Web Development with Perl

Coast Open Source Software Technology (COSST) Symposium

October 18, 2003

Eric Hammond
Director of Technology
Rent.com
This Talk

Perl Usage
Complete Web Example
Experience/Case Study
Web Architecture
Introduction to Perl
Survey of CPAN
Resources
Sites Built With Perl

- TicketMaster
- IMDb
- Webby Awards
- AvantGo
- Salon
- eToys
- ValueClick
- TechWeb

- Amazon
- Slashdot

Sites I helped build:

- Rent.com
- Stamps.com
- Citysearch
- notlong.com

thousands/millions more...
Netcraft mod_perl Survey
Netcraft mod_perl Survey, Cont'd

- According to Netcraft, 20% of web sites on the Internet say they are built with mod_perl, but...
  - Not all sites that are built with a module use it
  - Not all sites that use a module make the fact public
- Furthermore, it is possible to build a site with Perl without even using mod_perl
- Conclusion: We have no idea how many web sites are built with Perl
- ...but it seems to be popular and growing
Why Use Perl?

- Exceptional power, functionality, stability, performance, support
- Many great success stories from big names
- Good programmers matter more than the language or the platform, but...

Perl gives good programmers an edge
The Perl Language

- The Perl language:
  - Very fun, very high level language
  - Easy to learn the basics
  - Gradually pick up more advanced techniques
  - Power is there as you need it
- There's More Than One Way To Do It (TMTOWTDTI)
  - Basics get the job done
  - Advanced techniques improve speed and quality
  - ...far beyond most other languages
Complete Web Example

reallyshort.com
reallyshort.com

- Reallyshort.com requirements specification:
  - User enters long URL => Site generates short URL
  - Users going to short URL are redirected to long URL

- Built using:
  - Linux, Apache, mod_perl, Mason, Perl, MySQL

- Time to develop: 20 minutes

- ...including domain name registration
MySQL schema

- Set up database, user, and table with two columns:
- 01 CREATE DATABASE reallyshort;
  02 USE reallyshort;

  03 GRANT DELETE, INSERT, SELECT, UPDATE
  04 ON reallyshort.* TO reallyshort@localhost
  05 IDENTIFIED BY 'password';

  06 CREATE TABLE reallyshort.link ( 
     07 link_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
     08 long_url TEXT NOT NULL
  09 );
ReallyShort.pm

- Create class interface to database table.

01 use strict;
02 package ReallyShort;

03 use base 'Class::DBI::mysql' ;

04 ReallyShort->set_db('Main',
05       'DBI:mysql:reallyshort',
06       'reallyshort' ,
07       'password');

08 ReallyShort->set_up_table('link');
09 1;
Command Line

- Program to shorten URLs from the command line:
  reallyshort 'http://www.google.com/search?q=shorten+urls'

```perl
01  #!/usr/bin/perl -wT
02  use strict;
03  use ReallyShort;
04  my $long_url = shift;
05  my $reallyshort = ReallyShort->create({ long_url => $long_url });
06  my $link_id = $reallyshort->link_id;
07  my $short_url = "http://reallyshort.com/$link_id";
08  print "ReallyShort URL is: $short_url\n";
```
Mason Code (1/2)

01..03   `<%args> $long_url => undef </%args>
04   `<%init>
05   use ReallyShort;
06   my $link_id = $m->dhandler_arg;
07   if ( $link_id ) {
08       if ( my ($reallyshort) = ReallyShort->search({ link_id => $link_id }) ) {
09           $r->cgi_header_out( 'Location' => $reallyshort->long_url);
10           return REDIRECT;
11       }
12   }
13   my $short_url;
14   if ( $long_url ) {
15       my $reallyshort = ReallyShort->create({ long_url => $long_url });
16       $short_url = 'http://reallyshort.com/' . $reallyshort->link_id;
17   }
18   `</%init>
Mason Code (2/2)

19   <html><head><title>ReallyShort.com</title>
20   <link rel="stylesheet" href="/css/site.css" type="text/css" />
21   </head><body>
22   % if ( $short_url ) {
23       <p>
24           The ReallyShort URL: <a href="<% $short_url %>"><% $short_url %></a>
25           will redirect to: <a href="/" title="Another">Another</a>
26       <p>
27   } else {
28       <form>
29           Long URL: <input name="long_url" size="50">
30           <input type="submit" value="Make ReallyShort"/>
31       </form>
32   }
33   </body></html>
body {
  background-color: #336699;
  color: #FFFFFF;
  font-family: Helvetica, Verdana, Arial, sans-serif;
  font-size: 18pt;
  margin: 100px;
}

a {
  color: #FFFFFF;
  font-family: Helvetica, Verdana, Arial, sans-serif;
  font-size: 18pt;
}
Main Page: Enter Long URL

Long URL: [http://www.google.com/search?q=shorten+urls]
Submit Page: Show Short URL

The ReallyShort URL: http://reallyshort.com/28

will redirect to:
http://www.google.com/search?q=shorten+urls

Another
reallyshort.com/28 Sends You To...

Domain names, free domain name, web hosting and email address by ...
V3 Domain Names & Web Hosting, V3 Domain Names & Web Hosting Best Prices ... Great Service! ...
www.v3.com/ - 27k - Cached - Similar pages

notlong: Web Sites That Shorten Long URLs
Web Sites That Shorten Long URLs. The following free web sites can take a long URL and give you back a shorter URL without requiring registration. ...
notlong.com/links/ - 26k - Cached - Similar pages
Experience

Developing High Volume Commercial Web Sites
High Volume, Commercial Sites

- Millions of unique visitors per month
- Tens to hundreds of thousands of page views per hour
- 99.9+% uptime requirement
- Sites are critical to the:
  - success of the business
  - image of the business
  - valuation of the business
- Many, many millions of dollars riding on a web site
Planning

- Plan for growth
- Plan for peak usage
  - Peak hour can be 5-20+ times the average hour in a day
  - Peak minute can be hundreds of times the average
- Plan for hardware failure
- ...during peak usage
- Remove single points of failures
Architecture

- Load balancer in front
- Web servers:
  - Multiple (redundant)
  - Low cost, commodity boxes
  - It doesn't matter if some fail
- Database servers:
  - Expensive (redundant)
  - Failover or cluster
Development Process

- Individual web sites for each developer
  - On personal workstation
  - Fast change, test, debug cycle
  - Developers do not interfere with each other
- Central source repository and revision control
- Official QA platform and procedures
  - Lots of automated QA
- One button push to production when QA approves
Case Study

- Replace technology of an existing dot com
- New team hired to convert to open source software
- Goals:
  - Stability
  - Performance
  - Scalability
  - Uptime
  - Security
Legacy Site

- Technology:
  - IIS, ASP, Java running on NT
  - Oracle on Solaris

- Problems:
  - Capacity
  - Performance
  - Uptime
  - Stability
New Site

- Technology
  - Linux, Apache, mod_perl, Perl, Mason
  - Oracle on Solaris (not replaced)

- Team size:
  - 3 engineers, 1 dba

- Time to build new site:
  - 3 months (from date of hire)
Results

- Capacity / Scalability
  - Same hardware scaled to many times the traffic
- Performance
  - Good response even under high load
- High availability
  - Downtime practically eliminated
- Security
  - Not affected by prevalent Windows viruses
Growth
Cost Savings

- Reduced hardware costs
  - More performance from cheaper hardware
- Open source software
  - No initial cost
  - Reduced support costs
- Reduced team size by more than 50%
- Reduced maintenance $\Rightarrow$ More developer resources
- More new projects could be completed faster
Perl

- Critical part of achieving these goals
- Fast development
- Efficient programming
- Powerfully expressive
- Eliminates many types of bugs
- Large existing code base to draw on
(Disclaimer)

- Not trying to say:
  - Nobody can build great web services on Windows
  - Open source is always better than commercial
- Just sharing personal experience which is on the open source side
- I am trying to say:
  - Open source works
  - ...very well
  - ...for very serious applications
Challenges

- Case study had some interesting challenges...

- Requirements:
  - Change technology
  - No change to functionality or design
  - No downtime during development and launch
  - Seamless, live transition
Transition Plan

- Legacy support for:
  - Host names
  - URLs (even “/filename.asp”)
  - Query string parameters
  - Form parameters
  - Cookies
  - Database

- Backout procedures created
  - ...but not needed
Transition Results

- Users did not know the technology changed
  - ...even when it happened during their visit
- Clicks from old site to new site worked
- Users stayed logged in
- All data preserved
- No interruption of service
Software Architecture

Developing Web Applications With Perl
Compilation Phase

- Java, C#, C++ are compiled and run separately
- Perl is compiled on the fly
  - Simplifies development cycle
  - Adds a small startup cost
  - Increases performance of running code
  - Makes all source code available (has pros and cons)
Web Architecture

• Options for dynamic output generation include:
  • CGI
  • FastCGI
  • mod_perl
  • Apache::Registry
  • ...

CGI

- A common way to get introduced to Perl
- Pros
  - Easy to get started
  - Web server independent
  - Operating system independent
- Cons
  - Web server forks a new process to handle every request
  - CGI program must be compiled on every request
  - Not as powerful as mod_perl
FastCGI

- Web server communicates over network sockets to FastCGI server which runs Perl code

- Pros
  - Web server independent
  - Perl code compiled once at startup

- Cons
  - Few people use it
  - mod_perl is more advanced and better supported
mod_perl

- Best option for high performance, high volume

Pro
- Perl code compiled once at startup
- Perl code lives in same process space as Apache server
- Powerful hooks into Apache request and response process

Cons
- Only available with Apache
- More to learn to get started
- Not offered by cheap web hosting services
Apache::Registry

- An way to upgrade CGI programs to get benefits of mod_perl. (Don't start fresh with this.)

- Pros
  - Runs under mod_perl
  - Perl code compiled once at startup
  - Mostly compatible with CGI interface

- Cons
  - May not have access to all of the mod_perl features
  - Global and uninitialized variables may cause problems
Which Apache?

- Apache 2.x available and stable
- However, mod_perl 2 not yet completed
- If you are developing with Perl, stick with the tried and true (and still very supported) Apache 1.x
- Latest release Apache 1.3.28
Which Perl?

- Latest release of Perl (three weeks ago) is 5.8.1
- Two year old 5.6.1 is still very usable
Page Generation

- Lots of options including:
  - Mason (aka HTML::Mason)
  - Template Toolkit
  - EmbPerl
  - Apache::ASP
  - AxKit
  - HTML::Template
  - ...

- See Resources for article offering detailed comparison
Mason

- Allows Perl and HTML to be intermixed
- Supports modular component architecture (with OO)
- Pre-compiling of components for performance
- Caching (of HTML components and of data)
- Templates, filters
- Designed for large volume, commercial web sites
Mason Example

% foreach $item ( @cart_items ) {
<p>
name: <span><% $item{name} %></span> <br/>
price: <span><% $item{price} %></span> <br/>
% }

Template Toolkit

- Used by big sites including Slashdot and (late) eToys
- Pre-compiling of components for performance
- Back end programming done in Perl
- Forces separation of Perl code and user interface (HTML)
- Good for separate HTML coders and Perl coders
- Nice, clean template language
Template Toolkit Example

- [% FOREACH item = cart.items %]
  <p>
  name: [% item.name %] <br/>
  price: [% item.price %] <br/>
  [% END %]
Advanced Issues

• Likely order of bottlenecks for high volume server:
  • Database
  • Memory (!)
  • Network
  • CPU
Memory Issues

- Perl is memory hungry
- With mod_perl, each Apache process (child) contains all the Perl code and data
- Memory stays in use while feeding the generated page to the browser
- ...even if Perl generated the page quickly
Memory Answers

- Memory is cheap (relative to developer time). Buy more!
- Take advantage of shared memory (Unix/Linux):
  - Load all of your Perl code during Apache initialization
  - Pre-cache data during Apache initialization
- Use a split-proxy configuration
Perl

(Very Brief)
Introduction to Perl
My First Perl Program

- A simple (but complete) Perl program:

  ```perl
  print "Hello, Orange County!\n";
  ```
Variables

- Variable types:
  - `$scalar`
    - String
    - Number (integer, floating point)
    - Reference (to scalar, array, hash, or subroutine)
  - `@array` (ordered list of scalars)
    - `$array[$index] = $value`
  - `%hash` (unordered set mapping scalar key to scalar value)
    - `$hash{$key} = $value`
Subroutines

- `sub triple {`
  "my $number = shift;
  return $number * 3;
  }
- `sub trim {`
  "my $string = shift;
  $string =~ s/^\s+//;
  $string =~ s/\s+$//;
  return $string;
  }
- `print "Enter Name: ";`
  "my $name = <>;
  print trim($name), " should be earning ",
  triple($income), "\n";"
More Features

- Available if you desire:
  - Object oriented programming
  - Closures
  - Delegation
  - Tie
  - Operator overloading
  - Parser manipulation
  - Exception handing

- Persistence
- Taint checking
- C API
- Integration with other languages
- Embedding
- Extending
- On-the-fly definition of missing methods
One Liners

- Simple one liner:
  ```perl
  perl -pi.bak -e 's/Perl/Python/g' *.txt
  ```
- Replaces all instances of “Perl” with “Python” in all .txt files in the current directory
- Appends “.bak” extension to original versions of the files in case you didn't like the results and wanted to revert
Regular Expressions

- Perl regular expression features borrowed by Java, C#, and other languages

- Sample regular expression to match an IP address:
  \d{1,3} ( \. \d{1,3} ){3}
  1-3 digits followed by 3 more groups of 1-3 digits each preceded by a period

- Sample usage:
  
  if ( ! $input =~ /^\d{1,3}(\.\d{1,3}){3}$/ ) {
    die “Sorry, $input is not an IP address
  
  }
Good Advice

- When writing anything longer than one line, always:
  - Use the `-w` command line option
  - Put this line at the top of your program and packages:
    ```perl
    use strict;
    ```
  - Consider using the `-T` command line option
- By default, Perl is very lax and permissive. Very.
- The “use strict” and `-w` options are critical for catching mistakes like misspelled variables
- The `-T` adds a unique level of security
(-T) Taint Checking

- Problem:
  - User input cannot be trusted
  - ...a leading cause of web site attacks

- Perl's Solution:
  - All user input is marked “tainted”
  - Data touched by tainted data is itself tainted
  - Tainted data not allowed to be used in unsafe manner
  - Data can be explicitly checked and cleaned
Taint Checking

- Java security model is for untrusted code
  - Good for browser side
- Perl tracks and manages untrusted data
  - Good for server side
CPAN

Comprehensive Perl Archive Network
CPAN

- Comprehensive:
  - 5,348 modules
  - 3,160 authors
  - 2 GB of source code, ports, extensions, scripts, and documentation

- Perl: And is portable across operating systems

- Archive: Freely accessible; Online since 1995

- Network: Replicated on 229 machines around the world. Pick one (or more) that are close to you.
CPAN: Sample Modules

- CGI – CGI swiss army knife
- DBI – Vendor independent database interface
- Class::DBI – Easy object persistence with database
- HTML::Mason – Templating, web site building
- Template::Toolkit – Templating, web site building
- LWP – Web client programming
- Apache::Session – Session management for mod_perl
- XML::Simple – XML interface
CPAN: Sample Modules

- Real::Encode – Interface to Progressive Network's RealAudio
- Config::IniFiles – Read/write Windows INI files
- Memoize – Automatically cache results of functions
- Storable – Persistent data structure mechanism
- Statistics::ChiSquare – Chi Square test
- PHP::Include – Include PHP files from Perl
- GnuPG – Interface to GPG en/decryption
CPAN: Sample Modules

- GD::Graph – Create charts and graphs
- Email::Find – Find RFC822 email addresses in text
- DFA::Kleene – Kleene's algorithm for Discrete Finite Automata
- Class::Singleton – Implement the singleton pattern
- Net::IRC – Interface to IRC servers
- SOAP – SOAP implementation
- Business::CreditCard – CC number check digit test
CPAN: Sample Modules

- Date::Chinese – Calculations in the Chinese calendar
- Net::LDAP – Interface to LDAP servers
- Graph::Kruskal – Kruskal algorithm for minimal spanning trees
- Festival::Client – Interface to open source voice synthesizer
- Tie::File – Tie array to lines of a file
- Date::Tolkien::Shire - Hobbit calendar
CPAN: Sample Modules

- Math::Bezier – Solution of Bezier curves
- Math::BigInt – Arbitrary size integer math package
- Geo::WeatherNOAA – Current weather forecast
- Roman – Convert Roman numbers to/from Arabic
- ControlX10::CM10 – Control X10 modules (home automation)
- AI::Fuzzy – Extensions for Fuzzy Logic
- AI::NeuralNet – Back-prop neural net
CPAN: Sample Modules

- Time::Zone – Timezone info and translation
- Benchmark – Easy way to time code fragments
- Devel::DProf – Execution profiler
- Java: Perl front-end for JVM communication
- JavaScript – Allows JavaScript execution within Perl
- Math::Fleximal – Arithmetic in any base
- Math::Fourier – Fast Fourier Transforms
- Math::Matrix – Matrix arithmetic
CPAN: Sample Modules

- Crypt::Blowfish - Perl Blowfish encryption module
- Sub::Curry – Module to curry functions
- Test::Simple – Easy way to start writing unit tests
- VCS::PVCS – Interface to Intersolve's PVCS
- Log::Dispatch – Log messages to multiple outputs
- File::Find – Like Unix “find” command
- OpenGL – Interface to OpenGL
- ClearCase – ClearCase interface
CPAN: Sample Modules

- Mail::Internet – RFC 822 address manipulation
- Tcl – Complete access to Tcl
- Mail::Audit – Construct email filters
- BarCode::UPS – Produce PostScript UPC barcodes
- Net::POP3 – Interface to POP3 email servers
- Net::SMS – Send wireless SMS messages
- Net::DNS – Interface to DNS servers
- Net::FTP – Interface to FTP servers
CPAN: Sample Modules

- Audio::MPEG – En/decode MP3 audio
- Python – Interface Python API for embedded Python
- Mail::SpamAssassin – Identify spam mail
- Astro::SunTime – Calculate sun rise/set times
- CPAN – Interface to CPAN

- And 5,284 more modules...
CPAN: DBI/DBD

- DBI/DBD – Database Interface, plugins include:
  - MySQL
  - PostgreSQL
  - Oracle
  - Sybase
  - Informix
  - Illustra
- ODBC
- MS SQLServer
- Excel
- CSV files
- DB2
- InterBase
- Ingres
- Qbase
- PrimeBase
CPAN: DBI/DBD

- DBI/DBD continued...
  - Solid
  - Sprite
  - SQLFLEX
  - Unify
  - Msql
  - DtF/SQL (Max OS edition)
  - Altera SQL Server

- Empress RDBMS
  - Adaptive Server Anywhere
  - Xbase
  - ...

Where is CPAN?

- **www.cpan.org** [29] - Official top level web site
- **search.cpan.org** [31] - Find what you need
- CPAN.pm – The easiest way to get what you need
  (This is a Perl module, not a web site)

- [29] means go to http://reallyshort.com/29
Using CPAN

- Here's how to use CPAN.pm to install the Date::Tolkien::Shire module:
  
  perl -MCPAN -e 'install Date::Tolkien::Shire'

- Prompts for configuration the first time you use CPAN.

- Use the module you just installed:
  
  perl -MDate::Tolkien::Shire -e \
  'print Date::Tolkien::Shire->new(time)->as_string'

- Output (assuming today is October 18, 2003):
  Hevensday 26 Winterfilth 7467
Resources

Getting More Information
Resources: Books

- *Learning Perl* (3rd edition)
  by Randal Schwartz, Tom Phoenix

- *Learning Perl Objects, References & Modules*
  by Randal Schwartz, Tom Phoenix

- *Programming Perl* (3rd edition)
  by Larry Wall, Tom Christiansen, Jon Orwant

- *Writing Apache Modules with Perl and C*
  by Lincoln Stein, Doug MacEachern
Resources: More Books

- *Embedding Perl in HTML with Mason*
  by Dave Rolsky, Ken Williams
  Online: [www.masonbook.com](http://www.masonbook.com) [32]

- *Perl Template Toolkit (release date Nov 15, 2003)*
  by Darren Chamberlain, Dave Cross, Andy Wardley

- *Programming the Perl DBI*
  by Alligator Descartes, Tim Bunce

- *Object Oriented Perl*
  by Damian Conway
Resources: Even More Books

- *Perl Cookbook (2nd edition)*
  by Tom Christiansen, Nathan Torkington

- *Mastering Regular Expressions (2nd edition)*
  by Jeffrey E. Friedl

- *Practical mod_perl*
  by Stas Bekman, Eric Cholet
Resources: Periodicals

- The Perl Journal
  www.tpj.com [33]

- The Linux Journal
  www.linuxjournal.com [34]
Resources: Web

- **www.perl.com** [35] - By O'Reilly & Associates
- **www.perl.org** [36] - By The Perl Foundation
- **perl.apache.org** [39] - Main mod_perl site
- **www.perlmonks.org** [38] - Lots of friendly help from smart and experienced Perl experts
- **learn.perl.org** [40] - Good starting point for learners
- **jobs.perl.org** [41] - Open positions for Perl developers
Resources: More Web

- [www.masonhq.com](http://www.masonhq.com) [42] - Mason
- [www.template-toolkit.org](http://www.template-toolkit.org) [43] - Template::Toolkit
- [www.take23.org](http://www.take23.org) [44] – News/resources for mod_perl
Resources: Web Articles

- *Building a Large-scale E-commerce Site with Apache and mod_perl*, by Perrin Harkins and Bill Hilf
  http://perl.apache.org/docs/tutorials/apps/scale_etoys/etoys.html [45]

- *Choosing a Templating System*, by Perrin Harkins

- *Choosing the Right Strategy*, by Stas Bekman
  http://perl.apache.org/docs/1.0/guide/strategy.html [47]
Resources: Mailing Lists

- Perl mailing list list (over 200):
  http://lists.cpan.org/ [48]

- mod_perl users mailing list:

- Mason users mailing list:
  http://www.masonhq.com/resources/mailing_lists.html [50]

- Template Toolkit mailing list:
  http://template-toolkit.org/mailman/listinfo/templates [51]
Resources: User Groups

- Perl Mongers
  - oc.pm.org [52] - Orange County Perl Mongers
  - la.pm.org [53] - Los Angeles Perl Mongers
  - sandiego.pm.org [54] - San Diego Perl Mongers

- Perl Meetup
  - perl.meetup.com [55] – Make connections in your area
This Presentation

- Available online:
  - http://www.anvilon.com/talks/perlweb  [56]