Requesting an Order History on Amazon

Slide Note

Is This What You See?

Without logging in, the users are not allowed to view the order history and the app navigates them to the log-in screen to log in.

The login procedure is necessary so that the app can identify the user and appropriately fetch the information about the order history.

Slide Note

Deleting Git Repositories

Slide Note

Deletion of Git or Angular repository is allowed to the users who are either the owners or have permissions to delete

Think and Tell

- Can you delete the Git repository that you created on my repos?
- Can you delete the Angular repository on GitHub?

Slide Note

Exiting Web IDE on Git Without Saving Changes

Menu

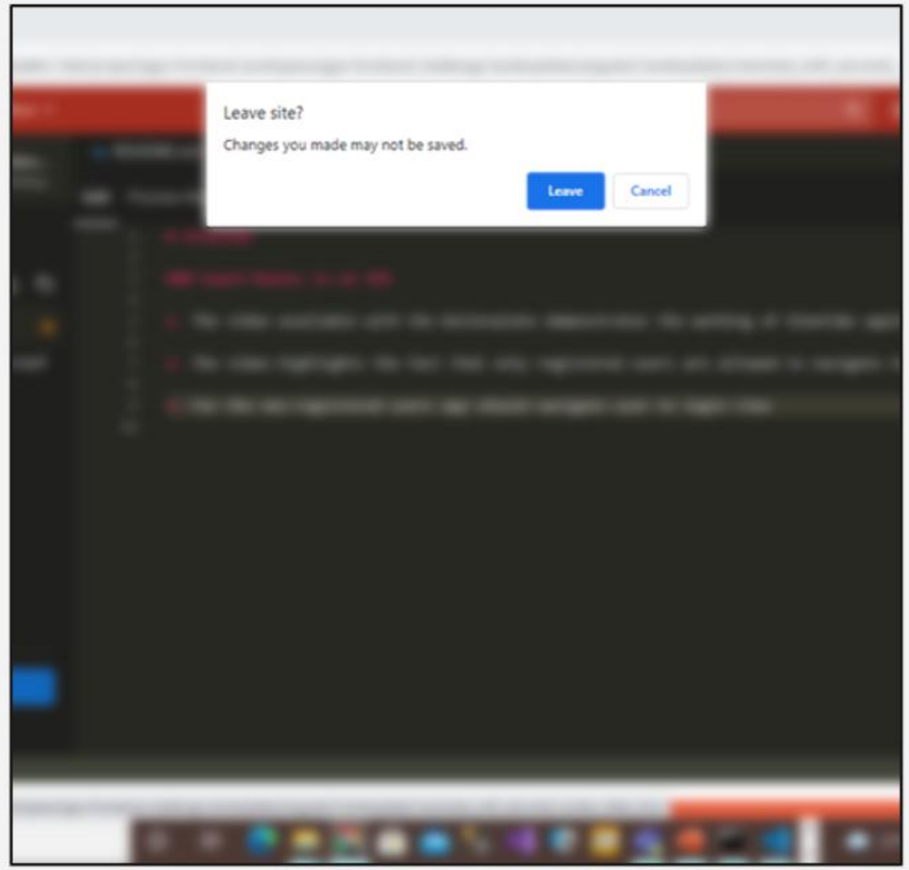
00:06:00:00:36

06/ 36

Slide Note

Exiting a Web Page

- Often, when users leave the web pages without saving changes, the pages raise an alert seeking confirmation from the users, whether to leave without saving the changes.



Slide Note

If the application does not address these concerns, it will impact the user's trust in the application. It will also result in a poor UX.

Think and Tell

- Should Amazon allow users to navigate to their order history without logging in?
- Should authenticated but unauthorized users be provided access to delete any of the repositories on GitHub?
- Should users be allowed to leave the web IDE view on Git without prompting them to save the unsaved changes?

Slide Note

Guard Routes in an SPA

Slide Note

Learning Objectives

- Control access to application parts using route guards
- Configure routes to add a route guard
- Add a guard to control route activation
- Add a guard to control route deactivation

Slide Note

How are parts of an SPA accessed?

Slide Note

Accessing Parts of an SPA

Routing in an SPA allows a user to navigate and access different parts of an application.

- Routing happens when:
  - A user clicks the navigation links to visit different views in SPA.
    - For example, the user clicks the log-in link and gets redirected to the log-in view.
  - The application performs an action, and the user is navigated to another view as a response.
    - For example, after a successful login, the user is navigated to the user profile view.

Slide Note

Should all the parts of an SPA always be accessible?

Slide Note

Navigational routes should be guarded to prevent unintended access to the restricted views of an SPA.

Slide Note

Why Should Access to Parts of an SPA Be Controlled?

- To protect parts of an application from unauthenticated or unauthorized access.
- To help ensure data is available before a part of the application is accessed.
- To allow the users to save the unsaved changes before exiting a component.
- To request confirmation from the user before leaving the component.

Slide Note

How can you control access to parts of an SPA?

Slide Note

Controlling Access Using Route Guards

- Angular allows you to control access to different parts of an application by using guards.
- Guards are added to the route configuration in the router module.
- The return value from guards helps control the route's behavior.
- If the returned value is:
  - True, the navigation process continues.
  - False, the navigation process stops, and the user stays on the current view.
  - `UrlTree`, the current navigation is canceled, and a new navigation process is initiated from the returned `UrlTree`.

Slide Note

Does the route guard return value synchronously or asynchronously?

Slide Note

Guard Return – Synchronous or Asynchronous

- The return from guards can be synchronous as well as asynchronous.
- If the value to return is available, the return is synchronous.
- However, at times, the guard must wait for a response from the user or a fresh set of data to be received from the server. In these situations, the return is asynchronous.
- To handle asynchronous returns, guards can return one of the following types of responses:
  - Observable<boolean>
  - Promise<boolean>

Slide Note

Does the route guard return value synchronously or asynchronously?

Slide Note

Guard Return – Synchronous or Asynchronous

- The return from guards can be synchronous as well as asynchronous.
- If the value to return is available, the return is synchronous.
- However, at times, the guard must wait for a response from the user or a fresh set of data to be received from the server. In these situations, the return is asynchronous.
- To handle asynchronous returns, guards can return one of the following types of responses:
  - Observable<boolean>
  - Promise<boolean>

Slide Note

Implementing Route Guards in an Angular SPA

Slide Note

Implementing Route Guards

- Guards in Angular are injectable classes that implement guard interfaces.
- The router supports multiple guard interfaces:
  - **`CanActivate`**: to mediate navigation to a route
  - **`CanActivateChild`**: to mediate navigation to a child route
  - **`CanDeactivate`**: to mediate navigation away from the current route
  - **`CanLoad`**: to mediate navigation to a feature module loaded asynchronously

Slide Note

Adding a Guard to Control Route Activation

Slide Note

Implementing `CanActivate` Route Guard

- In an SPA, that contains multiple views, to protect the views, access to the views should be restricted.
  - The access should be provided only to the authenticated users.
  - For example, in FruitFantasy app, only the site administrators should be allowed to add new fruit details.
  - All other users should be denied access to the add-fruit view.
- This functionality can be implanted using the `CanActivate` route guard.
- The `CanActivate` interface is a route guard that can be used by the router to decide if a route can activated.

Slide Note

Prevent Unauthenticated Access in Fruit-Fantasy App

The Fruit-Fantasy application allows the users to add, view, modify, and delete fruit details.

The application should ensure that these views are accessible only to the user with a valid vendor code. If an unauthenticated user attempts to navigate to these views, the user should be redirected to the log-in view.

Click here for the demo solution.

DEMO

Slide Note

Quick Check

What are the different types of value that can be returned by the `canActivate()` guard method? method?

1. `boolean`
2. `Observable<boolean>`
3. `Promise<boolean>`
4. `UrlTree`

Slide Note

Quick Check: Solution

What are the different types of value that can be returned by the `canActivate()` guard method? method?

1. `boolean`
2. `Observable<boolean>`
3. `Promise<boolean>`
4. `UrlTree`

Slide Note

Adding Guard to Prevent Navigation Away From a Component

Slide Note

Implementing `CanDeactivate` Route Guard

- In an SPA, there can be views that accept inputs from users.
  - For example, in FruitFantasy app, there are views that allow users to add or edit fruit details.
- Users may leave the views without saving the changes.
- As a best practice, the app should let the user know that there are unsaved changes and take confirmation from the user to discard changes before the user navigates away from this view.
  - This functionality can be implanted using the `CanDeactivate` route guard.
- The `CanDeactivate` interface is a route guard that can be used by the router to decide if a route can deactivated.

Preventing the Loss of Unsaved Changes

The Fruit-Fantasy app allows vendors to update fruit details.

The application should ensure that navigation to any other component is not permitted if the changes are not saved upon update.

Click here for the demo solution.

DEMO

Slide Note

Quick Check

The _____ parameter is passed to the canDeactivate() method of the CanDeactivate guard so that the method can be called on the component on which the guard is applied.

1. Generic component
2. ActivatedRouteSnapshot
3. RouterStateSnapshot
4. Activated Route

Slide Note

Quick Check: Solution

The _____ parameter is passed to the `canDeactivate()` method of the `CanDeactivate` guard so that the method can be called on the component on which the guard is applied.

1. `Generic component`
2. `ActivatedRouteSnapshot`
3. `RouterStateSnapshot`
4. `Activated Route`

Slide Note

With multiple guards implemented, what order do we use to execute the guard methods?

Slide Note

Guard Methods: Order of Execution

- Multiple guards can exist at every level of the routing hierarchy.
- The order of execution is as follows:
  - `CanDeactivate`:
    - The router checks these guards first, from the deepest child route to the top.
  - `CanActivate` and `CanActivateChild`:
    - The router checks these from the top down to the deepest child route.
  - `CanLoad`:
    - If the feature module is loaded asynchronously, the router checks for this guard before the module is loaded.
- If any of the guards return false, the execution of the pending guards is cancelled, and the entire navigation is cancelled.

Slide Note

Quick Check

If the guards CanActivate and CanActivateChild are on the same route, which guard method gets invoked first, canActivate() or canActivateChild()?

1. First `canActivate()` and then `canActivateChild()`

2. First `canActivateChild()` and then `canActivate()`

Slide Note

Quick Check: Solution

If the guards CanActivate and CanActivateChild are on the same route, which guard method gets invoked first, canActivate() or canActivateChild()?

1. **First `canActivate()` and then `canActivateChild()`**
2. First `canActivateChild()` and then `canActivate()`

Explanation:
The router checks the `CanActivate` and `CanActivateChild` guards from the top down to the deepest child route.