# Docu

# Vision

## Beliebige viele Ebenen von Normtext

Auf Navigationsebene eine Zieharmonika. Eventuell nicht ganz ausgeklappt.

## Searchtree

# Implementierung

## Beliebige viele Ebenen von Normtext

Content-elements: navContentElements

<nav-content-elements data=“ navContentElements[subProjectsIndex]“>

Falls navContentElements[subProjectsIndex] === undefined, ist keine Zieharmonika bzw. diese geschlossen. Nur im selectedElement wird navContentElements[subProjectsIndex] mit einem Wer befüllt => die Zieharmonika wird angezeigt.

# Detailimplementierung

**URL**: Ein URL besteht aus „denkwelten/user-link/userId#projects/0/subprojects/1“

## User-link oder userId ändert sich

**Korrekt**: user-link.component.ts wird mit folgenden URLS aufgerufen:

* „denkwelten/user-link/“ //Meldung: keine user-id - stopp
* „denkwelten/user-link/userId“ //fragments wird an content übergeben

Falls eine user-id vorhanden ist, wird über den central-service projects geladen („denkwelten/user-link/userId) - siehe Pkt1.

projects und fragment werden dann als Parameter an content.component.ts übergeben.

## Fragment ändert sich

In content.component.ts wird centralService.getPathNodes(fragment, projects) aufgerufen.

getPathNodes(fragment:**string**, dwNodes?: **any**[])

1. Falls keine projects vorhanden sind (this.projects === undefined) werden diese aus Firebase geladen. Dies ist (derzeit?) nur beim Aufruf/Reload des Browsers der Fall.
   * Es werden derzeit also alle projects auf einen Sitz aus Firebase geladen und zwar mit der URL „denkwelten/user-link/userId“.
   * Später kann man das aufdröseln, und für jedes project eine eigene Tabelle machen und erst bei Bedarf nachladen.
2. Falls die projects bereits geladen sind, (this.projects !== undefined) enfällt Schritt 1 und gleich weiter mit 3.
3. **Array pathNodes erstellen**: Es wird im centralService ein Array von pathNodes erstellt. Und zwar mittels folgendem Aufruf:

this.pathNodes = centralService.getPathNodes(this.projects, this.fragment). pathNodes;

1. Ein pathNode hat folgende Struktur:

type: „projects“ //

selectedNode: … //

selectedNodeIndex: … //

partialFragment: string // z.B. „projects“ oder „normtext“

fragment: string // „projects“/subprojects/normtext“

selectedNode und selectedNodeIndex können dabei undefined sein (z.B. bei „projects/0/subprojects“). Oder auch bei „/projects“ – wenn es also keine „dwNodesId“ gibt.

**Ergebnis - Assert**: Nach der Änderung im Fragment ist folgendes gegeben/asserted:

1. this.projects enthält die projects (genaugenommen ein array aus projects).
2. this. pathNodes enhält alle gültigen pathNodes

Jedem pathNode entspricht dabei eine Array-Ebene im Fragement/jsonFile.

* ???(jedes project enthält mindestens ein subproject – wobei das ein Dummy sein kann.)?

## Anzeige

this.pathNodes wird mit this.oldPathNodes verglichen.

Falls in einem pathNode nur der selectedNode/selectedNodeIndex geändert ist, muss lediglich die Selektierung geändert werden, die dargestellte Liste aber nicht neu aufgebaut werden.

*edit.component* ist die Schaltzentrale. Diese wird angestoßen durch

Änderung in nodePaths? Observable

Projects wird an *nav.component* geschickt. *nav.component* weiß, wie

Der nodePaths-Array wird vom Ende (Blattknoten, nodePaths [nodePaths.length-1] ) aus durchsucht.

Die Übergaberegeln für .

Wobei diese nur darin bestehen, welche

## Editieren

*edit.component* hat Symbole für das Editieren . **selector**: **'dw-edit'**

|  |  |  |
| --- | --- | --- |
| *edit.component* | | |
| **fa-plus-circle** | **insertNode()** | **(click)="insertNode()** |
|  |  |  |
|  |  |  |

Auszug aus: *normtext.component.html, searchtree.component.html*

<**dw-edit  
 [editNode]="{dwNode: undefined, index: dwNodes?.length, dwNodes: dwNodes}"  
 (onInsertNode)="onInsertNode(*$event*)"  
 (onDeleteNode)="onDeleteNode(*$event*)"  
 (onAddNodeLevel)="onAddNodeLevel(*$event*)"**>

|  |  |
| --- | --- |
|  | |
| *normtext.component.html*  *searchtree.component.html* |  |
| **[editNode]="{dwNode: dwNode, index: i, dwNodes: dwNodes}"** | @Input() **editNode**: **any**; |
|  |  |

*Edit-element.component* hat die Funktionen für das Editieren.

|  |  |  |  |
| --- | --- | --- | --- |
| *edit.component* | *edit.component* | *normtext.component* | *Edit-element.component* |
| insertNode()  (click)=“ insertNode“ | onInsertNode .emit(**this**.**editNode**) | onInsertNode  (editNode) | onInsertNode  (editNode) |
| deleteNode()  (click)=“deleteNode“ | onDeleteNode .emit(**this**.**editNode**) |  |  |
| addNodeLevel()  (click)=“ addNodeLevel“ | onAddNodeLevel .emit(**this**.**editNode**) |  |  |

## Speichern

Es gibt einen Speicher-Button ganz oben rechts. Damit werden alle projects gemeinsam gespeichert.