

today:
server side web technology:
CGI and PHP

course: Web Technology

Maarten Lamers

www.maartenlamers.com/WT2006/

Leiden University

client-side versus server-side

give some reasons:

1. for using client-side web applications
2. for using server-side web-application

today

1. Common Gateway Interface (CGI):
2. PHP
3. assignment 2

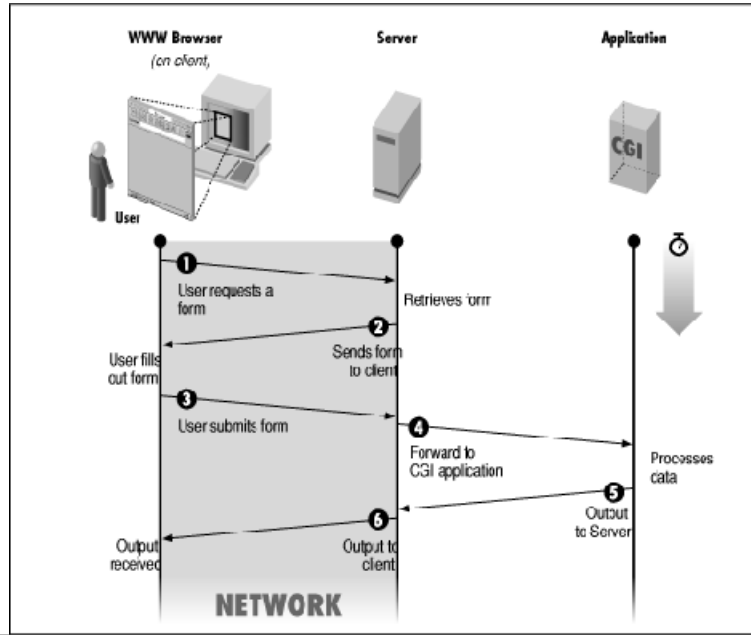
CGI: Common Gateway Interface

most generally:
standard for interfacing information servers (e.g. web-servers)
to external applications

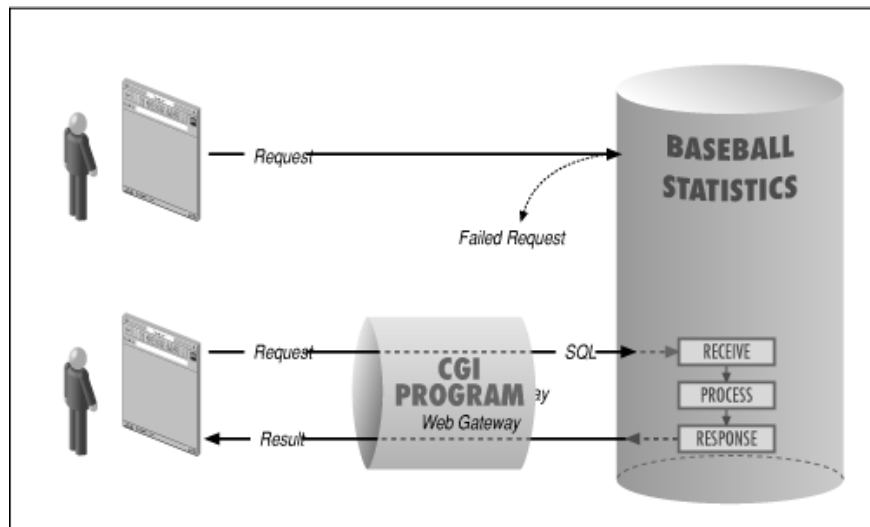
most commonly used as a:
standard way for a web-server for passing request data to an
application program, and to receive data back for the user

the information server acts as a gateway, hence the name
CGI is an interface, not a language!

CGI: example scenario



CGI: example, gateway to database



CGI and the web server

Q: how does a web server know that a URL points to a CGI-based program that must be executed?

A: it can be configured to map certain URLs to CGI-based programs

for example, the `cgi-bin` directory as in
<http://www.mediatech.nl/cgi-bin/hello.pl>

the web server will then recognize the resource as a program, execute it, and pass it the HTTP request data

writing CGI programs

"CGI program" is a confusing term (a.k.a. "CGI script", "CGI application")

it is not a program written in the CGI language

we use it to indicate executable programs written in any language that use the CGI interface for receiving HTTP request data

Perl, C, C++, Java, JavaScript, UNIX sh, ...

usually, a scripting language is used (interpreted, not compiled)

Perl is very popular for CGI scripting, because of its *regular expressions* that make processing of CGI data easier (they are used for handling of text data mostly)

CGI output

how is data retrieved from a CGI program?

1. the CGI program generates output in some form
2. the web-server adds HTTP headers to form a valid HTTP response
3. the result is sent to the client

CGI output

output of CGI programs begins with a small header
in same format as HTTP headers
any headers other than *server directives* are sent directly back
to client

common server directives:

Content-type:	specifies output content type
Location:	redirect to another URL
Status:	indicates HTTP status code

CGI input

how is data passed from a web request to a CGI program?

the server will:

- pass HTTP request headers as environment variables
- pass HTTP request body to the program's standard input
- pass query string part of URL as environment variable and translate it to command line arguments of CGI program call

```
http://www.car.com/products.pl?type=peugeot&year=1991
```

CGI input: environment variables

environment variables specific to the request being fulfilled by the web server:

`REQUEST_METHOD`

how was the request made? For HTTP this is "GET", "POST", "HEAD", ...

`QUERY_STRING`

whatever follows "?" in the URL which referenced the program

`CONTENT_TYPE`, `CONTENT_LENGTH`

for queries with information attached by the client, such as HTTP POST and PUT, this is the content type of the data and its length

CGI input: environment variables

http://www.liacs.nl/~joostd/cgi-bin/envs.cgi?flower=tulp

```
DOCUMENT_ROOT:/home/httpd/wwwhome/cswww
GATEWAY_INTERFACE:CGI/1.1
HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, ...
HTTP_ACCEPT_ENCODING:gzip, deflate
HTTP_ACCEPT_LANGUAGE:en-us
HTTP_CONNECTION:Keep-Alive
HTTP_HOST:www.liacs.nl
HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
PATH:/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin
QUERY_STRING:flower=tulp
REMOTE_ADDR:213.17.59.82
REMOTE_PORT:3039
REQUEST_METHOD:GET
REQUEST_URI:/~joostd/cgi-bin/envs.cgi?flower=tulp
SCRIPT_FILENAME:/home/httpd/wwwhome/joostd/cgi-bin/envs.cgi
SCRIPT_NAME:/~joostd/cgi-bin/envs.cgi
SERVER_ADDR:132.229.44.15
SERVER_ADMIN:www@liacs.nl
SERVER_NAME:www.liacs.nl
SERVER_PORT:80
SERVER_PROTOCOL:HTTP/1.1
SERVER_SOFTWARE:Apache/1.3.12 (Unix) (Red Hat/Linux) PHP/3.0.18
mod_perl/1.23
```

CGI input: passing content

```
POST /icecream.pl HTTP/1.1
...
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

scoops=2&flavors=cherry+vanilla
```

for requests with information attached after the header (e.g. HTTP POST or PUT), the information is sent to the script on *standard input*

standard input is a standard way in UNIX or DOS to pass data to any program. Perl handles it also.

basically, the server sends a series of bytes (`CONTENT_LENGTH`) to the CGI program file

CGI drawbacks

decoding of URLs can be a hassle:
the program must do this to access the individual chunks of data

inefficient:
for every request a new copy of the program is started; this can become a very big load on the server

too low level

solutions:

more efficient CGI (*FastCGI, mod_perl*)

server side "*inside out programming*" methods (*ASP, PHP, ...*)

printing a dynamic date using CGI

```
print("<HTML>");  
print("<H1>Sample</H1>");  
var now = new Date();  
print("It is now <strong>" + now + "</strong>");  
print("</HTML>");
```

HTML is embedded within the script-code

becomes unreadable if there is more HTML than code

printing a dynamic date using PHP

```
<HTML>
<H1>Sample</H1>
It is now <strong>
<?php echo date ?>
</strong>
</HTML>
```

script-code is embedded in HTML instead of other way around!
inside-out programming

remember JavaScript?

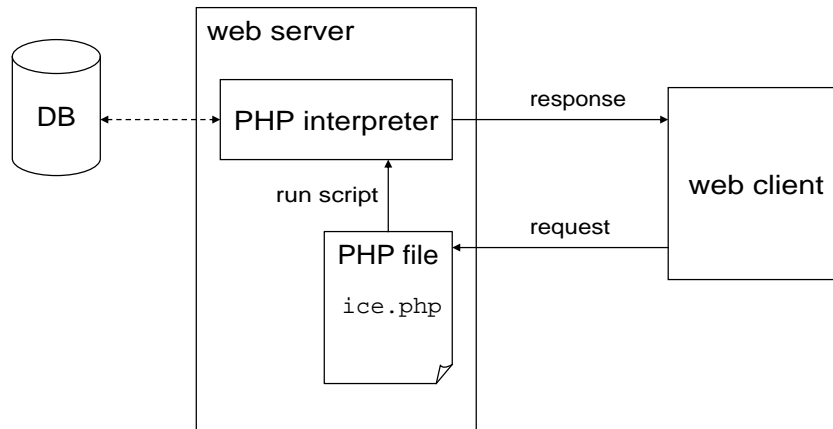
PHP

Personal Home Page Tool
PHP: Hypertext Preprocessor

server-side programming (scripting) language
freely available under open-source license
alternative to Microsoft's Active Server Pages (ASP)

inside-out programming
syntax is a mix of Java, C and Perl languages
files usually have the extension `.php` so that the web server
knows it should be processed by PHP interpreter.

PHP scenario



PHP: code example

```
<HTML>
  <HEAD> ... </HEAD>
  <BODY>
    <H1>Example</H1>
    <?php echo 'Maarten likes <B>kroketten</B>!' ; ?>
    Today is <?php echo date('\l, F dS Y. '); ?>
  </BODY>
</HTML>
```

script is included between `<?php ?>`
echo statement just prints output to HTML document
HTML tags can be in the output
every statement must end with `;` character

`<? ?>` is also allowed

PHP: comments

```
<HTML>
<HEAD> ... </HEAD>
<BODY>
  Dutch lesson number 1:<BR>
  <?php
    /*
     Here is a useful sentence.
    */
    echo '<EM>Twee kroketten, alstublieft!</EM>';
    // It means "Two kroketten, please."
  ?>
</BODY>
</HTML>
```

comment between `/* ... */` or after `//`
note: this is not HTML comment, but PHP comment!

PHP: variables

```
$dutch_dessert = 'Vanillevla'; // string
$x = 28; // integer
$pi = 3.1415; // double
$whatever = TRUE; // boolean
```

no declaration required, type is automatically deduced
variable names start with `$` character

they are case-sensitive, beware!

- `$Maarten != $maarten`

PHP: string variable interpolation

with double quotes ("), variable names within the text string are substituted for their value
with single quotes (') they are not
this is very useful!

```
$dutch_dessert = 'vanillevla';  
$scottish_food = 'Haggis';  
  
echo "Willem-Alexander loves $dutch_dessert<BR>";  
echo 'Why would anyone like $scottish_food?';
```

output:

```
Willem-Alexander loves vanillevla<BR>  
Why would anyone like $scottish_food?
```

PHP: strings

to include weird characters in a text string, you may have to use an *escape character*

the following statements will not work

```
$city = 's' Gravenhage';           // problem w/ quotes  
$price = "My car costs $200";     // problem w/ $-char
```

instead, do

```
$city = 's\ ' Gravenhage';        // escape character  
$city = "s' Gravenhage";         // double quotes  
$price = "My car costs \$200";    // escape character
```

PHP: string concatenation

string concatenation (pasting strings together)
use the " . operator"

example

```
echo 'Koninginne' . "dag" . '<BR>';  
  
$first = 'fiets';  
$second = 'zadel';  
echo '<B>' . $first . $second . '</B>';
```

output

```
Koninginnedag<BR>  
<B>fietszadel</B>
```

PHP: arrays

arrays can contain elements of different types

```
$myarray = array('een', 2, 'drie');  
  
echo $myarray[0];           // outputs 'een'  
echo $myarray[1];           // outputs '2'  
  
$myarray[1] = 'twee';       // changes element  
$myarray[3] = 'vier';       // creates new element  
  
$myarray[] = 'vijf';        // adds new element to array end  
echo $myarray[4];           // outputs 'vijf'
```

PHP: associative arrays

indices of array can be strings also !!!
these are 'associative arrays'
a bit strange, but quite useful

```
$country['Maarten'] = 'the Netherlands';  
$country['Maxima'] = 'Argentina';  
$country['Yiwei'] = 'China';  
  
$name = 'Maarten';  
echo "$name is from $country[$name]";
```

output:
Maarten is from the Netherlands

remember: HTML forms

GET request:

```
GET /icecream.php?scoops=2&flavor=cherry HTTP/1.0  
Referer: http://www.ml.com/shop.html  
Accept: */*
```

POST request:

```
POST /icecream.php HTTP/1.0  
Referer: http://www.ml.com/shop.html  
Accept: */*  
Content-type: application/x-www-form-urlencoded  
Content-length: 22
```

```
scoops=2&flavor=cherry
```

PHP: GET request

```
<?php
    $showmany = $_GET['scoops'];
    $which = $_GET['flavor'];

    echo "So, you want $showmany scoops of $which icecream";
?>
```

PHP automatically creates an associative array of all name-value pairs from the query part of a requested URL, called `$_GET`

PHP: POST request

```
<?php
    $showmany = $_POST['scoops'];
    $which = $_POST['flavor'];

    echo "Yes, I want $showmany scoops of $which icecream";
?>
```

PHP automatically creates an associative array of the URL-encoded name-value pairs in the entity-body of a POST request, called `$_POST`

PHP

alternatively, you can use `$_REQUEST` which contains both the GET and POST request name-value pairs

```
<?php
    $fname = $_REQUEST['firstname'];
    $lname = $_REQUEST['lastname'];
    echo "Welcome to this webpage, $fname $lname";
?>
```