Seminar: Web Engineering

Grundlagen von Webanwedungen

Von: Johannes Kettern
Benjamin Süß

Inhalt

Kapitel 1: Grundlagen

- Aufgaben von Webanwendungen
- Abgrenzung: Statische HTML vs. Dynamische Websites
- Architekturen
- Sicherheit von Webanwendungen

Kapitel 2: Technologien

- Ideen
- Serverseitige Technologien
- Clientseitige Technologien

Kapitel 3: Aktueller Stand und Entwicklung

- Rückverlagerung von Logik zum Client
- Probleme beim Aufbau von Webanwendungen
- SOA – Service Oriented Architecture

Aufgaben von Webanwendungen

Definition Webanwendung:

- Allgemeiner Name für **Anwendungen**, der für das **Internet**, ein **Intranet** oder ein **Extranet** erstellt wurde. Dabei wird ein **Browser** als **Benutzerschnittstelle** verwendet, während die **Verarbeitung** auf dem **Server** stattfindet.

Aufgaben von Webanwendungen

Vorteile:

- Clientseitige Voraussetzung: Browser
- Keine weitere Software erforderlich
- Plattformunabhängig
- Änderungen sind nur auf dem Webserver notwendig (Kostenreduzierung)
- Webanwendungen sind auf immer mehr Endgeräten zugänglich (Mobiltelefone)

Aufgaben von Webanwendungen

Nachteile:

- Sicherheitsproblematik
- Ständige TCP/IP Verbindung
- Bandbreite muss auf Anwendung ausgelegt sein
- Einsatzszenarien wie Offline-Benutzung sind nach Definition ausgeschlossen

Kapitel 1: Inhalt

Grundlagen

- Aufgaben von Webanwendungen
- Abgrenzung: Statische HTML vs. Dynamische Websites
- Architekturen
- Sicherheit von Webanwendungen

Statisch vs. dynamisch

Statische Webseiten

- Relativ einfach
- Bestehen aus HTML Code
- Aufbau im Browser durch Kopieren
- Keine Datenbank im Hintergrund
- Modifikationen relativ umständlich
- Erhaltung des Designs problematisch
- Geringe Anfangsinvestitionen, unter Umständen hohe Folgekosten

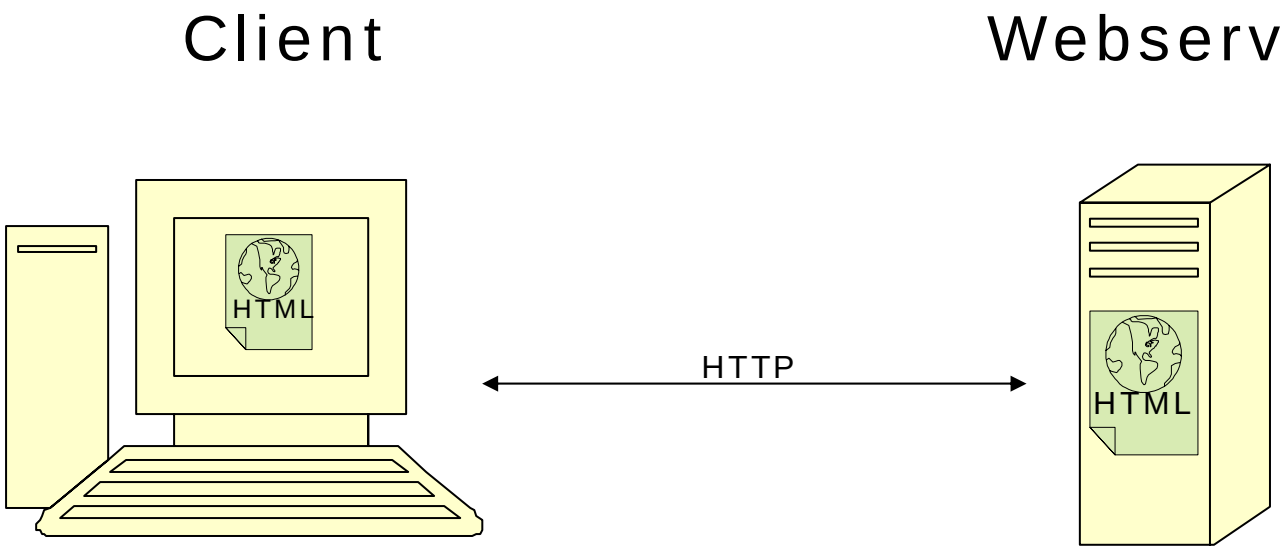
Statisch vs. dynamisch

Dynamische Webseiten

- Baukastenprinzip
- Basiert auf einer Datenbank
- Seite wird erst während des Aufrufs erzeugt
- Änderungen/ Erweiterungen vergleichsweise einfach
- Hohe Anfangsinvestitionen, geringe Folgekosten

Statisch vs. dynamisch

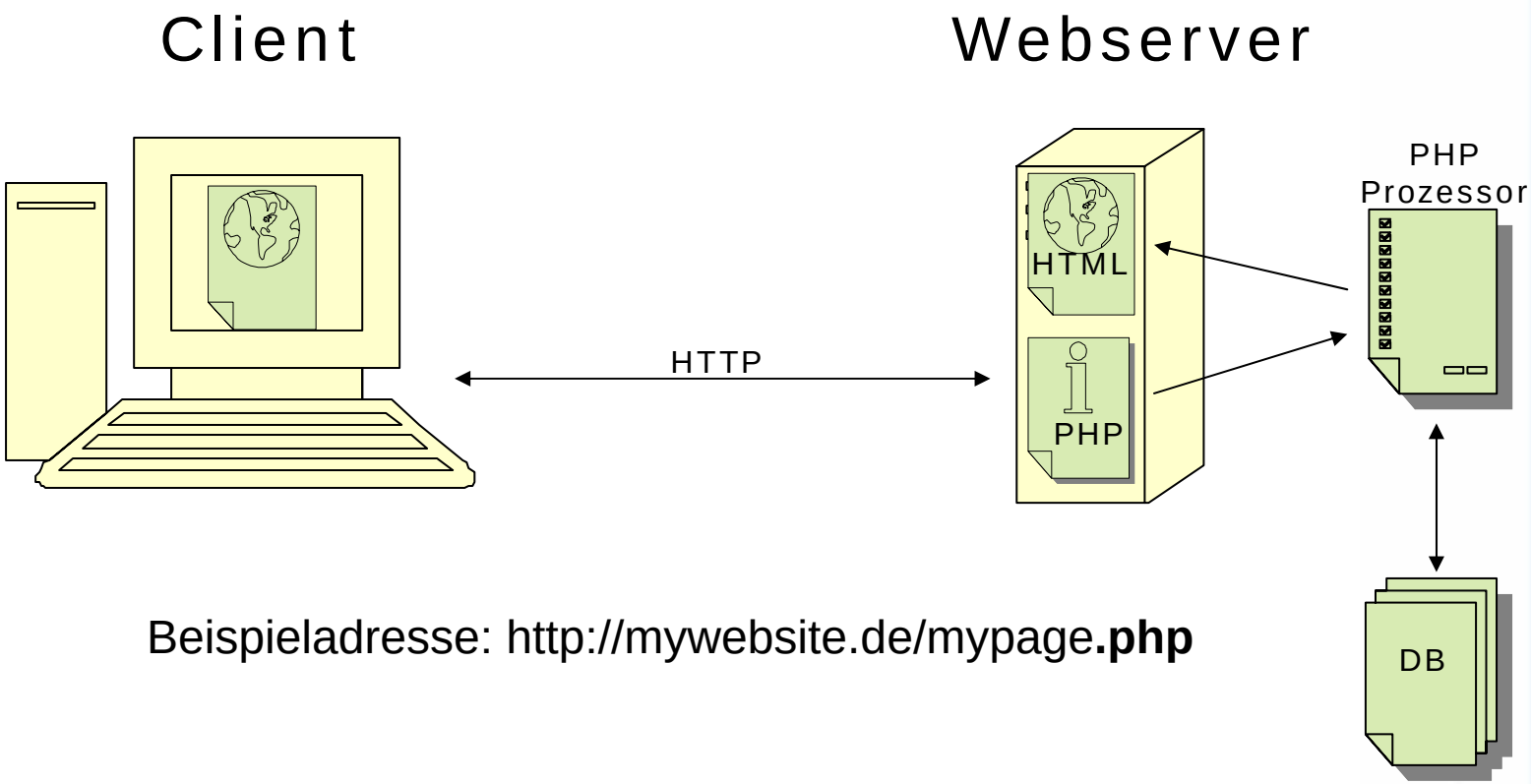
Aufbau statisch



Beispieladresse: http://mywebsite.de/mypage**.html**

Statisch vs. dynamisch

Aufbau dynamisch (PHP als Beispiel)



Beispieladresse: http://mywebsite.de/mypage**.php**

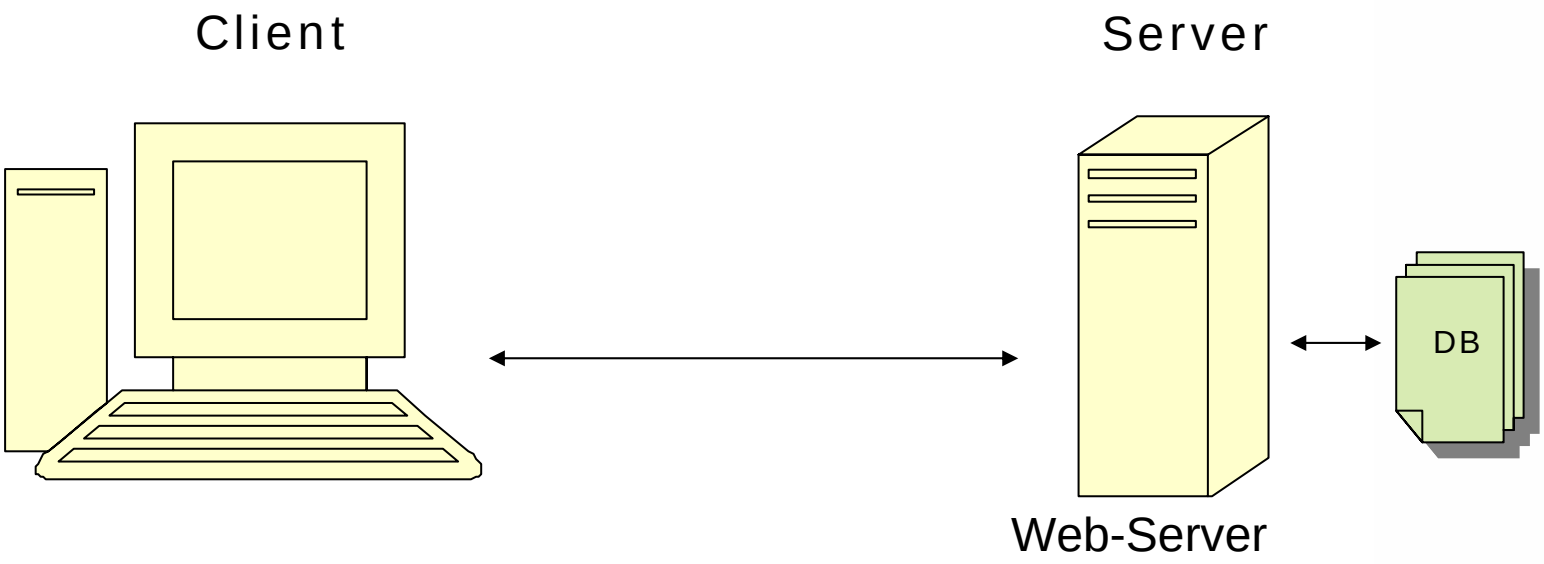
Kapitel 1: Inhalt

Grundlagen

- Aufgaben von Webanwendungen
- Abgrenzung: Statische HTML vs. Dynamische Websites
- Architekturen
- Sicherheit von Webanwendungen

Architekturen

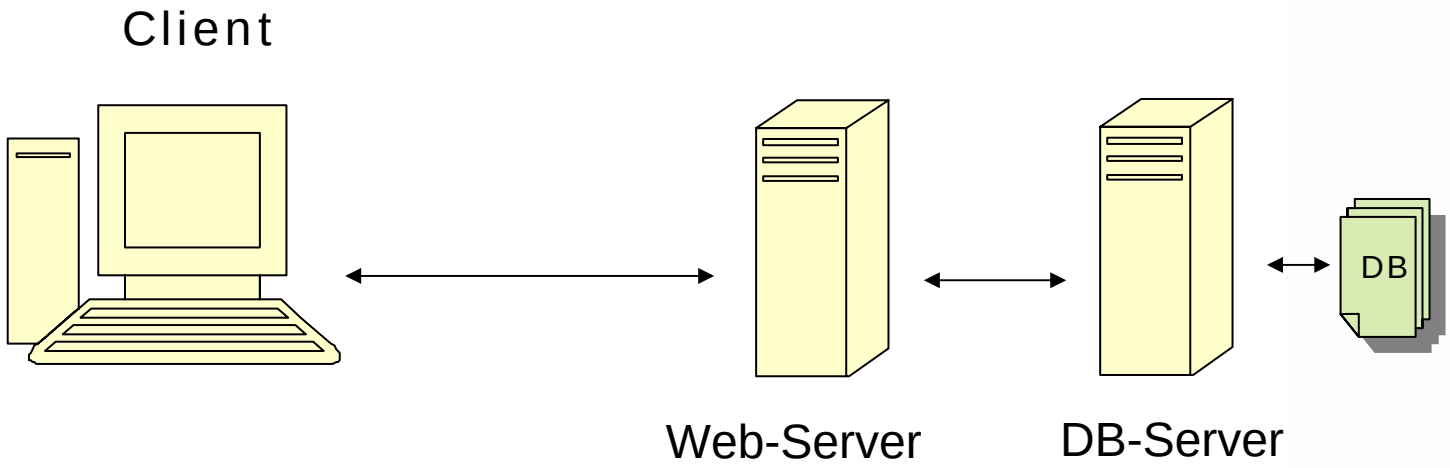
2-Tier



- **Eigenschaften:**
  - Hohe Serverlast
  - Skaliert nicht

Architekturen

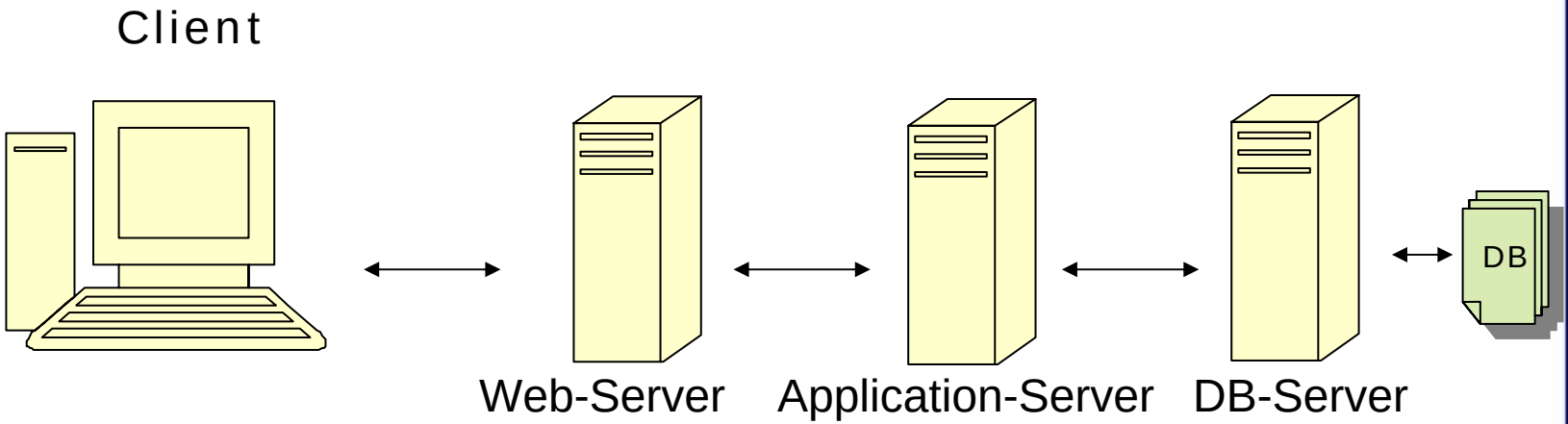
3-Tier



- **Eigenschaften:**
  - Verteilte Last
  - Höhere Skalierbarkeit

Architekturen

n-Tier



- **Eigenschaften:**
  - Höhere Performance
  - Server können für spezielle Aufgaben optimiert werden

Kapitel 1: Inhalt

Grundlagen

- Aufgaben von Webanwendungen
- Abgrenzung: Statische HTML vs. Dynamische Websites
- Architekturen
- Sicherheit von Webanwendungen

Sicherheit von Webanwendungen

Übersicht:

- Anforderungen
- Maßnahmen
- Typische Gefahren

Sicherheit von Webanwendungen

Anforderungen an die Sicherheit:

- Vertraulichkeit
  - Schutz gegen *unautorisierten* Zugriff auf Informationen.
- Integrität
  - Der Inhalt einer Nachricht wird *nicht verändert*.
- Authentizität
  - Der Sender einer Nachricht ist der, der er vorgibt zu sein.
- Verbindlichkeit
  - Der Sender einer Nachricht kann *nicht abstreiten* tatsächlich die Nachricht gesendet zu haben.
- Verfügbarkeit
  - Schutz gegen Beeinträchtigung der Systemfunktion

Sicherheit von Webanwendungen

Maßnahmen zur Erfüllung der Anforderungen

- Vertraulichkeit:
  - Verschlüsselung von Nachrichten
- Integrität
  - Hashwert mit einer Nachricht versenden
- Authentizität
  - Überprüfung von Zertifikaten
- Verbindlichkeit
  - Erzeugung und Überprüfung von digitalen Signaturen
- Verfügbarkeit
  - schwer zu erreichen
  - Replikation, Monitoring, Zugriffskontrollen

Sicherheit von Webanwendungen

Typische Gefahren:

- DoS / DDoS
- Buffer Overflow
- Sniffing & Man-in-the-Middle-Attacken
- Manipulation von Parametern
- Cross-Site-Scripting
- SQL Injection

Kapitel 2: Inhalt

Technologien

- **Ideen:**
  - Trennung von Layout und Inhalt
  - Zusammenfassung der Logik und aller ausführbaren Segmente auf dem Server

- **Serverseitige Technologien:**
  - CGI & Server-API
  - Server-Scripting

- **Clientseitige Technologien:**
  - HTML
  - CSS
  - Java (-Applets)
  - AJAX

Idee: Trennung von Layout und Inhalt

Problem

- Klassische Webseiten:
  - auszuführender Code, Bilder sowie Textpassagen sind in HTML-Seiten zusammengemischt
- Anpassung des Webdesigns an neue Anforderungen (z. B. neues Corporate Design) nach 3 oder 5 Jahren problematisch

Idee: Trennung von Layout und Inhalt

Lösung:

- **Trennung von Layout und Inhalt**
  - Separate Speicherung einzelner Bestandteile (Texte, Bilder oder andere Multimedia-Typen)
  - Zentrale Datenspeicherung:
    - z.B. in einer Datenbank
    - Inhalte bleiben erhalten

Idee: Trennung von Layout und Inhalt

Umsetzung

- Technische Umsetzung:
  - Templates (Vorlagen) = vordefinierte Grundgerüste für die Darstellung von Inhalten.
- Beim Aufruf eines Dokuments vom Server:
  - Zusammenführung der Inhalte aus verschiedensten Quellen (z.B. Datenbanken) zu einem gemeinsamen Format (z.B. HTML).

Idee: Trennung von Layout und Inhalt

Vorteile (1/2)

- Problemlose Einbindung in:
  - neue Strukturen
  - neues Layout
- Erheblich kostengünstiger als eine komplette Neuprogrammierung
- Große Zeitersparnis bei der Weiterentwicklung
- Einfachere Lokalisierung der Webanwendung

Idee: Trennung von Layout und Inhalt

Vorteile (2/2)

- Ablegen der Inhalte in einer Datenbank
  - Inhalte bleiben erhalten
  - Mehrfache Verwendung der Inhalte
  - Speicherung verschiedener Versionen der Inhalte
    - Je nach Datenstruktur: späteres Zurückkehren zu einer bestimmten Version möglich (siehe CMS)

- Mitarbeiter können entsprechend ihren Kernkompetenzen in den Prozess der Inhaltserstellung und -gestaltung integriert werden:
  - Autoren arbeiten an Inhalten
  - Grafiker erstellen die Vorlagen
  - usw.

Idee: Trennung von Layout und Inhalt

Nachteile

- Zu Beginn höherer Planungsaufwand (für Daten- und Anwendungsstruktur)
- Zu Beginn komplexere Entwicklung
- Gesteigerte Serverbelastung, da jede Seite vom Server interpretiert werden muss

Kapitel 2: Inhalt

Technologien

- **Ideen:**
  - Trennung von Layout und Inhalt
  - **Zusammenfassung der Logik und aller ausführbaren Segmente auf dem Server**
- **Serverseitige Technologien:**
  - CGI & Server-API
  - Server-Scripting
- **Clientseitige Technologien:**
  - HTML
  - CSS
  - Java (-Applets)
  - AJAX

Idee: Logik auf dem Server

Funktionsweise

- Alle Aufgaben zur Erstellung der HTML-Dokumente laufen auf dem Server ab:
  - Durchführung der ausführbaren Bestandteile der Webseite auf dem Server
    - Nur reine HTML-Seiten werden verschickt
  - Ggf. Zusammenführung von Layout und Inhalten (bei Trennung von Layout und Code)

Idee: Logik auf dem Server

Vorteile (1/2)

- Skalierbarkeit auf Serverebene
- Auslegung der (Server-) Hardware auf die Anforderungen der Software
- Keine speziellen Fähigkeiten beim Client (hier: Browser) erforderlich (siehe Javascript und Probleme)
  - **Weniger Inkompatibilitäten**
- Sicherheit des ausgeführten Codes
  - Der Code wird nicht zum Client übertragen
  - Der Code kann nicht während der Übertragung geändert werden
  - Weniger Probleme durch bösartigen Code beim Client.
- **-> Nur die übertragenen Daten sind besonders zu schützen**

Idee: Logik auf dem Server

Vorteile (2/2)

- Kein direkter Zugriff auf Ressourcen (wie Datenbanken) vom Client aus
- Mehr Sicherheit, da:
  - Datenbankzugriffe komplett vom externen Netz getrennt werden können
    -> DB nur erreichbar vom Web- oder Applicationserver
  - Dem Client weder Datenstrukturen noch Zugriffsmuster bekannt sind

- **Aber wie wir später sehen werden, findet zurzeit eine starke Entwicklung in Richtung „Rückverlagerung“ von Code zum Client statt.**

Kapitel 2: Inhalt

Technologien

- **Ideen:**
  - Trennung von Layout und Inhalt
  - Zusammenfassung der Logik und aller ausführbaren Segmente auf dem Server

- **Serverseitige Technologien:**
  - CGI & Server-API
  - Scripting

- **Clientseitige Technologien:**
  - HTML
  - CSS
  - Java (-Applets)
  - AJAX

CGI – Common Gateway Interface

Eigenschaften:

- Stellt externer Software eine Laufzeitumgebung zur Verfügung
- Startet neuen Betriebssystemprozess bei jeder Anfrage
  -> begrenzte Skalierbarkeit

Vorteil:

- Einfaches Mittel um dynamische Webinhalte zu erzeugen
- Sprachunabhängigkeit

Nachteil:

- Begrenzte Skalierbarkeit

Auf hochfrequentierten Seiten wird CGI heutzutage nicht mehr so oft eingesetzt.

Server API

Hier wird *beispielhaft* die ISAPI (Internet Server API) vorgestellt

- Programmierschnittstelle (API) zur Funktionserweiterung des „Microsoft Internet Information Services“ Server
- Für den Apache Server gibt es mittlerweile das „*mod_isapi*“
- Schneller und Speicherschonender als CGI

Kapitel 2: Inhalt

Technologien

- **Ideen:**
  - Trennung von Layout und Inhalt
  - Zusammenfassung der Logik und aller ausführbaren Segmente auf dem Server
- **Serverseitige Technologien:**
  - CGI & Server-API
  - **Scripting**
- **Clientseitige Technologien:**
  - HTML
  - CSS
  - Java (-Applets)
  - AJAX

Server–Scripting

Allgemein

- Serverseitig interpretierte Sprachen
  - Quelltext wird an Interpreter auf dem Webserver geschickt
  - Dessen Ausgabe wird an den Browser geschickt. (Meist ein HTML-Dokument)
- Durch eine Schnittstelle (z.B. ISAPI oder CGI) wird der Interpreter ausgeführt.

Server–Scripting

Hier beispielhaft erwähnte Server-Scriptsprachen:

- JSP – JavaServer Pages
- PHP
- ASP – Active Server Pages
- ASP.NET
  + Praxisbeispiel

Server–Scripting: JSP

JSP – Java Server Pages

- Die JSP-Syntax erlaubt:
  - Einbindung von Funktionalitäten in speziellen XML-Tags
  - Implementierung der Funktionalitäten in „Java-Code“ oder „JSP-Aktionen“
- Startet neuen Betriebssystemprozess bei jeder Anfrage
  -> begrenzte Skalierbarkeit

Server–Scripting: PHP

PHP - Hypertext Processor:

- Open-Source-Software
- Interpreter als Modul für alle gängigen Webserver verfügbar
- Weite Verbreitung durch:
  - Leichte Erlernbarkeit
  - Breite Datenbankunterstützung
  - Verfügbarkeit vieler zusätzlicher Funktionsbibliotheken

Server–Scripting: ASP

ASP - Active Server Pages

- Von Microsoft entwickelt
- Scripte werden in HTML-Seiten eingebettet
- Gute Performance und Skalierbarkeit
- Ursprünglich nur vom Webserver „Internet Information Services“ (IIS) interpretiert
- Wird von Microsoft nicht mehr weiterentwickelt.
- Die Nachfolgetechnologie ASP.NET hat ASP abgelöst.

Server–Scripting: ASP.NET

ASP.NET (Active Server Pages .NET)

- Technologie zum Erstellen von Webanwendungen auf Basis des Microsoft .NET-Frameworks
- Webanwendungen können in beliebigen von .NET unterstützten Sprachen erstellt werden
  (z. B.: C#, VB.NET, J# oder Managed C++)

Server–Scripting: ASP.NET

Unterschiede zu ASP (1/2)

- Das ***Code-Behind***-Konzept:
  - Jeder Web-Datei wird eine Klasse zugeordnet, von der die Seite erbt.
  - Vollständige Trennung von Programm-Code und HTML-Layout
  - Steigerung der Übersichtlichkeit
  - Entwicklung strukturierten Programmcodes
    (-> *wartbarer Code*)

Server–Scripting: ASP.NET

Unterschiede zu ASP (2/2)

- Konzept der ***Web-Controls**:*
  - Reduzierung das Codes durch Verwendung vordefinierter Web-Controls
  - Kapselung der Anwendungslogik durch *benutzerdefinierte Web-Controls*
  - Auch der Code der Web-Controls wird üblicherweise in Code-Behind-Dateien gespeichert
  - **Hoher Grad der Wiederverwendung** möglich

Server–Scripting: ASP.NET

Vorteile:

- Siehe Code-Behind und Web-Controls

Nachteile:

- Man ist de facto an einen unter einem Microsoft Betriebssystem laufenden Server gebunden.

- **-> Anhand von ASP.NET schauen wir uns jetzt beispielhaft eine Möglichkeit der praktischen Implementierung einmal an.**

Kapitel 2: Inhalt

Technologien

- **Ideen:**
  - Trennung von Layout und Inhalt
  - Zusammenfassung der Logik und aller ausführbaren Segmente auf dem Server
- **Serverseitige Technologien:**
  - CGI & Server-API
  - Scripting
- **Clientseitige Technologien:**
  - HTML
  - CSS
  - Java (-Applets)
  - AJAX

Clientseitige Technologien

HTML

- Sprache zur Beschreibung textorientierter Dokumente
- Keine Programmiersprache sondern „Strukturierungssprache“
- Enthält Metainformationen
- HTML selbst erfordert nur geringe Bandbreite

Clientseitige Technologien

Grenzen von HTML:

- Alles was über die Dokumentendarstellung hinaus geht muss über andere Sprachen erreicht werden.
  - Keine Darstellung von Videos
  - Bei Bildern wird nur GIF unterstützt
- Andere Sprachen können sein:
  - XML
  - CGI (Formularverarbeitung)
  - JavaScript
  - Java-Applets
  - Weitere Plug-Ins…

Clientseitige Technologien

CSS (Cascading Style Sheets)

- Trennung von Layout und Inhalt
- Formatierungssprache für HTML-Dokumente

- Layout leicht modifizierbar
- Kleinere Dateigrößen
- Keine HTML-Layouttabellen
- Barrierefrei
- www.csszengarden.com

Clientseitige Technologien

Java-Applets

- Clientseitiges Pendant zu Servlets
- Wird im Browser ausgeführt
- Browser benötigt Java VM

Clientseitige Technologien

Java-Applets

Vorteile:

- Interaktion ohne ständige Datenübertragung
- Ideal für komplexe Anwendungen

Nachteile:

- z.T. lange Initialisierungszeit der JVM
- Applet-Inhalte werden in Suchmaschinen nicht erfasst

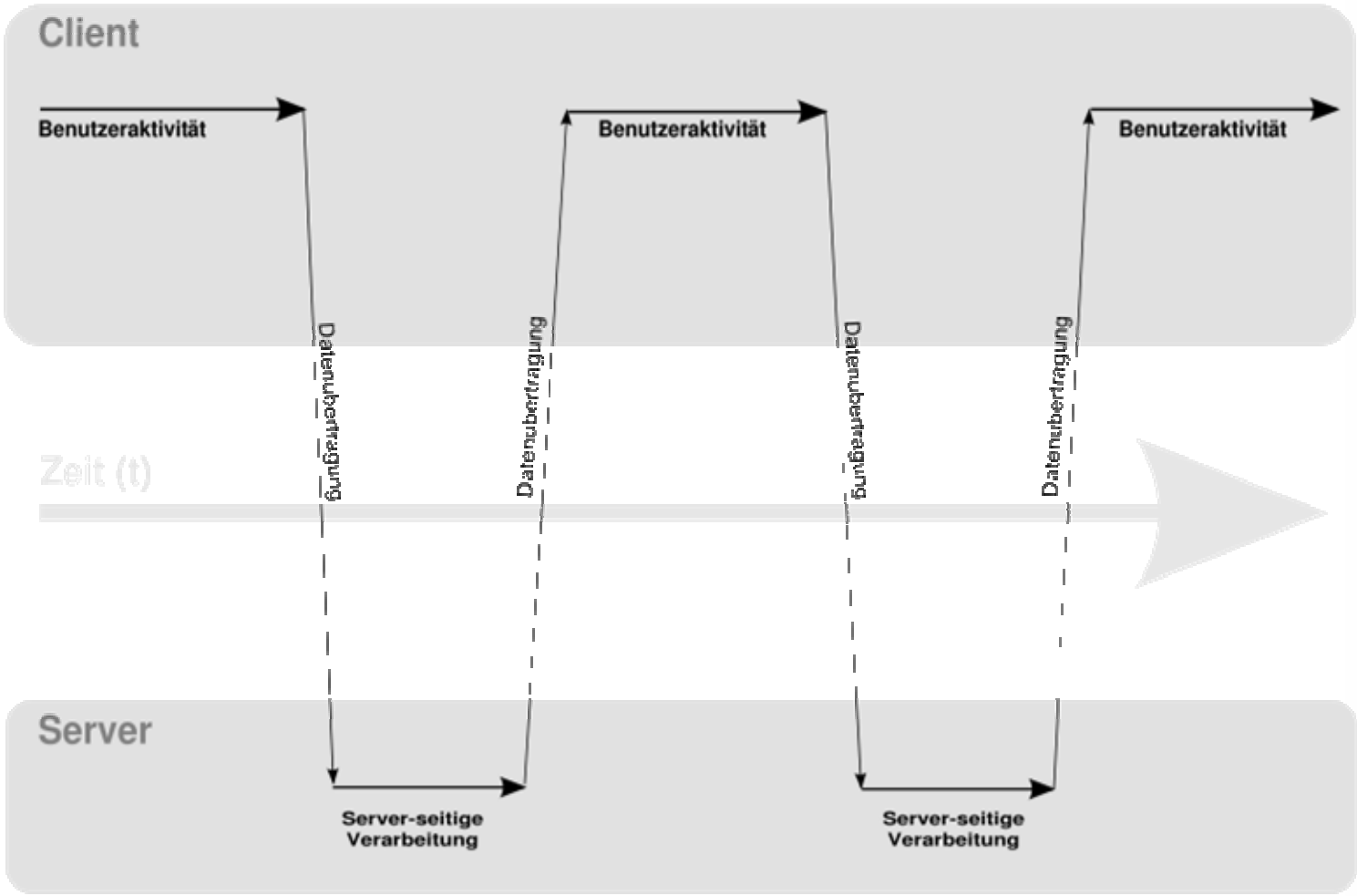
Clientseitige Technologien

AJAX (Asynchronous JavaScript and XML)

- Asynchrone Dateiübertragung
- Schlüsseltechnik zum Web2.0
- Nur bestimmte Teile einer Seite werden nachgeladen
- Voraussetzung:
  - JavaScript ist auf dem Client-Browser aktiviert

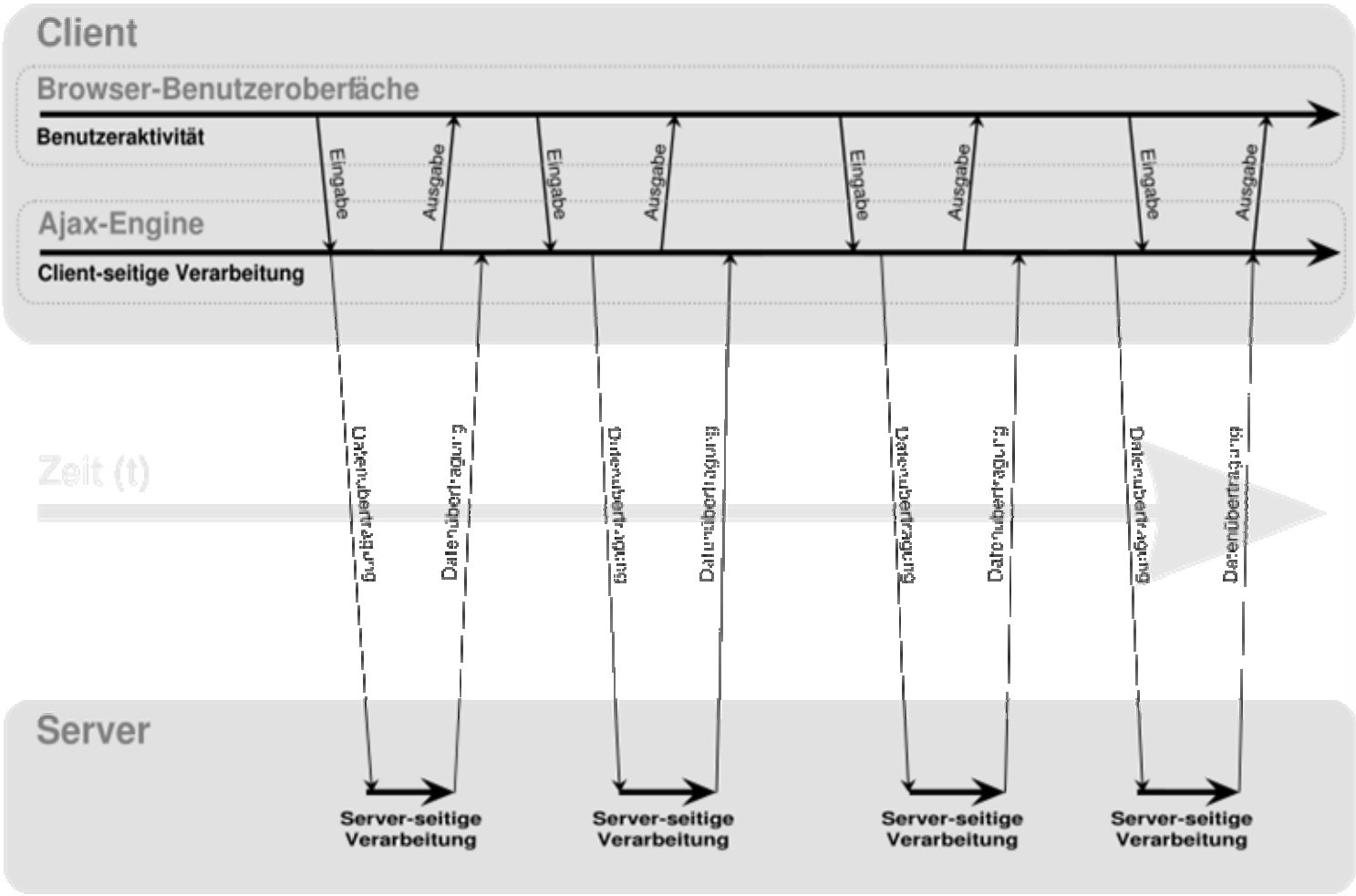
Clientseitige Technologien

Klassisches Modell einer Web-Anwendung (synchrone Datenübertragung)



Clientseitige Technologien

Ajax Modell einer Web-Anwendung (asynchrone Datenübertragung)



Clientseitige Technologien

AJAX

Vorteile:

- Schnelle Reaktion auf Benutzereingaben
- Nur „neue“ Daten werden geladen
- Benötigt kein Plugin

Nachteile:

- Funktion der „Zurück“- Taste eingeschränkt
- Keine Lesezeichen
- Polling

Kapitel 3: Inhalt

Aktueller Stand und Entwicklung

- Rückverlagerung von Logik zum Client
- Probleme beim Aufbau von Webanwendungen
- SOA – Service Oriented Architecture

Rückverlagerung von Logik zum Client

Rückverlagerung findet statt *weil*:

- Nutzer wollen komfortable Webanwendungen

- Webanwendungen sollen:
  - sich wie jedes andere lokale Computerprogramm bedienen lassen
  - schön/ansprechend aussehen und Animationen besitzen
  - multimediale Inhalte anzeigen können

Rückverlagerung von Logik zum Client

Probleme:

- Die Hardware des Client ist unbekannt:
  - evtl. **Performanceprobleme**
- Die Basissoftware (Browser) beim Client ist unbekannt:
  - **Kompatibilitätsprobleme** (evtl. Unterstützung verschiedener Scriptsprachen deaktiviert)

Rückverlagerung von Logik zum Client

Vorteile:

- Grafisch schönere und animierte Oberflächen möglich
- Bessere Menüführung möglich
- Die Webanwendung „fühlt“ sich wie ein lokal laufendes Programm an
  - Anfragen laufen meist unbemerkt im Hintergrund ab (Siehe asynchrone Datenübertragung und AJAX)
- Einbindung multimedialer Inhalte (z.B. Filme)

Kapitel 3: Inhalt

Aktueller Stand und Entwicklung

- Rückverlagerung von Logik zum Client
- Probleme beim Aufbau von Webanwendungen
- SOA – Service Oriented Architecture

Probleme beim Aufbau

Entwicklungsphase

- Fehlersuche und –behebung Zeitintensiv
- Schwer abschätzbare Maximalbelastung
- Lastverteilung problematisch

Benutzungsphase

- Verschiedene Browser
- Sicherheit, Zuverlässigkeit, Verfügbarkeit
- Wartung großer/verteilter Webanwendungen

Kapitel 3: Inhalt

Aktueller Stand und Entwicklung

- Rückverlagerung von Logik zum Client
- Probleme beim Aufbau von Webanwendungen
- SOA – Service Oriented Architecture

SOA

Serviceorientierte Architektur

- Derzeit stark nachgefragt
- Großes Potenzial
- Einzelne, lose gekoppelte Prozesse werden in gekapselten Modulen („Black Box“) realisiert
- Flexibel
- Wiederverwendbar (Kosten)
- An Geschäftsprozesse anpassbar

SOA

Serviceorientierte Architektur

- Schrittweiser Aufbau komplexer AWS
- Überschaubare Logik der einzelnen Dienste
- Programmlogik kann verteilt sein
- Leichtere Preisübersicht

- **Vielen dank für Ihre**
- **Aufmerksamkeit!**