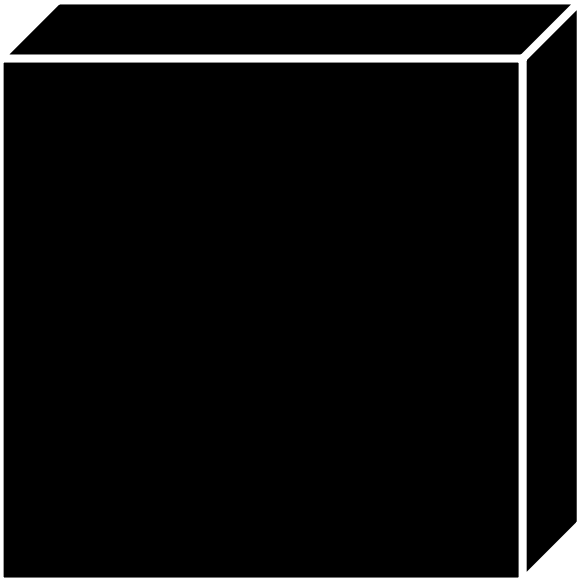
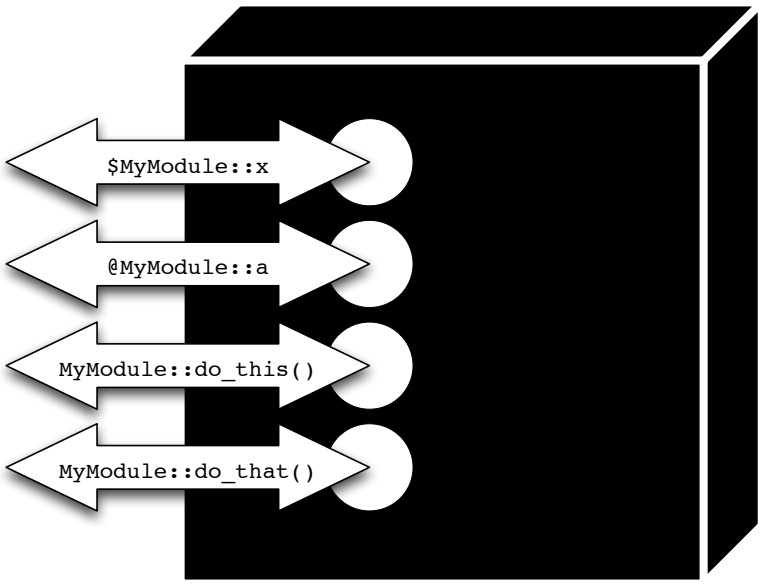
Modules

What is a module?



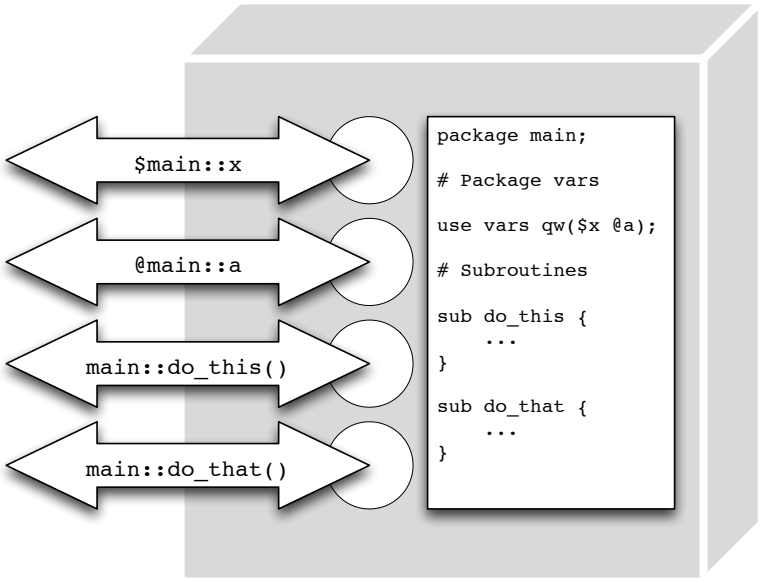
MyModule

What is a module?



MyModule

What is a module?



main

What's the point?

Protect one part of your program from another

Make your program easier to understand

Code reuse (laziness is a virtue!)

Objects and classes

A thumbnail sketch

Class

A module you can use to create objects

Just **tell** it to create one

```
use Some::Module;
my $object = Some::Module->new;
```


Object

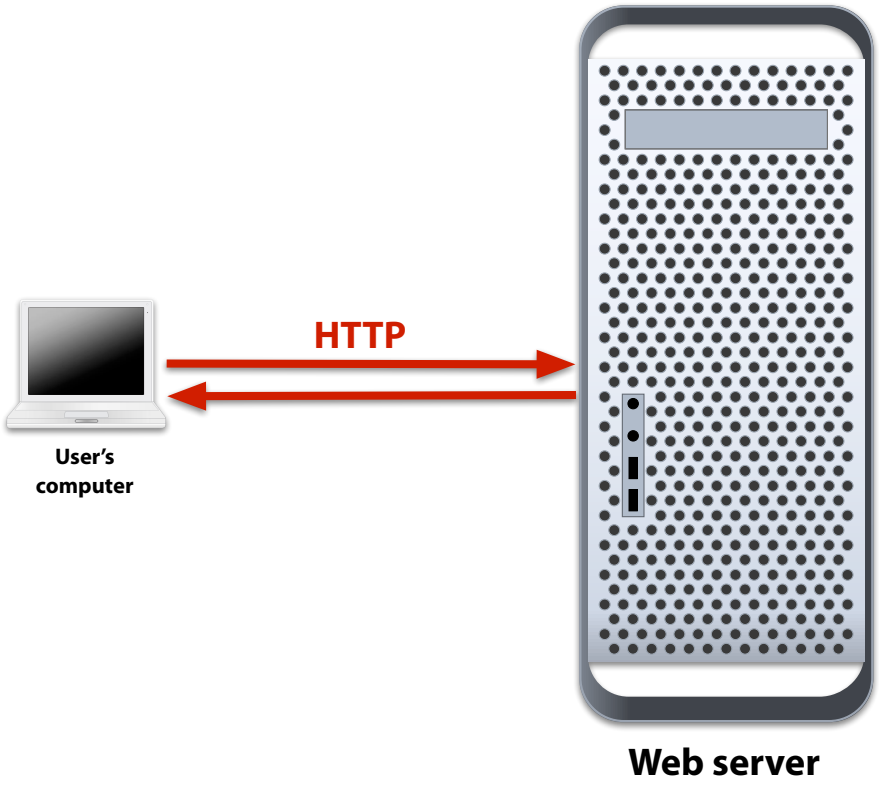
Another black box

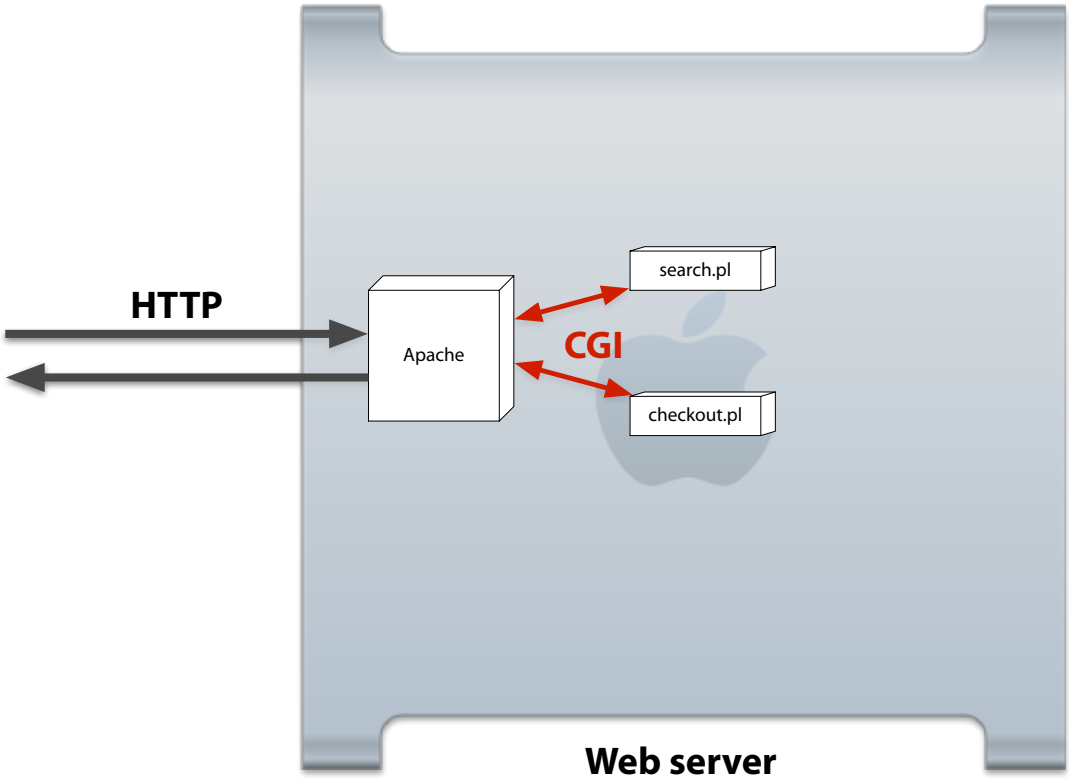
Has **properties** you can **tell** it to get or set

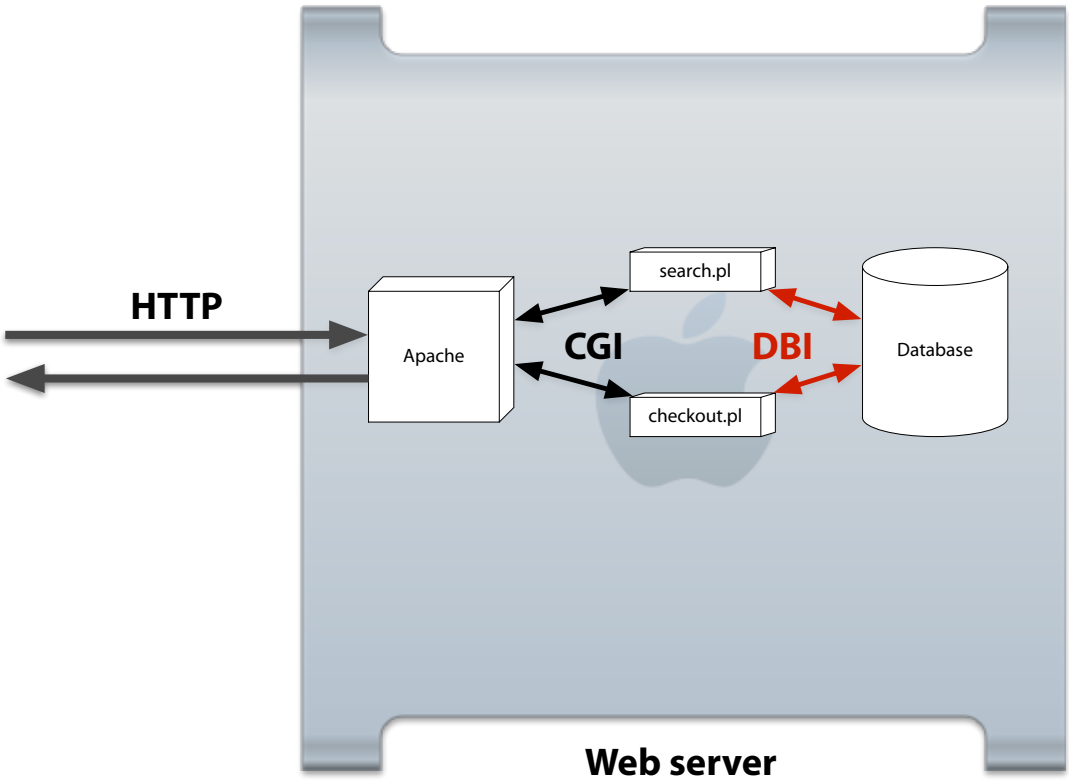
Has **behavior** – if you **tell** it do something, it reacts in a certain way

```
use Some::Module;
my $object = Some::Module->new;
print $object->name;
$object->jump('up');
```

HTTP, CGI, and DBI



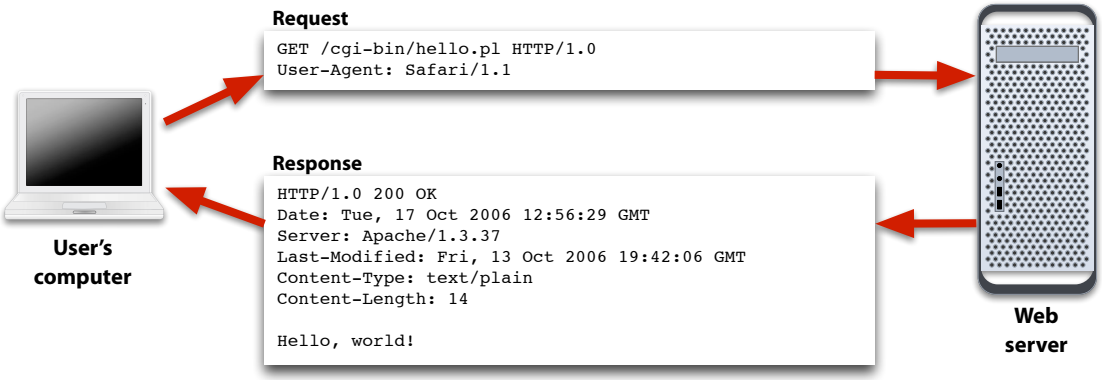

HTTP
Apache
search.pl
CGI
checkout.pl
Web server


HTTP
Apache
search.pl
CGI
DBI
Database
checkout.pl
Web server

HTTP

2 min. overview

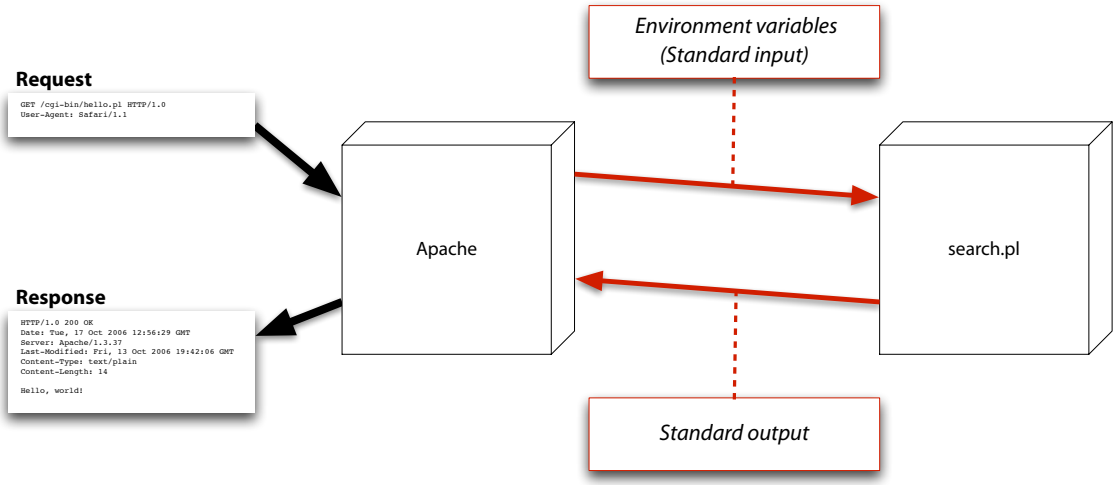
HTTP request/response exchange



CGI

Common Gateway Interface

CGI exchange



hello.html – the user interface

http://eowyn.simmons.edu/ceperl/hello.html

hello.pl – the CGI program

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

The program relies upon the CGI module. This statement checks to make sure it's available and – if it is – gives it an opportunity to perform any initialization it requires.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

We use a generic greeting by default.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

Ask the CGI module to create a new object containing the data that the web server program has passed to us.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

Look for a "name" parameter. This is the text that the user typed into the HTML input field named "name".

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

Don't assume that there was such a field! The "user" could be a malicious program in disguise.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

Check to see if the user left the "name" field blank.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

OK, use the name they supplied.

Normally, we would be careful to check to make sure the value supplied wasn't malformed, but we don't bother here because we're only going to send the name back to the user.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

The content that we're generating on the web server program's behalf is plain text.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

A blank line separates the HTTP header from the content.

hello.pl

```
use CGI;

my $user_name = 'whoever you are';

my $query = CGI->new;

my $name_supplied = $query->param('name');
if (defined $name_supplied
        and $name_supplied ne '') {
    $user_name = $name_supplied;
}

print "Content-Type: text/plain\n";
print "\n";
print "Hello, $user_name!\n";
```

Send our greeting. The web server program will take care of all other nitpicky details.

hello.pl – the result

```
Hello, Yolanda!
```


hello.pl – try it out

http://eowyn.simmons.edu/ceperl/cgi-bin/hello.pl

http://eowyn.simmons.edu/ceperl/cgi-bin/hello.pl?name=Yolanda

http://eowyn.simmons.edu/ceperl/cgi-bin/badger/hello.pl

http://eowyn.simmons.edu/ceperl/cgi-bin/loris/hello.pl

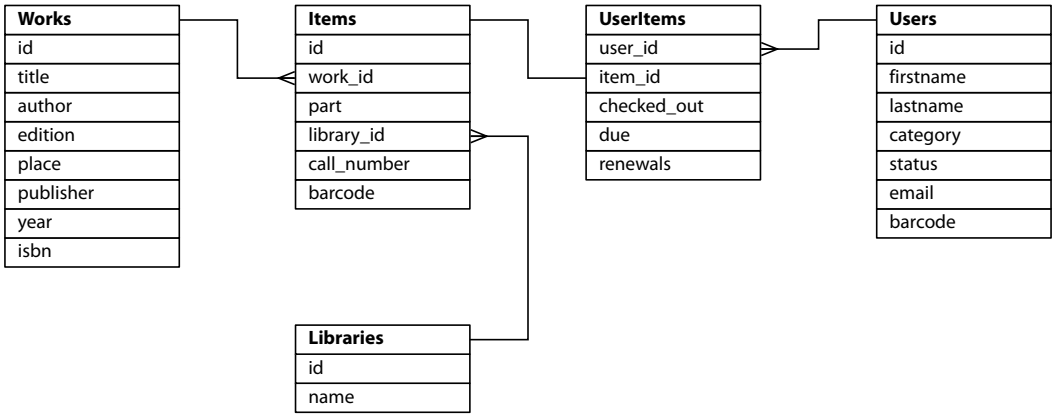
— *etc.*

KidCat

Databases

SQL and RDBMSes in a nutshell

Tables



Columns and rows

id	firstname	lastname	category	status	email	barcode
3	Yolanda	Ipswich	S	A	yoyo@mama.org	5019
1	Xerxes	Axelrod	T	A	xaxelrod@goatisland.edu	3928
6	Wilhelmina	Happenstantz	S	A	hellion604@yahoo.com	1085

DBI

Perl database interface

The DBI mantra

Connect

Prepare

Execute

Fetch

Error checking!

Connect

```
$dbh = DBI->connect(
    $data_source,
    $user, $password
);
```


Prepare

```
$dbh = DBI->connect(
    $data_source,
    $user, $password
);
$sth = $dbh->prepare($query);
```

Execute

```
$dbh = DBI->connect(
    $data_source,
    $user, $password
);
$sth = $dbh->prepare($query);
$sth->execute;
```


Fetch

```
$dbh = DBI->connect(
    $data_source,
    $user, $password
);
$sth = $dbh->prepare($query);
$sth->execute;
while (@row = $sth->fetchrow_array)
{
    ...
}
```

Error checking

```
$dbh = DBI->connect(
    $data_source,
    $user, $password
) or die;
$sth = $dbh->prepare($query)
  or die;
$sth->execute or die;
while (@row = $sth->fetchrow_array)
{
    ...
}
die if $DBI::err;
```


Error checking simplified

```
$dbh = DBI->connect(
    $data_source,
    $user, $password,
    { 'RaiseError' => 1 }
);
$sth = $dbh->prepare($query);
$sth->execute;
while (@row = $sth->fetchrow_array)
{
    ...
}
```