Vorlagen-*Management* für *Mail*-Service

Ing. Thomas Herzog

BACHELORARBEIT

Nr. S1310307011-A

eingereicht am
Fachhochschul-Bachelorstudiengang

Software Engineering

in Hagenberg

im Juli 2015

Diese Arbeit entstand im Rahmen des Gegenstands

Gegenstand??

im

Semester??

Betreuer:

FH-Prof. DI Dr. Dobler

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 14. Juli 2015

Ing. Thomas Herzog

Inhaltsverzeichnis

Erklärung **iii**

Kurzfassung **vi**

Abstract **vii**

1 Einleitung **1**
1.1 Das Unternehmen curecomp Software Service GmbH 1
1.2 Das Vorlagenmanagement für den *Mail*-Service 2
1.3 Die Rahmenbedingungen 2

2 Das Ziel des Projekts **4**
2.1 Die funktionalen Ziele 5
2.2 Die technischen Ziele 6

3 Das Lösungskonzept **7**
3.1 Die Spezifikation der Vorlagen-*API* 7
3.1.1 Die Schnittstellen und abstrakten Klassen 7
3.2 Die Spezifikation der Vorlagenintegration 7
3.2.1 Das Vorlagen-*Management* in Typescript und Javascript 7
3.2.2 Das Vorlagen-*Management* in CDI 7
3.2.3 Das Vorlagen-*Management* in JSF 7
3.2.4 Das Vorlagen-*Management* in *Mail*-DB-Schema 7

4 Die Realisierung **8**
4.1 Die Implementierung der Spezifikationen 8
4.1.1 Die Implementierung für *CKEditor* 8
4.1.2 Die Implementierungen für CDI 8
4.1.3 Die Implementierungen für JSF 8
4.2 Die Vorlagen-*Management* Beispielanwendung 8
4.2.1 Die Verwendung in einem *Business*-Service 8
4.2.2 Die Verwendung in der *Web*-Oberfläche 8

5 Die Analyse und Tests **9**

5.1 Die Tests 9
5.1.1 Die Tests der *Services* 9
5.1.2 Die Tests der *CDI*-Integration 9
5.1.3 Die Tests der *Web*-Oberfläche 9
5.2 Die erreichten Ziele 9
5.2.1 Das Vorlagen-*Management über* CKEditor 9
5.2.2 Das Vorlagen-*Management* in einer *CDI*-Umgebung . 9
5.2.3 Das Vorlagen-*Management* in JSF 9
5.2.4 Das Vorlagen-*Management* in *Mail*-DB-Schema 9

A Technische Informationen **10**
A.1 Aktuelle Dateiversionen 10
A.2 Details zur aktuellen Version 10
A.2.1 Allgemeine technische Voraussetzungen 10
A.2.2 Verwendung unter Windows 10
A.2.3 Verwendung unter Mac OS 11

Quellenverzeichnis **12**

Kurzfassung

TODO: Add german summary here

Abstract

TODO: Add english summary here

Kapitel 1

Einleitung

Die vorliegende Sachlage beschäftigt sich mit der Konzeption und Implementierung eines Vorlagen-*Management* für den in der theoretischen Bachelorarbeit konzipierten *Mail*-Service. Das Vorlagen-*Management* stellt einen essentiellen Teil des *Mail*-Service dar, mit dem sich parametrisierte *E-Mail*-Vorlagen erstellen lassen. Das Vorlagen-*Management* soll es den BenutzerInnen ermöglichen einfach eigene parametrisierte *E-Mail*-Vorlagen zu erstellen, die in einer Anwendung, die den *Mail*-Service nutzen, verwendet werden können, um benutzerspezifische *E-Mail*-Nachrichten zu versenden. Mit dem Vorlagen-*Management* ist es nicht mehr erforderlich die *E-Mail*-Vorlagen statisch zu definieren und die *E-Mail*-Vorlagen können von den Benutzerinnen nach ihren Wünschen angepasst werden.

1.1 Das Unternehmen curecomp Software Service GmbH

Diese Arbeit wird in Zusammenarbeit mit dem Unternehmen *curecomp Software Service GembH* erstellt. Das Unternehmen *curecomp* ist ein ein Dienstleister im *Supplier-Relationship-Management (SRM)* und betreibt eine eigene Softwarelösung namens *clevercure*. Die Softwarelösung *clevercure* besteht aus den folgenden Anwendungen:

- *CleverWeb* ist eine *Web*-Anwendung für den webbasierten Zugriff auf *clevercure*.
- *CleverInterface* ist eine Schnittstellen-Anwendung für den XML-basierten Datenimport /-export zwischen clevercure und den ERP-Systemen der Kunden.
- *CleverSupport* ist eine unternehmensinterne *Web*-Anwendung für die Abwicklung von *Support*-Prozessen.
- *CleverDocument* ist ein Dokumentenmanagementsystem für die Verwaltung aller anfallender Dokumente innerhalb von *clevercure*.

- *CCMail* ist die bestehende *Mail*-Anwendung für den Versand aller innerhalb *clevercure* anfallender *E-Mail*-Nachrichten.

Wie bereits in der theoretischen Bachelorarbeit behandelt, wird *CCMail* von *CleverMail* abgelöst werden, wobei dass in dieser Arbeit behandelte Vorlagenmanagement die Grundlage für *CleverMail* darstellt. Alle Anwendung innerhalb der Softwarelösung *clevercure* haben die Anforderung das *E-Mail*-Vorlagen parametrisiert und benutzerdefiniert erstellt werden können. Diese Anforderung wird mit dem Vorlagenmanagement erfüllt.

1.2 Das Vorlagenmanagement für den *Mail*-Service

Mit dem Vorlagenmanagement können *E-Mail*-Vorlagen einerseits von den EntwicklerInnen und BenutzerInnen benutzerdefiniert und parametrisiert erstellt werden. Damit können *E-Mail*-Vorlagen dynamisch auch zur Laufzeit erstellt, modifiziert und gelöscht werden. Es sind keine statischen *E-Mail*-Vorlagen mehr nötig und alle damit verbunden Nachteile wie z.B.

- das neu Kompilieren und Einspielen bei Änderungen der *E-Mail*-Vorlagen,
- keine Möglichkeit für benutzerdefinierten Vorlagen oder
- keine Möglichkeit der Nutzung von dynamischen Parametern in den *E-Mail*-Vorlagen

eliminiert werden. Das Vorlagenmanagement kann auch in einem anderen Kontext verwendet werden, wobei diese Arbeit sich ausschließlich mit der Verwendung des Vorlagenmanagement innerhalb des *Mail*-Service beschäftigen wird.
TODO: Add graphic about template management

1.3 Die Rahmenbedingungen

Das Vorlagenmanagement wird in Java in der Version 8 implementiert und wird sich an der *Java-Enterprise-Edition 7 (JEE-7)* Spezifikation orientieren, wobei folgende Teilspezifikationen Anwendung finden.

- *JPA 2.1* ist die Spezifikation für die Persistenz.
- *CDI 1.1* ist die Spezifikation für kontextabhängige Injektion innerhalb einer *JEE7*-Umgebung.
- *JSF 2.2* ist die Spezifikation der *View*-Technologie.

Damit wird das Vorlagenmanagement mit den aktuellsten Standards imple-

mentiert und wird daher für die Zukunft gut gewappnet sein. Die Funktionalität des Vorlagenmanagement wird weitestgehend ohne die Verwendung spezieller Bibliotheken implementiert, wobei folgende Integrationen zur Verfügung gestellt werden.

- *CDI*-Integration:
  Innerhalb eines *CDI-Containers* werden Objekte kontextabhängig zur Verfügung gestellt.
- *JSF*-Integration:
  Mit der *View*-Technologie *JSF* wird eine Webseite erstellt, über die die Vorlagen verwaltet werden können.
- *Typescript*-Integration:
  Mit *Typescript* wird ein *Plugin* für den *Rich-Editor CKEditor* implementiert, welches die Variablen für eine *E-Mail*-Vorlage innerhalb des *CKEditors* verwaltet.

Als Entwicklungsumgebung wird die *IDE Intellij* verwendet, die eine bekannte Entwicklungsumgebung im *Java*-Umfeld darstellt und ein Produkt des Unternehmens *Jetbrains* mit Sitz in Tschechien ist. Als Applikationsserver wird *Wildfly 10.x*, vormals *JbossAS* genannt, des Unternehmens *Redhat* verwendet, der ein zertifizierter *JEE-7*-Server ist und somit alle benötigten Spezifikationen unterstützt.

Kapitel 2

Das Ziel des Projekts

Ziel dieses Projekts ist die Konzipierung und Umsetzung des Vorlagenmanagement für den *Mail*-Service. Das Vorlagenmanagement stellt einen essentiellen Teil des *Mail*-Service dar und wird auch über Anwendungsgrenzen hinweg verwendet werden. Die verschiedenen Anwendungen werden auch unterschiedlichen Laufzeitumgebungen betrieben.

- *IBM-Integration-Bus (IIB)*
  ist ein proprietäres Produkt des Unternehmens IBM, für *XML*-Konvertierungen und *XML*-basierten Datenimport und -export.
- *Wildfly*
  ist ein zertifizierter *JEE-7* Applikationsserver des Unternehmens *Redhat*.

Die verschiedenen Anwendungen von *clevercure* sollen mit geringsten Aufwand in der Lage sein *E-Mail*-Vorlagen zu verwenden bzw. *E-Mails* auf Basis dieser Vorlagen zu erstellen und zu versenden. Dabei sollen die benötigten Abhängigkeiten der Anwendungen zu dem Vorlagenmanagement so klein wie möglich gehalten werden, sowie nur vorgegebene Schnittstellen verwendet werden.
Wird eine *E-Mail* aus einer Anwendung heraus auf Basis einer *E-Mail*-Vorlage erstellt, so müssen dessen enthaltene Parameter beim Zeitpunkt des Erstellens aufgelöst und serialisiert werden, damit die *E-Mail* mit denselben Daten zu jedem Zeitpunkt erneut versendet werden kann. Für die Anwendungen soll nicht ersichtlich sein wie die *E-Mails* nach ihrer Erstellung versendet bzw. verwaltet werden. Grundlegend sollen die Anwendungen, die das Vorlagenmanagement verwenden, so geringe Abhängigkeiten wie möglich zum Vorlagenmanagement besitzen.

2.1 Die funktionalen Ziele

Für das Vorlagenmanagement wurden folgende funktionalen Anforderungen definiert.

Die Persistenz der Vorlagen:
Die Vorlagen sollen über Anwendungsgrenzen hinweg innerhalb einer Datenbank persistent gehalten werden. Da das Vorlagenmanagement vorerst exklusiv für den *Mail*-Service verwendet wird, wird die Persistenz der Vorlagen innerhalb des *Mail*-DB-Schema realisiert.

Die Mehrsprachigkeit der Vorlagen:
Die Vorlagen sollen in mehreren Sprachen erstellt und verwaltet werden können, wobei eine Sprache immer als Standardsprache zu definieren ist, auf die zurückgefallen wird, wenn die gewünschte Sprache nicht vorhanden ist.

Die Mehrsprachigkeit der Vorlagenparameter:
Die zur Verfügung stehenden Vorlagenparameter werden durch die EntwicklerInnen spezifiziert und stellen einen Titel und eine Beschreibung des Parameters zur Verfügung. Der Titel und die Beschreibung des Vorlagenparameters sollen mehrsprachig zur Verfügung stehen, wobei als Standardsprache Englisch zu definieren ist.

Die Parametrierbarkeit der Vorlagen:
Die *E-Mail*-Vorlagen werden für einen bestimmten *Mail*-Typ definiert der einen bestimmten Kontext darstellt wie z.B.

- Ein Benutzer wurde erstellt,
- Der Zugang eines Benutzer wurde gesperrt oder
- Das Passwort wurde zurückgesetzt.

Für einen *Mail*-Typ bzw. einen Kontext, in dem eine *E-Mail*-Vorlage verwendet werden, sollen kontextabhängige Vorlagenparameter zur Verfügung stehen wie z.B.:

- *CURRENT_USER* ist der Benutzer der die *E-Mail* erstellt halt.
- *USER* ist der Benutzer, für den die *E-Mail* bestimmt ist.

Die EntwicklerInnen sollen für einen bestimmten Kontext in der Lage sein Parameter zu definieren. Die BenutzerInnen sollen innerhalb der definierten Menge von zur Verfügung stehenden Parametern frei wählen können und eine beliebige Anzahl der Parameter mit Wiederholungen der Parameter in der Vorlage verwenden dürfen.

Die automatische Registrierung der spezifizierten Vorlagenparameter:
Innerhalb einer *CDI*-Umgebung sollen die spezifizierten Vorlagenparameter beim Start des *CDI*-Containers automatisch aufgelöst und registriert werden. Dies ist über eine *CDI-Extension* realisierbar.

Die Verwaltung der Vorlagen über eine Webseite:
Die Vorlagen sollen über eine Webseite verwaltbar sein, wobei eine Vorlage explizit freigegeben werden muss bevor diese Vorlage verwendet wird. Die Webseite soll mit der *View*-Technologie *JSF* implementiert werden. Über einen *FacesConverter* soll die Vorlage von der *View*-Repräsentation in die Repräsentation der verwendeten *Template-Engine* überführt werden.

Da die Vorlagen formatierbar sein sollen wie z.B.: Schriftart und Schriftstil, soll der *Rich-Editor CKEditor* verwendet werden, da es für diesen *Javascript* basierten *Rich-Editor* bereits eine Integration in *JSF* gibt, die von *Primefaces-Extensions* zur Verfügung gestellt wird.

2.2 Die technischen Ziele

Folgende Punkte wurden als technische Ziele definiert.

Die Verwendung des Vorlagenmanagement in verschiedenen Umgebungen und Anwendungen
Die Kernfunktionalität des Vorlagenmanagement soll ohne Verwendung spezieller Bibliotheken implementiert werden. Diese Funktionalitäten sollen in den verschiedenen Umgebungen wie z.B. einer *CDI*-Umgebung integriert werden können. Innerhalb einer *CDI*-Umgebung können die benötigten Ressourcen vorkonfiguriert und kontextabhängig über Injektion zur Verfügung gestellt werden.

Das Vorlagenmanagement wird in mehreren Anwendungen verwendet, die jeweils unterschiedliche technische Anforderungen definieren, daher soll es jeweils eine eigene Integration pro Anwendung geben, die das Vorlagenmanagement und dessen zur Verfügung gestellten Funktionalitäten in die Anwendung integrieren.

Die Komponentenorientierte Entwicklung
Die einzelnen Softwarekomponenten des Vorlagenmanagement sollen so weit wie möglich unabhängig voneinander implementiert und über Schnittstellen aneinander gekoppelt werden. Damit soll die Modularität des Vorlagenmanagement gewährleistet werden, da es jeweils eigene In

Kapitel 3

Das Lösungskonzept

3.1 Die Spezifikation der Vorlagen-*API*

3.1.1 Die Schnittstellen und abstrakten Klassen

Interface 1

3.2 Die Spezifikation der Vorlagenintegration

3.2.1 Das Vorlagen-*Management* in Typescript und Javascript

3.2.2 Das Vorlagen-*Management* in CDI

3.2.3 Das Vorlagen-*Management* in JSF

3.2.4 Das Vorlagen-*Management* in *Mail*-DB-Schema

Kapitel 4

Die Realisierung

4.1 Die Implementierung der Spezifikationen

4.1.1 Die Implementierung für *CKEditor*

Das *CKEDitor-Plugin* in Typescript

Die Variablenrepräsentation in JSON

4.1.2 Die Implementierungen für CDI

Die Vorlagen-*Management CDI-Extension*

Der Vorlagen-*Management CDI-Producer*

Die Vorlagen-*Management CDI-Utility*

4.1.3 Die Implementierungen für JSF

Der Vorlagen *FacesConverter*

Die *Primefaces-Extension* für den *CKEditor*

4.2 Die Vorlagen-*Management* Beispielanwendung

4.2.1 Die Verwendung in einem *Business*-Service

4.2.2 Die Verwendung in der *Web*-Oberfläche

Kapitel 5

Die Analyse und Tests

5.1 Die Tests

5.1.1 Die Tests der *Services*

5.1.2 Die Tests der *CDI*-Integration

5.1.3 Die Tests der *Web*-Oberfläche

5.2 Die erreichten Ziele

5.2.1 Das Vorlagen-*Management über* CKEditor

5.2.2 Das Vorlagen-*Management* in einer *CDI*-Umgebung

5.2.3 Das Vorlagen-*Management* in JSF

5.2.4 Das Vorlagen-*Management* in *Mail*-DB-Schema

Anhang A

Technische Informationen

A.1 Aktuelle Dateiversionen

Datum	Datei
2015/09/19	`hgbthesis.cls`
2015/11/04	`hgb.sty`

A.2 Details zur aktuellen Version

Das ist eine völlig überarbeitete Version der DA/BA-Vorlage, die UTF-8 kodierten Dateien vorsieht und ausschließlich im PDF-Modus arbeitet. Der „klassische“ DVI-PS-PDF-Modus wird somit nicht mehr unterstützt!

A.2.1 Allgemeine technische Voraussetzungen

Eine aktuelle LaTeX-Installation mit

- Texteditor für UTF-8 kodierte (Unicode) Dateien,
- `biber`-Programm (BibTeX-Ersatz, Version $\geq$ 1.5),
- `biblatex`-Paket (Version $\geq$ 2.5, 2013/01/10),
- Latin Modern Schriften (Paket `lmodern`).[1]

A.2.2 Verwendung unter Windows

Eine typische Installation unter Windows sieht folgendermaßen aus (s. auch Abschnitt **??**):

1. **MikTeX 2.9**[2] (zurzeit am einfachsten die 32-Bit Version, da nur diese

[1] http://www.ctan.org/pkg/lm, http://www.tug.dk/FontCatalogue/lmodern

[2] http://www.miktex.org/ – **Achtung:** Generell wird die **Komplettinstallation** von MikTeX („Complete MiKTeX“) empfohlen, da diese bereits alle notwendigen Zusatzpakete und Schriftdateien enthält! Bei der Installation ist darauf zu achten, dass die automatische

das Programm `biber.exe` bereits enthält),

2. **TeXnicCenter 2.0**[3] (Editor-Umgebung, unterstützt UTF-8),
3. **SumatraPDF**[4] (PDF-Viewer),

Ein passendes TeXnicCenter-Profil für MikTeX, Biber und Sumatra ist in diesem Paket enhalten (Datei `_tc_output_profile_sumatra_utf8.tco`). Dieses sollte man zuerst über `Build` → `Define Output Profiles` in TeXnicCenter importieren. **Achtung**: Alle neu angelegten `.tex`-Dateien sollten in UTF-8 Kodierung gespeichert werden!

A.2.3 Verwendung unter Mac OS

Diese Version sollte insbesondere mit *MacTeX* problemlos laufen (s. auch Abschnitt **??**):

1. *MacTex* (2012 oder höher).
2. Die Zeichenkodierung des Editors sollte auf UTF-8 eingestellt sein.
3. Als Engine (vergleichbar mit den Ausgabeprofilen in TeXnicCenter) sollte *LaTeXMk* verwendet werden. Dieses Perl-Skript erkennt automatisch, wie viele Aufrufe von *pdfLaTeX* und *Biber* nötig sind. Die Ausgabeprofile *LaTeX* oder *pdfLaTeX* hingegen müssen mehrmals aufgerufen werden, zudem werden hierbei auch die Literaturdaten nicht verarbeitet. Dazu müsste extra die *Biber*-Engine aufgerufen werden, die jedoch noch nicht in allen Editoren vorhanden ist.

Installation erforderlicher Packages durch „*Install missing packages on-the-fly: = Yes*“ ermöglicht wird (NICHT „*Ask me first*“)! Außerdem ist zu empfehlen, unmittelbar nach der Installation von MikTeX mit dem Programm `MikTeX` → `Maintenance` → `Update` und `Package Manager` ein Update der installierten Pakete durchzuführen.

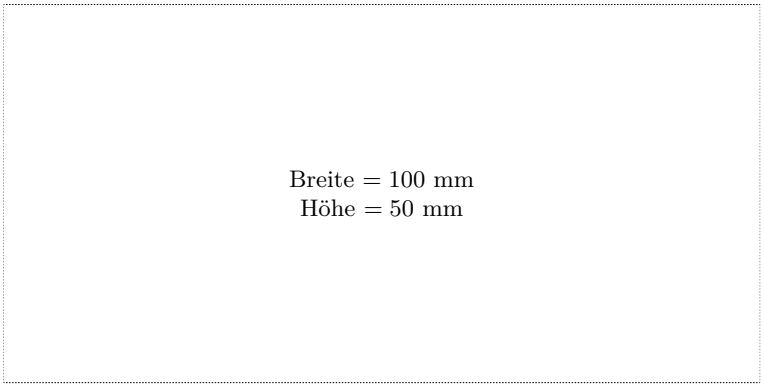
[3] http://www.texniccenter.org/

[4] http://blog.kowalczyk.info/software/sumatrapdf/

Quellenverzeichnis

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —