Vorlesungsreihe
Entwicklung webbasierter Anwendungen

Client-Server-Webanwendungen

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

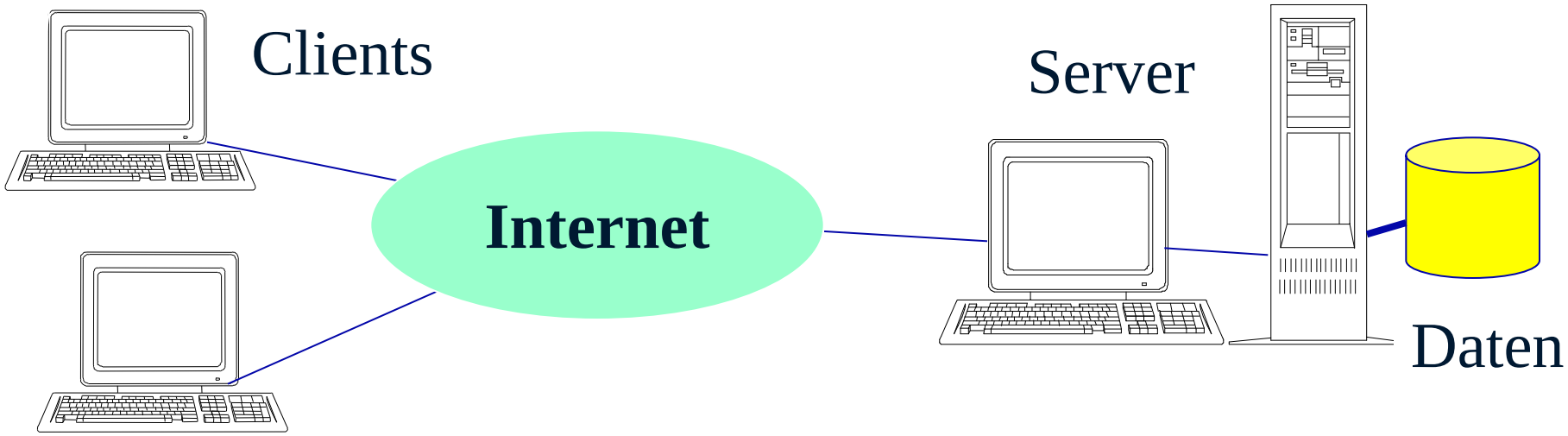
HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Gliederung

- Komplexe Client-Server-Applikationen
  - Allgemeine Optionen zur Webserveranbindung
  - CGI-Anwendungen
  - DLL´s und spezielle Server-Plugins
  - direkter Datenbankanschluß oder spezielle Applikationsserver
- Optionen der Datenverwaltung
- Arten der HTML-Generierung
- gegenwärtig verfügbare Skriptsprachen

Konfiguration komplexer Client-Server-Applikationen

- Daten befinden sich auf dem Server, räumlich entfernt von den Clients



Zwei Grundkonzepte bezüglich der Konfiguration komplexerer Anwendungen

Die Anwendung erfordert neben den Webseiten entsprechende **Anwendungslogik:**

1. Konzentration der gesamten Anwendungslogik auf Server, Clientrechner realisiert nur sehr einfache Anzeige und Visualisierungsfunktionen **(Thin-Client-Konzept)**
2. Anwendungslogik wird auf dem Client realisiert **(Fat Client)**, Server stellt nur Daten(bank)zugriff bereit
3. Mischformen sind möglich und sinnvoll – einfache Logik zwecks Plausibilitäts- und Fehlerkontrolle auf Client, komplexe Datenbankaktionen auf Server

CGI – Anbindung

Grundkonzept : Common Gateway Interface

- durch HTTP-Server wird ein externes Programm gestartet
- das Programm übernimmt die Requestparameter vom Server, führt die Datenbankanfrage aus und liefert HTML-Ergebnisdatei zurück an Server
- bei ersten CGI-Versionen erfolgte der gesamte Datenaustausch per Dateien
- bei verbesserten Versionen wie FastCGI wird über Systemvariable ausgetauscht

Vorteile von CGI:

- relativ robuste Trennung von Server und Verarbeitungsprogramm
- Fehler des CGI-Programms beeinflussen den Server kaum
- jede beliebige Programmiersprache einsetzbar (C, C++, VB, JAVA)

Probleme beim Einsatz von CGI-Programmen

- die mehrstufige Parameterübergabe und das stets notwendige Neustarten des CGI-Programms dauern relativ lange
- die meisten Server beschränken aus technischen Gründen die Anzahl gleichzeitig laufender CGI-Programme auf eine relativ geringe Zahl (5 - 10 - 20 ..)
- weitere Anfragen werden entweder abgewiesen oder in eine Warteschlange gestellt (schlechte Reaktionszeit)
- **bei hohem Anfrageaufkommen und komplexen Aufgaben ist die CGI-Schnittstelle damit nicht mehr einsetzbar (veraltet)**

DLL´s oder spezielle Plugins

Grundkonzept

- durch HTTP-Server werden beim Start oder beim ersten Aufruf einer Funktion DLL´s oder andere Plugins (bei Unix-Systemen) eingebunden
- Datenaustausch zwischen Server und DLL erfolgt durch direkten Funktionsaufruf

Vorteile :

- keine Verzögerungen durch Starten, sehr schnell
- jede Programmiersprache einsetzbar mit der Option für DLL-Generierung (C++, ...)
- gut skalierbar durch parallele Ausführung mehrerer DLL-Instanzen

Probleme beim Einsatz von DLL´s

- Fehler in DLL kann Server negativ beeinflussen, da u.U. im gleichen Adreßraum
- Änderungen bei Entwicklung erfordern Neustart des Servers oder spezielle Funktionen zum Entladen der DLL
- Debugging von DLL´s relativ kompliziert

Bewertung:

- **für professionelle Anwendungen sehr gut geeignet**
- **Entwicklung aber deutlich aufwändiger als alle anderen Verfahren**

Skriptlösungen

Grundkonzept

- Def. Skriptlösung: Anwendungslogik wird als Sourcecode in Textdateien abgelegt
- Skripte werden entweder einmalig beim ersten Aufruf und bei jedem Aufruf durch Server über einen Interpreter zur Ausführung gebracht
- Datenaustausch zwischen Skript und http-Server erfolgt über Dateien oder Systemvariable
- spezielle Servererweiterungen NSAPI für Netscape Server) Serverside API, und ISAPI für Microsoft Information Server (zurückgehender Anteil)

Vorteile :

- durch direkte Ausführung des Quellcodes (kein Compilieren oder Entladen notwendig) sind schnelle Entwicklungszyklen möglich
- Sourcecodes können auch per FTP auf entfernte Server geladen werden

Probleme beim Einsatz von Skriptlösungen

- bei reiner Interpretation meist Performanceprobleme
- bessere Leistung durch vorkompilierte Skripte (wie Java-Bytecode)

Bewertung:

- **für semiprofessionelle Anwendungen gut geeignet**
- **Optimum aus Entwicklungsaufwand und Leistung**
- **heute am häufigsten eingesetzte Lösung (siehe Vorlesung PHP)**

Optionen der Datenverwaltung bei Webanwendungen

- in proprietären Formaten mit eigenen Dateien
- in Datenbanken mit relationalen oder objektorientierten Schema

Vorteile von Datenbanken :

- im Vergleich zu Dateiablagen mehrfach leistungsfähiger bzgl. Selektion, Einfügen und Manipulieren der Daten
- standardisierte Schnittstellen (ODBC, Datensprache SQL)
- fast alle betriebswirtschaftlichen Anwendungen laufen auf Datenbanken (direkte Einspeisung möglich und erstrebenswert ! -> der Kunde wird zum **Prosument -** - d.h. er produziert seinen Konsum-Verwaltungsprozeß selbst)

Nachteile

- wesentlich aufwendiger (Computerressourcen, notwendige Performance muß ca. 10-fach größer sein bei gleichem Anforderungsvolumen)
- echte Highspeed-Anwendungen laufen auch heute noch auf Dateibasis oder eigenen, speziellen Datenbankformaten

Fazit:

- Datenbankeinsatz sollte heute bei Neuentwicklungen als Standard definiert werden
- sinnvoll sind auch Mischformen: bei echter dynamischer Transaktion Ausführung von Datenbankaktionen mit nachfolgender Aktualisierung statischer Seiten (sinnvoll bei Anzahl_Lesezugriffe >> Anzahl_ Schreibzugriffe (vgl. Ebay !!))

Prinzipielle Datenbankanbindung mit Skriptlösungen

Prinzipieller Ablaufe einer Datenbankabfrage

- **Aufruf einer Skriptseite oder HTML-Seite mit spez. Tags, diese öffnen eine Datenbank oder andere Datenquelle und führen eine Abfrage aus :**

```
....Name_ODBC_Datenquelle=Adressen , Paßwort=....
SQL_Statement=„SELECT Name,Adresse,Umsatz  From Adressen
        Where  Umsatz > %Umsatzabfragewert%“
 <!-KOPF>Name    Adresse    Umsatz     <!-/KOPF>
 <!-DETAILS><td>#Name# </td><td>#Adresse# </td><td> #Umsatz#
</td>...
```

- **Das Skript ersetzt die %..%-Platzhalter durch die Anfrageparameter und führt diese Abfrage auf der Datenbank aus.**

```
SELECT Name1, Adresse, Umsatz From Adressen Where  Umsatz>10000
```

- **bindet die DB-Daten statt der Platzhalter in eine HTML-Tabelle mit N-Zeilen ein**

N* `... <td>Lehman KG  </td><td> 52070 Aachen </td><td> 150.000 DM</td>`

- **Anzeige im Browser**

Name	Adresse	Umsatz
Meier GmbH	10132 Berlin	110.000 DM
Lehman KG	52070 Aachen	200.000 DM
.....		

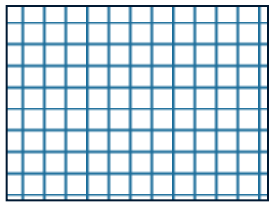
HTML-Generierung aus Datenbankdaten

Zwei Arten der HTML-Erzeugung :

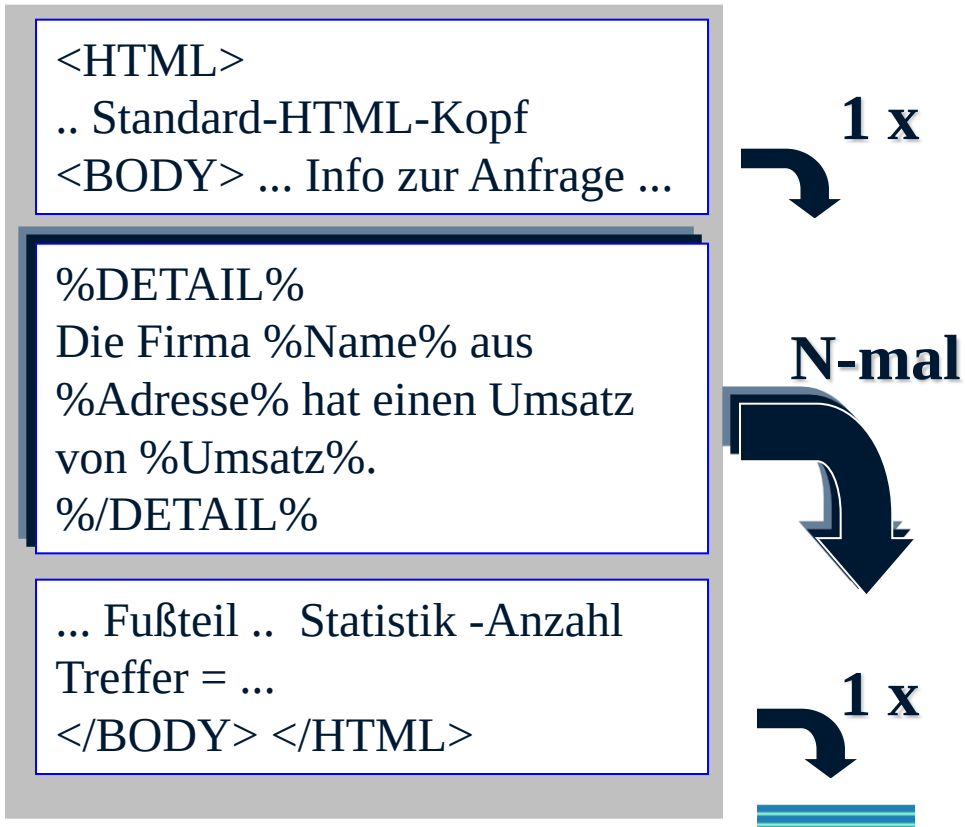
a.) innerhalb einer HTML-Datei (=Template / Schablonendatei) werden Platzhalter definiert, welche später durch DB-Schnittstelle durch Daten ersetzt werden

b.) durch Programmcode wird ein HTML-Texte erzeugt

Typ A günstig für stark layoutorientierte Ausgaben, da Einsatz von HTML-Tools möglich

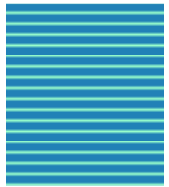


Rohdaten aus der Datenbank



Typ B sinnvoll für sehr komplexe Programmieraufgaben, welche in das relativ starre Templatekonzept nicht integrierbar sind

HTML-Ergebnisdatei (auch mit weiteren Links oder Verzweigungen)



Verfügbare Skriptsprachen

- **PHP -** schnell einsetzbare und sehr leistungsfähige Skriptsprache (-> VL)
- **JAVA :** für größere Projekte – große Bandbreite von Client-Programmierung (Applets – Javascript – Ajax) bis zur Enterprise-Applikationsprogrammierung
- **ASP .NET :** sind an MS-Technologien (IIS, Windows) gebunden, bei Nutzung von spezifischem MS-Code (.NET) notwendig !
- **PERL :** relativ komplex, da für allgemeine Programmieraufgaben entwickelt, (PHP entstand ursprünglich als eine Sammlung von PERL-Skripten) Schwerpunkt auf größere Textmanipulationen und Batch-Verarbeitung in UNIX
- **Cold FUSION** : - Middleware mit eigener Markup-Lang. (CFML), gegenwärtig durch Adobe (nach Allaire und Macromedia) in Entwicklung (kein Opensource)
  Ruby - interpretierte, objektorientierte Programmiersprache mit mehreren Programmierparadigmen, bis 2000 nur im asiatischen Raum, gegenwärtig starke Verbreitung auch in Europa/USA
- **Python** – Interpreter auf C-Basis für die objektorientierte, aspektorientierte und funktionale Programmierung (auch als universelle Programmiersprache ->100$PC)
- Weitere Optionen : Grails (Ruby- unter Java), **JavaScript mit NodeJS**