HTTP-Server

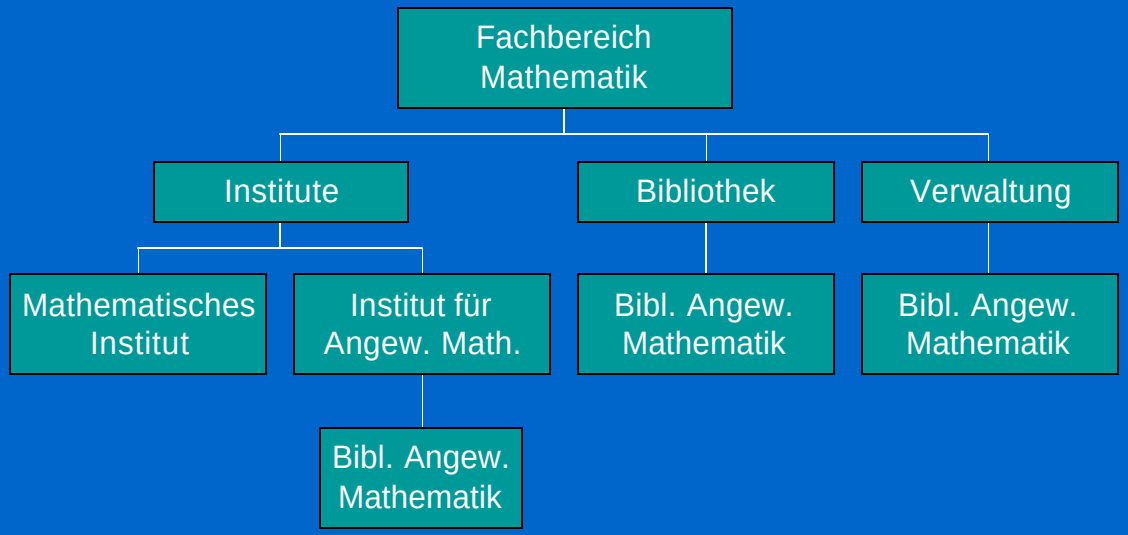
Frank Wübbeling
Universität Münster

Aufgaben des Servers

- Input/Output (Port eröffnen etc.)
- Aufnehmen der Anforderungen
- Bereitstellung des Inhalts
- Bereitstellung der Header
- Erzeugung von Hintergrundinfos (Logfiles)
- Fehlermanagement

Datenstruktur

Informations"baum" des Fachbereichs Mathematik



Datenstruktur

- Attributierte Baumstruktur
- Verbindungen zwischen Elementen möglich
- Realisierung durch einen Dateibaum
- Endknoten sind Dateien
- Zwischenknoten sind Verzeichnisse
- Problem: Verzeichnisse haben keinen Inhalt
- Lösung: spezielle Dateinamen für Verzeichnisse
- Alternative: Datenbanken

Realisierung im Server

- Konfigurationskommando: DocumentRoot
- Externe Verknüpfungen und Kreuzungen mit Symlinks oder Redirects
- Attribute, die nicht durch den Inhalt beschrieben werden, müssen extern angegeben werden:
- Nutzergruppe, Sprache, Passwoerter...

Konfigurationsparameter

- Interne Parameter (Arbeitsweise, Module...)
- Technische Parameter (Timeout...)
- Dateiparameter
- Logfiles
- Virtuelle Hosts

Systemparameter

- Auf welchen Ports wollen wir arbeiten?
- Wie heißt der Rechner?
- Email des Betreibers?
- Wo stehen die Dokumente?

Verzeichnisparameter

- FollowSymlinks
- Indexes
- MultiViews
- Einschränkung des Zugriffs: Allow/Deny
- Aktiver Inhalt (CGI)
- ServerSideIncludes
- Redirect

Beispiel: Passwortzugriff

- Verzeichniseigenschaften in .htaccess
- AuthType Basic
- AuthUserFile /.../
- AuthName „Name“
- Require ValidUser
- Optional: Get/Put einschränken

Beispiel: Virtuelle Hosts

- Für jeden virtuellen Host muß eine eigene DocumentRoot definiert werden.
- Systemparameter etc. sind im Allgemeinen für alle Hosts gleich.

Beispiel: Serverstatus

- Server liefert Informationen über den Status.
- Per WWW!
- Die entsprechende virtuelle Adresse muß gut abgesichert sein.
- Apache: /server-status, /server-info

Beispiel: Auswertungen/Logfiles

- Server liefert Logfiles in vielen Formaten.
- Auswertung geschieht durch externe Programme.
- Privatsphäre muß geschützt bleiben, d.h. Hostnamen dürfen nicht komplett gespeichert werden.

Statischer vs. Dynamischer Inhalt

- Viele (fast alle?) Webseiten haben keinen statischen Inhalt
- Beispiele:
- Webserverstatus
- Datenbankabfrage (Banking, Buchrecherche, persönliche Begrüßung...)

Realisierung von dynamischem Inhalt

- Webseite wird komplett durch Programme erstellt.
  - Shellskripte (batch-Dateien, perl-Dateien)
  - Allgemeine Programmiersprachen (C, Java,...)
  - Dedizierte Programmiersprachen(PHP, VisualBasic)
- Dynamischer Inhalt wird in eine Webseite eingebunden.
  - ServerSideIncludes
  - JSP-Java Server Pages, ASP-Active Server Pages

Probleme

- Resourcenaufwendig.
- Für jede zu holende Datei muß ein Programm gestartet werden.
- Wie kommunizieren Server und externe Programme? (Beispiel: Formulare!)
- Große Sicherheitsprobleme.

Konventioneller Ansatz (CGI)

- Die Dateien in einem Verzeichnis werden als ausführbare Programme gekennzeichnet.
- Wird eine Datei angefordert, wird sie ausgeführt und der Output gesendet.
- Variable werden an die Programme als Umgebungsvariable mitgegeben oder per I/O.
- Beispiel: printenv

Server Side Includes

- Eingebettet in den normalen Text (HTML).
- Spezielle Markierungen werden vom Server interpretiert.
- Vorteil: Übersichtlich, kann von Standard-HTML-Gestaltungsprogrammen erzeugt werden.
- Resourcenschonend.

SSI Kommandos

- Include (Füge den Inhalt einer Datei ein)
- Exec (Führe ein Kommando aus)
- Füge Variable ein.

SSI Wertung

- Gute Idee, aber zu wenig Möglichkeiten, um Inhalt zu erzeugen.
- Man sollte die Möglichkeit haben, kompletten Programmcode in HTML Dateien durch den Server ausführen zu lassen.
- Geeignet: JAVA, VisualBasic, PHP.