Introduction to PHP

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK
Outline

Server-Side Web Languages

PHP
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).
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)
- 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.
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.
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
<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

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

or via the Query String:

```
http://www.dcs.napier.ac.uk/~01234567/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:

```php
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);