CSI 3540

Structures, techniques et normes du Web

Programmation côté serveur : CGI

Objectif:

- Introduction à la programmation côté serveur
- Introduction aux **CGI**

Lectures:

- Common Gateway Interface 1.1

Plan

1. Introduction
2. Présentation détaillée du protocole CGI
    1. Exécution des CGIs
    2. **Passage de paramètres**
    3. Méthode Get
    4. Méthode Post
3. Remarques

Côté client

- Les navigateurs modernes savent :
  - Afficher des documents **XHTML**
  - Interpréter les feuilles de styles (**CSS**)
  - Exécuter des scripts **ECMAScript**
  - Donner accès aux documents et leur structure à l'aide du **DOM**

Côté client : **programmation**

- Par exemple, **valider** le format d'un numéro de téléphone, de carte de crédit, etc.
  - **Réduire** la charge du réseau et/ou du serveur
- **Construire/modifier** les pages côté client
  - Rendre l'utilisation de l'application **plus agréable** (Rich Internet Application)

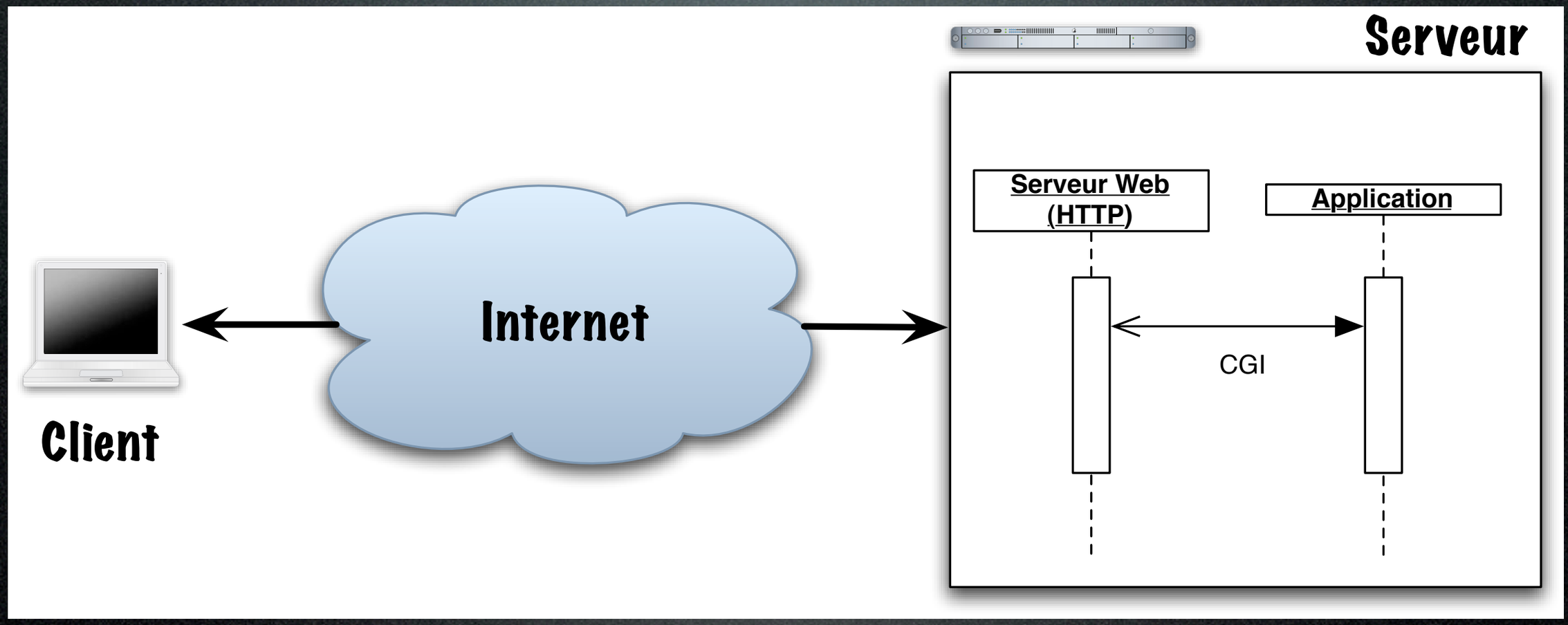
Côté serveur

- Jusqu'à maintenant, les pages servies étaient préexistantes et donc **statiques**

Côté serveur : **programmation**

- Nous abordons maintenant la génération de pages **dynamiques** par le serveur
  - **Exemple** : engins de recherche tels que Google
- 1993, **C**ommon **G**ateway **I**nterface (**CGI**)
- 1997, Sun Microsystems Java **Servlet**
- 2004, **RFC 3875** (CGI 1.1)

Common Gateway Interface



Common Gateway Interface (CGI 1.1)

- Le serveur **HTTP** est configuré afin de reconnaître les URLs d'application externes, par exemple, **/cgi-bin/**
- Définit le **passage de paramètres** entre un serveur HTTP et une application externe
- **Très général**, n'impose pas ou peu de restrictions quant au choix du langage de programmation

Configuration (Apache)

/etc/httpd/httpd.conf :

```
...
LoadModule cgi_module        libexec/httpd/mod_cgi.so
AddModule mod_cgi.c
...
ScriptAlias /cgi-bin/ "/Library/WebServer/CGI-Executables/"
...
<Directory "/Library/WebServer/CGI-Executables">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
</Directory>
...
AddHandler cgi-script .cgi
...
```

CGI : **sans** paramètre

- Il suffit de configurer le serveur
- Concevoir un programme **exécutable** produisant, sur la sortie standard (**standard i/o**), le corps du **message CGI réponse** (un en-tête, une ligne blanche, suivie du corps du message, typiquement un document XHTML)

/Library/WebServer/CGI-Executables/bonjour

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int main( int argc, char * argv[] )
{
  size_t maxsize = 256;
  char buffer[ maxsize ];
  time_t t; struct tm *timeptr;

  t = time( NULL ); timeptr = localtime( &t );
  strftime( buffer, maxsize, "%k h %M", timeptr );

  printf( "Content-type: text/html\n" );
  printf( "\n" );
  printf( "<!DOCTYPE html\n" );
  printf( "     PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\"\n" );
  printf( "     \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
  printf( "<html xmlns=\"http://www.w3.org/1999/xhtml\" lang=\"fr-CA\">\n" );
  printf( "  <head>\n" );
  printf( "    <title>Structures, techniques et normes du Web (CSI 3540)</title>\n" );
  printf( "    <meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\" />\n" );
  printf( "    <link rel=\"stylesheet\" type=\"text/css\" href=\"default.css\" media=\"all\" />\n" );
  printf( "  </head>\n" );
  printf( "  <body>\n" );
  printf( "    <h1>Exemple de CGI minimaliste</h1>\n" );
  printf( "    <p>Il est maintenant <b>%s</b>.</p>\n", buffer );
  printf( "  </body>\n" );
  printf( "</html>\n" );

  return( EXIT_SUCCESS );
}
```

```
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
    <title>Structures, techniques et normes du Web (CSI 3540)</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <link rel="stylesheet" type="text/css" href="default.css" media="all" />
  </head>
  <body>
    <h1 >Exemple de CGI minimaliste</h1 >
    Il est maintenant <b> 8 h 22</b>.
  </body>
</html>
```


```
#! /usr/bin/perl

use strict;
chomp( my $date = `date +"%k h %M"` );

print <<"EOF";
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
   <title>Structures, techniques et normes du Web (CSI 3540)</title>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <link rel="stylesheet" type="text/css" href="default.css" media="all" />
  </head>
  <body>
   <h1 >Exemple de CGI minimaliste</h1 >
   <p>Il est maintenant <b>$date</b></p>.
  </body>
 </html>
 EOF
```

En-tête du message HTTP réponse

```
#! /usr/bin/perl

use strict;
chomp( my $date = `date +"%k h %M"` );

print <<"EOF";
HTTP/1.1 200 OK
Server: www.site.uottawa.ca
MIME-Version: 1.0
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
 ...
</html>
EOF
```

CGI: Méta-Variables

1. Le serveur HTTP initialise des **variables d'environnement du système** d'exploitation pour : les paramètres (**QUERY_STRING**), le type de la requête (**REQUEST_METHOD**), ainsi que les en-têtes (**HTTP_USER_AGENT**, etc.)

```
char *getenv( const char *name );
```

Variables d'environnement : yin.c

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

char *env_init[] = { "COURSE=CSI3540", NULL };

int main( int argc, char * argv[] )
{
  int status;

  status = execle( "/Users/turcotte/bin/yan", "yan", NULL, env_init );

  return( status );
}
```

Variables d'environnement : yan.c

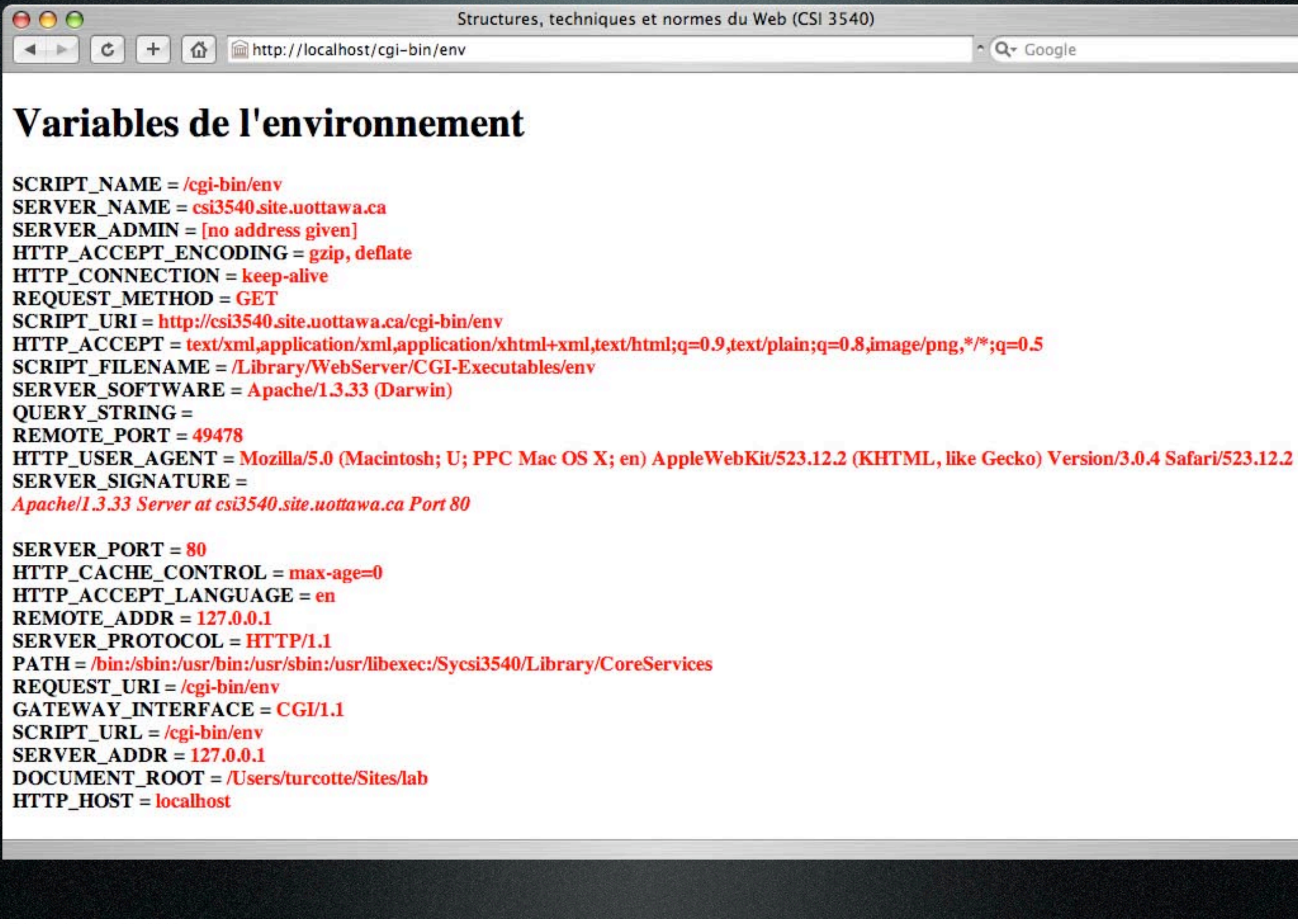
```
#include <stdlib.h>
#include <stdio.h>

int main(  int argc, char * argv[] )
{
  char *course;

  course = getenv( "COURSE" );

  printf( "course = %s\n", course );

  return( EXIT_SUCCESS );
}
```

```
$ /Users/turcotte/bin/yin
course = CSI3540
```

```c
#include <stdlib.h>
#include <stdio.h>

int main( int argc, char * argv[] )
{
  size_t maxsize = 256;
  char *method, *agent;

  method = getenv( "REQUEST_METHOD" );
  agent = getenv( "HTTP_USER_AGENT" );

  printf( "Content-type: text/html\n" );
  printf( "\n" );
  printf( "<!DOCTYPE html\n" );
  printf( "     PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\"\n" );
  printf( "     \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
  printf( "<html xmlns=\"http://www.w3.org/1999/xhtml\" lang=\"fr-CA\">\n" );
  printf( "  <head>\n" );
  printf( "    <title>Structures, techniques et normes du Web (CSI 3540)</title>\n" );
  printf( "    <meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-1\" />\n" );
  printf( "    <link rel=\"stylesheet\" type=\"text/css\" href=\"default.css\" media=\"all\" />\n" );
  printf( "  </head>\n" );
  printf( "  <body>\n" );
  printf( "    <h1>Exemple de CGI minimaliste</h1>\n" );
  printf( "    <p>Requête de type <b>%s</b>, par l'agent <b>%s</b></p>\n", method, agent );
  printf( "  </body>\n" );
  printf( "</html>\n" );

  return( EXIT_SUCCESS );
}
```

```
#! /usr/bin/perl

use strict;
my $method = $ENV{ "REQUEST_METHOD" };
my $agent = $ENV{ "HTTP_USER_AGENT" };

print <<"EOF";
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
    <title>Structures, techniques et normes du Web (CSI 3540)</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <link rel="stylesheet" type="text/css" href="default.css" media="all" />
  </head>
  <body>
    <h1 >Exemple de CGI minimaliste</h1 >
    <p>Requête de type <b>$method</b>, par l'agent <b>$agent</b></p>.
  </body>
</html>
EOF
```

```
#! /usr/bin/perl
use strict;
print <<"EOF";
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
 <head>
  <title>Structures, techniques et normes du Web (CSI 3540)</title>
 </head>
 <body>
  <h1>Variables de l'environnement</h1>
  <p>
EOF

  for my $var ( keys %ENV ) {
    print "<b>$var</b> = <b style=\"color:red\">$ENV{$var}</b><br/>\n";
  }

print <<"EOF";
  </p>
 </body>
</html>
EOF
```


Structures, techniques et normes du Web (CSI 3540)
http://localhost/cgi-bin/env
Google
Variables de l'environnement
SCRIPT_NAME = /cgi-bin/env
SERVER_NAME = csi3540.site.uottawa.ca
SERVER_ADMIN = [no address given]
HTTP_ACCEPT_ENCODING = gzip, deflate
HTTP_CONNECTION = keep-alive
REQUEST_METHOD = GET
SCRIPT_URI = http://csi3540.site.uottawa.ca/cgi-bin/env
HTTP_ACCEPT = text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
SCRIPT_FILENAME = /Library/WebServer/CGI-Executables/env
SERVER_SOFTWARE = Apache/1.3.33 (Darwin)
QUERY_STRING =
REMOTE_PORT = 49478
HTTP_USER_AGENT = Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/523.12.2 (KHTML, like Gecko) Version/3.0.4 Safari/523.12.2
SERVER_SIGNATURE =
Apache/1.3.33 Server at csi3540.site.uottawa.ca Port 80
SERVER_PORT = 80
HTTP_CACHE_CONTROL = max-age=0
HTTP_ACCEPT_LANGUAGE = en
REMOTE_ADDR = 127.0.0.1
SERVER_PROTOCOL = HTTP/1.1
PATH = /bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/Sycsi3540/Library/CoreServices
REQUEST_URI = /cgi-bin/env
GATEWAY_INTERFACE = CGI/1.1
SCRIPT_URL = /cgi-bin/env
SERVER_ADDR = 127.0.0.1
DOCUMENT_ROOT = /Users/turcotte/Sites/lab
HTTP_HOST = localhost

Common Gateway Interface (CGI)

2. Le serveur HTTP exécute le programme spécifié par l'URL, dans l'environnement fraîchement initialisé.

   Par exemple, étant donné l'URL suivante : http://localhost/**cgi-bin/params**, le serveur HTTP exécute l'application **params** (un script en langage Perl, ou un exécutable produit à partir d'un programme en C).

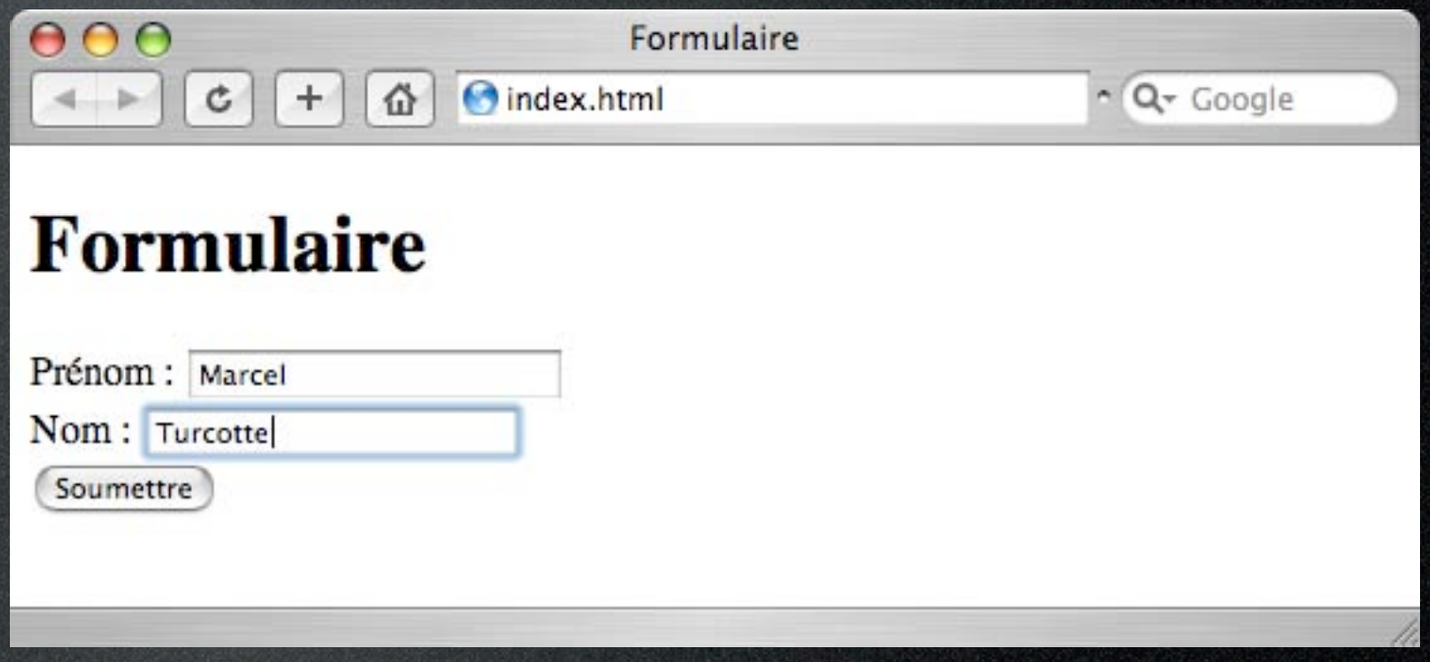
Common Gateway Interface (CGI)

3. L'application communique avec le serveur HTTP à l'aide d'un canal de communication. Les premières lignes de la sortie sont des directives pour le serveur HTTP, le reste constitue le corps de la réponse HTTP (en général, c'est un document XHTML)

CGI : Get

- Les paramètres sont passés à l'aide de la variable d'environnement **`QUERY_STRING`**

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
 <head>
  <title>Formulaire</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
 </head>
 <body>
  <h1>Formulaire</h1>
  <p>
   <form action="http://localhost/cgi-bin/args" method="get">
    <div>
     <label>Prénom : <input type="text" size="20" name="firstname" /></label><br />
     <label>Nom : <input type="text" size="20" name="lastname" /></label> <br />
     <input type="submit" value="Soumettre" />
    </div>
   </form>
  </p>
 </body>
</html>
```

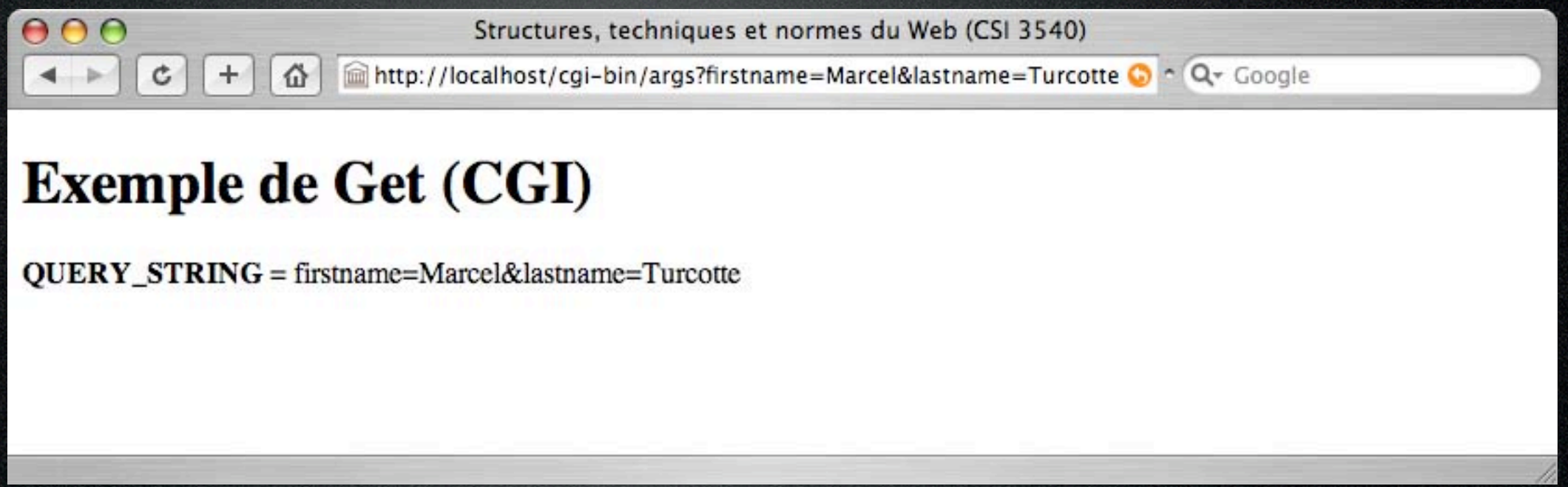


CGI : Get

- Lorsque l'usager appuie sur le bouton «**Soumettre**»
- Le navigateur saisit les **valeurs des champs** du formulaire et forme l'URL à partir de la valeur de l'attribut **action** et des valeurs des champs
- Il crée alors un message HTTP requête

```
GET /cgi-bin/args?firstname=Marcel&lastname=Turcotte HTTP/1.1
Host: localhost
```

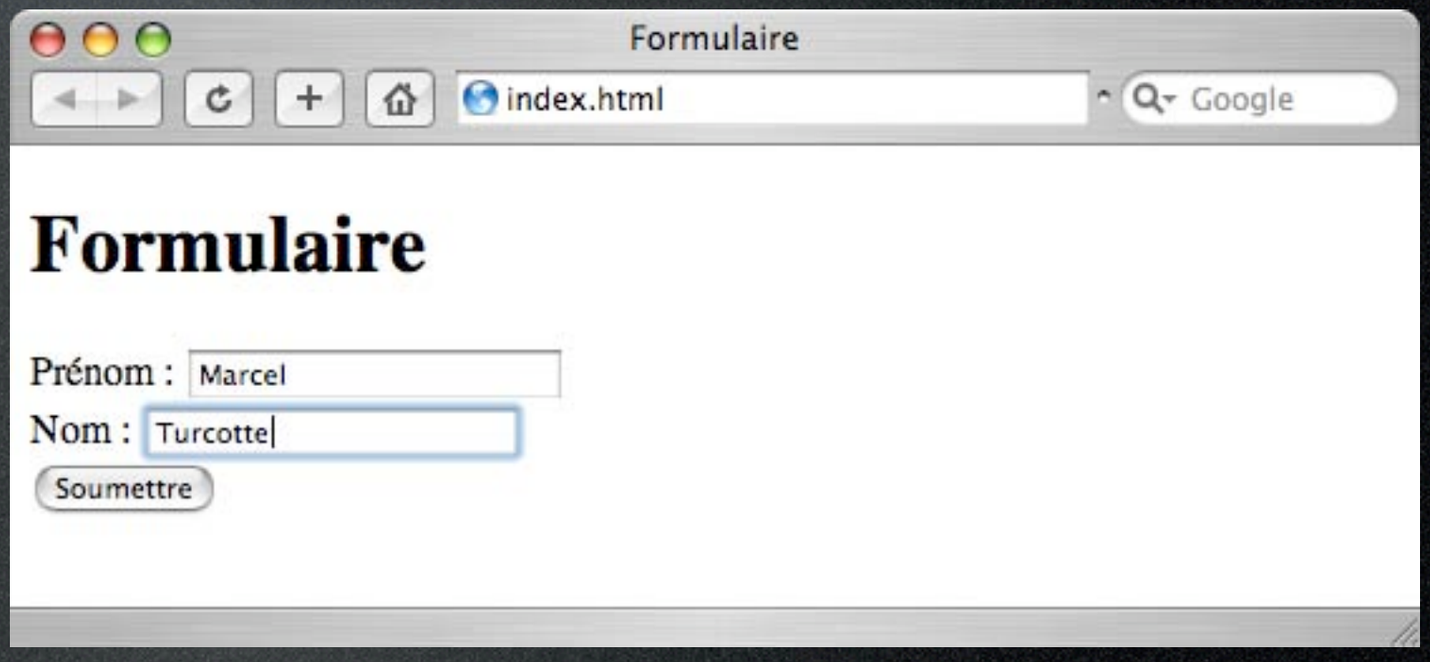
```
#! /usr/bin/perl
use strict;
my $query = $ENV{ "QUERY_STRING" };
print <<"EOF";
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
    <title>Structures, techniques et normes du Web (CSI 3540)</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <link rel="stylesheet" type="text/css" href="default.css" media="all" />
  </head>
  <body>
    <h1>Exemple de Get (CGI)</h1>
    <p><b>QUERY_STRING</b> = $query</p>
  </body>
</html>
EOF
```



CGI : Post

- Dans le cas de **POST**, le corps de la requête est passé à l'aide d'un canal de communication (pipe).
  Donc lu à partir de l'entrée standard (standard in).
- La variable d'environnement **CONTENT_LENGTH** indique le nombre d'octets à lire

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
 <head>
  <title>Formulaire</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
 </head>
 <body>
  <h1>Formulaire</h1>
  <p>
   <form action="http://localhost/cgi-bin/post" method="post">
    <div>
     <label>Prénom : <input type="text" size="20" name="firstname" /></label><br />
     <label>Nom : <input type="text" size="20" name="lastname" /></label> <br />
     <input type="submit" value="Soumettre" />
    </div>
   </form>
  </p>
 </body>
</html>
```



```
#! /usr/bin/perl
use strict;
my $content_length = $ENV{ "CONTENT_LENGTH" };
my $query;
read( STDIN, $query, $content_length );

print <<"EOF";
Content-type: text/html

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
    <title>Structures, techniques et normes du Web (CSI 3540)</title>
  </head>
  <body>
    <h1 >Exemple de Post (CGI)</h1 >
    <p><b>CONTENT_LENGTH</b> = $content_length</p>
    <p><b>Contenu</b> = $query</p>
  </body>
</html>
EOF
```

Remarques

- Dans le cas d'un **Get**, l'**URL contient les informations de la requête**
- Si l'usager sauve cette URL (**signet**), il sera en mesure de resoumettre la requête
- Dans le cas d'un **Post**, les **informations** sont passées dans le **corps du message** HTTP requête

CGI : Get/Post

```
#! /usr/bin/perl
use strict;

my $method = $ENV{ "REQUEST_METHOD" };
my $query;

if ( $method eq "GET" ) {
   $query = $ENV{ "QUERY_STRING" };
} ( $method eq "POST" ) {
   my $content_length = $ENV{ "CONTENT_LENGTH" };
   read( STDIN, $query, $content_length );
} else {
   ...
}
...
```

CGI : sans formulaire

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
 <head>
  <title>Sans formulaire</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
 </head>
 <body>
  <h1>Mes publications :</h1>
  <p>
   Consultez ce
   <a href="http://localhost/cgi-bin/search?first=Marcel&last=Turcotte">
     lien
   </a>.
   <a href="http://localhost/cgi-bin/search">
     Pour les recherches générales, c'est ici.
   </a>
  </p>
 </body>
</html>
```

application/x-www-form-urlencoded

- Les noms de commandes et les valeurs sont échappées. Les caractères « **espace** » sont **remplacés** par des caractères plus « **+** » puis les caractères réservés sont échappés comme décrit dans le document [RFC1738], section 2.2 : Les caractères non-alphanumériques sont remplacés par une séquence de la forme « %HH », un caractère pourcentage et deux chiffres hexadécimaux qui représentent le code ASCII du caractère en question. Les sauts de ligne sont représentés par des couples de caractères "CR LF" (i.e., "%0D%0A") ;

application/x-www-form-urlencoded

- Les couples **nom**/**valeur** des commandes sont listés selon leur ordre d'apparition dans le document. Le nom est **séparé** de la valeur par un caractère égal « **=** », et les couples nom/valeur sont séparés les uns des autres par des caractères esperluettes « **&** ».

http://www.la-grange.net/w3c/html4.01/cover.html

Résumé

- Le serveur HTTP initialise des variables d'environnement (**métavariables**) à partir des informations du message HTTP requête
- À partir de la portion chemin de l'URI, détermine le programme à exécuter et exécute ce programme

Résumé

- Dans le cas d'une requête POST, le serveur HTTP copie les données du client sur la sortie standard (entrée standard du CGI)
- Le CGI obtient les informations du message HTTP requête à l'aide des métavariables (variables d'environnement)

Résumé

- Dans le cas d'un message HTTP requête POST, le CGI fait la lecture des données sur l'entrée standard (utilise la variable CONTENT_LENGTH pour déterminer le nombre d'octets à lire)
- [Traitement]
- Le CGI construit une réponse CGI (entête, ligne blanche, contenu) qu'il affiche sur la sortie standard

Résumé

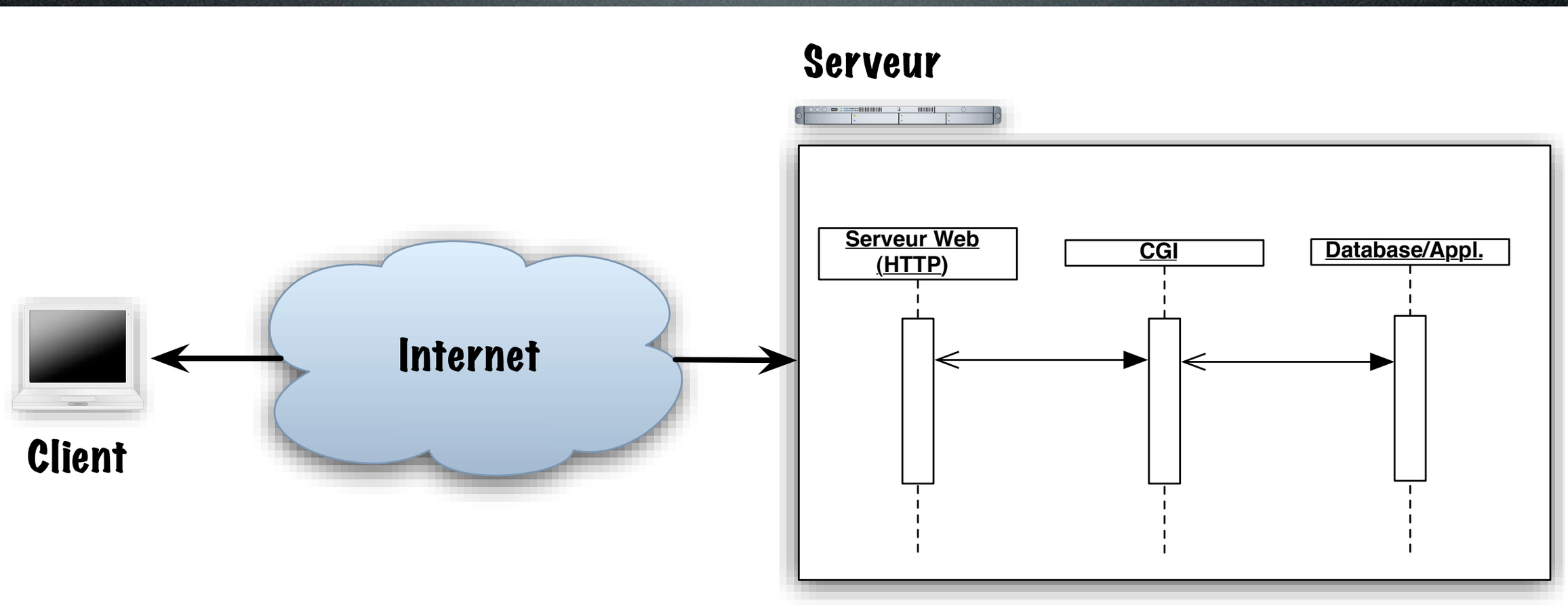
- Le serveur HTTP obtient la réponse CGI sur l'entrée standard, construit un message HTTP réponse à partir des entêtes et du corps de la réponse CGI
- Le serveur HTTP retourne le message HTTP réponse ainsi construit au client

Remarques

- Plusieurs bibliothèques de programmes existent, pour plusieurs langages de programmation, afin de faciliter l'écriture de CGI
- CGI.pm est l'une des bibliothèques pour Perl
- CGIC est une bibliothèque C http://www.boutell.com/cgic/

- Le démon HTTP (**httpd**) tourne généralement sous un nom d'usager sans privilège (deamon (2), nobody (60001), http...)
- Éviter de faire tourner sous root (0)
- Cet usager exécutera le programme CGI
- S'il y a des accès disques, cet usager doit avoir accès aux disques (E et/ou S)
- Le module **suEXEC** d'Apache HTTPD
- S'il y a des accès à une banque de données, il y aura un autre usager

Common Gateway Interface



- L’exécution de programmes CGI comporte plusieurs risques!
- En général, on ne permet pas à tous les usagers de créer des programmes CGI
- Les fichiers de configuration et les exécutables appartiennent à l’usager root
- Surveiller les fichiers journaux (logs)
- Certaines constructions sont à éviter eval `echo $QUERY_STRING | ... `

Pour le compteur : on peux sauvegarder la valeur dans un fichier qui est lu pour chaque

Discussion

- Quelle serait votre démarche pour créer un compteur de visiteurs ?
- Quelle serait votre démarche pour créer une application Web nécessitant plusieurs formulaires ?
- Pour de simples applications, on peut utiliser des champs cachés
- On sent le besoin d'une couche logicielle plus musclée

Remarques

- HTTP est un protocol sans état (**stateless)**
- Chaque requête force l'exécution d'une copie fraîche du programme

Alternatives

- Protocol CGI «Common Gateway Interface» (1995, NCSA)
  - C, C++, Perl, Python, etc.
  - PHP
  - ASP.NET
  - Java Server Pages (JSP)

Ressources

- CGI: Common Gateway Interface [http://www.w3.org/CGI/] 2007
- http://tools.ietf.org/html/rfc3875
- W. R. Stevens (1992) Advanced Programming in the UNIX(R) Environment. Addison-Wesley.