

# 1. Formulare [U-ogo]

Als Nutzer möchte ich neue Todos anlegen, um meine Todo-Liste auf dem aktuellen Stand zu halten. Des Weiteren möchte ich auf Fehlereingaben aufmerksam gemacht werden, damit ich diese korrigieren kann.

## 1. Lernziele

- ✓ Sie verwenden Template-Driven Forms um ein Eingabeformular zu erstellen.
- ✓ Sie setzen Two-Way-Binding ein.
- ✓ Sie verwenden Direktiven für die Validierung.
- ✓ Sie erklären, wie der Zustand eines Formulars geprüft werden kann.

## 2. Ergebnis

TodoApp

localhost:4200/todos/new

Todos

Neues Todo erstellen

## Todo erstellen

Titel

Bitte Titel angeben.

E-Mail

ha

Bitte eine gültige E-Mail-Adresse angeben.  
Die E-Mail muss mehr als 8 Zeichen besitzen.

Beschreibung

Speichern

### 3. Benötigte Dateien

- `src/app/todo-create/todo-create.component.html`
- `src/app/todo-create/todo-create.component.ts`
- `src/app/todo-create/todo-create.component.css`
- `src/app/app.component.html`
- `src/app/app.module.ts`
- `src/app/todo.service.ts`

### 4. Anleitung

#### ERSTELLEN EINER NEUEN FORM-KOMPONENTE

---

Schritt 1: Erstellen eines Formulars (**form**) mit den Eingabefeldern für **title**, **email** und **description** eines Todos sowie einen **Submit**-Button im Template der **TodoCreateComponent**

1. Erstellen Sie ein **<form>**-Element im leeren Template der **TodoCreateComponent**.
2. Erstellen Sie innerhalb des Formulars ein **<input>**-Element für den Titel. Geben Sie dabei für die Attribute **id** und **name** den Wert **"title"** an. Weisen Sie dem **<input>**-Element die CSS-Klasse „form-control“ zu.
3. Erstellen Sie als Beschriftung ein dazugehöriges **<label>**-Element. Geben Sie dabei als **for**-Attribut die ID des **<input>**-Elements an.
4. Gruppieren Sie Label und Input mit einem umschließenden **<div>**-Element. Diesem weisen Sie bitte die CSS-Klasse „form-group“ zu.
5. Erstellen sie analog dazu ein **<input>**-Element für **email**. (type: **"email"**). CSS-Klassen analog.
6. Erstellen sie analog dazu eine **<textarea>** für **description**. CSS-Klassen analog.
7. Zuletzt erstellen sie einen Speichern-Button vom Typ **submit**. Weisen Sie dem **<button>**-Element die CSS Klassen „btn btn-primary“ zu.

---

Schritt 2:      **FormsModule** in **AppModule** aufnehmen

8. Fügen Sie das **FormsModule** aus **@angular/forms** dem **imports**-Array der Klasse **AppModule** hinzu.

---

Schritt 3: In **todoCreateComponent** neue Objektvariable **todo** vom Typ **TodoItem** deklarieren und leeres **TodoItem** zuweisen (**id = undefined**)

9. Erstellen Sie in der Klasse **TodoCreateComponent** eine neue Objektvariable **todo** vom Typ **TodoItem**.
10. Weisen Sie **todo** eine neue Instanz eines **TodoItems** zu. Achten Sie dabei darauf, dass **id** dabei nicht gesetzt wird.
11. Betrachten Sie das Formular im Browser.

---

Schritt 4: Input-Felder mittels **ngModel**-Direktive an Eigenschaften von **todo** binden

12. Erstellen Sie mittels der **ngModel**-Direktive des **<input>**-Elements ein Two-Way-Binding des **title**-Eingabefeldes an die Eigenschaft **title** von **todo**.
13. Gehen Sie für **email** und **description** analog vor.

---

Schritt 5: Bei **submit**-Event des Forms die Objektvariable **todo** in Konsole ausgeben

14. Definieren Sie die Methode **onSubmit()** ohne Rückgabewert in der Klasse **TodoCreateComponent** und geben Sie darin die Objektvariable **todo** über die Konsole aus.
15. Binden Sie mittels Event-Binding die erstellte Methode an das **submit**-Event des **<form>**-Elements im Template.
16. Überprüfen Sie Ihre Implementation im Browser.

## NEUES TODO MITTELS SERVICE ANLEGEN

---

Schritt 6: **create(todo: TodoItem)**-Methode in **TodoService** erstellen, welche mittels POST neues **todo** auf dem Server anlegt und in **onSubmit** aufrufen.

17. Definieren Sie die Methode **create** in der Klasse **TodoService** mit dem Parameter **todo** vom Typ **TodoItem**.

18. Führen Sie einen POST-Request an die **BASE\_URL** aus. Orientieren Sie sich hierbei an der Umsetzung den Methoden **getAll()** und **update()** des **Services**.
19. Rufen Sie in der Methode **onSubmit** der Klasse **TodoCreateComponent** die Methode **create()** des **TodoServices** auf, um das anzulegende **Todo** an das Backend zu übergeben.
20. Speichern Sie ein ausgefülltes Formular ab und überprüfen dann die Konsolenausgabe.

---

Schritt 7: Formular zurücksetzen, nachdem das Todo auf dem Server angelegt wurde

21. Fügen Sie der **onSubmit**-Methode einen neuen Parameter **form** vom Typ **NgForm** aus **@angular/forms** hinzu.
22. Definieren Sie für das Formular eine Templatevariable **todoForm** und weisen Sie ihr **ngForm** zu.

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

23. Übergeben Sie **todoForm** beim Aufruf der **onSubmit**-Methode im Template der Komponente.
24. Rufen Sie die **then**-Methode auf dem durch die **create**-Methode von **TodoService** zurückgebenden **Promise**-Objekt auf. Übergeben Sie dabei eine Pfeilfunktion, innerhalb der Sie die Methode **reset** von **form** aufrufen.

## VALIDIERUNG DES FORMULARS

---

Schritt 8: Titel als Pflichtfeld deklarieren und bei **submit** prüfen, ob das Formular korrekt ist.

25. Ergänzen Sie im Template der **TodoCreateComponent** das **<input>**-Element des Titels um das Attribut **required**.
26. Prüfen Sie in der Methode **onSubmit** vor dem Aufruf der **create**-Methode auf Korrektheit des Forms (**form.valid**).

---

Schritt 9: Email als korrekte Email und mit einer Mindestlänge von 8 Zeichen validieren

27. Fügen Sie dem `<input>`-Element für **email** ein **email**-Attribut hinzu.

28. Fügen Sie ein **minlength**-Attribut mit dem Wert 8 hinzu.

## ANZEIGEN VON FEEDBACK

---

Schritt 10: Deaktivieren des **submit**-Buttons im Template in Abhängigkeit der Validierung

29. Binden Sie das Attribut **disabled** des **submit**-Buttons mittels Property-Binding an die negierte/umgekehrte Eigenschaft **valid** des Formulars.

---

Schritt 11: Fehlermeldung ausgeben, wenn kein Titel eingegeben wurde

30. Definieren Sie für das `<input>`-Element für den Titel eine Templatevariable **title** und weisen Sie ihr **ngModel** zu.

31. Fügen Sie innerhalb des gruppierenden `<div>`-Elements ein zusätzliches `<div>`-Element mit der vordefinierten CSS-Klasse **feedback** hinzu.

32. Zeigen Sie das `<div>`-Element mit Hilfe einer **ngIf**-Direktive nur dann an, wenn die Eigenschaft **title.invalid** den Wert **true** besitzt.

33. Geben Sie innerhalb des Elements den Text „Bitte Titel angeben“ aus.

---

Schritt 12: Fehlermeldung ausgeben, wenn das E-Mail-Feld nicht valide ist und dabei für **email** und **minlength** unterschiedliche Meldungen anzeigen

34. Definieren Sie für das `<input>`-Element für die E-Mail eine Templatevariable **email** und weisen Sie ihr **ngModel** zu.

35. Fügen Sie innerhalb des gruppierenden `<div>`-Elements ein zusätzliches `<div>`-Element mit der vordefinierten CSS-Klasse **feedback** hinzu.

36. Zeigen Sie das `<div>`-Element mit Hilfe einer **ngIf**-Direktive nur dann an, wenn die Eigenschaft **email.invalid** den Wert **true** besitzt.

37. Erstellen Sie innerhalb des Feedback-Elements ein neues **<div>**-Element, das nur angezeigt wird, wenn die Eigenschaft **email.errors** den Wert **minlength** hat und geben Sie darin den Text „Die E-Mail muss mehr als 8 Zeichen besitzen.“ aus.
38. Erstellen Sie analog dazu für den **error**-Wert **email** ein **<div>**-Element mit dem Text „Bitte eine gültige E-Mail-Adresse angeben.“.

## OPTIONAL

---

Schritt 13: Validierungszustand des Formulars über die CSS-Klassen **ng-valid** und **ng-invalid** darstellen

39. Erstellen Sie in der zu **TodoCreateComponent** zugehörigen CSS-Datei eine Definition der Klassen **ng-valid** und **ng-invalid**.
40. Färben Sie in der jeweiligen Klasse den linken Rand je nach Validierungszustand (**valid** : grün, **invalid**: rot) ein. Verwenden Sie dazu die CSS-Eigenschaft **border-left**.

---

Schritt 14: Fehlermeldung für invalide E-Mail nur dann anzeigen, wenn das Feld verändert (**dirty**) oder verlassen (**touched**) wurde

41. Fügen Sie der **ngIf**-Bedingung für das Feedback-**<div>**-Element der E-Mail eine Bedingung hinzu, welche prüft ob die Eigenschaften **touched** oder **dirty** der Templatevariable **true** ist.

Beispiel:

```
*ngIf= "field.invalid && (field.dirty || field.touched)"
```

## 5. Kontrollfragen

- Welche Aufgabe erfüllt die **NgForm**-Direktive?
- Welche Aufgabe erfüllt die **NgModel**-Direktive?
- Welche Direktiven zur Validierung kennen Sie?
- In welchen Zuständen kann sich ein **NgModel** befinden?

- Wie werden die Zustände bei der Validierung eingesetzt

## 6. Weiterführende Materialien

- Angular Dokumentation über Template-Driven Forms  
<https://tinyurl.com/gs-angular-forms>
- Angular Dokumentation über Validierung  
<https://tinyurl.com/gs-angular-validation>