



**One framework.
Mobile & desktop.**

6강. 서비스 및 모듈



DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

06 강.

서비스 (service) 및 모듈 (module)

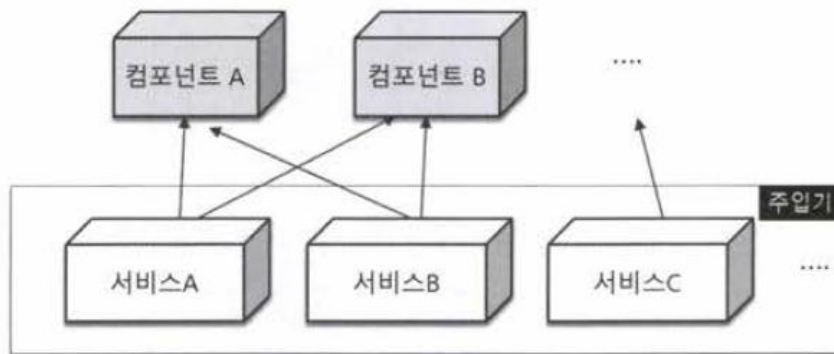
□ 1) 서비스 개요

* 서비스 개요

- Angular에서 서비스는 외부에 정의된 의존성 주입 대상이 되는 싱글톤 클래스로서 @Injectable() 을 사용하여 정의한다.
- 컴포넌트에서 사용하기 위해서는 providers 속성에 선언되어야 한다.
- 뷰와 관련된 로직만 component 로 만들고 다른 모든 로직은 서비스로 구현한다.

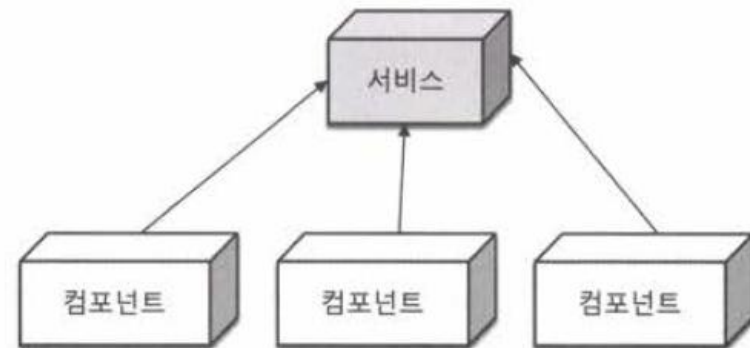
* 서비스 역할

가. 관심사의 분리
(AOP: Aspect Oriented Programming)



관심사에 따라 컴포넌트에 다르게 주입되는 서비스

나. 컴포넌트 중간에서 데이터 중개자의 역할



컴포넌트 중간에서 서비스가 데이터를 중개

□ 2) 서비스 사용 방법

1) 서비스를 생성한다.

Service

```
ng g service MyService // Creates MyService
```

TS hello.service.ts ×

my-app > src > app > TS hello.service.ts > ...

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class HelloService {
7
8   constructor() { }
9 }
```

```
C:\angular\my-app>ng g service hello
```

```
CREATE src/app/hello.service.spec.ts (352 bytes)
```

```
CREATE src/app/hello.service.ts (134 bytes)
```

▼ app

TS app-routing.module.ts

app.component.css

<> app.component.html

TS app.component.spec.ts

TS app.component.ts

TS app.module.ts

TS hello.service.spec.ts U

TS hello.service.ts U

□ 2) 서비스 사용 방법

2) 컴포넌트의 공통 기능 메서드 구현

“Hello 서비스” 문자열을 반환 해주는 공통 기능의 sayHello() 메서드를 추가한다.

```
hello.service.ts X
y-app > src > app > TS hello.service.ts > ...
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class HelloService {
7
8    constructor() { }
9
10     //공통 기능 구현
11     sayHello():string{
12       return "Hello 서비스";
13     }
14 }
```

□ 2) 서비스 사용 방법

3) 서비스 등록

서비스를 사용하기 위해서 컴포넌트에 providers 속성으로 등록한다

```
app.component.ts ×
y-app > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  import { HelloService } from './hello.service';
4
5  @Component({
6    selector: 'app-root',
7    templateUrl: './app.component.html',
8    styleUrls: ['./app.component.css'],
9    providers: [HelloService]
10 })
11 export class AppComponent {
12   title = 'my-app';
13 }
```

□ 2) 서비스 사용 방법

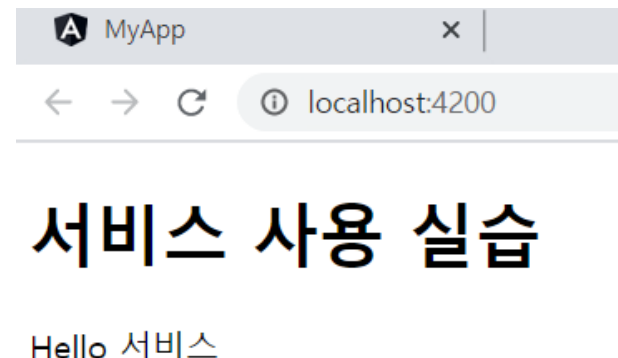
4) 의존성 주입 (Dependency Injection : DI)

컴포넌트 클래스의 생성자를 사용하여 Hello서비스를 의존성 주입(DI) 받는다.

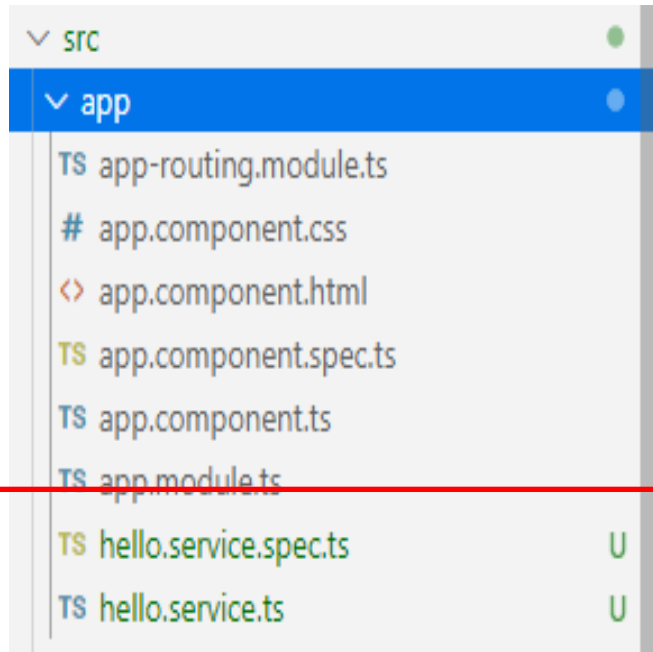
```
export class AppComponent {  
  title = 'my-app';  
  
  //생성자를 이용한 의존성 주입  
  constructor(public helloService:HelloService){}  
}
```

5) 서비스 호출

```
app.component.html ×  
y-app > src > app > < app.component.html > ...  
1 | <h1>서비스 사용 실습</h1>  
2 | {{helloService.sayHello()}}
```



□ 서비스의 생성



```
C:\angular_chul\chul-app>ng g service hello
```

```
An unhandled exception occurred. Schematic "service"
See "C:\Users\ledzep\AppData\Local\Temp\ng-G37kdv\..."
```

```
C:\angular_chul\chul-app>ng g service hello
```

```
CREATE src/app/hello.service.spec.ts (352 bytes)
```

```
CREATE src/app/hello.service.ts (134 bytes)
```

```
C:\angular_chul\chul-app>
```

TS hello.service.ts X

src > app > TS hello.service.ts > HelloService > sayHello

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class HelloService {
7
8   constructor() { }
9   //공통기능 구현
10  sayHello():string{
11    return "Hello 서비스";
12  }
13 }
```


□ 서비스의 등록



App.component.ts에서 사용하기 HelloService등록

TS app.component.ts × TS hello.service.ts

src > app > TS app.component.ts > AppComponent

```
2 import { HelloService } from './hello.service';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css'],
8   providers: [HelloService]
9 })
10 export class AppComponent {
11   title = 'chul-app';
12 }
13
```

<> app.component.html × TS app.component.ts TS

src > app > <> app.component.html > ...

```
1 <h1>서비스 사용의 실습</h1>
2 {{helloService.sayHello()}}
```

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers: [HelloService]
})
export class AppComponent {
  title = 'chul-app';
  //생성자를 통한 의존성 주입
  constructor(helloService:HelloService){}
}
```

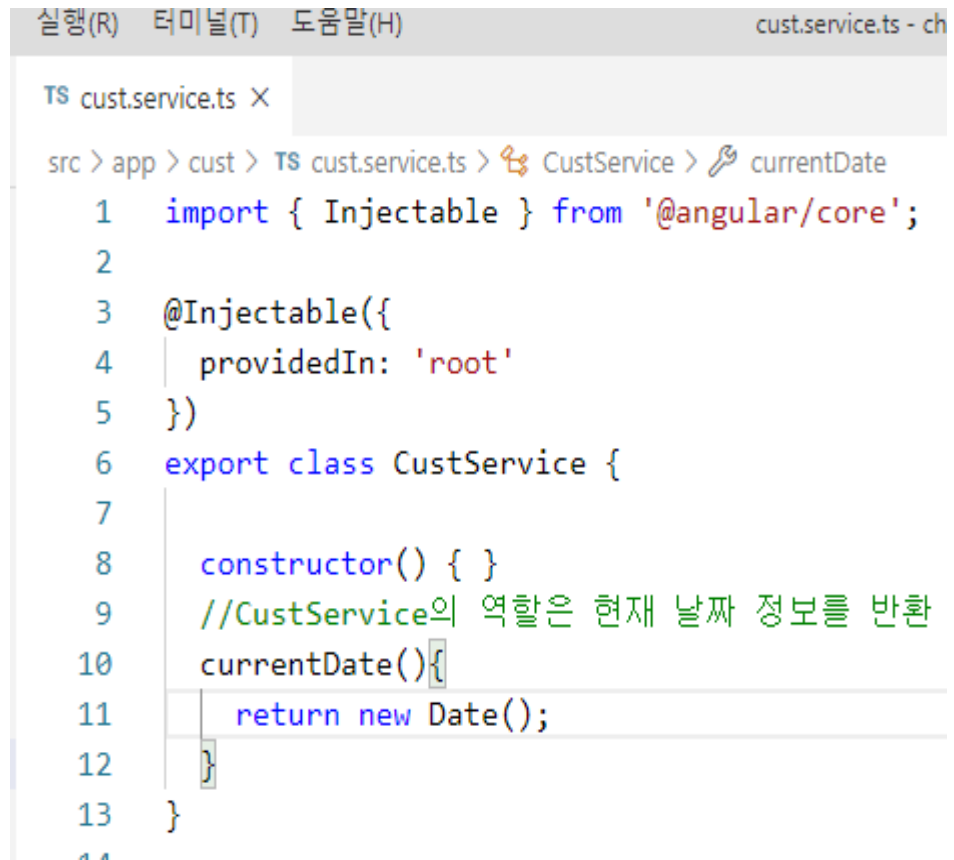
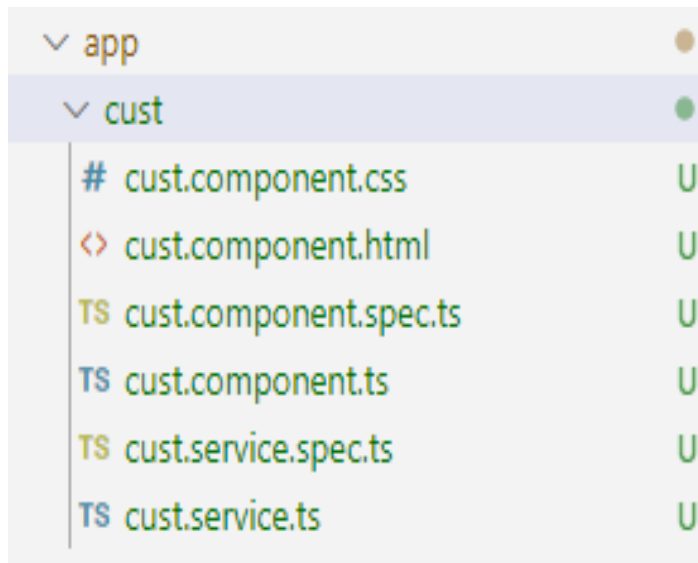
□ 자식컴포넌트에서 *service* 생성

Ng g component cust 후 cust폴더에서 서비스 생성

```
C:\angular_chul\chul-app\src\app\cust>ng g service cust
```

```
CREATE src/app/cust/cust.service.spec.ts (347 bytes)
```

```
CREATE src/app/cust/cust.service.ts (133 bytes)
```



□ 모듈에 service 등록



TS cust.component.ts X

src > app > cust > TS cust.component.ts > CustComponent

```
1 import { Component, OnInit } from '@angular/core';
2 import { CustService } from '../cust.service'
3
4 @Component({
5   selector: 'app-cust',
6   templateUrl: '../cust.component.html',
7   styleUrls: ['../cust.component.css'],
8   providers:[CustService]//서비스 등록
9 })
10 export class CustComponent implements OnInit {
11   //생성자를 통해 자동주입
12   constructor(public custService:CustService) { }
13   getCurrentDate():Date{
14     return this.custService.currentDate();
15   }
16 }
```

src > app > cust > <> cust.component.html > ...

```
1 <h1>모듈실습</h1>
2 현재시간 {{getCurrentDate()}}
```

TS cust.service.ts

<> app.component.html X

TS cust.com

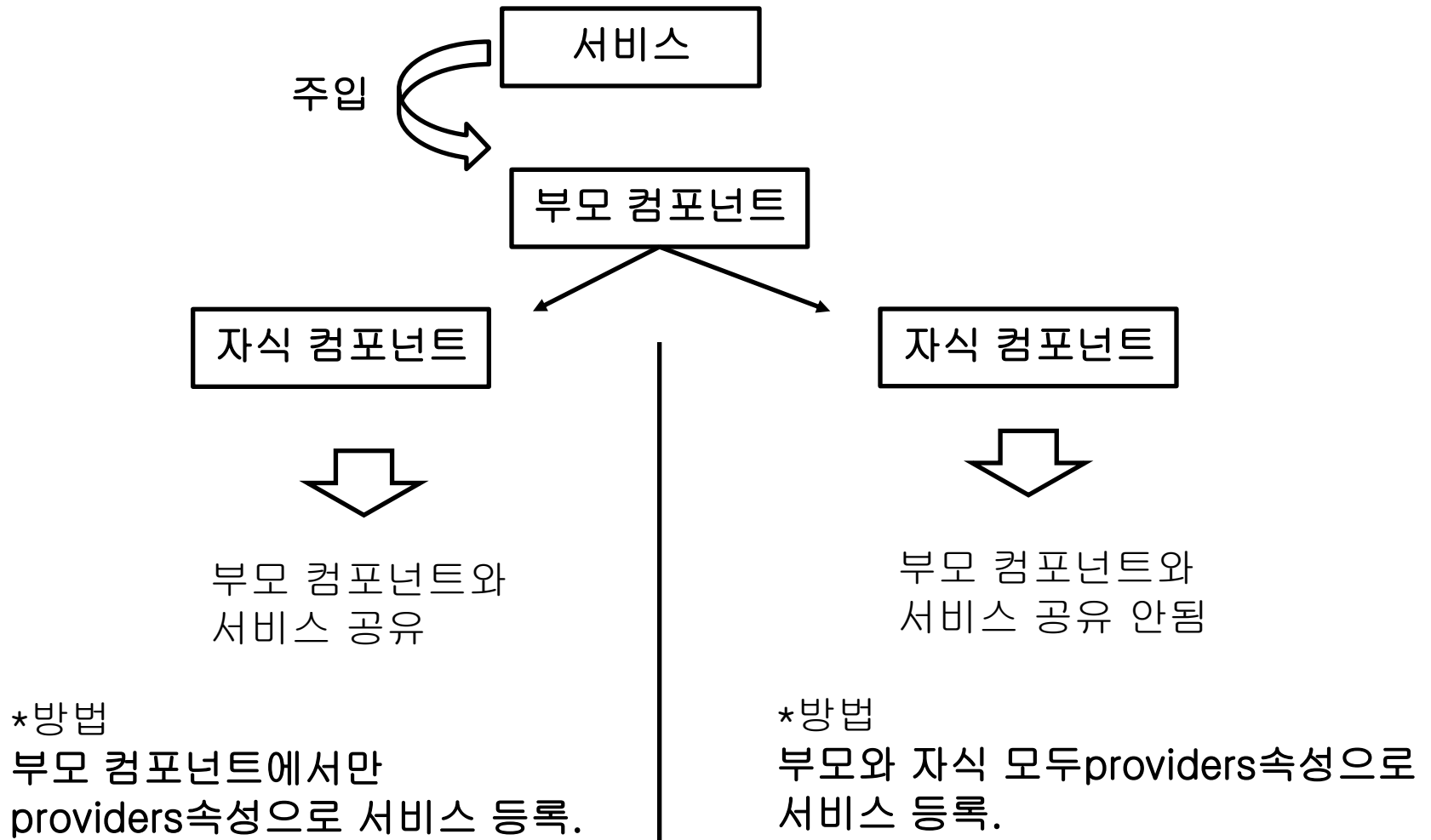
src > app > <> app.component.html > app-cust

```
1 <app-cust></app-cust>
```

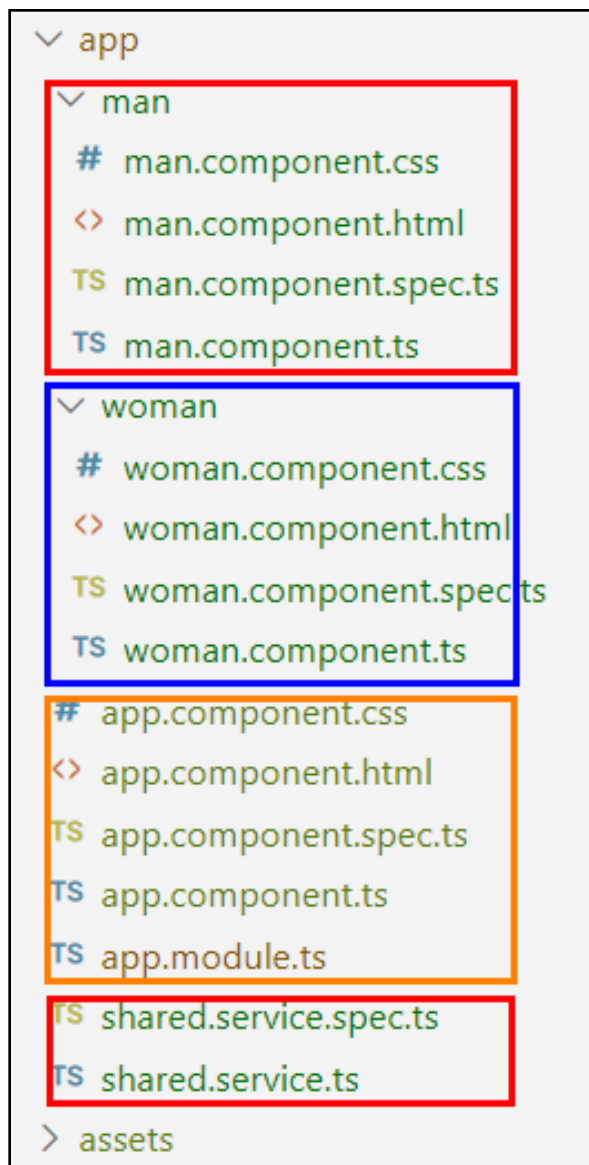
□ 3) 중첩 컴포넌트의 공유 서비스



중첩 컴포넌트인 경우 부모와 자식 컴포넌트에서 서비스를 공유해서 사용 가능.



□ 3) 중첩 컴포넌트의 공유 서비스



```

C:\angular\my-app>ng g component Man
CREATE src/app/man/man.component.html (18 bytes)
CREATE src/app/man/man.component.spec.ts (607 bytes)
CREATE src/app/man/man.component.ts (263 bytes)
CREATE src/app/man/man.component.css (0 bytes)
UPDATE src/app/app.module.ts (384 bytes)
  
```

```

C:\angular\my-app>ng g component Woman
CREATE src/app/woman/woman.component.html (20 bytes)
CREATE src/app/woman/woman.component.spec.ts (621 bytes)
CREATE src/app/woman/woman.component.ts (271 bytes)
CREATE src/app/woman/woman.component.css (0 bytes)
UPDATE src/app/app.module.ts (462 bytes)
  
```

```

C:\angular\my-app>ng g service shared
CREATE src/app/shared.service.spec.ts (357 bytes)
CREATE src/app/shared.service.ts (135 bytes)
  
```

❑ 3) 중첩 컴포넌트의 공유 서비스

Man:service등록안함,
Woman: service등록함

```
shared.service.ts X
y-app > src > app > TS shared.service.ts > ...
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class SharedService {
7
8    names:string=[];
9
10   addName(n:string){
11     this.names.push(n);
12   }
13 }
```

공유됨

```
export class ManComponent {

  constructor(public service:SharedService) { }

  addName(n:string){
    this.service.addName(n);
    console.log(this.service.names)
  }
}
```

```
export class AppComponent {
  title = 'my-app';

  constructor(public service:SharedService){}

  addName(n:string){
    this.service.addName(n);
    console.log(this.service.names)
  }
}
```

```
export class WomanComponent {

  constructor(public service:SharedService) { }

  addName(n:string){
    this.service.addName(n);
    console.log(this.service.names)
  }
}
```

공유안됨

3) 중첩 컴포넌트의 공유 서비스

The screenshot shows a web browser at `localhost:4200` displaying an Angular application. The application has three main components: `my-app`, `man works!`, and `woman works!`. Each component has a button labeled "추가" (Add). Red arrows point from these buttons to the corresponding log entries in the browser's console. The console shows the following log entries:

- Angular is running in the [WDS] Live Reloading enabled
- `["parent"]` (linked from the `parent 추가` button)
- `(2) ["parent", "man"]` (linked from the `man 추가` button)
- `["woman"]` (linked from the `woman 추가` button)

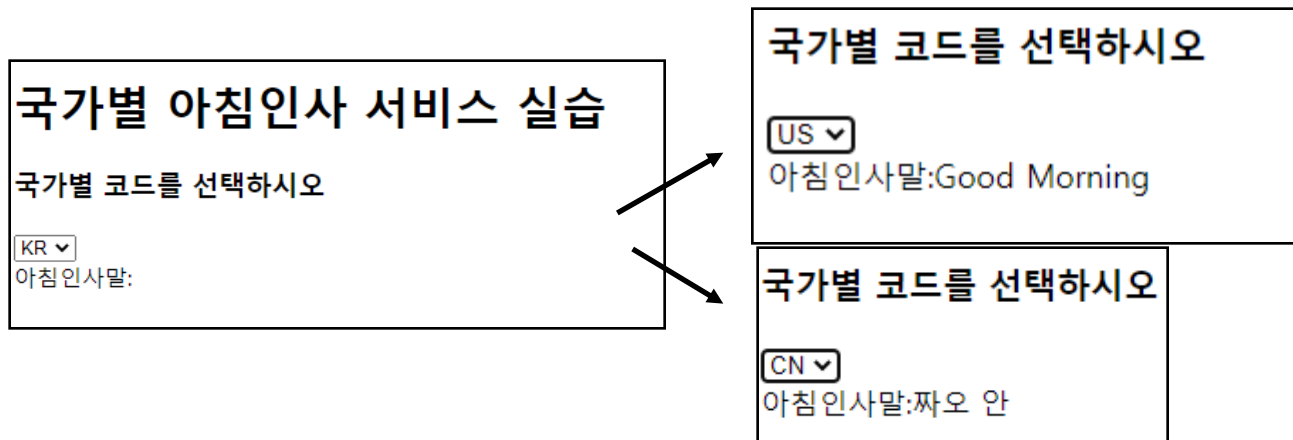
□ 실습 문제 1



다양한 언어로 아침인사를 반환하는 서비스를 구현 하시오.

- 컴포넌트에서 국가별 코드를 선택하여 서비스에 전달한다.
- 서비스에서 국가별 코드에 따라서 반환되는 아침인사가 달라진다.

국가별 코드	아침인사말
KR	안녕하세요. 좋은 아침입니다.
US	Good Morning
CN	짜오 안
JP	오하요~ 고자이마스



실습1

```
C:\angular_chul\chul-app>ng g service morning
CREATE src/app/morning.service.spec.ts (362 bytes)
CREATE src/app/morning.service.ts (136 bytes)
```

```
5 export class MorningService {
7
3   constructor() { }
3   //code입력 받고 인사말 반환
3   morning(code:string):string{
1       var mesg="";
2       if(code=="KR"){
3           mesg="안녕하세요. 좋은 아침입니다.
4       }else if(code=="US"){
5           mesg="Good Morning";
6       }else if(code=="CN"){
7           mesg="짜오안";
8       }else{
9           mesg="오하이요 고자이마스 ~";
10      }
11      return mesg;
12  }
13  }
```

```
1 @Component({
2   selector: 'app-root',
3   templateUrl: './app.component.html',
4   styleUrls: ['./app.component.css'],
5   providers:[MorningService] //서비스 등록
6 })
7 export class AppComponent {
8   title = 'chul-app';
9   //주입
10  constructor(public service:MorningService){}
11
12  mesg="";
13  //1.MorningService생성
14  //2.코드를 받아서 인사말을 반환하는 메서드 작성 : morning(){}
15  handleEvent(code:string){
16      this.mesg= this.service.morning(code);
17  }
18  }
```







```
<select (change)="handleEvent(kkk.value)" #kkk>
  <option>KR</option>
  <option>US</option>
  <option>CN</option>
  <option>JP</option>
</select><br>
인사말 : {{mesg}}
```

□ 실습 문제 2



도서목록 데이터를 서비스에 저장하고, 컴포넌트에서 주입(injection) 받아서 데이터를 출력하는 어플리케이션을 구현 하시오.

도서 목록 6권

-  위험한 식탁
-  공부의 비결
-  오메르타
-  행복한 여행
-  해커스 토익
-  도로 안내서

```
book.service.ts ×
key > src > app > book > TS book.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { Book } from './book';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class BookService {
8
9    constructor() { }
10
11    books:Book[] = [{id:'p01',name:'위험한 식탁',price:2000,date:'20170401',img:'a.jpg'},
12                    {id:'p02',name:'공부의 비결',price:3000,date:'20170402',img:'b.jpg'},
13                    {id:'p03',name:'오메르타',    price:2500,date:'20170401',img:'c.jpg'},
14                    {id:'p04',name:'행복한 여행',price:4000,date:'20170401',img:'d.jpg'},
15                    {id:'p05',name:'해커스 토익',price:2000,date:'20170401',img:'e.jpg'},
16                    {id:'p06',name:'도로 안내서',price:2000,date:'20170401',img:'f.jpg'},
17  ];
18
19    bookList():Book[] {
20      return this.books;
21    }
22  }
```

```
C:\angular_chul\chul-app>ng g service book
CREATE src/app/book.service.spec.ts (347 bytes)
CREATE src/app/book.service.ts (133 bytes)
```

```
> app > TS book.ts > Book >
1 export class Book{
2     id:string;
3     name:string;
4     price:number;
5     date:string;
6     img:string;
7 }
```

```
TS book.service.ts X TS book.ts
src > app > TS book.service.ts > BookService > bookList
2 import {Book} from "../book"
3 @Injectable({
4     providedIn: 'root'
5 })
6 export class BookService {
7
8     constructor() { }
9     books:Book[]=[
10         {id:"p01", name:"위험한 식탁", price:2000, date:'20170202', img:'a.jpg'},
11         {id:"p02", name:"공부의 비결", price:3000, date:'20170202', img:'b.jpg'},
12         {id:"p03", name:"오메르타", price:2400, date:'20201202', img:'c.jpg'},
13         {id:"p04", name:"행복한 여행", price:4000, date:'20180202', img:'d.jpg'},
14         {id:"p05", name:"해커스 토익", price:1000, date:'20130202', img:'e.jpg'},
15         {id:"p06", name:"도로 안내서 ", price:2000, date:'20150202', img:'f.jpg'}
16     ];
17     bookList():Book[]{
18         return this.books;
19     }
```



```
import { Component, OnInit } from '@angular/core';
import { BookService } from '../book/book.service';
import { Book } from '../book/book';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers:[BookService]
})
export class AppComponent{
  title = '도서 목록';

  books:Book[];
  constructor(public bookService:BookService){
    this.books = bookService.bookList();
  }
}
```

C:\angular_chul\chul-app>ng serve --open

chunk {main} main.js, main.js.map (main) 14.5 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 14.1 kB [initial]

app.component.html - chul-app - Visual Studio Code

book.ts

app.component.html X

TS book.component.ts

TS bookservice.service.ts

> app > app.component.html > ...

```
1 | <app-book [booklist]="books" [title]="title"> </app-book>
```

```

src > app > book > TS book.component.ts > BookComponent
1  import { Component, OnInit, Input } from '@angular/core';
2  @Component({
3    selector: 'app-book',
4    templateUrl: './book.component.html',
5    styleUrls: ['./book.component.css']
6  })
7  export class BookComponent implements OnInit {
8    @Input() booklist:[];
9    @Input() title:string;
10   constructor() { }
11
12   ngOnInit(): void {
13   }
14 }
src > app > book > <> book.component.html > ...
1  <p>book works!</p>
2  <ul *ngFor="let book of booklist">
3    <img src='../assets/image/{{book.img}}' width='100' height='100'>{{book.name}}
4  </ul>
5

```

□ 4) 모듈이란?

* 모듈 (module)

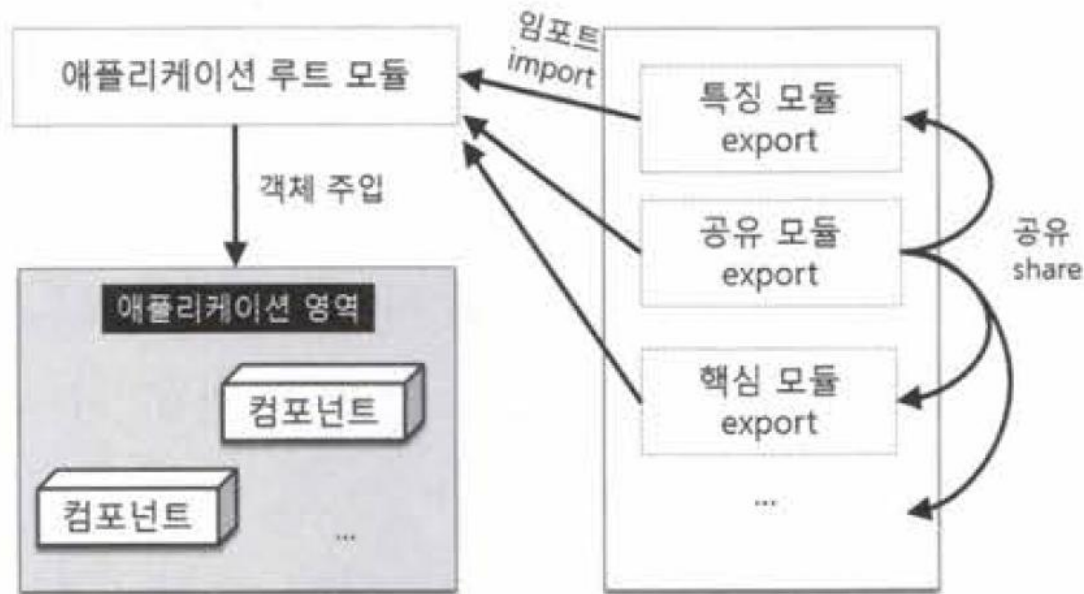
- Angular는 특정 범주의 각 기능들을 묶어서 관리할 수 있도록 모듈(module) 기능 제공.
- 모듈을 사용하면 그룹별 관리가 가능, 그룹 내에서 공동으로 사용할 수 있는 기능추가 가능.
- 기본적으로 제공되는 `app.module.ts` 파일이 애플리케이션의 루트 모듈이다.

Angular 라이브러리 모듈

모듈 패키지명	요약
@angular/common	파이프, 구조 지시자, 속성 지시자와 관련된 모듈을 포함합니다.
@angular/core	Angular를 구성하는 주요 요소의 장식자 및 핵심 모듈을 포함합니다.
@angular/forms	폼 관련 모듈이나 지시자를 포함합니다.
@angular/http	HTTP 통신과 관련된 모듈을 포함합니다.
@angular/platform-browser	브라우저 모듈, DOM 새니타이저(sanitizer) 등 브라우저와 관련된 모듈을 포함합니다.
@angular/router	라우터 관련 모듈이나 지시자를 포함합니다.
@angular/testing	테스팅 관련 모듈을 포함합니다.

□ 5) 모듈 구성 방법

* 모듈 구성 방법과 애플리케이션 루트 모듈



Angular의 모듈 구성 방법

□ 6) 커스텀 모듈 생성

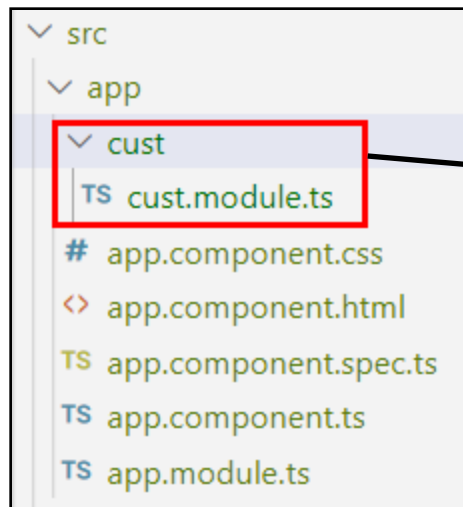
* 새로운 모듈 생성 방법

```
C:\angular_chul\chul-app>ng g module Cust  
CREATE src/app/cust/cust.module.ts (190 bytes)
```

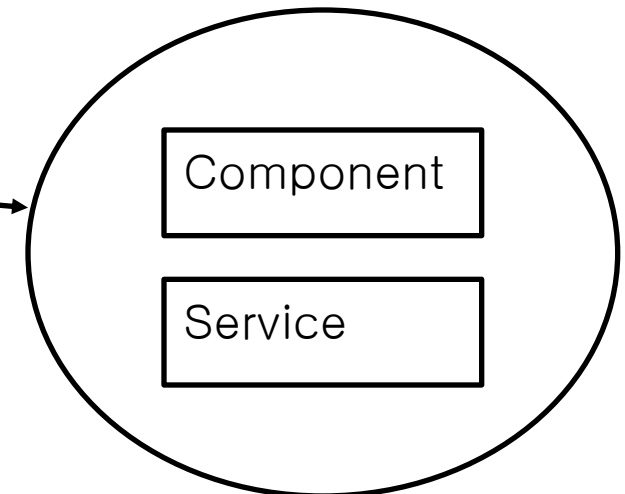
```
C:\angular_chul\chul-app>
```

1. 사용자 모듈을 생성한다. (예: 고객관리 목적의 cust 모듈)

```
C:\angular\my-app>ng g module Cust  
CREATE src/app/cust/cust.module.ts (190 bytes)
```



Cust 모듈



□ 6) 커스텀 모듈 생성

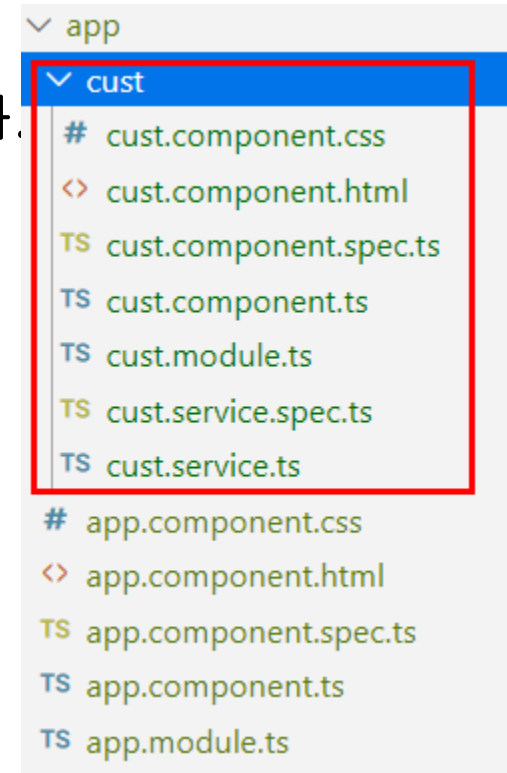
```
C:\angular\my-app>ng g module Cust  
CREATE src/app/cust/cust.module.ts (190 bytes)
```

2. 생성된 cust 모듈에 필요한 컴포넌트와 서비스를 추가한다.

```
C:\angular\my-app>ng g component Cust  
CREATE src/app/cust/cust.component.html (19 bytes)  
CREATE src/app/cust/cust.component.spec.ts (614 bytes)  
CREATE src/app/cust/cust.component.ts (267 bytes)  
CREATE src/app/cust/cust.component.css (0 bytes)  
UPDATE src/app/Cust/cust.module.ts (259 bytes)
```

```
C:\angular\my-app>cd src/app/cust
```

```
C:\angular\my-app\src\app\cust>ng g service Cust  
CREATE src/app/cust/cust.service.spec.ts (347 bytes)  
CREATE src/app/cust/cust.service.ts (133 bytes)
```



□ 6) 커스텀 모듈 생성

3. 컴포넌트와 서비스를 구현한다.

```
cust.service.ts X
~app > src > app > cust > TS cust.service.ts > ...
1  import { Injectable } from
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class CustService {
7
8    constructor() { }
9
10   currentDate(){
11     return new Date();
12   }
13 }
```

```
cust.component.ts X
~app > src > app > cust > TS cust.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2  import { CustService } from './cust.service';
3
4  @Component({
5    selector: 'app-cust',
6    templateUrl: './cust.component.html',
7    styleUrls: ['./cust.component.css'],
8    providers: [CustService]
9  })
10 export class CustComponent{
11
12   constructor(public custService:CustService) { }
13
14   getMesg(){
15     return this.custService.currentDate();
16   }
17 }
```

□ 6) 커스텀 모듈 생성

4. 컴포넌트를 cust 모듈에 등록하고 exports 한다.

Cust 컴포넌트를 app.module.ts 파일인 애플리케이션 루트 모듈이 아닌 사용자가 정의한 cust.module.ts 파일에 등록한다.

외부 모듈에서 CustComponent를 사용하기 위하여 반드시 exports로 처리한다.

TS cust.module.ts X

src_6강_모듈1 > app > cust > TS cust.module.ts > CustModule

```
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { CustComponent } from "../cust.component";
4  @NgModule({
5    declarations: [CustComponent],
6    exports: [CustComponent], //CustComponent를 사용하기 위해 export처리함
7    imports: [
8      CommonModule
9    ]
10  })
```

❑ 6] 커스텀 모듈 생성

5. Cust 모듈을 app 루트 모듈에 등록한다.

사용자가 정의한 cust.module.ts 파일을 app.module.ts 루트 모듈에 등록한다.

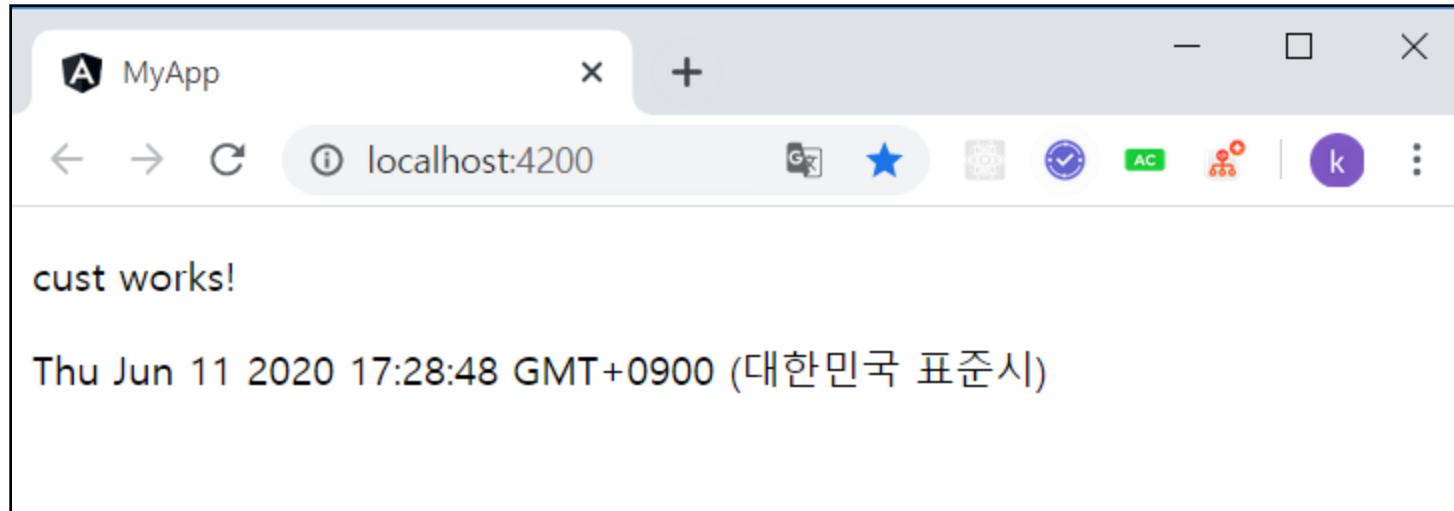
```
app.module.ts X
y-app > src > app > TS app.module.ts > ...
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { CustModule } from './cust/cust.module';
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent
10  ],
11   imports: [
12     BrowserModule,
13     CustModule
14  ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

```
ts <> app.component.html X TS cu
c > app > <> app.component.html > app-
1 | <app-cust></app-cust>
```

```
<> cust.component.html X TS cust.serv
src > app > cust > <> cust.component.htr
1 <p>cust works!</p>
2 {{getMesg()}}
3 |
```

□ 6) 커스텀 모듈 생성

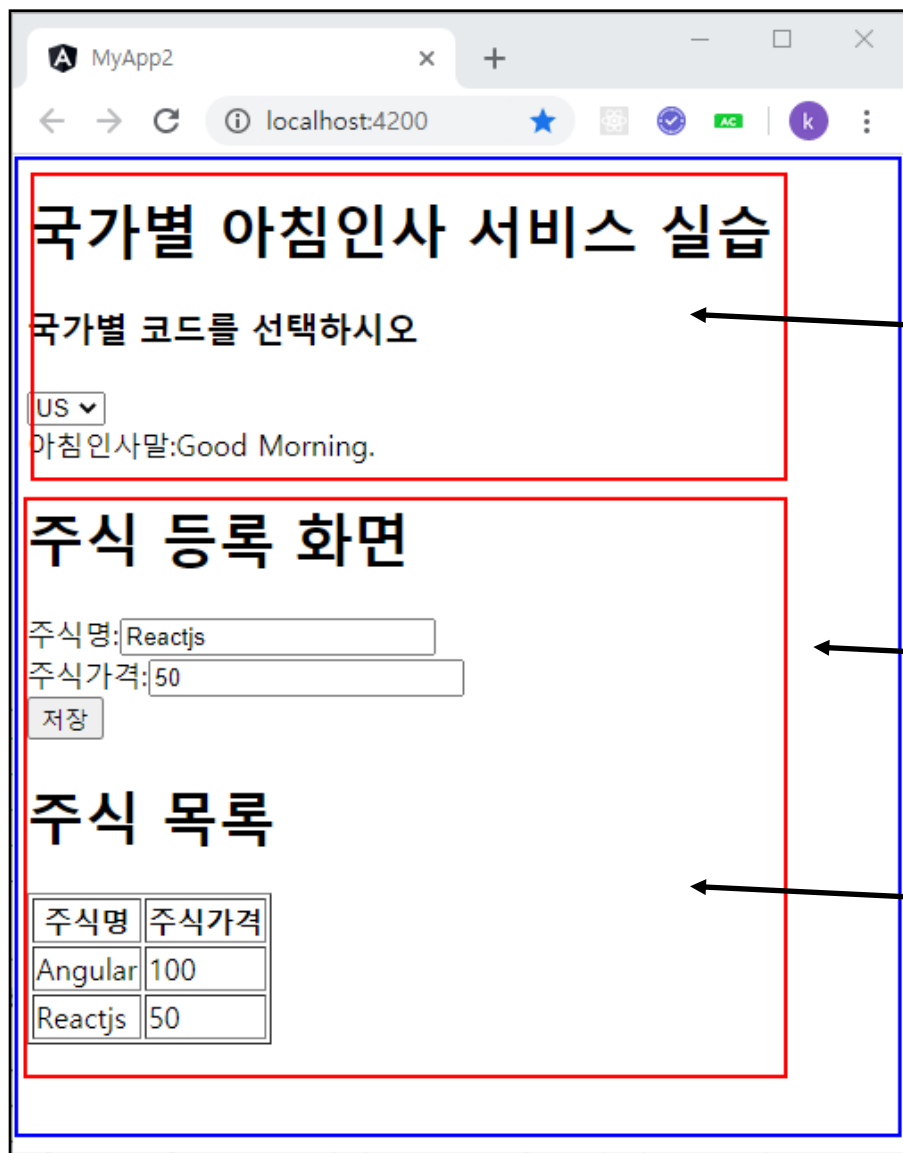
6. 실행



```
TS cust.service.ts    <> cust.component.html X <>
src > app > cust > <> cust.component.html > ...
1   <p>cust works!</p>
2   [{{getMesg()}}]
```

```
실행(R)  디버깅(D)  도움말(H)  ap
<> app.component.html X  TS cust.service.ts  <> cust.
src > app > <> app.component.html > app-cust
1 | <app-cust>[ ]/app-cust[ ]
```

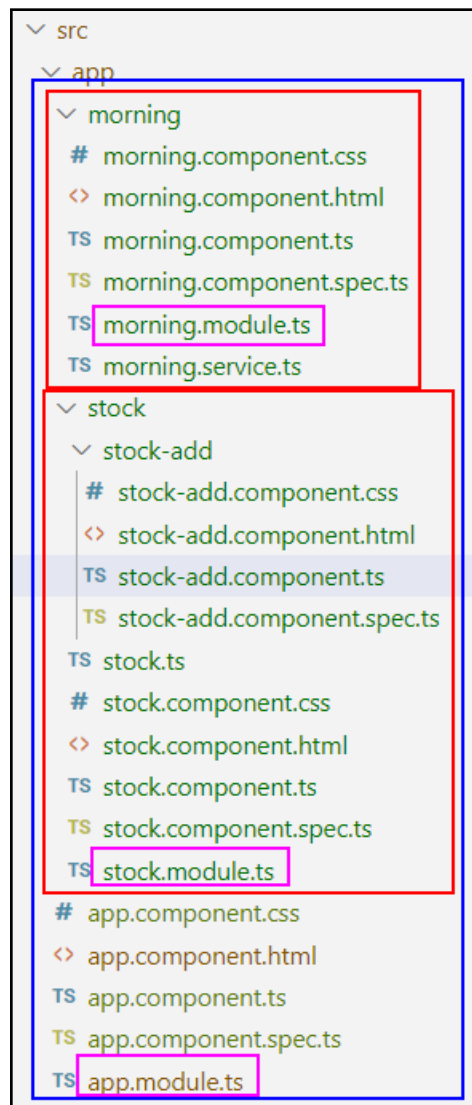
실습 문제 3



morning.module.ts

app.module.ts

stock.module.ts



```

C:\angular_chul\chul-app>ng g module stock
CREATE src/app/stock/stock.module.ts (191 bytes)

C:\angular_chul\chul-app>ng g module morning
CREATE src/app/morning/morning.module.ts (193 bytes)

C:\angular_chul\chul-app>ng g component morning
CREATE src/app/morning/morning.component.html (22 bytes)
CREATE src/app/morning/morning.component.spec.ts (633 bytes)
CREATE src/app/morning/morning.component.ts (279 bytes)
CREATE src/app/morning/morning.component.css (0 bytes)
UPDATE src/app/morning/morning.module.ts (265 bytes)

C:\angular_chul\chul-app>ng g component stock
CREATE src/app/stock/stock.component.html (20 bytes)
CREATE src/app/stock/stock.component.spec.ts (619 bytes)
CREATE src/app/stock/stock.component.ts (271 bytes)

```

```

C:\angular_chul\chul-app>cd src/app/morning

```

```

C:\angular_chul\chul-app\src\app\morning>ng g service morning
CREATE src/app/morning/morning.service.spec.ts (362 bytes)
CREATE src/app/morning/morning.service.ts (136 bytes)

```

```

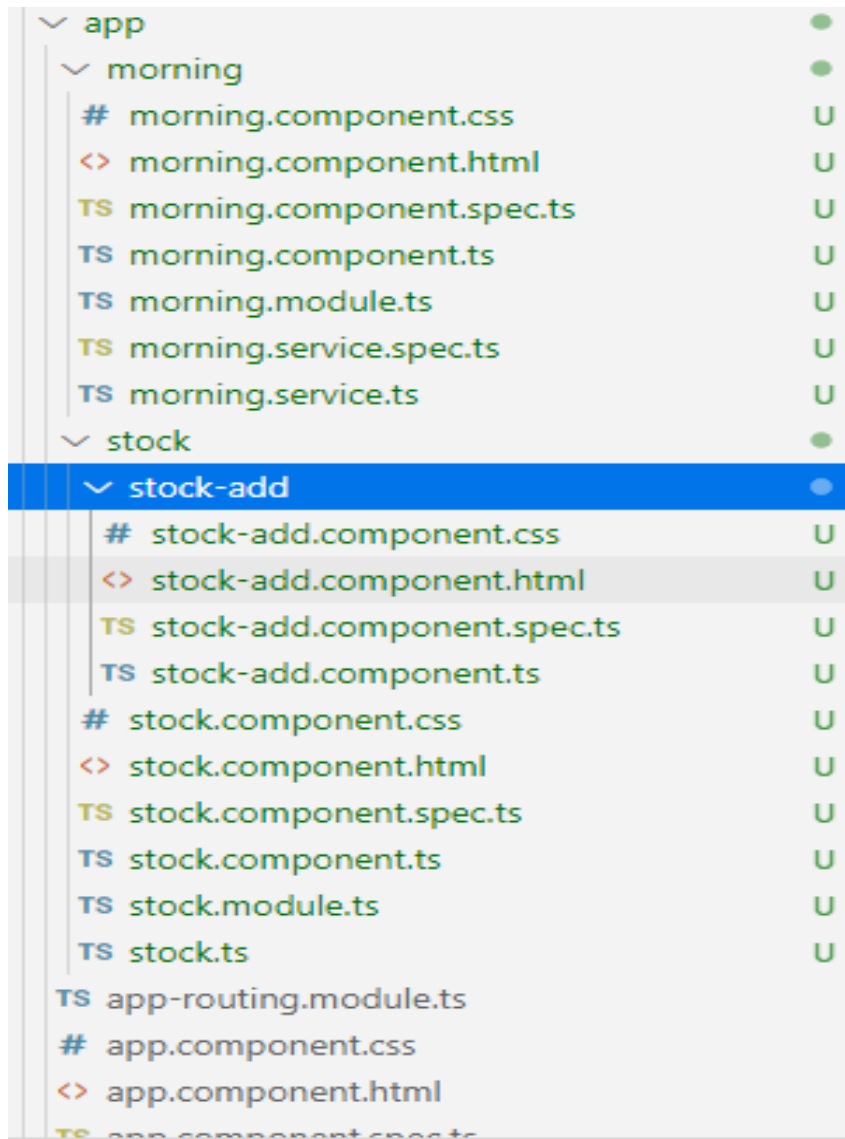
C:\angular_chul\chul-app\src>cd app/stock

```

```

C:\angular_chul\chul-app\src\app\stock>ng g component stock-add
CREATE src/app/stock/stock-add/stock-add.component.html (24 bytes)
CREATE src/app/stock/stock-add/stock-add.component.spec.ts (641 bytes)
CREATE src/app/stock/stock-add/stock-add.component.ts (286 bytes)
CREATE src/app/stock/stock-add/stock-add.component.css (0 bytes)
UPDATE src/app/stock/stock.module.ts (345 bytes)

```



TS stock.ts

src > app > stock > TS stock.ts > stock > stockF

```

1  export class stock{
2      stockSymbol:string;
3      stockPrice:number;
4  }
```


실습4-stock-add Component

```

src > app > stock > stock-add > <> stock-add.component.html > ...
<h1>주식등록화면</h1>
<div>
  주식명:<input type="text" #name><br>
  주식가격:<input type="text" #price><br>
  <button (click)="add({'stockSymbol':name.value, 'stockPrice':price.value})">저장</button>
</div>

```

```

TS stock.ts      <> stock-add.component.html  TS stock-add.component.ts ●
src > app > stock > stock-add > TS stock-add.component.ts > ⚙ StockAddComponent
10  constructor() {
11    stock:Stock;
12    //1.stock에 데이터 저장
13    //2. 부모컴포넌트에게 stock 전달=> @Output+이벤트(EventEmitter)
14    @Output() customEvent= new EventEmitter<Stock>();//<부모에게 전달한 데이터 객체 타입>
15    add(s:Stock){
16      this.stock= s;
17      this.customEvent.emit(this.stock);//emit (단하나의 값);
18    }
19  }
20

```

실습4-StockComponent



stock.component.html X TS stock.ts stock-add.component.html TS stock-add.component.ts

src > app > stock > stock.component.html > table > tr

```
1 <app-stock-add (customEvent)="handleEvent($event)"></app-stock-add>
2 <h1>주식 목록</h1>
3 <table border="1">
4   <tr>
5     <th>주식명</th>
6     <th>주식가격</th>
7   </tr>
8   <tr *ngFor="let stock of stocks">
9     <td>{{stock.stockSymbol}}</td>
10    <td>{{stock.stockPrice}}</td>
11  </tr>
12 </table>
```

```
export class StockComponent {
```

```
  constructor() { }
```

```
  //Stock배열 ==> stock 배열을 app.component.html 에서 바인딩해서 출력
```

```
  stocks:Stock[]=[];
```

```
  handleEvent(stock:Stock){
```

```
    this.stocks.push(stock);
```

```
  }
```

TS stock.module.ts X stock.component.html app.component.html TS

src > app > stock > TS stock.module.ts > StockModule

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { StockComponent } from './stock.component';
4 import { StockAddComponent } from './stock-add/stock-add.component';
5
6
7
8 @NgModule({
9   declarations: [StockComponent, StockAddComponent],
10  exports:[StockComponent], //StockComponent exports
11  imports: [
12    CommonModule
13  ]
14 })
15 export class StockModule { }
```

□ 실습4- morning Component

<> morning.component.html × <> app.component.html TS morning.module.ts

src > app > morning > <> morning.component.html > div

```

1 <h1>주식 등록 화면</h1>
2 <div>
3     주식명:<input type="text" #name><br>
4     주식가격:<input type="text" #price><br>
5     <button (click)="add({'stockSymbol':name.value, 'stockPrice':price.value})">저장</button>
6 </div>

```

```
export class MorningService {
```

```
    constructor() { }
```

```
    //code 입력받고 인사말 반환
```

```
    morning(code:string):string{
```

```
        var mesg = "";
```

```
        if(code == 'KR'){
```

```
            | mesg = "안녕하세요. 좋은 아침입니다.";
```

```
        }else if(code == 'US'){
```

```
            | mesg = "Good Morning.";
```

```
        }else if(code == 'CN'){
```

```
            | mesg = "짜오 안.";
```

```
        }else{
```

```
            | mesg = "오하이요~ 고자이마스";
```

```
        }
```

```
        return mesg;
```

```
    }
```

```
}
```

□ 실습4-Morning Component

```

8  //
9  export class MorningComponent {
10
11      constructor(public service:MorningService) { }
12      msg="";
13      //1.MorningService생성
14      //2.코드를 받아서 인사말을 반환하는 메서드 작성
15      handleEvent(code:string){
16          this.msg= this.service.morning(code);
17      }
18
19  }
20

```

```

TS morning.module.ts • <> morning.component.html TS stock.component.ts <> app
src > app > morning > TS morning.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { MorningComponent } from './morning.component';
4  @NgModule({
5      declarations: [MorningComponent],
6      exports:[MorningComponent], //MorningComponent exports
7      imports: [
8          CommonModule
9      ]
10 })
11 export class MorningModule { }

```

실습4- module의 import (appModule.ts)

```

src > app > TS app.module.ts > AppModule
4 import { AppRoutingModuleModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { MorningModule } from './morning/morning.module';
7 import { StockModule } from './stock/stock.module';
8 @NgModule({
9   declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModuleModule,
15     MorningModule,
16     StockModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
22

```

```

src > app > <> app.component.html > app-stock
1 <app-morning></app-morning>
2 <app-stock></app-stock>

```



수고하셨습니다.
