1. Kommunikation zwischen Komponenten [U-050]

> *Als Nutzer möchte ich die Möglichkeit haben die E-Mail-Adresse des Erstellers eines Todos ein- bzw. auszublenden. Auch möchte ich als Nutzer eine Anzeige angeboten bekommen, welche mir die Anzahl der hinterlegten Todos angibt.*

1. Lernziele

- ✓ Sie erstellen eine Kind-Komponente, um mehrfach in einer Komponente wiederkehrende Logik und dupliziertes HTML-Markup redundanzfrei umzusetzen.
- ✓ Sie verwenden ein Input Property, um Daten von der Eltern-Komponente an die Kind-Komponente zu übergeben.
- ✓ Sie realisieren ein Output Property, um Daten über Events von der Kind-Komponente an die Eltern-Komponente zu übertragen.
- ✓ Sie verwenden bestehende Komponenten in anderen Komponenten wieder.

2. Ergebnis

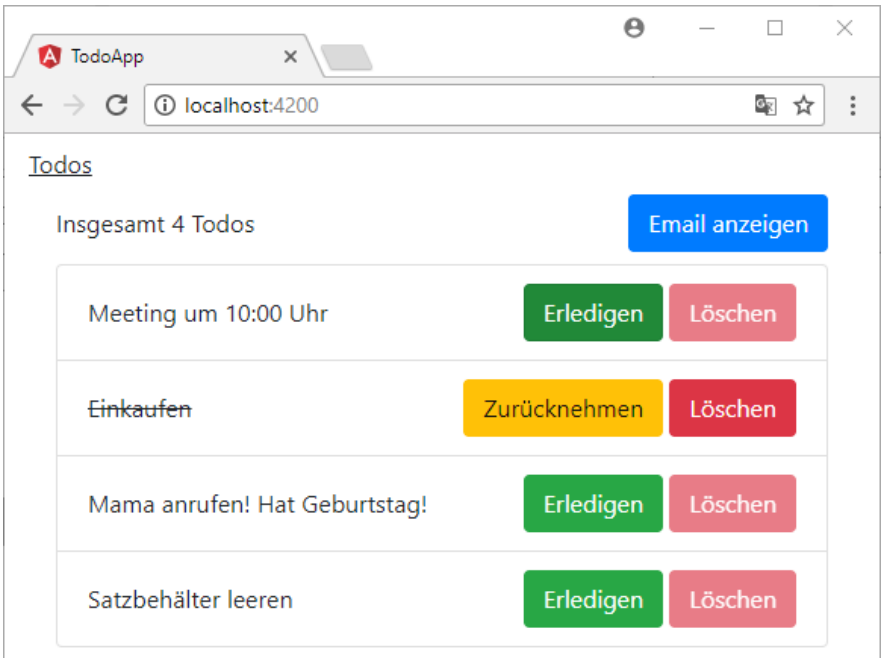

Abbildung 1 – Panel-Komponente

3. Benötigte Dateien

- `src/app/app.component.ts`
- `src/app/app.component.html`
- `src/app/todo-list/todo-list.component.ts`
- `src/app/todo-list/todo-list.component.html`
- `src/app/todo-control-panel/todo-control-panel.component.ts`
- `src/app/todo-control-panel/todo-control-panel.component.html`
- `src/app/app.module.ts`

4. Anleitung

LISTE VON TODOS IN EIGENE KOMPONENTE AUSLAGERN.

Schritt 1: Neue Komponente `TodoListComponent` erstellen

1. Erstellen sie die Datei **`src/app/todo-list/todo-list.component.ts`** und erstellen darin eine leere Klasse **`TodoListComponent`**.
2. Ergänzen Sie die Klasse mit den **`@Component`**-Decorator, dem Sie ein Objekt übergeben mit folgenden Einträgen:

   ```
   selector: 'todo-todo-list'
   ```

3. Erstellen Sie eine leere HTML Datei namens **`todo-list.component.html`** und geben Sie diesen Dateinamen als Wert für **`templateUrl`** im Dekorator an.
4. Ergänzen Sie das Feld **`declarations`** des **`@NgModule`**-Decorators der **`AppModule`**-Klasse (**`src/app/app.module.ts`**) um die Klasse **`TodoListComponent`**.
   (Wenn Ihr Editor es nicht automatisch macht, müssen Sie noch den entsprechenden **`import`** in **`app.module.ts`** eintragen)

Hinweis:
Die Angular cli bietet auch die Möglichkeit Komponenten, Services, etc. automatisiert zu erzeugen. Dabei werden die nötigen Dateien erstellt und im entsprechenden Modul hinzugefügt.

```
ng generate component todo-list
```

Schritt 2: Todo-Liste von `AppComponent` nach `TodoListComponent` auslagern

5. Verschieben Sie alle Methoden und Properties der `AppComponent`, bis auf die Objektvariable `title`, in die eben erstellte Klasse `TodoListComponent`. Beachten sie dabei auch die dazugehörigen `import`-Statements im Kopf der Datei.
6. Verschieben Sie alle Elemente innerhalb des `<main>`-Tags des Template der `AppComponent` (`src/app/app.component.html`), welche die Todo-Liste definieren, in das leere Template der `TodoListComponent` (`src/app/todo-list.component.html`).
7. Um die Todo-Liste weiterhin in der Applikation anzuzeigen, fügen Sie das `<todo-todo-list>`-Element im Template der `AppComponent` dort ein, wo Sie zuvor die Elemente entfernt haben.
8. Starten Sie die Anwendung im Terminal mit `ng serve` und überzeugen Sie sich, dass sich im Verhalten der Anwendung nichts geändert hat.

Erstellen eines Control-Panels in eigener Komponente

Schritt 3: Neue Komponente `TodoControlPanelComponent` erstellen und dem Modul `AppModule` bekannt machen

9. Erstellen sie die Klasse `TodoControlPanelComponent` in der Datei `src/app/todo-control-panel/todo-control-panel.component.ts` mit dazugehörigem Template. Fügen Sie im Template einen einfachen HTML Tag ein (z.B. ein `<h1>`) mit dem Sie einen Text ausgeben

10. Definieren Sie eine `@Component`-Annotation und geben Sie der Komponente den Selektor `todo-todo-control-panel` und verweisen auf das eben erstellte Template mit `templateUrl`.
11. Ergänzen Sie das Feld `declarations` in der Klasse `AppModule` um die Klasse `TodoControlPanelComponent.`
12. Fügen Sie in Zeile 1 der `todo-list.component.html`-Datei den Tag `<todo-todo-control-panel></todo-todo-control-panel>` ein.
13. Lassen Sie die Anwendung laufen und versichern sich, dass der erwartete Text auf dem Bildschirm auftaucht.

Template der Eltern-Komponente anpassen und Inputs in Template nutzen

---

Schritt 4: Anzahl von Todos an `TodoContolPanel` übergeben, anzeigen und `TodoControlPanel` in TodoList einbinden

14. Erweitern Sie die Komponentenklasse `TodoControlPanelComponent` um eine neue, öffentlich zugreifbare Objektvariable `todoCount` vom Typ `number`.
15. Dekorieren Sie `todoCount` mit dem `@Input`-Decorator.
16. Entfernen Sie Ihr HTML aus Schritt 7 und erstellen Sie im Template der Komponente ein `<div>`-Element mit der CSS-Klasse `„w-75 mb-4“`, in welches alle weiteren Elemente eingefügt werden.
17. Geben Sie `todoCount` innerhalb eines weiteren `<div>`-Elements aus. Beispiel: `Insgesamt {{todoCount}} Todos`
18. Binden Sie dabei `todoCount` an die Anzahl der Elemente im `todoList`-Array (`todoList.length`) im Template der `todo-list.component.html`.

Schritt 5: Mit Hilfe einer zweiten Input-Property die Objektvariable showEmail der Klasse `TodoListComponent` an die `TodoControlPanelComponent` übergeben

1. Erweitern Sie die Komponentenklasse `TodoControlPanelComponent` um eine neue Eigenschaft `displayAllEmails` vom Typ `boolean`.
2. Dekorieren Sie `displayAllEmails` mit dem `@Input`-Decorator.
3. Binden Sie die Eigenschaft `displayAllEmails` an das Attribut `showEmail` des `<todo-todo-control-panel>`-Elements mittels Property Binding.
4. Schauen Sie sich Ihre Anwendung jetzt an, verändern dann in Ihrem Editor den Wert von `showEmail` in `TodoListComponent` to `false` und beobachten, was mit der Anzeige der Todos passiert. Ändern Sie zum Schluss den Wert wieder auf `true`.

KOMPONENTENKOMMUNIKATION MITTELS OUTPUT

Schritt 6: Erstellen eines Buttons in `TodoControlPanelComponent` mit dynamischem Text in Abhängigkeit von `showEmail`. Bei Klick auf Button den Wert von `showEmail` invertieren/umkehren

19. Erstellen Sie im Template der `TodoControlPanelComponent` einen `<button>` und geben Sie als dessen Text abhängig vom Wert der Property `displayAllEmails` entweder „Email ausblenden“ oder „Email anzeigen“ aus. Weisen Sie dem <button>-Element die CSS-Klasse „float-right btn btn-primary“ zu.
20. Erstellen Sie in der Komponentenklasse die Methode `toggleShowEmail` ohne Parameter und ohne Rückgabewerten.
21. Invertieren Sie innerhalb der Methode den Wert von `displayAllEmails`.
22. Binden Sie die Methode mittels Event-Binding an das `click`-Event des zuvor erstellten Buttons.
23. Testen Sie die Funktionalität der Anwendung.

Schritt 7: Änderungen der Eigenschaft `displayAllEmails` mittels Output Event Emitters an Elternkomponente propagieren

24. Erstellen Sie die Objektvariable `displayAllEmailsChange` und weisen Sie dieser eine neue Instanz der Klasse `EventEmitter<boolean>` zu.
25. Dekorieren Sie die Eigenschaft `displayAllEmailsChange` mit dem `@Output`-Dekorator.
26. Propagieren Sie in der `toggleShowEmail` Methode durch Aufruf der `emit`-Methode des soeben initialisierten `EventEmitters` den Wert von `displayAllEmails`

Schritt 8: In TodoListComponent auf mittels Event-Binding `showEmailChange` Event der `TodoControlPanelComponent` reagieren

27. Erstellen Sie in der Klasse `TodoListComponent` eine neue Methode `onShowEmailChange` mit dem Parameter `value` vom Typ `boolean`. Diese Funktion wird keinen Rückgabewert haben.
28. Setzen Sie in der soeben angelegten Methode den Wert der Eigenschaft `showEmail` auf den übergebenen Wert von `value` und geben Sie den Wert von `showEmail` auf der Konsole aus, um zu sehen, welche Daten hier übermittelt werden.
29. Binden Sie mittels Event-Binding in `todo-list.component.html` das `displayAllEmailsChange`-Event des `<todo-todo-control-panel>`-Elements an die soeben erstellte Methode. Übergeben Sie dabei als Parameter die `$event`-Variable. Der Wert von `$event` beinhaltet den an die `TodoControlPanelComponent#displayAllEmailsChange.emit`-Methode übergebenen Wert.
30. Testen Sie ihre Implementierung in der Browserkonsole durch wiederholtes Klicken auf den Email-ein- bzw. ausblenden-Knopf und schauen sich die Ausgabe auf der Konsole an. Löschen Sie das `console.log` in der Methode `TodoListComponent.onShowEmailChange` danach wieder.

Schritt 9: Mittels Two-Way-Binding auf `displayAllEmailsChange` reagieren

31. Um die Invertierung der `displayAllEmails`- Eigenschaft mittels Two-Way-Binding zu realisieren, löschen Sie die Methode `onShowEmailChange` der Klasse und das Event-Binding des Elements im Template von `todo-list.component`.
32. Verwenden Sie nun Two-Way-Binding für das Attribut `displayAllEmails` des Elements. Dies ist durch die Namensgebung des `EventEmitters` mit `displayAllEmailsChange` möglich.

```
[(displayAllEmails)]="showEmail
```

33. Lassen Sie die Anwendung laufen und beobachten Sie die neue Funktionalität, die Emailadressen der Todos ein- bzw. auszublenden.

5. Kontrollfragen

- Wie erstellen Sie eine neue Komponente und integrieren diese in die bestehende Anwendung?
- Wie können Komponenten mittels Eingabe- und Ausgabe-Parametern mit den Elternkomponenten kommunizieren?
- Was ist notwendig, um Two-Way-Binding einsetzen zu können?

6. Weiterführende Materialien

- Kommunikation zwischen Komponenten
  `https://tinyurl.com/gs-angular-components-communi`