pedocs
Open Access Erziehungswissenschaften

DIPF

Safran, Christian; Lorenz, Anja; Ebner, Martin

Webtechnologien. Technische Anforderungen an Informationssysteme

Ebner, Martin [Hrsg.]; Schön, Sandra [Hrsg.]: L3T. Lehrbuch für Lernen und Lehren mit Technologien. 2. Auflage. 2013, [10] S.

Quellenangabe/ Reference:
Safran, Christian; Lorenz, Anja; Ebner, Martin: Webtechnologien. Technische Anforderungen an Informationssysteme - In: Ebner, Martin [Hrsg.]; Schön, Sandra [Hrsg.]: L3T. Lehrbuch für Lernen und Lehren mit Technologien. 2. Auflage. 2013, [10] S. - URN: urn:nbn:de:0111-opus-83347 - DOI: 10.25656/01:8334

https://nbn-resolving.org/urn:nbn:de:0111-opus-83347
https://doi.org/10.25656/01:8334

Nutzungsbedingungen

Dieses Dokument steht unter folgender Creative Commons-Lizenz: http://creativecommons.org/licenses/by-sa/3.0/de/deed - Sie dürfen das Werk bzw. den Inhalt vervielfältigen, verbreiten und öffentlich zugänglich machen sowie Abwandlungen und Bearbeitungen des Werkes bzw. Inhaltes anfertigen, solange sie den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen und die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrags identisch, vergleichbar oder kompatibel sind.
Mit der Verwendung dieses Dokuments erkennen Sie die Nutzungsbedingungen an.

Terms of use

This document is published under following Creative Commons-License: http://creativecommons.org/licenses/by-sa/3.0/de/deed.en - You may copy, distribute and transmit, adapt or exhibit the work or its contents in public and alter, transform, or change this work as long as you attribute the work in the manner specified by the author or licensor. New resulting works or contents must be distributed pursuant to this license or an identical or comparable license.

By using this particular document, you accept the above-stated conditions of use.

Kontakt / Contact:

peDOCS
DIPF | Leibniz-Institut für Bildungsforschung und Bildungsinformation
Informationszentrum (IZ) Bildung
E-Mail: pedocs@dipf.de
Internet: www.pedocs.de

Mitglied der Leibniz-Gemeinschaft

#webtech
#einfuehrung
#informatik

Christian Safran, Anja Lorenz, Martin Ebner

Webtechnologien

Technische Anforderungen an Informationssysteme

Dieses Kapitel gibt eine Einführung in die technischen Grundlagen von Webtechnologien, welche bei Lernsoftware Verwendung finden können. Basierend auf der zugrunde liegenden Infrastruktur des Internets hat vor allem das World Wide Web seit der Jahrtausendwende alle Arten von Informationssystemen, auch solche für das Lernen und Lehren, nachhaltig beeinflusst. Dementsprechend ist ein Grundverständnis der entsprechenden Technologien und technischen Anforderungen von Vorteil, um Chancen und Grenzen von webbasierter Lernsoftware zu erläutern. Auf Basis der Architekturen und Protokolle des World Wide Web haben sich in den letzten Jahren Technologien wie Webanwendungen und Webservices entwickelt, die größere und reichhaltigere Applikationen im Web ermöglichen als zu dessen Anfangszeit. Diese Möglichkeiten führten einerseits zu der Entwicklung von Rich Internet Applications als „Internetanwendungen mit Desktop-Feeling“ und andererseits zum Aufkommen von Mashups, in welchen diverse Inhalte und Funktionalitäten anderer Applikationen in innovativen Szenarien neu kombiniert werden. Beide Ansätze sind in der aktuellen Entwicklung von Informationssystemen für das Lernen und Lehren unabdingbar.

L3T Lehrbuch für Lernen und Lehren mit Technologien
http://l3t.eu
M. Ebner und S. Schön (Hrsg.)

Version 2013

CC BY-SA L3T | http://l3t.eu
http://creativecommons.org/licenses/by-sa/3.0/de/

1. Einleitung

Das World Wide Web (siehe #WWW) bzw. das Internet hat in den letzten Jahren einen großen Stellenwert in unserer heutigen „Informationsgesellschaft“ eingenommen. So verwundert es auch nicht, dass ein großer Teil der aktuellen Umsetzungen von Informationssystemen für das Lernen und Lehren (siehe #infosysteme) auf Webtechnologien basiert. Auch Trends wie Soziale Medien basieren auf Webtechnologien und sind inzwischen fixe Bestandteile verschiedener Lern- und Lehransätze mit Technologien.
Für das Verständnis der technischen Anforderungen an speziell für das Lernen und Lehren entwickelten Informationssystemen ist demnach auch ein zumindest grundlegendes technisches Verständnis der darunter liegenden Webtechnologien notwendig. Bedingt durch die Architektur und Geschichte des World Wide Web bieten diese Ansätze zwar ein großes Potenzial, sind aber auch mit einigen technischen Einschränkungen verbunden. Dieses Kapitel soll Basiswissen über diese Technologien vermitteln und das Interesse an detaillierten weiterführenden Informationen zu den angesprochenen Technologien, welche in einschlägiger Fachliteratur nachgelesen werden können, wecken.

2. Grundlegende Technologien

Das Internet ist eine weltweite Verknüpfung von Datennetzen, welche Ende der 1960er Jahre in den USA von der vom DARPA (Defense Advanced Research Projects Agency) initiiert wurde und aus dem daraus entstandenen ARPANET (Advanced Research Projects Agency Network) hervorging (Cerf et al., 2012).

Der Begriff Internet als solcher bezeichnet das entstehende „interconnected network“ zwischen unabhängigen und weltweit verteilten Computernetzen.

Die Kommunikation in einem Computernetzwerk wird über **Protokolle** geregelt. Ein Protokoll ist eine Regelvorschrift, welche den Datenaustausch und die Kommunikation zwischen Computern detailliert beschreibt. In der Netzwerkkommunikation hat sich eine Aufteilung der Verantwortung auf einzelne aufeinander aufbauende Protokolle als sinnvoll erwiesen. Diese Aufteilung kann man über das OSI-Schichtenmodell (Open System Interconnection) der internationalen Standardisierungsgesellschaft ISO (International Standardization Organisation), in dem Protokolle einer Ebene nur jeweils mit Protokollen der benachbarten Ebenen in Kontakt treten müssen, einheitlich beschreiben.

Die Ebenen des OSI-Schichtenmodells (siehe Tabelle 1) beschreiben die Kommunikation in verschiedenen Abstraktionsstufen. So beschäftigt sich die unterste Schicht (1) mit der physikalischen Übertragung, während sich die oberste Schicht (7) mit Anwendungen auseinandersetzt. Grob werden diese Schichten in anwendungs- und transportorientierte Schichten unterteilt. In den anwendungsorientierten Schichten finden sich Protokolle, welche dieeigentliche Benutzer/innen-Interaktion in (Web-)Anwendungen beschreibt. Die transportorientierten Schichten bilden die eigentlichen „Schienen“ des darunterliegenden Datenverkehrs. Im Rahmen dieses Kapitels werden wir uns nur mit anwendungsorientierten Protokollen beschäftigen. Die Kommunikation vieler Programme auf der Anwendungsschicht erfolgt nach dem Client-Server-Prinzip (siehe #infosysteme).

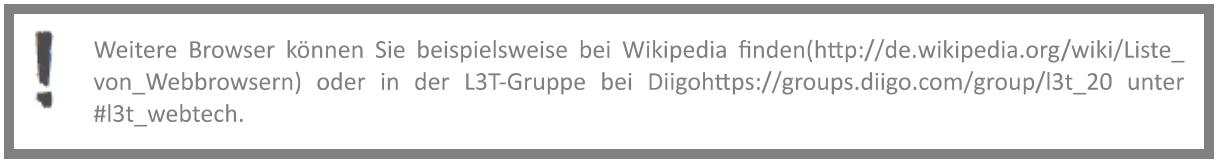
	Schicht	Ebene	Protokolle
7	Anwendungsschicht	Anwendungsorientierte Schichten	HTTP(S)
6	Darstellungsschicht		FTP
5	Kommunikationsschicht		POP, SMTP
4	Transportschicht	Transportorientierte Schichten	TCP UDP
3	Vermittlungsschicht		IP
2	Sicherungsschicht		Ethernet
1	Physikalische Schicht		

Tab. 1: OSI-Schichtenmodell

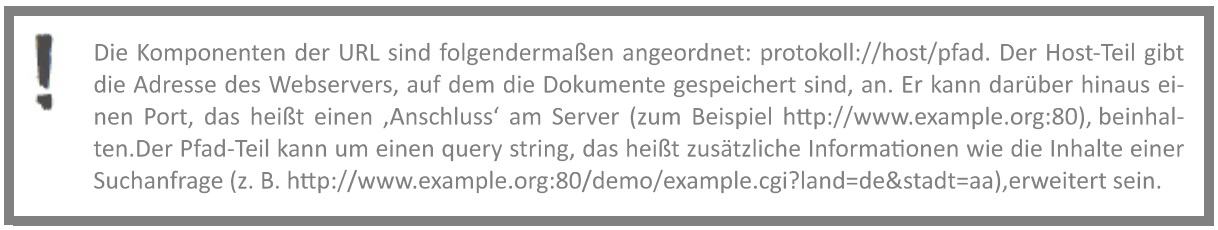
Basierend auf dem Internet als Verbindung unterschiedlicher Computernetze ermöglichen diverse Protokolle den Zugriff auf eine Vielzahl von Anwendungen. Die wohl prominenteste dieser Anwendungen ist das World Wide Web.

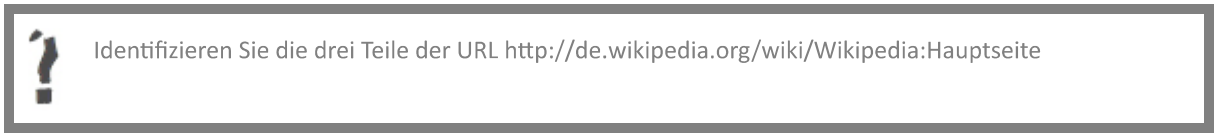
3. Das World Wide Web (WWW)

Seit seiner Entwicklung im Jahr 1989 durch die Forschergruppe um Sir Tim Berners-Lee (Cailliau, 1995), wurde aus dem World Wide Web die wohl bekannteste und meist benutzte Anwendung des Internets. In vielen Fällen wird der Begriff Internet, obwohl er eigentlich nur die darunterliegende Netzwerkinfrastruktur bezeichnet, als Synonym für das World Wide Web verwendet. Beim WWW handelt es sich in seiner Grundstruktur um eine verteilte Sammlung von Dokumenten, welche unter Verwendung von Internet-Protokollen über eine Anwendung abrufbar sind.Diese Dokumente sind untereinander mittels #Hyperlinks verknüpft und bilden dadurch ein weltweites Netz an Informationen. Die Anwendung, mit der auf diese Dokumente, man spricht häufig von Webseiten, zugegriffen werden kann, wird als Browser bezeichnet. Bekannte Browser sind der Internet Explorer von Microsoft, Chrome von Google, Firefox von Mozilla, Safari von Apple oder Opera von Opera Software.



Jedes Dokument im Web wird durch eine URL (Uniform Resource Locator) identifiziert. Diese URL besteht im Web zumeist aus drei Komponenten: Protokoll, Host und Pfad.

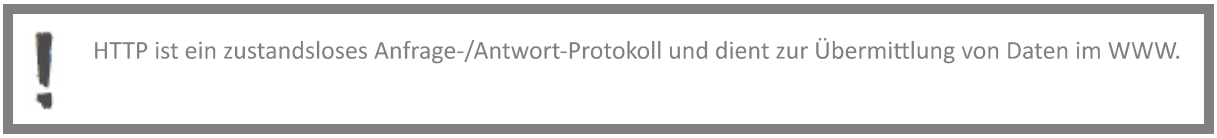




Hypertext Transfer Protocol (HTTP)

Der Browser verständigt sich mit dem Webserver, auf dem die Dokumente gespeichert sind, über das **Hypertext Transfer Protocol**. Wie jedes Protokoll beschreibt HTTP den Aufbau der Nachrichten vom Client an den Server. Die Kommunikation erfolgt immer über Anfragen des Clients an den Server und zugehörige Antworten. Alle Nachrichten werden als Klartext, also unverschlüsselt, übermittelt.

Durch diesen Aufbau der Kommunikation ist eine zentrale Problematik der Entwicklung von Webanwendungen bedingt: Der Server kann nicht von sich aus mit dem Client kommunizieren, da er immer auf eine Anfrage angewiesen ist. Wartet eine Anwendung nun auf ein bestimmtes Ereignis (zum Beispiel das Ergebnis einer aufwendigen Berechnung), kann der Server den Client nicht über dessen Eintritt verständigen, sondern der Client muss in regelmäßigen Abständen Anfragen über den Status an den Server schicken. Dies bringt natürlich einerseits zusätzlichen Netzwerkverkehr und anderseits zusätzliche Arbeitszeit des Servers mit sich. Bei HTTP handelt es sich um ein zustandsloses Protokoll. Alle Anfragen sind somit voneinander unabhängig zu betrachten. Für die meisten Anwendungen im Web ist es aber notwendig, mehrere Anfragen in Zusammenhang zueinander zu sehen. So besteht zum Beispiel der Bestellvorgang in einem Webshop durchaus aus mehreren sequentiellen Anfragen (Auswahl der Waren, Eingabe der Adresse, Bestätigung des Kaufes). Solche Zusammenhänge können nicht durch das Protokoll selbst verwaltet werden, sondern müssen durch die Anwendungen gehandhabt werden, welche ihre Daten über HTTP übertragen. Hierzu können Sitzungen (Sessions) verwendet werden, in denen eine eindeutige ID mit jeder Anfrage versendet wird. Eine andere Möglichkeit ist die Verwendung von Cookies, mit denen der Browser persistente Informationen (zum Beispiel Kundinnen- und Kundendaten) zu einem Webserver lokal speichern kann. Üblicherweise beginnen solche Sessions mit der Anmeldung mittels Login und Passwort und enden durch Abmeldung oder wenn der Browser geschlossen wird.



HTTPS

Die Übertragung der Daten in Klartextform bei HTTP ist nicht in jedem Fall wünschenswert. So könnten sensible Daten wie zum Beispiel Kennwörter durch beliebige ‚Zuhörer /innen' abgefangen werden. Aus diesem Grund wurde das **Hypertext Transfer Protocol Secure (HTTPS)** als Verfahren entwickelt, um Daten im Web abhörsicher zu übermitteln.Das Protokoll ist an sich identisch zu HTTP, allerdings werden die Daten mittels des Protokoll Secure Socket Layer bzw. Transport Layer Security (SSL/TLS) verschlüsselt. Zu Beginn der verschlüsselten Verbindung muss sich bei HTTPS der Server identifizieren. Dies geschieht über ein Zertifikat, welches die Identität bestätigen soll. Bei der ersten Anfrage an einen Webserver kann es notwendig sein, dass die Authentizität dieses Zertifikates bestätigt werden muss.

HTML

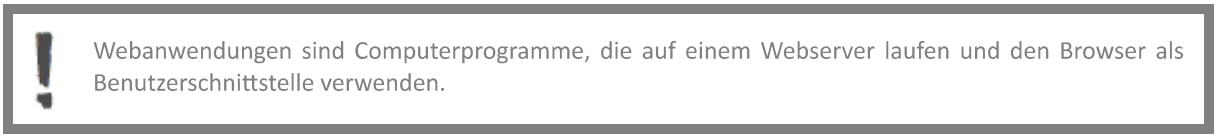
Die eigentlichen Dokumente, die über HTTP vom Server an den Client übermittelt werden, sind meist HTML-Dokumente. Die **Hypertext Markup Language (HTML)** ist eine Auszeichnungssprache, welche Dokumentinhalte beschreibt. Der Browser zeigt anhand von HTML und der mitgelieferten Formatierungsinformation diese Dokumente an. Üblicherweise wird also das Aussehen der HTML-Dokumente in einer separaten Datei beschrieben. Diese **Cascading Style Sheets (CSS)** werden ebenfalls vom Browser über HTTP angefordert. Mit HTML5 lassen sich neben Texten, Tabellen und Bildern auch multimediale Inhalte (Videos, Animationen) beschreiben, die dann unmittelbar vom Browser wiedergegeben werden können. Für die Darstellung solcher Inhalte waren bisher zusätzliche Softwareprodukte wie etwa Adobe Flash notwendig.

Webanwendungen

Im Laufe der Zeit haben sich die Anforderungen an Informationssysteme im Web von der bloßen Zurverfügungstellung von Dokumenten in Richtung ausgefeilter Programmlogik weiterentwickelt. Wie im Kapitel #infosysteme aufgezeigt, erfordern natürlich auch Informationssysteme für das Lernen und Lehren solch eine Programmlogik. Diese Programme, die auf einem Webserver ausgeführt werden, werden **Webanwendungen** genannt.

Wie bei statischen Webseiten, das heißt reinen HTML-Dokumenten, wird ein Browser zur Interaktion mit dem Webserver verwendet und die Daten werden mittels HTTP(S) übermittelt. Im Unterschied zu einer einfachen Webseite übermittelt der Webserver beim Aufruf der URL allerdings nicht ein bereits vorliegendes Dokument, sondern ruft ein Programm auf, welches aus einer Vorlage für Inhalt und Formatierung sowie aus variablen Daten dynamisch ein Dokument erstellt und an den Client übermittelt.

Der Browser übernimmt in diesem Fall also die Rolle der Benutzer/innen-Schnittstelle für ein auf dem Server ausgeführtes Programm. Mit Webanwendungen kann komplexe Software realisiert werden, welche clientseitig nur einen Webbrowser bzw. entsprechende Plug-Ins benötigt.Für die Entwicklerin oder den Entwickler der Software wird die Softwarewartung erheblich vereinfacht, da Anpassungen nur am Server erfolgen müssen. Für die Benutzerinnen und Benutzer ergibt sich der Vorteil, dass die zusätzliche Installation einer Software weitestgehend entfällt.



Andererseits hat dieser Ansatz auch einige Nachteile. So ist eine ständige Internetverbindung mit ausreichender Bandbreite nötig. Zudem kann der Server, bedingt durch die Tatsache, dass HTTP auf dem Anfrage-/Antwort-Prinzip basiert, nicht selbstständig Informationen an die Clients senden, sondern ist auf periodische Anfragen angewiesen.Zu guter Letzt bedeutet unter anderem das zustandslose Design des Protokolls, dass die Sicherheit der Webanwendung ein nicht zu vernachlässigender zentraler Bestandteil der Entwicklung der Software sein muss, um vor einer Vielzahl von möglichen Angriffsszenarien, wie das Ausspähen von Passwörtern, geschützt zu sein.

Webservices

Webservices sind eine spezielle Art von Webanwendungen, die der Bereitstellung von Daten für andere Applikationen dienen (World Wide Web Consortium, 2004). Sie sind üblicherweise Application Programming Interfaces (API) und stellen als solche eine einheitlich definierte Schnittstelle für fremde Anwendungen zur Verfügung, um auf die Funktionalität des Service zuzugreifen, indem der Service Daten bereitstellt.

In Zusammenhang mit Informationssystemen für das Lernen und Lehren bietet die Integration solcher Webservices innovative Ansätze für die Einbindung externer Ressourcen und Funktionalitäten in ein derartiges Informationssystem (Vossen & Westerkamp, 2003). Anders als bei Webanwendungen werden bei Webservices üblicherweise keine HTMLDokumente vom Server geladen, da für die aufrufenden Applikationen Formatierungen sowie die Lesbarkeit für menschliche Benutzer/innen irrelevant sind und der Fokus rein auf dem Inhalt liegt.

Der Grundgedanke von Webservices ist die Möglichkeit der automatischen Verarbeitung von Daten im Web durch Softwareagenten, welche lose gekoppelt Aufgaben für Benutzer/innen ausführen. Webservices stellen ihre detailliert beschriebene Funktionalität hierbei anderen Anwendungen zur Verfügung, seien dies autonome Agenten, Webanwendungen oder andere Webservices. Dadurch können Entwickler/innen bereits bestehende Funktionalitäten in ihren eigenen Anwendungen verwenden (Mashup, siehe Abschnitt 6).

Die technologische Umsetzung von Webservices erfolgt meist über eine der folgenden Möglichkeiten:

- **SOAP/WSDL**: Ein flexibles System, in dem Nachrichten über das **Simple Object Access Protocol (SOAP)** ausgetauscht werden. Wie diese Nachrichten für die einzelnen Webservices aussehen, wird über die **Web Services Description Language (WSDL)** beschrieben. Anfragen und Antworten sind in XML, einer allgemeinen Auszeichnungssprache für hierarchische Daten, geschrieben. In den Nachrichten werden die gewünschten Funktionen des Webservices aufgerufen, die Antworten werden vom Server retourniert.
- **Representational State Transfer (REST)** bezeichnet eine Technologie, in der jede einzelne Funktion des Webservices über eine individuelle URL aufgerufen wird. Die Kommunikation erfolgt zustandslos, ist also nicht von Sitzungen, Benutzer/innen-Daten oder ähnlichem abhängig

Beschreiben Sie die Unterschiede zwischen Webanwendungen und Webservices Vergleichen Sie Ihre Antwort mit Tabelle 2.

Webanwendung	Webservice
Zielgruppe: (menschliche) Benutzer/innen	Zielgruppe: andere Applikationen
Ausgabe als HTML	Ausgabe als XML o.ä. für Maschinen optimierte Formate
Clientseitiges Programm: Browser	Verarbeitung in anderen Applikationen

Tab. 2: Unterschiede von Webanwendung und Webservice

Webservices verwenden als Ausgabeformat häufig XML-Dokumente. **Die EXtensible Markup Language (XML)** beschreibt eine sehr allgemeine, flexible und für individuelle Bedürfnisse erweiterbare Auszeichnungssprache. Wie HTML dient sie zur Darstellung strukturierter Inhalte, ist aber in erster Linie nicht für die menschenlesbare Darstellung, wie beispielsweise in Webservices, gedacht. Sie ist von konkreten Plattformen und Implementierungen unabhängig und durch ihre Charakteristik als Metasprache vielseitig verwendbar.Letzteres bedeutet, dass sich auf Basis von XML durch die anwendungsspezifische Definition und Verwendung eines Schemas Datenaustauschformate, man spricht von Dialekten, für spezialisierte Anwendungen definieren lassen.

4. Einführung in die Applikationsentwicklung für das Web

Die Entwicklung von Webapplikationen und -services kann generell in zwei Gruppen von Ansätzen unterteilt werden. Bei **serverseitigen** Ansätzen erfolgt die Verarbeitung der Programmlogik am Webserver, der Client, und damit der Benutzer bzw. die Benutzerin, erhält lediglich das Ergebnis. Bei **clientseitigen** Ansätzen erfolgt zumindest ein Teil des Programmablaufes am Rechner der Benutzer/innen. In realen Anwendungen wird meist eine Kombination von Ansätzen beider Gruppen verwendet.

Serverseitige Ansätze

Der Vorteil von serverseitigen Ansätzen liegt darin, dass die Benutzer/innen außer einem Browser keine weiteren Programme benötigen. Der eigentliche Programmcode wird serverseitig verarbeitet und das Ergebnis, meist ein (X)HTML-Dokument, an den Client gesandt. Der Nachteil liegt in der Verminderung der Reaktionsgeschwindigkeit. Einerseits erfordert jede Aktion der Nutzerin bzw. des Nutzers einen erneuten HTTP-Request (Anfrage) und damit einen neuen Aufruf der Seite. Andererseits benötigt das Ausführen der Programmlogik Rechenzeit am Server und vermindert somit zusätzlich dessen Reaktionszeit.

PHP ist der heute wohl am meisten verbreitete serverseitige Ansatz für Webapplikationen. Die Abkürzung, ursprünglich ‚Personal Home Page tools', ist ein rekursives Akronym für PHP, ‚Hypertext Preprocessor' (The PHP Group, 2010). Es handelt sich bei PHP um eine Skriptsprache, das heißt, der eigentliche Programmcode wird nicht zu einem Bytecode kompiliert (der direkt vom Rechner ausgeführt werden kann), sondern bei jedem Aufruf von einem Interpreter neu verarbeitet. Dieser Performance-Nachteil kann aber durch optionale Erweiterungen der Software kompensiert werden. PHP-Programmcode wird innerhalb der Dokumente durch ein vorangestelltes am Ende des Codes gekennzeichnet. Der Interpreter ignoriert Inhalte außerhalb dieser Begrenzungen und übermittelt diese unverändert an den Client. So kann Programmlogik beispielsweise direkt in HTML-Auszeichnungen eingebettet werden.

Seine Verbreitung hat PHP mehreren Aspekten zu verdanken. Einerseits steht es im Rahmen vieler günstiger Webhosting-Angebote vorinstalliert zur Verfügung, da es einfach zu installieren, zu warten und in bestehende Webserver zu integrieren ist. Andererseits ist PHP für Entwickler/innen schnell zu erlernen und sehr flexibel. Nachteile liegen in der Tatsache, dass PHP potentiell erlaubt, schlechter skalierbare und unsichere Webanwendungen zu entwickeln, auch wenn die konkrete Umsetzung einen deutlichen Einfluss auf die Skalierbarbeit und Sicherheit haben kann. Speziell unerfahrenen Entwicklerinnen und Entwicklern fällt es mit anderen Technologien leichter, auf die Aspekte Skalierbarkeit und Sicherheit Rücksicht zu nehmen, da sie dort zur Einhaltung entsprechender Regeln gezwungen werden.

Java Servlets basieren auf der objektorientieren Programmiersprache Java. Sie sind Java-Klassen (logisch gekapselte Teile von Programmcode), welche auf einem Server für die Abarbeitung von Anfragen der Clients zuständig sind. Als Antwort auf diese Anfragen liefern Servlets dynamisch generierte HTML-Dokumente. Anders als bei PHP werden diese Programme nicht zur Laufzeit interpretiert, sondern liegen bereits kompiliert vor, was die Abarbeitung beschleunigt.

Obwohl die Entwicklung mit Java die Einarbeitung in eine komplexere Technologie bedeutet, bietet diese Technologie einige Vorteile. So kann sie durch Anwendung der richtigen Architektur große Verbesserungen in Skalierbarkeit und Erweiterbarkeit bedeuten. Um abgearbeitet zu werden, benötigen Java Servlets einen Servlet Container, wie Apache Tomcat, der den einfachen Webserver ersetzt. Diese technologische Einschränkung ist auch der Grund dafür, dass Java Servlets eher bei großen (Business-)Anwendungen als bei kleinen und mittleren Webanwendungen verwendet werden. Nur wenige Webhoster/innen bieten Pakete mit einem Servlet Container an, da die Installation und die Administration aufwendiger sind.

Java Server Pages (JSP) sind eine weitere auf Java basierende Technologie. Bei klassischen Servlets ist die Ausgabe der resultierenden Webseiten recht aufwendig. Syntaktisch wird bei JSP, ähnlich wie bei PHP, der Java Programmcode mit <%@ und %> abgegrenzt. Der Rest des Dokumentes wird nicht interpretiert und enthält die HTML-Beschreibung der Webseite. Somit sind JSP-Seiten ähnlich den Servlets, erlauben jedoch die Kombination von Programm- und HTML-Code in einem Dokument und erleichtern so, beide Technologien miteinander zu kombinieren.

Active Server Pages (ASP) war ursprünglich ein PHP ähnlicher Ansatz der Firma Microsoft. In der aktuellen Version ASP.NET basiert die Technologie auf dem Microsofts .NET- Framework. Die eigentlichen Anwendungen können hierbei in verschiedenen .NET-Programmiersprachen erstellt werden. Gebräuchlich sind hierbei C# (C Sharp) und VB.NET (Visual Basic.NET). Anders als bei PHP werden bei ASP.NET Programmcode und HTML voneinander getrennt. Da der Programmcode dadurch kompiliert vorliegen kann, wird die Abarbeitung beschleunigt. Der Nachteil von ASP.NET liegt in der Tatsache, dass die Technologie sowohl proprietär als auch kostenpflichtig ist und darüber hinaus einen Microsoft Webserver erfordert.

Clientseitige Ansätze

Im Gegensatz zu serverseitigen Ansätzen wird hier die Programmlogik direkt auf dem Client abgearbeitet. Dies bietet den Vorteil, dass sowohl weniger Datenverkehr notwendig ist als auch die Ressourcen des Webservers geschont werden. Von Nachteil ist allerdings, dass clientseitig eine komplexere Software erforderlich ist, der Client entsprechend leistungsfähig sein muss, um die Programme abarbeiten zu können, und potentiell sicherheitskritische Berechnungen nur auf der Server-Seite durchgeführt werden sollten.

Die wohl wichtigste Basistechnologie in diesem Zusammenhang ist **JavaScript**. Der Interpreter für diese Sprache ist bereits in den Browser integriert, wodurch keine zusätzliche Softwareinstallation notwendig ist. Allerdings sind diese Interpreter je nach Browser unterschiedlich, was für die Entwicklung der Software zusätzlichen Aufwand bedeutet, um JavaScript-Programme auf allen (beziehungsweise möglichst vielen) Browsern lauffähig (‚browsersave') zu machen. JavaScript-Programme können entweder direkt in HTML-Seiten integriert sein oder in eigene Dateien ausgelagert werden. Die implementierte Funktionalität kann hierbei von einfachen Aufgaben, wie der Validierung von Formulareingaben, bis hin zu dynamischer Manipulation der vom Webserver übertragenen Webseite reichen.

Asynchronous JavaScript and XML (AJAX) bezeichnet ein Konzept von Webanwendungen, bei denen JavaScript eingesetzt wird, um Informationen von einem Webserver anzufordern. Bei klassischen Webseiten muss wiederum für jede Aktion vom Client eine Anfrage erstellt und die Antwort des Servers vom Browser interpretiert werden. Dies erfordert ein komplettes Neuladen der Seite, auch wenn sich nur kleine Teile ändern. Mit AJAX werden vom Webserver nur mehr Teilinhalte angefordert. Diese werden als XML-Dokumente übermittelt und anhand der XMLStruktur interpretiert. AJAX manipuliert nun die bereits geladene Seite und ändert nur jene Daten, die wirklich notwendig sind.Dadurch erhält man wesentlich performantere (‚schnellere') Applikationen, mit denen ein Verhalten ähnlich zu Desktop-Anwendungen (in ‚KlickGeschwindigkeit') erreicht wird. Dies führte im nächsten Schritt zu RIA.

Als **‚Rich Internet Application' (RIA)** bezeichnet man jene Webanwendungen, welche durch clientseitige Anwendung von JavaScript Möglichkeiten wie vergleichbare DesktopAnwendungen bieten. So ist es zum Beispiel mit diesem Ansatz möglich, Office-Anwendungen als Webanwendungen zu implementieren. Anders als bei Desktop-Anwendungen müssen diese allerdings nicht installiert werden und bieten die Möglichkeit, auf jedem Rechner, welcher über einen kompatiblen Browser verfügt, ausgeführt zu werden (sind also unabhängig vom verwendeten Betriebssystem). Die Daten werden hierbei zentral auf dem Server hinterlegt. Die Interaktion mit dem Webserver ist auf ein Minimum beschränkt, was flüssiges Arbeiten mit RIA ermöglicht.

Stellen Sie die Vor- und Nachteile von server- und clientseitigen Ansätzen gegenüber. Vergleichen Sie Ihre Antwort mit Tabelle 3.

	Serverseitiger Ansatz	Clientseitiger Ansatz
Vorteile	• unabhängig von clientseitiger Softwareausstattung • gleiches Verhalten auf allen Clients	• Reaktionszeiten ähnlich zu Desktop-Anwendungen • nur benötigte Inhalte werden geladen und in die aktuelle Seite integriert
Nachteile	• jede Aktion erfordert einen Aufruf serverseitiger Funktionalität • jede Aktion erfordert komplettes Neuladen der aktuellen Seite	• Verhalten von Browser (JavaScript Engine) abhängig • Sicherheit der Anwendungen ist aufwendiger zu gewährleisten

Tab.3: Vor- und Nachteile des serverseitigen und des clientseitigen Ansatzes

5. Vor Gebrauch gut schütteln — Syndikation und Integration

Moderne Webapplikationen verdanken ihre Verbreitung großenteils der Tatsache, dass ihre Verknüpfung zentraler Bestandteil ihrer Funktionalität ist. Diese Verknüpfung (Syndikation) geht über einfache Hyperlinks weit hinaus: Inhalte und Funktionalitäten werden zur Verfügung gestellt und in andere Web-Applikationen integriert. So wird eine Kreativität bei der Erstellung von neuen Applikationen ermöglicht, wie sie ohne diese Offenheit und den daraus resultierenden Vorteil, bestehende Funktionalität nicht erneut selbst implementieren zu müssen, nur schwer vorstellbar wäre. Die Syndikation mehrerer fremder Funktionalitäten oder Integration fremder Inhalte in eine Webapplikation werden als **Mashup** bezeichnet. Dieser Begriff wurde ursprünglich in der Musikbranche verwendet, um Remixes zu beschreiben. Im Zusammenhang von Webapplikationen bedeutet er eine in der Anwendung transparente Integration fremder Dienste und Inhalte.

Damit sind jedoch auch Nachteile verbunden:

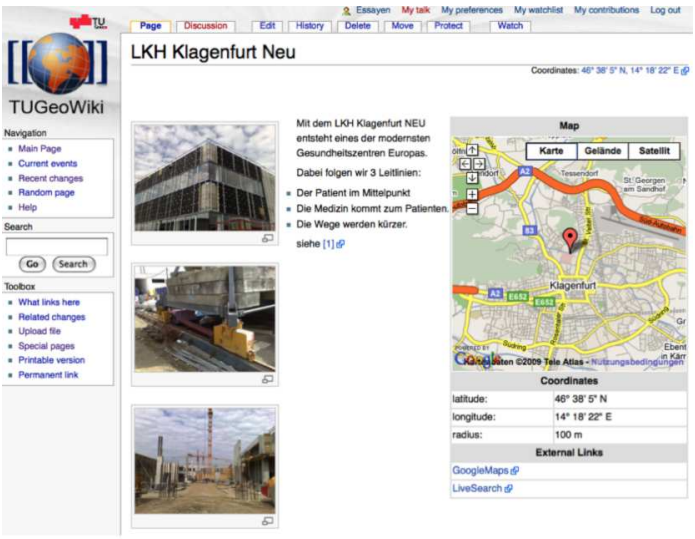
- Wer haftet bei sicherheitskritischen Anwendungen, wenn Funktionen nicht korrekt funktionieren, zum Beispiel wenn bei Lernsoftware eine Applikation die Prüfungsfragen falsch auswertet?
- Alle Server, welche die Services anbieten, müssen ständig verfügbar sein – dies liegt jedoch nicht in der Verantwortung der Betreiber/innen.
- Das Urheberrecht bzw. die Lizenzierung führt zu der Frage, ob man die Funktionen überhaupt integrieren darf.

> **!** Ein Mashup ist die Integration von Inhalten und Funktionalitäten anderer Webapplikationen in die eigene.

Programmierschnittstellen

Application Programming Interfaces (API) sind wohldefinierte Schnittstellen für die Interaktion mit Applikationen. Im Zusammenhang mit Webapplikationen werden sie üblicherweise als Webservices (siehe oben) implementiert und liefern entweder XML-Daten oder vergleichbar strukturierte Daten, zum Beispiel nach der einfacheren JavaScript Object Notation (JSON). Sowohl SOAP als auch REST-Ansätze sind möglich, auch wenn der Trend der letzten Jahre in Richtung REST-Anwendungen weist.Einerseits können APIs dazu verwendet werden, um Funktionalität zur Verfügung zu stellen, wie zum Beispiel die Darstellung von Koordinaten in Karten mittels der Google-Maps-API (siehe Beispiel in Abbildung 3). Andererseits können solche Applikationen auch Inhalte zur Verfügung stellen. So ist es mit der Flickr-API zum Beispiel möglich, auf Fotos des Internetdienstes zuzugreifen. Andere APIs ermöglichen beispielsweise die Integration von Daten aus sozialen Netzwerken, wie die Facebook-API.

Abb. 1: TUGeowiki: Integration von Kartenmaterial über die GoogleMaps API. Quelle: Safran und Zaka, 2008



Quelle: Safran und Zaka, 2008.TUGeowiki: Integration von Kartenmaterial über die GoogleMaps API http://dx.doi.org/10.1109/CSSE.2008.188.

RSS

Real Simple Syndication (RSS) beschreibt eine Technologie, mit deren Hilfe Inhalte von Websites, sogenannte Feeds, zur Verfügung gestellt werden. Die Daten liegen hierbei in einem XML-Format vor. RSS dient hauptsächlich zur Benachrichtigung bei häufig aktualisierten Informationen, wie Weblogs oder Nachrichtenseiten. Benutzer/innen können RSS-Feeds mit einem Client (z.B. auch einem Mailprogramm) abonnieren, der ihnen immer die jeweils neuesten Meldungen anzeigt. In Webapplikationen bietet RSS eine gute Möglichkeit zur Syndikation von Daten aus unterschiedlichen Quellen. Da die Daten in XML vorliegen, können sie einfach maschinell weiterverarbeitet und in die eigene Webseite integriert werden.

Widgets

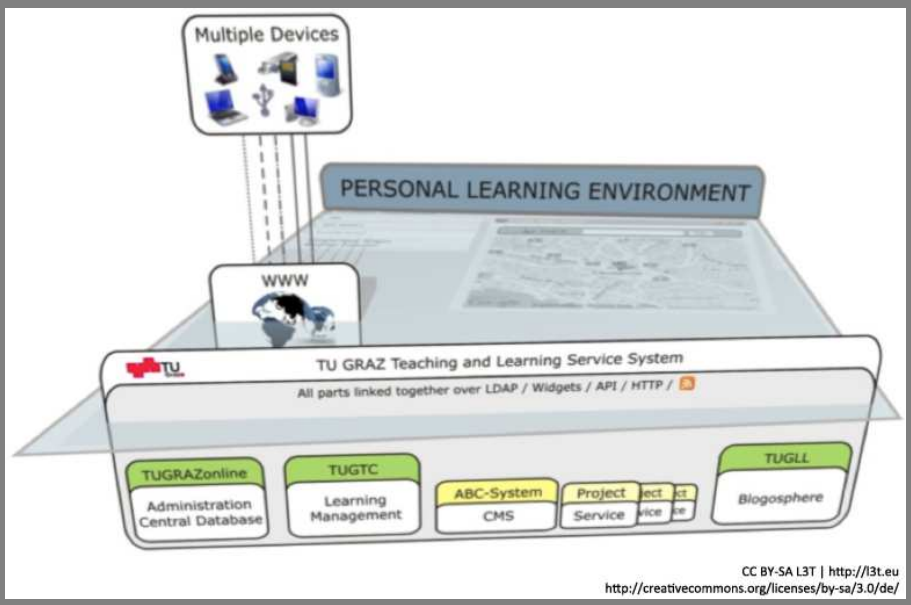
Eine dritte Technologie, die bei der Integration verschiedener Webanwendungen Anwendung findet, sind **Widgets**. Widgets sind kleine, in sich abgeschlossene Programme, die im Rahmen einer anderen grafischen Benutzeroberfläche ablaufen. Die Funktionalität solch eines Widgets ist üblicherweise spezialisiert und eingeschränkt. Im Desktop-Bereich haben sich Widgets inzwischen bei den meisten Betriebssystemen durchgesetzt und können entweder direkt als Teil des Desktops (wie Minianwendungen in Microsoft Windows) oder über eine getrennte WidgetEngine (wie das Dashboard bei Mac OS X) benutzt werden.

Im Zusammenhang mit Webapplikationen bezeichnet der Begriff Widget genauso eine eigenständige, abgeschlossene und in eine andere Applikation integrierte Funktionalität. Diese sind meist innerhalb der Benutzer/innen-Schnittstelle der eigentlichen Webapplikation mehr oder weniger frei positionierbar. So können Widgets zum Beispiel Daten über einen RSS-Feed abrufen oder auf Funktionalitäten mittels einer API zugreifen. Ein Beispiel für Widgets sind die Apps des sozialen Netzwerks Facebook, welche nicht selbst von Facebook, sondern von anderen Entwicklerinnen und Entwicklern erstellt, aber in die Webapplikation Facebook integriert werden. Für Benutzer/innen ist kaum ein Unterschied erkennbar. Über eine API greifen diese Apps zusätzlich auf die Funktionalität von Facebook zu.

Die Personal Learning Environment an der TU Graz

ls Praxisbeispiel für ein Mashup kann die Personal Learning Environment (PLE) an der Technischen Universität Graz genannt werden (Taraghi & Ebner, 2012). In dieser persönlichen Lernumgebung sind verschiedene, zum Teil unabhängige und verteilte Dienste der TU Graz sowie andere Webanwendungen aus dem Internet integriert. Abbildung 2 stellt dieses Grundkonzept graphisch dar. Universitätsdienste wie das Administrationssystem (TUGRAZ.online), LMS (TUGTC), Blogosphere (TUGLL) und viele Lernobjekte für unterschiedliche Lehrveranstaltungen sind in der PLE kombiniert (Ebner et al., 2011).Zusätzlich dazu können zahlreiche fremde Lernobjekte und webbasierte Dienste wie Google-Applikationen, Flickr, YouTube, Twitter, Facebook etc. integriert werden. Die Integration dieser Dienste erfolgt mittels der angebotenen API und darauf aufbauenden Widgets. Das PLE als eine Rich Internet Application (RIA) bietet ein Mashup von Widgets, die an die persönlichen Nutzungsbedürfnisse anpassbar sind. Neben einer Portierung dieser Widgets für mobile Endgeräte ist auch eine inter-Widget-Kommunikation möglich. Ein Widget kann mit einem anderen gekoppelt sein, in dem es zum Beispiel auf Google Maps eine Stadt sucht und das Wetter-Widget automatisch für diese Stadt die Wetterprognose anzeigt. Die Benutzer/innen sind in der Lage, sich die geeigneten, zu ihrem Studium benötigten Widgets auszusuchen und diese nach ihren aktuellen Bedürfnissen zu konfigurieren.

Abb. 2: PLE: Ein Mashup von Widgets ermöglicht die Integration von verschiedenen Universitätsdiensten sowie der fremden Ressourcen aus dem Internet.



6. Aktuelle Trends in der Entwicklung von Webanwendungen

In den letzten Jahren haben sich drei Ansätze bei der Entwicklung von Webanwendungen durchgesetzt. Vom serverseitigen Standpunkt aus bieten viele, vor allem bekannte und große Webanwendungen, wie die verschiedenen Google-Produkte oder Facebook, parallel zu ihren Webanwendungen Webservices als API an. Dies ermöglicht die Verwendung ihrer Funktionalität in anderen Webanwendungen und ermutigt dazu.

Vom clientseitigen Standpunkt ist RIA der Stand der Technik. Hier wird bei vielen Webanwendungen darauf Wert gelegt, dass den Benutzerinnen und Benutzern die Anmutung einer Desktop-Anwendung vermittelt wird. Durch die Anwendung von AJAX haben sich die Reaktionszeiten der Webanwendungen erheblich verbessert.

Zu guter Letzt gehört Syndikation und Integration zum Stand der Technik in vielen Anwendungsbereichen. Durch die Möglichkeiten, die Webservices und RIA bieten, kann auf Funktionalität und Inhalte anderer Anwendungen leicht zurückgegriffen werden. Diese können nahtlos in die eigene Benutzungsschnittstelle integriert werden.

Literatur

- Cailliau, R. (1995). A Little History of the World Wide Web. URL: http://www.w3.org/History.html [2013-07-31].
- Cerf, V. (1993). How the Internet Came to Be. In: B. Aboba (Ed.), The Online User's Encyclopedia Addison-Wesley, November 1993, URL: http://ads.ahds.ac.uk/project/strategies/toc.html (2013-10-18).
- Ebner, M.; Schön, S.; Taraghi, B.; Drachsler, H. & Tsang, P. (2011). First steps towards an integration of a Personal Learning Environment at university level. In: R. Kwan et al. (Eds.): ICT 2011, CCIS 177, Berlin/Heidelberg: Springer-Verlag, 22–36. URL/DOI: http://dx.doi.org/10.1007/978-3-642-22383-9_3 [2013-08-09].
- Safran, C. & Zaka, B. (2008). A Geospatial Wiki for m-Learning. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering, IEEE Computer Society, Washington, 5, 109-112. URL/DOI: http://dx.doi.org/10.1109/CSSE.2008.188 [2013-08- 09].
- Taraghi, B.; Ebner, M. (2012). Personal Learning Environment. In K. Wilbers & A. Hohenstein (Hrsg.), Handbuch E-Learning. Expertenwissen aus Wissenschaft und Praxis – Strategien, Instrumente, Fallstudien. Köln: Deutscher Wirtschaftsdienst (Wolters Kluwer Deutschland), 43. Erg.-Lfg. August 2012, 1-4. URL: http://elearningblog.tugraz.at /archives/5510 [2013-08-09].
- The PHP Group (2010). Hypertext Preprocessor. URL: http://www.php.net [2013-07-31].
- Vossen, G. & Westerkamp, P. (2003). E-Learning as a Web Service. In: Proceedings of the Seventh International Database Engineering and Applications Symposium, Los Alamitos, CA, USA: IEEE Computer Society, 242-249. URL/DOI: http://dx.doi.org/10.1109/IDEAS.2003.1214933 [2013-08-09].
- World Wide Web Consortium (2004). Web Services Architecture. URL: http://www.w3.org/TR/ws-arch/#id2268743 [2013-07-31].