Created by: Zoltán Richárd Jánki
Date: 15.04.2018

Angular 2+ Applications - Syllabus

Introduction

Introduction: The Angular 2 is a JavaScript framework used for creating web-applications. According to the conventions the recommended programming language is the TypeScript.

Design pattern: The Angular 2 framework uses the **MVC** (*Model-View-Controller*) design pattern. In an application we have a Model which contains the Business Logic. Mainly these are the classes of objects that we're using in our Object-Oriented Project. Since it's a web-application, we have a View part. This contains all the UI elements that are visible for the user. In our Angular 2 application, the View part will be the template, the actual HTML code. At the end we have a Controller, which works as a connector between the Model and the View.

E.g.: The user makes an action on the View and this is action is sent to the Controller. The Controller uses the Model in order to perform the action. Finally, the Controller sends back the informations to the View and make them visible on the View.

Creating an application

Angular CLI: The Angular CLI is a program that helps you to generate an empty project, and also add new components, modules, services to you existing application.

Firstly, install the Angular CLI globally in terminal:
```
$ npm install -g @angular/cli
```
(It is possible that the installation goes into an infinite loop. If it occurs, stop it with the CTRL+C combination, and follow the next step!)

If the installation has finished, check the installed version in terminal with the following command:
```
$ ng --version
```
(The up-to-date version of Angular CLI is 1.7.4.)

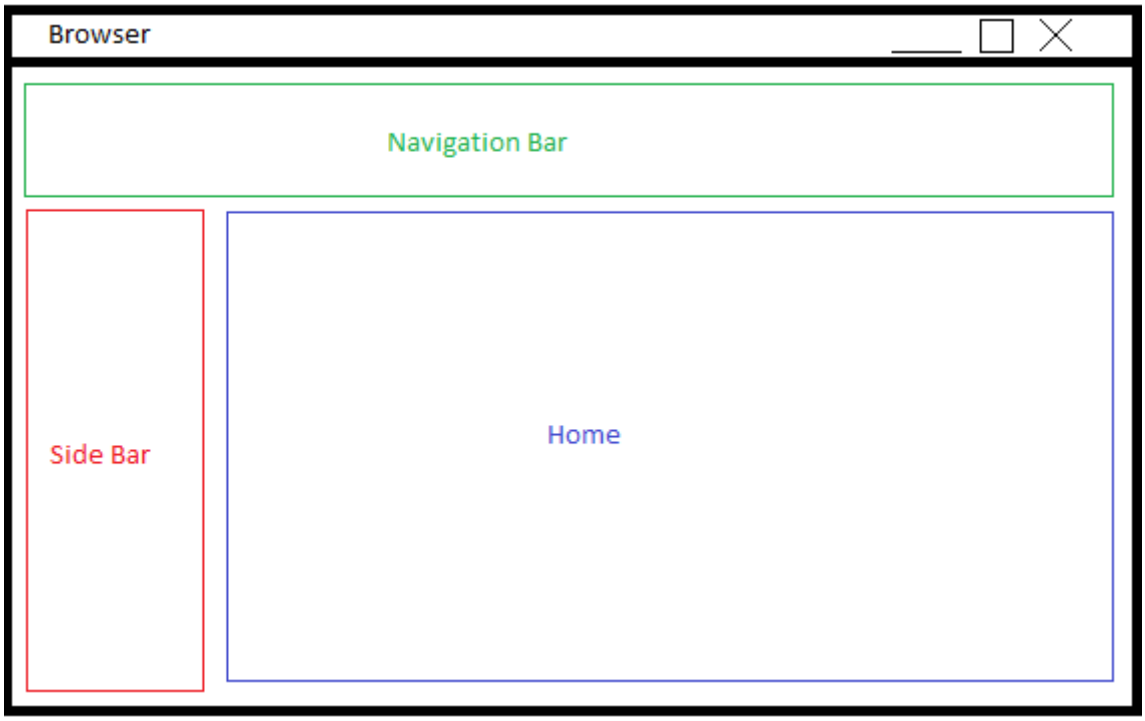
With the Angular CLI, we can generate a new – empty – project with the following command:
```
$ ng new project_name
```
(The generated project holds the recommended structure of the files and it contains only one component with test-cases.)

This generated project can be executed with the following command:
$ ng serve
(It is important that this command has to be executed in the root of our application, where the package.json can be found! After this command, the TypeScript compiler does the transpilation and makes the application available on localhost:4200. Open a browser, and open the app!)

Created by: Zoltán Richárd Jánki
Date: 15.04.2018

The structure of the project

Components: In Angular 2+ **everything is a component**. Let's see the following example!



We have a browser which opens our application. The UI have 3 different parts, **a navigation bar, a side bar and a home part**. All of them are different **Components** with different functionalities. All of them have its own ***Model, View and Controller***.
This is the main idea of the design pattern and the Angular 2 framework.

Module: a module is a comprehensive structure which contains the Components, the Routes and the Services. According to the conventions we should define a root module. This is the ***app.module.ts***.
What are the imports?

- *BrowserModule* - because we create a browser application
- *NgModule* – to define a module we have to use the `@NgModule({})` decorator
- *AppComponent* - this is the root component of our application (entry point)
- *MainComponent* – this is a user-defined component
- *AboutComponent* – this is another user-defined component
- *appRoutes* - all the routes that are defined in our application
- *RouterModule* - in order to use the routes defined in the appRoutes array

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { RouterModule } from '@angular/router';

import { appRoutes } from './app.routes';

import { AppComponent } from './app.component';
import { MainComponent } from './main/main.component';
import { AboutComponent } from './about/about.component';

@NgModule({
  declarations: [ AppComponent,
                  MainComponent,
                  AboutComponent
  ],
```

Created by: Zoltán Richárd Jánki
Date: 15.04.2018

```
  imports: [
    BrowserModule,
    RouterModule.forRoot(appRoutes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

As you can see, in the **`@NgModule({})`** decorator we define:

- declarations: the components that belong to this module
- imports: modules that are used in our application (mainly external ones)
  the **`RouterModule.forRoot(appRoutes)`** build up the route-tree from all the routes that are defined in the appRoutes array
- providers: here, we are going to define the services later
- bootstrap: if there's a component which is an entry point

Let's see the ***app.component.ts***!

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css],
})

export class AppComponent {
  title = 'app';
}
```

What are the imports here?

- *Component* - in order to define a component with the **`@Component({})`** decorator

What are defined in the @Component decorator?

- *selector*: we can refer to this Component in the View part with this tag **`(<app-root></app-root>)`**
- *templateUrl*: it refers to an HTML file that contains the View part of this component. The value of the templateUrl is the path of the proper HTML file

Routing definitions

Let's see the template (***app.component.html***)!

```
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="...">
  <router-outlet></router-outlet>
</div>
```

There is a short template with one greeting message (between **`<h1></h1>`**) and an image about Angular. The most important line is the 6th one. It contains a simple **`<router-outlet></router-outlet>`** tag pair. What is it?

Since this is the View part (template) of the AppComponent, it is responsible for some navigation that is defined in the appRoutes array (app.routes.ts). So, the **`<router-outlet></router-outlet>`** always brings a component into this component.

Created by: Zoltán Richárd Jánki
Date: 15.04.2018
Let's go to the ***app.routes.ts***!

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

import { MainComponent } from './main/main.component';
import { AboutComponent } from './about/about.component';

export const appRoutes: Routes = [
  { path: 'main', component: MainComponent },
  { path: 'about', component: AboutComponent },
  { path: '**', component: MainComponent }
];
```

The app.routes.ts file contains all the routes that are defined to this application. We have to create an array which is exported and it will be imported in the app.module.ts! The example works like the following:

- If we open **`localhost:4200/main`**, it navigates us to the MainComponent.
- If we open **`localhost:4200/about`**, it navigates us to the AboutComponent.
- If we open **`localhost:4200`**, it navigates us to the MainComponent.
- If we open **`localhost:4200/something`**, it navigates us to the MainComponent.

The first and the second routes are trivial. Since these routes are imported in the root module (AppModule), right after the localhost:4200 we can use the URL endings (like **`localhost:4200/main`**).
The third one works as a wildcard. The **`**`** means that if none of the above defined routes match, it navigates us to the MainComponent. It works as a default route.

Other components

As we've defined three different routes to two different components, we should create them. The creation can be done by using the Angular CLI. Use the following commands in terminal in order to generate two components in the **`app`** folder!

```
$ ng generate component main
$ ng generate component about
```

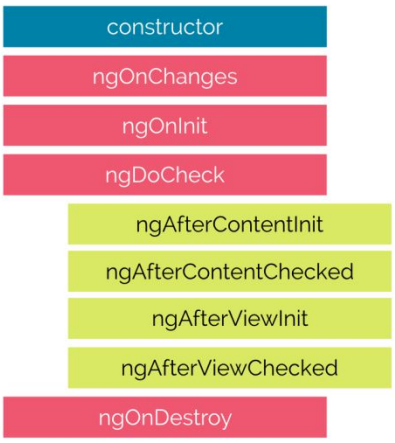
It created two components in the app folder and put everything into new folders.

- The main is available in **`app/main/main.component.ts`**.
- The about is available in **`app/about/about.component.ts`**.

Let's see the ***main.component.ts***!

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

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.css],
})

export class MainComponent implements OnInit {

  constructor() { }

  ngOnInit() { }
}
```

Created by: Zoltán Richárd Jánki
Date: 15.04.2018
The structure of this component is very similar to the original one of AppComponent, but there's an additional method called **`ngOnInit()`**. It is in connection with the Component Lifecycle Hooks. The following figure shows the basic lifecycle methods and the order of their execution. We can see that the **`ngOnInit()`** is executed after the **`constructor()`** and the **`ngOnChanges()`**.



Why is it good to have another method, that is executed at the beginning. The idea is to separate the functionalities of the constructor from the other parts. The constructor is responsible for the instantiation. In this case we are going to initialize the attributes of the components in the **`constructor()`** and we are going to put the other parts (e.g.: initial database queries) into the **`ngOnInit()`**.

If we check the AboutComponent, we can say that the structure and the content is the same except for the filenames.

Implementing the navigation in the View

We are going to introduce the navigation in Angular 2 in two different ways. It can be solved both in the View and in the Controller (in the Component). Firstly, we are going to implement it in the View part of the MainComponent.

Open the ***main.component.html*** file and extend it with the following code!

```
<p>
  main works!
  <a routerLink='/about' routerLinkActive="active">About</a>
</p>
```

The **`<a>`** tag can be familiar if you have ever edited an HTML file. It is responsible for hyperlinks. The **`routerLink`** option defines the destination of the route. It adds a **`/about`** to the URL.

With this solution we don't have to modify the component, only the View part. But, make sure that the route is defined in the **`appRoutes`**!

Implementing the navigation in the Component

For this solution we have to modify both the component code and the view code. The idea is to add an event to one of our DOM element (e.g.: to a button). The event will fire a method, that is responsible for the navigation.

First of all, create the view part (***about.component.html***)!

```
<p>
  about works!
  <button (click)="clickNavigation()">Click to navigate</button>
</p>
```

Created by: Zoltán Richárd Jánki
Date: 15.04.2018

In the example the first type of the navigation solutions is implemented, but now we are focusing on the event-driven one. We put a button labelled "Click to navigate" with the **<button></button>** tags. If we click on this button, it should navigate us to the Main component. The ***click*** is a DOM-event which is fired when we click on the used DOM element. It is put as an option in our tag with the **(click)** keyword. The value of the option is the name of the method that should be executed. In this case, this is the **clickNavigation()** method with no parameters.

Next, we have to implement this method in the component (***about.component.ts***).

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit() {
  }

  clickNavigation() {
    this.router.navigate(['/main', {id: "This is my id"}]);
  }

}
```

The **clickNavigation()** method is implemented as a method of the AboutComponent. Here, we have to use the Router component of the Angular Router module, that can handle the navigation. If we would like to instantiate another component, we can do it in the parameterlist of the constructor with ***private*** visibility. The router attribute has a **navigate()** method. Here, between array signs, we have to put the URL part, that should be added to our URL. Now, it is **/main**, because we would like to go to the MainComponent. After the URL part, we can define parameters as well. For example, if we would like to pass an id parameter with some value, we can do it here. The name of the parameter will be **id**, the value will be **"This is my id"**.

Another interesting question is that how can we receive and handle the sent parameters. These can be done by using the ActivatedRoute component of the Angular Router module. An example can be found in the ***main.component.ts*** file.

Created by: Zoltán Richárd Jánki
Date: 15.04.2018

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

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.css']
})
export class MainComponent implements OnInit {

  constructor(private router: Router, private route: ActivatedRoute) { }

  ngOnInit() {
    console.log(this.route.snapshot.params);
  }
}
```

Next week, we are going to continue with the data-binding and the services!