Navigation in a Multi-Page Application

Landing Page

http://eshop.com/index.html

Login Page

http://eshop.com/login.html

Cart Page

http://eshop.com/mycart.html

- Navigation takes place between the web pages.
- Every navigation causes a complete webpage reload that includes:
  - html + css + js + images + ... (all resources)

Slide Note

Note the headings on images, the word "Page" has been replaced with the word "View"

Since there is One Page with Multiple Views.

Navigation in a Single-Page Application

Landing View

http://eshop.com/index.html

Login View

http://eshop.com/login.html

Cart View

http://eshop.com/mycart.html

- Navigation takes place between the views.
- Every navigation causes only the view to change with the new set of data without causing a complete page reload.

Slide Note

Think and Tell

How does the user navigate from one view to the other in an SPA?

How is data passed during navigation from one view to the other?

Slide Note

Implement Navigation Using Angular Routing

Slide Note

Learning Objectives

- Explain navigation in a single-page application
- Enable routing with basic routes in an Angular Application
  - Define routes in an Angular application using the `Routes` array.
  - Add routes to the Angular application
- Access route information using `ActivatedRoute` interface
- Setup wildcard routes and redirects in route configuration
- Explain the significance of Route order
- Consume `Router` service to enable programmatic navigation between views

Slide Note

Refer to the instructions given in the .md files for the demo instructions.

Navigation in a Multi-Page Application

A multi-page application is developed to allow the user to manage notes.

Routing is implemented in the application for navigating between the pages.

Observe the changes recorded in the Network tab of Developer tools when navigation takes place from one page to another.

Click here for the demo solution.

DEMO

Slide Note

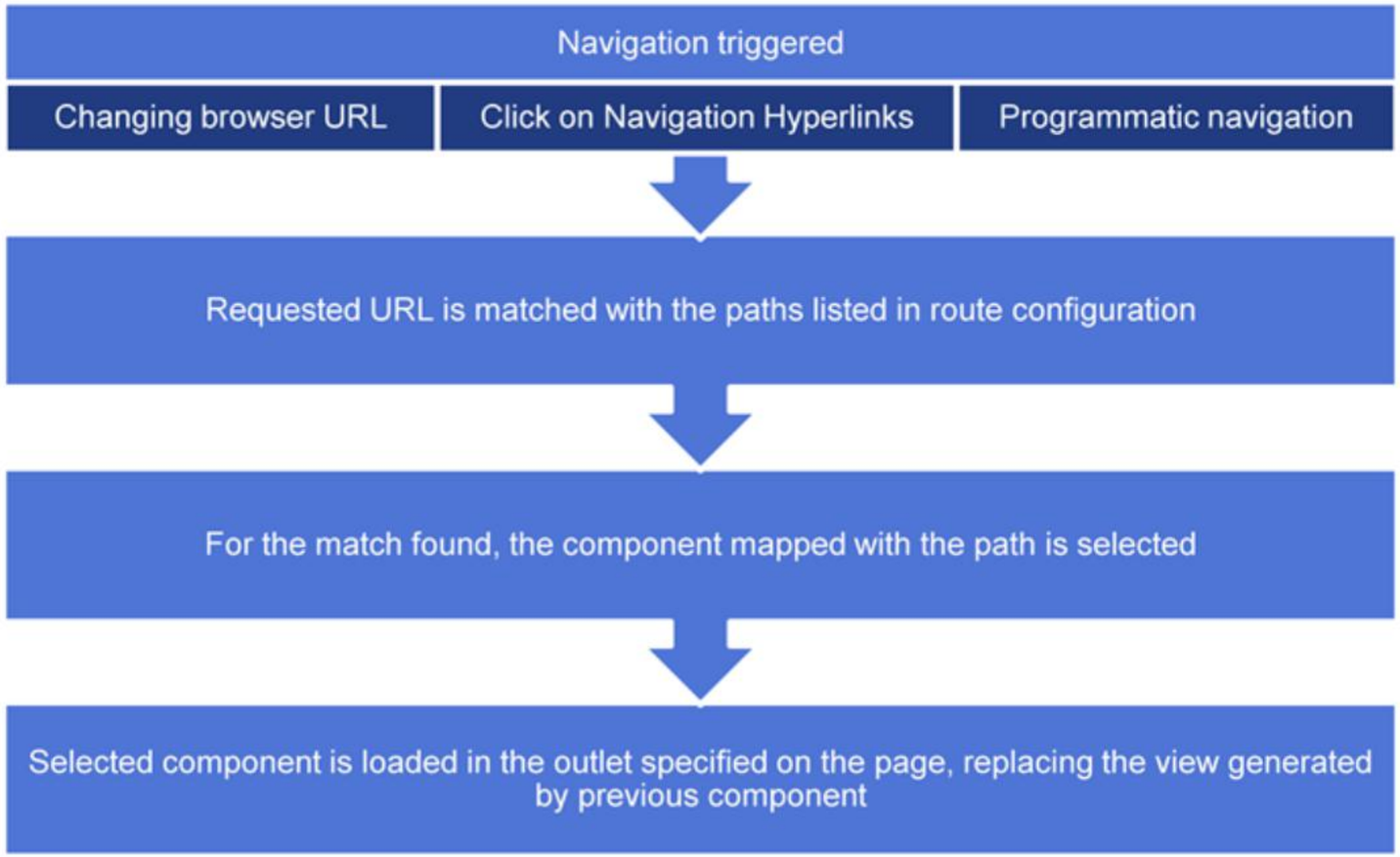
Navigation in a Single-Page Application

- In an SPA, navigation occurs among the views and not the pages.
- In an Angular application, the Router service provided by the framework handles navigation from one view to the other.
- Every time the URL in the browser changes, the Angular Router interprets the browser URL and triggers navigation.
- A navigable Angular SPA typically contains a route definition that provides mapping information between the component to be navigated to and the route URL.
- Upon navigation, the component is dynamically loaded at the location specified.
  - This location is considered an outlet for loading components.

Slide Note

During navigation, the previously loaded component is destroyed and the target component is dynamically loaded.

How Does Navigation Happen in an Angular Application?



Slide Note

Enabling Navigation in the Angular Application

- Following are the key requirements that need to be fulfilled while enabling routing in the Angular application:
  - Define Route Configurations:
    - Route configurations provide mappings between the route path and the component that needs to be loaded for the specified URL.
  - Specify the outlet for the routed component:
    - The routing components are dynamically loaded.
    - The physical location needs to be specified for the dynamically loaded components.
  - Determine the trigger for navigation:
    - Navigation is triggered by either of these user interactions:
      - By clicking on hyperlinks or by typing the URL in a browser.
    - As soon as some tasks are completed, the navigation is triggered internally by the application:
      - For example, navigating to the welcome view after successful login.

Slide Note

Common Routing Tasks

- What are the steps that are commonly performed while enabling routing in Angular?

  1. Create application with routing enabled:

     A. The `ng new` command prompts the developer to specify whether the Routing module should be added to the application or not.

     B. For creating a route-enabled application, the prompt should be responded with a “Y”

     Alternatively, in the existing Angular application:

     I. A new module can be added to configure routes.

        i. This module should be imported by the `App Module`.

     II. Or configure the routes in the existing `App Module`.

  2. Define routes.
  3. Add routes to application.
  4. Retrieve route information.

Slide Note

Enable Routing in the Fruit-Fantasy App

Add basic routes to `Fruit-Fantasy` app.

The app should have `home` links that navigate the viewers to the `Fruit-Manager` component.

Click here for the demo solution.

DEMO

Slide Note

Using Angular Router

Navigation in the Angular application is implemented with the help of the following key components of the `@angular/router` package.

Routing Module

- It is an `NgModule` that provides service and directives to the application.
- The `forRoot()` method of `RouterModule` is called to create `NgModule` with directives, route configurations, and `Router` service.

`Router Service`

- This service helps to manage navigations among views and URL manipulations.

`RouterLink` Directive

- This is an attribute directive that transforms the host element into the link element.
- The value of this directive is the path to the component that needs to be navigated.

`RouterOutlet` Directive

- Specifies location on the page for loading the navigated component dynamically.

Slide Note

Both `RouterModule` and `Routes` are imported from `@angular/router` package.

Configuring Routes

- The Route configuration can be written in the App module.
  - It can be written in a separate module and this module can be imported in the App module.
- Each Route configuration is an object of type `Route`.
- The simplest configuration requires the use of `path` and `component` properties of `Route` object to create mapping between URL and component.
  - `path` property contains the URL value as a string
  - `component` property contains the type of component as its value
- For a list of Route configurations use type `Routes` which is an array of type `Route`.

```
import { NgModule } from "@angular/core";
import { RouterModule, Routes } from "@angular/router";
import { FruitManagerComponent } from "./fruit-
manager/fruit-manager.component";

const routes: Routes = [{
    path: "home",
    component: FruitManagerComponent
}]

@NgModule({
    imports: [
        RouterModule.forRoot(routes)
    ],
    exports: [RouterModule]
})
export class AppRoutingModule {

}
```

Slide Note

Add Routes to the Application

- Assign the anchor tag to add the route to the `routerLink` attribute.
  - Set the value of the attribute to the path, which is mapped to the component that needs to be loaded when the user clicks on the link.
- Use `<router-outlet>` directive to update the component template.
  - This directive helps specify the location for rendering the routed component on the page.

```
<a routerLink="../home" mat-list-item>Home</a>
```

In Navigation-Panel

```
<div id="icons">
    <a routerLink="../home">
            <mat-icon>home</mat-icon>
    </a>
    <mat-icon>settings</mat-icon>
</div>
```

In Header

```
<router-outlet></router-outlet>
```

In Navigation-Panel

Slide Note

Quick Check

CLI imports a/an _____ into the AppModule that contains route configurations when creating a new Angular application with the routing option.

1. RoutingModule
2. AppRouterModule
3. AppRoutingModule
4. AppRouteModule

Slide Note

Quick Check: Solution

CLI imports a/an _____ into the AppModule that contains route configurations when creating a new Angular application with the routing option.

1. RoutingModule
2. AppRouterModule
3. **AppRoutingModule**
4. AppRouteModule

Slide Note

Enhancing the Fruit-Fantasy App

The current Fruit-Fantasy app displays a list of fruits available on the server. The app allows users to search for fruit or add new fruit.

Fruit details must now be modifiable, according to a new requirement. It is suggested that whenever a user clicks on a specific fruit card, the page view should be replaced by another view showing the existing fruit details and allowing those details to be modified.

So the app now needs multiple views (home and fruit-detail), and one of them should be visible depending on the user's request.

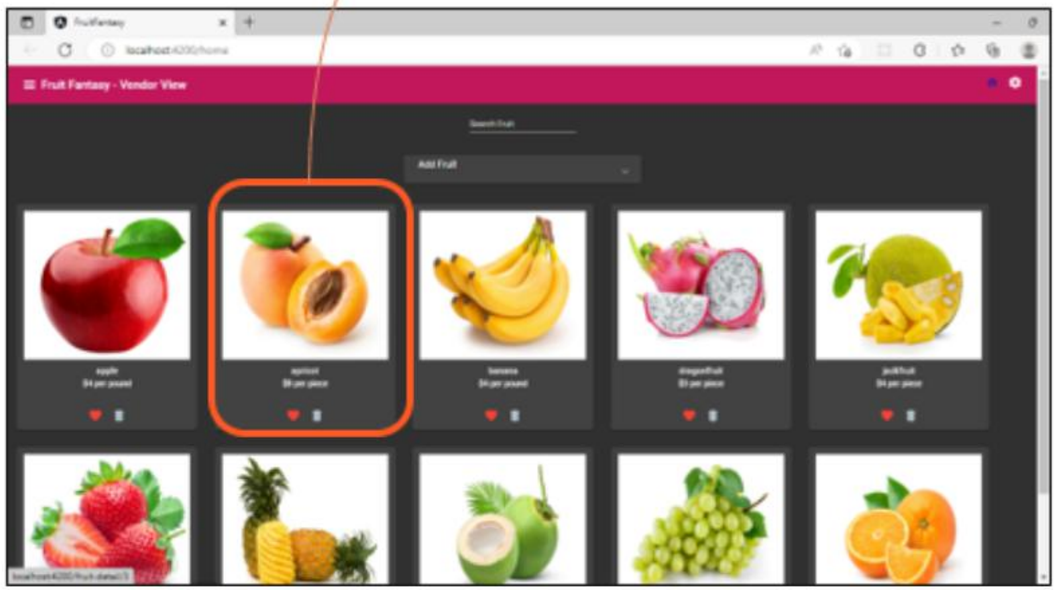
Also, an additional requirement needs to be fulfilled to enable routing in the app. The app needs to know which component has to be loaded and when:

- On page launch, the fruit-manager component should be loaded.
- When a user clicks on fruit, the fruit-detail component should be loaded.

Slide Note

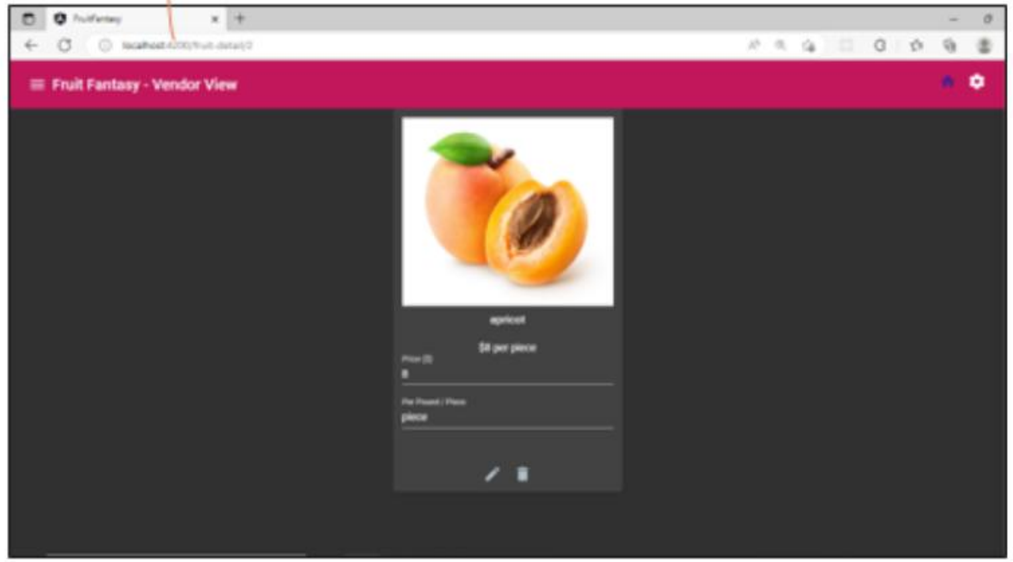
Passing Information through Routes

User selects fruit with id : 2



Home view showing fruit catalog

*http://localhost:4200/fruit-detail/**2***



Fruit Detail view showing enlarged image of the selected fruit

- How did the Detail View know which fruit was selected in the Home View?
- The URL of the Detail View shows the `id` of the fruit selected in the Home View
- How did the Home View communicate to the Detail View, the id of the selected Fruit?

Slide Note

Create a View to Edit a Selected Fruit

Enhance the Fruit-Fantasy app to allow users to select a fruit and modify its details. The details of the selected fruit should be displayed in a separate component.

Define a route with a parameter, where the parameter value is the `id` of the selected fruit.

The component displaying details of the selected fruit should read the id from the route and fetch the details of the fruit for this id.

Click here for the demo solution.

DEMO

Slide Note

Retrieve Route Information

- Information for application components can be passed through the route.
- The `ActivatedRoute` interface is used to retrieve this information.
- The route path and the parameters are available through an injected router service called `ActivatedRoute`.
- The route parameter data can be retrieved by subscribing to observable property `paramMap` of `ActivatedRoute`.

```
ngOnInit(): void {
  this.activatedRoute.paramMap.subscribe(params => {
    let id = params.get("id") ?? 0;
    this.fruitService.getFruit(+id).subscribe(data => {
      this.fruit = data;
    })
  });
}
```

Slide Note

Quick Check

How would you retrieve the route parameter using ActivatedRoute?

1. By subscribing to the param property of the ActivatedRoute
2. By subscribing to the query property of the ActivatedRoute
3. By subscribing to the queryParam property of the ActivatedRoute
4. By subscribing to the paramMap property of the ActivatedRoute

Slide Note

Quick Check: Solution

How would you retrieve the route parameter using ActivatedRoute?

1. By subscribing to the param property of the ActivatedRoute
2. By subscribing to the query property of the ActivatedRoute
3. By subscribing to the queryParam property of the ActivatedRoute
4. **By subscribing to the paramMap property of the ActivatedRoute**

Slide Note

Routes in Fruit-Fantasy App

- Following are the existing routes defined in the Fruit-Fantasy app:
  - /home
  - /fruit-detail/:id
- What will happen if the user types the path `` `http://localhost:4200/fruit-details` `` in the browser's address bar?
  - Can the route be configured to handle invalid routes?

Slide Note

Setting up Wild Card Routes

- An invalid route URL leads to error 404 (Not Found).
- Setting up wild card routes helps in handling this error.
- Angular recognizes the route in the router array as the wild card route if the value of the path is **.
- This route should be the last route in the router array.
- The router selects this route if none of the previous routes match the specified route URL.

```
const routes: Routes = [
 {
 path: "home",
 component: NavbarComponent,
 },
 {
  path: "login",
  component: LoginComponent
 },
 {
 path: "**",
 component: NotFoundComponent
}
```

Slide Note

Think and Tell

- As the application launches, should the user type the URL for home view?
- Or, should the home view by default be made available to the users?

Slide Note

Setting up Redirects

- When the application launches, the component rendering the landing view or the home view should by default be loaded.
- The route configuration would have a path to this home view component.
- Additionally, in the route configuration, a route configuration should be added that defaults to the home view.
- This is achieved with the help of `redirectTo` property and `pathMatch` property.
- This redirect should precede the wild-card route.

```
const routes: Routes = [{
    path: "home",
    component: FruitManagerComponent
},
{
    path: "fruit-detail/:id",
    component: FruitDetailComponent
},
{
    path: "",
    redirectTo: "/home",
    pathMatch: "full"
},
{
    path: "**",
    component: NotFoundComponent
}]
```

Slide Note

Configure Wild Card Routes and Route Redirects in the Fruit-Fantasy App

For the invalid route URLs, define the route in the Fruit-Fantasy app that navigates the user to the page that shows the error message.

Also, define the route in the Fruit-Fantasy app that redirects users to the home view of `Fruit-Fantasy` app as soon as the app is launched in browser. (without providing `\home` in the route URL).

Click here for the demo solution.

DEMO

Slide Note

The LoginComponent will never load since the path login will match wild card characters, so the app will navigate to NotFoundComponent

Significance of Route Order

- The router uses the first-match-wins strategy when matching routes.
- More specific routes should be placed above the less specific ones.
- List routes with a static path first, followed by an empty path route, that matches the default route.
- The wild card route comes last because it matches every URL.

Incorrect Route Order (logical error)

```
const routes: Routes = [
 {
path: "home",
component: NavbarComponent,
},
{
 path: "**",
 component: NotFoundComponent
},

{
 path: "login",
 component: LoginComponent
}
```

Correct Route Order

```
const routes: Routes = [
 {
path: "home",
component: NavbarComponent,
},
{
 path: "login",
 component: LoginComponent
},

{
 path: "**",
 component: NotFoundComponent
}
```

Slide Note

Quick Check

Which directive is used to provide the location placeholder to Angular to dynamically load the routed Angular component?

1. Router-Outlet
2. Outlet
3. RouterOutlet
4. RouterOutletDirective

Slide Note

Quick Check

Which directive is used to provide the location placeholder to Angular to dynamically load the routed Angular component?

1. Router-Outlet
2. Outlet
3. **RouterOutlet**
4. RouterOutletDirective

Slide Note

Revisiting Fruit-Fantasy App

- The user launches the app, and the app launches the home view
- User selects a particular fruit image, and the app navigates to fruit-detail view and shows the enlarged view of the image selected
- Let's say user has edited the fruit details.
- Now, how can the user navigate back to the landing view?
  - Currently, he needs to type the URL of the home view in the browser's address bar.
  - How would a user know what and where he has to write to navigate back to home view?
- Can the app automatically navigate the users to the home view once the edit operation has successfully completed?

Slide Note

Enable Navigation From Fruit-Detail to Home View

Update the solution code of the `Fruit-Fantasy` app to allow the app to navigate back to the home view after the user has edited the fruit details in the `Fruit-Detail` component.

Click here for the demo solution.

DEMO

Slide Note

Programmatic Navigation

- Programmatic Navigation in Angular is executed using the Angular `Router` service.
- The Angular Router service class provides the navigation methods that accept the navigation paths and perform navigation.
- The Angular Router service provides two methods for navigation:
  - `navigateByUrl()`
    - This method accepts an absolute path and requests navigation to a view.

```
router.navigateByUrl('categories/10/product/1001');
```

  - `navigate()`
    - This method accepts an array of commands and requests navigation.

```
router.navigate(['categories', 10, 'product', 1001]);
```