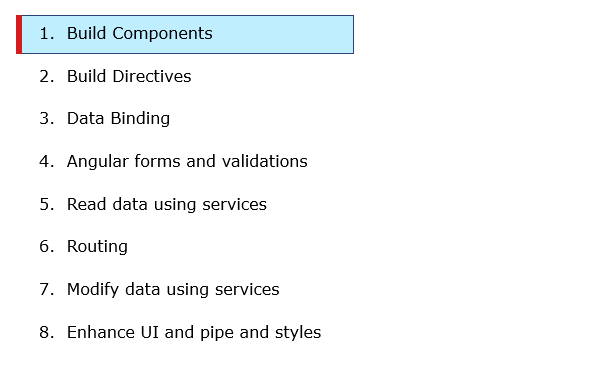
Components

You will learn about components in Angular in this section.



Component is a logical piece of code which is the basic building block for Angular application.

It consists of the following

* Template -  view for the application created in HTML
* Class - code with properties and methods which is used to support the view. It is defined in TypeScript.
* Metadata - extra data defined for the Angular class​​​​​​​​​​​​​​​​​​​

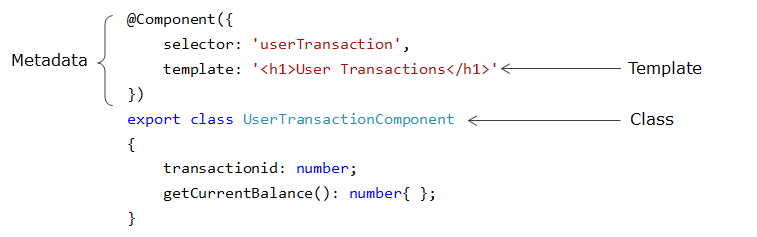
**Templates:**

Template is a part of component which is used as user interface by which the end-user can interact easily. We can create a template in two ways as follows:

1. Inline Template
2. External Template

Inline Template:

      When a template must be defined for a component in a .ts file inline templates are used. Inline templates are defined in the component decorator using the template property.



 External Template:

       The External templates define the HTML code in a separate file and that file is referred using **templateURL** property of Component decorator. The value for the **templateURL** will be the path of typeScript file that contains the HTML code.



You will learn to create an angular component for QuickKart application in the next demo.

# Creating Components - Demo

Highlights:

* Create a component using Angular CLI
* Observe the files created

Demosteps:

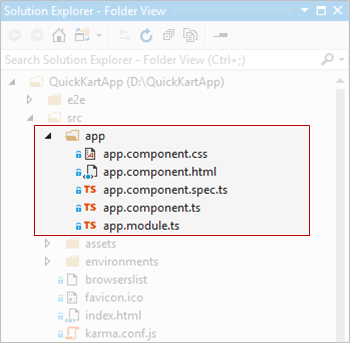
In order to create and learn how to use component in QuickKart application, try out the following demo steps:

**Step 1:**

Open QuickKartApp project in Visual studio.

**Step 2:**

Open QuickKartApp -> src -> app folder and observe the files for AppComponent



**Step 3:**

Open app.component.ts file and observe the code

1. import { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export class AppComponent {
8. title = 'QuickKartApp';
9. }

**export:**it is added to every class so that it becomes accessible in other files

**title:**Creates a property with name title and initializes it to value 'QuickKartApp'.

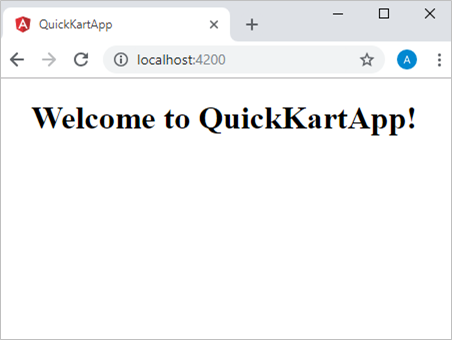
**Step 4:**

Open app.component.html and replace the code in it with the following lines of code

1. <div style="text-align:center">
2. <h1>
3. Welcome to {{title}}!
4. </h1>
5. </div>

Here we are accessing the class property title by placing it in **{{}}**. This is known as interpolation which is one of the data binding techniques which you will be learning later.

Now run the application in the browser



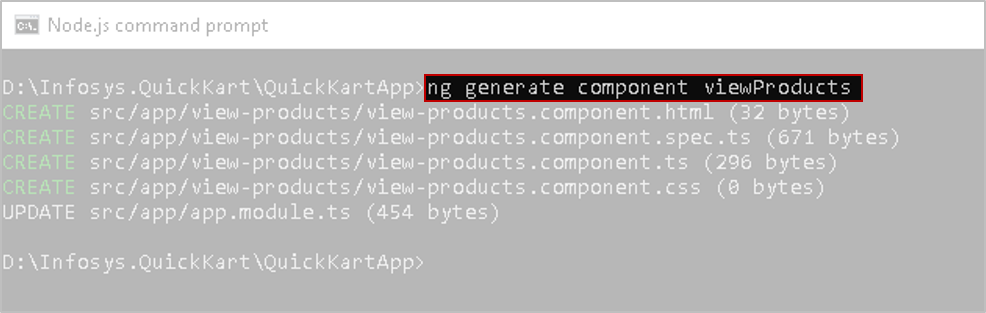
**Step 5:**

To create a new component using CLI in QuickKartApp - > app, execute the following command in command prompt.

1. ng generate component viewProducts

an alternate syntax to create the component.

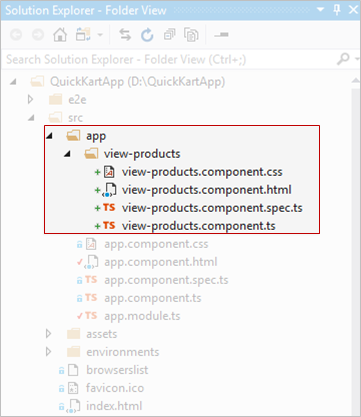
1. ng g c viewProducts



A component is a class with some metadata attached to it.

**Step 6:**

When the component is created, open the QuickKartApp folder in Visual Studio and observe the src->app folder



A component is created with the name view-products in app folder.

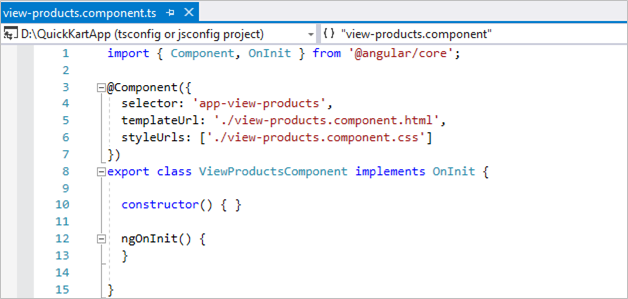
**Step 7:**

Open view-products.component.html file and replace the code with the following lines of code

1. <h2>
2. view-products works!
3. </h2>

Open view-products.component.ts file and observe the code

1. import { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-view-products',
4. templateUrl: './view-products.component.html',
5. styleUrls: ['./view-products.component.css']
6. })
7. export class ViewProductsComponent implements OnInit {
8. constructor() { }
9. ngOnInit() {
10. }
11. }



**@Component** : Although the name of the class is Component, it is a normal class. It will be considered as a component only when it is attached with a decorator @Component.

**Decorators** are functions that are identified with a prefixed '@' symbol, and are immediately followed by a class, parameter, method or property. The decorator is supplied information about the class, method or property. Whenever the class is used, it will get this metadata to configure the expected behavior.

**import { Component }**: To import **Component**class from **core**module present in **node\_modules** folder.

Observe the **templateUrl**property in line no 5 in the above image. In place of the path to the view-products.component.html file, you can directly write the html code using the property **template**and the html content should be given as a string enclosed in single quotation marks('). To specify html in multiple lines you have to use back ticks (`).

**export class  :** The view-products.component.ts should be exported so that it can be used in another TypeScript files.

**Note**: The naming convention followed for component class is <functionalityname> suffixed with Component.

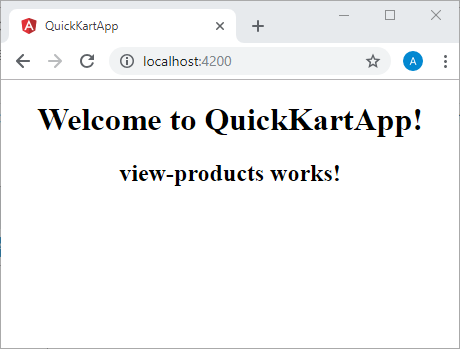
The selector of the component would be app-<componentname>. You can customise it based on the need.

**Step 8:**

Open app.component.html file and add the following code from line 5 - 7.

1. <div style="text-align:center">
2. <h1>
3. Welcome to {{title}}!
4. </h1>
5. <p>
6. <app-view-products></app-view-products>
7. </p>
8. </div>

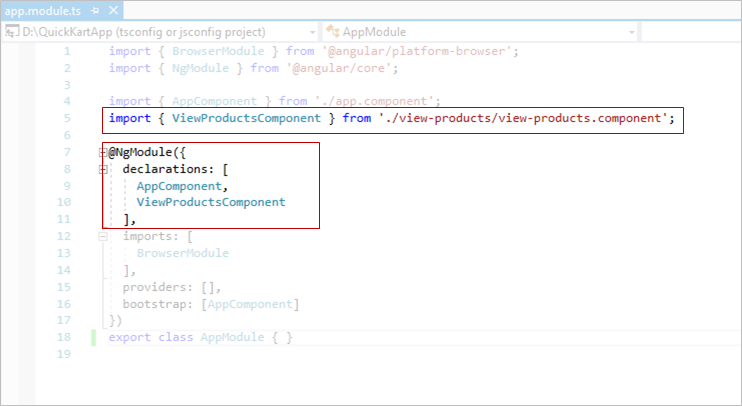
Here you have added the selector of ViewProducts component in the app.component.html file so that the template of ViewProducts can be rendered in the view.



**Step 9:**

Open app.module file and observe the code.

The ViewProductsComponent you created, is automatically added to the import statement and declarations array in app.module.ts



Any component class that you create will automatically be added in the declarations array of @NgModule decorator of the AppModule in order to make it visible to all the components.

**Note**:  You can now refer the selector of ViewProductsComponent in the AppComponent and directly in other components of the application. As you keep building more components in the application, the import statements will automatically be loaded into the AppModule.