# Exercise (Instructions): Angular HTTP Client: Error Handling

## Objectives and Outcomes

In this exercise we will look at error handling with the HTTP client. It is possible that several errors could arise during the client-server communication. It is the responsibility of the client to properly handle the errors and recover and continue its operation. At the end of this exercise you will be able to:

- Catch errors that might arise while communicating with the server
- Design appropriate error handling mechanisms for the HTTP client

## Update ProcessHTTPMSgService to Handle Errors

- Update process-httpmsg.service.ts to include a function to handle errors as follows:

```
. . .

import 'rxjs/add/observable/throw';


. . .


  public handleError (error: Response | any) {
    // In a real world app, you might use a remote logging infrastructure
    let errMsg: string;
    if (error instanceof Response) {
      const body = error.json() || '';
      const err = body.error || JSON.stringify(body);
      errMsg = `${error.status} - ${error.statusText || ''} ${err}`;
    } else {
      errMsg = error.message ? error.message : error.toString();
    }
    console.error(errMsg);
    return Observable.throw(errMsg);
  }
  . . .
```

## Update Dish Service

- Update dish.service.ts to catch and handle errors as follows:

```
. . .
import 'rxjs/add/operator/catch';

. . .

  getDishes(): Observable<Dish[]> {
    return this.http.get(baseURL + 'dishes')
                    .map(res => { return this.processHTTPMsgService.extractData
                      (res); })
                    .catch(error => { return this.processHTTPMsgService
                      .handleError(error); });
  }

  getDish(id: number): Observable<Dish> {
    return  this.http.get(baseURL + 'dishes/'+ id)
                    .map(res => { return this.processHTTPMsgService.extractData
                      (res); })
                    .catch(error => { return this.processHTTPMsgService
                      .handleError(error); });
  }

  getFeaturedDish(): Observable<Dish> {
    return this.http.get(baseURL + 'dishes?featured=true')
                    .map(res => { return this.processHTTPMsgService.extractData
                      (res)[0]; })
                    .catch(error => { return this.processHTTPMsgService
                      .handleError(error); });
  }

  getDishIds(): Observable<number[]> {
    return this.getDishes()
      .map(dishes => { return dishes.map(dish => dish.id) })
      .catch(error => { return error; } );
  }
  . . .
```

## Update Menu Component

- Update menu.component.ts as follows to handle errors:

```
. . .
  errMess: string;
. . .

    this.dishService.getDishes()
    .subscribe(dishes => this.dishes = dishes,
      errmess => this.errMess = <any>errmess);
. . .
```

- Update menu.component.html file as follows:

```
. . .
  <div [hidden]="dishes || errMess">
    <md-spinner></md-spinner><h4>Loading . . . Please Wait</h4>
  </div>
  <div *ngIf="errMess">
    <h2>Error</h2>
    <h4>{{errMess}}</h4>
  </div>
  . . .
```

- Similarly update dishdetail.component.ts and dishdetail.component.html file.
- Also update home.component.ts and home.component.html files similarly, but use the variable named dishErrMess instead of errMess.

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

## Conclusions

In this exercise you have updated the Angular application to catch and handle errors in client-server communication.

Mark as completed