Eine neue Client Technologie
für PAI Projekte?

Diplomarbeit

AJAX in the Enterprise

Studiengang:	Medieninformatik
Verfasser:	Knut Genthe
Vorgelegt bei:	Prof. Walter Kriha (1. Prüfer)
	Dipl. Ing. Gerrit Quast (2. Prüfer)
Bearbeitungszeitraum:	01.08.2006 – 31.01.2007

Abstract

Mit Rich Internet Applications wurde in letzter Zeit eine Lösung geschaffen, um reichhaltige, interaktive und ansprechende Benutzerschnittstellen für Webanwendungen zu entwickeln. RIAs können mit den verschiedensten Technologien entwickelt werden.

Seit geraumer Zeit ruft in diesem Umfeld die neue Technologie AJAX sehr viel Interesse hervor. Es liegt im Interesse von Daimler Chrysler neue Clientparadigmen zu erforschen.

Diese Arbeit untersucht, welche Vorteile, Risiken und Herausforderungen beim Einsatz von AJAX in Enterprise Umgebungen auftreten. Es werden diverser AJAX-Frameworks und die auftretenden Integrationsprobleme betrachtet. Außerdem werden die Vorteile von AJAX durch die Implementierung einer Beispielanwendung aufgezeigt.

Schlagwörter: AJAX, Sicherheit, Integration, PAI, J2EE, SiteMinder, Frameworks, JavaScript, Web 2.0, RIA, Thin Client, Rich Client, Infrastruktur

Inhaltsverzeichnis

Abstract **2**

Inhaltsverzeichnis **3**

Abbildungsverzeichnis **6**

Tabellenverzeichnis **8**

Abkürzungsverzeichnis **9**

1 Einleitung **11**

1.1 Motivation 11

1.2 Ziele der Arbeit 12

1.3 Strategie zur Zielerreichung 12

2 Grundlagen **13**

2.1 Web 2.0 13

2.1.1 Definition 13

2.1.2 Grundprinzipien 14

2.2 Einführung in AJAX 17

2.2.1 Warum AJAX? 17

2.2.2 Wie arbeitet AJAX? 20

2.2.3 AJAX Pioniere 25

2.3 Die Proactive Infrastukture 26

2.3.1 Hintergrund 26

2.3.2 Die Ziele von PAI 26

2.3.3 PAI Plattformen Konzept 27

3 Schichtenmodelle und Clientparadigmen **30**

3.1 Schichtenmodelle 30

3.1.1 2-Tier-Modell 31

3.1.2 3-Tier-Modell 31

3.1.3 n-Tier-Modell 33

3.2 Clientparadigmen 33

3.2.1 Thin Client 34

3.2.2 Rich Client 38

3.2.3 Rich Internet Applications 42

4 Kriterien der Clientparadigmen zur Erstellung einer Projektumfrage **45**

4.1 Aufbau des Kriterienkataloges 45

4.1.1 Funktionale Anforderungen 45

4.1.2 Nicht-funktionale Anforderungen 45

4.2 Einordnung der Kriterien 48
4.2.1 PAI Sichtweise 48
4.2.2 Projektsichtweise 49
4.3 Durchführung der Projektumfrage 49
4.3.1 Erstellung des Fragebogens 50
4.3.2 Ergebnis der Umfrage 50

5 AJAX im Detail 52
5.1 Problematiken 52
5.1.1 Browser 52
5.1.2 Entwicklung 53
5.1.3 Datenaustausch 58
5.2 Toolkits und Frameworks 60
5.2.1 Momentane Situation auf dem Markt 60
5.2.2 Anforderungen aus PAI Sicht 61
5.2.3 Warum sind Toolkits und Frameworks notwendig? 62
5.2.4 Standardisierung/Organisationen 64
5.2.5 Frameworkmodelle 65
5.2.6 Betrachtete AJAX Frameworks 75
5.2.7 Gegenüberstellung der Frameworks 78
5.2.8 Empfehlungen 79

6 AJAX und Sicherheit 80
6.1 Browsersicherheit 80
6.1.1 Browser Sicherheitsmodelle 80
6.1.2 Richtlinie des „Ursprungsservers“ 81
6.1.3 Zugriff auf Daten des Dateisystems 86
6.1.4 Benutzerschnittstelle als Sicherheitsrisiko 86
6.2 Angriffe auf AJAX-Anwendungen 87
6.2.1 JavaScript Schwachstellen 88
6.2.2 Häufige Attacken auf Webanwendungen 89
6.3 AJAX Anwendungen absichern 94
6.3.1 Absicherung des Transportweges 95
6.3.2 Sicherheit auf dem Client 95
6.3.3 Eine sichere Webschicht 96

7 Integration in die DCX Infrastruktur 98
7.1 SiteMinder 98
7.1.1 PAI Login Mechanismus 98
7.1.2 Session/Idle Timeout 100
7.1.3 Problem: Session/Idle Timeout mit AJAX 101
7.1.4 Lösungskonzepte 103
7.1.5 Zusammenfassung 108
7.2 Client Container 109
7.2.1 Einführung in den PAI Client Container 109

7.2.2 Client Container mit AJAX 112
7.3 Entwicklung einer Beispielanwendung 115
7.3.1 Hinter den Kulissen von Echo2 115
7.3.2 Admin Control Center Beispielanwendung 119
7.3.3 Zusammenfassung 123

8 Zusammenfassung und Fazit 124

Anhang A: Schnittstellen und Browserunterstützung 127
A1: Browserunterstützung 127
A2: API des XMLHttpRequest-Objectes 127

Anhang B: Relevante Browser Konfigurationen 129
B1: ActiveX im IE aktivieren 129
B2: IE Sicherheitsmodell 130

Anhang C: Model-View-Controller 131
C1: Prinzip von MVC 131
C2: MVC in normalen Webanwendungen 132

Ressourcen 133

Erklärung 143

Abbildungsverzeichnis

Abbildung 1: Mindmap zeigt die Prinzipien des Web 2.0 13
Abbildung 2: Synchrone Kommunikation mit traditionellen Webanwendungen 19
Abbildung 3: Asynchrone Kommunikation mit AJAX 19
Abbildung 4: Beispiel für den Aufbau einer XML-Datei 22
Abbildung 5: Browserunabhängige Erzeugung des XMLHttpRequest Objektes 22
Abbildung 6: Asynchrone Kommunikation mit dem XMLHttpRequest-Objekt 22
Abbildung 7: Aufbau eines HTML-DOM 23
Abbildung 8: Funktionsweise von AJAX 24
Abbildung 9: Screenshot von Google Maps 25
Abbildung 10: links: Infrastruktur vor PAI, rechts: Infrastruktur mit PAI 27
Abbildung 11: Die PAI Plattformen und ihre Abhängigkeiten voneinander. 28
Abbildung 12: Schichtenverteilung des 3-Tier-Modells 32
Abbildung 13: Historische Entwicklung der Clientparadigmen 34
Abbildung 14: Prinzip der Thin Client Architektur 35
Abbildung 15: Lebenszyklus von Thin Client Anwendungen 36
Abbildung 16: Prinzip der Rich Client Architektur 38
Abbildung 17: Lebenszyklus einer Rich Client Anwendung 39
Abbildung 18: RIAs vereinen die Vorteile von Rich und Thin Clients 42
Abbildung 19: Lebenzyklus von RIAs 43
Abbildung 20: PAI Sichtweise auf den Kriterienkatalog 49
Abbildung 21: Struktur PAI Fragebogen 50
Abbildung 22: Polling, an Beispiel einer AJAX-Anwendung 56
Abbildung 23: Comet, am Beispiel einer AJAX-Anwendung 56
Abbildung 24: Piggyback, am Beispiel einer traditionellen Webanwendung 57
Abbildung 25: Aufbau einer JSON Nachricht 59
Abbildung 26: Aufbau einer XML Nachricht 59
Abbildung 27: Marktsituation der Technologien für Webanwendungen 60
Abbildung 28: Anforderungen an ein Framework aus der PAI Sicht 61
Abbildung 29: Beziehungen zwischen Browser APIs, Toolkits und Frameworks 62
Abbildung 30: Anwendungsarchitektur von AJAX-Anwendungen 64
Abbildung 31: Architektur clientseitige AJAX-Frameworks 66
Abbildung 32: Clientseitige AJAX-Frameworks Single-DOM Ansatz 68
Abbildung 33: Clientseitiges AJAX-Framework Dual-DOM Ansatz 69
Abbildung 34: Architektur serverseitige AJAX-Frameworks 71
Abbildung 35: Serverseitiges AJAX-Framework Dual-DOM Ansatz 72
Abbildung 36: Architektur Multiplattform-Frameworks 74
Abbildung 37: Prinzip des UpdatePanel des Atlas Frameworks 78
Abbildung 38: Richtlinie des „Ursprungsservers“ 81
Abbildung 39: Firefox JavaScript-Konsole 83
Abbildung 40: Firefox Sichheitseinstellungen 83
Abbildung 41: Beispiel – Umgehen der Ursprungsserver Richtlinie mit dem PrivilegeManager 84

Abbildung 42: Firefox Sicherheitsdialog 84
Abbildung 43: Anwendungsszenario Proxys für Remote Dienste 85
Abbildung 44: Benutzerschnittstelle als Sicherheistrisiko 87
Abbildung 45: Angriff auf die Webseite der Bundesregierung 88
Abbildung 46: Überschreiben einer globalen JavaScript Funktion 89
Abbildung 47: links: clientseitiges XXS, rechts: serverseitiges XSS 90
Abbildung 48: Ungeschützte Bestandteile einer Webseite 91
Abbildung 49: Prinzip des Cross Side Request Forgery 92
Abbildung 50: Myspace JavaScript Wurm Samy 94
Abbildung 51: PAI Standartarchitektur 99
Abbildung 52: Session/Idle Timeout bei PAI 100
Abbildung 53: Session/Idle Timeout bei PAI mit AJAX 101
Abbildung 54: JavaScript Beispiel Session/Idle Timeout bei PAI 102
Abbildung 55: Bereitstellung eines Mechanismus durch das Framework 103
Abbildung 56: Lösung AJAX Session/Idle Timeout mit clientseitigen Frameworks 104
Abbildung 57: Beispielcode Lösung AJAX Session/Idle Timeout für clientseitige Frameworks 105
Abbildung 58: Lösung AJAX Session/Idle Timeout mit serverseitigen Frameworks 106
Abbildung 59: Integrationskosten der verschiedenen Frameworkmodelle 109
Abbildung 60: Laufzeitumgebung des PAI Client Container 110
Abbildung 61: Komponenten des PAI Client Container 110
Abbildung 62: Aufbau Client Container für clientseitige Frameworks 113
Abbildung 63: Client Container mit Multiplattform-Frameworks 114
Abbildung 64: Architektur vom Echo2 Framework 116
Abbildung 65: Beispiel für das Nachladen der Komponenten durch die AJAX-Engine 117
Abbildung 66: Initiale Webseite einer Echo2 Anwendung 118
Abbildung 67: Beispielanwendung zur Client und Server Synchronisation Echo2 118
Abbildung 68: XML Nachricht vom Client an den Server zur Aktualisierung des Modells 118
Abbildung 69: XML Nachricht vom Server an den Client zur Aktualisierung des Modells 119
Abbildung 70: Benutzerschnittstelle ACC aktuelle Version 120
Abbildung 71: ACC Beispielanwendung 123
Abbildung 72: ActiveX im IE aktivieren bzw. deaktivieren 129
Abbildung 73: IE Sicherheitsmodell 130
Abbildung 74: Prinzip von Model-View-Controller 131
Abbildung 75: MVC in normalen Webanwendungen 132

Tabellenverzeichnis

Tabelle 1: Gegenüberstellung der betrachteten AJAX-Frameworks 78
Tabelle 2: Browser Domänenverständnis 82
Tabelle 3: Use-Case Session/Idle Timeout bei PAI 100
Tabelle 4: Use Case AJAX Session/Idle Timeout bei PAI 102
Tabelle 5: Use-Case Lösung AJAX Session/Idle Timeout bei PAI für clientseitige Frameworks 105
Tabelle 6: Use-Case Lösung AJAX Session/Idle Timeout bei PAI für clientseitige Frameworks 107
Tabelle 7: Übersicht der ausgearbeiteten Schwächen des ACC 121
Tabelle 8: Übersicht der entwickelten Komponenten des ACC 122
Tabelle 9: AJAX Unterstützung der verschiedenen Browser 127
Tabelle 10: Methoden des XMLHttpRequest Objectes 127
Tabelle 11: Eigenschaften des XMLHttpRequest-Objectes 128

Abkürzungsverzeichnis

ACC	Admin Control Center
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
ATF	AJAX Toolkit Framework
BIB	Business Information Broker
CSS	Cascading Style Sheets
CMS	Content Management System
CPU	Central Processing Unit
CSRF	Cross Side Request Forgery
DCX	Daimler Chrysler
DIR	Directory Plattform
DOM	Document Object Model
DOS	Denial of Service Attack
EJB	Enterprice Java Beans
ERP	Enterprice Resource Planning
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IE	Internet Explorer
ISO	International Standards Organisation
J2EE	Java Platform, Enterprice Edition
JAAS	Java Authentication and Authorisation Service
JSF	Java Server Faces
JSP	Java Server Pages
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MVC	Model View Controller

PAI	Proactive Infrastructure
PC	Personal Computer
PHP	PHP: Hypertext Proprocessor
RAM	Random Access Memory
RAP	Rich AJAX Plattform
RCP	Rich Client Plattform
REST	Representational State Transfer Architektur
RIA	Rich Internet Applications
RSS	Realy Simple Syndication
RUP	Rational Unified Process
SBC	Server based Computing
SEC	Security Plattform
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSO	Single Sign On
SVG	Scalable Vector Graphics
SWT	Standard Widget Toolkit
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
UML	Unified Modeling Language
VM	Virtualle Maschine
W3C	World Wide Web Consortium
WAS	IBM Websphere Application Server
WLML	Weblogin Markup Language
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSS	Cross Site Scripting
XUL	XML User Interface Language
YUI	Yahoo! UI Library

1 Einleitung

Dieses Kapitel dient der Erläuterung der Aufgabenstellung und der Strategie zur Zielerreichung.

1.1 Motivation

Die Daimler Chrysler AG ist ein global tätiges Unternehmen, mit weltweiten Standorten. Dies hat eine sehr heterogene IT-Landschaft zur Folge. Die daraus entstehende Komplexität bedeutet, ohne ein strategisches Konzept, erhebliche Kosten und Aufwände für Softwareprojekte. Um die Komplexität zu vereinfachen setzt Daimler Chrysler (DCX) auf offene Standards und standardisierte Produkte.

PAI wurde gegründet, um eine standardisierte und zertifizierte Entwicklungsplattform bereitzustellen. Durch die Zentralisierung wird eine Bündelung von Wissen erreicht und den Projekten der nötige Service für die unterstützten Produkte bereitgestellt.

Diese Produkte sind vollständig in die DCX Sicherheitsinfrastruktur integriert, was besonders in Enterprise Umgebungen von großem Interesse ist. Somit ist die Komplexität der Infrastruktur für den Entwickler transparent und die notwendigen DCX Sicherheitsrichtlinien integriert und vorgegeben. Durch den Einsatz von PAI können Softwarelösungen risikoärmer und hochwertiger realisiert werden, woraus eine erhebliche Kostenoptimierung und Integrationsvereinfachung resultiert.

Zur Entwicklung von grafischen Benutzerschnittstellen werden zur Zeit die Thin und Rich Clientparadigmen unterstützt. Beide Paradigmen unterscheiden sich hinsichtlich ihrer Möglichkeiten interaktive Benutzerschnittstellen zu realisieren. Es ist jedoch fraglich, ob sie alle Anforderungen erfüllen können.

Es liegt daher im Interesse der Projekte und PAI neue Clientparadigmen zu erforschen. Mit Rich Internet Applications (RIA) wurde in letzter Zeit eine Lösung geschaffen, um reichhaltige, interaktive und ansprechende Benutzerschnittstellen für Webanwendungen zu entwickeln. RIAs können mit den verschiedensten Technologien entwickelt werden.

Seit geraumer Zeit ruft in diesem Umfeld die neue Technologie AJAX sehr viel Interesse hervor. Sie hat gegenüber anderen RIA Technologien den Vorteil, auf etablierten Webstandards zu basieren, sodass die Laufzeitumgebung auf dem Zielsystem vorausgesetzt werden kann.

Ziel dieser Diplomarbeit ist es zu erforschen, ob der Einsatz von AJAX in PAI sinnvoll ist.

1.2 Ziele der Arbeit

Zur Erreichung dieses Zieles werden folgende Vorgehensweisen betrachtet:

- Vorteile, Risiken und Herausforderungen beim Einsatz von AJAX in PAI
- Durchführung einer Projektumfrage zur Erforschung der Clientparadigmen von Projekten und deren Einsatzgebiete, um auf die Sinnmäßigkeit von AJAX zu schließen
- Betrachtung diverser Frameworks und Ableitung deren Einsatzgebiete
- Probleme bei der Integration von AJAX und die Entwicklung von Lösungen
- Implementierung einer Beispielanwendung, zur Aufzeigung der Vorteile von AJAX

1.3 Strategie zur Zielerreichung

Folgende Strategie wird zur Zielerreichung verfolgt:

- Kapitel 2 führt in die Grundlagen und Technologien ein, die sich hinter AJAX verbergen und stellt die PAI Plattform vor.
- Kapitel 3 beschreibt verschiedene Clientparadigmen, sowie deren Architekturen und die sich daraus ergebenden Vorteile und Probleme.
- Kapitel 4 erarbeitet einen Kriterienkatalog, mit dessen Hilfe eine Projektumfrage durchgeführt wird und erläutert die im Rahmen der Diplomarbeiten wichtigen Ergebnisse.
- Kapitel 5 geht grundlegenden Problemen die beim Einsatz von AJAX entstehen nach. Weiterhin werden diverse Frameworkmodelle, die Vorteile und beim Einsatz entstehenden Probleme betrachtet. Über die Einsatzgebiete dieser Frameworkmodelle wird eine Empfehlung abgegeben.
- Kapitel 6 beleuchtet die Sicherheit von AJAX-Anwendungen, wobei Browser-sicherheitsmodelle, verbreitete Angriffsmethoden und die Abwehrmöglichkeiten vorgestellt werden.
- Kapitel 7 betrachtet die Integration von AJAX in der DCX Infrastruktur. Es werden die aufgetretenen Probleme und deren Lösungsmöglichkeiten erläutert. Außerdem wird die im Rahmen der Diplomarbeit entwickelte Beispielanwen-dung vorgestellt.
- Kapitel 8 + 9 schließt die Diplomarbeit mit einer zusammenfassenden Beurteilung und Wertung über die Sinnmäßigkeit von AJAX in PAI ab.

2 Grundlagen

In diesem Kapitel werden die für das Verständnis der Diplomarbeit notwendigen Begrifflichkeiten und Technologien vorgestellt.

2.1 Web 2.0

Der Begriff Web 2.0 ist mittlerweile in Zeitschriften, Blogs oder Foren verstärkt präsent und wird viel diskutiert. Tim O'Reilly formuliert dazu in seinem Artikel ([OREIWEB20]) „What is Web 2.0" diverse Grundprinzipien, die das Wesen des Web 2.0 definieren sollen. Diese Grundprinzipien enthalten sowohl gesellschaftliche als auch technische Aspekte.

Abbildung 1: Mindmap[1] zeigt die Prinzipien des Web 2.0

2.1.1 Definition

Tim O'Reilly definiert den Begriff Web 2.0 wie folgt:

> "*Web 2.0 is the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to get better the more people use them. (This is what I've elsewhere called "harnessing collective intelligence.")*"

[1] http://kosmar.de/archives/2005/11/11/the-huge-cloud-lens-bubble-map-web20/

Nach seiner Meinung entwickelt sich das Internet von einer zusammenhanglosen Sammlung von Webseiten, hin zu einer Plattform, die die verschiedensten Anwendungen und Daten bereitstellt.

2.1.2 Grundprinzipien

Diese Grundprinzipien werden nachfolgend betrachtet und mit konkreten Beispielen untermauert.

2.1.2.1 Die Nutzung kollektiver Intelligenz

Im Web 2.0 wandelt sich die Rolle des Benutzers vom Konsumenten, hin zum Produzenten. Er kann Kommentare hinterlassen, Webseiten miteinander verlinken oder Wissen publizieren.

So basiert die freie Online-Enzyklopädie Wikipedia[2] auf einer ungewöhnlichen Idee, dass jeder Eintrag von jedem Benutzer bearbeitet werden kann. Hierbei wird großes Vertrauen in die Benutzer gesetzt, qualitativ hochwertige Beiträge zu verfassen und Beiträge mit geringer Qualität zu verbessern oder ergänzen.

Ein anderes Beispiel ist Flickr[3], das dem Benutzer die Möglichkeit gibt digitale Bilder einfach und intuitiv zu verwalten, sowie Anderen bereitzustellen. Flickr setzt auch das so genannte Tagging ein, um die Bilder mit zusätzlichen Metadaten anzureichern und somit die Auffindbarkeit zu verbessern. Denn ohne zusätzliche Information wären die Bilderwelten nicht zu erfassen und vom Anwender nicht nutzbar.

2.1.2.2 Daten als nächstes „Intel Inside“

Jede umfangreiche Internet Anwendung basiert auf einer spezialisierten Datenbank, z.B. Amazon’s Produktdatenbank[4], MapQuest’s Kartendatenbank[5] oder Napster’s Songdatenbank[6]. Die Erweiterung, Aktualisierung und Verarbeitung dieser Daten stellt einen wichtigen Wettbewerbsfaktor dar.

Amazon hat wie sein Mitbewerber Barnes and Noble[7] die ursprüngliche Datenbasis vom ISBN Registrar R.R. Bowker[8] eingekauft. Jedoch wurde diese von Amazon kontinuierlich um Cover, Inhaltverzeichnisse, Samples und Benutzerkommentare erweitert. Heute ist Amazon die wichtigste Quelle von biographischen Daten für Bücher, eine Referenz für Schüler, Studenten und natürlich Käufer.

---

[2] http://de.wikipedia.org/
[3] http://www.flickr.com/
[4] http://www.amazon.com/
[5] http://www.mapquest.com/
[6] http://www.napster.com/
[7] http://www.barnesandnoble.com/
[8] http://www.bowker.com/

Durch kontinuierliche Weiterentwicklung der Datenbestände hat sich Amazon eine einzigartige Marktposition geschaffen.

Viele Datenanbieter stellen Schnittstellen bereit, die von Programmierern kostenlos genutzt werden können. Dadurch entstehen viele so genannte Mashups[9], die verschiedene über das Internet frei verfügbare Datenquellen miteinander kombinieren. Ein Beispiel dafür ist „Housingmaps[10]“, das Google Maps[11] mit dem Immobilienmarkt von Craigslist[12] verbindet.

Ein Problem könnte sein, dass Unternehmen mit einzigartigen Datenquellen versuchen sich eine Monopolstellung aufzubauen, aber auf der anderen Seite entwickeln sich immer mehr freie Datenbestände. Ein Beispiel hierfür ist Wikipedia, deren Artikel häufig unter der Creative Commons Lizenz ([CreCoLiz]) zur Verfügung stehen.

2.1.2.3 Abschaffung des Software-Lebenszyklus

Die Betriebsabläufe werden zur Kernkompetenz von Unternehmen im Internet. Die Webdienste müssen ständig aktualisiert, gewartet und verbessert werden, damit sie ihre Leistungsfähigkeit nicht verlieren.

So muss Google[13] ständig das Web durchsuchen, seine Datenbanken aktualisieren, Link-Spam herausfiltern, Millionen von Suchanfragen beantworten und zum Inhalt passende Werbung einblenden.

Webdienste tragen bewusst über Jahre hinweg das Beta-Logo, um dem Benutzer die stetige Weiterentwicklung zu suggerieren. Dadurch ergibt sich die Möglichkeit den Benutzer aktiv in den Entwicklungsprozess, sei es durch Testen oder Verbesserungsvorschläge, mit einzubeziehen. Durch die Analyse des Benutzerverhaltens kann z.B. festgestellt werden, ob ein neues Feature angenommen oder abgelehnt wird. Somit ist es möglich den Webdienst wesentlich gezielter weiter zu entwickeln.

Flickr ist ein sehr extremes Beispiel, bei dem neue Versionen manchmal innerhalb einer halben Stunde veröffentlicht werden. Doch generell haben fast alle Web 2.0 Unternehmen einen Entwicklungszyklus, der sich von dem der PC- oder Client-Ära unterscheidet.

2.1.2.4 Software über die Grenzen einzelner Geräte hinaus

Jede Webanwendung kann als geräteunabhängig bezeichnet werden und kann auf jedem Gerät (z.B. PC, Mobile) ausgeführt werden, das einen Browser zur Verfügung stellt.

---

[9] Der Mashup bezeichnet die Erstellung neuer Inhalte durch die nahtlose (Re-)Kombination bestehender Inhalte.

[10] http://www.housingmaps.com/

[11] http://maps.google.de/maps

[12] http://www.craigslist.org/about/cities.html

[13] http://www.google.de

Doch wenn das Web 2.0 als Plattform betrachtet wird, ergeben sich eine Vielzahl von neuen und interessanten Möglichkeiten.

Ein populäres Beispiel stellt iTunes[14] dar. Hier haben die Entwickler den nahtlosen Übergang vom mobilen Endgerät, über ein Web-Backend, bis hin zum PC als lokale Kontrollstation geschaffen.

In diesem Bereich sind die größten Änderungen zu erwarten, denn was ergeben sich erst für Möglichkeiten, wenn die verschiedensten Endgeräte (z.B. Telefon, Auto) an die Plattform angeschlossen sind und ihre Daten zur Verfügung stellen.

2.1.2.5 Leichtgewichtige Programmiermodelle

Die Erfolgsstory des Internet wurde durch seine Einfachheit ermöglicht. Anstatt komplexe Webservices einzusetzen, wird der Wunsch nach einfachen und leichtgewichtigen Schnittstellen laut.

Es geht darum, die Daten über einfache Schnittstellen auszutauschen und zu verteilen, nicht darum zu kontrollieren was am anderen Ende der Leitung mit ihnen geschieht. Dadurch wird die uneingeschränkte Verbreitung der Daten ermöglicht.

Die Schnittstellen sollten so angeboten werden, damit sie möglichst einfach wieder verwendet werden können. Dadurch wird es möglich, die verschiedensten Schnittstellen zu kombinieren und einen neuen Nutzen daraus zu generieren.

2.1.2.6 Benutzerführung

Bereits vor einigen Jahren sollten Technologien etabliert werden, die das Web um aktive Benutzerelemente anreichern und somit die Erstellung von Benutzeroberflächen ermöglichen, die man von Desktop-Anwendungen gewohnt ist. Java-Applets und Flash waren solche Technologien, die aber den Durchbruch nie geschafft haben, weil sie eine spezielle Laufzeitumgebung benötigen. Außerdem standen JavaScript und DHTML als clientseitige Scriptsprachen zur Verfügung, um eine bessere Benutzerführung zu ermöglichen. Doch all diese Ansätze konnten sich nicht durchsetzten und führen seitdem ein Nischendasein.

Das wahre Potential des Webs in Bezug auf die Bereitstellung von vollwertigen Benutzeroberflächen wurde erst mit der Einführung von Gmail[15] und Google Maps[16] bekannt. Diese beiden Webanwendungen stellen eine Benutzeroberfläche bereit, die sich hinsichtlich der Interaktionsmöglichkeiten kaum von Desktop-Anwendungen unterscheidet. Zur Realisierung wurde eine Technologie verwendet, die schließlich auf den Namen AJAX getauft wurde.

---

[14] http://www.apple.com/de/itunes/

[15] http://gmail.google.com/

[16] http://maps.google.de/

AJAX ist bereits die Schlüsseltechnologie einer Vielzahl von Webanwendungen wie z.B. Flickr, Gmail.

2.2 Einführung in AJAX

Nachdem im Abschnitt 2.1 die Ideen vermittelt wurden, die hinter dem Begriff Web 2.0 stehen, soll in diesem Abschnitt eine der Schlüsselkomponenten vorgestellt werden. Asynchronous JavaScript and XML (AJAX) ist ein neuer Begriff, der von Jesse James Garrett von Adaptive Path ([JamGarr05]) geprägt wurde.

Mit AJAX ist es möglich interaktive Webseiten zu entwickeln. Die Idee dabei ist es, dass bei der Benutzerinteraktion mit der Webanwendung die Webseiten nicht ständig neu geladen werden, sondern im Hintergrund nachgeladen und aktualisiert werden. Dadurch verbessert sich die Benutzerfreundlichkeit von Webanwendungen enorm. Hieraus ergeben sich neue Möglichkeiten zur Entwicklung von Webseiten, an die vor einiger Zeit niemand gedacht hätte.

Im folgenden Abschnitt soll deshalb darauf eingegangen werden, was AJAX-Anwendungen von normalen Webanwendungen unterscheidet und welche Technologien sich hinter dem Begriff verbergen.

2.2.1 Warum AJAX?

In diesem Abschnitt sollen die Schwachstellen von normalen Webanwendungen angesprochen werden. Außerdem wird betrachtet, welche Möglichkeiten AJAX bietet, diese Schwachstellen zu beseitigen.

2.2.1.1 Benutzerschnittstelle

Eine Benutzerschnittstelle sollte im Idealfall für den Benutzer unsichtbar sein, ihn unterstützen und die Arbeit vereinfachen.

Desktop-Anwendungen kommen dieser Vorstellung schon sehr nahe. Sie stellen dem Benutzer reichhaltige Interaktionsmöglichkeiten bereit, wie z.B. Drag&Drop, Fenster, Dialoge, Text markieren oder formatieren.

Doch betrachtet man im Gegenzug eine Webanwendung, werden dem Benutzer zwar reichhaltige und dicht miteinander verzahnte (z.B. personalisierte Daten) Informationen präsentiert, aber die einzige Form der Interaktion mit der Anwendung besteht darin, auf Links zu klicken und Formulare auszufüllen. Falls sich der Inhalt der Webseite verändert, muss sie komplett neu geladen werden.

Durch den Einsatz von AJAX erhält der Anwender reichhaltige, interaktive und ansprechende Benutzeroberflächen, wie er sie von Desktop-Anwendungen kennt.

2.2.1.2 Netzwerklatenz

Das Internet besteht aus einer Vielzahl von verteilten Anwendungen, die ihre bereitgestellten Services gegenseitig nutzen können und von überall aus aufgerufen werden können. Zwischen diesen Punkten existieren Netzwerkverbindungen, die Verzögerungen verursachen.

Wenn jetzt eine Anfrage (Request) an ein entferntes System gesendet wird, setzt sich ein komplexer Mechanismus in Gang. Die Daten müssen ausgewertet, die Antwort zusammengestellt und an den Client (Response) zurück geschickt werden. Der Client muss wiederum die Daten auswerten und kann danach mit der Darstellung der Inhalte beginnen. Der Vorgang ist sehr komplex und zeitintensiv.

Diese Problematik macht sich bei der Bedienung von Webanwendungen deutlich bemerkbar, denn der Benutzer ist von Desktop-Anwendungen gewohnt möglichst schnelle Reaktionen auf die getätigten Aktionen zu bekommen. Jedoch kann sich bei Webanwendungen schon eine geringe Verzögerung auf den schnellen Seitenaufbau durch den Browser auswirken und den Benutzer irritieren oder stören. Ein Grund für die oft fehlende Benutzerfreundlichkeit von Webanwendungen ist die Tatsache, dass die Netzwerklatenz nicht vorhersehbar ist.

Mit AJAX werden Teile der Anwendungslogik an den Client übertragen und ermöglichen somit die Entkopplung der Benutzerschnittstelle von der Netzwerklatenz.

2.2.1.3 Asynchrone Kommunikation

In Abschnitt 2.2.1.2 wurde festgestellt, dass der Entwickler davon ausgehen sollte, dass jede Interaktion mit dem Server zeitaufwändig ist. Dem Entwickler steht nur ein Mittel zur Verfügung um das Problem der Netzwerklatenz zu lösen. Die Benutzerschnittstelle muss von den Anfragen über das Netzwerk entkoppelt werden. Aber wie kann der Entwickler das umsetzen? Der Weg der Entkopplung ist asynchrone Kommunikation. Bevor die Problematik weiter diskutiert wird, folgt eine Definition der Begriffe synchrone- und asynchrone Kommunikation.

Asynchrone Kommunikation ([ASYN])*:* Unter asynchroner Kommunikation versteht man in der Informatik und Netzwerktechnik einen Modus der Kommunikation, bei dem das Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses durch bspw. Warten auf die Antwort des Empfängers (wie bei synchroner Kommunikation der Fall) stattfindet.

Synchrone Kommunikation ([SYN])*:* Unter synchroner Kommunikation versteht man in der Informatik und Netzwerktechnik einen Modus der Kommunikation, bei dem die Kommunikationspartner (Prozesse) beim Senden oder beim Empfangen von Daten immer synchronisieren, also warten (blockiert), bis die Kommunikation abgeschlossen ist.

Das Problem ist, dass in normalen Webanwendungen ausschließlich synchrone Kommunikation eingesetzt wird. Die Daten zwischen Client und Server werden mit dem

Hypertext Transfer Protocol (HTTP) übertragen. HTTP ist ein Anfrage-Antwort-Protokoll, d.h. der Client stellt eine Anfrage (Response) und der Server antwortet (Request) darauf. Es verläuft nur in eine Richtung, sprich der Client kann zwar mit dem Server Kontakt aufnehmen, aber nicht umgekehrt.

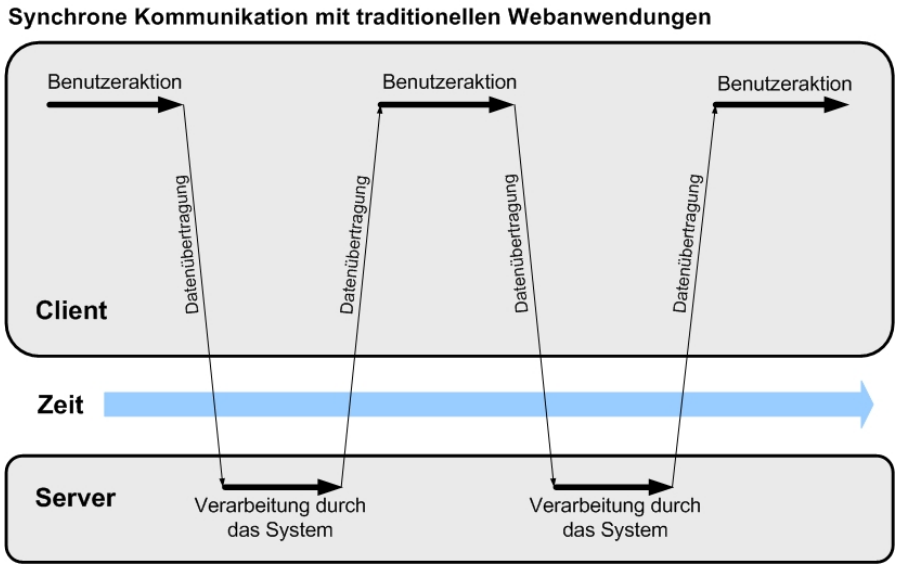

Abbildung 2: Synchrone Kommunikation mit traditionellen Webanwendungen

Aber bei asynchroner Kommunikation ist es notwendig, dass der Client zweimal benachrichtigt wird und zwar wenn die Anfrage gesendet wurde und die Antwort eingetroffen ist.

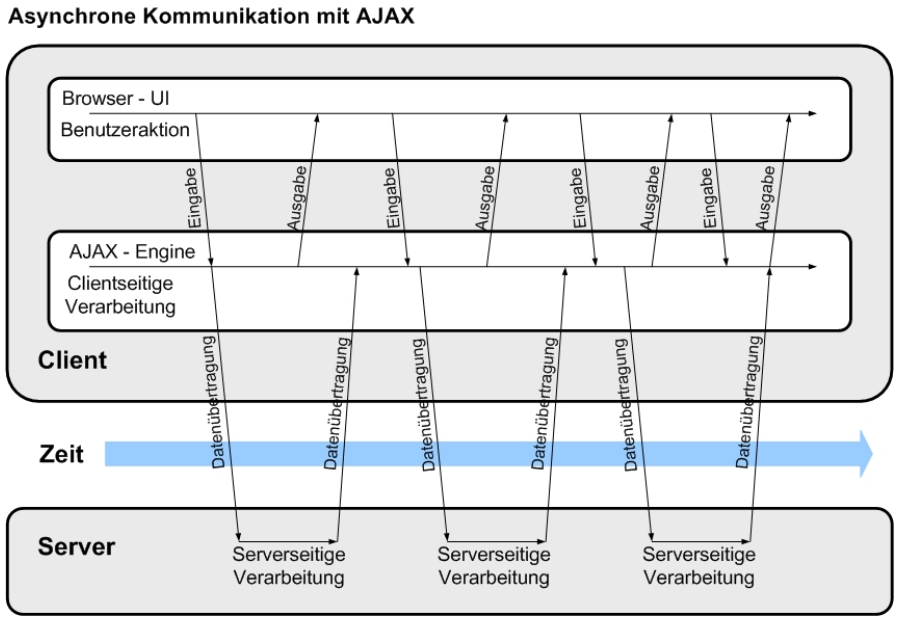

Abbildung 3: Asynchrone Kommunikation mit AJAX

Das kann jedoch reines HTTP und das synchrone Kommunikationsmodell nicht leisten. Mit AJAX ist das anders, denn die asynchrone Kommunikation ermöglicht es für jede Anfrage (Request) einen so genannten Callback-Handler zu definieren, der über den Zustand der Kommunikation benachrichtig wird. Das wird durch das so genannte XMLHttpRequest-Objekt ermöglicht, das ausführlich in Abschnitt 2.2.2.2.3 vorgestellt wird.

2.2.2 Wie arbeitet AJAX?

Wie bereits aus den Inhalten von Abschnitt 2.2.1 zu erkennen ist, eröffnet AJAX neue Möglichkeiten, wie man sie im Internet noch nie zuvor kannte. Doch um das wahre Potential von AJAX zu erkennen, muss das Prinzip der Webseite aufgegeben werden und ein Prozess des Umdenkens stattfinden. Aber um die Konsequenzen zu verstehen, sollen im folgenden Abschnitt die Technologien hinter dem Begriff AJAX vorgestellt und erklärt werden.

2.2.2.1 Historische Entwicklung

In diesem Abschnitt sollen die Technologien bzw. Techniken vorgestellt werden, die die Entstehung von AJAX ermöglichten.

World Wide Web: 1990 entwickelte Tim Berners Lee ein System auf der Basis von Hypertext zum weltweiten Austausch und zur Aktualisierung von Daten zwischen Wissenschaftlern. Im Rahmen dieses Projektes entstand der erste Browser „WorldWideWeb“ und Server „NeXTSTEP“. Diese Entwicklungen stellten den Ursprung des heutigen WWW dar.

JavaScript: Brendan Eich entwickelte 1995 den Vorgänger von JavaScript. Diese Scriptsprache nannte sich „LiveScript“ und wurde, im selben Jahr von Netscape[17] in einen Browser integriert, der Öffentlichkeit präsentiert. Wichtig an diesem Ansatz ist, dass erstmals eine Interaktion mit dem Benutzer möglich war. Ende 1995 wurde die Sprache von Netscape und SUN[18] aus Marketinggründen in JavaScript umbenannt.

Frames: Frames ermöglichen es eine Webseite in verschiedene Teilbereiche aufzuteilen. In diesen Teilbereichen werden HTML-Seiten angezeigt. Diese Technik wurde im Netscape 2.0 eingeführt und wird bis heute von den meisten Browsern unterstützt. Die Standardisierung wird vom World Wide Web Consortium (W3C) vorangetrieben (siehe Abschnitt 5.2.4.1). Doch die Implementierungen in den Browsern sind bis heute teilweise inkompatibel.

Hidden Frame Technik: Mit der Hidden Frame Technik wurde eine Möglichkeit geschaffen Daten in einem unsichtbaren Frame im Hintergrund zu laden und die Anwendung per JavaScript zu aktualisieren. Damit war das erste asynchrone Kommunikationsmodell im Internet entstanden.

---

[17] http://www.netscape.de/

[18] http://de.sun.com/homepage/index.html

Document Object Model (DOM): Das DOM ist eine Programmierschnittstelle für den Zugriff auf HTML- bzw. XML-Daten. Sie wird im Browser dazu eingesetzt den Inhalt einer Webseite in einer Baumstruktur abzubilden und dynamisch zu ändern. 1996 wurde sie im IE 4.0 zur Verfügung gestellt.

IFrames: 1997 wurde das „iframe“ Tag offiziell ein Teil von HTML 4.0. Es stellte eine weitere Verbesserung dar, da keine Frames mehr definiert werden müssen, sondern IFrames dynamisch erzeugt und on the Fly in den DOM eingefügt werden.

XMLHttp: Die letzte und zugleich wichtigste Komponente für AJAX, ein Objekt für die Client/Server Kommunikation, stand mit dem Release von IE 5.0 zur Verfügung. Es bietet dem Programmierer die Möglichkeit asynchrone HTTP Anfragen, per JavaScript an den Server abzusetzen und auf diese zu reagieren, unabhängig vom Page Load/Reload Zyklus. Mit zunehmender Popularität griffen andere Browserhersteller die Idee auf und implementierten ähnliche Objekte. Im Nachhinein stellten sich die verschiedenen Prinzipien der Implementierung als schwierig heraus. Das W3C nahm sich dieser Problematik an und definierte einen Standart für die Schnittstelle.

2.2.2.2 Vorstellung der Technologien

In Abschnitt 2.2.1 wurde über die Problematiken von heutigen Webanwendungen gesprochen und welche Möglichkeiten AJAX bietet. Doch die Technologien hinter dem Begriff AJAX wurden noch nicht definiert, dass soll nachfolgend geschehen.

2.2.2.2.1 JavaScript

JavaScript ist eine clientseitige, interpretierte und untypisierte Scriptsprache. Sie wird in einer in einer Sandbox[19] ausgeführt. Untypisiert bedeutet, dass definierte Variablen nicht wie in Java einen festen Datentyp zugeordnet sind, sondern in eine Variable, jeder Inhalt (z.B. String, Integer, Objekt) in beliebiger Reihenfolge geschrieben werden kann. Mit interpretiert ist gemeint, dass der Quellcode im Klartext an den Browser übertragen und dort ausgeführt wird. Dieser Ansatz stellt an den Browser eine Reihe von sicherheitsrelevanten Anforderungen, auf die in Abschnitt 6.1 näher eingegangen wird.

2.2.2.2.2 Extensible Markup Language

Extensible Markup Language (XML) ist ein Standart zur Strukturierung von Daten in einer Baumstruktur. Mit Hilfe von XML kann der Aufbau von Dokumenten definiert werden. Diese enthalten wiederum Daten, die einer fest vorgegeben Struktur entsprechen können.

[19] Eine Sandbox beschreibt in der Informatik eine Laufzeitumgebung, die die in ihr ausgeführte Software vom Rest des Systems abschirmt. Es können z.B. lediglich die Objekte des Browsers genutzt werden und der Zugriff auf das Dateisystem ist verboten.

Weitergehende Informationen können der Spezifikation ([W3CXML]) entnommen werden.

```
<message>
      <data>
            <name>Peter Mayer</name>
            <alter>30</alter>
       </data>
</message>
```

Abbildung 4: Beispiel für den Aufbau einer XML-Datei

2.2.2.2.3 XMLHttpRequest-Objekt

Das XMLHttpRequest-Objekt (siehe Anhang A2) stellt die notwendigen Funktionen bereit, um asynchrone Aufrufe mit JavaScript zu realisieren. Als Problem stellt sich die Inkompatibilität zwischen den verschiedenen Browser-Implementierungen dar. Die Erzeugung eines XMLHttpRequest-Objektes unterscheidet sich im Internet-Explorer (IE) gegenüber der von Mozilla basierten Browsern enorm (siehe *Abbildung 5*).

```
var xmlHttp = false;
//XMLHttpRequest-Object browserunabhängig erzeugen
try {
    xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
    try {
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e2) {
        xmlHttp = false;
    }
}
```

Abbildung 5: Browserunabhängige Erzeugung des XMLHttpRequest Objektes

Zuerst wird eine Instanz des XMLHttpRequest-Objektes erzeugt und anschließend mit der Methode „open“ die Verbindung zum Server aufgebaut (siehe *Abbildung 6*).

```
xmlHttp.open("GET", "http://www.example.com/test_servlet",  true);
xmlHttp.onreadystatechange = updatePage;
xmlHttp.send("Bespieldaten");
function updatePage() {
    alert("Daten vom Server sind da.");
}
```

Abbildung 6: Asynchrone Kommunikation mit dem XMLHttpRequest-Objekt

Durch zusätzliche Parameter kann zwischen asynchroner bzw. synchroner Kommunikation und verschiedenen Anfragemethoden (z.B. GET, POST) gewählt werden.

Anschließend muss der Name einer Callback-Funktion angegeben werden, um die eingehenden Daten vom Server zu verarbeiten und das Benutzerinterface zu aktualisieren. Mit der Methode „send“ werden die Daten an den Server übertragen.

2.2.2.2.4 Cascading Style Sheets

Mit Cascading Style Sheets (CSS) wird die Formatierung der Webseite von den Daten getrennt. Ein großer Vorteil ist, dass die Formatierung in eine separate Datei (*.css) ausgelagert wird und somit leicht auszutauschen ist. Auf eine nähere Betrachtung von CSS wird an dieser Stelle verzichtet und auf die Spezifikation ([W3CCSS]) verwiesen.

2.2.2.2.5 Dokument Object Model

Das Dokument Object Model (DOM) stellt ein vom W3C standardisiertes Objektmodell ([W3CDOM]) zur Verfügung, um XML Daten auszuwerten oder zu manipulieren. Ein großer Vorteil ist, dass die Elemente in einer hierarchischen Struktur abgebildet werden, dadurch stehen die einzelnen Elemente zueinander in einer Beziehung (Eltern, Geschwister, Kinder). Die Struktur einer Webseite ist ebenfalls als Baum hinterlegt, wie das Beispiel in *Abbildung 7* zeigt.

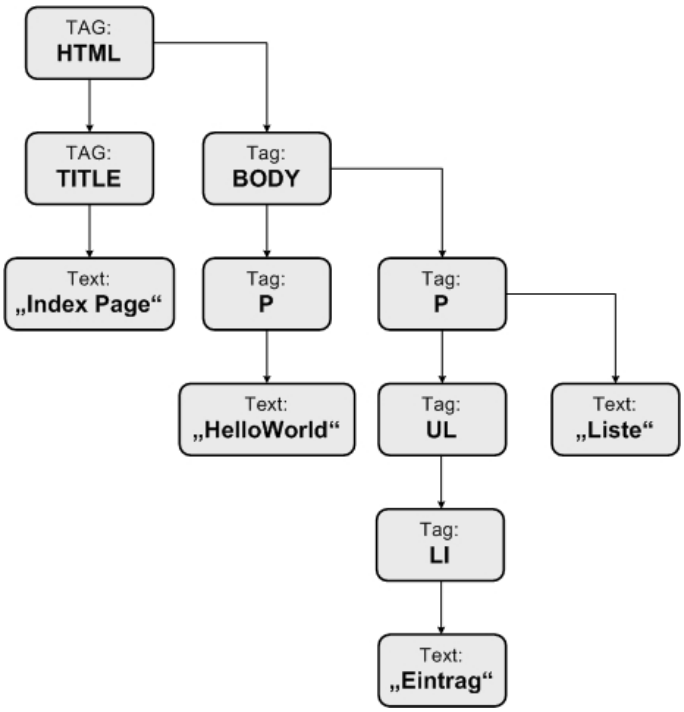

Abbildung 7: Aufbau eines HTML-DOM

Die einzelnen Elemente können, angefangen beim Root-Element, rekursiv auf- und abwärts durchlaufen werden. Dieser Umstand gibt dem Entwickler die Möglichkeit den DOM nach belieben zu manipulieren und somit die Benutzerschnittstelle anzupassen.

In JavaScript steht die globale Variable „document“ zur Verfügung, um dies zu realisieren.

2.2.2.3 Funktionsweise von AJAX

Die von AJAX eingesetzten Technologien wurden in Abschnitt 2.2.2.2 vorgestellt, doch es wurde noch nicht erklärt wie sie zusammenarbeiten. Das wird in diesem Abschnitt, mit dem in *Abbildung 8* dargestellten Szenario, nachgeholt.

Die Benutzerschnittstelle wird wie in normalen Webanwendungen mit HTML und CSS realisiert. Wenn der Benutzer eine Suchanfrage absetzt, wird eine asynchrone Nachricht an den entsprechenden Dienst gesendet. In diesem Fall besteht dieser Dienst aus einem einfachen Servlet. Das Verschicken der asynchronen Nachricht wird durch das XMLHttpRequest-Objekt ermöglicht. Die Logik wird jedoch in JavaScript implementiert. Nachdem der Dienst die Anfrage beantwortet hat, werden die eingehenden Daten von der Callback-Funktion ausgewertet und der HTML-DOM wird aktualisiert. Nun kann der Browser die Benutzerschnittstelle aktualisieren und dem Benutzer das Ergebnis präsentieren.

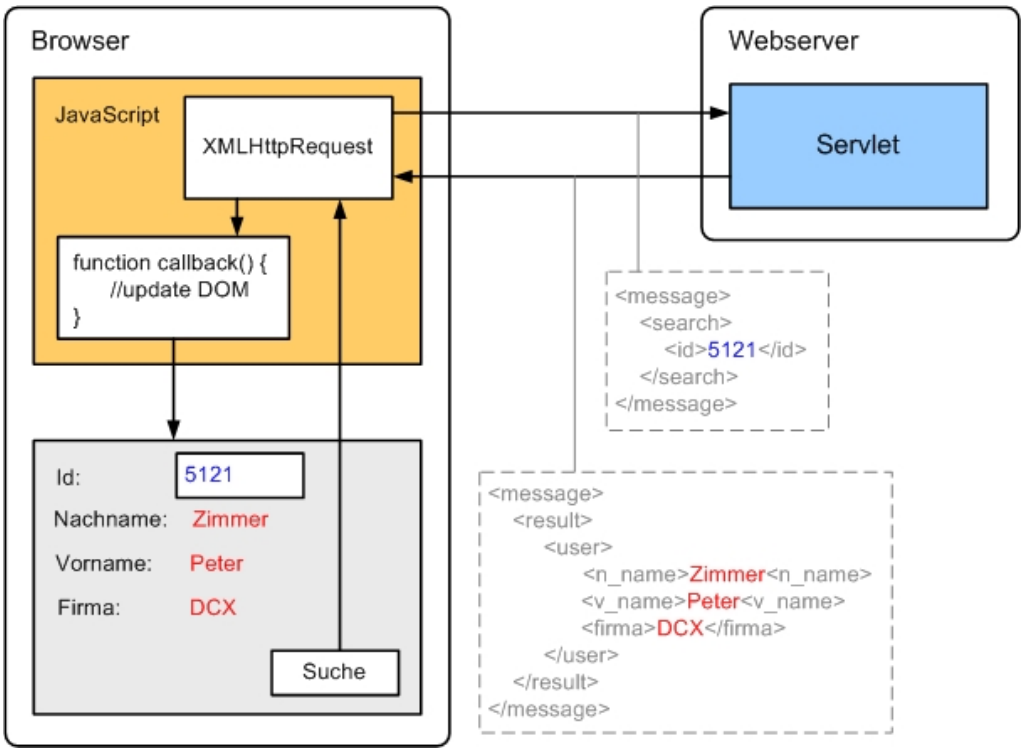

Abbildung 8: Funktionsweise von AJAX

Der Benutzer hat durch asynchrone Kommunikation den Vorteil, dass die Benutzerschnittstelle nicht blockiert und dadurch sein Arbeitsfluss nicht unterbrochen wird. Die Daten werden zwischen Client und Server häufig im XML-Format übertragen, das ist jedoch nicht zwingend notwendig (siehe Abschnitt 5.1.3).

All die verwendeten Technologien sind in den gängigsten Browsern verfügbar. Der Browser ist auf jedem modernen Betriebssystem vorinstalliert und somit sind eine riesige Menge von PCs, Laptops schon jetzt bereit AJAX-Anwendungen auszuführen. Das ist natürlich ein sehr großer Vorteil, wenn man die sich ergebenden Möglichkeiten (z.B. Ausbreitung: siehe Abschnitt 4.1.2.5) betrachtet.

2.2.3 AJAX Pioniere

An dieser Stelle sei Google als ein Unternehmen genannt, welches dazu beigetragen hat, AJAX-Anwendungen bekannt zu machen. 2004 wurde GMail in Deutschland veröffentlicht und erregte vor allem durch seine vielfältige Benutzerschnittstelle aufsehen.

Später lies Google weitere Systeme folgen. Dies waren z.B. Google Suggest[20] und Google Maps. Google Maps demonstriert auf beeindruckende Weise die Vorteile von AJAX-Anwendungen.

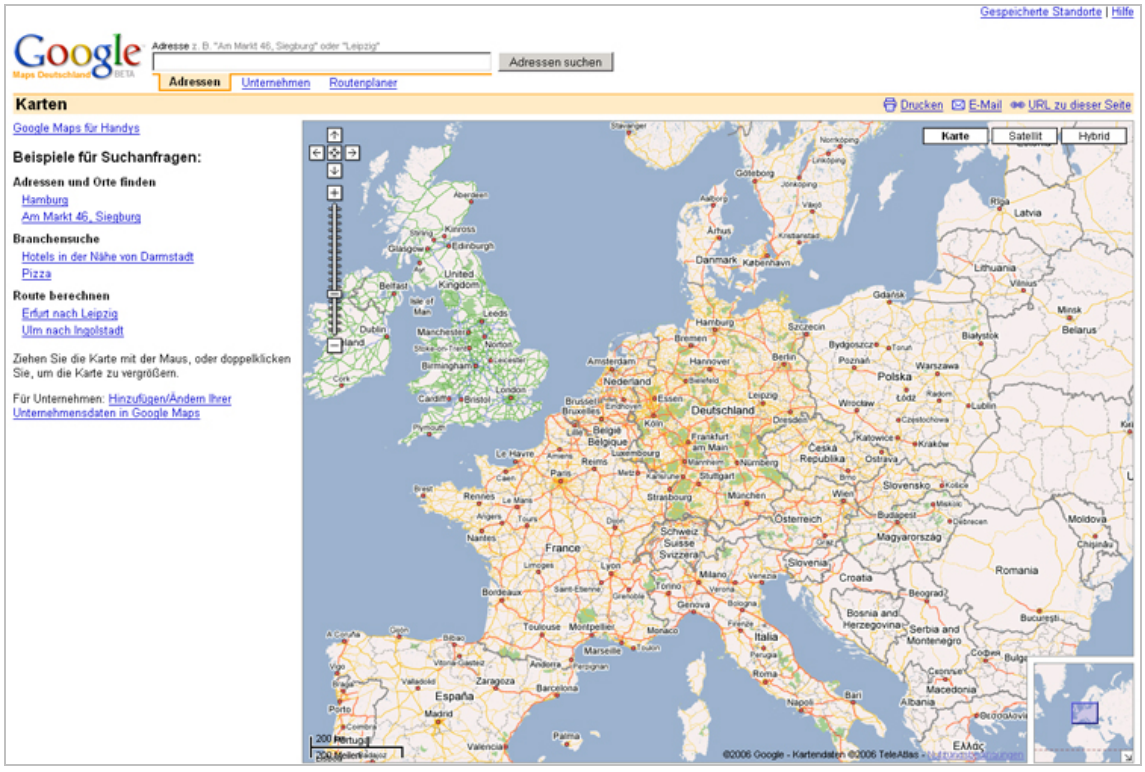

Abbildung 9: Screenshot von Google Maps

Durch Texteingaben kann der Benutzer Abfragen vornehmen und bis zu einzelnen Strassen, Hotels oder Restaurants zoomen. Bei einer Suche nach z.B. Restaurants werden zusätzliche Informationen angezeigt, indem der Anwender mit dem Mauszeiger

[20] http://www.google.com/webhp?complete=1&hl=en

über einen Symbol verharrt. Das wohl auffälligste Merkmal ist, dass der Benutzer die Karte verschieben kann.

Außerdem wird ein Service angeboten, die Funktionen von Google Maps in einer anderen Anwendung zu nutzen.

2.3 Die Proactive Infrastukture

Nachdem der vorherige Abschnitt in das Thema AJAX einführte, soll nun die Proactive Infrastrukture (PAI) vorgestellt werden. Dabei werden die folgenden Punkte angesprochen:

- Probleme der Projekte vor PAI
- Nutzen den die Projekte durch PAI haben
- Vorstellung der PAI Plattformen

2.3.1 Hintergrund

Die IT-Landschaft von DCX ist sehr heterogen (siehe *Abbildung 10*: links). Beim Entwickeln von Anwendung haben die Projekte viele Anforderungen an die Infrastruktur und verwendete Middleware. Einige dieser Anforderungen bzw. Probleme werden nun angesprochen:

- Die Projekte benötigen eine leistungsstarke, getestete und verlässliche Infrastruktur, welche die gemeinsamen Technologie Standards erfüllt und in einem gemeinsamen Rechenzentrum lauffähig ist.
- Die Projekte sollten sich auf die Entwicklung der Anwendung konzentrieren und nicht mit der Infrastruktur beschäftigen.
- Die Projekte sollten eine Anwendungsarchitektur vorfinden, in der die verschiedenen Produkte, Sicherheitsrichtlinien, usw. bereits berücksichtigt sind.

Die hier aufgeführten Anforderungen müssten von den Projekten erfüllt werden und stellen somit eine enorme finanzielle Belastung und ein hohes Risiko dar.

2.3.2 Die Ziele von PAI

PAI stellt eine standardisierte und zertifizierte Plattform zur Entwicklung von Unternehmenssoftware und Services bereit. Die Arbeit der Anwendungsentwickler soll durch PAI vereinfacht werden, da durch die Bereitstellung von verschiedenen integrierten Plattformen, die Komplexität der Infrastruktur und Middleware in den Hintergrund tritt (siehe *Abbildung 10*: rechts). Dadurch wird die Variantenvielfalt der verwendeten Middleware in den Rechenzentren minimiert.

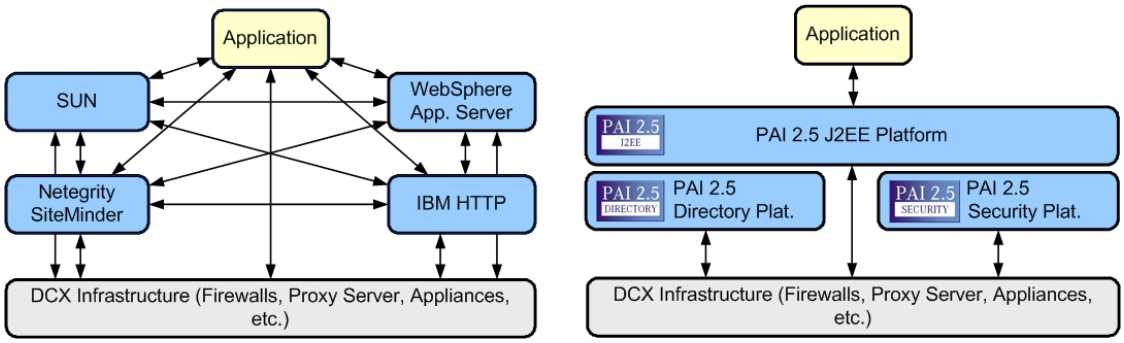

Abbildung 10: links: Infrastruktur vor PAI, rechts: Infrastruktur mit PAI

Nun sollen einige Vorteile vorgestellt werden, die Projekte durch den Einsatz von PAI haben:

- PAI vereinfacht die Planung und Entwicklung von Anwendungen durch die Bereitstellung standardisierter Komponenten in Enterprise Architekturen.
- PAI vereinfacht die Entwicklung von Anwendungen, indem die Komplexität der Infrastruktur vor dem Entwickler versteckt wird.
- PAI reduziert den Testaufwand für Anwendungen, durch die Bereitstellung von getesteten und integrierten Produkten.
- PAI erleichtert die Integration von Anwendungen, indem eine standardisierte Infrastruktur und Middleware zur Verfügung gestellt wird.
- PAI verbessert die Sicherheit von Anwendungen, indem ein standardisiertes Sicherheits-Framework bereitgestellt wird, das einige der DCX Richtlinien implementiert.
- PAI vereinfacht die Installation und Entwicklung von Anwendungen
    - Die Projekte liefern lediglich die Anwendung an das Rechenzentrum aus und nicht den gesamten Middleware-Stack
    - Dadurch haben die Rechenzentren die Möglichkeit den Middleware-Stack vorzuinstallieren

Durch die Bereitstellung einer einheitlichen Infrastruktur nimmt PAI den Projekten viele Probleme ab, mit denen sie sich sonst beschäftigen müssten. All diese Vorteile ermöglichen es den Projekten die Kosten zu optimieren, das Risiko zu minimieren und qualitativ hochwertigere Software bereitzustellen. Es sollte in diesem Zusammenhang ergänzend erwähnt werden, dass PAI ausschließlich DCX intern genutzt wird bzw. verfügbar ist.

2.3.3 PAI Plattformen Konzept

In diesem Abschnitt sollen die von PAI zur Verfügung gestellten Plattformen vorgestellt werden. In *Abbildung 11* sind die Plattformen durch das PAI Logo markiert, die ein Bestandteil von PAI sind.

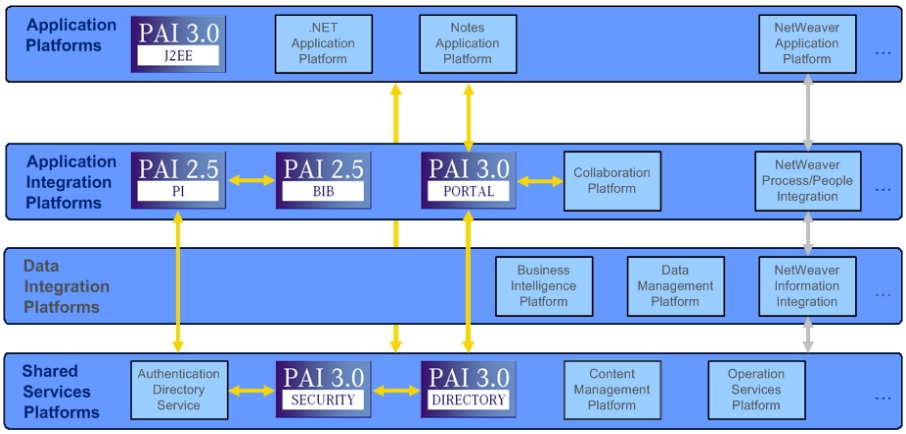

Abbildung 11: Die PAI Plattformen und ihre Abhängigkeiten voneinander.

Diese Plattformen sollen nachfolgend vorgestellt werden.

2.3.3.1 J2EE Plattform

Die J2EE Plattform ermöglicht die Erstellung von Java basierten, J2EE konformen und voll in die DCX Infrastruktur integrierten Anwendungen. Sie besteht aus den Basisprodukten IBM HTTP Server und IBM Websphere Application-Server (WAS) und stellt dem Entwickler diverse nützliche Mechanismen (z.B. Logging) zur Verfügung.

2.3.3.2 PI Plattform

Die PI Plattform stellt eine Laufzeitumgebung für Prozess gesteuerte Anwendungen bereit. Sie erleichtert die Implementierung, Ausführung und Überwachung von Business Prozessen und ist vollständig in die J2EE Plattform integriert. Sie ist vollständig in die Directory und Security Plattform integriert und erlaubt somit die zentrale Administration der Daten. Es werden außerdem Möglichkeiten zur leichteren Integration der BIB Plattform bereitgestellt.

2.3.3.3 BIB Plattform

Die Business Information Broker (BIB) Plattform ermöglicht die Integration der verschiedensten Business Anwendungen, indem ein Enterprise Bus zur Verfügung gestellt wird.

2.3.3.4 Portal Plattform

Die Portal Plattform stellt eine Portlet basierte Umgebung bereit, um die Erstellung von Webseiten zu vereinfachen, die auf die verschiedensten Quellen verweisen. Sie basiert auf dem IBM HTTP Server, WAS und IBM Portal Server und ist komplett in die PAI Security Infrastruktur integriert. Dadurch wird die Administration auf Basis des PAI Security Modells ermöglicht.

2.3.3.5 Directory Plattform

Die Directory Plattform (DIR) bildet die Basis der Sicherheitsinfrastruktur von PAI. Sie stellt das Datenmodell sowie den Datenspeicher für alle Benutzer und sicherheitsrelevanten Daten zur Verfügung. Diese Plattform basiert auf dem SUN System Directory.

2.3.3.6 Security Plattform

Die Security Plattform (SEC) baut auf der Direktory Plattform auf und ermöglicht die Realisierung von End-to-End Security. Sie basiert auf dem Basisprodukt Netegrity SiteMinder. Dieses Produkt stellt Funktionen zur Authentifizierung, Uniform-Resource-Locator (URL) basierten Autorisierung und Single-Sign-On (SSO) bereit. Die Plattform ist so konzipiert, das ohne großen Aufwand andere Basisprodukte integriert werden können (z.B. Active Directory).

3 Schichtenmodelle und Clientparadigmen

Das vorherige Kapitel stellt AJAX und die PAI Plattform vor. Es werden die Technologien erörtert, die sich hinter dem Begriff AJAX verbergen und die Ziele von PAI beschrieben.

Doch einige Fragen blieben unbeantwortet, wie: Welche Möglichkeiten zur Entwicklung von Anwendungen bietet PAI und welche Vorteile bzw. Probleme ergeben sich daraus? Das wird in diesem Kapitel nachgeholt.

Es werden zuerst gängige Schichtenmodelle betrachtet und anschließend die Clientparadigmen vorgestellt, mit denen Anwendungen im PAI Umfeld entwickelt werden können. Diese Clientparadigmen werden hinsichtlich ihrer Architektur und der daraus resultierenden Vorteile bzw. Probleme vorgestellt.

Abschließend werden die so genannten Rich Internet Applications (RIA) beleuchtet, die z.B. mit AJAX realisiert werden können.

3.1 Schichtenmodelle

Das klassische Interaktionsmodell für verteilte Anwendungen ist das Client/Server Modell. Der Client dient als Dienstnachfrager und der Server als Dienstanbieter. Die Kommunikation zwischen Client und Server erfolgt in der Regel über ein Netzwerk (Intranet, Internet). Dabei hat der Client die Möglichkeit auf verschiedene Server zuzugreifen und der Server verschiedene Clients zu bedienen. Der Server kann seinerseits, falls er einen Dienst benötigt, einem anderen Server gegenüber als Client auftreten.

Für das Client/Server Modell wurden diverse abstrakte Schichten (im englischen auch Tiers genannt) definiert.

- *Präsentation:* Die Präsentation stellt eine Schnittstelle für die Benutzerinteraktion zur Verfügung und wertet die empfangenen Daten aus bzw. aktualisiert die Anzeige.
- *Anwendungslogik:* Die Anwendungslogik beinhaltet die Verarbeitungsmechanismen der Anwendung. Die benötigten Daten werden von der Datenhaltung bezogen.
- *Datenhaltung:* Sie stellt die Daten zur Verfügung, die von der Anwendungslogik benötigt werden und ist für die Speicherung der dort produzierten Daten zuständig.

Diese Schichten lassen sich auf verschiedene Weise auf den Client und Server verteilen. Client/Server Architekturen lassen sich in 2-Tier, 3-Tier und n-Tier Modelle aufteilen, die wiederum abhängig von der Verteilung auf die Hardware-Komponenten

(Client, Webserver, Datenbankserver, usw.) sind. Die der PAI Plattform zugrunde liegende Basisarchitektur ist die 3-Tier-Architektur, die im Anschluss mit den häufigsten Schichtenmodellen besprochen wird.

3.1.1 2-Tier-Modell

Beim 2-Tier-Modell ist die gesamte Anwendung auf zwei Schichten verteilt: Auf Schicht eins befindet sich die Präsentation und die Anwendungslogik. Die Anwendungslogik kommuniziert über das Netzwerk mit der Datenhaltung, um die unternehmensspezifischen Daten zu lesen bzw. zu schreiben.

Der größte Nachteil dieses Schichtenmodells besteht darin, dass die Präsentation nicht klar von der Anwendungslogik getrennt ist und nicht in einer anderen Anwendung wieder verwendet werden kann. Weiterhin kann die Datenbankintegrität gestört werden, da jeder Client über seine eigene Anwendungslogik verfügt und somit ein Fehler andere Clients gefährden kann.

Die Skalierung von Anwendungen ist limitiert, da jeder Client eine oder mehrere Datenbankverbindungen verwaltet und die Anzahl der Verbindungen an die Server- bzw. Netzwerk-Hardware gebunden ist.

3.1.2 3-Tier-Modell

Die offensichtlichen Nachteile des 2-Tier-Modells werden durch das 3-Tier-Modell behoben. Das wird erreicht, indem die Präsentation von der Anwendungslogik getrennt wird. Die sich ergebende Verteilung der Schichten und daraus resultierenden Vorteile und Probleme werden in diesem Abschnitt besprochen.

3.1.2.1 Schichtenverteilung

Bei der 3-Tier-Architektur wird die gesamte Anwendung (Präsentationslogik, Anwendungslogik und Datenhaltung) auf drei logische Schichten (siehe *Abbildung 12*) verteilt:

- Die Präsentations-Tier ist für die Präsentation zuständig und kann eventuell die Präsentationslogik enthalten. Als Beispiel sind z.B. der Web-Browser oder ein GUI-Client zu nennen.
- Der Anwendungs-Tier enthält die gesamte Anwendungslogik und besteht aus einem Webserver oder Application-Server. Dadurch ist die Anwendungslogik zentral verfügbar und kann von verschiedenen Clients verwendet werden.
- Die Datenhaltungs-Tier ist für die Datenhaltung zuständig und kann z.B. aus einem Datenbankserver oder einem ERP-System[21] (z.B. SAP R3) bestehen.

[21] ERP-Systeme bestehen aus komplexer Anwendungssoftware zur Unterstützung der Ressourcenplanung einer ganzen Unternehmung (http://de.wikipedia.org/wiki/ERP-System).

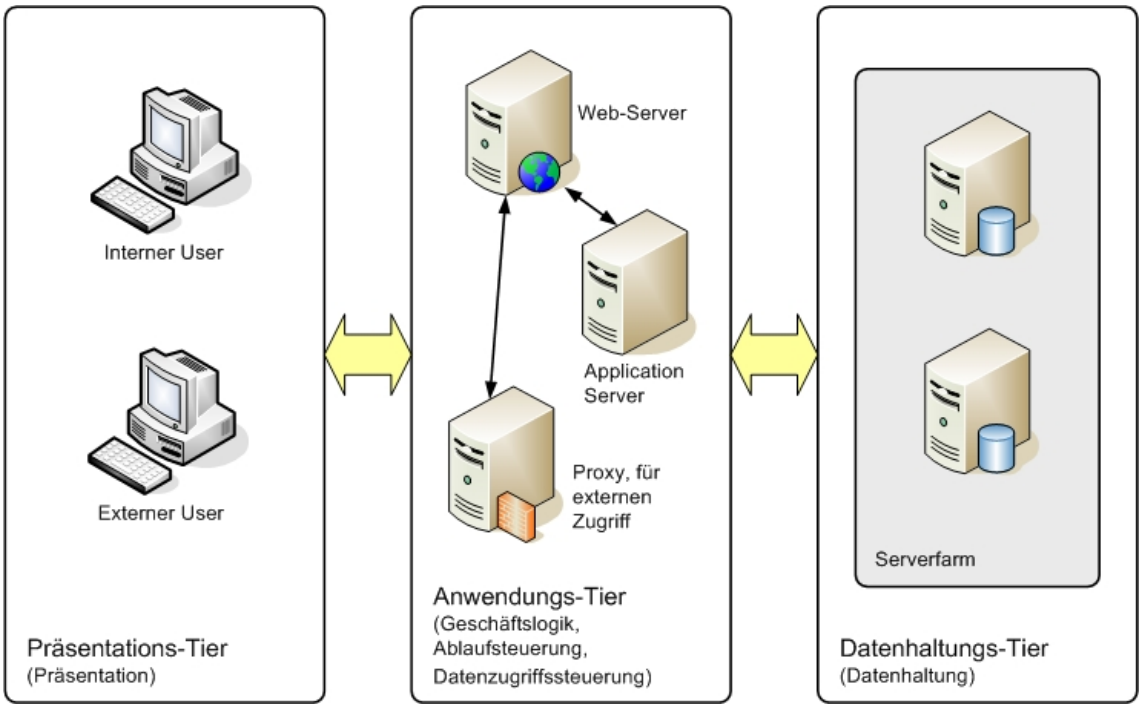

Abbildung 12: Schichtenverteilung des 3-Tier-Modells

3.1.2.2 Vorteile und Probleme

Durch den Einsatz des 3-Tier-Modells ergeben sich diverse Vorteile und Probleme, die nachfolgend aufgeschlüsselt und betrachtet werden.

Vorteile vom 3-Tier-Modell:

- Verbesserung der Skalierbarkeit jeder einzelnen Schicht
- klarere Architektur durch eine saubere Trennung der Schichten

Probleme vom 3-Tier-Modell:

- zahlreiche technische Schnittstellen
- aufwändiger Abstimmungsprozess bei der Entwicklung
- Portabilität von Anwendungen

3.1.2.2.1 Skalierbarkeit

Die verschiedenen Schichten lassen sich an die spezifischen Hardware-Anforderung anpassen. Das hat z.B. den Vorteil, wenn an einer Stelle des Systems die Datendurchsatzrate gesteigert werden muss, wird die Architektur des Systems und aller anderen Schichten nicht in Mitleidenschaft gezogen.

3.1.2.2.2 Klarere Architektur

Eine saubere Trennung der einzelnen Schichten untereinander ermöglicht es, die Architektur auf einfache Art und Weise zu erweitern. Außerdem wird die Aufgabenverteilung im Projekt unterstützt, da die Komponenten getrennt von einander entwickelt

werden können. Dadurch wird eine höhere Wiederverwendbarkeit und vor allem bessere Qualität der Komponenten bzw. Anwendungen erreicht.

Damit die Komponenten sich unterhalten können, müssen Schnittstellen spezifiziert werden. Der Kommunikationsbedarf kann jedoch an dieser Stelle ansteigen, dass dient aber der Entwicklung einer sauber definierten und gut dokumentierten Schnittstelle.

3.1.2.2.3 Zahlreiche technische Schnittstellen

An jeder Tier-Grenze entstehen zunächst einmal technische Schnittstellen, die die Komplexität des Gesamtsystems scheinbar erhöhen. Doch durch den Einsatz von z.B. Patterns können die geschaffenen technischen Rahmenbedingungen bestmöglich vor den Benutzern versteckt werden.

3.1.2.2.4 Portabilität von Anwendungen

Die Portabilität von Anwendungen ist enorm eingeschränkt, da die verschiedenen Hersteller komplette Produkte anbieten. Trotz der Standardisierung in diesem Bereich (J2EE Standart: [J2EE]) ist der Wechsel zwischen verschiedenen Produkten sehr schwierig. Die Anwendung ist sozusagen für ein Produkt maßgeschneidert.

3.1.3 n-Tier-Modell

Die verschiedenen abstrakten Schichten können theoretisch noch feiner unterteilt werden. So kann z.B. die Anwendungs-Tier in weitere logische Schichten aufgeteilt werden. Daraus ergeben sich so genannte n-Tier-Modelle, die in diesem Zusammenhang nicht näher betrachtet werden sollen.

3.2 Clientparadigmen

Die Entwicklung der Clientparadigmen ist in *Abbildung 13* dargestellt. Es ist zu erkennen, dass eine drastische Verschiebung von Interaktivität der Benutzeroberfläche hin zur Reichweite stattgefunden hat.

So entstanden zuerst Anwendungen, die auf Großrechnern (Mainframe) ausgeführt wurden und eine textbasierte Benutzeroberfläche boten. Mit der Entwicklung der Client/Server Architektur kamen grafische Benutzeroberflächen (Rich Client) auf den Desktops auf, die rechhaltigere Möglichkeiten boten.

Bedingt durch die Entwicklung des Internet konnte der Benutzer eine Verbindung zu jedem Server auf der Erde aufbauen. Aber das Internet brachte einen Rückschritt bezüglich der Interaktivität von Benutzeroberflächen mit sich. Mit den Rich Internet Applications (RIA) besteht jedoch die Möglichkeit die Interaktivität von Desktop-Anwendungen zurückzugewinnen und die Reichweite des Internets beizubehalten.

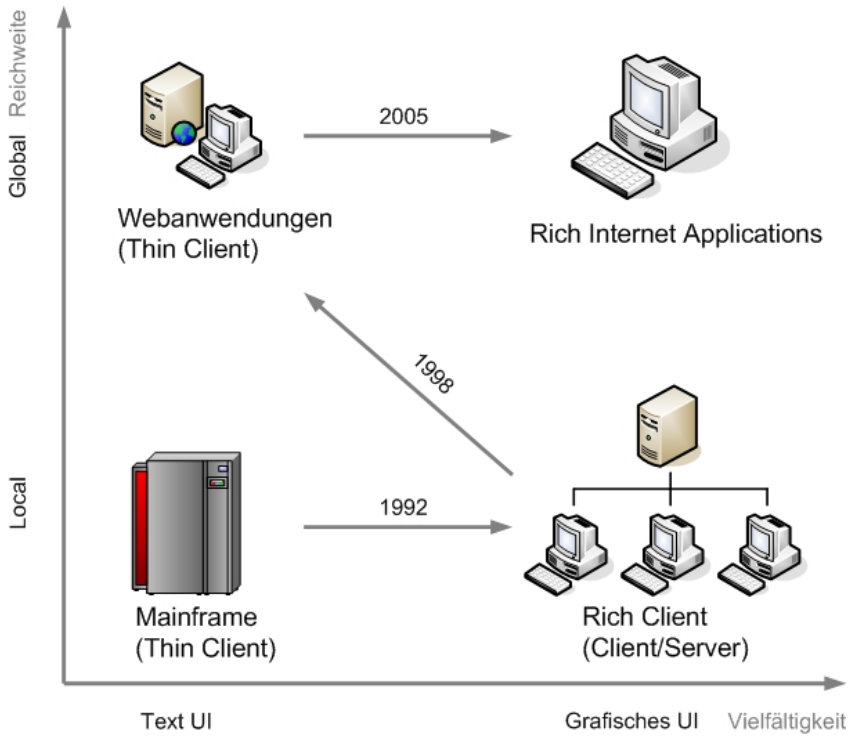

Abbildung 13: Historische Entwicklung der Clientparadigmen

3.2.1 Thin Client

Beim Thin Client Paradigma werden die Daten möglichst vollständig vom Server bezogen. Das kann im extremsten Fall das gesamte Betriebssystem beinhalten.

3.2.1.1 Architektur

Thin Client Anwendungen basieren auf dem Server based Computing (SBC) Konzept, bei dem die Anwendung zu 100 Prozent auf dem Server installiert und dort ausgeführt wird. Die Verteilung der abstrakten Schichten ist wie folgt:

- Präsentation: Der typische Thin Client übernimmt die Präsentation und nimmt die eingehenden Daten entgegen. Diese werden ungefiltert und unbearbeitet an den Server weitergeleitet. Die Präsentationslogik befindet sich auf dem Server und erzeugt eine generische Beschreibung der Benutzerschnittstelle, die dann vom Client ausgewertet und angezeigt wird.
- Anwendungslogik: Die Anwendungslogik ist in diesem Szenario komplett auf dem Server hinterlegt. Sie empfängt die eingehenden Daten, verarbeitet diese und sendet die in Zusammenarbeit mit der Präsentationslogik aktualisierte Beschreibung der Benutzerschnittstelle an den Client zurück.
- Datenhaltung: Der Zustand des Clients und die von der Anwendung benötigten Daten werden lediglich auf dem Server gespeichert.

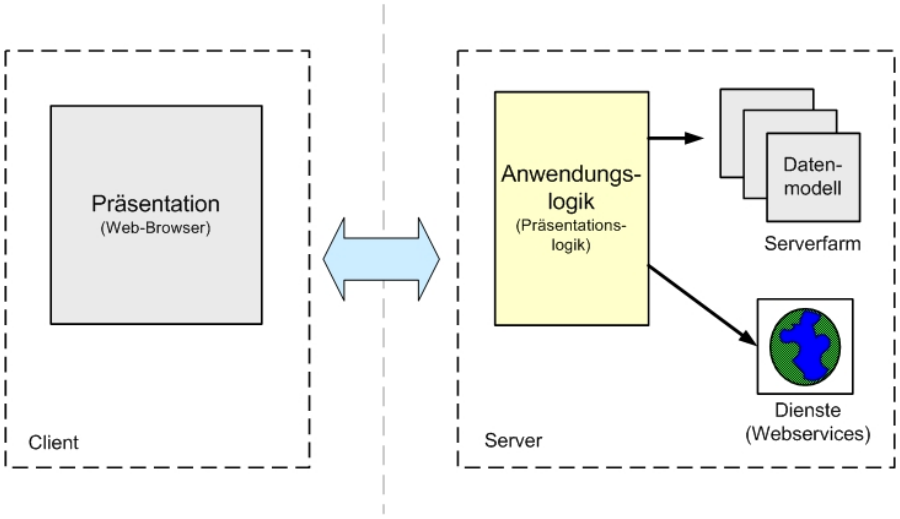

Abbildung 14: Prinzip der Thin Client Architektur.

Der Thin Client muss nicht immer von einer GUI-Komponente, sondern kann durch jede andere schlanke Input/Output-Komponente realisiert werden, die mit dem Endgerät Daten austauscht und diese unbearbeitet an den Server weiterleitet. Das kann z.B. ein Druckertreiber, Scanner oder Netzwerkadapter sein.

Das bekannteste Beispiel sind Webanwendungen, die typischerweise im Browser ausgeführt werden, der lediglich HTML-Daten präsentiert. Die eingehenden Daten werden an den Server gesendet, der Browser wartet auf die Antwort und aktualisiert die Benutzerschnittstelle.

3.2.1.2 Lebenszyklus

Thin Client Anwendungen sind ausschließlich seitenorientiert, der Client (z.B. Browser) weiß nichts über den Arbeitsfluss des Benutzers, denn er ist lediglich ein Terminal zur Anzeige der Benutzerschnittstelle. Um das Speichern von sitzungsspezifischen Daten zu ermöglichen, wurde die Session erfunden. Damit ist es möglich diese Art von Daten auf dem Server zu speichern.

Jede Interaktion des Benutzers hat zur Folge, dass eine neue Seite zum Client geschickt und angezeigt wird, die zuvor angezeigte Seite wird daraufhin verworfen. Die Benutzerschnittstelle blockiert während dieses Zeitraumes. Dieser Effekt tritt auf, weil das synchrone Kommunikationsmodell (siehe Abschnitt 2.2.1.3) eingesetzte wird.

Wenn der Benutzer die Sitzung beendet oder den Client schließt, wird die Anwendung beendet und die Sitzung zerstört. Alle wichtigen Daten, die beim nächsten Start der Anwendung vorhanden sein sollen, können serverseitig gespeichert werden.

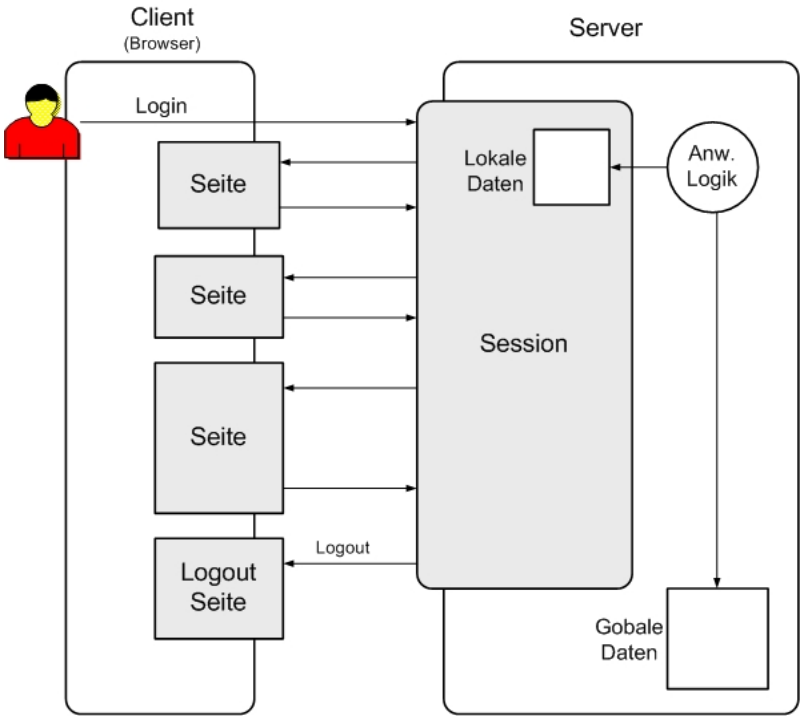

Abbildung 15: Lebenszyklus von Thin Client Anwendungen

3.2.1.3 Vorteile und Probleme

Durch den Einsatz von Thin Clients ergeben sich diverse Vorteile und Probleme, die nachfolgend aufgeschlüsselt und betrachtet werden.

Vorteile des Thin Clients:

- hohe Stabilität und niedrige Wartungsaufwände
- einfache Ausbreitung und Softwareänderungen sind sofort beim Client sichtbar
- geeignet für Clients mit wenig RAM-Speicher, langsamen Prozessor und vorhandenem Netzwerkanschluss
- Ausfall des Clients bleibt meist ohne Folgen und die Arbeit kann problemlos an einem anderen Arbeitsplatz fortgesetzt werden

Probleme durch Thin Clients:

- eingeschränkte Benutzerschnittstelle
- Antwortzeit ist stark abhängig vom Netzwerkverkehr und der Serverperformance
- Höhere Belastung des Servers durch zusätzliche Benutzerschnittstellenaufbereitung des Clients
- hohe Last auf den Netzwerk bei jeder Aktualisierung der Benutzerschnittstellen
- Ausfall des Servers hat den Ausfall des Clients zur Folge
- meist nur synchrone Kommunikation möglich

3.2.1.3.1 Benutzerschnittstelle

Mit Thin Client Architekturen lassen sich keine reichhaltigen und interaktiven Benutzerschnittstellen erzeugen.

Das größte Problem ist die Seitenorientierung (siehe Abschnitt 3.2.1.2), da die zur Verfügung gestellten Benutzerelemente keine Möglichkeit haben mit der Präsentationslogik auf dem Server zu interagieren und ihren Zustand separat zu aktualisieren.

Die generische Beschreibungssprache der Benutzerschnittstelle muss so ausgelegt sein, dass sie auf möglichst vielen Clients bzw. Endgeräten (z.B. PC, Mobile) funktionsfähig ist und deshalb werden nur die notwendigsten grafische Benutzerelemente (z.B. Textfeld, Listen, usw.) bereitgestellt.

3.2.1.3.2 Ausbreitung

Thin Client Anwendung werden ausschließlich auf dem Server installiert. Dieser Sachverhalt ermöglicht es, die Anwendungen zentral zu Betreiben und einer Vielzahl von Anwendern global zur Verfügung zu stellen. Der Konfigurations-, Administrations-, Wartungs- und Installationsaufwand der Anwendung verringert sich enorm.

So muss bei einer Aktualisierung der Anwendung lediglich der auf dem Server liegende Code angepasst werden und die Softwareänderungen sind sofort für alle Clients verfügbar.

3.2.1.3.3 Performance

Die Performance der Thin Client Anwendung ist sehr stark von der Leitungsfähigkeit des Servers und der Auslastung des Netzwerkes abhängig.

Ein Problem ist die Minimierung des Datenvolumens zwischen Client und Server. Die Webseiten werden immer als Ganzes übertragen. Es gibt keine Möglichkeit Teilbereiche, die sich geändert haben, an den Client zu senden. Deshalb werden bei der Aktualisierung der Benutzerschnittstelle oft redundante Daten mit übertragen. Durch clientseitiges zwischenspeichern der Daten (Caching) kann dieses Problem begrenzt werden.

Außerdem kann der Server nur eine begrenzte Anzahl von Anfragen bearbeiten. Aus diesem Grund werden verschiedene Server zur Verfügung gestellt, um die Anfragen pro Server an die Gegebenheiten der Hardware anzupassen.

3.2.1.3.4 Abhängigkeit vom Server

Da sich die Anwendung lediglich auf dem Server befindet, hat ein Ausfall gleichzeitig den Ausfall aller Clients zur Folge. Aus diesem Grund können Thin Client Anwendungen keinen „offline“ Modus bereitstellen.

Es müssen spezielle Maßnahmen getroffen werden, um die Ausfallsicherheit zu gewährleisten. Deshalb werden häufig mehrere Maschinen mit der gleichen Aus-

stattung (Cluster) aufgestellt und ein Load Balancer[22] vorgeschalten, um die Anfragen weiterzuleiten.

3.2.2 Rich Client

Das Rich Client Paradigma ist ein Ableger des Fat Client Paradigma und wird typischerweise durch ein atomares Programm repräsentiert. Das Programm kann aus mehreren Modulen bestehen und ist auf dem Zielsystem (z.B. Rechner) installiert.

Der Unterschied zwischen den beiden Lösungen besteht darin, dass beim Rich Client Paradigma ein Framework zur Verfügung steht, das reichhaltige Lösungsmöglichkeiten (z.B. Plugin-Konzept, Update-Mechanismus, usw.) bereitstellt, um die Anwendungsentwicklung zu erleichtern. Es existieren verschiedene Frameworks, wie z.B. Rich Client Platform (RCP: [RCP]), Netbeans ([NetBeans]) oder Spring ([SPRING]), die nicht näher vorgestellt werden sollen.

3.2.2.1 Architektur

Die grundlegende Architektur und die daraus resultierende Verteilung der abstrakten Schichten von Rich Clients sind in *Abbildung 16* dargestellt.

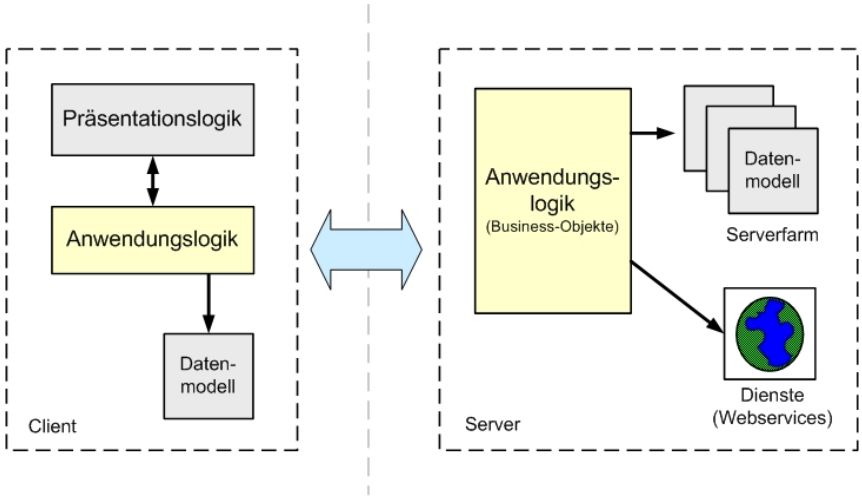

Abbildung 16: Prinzip der Rich Client Architektur

Die grundlegende Funktion der Schichten wurden bereits in Abschnitt 3.1 angesprochen, aus diesem Grund werden die für das Rich Client Paradigma wichtigen Änderungen angesprochen:

- *Präsentation:* Die Präsentation übernimmt der Client. Im Gegensatz zum Thin Client Paradigmen ist die Präsentationslogik vollständig auf dem Client vorhanden.

[22] Beschreibt die Lastverteilung des Netzwerktraffic auf verschiedene Rechner im Netzwerk.

- *Anwendungslogik:* Die Anwendungslogik kann zwischen Client und Server aufgeteilt werden. Hierbei wird oft die für die Datenhaltung zuständige Logik auf den Server ausgelagert, damit sie auch von anderen Clients genutzt werden kann. Die Verteilung der Anwendungslogik muss gut überdacht werden, um eine gute Balance zwischen Performance und Wiederverwendbarkeit zu garantierten.
- *Datenhaltung:* Der Server stellt ein globales Datenmodell zur Verfügung, mit dem der Client über die Anwendungslogik interagieren kann. Der Client hat in diesem Szenario die Möglichkeit sich ein eigenes Datenmodell vorzuhalten, um einen schnelleren Zugriff auf bestimmte oft benötigte Daten zu realisieren.

3.2.2.2 Lebenszyklus

Rich Client Anwendungen bleiben während des gesamten Nutzungszeitraumes aktiv (siehe *Abbildung 17*). Deshalb werden nur die Teile der Benutzerschnittstelle aktualisiert, die sich wirklich geändert haben.

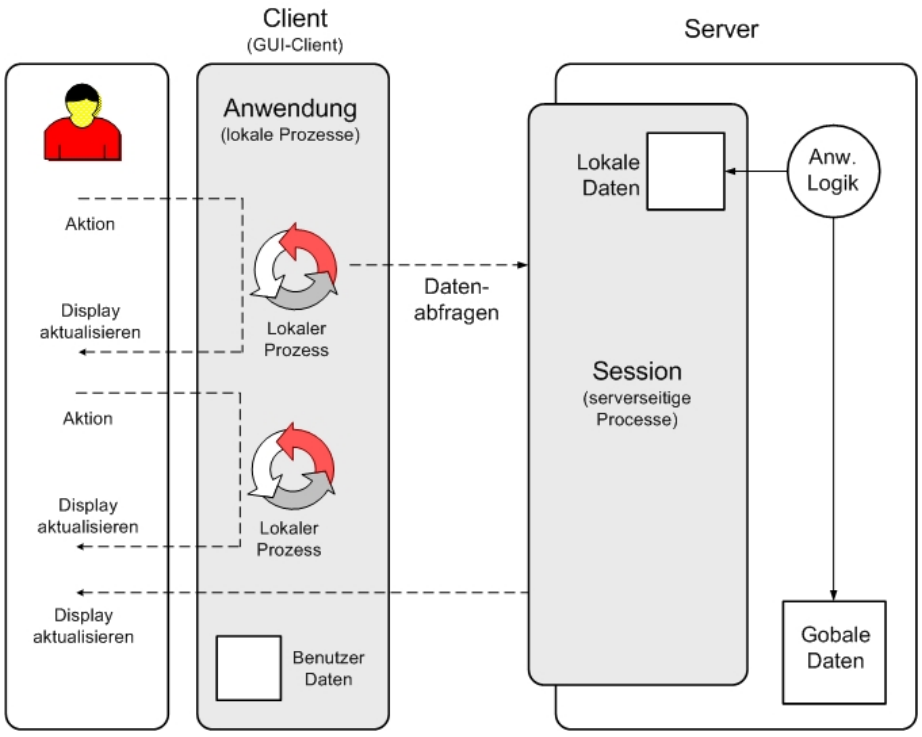

Abbildung 17: Lebenszyklus einer Rich Client Anwendung

Jede Interaktion des Benutzers mit der Benutzerschnittstelle hat zur Folge, dass ein lokaler Prozess im Hintergrund gestartet wird und die Aktion ausführt. Durch das Starten dieses Prozesses ist man unabhängig von der Netzwerklatenz und die Benutzerschnittstelle wird nicht blockiert.

3.2.2.3 Vorteile und Probleme

Durch den Einsatz von Rich Clients ergeben sich diverse Vorteile und Probleme, die nachfolgend aufgeschlüsselt und betrachtet werden.

Vorteile von Rich Clients:

- Verteilung der Rechenleistung auf n Clients, die unabhängig voneinander agieren können
- geeignet für Software die Spezialaufgaben für verschiedene Benutzergruppen übernimmt
- relativ ausfallsicher und unabhängig von Netzwerkhardware und verteilten Systembestandteilen
- Realisierung von ausgefeilten und anspruchsvollen Benutzerschnittstellen
- geeignet für Software mit hoher Stabilität

Probleme durch Rich Clients:

- unterschiedliche Performance der Anwendung, da an die lokalen Clients hohe Anforderungen gestellt werden
- aufwendige Ausbreitung der Software auf n Clients
- schwierige Überwachung der Anwendungen
- Isolation beim Zugriff auf die Ressourcen des Clients und die damit verbundenen Synchronisationsaufwände
- Bearbeitung großer Datenmengen, da Caching Mechanismen nicht von allen Benutzern verwendet werden können

3.2.2.3.1 Benutzerschnittstelle

Mit Rich Clients lassen sich dynamische und interaktive Benutzerschnittstellen erzeugen. Es stehen höherwertige Benutzerelemente (z.B. Tabellen, Listen, Baumstrukturen, usw.), Drag&Drop und ein ausgefeiltes Event-Handling zur Verfügung.

Da der Client auf dem Zielsystem installiert ist, ergeben sich Möglichkeiten die Benutzerschnittstelle besser an die Bedürfnisse des Anwenders anzupassen bzw. zuzuschneiden. So können Möglichkeiten implementiert werden, das Aussehen der Benutzerschnittstelle spezifisch anzupassen (z.B. Formatierungen, Menudarstellungen).

3.2.2.3.2 Ausbreitung

Rich Client Anwendungen müssen auf dem Zielsystem mit Hilfe eines Installationsprogramms oder manuell installiert werden. Außerdem müssen z.B. bei Java oder .NET Laufzeitumgebungen auf dem Zielsystem vorhanden sein. Diese Laufzeitumgebungen werfen bei der Installation Probleme auf, denn sie können nicht vorausgesetzt werden und müssen in diesem Fall nachinstalliert werden.

Die Systeme (z.B. Windows XP, Eclipse) sollen sich eigenständig aktualisieren und den Hersteller über aufgetretene Fehler informieren können.

Doch um solche Mechanismen zu ermöglichen, muss die Anwendung spezielle Updatemodule bereitstellen. Diese Module verbinden sich mit dem Updateserver, authentifizieren sich und verwenden einen gesicherten Übertragungskanal. Die Anwendung sollte aus mehreren Komponenten bestehen, damit das übertragene Datenvolumen reduziert werden kann. Aber dieses Vorgehen kann eine Vielzahl von Problemen mit sich bringen. So müssen die Abhängigkeiten zwischen den verschiedenen Komponenten definiert und festgestellt werden. Anschließend muss eine Überprüfung stattfinden, ob die aktualisierte Anwendung noch lauffähig ist und festgestellt werden, ob die Komponenten der Anwendung kompatibel zueinander sind.

3.2.2.3.3 Isolation und Synchronisation

Schwierig wird es, sobald Rich Client Anwendungen auf sicherheitsrelevante Daten zugreifen sollen (z.B. lokale Dateien, Starten von Anwendungen), Funktionalität die jedoch je nach Anforderungen an die Anwendungen für die Akzeptanz wichtig sind. Eine Lösung für diese Problematik wäre den Client in einer Sandbox laufen zu lassen. An die Sandbox ist in diesem Fall der Anspruch zu stellen, dass sie für jede Anwendung spezifisch konfiguriert werden kann.

Da sich der Client ein Datenmodell vorhalten kann, besteht die Notwendigkeit bei Synchronisation der lokalen Daten mit dem serverseitigen Datenmodell auf verschiedene sicherheitsrelevante Aspekte (z.B. Datenintegrität, Datensicherheit, Benutzerberechtigung) zu achten.

3.2.2.3.4 Performance

Die Verteilung der Rechenleistung ermöglicht die Entlastung der Server und Netzwerke, dadurch verbessern sich die Antwortzeiten für den Benutzer enorm. Für die Übertragung der Daten zwischen Client und Server können schnellere Protokolle (z.B. TCP) und binäre Datenformate eingesetzt werden.

Die Performance ist jedoch sehr stark von den Gegebenheiten des Rechners (z.B. RAM, Prozessor) abhängig. Hier kann allerdings nicht viel unternommen werden, auch wenn die JVM z.B. beim Laden von Klassen hilft. Jedoch kann ein Teil der Datenaufbereitung von der Datenbank übernommen werden, was aber zu einer unnötigen Belastung durch den Netzwerktraffic führen kann.

Im Besonderen Clients mit geringem Speicher können sehr schnell beim Bearbeiten großer Datenmengen blockieren. Hier kann eine Vorselektion der Daten helfen, was aber im Gegenzug lange Wartezeiten bedeutet. Die meisten Anwendungen halten häufig genutzte Daten (Prefetching) mit Hilfe mehrerer Prozesse (Threads) im Hintergrund vor.

3.2.3 Rich Internet Applications

RIAs sind Anwendungen, die in einem Browser ausgeführt werden, aber trotzdem eine Benutzeroberfläche zur Verfügung stellen, wie sie von Desktop-Anwendungen bekannt ist. Ihr Ziel ist es durch die Kombination der Vorteile von Rich und Thin Clients (siehe *Abbildung 18*) den Anwendungsbereich von Webanwendungen zu erweitern. Der Begriff Rich Internet Application lässt sich deshalb wie folgt aufschlüsseln:

- *Rich* steht für reichhaltige Möglichkeiten (z.B. Drag&Drop) und Leistungsfähigkeit (z.B. clientseitige Berechnungen)
- *Internet* steht für die Nutzung von Internettechnologien (z.B. HTTP)
- *Application* steht für die Realisierung von „Anwendungen“ trotz der Ausführung im Browser

RIAs können in den verschiedensten Technologien realisiert werden, die in Abschnitt 5.2.1 angesprochen werden.

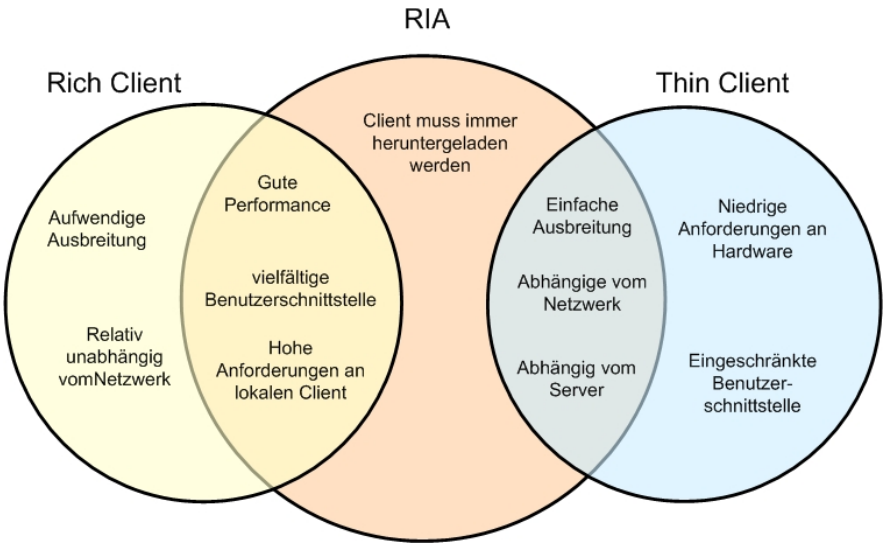

Abbildung 18: RIAs vereinen die Vorteile von Rich und Thin Clients

3.2.3.1 Lebenszyklus

Die Architektur von RIAs wird nicht separat besprochen, da sie der in Abschnitt 3.2.2.1 vorgestellten Architektur von Rich Clients gleicht. In *Abbildung 19* ist der Lebenszyklus von RIAs dargestellt. Im Gegensatz zu Thin Client Anwendungen wird im Browser eine Anwendung ausgeführt, die während der gesamten Sitzung aktiv ist. Der Client kann auf Benutzeraktionen reagieren, sie an den Server weiterleiten oder direkt bearbeiten. Die Benutzerschnittstelle wird während der Anfragen nicht mehr blockiert, wie es bei Thin Client Anwendungen der Fall ist (siehe Abschnitt 3.2.1). Das wird durch das asynchrone Kommunikationsmodell ermöglicht, das bereits in Abschnitt 2.2.1.3 besprochenen wurde.

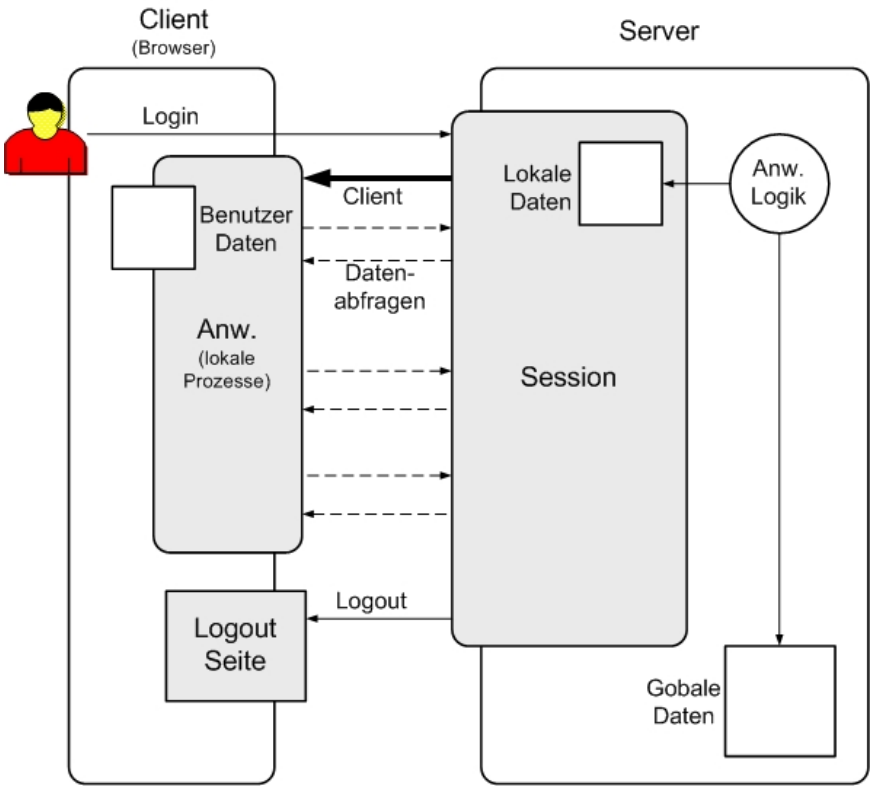

Abbildung 19: Lebenzyklus von RIAs

3.2.3.2 Vorteile und Probleme

Wie bereits erwähnt, vereinen RIAs die Vorteile von Rich (siehe Abschnitt 3.2.2) und Thin (siehe Abschnitt 3.2.1) Client Anwendungen, aus diesem Grund werden die Vorteile und Probleme prinzipiell angesprochen.

Vorteile von RIAs:

- Verteilung der Rechenleistung auf n Clients, die unabhängig voneinander agieren können
- Realisierung von ausgefeilten und anspruchsvollen Benutzerschnittstellen
- hohe Stabilität und niedrige Wartungsaufwände
- einfache Ausbreitung und Softwareänderungen sind sofort beim Client sichtbar

Probleme durch RIAs:

- unterschiedliche Performance der Anwendung, da an die lokalen Clients hohe Anforderungen gestellt werden
- Antwortzeit ist stark abhängig vom Netzwerkverkehr und der Serverperformance
- Ausfall des Servers hat den Ausfall des Clients zur Folge
- längere Wartenzeiten können unter Umständen durch das herunterladen des Clients entstehen

3.2.3.2.1 Wartezeiten durch herunterladen des Clients

Der Client muss beim Starten der Anwendung herunter geladen werden, da er nicht auf dem Zielsystem installiert ist. Dieser Zeitraum ist stark abhängig vom Umfang der Anwendung.

Aus diesem Grund sollten geeignete Mechanismen vorhanden sein, damit Teile der Anwendung nachgeladen werden können, wenn sie benötigt werden. Ein Beispiel ist in Abschnitt 7.3.1.2 verfügbar.

4 Kriterien der Clientparadigmen zur Erstellung einer Projektumfrage

In Abschnitt 3.2 wurden verschiedenen Clientparadigmen vorgestellt, darunter waren die Rich und Thin Clientparadigmen und die Rich Internet Applications. Momentan können Rich und Thin Client Anwendungen basierend auf der J2EE Plattform (siehe Abschnitt 2.3.3.1) entwickelt werden.

Ein Ziel der Diplomarbeit ist es, herauszufinden welche Clientparadigmen von Projekten eingesetzt werden und warum. Der Hintergrund besteht darin herauszufinden, ob es Sinn macht mit AJAX eine weitere Technologie zur Verfügung zu stellen, um PAI konforme Anwendungen zu entwickeln.

Da die Projekte jedoch sehr verschiedene Anforderungen an die zu entwickelnde Software, die eingesetzten Clientparadigmen und die PAI Plattform haben, ist es notwendig diese Anforderungen mit Hilfe einer Umfrage aufzunehmen. Damit die Umfrage durchgeführt werden kann, soll zuerst ein allgemeingültiger Kriterienkatalog erstellt werden. Mit dessen Hilfe ein Fragebogen zur Beurteilung der projektspezifischen Anforderungen entwickelt werden kann.

4.1 Aufbau des Kriterienkataloges

Der Kriterienkatalog soll aus allgemeingültigen Qualitätsanforderungen an Software aufgebaut sein. Qualitätsanforderungen können in so genannte funktionale und nichtfunktional Anforderungen unterteilt werden.

4.1.1 Funktionale Anforderungen

Funktionale Anforderungen legen fest welche Dienste das Softwaresystem anbietet und werden im Allgemeinen durch Use-Cases beschrieben, die z.B. mit der Unified Modeling Language (UML) abgebildet werden können. Sie unterscheiden sich von Anwendung zu Anwendung und haben keinen allgemeingültigen Charakter, deshalb werden in diesem Zusammenhang keine Bespiele genannt.

4.1.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben die gewünschte Qualität der Dienste und sind bedingt durch die umgangssprachliche Formulierung sehr missverständlich. Diese sind in den verschiedensten Qualitätsmodellen und Standards definiert, auf die in diesem Zusammenhang lediglich hingewiesen wird (z.B. Deutsche Gesellschaft für Qualität [DeGeQua] oder ISO 9126 [ISO9126]).

4.1.2.1 Wartbarkeit

Die Wartbarkeit beschreibt die anfallenden Kosten für Programmierarbeiten nachdem das Softwaresystem aufgeliefert wurde. Es hängt also maßgeblich von der Qualität der Softwareerstellung ab und ist ein auf keinen Fall zu unterschätzender Gesichtspunkt. Wichtige Punkte in diesem Zusammenhang sind:

- *Testbarkeit:* Der durch Testen entstehende Aufwand bei Änderungen an der Software.
- *Stabilität:* Beschreibt die unerwarteten Auswirkungen oder Seiteneffekte die bei einer Änderung am System auftreten können.
- *Analysierbarkeit:* Der entstehende Aufwand, um Mängel oder Ursachen von Fehlfunktionen und zu ändernde Bestandteile zu ermitteln.

4.1.2.2 Sicherheit

Mit Sicherheit ist die Fähigkeit von Software gemeint, den unautorisierten Zugriff, sowohl versehentlich als auch vorsätzlich, auf geschützte Daten und Software zu verhindern. Besonders im Umfeld von verteilten Anwendungen ist Sicherheit, sei es auf dem Transportweg, Client oder Server schwer zu realisieren. In diesem Zusammenhang müssen z.B. Integrität und Vertraulichkeit gewährleistet werden.

4.1.2.3 Zuverlässigkeit

Die Zuverlässigkeit von Software beschreibt die Fähigkeit die definierten Anforderungen über einen bestimmten Zeitraum hinweg zu erfüllen. Sie hängt zu einem Großteil von der Güte der Implementierung ab, die wiederum durch sorgfältiges Testen und eine gründliche Wartung gewährleistet wird. Wichtige Gesichtspunkte sind, wenn man von Zuverlässigkeit von Software spricht:

- *Reife:* Beschreibt die geringe Ausfallrate von Anwendungen im Fehlerfall.
- *Fehlertoleranz:* Das Leistungsniveau trotz Fehlern bzw. nicht Einhaltung von Schnittstellen zu halten.
- *Robustheit:* Die Fähigkeiten von Software trotz kritischen bzw. böswilligen Eingaben ein stabiles System zu gewährleisten.
- *Widerherstellbarkeit:* Der Aufwand, um die Daten bzw. das Leistungsniveau der Software nach einem Absturz einfach wiederherzustellen.

4.1.2.4 Portabilität

Die Portabilität beschreibt den Aufwand, um eine Anwendung auf eine andere Plattform zu übertragen (z.B. eine Windows Anwendung auf Linux portieren). Dieses Ziel kann leichter durch Standardisierung im Bereich Programmiersprachen oder den Einsatz von Virtuellen Maschinen (VM) erreicht werden.

4.1.2.5 Ausbreitung

Die Ausbreitung von Software beschreibt den Prozess von der Erstinstallation einer Anwendung über die Aktualisierung bis hin zur Deinstallation.

4.1.2.6 Bedienbarkeit

Eine einfache und leicht zu bedienende Benutzeroberfläche ist eine der wichtigsten Anforderungen an Software. Sie setzt sich aus folgenden Punkten zusammen:

- *Verständlichkeit:* Der Aufwand des Benutzers, um das Konzept der Anwendung zu verstehen.
- *Übersichtlichkeit:* Eine übersichtliche Benutzerschnittstelle ermöglicht den effizienten Einsatz der Anwendung.
- *Erlernbarkeit:* Beschreibt den vom Benutzer betriebenen Aufwand die Anwendung zu erlernen und zu verstehen.

4.1.2.7 Performance

Die Performance von Software kann mit Hilfe von verschiedenen Eigenschaften beurteilt und ausgedrückt werden:

- *Zeitverhalten:* Es beschreibt den Zeitaufwand für die Ausführung, Datendurchsatz und die Antwortzeit auf ein Ereignis.
- *Verbrauchsverhalten:* Beschreibt die Menge, der vom System zur Verfügung gestellten Mittel (z.B. CPU-Zeit, Speicher, Anzahl der Festplattenzugriffe), damit eine Funktion erfüllt werden kann.

Die Beurteilung der Performance von Software sollte mit Hilfe eines Benchmarks[23] durchgeführt werden.

4.1.2.8 Wirtschaftlichkeit

Die Wirtschaftlichkeit der Software lässt sich durch die Gegenüberstellung von Kosten und Nutzens beurteilen.

So müssen bei den Kosten die Lizenz- und Hardwarekosten, Beratungsleistungen sowie Schulungs- und Personalkosten berücksichtigt werden. Der Nutzen von Software lässt sich wesentlich schwieriger abschätzen. Aus diesem Grund sollte versucht werden, die durch den Einsatz der Software erzielte Kostenreduktion abzuschätzen. So können sich z.B. bessere Installations- und Update-Mechanismen in den Betriebskosten oder eine reichhaltige, interaktive und ansprechende Benutzeroberfläche in der Produktivität und somit in den Personalkosten niederschlagen.

---

[23] Beschreibt ein Konzept, um Verbesserungsmöglichkeiten durch den Vergleich von Leistungsmerkmalen mehrerer vergleichbarer Prozesse oder Programme zu finden (http://de.wikipedia.org/wiki/Benchmark).

4.1.2.9 Zukunftssicherheit

Die IT ist ein schnelllebiger Bereich, der von einer enormen Innovationsvielfalt durchsetzt ist. Neue Technologien kommen und gehen. Hier ist es aus Unternehmenssicht wichtig nicht auf jeden Trend zu reagieren, sondern kritisch zu betrachten, ob diese Technologien sich auf dem Markt behaupten können und nicht nur ein „Hype“ sind, sondern eine echte alternative zu vorhandenen Technologien.

4.1.2.10 Widerverwendbarkeit

Unter Widerverwendbarkeit ist die Eigenschaft von Software zu verstehen, Komponenten bzw. Module ganz oder teilweise in anderen Anwendungen wieder einzusetzen. Durch die stetige Widerverwendung und Weiterentwicklung dieser Bestandteile ergibt sich der positive Effekt, dass weniger Fehler in diesen Komponenten und somit der Software vorhanden sind, da sie ausführlich getestet sind.

4.1.2.11 Komplexität

Die Komplexität beschreibt den Aufwand, um eine Software zu verstehen und hat damit auch Auswirkungen auf die Wartbarkeit und Entwicklung von Software. Getrieben durch die heutige Vielfalt an Technologie, Programmiersprachen, usw. steigt die Komplexität von Software immer mehr an.

4.2 Einordnung der Kriterien

Die im Abschnitt zuvor besprochenen allgemeinen Qualitätsanforderungen haben nicht in jedem Projekt denselben Stellenwert. So kann z.B. die Sicherheit in einem Projekt eine sehr wichtige Rolle spielen, da die Benutzer der Software auf vertrauliche Daten zugreifen und bei diesem Zugriff bestimmte Sicherheitsrichtlinien gelten, während in einen anderen Projekt dieser Punkt eine eher untergeordnete Rolle spielt, da lediglich Informationen angezeigt werden und jeder das Recht hat diese zu lesen.

Die Sichtweisen auf den verschiedenen Anforderungen sind völlig flexibel und können auch auf andere Bereiche angewendet werden. In diesem Zusammenhang sollen zwei Sichtweisen auf die Anforderungen kurz vorgestellt werden, dies sind die PAI und Projektsichtweise.

4.2.1 PAI Sichtweise

Der Stellenwert der einzelnen Kriterien des Kataloges ist in *Abbildung 20* dargestellt. Durch die Integration der Plattform in die DCX Infrastruktur stehen Punkte wie z.B. Sicherheit, Wartbarkeit und Zuverlässigkeit an erster Stelle.

Abbildung 20: PAI Sichtweise auf den Kriterienkatalog

4.2.2 Projektsichtweise

Ein Softwareprojekt hat im Gegensatz zu PAI wesentlich andere Anforderungen an eine Technologie. Diese sind jedoch von Projekt zu Projekt so differenziert, dass unmöglich alle Kombinationen abgedeckt werden können und darauf verzichtet wird.

Ein wichtiger Punkt ist jedoch, dass die Kriterien je nach Projekt anders kategorisiert sein können oder einzelne Punkte wegfallen. Wie die verschiedenen Projekte die Kriterien im Betrieb für ihre Belange eingeschätzt haben kann der Auswertung der Umfrage entnommen werden, die in einem externen Dokument ([UMFRAGE]) verfügbar ist.

4.3 Durchführung der Projektumfrage

Die Umfrage unter den Anwendern der PAI Plattform soll helfen, die Stärken und Schwächen der eingesetzten Clientparadigmen in Bezug auf die jeweiligen Projektanforderungen zu identifizieren.

In der folgenden Liste werden die Fragen noch einmal zusammengefasst:

- Für welches Clientparadigma hat man sich entschieden und warum?
- In welchem Umfeld wurden die Clientparadigmen eingesetzt?
- Wie zufrieden sind die Projekte mit der Wahl des Clientparadigmas?
- Welche Probleme traten auf?
- Wo sieht man Potential für Verbesserungen?
- Was halten die Projekte von AJAX?
- Wo sehen sie Problematiken bei AJAX?

Die Befragung wurde mit ausgewählten Projekten durchgeführt, dabei wurde besonders darauf geachtet, dass möglichst eine ausgewogene Verteilung der von PAI unterstützten Clientparadigmen vorhanden war.

4.3.1 Erstellung des Fragebogens

Der Fragebogen besteht aus sechs Teilen. Der erste Teil besteht aus allgemeinen Fragen, die das Projekt bzw. die Person betreffen. Der Hauptteil wurde in vier Abschnitte eingeteilt, die sich am Rational Unified Process (RUP) orientieren. Abgerundet wurde der Fragebogen durch einige abschließende Fragen zum Thema AJAX.

Der Hauptteil soll näher betrachtet werden und ist in der folgenden Abbildung grob dargestellt. Die dargestellten Abschnitte repräsentieren die Meilensteine wie sie bei RUP ([RUP]) definiert sind.

- *Inception:* Konzeption/Spezifikation
- *Elaboration:* Design/Entwurf
- *Construction:* Implementierung/Test
- *Transition:* Produktübergabe

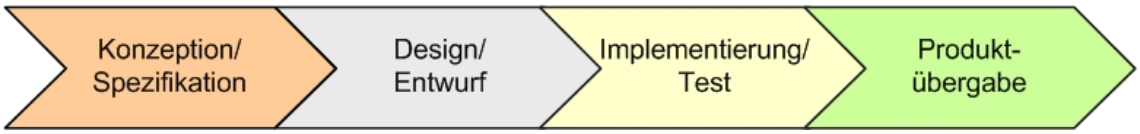

Abbildung 21: Struktur PAI Fragebogen

Die in Abschnitt 4.1 definierten Kriterien wurden den einzelnen Kategorien sinngemäß zugeordnet. Da die Zuordnung nicht immer eindeutig war, wurde die aus Projektsicht logischste Zuordnung gewählt. Der komplette Fragenkatalog wird an dieser Stelle jedoch nicht vorgestellt und ist in einem externen Dokument ([FRAGEBOGEN]) verfügbar.

4.3.2 Ergebnis der Umfrage

Die Auswertung der Umfrage ließ folgende Schlüsse zu, die nachfolgend betrachtetet werden.

4.3.2.1 Einsatzgebiete der Clientparadigmen

Die Einsatzgebiete der Clientparadigmen werden nun kurz dargestellt.

- *Thin Client:* Thin Client Anwendungen werden häufig dann eingesetzt, wenn Anwendungen global im Konzern erreichbar sein sollen oder die Anwendung extern (z.B. Zuliefererfirmen) genutzt wird.

- *Rich Client:* Rich Client Anwendungen werden häufig in einem Umfeld eingesetzt, wo eine reichhaltigere und interaktivere Benutzerschnittstelle von Nöten ist. Das sind zumeist Anwendungen die eher lokal verfügbar (z.B. nur an einem Standort) sind.

Dieser Umstand zeigt, dass verschiedene Clientparadigmen benötigt und eingesetzt werden.

4.3.2.2 Verschiedene Anforderungen an die Software

Die Projekte haben sehr differenzierte Anforderungen an die zu erstellende Software und deswegen gibt es die verschiedensten Gründe warum sie sich für oder gegen ein Clientparadigma entscheiden. Diese Gründe sollen jedoch an dieser Stelle nicht aufgezählt werden.

Die gestellten Anforderungen an eine Anwendung können nur schwerlich von einem Clientparadigma erfüllt werden. Die Unterstützung mehrerer Paradigmen durch die PAI Plattform ermöglicht es besser auf diese Anforderungen zu reagieren und sie zu erfüllen.

4.3.2.3 Einsatz von AJAX

In Bezug auf den Bekanntheitsgrad von AJAX unter den Projekten ließen sich die folgenden drei Sichtweisen ableiten:

- *AJAX ist völlig unbekannt:* In diesem Fall kann AJAX von den Projekten in den Entscheidungsprozess zur Auswahl des Clientparadigmas nicht mit einbezogen werden. Das stellt ein Problem dar, denn AJAX kann Anforderungen erfüllen, die von den betrachteten Paradigmen nicht erfüllt werden.
- *AJAX ist bekannt:* Einigen Projekten ist AJAX als Schlagwort bekannt. Jedoch wurde sich mit der Technologie noch nicht auseinandergesetzt, weshalb noch keine Wertung stattgefunden hat. Als der Begriff JavaScript fiel, wurde die Technologie generell als ein Sicherheitsrisiko eingestuft und somit abgelehnt.
- *AJAX wird eingesetzt:* Einige Projekte nutzen AJAX, um die Benutzeroberfläche ansprechender und flüssiger zu gestalten, im operativen Einsatz. Teilweise wurde jedoch nicht getestet, welche Integrationsprobleme beim Einsatz von AJAX mit der PAI Plattform entstehen. Dieser Punkt ist sehr bedenklich.

Es muss mehr Aufklärungsarbeit von PAI geleistet werden, um den Projekten ein Verständnis von AJAX zu vermitteln und das wahre Potential von AJAX näher zu bringen. Weiterhin müssen die Risiken, die durch den Einsatz von AJAX entstehen können, vermittelt werden, damit AJAX zu keinem Sicherheitsrisiko wird.

5 AJAX im Detail

Nachdem es in Abschnitt 2.2 eine Einführung in die hinter dem Begriff AJAX stehenden Technologien gab und in Abschnitt 3.2 verschiedene Clientparadigmen vorgestellt wurden, beschäftigt sich dieses Kapitel detaillierter mit AJAX.

Im ersten Teil werden Probleme aus verschiedenen Blickwinkeln betrachtet, auf die der Programmierer beim Entwickeln von AJAX-Anwendungen achten muss. Es werden teilweise Lösungsmöglichkeiten bzw. Prinzipien aufgezeigt, wie die Probleme gelöst werden können.

Nachfolgend werden verschiedene Lösungen vorgestellt, um AJAX-Anwendungen einfacher zu entwickeln. Sie werden hinsichtlich ihrer Vorteile und Probleme betrachtet.

Dieser Teil ist besonders für Entwickler interessant, da einige momentan auf dem Markt befindliche Produkte vorgestellt werden und eine Empfehlung ausgesprochen wird, wie diese eingesetzt werden können.

5.1 Problematiken

Der Entwickler muss sich beim Entwickeln von AJAX-Anwendungen verschiedener Probleme bewusst sein. Zum besseren Verständnis werden diese kategorisiert und angesprochen.

5.1.1 Browser

Beim Entwickeln von Anwendungen, die in einem Browser lauffähig sein sollen, ist man an gewisse Beschränkungen gebunden, die dem Entwickler durch das System vorgegeben werden.

5.1.1.1 Aktivierung von JavaScript

Um AJAX-Anwendungen nutzen zu können muss JavaScript aktiviert sein. Doch viele Benutzer deaktivieren JavaScript und können die Webseiten nicht nutzen.

Falls AJAX-Anwendungen mit dem IE genutzt werden sollen, muss ActiveX aktiviert sein, da die Versionen bis einschließlich 6.0, das für die asynchrone Kommunikation notwendige XMLHttpRequest-Objekt als ActiveX Komponente implementiert haben. Beim IE 7.0 wurde dies jedoch geändert und eine native Implementierung steht zur Verfügung (siehe Anhang B1).

5.1.1.2 Rückmeldungen

Asynchrone Aufrufe werden im Browser nicht visualisiert und der Benutzer kann den Eindruck bekommen, dass die Anwendung zähflüssig abläuft. Hier muss der Entwickler

dafür Sorge tragen, dass der Benutzer über laufende Aktionen durch visuelle Effekte (z.B. Sanduhr) informiert wird und somit weiss, dass Hintergrundaktivitäten stattfinden.

5.1.1.3 Barrierefreiheit

Barrierefreie Webseiten sind mit AJAX nicht zu realisieren, da sie ohne JavaScript lauffähig sein und dem W3C-WAI WCAG 1.0 Standard ([W3CWAI]) genügen müssen. Entwickler sollten aus diesem Grund eine statische Ersatzvariante der Webseite zur Verfügung stellen.

5.1.1.4 Interaktion mit Desktop-Anwendungen

Eine direkte Interaktion zwischen AJAX- und Desktop-Anwendungen ist nicht möglich, da aus Sicherheitsgründen von JavaScript nicht auf die Zwischenablage zugegriffen werden kann.

5.1.1.5 Browser Cache

In manchen Browsern (z.B. IE) kann es zu Problemen mit dem Cache bei einem asynchronen Aufruf kommen. Das Problem ist, dass Daten oftmals über dieselbe URL angefordert werden. Was den Browser dazu veranlassen kann diese Daten im Browser-Cache zu speichern und nicht die aktuellen sondern die gespeicherten Daten zurückzuliefern. Um dies zu umgehen, sollte bei jeder asynchronen Anfrage ein spezieller HTTP-Header gesetzt werden („no-cache“). Das selbe Problem kann übrigens auch beim Einsatz von Proxy-Servern auftreten.

5.1.2 Entwicklung

Abgesehen von den in Abschnitt 5.1.1 angesprochenen Problemen die durch den Browser entstehen, werden in diesem Abschnitt Probleme angesprochen die speziell beim Entwickeln von Interesse sind. Die aufgeführten Punkte können teilweise auch dem vorhergehenden Abschnitt zugeordnet werden, aber der Unterschied besteht darin, dass sie vom Programmierer gelöst werden können.

5.1.2.1 History Problem

Ein oft angesprochener Nachteil von AJAX-Anwendungen ist die mangelnde Unterstützung der Browser-History und des Vor- und Zurück-Knopfes. Das mag auf den ersten Blick vielleicht nicht so wichtig sein, aber es kann eine entscheidende Rolle für eine gute Benutzbarkeit von Anwendungen sein.

Der Grund warum diese Funktionalitäten mit AJAX nicht so einfach zu realisieren sind, ist die Tatsache, dass der Browser-Cache für statische Seiten ausgelegt ist. Eine AJAX-Anwendung besteht nicht aus vielen statischen Seiten, es wird lediglich eine initiale Seite geladen, deren Zustand über den DOM manipuliert wird. Die Seite existiert also nur im Speicher des Browsers und wird nicht in der Browser-History gespeichert.

Das Problem kann gelöst werden, indem die Browser-History über einen unsichtbaren iFrame gefüllt wird.

5.1.2.2 Browserinkompatibilitäten

Um eine auf allen Browser lauffähige Anwendung zu realisieren, sind viele Tests notwendig. Durch die verschiedenen Browser-Implementierungen (z.B. IE, Firefox, Opera) entstehen oft Kompatibilitätsprobleme, trotz der Bemühungen des W3C diverse APIs (siehe Abschnitt 5.2.4) zu standardisieren. Dieses Problem wird oft dadurch umgangen, indem die Webanwendungen nur für bestimmte Browser freigegeben werden.

5.1.2.3 Bookmarks

Das Problem besteht darin, den Zustand einer Webseite zu speichern, die dynamisch aktualisiert wird. Im Fall von AJAX-Anwendungen bedeutet dies, alle Daten im DOM die zur Wiederherstellung des Zustandes einer Webseite nötig sind zu speichern. Das ist natürlich völlig unrealistisch.

Es gibt auch für dieses Problem eine Lösung, meistens wird in dem Zusammenhang der Anker in der gegenwärtigen URL je nach Zustand der Seite gesetzt.

5.1.2.4 Memory Leaks

Wie bereits erwähnt wurde, sind normale Webanwendungen seitenorientiert. Bei jedem Klick auf einen Link wird die alte Seite durch die Neue überschrieben, wenn das passiert werden alle Ressourcen, die durch den DOM der alten Webseite benutzt waren, durch die Garbage-Collection[24] von JavaScript wieder freigegeben.

Das ist bei AJAX-Anwendungen nicht mehr der Fall, denn diese bestehen - zumindest meistens - aus einer initialen Webseite, die durch Manipulation des DOM in ihrem Aussehen verändert wird. Es werden Daten vom Server übertragen, die im Client ausgewertet werden. Mit Hilfe dieser Daten werden neue DOM-Objekte erstellt und das Aussehen der Benutzerschnittstelle angepasst. Der Programmierer ist an dieser Stelle für sämtliche Aufräumarbeiten zuständig, also muss er den Speicher, der durch die nicht mehr benötigten Objekte weiterhin belegt bleibt, freigeben. Falls das nicht geschieht entstehen Memory Leaks und es kann vorkommen dass der Speicherverbrauch von Browsern in die Höhe schnellt (z.B. 500MB Footprint im Firefox).

5.1.2.5 Speicherung von Daten auf dem Client

Die permanente Speicherung von Daten auf dem Client ist wegen den Sicherheitsrichtlinien der JavaScript Sandbox nicht möglich. Für die Akzeptanz von Anwendungen durch den Benutzer kann die Speicherung von Daten aber notwendig sein.

---

[24] Die Garbage-Collection ist ein Verfahren, um nicht mehr benötigten Speicher automatisch wieder freizugeben (http://de.wikipedia.org/wiki/Garbage-Collection).

Es besteht die Möglichkeit mit Hilfe von Flash oder Cookies Daten zu speichern. Diese Lösungen können jedoch nur ein sehr begrenztes Datenvolumen abspeichern und stellen somit keine ernste Alternative dar.

5.1.2.6 Suchmaschinen

Suchmaschinen haben das Problem Webseiten zu indexieren, wenn der Inhalt der Webseite dynamisch erzeugt wurde. Für dieses Problem gibt es verschiedene Lösungsansätze, so können z.B. zusätzliche Links eingefügt werden oder eine zweite statische Webseite wird angelegt.

5.1.2.7 Echtzeitinformationen

Eine Schwäche von AJAX ist, dass es keine Möglichkeit gibt Anwendungen zur Anzeige von Echtzeitinformationen zu entwickeln. Falls Daten den Server nach dem aktualisieren der Benutzeroberfläche erreichen, werden diese nicht mehr angezeigt. Das liegt an den vorhandenen Einschränkungen durch HTTP bzw. der synchronen Kommunikation (siehe Abschnitt 2.2.1.3).

Mit Reverse AJAX existiert jedoch ein Mechanismus, mit dem dieses Manko beseitigt werden kann. Reverse AJAX ist nicht als eine Technologie zu betrachten, es ist genau wie AJAX eine Kombination von bereits existierenden Technologien und erlaubt es Daten in Echtzeit vom Server an den Client zu übertragen. Im Folgenden werden drei Möglichkeiten vorgestellt Reverse AJAX zu realisieren.

5.1.2.7.1 Polling

Das Grundprinzip besteht darin, den Server im Sekundentakt nach neuen Informationen zu fragen. Die folgende Abbildung zeigt eine typische Client/Server Kommunikation, in der die Übertragung von Echtzeitinformation mit Hilfe von Polling realisiert wird. Wie im Schaubild zu sehen ist, existiert neben der AJAX-Anwendung eine kontinuierliche serverseitige Anwendung.

Falls eine Änderung der Daten (z.B. neue Aktienkurse) stattfindet, wird ein Ereignis ausgelöst, das der serverseitigen Verarbeitung der Client-Anfragen die neue Situation mitteilt. Die clientseitige Anwendung sendet dem Server, gekennzeichnet durch die gestrichelte Linie, in regelmäßigen Abständen Anfragen (Poll). Im Fall das keine neuen Informationen vorliegen, wird eine leere Antwort (Response) gesendet, aber falls neue Informationen existieren werden sie an den Client übermittelt (durchgezogene Linie). Wie im Diagramm zu sehen ist, können zwischen den Aktualisierungsanfragen (Poll) auch Anfragen der AJAX-Engine stattfinden.

Das Problem an dieser Lösung ist, dass eine erhebliche Belastung des Webservers durch die ständigen Anfragen (Poll) auftritt. Stellt man sich eine Webseite mit Millionen von Benutzern vor, so muss der Server ständig Millionen von Anfragen beantworten, was sehr schnell zu einem Ausfall führen kann.

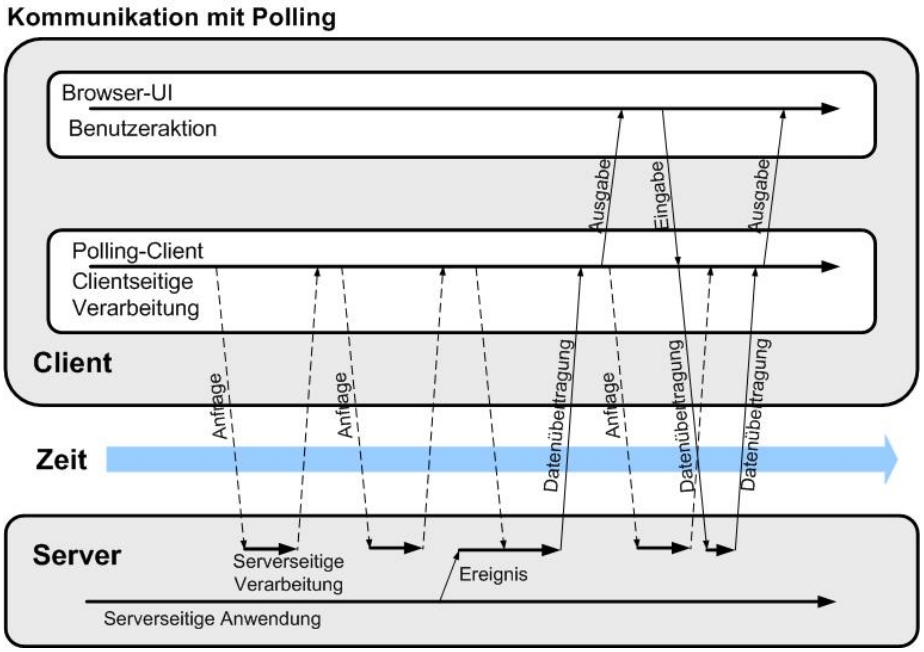

Abbildung 22: Polling, an Beispiel einer AJAX-Anwendung

5.1.2.7.2 Comet

Comet ermöglicht das kontinuierliche Aktualisieren einer Webseite, unter der Voraussetzung, dass die Verbindung zwischen Client und Server niemals abgebaut wird. Nachdem der Client die Verbindung zum Server aufgebaut hat, werden die Daten so langsam wie nur möglich an den Client gesendet.

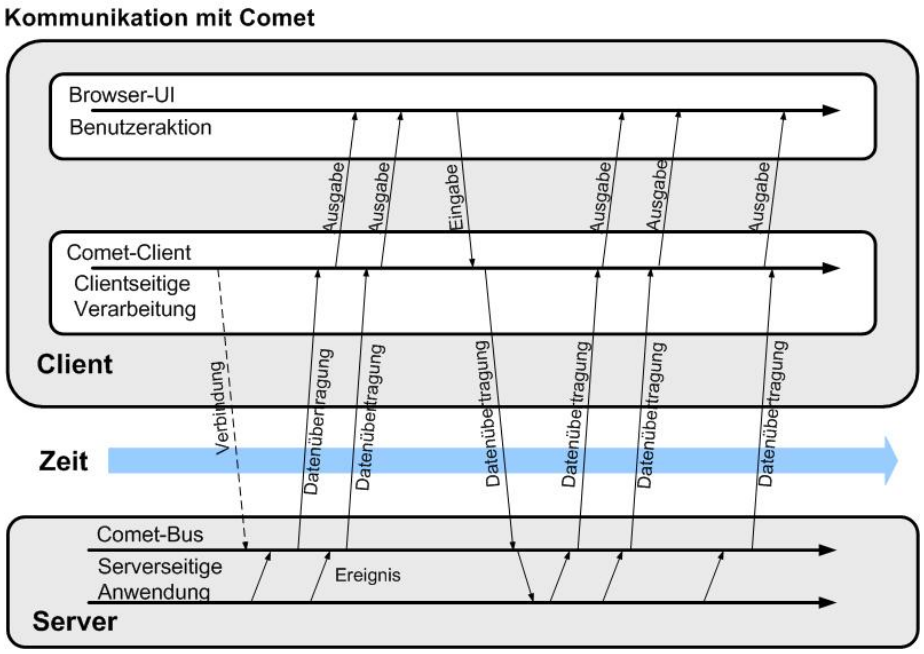

Abbildung 23: Comet, am Beispiel einer AJAX-Anwendung

An dieser Stelle stellt sich die Frage: Wie soll der Browser die Webseite jemals anzeigen? Comet arbeitet hier mit einem Trick. Dem Benutzer wird eine normale Webseite präsentiert, in der ein unsichtbarer iFrame eingebettet ist. In diesem iFrame werden die Daten im Hintergrund geladen und durch eine JavaScript Funktion in der normal geladenen Webseite ausgewertet (siehe *Abbildung 23*).

Auf diese Weise wird dem Benutzer eine Webseite präsentiert, aber es besteht die Möglichkeit Daten im Hintergrund zu laden und die Seite zu aktualisieren (asynchrone Kommunikation simulieren).

Der Vorteil an dieser Lösung gegenüber dem Polling ist, dass der Webserver nicht durch eine Unmenge von Verbindungsanfragen belastet wird.

5.1.2.7.3 Piggyback

Die Letzte der vorgestellten Lösungen ist Piggyback (siehe *Abbildung 24*). Falls neue Informationen den Server erreichen, wird ein Ereignis ausgelöst und die Daten zwischengespeichert.

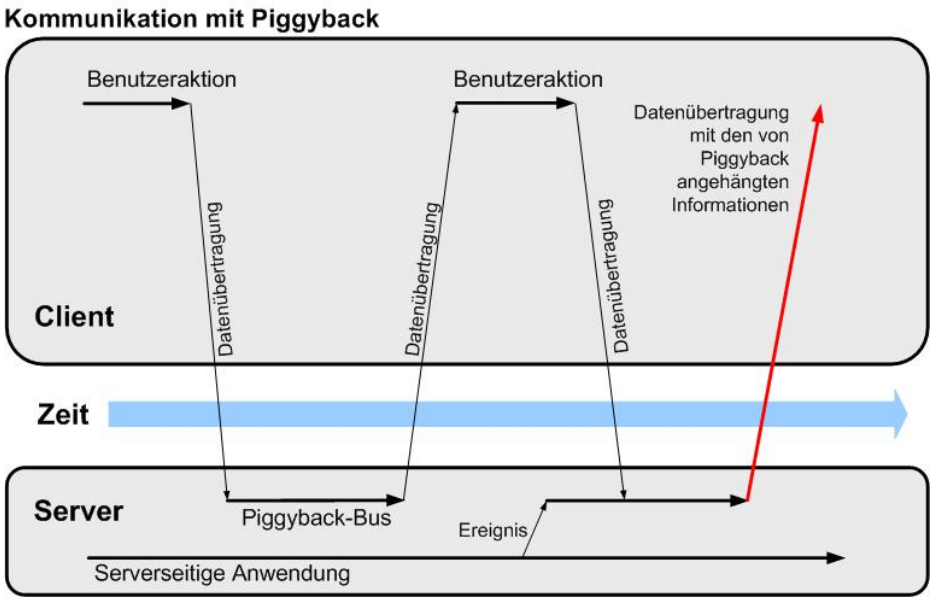

Abbildung 24: Piggyback, am Beispiel einer traditionellen Webanwendung

Wenn der Benutzer eine Aktion auslöst werden die zwischengespeicherten Daten an die Antwort auf die Anfrage angehängt und an den Client übertragen. Der Client hat nun die Möglichkeit diese Daten auszuwerten und die Benutzerschnittstelle zu aktualisieren. Piggyback ist eine sehr ressourcenschonende Lösung, aber bedingt durch die Tatsache, dass neue Daten nur durch zuvor stattgefundene Benutzeraktionen übertragen werden können, ist sie doch weniger dazu geeignet Echtzeitinformationen bereitzustellen.

All die vorgestellten Lösungen bieten die Möglichkeit, dem Benutzer Echtzeitinformationen bereitzustellen. Während sich Comet und Polling dadurch auszeichnen, dass keinerlei Benutzeraktionen nötig sind um Echtzeitinformationen an den Client zu übertragen, muss im Fall von Piggyback eine Benutzeraktion die Übertragung der Daten anstoßen. Der Einsatz in der DCX Infrastruktur ist im Fall von Polling und Comet sehr bedenklich, da durch die niemals abgebaute Verbindung kein Idle Timeout der Session stattfindet, was eine Verletzung der DCX Richtlinien darstellt.

An dieser Stelle muss auf Projektseite überprüft werden, welche Anforderungen bestehen und ob einer der vorgestellten Mechanismen dazu beitragen kann diese zu erfüllen.

5.1.3 Datenaustausch

Standardmäßig wird in AJAX das XML-Format (siehe Abschnitt 2.2.2.2.3) zum Datenaustausch zwischen Client und Server vorgeschlagen. Der Einsatz ist aber nicht zwingend notwendig und jedes andere Datenaustauschformat wie z.B. Klartext, Simple Object Access Protokoll (SOAP: [SOAP]), Representational State Transfer (REST: [REST]) oder JavaScript Object Notation (JSON: [JoGam06]) kann eingesetzt werden.

Hier stellt sich jedoch die Frage: Warum existieren so viel Datenaustauschformate und wo können sie am sinnvollsten eingesetzt werden? Diese Frage kann im Rahmen dieser Diplomarbeit nicht vollständig geklärt werden. Es währe jedoch sinnvoll die erwähnten Datenaustauschformate separat zu betrachten.

Da JSON ein relativ neues Datenaustauschformat ist, wird es im nächsten Abschnitt vorgestellt.

5.1.3.1 JavaScript Object Notation

JSON stellt ein leicht lesbares Datenaustauschformat für den Computer und Menschen dar. Es war ursprünglich für den Datenaustausch in JavaScript gedacht, hat aber Einzug in viele bekannte Programmiersprachen (z.B. PHP, C/C++, Java, C#, Perl) gehalten.

Mit XML (siehe Abschnitt 2.2.2.2.2) steht ein flexibles Datenaustauschformat zur Verfügung. Die Verarbeitung von XML mit DOM (siehe Abschnitt 2.2.2.2.5) ist zwar sehr bequem aber auch sehr langsam, zudem generiert XML viel Overhead. Oft wird aber ein schlankes Datenformat benötigt, das leicht weiter verarbeitet werden kann. Der Vorteil von JSON ist, dass es leichter zu parsen und von der Dateigröße schlanker als XML ist. Selbst der Austausch von Daten zwischen verschiedenen Programmiersprachen ist möglich, denn die Struktur von JSON ist in Objekten, Arrays, Daten und Zeichen unterteilt. Dabei sollten die folgenden Regeln beachtet werden:

- Objekte sind ungeordnete Sammlung von name/value Paaren und werden in geschweiften Klammern ({…}) notiert. Dem Objektnamen folgt ein Doppelpunkt (:), die name/value Paare sind mit einem Komma (,) voneinander getrennt

- Arrays sind geordnete Sammlung von Werten in eckigen Klammern ([…]). Die Werte sind mit einem Komma (,) voneinander getrennt
- Daten können Strings (in Anführungszeichen "…"), Zahlenwerte, boolsche Werte (true, false), Objekte oder Arrays sein. Die Strukturen können ineinander verschachtelt sein

Ein einführendes Beispiel in JSON soll mit Hilfe der JavaScript-Variante gegeben werden. In JavaScript wird JSON den Einsatz von Literal-Objekten realisiert. Die folgenden Abbildungen zeigen dieselbe Datenstruktur auf der Basis von JSON und XML.

```
var musik = {
      "data" : [
          {
                "Interpret" : "Kettcar",
                "Tracks" : ["Deiche", "48 Stunden", "Einer"]
          },     …….
     ]
};
```

Abbildung 25: Aufbau einer JSON Nachricht

Anhand des Beispiels ist gut zu erkennen, das die XML-Variante wesentlich aufwendiger zu erzeugen und fast doppelt so groß ist.

```
<data>
     <item>
          <interpret>Kettcar</interpret>
          <tracks>
               <track>Deiche</track>
               <track>48 Stunden</track>
               <track>Einer</track>
          </tracks>
      </item>     …….
</data>
```

Abbildung 26: Aufbau einer XML Nachricht

Der Zugriff auf die name/value Paare erfolgt nach dem JavaScript Syntax (Objekte, Arrays) und soll hier nicht näher erläutert werden. Nach der Übertragung liegt ein JSON-Objekt als String vor und muss vor der weiteren Verarbeitung wieder in sein Ausgangsformat umgewandelt werden. Dazu biete sich in JavaScript die eval() Funktion an, die es ermöglicht einen String als JavaScript-Code auszuführen. Diese Funktion ist jedoch sicherheitstechnisch bedenklich, da nur vertrauenswürdiger Code ausgeführt werden sollte (siehe Kapitel 6.2.1).

Ein anderes Problem beim Einsatz von JSON besteht darin, dass keine Beschreibung existiert, die definiert, wie die übertragenen Daten strukturiert sind. Hier spielen sicherheitstechnische Gründe (siehe Abschnitt 6.2.2) hinein, denn die Anwendung muss feststellen können ob die übertragenen Daten sich an die vereinbarten Konventionen halten. In XML existiert zu diesem Zweck die Document Type Definition (DTD). Diese Datei ist sehr wichtig, denn sie enthält eine Beschreibung, welche Elemente das Dokument enthalten darf und wie das Dokument strukturiert ist ([W3CXML]). Beim Einsatz von JSON hat der Entwickler nur die Möglichkeit diese Überprüfung selber vorzunehmen.

Abschließend ist noch zu sagen, dass JSON durchaus eine Alternative zu XML darstellen kann, denn der Ruf nach Einfachheit im Web 2.0 (siehe Abschnitt 2.1.2.5) spiegelt sich bei diesem Ansatz 1:1 wieder.

5.2 Toolkits und Frameworks

In Abschnitt 5.1 wurden diverse Probleme vorgestellt, die beim Entwickeln von AJAX-Anwendungen zu beachten sind. Diese Probleme müssen bei jeder Anwendung immer wieder neu gelöst werden. Deshalb wird nachfolgend betrachtet, welche Lösungsmöglichkeiten existieren, um AJAX-Anwendungen einfacher zu entwickeln.

5.2.1 Momentane Situation auf dem Markt

Zur Zeit existieren zwei Paradigmen zur Entwicklung von webtauglichen Benutzerschnittstellen. Das sind Anwendungen oder Technologien (z.B. JSP, PHP, ASP), die aus HTML bestehen bzw. generieren. Diese Technologien werden jedoch hier nicht betrachtet und wurden lediglich zur Vollständigkeit erwähnt.

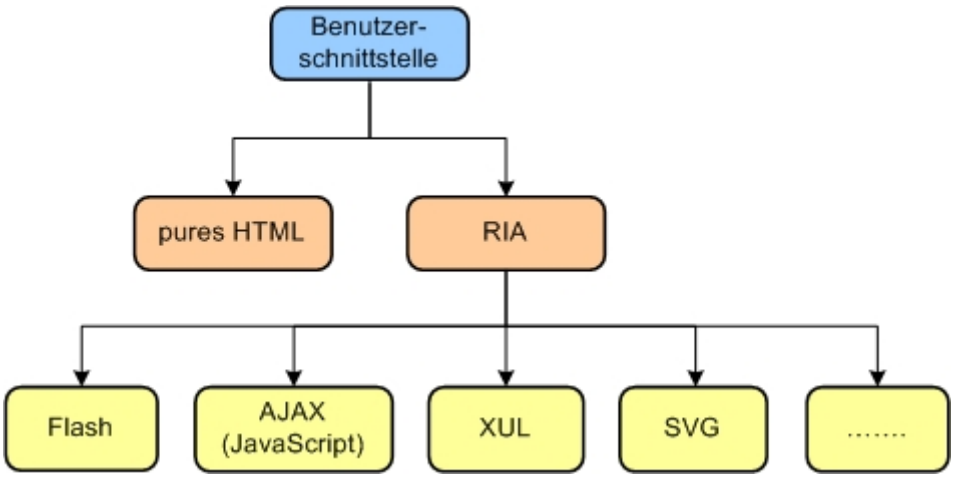

Abbildung 27: Marktsituation der Technologien für Webanwendungen

Außerdem gibt es verschiedene Technologien, die die Entwicklung von den bereits in Abschnitt 3.2.3 vorgestellten RIAs unterstützen. So können RIAs mit AJAX, Flash

([FLASH]), der XML User Interface Language (XUL: [XUL]) oder Scalable Vector Graphics (SVG: [SVG]) realisiert werden. Für Flash sind ausgereifte Lösungen vorhanden, wie z.B. Adobe Flex 2 ([FLEX2]) und OpenLaszlo ([OPENLASZ]). Wohingegen XUL und SVG nicht sehr verbreitet und noch zu fehleranfällig sind, um komplexe Anwendungen zu entwickeln. Auf diese Technologien wird jedoch nicht näher eingegangen und auf die Quellen verwiesen.

AJAX-Anwendungen können mit den verschiedensten Produkten realisiert werden, die ihrerseits auf den verschiedensten Technologien, wie z.B. ASP, JSF, JSP, PHP, usw. aufbauen. Einige Produkte werden im weiteren Verlauf dieser Arbeit vorgestellt (siehe Kapitel 5.2.6.1).

5.2.2 Anforderungen aus PAI Sicht

Beim Einsatz bzw. der Integration von neuen Technologien in die PAI Plattform, werden eine Vielzahl von Anforderungen an die Produkte gestellt. Es werden die in diesem Zusammenhang wichtigsten Anforderungen angesprochen. Eine vollständige Liste ist in einem externen Dokument verfügbar.

Architektur: PAI Projekte nutzen eine große Bandbreite an serverseitigen Technologien (z.B. JSP, JSF, Struts, Servlets, usw.), die vom eingesetzten Produkt möglichst vollständig unterstützt werden sollten.

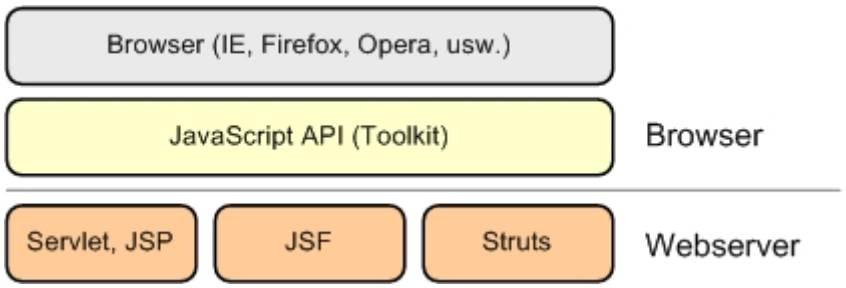

Abbildung 28: Anforderungen an ein Framework aus der PAI Sicht

Entwicklung: Das Entwickeln von AJAX-Anwendungen sollte möglichst einfach sein und die bedingt durch den Einsatz zusätzlicher Technologien steigende Komplexität bei der Entwicklung, sollte dem Programmierer möglichst verborgen bleiben. Außerdem sollten Entwicklungswerkzeuge und eine ausführliche Dokumentation vorhanden sein.

Integration: Eine leichte, umfassende und transparente Integration in die PAI Sicherheitsinfrastruktur muss möglich sein, damit Projekte die gewohnten Sicherheitsmechanismen nutzen können. Wünschenswert wäre eine nahtlose Integration in die vorherrschenden Programmierparadigmen der J2EE Plattform und die gewohnte Nutzung des PAI Login Modules.

Sicherheit: Die Kommunikation zwischen Client und Server sollte gegen Angriffe gesichert werden können. Die Nutzung von HTTP(s) sollte zur Verschlüsselung des Transportkanals eingesetzt werden können.

Support: Der Support für von PAI unterstützte Technologien muss möglichst lange gewährleistet werden, damit muss von Herstellerseite eine kontinuierliche Weiterentwicklung und ein ausreichender Support gewährleistet werden.

5.2.3 Warum sind Toolkits und Frameworks notwendig?

Dem Entwickler stehen zur Programmierung von AJAX-Anwendungen HTML, CSS, JavaScript und diverse Browserschnittstellen (z.B. DOM, XPath, XMLHttpRequest) zur Verfügung. Doch daraus ergeben sich einige Probleme, denn die Implementierung der Browserschnittstellen in den verschiedenen Browsern unterscheidet sich erheblich (siehe Abschnitt 5.1.2.2) und zur Realisierung von grafischen Benutzeroberflächen stehen HTML und CSS zur Verfügung. Die Fähigkeiten von HTML sind jedoch sehr begrenzt und der Entwickler muss Funktionen, die über die Basisfunktionalität hinaus gehen, mühsam nachimplementieren.

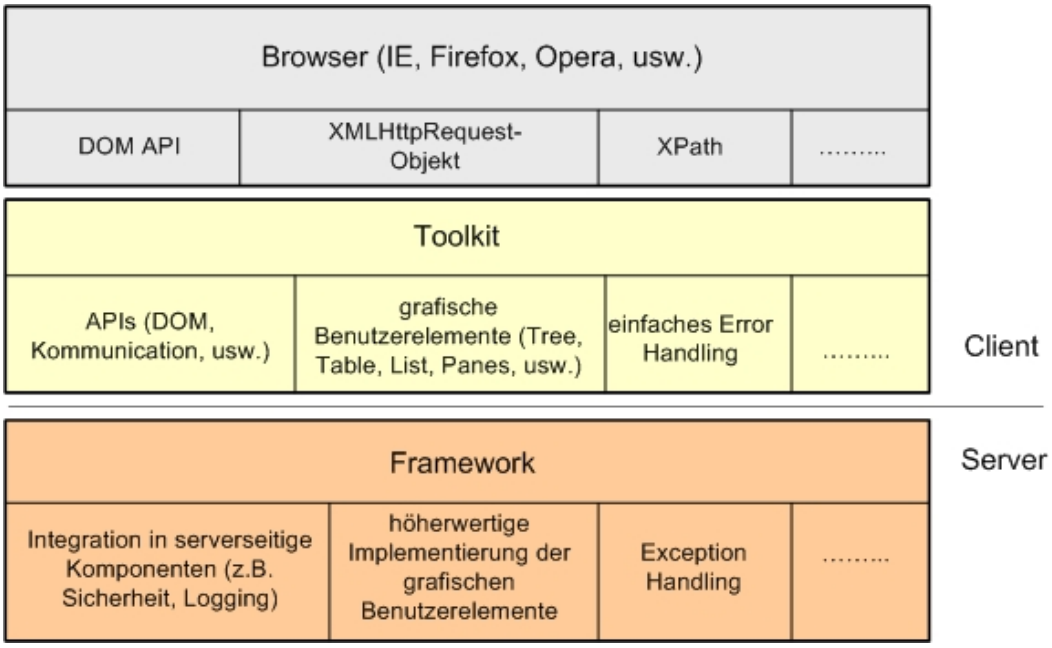

Abbildung 29: Beziehungen zwischen Browser APIs, Toolkits und Frameworks

Aus diesen Gründen wurden so genannte Toolkits entwickelt. Toolkits sind in JavaScript implementiert und kommen somit clientseitig zum Einsatz. Sie beseitigen die Inkompatibilitäten zwischen den Schnittstellen (z.B. DOM, XPath) der verschiedenen Browser-Implementierungen, durch die Bereitstellung von APIs. Des weiteren werden grafische Benutzerelemente bereitgestellt, die sich hinsichtlich ihres Funktionsumfangs nicht von den aus der Desktop-Programmierung bekannten grafischen Benutzerelementen unterscheiden. Durch die Bereitstellung eines einfachen

und durchgängigen Error-Handling, kann in der Anwendung leichter auf Fehler reagiert werden.

Der Einsatz von Toolkits vereinfacht die Anwendungsentwicklung erheblich, denn der Entwickler kann auf vielen vorgefertigten Komponenten aufbauen (z.B. APIs, grafische Benutzerelemente). Doch durch die Beteiligung einer Vielzahl an Technologien (z.B. JavaScript, CSS, HTML, usw.) ist die Anwendungsentwicklung sehr komplex. Deshalb wurden so genannte Frameworks entwickelt. Sie ermöglichen die Entwicklung von AJAX-Anwendungen in den verschiedensten serverseitigen Technologien (z.B. JSF, JSP, ASP) und sind in diese vollständig integriert. Außerdem stellen sie ein durchgängiges Exception-Handling und höherwertige Implementierungen der grafischen Benutzerelemente zur Verfügung.

Zusammenfassend lässt sich feststellen, dass Toolkits von den verschiedenen Browserimplementierungen abstrahieren und die entstehenden Probleme (z.B. Browserinkompatibilitäten) beheben, während Frameworks in serverseitige Technologien integriert sind und somit die Entwicklung von Anwendungen mit höherwertigen Programmierparadigmen ermöglichen. Auf eine nähere Betrachtung dieser Modelle wird an dieser Stelle verzichtet und auf Abschnitt 5.2.5 verwiesen.

5.2.3.1 Anwendungsarchitektur

Als durchgängiges Architekturmuster wird für die Strukturierung von Anwendungen, Toolkits und Frameworks das Model-View-Controller (MVC) Entwurfsmuster (siehe Angang C1) eingesetzt. Dieses Entwurfsmuster ist bereits aus der Programmierung von Desktop-Anwendungen und normalen Webanwendungen bekannt (siehe Anhang C2).

In AJAX-Anwendungen wird das MVC Entwurfsmuster auf verschiedenen Ebenen eingesetzt (siehe *Abbildung 30*), die nachfolgend vorgestellt werden:

- *Benutzerelement:* Jedes grafische Benutzerelement ist nach dem MVC Entwurfsmuster aufgebaut. So kann z.B. eine Baumstruktur folgendermaßen realisiert werden. Das Modell besteht aus den Knoten in einer Baumstruktur, die ihren Status verwalten und ihrerseits auf ein Objekt in der Verzeichnisstruktur verweisen. Der View besteht aus den Zeichen und Linien die dargestellt werden, während der Controller die Benutzereignisse verarbeitet und die Darstellung ständig aktualisiert.
- *Browser:* Im Browser besteht das Modell aus Business-Objekten, der View ist die im Browser angezeigt Webseite, die aus verschiedenen Benutzerelementen bestehen kann, und der Controller besteht aus den clientseitigen JavaScript Ereignis-Handlern, die mit dem Server kommunizieren.
- *AJAX-Anwendung:* Auf Anwendungsebene ist das MVC Entwurfsmuster ebenfalls präsent. Das Modell verwaltete sämtliche serverseitig gespeicherten Daten, während der View die verschiedenen serverseitig hinterlegten Seiten

darstellt und der Controller für die Behandlung von Anfragen an den Server zuständig ist.

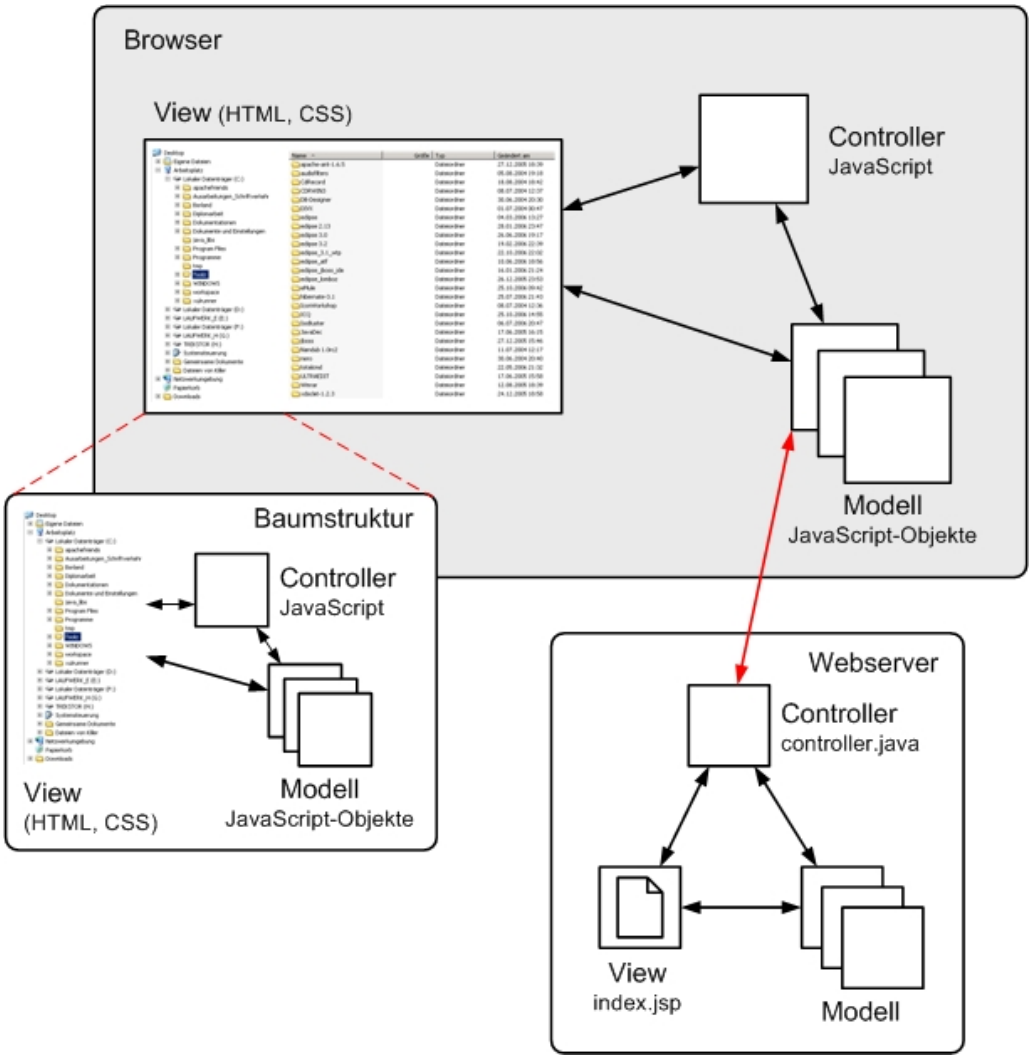

Abbildung 30: Anwendungsarchitektur von AJAX-Anwendungen

5.2.4 Standardisierung/Organisationen

In Abschnitt 5.1.2 wurden zahlreiche Probleme angesprochen, auf die bei der Programmierung von AJAX-Anwendungen zu achten ist. Diese Problematiken lassen sich durch den Einsatz von Toolkits bzw. Frameworks einschränken, aber nicht gänzlich vermeiden. Weitere Möglichkeiten diese Mankos zu beheben sind z.B. die Etablierung von Standards. Deshalb werden in diesem Abschnitt Organisationen vorgestellt, die sich in diesem Bereich engagieren.

5.2.4.1 World Wide Web Consortium

Das W3C ist ein Gremium zur Standardisierung der verschiedensten World Wide Web Technologien. Da es keine anerkannte Standardisierungsorganisation ist, wie z.B. ISO

([ISO]), werden lediglich De-facto-Standards verabschiedet an die sich die Browserhersteller halten können. So werden in Bezug auf AJAX alle wichtigen Technologien vom W3C spezifiziert. Spezifikationen die verabschiedet und stetig weiterentwickelt werden, sind z.B. DOM, XMLHttpRequest, HTML, XML, CSS bzw. diverse andere JavaScript Schnittstellen. Die in heutigen Browsern zum Teil noch sehr unterschiedlichen Implementierungen der Standards, wie in Abschnitt 5.1.2.2 bereits erwähnt wurde, sind für die Entwicklung von AJAX-Anwendungen sehr hinderlich, obwohl diese Problematiken durch Toolkits und Frameworks behoben wurden. Abschließend ist zu sagen, dass die vom W3C verabschiedeten Empfehlungen entscheidende Faktoren für die Verbreitung und Akzeptanz von AJAX sind.

5.2.4.2 OpenAJAX-Allicance

Unter dem Begriff OpenAJAX-Alliance haben sich zahlreiche namhafte IT-Unternehmen zusammengefunden, darunter z.B. IBM[25], Borland[26] oder SUN, um die Technologie AJAX populärer zu machen. Das größte Problem ist die fehlende Interoperabilität zwischen den verschieden sich auf dem Markt befindlichen Produkten, außerdem will man sicherstellen, dass sich AJAX an offenen Standards und der Open Source Technik orientiert. Dieses Ziel soll z.B. mit Best Practises erreicht werden. Als Standardisierungsorganisation sieht man sich allerdings nicht, aber eine enge Zusammenarbeit mit verschiedenen Standardisierungsorganisationen (z.B. W3C) bzw. Open Source Projekten soll erfolgen.

So entstand in Zusammenarbeit mit IBM und Eclipse das AJAX Toolkit Framework (ATF), um eine bessere Entwicklung von AJAX-Anwendungen zu ermöglichen. Es unterstützt diverse Toolkits und stellt verschiedene nützliche Hilfsmittel (z.B. DOM-Browser, JavaScript-Debugger, embedded Browser) bereit. Ein weiteres Projekt ist der OpenAJAX Hub, um die Interoperabilität zwischen verschiedenen Toolkits zu ermöglichen. Die erste Version ist frühestens 2007 verfügbar, aber damit die Idee funktioniert, müssen sich die Hersteller der Toolkits an bestimmte Schnittstellen halten und das kann zu Problemen führen, da es viele Toolkits auf dem Markt gibt, die sehr unterschiedliche Entwicklungsstände haben. Aus diesem Grund ist es fraglich, ob sich dieser Ansatz durchsetzen wird.

5.2.5 Frameworkmodelle

Wie bereits in Abschnitt 5.2.1 beschrieben wurde, existieren verschiedene Techno-logien, um RIAs (siehe Abschnitt 3.2.3) zu entwickeln. Da sich das Thema der Diplomarbeit auf AJAX beschränkt werden in diesem Abschnitt verschiedene Toolkits bzw. Frameworks aus diesem Bereich vorgestellt.

[25] http://www.ibm.com/de/
[26] http://www.borland.com/de/

Die verschiedenen Lösungsansätze können folgendermaßen kategorisiert werden:

- *Clientseitige Frameworks:* Sie werden ausschließlich clientseitig, d.h. im Browser, ausgeführt.
- *Serverseitige Frameworks:* Diese Frameworks kommen ausschließlich serverseitig zum Einsatz.
- *Multiplattform Frameworks:* Es wird versucht die verschiedensten Programmierparadigmen mit einem Framework abzudecken.

Diese verschiedenen Modelle werden nun hinsichtlich ihrer Architektur und der sich daraus ergebenden Vorteile bzw. Probleme ausführlich betrachtet.

5.2.5.1 Clientseitige Frameworks

Clientseitige AJAX Frameworks sind typischerweise in JavaScript implementiert und können auch als *Native Frameworks* bzw. Toolkits bezeichnet werden.

5.2.5.1.1 Architektur

Die Architektur von clientseitigen Frameworks wird in *Abbildung 31* dargestellt. AJAX-Anwendungen, die auf clientseitigen Frameworks basieren, bestehen aus einem client- und serverseitigen Bestandteil. Diese beiden Bestandteile sind vollständig voneinander entkoppelt und können unabhängig voneinander zur Verfügung gestellt werden. Aus diesem Grund können beliebige Formate für den Datenaustausch zwischen Client und Server eingesetzt werden.

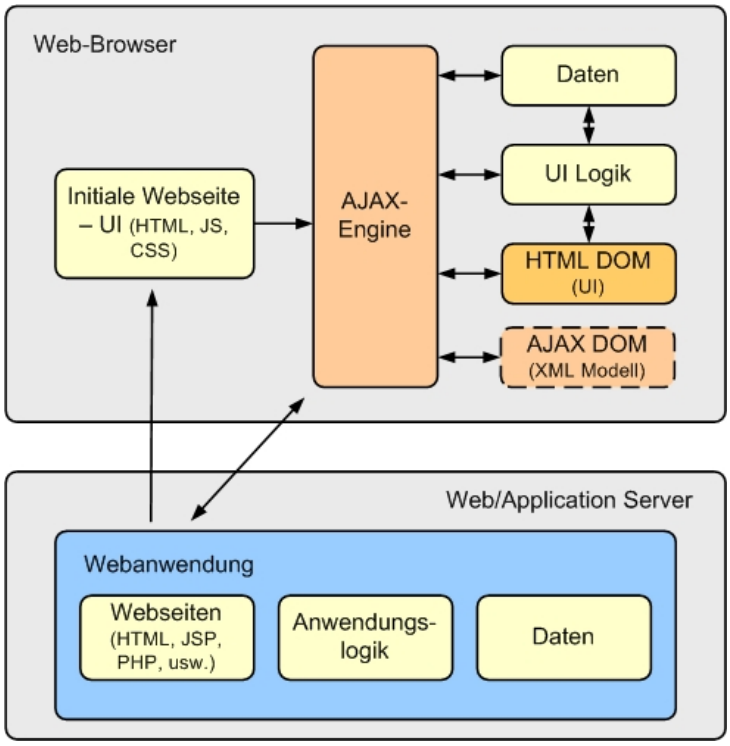

Abbildung 31: Architektur clientseitige AJAX-Frameworks

Die Anwendung wird über eine initiale Webseite aufgerufen und der Client (siehe Abschnitt 3.2.2.2) wird an den Browser übertragen. Dieser Client besteht im Fall von AJAX aus einem Mix von HTML, CSS und JavaScript. Der wichtigste Bestandteil ist die so genannte AJAX-Engine, die ein fester Bestandteil des Frameworks ist. Sie wird im Browser ausgeführt und wandelt die vom Server empfangenen Inhalte (HTML, XML, JavaScript, usw.) in browserverständliche Inhalte um. Der Browser kann diese Inhalte nun anzeigen.

Der Entwickler muss die Logik der Benutzerschnittstelle implementieren, die z.B. die Ereignisbehandlung enthält. Die Ereignisbehandlung nimmt die Benutzeraktionen entgegen und entscheidet ob sie zentral behandelt oder an den Server weitergereicht wird. Des weiteren implementiert er das Datenmanagement, dass für die Kommunikation zwischen Client und Server zuständig ist und die Benutzerschnittstelle aktualisiert, falls neue Inhalte angezeigt werden sollen.

Der serverseitige Teil der Anwendung ist eine Webanwendung und kann als ein Dienst betrachtet werden, der vom Client genutzt wird. Dieser Dienst kann in den verschiedensten serverseitigen Technologien z.B. JSP, ASP, PHP, usw. realisiert werden und beinhaltet seinerseits wiederum Anwendungslogik und Datenmanagement (z.B. SQL, Webservices).

Clientseitige Frameworks können auf zwei Ansätzen basieren, wie das HTML-Markup der initialen Webseite und das von Framework erzeugte HTML-Markup gespeichert wird. Das sind der Single- und Dual-DOM Ansatz, die nun vorgestellt werden.

5.2.5.1.1.1 Single-DOM

Beim Single-DOM Ansatz werden die Inhalte der initialen Webseite und die von der Transformationslogik der AJAX-Engine erzeugten Inhalte in einem DOM (HTML-DOM) gespeichert. Dieser Ansatz wird am Beispiel einer Baumstruktur (siehe *Abbildung 32*) näher betrachtet.

Auf der linken Seite ist das HTML-Markup der initialen Webseite und das von AJAX-Engine erzeugte HTML-Markup (rot markiert) zu sehen. Die resultierenden DOM-Objekte sind auf der rechten Seite zu sehen. Die nicht markierten Objekte repräsentieren die HTML-Tags der initialen Webseite (z.B.), während die markierten Objekte von der AJAX-Engine erzeugt wurden.

So werden typischerweise zusätzliche Elemente (z.B. , <img>, <p>) von der AJAX-Engine an die bereits vorhandenen Elemente der initialen Webseite angehängt. Diese zusätzlichen Elemente können z.B. die notwendigen Texte oder Grafiken beinhalten, die zum Erzeugen der Baumstruktur benötigt werden.

Auf der rechten Seite sind bestimmte Bereiche markiert, die private Daten des Frameworks, wie z.B. den Zustand der Anwendung oder Elemente repräsentieren.

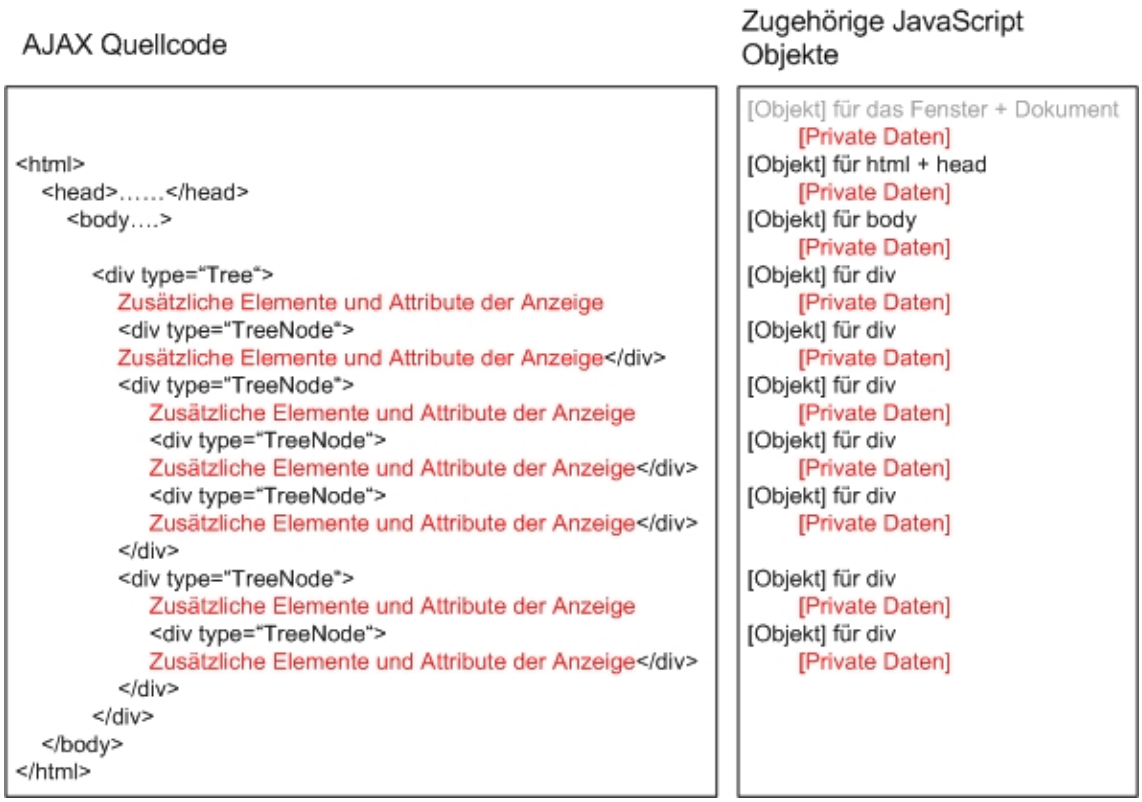

Abbildung 32: Clientseitige AJAX-Frameworks Single-DOM Ansatz

Dieser Ansatz ist besonders für Anwendungen geeignet, die um AJAX Funktionalitäten erweitert werden sollen, aber nicht die gesamte Internetpräsentation umstellen wollen.

5.2.5.1.1.2 Dual-DOM

Beim Dual-DOM Ansatz werden die Inhalte der initialen Webseite (HTML-DOM) und die von der Transformationslogik der AJAX-Engine erzeugten Inhalte (AJAX-DOM) jeweils in einem separaten DOM gespeichert. Dieser Ansatz wird am Beispiel einer Baumstruktur (siehe *Abbildung 33*) näher betrachtet.

Auf der linken Seite ist der Inhalt der initialen Webseite und getrennt davon die Beschreibung der Benutzerschnittstelle, der Komponente (Datei: „mytree.tpl“) zu sehen. Diese zusätzliche Datei ist über das Attribut (id=“mytree“) mit dem HTML-Markup der Webseite verknüpft. Beim Laden der initialen Webseite wird diese Datei mit Hilfe von JavaScript eingebunden und vom Framework internen XML-Parser ausgewertet.

Dieser Ansatz ermöglicht eine wesentlich bessere Strukturierung der Webseiten nach dem MVC-Muster. Zu beachten ist hier jedoch, dass der AJAX- und HTML-DOM über geeignete Mechanismen synchronisiert werden müssen, diese werden aber in der Regel vom Framework zur Verfügung gestellt.

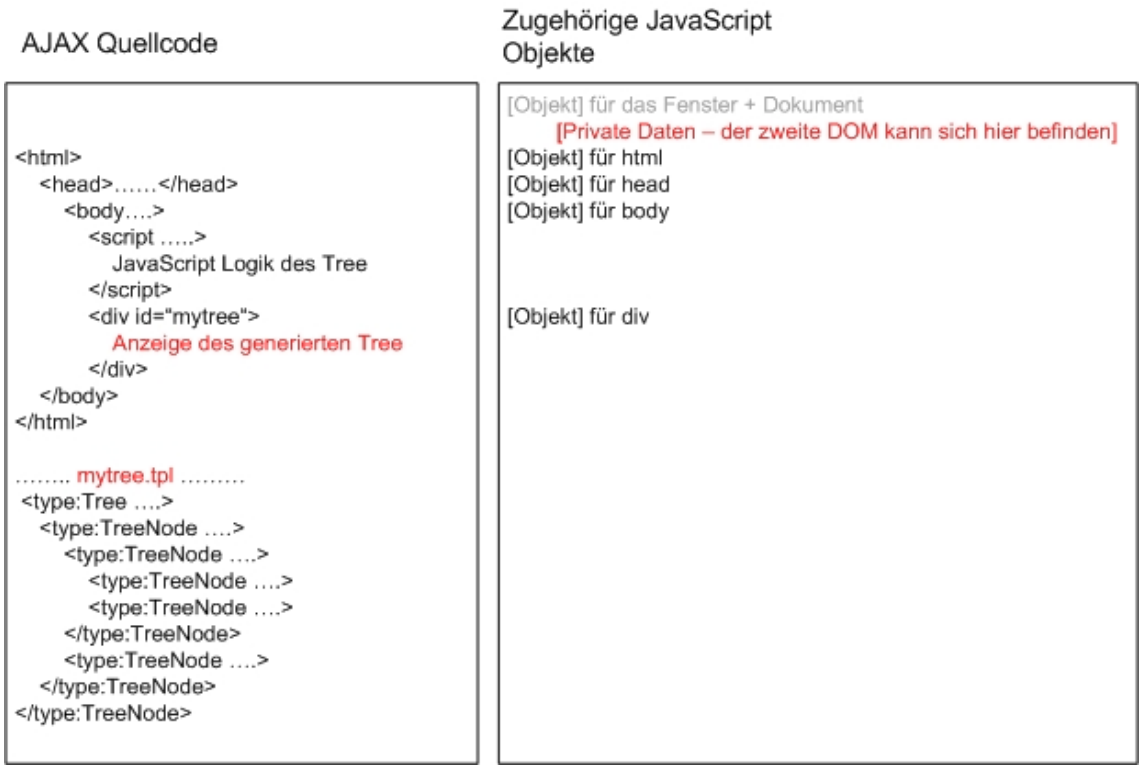

Abbildung 33: Clientseitiges AJAX-Framework Dual-DOM Ansatz

Der Dual-DOM Ansatz ist für Anwendungen geeignet, die durchgängig mit AJAX realisiert werden sollen.

5.2.5.1.2 Vorteile und Probleme

Durch den Einsatz von clientseitigen Frameworks ergeben sich diverse Vorteile bzw. Probleme, die nachfolgend aufgeschlüsselt und betrachtet werden.

Vorteile von clientseitigen Frameworks:

- Entkopplung von serverseitigen Technologien
- Entlastung der Webserver, durch die Verlagerung von Anwendungslogik auf den Client

Probleme durch clientseitige Frameworks:

- Sicherheit, mangelnde Integration in die Sicherheitsinfrastruktur
- steigende Komplexität, durch die Vielzahl an eingesetzten Technologien
- Entwicklung von Anwendungen

5.2.5.1.2.1 Entkopplung von serverseitigen Technologien

Client- und serverseitige Bestandteile sind von einander entkoppelt und der Zustand der Anwendung wird lediglich clientseitige verwaltet. Dadurch können die verschiedensten Dienste in der Anwendung genutzt werden. Diese Dienste können unterschiedliche Datenformate anbieten, die von der Anwendung verstanden werden

müssen. Somit ist es möglich z.B. die Google Maps oder Flickr in der Anwendung zu nutzen (Mashup).

5.2.5.1.2.2 Entlastung der Webserver

Beim Entwickeln von Anwendungen mit clientseitigen Frameworks kann die komplette Anwendungslogik auf den Client ausgelagert werden. Es findet also eine Entlastung der Webserver statt, denn der Client übernimmt Aufgaben wie z.B. die Auswertung der Daten und die Aktualisierung der Benutzeroberfläche.

In diesem Zusammenhang ist noch zu erwähnen, dass sich die Performance der Anwendung verschlechtern kann, wenn zu viel Anwendungslogik auf den Client verlagert wird.

5.2.5.1.2.3 Sicherheit

Durch die Entkopplung von client- und serverseitigen Bestandteilen ist eine Integration in die PAI Sicherheitsinfrastruktur nicht ohne weiteres möglich. An dieser Stelle soll jedoch auf diese Problematik nicht näher eingegangen werden, da sie in Kapitel 7 aufgegriffen wird.

5.2.5.1.2.4 Komplexität

Beim Entwickeln von AJAX-Anwendungen auf der Basis von clientseitigen Frameworks muss der Entwickler die Anwendungslogik, Datenmanagement und die Logik der asynchronen Anfragen selbst implementieren. Somit sind Kenntnisse wie z.B. JavaScript, HTML, CSS bzw. den eingesetzten serverseitigen Technologien notwendig. Durch die Anzahl der beteiligten Technologien wird die Anwendung sehr komplex.

5.2.5.1.2.5 Entwicklung von Anwendungen

Der Entwickler muss sein gewohntes Programmierumfeld verlassen und neue Programmiersprachen bzw. Programmierparadigmen erlernen. Es müssen neue Entwicklungswerkzeuge einsetzt werden, um z.B. Debugging zur Verfügung zu stellen. Das Logging und Error-Handling zwischen client- und serverseitigem Teil der Anwendung gestaltet sich, durch die Entkopplung der beiden Bestandteile, als schwierig. Außerdem wird der Aufwand für das Testen und Refactoring der Komponenten bzw. Anwendung durch die Vielzahl an Technologien enorm gesteigert.

5.2.5.2 Serverseitige Frameworks

Serverseitige Frameworks sind typischerweise in serverseitigen Programmiersprachen implementiert (z.B. Java, PHP) und können in zwei Gruppen unterteilt werden:

- *Serverside AddOn:* Diese Frameworks basieren auf etablierten serverseitigen Technologien (z.B. JSP, JSF, ASP) und erweitern ihre Funktionalität um AJAX Komponenten.

- *Integrated Component based:* Sie stellen dem Entwickler eine komponentenbasierte Architektur bereit. Das wird erreicht, indem die Java Servlet Technologie durch das Framework erweitert wird. Diese Frameworks orientieren sich typischerweise am Aufbau der Schnittstellen von bekannten grafischen Bibliotheken wie Swing, JFace oder SWT.

Die verschiedenen Ausprägungen serverseitiger Frameworks unterscheiden sich hinsichtlich der zugrunde liegenden Architektur und der daraus resultierenden Vorteile und Probleme nicht und werden deshalb nicht separat betrachtet.

5.2.5.2.1 Architektur

Die Architektur von serverseitigen Frameworks wird in *Abbildung 34* dargestellt. AJAX-Anwendungen die serverseitige Frameworks einsetzen werden ausschließlich serverseitig implementiert (z.B. Webseiten, Anwendungslogik, Datenmanagement). Die eingesetzten serverseitigen Komponenten generieren sämtliche Inhalte (HTML, XML, JavaScript, usw.).

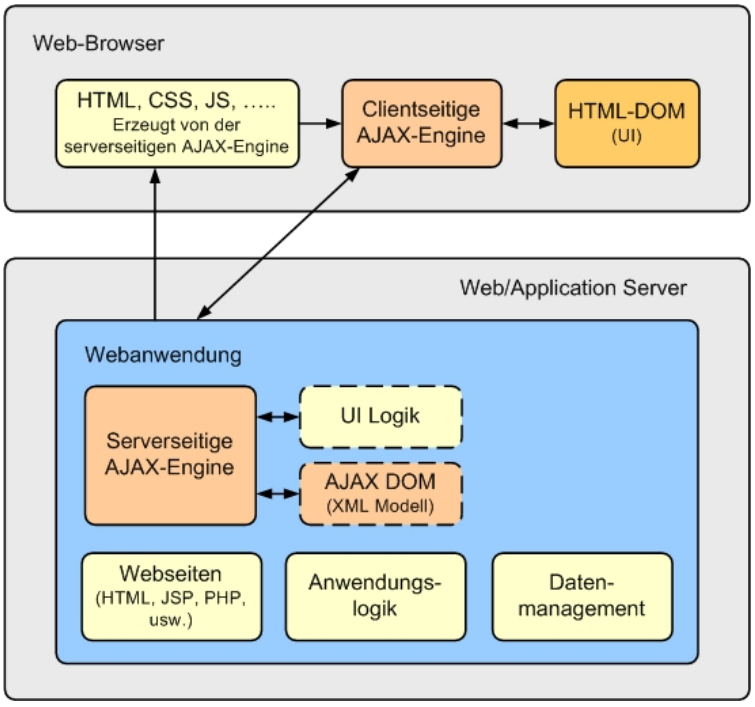

Abbildung 34: Architektur serverseitige AJAX-Frameworks

Diese Frameworks beinhalten ihre eigene clientseitige AJAX-Engine, die mit der serverseitigen AJAX-Engine über ein proprietäres Protokoll kommuniziert und ihren Zustand synchronisiert. Diese beiden Bestandteile sind voneinander abhängig und gehen ein Vertrauensverhältnis miteinander ein.

Es gibt verschiedene Strategien wie ein Framework implementiert sein kann. Auf der einen Seite können alle eingehenden Benutzeraktionen clientseitig, auf der anderen Seite teilweise client- oder serverseitig bearbeitet werden.

Serverseitige Frameworks können ebenfalls auf dem Single- bzw. Dual-DOM Ansatz basieren. Die client- bzw. serverseitigen Frameworks unterscheiden sich jedoch nur hinsichtlich des Dual-DOM Ansatzes. Aus diesem Grund kann der Single-DOM Ansatz in Abschnitt 5.2.5.1.1.1 nachgelesen werden und der Dual-DOM Ansatz wird im folgenden Abschnitt betrachtet.

Ein Beispielanwendung für den Einsatz von serverseitigen Frameworks ist in Abschnitt 7.3 zu finden.

5.2.5.2.1.1 Dual-DOM

Beim serverseitigen Dual-DOM Ansatz befindet sich der größte Teil der Anwendungslogik auf dem Server. Die clientseitige AJAX-Engine hat lediglich die Aufgabe auftretende Benutzerereignisse an den Server zu melden und die empfangenen Inhalte (HTML, Daten, CSS, JavaScript) aufzubereiten. Der Browser aktualisiert danach die Benutzerschnittstelle.

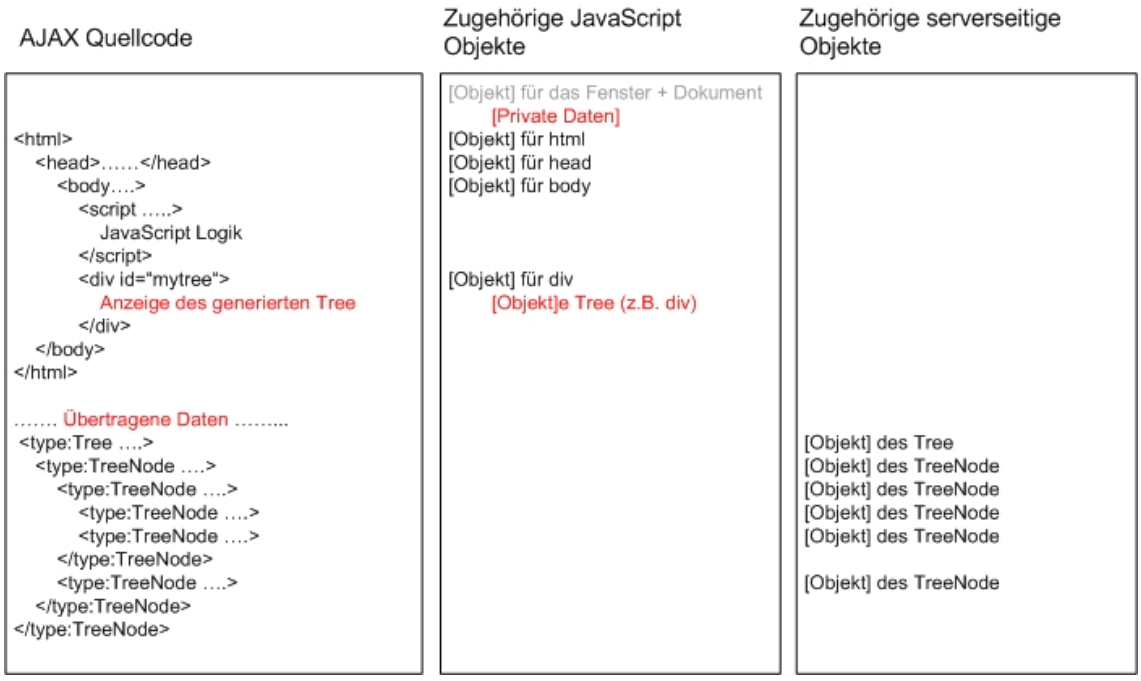

Abbildung 35: Serverseitiges AJAX-Framework Dual-DOM Ansatz

Der Server verwaltet den Zustand der AJAX-Anwendung, generiert die Beschreibung der Benutzerschnittstelle und bearbeitet die Benutzeraktionen. Dieser Ansatz wird am Beispiel einer Baumstruktur näher betrachtet.

Auf der linken Seite von *Abbildung 35* ist der Quellcode der initialen Webseite zu sehen. Die Abgleich zwischen client- und serverseitigen Komponenten erfolgt über das Attribut „id“.

Des weiteren ist ein Beispiel für die zwischen Client und Server ausgetauschten Daten zu sehen („Übertragene Daten“). Diese Daten repräsentieren eine generische Beschreibung der Baumstruktur und werden von der AJAX-Engine ausgewertet.

Die zugehörigen JavaScript Objekte sind in der Mitte der Abbildung zu sehen (siehe Abschnitt 5.2.5.1.1.2). Auf der rechten Seite sind die serverseitigen Objekte der Baumstruktur dargestellt. Wenn sich der Zustand der Anwendung (z.B. Benutzeraktion) ändert, werden diese Objekte aktualisiert und falls notwendig wird die Benutzerschnittstelle aktualisiert.

5.2.5.2.2 Vorteile und Probleme

Durch den Einsatz von serverseitigen Frameworks ergeben sich diverse Vorteile und Probleme, die nachfolgend aufgeschlüsselt und betrachtet werden sollen.

Vorteile von serverseitigen Frameworks:

- Integration in die Sicherheitsinfrastruktur
- Verbergen jegliche Komplexität vor dem Entwickler
- Entwicklung von Anwendungen

Probleme durch serverseitige Frameworks:

- Abhängigkeit der client- und serverseitigen Bestandteile
- Steigende Belastung der Webserver

5.2.5.2.2.1 Sicherheit

Die Frameworks sind komplett in serverseitige Technologien wie z.B. JSF, JSP integriert. Dadurch können alle vom Application-Server bereitgestellten Features genutzt werden (z.B. Sicherheitsmodell, Schnittstellen).

5.2.5.2.2.2 Komplexität

Das Framework versteckt jegliche Komplexität vor dem Entwickler. Der Entwickler muss sich nicht mehr mit clientseitiger Anwendungslogik, Datenmanagement und asynchroner Kommunikation beschäftigen, denn das Framework übernimmt all diese Dinge. So beinhalten die serverseitigen Komponenten die clientseitigen Bestandteile (z.B. JavaScript, HTML, CSS), die bei Bedarf übertragen werden. Falls aber neue Komponenten implementiert werden müssen, sind Kenntnisse in JavaScript notwendig, um die clientseitigen Bestandteile zu implementieren.

5.2.5.2.2.3 Entwicklung von Anwendungen

Der Entwickler kann in seinem gewohnten Programmierparadigmen (z.B. Java, JSP, JSF) Anwendungen erstellen und hat die Möglichkeit ein durchgängiges Exception-Handling bzw. Logging zu verwenden. Außerdem kann er seine gewohnte Entwicklungsumgebung einschließlich aller Features (z.B. Debugging, Refactoring) weiterhin nutzen.

5.2.5.2.2.4 Abhängigkeit

Der Zustand der Anwendung wird ausschließlich serverseitig verwaltet. Aus diesem Grund sind die client- und serverseitigen Bestandteile von einander abhängig. Falls in diesem Szenario zusätzliche Dienste (z.B. Flickr) genutzt werden sollen, ist das ausschließlich durch eine serverseitige Einbindung möglich.

5.2.5.2.2.5 Steigende Belastung der Webserver

Beim Entwickeln von Anwendungen mit diesen Frameworks befindet sich die vom Entwickler implementierte Anwendungslogik ausschließlich serverseitig. Lediglich die JavaScript Bestandteile der Komponenten werden clientseitig ausgeführt und können minimale Aufgaben ausführen. Es findet beim Einsatz somit eine Lastverschiebung in Richtung Webserver statt.

5.2.5.3 Multiplattform-Frameworks

In dem vorherigen Abschnitten wurden client- und serverseitige Frameworks detailliert vorgestellt. Multiplattform-Frameworks vereinen diese beiden Frameworktypen in einem Framework und versuchen möglichst viele Programmierparadigmen abzudecken. Die Architektur und die sich daraus ergebenden Vorteile und Probleme werden in diesem Abschnitt angesprochen.

5.2.5.3.1 Architektur

Multiplattform-Frameworks basieren auf einem clientseitigen Framework und stellen basierend auf diesem Framework weitere Implementierungen diverser serverseitiger Technologien zur Verfügung. Der Programmierer hat die Möglichkeit Anwendungen in JavaScript und anderen serverseitigen Technologien (z.B. PHP, JSF, JSP, ASP, usw.) zu entwickeln.

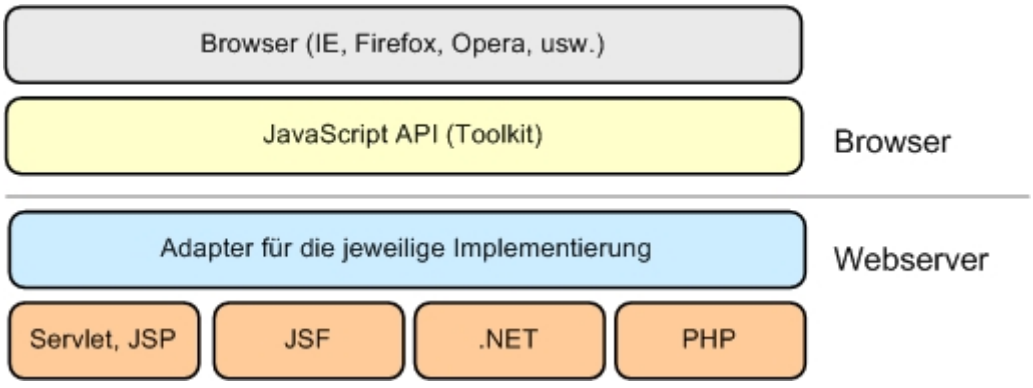

Abbildung 36: Architektur Multiplattform-Frameworks

Je nachdem, ob client- bzw. serverseitige Bestandteile des Frameworks eingesetzt werden, sind die in den vorherigen Abschnitten aufgeschlüsselten Vorteile und Probleme anzuwenden. In den folgenden Abschnitten werden lediglich die speziellen

Vorteile und Probleme angesprochen die durch den Einsatz von Multiplattform-Frameworks entstehen.

5.2.5.3.2 Vorteile

Der Entwickler hat die Möglichkeit mit einem Framework möglichst viele Technologien abzudecken und muss nicht für jede ein separates Framework einsetzten. Er kann für jede dieser Technologien die gleiche API verwenden. Dadurch wird die Integration in die Sicherheitsinfrastruktur erheblich vereinfacht, denn durch die Anpassung der JavaScript Implementierung kann dies an zentraler Stelle geschehen und alle abhängigen Implementierungen profitieren davon. Dieses Thema wird ausführlicher in Abschnitt 7.2.2.1.3 betrachtet.

5.2.5.3.3 Probleme

Der Support für die Bestandteile des Frameworks wird erheblich erschwert, denn falls sich die API des JavaScript Bestandteiles ändern, müssen alle davon abhängigen Implementierungen mühsam nachgezogen werden. Dies kann einen erheblichen Aufwand bedeuten, der je nach Umfang der Änderungen überproportional anwachsen kann.

5.2.6 Betrachtete AJAX Frameworks

Nachdem in den vorherigen Abschnitten die verschiedenen Frameworkmodelle beleuchtet wurden, werden in diesem Abschnitt verschiedene AJAX-Frameworks vorgestellt. Mit dem Hintergrund, sie auf ihre Eignung hin zu überprüfen und ein Framework auszuwählen, mit dem eine Beispielanwendung implementiert werden kann.

Außerdem wird ein Blick auf die Entwicklungen im .NET Bereich geworfen, um abschätzen zu können, in welche Richtung sich die Frameworks dort entwickeln.

5.2.6.1 Vorstellung der Frameworks

Die vorgestellten Frameworks wurden aus den folgenden Gründen ausgewählt:

- unterschiedliche Frameworkmodelle
- unterschiedliche Technologien (JavaScript, Java, JSF)
- an verschiedene Zielgruppen gerichtet
- Auswahl an kommerziellen und Open Source Produkten

5.2.6.1.1 Dojo

Eines der bekanntesten AJAX-Frameworks ist Dojo ([DOJO]), das von der Dojo Foundation einer gemeinnützigen Organisation weiterentwickelt wird. Dojo ist eine Vertreter der clientseitigen Frameworks (siehe Abschnitt 5.2.5.1) und hat deshalb den Vorteil mit jeder serverseitigen Technologie zu funktionieren.

Unterstützt wird die stetige Weiterentwicklung des Frameworks durch einige namhafte Firmen, darunter sind z.B. IBM und AOL[27]. Es steht unter der BSD Lizenz ([BSDLiz]) zur Verfügung und kann auf der Homepage herunter geladen werden.

Der Entwickler wird von vielen Aufgaben durch den hohen Abstraktionsgrad befreit und kann mit einem Mix aus JavaScript und Markup die zur Verfügung gestellten Komponenten nutzen. Bedingt durch die Vielfalt der eingesetzten Technologien ist eine gute Toolunterstützung empfehlenswert, die durch das ATF gegeben ist.

5.2.6.1.2 Yahoo User Interface

Das Yahoo User Interface (YUI) wird von Yahoo zur Verfügung gestellt und kann im Developer Bereich ([YUI]) herunter geladen werden. Der einzige Unterschied zu Dojo (siehe Abschnitt 5.2.6.1.1) besteht darin, dass YUI nicht so viele Komponenten bereitstellt.

5.2.6.1.3 Backbase

Backbase ([BACKBASE]) ist ein Multiplattform-Framework (siehe Abschnitt 5.2.5.3) und stellt verschiedene Versionen bereit. Der Entwickler kann sich zwischen der Community, Standart, Java und .NET Version entscheiden und muss je nach eingesetzter Version eine Lizenzgebühr bezahlen.

Die verschiedenen Versionen basieren auf dem gleichen Toolkit, werden aber sowohl client- bzw. serverseitig eingesetzt. Die Community und Standart Version sind clientseitige Frameworks, während die Java und .NET Version ausschließlich serverseitig eingesetzt werden.

Die Komponenten werden durch eine Proprietäre Markup-Language beschrieben und unterscheiden sich damit hinsichtlich der eingesetzten Version nicht. Die Java (JSF) und .NET (ASP) Version sind vollständig in die vorhandenen Entwicklungsumgebungen integriert und erleichtern dadurch die Entwicklung von Anwendungen enorm.

5.2.6.1.4 Google Webtools

Die Google Webtools (GWT) werden, wie der Name schon sagt, von Google ([GWT]) kostenlos zur Verfügung gestellt. Doch es unterscheidet sich enorm von den anderen Frameworks, denn die Entwicklung erfolgt unter Java und zum Testen wird ein spezieller Browser mit integriertem Debugging bereitgestellt. Aber beim Ausbreiten der Anwendung wird der Java Quellcode durch einen Compiler in eine Mischung aus JavaScript, HTML und XML umgewandelt.

Man kann also sagen, dass die Google Webtools sich beim Entwickeln und Testen als ein serverseitiges Framework darstellen, aber zur Laufzeit als ein clientseitiges Framework. Beim Entwickeln entstehen keinerlei Probleme falls der Quellcode

[27] http://www.aol.de/

analysiert (Debugging) werden muss, aber sobald die Anwendung kompiliert wurde sind die eventuell auftretenden Fehler nicht nachzuvollziehen und alles andere als transparent für den Entwickler.

5.2.6.1.5 Echo2

Das Echo2 Framework wird von der Firma Nextapp ([ECHO2]) unter der Mozialla Public Lizenz ([MozPuLiz]) zur Verfügung gestellt und ist ein serverseitiges Integrated Component based Framework (siehe Abschnitt 5.2.5.2). Es erweitert die J2EE Plattform um zusätzliche Funktionalität. Der Aufbau der API erinnert sehr stark an Swing und erleichtert somit die Entwicklung von Anwendungen erheblich. Nähere Informationen über das Echo2 Framework können dem Abschnitt 7.3.1 entnommen werden.

5.2.6.1.6 W4Toolkit

Das W4Toolkit ([W4T]) ist ein serverseitiges Integrated Component based Framework, das von der Firma INNOOPRACT zur Verfügung gestellt wird. Die API hat dieselbe Struktur wie SWT und JFace.

INNOOPRACT hat den Quellcode dem Rich AJAX Plattform (RAP: [RAP]) Projekt zur Verfügung gestellt, um eine stetige Weiterentwicklung und die bessere Integration in zukünftige Eclipse Projekte (Server side Equinox[28]) zu gewährleisten. Die Anlehnung an Eclipse Schnittstellen kann für die Anwendungsentwicklung in diesem Bereich einen erheblichen Schritt vorwärts bedeuten, denn RCP und RAP Anwendungen können generiert werden. Ein Problem ist, dass keine clientseitige AJAX-Engine vorhanden ist und alle Events an den Server übertragen werden. Der größte Vorteil von AJAX die asynchrone Kommunikation wird nicht genutzt.

5.2.6.2 .NET

Microsoft[29] stellt AJAX Funktionalitäten durch das Atlas Framework ([ATLAS]) zur Verfügung. Prinzipiell handelt es sich bei Atlas um ein JavaScript Toolkit, das clientseitig eingesetzt wird und serverseitige Komponenten aufruft. Zusätzlich werden serverseitige Komponenten zur Verfügung gestellt, die leicht in ASP 2.0 Anwendungen genutzt werden können.

Sie können mit spezielle ASP-Tags genutzt und bequem über das Visual Studio mit Event-Handlern verknüpft werden. Ein wirklich sehr interessanter Ansatz ist das so genannte UpdatePanel (siehe *Abbildung 37*). Es ermöglicht Standard ASP Komponenten um AJAX Funktionalitäten zu ergänzen. Die synchronen Komponenten werden in einem UpdatePanel platziert, wodurch diese automatisch asynchron aktualisiert werden. Dadurch können ASP 2.0 Anwendungen sehr einfach erweitert werden.

---

[28] http://www.eclipse.org/equinox/server/

[29] http://www.microsoft.com/de/de/default.aspx

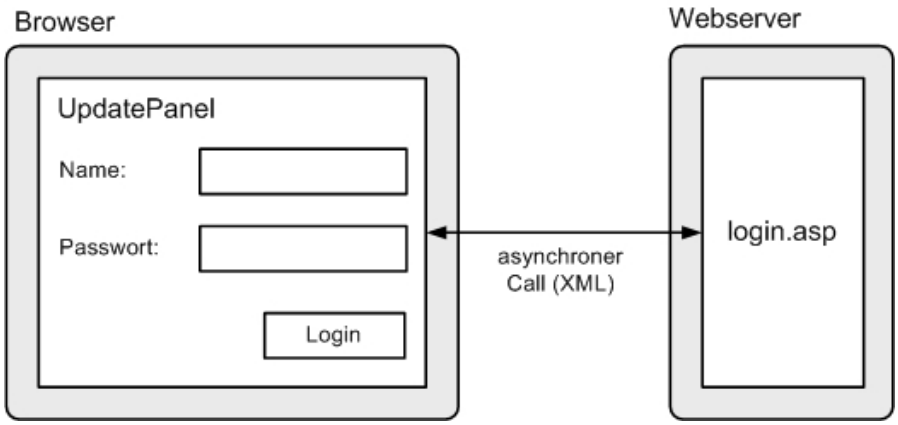

Abbildung 37: Prinzip des UpdatePanel des Atlas Frameworks

Ein Vorteil bei allen Microsoft Produkten besteht darin, dass eine exzellente Integration ins Visual Studio vorhanden ist.

5.2.7 Gegenüberstellung der Frameworks

Anhand eines Bewertungssystems wurden die Frameworks hinsichtlich ihr Eignung für PAI betrachtetet.

	Google	Echo2	W4Toolkit	Dojo	YUI	Backbase
Architektur	0	++	0	0	0	0
Technologien	+	+	+	0	0	-
Standardisierung	+	-	++	+	+	-
Entwicklung	+	+	++	0	0	-
Produktsupport	+	-	0	0	+	-
Tooling	+	++	++	0	0	+
Kenntnisse des Entwicklers	+	+	+	-	-	-
Erweiterbarkeit	+	+	0	+	0	-
Look&Feel	0	+	+	0	0	0
Pricing Toolkit	+	+	-	+	+	--
Pricing IDE	+	-	0	+	+	0

Tabelle 1: Gegenüberstellung der betrachteten AJAX-Frameworks

Die folgende Tabelle dient lediglich der Übersichtlichkeit und soll helfen festzustellen in welchen Rubriken die Frameworks Stärken und Schwächen haben. Es soll jedoch nicht darauf eingegangen werden, warum die Frameworks in den Punkten so bewertet wurden, dafür steht ein externes Dokument ([FRAMEWORKS]) zur Verfügung.

5.2.8 Empfehlungen

In diesem Abschnitt soll eine Empfehlung ausgesprochen werden, für welche Anwendungsszenarien die verschiedenen Frameworkmodelle am besten geeignet sind.

Clientseitige Frameworks:

- *Native Frameworks* können am sinnvollsten zur Implementierung von privaten Webseiten, als Basiskomponente für Frameworks oder zur Programmierung von grafischen Benutzerelementen eingesetzt werden, denn sie können nicht ohne die Entwicklung von zusätzlichen Komponenten in die J2EE Sicherheitsinfrastruktur integriert werden. In Abschnitt 7.2 wird eine Möglichkeit vorgestellt, wie das geschehen kann.

Serverseitige Frameworks:

- *Serverside AddOns* haben ihren größten Nutzen darin, indem sie für die Erweiterung von Webanwendungen um AJAX Funktionalitäten eingesetzt werden, die auf der gleichen serverseitigen Technologie (z.B. JSF, JSP, ASP) basieren. Außerdem können sie für die Entwicklung neuer Anwendungen eingesetzt werden, weil sie in die J2EE Sicherheitsinfrastruktur integriert sind.
- *Integrated Component based* können am sinnvollsten zur Implementierung von Business Anwendungen eingesetzt werden, weil sie in die J2EE Sicherheitsinfrastruktur integriert sind.

Multiplattform Frameworks:

- Da Multiplattform Frameworks sowohl client- und serverseitig eingesetzt werden können, können sie in den jeweiligen zuvor angesprochenen Einsatzgebieten verwendet werden.

Durch den Einsatz der verschiedenen Frameworks entstehen jedoch Probleme, die im Abschnitt 7.1 beleuchtet werden.

Die von PAI gestellten Anforderungen (siehe Abschnitt 5.2.2) an die Frameworks, können nur von Backbase, Echo2 und W4T erfüllt werden. Aus diesen drei Frameworks wurde Echo2 zur Implementierung einer Beispielanwendung ausgewählt, weil es die Vorteile von AJAX einsetzt und interessante Möglichkeiten bietet, komponentenbasierte Webanwendungen zu implementieren. Es macht aber durchaus Sinn Backbase und W4T zu testen.

6 AJAX und Sicherheit

Die Sicherheit spielt bei Webanwendungen und Diensten eine ständig wachsende Rolle, denn das Internet ist ein unsicheres Medium und geeignete Sicherheitsmaßnahmen sind ein wichtiges Qualitätskriterium von Anwendungen. An dieser Stelle kann keine umfassende Einführung in das Thema stattfinden, deshalb werden lediglich sicherheitsrelevante Probleme vorgestellt, die sich besonders auf AJAX-Anwendungen auswirken. Dabei werden folgende Punkte betrachtet:

- Maßnahmen der Browserhersteller, um den über das Netzwerk übertragenen Quellcode in einer sicheren Umgebung ablaufen zu lassen
- Vorstellung verschiedene Angriffsmethoden und deren Auswirkungen
- Möglichkeiten zum Absichern von AJAX-Anwendungen

6.1 Browsersicherheit

Beim Ausführen von AJAX-Anwendungen wird der Clientcode vom Webserver an den Browser übertragen. Der Benutzer demonstriert mit der Ausführung der AJAX-Anwendung im Browser gegenüber dem Clientcode und dessen Autor großes Vertrauen.

Von Seite der Browserhersteller und Standardisierungsorganisationen wurden Maßnahmen getroffen, die einen Missbrauch dieses Vertrauens verhindern sollen. Bevor auf diese Maßnahmen eingegangen wird, sollen die Sicherheitsmodelle des Internet Explorers (IE: [IE]) und Firefox ([FIREFOX]) betrachtet werden.

6.1.1 Browser Sicherheitsmodelle

Der IE (Version 6 und 7) setzt ein Sicherheitsmodell ein, das auf so genannten Zonen basiert. Die Einschränkungen, die in ihnen gelten, können frei konfiguriert werden (siehe Anhang B2). So haben z.B. standardmäßig Dateien, die sich im lokalen Dateisystem befinden, die Möglichkeit Webseiten zu kontaktieren ohne den Benutzer um Erlaubnis zu fragen, da das Dateisystem als sichere Zone gilt. Falls der gleiche Code in einen Webserver ausgeführt wird, löst er eine Sicherheitswarnung aus, die dem Benutzer als Dialog präsentiert wird. Dort kann er dann entscheiden, ob die Aktion genehmigt wird oder nicht. Die vom Benutzer getroffene Auswahl wird im Browser gespeichert und beim erneuten Zugriff wieder verwendet.

Das Sicherheitsmodell vom Firefox verhält sich etwas anders und basiert auf so genannten Privilegien. Einige grundlegende Aktivitäten z.B. das Schreiben und Lesen von Dateien, Verbindung zu einem anderen Server aufbauen, außer dem Ursprungsserver, werden als unsicher angesehen. Falls im Code eine dieser Aktionen durchgeführt werden soll, wird zuerst beim PrivilegeManger die Erlaubnis zur Ausführung

angefordert. Er kann jedoch so eingestellt werden, dass den Anforderungen des Codes keinerlei Beachtung geschenkt wird. Dem Benutzer wird ein Dialog präsentiert, um sein Einverständnis für die folgende Aktion zu geben. Im Gegensatz zum IE erteilt der Firefox das Privileg nur für die Dauer der Funktion, die es angefordert hat.

Die Sicherheitsmodelle des IE und Firefox sind grundverschieden und erschweren erheblich die Implementierung einer Anwendung, denn beim Testen müssen diese unterschiedlichen Modelle berücksichtigt werden. Es besteht die Möglichkeit eine AJAX-Anwendung, mit auf dem Dateisystem zur Verfügung stehenden Dummydaten, zu testen. Hier kann es allerdings mit den verschiedenen Sicherheitsmodellen der Browser Probleme geben, wenn man die Anwendung auf dem Webserver (andere Zone) testet.

6.1.2 Richtlinie des „Ursprungsservers“

JavaScript wird im Browser in einer Sandbox ausgeführt. Dadurch wird kontrolliert, dass die Anwendung nur Zugriff auf die vom Browser zur Verfügung gestellten Java-Script-Objekte hat, nicht lesend bzw. schreibend auf das Dateisystem zugreifen und nicht mit anderen Ressourcen außer dem eigenen Ursprungsserver (siehe *Abbildung 38*) kommunizieren kann.

Es gibt in HTML Tags (z.B. iFrame), denen es erlaubt ist Webseiten einer anderen Domäne zu laden und den JavaScript-Code auszuführen. Die Scripte der verschiedenen Domänen können jedoch nicht miteinander interagieren. Wenn sich die beiden Scripte auf derselben Domain befinden, können sie auf die jeweiligen Ressourcen (Variablen, Funktionen) des Anderen zugreifen und diese nutzen. Falls sich beide Scripte auf unterschiedlichen Domänen befinden, wird ein JavaScript-Fehler angezeigt.

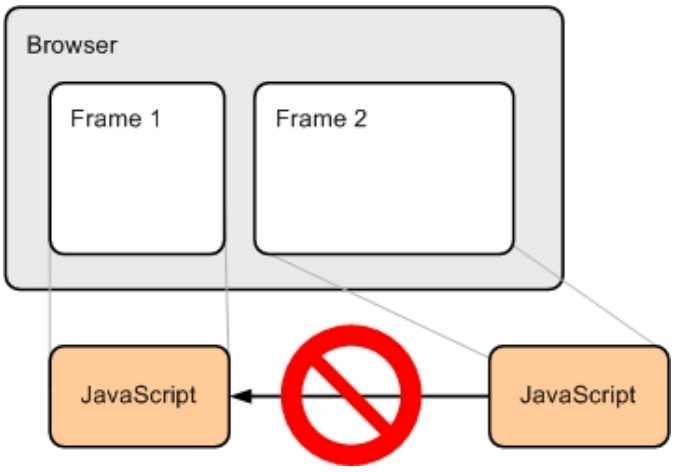

Abbildung 38: Richtlinie des „Ursprungsservers“

Ein weiteres Problem in diesem Zusammenhang ist die Tatsache, dass der Browser ein stark eingeschränktes Verständnis davon hat, was sich auf der gleichen Domain

befindet. Um eine Domain zu identifizieren, wird lediglich der Anfang der URL betrachtet. Es wird nicht überprüft, ob zwei Domänen die gleiche IP-Adresse zu Grunde liegt. In *Tabelle 2* soll mit Hilfe von detaillierten Beispielen versucht werden das Verständnis dafür zu wecken, was ein Browser unter einer Domain versteht und wie er sie identifiziert.

URLs	erlaubt?	Beschreibung
http://www.example.de/page1.html http://www.example.de/page2.html	ja	Zugriff erfolgreich.
http://www.example.de:8080/page1.html http://www.example.de/page2.html	nein	Der Port stimmt nicht überein.
http://www.example.de/page1.html https://www.example.de/page2.html	nein	Die Protokolle stimmen nicht überein.
http://www.example.de/page1.html http://160.14,15.21/page2.html	nein	Die erste URL hat zwar dieselbe IP, der Browser überprüft das aber nicht.
http://www.example.de/page1.html http://pages.example.de/page2.html	nein	Subdomänen werden als eigenständige Domänen behandelt.
http://www.example.de/hg/page1.html http://www.example.de/zg/page2.html	ja	Nur die Domäne wird überprüft.

Tabelle 2: Browser Domänenverständnis

Würde der Benutzer die Ausführung von Code erlauben, der von unterschiedlichen Domänen stammt, entstünde eine sehr große Angriffsfläche für XSS (siehe Abschnitt 6.2.2.1) Angriffe. Dadurch kann z.B. der DOM der aktuellen Webseite manipuliert werden. Das Sicherheitsmodell soll verhindern, dass solche Angriffe durchgeführt werden können. Es verhindert außerdem, dass der Code von einer anderen Webseite geladen und genutzt werden kann. Die Sicherheitsmodelle der Browser sind nicht 100% sicher und es existieren diverse Angriffe, die in Abschnitt 6.2 vorgestellt werden, um die Sicherheitsvorkehrungen zu umgehen. Falls die Anwendung Daten bzw. Dienste (z.B. Google Maps, Flicker) einer anderen Domäne nutzen will, führt das zu Problemen, da sie sich auf verschiedenen Servern befinden.

Es gibt verschiedene Möglichkeiten dieses Problem zu lösen, zwei Varianten sollen in den nachfolgenden Abschnitten angesprochen werden.

6.1.2.1 Nutzung des PrivilegeManagers

Auf dem Client werden asynchrone Aufrufe (siehe Abschnitt 2.2.1.3) mittels des im Browser verfügbaren XMLHttpRequest-Objektes (siehe Abschnitt 2.2.2.2.3) realisiert.

Für dieses Objekt gelten dieselben Einschränkungen wie für den Browser. Der Zugriff auf Daten, die nicht vom Ursprungsserver stammen ist verboten.

Es besteht die Möglichkeit, diese Einschränkung mit Hilfe von JavaScript zu umgehen. Das ist ein großes Problem, denn es bestünde nun die Möglichkeit, Dateien von fremden Servern aufzurufen. Damit wird es potentiellen Angreifern leichter gemacht schädlichen Code in das System zu schmuggeln und auszuführen.

Die Realisierungen unterscheiden sich hinsichtlich der verschiedenen Sicherheitsmodelle (siehe Abschnitt 6.1.1), die im IE und Firefox implementiert sind. Wie in diesem Abschnitt beschrieben, wird im IE ein Dialog geöffnet der dem Benutzer eine Entscheidung abverlangt. Die Abfrage kann mit ja oder nein beantwortet werden und die Aktion wird entweder ausgeführt oder ein JavaScript-Fehler tritt auf.

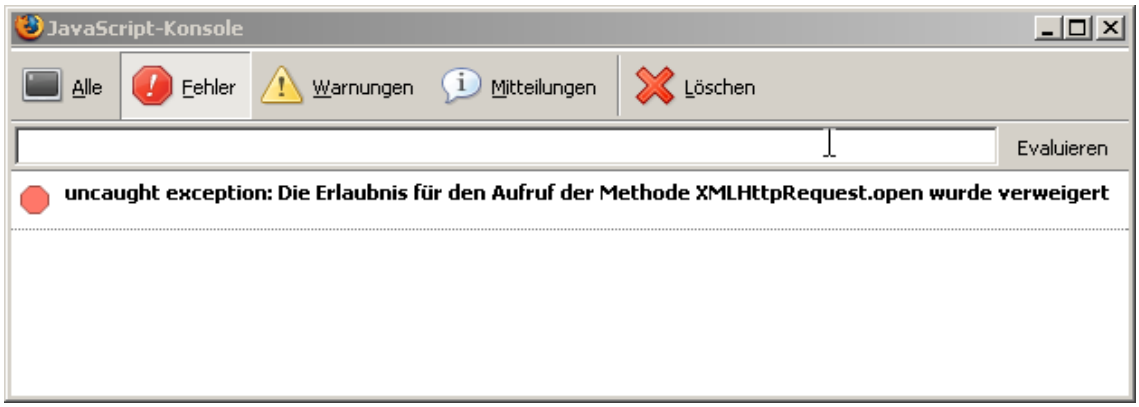

Abbildung 39: Firefox JavaScript-Konsole

Im Mozilla werden Warnungen und Fehler nicht in einer Dialogbox auf dem Bildschirm angezeigt, sondern in der JavaScript-Konsole ausgegeben. Dieses Vorgehen ist nicht sehr transparent und kann zu Missverständnissen auf Benutzerseite führen.

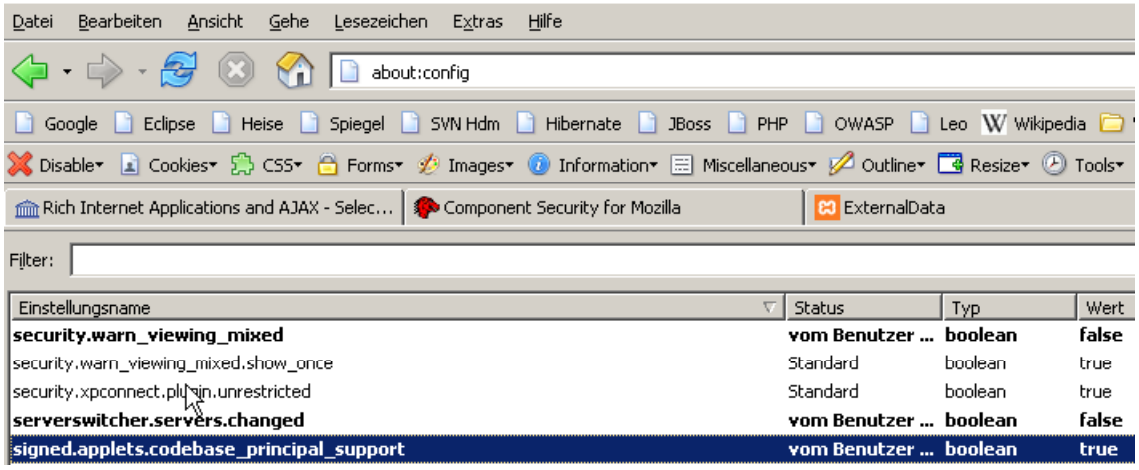

Abbildung 40: Firefox Sichheitseinstellungen

Der Benutzer kann den Zugriff auf externe Quellen (Server) per Hand erlauben, das ist jedoch für AJAX-Anwendungen weniger von Interesse, denn jeder Benutzer müsste seine Einstellungen anpassen.

Um den Zugriff zu erlauben, muss in die URL „about:config“ eingegeben werden. Der in *Abbildung 40* dargestellte Dialog erscheint und die im Bild selektierte Option wird auf true gesetzt. Nun können externe Datenquellen aufgerufen werden.

Es gibt die Möglichkeit mit Hilfe des PrivilegeManager und mittels JavaScript die nötigen Privilegien zu setzen. Diese müssen jedoch für jede kritische Aktion (z.B. open, send des XMLHttpRequest-Objektes) erneut gesetzt werden.

```
window.onload = function()  {
   try  {
      if (netscape.security.PrivilegeManager.enablePrivilege)  {
         netscape.security.PrivilegeManager.enablePrivilege('UniversalBrowserRead');
      }
   }
   catch (err) { }

   var req = new ActiveXObject("Microsoft.XMLHTTP");
   req.open("GET","http://www.spiegel.de/schlagzeilen/rss/0,5291,24,00.xml",true);
   req.onreadystatechange = function()    {  …… }
   req.send(null);
}
```

Abbildung 41: Beispiel – Umgehen der Ursprungsserver Richtlinie mit dem PrivilegeManager

Wenn eine nicht sichere Aktion ausgeführt wird, erscheint im Mozilla eine verzögerte Dialogbox mit einer Warnung. Durch das Selektieren der Checkbox (siehe *Abbildung 42*) wird die Einstellung permanent gespeichert. Nach dem Bestätigen dieser Dialogbox wird das JavaScript ausgeführt und kann auf externe Daten zugreifen.

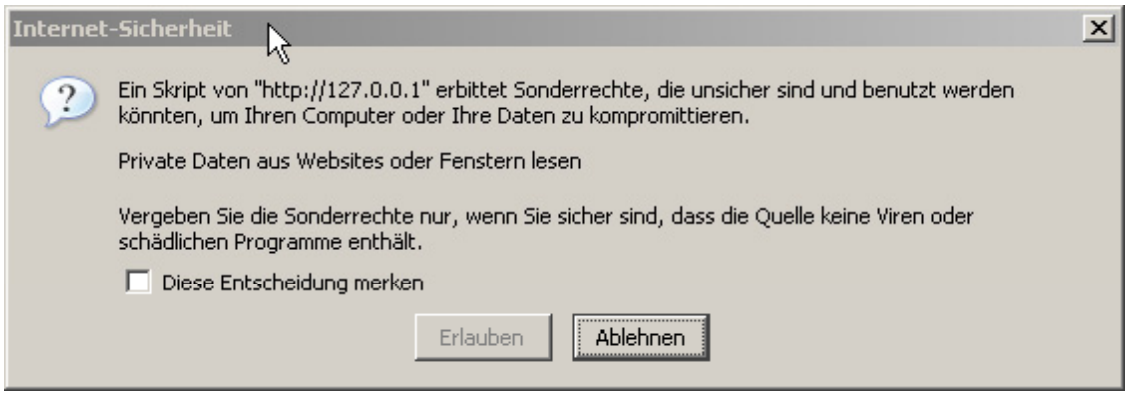

Abbildung 42: Firefox Sicherheitsdialog

Diese Variante scheint auf den ersten Blick sehr komfortabel und flexibel zu sein, aber die Möglichkeit mit Hilfe von JavaScript die Sicherheitseinstellungen des Browsers zu umgehen ist sehr bedenklich. Damit bietet sich Angreifern die Möglichkeit einen Dienst aufzurufen bzw. zu nutzen, der sich auf einer anderen Domäne befindet. Dieser aufgerufene Dienst kann bösartigen Quellcode enthalten und den Browser bzw. das Betriebssystem infizieren. Der Benutzer bekommt zwar einen Dialog mit einer Warnung präsentiert, aber die Chance dass ein durchschnittlicher Internetnutzer diese Warnung liest bzw. versteht ist sehr gering. Die meisten Benutzer würden ohne groß darüber nachzudenken einfach auf ja klicken.

6.1.2.2 Proxys für Remotedienste

Bei diesem Ansatz wird die oben genannte Beschränkung umgangen, indem der Zugriff auf den Dienst über einen Proxy (z.B. Servlet) realisiert wird, der sich auf dem Ursprungsserver befindet. Dem Client wird vorgespiegelt, dass die Daten vom Ursprungsserver stammen.

Der Vorteil dieser Lösung besteht darin, dass der Proxy noch diverse Arbeiten auf den Daten, wie z.B. Filtern nach böswilligen Daten und Code, vornehmen kann und somit wertvolle Vorarbeit leistet. Aber durch die Bereitstellung von Webservices sollten Server entlastet werden, dieser Vorteil wandelt sich durch diese Architektur in einen Nachteil um, denn der Server muss sämtliche Anfragen die von der Webanwendung ausgelöst werden bearbeiten.

Dieser Ansatz sollte nicht mit der simplen Idee von Mashups verwechselt werden.

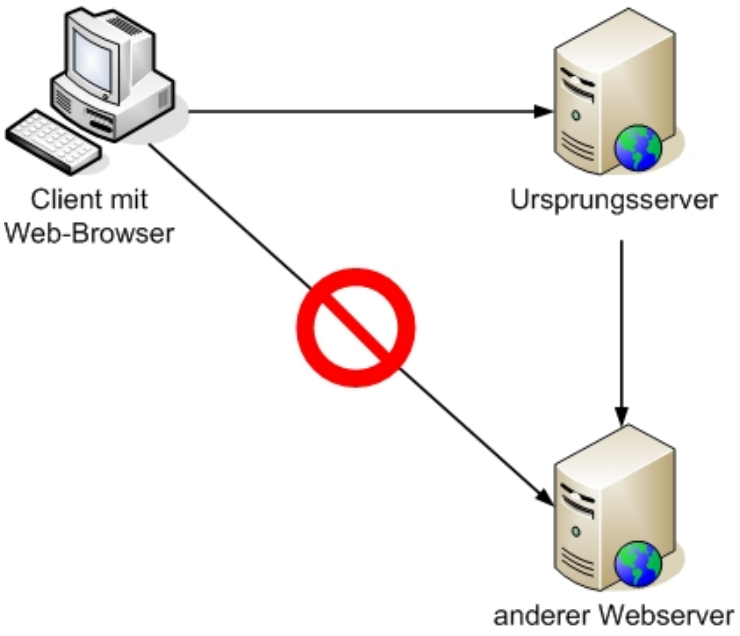

Abbildung 43: Anwendungsszenario Proxys für Remote Dienste

6.1.3 Zugriff auf Daten des Dateisystems

Wie bereits bekannt ist, hat JavaScript Code keinen Zugriff auf Dateien, die sich im lokalen Dateisystem des Benutzers befinden. Das wird erreicht, indem JavaScript keinerlei Funktionen bereitstellt, die die Bearbeitung von Dateien ermöglichen und der Code aus Sicherheitsgründen in einer Sandbox ausgeführt wird.

Diese Beschränkungen sind sehr sinnvoll, denn wer möchte, dass ein Script von einem unbekannten Server und Autor den Zugriff auf Ressourcen der Festplatte erhält. Im IE ist es möglich diesen Mechanismus durch den Einsatz einer ActiveX Komponente zu umgehen. Dadurch werden Dinge wie das Einblenden bzw. Ausblenden der Statusleiste, der Zugriff auf Dateien oder das Auslesen der zuletzt besuchten Webseite ohne die Zustimmung des Benutzers möglich. Die einzige Möglichkeit wie der Benutzer sich vor solchen Angriffen schützen kann, ist die Deaktivierung von ActiveX in den Browsereinstellungen.

6.1.4 Benutzerschnittstelle als Sicherheitsrisiko

In diesem Abschnitt soll auf ein Problem mit der Benutzerschnittstelle des Browsers hingewiesen werden, dass den Benutzer sehr verwirren kann und böswilligem Code reichhaltige Möglichkeiten bietet, ihn mit Hilfe von JavaScript auszuspionieren.

Beim Ausführen von synchronen Aufrufen (z.B. Aufruf der Spiegel Webseite) wird der aktive Vorgang, sei es nun im Firefox oder IE, in der Statusleiste des Browsers angezeigt (siehe *Abbildung 44*). Der Benutzer kann sich über die Fortschritte informieren und gegebenenfalls den Vorgang abbrechen. Kurz gesagt er hat Kontrolle über den Prozess. Wenn aber eine asynchrone Abfrage mittels des XMLHttpRequest-Objektes stattfindet, erhält der Benutzer keinerlei Informationen über die im Hintergrund laufenden Vorgänge (siehe Abschnitt 5.1.1.2). In diesem Fall wird die Statusleiste nicht aktualisiert und enthält weiterhin den Status „Fertig“. Die Verantwortung um solche Aktionen dem Benutzer anzuzeigen, wird in diesem Fall komplett dem Entwickler übertragen.

Das dem Entwickler entgegengebrachte Vertrauen kann ausgenutzt werden, wie das folgende Beispiel zeigt. Ein Formular zum Ausfüllen eines Versicherungsfalles kann mit JavaScript so erstellt werden, dass beim Betätigen der Löschtaste der Inhalt im Hintergrund in einer Datenbank gespeichert wird und auf diese Weise alle Änderungen am Formular nachvollzogen werden können. Dadurch wird es der Versicherung möglich Informationen zu erhalten, die z.B. auf einen Betrug hinweisen. Der Benutzer bekommt davon nichts mit, weil der Browser keine Aktivitäten anzeigt. Die einzige Möglichkeit diesen Mechanismus zu erkennen ist mit Hilfe eines Proxy den Transportkanal zu überwachen oder eine komplette Analyse des Quellcodes durchzuführen.

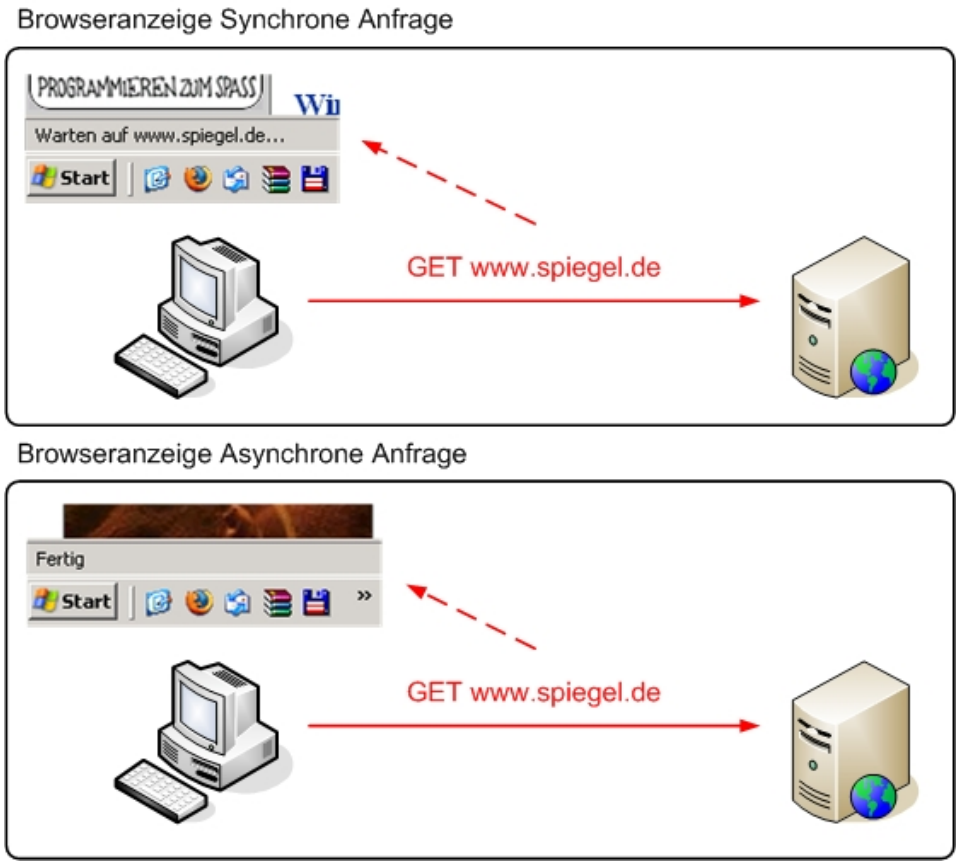

Abbildung 44: Benutzerschnittstelle als Sicherheistrisiko

Die Möglichkeiten, die sich Angreifern oder Webseiten Betreibern bieten, sind nicht zu unterschätzen und sehr bedenklich, denn Webseiten die manipuliert sind, können nur sehr schwer erkannte werden. Abhilfe kann z.B. durch einen „Task-Manager“ geschaffen werden, der eine Übersicht der asynchronen Aufrufe anzeigt.

6.2 Angriffe auf AJAX-Anwendungen

Durch die wachsende Beliebtheit von AJAX werden die entstehenden Sicherheitsprobleme und ihre Lösung immer wichtiger. Dem Symantec Internet Security Thread Report[30] ist zu entnehmen, dass Webanwendungen einem potentiell hohen Risiko ausgesetzt sind angegriffen zu werden. Jeder ist gefährdet und kann Opfer eines Angriffes werden, wie *Abbildung 45* verdeutlicht.

Am 4. September 2006 wurde z.B. Angelika Merkel Opfer eines Angriffes. Auf vier bekannten Webseiten (Bundesregierung[31], Spiegel[32], Financial Times Deutschland[33],

[30] http://www.symantec.com/region/de/PressCenter/Threat_Reports.html
[31] http://www.bundesregierung.de/
[32] http://www.spiegel.de/
[33] http://www.ftd.de/

Stern[34]) wurde ihr Rücktritt bekannt gegeben. Was auf den ersten Blick sehr erheiternd wirkt, kann für den Betroffenen einen enormen Schaden bedeuten.

Abbildung 45: Angriff auf die Webseite der Bundesregierung[35]

Aus diesem Grund sollen die Schwachstellen von JavaScript und verbreitete Angriffe auf Webanwendungen prinzipiell vorgestellt werden. Woran liegt es aber das immer mehr Angriffe auf Web- bzw. AJAX-Anwendungen durchgeführt werden? Die folgenden Punkte sind Gründe dafür:

- Verlagerung der Anwendungslogik auf den Client
- wachsendes JavaScript Know-How
- neue Angriffsvektoren durch Frameworks, RSS, REST bzw. JSON

6.2.1 JavaScript Schwachstellen

JavaScript (siehe Abschnitt 2.2.2.2.1) wird im Gegensatz zu anderen Programmier-sprachen nicht serverseitig, sondern clientseitig ausgeführt. Dabei wird der im Klartext übertragene JavaScript- Code durch einen in den Browser eingebauten Interpreter zur Laufzeit der Anwendung ausgewertet. Ein potentieller Angreifer hätte schon auf dem Transportweg die Möglichkeit bösartigen Quellcode einzuschleusen und niemand würde es merken. Dieser Umstand bedeutet schon vor der Ausführung einen voll-ständigen Vertrauensverlust in die Anwendung.

[34] http://www.stern.de/
[35] http://consti.de/2006/09/04/bundeskanzlerin-merkel-tritt-zuruck/

Ein weiteres Problem ist die Tatsache, dass Variablen in JavaScript nicht typisiert sind und sich somit während der Ausführung des Skriptes ändern können. So hat man z.B. die Möglichkeit einer definierten Variablen, die numerische Werte enthalten sollte, zur Laufzeit einen String zuweisen.

Die Objekteigenschaft „prototype“ ist ein weiterer Schwachpunkt, denn auf diese Art und Weise können leicht Methoden und Eigenschaften zu bereits definierten Objekten zur Laufzeit hinzugefügt werden. Auch Systemfunktionen können durch benutzerdefinierte Funktionen ersetzten werden, wie das folgende Beispiel zeigt.

```
old_alert = alert;
function myalert(str)  {
      old_alert('myalert: '+str);
}
alert = myalert;
alert('Test');
```

Abbildung 46: Überschreiben einer globalen JavaScript Funktion

Wie zu sehen ist, wird die globale Funktion altert() durch eine eigene Definition ersetzt. Beim Aufruf dieser Funktion im nachfolgenden Quellcode wird nicht mehr die ursprüngliche Implementierung aufgerufen, sondern die soeben definierte Variante.

Die Systemfunktion eval() erlaubt es, einen übergebenen String on the Fly als JavaScript-Code auszuführen. Bösartiger Code kann zum jetzigen Zeitpunkt nur unzureichend identifiziert (z.B. Filter) werden, dadurch stellen Funktionen mit dem Umfang von eval() ein enormes Sicherheitsrisiko dar.

JavaScript ist eine sehr flexible Scriptsprache und bietet dem Anwender vielfältige Möglichkeiten Anwendungen zu schreiben. Durch das wachsende Know-How in diesem Bereich werden immer innovativere Anwendungen bzw. Frameworks entstehen, von denen man nie gedacht hätte, dass sie mit JavaScript realisiert werden. Aber Tatsche ist, dass JavaScript ein enormes Sicherheitsrisiko darstellt.

6.2.2 Häufige Attacken auf Webanwendungen

Einige der beliebtesten Angriffsmethoden auf Webanwendungen sollen in diesem Abschnitt vorgestellt werden. Es werden lediglich die Prinzipien vorgestellt, die hinter den Angriffen stehen und keinerlei Anleitungen gegeben.

6.2.2.1 Cross Site Scripting Angriffe

Beim Cross Site Scripting (XSS) geht es darum manipulierte Benutzereingaben an die Webanwendung zu übergeben. Dabei werden vorhandene Sicherheitslücken ausgenutzt, die es dem Angreifer erlauben bösartigen Programmcode in eine normalerweise für den Benutzer vertrauenswürdige Umgebung einzuschleusen. Außerdem kann versucht werden die Kontrolle über die Anwendungen zu erlangen, um sie für eigene

Zwecke zu missbrauchen oder Benutzerdaten auszuspionieren bzw. zu verändern. XSS Angriffe werden typischerweise in Sprachen durchgeführt, die von aktuellen Browsern interpretiert werden können, z.B. JavaScript, VBScript oder HTML Tags und aktive Komponenten wie Applets oder ActiveX einbinden. In der folgenden Abbildung werden zwei Varianten des XSS am Beispiel eines Online Banking Systems vorgestellt, das client- und serverseitige XSS.

Die linke Abbildung zeigt das hinter clientseitigem XSS stehende Prinzip. Dabei muss der Angreifer zuerst Kontakt (2) mit seinem Opfer aufnehmen. Das geschieht in der Praxis häufig durch anonyme Kontaktmails oder SPAM-Mails in denen XSS Angriffe eingearbeitet sind und nutzt dabei Sicherheitslücken in der Browsersoftware aus. Der Browser sendet die vertraulichen Daten, wenn der Benutzer in seinen Online Banking Account eingeloggt (1) ist, an den Angreifer (3) und untergräbt damit die Sicherheitsmaßnahmen des Online Banking Systems.

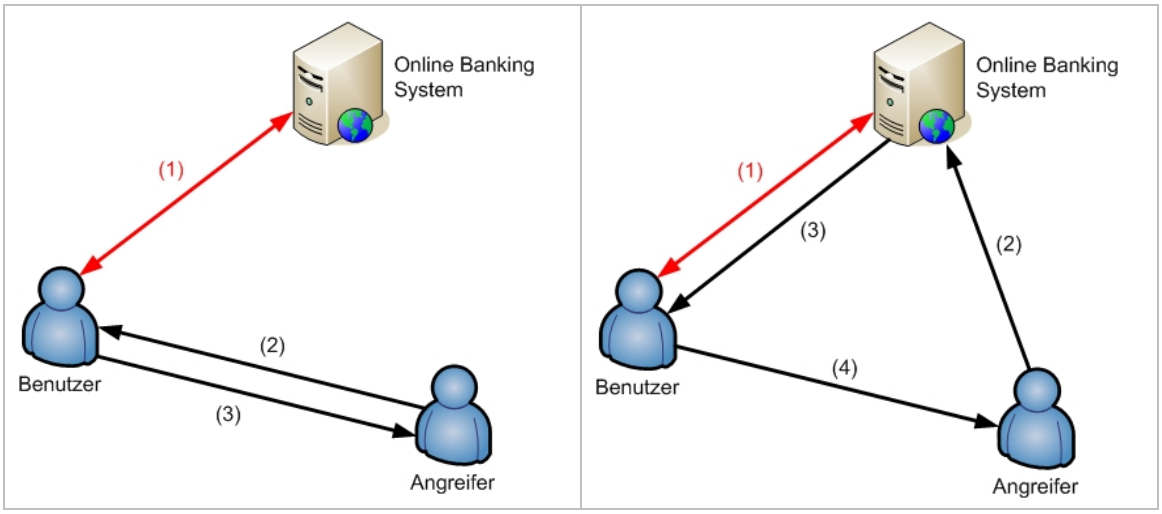

Abbildung 47: links: clientseitiges XXS, rechts: serverseitiges XSS

Eine andere Variante wird in der rechten Abbildung dargestellt, dort ist das Prinzip des so genannten serverseitigen XSS zu sehen. Bei diesem Angriff nimmt der Angreifer zu keinem Zeitpunkt direkten Kontakt mit seinem Opfer auf. Hierbei wird versucht die Anwendung serverseitig zu manipulieren und ihr einen Befehl einzuflüstern (2) die Daten an den Angreifer (4) zu senden. Dabei muss das Opfer eingeloggt (1) sein, denn zwischen ihm und dem Online Banking System besteht ein Vertrauensverhältnis. Bei dieser Art von Angriff werden die Sicherheitslücken der Webanwendungen ausgenutzt und die Verantwortung für das Beheben dieser Fehler liegt beim Betreiber.

Angriffsflächen bieten hier vor allem der DOM des Browsers, dort versucht man neue Elemente einzuschmuggeln und so das aussehen der Webseite zu verändern, um den Anwender zu täuschen. Aber auch andere Technologien sind ein guter Nährboden, so sind z.B. SQL, JSON oder XPath beliebte Angriffsziele.

Aber wie kann man sich gegen solche Angriffe schützen? Auf Anwenderseite sollte man skeptisch gegenüber E-Mails mit unbekanntem Absender sein. Außerdem ist das gleichzeitige Surfen auf andern Webseiten riskant und sollte so lange der Anwender im Online Banking System eingeloggt ist eingestellt werden. Auf Betreiberseite kann durch überprüfen der Benutzereingaben die größte Gefahr abgewendet werden, dies ist aber oft zu teuer und stellt kein sichtbares Feature der Anwendung dar, deshalb spart man sich diese Funktionen oft bei der Entwicklung.

6.2.2.2 Forceful Browsing

Mit Forceful Browsing soll erreicht werden, nicht zugängliche, also vom Betreiber nicht verlinkte Bestandteile einer Webseite bzw. nicht öffentliche Teile, zugänglich zu machen. Der Angreifer hat die Möglichkeit beliebige URLs zu senden, die übergebenen Parameter frei zu wählen und er muss sich nicht an die durch HTML vorgegebenen Längenbeschränkungen (GET, POST) halten. Außerdem besteht die Möglichkeit Formularfelder und Cookies zu manipulieren.

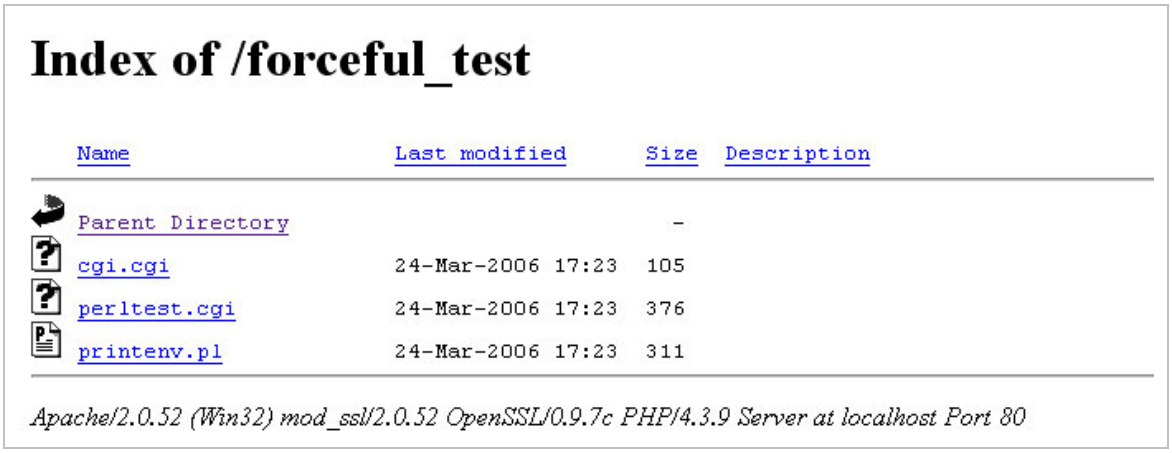

Abbildung 48: Ungeschützte Bestandteile einer Webseite

Ein sehr einfaches Beispiel für einen Angriff ist z.B., dass der Angreifer versucht Namen von Dateien zu erraten, die ungeschützt auf dem Server liegen. Das können z.B. nicht geschützte Konfigurationsfiles oder eine ältere Version der Webanwendung sein. Durch die Analyse dieser Dateien können wertvolle Informationen z.B. Passwörter, Datenbankinformationen oder Schwachstellen im Quellcode der Webanwendung herausgefunden werden. Der Server kann jedoch sehr leicht vor diesen Angriffen geschützt werden, indem alle nicht öffentlichen Dateien in einem nicht zugänglichen Verzeichnis abgelegt werden.

So kann ein anderes Beispiel wie folgt aussehen. Die meisten Content Management Systeme (CMS) haben einen Mechanismus um Dokumente anzusprechen, die in der Datenbank abgelegt sind. Das geschieht in den meisten Fällen durch die Übergabe eines Parameters mit GET (z.B. id=2345). Dieser Parameter kann vom Angreifer auch zufällig gewählt werden und dieser kann Zugriff auf ein geschütztes Dokument

erlangen. Hier müssen die verschiedenen Anbieter solcher Systeme geeignete Mechanismen implementieren, um nur befugten Personen den Zugriff zu erlauben.

Forceful Browsing Angriffe sind durch ihre Einfachheit sehr leicht zu automatisieren und werden meistens in Verbindung mit anderen Angriffsmethoden (z.B. XSS, Buffer-Overflows[36]) genutzt. Wie aber gezeigt wurde, kann man sich sehr leicht vor diesen Angriffen schützen.

6.2.2.3 Cross Side Request Forgery

Cross Side Request Forgery (CSRF) wird auch als Session Riding bezeichnet und ist das Gegenteil von XSS. Die im Browser gespeicherten Informationen (z.B. Cookies) werden missbraucht, um im Namen des Benutzers eine Aktion auf der Webseite auszuführen. Dabei wird wie der Name schon sagt, die aktive Session des Benutzers ausgenutzt. Session Riding ist ein technisch sehr anspruchvoller Angriff und das Grundprinzip ist in der folgenden Abbildung dargestellt.

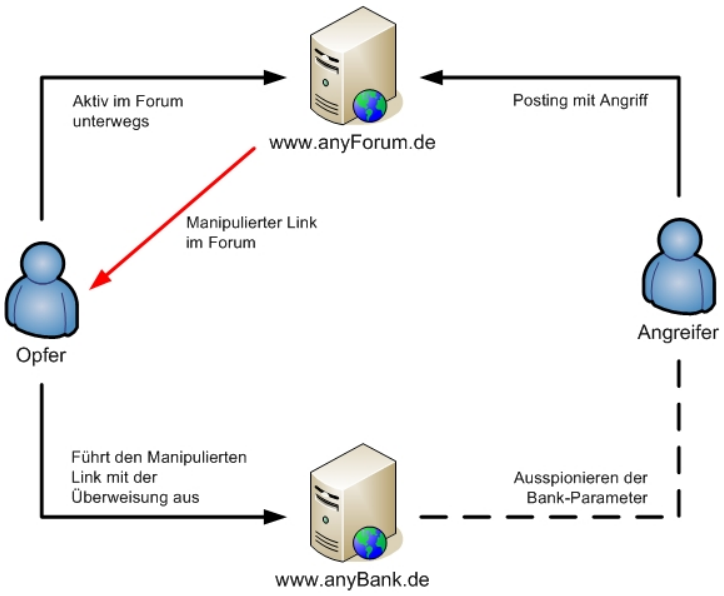

Abbildung 49: Prinzip des Cross Side Request Forgery

Der erste Schritt des Angreifers ist es den Angriff gründlich vorzubereiten, dass tut er, indem er Informationen über das Opfer sammelt, z.B. das er ein aktives Mitglied des Forums (anyForum) ist und bei der Bank (anyBank) ein Konto hat. Der nächste Schritt ist es zu überprüfen, ob die Bank gegenüber CSFR Angriffen anfällig ist und mit Hilfe welcher Parameter Überweisungen durchgeführt werden. Nun hat der Angreifer alle notwendigen Information um den Angriff durchzuführen. Das kann er z.B. tun, indem er in demselben Forum ein Beitrag eröffnet, indem ein Bild einfügt wird. Diese URL enthält jedoch nicht den Link auf eine Bild, sondern auf das Überweisungsformular der

[36] http://www.heise.de/security/artikel/37958

Bank mit allen notwendigen Parametern (z.B. Kontonummer, Betrag). Jeder der diesen Beitrag aufruft bekommt lediglich ein Broken-Image angezeigt, aber im Hintergrund wird versucht eine Überweisung durchzuführen.

Für die Benutzer, die kein Konto bei der Bank hat ist der Aufruf ungefährlich, aber falls jemand eine gültige Session hat wird die Überweisung unsichtbar für den Benutzer durchgeführt. Er wird frühestens auf den Betrug aufmerksam, wenn er seinen nächsten Kontoauszug sorgfältig kontrolliert und selbst dann hat er als einzigen Anhaltspunkt den Beitrag im Forum. Doch der Anwender kann sich auf sehr einfache Weise vor solchen Angriffen schützen. Wenn er z.B. Online Banking Systeme nutzt, sollten alle anderen Browserfenster geschlossen sein, somit kann die aktive Session nicht missbraucht werden.

6.2.2.4 Men in the middle Attack

Eine weit verbreiteter und jedem bekannter Angriff auf Anwendungen ist die so genannte Men in the middle Attack. Der Angreifer versucht sich auf dem Kommu-nikationsweg in die Verbindung zwischen Client und Server einzuschleusen. Es bieten sich auf dieser Strecke eine Reihe von Möglichkeiten dies zu tun, so kann z.B. ein Router gehackt worden sein. Er hat damit die Möglichkeit die übertragenen Daten einzusehen oder zu verändern. Das stellt eine große Gefahr für AJAX-Anwendungen dar, denn jegliche Kommunikation, sei es die Übertragung des JavaScript-Codes, die Datenübertragung mit dem XMLHttpRequest-Objekt, die synchrone Kommunikation (siehe 2.2.1.3) oder die Nutzung eines Webservices basiert auf dem HTTP Protokoll. Dadurch bieten sich dem Angreifer zusätzliche Möglichkeiten die Anwendung zu attackieren.

6.2.2.5 Viren und Würmer

Viren und Würmer haben durch das Web 2.0 ideale Möglichkeiten um sich zu verbreiten. AJAX steht zur Verfügung, um den schädlichen Code in das System des Benutzers zu schleusen und dort vom Browser ausführen zu lassen. Es ergeben sich neue Verbreitungswege durch RSS und REST. Mashups und Wikis können genutzt werden, um Viruscode in die Systeme der Benutzer dieser Dienste einzuschleusen.

Der Myspace[37] Wurm - “Samy is my Hero” war einer der ersten Schädlinge, die AJAX für ihre Zwecke ausgenutzt haben. Einer der Benutzer (Samy) von Myspace fand, dass er zu wenig Freund hatte (73 an der Zahl) und wollte diese Situation ändern. Er benutzte JavaScript im eigenen Profil, das mit Hilfe des XMLHttpRequest-Objektes Samy zum Freund machte. Nach etwa 20 Stunden hatte Samy mehr als 1.000.000 Freunde. Der Kern des Problems war an dieser Stelle, dass JavaScript im Profil des Benutzers erlaubt war. Der Schütz vor solchen Angriffen ist sehr schwierig, denn man hat als Betreiber nicht all zu viele Möglichkeiten. Trotz eines JavaScript Filters hat es auch Myspace getroffen. Aber wie soll mit Hilfe eines Filters gutartiger von bösartigem

[37] http://www.myspace.com/

JavaScript Code unterschieden werden. Diese Aufgabe ist nicht lösbar, denn Filter funktionieren heute meisten auf der Basis von Suchmustern und diese können einfache manipuliert werden.

Die Betreiber (Communities, Wikis) haben die alternative JavaScript nicht oder komplett zuzulassen. Eine andere Möglichkeit besteht darin, eine eigene JavaScript API zur Verfügung zu stellen und jeglichen Code, der diese API nicht nutzt, heraus zu filtern. All diese Lösungen bieten jedoch keinen idealen Schutz vor Angriffen.

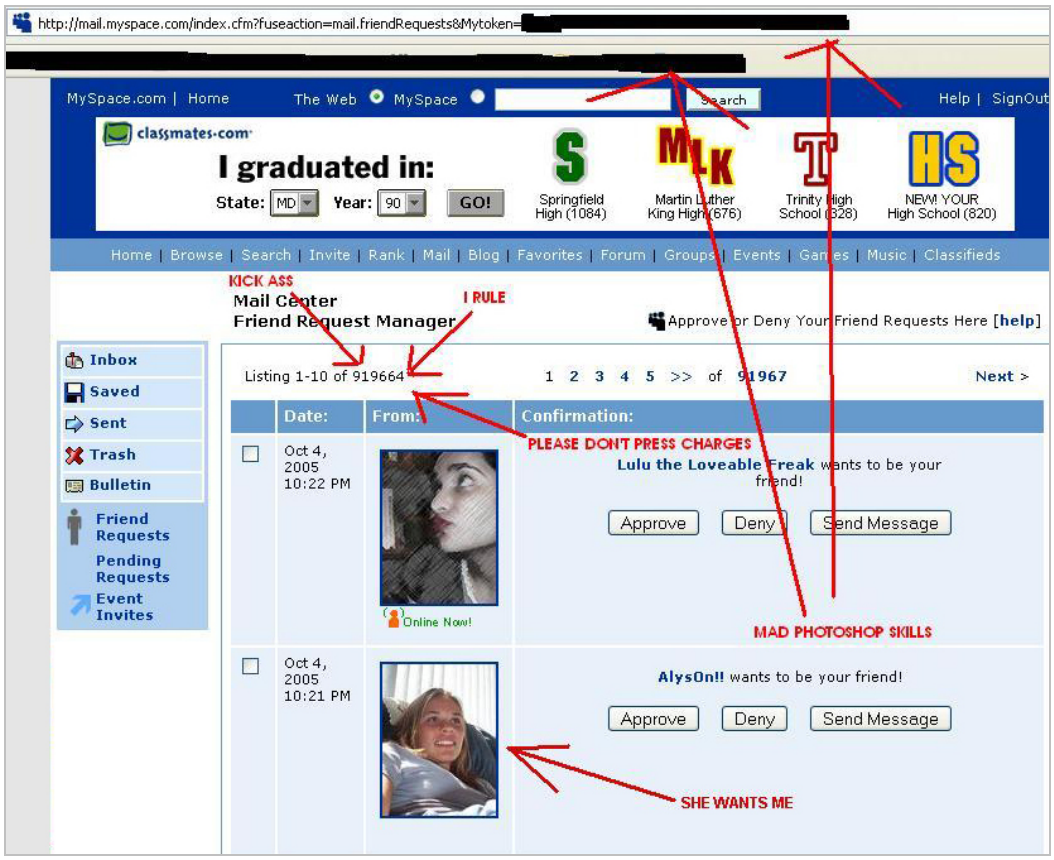

Abbildung 50: Myspace JavaScript Wurm Samy[38]

6.3 AJAX Anwendungen absichern

Die wachsende Beliebtheit und zunehmende Verbreitung von Frameworks sowie die Verlagerung von mehr Anwendungslogik auf den Client zieht mehr potentielle Sicherheitslücken in AJAX-Anwendungen nach sich. JavaScript etabliert sich zunehmend als Angriffsplattform und immer mehr Viren, Würmer, domainübergreifende Angriffe auf Webanwendungen werden stattfinden. Deshalb werden zukünftig Anwendungen einer stetig wachsenden Gefahr ausgesetzt sein gehackt zu werden.

[38] http://fast.info/myspace/

Aber wie können AJAX-Anwendung geschützt werden und ist ein hinreichender Schutz vor Angriffen möglich? Bei der Betrachtung der AJAX-Standardarchitektur, können leicht die Schwachstellen ausgemacht werden. Der Client sendet Anfragen über das HTTP Protokoll im Klartext an den Server. Diese Anfragen werden von serverseitigen Prozessen (z.B. Servlets) bearbeitet und die resultierenden Antworten werden wieder an den Client gesendet. Der Client analysiert die eingehenden Daten und verarbeitet sie. Die Schwachstellen sind der Transportweg über HTTP, der Client, welcher im Browser ausgeführt wird und die serverseitigen Bestandteile der Anwendung.

6.3.1 Absicherung des Transportweges

AJAX greift wie es auch normale Webanwendungen tun, bei der Übermittlung von Anfragen an den Server auf HTTP zurück. Der Browser hat dabei keine direkte Verbindung zum Webserver. Die Daten werden über diverse Zwischenstationen (z.B. Router) übertragen und können auf diesem Weg von jeder Person analysiert und manipuliert werden. Angriffe auf diesen Schwachpunkt der Anwendung bezeichnet man als Man in the Middle Attack (Siehe Abschnitt 6.2.2.4).

Wenn man den Netzverkehr zwischen dem Client und Server schützen möchte, kann auf ein alt bewährtes Mittel zurückgegriffen werden. HTTP über Secure Socket Layer (SSL) ermöglicht die Verschlüsselung der Daten in beiden Richtungen. Es besteht zwar immer noch die Möglichkeit, dass die Daten von einem Angreifer mitgeschnitten werden, aber er kann mit den verschlüsselten Daten nichts anfangen. HTTP(s) ist keine vollständige Sicherheitslösung, denn nur die Datenübertragung ist durch Verschlüsselung abgesichert. Es bietet keine Möglichkeiten die Daten im Client bzw. Server zu schützen, hier müssen eigene Sicherheitslösungen für die betroffenen Bereiche implementiert werden.

6.3.2 Sicherheit auf dem Client

Der Quellcode wird komplett über das Netzwerk an den Browser übertragen und zur Ausführung muss JavaScript aktiviert sein. Die von Herstellerseite unternommenen Sicherheitsmaßnahmen, zur Ausführung von JavaScript-Code in einer sicheren Umgebung, wurden bereits in Abschnitt 6.1 besprochen. Es wurde festgestellt, dass dem übertragenen JavaScript-Code nicht vertraut werden kann und eine Überprüfung sehr aufwändig ist.

Doch die zwischen Client und Server übertragenen Daten (Header, Cookies, Benutzerdaten) sollten überprüft werden. Dies kann durch eine Definition der Datenformate erheblich erleichtert werden und somit XSS-Angriffe, von denen die meiste Gefahr ausgeht, erheblich einschränken.

Ein anderes Problem ist, dass immer mehr Frameworks entstehen und eingesetzt werden. Diese Frameworks agieren frei nach dem Motto: „Man muss nicht verstehen, wie es funktioniert“. Dies ist sehr gefährlich, da der Entwickler sich mit dem Thema

JavaScript intensiv auseinandergesetzt und die Funktionsweise verstanden haben muss, um sichere AJAX-Anwendungen zu entwickeln.

Ein anderer Punkt ist das die betrachteten Frameworks (siehe Abschnitt 5.2.6.1) keine durchgehenden Sicherheitskonzepte zur Verfügung stellen. Hier muss der Entwickler sehr viel Arbeit investieren, um ein Sicherheitskonzept zu implementieren.

Noch bedenklicher ist die Tatsache, dass viele sicherheitsrelevante Vorgänge vom Benutzer beeinflusst werden können (siehe Abschnitt 6.1.4). Die integrierten Sicherheitskonzepte der Browser sind nur bedingt für AJAX-Anwendungen geeignet.

In Bezug auf die Nutzung von AJAX ist Aufklärung auf Benutzer- und Entwicklerseite notwendig. Der Entwickler muss sich über die Angriffe, die sich nur in ihrem Umfang von denen auf normale Webanwendungen unterscheiden, informieren. Außerdem müssen die Benutzer über die Sicherheitsrisiken besser aufgeklärt werden.

Trotz aller getroffenen Vorsichtmaßnahmen und noch so aufwändigen Sicherheitskonzepte können AJAX-Anwendungen niemals sicher sein.

6.3.3 Eine sichere Webschicht

Den serverseitigen Teil einer AJAX-Anwendung kann man auch als einen Webdienst bezeichnen. Doch dieser Dienst kann auch von Außenstehenden genutzt werden. Es kann nicht festgestellt werden, ob die Daten vom Client oder von Außenstehenden stammen, die den Dienst missbrauchen. Es müssen geeignete Maßnahmen getroffen werden, die einen Missbrauch des Dienstes verhindern können. So sollten die eingehenden Daten überprüft werden. Hier stehen diverse serverseitige Mechanismen, z.B. Filter, Frameworks ([OWASPVAL]) zur Verfügung. Außerdem kann der Webdienst so entworfen werden, dass nur bestimmte Anwender Zugriff haben.

Bei normalen Webanwendungen werden häufig Filter eingesetzt, die ein- und ausgehende Scripte überprüfen. In diesem Umfeld ist die Aufgabe noch überschaubar und mit relativ wenig Aufwand zu erfüllen. Bezogen auf AJAX ist das Herausfiltern von schädlichen Scripten nicht mehr so einfach, denn die Anzahl der Angriffsmöglichkeiten ist stark gestiegen. Es muss also auf speziell zugeschnittene Lösungen zurückgegriffen werden.

Der Entwickler muss beim Erstellen des serverseitigen Teils der Anwendung gegenüber normalen Webanwendungen umdenken. Bei normalen Webanwendungen sind die Daten unter einer Menge von HTML, CSS und anderen dekorativen Inhalten versteckt. Im Gegensatz dazu sind bei AJAX-Anwendungen die Daten wesentlich einfacher und strukturierter, damit sie clientseitig leichter und schneller analysiert werden können. Dies erleichtert so genannte Screen Scraping Angriffe[39] erheblich.

[39] http://www.tecchannel.de/news/themen/sicherheit/432587/index2.html

Beim Design des serverseitigen Bestandteils der AJAX-Anwendung oder des Dienstes, müssen diese Aussagen vom Entwickler beachtet werden, um die Sicherheit zu gewährleisten.

7 Integration in die DCX Infrastruktur

Nachdem AJAX theoretisch vorgestellt und verschiedene Frameworks analysiert wurden, werden in diesem Kapitel verschiedene Aspekte betrachtetet, die für die Integration von AJAX in die DCX Infrastruktur wichtig sind. Das sind folgende Punkte:

- Es wird ein Problem vorgestellt, das beim Zusammenspiel von AJAX und SiteMinder aufgetreten ist.
- Weiterhin wird betrachtet, ob es Sinn macht den PAI Client Container mit AJAX abzubilden bzw. einzelne Komponenten.
- Die im Rahmen der Diplomarbeit entstandene Beispielanwendung wird vorgestellt.

7.1 SiteMinder

In Abschnitt 5.2.5 wurden die verschiedenen AJAX-Frameworkmodelle hinsichtlich ihrer Architektur und der sich daraus ergebenen Vorteile bzw. Probleme ausführlich betrachtet. Bei der Integration von AJAX bzw. der Frameworkmodelle in die PAI Infrastruktur trat nachfolgendes Problem auf.

7.1.1 PAI Login Mechanismus

Von der PAI Plattform werden verschiedenen Mechanismen unterstützt, um den Benutzer von Anwendungen den Login zu ermöglichen. Das ist auf der einen Seite der lokale Login Mechanismus und auf der anderen Seite der globale Login Mechanismus. Dabei wird der Login prinzipiell in zwei Schritten durchgeführt. Die Authentifizierung des Benutzers erfolgt gegen das Authentication Direktory, um zu überprüfen ob der Benutzer eine ID hat, während im zweiten Schritt mit Hilfe des Autorisation Direktory überprüft wird, ob der Benutzer die Startautorisierung für die jeweilige Anwendung hat.

7.1.1.1 Standartarchitektur von PAI

Wie bereits in Abschnitt 2.3.3 angesprochen wurde, besteht die J2EE Plattform aus dem IBM Applikation Server. Die Besonderheit ist, dass der IIS Webserver (ein von IBM modifizierter Apache) dem eigentlichen Application-Server vorangestellt ist, um nur erlaubten Anfragen den Zugriff auf den Applikation-Server zu erlauben. Der Webserver enthält den SiteMinder-Agent, der die Kommunikation mit SiteMinder ermöglicht und das Schützen von auf dem Application-Server installierten Anwendungen ermöglicht. SiteMinder dient zur Authentifizierung und Autorisierung von Identitäten, die dann gegen spezielle Direktories auf Echtheit geprüft werden. Wichtig ist das beim globalen Login Szenario SiteMinder diese Aufgaben übernimmt und beim lokalen Login Szenario der Application-Server.

7.1.1.2 Globales Login Szenario

Dieses Szenario kann für Internet- und Intranet Anwendungen eingesetzt werden und ermöglicht die Realisierung von SSO. Jeder Aufruf einer URL wird vom SiteMinder-Agent überprüft, um festzustellen, ob die Anwendung geschützt ist und der Benutzer eine gültige SiteMinder-Session hat. Wenn die Anwendung nicht durch SiteMinder geschützt ist bzw. eine gültige SiteMinder-Session existiert, wird der Aufruf an den Application-Server weitergeleitet und die zugehörige Anwendung ausgeführt. Falls der Benutzer keine gültige SiteMinder-Session hat, bemerkt das der Agent und forciert einen Login. Der Login wird mit Hilfe des Weblogin ([WEBLOGIN]) durchgeführt.

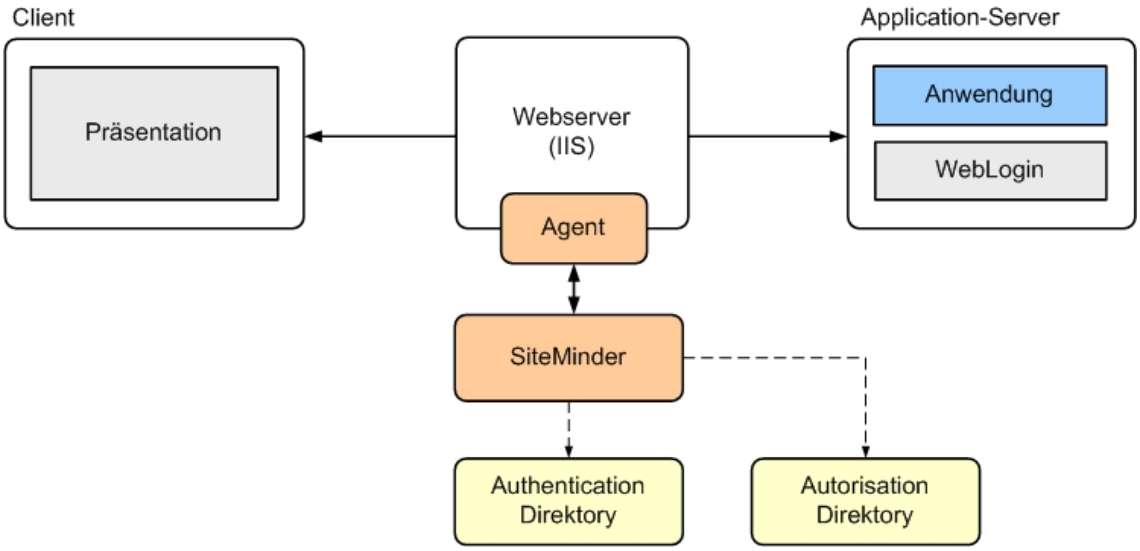

Abbildung 51: PAI Standartarchitektur

Der Weblogin ist eine Webanwendung, die nicht durch SiteMinder geschützt ist und die Eingaben des Benutzers (Passwort und BenutzerID) entgegennimmt. Diese werden dann mit Direktories gegen geprüft und der Benutzer kann nach einer positiven Überprüfung die Anwendung nutzen. Andere Aufgaben dieser sehr komplexen Webanwendung sind die Auswertung der SiteMinder Fehlercodes, sowie ein Login Formular zur Verfügung zu stellen, das von jeder Webanwendung genutzt werden kann. Der Entwickler hat somit die Möglichkeit Webanwendungen ausschließlich durch Konfiguration zu schützen.

7.1.1.3 Locales Login Szenario

Beim lokalen Login Szenario übernimmt der Application-Server die Authentifizierung des Benutzers und die Startautorisierung der Anwendung. Der Application-Server kann auf die Direktories zugreifen und die notwendigen Überprüfungen durchführen. Es werden lediglich Intranet Anwendungen unterstützt und SSO ist nur Application-Server weit möglich, sprich nur für Anwendungen, die auf dem selben Server installiert sind.

7.1.2 Session/Idle Timeout

Für das Verständnis des eigentlichen Problems ist es wichtig zu verstehen, wie SiteMinder reagiert, falls die Session abläuft. Der vorgestellte Mechanismus wird auch beim Login angewendet.

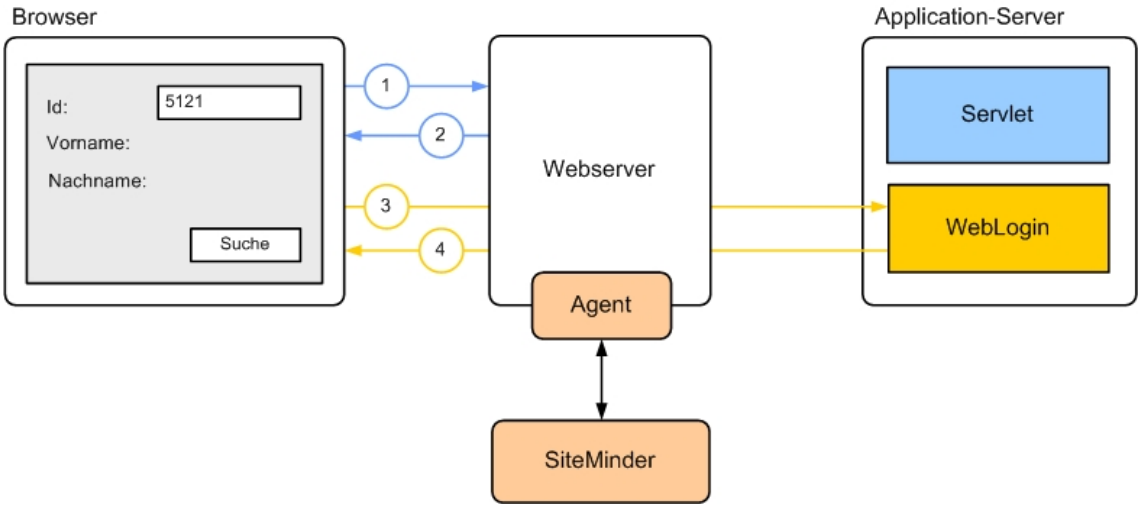

Abbildung 52: Session/Idle Timeout bei PAI

In *Abbildung 52* und *Tabelle 3* wird der Session/Idle Timeout beschrieben.

Schritt	Actor	Beschreibung
1.	Benutzer	Status: Session/Idle Timeout
2.	Benutzer	klickt auf einen Link
3.	Browser	ruft die Webseite auf (1), per GET
4.	Agent	überprüft die Gültigkeit der Session
5.	Agent	leitet auf Weblogin weiter, wegen Session/Idle Timeout, Statuscode 302 (2)
6.	Browser	leitet den Aufruf transparent weiter (3)
7	Weblogin	sendet das Login Formular an den Browser (4)
8.	Browser	zeigt das Login Formular an, Statuscode 200
9.	Benutzer	loggt sich mit ID und Passwort ein

Tabelle 3: Use-Case Session/Idle Timeout bei PAI

7.1.3 Problem: Session/Idle Timeout mit AJAX

Beim Testen des Session/Idle Timeout Szenarios mit einer AJAX-Anwendung trat ein unerwarteter Effekt auf. Falls die SiteMinder-Session ausgelaufen ist, wird der gleiche Mechanismus, wie zuvor beschrieben, in Gang gesetzt. Da asynchrone Anfragen aber mit Hilfe des XMLHttpRequest-Objektes ausgeführt und bearbeitet werden, befindet sich der HTML-Stream des Login Formulars in der Ergebnisvariablen der Scripts. Der Effekt tritt ausschließlich bei asynchronen Aufrufen in Verbindung mit dem Weblogin auf. Diese Problematik wird nun etwas näher betrachtet.

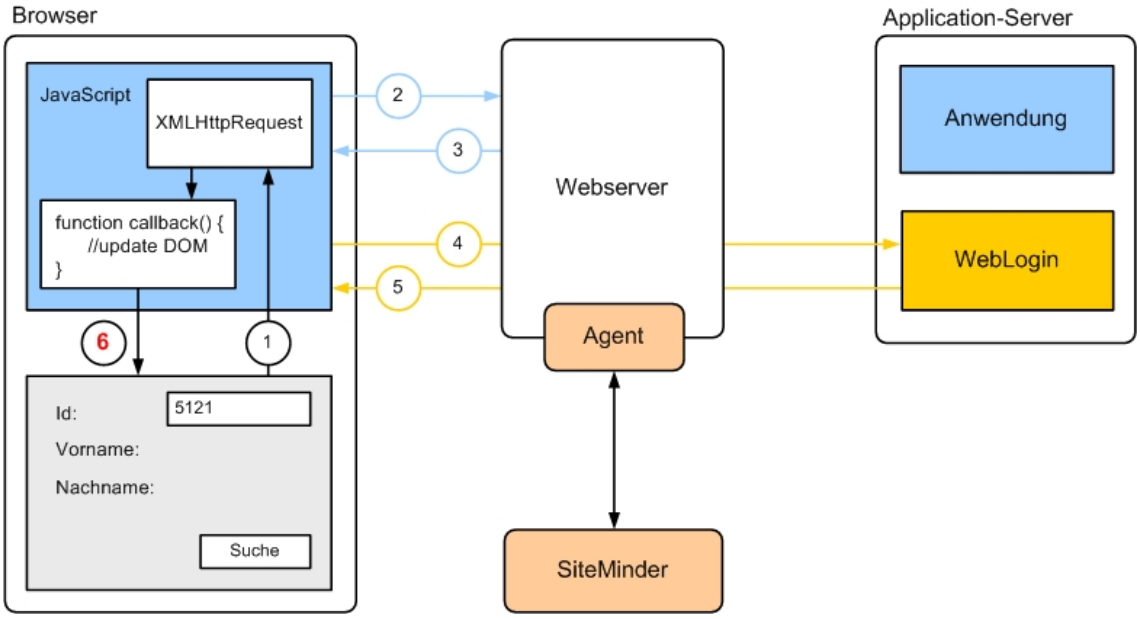

Abbildung 53: Session/Idle Timeout bei PAI mit AJAX

In *Abbildung 53* ist der Vorgang des Session/Idle Timeout dargestellt. Der entsprechende Use-Case ist in *Tabelle 4* zu finden.

Schritt	Actor	Beschreibung
1.	Benutzer	Status: Session idle timeout
2.	Benutzer	klickt auf einen Button, um Daten abzufragen (1)
3.	XMLHttpRequest	asynchroner Aufruf eines Dienstes - URL (2)
4.	Agent	überprüft die Gültigkeit der Session (3), Statuscode 302
5.	Agent	leitet zum Weblogin weiter, wegen einem Session Time-out
6.	XMLHttpRequest	leitet den Aufruf transparent zum Weblogin weiter (4)
7.	Weblogin	sendet das Login Formular an den Browser (5)

8.	XMLHttpRequest	Antwort wird in der Callback-Funktion ausgewertet, Statuscode 200
9.	XMLHttpRequest	analysiert die empfangenen Daten
10.	XMLHttpRequest	Error: Daten haben nicht das definierte Format, da es der HTML-Stream des Login Formulars ist
11.	XMLHttpRequest	Error: Callback-Funktion kann die Daten nicht auswerten
12.	XMLHttpRequest	Error: DOM kann nicht aktualisiert werden
13.	Browser	Login Formular kann nicht angezeigt werden (6)
14.	Benutzer	kann sich nicht einloggen

Tabelle 4: Use Case AJAX Session/Idle Timeout bei PAI

Der Grund ist, wie die Kommunikation mit Hilfe des XMLHttpRequest-Objektes (siehe Abschnitt 2.2.2.2.3) durchgeführt wird. In der *Abbildung 54* ist ein Beispiel für einen asynchronen Aufruf (siehe Abschnitt 2.2.1.3) zu sehen. Nach dem erfolgreichen Ausführen des Aufrufes, wird das Ergebnis in einer Callback-Funktion ausgewertet.

```
var xmlHttp = false;
//XMLHttpRequest-Object browserunabhängig erzeugen
…….
// zusammensetzen der URL, die Sie aufrufen wollen
var url = "/call_servlet";
// öffnen der asynchronen Verbindung zum Server
xmlHttp.open("GET", url, true);
// festlegen der Callback-Funktion
xmlHttp.onreadystatechange = updatePage;
// Absenden der Anfrage
xmlHttp.send(null);
function updatePage() {
    if (xmlHttp.readyState == 4)  {
        //Ergebnisvariable der Anfrage
        var response = xmlHttp.responseText;
        …….
        }
}
```

Abbildung 54: JavaScript Beispiel Session/Idle Timeout bei PAI

In dieser Funktion kann außerdem der gesamte Verlauf der Kommunikation mit Hilfe des Ready-Status und HTTP-Statuscodes verfolgt werden. Das XMLHttpRequest-Objekt hat die Einschränkung, dass keine Weiterleitungen (Statuscode 302) verarbeitet werden können. Laut der W3C Spezifikation ([W3CXMLHTTP]) werden sie transparent durch den Browser geleitet.

Es steht ausschließlich der HTML-Stream des Login Formulars in der Ergebnisvariablen zur Verfügung und das Login Formular kann dem Benutzer nicht präsentiert werden, da keine Logik vorhanden ist, die diese Daten interpretieren kann.

Es muss also ein Mechanismus bereitgestellt werden, der auch mit dem asynchronen Kommunikationsmodell funktionsfähig ist .

7.1.4 Lösungskonzepte

In diesem Abschnitt sollen die verschiedenen Lösungsmöglichkeiten vorgestellt werden, wie der Weblogin für AJAX-Anwendungen bereitgestellt werden kann.

7.1.4.1 Modifikation des XMLHttpRequest-Objektes

Die browserseitigen Implementierungen des XMLHttpRequest-Objektes können so angepasst werden, dass der Entwickler den HTTP-Statuscode (302) in der Callback-Funktion auswerten kann.

Der Nachteil dieses Ansatzes ist, dass alle gängigen Browser Implementierungen (IE, Firefox, Opera: [OPERA]) angepasst und gewartet werden müssten. Dies würde einen enormen Kostenfaktor darstellen und da für einige Browser (z.B. IE) kein Quellcode zu beschaffen wäre, ist diese Lösung nicht zu realisieren.

7.1.4.2 Framework stellt einen Mechanismus bereit

Das Framework stellt einen Mechanismus bereit, um die empfangenen Daten zu überprüfen. Das kann z.B. durch einen Tag geschehen, der die Echtheit der Daten garantiert (z.B. Hash). Falls die Daten diesen Tag nicht enthalten, wird automatisch eine definierte Webseite angezeigt und der Benutzer muss sich authentifizieren.

```
<validate id="Hash">
    <data>
        <name>Hans Muster</name>
       <article>a_123456789</article>
    </data>
</validate>
```

Abbildung 55: Bereitstellung eines Mechanismus durch das Framework

In *Abbildung 55* ist ein Beispiel für diesen Ansatz zu sehen, falls die “id” nicht den Richtlinien entspricht, wird automatisch auf eine zuvor in einer Konfigurationsdatei hinterlegte URL verbunden. Das Framework müsste jedoch einen transparenten Mechanismus bereitstellen, der vom Entwickler erweitert werden kann. Diese Lösung hätte den Vorteil, dass das Framework alle Entscheidungen trifft und die Weiterleitung initialisiert. Keines der in Abschnitt 5.2.6.1 betrachteten Frameworks stellt einen solchen Mechanismus zur Verfügung, was für PAI bedeutet, dass die Frameworks um diese Funktionalität erweitert werden müssen.

7.1.4.3 Automatische Weiterleitung durch SiteMinder

Es kann ein Feature von SiteMinder verwendet werden, um im Fall eines Session/Idle Timeout auf eine definierte URL weiterzuleiten. Die Lösungen sind je nachdem ob ein client- bzw. serverseitiges Framework verwendet wird unterschiedlich (siehe Abschnitt 5.2.5), deshalb werden sie gesondert betrachtet.

7.1.4.3.1 Clientseitiges Framework

Die folgende Lösung ist ausschließlich für clientseitige Frameworks geeignet. Damit der Mechanismus zuverlässig funktionieren kann, muss die Anwendung durch SiteMinder geschützt werden. Außerdem muss eine spezielle Webanwendung (Redirect) in der Konfiguration als Rücksprungadresse, im Fall eines Session/Idle Timeouts, angegeben werden.

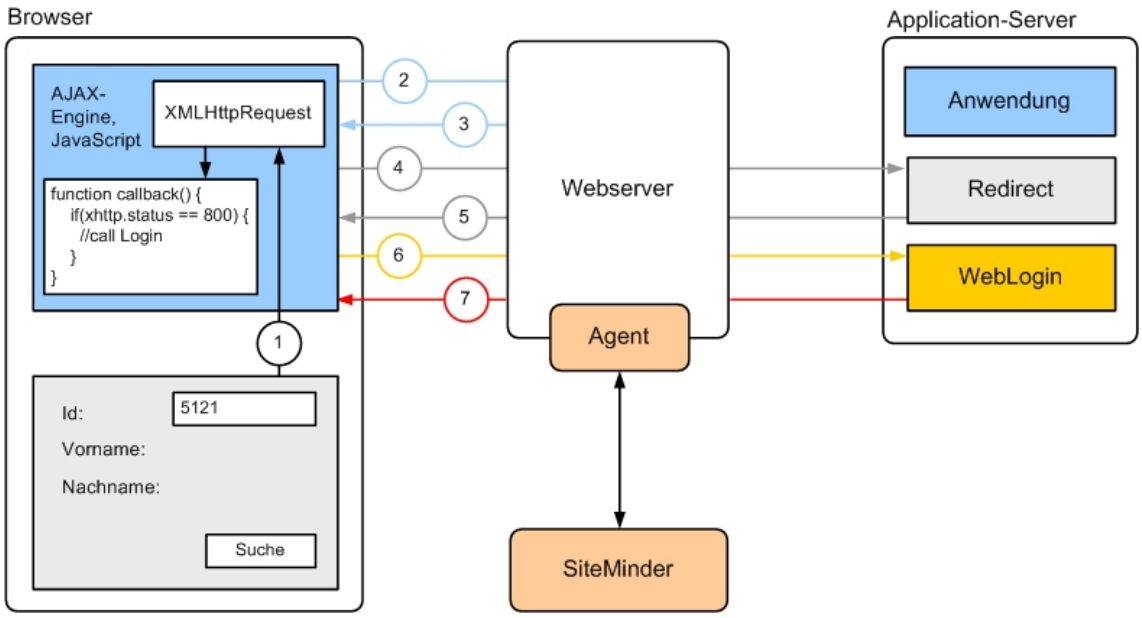

Abbildung 56: Lösung AJAX Session/Idle Timeout mit clientseitigen Frameworks

Das Szenario wird mit Hilfe von *Abbildung 56* und dem dazugehörigen Use-Case (siehe Tabelle 5) näher beschrieben.

Schritt	Actor	Beschreibung
1.	Benutzer	Status: Session/Idle timeout
2.	Benutzer	klickt auf einen Button, um Daten abzufragen (1)
3.	XMLHttpRequest	asynchroner Aufruf eines Dienstes - URL (2)
4.	Agent	überprüft die Gültigkeit der Session

5.	Agent	leitet zur Webanwendung (Redirect) weiter, wegen einem Session/Idle Timeout (3)
6.	XMLHttpRequest	leitet den Aufruf transparent weiter (4), Statuscode 302
7.	Redirect	Modifizierung des HTTP Statuscodes 302 in der Webanwendung (Redirect) auf Statuscode 800
8.	XMLHttpRequest	Antwort wird in der Callback-Funktion ausgewertet (5)
9.	XMLHttpRequest	analysiert die empfangenen Daten
10.	XMLHttpRequest	wertet den Statuscode (800) aus
11.	XMLHttpRequest	GET: auf den Weblogin (6)
12.	Weblogin	sendet das Login Formular an den Browser (7)
13.	Browser	zeigt das Login Formular an (refresh wird ausgeführt)
14.	Benutzer	kann sich einloggen

Tabelle 5: Use-Case Lösung AJAX Session/Idle Timeout bei PAI für clientseitige Frameworks

Die Lösung ist vom Entwickler auf einfache Art und Weise zu implementieren, denn er muss lediglich in der Callback-Funktion auf Statuscode 800 prüfen und einen GET an die Weblogin URL absetzen (siehe *Abbildung 57*). Als serverseitige Komponente kann ein Servlet dienen, das den Statuscode von 302 auf 800 umschreibt. Der Statuscode 800 muss im PAI Umfeld für diesen Zweck reserviert werden.

```
…….
function updatePage() {
    if (xmlHttp.readyState == 4)  {
        switch(xmlHttp.status) {
            case 800:
                location.href = "Weblogin URL";
            default:
                …….
        }
    }
}
```

Abbildung 57: Beispielcode Lösung AJAX Session/Idle Timeout für clientseitige Frameworks

Der Nachteil ist, dass diese Lösung für den Entwickler nicht transparent ist. Er kann nicht die URL der Anwendung durch SiteMinder schützen lassen, sondern muss über die Problematik Bescheid wissen und in der Webanwendung darauf reagieren. Der Entwickler muss client- und serverseitige Komponenten implementieren, die die Anzeige des Login Formulars ermöglichen.

PAI kann durch die Bereitstellung einer serverseitigen Komponente und clientseitigen Bibliothek den Entwickler entlasten und eine PAI konforme Lösung bereitstellen. Eine Möglichkeit für clientseitige Frameworks wird in Abschnitt 7.2.2.1.1 näher betrachtet.

7.1.4.3.2 Serverseitige Frameworks

Bei serverseitigen Frameworks ist die Lösung des Problems schwieriger, denn die client- und serverseitige AJAX-Engine synchronisieren ihren Zustand ständig und bilden somit eine Einheit. Es werden in diesem Zusammenhang nur Componet based Frameworks betrachtet und Serverside AddOns werden nicht beleuchtet, um den Sachverhalt nicht unnötig kompliziert zu gestalten.

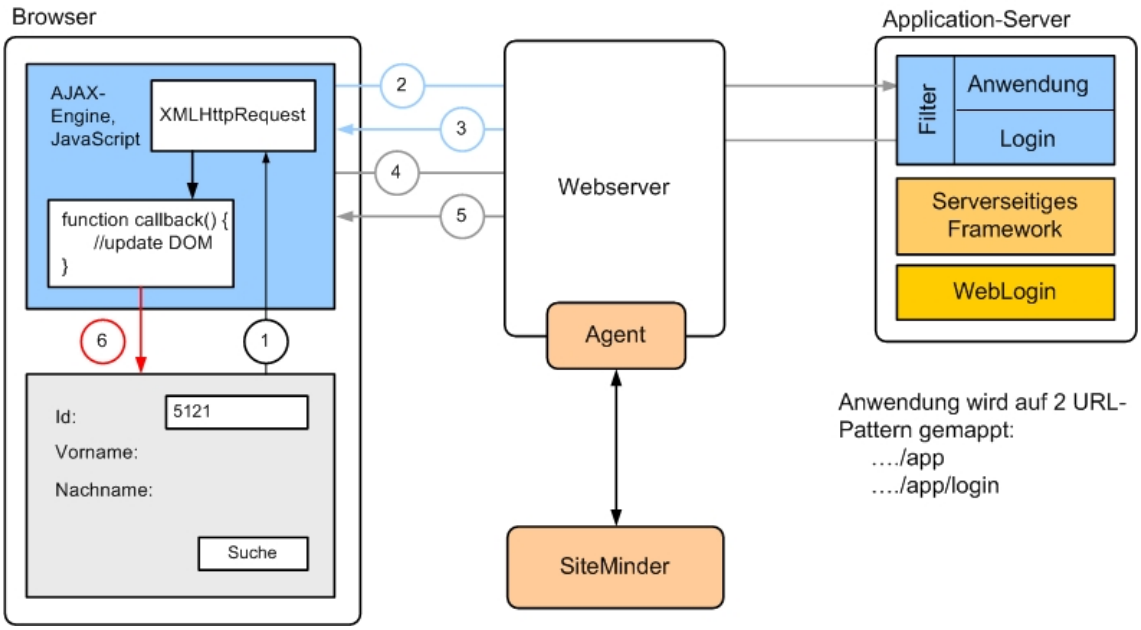

Abbildung 58: Lösung AJAX Session/Idle Timeout mit serverseitigen Frameworks

Der Einstieg in die Anwendung erfolgt in den meisten Frameworks über ein Dispatcher-Servlet, dadurch bleibt das URL-Pattern immer gleich. Diese URL wird SiteMinder geschützt und kann nur aufgerufen werden, wenn der Benutzer eine gültige SiteMinder-Session hat. Damit dieses Szenario funktionieren kann, muss der SiteMinder-Agent im Fall eines Session/Idle Timeout auf eine gültige, nicht geschützte URL der Webanwendung weiterleiten. Aus diesem Grund wird ein 2tes URL-Pattern definiert. Der SiteMinder-Agent kann nun im Fall eines Timeout auf diese URL weiterleiten und der serverseitige Teil der Anwendung kann das Login Formular an den Client übertragen und der Benutzer kann sich einloggen.

Schritt	Actor	Beschreibung
1.	Benutzer	Status: Session idle timeout
2.	Benutzer	klickt auf einen Button, um Daten abzufragen (1)
3.	XMLHttpRequest	asynchroner Aufruf eines Dienstes - URL (2)
4.	Agent	überprüft die Gültigkeit der Session
5.	Agent	leitet auf das 2te URL-Pattern der Anwendung weiter, wegen einem Session/Idle Timeout, Statuscode 302 (3)
6.	XMLHttpRequest	leitet den Aufruf transparent weiter (4), Statuscode 302
7.	Filter	erkennt das auf das 2te URL-Pattern weitergeleitet wurde
8.	Filter	ruft den Login Bestandteil der Anwendung auf
9.	Anwendung	übertragt das Login Formular an den Browser (5)
10.	XMLHttpRequest	Antwort wird in der Callback-Funktion ausgewertet
11.	XMLHttpRequest	analysiert die empfangenen Daten
12.	XMLHttpRequest	aktualisiert den DOM
13.	Browser	Zeigt das Login Formular an (6)
14.	Benutzer	kann sich einloggen

Tabelle 6: Use-Case Lösung AJAX Session/Idle Timeout bei PAI für clientseitige Frameworks

Es besteht keine Möglichkeit eine allgemeingültige Erweiterung von PAI Seite zur Verfügung zu stellen, denn die Implementierungen der serverseitigen Frameworks sind zu unterschiedlich. Etwaige Lösungen müssten für jedes Framework bereitgestellt werden, was eine enorme Ressourcen- und Kostenbelastung zur Folge hätte. Eine Alternative können Multiplattform-Frameworks sein, denn die Anpassung muss nur einmal vorgenommen werden und alle unterstützen Technologien profitieren davon (siehe Abschnitt 7.2.2.1.3).

Hinweis: Die vorgestellte Lösung wurde nicht getestet und daher kann nicht sicher gesagt werden, ob sie funktionsfähig ist, denn ohne einen Prototypen zu implementieren kann der Entwickler sich nicht sicher sein wie die clientseitige AJAX-Engine reagiert, wenn sich die URL der Anwendung ändert. Es kann sein, dass die AJAX-Engine nur mit der initialen URL kommunizieren kann. In diesem Fall würde das Szenario nicht funktionieren und müsste angepasst werden.

7.1.4.4 Weblogin Markup Language

Im PAI Release 3.0.3 wurde die so genannte Weblogin Markup Language (WLML) eingeführt. Es besteht die Möglichkeit den Aufbau des Weblogin mit einem proprietären XML-Format zu beschreiben.

Dadurch ergeben sich neue Möglichkeiten, denn es müssen nicht die Webseiten des Weblogin angezeigt, sondern ein Parser kann bereitgestellt werden, der WLML versteht. Aus den gewonnenen Informationen kann das Login Formular mit den zur Verfügung stehenden grafischen Benutzerelementen erzeugt werden.

Es besteht die Möglichkeit für clientseitige Frameworks eine Lösung bereitzustellen, die mit jedem clientseitigem Framework genutzt werden kann. Das trifft nicht auf serverseitige Frameworks zu, denn die Lösungen müssen speziell auf das Framework zugeschnitten sein.

7.1.4.5 Anderes Sicherheitsprodukt

Das zur Zeit eingesetzte Sicherheitsprodukt SiteMinder kann beliebig durch ein anderes Sicherheitsprodukt ersetzt werden. Deshalb wurden der SUN Access Manager ([SUNAcc]) und IBM Tivoli Access Manager ([IBMTivo]) betrachtet.

Diese Produkte sind jedoch keine Alternative zu SiteMinder, denn beide Produkte arbeiten nach dem gleichen Prinzip wie SiteMinder und installieren einen Agent auf dem Webserver. Es entstehen beim Einsatz die gleichen Probleme, die bereits in Abschnitt 7.1.3 angesprochen wurden.

7.1.5 Zusammenfassung

Die verschiedenen Lösungskonzepte für das Session/Idle Timeout Problem mit AJAX wurden im Laufe des Abschnittes vorgestellt. Einige dieser Lösungsansätze, wie die Re-Implementierung des XMLHttpRequest-Objektes und der Umstieg auf ein anderes Sicherheitsprodukt konnten direkt ausgeschlossen werden.

Außerdem stellt keines der im Rahmen dieser Diplomarbeit betrachteten Frameworks (siehe Abschnitt 5.2.6.1) Schnittstellen zur Erweiterung des Frameworks bereit. Deshalb sollten die in Abschnitt 7.1.4.3 beschriebenen Ansätze verfolgt werden. Wie bereits erwähnt wurde, machen die verschiedenen Frameworkmodelle die Bereit-stellung eines einheitlichen Ansatzes unmöglich. Für clientseitige Frameworks kann über Frameworkgrenzen hinweg eine Implementierung geschaffen werden. Das ist für serverseitige Frameworks unmöglich und für jedes unterstützte Framework müsste eine separate Implementierung zur Verfügung gestellt werden. Multiplattform-Frameworks bieten wesentlich bessere Möglichkeiten, denn durch die Unterstützung eines Frameworks können viele Programmierparadigmen abgedeckt werden.

Abschließend ist zu bemerken, dass je integrierter und abstrakter die Frameworks sind, der Aufwand zur Bereitstellungen einer Lösung aufwändiger und kostenintensiver wird. Doch durch die Bereitstellung einer Lösung durch PAI steigen die Integrationskosten aus PAI Sicht zwar erheblich, aber im Gegenzug sinken die Kosten aus Projektsicht.

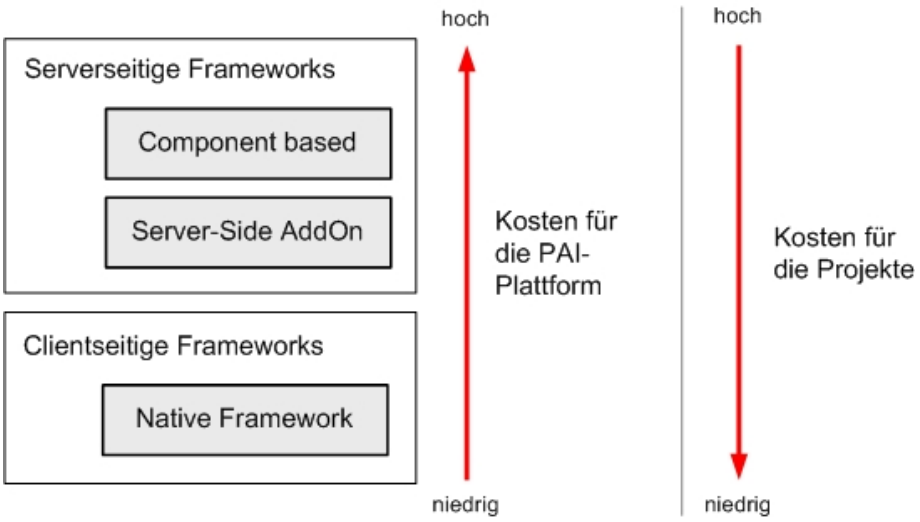

Abbildung 59: Integrationskosten der verschiedenen Frameworkmodelle

7.2 Client Container

Die verschiedenen Frameworkmodelle wurden in Abschnitt 5.2.5 hinsichtlich ihrer Architektur und der sich daraus ergebenen Vorteile bzw. Probleme ausführlich betrachtet. Bei der Integration dieser Frameworkmodelle in die PAI Infrastruktur trat ein Problem auf, das in Abschnitt 7.1 angesprochen wird.

Ähnliche Problematiken ergaben sich bei der Integration des Rich Client Paradigmas. Aus diesem Grund wurde der so genannte PAI Client Container entwickelt, der in diesem Abschnitt vorgestellt wird.

Basierend auf den Erkenntnissen wird betrachtet, ob es Sinn macht den PAI Client Container mit AJAX nachzubilden bzw. einige Komponenten zu übernehmen, um die Integrationsprobleme der verschiedenen Frameworkmodelle zu lösen.

7.2.1 Einführung in den PAI Client Container

Im folgenden Abschnitt werden die Architektur und grundlegenden Komponenten des PAI Client Containers ([CLIENTCONT]) vorgestellt.

Die Verteilung der Anwendung zwischen Client und Server ist in *Abbildung 60* dargestellt. Im PAI Client Container laufen die clientseitigen Bestandteile der Anwendung, während die serverseitigen Bestandteile in einem Application-Server ausgeführt werden. Der PAI Client Container stellt dem clientseitigen Teil der Anwendung verschiedene Komponenten bereit, um z.B. Benutzerinformationen abzurufen und somit den Zugriff auf die Anwendung nur für eine bestimmte Benutzergruppe zuzulassen.

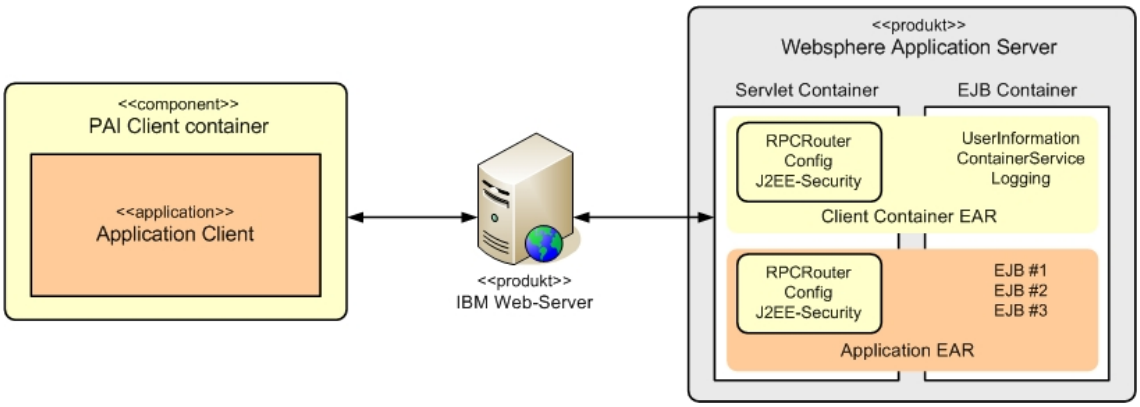

Abbildung 60: Laufzeitumgebung des PAI Client Container

7.2.1.1 Vorstellung der Komponenten

Die vom PAI Client Container zur Verfügung gestellten Komponenten sind in *Abbildung 61* zu sehen und werden anschließend vorgestellt.

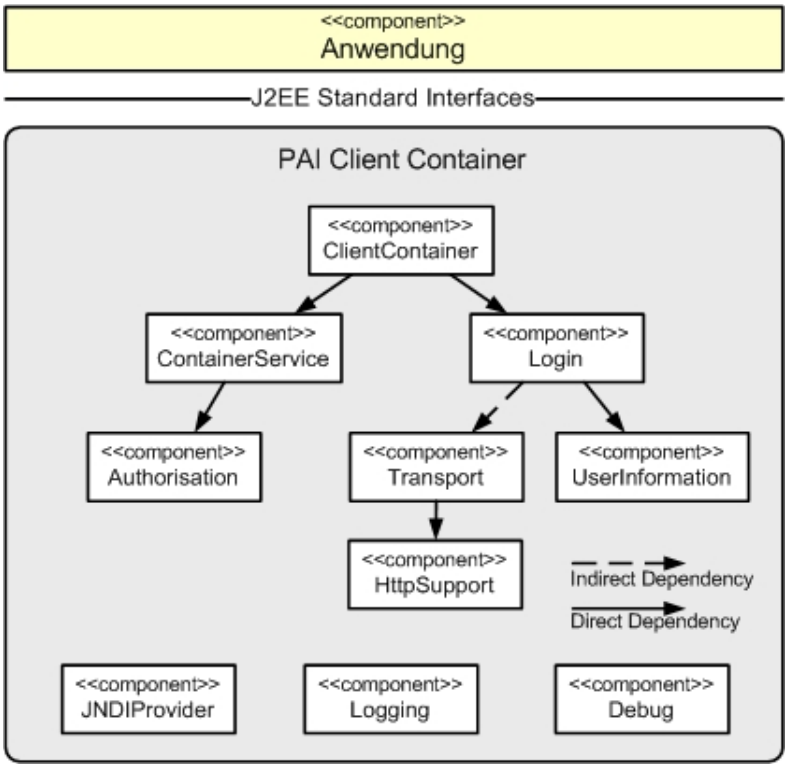

Abbildung 61: Komponenten des PAI Client Container

UserInformation: Die UserInformation Komponente stellt die verschiedensten Benutzerspezifischen Daten zur Verfügung (z.B. Name, Ort, PLZ, Abteilung). Diese Daten werden typischerweise aus einem Direktory geladen, können aber auch in einer Datei zur Verfügung gestellt werden.

Containerservice: Die Containerservice Komponente hat die Aufgabe, die Startberechtigung des Benutzers, die Autorisierungsinformationen der startfähigen Anwendungen und die Konfiguration der Anwendung zu lesen.

Authorisation: Damit den Anwendungen verschiedene Sicherheitsmechanismen zur Verfügung gestellt werden können, wird der Java Authentication and Authorisation Service (JAAS) eingesetzt. Dadurch ist es möglich die Authentifizierungsmodule leicht auszuwechseln und somit die Authentifizierungsinformationen von verschiedenen Dienstanbietern abzurufen.

Clientcontainer: Der Clientcontainer stellt die Laufzeitumgebung für Anwendungen zur Verfügung, die im PAI Client Container ausgeführt werden sollen. Er stellt eine dem J2EE Client Container konforme Umgebung und einen JNDI Namensraum sowie die notwendigen Autorisierungs- und Authentisierungsmechanismen für Clientanwendungen bereit.

Der PAI Client Container ist eine Erweiterung des J2EE Application Client Containers. Im J2EE Application Client Container wird der Zugriff auf serverseitige EJBs mittels IIOP realisiert und ist deshalb nur für Anwendungen geeignet, die auf das Intranet beschränkt sind. Der PAI Client Container sollte es ermöglichen Anwendungen zu entwickeln, die über das Internet operieren können und die Ausführung von mehreren parallelen Anwendungen unterstützen. Diese Anforderungen können jedoch nicht mit dem J2EE Application Client Container realisiert werden.

Anwendung: Eine Anwendung die im PAI Client Container lauffähig sein soll, besteht aus zwei Bestandteilen, dem clientseitige Teil der Anwendung, der der J2EE Spezifikation ([J2EE]) und den Gegebenheiten des PAI Client Containers entspricht. Der Client ist dabei in eine jar Datei verpackt und der Einstiegspunkt in die Anwendung wird in der „application.xml“ Datei angegeben. Diese Datei enthält die Konfiguration der Anwendung und soll an dieser Stelle nicht besprochen werden ([CLIENTCONT]).

Der serverseitige Bestandteil der Anwendung stellt sämtliche Dienste in Form von stateless Session Beans zur Verfügung und kann außerdem die abhängigen jar Dateien beinhalten. Sämtliche Web- und EJB Module müssen in einem Application-Server ausführbar sein. Eine Anwendung wird in Form eines EAR Datei ausgeliefert.

Transport: Die Kommunikation zwischen Client und Server wird mit Hilfe der Transport API realisiert. Dadurch kann die Verbindung über die verschiedensten Kanäle aufgebaut werden. Damit die Verbindung über das Internet realisiert werden kann, stellt der PAI Client Container neben der RMI Komponente zusätzlich eine HTTP Komponente zur Verfügung.

Login: Die Login Komponente stellt ein PAI konformes Login Formular bereit, das von den Anwendungen genutzt werden kann.

Logging: Das Logging EJB stellt einen durchgängigen Mechanismus bereit, um das Loggen von Daten zwischen client- und serverseitigem Bestandteil der Anwendung zu ermöglichen.

7.2.1.2 Eigenschaften

Einige ausgesuchte Eigenschaften des PAI Client Containers werden in diesem Abschnitt vorgestellt. Auf eine detaillierte Einführung wird jedoch an dieser Stelle verzichtet und auf die Dokumentation verwiesen ([CLIENTCONT]).

Mehrere Anwendungen: Im PAI Client Container können mehrere Anwendungen ausgeführt werden. Damit Versionskonflikte zwischen den verschiedenen Anwendungen und von ihnen genutzten Bibliotheken vermieden werden, wird jede Anwendung von einem separaten Classloader geladen und die Ressourcen bzw. Klassen werden in separaten Ordnern vorgehalten. Das Starten einer Anwendung erfolgt durch den Aufruf der main() Methode der Startklasse.

Sicherheit: Der Authentifizierungsmechanismus basiert, wie bereits in Abschnitt 7.2.1.1 angesprochen wurde, auf JAAS. Er stellt Login Module bereit, die HTTP Basic und Digest Authentikation unterstützen. Es werden lediglich der Benutzername und Passwort abgefragt, die eigentliche Authentifizierung übernimmt SiteMinder oder WAS. Zusätzlich ist es möglich die Startberechtigung der Anwendung zu überprüfen. Nach dem Start der Anwendung ist es möglich bestimmte Funktionen der Anwendung nur bestimmten Benutzern zugänglich zu machen. Die Anwendung kann diese Information direkt beim PAI Client Container erfragen, ohne sich mit dem Server zu verbinden.

7.2.2 Client Container mit AJAX

In diesem Abschnitt werden mögliche Lösungen vorgestellt, wie der PAI Client Container mit AJAX abgebildet werden könnte. Die besprochen Ansätze werden hinsichtlich ihrer Architektur, der Unterschiede zum PAI Client Container und der resultierenden Vorteile bzw. Probleme vorgestellt.

7.2.2.1 Architektur

In diesem Abschnitt werden mögliche Lösungen vorgestellt, um die Integrations-probleme der unterschiedlichen Frameworkmodelle zu bewältigen.

7.2.2.1.1 Clientseitige Frameworks

Beim Einsatz von clientseitigen Frameworks (siehe Abschnitt 5.2.5.1) sind die client- bzw. serverseitigen Bestandteile der Anwendung entkoppelt, d.h. die AJAX-Anwendung kann die verschiedensten Dienste clientseitig nutzen. Aus diesem Grund ist es nur schwer möglich den Client in die PAI Sicherheitsinfrastruktur zu integrieren.

In diesem Abschnitt wird ein Ansatz (siehe *Abbildung 62*) vorgestellt, wie Komponenten des PAI Client Containers clientseitig zur Verfügung gestellt werden. Der Ansatz nutzt dabei die serverseitigen Komponenten des PAI Client Containers und stellt JavaScript Schnittstellen für sie bereit.

Zur Realisierung wird das Direct Web Remoting (DWR) Framework eingesetzt. DWR ermöglicht die Generierung von JavaScript Interfaces, die auf Klassen abgebildet

werden, die in Java implementiert sind und auf dem Server ablaufen. Das Mapping der entfernten Aufrufe übernimmt das Framework und bildet das Ergebnis wieder auf JavaScript Schnittstellen ab. Auf eine nähere Einführung in DWR wird an dieser Stelle verzichtet und auf die Dokumentation ([DWR]) verwiesen.

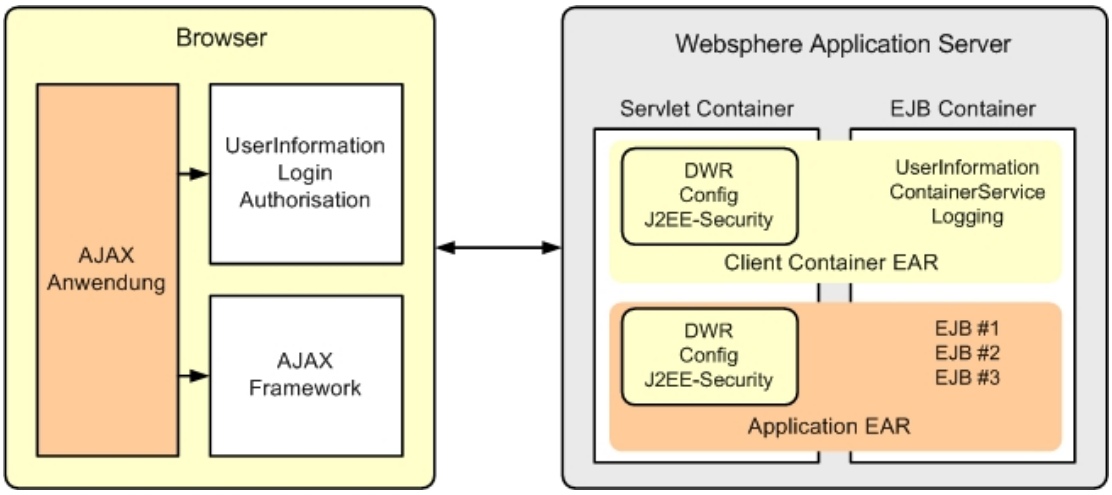

Abbildung 62: Aufbau Client Container für clientseitige Frameworks

Als Ergänzung zu dem clientseitig genutztem Framework stehen Bibliotheken zur Verfügung, die einen Zugriff auf Komponenten des Client Containers ermöglichen (Benutzerinformationen, Autorisation, Logging, Login). Neben den zur Verfügung gestellten Komponenten könnten weitere anwendungsspezifische Komponenten erzeugt und genutzt werden.

7.2.2.1.2 Serverseitige Frameworks

Da serverseitige Frameworks (siehe Abschnitt 5.2.5.2) auf verschiedenen serverseitigen Technologien basieren und somit in die Sicherheitsinfrastruktur integriert sind, können die serverseitigen Bestandteile des PAI Client Containers (z.B. Benutzerinformationen, Autorisation, Logging) bereits eingesetzt werden.

Es ist aber sinnvoll eine Login Komponente bereitzustellen, die das in Abschnitt 7.1 angesprochene Problem löst.

7.2.2.1.3 Multiplattform-Frameworks

Multiplattform-Frameworks (siehe Abschnitt 5.2.5.3) vereinen die Eigenschaften von client- und serverseitigen Frameworks, denn sie stellen eine Schnittstelle zur clientseitigen Programmierung und eine oder mehrere Schnittstellen zur serverseitigen Programmierung bereit. In diesem Abschnitt wird deshalb ein Ansatz vorgestellt, wie Multiplattform Frameworks integriert werden können.

Die Grundidee des gezeigten Ansatzes (siehe *Abbildung 63*) besteht darin, das Framework um zusätzliche clientseitige Komponenten zu erweitern, die eine Integration in die

PAI Sicherheitsinfrastruktur ermöglichen. Durch die Bereitstellung dieser Komponenten ist es möglich, die verschiedenen serverseitigen Implementierungen um zusätzliche Funktionalität zu erweitern und somit eine Lösung bereitzustellen, die verschiedene Programmierparadigmen abdeckt.

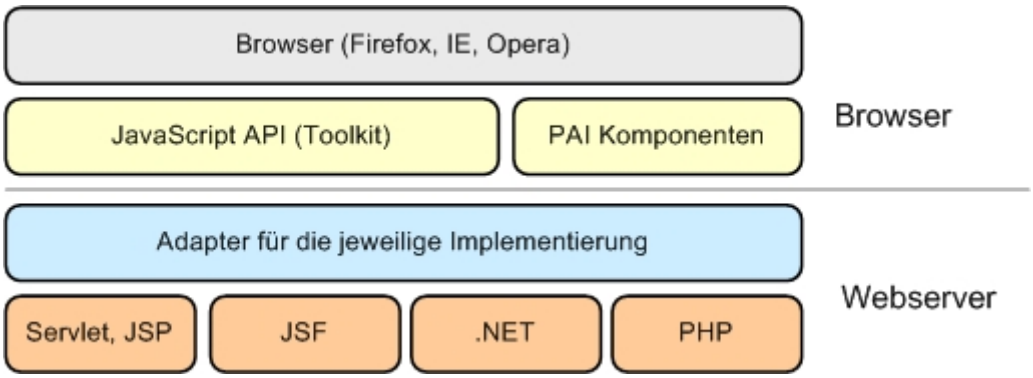

Abbildung 63: Client Container mit Multiplattform-Frameworks

7.2.2.2 Unterschiede

Im bestehenden PAI Client Container sind sämtliche Bestandteile in Java realisiert. Bei clientseitigen Frameworks verhält es sich anders, denn die clientseitigen Bestandteile sind in JavaScript und die serverseitigen Bestandteile in Java realisiert.

Außerdem wird die Laufzeitumgebung durch den Browser bereitgestellt und einige Komponenten stehen bereits zur Verfügung. Dies sind das Transportprotokoll und Debugging. Das Transportprotokoll basiert auf HTTP(s) und wird bereits vom Browser standardmäßig eingesetzt. Der Browser stellt bereits grundlegende Debugging Funktionalitäten zur Verfügung und kann durch Plugins erweitert werden.

Im Gegensatz zum PAI Client Container, kann in einem Browserfenster nur eine Anwendung ausgeführt werden und aus diesem Grund kann auf die Funktionen mehrere Anwendungen auszuführen verzichtet werden.

Beim Einsatz von Multiplattform-Frameworks können mehrere Programmier-paradigmen mit einem Framework abgedeckt werden.

7.2.2.3 Vorteile und Probleme

Prinzipiell lehnen sich die Vorschläge an die Architektur des PAI Client Containers an und versuchen möglichst viele Komponenten weiter zu benutzen, um eine Neu-implementierung zu vermeiden.

Bedingt durch die Programmiersprache JavaScript ergibt sich ein erhebliches Manko, denn in JavaScript können keine gleichnamigen Funktionen mit unterschiedlichen Parametern erzeugt werden. Dadurch müssen diese Funktionen umbenannt und durch das Mapping auf die richtige Funktion abgebildet werden. Das verändert die Schnitt-stelle erheblich, denn sie müsste an die Gegebenheiten angepasst werden.

Der Einsatz von Multiplattform-Frameworks ermöglicht die Unterstützung verschiedener Programmierparadigmen, aber bei jedem Update des Basisframeworks müssen alle höherwertigen Implementierung nachgezogen werden und aus diesem Grund auch alle PAI spezifischen Erweiterungen.

7.2.2.4 Zusammenfassung

In diesem Abschnitt werden die Architektur und die Komponenten des PAI Client Containers vorgestellt.

Basierend auf diesen Erkenntnissen werden verschiedene Möglichkeiten vorgestellt, wie client- und serverseitige Frameworks bzw. Multiplattform-Frameworks besser in die PAI Sicherheitsinfrastruktur integriert werden können. Weiterhin werden die Unterschiede zwischen dem PAI Client Container und den mit AJAX realisierten Varianten, sowie die sich daraus ergebenden Vorteile und Probleme angesprochen.

Abschließend ist zu sagen, dass es keinen Sinn macht den PAI Client Container mit AJAX abzubilden, denn die angesprochenen Gegebenheiten lassen eine vollständige Re-Implementierung des PAI Client Containers in JavaScript nicht zu. Doch je nach Frameworkmodell können verschiedene Komponenten portiert werden und somit dem Entwickler das entwickeln von PAI konformen AJAX-Anwendungen erheblich erleichtern.

7.3 Entwicklung einer Beispielanwendung

Zur Implementierung einer Beispielanwendung wurde Echo2 ausgewählt, die Gründe die zu dieser Entscheidung führten wurden bereits in Abschnitt 5.2.8 aufgeführt.

7.3.1 Hinter den Kulissen von Echo2

Dieser Abschnitt beschäftigt sich ausführlicher mit dem Echo2 Framework und folgende Punkte werden besprochen:

- das detaillierte Frameworkmodell von Echo2 und die Funktionsweise der Bestandteile.
- die Synchronisation zwischen den client- und serverseitigen Komponenten anhand eines Beispiels

7.3.1.1 Detailliertes Frameworkmodell

Das Echo2 Frameworkmodell ist in *Abbildung 64* dargestellt und besteht aus den folgenden Modulen:

- Web Application Container
- Allication Framework
- Web Rendering Engine

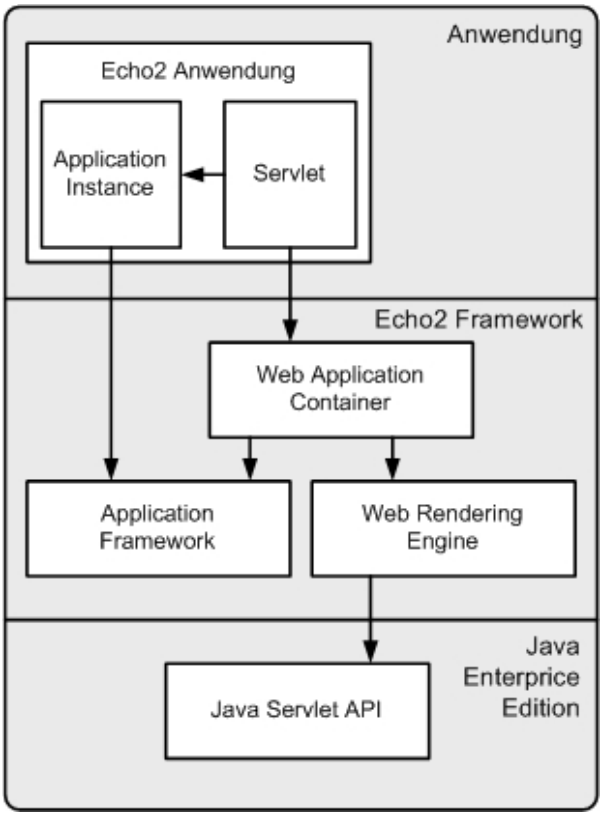

Abbildung 64: Architektur vom Echo2 Framework

Die einzelnen Module sind so ausgelegt, das möglichst wenig Anhängigkeiten zwischen den verschiedenen Bestandteilen entstehen. Eine Ausnahme ist hier der Web Application Container, der vom Application Framework und der Web Rendering Engine abhängig ist, aber nicht umgekehrt.

7.3.1.1.1 Application Framework

Das Application Framework stellt dem Entwickler die höherwertigen Komponenten und Listener zur Verfügung, um Anwendungen zu entwickeln. Diese Komponenten beinhalten keinerlei HTML und JavaScript Quellcode, sondern sind ausschließlich in Java programmiert. Dadurch könnte das Application Framework in einem Szenario eingesetzt werden, indem die Benutzerschnittstelle basierend auf Flash, Swing oder SWT erzeugt wird. Damit die sichtbaren Teile der Benutzerschnittstelle mit dem auf dem Server befindlichen Komponenten synchronisiert und ihr Status aktualisiert werden kann, existiert der UpdateManager.

7.3.1.1.2 Web Rendering Engine

Die Web Rendering Engine übernimmt die Kommunikation zwischen dem client- und serverseitigen Teil der Anwendung. Sie besteht aus einem serverseitigen Anteil, der in Java geschrieben ist und auf der Servlet API basiert. Sie besteht prinzipiell aus einem HttpServlet, das Clientanfragen entgegen nimmt und mit Hilfe der „serviceId“ an einen

bestimmten Service weiterleitet. Echo2 stellt die verschiedensten Services bereit, z.B. generieren der HTML Benutzerschnittstelle, bereitstellen von JavaScript Modulen, Bildern und erstellen von XML Nachrichten, um partielle DOM Aktualisierungen auf dem Client zu ermöglichen.

Die clientseitige Engine ist in JavaScript implementiert und nutzt das XMLHttpRequest-Objekt für die Kommunikation mit den serverseitigen Bestandteilen. Sie ist ausschließlich für die Synchronisation von Client und Server zuständig, wenn Benutzeraktionen erfolgen und sich der Zustand der Anwendung ändert. Außerdem werden verschiedene APIs zur Verfügung gestellt, um mit dem Web-Browser plattformunabhängig zu interagieren.

Die Web Rendering Engine stellt eine generische API dar, um den client- bzw. serverseitigen Bestandteile der Anwendung zu synchronisieren. Die Komponente wurde so entwickelt, dass keinerlei Abhängigkeiten zum Application Framework exisitieren.

7.3.1.1.3 Web Application Container

Der Web Application Container erweitert die Web Renderng Engine und stellt Möglichkeiten zur Verfügung, um die client- bzw. serverseitigen Komponenten der Anwendung zu synchronisieren. So enthält er z.B. die JavaScript Bestandteile der Komponenten und den Mechanismus, der die client- bzw. serverseitigen Bestandteile synchronisiert.

7.3.1.2 Client und Server Synchronisation

Der zentrale Einstiegspunkt in Echo2 Anwendungen ist ein Dispatcher-Servlet. Ein Beispiel für den Aufruf (rot markiert) ist in *Abbildung 65* zu sehen.

Der Quellcode der initialen Webseite ist in *Abbildung 66* dargestellt. Es wird lediglich die AJAX-Engine an den Client übertragen und durch den Aufruf der Methode „init“ (rot markiert) wird die Anwendung gestartet.

Nachdem diese Methode erfolgreich ausgeführt wurde, sendet die AJAX-Engine das Kommando „Echo.Synchronize“ an den Server. Dieses Kommando bewirkt, dass sich die client- und serverseitigen Bestandteile der Anwendung synchronisieren.

```
GET http://localhost:8124/app HTTP/1.1 => HTTP/1.1 200 OK
GET http://localhost:8124/app?serviceId=Echo.ClientEngine HTTP/1.1 => HTTP/1.1 200 OK
POST http://localhost:8124/app?serviceId=Echo.Synchronize HTTP/1.1 => HTTP/1.1 200 OK
GET http://localhost:8124/app?serviceId=Echo.WebContainer HTTP/1.1 => HTTP/1.1 200 OK
GET http://localhost:8124/app?serviceId=Echo.TextComponent HTTP/1.1 => HTTP/1.1 200 OK
GET http://localhost:8124/app?serviceId=Echo.ContentPane HTTP/1.1 => HTTP/1.1 200 OK
GET http://localhost:8124/app?serviceId=Echo.Button HTTP/1.1 => HTTP/1.1 200 OK
```

Abbildung 65: Beispiel für das Nachladen der Komponenten durch die AJAX-Engine

Die an den Client übertragenen Daten sind in *Abbildung 65* blau markiert. Hier ist noch zu erwähnen, dass die benötigten Komponenten durch das Attribut „serviceId“ identifiziert werden.

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
        ……..
       <script src="/app?serviceId=Echo.ClientEngine" type="text/javascript"></script>
   </head>
   <body onload="EchoClientEngine.init('/app', true);" style="……. ">
        <form action="#" id="c_root" onsubmit="return false;" style="……">
            <div id="loadstatus">//</div>
        </form>
    </body>
</html>
```

Abbildung 66: Initiale Webseite einer Echo2 Anwendung

Die nun im Browserfenster angezeigte Anwendung enthält, ein Label, Textfeld und einen Button (siehe *Abbildung 67*: links).

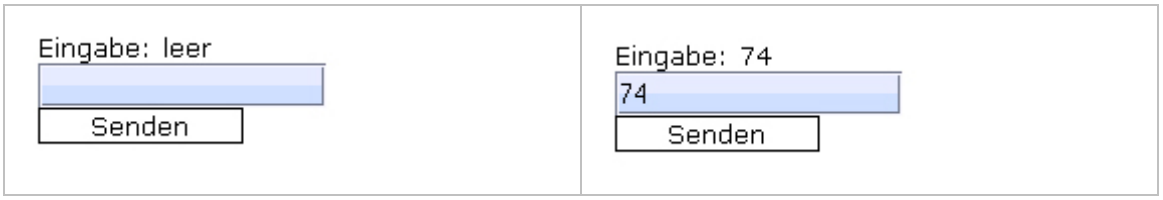

Abbildung 67: Beispielanwendung zur Client und Server Synchronisation Echo2

Wenn der Benutzer einen Wert in das Textfeld eingibt und das Formular absendet, wird eine XML-Nachricht (siehe *Abbildung 68*) an den Server geschickt. Der Aufbau dieser Nachricht wird im Fall von Echo2 durch ein proprietäres XML Protokoll festgelegt.

```
<client-message xmlns="http://www.nextapp.com/products/echo2/climsg" trans-id="1" focus="c_4">
   <message-part processor="EchoPropertyUpdate">
       <property component-id="c_4" name="text">74</property>
   </message-part>
   <message-part processor="EchoAction">
       <action component-id="c_5" name="click"/>
   </message-part>
</client-message>
```

Abbildung 68: XML Nachricht vom Client an den Server zur Aktualisierung des Modells

Die aktualisierten serverseitigen Komponenten können ihrerseits Events auslösen und Listener benachrichtigen. Im hier dargestellten Beispiel, ist der Button mit einem ActionListener verknüpft, der in das Label die eingegebene Zahl schreibt.

```
<?xml version="1.0" encoding="UTF-8"?>
<server-message xmlns="http://www.nextapp.com/products/echo2/svrmsg/servermessage"
                async-interval="disable" modal-id="" trans-id="2">
    <message-part-group id="preremove"/>
    <message-part-group id="remove">
        <message-part processor="EchoDomUpdate.MessageProcessor">
            <dom-remove target-id="c_3"/>
        </message-part>
    </message-part-group>
    <message-part-group id="update">
        <message-part processor="EchoDomUpdate.MessageProcessor">
            <dom-add>
                <content parent-id="c_2_cell_c_3">
                    <span id="c_3" xmlns="http://www.w3.org/1999/xhtml">Eingabe: 74 </span>
                </content>
            </dom-add>
        </message-part>
    </message-part-group>
    <message-part-group id="postupdate"/>
</server-message>
```

Abbildung 69: XML Nachricht vom Server an den Client zur Aktualisierung des Modells

Der Server erzeugt daraufhin eine „server-message“, um den Status der clientseitigen Komponenten (siehe *Abbildung 69*) zu aktualisieren.

Die aktualisierten Bestandteile der Benutzerschnittstelle werden auf dem Server bereits generiert und clientseitig in den DOM eingefügt. Das aktualisierte Label wird zuerst aus dem DOM gelöscht „dom-remove“, um dann durch den neuen Inhalt ersetzt zu werden „dom-add“ (siehe *Abbildung 67*: rechts).

7.3.2 Admin Control Center Beispielanwendung

In Abschnitt 5.2 wurden verschiedene AJAX-Frameworks vorgestellt. Aus diesen Frameworks wurde Echo2 ausgewählt, um eine Beispielanwendung zu implementieren. Diese Anwendung realisiert Teile des Admin Control Center (ACC) mit Echo2.

Sie wird nicht detailliert besprochen, sondern nur die Schwächen des ACC aufgezeigt und wie sie durch den Einsatz von AJAX verbessert werden kann.

7.3.2.1 Vorstellung des ACC

Das ACC ist eine Webanwendung, um die Inhalte eines Direktory zu verwalten. Auf den grundlegenden Aufbau wird nicht eingegangen und auf die Dokumentation verwiesen ([NUTSHELL]). Aber prinzipiell werden Benutzer-, Anwendungs- und Organisationsdaten verwaltet, die eine Anwendung nutzen kann. Diese hierarchischen Strukturen können mit einander verknüpft sein und sind somit teilweise abhängig voneinander. Der Zugriff erfolgt über die Admin-API, die als Anwendung auf dem Webserver installiert ist.

7.3.2.2 Schwächen des ACC

Der Aufbau der Benutzeroberfläche ist in *Abbildung 70* zu sehen. Das Problem, von generell allen Webanwendungen ist, dass bei jeglicher Aktion, die der Benutzer ausführt, die Seite komplett neu geladen wird. Diese seitenbasierte Kommunikation hat zur Folge, dass sehr viel redundanter Inhalt übertragen wird und durch die Blockierung der Benutzerschnittstelle der Eindruck entsteht, dass die Anwendung unperformant ist.

Außerdem stehen in HTML nur sehr rudimentäre Benutzerelemente zur Verfügung und jegliche Funktionalität, die darüber hinaus geht, muss selber implementiert werden.

All diese Probleme wirken sich negativ auf eine Anwendung aus, die dazu gedacht ist ein Direktory mit seinen unzähligen Datensätzen zu verwalten, denn es müssen Möglichkeiten geschaffen werden einfach durch diese Menge zu navigieren und die Beziehungen zwischen den Datensätzen anzuzeigen.

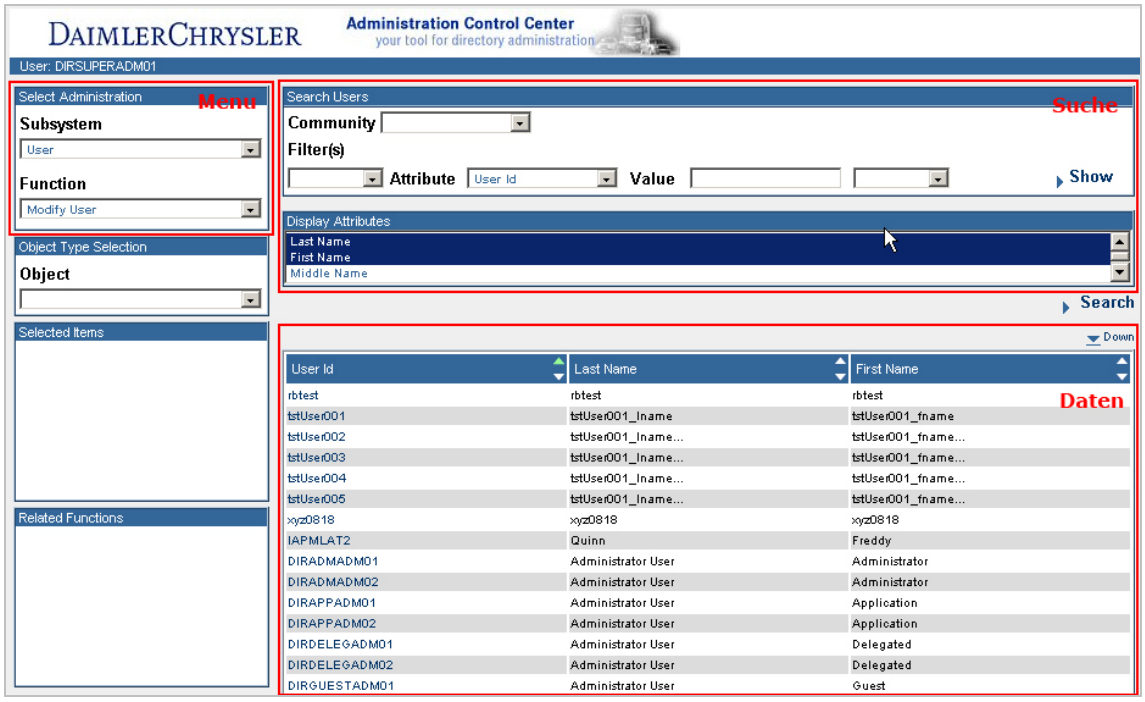

Abbildung 70: Benutzerschnittstelle ACC aktuelle Version

Deshalb hat es Sinn gemacht, durch ausgiebiges Testen die Schwächen (siehe *Tabelle 7*) des ACC herauszufinden.

Schwachstelle	Beschreibung
(1) Menu	Der Benutzer muss zuerst das gewünschte Subsystem auswählen und danach die gewünschte Funktion, anschließen wird die Webseite aktualisiert und der neue Inhalt angezeigt.
(2) Verschiedene Reiter	Wenn ein neuer Menupunkt angewählt wird, wird der bisherige Inhalt überschrieben. Was geschieht wenn der überschriebene Inhalt wichtig war? Der Benutzer hat nur die Möglichkeiten mit verschiedenen Browserfenstern zu arbeiten oder die Suche neu auszuführen.
(3) Suche	Bei der Suche bekommt der Benutzer keinerlei Informationen, in wie weit sich die Ergebnismenge ändert, wenn er neue Suchfilter hinzufügt und ob mit der jeweiligen Filterzusammenstellung ein Suchergebnis erwartet werden kann. Es kann sein, dass ein Filter zusammengestellt wird und keine Einträge gefunden werden. Außerdem besteht keine Möglichkeit den Filter wieder herzustellen oder zu merken.
(4) Anzeige der Daten und Navigation	Durch große Ergebnismengen wird die Navigation unübersichtlich, da zu viele Datensätze angezeigt werden, was extra noch erhebliche Ladezeiten bedeutet.
(5) Daten bearbeiten	Das Bearbeiten von Daten ist sehr rudimentär, denn dem Benutzer stehen unzureichende Möglichkeiten bereit direkt Daten in Listen zu bearbeiten oder einzufügen.
(6) Inhalte zwischenspeichern	Dem Benutzer steht kein Container zur Verfügung, um mehrere Datensätze (z.B. Organisationen, Anwendungen, Benutzer) zwischenzuspeichern und zu einem späteren Zeitpunkt wieder darauf zurück zugreifen.

Tabelle 7: Übersicht der ausgearbeiteten Schwächen des ACC

7.3.2.3 Verbesserungen durch Echo2

Basierend auf den in Abschnitt 7.3.2.2 angesprochenen Schwächen des ACC, konnten Komponenten identifiziert werden, die die grundlegenden Funktionen des ACC abbilden und somit die Entwicklung erheblich erleichtern. Diese wieder verwendbaren Komponenten basieren durchgehend auf dem MVC Muster und stellen Schnittstellen bzw. Listener zur Interaktion bereit. Dadurch werden sie wieder verwendbarer, flexibler und die Wartbarkeit bzw. Skalierbarkeit der Anwendung wird entscheidend verbessert.

Komponente	Beschreibung
(1) Menu	Diese Komponente wird von der API bereitgestellt und ermöglicht die Erzeugung einer Menubar.
(2) TabPane	Sie ermöglicht das Anzeigen von verschiedenen Tabs in einer TabPane, dadurch können verschiedene Inhalte durchgeblättert werden. Die Komponente wird ebenfalls von der API zur Verfügung gestellt.
(3) SearchPanel	Es stellt ein Suchformular zur Verfügung, um Suchfilter zu erzeugen. Die Filter können importiert bzw. exportiert werden, dadurch kann das Suchformular basierend auf einem Filter wiederhergestellt werden. Das Modell dieser Komponente kann konfiguriert werden, da sich die Suchkriterien z.B. bei Benutzern und Organisationen ändern.
(4) NavigatableTable	Diese Komponente wird von der API zur Verfügung gestellt und ermöglicht es eine Tabelle zu erstellen, durch die bequem navigiert werden kann. Es werden verschiedene Funktionen unterstützt, z.B. chronologisches blättern, Anzahl der Datensätze pro Seite festlegen, Sortierung von Tabellenspalten und die direkte Anwahl von Seiten.
(4) InfoPanel	Das InfoPanel zeigt dem Benutzer verschiedene allgemeine Informationen an (z.B. die Anzahl der gefundenen Datensätze).
(5) InfoDialog	Mit Hilfe dieser Komponente können die verschiedensten Datensätze (z.B. Benutzer, Organisationen) angezeigt werden. Eine Erweiterung dieser Komponente ist sinnvoll, um z.B. einen generischen Editor für Datensätze anzubieten.
(6) DataPanel	Das DataPanel ermöglicht es die verschiedensten Datensätze zu speichern. Um diese Datensätze zu speichern, muss ein Adapter für den jeweiligen Typ bereitgestellt werden. Außerdem ist eine beispielhafte Implementierung von Drag&Drop enthalten.

Tabelle 8: Übersicht der entwickelten Komponenten des ACC

7.3.2.4 Das ACC mit Echo2

Die im vorherigen Abschnitt vorgestellten Komponenten werden dazu verwendet einen Teil des ACC mit Echo2 (siehe *Abbildung 71*) zu implementieren.

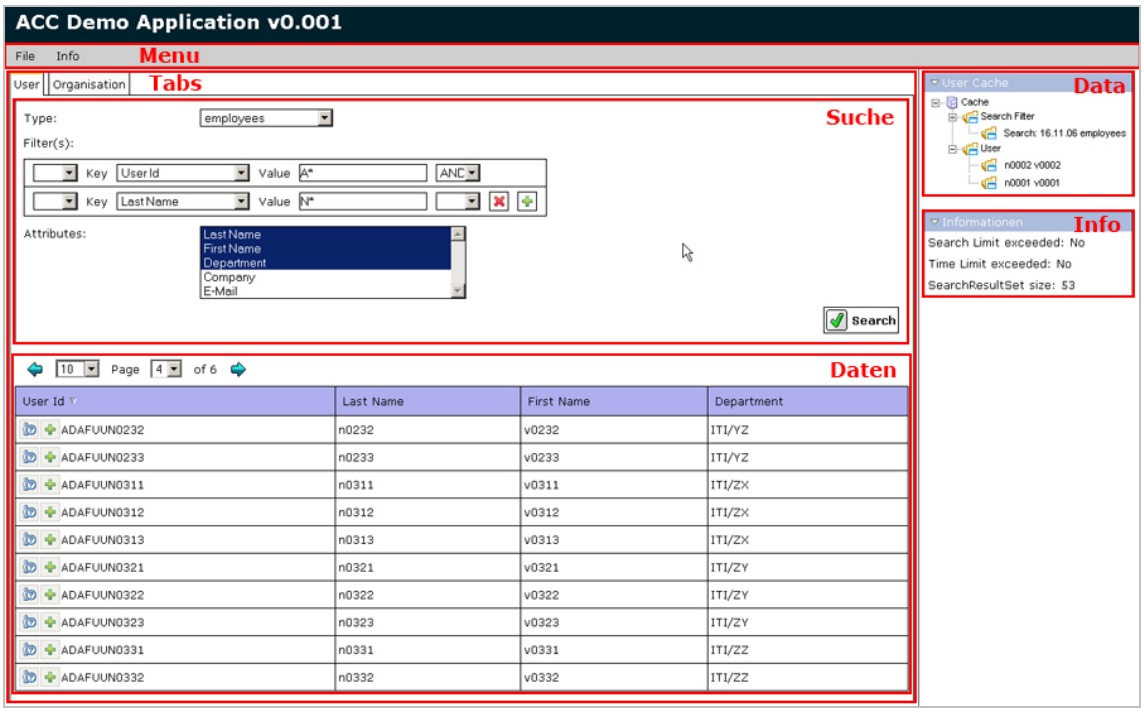

Abbildung 71: ACC Beispielanwendung

Durch den Einsatz der beschriebenen Komponenten kann die Bedienbarkeit der Benutzeroberfläche entscheidend verbessert werden. Der Benutzer kann wesentlich einfacher und komfortabler durch große Datenmengen (NavigatableTable) navigieren und bestimmte Objekte im DataPanel speichern und zu einem späteren Zeitpunkt wieder verwenden. Weiterhin hat er die Möglichkeit mit Hilfe von verschiedenen Tabs (TabPane) zwischen z.B. Benutzer- und Organisationsdaten hin und her zu schalten bzw. unabhängige Suchergebnisse anzuzeigen.

7.3.3 Zusammenfassung

Dieser Abschnitt befasst sich detailliert mit dem Echo2 Framework und der mit Hilfe des Frameworks realisierten Beispielanwendung.

Es wird aufgezeigt, aus welchen Modulen das Framework aufgebaut ist und wie sie zusammenarbeiten bzw. funktionieren. Außerdem wird mit Hilfe eines Beispiels erklärt, wie sich die client- und serverseitigen Bestandteile der Komponenten synchronisieren. Dabei werden die relevanten Teile des eingesetzten XML Protokolls angesprochen.

Die Beispielanwendung realisiert einen Teil des in der PAI Plattform eingesetzten ACC. Deshalb werden zuerst die Schwächen des ACC angesprochen und daraufhin die resultierenden Komponenten vorgestellt. Die Komponenten werden in der ACC Beispielanwendung eingesetzt und ermöglichen die einfache und schnelle Erstellung einer komfortableren Benutzerschnittstelle. Des weiteren sind die entstandenen Komponenten sehr leicht wieder verwendbar und zeigen die Vorteile von AJAX-Frameworks auf.

8 Zusammenfassung und Fazit

Das Ziel dieser Diplomarbeit war es zu beleuchten, ob der Einsatz von AJAX in PAI sinnvoll ist. Es wird betrachtet welche Vorteile, Risiken und Herausforderungen beim Einsatz von AJAX auftreten. Durch eine Projektumfrage wird festgestellt, wie die von PAI unterstützen Clientparadigmen eingesetzt wurden, um Rückschlüsse auf die Sinnhaftigkeit von AJAX zu ziehen. Außerdem wurden verschiedene AJAX-Frameworks hinsichtlich ihrer Einsatzgebiete betrachtet. Anschließend wurden die Integrationsprobleme in die PAI Plattform beleuchtet und die im Rahmen der Diplomarbeit entwickelte Beispielanwendung vorgestellt.

Der Einsatz von AJAX ermöglicht es Anwendungen zu entwickeln, die reichhaltige, interaktive und ansprechende Benutzerschnittstellen zur Verfügung stellen. Sie vereinen die Vorteile von normalen Webanwendungen und Desktop-Anwendungen. Dadurch ergeben sich neue Möglichkeiten Anwendungen zu entwickeln, die bisher noch nicht im Internet realisiert werden können. Durch den Einsatz standardisierter Web-Technologien können AJAX-Anwendungen auf einer großen Anzahl von Zielsystemen ausgeführt werden. Die zentrale Installation auf dem Webserver ermöglicht es Anwendungen einfach und global zur Verfügung zu stellen.

Die von PAI unterstützen Clientparadigmen wurden hinsichtlich ihrer Architektur und der sich daraus ergebenden Vorteile bzw. Probleme betrachtet. Mit Hilfe dieser Betrachtungen und allgemeiner Qualitätsanforderungen an Software, wurde ein Fragebogen zur Durchführung einer Projektumfrage entwickelt. Neben den Einsatzgebieten in Softwareprojekten konnte vor allem die Erkenntnis gewonnen werden, dass die von den Projekten gestellten Anforderungen an die Clientparadigmen nicht von den momentan zur Verfügung gestellten Rich und Thin Clientparadigmen erfüllt werden können. Eine weitere und durchaus bedenkliche Feststellung ist, dass AJAX in Softwareprojekten eingesetzt wird, ohne die Problematiken beim Einsatz ausreichend zu erforschen. Hier muss von PAI Seite mehr Aufklärungsarbeit geleistet werden, um den Projekten die Vorteile und entstehenden Probleme bei der Verwendung von AJAX zu vermitteln.

Beim Einsatz von AJAX muss der Entwickler diverse Probleme, wie z.B. Browserinkompatibilitäten, Barrierefreiheit oder Memory Leaks, bewältigen. Diese können durch den Einsatz von Frameworks teilweise behoben werden und somit die Entwicklung von AJAX-Anwendungen erheblich vereinfachen. Aus diesem Grund wurden verschiedene Frameworks betrachtet, um die Vorteile bzw. Probleme im praktischen Einsatz zu beleuchten. Diese Frameworks werden in verschiedene Frameworksmodelle unterteilt. Das sind client-, serverseitige Frameworks und Multiplattform-Frameworks. Clientseitige Frameworks steigern, bedingt durch die Anzahl der beteiligten Technologien (z.B. JavaScript, HTML, CSS), die Komplexität bei der Anwendungsentwicklung enorm und sind durch die Entkopplung von der server-

seitigen Technologie nicht in die PAI Sicherheitsinfrastruktur integriert. Mit serverseitigen Frameworks kann der Entwickler in seinem gewohnten Programmierparadigmen entwickeln. Außerdem sind sie vollständig in die serverseitige Sicherheitsinfrastruktur integriert. Multiplattform-Frameworks stellen Möglichkeiten bereit client- und serverseitig zu programmieren. Sie stellen verschiedene Programmierschnittstellen für die bereits von PAI unterstützen serverseitigen Technologien bereit und sind somit ein guter Kompromiss. Die vorgestellten Frameworks sind jedoch, bedingt durch die von PAI gestellten Anforderungen und die aufgezeigten Schwachstellen, nur in bestimmten Szenarien einsetzbar. Zu erwähnen ist, dass keines der Frameworks die von PAI gestellten Anforderungen erfüllen kann. Aus diesem Grund ist es notwendig verschiedene Frameworks zu unterstützen.

Sicherheit ist vor allem in Enterprise Umgebungen ein wichtigster Punkt, deshalb werden verschiedene Sicherheitsaspekte (z.B. Browsersicherheitsmodelle, beliebte Angriffsmethoden und Möglichkeiten AJAX-Anwendungen abzusichern) betrachtet, die sich besonders auf AJAX-Anwendungen auswirken. In diesem Zusammenhang bleibt anzumerken, dass JavaScript eine große Schwachstelle ist, da z.B. der Quellcode im Klartext übertragen wird. Die Sicherheitsmodelle der Browser können leicht umgangen werden und bieten somit keinen ausreichenden Schutz. Durch den Einsatz von Frameworks und dem wachsenden Know-How bei JavaScript entstehen zusätzliche Angriffsvektoren. Weiterhin stellen die betrachteten Frameworks keine ausreichenden Sicherheitsmaßnahmen zur Verfügung, wodurch momentan AJAX-Anwendungen nicht ausreichend abgesichert werden können.

Bei der Integration von den verschiedenen Frameworkmodellen in PAI bestand die Problematik, dass der Weblogin bei einem Session/Idle Timeout, der SiteMinder-Session im Zusammenspiel mit dem asynchronen Kommunikationsmodell nicht angezeigt wird. Von den beschriebenen Möglichkeiten kommt nur die Lösung in Frage, die automatische Weiterleitung von SiteMinder an die definierte URL zu nutzen. Diese Lösung muss jedoch für die verschiedenen Frameworkmodelle speziell zugeschnitten werden. Basierend auf den beschriebenen Problemen wurden diverse Lösungsmöglichkeiten vorgestellt, um die verschiedenen Frameworkmodelle besser in die PAI Sicherheitsinfrastruktur zu integrieren. Das soll durch die Wiederverwendung verschiedener Komponenten des PAI Client Container erreicht werden.

Bei clientseitigen Frameworks ist es sinnvoll Komponenten bereitzustellen, um eine bessere Integration in die Sicherheitsinfrastruktur zu gewährleisten. Multiplattform-Frameworks bieten die Möglichkeit durch die einmalige Integration der clientseitigen Bestandteile des Frameworks, auch die serverseitigen Bestandteile einfacher zu integrieren. Je abstrakter die Frameworks sind, desto aufwändiger und kostenintensiver wird die Integration in PAI.

Mit Echo2 wurde eine Beispielanwendung implementiert, die Bestandteile des ACC umsetzt und deutlich die Vorteile von AJAX aufzeigt.

Abschließend ist festzustellen, dass diese Arbeit nur eine Momentaufnahme von AJAX darstellt. Es wurde gezeigt welche Probleme beim Einsatz von AJAX in Enterprise Umgebungen auftreten und Lösungsansätze dafür vorgestellt. AJAX ist eine neue Technologie, deren Entwicklung nicht absehbar ist. Die zur Zeit angebotenen Frameworks stellen noch nicht die Mechanismen bereit, um eine einfache Integration in PAI zu ermöglichen.

Aufgrund aller zuvor aufgeführten Probleme, ist eine Integration von AJAX in PAI zum jetzigen Zeitpunkt nicht zu empfehlen.

AJAX wird bereits in verschiedenen Projekten eingesetzt und deshalb sollten die im Rahmen dieser Diplomarbeit gewonnenen Erkenntnisse und aufgeführten Probleme beachtet werden. Dadurch können Softwareprojekte Informationen über Vorteile und Risiken beim Einsatz von AJAX in PAI erhalten.

Anhang A: Schnittstellen und Browserunterstützung

A1: Browserunterstützung

Browser	AJAX Support	Native	per Plugin
IE	5.0	ab 7.0	ja (ActiveX)
Firefox	1.0	ja	-
Opera	8.0	ja	-
Safari	1.2	ja	-
Netscape	7.1	ja	-
Konquerer	3.2	ja	-
iCab	3.0 beta	ja	-

Tabelle 9: AJAX Unterstützung der verschiedenen Browser

A2: API des XMLHttpRequest-Objectes

Funktion	**Beschreibung**
abort()	Den aktuell laufenden Request beenden.
getAllResponseHeaders()	Gibt eine Liste aller vorhandenen Header, als key/value-Paare in einem String zurück.
getAllResponseHeader()	Gibt den Wert für den im Argument angegebenen Header zurück.
open(Method, url [, synflag , user, password])	Der Request wird an die gewünschte Zieladresse gesendet und die Art der Übertragung (synchron/asynchron) vereinbart.
send(body | null)	Sendet den Request an den Server mit einem optionalen Body.
setRequestHeader (key, value)	Einen optionalen Header für den Request als key/value Paar setzen.

Tabelle 10: Methoden des XMLHttpRequest Objectes

<table>
<tr><th>Eigenschaft</th><th>Beschreibung</th></tr>
<tr><td>onreadystatechange</td><td>Event-Handler, der eine zugeordnete Funktion bei jeder Änderung im Status des Request ausführt. Wird in Verbindung mit der Eigenschaft readyState verwendet.</td></tr>
<tr><td>readystate</td><td>Diese Eigenschaft gibt den aktuellen Status des Requests zurück und ändert sich solange, bis das Ende der Anfrage erreicht ist.
<table>
<tr><td>Wert</td><td>Bedeutung</td><td>Beschreibung</td></tr>
<tr><td>0</td><td>uninitialized</td><td>Request wurde noch nicht durch open() ausgelöst</td></tr>
<tr><td>1</td><td>loading</td><td>Request wurde gestartet aber nicht gesendet</td></tr>
<tr><td>2</td><td>loaded</td><td>Request wurde durch send() abgeschickt</td></tr>
<tr><td>3</td><td>interactive</td><td>Übertragung läuft, Teile davon sind bereit im Buffer und mit responseText oder responseXML abrufbar</td></tr>
<tr><td>4</td><td>complete</td><td>Request aufgeführt und beendet</td></tr>
</table></td></tr>
<tr><td>responseText</td><td>Die Antwort des Servers als String.</td></tr>
<tr><td>responseXML</td><td>Die Antwort als XML-Ojekt, falls eine XML-Datei angefordert wurde.</td></tr>
<tr><td>status</td><td>HTTP-Status, sobald readyState den Wert 4 hat</td></tr>
<tr><td>statusText</td><td>HTTP-Statustext, sobald readyState den Wert 4 hat</td></tr>
</table>

Tabelle 11: Eigenschaften des XMLHttpRequest-Objectes

Anhang B: Relevante Browser Konfigurationen

B1: ActiveX im IE aktivieren

Die Aktivierung bzw. Deaktivierung von ActiveX im IE erfolgt in vier Schritten (siehe *Abbildung 72*), die nachfolgend beschrieben werden:

- Schritt 1: im IE auf den Menupunkt „Extras“ klicken
- Schritt 2: im sich öffnenden Fenster den Reiter „Sicherheit“ auswählen
- Schritt 3: in diesem Reiter auf den Button „Stufe anpassen …“ klicken
- Schritt 4: im sich öffnenden Fenster werden die ActiveX Optionen angepasst

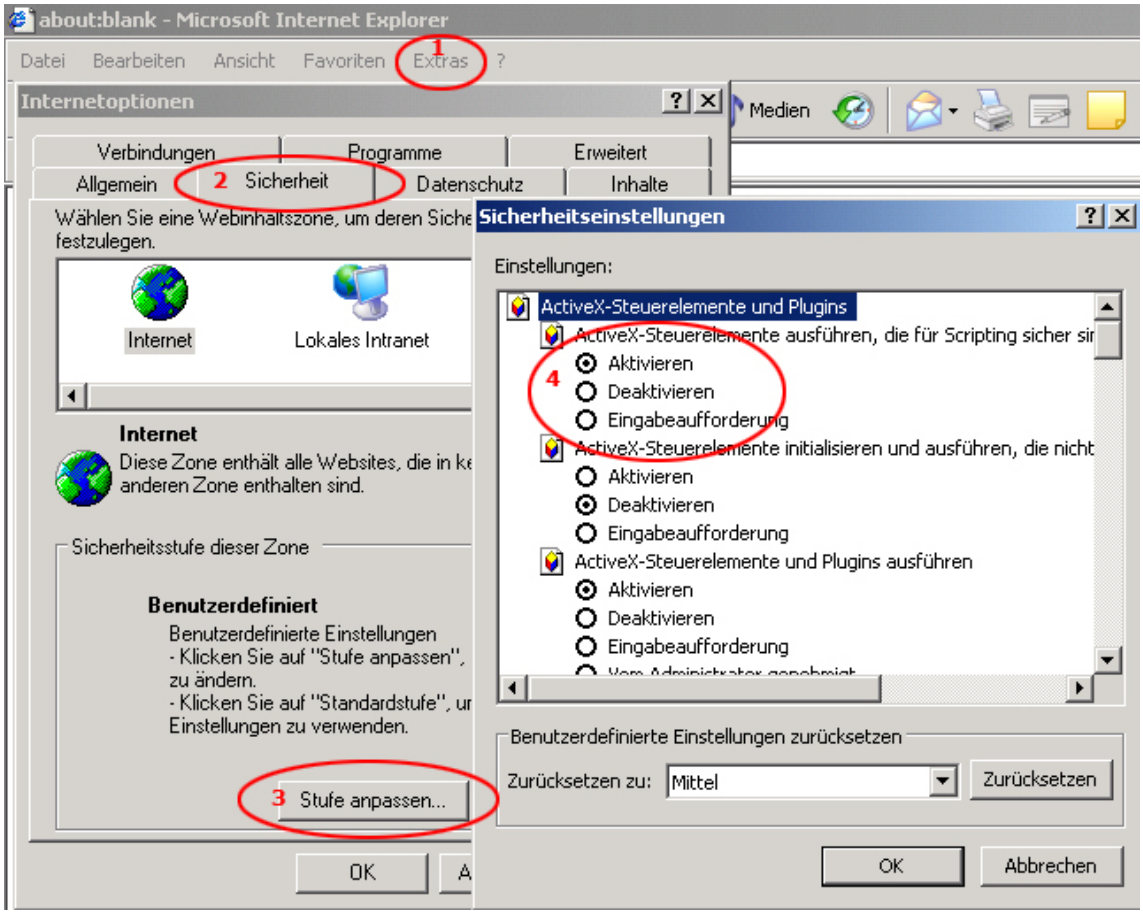

Abbildung 72: ActiveX im IE aktivieren bzw. deaktivieren

B2: IE Sicherheitsmodell

Das Sicherheitsmodell im IE kann wie in *Abbildung 73* beschrieben eingesehen bzw. angepaßt werden:

- Schritt 1: im IE auf den Menupunkt „Extras“ klicken
- Schritt 2: im sich öffnenden Fenster den Reiter „Sicherheit“ auswählen
- Schritt 3: in diesem Reiter werden die Einstellungen, durch Auswahl der entsprechenden Sicherheitszone, eingesehen bzw. angepasst

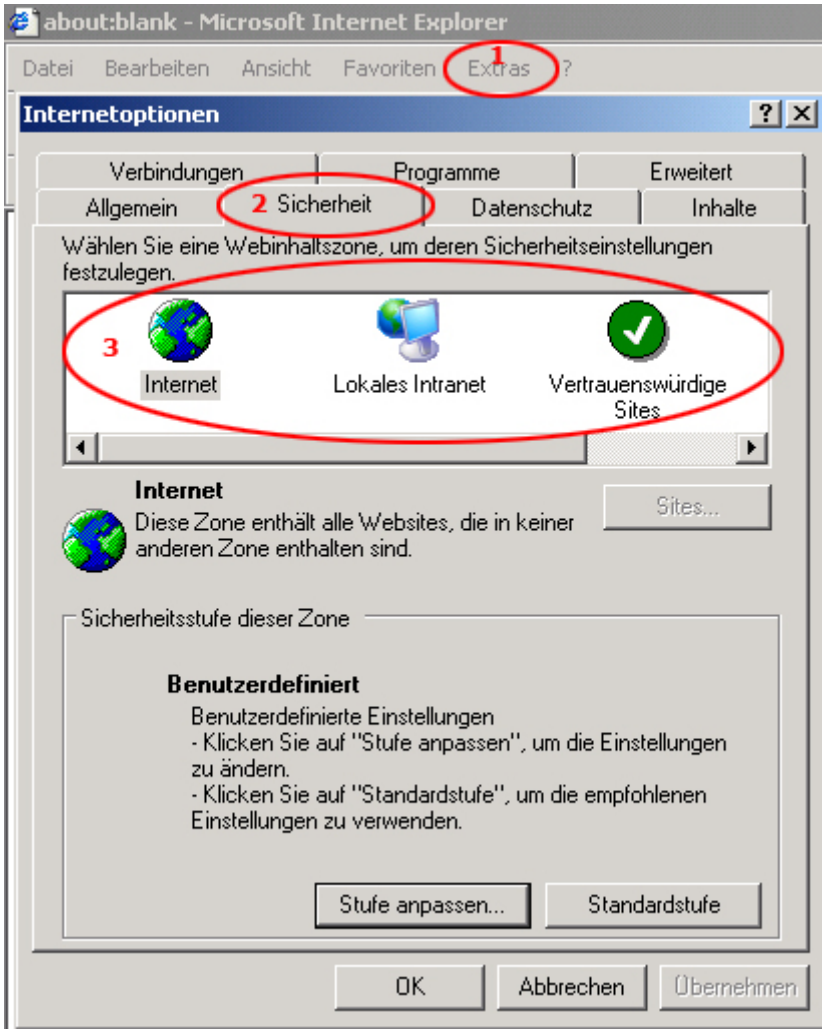

Abbildung 73: IE Sicherheitsmodell

Anhang C: Model-View-Controller

C1: Prinzip von MVC

Der Begriff Model-View-Controller (MVC) bezeichnet eine Architekturmuster, dass die Aufteilung von Softwarekomponenten in drei Einheiten beschreibt:

- *Model (Datenhaltung)l:* Das Modell enthält die darzustellenden Daten der Komponente bzw. Anwendung. Ihm ist nicht bekannt, woher diese Daten kommen, wie sie zusammenhängen bzw. dargestellt und verändert werden. Außerdem sollte das Modell beobachtbar sein, damit andere Bestandteile auf Änderungen reagieren können.
- *View (Präsentation):* Der View ist für die Präsentation der Benutzerschnittstelle zuständig. Er beschafft sich die Daten des zugeordneten Modells, präsentiert sie und reagiert auf Änderungen im Modell mit einer Benutzerschnittstellen-aktualisierung.
- *Controller (Anwendungslogik):* Der Controller nimmt die Benutzeraktionen entgegen, wertet sie aus und reagiert dementsprechend darauf.

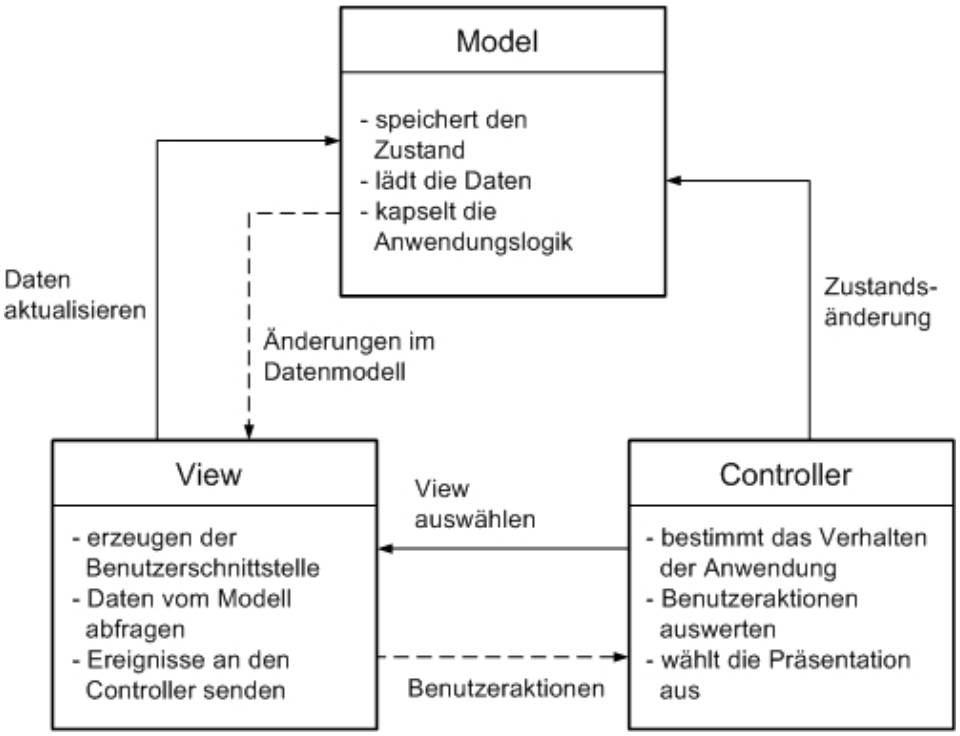

Abbildung 74: Prinzip von Model-View-Controller

Das Ziel des MVC Modells ist eine flexible Anwendungsarchitektur, die leicht erweitert und wieder verwendet werden kann. Des weiteren sind umfangreiche Anwendungen durch den Einsatz von MVC wesentlich übersichtlicher.

C2: MVC in normalen Webanwendungen

In normalen Webanwendungen wird MVC zur groben Strukturierung der Anwendung eingesetzt.

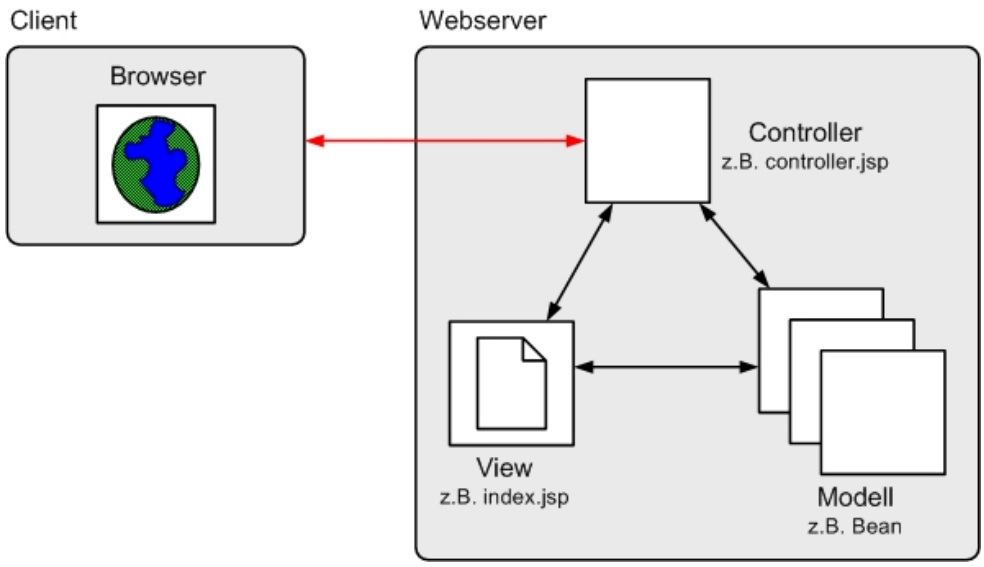

Abbildung 75: MVC in normalen Webanwendungen

Beschreibung der Einheiten:

- *Controller:* Der Controller nimmt die Anfragen des Browsers entgegen, überprüft gegebenenfalls die eingehenden Daten und gibt die Kontrolle an den entsprechenden View weiter. Es beantwortet selbst keine Anfragen.
- *View:* Der View erzeugt daraufhin die Webseite und fügt die entsprechenden Daten ein. Diese Webseite wird an den Browser geschickt, der sie anzeigen kann.
- *Modell:* Das Modell verwaltet die Daten und stellt Möglichkeiten zur Abfrage bzw. Bearbeitung bereit.

Abschließend ist zu sagen, dass das MVC Entwurfsmuster in normalen Webanwendungen nur auf Anwendungsebene eingesetzt wird.

Ressourcen

AJAX

[JoGam06]: Johannes Gamperl

AJAX-Web 2.0 in der Praxis

ISBN: 3-89842-764-1

Sand: 2006

[NzJMcJFa06]: Nicholas C.Zakas, Jeremy McPeak, Joe Fawcett

Professional AJAX

ISBN: 0-471-77778-1

Stand 2006

[DaCEPaDJa05]: Dave Crane, Eric Pascarello, Darren James

Manning: AJAX in Action

ISBN: 1932394613

Stand: 2005

[AJAXPAT]: AjaxPatterns

http://ajaxpatterns.org/

[JamGarr05]: Jesse James Garrett

Erklärung des Begriffes AJAX und der verwendeten Technologien.

http://www.adaptivepath.com/publications/essays/

archives/000385.php

Stand: 2005

[MASHUP]: MashUp

http://de.wikipedia.org/wiki/Mashup

AJAX-Frameworks

[ECHO2]: Echo 2

Unterstützt die AJAX Entwicklung mit Java.

http://www.nextapp.com/platform/echo2/echo/

[ECHOEXTRAS]: Echo Extras

Stellt weitere Benutzerelemente für Echo2 zur Verfügung.

http://www.nextapp.com/platform/echo2/extras/

[ECHOPNG]: EchoPointNG

Ein auf Echo2 basierendes Framework.

http://echopoint.sourceforge.net/

[ECHOSTRUTS]: Echo Struts

Ein auf Echo2 basierendes Framework zur Entwicklung von Anwendungen mit Struts.

http://wiki.nextapp.com/echowiki/EchoStruts/

[GWT]: Google Webtools

http://code.google.com/webtoolkit/

[DOJO]: Dojo

http://dojotoolkit.org/

[W4T]: INNOOPRACT - W4T

http://www.w4toolkit.de/index.php?bsID=302

[YUI]: Yahoo! UI Library

http://developer.yahoo.com/yui/

[BACKBASE]: Backbase

http://www.backbase.com/#home/home.xml[0]

[ICEFACES]: ICEfaces

http://www.icefaces.org/main/home/index.jsp

[JMAKI]: jMaki

https://ajax.dev.java.net/

[WEBORB]: WebORB

http://www.themidnightcoders.com/weborb/aboutWeborb.htm

[ZK]: ZK

http://www.zkoss.org/

[WIDSERV]: WidgetServer

http://www.c1-setcon.de/widgetserver/

[DWR]: Direct Web Remoting

http://getahead.ltd.uk/dwr/

[MOCHIKIT]: MochiKit
http://mochikit.com/

[RAP]: Rich Application Platform
http://www.eclipse.org/rap/

[RIALTO]: Rich Internet AJAX Toolkit
http://rialto.application-servers.com/wiki/

[PROTOTYPE]: Prototype
http://prototype.conio.net/

[RICO]: Rico
http://openrico.org/

[ATLAS]: Atlas
Komponenten für die .NET Entwicklung.
http://ajax.asp.net/Default.aspx

Browser

[IE]: Internet Explorer
http://www.microsoft.com/germany/windows/ie/default.mspx

[FIREFOX]: Mozilla Firefox
http://www.mozilla-europe.org/de/products/firefox/

[OPERA]: Opera
http://www.opera.com/products/desktop/?htlanguage=de/

Clientparadigmen/Schichtenmodelle

[LaWun05]: Lars Wunderlich
Software-Architekturen in Java
ISBN: 3-8266-1537-9
Stand: 2005

[ToLanDaRei05]: J2EE und JBoss
Softwareentwicklung mit J2EE und JBoss
ISBN: 3-4464-0508-9
Stand: 2005

[THINCLIENT]: Thin Client
http://de.wikipedia.org/wiki/Thin_Client

[RICHClient]: Rich Client
http://de.wikipedia.org/wiki/Rich-Client#Rich-Client

Entwurfsmuster

[MVC]: Model-View-Controller
http://de.wikipedia.org/wiki/Model_View_Controller

Entwicklungsumgebungen

[ATF]: AJAX Toolkit Framework
Auf Eclipse basierende IDE, zur Entwicklung von AJAX-Anwendungen mit JavaScript.
http://www.eclipse.org/atf/

[APTANA]: APTANA-DIE
Eine weitere Entwicklungsumgebung für AJAX-Anwendungen.
http://www.aptana.com/

[ECHOSTUDIO]: ECHO2 Studio
Eclipse Plugin, für die Entwicklung von Echo2 Anwendungen.
http://nextapp.com/platform/echo2/echostudio/

[GWTDESIGNER]: GWT-Designer
Entwicklungsumgebung für die Google Webtools.
http://www.instantiations.com/gwtdesigner/

[MYECLIPSE] MyEclipse
Unterstützt die Programmierung verschiedenen Anwendungstypen und beinhaltet eine AJAX Unterstützung.
http://www.myeclipseide.com/

[NEXAWEB]: Nexaweb Enterprise Web 2.0 Suite
http://www.nexaweb.com/

Externe Dokumente

[FRAGEBOGEN]:	Fragebogen der Projektumfrage [intern] Fragenkatalog der durchgeführten Projektumfrage. Stand: 2006
[UMFRAGE];	Projektumfrage [intern] Auswertung der Projektumfrage. Stand: 2006
[FRAMEWORKS]:	Vortrag: Betrachtung von AJAX-Frameworks [intern] Ergebnisse der betrachteten AJAX-Frameworks. Stand: 2006

Kommunikationsmodelle

[ASYN]:	Asynchrone Kommunikation http://de.wikipedia.org/wiki/Asynchrone_Kommunikation
[SYN]:	Synchrone Kommunikation http://de.wikipedia.org/wiki/Synchrone_Kommunikation

Lizenzen

[CreCoLiz]:	Creative Commons Lizenz http://de.creativecommons.org/
[BSDLiz]:	BSD-Lizenz http://de.wikipedia.org/wiki/BSD-Lizenz
[MozPuLiz]:	Mozilla Public Lizenz http://www.mozilla.org/MPL/MPL-1.1.html

PAI

[NUTSHELL]:	PAI in a Nutshell v. 3.0 [intern] Gesamtüberblick der PAI Plattform. Stand: 2006

[DIRSEC]: PAI Directory and Security Handbook v. 3.0 [intern]
Beschreibung der PAI Sicherheitsinfrastruktur.
Stand: 2006

[WEBLOGIN]: Component User Guide Weblogin v. 4.0 [intern]
Beschreibung der Weblogin Komponente.
Stand: 2006

[CLIENTCONT]: Vortrag: PAI Client Container [intern]
Stand: 2003

[SECDIRADCo]: Vortrag: Security Directory Administration Concepts [intern]
Stand: 2005

[LOGINGUIDE]: Login User Guide v. 1.4 [intern]
Beschreibung des Login Mechanismus.
Stand: 2003

RIA Technologien

[FLASH]: Flash
http://www.adobe.com/support/documentation/de/flash/

[XUL]: XML User Interface Language
http://www.mozilla.org/projects/xul/

[SVG] Scalable Vector Graphics
http://de.wikipedia.org/wiki/Scalable_Vector_Graphics

[OPENLASZ] OpenLaszlo
http://www.openlaszlo.org/

[FLEX2] Adobe Flex 2
http://www.adobe.com/de/products/flex/

Rich Client Frameworks

[RCP]: Rich Client Platform
http://www.eclipse.org/rcp/

[NetBeans]: NetBeans
http://www.netbeans.org/

[SPRING] Spring Rich Client

http://spring-rich-c.sourceforge.net/

Sicherheit

[OWASP]: OWASP

Community die sich mit der Sicherheit von Anwendungen beschäftigt.

http://www.owasp.org/index.php/Main_Page

[OWASPVAL] OWASP Validation Framework

Framework, um Benutzereingaben bzw. Daten leicht überprüfen zu können.

http://www.owasp.org/index.php/Category:

OWASP_Validation_Project

Sicherheitsprodukte

[IBMTivo]: IBM-Tivoli Access Manager

http://www-306.ibm.com/software/info/ecatalog/de_DE/

products/U106004N38762S58.html?

[SUNAcc]: SUN-Access Manager

http://www.sun.com/software/products/access_mgr/

Standards

[J2EE]: Java Platform, Enterprise Edition

http://java.sun.com/javaee/

[WebServ]: Webservices

Zusammenfassung der Webservice Spezifikation.

http://www.oio.de/public/xml/web-service-specifications.htm

[W3C]: W3C

http://www.w3.org/

[W3CXMLHTTP]: W3C XMLHttpRequest-Object Spezifikation

http://dev.w3.org/cvsweb/2006/webapi/XMLHttpRequest/

Overview.html?rev=1.28

[W3CCSS]: W3C CSS Spezifikation
http://www.w3.org/Style/CSS/

[W3CDOM]: W3C DOM Spezifikation
http://www.w3.org/DOM/

[W3CXML]: W3C XML Spezifikation
http://www.edition-w3c.de/TR/2000/REC-xml-20001006/

[W3CWDSL]: W3C WDSL Spezifikation
http://www.w3.org/2002/ws/desc/

[W3CSOAP]: W3C SOAP Spezifikation
http://www.w3.org/2000/xp/Group/

[W3CWAI]: Zugänglichkeitsrichtlinien für Web-Inhalte 1.0
http://www.w3c.de/Trans/WAI/webinhalt.html

[OpenAJAX]: OpenAjax Alliance
http://www.openajax.org/

[DeGeQua]: Deutsche Gesellschaft für Qualität
http://de.wikipedia.org/wiki/DGQ

[ISO]: International Organization of Standardisation
http://www.iso.org/iso/en/ISOOnline.frontpage

[ISO9126]: ISO-9126
http://de.wikipedia.org/wiki/ISO_9126

[RUP]: Rational Unified Process
http://en.wikipedia.org/wiki/Rational_Unified_Process

Testing

[J3UNIT]: J3 Unit
Unit Testing Framework für JavaScript.
http://j3unit.sourceforge.net/

[SELENIUM]: Selenium
Framework zur Beschreibung und Programmierung von Testszenarien.
http://openqa.org/selenium/

Tools

[PAROS]: Paros

Proxy zur Manipulation und Verfolgung der HTTP Kommunikation.

http://www.parosproxy.org/index.shtml

[WEBSCARAB]: Webscarab

Proxy zur Manipulation und Verfolgung der HTTP Kommunikation.

http://www.owasp.org/index.php/OWASP_WebScarab_Project

[SELENIUMIDE]: Selenium IDE

Firefox Addon zum erzeugen von Testszenarien für Selenium

http://openqa.org/selenium-ide/

[MozLiHead]: Daniel Savard, Nikolas Coukouma

Live HTTP Headers

Firefox Addon zum verfolgen von HTTP Headern

http://livehttpheaders.mozdev.org/

[ModHeaders]: Gareth Hunt

Modify Headers

Erlaubt die Modifikation von HTTP-Headern.

https://addons.mozilla.org/firefox/967/

[CSSVIEWER]: Nicolas Huon

CSS Viewer

Firefox Addon zur Anzeige von CSS Properties.

https://addons.mozilla.org/firefox/2104/

[LoadTimeAna]: Load Time Analyzer

https://addons.mozilla.org/firefox/3371/

[FIREBUG]: Joe Hewitt

Firebug

JavaScript Debugger für Firefox

https://addons.mozilla.org/firefox/1843/

[VENKMAN]: James Ross, Robert Ginda
Venkman
Ausgezeichneter JavaScript Debugger für Firefox
https://addons.mozilla.org/firefox/216/

Web 2.0

[WEB20]: Web 2.0
http://de.wikipedia.org/wiki/Web_2.0

[OREIWEB20]: Tim O'Reilly
Erklärung des Begriffes Web 2.0
http://www.oreillynet.com/pub/a/oreilly/tim/news/
2005/09/30/what-is-web-20.html
Stand: 2005

[DEFWEB20Co]: Web 2.0 Definition von Tim O'Reilly
http://radar.oreilly.com/archives/2006/12/
web_20_compact.html#ping

Webservices

[REST]: Thomas Bayer
Eine Einführung in die Entwicklung von REST Webservices.
http://www.oio.de/public/xml/rest-webservices.htm

[SOAP]: Grundlagen des SOAP Protokolls.
http://de.wikipedia.org/wiki/SOAP

[WDSL]: WDSL Grundlagen.
http://de.wikipedia.org/wiki/WSDL

[JSON-RCP]: JSON-RCP
http://en.wikipedia.org/wiki/JSON-RPC

[SOAP-IBM]: James Snell
Tutorial zur Erstellung von Webservices mit AJAX
http://www-128.ibm.com/developerworks/webservices/

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum

Unterschrift