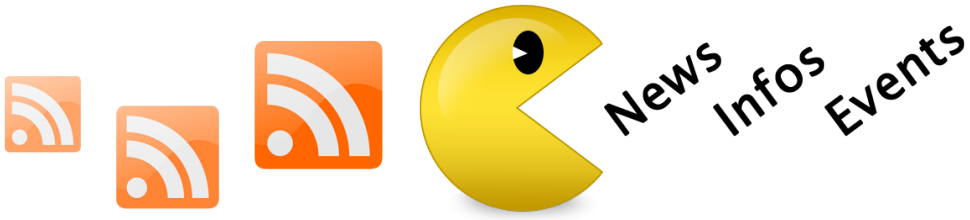
HOCHSCHULE DER MEDIEN

Projekt Studiengang Medieninformatik

Feed It



Autor: Michael Raith
Betreuer: Prof. Dr. Fridtjof Toenniessen

Sommersemester 2009
Erstellt am: 12. Juli 2009

Hochschule der Medien Stuttgart

Inhaltsverzeichnis

1 Einführung **1**
- 1.1 Motivation . . . 1
- 1.2 RSS-Feed . . . 2
- 1.3 Erweiterungen . . . 3
  - 1.3.1 Was sind Erweiterungen? . . . 3
  - 1.3.2 Verwendete Begriffe . . . 4
    - 1.3.2.1 XUL . . . 4
    - 1.3.2.2 Overlay . . . 4
    - 1.3.2.3 XPI . . . 5
    - 1.3.2.4 Chrome . . . 5
    - 1.3.2.5 DOM . . . 5
  - 1.3.3 Overlays und das ID-Attribut . . . 6
  - 1.3.4 Benutzeroberflächen wiederverwenden mit Overlays . . . 6

2 Projektbeschreibung **8**
- 2.1 Projektdefinition . . . 8
  - 2.1.1 Scope . . . 8
  - 2.1.2 Ziele . . . 9
  - 2.1.3 Umfang und Abgrenzungen . . . 9
  - 2.1.4 Schnittstellen . . . 10
  - 2.1.5 Einschränkungen . . . 10
  - 2.1.6 Voraussetzungen / Rahmenbedingungen . . . 10
- 2.2 Kunde(n) . . . 11
- 2.3 Abnahmekriterien . . . 11
- 2.4 Risiken . . . 11

3 Projektlösungsansatz **12**
- 3.1 Erforderliche Qualifikationen . . . 12
- 3.2 Vorhandene Infrastrukturen . . . 12
- 3.3 Wartungs- und Betriebsanforderungen . . . 12
- 3.4 Sicherheitsanforderungen . . . 13
- 3.5 Alternativen . . . 14

4 Produktbeschreibung **15**
- 4.1 Produktstrukturplan . . . 15
- 4.2 Produktflussdiagramm . . . 16
- 4.3 Managementprodukte . . . 17
  - 4.3.1 Einführung . . . 17
  - 4.3.2 Projektbeschreibung . . . 17
  - 4.3.3 Projektlösungsansatz . . . 17

4.3.4 Produktbeschreibung . . . 17
4.3.5 Projektplan . . . 18
4.3.6 Projektflussdiagramm . . . 18
4.3.7 Risikomanagement . . . 18
4.3.8 Fazit . . . 18
4.3.9 Quellen . . . 18
4.4 Spezialistenprodukte . . . 19
4.4.1 Client . . . 19
4.4.1.1 GUI . . . 19
4.4.1.2 Logik hinter der GUI . . . 19
4.4.1.3 Weitere Funktionen der GUI . . . 22
4.4.1.4 Anwendung . . . 22
4.4.2 Server . . . 22
4.4.2.1 Erstellen von News-Feeds . . . 22
4.4.2.2 Neu-Erstellen / Aktualisieren von News-Feeds . 23
4.4.2.3 Datenbank . . . 23
4.4.3 Website . . . 24
4.4.3.1 Aufbau . . . 24
4.4.3.2 Suchfunktion . . . 25
4.5 Qualitätsmanagement . . . 25

5 Projektplan **27**
5.1 Wichtige Punkte . . . 27
5.2 Abschätzungen . . . 27
5.3 Gant-Diagramm . . . 29

6 Risikomanagement **30**

7 Fazit **32**
7.1 Projektmanagementtool . . . 32
7.2 Findung der Projektidee . . . 32
7.3 Client . . . 33
7.3.1 Probleme . . . 33
7.3.1.1 Event-Listener . . . 33
7.3.1.2 Datenaustausch . . . 34
7.3.2 Überladen von GUI Elementen . . . 35
7.4 Server . . . 35
7.4.1 News-Feed erstellen . . . 35
7.4.2 Probleme . . . 36
7.5 Website . . . 36
7.6 Schlußfazit . . . 37

8 Quellenangabe **38**
8.1 Internet . . . 38
8.2 Programme, Tools . . . 38
8.3 Sonstige . . . 38

1 Einführung

1.1 Motivation

In den letzten Jahren haben sich die News-Feeds immer weiter im Internet „ausgebreitet“. Durch ihre einfache Integration in News-Feed Readern, wie z.B. in Thunderbird[1], kann man sehr einfach viele News-Feeds auf einmal überblicken und sich die wichtigsten Information herauspicken und ggf. genauer lesen.

In einer Zeitung gibt es auf der ersten Seite Kurznachrichten mit Verweisen auf Zeitungsartikel, die in der Zeitung weiter hinten stehen. Ein News-Feed ist das digitale Gegenstück dazu. In einem News-Feed stehen die wichtigsten Informationen in Form einer Überschrift und Kurzbeschreibung zusammengefasst. Liegt Interesse vor mehr über einen Eintrag in einem News-Feed zu erfahren, klickt man auf den entsprechenden Link um auf eine Seite des News-Feed Anbieters weiter geleitet zu werden.

Ein Novum von News-Feeds ist, dass der Benutzer sich die Informationen holt, indem er einen News-Feed abonniert und nicht wie bei Newslettern eine eMail-Adresse angeben muss.

Im Internet gibt es viele Websites, die keine News-Feeds anbieten. Dies sind in der Regel Websites mit geringem bis mittlerem Besucheraufkommen. Jedoch finden sich auch auf diesen wichtige Informationen. Die Idee besteht nun darin, für solche Websites einen News-Feed für die *eigene Benutzung* zu erstellen. Es wäre empfehlenswert einen News-Feed direkt aus dem Datenbestand der Website zu erstellen. Das können aber nur die entsprechenden Webmaster der jeweiligen Website!

Die Idee besteht nun darin aus diesen Websites einen News-Feed für die eigene Benutzung zu generieren.

Im vorliegenden Projekt geht es darum, Strukturen in Websites zu erfassen und in einem allgemein lesbaren Format zur Verfügung zu stellen: In einem News-Feed oder - wie auch im weiteren Kontext verwendet - einem RSS-Feed.

[1] ist ein Open-Source eMail-Client → www.mozilla.org/thunderbird

1.2 RSS-Feed

Abbildung 1.1: Das (inzwischen) Standard Icon für RSS-Feeds

Eine Neuerung des Internet-Zeitalters war in den letzten Jahren sicherlich die Einführung von Format neutralen RSS[2] News-Feeds. Diese basieren auf dem XML-Standard und sind deshalb besonders gut für den Plattform unabhängigen Informationsaustausch geeignet. Der Begriff *Feed* setzt sich aus dem engl. *to feed* zusammen und bedeutet Wort wörtlich: *Den Benutzer mit Informationen füttern*. Falls ein RSS-Feed auf einer Website vorhanden ist, dann wird dies in der Regel direkt im Browser angezeigt → siehe Abbildungen 1.1, 1.2, 1.3, 1.4, 1.5

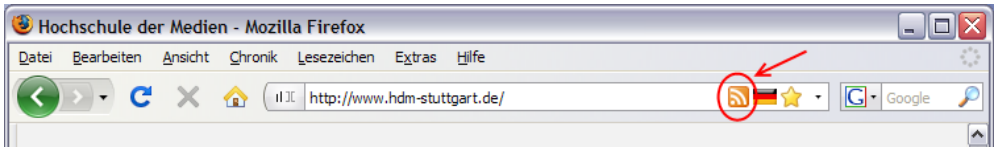


Abbildung 1.2: RSS-Feed im Firefox

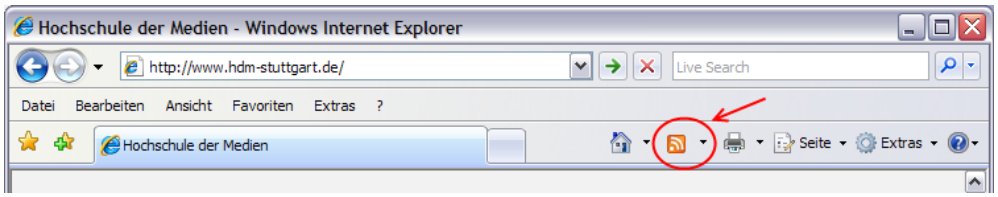


Abbildung 1.3: RSS-Feed im Internet Explorer

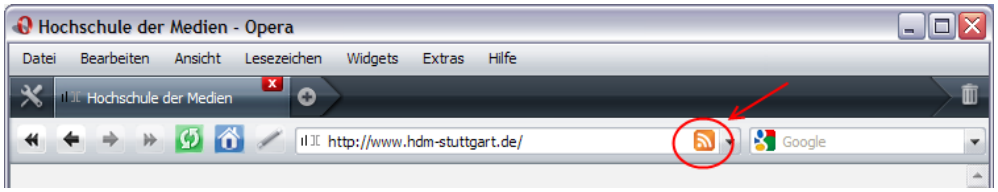


Abbildung 1.4: RSS-Feed im Opera

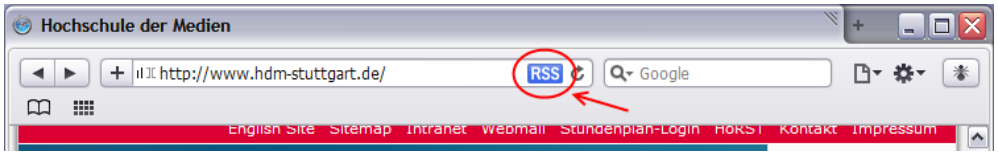


Abbildung 1.5: RSS-Feed im Safari

[2] Abkürzung RSS: **Rich Site Summary** in den RSS-Versionen 0.9x, **RDF Site Summary** in den RSS-Versionen 0.9 und 1.0, **Really Simple Syndication** in RSS 2.0

Listing 1.1: Code Beispiel für einen RSS-Feed

```
1 <?XML version="1.0" encoding="utf-8"?>
2 <RSS version="2.0">
3
4   <channel>
5     <title>Titel des Feeds</title>
6     <link>URL der Webpraesenz</link>
7     <description>Kurze Beschreibung des Feeds</description>
8     <language>Sprache des Feeds (z. B. "de-DE")</language>
9     <copyright>Autor des Feeds</copyright>
10    <pubDate>Erstellungsdatum("Tue, 8 Jul 2008 2:43:19")</pubDate>
11
12    <item>
13      <title>Titel des Eintrags</title>
14      <description>Kurze Zusammenfassung des Eintrags</description>
15      <link>Link zum vollstaendigen Eintrag</link>
16      <author>Autor des Artikels, E-Mail-Adresse</author>
17      <guid>Eindeutige Identifikation des Eintrages</guid>
18      <pubDate>Erstelldatum des Items</pubDate>
19    </item>
20
21    <item>
22      ...
23    </item>
24
25  </channel>
```

Diese RSS-Feeds sind immer nach dem gleichen Schema aufgebaut. Information über den Feed selbst (Titel, Beschreibung → Zeile 5-10), sowie die eigentlichen News Elemente (Titel, kurze Beschreibung, Link, Datum ... → Zeile 12-19). Die zuletzt genannten News Elemente (item) können sich mehrfach wiederholen (Zeile 12-19, 21-23, ...).

1.3 Erweiterungen

In den folgenden Abschnitten 1.3.3 und 1.3.4 werden die Grundprinzipien und Mechanismen der Mozilla AddOns (Erweiterung) besprochen. Dies gibt einen Überblick darüber, wie sich Erweiterungen in Mozilla-Firefox einbinden lassen.

1.3.1 Was sind Erweiterungen?

Wie der Name schon sagt, statten Erweiterungen den Browser mit Funktionen aus, die dieser normalerweise nicht hat. Man kann nicht „alles“ in einen Browser einbauen, da er sonst zu groß und überladen wäre. Außerdem brauchen nicht alle Benutzer alle Funktionen. Deshalb wurde Firefox von Haus aus schlank gehalten, sodass das ganze Mozilla-Projekt einfacher Hand zu haben ist sowohl für die Entwickler als auch für die Benutzer.

Neue Funktionen können über Erweiterungen in den Browser gebracht werden. Dabei muss man jedoch Erweiterungen von Plugins und Suchplugins unterscheiden. Durch Plugins kann der Browser neue mediale Inhalte wiedergeben, wie z.B. PDFs anzeigen. Suchplugins erweitern den Browser um neue Suchmaschineneinträge in der Suchleiste (rechts oben im Browser). Erweiterungen hingegen bringen neue Funktionen in den Browser, welche dieser zuvor nicht oder nur eingeschränkt hatte, wie z.B. eine erweiterte Fehlerkonsole, Blocker für Werbung usw. ...

1.3.2 Verwendete Begriffe

1.3.2.1 XUL

Die Abkürzung XUL steht für: *„**X**ML **U**ser Interface **L**anguage“* gesprochen: *zool*.

XUL ist eine auf XML basierende Beschreibungssprache für grafische Benutzeroberflächen. Diese wurde für den Firefox Browser entwickelt, findet aber immer öfter auch Einzug in andere Programme, wie z.B. in Thunderbird und in Songbird[3]

Sowohl grafischen Benutzeroberflächen, welche durch die XUL-Dateien definiert werden, als auch die Website selbst werden von der „Gecko Rendering Engine“[4] dargestellt.

1.3.2.2 Overlay

Wichtiger Bestandteil einer Erweiterung sind die XUL Dateien und die XUL Overlays[5]. Overlays bestehen aus XUL Dateien, welche im Gegensatz zu XML-Dateien zusätzliche Benutzeroberflächen, Inhalte oder Informationen enthalten. Overlays stellen einen allgemeinen Mechanismus zur Verfügung zum:

- Hinzufügen von zusätzlichen Benutzeroberflächen in den Kontext
- Überschreiben von kleinen, schon vorhandenen Teilen in XUL Dateien, ohne dass man dabei die komplette Benutzeroberfläche neu gestalten und erstellen muss, z.B. den Skin des Browsers ändern
- Wiederverwendung von schon erstellten Elementen/Teilen der Benutzeroberflächen wie z.B. Dialog Buttons

Es gibt keine Definition, was genau in XUL Dateien und in Overlays stehen muss!

Mit Overlays werden in der Regel Elemente/Teile zum Gesamtkontext hinzugefügt. Deshalb sollten alle Plugins, Erweiterungen oder Anwendungen, die die Benutzeroberfläche ändern, als `<overlay />` definiert sein!

[3] Songbird ist der Mediaplayer des Mozilla Projektes
[4] Gecko Rendering Engine → http://de.wikipedia.org/wiki/Gecko_(Rendering_Engine)
[5] engl. Overlay: Überlagerung, Auflage

Um dies noch einmal zu verdeutlichen: XUL Dateien sind Beschreibungselemente zur Darstellung von Informationen in einem Browser. Der Browser selbst besteht u.a. aus der XUL Datei `"Browser.XUL"`. Um neue grafische Elemente in den Kontext zu laden, benutzt man eine weitere XUL Datei, die Overlay XUL Datei. Diese erweitert Browser um bestimme Teile. Wie dies geschieht, wird im folgenden Abschnitten 1.3.3 und 1.3.4 beschrieben.

1.3.2.3 XPI

Erweiterungen im Firefox sind normale Zip-Archive mit der Dateiendung XPI. Die Endung XPI oder XPInstall bedeutet „Cross-Platform Install“ und wird wie folgt ausgesprochen: „zippy“ (*/z?p.i/*).

XPI Archiv-Dateien bestehen hauptsächlich aus XUL und Javascript Dateien, welche zur Darstellung und für Aktionen benötigt werden. Außer den XUL und Javascript Dateien gibt es noch ein paar andere Dateien mit Eigenschaften, Darstellungsoptionen etc.

Dadurch, dass die Erweiterungen als Zip gepackt sind und innerhalb von diesen Zip-Archiven standardisierte Dateien liegen, sind Erweiterungen Plattform unabhängig und können in jedem Firefox Browser (mit der richtigen Versionsnummer) verwendet werden.

1.3.2.4 Chrome

Ein Firefox-Programmierer versteht unter Chrome den Bereich, in dem die Benutzeroberfläche abgespeichert ist. Um auf die Dateien, die im Chrome-Bereich gespeichert sind, zuzugreifen, existiert ein eigenes Protokoll im Firefox.

Hier ein kleiner Auszug von Chrome Adressen:

Bedeutung	Chrome Adresse
Browser-Interface	chrome://browser/content/browser.xul
Chronik	chrome://browser/content/history/history-panel.xul
Extras/Downloads	chrome://mozapps/content/downloads/downloads.xul
Lesezeichen	chrome://browser/content/bookmarks/bookmarksPanel.xul
Fehlerkonsole	chrome://global/content/console.xul

Tabelle 1.1: Wichtige Chrome Adressen

1.3.2.5 DOM

DOM steht für „**D**ocument **O**bject **M**odel“. Diese Schnittstelle ermöglicht den Zugriff auf HTML und XML Dokumente (hier: XUL Dokumente). DOM stellt dabei eine strukturierte Darstellung der Elemente in diesen Dokumenten zur Verfügung. Außerdem definiert DOM Standard Methoden, welche die Manipulation der Elemente oder die Abfrage von Eigenschaften dieser Elemente erleichtern. Der Zugriff auf die DOM Elemente erfolgt bei der Webentwicklung in der

Regel durch Javascript. DOM wurde vom W3C[6] Konsortium standardisiert[7]. Dadurch ist es möglich, Browser übergreifende Anwendungen zu entwickeln.

1.3.3 Overlays und das ID-Attribut

Was ist das „ID-Attribut“? Dieses gibt einem Element eine eindeutige Zuordbarkeit. Man könnte z.b. ein Textfeld `<textbox id="TextFeld1"value="Dies ist ein Textfeld"/>` ganz einfach über dessen Attribut `id="TextFeld1"` erreichen. Dadurch kann man eine Referenz zu diesem herstellen.

Das ID-Attribut ist wichtig. Dadurch weiß die Layout Engine, an welcher Stelle die neuen Elemente der XUL Overlay Datei eingebunden und mit den Elementen der schon vorhandenen XUL Datei verschmolzen werden sollen. Dabei ist zu beachten, dass die Elemente, welche überschrieben/erweitert werden sollen identische ID und Tag-Namen besitzen, wie die Elemente in der schon vorhandenen XUL Datei!

Durch die Anweisung `<overlay xmlns="[...]/there.is.only.xul"> [...] </overlay>` ist der Browser bzw. die Rendering Engine angewiesen schon vorhandene Elemente im Browser Kontext zu „überladen“.

Das folgende Beispiel soll dies erläutern:

```
1 <?xml version="1.0"?>
2 <overlay id="singleItemExample"
3   xmlns:html="http://www.w3.org/1999/xhtml"
4   xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
5
6   <menupopup id="menu_FilePopup">
7     <menuitem label="Neuer Menu Eintrag"/>
8   </menupopup>
9 </overlay>
```

In diesem Beispiel wird dem Datei-Menü von Mozilla-Firefox ein weiterer Menüpunkt hinzugefügt: `<menuitem label="Neuer Menu Eintrag"/>`. Dies erreicht man, indem man das originale Datei-Menü über das ID-Attribut `<menupopup id="menu_FilePopup">...</menupopup>` „an wählt“.

Dadurch wird der Layout Engine mitgeteilt, dass im Datei-Menü (menu_FilePopup) ein neuer Eintrag `<menuitem label="Neuer Menu Eintrag"/>` am Ende des Menüs eingefügt werden soll. Somit wird das schon existierende Menü um einen weiteren Menüpunkt erweitert, ohne dass der gesamte Code der Benutzeroberfläche angefasst bzw. neu erstellt werden muss. Die Grafiken 1.6 und 1.7 zeigen dieses Beispiel.

1.3.4 Benutzeroberflächen wiederverwenden mit Overlays

Die Overlay Methode hat den großen Vorteil, dass Elemente oder sogar ganze Gruppen von Elementen wiederverwendet werden können. Um einen neuen

[6] W3C → „**W**orld **W**ide **W**eb **C**onsortium“
[7] W3C DOM Standard → http://www.w3.org/dom

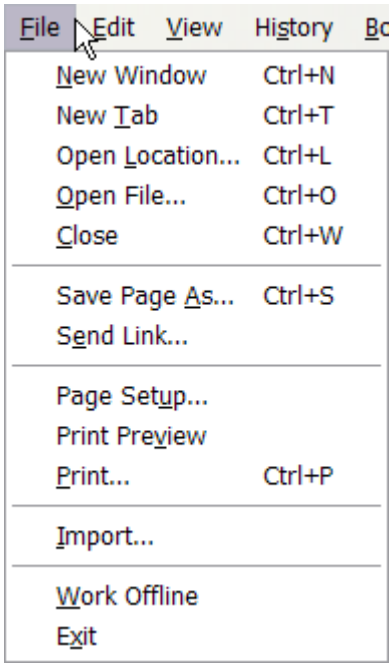


Abbildung 1.6: vorher

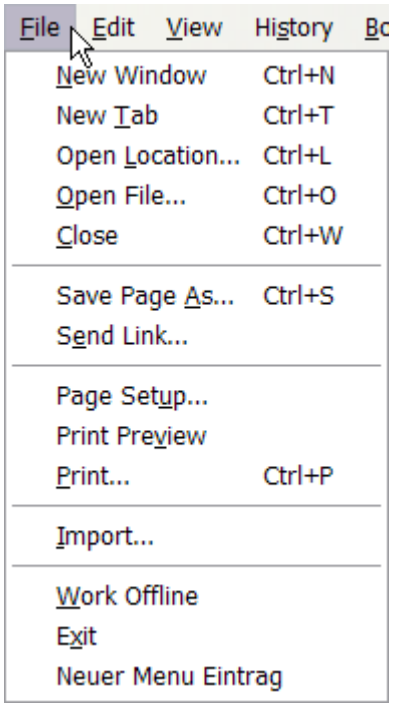


Abbildung 1.7: nacher

Abbildung 1.8: Overlays und das ID-Attribut

Dialog zu entwickeln, kann man vorhandene Elemente wiederverwenden. Dabei können Standard Elemente wie z.B. *Ok*-und *Abbrechen*-Buttons aus dem Standarddialog übernommen werden ohne den Code für diese neu zu schreiben. Folgendes Beispiel soll dies verdeutlichen:

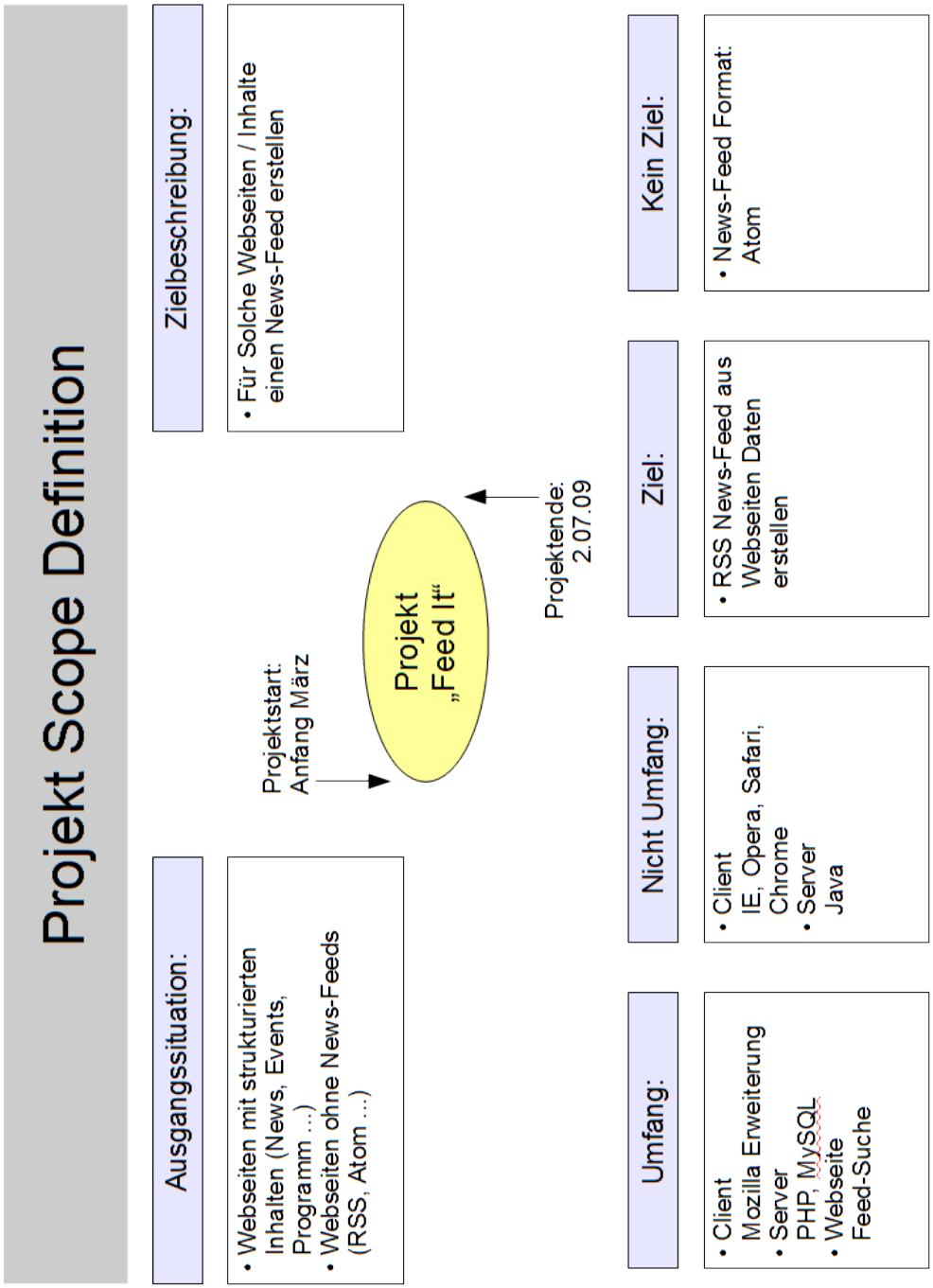
```
1 <?xml version="1.0"?>
2 <?xul-overlay href="chrome://global/content/dialogoverlay.xul"?>
3
4 <dialog
5   xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
6
7   <box>
8     <label value="Dies ist ein Testdialog" />
9   </box>
10
11  <box id="okCancelButtons"/>
12 </dialog>
```

Dabei stehen die Dialog-Buttons in der Datei `dialogoverlay.xul`. Diese wird in der Zeile 2 eingebunden. Die Ok- und Abbrechen-Buttons werden an der Stelle `<box id="okCancelButtons"/>` durch den Overlay Mechanismus wie oben beschrieben eingebunden. Die restlichen Angaben in diesem Beispiel sollen uns erst einmal nicht weiter interessieren.

2 Projektbeschreibung

2.1 Projektdefinition

2.1.1 Scope



2.1.2 Ziele

In Abschnitt 1.1 (Seite 1) wurde schon einmal angesprochen, was die Motivation für dieses Projekt war. Im Projekt Scope wurde kurz auf die Ziele hingewiesen. Das Hauptziel des Projektes ist es aus strukturierten Websites ausgewählte Inhalte zu erfassen und aus diesen einen News-Feed zu erstellen. Die Inhalte werden dabei mit der Client-Software erfasst (Client, Seite 19) und aus dieser auf dem Server (Server, Seite 22) ein News-Feed erstellt. Es wird im Kontext des Projektes immer von einem News-Feed gesprochen. Ziel ist es, ein News-Feed des Typs „RSS“ zu erstellen. Es soll jedoch kein „Atom“[8] Feed erstellt werden! Der Grund dafür ist, dass RSS-Feeds einfacher Hand zu haben sind und weniger Elemente im eigentlichen Feed besitzen.

Alle Ziele kompakt aufgelistet:

- Erstellung von Client und Server Anwendung
- Client soll Daten erfassen, Server soll News-Feed daraus erstellen
- News-Feed soll vom Format „RSS“ sein

2.1.3 Umfang und Abgrenzungen

Umfang dieses Projektes ist es, einen Client, Server und eine Website zu erstellen (siehe Produktbeschreibung, Seite 15).

- Der Client soll dabei eine Erweiterung für den Webbrowser Mozilla-Firefox sein und dazu dienen, die Daten zu erfassen.
- Auf dem Server soll ein PHP-Skript laufen, welches die News-Feeds erstellt und bei Suchanfragen von der Website ggf. News-Feeds in der Datenbank sucht. Die ganzen Daten zum Erzeugen der News-Feeds werden in einer MySQL Datenbank abgespeichert. Dadurch ist es möglich bei Änderungen der Website dynamisch eine neue Version des News-Feeds zu erstellen.
- Auf der Website soll es nur ein Suchformular geben. Dadurch soll der Benutzer die Datenbank nach schon erstellten News-Feeds durchsuchen können. Falls nichts gefunden wird, soll der Benutzer die Möglichkeit erhalten über die Mozilla-Firefox Erweiterung selbst einen solchen News-Feed zu erstellen.

Dieses Projekt umfasst nicht:

- Erweiterungen für die Webbrowser *Internet Explorer*, *Opera*, *Safari*, *Chrome* ... zu erstellen. Der Grund hierfür: Mozilla-Firefox bietet ein einfaches Konzept und eine einfache Schnittstelle (Schnittstellen 2.1.4) für die Entwicklung von eigenen Erweiterungen. Außerdem ist Mozilla-Firefox ein

[8] *Atom: „Atom Syndication Format“* ist die moderne Version des RSS-Feeds → http://de.wikipedia.org/wiki/Atom_(Format)

weit verbreiteter Open-Source Browser. Ein weiterer Grund war, dass das Erstellen einer Erweiterung nichts Alltägliches ist, was im Kontext einer Vorlesung bisher noch nicht behandelt wurde.

- Server seitig soll keine Java Anwendung laufen. Der Grund hierfür ist, dass Java im Kontext einer Vorlesung schon behandelt wurde, PHP jedoch noch nicht. Ein weitaus wichtigerer Grund ist, dass in vielen Webhosting Angeboten im Internet ein Apache-Server mit PHP und MySQL-Datenbank Standard ist. In diesem Rahmen würde der Serverwechsel leichter fallen.

Der Umfang des Projektes kann aus der Abbildung „Produktstrukturplan“ auf Seite 15 entnommen werden.

2.1.4 Schnittstellen

Eine wichtige Schnittstelle, ohne welche dieses Projekt wohl kaum realisierbar wäre, ist die Erweiterungsschnittstelle von Mozilla-Firefox. Dieses stellt dem Entwickler alle Methoden zur Verfügung, die auch der Browser selbst benutzt. Bewußtes Vorgehen ist hier angebracht, ansonsten öffnet man Angreifen Tür und Tor! Innerhalb des Erweiterungssystem ist die DOM-Schnittstelle (DOM, Seite 5) sehr wichtig. Ohne diese wäre der Zugriff und die Manipulation der Elemente in einer Website nicht möglich!

2.1.5 Einschränkungen

Es soll, wie oben beschrieben (2.1.3), auf der Client-Seite nur eine Anwendung für Mozilla-Firefox entwickelt werden. Auf der Server -Seite sollen PHP und MySQL eingesetzt werden. Des weiteren gibt es zeitliche Einschränkungen von 4vier Monaten (Anfang März bis Anfang Juli, siehe Scope). Außerdem ist nicht vorgesehen, finanzielle Mittel auszugeben. Da in diesem Projekt nur Software entwickelt wird, ist dies kein Problem.

2.1.6 Voraussetzungen / Rahmenbedingungen

- Eine wichtige Voraussetzung ist ein Internet-Zugang, da dies ein Webprojekt ist.
- Ein Server im Inter- oder Intranet zur Erstellung der News-Feeds (siehe Server, Seite 22) muss vorhanden sein.
- Eine Website, aus welcher ein News-Feed erstellt werden soll, sollte dem W3C Standard genügen. Für Websites, die diesem Standard nicht entsprechen, ist nicht garantiert, dass die Erstellung eines News-Feeds erfolgreich ist.
- Die Website enthält keine Tabellenstrukturen.
- Die Website enthält eindeutig definierbare Elemente.
  `<h1 class="ueberschrift_1">...</h1>` → Dieses Element ist eindeutig

definiert, da es über den Tag-Namen `<h1>` und dessen Attribut `class="ueberschrift_1"` im Kontext der Website erreichbar ist.

- Zu erfassende Daten dürfen nicht dynamisch per Ajax o.ä. geladen werden! Solche Daten können vom Serverskript nicht erfasst werden.

2.2 Kunde(n)

„Kunden“ sind hier alle Personen, die aus einer Website einen News-Feed erstellen wollen (siehe Motivation). Dieses Projekt ist nicht für Unternehmen oder Website Administratoren gedacht! Diese betreiben die Website und haben dadurch Zugriff auf die Daten hinter der Website, d.h. sie haben die Möglichkeit, aus dem Datenbestand direkt einen News-Feed zu erstellen. Dies ist besser, da man dort direkten Zugriff auf die Daten hat und genau bestimmen kann, was in den News-Feed soll!

Da sich aber nicht jeder Website-Betreiber die Mühe macht, einen News-Feed in sein Webangebot einzubauen, soll mit diesem Projekt den Website-Benutzern eine Möglichkeit zur Verfügung gestellt werden selbst einen News-Feed mit den Daten der Website zu erstellen.

2.3 Abnahmekriterien

- Die Erweiterung muss
  - funktionsfähig sein.
  - bedienbar sein. Es sollte eine Übersichtsseite geben, mit allen angebotenen Funktionen der Erweiterung (siehe 4.4.1.1 „GUI“, Seite 19). Außerdem sollte der Vorgang in der Erweiterung für das Auswählen der Elemente und Einstellen der Eigenschaften in einem drei-Schritte-Vorgang ablaufen.
- Innerhalb einer Website können Strukturen erfasst werden unter den Bedingungen von 2.1.6.
- Der Server erstellt einen News-Feed mit den vorher gewählten Inhalten der Website.

2.4 Risiken

Siehe Risikomanagement auf Seite 30.

3 Projektlösungsansatz

3.1 Erforderliche Qualifikationen

In diesem Projekt wird hauptsächlich mit folgenden Dokument-Typen gearbeitet:

- XML hier XUL (evtl. auch HTML)
- Javascript
- PHP
- SQL
- CSS

Die Qualifikationen zum Bearbeiten solcher Dokumente bringt in der Regel heute jeder Webentwickler mit.

3.2 Vorhandene Infrastrukturen

Im Vorfeld muss nur ein Internetzugang und ein Computer vorhanden sein. Für das Projekt selbst müssen noch folgende Dinge installiert/eingerichtet werden:

- Mozilla Firefox Browser - in diesem läuft die Erweiterung (Client) und er dient zur Darstellung der Website.
- Server - falls nicht schon vorhanden, muss hier ein Webserver , wie z.B. Apache, eingerichtet werden. Außerdem müssen auf diesem noch PHP und die Datenbank MySQL[9] laufen.
- Es wird keine spezielle Entwicklungsumgebung benötigt. Es reicht ein einfacher Texteditor.

3.3 Wartungs- und Betriebsanforderungen

Betriebsanforderungen

- Der Server sollte eine geringe Downtime haben. Über diesen läuft später der ganze Datenverkehr - sowohl zur Erstellung von News-Feeds, als auch beim Abrufen schon erstellter News-Feeds. Dieser stellt einen wichtigen Knotenpunkt im Projekt dar.

[9] MySQL ist ein Relationales Datenbankverwaltungssystem auf Open-Source Basis → www.mysql.de/

- Die zwischengespeicherten Daten (Informationen zur dynamischen Erstellung der News-Feeds) sollten schnell verfügbar sein, deshalb wird hier die Datenbank MySQL eingesetzt.

- Um den Server zu entlasten, sollte nicht bei jeder News-Feed Anfrage[10] an den Server ein neuer News-Feed erstellt werden! Deshalb wird immer beim Aktualisieren eines News-Feeds ein Hash[11] der Website erzeugt und ein Zeitstempel in die Datenbank geschrieben. Dadurch wird sichergestellt, dass z.B. maximal alle 30 Minuten eine neue Version des News-Feeds erstellt wird oder es wird kein neuer erstellt, falls sich der vorher erzeugte Hash der Website nicht von der aktuellen Version unterscheidet.

Wartungsanforderungen

- Falls der Server (Hardware) gewartet werden muss oder ausfällt, wäre es optimal, einen Ersatzserver (ggf. an einem anderen Standort) zu haben.

- Bei Wartung der Server Anwendungen (Software), kann man diese erst einmal auf einem „Dummy"-System testen. Falls die Software dort optimal und ohne Fehler läuft, kann man den eigentlichen Server aktualisieren. Dies sollte jedoch am besten zu einer Zeit geschehen, wenn der Server wenig belastet ist.

- Die Wartung der Datenbank ist einfacher, wenn man das Interface „PHPMyAdmin"[12] auf dem Server eingerichtet hat. Dieses ermöglicht die einfache Wartung und hat darüberhinaus ein integriertes Backup-Tool zum Export und Sicherung der Datenbank.Mit Hilfe eines Skriptes kann dies im Hintergrund als Batch-Job laufen.

- Die Wartung oder besser die Aktualisierung der Mozilla Firefox Erweiterung ist einfach. In gepackter Form (XPI) enthält die Erweiterung einen Eintrag, auf welchem Server diese Datei liegt. Der AddOn-Manager von Firefox schaut nun regelmäßig nach, ob sich die Datei geändert hat. Man braucht nur eine aktuelle Version online zu stellen und Firefox lädt sich automatisch die neueste Version herunter.

3.4 Sicherheitsanforderungen

- Da dieses Projekt zum größten Teil im Internet abläuft, sollte auf eine verschlüsselte Übertragung bei wichtigen Daten geachtet werden.

[10] Ein Reader öffnet einen News-Feed. Dabei wird eine Anfrage an den Server geschickt. Dieser prüft ob die Website neue Informationen enthält und erstellt ggf. einen aktualisierten News-Feed.

[11] Ein Hash (hier: md5 Hash) ist eine eindeutige 32-stellige alphanumerische Zeichenkette. Falls sich in einer Website vom Inhalt nichts ändert, bleibt dieser Hash gleich!

[12] *„PHPMyAdmin"* ist eine freie PHP-Anwendung zur Wartung von MySQL Datenbanken → www.phpmyadmin.net/

- Die Datenbank muss vor externen - nicht qualifizierten - Zugriffen geschützt werden. Hier stecken die Informationen, aus welchen die News-Feeds dynamisch neu generiert werden. Falls hier etwas geändert wird (böswillige Absicht), könnten bei der nächsten Neuerstellung der News-Feeds z.B. Links zu Viren verseuchten Websites eingebunden werden. Dies muss auf jeden Fall vermieden werden!

- Das Serversystem muss komplett vor nicht autorisierten Zugriffen gesichert werden. Gründe hierfür wie beim vorigen Punkt.

- Die Client-Anwendung - die Firefox Erweiterung - darf nicht von Unbefugten verändert werden und die Verbindung beim Update sollte geschützt sein, Stichwort SSL[13]. Gründe: siehe vorige Punkte. Außerdem könnte durch ein infiziertes Update der ClientAnwendung eventuell Schadcode auf dem Client direkt ausgeführt werden.

3.5 Alternativen

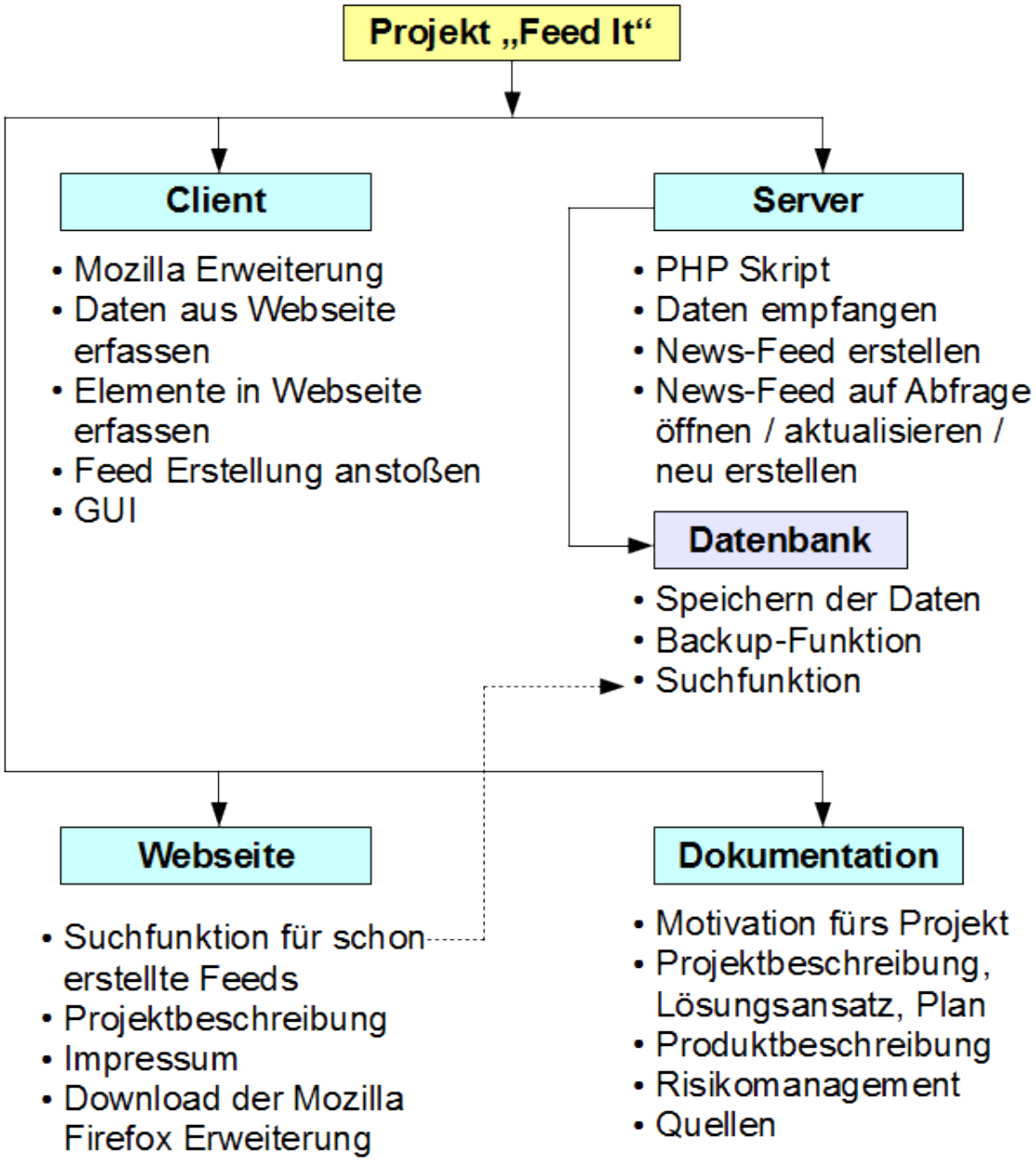
Eine Alternative wäre, dieses Projekt komplett in einer Hochsprache wie z.B. Java zu realisieren. Dabei könnte man die Client- und Server -Funktionen in einer Anwendung erstellen. Hierfür bräuchte man nicht unbedingt einen extra Server , da die ganze Logik in einer Anwendung steckt. Eine weitere Möglichkeit wäre, als Client eine Java Anwendung einzusetzen und als Server z.B. Java Server Pages (JSP) zu benutzen.

Diese Alternative wurden in Betracht gezogen, aber nicht umgesetzt. Wie schon in 2.1.3 Umfang und Abgrenzungen, Seite 9, erwähnt wurde Java in Vorlesungen schon behandelt, das Erweiterungssystem von Mozilla Firefox jedoch nicht. Aus diesen und den vorigen Gründen (siehe „Umfang und Abgrenzungen“, Seite 9) wurde das Projekt als Mozilla Firefox Erweiterung umgesetzt. Des weiteren sind schon gute bis sehr gute Vorkenntnisse im Bereich Webdesign und Entwicklung von Website vorhanden. Dies sind die erforderlichen Qualifikationen für dieses Projekt, siehe nächstes Kapitel.

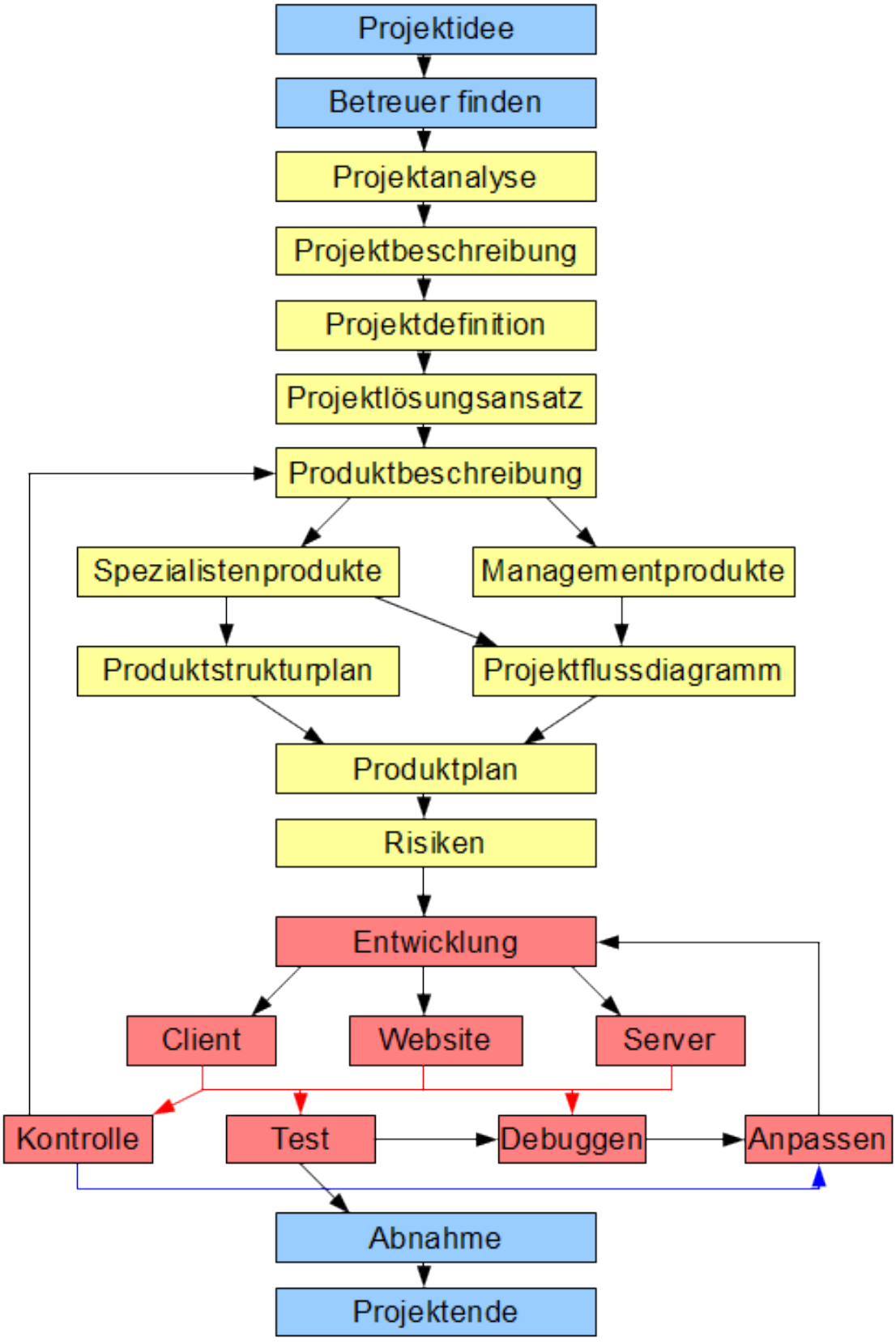
[13] Das „**S**ecure **S**ockets **L**ayer“ Protokoll dient zur Verschlüsselten Dateiübertragung zwischen zwei Gegenstellen.

4 Produktbeschreibung

4.1 Produktstrukturplan



4.2 Produktflussdiagramm



4.3 Managementprodukte

4.3.1 Einführung

In der Einführung wurde beschrieben, wie und aus welchen Gründen es zu diesem Projekt gekommen ist (Motivation, Seite 1). Außerdem wird dort beschrieben, was eigentlich RSS-Feeds (News-Feeds) - um welche es sich hauptsächlich in diesem Projekt handelt - sind. Dabei wurde kurz darauf eingegangen, wie man einen solchen Feed im Webbrowser anhand des Icons erkennt und wie ein solcher RSS-Feed aufgebaut ist (RSS-Feed, Seite 2). Gegen Ende der Einführung wurde ein weiteres wichtiges Thema dieses Projekts angesprochen - die Erweiterungen / das Erweiterungsframework von Mozilla Firefox. Dort wurde grundlegend beschrieben was Erweiterungen sind (Was sind Erweiterungen?, Seite 3) und wie man diese einsetzen kann, um Elemente im Kontext zu manipulieren (Overlays und das ID-Attribut, Seite 6).

4.3.2 Projektbeschreibung

Im Kapitel 2 wird das eigentliche Projekt beschrieben. Im Projektscope (Scope, Seite 8) erhält man einen kurzen und knappen Überblick über die Ziele und den Umfang des Projektes. Diese werden in den darauf folgenden Absätzen genauer erläutert. Des weiteren ist noch angegeben, für welche Kunden (Benutzergruppe) dieses Projekt von Nutzen ist und welche Abnahmekriterien es gibt.

4.3.3 Projektlösungsansatz

Im Projektlösungsansatz werden u.a. die alternativen Lösungsmöglichkeiten dieses Projektes erwähnt und warum die jetzige Lösung - mit Mozilla Firefox, Server und Website - gewählt wurde. Darauf folgen die nötigen Qualifikationen zum Lösen dieses Projektes. Ein weiterer wichtiger Punkt sind schon vorhandenen Infrastrukturen, da man auf diesen - Web- und DatenbankServer - aufbauen kann. In diesem Kontext werden sowohl die Wartungs- und Betriebsanforderungen erwähnt als auch die Sicherheitsanforderungen an das ganze System / Projekt erwähnt.

4.3.4 Produktbeschreibung

Die Produktbeschreibung ist in zwei Bereiche aufgeteilt: Die Managementprodukte und die Spezialistenprodukte.

Die Produkte beruhen auf der Projektmanagement-Methode von Prince2[14].

Managementprodukte beschreiben die ganze Organisation und die Theorie um das Projekt herum. Diese sind wichtig, da erst dadurch im voraus klar wird, ob das Projekt realisierbar ist. Während des Projektes dienen diese u.a.

[14] PRINCE2: „**Pr**ojects **in** **C**ontrolled **E**nvironments“ Projektmanagement-Methode die Management, Organisation und Steuerung eines Projektes behandelt. → http://de.wikipedia.org/wiki/PRINCE2

als Leitfaden. Nach dem Projekt sind sie nützlich um nachzuvollziehen, was überhaupt und wie gemacht wurde.

Spezialistenprodukte beschreiben die Produkte, welche programmiert werden müssen. Dies sind nach dem Projektabschluss die realisierten Produkte, welche das eigentliche Projekt darstellen. Dazu gehören u.a. die Mozilla Firefox Erweiterung (Client), die Anwendung, welche den News-Feed erstellt und verwaltet (Server), und die Oberfläche, um ggf. News-Feeds zu suchen oder Information zu dem Projekt zu erhalten (Website).

4.3.5 Projektplan

Im Projektplan steht der eigentliche Zeitplan des Projektes. Hier werden wichtige Termine und der Zeitaufwand erwähnt als auch die Methoden zur Abschätzung des zeitlichen Rahmens. Am Ende dieses Kapitels steht ein Gant-Diagramm, welches den zeitlichen Ablauf visualisiert.

4.3.6 Projektflussdiagramm

Im Projektflussdiagramm stehen die zeitlichen Abläufe der einzelnen Produkte untereinander. Was zuerst erledigt werden muss, welche Punkte darauf folgen und wo eventuelle Kontrollen und Korriguren stattfinden.

4.3.7 Risikomanagement

Im Risikomanagement stehen Risiken, welche im Projektverlauf eintreten können. Diese werden mit Nummern versehen, beschrieben, die Wahrscheinlichkeit des Auftretens angegeben, Gegenmaßnamen bei Auftreten und deren Status aufgelistet. Dadurch kann beim Auftreten eines der Risiken durch einen Blick in diese Liste gleich die entsprechende Gegenmaßname eingeleitet werden. Eine solche Liste ist gegenüber dem Auftraggeber eine Absicherung und erleichtert dem Projektmanager die Arbeit beim Eintreten eines solchen Risikos.

4.3.8 Fazit

In diesem Abschnitt steht das persöhnliche Fazit des Autors zu diesem Projekt

4.3.9 Quellen

Am Ende des Projektes stehen wichtige Quellenangaben und Verweise, welche für dieses Projekt verwendet wurden. Außerdem sind hier die verwendeten Programme / Tools aufgelistet.

4.4 Spezialistenprodukte

4.4.1 Client

In der ClientAnwendung soll der Benutzer einen News-Feed „zusammenklicken“ können. Dafür geht der Benutzer erst auf eine Website und ruft dann die Erweiterung auf (über das Kontext Menü oder über eine Tastenkombination). In der nun sich öffnenden Erweiterung kann man über einen Button „Neuen News-Feed erstellen“ wählen. Darauf erstellt man in drei Schritten einen News-Feed.

In den folgenden Abschnitten wird diese Vorgehensweise genauer beschrieben.

4.4.1.1 GUI

Die Hauptmenü-GUI der Erweiterung ist ähnlich wie folgende Grafik aufgebaut:

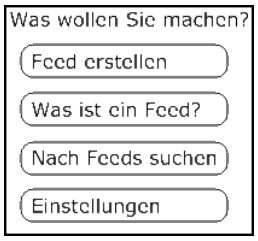


Abbildung 4.1: GUI Hauptmenü

Mit einem Klick auf „Feed erstellen“ gibt man in einem drei-Schritte-Vorgang Informationen über den News-Feed an, wählt die Elemente einer Website aus (aus dem der News-Feed später erstellt wird) und erstellt im letzten Schritt den eigentlichen News-Feeds. Dies soll in den drei folgenden Grafiken verdeutlicht werden:

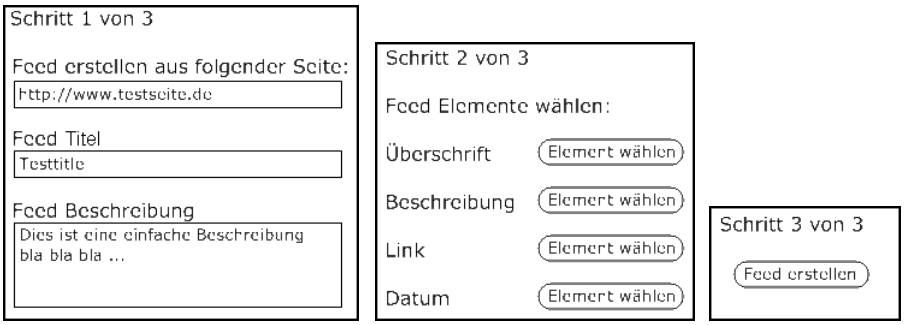


Abbildung 4.2: Schritt 1 bis 3 der News-Feed Erstellung

4.4.1.2 Logik hinter der GUI

GUI Schritt 1-3 Die nun folgenden Erläuterungen zu den Schritten beziehen sich auf die Abbildung 4.2 „Schritt 1 bis 3 der News-Feed Erstellung“.

Schritt 1 Beim ersten Aufruf wird im Schritt 1 das Formular automatisch ausgefüllt. So spart man dem Benutzer Tipparbeit. Dabei kommen die Daten direkt vom Browser. Die Website-Adresse (Feld 1) wird aus der Adresszeile gelesen, der Website-Titel wird in „Titel“ (Feld 2) geschrieben und die Beschreibung (Feld 3) kommt aus den Metadaten der Website.

Falls die Website evtl. unbeabsichtigt geschlossen wurde, kann mittels eines Buttons unterhalb des Feldes „Feed erstellen aus folgender Seite:“ die Website wieder geöffnet werden.

Zusätzlich gibt es einen Button, mit dem man die am Anfang dieses Absatzes beschriebenen Funktionen manuell auslösen kann. Dies ist von Vorteil, wenn man z.B. eine neue Seite aufgerufen hat und aus dieser nun einen News-Feed erstellen möchte.

Schritt 2 Im zweiten Schritt geht es um die eigentliche Elementauswahl. Hier drückt man auf einen Button „Element wählen“ und kann dann innerhalb der Website ein Element markieren. Hat man nun z.B. eine Überschrift markiert, so soll diese später im News-Feed auch als Überschrift eines Eintrages fungieren. Zusätzlich sollen die gewählten Elemente optisch markiert sein, um bereits gewählte Elemente erkennen zu können. Außerdem gibt es noch einen zusätzlichen Button hinter jedem „Element wählen“, um ein Element wieder abzuwählen, da dieses eventuell nicht mehr benötigt wird.

Schritt 3 Im letzten Schritt wird per Button-Klick der Mechanismus zum News-Feed Erstellen ausgelöst. Dabei wird ein Aufruf mit den zuvor gewählten Daten an den Server (4.4.2, Seite 22) geschickt. Falls der Vorgang erfolgreich war, wird ein Feld mit dem Link unterhalb des Buttons ausgegeben und ein zusätzlicher Button dahinter eingefügt, der den News-Feed öffnen kann. Falls Fehler auftreten, muss eine aussagekräftige Fehlermeldung ausgegeben werden.

UML Diagramm Hinter Schritt 1 & 2 verbergen sich wichtige Datenstrukturen. Die folgenden UML-Diagramme erläutern diese.

In **Schritt 1** werden die wichtigen Informationen des News-Feed erfasst - dessen Ursprungs-Webadresse, der Titel und die Beschreibung. Diese Daten werden in Objekten innerhalb der Erweiterung gespeichert. Objekte wurden verwendet, da man diese später einfacher erweitern kann.

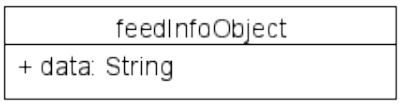


Abbildung 4.3: Objekt in Schritt 1

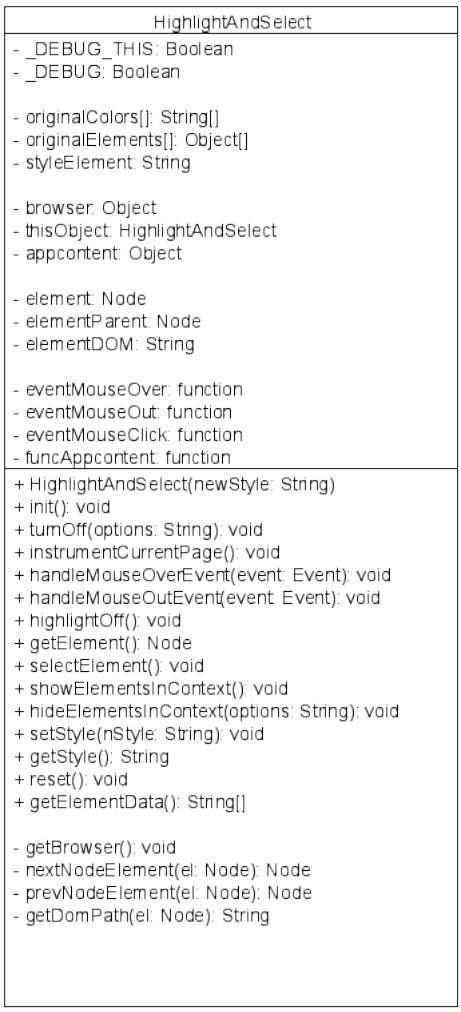


Abbildung 4.4: Objekt in Schritt 2

In **Schritt 2** werden die für den News-Feed wichtigsten Daten erfasst - die Elemente welche nachher die einzelnen Einträge im News-Feed darstellen. Da diese Elemente genau erfasst werden müssen, werden auch Daten aus dem Umfeld der Elemente selbst erfasst → elementParent. Außerdem wird der genaue Pfad im DOM zu diesem Element ermittelt, um später das Element im Kontext zu finden.

Mit „Element wählen“ ruft man die Methode *instrumentCurrentPage* in dem Objekt auf. Diese sorgt dafür, dass man Elemente anwählen kann per Klick (*handleMouseOverEvent*, *handleMouseOutEvent*, *selectElement*) und dass ähnliche Elemente auch farbig gekennzeichnet werden (*showElementsInContext*).

Nachdem ein Element gewählt wurde, muss dieses Objekt wieder deaktiviert werden. Dies ist erforderlich, da auf die Website EventListener angewendet wurden und diese nur für dieses Objekt Daten liefern (Element wählen etc.). Das Objekt kann per *turnOff* deaktiviert werden. Dabei werden noch weitere Methoden aufgerufen u.a. *highlightOff* und *hideElementsInContext*, um die farbig markierten Elemente wieder zurückzusetzen.

Im letzten **Schritt 3** werden die Daten letztendlich übermittelt. Dazu wird die Methode *getElementData* aufgerufen, welche die ganzen relevanten Information als String packt und dadurch für die Übermittlung vorbereitet.

4.4.1.3 Weitere Funktionen der GUI

In Abbildung 4.1 gibt es weitere Funktionen:

- Mit einem Klick auf den Button „Was ist ein Feed?“ wird man auf eine entsprechende Website weitergeleitet, welche wie in Kapitel 1.2 „RSS-Feed“ erklärt, was ein RSS-Feed ist. Diese Website kann entweder ein Teil der Projekt Website sein oder z.B. ein Verweis auf einen entsprechenden Eintrag bei Wikipedia.
- Die Methode hinter „Nach Feeds suchen“ soll auf die Projekt Website verweisen. Dort gibt es eine Methode zum Suchen von schon erstellten News-Feeds in der Datenbank (siehe 4.4.3 Website, Seite 24).
- Der vierte Button sieht einen Einstellungs- oder Eigenschafts-Dialog vor. In diesem können Einstellungen vorgenommen werden. Dieser Punkt wird optional gehalten, da er erst einmal nur das Debugging vereinfachen soll, durch Auswahl von entsprechenden Eigenschaften.

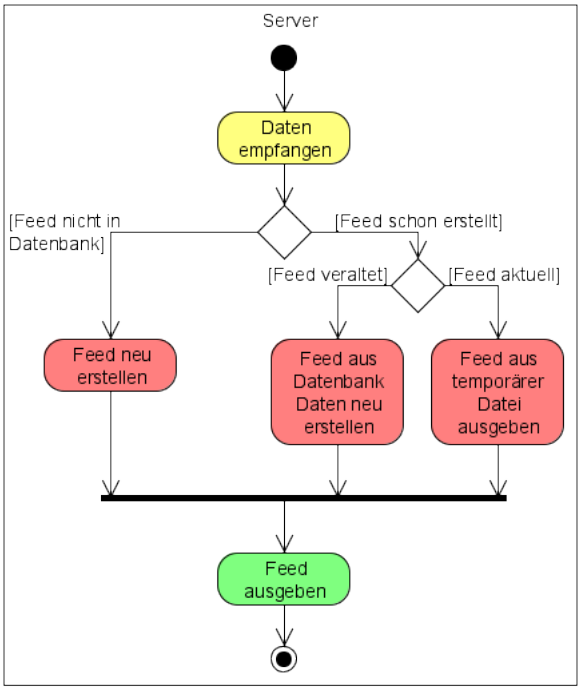
4.4.1.4 Anwendung

Zuerst war geplant die Anwendung in zwei Frames im Webbrowser ablaufen zu lassen - der eine greift auf die Inhalte des anderen zu. Somit spielt der eine Frame die Anwendung und der andere ist die Website, deren Inhalte zu einem News-Feed werden soll. Im voraus wurde aber schon festgestellt das dies nicht machbar ist! Siehe Risikomanagement Eintrag Nr. 1.

Deshalb wurde der Client als eine Mozilla Firefox Erweiterung entwickelt. Dies hat den Vorteil, dass man so „über“ der Website steht d.h., man hat mehr Möglichkeiten und Freiheiten auf den Kontext der Website zuzugreifen. Außerdem bietet Mozilla Firefox eine einfache Schnittstelle für die Entwicklung eigener Erweiterungen → 2.1.4 Schnittstellen, Seite 10.

4.4.2 Server

Auf dem Server sollen die eigentlichen Erstell- und Aktualisierungsprozesse ablaufen. Der Einfachheit halber werden diese beiden Vorgänge getrennt betrachtet. Die folgende Grafik soll die Vorgänge darstellen, welche auf dem Server ablaufen.



4.4.2.1 Erstellen von News-Feeds

Nachdem ein Benutzer in der Client-Anwendung (Client, Seite 19) einen News-Feed „zusammen geklickt“ hat, werden diese Informationsdaten an den Server übermittelt. Dieser überprüft zuerst, ob ihm die Daten

schon bekannt sind und bereits in der Datenbank liegen. Falls dies der Fall ist, wird mit „4.4.2.2 Neu-Erstellen / Aktualisieren von News-Feeds“ fortgefahren. Wenn die Daten „neu“ sind und noch kein News-Feed davor aus dieser Website erstellt wurde, wird als erstes eine leere RSS Hülle erstellt. Dabei wird per PHP ein XML-Dokument vom Typ RSS Version 2.0 erstellt. In dieses werden Metainformationen wie z.B. Titel, Beschreibung ... des News-Feed (siehe „Code Beispiel für einen RSS-Feed“, Seite 3) eingetragen.

Darauf folgt der wichtigste und komplizierteste Schritt: Die eigentliche Datenerfassung. Dabei wird die Ursprungs-Website nach den vorher in der Client-Anwendung definierten Elementen durchsucht und diese extrahiert. Diese Daten finden sich im XML-Dokument als `<item>` wieder. Nachdem jedes Element erfasst wurde wird ein News-Feed temporär zwischengespeichert (optimierter Zugriff), die übermittelten Daten in die Datenbank gespeichert und der News-Feed bzw. der Link zu diesem an die Client-Anwendung zurückgeliefert.

4.4.2.2 Neu-Erstellen / Aktualisieren von News-Feeds

Die Server Anwendung stellt fest, dass der News-Feed schon einmal erstellt wurde (Daten schon in der Datenbank vorhanden). Daraufhin wird geprüft, ob sich in der Ursprungs-Website etwas geändert hat. Falls ja, werden die Elementdaten, welche zuvor in der Datenbank gespeichert wurden, aus der Datenbank gelesen und daraus wie in 4.4.2.1 beschrieben ein neuer News-Feed erstellt oder besser gesagt, der schon vorhandene aktualisiert. Dieser Vorgang soll aus Performance Gründen nur in bestimmten Zeitintervallen ablaufen, weswegen ein Zeitstempel mit in die Datenbank geschrieben wird. Daraufhin wird auch hier der aktualisierte News-Feed entweder an die Client-Anwendung als Link zurückgeliefert oder direkt im Browser ausgegeben. Es kann aber auch sein, dass der News-Feed innerhalb dieses Zeitintervalls schon einmal (neu) erstellt wurde. Deswegen wird dieser dann nicht noch einmal erstellt, sondern nur der temporär zwischengespeicherte News-Feed zurückgeliefert.

Für eine detailliertere Beschreibung, wie ein News-Feed erstellt wird, siehe Kapitel „Fazit“ → 7.4.1 „News-Feed erstellen“ auf Seite 35.

4.4.2.3 Datenbank

Da die News-Feeds aktuell sein sollen, müssen sie auch regelmäßig aktualisiert werden. Dazu müssen aber die Daten, welche zur Neuerstellung notwendig sind, zwischengespeichert werden. Dazu bietet sich eine Datenbank an, wie die in diesem Projekt verwendetet MySQL Datenbank, auf welche man leicht per PHP zugreifen kann. Das verwendete Datenbankschema sieht wie in folgender Grafik (Abbildung 4.5 „Datenbank Schema“) aus:

Tabelle: RSS In dieser stehen die Informationen über einen News-Feed. Name, Beschreibung, URL der Ursprungs-Website, md5-Hash um zu überprüfen, ob es Änderungen gab, Dateiname für temporäre Dateien und ein Erzeugungsdatum.

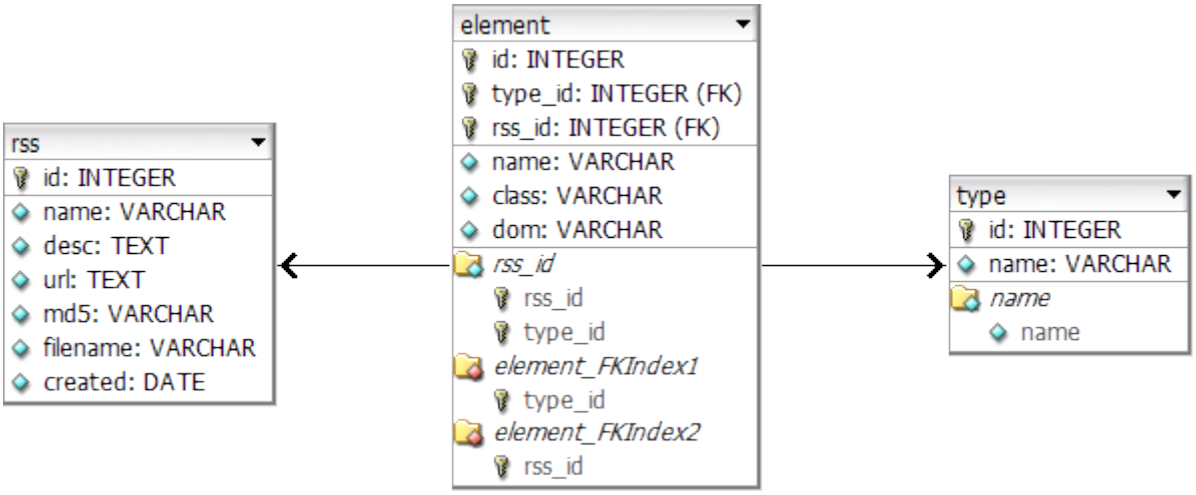


Abbildung 4.5: Datenbank Schema

Diese Tabelle benötigt nur einen Primärschlüssel, da es möglich sein soll, Feeds von gleichen Websites (mit unterschiedlichen gewählten Elementen) zu erstellen.

Tabelle: type In dieser stehen Namen von Elementtypen. Die Namen finden sich in der Client-Anwendung wieder. Die Elementtypen bezeichnen außerdem die Objekte, aus denen der News-Feed erstellt wird. Durch dieses Design ist es möglich, Elementtypen einfach hinzuzufügen / zu entfernen ohne dabei komplexe Zusammmenhänge in der Datenbank abändern zu müssen.

Tabelle: element Diese Tabelle enthält alle Elemente, welche bisher ausgewählt wurden. Hierzu werden der Name, die Stylesheet Klasse und der Pfad im DOM zu diesem Element abgespeichert. Außer einem Primärschlüssel enthält diese Tabelle noch zwei weitere Indizes: Zwei Fremdschlüssel auf die Primärschlüssel der Tabellen *RSS* und *type* und einen Unique Index auf die Schlüssel *RSS_id* und *type_id*. Dies bewirkt, dass nur ein Elementtyp zu einem RSS Feed zugeordnet wird. Es können also nicht zwei gleiche Typen zu einem News-Feed zugeordnet werden.

4.4.3 Website

Die Website soll nicht nur allgemeine Informationen über das Projekt, den Autor usw. enthalten, die Website soll auch eine Suchfunktion enthalten.

4.4.3.1 Aufbau

Die Website soll einfach gehalten werden und ähnlich wie Abbildung 4.6 aufgebaut sein. Oben stehen der Projekttitel und das Logo

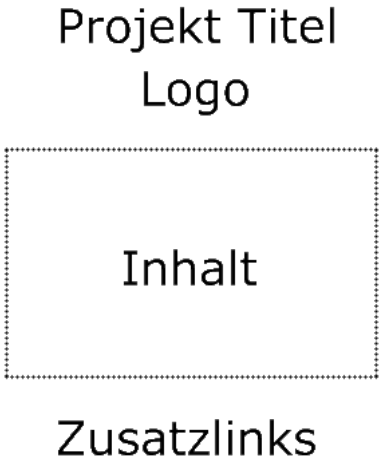


Abbildung 4.6: Website grober Aufbau

(siehe Deckblatt). In der Mitte (Inhalt) soll auf der Startseite die Suchfunktion angeboten werden. Auf weiteren Seiten soll hier der Inhalt geladen werden. Unterhalb des Inhaltes stehen zusätzliche Links. Ein „Home“-Button, „Über das Projekt“, „Download“ und „Impressum“. Zur besseren Bedienbarkeit wird auf der linken Seite noch ein zusätzliches Menü angezeigt.

4.4.3.2 Suchfunktion

Auf der Startseite ist die Suchfunktion zu sehen. Mit Hilfe dieser kann in der Datenbank nach schon erstellten News-Feeds gesucht werden. Dabei sollen schon bei der Eingabe grobe Informationen zu News-Feeds aus der Datenbank ausgegeben werden, ähnlich wie bei Google-Suggest[15]. Diese Funktion holt per Ajax[16] im Hintergrund Daten von dem Server . Dadurch soll nicht nur die Suche verbessert werden sondern auch eleganter wirken, da es Web-2.0-Stil ist. Nachdem die eigentlich Suchanfrage abgeschickt wurde gibt es zwei Möglichkeiten:

News-Feed gefunden: Die Suche in der Datenbank war erfolgreich. Die gefundenen News-Feeds werden nacheinander aufgelistet, eventuell mit Informationen über diese. Auf jeden Fall muss der Link zu diesen News-Feeds ausgegeben werden!

Keine Daten gefunden: Es wurden keine Treffer in der Datenbank gefunden. Man könnte nun entweder eine entsprechende Fehlermeldung ausgeben oder man bietet dem Besucher an, selbst einen News-Feed mit der Mozilla Firefox Erweiterung zu erstellen. Für letzteres wäre ein Tutorial angebracht, das den Benutzer bei der Erstellung anleitet und unterstützt.

Bisher wird nur eine Fehlermeldung ausgegeben. Ein Tutorial wird zu einem späteren Zeitpunkt noch implementiert.

4.5 Qualitätsmanagement

In dem Projekt gibt es kein explizites Qualitätsmanagement. Das Qualitätsmanagement setzt sich viel mehr aus den Punkten in den Kapitel 2.1.2, 2.1.3 und 2.3 zusammen. Die Ziele und Punkte, die dort definiert wurden, sollten umgesetzt werden.

Durch konsequente und erfolgreiche Umsetzung der Ziele entstand ein gut strukturierter und vor allem kommentierter Code, sodass eine hohe Qualität gewährleistet ist. Teilweise wurden Code-Reviews durchgeführt.

[15] Bei Eingabe eines Suchwortes werden entsprechende Stichworte, welche in einer Datenbank auf dem Server gesucht werden, in einer Liste unterhalb des Suchfeldes als Vorschläge eingeblendet.

[16] Ajax: *„**a**synchron **J**avaScript **a**nd **X**ML“*; Dadurch können Daten dynamisch und asynchron ohne erneutem Laden der Website vom Server abgerufen werden → http://de.wikipedia.org/wiki/Ajax_(Programmierung)

Zur Entwicklung wurden Debugging-Schnittstellen[17] verwendet. Kernfunktionen wurden in ihren Komponenten Funktions-getestet bevor sie zum Ganzen integriert wurden.

Begleitend wurde ein Managementtool verwendet. In dem Managementtool sind wichtige Punkte und Ziele definiert in Form von Tickets, welche erreicht werden müssen. Diese Tickets enthalten auch genauere Beschreibungen, was wann wo und wie gemacht und erledigt wurde. An diesen kann die zusätzlich investierte Zeit ablesen werden.Auch dieser Ansatz dient der Qualitätssicherung.

Das sich das Qualitätsmanagement mehr innerhalb des Managementtools abgespielt hat, wurde für das Qualitätsmanagement kein extra Kapitel beschrieben. Jedoch würde ein komplexeres Projekt mit mehreren Beteiligten und zusätzlichen Schnittstellen ein weitergehendes Qualitätsmanagement erfordern.

[17] Debugging-Schnittstellen in Firefox: *„Error Console“* → https://developer.mozilla.org/en/Error_Console, *„Firebug“* → https://addons.mozilla.org/en-US/firefox/addon/1843

5 Projektplan

5.1 Wichtige Punkte

In diesem Abschnitt werden wichtige Punkte im Projektablauf aufgelistet. Die komplette Übersicht findet sich unter 5.3 Gant-Diagramm.

Typ	Beschreibung	Zeit
Analyse	Hier wird analysiert, ob das Projekt wie geplant möglich ist und ob es mit den vorhandenen Ressourcen umsetzbar ist.	max. 1 Woche
Client	Umsetzung der Client Anwendung - dem Mozilla Firefox Plugin. Siehe Client, Seite 19.	6 Wochen
Server	Umsetzung der Server Anwendung - diese erstellt den eigentlichen News-Feed. Siehe Server, Seite 22.	1 Monat
Website	Erstellen einer Website zur angenehmeren Suche nach schon vorhandenen News-Feeds in der Datenbank. Siehe Website, Seite 24.	2-3 Wochen
Testen & Debuggen	Testen des Projektes - testen der Client-Server -Kombination	3-4 Wochen
Dokumentation	Schreiben der Projektdokumentation.	parallel zum Projekt
Abgabe/Ende	Abgabe der Dokumentation. Bis dahin müssen alle Tests und Bugfixes abgeschlossen sein!	2. Juli

5.2 Abschätzungen

Hier wird darauf eingegangen, wie die Zeit für die einzelnen wichtigen Punkte abgeschätzt wurde.

- Bei der **Analyse** gab es nicht viel abzuschätzen. Die Idee zu diesem Projekt kam in der zweiten Februarhälfte. Ab da an waren es noch 1-2 Wochen bis zum Beginn des Sommersemesters. In dieser Zeit wurde überprüft, ob das Projekt mit den mir vorhandenen Mitteln umsetzbar ist und ob es eventuell Komplikationen gibt wie z.B. Risiko Nr. 1 → „Risikomanagement“, Seite 30.

- Bei der **Client**-Anwendung war alles „Neuland“. Die Dateiformate wie XML (XUL), Javascript, CSS ... waren bekannt, aber die komplette Entwicklungsumgebung - Mozilla Firefox - war neu. Bei älteren Projekten, bei denen vieles „neu“ war, wurde die meiste Zeit mit dem Lesen von Dokumentation verbracht. Deshalb wurden hier 6 Wochen angesetzt.

- Bei der **Server** -Anwendung war es ähnlich wie bei der Client-Anwendung. Hier wurden viele neue Methodiken verwendet, wie z.B. das Erstellen von Dokumenten mit dem DOM[18] Modell direkt aus PHP, Zugriff auf Inhalte per XPath[19] oder das Bereinigen von Dokumenten mittels Tidy[20]. Außerdem wurde für die Server seitige Anwendung viel Zusatzzeit für Bugfixing, Testen und Optimierung eingeplant, da viele Websites weder valide (HTML inkorrekt) noch optimal aufgebaut und strukturiert sind! Deshalb wurde auch hier ein Monat für die Entwicklung angesetzt.

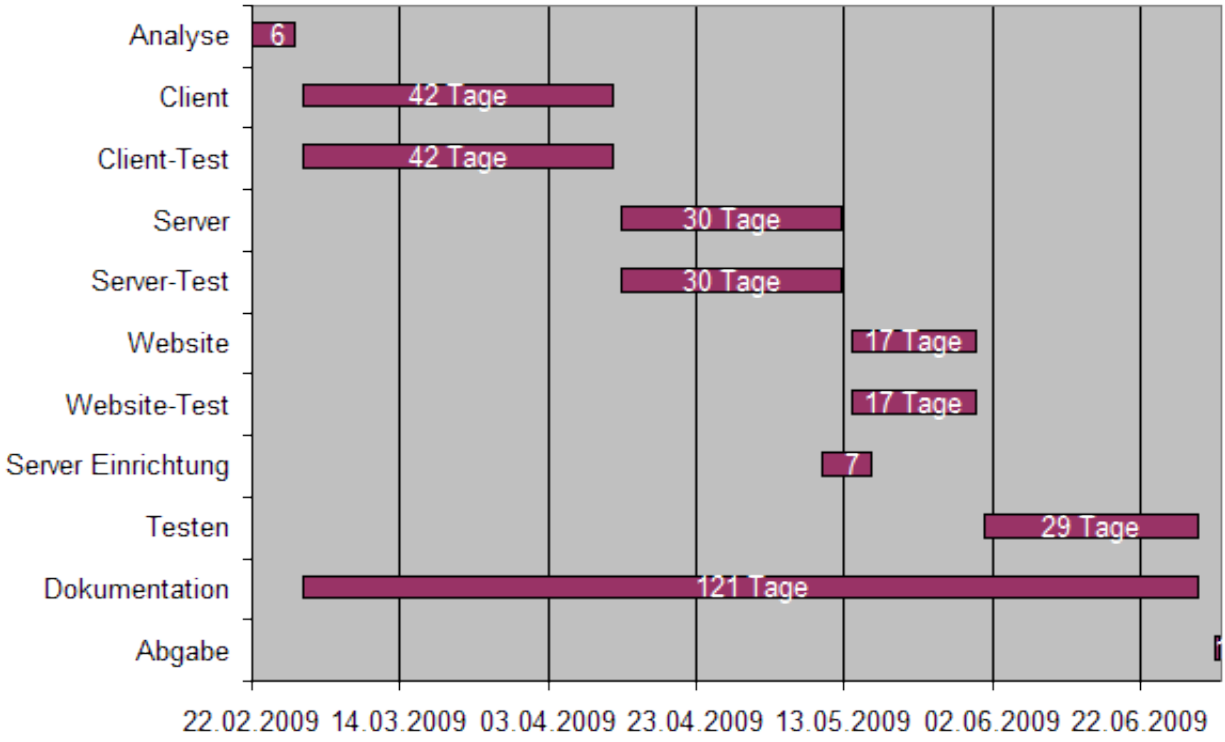
- Die **Website** soll nur Suchplattform für schon vorhandene News-Feeds sein. Diese sollen mittels einer einfachen Website in der Datenbank gesucht und ggf. aufgelistet werden. Außerdem soll eine Projektbeschreibung auf der Website stehen. Dies ist kein Hexenwerk und kann von einem Webentwickler ohne Probleme in der angesetzten Zeit von einer Woche realisiert werden. Da zusätzliche Gimmicks und Javascript Libraries verwendet werden, wird zur Sicherheit noch eine Woche mehr dazu geplant.

- Für das **Testen** / Debuggen wurden 4 Wochen angesetzt. Während der Entwicklung wurde schon getestet, am Ende soll aber noch einmal das Projekt als Gesamtes getestet werden und auftretende Fehler beseitigt werden.

- Die **Dokumentation** wurde parallel zum Projekt geschrieben.

[18] DOM siehe Seite 5
[19] XPath: *„XML Path Language“* → http://www.w3.org/TR/xpath
[20] Durch Tidy können HTML Dokumente bereinigt und ggf. valide Dokumente in korrekter HTML-Notation daraus erzeugt werden → http://www.w3.org/People/Raggett/tidy/

5.3 Gant-Diagramm



6 Risikomanagement

Nr.	Beschreibung	Auswirkungen auf das Projekt	Wahr.	Gegenmaßnahme	Status
1	Durch Cross-Site Scripting können die Frames nicht untereinander auf sich zugreifen.	Das Projekt sollte zuerst in einem Frame auf Website Ebene realisiert werden. Dieser Ansatz ist so nicht realisierbar	5	Keine Anwendung auf Website Ebene sondern eine Erweiterung in Mozilla Firefox entwickeln	erledigt
2	Server überlastet	Server könnte ausfallen.	3	Skripte optimieren falls nicht schon geschehen, ansonsten Server Hardware aufrüsten.	noch nicht eingetreten
3	Websites sind nicht valide, Daten können nicht erfasst werden	Keine Datenerfassung auf dem Server möglich, wodurch kein News-Feed erstellt werden kann.	3	Es wird in Kapitel 2.1.6 Voraussetzungen / Rahmenbedingungen definiert welche Website erfasst werden können. Ansonsten muss das Skript an neue Bedingungen angepasst werden.	eingetreten
4	Das System wird gehackt	News-Feeds könnten mit Viren oder mit Links zu Viren etc. verseucht werden.	1	System offline nehmen und bereinigen oder Ersatzsystem bereit halten.	offen
5	Einlesen in die Dokumentationen dauert zu lange	Stäniges Nachlesen von Informationen kann zur Vernachlässigung / Verzögerung des eigentlichen Projektes führen	3	Genügend Vorwissen mitbringen	erledigt
6	Ausfall eines Entwicklers	Falls der Entwickler ausfällt, würde das Projekt stehen!	2	weitere Entwickler	offen

Wahr. → Wahrscheinlichkeit für das Auftreten: **1**) unwahrscheinlich ($< 0,1\%$), **2**) geringe Wahrscheinlichkeit ($0,1 - 1\%$), **3**) möglich ($1 - 10\%$), **4**) wahrscheinlich ($10 - 50\%$), **5**) höchstwahrscheinlich ($> 50\%$)

7 Fazit

7.1 Projektmanagementtool

Das Projekt hat mir sehr viel Spaß gemacht und im Bereich Management weitergebracht. Es war neu für mich, ein Projekt von Grund auf zu planen und durchzuführen. Das habe ich früher schon gemacht, aber diesmal kam ein Projektmanagement[21] mit Ticketsystem, kalendarischer Übersicht und mit Wiki zum Einsatz. Durch dieses Tool wurde die Arbeit um einiges erleichtert, lief strukturierter ab und war zeitlich besser geplant. Ohne diese Mittel wäre die optimale Umsetzung des Projektes wohl nicht in der zeitlichen Begrenzung von 4 Monaten (Anfang März bis Ende Juni) möglich gewesen.

7.2 Findung der Projektidee

Es ist interessant, wie sich aus einer einfachen Idee ein Projekt entwickelt. Die Idee kam mir, wie schon in Kapitel 1.1 „Motivation“ erwähnt, beim Surfen im Internet. Ich wollte mich nicht mehr jedes Wochenende durch die gleichen Websites klicken um das Programm oder Veranstaltungen in der Umgebung ausfindig zu machen. Die betreffenden Websites besitzen keinen News-Feed (zum Zeitpunkt der Idee). Deshalb überlegte ich mir: *„Warum kann man nicht selbst aus den strukturierten Inhalten dieser Websites einen News-Feed erstellen?“*. Von dieser Idee geplagt machte ich mich auf die Suche im Internet, ob es so ein Tool gibt. Dabei habe ich sehr wenige Umsetzungen gefunden, die zudem nicht optimal arbeiten. Deshalb analysierte ich, ob dieses Projekt mit den Mitteln, welche mir zur Verfügung standen, umsetzbar wäre.

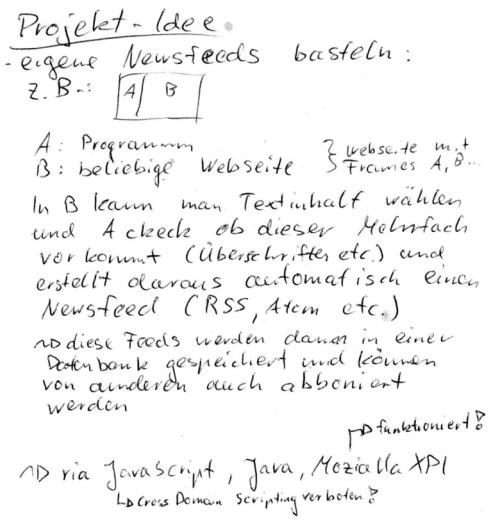


Abbildung 7.1: Erste Projektidee

[21] Eingesetztes Projektmanagement-Tool → www.redmine.org
Link zum Tool auf der HdM-Website → http://projekte.mi.hdm-stuttgart.de

7.3 Client

7.3.1 Probleme

Bei der Umsetzung des Client gab es einige Probleme. Lösbare Probleme, wie z.B. nicht vorhandenes Wissen, konnten durch Lesen von Dokumentationen beseitigt werden. Das Einlesen in die Dokumentationen von Mozilla Firefox war jedoch nicht immer einfach und ich konnte nicht auf Anhieb das finden, was ich suchte bzw. brauchte. Ein Buch „Firefox Erweiterungen für Dummys“ wäre hier nicht schlecht gewesen, letztendlich ging die Umsetzung aber auch so.

Weitere und zum Teil sehr Zeit raubende Probleme kamen im Zusammenhang mit der Programmiersprache Javascript. Javascript wird in den Erweiterungen für die Logik im Hintergrund verwendet.

7.3.1.1 Event-Listener

Eine Problem trat im Zusammenhang mit Event-Handlern auf. Einem Objekt können Events zugeteilt (*addEventListener*) und wieder entzogen werden (*removeEventListener*). Beim Versuch, einem Objekt erneut einen Event-Listener zuzuteilen, taucht ein Fehler auf: Es kann dem Objekt dieser Listener nicht erneut zugewiesen werden! Meine Vermutung war, dass es zu Fehlern kommen kann, wenn die Event-Listener innerhalb des Objekts angelegt und gelöscht werden. Es ist anscheinend keine korrekte Zuordnung mehr möglich. Dies scheint eine Eigenheit von Javascript zu sein, welche ohne weiteres nicht erklärbar ist. Hier ein Auszug aus dem Quellcode:

```
1  function HighlightAndSelect() {
2   [...]
3
4   /**
5    * turns off the event listeners
6    */
7   this.turnOff = function() {
8    browser.removeEventListener("mouseover",
9                                this.handleMouseOverEvent(event));
10   browser.removeEventListener("mouseout",
11                               this.handleMouseOutEvent(event));
12   browser.removeEventListener("click",
13                               this.selectElement(event));
14  }
15
16  [...]
17
18  /**
19   * instrument current page
20   */
21  this.instrumentCurrentPage = function() {
22   [...]
23
24   /*
25    * add event-listener to watch changes
```

```
26      */
27     browser.addEventListener("mouseover",
28                              this.handleMouseOverEvent(event));
29     browser.addEventListener("mouseout",
30                              this.handleMouseOutEvent(event));
31     browser.addEventListener("click",
32                              this.selectElement(event));
33   }
34
35   [...]
36 }
```

Der Lösungansatz besteht darin, dass die Event-Listener nicht mehr im Objekt selbst erstellt werden, sondern dieser Vorgang an eine Methode außerhalb des Objektes weiter gereicht wird. Der folgende Code soll die Lösung verdeutlichen:

```
1 addEvent = function(obj, event, fnc, bool) {
2   if(obj) {
3     obj.addEventListener(event, fnc, bool);
4   }
5 }
6
7 removeEvent = function(obj, event, fnc, bool) {
8   if(obj) {
9     obj.removeEventListener(event, fnc, bool);
10  }
11 }
```

7.3.1.2 Datenaustausch

Ein weiteres Problem war der Datenaustausch innerhalb der unterschiedlichen Dateien der Erweiterung. Dabei bin ich auf eine interessante Methode gestoßen, welche von vornherein schon in Firefox Version 3 integriert ist: Das *„Fuel Application Object“*.[22] Mit Hilfe dieses Hintergrundsystems in Firefox können Daten im Bereich einer Erweiterung zwischen gespeichert werden. Andere Erweiterungen haben dabei keinen Zugriff auf diese Daten, wodurch die Sicherheit erhöht wird! Der folgende Code-Auszug von der Entwicklerseite soll die Funktion dieses einfachen Systems erläutern:

```
1 Application.storage.set(keyname, data);
2
3 var data = Application.storage.get(keyname, default);
4
5 where:
6     keyname is a string used to identify the data
7     data is the data
8     default is the data value to return if keyname does not
          exists
```

[22] *„Fuel Application Object“* → https://developer.mozilla.org/en/Working_with_windows_in_chrome_code#Using_FUEL_Application_object

Dadurch wurde die lokale Datenspeicherung vereinfacht, ohne explizit eine Datenbank zu erstellen oder zu viele Daten auf andere Art und Weise preis zu geben.

7.3.2 Überladen von GUI Elementen

Ansonsten ist zum Mozilla Firefox Erweiterungssytem noch zu sagen, dass es für Webentwickler einfach ist, selbst Erweiterungen zu schreiben (siehe Kapitel 3.1 „Erforderliche Qualifikationen“, Seite 12). Es ist auch vergleichsweise einfach, eigene neue Elemente in die GUI einzubinden oder gegebene Elemente abzuändern / zu überschreiben (siehe „1.3.3“, Seite 6 und „1.3.4“, Seite 6).

Auf die Idee, selbst eine Erweiterung zu schreiben, bin ich durch einen Artikel in der Computerzeitschrift c’t[23] gekommen.

7.4 Server

7.4.1 News-Feed erstellen

Die Erstellung eines News-Feeds oder allgemein eines Dokumentes per PHP war etwas Neues und auch Interessantes. Zuerst legt man ein leeres DOM Dokument an („neues DOM Dokument“, blau). DOM deshalb, da dieses Modell einfache Methoden für den Zugriff auf Elemente bietet. In dieses leere Dokument wird im nächsten Schritt das eigentliche Dokument („Original Dokument“, gelb) - die Website - geladen, aus welcher ein News-Feed erstellt werden soll.

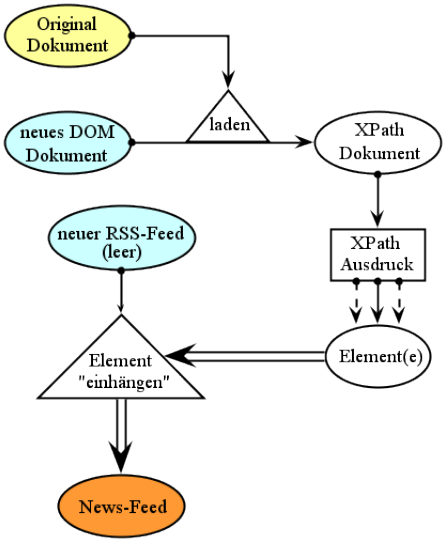


Abbildung 7.2: Erstellung eines News-Feeds

Aus diesem Vorgang resultiert ein XPath Dokument. Mit Hilfe von XPath Ausdrücken kann man sich Elemente oder Attribute zurückgeben lassen. Dieser Schritt erleichtert das Auffinden von Elementen anhand ihres Tag-Namens und ggf. an Hand eines Attributs. Dieser „Suchvorgang“ liefert beliebig viele Elemente, die dann überprüft werden, ob sie den zuvor gewählten entsprechen. Ein solches Element kann in einen zuvor erstellten, leeren News-Feed („neuer RSS-Feed“, blau) eingehängt werden. Dies geschieht auch per DOM. Durch DOM sind die Vorgänge mit wenigen Methodenaufrufen zu bewältigen.

Dadurch, dass in PHP (DOM) Dokumente erzeugt und diese mittels XPath

[23] c’t Artikel: „Reisegepäck - Firefox-Erweiterungen selbst entwickeln„ → http://www.heise.de/ct/Firefox-aufbohren--/artikel/133152

zerlegt werden können, ist es einfach, Inhalte aus einem Dokument (hier: Webseite) zu entnehmen und diese Inhalte in ein neues Dokument einzubinden.

7.4.2 Probleme

Auch wenn das Erstellen eines News-Feeds wie im vorigen Kapitel beschreiben recht einfach erscheint, gibt es doch einige Tücken. Wie schon in Kapitel 2.1.6 „Voraussetzungen / Rahmenbedingungen“ beschrieben, muss die Website, aus welcher der News-Feed erstellt werden soll, einige Bedingungen erfüllen. Leider hat sich gezeigt, dass viele Webseiten nicht valide (HTML konform) sind und somit die Erfassung von Elementen mittels XPath und per DOM sich als äußerst schwierig herausstellt. Eine Option war, die Seiten per HTML Tidy zu optimieren, also das HTML Dokument wieder valide zu machen. Dabei sind aber zum Teil zu viele Informationen (hier: Elemente, Inhalte) verloren gegangen. Deshalb wurde dieser Schritt erst einmal außer Funktion gesetzt.

Um dieses Problem zu umgehen, habe ich schließlich versucht, meine Algorithmen zu verbessern, um trotzdem nicht-valide Elemente erfassen zu können. Dies funktioniert auch soweit ganz gut, jedoch gibt es immer noch Websites die auch mit dem verbesserten Algorithmus nicht zu analysieren sind.

Ein weiteres sehr großes Problem sind Daten, die per Ajax in den Website Kontext geladen wurden. Bisher sehe ich keine Möglichkeit, solche Daten mit Hilfe eines Server-Skriptes zu erfassen.

Tabellen stellen ein weiteres Problem dar. Falls die Tabellen-Elemente sich nicht voneinander unterscheiden (unterschiedliche Attribute), gibt es bisher keine Möglichkeit, sie zu erfassen. Man könnte den Algorithmus jedoch in Zukunft so optimieren, sodass dieser die Tabellen-Elemente mitzählt. Dadurch kann man abschätzen, wo an welcher Stelle die für das Server-Skript relevanten Daten stehen.

7.5 Website

Bei der Erstellung der Website gab es keine großen Probleme, außer den üblichen Schwierigkeiten mit dem Internet Explorer. Dieser verweigert in den Versionen 6-8 die korrekte Darstellung von PNG Bildern und macht Probleme bei Abfragen per Ajax. Da diese Fehler, wie beim Internet Explorer üblich, nicht schnell zu lösen waren, habe ich mich auf wichtigere Dinge konzentriert. Dies war unter anderem die korrekte Suche nach News-Feeds in der Datenbank und die darauf anschließende Ausgabe eben dieser. Außerdem habe ich aus Usability Gründen am linken Rand ein zusätztliches Navigationsmenü eingerichtet.

Bei der Gestaltung habe ich Wert darauf gelegt, dass die Webseite einfach und schlicht ist. Man sieht das Wichtigste und mehr soll man auch nicht machen können. Die Webseite soll ein einfaches Suchportal auf der Startseite bieten, auf einer Unterseite den Download der Mozilla Firefox Erweiterung anbieten und unter FAQ eventuell auftretende Fragen beantworten.

Außerdem habe ich aus Performance Gründen die Standard Link-Funktion mit einer eigenen überschrieben. Bei aktiviertem Javascript werden die Seiten nicht komplett neu geladen, sondern nur die Inhalte der Unterseiten in die

Hauptseite geladen. Dies geschieht per Ajax. Falls einmal Javascript deaktiviert sein sollte, so funktioniert der Link wie üblich als „normaler“ Link und verweist auf die korrekten Unterseiten.

Diese Funktionen wurden mit Mootools[24] als auch mit jQuery[25] realisiert. Das sind Javascript Libraries, welche sehr einfach gehalten sind. Außerdem muss man sich bei diesen Libraries kaum noch Gedanken um die verschiedenen Browser machen, da intern eine automatische Browser-Anpassung geschieht.

7.6 Schlußfazit

Alles in allem hat dieses Projekt sehr viel Spaß gemacht und war eine angenehme Herausforderung. Außerdem habe ich dadurch interessante Einblicke in Techniken wie z.B. Erweiterungssystem in Firefox, Dokumenterstellung in PHP (DOM, XPath, Tidy) und Arbeiten mit News-Feeds bekommen, welche ich sonst so wahrscheinlich nicht eingesetzt hätte.

[24] Mootools → http://mootools.net
[25] jQuery → http://jquery.com

8 Quellenangabe

8.1 Internet

- http://de.wikipedia.org/wiki/Rss
- http://de.wikipedia.org/wiki/Mozilla_Firefox
- https://developer.mozilla.org/en/XUL_Overlays
- https://developer.mozilla.org/de/DOM
- http://www.firefox-browser.de/wiki/Chrome
- http://forums.mozillazine.org/ → Forum für Entwickler von Mozilla Anwendungen, Erweiterungen, etc.

8.2 Programme, Tools

- Eclipse IDE für AddOn-Packen (XUL-Runner), PHP, Javascript, HTML, CSS → www.eclipse.org
- Notepad++ Texteditor für viele Sprachen mit Syntax-Highlighting und Autovervollständigung → http://notepad-plus.sourceforge.net
- Apache Webserver → http://www.apache.org/
- Mozilla Firefox Browser → http://www.mozilla.org/

8.3 Sonstige

- Projektmanagement Unterlagen: Vorlesung *IT-Projektmanagement* bei *Dr. Iwanowski*
- LaTeX Wikibook: http://en.wikibooks.org/wiki/LaTeX/
- Eigene Präsentation zu „Erweiterungen in Mozilla Firefox“ in Design Patterns bei Prof. Kriha → http://www.kriha.de/krihaorg/patterncatalog.html

Stichwortverzeichnis

A

AddOn . . . . . *siehe* Firefox Erweiterungen, 38
Ajax . . . . . 11, 25, 36 f
Analyse 27
Anwendung 4, 6, 9 f, 13 f, 18 f, 22 ff, 27 f, 31
Apache 10, 12, 38

B

Betrieb 12
  Downtime 12
Browser . . 2 – 6, 9 f, 12, 17, 20, 22 f, 37 f
  Chrome . . . . . *siehe* Chrome (Browser)
  Firefox . . . . . *siehe* Firefox
  IE . . . . . *siehe* Internet Explorer
  Opera . . . . . *siehe* Opera
  Safari . . . . . *siehe* Safari

C

Chrome 5
Chrome (Browser) 9
Client . . . . . 9 f, 14, 18 f, 22 ff, 27 f
CSS 12, 28, 38

D

Datenbank . . . . 9 f, 12 ff, 17, 22 – 25, 27 f, 36
  MySQL . . . . . 9 f, 12 f, 23
Datenerfassung 23, 31
DOM . . . . . 5 f, 10, 21, 24, 28, 35 ff

E

eMail-Client . . . . *siehe* Thunderbird

F

Feed . 1 ff, 9 – 14, 17 – 25, 27 f, 31 f, 35 ff
  Atom 9
  Code Beispiel 3
  RSS . . . . . 1 ff, 9, 17, 22 ff
    Aufbau 3
Firefox . 2 – 6, 9 f, 12 ff, 17 f, 22, 25, 27 f, 31, 34 – 38
  Erweiterungen . . . . 3 ff, 11 – 14, 17 ff, 21 f, 25, 31, 34 ff, 38
    Beispiel . . . . . 6 f
  Plugins 4, 27
  Suchplugins 4

G

GUI 19

H

Hash 13, 23
  md5 . . . . . 13, 23
HTML 5, 12, 28, 38

I

Internet . . . . . 1 f, 10, 13, 32
  Zugang 10, 12
Internet Explorer 2, 9, 36

J

Java 10, 14
Javascript . . . . . 5 f, 12, 28, 33, 36 ff
  jQuery 37
  Mootools 37

M

Mozilla . . 3, 6, 9 f, 12 ff, 17 f, 22, 25, 27 f, 31, 35 f, 38
Firefox . . . . . *siehe* Firefox
Gecko Rendering Engine 4
MySQL . . . . . 9 f, 12 f, 23

O

Open-Source 1, 10, 12
Opera 2, 9
Overlay . . . . . 4 – 7

P

PHP . . . . . 9 f, 12 f, 23, 28, 35, 37 f

R

Risiken 11, 18
Risiko 18, 27

S

Safari 2, 9
Server 9 – 14, 17 f, 22 f, 25, 27 f, 31, 36
Software . . . . . 9 f, 13
Songbird 4
SQL . . . . . 9 f, 12 f, 23
SSL 14

T

Thunderbird 1, 4
Tidy . . . . . 28, 36 f

U

UML-Diagramm . . . . . 21 ff

W

W3C 6, 10
Wartung . . . . . 12 f
Webdesign 14
Website . 1 f, 4, 9 – 14, 17 – 25, 27 f, 31 f, 35 f

X

XML . . . . . 2, 4 f, 12, 23, 28
XPath . . . . . 28, 35 ff
XPI 5
XUL . . . . . 4 ff, 28, 38