Web database programming with PHP

- Overview
- Structured, semi-structured, unstructured data
- PHP
- A PHP Example
- Basic features of PHP
- Overview of PHP Database programming

Overview I

- How are databases used and accessed from the Internet.
- Many applications provide Web interfaces to access information stored in one or more databases.
- Internet database applications interact with the user via Web interfaces that display Web pages.
- Common method to specify contents/formatting of Web pages: **hypertext documents**

Overview II

- Languages for hypertext documents
  - HTML (HyperText Markup Language)
    - Used for generating **static** web pages
    - Not suitable for specifying database data
  - XML (eXtensible Markup Language)
    - Standard for exchanging data over the Web
    - Provides information on the **structure** of the data
  - PHP (a PHP interpreter provides a Hypertext Preprocessor that executes PHP commands in a text file, to create **Dynamic** web pages)

Overview III

- Dynamic Web pages: the flight info example
- PHP is used to program dynamic features into Web pages.
- To access a database via PHP, we need to include a library of PHP functions in the PHP interpreter
- PHP is an open source scripting language, written in C
- PHP programs are executed on the Web server computer (as opposes to Javascript for instance)

Structured, semi-structured, and unstructured data

- Structured data
  - Information stored in a database
  - Represented in a strict format (tables/attributes, objects)
  - Limitation: Not all data collected is structured
- Semi-structured data
  - Data may have certain structure but not all information collected has identical structure
  - Some attributes may exist in some of the entities of a particular type but not in others
  - There are data models to represent sem.struct. data using trees of graphs
- Unstructured data
  - Very limited indication of data type
    - E.g., a simple text document

Semi-structured data

- Graph representation of semi-structured data
  Note the difference between the two workers' data

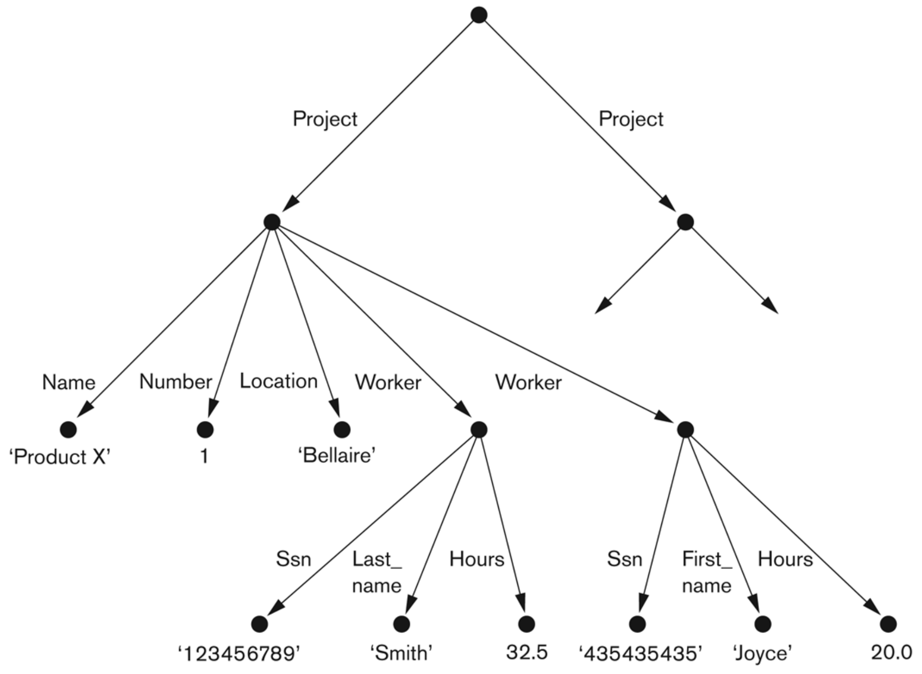


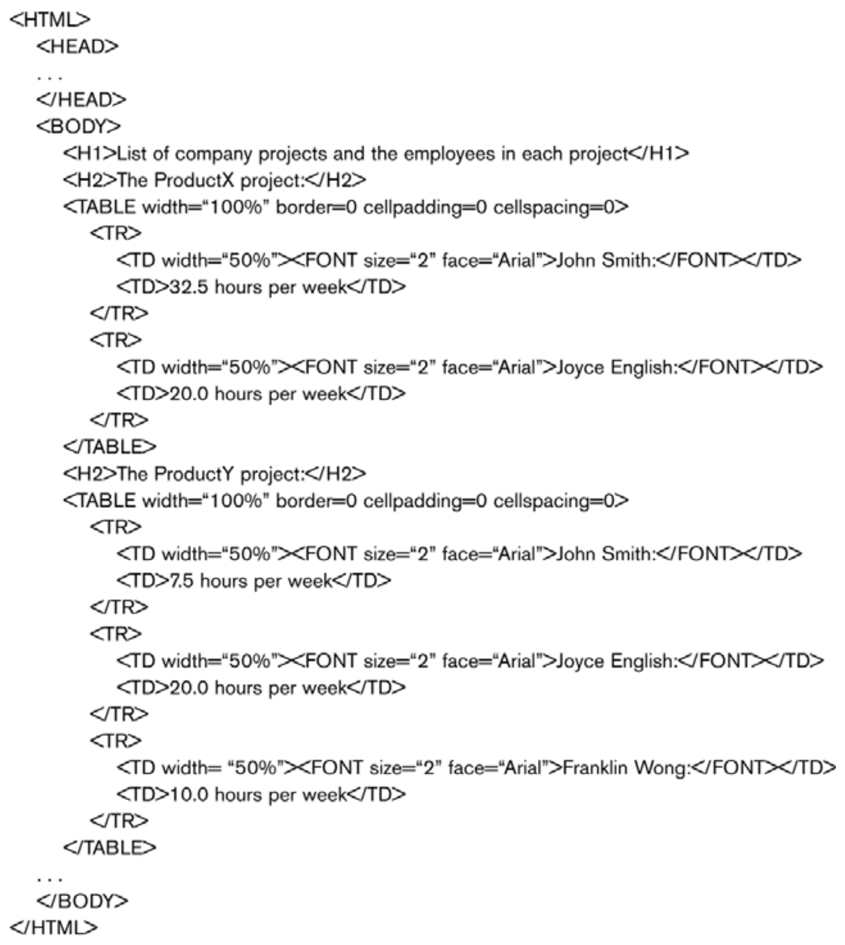
Figure 26.1
Representing semistructured data as a graph.

Semi-structured data

- Key difference between semi-structured and structured data:
  - Semi-structured data values are **mixed in** with their schema
    (i.e with the attribute names, relationships, entity types)
- **Example:** collect a list of bibliographic references (books, tech. reports, research papers in journals/conference proc.) may have different attributes new types of references appear: web pages, tutorials ...

Unstructured data

- Limited indication of data types
  - HTML web pages in contain some unstructured data
  - part of an HTML document representing unstructured data
  - HTML **tags**: <…> <…/>
  - tags **mark up** the document to instruct the HTML preprocessor how to display the text between a **start tag** and an **end tag**



```
<HTML>
  <HEAD>
  . . .
  </HEAD>
  <BODY>
    <H1>List of company projects and the employees in each project</H1>
    <H2>The ProductX project:</H2>
    <TABLE width="100%" border=0 cellpadding=0 cellspacing=0>
      <TR>
        <TD width="50%"><FONT size="2" face="Arial">John Smith:</FONT></TD>
        <TD>32.5 hours per week</TD>
      </TR>
      <TR>
        <TD width="50%"><FONT size="2" face="Arial">Joyce English:</FONT></TD>
        <TD>20.0 hours per week</TD>
      </TR>
    </TABLE>
    <H2>The ProductY project:</H2>
    <TABLE width="100%" border=0 cellpadding=0 cellspacing=0>
      <TR>
        <TD width="50%"><FONT size="2" face="Arial">John Smith:</FONT></TD>
        <TD>7.5 hours per week</TD>
      </TR>
      <TR>
        <TD width="50%"><FONT size="2" face="Arial">Joyce English:</FONT></TD>
        <TD>20.0 hours per week</TD>
      </TR>
      <TR>
        <TD width= "50%"><FONT size="2" face="Arial">Franklin Wong:</FONT></TD>
        <TD>10.0 hours per week</TD>
      </TR>
    </TABLE>
  . . .
  </BODY>
</HTML>
```

Figure 26.2
Part of an HTML document representing unstructured data.

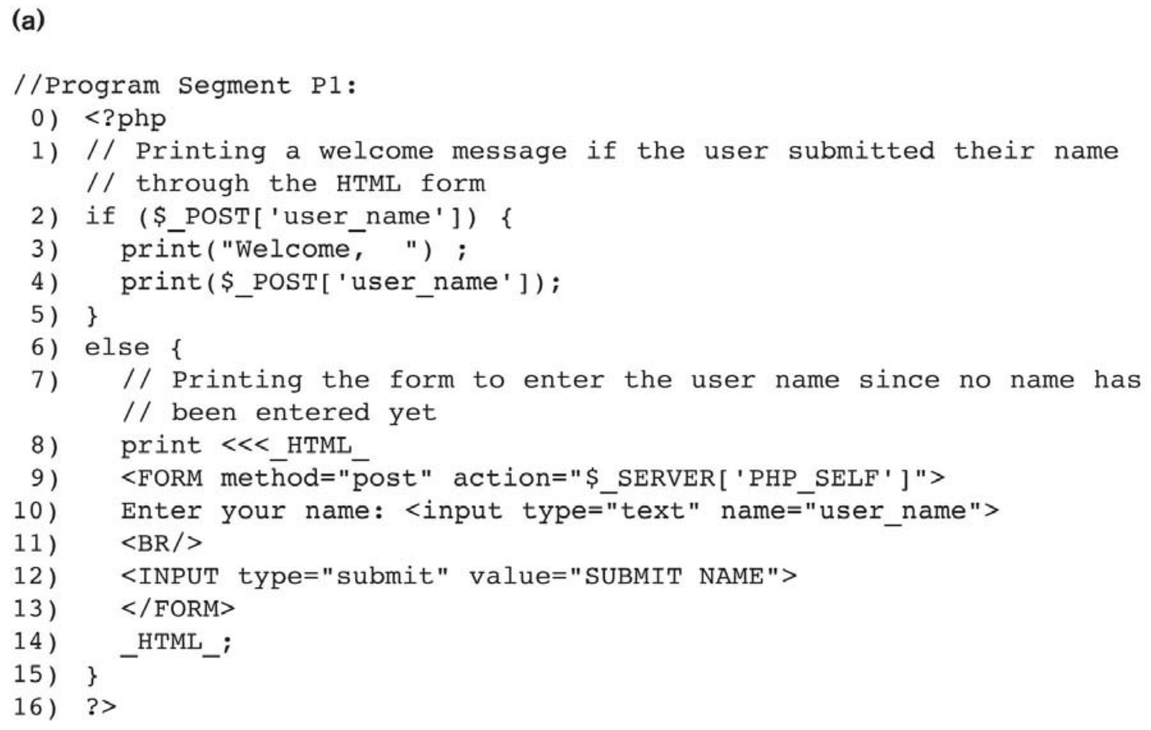
PHP

- Open source general-purpose scripting language, whose interpreter engine is written in C
- Particularly suited for manipulation of text pages
- Has libraries of functions for accessing databases, for various types of relational database systems such as Oracle, MySQL and any ODBC-compliant system

A simple PHP Example

- Suppose the file containing program segment P1 is stored at www.myserver.com/example/greeting.php

(a)



```
//Program Segment P1:
<?php
// Printing a welcome message if the user submitted their name
// through the HTML form
if ($_POST['user_name']) {
  print("Welcome,  ") ;
  print($_POST['user_name']);
}
else {
  // Printing the form to enter the user name since no name has
  // been entered yet
  print <<<_HTML_
  <FORM method="post" action="$_SERVER['PHP_SELF']">
  Enter your name: <input type="text" name="user_name">
  <BR/>
  <INPUT type="submit" value="SUBMIT NAME">
  </FORM>
  _HTML_;
}
?>
```

A simple PHP Example

- When the user accesses this URL, the PHP interpreter will start interpreting the PHP commands and will produce the form shown:

(a)

```
//Program Segment P1:
<?php
// Printing a welcome message if the user submitted their name
// through the HTML form
if ($_POST['user_name']) {
  print("Welcome,  ") ;
  print($_POST['user_name']);
}
else {
  // Printing the form to enter the user name since no name has
  // been entered yet
  print <<<_HTML_
  <FORM method="post" action="$_SERVER['PHP_SELF']">
  Enter your name: <input type="text" name="user_name">
  <BR/>
  <INPUT type="submit" value="SUBMIT NAME">
  </FORM>
  _HTML_;
}
?>
```

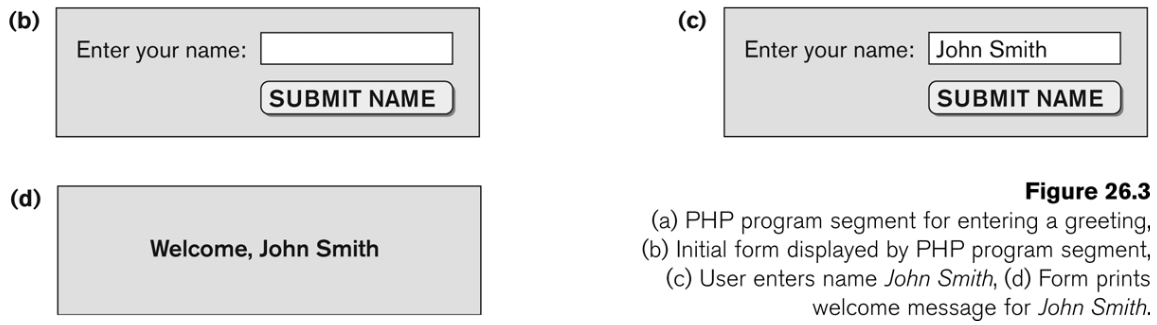


Figure 26.3
(a) PHP program segment for entering a greeting, (b) Initial form displayed by PHP program segment, (c) User enters name *John Smith*, (d) Form prints welcome message for *John Smith*.

Notes on the PHP program segment

- If the user types this URL on the browser, the PHP interpreter will start interpreting the code and produce the form shown in (b)
- Line 0 shows the PHP start tag <?php,
  which indicates to the PHP interpreter engine that it should process all subsequent text lines until it encounters the PHP end tag ?>
- Text outside of these tags is printed as is. This allows PHP code segments to be included within a larger HTML file.
- Only the sections in the file between <?php and ?> are processed by the PHP preprocessor
- Line 1 shows how to enter comments in a PHP program: lines starting with //
- Line 2 contains a predefined PHP variable $_POST, an array that holds all the values entered through form parameters

- Arrays in PHP are dynamic, i.e. no fixed number of elements, indexed by numbers or strings (associative arrays)
- $_POST is an associative array indexed by the name of the posted value user_name that is specified in the name attribute of the input tag on line 10
- So $_POST['user_name'] will contain the value typed by the user
- When the web page is first accessed, the if condition in line 2 will evaluate to false, because $_POST['user_name'] does not yet have a value
- So the PHP interpreter will execute lines 6-15, which create the text for an HTML file that displays the form shown in (b), this form will be displayed at the client side by the browser
- Line 8 creates a long text string in an HTML file
- All text between an opening <<<_HTML and a closing _HTML; is printed into the HTML file as is
- The closing _HTML; must appear alone on a separate line
- So the text added to the HTML file sent to the client will be the text between lines 9-13. This includes HTML tags to create the form in (b)

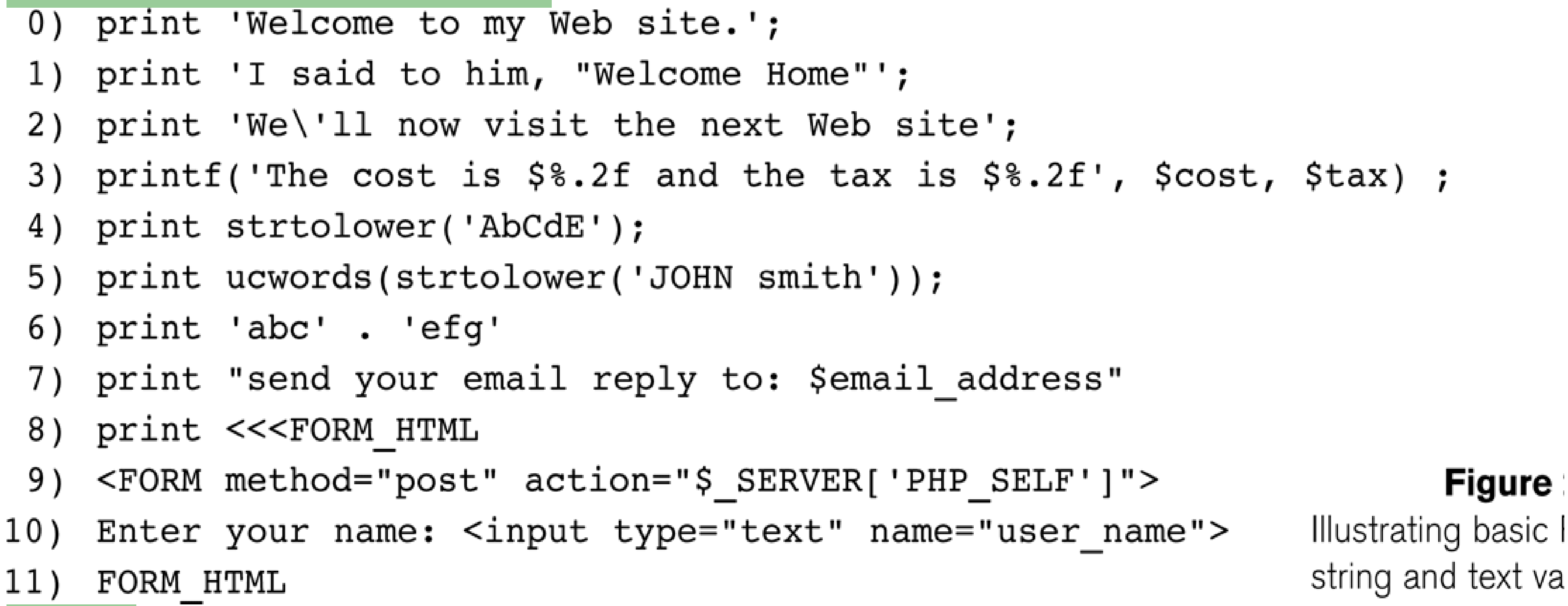
- The PHP predefined variable $_SERVER (line 9) , is an array that contains information about the local server
- The element $_SERVER['PHP_SELF'] of the array is the path name of the PHP file currently being executed on the server
- The action attribute of the form tag (line 9) instructs the PHP interpreter to reprocess the same file, once the form parameters are entered by the user
- Once the user types John Smith in the text box and clicks on the SUBMIT NAME button, the program segment is reprocessed
- Now $_POST['user_name'] contains the value/string "John Smith", so lines 3 and 4 will be placed in the HTML file sent to the client, which displays the message in (d)

Overview of basic features of PHP

- PHP variables, data types, and programming constructs
  - Variable names start with $ and can include characters, letters, numbers, and _ characters
    - No other special characters are permitted
    - Variable names are case sensitive
    - Variable names cannot start with a digit
  - Variables are **not typed**
    - The values assigned to variables determine their type
    - Assignments can change the type
  - Variable assignments are made by the operator =

Overview of basic features of PHP

- PHP types of string values:
- **Single-quoted strings** (lines 0, 1, 2) escape character: \
- **Double-quoted strings** (line 7)
  Variable names appearing within the string are replaced by their values. (this is called Variable Interpolation)
  it does not occur in single-quoted strings
- **Here documents** (lines 8-11)
  Enclose a part of a document between <<<DOCNAME and end it with a single line containing the document name DOCNAME (Variable Interpolation occurs)



```
print 'Welcome to my Web site.';
print 'I said to him, "Welcome Home"';
print 'We\'ll now visit the next Web site';
printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
print strtolower('AbCdE');
print ucwords(strtolower('JOHN smith'));
print 'abc' . 'efg'
print "send your email reply to: $email_address"
print <<<FORM_HTML
<FORM method="post" action="$_SERVER['PHP_SELF']">
Enter your name: <input type="text" name="user_name">
FORM_HTML
```

Figure
Illustrating basic
string and text va

- the period . serves as a string concatenation operator
- other string functions: strtolower, ucwords
- rule of thumb: use single-quoted strings when no variables are present, use double-quoted strings or here documents when variables need to be interpolated

Overview of basic features of PHP

- PHP has numeric data types for integers, floats, generally following the C types
- PHP has for-loops, while-loops, if-statements
- PHP has Boolean logic
  - True/false is equivalent no non-zero/zero
  - Comparison operators
    - ==, !=, >, >=, <, <=

Overview of basic features of PHP

- **PHP Arrays**
  - Allows to form lists of elements
  - Used frequently in forms that employ pull-down menus, to hold the list of choices
  - Can be 1-dimensional or multi-dimensional
  - 2-dim. arrays are used for relational database data
  - Arrays can be **numeric** or **associative**
    - Numeric array is based on a numeric index (starts ate zero)
    - Associative array is based on a key => value relationship
    - Element values are accessed via their keys. Keys are unique.

Overview of basic features of PHP

- Examples of two PHP Arrays
  - Line 0: $teaching is a **associative** array
    - Line 1 shows how the array can be updated/accessed
  - Line 5: $courses is a **numeric** array (No key is provided)

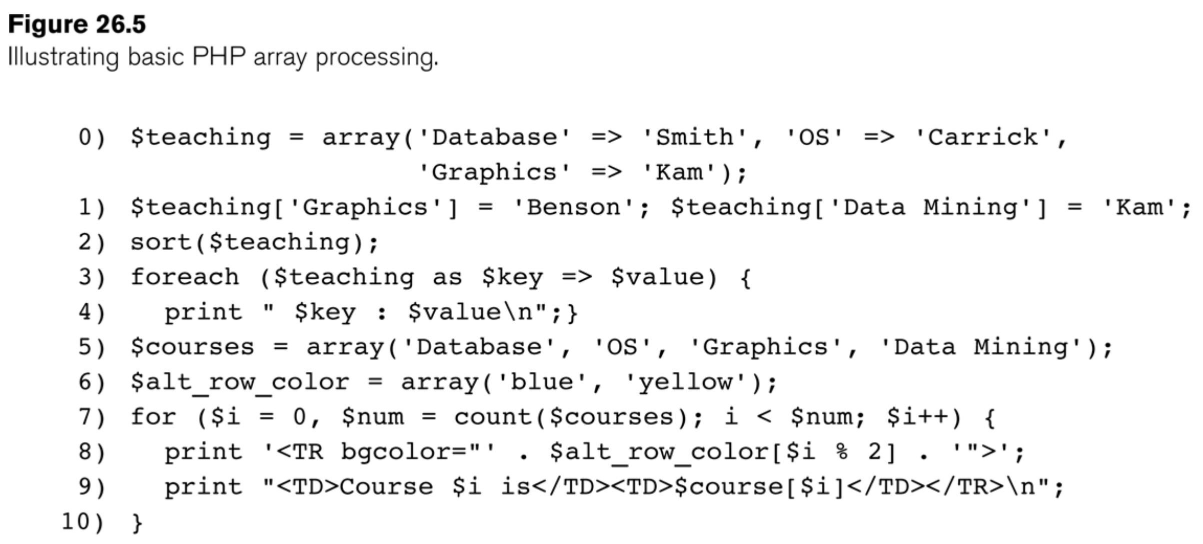


Figure 26.5
Illustrating basic PHP array processing.

```
 0) $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                      'Graphics' => 'Kam');
 1) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
 2) sort($teaching);
 3) foreach ($teaching as $key => $value) {
 4)   print " $key : $value\n";}
 5) $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
 6) $alt_row_color = array('blue', 'yellow');
 7) for ($i = 0, $num = count($courses); i < $num; $i++) {
 8)   print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
 9)   print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10) }
```

Overview of basic features of PHP

- The sort function sorts the array based on the elements values (not the keys)
- The count function returns the current number of elements in the array
- Looping mechanisms for PHP Arrays
  - Line 3 and 4 show “**for each**” construct for looping through each and every element in the array
  - Line 7 and 10 show a traditional “**for loop**” construct for iterating through an array

Overview of basic features of PHP

```
//Program Segment P1':
 0)  function display_welcome() {
 1)      print("Welcome,  ") ;
 2)      print($_POST['user_name']);
 3)  }
 4)
 5)  function display_empty_form(); {
 6)  print <<<_HTML_
 7)  <FORM method="post" action="$_SERVER['PHP_SELF']">
 8)  Enter your name: <INPUT type="text" name="user_name">
 9)  <BR/>
10)  <INPUT type="submit" value="Submit name">
11)  </FORM>
12)  _HTML_;
13)  }
14)  if ($_POST['user_name']) {
15)    display_welcome();
16)  }
17)  else {
18)    display_empty_form();
19)  }
```

- PHP Functions
  - two functions:
    - display_welcome()
    - display_empty_form()
  - Lines 14-19 show function calls

The function course_instructor($course,$teaching_assignments) has 2 parameters

$course	string holding the course name
$teaching_assignments	assoc. array holding teaching assignments

the function finds the name of the instructor who teaches a course.

the function call in Line 11 will return the string “Smith is teaching Database”.

```
0)  function course_instructor ($course, $teaching_assignments) {
1)    if (array_key_exists($course, $teaching_assignments)) {
2)      $instructor = $teaching_assignments[$course];
3)      RETURN "$instructor is teaching $course";
4)    }
5)    else {
6)      RETURN "there is no $course course";
7)    }
8)  }
9)  $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                       'Graphics' => 'Kam');
10) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
11) $x = course_instructor('Database', $teaching);
12) print($x);
13) $x = course_instructor('Computer Architecture', $teaching);
14) print($x);
```

Overview of basic features of PHP

- PHP Observations
  - Built-in PHP function **array_key_exists($k,$a)** returns true if the value in $k exists as a key in the associative array $a
  - Function arguments are **passed by value**
  - Return values are placed after the RETURN keyword. Functions can return any value.
  - Scope rules apply as with other programming languages

Overview of basic features of PHP

- PHP Server Variables and Forms
  - There a number of built-in entries in PHP functions
  - Example: built-in array variable **$_SERVER**
    - **$_SERVER['SERVER_NAME‘]**
      - This provides the Website name of the server computer where PHP interpreter is running
    - **$_SERVER['REMOTE_ADDRESS']**
      - IP address of client user computer that is accessing the server
    - **$_SERVER['REMOTE_HOST']**
      - Website name of the client user computer

Overview of basic features of PHP

- **$_SERVER['PATH_INFO']**
  - The part of the URL address that comes after backslash (/) at the end of the URL
- **$_SERVER['QUERY_STRING']**
  - The string that holds the parameters in the URL after ?
  - Common usage: search parameters
- **$_SERVER['DOCUMENT_ROOT']**
  - The root directory that holds the files on the Web server

Another important built-in array variable: **$_POST**

Provides the program with input values submitted via an HTML form (<INPUT> tag)