Server-side scripting languages

PHP, Perl, Java servlets -- Which one's right for you?

Erik Zoltán (erik@zoltan.org)
Advanced Systems Engineer, EDS
April 2001

Still can't decide whether to use PHP scripts, Perl CGIs, or Java servlets for your next Web development project? This article will help you decide by providing a side-by-side comparison of the functioning source code of all three languages. The three simple example programs provided take you from the most basic server-side scripts through object orientation to a simple Web storefront presenting product information to a user. Reading this article will give you a good idea of the difference between these three languages, and a better idea of which one is right for you.

Contents:

This article explains how PHP scripts, Perl CGIs, and Java servlets work; it also covers the issues that separate the three languages. You don't need to know the languages in order to comprehend this piece, but you do need to have a passing familiarity with HTML to make reading it worth your while.

If you have a great deal of experience in any of these languages, you'll notice that this article neither discusses the in-depth details and advanced alternatives that an expert might choose, nor does it introduce many of the technical terms and underlying language features that appear confusing when you are familiarizing yourself with a programming language (of course, this is what makes the language interesting after years of programming with it). *This is intentional*.

Example 1: hello

Our first example is the Web developer's version of "hello world" in all three languages. It simply assigns a value to two variables and uses those variables in the text of a bare-bones Web page. The point of this example is to show you how the three languages work without getting into any real content issues.

Listing 1 shows the HTML source code we're trying to produce.

Listing 1: HTML version

```
<html>
 <head>
  <title>My First Script</title>
 </head>
 <body>
  <h1>My First Script</h1>
   <p>Welcome to my first script.</p>
 </body>
</html>
```

As you can see, it has a title and a heading line that both contain the same text. It then has a paragraph line containing somewhat different text.

PHP version
The PHP version is quite simple (see Listing 2). It looks a lot like the HTML source, but with a couple of extra twists added.

Listing 2: hello.php3

```
<?php
        $title = "My First Script";
        $greeting = "Welcome to my first script.";
?>
<html>
 <head>
  <title><?php echo($title) ?></title>
 </head>
 <body>
  <h1><?php echo($title) ?></h1>
   <p><?php echo($greeting) ?></p>
 </body>
</html>
```

A PHP script starts out in HTML mode, then switches over to PHP mode when it encounters a <?php starting tag. This script assigns values to the $title and $greeting variables. All PHP variable names begin with a $-sign. It then switches back to HTML mode when it hits a ?> ending tag. It pops into PHP mode several more times to echo the values of those variables into the output stream.

Perl version
The Perl version in Listing 3 is very similar to the PHP version:

Listing 3: hello.cgi

```
#!/usr/bin/perl
$title = "My first Script";
$greeting = "Welcome to my first script.";
print <<EOF;
Content-type: text/html

<html>
 <head>
  <title>$title</title>
 </head>
 <body>
  <h1>$title</h1>
  <p>$greeting</p>
 </body>
</html>
EOF
```

The first line, #!/usr/bin/perl (which may vary slightly from system to system), tells Unix that this file contains a Perl script and not something else. (Of course, Perl runs just fine on non-Unix servers, which generally interpret the first line as a comment.) This program contains two variable assignments. The third statement, print<<EOF, sends all the succeeding lines directly to the browser as input until it reaches the EOF tag.

A Perl program needs to obey the *Common Gateway Interface* (CGI) protocol. This means that you have to include the MIME statement Content-type: text/html followed by a blank line before the HTML begins; this tells the browser what kind of input it's receiving. PHP handles this automatically, but Perl and Java let you control it directly. Notice, too, that Perl automatically substitutes variable references such as $title and $greeting. Unlike PHP, with Perl you don't have to use an echo statement. (If you really wanted it to say $title in the output text, you'd have to escape the $-sign by typing \$title instead.)

Java Servlet version
Perl and PHP are interpreted languages while Java is a compiled language. This means that when you modify your Java code, you have to run the Java compiler before you can try out your changes. In order to write Java servlets, you also need to have a copy of the Java Servlet Development Kit (JSDK) (see the Resources section under Java). It's not enough to have the simple Java Development Kit (JDK). For example, on a Windows 98 machine you have to download the JSDK and start the servlet server process before trying out your servlet. You might then have to stop and restart the server process each time you recompile your code for the changes to take effect. This adds process overhead to your development time, especially during the debugging phase when you may be making a lot of small changes.

Listing 4: HelloServlet.java

```
import java.io.*;
import javax.servlet.*;

public class HelloServlet extends GenericServlet {

        public void service (ServletRequest request,
                        ServletResponse response)
                        throws ServletException, IOException {
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                this.MyContentHere(pw);
                pw.close();
        }

        public void MyContentHere(PrintWriter pw) {
                String title = "My First Script";
                String greeting = "Welcome to my first script.";
                pw.println("<html>");
                pw.println(" <head>");
                pw.println("  <title>" + title + "</title>");
                pw.println(" </head>");
                pw.println(" <body>");
                pw.println("  <h1>" + title + "</h1>");
                pw.println("   <p>" + greeting + "</p>");
                pw.println(" </body>");
                pw.println("</html>");
        }
```

As you can see in Listing 4, Java examples are somewhat longer. After the obligatory import statements, you need to create a new subclass of GenericServlet. In this case, I've chosen the name HelloServlet. GenericServlet provides a lot of automatic functionality that we don't need to worry about in this article. But we do need to override the service method in order to get our servlet to do something. Note the statement response.setContentType("text/html");, which is equivalent to the MIME header that we printed out directly in the Perl example. Rather than embed my functionality directly in the service method, I've created a new method called MyContentHere, which the service method invokes.

The MyContentHere method looks quite similar to the Perl and PHP scripts, but Java imposes a couple of philosophical and syntax differences. It accepts a PrintWriter object pw as input and sends its output to the browser by invoking the println method of that object. It needs to define the class of every object it uses, including String objects, and it uses the + operator to concatenate strings together.

Example 2: objects

Although the Java version of the first example forced us to use objects, the Perl and PHP examples didn't. The second example of this paper demonstrates the object-oriented features of all three languages. I've created a Month class and implemented two attributes, the name of the month and the number of days. There is also a constructor that allows a new instance to be created while specifying the values of both attributes.

> **Note:** while my explanations focus only on the class definitions, you can click on a link to get the full source code of each example if you want more details.

PHP objects
PHP is simple when it comes to objects, as Listing 5 shows.

Listing 5: Month class in PHP

```
class Month {
        var $name;
        var $days;
        function Month($n, $d) {
                $this->name = $n;
                $this->days = $d;
        }
}
```

PHP lets you create the class in the same file as the rest of the script, or you can have it in a different file for greater reusability. To create an instance of the Month class in the variable $thismonth, you'd now use the statement $thismonth = new Month("January","31");. After that, you can reference the properties with a statement such as: echo($thismonth->name . " has " . $thismonth->days . " days.");.

Click here to see the PHP script monthlist.php3.

Perl objects
The PHP approach to defining and implementing classes is simpler and cleaner than the Perl approach. (PHP had the advantage of coming later, allowing its creators to include the Perl features they wanted and change the ones they didn't.) On the other hand, Perl classes work fine and there's no reason not to use them. The syntax is a little odd, but if this bothers you, then you're certainly not ready for Perl anyway. Listing 6 provides the code

Listing 6: Perl module Month.pm

```
#!/usr/bin/perl

package Month;

sub new {
   my ($class, $name, $days) = @_;
   return bless {
      monthname => $name,
      days => $days
   }, $class;
}
1
```

Apparently, Perl already had some of the ingredients of object orientation before objects were added to the language. Rather than choose a simple abstract syntax like the one we can see in PHP above, Perl developers decided to build on the existing features. The result is that the Perl syntax for using objects is much less clear-cut and harder to remember.

So when you define a class Month in Perl, you need to create a package in a file called Month.pm (the "pm"

extension stands for "Perl Module"). When you assign values to attributes, you're really creating a hash (which is a collection of name/value pairs that acts as a convenient symbol table) and "blessing" it into an externally-accessible form. The actual CGI (which is in the file monthlist.cgi) needs to use a statement like require Month; in order to import the Month class into the program. The number 1 at the end of the file is not a typographical error. It's needed for the require statement to work.

The new subroutine acts as a constructor. It accepts three parameters which are stored in the @_ array (the Perl approach to parameter passing is likewise unorthodox but effective.) Note that you can copy the three array elements of the @_ array into temporary scalar variables with the statement my ($class, $name, $days) = @_;. This is a *really cool* feature. Perl does interesting things with arrays.

To create an instance of the Month class in the variable $thismonth, you'd now use the statement $thismonth = new Month("January","31");. After that, the properties might be referenced with a statement such as: print $thismonth{monthname} . " has " . $thismonth{days} . " days.");.

Click here to see the Perl module Month.pm.

Click here to see the CGI script monthlist.cgi.

Your own Java objects
While classes and objects are optional in Perl and PHP, they are mandatory in Java. Java allows you to create your new class in the same source code file as your servlet, or you can create it in a different file. However, a different file compiles each class into its own .class file. Java certainly has the most comprehensive and robust object syntax of the three languages.

Listing 7: Month class in Java

```
class MyMonth {
        String name;
        int days;
        MyMonth(String n, int d) {
                name = n;
                days = d;
        }
}
```

I decided to name the class MyMonth instead of Month to illustrate that in Java it is sometimes necessary to avoid confusion with another class name that might exist in some library you happen to be importing. To create a new MyMonth object in the variable thismonth, you'd use the statement MyMonth thismonth = new MyMonth("January",31);. After that, the properties might then be referenced with a statement such as pw.println(thismonth.name + " has " + thismonth.days + " days.");.

Like Perl, Java also has a shorthand syntax to define an entire array at once. However, it lacks some of Perl's quirky and powerful array manipulation features, not to mention hashes. Still, if you miss those features, you can write your own classes to provide whatever services you want.

Click here to see the full Java version.

Example 3: Web store
This example presents a very simple Web storefront. The user selects from a menu of categories along the top of the page. A list of products is displayed on the left hand side of the page below the category list. When the user selects a product, the main body of the page contains a full product description.

Supporting files

File Name	Description
sections.txt	A list of the sections of the store. Examples might include "Main," "Food," and "Candles." Each section name appears by itself on a separate line.
sMain.txt	A list of the products in the "Main" section. If you have a section called "Electronics," then the file Electronics.txt must contain a list of the products in that section. Each line in this file contains the name of one product in that section.
pMain1.txt	Contains the HTML code to display information about a single product. In this case, it's product 1 in the section called "Main." The eighth product in a section called "Fun" would be in pFun8.txt.

Functions/Subroutines/Methods

Name	Description
`display_menu(section)`	Displays a menu of all the sections in the store. This is a horizontal menu appearing along the top of the page. Each section is a link, except the currently selected section which uses a different background color but is not a link.
`product_list(section,product)`	Displays a vertical list of products in the currently selected section. This menu appears along the left-hand side of the page. Each product name is a link, except the currently selected product which uses a different background color but is not a link.
`display_product(section,product)`	This fills up the main body of the page with the currently-selected product in the currently-selected section. The contents of a text file appear in the lower left part of the page. Content authors can embed any HTML text they like (so your author must be someone you trust).
`main code`	Outputs the outer shell of the Web page and the display_menu, display_product, and product_list functions/subroutines or methods.

Click here to try out the PHP version of the Web store.

Click here for the full PHP source code.

Click here for the full Perl source code.

Click here for the full Java source code.

Conclusion

Perl creator Larry Wall says that he designed Perl to make the easy things easy while making the difficult things possible. By contrast, PHP makes some of the easy things even easier and still gets the harder ones done. Java makes the big things more scalable and makes everything more object-centered.

On the other hand, Perl makes a lot of things weirder. It's a simple language to learn, once you can get past the culture shock. PHP has removed much of the quirkiness from Perl, but in the process it has lost just a little of the power. Java makes the hard things easier, especially since a lot of them have already been done and can be included or inherited efficiently. But it also makes some of the easy things a bit harder than perhaps they should be.

You make the call.

Resources

PHP

- PHP.net is the official home page of the PHP language. Here you can download the latest version of PHP for free, and check out what's going on in PHP development. There is an FAQ, a manual, and numerous other resources.
- PHPbuilder.com is geared toward developers; it contains numerous articles and code samples, job postings, and links to PHP resources.
- Webmonkey's PHP Section contains a number of articles on PHP starting at the introductory level.
- The PHP Pocket Reference by Rasmus Lerdorf. O'Reilly & Associates, January 2000 (ISBN: 1565927699). This pocket guide stands out because it was written by the creator of PHP, and because it actually contains some real source code examples. You can't actually learn PHP just by reading the pocket guide, but you can come close.
- The Professional PHP Programming by Jesus Castagnetto et al. Wrox Press, September 1999 (ISBN: 1861002963). An in-depth reference with extensive working example applications that you can download from their Web site.

Perl

- Perl.com contains a variety of links to useful Perl information and groups, as well as information about the latest development efforts.
- Freecode.com contains a lot of free source code examples, many of which are written in Perl.
- Perl and CGI for the World Wide Web, Visual Quickstart Guide by Elizabeth Castro. Peachpit Press, November 1998 (ISBN: 020135358X). This is an *excellent* book for learning Perl and I strongly recommend it. It contains many simple examples and the book itself is easy to follow. You'll be up and running in no time, really.
- Perl 5 Pocket Reference by Johan Vromans. O'Reilly & Associates, May 2000 (ISBN: 0596000324). This is a good quick reference if you already know the language, but not the way to learn the language.
- The Perl CD Bookshelf, O'Reilly & Associates, August 1999 (ISBN: 1565924622). This contains a copy of *Perl in a Nutshell*, as well as six books on CD. It's the most bang for the buck, as long as you don't mind pointing and clicking your way to the correct information. The six books are presented in HTML format with an automatic search engine.

Java

- Java.Sun.Com is the definitive source for information about developing in Java. It is also the place to find a copy of the Java Servlet Development Kit (JSDK). Sun Microsystems maintains this Web site and you absolutely need to refer to it if you're working in Java.
- ZDNet Developer contains a lot of information about Java development and numerous examples with source code.
- Java for the World Wide Web, Visual QuickStart Guide by Dori Smith. Peachpit Press, September 1998 (ISBN: 0201353407). Not as good as the Perl version mentioned above. This book is well-written, but the author has written code examples in such a way that they don't compile unless you change the HTML code or place them in a certain directory, something she doesn't bother to mention. This also applies to the examples you can download from her Web site. It is oriented towards applets rather than servlets, but it does serve as a good introduction to the Java language itself.
- Java2, the Complete Reference by Patrick Naughton and Herbert Schildt. Osborne/McGraw Hill, February 1999 (ISBN: 0072119764). This is a truly complete reference and it contains descriptions of the methods you can call for most of the important classes in Java 2. It also contains a special chapter

to get you started writing servlets.

About the author

Erik Zoltán has been writing code professionally since 1990. He's a full-time telecommuter living in Leominster, Massachusetts. You can e-mail him at erik@zoltan.org or view his home page at www.zoltan.org. Erik can frequently be seen driving around in a 1996 Saturn painted in Vincent Van Gogh's The Starry Night.

What do you think of this article?

Killer! (5) Good stuff (4) So-so; not bad (3) Needs work (2) Lame! (1)

Comments?

Submit feedback