ngOnInit not being called when Injectable class is Instantiated

Asked 3 years, 8 months ago Active 1 month ago Viewed 86k times

175

27

Why isn't `ngOnInit()` called when an `Injectable` class is resolved?

Code

```
import {Injectable, OnInit} from 'angular2/core';
import { RestApiService, RestRequest } from './rest-api.service';

@Injectable()
export class MovieDbService implements OnInit {

    constructor(private _movieDbRest: RestApiService){
        window.console.log('FROM constructor()');
    }

    ngOnInit() {
         window.console.log('FROM ngOnInit()');
    }

}
```

Console Output

```
FROM constructor()
```

javascript angular typescript

edited Aug 14 '17 at 19:45
student
13.4k 10 87 144

asked Jan 31 '16 at 5:33
Levi Fuller
5,151 4 31 40

4 Answers

273

Lifecycle hooks, like `OnInit()` work with Directives and Components. They do not work with other types, like a service in your case. From docs:

> A Component has a lifecycle managed by Angular itself. Angular creates it, renders it, creates and renders its children, checks it when its data-bound properties change and destroy it before removing it from the DOM.
>
> Directive and component instances have a lifecycle as Angular creates, updates, and destroys them.

edited Sep 7 '18 at 5:26
Rohit Sharma
2,677 2 13 28

answered Jan 31 '16 at 5:49
Sasxa
30.5k 11 76 95

31 So simply move my `ngOnInit` logic to the constructor for `Injectable` classes? I just remembered reading that you should keep any logic out of the constructor for whatever reason. – Levi Fuller Jan 31 '16 at 6:00

40 @LeviFuller Yea, for services you can do initialization in the constructor, or better make init method and call it from the constructor (in case you need to reset a service later). – Sasxa Jan 31 '16 at 6:03

9 Not exactly, OnInit has to do with binding. You use it when you want to wait for input values to be resolved, as they are not available in the constructor. Constructor can be called multiple times. For example, when you change route and load different components they get "constructed" and destroyed each time... – Sasxa Jan 31 '16 at 6:10

5 It's also worth noting here that Sevices *can* be instantiated multiple times, depending on where you include them in `providers` arrays. If you want a singleton service, put it in your main module's `providers` , and if you want per-component services add them to the component directly. – Askdesigners Apr 5 '17 at 12:38

4 This is not entirely incorrect, as @SBD580 points in his answer, `ngOnDestroy` is called for injected services – shusson Apr 12 '17 at 7:58

45

I don't know about all the lifecycle hooks, but as for destruction, `ngOnDestroy` actually get called on Injectable when it's provider is destroyed (for example an Injectable supplied by a component).

From the `docs :`

> Lifecycle hook that is called when a directive, pipe or **service** is destroyed.

Just in case anyone is interested in *destruction* check this question:

Abdullah Khan
6,530 3 32 54

SBD580
481 4 3

2 such a shame `ngOnInit` isn't called :-(I really wanted to do some transparent delayed initialization magic. If `ngOnDestroy` can be called I don't see why init can't – Simon_Weaver Mar 11 at 8:55

3

Note: this answer applies only to Angular components and directives, NOT services.

I had this same issue when `ngOnInit` (and other lifecycle hooks) were not firing for my components, and most searches led me here.

The issue is that I was using the arrow function syntax (`=>`) like this:

```
class MyComponent implements OnInit {
    // Bad: do not use arrow function
    public ngOnInit = () => {
        console.log("ngOnInit");
    }
}
```

Apparently that does not work in Angular 6. Using non-arrow function syntax fixes the issue:

```
class MyComponent implements OnInit {
    public ngOnInit() {
        console.log("ngOnInit");
    }
}
```

edited May 27 at 14:31

answered May 29 '18 at 14:15
AJ Richardson
4,849 33 51

Injectable classes are Services, not Components. OP asks about a Service. Although your answer may be correct from a language POV, it doesn't answer this question. You should delete it. – Andrew Philips May 16 at 6:41

@AndrewPhilips while my use case is slightly different from the OP's, a google search for "ngOnInit not called" leads people here. My goal with this answer is not to help the OP, but to help any of the other 70,000+ viewers who might have has similar issues with `ngOnInit` – AJ Richardson May 16 would have confused me two weeks ago. Both from an SO and NG perspective, I think your answer is worthy of its own Question, so not deleted but "recategorized". Who knows? Perhaps it might reach an even wider audience. Cheers. – Andrew Philips May 16 at 12:33

1 Makes sense - I edited my answer to make it clear that this is not for services. – AJ Richardson May 27 at 14:32

0

In your service you can use `resolvers` for the same purpose, as explained in this post: https://codeburst.io/understanding-resolvers-in-angular-736e9db71267

> To understand the use of resolvers, Lets see how the flow happens, when someone clicks the link.

General Routing Flow.

- User clicks the link.
- Angular loads the respective component.

Routing Flow with Resolver

- User clicks the link.
- Angular executes certain code and returns a value or observable.
- You can collect the returned value or observable in constructor or in ngOnInit, in class of your component which is about to load.
- Use the collected the data for your purpose.
- Now you can load your component.

> Steps 2, 3 and 4 are done with a code called Resolver.

answered Aug 25 at 7:59
118218
433 8 32