Langage Perl appliqué au Net

Michel Dubois

duboism@iu-vannes.fr

Objectifs du cours

- Utilisation des principales bibliothèques objets du CPAN fournissant des solutions web simple mais performantes.
- Exposer la facilité de l'utilisation sans éluder les problèmes d'installation et de configuration des différents modules
- Montrer les synergies possibles entre les différents modules.

Plan

- CPAN pour installer ces modules
- API offrant des possibilités clientes
  - libnet
  - libwww-perl
- API offrant des possibilités plutôt serveur
  - DBI
  - CGI
  - Apache

L'aide en ligne : perldoc

- Les fonctions de perl : perldoc perlfunc
- La syntaxe d'une fonction : perldoc -f open
- Chercher dans la FAQ: perldoc -q open

A noter: il existe un site de traduction en français des informations sur la distribution de base.

- S'informer sur un package: `perldoc DBI`
- Les modules installés en sus: `perldoc perllocal`

Constantes et Scalaires

- 1, -12345, 1.6E16 *(signifie 160 000 000 000 000 000)*,
- 'cerise', 'a', 'les fruits du palmier sont les dattes'.
- Les scalaires sont précédés du caractère $

```
$i = 0; $c = 'a';
$mon_fruit_prefere = 'kiwi';
$racine_carree_de_2 = 1.41421;
$chaine = '100 grammes de $mon_fruit_prefere';
=> '100 grammes de $mon_fruit_prefere'
$chaine = "100 grammes de $mon_fruit_prefere";
=> '100 grammes de kiwi'
```

- Attention : Pas d'accents ni d'espaces dans les noms de variables
- Par contre un nom peut être aussi long qu'on le veut.

Tableaux, listes

- En Perl, les tableaux peuvent être utilisés comme des ensembles ou des listes.
- Toujours précédés du caractère « @ »

@chiffres = (1,2,3,4,5,6,7,8,9,0);

@fruits = ('amande','fraise','cerise');

@alphabet = ('a'..'z'); *Les deux points signifient de "tant à tant"*

@a = ('a'); @nul = ();

@cartes = ('01'..'10','Valet','Dame','Roi');

- On fait référence à un élément du tableau selon son indice par :

$chiffres[1] (=> 2)

$fruits[0] (=> 'amande')

- REMARQUE : En Perl (comme en C) les tableaux commencent à l'indice 0

Tableaux, listes (2)

- On peut affecter un tableau à un autre tableau :

```
@ch = @chiffres;
@alphanum = (@alphabet, '_', @chiffres);
=> ('a','b,',...,'z','_','1','2','3','4','5','6','7','8','9','0')
@ensemble = (@chiffres, 'datte', 'kiwi', 12.45);
```

- Remarques :
- On dispose d'un scalaire spécial : $#tableau qui indique le dernier indice du tableau (et donc sa taille - 1) : $fruits[$#fruits] (=> 'cerise')
- *Possibilité de référencer une partie d'un tableau*

```
@cartes[6..$#cartes] => ('07','08','09','10','Valet','Dame','Roi')
@fruits[0..1] => ('amande', 'fraise')
```

Tableaux indicés (ou associatifs)

- Ils sont toujours précédés du caractère % :

```
%prix = ('amande'=>30, 'fraise'=>9, 'cerise'=>25);
%prix = (amande=>30, fraise=>9, cerise=>25);
%prix = ('amande',30, 'fraise',9, 'cerise',25);
```

- On référence ensuite un élément du tableau par : $prix{'cerise'} (=> 25)

(ou $prix{cerise})

- Exemples:

```
%chiffre = ();
$chiffre{'un'} = 1; => ou $chiffre{un} = 1;
print $chiffre{un};
$var = 'un'; print $chiffre{$var};
```

Structures complexes

- En Perl (version 5), on utilise la syntaxe des tableaux indicés pour désigner une structure complexe, exemple:

```
$patient->{nom} = 'Dupont';
$patient->{prenom} = 'Albert';
$patient->{age} = 24;
```

- On peut initialiser la structure comme lorsqu'on initialise un tableau indicé :

```
$patient = {
nom => 'Dupont',
prenom => 'Albert',
age => 24};
```

- On utilise les accolades plutôt que les parenthèses (référence vers un tableau indicé)

Structures complexes (2)

- Pour afficher le contenu d'une structure complexe :

```
print "Nom:$patient->{nom} $patient->{prenom},âge:$patient->{age}";
foreach $champ ('nom','prenom','age') {
                        print "$champ : $patient->{$champ}\n";}
```

- ou encore :

```
foreach $champ (keys %$patient) {
                        print "$champ : $patient->{$champ} ";}
```

- Les champs apparaissent dans le "désordre" comme dans l'affichage d'un tableau indicé
- Pour référencer $patient comme un tableau indicé on utilise cette syntaxe

Perl Objet

- Perl (version 5) permet de définir des objets.
- En résumé on associe à un « objet » des données et des « méthodes ». Ce qui simplifie l'utilisation ensuite de ces objets.
- En Perl objet : un « objet » est une référence , une « méthode » est une procédure , une « classe » d'objet est un package .

En pratique

- On crée un fichier Perl (avec le suffixe .pm) qui contient la classe d'objet que l'on veut définir
- Deux méthodes particulières :
  - **new**, constructeur, appelée automatiquement à la création de l'objet
  - et **DESTROY**, destructeur, appelée automatiquement à la destruction de l'objet
- *Notes:*

  ⇒ On déclare les variables avec my plutôt que local en Perl 5

  ⇒ La fonction bless permet de rendre un objet « visible » de l'extérieur.

Exemple de classe (1/2)

```
#!/bin/perl
package patient; # Déclaration d'un package
# Un nouveau patient consiste à lui donner un nom
# et éventuellement une unité
sub new { # Constructeur
my ($class, $nom, $unite) = @_;
my $patient = {}; # Structure complexe
$patient->{nom} = $nom;
if ($unite) { # Si unité définie
$patient->{unite} = $unite;
} else {
$patient->{unite} = 'Non définie';
}
return bless $patient;
}
```

Exemple de classe (2/2)

```
sub transfert { # Transfert du patient vers nouvelle unité
my ($patient, $nvunite) = @_;
$patient->{unite} = $nvunite;
}
sub affiche { # Affichage de la situation du patient
my ($patient) = @_;
print "Le patient $patient->{nom} est dans l'unité
$patient->{unite}\n";
}
sub DESTROY {
my ($patient) = @_;
print "Le patient $patient->{nom} est parti !\n";
}
1; # Une classe d'objet se termine toujours par 1;
```

Exemple d ’utilisation de classe (1/2)

```
#!/bin/perl
use sih; # Directive pour dire qu'on utilise le package SIH
# Déclaration de deux nouveaux patients
$patient1 = new patient('Dupont Pierre');
$patient2 = new patient('Durand Albert', 'urgences');
$patient1->affiche; # Appel d'une méthode pour patient
$patient2->affiche; # (affichage de sa situation)
$patient1->transfert('cardio'); # transfert vers nelle unite
$patient2->transfert('pneumo');
$patient1->affiche; # Affichage de la situation des 2 patients
$patient2->affiche;
# fin du programme, appel des destructeurs
```

Exemple d ’utilisation de classe (2/2)

Le résultat de ce programme sera :

Le patient Dupont Pierre est dans l'unité Non définie
Le patient Durand Albert est dans l'unité urgences
Le patient Dupont Pierre est dans l'unité cardio
Le patient Durand Albert est dans l'unité pneumo
Le patient Durand Albert est parti !
Le patient Dupont Pierre est parti !

Le module Perl CPAN.pm
Comprehensive Perl Archive Network

Michel Dubois

duboism@iu-vannes.fr

Objectifs

- Faciliter les installations
- Faciliter les mises à jour
- Centralisation des modules

- Miroirs dans le monde entier (+56 en europe)
- Plus de 400 modules (de haut et de bas niveau)
- +300 Contributeurs

Configuration

```
$CPAN::Config = {
 'build_cache' => q[10],
 'build_dir' => q[/home/prof/duboism/.cpan/build],
 'cpan_home' => q[/home/prof/duboism/.cpan],
 'ftp_proxy' => q[www-cache:3128],
 'http_proxy' => q[www-cache:3128],
 'keep_source_where' => q[/home/prof/duboism/.cpan/sources],
 'make_arg' => q[],
 'make_install_arg' => q[],
 'makepl_arg' =>
  q[PREFIX=/devlpt/bilbo/etud/fcduia/WebDev/local/perl],
 'prerequisites_policy' => q[follow],
 'scan_cache' => q[atstart],
 'urllist' => [q[ftp://ftp.lip6.fr/pub/perl/CPAN/]],
 'wait_list' => [q[wait://ls6.informatik.uni-
  dortmund.de:1404]],
};
```

Mise en oeuvre

```
Perl -MCPAN -eshell
```

Les commandes :

```
help
install
```

Exemple: installation de DBD::CSV (1)

```
duboism@pc-b020-01:~/PerlTP 72 > perl -MCPAN -eshell
cpan shell -- CPAN exploration and modules installation (v1.50)
ReadLine support enabled
cpan> install DBD::CSV
Going to read
/home/prof/duboism/.cpan/sources/authors/01mailrc.txt.gz
Going to read
/home/prof/duboism/.cpan/sources/modules/02packages.details.txt.gz
Scanning cache /home/prof/duboism/.cpan/build for sizes
Deleting from cache: /home/prof/duboism/.cpan/build/libnet-1.0607
   (10.3>10.0 MB)
  There's a new CPAN.pm version (v1.52) available!
  You might want to try
    install Bundle::CPAN
    reload cpan
  without quitting the current session. It should be a seamless
   upgrade
  while we are running...
```

Exemple: installation de DBD::CSV (2)

```
Going to read
   /home/prof/duboism/.cpan/sources/modules/03modlist.data.gz
Running make for JWIED/DBD-CSV-0.1022.tar.gz
CPAN: MD5 loaded ok
Checksum for /home/prof/duboism/.cpan/sources/authors/id/JWIED/DBD-CSV-
   0.1022.tar.gz ok
DBD-CSV-0.1022/
DBD-CSV-0.1022/Makefile.PL
...
DBD-CSV-0.1022/README

  CPAN.pm: Going to build JWIED/DBD-CSV-0.1022.tar.gz


Checking for DBI, 1.00 or later ... ok
Checking for Text::CSV_XS, 0.16 or later ... ok
Checking for SQL::Statement, 0.1011 or later ... ok
Looks good
Writing Makefile for DBD::CSV
```

Exemple: installation de DBD::CSV (3)

```
mkdir blib/man3
cp lib/DBD/File.pm blib/lib/DBD/File.pm
cp lib/DBD/CSV.pm blib/lib/DBD/CSV.pm
cp lib/Bundle/DBD/CSV.pm blib/lib/Bundle/DBD/CSV.pm
Manifying blib/man3/DBD::File.3
Manifying blib/man3/DBD::CSV.3
Manifying blib/man3/Bundle::DBD::CSV.3
  /usr/bin/make  -- OK
Running make test
PERL_DL_NONLAZY=1 /devlpt/bilbo/etud/fcduia/WebDev/local/perl/bin/perl
   -Iblib/arch -Iblib/lib -
   I/devlpt/bilbo/etud/fcduia/WebDev/local/perl/lib/5.00503/i586-linux
   -I/devlpt/bilbo/etud/fcduia/WebDev/local/perl/lib/5.00503 -e 'use
   Test::Harness qw(&runtests $verbose); $verbose=0; runtests @ARGV;'
   t/*.t
t/00base.............ok
t/10dsnlist..........ok
t/20createdrop......ok
All tests successful.
```

Exemple: installation de DBD::CSV (4)

```
/usr/bin/make test -- OK
Running make install
Installing
   /devlpt/bilbo/etud/fcduia/WebDev/local/perl/lib/site_perl/5.005/DBD/
   CSV.pm
Writing
   /devlpt/bilbo/etud/fcduia/WebDev/local/perl/lib/site_perl/5.005/i586
   -linux/auto/DBD/CSV/.packlist
Appending installation info to
   /devlpt/bilbo/etud/fcduia/WebDev/local/perl/lib/5.00503/i586-
   linux/perllocal.pod
  /usr/bin/make install  -- OK


cpan>
```

Le module Perl Net.pm ou libnet

Michel Dubois

duboism@iu-vannes.fr

Configuration de libnet (Net::Config)

```
'test_hosts' => 1,
'nntp_hosts' => ['news'],
'snpp_hosts' => [],
'pop3_hosts' => ['mailens'],
'ftp_ext_passive' => 0,
'smtp_hosts' => ['mailens'],
'ftp_testhost' => 'ftp',
'inet_domain' => 'iu-vannes.fr',
'ph_hosts' => [],
'test_exist' => 1,
'daytime_hosts' => [],
'ftp_int_passive' => 0,
'ftp_firewall' => undef,
'time_hosts' => [],
```

La classe Net::FTP

- Elle permet d'intégrer dans vos scripts Perl le téléchargement de fichiers via le protocole FTP

- La documentation relative :

```
perldoc  Net::FTP
man Net::FTP
```

La classe Net::FTP

- On se connecte au serveur (méthode `new()`)
- On s'identifie auprès du serveur (`login()`)
- On se place dans le répertoire distant où se trouve le fichier (méthode `cwd()`)
- On change le mode de transfert (`type()`)
- On lance la récupération (get()) ou son envoi (`put()`)
- On ferme la connexion par la méthode `quit()`.

Exercice Net::FTP

Faire un programme Perl qui fasse un ftp anonyme sur le serveur `ftp` en vue de récupérer un fichier `README` se trouvant dans le répertoire `/pub/perl/CPAN`

Récupération de fichier

```
#!/usr/bin/perl -w
use strict;
use Net::FTP;
my $serveur= 'ftp';
my $source='README';
### on crée l'objet qui va faire le travail
my $ftp=Net::FTP->new($serveur);
### on fait une connexion ftp anonyme
$ftp->login();
### on se déplace dans l'arborescence du serveur ftp
$ftp->cwd('/pub/perl/CPAN');
### on récupère le fichier dans le répertoire courant
### le fichier cible peut être le deuxième argument
$ftp->get("$source");
$ftp->quit;
exit 0;
```

La classe Net::SMTP

- Elle permet d'intégrer dans vos scripts Perl l'envoi d'un courrier en utilisant le protocole Simple Mail Transfer Protocole (RFC821).

- La documentation relative :

```
perldoc  Net::SMTP
man Net::SMTP
```

Exercice Net::SMTP

Faire un programme Perl qui envoie un mail au serveur `mailens` dont le corps est contenu dans le fichier `article.txt` .

Le destinataire est l'expéditeur. Il s'agit de vous même.

Envoi d'un mail (1)

```
#!/usr/local/bin/perl -w
use strict;
use Net::SMTP;

my $expediteur='Michel DUBOIS <duboism\@iu-vannes.fr>';
my $destinataire='Michel DUBOIS <duboism\@iu-vannes.fr>';
my $message="article.txt";
$smtp = Net::SMTP->new('mailens');
$smtp->mail($expéditeur);
$smtp->to($destinataire);
$smtp->data();
$smtp->datasend("To: $destinataire\n");
$smtp->datasend("Subject: Essai de Net::SMTP\n");
$smtp->datasend("\n");
```

Envoi d'un mail (2)

```
open(SOURCE,$message) or die "Impossible d'ouvrir le fichier
   $message en lecture...\n";
while (<SOURCE>)
{
$smtp->datasend($_);
}
close(SOURCE);
$smtp->dataend();
$smtp->quit;
quit 0;
```

La classe Net::POP3

- C'est une classe qui permet d'interroger un serveur POP (Post Office Protocole :`RFC1081`)
- La documentation relative :

```
perldoc  Net::POP3
man Net::POP3
```

Exercice Net::POP3

Faire un programme Perl qui interrogera votre boite POP, située sur le serveur `mailens.`

Pour chaque message encore disponible, il affichera sa taille en octets, l'expéditeur et le sujet.

Interrogation de la boite POP (1)

```
#!/usr/local/bin/perl -w
use strict;
use Net::POP3;

my $serveur="mailens";
my ($user, $passwd)=("duboism","xxxx");
my $pop = Net::POP3->new('mailens') or die ("Erreur de
   connexion à $serveur : $!\n");
### utilisation de MD5 pour l'authentification
$pop->apop($user, $passwd) or die ("Echec de
   l'authentification : $!\n");
my $ref_liste_messages=$pop->list();
my ($num_message, $ref_contenu_message);
my ($from, $subject);
printf "Message\tTaille\tFrom:\tSubject:\n\n";
```

Interrogation de la boite POP (2)

```
### la méthode $pop->list() renvoie une table de hash d'ou
   le préfixe %
foreach $num_message (keys %$ref_liste_messages)
{
   print "$num_message\t$ref_liste_messages-
   >{$num_message}";j
   $ref_contenu_message = $pop->top($num_message);
### la méthode $pop->top() renvoie un tableau d'ou le
   prefixe @$
   foreach (@$ref_contenu_message)
   {
       if (/From:(.*)/){$from=$1;}
       if (/Subject:(.*)/){$subject=$1;}
   }
   print "\t$from\t$subject\n";
}
$pop->quit();
exit 0;
```

La classe Net::NNTP

- C'est une classe implémentant en Perl un client NNTP simple, conforme à la description qu'en fait le RFC977. Net:NNTP hérite ses méthodes de communication de Net::Cmd.

- La documentation relative :

```
perldoc  Net::NNTP
man Net::NNTP
```

Exercice Net::NNTP

Faire un programme Perl qui poste un article de test dans `fr.test`. Les données sont les suivantes :

- Le serveur de news est : `news`
- Le contenu de l'article se trouve dans : `article.txt`
- Un fichier signature (fichier texte n'excédant pas 4 lignes de 72 caractères) est `.signature`.

Postage d'un article (1)

```
#!/usr/bin/perl -w
use strict;
use Net::NNTP;
my $serveur= 'news';
my $groupe='fr.test';
my $article='article.txt';
open(ARTICLE,$article) or die "$article est introuvable\n";
### on crée l'objet qui va faire le travail et on se connecte
my $nntp=Net::NNTP->new($serveur);
$nntp->postok() or die "Non autorisé à poster \n";
### on est un lecteur et non un serveur
$nntp->reader;
### toutes nos commandes NNTP vont s'appliquer au groupe
$nntp->group($groupe);
### on poste notre article dans le groupe.
### aucun paramètre : il faut utiliser
### les méthodes datasend() et dataend() du module Net::Cmd
### dont Net::NNTP hérite.
$nntp->post();
```

Postage d'un article (2)

```
### On poste l'entête de l'article
### La ligne vide permet de séparer l'entete du corps
$nntp->datasend(<<EOF);
From: Michel DUBOIS <duboism\@iu-vannes.fr>;
Subject: Test d'envoi par un script Perl
Followup-To: poster
Newsgroups: $groupe

EOF
my $signature='.signature';
### L'operateur diamant <> dans un contexte de liste
### est l'ensemble des lignes du fichier représenté
### par le descripteur ARTICLE
$nntp->datasend(<ARTICLE>);
close(ARTICLE);
$nntp->datasend("\n");
$nntp->datasend("-- \n");
```

```
open(SIGN,$signature);
$nntp->datasend(<SIGN>);
close(SIGN);
$nntp->dataend();
$nntp->quit;
exit 0;
```

Le module Perl LWP.pm ou libwww-perl

Michel Dubois

duboism@iu-vannes.fr

Bibliothèque Web pour Perl

LWP exécute les requêtes clients sur le réseau :

LWP::Simple pour faire un client web rapidement

LWP::UserAgent plus complexe

LWP::RobotUA : recherche automatisée de ressources sur le net

LWP::Simple

```
#!/usr/bin/perl
use LWP::Simple;
print (get ($ARGV[0]));
```

LWP::Simple est simple mais limité :

pas de code d'état(get renvoie `undef` en cas d'erreur), pas d'identification de la part du programme, pas de support de serveur mandateur.

LWP::UserAgent (1)

On crée une requête avec un objet `HTTP::Request`

On passe cette requête à la méthode `request` de `LWP::UserAgent` qui fait la demande réelle.

Il renvoie un objet `HTTP::Response` qui fournit les méthodes pour déterminer le code de réponse et éditer le résultat.

LWP::UserAgent (2)

```
#!/usr/bin/perl
use LWP::UserAgent;
use LWP::Request;
use LWP::Response;
my $ua=new LWP::UserAgent;
my $request=new HTTP::Request('Get',$ARGV[0]);
my $response=$ua->request($request);
if ($response->is_success){
  print $response->content;
} else {
  print $response->error_as_HTML;
}
```

LWP::RobotUA

C'est une sous-classe de `LWP::UserAgent`.

Des sites ont un fichier `robots.txt` dans leur racine. Ceci permet de contrôler le fonctionnement de ces clients automatiques (et de les exclurent). `LWP::RobotUA` respecte les normes `Robot Exclusion Standard`.

Les passerelles SGBD
Le module DBI.pm

Michel Dubois

duboism@iu-vannes.fr

Plan

Introduction

I Mise en oeuvre de DBI

- installation des DBD et /ou du proxy
- utilisation de DBI
- débogage de DBI

II Utilisation de DBI

- Requêtes & autres ordres SQL
- Gestion des erreurs
- Gestion des transactions

III Un exemple : la fnuc

Conclusion

Les objectifs de DBI

- Etre d'une utilisation aisée pour les applications simples
- Etre suffisamment flexible pour pouvoir s'appliquer à plusieurs types de sources de données (même des bases non SQL)
- Permettre la création de script Perl indépendant des bases de données utilisées.
- Etre gratuit avec un code sous GPL

Un petit exemple

```
use DBI;

$dbh = DBI->connect('dbi:ODBC:PRICE', 'user', 'password',
        { RaiseError => 1, AutoCommit => 0 });
$upd = $dbh->prepare('UPDATE prices SET price=? WHERE prod=?');
$ins = $dbh->prepare('INSERT INTO prices(prod,price) VALUES (?,?)');

while ($line = <COST>) {
    chop $line;
    ($prod, $price) = split /,/, $line;
    $rows = $upd->execute($price, $prod);
    $ins->execute($prod,$price) if $rows == 0;
}
$db->commit;
```

DBI versus ODBC et JDBC

- ODBC, JDBC et DBI partagent le même souci d'indépendance de l'interfacage avec les SGBD.
- ODBC est certainement l'API la plus riche mais aussi la plus complexe.DBI se caractérise comme JDBC par sa simplicité liée notamment par des fonctionnalités plus restreintes.
- DBI est une API de plus haut niveau que JDBC donc d'ODBC.
- Le pilote `DBD::ODBC` permet la connexion à un SGBD de la plateforme Windows (Microsoft Access)

Les origines de DBI

- OraPerl, IngPerl, SybPerl sont des modules dédiés à l'interfaçage d'un SGBD spécifique.
- DBI est le module de la version 5 de perl
- DBI fourni une émulation pour assurer une compatibilité ascendante.

Quelques sites interessants

- Le site de DBI
  - http://www.symbolstone.org/technology/perl/DBI

- ``Programming the Perl DBI'', par Alligator Descartes et Tim Bunce, sera bientôt disponible

Plan

Introduction

I Mise en oeuvre de DBI

- installation des DBD et /ou du proxy
- utilisation de DBI
- débogage de DBI

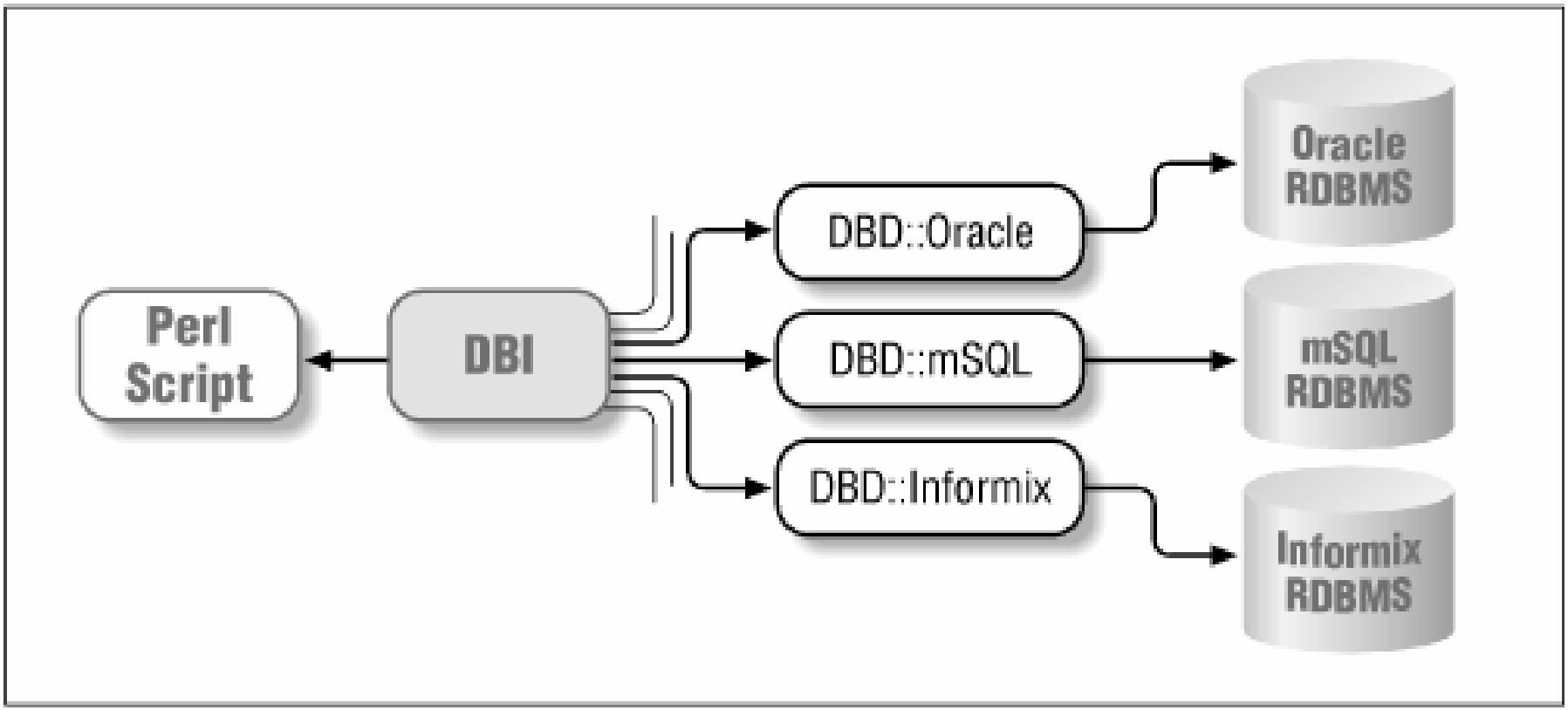
II Utilisation de DBI

- Requêtes & autres ordres SQL
- Gestion des erreurs
- Gestion des transactions

III Un exemple : la fnuc

Conclusion

L'Architecture de DBI



DBI

- DBI = *D*ata*B*ase *I*nterface
- DBI pour *database-independent*
- DBI permet d'écrire des programmes en perl indépendamment des SGBD cibles

DBD

- DBD = *D*ata*B*ase *D*river
- DBI utilise les DBD pour accéder aux SGBD
- DBD pour *database-dependent*
- De plus en plus de pilotes sont disponibles : Oracle, Informix, Ingres, mSQL, MySQL, Solid PostgreSQL, fichiers CSV, etc.

Se connecter à une base de donnée

- Tout script DBI doit faire cette première opération
- Appel de la méthode `DBI->connect()`

```
$dbh = DBI->connect( 'dbi:Oracle:base',
                              'dubois', 'pass' );
$dbh = DBI->connect(
'dbi:Pg:dbname=mdtest;host=bilbo;port=5432'
,'user', 'pass' );
```

- Une connexion effective est représenté par un database handle en resultat de l'appel.

Se déconnecter d'une base de données

- Lorsque tous les ordres SQL ont été executés et exploités, il faut se déconnecter de la base
- Ceci permet de libérer les ressources prises par la connexion. Une déconnexion peut avoir une influence sur les transactions.
- Ceci est fait par l'appel de `disconnect()` d'un objet database handle valide. Par exemple :

```
$dbh->disconnect();
```

Example d'une connexion

```
#!/usr/bin/perl -w

### Load the DBI module
use DBI;

### Perform the connection using the Oracle driver
$dbh = DBI->connect( "dbi:Oracle:DEV", "username",
                     "password" )
    or die "Can't connect to Oracle database: $DBI::errstr\n";

### Disconnect from the database
$dbh->disconnect();

exit;
```

Installation des DBD

- Les DBD doivent être correctement installées avant leur utilisation.
- Bien souvent, ils utilisent des bibliothèques clientes des SGBD pour realiser des connexions aux SGBD. Il faut installer le DBD qui est compatible avec la bibliothèque cliente.
- Le paramétrage du DBD peut nécessiter la définition de variables d'environnement.

Exemple d'installation du DBD::PG

- La dernière version de DBD::PG necessite des bibliothèques clientes PostgreSQL 6.5.x. La version sur bilbo de PostgreSQL est 6.4.2. Par contre la version 0.89 de DBD::PG est compatible avec la 6.4.x.
- Des variables d'environnements doivent être définies.

Le proxy pour le client fin

- L'installation de bibliothèque clientes du SGBD correspond à un client obèse. La solution proxy de DBI peut être une solution client fin
- De cette manière, on peut accéder à des machines distantes d'une manière plus aisée que la première solution, surtout quand les bases de données ne sont pas dotées de fonctionnalités réseau comme les fichiers CSV !

L'architecture client/serveur Proxy

- La connexion client/serveur se fait à l'aide d'un protocole indépendant de tout SGBD (TCP/IP)
- Un *DBI Proxy Server* (non WWW proxy server) tourne sur le système hôte du SGBD.
- Un *DBI Proxy Client* correspondant à un script Perl qui utilise le pilote `DBD::Proxy` pour établir la connexion et non un véritable DBD (spécifique à un type de source de données)

La configuration du serveur proxy DBI

- DBI est livré avec un programme `dbiproxy` qui est un petit server proxy DBI.
- Vous devez installer le DBD spécifique au SGBD sur le serveur hôte du SGBD. A noter qu'il dispose déja des bibliothèques clientes.
- dbiproxy peut être lancé avec comme argument un numéro de port à écouter.

```
dbiproxy --localport 3333
```

La configuration du client proxy DBI (1)

- La connection à la base de données distante se fait via la connexion au serveur proxy DBI se trouvant .

- Cette connexion se fait tout simplement.

La configuration du client proxy DBI (2)

- Par exemple, si votre script Perl local sur une partition linux se connecte à une base Oracle sur AIX, le code normal aurait été:

```
$dbh = DBI->connect( 'dbi:Oracle:base', '', '' );
```

- Cependant le DSN doit permettre de se connecter au serveur proxy DBI au lieu d'une connexion directe à la base :

```
$dbh = DBI->connect(
"dbi:Proxy:hostname=helios;port=3333;dsn=dbi:Oracle:base"
, '', '' );
```

Les flux de données du proxy

- Le client se connecte par TCP/IP au port du serveur proxy.
- Le serveur proxy récupère le DSN et se connecte à la base. Tout appel à la base par le script client sera repercuté par le serveur proxy à la base de données.
- De même le serveur proxy transmet les résultats de ses appels au script client.

Utilisation du module DBI

- DBI nécessite au debut du script l'instruction

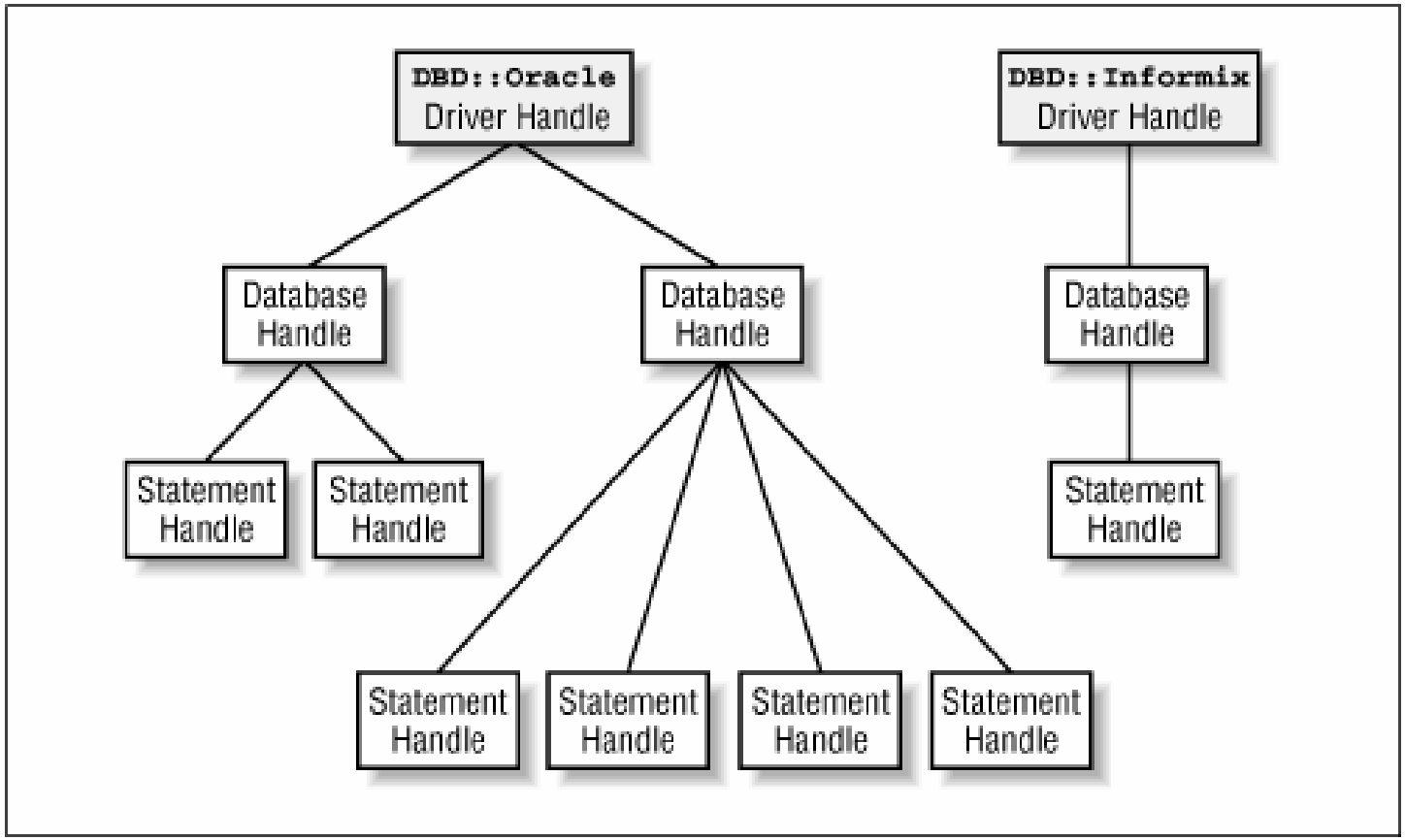
```
use DBI;
```

- DBI ne nécessite pas pour les DBD d'instruction particulière comme `use` ou `require`. Par contre il faut installer de manière correcte les DBD .

Handles

- Toute interaction avec les SGBD se fait *via* les méthodes du module DBI et les méthodes d'instances d'objets de 3 types :
  - *Driver Handle*
  - *Database Handle*
  - *Statement Handle*
- Les *Driver Handles* ne sont pas utilisés de manière explicite dans un script Perl

L'utilisation des Handles



Database Handles

- *Database Handles* encapsule une connexion à une base de données
- Les Database handles sont créés d'une manière interne par un driver handle correspondant au SGBD cible lors de l'appel à la méthode `DBI->connect()` dans un script Perl.

```
$dbh = DBI->connect( … );
```

Statement Handles

- *Statement Handles* encapsule un ordre SQL à soumettre au SGBD.
- Les Statement handles sont créés *via* un database handle. L'ordre est ainsi soumis au SGBD représenté par le database handle

```
$sth = $dbh->prepare( "SELECT yadda FROM blah" );
```

- Statement handles permet de retirer les résultats des ordres soumis.

Data Sources (1)

- Un Data Source comprend au moins deux éléments:
  - Le type de base de données
  - Le nom ou l'emplacement de cette base
- Les chaînes Data Source ont la forme suivante :

```
dbi:database_type:database_location:key=value;...
```

Data Sources (2)

- Pour se connecter à une base de données ORACLE appelées DEV, on peut utiliser DSN:

```
dbi:Oracle:DEV
```

- mSQL, MySQL, PostgreSQL nécessitent le nom de l'hôte du SGBD et le numéro de port écouté par le SGBD.
- Les DBD sur des fichiers comme DBD::CSV necessite le répertoire contenant les fichiers de données

Data Sources (3)

- Les DSN sont spécifiques aux SGBD.

- Lors d'un changement de SGBD :
  - Les chaînes de connexion au niveau des appels `DBI->connect()` doivent être changées.
  - Elles doivent être les seuls éléments à changer dans un script Perl DBI bien écrit.

Les pilotes disponibles

Les méthodes suivantes peuvent être utilisées avant une tentative de connexion à une base de données :

- Liste des pilotes installés :

```
DBI->available_drivers()
```

- Liste des sources de données d'un pilote :

```
DBI->data_sources( $driver_name )
```

`listdsns`: Listage des data sources

```
#!/usr/bin/perl -w

### Load DBI
use DBI;

### Probe DBI for the installed drivers
@drivers = DBI->available_drivers();

### Iterate through the drivers and list the data
### sources for each one
foreach $driver ( @drivers ) {
    print "Driver: $driver\n";
    @dataSources = DBI->data_sources( $driver );
    foreach $dataSource ( @dataSources ) {
        print "\tData Source is $dataSource\n";
    }
    print "\n";
}
exit;
```

Un exemple de sortie de `listdsns`

```
Driver: ADO

Driver: CSV
    Data source is DBI:CSV:f_dir=megaliths
    Data source is DBI:CSV:f_dir=pictish_stones

Driver: ExampleP
    Data Source is dbi:ExampleP:dir=.

Driver: ODBC

Driver: Oracle
     Data Source is DBI:Oracle:DEV

Driver: Proxy

Driver: XBase
    Data Source is dbi:XBase:.
```

Activer le mode Trace de DBI

- Au niveau d'un handle

```
$h->trace($level);
$h->trace($level, $filename);
```

  - Ne concerne que cet handle et ses fils (les handles qu'il crée)
  - Lors de la création d'un handle fils, ce dernier prend le niveau de son père
- Au niveau global (de façon interne à l'application)

```
DBI->trace(...);
```

  - Affecte le niveau minimum de traçage au niveau global
- Au niveau global (par des variables d'environnement)

(extra)

Le niveau 1 de traçage

- Le niveau 1 de traçage : les resultats des méthodes et les numéros de ligne :

```
<- connect= DBI::db=HASH(0xe0abc) at DBI.pm line 356.
<- STORE('PrintError', 1)= 1 at DBI.pm line 382.
<- STORE('AutoCommit', 1)= 1 at DBI.pm line 382.
<- STORE('RaiseError', 1)= 1 at DBI.pm line 382.

<- prepare('UPDATE …')= DBI::st=HASH(0xe1238) at test.pl line 7.
<- prepare('INSERT …')= DBI::st=HASH(0xe1504) at test.pl line 8.
<- execute= '0E0' at test.pl line 9.
<- execute= 1 at test.pl line 10.
<- disconnect= 1 at test.pl line 11.

<- DESTROY= undef
```

Le niveau 2 et supérieur de traçage

- Le niveau 2 de traçage permet de voir les appels paramétrés

```
 -> connect for DBD::ODBC::dr (DBI::dr=HASH(0x13dfec)~0xe14a4
                                '' '' **** HASH(0xe0a10))
 <- connect= DBI::db=HASH(0xe0ab0) at DBI.pm line 356.

 -> STORE for DBD::ODBC::db (DBI::db=HASH(0xe0abc)~INNER 'PrintError'
1)
 <- STORE= 1 at DBI.pm line 382.

 -> prepare for DBD::ODBC::db (DBI::db=HASH(0xe0ab0)~0xe0abc
                                'UPDATE prices SET price=? WHERE
prod=?')
 <- prepare= DBI::st=HASH(0xe1274) at test.pl line 7.
```

- Le niveau 3 de traçage est plus prolixe.

Plan

Introduction

I Mise en oeuvre de DBI

- installation des DBD et /ou du proxy
- utilisation de DBI
- débogage de DBI

II Utilisation de DBI

- Requêtes & autres ordres SQL
- Gestion des erreurs
- Gestion des transactions

III Un exemple : la fnuc

Conclusion

Les requêtes

Le cycle de vie d'une requête comprend 4 états.

- Preparée
- executée
- exploitée (toutes les données résultat sont extraites)
- finie

Preparer la requête

- Utilisation de la méthode `$dbh->prepare( $statement )` qui retourne un statement handle.
- Par exemple :

```
#!/usr/bin/perl -w

### Load the DBI module
use DBI;

### The database handle
$dbh = DBI->connect( "dbi:Oracle:DEV", "username", "password" );

### Prepare the statement handle
$sth = $dbh->prepare( "SELECT id, name FROM megaliths" );

...

exit;
```

Executer la requête

- Par l'appel de la méthode du statement handle `$sth->execute()`.

- Si l'execution de cette méthode se fait avec succès alors les données resultats de la requête sont prêtes à être extraites pour une exploitation dans le script Perl

Extraction de données

- DBI utilise un curseur. Aussi le retrait des données se fait ligne par ligne.
- Une liste peut être utilisée pour le retrait d'une ligne.

```
@row = $sth->fetchrow_array()
```

- Il faut utiliser une boucle appelant `fetchrow_array()` jusqu'a ce toutes les lignes soient extraites

Retrait des données - Exemple 1

```
#/usr/bin/perl -w

### Load the DBI module
use DBI;

### Connect to the database
$dbh = DBI->connect( 'dbi:Oracle:DEV', 'username', 'password' );

### Prepare and execute the statement
$sth = $dbh->prepare( "SELECT name, type FROM megaliths" );
$sth->execute();

### Fetch the data
while ( @row = $sth->fetchrow_array() ) {
    print "Megalith site $row[0] is a $row[1]\n";
  }

### Disconnect from the database
$dbh->disconnect();

exit;
```

Retrait des données - Example 2

```
#/usr/bin/perl -w

### Load the DBI module
use DBI;

### Connect to the database
$dbh = DBI->connect( 'dbi:Oracle:DEV', 'username', 'password' );

### Prepare and execute the statement
$sth = $dbh->prepare( "SELECT name, type FROM megaliths" );
$sth->execute();

### Fetch the data
while ( ( $sitename, $sitetype ) = $sth->fetchrow_array() ) {
    print "Megalith site $sitename is a $sitetype\n";
  }

### Disconnect from the database
$dbh->disconnect();

exit;
```

Retrait des données

- D'autres méthodes de retrait des résultats sont fournies par les statement handle:

```
fetchrow_arrayref()
fetchrow_hashref()
```

Finir la requête

- Lorsque le retrait de toutes les données résultats a été effectué, DBI rend inactif le Statement Handle.
- `La méthode $sth->finish()` permet de rendre inactif le Statement Handle de façon explicite. Cette méthode est rarement utilisé dans des scripts.

Les autres directives SQL

- Les ordres SQL autres que les requêtes peuvent être traités par DBI en appelant `$dbh->do()`
  - `CREATE DATABASE`, `DROP DATABASE (pg)`
  - `CREATE TABLE`, `DROP TABLE`
  - `UPDATE`, `DELETE` et `INSERT`.
- Ces ordres ne nécessitent pas plusieurs états : prepare, execute et finish se succèdent automatiquement.

Ordres paramétrés (1)

- L'ordre SQL préparé contient des emplacements pour des paramètres.
- Cela ressemble à la création à la volée d'un ordre SQL:

```
$sth = $dbh->prepare( "
            SELECT name, location
            FROM megaliths
            WHERE name = " . $dbh->quote( $siteName ) . "
        " );
```

Ordres paramétrés (2)

- Cependant, au lieu de spécifier la variable perl contenant le paramêtre, on utilise un "?" à l'endroit ou le paramètre doit être placé dans l'ordre.
- Par exemple :

```
$sth = $dbh->prepare( "
            SELECT name, location
            FROM megaliths
            WHERE name = ?
        " );
$sth->bind_param( 1, $dbh->quote( 'Avebury' ) );
```

Ordres paramétrés (3)

- L'intérêt dépend du SGBD.

- Pour Oracle, les ordres paramétrés sont utilisées pour être réutilisés lors de la phase de préparation grace à un cache, ce qui peut faire l'économie de l'analyse syntaxique d'ordres paramétrés répétitifs.

Ordres paramétrés (4)

- Un ordre paramétré peut contenir plusieurs paramètres, *e.g.*:

```
$sth = $dbh->prepare( "
            SELECT name, location
            FROM megaliths
            WHERE name = ?
            AND mapref = ?
            AND type LIKE ?
        " );
$sth->bind_param( 1, $dbh->quote( "Avebury” ) );
$sth->bind_param( 2, $dbh->quote( $mapreference ) );
$sth->bind_param( 3, $dbh->quote( "%Stone Circle%” ) );
```

Ordres paramétrés (5)

- Le type des paramètres doit parfois être explicité.

```
### No need for a data type for this value. It's a string.
$sth->bind_param( 1, $dbh->quote( "Avebury” ) );

### This one is obviously a number, so no type again
$sth->bind_param( 2, 21 );

### However, this one is a string but looks like a number
$sth->bind_param( 3, $dbh->quote( 123500 ), { TYPE => SQL_VARCHAR } );

### Alternative short-hand form of the previous statement
$sth->bind_param( 3, $dbh->quote( 123500 ), SQL_VARCHAR );
```

Ordres paramétrés (6)

- L'assignation des paramètres effectifs peut se faire grâce à l'appel de la méthode `$sth->execute()` en utilisant sa version paramétrée.

```
$sth->execute( $dbh->quote( “Avebury” ), 21,  $dbh->quote( 123500 ),
               $dbh->quote( “%Stone Circle%” ) );
```

- Le typage explicite des valeurs se fait à l'aide de `bind_param()`. Il perdure jusqu'a nouvel ordre ou l'inactivation du Statement Handle.

```
$sth->bind_param( 3, $dbh->quote( ‘’ ), SQL_VARCHAR );
$sth->execute( $dbh->quote( 123500 ) );
```

Ordres paramétrés & `$dbh->do()`

- Par défaut, l'appel `$dbh->do()` correspond à `prepare()`, `execute()` and `finish()`
- En cas d'utilisation itérée de cet appel, une optimisation possible consiste à préparer une seule fois l'ordre puis, à l'exécuter autant de fois que nécéssaire dans une boucle avec les paramètres effectifs.

Optimisation de `$dbh->do()`

```
### Prepare the statement handle
$sth = $dbh->prepare( “DELETE FROM megaliths WHERE id = ?” );

### Remove the first 100 rows one-by-one...
$loopCounter = 0;
while ( $loopCounter < 100 ) {
    $sth->execute( $loopCounter );
    $loopCounter++;
  }

$sth->finish();

...
```

- Ceci évite de préparer continuellement le même ordre de modification de schéma ou de données.

La gestion des erreurs

- La méthode manuelle

```
$h->method || die "DBI method failed:
  $DBI::errstr";
$h->method || die "DBI method failed:
  $DBI::errstr";
```

- La méthode automatique

```
$h->{RaiseError} = 1;
$h->method;
$h->method;
```

La gestion automatique des erreurs

- Elle entraine normalement l'arret de l'application.
- On peut capturer l'exception levée lors de l'erreur gràce à `eval { … }:`

```
$h->{RaiseError} = 1;
eval { … $h->method; … };
if ($@) { … }
```

- `eval { … }` capture même les erreurs étrangères à DBI !

Gestion des transactions avec DBI

- en désactivant l'Autocommit (rollback en cas d'erreur),
  `$dbh->{AutoCommit} = 0;`
- en activant la gestion automatique des erreurs,
  `$dbh->{RaiseError} = 1;`
- en faisant appel de manière explicite aux méthodes
  `$dbh->commit;` et `$dbh->rollback;`
- enfin en utilisant la fonction `eval { ... }` autour du code critique, puis en testant `$@.` `rollback`() si vrai sinon `commit`()

Transactions - Exemple

```
$dbh->{AutoCommit} = 0;
$dbh->{RaiseError} = 1;
eval {
    $dbh->method(…);    # assorted DBI calls
    foo(...);           # application code
    $dbh->commit;       # commit the changes
};
if ($@) {
    warn "Transaction aborted because $@";
    $dbh->rollback;
    ...
}
```

Plan

Introduction

I Mise en oeuvre de DBI

- installation des DBD et /ou du proxy
- utilisation de DBI
- débogage de DBI

II Utilisation de DBI

- Requêtes & autres ordres SQL
- Gestion des erreurs
- Gestion des transactions

III Un exemple : la fnuc

Conclusion

Le schéma de la base fnuc

clients(id:int, nom:string, motdepasse:string, CACumul:float)

commandes(id:int, datecom:date, #article:int, #client:int, quantite:int)

stocks(id:int ,#article:int, niveau:int, securite:int)

livres(id:int, titre:string , auteurs:string, resume_url:string,
couverture_url:string, prix:float)

sujets(id:int , libelle:string)

motscles(id:int , libelle:string)

livres_motscles(#book_id:int, #keyword_id:int)

livres_sujets(#book_id:int ,#topic_id:int);

Le module Perl CGI.pm

Michel Dubois

duboism@iu-vannes.fr

Premier exemple

```
#!/usr/bin/perl

use CGI ':standard';
print
     header,
     start_html('Example 1'),
          h1( 'Bonjour le Monde!'),
          "Wow, Je parle HTML!",
          hr,
     end_html;
```

Sortie de cet exemple

```
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML><HEAD>
<TITLE>Example 1</TITLE>
</HEAD>
<BODY>
<H1>Bonjour le Monde!</H1>
Wow, Je parle HTML!
<HR>
</BODY>
</HTML>
```

Objectifs de CGI.pm

Fournir une interface de haut niveau au HTML (balises avec propriétés) et au CGI

- Gérer les méthodes GET et POST indifféremment.
- Cookies
- Upload
- Server Push
- Interactions avec les Frames, Javascripts
- Optimisations : FastCGI et mod_perl

Gestion des méthodes GET & POST

Elle est transparente pour le programmeur !

```
use CGI;
$query = new CGI;
```

- Création d'un objet CGI stockée dans $query.
- Le constructeur traite les paramètres de la requête (quelque soit la méthode d'invocation)

Gestion paramètres CGI

```
$query = new CGI;
```

Pour un paramètre à valeur unique :

```
$value = $query->param('monparam');
```

Pour un paramètre à valeur multiple :

```
@values = $query->param('monparam');
```

Déboguage (1)

- Ligne de commande :

```
my_script.pl name1=value1 name2=value2
my_script.pl name1=value1&name2=value2
```

- Ajout de valeur sur demande (CONTROL D pour finir)

```
my_script.pl
nom=Michel
prenom=DUBOIS
^D
```

Déboguage (2)

La liste des paramètres du CGI :

```
print $query->dump
```

ou

```
print "<H2>DEBUG</H2>\n$query\n";
```

Résultat :

```
<UL>
  <LI>name1
    <UL>
    <LI>value1
    <LI>value2
    </UL>
  <LI>name2
    <UL>
    <LI>value1
    </UL>
  </UL>
```

Variables d'environnement utiles

- path_info()
- path_translated()
- remote_user()
- request_method()
- auth_type()
- raw_cookie()
- cookie()

Environnement CGI

Client

- remote_host()
- user_name()
- user_agent()
- referer()
- accept()
- query_string()
- path_info()

Server

- server_name()
- server_software()
- virtual_host()
- server_port()
- script_name()

La fonction `header()`

- `print header();`

  ```
  Content-type: text/html\r\n\r\n
  ```

- `print header('image/gif');`

  ```
  Content-type: image/gif\r\n\r\n
  ```

- `print header(-type=>'image/gif');`

  ```
  Content-type: image/gif\r\n\r\n
  ```

- ```
  print header(-type=>'image/gif',
               -expires=>'+3d');
  Expires: 22-Aug-1997 08:20:33 GMT
  Content-type: image/gif\r\n\r\n
  ```
  ```

Argument `-status`

- `print header(-status=>'403 Forbidden');`
  - Génère un message "you are not authorized to access this page".
- ```
  print header(
      -status=>'401 Authentication required',
      '-auth-type'=>'Basic'
      );
  ```
  - Entraîne l ’authentification basique du client à l ’aide d ’un dialogue avec le navigateur.
  ```

Autres arguments

```
-type              -content-encoding
-expires           -content-language
-status            -content-transfer-encoding
-location          -date
-server            -auth-type
d 'autres extension de la norme ...
```

```
print header(-type=>'image/gif',
             -status=>'402 Payment Required',
             -cost=>'$0.02');
```

L'entête de la réponse

```
print $query->redirect('http://localhost/');
```

(pour la redirection)

et

```
print $query->header(-type=>'image/gif',
    -status=>'402 Payment Required',
    -expires=>'+3d',
    -cookie=>$my_cookie,
    -Cost=>'$0.02');
```

Les fonctions ou raccourcis HTML

```
$q->start_html(-title=>"Bonjour");
$q->startform;
$q->radio_group(-name=>'genre',
                -values=>['M.','Mme','Mlle'],-default=>'M.');
$q->br;
$q->textfield(-name=>'Nom:',-default=>'Dubois',-size=>50,
              -maxlength=>80);
$q->scrolling_list(-name=>'pays',-values=>['fr','de']);
$q->submit;
$q->endform;
$q->end_html;
```

Les raccourcis HTML

```
h1() strong()  em() pre()
h2() li()      ol() ul()
h3() dl()      dt() dd()
h4() tt()      b()  i()
h5() p()       hr() blockquote()
h6() a()

etcetera
```

Utilisation de ces raccourcis

- `print p();`

  ```
  <P>
  ```

- `print p('This is a paragraph.');`

  ```
  <P>This is a paragraph.</P>
  ```

- `print p('This is ',b(bold),' face.');`

  ```
  This is <B>bold</B> face.
  ```

- `print p({-align=>CENTER},`
  `'This is centered.');`

  ```
  <P ALIGN="CENTER">This is centered.</P>
  ```

- `print hr({-width=>3,-style=>raised});`

  ```
  <HR WIDTH="3" STYLE="raised">
  ```

- `print a({-href=>'index.html'},'Home');`

  ```
  <A HREF="index.html">Home</A>
  ```

L ’imbrication des raccourcis

```
print ol({-type=>'A'},
          li('FORTRAN'),
          li('C')
          li('Java'),
          li('Perl')
      );
```

```
<OL TYPE="A">
      <LI>FORTRAN
      <LI>C
      <LI>Java
      <LI>Perl
</OL>
```

La distributivité de ces raccourcis

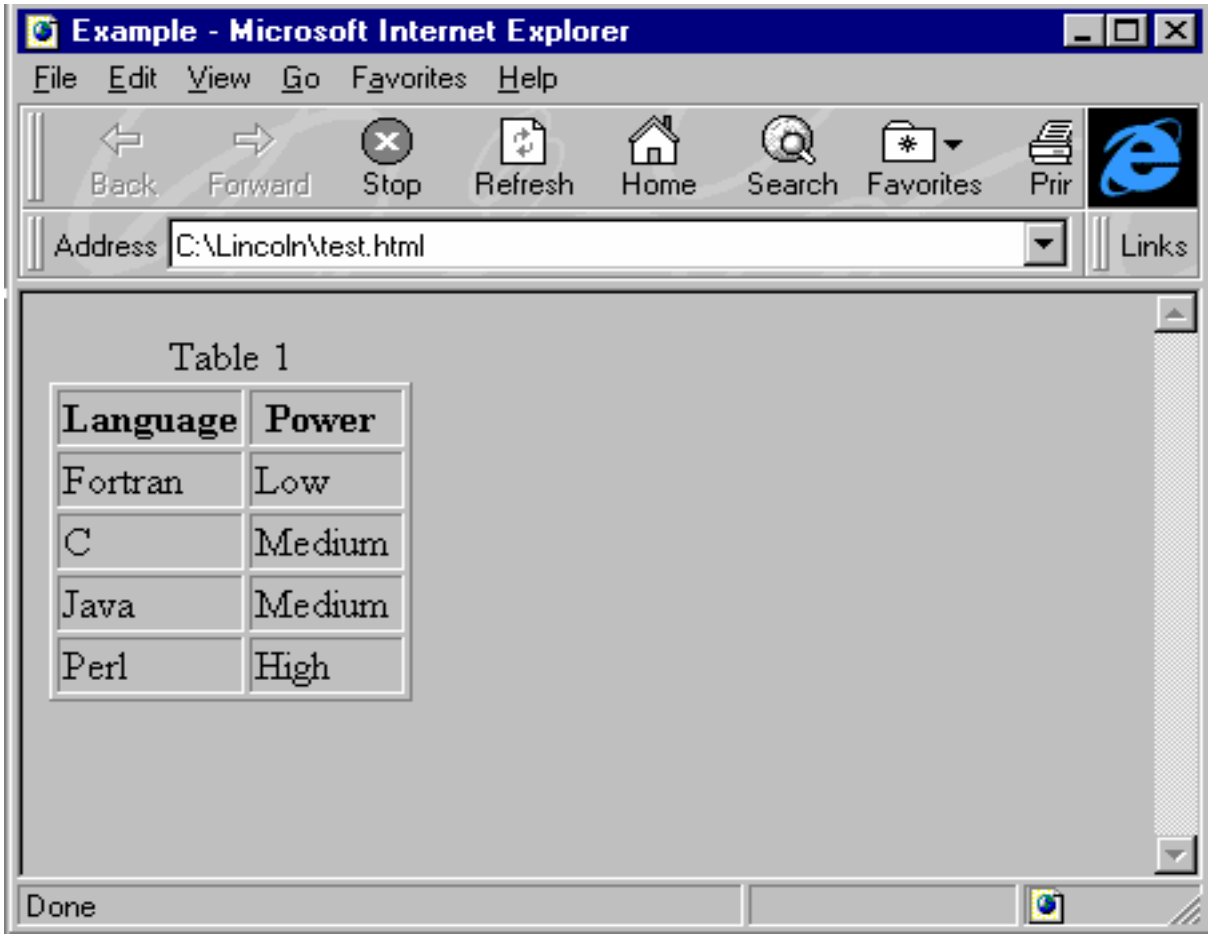
```
@lang = qw(FORTRAN C Java Perl);
print ol({-type=>'A'},li(\@lang));
```

```
<OL TYPE="A">
        <LI>FORTRAN
        <LI>C
        <LI>Java
        <LI>Perl
</OL>
```

Les extensions HTML 3.2

```
use CGI qw(:standard :html3);

print
      table({-border=>''},
         caption('Table 1'),
         TR([th(['Language','Power']),
             td(['Fortran', 'Low']),
             td(['C',       'Medium']),
             td(['Java',    'Medium']),
             td(['Perl',    'High'])
             ]
           )
      );
```

Ce qui donne ...


Example - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Prir
Address C:\Lincoln\test.html
Links
Table 1
Language Power
Fortran Low
C Medium
Java Medium
Perl High
Done

Les sous-ensembles importables de raccourcis

- :cgi
  - protocole CGI
- :form
  - formulaires
- :html2
  - HTML 2.0
- :html3
  - HTML 3.2
- :netscape
  - Les spécificités de Netscape
- :html
  - :html2 + :html3 + :netscape
- :standard
  - :cgi + :form + :html2
- :all
  - La totalité des raccourcis

Raccourcis Non-Standard

- `start_html()`
- `end_html()`
- `start_form()`
- `end_form()`
- Tous les éléments des formulaires

start_html()

- `print start_html(-title=>'ABC DEF');`

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML><HEAD>
<TITLE>ABC DEF</TITLE>
</HEAD>
<BODY>
```

- `print end_html;`

```
</HTML>
```

- Evite une couche de groupe

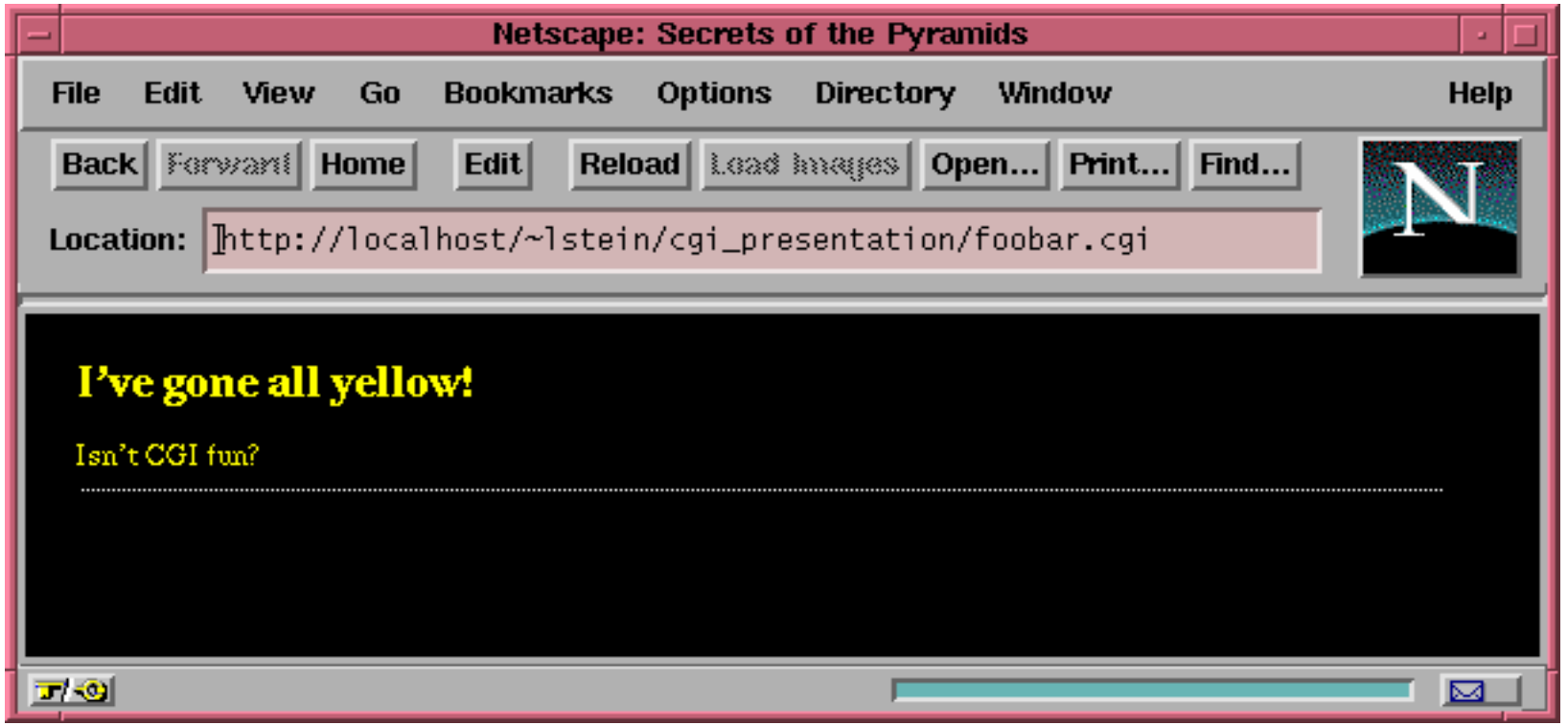
`start_html()` Arguments

- -title
  - titre de la page
- -author
  - adresse e-mail des auteurs
- -meta
  - <META> information, comme des mots-clés
- -bgcolor, -text, -vlink, -alink...
  - couleur de la page
- des arguments pour le JavaScript

Un exemple de `start_html()`

```
#!/usr/bin/perl

use CGI ':standard';
print
   header,
   start_html(-title=>'Secrets of the Pyramids',
              -author=>'fred@capricorn.org',
              -meta=>{'keywords'=>'pharoah secret mummy',
                      'copyright'=>' 1996 King Tut'},
              -text=>'yellow',
              -bgcolor=>'black');

print h1("I've gone all yellow!"),
      "Isn't CGI fun?",
      hr(),
      end_html();
```

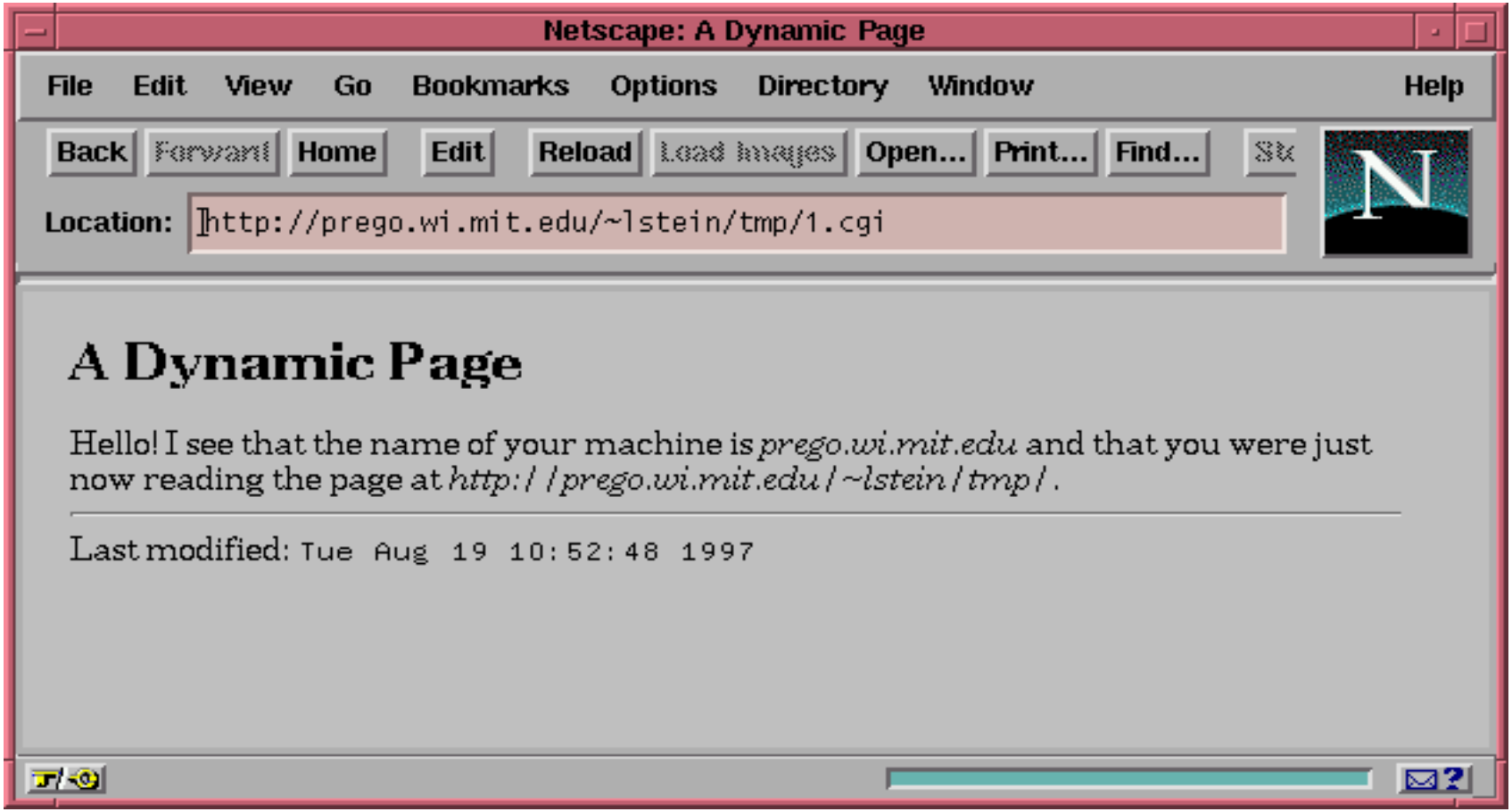
Ce qui donne


Netscape: Secrets of the Pyramids
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Load Images Open... Print... Find...
Location: http://localhost/~lstein/cgi_presentation/foobar.cgi
I've gone all yellow!
Isn't CGI fun?

Une page dynamique ...

```
use CGI ':standard';
$date = localtime();
$host = remote_host();
$referer = referer();
print
   header,
   start_html(-title=>'A Dynamic Page'),
   h1('A Dynamic Page'),
   <<END,end_html();

Hello!  I see that the name of your machine is
<I>$host</I> and that you were just now reading
the page at <I>$referer</I>.
<HR>
Last modified: <TT>$date</TT>
END
```

Ce qui donne


Netscape: A Dynamic Page
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Load Images Open... Print... Find... Stop
Location: http://prego.wi.mit.edu/~lstein/tmp/1.cgi
A Dynamic Page
Hello! I see that the name of your machine is prego.wi.mit.edu and that you were just now reading the page at http://prego.wi.mit.edu/~lstein/tmp/.
Last modified: Tue Aug 19 10:52:48 1997

Les formulaires

- start_form()
- end_form()
- start_multipart_form()
- submit()
- reset()
- defaults()
- textfield()
- textarea()
- password_field()

- popup_menu()
- scrolling_list()
- checkbox()
- checkbox_group()
- radio_group()
- filefield()
- hidden()
- image_button()
- button()

start_form() & end_form()

- `print start_form();`

  `<FORM METHOD=POST>`
- `print end_form();`

  `</FORM>`
- startform() & endform() peuvent être aussi utilisées.
- Evite une couche de regroupement.

Les paramètres de start_form

- -action
  - URL to execute (default: current)
- -method
  - GET or POST
- -encoding
  - *application/x-www-encoded* or *multipart/form-data*
- -frame
- des arguments variés de JavaScript

Arguments d ’un élément de formulaire

- `-name`
  - field name
- `-default` (AKA `-value`)
  - valeur d ’origine du champs
- `-values`
  - pour les champs multivalués
- `-override`
  - Si non-zero, écrasement possible de la valeur par défaut.
- des arguments spécifique à l’élément

textfield()

- ```
  print textfield(-name=>'comments',
                  -default=>'CGI rocks');
  ```

  ```
  <INPUT TYPE="textfield" NAME="comments"
    VALUE="CGI rocks">
  ```

- ```
  print textfield(-name=>'comments',
                  -size=>50,
                  -maxlength=>80);
  ```

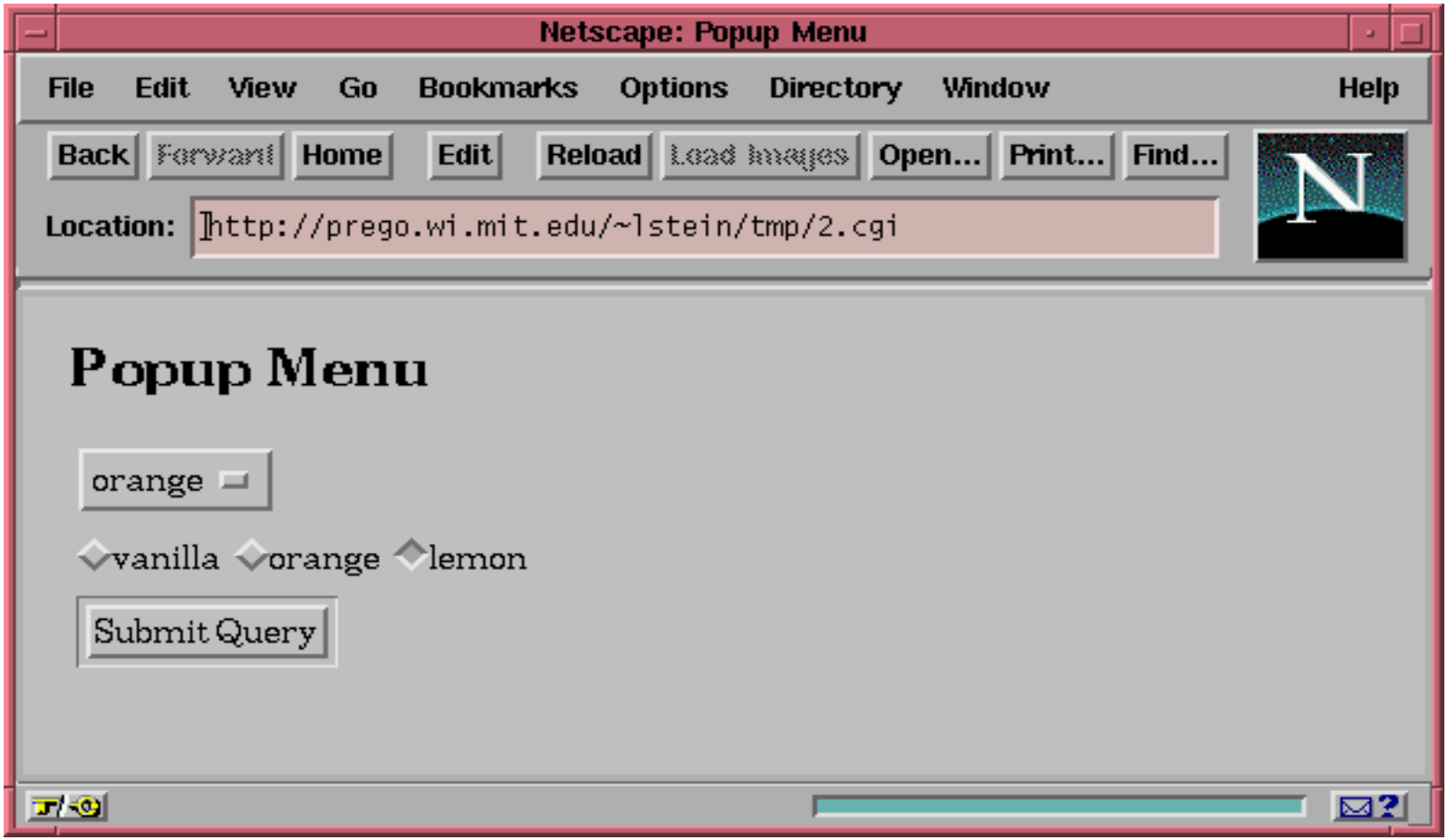
  ```
  <INPUT TYPE="textfield" NAME="comments"
    SIZE=50 MAXLENGTH=80>
  ```

- *idem pour* textarea(), password_field(), hidden() and filefield()

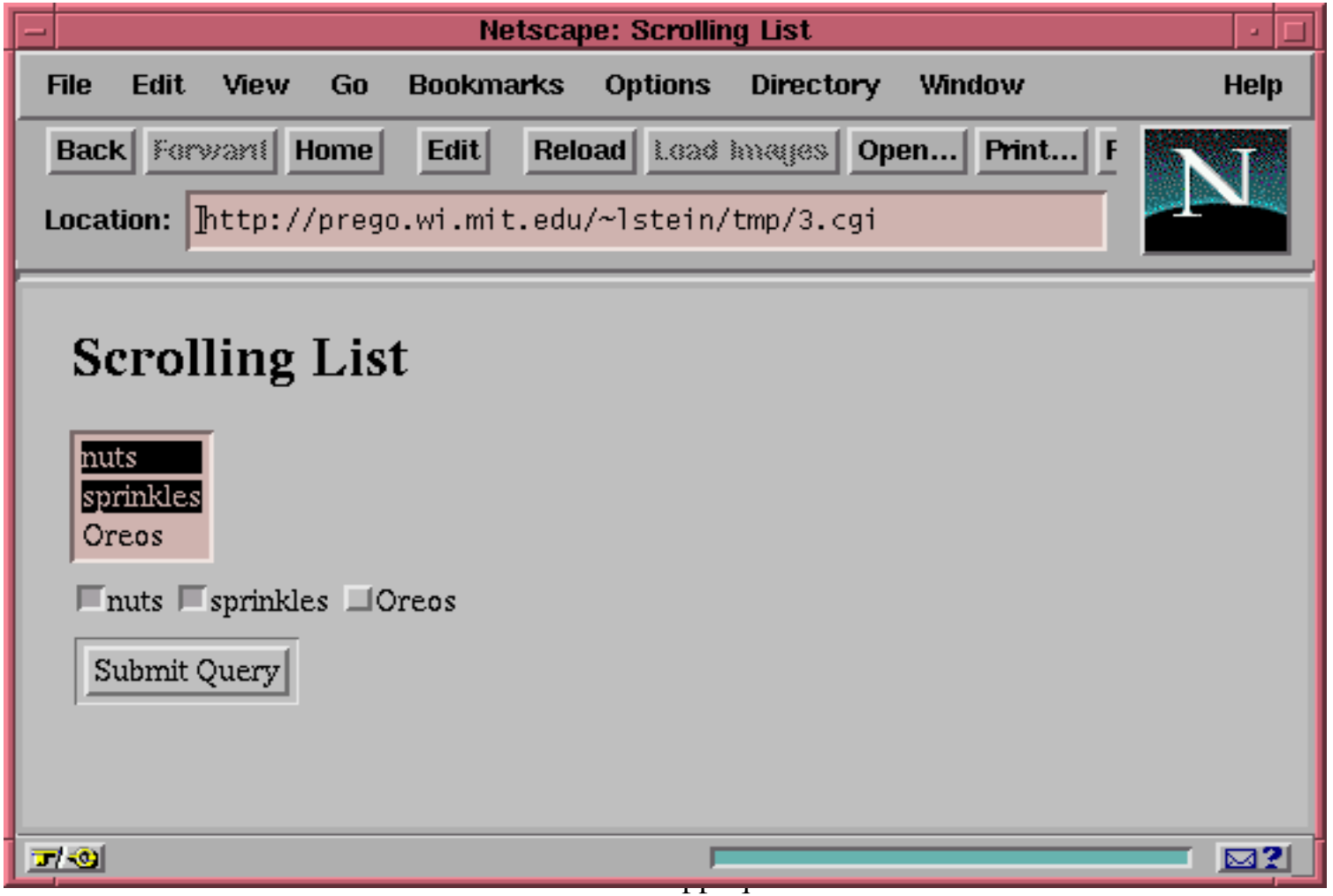
popup_menu()

- ```
  popup_menu(-name=>'flavor',
     -values=>['vanilla','orange',
               'lemon'],
     -default=>'orange');
  ```

```
<SELECT NAME="flavor" SIZE=1>
  <OPTION>vanilla</OPTION>
  <OPTION SELECTED>orange</OPTION>
  <OPTION>lemon</OPTION>
</SELECT>
```

- *idem pour* radio_group()
```

La sortie de popup_menu()



scrolling_list()

- ```
  scrolling_list(-name=>'toppings',
     -values=>['nuts','sprinkles','Oreos'],
     -defaults=>['sprinkles','nuts'],
     -multiple=>1);
  ```

```
<SELECT NAME="toppings" MULTIPLE SIZE=3>
  <OPTION SELECTED>nuts</OPTION>
  <OPTION SELECTED>sprinkles</OPTION>
  <OPTION>Oreos</OPTION>
</SELECT>
```

- *idem pour* checkbox_group()
```

La sortie de scrolling_list()

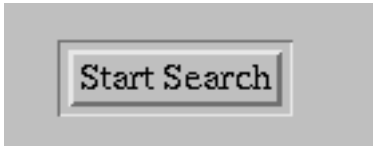


Des labels pour la liste ...

```
• scrolling_list(-name=>'toppings',
      -values=>['nuts','sprinkles','Oreos'],
      -defaults=>['sprinkles','nuts'],
      -multiple=>1,
      -labels=>{nuts=>'Mixed Nuts',
                sprinkles=>'Jimmies',
                Oreos=>'Cookie Bits'}
      );
```

submit(), reset() & defaults()

- `submit(-name=>'search',`
  `-value=>'Start Search');`
  - Envoie les valeurs courantes du formulaire au script.

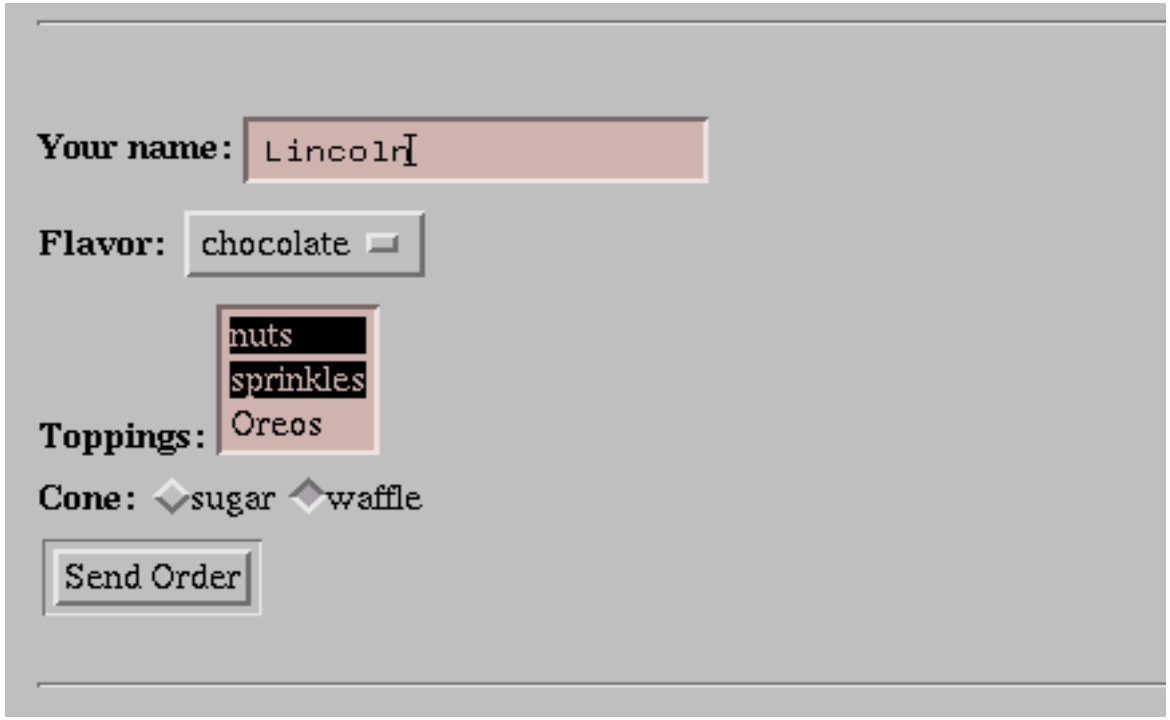


- `reset(-value=>'Reset Form');`
  - Défait tous les changements faits par l 'utilisateur
- `defaults(-value=>'Defaults');`
  - Force les valeurs par défaut.

Un exemple varié

```
sub generate_form {
  print hr, start_form,
   strong('Your name: '),
      textfield(-name=>'customer'),br,
   strong('Flavor: '),
      popup_menu(-name=>'flavor',
        -values=>[qw/chocolate vanilla lemon/]),br,
   strong('Toppings: '),
      scrolling_list(-name=>'toppings',
        -values=>[qw/nuts sprinkles Oreos/]),br,
   strong('Cone: '),
      radio_group(-name=>'cone',-multiple=>1,
        -values=>[qw/sugar waffle/]),br,
   submit(-value=>'Send Order'),
   end_form,hr;
}
```

Ce qui donne ...


Your name: Lincoln
Flavor: chocolate
Toppings: nuts sprinkles Oreos
Cone: sugar waffle
Send Order

Rappel sur le retrait des paramètres

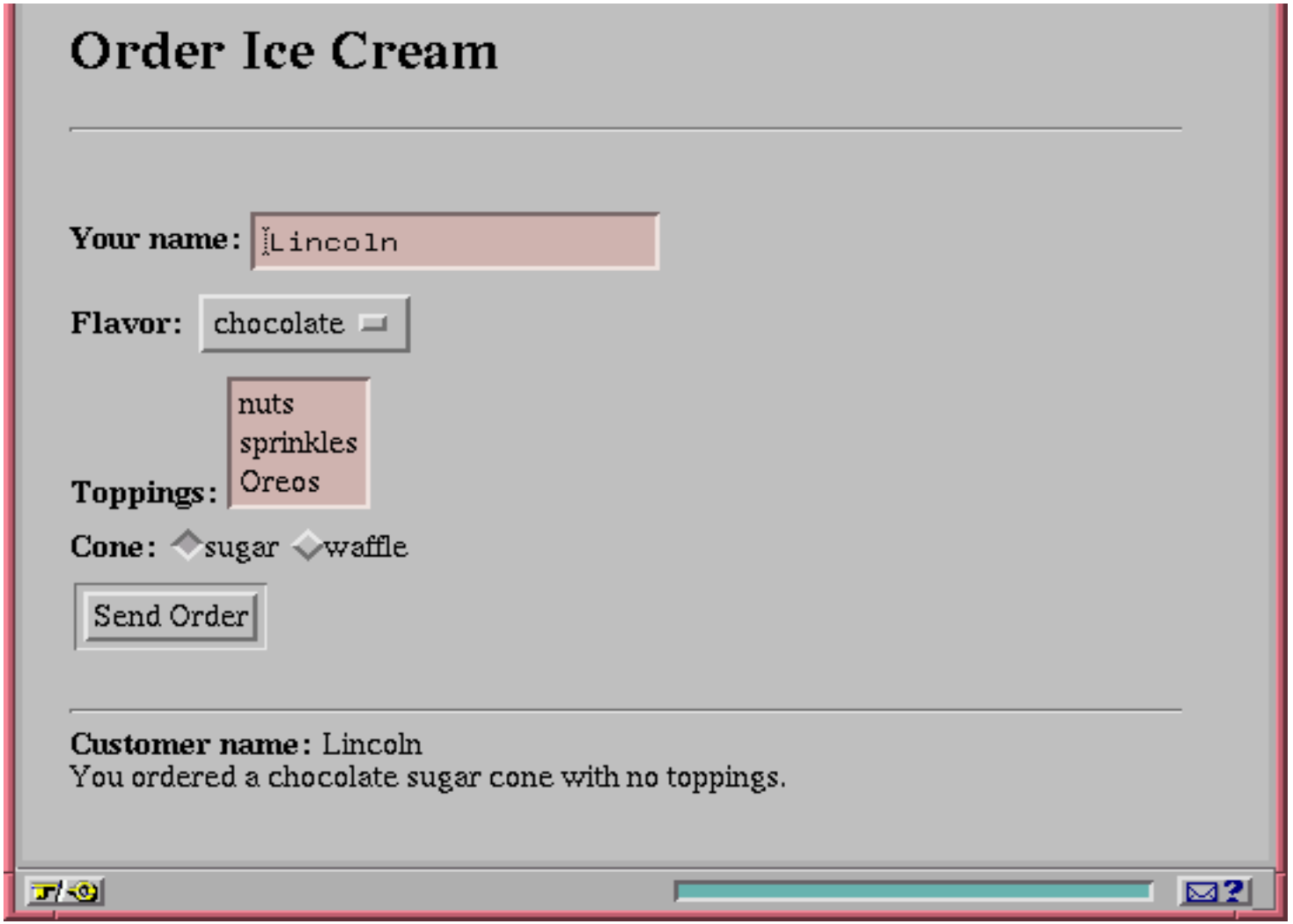
- param(*field_name*);
  - Retourne la valeur du champs nommé
- param() sans argument
  - Retourne un tableau de tous les champs
- Que ce soit pour les champs scalaires ou multivalués

```
$customer = param('customer');
@toppings = param('topping');
```

Un exemple complet

```
#!/usr/bin/perl
use CGI ':standard';

print header,start_html('Order Ice Cream'),
      h1('Order Ice Cream');

generate_form();
print_results if param();
print end_html();

sub print_results {
   my @top = param('toppings');
   print b('Customer name: '),param('customer'),br,
         "You ordered a ",param('flavor'),' ',
         param('cone'),' cone with ';
   print @top ? join(',',@top) : 'no',' toppings';
}
```

Ce qui donne ...


Order Ice Cream
Your name: Lincoln
Flavor: chocolate
Toppings: nuts sprinkles Oreos
Cone: sugar waffle
Send Order
Customer name: Lincoln
You ordered a chocolate sugar cone with no toppings.

Des valeurs collantes pour les champs ...

- Les éléments du formulaire garde en mémoire les valeurs précédantes.

- A la première invocation du script :
  - Aucun champ CGI n'a été passé, donc on utilise les valeurs par défaut.

- Après la première invocation
  - L 'élément n 'utilise pas la valeur par défaut mais la valeur précédemment validée.

- On peut empêcher cette rémanence
  - grâce au paramètre `-override`
  - en précisant la valeur du champs de manière manuelle.

Par exemple

- `textfield(-name=>'customer',`
  `-value=>'Dr. Smith');`

- *http://your.site/cgi-bin/order*

```
<INPUT TYPE="TEXTFIELD" VALUE="Dr. Smith">
```

- *http://your.site/cgi-bin/order?customer=Will%20Robinson*

```
<INPUT TYPE="TEXTFIELD" VALUE="Will Robinson">
```

Remplir les champs manuellement

- `param(-name=>'customer',`
  `-value=>'Dr. Smith');`

- `param(-name=>'toppings',`
  `-value=>['nuts','Oreos']);`

- `param('toppings','nuts','oreos')`

Importation des champs CGI

- import_names(*namespace*)
- Exemple

```
param(-name=>'customer',
      -value=>'Dr. Smith');
param(-name=>'toppings',
      -value=>['nuts','Oreos']);
import_names('Q');
print "Customer = $Q::customer";
 Customer = Dr. Smith
print "Toppings = @Q::toppings";
 Toppings = nuts Oreos
```

Exemple avec le style OO

```
#!/usr/bin/perl
use CGI;
$q = new CGI;
print $q->header(),
  $q->start_html('Object Oriented'),
  $q->h1('Object Oriented');
if ($q->param) {
  print "Your name is ",
   $q->param('name');
}
```

Initialisation de l 'instance CGI

- A partir d 'un tableau associatif

```
$q = new CGI({'dinosaur'=>'barney',
   'color'=>'purple',
   'friends'=>[qw/Jessica George Ann/]});
```

- A partir d 'une chaîne de requête

```
$q=new CGI('dinosaur=barney&color=purple')
```

- A partir d 'un descripteur de fichier

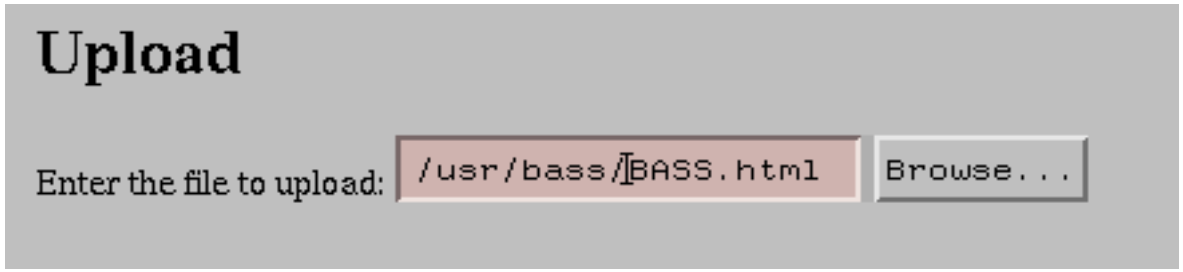
```
open(INPUT,"foobar.txt");
  $q=new CGI(INPUT);
```

Sauvegarde et restauration des instances CGI

```
use CGI;
$q = new CGI;
open (STATE,">save.txt");
$q->save(STATE); # save(\*STATE);
close STATE;

open (IN,"save.txt"):
$r = new CGI(IN); # new CGI(\*IN)
close IN;
```

File Upload

```
print start_multipart_form(),
  "Enter the file to upload: ",
  filefield(-name=>'uploaded file'),
  end_form();
```



Lecture du fichier

- Récupération du descripteur de fichier

```
$filehandle = param('uploaded file');
exit 0 unless $filehandle;
```

- Pour un fichier texte

```
while (<$filehandle>) { do_something(); }
```

- Pour un fichier binaire

```
while (read($filehandle,$scalar,1024)) {
  do_something($scalar);
  }
```

DE PLUS ...

- Le nom du fichier original

```
$filename = param('uploaded file');
```

  – (Le descripteur correspond au nom!)

- Le type MIME du fichier

```
$filename = param('uploaded file');
$info = uploadInfo($filename);
$type = $info->{'Content-Type'};
die "Text files only"
  unless $type =~ /^text/;
```

Les arguments de `cookie()`

- `-name, -value`
  - nom et valeur du cookie
- `-expires`
  - délai d ’expiration du cookie (*+3s, +3m, +3d, +3M, +3y*)
- `-path, -domain`
  - path (*/cgi-bin/database*)
  - domain (*.capricorn.com*)
- `-secure`
  - cookie envoyé si SSL utilisé

`-value` flexible

- Une valeur scalaire

```
$c = cookie(-name=>'color',-value=>'puce';
```

- Un tableau

```
$c = cookie(-name=>'colors',
        -value=>['puce','mauve','red'];
```

- Une table de hash

```
$c = cookie(-name=>'preferences',
        -value=>{color=>'red',
                 font=>'Helvetica',
                 fontsize=>14}
        );
```

Envoi de plusieurs cookies

```
$c1 = cookie(
        -name=>'favorite_color',
        -value=>'puce',
        -expires=>'+3d');
$c2 = cookie(
        -name=>'favorite_shape',
        -value=>'octagon');

print header(-cookie=>[$c1,$c2]);
```

Récupération des cookies existants

- `cookie(`*`name`*`)` pour un cookie identifié

```
$color = cookie('favorite_color');
@color = cookie('colors');
%preferences = cookie('preferences');
```

- `cookie()` sans argument : le tableau de tous les cookies

```
foreach $c (cookie()) {
  process_cookie(cookie($c));
}
```

Les cookies : en résumé

Création d'un cookie :

```
$cookie = $query->cookie(-name=>'sessionID',
                         -value=>'xyzzy',
                         -expires=>'+1h',
                         -path=>'/cgi-bin/database',
                         -domain=>'.capricorn.org',
                         -secure=>0);
```

Passage du cookie dans la réponse :

```
print $query->header(-cookie=>$cookie);
```

Récupération de la valeur d'un cookie passé au script (requête) :

```
%publicites = $query->cookie(-name=>'pub')
```

Cascading Style Sheets

- Permet la séparation contenu/présentation
- On peut définir des styles particuliers

```
print start_html(
  -style=>{-src=>'/style/style1.css'});
```

- Qui s ’appliqueront à des blocks déterminés

```
print h1({-class=>'Fancy'},
  "Style sheets are fun");
```

Définir un Style Sheet

- Utilisation de `start_html()` avec l ’argument `-style`
- `-style` peut pointer sur
  - un scalaire contenant le texte d ’une feuille de style locale
  - Un tableau associatif
    - `-src` => URL de la feuille de style globale
    - `-code` => texte de la feuille de style locale
  - Un style local ayant le même nom qu ’un style global est prioritaire sur le global

Exemple avec CSS

```
$newStyle=<<END;
<!-- P.Tip {
         margin-right: 50pt;
         margin-left: 50pt; }
    P.Alert {
         font-size: 30pt;
         font-family: sans-serif;
         color: red; } -->
END
print start_html(-style=>$newStyle);
print p({-class=>Alert},"Watch out!");
print span({-style=>"Color: magenta"},
  "Here's some magenta text.");
```

Erreur sur le serveur Web

En cas d'erreur non capturée, les messages d'erreur sont envoyés au fichier log du serveur HTTP.

```
use CGI::Carp;
```

permet d'envoyer aussi dans la log les informations temporelles des incidents.

Pour envoyer des info dans la log :

```
print STDERR 'vnom="',$vnom,'"';
carp 'vnom="'. $vnom . '"'; #le temps !
```

Notification d'erreur ...

- Renvoyer le plus tôt possible le Content-type
- Notifier l'erreur à l'utilisateur ... Et au Webmestre

Par exemple :

```
Open (FILE, "$filename") or &log_and_die("cannot open $filename
   ");
...
Sub log_and_die {
   my $string=shift;
   print $query->start_html(-title=>"Erreur");
   print "<P> Malheureusement ce programme a rencontre une
   erreur. Mais elle a ete notifiee au webmestre !</P>";
   print $query->end_html();
   die $string;
}
```

Notification d'erreur ...

- Envoyer les erreurs fatales non seulement à la log mais aussi vers l'utilisateur de manière automatique !

```
use CGI::Carp qw(fatalsToBrowser);
```

- Changer le message par défaut :

```
use CGI::Carp qw(fatalsToBrowser set_message);
    BEGIN {
       sub handle_errors {
          my $msg = shift;
          print "<h1>Oh gosh</h1>";
          print "Got an error: $msg";
      }
      set_message(\&handle_errors);
    }
```

Paramétrages de scripts CGI (1)

- Le nombre de scripts CGI formant une application web est souvent fonction de l'interaction avec le client et de l'ergonomie voulue : (définition des écrans, IHM HTML ...)
- Les paramètres globaux à l'application n'évoluent que très peu au cours de l'interaction. Lorsqu'ils évoluent, ce changement doit être pris en compte par tous les scripts.
- L'utilisation d'un package spécifique à la configuration peut être une solution.

Paramétrages de scripts CGI (2)

```
Package MaConfiguration;
use strict;
use vars qw(%conf);
my $filebase="/home/helios/prof/duboism/apache";
my $urlbase="http://bilbo:8080";
%conf=
   (
       dir=>{ cgi=>"$filebase/cgi-bin",
              docs=>"$filebase/htdocs",
              img=>"$filebase/htdocs/images",},
       url=>{ cgi=>"$urlbase/perl",
              docs=>"$urlbase/",
              img=>"$urlbase/images",},
       dbi=>{ dsn=>'dbi:Pg:dbname=mdtest;host=bilbo;port=5432',
              user=>'',
              passwd=>'',},
   );
1;
```

Paramétrages de scripts CGI (3)

Dans tout script CGI devant avoir accès à ces variables

- avec obligation de qualifier de manière totale la table de hash :

```
use strict;
use MaConfiguration;
...
 Print "Mes documents sont accessibles a
   $MaConfigiration::conf{url}{docs}\n";
```

- avec la possibilité d'utiliser un alias :

```
use strict;
use MaConfiguration;
use vars qw(%conf);
*c=\%MaConfiguration::conf;
...
 Print "Mes documents sont accessibles a $c{url}{docs}\n"
```

Quelques conseils et liens

- Les conseils :
  - "-w" pour les avertissements
  - "-T" : mode d'entachement (sécurité)
  - "use strict;" : des déclarations plus formelles pour une vérification plus efficace.
  - "use diagnostics;" : le compilateur est plus prolixe
- Les liens :
  - La WWW Security FAQ:
    - http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html
  - La Perl CGI FAQ:
    - http://www.perl.com/cgi-bin/pace/pub/doc/FAQs/cgi/perl-cgi-faq.html

Apache.pm
l'API de mod_perl

Michel Dubois

duboism@iu-vannes.fr

Objectifs mod_perl

- Réduire l'invocation de la machine virtuelle de Perl en la préchargeant
- Précompiler les modules les plus fréquemment utilisés
- Avoir en cache les scripts les plus frequemment utilisés
- Avoir des connexions aux bases de données persistantes
- Accéder aux API d'Apache

Installation de mod_perl

I - Prérequis

- Installer Perl avec LWP
- Fixer la machine virtuelle Perl pour Apache

II - Installation du module

- Installer MOD_PERL
- Installer APACHE

III- Finitions

- Configurer APACHE

Installation de Perl avec LWP

```
#!/bin/tcsh
set MDRoot="/devlpt/bilbo/etud/fcduia/WebDev/local"
set MDSrc="/devlpt/bilbo/etud/fcduia/WebDev/src"
setenv PATH "/devlpt/bilbo/etud/fcduia/WebDev/local/perl/bin:$PATH "
setenv PERL5LIB "/devlpt/bilbo/etud/fcduia/WebDev/local/perl:$HOME/perl
   "
cd $MDSrc
tar -zxvf perl5.005_03.tar.gz
cd perl5.005_03
rm -f config.sh Policy.sh
sh Configure -Dprefix=$MDRoot/perl \
-Dinstallprefix=$MDRoot/perl \
-des
make
make test
make install
perl -MCPAN -install LWP
```

Fixer la machine virtuelle Perl d'Apache

```
#!/bin/tcsh
set MDRoot="/devlpt/bilbo/etud/fcduia/WebDev/local"
set MDSrc="/devlpt/bilbo/etud/fcduia/WebDev/src"
cd $MDSrc
wget http://www.apache.org/dist/apache_1.3.6.tar.gz
tar -zxvf apache_1.3.6.tar.gz
cd apache_1.3.6
./configure --prefix=$MDRoot/apache \
--enable-module=info \
--enable-module=mime_magic \
--enable-module=headers \
--enable-module=speling \
--enable-module=unique_id \
--enable-module=digest \
--enable-module=usertrack \
--with-perl=$MDRoot/perl/bin/perl
```

Installer MOD_PERL (manuel)

```
#!/bin/tcsh
set MDRoot="/devlpt/bilbo/etud/fcduia/WebDev/local"
set MDSrc="/devlpt/bilbo/etud/fcduia/WebDev/src"
setenv PATH "/devlpt/bilbo/etud/fcduia/WebDev/local/perl/bin:$PATH "
setenv PERL5LIB "/devlpt/bilbo/etud/fcduia/WebDev/local/perl:$HOME/perl
cd $MDSrc
tar -zxvf mod_perl-1.21.tar.gz
cd mod_perl-1.21
perl Makefile.PL PREFIX=$MDRoot/perl APACHE_SRC=$MDSrc/apache_1.3.6 \
DO_HTTPD=1 \
USE_APACI=1 \
PREP_HTTPD=1 \
PERL_SSI=1 \
PERL_SECTION=1 \
EVERYTHING=1
make
make test
make install
```

Installer MOD_PERL (CPAN)

```
duboism@pc-b020-01:~/PerlTP 72 > perl -MCPAN -eshell
cpan shell -- CPAN exploration and modules installation (v1.50)
ReadLine support enabled
cpan> install Bundle::Apache


...

Demande du répertoire des sources d'Apache
Demande du répertoire d'installation d'Apache

...

Par contre l'installation d'Apache n'est pas automatique
```

Installer Apache avec mod_perl

```
#!/bin/tcsh
set MDRoot="/devlpt/bilbo/etud/fcduia/WebDev/local"
set MDSrc="/devlpt/bilbo/etud/fcduia/WebDev/src"
cd $MDSrc/apache_1.3.6
./configure --prefix=$MDRoot/apache \
--enable-module=info \
--enable-module=mime_magic \
--enable-module=headers \
--enable-module=speling \
--enable-module=unique_id \
--enable-module=digest \
--enable-module=usertrack \
--with-perl=$MDRoot/perl/bin/perl \
--activate-module=src/modules/perl/libperl.a
cd ..
```

Le préchargement /compilation

Httpd.conf: Avant tout autre directive mod_perl

```
<IfModule mod_perl.c>
   Perlrequire /home/helios/prof/duboism/apache/conf/startup.pl
</IfModule>
```

/home/helios/prof/duboism/apache/conf/startup.pl:

```
use strict;
$ENV{GATEWAY_INTERFACE}=~ /^CGI-Perl/ or die "La CGI n'est pas perl !";
use Apache::Registry ();
use Apache::Status ();
use Apache::DBI (); # use DBI () n'est pas necessaire !;
# Des avertissements plus complets
use Carp();
$SIG{__WARN__}=\&Carp::cluck;
#préchargement et précompilation
use CGI qw(-compile :all);
```

Apache::Status

```
Httpd.conf:
<IfModule mod_perl.c>
   <Location /perl-status/>
        SetHandler perl-script
        PerlHandler Apache::Status
        order deny,allow
        allow from all
   </Location>
</IfModule>
```

- Le test est immédiat (Location du navigateur)

Apache::Registry

```
Httpd.conf:
<IfModule mod_perl.c>
   Alias /perl/ "/home/helios/prof/duboism/apache/cgi-bin/"
   <Location /perl/>
       SetHandler perl-script
       PerlHandler Apache::Registry
       Options ExecCGI
       allow from all
       PerlSendHeader On
   </Location>
</IfModule>
```

- Tout script CGI (Perl sinon danger !) invoqué par une URL comportant /perl/ (ex:"http://localhost/perl/showalltopics.pl") au lieu de /cgi-bin/ (ex:"http://localhost/cgi-bin/showalltopics.pl") sera géré par le module Apache::Registry.
- Apache::Registry offre les performance de mod_perl à un script non écrit pour mod_perl !

Apache::DBI

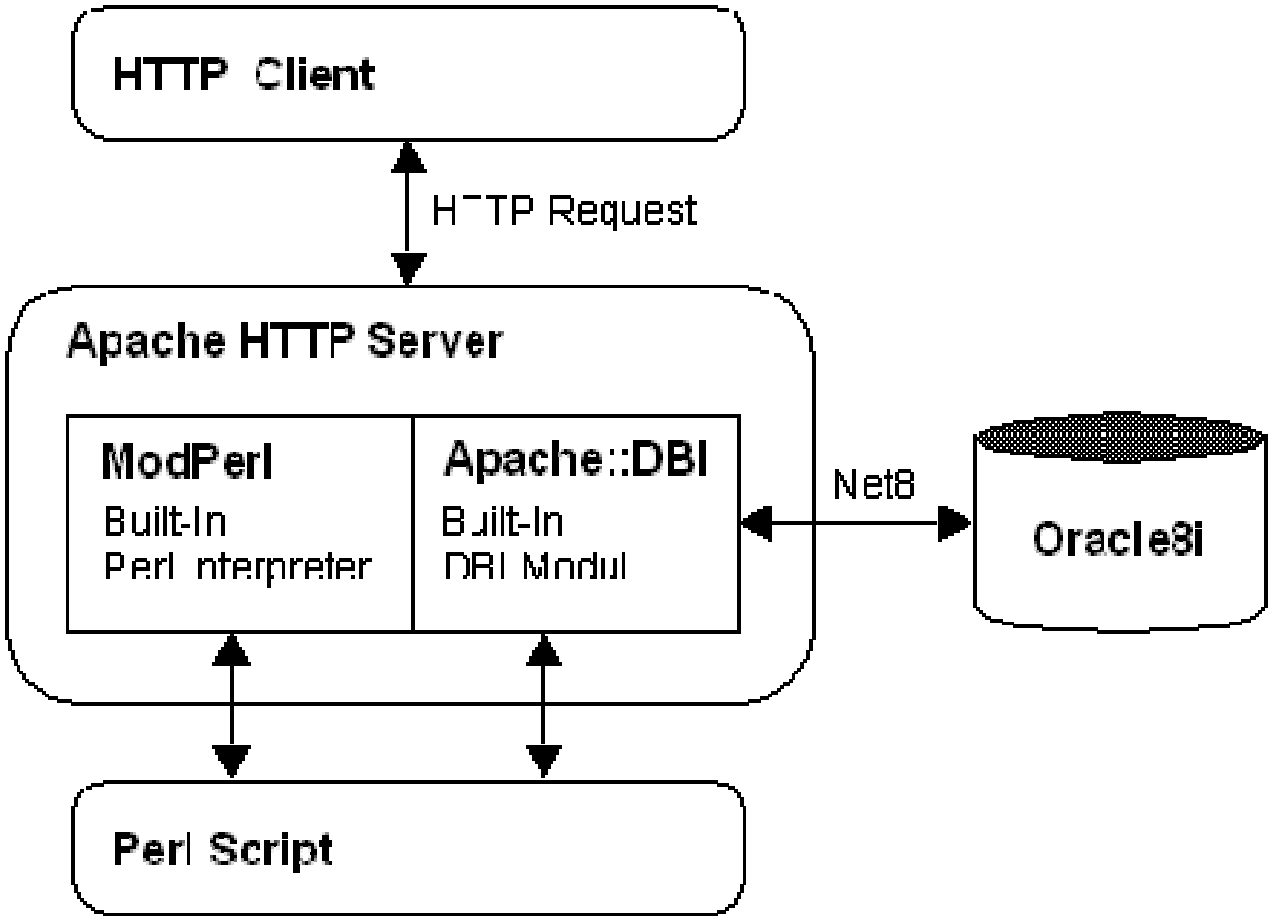
- Il faut charger ce module avant tout autre module utilisant DBI.

Httpd.conf:

```
<IfModule mod_perl.c>
  PerlModule Apache::DBI
</IfModule>
```

- Une connexion à Oracle peut se décomposer en plusieurs phases :
  - créer une structure de données côté client pour la connexion
  - récupérer l'alias Oracle dans un fichier
  - récupérer le nom de l'hôte d'Oracle
  - ouvrir une socket vers ce serveur de données
  - construire une structure de données coté serveur pour la connexion.
- Un programmeur naïf va faire une connexion/déconnexion pour chaque invocation du CGI. Apache::DBI intercepte les appels de connect() et disconnect() de DBI. Il met la connexion dans un cache à la première demande , la réutilise après et ignore les ordres de déconnexions. C'est à la mort du processus fils d'Apache que les déconnexions se feront !
- Bref aucune modification du code n'est nécessaire pour un gain non négligeable !

MOD_PERL / CGI & DBI



Ecrire un module pour mod_perl

```
package Apache::MonModule;
# fichier : Apache/MonModule.pm
use Apache::Constants ':common';
sub handler {
   my $r=shift;
   $r->content_type('text/html');
   $r->send_http_header;
   return OK if $r->header_only;
   my $host=$r->get_remote_host;
   $r->print(<<END);
<HTML>
<HEADER><TITLE> Bienvenue sur MonModule </TITLE></HEADER>
<BODY><H1> Cher $host, </H1> Bon courage, mod_perl est facile !</BODY>
</HTML>
END
   return OK;
}
1;
```

Utiliser un module pour mod_perl

- Il faut déployer le module MonModule dans la hiérarchie des classes

- Il faut configurer Httpd.conf:

```
<IfModule mod_perl.c>
   PerlModule Apache::MonModule
   <Location /monmodule/>
       SetHandler perl-script
       PerlHandler Apache::MonModule
   </Location>
</IfModule>
```

- Il faut redémarrer Apache. S'il y a une erreur dans le module ou si Apache n'arrive pas à le trouver, Apache refusera de démarrer.

Apache::Session (1)

- API faite pour le maintien de l'état de la session. Il s'agit d'identifier les utilisateurs.

- `Apache::Session::DBI` stocke dans une base de données les informations particulières à un utilisateur. Pour cela elle utilise une table à la structure suivante:

```
CREATE TABLE sessions (
   id char(16),
   length int(11),
   a_session text
);
```

Apache::Session (2)

Dans un module perl pour Apache :

```
use Apache::Session::DBI;
```

#Au début du code, on retire l'identifiant de la session par la lecture du cookie

```
my $id=$->header_in('Cookie');
$id=~s|SESSION_ID=(\w*)|$1|;
```

#Il faut lier la table de hash %session à la table session de la base de données.

```
my %session;
tie %session,'Apache::Session::DBI', $id, {
DataSource=>'dbi:Pg:dbname=mdtest;host=bilbo;port=5432',
UserName=>'',
Password=>''};
```

#pour une nouvelle session, `tie %session,'Apache::Session::DBI',undef,...`

#Récupération des variables de sessions : à partir de ce moment, toute variable de session scalaire clef peut être atteinte par :

```
my $session{clef};
```

#Toute table de hash de session peut être récupérée par :

```
my %ses_hash=defined $session{clef} ? %{$session{clef}}:();
```

Apache::Session (3)

Affectation (dans l'entête de la réponse HTTP) :

#Une variable scalaire de session peut subir une affectation :

```
$session{clef}="valeur";
```

#Pour stocker une table de hash %hash, on crée une référence et on l'affecte à la variable de session :

```
$session{clef}=\%hash;
```

Pour créer et envoyer un nouveau cookie au client :

```
my $session_cookie = "SESSION_ID=$session{_session_id};";
$r->header_out("Set-Cookie" => $session_cookie);
```

```
#pour invalider une session :
tied(%session)->delete;
```