# تعاریف

|  |  |  |  |
| --- | --- | --- | --- |
|  | -- | | |
| شماره سند: | شناسه سند | |  |
| طـبقه‌بندی: | عادی/حساس/محرمانه | |  |
| تــاریــخ: | روز/ماه/سال |  | |

v

SPA: اپلیکیشن تک صفحه ای Single Page Application

Directives: دستورالعمل‌ها (دایرکتیو) در انگولار یک کلاس است که به صورت @directive اعلان شده است.

# پیش نیازها

## محیط توسعه (IDE)

در این مستند از Visual Studio Code برای کد نویسی استفاده می کنیم.

## پلتفرم Node.JS

نودجی‌اس یک محیط اجرای جاوااسکریپتی است که روی موتور اجرای کد جاوااسکریپتِ مرورگر کروم تحت عنوان V8 ساخته شده است. برای نصب Packege ها از npm موجود در Node.JS استفاده می کنیم.

فایل نصبی آن را از سایت <https://nodejs.org/en> دانلود کنید و برای اطمینان از نصب صحیح Node.JS و npm از دو دستور زیر در cmd یا PowerShell ویندوز استفاده می‌کنیم.

|  |
| --- |
| node -v  npm -v |

## Angular CLI

Angular cli ابزاری است که به برنامه نویسان Angular کمک میکنند تا بتوانند بهتر و راحتتر با Angular کار کنند و تمرکز کامل خود را بر روی توسعه اپلیکشن خود قرار دهند و بخش های ایجاد فایل های مختلف، تست کدها، مشاهده خطاهای کاربری و ... رو به Cli واگذار کنند.

برای نصب Angular CLI از دستور زیر در cmd استفاده کنید.

|  |
| --- |
| npm install –g @angular/cli |

-g : این آپشن مشخص می کند که Angular CLI به صورت Global نصب شود. در صورتی که از پرچم -g استفاده نکنید فقط برای یک پروژه نصب می شود.

به علت مشکلات تحریم ممکن است برای نصب ابزار به پروکسی یا فیلترشکن نیاز داشته باشید.

# ایجاد پروژه جدید

برای ایجاد یک پروژه Angular جدید از کد زیر در cmd استفاده کنید. دقت کنید که فایل های پروژه در مسیری که اکنون قرار دارید ساخته خواهند شد برای اینکه در مسیر دلخواه شما ساخته شود از دستورات cmd برای رفتن به پوشه مورد نظر خود استفاده کنید.

|  |
| --- |
| ng new ProjectName |

پس از اجرای دستور فوق محتویات پروژه در فولدری با نام پروژه ای که وارد کرده اید ساخته می شود.

به علت مشکلات تحریم ممکن است به پروکسی یا فیلترشکن نیاز داشته باشید.

با زدن دستور زیر به پوشه پروژه وارد شوید.

|  |
| --- |
| cd ProjectName |

برای اجرای برنامه از دستور زیر استفاده کنید.

|  |
| --- |
| ng serve |

پس از اتمام کامپایل یک لینک به شما نمایش داده می شود که پورت اپلیکیشن در آن مشخص شده است. با وارد کردن لینک نمایش داده شده در مرورگر خود می توانید اپلیکیشن ساخته شده را مشاهده کنید.

# آشنایی با TypeScript

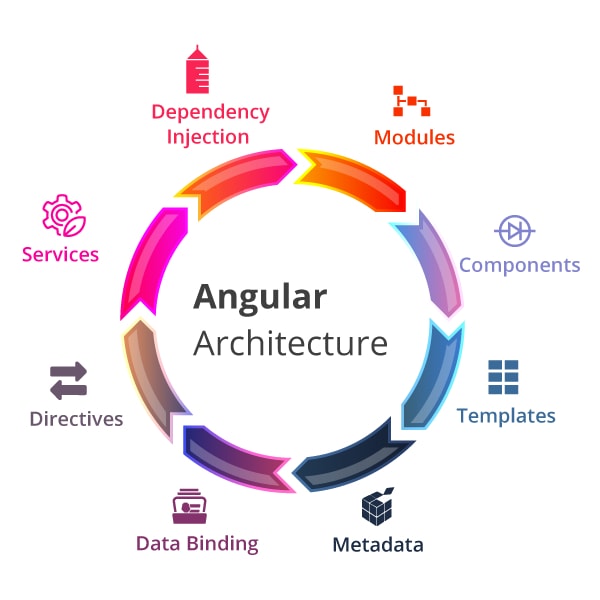
متن

# آشنایی با Webpack

متن

# ساختار پروژه انگولار

معماری انگولار دارای قابلیت‌های بسیاری است که به توسعه‌دهندگان برای ساخت برنامه کمک می‌کند.



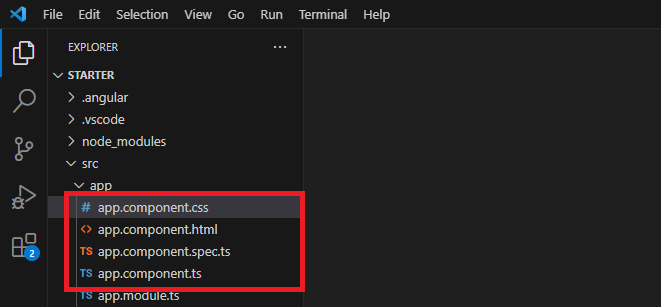
اساسا می‌توان گفت انگولار (Angular) یک چارچوب و همچنین بستری برای ساخت برنامه‌های مشتری (client) در HTML و TypeScript است. عملکرد انگولار توسط مجموعه از کتابخانه‌های TypeScript اداره می‌شود.

## اجزا (Components)

هر برنامه انگولار دارای حداقل یک کامپوننت است و به عنوان مؤلفه اصلی شناخته می‌شود که به همراه ایجاد یک پروژه جدید Angular ایجاد می شود که با عنوان کامپوننت اصلی یا ریشه نیز شناخته می شود.

کامپوننت از چهار فایل به شرح زیر تشکیل شده است:

* component.css: استایل های css مربوط به کامپوننت درون این فایل قرار می‌گیرد.
* component.html: کد های html مربوط به کامپوننت درون این فایل قرار می‌گیرد.
* component.spec.ts: این فایل مربوط به تست‌های برنامه نویسی است.
* component.ts: فایل اصلی کامپوننت است که شامل کلاس‌های تایپ اسکریپت است.



### ساخت کامپوننت با Angular CLI

برای ساخت یک کامپوننت در Angular می‌توانید از دستور زیر در Terminal ابزار Visual Studio Code یا هر IDE دیگری استفاده کنید.

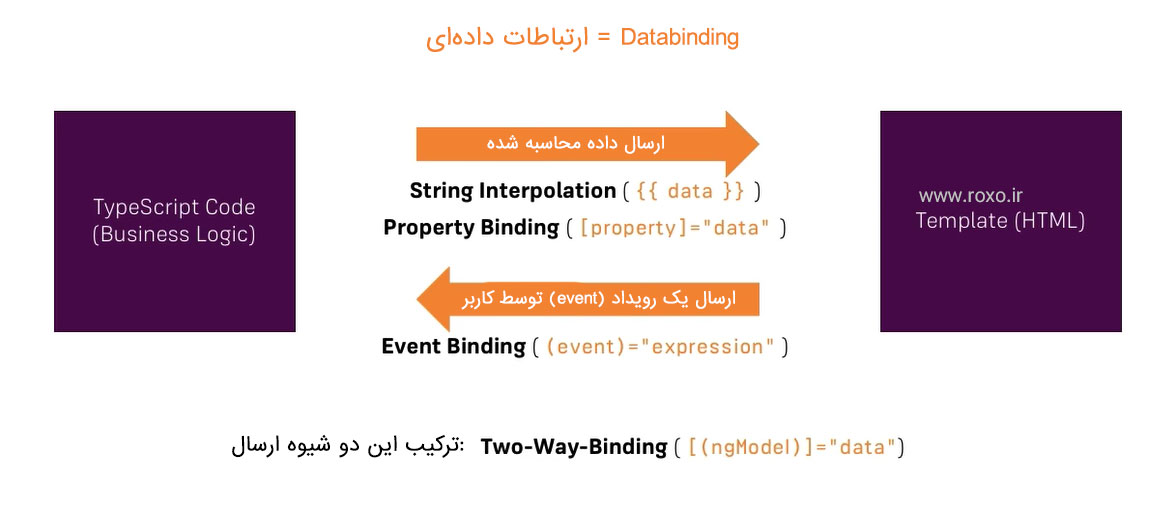
|  |
| --- |
| ng generate component about |

about: یک نام اختیاری برای کامپوننت است.

## Data Binding

در انگولار مفهومی تحت عنوان databing مطرح شد که به معنی ارتباطات داده‌ای بین قسمت منطق یا کنترلر برنامه که به زبان تایپ اسکریپت نوشته می‌شود با قسمت نمایشی برنامه که به زبان Html نوشته شده است.

به عبارت دیگر databinding به معنی ارتباط app.component.ts با app.component.html است. برای تفهیم بیشتر به تصویر زیر دقت کنید:



### نمایش داده با Interpolation

رایج‌ترین روش برای نمایش مقادیر اتصال‌یافته این است که آن‌ها را به صورت فرزندان عنصر DOM دیگر نمایش دهیم. ما می‌توانیم متغیر خود را با استفاده از نام آن از قالب‌ها و به وسیله دو آکولاد در طرفین مورد ارجاع قرار دهیم. به خاطر بسپارید که این روش تنها اتصال داده یک‌طرفه ارائه می‌کند:

|  |
| --- |
| <p>{{ first\_name }}</p> |

نکته: ابتدا باید یک متغییر در فایل component.ts با نام first\_name بسازید.

### Property binding

برخی عناصر HTML و کامپوننت‌های انگولار این الزام را دارند که مقادیر با استفاده از «خصوصیت‌ها» (attributes) به آن‌ها ارسال شود. این اتصال‌های داده به نام «اتصال‌های مشخصه» (property bindings) شناخته می‌شوند.

ساختار این روش نیز مشابه روش قبلی است و مانند این است که مقدار را به فرزندان عنصر الحاق می‌کنیم، تنها تفاوت در این روش آن است که از ساختاری به همراه خصوصیت‌ها استفاده می‌کنیم.

یکی از عناصری که مقدار را از طریق خصوصیت‌ها می‌گیرد، تگ input است:

|  |
| --- |
| <input id="first\_name" [value]="first\_name" /> |

### Event binding

تا به این جا تنها از انتقال داده برای نمایش مقادیر متغیرها از کنترلر به نما استفاده کرده‌ایم، اما اگر بخواهیم به اعمال انجام یافته از سوی کاربران واکنش نشان دهیم باید رویداد متناظر، برای نمونه رویداد کلیک را به یک متد اتصال دهیم که درون منطق کامپوننت اعلان شده است.

به این منظور از خصوصیت‌ها استفاده می‌کنیم و نام رویدادی که می‌خواهیم به آن اتصال پیدا کنیم را درون پرانتز می‌آوریم:

|  |
| --- |
| <button (click)="onButtonClicked()">Change First Name</button> |

در component.ts نیز متد خود را قرار می دهیم.

|  |
| --- |
| import { Component } from '@angular/core'  @Component({  selector: 'my-app',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css'],  })  export class AppComponent {  first\_name = 'Angular'    onButtonClicked() {  this.first\_name = 'Charlie'  }  } |

### Class and style binding

برای اتصال داده ها از جزء به ویژگی کلاس HTML استفاده می شود.

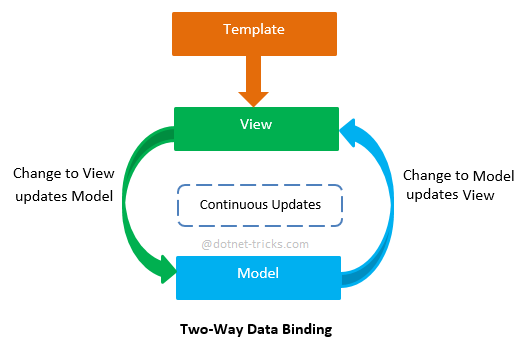
|  |
| --- |
| <p [class]="myClasses"> |

Style binding برای اتصال داده ها از جزء به ویژگی استایل HTML استفاده می شود.

|  |
| --- |
| <p [style.color]="myParaColor"> ... </p> |

### اتصال دو طرفه (Two-way binding)

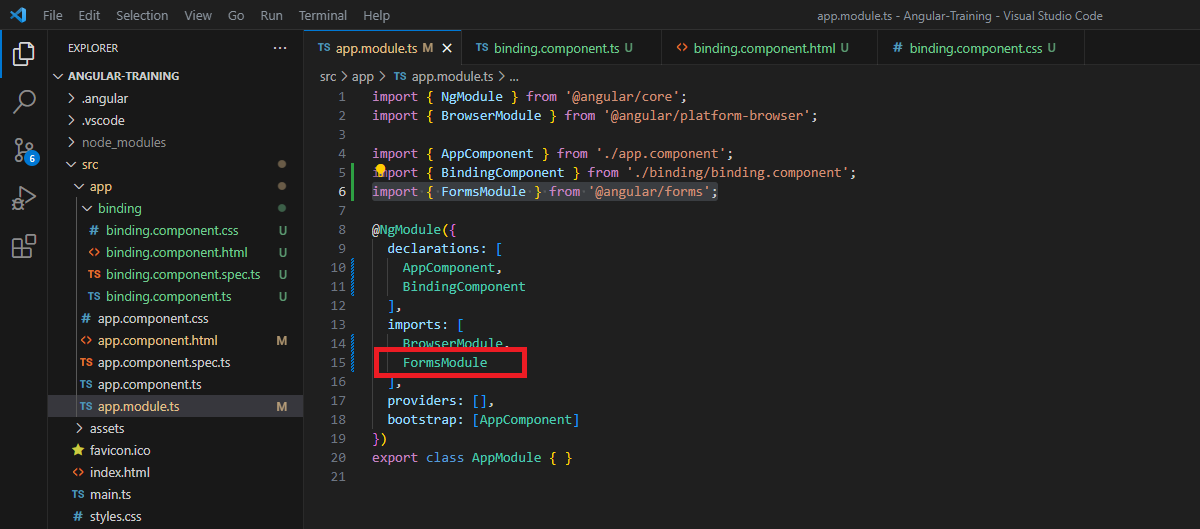
انگولار با استفاده از اتصال دوطرفه نه تنها تغییرهای متغیرهای ما را رصد می‌کند، بلکه اعمالی که از سوی کاربران انجام می‌گیرد را نیز ردگیری کرده و متغیرهای اتصال‌یافته را بر این اساس به‌روزرسانی می‌کند.



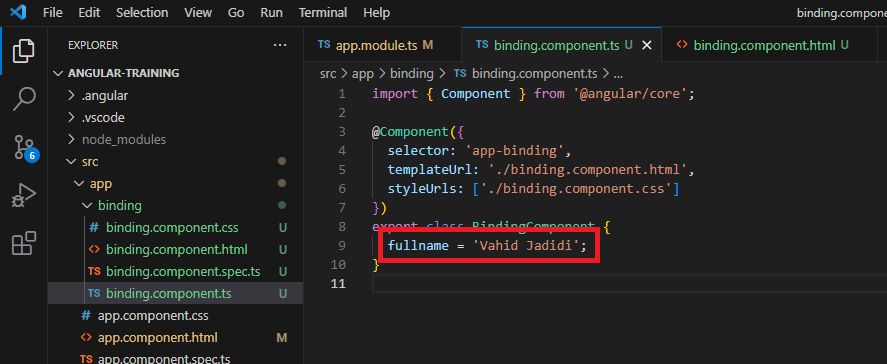
برای استفاده از این قابلیت کد زیر را در فایل app.module.ts اضافه می کنیم تا از کتابخانه‌های انگولار قابلیت FormsModule را به کامپوننت‌ها اضافه کنیم.

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

و پس از آن FormsModule را به imports ماژول اضافه می کنیم.



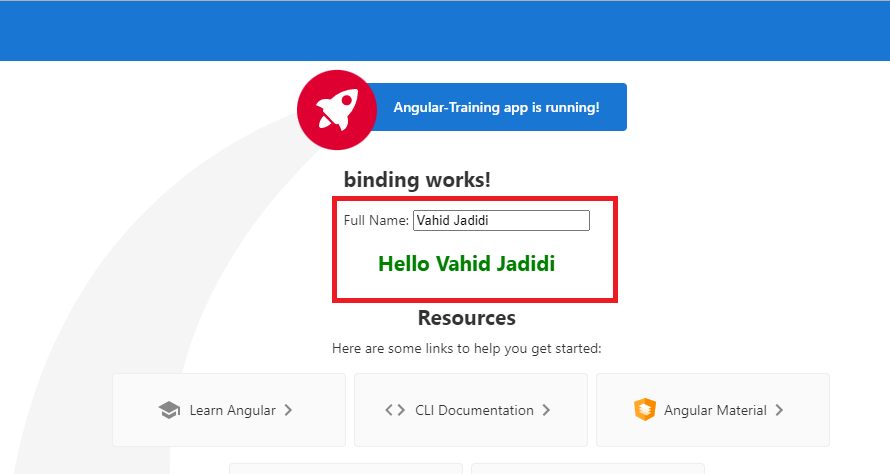
سپس یک متغیر در کامپوننت خود ایجاد می‌کنیم و آن را مقدار دهی می کنیم(نام کامپوننت من binding است).



و در فایل binding.component.html کد زیر را قرار می‌دهیم.

|  |
| --- |
| <p>Full Name: <input type="text" [(ngModel)]='fullname'></p>  <h2 class="welcome">Hello {{fullname}}</h2> |

اکنون دو تگ input و h2 بین قسمت منطق(logic) برنامه و قسمت نمایشی(UI) اتصال دو طرفه ایجاد کرده‌اند.



فراموش نشود که selector کامپوننت در فایل app.component.html قرار گیرد.

|  |
| --- |
| <app-binding></app-binding> |

## ng-content

از این قابلیت می توان برای ارسال المنت های HTML از کامپوننت ریشه(app.component.html) به کامپوننت های دیگر استفاده کرد.

به عنوان مثال کد زیر را در کامپوننت ریشه قرار دهید.

|  |
| --- |
| <app-binding>  <h1>Hello ...</h1>  <p>Welcome to my website</p>  </app-binding> |

تگ <app-binding> مربوط به سلکتور کامپوننت binding.component.html است که در کامپوننت ریشه app.component.html قرار می گیرد.

کد زیر را در کامپوننت binding.component.html قرار دهید.

|  |
| --- |
| <ng-content></ng-content> |

تگ <ng-content> را در هر جایی از کد HTML خود قرار دهید المنت های ارسال شده از کامپوننت ریشه در آنجا نمایش داده خواهند شد.

شما همچنین می توانید المنت های ارسال شده را Select کنید به عنوان مثال فقط تگ های p نمایش داده شوند و یا فقط المنت‌هایی که کلاس Active را دارند نمایش داده شوند. در زیر مثال هایی در این خصوص آورده شده است.

|  |
| --- |
| <ng-content select="p"></ng-content>  <ng-content select=".Active"></ng-content> |

## اشتراک گذاری داده بین کامپوننت‌های Parent و Child

یک الگوی رایج در Angular به اشتراک گذاری داده ها بین یک مؤلفه والد و یک یا چند مؤلفه فرزند است. این الگو را می‌توانید با دکوراتورهای @Input و @Output پیاده سازی کنید.

### ارسال داده به کامپوننت Child

با استفاده از دکوراتور @Input می‌توان مقدار یک property را از کامپوننت پدر(Parent) به کامپوننت فرزند(Child) ارسال کرد.

ابتدا یک property در کامپوننت app.component.ts ایجاد کنید و آن را مقدار دهی کنید.

|  |
| --- |
| Sent = 'Sent Text ...'; |

سلکتور مربوط به کامپوننت فرزند را در کامپوننت app.component.html قرار دهید.

|  |
| --- |
| <app-io [inputtext]="Sent"></app-io> |

با استفاده از [inputtext] مقدار property را برای کامپوننت فرزند ارسال کنید.

کد زیر را در کامپوننت فرزند با پسوند ts قرار دهید(io.component.ts).

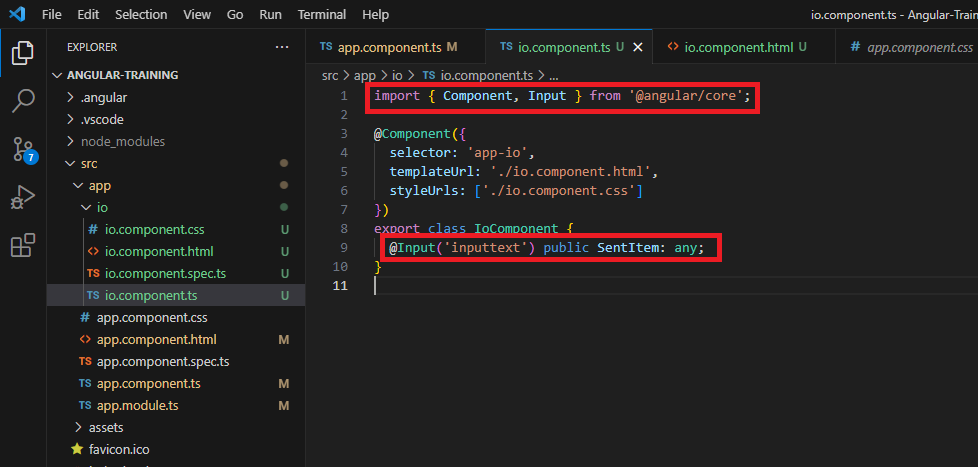
|  |
| --- |
| @Input('inputtext') public SentItem: any; |

مقدار inputtext همنام با مقدار وارد شده در سلکتور کامپوننت می باشد و SentItem یک نام اختیاری است.

و همچنین متد Input را به کامپوننت ایمپورت کنید.

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

در شکل زیر نیز این کد ها نمایش داده شده اند.



در انتها کد زیر را برای نمایش داده ارسال شده به HTML کامپوننت فرزند اضافه کنید.

|  |
| --- |
| <p>io works! {{SentItem}}</p> |

### دریافت داده از کامپوننت Child

22

## NgIf

شکل ساده استفاده از NgIf به شکل زیر است

|  |
| --- |
| <div \*ngIf="condition">Content to render when condition is true.</div> |

نحوه استفاده از آن به همراه else به شکل زیر است

|  |
| --- |
| <div \*ngIf="condition; else elseBlock">Content to render when condition is true.</div>  <ng-template #elseBlock>Content to render when condition is false.</ng-template> |

نحوه استفاده به همراه then و else

|  |
| --- |
| <div \*ngIf="condition; then thenBlock else elseBlock"></div>  <ng-template #thenBlock>Content to render when condition is true.</ng-template>  <ng-template #elseBlock>Content to render when condition is false.</ng-template> |

برای مشاهده نمونه های بیشتر از [سایت رسمی](https://angular.io/api/common/NgIf) آن دیدن کنید.

## NgSwitch

شکل ساده استفاده از NgSwitch به شکل زیر است

|  |
| --- |
| <container-element [ngSwitch]="switch\_expression">  <!-- the same view can be shown in more than one case -->  <some-element \*ngSwitchCase="match\_expression\_1">...</some-element>  <some-element \*ngSwitchCase="match\_expression\_2">...</some-element>  <some-other-element \*ngSwitchCase="match\_expression\_3">...</some-other-element>  <!--default case when there are no matches -->  <some-element \*ngSwitchDefault>...</some-element>  </container-element> |

## NgFor

شکل ساده استفاده از NgFor به شکل زیر است

|  |
| --- |
| <table>  <thead>  <th>id</th>  <th>Name</th>  </thead>  <tr \*ngFor="let item of persons">  <td>{{item.id}}</td>  <td>{{item.name}}</td>  </tr>  </table> |

# تفاوت Constructor و OnInit

سازنده یا Constructor یک متد پیش فرض می باشد که در تمام کلاس ها وجود دارد. این متد پیش فرض در تمام زبان های برنامه نویسی که قابلیت شیءگرایی دارند مورد استفاده قرار می گیرد و وظیفه آن مقدار دهی اولیه به ویژگی ها یا فیلدها است.

در انگولار هنگام تزریق وابستگی (Dependency Injection) یا DI معمولا سازنده ها به عنوان پارامترهای خود، کلاس های دیگر را درون خود قرار می دهند. این بدین معنی است که وقتی شما قصد ساخت یک نمونه از یک کلاس را دارید، می توانید به سایر کلاس هایی که به عنوان پارامتر به سازنده تزریق شده اند، دسترسی داشته باشید.

از طرفی دیگر ngOnInit به عنوان یک چرخه حیات از انگولار به حساب می آید که از نسخه ۲ به بعد انگولار معرفی شد و بیانگر این است که کلاس کامپوننت انگولار مقدار دهی اولیه شده است.

برای استفاده از این متد یا چرخه حیات باید حتما کلاس OnInit را از angular/core فراخوانی کنید:

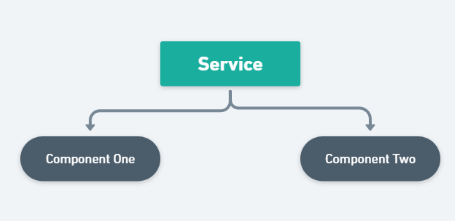
|  |
| --- |
| import {Component, OnInit} from '@angular/core'; |

از طرفی برای استفاده از متد ngOnInit آن را به صورت زیر تعریف می کنید:

|  |
| --- |
| export class App implements OnInit{  constructor(){  //قبل از ngOnInit اجرا می شود  }  ngOnInit(){  //بعد از سازنده و درست قبل از ngOnChanges()  }  } |

# سرویس ها

سرویس ها، کلاس هایی می باشند که مسؤلیت تبادل اطلاعات بین کامپوننت ها را برعهده دارند.



## مفهوم Injector

در انگولار مفهومی به نام Injector‌ به عنوان ابزاری جهت دسترسی به سرویس‌ها معرفی شد. در واقع با استفاده از Injector می‌توان یک وابستگی را به یک کامپوننت تزریق کرد. یعنی یک نمونه از یک کلاس را به صورت خودکار در کامپوننت موردنظر ایجاد کرد. برای دسترسی به این نمونه باید با مفهومی به نام provider آشنا شوید.

## مفهوم Provider

یک service provider به عنوان بستری برای تولید خودکار نمونه از یک کلاس معرفی می‌شود.

## ایجاد سرویس

به منظور ایجاد یک سرویس باید از خط فرمان دستور زیر را استفاده کنید.

|  |
| --- |
| ng g service (service name) |

G مخفف شده generate می باشد.

به عنوان مثال

|  |
| --- |
| ng g service myservice |

با این خط کد دو فایل برای ما ایجاد می شود

|  |
| --- |
| CREATE src/app/myservice.service.spec.ts  CREATE src/app/myservice.service.ts |

محتوای فایل myservice.service.ts

|  |
| --- |
| import { Injectable } from '@angular/core';  @Injectable({  providedIn: 'root'  })  export class MyserviceService {  constructor() { }  } |

در کد بالا ماژول Injectable از angular/core وارد شده و شامل یک متد با نام Injectable@ و یک کلاس با نام MyserviceService است. تابع مربوط به سرویس در این کلاس ایجاد خواهد شد.

قبل از استفاده از سرویس ایجاد شده باید آن را به فایل app.module.ts اضافه کنیم.

|  |
| --- |
| import { MyserviceService } from './myservice.service';  @NgModule({  declarations: [  AppComponent,  DetailsComponent  ],  imports: [  BrowserModule  ],  providers: [MyserviceService],  bootstrap: [AppComponent]  })  export class AppModule { } |

در کد فوق سرویس ایجاد شده را import کرده و سپس MyserviceService را به بخش providers اضافه کرده ایم.

حالا در myservice.service.ts، یک تابع برای نمایش ساعت و تاریخ روز جاری ایجاد می کنیم.

|  |
| --- |
| import { Injectable } from '@angular/core';  @Injectable({  providedIn: 'root'  })  export class MyserviceService {  constructor() { }  showTodayDate() {  let date = new Date();  return date  }  } |

در کد فوق یک تابع با نام showTodayDate که یک نمونه از کلاس Date باز می گرداند را ایجاد کرده ایم.

حال برای دسترسی به این تابع در سایر کامپوننت ها می توانید به صورت زیر عمل کنید.

|  |
| --- |
| import { MyserviceService } from './myservice.service';  import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css']  })  export class AppComponent implements OnInit {  title = 'Angular-Training';  constructor(private myservice: MyserviceService){    }  ngOnInit(): void {  this.today = this.myservice.showTodayDate();  }  today:any;  } |

برای دسترسی به اجزای سرویس مورد نظر باید آن را import کنید و در بخش constructor سرویسی که ساختیم را معرفی کنیم.

(تابع ngOnInit به صورت پیشفرض در زمان ایجاد شدن کامپوننت فراخوانی می شود و تاریخ را با استفاده از تابع showTodayDate می گیرد و در متغیر todayDate قرار می دهد.)

حال به راحتی می توانیم آن را در فایل html نمایش دهیم.

|  |
| --- |
| <h1>Today's Date {{today}}</h1> |

# Observable

Observable به عنوان یک شیء و منبع داده‌ی متنوع است. یعنی یک Observable شیء ای می‌باشد که انواع مختلف داده را درون خود ذخیره می‌کند و از پکیج Rxjs دریافت و به انگولار اضافه می‌شود.

در مثال زیر نحوه دریافت Data از یک API و نمایش آن در یک کامپوننت را بررسی می کنیم.

ابتدا ماژول HttpClientModule را در app.module.ts اضافه می کنیم.

|  |
| --- |
| import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { AppComponent } from './app.component';  import { DetailsComponent } from './details/details.component';  import { MyserviceService } from './myservice.service';  import {HttpClientModule} from '@angular/common/http'  @NgModule({  declarations: [  AppComponent,  DetailsComponent  ],  imports: [  BrowserModule, HttpClientModule  ],  providers: [MyserviceService,PersonInformationService],  bootstrap: [AppComponent]  })  export class AppModule { } |

یک Interface جدید در root پروژه با نام دلخواه (IPerson.ts) وارد می کنیم و نوع خروجی را (مطابق با API مورد نظر) در آن مشخص می کنیم.

|  |
| --- |
| export interface IPerson  {  id: number,  name: string,  email: string,  gender: string,  status: string  } |

در فایل مربوط به سرویس (به عنوان مثال myservice.service.ts) HttpClient را در constructor فراخوانی کرده و متد getData را مطابق کد زیر ایجاد می کنیم.

|  |
| --- |
| import {HttpClient} from '@angular/common/http'  import { Injectable } from '@angular/core';  import { IPerson } from 'src/IPerson';  import{Observable} from 'rxjs'  @Injectable({  providedIn: 'root'  })  export class MyserviceService {  private url: string = 'https://gorest.co.in/public/v2/users';  constructor( private http: HttpClient) { }  getData(): Observable<IPerson[]>  {  return this.http.get<IPerson[]>(this.url);  }  } |

متغیر تعریف شده با عنوان url لینک API است.

در کامپوننت مورد نظر دیتای خروجی را subscribe می کنیم.

|  |
| --- |
| @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css']  })  export class AppComponent implements OnInit {  title = 'Angular-Training';  constructor(private myservice: MyserviceService){  }  ngOnInit(): void {  this.myservice.getData().subscribe(data => this.persons = data)  }  persons: any = [];  } |

آرایه ایجاد شده را در کامپوننت html استفاده می کنیم.

|  |
| --- |
| <table>  <thead>  <th>id</th>  <th>Name</th>  </thead>  <tbody>  <tr \*ngFor="let item of persons">  <td>{{item.id}}</td>  <td>{{item.name}}</td>  </tr>  </tbody>  </table> |

# HTTP Error Handling

اگر درخواست در سرور ناموفق باشد، HttpClient به جای پاسخ موفق، یک شی خطا را برمی گرداند. همان سرویسی که تراکنش های سرور شما را انجام می دهد باید بازرسی، تفسیر و حل خطا را نیز انجام دهد.

هنگامی که خطایی رخ می دهد، می توانید جزئیاتی خطای رخ داده شده را به دست آورید. در برخی موارد، ممکن است بخواهید به طور خودکار درخواست را دوباره امتحان کنید.

دو نوع خطا ممکن است رخ دهد:

خطای سمت سرور، سرور ممکن است درخواست را رد کند و یک پاسخ HTTP با کد وضعیت مانند 404 یا 500 برگرداند.

خطای سمت کلاینت، ممکن است مشکلی در سمت کلاینت رخ دهد، مانند یک خطای شبکه که از تکمیل موفقیت آمیز درخواست جلوگیری می کند یا رخ دادن یک خطای نحوی، وضعیت این خطاها روی 0 تنظیم شده است و ویژگی error حاوی یک شی ProgressEvent است که نوع آن ممکن است اطلاعات بیشتری را ارائه دهد.

HttpClient هر دو نوع خطا را در HttpErrorResponse خود می گیرد. برای شناسایی علت خطا، آن پاسخ را بررسی کنید.

به عنوان مثال در سرویسی که دیتا را دریافت می کنید(people.service) کد زیر را قرار دهید.

|  |
| --- |
| import { Injectable } from '@angular/core';  import { HttpClient, HttpErrorResponse } from '@angular/common/http';  import { Observable, catchError, retry, throwError } from 'rxjs'  import { IPeople } from 'src/IPeople';  @Injectable({  providedIn: 'root'  })  export class PeopleService {  private url: string = 'https://jsonplaceholder.typicode.com/users'  constructor(private http: HttpClient) { }  getData(): Observable<IPeople[]> {  return this.http.get<IPeople[]>(this.url).pipe(retry(3), catchError(this.handleError));  }  handleError(error: HttpErrorResponse) {  if (error.status === 0) {  // A client-side or network error occurred. Handle it accordingly.  console.error('An error occurred:', error.error);  } else {  // The backend returned an unsuccessful response code.  // The response body may contain clues as to what went wrong.  console.error(  `Backend returned code ${error.status}, body was: `, error.error);  }  // Return an observable with a user-facing error message.  return throwError(() => new Error('Something bad happened; please try again later.'));  }  } |

جهت کسب اطلاعات بیشتر می توانید مستندات مربوط به HTTP Error Handling را در مستندات انگولار مشاهده کنید.

# Routing

در یک برنامه تک صفحه ای (Single Page App) کاربر با کلیک کردن روی لینک‌های مختلف بدون لود کردن و بارگذاری مجدد صفحه، می‌تواند به لینک موردنظر انتقال پیدا کند.

برای مدیریت ناوبری از یک نمای به نمای دیگر، از Angular Router استفاده می شود. روتر با تفسیر URL مرورگر به عنوان دستورالعملی برای تغییر نما، ناوبری را فعال می کند.

1. ابتدا RouterModule و Routes را به ماژول مسیریابی خود ایمپورت کنید. Angular CLI این مرحله را به صورت خودکار انجام می دهد. CLI همچنین یک آرایه Routes برای مسیرهای شما تنظیم می‌کند و آرایه‌های Import و Export را برای @NgModule() پیکربندی می‌کند.
2. برای تعریف مسیردهی در نرم‌افزار خود ابتدا وارد فایل app.module.ts‌ شده و تمام مسیرهای نرم‌افزار خود را در آن وارد کنید.

|  |
| --- |
| const routes: Routes = [  { path: 'first-component', component: FirstComponent },  { path: 'second-component', component: SecondComponent },  ]; |

کلمه‌کلیدی path: این کلمه برای نشان دادن مسیر صفحه می‌باشد. به عنوان مثال first-component' به مسیر http://localhost:4200/ first-component' اشاره می‌کند.

کلمه‌کلیدی component: از این کلمه برای تعریف کامپوننت مرتبط با مسیر استفاده می‌شود به عنوان مثال کامپوننت مرتبط با مسیر http://localhost:4200/ first-component معادل FirstComponent است که لیست کاربران را نمایش می‌دهد.

1. فایل app.component.html‌ را باز کنید و در نهایت به جای اسامی کامپوننت‌ها که به صورت تگ app-component‌ هستند یک ساختار جامع به نام router-outlet قرار دهیم تا به قالب HTML اطلاع دهیم که مسیرهای ما مشخص هستند و دیگر تمام کامپوننت‌ها را در یک صفحه لود نکن و متناسب با هر مسیری اطلاعات کامپوننت را نمایش بده! بنابراین در این فایل داریم:

|  |
| --- |
| <h1>Angular Router App</h1>  <!-- This nav gives you links to click, which tells the router which route to use (defined in the routes constant in AppRoutingModule) -->  <nav>  <ul>  <li><a routerLink="/first-component" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a></li>  <li><a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a></li>  </ul>  </nav>  <!-- The routed views render in the <router-outlet>-->  <router-outlet></router-outlet> |

در صورتی که بخواهید صفحه 404 مربوط به پیدا نکردن صفحه مورد نظر را پیاده سازی کنید، کافی است یک کامپوننت جدید ایجاد کنید و روت خود را به صورت زیر تعریف کنید.

|  |
| --- |
| const routes: Routes = [  { path: 'first-component', component: FirstComponent },  { path: 'second-component', component: SecondComponent },  { path: '\*\*', component: PageNotFoundComponent },  ]; |

## RouterLinkActive

برای متمایز کردن لینک صفحه ای که در آن قرار داریم استفاده می شود. کافی است routerLinkActive را به لینک خود اضافه کنید و کلاس های لازم را به آن اختصاص دهید.

|  |
| --- |
| <nav>  <ul>  <li><a routerLink="/first-component" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a></li>  <li><a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a></li>  </ul>  </nav> |

اگر در روت های تعریف شده از گزینه “” استفاده کرده اید(به این منظور استفاده می شود که اگر در لینک به چیزی اشاره نشده بود به کامپوننت تعریف شده مراجعه شود) گزینه زیر را در تعریف لینک های خود استفاده کنید.

|  |
| --- |
| <nav>  <ul>  <li><a routerLink="/first-component" routerLinkActive="active" ariaCurrentWhenActive="page" [routerLinkActiveOptions]="{exact: true}">Home</a></li>  <li><a routerLink="/first-component" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a></li>  <li><a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a></li>  </ul>  </nav> |