Session 6

How to test and debug a PHP application

Objectives

- An introduction to testing and debugging
- How to debug with xDebug and NetBeans

An introduction to testing and debugging

Typical test phases for a PHP application

- In the first phases, as you test the user interface.
- In the second phase, you should test the application with valid data.
- In the third phase, you go all to make the application fail by testing every combination of invalid data and user action that you can think of.

The three types of errors that can occur

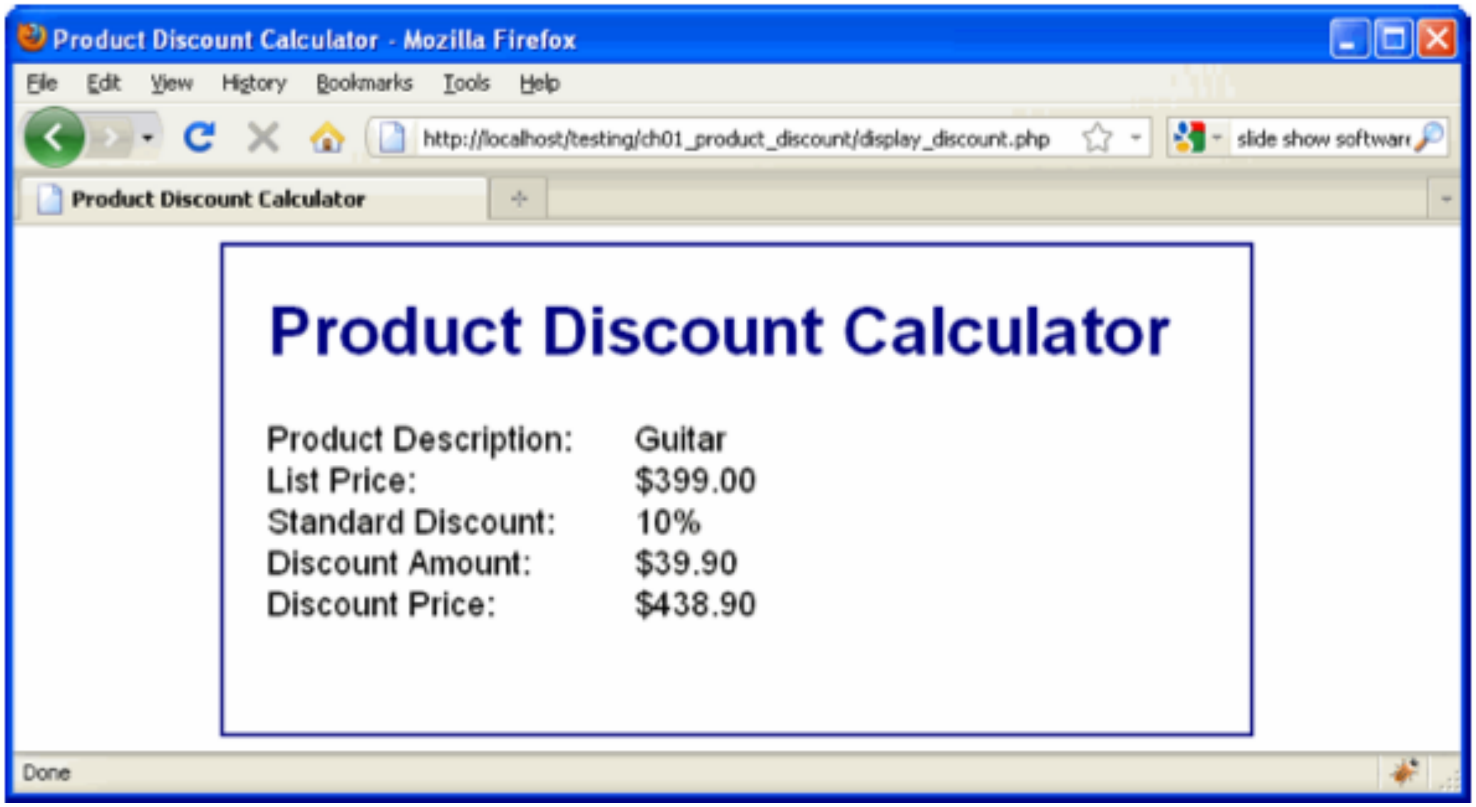
- The goal of testing: To find all errors before the application is put into production.
- The goal of debugging: to fix all errors before the application is put into production.
- Three test phases
  - Check the user interface to make sure that it works correctly
  - Test the application with valid input data to make sure the results are correct
  - Test the application with invalid data or unexpected user actions.

The three types of errors that can occur (cont.)

- The Three type of errors that can occur
  - Syntax errors
  - Runtime error
  - Logic errors

The three types of errors that can occur (cont.)

- The Product Discount application with a logic error


Product Discount Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/testing/ch01_product_discount/display_discount.php
slide show software
Product Discount Calculator
Done

Product Discount Calculator

Product Description:	Guitar
List Price:	$399.00
Standard Discount:	10%
Discount Amount:	$39.90
Discount Price:	$438.90

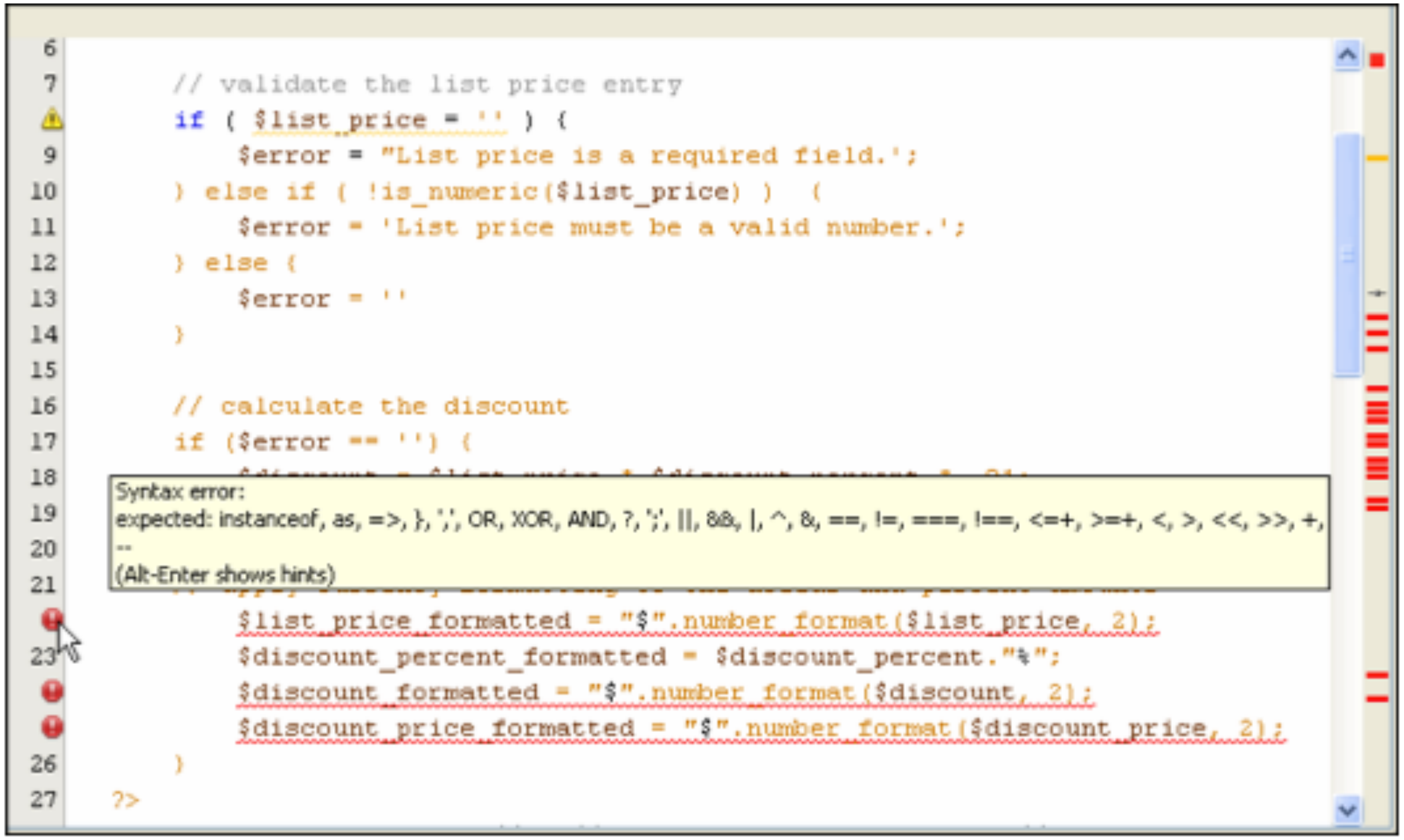
Common PHP errors

- PHP code that contains errors

```
// validate the list price entry
if ( $list_price = '' ) {
    $error = "List price is a required field.';
} else if ( !is_numeric($list_price) )  {
    $error = 'List price must be a valid number.';
} else {
    $error = ''
}
```

Common PHP errors (cont.)

- The PHO code that contains errors in NetBeans



```
    // validate the list price entry
    if ( $list_price = '' ) {
        $error = "List price is a required field.';
    } else if ( !is_numeric($list_price) )  {
        $error = 'List price must be a valid number.';
    } else {
        $error = ''
    }

    // calculate the discount
    if ($error == '') {
```

```
Syntax error:
expected: instanceof, as, =>, }, ',', OR, XOR, AND, ?, ';', ||, &&, |, ^, &, ==, !=, ===, !==, <=+, >=+, <, >, <<, >>, +,
--
(Alt-Enter shows hints)
```

```
        $list_price_formatted = "$".number_format($list_price, 2);
        $discount_percent_formatted = $discount_percent."%";
        $discount_formatted = "$".number_format($discount, 2);
        $discount_price_formatted = "$".number_format($discount_price, 2);
    }
?>
```

Common PHP errors (cont.)

- Common syntax errors
  - Misspelling keywords
  - Forgetting an opening or closing parenthesis, bracket, brace, or comment character.
  - Forgetting to end a PHP statement with a semicolon
  - Forgetting an opening or closing quotation mark
  - Not using the same opening and closing quotation mark.

Common PHP errors (cont.)

- Problems with variable names
  - Misspelling or incorrectly capitalizing a variable name
  - Using a keyword as a variable name
- Problem with values
  - Not checking that a value is the right data type before processing it.
  - Using one equal sign instead of two when testing for equality.

An easy way to trace the execution of your PHP code

- An easy way to trace the execution of a PHP application is to insert echo statements at key points in the code.
- The echo statement can display the values of variables or display messages that indicate what portion of the code is being executed.
- The incorrect value displayed, there is a good chance that you have a logic error between the current echo statement and the previous one.

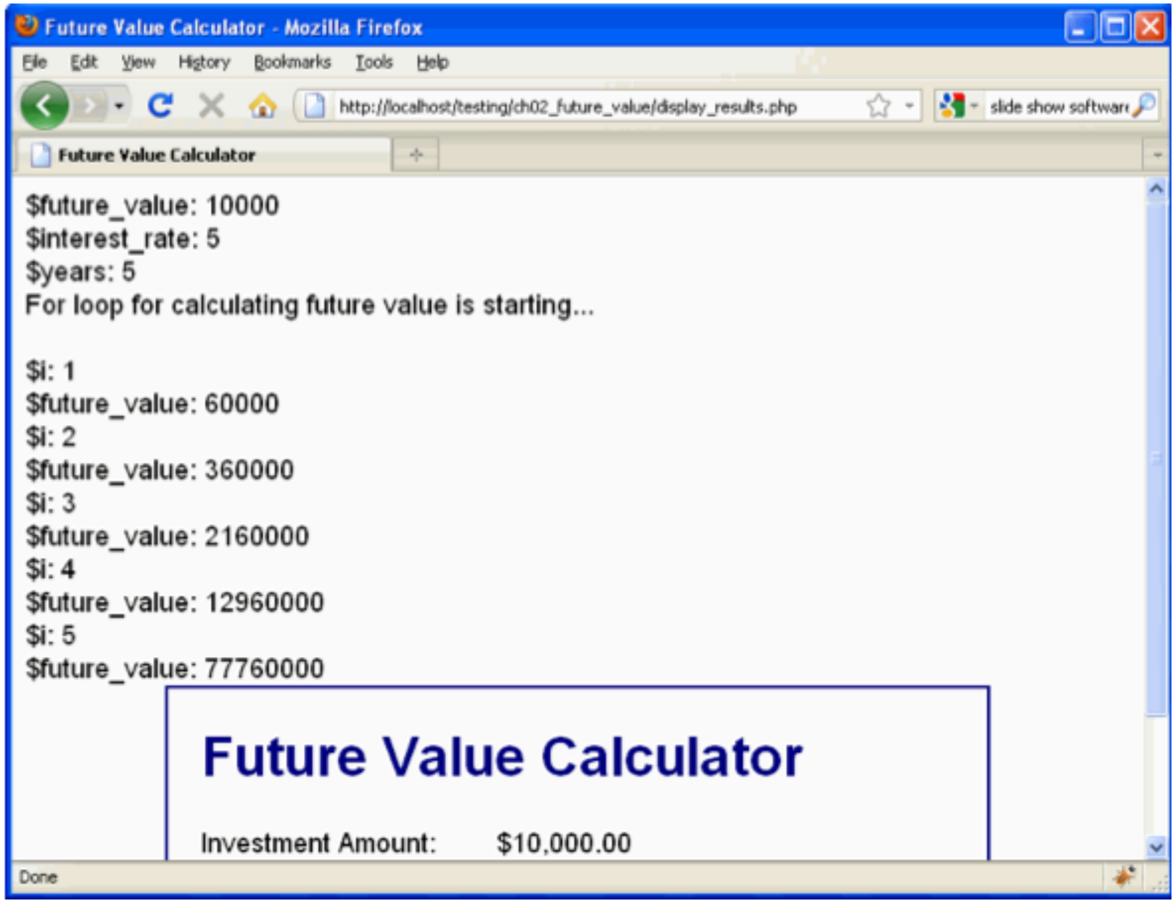
An easy way to trace the execution of your PHP code (cont.)

- PHP with echo statements that trace the execution of the code

```
// calculate the future value
$future_value = $investment;
echo '$future_value: ' . $future_value . '<br />';
echo '$interest_rate: ' . $interest_rate . '<br />';
echo '$years: ' . $years . '<br />';
echo 'For loop for calculating future value is starting...<br /><br />';
for ($i = 1; $i <= $years; $i++) {
    $future_value = ($future_value + ($future_value * $interest_rate));
    echo '$i: ' . $i . '<br />';
    echo '$future_value: ' . $future_value . '<br />';
}
```

An easy way to trace the execution of your PHP code (cont.)

- The data displayed in a browser


Future Value Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/testing/ch02_future_value/display_results.php
slide show software
Future Value Calculator
Done

```
$future_value: 10000
$interest_rate: 5
$years: 5
For loop for calculating future value is starting...

$i: 1
$future_value: 60000
$i: 2
$future_value: 360000
$i: 3
$future_value: 2160000
$i: 4
$future_value: 12960000
$i: 5
$future_value: 77760000
```

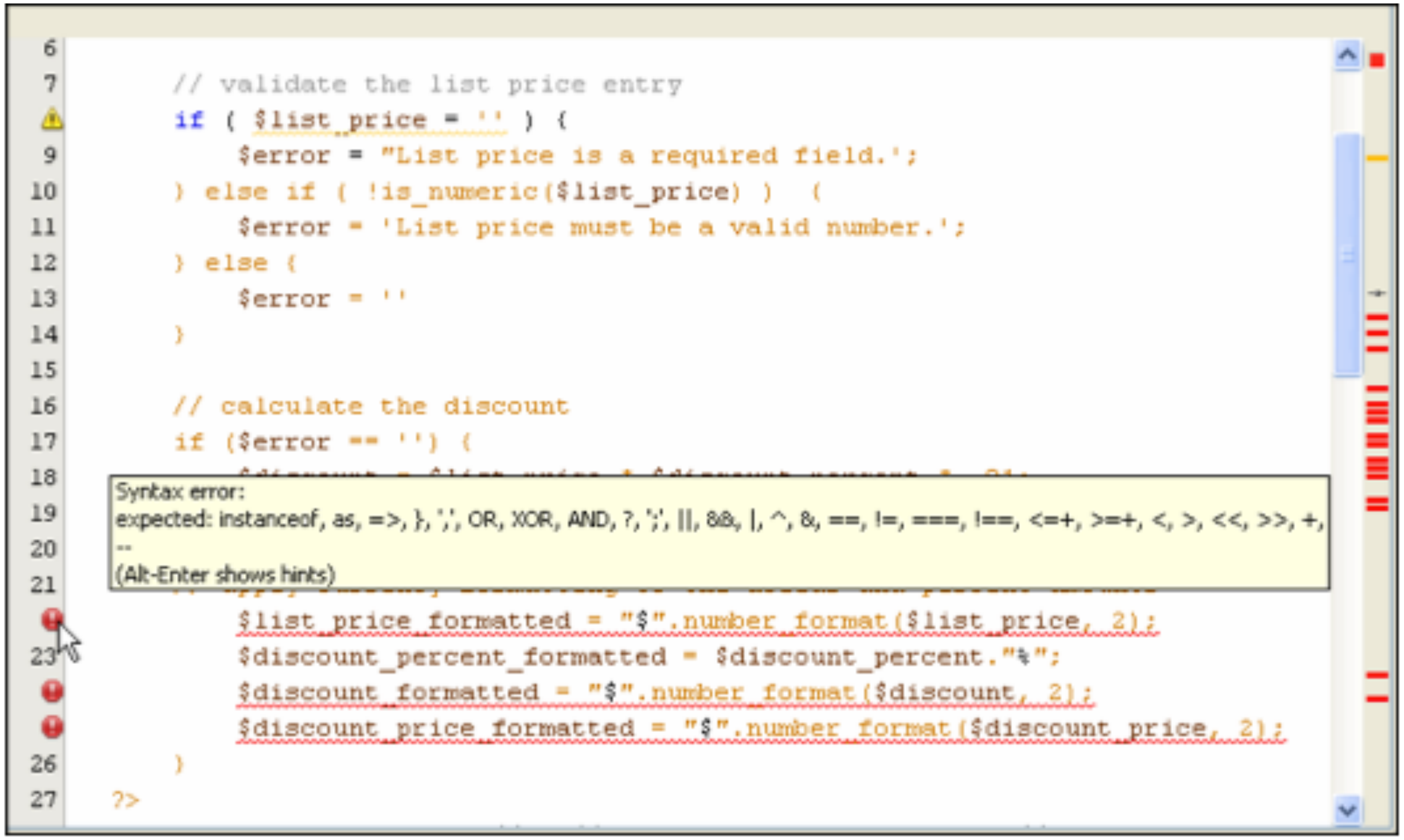
Future Value Calculator

Investment Amount: $10,000.00

How to debug with xDebug and NetBeans

How to set and remove breakpoints

- A code editor window with a breakpoint



```

    // validate the list price entry
    if ( $list_price = '' ) {
        $error = "List price is a required field.';
    } else if ( !is_numeric($list_price) )  {
        $error = 'List price must be a valid number.';
    } else {
        $error = ''
    }

    // calculate the discount
    if ($error == '') {
        $discount = $list_price * $discount_percent * .01;
[illegible]
[illegible]
        // apply currency formatting to the dollar and percent amounts
        $list_price_formatted = "$".number_format($list_price, 2);
        $discount_percent_formatted = $discount_percent."%";
        $discount_formatted = "$".number_format($discount, 2);
        $discount_price_formatted = "$".number_format($discount_price, 2);
    }
?>
```

```
Syntax error:
expected: instanceof, as, =>, }, ',', OR, XOR, AND, ?, ';', ||, &&, |, ^, &, ==, !=, ===, !==, <=+, >=+, <, >, <<, >>, +,
...
(Alt-Enter shows hints)
```

How to set and remove breakpoints (cont.)

- A breakpoint is indicated by a small red square.
- Set a breakpoint.
- Remove a breakpoint
- Use the Debug Main project button on the toolbar to begin debugging.
- When you begin debugging, NetBeans may display a dialog box that asks you want to debug.

How to step through code

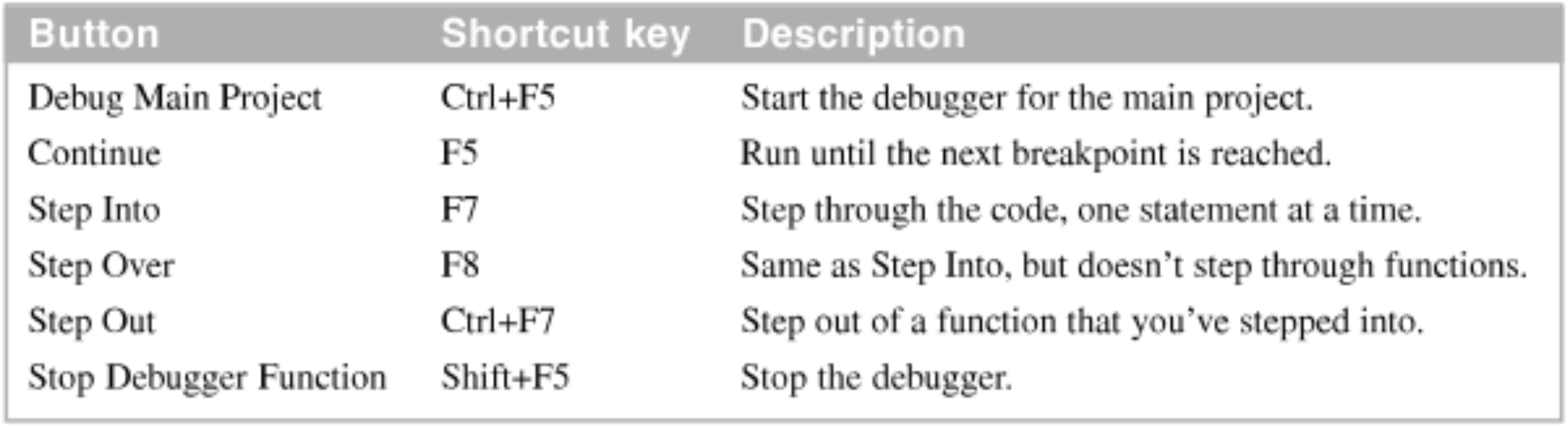
- When you run an application with the debugger, it stops when it encounters the first PHP statement in the application.

How to inspect variables

- When you being a debugging session, NetBeans stops on the first PHP statement that it encounters.
- At this point, you can use the buttons in the debugging toolbar to continue program execution.

How to inspect variables (cont.)

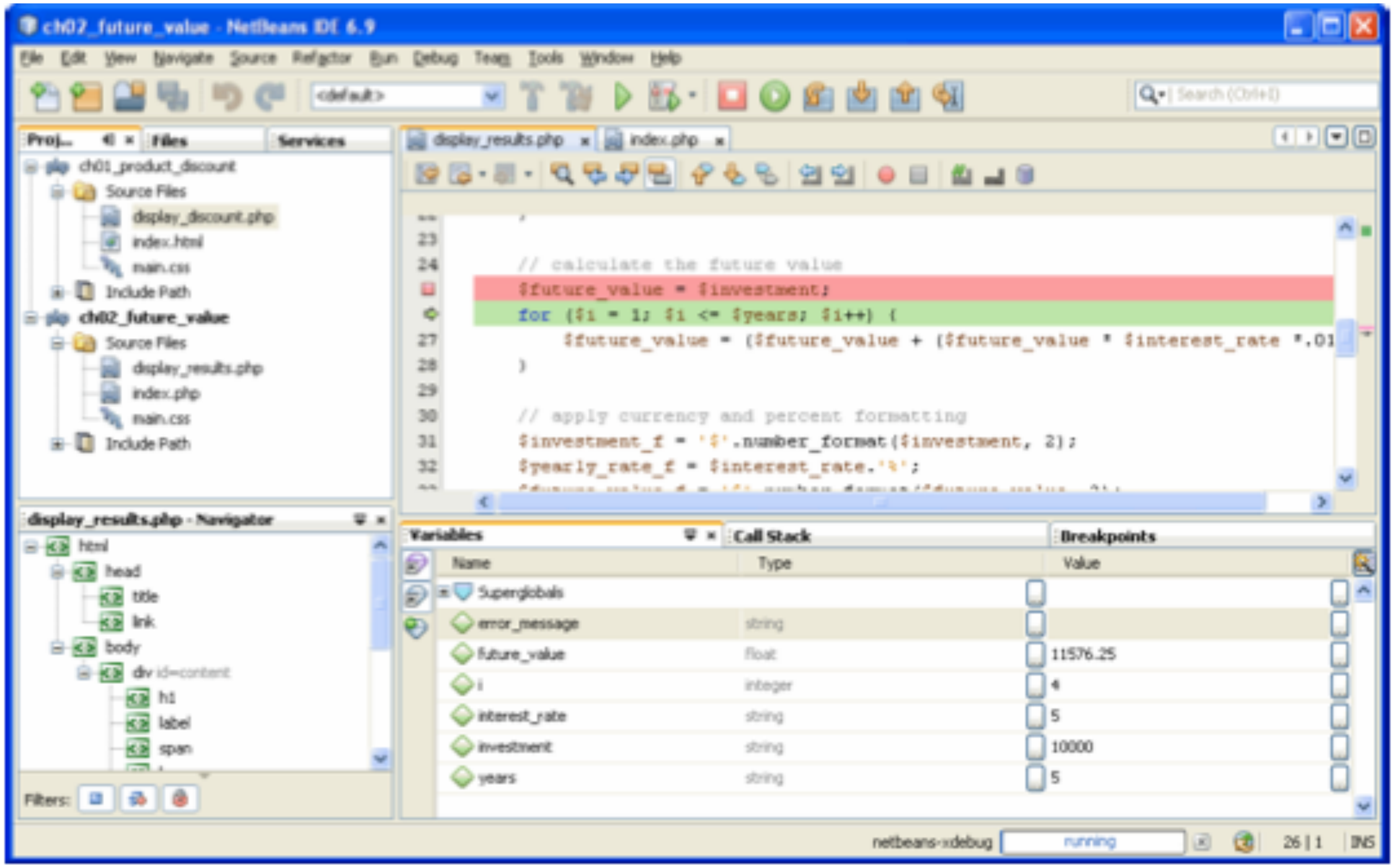
- The toolbar buttons and shortcut keys for stepping through the code



Button	Shortcut key	Description
Debug Main Project	Ctrl+F5	Start the debugger for the main project.
Continue	F5	Run until the next breakpoint is reached.
Step Into	F7	Step through the code, one statement at a time.
Step Over	F8	Same as Step Into, but doesn't step through functions.
Step Out	Ctrl+F7	Step out of a function that you've stepped into.
Stop Debugger Function	Shift+F5	Stop the debugger.

How to inspect variables (cont.)

- A debugging session with variables displayed



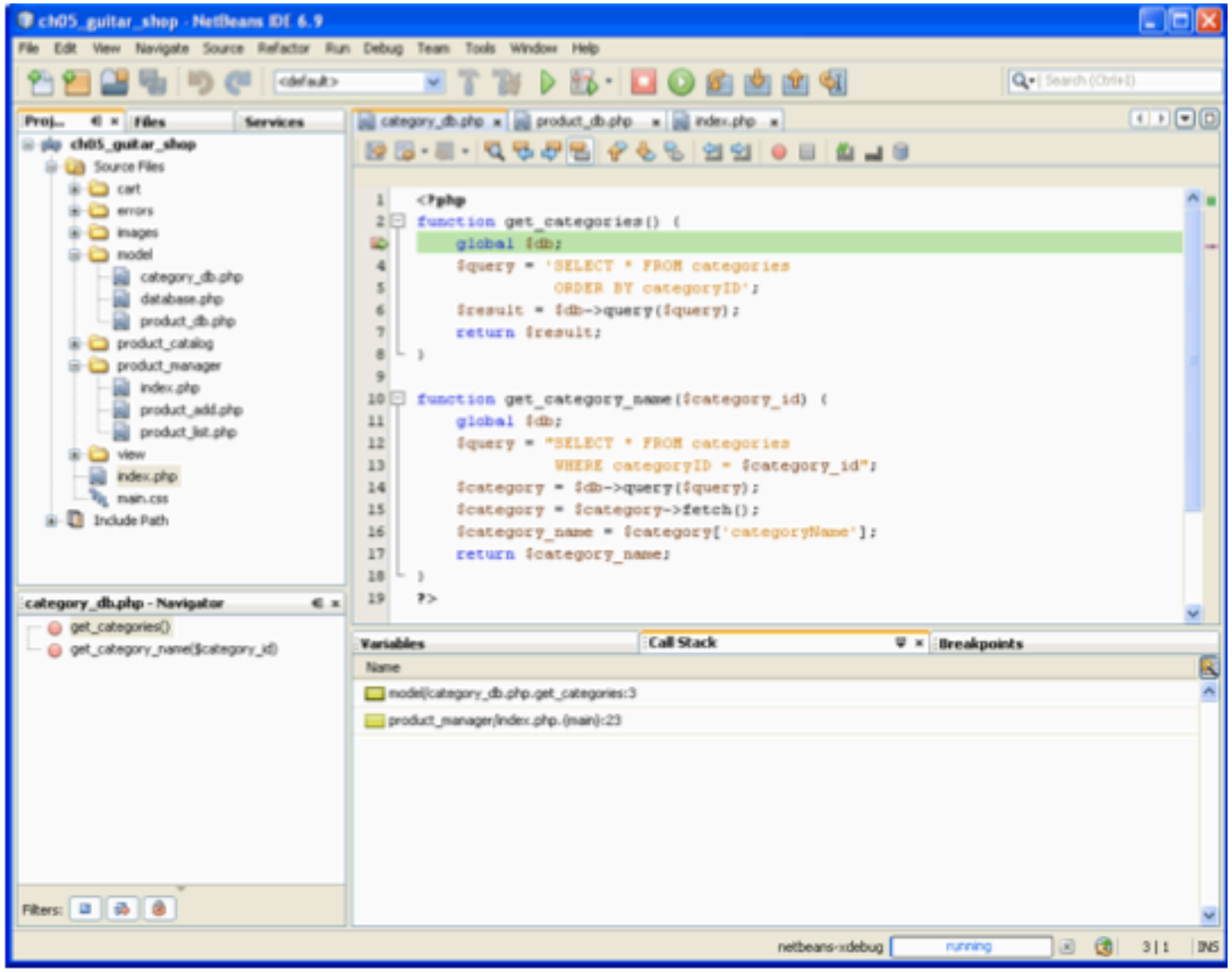
```
    // calculate the future value
    $future_value = $investment;
    for ($i = 1; $i <= $years; $i++) {
        $future_value = ($future_value + ($future_value * $interest_rate *.01
    }

    // apply currency and percent formatting
    $investment_f = '$'.number_format($investment, 2);
    $yearly_rate_f = $interest_rate.'%';
```

Name	Type	Value
Superglobals		
error_message	string	
future_value	float	11576.25
i	integer	4
interest_rate	string	5
investment	string	10000
years	string	5

How to inspect the stack trace

- The Call Stack window show the stack trace, which is a list of function in the reverse order in which they were called.
- A debugging session with a stack trace displayed.

How to inspect the stack trace (cont.)



```php
<?php
function get_categories() {
    global $db;
    $query = 'SELECT * FROM categories
              ORDER BY categoryID';
    $result = $db->query($query);
    return $result;
}

function get_category_name($category_id) {
    global $db;
    $query = "SELECT * FROM categories
              WHERE categoryID = $category_id";
    $category = $db->query($query);
    $category = $category->fetch();
    $category_name = $category['categoryName'];
    return $category_name;
}
?>
```

```
model/category_db.php.get_categories:3
product_manager/index.php.{main}:23
```

Summary

- The goal of testing is to find all the errors in an application.
- The goal of debugging is to fix all the errors before the application is put into production.
- Three type of errors can occur when you test an application: syntax errors, runtime errors, and logic errors.
- You can trace the execution of an application by inserting echo statement at appropriate.

Summary (2)

- XAMPP includes a debugger known as xDebug that can be used with an IDE like Netbeans.
- You can set breakpoints when you use a debugger.
- You can get a stack trace when you use a debugger.