CGI – Common Gateway Interface

„Web–Based Programming“

Wintersemester 1998/99

Lars Oergel

Thorsten Schultzke

Einleitung

- „CGI“: Common Gateway Interface
- standardisiertes Interface zwischen Anwendungen und HTTP–Servern
- „normale“ HTML–Seiten haben statischen Charakter
- CGI erlaubt dynamisch erstellte Dokumente, d.h. echte Interaktion
- Ablauf: Eingabe vom HTTP–Server, Verarbeitung, Ausgabe an Client
- Anwendungsbeispiele:
  - Suchmaschinen/Datenbankzugriff
  - E-Commerce–Anwendungen
  - Gästebücher
  - Zugriffszähler

Aufruf von CGI–Programmen

- Links, Grafikverweise, Server Side Includes
- automatisches Laden mittels „Meta“–Statement
- Datenübermittlungsmethoden: GET, POST
- HTML–Formulare
  - Versand der Formulardaten per E-Mail möglich (MAILTO)
  - ein- und mehrzeilige Textfelder, Paßwortfelder (nur Sichtschutz)
  - Auswahllisten mit Einfach- und Mehrfachauswahl
  - Radio–Buttons, Check–Boxen
  - Buttons zum Absenden und Abbrechen
  - „intelligente“ Formulare mittels JavaScript realisierbar

Allgemeines zur Eingabe

- Anforderungsmethoden:
  1. GET
     - ISINDEX – Abfrage
     - Namens-/Wertepaare (Formularfeldname, Inhalt)
  2. POST
     - Namens-/Wertepaare (Formularfeldname, Inhalt)
- optional: „Path Info“ angehängt an Skriptname in URL
- Kodierung der Daten: „URL Encoding“ (nicht eindeutig!)

Die GET–Methode

- Übertragung der Eingabedaten im HTTP–Header: (URL)„?“(Daten)

⇒ Längenbeschränkung!

⇒ kann als Bookmark gespeichert werden

⇒ kann von Proxies abgefangen werden

- Zugriff über $QUERY_STRING
- ISINDEX – Abfrage:
  - Kennzeichen: keine Wertepaare, d.h. kein unkodiertes „=“
  - Übergabe zusätzlich als Kommandozeilenparameter
  - Auflösung von „+“ ⇒ mehrere Parameter möglich

Die POST–Methode

- Übertragung der Eingabedaten hinter dem HTTP–Header (im Body)

$\Rightarrow$ keine Längenbeschränkung

$\Rightarrow$ nicht als Bookmark speicherbar

$\Rightarrow$ nicht proxyfähig

- Zugriff über Standardeingabe und $CONTENT_LENGTH
- Achtung, kein Verlaß auf EOF!

Umgebungsvariablen (Auswahl)

- allgemein
  - $SERVER_SOFTWARE, $SERVER_NAME
- anforderungsspezifisch
  - $SERVER_PROTOCOL, $SERVER_PORT
  - $REQUEST_METHOD („GET“/„POST“)
  - $CONTENT_TYPE (z.B. „application/x-www-form-urlencoded“)
  - $PATH_INFO, $PATH_TRANSLATED
  - $QUERY_STRING, $CONTENT_LENGTH
  - $REMOTE_HOST, $REMOTE_ADDRESS
  - $HTTP_XXX – HTTP-Header

Realisierung/Implementierung

- jede Sprache möglich, für die Compiler/Interpreter auf Server–BS existiert
- Auswahlkriterien für die Programmier-/Skriptsprache:
  - Performance
  - Sicherheit
  - Zuverlässigkeit
  - Pflegefreundlichkeit
  - Portabilität

  ⇒ Sprache, in der man sich „wohlfühlt“
- Problem der Nebenläufigkeit (File–Locking)
- „Zustandslosigkeit“ (versteckte Formularfelder, Cookies)

Ausgabe von CGI–Programmen

- Rücklieferung von STDOUT als Ergebnis
- alle Arten von (MIME–)Dokumenten lieferbar, z.B.
  - HTML–Seite, Grafik, Video, Sound
  - Referenz („Location"–Header)
- Ausgabeformat
  - HTTP–Header (Case–sensitiv)
  - eine Leerzeile (auch bei leerem Body)
  - optionaler Inhalt
- Ankündigung des Ausgabeformates per HTTP–Header „Content-type" (z.B. „text/html" oder „image/jpeg")

Sicherheit

- wichtig, da CGI anonymen Benutzern Programmstart auf Server ermöglicht
- Prinzipien:
  - erwarte keine bestimmte Eingabe/Reaktion vom Benutzer
  - gib nie mehr Informationen als nötig heraus
- Maßnahmen:
  - CGI–Skripte nur in einem bestimmten Verzeichnis halten (Web–Server–Konfiguration!)
  - Vorsicht bei Ausführung (auch gefilterter) Benutzereingaben via „exec“, „system“ oder „popen“
  - Server Side Includes für das cgi–bin Verzeichnis deaktivieren

⇒ Backup als Selbstverständlichkeit

Server Side Includes

- Problem: normale CGI–Skripte müssen gesamte HTML–Seite erzeugen
- Lösung: Ersetzen einzelner Statements innerhalb einer Seite durch Server
  - allgemeine Syntax: <!--#command tag="argument" -->
  - „include file“ – einfaches Einsetzen der genannten Datei
  - „echo var“ – Ausgabe von Environment–Variablen
  - „fsize file“ – ergibt die Filegröße der Datei
  - „flastmod file“ – Datum der letzten Änderung der Datei
  - „exec cgi/cmd“ – Ausführen von CGI- oder normalen Programmen
- Behandlung muß explizit im Web–Server konfiguriert sein
- Namenskonvention: *.shtml, *.sht oder *.shtm

Setzen und Lesen von Cookies

- Problem: Zustandslosigkeit
- Lösungsmöglichkeit: HTTP–Cookies
- Speicherung von Informationen auf Client–Seite
- Cookie: Domain, Pfad, Lebenszeit, Wert, „secure“
- Setzen über HTTP–Header:
  `Set-cookie: name=wert; domain=tu-berlin.de; path=/;`
  `expires=Sun, 31-Jan-99 23:59:00 GMT`
- Auslesen mittels Umgebungsvariable $HTTP_COOKIE
  (Inhalt z.B. „name1=wert1;name2=wert2“)

Zusammenfassung: Vor-/Nachteile

- Vorteile:
  - echte Interaktion realisierbar
- Nachteile:
  - Kodierung mittels „URL–Encoding“ nicht eindeutig
  - Wartezeiten bei Netz-, Prozessor-, Datenbanküberlastung möglich
  - Sicherheitsrisiken