## All in one

App.module.ts

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    FormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

App.component.html

<div class="container">

  <div class="row">

    <div class="col">

      <p>Add new Servers or blueprints!</p>

      <label>Server Name</label>

      <input type="text" class="form-control" [(ngModel)]="newServerName">

      <label>Server Content</label>

      <input type="text" class="form-control" [(ngModel)]="newServerContent">

    </div>

  </div>

  <br>

  <div class="row">

    <div class="col">

      <button class="btn btn-primary" (click)="onAddServer()">Add Server</button>

    </div>

    <div class="col">

      <button class="btn btn-primary" (click)="onAddBlueprint()">Add Server Blueprint</button>

    </div>

  </div>

  <hr>

  <div class="row">

    <div class="col">

      <div class="card" \*ngFor="let element of serverElements">

        <div class="card-body">

          <div>

            <h5 class="card-title">{{ element.name }}</h5>

          </div>

          <p class="card-text">

            <strong \*ngIf="element.type === 'server'" style="color: red">{{ element.content }}</strong>

            <em \*ngIf="element.type === 'blueprint'">{{ element.content }}</em>

          </p>

        </div>

      </div>

    </div>

  </div>

</div>

App.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  serverElements = [];

  newServerName = '';

  newServerContent = '';

  onAddServer() {

    this.serverElements.push({

      type: 'server',

      name: this.newServerName,

      content: this.newServerContent

    });

  }

  onAddBlueprint() {

    this.serverElements.push({

      type: 'blueprint',

      name: this.newServerName,

      content: this.newServerContent

    });

  }

}

## Splitting App into components

Create two components cockpit and server-element and refactor the code

## Binding to custom properties

Cockpit.component.html

<p>Add new Servers or blueprints!</p>

<label>Server Name</label>

<input type="text" class="form-control" [(ngModel)]="newServerName">

<label>Server Content</label>

<input type="text" class="form-control" [(ngModel)]="newServerContent">

<br>

<div class="row">

    <div class="col">

        <button class="btn btn-primary" (click)="onAddServer()">Add Server</button>

    </div>

    <div class="col">

        <button class="btn btn-primary" (click)="onAddBlueprint()">Add Server Blueprint</button>

    </div>

</div>

Cockpit.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-cockpit',

  templateUrl: './cockpit.component.html',

  styleUrls: ['./cockpit.component.css']

})

export class CockpitComponent implements OnInit {

  newServerName = '';

  newServerContent = '';

  constructor() { }

  ngOnInit(): void {

  }

  onAddServer() {

    // this.serverElements.push({

    //   type: 'server',

    //   name: this.newServerName,

    //   content: this.newServerContent

    // });

  }

  onAddBlueprint() {

    // this.serverElements.push({

    //   type: 'blueprint',

    //   name: this.newServerName,

    //   content: this.newServerContent

    // });

  }

}

Server.component.html

<div class="row">

    <div class="col">

      <div class="card">

        <div class="card-body">

          <div ><h5 class="card-title">{{ element.name }}</h5> </div>

          <p class="card-text">

            <strong \*ngIf="element.type === 'server'" style="color: red">{{ element.content }}</strong>

            <em \*ngIf="element.type === 'blueprint'">{{ element.content }}</em>

          </p>

        </div>

      </div>

    </div>

  </div>

Server.component.ts

import { Component, OnInit, Input } from '@angular/core';

@Component({

  selector: 'app-server-element',

  templateUrl: './server-element.component.html',

  styleUrls: ['./server-element.component.css']

})

export class ServerElementComponent implements OnInit {

  @Input() element: {type: string, name: string, content: string};

  constructor() { }

  ngOnInit(): void {

  }

}

App.component.html

<div class="container">

  <div class="row">

    <div class="col">

      <app-cockpit></app-cockpit>

      <app-server-element \*ngFor="let serverElement of serverElements" [element]=serverElement></app-server-element>

    </div>

App.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  serverElements = [{type:'server', name:'Test', content: 'For testing'}];

}

1. Assigning an alias to custom properties

App.component.html

<div class="container">

  <div class="row">

    <div class="col">

      <app-cockpit></app-cockpit>

      <app-server-element \*ngFor="let serverElement of serverElements" [srvElement]=serverElement></app-server-element>

    </div>

Server-element.component.ts

import { Component, OnInit, Input } from '@angular/core';

@Component({

  selector: 'app-server-element',

  templateUrl: './server-element.component.html',

  styleUrls: ['./server-element.component.css']

})

export class ServerElementComponent implements OnInit {

  @Input('srvElement') element: { type: string, name: string, content: string };

  constructor() { }

  ngOnInit(): void {

  }

}

1. Binding to custom events

Cockpit.component.ts

import { Component, OnInit, Output } from '@angular/core';

import { EventEmitter } from '@angular/core';

@Component({

  selector: 'app-cockpit',

  templateUrl: './cockpit.component.html',

  styleUrls: ['./cockpit.component.css']

})

export class CockpitComponent implements OnInit {

  @Output() serverCreated = new EventEmitter<{ serverName: string, serverContent: string }>();

  @Output() blueprintCreated = new EventEmitter<{ serverName: string, serverContent: string }>();

  newServerName = '';

  newServerContent = '';

  constructor() { }

  ngOnInit(): void {

  }

  onAddServer() {

    this.serverCreated.emit({serverName: this.newServerName, serverContent: this.newServerContent});

  }

  onAddBlueprint() {

    this.blueprintCreated.emit({serverName: this.newServerName, serverContent: this.newServerContent});

  }

}

App.component.html

<div class="container">

  <div class="row">

    <div class="col">

      <app-cockpit (serverCreated)="onServerAdded($event)" (blueprintCreated)="onBlueprintAdded($event)"></app-cockpit>

      <hr>

      <app-server-element \*ngFor="let serverElement of serverElements" [srvElement]=serverElement>

      </app-server-element>

    </div>

App.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  serverElements = [{ type: 'server', name: 'Test', content: 'For testing' }];

  onServerAdded(serverData: { serverName: string, serverContent: string }) {

    this.serverElements.push({

      type: 'server',

      name: serverData.serverName,

      content: serverData.serverContent

    });

  }

  onBlueprintAdded(blueprintData: { serverName: string, serverContent: string }) {

    this.serverElements.push({

      type: 'blueprint',

      name: blueprintData.serverName,

      content: blueprintData.serverContent

    });

  }

}

## Assigning an alias to custom events

App.component.html

  <div class="container">

    <div class="row">

      <div class="col">

        <app-cockpit (serverCreated)="onServerAdded($event)" (bpCreated)="onBlueprintAdded($event)"></app-cockpit>

        <hr>

        <app-server-element \*ngFor="let serverElement of serverElements" [srvElement]=serverElement>

        </app-server-element>

      </div>

Cockpit.component.ts

import { Component, OnInit, Output } from '@angular/core';

import { EventEmitter } from '@angular/core';

@Component({

  selector: 'app-cockpit',

  templateUrl: './cockpit.component.html',

  styleUrls: ['./cockpit.component.css']

})

export class CockpitComponent implements OnInit {

  @Output() serverCreated = new EventEmitter<{ serverName: string, serverContent: string }>();

  @Output('bpCreated') blueprintCreated = new EventEmitter<{ serverName: string, serverContent: string }>();

  newServerName = '';

  newServerContent = '';

  constructor() { }

  ngOnInit(): void {

  }

  onAddServer() {

    this.serverCreated.emit({serverName: this.newServerName, serverContent: this.newServerContent});

  }

  onAddBlueprint() {

    this.blueprintCreated.emit({serverName: this.newServerName, serverContent: this.newServerContent});

  }

}

## View Encapsulation

Show the ng\_attributes

Server-element.component.ts

import { Component, OnInit, Input, ViewEncapsulation } from '@angular/core';

@Component({

  selector: 'app-server-element',

  templateUrl: './server-element.component.html',

  styleUrls: ['./server-element.component.css'],

  //encapsulation: ViewEncapsulation.Emulated

})

export class ServerElementComponent implements OnInit {

  @Input('srvElement') element: { type: string, name: string, content: string };

  constructor() { }

  ngOnInit(): void {

  }

}

Server-element.component.css

p {

    color:blue;

}

label

{

    color: red;

}

## Local reference in templates

Cockpit.component.html

<p class="display-4">Add new Servers or blueprints!</p>

<label>Server Name</label>

<!-- <input type="text" class="form-control" [(ngModel)]="newServerName"> -->

<input type="text" class="form-control" #serverNameInput>

<label>Server Content</label>

<input type="text" class="form-control" [(ngModel)]="newServerContent">

<br>

<div class="row">

    <div class="col">

        <button class="btn btn-primary" (click)="onAddServer(serverNameInput)">Add Server</button>

    </div>

    <div class="col">

        <button class="btn btn-primary" (click)="onAddBlueprint(serverNameInput)">Add Server Blueprint</button>

    </div>

</div>

Cockpit.component.ts

import { Component, OnInit, Output } from '@angular/core';

import { EventEmitter } from '@angular/core';

@Component({

  selector: 'app-cockpit',

  templateUrl: './cockpit.component.html',

  styleUrls: ['./cockpit.component.css']

})

export class CockpitComponent implements OnInit {

  @Output() serverCreated = new EventEmitter<{ serverName: string, serverContent: string }>();

  @Output('bpCreated') blueprintCreated = new EventEmitter<{ serverName: string, serverContent: string }>();

  newServerName = '';

  newServerContent = '';

  constructor() { }

  ngOnInit(): void {

  }

  onAddServer(nameInput: HTMLInputElement) {

    this.serverCreated.emit({serverName: nameInput.value, serverContent: this.newServerContent});

  }

  onAddBlueprint(nameInput: HTMLInputElement) {

    this.blueprintCreated.emit({serverName: nameInput.value, serverContent: this.newServerContent});

  }

}