PENGENALAN PHP
NASRUL@NURULFIKRI.COM

OVERVIEW

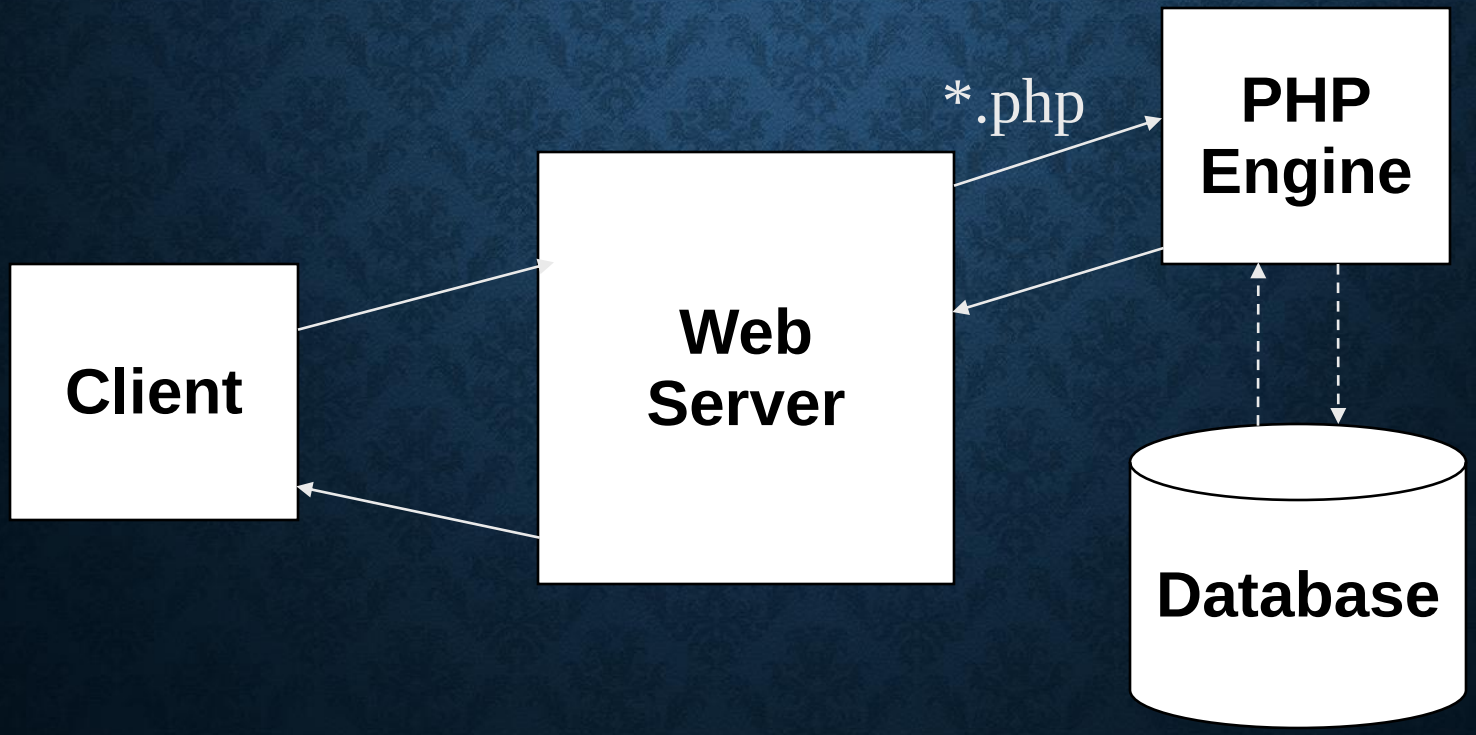
- PHP Scripting Language
  - Variables & Operators
  - Control Structures & Functions
  - Object-oriented PHP
- Form Processing
- File Operations
  - Filesystem
  - Reading, Writing
- Database Access with PHP
  - MySQL

BACKGROUND ON PHP

- PHP Hypertext Processor (PHP)
  - Developed by Rasmus Lerdorf in 1995 as a set of macro commands in PERL (later, C) and an interpreter.
  - Succeeds PHP/FI - "Personal Home Page/ Form Interpreter"
  - Started on Linux; now on multiple platforms
  - Gradually added more functionality – email, cookies, DB access

HOW PHP WORKS

- PHP is an interpreted language
  - Scripts are interpreted by PHP's Zend parsing engine
  - Output is HTML



WHY IS PHP SO POPULAR?

- Open-source & free
- Multi-platform
- (Relatively) easy to learn – if you're comfortable with C, JavaScript, or other scripting languages
- More efficient use of server resources, especially on Linux/UNIX
- *PHP5* – better object-oriented and XML language support

CODE WITH XAMP

- Open-source & free
- Multi-platform
- (Relatively) easy to learn – if you're comfortable with C, JavaScript, or other scripting languages
- More efficient use of server resources, especially on Linux/UNIX
- *PHP5* – better object-oriented and XML language support

BASIC SYNTAX

```
<html>
...
<?php
  //Some code
  echo '<p>Hello World</p>';
?>
...
</html>
```

VARIABLES

- Variables in PHP are represented by a $: *$var_name*
- Valid variable names
  - Starts with a letter or underscore
  - Can only contain letters, numbers, or underscores
  - Cannot be one of PHP's reserved words (e.g. "class")
- Variables can be assigned by *value* or *reference* (if the variable is named).
  - Value:: $myVar = 25;
  - Reference:: $myRef = &$myVar;
  - Invalid:: $myRef = &(7 * 52) //Expression is unnamed
- Variables do not have to be initialized, *but it's very good practice*.

WORKING WITH VARIABLES

- PHP supports the following:
  - Boolean: $isActive = TRUE;
  - Integer:: $zip = 15213;
  - Float:: $myBalance = 567.89;
  - String:: $isActive = 'TRUE';
  - Arrays:: $myInfo = array("name" => "Bob");
  - Objects:: $myBook = new Book( );
  - Constants:: define("FOO", "something");
- Like JavaScript, PHP is loosely-typed – interpreter handles variable type on-the-fly.

MORE ON STRINGS

- Single-quoted strings
  - Ex.:: $quote = 'I\'ll be back';
  - Use \' to specify a single quote character.
  - Does not expand newlines or variables.
- Double-quote strings
  - Does expand newlines & variables.
  - Use \ to escape characters.
  - *Example::* $q = "Arnold said, '$quote'\n";
  - *Result::* Arnold said, 'I'll be back' (followed by a linefeed.)
- Use . to concatenate strings
  - Example $q = 'Arnold said,'. $quote;

MORE ON ARRAYS

- Arrays in PHP are actually *ordered maps*
- Can be used for many data structures
  - Ex. Real arrays, lists, hashtables, stacks, queues.
- array (*key1 => value1, keyN => valueN*);
  - Key can be string or integer
  - Value can be any type.
- $arr = array (1 => “red”, 2=> “blue”);
- $result = $arr[1]; // prints ‘red’
- $arr[“a”] = “green”; // adds new element
- unset($arr[“a”]); // removes element
- unset($arr); // deletes array

PHP'S PREDEFINED VARIABLES

- PHP includes a series of predefined arrays available in every scope called *superglobals*.
  - $_SERVER
  - $_GET and $_POST
  - $_COOKIE
  - $_REQUEST
  - $_FILES
  - $_ENV
  - $_SESSION

BASIC OPERATORS

- PHP has most of the operators one would find in other scripting languages.
- These include
  - Arithmetic:: + - * / %
  - Assignment:: +=, etc
  - Increment/decrement:: ++ --
  - Comparison:: <, >, >=, <=, ==, != , < >, ===
  - Logical:: &&, ||
  - String:: ., .=
  - Type:: instanceof *someClass*;

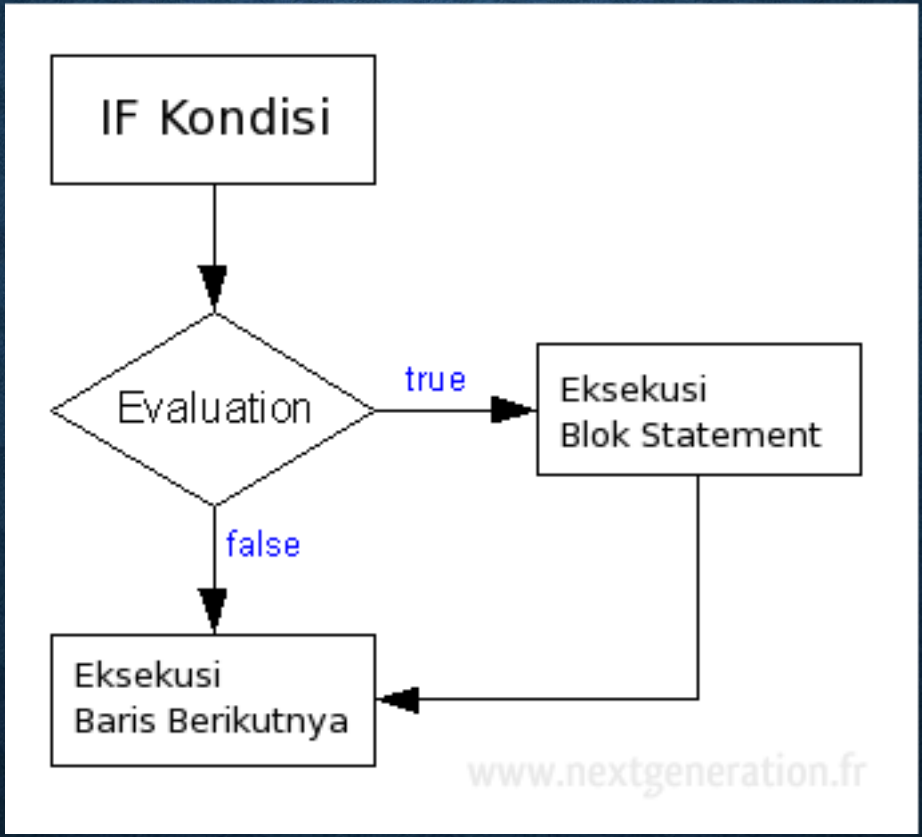
CONTROL STRUCTURES IN PHP

- if (*expression*) {*block*} elseif {*block*} else {*block*}
- while (*expression*) {*block*} OR
- do {*block*} while (*expression*)
- for (expression) {block}
- foreach var (list) {block}
- switch (*variable*) //variable can be int or string

```
case 0:
    // do something;
    break;
case 1:
    // do something else;
    break;
```

- break;
- continue;
- return;

KONDISI IF


IF Kondisi
Evaluation
true
Eksekusi
Blok Statement
false
Eksekusi
Baris Berikutnya
www.nextgeneration.fr

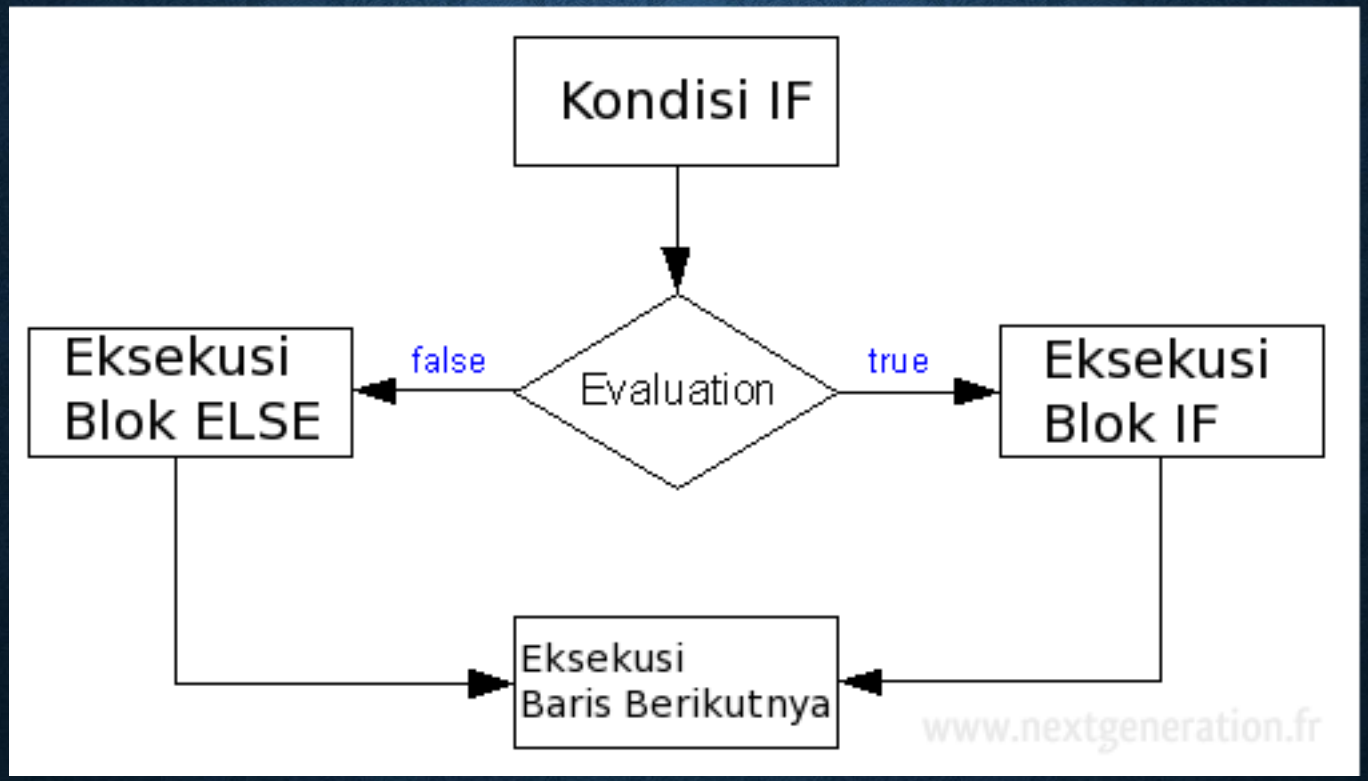
KONDISI IF

```
<?php
   if(condition)
   {
      // Blok Statement
   }
?>
```

Contoh :

```
<?php
if (!empty($nama))
{
echo 'Nama ' . $nama;
}
?>
```

KONDISI ELSEIF



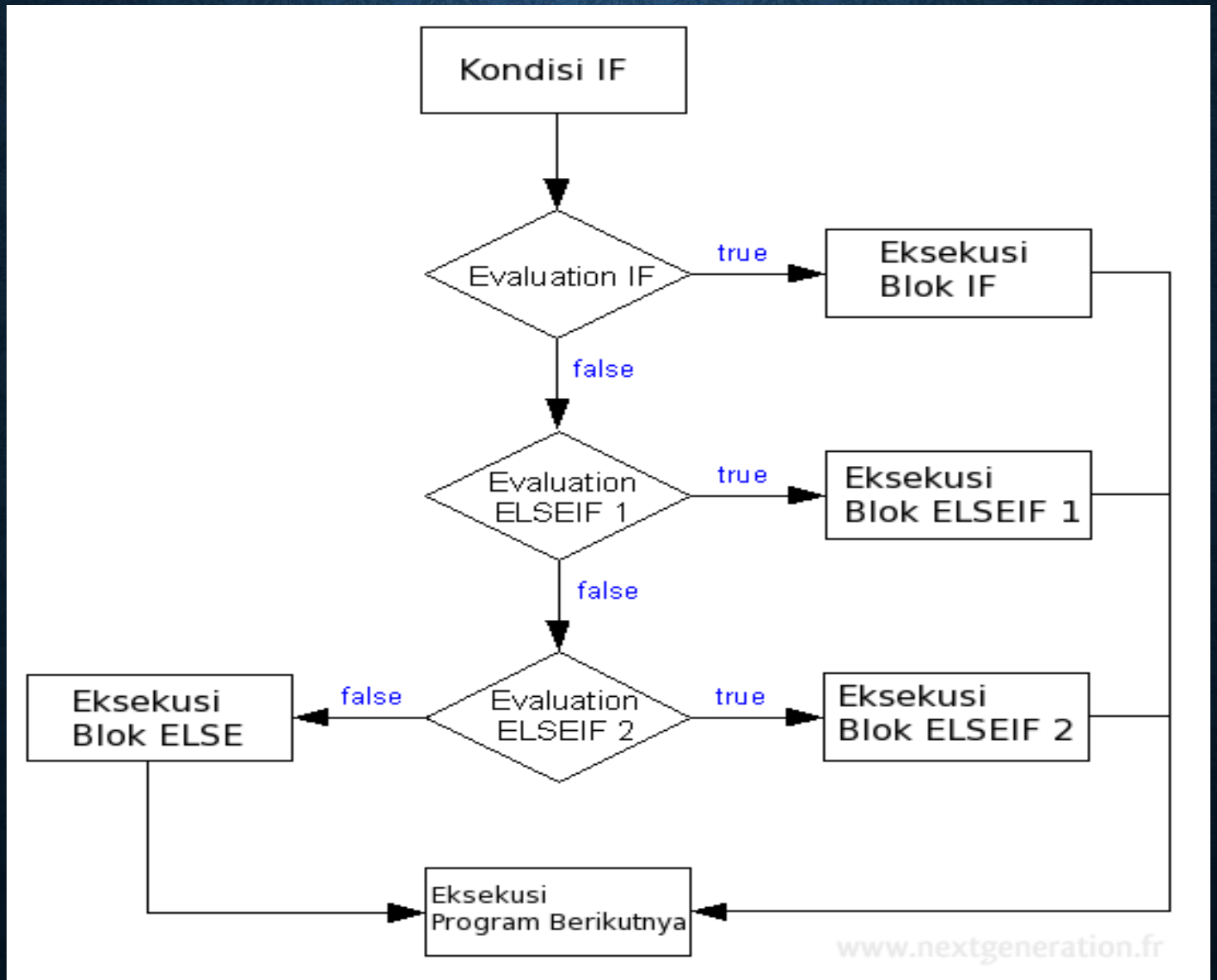
KONDISI ELSEIF

```
<?php
   if(condition)
   {
      // Blok Kondisi IF
   } else {
      // Blok Kondisi ELSE
   }
?>
```

KONDISI ELSEIF

```
<?php
   if(condition)
   {
      // Blok Kondisi IF
   } else {
      // Blok Kondisi ELSE
   }
?>
```

MULTI KONDISI ELSEIF



MULTI KONDISI ELSEIF

```
<?php
 if(condition 1)
 {
    // Blok Kondisi IF
 } elseif (condition 2) {
    // Blok Kondisi ELSEIF 1
 } elseif (condition 3) {
    // Blok Kondisi ELSEIF 2
 } else {
    // Blok Kondisi ELSE
 }
?>
```

SWITCH CONDITONAL

```
switch (  $var )
{
    case  'value-1' :
    statement/blok statement ;
    break;
case  'value-2' :
        statement/blok statement ;
    break;
--
default :
    statement/blok statement;
}
```

SWITCH CONDITONAL

```
switch (  $grade )
{
    case  'E' :
    $predikat = 'SANGAT KURANG';
    break;
case  'D' :
        $predikat = 'KURANG';
    break;
--
default :
    $predikat = 'TIDAK ADA';
}
```

LOOPING: FOR

```
for  (expr1 ; expr2 ; expr3)
{
   blok statement
}


// Contoh : cetak bilangan 1 s/d 30

for ( $x = 1 ; $x <= 30 ; $x++)
{
   echo '<br>Bilangan : '.$x;
}
   echo '<br>Selesai';
```

LOOPING: WHILE

```
while  (condition)
{
  blok statement // akan di eksekusi selama kondisi benar
}

// Contoh : cetak bilangan 1 s/d 20

$x = 0;
while( $x <= 20)
{
  $x++ ; // sama dengan $x = $x + 1
  echo '<br>Bilangan : '.$x;
}
  echo '<br>Selesai';
```

BREAK & CONTINUE

- Keyword break : digunakan untuk keluar dari looping
- Keyword continue : digunakan untuk melanjutkan looping berikutnya dengan mengabaikan baris dibawahnya

```
// Contoh : cetak bilangan 1 s/d 12, 14 s/d 17

$x = 0;
while ( $x <= 20)
{
  $x++;
  if ( $x == 13 ) continue;
  echo '<br>Bilangan : '.$x;
  if ( $x == 17 ) break;
}
  echo '<br>Selesai';
```

PHP FUNCTIONS

- Functions follow the same naming rules as variables.
- Any type can be returned.

```
function doSomething($arg1, $arg2) {
  $result = $arg1 + $arg2;
  return $result;
}
```

PHP INTERNAL FUNCTIONS

- PHP includes over 700 built-in functions, including those for…
  - Web-server/platform specific
  - Database & Network
  - XML
  - Images & PDF
  - Filesystem
  - Strings, Math, Mail
  - Dates
- Some functions require certain PHP libraries to be included at compilation.
- PHP Manual includes 187-part function reference.

SOME USEFUL STRING FUNCTIONS

- addslashes(*$string*)
  - escapes strings with quotes
- stripslashes(*$string*)
  - removes slashes added by addslashes( );
- explode($*delimiter, $string*)
  - Splits a string by a delimiter string.
- implode(*$glue, $string*)
  - Joins array elements to with a string
- htmlspecialchars(*$string*)
  - transforms special characters to HTML entities
- str_replace(*$search, $replace, $subject*)
  - Replaces all occurrences of a search string with a replacement string

MORE USEFUL STRING FUNCTIONS

- strip_tags(*$string*)
  - removes HTML tags
- strlen(*$string*)
  - gets the length of a string
- strstr(*$haystack, $needle*)
  - case-sensitive substring search
- stristr(*$haystack $needle*)
  - case-insensitive substring search
- strtok(*$string, $delimiter*)
  - tokenizes a string
- substr(*$string, $start, $length*)
  - gets part of string
- trim(*$string*)
  - removes whitespace from beginning and end of string

SOME USEFUL MATH FUNCTIONS

- abs
  - absolute value
- ceil
  - rounds a float up. Ex. ceil(4.2) => 5
- floor
  - Rounds a float down. Ex. floor(4.2) => 4
- decbin; dechex
  - decimal to binary; decimal to hexadecimal
- max, min
- rand
  - random number generator
- Sqrt
  - square root
- Log functions (log)
- Trigonometric functions (cos, sin, tan)

SOME USEFUL DATE FUNCTIONS

- date(*$format, $[timestamp]*)
  - Creates a new date in the specified format using the given timestamp (or today's date, if no timestamp)
  - PHP manual lists the characters you use to format a date (hour, minute, day, month, year, AM/PM, etc)
  - Examples:
    - date("m.d.y") => '06.20.07'
    - date("F j, Y, g:i A") => 'June 20, 2007, 5:30 PM'
    - date("l \\t\h\e jS") => 'Wednesday the 20th'
- time( )
  - Gives you the current time as a UNIX timestamp
- strtotime(*$timeString*)
  - Parses a string into a UNIX timestamp

SENDING MAIL IN PHP

- mail (*$to, $subject, $message, [$other_headers]*);
- Returns TRUE if successful, FALSE otherwise.
- Important rules
  - PHP must have access sendmail or an equivalent program; otherwise, no additional libraries required.
  - No newline characters in subject.
  - Each line of the message should end with a \n, and should be no longer than 70 characters.
- Example

```
$to = 'abc@pitt.edu, def@pitt.edu, ghi@pitt.edu';
$message = 'Hello all,\n\nI just wanted to say…";
$message = wordwrap($message, 70);
// Send the mail
mail($to, 'Important Announcement', $message);
```

SENDING HTML MAIL IN PHP

- HTML-formatted can be also sent using mail( ), but additional headers must be specified.
- Example

```
$to = ‘abc@pitt.edu, def@pitt.edu, ghi@pitt.edu’;
$subject = ‘Important Announcement’;
$headers = ‘MIME-version: 1.0’.“\r\n”;
$headers .= ‘Content-type: text/html’.“\r\n”;
$headers .= ‘From: xyz@pitt.edu’.“\r\n”;

$message = ‘<p>Hello all,<p>’;
$message .= ‘<p>I just wanted to say…</p>’;

// Send the mail
mail($to, $subject, $message, $headers);
```

PHP ERROR HANDLING

- Paradox (understandably) suppresses all PHP errors.
- Use the following code to print errors (for debugging purposes only!)

```
<?php
  //error handler function
  function customError($errno, $errstr) {
   echo "<b>Error:</b> [$errno] $errstr";
  }
  //set error handler
  set_error_handler("customError");
  //trigger error
  echo($test); ?>
```

PHP & HTML FORM PROCESSING

SERVER-SIDE FORM PROCESSING

- One of PHP's original core functions was to process HTML form input
  - Storing input in a file or database
  - Email input to customers, other users
  - Validating input (security)
- *Every element* of a form is available to PHP.
- PHP5 and later store form information in the $_GET, $_POST, and $_REQUEST superglobal associative arrays.

AN EXAMPLE USING GET - HTML

```
<form name="f1" method="get" action="thanks.php">
  <input type="hidden" name="fType" value="GET" />
  <input type="text" name="fullName" /><br />
  <select name="program" size="1">
    <option>Undergraduate</option>
    <option>Masters</option>
    <option>Doctoral</option>
  </select>
  <input type="submit" value="Submit" />
</form>
```

AN EXAMPLE USING GET - PHP

- URL: http://*some_domain*/thanks.php?fullName=John&program=Masters

```
<?php
    $formType = $_GET[‘fType’];
    $fullName = $_GET[‘fullName’];
    $program = $_GET[‘program’];

    echo (‘<p>’);
    echo (‘Hello,’.$fullName.“<br>\n”);
    echo (‘You are in the ’.$program.‘ program.“<br>\n”);
    echo (‘Form method was ‘.$formType.“</p>\n”);
?>
```

AN EXAMPLE USING POST - HTML

```
<form name="f2" method="post" action="thanks.php">
  <input type="hidden" name="toEmail"
  value="xyz@pitt.edu" />
  <input type="text" name="firstName" /><br />
  <input type="text" name="lastName" /><br />
  <textarea name="comment"></textarea>
  <input type="submit" value="Submit" />
</form>
```

AN EXAMPLE USING POST - PHP

- URL: http://*some_domain*/thanks.php

```
<?php
   $to = $_POST['toEmail'];
   $firstName = $_POST['firstName'];
   $lastName = $_POST['lastName'];
   $comment = addslashes($_POST['comment']);

   echo ('<p>');
   echo ('Thank you,'.$firstName.' for your feedback\.');
   echo ( "</p>\n");
?>
```

VALIDATING FORMS AT THE SERVER

- Though JavaScript (i.e. client-side) is best for validation, your PHP code should still provide an extra layer of protection.
  - Verifying unique attributes (usernames, etc.)
  - Preventing code injections
  - Especially important if submitted information will be used to perform system-level operations!
- Checking & sanitizing input
  - Write your own expressions.
  - Use PHP5's built-in filters.

PHP FILTERS

- PHP filters can automatically validate data types or sanitize strings for you.
- filter_input(*$input_type, $var, [$filter_type]*);
  - *$input_type* can be INPUT_POST or INPUT_GET
  - *$var* is the variable to check.
  - *$filter* is an optional filter, specified by text or ID.
    - FILTER_SANITIZE_SPECIAL_CHARS
    - FILTER_VALIDATE_INT
    - FILTER_VALIDATE_EMAIL
  - Default filter is FILTER_SANITIZE_STRING, which strips out HTML tags.

PHP & FILE HANDLING

INTRODUCTION

- PHP includes a series of functions that allow you to manipulate files on the server's filesystem.
- If PHP is run under a superuser account, these can be VERY DANGEROUS.
- 2 simple file functions
  - include(*$file*) – inserts the contents of an external file into your PHP file before execution.
  - require(*$file*) – same thing, but halts script execution if an error occurs.

PHP FILESYSTEM FUNCTIONS

- unlink(*$files*)
  - Removes files
- chmod(*$file*)
  - changes protections of file to list
- chown($file)
  - changes owner and group of a file
- mkdir(*$pathname*)
  - creates a directory with protections
- rmdir(*$pathname*)
  - deletes a directory if empty

MORE PHP FILESYSTEM FUNCTIONS

- copy(*$source, $dest*)
  - Copies the source file to the destination.
- file_exists(*$filename*)
  - Checks whether a file or directory exists.
- rename(*$oldname, $newname*)
  - Moves a file.
- stat(*$filename*)
  - Gives you an array of file information.
- tmpfile ( )
  - Creates a temporary file; deleted when script ends.

FILE ACCESS

- To open a file:
  - fopen (*$filename, $accessmode*)
  - Access mode flags include:
    - r :: read-only
    - w :: write-only, overwriting existing file
    - a :: write-only, appending to end of existing file
    - x :: write-only, file must be not exist
    - t :: text mode, translates \n to \r\n
    - b :: binary mode
    - + :: allows read and write
  - Windows: remember to escape the \'s in the filepath.

FILE ACCESS (CONTINUED)

- To write to a file:
  - fwrite (*$file, $string, [$length]*)
  - Writes the given string to the specified file.
  - If *$length* is specified, writing will stop after *$length* bytes.
  - Use is_writable(*$file*) function to make sure you can write to the file!
  - When you're finished, close the file with fclose(*$file*);

IMPORTANT FILE FUNCTIONS

- After opening a file, many functions exist to help you read its contents.
- feof(*$file*)
  - Checks if the end of the file (EOF) has been reached.
- fgets(*$file*)
  - Reads a single line (string) from the file.
- fgetc(*$file*)
  - Reads a single character from the file.

SIMPLE PHP FILE EXAMPLE

```
<?php
  // Open the file.  Exit the script if we cannot
  $file=fopen("abc.txt","r") or die("Cannot open file.");

  // Until we reach the end of the file, read each line and
  add an HTML break.
  while (!feof($file)) {
    echo fgets($file). "<br />";
  }
  //We're done, so close the file.
   fclose($file);
?>
```

DATABASE ACCESS WITH PHP

INTRODUCTION

- Any large-scale Web application built with PHP is also likely to have database connectivity.
  - Store & retrieve information
  - Provided customized content
- Initially, PHP only worked with open-source databases (e.g. MySQL), but now works many more.
- Database connectivity is provided through separate extensions that must be configured when you compile your PHP engine.
- Earlier versions of PHP had MySQL support enabled by default, but not PHP5.

A FEW WORDS ABOUT MYSQL

- MySQL is one of the most popular open-source database packages.
  - Highly scalable
  - Runs on more than 20 platforms
  - Good reputation for reliability, ease-of-use
  - Adopted by many organizations in government and industry
  - Community version is free

USING MYSQL AND PHP

- Creating a database connection
- Selecting a database
- Running an SQL query
- Processing the query results
- Closing the database connection

CREATING A DATABASE CONNECTION

- mysql_connect(*$host, $username, $passwd*)
  - Opens a connection to the specified database server (host) using the given credentials.
  - Connection is closed automatically at the end of the script.
  - $host is often "localhost" or the name of the machine on which MySQL is running.
  - Don't forget to append the port # - default is 3309

```
$conn = mysql_connect("localhost:3309","sa","pass")
    or die("Could not connect: ".mysql_error());
```

SELECTING A DATABASE

- mysql_select_db(*$dbname, [$connection]*)
  - Selects a MySQL database by name.
  - If you don't specify the connection variable ($conn on the previous slide), the most recently declared mysql_connect( ) is assumed.

```
$myDb = mysql_select_db("booksDB", $conn);
if ( !$myDb) {
 die('Cannot use db: '.mysql_error( ));
}
```

RUNNING AN SQL QUERY

- mysql_query(*$sql, [$connection]*)
  - Sends an SQL query to the active database.
  - For SELECT queries, returns a *resource* if successful.
  - For UPDATE, INSERT, DELETE, returns TRUE.
  - For all query types, returns FALSE on error.

```
$sql = "SELECT * FROM books";
$rs = mysql_query( $sql, $conn);
if ( !$rs) {
 die('Could not run query\. Sorry\.');
}
```

PROCESSING THE QUERY RESULTS

- mysql_fetch_array(*$resource*)
  - Returns a row of the results set as an array .
  - As written above, can be an associative array (column names) or numeric array (column indices).
  - To read all (or some) rows, use a loop.
- mysql_num_rows(*$resource*)
  - Returns the number of rows in the result set.

```
while ($dbrec = mysql_fetch_array($rs)) {
 echo ($dbrec['Title'] .' was written by ');
 echo ($dbrec['Author'] ."<br>");
}
```

CLOSING THE DATABASE CONNECTION

- mysql_free_result(*$resource*)
  - Frees memory used by result set's resource .
- mysql_close(*$connection*)
  - Closes the MySQL server connection.
- PHP automatically does both when script ends (assuming mysql_connect( ) was used.)

```
// Clean up after ourselves
mysql_free_result($rs);
mysql_close($conn);
```

PUTTING IT ALL TOGETHER…

```
<?php
// Open the database connection
$conn = mysql_connect("localhost:3309", "sa", "pass")
        or die("Could not connect: ".mysql_error());

// Choose the 'booksDB' database
$myDb = mysql_select_db("booksDB", $conn);
  if ( !$myDb) {
    die('Cannot use db: '.mysql_error( )); }

// Select everything in the 'books' table
$sql = "SELECT * FROM books";
$rs = mysql_query( $sql, $conn);
  if ( !$rs) {
    die('Could not run query\. Sorry\.'); }
```

PUTTING IT ALL TOGETHER…(CONT.)

```
// Loop through all rows; print out each record's title and
// author on separate lines.
while ($dbrec = mysql_fetch_array($rs)) {
    echo ($dbrec['Title'] .' was written by ');
    echo ($dbrec['Author'] ."<br>");
  }

// Clean up after ourselves
mysql_free_result($rs);
mysql_close($conn);
?>
```

REFERENSI

Jonathan P. Grady

Department of Information Science and Telecommunications

University of Pittsburgh

Jpg14@pitt.edu

http://www.sis.pitt.edu/~jgrady