Phoenix one
The Knowledge Institute

# ANGULAR 2 FUNDAMENTALS

*December 13, 2017*

# PART 15

## FORM VALIDATIONS USING TEMPLATE DRIVEN APPROACH

## VALIDATION

- First we have to add novalidate attribute to form.
- It specifies that the form data should not be validated by the browser when submitted.

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

- Here is how we add some validation rules in an input field:

```
<input type="text" class="form-control" id="thumbnail" name="thumbnail" placeholder="T
humbnail of the image" ngModel #thumbnail="ngModel" required minlength="2">
```

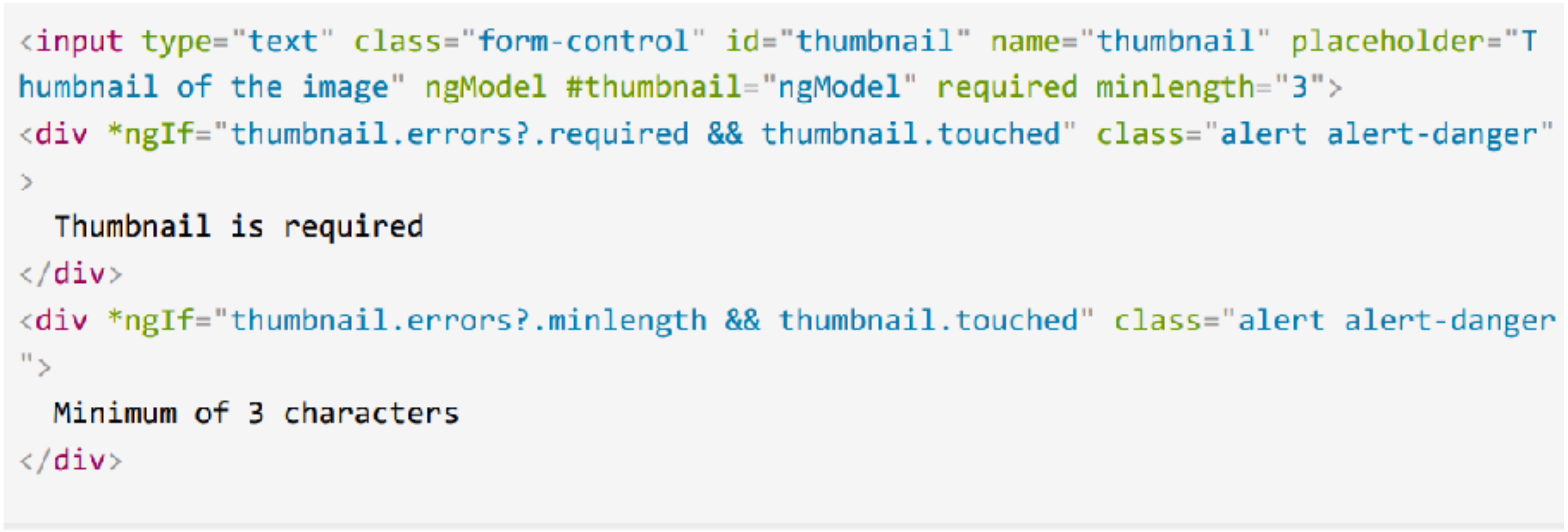
- By adding

```
#thumbnail="ngModel" required minlength="3"
```

- We assign input data to a local variable thumbnail

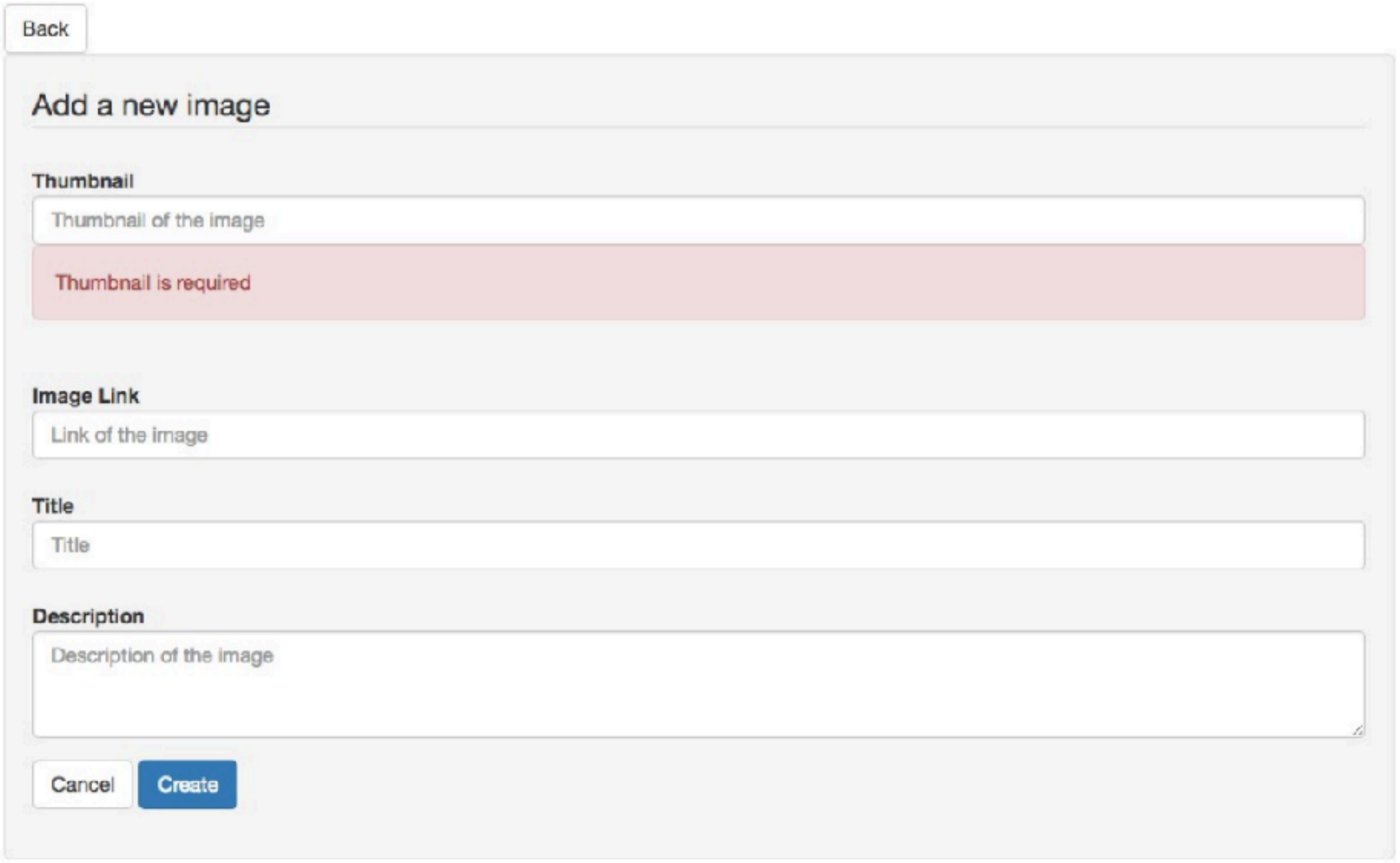
## VALIDATION

- Angular will know that this field is an angular model
- The field should be required
- And minimum length is 3
- Now we can display errors by adding



```
<input type="text" class="form-control" id="thumbnail" name="thumbnail" placeholder="T
humbnail of the image" ngModel #thumbnail="ngModel" required minlength="3">
<div *ngIf="thumbnail.errors?.required && thumbnail.touched" class="alert alert-danger"
>
  Thumbnail is required
</div>
<div *ngIf="thumbnail.errors?.minlength && thumbnail.touched" class="alert alert-danger
">
  Minimum of 3 characters
</div>
```

## TRY IT OUT!


Back

Add a new image

Thumbnail

Thumbnail of the image

Thumbnail is required

Image Link

Link of the image

Title

Title

Description

Description of the image

Cancel Create

## VALIDATIONS PROVIDED BY THE FRAMEWORK

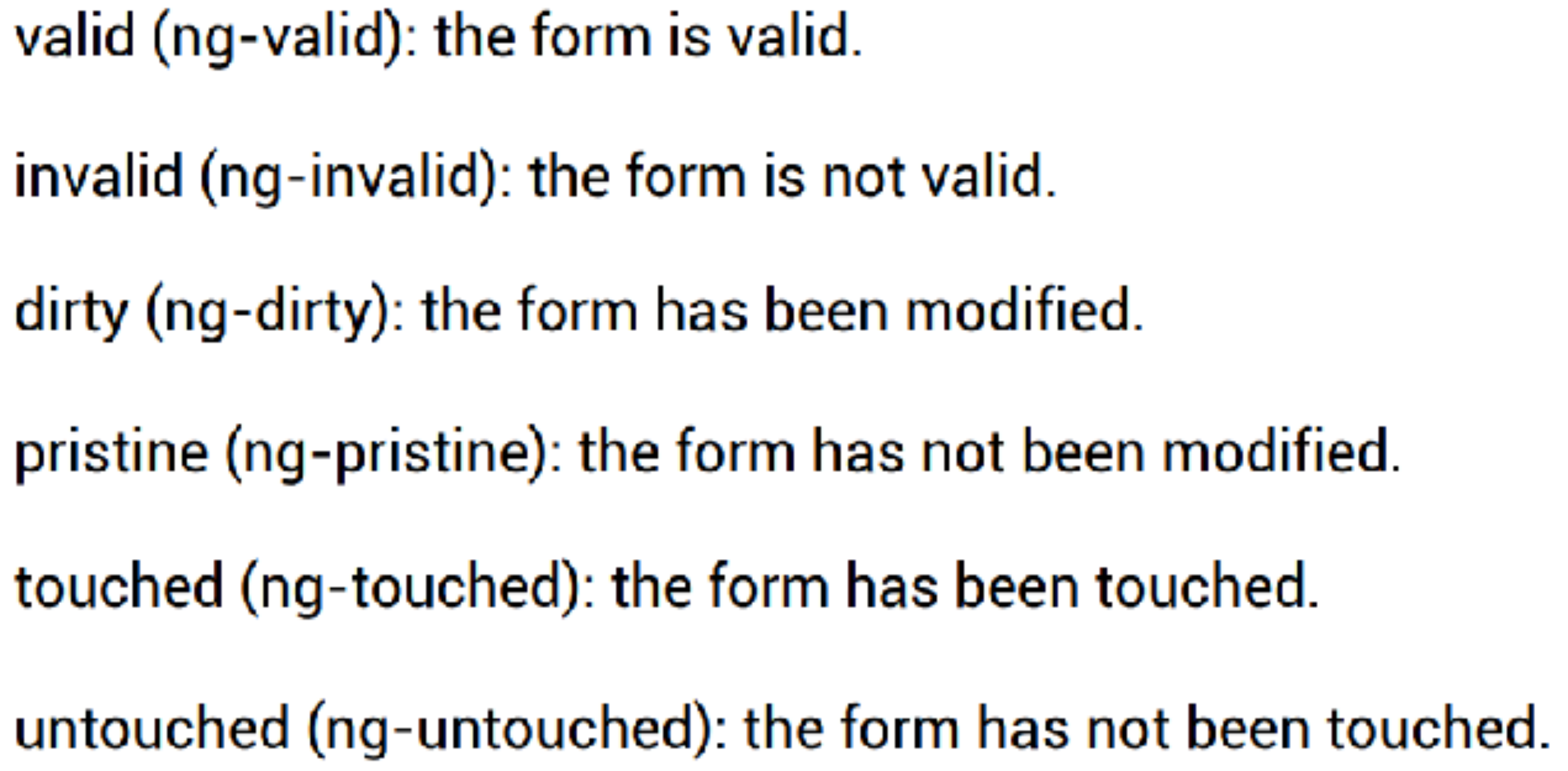
- Required
- Min length
- Max length
- Pattern (regular expressions)

## MORE ON VALIDATIONS

- So how do we know what was touched?
- It call angular states

## STATES OF ANGULAR FORM

- Angular form is smart
- It knows when a user touched the control
- A form will be valid and has the ng-valid class when validators succeed



valid (ng-valid): the form is valid.

invalid (ng-invalid): the form is not valid.

dirty (ng-dirty): the form has been modified.

pristine (ng-pristine): the form has not been modified.

touched (ng-touched): the form has been touched.

untouched (ng-untouched): the form has not been touched.

## SAMPLES

- We can disable submit button if using the invalid state
- Find:

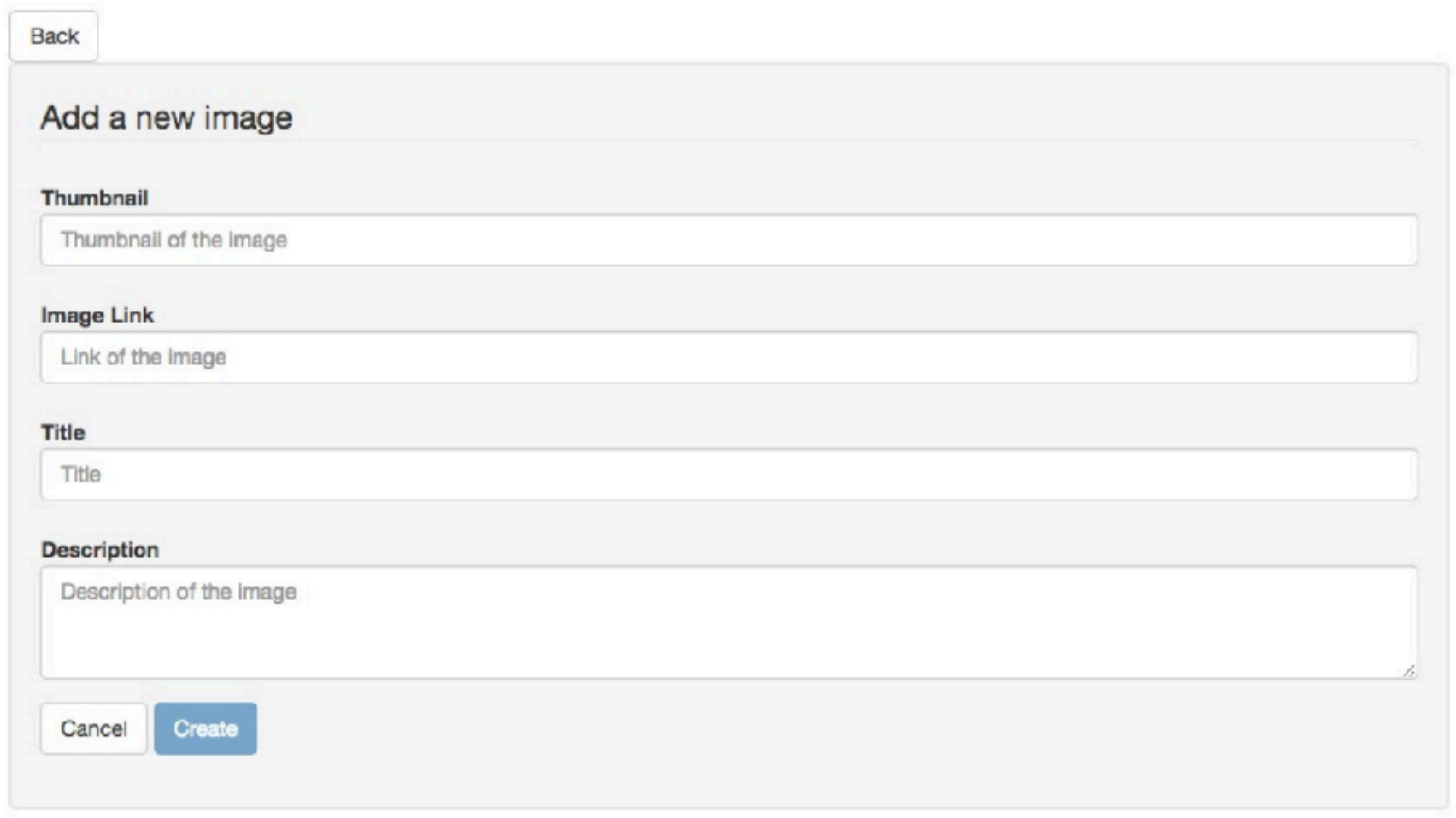
```
<button type="submit" class="btn btn-primary">Create</button>
```

- Modify:

```
<button type="submit" class="btn btn-primary" [disabled]="createImageForm.invalid">Cre
ate</button>
```

- Try it our again

# SITE TESTING



## LET'S WORK ON CANCEL BUTTON ALSO

- Find:

```
<button class="btn btn-default">Cancel</button>
```

- Modify:

```
<a [routerLink]="['/admin/']" class="btn btn-default"> Cancel</a>
```

- More validations on admin-create-image.component.html

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

# MORE ON ADMIN-CREATE-IMAGE.COMPONENT.HTML

```html
<div class="col-md-10 col-md-offset-1">
  <div class="well well bs-component">
    <form novalidate class="form-horizontal" (ngSubmit)="createImage(createImageForm.v
alue)" #createImageForm="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 #thumbnail="ngModel" required minlength="3">

            <div *ngIf="thumbnail.errors?.required && thumbnail.touched" class="alert
alert-danger">
              Thumbnail is required
            </div>
            <div *ngIf="thumbnail.errors?.minlength && thumbnail.touched" class="alert
 alert-danger">
              Minimum of 3 characters
            </div>
          </div>
        </div>
        <div class="form-group">
```

# MORE ON ADMIN-CREATE-IMAGE.COMPONENT.HTML

```
      <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" ngModel #imageLink="ngModel" required>
       <div *ngIf="imageLink.errors?.required && imageLink.touched" class="aler
t alert-danger">
        ImageLink is required
       </div>
      </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 #title="ngModel" required>
      <div *ngIf="title.errors?.required && title.touched" class="alert alert-da
nger">
       Title is required
      </div>
     </div>
    </div>
```

## MORE ON ADMIN-CREATE-IMAGE.COMPONENT.HTML

```
    <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 #description="ngModel" required></text
area>
      <div *ngIf="description.errors?.required && description.touched" class="al
ert alert-danger">
       Description is required
      </div>
     </div>
    </div>

    <div class="form-group">
     <div class="col-lg-10 col-lg-offset-2">
      <a [routerLink]="['/admin/']" class="btn btn-default"> Cancel</a>
      <button type="submit" class="btn btn-primary" [disabled]="createImageForm.
invalid">Create</button>
     </div>
    </div>
   </fieldset>
  </form>
 </div>
</div>
```

## TEST IT!

- Click the cancel button and it should redirect

## EDITING AND IMAGE

## NOW WE WILL CREATE AND EDIT COMPONENT

- ng g c admin-image-edit
- What are route parameters?
  - It allows us to pass values from url to our component
  - Like /admin/images/edit/1
- Let’s proceed open admin.routes.ts and add:

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

- Import AdminImageComponent

```
import {AdminImageEditComponent} from './admin-image-edit/admin-image-edit.component';
```

## EXPLAINING THE ROUTE

- As you can see images/edit/:id
- It denotes its a route parameter
- And angular should get this id from the url
- So how do we get the route parameter id?
- Open admin-image-edit.component.ts and find:

```
export class AdminImageEditComponent implements OnInit {
```

- Add below:

```
id: any;
params: any;
```

## STILL ON ROUTE PARAMETER

- We create 2 variables id and params
- After that we inject ActivatedRoute from @angular/router into our component

```
constructor(private activatedRoute: ActivatedRoute) { }
```

- Don’t forget to import:

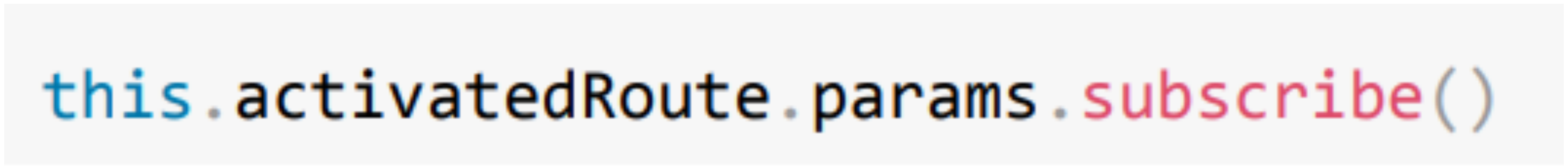
```
import {ActivatedRoute} from '@angular/router';
```

- Here is how we get the id of the page:

```
ngOnInit() {
  this.params = this.activatedRoute.params.subscribe(params => this.id = params['id']);

}
```

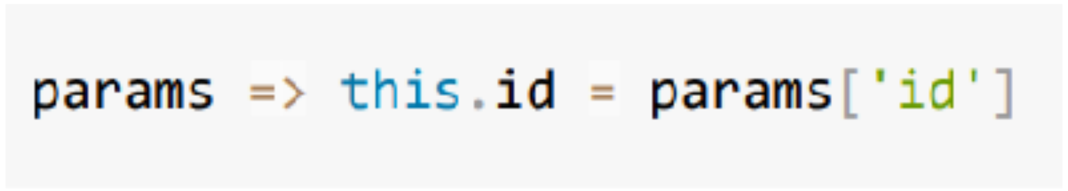
## ACTIVATEDROUTE

- We are basically subscribing to an observable using:



```
this.activatedRoute.params.subscribe()
```

- When the route changes we will know immediately. We can grab the value (which is the id) using params['id'] and assign it to our id variable



```
params => this.id = params['id']
```

- To prevent memory leaks we should unsubscribe the params when our component is destroyed

## UNSUBSCRIBING

- Find:

```
ngOnInit() {
  this.params = this.activatedRoute.params.subscribe(params => this.id = params['id']);

}
```

- Add below:

```
ngOnDestroy() {
  this.params.unsubscribe();
}
```

- Find:

```
export class AdminImageEditComponent implements OnInit {
```

## UNSUBSCRIBING

- ➤ Update to:

```
export class AdminImageEditComponent implements OnInit, OnDestroy {
```

- ➤ Import OnDestroy into our component:

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

# ADMIN-IMAGE-EDIT.COMPONENT.TS

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

@Component({
  selector: 'ng-admin-image-edit',
  templateUrl: './admin-image-edit.component.html',
  styleUrls: ['./admin-image-edit.component.css']
})
export class AdminImageEditComponent implements OnInit, OnDestroy {
  id: any;
  params: any;

  constructor(private activatedRoute: ActivatedRoute) { }

  ngOnInit() {
    this.params = this.activatedRoute.params.subscribe(params => this.id = params['id']
);
  }

  ngOnDestroy() {
    this.params.unsubscribe();
  }
}
```

## ADMIN-IMAGE-EDIT.COMPONENT.HTML

- Find:

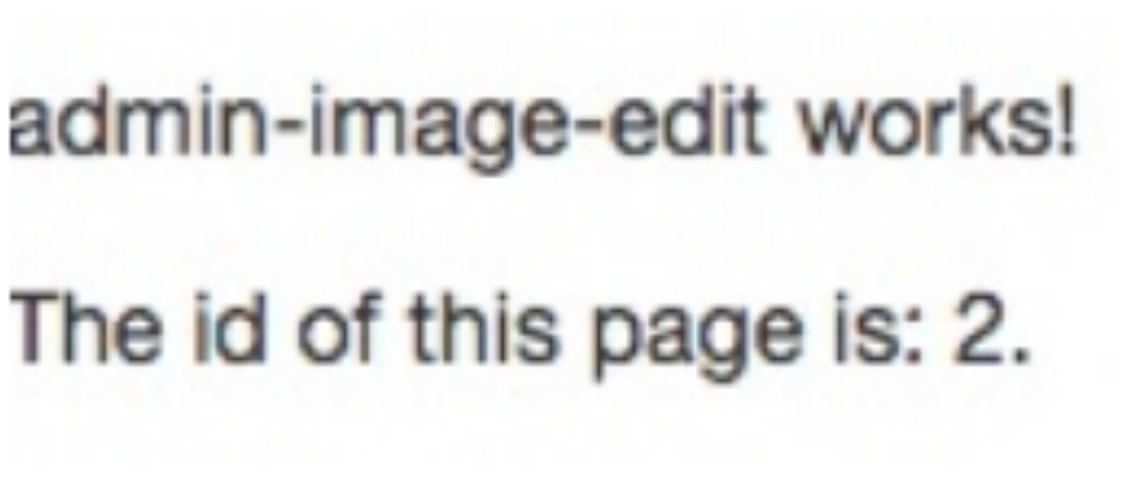
```
<p>
  admin-image-edit works!
</p>
```

- Add below:

```
<p>The id of this page is: {{id}}.</p>
```

- We use {{id}} to display the id of the page
- Visit http://localhost:4200/admin/images/edit/2

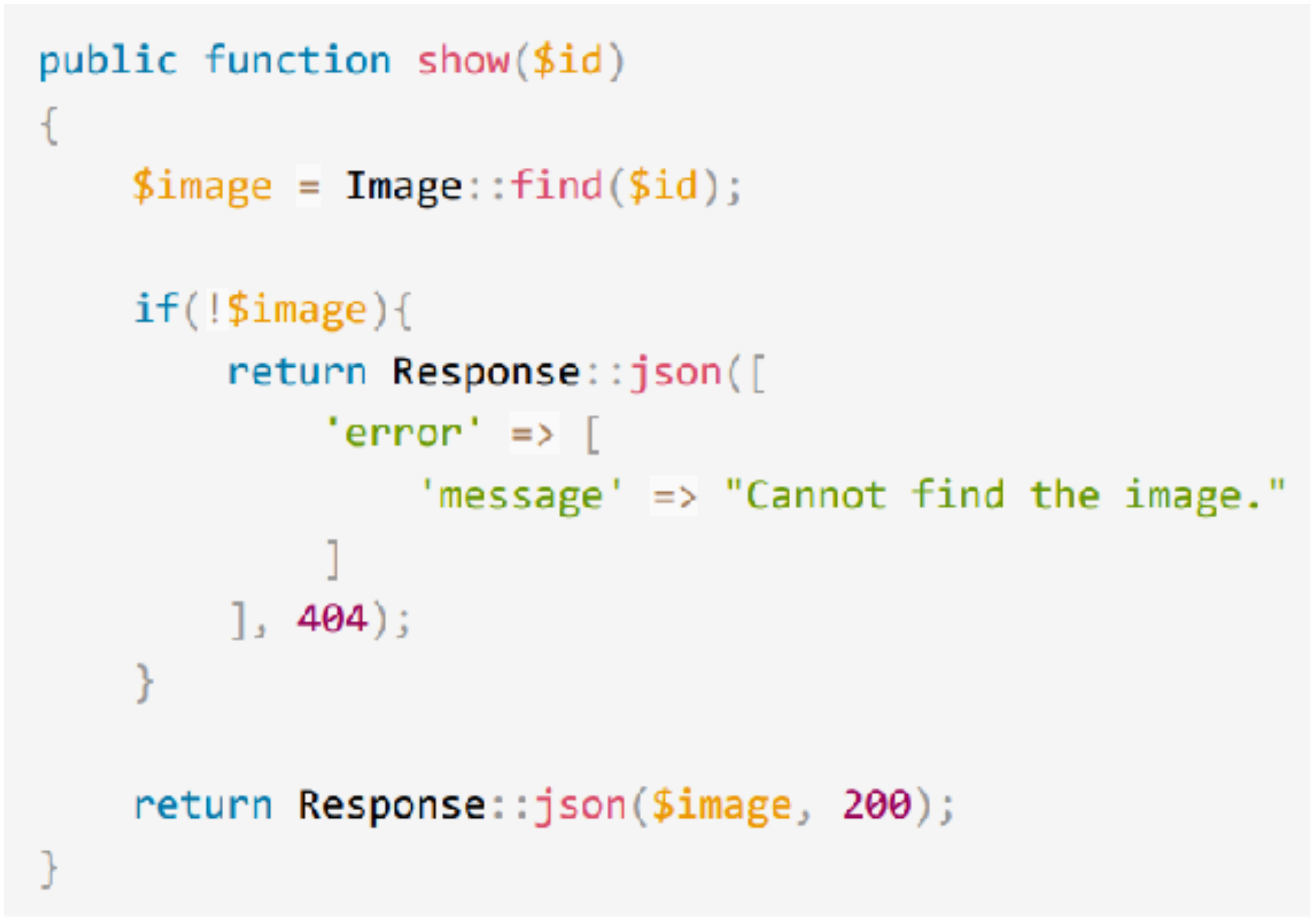
## CHECK IT OUT


admin-image-edit works!

The id of this page is: 2.

## GET A SINGLE IMAGE USING LARVAL

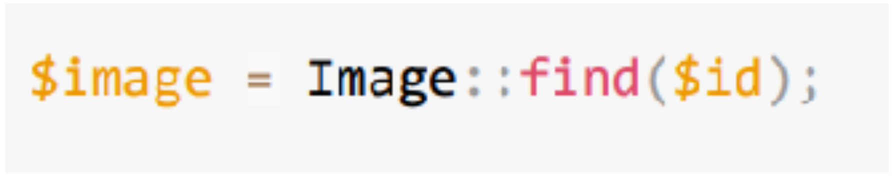
➤ Open ImagesController.php update the show method



```php
public function show($id)
{
    $image = Image::find($id);

    if(!$image){
        return Response::json([
            'error' => [
                'message' => "Cannot find the image."
            ]
        ], 404);
    }

    return Response::json($image, 200);
}
```

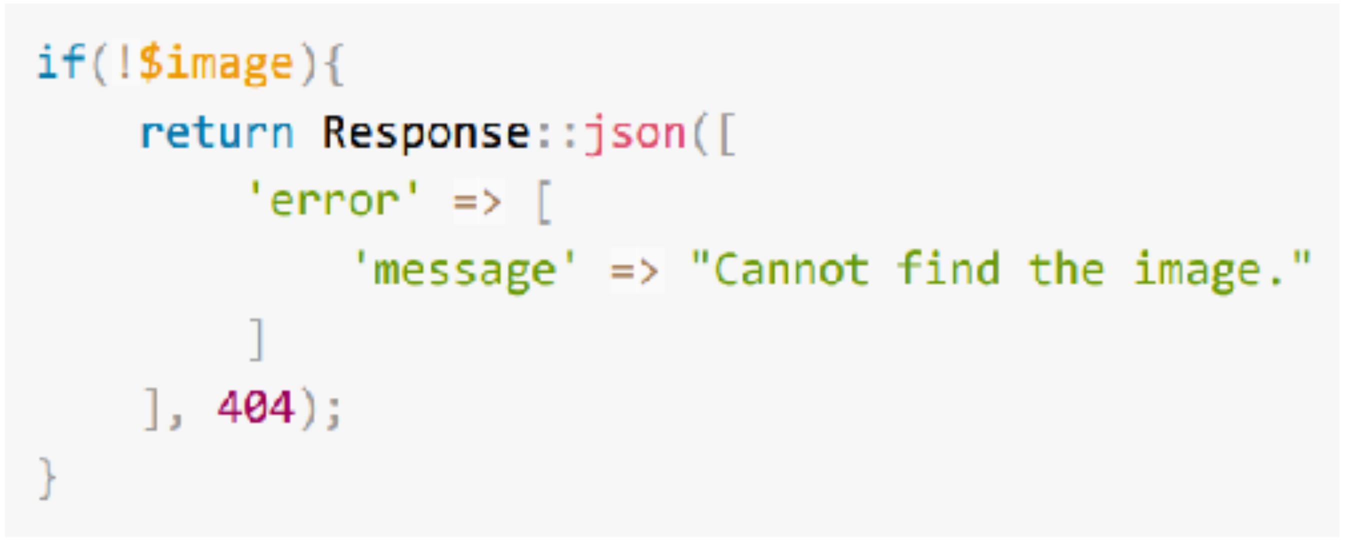
## EXPLANATION

- We find the image based on its id



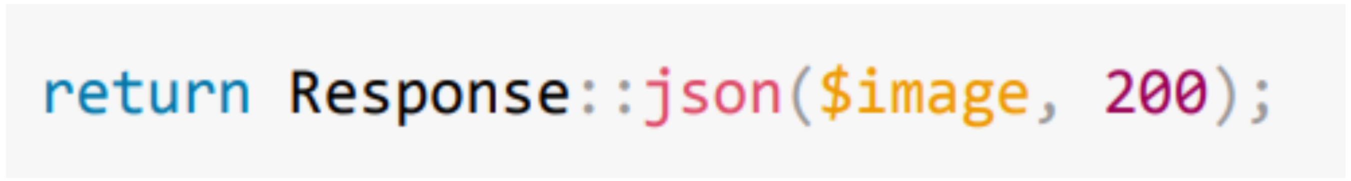
```
$image = Image::find($id);
```

- If cant be found an error is returned



```
if(!$image){
    return Response::json([
        'error' => [
            'message' => "Cannot find the image."
        ]
    ], 404);
}
```

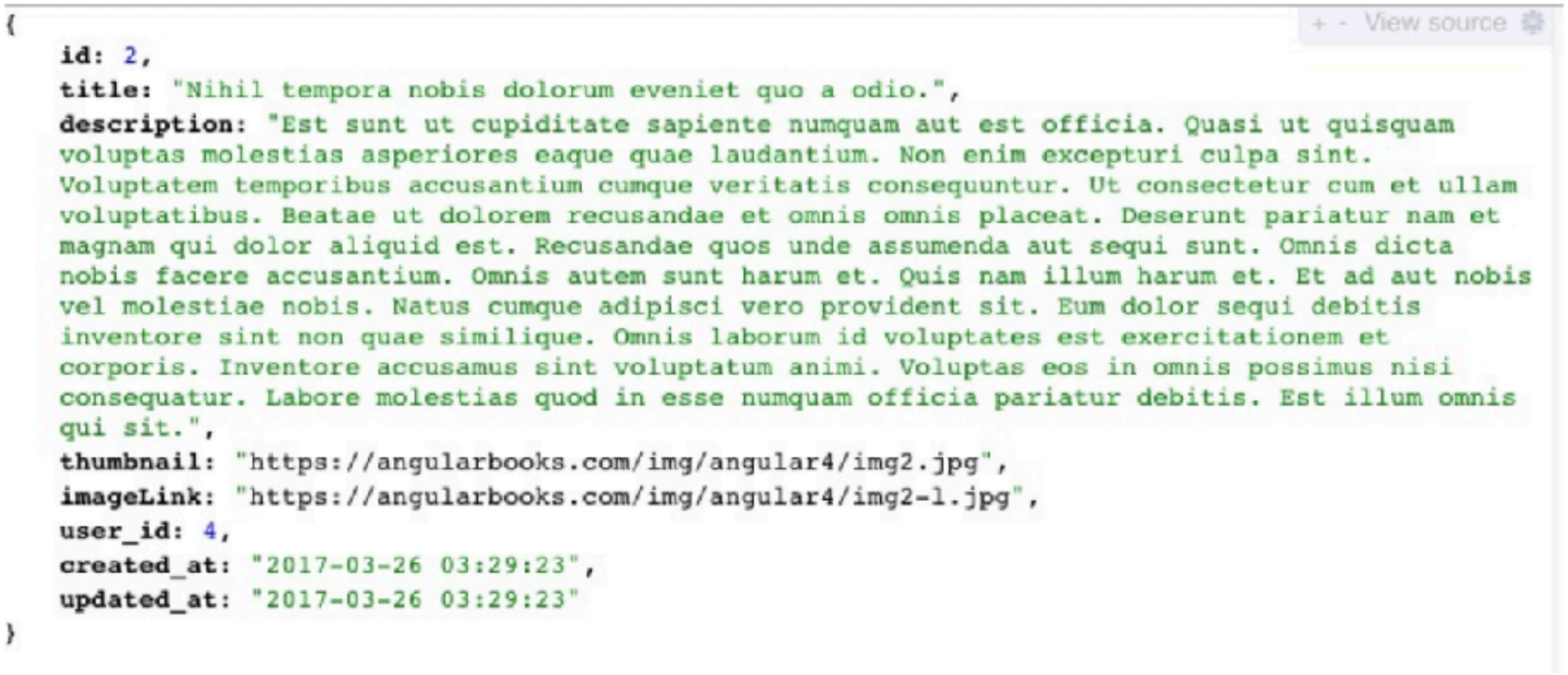
- If found 200 response is sent



```
return Response::json($image, 200);
```

## CHECK IN LARAVEL

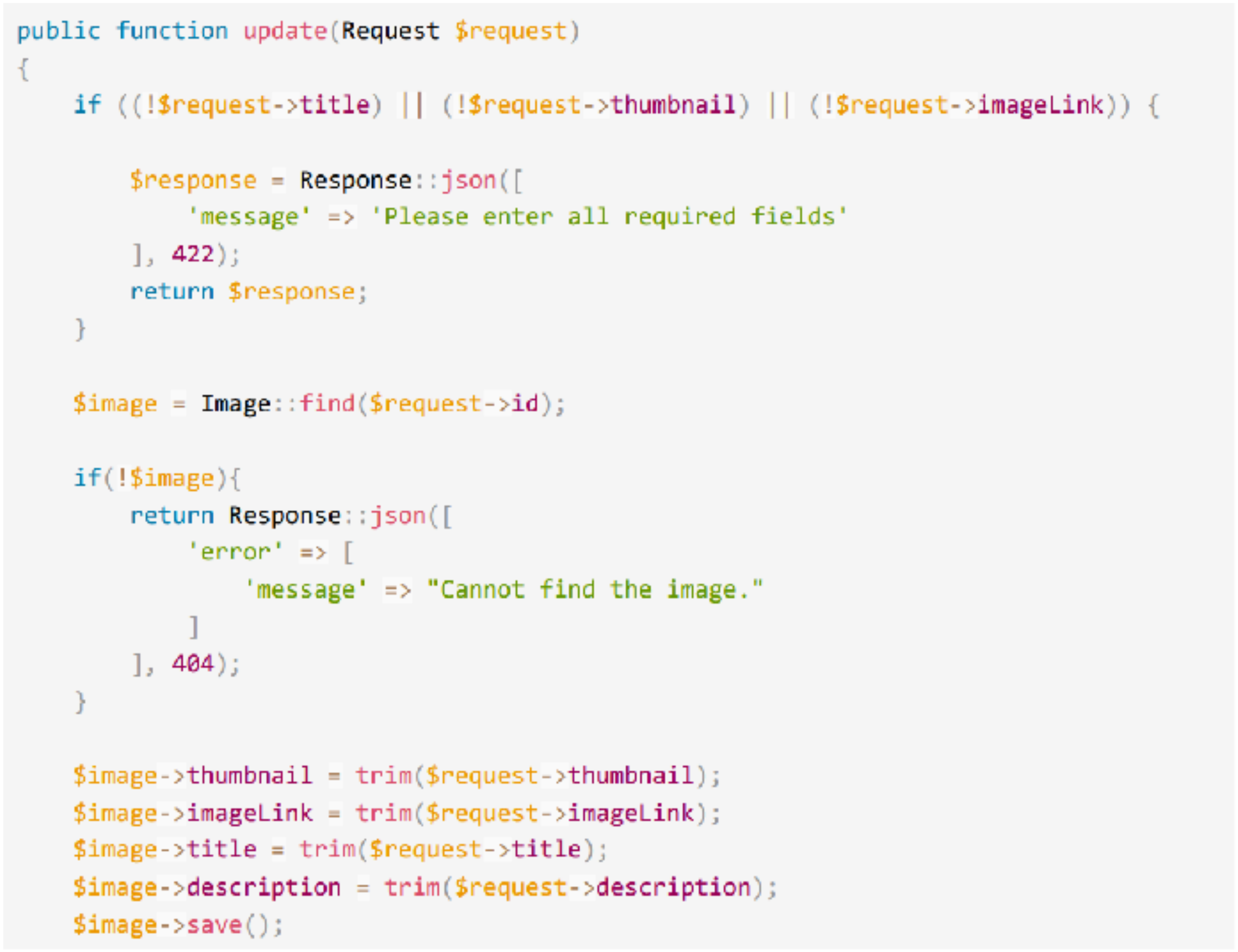
- Open http://localhost:8000/api/v1/images/2



```
{
    id: 2,
    title: "Nihil tempora nobis dolorum eveniet quo a odio.",
    description: "Est sunt ut cupiditate sapiente numquam aut est officia. Quasi ut quisquam
    voluptas molestias asperiores eaque quae laudantium. Non enim excepturi culpa sint.
    Voluptatem temporibus accusantium cumque veritatis consequuntur. Ut consectetur cum et ullam
    voluptatibus. Beatae ut dolorem recusandae et omnis omnis placeat. Deserunt pariatur nam et
    magnam qui dolor aliquid est. Recusandae quos unde assumenda aut sequi sunt. Omnis dicta
    nobis facere accusantium. Omnis autem sunt harum et. Quis nam illum harum et. Et ad aut nobis
    vel molestiae nobis. Natus cumque adipisci vero provident sit. Eum dolor sequi debitis
    inventore sint non quae similique. Omnis laborum id voluptates est exercitationem et
    corporis. Inventore accusamus sint voluptatum animi. Voluptas eos in omnis possimus nisi
    consequatur. Labore molestias quod in esse numquam officia pariatur debitis. Est illum omnis
    qui sit.",
    thumbnail: "https://angularbooks.com/img/angular4/img2.jpg",
    imageLink: "https://angularbooks.com/img/angular4/img2-1.jpg",
    user_id: 4,
    created_at: "2017-03-26 03:29:23",
    updated_at: "2017-03-26 03:29:23"
}
```

## UPDATE IMAGES USING LARAVEL

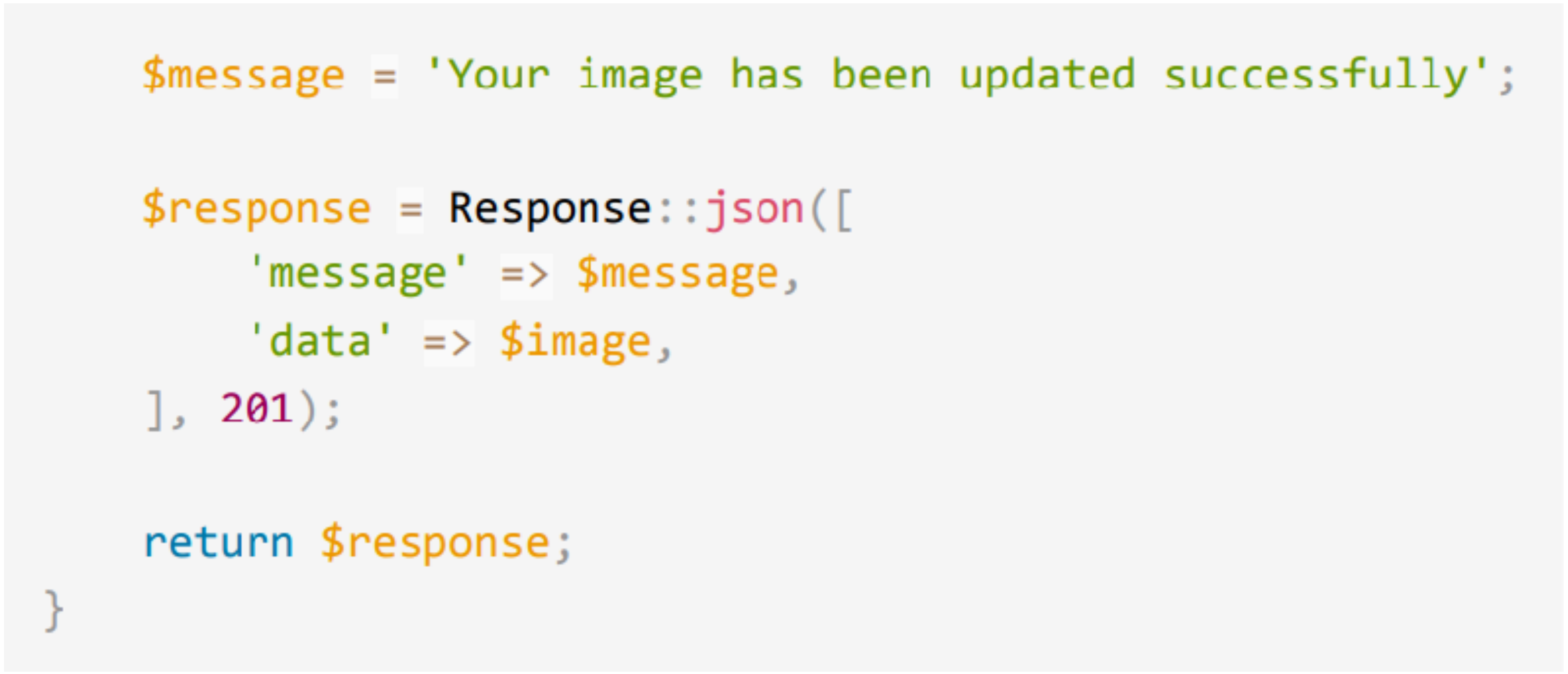
- Open ImagesController.php



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

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

    $image = Image::find($request->id);

    if(!$image){
        return Response::json([
            'error' => [
                'message' => "Cannot find the image."
            ]
        ], 404);
    }

    $image->thumbnail = trim($request->thumbnail);
    $image->imageLink = trim($request->imageLink);
    $image->title = trim($request->title);
    $image->description = trim($request->description);
    $image->save();
```

## UPDATE IMAGES USING LARAVEL

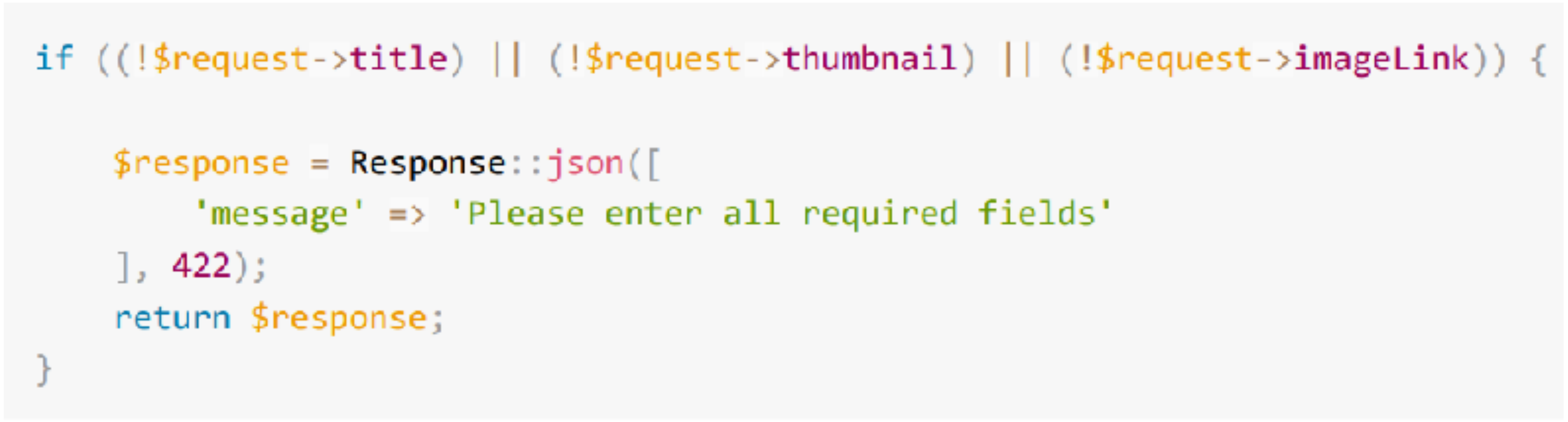
- Continue on ImagesController.php



```php
    $message = 'Your image has been updated successfully';

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

    return $response;
}
```

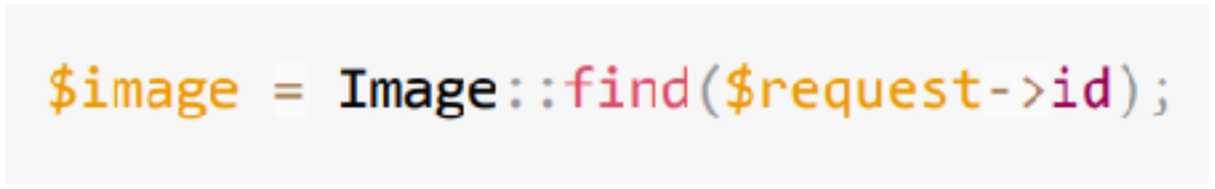
## EXPLANATION

- Just like image creation we check for empty fields



```
if ((!$request->title) || (!$request->thumbnail) || (!$request->imageLink)) {

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

- After we will find image based on id:



```
$image = Image::find($request->id);
```

## EXPLANATION

- If we cant find image we send back an error

```
if(!$image){
    return Response::json([
        'error' => [
            'message' => "Cannot find the image."
        ]
    ], 404);
}
```

- If everything is okay we save

```
$image->thumbnail = trim($request->thumbnail);
$image->imageLink = trim($request->imageLink);
$image->title = trim($request->title);
$image->description = trim($request->description);
$image->save();

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

## EXPLANATION

- Then we send a 201 response

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

## GETTING A SINGLE IMAGE USING ANGULAR

- When editing an image we have to send a request to the backend to get the image data so we can display information on our edit form

- Let us add getImage method to image.service.ts

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

- Open admin-image-edit.component.ts find:

```
this.params = this.activatedRoute.params.subscribe(params => this.id = params['id']);
```

## ADMIN-IMAGE-EDIT.COMPONENT.TS

- Add below:

```
this.imageService.getImage(this.id).subscribe(
  data => {
    console.log(data);
  },
  error =>  console.log(<any>error));
```

- Import the image service:

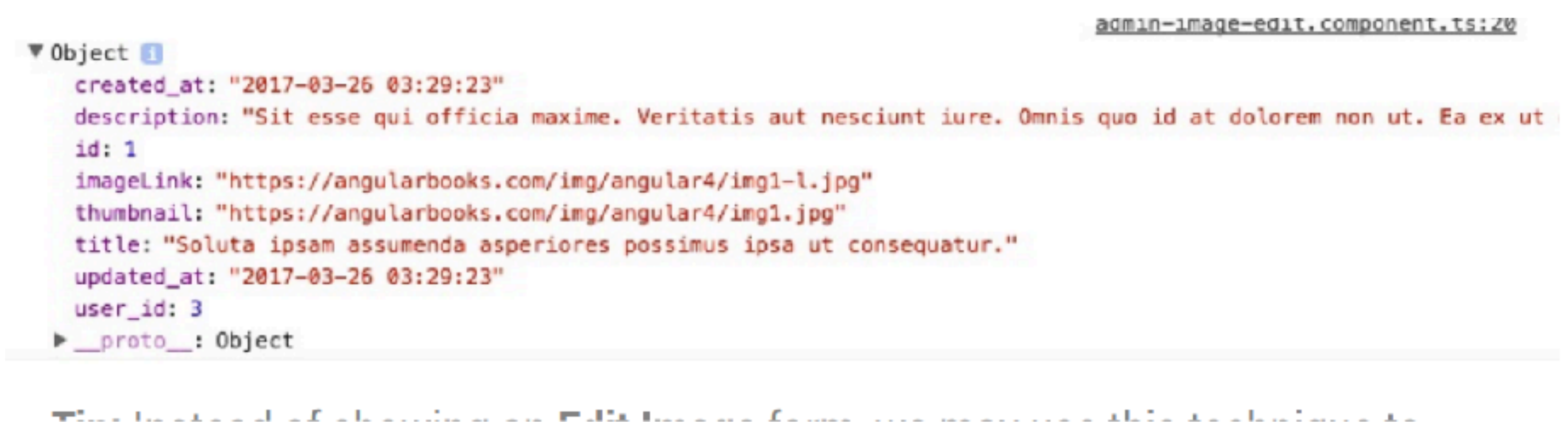
```
import {ImageService} from '../../services/image.service';
```

- Inject into our component:

```
constructor(private activatedRoute: ActivatedRoute, private imageService: ImageService)
 { }
```

- Visit http://localhost:4200/admin/images/edit/1

## CHECK CONSOLE.LOG



```
                                                        admin-image-edit.component.ts:20
▼ Object ℹ
    created_at: "2017-03-26 03:29:23"
    description: "Sit esse qui officia maxime. Veritatis aut nesciunt iure. Omnis quo id at dolorem non ut. Ea ex ut
    id: 1
    imageLink: "https://angularbooks.com/img/angular4/img1-l.jpg"
    thumbnail: "https://angularbooks.com/img/angular4/img1.jpg"
    title: "Soluta ipsam assumenda asperiores possimus ipsa ut consequatur."
    updated_at: "2017-03-26 03:29:23"
    user_id: 3
  ▶ __proto__: Object
```

Tip: Instead of showing an Edit Image form, we may use this technique to

## IT'S TIME TO SHOW THE DATA IN THE PAGE

- Two way data binding:
- Open admin-image-edit.component.ts find:

```
constructor(private activatedRoute: ActivatedRoute, private imageService: ImageService)
 { }
```

- Add above:

```
image = new Image('id', 'title', 'description', 'thumbnail', 'imageLink');
```

- Import Image model

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

## MORE ON THE DATA

- Find:

```
this.imageService.getImage(this.id).subscribe(
  data => {
    console.log(data);
```

- Add below:

```
this.image.description = data['description'];
this.image.title = data['title'];
this.image.imageLink = data['imageLink'];
this.image.thumbnail = data['thumbnail'];
this.image.id = data['id'];
```

## LET'S BIND OUR DATA TO THE FORM

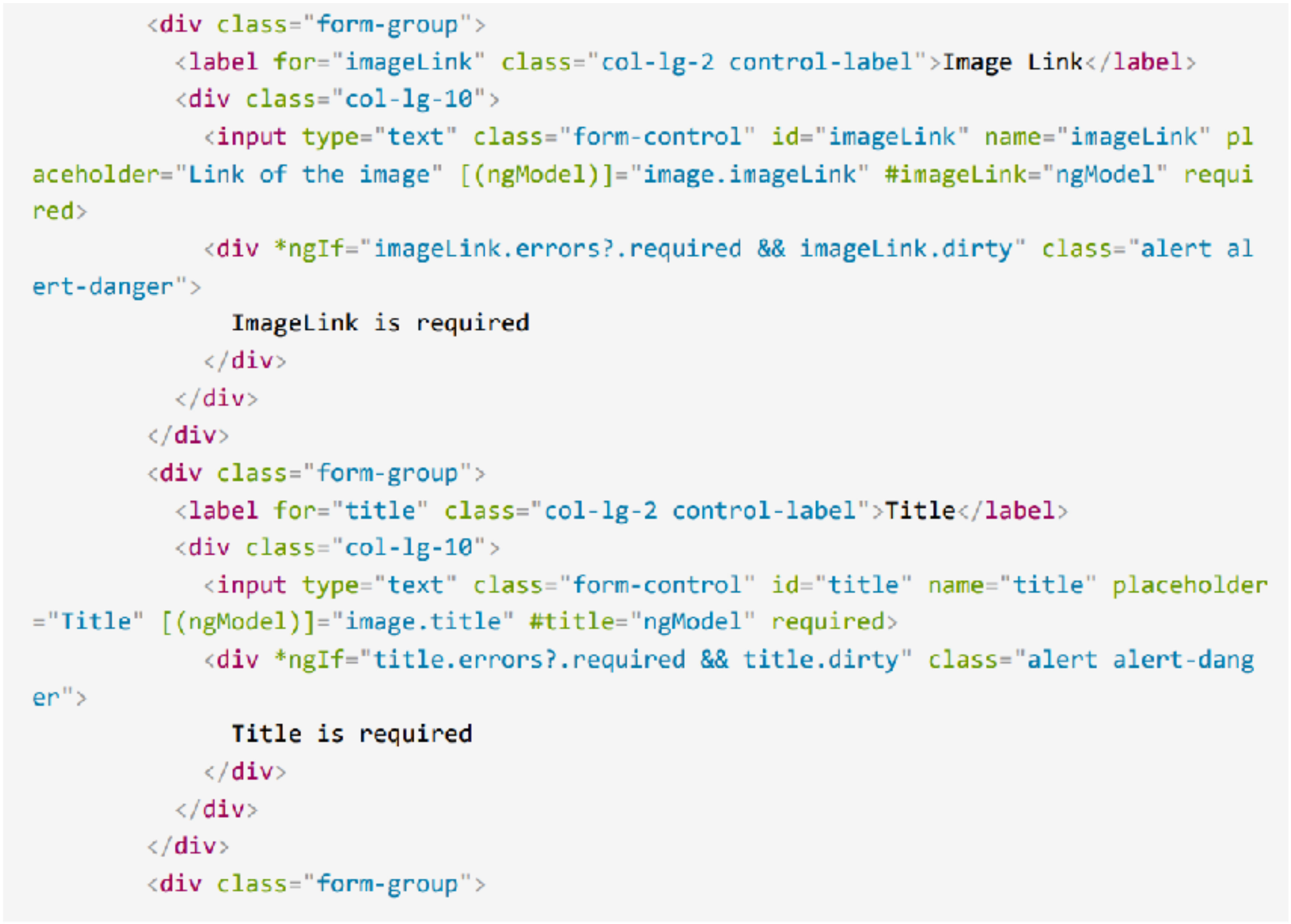
- Open admin-image-edit.component.html and update

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

# CODE CONTINUED

```
<div class="col-md-10 col-md-offset-1">
  <div class="well well bs-component">
    <form novalidate class="form-horizontal" (ngSubmit)="updateImage(image)" #editImag
eForm="ngForm">
      <fieldset>
        <legend>Updating: {{image.title}}</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)]="image.thumbnail" #thumbnail="ngModel"
required minlength="3">
            <div *ngIf="thumbnail.errors?.required && thumbnail.dirty" class="alert al
ert-danger">
              Thumbnail is required
            </div>
            <div *ngIf="thumbnail.errors?.minlength && thumbnail.touched" class="alert
 alert-danger">
              Minimum of 3 characters
            </div>
          </div>
        </div>
```

## CODE CONTINUED



```
          <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)]="image.imageLink" #imageLink="ngModel" requi
red>
              <div *ngIf="imageLink.errors?.required && imageLink.dirty" class="alert al
ert-danger">
                ImageLink is required
              </div>
            </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)]="image.title" #title="ngModel" required>
              <div *ngIf="title.errors?.required && title.dirty" class="alert alert-dang
er">
                Title is required
              </div>
            </div>
          </div>
          <div class="form-group">
```

## CODE CONTINUED

```
          <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"
 [(ngModel)]="image.description" #description="ngModel" required></textarea>
            <div *ngIf="description.errors?.required && description.dirty" class="aler
t alert-danger">
              Description is required
            </div>
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-10 col-lg-offset-2">
            <a [routerLink]="['/admin/']" class="btn btn-default"> Cancel</a>
            <button type="submit" class="btn btn-primary" [disabled]="editImageForm.in
valid">Update</button>
          </div>
        </div>
      </fieldset>
    </form>
  </div>
</div>
```

## MINOR CHANGE

- Similar to the createImage method update it to updateImage method:

```
<form novalidate class="form-horizontal" (ngSubmit)="updateImage(image)" #editImageForm
="ngForm">
```

- To bind our image to the input field:

```
<input type="text" class="form-control" id="thumbnail" name="thumbnail" placeholder="T
humbnail of the image" [(ngModel)]="image.thumbnail"
#thumbnail="ngModel" required minlength="3">
```

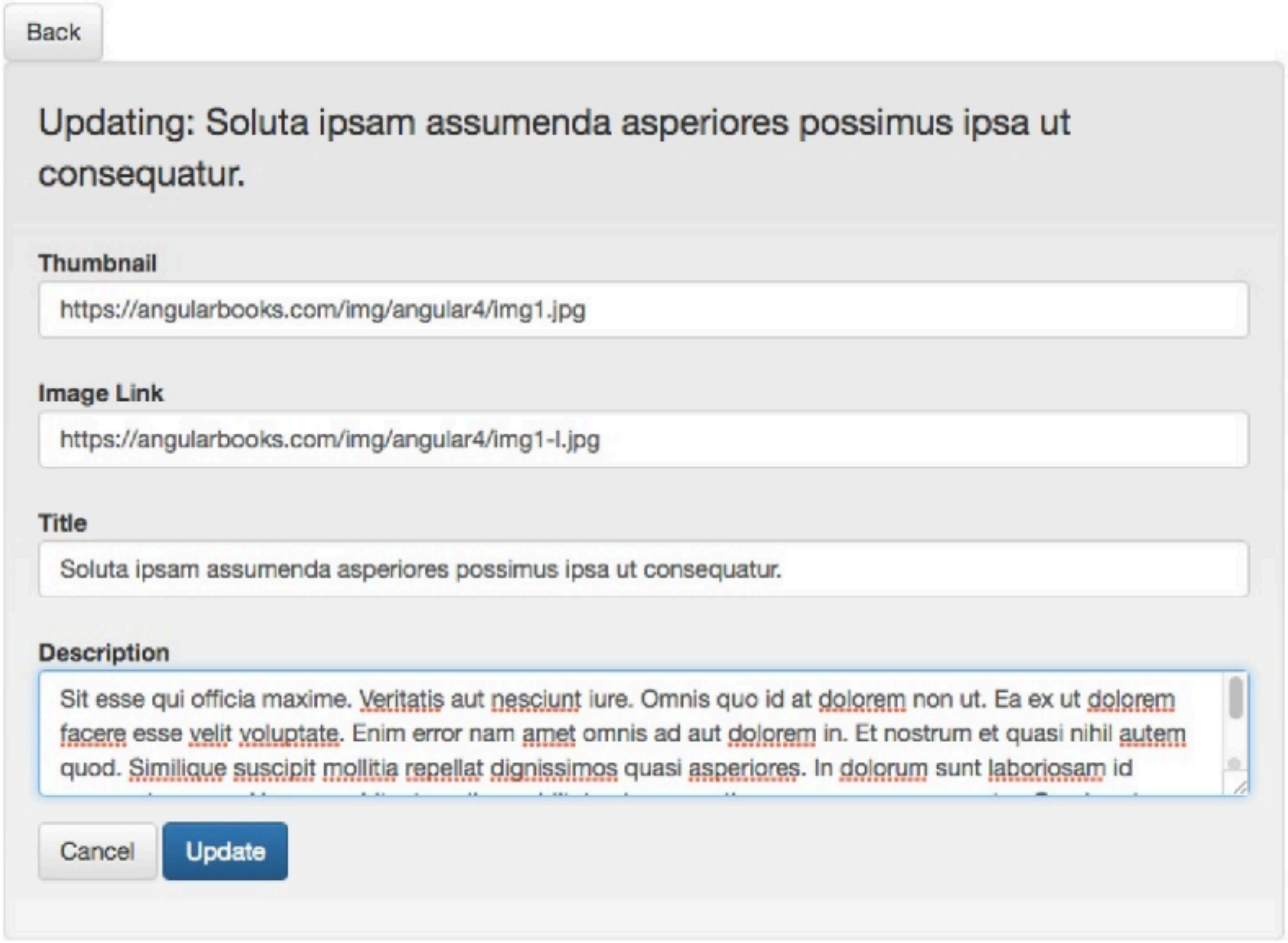
- This code means we are using data binding:

```
[(ngModel)]="image.thumbnail"
```

## WE DO THE SAME THING FOR OTHER FIELDS

```
<input type="text" class="form-control" id="imageLink" name="imageLink" placeholder="L
ink of the image" [(ngModel)]="image.imageLink" #imageLink="ngModel" required>

<input type="text" class="form-control" id="title" name="title" placeholder="Title" [(
ngModel)]="image.title" #title="ngModel" required>

<textarea class="form-control" rows="3" id="description" name="description" [(ngModel)]
="image.description" #description="ngModel" required></textarea>
```

## CHECK IN BROWSER


Back

Updating: Soluta ipsam assumenda asperiores possimus ipsa ut consequatur.

Thumbnail
https://angularbooks.com/img/angular4/img1.jpg

Image Link
https://angularbooks.com/img/angular4/img1-l.jpg

Title
Soluta ipsam assumenda asperiores possimus ipsa ut consequatur.

Description
Sit esse qui officia maxime. Veritatis aut nesciunt iure. Omnis quo id at dolorem non ut. Ea ex ut dolorem facere esse velit voluptate. Enim error nam amet omnis ad aut dolorem in. Et nostrum et quasi nihil autem quod. Similique suscipit mollitia repellat dignissimos quasi asperiores. In dolorum sunt laboriosam id

## LAST WORDS ON BINDING

- Check the title and the updating {{ title }}
- Thats 2 way data binding in action!

## BUT WAIT WE STILL NEED TO ADD THE UPDATEIMAGE METHOD

- Open admin-image-edit.component.ts and find:

```
ngOnDestroy() {
  this.params.unsubscribe();
}
```

- Add below:

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

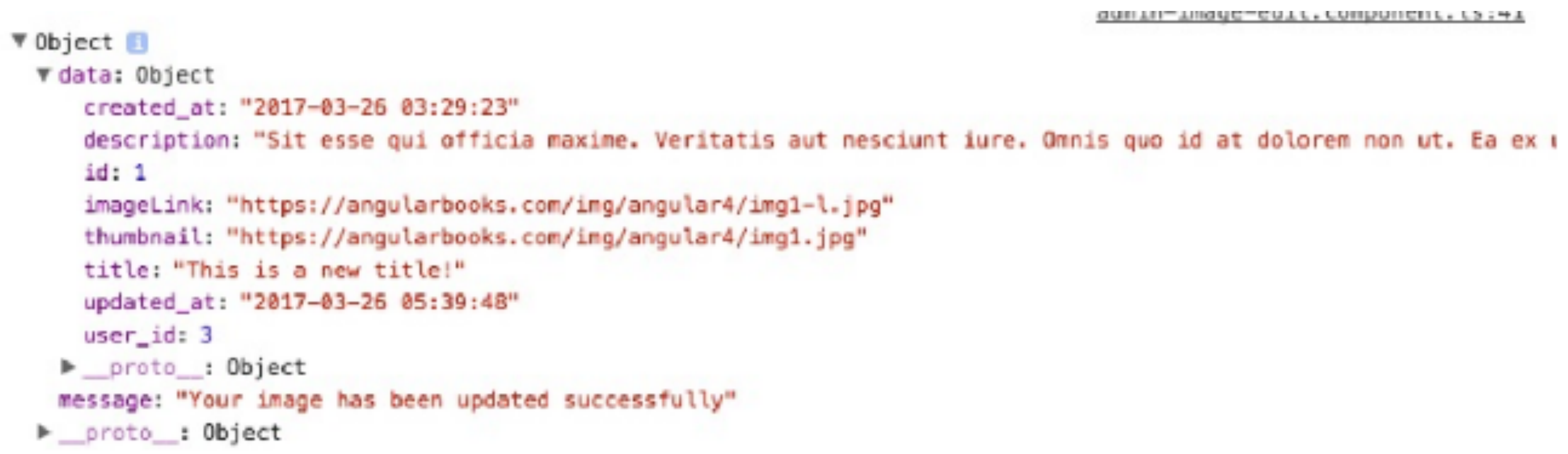
## CONTINUE

- We should add a new updateImage method to image.service.ts

```
updateImage(image: Object): Observable<Image[]> {
  const apiUrl = 'http://angularbook.app/api/v1/images';
  const url = `${apiUrl}/${image['id']}`;
  return this.http.put(url, image)
```

```
    .map((response) => response)
    .catch((error: any) => Observable.throw(error.error || {message: 'Server Error'}));

}
```

- Try to modify an image and click update and check console tab



```
                                                                admin-image-edit.component.ts:41
▼ Object
  ▼ data: Object
      created_at: "2017-03-26 03:29:23"
      description: "Sit esse qui officia maxime. Veritatis aut nesciunt iure. Omnis quo id at dolorem non ut. Ea ex
      id: 1
      imageLink: "https://angularbooks.com/img/angular4/img1-l.jpg"
      thumbnail: "https://angularbooks.com/img/angular4/img1.jpg"
      title: "This is a new title!"
      updated_at: "2017-03-26 05:39:48"
      user_id: 3
    ► __proto__: Object
    message: "Your image has been updated successfully"
  ► __proto__: Object
```

## DISPLAYING SUCCESS OR ERROR MESSAGES

- Open admin-image-edit.ts find:

```
export class AdminImageEditComponent implements OnInit {
```

- Add below:

```
status: string;
message: string;
```

- Edit updateImage method: ->

```
updateImage(image) {
  this.imageService.updateImage(image)
    .subscribe(
      image  => {
        console.log(image);
        this.status = 'success';
        this.message = image['message'];
      },
      error =>  {
        console.log(<any>error);
        this.status = 'error';
        this.message = error['message'];
      }
    );
}
```

## ADMIN-IMAGE-EDIT.COMPONENT.HTML

- Find:

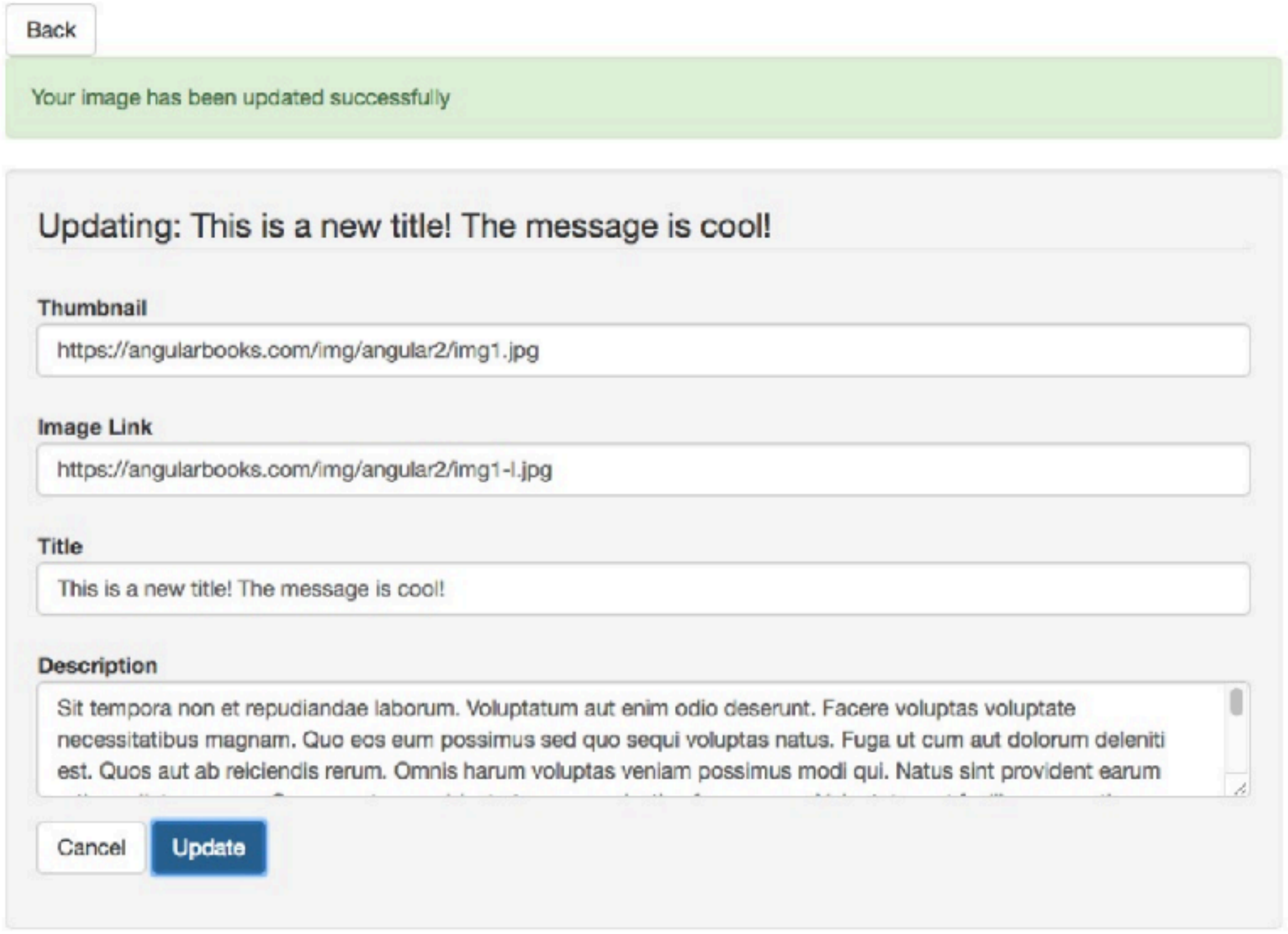
```
<div class="well well bs-component">
  <form novalidate class="form-horizontal" (ngSubmit)="updateImage(image)" #editImageF
orm="ngForm">
```

- Add above:

```
<div *ngIf="status=='success'" class="alert alert-success" role="alert"> {{ message }}
</div>
<div *ngIf="status=='error'" class="alert alert-danger" role="alert"> {{ message }} </
div>
```

- ngIf is used to show alert based on what was returned

## IF SUCCESSFUL



Your image has been updated successfully

Updating: This is a new title! The message is cool!

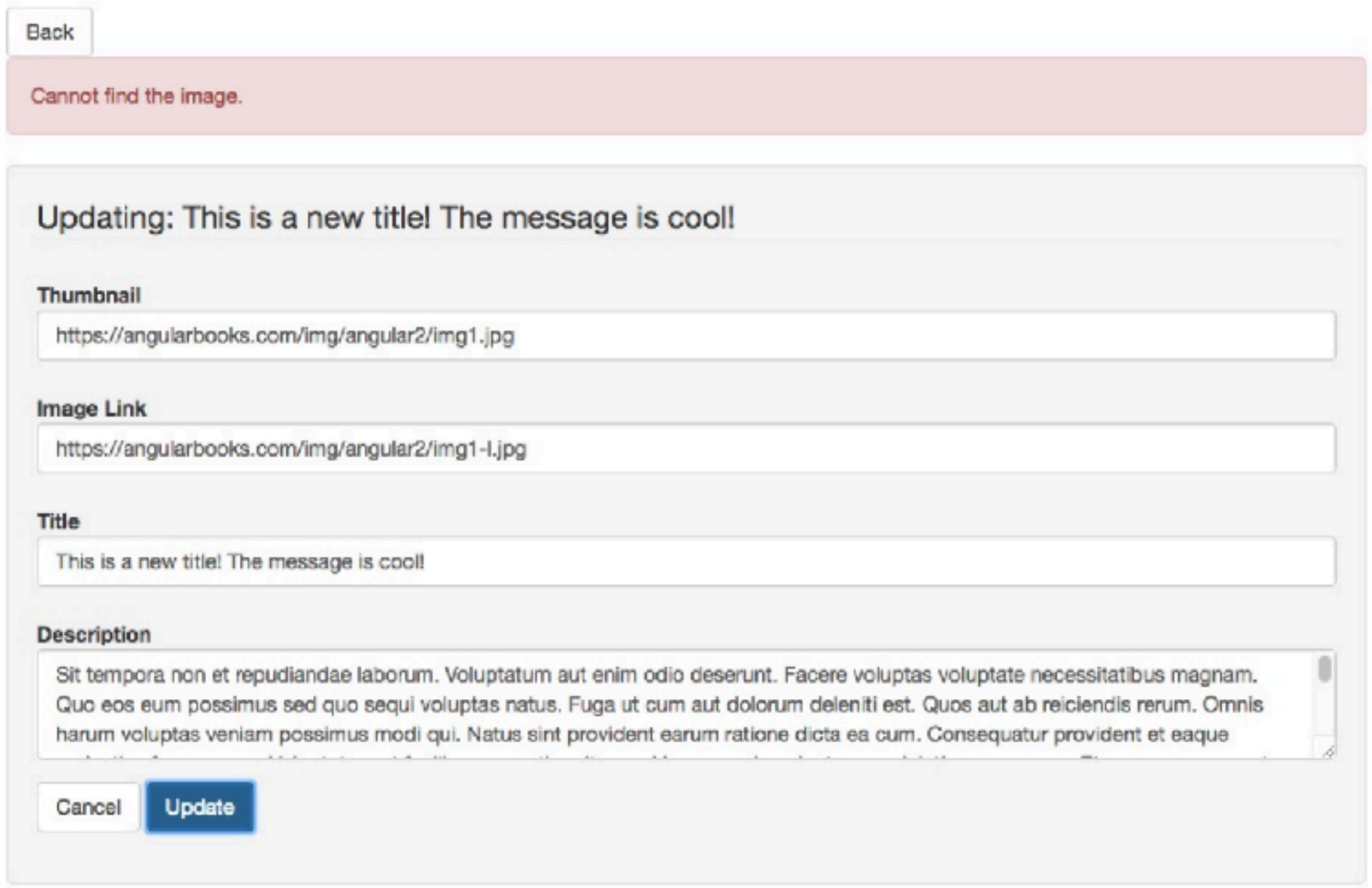
Thumbnail: https://angularbooks.com/img/angular2/img1.jpg

Image Link: https://angularbooks.com/img/angular2/img1-l.jpg

Title: This is a new title! The message is cool!

Description: Sit tempora non et repudiandae laborum. Voluptatum aut enim odio deserunt. Facere voluptas voluptate necessitatibus magnam. Quo eos eum possimus sed quo sequi voluptas natus. Fuga ut cum aut dolorum deleniti est. Quos aut ab reiciendis rerum. Omnis harum voluptas veniam possimus modi qui. Natus sint provident earum

## IF ERROR



Back

Cannot find the image.

Updating: This is a new title! The message is cool!

**Thumbnail**

https://angularbooks.com/img/angular2/img1.jpg

**Image Link**

https://angularbooks.com/img/angular2/img1-l.jpg

**Title**

This is a new title! The message is cool!

**Description**

Sit tempora non et repudiandae laborum. Voluptatum aut enim odio deserunt. Facere voluptas voluptate necessitatibus magnam. Quo eos eum possimus sed quo sequi voluptas natus. Fuga ut cum aut dolorum deleniti est. Quos aut ab reiciendis rerum. Omnis harum voluptas veniam possimus modi qui. Natus sint provident earum ratione dicta ea cum. Consequatur provident et eaque

## LAST CHANGES

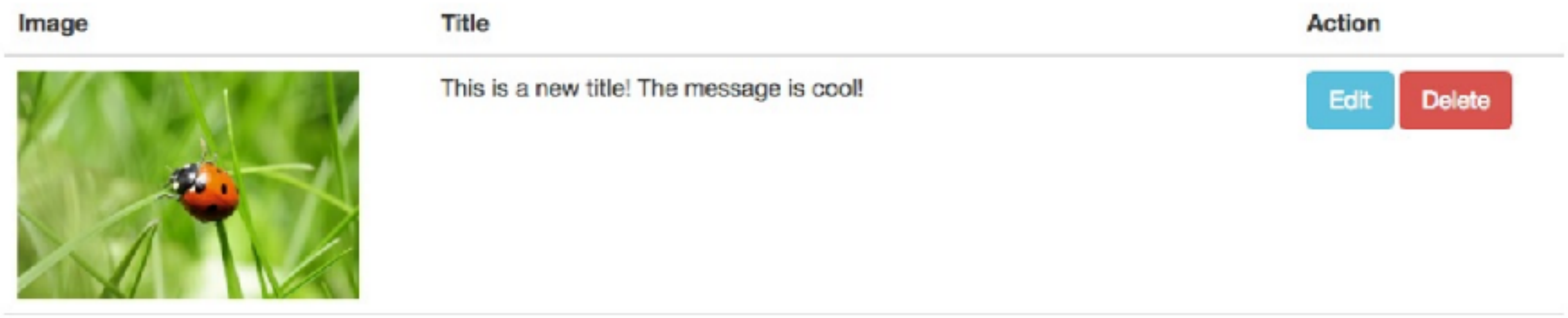
- Open admin-image-list.component.html Find:

```
<a [routerLink]="['/admin/images/edit']" class="btn btn-info"> Edit</a>
<a [routerLink]="['/admin/images/delete']" class="btn btn-danger"> Delete</a>
```

- Change to:

```
<a [routerLink]="['/admin/images/edit', image.id]" class="btn btn-info"> Edit</a>
<a [routerLink]="['/admin/images/delete', image.id]" class="btn btn-danger"> Delete</a>
```

## TEST EDIT BUTTON



| Image | Title | Action |
|---|---|---|
| | This is a new title! The message is cool! | Edit Delete |

END OF PART 15