Forms and Validation

Hosted by Alex Thalhammer

Outline

- Approaches
- Template-driven forms
  - How to use
  - Validation
- Reactive forms
  - How to use
  - Validation

Forms in Angular

Template-driven

- ngModel in the HTML-template
- Angular creates object tree for form
- FormsModule

Reactive

- We create the object tree in our component (TS-file)
- More control, more power
- ReactiveFormsModule

Data-driven

- Angular generates a form for a data model
- Handed over to the community ("formly")

Template-driven Forms

Template-driven Forms

```
export class FlightSearchComponent {

    from: string;
    to: string;

    constructor(flightService: FlightService) {
            from = 'Graz';
            to = 'Hamburg';
    }
}
```

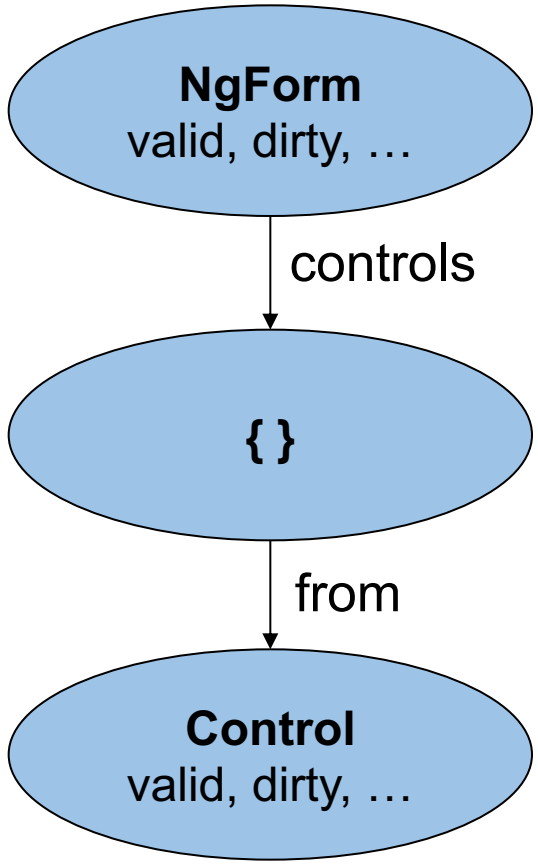
View

```
<form>

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

   […]

</form>
```



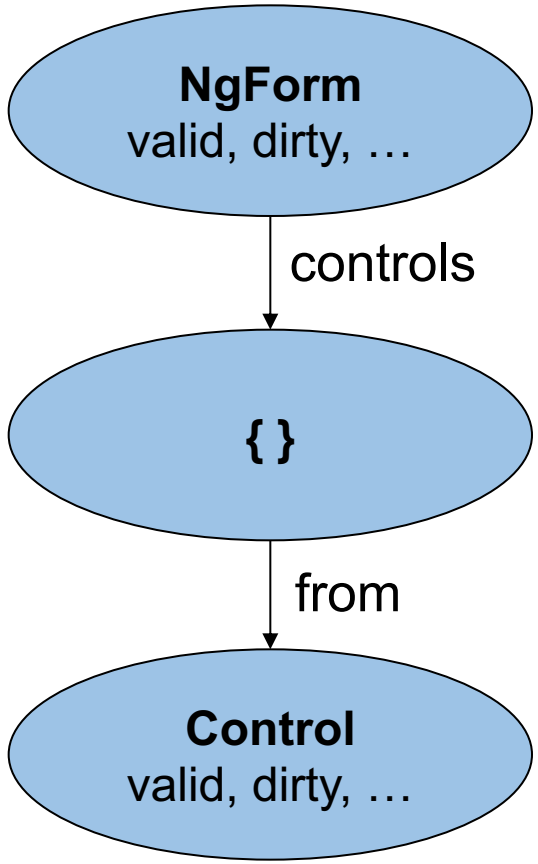
View

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

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

    […]

</form>
```



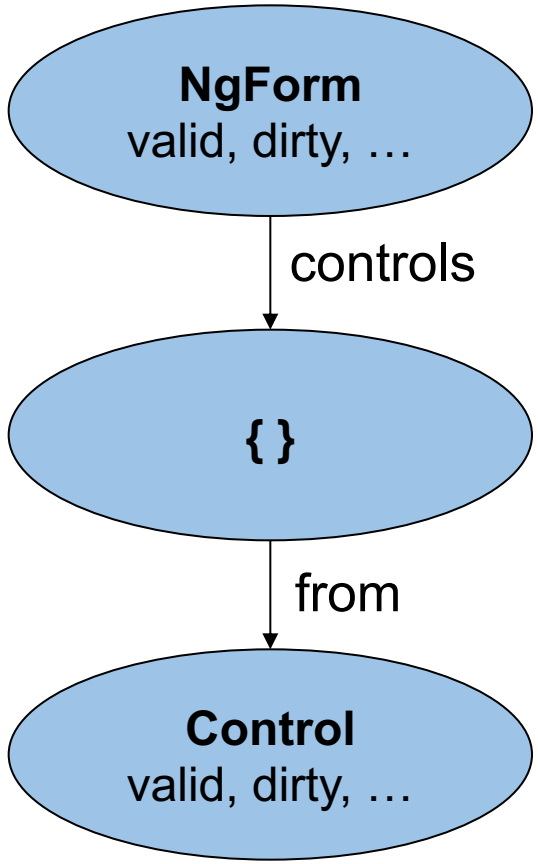
View

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

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

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

</form>
```



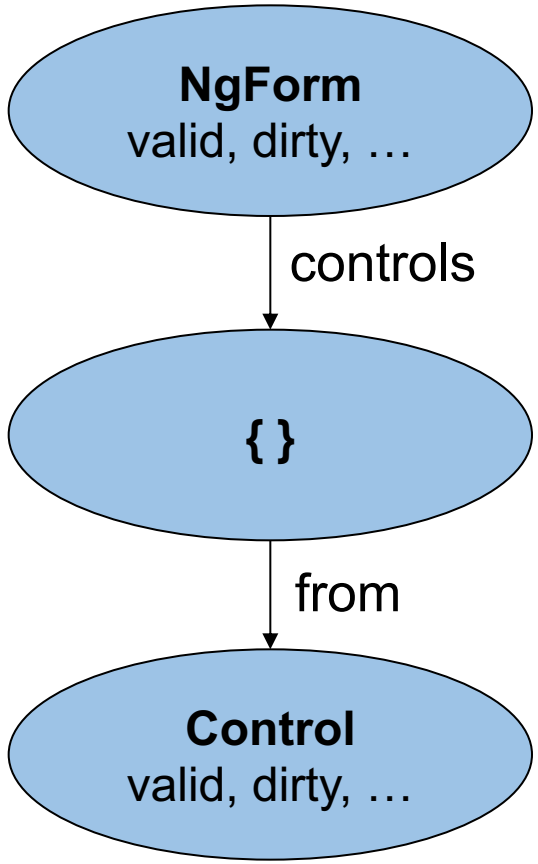
View

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

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

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

</form>
```



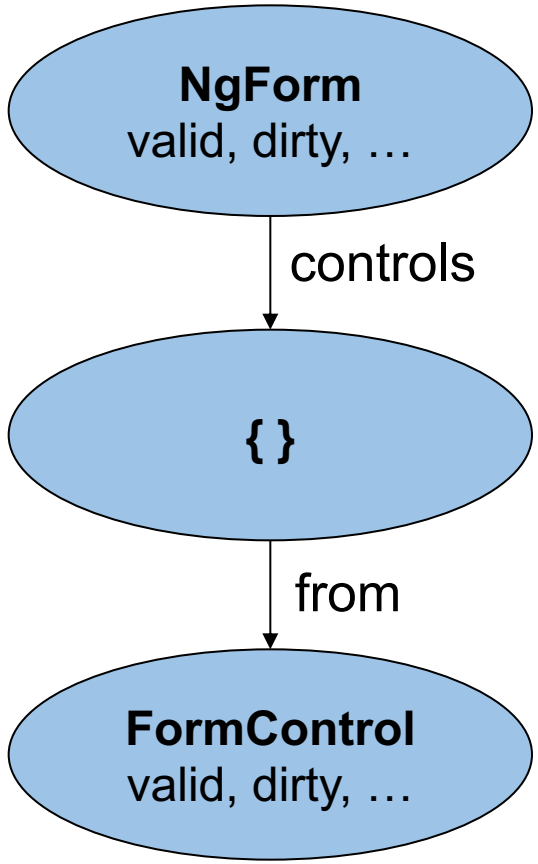
View

```
<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>
```



DEMO

Own Valididators

Directives

- Add behaviour to a component or any other HTML tag
- Built in examples
  - Attribute directives: ngModel, ngClass, ngStyle
  - Structural directives: *ngIf, *ngFor, *ngSwitch
- Custom attribute directives
  - E.g. validation directive
- No template (contrary to components)

Validation directive

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

Validation directive

```
@Directive({
    selector: 'input[city]'
})
export class OrtValidatorDirective implements Validator {

    validate(c: AbstractControl): ValidationErrors {
        let value = c.value;

        […]

        if (…) return { city: true }; // error

        return {}; // no error
    }
}
```

Validation directive

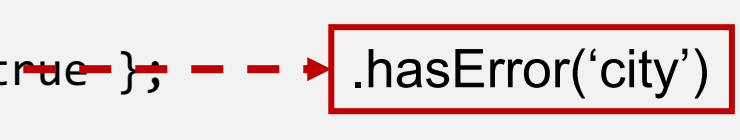
```
@Directive({
    selector: 'input[city]',
    providers: [{ provide: NG_VALIDATORS,
                  useExisting: OrtValidatorDirective, multi: true}]
})
export class OrtValidatorDirective implements Validator {

    validate(c: AbstractControl): ValidationErrors {
        let value = c.value;

        […]

        if (…) return {city: true};

        return {}; // no error
    }
}
```



Using attributes

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

Using attributes

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

    @Input() city: string[];

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

Using attributes

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

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

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

Using attributes

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

Using attributes

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Multifield Validators

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

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

Multifield Validators

```
export class RoundTripValidatorDirective implements Validator {

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

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

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

        […]
}
```

Multifield Validators

```
export class RoundTripValidatorDirective implements Validator {

    validate(control: AbstractControl): ValidationErrors {
        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 { };
    }
}
```

Asynchrone Validierungs-Directives

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

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

}
```

Asynchrone Validierungs-Directives

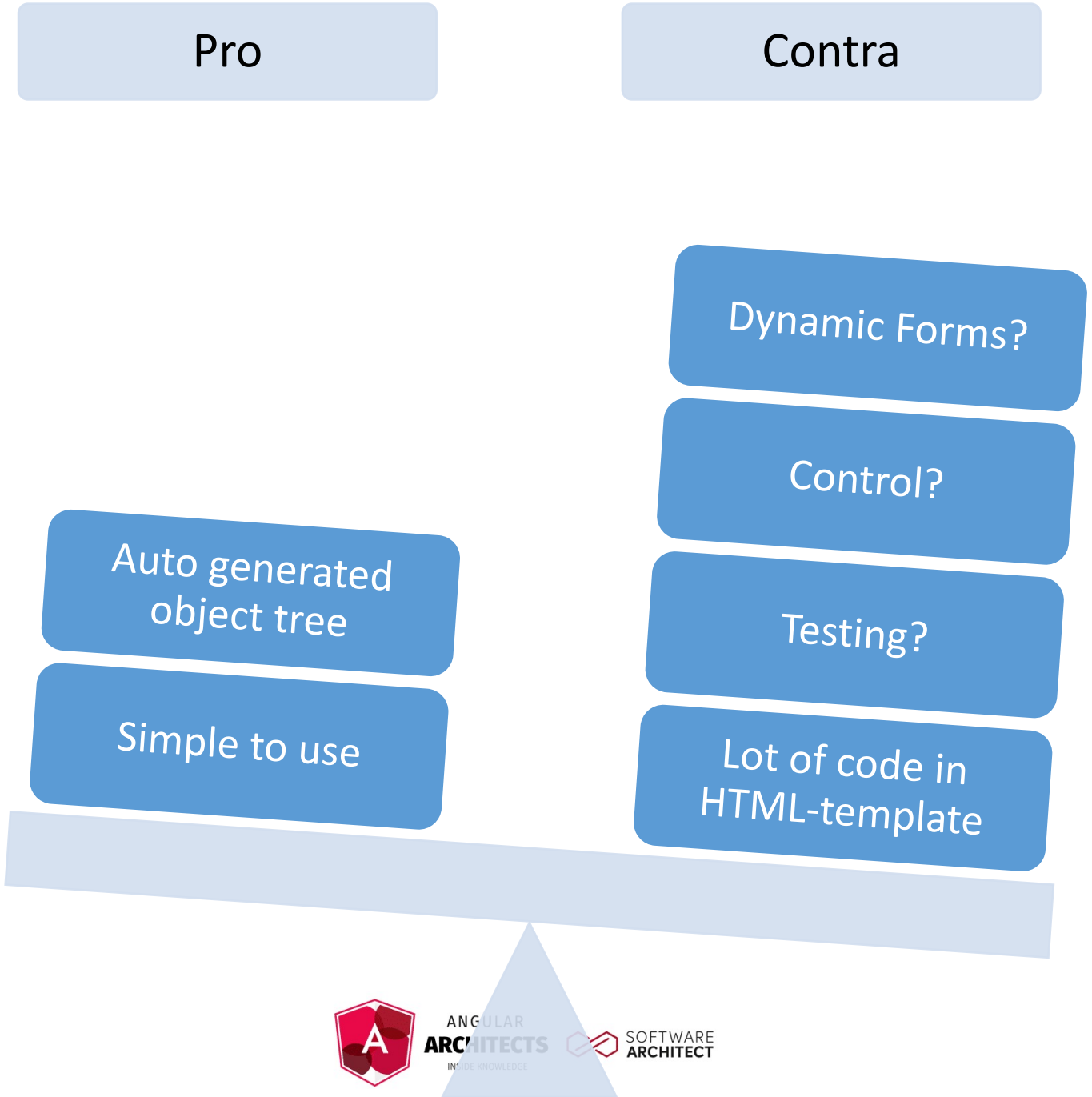
Token: NG_ASYNC_VALIDATORS

DEMO

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT



Reactive Forms

ReactiveFormsModule

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

Reactive Forms

```
export class FlightSearchComponent {

    form: FormGroup;

    […]
}
```

Reactive Forms

```
export class FlightSearchComponent {

    form: FormGroup;

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

        […]

    }
}
```

Reactive Forms

```
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;
        […]
    }
}
```

Reactive Forms

```
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)]);
    }
}
```

Reactive Forms

```
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]
        });
        […]
    }

}
```

FormBuilder

```
export class FlightSearchComponent {

    form: FormGroup;

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

}
```

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]
        });
        […]
    }

}
```

API

```
this.form.valueChanges.subscribe(change => {
    console.debug('the form has been changed', change);
});
```

```
this.form.controls['from'].valueChanges.subscribe(change => {
    console.debug('from input has been changed ', change);
});
```

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

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

Reactive Forms

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

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

   […]

</form>
```

Reactive Forms

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

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

   […]

</form>
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Validators for Reactive Forms

Reactive Validators === functions

A simple validator

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

Apply validators

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

Parametrizable validators

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

        […]

}
```

Parametrizable validators

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

        […]

}
```

Parametrizable validators

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

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

}
```

Parametrizable validators

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

    return (c: AbstractControl): object => {

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

        return { city: true };

    };
}
```

Apply validators

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Asynchrone Validatoren

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

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

Apply validators

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

Multifield validators

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

    return (control: AbstractControl): ValidationErrors {

        const formGroup = control as FormGroup;

        […]
    }

};
```

Apply validators

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Homework for this evening

Check at least one of your teams Angular projects
and find out what type of forms are being used there

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT