Angular CanLoad Guard Example

2 Comments / 5 minutes of reading / March 9, 2023

← PreLoading Strategy | Angular Tutorial | Ng-Content & Content Projection in Angular →

In this tutorial, we will show you how to make use of Angular `Canload` Guard with an example. The Angular Route Guards are used to control, whether the user can navigate to or away from a given route. The `canload` guard determines whether a particular lazy loaded child route can be loaded.

The Angular modules can be either eagerly loaded or lazy loaded. By default Angular loads all the modules eagerly. To lazy load a module we need to use the `loadChildren` in the route definition

For Example, the following route loads the `AdminModule` lazily. Angular loads it only when the user navigates to the `admin` route.

```
{path: "admin", loadChildren:'./admin/admin.module#AdminModule'},
```

Sometimes, we want to prevent the loading of the modules for unauthorized users. This is where we use the `CanLoad` Guard.

Table of Contents

CanLoad Guard

The `CanLoad` Guard prevents the loading of the Lazy Loaded Module. We generally use this guard when we do not want to unauthorized user to navigate to any of the routes of the module and also stop then even see the source code of the module.

The Angular provides `canActivate` Guard, which prevents unauthorized user from accessing the route. But it does not stop the module from being downloaded. The user can use the chrome developer console to see the source code. The `CanLoad` Guard prevents the module from being downloaded.

How to use CanLoad Guard

First, we need to create a Angular Service, which implements the `CanLoad` Interface

The service must implement the `canLoad` method. This method must return either true or false. The Angular evaluates the `canLoad` and loads the lazy loaded module only if it returns true.

```
1
2 @Injectable()
3 export class AuthGuardService implements CanLoad {
4   constructor(private router: Router) {
5   }
```

```
6
7   canLoad(route: Route): boolean {
8
9     //determine whether you want to load the module
10    //return true or false
11
12    return true;
13  }
14 }
15
```

Next, we need to register the Service in the Root Module.

```
1
2  @NgModule({
3    declarations: [
4      AppComponent,
5    ],
6    imports: [
7      BrowserModule,
8      AppRoutingModule,
9    ],
10   providers: [AuthGuardService],
11   bootstrap: [AppComponent]
12 })
13 export class AppModule { }
14
```

Finally, we need to add the guards to the lazy loaded routes as shown below. Note that you can create more than one CanLoad guard, each guard runs in the order added.

```
1
2  {path: "admin", loadChildren:'./admin/admin.module#AdminModule', canLoad:[AuthGuardS
3
```

CanLoad Guard Example

Create a new angular app with two modules AdminModule & TestModule. Let us build a CanLoad Guard, which stops AdminModule from loading.

First, we build a `AuthGuardService` which Implements the `CanLoad` Interface as shown below.

In the `canLoad` method check if the `route` is **admin** and return false else return true. In real life application, you can use dependency injection to inject the authentication service and check to see if the user is authorized or not.

```
1
2  import { Injectable }       from '@angular/core';
3  import { CanLoad, Route, Router } from '@angular/router';
4
5  @Injectable()
6  export class AuthGuardService implements CanLoad {
7
8    constructor(private router: Router) {
9    }
10
11   canLoad(route: Route): boolean {
12
13     let url: string = route.path;
14     console.log('Url:'+ url);
15     if (url=='admin') {
16       alert('You are not authorised to visit this page');
17       return false;
18     }
19     return true;
20   }
21 }
22
```

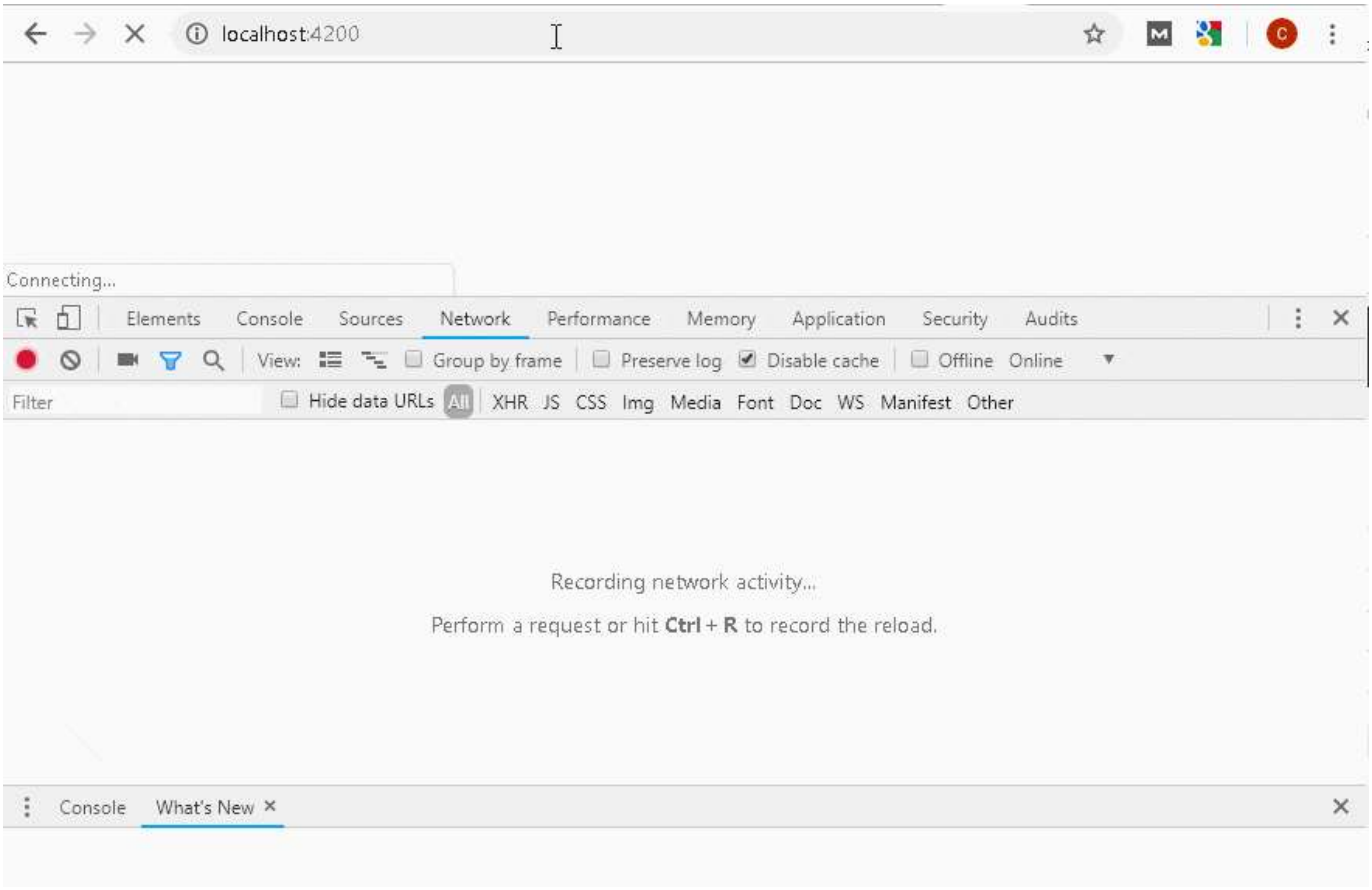
Next, in the route definition include `AuthGuardService` under `canLoad`.

```
1
2  const routes: Routes = [
3    {path: "admin", loadChildren:'./admin/admin.module#AdminModule', canLoad:[AuthGuard
4    {path: "test", loadChildren:'./test/test.module#TestModule', canLoad:[AuthGuardService]}
5  ];
6
```

Finally, register the service in the `AppModule`.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';
import { AuthGuardService } from './auth-gaurd.service';

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
  ],
  providers: [AuthGuardService],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Run the Application. Open the developer console and you will see that only test module is downloaded and not admin module.



Complete Example

app.module.ts

```
1
2  import { BrowserModule } from '@angular/platform-browser';
3  import { NgModule } from '@angular/core';
4
5  import { AppRoutingModule } from './app-routing.module';
6
7  import { AppComponent } from './app.component';
8  import { AuthGuardService } from './auth-gaurd.service';
9
10
11 @NgModule({
12   declarations: [
13     AppComponent,
14   ],
15   imports: [
16     BrowserModule,
17     AppRoutingModule,
18   ],
19   providers: [AuthGuardService],
20   bootstrap: [AppComponent]
```

```
})
export class AppModule { }

```

app.component.ts

```

import { Component } from '@angular/core';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Module Demo';
}

```

app.component,html

```

<ul>
    <li>
      <a class="navbar-brand" routerLink="">home</a>
    </li>
    <li>
      <a class="navbar-brand" routerLink="/admin/dashboard">Admin</a>
    </li>
    <li>
      <a class="navbar-brand" routerLink="/test">Test</a>
  </li>

  </ul>

<h1>Angular CanLoad Guard Example</h1>

<router-outlet></router-outlet>


```

app.component.ts

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333333;
}

li {
    float: left;
}

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 16px;
    text-decoration: none;
}

li a:hover {
    background-color: #111111;
}

```

app.routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AuthGuardService } from './auth-gaurd.service';


const routes: Routes = [
  {path: "admin", loadChildren:'./admin/admin.module#AdminModule', canLoad:[AuthGuar
  {path: "test", loadChildren:'./test/test.module#TestModule', canLoad:[AuthGuardService]
];


@NgModule({
  imports: [RouterModule.forRoot(routes)],
```

```
  exports: [RouterModule]
})
export class AppRoutingModule { }


```

auth-gaurd.service.ts

```
import { Injectable }       from '@angular/core';
import { CanLoad, Route, Router } from '@angular/router';


@Injectable()
export class AuthGuardService implements CanLoad {
  constructor(private router: Router) {
  }


  canLoad(route: Route): boolean {

    let url: string = route.path;
    console.log('Url:'+ url);
    if (url=='admin') {
      alert('You are not authorised to visit this page');
      return false;
    }

    //det

    return true;
  }

}

```

admin/admin.module.ts

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

import { AdminRoutingModule } from './admin.routing.module';
import { DashboardComponent } from './dashboard.component';
```

```
6
7
8  @NgModule({
9    declarations: [DashboardComponent],
10   imports: [
11     AdminRoutingModule,
12   ],
13   providers: [],
14 })
15 export class AdminModule { }
16
```

admin/admin.routing,module.ts

```
1
2  import { NgModule } from '@angular/core';
3  import { Routes, RouterModule } from '@angular/router';
4
5  import { DashboardComponent } from './dashboard.component';
6
7
8
9  const routes: Routes = [
10     { path: 'dashboard', component: DashboardComponent},
11     { path: '', redirectTo:'dashboard'}
12 ];
13
14
15 @NgModule({
16   imports: [RouterModule.forChild(routes)],
17   exports: [RouterModule]
18 })
19 export class AdminRoutingModule { }
20
21
```

admin/dashboard.component.ts

```
1
2   import { Component } from '@angular/core';
3
4   @Component({
5     template: `<h1>Dashboard Component</h1>`,
```

```
  })
  export class DashboardComponent {
    title = '';
  }

```

test/test.module.ts

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

import { TestRoutingModule } from './test.routing.module';
import { TestComponent } from './test.component';


@NgModule({
  declarations: [TestComponent],
  imports: [
    TestRoutingModule,
  ],
  providers: [],
})
export class TestModule { }

```

test/test.routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { TestComponent  } from './test.component';


const routes: Routes = [
    { path: 'list', component: TestComponent},
    { path: '', redirectTo:'list'},
];


@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
```

```
})
export class TestRoutingModule { }
```

test/test.component.ts

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

@Component({
  template: `<h1>Test Component</h1>`,
})
export class TestComponent {
  title = '';
}
```

References

Canload

← PreLoading Strategy | Angular Tutorial | Ng-Content & Content Projection in Angular →

Related Posts