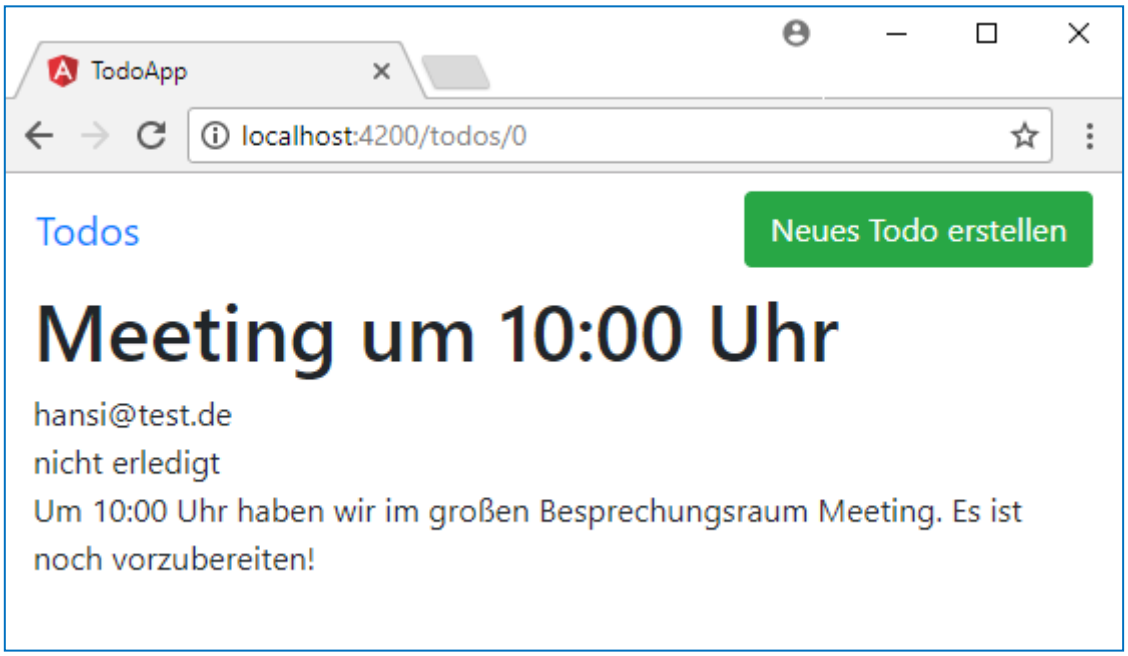
1. Routing [U-080]

> *Als Nutzer möchte ich die Möglichkeit haben, Details zu einem Todo in einer eigenen Seite anzuzeigen. Diese Seite möchte ich durch Klick auf ein Todo in der Todo-Liste angezeigt bekommen, sowie direkt über Eingabe der URL. Auch möchte ich die Möglichkeit haben, über einen Button zu einer Seite gebracht zu werden, in welcher ich später ein neues Todo anlegen kann.*

1. Lernziele

- ✓ Sie definieren eine neue Route im Routing-Modul.
- ✓ Sie nutzen Child-Routing um die Hierarchie in einer SPA abzubilden
- ✓ Sie setzen das Angular Router-Model für Navigation in der SPA ein und nutzen dabei sowohl Links als auch programmatische Navigation.
- ✓ Sie wandeln eine interne Komponente in eine navigierbare Komponente um.

2. Ergebnis



© ARS Computer und Consulting GmbH, 2023

3. Benötigte Dateien

- `src/app/app-routing.module.ts`
- `src/app/app.module.ts`
- `src/app/app.component.html`
- `src/app/todo-detail.component.ts`
- `src/app/todo-tetail.component.html`
- `src/app/todo.service.ts`
- `src/app/todo-list.component.html`

4. Anleitung

App Routing Module erstellen

Schritt 1: Router-Modulklasse **`AppRoutingModule`** erstellen und in **`AppModule`** aufnehmen

1. Erstellen Sie die Datei **`src/app/app-routing.module.ts`**. Definieren und exportieren Sie dort die Klasse **`AppRoutingModule`**.
2. Dekorieren Sie die erstellte Klasse mit dem **`@NgModule`**-Dekorator und den Parametern **`imports`** und **`exports`** wie folgt:

```
@NgModule({
      imports: [RouterModule.forRoot(routes)],
      exports: [RouterModule]
})
```

3. Nehmen Sie **`AppRoutingModule`** unter **`imports`** in **`AppModule`** auf.
4. Definieren Sie außerhalb der Klasse eine Konstante **`routes`** vom Typ **`Routes`** aus dem Modul **`@angular/router`** und weisen Sie dieser ein leeres Array zu.

Schritt 2: Neue Route `/todos` für Komponente `TodoListComponent` erstellen

5. Definieren Sie für `TodoListComponent` eine neue Route mit Pfad `todos`, indem Sie dem `routes`-Array ein neues Objekt/Element hinzufügen. Beispiel:

```
{
      path: 'url/to/component',
      component: ComponentReferenz
}
```

Schritt 3: `TodoListComponent` im Template `AppComponent` durch ein Router-Outlet ersetzen

6. Ersetzen Sie im Template der `AppComponent` das Element `<app-todo-list>` durch ein neues Element `<router-outlet>.`

Schritt 4: Redirect von Seiten-Root (‚/') auf Route `todos` hinzufügen

7. Erstellen sie eine Weiterleitung von `“ “` (leerer string) zu `‘/todos‘`. Hierfür erstellen Sie ein neues Objekt im `routes`-Array, welches den Pfad `“ “` spezifiziert und mittels des Parameters `redirectTo` mit dem Wert ‘`todos`‘ auf die Seite der Todos verweist.
8. Fügen Sie noch eine Definition zum Objekt hinzu, welche der Objektvariablen `pathMatch` den Wert `full` zuweist.

Neue Komponente für Detailansicht erstellen und aufrufbar machen

Schritt 5: Erstellen der Komponente `TodoDetailComponent` und das Einbinden der URL `todos/:id` für die Komponente in das `Routes`-Array im `AppRoutingModule`

9. Erstellen Sie die Komponente `TodoDetailComponent` mit dem Selector `app-todo-detail` und allen benötigten Dateien (`.ts`, `.html`).

10. In `app-routing-module.ts` definieren Sie die Route `todos/:id` für die Komponente. Orientieren Sie sich hierfür an der `TodoListComponent`.

Schritt 6: Bei Klick auf Titel (Link) eines Todos in `TodoListComponent` auf `TodoDetailComponent` mit Angabe der ID navigieren

11. Umschließen Sie die Ausgabe des Todo-Titels im Template der Komponente `TodoListComponent` mit einem Link (`<a>`-Element).
12. Binden Sie darin das Attribut `[routerLink]` an die ID des Todos.

Schritt 7: Über `ActivatedRoute` in `ngOnInit()` der Klasse `TodoDetailComponent` den Parameter `id` ermitteln und auf der Konsole ausgeben

13. Injizieren Sie `ActivatedRoute` im Konstruktor von `TodoDetailComponent`.
14. Spezifizieren Sie, dass `TodoDetailComponent` das Interface `OnInit` implementiert.
15. Erstellen Sie die Methode `ngOnInit` um das `OnInit`-Interface zu implementieren.
16. In `ngOnInit` rufen Sie die Methode `subscribe()` auf der Eigenschaft `paramMap` der im Konstruktor übergebenen Instanz der `ActivatedRoute` auf. Übergeben Sie eine Pfeilfunktion mit dem Parameter `paramMap` an die `subscribe()` Funktion:

```
this.route.paramMap.subscribe(paramMap => { ... });
```

17. In der Pfeilfunktion rufen Sie auf `paramMap` die Methode `get` mit dem Parameter `id` auf. Von welchem Datentyp ist die Konstante `id`?
18. Wandeln Sie den Parameterwert von `id` in eine `Number` um und legen den erhaltenen Wert in einer Konstante `todoId` ab. Beispiel:

```
const todoId = Number(idString)
```

Geben Sie `todoId` auf der Konsole aus.

Anzeigen eines Todo in der Detailansicht

Schritt 8: Neue Service-Methode `getById` in `TodoService` erstellen, die unter Angabe einer ID das entsprechende Todo in einem `Promise`-Objekt vom Server zurückliefert

19. Erstellen Sie in der Klasse `TodoService` eine neue Methode `getById` mit einem Parameter `id` vom Typ `number`.
20. Verwenden Sie die Methode `get` von `http` In der Funktion. Geben Sie dabei `TodoItem` als generischen Parameter an.

```
this.http.get<genericType>(...);
```

21. Konstruieren Sie aus `BASE_URL` und dem Parameter `id` einen neuen String welchen Sie der `get`-Methode als Parameter übergeben.
22. Konvertieren Sie die Rückgabe von `get()` mit der Methode `firstValueFrom()` in ein `Promise`-Objekt um und geben Sie dieses als Rückgabewert der Methode zurück.

Schritt 9: Service-Methode `getById` in `TodoDetailComponent` verwenden, um eine Objektvariable `todo` vom Typ `TodoItem` zu initialisieren.

23. Deklarieren Sie die Objektvariable `todo` vom Typ `TodoItem` in der Klasse `TodoDetailComponent`.
24. Injizieren Sie `TodoService` im Konstruktor von `TodoDetailComponent`.
25. Rufen Sie die Methode `getById` des `TodoService` innerhalb von `ngOnInit` mit dem ermittelten Parameter `todoId` auf.
26. Erweitern Sie die Promise-Kette nach `getById()` mit einem `then()`-Aufruf, in dem Sie mit Hilfe einer Funktion den erhaltenen Wert in der Objektvariablen `todo` ablegen.

Schritt 10: Eigenschaften von `Todo` im Template von `TodoDetailComponent` anzeigen, nachdem `todo` initialisiert wurde.

27. Geben Sie mittels Interpolation die Eigenschaften (`title`, `description`, `email`, `completed`) von `todo` im Template von `TodoDetailComponent` aus.
28. Der Abruf des Todos vom Server geschieht asynchron. Prüfen Sie daher im Template der Komponente, ob `todo` einen Wert besitzt. Umschließen Sie dazu die Elemente des Template mit einem neuen `<div>`-Element, welches über eine `ngIf`-Direktive nur dann angezeigt wird, wenn `todo` ein Wert zugewiesen wurde, d.h `todo` nicht leer ist.

Neue Komponente für das Erstellen neuer Todos erstellen und aufrufbar machen (wird in der nachfolgenden Übung fertig gestellt)

Schritt 11: `title` in Template der `AppComponent` als Link mit `routerLink` nach ‚/' definieren, sowie `css`-Klasse `navbar-brand` zuweisen. Letztlich .css-File der `AppComponent` löschen.

29. Umschließen Sie im Template der `AppComponent` die Interpolation der Objektvariablen `title` mit einem `<a>`-Element.
30. Weisen Sie dem Attribut `routerLink` des `<a>`-Elements den root-Pfad **"/"** zu.
31. Weisen Sie dem `<a>`-Element die `css`-Klasse `navbar-brand` zu.
32. Entfernen Sie die .css-Datei der `AppComponent`.

Schritt 12: Erstellen der Komponente `TodoCreateComponent`

33. Erstellen Sie die Komponente `TodoCreateComponent` mit dem Selector `app-todo-create` und allen benötigten Dateien `(.ts, .html)`

Schritt 13: Neue Route `todos/new` für Komponente `TodoCreateComponent` erstellen

34. Definieren Sie die Route `todos/new` für die Komponente in `app-routing.module.ts`. Orientieren Sie sich hierfür an der `TodoListComponent`. Beachten Sie, dass Sie diese Route oberhalb von `todos/:id` definieren! Verstauschen Sie testweise die Reihenfolge von `todos/new` und `todos/:id` und rufen Sie `todos/new` auf. Was können Sie beobachten?

Schritt 14: Neuen Button in `<nav>`-Element der `AppComponent` hinzufügen, der programmatisch auf `TodoCreateComponent` navigiert

35. Injizieren Sie die Klasse `Router` mittels Dependency Injection im Konstruktor der Klasse `AppComponent`.
36. Erstellen Sie in der Klasse `AppComponent` eine neue öffentliche Methode `navigate` mit einem Parameter `url` vom Typ `string` und ohne Rückgabewert.
37. Rufen Sie innerhalb der erstellten Methode `navigateByUrl()` der Router-Instanz mit dem Parameter `url` auf.
38. Fügen Sie im Template innerhalb des `<nav>`-Elements einen neuen Button mit dem Text „Neues Todo erstellen“ hinzu.
39. Optional: Geben Sie dem Button die CSS-Klasse `btn-success`.
40. Binden Sie das `click`-Event des Buttons an die `navigate`-Methode. Geben Sie dabei den String `todos/new` als Übergabeparameter an.

Optional: Routing mittels Hierarchie

Schritt 15: Refactoring des Routings durch Einführung einer Hierarchie mit Hilfe der Eigenschaft `children`

41. Ersetzen Sie in `app-routing.module.ts` der Route `todos` die Eigenschaft `component` mit einer neue Eigenschaft `children`. Diese ist ein Array von Routendefinitionen.

42. Erstellen Sie in dem `children`-Array eine neue Route mit einem leeren String als `path` und der Objektvariablen `component`, welcher Sie `TodoListComponent` zuweisen.
43. Verschieben sie analog die Routen `todos/new` und `todos/:id` in das `children`-Array und entfernen Sie dabei das `todos/`-Präfix.

5. Kontrollfragen

- Wie wird das Routing-Module konfiguriert?
- Wie kann in HTML über Links navigiert werden=
- Ist es möglich auf programmatische Weise Routen aufzurufen?
- Wie werden Variablen über eine Route an Komponenten übergeben?

6. Weiterführende Materialien

- Angular-Dokumentation über Routing
  `https://tinyurl.com/gs-angular-routing`