

Bioinformatics Programming 2013

# Perl – Modules

**Chia-Lang Hsu (許家郎)**

Department of Life Science,  
National Taiwan University

# What is a module?

- A module is a library of Perl code that can be included in your Perl program.
- When you include a Perl module in a program, the functionality of that module is available for you to use inside your own program.

# Perl Modules

- **require** or **use** statements both pull a module into your program
- **require** loads modules at runtime, with a check to avoid the redundant loading of a given module
- **use** is like **require**, with two added properties: compile-time loading and automatic importing
- The required file extension for a Perl module is **".pm"**
- The full path to the file depends on your include path, which is stored in the global **@INC** variable `use lib "$path";`
- If the module name itself contains one or more double colons, these are translated into your system's directory separator
  - `File::Find` module resides in the file `File/Find.pm`

# CPAN (<http://www.cpan.org/>)



## Comprehensive Perl Archive Network

YOU CAN NEVER HAVE TOO MANY PERL MODULES

[Home](#) [Modules](#) [Ports](#) [Perl Source](#) [FAQ](#) [Mirrors](#)

Search:

### Welcome to CPAN

The Comprehensive Perl Archive Network (CPAN) currently has [123,943 Perl modules](#) in 28,120 distributions, written by 10,860 authors, [mirrored](#) on 271 servers.

The archive has been online since October 1995 and is constantly growing.

### Search CPAN via

- [metacpan.org](http://metacpan.org)
- [search.cpan.org](http://search.cpan.org)

### Recent Uploads

- [Template-Twostep-1.02](#)
- [Yeb-0.012](#)
- [Template-Twostep-0](#)
- [Gedcom-1.19](#)
- [Time-HiRes-1.9726](#)
- [ExtUtils-MakeMaker-6.73\\_11](#)
- [Devel-MAT-0.02](#)
- [App-highlight-0.14](#)
- [Alien-SDL2-0.002](#)
- [App-Followme-0.78](#)
- [more...](#)

### Getting Started

- [Installing Perl Modules](#)
- [Learn Perl](#)

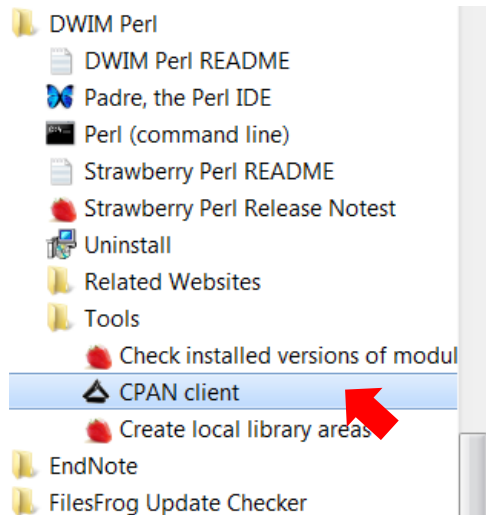
### Perl Resources

- [The Perl Programming language](#)
- [Perl Documentation](#)
- [Mailing Lists](#)
- [Perl FAQ](#)
- [Scripts Repository](#)

Yours Eclectically, The Self-Appointed Master Librarians (OOK!) of the CPAN.  
© 1995-2010 Jarkko Hietaniemi. © 2011-2013 [Perl.org](http://Perl.org). All rights reserved. [Disclaimer](#).

Master mirror hosted by [YellowBot](#)®

# How to install Perl modules



```
cpan shell -- CPAN exploration and modules
Enter 'h' for help.
```

```
cpan>
```

```
cpan shell -- CPAN exploration and modules
Enter 'h' for help.
```

```
cpan> install List::Compare
```

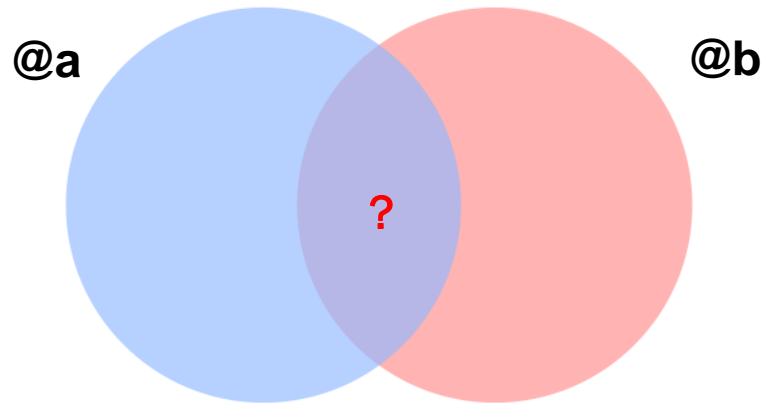
```
Running make install
Installing D:\Dwimperl\perl\site\lib>List\Compare.pm
Installing D:\Dwimperl\perl\site\lib>List\Compare\Functional.pm
Installing D:\Dwimperl\perl\site\lib>List\Compare\Base\_Auxiliary.pm
Installing D:\Dwimperl\perl\site\lib>List\Compare\Base\_Engine.pm
Appending installation info to D:\Dwimperl\perl\lib\perllocal.pod
  JKEENAN/List-Compare-0.37.tar.gz
  D:\Dwimperl\c\bin\dmake.EXE install UNINST=1 -- OK
```

```
cpan>
```

**Install successfully !!**

# Examples

- I: common elements



- II: randomization

@a = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)



@a = (3, 1, 8, 9, 10, 5, 2, 4, 7, 6)

# Answers

- I: common elements

- List::Compare

① qw() = ()

```
@Llist = qw(abel abel baker camera delta edward fargo golfer);
@Rlist = qw(baker camera delta delta edward fargo golfer hilton);

$lc = List::Compare->new(\@Llist, \@Rlist);

@intersection = $lc->get_intersection;
@union = $lc->get_union;
```

- II: randomization

- Algorithm::Numerical::Shuffle

```
use Algorithm::Numerical::Shuffle qw /shuffle/;

@shuffled = shuffle (1, 2, 3, 4, 5, 6, 7);

$in_situ = [qw /one two three four five six/];
shuffle $in_situ;
```





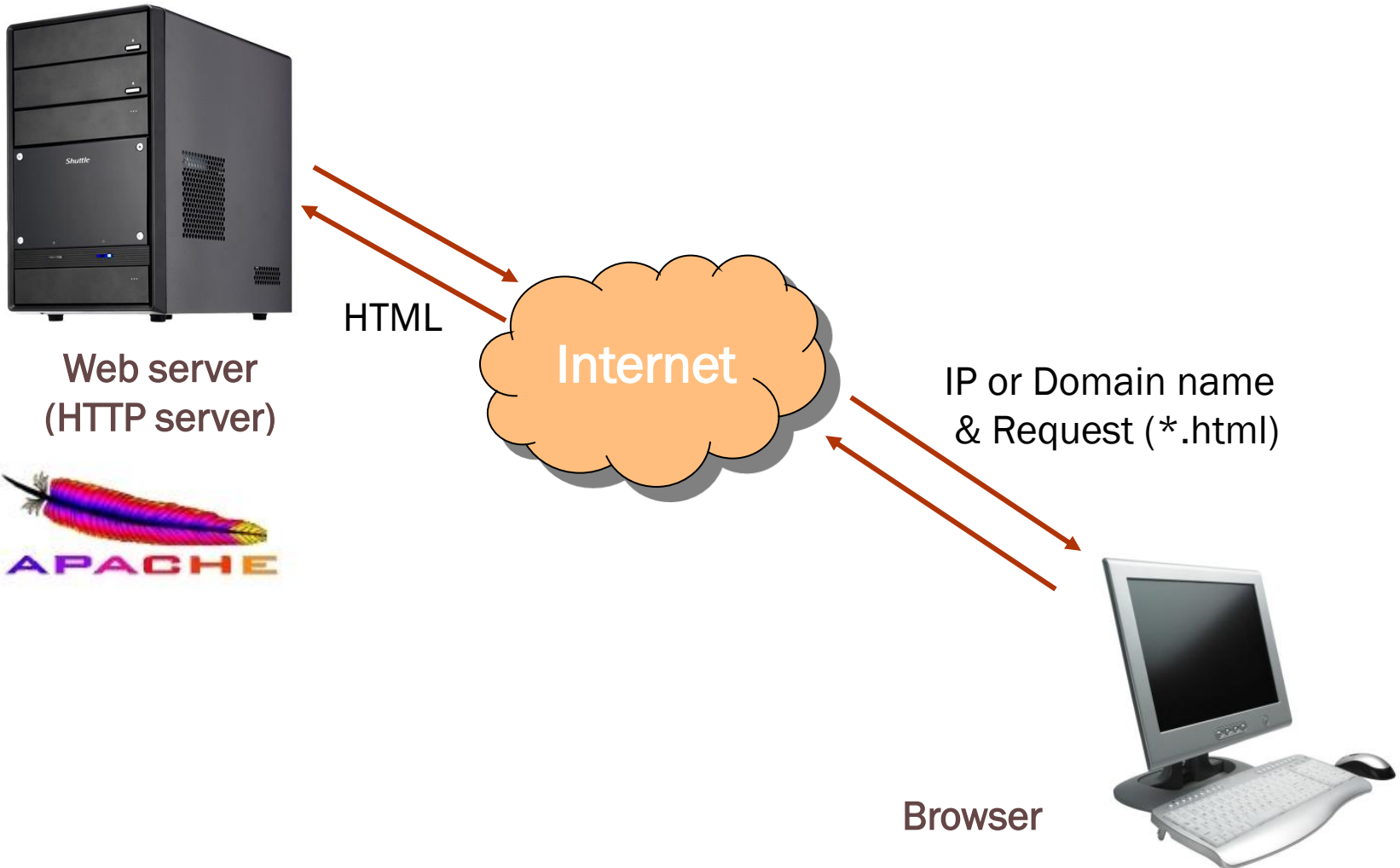
# PART I: CGI module

---

Reference: <http://search.cpan.org/~markstos/CGI.pm-3.63/lib/CGI.pm>



# Internet concept



# Install your own web server

---

Apache

AppServ (including Apache, PHP, MySQL)



# Understanding of the “HTML”

---

Tutorial <http://www.w3schools.com/html/default.asp>

Reference: <http://www.w3schools.com/tags/default.asp>

Google search for “HTML 教學”

# About HTML

- HTML stands for **Hyper Text Markup Language**
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

# HTML Document = Web Page

```
<html>
  <head>
    <title>    </title>
  </head>
  <body>
    [Empty Body]
  </body>
</html>
```



# A common form of an HTML tags

```
<tag attribute1="value1" attribute2="value2" ... >  
    Content to be rendered  
</tag>
```

Example:

```
<font col="red" size="2" face="Verdana">  
    Hello World !!  
</font>
```



# Heading, paragraph, and newline

## Heading

```
<h1>Heading1</h1>  
<h2>Heading2</h2>  
<h3>Heading3</h3>  
<h4>Heading4</h4>  
<h5>Heading5</h5>  
<h6>Heading6</h6>
```

## Paragraph

```
<p>Paragraph 1</p> <p>Paragraph 2</p>
```

## Newline

```
Content 1<br/>Content 2
```

**Heading1**

**Heading2**

**Heading3**

**Heading4**

**Heading5**

**Heading6**

Paragraph 1

Paragraph 2

Content 1

Content 2

# List

```
<h4>An Unordered List:</h4>
```

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

```
<h4>An Ordered List:</h4>
```

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

## An Unordered List:

- Coffee
- Tea
- Milk

## An Ordered List:

1. Coffee
2. Tea
3. Milk

# Table

```
<h4>This is a table:</h4>
<table border="1">
  <tr>
    <th>header 1</th>
    <th>header 2</th>
  </tr>
  <tr>
    <td>row 1, cell 1</td>
    <td>row 1, cell 2</td>
  </tr>
  <tr>
    <td>row 2, cell 1</td>
    <td>row 2, cell 2</td>
  </tr>
</table>
```

**This is a table:**

<b>header 1</b>	<b>header 2</b>
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

# Link and image

```
<h4>This is a link:</h4>  
<a href="http://www.ym.edu.tw/" target=_blank>Link to YM</a>  
  
<h4>Show a impage:</h4>  

```

**This is a link:**

[Link to YM](http://www.ym.edu.tw/)

**Show a impage:**



**\_blank:** open in a new window  
**\_self:** open in the current window

# Form - I

```
<form>
```

```
Text field <input type="text" name="keyword"><br>  
Password field <input type="password" name="pswd"><br>  
Radio button <input type="radio" name="botton"><br>  
Checkbox <input type="checkbox" name="check"><br>  
File field <input type="file" name="file"><br>  
Submit button <input type="submit" name="submit"><br>  
Reset button <input type="reset"><br>  
Text area <textarea name="text" rows=10 cols=15></textarea>
```

```
</form>
```

Name attributes are essential.

Text field

Password field

Radio button

Checkbox

File field  Cusick\_NatM...ds\_2009.pdf

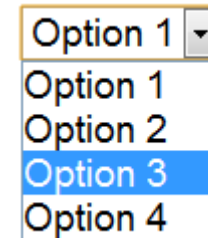
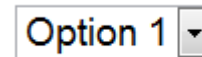
Submit button

Reset button

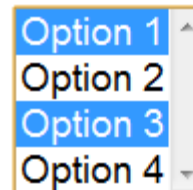
Text area

# Form - II

```
<select name="choose1">
  <option value=1>Option 1</option>
  <option value=2>Option 2</option>
  <option value=3>Option 3</option>
  <option value=4>Option 4</option>
</select>
```



```
<select name="choose2" multiple="multiple" size="4">
  <option value='a'>Option 1</option>
  <option value='b'>Option 2</option>
  <option value='c'>Option 3</option>
  <option value='d'>Option 4</option>
</select>
```



# Hand-on Practice I

Create a HTML document to show following page:

## Hand-on Practice

What's your name?

Gender:  Male or  Female

Select one or more your interested sport(s):

1.  Baseball
2.  Baseketbal
3.  Volleyball
4.  Tennis

Today is

---

# Solution

```
<h3>Hand-on Practice</h3>
<form>
What's your name? <input type="text" name="name" value="who are you?">
<br>
Gender: <input type="radio" name="gender" value="m" checked> Male or
       <input type="radio" name="gender" value="f"> Female
<br>
Select one or more your interested sport(s):
<ol>
  <li><input name="sport" type="checkbox" value="Baseball">Baseball</li>
  <li><input name="sport" type="checkbox" value="Baseketball">Baseketbal</li>
  <li><input name="sport" type="checkbox" value="Volleyball">Volleyball</li>
  <li><input name="sport" type="checkbox" value="Tennis" checked>Tennis</li>
</ol>
Today is
<select name="week">
  <option value="Mon">Mon</option>
  <option value="Tus">Tus</option>
  <option value="Wed">Wed</option>
  <option value="Thu">Thu</option>
  <option value="Fri">Fri</option>
  <option value="Sat">Sat</option>
  <option value="Sun">Sun</option>
</select>
<hr>
<input type="submit">
<input type="reset">
</form>
```



# Static HTML → Dynamic HTML

---

CGI

# What's CGI?

- The **Common Gateway Interface (CGI)** is a standard protocol for interfacing external application software with an information server, commonly a web server. – from Wikipedia



# Our 1<sup>st</sup> CGI example

```
#!C:\Perl\bin\perl
use CGI qw/:standard/;

print header,
      start_html("A simple CGI example"),
      h1("My first CGI script"),
      "This is content...",
      end_html
```



```
1 <!DOCTYPE html
2     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-US">
5 <head>
6 <title>A simple CGI example</title>
7 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
8 </head>
9 <body>
10 <h1>My first CGI script</h1>This is content...
11 </body>
12 </html>
```



# Basic HTML Tags

HTML	Code	HTML	Code	HTML	Code
<font>	font()	<i>	i()	<h1>	h1()
<b>	b()	<strike>	strike()	<a>	a()
<p>	p()	<big>	big()	<img>	img()
 	br()	<small>	small()		
<hr>	hr()	<sup>	sup()		
<em>	em()	<sub>	<b>sub()</b>		
<u>	u()	<strong>	strong()		

Code

----

```
h1 ()
h1 ('some', 'contents');
h1 ({-align=>left});
h1 ({-align=>left}, 'contents');
```

Generated HTML

-----

```
<h1>
<h1>some contents</h1>
<h1 align="LEFT">
<h1 align="LEFT">contents</h1>
```

# List

```
print ul(  
  li({-type=>'disc'}, ['Sneezy', 'Doc', 'Sleepy', 'Happy'])  
);
```

```
<ul>  
  <li type="disc">Sneezy</li>  
  <li type="disc">Doc</li>  
  <li type="disc">Sleepy</li>  
  <li type="disc">Happy</li>  
</ul>
```

- ◆ Sneezy
- ◆ Doc
- ◆ Sleepy
- ◆ Happy

# Table

```
print table({-border      => 1,
            -bordercolor => 'red',
            -width       => 500},
            caption("When Should You Eat Your Vegetables?"),
            Tr({-align => 'center',
                -valign => 'top'},
                [
                    th({-bgcolor => '#CCCCFF'},
                        ['Vegetable', 'Breakfast', 'Lunch', 'Dinner']),
                    td(['Tomatoes', 'no', 'yes', 'yes']),
                    td({-align => 'right', -bgcolor => 'yellow'},
                        ['Broccoli', 'no', 'no', 'yes']),
                    td(['Onions', 'yes', 'yes', 'yes'])
                ])
            );
```

```
<table bordercolor="red" border="1" width="500">
  <caption>When Should You Eat Your Vegetables?</caption>
  <tr align="center" valign="top">
    <th bgcolor="#CCCCFF">Vegetable</th>
    <th bgcolor="#CCCCFF">Breakfast</th>
    <th bgcolor="#CCCCFF">Lunch</th>
    <th bgcolor="#CCCCFF">Dinner</th>
  </tr>
  <tr align="center" valign="top">
    <td>Tomatoes</td>
    <td>no</td>
    <td>yes</td>
    <td>yes</td>
  </tr>
  <tr align="center" valign="top">
    <td bgcolor="yellow" align="right">Broccoli</td>
    <td bgcolor="yellow" align="right">no</td>
    <td bgcolor="yellow" align="right">no</td>
    <td bgcolor="yellow" align="right">yes</td>
  </tr>
  <tr align="center" valign="top">
    <td>Onions</td>
    <td>yes</td>
    <td>yes</td>
    <td>yes</td>
  </tr>
</table>
```

When Should You Eat Your Vegetables?

Vegetable	Breakfast	Lunch	Dinner
Tomatoes	no	yes	yes
Broccoli	no	no	yes
Onions	yes	yes	yes



# Hand-on Practice II

Create a CGI script to show following page:

## Hand-on Practice

What's your name?

Gender:  Male or  Female

Select one or more your interested sport(s):

1.  Baseball
2.  Baseketbal
3.  Volleyball
4.  Tennis

Today is

---



```

use strict;
use CGI qw/:standard/;

print
    header,
    start_html('Simple Script'),
    h3('Hand-on Practice');
print start_form,
    "What's your name? ",textfield(-name => "name", -value => "who are you?"),
    br,
    "Gender: ",
    radio_group(-name => "geneder",
                -value => ["Male", "Female"],
                -default => "Male"),
    br,
    "Select one or more your interested sport(s):",
    ol(li[checkbox_group(-name=>'sport',
                    -values=>['Baseball','Baseketball','Volleyball','Tennis'],
                    -defaults=>['Tennis'])]),
    "Today is",
    popup_menu(-name=>'week',
              -values=>['Mon','Tus','Wed','Thu','Fri','Sat','Sun']),
    hr,
    submit,
    reset,
    end_form;
print end_html;

```

# How to transfer data?

```
<form action="mycgi.cgi" method="get"  
enctype="application/x-www-form-urlencoded">
```

**action:** URL of the CGI program

**method:** how to transfer the data to CGI  
(**get** | **post**)

**enctype:** form type  
(**"multipart/form-data"** |  
**"application/x-www-form-urlencoded"** |  
**"text/plain"**)

If you want to do **file uploads**, you should use **"multipart/form-data"** .

# Get & Post

```
<form action="mycgi.cgi" method="get">  
  <input type="text" name="keyword" value="hello">  
  <input type="submit">  
</form>
```

```
http://localhost/cgi-bin/mycgi.cgi?keyword=hello
```

```
<form action="mycgi.cgi" method="post">  
  <input type="text" name="keyword" value="hello">  
  <input type="submit">  
</form>
```

```
http://localhost/cgi-bin/mycgi.cgi
```

If you want to do file uploads, you should use "post" ..

# An example of dynamic document

## Simple Script

What's your name?

What's the combination?  eenie  meenie  minie  moe

What's your favorite color?

## Simple Script

What's your name?

What's the combination?  eenie  meenie  minie  moe

What's your favorite color?

Your name is *Wang Shou Ming*

The keywords are: *eenie, minie, moe*

Your favorite color is *blue*.



# CGI code

Form  
section

```
#!/C:\Perl\bin\perl
use CGI qw/:standard/;
print
    header,
    start_html('Simple Script'),
    h1('Simple Script'),
    start_form,
    "What's your name? ",textfield('name'),p,
    "What's the combination?",
    checkbox_group(-name=>'words',
        -values=>['eenie','meenie','minie','moe'],
        -defaults=>['eenie','moe']),p,
    "What's your favorite color?",
    popup_menu(-name=>'color',
        -values=>['red','green','blue','chartreuse']),p,
    submit,
    end_form,
    hr,"\n";

if (param) {
    print
        "Your name is ",em(param('name')),p,
        "The keywords are: ",em(join(", ",param('words'))),p,
        "Your favorite color is ",em(param('color')),"\n";
}
print end_html;
```

# Fetch values of named parameters

- use param()

```
#!C:\Perl\bin\perl
use CGI qw/:standard/;
print
    header,
    start_html('Simple Script'),
    h1('Simple Script'),
    start_form,
    "What's your name? ",textfield('name'),p,
    "What's the combination?",
    checkbox_group(-name=>'words',
        -values=>['eenie','meenie','minie','moe'],
        -defaults=>['eenie','moe']),p,
    "What's your favorite color?",
    popup_menu(-name=>'color',
        -values=>['red','green','blue','chartreuse']),p,
    submit,
    end_form,
    hr,"\n";
```

```

}
if (param) {
    print
        "Your name is ",em(param('name')),p,
        "The keywords are: ",em(join(", ",param('words'))),p,
        "Your favorite color is ",em(param('color')),"\n";
    }
print end_html;
```

# Hand-on Practice III

Create two CGI scripts to show following page:

input\_form.cgi

## Hand-on Practice

What's your name?

Gender:  Male or  Female

Select one or more your interested sport(s):

1.  Baseball
2.  Baseketbal
3.  Volleyball
4.  Tennis

Today is

---

show\_info.cgi

My name is *Wang Shou Ming*.

I'm a **girl**, not a **boy**.

I play "Baseball, Volleyball".

Today is **Thuseday**.

# Solution

```
my %weeks =
(
    "Mon" => "Monday",
    "Tue" => "Tuesday",
    "Wed" => "Wednesday",
    "Thu" => "Thursday",
    "Fri" => "Friday",
    "Sat" => "Saturday",
    "Sun" => "Sunday"
);

if (param)
{
    my $name = param("name");
    my $gender = param("gender");
    my $sport = join("\",", param("sport"));
    my $week = param("week");
    print "My name is ", em($name), br;
    if ($gender eq 'Male')
    {
        print "I'm a ", b("boy"), ", not a ", b("girl"), ".";
    }
    else
    {
        print "I'm a ", b("gril"), ", not a ", b("boy"), ".";
    }
    print br;
    print "I play \"\$sport\"", br;
    print "Today is ", font({-color => "blue"}, $weeks{$week});
}
```

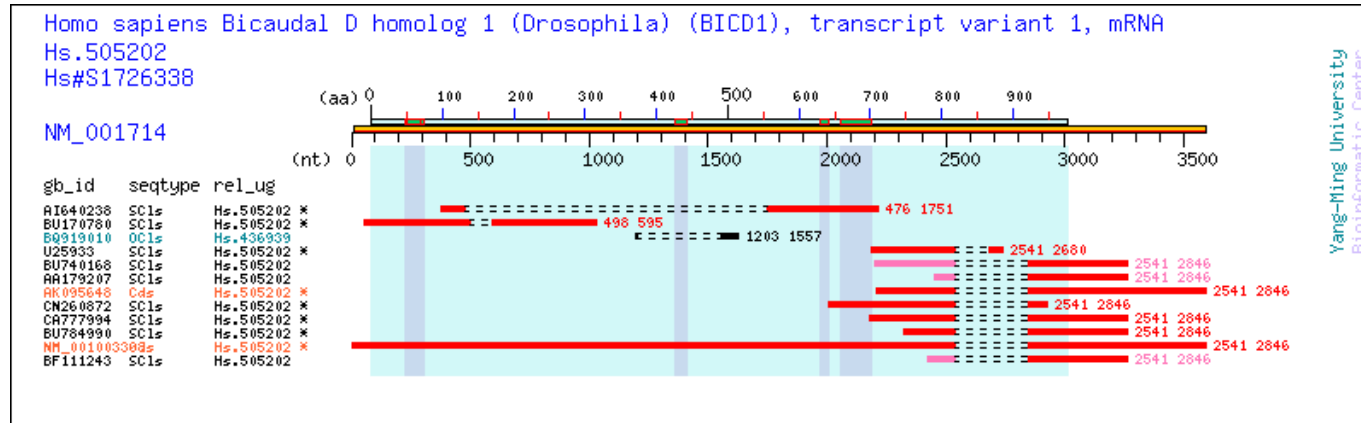


# PART II: GD module

---

Reference: <http://search.cpan.org/~lds/GD-2.50/GD.pm>

# GD Examples



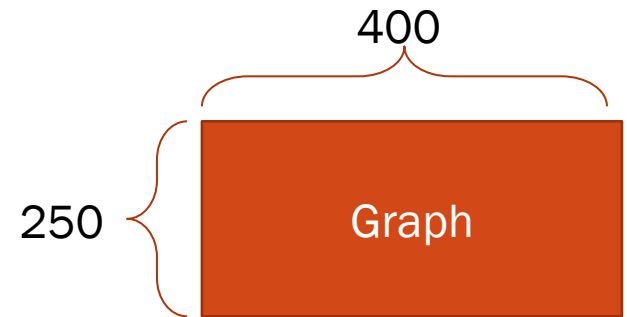
Yang-Ming University  
 Bioinformatic Center

# Quick Start

- Importing GD package (use GD;)
- Creating the image
- Allocating colors into the image colormap
- Drawing the image
- Image output

# Create a new image

- Create new and empty 400 \* 250 pixel image  
`$image = new GD::Image(400, 250)`



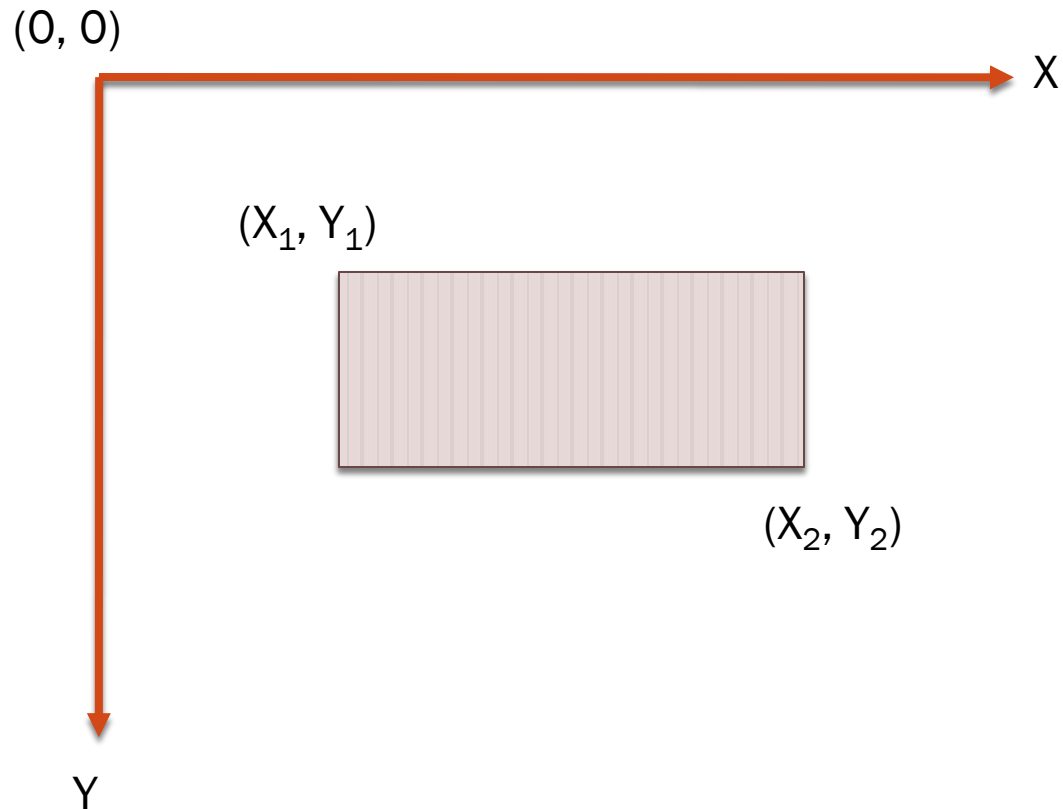
# Color Allocate

- Use `colorAllocate(int R, int G, int B)` to assign colors.
- Ex:
  - `$white = $image -> colorAllocate(0, 0, 0);`
  - `$black = $image -> colorAllocate(255, 255, 255);`
  - `$red = $image -> colorAllocate(255, 0, 0);`
  - `$green = $image -> colorAllocate(0, 255, 0);`
  - `$blue = $image -> colorAllocate(0, 0, 255);`
- Other colors refer to <http://www.pitt.edu/~nisg/cis/web/cgi/rgb.html>

# Drawing image

- `$image->transparent($white);`
  - Make color 'white' to be invisible
  - -1 to disable it
- `$image->interlaced('true');`
  - 'undef' to disable it.

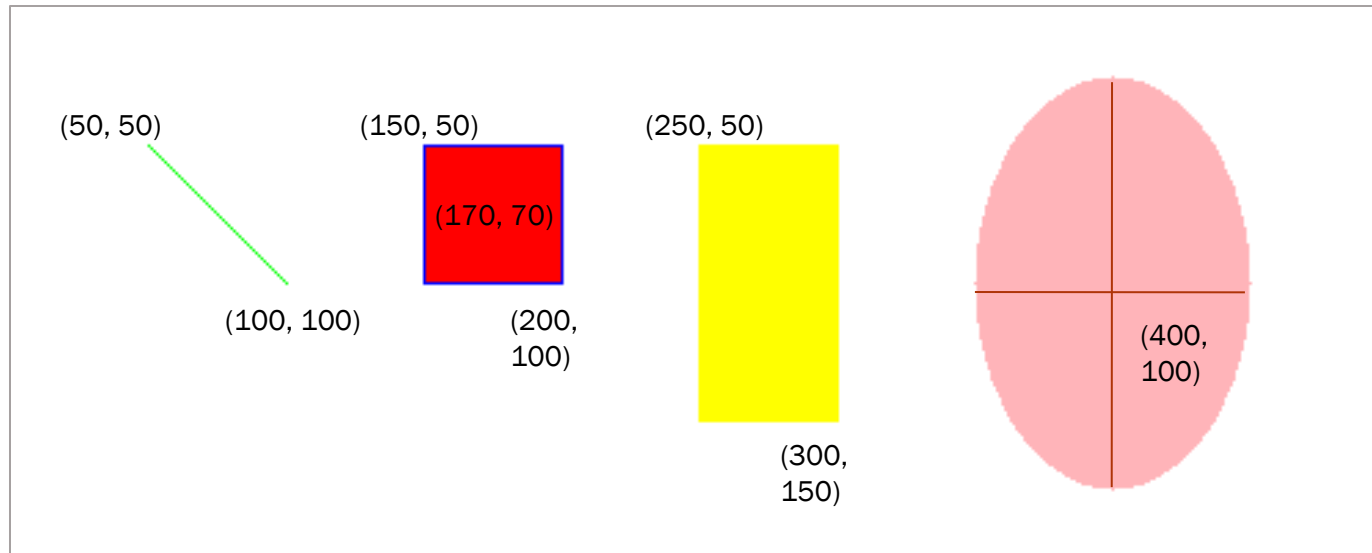
# Localization



```
$image -> filledRectangle($x1, $y1, $x2, $y2,  
$color);
```

# Drawing image – Basic methods

```
$image->line(50, 50, 100, 100, $green);  
$image->rectangle(150, 50, 200, 100, $blue);  
$image->fill(170, 70, $red);  
$image->filledRectangle(250, 50, 300, 150, $yellow);  
$image->filledEllipse(400, 100, 100, 150, $pink);
```





# Drawing image - Strings

- `$image -> string(**font, X, Y, string, color);`
- There are 5 fonts in GD.
  - `gdSmallFont`
  - `gdMediumBoldFont`
  - `gdTinyFont`
  - `gdLargeFont`
  - `gdGiantFont`
- Example:  
`$image->string(gdTinyFont, 10, 25, "hello world", $black);`

# Image Output

- Convert image data to supporting format, then you can print it, pipe it or write to file.

```
### Method I ###  
binmode STDOUT;  
print STDOUT $image->png;  
  
### Method II ###  
open OUT, ">test.gif";  
binmode OUT;  
print OUT $image->gif;  
close OUT;
```

# Practice - Basic

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

my $image = new GD::Image(300, 400);

### Color Allocation ###
my $white = $image -> colorAllocate(255, 255, 255);
my $violet = $image -> colorAllocate(147, 112, 219);
my $pink = $image -> colorAllocate(255, 180, 185);

### Drawing Image ###
$image -> transparent($white);
$image -> interlaced('true');

$image -> rectangle(0, 0, 250, 50, $violet);
$image -> filledRectangle(50, 70, 280, 100, $pink);
$image -> string(gdTinyFont, 50, 25, "frame in violet", $violet);
$image -> string(gdTinyFont, 100, 80, "pink rectangle", $white);

### Image Output ###
open OUT, ">test.png" or die "Can't open the file test.png\n";
binmode OUT;
print OUT $image->png;
close OUT;
```

# Practice - CGI

```
#!/usr/bin/perl

use strict;
use GD;

### Note this description ###
print "content-type:image/png\n";

my $image = new GD::Image(300, 400);

### Color Allocation ###
my $white = $image -> colorAllocate(255, 255, 255);
my $violet = $image -> colorAllocate(147, 112, 219);
my $pink = $image -> colorAllocate(255, 180, 185);

### Drawing Image ###
$image -> transparent($white);
$image -> interlaced('true');

$image -> rectangle(0, 0, 250, 50, $violet);
$image -> filledRectangle(50, 70, 280, 100, $pink);
$image -> string(gdTinyFont, 50, 25, "frame in violet", $violet);
$image -> string(gdTinyFont, 100, 80, "pink rectangle", $white);

### Image Output ###
binmode STDOUT;
print STDOUT $image->png;
```

# Result



frame in violet

pink rectangle

# Hand-on Practice IV

## Hand-on Practice

Name:

Shape:  Square  Ellipse

Color:

---



## Hand-on Practice

Name:

Shape:  Square  Ellipse

Color:

---



```

#!C:\Perl\bin\perl
use strict;
use CGI qw/:standard/;

print
    header,
    start_html('Simple Script'),
    h3('Hand-on Practice');
print start_form(-action => "exercise4.cgi", -method => "get", -enctype => "multipart/form-data"),
    "Name: ", textfield(-name => "name"),
    br,
    "Shape: ",
    radio_group(-name => "shape",
        -value => ["Square", "Ellipse"],
        -default => "Square"),
    br,
    "Color: ",
    popup_menu(-name=>'color',
        -values=>['red', 'blue', 'yellow', 'green']),
    br,
    submit(-value => "Submit Query"),
    reset,
    end_form,
    hr;

if (param)
{
    my $name = param("name");
    my $shape = param("shape");
    my $color = param("color");
    print img{src => "img.cgi?name=$name&shape=$shape&color=$color"};
}

print end_html;

```

# img.cgi

```
#!C:\Perl\bin\perl

use strict;
use CGI qw/:standard/;
use GD;

print "Content-type: image/png\n\n";

my $image = new GD::Image(200, 200);

my $white = $image -> colorAllocate(255, 255, 255);
my $black = $image -> colorAllocate(0, 0, 0);

my %color_map =
(
    "red"    => $image -> colorAllocate(255, 0, 0),
    "blue"   => $image -> colorAllocate(0, 62, 255),
    "yellow" => $image -> colorAllocate(255, 255, 0),
    "green"  => $image -> colorAllocate(0, 255, 0)
);

my $color = $color_map{param("color")};

$image -> transparent($white);
$image -> interlaced('true');

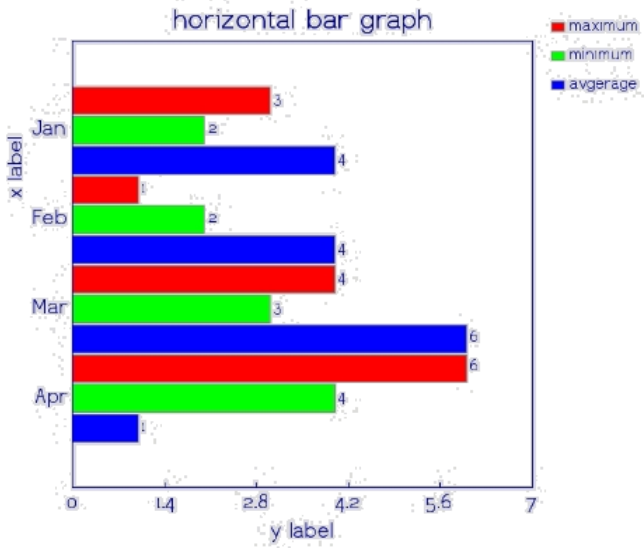
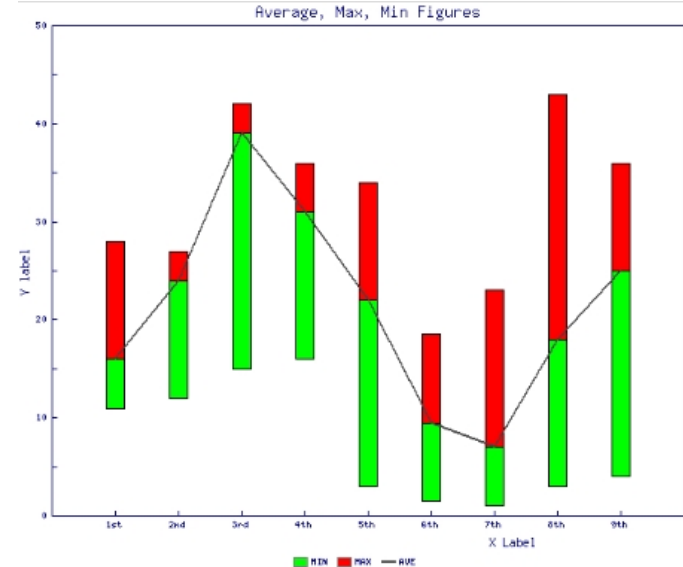
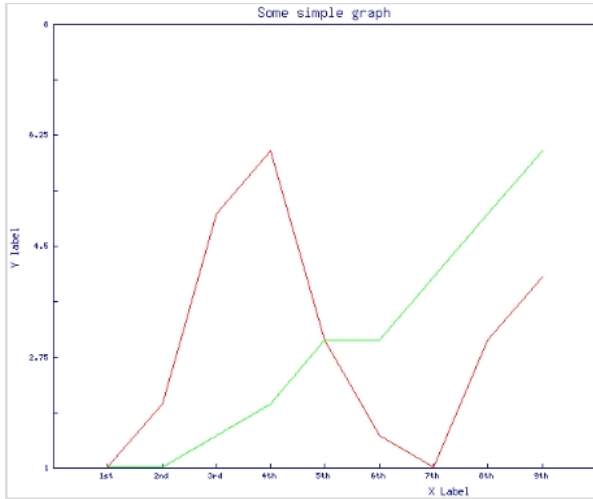
if (param("shape") eq 'Square')
{
    $image -> filledRectangle(50, 50, 150, 150, $color);
}
elsif (param("shape") eq 'Ellipse')
{
    $image -> filledEllipse(100, 100, 100, 100, $color);
}

my $name = param("name");
$image -> string(gdLargeFont, 50, 100, $name, $white);

binmode STDOUT;
print STDOUT $image->png;
```



# GD::Graph



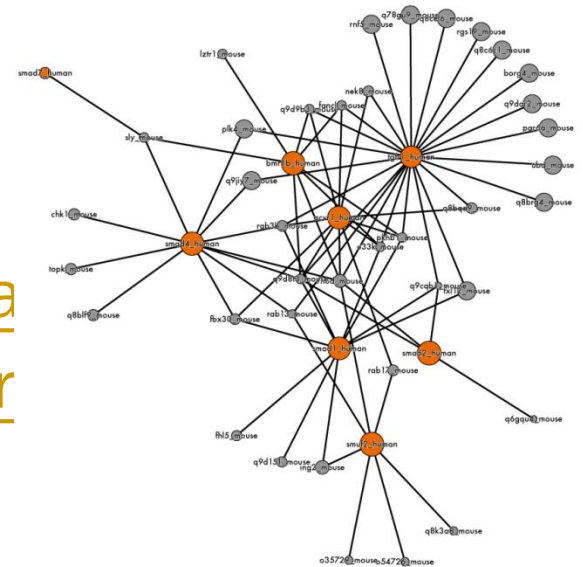
# Other Tools for Generating Image

## Charts

- GD::Graph:  
<http://search.cpan.org/~bwarfield/GDGraph-1.44/Graph.pm>
- Google::Chart:  
<http://search.cpan.org/~marcel/Google-Chart-0.04/lib/Google/Chart.pm>

## Networks

- eXpanda:  
<http://medcd.iab.keio.ac.jp/expa>
- Graphviz: <http://www.graphviz.org>



# Mathematics and statistics-related modules

- Math::Cephes
- Math::MatrixReal
- Statistics::Descriptive
- Statistics::Distribution
- Math::CDF
- Math::Random
- Statistics::ANOVA
- Statistics::Ttest
- Statistics::Basic::Correlation
- PDL
- Statistics::R

# Bioinformatics and Biomedical-related modules

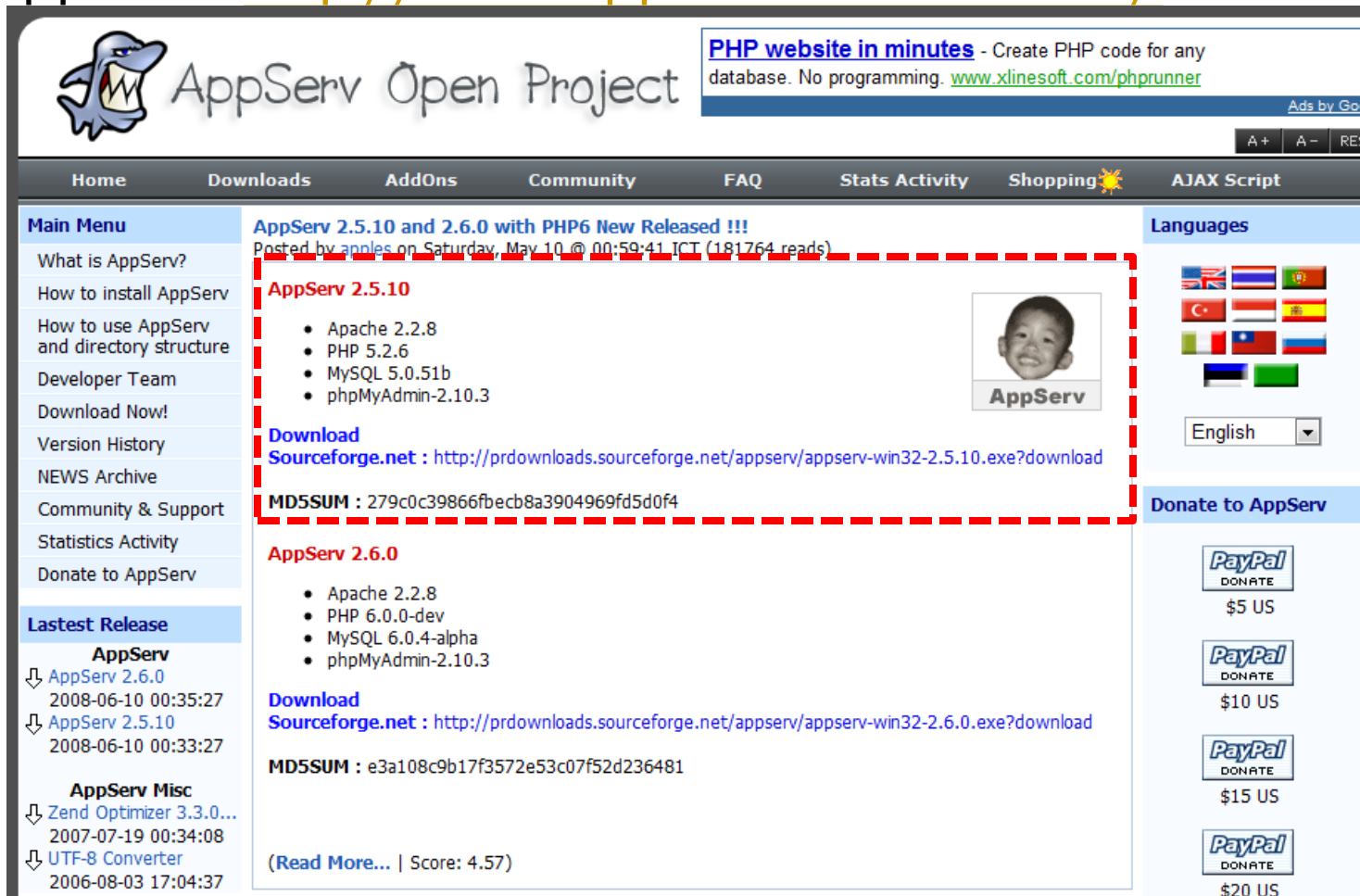
- BioPerl ([www.bioperl.org](http://www.bioperl.org))
- Network analysis:
  - Graph
  - Graph::Maker
- Machine learning & data mining
  - Algorithm::SVM
  - AI::ANN
  - Algorithm::DecisionTree
  - Statistics::PCA
  - Algorithm::Kmeans
  - Algorithm::Cluster

# Q & A

---

# Install your web server (for windows)

- AppServ: <http://www.appservnetwork.com/>



The screenshot shows the AppServ Open Project website. The main content area is titled "AppServ 2.5.10 and 2.6.0 with PHP6 New Released !!!" and is highlighted with a red dashed border. It lists the components included in the release: Apache 2.2.8, PHP 5.2.6, MySQL 5.0.51b, and phpMyAdmin-2.10.3. Below this, there are download links for Sourceforge.net and MD5SUM values for both versions. The website also features a navigation menu, a main menu, a languages dropdown set to English, and a donate section with PayPal buttons for \$5, \$10, \$15, and \$20.

**AppServ 2.5.10 and 2.6.0 with PHP6 New Released !!!**  
Posted by apples on Saturday, May 10 @ 00:59:41 ICT (181764 reads)

**AppServ 2.5.10**

- Apache 2.2.8
- PHP 5.2.6
- MySQL 5.0.51b
- phpMyAdmin-2.10.3

**Download**  
**Sourceforge.net** : <http://prdownloads.sourceforge.net/appserv/appserv-win32-2.5.10.exe?download>

**MD5SUM** : 279c0c39866fbecb8a3904969fd5d0f4

**AppServ 2.6.0**

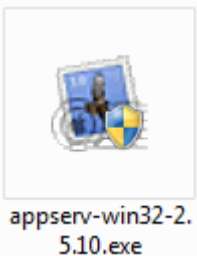
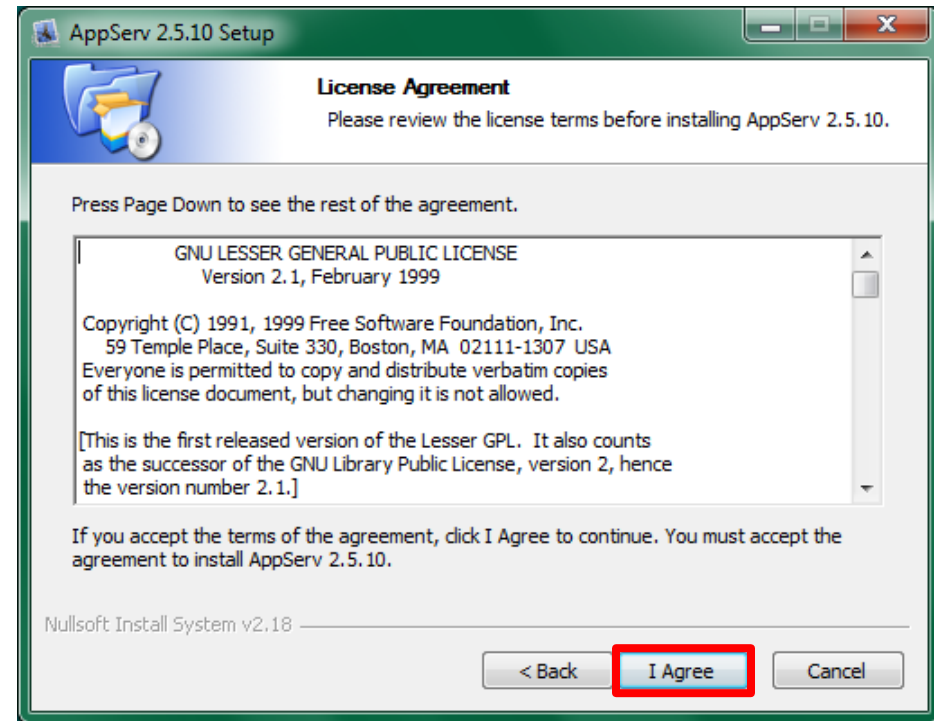
- Apache 2.2.8
- PHP 6.0.0-dev
- MySQL 6.0.4-alpha
- phpMyAdmin-2.10.3

**Download**  
**Sourceforge.net** : <http://prdownloads.sourceforge.net/appserv/appserv-win32-2.6.0.exe?download>

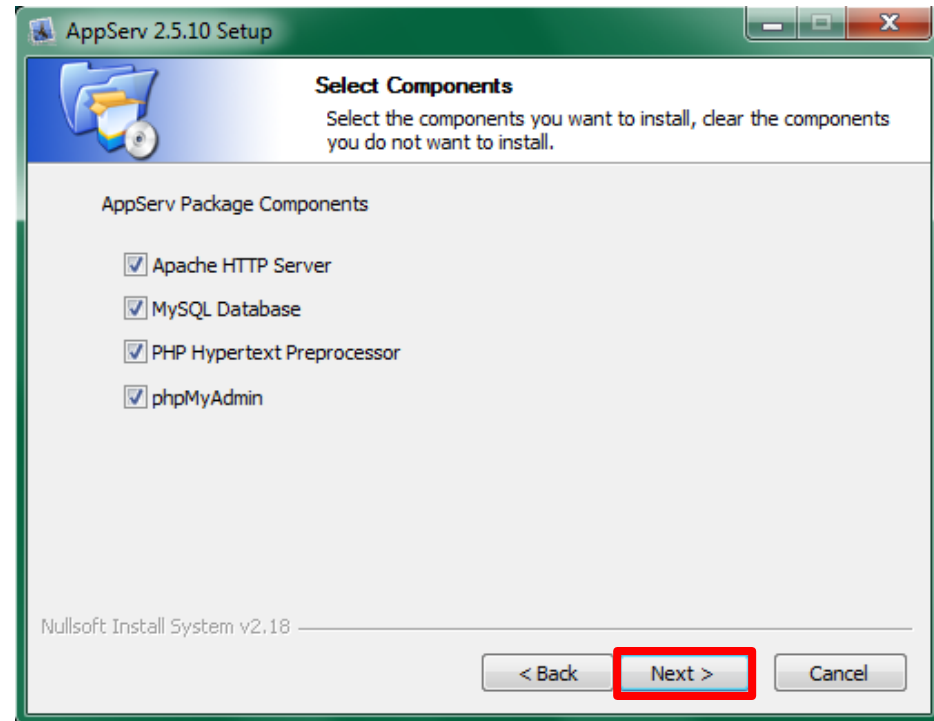
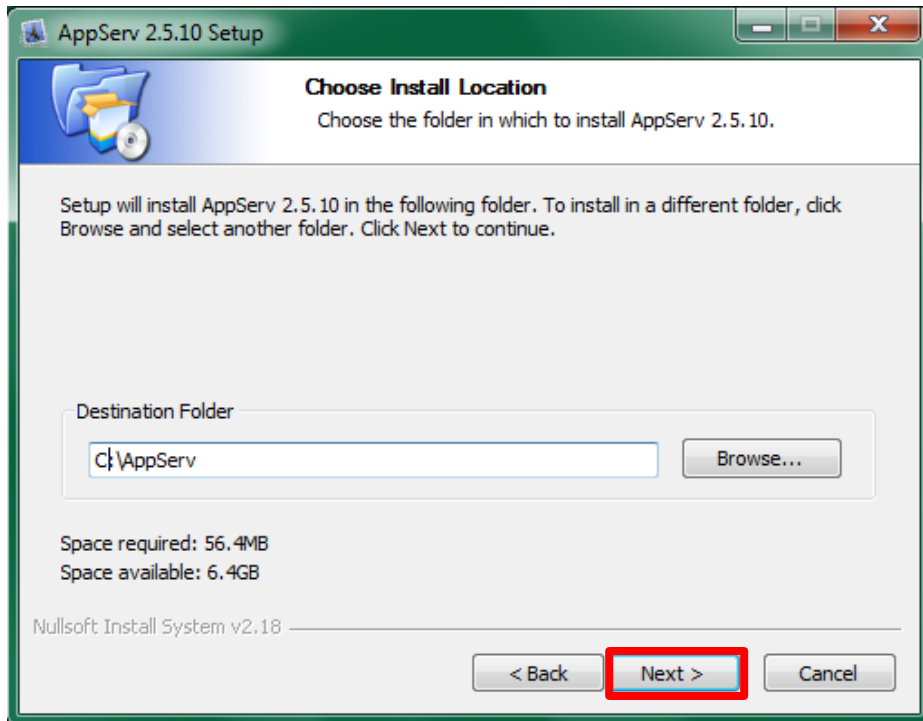
**MD5SUM** : e3a108c9b17f3572e53c07f52d236481

([Read More...](#) | Score: 4.57)

# Installation of AppServ – Step 1



# Installation of AppServ – Step 2





# Installation of AppServ – Step 3

AppServ 2.5.10 Setup

**Apache HTTP Server Information**  
Please enter your server's information.

Server Name (e.g. www.appservnetwork.com)

Administrator's Email Address (e.g. webmaster@gmail.com)

Apache HTTP Port (Default : 80)

Nullsoft Install System v2.18

< Back **Next >** Cancel

AppServ 2.5.10 Setup

**MySQL Server Configuration**  
Configure the MySQL Server instance.

Please enter Root password for MySQL Server.

Enter root password

Re-enter root password

MySQL Server Setting

Character Sets and Collations  
UTF-8 Unicode

Old Password Support (PHP MySQL API function.)

**Enable InnoDB**

Nullsoft Install System v2.18

< Back **Install** Cancel

# Installation of AppServ – Step 4

<http://localhost:port/>



## The AppServ Open Project - 2.5.10 for Windows



 **phpMyAdmin Database Manager Version 2.10.3**  
 **PHP Information Version 5.2.6**


About AppServ Version 2.5.10 for Windows

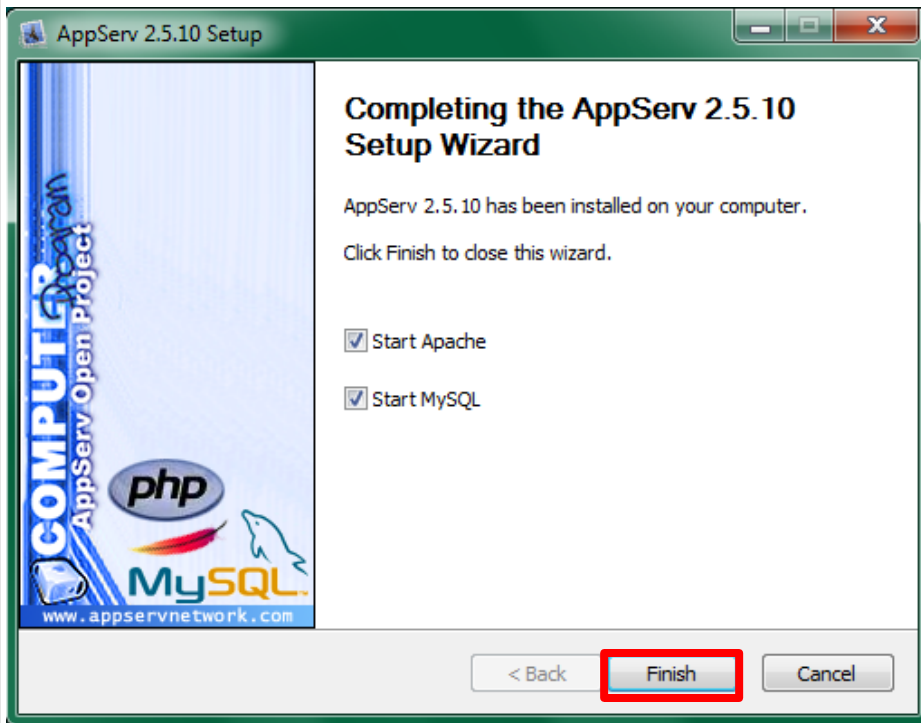
AppServ is a merging open source software installer package for Windows includes :

- **Apache Web Server** Version 2.2.8
- **PHP Script Language** Version 5.2.6
- **MySQL Database** Version 5.0.51b
- **phpMyAdmin Database Manager** Version 2.10.3

- ChangeLog
- README
- AUTHORS
- COPYING
- **Official Site** : <http://www.AppServNetwork.com>
- **Hosting support by** : <http://www.AppServHosting.com>

Change Language :  

 **Easy way to build Webserver, Database Server with AppServ :-)**



# Location of your HTML documents

