Guarding Routes & Guard Interfaces

Matthew R. Howard

Bellevue University

**Guarding Routes & Guard Interfaces**

Route Guards prevent unauthorized and unauthenticated users from navigating anywhere they’d like in the application (Fain, Moiseev, 2017). The return value of the guard control’s it’s behavior. If the value returns as true, the navigation process can continue. If it returns false, the navigation process will either stop or be re-directed elsewhere (angular.io, 2018). The guard cannot answer synchronously. However, it can return an Observable<boolean> or a Promise<boolean> which will have the router wait for a true or false observable (angular.io, 2018). A couple of examples of route guard interfaces include: (angular.io, 2018)

* CanActivateChild() = mediates navigation to a child route.
* CanDeactivate() = mediates navigation away from the current route.

CanActivateChild() guard protects child routes and is ran before any child routes are activated. And with the AuthGuard implemented, it also protects when navigating between the admin routes. CanActivateChild() is implemented by opening the auth-guard.service.ts and then add the CanActivateChild() method (angular.io, 2018). Additionally, the CanActivateChild() method can return Observable<Boolean> or Promise<boolean> for asynchronous checks and a boolean for synchronous checks. you can implement CanDeactivate() to stop the navigation while you wait for the server to return with its answer. Doing this allows you to accumulate user changes, validate on the server, or hold changes in a pending state (angular.io, 2018).

References

angular.io (n.d.). Router: Routes & Navigation. Retrieved July 23, 2018, from https://angular.io/guide/router

Fain, Y., Moiseev, A. (2017). Angular 2 with TypeScript, Manning Publications Co., Shelter Island, NY