1. **What are Directives:**

* They are instructions in DOM
* Components are such instructions in DOM
* Once we place the selector of our component somewhere in our templates – Then we are instructing angular to add the content of our component template and the business logic in our typescript code, in the particular place where we use the selector in our html template.
* Components are directives. But directives with a template
* There are directives without a template.

Eg:

*<p appTurnGreen> Receives a green background!</p>*

@Directive({

Selector:’[appTurnGreen]’

})

Export class TurnGreenDirective{

……

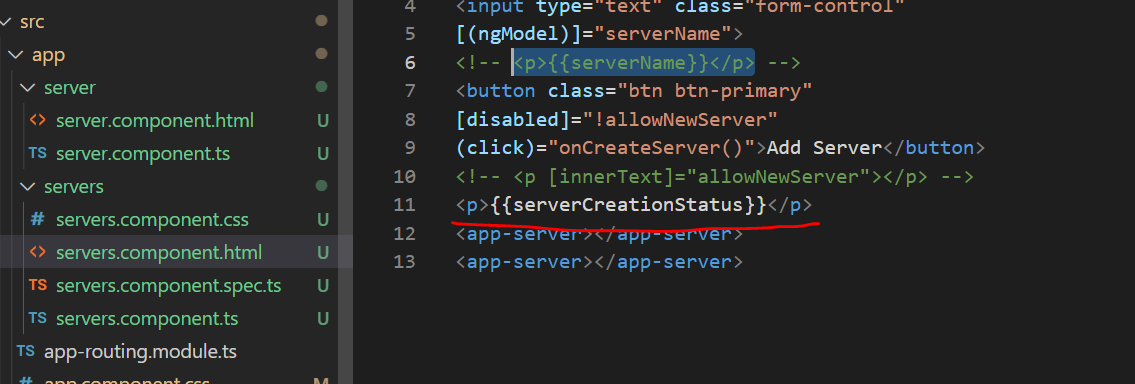
}

* Here appTurnGreen is a custom directive built by us
* Typically we add directives with attribute selector , but technically the selector of a directive can be configured just like the selector of a component.
* So we can use CSS classes or the element style, but typically use the attribute style.
* In the above example, directive simply colours the text green.
* In this course, before looking on custom directives – lets have a look at the built in directives

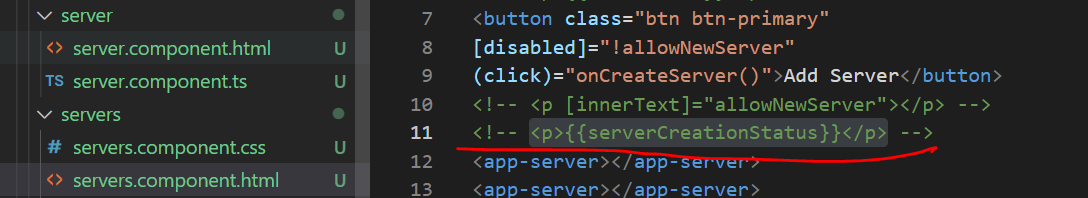
**Built in Directives**

1. **Using nglf to Output Data Conditionally:**

* So far other than component, we have not used any built in directives
* To conditionally show a message: ngIf directive can be used.
* It works like a if statement and to be precise: in our servers component(servers.component.html) , where we output the server creation status :



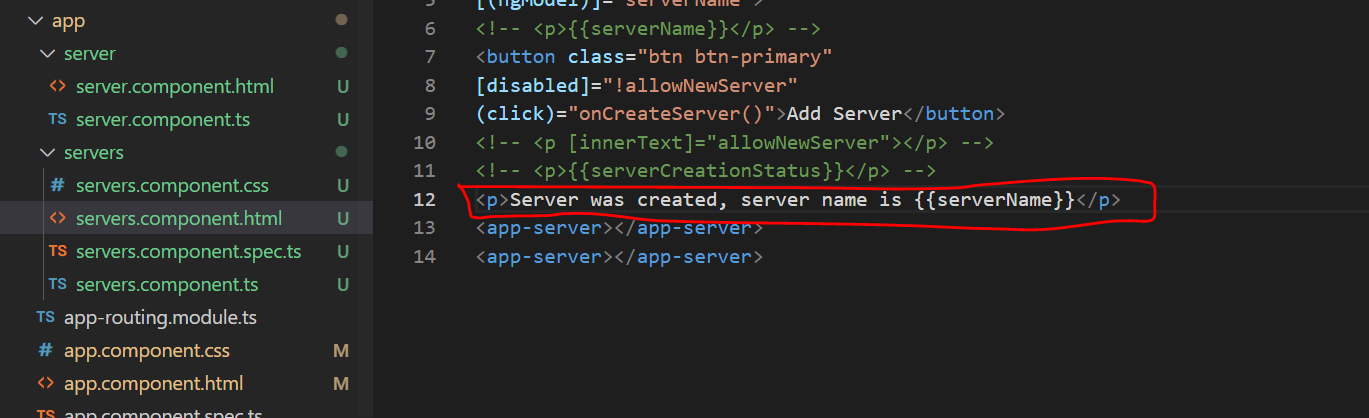
* We can comment this out, so that we can still reference this code .



Graphical user interface, text, application, email

Description automatically generated

* We will add a new paragraph, where we simply say : Server was created and server name is bind by serverName.



Graphical user interface, text, application, email

Description automatically generated

* IT will work but every key stroke the server name will get changed, which is not we needed. We want it to be outputted, only after we click add server button

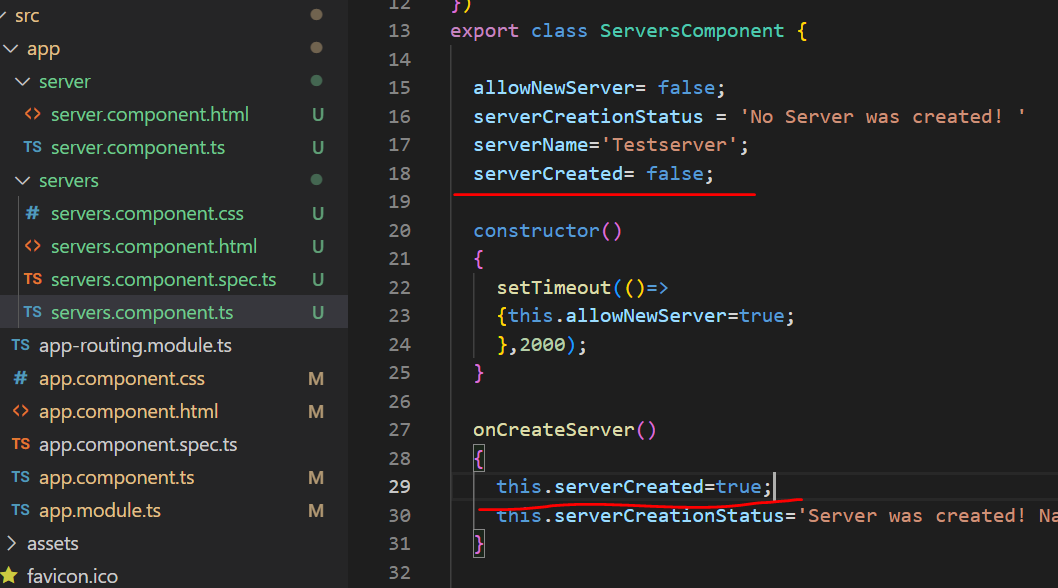
Graphical user interface, text, application, email

Description automatically generated

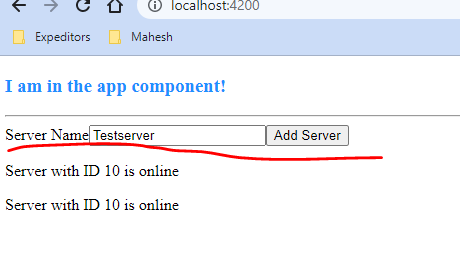
* To solve this issue, we can add a directive here. We know that directives are added by using a attribute selector. Almost all the inbuilt directives use that selector. In our case ngIf also does.

**ngIf**

* ngIf is added by adding a star
* Star is required because ngIf is a structural directive. Meaning it changes the structure of your dom. It either adds the element or it does not add it.
* So that’s an extra information for angular to indicate that it is structural directive
* For ngIf we can set our condition between the quotation marks
* For ngIf it has to be any expression returning true or false. Deciding whether this should be added or not
* <p \*ngIf="">Server was created, server name is {{serverName}}</p>
* In our typescript file, it would make sense to add a new property :serverCreated and we will set it to false. We will set it to true, once the button is clicked.
* In onCreateServer, which is triggered when the button is clicked – in that function we set serverCreated=true.



* With this change, we can go back to our template and simply bind ngIf to serverCreated. This could also call a method which could return true or false. Or we can directly perform the check between the quotation marks.
* <!-- <p [innerText]="allowNewServer"></p> -->
* <!-- <p>{{serverCreationStatus}}</p> -->
* <p \*ngIf="serverCreated">Server was created, server name is {{serverName}}</p>
* <app-server></app-server>
* Now if we check the same in browser,



* On Clicking add server we can see text gets printed on clicking Add Server button

Graphical user interface, text, application, email

Description automatically generated

* Another interesting thing is , if we reload the app and inpsect the dom here , we can see that on clicking that button, a new element paragraph gets added here

Text

Description automatically generated

* Important thing to note is, here the element gets added based on the condition. If condition is not satisfied element does not get added.

1. **Enhancing ngIf with an Else Condition:**

* In last section, we had a look with basic ngIf syntax which we will be using most of the time
* There is an alternative for the same also.
* Sometimes we have not only the if condition but also the else condition
* In our scenario, we may want to say “Server was created server name is something with ngIf”
* But we can also add a else block where we want to say “No server was created”
* Earlier in another section we deleted it, but imagine we want to show it now.
* <p \*ngIf="serverCreated">Server was created, server name is {{serverName}}</p>
* <p \*ngIf="serverCreated">No server was created </p>
* We want to show this paragraph, only if ngIf is not true
* This can be achieved by placing a local reference on this element here.

**What is local reference:**

* Assume it to be a marker. Local reference can be mentioned by using # symbol
* # symbol followed by any name you like.
* Also we need to change the paragraph to <ng-template>
* That is a component – a directive shipping with angular. – which you can use to mark places in the dom
* Then I will add the paragraph back in this template with the text:”No server was created!” in between
* Now with ng-template, with local references marker on it, we mark a certain spot in the template which we want to show conditionally.
* To show the text conditionally, we simply enhance ngIf , by adding else and then no server.
* So this marker(ngServer) we placed on ng-template
* <p \*ngIf="serverCreated;else noServer">Server was created, server name is {{serverName}}</p>
* <ng-template #noServer>
* <p>No Server was created!</p>
* </ng-template>
* <app-server></app-server>
* <app-server></app-server>
* Now if we check in the browser, by default “No Server was created” will be displayed.

Graphical user interface, text, application, email

Description automatically generated

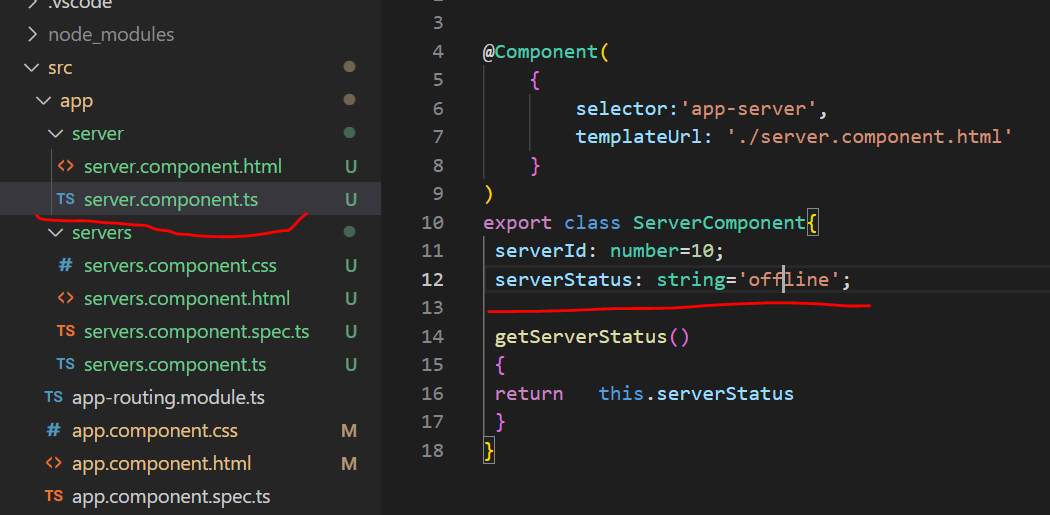
* Now on clicking Add Server, “Server was created, server name is TestServer”

Graphical user interface, text, application, email

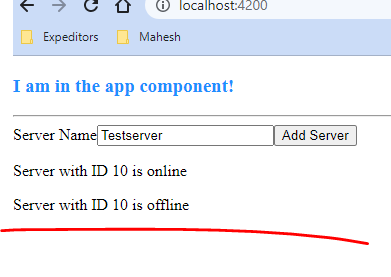
Description automatically generated

1. **Styling elements dynamically with ngStyle**

* While ngIf is structural directive. Other type of directive is Attribute directive
* Unlike structural directives, attribute directives don’t add or remove elements. They only change the element, they are placed on
* Attribute directives look like normal HTML attributes without any star



* Lets add one thing: On the server component, we output the server status. Server status normally is offline, lets mix this up
* Assume we want to dynamically create this . To do that add the constructor which is called, once the component is created. In constructor I will set the server status to a random value.
* export class ServerComponent{
* serverId: number=10;
* serverStatus: string='offline';
* constructor(){
* this.serverStatus=Math.random()>.5?'online':'offline';
* }
* getServerStatus()
* {
* return   this.serverStatus
* }
* Now based on this UI will randomly display online and offline status



**Change background colour of this component based on Service status:**

**ngStye directive:**

* For this we can use another directive: ngStyle
* For ngStyle we need to give some configuration to do something
* That is why we will use property binding on this directive
* It is Important to understand that the square brackets mentioned is not part of this directive name. Here the directive name is only ngStyle
* Here the square brackets indicate that we want to bind to some property on this directive. Here the property name also happens to be ngStyle.
* We will see this in practice in our own directives, in the directives section which comes later.
* But it is super important to understand that the property binding is not the same as a directive . It is totally different.
* We are binding to a property of the directive
* The ngStyle property expects to get a Javascript object. Here we define key value pairs.

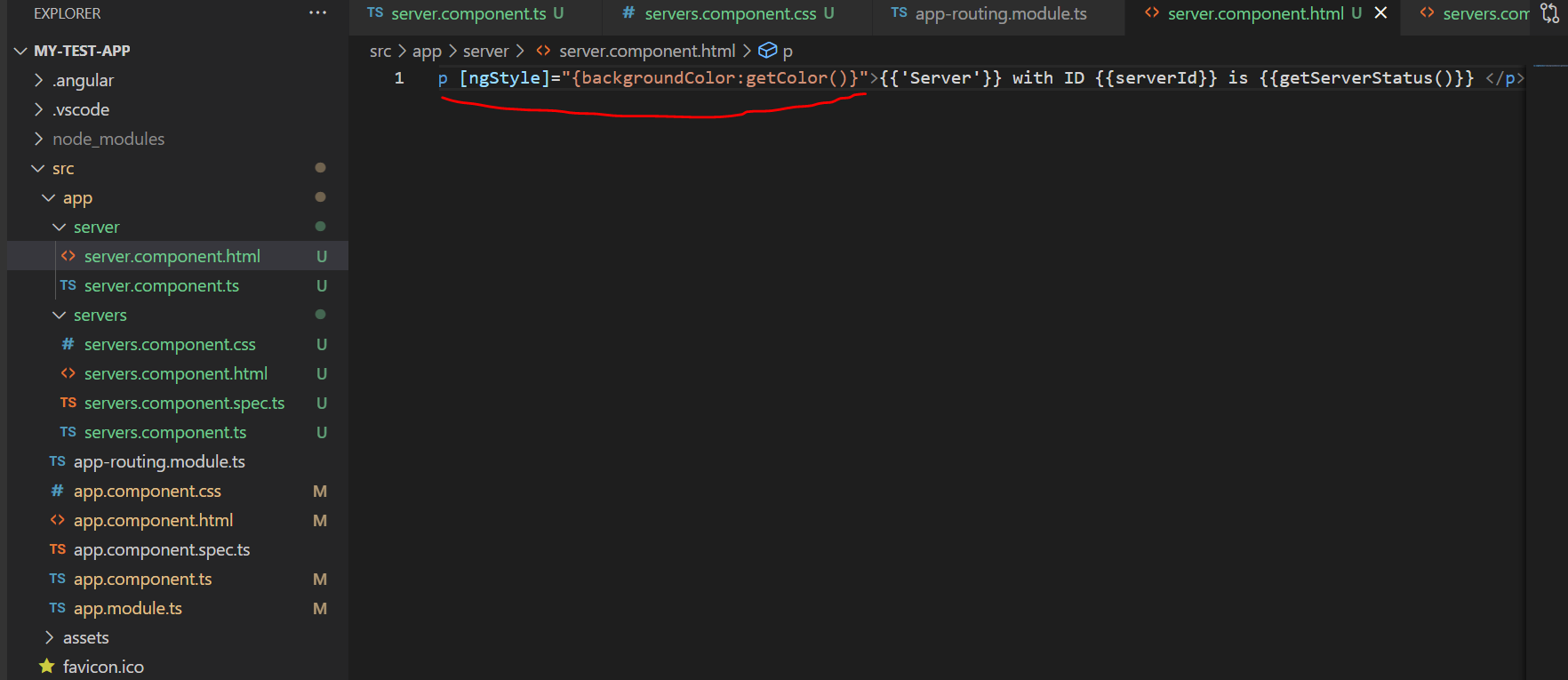
The style name is the key and the value of the style as the “value”

* For eg: we could bind background colour , depending on the serverStatus
* Also getColor is the method we add to server.component.ts . Here if colour is green, if status is online and red otherwise

Text

Description automatically generated

* This method will be used to assign background colour in server.component.html file



* Now if we check the same in browser: we can see the output color based on the condition

Graphical user interface, text, application, email

Description automatically generated

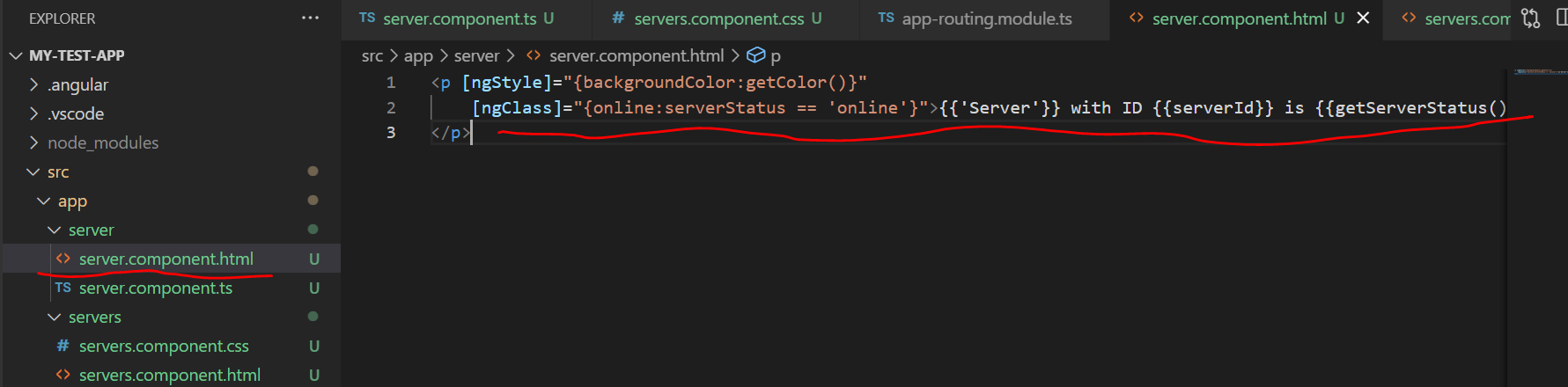
* Thus ngStyle helps to dynamically update the styles

1. **Applying CSS classes dynamically with ngClass:**

* While ngStyle helped to dynamically change the CSS styles, ***ngClass allows us to dynamically add or remove CSS classes***
* So lets create such a class and for this, lets first of all add styles here: server.component.ts
* Lets color the text white.



* Lets make our changes In server.component.html
* Ngclass directive works as intended only when using property binding.
* So let's wrap it in square brackets and we need to pass a JavaScript object.
* Each property you bind may take a different value like disabled, which took true or false.
* So here it's a JavaScript object.
* For ngClass, object has key value pairs.
* Key is: CSS class name and the values are the conditions determining whether this class should be attached or not.



* Only if the status is online,we should attach the CSS class online to this paragraph. Otherwise, it should not get attached.
* Let's view this in the running application in browser. Both are offline, so none of the two should have the online CSS class added to it.

A picture containing graphical user interface

Description automatically generated

* Now when both or one of the server is online, we can find the class online added:

A picture containing graphical user interface

Description automatically generated

1. **Outputting Lists with ngFor:**

* NgFor is a built-in template directive that makes it easy to iterate over something like an array or an object and create a template for each item.
* we can click the add server button. but we don't actually add servers to our list here, right? This list doesn't grow. This list is totally static. We can change this with the ngFor directive.
* In our servers component here(servers.component.html), we manually add our app-server component twice.

A screenshot of a computer

Description automatically generated with medium confidence

* It would be nicer to have an array of servers, which adds it dynamically.
* So in the servers component here(servers.component.ts) , I'll add a new property servers, and this is an array.
* Also in onCreateServer() function, I will push the serverName entered into the servers array.

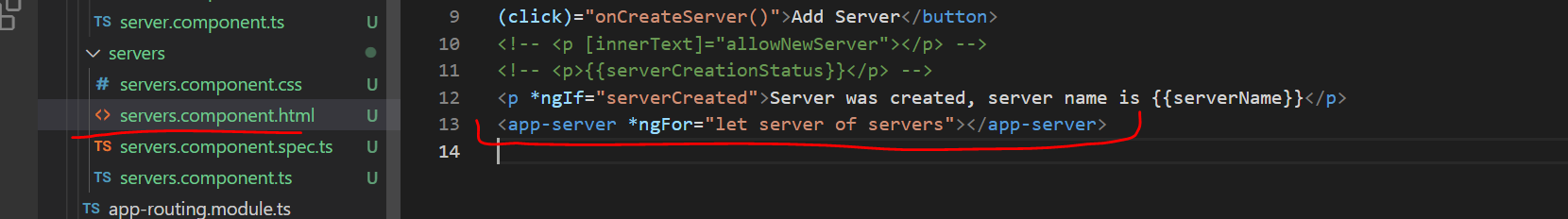
Text

Description automatically generated

* It would now be great if we could replicate the app-server component as often as needed to have a server for each element in the array. So we will have two initially, and then a new one for each server we add.
* We will do below changes to achieve the same in servers.component.html
* We will add ngFor Direction with star, since it is structural directive, which could change the dom

**ngFor Base Syntax:**

* We define a temporary variable inside a loop with ***let*** **server *of* Servers**
* Here server is the temporary variable for each element in the servers array
* here the ngFor defined here loops through all the elements in servers array and assign the individual elements to server variable .
* This server variable can now be used in the template



* but here we don't really need it to be honest. We will soon learn in the next core section how
* For now, lets save and see what gets outputted in the application

Graphical user interface, application

Description automatically generated

1. **Getting Index when using ngFor:**

* In Last assignment solution, we did it via incrementing number approach
* Another option is to use any other content. Eg Timestamp or any other text .
* In below file we can see i is assigned with index. Index is reserved keywor which prints current value of index

Text

Description automatically generated

* In the app.component.ts file, make the following changes. Here we will be pushing the date in the log array. Date is a built in Java script object.

Text

Description automatically generated

* When we check the same in application we can see the timestamp displayed as part of the array

Text

Description automatically generated