Webbasierte Informationssysteme – Einführung

Integriertes Seminar
AG DBIS

Golo Haas
Matrikelnummer 342773
webmaster@golohaas.de

25. Juni 2004

Inhaltsverzeichnis

1 Einführung

1.1 Informationssysteme

In den letzten Jahren hat sich das Internet von einem rein akademisch genutzten Forschungsnetz zu einem alltagstauglichen Massenmedium gewandelt, wobei diese Entwicklung vor allem durch die Erfindung des World Wide Web (WWW) vorangetrieben wurde.

Waren in der Anfangszeit des WWW nur statische und wenig komplexe Webseiten vorhanden, wuchs im Laufe der Zeit ein Bedarf nach umfangreicher, dynamischer Informationsaufbereitung. Mit der Möglichkeit, auf Webseiten Formulare zur Dateneingabe zu definieren, und diese Daten dann serverseitig weiterzuverarbeiten, war das Fundament für diesen Bedarf gelegt.

Heute ist dynamische Informationsverarbeitung keine Ausnahme mehr. Nicht nur Webseiten großer Firmen, sondern auch die Webseiten von kleinen und mittelständischen Unternehmen oder sogar private Webseiten bauen auf entsprechenden Techniken auf, um ihr Angebot übersichtlich und personalisiert darzustellen.

In diesem Zusammenhang fällt oft der Begriff des „webbasierten Informationssystems“, das ebenso wie die klassischen Informationssysteme ein Werkzeug zur Dateneingabe und -verarbeitung darstellt, im Gegensatz zu diesen aber auf diversen Technologien des Internet beruht.

1.2 Protokolle

Unter diesen Technologien finden sich zunächst zahlreiche Protokolle, vom etablierten Hypertext Transfer Protocol (HTTP) zur Übertragung von Hypertextdokumenten bis hin zu modernen, auf XML basierenden Protokollen wie dem Simple Object Access Protocol (SOAP), der Web Services Description Language (WSDL) und der Universal Discovery, Description and Integration (UDDI), die vor allem im Bereich der Web Services ihre Bedeutung haben.

1.3 Datenformate und Präsentation

Während Hypertextdokumente üblicherweise in der von Tim Berners Lee entworfenen und vom World Wide Web Consortium (W3C) [1] weiter entwickelten Seitenbeschreibungssprache Hypertext Markup Language (HTML) verfasst sind, gibt es zahlreiche weitere Datenformate wie beispielsweise die Extensible Markup Language (XML) und alle von ihr abgeleiteten Sprachen wie beispielsweise die Extensible Hyptertext Markup Language (XHTML), die Mathematical Markup Language (MathML), die Synchronized Multimedia Integration Language (SMIL) oder Scalable Vector Graphics (SVG). Diese Sprachen definieren jedoch nur die semantische Struktur eines Dokumentes, nicht dessen optische Gestaltung. Hierzu existieren weitere Sprachen, wie beispielsweise Cascading Style Sheets (CSS) oder die Extensible Stylesheet Language (XSL).

2 Einsatzmöglichkeiten

2.1 „Information anywhere, anytime on any device“

Obwohl die derzeitige Devise „Information anywhere, anytime on any device“ [2] der Firma Microsoft nicht speziell im Hinblick auf webbasierte Informationssysteme entwickelt wurde, beschreibt sie deren Einsatzmöglichkeiten dennoch treffend.

Denn schließlich wird in der Regel versucht, umfangreiche und komplexe Informationen verfügbar zu machen. Dass die Verfügbarkeit dieser Informationen bei webbasierten Informationssystemen anders als in klassischen Systemen im Idealfall absolut orts- und zeitunabhängig sein soll, liegt im Hinblick auf die bereits in der Einführung angesprochene Entwicklung des Internet hin zu einem jederzeit nutzbaren und allgegenwärtigen Medium auf der Hand.

Damit sind die beiden ersten Aspekte, „anywhere“ und „anytime“, erläutert, weshalb der Fokus noch auf „on any device“ gelegt werden muss. War das Internet ursprünglich eine Domäne des Computers in seinen klassischen Ausprägungen, finden sich heute bereits Zugangs- und Zugriffsmöglichkeiten über mobile Geräte von Handys über Personal Digital Assistants (PDA) bis hin zu Wearable Devices, die zum Teil sogar einen generischen, in jedem Fall aber einen Zugriff auf ausgewählte, proprietäre Dienste ermöglichen.

Der Trend zur jederzeit nutzbaren und allgegenwärtigen Informationsplattform wird also anhalten und sich voraussichtlich sogar noch verstärken, indem webbasierte Informationssysteme auf weiteren Geräten verfügbar gemacht werden. Der technische Aspekt des Internet tritt dabei jedoch in der Mehrheit der Fälle immer mehr in den Hintergrund und die Vernetzung wird weniger sichtbar.

Dem Endkunden werden somit mehr und mehr einzelne, spezialisierte Appliances und weniger komplexe, technische Systeme geliefert, die ein möglichst breites Einsatzspektrum abdecken sollen.

Schließlich bleibt noch die Frage, welcher Art die über ein webbasiertes Informationssystem angebotenen Informationen sein können. Beachtet man dabei insbesondere die eben angesprochene Entwicklung zu Wearable Devices und Appliances [GAT95], steht fest, dass neben den klassischen Informationssystemen, die hauptsächlich der (statistischen) Analyse umfangreicher Datenmengen, dem Nachschlagen von Informationen oder der Bestellabwicklung dienen, und deren Standort lediglich in das Internet verlagert wurde, vor allem personalisierte Dienste und Location Based Services (LBS) interessant sind.

2.2 OLAP und OLTP

Bei webbasierten Informationssystemen wird häufig zwischen zwei Typen von Anwendungen unterschieden. Auf der einen Seite stehen die Anwendungen, die hauptsächlich der benutzerfreundlichen Analyse bestehender Datenmengen, oft von Data Warehouses, dienen.

Der Typ dieser Anwendungen wird als Online Analytical Processing (OLAP) [3, 4] bezeichnet,

Sie arbeiten in der Regel multidimensional und dienen der Extraktion betriebswirtschaftlich relevanter Kennzahlen, auf deren Basis dann mit Hilfe von Decision Support Systems (DSS) Entscheidungen getroffen werden können.

Die Darstellung dieser Kennzahlen erfolgt oft in Form eines mehrdimensionalen Würfels, der in weitere, kleinere Würfel unterteilt werden kann, um zu Detailansichten zu gelangen.

Auf der anderen Seite stehen die Applikationen, die hauptsächlich nicht der Analyse, sondern der erzeugenden und verändernden Verarbeitung der Daten dienen.

Dieser Typ von Anwendung wird als Online Transactional Processing (OLTP) [5] bezeichnet.

Während OLAP-Anwendungen in der Regel umfangreiche, aber vorgefertigt vorliegende Anfragen verarbeiten, müssen OLTP-Anwendungen kurze, aber dafür individuelle Anfragen verarbeiten. Da

die Anfragen hierbei im Voraus nicht optimiert werden können, müssen OLTP-Systeme einen besonders hohen Datendurchsatz ermöglichen.

3 Anforderungen

3.1 Schnittstelle zum Menschen

Webbasierte Informationssysteme können eingesetzt werden, um einen vollkommen elektronisch ablaufenden Austausch von Informationen über standardisierte Protokolle zu ermöglichen. Obwohl dieses Szenario im B2B-Bereich (Business to Business) durchaus seine Anwendung findet, sind die meisten webbasierten Informationssysteme zur Kommunikation mit einem Menschen ausgelegt.

Dies bedeutet allerdings, dass diese Systeme über eine verständliche, übersichtliche und möglichst intuitive Benutzeroberfläche verfügen müssen. Auch wenn die Bedienbarkeit beispielsweise von Webseiten in den letzten Jahren im Durchschnitt verbessert hat und auch die entsprechenden Werkzeuge (Browser, ...) verbesserte Oberflächen aufweisen, tun sich dennoch vor allem Einsteiger beispielsweise mit unterschiedlichen Terminologien schwer (wie zum Beispiel mit „registrieren“, „anmelden“ und „login“).

In diesem Bereich sind daher noch zahlreiche Verbesserungen möglich. Auch die Bindung webbasierter Informationssysteme an ein bestimmtes Gerät – in der Regel den Computers – könnte gelöst werden, indem Technologien vermehrt in alltäglich genutzte Geräte oder sogar Kleidung integriert werden.

3.2 Interoperabilität von Applikationen

Je mehr webbasierte Informationssysteme eingesetzt werden, desto mehr gewöhnen sich Benutzer an die vielfältigen Möglichkeiten und desto komplexer werden auch die Anforderungen, die an die Systeme gestellt werden.

Um den immer komplexer werdenden Anforderungen gerecht zu werden, bietet es sich oftmals an, an Stelle eines großen, monolithischen Informationssystems mehrere kleinere aneinander zu koppeln und deren Datenbestände zu verbinden.

Dazu ist es allerdings notwendig, dass die einzelnen Systeme sich untereinander in standardisierten Sprachen und über genormte Protokolle austauschen können, das heißt, ihre Interoperabilität muss sichergestellt werden.

3.3 Integration von Altapplikationen

Häufig lösen webbasierte Informationssysteme bestehende Systeme ab, mit der Zielsetzung, die Handhabung und Bereitstellung der Daten zu vereinfachen. Allerdings ist es oft nicht möglich, bestehende Altapplikationen von heute auf morgen zu ersetzen, so dass häufig eine komplette oder teilweise Integration notwendig ist.

In manchen Domänen sind auch noch Systeme zu finden, die so alt sind, dass eine Veränderung des Status quo gefährlich für den Ablauf des Geschäftsbetriebes sein könnte, da niemand die Folgen abschätzen kann. Man denke beispielsweise an die in zahlreichen Banken und Versicherungen noch eingesetzten in Cobol oder Fortran geschriebenen Großrechnerapplikationen.

Deshalb bieten einige Hersteller, wie beispielsweise Microsoft mit dem Host Integration Server [6], eigens Produkte für den Zweck an, diese Altapplikationen zu kapseln und sie so mit Schnittstellen zu modernen System auszustatten.

3.4 Verteilte Applikationen

Da webbasierte Informationssysteme teilweise zur Bearbeitung von Aufgaben in geschäftskritischen Bereichen eingesetzt werden und ein Ausfall derselben einen schweren wirtschaftlichen Schaden bedeuten könnte, müssen diese Systeme als verteilte Applikationen realisiert werden können.

Dies bedeutet zum einen die Bereitstellung redundanter Systeme, bei denen beispielsweise – vom Anwender unbemerkt – bei einem Ausfall ein zweites System automatisch das erste ersetzt, so dass dieses ersetzt oder repariert werden kann.

Zum anderen spielt auch die Replikation [7] eine wichtige Rolle, die oft bei Datenbanken zum Einsatz kommt und bei welcher der gleiche Datenbestand konsistent in mehreren Systemen gehalten wird, die sich beispielsweise die Last bei vielen gleichzeitig auftretenden Anfragen teilen können. Ein nicht zu vernachlässigendes Problem bei der Replikation ist die Sicherstellung der Synchronisation und der Konsistenz der einzelnen Systeme sowie die Personalisierung von großen Anfragen, die sich über mehrere Schritte erstrecken.

3.5 Push-Technologien

Mit einer steigenden Zahl von angebotenen Diensteleistungen und Nachrichten, die über webbasierte Infornamtionssysteme angeboten werden, wird es für den Anwender zunehmend schwieriger, den Überblick über alle Informationsquellen zu behalten, sofern er die für ihn potentiell relevanten Informationen händisch abrufen muss (Pull-Technologie).

Daher bietet es sich an, dem Anwender automatisch die für ihn interessanten Informationen zuzustellen, so dass ihm diese ohne sein Zutun zur Verfügung stehen (Push-Technologie). Ein solches Vorgehen bietet sich vor allem für Informationen an, die regelmäßig erscheinen oder regelmäßig aktualisiert werden, wie beispielsweise Nachrichtendienste.

Im Idealfall stellt ein webbasiertes Informationssystem seine Daten so wohl mittels Pull- wie auch mittels Push-Technologie bereit. Eine entsprechende Technologie, die sich so wohl für Pull- wie auch für Push-Übertragung von Daten eignet, ist beispielsweise das auf XML basierende RSS (Really Simple Syndication) [8].

3.6 Skalierbarkeit

Häufig werden webbasierte Informationssysteme zunächst nur von einem kleinen Kreis von Anwendern genutzt. Oft genug steigt aber auch der Bekanntheitsgrad der Systeme nach einziger Zeit rapide an, so dass sie ein Vielfaches der ursprünglichen Last zu tragen haben.

Hierfür ist es vorteilhaft, wenn diese Systeme gut skalieren, das heißt, dass sie zum einen überhaupt in der Lage sind, deutlich höhere Lasten zu verarbeiten, und, dass sie zum anderen proportional mehr Rechenleistung oder Speicherplatz benötigen (wenn sich die Last beispielsweise verzehnfacht, sollte die für die Bearbeitung nötige Rechenleistung nicht um den Faktor hundert steigen).

Bis heute ist es noch nicht gelungen, ein perfekt skalierendes System zu bauen, da alle derzeit existierenden Systeme bei höherer Last auch mehr Overhead erzeugen, der einen Teil der zusätzlich bereit gestellten Rechenleistung oder des zusätzlichen Speicherplatzes wieder zunichte macht.

3.7 Sicherheit

Der Begriff „Sicherheit“ wird für webbasierte Informationssysteme in zweierlei Hinsicht verwendet. Zum einen ist damit die Betriebssicherheit gemeint, das heißt, wie stabil ein solches Produkt läuft, wie zuverlässig es erreichbar ist und wie zuverlässig es bei der Ermittlung von Resultaten ist.

Zum zweiten – und das ist der kritischere der beiden Punkte – ist mit Sicherheit aber auch die Wahrung der Privatsphäre gemeint. Gerade bei Systemen, die weltweit und jederzeit, und somit quasi auch für jedermann, zur Verfügung stehen, und die häufig auch sehr viele persönliche Daten speichern, ist dies ein wesentlicher Aspekt.

Ebenso wie bei der Skalierbarkeit gibt es hierbei noch keinen wirklich zufriedenstellenden und überzeugenden Ansatz, da viele Entwickler, die webbasierte Informationssysteme erstellen, Sicherheit eher als lästiges Übel denn als Selbstverständlichkeit betrachten. Da es keine technischen Verfahren gibt, die Sicherheit eines Systems zu garantieren, ist gerade bei

komplexen Systemen eine entsprechend ausgereifte Planung und ein umfangreiches Testen notwendig.

4 Architekturen

4.1 Dokumente

Eine dokumentenzentrierte Architektur stellt die einfachste Architektur dar, Daten zu teilen, da hierbei einfach Dateien innerhalb eines oder mehrerer Dateisysteme freigegeben werden und auf diese somit von mehren Benutzern zugegriffen werden kann.

Der größte Nachteil an dieser Architektur ist, dass eine Datei (beziehungsweise ein Dokument) nicht notwendigerweise eine in sich geschlossene logische Einheit ist. So können Dateien entweder mehrere voneinander unabhängige logische Einheiten enthalten (ein Exceldokument kann beispielsweise mehrere Arbeitsblätter enthalten, bei denen keine semantische Abhängigkeit vorliegt) oder es können mehrere Dateien notwendig sein, um die gewünschte Information darzustellen (wie beispielsweise bei Webseiten, die in der Regel aus Text und Bildern bestehen, die sich in unterschiedlichen Dateien befinden).

Ein Beispiel für eine entsprechende Software ist beispielsweise der als Opensource entwickelte Dateifreigabeserver Samba [9], der in einem Netzwerk ein zentrales Dateienverzeichnis anbietet, auf das von allen angeschlossenen Arbeitsstationen zugegriffen werden kann.

4.2 Daten

Eine datenzentrierte Architektur versucht die Nachteile der dokumentenzentrierten Architektur zu lösen, indem nicht eine Datei als die kleinste eigenständige Einheit gesehen wird, sondern Daten als Informationsblöcke.

Diese Informationsblöcke werden dabei unabhängig von der zu Grunde liegenden Dateiorganisation betrachtet. Eine solche Architektur bietet Microsoft in Windows beispielsweise seit geraumer Zeit mit den Technologien OLE (Object Linking and Embedding) und DDE (Dynamic Data Exchange) [10].

4.3 Dienste

Eine dienstzentrierte Architektur stellt nicht die Daten in den Vordergrund, sondern die Methoden, die sich auf die Daten anwenden lassen.

Neuerdings werden mittels dienstzentrierter Archiutekturen oftmals Web Services angeboten, die eine gewisse Funktionalität eines webbasierten Informationssystems zur Weiterverwendung für andere Dienste bereitstellen.

Eine dienstzentrierte Architektur bietet für kommerzielle Anbieter auch neue Wege der Lizenzierung, indem Benutzer beispielsweise nur für gewisse Funktionen freigeschaltet werden, so dass sie zwar vollen Zugriff auf die Daten haben, diese aber nur ihrem Kundenstatus entsprechend bearbeiten können.

4.4 Mainframes

Mainframes sind extrem leistungsstarke Computer, die vor allem auf hohen Datendurchsatz und einen stabilen Betrieb ausgelegt sind. Sie bieten vielen Benutzern die Möglichkeit, sich mittels Terminals zu verbinden und Rechenleistung und Speicherplatz in Anspruch zu nehmen.

Mit der zunehmenden Verbreitung von PCs sank die Verbreitung von Großrechnern in den 80er Jahren drastisch, so dass sie heute im Wesentlichen nur noch zur Bereitstellung von Altapplikationen dienen und meistens zunächst in eine moderne Client-Server-Landschaft integriert werden und über kurz oder lang durch ein entsprechend modernes System abgelöst werden.

4.5 Client / Server

Das Client-Server-Modell ist aus dem der Mainframes hervorgegangen und bietet mit dem Server

einen zentralen Rechner, der Rechnleistung und Speicherplatz zur Verfügung stellt. Als Clients fungieren hierbei im Gegensatz zu den Mainframes, bei denen proprietäre Terminals benötigt wurden, normale PCs, die nicht nur mehr Aufgaben übernehmen als die Terminals, sondern teilweise auch eigenständige Aufgaben erledigen können.

Hierbei muss noch unterschieden werden zwischen der Serverhardware und der Serversoftware. Während Serverhardware in der Regel spezielle Computer sind, die mit besonders zuverlässigen Komponenten ausgestattet sind, kann Serversoftware prinzipiell auf jedem beliebigen Computer eingesetzt werden.

Da in einem Netzwerk oftmals mehrere Arten von Serversoftware gleichzeitig zum Einsatz kommen, kann es durchaus mehr als einen zentralen Server geben, zudem können einzelne Rechner so wohl als Server wie auch als Client dienen.

4.6 Multi-Tier

Die einfachste Form einer Multi-Tier-Architektur [CUN02, SCH02] besteht nur aus einem einzigen Tier, so dass das webbasierte System auf der selben Plattform bereitgestellt wird, auf der auch die Abfragen ausgeführt werden – eventuell handelt es sich bei beiden Teilen sogar um die gleiche Applikation, so dass nicht einmal hier eine Trennung vorliegt. Diese Architektur wird bei webbasierten Informationssystemen in der Regel nicht eingesetzt, da sie den webbasierten Ansatz ad absurdum führt.

Die nächstkomplexere Form von Multi-Tier-Architekturen besteht aus zwei Tiern und ist bereits für webbasierte Informationssysteme anwendbar. Sie spiegelt im Wesentlichen die bereits erwähnte Struktur aus einem zentralen Server und mehreren Clients wieder. Hierbei stellt der Server in der Regel so wohl die Daten wie auch die eigentliche Suche auf den Daten zur Verfügung, die Clients verfügen dagegen nur über eine Oberfläche zur Anfragegestaltung und zur Anzeige der Ergebnisse.

Zwei Schlagworte in diesem Bereich sind „Thick Client“ und „Thin Client“, die sich darauf beziehen, wie viel Funktionalität direkt auf dem Client bereitgestellt wird. Während Thick Clients viele Funktionen enthalten und somit für mehrere aufeinander folgende Abfragen eventuell weniger Roundtrips zum Server benötigen, bieten Thin Clients im Wesentlichen nur die Benutzeroberfläche zur Steuerung der auf dem Server laufenden Applikation an.

Multi-Tier-Architekturen können über weitere Tier verfügen, wobei in der Regel die Serverseite weiter aufgespalten wird, beispielsweise in einen Webserver und einen nochmals dahinter liegenden, eigenständigen Datenbankserver. Mehr als zwei Tier einzusetzen kann beispielsweise zur Sicherung der Redundanz und somit der Bereitstellung von Hochverfügbarkeitssystemen dienen.

5 Web Services

Web Services bieten eine Möglichkeit, die angebotenen Funktionen von webbasierten Informationssystemen von der Darstellung der Daten zu trennen. Web Services sind letztendlich nichts anderes als über ein Netzwerk (in der Regel das Internet) leicht aufrufbare Methoden, die Daten abfragen oder manipulieren und das Ergebnis in einer standardisierten Form an den Aufrufer zurückgeben.

Die häufigste Anwendung finden Web Services im Bereich des Enterprise Computing, wo deren Erstellung durch entsprechende Tools erleichtert wird. So bieten beispielsweise die beiden im Enterprise-Umfeld weit verbreiteten Architekturen J2EE [11] und Microsoft .NET [12] beide einfache Möglichkeiten an, Web Services zu erstellen und zu publizieren.

Durch die Verwendung von Web Services werden einige der Anforderungen an webbasierte Informationssysteme besser erfüllt als durch die Anwendung klassischer Methoden. So ist beispielsweise auf Grund der verwendeten standardisierten Sprachen und Protokolle für Web Services – diese basieren in der Regel auf XML oder einem XML-Dialekt und nutzen unter anderem SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) und UDDI (Universal Discovery and Description Interface) [PLA01, VAS01] - eine sehr gute Interoperabilität realisierbar.

Allerdings liefern Web Services ihre Ergebnisse an den Aufrufer in der Regel in XML – also einem eher von Maschinen als von Menschen lesbaren Format – zurück, so dass es hier zusätzlicher Komponenten bedarf, um die Schnittstelle zum Menschen herzustellen.

6 Grids

Grids sind eine sehr moderne Entwicklung, die unter anderem vom Kernforschungszentrum CERN vorangetrieben wird. Während inzwischen für viele Aufgaben genügend Rechenleistung und Speicherplatz zur Verfügung steht, mangelt es oft an der Verknüpfbarkeit der angebotenen Informationen.

Deshalb forscht das CERN an einer neuartigen Methode, Rechner miteinander zu vernetzen, so dass ein Benutzer nicht mehr – wie bisher – mittels einer Suchmaschine nur nach vorgegebenen und vorher indizierten Begriffen suchen kann, sondern dass er beliebige Fragen an „das Grid“ stellen kann, welches sich die zur Antwort nötigen Informationen, Daten und Zusammenhänge selbständig sucht.

Des weiteren verwenden einige Hersteller (beispielsweise IBM und Oracle) den Begriff „Grid“ bewusst in ihren Produktnamen, um eine besonders leistungsfähige Netzwerkfähigkeit hervorzuheben. Allerdings wird dieses Wort genau wie der Begriff „Web Services“ aus Marketinggründen häufig in einem falschen Kontext gebraucht.

7 P2P-Computing

P2P (Peer to Peer) stellt ebenso wie ein Grid eine Methode zur Zusammenschaltung einzelner Resourcen für ein komplexes webbasiertes Informationssystem dar. Der wesentliche Charakterzug beim P2P-Computing ist, dass die einzelnen Teilnehmer alle gleichberechtigt sind und es keinen zentralen Server als Anlaufstelle gibt.

Anfang der 90er Jahre versuchte Microsoft, mit Windows 3.11 für Workgroups P2P-Netze für den Gebrauch in kleinen und mittleren Netzwerken als schlanke Alternative zu umfangreichen und kostspieligen Client-Server-Verbindungen zu etablieren, was allerdings scheiterte.

Danach wurde es um P2P-Technologien relativ ruhig, bis die ersten Tauschbörsen auftauchten. Seitdem hat P2P eine enorme Verbreitung erfahren und wird tagtäglich von Millionen von Menschen genutzt, um Daten wie Musik, Videos oder Textdokumente untereinander auszutauschen.

Ein Vorteil bei P2P ist die Dezentralisierung der Verwaltung, das heißt, es gibt nicht einen zentralen Knoten, von dessen Funktionsfähigkeit das ganze Netzwerk abhängt. Der Wegfall eines Knotens schmälert die Leistung des Gesamtsystems daher nur unwesentlich, wodurch sich deutlich robustere Netze aufbauen lassen.

8 Webbasierter Datenbankzugriff

8.1 CGI

Die einfachste und historisch gesehen wohl auch älteste Möglichkeit, aus einem webbasierten Informationssystem auf Datenbanken zuzugreifen, stellt ein CGI-Skript [DOU02] dar (Common Gateway Interface), das in einer beliebigen Programmiersprache geschrieben werden kann, wobei in der Praxis häufig Perl dazu eingesetzt wird.

CGI-Skripte sind in der Regel unkompliziert zu entwickeln und können relativ schnell abgearbeitet werden, haben aber auch etliche Nachteile, die sich vor allem im Mehrbenutzerbetrieb bemerkbar machen. So muss beispielsweise für jeden Zugriff das CGI-Skript neu geladen und (je nach verwendeter Sprache) eventuell noch interpretiert werden.

Ein Caching oder ein Pooling ist nicht vorgesehen. CGI ist daher eher für kleinere Aufgaben geeignet, bei denen es nicht all zu sehr auf Performance ankommt.

8.2 SSI

SSI (Server Side Includes) [13] sind eine Erweiterung der Sprache HTML, die dynamische Aspekte in das an sich statische HTML bringen können. SSI wird – ähnlich wie beispielsweise PHP – auf dem Server vor der Auslieferung einer Webseite interpretiert und das Ergebnis in die HTML-Seite eingefügt, die dann an den Client ausgeliefert wird.

Da diese Interpretation bei jedem Aufruf der Seite neu durchgeführt wird, ist SSI eine relativ langsame Technologie, die heutzutage auch nur noch sehr vereinzelt eingesetzt wird. Zudem sind die sprachlichen Möglichkeiten und damit die Möglichkeiten des Einsatzes im Vergleich zu moderneren Technologien stark eingeschränkt.

8.3 Java Applets

Java-Applets [11] waren der erste Versuch Suns, die Programmiersprache Java für den Einsatz im Web zu positionieren. Java-Applets haben prinzipiell den vollen Zugriff auf alle Bibliotheken der Java-Architektur und können somit auch problemlos auf Datenbanken zugreifen.

Allerdings können Java-Applets nicht als eigenständige Programme ausgeführt werden, sondern benötigen einen Browser, der mit einem entsprechenden Plug-In ausgestattet sein muss. Dieser Umstand sowie die langen Ladezeiten führten dazu, dass sich Java-Applets nie wirklich durchsetzen konnten.

Nachdem Sun diesen Missstand erkannt hatte, wurde Java mittels J2EE (Java 2 Enterprise Edition) für den Einsatz auf Servern neu konzipiert, wobei der Datenbankzugriff hier in der Regel über JDBC (siehe unten) oder EJB (Enterprise Java Beans) abgebildet wird.

8.4 ISAPI

Um die Leistungs- und Skalierungsprobleme von CGI zu überwinden, entwickelte Microsoft die ISAPI-Technologie (Internet Server Application Programming Interface) [DOU02]. Mit ISAPI war es nun möglich, serverseitige Erweiterung in kompilierter Form als von mehreren Webapplikationen gemeinsam nutzbare DLL bereitzustellen.

Obwohl mit ISAPI etliche Probleme von CGI gelöst waren, ergaben sich neue, da ISAPI ein völlig neues Programmiermodell verwendete und erforderte, dass sich ein Programmierer nicht nur in HTML, sondern auch in C++ und der MFC (Microsoft Foundation Classes) auskannte.

Beide Fähigkeiten für sich waren zum Zeitpunkt der Einführung von ISAPI durchaus gängig, aber es gab kaum Programmierer, die beides entsprechend gut beherrschten.

8.5 JDBC

JDBC (Java Database Connectivity) [11] ist eine in Java integrierte Technologie, die es in Java geschriebenen Applikationen ermöglicht – ähnlich wie mit ODBC – auf Datenbanken zuzugreifen. Dabei ist allerdings für jede Datenbank, die angesprochen werden soll, ein eigener JDBC-Treiber nötig.

Es gibt zwar einige weitere Technologien unter Java (beispielsweise JDO oder EJB), um auf Datenbanken zuzugreifen, aber diese befinden sich entweder noch im Entwicklungsstadium oder werden eher in Spezialfällen eingesetzt. Im Regelfall wird der Datenbankzugriff unter Java mittels JDBC hergestellt.

8.6 ADO .NET

Mit ADO .NET (ActiveX Data Objects .NET) [SCE02, SCH02] bietet Microsoft seit 2001 einen objektorientierten Nachfolger der ADO-Technologie (ActiveX Data Objects) an, die vor allem unter COM, COM+ und DCOM weit verbreitet war.

Um der zunehmenden Verbreitung von webbasierten Informationssystemen gerecht zu werden, arbeitet ADO .NET im Gegensatz zu seinem Vorgänger nun verbindungslos. Das heißt, dass eine Verbindung zu einer Datenbank nur so lange offen gehalten wird, wie für die Übertragung der Daten notwendig ist.

Danach können diese Daten ohne Einschränkung verarbeitet und editiert werden, ihre Aktualisierung findet aber zunächst nur auf dem Client statt. Um die Daten schließlich auch in der Datenbank und somit auf dem Server zu aktualisieren, muss eine erneute Verbindung aufgebaut werden.

Dieses Verfahren erzeugt auf Grund der notwendigen Verbindungssteuerung zwar einen größeren Overhead als verbindungsorientierte Systeme, allerdings eignet sich dieser Ansatz deutlich besser, um Datenbanken über unzuverlässige Verbindungen (wie beispielsweise das Internet) zu verwenden. Zudem können die Performancenachteile, die durch das ständige Auf- und Abbauen der Verbindungen notwendig sind, teilweise durch Methoden wie beispielsweise durch Connection Pooling aufgewogen werden.

9 Technologien

9.1 Dynamic HTML

Da die meisten webbasierten Informationssysteme heutzutage auf Webseiten aufbauen, denen eine Datenbank zu Grunde liegt, werden im Folgenden einige Technologien vorgestellt, um solche Systeme zu entwickeln.

Die einfachste Lösung, eine Datenbankanbindung für eine Webseite bereitzustellen, führt über Dynamic HTML [ISA97], wobei diese Lösung etliche Nachteile aufweist. Zum einen findet die Datenbankanbindung clientseitig statt, so dass zunächst alle Datenzugriffsmethoden an den Client übertragen werden müssen, und danach nochmals die eigentlich Daten vom Server nachgeladen werden müssen.

Dieser mehrfache Ladeaufwand lässt sich unter Umständen dadurch kompensieren, dass viele Datenmanipulationen auf den Client verlagert und somit sehr schnell abgearbeitet werden können, da nicht für jede kleine Änderung ein erneuter Roundtrip zum Server notwendig ist.

Der zweite, weitaus gravierendere, Nachteil dieser Lösung ist allerdings, dass die notwendigen Technologien browserabhängig und proprietär sind. Im wesentlichen unterstützt nur der Internet Explorer ab Version 4 mit seinem DOM (Document Object Model) die entsprechenden Technologien.

9.2 PHP

Um Browserunabhängigkeit zu ermöglichen, muss die Datenbankanbindung in der Praxis bereits auf dem Server stattfinden, so dass weder die Datenzugriffsmethoden noch die Gesamtheit aller Daten an den Client übertragen werden müssen, sondern nur die für die jeweilige Anfrage relevanten Ergebnisdaten.

Das Opensource-Projekt PHP (Personal Hypertext Preprocessor) [14] bietet hierzu eine für den Einsteiger leicht zu erlernende Möglichkeit, bei welcher der auf dem Server auszuführende Code ähnlich wie bei SSI in HTML-Seiten eingebettet wird.

Dies ermöglicht auf der einen Seite eine gute Integration von Datenbank- und HTML-Code, verschlechtert auf der anderen Seite aber die Wartbarkeit drastisch, was durch die „fehlertolerante“ Syntax von PHP noch verstärkt wird.

Zudem ist PHP auf Grund der Tatsache, dass es sich um eine interpretierte Sprache handelt, und einer schlechten Skalierbarkeit sowie fehlender Replizierungsmechanismen nur bedingt für den Einsatz im Enterprise Computing geeignet, wo es auf Hochverfügbarkeit und hohe Performance ankommt.

9.3 J2EE

Mit J2EE [11] bietet Sun eine Erweiterung der Bibliotheken von Java an, die speziell im Hinblick auf den Einsatz auf Servern und im Enterprise Computing entwickelt wurden. J2EE ermöglicht es, komplette Java-Applikationen auf einem Server abzulegen und diese als Basis für HTML-Seiten zu nutzen, welche die Schnittstelle zu dem Benutzer bereitstellen.

J2EE bietet zum einen die Möglichkeit, den Javacode direkt in die HTML-Seiten einzubetten und somit JSP-Seiten (Java Server Pages) zu erstellen, wobei sich prinzipbedingt aber ähnliche Probleme bezüglich der Wartbarkeit wie bei PHP ergeben. Zum anderen kann der Javacode in eigene Klassen (sogenannte Servlets) ausgelagert werden, wodurch eine saubere Trennung von Geschäftslogik und Darstellung möglich ist.

Für J2EE gibt es zudem eine ganze Reihe von Design Patterns [15], deren Einsatz für den Bau von Applikationen im Enterprise-Markt unerlässlich sind, wie beispielsweise MVC (Model View Controller) oder Front Controller.

Allerdings hat auch J2EE einige Nachteile, so wird zum einen für die Ausführung der Java-Applikation auf dem Server eine eigene Software, ein sogenannter Application Server, benötigt, die häufig nicht untereinander kompatibel sind und somit die Portabilität von Anwendungen erschweren. Zum anderen ist man als Entwickler – wie generell bei der Java-Architektur – auf die Sprache Java festgelegt.

9.4 ASP .NET / Mono

Im Rahmen von .NET [12] bietet inzwischen auch Microsoft eine mit J2EE vergleichbare Technolie namens ASP .NET (Active Server Pages .NET) [16, DOU02, SCH02] an. Die prinzipielle Herangehensweise ist sehr ähnlich zu der von J2EE, so werden bei ASP .NET beispielsweise fast durchgängig die gleichen Design Patterns [18, CUN02] zur Anwendung empfohlen.

Einer der wesentlichen Unterschiede zwischen ASP .NET und J2EE ist zum einen die fehlende Notwendigkeit, einen eigenen Application Server einzusetzen. Ein simpler Webserver (wie beispielsweise der Microsoft IIS, Apache, Cassini oder XSP) reichen zur Bereitstellung von ASP .NET-Applikationen vollständig aus.

Zum Zweiten bringt ASP .NET keine eigene Klassenbibliothek mit, sondern es steht der komplette Umfang des .NET Frameworks zur Verfügung, das heißt, alle Klassen und Methoden, die auch für die Entwicklung von Windows-Applikationen verfügbar sind. Der Datenbankzugriff unter ASP .NET wird mittels des bereits angesprochenen ADO .NET durchgeführt.

Der dritte und vielleicht wesentlichste Unterschied zwischen ASP .NET und J2EE liegt allerdings in der Sprachunabhängigkeit von ASP .NET. So kann die Applikation, die hinter einer ASP .NET-Webseite liegt, in einer beliebigen Sprache geschrieben werden, sofern ein Compiler für .NET existiert. Dies sind derzeit weit über 20 Sprachen, darunter beispielsweise C#, Visual Basic .NET und C++ .NET. Mit J# steht sogar die Sprache Java für ASP .NET zur Verfügung.

Da ASP .NET mit dem kompletten .NET Framework von Microsoft quasi „aus einer Hand“ geliefert wird, ist die Konsistenz innerhalb des Systems und die Integration der einzelnen Techniken deutlich besser als bei J2EE.

Als Nachteil von ASP .NET wird oft angeführt, dass man sich mit dessen Einsatz auf reine Microsoft-Technologien stützen würde. Seit einiger Zeit gibt es als Alternative allerdings das Projekt Mono [17] der Firma Ximian, welches das komplette .NET Framework einschließlich von ASP .NET und ADO .NET als Opensource und auch unter zahlreichen Betriebssystemen wie Linux, Windows und Mac OS X zur Verfügung stellt.

10 Ausblick

Wie eingangs bereits erwähnt, erfreuen sich internetbasierte Anwendungen steigender Beliebtheit. Dieser Trend wird sich auf Grund der immer leichter verfügbaren und zugleich immer mächtigeren Technologien zur Erstellung webbasierter Informationssysteme zukünftig mit Sicherheit verstärken.

Allerdings dürfen bei aller Euphorie über die zahlreichen, neuen Möglichkeiten die Schattenseiten und Probleme dieser neuen Systeme nicht vernachlässigt werden. Gerade in den Bereichen Zuverlässigkeit, Datenschutz und allgemeiner Sicherheit gibt es noch etliche Probleme.

Gelingt es aber, diese Probleme in den Griff zu bekommen, dann bieten webbasierte Informationssysteme auf Grund ihrer allgegenwärtigen und jederzeitigen Verfügbarkeit zahlreiche Möglichkeiten, das Leben der Menschen zukünftig einfacher und komfortabler zu gestalten.

11 Quellenangaben

11.1 Bücher

[CUN02]
Enterprise solution patterns using Microsoft .NET
Ward Cunningham, Microsoft Press 2003

[DOU02]
ASP .NET Anwendungsdesign
Douglas J. Reilly, Microsoft Press 2002

[GAT95]
Der Weg nach vorn
Bill Gates, Hoffmann & Campe 1995

[ISA97]
Inside Dynamic HTML
Scott Isaacs, Microsoft Press 1997

[PLA01]
Microsoft .NET – Eine Einführung
David S. Platt, Microsoft Press 2001

[SCE02]
Microsoft ADO .NET Entwicklerbuch
David Sceppa, Microsoft Press 2002

[SCH02]
Microsoft ASP .NET Entwicklerbuch
Holger Schwichtenberg, Microsoft Press 2002

[VAS01]
Microsoft .NET Crashkurs
Clemens Vasters, Microsoft Press 2001

11.2 Online

[1] **http://www.w3.org**
[2] **http://www.microsoft.de**
[3] **http://www.germany.seagatesoftware.com/holos**
[4] **http://de.wikipedia.org/wiki/OLAP**
[5] **http://de.wikipedia.org/wiki/OLTP**
[6] **http://www.microsoft.com/germany/ms/his**
[7] **http://dev.mysql.com/doc/mysql/de/Replication.html**
[8] **http://www.server-wg.de:8080/schockwellenreiter.de/webworking/rss.html**
[9] **http://www.samba.org**
[10] **http://msdn.microsoft.com**
[11] **http://java.sun.com**
[12] **http://msdn.microsoft.com/net**
[13] **http://www.selfhtml.org**
[14] **http://www.php.net**
[15] **http://java.sun.com/blueprints/corej2eepatterns/Patterns/**
[16] **http://www.asp.net**
[17] **http://www.go-mono.com**
[18] **http://msdn.microsoft.com/patterns**