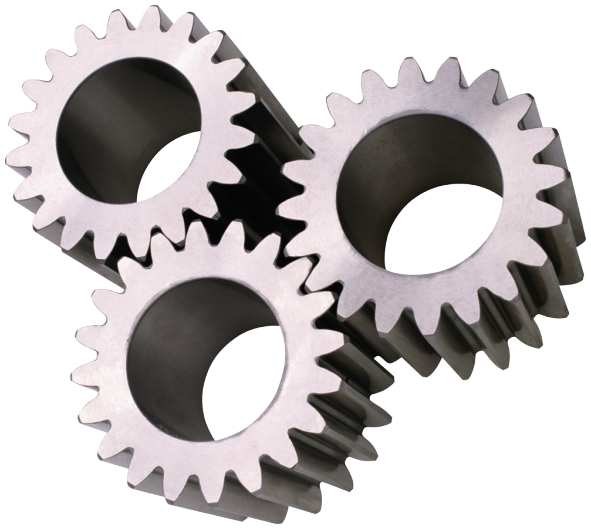
COSC344
Database Theory and Applications



Lecture 14:
PHP & SQL

Overview

- Last Lecture
  - Java & SQL
- This Lecture
  - PHP & SQL
  - Revision of the first half of the lectures
  - Source: Lecture notes,
    Textbook: Chapter 11
    Program examples
- Next Lecture
  - View & NULL

Goals of this lecture

- Not to teach advanced topics on PHP & SQL programming, but provide the basic knowledge
- Using sample codes to give you a taste for PHP & SQL programming
- Provide some useful references for further study

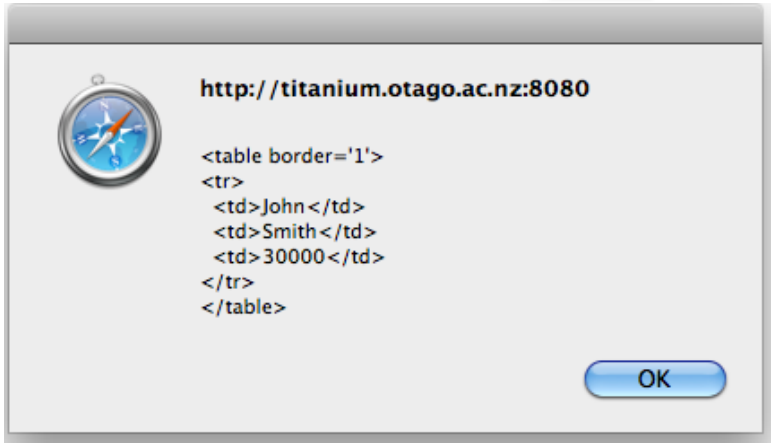
PHP

- 'PHP: Hypertext Preprocessor': an open source general-purpose scripting language
  - Interpreter engine is written in C programming language
  - PHP scripts are executed on the server
  - Suitable to generate dynamic web pages
- PHP supports many databases
  - Oracle,
  - MySQL
  - PostgreSQL
  - Informix
  - …

An Example


```
<html>
<body>
<h1>Salary Lookup</h1>

<div id="descriptionNode">This page finds salary for an employee</div>
<form name="EmployeeForm" method="post" action="http://titanium.otago.ac.nz:
8080/~USERNAME/employee_salary.php">

  Enter an Employee ID:
  <input type="text" value="" name="id">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

An Example (cont'd)




```
<?php

if (!isset($_POST['id'])) {
  echo 'No id passed';
} else {
$id = $_POST['id'];

  include("/home/includes/USERNAME/connenv");
  $query = 'select fname,lname, salary from employee where ssn = :id';

  $s = oci_parse($conn, $query);
  oci_bind_by_name($s, ":id", $id);
  oci_execute($s);

  echo "<table border='1'>".PHP_EOL;
  while ($row = oci_fetch_array($s, OCI_ASSOC+OCI_RETURN_NULLS)) {
                echo "<tr>".PHP_EOL;
                foreach ($row as $item) {
                  echo "  <td>".($item?htmlentities($item):" ")."</td>".PHP_EOL;
                }
                echo "</tr>".PHP_EOL;
  }
  echo "</table>".PHP_EOL;
}

oci_free_statement($s);
oci_close($conn);

?>
```

PHP Variables

- PHP variables
  - Start with the $ symbol
  - Case sensitive
  - Not typed: determined by the assigned value
- Reserved Variables
  - Form variables ($_POST, $_GET)
  - Server variables ($_SERVER)
- String values and variables
  - Single-quoted strings: most escape sequences will not be interpreted except \'
  - Double-quoted strings: interpret more escape sequences for special characters

PHP Array (1)

- Array are important in PHP as they allow lists of elements
  - A *single-dimensional array* can be used to hold the list of choices in the pull-down menu in forms.
  - A *two-dimensional array* can be used to hold the query result.
- **Numeric array:** associates a numeric index

```
<? PHP

   $emp_name[0]= “John”;
   $emp_name[1]= “B”;
   $emp_name[2]= “Smith”;

   echo $emp_name[0], “ ”, $emp_name[1], “ ”, $emp_name[2];

?>
```

PHP Array (2)

- **Associative array:** provide pairs of (key=>value).
  - All key values in a particular array must be unique

```
<? PHP
   $emp_name=array(“fname”=>“John”, “minit”=>”B”, “lname”=>“Smith”);
?>
```

```
<? PHP
   $emp_name[“fname”]= “John”;
   $emp_name[“minit”]= “B”;
   $emp_name[“lname”]= “Smith”;
?>
```

  - If we provide values without keys, the keys are automatically numeric and numbered 0,1,2, …

```
<? PHP
   $emp_name=array(“John”, ”B”, “Smith”);
?>
```

PHP Functions

- Functions MUST be defined before they can be called
- Return type does not need to be specified
- Unlike variables, function names are not case sensitive

```
<?php
    // This is a function
    function foo($arg_1, $arg_2)
    {
      $arg_2 = $arg_1 * $arg_2;
      return $arg_2;
    }

    $result_1 = foo(12, 3);
    echo $result_1;              // Outputs 36
    echo foo(12, 3);             // Outputs 36
?>
```

Create/Close database Connection

- Create a database connection

resource oci_connect (string $username, string $password, [string *$connection_string*])

- Close a database connection

bool oci_close (resource *$connection*)

```
<?php
// Create connection to Oracle
$conn = oci_connect("phphol", "welcome”);
if (!$conn) {
   $m = oci_error();
   echo $m['message'], "\n";
   exit;
}
else {
   print "Connected to Oracle!";
}
// Close the Oracle connection
oci_close($conn);
?>
```

Setting for Labs

- Environmental variable settings
  - The connenv file
  - Include this file in your PHP scripts to connect to Oracle

```
<?php

// set up Oracle environmental variables
//
putenv("ORACLE_SID=".getenv("ORACLE_SID"));
putenv("ORACLE_HOME=".getenv("ORACLE_HOME"));
putenv("ORACLE_BASE=".getenv("ORACLE_BASE"));
putenv("TWO_TASK=".getenv("TWO_TASK"));

$conn = oci_connect("ORACLE-USERNAME", "ORACLE-PASSWORD");

?>
```

Fetching Data

Four steps to query a database and display the results in a webpage

- **Parse** the statement for execution.

  resource **oci_parse** (resource *$connection* , string *$sql_text*)

- **Bind** data values (optional).

  bool **oci_bind_by_name** (resource *$statement* , string *$bv_name* , mixed *&$variable*)

- **Execute** the statement.

  bool **oci_execute** (resource *$statement*)

- **Fetch** the results from the database.

  array **oci_fetch_array** (resource *$statement*)

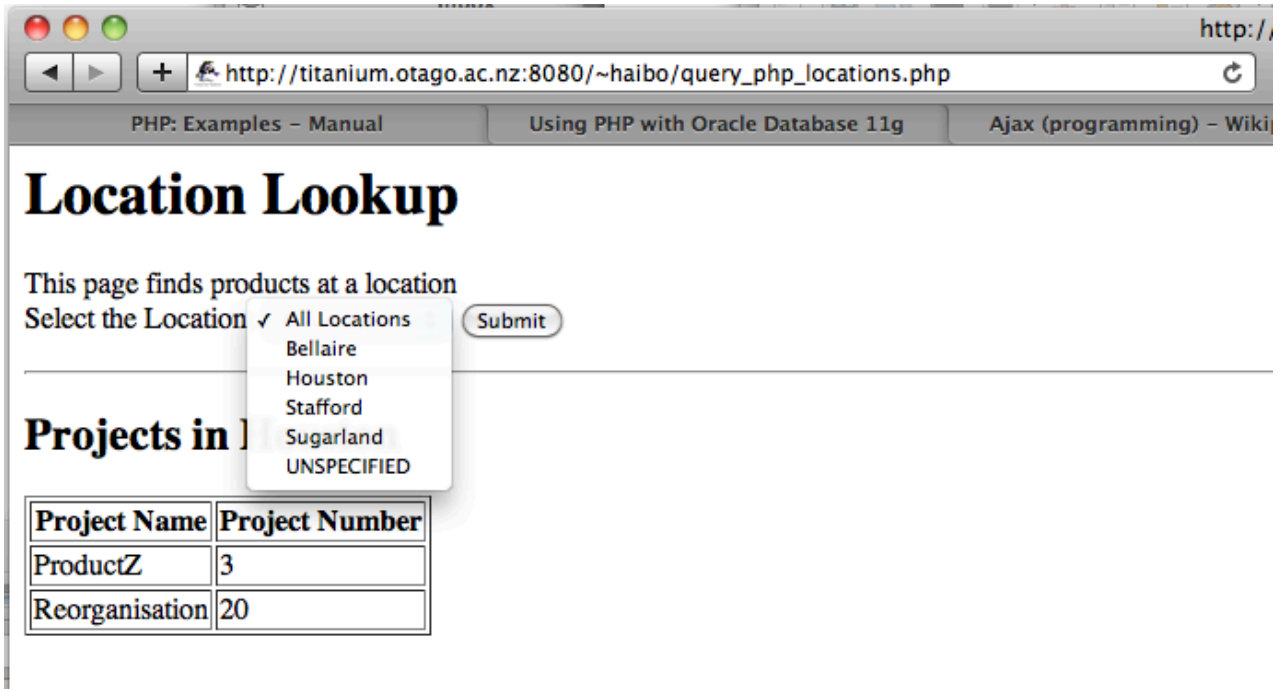
  – This function retrieves a row of results of the query as an associative array.

Fetch Data Example

```
<?php

// Create connection to Oracle
// include file containing your login details

include("/home/includes/USERNAME/connenv");

$query = 'select * from department';
$stid = oci_parse($conn, $query);
$r = oci_execute($stid);

// Fetch each row in an associative array

print '<table border="1">';
while ($row = oci_fetch_array($stid, OCI_RETURN_NULLS+OCI_ASSOC)) {
   print '<tr>';
   foreach ($row as $item) {
       print '<td>'.($item !== null ? htmlentities($item, ENT_QUOTES) : ' ').'</td>';
   }
   print '</tr>';
}
print '</table>';

?>
```

A more complex example

- Retrieve the projects located in a specified location
  - A drop down menu to show all options for locations
  - Display the output in the same page
- Source code
  - /coursework/344/oracle-php/query_php_locations.php



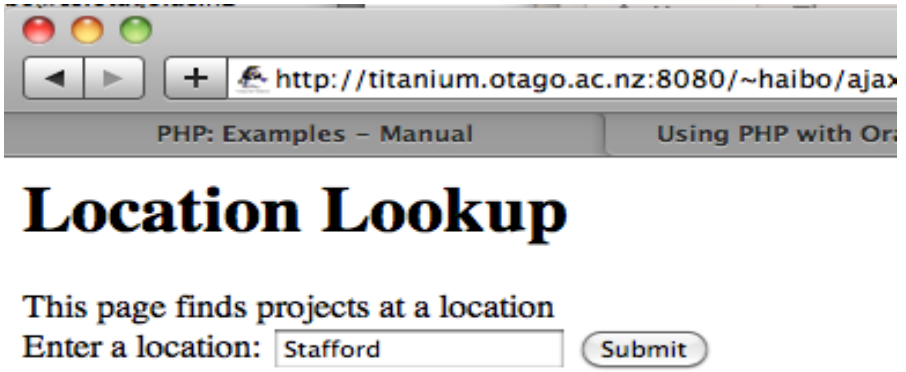
Incorporating Ajax (1)

- Ajax - asynchronous JavaScript and XML
  - web applications can send data to, and retrieve data from, a server asynchronously without interfering with the display and behavior of the existing page.
  - Avoid full page reloads.
  - JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously.

Incorporating Ajax (2)

- Client side

```
<html>
<head>
<script type="text/javascript">
  function makeRequest(id)
  {
     httpRequest = new XMLHttpRequest();
     httpRequest.open('GET', 'http://titanium.otago.ac.nz:8080/~USERNAME/ajax_query.php?id=' + id);
     httpRequest.onreadystatechange = function()
     {
        if (httpRequest.readyState == 4) {       // The request is complete
          alert(httpRequest.responseText);       // Display the result
        }
     }
     httpRequest.send(null);
  }
  </script>
</head>

<body>
<h1>Location Lookup</h2>
<div id="descriptionNode">This page finds products at a location</div>
<form name="LocForm" method="post">
  Enter Employee ID:
  <input type="text" value="Stafford" name="id">
  <input type="button" value="Submit" onclick="makeRequest(LocForm.id.value);">
</form>
</body>
</html>
```



Incorporating Ajax (3)

- Server side

```
<?php

// This script queries the employee table which is loaded by running the sql script

if (!isset($_GET['id'])) {
  echo 'No id passed';
}
else {
  $id = $_GET['id'];

  $query = 'select pname, pnumber from project where plocation = :id';
  $s = oci_parse($conn, $query);
  oci_bind_by_name($s, ":id", $id);
  oci_execute($s);

  echo "<table border='1'>".PHP_EOL;
  while ($row = oci_fetch_array($s, OCI_ASSOC+OCI_RETURN_NULLS)) {
                echo "<tr>".PHP_EOL;
                foreach ($row as $item) {
                  echo "  <td>".($item?htmlentities($item):" ")."</td>".PHP_EOL;
                }
                echo "</tr>".PHP_EOL;
  }
  echo "</table>".PHP_EOL;
}

?>
```

Useful References

- http://www.php.net/manual/en/book.oci8.php

- http://php.resourceindex.com/

Revision

- Conceptual Design
  - ER modelling
- Relational Model and Relational Algebra
  - Relation schema and relation operators
- Logical Design
  - ER diagram to relation schema mapping
- DDL and DML languages
  - SQL
- Functional dependencies and Normalisation
- PL/SQL and Trigger
- Application programming
  - Java &SQL
  - C&SQL
  - PHP&SQL