Comparing Server-Side Web Languages

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK

Outline

Comparing Server-Side Languages

Empirical Comparisons

Which Server-Side Web Language is best?

What about ...

- Php -

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl -

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET -

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.
- Python -

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.
- Python - Object oriented multi-purpose scripting language. Really good for server-side tasks! But not that well known.

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.
- Python - Object oriented multi-purpose scripting language. Really good for server-side tasks! But not that well known.
- Java Servlets -

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.
- Python - Object oriented multi-purpose scripting language. Really good for server-side tasks! But not that well known.
- Java Servlets - Java server-side programming, requires Apache Tomcat engine or similar.

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.
- Python - Object oriented multi-purpose scripting language. Really good for server-side tasks! But not that well known.
- Java Servlets - Java server-side programming, requires Apache Tomcat engine or similar.
- JSP -

Which Server-Side Web Language is best?

What about ...

- Php - Probably currently most popular! But not a general purpose language.
- Perl - Older than Php. More difficult to use, but general purpose. Good for general Unix system admin tasks.
- ASP.NET - Microsoft's server-side language; commercial; platform dependent.
- Python - Object oriented multi-purpose scripting language. Really good for server-side tasks! But not that well known.
- Java Servlets - Java server-side programming, requires Apache Tomcat engine or similar.
- JSP - Sun's version of Java server-side programming.

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?
- Flexibility: are there many ways to achieve a solution?

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?
- Flexibility: are there many ways to achieve a solution?
- Usability: how difficult is it to learn and to use the language?

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?
- Flexibility: are there many ways to achieve a solution?
- Usability: how difficult is it to learn and to use the language?
- Security: is security built into the language or do programmers have to write code to ensure security? Are there security holes in the language?

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?
- Flexibility: are there many ways to achieve a solution?
- Usability: how difficult is it to learn and to use the language?
- Security: is security built into the language or do programmers have to write code to ensure security? Are there security holes in the language?
- Speed of execution: how fast is a script executed?

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?
- Flexibility: are there many ways to achieve a solution?
- Usability: how difficult is it to learn and to use the language?
- Security: is security built into the language or do programmers have to write code to ensure security? Are there security holes in the language?
- Speed of execution: how fast is a script executed?
- Generality: is the language special purpose or general purpose?

Differences among Server-Side Languages

- Embedding: is HTML embedded into the code (using print statements) or is the code embedded into HTML (using templates)?
- Flexibility: are there many ways to achieve a solution?
- Usability: how difficult is it to learn and to use the language?
- Security: is security built into the language or do programmers have to write code to ensure security? Are there security holes in the language?
- Speed of execution: how fast is a script executed?
- Generality: is the language special purpose or general purpose?

How do Perl and Php compare with respect to these categories?

CGI - Common Gateway Interface

HTML requests are handled by a webserver, such as Apache.

There are different ways in which server-side scripting languages can interact with webservers. CGI is fairly old-fashioned, slow, but simple. Problems of CGI are:

> *each new CGI request spawns a new process and*
> *session tracking is difficult.*

Webserver Extensions

Webserver extensions (such as mod_perl and mod_php) are faster than CGI because the server-side language is loaded into Apache instead of restarting it new for each request.

Database connections and session parameters can be kept persistent.

Although most of the Perl code is the same for use under CGI or mod_perl, mod_perl requires a bit more programming experience to be used safely.

Empirical Comparisons

Below are some examples from a paper by Lutz Prechelt

An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl

Even though the paper is a bit old (from 2000), it illustrates the kinds of methods which can be used to compare programming languages.

From Prechelt (2000):

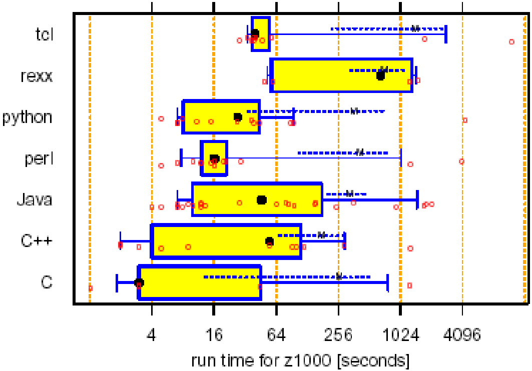

Figure 1: Program run time on the z1000 data set. Three programs were timed out with no output after about 21 minutes. The bad/good ratios range from 1.5 for Tcl up to 27 for C++. Note the logarithmic axis. ◆◆

From Prechelt (2000):

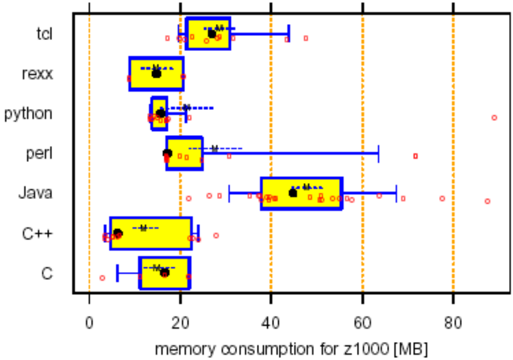

Figure 4: Amount of memory required by the program, including the interpreter or run time system, the program itself, and all static and dynamic data structures. The bad/good ratios range from 1.2 for Python up to 4.9 for C++.

From Prechelt (2000):

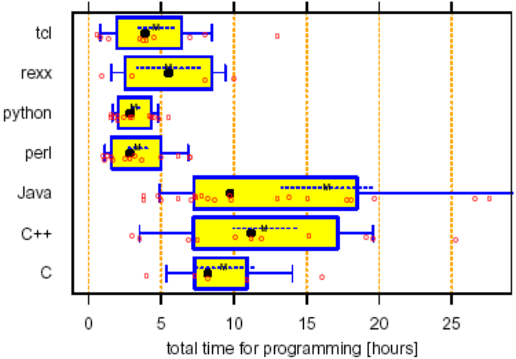

Figure 6: Total working time for realizing the program. Script group: times as measured and reported by the programmers. Non-script group: times as measured by the experimenter. The bad/good ratios range from 1.5 for C up to 3.2 for Perl. Three Java work times at 40, 49, and 63 hours are not shown. ⬌

A similar paper is Gousios & Spinellis (2002) 'A Comparison of Portable Dynamic Web Content Technologies for the Apache Server'.

The paper explains in more detail how CGI, templating (Php), servlets and so on work. It contains sample code for several different languages and some benchmark results for comparisons.

Both papers can be downloaded from the module website.

Questions for the Gousios/Spinellis paper

- What are the four different approaches for server-side web languages?
- Which of the following three use essentially the same scripts: Perl/CGI, FastCGI, mod_perl?
- Which of the following three is slowest/fastest: Perl/CGI, FastCGI, mod_perl?
- What is the main difference between PHP/ASP and Perl/Servlets?
- Which of the following six is worst/best at maintaining states: FastCGI, Perl/CGI, mod_perl, ASP, Servlets, PHP?

Questions for both papers

- Which server-side technologies are best for fast prototyping?
- Is there any technology among the following that runs so slowly that it should not be used for complicated applications: Perl/CGI, FastCGI, PHP, Java Servlets, Python, C++?
- Describe a low-cost method for comparing different server-side technologies.