

Front-end Development With Angular5

2주차

네이버카페: angular & spring boot 으로하는 풀스택 웹 개발
<http://cafe.naver.com/ng2spring>

eastflag@gmail.com

010-3010-1482

이동기

* 신규 프로젝트 생성

ng new ngTodo --style=scss

만일 이미 생성되었다면,

ng set defaults.styleExt scss

* 라우팅 설계

사용자 사이트, 관리자 사이트 등 2개 이상의 사이트가 필요

⇒ appComponent에는 router-outlet만 뒤야 한다.

사용자 사이트 상단 공통 메뉴는 IndexComponent로 구성

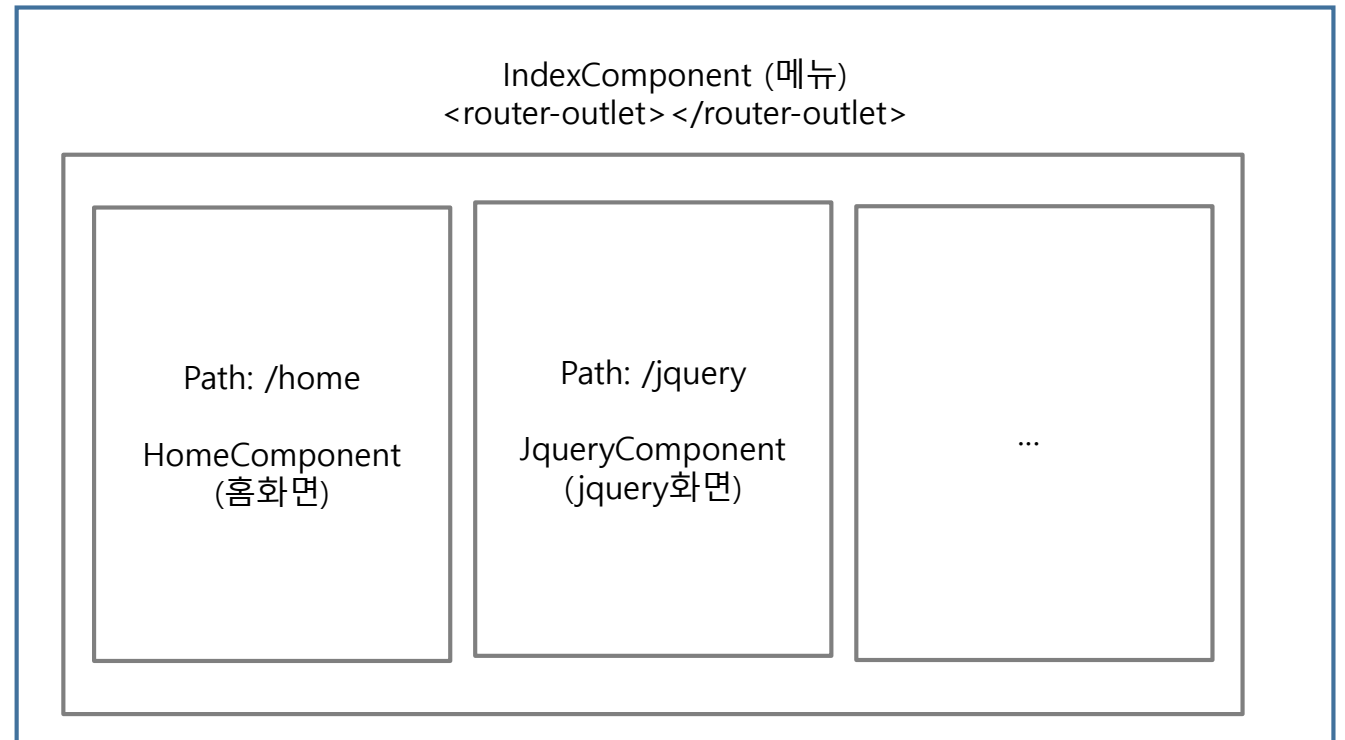
* 각 페이지 생성

ng g component index

ng g component home

ng g component jquery

AppComponent
<router-outlet> </router-outlet>

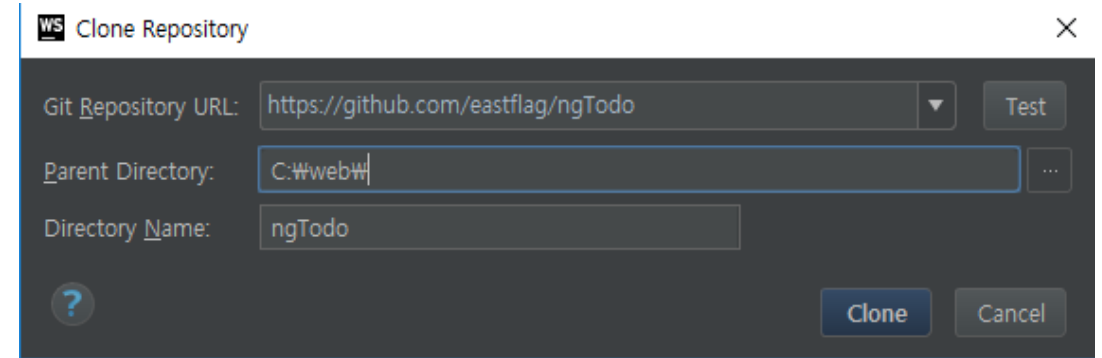


WebStrom은 프로젝트당 한 개의 윈도우를 가진다.
하나의 윈도우에는 본인 프로젝트
또다른 윈도우에 강사님의 프로젝트를 띄운다.

VCS > Checkout From Version Control > GitHub
오른쪽 정보를 입력.
본인 프로젝트와 이름이 같다면 이름을 다르게 주자.

Clone후 npm install로 라이브러리 설치

그리고, 수업중 강사님이 push를 하면
Git pull을 해서 소스를 참조하자.



* 라우팅 설정

app-routing.module.ts 생성

```
const routes: Routes = [  
  { path: '', component: IndexComponent, children: [  
    { path: '', component: HomeComponent},  
    { path: 'jquery', component: JQueryComponent},  
  ]},  
  // 참고: 향후 관리자 생성 모듈  
  // { path: 'admin', loadChildren: 'app/admin/admin.module#AdminModule'}  
];  
  
@NgModule({  
  imports: [ RouterModule.forRoot(routes) ],  
  exports: [ RouterModule ]  
})  
export class AppRoutingModule {}
```

Reference)

<https://angular.io/guide/router>

* 실습

/ 이면 홈화면

/jquery 이면 jquery 화면이 나오도록

app.module.ts,

AppComponent

IndexComponent

를 수정하자.

Index works

Home works or jquery works

* Material 디자인 적용

<https://material.angular.io>

npm install --save @angular/cdk @angular/material

Theme 적용

=> <https://material.angular.io/guide/theming>

* IndexComponent에 메뉴 적용

MatModule 적용

mat-toolbar, color='primary' 적용

* style.scss 수정

```
* {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
  word-break: break-all;  
}  
html, body {  
  margin: 0;  
  padding: 0;  
  height: 100%;  
  font-size: 14px;  
}  
@media screen and (max-width: 599px) {  
  html, body {  
    font-size: 12px;  
  }  
}
```

Reference)

<https://material.angular.io>

<https://material.angular.io/guide/theming> (테마적용)

AngularTodo

home works!

* 메뉴 버튼 적용

```
<button mat-icon-button [matMenuTriggerFor]="menu"> <mat-icon>menu</mat-icon> </button>
<mat-menu #menu="matMenu">
  <button mat-menu-item routerLink="/jquery">
    <mat-icon>input</mat-icon>
    <span>jquery</span>
  </button>
  <button mat-menu-item >
    <mat-icon>input</mat-icon>
    <span>angular</span>
  </button>
  <button mat-menu-item >
    <mat-icon>input</mat-icon>
    <span>http</span>
  </button>
  <button mat-menu-item >
    <mat-icon>input</mat-icon>
    <span>news</span>
  </button>
</mat-menu>
```

* 머티리얼 아이콘 적용

Index.html에 추가

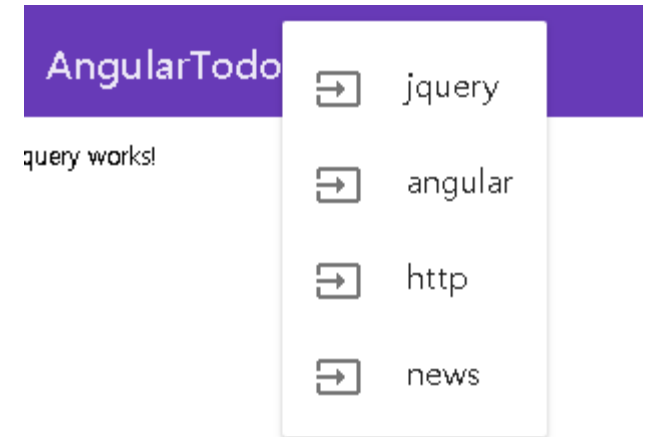
```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

아이콘 폰트 사이트

<https://material.io/icons/>

Reference)

<https://material.angular.io/components/menu/overview>



* 메뉴 아이콘을 오른쪽에 배치하기

라이브러리 설치

npm install --save @angular/flex-layout

⇒ <https://github.com/angular/flex-layout>

⇒ <https://tburleson-layouts-demos.firebaseio.com/#/docs>

* CSS3의 flex 속성 알아보기

- 부모(컨테이너) 속성

display: flex

flex-direction: row or column

justify-content: (direction 방향의 정렬)

align-items: (direction과 직각 방향 정렬

)

- 자식 속성:

flex-grow: (남은 공간을 배분하는 방법),

flex-shrink: (모자라는 공간을 배분하는 방법),

flex-basis: (넓이 지정 방법)

=> 3개를 한번에

flex: flex-grow, flex-shrink, flex-basis

* Flex 속성을 삽입하기

mat-toolbar 사이에 삽입

[Reference\)](#)

<https://github.com/angular/flex-layout>

<https://tburleson-layouts-demos.firebaseio.com/#/docs>

https://www.w3schools.com/css/css3_flexbox.asp



jquery works!

* MatCard 모듈 사용하기

```
<mat-card>
  <mat-card-header>
    <mat-card-title>My Todo's</mat-card-title>
  </mat-card-header>
  <mat-card-content>
    내용
  </mat-card-content>
</mat-card>
```

* IndexComponent에 body에 여백주기

```
<div class="container">
  <router-outlet> </router-outlet>
</div>
```

⇒ 스타일 지정하기

```
.container {
  margin: 0 auto;
  padding: 1rem;
  max-width: 1280px;
}
```

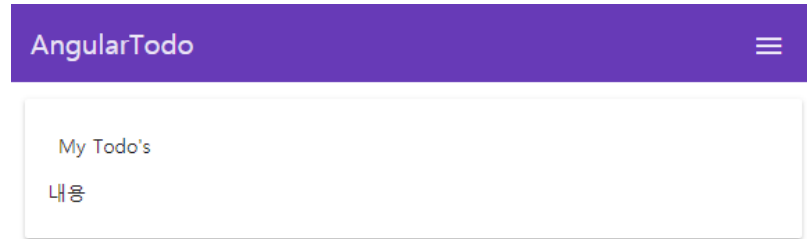
* mat-card 커스터마이징 하기

스타일을 전체적으로 적용하는 방법 + SCSS 구조 잡기

assets/styles 에 app.scss 추가

styles.scss에 import

=> @import 'assets/styles/app';



* mat-card 커스터마이징 하기

같은 폴더에 _material.scss, _variable.scss 추가후 app.scss에

```
@import "material";
```

```
@import "variable";
```

=> Q) _material.scss 파일 앞에 붙은 언더바는 무엇일까?

* _material.scss

```
$gutter: 1rem;
```

* _material.scss

```
.mat-card {
  padding: 0;
  margin: ($gutter/2);
  .mat-card-header {
    height: auto;
    background-color: #eeeeee;
  }
  .mat-card-title {
    padding-top: $gutter;
    padding-left: $gutter;
    padding-right: $gutter;
    line-height: 1;
    font-size: 1.3rem;
    font-weight: 400;
  }
}
```

```
.mat-card-content {
  padding: $gutter;
  margin-bottom: 0;
  position: relative;
}
}
```

Q) FHD 모바일에서 mat-card 의 마진은 몇 px인가?

Q) FHD 모바일에서 card-title의 font-size 는 몇 px인가?

AngularTodo



My Todo's

내용

* Todo 화면 구성

=> FlexLayoutModule 사용하기

```
<mat-card-content>
  <div fxLayout="row">
    <input type="text" placeholder="할 일" id="input_todo" fxFlex="4 1 auto">
    <button type="button" fxFlex="1 1 auto">추가</button>
  </div>

  <hr>

  <table class="table">
    <thead>
      <tr>
        <th>완료</th>
        <th>todo</th>
        <th>생성일</th>
        <th>수정일</th>
        <th>삭제</th>
      </tr>
    </thead>
    <tbody id="todo_list">
    </tbody>
  </table>
</mat-card-content>
```

* style.scss

```
table {
  background-color: transparent;
  border-collapse: collapse;
  border-spacing: 0;
}

.table {
  margin: 0;
  width: 100%;
  max-width: 100%;
  thead, tbody {
    tr {
      th, td {
        padding: 8px;
        line-height: 1.42857143;
        vertical-align: top;
        border-top: 1px solid #ddd;
      }
    }
  }
  thead {
    background-color: #dddddd;
  }
}
```

* Todo 화면 구성

⇒ Emmet 사용하기

⇒ <https://docs.emmet.io/>

tody에 tr과 td 4개를 emmet을 이용해서 한번에 입력하기

⇒ tr>td*4 다음에 탭 키

```
<tr>
  <td>
    <input type="checkbox" checked>
  </td>
  <td>빨래하기</td>
  <td>2017-10-11 18:11</td>
  <td>2017-10-11 18:11</td>
  <td>
    <button type="button">삭제</button>
  </td>
</tr>
```

=> 이 부분은 동적으로 생성되는 부분이다. 눈으로 확인만 하자.

AngularTodo



My Todo's

할 일

추가

완료

todo

생성일

수정일

삭제



빨래하기

2017-10-11 18:11

2017-10-11 18:11

삭제

* Jquery 추가

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
```

* onclick="add_todo()" 추가

* Script 작성 : index.html<script>
var todoList = [];

```
function add_todo() {
    var todo = {
        isFinished: false,
        todo: $('#input_todo').val(),
        created: new Date(),
        updated: new Date()
    };

    todoList.push(todo);

    refresh();
}
```

```
function refresh() {
    $('#todo_list').empty();

    todoList.forEach(function(item, index) {
        var todo =
            '<tr>' +
            '<td>' +
            '<input type="checkbox" ' + (item.isFinished?'checked:') + ' value="" + item.isFinished +
            '" onchange="update(' + index + ')">' +
            '</td>' +
            (item.isFinished?'<td style="text-decoration: line-through">':'<td>') + item.todo + '</td>' +
            '<td>' + item.created + '</td>' +
            '<td>' + item.updated + '</td>' +
            '<td>' +
            '<button type="button" onclick="remove(' + index + ')">삭제</button>' +
            '</td>' +
            '</tr>';
        $('#todo_list').append(todo);
    });
}
```

Q) 할일추가시 맨 나중에 추가한것이 맨 위로 올라가게 하기.

* 웹페이지가 로딩될때 ajax로 서버에서 데이터를 가져와서 갱신한다.

```
$(document).ready(function () {  
    console.log('docunet ready');  
    getTodoList();  
});  
  
function getTodoList() {  
    $.ajax({  
        url: 'http://www.javabrain.kr:8080/api/todo',  
        method: 'GET',  
        datatype: 'json',  
        success: function(data) {  
            console.log(data);  
            todoList = data;  
            refresh();  
        }  
    });  
}
```

Q) REST api를 postman 으로 테스트 하시오.

Q) datatype: 'json' 은 무엇을 얘기하는가?

Q) 이 유알엘은 CORS 문제를 야기한다. 왜?

* 서버에 연동하여 할일을 추가한다.

```
function add_todo() {  
    $.ajax({  
        url: 'http://www.javabrain.kr:8080/api/todo',  
        method: 'POST',  
        data: JSON.stringify({todo: $('#input_todo').val(), isFinished: false}),  
        contentType: 'application/json',  
        datatype: 'json',  
        success: function(data) {  
            console.log(data);  
            getTodoList();  
        }  
    });  
}
```

Q) REST api를 postman 으로 테스트 하시오.

Q) 보내는 데이터 타입은 무엇인가?

* 체크박스 클릭시 서버에 업데이트 구현

```
function update(index) {  
    var updateTodo = {  
        todo_id: todoList[index].todo_id,  
        isFinished: !todoList[index].isFinished,  
        todo: todoList[index].todo  
    };  
  
    $.ajax({  
        url: 'http://www.javabrain.kr:8080/api/todo',  
        method: 'PUT',  
        data: JSON.stringify(updateTodo),  
        contentType: 'application/json',  
        datatype: 'json',  
        success: function(data) {  
            console.log(data);  
            getTodoList();  
        }  
    });  
  
    refresh();  
}
```

Q) REST api를 postman 으로 테스트 하시오.

* 삭제 버튼 클릭시 삭제

```
function remove(index) {  
  
    $.ajax({  
        url: 'http://www.javabrain.kr:8080/api/todo?todo_id=' + todoList[index].todo_id,  
        method: 'DELETE',  
        datatype: 'json',  
        success: function(data) {  
            console.log(data);  
            getTodoList();  
        }  
    });  
}
```

Q) REST api를 postman 으로 테스트 하시오.

Q) DELETE 메서드와 다른 메서드의 차이점은 무엇인가?

* Button에 class="active" 추가

```
$buttonColor: #337ab7;
$buttonDark: darken($buttonColor, 10%);
$buttonDarker: darken($buttonDark, 10%);

button.active {
  background-color: $buttonColor;
  border: 0;
  padding: 0.5rem;
  color: white;
  box-shadow: 4px 4px 6px $buttonDark;

  &:hover {
    background: $buttonDark;
    box-shadow: 4px 4px 6px $buttonDarker;
  }
  &:active {
    box-shadow: none;
    -webkit-transform: translate(4px, 4px);
    -moz-transform: translate(4px, 4px);
    -ms-transform: translate(4px, 4px);
    -o-transform: translate(4px, 4px);
    transform: translate(4px, 4px);
  }
}
```

reference:

<https://velopert.com/1712>

<http://sass-lang.com/guide>

SCSS 8가지 특징

- 1) 변수 사용
- 2) 수학 연산자
- 3) 내장함수
- 4) 중첩
- 5) Import
- 6) Extend
- 7) Mixin
- 8) Function

Q) 브라우저별로 중복적으로 지정하는 transform 속성 5가지를 sass를 사용해서 한줄로 작성하시오.

(@mixin, @include 사용)

실습

- * angular 메뉴 추가
 - ⇒ angular-cli 로 메뉴 생성
 - ⇒ 라우팅 설정
 - ⇒ 상단에 메뉴 생성하기

* 타입 정의하기

=> TodoVO 객체 정의

```
export class TodoVO {  
  todo_id: number;  
  isFinished: boolean;  
  todo: string;  
  created: string;  
  updated: string;  
}
```

* 개발순서

- 1) REST API 테스트
- 2) 객체 정의
- 3) 서비스 구현
- 4) 서비스 호출 및 Model 생성
- 5) Html 바인딩

완료	todo	생성일	수정일	삭제
----	------	-----	-----	----

* 서비스 정의

Http를 통해서 CRUD를 수행할 서비스를 만든다. HttpClientModule 추가

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { environment } from '../environments/environment';
import { TodoVO } from '../domain/todo.vo';
```

```
@Injectable()
```

```
export class UserService {
```

```
  private SERVER: string;
```

```
  private headers: HttpHeaders;
```

```
  constructor(private http: HttpClient) {
```

```
    this.SERVER = `${environment.HOST}`;
```

```
    this.headers = new HttpHeaders({
      'Content-Type': 'application/json'
    });
```

```
  }
```

```
}
```

```
  addTodo(params: TodoVO) {
```

```
    return this.http.post(this.SERVER + '/api/todo', JSON.stringify(params), {headers: this.headers}).toPromise();
```

```
  }
```

```
  getTodoList() {
```

```
    return this.http.get(this.SERVER + '/api/todo').toPromise();
```

```
  }
```

```
}
```

Reference)

<https://angular.io/guide/http>

* 인스턴스 멤버 변수 정의

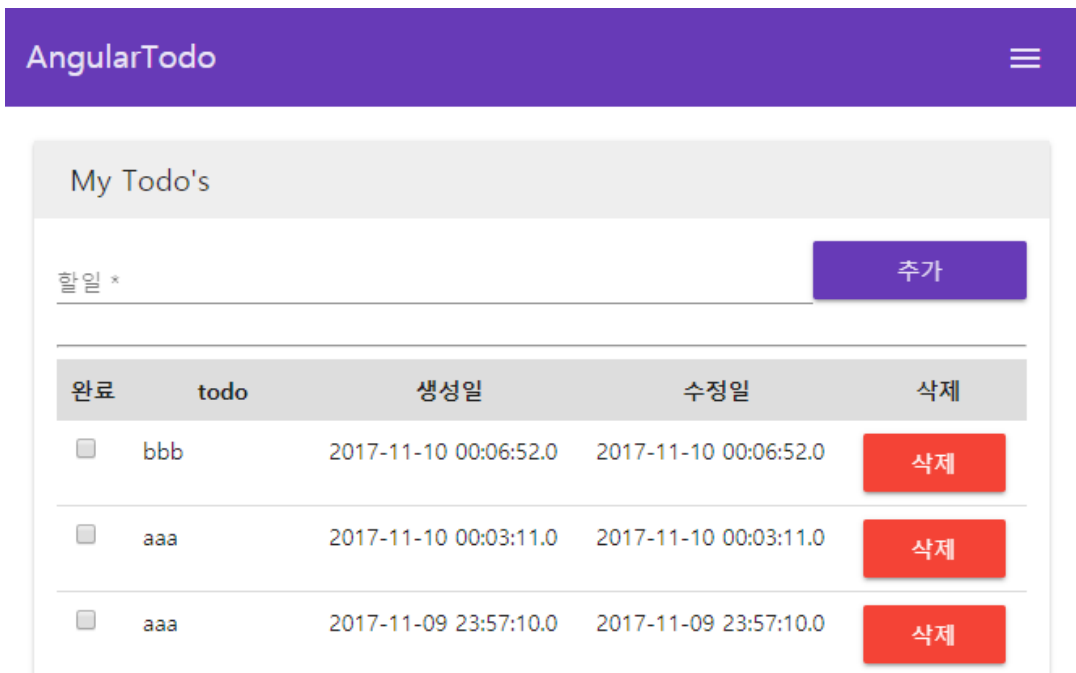
```
todoList = new Array<TodoVO>();
```

* ngOnInit()에서 해당 함수 호출

```
getTodoList() {
  console.log('getTodoList');
  this.userService.getTodoList()
    .then(data => {
      this.todoList = data;
    });
}
```

* tbody 구현

```
<tr *ngFor="let item of todoList">
  <td>
    <input type="checkbox" (change)="update(item)">
  </td>
  <td [class.todo_canceled]="item.isFinished">{{item.todo}}</td>
  <td>{{item.created}}</td>
  <td>{{item.updated}}</td>
  <td>
    <button mat-raised-button color="warn">삭제</button>
  </td>
</tr>
```



* 인스턴스 멤버 변수 정의
newTodo = new TodoVO();

* add_todo 함수 구현

```
add_todo() {  
  console.log('click');  
  
  this.userService.addToDo(this.newTodo)  
    .then((data: TodoVO) => {  
      console.log(data);  
      this.todoList.unshift(data);  
    });  
  
  this.newTodo = new TodoVO();  
}
```

화면 만들기

* Input box 머티리얼 적용

FormsModule, MatFormFieldModule, MatInputModule 추가

```
<mat-form-field fxFlex="4 1 auto">
  <input matInput placeholder="할일" [(ngModel)]="todo" >
</mat-form-field>
```

* 버튼에 머티리얼 적용

```
<button mat-raised-button color="primary" (click)="add_todo()" fxFlex="1 1 auto">추가</button>
⇒ Q) 버튼의 높이가 커진다 이유는 무엇이고 어떻게 수정해야 하나? (flex 정렬)
```

* Html5 Form API를 이용하여 값이 없이 입력 되는 것을 방지하기

```
<form ngNativeValidate #myForm="ngForm" (submit)="add_todo()">
  <div fxLayout="row" fxLayoutAlign="center start">
    <mat-form-field fxFlex="4 1 auto">
      <input matInput placeholder="할일" [(ngModel)]="todo" name="todo" required>
    </mat-form-field>
    <button type="submit" mat-raised-button color="primary" fxFlex="1 1 auto">추가</button>
  </div>
</form>
```

* Html5 폼 API,
 * validation 방법 알기
 11가지 validity 속성 이해하기
 예) required 속성 => valueMissing
 Type 속성 => typeMismatch
 Pattern 속성 => PatternMismatch
 ...
 10가지가 모두 false 일때 valid가 true

AngularTodo

My Todo's

할일 *

추가

완료

todo

생성일


수정일

삭제

* Angular Form validation 방식

```
<form #myForm="ngForm" (ngSubmit)="add_todo()">
  <div fxLayout="row" fxLayoutAlign="center start">
    <mat-form-field fxFlex="4 1 auto">
      <input matInput placeholder="할일" [(ngModel)]="newTodo.todo" name="todo"
#name="ngModel" required>
      <mat-hint align="start" style="color: red;">{{!name.value? '할일을 입력하세요' : ''}}</mat-hint>
    </mat-form-field>
    <button type="submit" mat-raised-button color="primary" fxFlex="1 1 auto">추가</button>
  </div>
</form>
```

* angular 폼 유효성 체크 방법

AngularTodo 

My Todo's

할일 *

추가

완료

todo

생성일

수정일

삭제

* BrowserModule 이 추가로 필요하다.

* Trigger 바로 아래에 state는 상태유지가 필요할때 정의
Transition의 style은 상태 유지가 안된다.

=> 추가시 왼쪽에서, 삭제이 오른쪽으로 이동하는 애니메이션을 작성하자.

```
animations: [  
  trigger('flyInOut', [  
    state('In', style({transform: 'translate(0, 0)'})),  
    transition('void => *', [  
      style({transform: 'translate(-100%, 0)'}),  
      animate(300)  
    ]),  
    transition('* => void', [  
      animate(300, style({transform: 'translate(0, -100%)', opacity: '0'}))  
    ])  
  ])  
]
```

* Html에 애니메이션을 적용

```
<tr *ngFor="let item of todoList;" [@flyInOut] = "in">
```

Reference)

<https://angular.io/guide/animations>

Q) Web Animations는 standar인가?

<https://drafts.csswg.org/web-animations-1/>

* 삭제시 keyframe을 이용해보자

```
transition('in => void', [  
  animate(300, keyframes([  
    style({opacity: 1, transform: 'translateX(0)', offset: 0}),  
    style({opacity: 1, transform: 'translateX(-50px)', offset: 0.7}),  
    style({opacity: 0, transform: 'translateX(100%)', offset: 1.0})  
  ]))  
]),
```

* 수정버튼시 선택된 영역을 아래위로 늘리는 애니메이션을 적용하자.

```
state('in', style({opacity: 1, transform: 'translate(0, 0)'})),  
state('active', style({opacity: 1, transform: 'scale(1, 1.3)'})),  
transition('void => in', [  
  style({opacity: 0, transform: 'translate(-100%, 0)'}),  
  animate(300)  
]),  
transition('in => void', [  
  animate(300, keyframes([  
    style({opacity: 1, transform: 'translateX(0)', offset: 0}),  
    style({opacity: 1, transform: 'translateX(-50px)', offset: 0.7}),  
    style({opacity: 0, transform: 'translateX(100%)', offset: 1.0})  
  ]))  
]),  
// 선택시 애니메이션  
transition('void => active', [  
  animate(600, keyframes([  
    style({transform: 'scale(1, 1)', offset: 0}),  
    style({transform: 'scale(1, 1)', offset: 0.5}),  
    style({transform: 'scale(1, 1.3)', offset: 1.0})  
  ]))  
]),  
])
```

* CSS Animation

참조) <https://css-tricks.com/almanac/properties/t/transform/>

```
.element {  
  animation-name: stretch;  
  animation-duration: 1.5s;  
  animation-timing-function: ease-out;  
  animation-delay: 0s;  
  animation-direction: alternate;  
  animation-iteration-count: infinite;  
  animation-fill-mode: none;  
  animation-play-state: running;  
}
```

```
/*  
  is the same as:  
*/
```

```
.element {  
  animation:  
    stretch  
    1.5s  
    ease-out  
    0s  
    alternate  
    infinite  
    none  
    running;  
}
```

* 단일 키프레임

```
@keyframes example {  
  
  from {background-color: red;}  
  
  to {background-color: yellow;}  
  
}
```

* 다중 키프레임

```
@keyframes example {  
  0%   {background-color:red; left:0px; top:0px;}  
  25%  {background-color:yellow; left:200px; top:0px;}  
  50%  {background-color:blue; left:200px; top:200px;}  
  75%  {background-color:green; left:0px; top:200px;}  
  100% {background-color:red; left:0px; top:0px;}  
}
```

* Html 템플릿 변경

```
<ng-template ngFor let-item [ngForOf]="todoList">
  <tr *ngIf="!item.isEdited" [@flyInOut] = "'in'">
    <td>
      {{item.isFinished ? '완료' : '미완료'}}
    </td>
    <td [class.todo_canceled]="item.isFinished">{{item.todo}}</td>
    <td>{{item.created}}</td>
    <td>{{item.updated}}</td>
    <td>
      <button mat-raised-button color="accent" (click)="save(item)">수정 </button>
      <button mat-raised-button color="warn" (click)="remove(item)">삭제 </button>
    </td>
  </tr>
  <tr *ngIf="item.isEdited">
    <td>
      <input type="checkbox" [(ngModel)]="item.isFinished">
    </td>
    <td [class.todo_canceled]="item.isFinished">
      <input [(ngModel)]="item.todo">
    </td>
    <td>{{item.created}}</td>
    <td>{{item.updated}}</td>
    <td>
      <button mat-raised-button color="accent" (click)="modify(item)">저장 </button>
      <button mat-raised-button color="warn" (click)="restore(item)">취소 </button>
    </td>
  </tr>
</ng-template>
```

Reference)

=><https://angular.io/guide/template-syntax>

* todo도 수정 가능하게 한다.

수정 버튼을 누르면 수정, 삭제 버튼이 저장 취소 버튼으로 바뀐다.

취소를 누르면 기존 데이터로 돌아가고 저장을 누르면 update한다.

AngularTodo

My Todo's

추가

완료	todo	생성일	수정일	삭제
미완료	asdf	2017-11-10 01:06:36.0	2017-11-10 01:06:36.0	<div>수정</div> <div>삭제</div>
완료	kkk	2017-11-10 00:38:16.0	2017-11-10 01:02:45.0	<div>수정</div> <div>삭제</div>
미완료	빨래하기	2017-11-08 23:18:00.0	2017-11-08 23:18:00.0	<div>수정</div> <div>삭제</div>

AngularTodo

My Todo's

추가

완료	todo	생성일	수정일	삭제
<input type="checkbox"/>	<input type="text"/> asdf	2017-11-10 01:06:36.0	2017-11-10 01:06:36.0	<div>저장</div> <div>취소</div>
완료	kkk	2017-11-10 00:38:16.0	2017-11-10 01:02:45.0	<div>수정</div> <div>삭제</div>
<input type="checkbox"/>	<input type="text"/> 빨래하기	2017-11-08 23:18:00.0	2017-11-08 23:18:00.0	<div>저장</div> <div>취소</div>

* 인스턴스 멤버 추가

```
// 취소시 복원하기 위한 데이터를 저장하는 컬렉션 : number 에는 todo_id 저장
tempTodoList: Map<number, TodoVO> = new Map<number, TodoVO>();
```

* 함수 구현

```
modify(item: TodoVO) {
  this.userService.modifyTodo(item)
    .then((data: TodoVO) => {
      item.isFinished = data.isFinished;
      item.todo = data.todo;
      // 에디터 상태 복원
      item.isEdited = false;
    });
}
```

```
restore(todoVo: TodoVO) {
  todoVo.isEdited = false;
```

```
  let tempTodo = this.tempTodoList.get(todoVo.todo_id);
  todoVo.isFinished = tempTodo.isFinished;
  todoVo.todo = tempTodo.todo;
}
```

```
save(todoVo: TodoVO) {
  todoVo.isEdited = true;

  let tempTodo = new TodoVO();
  tempTodo.isFinished = todoVo.isFinished;
  tempTodo.todo = todoVo.todo;
  this.tempTodoList.set(todoVo.todo_id, tempTodo);
}
```

* 함수 구현

=> Alert 창을 띄우고 확인시 삭제한다. 향후에 모달창으로 구현해보자.

```
remove(todoVo: TodoVO) {  
  const result = confirm(todoVo.todo + '을(를) 삭제하시겠습니까?');  
  if (result) {  
    this.userService.removeTodo(todoVo.todo_id)  
      .then(res => {  
        if (res.result === 0) {  
          this.todoList.forEach((item, index) => {  
            if (item.todo_id === todoVo.todo_id) {  
              this.todoList.splice(index, 1);  
            }  
          });  
        }  
      });  
  }  
}
```

ng g directive highlight 로 생성

```
@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {

  constructor(private el: ElementRef) {
    // el.nativeElement.style.backgroundColor = 'yellow';
  }

  @Input() highlightColor: string;

  @HostListener('mouseenter') onMouseEnter() {
    this.highlight(this.highlightColor || 'red');
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.highlight(null);
  }

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

Reference)

<https://angular.io/guide/attribute-directives>

1. 테이블 tr 컬럼 색깔을 노란색으로 칠하기
2. 입력 색깔을 받아서 칠하기
3. 마우스오버시에만 색깔 칠하기
4. Javascript로 색깔 칠하기

* 생성하기

ng g pipe mydate

```
@Pipe({
  name: 'myDate'
})
export class MyDatePipe implements PipeTransform {

  transform(value: string): string {
    const date = new Date();
    return date.getFullYear() + "-" + (this.addZero(date.getMonth() + 1)) + "-" +
    this.addZero(date.getDate()) + " "
      + this.addZero(date.getHours()) + ":" + this.addZero(date.getMinutes()) + ":"

  addZero(digit: number): string {
    // digit 가 문자가 아니라 숫자이다 digit.length로는 안됨.
    if (digit < 10) {
      return "0" + digit;
    } else {
      return "" + digit;
    }
  }
}
```

* reference)

<https://angular.io/guide/pipes>

1. 1999-12-21 12:42 형태로 날짜 바꾸기

2. Angular 에 내장된 파이프 사용해보기

=> <https://angular.io/api/common/DatePipe>