Berufsakademie Mosbach
- Staatliche Studienakademie -
Lohrtalweg 10

74821 Mosbach

Erstellungszeitraum:
16.12.2006 – 16.02.2007

Studienarbeit im Studiengang Digitale Medien

Web (2.0) APIs aus Anwendersicht

Vorgelegt von: Richard Sternagel
Geburtsort: Achern
Geburtsdatum: 26.12.1983

Fachrichtung: Digitale Medien
Studiengang: DM04

betreuender Dozent: Prof. Dr. Arnulf Mester

Firma / Anschrift: 1&1 Internet AG
Brauerstraße 48
76135 Karlsruhe

Abgabedatum: 16.02.07

Inhaltsverzeichnis

1 Einleitung 1
2 Web 2.0 als Katalysator 2
2.1 Eigenschaften des Web 2.0 3
2.2 Kritik am Begriff „Web 2.0“ 4
3 Begriffsdefinitionen & Eigenschaften von Web APIs 6
3.1 (Web) APIs 6
3.2 Web Services 7
3.3 Warum Web APIs – aus Anbietersicht? 8
3.4 Warum Web APIs – aus Anwendersicht? 11
3.5 Risiken / Probleme bei der Nutzung von Web APIs 12
3.6 Kommunikationsarten: SOAP/WSDL, XML-RPC, REST 14
4 Kategorisierung & Anwendung von Web APIs 16
4.1 Klassifizierung von Web APIs 16
4.2 Web API-Anwendungen 18
4.3 Praktische Verwendung von drei unterschiedlichen Web APIs 19
4.3.1 Amazon E-Commerce Service (ECS) 19
4.3.2 Google Maps 24
4.3.3 Flickr API 28
5 Mashups – Web APIs miteinander kombiniert 31
6 Fazit und Ausblick 34

A Quellen- und Literaturverzeichnis 35
B Ehrenwörtliche Erklärung 40

1 Einleitung

Das World Wide Web (WWW) hat sich von Beginn an stetig verändert und tut dies fortwährend. Im Gegensatz zu den klassischen Medien besitzt das Web eine herausstechende Eigenschaft: Die Möglichkeit der Interaktion. Benutzer können unmittelbar zwischen der Rolle des Konsumenten und des Produzenten wechseln. Im Zuge dieses Mentalitätswandels, den u.a. das sog. „Web 2.0“ – auch treffend als „Mitmach-Web“ (vgl. [Sto06]) bezeichnet – ausmacht, sind viele Webgrößen dazu übergegangen Web APIs anzubieten. Dadurch wurden Stammdaten mittels Web Services, der „latest evolution of distributed computing“ ([Ive04], S. 1), gebündelt in Web APIs der Öffentlichkeit zugänglich. Schnell entstanden erste sog. „Mashups“, die auf Basis der veröffentlichten Schnittstellen „in ways that the original designers of those services never considered“ ([GE06], Vorwort) Informationen miteinander verknüpften.

Unter einem „Anwender“ wird in der vorliegenden Arbeit jemand mit fortgeschrittenen Programmierkenntnissen in einer beliebigen netzwerkfähigen Programmiersprache verstanden.

Ziel der Arbeit soll sein mithilfe der zusammengetragenen Informationen und anhand der Beispiele einen schnellen Einstieg in die Nutzung von Web APIs zu ermöglichen und den Hintergrund für eine eigene kritische Bewertung von Web APIs zu schaffen.

Dazu ist es zunächst erforderlich, den Begriff des Web 2.0 zu erläutern, der eng mit der Veröffentlichung von Web APIs bzw. Web Services verbunden ist. Nach den notwendigen technischen Begriffsdefinitionen folgt eine Aufzählung von Gründen aus Anbieter- und Anwendersicht, die für eine API-Nutzung sprechen. Anschließend wird eine Klassifizierung von Web APIs vorgestellt und exemplarisch drei APIs detaillierter betrachtet und diese in selbst konstruierten Szenarios verwendet. Zum Abschluss wird der Begriff Mashup als Kombination mehrerer Web APIs erläutert und prominente Mashups vorgestellt.

Im Verlauf der Arbeit werden notwendige Begriffe definiert, allerdings stellen technische Grundlagen nicht den Fokus der Studienarbeit dar. Dies finden die Leser bei Bedarf z. B. in „Service-orientierte Architekturen mit Web Services“, das im Spektrum Akademischer Verlag erschienen ist ([DJMZ05]).

2 Web 2.0 als Katalysator

Die Bereitschaft von Firmen, ihre Daten via Web APIs zugänglich zu machen, ist ein Bestandteil einer Entwicklung, die als Web 2.0 bezeichnet wird. Somit ist ein Verständnis des Begriffes Web 2.0 notwendig, um das entstandene Angebot an Web APIs nachvollziehen zu können. Dies soll im folgenden durch Erläuterung der *Herkunft* des Begriff, einer *Definition*, der Erarbeitung der charakteristischen *Eigenschaften* bzgl. Web APIs und einer abschließenden *Kritik* erfolgen.

Es ist sicherlich kein Zufall, dass „Web 2.0“ laut Tim O'Reilly der meistzitierteste Wikipediaeintrag von 2006 ist ([ORe07]). Entstanden in einem Brainstorming zwischen O'Reilly (Dale Dougherty und Tim O'Reilly) und MediaLive International war er 2004 das Ergebnis einer Namenssuche für eine Konferenz, die die neuen überraschenden Entwicklungen im Web zum Gegenstand haben sollte ([ORe05a], S. 1). Obwohl der Begriff sehr einprägsam ist und durch seine Plakativität eine leichte Verständlichkeit suggeriert (die nächste Version des Webs), offenbarte er schnell seine schwammige Bedeutung: Alles was neu war, schien dem Etikett Web 2.0 gerecht zu werden und wurde demnach auch inflationär so bezeichnet. Der Begriff Web 2.0 wurde zunehmend zum Marketing-Buzzword. Diese Entwicklung beobachtend versuchte Tim O'Reilly ein Jahr später (2005) die Bedeutung des Begriffes einzuschränken und in einer Art Mindmap die Charakteristiken aufzuzählen, die für ihn das Web 2.0 ausmachten ([ORe05a], S.1). Schließlich lieferte er kurz darauf eine kompakte Definition:

> „Web 2.0 is the network as platform, spanning all connected devices; Web 2.0 applications are those that make the most of the intrinsic advantages of that platform: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources, including individual users, while providing their own data and services in a form that allows remixing by others, creating network effects through an "architecture of participation," and going beyond the page metaphor of Web 1.0 to deliver rich user experiences.“ ([ORe05b])

Im Dezember 2006 versuchte sich O'Reilly erneut an einer (Re-)Definition (vgl. [ORe06]). Aber auch diese bleibt eher umschreibend als präzise. Daran anschließend zitiert er Googles CEO Eric Schmidt, der es in eine simple Regel presst: „Don't fight the internet“. O'Reilly führt weiter aus: „That's actually a wonderful way to think about it.

Think deeply about the way the internet works, and build systems and applications that use it more richly, freed from the constraints of PC-era thinking, and you're well on your way.“ (vgl. ebenda).

2.1 Eigenschaften des Web 2.0

O'Reilly stellte 2005 in seinem umfangreichen Artikel „What is Web 2.0“ ([ORe05a], S.1-5) etablierte Web 1.0-Webseiten ihren Web 2.0-Pendants gegenüber. Die dabei extrahierten Eigenschaften versteht er als eine Reihe von Prinzipien und Praktiken, die das Web 2.0, das keine harten Grenzen besitzen soll, ausmachen. Die wichtigsten mit dem Fokus auf Web APIs sind dabei:

„Web as Platform & End of the Software Release Cycle“
Das Internet ist zur Plattform (sog. Internet Operating System) geworden, mit eigenen (Web) Applikationen, die nicht verkauft oder lizensiert werden, sondern als Service nutzbar sind. Sie verfolgen keine klassischen Releasezyklen, sondern unterliegen einem kontinuierlichen Verbesserungsprozess (vgl. agile Methoden als Vorgehensmodell in der Programmierung – [Fow05]). Dies bezeichnet O'Reilly als „Software as a Service“. Beispiele hierfür sind Google, Amazon und eBay (vgl. [ORe05a], S.1-5).

„Harnessing Collective Intelligence & Data as the Intel Inside“
Die kollektive Intelligenz, die auch als „Wisdom of Crowds“ (vgl. [Sur04]) bezeichnet wird, ist der treibende Motor hinter Projekten wie Wikipedia, Flickr oder del.icio.us – sog. „Social Software“ oder beschreibender, kollaborativer Software. Dabei werden die Inhalte von den Nutzern erstellt und anstatt starrer Taxonomien mit „Tags“ (Schlagworte) versehen und dadurch gleichzeitig kategorisiert (sog. Folksonomy). Je mehr Nutzer partizipieren, desto besser wird der „user generated content“. Diese Idee findet sich auch bei den „Global Playern“ wieder: Der PageRank von Google beruht genau auf diesem Prinzip der kollektiven Intelligenz und nutzt die Anzahl von Verlinkungen als Qualitätsmaß, Amazons Alleinstellungsmerkmal sind seit jeher Benutzerreviews und eBay ist per Definition eine „Auktionsplattform“ für deren Nutzer. Jede Interaktion eines Benutzers mit dem System kann dabei aufgezeichnet und im Sinne eines Feedbacks genutzt werden, um die Benutzerschnittstellen weiter zu verbessern. User werden als „Co-Developer“ betrachtet. Die daraus resultierenden Stammdaten bilden das wichtigste Gut, da sie von Tag zu Tag besser und einzigartiger werden und somit schwer nachzubilden sind (vgl. [ORe05a]). Daraus lassen sich Features wie die

Produktempfehlungen von Amazon und die Vorschläge von Google bei falsch geschriebenen Wörtern erzeugen.

Lightweight Programming Models

Als sozusagen ersten weit verbreiteten Web Service sieht O'Reilly RSS an. Aufgrund der Einfachheit von RSS wurde Content-Syndikation weitflächig angenommen und Inhalte auf vielfältige Arten wiederverwendet. Webseiten wie ChicagoCrime.org oder Busmonster.com veranschaulichen die Experimentierfreudigkeit findiger Programmierer, die teilweise schon, bevor es ein API zu Google Maps gab, Google Maps als Plattform nutzten (vgl. [Gra05]). Web APIs nutzen also das Potential der kollektiven Intelligenz für die darunter liegende Plattform konsequent aus. Für Spool stellen sie die treibende Kraft des Web 2.0 dar: „One tool that is making this all possible is the increasing availability of Application Programming Interfaces (APIs)." ([Spo06]). Porter und MacManus konstatieren:

> „The Web of documents has morphed into a Web of data. We are no longer just looking to the same old sources for information. Now we're looking to a new set of tools to aggregate and remix microcontent in new and useful ways." ([McP05])

2.2 Kritik am Begriff „Web 2.0"

Bei genauerer Betrachtung existierten viele der „neuen" Technologien und Ideen des Web 2.0 bereits vor der Begriffsfindung (z. B. Amazon / eBay und „user generated content"), oder setzten sich aus ihnen zusammen (z. B. AJAX). Durch die gestiegene Verfügbarkeit von Breitbandanschlüssen und Flatrates erhöhte sich auch die Zahl der Internetnutzer (vgl. [AZS06]). Dies ermöglichte erst Anwendungen wie Flickr oder YouTube, die mit größeren Datenmengen hantieren. Fraglich ist also letztlich, ob die natürliche Evolution, die das Web 2.0 darstellt, überhaupt eine Definition braucht?

Für Tim Berners-Lee ist der Aspekt, dass Menschen miteinander interagieren, bereits eine inhärente Eigenschaft des „Web 1.0": „If Web 2.0 for you is blogs and wikis, then that is people to people. But that was what the Web was supposed to be all along.". Für ihn ist Web 2.0 „a piece of jargon, nobody even knows what it means." (vgl. [LBL06]).

Auch für Paul Graham bedeutet der Begriff nichts: „It seemed that it was being used as

a label for whatever happened to be new-- that it didn't predict anything. [...] The "trends" we're seeing now are simply the inherent nature of the web emerging from under the broken models that got imposed on it during the Bubble.“([Gra05]).

Russell Shaw hat ebenso Probleme mit dem Begriff: „[...] it is a contrivance, meant to imply a unified movement or wave toward a better Web.“ Er erkennt an, dass neue Elemente im Web entstanden sind, aber diese zu unterschiedlich sind „... they cannot be classified under a common umbrella. [...] Some revolutionary, some evolutionary, some impractical. Some are collaborative, others are highly competitive with each other.“, um unter einem Begriff sinnvoll subsumiert werden zu können ([Sha05]).

Begriffe im Umfeld des Web 2.0 Abb. 1
(http://kosmar.de/archives/2005/11/11/the-huge-cloud-lens-bubble-map-web20/)

Es lässt sich also ein Konsens feststellen: Wenn Web 2.0 lediglich die Bezeichnung für alle neuen Entwicklungen (sowohl Technologien, Prinzipien als auch Praktiken, was einen sehr großen Geltungsbereich aufspannt - s. Abb. 1) im Web ist, dann ist der Begriff überflüssig. Er wird nicht benötigt, da keine auf einen Zeitpunkt datierbare Revolution stattfand, sondern nur eine andauernde evolutionäre Entwicklung des Webs existiert, in der sich das Web auf der Grundlage der konsequenten Ausnutzung seiner Charakteristiken zu einem eigenständigen Medium entwickelt (vgl. [Yan06]).
Wenn im Web also Applikationen die klassischen Websites ersetzen, ist es sinnvoll, zu diesen auch (Web) APIs anzubieten, um die eigenen Plattformen zu etablieren.

3 Begriffsdefinitionen & Eigenschaften von Web APIs

Wissend um das Phänomen Web 2.0 und, dass Web APIs eine Begleiterscheinung davon sind, sollen in diesem Kapitel nun die technischen Grundlagen dargelegt werden, die zum Verständnis und dem praktische Anwenden in Kapitel 4 benötigt werden. Dazu werden die Begriffe *Web API*, *Web Service* und *serviceorientierte Architektur* erläutert, Motive für das Angebot an Web APIs und deren Nutzung dargestellt und schließlich wird kurz auf die möglichen Kommunikationsarten zwischen Web Service Client und Web API eingegangen.

3.1 (Web) APIs

Als ein *„Application Programming Interface“*, kurz API, wird die abstrakte Schicht, auch Interface bzw. Schnittstelle bezeichnet, die anderen Programmen erlaubt, mit dem System, welches das API anbietet, in einer definierten Art und Weise zu kommunizieren und damit auf dessen Funktionalitäten zuzugreifen (vgl. [CS03], S.51).
Dadurch ergeben sich folgende Vorteile:

1. Der Anbieter des API kann darüber entscheiden, welche Anfragemöglichkeiten angeboten werden sollen und wie auf diese zugegriffen werden kann.
2. Solange eine bestehende Schnittstelle nicht verändert wird, kann das API ohne Auswirkungen für bisherige Anwender weiterentwickelt werden.
3. Für den Anwender, der das API nutzt, ist die Komplexität der nötigen Geschäftslogik, die hinter dem API liegt, transparent: Er muss nicht wissen, wie das API *funktioniert*, sondern nur, wie man es *bedient.*

Der Begriff *Web API* besitzt keine standardisierte Definition. Er kann allerdings, da er aus den Begriffen Web und API zusammengesetzt ist, hergeleitet werden: Ein API, das über das World Wide Web (WWW) angesprochen werden kann. Web APIs können als Schnittstelle zu (mehreren) Web Services begriffen werden. Eine Ausnahme bilden dabei Web APIs, die via JavaScript zur Verfügung gestellt werden (z. B. Google Maps): Sie arbeiten ohne Web Services und ähneln in ihrer Nutzung eher einem „klassischen“ API.

3.2 Web Services

Eine Definition von *Web Services* (in der Literatur sowohl als „Web services“ als auch „Web Services“) fällt schwer. Selbst die W3C Web Services Architecture Group des World Wide Web Consortium (W3C) hat seine Definition eines Web Services im Laufe der Zeit geändert (zuletzt August 2003), (vgl. [DJMZ05], S. 26):

> „A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.“ ([WSAG04])

In dieser Definition des W3C werden bereits konkrete Technologien genannt:

- Die Web Service Description Language (WSDL) mit der die angebotenen Dienste und die Operationen beschrieben werden können.
- SOAP-Nachrichten, die die Kommunikation der zu übermittelnden Inhalte realisieren, typischerweise über das Hypertext Transport Protocol (HTTP).

Sowohl SOAP als auch WSDL sind XML-Formate, die die angestrebte Interoperabilität begründen und dadurch den Einsatz in heterogenen Systemen ermöglichen.

Die Kommunikation zwischen einem System und einem Web Service stellt somit eine verteiltes System dar. Es ist üblich, auch von Web Services zu sprechen, wenn eine Schnittstellenbeschreibung via WSDL fehlt oder eine Alternative zu SOAP verwendet wird: „...it's possible to create a Web service that follows no standard other than XML and HTTP“ ([NL05], S. 168). Dies entspricht der ursprünglichen Definition des W3C von 2002, in der nur von Schnittstellenbeschreibungen in Form von XML-Artefakten und der Verwendung von Internetprotokollen gesprochen wird (vgl. [DJMZ05], S.26).

Web Services sind darüber hinaus die gebräuchlichste technologische Grundlage – neben u.a. CORBA, DCOM oder EJB – für Serviceorientierte Architekturen (SOA). Eine SOA ist ein fachliches Architekturmuster, das eine Anwendungslandschaft aus voneinander unabhängigen lose gekoppelten Komponenten (Services) beschreibt. Ein Service kann dabei sehr einfach wiederverwendet werden, da er eine feste, definierte

Leistung kapselt (vgl. [RH05]). Anzumerken ist, dass SOA-Applikationen und Mashups prinzipiell Ähnlichkeiten aufweisen (vgl. [McK06]).

Die Gesellschaft für Informatik e. V. konstatiert dabei, dass eine SOA fälschlicherweise häufig „mit der Verwendung von Web Services und deren assoziierten Technologien“ gleichgesetzt wird ([RH05]).

3.3 Warum Web APIs – aus Anbietersicht?

Im Folgenden werden zunächst das „Screen Scraping“ als Vorläufer von Web APIs erläutert und dann die Vorteile von Web APIs für Anbieter aufgezählt.

Das World Wide Web sieht für den Datenaustausch menschenlesbare Auszeichnungssprachen („Markup Languages“) vor, die den Inhalt lediglich strukturell kennzeichnen. Deshalb kann man jederzeit den Quelltext einer empfangenen Website ansehen und studieren. Der eigentliche Inhalt einer Seite kann programmatisch geparst, extrahiert und verarbeitet werden. Dieses Verfahren wird als „Screen Scraping“ bezeichnet, besitzt aber mehrere Nachteile:

1. Eine automatisierte Verarbeitung ist nur über Umwege möglich und man begibt sich in eine starke Abhängigkeit zur vorliegenden (X)HTML-Struktur. Sobald der Seitenbetreiber seine Inhalte anders auszeichnet, werden Anpassungen des Parsingvorgangs nötig (vgl. [Ive04], S. 4).
2. Ein weiterer Nachteil des Screen Scraping-Verfahrens ist der erzeugte Overhead und die damit verbundene Ineffizienz. Soll beispielsweise stündlich die Preisentwicklung eines bestimmten Produktes verfolgt werden, beschränkt sich der eigentliche Anteil an erforderlichen Nutzdaten auf einige wenige Zahlen und ein Währungssymbol, obwohl immer die gesamte Datei angefragt und übertragen werden muss (vgl. [Ive04], S. 17).
3. Schließlich ist es illegal, sich Daten anderer Webseitenbetreiber anzueignen, wenn diese unter Nichtangabe der Originalquelle genutzt oder gewinnbringend verwertet werden. Darüber hinaus kann Screen Scraping auch ausdrücklich durch den Webseitenbetreiber verboten werden (vgl. [Ive04], S. 17).

Die Konsequenz daraus war, dass Webseitenbetreiber Screen Scraping-Zugriffe einerseits unterbinden wollten, aber andererseits auch das Interesse an definierten Zugriffsmöglichkeiten (also APIs) auf ihre Angebote und Produkte erkannten. Als diese

Anfragen mit Web Services realisierbar wurden, reagierten die Betreiber großer Webseiten letztendlich und veröffentlichen Web APIs, die einen Zugriff auf „rohe Daten“ / Informationen erlauben (vgl. Amazon [Bau03], S. 191).

Für den Webseitenbetreiber ergeben sich durch die Nutzung von Web APIs folgende Vorteile:

1. definierte kontrollierbare Zugriffe
2. zusätzliche Einkommensquellen
3. eine Anwenderbindung an die eigene Plattform
4. zusätzliche statistische Auswertungsmöglichkeiten
5. eine Steigerung der Popularität und der Nutzerzahlen (virales Marketing)
6. Vermietung von nicht genutzten Ressourcen
7. eine Innovationsquelle, die neue Ideen und Features hervorbringt

ad 1) Für den API-Anbieter fallen zwar Kosten für Rechnerkapazitäten, Entwicklung und ggf. Supportpersonal an, aber dafür kann er Bedingungen aufstellen und durch eine verpflichtende Registrierung die Kontrolle über die Herausgabe und Verwendung seiner Daten behalten.

ad 2) Es hat sich gezeigt, dass sich mit den offenen Schnittstellen bereits *Gewinne erzielen* lassen. Amazon und eBay beispielsweise fokussieren sich deutlich auf eine gewinnbringende Zweitverwertung ihrer Daten – so erwirtschaftete Amazon im zweiten Quartal 2005 ca. 28% Prozent (490 Millionen Dollar) seines Umsatzes über Fremdsysteme. Amazon gibt aber nicht bekannt, wie hoch der Anteil der Web Services daran ist (vgl. [Cla05]). Bei eBay betrug die Zahl der durchgeführten API-basierten Transaktionen in 2005 bereits 8 Milliarden (vgl. [MTO06], S. 4). Sofern die eigenen Dienste keinen direkten Umsatz erzeugen, kann dieser mittels Werbung (s. Google) oder durch ein Nutzungslimit erreicht werden, ab welchem der Service kostenpflichtig wird (vgl. [ORe02], Punkt 2).

ad 3) Sobald ein Anwender etwas entwickelt, was einen zusätzlichen Nutzen für eine Plattform bringt und sich auf sie stützt, begibt er sich in eine gewisse Abhängigkeit zu ihr. Aufgrund seines erworbenen Wissens ist es wahrscheinlich, dass er bei der Plattform bleibt, da er dieses wiederverwenden will (vgl. [ORe02], Punkt 1).

ad 4) API-Anbieter können aufgrund der von API-Anwender erstellten alternativen Nutzungsmöglichkeiten ihres Angebots kostengünstiges „*Interface-Researching*“ ihrer eigenen Plattform betreiben und *statistisch erfassen*, welche Nischenprodukte sich zum Beispiel besser über Fremdsysteme verkaufen o. ä. (vgl. [Bau03], S.191).

ad 5) Es gibt Anbieter, die scheinbar *keine primären kommerziellen Zwecke* verfolgen wie Google, Yahoo, Flickr, YouTube uvm., sondern mit ihren APIs vorrangig ihre *Popularität steigern* und mit der gewonnenen Reichweite ihre *Nutzerzahlen erhöhen* wollen. Webseitenbetreiber, die beispielsweise ihre Flickr-Bilder bei sich integrieren, bewirken durch jeden Klick ihrer Besucher neuen Traffic für Flickr und dadurch potentielle Neukunden. Diese Art eines viralen Marketings ist nicht zu unterschätzen (vgl. [War06a]).

Bei näherem Hinsehen erkennt man aber, dass z. B. Google bereits Ideen hat, wie mit den eigenen APIs Umsatz erzielt werden kann – evtl. sobald eine kritische Masse erreicht ist. So haben viele APIs einen Punkt in der Lizenzvereinbarung, der zukünftige Werbeeinblendungen in den Daten nicht ausschließt. Und auch für kleinere Anbieter kann die Verfügbarkeit ihrer Daten via Web API ein Schritt in die Popularität sein, die in ein paar Jahren womöglich zur Übernahme durch Google, Yahoo und Co. führt.

ad 6) Temporär nicht genutzte Ressourcen (z. B. Speicher- und Rechenkapazitäten von Computerclustern) können über Web Services vermietet werden, um Instandhaltungskosten zu decken. Was sich abenteuerlich anhört, hat Amazon bereits mit der Einführung des Amazon Simple Storage Services (Amazon S3 - vgl. [AS3oD]) und der Amazon Elastic Computer Cloud (Amazon EC2 – vgl. [AECoD]) eindrucksvoll bewiesen.

Diese Art von Web Services sind auch für den Anwender von Vorteil (s. Kapitel 3.4) und werden ebenfalls bei der API-Klassifizierung aufgegriffen (s. Kapitel 4.1).

ad 7) Schließlich vergrößert sich der Kreis der für die Plattform arbeitenden Entwickler von selbst, sobald lediglich eine gut dokumentierte und einfach zu benutzende Schnittstelle zu attraktiven Daten existiert. So kann jeder innovativer Ansatz für die eigene Plattform übernommen werden oder ggf. die Web-Applikation des API-Anwenders aufgekauft werden.

3.4 Warum Web APIs – aus Anwendersicht?

Auch der API-Anwender hat durch den neuen Zugriff auf große Datenarchive Vorteile:

1. die Möglichkeit der nahtlosen Integration der abgerufenen Daten
2. Provisionen bei erfolgreichen Transaktionen
3. innovatives Experimentierfeld für Entwickler
4. ein Nutzungsargument für die Web Applikation: die eigenen Daten können jederzeit exportiert werden
5. ökonomische Dienst- oder Infrastrukturleistungen können gegen Bezahlung in Anspruch genommen werden

ad 1) und 2) Mittels Web APIs können *fremde Daten nahtlos* in eigene Webseiten, Applikationen o. ä. *eingebunden werden* – als Verkäufer oder sog. Affiliate wird man zusätzlich mit einer *Provision* belohnt. Beispielsweise lässt sich schnell eine Karte von Google Maps auf der Webseite des eigenen Unternehmens einbauen (s. Szenario 2 in Kapitel 4.3.2).

ad 3) Ein weiterer Beweggrund, sich mit APIs zu befassen, ist für viele Entwickler die Möglichkeit mit *echten Daten* experimentieren zu können, zu denen man als Einzelperson keinen Zugang hat (Satellitenbilder, Produktkataloge), um womöglich durch Kombination mehrerer Dienste (sog. Mashups - s. Kapitel 5) einen neuen Mehrwert zu kreieren.

ad 4) Zu unterscheiden ist außerdem, ob man fremde Daten über ein Web API abfragt oder Daten, die man selbst erstellt hat (z. B. eigene Fotos von Flickr oder eigene Bookmarks bei del.icio.us). Wenn es sich um selbst erstellte Daten handelt, stellt ein Web API ein zusätzliches *Nutzungs- und Vertrauensargument* dar, da man mit dem erforderlichen Know-how theoretisch immer und überall über seine Daten verfügen kann (vgl. [Kei06])

ad 5) In der Regel können Web APIs / Web Services kostenlos verwendet werden, wohingegen bereits Web Services veröffentlicht worden sind, deren Nutzung kostenpflichtig ist. Die häufigste Bezahlungsart ist dabei „Pay per Use“. Wie in Kapitel 3.3 ad 6) erwähnt, erlauben es diese Art von Web Services z. B. kostengünstige Speicher- oder Rechenkapazitäten zu mieten (vgl. Amazon S3 [AS3oD] und Amazon EC2 [AECoD]). Darüber hinaus können kurzfristig und unverbindlich Arbeitskräfte für

qualitativ anspruchslose aber quantitativ zeitaufwändige Aufgaben mobilisiert werden (vgl. Amazon Mechanical Turk [AMToD]).

3.5 Risiken / Probleme bei der Nutzung von Web APIs

Wenn sowohl Anbieter als auch Anwender Vorteile bei der Nutzung von Web Services haben, gibt es dann überhaupt Nachteile? Wo liegen die Risiken für Anwender?
Folgende Punkte sind zu beachten:

1. Voraussetzung eines API Keys
2. Restriktionen der Lizenzvereinbarungen sowie eine Unterscheidung von Privatpersonen und Unternehmen bzgl. der Nutzung der APIs
3. APIs erlauben nur den (definierten) Zugriff auf Daten
4. einseitige Abhängigkeit der Anwender
5. Migrationszwänge

ad 1) Um die verschiedenen Web Services nutzen zu können, ist immer ein *API-Key* o. ä. des Web Service Anbieters nötig. Bevor man diesen Schlüssel erhält, der den Anwender eindeutig identifiziert und dem API-Betreiber eine genaue Verfolgung „wer was wann tut“ ([War06a]) erlaubt, muss man den jeweiligen *Lizenzvereinbarungen* zustimmen, die gewisse Restriktionen beinhalten. Oft gibt es ein Nutzungslimit (z. B. bei Amazon: „do not exceed 1 call per second per IP address“ [WSL06]) und die Auflage, die Daten entsprechend mit der Quelle zu verlinken oder den Eigentümer der Daten deutlich kenntlich zu machen. Es ist also ratsam, die Lizenzvereinbarungen auf ähnliche Punkte zu überprüfen.

ad 2) Google Maps behält sich beispielsweise ausdrücklich vor ([GMToD], Punkt 1.5) in Zukunft Werbung anzuzeigen und bietet Unternehmen bereits einen Tarif an, der verbesserten Support gewährleistet und die Möglichkeit, die Werbung zu aktivieren oder deaktivieren ([GMEoD]). Hier entsteht ein übliches Bezahlsystem: Kostenlose Basisfunktionen und darauf aufbauende Premiumangebote. Zwar ist das Google Maps API auch von Unternehmen frei nutzbar, allerdings ist eine Einschränkung der Nutzung auf *Privatpersonen* oder für *nicht-kommerzielle Zwecke* häufiger Bestandteil der Lizenzvereinbarungen.

ad 3) Flash-Experte Aral Balkan, macht auf seinem Vortrag „Mash my Flex up“ auf der d.Construct 2006 deutlich, dass APIs *nicht offene* / *freie Daten* („APIs != open data“),

sondern nur den *Zugriff auf Daten* („ ... but access to data“) nach den Vorstellungen der Anbieter bedeuten (vgl. [Bal06] Folie 44-49). Frank Westphal bezeichnet das als Kernpunkt: „Jeder versucht in den Besitz von Stammdaten zu kommen, die er dann irgendwie kostbar verwerten kann“ ([War06a]). Offensichtlich dabei ist, dass manche Lizenzvereinbarungen auch „non-competition clauses“ ([Bal06], Folie 48) beinhalten, die eine Datenverwertung für Konkurrenten verbieten.

ad 4) Problematisch kann die *einseitige Abhängigkeit* werden: Wenn man als Entwickler ein innovatives Konzept entwirft und damit Aufmerksamkeit erreicht, hängt man immer noch an der existentiellen „Nabelschnur“ des API-Betreibers. Man sollte sich dieser Abhängigkeit immer bewusst sein, sobald man versucht, auf Web APIs ein Geschäftsmodell aufzubauen. Bei eBay kann man sich allerdings mit genau dieser Strategie schon den Lebensunterhalt verdienen (vgl. [Sch03]).

ad 5) Bei manchen API-Anbietern kann man zur *Migration* oder gar zum *Umstieg* auf einen evtl. existierenden Konkurrenzservice eines anderen Anbieters (z. B. Google Maps und Yahoo! Maps) gezwungen sein, da neue Web API- bzw. Web Service-Versionen veröffentlicht werden und das Unterstützen von obsoleten Services keinen Gewinn bringt. Ein Beispiel dafür ist Google Maps. In der Lizenzvereinbarung heißt es:

„1.3 Modifications. Google reserves the right to release subsequent versions of the API and to require You to obtain and use the most recent version.“ ([GMT05])

Vor kurzem wurde die Weiterentwicklung der 2002 ersten veröffentlichten API von Google („Google SOAP Search API“) eingestellt ([GSS06]), um der AJAX Search API den Vorzug zu geben. Es werden keine neuen API Keys mehr vergeben. Bisherige Keys funktionieren noch weiter, allerdings ist fraglich, ob das API nicht bald ganz eingestellt wird. Die im Web 2.0 inflationär verwendete Bezeichnung „Beta“, als Hinweis auf den Zustand eines Dienstes, wird hier konsequent realisiert – mit der Einstellung des Dienstes. Dies ist aber von Servicebetreiber zu Betreiber unterschiedlich: Bei Amazon sind auch ältere WSDL-Versionen verfügbar – die anfangs eingeführte „Subscription ID“ wurde zwar mittlerweile durch die „AWS Access Key ID“ ersetzt, ist aber weiterhin gültig und verwendbar (vgl. [ECS06]). Generell sind Web APIs, die von vorne herein darauf ausgelegt sind, Umsatz zu erzielen (Amazon, eBay) beständiger, als experimentelle und kurzfristig einstellbare Schnittstellen, bei denen eine profitable Vermarktung noch aussteht (z. B. bei vielen Google APIs).

3.6 Kommunikationsarten: SOAP/WSDL, XML-RPC, REST

Bisher wurden die Begriffe Web API und Web Services definiert und deren Entstehungsgeschichte, Vorteile und Risiken für Anwender vorgestellt. Um einen Web Service nutzen zu können, muss zusätzlich in die potentiellen Kommunikationsarten, die in den Szenarios (s. Kapitel 4.3) eingesetzt werden, kurz eingeführt werden. Dabei werden die drei Vertreter *SOAP/WSDL*, *XML-RPC* und *REST* vorgestellt und die sich momentan etablierenden *„nicht-XML-Formate“* skizziert.

Aus Anwendersicht ist ein Web Service, der auf der Basis von *SOAP/WSDL* bereitgestellt wird, aufwändiger zu nutzen als vergleichbar schlankere Ansätze wie XML-RPC oder REST. Dies schlägt sich auch in der Nutzungspräferenz nieder: Nur ca. 20% der Entwickler, welche die Amazon Web Services nutzen, entscheiden sich für SOAP (vgl. [And06]). Viele Standards aus dem „WS-Stack“ sind für sehr komplexe Anwendungszenarien entworfen worden und spielen ihre Vorteile erst aus, wenn Anforderungen wie Sicherheit (Verschlüsselung), transaktionales Verhalten, Verlässlichkeit und Geschäftsprozess-Modellierungen (aus dem Zusammenspiel mehrerer einzelner Dienste entsteht ein geschäftlicher Mehrwert) benötigt werden. Ein Einsatz von mehreren (z. T. noch entstehenden) Standards aus dem WS-Stack ist daher eher in komplexen geschäftskritischen Unternehmensbereichen angesiedelt.

Eine Alternative zu SOAP/WSDL ist die Verwendung von *XML-RPC*. Wie die Bezeichnung nahe legt, wird hier ein Remote Procedure Call (RPC) ausgeführt, dessen Artefakte (Request & Response) in XML definiert sind. Die schlanke Spezifikation, die von Dave Winer 1998 veröffentlicht wurde (vgl. [Win99]), ist dabei eher für den direkten Praxiseinsatz ausgelegt als für komplexe Geschäftsanwendungen. Die Einfachheit wird durch ein beschränktes Typsystem, das Nicht-Vorhandensein von Namensräumen und der Festlegung auf synchrone Methodenaufrufe (Request / Response) erreicht. XML-RPC bildet damit den inoffiziellen Vorläufer für das komplexere SOAP (vgl. [DJMZ05] S. 69 ff.).

REST konforme Web Services („RESTful“ Web Services) erfordern keinen XML-Request, sondern erwarten diesen in Form einer HTTP GET-Operation an eine URL. Roy Thomas Fielding stellte das Architekturstil REpresentational State Transfer

(REST) 2000 erstmals in seiner Dissertation (vgl. [Fie00], Kapitel 5 ff.) vor. Viele API-Betreiber unterstützen REST aufgrund der Einfachheit: Ein Request lässt sich direkt im Browser testen, da der empfangene XML-Response unmittelbar angezeigt wird und analysiert werden kann (vgl. [DJMZ05], S. 72).

Bei den Web APIs scheinen sich momentan eher die leichtgewichtigen Lösungen (z. B. XML-RPC und vor allem REST) durchzusetzen (vgl. [And06]). Neuerdings auch „nicht-XML-Formate“ wie beispielsweise „JSON“ ([Cro06]) und „serialisiertes PHP“ ([POS07]) bei Flickr (vgl. auch [And06]). Die Codegenerierung von SOAP/WSDL, die ein SOAP-Framework voraussetzt, erspart zwar das manuelle Parsen des (XML-)Responses, erschwert aber auch die Fehlerdiagnose und erhöht die Einstiegshürde. Außerdem müssen die Konzepte hinter SOAP und WSDL zuerst verstanden und der Umgang mit dem gewünschten SOAP-Framework erlernt werden.

4 Kategorisierung & Anwendung von Web APIs

Im Folgenden werde ich eine Einordnung existierender Web APIs vornehmen und einige interessante Anwendungen vorstellen, die auf Web APIs basieren.

Daran anschließend werde ich den *E-Commerce Service (ECS)* von Amazon, das *Google Maps API* und das *Flickr API* genauer vorstellen und jeweils anhand von einem selbst konstruierten Beispiel vertiefen.

Die Beispiele sind in Java oder in PHP programmiert und setzen zur Nutzung den StAX-Parser ([StA06]) von Java 1.6 und die SimpleXML-Extension ([PSX07]) von PHP 5 voraus. Außerdem wird ein gültiger API-Key des jeweiligen API / Web Service-Betreibers benötigt, der an den mit „[api_key]“ gekennzeichneten Stellen einzufügen ist.

Um sich auf das Wesentliche zu konzentrieren, belegen alle Beispiele nur die prinzipielle Machbarkeit und haben kein aufwändiges Layout oder ein GUI.

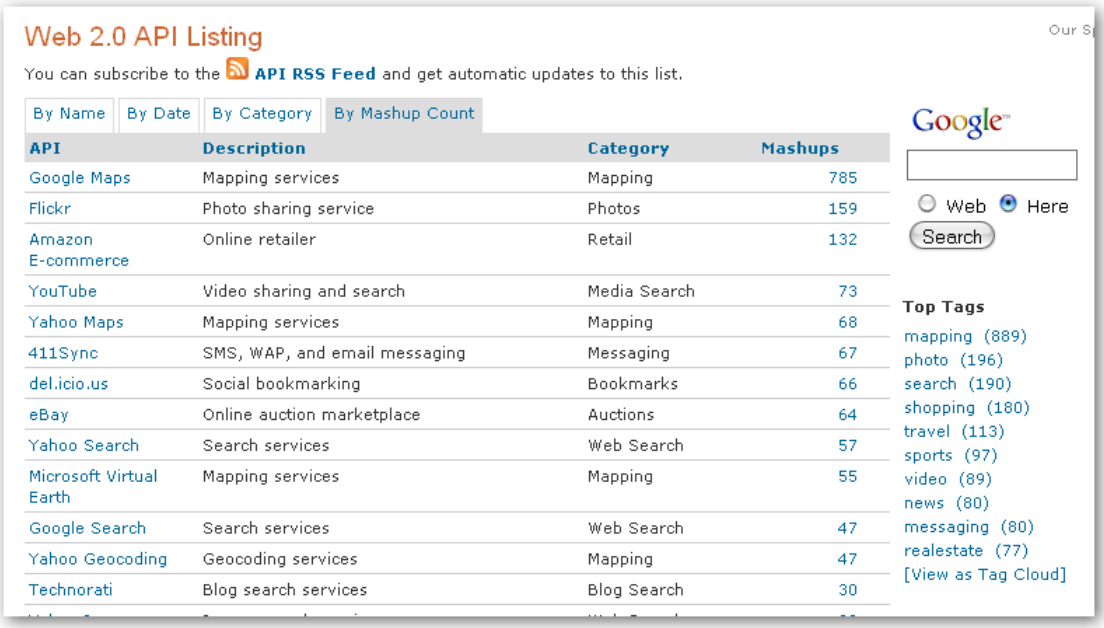
4.1 Klassifizierung von Web APIs

Im Top-Down Verfahren erarbeite ich eine Klassifizierung der momentan verbreiteten Web APIs mittels drei unterschiedlichen Kriterien:

- kumulative Häufigkeit
- gebührenpflichtige & gebührenfreie Web APIs
- Art des Web API:
  - Zugriff auf Daten
  - ein Interface bzw. Plattform
  - eine Dienst- oder Infrastrukturleistung

Anhand von drei Web API-Verzeichnissen (vgl. [Mus07a], [WSF06], [MaA07]) werde ich versuchen die *kumulative Häufigkeit* von Web API Kategorien festzustellen. Aufgrund der unterschiedlichen Benennung von Kategorien innerhalb der Verzeichnisse, die offensichtlich das Gleiche bedeuten, (z. B. „messaging“ vs. „mobile“ oder „retail“ vs. „shopping“) wird eine absolute Aussage erschwert. Trotzdem lassen sich Trends erkennen: Weit abgeschlagen vom Rest gibt es im Kartenbereich

(„Mapping“) sowohl die meisten Web APIs (vgl. ebenda) als auch auf deren Basis die meisten Mashups (s. Abb. 2 – rechter Rand). Google Maps ist dabei mit Abstand das am meisten verwendete API: Von insgesamt 889 Mashups, die mit „mapping“ getagged sind, entfallen 785 auf Google Maps (s. Abb. 2). Nach „Mapping“ sind die Themenbereiche „Photo“, „Search“, „Messaging“, „Shopping“, „News“, „Travel“, „Bookmarking“, „Blogging“ und „Music“ sehr häufig vertreten (vgl. ebenda). Betrachtet man die häufigsten Mashupkategorien, so fällt auf, dass die führenden Kategorien meistens auch direkt mit einem API korrelieren, d.h. oft ist ausschließlich ein API für die gute Platzierung verantwortlich (s. Abb. 2).



Web 2.0 API Listing

You can subscribe to the API RSS Feed and get automatic updates to this list.

By Name | By Date | By Category | By Mashup Count

API	Description	Category	Mashups
Google Maps	Mapping services	Mapping	785
Flickr	Photo sharing service	Photos	159
Amazon E-commerce	Online retailer	Retail	132
YouTube	Video sharing and search	Media Search	73
Yahoo Maps	Mapping services	Mapping	68
411Sync	SMS, WAP, and email messaging	Messaging	67
del.icio.us	Social bookmarking	Bookmarks	66
eBay	Online auction marketplace	Auctions	64
Yahoo Search	Search services	Web Search	57
Microsoft Virtual Earth	Mapping services	Mapping	55
Google Search	Search services	Web Search	47
Yahoo Geocoding	Geocoding services	Mapping	47
Technorati	Blog search services	Blog Search	30

Google

Web Here

Search

Top Tags

mapping (889)
photo (196)
search (190)
shopping (180)
travel (113)
sports (97)
video (89)
news (80)
messaging (80)
realestate (77)
[View as Tag Cloud]

APIs aufgelistet und sortiert nach der Anzahl an Mashups (http://programmableweb.com/apilist/bycat) Abb. 2

Eine Unterscheidung auf erster Ebene nach *gebührenpflichtigen und gebührenfreien Web APIs* ist ebenso möglich. Allerdings ist diese nicht sehr ergiebig. Gebührenfreie Services etablieren sich natürlich viel schneller und einfacher als gebührenpflichtige, dennoch existieren auch „Bezahl“-Web Services. Diese sind vor allem im Finanzbereich angesiedelt als auch bei „Dienstleistungs“-Services, wo über einfache Daten hinaus Ressourcen genutzt werden können (Rechner, Menschen).

Das letzte Kategorisierungskriterium stellt die *Art des Web API* bzw. des Web Services selbst dar:

- Bei den gewöhnlichen Web APIs werden nur „*rohe Daten*“ abgefragt, z. B. bei einer Preisanfrage an den E-Commerce Service von Amazon oder einer Anfrage nach Videos über das API von YouTube.
- Bei Web APIs, die über ein eingebettetes JavaScript funktionieren, werden keine reinen Daten bereitgestellt, sondern die direkte Funktionalität des Web APIs nutzbar gemacht. Google Maps stellt beispielsweise gleich ein Interface zur Verfügung anstatt lediglich Kartenausschnitte als Bilder.
- Es können auch gesamte Dienst- bzw. Infrastrukturleistungen in Anspruch genommen werden, so bieten beispielsweise einige Amazon Web Services an, große Rechnercluster zu mieten (Amazon EC2 – vgl. [AECoD]) oder Speicherkapazitäten im Gigabytebereich zu nutzen (Amazon S3 – vgl. [AS3oD]). Der Web Service „Amazon Mechanical Turk“ ([AMToD]) übernimmt sogar die Vermittlung an menschliche Arbeiter, basierend auf der Erkenntnis, dass bestimmte Aufgaben trotz ihre Einfachheit besser von Menschen als von Computern erledigt werden können. So könnte man beispielsweise eine sehr große Anzahl von Fotos nach spezifischen Kriterien sortieren lassen.

4.2 Web API-Anwendungen

Um nachvollziehen zu können, was alles mit Web APIs möglich ist, verweist man am Besten auf exemplarische Webseiten, die das Potenzial der Web APIs anschaulich aufzeigen. Dafür werde ich drei Webseiten vorstellen, die von unterschiedlichen Web APIs Gebrauch machen:

- elsewhere.adactio.com
- retrievr
- Amazon Light

Jeremy Keith, DOM Scripting Task Force Leader des Web Standards Project und Berater von Clearleft demonstriert mit seiner eigens eingerichteten Subdomain *http://elsewhere.adactio.com* den Einsatz mehrerer APIs. Er greift dabei auf das Web API von Flickr, den E-Commerce Service von Amazon, das del.icio.us API, das API von Upcoming.org und auf Feeds seiner Freunde und Kollegen zu. Die angeforderten

Daten werden mittels AJAX nachgeladen.

Eine andere Webapplikation, namentlich *retriever* (s. http://labs.systemone.at/retrievr/), das auf dem Flickr API basiert, zeigt anhand einer angegeben Farbe oder eines hochgeladenen Bildes farblich passende Bilder aus dem Datenbestand von Flickr an. Allerdings werden leider immer nur 20 Bilder angezeigt und somit ist die Anwendung nicht wirklich im Alltag einsatzfähig.

Amazon Light (s. http://www.kokogiak.com/amazon/) stellt einen abgespeckten Amazonshop dar, der aufgrund seiner Schlichtheit schnell und einfach zu bedienen ist. Mit dem E-Commerce Service (ECS) von Amazon lassen sich, wie man sieht, nahezu alle Daten der Amazon-Webseite abfragen.

4.3 Praktische Verwendung von drei unterschiedlichen Web APIs

Die nächsten drei Kapitel stellen die am Kapitelanfang angekündigten APIs/Web Services von *Amazon*, *Google* und *Flickr* vor. Jedes API fasse ich zu Beginn in einer Art Steckbrief zusammen, der einen groben Überblick über die grundlegenden Eigenschaften des APIs gibt. Mit diesen drei APIs habe ich die Pioniere sehr unterschiedlicher Kategorien ausgewählt: Mapping, Shopping und Photos (vgl. Kapitel 4.1). Darüber hinaus führen diese drei APIs die Mashupliste (s. Kapitel 5: Abb. 3) an, als am häufigsten verwendete APIs für Mashups. Die Kapitel schließen mit gängigen beispielhaften Szenarios ab.

4.3.1 Amazon E-Commerce Service (ECS)

Einstiegspunkt:	http://aws.amazon.com/ecs
API verfügbar seit:	Juli 2002[2]
Entwicklerzahl:	200.000 (Stand 20.12.2006)[3] (AWS allg.)
Kosten:	kostenlos[4]
Entwicklerblog:	http://aws.typepad.com (AWS allg.)
Dokumentation:	aktuell & umfassend[4]
Weiterentwicklung:	stetig[4] (8 Updates in 2006!)
Abwärtskompatibilität:	Versionierung / großzügige Portierungsfristen[1]
Lokalisierte Daten:	us, ca, uk, de, fr, jp (ECS)[1]
Community:	- wird aktiv von Amazon unterstützt - Sucess Stories von Websites, die API nutzen

	- Forum, das von Amazonmitarbeitern betreut wird
	- Newsletter von Amazon[4]
Verfügbare Daten:	- gesamter Produktkatalog (Preise, Produktbilder, Reviews, Verkaufsrang, ähnliche Produkte usw.)
	- Verfügbarkeit von Drtittanbietern
	- Zugriff auf den Amazon-Einkaufswagen
	- Preisentwicklung/-historie
	- Amazon Wunschlisten[1]
Nutzungsarten:	REST, SOAP/WSDL[1]
	XSLT-Verarbeitung (eigenes XSLT verwendbar)[1]
Responseformate:	XML, HTML, jedes XML-Format (XSLT)[1]

(vgl. [ECS06][1], [Bau03] S.191[2] ,[OB06][3] , http://aws.amazon.com[4],)

Mit dem E-Commerce Service (ECS) kann kostenlos auf sämtliche Daten zugegriffen werden, die auch über die Amazon-Produktseiten abrufbar sind. Darunter fallen z. B. Metadaten zu Produkten je nach Produktkategorie (Autor, Regisseur, Interpret usw.) sowie Preisinformationen, Kundenbewertungen, Wunschlisten uvm.

Szenario 1:
Ein Bibliothekssystem, das auf Java basiert, möchte Cover von Büchern direkt in einem systemeigenen Ergebnisfenster anzeigen.

> Nutzungsart: SOAP/WSDL
> Responseformat: SOAP
> Sprache: Java

Mit Java 6 wurde das Web Service Development Pack von Sun in die Java SE integriert. Dadurch ist es möglich, ohne Zuhilfenahme einer SOAP-Engine wie Apache Axis (s. http://ws.apache.org/) oder XFire (http://xfire.codehaus.org/) Web Services anzubieten und Clients für deren Verwendung zu schreiben. Die Klassen, die der Client zur Kommunikation mit dem Server benötigt, werden mit dem Tool `wsimport` (vgl. [Kne07]) generiert:

```
C:\mkdir ECS
C:\ECS\wsimport -d . -keep http://webservices.amazon.com/AWSECommerceService/
                                                    2006-11-14/DE/AWSECommerceService.wsdl
```

Es ist empfehlenswert, sich gleich für ein lokalisierte (`DE`) Variante zu entscheiden und auch eine explizite Version (`2006-11-14`) anzugeben, da ohne diese Angabe das aktuellste WSDL-Dokument abgefragt wird und es irgendwann zu Inkompatibilitäten kommen könnte (vgl. [ECS06], S. 434 ff.). Der Parameter `-d` gibt dabei das Zielverzeichnis an und `-keep` sorgt dafür, dass die Klassen erhalten bleiben. Die erzeugten Klassen müssen nun in das eigene Client-Projekt kopiert werden.

Wichtig zu wissen ist, dass jedes Produkt bei Amazon eine Amazon Standard Item Number (ASIN) besitzt. Diese fällt bei Büchern mit deren International Standard Book Number (ISBN) zusammen (vgl. [Bau03], S.5 ff.). Deshalb wird nur eine gültige ISBN benötigt, um Daten zu einem bestimmten Buch abzufragen.

```
import java.net.MalformedURLException;
import java.util.List;
import javax.xml.ws.Holder;
import com.amazon.webservices.awsecommerceservice._2006_11_14.*;

public class BookCoverRetriever {
    public static void main(String[] args) throws MalformedURLException {

        // Das PortType-Objekt kapselt die SOAP-Kommunikation
        AWSECommerceService aws_ecs = new AWSECommerceService();
        AWSECommerceServicePortType p = aws_ecs.getAWSECommerceServicePort();

        // Eine Anfrage wird erzeugt
        ItemLookupRequest itLoReq = new ItemLookupRequest();
        // Die ASIN wird der Anfrage hinzugefügt
        // hier: „Der Schwarm“ von Frank Schätzing
        itLoReq.getItemId().add("3596164532");
        // Die Ergebnismenge wird auf Bilder beschränkt
        itLoReq.getResponseGroup().add("Images");

        // Ein Containerobjekt für den Response wird erzeugt
        Holder<List<Items>> hl = new Holder<List<Items>>();

        // Diese Parameter können bei Bedarf gesetzt werden
        String marketplaceDomain = "";
```

```
        String subscriptionId = "";
        String associateTag = "";
        String validate = "";
        String xmlEscaping = "";
        // ItemLookupRequest shared = null;
        // List<ItemLookupRequest> itLoRequestList = null;

        // Eine Anfrage wird erzeugt.
        // Da die Methode keinen Rückgabewert hat, wird ein Containerobjekt
        // übergeben (hl), das die gewünschten Daten aufnimmt.
        p.itemLookup(marketplaceDomain, "[api_key]", subscriptionId,
                                        associateTag, validate, xmlEscaping,
                                                itLoReq, null, null, hl);

        // Aufgrund der Möglichkeit, gleich eine Liste von Requests zu
        // übergeben, ist das Holderobjekt sehr tief verschachtelt.
        System.out.println(hl.value.get(0).getItem().get(0).getImageSets().
                        get(0).getImageSet().get(0).getLargeImage().getURL());
    }
}
```

Auf der Konsole wird folgende URL ausgegeben:

```
http://ec1.images-amazon.com/images/P/3596164532.01._SCLZZZZZZZ_V38662278_.jpg
```

Zusätzlich sollen ähnliche Bücher angezeigt werden, die ggf. in Verbindung mit dem angezeigten Buch stehen, da sie ähnliche Themenbereiche behandeln.

> Nutzungsart: REST
> Responseformat: XML
> Sprache: Java

Bei der Nutzung von REST müssen lediglich mehrere Parameter zu einer URL zusammengebaut werden, die dann als HTTP-Anfrage abgeschickt wird. Das empfangene XML-Dokument muss dann bezüglich der benötigten Informationen geparst werden.

```
import java.io.*;
import java.net.*;
import javax.xml.stream.*;
import javax.xml.stream.events.XMLEvent;
```

```
public class BookSimilaritiesRetriever {

    // Natürlich darf der URL keine Leerstellen enthalten, er ist hier nur zu
    // Anschauungszwecken aufgetrennt. Die Bedeutung der einzelnen Elemente
    // entspricht der SOAP-Variante (s. vorheriges Beispiel)
    private static final String URL = "http://ecs.amazonaws.de/onca/xml
                          ?Service=AWSECommerceService&Version=2006-11-14
                          &Operation=ItemLookup
                          &ContentType=text%2Fxml
                          &AWSAccessKeyId=[api_key]
                          &ItemId=3596164532
                          &ResponseGroup=Similarities";

    public static void main(String[] args)
              throws MalformedURLException, IOException, XMLStreamException {
        URL url = new URL(URL);
        InputStream in = url.openStream();

        // Ein StreamReader-Objekt wird erzeugt
        XMLInputFactory factory = XMLInputFactory.newInstance();
        XMLStreamReader parser = factory.createXMLStreamReader(in);

        // Mittels des StreamReader-Objekts werden die Elemente des
        // empfangenen XML-Dokuments in einem Pull-Verfahren einzeln
        // abgerufen und geprüft (vgl. [Ull06])
        while (parser.hasNext()) {
            int event = parser.next();
            if (event == XMLStreamConstants.START_ELEMENT &&
                                parser.getName().getLocalPart() == "Title") {
                System.out.println(parser.getElementText());
            }
        }
        parser.close();
    }
}
```

Auf der Konsole werden ähnliche Artikel ausgegeben:

```
Sakrileg. The Da Vinci Code
Der Schatten des Windes.
Illuminati.
Tod und Teufel.
Lautlos.
```

Anmerkung:
Die `while-Schleife`, die den Zugriff auf das empfangene XML-Dokument erlaubt, könnte z. B. durch die Verwendung eines Filters einen Performancegewinn erreichen. Es muss klar sein, dass die zugrunde liegende Struktur des XML-Dokumentes zunächst analysiert werden muss, bevor es programmatisch verarbeitet werden kann.

4.3.2 Google Maps

Einstiegspunkt:	http://www.google.com/apis/maps/
erste API(s) verfügbar:	Juni 2005[1]
Entwicklerzahl:	-
Kosten:	kostenlos
Entwicklerblog:	http://googlemapsapi.blogspot.com/
Dokumentation:	tutorialartiger Einstieg, viele Beispiele, API-Referenz[3]
Weiterentwicklung:	stetig (verbessertes Kartenmaterial, neue Features)
Abwärtskompatibilität:	wird nicht gewährleistet[2]
Lokalisierte Daten:	globales Kartenmaterial, unterschiedliche Detailstufen
Community:	Forum[3]
Verfügbare Daten:	Kartenmaterial
Nutzungsarten:	einzubettendes JavaScript
Responseformate:	keine Web API im „klassischen“ Sinn (s.u.)

(vgl. [GE06][1] S. 35, http://www.google.com/apis/maps/terms.html[2], http://www.google.com/apis/maps/[3])

Nicht die Tatsache, dass es zum ersten Mal programmatisch nutzbares Kartenmaterial im Internet gab, war der Auslöser für die erstaunlich schnell gestiegene Popularität von Google Maps sondern das intelligente Nachladen der Kartenausschnitte mittels AJAX. Sogar noch bevor ein API überhaupt veröffentlicht wurde, gab es schon erste Mashups auf der Basis von Google Maps (vgl. [Gra05]). Die simple API und das schlanke User Interface verhalfen dann zu der endgültigen Popularität (vgl. [GE06], Vorwort).

Kartenmaterial, also die zusätzliche Dimension der Ortsangabe, bietet sich geradezu an, mit anderen Daten überlagert zu werden, um eine visualisierte Aufbereitung oder sogar eine Benutzeroberfläche zu den Daten bereitzustellen.

Google Maps ist kein gewöhnliches Web API (s. Amazon und Flickr), sondern eher ein API im klassischen Sinn der Softwareentwicklung. Anstatt auf Web Services zuzugreifen, wird ein JavaScript referenziert, in dem die benötigten Methodenaufrufe und Objekte erzeugt und aufgerufen werden können. Außerdem erhält man keine rohen Daten (also zum Beispiel einzelne Kacheln von einen Kartenausschnitt), sondern gleich ein Interface mitgeliefert.

Szenario 2:
Ein renommierter Experte möchte auf seiner Webseite seine nächsten Vortragsorte (Mannheim, Karlsruhe, Heilbronn) mit Bild und den zugehörigen Daten (Datum, Uhrzeit, Thema) veröffentlichen.

Anders als bei anderen APIs muss bei Google Maps für jedes Verzeichnis (z B. http://www.example.com/dir), in dem das API genutzt wird, ein neuer Key angefordert werden. Der Key gilt dann auch für die Unterverzeichnisse. Solange man nur auf seinem eigenen Rechner entwickelt, kann dieser ignoriert werden – und somit dieses Beispiel direkt ausgeführt werden.

Der Verweis zum externen JavaScript enthält die Version des API (`v=2`) und den Key.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="de" xml:lang="de">
  <head>
    <title>Vortragsorte</title>
    <style type="text/css">
      #map {
        width: 800px;
        height: 500px;
        font-family: verdana;
        font-size: 12px;
      }
      #map img {
        margin-top: 15px;
      }
    </style>
    <script src="http://maps.google.com/maps
                          ?file=api
```

```
                    &v=2
                    &key=[api_key]" type="text/javascript"></script>
<script type="text/javascript">
//<![CDATA[
  var map = null;

  // Funktion, um die Marker zu setzen und das Registrieren
  // eines EventListeners vorzunehmen
  function createMarker(point, html) {
    var marker = new GMarker(point);
    GEvent.addListener(marker, "click", function() {
      marker.openInfoWindowHtml(html);
    });
    return marker;
  }

  function load() {
    if (GBrowserIsCompatible()) {
      map = new GMap2(document.getElementById("map"));

      // Initialer Startpunkt (Längen- und Breitengrad) und Zoomstufe
      map.setCenter(new GLatLng(49.222979, 8.876953), 9);

      // schrittweise Skalierungsmöglichkeit
      map.addControl(new GLargeMapControl());

      // Alle drei Kartentypen: Satellit, Straßenkarte, Hybrid
      map.addControl(new GMapTypeControl());

      // Maßstabsanzeige
      map.addControl(new GScaleControl());

      // Übersichtskarte in der rechten unteren Ecke
      map.addControl(new GOverviewMapControl());

      // Längen- und Breitengrade von Mannheim, Karlsruhe und Heilbronn
      var points = [
                    new GLatLng(49.490208, 8.476639),
                    new GLatLng(49.147131, 9.221992),
                    new GLatLng(49.01378, 8.40385)
                  ];
      // die darzustellenden Inhalte
      var htmlContent = [
                    "<strong>Datum:<\/strong> 30.05.2007<br/>"
```

```
                        +"<strong>Thema:<\/strong> eXtreme Programming<br/>"
                        +"<strong>Uhrzeit:<\/strong> 08 Uhr - 14 Uhr<br/>"
                        +"<img src='mh.jpg' alt='Mannheim'/>",

                        "<strong>Datum:<\/strong> 16.07.2007<br/>"
                        +"<strong>Thema:<\/strong> Web Services & SOA<br/>"
                        +"<strong>Uhrzeit:<\/strong> 09 Uhr - 12 Uhr<br/>"
                        +"<img src='hb.jpg' alt='Heilbronn'/>",

                        "<strong>Datum:<\/strong> 23.09.2007<br/>"
                        +"<strong>Thema:<\/strong> MDA Tools<br/>"
                        +"<strong>Uhrzeit:<\/strong> 08 Uhr - 16 Uhr<br/>"
                        +"<img src='ka.jpg' alt='Karlsruhe'/>",
                      ];

          for (var i=0; i<3; i++) {
            // Setzen der Marker und des Inhalts
            map.addOverlay(createMarker(points[i],htmlContent[i]));
          }

          // Blaue Verbindungslinie zwischen den Orten ziehen
          var polyline = new GPolyline([
            points[0],
            points[1],
            points[2]
          ], "#0000FF", 10);
          map.addOverlay(polyline);
        }
      }
    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <div id="map"></div>
  </body>
</html>
```

Anmerkung:

Der Einfachheit halber sind die drei Orte direkt im Quelltext vorgegeben und auf die `width`- und `height`-Attribute des `<img>`-Elements wurde verzichtet. Über ein `Geocoder`-Objekt wäre es auch möglich, aufgrund von Adressen die richtigen Breiten- und Längengrade abzufragen. Allerdings wird dazu ein gültiger API-Code benötigt.

4.3.3 Flickr API

Einstiegspunkt:	http://flickr.com/services/
API verfügbar seit:	Februar 2005[3]
Entwicklerzahl:	-
Kosten:	„availible for non-commercial use“ / „Commercial use is possible by prior arrangement“
Entwicklerblog:	nur allgemeines Flickr-Blog: http://blog.flickr.com/
Dokumentation:	kurz, prägnant und aktuell[1] [2]
Weiterentwicklung:	Betastadium, keine Garantie der Weiterentwicklung[1]
Abwärtskompatibilität:	-
Lokalisierte Daten:	es gibt bisher nur den englischen Ableger
Community:	- es existiert eine Mailingliste und ein Forum - Verlinkung von API Kits (abstrahierende Bibliotheken) - Vorstellung einiger „Third Party Apps“[1] [2]
Verfügbare Daten:	auf Daten der Websites beschränkt[2]
Nutzungsarten:	REST / XML-RPC / SOAP[2]
Responseformate:	XML / JSON / PHP[2]

(vgl. http://flickr.com/services/[1], http://flickr.com/services/api[2], [SBC05][3])

Schon vor der Übernahme durch Yahoo im Jahre 2006 für ca. 30 Millionen Dollar (vgl. [War06b]) ist Flickr eine sehr bekannte und beliebte Fotocommunity gewesen. Als registrierter Nutzer kann man bei Flickr Fotos hochladen, diese taggen (mit Stichwörtern versehen) und Kontaktlisten pflegen. Die Bilder können dabei zusammenhängend in Gruppen organisiert werden und von anderen Flickrnutzern kommentiert werden. Anhand der vergebenen Tags kann nach Bildern gesucht werden.

All diese Informationen sind über das API verfügbar. Anfangs ist es wichtig die Links im Bereich mit der Überschrift „read these first“ auf der API-Seite zu besuchen. Die URLs zu den Bildern werden nämlich nicht direkt im Response übergeben, sondern müssen aus Attributwerten konstruiert werden. Dies wird im folgenden Beispiel illustriert.

Szenario 3:
Eine Privatperson möchte mit wechselnden eindrucksvollen Bildern von Flickr seine Webseite bereichern.

> Nutzungsart: REST
> Responseformat: XML
> Sprache: PHP

Um regelmäßig qualitativ hochwertig Fotos abzufragen, kann die Methode `flickr.interestingness.getList` verwendet werden. Flickr besitzt einen eigenen Algorithmus, der Bilder als interessant auszeichnet (vgl. [Fli07]).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="de" xml:lang="de">
  <head>
    <title>Flickr-Bilder</title>
    <style type="text/css">
       /* dies verschönert die Ausgabe rudimentär */
      body {
        width: 500px;
        margin: 20px 0 0 20px;
      }
      img {
        padding: 4px;
        border-color: #cecece;
        margin: 10px;
      }
    </style>
  </head>
  <body>
    <?php
      // Die Leerzeichen müssen vor dem Einsatz entfernt werden. Diese
      // Darstellung dient nur der besseren Lesbarkeit
      define ('URL', 'http://api.flickr.com/services/rest/
                          ?method=flickr.interestingness.getList
                          &api_key=[api_key]');

      $rsp = file_get_contents(URL);
      $xml = new SimpleXMLElement($rsp);
```

```
        // So sehen die zurückgegebenen Photo-Elemente aus
        /*
          <photo id="336205807"
                 owner="74379771@N00"
                 secret="ef04d6429a"
                 server="142"
                 farm="1"
                 title="What becomes of the broken hearted?"
                 ispublic="1"
                 isfriend="0"
                 isfamily="0"/>
        */

        // So muss der finale URL zu den Photos aussehen
        // http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}.jpg
        for ($i=0; $i<=5; $i++) {
          $finalURL = 'http://farm1.static.flickr.com/';
           // mittels der SimpleXML-Extension kann sehr einfach auf die Elemente
          // & Attribute zugegriffen werden, um den finalen URL zu konstruieren.
          $finalURL .= $xml->photos->photo[$i]['server'] . '/';
          $finalURL .= $xml->photos->photo[$i]['id'] . '_';
          $finalURL .= $xml->photos->photo[$i]['secret'] . '.jpg';
          $title = $xml->photos->photo[$i]['title'];
           // Schließlich werden die Bilder verkleinert ...
          $dim = getimagesize($finalURL);
          $width = round($dim[0] * 0.25);
          $height = round($dim[1] * 0.25);
           // ... und ausgegegeben
          $img = "<a href='$finalURL'>
          <img src='$finalURL' width='$width' height='$height'
                                                    alt='' title='$title'/>
          </a>\n";
          print($img);
        }
      ?>
    </body>
</html>
```

Anmerkung:

Da die Flickr Bilder meistens nur sehr langsam geladen werden, macht es Sinn diese zu cachen. In vielen Programmiersprachen existieren bereits API-Kits, die z. B. dem Nutzer die Arbeit der URL-Konstruktion oder des Cachings abnehmen. Die Kits sind auf den Seiten von Flickr verlinkt.

5 Mashups – Web APIs miteinander kombiniert

Nachdem nun eine grobe Klassifizierung, die Vorstellung existierender Web-Applikationen, die APIs einsetzen und exemplarische Szenarios, die Schritt für Schritt die Verwendung der APIs zeigen, dargelegt wurde, kann jetzt darauf aufbauend der Begriff „Mashup“ erläutert werden. Dieser wird zuerst *definiert* und danach *klassifiziert.* Abschließend werden *Nutzungsdaten* betrachtet und *prominente Mashups* vorgestellt.

> „A web mashup is a web page or application that combines data from two or more external online sources. The external sources are typically other web sites and their data may be obtained by the mashup developer in various ways including, but not limited to: APIs, XML feeds, and screen-scraping.“ ([Mus07b])

Der Begriff „Mashup“ (semantisch übersetzt: „Kombination“ / „Vermischung“) bezeichnet eine neue Art von Web Applikationen. Im Gegensatz zu Webseiten, die nur ein Web API nutzen, kombinieren Mashups Inhalte aus mehreren voneinander unabhängigen Quellen. Als potentielle Quellen kommen neben Web APIs auch Feeds (z. B. RSS oder ATOM) und in Zukunft wahrscheinlich vermehrt Microformats ([Mic07]) in Frage.

Die Etymologie des Begriffs erleichtert das Verständnis: Entliehen aus der Pop-Musik Szene bezeichnet er dort einen Remix, der sich Elementen vorhandener Songs bedient und dadurch eine neue Komposition schafft. (vgl. [Mer06])

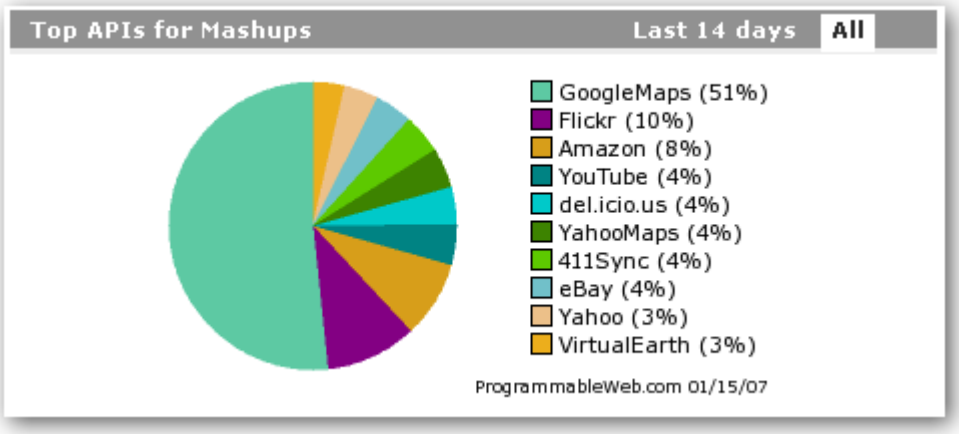
Merrill (vgl. ebenda) ordnet Mashups vier grundlegende Kategorien zu:

- Mapping mashups
- Video and photo mashups
- Search and Shopping mashups
- News mashups

Während bei *Mapping Mashups* Kartenmaterial mit geografisch annotierten Daten aufbereitet wird, bilden *Video und photo Mashups* die Wirklichkeit in unterschiedlicher Art und Weise ab und erlauben den Zugriff über die Metadaten des Mediums. Unter *Search and Shopping Mashups* können beispielsweise Preisvergleich-Applikationen verstanden werden oder Mashups, die sich u. a. mit Daten von Amazon oder eBay

versorgen. *News Mashups* bedienen sich Web Feeds und erzeugen mit den aggregierten Meldungen einen Katalog übergreifender Pressestimmen oder z. B. eine individuell zusammengestellte Informationszentrale.

Die Website ProgrammableWeb ([Mus06]) listet 358 APIs und 1449 Mashups (Stand 16.01.2007). Das ist beachtlich, wenn man bedenkt, dass die meisten Web APIs erst seit ein paar Jahren existieren. Täglich kommen ca. 3 Mashups hinzu (vgl. ebenda).



Die meist genutzten APIs für Mashups (http://programmableweb.com/apis) Abb. 3

Kartenmaterial bietet sich geradezu an, mit einer informativen Datenbasis überlagert zu werden, um komplexe textuelle Informationen durch eine visuelle Darstellung intuitiv verständlich zu machen. So ist es nicht verwunderlich, dass Google Maps bisher sehr viele Entwickler (51%) zu Mashups inspirierte (s. Abb. 2) und einen Mashup-Boom entfachte (s. http://googlemapsmania.blogspot.com/). Auch die Mapping-Konkurrenz (Yahoo! Maps, Microsoft Virtual Earth) ist unter den Top Ten vertreten. Flickr ist mit 10% aufgrund des Fachgebietes interessant: Mit Fotos, die die Realität abbilden, bietet sich eine interessante Datenquelle an. Amazon, dritter dieser Statistik (8%), hat seine Stellung wahrscheinlich dem Umfangs seiner veröffentlichen Produktdatenbank und den innovativen Web Services zu verdanken.

Abschließend eine Auswahl prominenter Mashups:
„Housing Maps“ (s. http://www.housingmaps.com/), stellt Mietangebote von Craigslist.com, eine Seite für Online-Anzeigen, auf dem von Google Maps bereitgestellten Kartenmaterial dar und erlaubt einen direkten Vergleich auf Basis der Lage der Wohnungen.

Schienennetze bieten sich ebenfalls an visualisiert zu werden. So gibt es bereits eine interaktive „Subway Map“ von New York (s. http://www.onnyturf.com/subway/), die Haltestellen anzeigt und auf Abfahrtszeiten verlinkt. Analog dazu existiert eine S-Bahn Karte von Dublin, die zusätzlich zu den Haltestellen in Echtzeit die Positionen der Züge anzeigt (s. http://dartmaps.mackers.com/).

Nach den vielen Mashups, die auf Kartenmaterial basieren, nun ein Mashup, das auf den APIs von YouTube und Amazons E-Commerce Service aufbaut: Zontube (s. http://pulpsite.net/zontube/). Bei diesem Mashup können die Seiten von Amazon US, UK und Japan nach Musik durchsucht werden und YouTube liefert die dazu passend getaggten Videos – von offiziellen Videos und Liveauftritten bis hin zu (Amateur-) Cover-Versionen.

Die personalisierbaren Startseiten von Google (http://www.google.de) und Windows Live (http://www.live.com/) sowie die Webseiten Netvibes (http://www.netvibes.com/) oder Pageflakes (http://www.pageflakes.com/) erlauben dagegen die Zusammenstellung einer individuellen „Informationszentrale“ (s. News Mashups) und nutzen damit die Möglichkeit der losen Kopplung von Web Services konsequent aus.

Weitere Beispiele finden die interessierten Leser unter http://programmableweb.com/.

6 Fazit und Ausblick

Mit der vorliegenden Arbeit liegt eine Einführung in Web APIs vor, welche deren Nutzung sowohl kritisch beleuchtet als auch exemplarisch deren geringe Einstiegshürden in der praktischen Verwendung aufzeigt. Es wurde eine Klassifizierung von Web APIs vorgenommen und Applikationen vorgestellt, die ein API oder mehrere gleichzeitig als Mashup nutzen und daraus einen Mehrwert generieren.

Sofern sich in Zukunft vermehrt kostenpflichtige Web Services, wie bei Amazon, oder sich generell „more sophisticated“ Web Services etablieren und Eigenschaften, wie eine Abrechnungsmöglichkeit, transaktionales Verhalten und Sicherheit benötigt werden, könnte die Kombination SOAP/WSDL wieder häufiger von den API-Betreibern angeboten werden als momentan REST.

Meta-APIs, die direkt über (gleichartigen) APIs ansetzen und einen Umstieg auf ein direktes Konkurrenz-API mit minimalem Aufwand ermöglichen, könnten ein Trend werden. Dadurch haben Anwender eine größere Sicherheit, den angebotenen Service auch langfristig nutzen zu können. So ist Mapstraction (vgl. [Map06]) bspw. ein Meta-API für Google Maps, Yahoo! Maps und Microsoft Virtual Earth.

Da jedes API immer nur einen restriktiven Zugriff auf Daten zulässt, existieren bereits Projekte, die es sich zum Ziel gesetzt haben, offene Datenarchive aufzubauen. So erlaubt OpenStreetMap bspw. einen Zugriff auf freie geografische Daten (vgl. [OSM07]) und FreeThePostcode! stellt passende Breiten- und Längengrade von Postleitzahlen in England zur Verfügung (vgl. [FTPoD]). Diese Projekte könnten abhängig vom Verhalten der API-Anbieter bzgl. sich ändernder Nutzungsbedingungen ihrer Web Services stärker genutzt und dementsprechend unterstützt werden.

Ein weiterer signifikanter Trend ist die wachsende Verbreitung und Verarbeitung von Microformats. Damit lassen sich Teile einer Webseite semantisch annotieren und eine programmatische Verarbeitung dieser wird ermöglicht (vgl. [Mic07]). So soll die dritte Version des Mozilla Firefox die Rolle eines Information Broker einnehmen, der mit Microformats ausgezeichnete Informationen direkt in externe Applikationen (z. B. den Kalender oder das Adressbuch) exportieren kann (vgl. [Faa06] und [McM07]). Dadurch könnte sich das gesamte Web langsam in ein gigantisches API (vgl. [Kei06]) – und bereits grob in die Richtung eines „semantischen Webs“ (vgl. [Her07]) entwickeln.

A Quellen- und Literaturverzeichnis

[AECoD] k. A.: Amazon EC2, Amazon Elastic Compute Cloud, Online im Internet, http://www.amazon.com/ec2/ v. k. A., Abfrage vom 31.01.07

[AMToD] k. A.: Amazon Mechanical Turk, Online im Internet, http://www.amazon.com/mturk/ v. k. A., Abfrage vom 31.01.07

[And06] Anderson, T.: WS-* vs the REST, Online im Internet, http://www.regdeveloper.co.uk/2006/04/29/oreilly_amazon/ v. 29.04.06, Abfrage vom 16.01.07

[AS3oD] k. A.: Amazon S3, Amazon Simple Storage Service, Online im Internet, http://www.amazon.com/s3/ v. k. A., Abfrage vom 31.01.07

[AZS06] k. A.: ARD/ZDF-Online-Studie 2006, Online im Internet, http://www.daserste.de/service/studie.asp v. 2006, Abfrage vom 20.01.07

[Bal06] Balkan, A.: d.Construc 2006 - Mash my Flex up, Online im Internet, http://aralbalkan.com/presentations/dconstruct2006/Mash_my_Flex_up.pdf v. 09.09.06, Abfrage vom 16.01.07

[Bau03] Bausch, P.: Amazon Hacks, O'Reilly Media, CA Sebastopol 2003

[Cla05] Claburn T.: APIs Make Money For Amazon, Online im Internet, http://informationweek.com/showArticle.jhtml?articleID=172302181 v. 18.10.2005, Abfrage vom 08.01.2007

[Cro06] Crockford, D.: Introducing JSON, Online im Internet, http://json.org/ v. 2006, Abfrage vom 16.01.07

[CS03] Claus, V. / Schwill, A.: Duden Informatik, Bibliographisches Institut, Mannheim 2003

[DJMZ05] Dostal M. / Jeckle M. / Melzer I. / u. a.: Service-orientierte Architekturen mit Web Services, Spektrum Akademischer Verlag, München 2005

[ECS06] k. A.: Amazon E-Commerce Service - Developer Guide, Amazon, 2006

[Faa06] Faaborg, A.: Microformats - Part 0: Introduction, Online im Internet, http://blog.mozilla.com/faaborg/2006/12/11/microformats-part-0-introduction v. 11.12.06, Abfrage vom 25.01.07

[Fie00] Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures, Diss., Irvine 2000

[Fli07] k. A.: Flickr: Explore interesting photos around Flickr, Online im Internet, http://www.flickr.com/explore/interesting/ v. 2007, Abfrage vom 31.01.07

[Fow05] Fowler, M.: The New Methodology, Online im Internet, http://www.martinfowler.com/articles/newMethodology.html v. 13.12.05,

Abfrage vom 10.02.07

[FTPoD] k. A.: Free The Postcode!, Online im Internet, http://www.freethepostcode.org/ v. k. A., Abfrage vom 02.02.07

[GE06] Gibson, R / Erle S.: Google Maps Hacks, O'Reilly Media, CA Sebastopol 2006

[GMEoD] k. A.: Enterprise Solutions: Google Maps - FAQ, Online im Internet, http://www.google.de/enterprise/maps/faq.html v. k. A., Abfrage vom 16.01.07

[GMT05] k. A.: Google Maps API Terms of Use, Online im Internet, http://www.google.com/apis/maps/terms.html v. k. A., Abfrage vom 16.01.07

[GMToD] k. A.: Google Maps API Terms of Use, Online im Internet, http://www.google.com/apis/maps/terms.html v. k. A., Abfrage vom 16.01.07

[Gra05] Graham, P.: Web 2.0, Online im Internet, http://www.paulgraham.com/web20.html v. November 2005, Abfrage vom 22.01.07

[GSS06] k. A.: Google SOAP Search API (Beta), Online im Internet, http://code.google.com/apis/soapsearch/ v. 05.12.06, Abfrage vom 12.02.07

[Her07] Herman, I.: W3C Semantic Web Activity, Online im Internet, http://www.w3.org/2001/sw/ v. 04.01.07, Abfrage vom 02.02.07

[Ive04] Iverson, W.: Real World Web Services, O'Reilly Media, CA Sebastopol 2004

[Kei06] Keith, J.: Adactio: Articles - The Joy of API, Online im Internet, http://adactio.com/articles/1181/ v. 06.10.06, Abfrage vom 16.01.07

[Kne07] Knecht, J.: WS4Java6, in: Javamagazin, Januar 2007, S. 52 - 54

[LBL06] Laningham, S. / Berners-Lee, T.: developerWorks Interviews: Tim Berners-Lee, Online im Internet, http://www-128.ibm.com/developerworks/podcast/dwi/cm-int082206.txt v. 28.07.06, Abfrage vom 27.01.07

[MaA07] k. A.: Mashup APIs, Online im Internet, http://www.webmashup.com/Mashup_APIs/index.html v. 29.01.07, Abfrage vom 29.01.07

[Map06] k. A.: Mapstraction - a javascript library to hide differ, Online im Internet, http://www.mapstraction.com/ v. 2006, Abfrage vom 31.01.07

[McK06] McKendrick, J.: Mashup vs. SOA app: what's the difference?, Online im Internet, http://blogs.zdnet.com/service-oriented/?p=647 v. 21.06.06, Abfrage vom 14.01.2007

[McM07] McManus, R.: Mozilla Does Microformats, Online im Internet, http://www.readwriteweb.com/archives/mozilla_does_microformats_firefox3.php v. 02.01.07, Abfrage vom 02.02.07

[McP05] MacManus, R. / Porter, J.: Digital Web Magazine - Web 2.0 for Designers, Online im Internet, http://www.digital-web.com/articles/web_2_for_designers/ v. 04.05.05, Abfrage vom 22.01.07

[Mer06] Merrill, D.: Mashups: The new breed of Web app, Online im Internet, http://www-128.ibm.com/developerworks/web/library/x-mashups.html?ca=drs- v. 16.10.06, Abfrage vom 02.01.07

[Mic07] k. A.: microfromats, Online im Internet, http://microformats.org/ v. 2007, Abfrage vom 31.01.07

[MTO06] John, M. / O'Reilly, T. / O'Reilly Radar Team: Web 2.0 - Principles and Best Practices (Excerpt), Online im Internet, http://www.oreilly.com/catalog/web2report/chapter/web20_report_excerpt.pdf v. November 2006, Aufruf vom 16.01.07

[Mus06] Musser, J.: ProgrammableWeb: Mashups and the Web as Platform, Online im Internet, http://www.programmableweb.com/ v. 15.01.07, Abfrage vom 16.01.07

[Mus07a] Musser, J.: Web 2.0 API Listing, Online im Internet, http://programmableweb.com/apilist/bycat v. 29.01.07, Abfrage vom 29.01.07

[Mus07b] Musser, J.: ProgrammableWeb: FAQ, Online im Internet, http://www.programmableweb.com/faq v. 02.01.07, Abfrage vom 02.01.07

[NL05] Newcomer, E. / Lomow, G.: Understanding SOA with Web Services, Addison-Wesley, Hagerstown 2005

[OB06] O'Reilly, T. / Bezos, J.: Web 2.0 Podcast: A Conversation with Jeff Bezos, Online im Internet, http://www.oreillynet.com/pub/a/network/2006/12/20/web-20-bezos.html v. 20.12.2006, Abfrage vom 04.01.2007

[ORe02] O'Reilly, T.: Amazon Web Services API, Online im Internet, http://www.oreillynet.com/pub/wlg/1707?wlg=yes v. 18.07.02, Abfrage vom 23.01.07

[ORe05a] O'Reilly, T.: What Is Web 2.0, Online im Internet, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1 v. 30.09.05, Abfrage vom 16.01.07

[ORe05b] O'Reilly, T.: Web 2.0: Compact Definition?, Online im Internet, http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html v. 01.10.05, Abfrage vom 21.01.07

[ORe06] O'Reilly, T.: Web 2.0 Compact Definition: Trying Again, Online im Internet, http://radar.oreilly.com/archives/2006/12/web_20_compact.html v. 10.12.06, Abfrage vom 21.01.07

[ORe07] O'Reilly, T.: Web 2.0 Most Cited Wikipedia Entry of the Year, Online im Internet, http://radar.oreilly.com/archives/2007/01/web_20_most_cit.html v. 03.01.2007, Abfrage vom 13.01.2007

[OSM07] k. A.: OpenStreetMap, Online im Internet, http://wiki.openstreetmap.org/index.php/Main_Page v. 13.01.07, Abfrage vom 02.01.07

[POS07] k. A.: Objekte serialisieren - Objekte in Sessions, Online im Internet, http://www.php.net/manual/de/language.oop.serialization.php v. 09.01.07, Abfrage vom 27.01.07

[PSX07] k. A.: SimpleXML Funktionen, Online im Internet, http://de.php.net/manual/de/ref.simplexml.php v. 09.01.07, Abfrage vom 27.01.07

[RH05] Richter, J.-P. / Haller Harald: Informatiklexikon: Serviceorientierte Architektur, Online im Internet, http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/118/ v. 2005, Abfrage vom 16.01.07

[SBC05] k. A.: SBC DotNet Weblog : Flickr Service has an API, Online im Internet, http://weblogs.asp.net/sbchatterjee/archive/2005/02/13/372024.aspx v. 13.02.05, Abfrage vom 16.01.2007

[Sch03] Schmidt, H.: Ebay ist Arbeitgeber für über 10.000 Menschen, Online im Internet, http://ockenfels.uni-koeln.de/download/press/faz-18082003.pdf v. 18.08.2003, Abfrage vom 16.01.07

[Sha05] Shaw, R.: Web 2.0? It doesn't exist, Online im Internet, http://blogs.zdnet.com/ip-telephony/?p=805 v. 17.12.05, Abfrage vom 22.01.07

[Spo06] Spool, J. M.: Web 2.0: The Power Behind the Hype, Online im Internet, http://www.uie.com/articles/web_2_power/ v. 30.11.2006, Abfrage vom

22.01.07

[StA06] k. A.: JSR 173: Streaming API for XML, Online im Internet, http://www.jcp.org/en/jsr/detail?id=173 v. 29.09.06, Abfrage vom 27.01.07

[Sto06] Stöcker, C.: Zerreiß mich, kopier mich, Online im Internet, http://www.spiegel.de/netzwelt/web/0,1518,411147,00.html v. 13.04.07, Abfrage vom 08.01.07

[Sur04] Surowiecki, J.: The Wisdom of Crowds, Online im Internet, http://www.randomhouse.com/features/wisdomofcrowds/ v. 2004, Abfrage vom 22.01.07

[Ull06] Ullenboom, C.: Java ist auch eine Insel - Serielle Verarbeitung mit StAX, Online im Internet, http://www.galileocomputing.de/openbook/javainsel6/javainsel_13_004.htm v. 2006, Abfrage vom 31.01.07

[War06a] Wartala R.: Weltbaukasten - Mashup: eine Revolution in Zeiten des Web 2.0, in: iX - Magazin f. professionelle Informationstechnik, Juli 2006, S.54 - 59

[War06b] Wartala R.: Bildergeflimmer - Mashups mit der Flickr-API erstellen, in: iX - Magazin f. professionelle Informationstechnik, Juli 2006, S. 62 - 67

[Win99] Winer, D.: XML-RPC Specification, Online im Internet, http://www.xmlrpc.com/spec v. 15.06.99, Abfrage vom 16.01.07

[WSAG04] Booth, D. / Haas, H. / McCabe F. / u. a.: Web Services Architecture - W3C Working Group Note, Online im Internet, http://www.w3.org/TR/ws-arch/#whatis v. 11.02.2004, Abfrage vom 14.01.2007

[WSF06] k. A.: The Wiki for Finding Web Service and Open APIs, Online im Internet, http://wsfinder.jot.com/WikiHome v. 12.08.06, Abfrage vom 29.01.07

[WSL06] k. A.: ECS - Web Service Licence Agreement, Online im Internet, http://www.amazon.com/AWS-License-home-page-Money/b/?node=3440661 v. 03.10.06, Abfrage vom 16.01.07

[Yan06] Yank, K.: Free Video: Recognizing Web 2.0, Online im Internet, http://www.sitepoint.com/blogs/2006/12/06/free-video-recognizing-web-20/ v. 06.12.06, Abfrage vom 27.01.07

B Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich diese Studienarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

.....	.....
Ort, Datum	Unterschrift