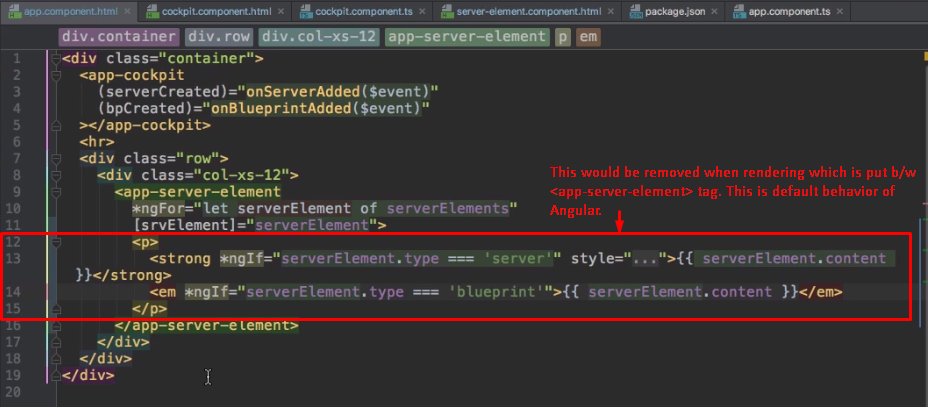
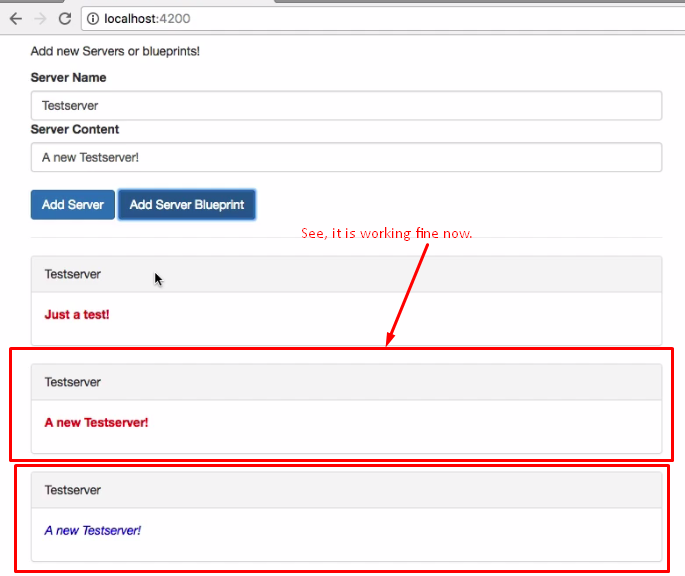
1. We learned a lot till now such as
   1. How to pass data around 🡺 @Input, @Output
   2. How to get access to elements in the DOMs 🡺 #templateRefVariable, **@ViewChild(“templateRefVariable”)**
   3. How to use local references
      1. {{templateRefVariable.value}}
      2. (click)=**”**handler(templateRefVariable)**”**
      3. **@ViewChild(“templateRefVariable”) templateReferenceVariableName: ElementRef  
         method(){**

Console.log(**templateReferenceVariableName.nativeElement.value**)**;**

**}**

1. There is **one more but last** way to pass data around.
2. See, whatever you put b/w <opening> and </closing> for a component, everything is lost by default when rendering.  
   <server-component>SomeHTML</server-component>  
   Here SomeHTML would be removed by default by the Angular when rendering and only the Server.component.html (template) would be placed in place of   
   <server-component> tag. 
3. But, you can change it. There is a special directive even though it looks like component 🡺 <ng-content>
   1. This serves a hook in you template to mark the place for Angular where the content (ignored above) must be added.
4. 
5. Whatever, you place anything b/w <**customComponent>**SomeTemplateToPlaceToThis\_customComponentTemplate**</customComponent>** can be accessed via   
   **<ng-content></ngcontent**> directive in customComponent’s template.
6. **Real Example**:
   1. **Suppose**, you want to build reusable widget such as **Tab Widget**. Now, suppose, someone else is using this tab widget and that someone wants to pass the content of tab which is HTML. See, the one using the tab widget is parent component and tab widget is child widget.   
      So, you can say that to pass HTML from parent to child component, we can use property binding.   
      Actually, property binding is used to pass basic value not complex HTML but here parent wants to pass tab’s content (html) to child template.
   2. The same you can do without **<ng-content>** directive but with property binding but is not recommended to prevent **cross-site scripting** even though you can do something so prevent **cross-site-scripting**.