Angular Research

Nils Nieuwenhuis

Fontys of Hoghschool Techniek en Logistic, Dezember 7, 2017

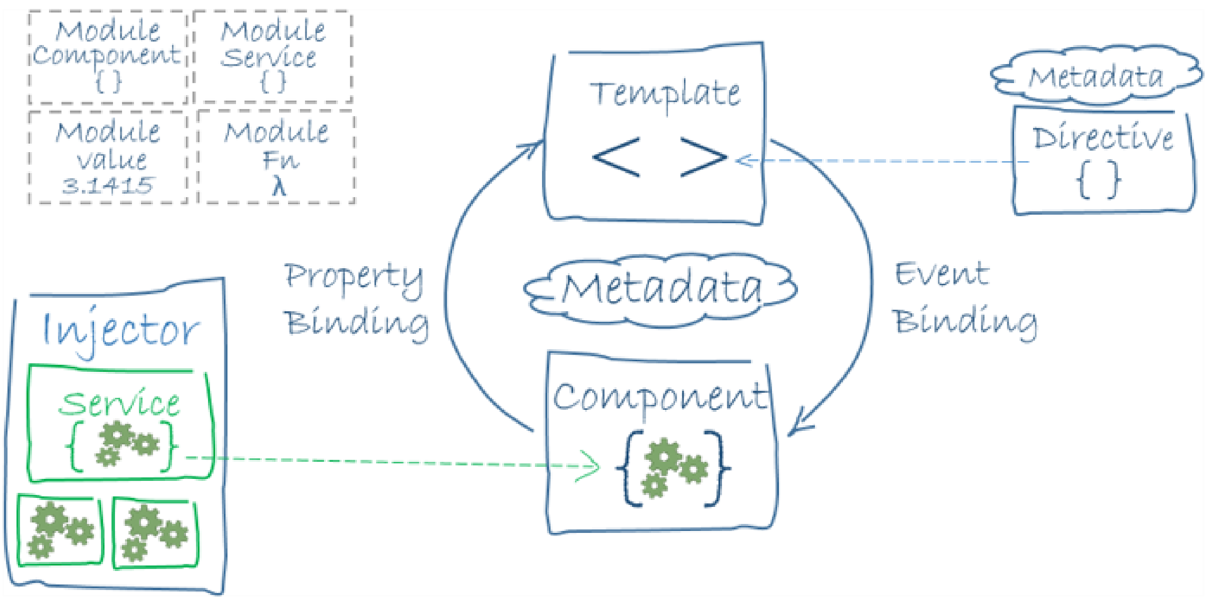
1. Introduction

This research is part of the SOFA module in Semester seven at the Fontys University of Applied Science. Our group of four students, three from the Software department and one from the business informatics department creating the “Fontys Module Managementsystem” for the Fontys University of Applied Science to view and edit Module informations. to accomplish that, we decided to use the Angular Framework for the Frontend. This research is about the Angular technics and common best practices we use to build the application.

2. Inhaltsverzeichnis

3. General Architecture

Angular is a framework for building client applications, it consists of several libraries, some of them core and some optional. You write Angular applications by combining HTML templates with Angularized markup, writing component classes to manage those templates, adding application logic in services, and boxing components and services in modules. The app is launched by bootstrapping the root module. Angular takes over, presenting your application content in a browser and responding to user interactions according to the instructions provided.



4. Modules

Angular apps are modular which is called “NgModules” in Angular. Every Angular app has at least one “NgModule” class the root module. An “NgModule” is a decorator, a decorator is function that modifys JavaScript classes, more about decorators later.

NgModule is a decorator function that has a single metadata object whose properties describe the module:

- **declarations:** the *view classes* that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
- **exports:** the subset of declarations that should be visible and usable in the component templates of other modules.
- **imports:** other modules whose exported classes are needed by component templates declared in *this* module.
- **providers**: creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
- **bootstrap:** the main application view, called the *root component*, that hosts all other app views. Only the *root module* should set this bootstrap property.

Here is an example of a typical root module:

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Libaries

Angular comes with a collection of javascript libraries, installed with the npm package manager. They are imported with the “import” statement and every angular library start with the “@angular” prefix. An import in angular looks like this:

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

5. Components

Components are the main way to build and specify elements and logic on the page, through both custom elements and attributes that add functionality to existing components. Everything in Angular is a component for example a List or a form.

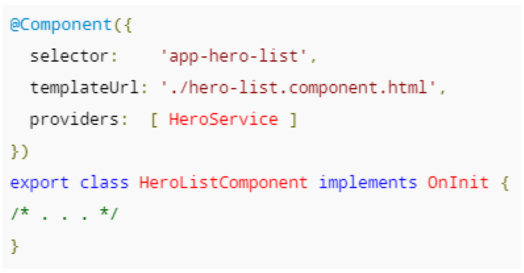
```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

To change a normal class into a component there is a “@Component” decorator, which adds some metadata to the class. These metadata tells Angular what is part of the Component, the most used configurations options are:

- **selector:** The selector is used to insert the component inside of the HTML code of a template.
- **templateUrl:** specifies the path to the template html file for the view.

- **providers:** An array of dependency injections for services that the component requires.

Here is an example of a component declaration:



```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

6. Templates

A Template is basically a view of a component and is built with regular HTML code. In addition to the regular HTML code there are special angular HTML tags. More of the angular HTML tags will follow in this research.

7. Data binding

Angular supports data binding with the possibility to combine parts of a template with parts of a component. Add binding markup to the template HTML to tell Angular how to connect both sides.

Angular supports four form of data binding:

- **Interpolation:** With this data binding method you can display a component property. To use it curly brackets are used, like this {{myComponent}}
- **Property Binding:** This method is used to fill a html tag with data from a component, for example: <img [src]=”myComponentURL”>
- **Event Binding:** To add an event handler to a template user action, for example a click this method is used: <button (click)=”myClickFunction()”>Click me!</button>
- **Two-way data binding:** With this method an asynchronous dataflow is realized. For example, the value of a component property is used for an input field and displayed in the template. Now if the user changes the value of the input field, the component property is changed too.

8. Directives

A directive is basically a class with a @Directive decorator or a like a component with a @Component decorator for the metadata. There are two important decorator types, the structural and attribute directives. Structural directives alter the layout by adding, removing and replacing elements in the DOM. Attribute directives changing the appearance or behaviour of existing elements.