Debugging

Chapter 12

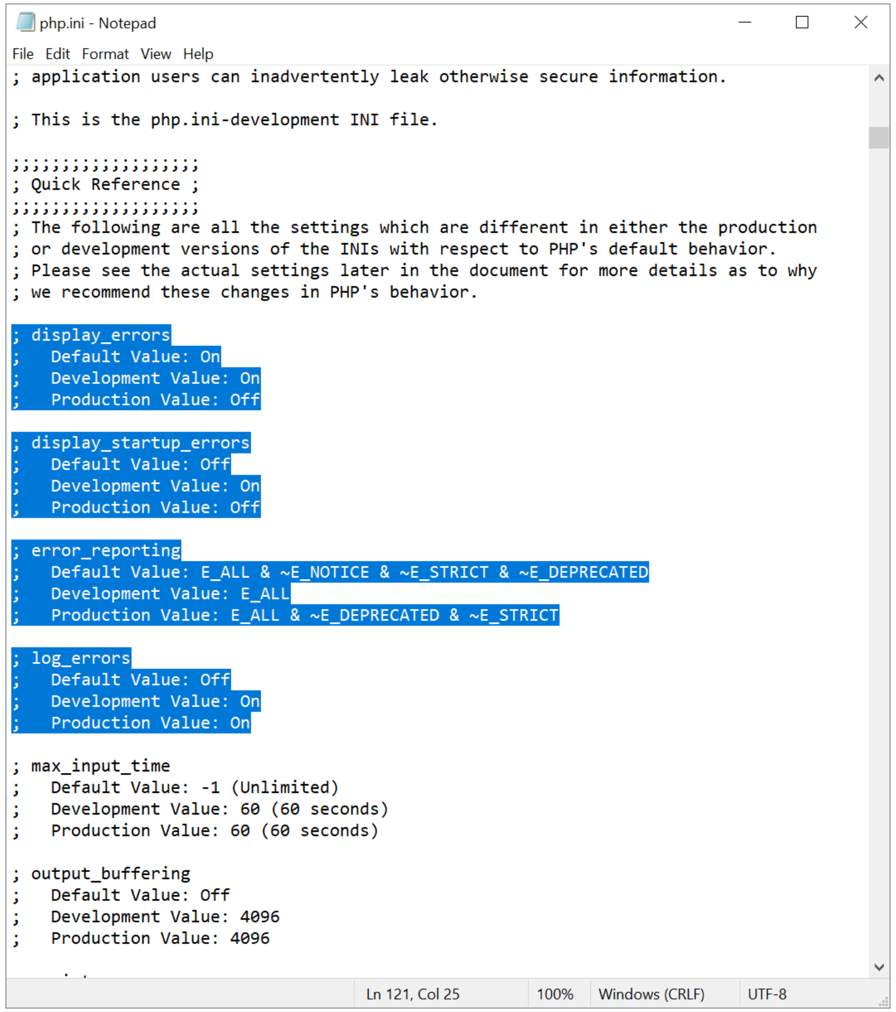
Debugging

- Programs rarely work correctly the first time you run them.
- We must use some tools and techniques to find and fix errors.

Where Errors Messages Appear?

- Displayed to the web browser. (along with the output of the program)
  - To make error messages display in the browser, set the display_errors configuration directive to On in php.ini
- Written to the web server error log.
  - To make sure errors end up in the web server error log, keep log_errors set to On in php.ini

- php.ini is located at where you installed PHP. for example: C:\wamp64\bin\php\php7.4.9



```
; application users can inadvertently leak otherwise secure information.

; This is the php.ini-development INI file.

;;;;;;;;;;;;;;;;;;;
; Quick Reference ;
;;;;;;;;;;;;;;;;;;;
; The following are all the settings which are different in either the production
; or development versions of the INIs with respect to PHP's default behavior.
; Please see the actual settings later in the document for more details as to why
; we recommend these changes in PHP's behavior.

; display_errors
;   Default Value: On
;   Development Value: On
;   Production Value: Off

; display_startup_errors
;   Default Value: Off
;   Development Value: On
;   Production Value: Off

; error_reporting
;   Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
;   Development Value: E_ALL
;   Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT

; log_errors
;   Default Value: Off
;   Development Value: On
;   Production Value: On

; max_input_time
;   Default Value: -1 (Unlimited)
;   Development Value: 60 (60 seconds)
;   Production Value: 60 (60 seconds)

; output_buffering
;   Default Value: Off
;   Development Value: 4096
;   Production Value: 4096
```

Types of Error Messages

- PHP engine generates **5 different error messages**:
  - **Parse error:** *syntax of your program*, such as leaving a **semicolon**.
    - *The engine stops running your program*
  - **Fatal error:** A severe error, such as *calling a function* that hasn't been defined.
    - *The engine stops running your program*
  - **Warning:** An advisory that something is fishy in your program, such as *wrong number of arguments* when you call a function
    - *the engine can keep going.*
  - **Notice:** A tip from the PHP engine for possible avoiding error/improving. Such as, *printing a variable without first initializing*.
    - *the engine can keep going.*
  - **Strict notices or deprecation warning:** An notice about your coding style, or **something you're doing will stop working** in a *future version of PHP.*

Error_reporting

- By **error_reporting** in **php.ini** you can control which kinds of errors the PHP engine reports.
- **Constants** to set the value of **error_reporting**:
  - E_ALL (for all errors)
  - E_PARSE (parse errors)
  - E_ERROR (fatal errors)
  - E_WARNING (warnings)
  - E_NOTICE (notices)
  - E_STRICT (strict notices, in versions of PHP before 7.0.0)

- The **default value** for **error_reporting** is **E_ALL & ~E_NOTICE & ~E_DEPRECATED**
  - Means report **All errors AND Not** notices **AND NOT** deprecation warnings

Fixing Parse Errors

- Sometimes error messages are not clear.
- For example:
  - **missing semi-colon**, or **starting with double-quote** and **ending with single-quote**
  - Reports “parse error”
- Solutions:
  - Use a PHP-aware editor
    - *syntax highlighting: different colors for different parts of the code*
    - *quote and bracket matching:*

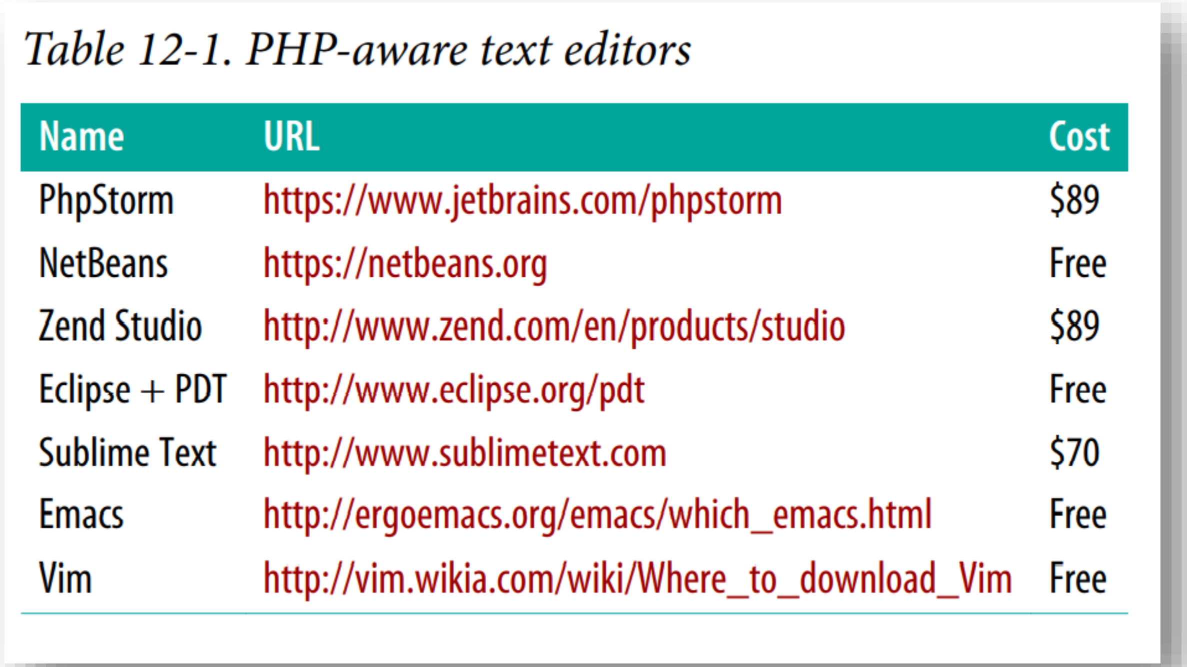
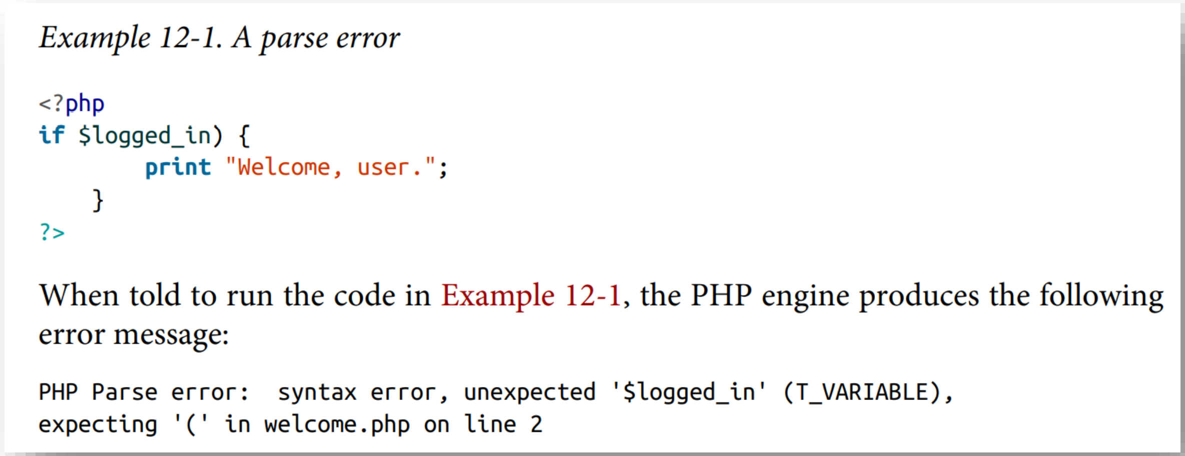


Table 12-1. PHP-aware text editors

Name	URL	Cost
PhpStorm	https://www.jetbrains.com/phpstorm	$89
NetBeans	https://netbeans.org	Free
Zend Studio	http://www.zend.com/en/products/studio	$89
Eclipse + PDT	http://www.eclipse.org/pdt	Free
Sublime Text	http://www.sublimetext.com	$70
Emacs	http://ergoemacs.org/emacs/which_emacs.html	Free
Vim	http://vim.wikia.com/wiki/Where_to_download_Vim	Free

Some Examples of Parse Errors

- T_VARIABLE: Tokens (variables are one type of token)



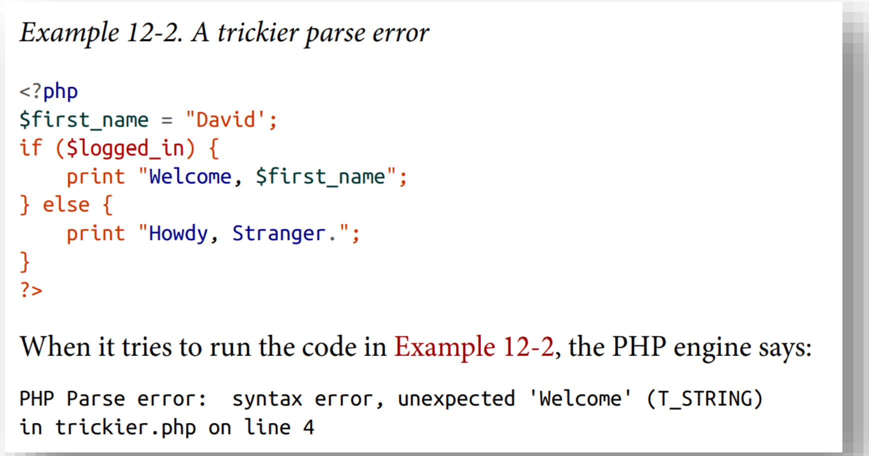
Example 12-1. A parse error

```
<?php
if $logged_in) {
        print "Welcome, user.";
    }
?>
```

When told to run the code in Example 12-1, the PHP engine produces the following error message:

```
PHP Parse error:  syntax error, unexpected '$logged_in' (T_VARIABLE),
expecting '(' in welcome.php on line 2
```

- Sometimes the **line number is not correct**!
- Check few **lines before and after**



Example 12-2. A trickier parse error

```
<?php
$first_name = "David';
if ($logged_in) {
    print "Welcome, $first_name";
} else {
    print "Howdy, Stranger.";
}
?>
```

When it tries to run the code in Example 12-2, the PHP engine says:

```
PHP Parse error:  syntax error, unexpected 'Welcome' (T_STRING)
in trickier.php on line 4
```

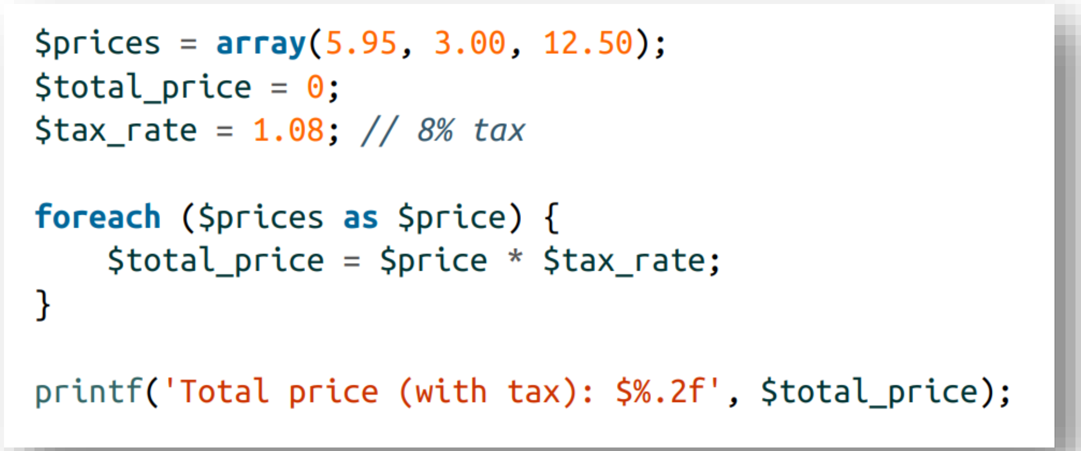
Catching Logical Errors

- A program may have correct syntax but not correct logic
  - For example, area = PI * r; instead of area = PI * r * r; for calculating the area of a circle

- Two solutions:
  - Adding Debug Output
  - Using a Debugger

Adding Debug Output

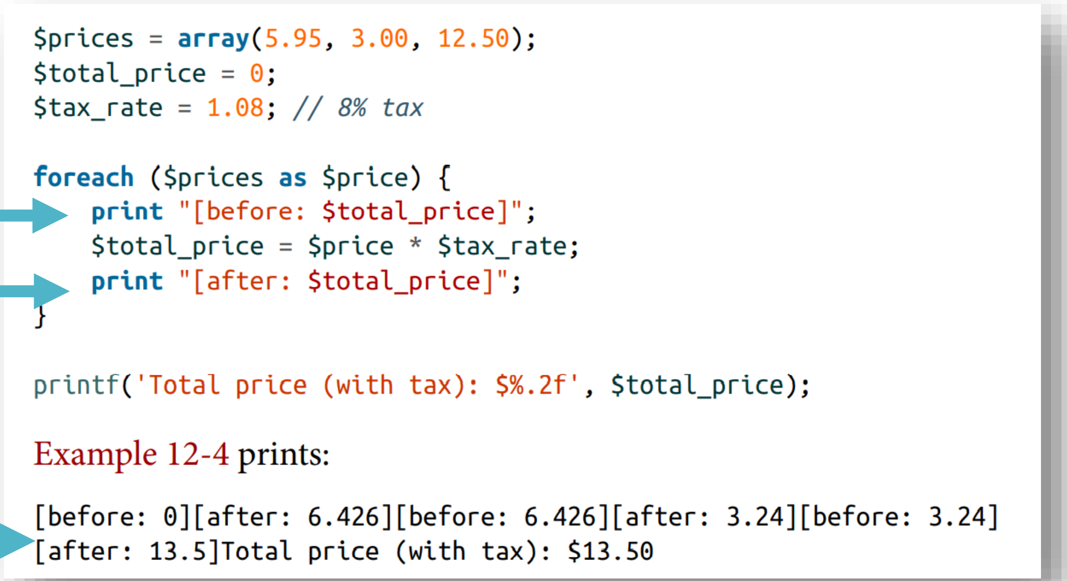
- If your program does not work, add some checkpoints that display the values of variables.



```
$prices = array(5.95, 3.00, 12.50);
$total_price = 0;
$tax_rate = 1.08; // 8% tax

foreach ($prices as $price) {
    $total_price = $price * $tax_rate;
}

printf('Total price (with tax): $%.2f', $total_price);
```



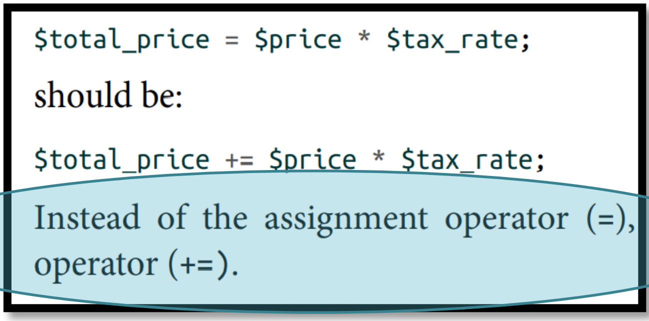
```
$prices = array(5.95, 3.00, 12.50);
$total_price = 0;
$tax_rate = 1.08; // 8% tax

foreach ($prices as $price) {
    print "[before: $total_price]";
    $total_price = $price * $tax_rate;
    print "[after: $total_price]";
}

printf('Total price (with tax): $%.2f', $total_price);
```

Example 12-4 prints:

```
[before: 0][after: 6.426][before: 6.426][after: 3.24][before: 3.24]
[after: 13.5]Total price (with tax): $13.50
```



```
$total_price = $price * $tax_rate;
```

should be:

```
$total_price += $price * $tax_rate;
```

Instead of the assignment operator (=), operator (+=).

Editing the Right File

- Sometimes, when you apply some changes, you may not apply to the file on the server!

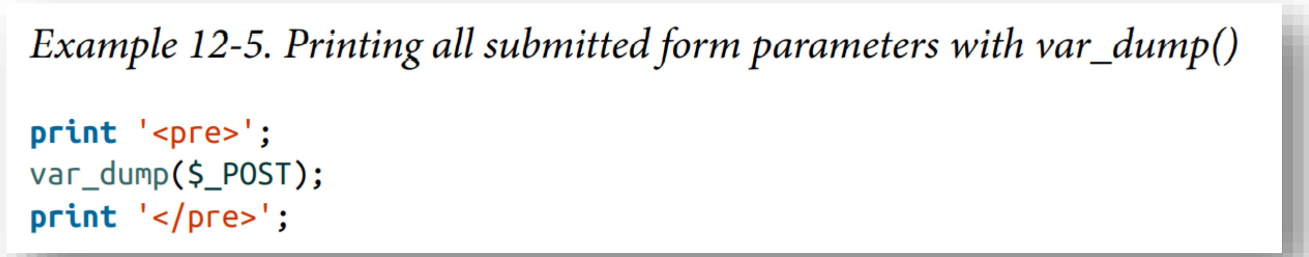
- Temporarily **add a line at the top of the program**.

```
die('This is: ' . __FILE__);
```

- **die()** is a function for terminating the program. It displays the message before exiting
- **__FILE__** referres to the current file is being executed.
- The output from the code above is: `This is: /usr/local/htdocs/catalog.php`

var_dump

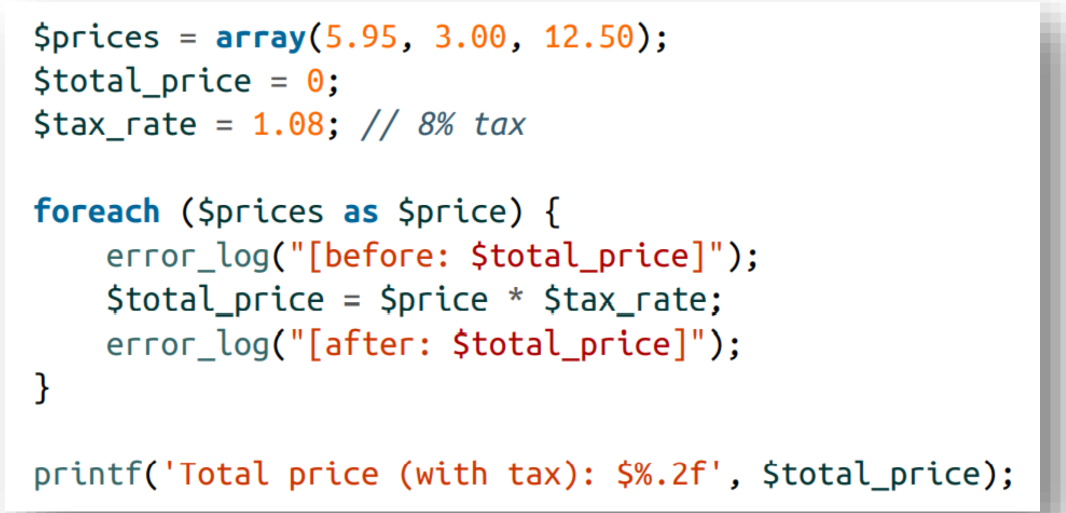
- Use var_dump to find debugging information about a varaible



Example 12-5. Printing all submitted form parameters with var_dump()

```
print '<pre>';
var_dump($_POST);
print '</pre>';
```

- Debugging messages can be confusing or disruptive when mixed with the page output.
- use the **error_log()** to send to the web server error log, instead of the web browser.
- **php_error.log** can be find in, <u>for example</u>, **C:\wamp64\logs**



```
$prices = array(5.95, 3.00, 12.50);
$total_price = 0;
$tax_rate = 1.08; // 8% tax

foreach ($prices as $price) {
    error_log("[before: $total_price]");
    $total_price = $price * $tax_rate;
    error_log("[after: $total_price]");
}

printf('Total price (with tax): $%.2f', $total_price);
```

Using a Debugger

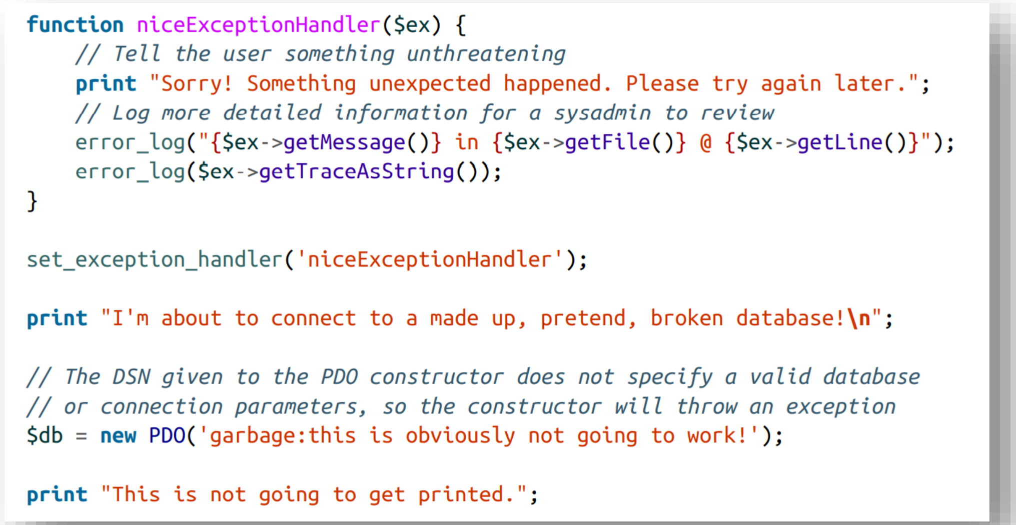
- The phpdbg debugger is part of PHP versions 5.6 and later
- To start a debugging session with phpdbg: `phpdbg -e broken.php`
- To set a breakpoint: `prompt> break 7`
- To run to the break point: `prompt> run`
- To pause the program each time the value a variable (e.g., $total_price) changes: `prompt> watch $total_price`
- To delete a break point: `prompt> break del 0`
- To continue after a pause on the break: `prompt> continue`

Handling Uncaught Exceptions

- if an exception is thrown but not caught: your PHP program stops running and the PHP engine prints out error information and a stack trace.
- IMPORTANT: always include try/catch blocks around any code that might throw an exception
- But you may not be able to catch all exceptions!
  - You can write an **Exception Handler** for catching any **missed** exceptions

  1. Write an **exception handler function**. It takes one **argument**: the exception.
  2. Use **set_exception_handler()** to tell the PHP engine about your function.

- This prevents the user from seeing confusing technical details that could potentially leak secure information



```php
function niceExceptionHandler($ex) {
    // Tell the user something unthreatening
    print "Sorry! Something unexpected happened. Please try again later.";
    // Log more detailed information for a sysadmin to review
    error_log("{$ex->getMessage()} in {$ex->getFile()} @ {$ex->getLine()}");
    error_log($ex->getTraceAsString());
}

set_exception_handler('niceExceptionHandler');

print "I'm about to connect to a made up, pretend, broken database!\n";

// The DSN given to the PDO constructor does not specify a valid database
// or connection parameters, so the constructor will throw an exception
$db = new PDO('garbage:this is obviously not going to work!');

print "This is not going to get printed.";
```

Exercises

1. This program has a syntax error in it:

```php
<?php
$name = 'Umberto';
function say_hello() {
    print 'Hello, ';
    print global $name;
}
say_hello();
?>
```

   Without running the program through the PHP engine, figure out what the parse error that gets printed when the engine tries to run the program looks like. What change must you make to the program to get it to run properly and print `Hello, Umberto`?

2. Modify the `validate_form()` function in your answer to Exercise 3 in Chapter 7 (see "Exercise 3" on page 345) so that it prints in the web server error log the names and values of all of the submitted form parameters.
3. Modify your answer to Exercise 4 in Chapter 8 (see "Exercise 4" on page 357) to use a custom database error-handling function that prints out different messages in the web browser and in the web server error log. The error-handling function should make the program exit after it prints the error messages.

Exercises

4. The following program is supposed to print out an alphabetical list of all the customers in the table from Exercise 4 in Chapter 8 (see "Exercise 4" on page 357). Find and fix the errors in it.

```
<?php
// Connect to the database
try {
    $db = new PDO('sqlite::/tmp/restaurant.db');
} catch ($e) {
    die("Can't connect: " . $e->getMessage());
}
// Set up exception error handling
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
// Set up fetch mode: rows as arrays
$db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
// Get the array of dish names from the database
$dish_names = array();
$res = $db->query('SELECT dish_id,dish_name FROM dishes');
foreach ($res->fetchAll() as $row) {
    $dish_names[ $row['dish_id']]] = $row['dish_name'];
}
$res = $db->query('SELECT ** FROM customers ORDER BY phone DESC');
$customers = $res->fetchAll();
if (count($customers) = 0) {
    print "No customers.";
} else {
    print '<table>';
    print '<tr><th>ID</th><th>Name</th><th>Phone</th>
    <th>Favorite Dish</th></tr>';
    foreach ($customers as $customer) {
        printf("<tr><td>%d</td><td>%s</td><td>%f</td><td>%s</td></tr>\n",
               $customer['customer_id'],
               htmlentities($customer['customer_name']),
               $customer['phone'],
               $customer['favorite_dish_id']);
    }
    print '</table>';
?>
```