Angular 2 Routing
Part 2

Sang Shin
JPassion.com
"Code with Passion!"

Topics

- Child routes
- Styling
- Guard
  - Introduction
  - Steps for implementing CanActivate guard
  - Steps for implementing CanDeactivate guard

Child Routes

Child Routes

- We want to support routes such as

  http://localhost:4200/user/3/school

  http://localhost:4200/user/3/hobby

Children Routes Configuration

- Create child routes

```
export const childRoutes: Routes = [
    { path: 'school', component: UserSchoolComponent },
    { path: 'hobby', component: UserHobbyComponent }
]
```

- Add children routes to the parent route

```
export const routes: Routes = [
    {path:'home', component: HomeComponent},
    {path:'userlist', component: UserListComponent},
    {path:'user/:id', component: UserDetailComponent},
    {path:'user/:id', component: UserDetailComponent, children: childRoutes},
    {path:'', redirectTo:"home", pathMatch: "full"},
    {path:'**', redirectTo:"home"}
]
```

Router Outlet configuration

- Router outlet needs to be specified in the parent template

```
<p>
  user-detail works!
</p>
 id parameter entered: {{id}}
<br>
<router-outlet></router-outlet>
```

Styling

Styling of Active Link with routerLinkActive

- *routerLinkActive* directive can be used to apply CSS style (in the example below “active”) when the link is active

```
<nav>
  <a [routerLink]="[' ']"
     routerLinkActive="active" [routerLinkActiveOptions]="{exact: true}">Default</a>
  <a [routerLink]="['/home']" routerLinkActive="active">Home</a>
  <a [routerLink]="['/userlist']"
     [queryParams]="{country:'Korea', city:'Seoul'}" routerLinkActive="active">User list</a>
  <br>
  <input type="text" (input)="onInput(id.value)" #id>
  <a [routerLink]="['/user', id.value]" routerLinkActive="active">User Detail</a>
</nav>
<router-outlet></router-outlet>
```

- Style

```
.active {
    color: red
}
```

Guard: Introduction

What is a Guard?

- Why do we want to restrict navigation through routing?
  - Perhaps the user is not authorized to navigate to the target component.
  - Maybe the user must login (authenticate) first.
  - Maybe we should fetch some data before we display the target component.
  - We might want to save pending changes before leaving a component.
  - We might ask the user if it's OK to discard pending changes rather than save them
- We can add guards to our route configuration to handle these scenarios

How does a Guard work?

- A guard's return value controls the router's behavior
  - If it returns *true*, the navigation process continues
  - If it returns *false*, the navigation process stops and the user stays put
- The guard can also tell the router to navigate elsewhere, effectively canceling the current navigation
- A guard performs the guarding operation asynchronously
  - It returns either Observable<boolean> or a Promise<boolean>

Types of Guards

- CanActivate
  - mediate navigation to a route
- CanActivateChild
  - mediate navigation to a child route
- CanDeactivate
  - mediate navigation away from the current route
- Resolve
  - perform route data retrieval before route activation
- Load
  - mediate navigation to a feature module loaded asynchronously

Guard: Steps for Implementing CanActivate Guard

Steps for Implementing a Guard

- Step #1: Create Guard class
- Step #2: Configure the guard for the route to be guarded
- Step #3: Configure the guard as a provider in the module class

Step #1: Create Guard Class

- The guard class needs to implement *CanActivate* interface

```
// This is user-detail.guard.ts
export class UserDetailGuard implements CanActivate {

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
                    Observable<boolean> | Promise<boolean> | boolean {

        return confirm('Are you sure?');

    }
}
```

Step #2: Configure Guard Class for the routes

- Each route can be configured with any number of guards

```
export const routes: Routes = [
    {path:'home', component: HomeComponent},
    {path:'userlist', component: UserListComponent},
    {path:'user/:id', component: UserDetailComponent,
                                canActivate: [UserDetailGuard]},
    {path:'user/:id', component: UserDetailComponent, children: childRoutes},
    {path:'', redirectTo:"home", pathMatch: "full"},
    {path:'**', redirectTo:"home"}
]
```

Step #3: Configure Guard as a Provider

- The guard class needs to be configured as a provider in the module

```
@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    UserListComponent,
    UserDetailComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
    RouterModule.forRoot(routes)
  ],
  providers: [UserDetailGuard],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Guard: Steps for Implementing CanDeactivate Guard

Deactivate Guard

- Unlike Activate guard, Deactivate guard depends on the component from which it is guarding
- Create an interface

```
export interface ComponentDeactivate {
    canDeactivate: () => Observable<boolean> | boolean;
}
```

- and have the component to implement it

```
export class UserDetailComponent implements ComponentDeactivate {
 ...

  canDeactivate(): Observable<boolean>| boolean {
    return confirm('Are you sure?');
  }

}
```

Step #1: Create Guard Class

- The guard class needs to implement *CanDeactivate* interface

```
// This is user-detail.guard.ts
export class UserDetailGuard implements CanActivate,
                                        CanDeactivate<ComponentDeactivate> {

   canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable<boolean> | boolean {
      return confirm('Are you sure?');
   }


   canDeactivate(component: ComponentDeactivate,
                  route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
         Observable<boolean> | boolean {
            return component.canDeactivate();
   }

}
```

Step #2: Configure Guard Class for the routes

- Each route can be configured with any number of guards

```
export const routes: Routes = [
    {path:'home', component: HomeComponent},
    {path:'userlist', component: UserListComponent},
    {path:'user/:id', component: UserDetailComponent,
                                canActivate: [UserDetailGuard],
                                canDeactivate: [UserDetailGuard]},
    {path:'user/:id', component: UserDetailComponent, children: childRoutes},
    {path:'', redirectTo:"home", pathMatch: "full"},
    {path:'**', redirectTo:"home"}
]
```

Code with Passion!
JPassion.com