Day4

Thursday, September 1, 2022 9:18 AM

Angular Routing

How we navigate in SPA's
- Switching the content on the page

In Angular you must implement/provide the RouterModule

The Router defines routes for our SPA based on URL's

The RouterModule creates an app-router component, and we also include all the routes inside of the app.routing.module

To create the router: ng g m app-routing --flat --module=app

Angular Services and Dependency Injection

Services are used to organize and share business logic, models, data, or functions with different components

Each service is a singleton, we can inject these into multiple components

These services are just classes in angular
- denoted with .service.ts

Create these with the CLI using ng g s service-name

Angular uses constructor dependency injection

The framework uses an injector where all injectible classes are registered
- Responsible for creating instances of services
- Responsible for injecting services into components

The injector is registered with NgModule

The services will use the @Injectible decorator

PubSub Design Pattern

Publisher Subscriber design pattern which describes the flow of messages between applications, or devices, or components

A message (info) is published to a channel, any consumers will subscribe to the channel
- Useful for async workflows

HttpClient: service used to communicate via HTTP to servers

It is injected into classes, and its included when you create a new app

Need to declare the HttpClientModule in the app.module

Instead of promises HttpClient uses pub/sub with the use of observables

Handling Errors with HttpClient

We will use an Observable .pipe() with the catchError() function where every we making the request

RxJS Observables

Reactive Extension for Javascript
- framework for JS to help Asyc programming

Observables provide support for passing messages

- framework for JS to help Asyc programming

Observables provide support for passing messages between parts of your application, and event handling, asyn programming, and handling multiple values

Define a function for publishing values, function is then called when the consumer wants to subscribe to the data

The subscriber will recieve notifications about new data until they unsubscribe

To subscribe to data you use .subscribe()
To unsubscrib use .unsubscribe()

The observable also has three callback methods:
- .next() called when a new value arrives
- .error() called if an error occurs
- .complete() called when the stream is over

Promises vs Observables
- Promise will only emit a single peice of data
- Observable will emit data to a stream until complete

Subjects: special Observables which allow multicasting to many Observables

Will have all the methods of an observable, however slightly different implementation

The subject .next() will multicast the next value of the stream to multiple observers subscribed to it

Three types of subjects

Behavior: will temporarily store a value/values until observers subscribe

Relay: provides an option to choose how many values we can emit from the last observer
- It stores and passes the last specified values to the new observer

Asyc: emit the last value to observables when the sequence is complete only executed when complete is called

Angular Pipes:

Provide a way to transform values in an Angular template, use the pipe symbol and the name of the pipe

Built in pipes include
- Date pipe
- Currency Pipe
- Async used to placehold while we wait for data

Custom Pipes

Use the CLI ng g pipe pipe-name

pipe.ts and pipe.spec.ts

In the pipe.ts file you implement the transform method which returns the transformed data you want to display

Route Guards: used to check if a user is allowed to load a certain Route

4 Different Route Guards

CanActivate: can the route be activated
CanActivateChild: can you activate children of that route
CanLoad: can the route be loaded

CanActivate: can the route be activated
CanActivateChild: can you activate children of that route
CanLoad: can the route be loaded
CanDeactivate: can the user leave the route

To create a route guard use ng g guard name
- implement the inherited method