Implementation of AngularJS in Developing Small Simple Website Project

Devin Lukianto - 13514040
Informatics / Computer Science Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514040@std.stei.itb.ac.id

***Abstract* — AngularJS is a JavaScript framework that can be used to help developer build a HTML web applications. There are many features provided by this framework that we can use, moreover to build a less refreshes and redirects website but with many tasks. To sum up, AngularJS is a great framework for us, the developers, to start making website project, especially the one which implements the Single Page Application concept.**

***Keywords* — AngularJS, framework, HTML, JavaScript, Single Page Application, website**

I. Introduction

AngularJS is a JavaScript framework released by Google in 2009. It is released as an open source-based JavaScript (.js) file that contains many functionalities as a framework to help the user develop HTML web applications. It is usually used to develop a Single Page Application website. [1]

Because it is distributed as a separated JavaScript file, it has to be added to the HTML file using <script> tag. As a framework, AngularJS also adopts the concept of MVC (Model, View, and Controller) though the implementation is a bit different from the usual MVC concept.

Figure 1. AngularJS Logo
(Source :https://d1067y8t86k9le.cloudfront.net/wp-content/uploads/2016/04/22150642/angular_js.jpg accessed on May 4th 2017, 13:27)

AngularJS has unique implementation in HTML. It actually extends the usage of HTML attributes Directives, and it can bind the data we use using Expressions. Further explanations of these two elements will be discussed on the next chapter.

II. Basic Theory

There are two main syntax that will be used throughout the project:

A. Expressions

Expressions are used when we want to "output" the data. We use double braces to state an AngularJS expression: `{{expression}}`. Uniquely in AngularJS, expressions can also be used to do some calculation automatically, for example an expression `{{ 5*3 }}` will give an output `15` rendered in our HTML page. Therefore, it can give us some flexibility and benefits in developing HTML website applications.

B. Directives

In this paper, we are going to discuss some of AngularJS' directives that is usually called as ng-directives.[2] These are some examples of ng-directives: ng-app, ng-model, ng-init, ng-controller, ng-view, ng-include, ng-src, and ng-repeat. In a brief, these are the explanations of some directives we will use in this project:

1. ng-app

   This directive has the functionality of telling the AngularJS that the scope of the HTML belongs to this root application. There can be only one ng-app in a whole HTML document. If there are more than two ng-app directives, the first appearance will be used.

2. ng-model

   This directive has the functionality of binding the data view from other directives such as `input`, `select`, or `textarea`, to a model property on the relevant scope.

3. ng-init

This directive has the functionality of giving an evaluation of an expression in the current scope.

4. ng-controller[3]

This directive will add a controller class to the view. It can be used to control the data of AngularJS applications. We can add some functions and values in a scope with this controller.

5. ng-view

This directive has the functionality as a complement of routing feature in AngularJS. It will include the rendered template of current files in the current scope route to the main HTML layout file.

6. ng-include

In a brief, this directive has the functionality of fetching, compiling, and rendering the external fragment of HTML files.

7. ng-src

This directive will directly override the original `src` attribute of an `<img>` element in HTML so that the user will be able to user AngularJS code inside or as the `src` value of an image.

8. ng-repeat

This directive access and instantiate a template once per item from a collection of data. Each template instantiation will get its own scope where the variable can be set to that scope item using loop.

III. Implementation

In this paper, we are going to make a simple bakery website applications[3]. This website will consist of one main page, containing the list of products. Below here is the figure of the website's layout we are going to make.

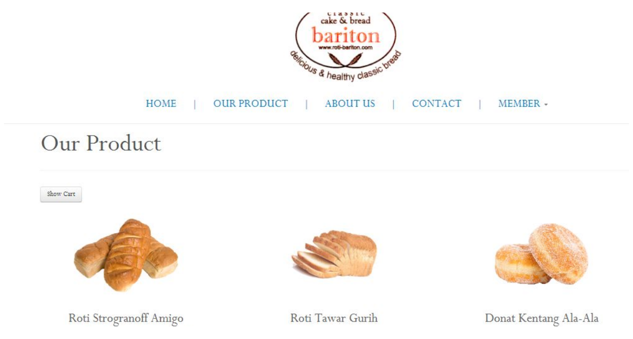


Figure 2. Bakery Website Layout
(Source: "http://product.html ", created by Author on May 2nd 2017)

Before we implement all the directives method in this project, make sure that we have implemented ng-app directive in in the HTML tag like this:

```
<html lang="en" ng-app=productApp>
```

A. Templating

As we can see in the Figure 2, the website has a menubar section which will look the same in all pages in this website project. Also, assume there is a section of style overriding code in `<style>` tag that we want to include in every HTML page we have. Therefore, we can make a template of those two sections using ng-include. But, we can also use Custom Directive as an alternative to include another HTML to our main HTML.

a. Using ng-include

We will include our custom style override in `<style>` tag using ng-include. First, we can type all part of codes which belong to `<style>` tag in a separated HTML file for example: "style.html", and include this file into our main "product.html" file. Then, we use ng-include in "product.html" with syntax:

```
<div ng-include="'style.html'"></div>
```

This syntax will be automatically interpreted as an include process by AngularJS, so it will include all the content of "style.html" into our "product.html".

b. Using Custom Directive

We can also include another HTML page to one HTML page using a method called Custom Directive. The main idea of Custom Directive is to make a custom tag as a sign of our included part of code. We can use custom element or attribute in this method.

In this case, we would like to use custom element to include our menubar template from a separated HTML file called "menubar.html" Therefore, after typing all the menubar code in "menubar.html" file, we can type a new custom element tag in our "product.html" body, for example:

```
<div-menubar></div-menubar>
```

Next, we have to make a new module corresponding to the ng-app we made before. Also, we have to implement our "divMenubar" element in that script section with this code:

```
<script>
  //MODULE
```

```
var app = angular.module('productApp',
['ngRoute']);

  //DIRECTIVE
  app.directive('divMenubar', function(){
      return{
        restrict : 'E', //E means Element
        templateUrl : "menubar.html" }
  });
</script>
```

This script will automatically run soon the page is loaded. Thus, it will include all the content of "menubar.html" into our "product.html".

B. Data Binding Expression and Controller

In this section, we would like to show our products in a `div` and interface we've made. Basically, showing products process is just repeating the render of interface code, with different product name or id in every repetition. With AngularJS, we can implement ng-repeat so that we don't have to type the same code many times.

In this case, we can keep our products data in our controller we will define. So firstly, we have to declare the ng-controller directive in the scope where the ng-repeat directive is also placed in the part of code where we want the repetition to occur.

```
<div class="container" ng-controller =
"productCtrl">
 <div class="row">
  <div class="span4" ng-repeat="dr in
daftarroti">
   <div class="media">
    <div class="span3 centeralign">
     <a class="centeralign"><img
class="imgadjustproduct" ng-src="{{
dr.link }}" class="media-object" alt=''
/></a>
    </div>
    <div class="span3">
    <div class="media-body">
     <h4 class="rotifontsize">{{ dr.nama
}}</h4>
     <p class="centeralign">{{ dr.desc
}}</p>
     <p class="centeralign">Harga: IDR
{{ dr.harga }}</p>
    </div>
   </div>
  </div>
 </div>
</div>
```

We mustn't forget that we have to declare a template variable to clone the content of the data from our controller. Therefore, we add string parameter in ng-repeat with `dr` in `daftarroti`, where `dr` is our template variable and `daftarroti` is the name of our data scope in the controller. To access the sub-data, we can just use dot (.) followed by the name of data we would like to get and show in our expression (i.e. `dr.nama, dr.desc`, etc.).

Secondly, we have to define our controller and our product data. Thus, in our script part, we can now add our controller:

```
//CONTROLLER
app.controller('productCtrl',
function($scope) {
      $scope.daftarroti = [
{nama : 'Roti Strogranoff Amigo', harga
: '8000', link : 'img/bread.png', desc :
'Roti ala Perancis, kini hadir di
Bandung!'},
{nama : 'Roti Tawar Gurih', harga :
'9000', link : 'img/bread2.png', desc :
'Tawar, namun gurih di lidah'},
{nama : 'Donat Kentang Ala-Ala', harga :
'18000', link : 'img/bread3.png', desc :
'Donat kentang dipanggang dadakan'},
{nama : 'Yellow Sponge Cake', harga :
'24000', link : 'img/cake.png', desc :
'Spongebob kini hadir di mulut anda'},
{nama : 'Tart Birthday', harga :
'90000', link : 'img/tart.jpg', desc :
'Ulang tahun? Beli tart ini pasti
cocok!'},
{nama : 'Roti Coklat Normal', harga :
'5000', link : 'img/roticoklat.jpg',
desc : 'Roti coklat kasual
sehari-hari'}];
});
```

C. "Add to Cart", using ng-click, Function, and Routing

In this section, we will make shopping cart in the same page as our main product page. We will utilize the Single Page Application feature from AngularJS here, where we can show or hide shopping cart in the same page, and add product to that shopping cart as well.

First, we have to make the code to render the shopping cart. We are still using the same method as part A and B: implementation of controller and Custom Directives. Therefore, we make the code of the shopping cart in the different file separated from the main "product.html" file.

In the controller, we have to make another scope variable to save the shopping cart data. It can be implemented with this code inside the controller:

```
        $scope.daftarkeranjang = [];
```

After that, we can make a button that can be used by the users to add product to their shopping cart. We have to implement the ng-click here, where the ng-click will access the function we give:

```
        <a class="btn" type="button"
  ng-click="addToCart(dr.nama, dr.harga,
        dr.link)">Add to Cart!</a>
```

and in controller `<script>` tag:

```
$scope.addToCart = function(name, price,
url){
        $scope.daftarkeranjang.push({
            nama: name,
            harga: price,
            link: url
        });
   };
```

The addToCart function have 3 parameters that will be passed into the controller. The data passed has to be saved in the "daftarkeranjang" array we've made before.

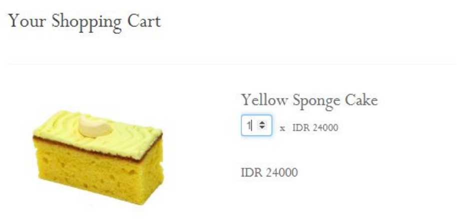
Because we want the shopping cart to be shown in the same page with only a word added in the URL (i.e. product.html/cart), we have to use router. The router itself can be implemented inside the script tag in "product.html" like this code:

```
 //ROUTING
  app.config(function($routeProvider){
    $routeProvider
      .when('/cart',{
      templateUrl : "shoppingcart.html"
      })
      .otherwise({redirectTo: '/'});
  });
```

To ease the user to show the shopping cart, we can also add a new button "Show Cart" with this code:

```
        <a class="btn" type="button"
 href="product.html#!/cart">Show Cart</a>
```

After doing all those steps, we can now show the shopping cart using the button we made, and add products we want to buy into the shopping cart directly in one page.



Your Shopping Cart

Yellow Sponge Cake

1 x IDR 24000

IDR 24000

Figure 3. Bakery Website Shopping Cart
(Source: "http://product.html/cart ", created by Author on May 2nd 2017)

IV. Summary

AngularJS is a great JavaScript framework that can be used by developers to make a static website applications. Its features make the implementation of Single Page Application website possible so that it will reduce the number of refreshes or redirects occur when the users are browsing the pages. Users can do many tasks without having to be redirected to another page, or change the page they are browsing.

References

[1] https://www.w3schools.com/angular/ accessed on May 3rd 2017, 19:23

[2] https://docs.angularjs.org/api/ng/directive accessed on May 3rd 2017, 20:15

[3] https://www.youtube.com/watch?v=YqdG-K2FDMY&list=PLCZlgfAG0GXBD2nko3bXRs-JSDNA8tNvq accessed on May 1st 2017, 21:48.

Acknowledgment

I would like to show my gratitude to Mr. Rinaldi Munir as my lecturer for giving this task in order to push us to learn something new. I also thank my colleagues from Institut Teknologi Bandung who provided insight for the topic and project for this paper.

With this, I declare that this paper is an original work and not a form of plagiarism.

Bandung, May 5th 2017

[illegible]

Devin Lukianto
13514040