DI

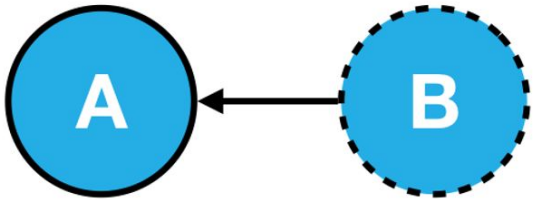
Nicola Toledo

DI

Cos'è una Dependency?

Quando il modulo A in un' applicazione richiede il modulo B da eseguire,

il modulo B è una *dipendenza* del modulo A.



DI Angular

Token

Questo identifica in modo univoco qualcosa che vogliamo iniettare. Una *dipendenza* del nostro codice.

Dependancy

Il codice effettivo che vogliamo iniettare.

Provider

Questa è una mappa tra un *token* e un elenco di *dipendenze* .

Injector

Questa è una funzione che quando viene passato un *token* restituisce una *dipendenza* (o un elenco di dipendenze)

Injectors

Al centro del framework DI c'è un *iniettore* .

Un iniettore riceve un *token* e restituisce una *dipendenza* (o un elenco di).

Diciamo che un iniettore *risolve* un token in una dipendenza.

Injector

```
import { ReflectiveInjector } from '@angular/core'; (1)

class MandrillService {}; (2)
class SendGridService {};

let injector = ReflectiveInjector.resolveAndCreate([  (3)
  MandrillService,
  SendGridService
]);

let emailService = injector.get(MandrillService); (4)
console.log(emailService);
```

Injector

Un *diverso* iniettore per lo *stesso* token potrebbe restituire un'istanza *diversa* di una dipendenza, ma lo *stesso* iniettore restituirà sempre la *stessa* istanza.

```
let emailService1 = injector.get(MandrillService);
let emailService2 = injector.get(MandrillService);
console.log(emailService1 === emailService2); // true
```

Injector

quindi possiamo condividere lo *stato* tra due diverse parti della nostra applicazione iniettando la stessa dipendenza, in questo modo:

```
let emailService1 = injector.get(MandrillService);
emailService1.foo = "moo";

let emailService2 = injector.get(MandrillService);
console.log(emailService2.foo); // moo (1)
```

Child Injectors

Gli iniettori possono avere uno o più iniettori *child*

Ogni iniettore crea la propria *istanza* di dipendenza

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

class EmailService {}

let injector = ReflectiveInjector.resolveAndCreate([EmailService]); (1)
let childInjector = injector.resolveAndCreateChild([EmailService]);

console.log(injector.get(EmailService) === childInjector.get(EmailService)); // false (2)
```

Child Injectors

Gli iniettori secondari inoltrano le richieste al proprio genitore che si trova in stato di Injection se non riescono a risolvere il token localmente.

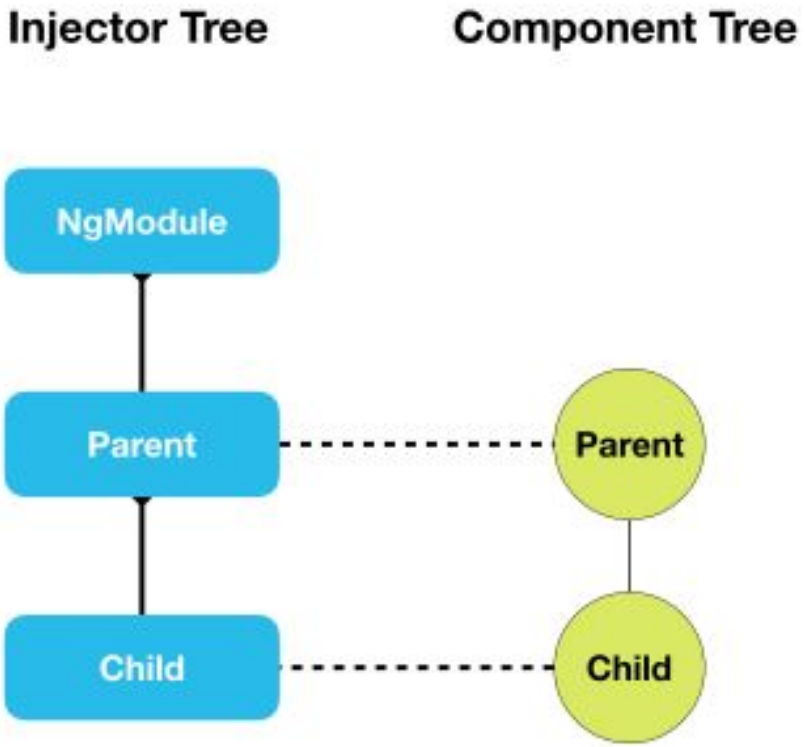
```
import { ReflectiveInjector } from '@angular/core';

class EmailService {}

let injector = ReflectiveInjector.resolveAndCreate([EmailService]); (1)
let childInjector = injector.resolveAndCreateChild([]); (2)

console.log(injector.get(EmailService) === childInjector.get(EmailService)); // true (3)
```

The Injector Tree



Injector configuration

Module

```
@NgModule({
  providers: [EmailService, JobService]
})
class AppModule { }
```

Injector configuration

Component e Direttive

```
@Component({
  selector: 'my-comp',
  template: `...`,
  providers: [EmailService]
})
```

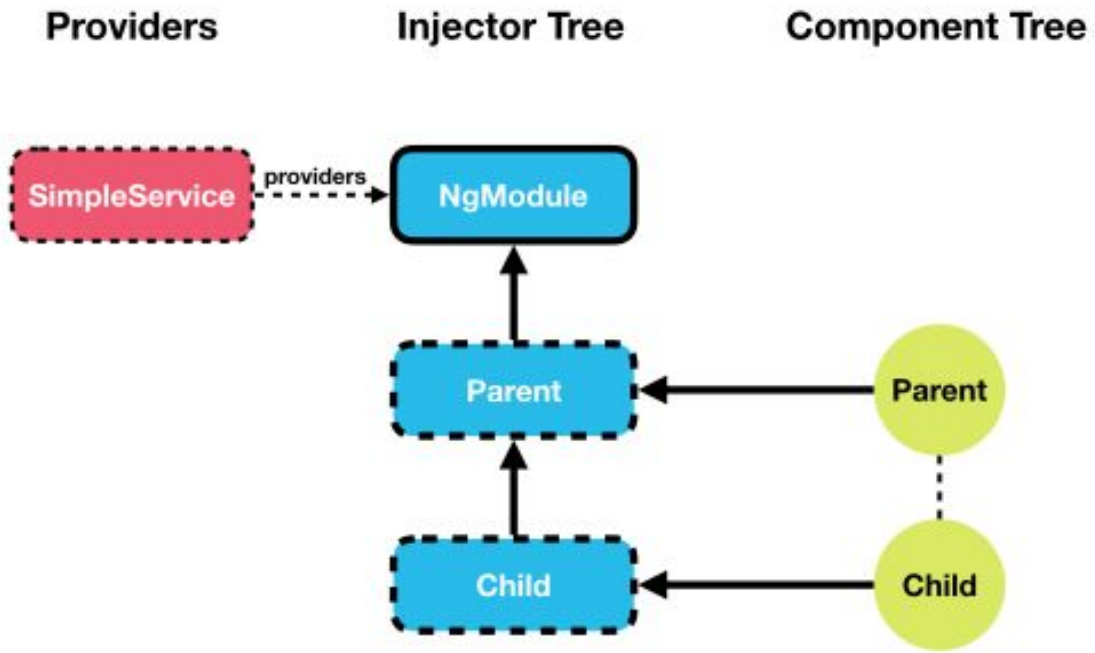
Injector configuration

viewProviders che crea uno speciale injector che risolve le dipendenze solo per *questo* componente,

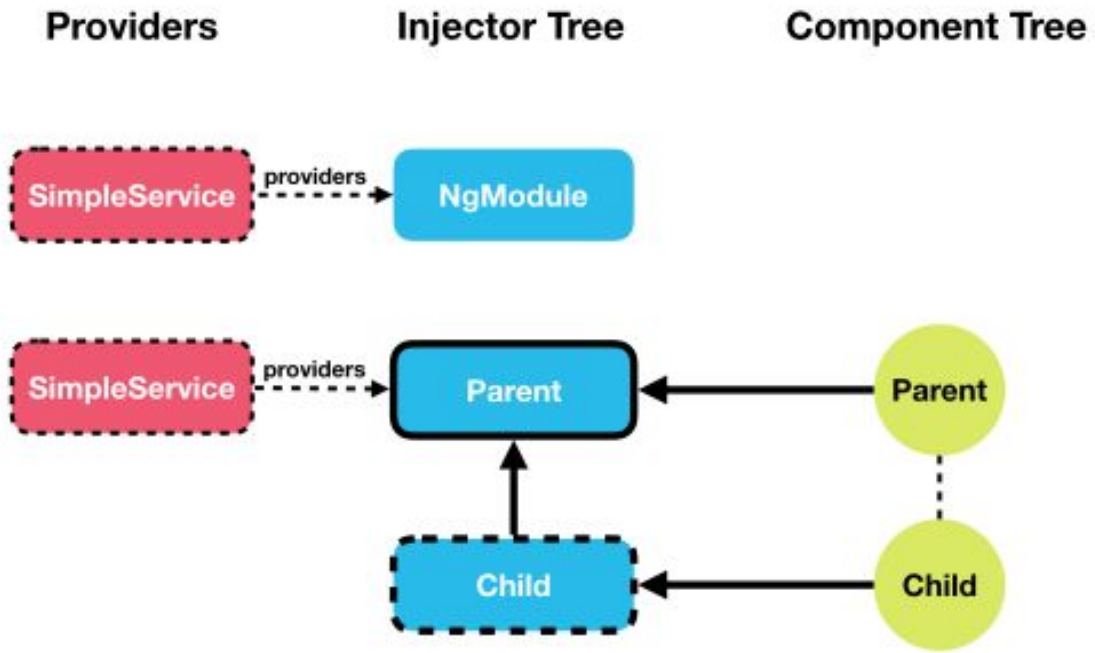
non agisce come un iniettore padre per qualsiasi contenuto di child

```
@Component({
  selector: 'my-comp',
  template: `...`,
  viewProviders: [EmailService]
})
```

Providers visibility



Providers visibility



Providers visibility