

Eine Webanwendung oder **Webapplikation** ist ein Computer-Programm, das auf einem Webserver ausgeführt wird, wobei eine Interaktion mit dem Benutzer ausschließlich über einen Webbrowser erfolgt. Hierzu sind der Computer des Benutzers (Client) und der Server über ein Netzwerk, wie das Internet oder über ein Intranet miteinander verbunden, so dass die räumliche Entfernung zwischen Client und Server unerheblich ist.

Vorteile

Webanwendungen setzen auf dem Computer des Benutzers nur einen Webbrowser voraus, welcher auf den meisten Systemen schon vorhanden ist. Im Gegensatz zu herkömmlichen Client-Server-Anwendungen ist also keine weitere Installation von Software auf den Computern der Benutzer notwendig, wenn man von Browser-Plugins wie Flash absieht. Dadurch erreichen Webanwendungen einen hohen Grad an Plattformunabhängigkeit, sofern bei der Entwicklung darauf geachtet wurde, dass alle Browser unterstützt werden.

Muss die Logik einer Webanwendung geändert werden, sind Änderungen nur an einer (zentralen) Stelle, nämlich auf dem Webserver, notwendig, was sich günstig auf die Wartungskosten auswirkt.

Durch die immer weiter gehende Verbreitung von Browsern auf andere Endgeräte, wie Mobiltelefone oder PDAs finden Webanwendungen schnell eine Verbreitung jenseits der klassischen Softwareumgebungen.

Nachteile

Eine Webanwendung benötigt eine ständige TCP/IP-Verbindung zwischen dem Client und dem Server; eine dauerhafte Verbindung kann nur durch eine Session erreicht werden. Daraus können sich Sicherheitsprobleme ergeben (siehe unten). Die Bandbreite der Verbindung muss außerdem auf die Anforderungen der Webanwendung ausgelegt sein. Dieser Umstand schließt Webanwendungen für eine Reihe von Einsatzszenarien, wie z. B. die mobile Offline-Benutzung, per Definition aus.

Webanwendungen sollten im Idealfall mit allen Webbrowsern richtig funktionieren, was nicht selbstverständlich ist, da die Browser HTML, trotz bestehender Standards (W3C), unterschiedlich interpretieren. Die leichte Abweichung in der Darstellung zwischen verschiedenen Browsern ist meist unerheblich, verheerender sind Unterschiede in der JavaScript-Interpretation, weshalb häufig Browserweiche verwendet werden müssen, teilweise sogar für unterschiedliche Browser-Versionen. Außerdem ist durch den oben dargestellten Request-Cycle nur eine asynchrone Verarbeitung möglich, was eine Reihe von Anwendungsgebieten (z. B. die Bearbeitung von Videos) als Webanwendung ausschließt oder deutlich erschwert.

Auf dem heutigen Markt für webbasierte Anwendungsentwicklung, lassen sich zwei Technologie-Plattformen identifizieren, die diese Multi-Tier-Architektur unterstützen. Dies sind Microsofts .NET und Sun Microsystems Java EE (früher J2EE).

Unterschiede .NET und Java EE

JAVA EE

Die Java EE-Plattform ist der defacto Standard für die Entwicklung mehrschichtiger, portabler, skalierbarer und stabiler Business-Anwendungen, die mit Basiskomponenten wie z.B. Java Servlet, JavaServer Pages (JSP) oder JavaBeans (EJB) realisiert werden. Ist eine Anwendung Java EE -konform, kann sie auf andere Java EE -Application-Server portiert werden. Da Java EE-Umgebungen Webservices unterstützen, bietet sich diese Plattform für Enterprise Application Integration (EAI) an.

Geprägt und weiterentwickelt wurde die Java EE-Technologie von Sun Microsystems und anderen Key Playern der Softwarebranche. Im Gegensatz zu .NET ist Java EE **herstellerunabhängig** und **läuft auf fast allen Betriebssystemen**. Die Plattform baut auf der objektorientierten Programmiersprache JAVA auf.

Wenn Java EE-Anwendungen gemäß den Java EE-Spezifikationen und offenen Standards entwickelt werden, sind sie portierbar und lassen sich problemlos von einem Java EE - Container zu einem anderen migrieren.

.NET

.NET ist eine von [Microsoft](#) entwickelte Softwareplattform. Diese umfasst eine [Laufzeitumgebung](#), eine für Programmierer bestimmte Sammlung von [Klassenbibliotheken \(API\)](#), und angeschlossene Dienstprogramme (*Services*). Die Plattform ist derzeit in ihrem vollen Umfang nur für Windows verfügbar.

Im Gegensatz zu [Java](#), wurde .NET von Anfang an für den Betrieb mit mehreren Programmiersprachen entwickelt. Die Vorteile der Unterstützung gemischtsprachiger Programmierung von .NET sind nicht unumstritten. Die Wartbarkeit eines Projektes, welches in mehreren Sprachen implementiert wurde, ist geringer als bei der Entwicklung mit nur einer Sprache.

Allerdings wurde im Gegensatz zu Java kein großer Wert auf Plattformunabhängigkeit gelegt.

Eines der wichtigsten Konzepte ist das Sicherheitskonzept von .NET, das weit über das bisher in Windows verankerte oder etwa das von Java hinausgeht.

Das Sicherheitskonzept von .NET fängt an bei Mechanismen, die die Identität des Programmherstellers gewährleisten sollen (Authentizität), geht über Mechanismen zum Schutz der Programme vor Veränderung (zum Beispiel durch Programmviren) bis hin zu Schutzmechanismen, die den Ort der Herkunft bzw. Programmausführung (zum Beispiel Internet) einbeziehen. Es gibt technisch betrachtet, sowohl ein codebasiertes, als auch ein nutzerbasiertes Sicherheitsmodell.

Verteilte Programmierung und Web Services

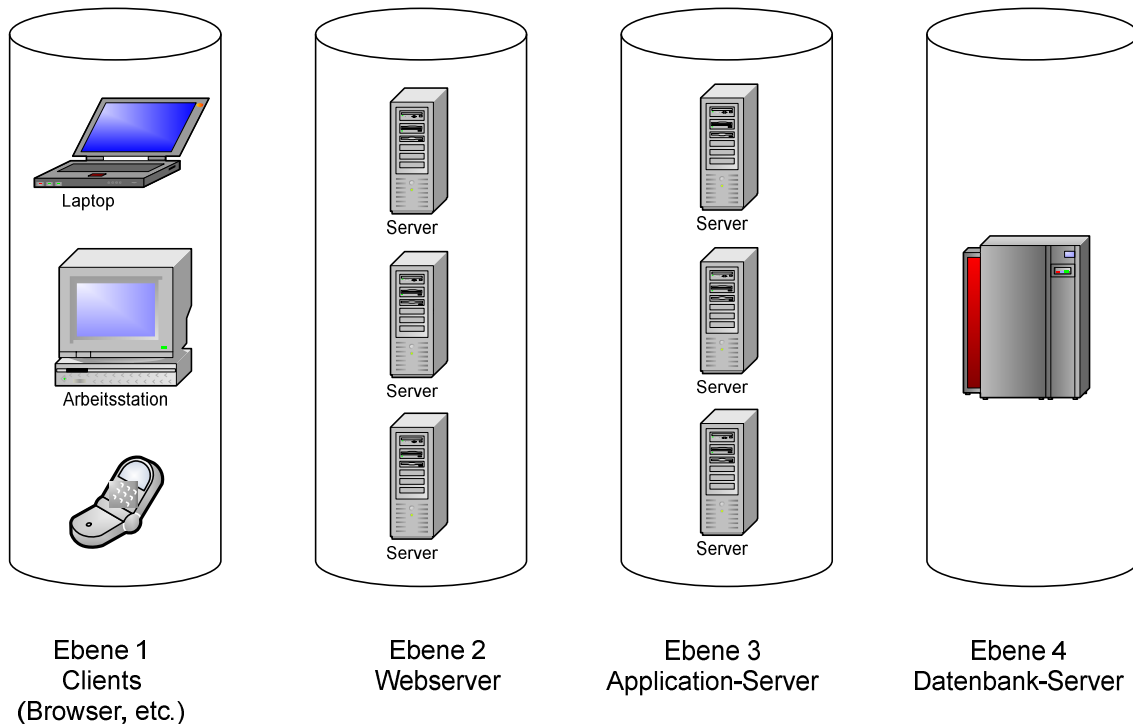
Microsoft bietet hier nicht einen übergreifenden Ansatz, sondern gleich (mindestens) drei unterschiedliche, mit jeweils eigenen Vor- und Nachteilen:

- eine moderne Implementierung des neuen Industriestandards Web Services (manchmal auch „XML Web Services“ genannt): eine Struktur für verteilte Programme, im LAN, aber mit zunehmender Bedeutung auch für nicht kommerzielle Anwendungsgebiete im Internet
- die .NET Remoting Services
- die .NET Enterprise Services

Letztere sollen eine Konkurrenztechnik zu der erfolgreichen J2EE werden.

4-Tier Architektur

Auf dem heutigen Markt für webbasierte Anwendungsentwicklung, lassen sich zwei Technologie-Plattformen identifizieren, die die nachfolgend beschriebene 4-Tier-Architektur unterstützen.
Dies sind Microsofts .NET Plattform und Sun Microsystems J2EE-Plattform.



Ebene 1 – WebClient: Sie visualisiert die Benutzeroberfläche. Hier werden die Daten mittels eines Browsers visualisiert, die auf der Ebene 2 präsentationstechnisch zusammengestellt werden. Gleichzeitig nimmt der Browser die Eingaben des Anwenders entgegen und sendet diese zu Verarbeitungszwecken an die Applikationen auf Ebene 2.

Ebene 2 – WebServer: Ist die Präsentationsebene. Hier werden die Daten applikationstechnisch ausgewertet, die von der Ebene 1 gesendet werden. Diese Daten werden analysiert und ausgewertet, wobei die involvierte Geschäftsprozesslogik auf Ebene 3 angesprochen wird.

Ebene 3 – AnwendungsServer: Hier ist die Geschäftslogik implementiert. Die Daten hierzu werden von Ebene 4 bezogen.

Ebene 4 – DatenbankServer: Hier werden die Daten zur Verfügung gestellt, die für die Geschäftsprozesse benötigt werden, bzw. die Daten gespeichert, welche produziert werden.