Exercise (Instructions): Angular and RxJS Part 1

Objectives and Outcomes

In this exercise you will get your first experience with Observables in Angular. You will convert the services to use Observables in place of promises. You will then also enable the components to subscribe to the observables and make use of them to obtain the data. At the end of this exercise you will be able to:

- Make use of observables within your Angular application
- Obtain data through subscribing to the observables within your components.

Updating the Services

- As a first step update the services to operate using observables and then convert them to promises and deliver to the components. Open dish.service.ts and update its methods as follows:

```
. . .
import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/toPromise';
import 'rxjs/add/operator/delay';

. . .

  getDishes(): Promise<Dish[]> {
    return Observable.of(DISHES).delay(2000).toPromise();
  }

  getDish(id: number): Promise<Dish> {
    return Observable.of(DISHES.filter((dish) => (dish.id === id))[0]).delay
      (2000).toPromise();
  }

  getFeaturedDish(): Promise<Dish> {
    return Observable.of(DISHES.filter((dish) => dish.featured)[0]).delay(2000
      ).toPromise();
  }
}
```

- Similarly update leader.service.ts and promotion.service.ts to use observables.
- However, we would rather directly operate with observables. So update the dish.service.ts as follows:

```
  getDishes(): Observable<Dish[]> {
    return Observable.of(DISHES).delay(2000);
  }

  getDish(id: number): Observable<Dish> {
    return Observable.of(DISHES.filter((dish) => (dish.id === id))[0]).delay
      (2000);
  }

  getFeaturedDish(): Observable<Dish> {
    return Observable.of(DISHES.filter((dish) => dish.featured)[0]).delay(2000);
  }
```

- Remove the import of the rxjs promise operator from the file. Similarly update leader.service.ts and promotion.service.ts.

Updating the Components

- Update menu.component.ts as follows to subscribe to the observable returned from the service:

```
    this.dishService.getDishes().subscribe(dishes => this.dishes = dishes);

```

- Similarly update about.component.ts, dishdetail.component.ts and home.component.ts to subscribe to the observables from the services.
- Save all changes and commit the changes to your Git repository with the message "RxJs Part 1".

Conclusions

In this exercise you learn to use observables within your services and components.

Mark as completed