SmartWebAppComponents

Dokumentation

Version 0.3.0

Stand: 05.12.2022

Inhalt

[Theoretische Grundlagen 5](#_Toc111277500)

[InformationsOrientierteProgrammierung (IOP) 5](#_Toc111277501)

[Deklarative Sprachen 5](#_Toc111277502)

[DataBinding 5](#_Toc111277503)

[Konzepte 6](#_Toc111277504)

[Komponenten 6](#_Toc111277505)

[Schichten 6](#_Toc111277506)

[Model-Bucket-View-Work 7](#_Toc111277507)

[Model 7](#_Toc111277508)

[Bucket 8](#_Toc111277509)

[View 8](#_Toc111277510)

[Work 10](#_Toc111277511)

[Deklarative Verwendung 10](#_Toc111277512)

[Options-Mechanismus 10](#_Toc111277513)

[Visuelle Komponenten 10](#_Toc111277514)

[Nebenläufiges laden visueller Komponenten 11](#_Toc111277515)

[Custom Templates 11](#_Toc111277516)

[Algorithmen Komponenten 11](#_Toc111277517)

[Adaptive Source Binding 11](#_Toc111277518)

[Deklarative Nennung 12](#_Toc111277519)

[Datenquellen 12](#_Toc111277520)

[Datensparsamkeit 12](#_Toc111277521)

[Parameter für Datenabfragen 13](#_Toc111277522)

[Zugriffsstati 13](#_Toc111277523)

[Teilen und Verteilen 13](#_Toc111277524)

[Lazy-Databinding 13](#_Toc111277525)

[WatchableSource-Konsistenz 14](#_Toc111277526)

[Visuelle Komponenten 17](#_Toc111277527)

[Navigation 17](#_Toc111277528)

[User 17](#_Toc111277529)

[Viewer 17](#_Toc111277530)

[Editor 17](#_Toc111277531)

[Translator 17](#_Toc111277532)

[Algorithmische Komponenten 18](#_Toc111277533)

[Implementierung 19](#_Toc111277534)

[View-Bind-Model-Work 19](#_Toc111277535)

[Model 21](#_Toc111277536)

[Visuelle Komponenten 21](#_Toc111277537)

[Strukturansicht und –bearbeitung 22](#_Toc111277538)

[Informationsansicht und -bearbeitung 23](#_Toc111277539)

[Personalisierungen 27](#_Toc111277540)

[Strukturansicht und –bearbeitung 28](#_Toc111277541)

[Informationsansicht und -bearbeitung 28](#_Toc111277542)

[Personalisierungen 30](#_Toc111277543)

[Administration 32](#_Toc111277544)

[Installation 32](#_Toc111277545)

[Installation (Manuell): 32](#_Toc111277546)

[Deaktivierung 32](#_Toc111277547)

[Konfiguration 32](#_Toc111277548)

[Entwicklung 33](#_Toc111277549)

[Algorithmen 33](#_Toc111277550)

[*Informationsverarbeitung* 33](#_Toc111277551)

[Dokumentenbasiert 33](#_Toc111277552)

[Identifizierung 33](#_Toc111277553)

[Assoziierung 33](#_Toc111277554)

[Objektisierung 34](#_Toc111277555)

[Attributisierung 34](#_Toc111277556)

[Relationalisierung 34](#_Toc111277557)

[Kontexterkennung 35](#_Toc111277558)

[Abstraktion 35](#_Toc111277559)

[Kategorisierung 36](#_Toc111277560)

[Versionshistorie 37](#_Toc111277561)

[Testfälle 38](#_Toc111277562)

[Literaturverzeichnis 39](#_Toc111277563)

# Theoretische Grundlagen

## InformationsOrientierteProgrammierung (IOP)

Die InformationsOrientierteProgrammierung bietet einen Ansatz um die Entwicklung von Anwendungen zu vereinfachen, bei denen mit einer Menge von dynamisch erweiterbaren Daten gearbeitet wird. Grundsatz ist die Annahme, das alles als eine Information formuliert werden kann und das Algorithmen unabhängig von einer konkreten Information oder Klasse von Objekten implementiert werden können. Dadurch wird eine höhere Abstraktion und Wiederverwendbarkeit erreicht, was späteren Anpassungsaufwand erheblich reduziert. Dafür ist ein anfänglich höherer Implementierungsaufwand notwendig.

SWAC wird als Anwendung für das IOP Konzept implementiert und ist Demonstrations- und Forschungsobjekt für die Verwendung der IOP. Sie wurde zeitgleich entwickelt um eine hohe Vollständigkeit der IOP zu erreichen und konzeptuelle Fehler, sowie Fehler in der Implementierung zeitnah zu entdecken. Die Trennung von Daten und Algorithmen wird hier logisch fortgeführt und um die Komponente der Visualisierung ergänzt. Dabei wird auch bei der Visualisierung darauf geachtet, dass die visuellen Elemente von den Algorithmen, welche die Visualisierung aufbauen getrennt bleiben. Dadurch soll eine einfache Anpassung der Darstellung der Komponenten erreicht werden.

SWAC ist so strukturiert, dass die Komponenten auch ohne tiefgehende Kenntnisse des IOP-Konzepts als Darstellungselemente verwendet werden können.

## Deklarative Sprachen

Bei der Verwendung von deklarativen Programmiersprachen wird so programmiert, dass man sagt, was man für Informationen haben möchte, nicht wie man diese bekommt. Das Wie wird den in der Sprache implementierten Algorithmen überlassen.

Dieses Konzept wurde auf die Anzeigekomponenten übertragen, der Programmierer sagt, welche Informationen er mit welcher Ansicht dargestellt haben möchte und welche Funktionalitäten zur Verfügung stehen sollen, jedoch nicht, wie das Ziel erreicht werden soll.

## DataBinding

Unter DataBinding versteht man den Mechanismus, Datenobjekte an GUI Elemente unter Verwendung des Observer-Patterns zu koppeln. Dies hat den Vorteil, dass die GUI Elemente bei Änderungen an den Daten diese veränderten direkt darstellen können.

In SWAC wurde ein Komponentenübergreifendes DataBinding implementiert. Finden Änderungen in einem Datensatz einer Komponente statt, werden diese Änderungen auch anderen Komponenten bekanntgegeben. Dieser Mechanismus geht somit über das DataBinding anderer Frameworks hinaus, welche ein DataBinding nur auf Ebene der Anzeigeelemente vorsehen.

# Konzepte

## Komponenten

SmartWebApplicationComponents besteht aus einer Reihe von Komponenten, die unabhängig voneinander verwendet werden können und eine gemeinsame Kernfunktionalität nutzen und zur Verfügung stellen.

Eigenschaften sind:

* Deklarative Einbindung

Komponenten werden der Webapplikation durch deklarative Anweisungen hinzugefügt

* Automatisches Auffinden

Die Verwendung einer Komponente wird durch den Interpreter automatisch festgestellt

* Automatisches Laden benötigter Ressourcen

Für eine Komponente benötigte Ressourcen werden durch den Interpreter automatisiert bereitgestellt.

* Sparsam im Verbrauch von Datenvolumen

Es wird niemals mehr Programmcode geladen, als unbedingt erforderlich.

TODO: Dokumentieren dass custom… Funktionen und Funktionen in der Componenten-Hierarchie. Custom Funktionen immer im Kontext der Componente.

## Schichten

Anwendungsseiten

Anwendungsfall-Schicht

REST-Schnittstelle

Komponenten

Abstraktions-Schicht

Daten- zugriff / -alg.

Kernkomponenten

Allgemein-Schicht

Implementierungsaufwand

* Anwendungsfall-Schicht

Diese Schicht enthält lediglich Programmcode, der für Aussehen und Verhalten der Anwendung für den Anwendungsfall sorgt. Z.B. Konfigurationen der Komponenten, HTML Seiten, spezielles CSS, Event-Handler die nur im Rahmen der Anwendung sinnvoll sind. Programmcodes, die Komponenten untereinander für eine Zusammenarbeit verbinden

* Abstraktions-Schicht

In dieser Schicht sind die Komponenten zur Anzeige und die Algorithmen realisiert. Ihr Programmcode ist unabhängig vom Anwendungsfall. Eine Komponente stellt eine Ansicht und Funktionalitäten zur Verfügung die über die Daten und Optionen gesteuert werden.

* Allgemein-Schicht

In dieser Schicht werden allgemeine Mechanismen implementiert. Dies Umfasst das Grundsystem zur Erkennung von Komponenten, automatisches Einbinden, Datenzugriffe und Datenverwaltung.

## Model-Bucket-View-Work

Das Model-Bucket-View-Work ist im Wesentlichen eine Weiterentwicklung des Model-View-ViewModel-Musters. Jedoch gibt es zahlreiche Unterschiede in der Aufgabenverteilung. Insbesondere steht die Trennung von Daten, View und Logik im Mittelpunkt.

View, Model und Work kennen sich untereinander nicht. Lediglich die Buckets kommunizieren die Daten untereinander.

View

Bucket

Model

Work

Ließt Daten zur Anzeige

startet

Schreibt Daten von der Eingabe

läd

liefert Daten

aktualisiert Werte

### Model

Das Model kümmert sich ausschließlich um Daten. Es wird keinerlei Anwendungslogik im Model implementiert. Es organisiert das Laden der Daten aus unterschiedlichen Quellen, wie JavaScript Objekten, JSON-Dateien und REST-Schnittstellen.

Aufgaben des Models:

* Laden der Daten von einer der definierten Datenquellen
* Transformation der Daten in das interne Datensatzformat
* Sicherstellung das Daten nicht doppelt geladen werden
* Sicherstellung, dass Datensätze nicht doppelt geladen werden
* Verteilung der Datensätze auf die Komponenten
* Synchronisation der Datensätze unter den Komponenten (über zentrale Buckets)

Dies ist eine unmittelbare Weiterführung des IOP Konzepts und erlaubt es, dass das Model generisch verwendet werden und nicht für einzelne Komponenten angepasst werden muss. Jedes JavaScriptObjekt kann als Datenquelle für Komponenten im SWAC genutzt werden, unabhängig davon, ob sie ursprünglich dafür implementiert wurden oder nicht. Funktionale Attribute werden für das Model ignoriert.

### Bucket

Ein Bucket enthält alle Datensätze, die für die zugeordnete Komponente von Interesse sind. Die Buckets einer Komponente enthalten somit die Datensätze, die in dieser Komponente verarbeitet werden sollen. Ein Bucket kann mit einer einzelnen Information (bei objektorientierter Betrachtung als Objekt und bei relationaler Betrachtung als Datensatz betrachtet) oder einer Liste von Informationen (entsprechend einer Objektliste) gefüllt sein.

Beispiel für eine einzelne Information als JavaScriptObjekt modelliert:

let info = { id=1, title=“Geschichte“, content=“Dies ist ein Blindtext“};

Beispiel für eine Liste von Informationen:

let infos = [

{ id=1, title=“Geschichte“, content=“Dies ist ein Blindtext“},

{ id=2, title=“Neuere Geschichte“, content=“Dies ist ein Blindtext“}

]

Eine Komponente kann mehrere Datenquellen gleichzeitig verwenden. Dazu besitzt jede Komponente für jede Datenquelle ein Bucket. Außerdem existiert für jede Datenquelle genau ein Bucket im Model.

Komponente 1

Bucket/Komp. 1

„Stammdaten“ 

Komponente 2

Bucket/Komp. 2

„Stammdaten“

Model

Bucket/Model

„Stammdaten“

### View

Der View besteht aus den visuellen Elementen, welche für die Darstellung einer Information notwendig sind. Der View muss dafür weder den Bind noch das Model oder den Work kennen. Er besteht aus einer Datei mit einem HTML-Fragment, welches die Ansicht strukturiert und gegebenenfalls einer Stylesheet-Datei zur Formatierung der enthaltenen Elemente. Mit Hilfe von Markern werden die Plätze für das Einfügen der Attribute festgelegt. Somit ist der View leicht veränder- und austausch-bar.

Ein View wird verwendet, indem eine Instanz von ihm deklariert wird. Mehr dazu im nächsten Abschnitt.

#### Content-Marker

Die Marker werden durch den Interpreter in bindPoints umgewandelt, die zunächst ungebunden zur Verfügung stehen um an Attribute aus dem Model gekoppelt zu werden.

Beispiel für ein einfaches HTML-Fragment mit Marker, der in einen bindPoint transformiert wird:

<p>{content}</p>

#### Allgemeiner Marker

Für manche Anwendungsfälle ist es egal, welches Attribut einer Information an welcher Stelle ausgegeben wird, in diesem Fall kann der allgemeine Marker genutzt werden, welcher an jedes beliebige Attribut gebunden werden kann.

Beispiel für den allgemeinen Marker:

<p>{\*}</p>

#### RepeatForValue

Sollen mehrere Werte mit gleichem Namen dargestellt werden (aus unterschiedlichen Informationen), so können beliebige Strukturelemente als wiederholbar für Werte gekennzeichnet werden. Damit ist es möglich gleiche Werte gleich darzustellen.

Beispiel für einen wiederholbares HTML-Element, welches für jeden Wert im Datensatz einen Absatz anlegt:

<p class=“swac\_repeatForValue“>{\*}</p>

#### RepeatForSet

Häufig sollen alle Werte eines Datensatzes in einer bestimmten Form repräsentiert werden und jeder neue Datensatz soll auf die gleiche Art dargestellt werden. Für diesen Anwendungsfall können beliebige HTML-Elemente als wiederholbar für Datensätze gekennzeichnet werden.

Wiederholbare Elemente für Datensätze können mit wiederholbaren Elementen für Werte kombiniert werden.

Beispiel für die Erstellung eines Containers für Werte eines Datensatzes:

<div class=“swac\_repeatForSet“>

<h1>{title}</h1>

<p>{content}</p>

</div>

#### RepeatForChild

Datensätze können Hauptdatensätze eines View sein, oder Kinddatensätze (die eine Verbindung zu einem Hauptdatensatz besitzen). Diese Kinddatensätze können innerhalb des repeatForSet separat aufgelistet werden.

<div class=“swac\_repeatForSet“>

<h1>{title}</h1>

<p>{content}</p>

<div class=“swac\_repeatForChild“>{childsValue}</div>

</div>

#### Design by Example

Die vorgenannten Techniken des View ermöglichen es, die Webanwendung mittels Design by Example zu entwerfen. Bereits bei der Ansicht ohne Datenanbindung ist erkennbar, wie die Seite nachher aussehen wird, notwendige Elemente werden für Datensätze oder für Werte gegebenenfalls wiederholt.

### Work

Der Work bildet das Pendant zu den Algorithmen in der IOP. Hier befindet sich die Geschäftslogik, nicht jedoch die Logik die zur Steuerung der Anzeige oder zur Behandlung von Interaktionsereignissen notwendig ist. Dadurch ist auch die Geschäftslogik leicht austauschbar und dem Konzept der IOP folgend sind Algorithmen nicht auf die Verwendung mit einer festgelegten Objektklasse beschränkt.

## Deklarative Verwendung

Die Informationsorientierung und der hohe Grad der Abstraktion ermöglichen es, Programme mit einer deklarativen Sprache zu formulieren. Diese trägt wesentlich zur einfachen und schnellen Verwendung von SWAC und seinen Komponenten bei. Für die Verwendung einer Komponente ist es nicht mehr erforderlich bestimmte Auszeichnungen an HTML Elementen zu setzen, Funktionen an diese zu binden oder Funktionen aufzurufen. Es muss nur noch gesagt werden, dass die Komponente verwendet werden soll und mit welchen Daten.

Die Verwendung von Komponenten erfolgt dabei nach dem Schema:

[Komponente] FROM [Datenquelle]

Beispielsweise reicht es aus, folgendes Statement zu schreiben, um eine Liste von Informationen (egal welcher Art) darzustellen:

<div id=“instanzid“ swac=“swac\_list FROM objects“></div>

Wobei objects ein Array von Informationen ist.

### Options-Mechanismus

Müssen oder können Einstellungen für Komponenten gesetzt werden, so werden diese über den allgemeinen Options-mechanismus übergeben. Dieser generische Mechanismus ermöglicht es Komponenten mit Optionen auszustatten, ohne dass dies bei jeder Komponente neu implementiert werden muss.

Optionen werden als JavaScriptObjekt oder als Datenquelle gespeichert. Standardmäßig werden die Optionen in einer globalen Variable die den Namen des

## Visuelle Komponenten

View und Bind werden jeweils für eine Aufgabe spezialisiert zu Komponenten zusammengefasst. Eine Komponente übernimmt die Darstellung und Interaktivitätshandlung, inklusive der Plausibilitätsprüfung. Der Bind sorgt dabei dafür, dass die Daten im View und Model konsistent bleiben.

Eine visuelle Komponente besteht somit aus einer html-Datei mit der View (HTML-Fragment) und einer JavaScriptDatei mit dem Bind, gegebenenfalls gehört noch ein Stylesheet dazu. Die View wird als Design by Example implementiert. Der Bind besteht im einfachsten Fall lediglich aus dem Anstoßen des Bind-Mechanismus, hat jedoch die Möglichkeit eigenen Funktionen zu implementieren, die nach dem grundsätzlichen Bind ausgeführt werden.

Beispiel für den einfachsten Bind einer Komponente:

// Create data binding

let bind = SWAC\_bind.bind(requestor, SWAC\_present);

// Wait for view and bind to load

bind.then(function (bindetPoints) {

//Nothing to implement here, because the view-bind-model mechanism

// inserts everything into the provided template

});

### Nebenläufiges laden visueller Komponenten

Damit die Benutzer ein angenehmes Nutzungserlebnis erwartet sollten alle Komponenten so implementiert werden, dass sie nebenläufig geladen werden. Dies geschieht einfach durch das Einfassen des Bind-Aufrufs und des ggf. vorhandenen Code des Binds in eine Promise-Funktion.

Der Interpreter stellt automatisch einen entsprechenden Platzhalter bereit, der nach vollständigem Laden der Komponente ausgeblendet wird.

### Custom Templates

Alle Komponenten werden mit einer Standard-View Implementierung versehen. Soll ein speziell angepasstes HTML-Fragment für die View verwendet werden, so kann dies einfach in das deklarative swac-Tag eingeschlossen werden.

Beispiel für die Verwendung eines Custom Templates:

<div swac=“swac\_present FROM data“>

<p>Dies ist mein Custom Template</p><i>[title]</i><p>[content]</p>

</div>

## Algorithmen Komponenten

Algorithmen Komponenten besitzen ebenfalls Buckets für ihre Datenquellen und stellen Methoden für die Arbeit mit den Daten zur Verfügung. Sie können auch Daten in Buckets verändern und veränderte Daten an das Model zur Speicherung übergeben. Sie besitzen jedoch keine Darstellungsmechanismen.

## Adaptive Source Binding

Als Adaptive Source Binding wird die Möglichkeit bezeichnet, dass sich der Bind Mechanismus, die zu bindenden Daten selbstständig organisiert.

Das Adaptive Source Binding hat dabei folgende Ziele:

* Datensparsamkeit

Es werden nur Daten abgefragt, welche zur Erfüllung der Aufgabe benötigt werden.

* Einsparung von Transferkosten

Wenn möglich, werden günstigere Datenquellen, vor teureren bevorzugt

* ApplicationCaching

Einmal abgerufene Daten werden in einem lokalen Speicher hinterlegt, um sie bei späteren Zugriffen schneller und günstiger zur Verfügung zu haben

### Deklarative Nennung

Bei der deklarativen Nutzung einer Komponente gibt man mit der FROM-Klausel an, welche Daten verwendet werden sollen. Die Angabe bezieht sich auf den Namen der globalen Variable, den Key eines Eintrages im lokalen Speicher oder dem Identifizierungsmerkmal der Datensätze in einer externen Datenquelle.

### Datenquellen

SWAC verwendet flexibel globale Variablen, lokale Speicher oder entfernte Dienste als Quelle für die zu verwendenden Daten. Auch externe Datenquellen können in der Konfiguration hinterlegt werden. Diese externen Datenquellen müssen der Spezifikation nach IOP-REST implementiert sein.

Welche Datenquelle verwendet wird, entscheidet der Interpreter nach dem folgenden Algorithmus:

1. Prüfe, ob eine globale Variable mit dem Namen existiert
   1. Wenn ja: lade die Daten aus der globalen Variable in den Komponenten-Datenspeicher.
   2. Weiter mit Schritt 2
2. Prüfe, ob ein Schlüssel zu einem Eintrag im lokalen Speicher mit dem angegebenen Namen existiert
   1. Wenn ja: lade die Daten aus dem lokalen Speicher und führe sie mit den bereits im Komponenten-Datenspeicher vorhandenen Daten zusammen.
   2. Weiter mit Schritt 3
3. Frage alle externen Datenquellen, ob sie Daten unter dem Identifizierungsmerkmal zur Verfügung stellen. Wenn bereits Daten im Komponenten-Datenspeicher vorhanden sind, wird übermittelt, wann der letzte Datenabgleich stattgefunden hat.
   1. Neu verfügbare Daten werden dem Komponenten-Datenspeicher hinzugefügt
4. Der Komponenten-Datenspeicher wird im lokalen Speicher gecached.

Einmal über eine Netzwerkschnittstelle abgerufene Daten werden in einem lokalen Speicher der Anwendung nachgehalten, um Transfehrkosten zu sparen. Bei darauffolgenden Aufrufen, werden nur noch die Veränderungen abgefragt und im lokalen Speicher aktualisiert.

### Datensparsamkeit

Unter der Datensparsamkeit versteht man, dass nur so viel Daten angefragt werden, wie benötigt werden. Daten werden von Komponenten benötigt. Die Komponenten müssen somit die Information liefern, welche Daten von ihnen benötigt werden.

#### Ermittlung benötigter Daten visueller Komponenten

Die visuellen Komponenten erzeugen bei ihrer Instanziierung eine Liste von (noch ungebundenen) bindPoints, aus diesen lassen sich die benötigten Attribute ermitteln. Außerdem können an Views Anzeigebedingungen geknüpft werden, die ihrerseits Attribute benötigen. Auch diese Information kann verwendet werden, um die vom View benötigten Attribute zu ermitteln.

#### Ermittlung benötigter Daten algorithmischer Komponenten

### Parameter für Datenabfragen

Manche Schnittstellen benötigen Parameter, anhand derer die gelieferten Daten eingeschränkt werden können. Damit kann auch die Datenabfrage optimiert werden, indem unnötige Daten nicht abgefragt werden. Die Parameter können direkt in der deklarativen Anfrage für eine Komponente mit angegeben werden.

Beispiel für die Nutzung von Parametern:

<div swac=“Present FROM data WHERE id=2 AND title=hallo“

### Zugriffsstati

Bei der Verwendung mehrerer Datenquellen können einige der Datenquellen evtl. keine Daten unter der gewünschten Bezeichnung liefern. In diesem Fall wird die Schnittstelle mit 404 antworten. Um unnötige Anfragen bei Folgeaufrufen zu vermeiden, merkt sich das Model welche Daten es von wo nicht erhalten konnte und ignoriert diese Datenquellen bei der nächsten Abfrage.

Der Cache für die Zugriffsstati kann gelöscht werden, indem beim Seitenaufruf der Parameter clearsyncstates=true mit übergeben wird.

## Teilen und Verteilen

Der Work-Teil übernimmt nicht einfach die Ausführung der Algorithmen, er ist wie das Algorithmenregister des IOP dafür zuständig die passenden Algorithmen zu finden. Auch die Entscheidung ob ein clientseitiger oder ein serverseitiger Algorithmus verwendet werden soll, liegt in seinem Bereich. Dazu besitz er die Anbindung an das serverseitige Algorithmenregister.

## Lazy-Databinding

Das Databinding muss für SWAC nicht explizit im Model definiert werden, wie dies beispielsweise im Knockout-Framework der Fall ist. Wird ein Model an den Bind übergeben, damit dieser es an eine bestimmte View bindet, werden die für das Binding notwendigen Mechanismen automatisch im Objekt installiert. Alle Änderungen über Views oder den Work werden mittels dieser Mechanismen erfasst. Änderungen von Außerhalb des Framework können ebenfalls über die Mechanismen erfolgen. Das hat zwar den Nachteil, dass Änderungen entsprechend nicht, oder (bei Verwendung eines aktiven Polling-Mechanismus) nur mit Verzögerung erkannt werden, es bietet jedoch den Vorteil, dass bestehende Programmstrukturen einfach verwendet werden können. Letztlich sollen aber alle Algorithmen über den Work ausgeführt werden, womit die konsistente Änderung sichergestellt ist.

model

view

## WatchableSource-Konsistenz

Jede Komponente besitzt für jede Datenquelle, die es verwendet, eine WatchableSource. In dieser werden alle Datensätze, welche die Komponente verwendet, gespeichert. Eine andere Komponente kann exakt diese Menge an Datensätzen verwenden, oder eine kleinere oder eine größere Menge.

Zusätzlich gibt es für jede Datenquelle eine zentrale Model-WatchableSource, welche alle derzeit im Programm verwendeten Datensätze enthält.

Daten der Datenquelle „Tabelle 1“

Komponente 1

Datensätze 1 - 10

Komponente 2

Datensätze 5 - 20

Model – Alle Datensätze

Wird über eine Komponente ein Datensatz hinzugefügt oder entfernt, informiert die WatchableSource der Komponente zunächst die zentrale Model-WatchableSource. Die Model-WatchableSource informiert dann ihrerseits alle WatchableSources der Komponenten über die Änderung. Die Komponenten-WatchableSource fragt bei der Komponente nach, ob diese den angebotenen Datensatz haben möchte und nimmt ihn bei sich auf, wenn die Komponente dies bestätigt.

Daten der Datenquelle „Tabelle 1“

Komponente 1

Datensätze 1 - 10

Komponente 2

Datensätze 5 - 20

Model-Bucket – Alle Datensätze

1. Datensatz hinzufügen

2. Model-WatchableSource informieren

3. Andere WatchableSource informieren

4. Komponente informieren

# Visuelle Komponenten

In diesem Kapitel werden die visuellen Komponenten in ihrem Konzept vorgestellt. Alle visuellen Komponenten bestehen aus einer html-Datei, in welcher ausschließlich die visuelle Struktur definiert wird (insbesondere auch keine event-Handler registriert werden), sowie ggf. einer zugehörigen css-Datei mit Formatanweisungen und dem Controller-Javascript.

In der HTML Datei werden alle Elemente, in welche der Controller Daten einfügen kann, mit einer Klasse die mit swac\_ beginnt und den Namen den Komponente beinhaltet, gekennzeichnet.

## Navigation

Mit der Navigations-Komponente können Navigationsmenüs, sowie einzelne Navigationslinks erzeugt werden.

## User

## Viewer

## Editor

## Translator

# Algorithmische Komponenten

In diesem Kapitel finden Sie die Konzepte und Beschreibungen der algorithmischen Komponenten, welche Daten verarbeiten und neue Daten erzeugen oder bestehende modifizieren.

# Implementierung

In diesem Kapitel wird die Implementierung der einzelnen Programmbausteine näher beschrieben.

## View-Bind-Model-Work

|  |  |  |  |
| --- | --- | --- | --- |
| View | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die allgemeinen Funktionalitäten einer View werden hier implementiert. Die konkrete Komponente kann anschließend diese View-Funktionalität zusammen mit einem Bind nutzen. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Abgeschlossen | | | |
| **Anmerkungen Entwickler** | | | |
| * View-Basisobjekt 13.06.2018 * Einlesen der HTML-Fragmente 13.06.2018 * Platzieren der HTML-Fragmente 13.06.2018 * Einlesen und Einbinden Komponenten-CSS 14.06.2018 * Extraktion von BindPoints aus Attributen des HTML-Fragments 15.06.2018 * Extraktion von BindPoints aus Textteilen des HTML-Fragments 15.06.2018 * Erkennung von BindPoints aus BindPoint-Tags 16.06.2018 * Listen von BindPoints zusammenführen 15.06.2018 * Update(value) für BindPoints 15.06.2018 * \*-BindPoints für alle nicht im Template direkt benannten Daten (16.06.2018) * Custom-Templates innerhalb des Bind-Elements (16.06.2018) * Wiederholbare Bereiche für Datensätze (18.06.2018) * Ausgaben für Metadaten (19.06.2018) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Model | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Das Model-Objekt verfügt über eine Methode, um ein beliebiges Javascript-Objekt zum Model zu transformieren. Die nicht-funktionalen Attribute werden als Datenattribute fungieren. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Basis-Model Klasse 13.06.2018 * Adaptive Source Binding   + Finden der Daten aus einer client-seitigen, globalen Variable (13.06.2018)   + Finden der Daten aus dem localStorage (21.06.2018)   + Finden von Daten aus Datenquellen     - Abrufen einer Liste von Informationen (21.06.2018)     - Abrufen einer Information mit id (21.06.2018)     - Abrufen von Informationen mit Parametern (21.06.2018)     - Abrufen mit Angabe eines Abgleichdatums (21.06.2018)   + Speichern einmal geladener Daten im lokalen Speicher (21.06.2018)   + Merken wenn Daten über eine Datesource nicht verfügbar sind und vermeiden das es immer wieder neu versucht wird. (21.06.2018)   + Nur laden von Attributen, die von der Komponente benötigt werden     - Entsprechend teilweise geladene Objekte im lokalen Speicher * Lazy-Databinding   + Umwandlung von Daten-Objekten in Model bzw. datasets (14.06.2018)   + set(name,value) zum Setzen eines Attributwertes (14.06.2018)   + addObserver() zum Hinzufügen eines beobachtbaren Objekts (14.06.2018) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Bind | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Der Bind läd View und Model einer Komponente und bindet die Daten des Model an den View. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Basis-bind Objekt (13.06.2018) * BindPoints vom View holen (13.06.2018) * Data vom Model holen (13.06.2018) * BindPoints und Data zusammenbringen (16.06.2018) * Repeatable-Elemente finden und BindPoints duplizieren (16.06.2018) * BindPoints in der View mit Wert füllen (16.06.2018) * BindPoints als Observer des model registrieren (16.06.2018) * Registrierung von Funktionen für Anzeige-Bedingungen (15.06.2018) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Work | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Der Work stellt die Anbindung zu datenverarbeitenden Algorithmen dar. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Basis Work-Objekt | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Deklarative Einbindung | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Was-orientierten Tags stellen den Einstiegspunkt für die Nutzung von Komponenten dar. Hier wird in einem speziellen Attribut angegeben, was für eine Komponente man aus welchen Daten erzeugen möchte. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * swac.js als einzige Voraussetzung zur Nutzung des Frameworks (12.06.2018) * Einbindung benötigter Kern-Komponenten Model,View,Bind,… (12.06.2018) * swac.js Algorithmus zur Erkennung von Component-Tags durch Vorhandensein des Attributs swac (12.06.2018) * Extraktion des Komponentennamens (12.06.2018) * Extraktion des Namens des Datenobjekts (12.06.2018) * Nebenläufiges Laden der Komponenten (12.06.2018) | | | |

### Model

Daten Laden über Model:

Model.load()

-> Component.addSet()

-> WatchableSource.addSet() (Bucket von Component)

-> notifyObservers()

-> WatchableSource.addSet() (Bucket von Model)

-> notifyObservers()

-> Component.afterAddSet()

-> Component.customAfterAddSet()

Information eines Buckets:

-> WatchableSource.notifyAddSet()

-> Component.checkAccepted()

-> WatchableSource.addSet()

## Visuelle Komponenten

|  |  |  |  |
| --- | --- | --- | --- |
| Navigation | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Aufbau einer Seitennavigation aus einem Datensatz von Navigationsobjekten. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * HTML-Fragment für responsive HTML Menü (13.06.2018) * Menü platzierbar in Kopf oder Fuß (15.06.2018) * Anzeige-Bedingungen:   + From-Attribut [name] erlaubt Anzeige auf der Seite [name] (15.06.2018)   + From-Attribut \* erlaubt Anzeige auf allen Seiten (15.06.2018)   + Nicht passendes From-Attribut verhindert Anzeige (15.06.2018)   + Nur erreichbare Links werden angezeigt * Options-Mechanismus (11.07.2018) | | | |

## Strukturansicht und –bearbeitung

|  |  |  |  |
| --- | --- | --- | --- |
| Datenstrukturübersicht | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Datenstrukturübersicht bietet die Möglichkeit, sich die Struktur der Daten übersichtlich anzeigen zu lassen. Es werden Kind-Informationsarten und Eltern-Informationsarten angezeigt. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige aller Kind-Informationsarten (12.05.2015) * Anzeige aller Eltern-Informationsarten * ~~Anzeige aller Geschwister-Informationsarten~~ * ~~Hinzufügen neuer Eltern- oder Kind-Informationsarten~~ * (Entfällt weil: Automatisch durch das Hinzufügen einer Information als Kind oder Eltern-Information) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Datenstrukturkarte | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| In einer MindMap kann durch die Informationsarten navigiert werden. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige aller Kind-Informationsarten * Anzeige aller Eltern-Informationsarten | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsstrukturübersicht | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| In der Informationsstrukturübersicht kann die Struktur einer Information (die Dimensionen aus denen sie sich zusammensetzt) betrachtet und bearbeitet werden. | | | |
| **Notwendige Vorbereitungen** | | | |
| - Definieren einer Informationsart | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige der vorhandenen Dimensionen (13.05.2015) * Hinzufügen von Dimensionen (23.06.2015) * Hinzufügen von berechneten Dimensionen * Ändern der Bezeichnung für Dimensionen * Ändern des Typs einer Dimension * Hinzufügen weiterer Meta-Daten * Hinzufügen von Synonymen für die Benennung von Dimensionen | | | |

## Informationsansicht und -bearbeitung

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsübersicht | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Informationsübersicht bietet die Möglichkeit, sich Informationen anzeigen zu lassen. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige einer Informationsart:   + Anzeige aller Informationen einer Informationsart mit Titel (12.03.2015)   + Anzeige aller Informationen einer Informationsart mit Titel und Kurzbeschreibung (13.03.2015)   + Anzeige aller Informationen einer Informationsart mit Titel, Kurzbeschreibung und Datum. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsdetailansicht für Strukturierte Informationen | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Informationsdetailansicht stellt eine Information mit ihren Attributen dar. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| abgeschlossen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige der Kurzbezeichnung (03.03.2015) * Anzeige der Dimensionen (04.03.2015) * Werte der Dimensionen können geändert werden (30.06.2015) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Dokumenteneditor | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Der Dokumenteneditor bietet die Möglichkeit Dokumente zu bearbeiten. Außerdem können hier manuell Informationen extrahiert werden. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Dokumenteneingabe   + Eingeben von (unstrukturierten) Texten (13.11.2014)   + Erfassen von Kurzbeschreibung und Exposé (13.11.2014)   + Erfassen von Meta-Daten (ergänzende Dimensionen) zum Text (04.03.2015) * Dokumentenstatistik   + Anzahl der Zeichen (18.06.2015)   + Anzahl der Sätze (18.06.2015)   + Anzahl der Wörter (18.06.2015)   + Anzahl Informationen (18.06.2015)   + Anzahl Verschiedene Informationen (18.06.2015)   + Anzahl der Informationsdichte (18.06.2015)   + Anzahl der Stoppwörter   + Hover-Anzeige der Stoppwörter   + Anzahl der am häufigsten verwendeten Wörter   + Dichte der am häufigsten verwendeten Wörter   + Hover-Anzeige der am häufigsten verwendeten Wörter * Informationsmarkierung   + Manuelle Markierung von Informationen (Auswahl aus bereits referenzierten) (20.08.2015)   + Manuelle Markierung von Informationen (Neuanlage)   + Manuelle Markierung von Informationen (Suche)   + Anstoßen der automatischen Extraktion von Informationen * Bearbeiten von markierten Informationen   + Entfernung von Markierungen für Informationen und Dimensionen (23.06.2015) * Markieren der Vorkommen der gleichen Information, beim Klicken auf eine Information und Anzeige der Anzahl.   + Markierung aller Dimensionen, wenn auf eine Dimension im Dokument geklickt wird (22.06.2015)   + Markierung aller Informationsvorkommen, wenn auf eine Information im Dokument geklickt wird. (22.06.2015) * Anzeige der im Dokument vorhandenen Informationen (02.06.2015)   + Anzeige der Positionen der Informationen im Text, wenn mit der Maus über eine Information gefahren wird. (03.06.2015)   + Anzeige der Anzahl der Vorkommen im Text von überfahrenen Informationen. (10.06.2015)   + Dauerhaftes halten (sticky) einer Informationsmarkierung (10.06.2015)   + Anzeige der Positionen der Dimensionen im Text, wenn mit der Maus über eine Dimension gefahren wird. (18.06.2015)   + Anzeige der Anzahl der Vorkommen im Text von überfahrenen Dimensionen. (18.06.2015)   + Dauerhaftes halten (sticky) einer Dimensionsmarkierung (22.06.2015)   + Anzeige der, dem Dokument übergeordneten Informationen (05.06.2015)   + Markierung von Referenzen im Dokument und Anzahl der Vorkommen (19.06.2015) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Schreibassistent | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Der Schreibassistent warnt bei zu häufiger Verwendung von Wörtern und anderen Stil-Fehlern. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Warnung bei zu häufiger Verwendung von Wörtern | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsassistent | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Der Informationsassistent unterstützt bei der Markierung von Informationen. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Informationen hinzufügen   + Vorschläge bei Eingabe: Referenzierte Informationen     - Automatische markieren bei Eindeutigkeit (14.08.2015)     - Anbieten von Vorschlägen bei Nichteindeutigkeit (17.08.2015)     - Vorschläge auch bei Satzendzeichen erhalten (20.08.2015) * Überwachung von Informationen   + Warnen wenn der Wert einer Dimension anders ist, als der Wert des markierten Bereichs, dem die Dimension zugeordnet werden soll. (12.08.2015)   + Prüfen ob eine Information oder Dimension noch länger im Dokument referenziert ist, wenn diese entfernt wurde   + Warnung, wenn ein Satz keine Information enthält. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsvisualisierung | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Informationsvisualisierung unterstützt bei der Interpretation der Informationen indem sie Zusammenhänge darstellt. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Visualisierung Informationen   + Anzeige: „Wer sagt was“   + Anzeige: „Wer redet mit wem“   + Visualisierung Informationsfluss   + Pfeile auf die nächste Verwendung einer Information. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsrelationsanzeige | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| In der Informationsrelationsanzeige werden alle Informationen die in irgendeiner Weise mit einer vorgegebenen Information verbunden sind, angezeigt | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige aller Informationen die Eltern der vorgegebenen Information sind. (15.05.2015) * Anzeige aller Informationen, die Kinder der vorgegebenen Information sind. (15.05.2015) * Anzeige aller Geschwisterinformationen zu einer konkreten Information. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsrelationdefinierung | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| In der Informationsrelationsdefinierung können neue Relationen zwischen Informationen angelegt werden. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Auswahl der beteiligten Informationen * Auswahl der Art der Relation * Festlegen der Zeit, in der die Relation gilt | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationszeitrelationsanzeige | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| In der Informationszeitrelationsanzeige werden Informationen in ihrem zeitlichen Zusammenhang dargestellt. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige der Informationen, die Vorgänger oder Nachfolger einer konkreten Information sind. * Anzeige aller Informationen die zum Zeitpunkt einer konkreten Information ebenfalls gültig sind. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationseingabedialog | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Im Informationseingabedialog können neue Informationen eingegeben werden. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anlegen von Werten die zu einer Dimension gehören * Anlegen von Texten die zu einer Dimension gehören * Aufruf der Analyse-Funktion vom IOP | | | |

## Personalisierungen

|  |  |  |  |
| --- | --- | --- | --- |
| Dasboard | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Das Dashboard ist die Einstiegsseite für einen Benutzer. Diese Seite ist in der Anzeige der Informationen konfigurierbar. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Menü zu den untergeordneten Dialogen * Einstellbarkeit: Auswahl der Informationen, die als Einstiegspunkt dienen * Einstellbarkeit: Auswahl der Information, die als Einstiegspunkt dient. * Einstellbarkeit: Auswahl der anzuzeigenden Attribute. * Einstellbarkeit: Auswahl der anzuzeigenden Elterninformationen. * Einstellbarkeit: Auswahl der anzuzeigenden Kindinformationen. * Darstellung einer Liste von Informationen (Abhängig von der Einstellung) * Link zur Informationsdetailansicht einer Information * Sortierbarkeit der Informationen nach allen angezeigten Attributen. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Dokumentenräume | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Dokumentenräume fassen eine Menge von Dokumenten zusammen. Die Dokumentenräume sind losgelöst von Informationen und ihre Inhalte daher frei definiert. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Zusammenfassen von Dokumenten zu einer organisatorischen Einheit * Abrufen von Dokumenten aus externen Quellen * Extraktion von Merkmalen aus den Dokumenten eines Dokumentenraums * Bewerten der Qualität von Dokumenten durch Vergleich von Dokumentenräumen | | | |

Quellen für weitere Funktionalitäten:

* Daten im Sinne der Wikidata:

https://www.wikidata.org/wiki/Help:About\_data

* Verknüpfung zu Daten der wikidata / wikipedia

<https://www.mediawiki.org/wiki/Wikidata_Toolkit>

Beispiele:

<https://github.com/Wikidata/Wikidata-Toolkit-Examples/tree/master/src/examples>

Wikidata abfrage:

https://tools.wmflabs.org/reasonator/

Verwendung von HTML Micordata:

## Strukturansicht und –bearbeitung

|  |  |  |  |
| --- | --- | --- | --- |
| Strukturübersicht | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Strukturübersicht bietet die Möglichkeit, sich die Struktur der Informationen übersichtlich anzeigen zu lassen. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige aller Kind-Informationsarten * Anzeige aller Eltern-Informationsarten * Anzeige aller Geschwister-Informationsarten * Hinzufügen einer neuen Informationsart mit Angabe der möglichen Attribute | | | |

## Informationsansicht und -bearbeitung

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsübersicht | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Informationsübersicht bietet die Möglichkeit, sich Informationen anzeigen zu lassen. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige einer Informationsart:   + Anzeige aller Datensätze einer Informationsart mit Titel   + Anzeige aller Datensätze einer Informationsart mit Titel und Kurzbeschreibung   + Anzeige aller Datensätze einer Informationsart mit Titel, Kurzbeschreibung und Datum. * Anzeige relevanter Informationen:   + Anzeige aller Datensätze einer Informationsart, die mit einer konkreten Information verbunden sind.   + Anzeige aller Datensätze einer Informationsart, die mit einer ausgewählten Informationsart verbunden sind.   + Anzeige aller Unterinformationen zu einer konkreten Information.   + Anzeige aller direkten Unterinformationen zu einer konkreten Information.   + Anzeige aller übergeordneten Informationen zu einer konkreten Information.   + Anzeige aller Geschwisterinformationen zu einer konkreten Information.   Definition Geschwister: Haben mindestens einen gleichen Vorfahren   * Anzeige zeitlich verbundener Informationen:   + Anzeige der Informationen, die Vorgänger oder Nachfolger einer konkreten Information sind.   + Anzeige aller Informationen die zum Zeitpunkt einer konkreten Information ebenfalls gültig sind. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Informationsdetailansicht | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Die Informationsdetailansicht stellt eine Information mit ihren Attributen dar. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Anzeige der Kurzbezeichnung * Anzeige der Attribute * Attribute können geändert werden | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Dokumenteneditor | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Der Dokumenteneditor bietet die Möglichkeit Dokumente zu bearbeiten. Außerdem können hier manuell Informationen extrahiert werden. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Eingeben von (unstrukturierten) Texten * Erfassen von Meta-Daten zum Text * Extraktion von Merkmalen zum Text * Manuelle Extraktion von Informationen * Anstoßen der automatischen Extraktion von Informationen * Anzeige der Informationszugehörigkeit im Text * Informationsdetailansicht beim Markieren von Informationen * Anzeige der im Dokument vorhandenen Informationen * Anzeige der, dem Dokument übergeordneten Informationen * Dokumentenstatistik   + Anzahl der Zeichen   + Anzahl der Sätze   + Anzahl der Wörter   + Anzahl Informationen   + Anzahl Verschiedene Informationen   + Anzahl der Informationsdichte   + Anzahl der Stoppwörter   + Hover-Anzeige der Stoppwörter   + Anzahl der am häufigsten verwendeten Wörter   + Dichte der am häufigsten verwendeten Wörter   + Hover-Anzeige der am häufigsten verwendeten Wörter * Schreibassistent   + Warnung, wenn ein Satz keine Information enthält.   + Warnung bei zu häufiger Verwendung von Wörtern * Visualisierung Informationen   + Anzeige: „Wer sagt was“   + Anzeige: „Wer redet mit wem“   + Visualisierung Informationsfluss   + Pfeile auf die nächste Verwendung einer Information. | | | |

## Personalisierungen

|  |  |  |  |
| --- | --- | --- | --- |
| Dasboard | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Das Dashboard ist die Einstiegsseite für einen Benutzer. Diese Seite ist in der Anzeige der Informationen konfigurierbar. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Menü zu den untergeordneten Dialogen * Einstellbarkeit: Auswahl der Informationen, die als Einstiegspunkt dienen * Einstellbarkeit: Auswahl der Information, die als Einstiegspunkt dient. * Einstellbarkeit: Auswahl der anzuzeigenden Attribute. * Einstellbarkeit: Auswahl der anzuzeigenden Elterninformationen. * Einstellbarkeit: Auswahl der anzuzeigenden Kindinformationen. * Darstellung einer Liste von Informationen (Abhängig von der Einstellung) * Link zur Informationsdetailansicht einer Information * Sortierbarkeit der Informationen nach allen angezeigten Attributen. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Dokumentenräume | | | |
| **Priorität** |  |  |  |
| **Beschreibung** | | | |
| Dokumentenräume fassen eine Menge von Dokumenten zusammen. Die Dokumentenräume sind losgelöst von Informationen und ihre Inhalte daher frei definiert. | | | |
| **Notwendige Vorbereitungen** | | | |
|  | | | |
| **Status** | | | |
| Nicht Begonnen | | | |
| **Anmerkungen Entwickler** | | | |
| * Zusammenfassen von Dokumenten zu einer organisatorischen Einheit * Abrufen von Dokumenten aus externen Quellen * Extraktion von Merkmalen aus den Dokumenten eines Dokumentenraums * Bewerten der Qualität von Dokumenten durch Vergleich von Dokumentenräumen | | | |

# Administration

## Installation

1. Installieren sie zunächst die Modifikation „ILIAS AdvancedRTEService“
2. Installieren sie diese Modifikation mit Hilfe des Installationspaketes über den ModificationManager.
3. Anmeldung als Administrator in ILIAS
4. Legen sie entsprechend der Dokumentation des „ILIAS AdvancedRTEService“ eine neue Konfiguration an.
5. Aktivieren sie in der Konfiguration das Plugin „mathml“
6. Platzieren sie die mathml-Buttons in der Buttonleiste.
7. Speichern sie die Konfiguration.

## Installation (Manuell):

1. Installieren sie zunächst die Modifikation „ILIAS AdvancedRTEService“
2. Legen sie eine Sicherung aller Dateien die geändert werden an. (siehe [Modifizierte Dateien](#_Modifizierte_Dateien))
3. Entpacken sie die Inhalte des Verzeichnisses in das Stamm-Verzeichnis ihrer ILIAS Installation.
4. Führen sie die auf 2. Folgenden Schritte unter „Installation“ aus.

## Deaktivierung

Die Verwendung des MathML Editors kann für jeden Editor separat eingestellt werden. Um die Verwendung für einen Editor zu deaktivieren, führen sie folgende Schritte aus:

1. Anmeldung als Administrator in ILIAS
2. Rufen sie die entsprechende Konfiguration des „ILIAS AdvancedRTEService“ auf.
3. Entfernen sie das Plugin „mathml“ aus der Auflistung.
4. Speichern

Beachten sie:

Auch wenn die Verwendung des MathML Editors deaktiviert ist, werden bereits existierende Formeln im Editor angezeigt. Beim erneuten Speichern können sie jedoch verloren gehen.

## Konfiguration

Der VisualMathEditor hat keine Konfigurationsmöglichkeiten im administrativen Bereich.

# Entwicklung

## Algorithmen

## *Informationsverarbeitung*

### Dokumentenbasiert

Jede Information wird zunächst als Dokument erfasst. Ein Dokument kann dabei ein beliebiger Text, ein Bild oder ähnliches sein. Im Rahmen der Arbeit konzentrieren wir uns auf Daten die in Textform vorliegen.

### Identifizierung

Der erste Arbeitsschritt in der Informationsverarbeitung ist die Identifizierung wesentlicher Wörter. In diesem Schritt wird die Sprache des Dokuments vereinfacht.

Algorithmus:

1. Entferne alle Wörter die keine Information tragen
   1. Suchen und Entfernen anhand einer Blacklist. (Dynamische Liste die durch die späteren Algorithmen gefüllt werden kann.)
2. Aufteilung in Sätze
   1. Erkennen von kompletten und von Teilsätzen.

### Assoziierung

Die einzelnen Begriffe werden bekannten Begriffen zugeordnet.

Algorithmus:

1. Wiederhole für jeden Satz.
2. Zerlege den Satz in seine Wörter
3. Suche zu jedem Wort Einträge in der Datenbank
   1. Wenn das Wort einmal vorhanden ist:
      1. Markiere das Wort im Dokument entsprechend des Informationstyps.
   2. Wenn das Wort mehrfach vorhanden ist:
      1. Füge das Wort und seine möglichen Bedeutungen zum Stack für die Kontexterkennung hinzu
   3. Wenn das Wort nicht vorhanden ist:
      1. Füge das Wort zum Stack für die Objektisierung hinzu.

Auswirkungen:

* Reduzierung der Komplexität der nachfolgenden Schritte durch Reduzierung auf unbekannte Begriffe.
* Reichert das Dokument mit Daten für die Kontexterkennung an.

### Objektisierung

Im Schritt der Objektisierung werden alle Begriffe identifiziert, die Objekte darstellen.

Algorithmus:

1. Wiederhole für jedes Wort auf dem Stack für die Objektisierung:
2. Prüfe die Spracheigenschaft
   1. Wenn die verwendete Sprache für Nomen eine Großschreibung vorsieht.
      1. Füge jedes groß geschriebene Wort dem Stack für Objekte hinzu
      2. Füge jedes klein geschriebene Wort dem Stack zur Attributisierung hinzu.
3. Frage nach
   1. Prüfe ob es für das Wort einen Eintrag in der Wikipedia gibt.
      1. Wenn ja füge das Wort dem Stack für Objekte hinzu
   2. Frage den Benutzer: „Bezeichnet xyz ein Objekt?“
      1. Wenn ja, füge das Wort dem Stack für Objekte hinzu
      2. Wenn nein, füge das Wort dem Stack zur Attributisierung hinzu.

### Attributisierung

Im Schritt der Attributisierung werden alle Begriffe identifiziert, welche Attribute von Objekten bezeichnen.

Algorithmus:

1. Wiederhole für jedes Wort auf dem Stack zur Attributisierung:

### Relationalisierung

Im Schritt der Relationalisierung werden alle Begriffe identifiziert, die Beziehungen zwischen Objekten und zwischen Objekten und Attributen definieren.

Algorithmus:

1. Prüfe ob es in dem Satz ein oder mehrere Objekte gibt.
   1. Wenn es kein Objekt gibt:
      1. Prüfe ob es Objektreferenzen gibt.
         1. Wenn ja: Vermerke Referenz auf Objekt.
         2. Wenn nein: Frage den Benutzer: „Worauf bezieht sich xyz?“

Vermerke Referenz auf Objekt.

* 1. Wenn es ein Objekt oder eine Objektreferenz gibt:

1. Bestimme die Art der Beziehung:

### Kontexterkennung

Bei der Kontexterkennung werden Zusammenhänge zwischen Informationen hergestellt.

Algorithmus:

1. Wiederhole für jeden Satz:
2. Prüfe ob eines der im Satz vorhandenen Objekte im nächsten Satz wieder vorkommt. (Objekt oder Objektreferenz)
   1. Wenn ja: Vermerke den Kontext beider Sätze.
   2. Wenn nein:Prüfe den übernächsten Satz mit 2.

### Abstraktion

Im Schritt der Abstraktion werden Attribute und Beziehungen ermittelt, die für mehr als ein Objekt anwendbar sind.

Algorithmus:

1. Für alle gefundenen Objekte im Dokument
2. Prüfe für jedes Attribut eines Objekts, ob es bei einem anderen Objekt ebenfalls vorkommt.
   1. Wenn ja:
      1. Prüfe ob beide Attribute den gleichen Wert haben:
         1. Wenn ja: Lege eine neue Attributtabelle für das Attribut an. Lege eine Relation zwischen den Objekten und dem Attribut an.
         2. Wenn nein:

### Kategorisierung

Im Rahmen der Kategorisierung werden Informationen in eine Hierarchie gebracht.

## Versionshistorie

|  |  |  |
| --- | --- | --- |
| **Version** | **Datum** | **Änderungen** |
| 0.0.1 | 23.05.2013 | Neu:  - TinyMCE3.x.x global ausgetauscht gegen TinyMCE 4.0.0b2 (Beta)  - Fragentyp „Fehler markieren“ modifiziert, damit TinyMCE Suchtexte erstellt und abgespeichert werden können.  - Erste Preview des VisualMathEditor. |

# Testfälle

|  |  |
| --- | --- |
| ID: | 1 |
| Titel: | Login eines Benutzers wird geloggt |
| Aktion: | Ein Benutzer loggt sich ein. |
| Reaktion: | In der Datenbank wird in der Tabelle evnt\_evhk\_isl\_usract ein Eintrag mit Zeitpunkt, Benutzer Id und activity=“login“ angelegt. |
| Tests: | 0.0.25/ILIAS4.3.4: OK |

# Literaturverzeichnis

*jsMath Kompatibilitätsliste*. (kein Datum). Abgerufen am 19. 12 2012 von http://www.math.union.edu/~dpvc/jsmath/browsers.html

*mathJax Kompatibilitätsliste*. (kein Datum). Abgerufen am 19. 12 2012 von http://www.mathjax.org/resources/browser-compatibility/