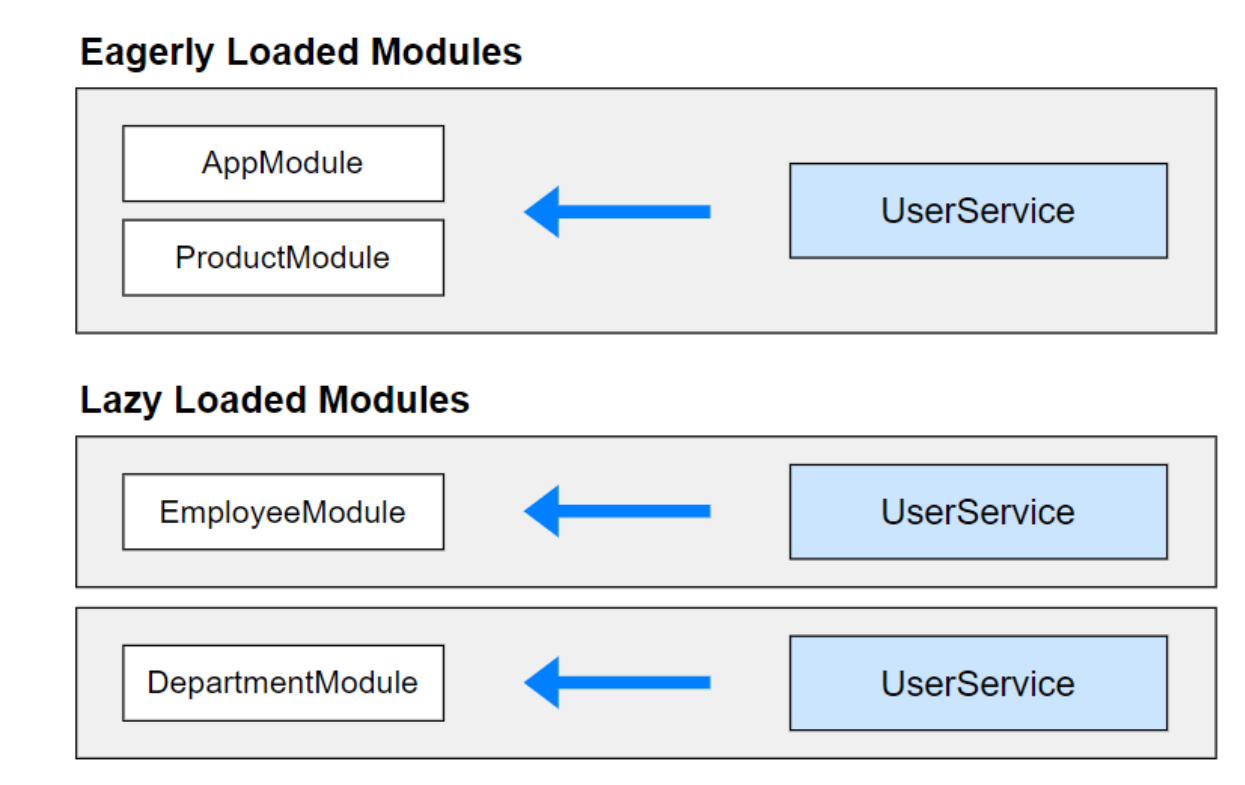
```
@NgModule({
    ...
    providers: [
        HeroService
    ]
})
export class HeroPageModule { }
```

```
@Component({ ... })
export class HeroListComponent {
    constructor(
        public heroService: HeroService
    ) { }
    ngOnInit(): void {
        this.heroService.getHeroes();
    }
}
```

# Services - provideIn

- ‘root’ - singleton su tutta l’applicazione
- ‘any’ - singleton sui moduli caricati “eagerly”, e istanza separata per quelli caricati “lazy”
- HeroPageModule - caricamento solo nel modulo selezionato
- provides nel modulo lazy di caricamento

# Services - provideIn: ‘any’



# Services - useFactory

```
@Injectable({
    useFactory: heroServiceFactory,
    deps: [
      HttpClient,
      HeroStore
   ]
})
export class HeroService {
    constructor( ... ) { }
    public getHeroes() { ... }
}
```

```
export function heroServiceFactory(
    httpClient: HttpClient,
    heroStore: HeroStore
) {
    if (!!environment.production) {
        return new HeroService(httpClient, heroStore)
    }
    return new HeroServiceTest(httpClient, heroStore);
}
```

# Services - ngModule providers array

```
@NgModule({
    ...
    providers: [
        HeroService,
        HeroStore
    ]
})
export class HeroPageModule { }
```

# Services - forRoot() pattern

- Use the providedIn syntax instead of registering the service in the module.
- Separate your services into their own module.
- Define forRoot() and forChild() methods in the module.

```
NgModule({
    declarations: [ ... ],
    imports: [ ... ],
    exports: [ ... ]
})
export class CoreModule {
    public static forRoot() {
        return {
            ngModule: CoreModule,
            providers: [
                ConfigService
            ]
        }
    }
```

```
public static forChild() {
        return {
            ngModule: CoreModule
        };
    }

    constructor(@Optional() @SkipSelf() parentModule: CoreModule) {
        if (parentModule) {
            throw new Error('CoreModule is already loaded.');
        }
    }
}
```

# DEMO

AlmavivA
digitale assoluto

almaviva.it