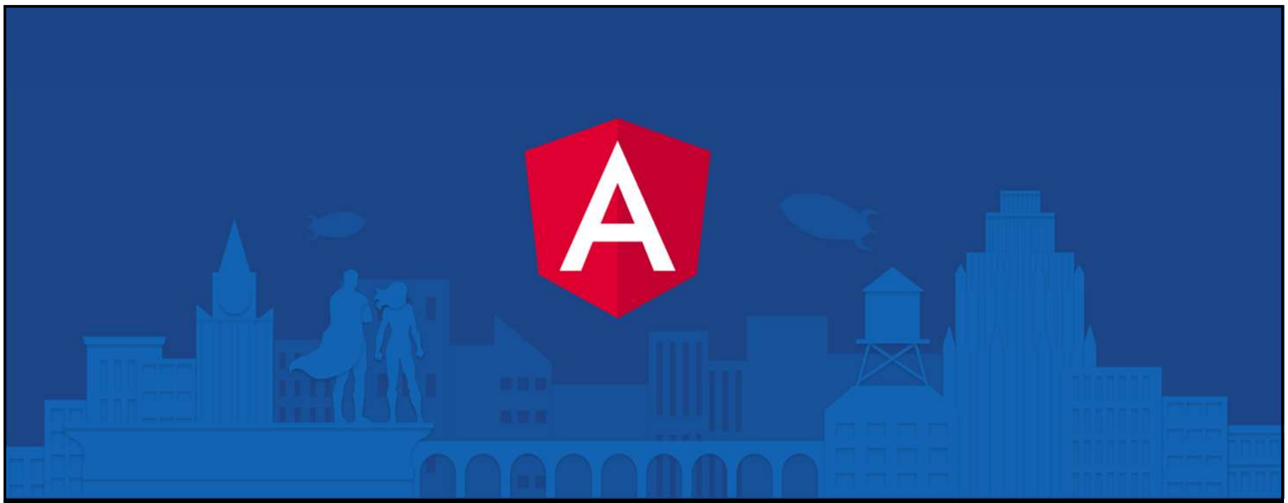


Formulare und Validierung

Michael Zikes
SOFTWAREarchitekt.at

Inhalt

- Ansätze
- Template-getriebene Formulare
- Reaktive Formulare
- Validierung

SOFTWAREarchitekt.at

Ansätze in Angular

Template-getrieben
- ngModel im Template
- Angular erzeugt Objektgraph für Formular
- FormsModule

Reaktiv
- Anwendung erzeugt Objektgraph
- Mehr Kontrolle
- ReactiveFormsModule

Daten-getrieben
- Angular generiert Formular für Datenmodell
- An Community übergeben

SOFTWAREarchitekt.at

Template-getriebene Formulare

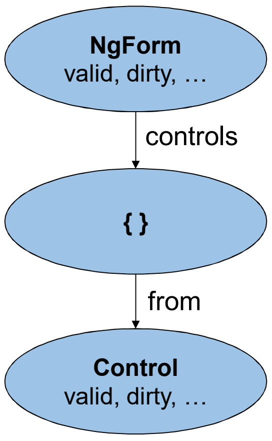
Template-getriebene Formulare

```
export class FlightSearchComponent {

  from: string;
  to: string;

  constructor(flightService: FlightService) {

      from = 'Graz';
      to = 'Hamburg';
  }
}
```

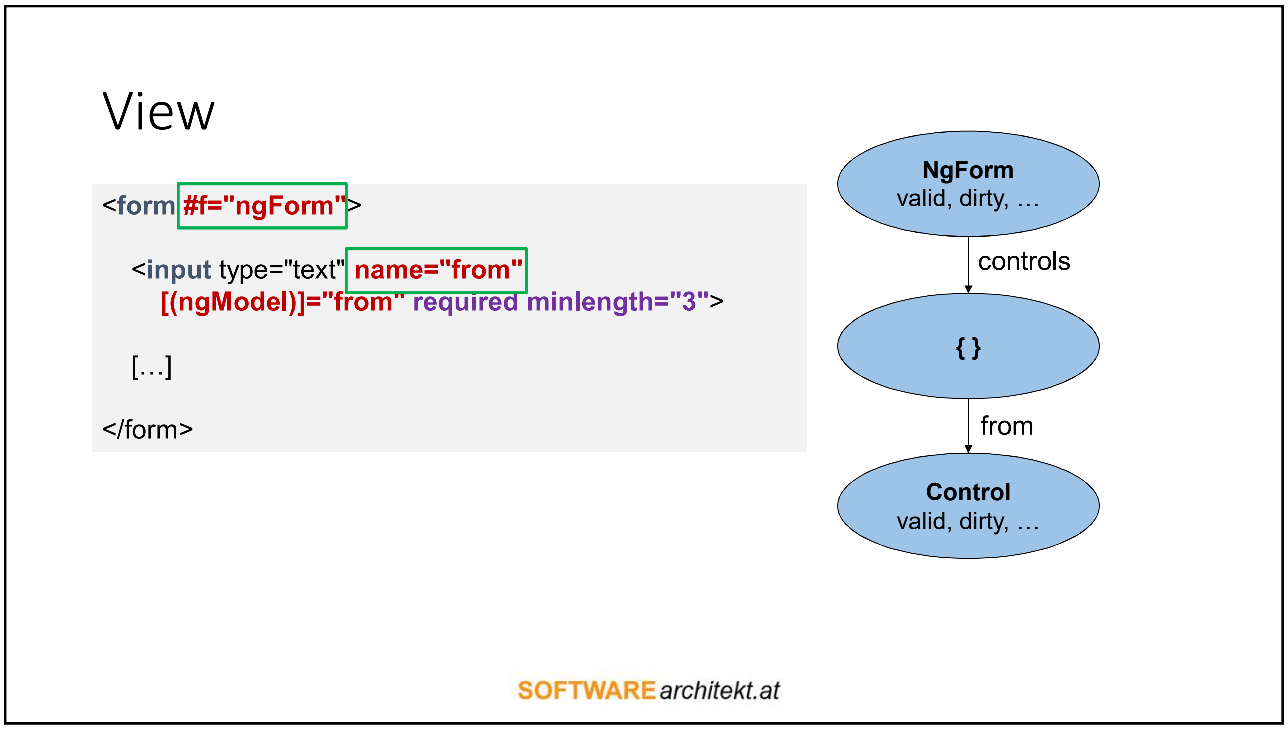
SOFTWARE architekt.at

View

```
<form>

  <input type="text" name="from"
    [(ngModel)]="from" required minlength="3">

  [...]

</form>
```

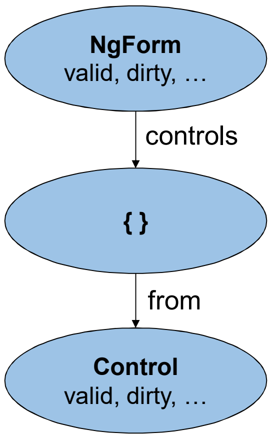


SOFTWARE architekt.at

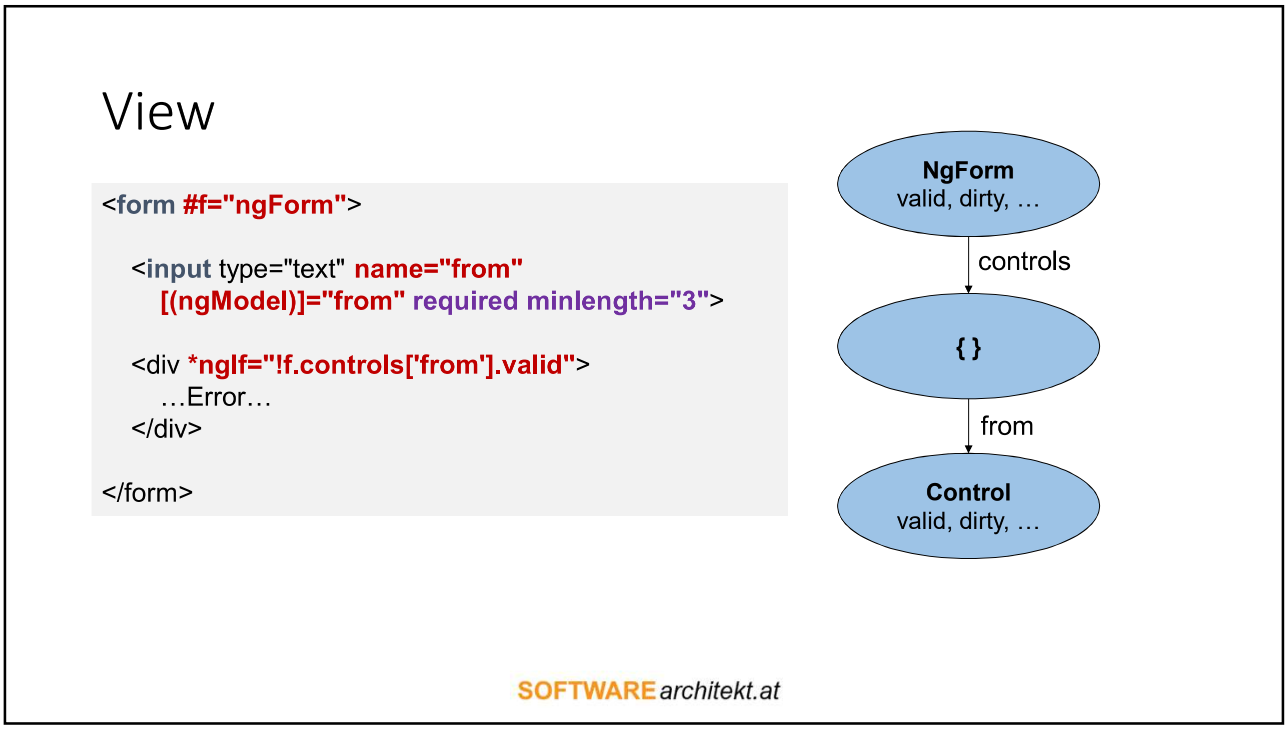


View

```
<form #f="ngForm">

   <input type="text" name="from"
      [(ngModel)]="from" required minlength="3">

   […]

</form>
```

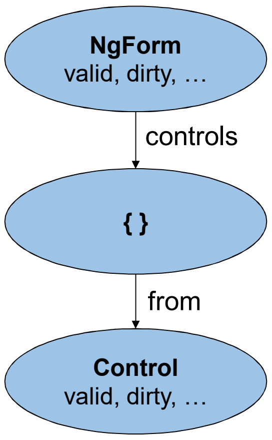


SOFTWAREarchitekt.at



View

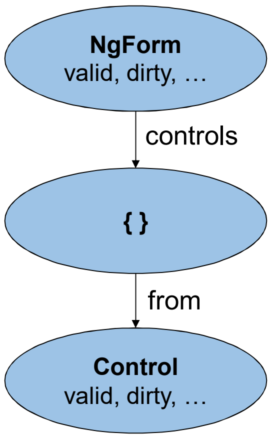
```
<form #f="ngForm">

   <input type="text" name="from"
      [(ngModel)]="from" required minlength="3">

   <div *ngIf="!f.controls['from'].valid">
      …Error…
   </div>

</form>
```



SOFTWAREarchitekt.at

View

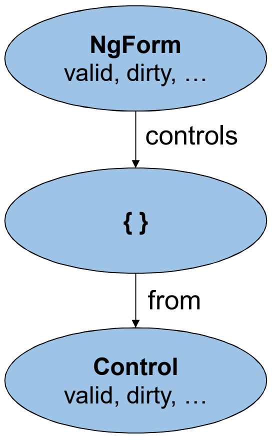
```
<form #f="ngForm">

   <input type="text" name="from"
      [(ngModel)]="from" required minlength="3">

   <div *ngIf="!f?.controls['from']?.valid">
      …Error…
   </div>

</form>
```



SOFTWARE*architekt.at*

View

```
<form #f="ngForm">

   <input type="text" name="from"
      [(ngModel)]="from" required minlength="3">

   <div *ngIf="!f?.controls['from']?.valid">
      …Error…
   </div>

   <div
      *ngIf="f?.controls['from'].hasError('required')">
      …Error…
   </div>
</form>
```



SOFTWARE*architekt.at*

DEMO

SOFTWAREarchitekt.at

Eigene Validierungs-Regeln

Direktiven

- Fügen Verhalten zur Seite hinzu
- Beispiel: ngModel, ngClass, ngIf, ngFor
- Kein Template im Gegensatz zu Komponenten

Validierungs-Direktive

```
<input [(ngModel)]="from" name="from" city>
```

Validierungs-Direktive

```
@Directive({
    selector: 'input[city]'


})
export class CityValidatorDirective implements Validator {

    validate(c: AbstractControl): object {
        let value = c.value;
        […]
        if (…) return { city: true };
        return {}; // Kein Fehler
    }
}
```

SOFTWARE*architekt.at*

Validierungs-Direktive

```
@Directive({
    selector: 'input[city]',
    providers: [{ provide: NG_VALIDATORS,
                  useExisting: CityValidatorDirective, multi: true}]
})
export class CityValidatorDirective implements Validator {

    validate(c: AbstractControl): object {
        let value = c.value;
        […]
        if (…) return { city: true };     .hasError('city')
        return {}; // Kein Fehler
    }
}
```

SOFTWARE*architekt.at*

Attribute berücksichtigen

```
<input [(ngModel)]="from" name="from"
              [city]="['Graz', 'Hamburg', 'Zürich']">
```

SOFTWAREarchitekt.at

Attribute berücksichtigen

```
@Directive({
    selector: 'input[city]',
    providers: [{ provide: NG_VALIDATORS,
                  useExisting: CityValidatorDirective,
                  multi: true }]
})
export class CityValidatorDirective implements Validator {

    @Input() city: string[];

    validate(c: AbstractControl): object {
        […]
    }
}
```

SOFTWAREarchitekt.at

Attribute berücksichtigen

```
@Directive({
    selector: 'input[city]',
    providers: [{ provide: NG_VALIDATORS,
                  useExisting: CityValidatorDirective,
                  multi: true }]
})
export class CityValidatorDirective implements Validator {

    @Input() city: string;
    @Input() strategy: string;

    validate(c: AbstractControl): object {
        […]
    }
}
```

SOFTWARE architekt.at

Attribute berücksichtigen

```
<input [(ngModel)]="from" name="from"
        [city]="['Graz', 'Hamburg', 'Zürich']" [strategy]="'strict'">
```

SOFTWARE architekt.at

DEMO

SOFTWAREarchitekt.at

Multi-Field-Validatoren

```
@Directive({
    selector: 'form[roundTrip]',
    providers: [ … ]
})
export class RoundTripValidatorDirective implements Validator {

    validate(control: AbstractControl): object {
        […]
    }
}
```

SOFTWAREarchitekt.at

Multi-Field-Validatoren

```
export class RoundTripValidatorDirective implements Validator {

    validate(control: AbstractControl): object {
        let group = control as FormGroup;

        let from = group.controls['from'];
        let to = group.controls['to'];

        if (!from || !to) return { };

        […]
}
```

SOFTWARE architekt.at

Multi-Field-Validatoren

```
export class RoundTripValidatorDirective implements Validator {

    validate(control: AbstractControl): object {
        let group = control as FormGroup;

        let from = group.controls['from'];
        let to = group.controls['to'];

        if (!from || !to) return { };

        if (from.value === to.value) return { roundTrip: true };

        return { };
    }
}
```

SOFTWARE architekt.at

Asynchrone Validierungs-Direktiven

```
@Directive({
    selector: 'input[asyncCity]',
    providers: [ … ]
})
export class AsyncCityValidatorDirective {

    validate(control: AbstractControl): Observable<object> {
        […]
    }

}
```

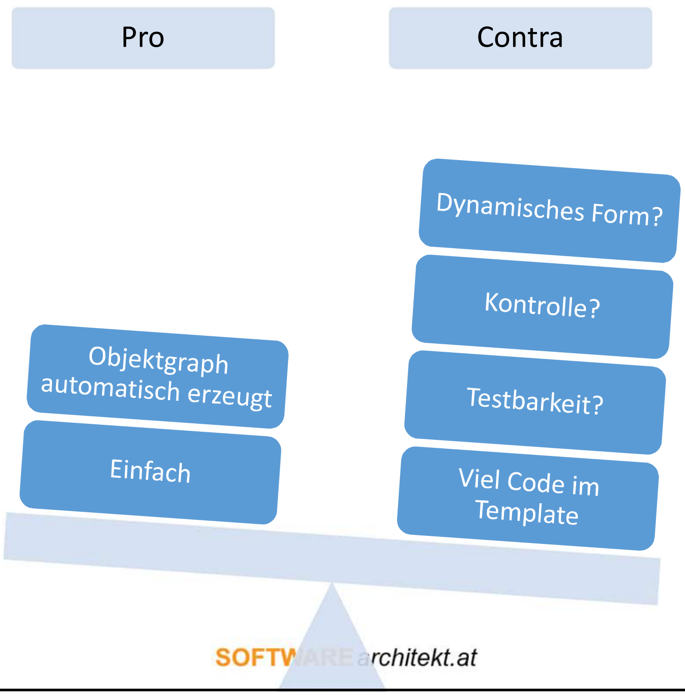
SOFTWARE architekt.at

Asynchrone Validierungs-Direktiven

Token: NG_ASYNC_VALIDATORS

SOFTWARE architekt.at

DEMO


Pro
Contra
Objektgraph automatisch erzeugt
Einfach
Dynamisches Form?
Kontrolle?
Testbarkeit?
Viel Code im Template

Reaktive Formulare

ReactiveFormsModule

```
@NgModule({
   imports: [
      ReactiveFormsModule,
      CommonModule,
      SharedModule,
      [...]
   ],
   [...]
})
export class FlightBookingModule { }
```

SOFTWARE architekt.at

Reaktive Formulare

```
export class FlightSearchComponent {

   form: FormGroup;

   […]
}
```

SOFTWAREarchitekt.at

Reaktive Formulare

```
export class FlightSearchComponent {

   form: FormGroup;

   constructor(…) {
      let fromControl = new FormControl('Graz');
      let toControl = new FormControl('Hamburg');
      this.form = new FormGroup({ from: fromControl, to: toControl});

      […]

   }
}
```

SOFTWAREarchitekt.at

Reaktive Formulare

```
export class FlightSearchComponent {

   form: FormGroup;

   constructor(…) {
      let fromControl = new FormControl('Graz');
      let toControl = new FormControl('Hamburg');
      this.form = new FormGroup({ from: fromControl, to: toControl});

      fromControl.validator = Validators.required;
      […]
   }
}
```

SOFTWARE architekt.at

Reaktive Formulare

```
export class FlightSearchComponent {

   form: FormGroup;

   constructor(…) {
      let fromControl = new FormControl('Graz');
      let toControl = new FormControl('Hamburg');
      this.form = new FormGroup({ from: fromControl, to: toControl});

      fromControl.validator =
               Validators.compose([Validators.required, Validators.minLength(3)]);
   }
}
```

SOFTWARE architekt.at

Reaktive Formulare

```
export class FlightSearchComponent {

  form: FormGroup;

  constructor(…) {
     let fromControl = new FormControl('Graz');
     let toControl = new FormControl('Hamburg');
     this.form = new FormGroup({ from: fromControl, to: toControl});

     fromControl.validator =
              Validators.compose([Validators.required, Validators.minLength(3)]);

    fromControl.asyncValidator =
              Validators.composeAsync([…]);
  }
}
```

FormBuilder

```
export class FlightSearchComponent {

  form: FormGroup;

  constructor(fb: FormBuilder, …) {
    this.form = fb.group({
      from: ['Graz', Validators.required],
      to: ['Hamburg', Validators.required]
    });
    […]
  }

}
```

SOFTWAREarchitekt.at

FormBuilder

```
export class FlightSearchComponent {

    form: FormGroup;

    constructor(fb: FormBuilder, …) {
        this.form = fb.group({
            from: ['Graz', [Validators.required, Validators.minLength(3)] ],
            to: ['Hamburg', Validators.required]
        });
        […]
    }

}
```

SOFTWARE architekt.at

FormBuilder

```
export class FlightSearchComponent {

    form: FormGroup;

    constructor(fb: FormBuilder, …) {
        this.form = fb.group({
           from: ['Graz', [Validators.required, Validators.minLength(3)], [ /* asyncValidator */ ] ],
           to: ['Hamburg', Validators.required]
        });
        […]
    }

}
```

SOFTWARE architekt.at

API

```
this.form.valueChanges.subscribe(change => {
    console.debug('formular hat sich geändert', change);
});
```

```
this.form.controls['from'].valueChanges.subscribe(change => {
    console.debug('from hat sich geändert', change);
});
```

```
let fromValue = this.form.controls['from'].value;
let toValue = this.form.controls['to'].value;
```

```
let formValue = this.form.value;
```

Reaktive Formulare

```
<form [formGroup]="form">

   <input id="from" formControlName="from" type="text">

   […]

</form>
```

Reaktive Formulare

```
<form [formGroup]="form">

   <input id="from" formControlName="from" type="text">
   <div *ngIf="!form.controls['from'].valid">…Error…</div>

   […]

</form>
```

SOFTWARE architekt.at

DEMO

SOFTWARE architekt.at

Validatoren für reaktive Formulare

Reaktive Validatoren == Funktionen

SOFTWARE architekt.at

Ein einfacher Validator

```
function validate (c: AbstractControl): object  {
    if (c.value == 'Graz' || c.value == 'Hamburg') {
        return { };
    }
    return { city: true };
}
```

SOFTWARE architekt.at

Validatoren anwenden

```
this.form = fb.group({
    from: [
        'Graz',
        [
            validate
        ],
        [
            /* asyncValidator */
        ]
    ],
    to: ['Hamburg', Validators.required]
});
```

SOFTWARE architekt.at

Parametrisierte Validatoren

```
function validateWithParams(allowedCities: string[]) {

        […]

}
```

SOFTWARE architekt.at

Parametrisierte Validatoren

```
function validateWithParams(allowedCities: string[]) {

    return (c: AbstractControl): object => {
            […]
    };

}
```

SOFTWARE architekt.at

Parametrisierte Validatoren

```
function validateWithParams(allowedCities: string[]) {

    return (c: AbstractControl): object => {

        if (allowedCities.indexOf(c.value) > -1) {
            return { }
        }

        return { city: true };

    };
}
```

SOFTWARE architekt.at

Validatoren anwenden

```
this.form = fb.group({
    von: [
        'Graz',
        [
            validateWithParams(['Graz', 'Hamburg'])
        ],
        [
            /* asyncValidator */
        ]
    ],
    nach: ['Hamburg', Validators.required]
});
```

SOFTWARE architekt.at

DEMO

SOFTWAREarchitekt.at

Asynchrone Validatoren

```
export function cityValidatorAsync(flightService) {

    return (control: AbstractControl) => {
       […]
       return observable;
    }
}
```

SOFTWAREarchitekt.at

Validatoren anwenden

```
this.form = fb.group({
    from: [
        'Graz',
        [
            validateWithParams(['Graz', 'Hamburg'])
        ],
        [
            cityValidatorAsync(this.flightService)
        ]
    ],
    to: ['Hamburg', Validators.required]
});
```

SOFTWAREarchitekt.at

Multifield-Validatoren

```
export function validateMultiField([…]) {

    return (control: AbstractControl) => {

        const formGroup = control as FormGroup;

        […]
    }

};
```

SOFTWAREarchitekt.at

Validatoren anwenden

```
this.form = fb.group({ … });
this.form.validator = validators.compose([validateMultiField([…])])
```

SOFTWAREarchitekt.at