

■ Ajax: neuer Weg für Webanwendungen

Asynchroner Datenaustausch bringt mehr Dynamik und schnellere Benutzerinteraktion





Wer auf den Webseiten von Google Suggest, Gmail, Google Maps oder Flickr surft, stellt fest, dass einige Webanwendungen aufgetaucht sind, die sich sehr ähnlich wie Desktop-Anwendungen verhalten: Sie sind viel dynamischer, die Benutzerinteraktionen viel schneller im Vergleich zu den klassischen Webanwendungen.

Klassisch vs. Ajax

Traditionelle Webanwendungen sind „Inhalt basiert“, das heißt sie bauen auf HTML-Seiten (Präsentationssicht) auf, die immer wieder neu geladen werden, wenn der Inhalt der Webseite verändert wurde, beispielsweise durch die Reaktion auf Benutzereingaben. Das klassische Model sieht eine synchrone Dateninteraktion zwischen Client und Server vor: Durch Benutzeraktionen auf der Webseite wird ein HTTP-Request generiert, der zum Webserver abgeschickt wird. Die Datenverarbeitung findet auf dem Webserver statt, beispielsweise Datenvalidierung oder Datenverarbeitung. Der Server bereitet eine HTTP-Response und schickt sie in Form einer HTML-Seite zum Client zurück. Der Benutzer kann erst anschließend wieder mit der Anwendung interagieren.

Das Verhalten der Anwendung ist technologisch erklärbar: Die Datenverarbeitung findet auf dem Server statt, der Web-Client muss die notwendigen Informationen für die Seite aus dem Server holen. Für den Benutzer ist es andererseits weniger verständlich. Was macht er, während der Server die Verarbeitung durchführt? Er wartet, besonders wenn die Datenmenge sehr groß ist, die Datenverarbeitung lang dauert und der Server überlastet ist. Er „sieht“ sozusagen, dass die Anwendung mit dem Server kommuniziert, und ist oft frustriert über lange Antwortzeiten. Und warum sollen Benutzer und Benutzerinteraktion abwarten? Warum soll die ganze HTML-Seite neu geladen werden, auch wenn nur ein kleiner Teil der Information auf der Webseite geändert wurde?

Interaktiv arbeiten

Wer mit Desktop-Anwendungen gearbeitet hat, weiß ganz genau, dass im Vergleich die klassischen Webanwendungen weit zurück sind: Mit der Desktop-Anwendung arbeiten die Benutzer praktisch interaktiv.

Wenn die Benutzerinteraktion mit der Anwendung und der Datenaustausch mit dem Server asynchron und im Hintergrund erfolgen könnten und nur der geänderte Inhalt auf der Seite neu geladen werden könnte, dann wären die Webanwendungen viel dynamischer und viel interaktiver. So ein Verhalten kann man mit Ajax erreichen.

Was ist Ajax?

Ajax steht für „Asynchronous Javascript and XML“, das heißt der Datenaustausch zwischen Client und Server erfolgt asynchron in XML mittels Javascript. Es handelt sich um eine Art „Remote Javascript“, bei dem die Benutzerinteraktion, die einen HTTP-Request generiert, jetzt ein Javascript-Aufruf ist, der im Hintergrund mit dem Server kommuniziert, ohne dass

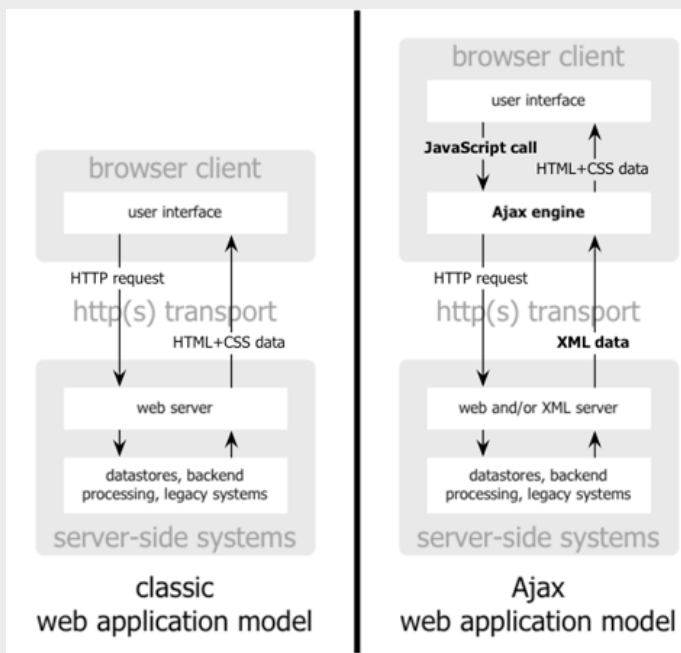


Abb. 1: Klassische Webanwendung und Ajaxanwendung (J. J. Garret)

der Benutzer es weiß. Der Benutzer kann weiter interaktiv mit der Anwendung arbeiten. Der asynchrone Datenaustausch wird durch ein XML HTTP-Request-Objekt ermöglicht, das die Datenschnittstelle definiert.

Die technologischen Voraussetzungen, um damit zu arbeiten, sind heutzutage weit verbreitet:

- ein Javascript-aktivierter Browser, der die XML HTTP-Request-Objekte unterstützt
- ein HTTP-Server, der auf XML ankommende Requests in XML beantworten kann

Neuartige Technologieverbindung

Ajax ist nicht wirklich eine neue Technologie, sondern sie besteht aus verschiedenen verfügbaren Technologien, die auf neue Weise zusammenspielen (siehe auch Glossar):

- XHTML und CSS für die Standardpräsentation der Webseite
- DOM für die Dynamische Anzeige und Dateninteraktion
- XML und XSLT für den Datenaustausch und die Datenmanipulation
- XML HTTP-Request bzw. XML HTTP-Objekte für den asynchronen Datenaustausch zwischen Client und Server

■ Javascript, das alles zusammenbindet.
Das besondere Verhalten einer Ajax-Anwendung wird durch drei Grundkonzepte gesteuert:

- den Ajax Engine,
- das XML HTTP-Request-Objekt,
- das Document Object Model (DOM).

Der Ajax Engine: Der sogenannte Ajax Engine ist eine Schicht zwischen Web-Client und Server, der für die Darstellung der Webseite und der Kommunikation mit dem Server verantwortlich ist. Er verhindert die „Start-Stop-Start-Stop“-Natur der Webinteraktion einer Ajax-Anwendung, das heißt die Benutzerinteraktion mit der Anwendung erfolgt damit asynchron.

Der Ajax Engine ist ein Javascript-Programm, das normalerweise in einem unsichtbaren Fenster im Browser versteckt ist. Geladen mit Beginn einer Session, steuert der Ajax Engine die Zugriffe auf den Server: Er entscheidet, ob der Zugriff auf dem Server notwendig ist oder die direkte Aktion von Ajax

ausreicht. Dadurch ist die gesamte Interaktion mit der Anwendung asynchron, unabhängig von der Kommunikation mit dem Server.

Jede Benutzerinteraktion, die ein HTTP-Request generiert, ist jetzt ein Javascript- Aufruf zum Ajax Engine, der im Hintergrund mit dem Server kommuniziert, ohne dass der Benutzer es weiß. Eine Antwort zur Benutzerinteraktion braucht nicht mehr die Antwort des Servers, wie für Datenvalidierung, Maskennavigation oder Datenverarbeitung: Der Engine handelt alles im Hintergrund ab. Wenn der Engine etwa Informationen vom Server braucht, um zu antworten, holt er die Informationen asynchron aus dem Server via XML, ohne dass die Benutzerinteraktion mit der Applikation behindert wird.

Das XML HTTP-Request-Objekt: Die Datenschnittstelle zwischen Client und Server wird vom XML HTTP-Request-Objekt definiert.

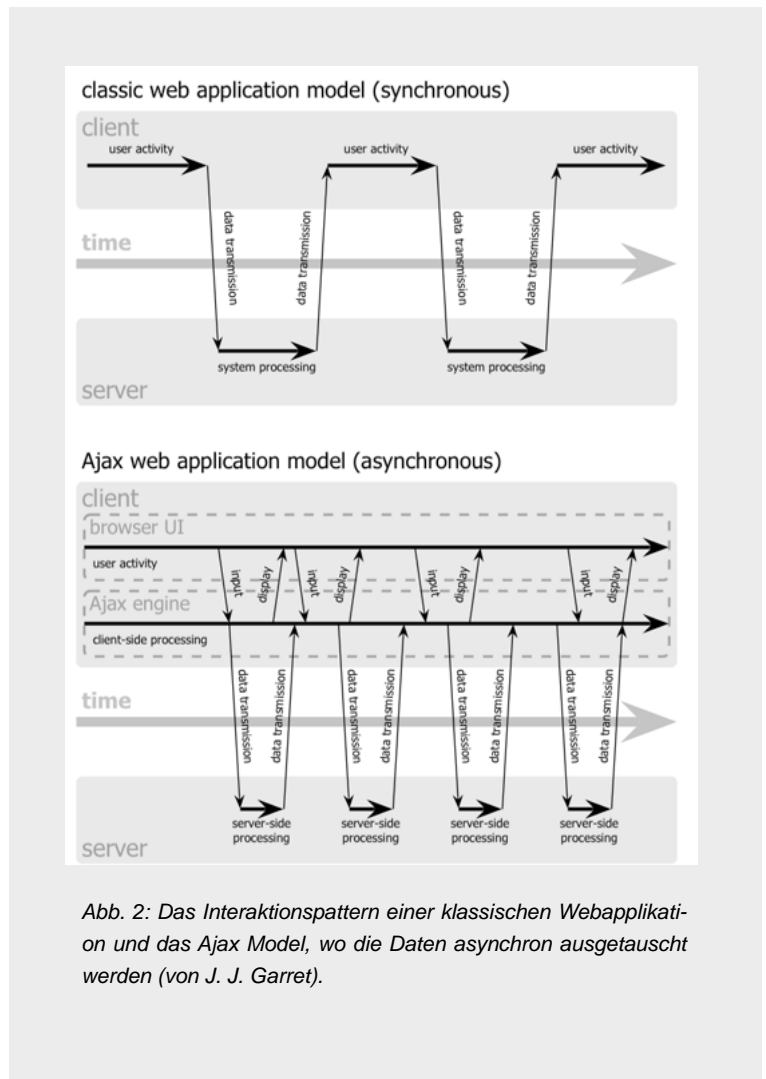


Abb. 2: Das Interaktionsmuster einer klassischen Webapplikation und das Ajax Model, wo die Daten asynchron ausgetauscht werden (von J. J. Garret).



Das Objekt ermöglicht, XML-Daten vom Webserver anzufordern, ohne dass die Seite neu aufgebaut werden muss. Die Benutzerinteraktion ist dadurch kontinuierlich und nicht unterbrochen.

Mozilla-basierte Browser definieren XML HTTP-Request-Objekte, während Internet Explorer ab Version 5.0 ein XML HTTP definiert. Prinzip und Funktionalität sind gleich.

Das Document Object Model: Der Ajax Engine ist für den Datenaustausch verantwortlich. Durch einen DOM-fähigen Browser können Teilinformationen der Seite anstelle des gesamten Seiteninhalts ausgetauscht werden. Dieses Datenmodell ermöglicht, nur die geänderten Daten der Seite anzufordern und zu aktualisieren, wodurch eine Webanwendung auf Benutzerangaben und Interaktionen deutlich schneller antwortet.



Ajax-Nutzer und Ajax-Frameworks

Ajax ist nicht nur Theorie, sondern wird, obwohl noch jung, in der Praxis stark verwendet. Beispiele von realisierten weltweit

bekannten Webanwendungen finden Sie bei Google Suggest, Google Maps, Google Groups oder Gmail: Diese Seiten sind alle in Ajax entwickelt. Die Möglichkeiten, die sie bieten, können Sie selbst anschauen. Sie sind selbsterklärend.

Ajax-Frameworks: Die Entwicklung von Ajax-Anwendungen wird derzeit durch neue Frameworks unterstützt, obwohl die meisten noch im Betatest sind. Für Eclipse- und Java-basierte Entwicklungen eignet sich beispielsweise das Open Source-Projekt der Eclipse Community „RAP“ (Rich Ajax Platform), das die Webentwicklung auf Basis der bekannten Eclipse W4T-Bibliothek sehr ähnlich zur RPC (Rich Client Platform) ermöglicht. Google stellt auch ein Open Source-Framework zur Verfügung, das Java-basierte GWT Framework (Google Web Toolkit), das nur mit Zusatzaufwand mit Eclipse kompatibel wäre. Im .NET-Umfeld stehen u.a.

Ajax im Überblick

	Beschleunigt die Webanwendungen
	Vereinfacht es, Fehleingaben der Benutzer abzufragen
	Verhindert, dass Webseiten stets komplett neu geladen werden müssen
	Ermöglicht Operationen mit Daten aus dem Server, ohne dass die Seite immer wieder geladen werden muss
	Ermöglicht Echtzeit-Datenvalidierung
	Ermöglicht das „Autovervollständigen“ während der Dateneingabe mit Daten aus dem Server
	Ermöglicht anspruchsvolle User Interface Controls wie Tree Controls, Menus, Ladevorgang usw., ohne dass die Seite immer wieder geladen werden muss
	Reduziert Netzwerk-Zugriffe, da nur die notwendige Informationen vom Server geholt werden. Dadurch kann die Performance erhöht werden
	Automatische Aktualisierung von Daten auf der HTML-Seite ohne Benutzerinteraktion
	Serverseitige Notifikation des Clients
Entwicklung mit geringem Aufwand durch die Unterstützung von Frameworks	
	Läuft nur mit modernen Web-Browsern wie Internet Explorer 5 oder Firefox
	Setzt Javascript-fähige und DOM-fähige Browser voraus
	Setzt einen Server voraus, der Daten im XML-Format austauschen kann
	Höhere Komplexität der Präsentations- (Javascript) und der Serversicht (XML-Schnittstellen)
	Javascript-Skills für die Präsentationsschicht, Javascript- und XML-Skills für die Serversicht erforderlich
	Schwieriger zu debuggen
	Mögliche Sicherheitslücken durch Javascript
Einarbeitungsaufwand für die Entwickler (neue Frameworks beherrschen)	

das Microsoft Atlas, der Open Source MagicAjax.NET und der bekannteste, obwohl aufwändig zu konfigurierende Ajax.NET zur Verfügung.

Wie jeder Entwickler weiß, erleichtern und beschleunigen Frameworks die Implementierungsarbeit. Dadurch haben Entwickler Einarbeitungsaufwand. Da jeder Framework unterschiedliche Ansätze bei der Implementierung verwendet, bringen sie viele Vorteile, aber auch Nachteile mit, die die Entwickler unbedingt kennen müssen, um unnötige Aufwände zu vermeiden.

Ergebnis

Die Prinzipien einer Ajax-Anwendung lassen sich folgendermaßen zusammenfassen:

- Der Browser ist das Frontend einer Applikation, nicht „Inhalt-Behälter“.
- Der Server liefert Daten zurück, nicht Inhalt.
- Die Benutzerinteraktionen mit der Anwendung sollten flüssig und ununterbrochen sein.
- Das Ganze ist viel Codierung und benötigt viel Disziplin und Javascript-Knowhow.

Beim näheren Betrachten von Ajax zeigt sich, dass die neuen technologischen Möglichkeiten der Webanwendungen den Abstand zu Desktop-Anwendungen verkürzen. Sie sind viel reicher und interaktiver als die klassischen Webanwendungen. Wenn große Unternehmen wie Google und Amazon viel

Geld dafür investieren, ist zu überlegen, die Technologie auch in den Kompetenzteams von DMC stärker zu nutzen oder zumindest das Knowhow dafür aufzubauen.

Ob man die Ajax-Technologie einsetzen soll oder nicht, hängt, wie jeder Entwickler weiß, stark von der Fachlichkeit ab. Wie immer gilt die Goldene Regel der Softwareentwicklung: die Trennung der Fachlichkeit von der Technik. Die Fachlichkeit bestimmt die zu verwendende Technologie und damit auch, ob der Einsatz der Ajax-Technologie sinnvoll ist. Und die Auswahl des richtigen Frameworks dafür wird wiederum von der Fachlichkeit entschieden, da die Entwicklung sonst unnötig kompliziert werden kann.

Glossar	
Webserver	Ein Webserver ist im engeren Sinne ein Serverdienst (Software), der Informationen über das HyperText Transfer Protocol (HTTP) zur Verfügung stellt.
Thin Client	bezeichnet eine Anwendung oder einen Computer als Endgerät (Terminal) eines Netzwerkes, dessen funktionale Ausstattung auf die Ein- und Ausgabe beschränkt ist.
Rich/Fat Client	verwendet für einen Client, bei dem die eigentliche Verarbeitung der Daten vor Ort auf dem Client vollzogen wird. Der Client stellt auch meistens die grafische Benutzeroberfläche zur Verfügung.
Desktop Anwendung	ist ein zusammenfassender Begriff für Anwendungssoftware, die lokal auf dem Arbeitsplatzrechner installiert ist und über ein eigenes User Interface verfügt – als Abgrenzung zur serverbasierten Webanwendung, welche in einem Browser läuft. Desktop-Anwendungen werden auch als Fat Client- bzw. Rich Client-Anwendungen bezeichnet.
Session	ist eine logische Verbindung zwischen zwei adressierbaren Einheiten in einem Leitungsnetz, um Daten auszutauschen. In der Client-Servertechnik ist es die zeitweise bestehende Verbindung eines Clients mit einem Server.
DOM	Das Document Object Model (DOM) ist eine Programmierschnittstelle (API) für den Zugriff auf HTML- oder XML-Dokumente, die Computerprogrammen erlaubt, dynamisch den Inhalt, die Struktur und das Layout eines Dokuments zu verändern. Sie wird vom World Wide Web Consortium definiert.
CSS	Cascading Style Sheets ist eine deklarative Stylesheet-Sprache für strukturierte Dokumente.
XHTML	Extensible Hyper Text Markup Language ist eine Neuformulierung von HTML 4 in XML 1.0: Im Gegensatz zu ihrem Vorgänger HTML, entsprechen XHTML-Dokumente den Syntaxregeln von XML.
XSLT	XSLT ist eine Programmiersprache zur Transformation von XML-Dokumenten.

Weiterführende Information

Dave Crane, Eric Pascarello, Darren James
„Ajax in Action“, Manning Publications (2005)
<http://adaptivepath.com/publications/essays/archives/000385.php>

Beispiele in Internet

<http://www.google.com/webhp?complete=1&hl=en>
<http://maps.google.com/>
<http://a9.com/-/home.jsp?nc=1>



Ajax- vs. Desktop-Anwendungen

Mancher wird sich nun die Frage nach dem Verhältnis von Ajax- und Desktop-Anwendungen stellen. Zwar ist ein Vergleich zwischen den beiden Technologien schwierig, da sie grundsätzlich unterschiedlich sind. Generell aber lässt sich sagen: Ajax-Anwendungen sind Webanwendungen, die sich sehr ähnlich wie traditionelle Standalone-Desktop-Anwendungen verhalten können. Ein Beispiel ist die schnelle Benutzerinteraktion. Als Webanwendung sind Ajax-Anwendungen einfacher zu bedienen, Deployment und Maintenance sind beispielsweise mit wenig Aufwand und Kosten verbunden. Ein aktueller Javascript-basierter Browser reicht, um auf jedem PC eine Ajax-Anwendung aufzurufen. Sie benötigt jedoch im Allgemeinen mehr Bandbreite für die Kommunikation mit dem Server.

Die Entwickler müssen zusätzliche Technologien und Frameworks beherrschen, und sowohl Client als auch Serverlogik werden komplizierter und umfangreicher. Angesichts dieses zusätzlichen Aufwands muss im Einzelfall abgeschätzt werden, ob er durch die Vorteile der Ajax-Technologie gerechtfertigt und ob deren Nutzung notwendig ist. Auch hier entscheidet wieder die Fachlichkeit.

Desktop-Anwendungen sind dagegen Standalone-Applikationen, die auf den lokalen Arbeitsplätzen installiert, konfiguriert und gewartet werden müssen. Sie funktionieren auch offline und können auf lokale Ressourcen zugreifen. Diese Technologien werden mittlerweile von vielen beherrscht.

■ Ansprechpartner

Dr. John D'Avanzo, Seniorberater, Gruppenleiter Java
Tel. 089 42774-157 • E-Mail john.davanzo@dmc-group.de

■ DMC Datenverarbeitungs- und Management-Consulting GmbH

Valentin-Linhof-Straße 8 • 81829 München
Tel. 089 42774-0 • Fax: 089 42774-199
E-Mail: dmc@dmc-group.de • www.dmc-group.de
Geschäftsführer: Angelo W. Zenz • Dr. Matthias Kulesa