Formulare und Validierung

Hosted by Alex Thalhammer

Inhalt

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

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

Template-getriebene Formulare

Template-getriebene Formulare

```
export class FlugSuchenComponent {

    von: string;
    nach: string;

    constructor(flugService: FlugService) {

        von = 'Graz';
        nach = 'Hamburg';
    }
}
```

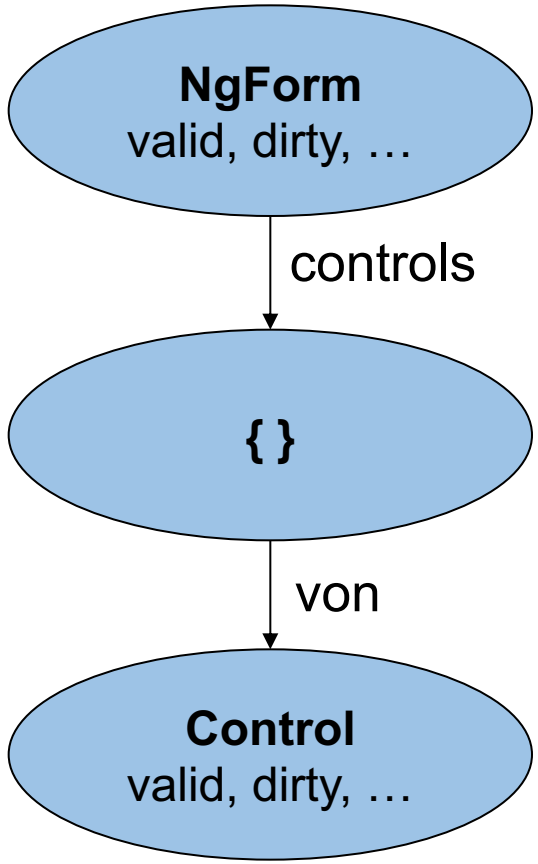
View

```
<form>

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

   […]

</form>
```



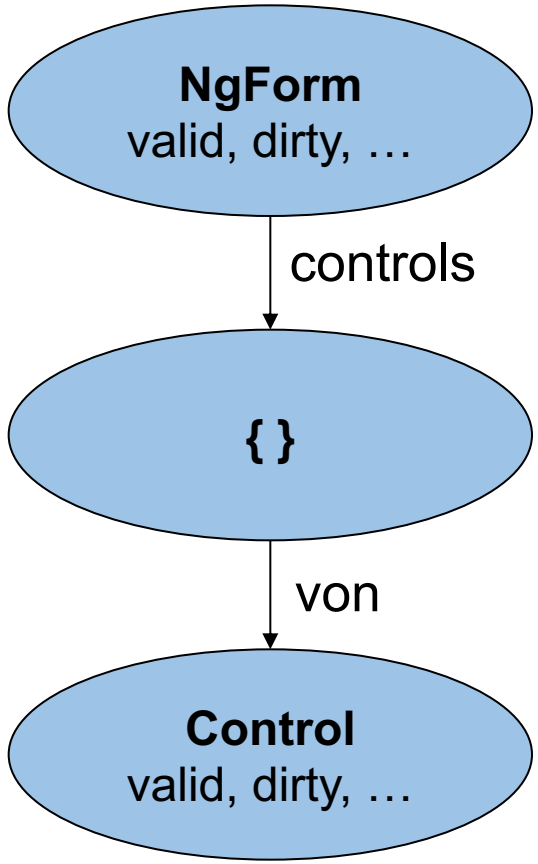
View

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

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

    […]

</form>
```



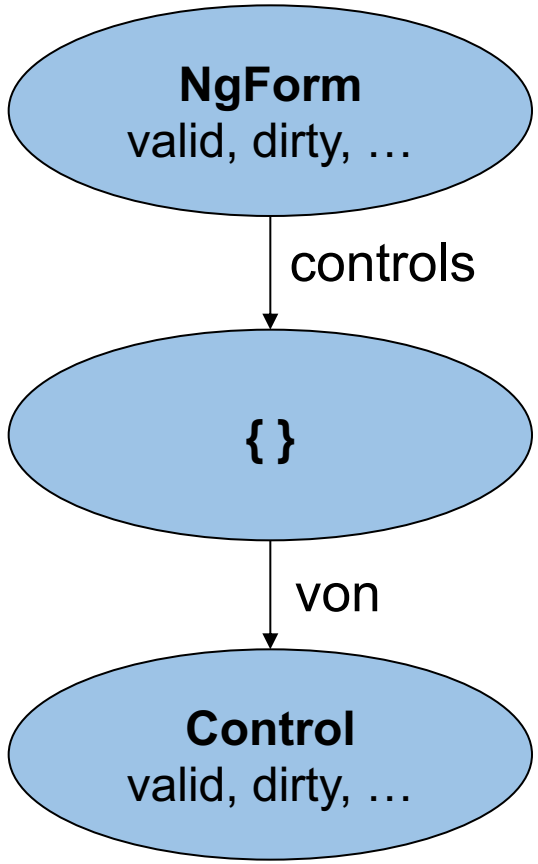
View

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

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

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

</form>
```



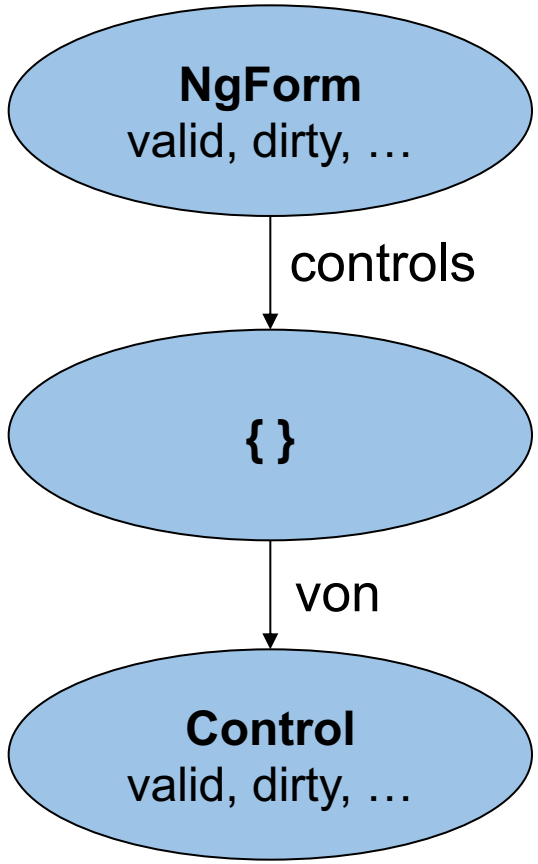
View

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

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

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

</form>
```



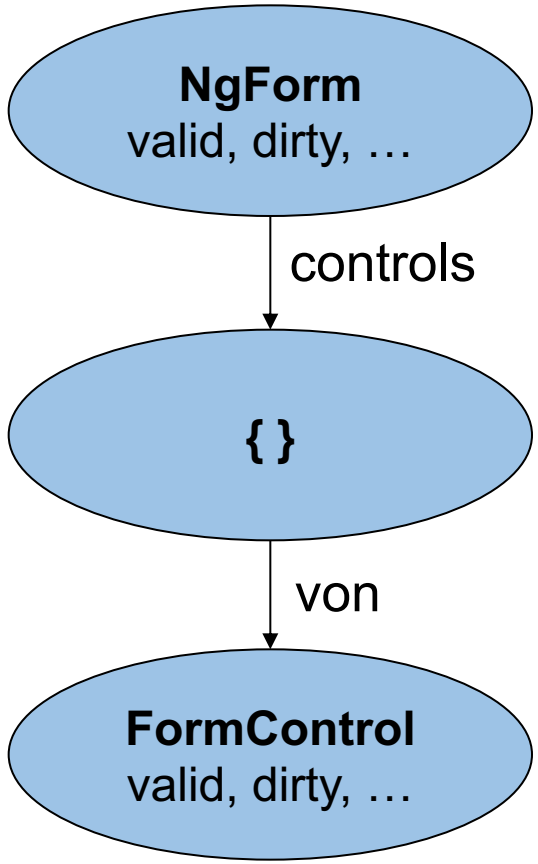
View

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

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

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

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



DEMO

LAB

Eigene Validierungs-Regeln

Direktiven

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

Validierungs-Direktive

```
<input [(ngModel)]="von" name="von" ort>
```

Validierungs-Direktive

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


})
export class OrtValidatorDirective implements Validator {

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

Validierungs-Direktive

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

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

Attribute berücksichtigen

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

Attribute berücksichtigen

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

    @Input() ort: string[];

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

Attribute berücksichtigen

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

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

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

Attribute berücksichtigen

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

Attribute berücksichtigen

```
<input [(ngModel)]="von" name="von"
        ort="Graz, Hamburg, Zürich" strategy="strict">
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Multi-Field-Validatoren

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

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

Multi-Field-Validatoren

```
export class RoundTripValidatorDirective implements Validator {

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

        let von = group.controls['von'];
        let nach = group.controls['nach'];

        if (!von || !nach) return { };

        […]
}
```

Multi-Field-Validatoren

```
export class RoundTripValidatorDirective implements Validator {

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

        let von = group.controls['von'];
        let nach = group.controls['nach'];

        if (!von || !nach) return { };

        if (von.value === nach.value) return { roundTrip: true };

        return { };
    }
}
```

Asynchrone Validierungs-Direktiven

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

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

}
```

Asynchrone Validierungs-Direktiven

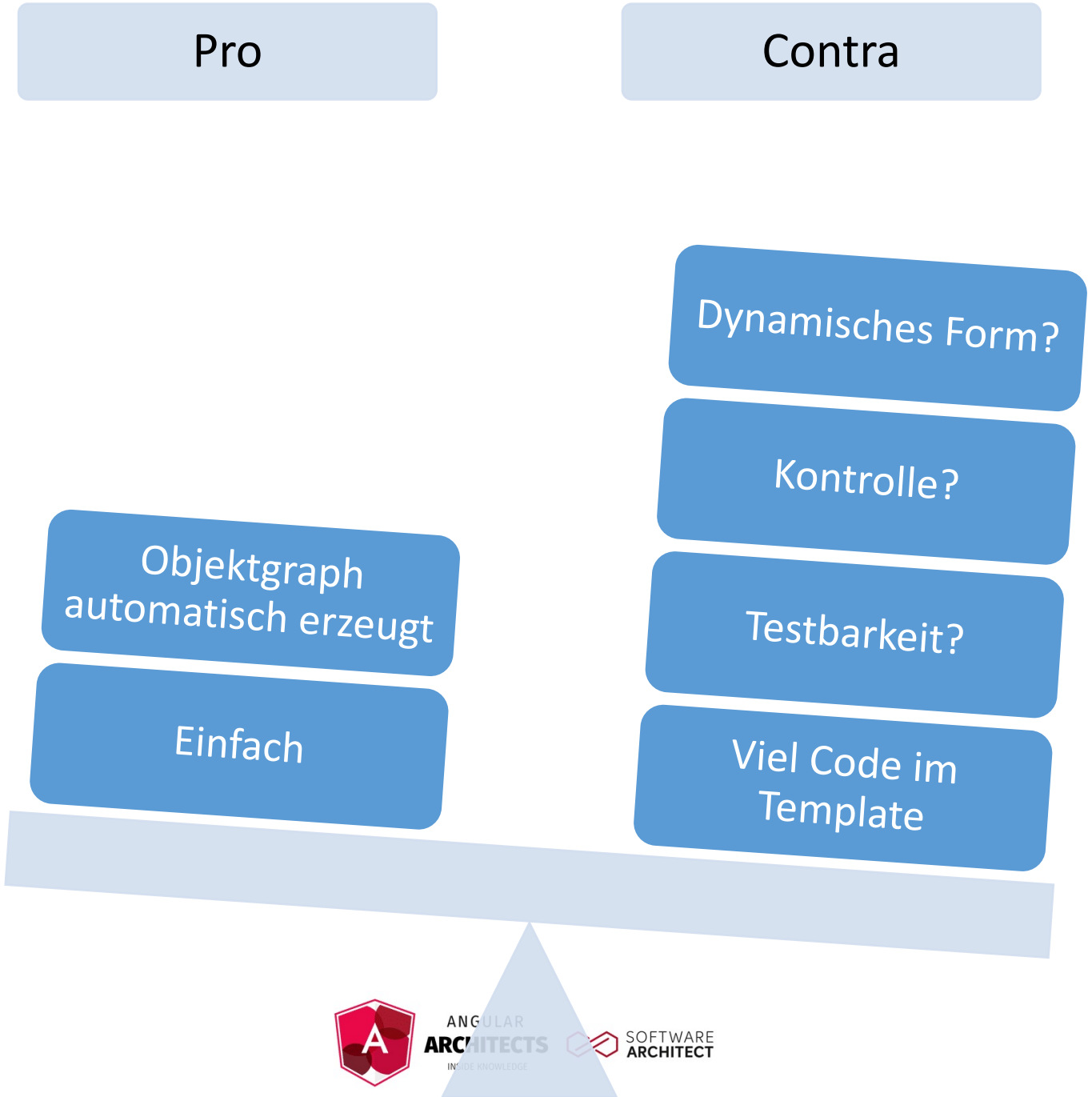
Token: NG_ASYNC_VALIDATORS

DEMO

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT



Reaktive Formulare

ReactiveFormsModule

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

Reaktive Formulare

```
export class FlugSuchenComponent {

    form: FormGroup;

    […]
}
```

Reaktive Formulare

```
export class FlugSuchenComponent {

    form: FormGroup;

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

        […]

    }
}
```

Reaktive Formulare

```
export class FlugSuchenComponent {

    form: FormGroup;

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

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

Reaktive Formulare

```
export class FlugSuchenComponent {

    form: FormGroup;

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

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

Reaktive Formulare

```
export class FlugSuchenComponent {

   form: FormGroup;

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

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

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

FormBuilder

```
export class FlugSuchenComponent {

    form: FormGroup;

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

}
```

FormBuilder

```
export class FlugSuchenComponent {

    form: FormGroup;

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

}
```

FormBuilder

```
export class FlugSuchenComponent {

    form: FormGroup;

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

}
```

API

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

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

```
let vonValue = this.form.controls['von'].value;
let toValue = this.form.controls['to'].value;
```

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

Reaktive Formulare

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

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

   […]

</form>
```

Reaktive Formulare

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

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

   […]

</form>
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Validatoren für reaktive Formulare

Reaktive Validatoren == Funktionen

Ein einfacher Validator

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

Validatoren anwenden

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

Parametrisierte Validatoren

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

        […]

}
```

Parametrisierte Validatoren

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

        […]

}
```

Parametrisierte Validatoren

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

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

}
```

Parametrisierte Validatoren

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

    return (c: AbstractControl): object => {

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

        return { city: true };

    };
}
```

Validatoren anwenden

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Asynchrone Validatoren

```
export function cityValidatorAsync(flightService: FlightService) {

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

Validatoren anwenden

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

Multifield-Validatoren

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

    return (control: AbstractControl): ValidationErrors {

        const formGroup = control as FormGroup;

        […]
    }

};
```

Validatoren anwenden

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT