ANGULAR 2 FUNDAMENTALS

December 13, 2017

PART 14

FORMS IN ANGULAR

APPROACHES

- Most crucial part of an application
  - Validate users inputs
  - Display errors
  - Transform data
- Two options
  - Template driven
  - Code driven
- We will use both

CREATING OUR FIRST FORM

- Template driven way
- We need to import FormsModule
- Open app.module.ts import:

```
import { FormsModule } from '@angular/forms';
```

- Make sure its on imports below also:

```
imports: [
  routes,
  BrowserModule,
  HttpClientModule,
  FormsModule,
],
```

FORMSMODULE

- It contains some directives that we can add to a normal form and turn it into a powerful angular form.
- Let's create a new admin image create component!
- ng g c admin-image-create (inside admin folder ok!)
- Create image form will be at http://localhost:4200/admin/images/new
- Open admin.routes.ts and add a new route

```
export const adminRoutes: Routes = [
  { path: '', component: DashboardComponent},
  { path: 'images', component: AdminImageListComponent},
  { path: 'images/create', component: AdminImageCreateComponent}
];
```

MORE ON FORMS MODULE

- Don’t forget to import the component

```
import {AdminImageCreateComponent} from './admin-image-create/admin-image-create.compo
nent';
```

- Now we build our form using bootstrap and normal html:

- Admin-image-create.component.html

```
<div class="col-md-10 col-md-offset-1">
  <div>
    <a [routerLink]="['/admin/']" class="btn btn-default"> Back</a>
  </div>
</div>
```

ADMIN-IMAGE-CREATE.COMPONENT.HTML

```
<div class="col-md-10 col-md-offset-1">
  <div class="well well bs-component">
    <form class="form-horizontal">
      <fieldset>
        <legend>Add a new image</legend>
        <div class="form-group">
          <label for="thumbnail" class="col-lg-2 control-label">Thumbnail</label>
          <div class="col-lg-10">
            <input type="text" class="form-control" id="thumbnail" name="thumbnail" pl
aceholder="Thumbnail of the image">
          </div>
        </div>
        <div class="form-group">
            <label for="imagelink" class="col-lg-2 control-label">Image Link</label>
            <div class="col-lg-10">
              <input type="text" class="form-control" id="imageLink" name="imageLink"
placeholder="Link of the image">
            </div>
        </div>
```

ADMIN-IMAGE-CREATE.COMPONENT.HTML

```
        <div class="form-group">
          <label for="title" class="col-lg-2 control-label">Title</label>
          <div class="col-lg-10">
            <input type="text" class="form-control" id="title" name="title" placeholder
="Title">
          </div>
        </div>
        <div class="form-group">
          <label for="description" class="col-lg-2 control-label">Description</label>
          <div class="col-lg-10">
            <textarea class="form-control" rows="3" id="description" name="description"
 placeholder="Description of the image"></textarea>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-10 col-lg-offset-2">
            <button class="btn btn-default">Cancel</button>
            <button type="submit" class="btn btn-primary">Create</button>
          </div>
        </div>
      </fieldset>
    </form>
  </div>
</div>
```

CHECK HTTP://LOCALHOST:4200/ADMIN/IMAGES/CREATE

BUT WAIT THIS IS STILL PLAIN HTML LET'S MAKE IT ANGULAR

- Find:

```
<form class="form-horizontal">
```

- Update:

```
<form class="form-horizontal" (ngSubmit)="createImage(createImageForm.value)" #createI
mageForm="ngForm">
```

- What we are doing is we are assigning our form data to a variable called createImageForm using ngForm.
- Its like telling angular this is the form we want you to handle
- We are also using output binding here to respond to an event:

```
(ngSubmit)="createImage(createImageForm.value)"
```

MORE ON FORMS

- Using **ngSubmit** when we hit the submit button, calls the **createImage()** method. The **createImageForm.value** contains our data.
- Next we have to add ngModel directives to all input fields
- Find:

```
<input type="text" class="form-control" id="thumbnail" name="thumbnail" placeholder="T
humbnail of the image">

<input type="text" class="form-control" id="imageLink" name="imageLink" placeholder="L
ink of the image">

<input type="text" class="form-control" id="title" name="title" placeholder="Title">

<textarea class="form-control" rows="3" id="description" name="description" placeholder
="Description of the image"></textarea>
```

NGMODEL

- Modify to:

```
<input type="text" class="form-control" id="thumbnail" name="thumbnail" placeholder="T
humbnail of the image" ngModel>

<input type="text" class="form-control" id="imageLink" name="imageLink" placeholder="L
ink of the image" ngModel>

<input type="text" class="form-control" id="title" name="title" placeholder="Title" ng
Model>

<textarea class="form-control" rows="3" id="description" name="description" placeholder
="Description of the image" ngModel></textarea>
```

ADMIN-IMAGE-CREATE.COMPONENT.HTML SHOULD NOW LOOK

```
<div class="col-md-10 col-md-offset-1">
  <div>
    <a [routerLink]="['/admin/']" class="btn btn-default"> Back</a>
  </div>
</div>

<div class="col-md-10 col-md-offset-1">
  <div class="well well bs-component">
    <form class="form-horizontal" (ngSubmit)="createImage(createImageForm.value)" #cre
```

```
ateImageForm="ngForm">
      <fieldset>
        <legend>Add a new image</legend>
        <div class="form-group">
          <label for="thumbnail" class="col-lg-2 control-label">Thumbnail</label>
          <div class="col-lg-10">
            <input type="text" class="form-control" id="thumbnail" name="thumbnail" pl
aceholder="Thumbnail of the image" ngModel>
          </div>
        </div>
```

ADMIN-IMAGE-CREATE.COMPONENT.HTML SHOULD NOW LOOK

```
        <div class="form-group">
          <label for="imagelink" class="col-lg-2 control-label">Image Link</label>
          <div class="col-lg-10">
            <input type="text" class="form-control" id="imageLink" name="imageLink" pl
aceholder="Link of the image" ngModel>
          </div>
        </div>
        <div class="form-group">
          <label for="title" class="col-lg-2 control-label">Title</label>
          <div class="col-lg-10">
            <input type="text" class="form-control" id="title" name="title" placeholder
="Title" ngModel>
          </div>
        </div>
        <div class="form-group">
          <label for="description" class="col-lg-2 control-label">Description</label>
          <div class="col-lg-10">
            <textarea class="form-control" rows="3" id="description" name="description"
 placeholder="Description of the image" ngModel></textarea>
          </div>
        </div>
```

ADMIN-IMAGE-CREATE.COMPONENT.HTML SHOULD NOW LOOK

```
        <div class="form-group">
          <div class="col-lg-10 col-lg-offset-2">
            <button class="btn btn-default">Cancel</button>
            <button type="submit" class="btn btn-primary">Create</button>
          </div>
        </div>
      </fieldset>
    </form>
  </div>
```

ADMIN-IMAGE-CREATE.COMPONENT.TS

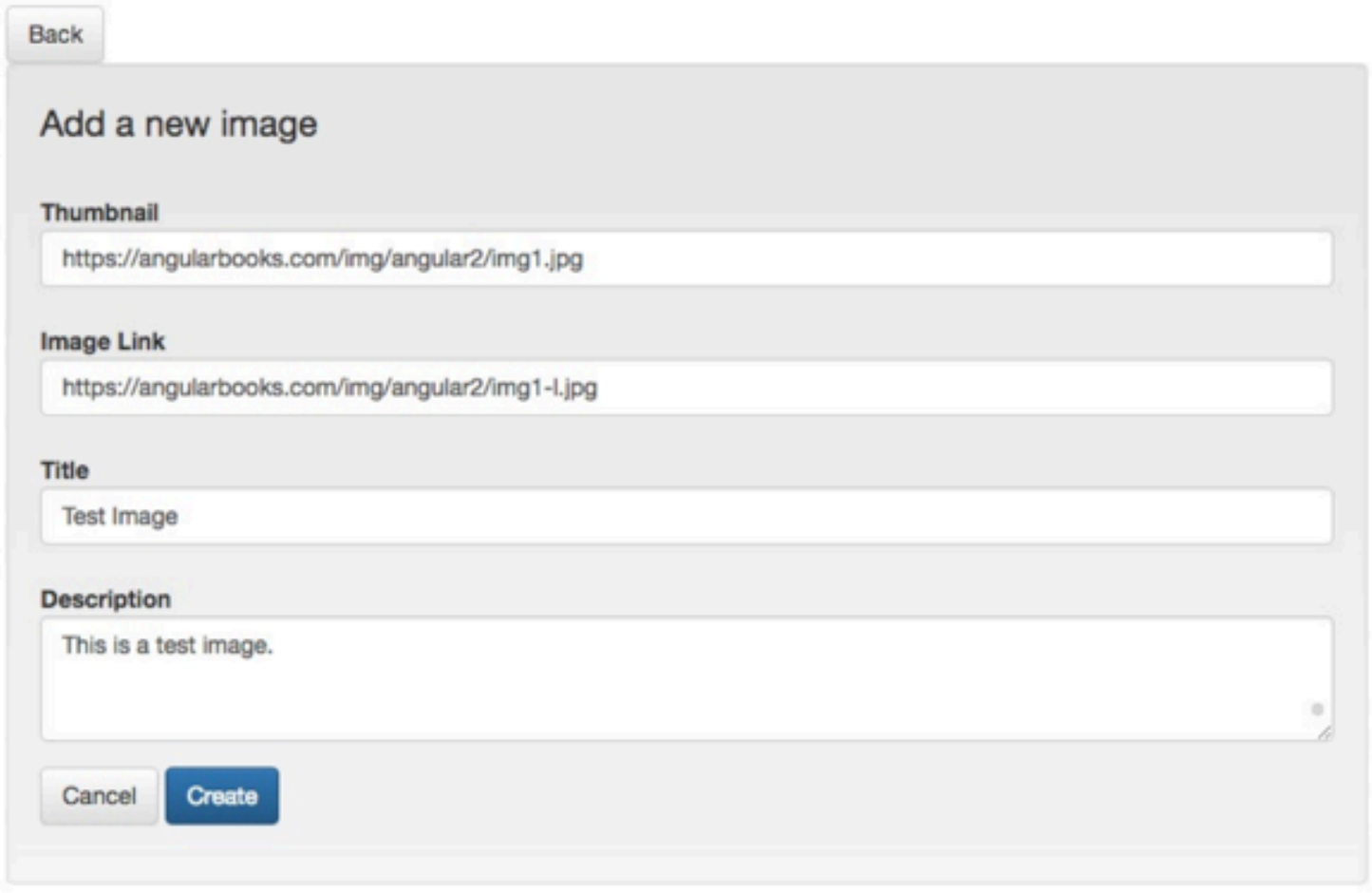
- Add createImage method:

```
import { Component, OnInit } from '@angular/core';
```

```
@Component({
  selector: 'ng-admin-image-create',
  templateUrl: './admin-image-create.component.html',
  styleUrls: ['./admin-image-create.component.css']
})
export class AdminImageCreateComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

  createImage(image) {
    console.log(image);
  }
}
```

TRY TO SUBMIT THE FORM AND SEE DATA IN CONSOLE

- Right click inspect on browser and go to console.




```
▼ Object                                                    main.bundle.js:774
    description: "This is a test image."
    imagelink: "https://angularbooks.com/img/angular2/img1-l.jpg"
    thumbnail: "https://angularbooks.com/img/angular2/img1.jpg"
    title: "Test Image"
  ▶ __proto__: Object
```

CONSOLE IS USELESS! LET'S SEND THIS POST REQUEST

- Let's use our Image Service
- image.service.ts
- Add a new **addImage** method:

IMAGE.SERVICE.TS

```
import { Injectable } from '@angular/core';
import {HttpClient} from '@angular/common/http';
import 'rxjs/Rx';
import {Observable} from 'rxjs/Observable';
import {Image} from '../models/image';

@Injectable()
export class ImageService {

  constructor(private http: HttpClient) {
  }

  getImages(): Observable<any> {
    return this.http.get('http://angularbook.app/api/v1/images')
      .map((response) => response);
  }

  addImage(image: Object): Observable<Image[]> {
    return this.http.post('http://angularbook.app/api/v1/images', image)
      .map((response) => response)
      .catch((error: any) => Observable.throw(error.json().error || {message: 'Server
Error'} ));
  }
}
```

MORE ON IMAGE SERVICE

- Instead of using:

```
addImage(image: Object): Observable<any>
```

- We use:

```
addImage(image: Object): Observable<Image[]>
```

- Because we know that the observable is a type of image, if we code like this we have to import the Image model.

```
import {Image} from '../models/image';
```

- You can still use any….tamad ka!

STILL ON SERVICE

- This method will return an observable
- We use http.post to send post data request to server
- The second argument is the image object
- Additionally we added one more line

```
.catch((error:any) => Observable.throw(error.json().error || {message:"Server Error"} )
);
```

- This is how we show errors if something goes wrong
- You can also add this line in getImages

ONE MORE THING ABOUT SERVICES

- http.post has a third argument which is used to add extra information such as request headers for example:

```
let headers = new Headers({ 'Content-Type': 'application/json' });
let options = new RequestOptions({ headers: headers });

return this.http.post(this.commentsUrl, body, options)
```

- If you use Headers and Request Options please don't forget to import

```
import {Http, Response, Headers, RequestOptions} from "@angular/http";
```

Newer angular version autodetects type of object no need to add request header

BACK TO ADMIN-IMAGE-CREATE.COMPONENT.TS

- Update the createImage method:

```
createImage(image) {
 this.imageService.addImage(image)
    .subscribe(
      image  => console.log(image),
      error =>  console.log(<any>error)
    );
}
```

- don't forget to import the service:

```
import {ImageService} from '../../services/image.service';
```

- And inject the service

```
constructor(private imageService: ImageService) { }
```

ADMIN/ADMIN-IMAGE-CREATE.COMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import {ImageService} from '../../services/image.service';

@Component({
  selector: 'ng-admin-image-create',
  templateUrl: './admin-image-create.component.html',
  styleUrls: ['./admin-image-create.component.css']
})
export class AdminImageCreateComponent implements OnInit {

  constructor(private imageService: ImageService) { }

  ngOnInit() {
  }

  createImage(image) {
    this.imageService.addImage(image)
      .subscribe(
        image  => console.log(image),
        error =>  console.log(<any>error)
      );
  }
}
```

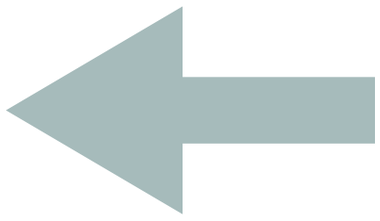
FIXING CORS

- When sending post data you might see this error

```
Response to preflight request doesn't pass access control check: No 'Access-Control-Al
low-Origin' header is present on the requested resource
```

- This is the admins job on the server side easiest is to open public/index.php on laravel and above: `require __DIR__.'/../bootstrap/autoload.php';`

```php
$allowedOrigins = array(
    '(http(s)://)?(www\.)?angularbook\.app',
    'http://localhost:4200'
);

if (isset($_SERVER['HTTP_ORIGIN']) && $_SERVER['HTTP_ORIGIN'] != '') {
    foreach ($allowedOrigins as $allowedOrigin) {
        if (preg_match('#' . $allowedOrigin . '#', $_SERVER['HTTP_ORIGIN'])) {
            header('Access-Control-Allow-Origin: ' . $_SERVER['HTTP_ORIGIN']);
            header('Access-Control-Allow-Methods: GET, PUT, POST, DELETE, OPTIONS');
            header('Access-Control-Max-Age: 1000');
            header('Access-Control-Allow-Headers: Content-Type, Authorization, X-Reque
sted-With');
            break;
        }
    }
}
```



LAST CONFIG ON CORS

- Open Cors middleware and update handle function

```
public function handle($request, Closure $next)
{
    return $next($request)
        ->header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS')
        ->header('Access-Control-Allow-Headers', 'content-type, withcredentials, Acces
s-Control-Allow-Headers, Origin, Accept, X-Requested-With, Content-Type, Access-Contro
l-Request-Method, Access-Control-Request-Headers');
}
```

- This should have been handled by the barred cars module which we have done before so don’t bother

CREATE A NEW IMAGE AND REDIRECT USERS TO ANOTHER ROUTE

- Our backend doesn't handle the post yet so open ImagesController in larval and update the store method

HTTP/CONTROLLERS/IMAGESCONTROLLER.PHP

```php
public function store(Request $request)
{
    if ((!$request->title) || (!$request->thumbnail) || (!$request->imageLink)) {

        $response = Response::json([
            'message' => 'Please enter all required fields'
        ], 422);
        return $response;
    }

    $image = new Image(array(
        'thumbnail' => trim($request->thumbnail),
        'imageLink' => trim($request->imageLink),
        'title' => trim($request->title),
        'description' => trim($request->description),
        'user_id' => 1
    ));
    $image->save();

    $message = 'Your image has been added successfully';

    $response = Response::json([
        'message' => $message,
        'data' => $image,
    ], 201);

    return $response;
}
```

JUST SOME EXPLANATION ON CODE

- First we check the request if one of the fields is empty if yes we send back a response telling “Please enter all fields”
- If okay we create the new image:

```
$image = new Image(array(
    'thumbnail' => trim($request->thumbnail),
    'imageLink' => trim($request->imageLink),
    'title' => trim($request->title),
    'description' => trim($request->description),
    'user_id' => 1
));
$image->save();
```

- If saved we send a success response back along with image data.

```
$message = 'Your image has been added successfully';
```

JUST SOME EXPLANATION ON CODE

```php
$response = Response::json([
    'message' => $message,
    'data' => $image,
], 201);

return $response;
```

ITS TIME TO ADD THE IMAGE, TRY IT

- Check the console tab and you shall see:


```
▼Object
  ▼data: Object
      created_at: "2017-03-26 04:50:15"
      description: "This is a test image."
      id: 9
      imageLink: "https://angularbooks.com/img/angular2/img1-l.jpg"
      thumbnail: "https://angularbooks.com/img/angular2/img1.jpg"
      title: "Test image"
      updated_at: "2017-03-26 04:50:15"
      user_id: 1
    ▶__proto__: Object
   message: "Your image has been added successfully"
  ▶__proto__: Object
```


- Yahoo! Let's now go home…..
- Wait normally in a real app we redirect the users to another page so we use the router.navigate

ADMIN-IMAGE-CREATE.COMPONENT.TS

- Add to createImage method:

```
createImage(image) {
  this.imageService.addImage(image)
    .subscribe(
      image  => {
        console.log(image);
        this.router.navigate(['/admin/images']);
      },
      error =>  console.log(<any>error));
}
```

- Don’t forget to import router:

```
import { Router} from '@angular/router';
```

- Inject router in the constructor:

```
constructor(private imageService: ImageService, private router: Router) { }
```

THE UPDATED CODE SUMMARISED

```
import { Component, OnInit } from '@angular/core';
import {ImageService} from '../../services/image.service';
import { Router} from '@angular/router';

@Component({
  selector: 'ng-admin-image-create',
  templateUrl: './admin-image-create.component.html',
  styleUrls: ['./admin-image-create.component.css']
})
export class AdminImageCreateComponent implements OnInit {

  constructor(private imageService: ImageService, private router: Router) { }

  ngOnInit() {
  }

  createImage(image) {
    this.imageService.addImage(image)
      .subscribe(
        image  => {
          console.log(image);
          this.router.navigate(['/admin/images']);
        },
        error =>  console.log(<any>error));
  }
}
```

You may remove console .log method if you like

TRY IT AGAIN

- You should be redirected to http://localhost:4200/admin/images

END OF PART 14