Programming the Web with Perl

Chris Bates
Sheffield Hallam University, U.K.
c.d.bates@shu.ac.uk

March 7, 2001

Keywords Software, computer languages, web page design.

Summary

Web development is one of the growth areas of programming. The creators of e-commerce and small hobby sites use many of the same tools and techniques. Development languages become fashionable, wildly popular then fade away to be replaced by *the next big thing*. Some languages have such power and suitability that they deserve to be taken seriously and used widely. This paper presents Perl which has been used on many Web servers. In particular the focus here is on those features of Perl which make it so well suited for Web development and worth considering for both experienced programmers and beginners alike.

1 Introduction

Programming for the Web tends to be discussed and taught as if it is somehow different to other types of programming. There sometimes appears to be an expectation that using the Web presents a series of radical problems which are not encountered when, for instance, building traditional desktop applications. In reality, many modern programs include some of the features which we associate with Web based applications. Perhaps the myths about the difficulties associated with Web development have arisen because the Web has never been the exclusive development environment of traditional software engineers.

Software engineering as described in texts like [21] is the process of creating quality software solutions to well-defined problems. The software engineering process involves consultation with users, the creation of detailed designs, extensive testing and an iterative approach to the development process. Engineering led software development is the dominant philosophy in the industry today. An alternative *craft* view of software development dominated the industry into the 1970's, although it was rarely characterised as such. Treating software development as a craft means building code around primitive, or non-existent, design documents, with the code often regarded as providing its own documentation. Development as craft involves being adaptable during the development process and attempting to use the best tool for the job rather than deciding upon tools and techniques before writing any code. Development as craft is often called *hacking*, although that term has unfortunate connotations outside of programming.

Software engineering grew out of the many failures of the industry in the 1960's and 1970's when large systems regularly failed to meet their design goals. It is clear that much software and many systems continue to *fail* when outcomes are compared to original goals, but the amount of successful software available today far exceeds the amount which fails. Developing software as a craft has a long and illustrious history exemplified by the "hacker" tradition which has given us the GNU tools[14], the Linux kernel[5], and the Emacs editor[19].

Web development sometimes happens in an engineering context. Large e-commerce Web sites tend to be designed and tested thoroughly and are often built using industry standard technologies. The combination of plentiful UML documentation[9], Microsoft Active Server Pages[26], and an Oracle database has much to recommend it to conservative IT managers. Such technologies have a reputation for stolid reliability and good support from their manufacturers, although that clearly comes with a hefty price tag attached.

Much Web development is undertaken by non-programmers. These people may be designers who need to add some code to a site, the *nominated* person within a small company or dedicated hobbyists with a good idea and some free Web space. Developers who are not professional programmers are likely to look widely for solutions. They may be less likely to hide behind a brand-name than industry managers will, and are more open to the use of free software [12], or

open-source solutions. A plethora of tools have been created for, and *by*, this particular group.

Even non-specialists will be aware of the Java programming language from Sun Microsystems [7], and its use on the Web. Many other languages have been developed or adapted or use on Web servers. These languages are often pleasingly free of the compromises that the designers of more *industrial* languages must accept. Perl is just such a language. This paper presents Perl those features of Perl which make it ideal for Web development. The intention is to show those who have never looked at it what the language has to offer, and to demonstrate that Perl is worth looking at again if the, admittedly cryptic, syntax has proved off-putting in the past.

Section 2 demonstrates that scripting languages are appropriate even on high-load systems, section 3 briefly describes the Perl language. In section 4 the CGI.pm module is described, finally sections 5 shows that Perl can be used to build complex template-based sites.

2 Scripting

Programming for Web sites usually involves either adding dynamic aspects to HTML pages using JavaScript or creating entire dynamic Web sites using programs on the server. The processing of data on a Web server is now common place in many large Intranet and e-commerce Web sites. Data is collected from users via an HTML form, some characters such as spaces are replaced with *escape sequences*, and the data returned to the server. The data is extracted from the return messages and passed to a program which replaces the escape sequences with the original characters then processes the data.

Server-side programs can be written in just about any language, the only restriction being that the server is actually able to execute the finished program. The main facility which a good Web programming language needs is the ability to handle text strings. Data received from Web forms comes as strings and the Web pages which are sent back to the browser are primarily textual. Choosing a development language which has good facilities for the handling of text is therefore important, but sometimes developers and managers worry about performance and suggest that a fast language such as C should be used. Today this is rarely an issue as technologies exist which

give rapid execution speeds even when using interpreted languages such as Perl.

3 Why Perl?

Perl is probably the dominant language in Internet development today. Perl was created in the late 1980's by Larry Wall, who also co-authored [25] which is the standard textbook on the language. Right from its inception Perl was a language with excellent text manipulation facilities, in fact that was its original purpose and remains central to its use and development today.

Perl is ideally suited to the creation of Web applications. The language handles text very well, its regular expression engine alone is worth the effort of learning the language. If you want to search text strings, make changes and then get those changes back out to a user, Perl and regular expressions make the ideal combination. [13] describes Perl and other languages which implement regular expressions, but only in Perl makes them a central feature.

Documentation is yet another strong feature of Perl, The standard distribution comes with thousands of pages of online documents which can be accessed using a command-line facility called `perldoc`. Unix distributions have much of this material available as `man` pages, in the distribution from [2] which runs on Microsoft Windows it is provided as HTML pages.

Many introductory Web programming texts such as [8], [15] and [11] include tutorial material on the use of the language. Slightly older texts such as [20] and [16] still contain some useful information but the pace of Web development is so fast that books can easily get left behind. There are also a number of good Perl programming texts available including [17], [18] and [25]. [10] is a vital source of hints, tips and useful code for anyone who is serious about their Perl programming.

3.1 The Comprehensive Perl Archive Network

Many important modules such as `CGI.pm` are part of the core Perl distribution, updates are available from [4] as are numerous other useful libraries. Perl really demonstrates many of the benefits of co-operative development using the Internet. Programmers around the world work to create

not only the core language but vast libraries of code *modules* which perform myriad complex tasks. Installing and using these is generally not an onerous task although if a module is being actively developed, handling regular updates can be time consuming. Fortunately many of the most important modules for Web development are relatively stable and will only need updating occasionally.

Thousands of Perl modules are made freely available at the Comprehensive Perl Archive Network (CPAN) which is a worldwide network of FTP sites. The CPAN archives can be accessed centrally at [4] via the Web, but are also available via anonymous FTP. Once a running installation of Perl exists on a machine the most effective way of updating it and adding new modules is via its inbuilt facilities. Typing `perl -MCPAN -e shell` at a command prompt brings up an interactive utility which can be used to access the archive. There's not even a need to find a suitable mirror, that is done automatically by CPAN.

4 CGI.pm

Extracting the data returned from a form and presenting it to an application is a relatively trivial task. It is also something which almost every Web script needs to do at some point. Whether a user is uploading data onto a server, purchasing from an online store or simply requesting a page, this process lies at the heart of Web programming. Rather than write the code to perform these processes in each script, library routines are used. These have many benefits, not least of which is robustness. Even the best programmers make mistakes but in a well tested library those mistakes will have been discovered and removed. The library of choice for Web developers is `CGI.pm` written by Lincoln Stein, [22] and [23]. It handles all of the complexity of Web development and presents an extremely usable interface to application developers.

Many people would use Perl on Web servers even if `CGI.pm` did not exist, but it's easy availability and excellent functionality go some way to compensating for the obscure syntax of the language. Like many Perl modules, `CGI.pm` is available either through its object-oriented or procedural interfaces. This makes slotting it into existing code very simple, but also means that

developers can choose the approach with which they are most comfortable.

`CGI.pm` has two purposes: the creation of Web pages, especially those which contain forms at run-time, and the parsing of data returned from Web forms. The creation of HTML pages, especially the forms within them, is accomplished by *shortcuts* which are simple function calls. The intention was that by providing shortcuts coding errors would be reduced. In reality, using the shortcuts is often more complex than writing the code out fully.

Many developers end up using only a subset of the facilities which the module provides. Its ability to seamlessly extract data from forms, including files and other multi-part data, is always used. The module includes excellent boilerplate functions which create the headers and footers for both HTTP message and HTML page when building a Web page to return to the browser.

Few scripts work perfectly *out of the box*. Most have some errors which prevent them working properly. The `CGI.pm` module has a feature which means that scripts can be tested and debugged on the workstation *without* needing a Web server. This is such a useful facility that it should be used in most development situations. Scripts which create errors when running on a Web server tend to write those errors straight to the server log where they are inaccessible to most developers who do not have system administrator privileges. All Perl scripts should be developed with the warning flag set, the first line of every script should be:

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

The `w` flag turns on warnings about potential syntax errors, the`T` flag initiates taint checking which looks for potentially dangerous uses of data received from a form. The warnings flag can create many errors even from a script which appears to run perfectly. Debugging is needed to remove those errors before labelling the script as *production code*. If a script is run from the command line, `CGI.pm` gives the opportunity to enter parameter names and values in pairs. When all parameters have been entered, `Control-D` runs the script with errors displayed on the terminal.

4.1 Handling Error Messages

A running script can create errors as part of its normal operation. As previously noted these tend to be written into the server logs which are not accessible by normal users on most systems. In addition, the server will not usually return a message back to the browser. The user has no way of knowing that the Web server just failed to handle their request and may assume that all is well. One way of dealing with this problem is to redirect the error message back to the originating browser. This is done by using the `CGI::Carp` module:

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

4.2 Perl and Databases

Large Web sites, especially those which are run as commercial endeavours, create and store massive amounts of data. This storage can be done in a variety of ways including flat files, XML files or relational databases. The use of databases has numerous advantages but accessing them can be difficult. The database will often be running on a different server to the Web application, database commands are often written in SQL. Ideally a Web application will be able to talk to any database backend although changing the backend is unlikely. Perl has a database neutral system called `Perl::DBI`.

DBI is an API which provides a library of routines which provide a consistent interface for accessing any relational database. The actual database access is provided by a database specific driver. Therefore the only databases which can be used with DBI are those for which someone has taken the time to develop a driver. It is worth noting that DBI is not a Web only technology, it can be used in any Perl application which requires database access, however it is an important technology on many commercial Web sites.

Unix systems have numerous databases and require myriad drivers, under Windows the whole process is simplified somewhat. Microsoft have a technology called ODBC which gives a consistent interface to any database system. Using DBI on Windows is therefore very simple: simply use the solid and reliable ODBC backend which will talk to the operating system's ODBC

manager.

5 HTML::Mason, Web Templates in Perl

Web site development has moved from a purely scripted model in recent years. Scripted sites tend to embed HTML code inside scripts but for many non-programmers this can seem overly complex. These developers tend to think in terms of documents rather than scripts and prefer to use *templating systems*. The best known templating systems are [1] and [6], both of which are widely used. The number of sites using ASPs is not surprising given the close integration that technology has with Microsoft's server products, PHP is much more of a grass roots movement whose use has spread almost by word-of-mouth. Both show that plenty of people want to embed small scripts inside HTML pages.

Perl developers are usually going to be happier working with a scripting model of development but this may not always be an appropriate solution. `HTML::Mason`, available from [24], is a recent development which combines a template-based model with the power of Perl. Of course this can be done using ASP and Perlscript, but `HTML::Mason` has the advantages of being a pure Perl solution which is hosted, and can be closely integrated with [3]. The documentation supplied with `HTML::Mason` states that it

> is a tool for building, serving and maintaining large Website... serving dynamic content.

Mason bases sites around *components* which are files containing a mixture of HTML, Perl and special Mason commands. Components may represent complete pages but more commonly they are smaller objects which can be embedded within each other to create a complete page. This idea is powerful and intuitively sensible to anyone who is used to object-based design or programming. Breaking a complex page into it's constituent pieces means that navigation bars, headers or page footers can be changed without affecting the rest of the page. It also means that a developer can create a component which represents the structure of the whole site in a single file, but embed that content into every page on the site.

Mason was designed to integrate closely with the Apache Web server. It does this through Apache's commonly used `mod_perl` extension which is widely used to improve the performance of CGI scripting solutions. The module answers one of the commonest objections raised to the use of scripting languages on Web servers: poor runtime performance. In traditional CGI work, each time that a script is called the interpreter and script must be loaded before execution. With `mod_perl` the interpreter is only loaded the first time it is needed, it then remains resident. Similarly scripts are loaded and interpreted only once, the interpreted, usually bytecode, version is executed thereafter. [24] claims speedup from 400 to 2,000 per cent is possible using `mod_perl`. Given the relatively small size of most scripts this speedup brings run times towards the level of compiled code.

6 Conclusions

Programmers are as susceptible to the vagaries of fashion as anyone else. New technologies supersede old ones through hype or boredom. Often the new saviour is no better than the old and after a few years developers drift back to their previous language or system. This dynamic can be seen in the ongoing movements between C++ and Java or mainframe and client-server development.

This paper has shown a tiny subset of the features of Perl. As a language Perl has much to recommend it for many development tasks, its suitability as a development language for the Web is clear. Newer technologies such as PHP, Java servlets or ASP may be more fashionable or more widely hyped but they lack the range of applications, extensions and facilities which Perl provides. Anyone who needs to develop a major Web-based application would benefit from using Perl as their server-side language in reduced development times and increased robustness.

References

[1] About Active Server Pages. `http://msdn.microsoft.com/library/psdk/cdo/_olemsg_about_active_server_%pages.htm`.

[2] The ActiveState Web Site. `http://www.activestate.com`.

[3] The Apache Group. `http://www.apache.org`.

[4] The Comprehensive Perl Archive Network. `http://www.perl.com/CPAN`.

[5] The Linux Kernel Archives. http://www.kernel.org.

[6] PHP Hypertext Processor. http://www.php.net.

[7] Sun Microsystems Servlet Page. http://www.java.sun.com/products/servlet/index.html.

[8] Chris Bates. *Web Programing: Building Internet Applications.* John Wiley and Sons, 2000.

[9] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide.* Addison-Wesley, 1999.

[10] Tom Christiansen and Nathan Torkington. *Perl Cookbook.* O'Reilly and Associates, 1999.

[11] Dave Cintron. *Fast Track Web Programming.* John Wiley and Sons, 1999.

[12] Free Software Foundation. Free Software Foundation. http://www.fsf.org.

[13] Jeffrey E. F. Friedl. *Mastering Regular Expressions.* O'Reilly and Associates, 1998.

[14] GNU. The GNU Project. http://www.gnu.org.

[15] Shishir Gundavaram. *CGI Programing on the World Wide Web.* O'Reilly and Associates, 1996.

[16] J.M. Ivler and Kamran Husain. *CGI Developer's Resource.* Prentice Hall, 1997.

[17] Randal L. Schwartz. *Learning Perl.* O'Reilly and Associates, 1997.

[18] Randal L. Schwartz, Eric Olsen, and Tom Christiansen. *Learning Perl on Win32 Systems.* O'Reilly and Associates, 1997.

[19] Free Software Foundation. The Emacs Editor. http://www.gnu.org/software/emacs.

[20] Selena Sol and Gunther Birzniecks. *Instant Web Scripts With CGI Perl.* M and T, 1996.

[21] Ian Sommerville. *Software Engineering.* Addison-Wesley, sixth edition, 2000.

[22] Lincoln D. Stein. The Home of CGI.pm. http://www.genome.wi.mit.edu/ftp/pub/software/WWW/cgi_docs.html.

[23] Lincoln D. Stein. *How to Set-up and Maintain a Web Site.* Addison-Wesley, 1997.

[24] Jonathan Swartz. Mason HQ. http://www.masonhq.org.

[25] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl.* O'Reilly and Associates, 1996.

[26] A. Keyton Weissinger. *ASP in a Nutshell.* O'Reilly and Associates, 1999.