Angular 6 Training Course

Exercise H-vote

- This example reads and writes data to a **Node server** using GET and PUT calls from the Angular HTTPClient.

Review the Javascript-only version

- Open the **useful/vote** folder.
- Open the front-end **form/vote.html** in a browser.
- Choosing a village from the comboBox will add a vote
- Click on the postcard image will delete all votes.
- Open the back-end **form/server** folder in the terminal.
- This contains a Node server.
- The Node server uses Express, Body-Parser, Debug and Node-Persist packages, which need to be installed:

  ```
  npm install
  ```
- Run the Node server

  ```
  node server.js
  ```
- The server is now running on **port 4000**.
- It defines/exposes several **routes**.
- The **/getvotes** route uses a GET to return the votes cast.
- The **/vote** route lets POST a new vote.
- The **/empty** route uses a GET to deletes all votes.
- The **/testdata** route creates test data.

Converting the front-end to Angular.

- Leave the Node server running in port 4000.
- Rebuild the project in **starter/i-vote/angular**

  ```
  npm install
  ng serve --open
  ```

Add HTTP code to read votes cast

- We will use an **HTTP get** to read votes cast from the getVotes route on the server.
- Update the module file **app.module.ts**.

```
import { HttpClientModule } from '@angular/common/http';

imports:[ HttpClientModule ]
```

- Import the HTTPClient into the component.

```
import { HttpClient } from '@angular/common/http';
```

- Inject the HTTPClient in to the constructor:

```
 constructor( private fb:FormBuilder, private http: HttpClient )
{}
```

- Define the server routes as constants.

```
SERVER : string = "http://localhost:4000/village/";
GET_VOTES : string = this.SERVER + "getVotes";
```

- Write a method that uses the HTTPClient get() to read votes from the server.

```
getVotes() {
    this.http.get( this.GET_VOTES )
    .subscribe( data => console.log( data ))
}
```

- Call that method in the constructor

```
this.getVotes();
```

- It should log an array of selected villages to the console.

```
["Dumpling Green", .... ]
```

- Create a class property to display the votes cast.

```
votes : any;
```

- Change getVotes to set this property.

```
getVotes() {
    this.http.get( this.GET_VOTES )
    .subscribe( data => this.votes = data )
}
```

- Update the template to display that property.

```
<section class="votes">
```

```
<span class="village" *ngFor="let v of votes">{{ v }}</span>
```

Create a function to vote.

- We need to send a stringified JSON object to the server containing the name of the village.

```
"{"village": "Fiddlers Green"}"
```

- Update the selectVillage method.

```
selectVillage( form ) {
    let vote = JSON.stringify( { village:form.village });
    console.log( vote );
}
```

- Define a property for the vote route.

```
VOTE : string = this.SERVER + "vote";
```

- We can call the **HTTP post** method, passing 2 arguments: the route and the vote data.

```
this.http.post( this.VOTE , vote )
.subscribe( data => this.votes = data);
```

- This does not work correctly. Null values are added to the array of villages in the Node server.

Set JSON headers for the post.

- We need to set HTTP headers to post this JSON data correctly.
- Import the relevant class.

```
import { HttpHeaders } from '@angular/common/http';
```

- In the selectVillage method set the headers.

```
let jsonHeaders:HttpHeaders = new HttpHeaders()
.set('Accept', 'application/json').set('Content-Type',
'application/json');
```

- Pass these headers as an additional parameter.

```
this.http.post( this.VOTE , vote, { headers: jsonHeaders })
.subscribe( data => this.votes = data)
```

- The complete selectVillage method:

```
    selectVillage( form ) {
        let vote = JSON.stringify( { village:form.village });

        let jsonHeaders:HttpHeaders = new HttpHeaders()
        .set('Accept', 'application/json').set('Content-Type',
'application/json');

        this.http.post( this.VOTE , vote,{ headers: jsonHeaders
} )
        .subscribe( data => this.votes = data);
    }
```

Create a function to delete all votes.

- When the user clicks on the postcard, we will call an empty route on the server to delete all votes.
- Define the empty route.

  ```
  EMPTY : string = this.SERVER + "empty";
  ```

- Write a method to call this route with an http GET.

  ```
  empty() {
  this.http.get( this.EMPTY )
  .subscribe( data => this.votes = data)
  ```

  }
- Add a click event in the template

  ```
  <img (click)="empty()" src="assets/card.jpg" />
  ```

- Test this code by clicking on the postcard.