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

---

Revision #2
Created 28 February 2025 21:10:49 by Bernd Grabner
Updated 13 February 2026 14:18:15 by Github Admin