Tuesday: Passing Parameters

Passing Parameters

We can make one more change in our application. We can have the details of a goal displayed separately. This will help us learn how to pass parameters in a route. Let's create a route that maps to a specific goal:

src/app/app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { GoalComponent } from './goal/goal.component';
import { AboutComponent } from './about/about.component';
import { NotFoundComponent } from './not-found/not-found.component';
import { GoalDetailComponent } from './goal-detail/goal-detail.component';

const routes: Routes = [
  { path: 'goals', component: GoalComponent},
  { path: 'about', component: AboutComponent},
  { path: 'goals/:id', component: GoalDetailComponent },
  { path: '', redirectTo:"/goals", pathMatch:"full"},
  { path:'**', component:NotFoundComponent},
];

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

We have added a route with the path goals/:id which displays the GoalDetailComponent. The :id token allows us to pass a parameter in the route. We will pass in the id of a goal to retrieve a specific goal. We'll also need to make some changes in our templates.

src/app/goal/goal.component.html

```
<div class="container">
  <ng-progress></ng-progress>
  <h1>MY GOALS</h1>
  <hr>

  <div class="row">
    <div class="col-md-6">
      <div *ngFor='let goal of goals;let i = index'>
        <div>
          <h6 id={{i}} appStrikethrough>{{goal.name}} due on {{goal.completeDate|date|uppercase}}</
h6>
          <button class="btn btn-primary" (click)='goToUrl(goal.id)'>View Details</button>
          <button class="btn btn-outline-danger" (click)='deleteGoal(i)'>Delete Goal</button>
        </div>
      </div>
    </div>
    <div class="col-md-6">
      <app-goal-form (addGoal)="addNewGoal($event)"></app-goal-form>
    </div>
  </div>
  <p *ngIf='goals.length > 5'>You have too many goals</p>
  <div class="row">
    <blockquote class="blockquote text-center">
      <p class="mb-0">{{quote.quote}}</p>
      <footer class="blockquote-footer"><cite>{{quote.author}}</cite></footer>
    </blockquote>
  </div>
</div>
```

Notice we have gotten rid of the toggle Details button and replaced it with a view Details button and next to it a Delete Goal button. We have also gotten rid of the div that was displaying the Goal Detail component. The View Details button has a click event listener which calls the goToUrl() function that takes in the goal id that is responsible for triggering the navigation to the GoalDetails component of a specific goal. The Delete Goal button also has a click event listener that calls the deleteGoal() function which deletes a goal from the goals array.

Let's create these functions that we have added to the GoalComponent.

src/app/goal/goal.component.ts

```
...
import { Router } from '@angular/router';
...
export class GoalComponent implements OnInit {

goToUrl(id){
    this.router.navigate(['/goals',id])
  }

  deleteGoal(index){
    let toDelete = confirm(`Are you sure you want to delete ${this.goals[index].name}`)

    if (toDelete){
      this.goals.splice(index,1)
      this.alertService.alertMe("Goal has been deleted")
    }
  }
...

  constructor(goalService:GoalService, alertService:AlertService, private quoteService:QuoteRequest
Service, private router:Router) {
    this.goals = goalService.getGoals()
    this.alertService = alertService;
  }
...
}
```

At the top, we have imported the Router module and injected it into the constructor with the router property. We have then created the goToUrl() function with an id as an argument. We have used the router's navigate function and passed an array that has the first part of the arguments as the path to the goals and the second part being the id of the goal. We have then kept the deleteGoal() function which deletes a goal from the array of goals.

Our buttons are too close to each other. Let's add some CSS to create some space between them.

src/app/goal/goal.component.css

```
.btn{
  margin:2px;
}
```

We have added a margin to the elements in our component that have the btn class. Remember that this style will only apply for those elements in the goal component. If we wanted to make this style global, we would put it in the app.component.css file instead.

Creating a getGoal method

Since our goal service is still delivering our goals, let's create a method in the service that will help us retrieve one goal.

src/app/goal-service/goal.service.ts

```
import { Injectable } from '@angular/core';
import { Goals } from '../goals';

@Injectable({
  providedIn: 'root'
})
export class GoalService {

  getGoals(){
    return Goals
  }

  getGoal(id){
    for (let goal of Goals){
      if (goal.id == id){
        return goal;
      }
    }
  }

  constructor() { }
}
```

We have created a getGoal() method that takes in the id of a goal. We have then created a for-loop that checks whether the id of a goal is the same as the id parameter passed in, and if it is, then that specific goal is returned and we break out of that loop.

It will be the GoalDetail Component that will be displaying the specific goal so let's get it ready for that.

src/app/goal-detail/goal-detail.component.ts

```
...
import { Goal } from '../goal';
import {  ActivatedRoute, ParamMap } from '@angular/router';
import { GoalService } from '../goal-service/goal.service';
...
export class GoalDetailComponent implements OnInit {

  goal:Goal;

  constructor(private route:ActivatedRoute, private service:GoalService) { }

  ngOnInit() {
    let id = this.route.snapshot.paramMap.get('id');
    this.goal = this.service.getGoal(id)
  }

}
```

At the top, we have imported GoalService, ActivatedRoute to retrieve parameters from the route and ParamMap to provide methods that handle parameter access from the router. We have then injected the ActivatedRoute and GoalService in our constructor. Inside the lifecycle hook, ngOnInit(), we have used the route.snapshot to get the initial value of the route parameter, then we have extracted the id using the get method provided by paramMap function. We have passed this id we have retrieved to our service's getGoal() method which returns the specific goal which we assign to our goal property.

Let's change the look of our GoalDetail.

src/app/goal-detail/goal-detail.component.html

```
<div class="row">
  <div class="col-md-2">

  </div>
  <div class="col-md-8">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">{{goal.description}}</h5>
        <p class="card-text">This goal will be complete in {{goal.completeDate|dateCount}} days.</p
>
      </div>
    </div>
  </div>
  <div class="col-md-2">

  </div>
</div>
```

We have put our goal details in a bootstrap card. We can now serve our application and see what happens when we click the view details button. That's exactly how we wanted to display our goal details.