Angular Training

Forms

Peter Bouda, hey@peterbouda.eu

ng-lisbon.com

Introduction

- There are two ways of processing form data in Angular:
  - Template-driven: the main logic is in the template
  - Reactive: you implement the logic in the component class and exchange data between viewand class via Observables

Preparation: Import FormModule

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic'
import { FormsModule } from '@angular/forms';
// Oder ReactiveFormsModule
import { AppComponent } from './components'
@NgModule({
        imports: [
                BrowserModule,
                FormsModule,
        ],
        declarations: [AppComponent],
        bootstrap: [AppComponent]
})
export class AppModule {
}
platformBrowserDynamic().bootstrapModule(AppModule)
```

Preparation: Template-driven

- `<input>` elements have the standard HTML5 attributes
- The `name` attribute is mandatory
- For attributes like `required` and `pattern` Angular will add logic automatically
- Angular will detect the `<form>` element and create a `NgForm` object for it
- Angular will add all `<input>` elements with the `ngModel` attribute to the `NgForm` object

ng-lisbon.com

Basis Template-driven View

```
<form id="signup-form">
  <label for="email">E-Mail</label>
  <input type="text" name="email" id="email">
  <label for="password">Passwort</label>
  <input type="password" name="password" id="password">
  <button type="submit">Registrieren</button>
</form>
```

Angular Template-driven View

```
<form #signupForm="ngForm" (ngSubmit)="onSubmit(signupForm)">
  <label for="email">E-Mail</label>
  <input type="text" name="email" id="email" ngModel>
  <label for="password">Passwort</label>
  <input type="password" name="password" id="password" ngModel>
  <button type="submit">Registrieren</button>
</form>
```

Component Template-driven

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
@Component({
  selector: 'appSignupForm',
  templateUrl: 'app/signup-form.component.html',
})
export class SignupFormComponent {
  onSubmit(form: NgForm) {
    console.log(form.value);
    // {email: '...', password: '...'}
    // ...
  }
}
```

Grouping Data

```
<form #paymentForm="ngForm" (ngSubmit)="onSubmit(paymentForm)">
  <fieldset ngModelGroup="contact">
    <legend>Contact</legend>
    <label>
      Vorname <input type="text" name="firstname" ngModel>
    </label>
    <label>
      Nachname <input type="text" name="lastname" ngModel>
    </label>
    <label>
      Email <input type="email" name="email" ngModel>
    </label>
    <label>
      Telefon <input type="text" name="phone" ngModel>
    </label>
  </fieldset>
  <fieldset ngModelGroup="address">
    <!-- ... -->
  </fieldset>
  <!-- ... -->
</form>
```

Data from the Component via One-Way Binding

```
<form #signupForm="ngForm" (ngSubmit)="register(signupForm)">
  <label for="username">Benutzername</label>
  <input type="text" name="username" id="username"
    [ngModel]="generatedUserName">
  <label for="email">E-Mail</label>
  <input type="email" name="email" id="email" ngModel>
  <button type="submit">Registrieren</button>
</form>
```

Component One-Way Binding

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
// ...
@Component({
// ...
})
export class SignupFormComponent {
  generatedUser: string = generateUniqueUserID();
  onSubmit(form: NgForm) {
    console.log(form.value);
    // ...
  }
}
```

Exchange Data via Two-Way Binding

```
<form #signupForm="ngForm" (ngSubmit)="onSubmit(signupForm)">
  <label for="username">Benutzername</label>
  <input type="text" name="username" id="username" [(ngModel)]="username">
  <label for="email">E-Mail</label>
  <input type="email" name="email" id="email" [(ngModel)]="email">
  <button type="submit">Registrieren</button>
</form>
```

Component Two-Way Binding

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
@Component({
// ...
})
export class SignUpFormComponent {
  username: string = generateUniqueUserID();
  email = '';
  onSubmit(form: NgForm) {
    console.log(form.value.username);
    console.log(this.username);
    // ...
  }
}
```

Data Validation

```
<input type="text" required>

<input type="text" pattern=".{3,8}">

<input type="text" pattern=".{3,8}" required>

<input type="text" pattern="[A-Za-z0-9]{0,5}">

<input type="submit" [disabled]="!signupForm.valid">
```

Feedback via CSS classes and Styles

State	Class when in state	Class when not in state
Element visited	ng-touched	ng-untouched
Value modified	ng-dirty	ng-pristine
Value valid	ng-valid	ng-invalid

Styling for feedback

```
...
@Component({
  selector: 'app-product-edit',
  templateUrl: './product-edit.component.html',
  styles: [`
    input.ng-touched.ng-invalid, textarea.ng-touched.ng-invalid {
          border: 1px solid red;
    }
  `]
})
...
```

Reactive Forms

- More flexibility and control over data validation
- You implement all logic in the component programmatically
- You need to import the `ReactiveFormsModule`

ng-lisbon.com

Reactive Forms Component

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, FormBuilder } from '@angular/forms';
@Component({
  selector: 'app-root',
  templateUrl: 'app/app.component.html'
})
export class AppComponent {
  username = new FormControl('')
  password = new FormControl('')
  loginForm: FormGroup = this.builder.group({
    username: this.username,
    password: this.password
  });
  constructor(private builder: FormBuilder) { }
  onLogin() {
    console.log(this.loginForm.value);
    // Hier wird der Login gemacht
  }
}
```

Reactive Forms Template

```
<form [formGroup]="loginForm" (ngSubmit)="onLogin()">
  <label for="username">Benutzername</label>
  <input type="text" name="username" id="username"
    [formControl]="username">
  <br>
  <label for="password">Passwort</label>
  <input type="password" name="password" id="password"
    [formControl]="password">
  <br>
  <button type="submit">Anmelden</button>
</form>
```

Validation

```
import { Component } from '@angular/core';
import { Validators, FormBuilder, FormControl } from '@angular/forms';
@Component({
  // ...
})
export class AppComponent {
  username = new FormControl('', [
    Validators.required,
    Validators.minLength(5)
  ]);
  password = new FormControl('', [Validators.required]);
  ...
}
```

Feedback for Validation

```
<div [hidden]="username.valid || username.untouched">
  <div>
    The user name is not valid:
  </div>
  <div [hidden]="!username.hasError('minlength')">
    The minimum length is 5 characters.
  </div>
  <div [hidden]="!username.hasError('required')">
    The user name is required.
  </div>
</div>
```

Custom Validators

```
function hasExclamationMark(input: FormControl) {
  const hasExclamation = input.value.indexOf('!') >= 0;
  return hasExclamation ? null : { needsExclamation: true };
}
password = new FormControl('', [
  Validators.required,
  hasExclamationMark
]);
```

In the template:

```
<div [hidden]="!password.hasError('needsExclamation')">
  The password must contain an exclamation mark.
</div>
```

Validation in Relation to Other Elements

```
function duplicatePassword(input: FormControl) {
  if (!input.root || !input.root.controls) {
    return null;
  }
  const exactMatch = input.root.controls.password === input.value;
  return exactMatch ? null : { mismatchedPassword: true };
}

...

this.duplicatePassword = new FormControl('', [
  Validators.required,
  duplicatePassword
]);
```