NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Praxisbericht Nr. 4

Prüfling (Name)	Niels Gundermann
Matrikel-Nr.	5023
Studiengang, Zenturie	Wirtschaftsinformatik, I11b
Thema	Praxisbericht: Prototypische Implementierung eines JavaFX/Web-Channels zur Integration ins Multichannel-Framework der deg
Ausbildungsbetrieb	data experts gmbH
Prüfer/in	Dr. Stephan Neuthe
E-Mail der/ des Prüfers/in	stephan.neuthe@data-experts.de
Datum der Themenausgabe	16.05.2014
Datum der Abgabe	12.06.2014

fristgerechte Abgabe[1]	Ja ☒	Nein ☐
Sperrvermerk[2]	Ja ☐	Nein ☒
bestanden[3]	Ja ☒	Nein ☐

Datum 4.12.2014 Unterschrift Prüfer/in

[1] Die Bearbeitungsdauer für Praxisberichte soll vier Wochen nicht überschreiten (vgl. Merkblatt Praxisberichte). Wenn der Praxisbericht nicht innerhalb der vereinbarten Frist abgegeben wurde, dann gilt die Studienleistung als nicht bestanden.
[2] Der Praxisbericht enthält interne, vertrauliche Daten, die nicht zur Weitergabe an Dritte bestimmt sind.
[3] Als Bewertung kommen nur „bestanden" oder „nicht bestanden" in Betracht. Prüfer nutzen hierfür das beigefügte Bewertungsschema.

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Eidesstattliche Erklärung des Studenten / der Studentin

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder von mir noch von jemand anderem als Prüfungsleistung vorgelegt. Der von mir angegebene Prüfer hat die Arbeit zur Kenntnis genommen und als „bestanden" bewertet.

Datum 04.12.2014 ..... Unterschrift Student/in .....

Köllner Chaussee 11, 25337 Elmshorn, Telefon 0 41 21/40 90-0, Telefax 0 41 21/40 90 40

Praxisbericht: Prototypische Implementierung eines JavaFX/Web-Channels zur Integration ins Multichannel-Framework der deg

Niels Gundermann

12. Juni 2014

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Die deg entwickelt eine Software names profil c/s. Dabei handelt es sich um eine Client-Server Anwendung zur Fördermittelverwaltung in der Landwirtschaft. Dieses Programm wurde speziell für die Umsetzung eines integrierten Verwaltungs- und Kontrollsystems (INVEKOS) in den Ämtern der Landwirtschaftsministerien entwickelt.
Dort werden zwei unterschiedlich Clients für die Arbeit mit profil c/s eingesetzt. Dabei handelt es sich einerseits um einen Standalone-Client und andererseits um einen Web-Client. Beide Clients sind Kundenanforderungen für deren Nutzung keine weiteren Installationen von Nöten sein sollen.
Um dieser Anforderung gerecht zu werden, muss die deg die Userinterfaces (UIs) für die Clients auf zwei unterschiedlichen Zielplattformen darstellen. Dabei kommen zwei Frameworks zum Einsatz. Für den Standalone-Client nutzt die deg das *Swing*-Framework. Bei der Visualisierung des Web-Clients kommt derzeit das wingS-Framework zur Anwendung. Das Problem dabei ist, dass beide Frameworks (Swing und wingS) veraltet sind. In dieser Arbeit wird der Einsatz des wingS-Frameworks mit dem Einsatz von JavaFX im Web und Vaadin verglichen. Dafür wurde jeweils ein Prototyp mit JavaFX und Vaadin implementiert.

2. Grundlegendes

2.1. Multichannel-Framework

Die beiden Clients die bei profil c/s zum Einsatz kommen, haben unabhängig von der Zielplattform den gleichen Aufbau. Die beiden dafür eingesetzten GUI-Frameworks fordern unterschiedliche Implementierungen für diesen Aufbau. Um den Aufwand für die Implementierung beider UIs gering zu halten, hat die deg ein eigenes Mutlichannel-Framework (MC-Framework) entwickelt. Dabei handelt es sich um ein Rahmenwerk, mit dessen Hilfe eine UI einmal implementiert wird und über unterschiedliche Kanäle auf unterschiedlichen Zielplattformen dargstellt werden kann. Die Kanäle nutzen dabei jeweils ein GUI-Framework.
Folgenden Abbildungen ist die Integration der einzelnen Kanäle (hier der Swing- und der wingS-Kanal)[1] zu entnehmen. Einfach erklärt arbeitet der Entwickler bei der Umsetzung einer UI mit Präsentationsformen. Durch Umgebungseinstellungen wird zur Laufzeit ent-

[1] ULC: Ultra Light Client (ist bei profil c/s nicht mehr im Einsatz)

schieden, über welchen Kanal die Ausgabe stattfinden soll.
Die einzelnen Kanäle implementieren eine ComponentFactory, welche die entsprechenden Wrapper zu den Präsentationsformen erzeugt. Die Wrapper deligieren die Aufrufe, die in den Präsentationsformen stattfinden, an die entsprechenden Komponenten des jeweiligen Frameworks weiter.

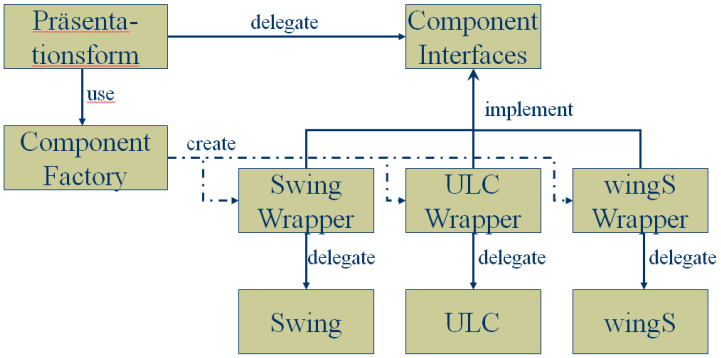

Abbildung 1: *Multichannel* - UI-Komponenten Schema [Maa07]

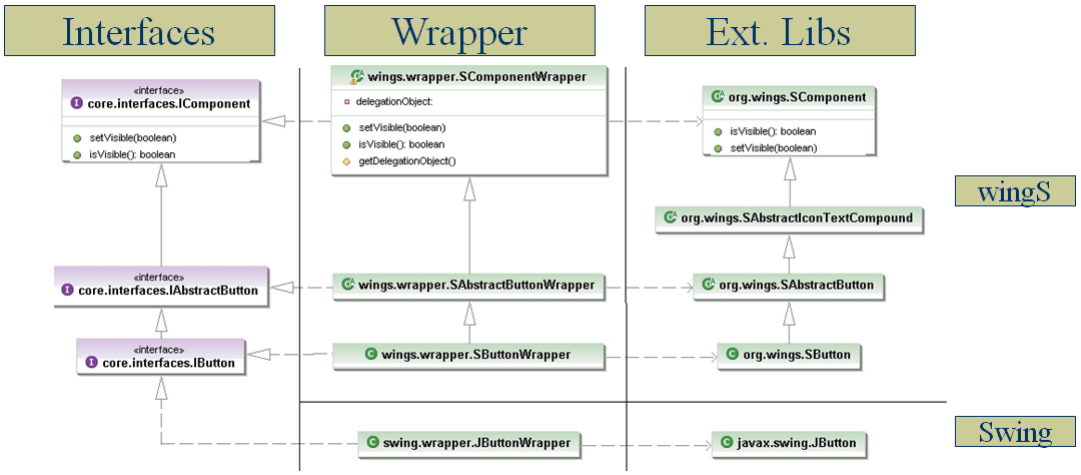

Abbildung 2: Integration von unterschiedlichen Frameworks in das MC-Framework [Maa07]

2.2. wingS

Bei wingS handelt es sich um ein Open-Source Web-Framework. Wie eingangs schon erwähnt, ist dieses Framework veraltet. Die letzten Updates für wingS stammen aus dem Jahr 2008. Seitdem wurde das Framework nicht weiterentwickelt. Das größte Problem beim Umgang mit wingS ist der Mangel an Informationen zur Nutzung des Frameworks. Abbildung 3 ist die Architektur wingS-Frameworks und die Kommunikation zwischen Clienten und Server zu entnehmen.

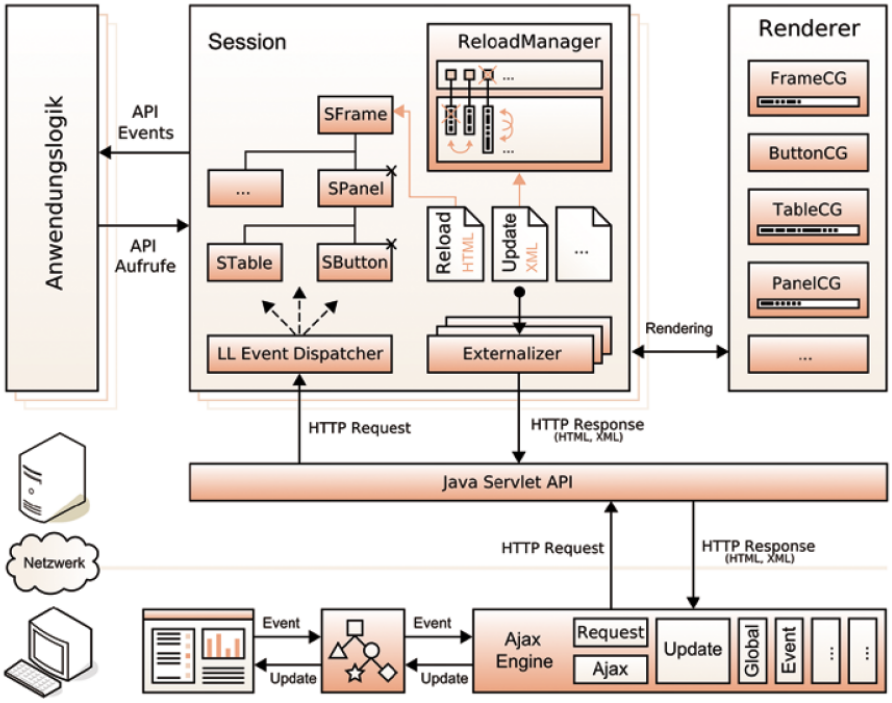

Abbildung 3: wingS-Architektur [Sch08]

Die deg hat das wingS-Framework für ihre Zwecke erweitert. Vor allem Darstellungen für Tabellen und Layouts (v.a. GridBagLayout) wurden angepasst.
Das Problem vor dem die deg steht, ist dass die Anforderungen und Maßstäbe im Web-Bereich immer weiter wachsen und wingS neue Web-Technologien[2] nicht mehr unterstützt. Somit besteht die Gefahr, dass künftige Anforderungen bzgl. der Visualisierung nicht erfüllt werden können.
Um den Web-Client von profil c/s mit solchen Anforderungen in Zukunft umsetzen zu können, wird ein Framework mit folgenden Eigenschaften gesucht.

- Aktueller Web-Client kann umgesetzt werden
- Integration in das MC-Framework ist möglich
- Neue Web-Technologien werden unterstützt

[2]Bspw. jQuery, Flash, HTML5, RSS

3. JavaFX

3.1. Allgemeines

JavaFX ist ein von Oracle entwickeltes Framework für Rich Internet Applications, welches in der Version 1.0 am 4. Dezember 2008 veröffentlicht wurde. Die aktuelle Version 2.2 wurde im August 2012 freigegeben. Es handelt sich dabei um ein Framework, welches laut Oracle in Lage ist, sowohl als Standalone- als auch als Web-Variante zu fungieren. Die Architektur wird mit Abbildung 4 verdeutlicht.

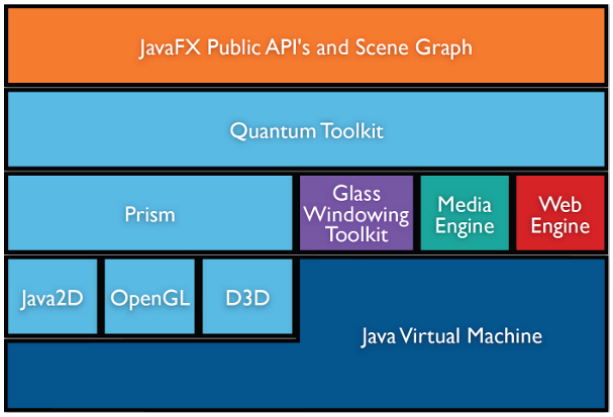

Abbildung 4: JavaFX-Architektur [Gru13]

Die Darstellung im Web kann über ein Java-Applet realisiert werden [Ora13a]. Die Vor- und Nachteile bei der Nutzung von Java-Applets sind ansatzweise in einer Tabelle in Abbildung 5 zusammengefasst.

Vorteile
J2SE-API ist im vollen Umfang nutzbar
komplexe Anwendungen in Zusammenarbeit mit Servlets und Applikationsservern möglich
In unterschiedlichen Browsern nutzbar
Nachteile
Sicherheitsrichtlinen müssen herabgesetzt werden
Java-Script muss aktiviert sein (Sicherheitsrisiko)
Abgleich der Java-Versionen notwendig
lange Initialisierung der JVM

Abbildung 5: Vor- und Nachteile von der Nutzung des Java-Applets [Ame11, Ora13b]

Für die Nutzung von Applets muss der Anwender mit einem Browser arbeiten, in dem das Java-Plugin installiert werden kann. [Ora14] Weiterhin notwendig für die Ausgabe der JavaFX-GUI über ein Applet im Browser, ist eine entsprechende html-Datei, in der eine jnlp-Datei mittels Java-Script eingebunden wird.[3] Abbildung 6 stellt die Kommunikation zwischen Applet und Java-Plugin im Browser dar.

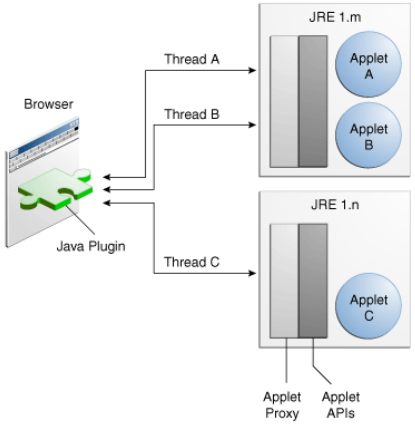

Abbildung 6: Applet's Execution Enviorment [Ora14]

3.2. Beispiele für die Nutzung

Der Aufbau einer JavaFX-UI wird in Abbildung 7 verdeutlicht.

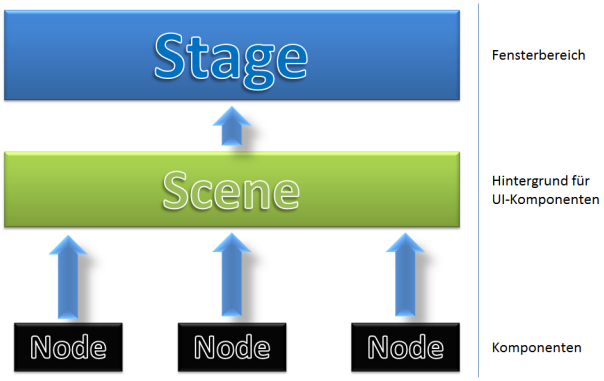

Abbildung 7: JavaFX: Allgemeiner Aufbau einer grafischen Oberfläche

Die Stage bildet die Abgrenzung zu anderen Komponenten, die auf der Zielplattform dargestellt werden. Die Scene ist der Bereich, in dem UI-Komponenten (Node) erstellt werden können.

Im Praxisbericht *Prototypische Implementierung eines JavaFX-Channels zur Integration*

[3] Beispiele sind im Anhang B zu finden

ins MultiChannel-Framework der deg wurde ein Prototyp für einen Standalone-Client mit JavaFX implementiert. Der Code kann weitesgehend für einen Web-Client übernommen werden. Aus diesem Grund sind Informationen zu der Implementierung der GUI-Komponenten der genannten Arbeit zu entnehmen.
Der Prototyp für den Web-Client, welcher bei der Bearbeitung dieser Arbeit entstand, befindet sich im Anhang B.
Bei der Umsetzung des Prototypen ist aufgefallen, dass sich die Handhabung mit mehreren Fenstern zum Standalone-Client unterscheidet. Beim Standalone-Client wird mit mehrern Stages gearbeitet. Ein neues Fenster wird dabei mit dem Aufruf *new Stage()* erzeugt.
Im Browser gibt es nur ein Fenster. Das was beim Standalone-Client mit unterschiedlichen Stages realisiert wird, muss im Web-Client mittels Tabe realisiert werden.[4] Neue Stages dürfen nicht erzeugt werden.

```
1 @Override
2 public void start(Stage primaryStage) throws Exception {
3                 root = new TabPane();
4                 scene = new Scene(root, 800, 500);
5                 BorderPane borderPane = new BorderPane();
6                 borderPane.setTop(new MenuPane());
7                 borderPane.setCenter(new BearbeitungsPane(currentMappe));
8                 borderPane.setBottom(new Statusbar(currentMappe));
9                 Tab tab = new Tab(currentMappe.getTitel());
10                tab.setContent(borderPane);
11                root.getTabs().add(tab);
12                primaryStage.setTitle("JavaFXSamlpe - Profil");
13                primaryStage.setScene(scene);
14                primaryStage.show();
15 }
```

3.3. Ansatz zur Integration in das *MC*-Framework

Zur Integration von JavaFX in das MC-Framework ist es notwendig, die entsprechende ComponentFactory und Wrapper-Klassen zu implementieren. Die Wrapper-Klassen müssen in der ComponentFactory angemeldet werden. Abbildung 11 zeigt, die Einordnung eines JavaFX-Channels in das *MC*-Framework neben anderen Kanälen (Neuerungen sind blau gekennzeichnet).
Beispielhaft wird die Komponente *javafx.scene.control.Button* in das MC-Framework eingebunden. Das resultierende Diagramm ist Abbildung 9 zu entnehmen (Neuerungen sind

[4] Wird in bestehenden Web-Client (wingS) auch mit Tabe realisiert.

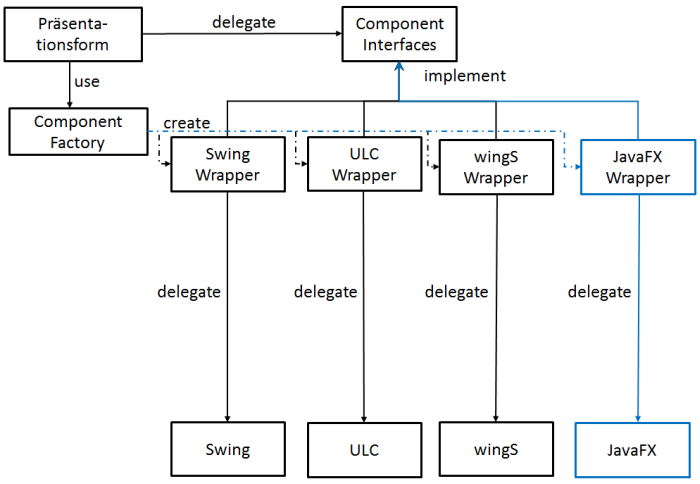

Abbildung 8: Integration von JavaFX in das UI-Komponenten Schema des MC-Framework

blau gekennzeichnet. Swing-Komponenten zum Vergleich sind gelb gekennzeichnet). Dies ist mit der Integration der wingS-Komponente *org.wings.SButton* vergleichbar. (Abbildung 2)

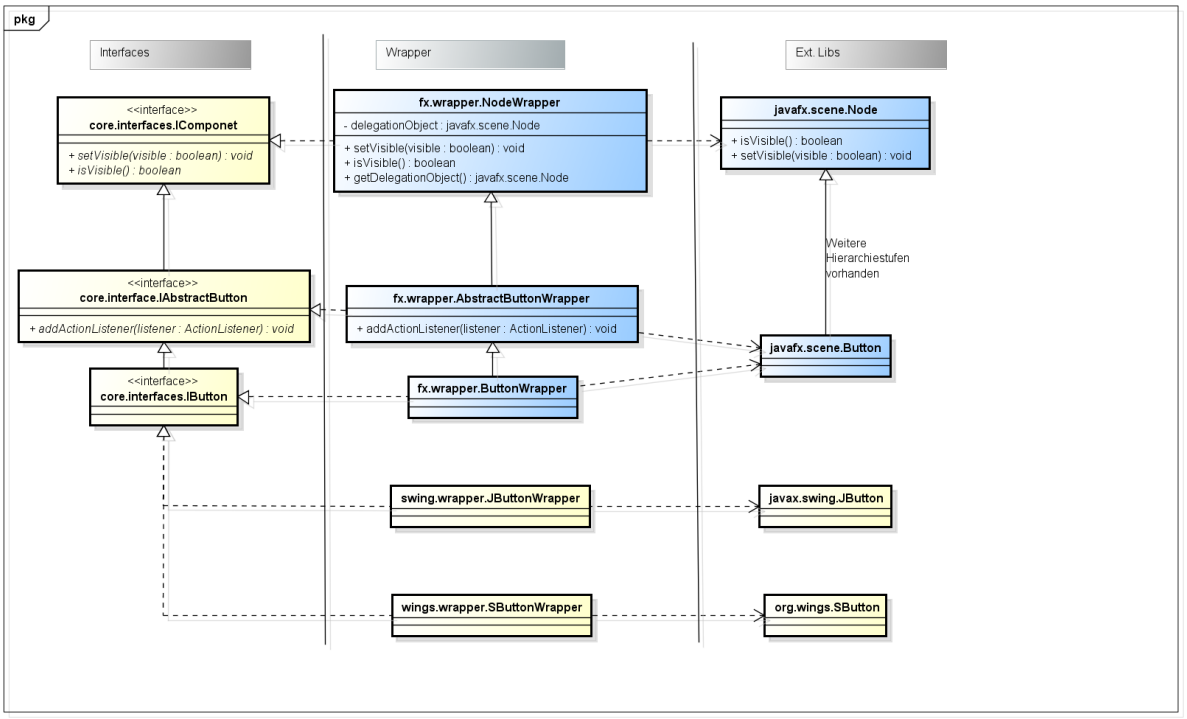

Abbildung 9: Integration des JavaFX-Buttons in das *MC*-Framework

3.4. Einschätzung

Der große Mehrwert des Einsatzes von JavaFX ist, dass die Implementiertungen für Web- und Standalone-Client kaum Unterschiede aufweisen. Beide Clients könnten in großen Teilen mit einem Framework realisiert werden. Die Integration in das MC-Framework ist theoretisch möglich. Jedoch müssen Darstellungen von Tabellen und Layouts angepasst werden.
Allerdings fordert die Nutzung von Applets die Installation des Java-Plugins, was laut Anforderung der Kunden unerwünscht ist.

4. Vaadin

4.1. Allgemeines

Das Vaadin-Framework ist ein Open-Source Framework, welches es erlaubt UIs für Web-Oberflächen zu implementieren. Dies kann auf zwei Wegen geschehen. Zum einen durch Java-Code und zum anderen durch einen Editor [Mar13][S. 237ff.].
Der Service bzw. die Unterstützung ist durch verständliche Tutorials, eine gute Dokumentation, sowie entsprechende Plugins für unterschiedliche Entwicklungsumgebungen gegeben. Dadurch findet man leicht einen Zugang zu dem Framework. Des Weiteren stellen die Vaadin-Entwickler ihr Know-How als externe Berater zur Verfügung.
Vaadin kann auf allen Servern bereitgestellt werden, die einen Servlet- oder Portlet-Container enthalten. Beispiele dafür sind Tomcat, Glasfish oder der in der deg eingesetzte JBoss. Die Entwicklung findet auf dem Server statt. Abbildung 10 zeigt den Aufbau des Vaadin-Frameworks sowie die Kommunikation zwischen Server und Client.

4.2. Beispiele für die Nutzung

Der Vaadin-Prototyp ist im Anhang B zu finden. Zur Implementierung der Menüleiste bietet Vaadin einen entsprechend vorgefertigten MenuBar-Typ an. Dieser kann mit Elementen gefüllt werden, wie es auch schon von JavaFX bekannt ist.

```
1 MenuBar menu = new MenuBar();
2 MenuBar.MenuItem schliessen = menu.addItem("Schließen", null);
```

Um Aktionen an diesem Menü ausführen zu können, muss auch bei Vaadin ein entsprechender Listener implementiert.

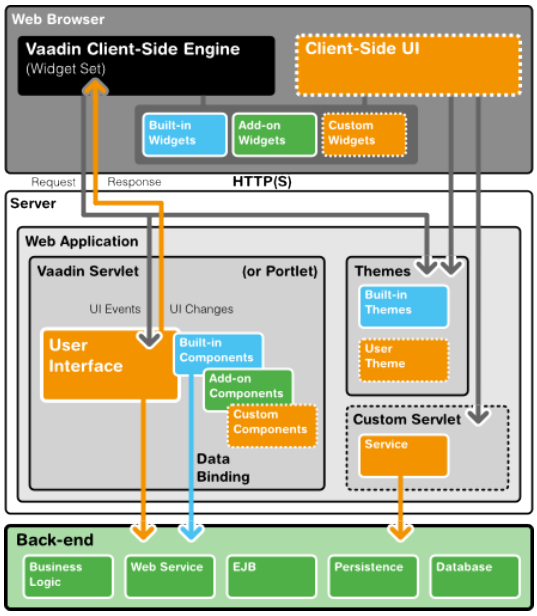

Abbildung 10: Vaadin-Architektur [Mar13]

```
1 public class SchliessenCommand implements Command {
2         private MappenView vertragsMappenView;
3
4         @Override
5         public void menuSelected(MenuItem selectedItem) {
6                 vertragsMappenView.getUI().close();
7         }
8 }
9 });
```

Für die Umsetzung geteilter Layouts bietet Vaadin zwei Komponenten an, die je nach Orientierung der Teilungsrichtung verwendet werden. Dazu gehört das VerticalSplitPanel - für die vertikale Teilung - und das HorizontalSplitPanel - für die horizontale Teilung. Voreinstellung für die Aufteilung der geteilten Bereiche können durch die Methode *setSplitPosition(VALUE, UNIT)* getroffen werden. Die UNIT bestimmt dabei, ob es sich um einen absoluten, oder um einen prozentualen Wert handelt.

```
1 VerticalSplitPanel vertikalerSplit = new VerticalSplitPanel();
2 vertikalerSplit.setSplitPosition(30, Sizeable.Unit.PERCENTAGE);
3 vertikalerSplit.setFirstComponent(Component);
4 vertikalerSplit.setSecondComponent(Component);
```

Bei der Darstellung von Tabellen werden die Spalten bei Vaadin als Container betrachtet. Das hat den Vorteil, dass je Spalte auch festgelegt werden kann, welcher Datentyp dort angezeigt wird. Das geschieht mit der Methode *addContainerProperty(TEXT, TYPE, DEFAULT)*.

```
Table table = new Table();
table.addContainerProperty("Vorgang", String.class, "");
table.addContainerProperty("Status", String.class, "");
table.addContainerProperty("Zuwendungs-\nsumme[EUR]", String.class, "");
table.addContainerProperty("Zahlungs-\nbetrag[EUR]", String.class, "");
table.addContainerProperty("Zahlungs-\ndatum", String.class, "");
initLines(table);
...

private void initLines(Table table) {
        int i = 0;
        for(Document teilvorgang : teilvorgaenge){
                TeilvorgaeneTableData data = new TeilvorgaeneTableData(teilvorgang);
                table.addItem(new Object[]{ data.getVorgang(), data.getStatus(),
                data.getZuwendungssumme(), data.getZahlungsbetrag(), data.getZahlungsdatum()}, i++);
        }
}
```

Bei dem Verweise- und Inhaltsbaum handelt es sich um eine Eltern-Kind-Beziehung. Beim Einfügen von Elementen in den Baum muss das Elternelement explizit gesetzt werden.

```
private void appendDocuments(Document doc, Tree tree) {
        for (Document children : doc.getUnterDokumente()) {
                tree.addItem(children.getTitel());
                tree.setParent(children.getTitel(), getRootItem(tree));
                tree.setChildrenAllowed(children.getTitel(), false);
        }
}
```

Bei den Click-Events für die Bäume ist folgende Codezeile besonders wichtig.

```
setImmediate(true);
```

Das hat zur Folge, dass das Event sofort an den Server geschickt wird. Ansonsten erfolgt das Senden erst, wenn der Thread in der Methode *access()* der Vaadin-Klasse UI beendet wurde. [Mar13]

Probleme entstanden bei der Umsetzung der Toolbar. Hierfür gibt es keine brauchbare Standardkomponente, wie man es von anderen UI-Frameworks gewohnt ist. Es gibt zwar Toolbar-Add-Ons, die in das Projekt eingebunden werden können [Jon, Jou]. Diese werden jedoch nur bis zur Version 6 unterstützt.

Die Toolbar für diesen Prototypen wurde wie folgt implementiert:

```
HorizontalLayout toolBar = new HorizontalLayout();
String basepath = VaadinService.getCurrent().getBaseDirectory().getPath();
FileResource imageDrop = new FileResource(new File(basepath+"/img/TbCopy.gif"));
FileResource imagePrint = new FileResource(new File(basepath+"/img/TbPrint.gif"));
Button btDrop = new Button();
btDrop.setIcon(imageDrop);
```

```
7 Button btPrint = new Button();
8 btPrint.setIcon(imagePrint);
9 btPrint.setEnabled(false);
10
11 toolBar.addComponents(btDrop, btPrint, btLossOrg, btGetOrg, btHelp);
```

Die Tab-Ansicht für den Web-Client wird durch ein TabSheet realisiert. Diese Komonente ist global zugänglich, da sie bspw. durch Aktionen, die von den Tree-Controller ausgehen, verändert werden kann.

```
1 static TabSheet tabsheet = new TabSheet();
2 VertragsMappenView vertragsMappe = new VertragsMappenView(null);
3 // Parameter ist die Bezeichnung der zu oeffnenden Mappe
4 // null = Standard-Mappe
5 tabsheet.addTab(vertragsMappe).setCaption(vertragsMappe.getTitle());
6 tabsheet.getTab(vertragsMappe).setClosable(true);
```

4.3. Ansatz zur Integration in das MC-Framework

Die Integration von Vaadin in das MC-Framework gestaltet sich analog zu der Integration von JavaFX. Die Wrapper-Klassen müssen implementiert werden und in der Component-Factory angemeldet werden. Abbildung 11 zeigt, die Einordnung eines Vaadin-Channels in das MC-Framework neben anderen Kanälen (Neuerungen sind blau gekennzeichnet).

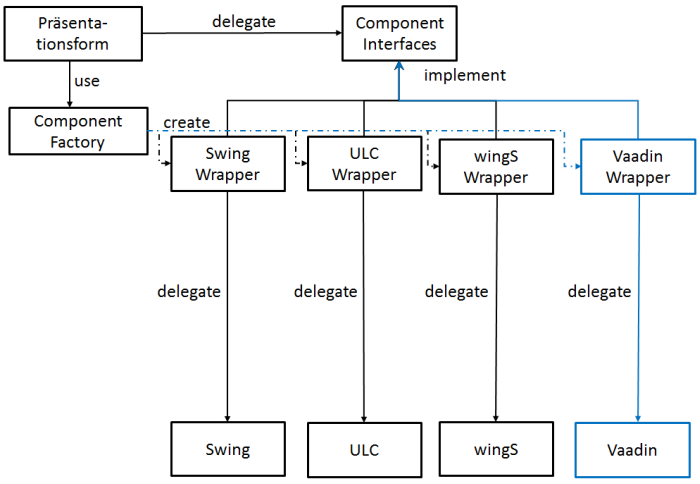

Abbildung 11: Integration von *Vaadin* in das UI-Komponenten Schema des *MC*-Framework

Beispielhaft wird die Komponente *com.vaadin.ui.Button* in das MC-Framework eingebunden. (Neuerungen sind blau gekennzeichnet. Swing-Komponenten zum Vergleich sind gelb gekennzeichnet.)
Dies ist wiederum mit der Integration der wingS-Komponente *org.wings.SButton* vergleichbar. (Abbildung 2)

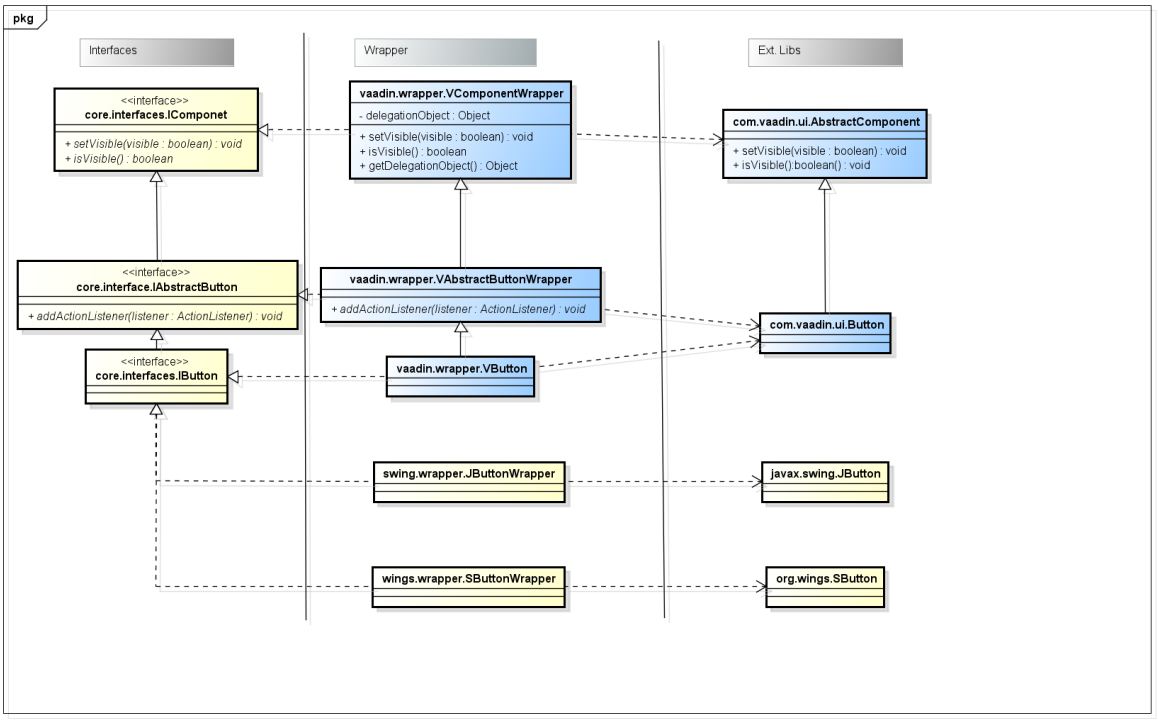

Abbildung 12: Integration des *Vaadin*-Buttons in das *MC*-Framework

4.4. Einschätzung

Vaadin ist ein sehr aktuelles Web-Framework, für das es immer wieder Updates und Erweiterungen gibt. Der Support, der durch die Community und die Entwickler gewährleistet wird, ist sehr groß. Der Ansatz zur Integration in das MC-Framework ist in der deg bereits umgesetzt. Allerdings gab es auch dabei Probleme bei der Darstellung von Tabellen und Layouts.

5. Fazit

wingS ist zwar veraltet, aber funktioniert für profil c/s den Anforderungen entspreched. Sollten die Anforderungen bzgl. der Visualisierung wachsen, wird die deg viel Arbeit damit haben, Funktionalitäten mit wingS umzusetzen, die in anderen Frameworks schon vorgegeben sind. Der Mangel an Informationen wird diese Umsetzungen, trotz des großen Know-Hows bzgl. wingS, erschweren.

JavaFX wird in der deg für den Web-Client nicht zum Einsatz kommen. Grund dafür ist die Notwendigkeit des Java-Plugins im Browser. Das würde eine lokale Installation voraussetzen, was nicht den Anforderungen an den Web-Client entspricht.

Vaadin eignet sich als Framework für einen neuen Channel für das MC-Framework der

deg. Dennoch müssen Anpassungen gemacht werden, von denen nicht sicher ist, ob sie die Anforderung mindestens im gleichen Umfang erfüllen, wie es der wingS-Channel tut.
Folgender Abbildung ist eine Gegenüberstellung der vorgestellten Frameworks zu entnehmen. Dabei werden die Wartung und Erweiterung des Frameworks, der Support, die Machbarkeit der Integration in das MC-Framework und die Abdeckung bestehender Funktionen betrachtet.

Kriterium	**wingS**	**JavaFX**	**Vaadin**
Wartung und Erweiterungen	-	+	+
Support	-	+	+
Machbarkeit d. Integration	+	o	o
Abdeckung best. Funktionen	+	-	o

Abbildung 13: Gegenüberstellung: wingS - JavaFX - Vaadin

In Bezug auf Vaadin ist zu beachten, dass viele Neuerungen gibt. Aufgrund dessen sollte das Framework weiter beobachtet werden.

Anhänge

A. Literaturverzeichnis

[Ame11] America, Oracle: *Java Virtual Machine Specification - Chapter 5. Loading, Linking, and Initializing.* URL: http://docs.oracle.com/javase/specs/jvms/se7/html/jvms-5.htmljvms-5.5, Juli 2011. [Online, eingesehen 30. März 2014].

[Gru13] Grunwald, Gerrit: *Visualisierung in Java mit JavaFX.* URL: http://www.heise.de/developer/artikel/Visualisierung-in-Java-mit-JavaFX-1902233.html, Juni 2013. [Online, eingesehen 29. April 2014].

[Jon] Jonathan Nash: *Toolbar.* URL: https://vaadin.com/directoryaddon/toolbar. [Online; eingesehen 12. März 2014].

[Jou] Jouni Koivuviita: *ToolbarWindow.* URL: https://vaadin.com/directoryaddon/toolbarwindow. [Online; eingesehen 12. März 2014].

[Maa07] Maass, Dirk: *JWAMMC.* Das Multichannel-Framework der data-experts gmbh, 2007.

[Mar13] Marko Grönroos: *Book of Vaadin - Vaadin 7 Edition - 1st Revision.* Vaadin Ltd, 2013. 598 S.

[Ora13a] Oracle: *JavaFx Documentation Home: Deploying JavaFX Applications.* URL: http://docs.oracle.com/javafx/2/deployment/jfxpub-deployment.htm, 2013. [Online; eingesehen 30. August 2013].

[Ora13b] Oracle: *Oracle Deployment Tutorial.* URL: http://docs.oracle.com/javase/tutorial/deployment/jar/intro.html, 2013. [Online; eingesehen 30. August 2013].

[Ora14] Oracle: *Oracle Deployment Tutorial.* URL: http://docs.oracle.com/javase/tutorial/deployment/applet/index.html, 2014. [Online; eingesehen 28. Januar 2013].

[Sch08] Schmid, Benjamin: *Get your wingS back!* Javamagazin, 2–7, Januar 2008.

B. Implementierung

- Java-Projekt: JFXprofilSample (jfxprofilsample.zip)
- Java-Projekt: VaadinProfilSample (vaadinprofilsample.zip)

Integration einer JavaFX-Anwendung in ein Applet

HTML-Datei

```
<html><head>
  <SCRIPT src="./web-files/dtjava.js"></SCRIPT>
<script>
    function launchApplication(jnlpfile) {
        dtjava.launch(            {
                url : 'FXprofilSample.jnlp',
              },
            {
                javafx : '2.2+'
            },
            {}
        );
        return false;
    }
</script>

<script>
    function javafxEmbed() {
        dtjava.embed(
            {
                url : 'FXprofilSample.jnlp',
                placeholder : 'javafx-app-placeholder',
                width : 800,
                height : 600,
            },
            {
                javafx : '2.2+'
            },
            {}
        );
    }
    <!-- Embed FX application into web page once page is loaded -->
    dtjava.addOnloadCallback(javafxEmbed);
</script>
```

```
35
36 </head><body>
37   <!-- Applet will be inserted here -->
38   <div id='javafx-app-placeholder'></div>
39 </body></html>
```

JNLP-Datei

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <jnlp spec="1.0" xmlns:jfx="http://javafx.com" href="FXprofilSample.jnlp">
3   <information>
4     <title>FXprofilSample</title>
5     <vendor>Niels Gundermann</vendor>
6     <description>null</description>
7     <offline-allowed/>
8   </information>
9   <resources>
10    <jfx:javafx-runtime version="2.2+" href=
11    "http://javadl.sun.com/webapps/download/GetFile/javafx-latest/windows-i586/javafx2.jnlp"/>
12  </resources>
13  <resources>
14    <j2se version="1.6+" href="http://java.sun.com/products/autodl/j2se"/>
15    <jar href="JavaFXApplication2.jar" size="41820" download="eager" />
16  </resources>
17  <applet-desc  width="800" height="600"
18  main-class="com.javafx.main.NoJavaFXFallback"  name="FXprofilSample" >
19    <param name="requiredFXVersion" value="2.2+"/>
20  </applet-desc>
21  <jfx:javafx-desc  width="800" height="600" main-class="helper.Starter"  name="FXprofilSample" />
22  <update check="always"/>
23 </jnlp>
```

C. Abbildungen

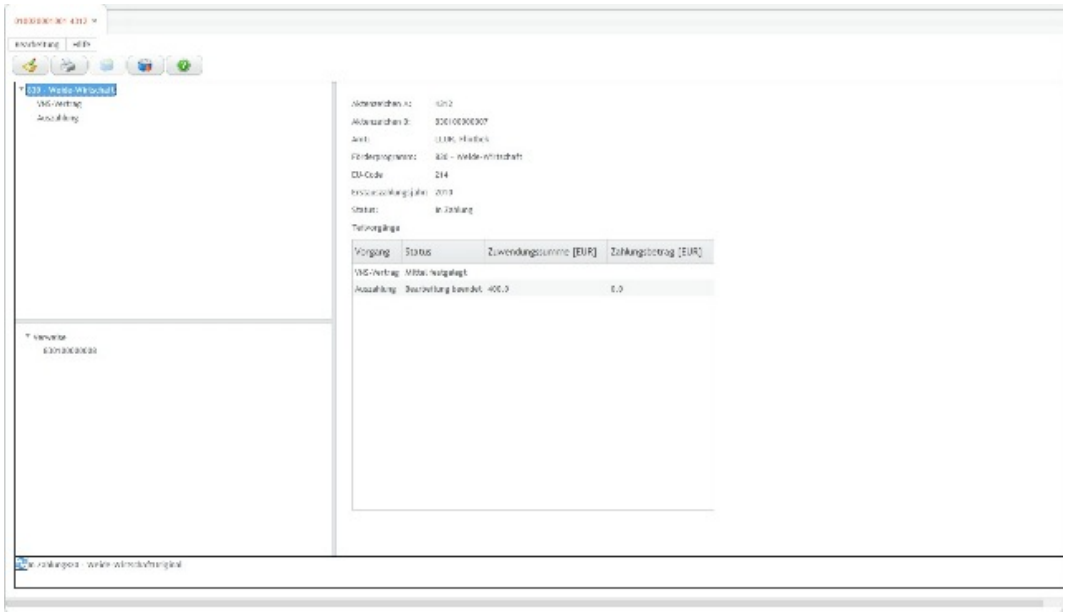

Abbildung 14: Prototyp - Vaadin