Grundlagen der CGI-Programmierung

Martin Vorländer

PDV-Systeme GmbH, Bornhardtstr. 3, 38644 Goslar
Telefon 05321-3703-0, Fax 05321-8924
info@pdv-systeme.de, www.pdv-systeme.de

Was ist CGI?

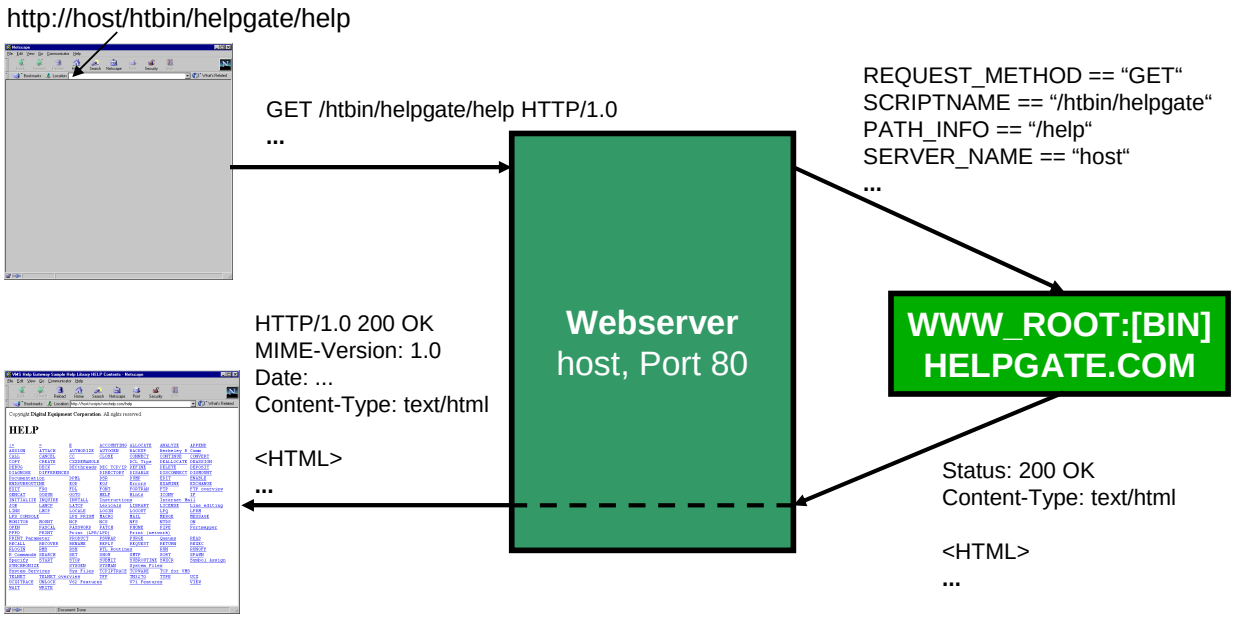
„**C**ommon **G**ateway **I**nterface“

Protokoll für die Kommunikation zwischen Webserver und externen Programmen

- dynamische Erzeugung von Webserver-Ausgaben z.B. HTML, Bilder, PDF, ...
- Verarbeitung von Formularen
- Gästebücher, Besucher-Zähler, Chatrooms, ...

entwickelt für den „Ur-Webserver“ NCSA HTTPd

lange Zeit ein de-facto Standard ohne RFC
erst im Oktober 2004 als RFC 3875 standardisiert



Webserver „weiß" durch Konfiguration, welches Verzeichnis CGI-Skripte enthält, und wie diese zu behandeln sind.

Datenübergabe durch:

- ↘ „Meta-Variablen" (i.d.R. Environment-Variablen)
- ↘ Standard Input
- ↘ Kommandozeile (nur in Spezialfällen)
- ↖ Standard Output

⇒ als Programmiersprachen eignen sich sowohl Compilersprachen (z.B. C, C++, Pascal) als auch Interpreter (z.B. Perl, Python, Tcl)

URIs *(Uniform Resource Identifiers)*

auch URLs (Uniform Resource Locators) genannt

Format einer absoluten HTTP-URI:

```
http://host[:port]/[path][?query][#fragment]
```

- Default für `port`: 80
- `path`: hierarchischer Pfad zur Ressource
  - Pfadtrenner ist /
  - muß kein physikalisch vorhandener Pfad sein!
- Format für `query`:
  - `name=wert&name=wert...`
  - `keyword+keyword...`
- `fragment`: Sprungziel innerhalb einer HTML-Resource

URI-Encoding

nicht-reservierte Zeichen:

```
a-z A-Z 0-9 - _ . ! ~ * ' ( )
```

reservierte Zeichen: `; / ? : @ & = + $ ,`

sonstige Zeichen sollten nicht enthalten sein

reservierte (und sonstige) Zeichen müssen kodiert werden

Kodierung: % und zwei Hexadezimal-Ziffern

z.B. `%20` für Leerzeichen

Anfrage-/Antwort-Protokoll

Anfrage besteht aus

- Anfrage-Methode, Anfrage-URI, Protokollversion; z.B.

  `GET http://host/htbin/helpgate/help HTTP/1.0`
- MIME-Kopfzeilen mit Zusatz-Information
- evtl. einem Anfrage-Rumpf

Antwort besteht aus

- Protokollversion, Status-Code
- MIME-Kopfzeilen mit Meta- und Zusatz-Information
- evtl. einem Antwort-Rumpf

Problem: RFC 822 definiert Nachrichten-Format nur für 7-Bit-Zeichen und maximal 1000 Zeichen/Zeile

MIME-Kopfzeilen

- `MIME-Version: 1.0`
- `Content-Type: type/subtype [; param=wert]`

  z.B. `text/plain; charset=ISO-8859-1`
  oder `image/gif`
- `Content-Transfer-Encoding`

  Werte: `7bit, 8bit, binary, base64, quoted-printable`
- `Content-ID`
- `Content-Description`

mehrteilige Rümpfe möglich

`Content-Type: multipart/mixed; boundary=`*XYZ*

der Kopf jedes Rumpf-Teils enthält höchstens `Content-*` Kopfzeilen

Einleitung für jeden Rumpf-Teil: `--`*XYZ*

Abschluß des letzten Rumpf-Teil: `--`*XYZ*`--`

REQUEST_METHOD

- GET / POST
  - GET für Anfragen ohne Seiteneffekte (Konvention!)
- HEAD
  - Kopf-Daten der entsprechenden GET-Anfrage (ohne Rumpf)
- PUT
- DELETE
- OPTIONS
- TRACE

Meta-Variablen 2

QUERY_STRING

- Daten hinter dem ? des Anfrage-URI
- genau so, wie die Anfrage gestellt wurde, d.h. URI-kodiert

PATH_INFO

- Daten im Pfad-Teil des Anfrage-URI hinter dem Namen des CGI-Skripts

PATH_TRANSLATED

- Versuchte Übersetzung von PATH_INFO in einen physikalischen Pfad

Meta-Variablen 3

CONTENT_TYPE und CONTENT_LENGTH

- Informationen über Daten im Rumpf der Anfrage

SCRIPT_NAME

- Teil des Anfrage-URI mit URI-Pfad und Name des CGI-Skripts

REMOTE_ADDR und REMOTE_HOST

- REMOTE_HOST i.d.R. nicht gesetzt

REMOTE_USER und AUTH_TYPE

- nur gesetzt bei authentisierten Anfragen

REMOTE_IDENT

Meta-Variablen 4

SERVER_NAME

- Host-Teil des Anfrage-URI

SERVER_PORT

- Port-Teil des Anfrage-URI oder der tatsächliche Port

SERVER_SOFTWARE

SERVER_PROTOCOL

- i.d.R. `HTTP/1.0` oder `HTTP/1.1`

GATEWAY_INTERFACE

- i.d.R. `CGI/1.1`

Meta-Variablen 5

HTTP_*

- zusätzliche HTTP-Kopfzeilen vom Klienten, z.B.
  - HTTP_ACCEPT
  - HTTP_USER_AGENT
  - HTTP_REFERER

je nach Webserver zusätzliche Variablen

- z.B. bei Apache:
  - DOCUMENT_URI
  - DOCUMENT_ROOT
  - FILEPATH_INFO

HTML-Eingabemöglichkeiten

ISINDEX

- HTML-HEAD-Element

FORM

- ACTION
  - URI des aufzurufenden CGI-Skripts
- METHOD
  - GET oder POST
- ENCTYPE
  - Default: `application/x-www-form-urlencoded`
  - Spezialfall: `multipart/form-data`

HTML-Formulare

INPUT

- TYPE
  - TEXT, PASSWORD
  - CHECKBOX, RADIO
  - IMAGE
  - HIDDEN
  - SUBMIT, RESET
  - FILE (Erweiterung zum HTML- Standard)
    - *nur mit METHOD=POST erlaubt!*

TEXTAREA

SELECT / OPTION

Kodierung der Formulardaten 1

bei „normaler“ FORM

- Leerzeichen in Feldnamen und -werten werden durch + ersetzt
- URI-Encoding der Feldnamen und -werte
- Feldnamen und -werte werden mit = zusammengesetzt
- alle solchen Strings werden mit & oder ; zusammengesetzt
- dieser String wird
  - bei METHOD=GET mit ? an die ACTION-URI angehängt
  - bei METHOD=POST als Rumpf verschickt

Kodierung der Formulardaten 2

bei FORM mit INPUT TYPE=FILE

- MIME-Nachricht mit
  - `Content-Type: multipart/form-data; boundary=...`
  - pro Eingabefeld ein Rumpf-Teil mit
    - *`Content-Disposition: form-data; name="name"`*
      `[; filename="filename"]`
    - *`Content-Type: type/subtype`*

bei ISINDEX

- URI-Encoding der Schlüsselwörter
- Schlüsselwörter werden mit + zusammengesetzt
- dieser String wird mit ? an die Basis-URI angehängt

Datenübergabe

bei FORM METHOD=GET

- über QUERY_STRING

bei FORM METHOD=POST

- über Standard Input
- end-of-file kann fehlen!
- Länge in CONTENT_LENGTH

bei ISINDEX

- über QUERY_STRING
- *kann* auch über die Kommandozeile

Ausgabe 1

nach Standard Output

„Parsed Header“-Skripts

- CGI-Kopfzeilen (werden vom Webserver ausgewertet)
  - `Status`
  - `Content-Type`
  - `Location`
    - *dann keine HTTP-Kopfzeilen!*
- HTTP-Kopfzeilen
  - z.B. `Expires`

Ausgabe 2

„Non-Parsed Header“-Skripts (NPH)

- müssen komplette HTTP-Nachricht ausgeben
  - 1. Zeile: HTTP-Statuszeile, z.B. `HTTP/1.0 200 OK`
  - HTTP-Kopfzeilen
    - *`Date`, z.B. `Thu, 30 Mar 2000 08:25:00 GMT`*
    - *`Server`*
- Kennzeichnung für den Webserver ist implementationsabhängig
  - Apache: Dateiname beginnt mit "nph-“
  - Microsoft IIS: Alle Skripte sind NPH-Skripte

Status-Codes

1xx Information

- erst genutzt ab HTTP/1.1

2xx Erfolg

3xx Umleitung

4xx Klienten-Fehler

5xx Server-Fehler

Tipps

Das aktuelle Verzeichnis wird vom Standard nicht festgelegt!

Das Akte-X-Motto: *Trust Noone!* (und schon gar nicht dem Klienten)

kein Ausführen von Programmen mit Daten vom Klienten, ohne daß diese geprüft werden!

- Perl: 'eval'-Gefahren, -T
- C: popen(), system(), Escapen (mit \) von Sonderzeichen

Interpreter darf nicht vom Browser aus direkt erreichbar sein!

Literatur 1

RFCs

822	Text Message Format
1738	URLs
1808	relative URLs
2396	URIs
1945	HTTP/1.0
2616	HTTP/1.1
2045-2048	MIME
1866	HTML 2.0
1867	Form-Based File Upload
3875	CGI 1.1

Literatur 2

RFCs

- ftp://ftp.isi.edu/in-notes/rfc*XXXX*.txt
- http://www.rfc-editor.org/rfc.html
- http://www.faqs.org/rfcs/

CGI Standard

- http://hoohoo.ncsa.uiuc.edu/cgi/
- http://www.w3.org/CGI/

CGI Programming FAQ

- http://www.htmlhelp.org/faq/cgifaq.html

  auch http://www.webthing.com/tutorials/cgifaq.html

Literatur 3

CGI Tutorials

- http://www.selfhtml.org/
- http://wdvl.com/Authoring/CGI/
- news:comp.infosystems.www.authoring.cgi

CGI und Sicherheit

- http://www.w3.org/Security/Faq/wwwsf4.html

Literatur 4

C

- http://www.cs.tut.fi/~jkorpela/forms/cgic.html

Perl

- http://www.stonehenge.com/merlyn/WebTechniques/
- http://search.cpan.org/dist/CGI.pm/
- news:de.comp.lang.perl.cgi

Literatur 5

CGI-Skripte

- http://www.cgi-resources.com/
- http://icthus.net/CGI-City/
- http://www.wyenet.com/cgi_scripts/
- http://nms-cgi.sourceforge.net/

  als (sicherer) Ersatz für http://www.scriptarchive.com/
- http://awsd.com/scripts/

Fragen?