Zero1one

Dokumentation von Gruppe 2

vorgelegt am 28. Januar 2020

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs [Kursbezeichnung]

von

[Vorname Nachname]

|  |  |
| --- | --- |
| Betreuer in der Ausbildungsstätte: | DHBW Stuttgart: |
| [Name des Unternehmens] [Titel, Vorname und Nachname d. Betreuers] [Funktion des Betreuers] [Unterschrift des Betreuers] | [Titel, Vorname und Nachname des wissenschaftl. Betreuers/Prüfers] |

**Inhaltsverzeichnis**

[Abkürzungsverzeichnis (bei Bedarf) III](#_Toc31093439)

[Abbildungsverzeichnis (bei Bedarf) IV](#_Toc31093440)

[Tabellenverzeichnis (bei Bedarf) V](#_Toc31093441)

[1 Einleitung 1](#_Toc31093442)

[2 Hauptteil 2](#_Toc31093443)

[2.1 Eine Abschnittsüberschrift 2](#_Toc31093444)

[2.2 Zweiter Abschnitt 2](#_Toc31093445)

[2.2.1 Ein Abbildungsbeispiel 2](#_Toc31093446)

[2.2.2 Ein Tabellenbeispiel 3](#_Toc31093447)

[3 Nächstes Kapitel 4](#_Toc31093448)

[3.1 Abschnitt 4](#_Toc31093449)

[3.1.1 Unterabschnitt 4](#_Toc31093450)

[3.1.2 Unterabschnitt 4](#_Toc31093451)

[3.2 Entsprechend 4](#_Toc31093452)

[3.2.1 Fortgesetzt 4](#_Toc31093453)

[3.2.2 Nochmal fortgesetzt 4](#_Toc31093454)

[4 Schlussteil 5](#_Toc31093455)

[Literaturverzeichnis 8](#_Toc31093456)

# Abkürzungsverzeichnis (bei Bedarf)

AktG = Aktiengesetz

BFH = Bundesfinanzhof

# Abbildungsverzeichnis (bei Bedarf)

[Abb. 1: Eine Abbildung 2](#_Toc31092940)

# Tabellenverzeichnis (bei Bedarf)

[Tab. 1: Eine Beispieltabelle 3](#_Toc31092943)

1. Einleitung

In der Einleitung sind die Problemstellung und die daraus resultierende Zielsetzung der Arbeit exakt zu formulieren und die Vorgehensweise zu begründen. Notwendige Abgrenzungen sind hier vorzunehmen. Anstelle der Überschrift „Einleitung“ kann auch eine aussagekräftigere Überschrift verwendet werden.

1. Webanbindung

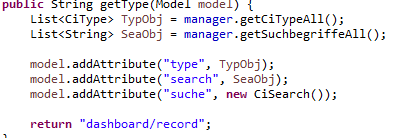
Um die Webanbindung der Applikation zu ermöglichen, mussten zuerst ein Controller und ein HTML-Template dem Projekt hinzugefügt werden. Zum Testen während der Entwicklung wurde auf den Spring internen Apache Tomcat Server zurückgegriffen. Hierfür mussten keine weiteren Voreinstellungen vorgenommen werden. Der Tocmcat-Server startet automatisch auf Port 8080.

* 1. Dateien, Struktur und Übermittlung

Um nun alles für die Darstellung der Software vorzubereiten, haben wir einen Controller-Package namens „com.gruppezwei.zero1one.controller“ im Java-Projekt angelegt. Darin wurden die Controller-Klassen, wie bereits im Kapitel über Thymleaf beschrieben, angelegt. Begonnen mit dem „Homecontroller“, der für das Dashboard und somit für das Anzeigen der Ci-Typen und Ci-Records zuständig ist. Innerhalb des Controllers wurden „@SpringBootApplication“ zur Ausführbarkeit der Klasse und „@Controller“ um Spring zu sagen, dass es sich um einen Controller handelt. Außerdem wurde mit „@Autowired“ eine Verbindung zur Java-Klasse „PersistenceManager“ geschaffen, um auf die Methoden zum Abrufen und Speichern in der Datenbank zuzugreifen.

Innerhalb der „public class Homecontroller“ wurden dann die Get- und Post-Methoden mit „@GetMapping“ oder „@PostMapping“ definiert. In Klammern dahinter, wird als „value“ die URL unter der die nachfolgende Java-Methode erreichbar ist, mitgegeben. Beispielsweise: @GetMapping(value = "/dashboard"

Danach folgt die Methode, die ein Model mitgegeben bekommt. In diesem Model können beliebig Attribute gespeichert werden, die dann an das Frontend übermittelt werden. Innerhalb der Methode werden dann alle Daten Java-Seitig vorbereitet und an das Model übergeben.



Beispielsweise wird in dieser Methode zuerst eine Liste von CiType erstellt. Diese bekommt durch die Methode „getCiTypeAll()“ aus dem „PersistenceManager“ alle CiTypen geliefert. Diese Liste wird nachher zur Darstellung aller vorhandenen Ci-Typen benötigt. Danach wir deine Liste vom Datentyp String erstellt, die von der Methode „getSucbegriffeAll()“ die ebenfalls im „PersistanceManager“ definiert ist, alle Suchbegriffe bekommt. Diese Suchbegriffe sind dann in der Suche als Vorschlag sichtbar.

Beiden Listen werden dann in den Zeilen darunter an das Model übergeben. Im Model sind diese dann unter „type“, bzw. „search“ aufrufbar. Außerdem geben wir dem Model noch ein leeres Objekt vom Datentyp CiSearch mit. In diesem wird nachher unser Suchwort gespeichert und dann wieder über eine Post-Methode an die Java-Applikation zurückgegeben.

Im Return der Methode wird ein String zurückgegeben. Dieser beinhaltet die Adresse des Templates, dass für die Darstellung genutzt werden soll. In diesem Fall heißt das Template „record“ und befindet sich im Ordner „dashboard“.

Wie bereits im Kapitel Thymleaf beschrieben, sind die Templates HTML-Dateien, die im Templates-Ordner in den „resources“ des Java-Projekts abgespeichert sind. Diese Templates wurden von den Kollegen die das Frontend gestaltet haben erstellt.

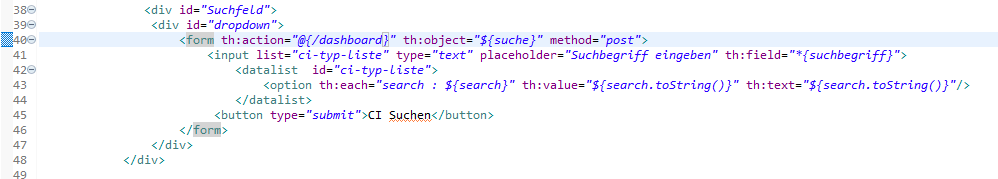
* 1. Darstellung auf dem Client

Wie die eben übermittelten Daten nun dargestellt werden, ist mittels bestimmter Thymleaf-Syntax beschrieben.



Hier sieht man den typischen Head einer HTML-Datei. Zuerst wurde dem Template die Ergänzungen in Zeile 2 „xmlns:th=*"http://www.thymleaf.org/"*> hinzugefügt, um die Datei für Thymleaf vorzubereiten. Außerdem wurde in Zeile 6 der Quellordner für Inhalte wie Bilder oder CSS-Dateien angegeben, um die Daten, die bereits über HTML eingebunden waren, entsprechend zu hinterlegen.

Schauen wir uns nun exemplarisch die Einbindung der Suche an.



In der Zeile 40 wurde ein <form> Tag, der bis Zeile 46 geht hinzugefügt. Dazwischen haben wir alle wichtigen Funktionen für die Suche untergebracht. Der Form-Tag ist ein HTML-Tag und kennzeichnet ein Formular. Im ersten Tag sehen wir bereits typische Thymleaf-operationen. Die „th:action“ gibt dabei an, an welche Post-Methode die Eingabe gesendet werden soll. „th:object“ betitelt das Obejekt, dass übermittelt werden soll. In unserem Fall ist es das „suchen“ Objekt, das wir unserem Model mitgegeben haben. Dies ist vom Datentyp „CiSearch“. Außerdem definieren mit „method“ noch die Übermittlungsmethode. Dies ist eine Standard http-Methode.

Mit dem nachfolgenden „<input>“ Tag, erstellen wir ein Feld, in das der Benutzer etwas eintragen kann. In unserem Fall das Suchfeld. Hier benutzen wir „th:field“\*{Suchbegriff}“>“ um den eingegebenen Begriff in den String „Suchbegriff“§ innerhalb des CiSearch-Obejkts zu speichern. Dieser String ist in der entsprechenden Java-Klasse definiert. Die <datalist> ermöglicht, dass eine Liste mit Vorschlägen zur Verfügung steht.  
 Sie wird mit <option th:each=*"search : ${search}"* th:value=*"${search.toString()}"* th:text=*"${search.toString()}"*/> befüllt. Dabei wird mit “th:each” jedes Objekt der Liste, die im Model unter „search“ gespeichert ist, angefasst und mit „th:text“ als „option“ in die Aufklappliste aufgenommen.

In Zeile 45 ist Button vom Typ Submit definiert. Dieser löst die ganze Aktion aus und veranlasst somit das Senden des Suchbegriffs über eine Post-Methode.

1. Nächstes Kapitel

Text

* 1. Abschnitt

Text

* + 1. Unterabschnitt

Text

* + 1. Unterabschnitt

Text - bitte daran denken, dass jede Gliederungsebene mindestens zwei Elemente umfassen muss.

* 1. Entsprechend

Text

* + 1. Fortgesetzt

Text

* + 1. Nochmal fortgesetzt

Text

1. Schlussteil

**Anhang (bei Bedarf)**

[Anhang 1: Beispielüberschrift 5](#_Toc505081892)

[Anhang 2/1: Teil 1 des Anhangs 2 5](#_Toc505081893)

[Anhang 2/2: Teil 2 des Anhangs 2 6](#_Toc505081894)

Anhang 1: Beispielüberschrift

Hier können Sie dann Unterlagen einfügen (Empfehlung: Im Text auf den Anhang verweisen). Die Überschrift des Anhangs bitte über Start – Formatvorlagen – Anhang formatieren; nur dann kann ein korrektes Anhangsinhaltsverzeichnis erstellt werden, das nicht im Inhaltsverzeichnis vorne erscheint.

Anhang 2/1: Teil 1 des Anhangs 2

Bei Anhängen, die nicht auf eine Seite passen, können Sie den Anhang ggf. aufteilen.

Anhang 2/2: Teil 2 des Anhangs 2

Bei Anhängen, die nicht auf eine Seite passen, können Sie den Anhang ggf. aufteilen.

# Literaturverzeichnis

**Preiß, N. (2007):** Entwurf und Verarbeitung relationaler Datenbanken, München/Wien: Oldenbourg

**Steger, J. (2006):** Kosten- und Leistungsrechnung, 4. Aufl., München/Wien: Oldenbourg

**Stoi, R. (2003):** Management und Controlling von Intangibles, in: Studium & Praxis, 4. Jg., Nr. 1, S. 34-46

**Falls Sie Citavi nutzen, müssen Sie dieses Literaturverzeichnis löschen!**

**Erklärung**

Ich versichere hiermit, dass ich meine *[Projektarbeit / Bachelorarbeit / Seminararbeit] mit dem Thema: [Thema einfügen]* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Unterschrift)