Exercise (Instructions): Angular and Promise Part 2

Objectives and Outcomes

In this exercise you will learn to deal with a service that takes some time to fetch and return the results by simulating a time delay in the service. You will also use the Angular Material progress spinner component to keep the users informed about the delay. At the end of this exercise you will be able to:

- Simulate a time delay within the service
- Update your components to be able to deal with the delay in obtaining the results
- Use the Angular Material progress spinner component to keep the user informed

Simulating Time Delay within the Service

- Open dish.service.ts and update its methods as follows:

```
getDishes(): Promise<Dish[]> {
  return new Promise(resolve=> {
    // Simulate server latency with 2 second delay
      setTimeout(() => resolve(DISHES), 2000);
  });
}

getDish(id: number): Promise<Dish> {
  return new Promise(resolve=> {
    // Simulate server latency with 2 second delay
      setTimeout(() => resolve(DISHES.filter((dish) => (dish.id === id))[0]), 2000);
  });
}

getFeaturedDish(): Promise<Dish> {
  return  new Promise(resolve=> {
    // Simulate server latency with 2 second delay
      setTimeout(() => resolve(DISHES.filter((dish) => dish.featured)[0]), 2000);
  });
}
```

- Update the promotion.service.ts and leader.service.ts files in the same manner.

Using Angular Material Progress Spinner

- Open menu.component.html file and update it by adding the following to the template:

```
<div fxFlex *ngIf="dishes">
. . .
</div>
<div [hidden]="dishes">
  <md-spinner></md-spinner><h4>Loading . . . Please Wait</h4>
</div>
```

- Update about.component.html, home.component.html and dishdetail.component.html similarly.

- Save the changes and do a Git commit with the message "Promise Part 2".

Conclusions

In this exercise you learnt to deal with delays in obtaining the results, and also dealing with promise reject.

Mark as completed