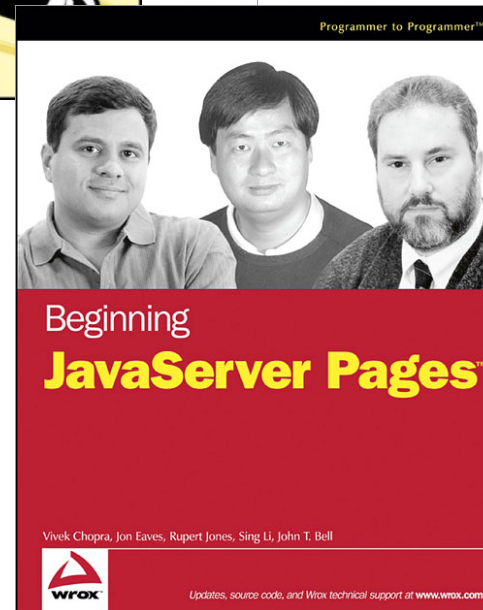
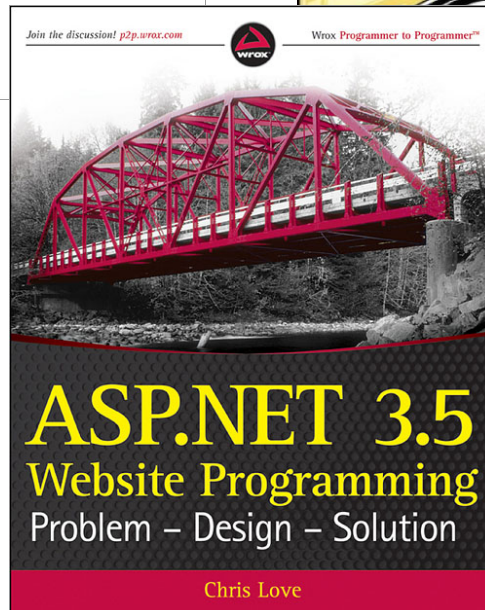
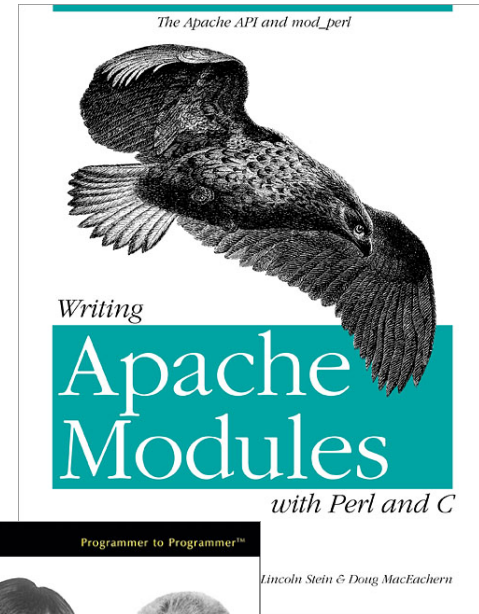
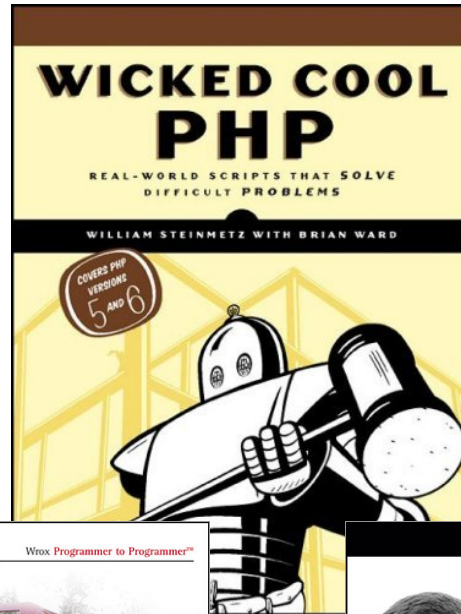
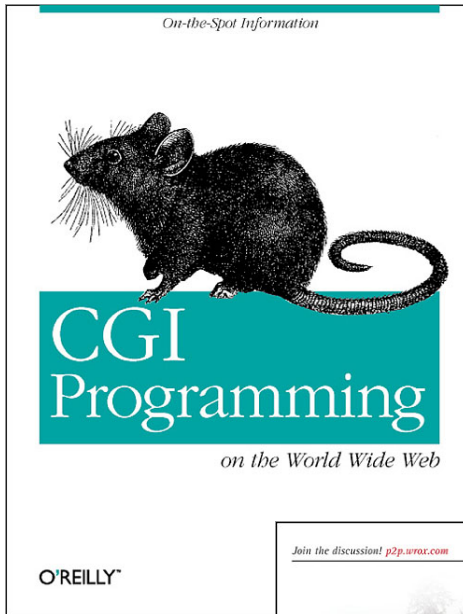


Dynamic web content technologies



Overview

- Dynamic content
 - What it is
 - Sources of input
- CGI (Common Gateway Interface)
 - Benchmark results from assignment #1
- FastCGI
- Server-side scripting
 - Assignment #2
- Web server modules

Static vs. dynamic

- **Static content**

- Images and pages don't change
 - Always the same, like a file server
- Fast to deliver, easy to cache, etc.

- **Dynamic content**

- Same URL may result in different delivered HTML
 - e.g. different preference on number of products to display
- May change as user interaction progresses
 - e.g. adding items to a shopping cart
- Need something besides just HTTP and HTML
 - HTTP is stateless
 - HTTP not programmable (e.g. conditional, loops)

Input to dynamic pages

- Form fields

- <INPUT> tags inside the <FORM> tag
- Data is URL encoded (percent-encoded)
 - In the URL if GET, in the HTTP payload if POST
- Unreserved characters:
 - ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-._~
- Reserved characters:
 - !*'();:@&=,\$//?#[]
 - Converted to %XX, where XX is ASCII in hexadecimal
 - %20 = space (also +), %21 = !, %23 = #, %25 = %, ...

Input to dynamic pages

- Cookies
 - Differentiate different clients hitting same page
- Other sources
 - User agent (browser)
 - HTTP referer (misspelled since RFC 1945)
 - The page you came from to get to this one
 - Client's IP address
 - Time of day
 - ...

CGI

- CGI (Common Gateway Interface)
 - In use since 1993
 - User requests a URL in a special location/file extension
 - e.g. <http://www.blah.com/cgi-bin/lookup>
 - Web server passes request to script/program
 - Sets a laundry list of environment variables
 - Creates new process and runs program
 - Program's output sent to web client
 - Notes:
 - Program must have read+execute permission by web server user
 - Probably shouldn't be writeable by anybody

CGI pros and cons

- Advantages

- Simple to write

- Input from environment variables (GET) or standard input (POST)
 - Output via standard output

- Highly portable

- Supported by virtually all web servers
 - Write in portable scripting language, e.g. Perl, Python

- Safer?

- Runs outside web server process space, can run as different user

- Disadvantages

- Expensive

- Most fork process for each HTTP request
 - Process has to fire up resources, e.g. interpreter, DB connection

How expensive is CGI?

- Creating/destroying a process?
- Starting/ending Perl?
- Starting/ending Python?
- Experiment:
 - Test options on a page that doesn't do anything
 - Isolates process startup expense
 - Isolate interpreter startup expense

```
#!/usr/bin/perl  
  
print "Content-Type: text/plain;charset=us-ascii\n\n";
```


CGI noop results

- **Test setup:**
 - Pentium 4 @ 3.0 Ghz, 1GB memory, 512K L2 cache
 - Apache 2.2.21
 - Apache benchmark (ab) to localhost "noop" pages
 - 1000 requests each at concurrency {1, 5, 10, 20}
 - No CGI, CGI C, CGI Perl, CGI Python

Test case	Requests / second
No CGI	1272.2
CGI C program	389.7
CGI Perl script	236.1
CGI Python script	24.7

CGI results

- CGI benchmark on a "real" application:
 - Return value given key
 - Text file containing 100, 1000, or 10000 identify pairs

Name	Language	System	Requests / second
Keith	C	lab	327.3
Ryan	C	?	2172.8
Miles	Python	lab	19.8
Chris	Python	lab	22.5
Jake	Python	lab	15.0

C CGI results in detail

file	c	requests/sec	
100	1	1408.87	
100	5	4035.85	
100	10	4107.28	
100	20	4125.28	3419.32
1000	1	885.79	
1000	5	2994.63	
1000	10	3030.05	
1000	20	3043.18	2480.91
10000	1	199.67	
10000	5	743.88	
10000	10	772.82	
10000	20	756.58	618.23
mean	-	2172.82333	

Ryan, home computer, 4 cores

file	c	requests/sec	
100	1	350.66	
100	5	429.90	
100	10	427.01	
100	20	426.31	408.47
1000	1	309.90	
1000	5	400.93	
1000	10	397.84	
1000	20	383.55	373.06
10000	1	171.59	
10000	5	213.30	
10000	10	208.12	
10000	20	207.81	200.21
mean	-	327.24	

Keith, lab machine, 1 core

C CGI results in detail

file	c	requests/sec	
100	1	1408.87	
100	5	4035.85	
100	10	4107.28	
100	20	4125.28	3419.32
1000	1	885.79	
1000	5	2994.63	
1000	10	3030.05	
1000	20	3043.18	2480.91
10000	1	199.67	
10000	5	743.88	
10000	10	772.82	
10000	20	756.58	618.23
mean	-	2172.82	

Ryan, home computer, 4 cores

file	c	requests/sec	
100	1	336.71	
100	5	3530.49	
100	10	4295.64	
100	20	4291.02	3113.47
1000	1	298.63	
1000	5	3549.84	
1000	10	4059.63	
1000	20	4017.70	2981.45
10000	1	141.43	
10000	5	2203.17	
10000	10	2520.40	
10000	20	2547.34	1853.09
mean	-	2649.33	

Keith, home machine, 6 cores

Improving CGI

- **FastCGI**

- Developed by third party: Open Market
- Language independent (like CGI)
 - Supported by many languages: C, C++, Java, Python, Perl, Ruby, ...
- Isolated from web server process space (like CGI)
- Supported by many servers: Apache, IIS, Lighttpd, Nginx, ...

- **Benefits:**

- Faster! Persistent process serves many requests in its life
 - Amortizes process/interpreter startup
- Can proxy work to another server

FastCGI sample code

```
#!/usr/bin/perl

use FCGI;
use strict;

my $i = 0;

while (FCGI::accept() >= 0)
{
    print "Content-Type: text/plain;charset=us-ascii\n\n";
    print "Hello world! i = " . $i;
    $i++;
}
```

```
#include "fcgi_stdio.h"
#include <stdlib.h>

int main(void)
{
    int count = 0;
    while (FCGI_Accept() >= 0)
    {
        printf("Content-Type: text/plain;charset=us-ascii\n\n");
        printf("Hello world! count = %d\n", count);
        count++;
    }
}
```

```
# Adding FastCGI support in its own directory
LoadModule fastcgi_module modules/mod_fastcgi.so
<Directory "/usr/local/apache2/fcgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
    SetHandler fastcgi-script
</Directory>
```

FastCGI C results

- Name/value lookup from file application

file	c	requests/sec	
100	1	350.66	
100	5	429.90	
100	10	427.01	
100	20	426.31	408.47
1000	1	309.90	
1000	5	400.93	
1000	10	397.84	
1000	20	383.55	373.06
10000	1	171.59	
10000	5	213.30	
10000	10	208.12	
10000	20	207.81	200.21
mean	-	327.24	

Results using normal CGI

file	c	requests/sec	
100	1	1064.58	
100	5	1268.05	
100	10	1259.81	
100	20	1282.40	1218.71
1000	1	843.90	
1000	5	985.43	
1000	10	982.07	
1000	20	940.33	937.93
10000	1	258.12	
10000	5	272.00	
10000	10	278.23	
10000	20	281.88	272.56
mean	-	809.73	

Results using FastCGI

Server-side scripting

- **Server-side scripting**
 - Do your smart stuff inside the web server process
 - No process/interpreter startup expense
 - But still interpreted
 - Embed code into your HTML page
 - `<? echo "hello world!"; ?>`
 - Embedded code run by web server
 - Results are sent to client, not the code itself
 - Wide-variety of choices:
 - PHP: Hypertext Preprocessor
 - ASP (Active Server Pages), now ASP.NET
 - JSP (Java Server Pages)
 - ColdFusion

Scripting example

```
<html>
  <head>
    <script language="JavaScript">
      var secret = Math.floor(Math.random() * 10) + 1;
    </script>
  </head>
  <body onload="document.getElementById('num').innerHTML = secret;">
    Pssst, the secret number is <span class="num" id="num"></span>
    <br /><br />
    Your IP address is :
    <?php echo $_SERVER["REMOTE_ADDR"]; ?>
    <br /><br />
    <?php
    for ($i = 0; $i < 10; $i++)
      echo $i . " ";
    echo "<br />";
    ?>
  </body>
</html>
```

File as seen on the web server.

Scripting example

```
<html>
  <head>
    <script language="JavaScript">
      var secret = Math.floor(Math.random() * 10) + 1;
    </script>
  </head>
  <body onload="document.getElementById('num').innerHTML = secret;">
    Pssst, the secret number is <span class="num" id="num"></span>
    <br /><br />
    Your IP address is : ::1<br />
    <br />
    0 1 2 3 4 5 6 7 8 9 <br />  </body>
</html>
```

File as seen by the client's browser.

PHP scripting



- **PHP: Hypertext Preprocessor**

- For documentation see <http://php.net>

- Syntax very C/Java/Perl like

- Variables

- Prefixed with \$
- Loosely typed
- Don't need to be declare beforehand

- Hundreds of base functions:

- e.g. File I/O functions:
 - fopen(), fgets(), fputs(), fclose(), feof()

- Normally interpreted

- Some compiled versions, e.g. HipHop by Facebook



PHP results

- Name/value lookup from file application
 - Assignment #2
 - Reimplement lookup application in PHP
 - Two new web pages:
 - get.php
 - set.php
 - Create standard web site header/footer
 - Use PHP's include() function
 - Common way to avoid repeated code in web application
 - Benchmark with ab
 - Use "-n 1000" so it doesn't take forever

Going even faster...

- **Server-side scripting**
 - Normally interpreted
 - Not as fast as native code
- **Put native code in web server process**
 - e.g. Apache module, IIS ISAPI, Netscape NSAPI
 - Fast!
 - No startup expense
 - Runs as native compiled code
 - But not as portable, tied to specific web server
 - But web server process exposed to bugs in module

Apache hello world module

```
#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"

/* The sample content handler */
static int poc_rest_handler(request_rec *r)
{
    if (strcmp(r->handler, "poc_rest"))
        return DECLINED;
    r->content_type = "text/html";
    if (!r->header_only)
        ap_rputs("Hello world!\n", r);
    return OK;
}

static void poc_rest_register_hooks(apr_pool_t *p)
{
    ap_hook_handler(poc_rest_handler, NULL, NULL, APR_HOOK_MIDDLE);
}

/* Dispatch list for API hooks */
module AP_MODULE_DECLARE_DATA poc_rest_module = {
    STANDARD20_MODULE_STUFF,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    poc_rest_register_hooks /* register hooks */
};

# My custom hello world Apache module
LoadModule hello_module modules/mod_hello.so
<Location /hello>
    SetHandler hello
</Location>

/* table of config file commands */
poc_rest_register_hooks /* register hooks */
```

Apache module results 1

- Name/value lookup from file application

file	c	requests/sec	
100	1	350.66	
100	5	429.90	
100	10	427.01	
100	20	426.31	408.47
1000	1	309.90	
1000	5	400.93	
1000	10	397.84	
1000	20	383.55	373.06
10000	1	171.59	
10000	5	213.30	
10000	10	208.12	
10000	20	207.81	200.21
mean	-	327.24	

Results using normal CGI

file	c	requests/sec	
100	1	1552.30	
100	5	1851.48	
100	10	1853.66	
100	20	1900.98	1789.61
1000	1	1179.88	
1000	5	1296.82	
1000	10	1320.85	
1000	20	1285.90	1270.86
10000	1	311.48	
10000	5	330.26	
10000	10	329.59	
10000	20	333.12	326.11
mean	-		

Results using custom Apache module

Apache module results 2

- Name/value lookup from file application

file	c	requests/sec
100	1	1064.58
100	5	1268.05
100	10	1259.81
100	20	1282.40 1218.71
1000	1	843.90
1000	5	985.43
1000	10	982.07
1000	20	940.33 937.93
10000	1	258.12
10000	5	272.00
10000	10	278.23
10000	20	281.88 272.56
mean	-	809.73

Results using FastCGI

file	c	requests/sec
100	1	1552.30
100	5	1851.48
100	10	1853.66
100	20	1900.98 1789.61
1000	1	1179.88
1000	5	1296.82
1000	10	1320.85
1000	20	1285.90 1270.86
10000	1	311.48
10000	5	330.26
10000	10	329.59
10000	20	333.12 326.11
mean	-	1128.86

Results using custom Apache module

Apache module results 3

- Name/value lookup from file application

```
+-----+-----+-----+
| file | c | requests/sec
+-----+-----+-----+
| 100 | 1 | 1552.30
| 100 | 5 | 1851.48
| 100 | 10 | 1853.66
| 100 | 20 | 1900.98 1789.61
+-----+-----+-----+
| 1000 | 1 | 1179.88
| 1000 | 5 | 1296.82
| 1000 | 10 | 1320.85
| 1000 | 20 | 1285.90 1270.86
+-----+-----+-----+
| 10000 | 1 | 311.48
| 10000 | 5 | 330.26
| 10000 | 10 | 329.59
| 10000 | 20 | 333.12 326.11
+-----+-----+-----+
| mean | - | 1128.86
+-----+-----+-----+
```

Apache module, lab machine, 1 core

```
+-----+-----+-----+
| file | c | requests/sec
+-----+-----+-----+
| 100 | 1 | 1794.89
| 100 | 5 | 8401.91
| 100 | 10 | 20268.64
| 100 | 20 | 15072.73
+-----+-----+-----+
| 1000 | 1 | 1189.34
| 1000 | 5 | 5346.56
| 1000 | 10 | 19515.01
| 1000 | 20 | 18133.85
+-----+-----+-----+
| 10000 | 1 | 314.18
| 10000 | 5 | 2350.68
| 10000 | 10 | 2911.20
| 10000 | 20 | 3416.14
+-----+-----+-----+
| mean | - | 8226.26
+-----+-----+-----+
```

Apache module, home machine, 6 cores

Final option...

- What could possibly be better than running native code inside the Apache server process?
 - Create custom web server
 - HTTP is simple protocol
 - Why reinvent your own socket protocol?
 - Only handles HTTP requests for name/value get/set
 - Could be deployed on separate servers
 - Maybe faster?
 - TBD
 - Probably smaller memory footprint than Apache
 - Helps with scalability

Summary

- Dynamic content
 - Many ways to skin a cat?
 - Each has pros and cons
- CGI (Common Gateway Interface)
- FastCGI
- Server-side scripting
- Web server modules
- Building a custom web server