Angular 2 - Routing - CanActivate work with Observable

Asked 3 years, 3 months ago Active 7 months ago Viewed 54k times

73

31

I have an **AuthGuard** (used for routing) that implements **CanActivate**.

```
canActivate() {
    return this.loginService.isLoggedIn();
}
```

My problem is, that the CanActivate-result depends on a http-get-result - the **LoginService** returns an **Observable**.

```
isLoggedIn():Observable<boolean> {
    return this.http.get(ApiResources.LOGON).map(response => response.ok);
}
```

How can i bring those together - make CanActivate depend on a backend state?

angular angular2-routing angular2-http

asked Jun 21 '16 at 15:04

Philipp
1,809 6 17 36

1 Have you read here? angular.io/docs/ts/latest/guide/router.html search for Route Guards Here is api reference for CanActivate: angular.io/docs/ts/latest/api/router/index/… as you see it can return either boolean or Observable<boolean> – mollwe Jun 22 '16 at 2:04

4 `canActivate()` can return an `Observable`, just make sure that the `Observable` has completed (ie. `observer.complete()`). – Philip Bulley Jul 14 '16 at 13:31

1 @PhilipBulley what if the observable emits more values and then completes? What does the guard do? What I have seen so far is use of `take(1)` Rx operator to achieve the completnes of stream, What if I forget to add it? – Felix Nov 9 '17 at 14:16

2 Answers

122

You should upgrade "@angular/router" to the latest . e.g."3.0.0-alpha.8"

modify AuthGuard.ts

```
@Injectable()
export class AuthGuard implements CanActivate {
    constructor(private loginService:LoginService, private router:Router) { }

    canActivate(next:ActivatedRouteSnapshot, state:RouterStateSnapshot) {
        return this.loginService.isLoggedIn().map(e => {
            if (e) {
                return true;
            }
        }).catch(() => {
            this.router.navigate(['/login']);
            return Observable.of(false);
        });
    }
}
```

If you have any questions, ask me !

edited Oct 25 '16 at 20:37

Nick Gent
50 1 6

answered Jun 27 '16 at 1:15

Kery Hu
2,161 5 22 44

3 It's worth pointing out that this works with promises in a very similar way. For my implementation, assuming `isLoggedIn()` is a `Promise` , you can do `isLoggedIn().then((e) => { if (e) { return true; } }).catch(() => { return false; });` Hopefully this helps future travelers! – kbpontius Jan 26 '17 at 20:32

5 I had to add `import 'rxjs/add/observable/of';` – marc_aragones Mar 7 '17 at 12:25

not a good answer now IMO .. it provides no detail of what is happening server side ... it is out of date at alpha! ... it does not follow this best practice .. angular.io/docs/ts/latest/guide/... .. see my updated answer below (hopefully one day above) – danday74 Mar 9 '17 at 18:15

canActivate should retun Observable<boolean> – Yoav Schniederman Mar 28 '17 at 11:08

Great Help :) Thanks – gschambial Jul 31 '17 at 17:38

36

```
import { Injectable } from '@angular/core';
import { CanActivate, Router, ActivatedRouteSnapshot, RouterStateSnapshot } from
'@angular/router';
import { Observable } from 'rxjs/Observable';
import { catchError, map} from 'rxjs/operators';
import { of } from 'rxjs/observable/of';

@Injectable()
export class AuthGuard implements CanActivate {

  constructor(private loginService: LoginService, private router: Router) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable<boolean>  {
    return this.loginService.isLoggedIn().pipe(
      map(e => {
        if (e) {
          return true;
        } else {
          ...
        }
      }),
      catchError((err) => {
        this.router.navigate(['/login']);
        return of(false);
      })
    );
  }

}
```

answered May 2 '18 at 21:21

Derek Hill

2,255 1 33 55

It worked, thanks. – Kshitij Tiwari Mar 25 at 9:31

I have 1 more XHR call after isLoggedIn() , and result of XHR is used in 2nd XHR call. How to have 2nd ajax call which will accept for 1st result? The example you gave is pretty easy, can you pls let me know how to use pipe() if I have another ajax too. – Pratik Aug 8 at 3:16

8

You can use the `.map` operator to transform the `Observable<Response>` to `Observable<boolean>` like so:

```
canActivate(){
    return this.http.login().map((res: Response)=>{
       if ( res.status === 200 ) return true;
       return false;
    });
}
```

edited Oct 27 '17 at 13:39

CodyBugstein
6,843 36 124 253

answered Jan 23 '17 at 10:59

mp3por
939 3 16 33

1 what about a catch block? the catch block is called if its a 401 right? – danday74 Mar 9 '17 at 17:13

1 In angular 4 doesn't work. It needs somewhere to define a generic type. – gtzinos Jun 7 '17 at 13:18

2

I've done it in this way:

```
canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable<boolean> {
return this.userService.auth(() => this.router.navigate(['/user/sign-in']));}
```

As you can see I'm sending a fallback function to userService.auth what to do if http call fails.

And in userService I have:

```
import 'rxjs/add/observable/of';

auth(fallback): Observable<boolean> {
return this.http.get(environment.API_URL + '/user/profile', { withCredentials: true })
  .map(() => true).catch(() => {
    fallback();
    return Observable.of(false);
  });}
```

22.7k 19 113 149

81 4

really good answer in terms of the .map() function - really really good :) - did not use the fallback callback - instead i subscribed to the auth Observable in the canActivate method - thanks very much for the idea – danday74 Mar 9 '17 at 17:21

1

This may help you

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { Select } from '@ngxs/store';
import { Observable } from 'rxjs';
import { map, take } from 'rxjs/operators';
import { AuthState } from 'src/app/shared/state';

export const ROLE_SUPER = 'ROLE_SUPER';

@Injectable()
export class AdminGuard implements CanActivate {

 @Select(AuthState.userRole)
  private userRoles$: Observable<string[]>;

  constructor(private router: Router) {}

 /**
   * @description Checks the user role and navigate based on it
   */

 canActivate(): Observable<boolean> {
    return this.userRoles$.pipe(
      take(1),
      map(userRole => {
        console.log(userRole);
        if (!userRole) {
          return false;
        }
        if (userRole.indexOf(ROLE_SUPER) > -1) {
          return true;
        } else {
          this.router.navigate(['/login']);
        }
```

```
  } // canActivate()
} // class
```

answered Jan 31 at 5:47

Kalanka

342 1 9 25

0

How does it work for you without calling subscribe? for me when I call this through my .net API nothing is returning. I have to call subscribe on my auth guard service like this then only it makes actual API call. But since subscribe is async my canActivate guard is not working and user can get into page.

```
AuthGauard service:

canActivate() { this.loginService.isLoggedIn()
      .subscribe(response => {
          if (!response) return false;})
return true;
}

loginService:
    isLoggedIn():Observable<boolean> {
        return this.http.get(ApiResources.LOGON).pipe(map(response => response.ok));
    }
```

answered Sep 23 '18 at 14:15

Paresh Trivedi

1

-1

CanActivate does work with Observable but fails when 2 calls are made like CanActivate:[Guard1, Guard2].
Here if you return an Observable of false from Guard1 then too it will check in Guard2 and allow access to route if Guard2 returns true.
In order to avoid that, Guard1 should return a boolean instead of Observable of boolean.

answered Aug 1 '17 at 17:05

Arjunsingh

-6

in canActivate(), you can return a local boolean property (default to false in your case).

```
private _canActivate: boolean = false;
canActivate() {
  return this._canActivate;
}
```

And then in the result of the LoginService, you can modify the value of that property.

```
//...
this.loginService.login().subscribe(success => this._canActivate = true);
```

answered Jun 22 '16 at 4:33

Nguyễn Việt Trung
218 1 6

you're a genius, sir. – Kien Nguyen Ngoc Apr 12 at 6:52