Spickzettel („Cheat Sheet“) Angular mit TypeScript

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de)

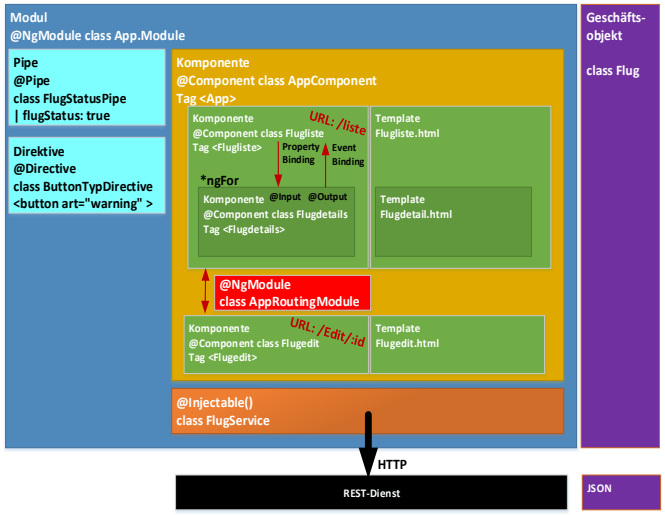
www.IT-Visions.de
Dr. Holger Schwichtenberg

Architektur

Aufbau des hier verwendeten Beispiels

Quellcode dazu unter http://www.it-visions.de/p9117

Angular-Website: http://www.angular.io



Importe

Wichtige Angular-Typen

```
import { NgModule, Component, Input, Output, EventEmitter, OnInit,
    ChangeDetectionStrategy, Injectable, Pipe, PipeTransform }
    from '@angular/core';
import { platformBrowserDynamic }
    from '@angular/platform-browser-dynamic';
import { LocationStrategy, HashLocationStrategy}
   from '@angular/common';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { Router, ActivatedRoute, Params } from '@angular/router';
import { HttpClientModule } from '@angular/http';
```

Andere Frameworks

```
import { Observable } from 'rxjs/Rx'; // Reactive Extensions
import * as _ from 'underscore'; // underscore (~LINQ)
```

Eigene Komponenten, Direktiven, Pipes, Services

```
import { Flug } from '../Modell/Flug' // Geschäftsobjekt (Modell)
import { FlugService } from '../Services/FlugService' // Dienst
import { Flugdetail } from '../Flugdetail/Flugdetail'; // Komponente
```

Startcode (main.ts)

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

app.module.ts

```
@NgModule({
 imports: [BrowserModule, FormsModule, HttpClientModule,
   AppRoutingModule ], // Angular-Module
 declarations: [AppComponent, Flugliste, Flugdetail, FlugEdit,
   FlugStatusPipe, ButtonTypDirective],  // Komponenten und Direktiven
 bootstrap: [AppComponent], // Startkomponente
 providers: [  // Dependency Injection
  [FlugService],
  { provide: LocationStrategy, useClass: HashLocationStrategy } ]
 })
export class AppModule { }
```

Komponente FlugListe.ts mit OnInit()

```
@Component({
 selector: 'Flugliste',
 templateUrl: 'App/Flugliste/Flugliste.html',
 styleUrls: ['App/Flugliste/Flugliste.css'],
 providers: [FlugService], // Registrierung eines Dienstes für Injektion
})
export class Flugliste implements OnInit {

 constructor(private flugService: FlugService,  private router: Router)
// hier wird FlugService und Router injiziert (DI)  { }

status: string;
flugSet: Array<Flug>;

ngOnInit() { this. DatenLaden();  }

async DatenLaden() {
  this.flugSet = await this.flugdatenProxy.getFluege("Essen").toPromise();
  this.status = this.flugSet.length + " Flüge geladen.";
}

Loeschen(flug: Flug) {
  this.flugService.delete(flug); // Dienst aufrufen
  this.status = `Flug ${flug.FlugNr} gelöscht!`;
 }

Aendern(flug: Flug) {
  var link = ['/edit', flug.FlugNr];
  this.router.navigate(link); // Ansicht aufrufen
 }

// Event-Handler für Nachricht von untergeordneter Komponente, wenn
    diese einen Flug gelöscht hat
onFlugGeloescht(flug: Flug) {
  this.status = `Flug ${flug.FlugNr} gelöscht!`;
  this.DatenLaden();
 }
}
```

Template Flugliste.htm

```
<ul><li *ngFor="let f of flugSet ; let istGerade = even; let i = index">
  <span [ngClass]="{'text-primary': istGerade, 'text-info': !istGerade}">
  <span class="badge">{{i+1}}:
    Flug #{{f.FlugNr | number:'3.0' }}</span>
    {{ f.FreiePlaetze | flugStatus:true }}
  <Flugdetail [flug]="f"
    (flugGeloeschtEvent)="onFlugGeloescht($event)"></Flugdetail>
  <button *ngIf="f.FreiePlaetze>0"
    (click)="Aendern(f)">Ändern</button>
  <button [disabled]="f.FreiePlaetze<=0 ||f.AbflugOrt=='Essen'"
    (click)="Loeschen(f)">Löschen</button>
  </span>
 </li></ul>
```

Untergeordnete Komponente Flugdetail.ts

```
@Component({
 selector: 'Flugdetail',
 templateUrl: 'App/Flugdetail/Flugdetail.html'
})
export class Flugdetail implements OnInit {

@Input() // eingehende Daten
flug: Flug;

@Output() // ausgehende Ereignisse
flugGeloeschtEvent = new EventEmitter<Flug>();

constructor(private flugService: FlugService, private router: Router) { }

Loeschen() {
  this.flugService.delete(this.flug); // Dienst aufrufen
  this.flugGeloeschtEvent.emit(this.flug); // Ereignis auslösen
 }
}
```

Strukturelle Direktiven

Schleife

```
<li *ngFor="let f of flugSet;let istGerade=even; let i=index"
[ngClass]="{'text-primary': istGerade, 'text-info': !istGerade}">
{{i | number:'3.0-0' }}: {{f.FlugNr}}</li>
```

Bedingungen

```
<div *ngIf="flug.FreiePlaetze<0">Dieser Flug ist überbucht!</div>
<ng-template #keineFluege>
   <p>keine Flüge gefunden</p>
 </ng-template>
 <p *ngIf="flugSet; else keineFluege">{{flugSet.length}} Flüge</p>
 <div [ngSwitch]="flug.FreiePlaetze">
   <span *ngSwitchCase="0">ausgebucht</span>
   <span *ngSwitchCase="1">noch ein Platz verfügbar</span>
   <span *ngSwitchDefault>mehrere Plätze verfügbar</span>
 </div>
```

Spickzettel („Cheat Sheet“) Angular mit TypeScript

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de)

www.IT-Visions.de
Dr. Holger Schwichtenberg

Datenbindungssyntax

Statischer Text (Interpolation), mit Pipe

```
Flug {{flug.FlugNr}}: {{flug.FreiePlaetze | number:'3.0-0'}}
```

Ausgabe von HTML-Tags

```
<div [innerHTML]="flug.HtmlMemo"></div>
```

Ausdrücke

```
72 + 8 ist {{72+8}}
Flugnummer, wenn vorhanden {{flug?.FlugNr}}
```

Text in Eigenschaft

```
<div title="Flug #{{ flug.FlugNr }}"> oder
<div [title]="'Flug #' + flug.FlugNr">
```

Ein-Wege-Bindung an Eigenschaft

```
<button [disabled]="flug.FreiePlaetze<=0 || flug.AbflugOrt=='Essen'"
...">Buchen</button>
<img src="balken.png" style="height:40px" [style.width.px]=
"flug.FreiePlaetze" [hidden]= "flug.FreiePlaetze===0">
```

Zwei-Wege-Bindung an Eigenschaft

```
<input [(ngModel)]="flug.FreiePlaetze" name="FreiePlaetze">
```

CSS-Klassenbindung mit Bedingungen

```
<div [ngClass]="{'text-danger': flug.FreiePlaetze<3,
 'text-warning': flug.FreiePlaetze>=3,
 'text-success': flug.FreiePlaetze==0}">
 {{flug.FreiePlaetze}} freie Plätze</div>
```

Ereignis-Bindung

```
<button (click)="Loeschen(flug)">Löschen</button>
```

Lokale Variable

```
<input #freiePlaetze placeholder="anzahl" value="123">
<br>Sie haben eingegeben: {{freiePlaetze.value}}
```

Pipes

Eingebaute Pipes

date, number, slice, uppercase, lowercase, currency, percent

Verkettung von Pipes

```
{{flug.Datum | date:"EEE dd.MM.yy" | uppercase }}
```

Eigene Pipe (wird verwendet in Flugliste.htm)

```
@Pipe({name: 'flugStatus', pure: true})
export class FlugStatusPipe implements PipeTransform {
  transform(value: number, kurz: boolean): string {
    var text = "";
    if (!kurz) text = "Flug ist "
    switch(value)   {
    case 0: return text + "ausgebucht"; break;
    default: return text + "verfügbar";
    } }
}
```

Formular mit Validierung (FlugEdit.html)

```
<form (ngSubmit)="onSubmit(flugForm)" #flugForm="ngForm">
 <!--Auswahlfeld für Abflugort-->
 <div class="form-group">
   <label for="Abflugort">Abflugort</label>
   <select id="Abflugort" required
     [(ngModel)]="flug.Abflugort" name="Abflugort"
     (change)="ortGeaendert()" class="form-control" >
    <option *ngFor="let o of orte" [value]="o" >{{o}}</option>
   </select> </div>
 <!--Eingabefeld für Zielort-->
 <div class="form-group">
   <label for="Zielort">Zielort</label>
   <input type="text" id="Zielort" name="Zielort"
     [(ngModel)]="flug.Zielort" (keyup)="tasteGedrueckt($event)"
     required minlength="3" maxlength="20" pattern="[a-zA-Z- ]*"
     class="form-control"> </div>
<!--Validatorausgabe für Zielort-->
<div [hidden]="Zielort.valid || Zielort.pristine" class="alert alert-
danger">Zielort muss Text sein, 3 bis 20 Zeichen!</div>
<!--Speichern-Button-->
<button type="button" class="btn btn-default" (click)="save(flugForm)"
[disabled]="! flugForm.form.valid">Speichern</button>
</div> </form>
```

In FlugEdit.ts:

```
save(form: NgForm) {  if (form.invalid) return;   // erneute Prüfung
    … }
```

Router

app-routing.module.ts

```
const routes: Routes = [
  { path: '', redirectTo: 'liste', pathMatch: 'full'
  { path: 'liste',  component: Flugliste },
  { path: 'edit', component: FlugEdit },
  { path: 'edit/:id', component: FlugEdit }
];
@NgModule({
  imports: [ RouterModule.forRoot(routes) ],
  exports: [ RouterModule ]
})
export class AppRoutingModule {}
```

app.component.html

```
<nav>
   <a routerLink="/liste" btyp="success">Liste</a>
   <a routerLink="/edit" btyp="warning"
       routerLinkActive="active">Edit</a>
 </nav>

<router-outlet></router-outlet>
```

HTTP-Client: AJAX-Aufruf eines REST-API-Dienstes

```
@Injectable() export class FlugService  {
 constructor(private http: HttpClient) { }
 getFluge(ort: string): Observable<Flug[]> {
  return this.http.get("http://server/api/flugdaten?ort=" + ort)
   .map(resp => { console.log("Geladen!"); return resp() })
   .catch((error: any) => {
    console.log("FlugService-Fehler", error);
    return Observable.throw(error.json().error || 'Server error');
   }) } }
```

Attribut-Direktive für <... btyp="warning">

Diese Direktive wird verwendet in app.component.html:

```
@Directive({ selector: '[btyp]' })
export class ButtonTypDirective implements OnInit {
 @Input('btyp') btyp: string; // Wert des Erweiterungsattributs typ
 constructor(private el: ElementRef, private renderer: Renderer) {  }
 ngOnInit() {
  this.renderer.setElementClass(this.el.nativeElement, 'btn', true);
  this.renderer.setElementClass(this.el.nativeElement, 'btn-' + (this.btyp
     || "default"), true);
  this.renderer.setElementStyle(this.el.nativeElement, 'color', 'black');
 }
 @HostListener('mouseenter') onMouseEnter() {
  this.renderer.setElementStyle(this.el.nativeElement, 'color', 'red');
 }
 @HostListener('mouseleave') onMouseLeave() {
  this.renderer.setElementStyle(this.el.nativeElement, 'color', 'black');
 }
}
```

Angular CLI (https://cli.angular.io)

```
ng new FluggesellschaftApp // Grundgerüst anlegen
ng generate component Flugdetail // Komponente anlegen
ng serve // Übersetzen und Webserver starten
npm install --save jquery // Paket installieren
npm install --save-dev @types/jquery // Paket für Entwicklungszeit
ng build --prod // Produktionsbuild
ng eject  // Ausgabe der Webpack-Konfiguration
```

Über den Autor

Dr. Holger Schwichtenberg gehört zu den bekanntesten Experten für Webtechniken und .NET in Deutschland. Er hat zahlreiche Fachbücher veröffentlicht und spricht regelmäßig auf Fachkonferenzen. Sie können ihn und seine Kollegen für Entwicklungsarbeiten, Schulungen, Beratungen und Coaching buchen.
E-Mail: **anfragen@IT-Visions.de**
Website: **www.IT-Visions.de**
Weblog: **www.dotnet-doktor.de**