Introduction to PHP

Web Programming

Uta Priss
ZELL, Ostfalia University

2013

Outline

Server-Side Languages

Server-side languages are implemented and executed in a webserver environment.

- Typical server-side languages are Perl, PHP, Python, Asp.
- A typical client-side language is Javascript.

Java can be used either server-side (as Java Servlets) or client-side (using applets and web start).

Static HTML pages

Many HTML documents provide **static** content, which is

- stored on a webserver,
- retrieved via the HTTP protocol,
- displayed to a client via a browser.

Dynamic Content

Some HTML documents provide **dynamic** content. This content

- is generated by a computer program;
- can retrieve information from a database;
- can respond to a specific user request (e.g. a webform);
- is converted into an HTML page;
- which is retrieved via HTTP by the user's browser.

Advantages of Dynamic Content

Dynamic content

- is more flexible than static content (e.g. on-line newspapers);
- can respond to specific user requests (e.g. e-commerce);
- can collect user information (e.g. on-line surveys, guestbooks);
- can provide an interface to a database (e.g. search engines);
- can facilitate basic user interaction (e.g. on-line shopping).

Server-side or Client-side?

Which of these are better implemented client-side or server-side?

- Applications with many graphics (e.g. certain computer games) -

Server-side or Client-side?

Which of these are better implemented client-side or server-side?

- Applications with many graphics (e.g. certain computer games) - Client-side (because graphics are slow over the web)

Server-side or Client-side?

Which of these are better implemented client-side or server-side?

- Applications with many graphics (e.g. certain computer games) - Client-side (because graphics are slow over the web)
- Database interfaces -

Server-side or Client-side?

Which of these are better implemented client-side or server-side?

- Applications with many graphics (e.g. certain computer games) - Client-side (because graphics are slow over the web)
- Database interfaces - Server-side (because the user usually only needs a few records from a database; transmitting the whole database would be slow)

Server-side or Client-side?

Which of these are better implemented client-side or server-side?

- Applications with many graphics (e.g. certain computer games) - Client-side (because graphics are slow over the web)
- Database interfaces - Server-side (because the user usually only needs a few records from a database; transmitting the whole database would be slow)
- On-line maps -

Server-side or Client-side?

Which of these are better implemented client-side or server-side?

- Applications with many graphics (e.g. certain computer games) - Client-side (because graphics are slow over the web)
- Database interfaces - Server-side (because the user usually only needs a few records from a database; transmitting the whole database would be slow)
- On-line maps - Server-side (because a user who searches for an address does not need to download a whole atlas).

But many applications (such as on-line banking applications) have both client-side components (using Javascript or Java for better usability and graphics of the client browser window) and server-side components for storing the data in a database on the server.

Challenges for Server-Side Applications

The biggest challenge is **Security**! The user can never be trusted because for each click the user makes on a browser a new connection is established.

A second challenge is the limitation of the HTTP protocol. User activities are limited by what is possible via HTML and common browsers.

PHP code is embedded into websites

```
<html>
<head><title>Hello World</title></head> <body>
<?php
echo "What is your name?";
echo "Hello, {$_REQUEST['name']}! How are you?";
?>
</body></html>
```

User input comes from a web form

```
<form action='example.php' method='get'>
<input type='textbox' name='name'>
<input type='submit' value='submit'>
</form>;
```

or via the Query String:

```
http://hostname/~username/php/example.php?name=Snoopy
```

and is available using special variables:

```
$_REQUEST['name']
```

Primitive datatypes

Some of PHP's data types are:

- boolean ($foo = true; $foo = false;)
- integer ($a = 1; $b = 15; $c = 300000000)
- float ($a = 3.14159, $b = 1.2e3;)
- string ($a = 'Hello World'; $b = "Hello World\n";)
- array ($list = array("key" => "value", 1 => 2));

Similar operators as in other languages:

- arithmetic: + − * / ++ −− %
- comparison: == != < >
- logical: and or xor !
- string concatenation: . .=
- array: + == !=

The usual control structures:

```
if (... ) { ... } else if ( ... ) { ... } else { ... }

while (... ) { ... }

for ($i = 1; $i <= 10; $i++) { echo $i; }

foreach ($arr as $value) {echo $value }
```

PHP has lots and lots of predefined functions

For example, for arrays
$zoo = array("monkey", "tiger", "eagle");

- count($zoo);
- implode (" ",$zoo);
- array_push($zoo, $newanimal);
- array_pop($zoo);
- sort($zoo);
- rsort($zoo);

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)?
- 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 languages 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.

For Python: WSGI (Web Server Gateway Interface).

HTML pages

- HTML pages must be stored in a dedicated directory on the webserver. The name of the directory depends on server settings. A common name is "public_html".
- File permissions must be set to allow other users to view the HTML pages.
  - The user's home directory must be executable by others.
  - The public_html directory and its subdirectories must be readable by others.
  - Each HTML page must be readable by others.

Viewing HTML pages

The URL of a HTML page often consists of the server, a tilde, the username, and the path and filename of the html page starting below "public_html".

Example: A file

/home/username/public_html/hello.html

would be available at the URL

http://servername/∼username/hello.html

Script pages

- Script pages are usually stored in a dedicated directory under the public_html directory. The name of this directory depends on the language used and on the server settings. Common names are “php” or “cgi-bin”.
- File permissions must be set to allow script pages to be executed. Depending on the server settings, scripts are either executed
  - as a general www user ($\rightarrow$ set permissions to 755) or
  - as the owner of the script ($\rightarrow$ set permissions to 700).

Viewing script pages

The URL of a script page is formed in the same manner as for other HTML pages. In this case, the directory name (cgi-bin) is included.

Example: A file

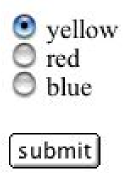
/home/username/public_html/cgi-bin/hello.pl

would be available at the URL

http://servername/∼username/cgi-bin/hello.pl

HTML content can be dynamically generated in response to a form which a user has filled in.

For example, a user could be asked to check one out of three checkboxes and then click a submit button:



After submitting the button the user could then be presented with:

Hello! Your choice was yellow

Implementing this example

Two HTML pages need to be created for this example:

- An HTML page which contains the webform (i.e., the radio buttons and the submit button).
- A dynamically generated HTML page, which uses the colour which the user had selected.

The second page is generated by a script. The script stores the parameters from the webform as variables, which can then be used within the HTML:

```
<p>Hello! Your choice was
<font color=$formparameter>$formparameter</font><p>
```