Tutorial And Example

A Tutorial Website with Real Time Examples

Angular 8 Components

July 13, 2019

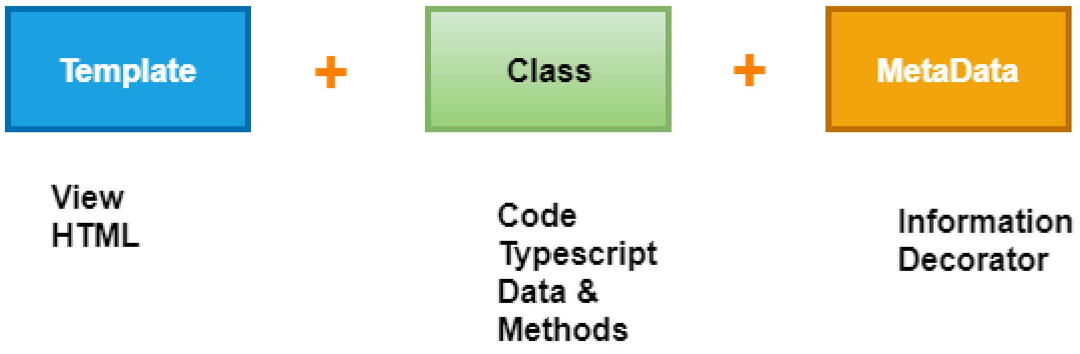
by

Angular is used for *building mobile and desktop web applications*. The component is the **basic buil block of Angular**. It has a selector, template, style, and other properties, and it specifies the metad required to process the component.

Components are defined using a **@component** decorator and a tree of the angular component. It makes our complex application in reusable parts which we can reuse easily.

The component is the most critical concept in Angular and a tree of components with a root component. The root component is one contained in the bootstrap array in the main **ngModule** module defined in the **app.module.ts** file.

One crucial aspect of components is re-usability. A component can be reused throughout the application and even in other applications. Standard and repeatable code that performs a specific task can be encapsulated into a reusable component.



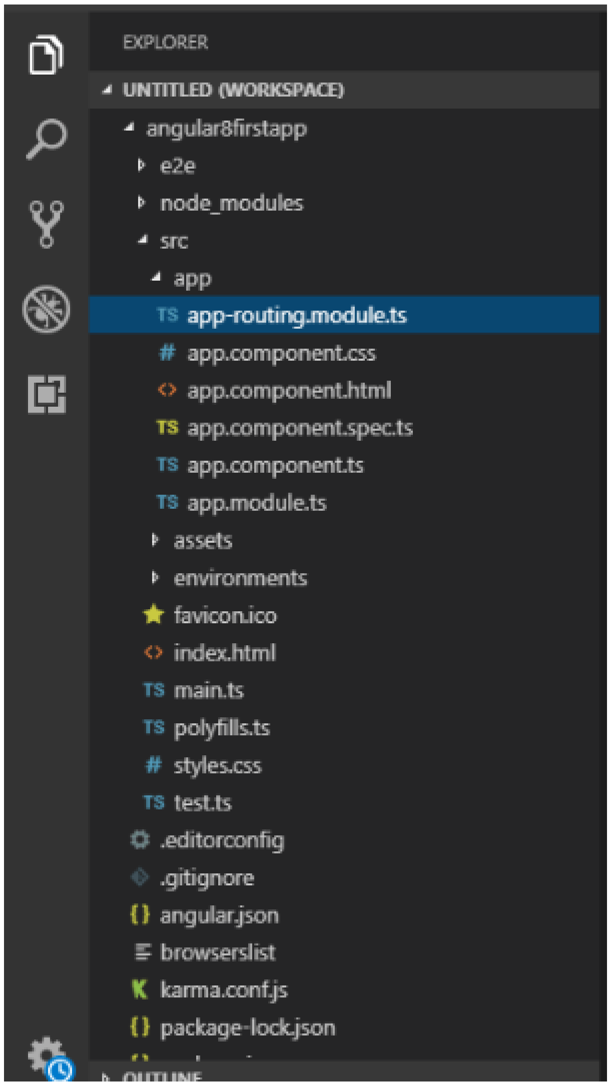
What is Component-Based Architecture?

A component is an independent block of an extensive system that communicates with the other building blocks of the systems using inputs and outputs. It has an associated view, data, and behavior and has parent and child components.

The component allows maximum re-usability, secure testing, maintenance, and separation of concerns. Go to our Angular project folder and open the src/app folder, which we will see the following files.

App folder: The app folder contains the data we have created for app components.

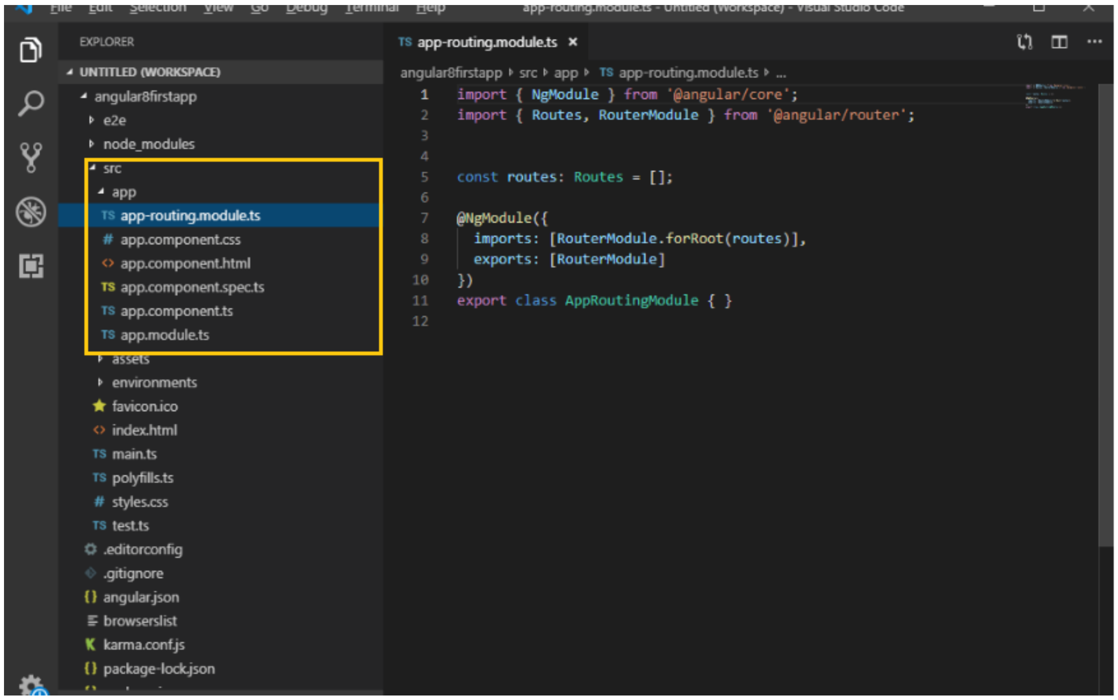
- **app.component.css**: The component CSS file.
- **app.component.html**: This component is used for HTML view.
- **app.component.spec.ts**: The HTML view of the component
- **app.component.ts**: Component code (data and behavior)
- **app.module.ts**: The main application module.



```
EXPLORER
UNTITLED (WORKSPACE)
  angular8firstapp
    e2e
    node_modules
    src
      app
        app-routing.module.ts
        app.component.css
        app.component.html
        app.component.spec.ts
        app.component.ts
        app.module.ts
      assets
      environments
      favicon.ico
      index.html
      main.ts
      polyfills.ts
      styles.css
      test.ts
    .editorconfig
    .gitignore
    angular.json
    browserslist
    karma.conf.js
    package-lock.json
OUTLINE
```

Default Code:

```
Import { Component } from ‘@angular/core’;
 @Component ({
 Selector: ‘app-root’,
 templateUrl: ‘./app.component.html’ ,
 styleUrls: [‘./app.component.css’]
 })
 export class AppComponent {
 title = ‘myfirstapp’;
 }
```



```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';


const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Firstly, we import the Component decorator from **@angular/core** and then we use it to preserve the Typescript AppComponent class. The Component decorator takes an object with multiple parameters, however:

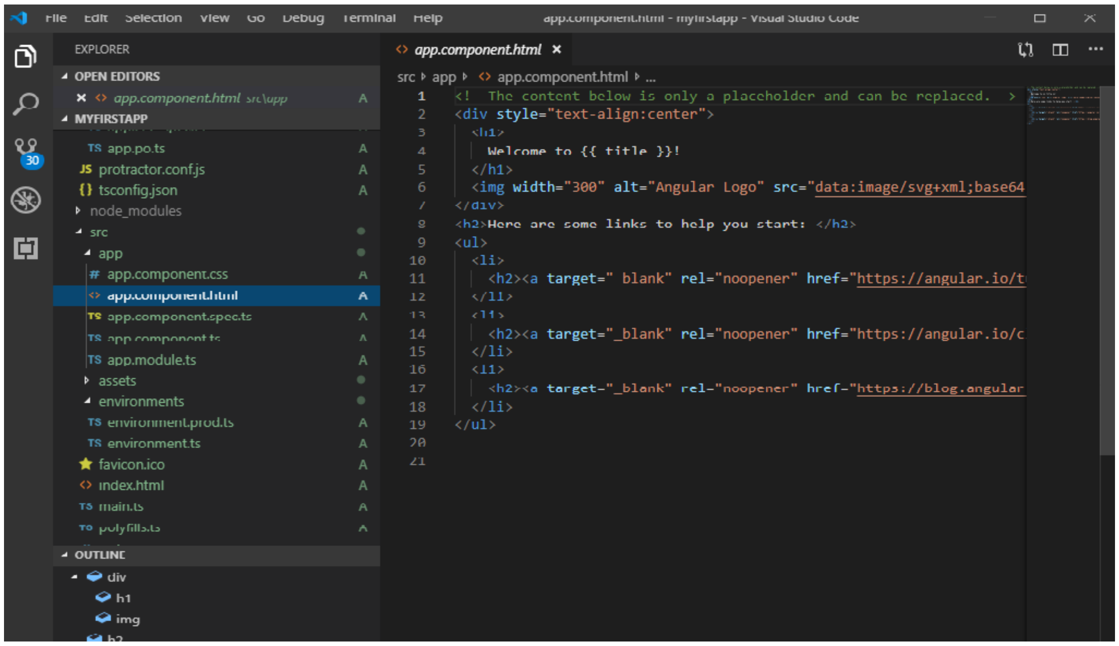
- **selector:** It specifies the tag that can be used to call this component in HTML templates just like the standard HTML tags
- **templateUrl:** It indicates the path of the HTML template that will be used to display this component.
- **styleUrls:** It specifies an array of URLs for CSS style-sheet in the component.

The export keyboard is used to export the component, and it has imported from other components and modules in the application file.

Let's see the corresponding template for that component. If we open **src/app.component.html**

Default Code:

```
<div style="text-align: center">
<h1>
Welcome to!
</h1>
<img width="300" alt="Angular Log" src="data: image/svg+xml;….">
</div>
<h2>Here are some links to help you start:</h2>
<ul>
<li>
<h2><a target="_blank" rel="noopener noreferrer">
https://angular.io/tutorial</a>">Tour of    Heroes</a></h2>
</li>
<li>
<h2><a target="_blank" rel="noopener noreferrer" href="https://github.com/angular/angu
</li>
<li>
<h2><a target="_blank" rel="noopener noreferrer" href="https://blog.angular.io">Ang
</li>
</ul>
```



```
<!  The content below is only a placeholder and can be replaced.  >
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="data:image/svg+xml;base64
</div>
<h2>Here are some links to help you start: </h2>
<ul>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/t
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/c
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://blog.angular
  </li>
</ul>
```

The template is an HTML file is used inside an HTML template except for the following tags <script>, <html>, and <body> with the exception that it contains template variable or expression and it can be used to insert values in DOM automatically. It is called **interpolation** and **data binding.**

This is the basic building block of Angular.

Open **VS code** then go to your project source folder then expand the app directory and create a new list named '**server**.'

Now, create a component in the server directory. Right-click on server directory and create a new file named as 'server.component.ts.' It is the recently created component.

Components are used to build webpages in all versions of Angular, but they require modules to bundle them together. Now, we have to register our new component in the module.

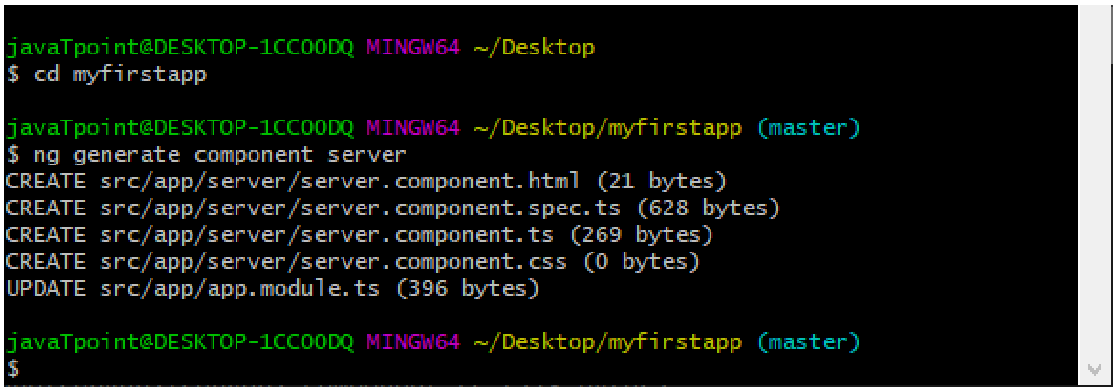
Creating a component with CLI

Syntax:

```
ng generate component component_name
Or
ng g c Component-name
```

Let's see how to create an element in the command line.

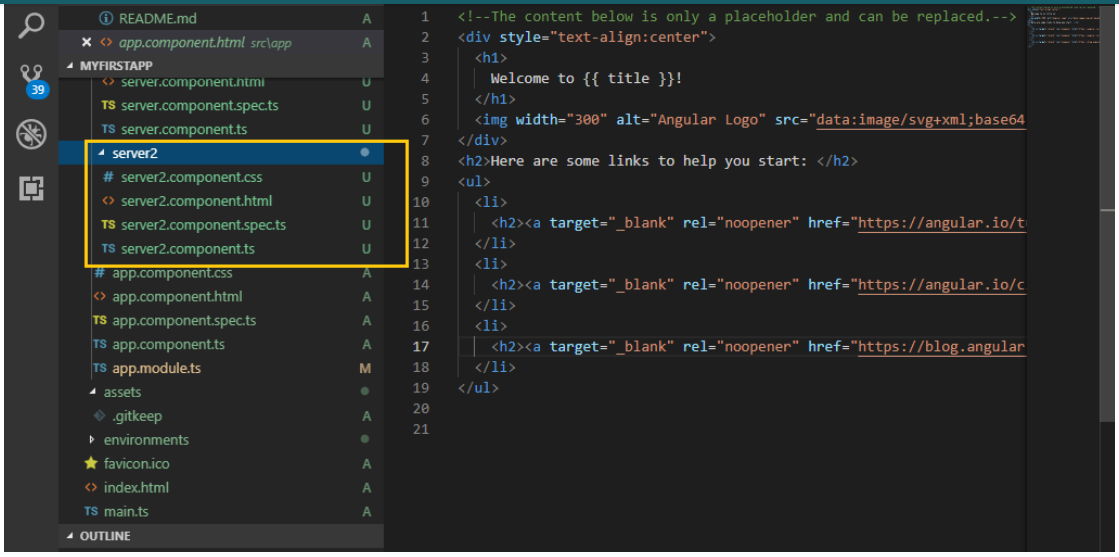
Open the command prompt and stop **ng serve** Command if it is running on the browser. Type **ng generate component server (**where the server is the name of the component we are created to create a new component named server2). We can also use a shortcut command**ng g c server** for the same task. Firstly, we have to open our Project with **cd myfirstapp** and then create the component **server2.**



```
javaTpoint@DESKTOP-1CCOODQ MINGW64 ~/Desktop
$ cd myfirstapp

javaTpoint@DESKTOP-1CCOODQ MINGW64 ~/Desktop/myfirstapp (master)
$ ng generate component server
CREATE src/app/server/server.component.html (21 bytes)
CREATE src/app/server/server.component.spec.ts (628 bytes)
CREATE src/app/server/server.component.ts (269 bytes)
CREATE src/app/server/server.component.css (0 bytes)
UPDATE src/app/app.module.ts (396 bytes)

javaTpoint@DESKTOP-1CCOODQ MINGW64 ~/Desktop/myfirstapp (master)
$
```

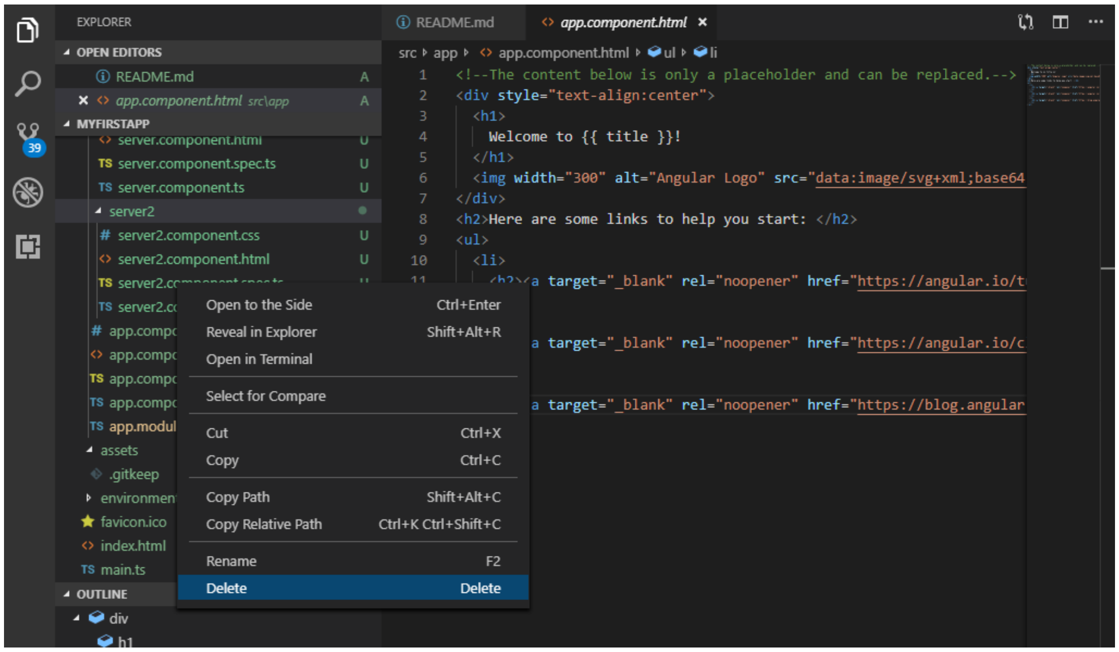


```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="data:image/svg+xml;base64
</div>
<h2>Here are some links to help you start: </h2>
<ul>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/t
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/c
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://blog.angular
  </li>
</ul>
```

In the above screenshot, we see that a new component named “server2” is created. It contains the same other components which we have seen when we generate the first app.

```
server2.component.css
server2.component.html
server2.component.spec.ts
server2.component.ts
```

server2.component.spec.ts is usually used for testing purpose. We can delete it by clicking right on it.



```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="data:image/svg+xml;base64
</div>
<h2>Here are some links to help you start: </h2>
<ul>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/t
a target="_blank" rel="noopener" href="https://angular.io/c
a target="_blank" rel="noopener" href="https://blog.angular
```