### 开发和部署 Oracle 和 PHP

目的：本教程介绍如何为企业构建功能强大的 PHP 应用程序。

所需时间：大约 1 小时

http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm

### 主题

本教程包括下列主题：

|  |  |
| --- | --- |
|  | [概述](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#o) |
|  | [前提条件](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#o) |
|  | [1. 创建标准连接](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t1) |
|  | [2. 创建简单查询](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t2) |
|  | [3. 创建持久连接](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t3) |
|  | [4. 创建事务](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t5) |
|  | [5. 数据获取功能](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t6) |
|  | [6. 使用绑定变量](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t8) |
|  | [7. 使用存储过程](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t9) |
|  | [8. 使用集合](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t10) |
|  | [9. 错误处理](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t11) |
|  | [10. 使用 LOB：上载和查询图像](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t12) |
|  | [11. 使用 XML](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#txml) |
|  | [总结](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#s) |
|  | [附录：PHP 入门](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#app) |

**概述**

PHP 是一种流行的 Web 脚本语言，通常用于创建数据库驱动的 Web 站点。如果您要使用 PHP 和 Oracle 数据库开发 Web 应用程序，本教程提供了有关如何在 Oracle 上使用 PHP 的示例，以带您起步。如果您是初次接触 PHP，请参阅[附录：PHP 入门](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#app)，了解 PHP 语言。

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**前提条件**

开始本教程之前，您应该：

|  |  |
| --- | --- |
| **1.** | 安装 Oracle 数据库 11g 或 Oracle 数据库快捷版。 |
| **2.** | 安装 PHP 5.2.4。 |
| **3.** | 配置 Linux Apache 服务器。 |
| **4.** | 下载 [php.zip](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/php.zip) 文件，并将其解压缩到 Apache 服务器可以找到文件的目录（即$HOME/public\_html）中。 |

有关本前提条件部分中的第 1-3 步的安装指导，可在 [OTN](http://www.oracle.com/technology/global/cn/tech/php/index.html) 上找到。

注：<localhost> 是本教程中的主机名。如果不同，则将该值更改成您的主机名。此外，本教程中使用 HR，HRPWD 是假设的口令。如果您打算使用快捷版而非 Oracle 数据库 11g，则需要将整个教程的 SID 从 localhost/orcl 更改为 localhost/XE。

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**1. 创建标准连接**

要创建一个可在 PHP 脚本生命周期内使用的到 Oracle 的连接，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 查看 **$HOME/public\_html** 目录的 [**connect.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/connect.php) 文件中包含的以下代码。  <?php  // Create connection to Oracle  $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl");  if (!$conn) {  $m = oci\_error();  echo $m['message'], "\n";  exit;  }  else {  print "Connected to Oracle!";  }  // Close the Oracle connection  oci\_close($conn);  ?>  oci\_connect() 函数包含连接信息。在本示例中，使用简化的连接字符串。  如果脚本结束时会自动关闭连接，则不需要 oci\_close() 函数。 |
| **2.** | 打开一个 Web 浏览器，输入以下 URL 以显示输出：  **http://localhost/~phplab/connect.php**  如果连接成功，则显示“Connected to Oracle!”。  如果创建数据库连接过程中出现问题，则显示错误“Error connecting to Oracle”。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**2. 创建简单查询**

开发 Web 应用程序时的一个常见任务是，查询一个数据库然后在 Web 浏览器中显示结果。您可以使用许多函数来查询一个 Oracle 数据库，但查询的基础始终是相同的：

|  |  |
| --- | --- |
| **1.** | **分析**要执行的语句。 |
| **2.** | **绑定**数据值（可选）。 |
| **3.** | **执行**语句。 |
| **4.** | 从数据库中**获取**结果。 |

要创建一个简单查询并在 HTML 表格中显示结果，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 查看 **$HOME/public\_html** 目录的 [**query.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/query.php) 文件中包含的以下代码。  <?php  // Create connection to Oracle  $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl");  $query = 'select \* from departments';  $stid = oci\_parse($conn, $query);  $r = oci\_execute($stid);  // Fetch the results 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?htmlentities($item):' ').'</td>';  }  print '</tr>';  }  print '</table>';  // Close the Oracle connection  oci\_close($conn);  ?>  oci\_parse() 函数分析语句。  oci\_execute() 函数执行经过分析的语句。  oci\_fetch\_array() 函数将查询结果作为一个关联数组进行检索，包括空值。 |
| **2.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/query.php**  查询结果显示在 Web 浏览器中。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**3. 创建持久连接**

到 Oracle 的持久连接可以在多个脚本上重用。对 Oracle 环境所做的更改将反映在所有访问该连接的脚本中。本主题通过创建一个持久连接，然后用另一个脚本更改 Oracle 环境来对此进行了演示。

要创建一个可在多个 PHP 脚本中重用的持久连接，执行以下步骤：

|  |  |
| --- | --- |
| **1.** | 查看 **$HOME/public\_html** 目录的 [**pconnect.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/pconnect.php) 文件中包含的以下代码。  <?php  // Create a persistent connection to Oracle  // Connection will be reused over multiple scripts  $conn = **oci\_pconnect**("hr", "hrpwd", "//localhost/orcl");  if (!$conn) {  $m = oci\_error();  echo $m['message'], "\n";  exit;  }  else {  print "Connected to Oracle!";  }  // Close the Oracle connection oci\_close($conn);  ?>  oci\_pconnect() 函数创建了一个到 Oracle 的持久连接。  使用 oci\_close() 函数不会关闭持久连接，在该脚本中是多余的。 |
| **2.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/pconnect.php**  现在已经创建了持久连接。该连接仍可用于那些使用相同登录凭证并且由相同的 http 进程支持的脚本。 |
| **3.** | 查看 **$HOME/public\_html** 目录的 [**usersess.sql**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/usersess.sql) 文件中包含的以下代码。  column username format a30  column logon\_time format a18  set pagesize 1000 feedback off echo on  alter session set nls\_date\_format = 'DD-MON-YY HH:MI:SS';  select username, logon\_time from v$session where username is not null;  您现在已经创建了一个要在 SQL\*Plus 中运行的 SQL\*Plus（Oracle 的命令行 SQL 脚本工具）脚本文件。该 SQL\*Plus 脚本更改了数据库的国家语言字符日期格式，显示当前的数据库会话。日期格式更改只与 SQL\*Plus 会话有关，用于规定登录时间的输出格式。 |
| **4.** | 打开一个终端窗口，输入以下命令。注意，您也可以在 SQL Developer 中执行脚本。  **cd $HOME/public\_html**  **sqlplus system/oracle@//localhost/orcl**  **@usersess.sql**  SQL\*Plus 脚本列出当前数据库会话。PHP 脚本创建的会话仍处于活动状态，在结果的第一行显示为用户名 HR。尽管调用了 oci\_close() 函数，但这并不会关闭持久连接，该连接可以用于其他脚本。 |
| **6.** | 为了说明该持久连接正由其他 PHP 脚本重用且会话设置相同，查看**$HOME/public\_html** 目录的 [**pconnect2.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/pconnect2.php) 文件中包含的以下代码。  <?php  // Function to execute a query  function do\_query($conn, $query)  {  $stid = oci\_parse($conn, $query);  oci\_execute($stid);  oci\_fetch\_all($stid, $res);  echo "<pre>";  var\_dump($res);  echo "</pre>"; }  // Create a persistent connection to Oracle  $c = oci\_pconnect("hr", "hrpwd", "//localhost/orcl");  // Query the database system date  do\_query($c, "select sysdate from dual");  // Change the NLS Territory  $s = oci\_parse($c, "alter session set nls\_territory=germany");  $r = oci\_execute($s);  // Query the database system date again  do\_query($c, "select sysdate from dual");  ?>  该脚本将创建一个新的持久连接，或者使用相同的登录凭证重用现有的持久连接。  然后，该脚本使用 do\_query() 函数查询并获取数据库系统日期。它使用 var\_dump 调试函数输出包含日期查询结果的 PHP 变量的值和结构。  然后，该脚本更改国家语言地区设置以适合德国的格式显示输出，再次调用 do\_query()函数以再次显示数据库系统日期。 |
| **7.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/pconnect2.php**  注意，当 ALTER SESSION 命令更改为其他地区时，你将看到两个查询日期格式的区别。  查看使用持久连接重新加载脚本的效果。您可能希望执行该操作多次，直至重用原始 PHP 会话（Apache 进程）。两个查询现在使用相同的新的日期格式。这表明已重用该连接，当后面的脚本运行时，仍然会设置原来脚本中设置的日期格式。该连接仍然处于活动状态（持久的），可供其他使用相同登录凭证的 PHP 脚本使用。如果脚本更改为使用一个标准连接，将始终输出两种不同的时间格式。  您应该了解持久会话期间您进行的环境更改，因为它们可能还会影响其他脚本。但事务不会跨 PHP 脚本，未提交的数据将在脚本结束处回滚。 |
| **8.** | 再次运行 SQL\*Plus 脚本 **usersess.sql**，查看哪些连接处于打开状态。  现在，许多由 HR 用户创建的数据库会话都处于打开状态。这将显示当前可用的持久会话。在 Linux 上，Apache 作为多个独立进程运行。PHP 不会在进程间共享任何信息（包括连接）。由于您每次运行一个脚本时，该脚本都可能由其他 httpd 进程执行，因此当您使用 oci\_pconnect() 时，您最后可能有多个打开的数据库连接。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**4. 创建事务**

在 Oracle 数据库中操作数据（插入、更新或删除数据）时，更改的数据或新数据在提交到数据库之前仅在数据库会话中可用。更改的数据提交至数据库后，可供其他用户和会话使用。这是一个数据库事务。  
  
单独提交每个更改会额外增加服务器的负载。通常，您希望提交所有数据或者不提交任何数据。执行自己的事务控制具有性能和数据完整性优势。  
  
默认情况下，oci\_execute() 函数立即提交更改。  
  
如果数据无法自动提交，或者在您使用 oci\_commit() 将数据显式提交给数据库之前不能用于其他会话，则使用 OCI\_DEFAULT 参数。您也可用 oci\_rollback() 进行回滚。  
  
Oracle 建议您将 OCI\_DEFAULT 用作一个通常包含多个数据库交互的事务（即 DML）。  
  
要通过 Oracle 数据库了解 PHP 中的事务管理，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 在 SQL\*Plus 会话中，输入以下命令，以 HR 用户身份登录到数据库并创建一个新表：  **connect hr/hrpwd@//localhost/orcl create table mytable (col1 date);** |
| **2.** | 查看 **$HOME/public\_html** 目录的 [**trans1.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/trans1.php) 文件中包含的以下代码。  <?php     echo "<pre>";      // Execute a query     function do\_query($conn)     {       $stid = oci\_parse($conn,           "select to\_char(col1, 'DD-MON-YY HH:MI:SS') from mytable");       oci\_execute($stid, OCI\_DEFAULT);       oci\_fetch\_all($stid, $res);       foreach ($res as $v) {         var\_dump($v);       }     }      // Create a database connection     function do\_connect()     {       $conn = oci\_new\_connect("hr", "hrpwd", "//localhost/orcl");       return($conn);     }     $d = date('j:M:y H:i:s');      // Create a connection     $c1 = do\_connect();      // Insert the date into mytable     $s = oci\_parse($c1,         "insert into mytable values (to\_date('"         . $d . "', 'DD:MON:YY HH24:MI:SS'))");      // Use OCI\_DEFAULT to execute the statement without committing     $r = oci\_execute($s, OCI\_DEFAULT);      // Query the current session/connection     echo "Query using connection 1<br>\n";     do\_query($c1);      // Create a new connection and query the table contents     $c2 = do\_connect();     echo "<br>Query using connection 2<br>\n";     do\_query($c2);     echo "</pre>";     ?>  该脚本中使用了两个连接。  该脚本使用 oci\_new\_connect() 创建一个唯一的、非持久性的数据库连接，然后将日期插入到 mytable 表中并进行查询。  然后，该脚本再创建一个唯一的数据库连接，再次查询此表以显示对第二个连接可见的内容。 |
| **3.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/trans1.php**  该脚本使用 $c1 连接向表中插入一行。  数据尚未提交到数据库，因为每个 oci\_execute() 调用都使用 OCI\_DEFAULT，没有调用oci\_commit()。其他数据库用户目前还无法看到该行。使用第二个连接 $c2 的查询返回一个空数组。 |
| **4.** | 由于没有进行任何提交，因此在脚本结束时，PHP 将回滚数据。为了查看是否有尚未提交的数据，查询该表以查看是否有任何插入的行。在 SQL\*Plus 会话中，输入以下命令，从 mytable 表中选择任何行：  **select \* from mytable;** |
| **5.** | 查看 **$HOME/public\_html** 目录的 [**trans2.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/trans2.php) 文件中包含的以下代码。  <?php     echo "<pre>";      // Execute a query     function do\_query($conn)      {        $stid = oci\_parse($conn,            "select to\_char(col1, 'DD-MON-YY HH:MI:SS') from mytable");        oci\_execute($stid, OCI\_DEFAULT);        oci\_fetch\_all($stid, $res);        foreach ($res as $v) {          var\_dump($v);        }      }      // Create a database connection      function do\_connect()      {        $conn = oci\_new\_connect("hr", "hrpwd", "//localhost/orcl");        return($conn);      }      $d = date('j:M:y H:i:s');      // Create a connection      $c1 = do\_connect();      // Insert the date into mytable      $s = oci\_parse($c1,          "insert into mytable values (to\_date('"          . $d . "', 'DD:MON:YY HH24:MI:SS'))");      $r = oci\_execute($s);  // no OCI\_DEFAULT means automatically commit      // Query the current session/connection      echo "Query using connection 1<br>\n";      do\_query($c1);      // Create a new connection and query the table contents      $c2 = do\_connect();      echo "<br>Query using connection 2<br>\n";      do\_query($c2);      echo "</pre>";      ?>  该脚本与 trans1.php 的不同之处在于，插入数据时没有 OCI\_DEFAULT。这意味着将提交新数据。 |
| **6.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/trans2.php**  现在，数据已经提交，因此两个查询都将返回表中的新行。  重新加载页面。您每次进行重新加载时，都会看到表中添加了许多行。 |
| **7.** | 在 SQL\*Plus 会话中，输入以下命令，从 **mytable** 表中删除任何行：  **delete from mytable; commit;** |
| **8.** | 您可以比较单独提交每行与在事务结束后提交之间的性能差异。  要测试差异，查看 **$HOME/public\_html** 目录的 [**trans3.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/trans3.php) 文件中包含的以下代码。  <?php      function currTime()      {        $time = microtime();        $time = explode(' ', $time);        $time = $time[1] + $time[0];        return $time;      }      function elapsedTime($start)      {        return (currTime() - $start);      }      function do\_query($conn)      {        $stid = oci\_parse($conn,            "select count(\*) c from mytable");        oci\_execute($stid, OCI\_DEFAULT);        oci\_fetch\_all($stid, $res);        echo "Number of rows: ", $res['C'][0], "<br>";      }      function do\_delete($conn)      {        $stmt = "delete from mytable";       $s = oci\_parse($conn, $stmt);        $r = oci\_execute($s);      }      function do\_insert($conn)      {        $d = date('j:M:y H:i:s');        $stmt = "insert into mytable values (to\_date('"                . $d . "', 'DD:MON:YY HH24:MI:SS'))";        $s = oci\_parse($conn, $stmt);        $r = oci\_execute($s);      }      $c = oci\_connect("hr", "hrpwd", "//localhost/orcl");      $start = currTime();      for ($i = 0; $i < 10000; $i++) {        do\_insert($c);      }      $et = elapsedTime($start);      echo "Time was ".round($et,3)." seconds<br>";      do\_query($c);  // Check insert done      do\_delete($c); // cleanup committed rows      ?>  运行这段代码几次，您将了解插入 10,000 行所需的时间。 |
| **9.** | 现在，运行 [**trans4.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/trans4.php) 脚本。该脚本唯一的区别在于已插入 do\_insert() 函数OCI\_DEFAULT，这样就不会自动提交，并且在插入循环末尾添加了显式提交：  ...  function do\_insert($conn) {  $d = date('j:M:y H:i:s');  $stmt = "insert into mytable values  (to\_date('" . $d . "', 'DD:MON:YY HH24:MI:SS'))";  $s = oci\_parse($conn, $stmt);  **$r = oci\_execute($s, OCI\_DEFAULT);**  }  $c = oci\_connect("hr", "hrpwd", "//localhost/orcl");  $start = currTime();  for ($i = 0; $i < 10000; $i++) {  do\_insert($c);  }  **oci\_commit($c);**  $et = elapsedTime($start);  ...  重新运行该测试。插入时间减少。  通常，您希望提交所有数据或者不提交任何数据。进行您自己的事务控制具有性能和数据完整性优势。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**5. 数据获取功能**

从 Oracle 数据库中获取数组数据的方式有多种。您可以将数组作为关联数组和/或数字数组进行获取。

要了解如何使用数组获取参数，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 第一部分显示如何使用 oci\_fetch\_array() 的默认输出获取数组，它是通过关联索引和数字索引获取数组的。  查看 **$HOME/public\_html** 目录的 [**fetch.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/fetch.php) 文件中包含的以下代码。查看以下代码：  <?php echo "<pre>"; $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl"); $query = 'select \* from employees where employee\_id = 101'; $stid = oci\_parse($conn, $query); oci\_execute($stid); while ($row = oci\_fetch\_array($stid)) {  var\_dump($row); // display PHP's representation of $row } oci\_close($conn); echo "</pre>"; ?> |
| **2.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/fetch.php**  该输出显示结果既包含关联索引又包含数字索引。虽然这可以为您处理结果的方式提供更大的灵活性，但会产生很大的网络和内存开销。 |
| **3.** | 而您可能希望仅将数组作为一个关联数组获取。本部分显示如何只获取一个关联数组。  将 *oci\_fetch\_array()* 调用更改为如下内容：  oci\_fetch\_array($stid, OCI\_ASSOC)  重新运行以下 URL：  **http://localhost/~phplab/fetch.php**  如输出所示，OCI\_ASSOC 参数将数组作为一个关联数组获取。 |
| **4.** | 最后一个选项是将数组作为一个数字数组获取。本部分显示如何只获取一个数字数组。  再次将 oci\_fetch\_array() 调用更改为如下内容：  oci\_fetch\_array($stid, OCI\_NUM)  重新运行以下 URL：  **http://localhost/~phplab/fetch.php**  输出显示 OCI\_NUM 参数将数组作为一个数字数组获取。  您还可以使用其他 oci\_fetch\_array() 参数和组合，如：   * oci\_fetch\_array($stid, OCI\_BOTH)，它同时返回关联索引和数字索引 * oci\_fetch\_array($stid, OCI\_ASSOC+OCI\_RETURN\_NULLS)，它返回一个关联索引，包括 NULL。   PHP 文档包含获取选项的完整列表。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**6. 使用绑定变量**

绑定变量允许您使用新值重新执行查询，避免了重新分析语句的开销。 绑定变量提高了代码可重用性，降低了 SQL 注入攻击的风险。

要在本示例中使用绑定变量，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 查看 **$HOME//public\_html** 目录的 [**bind.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/bind.php) 文件中包含的以下代码。  <?php     function do\_fetch($myeid, $s)     {       // Fetch the results in an associative array       print '<p>$myeid is ' . $myeid . '</p>';       print '<table border="1">';       while ($row = oci\_fetch\_array($s, OCI\_RETURN\_NULLS+OCI\_ASSOC)) {           print '<tr>';           foreach ($row as $item) {               print '<td>'.($item?htmlentities($item):'&nbsp;').'</td>';           }           print '</tr>';       }       print '</table>';     }     // Create connection to Oracle     $c = oci\_connect("hr", "hrpwd", "//localhost/orcl");     // Use bind variable to improve resuability, and to     // remove SQL Injection attacks.     $query = 'select \* from employees where employee\_id = :eidbv';     $s = oci\_parse($c, $query);     $myeid = 101;     oci\_bind\_by\_name($s, ":EIDBV", $myeid);     oci\_execute($s);     do\_fetch($myeid, $s);     // Redo query without reparsing SQL statement     $myeid = 104;     oci\_execute($s);     do\_fetch($myeid, $s);     // Close the Oracle connection     oci\_close($c);     ?> |
| **2.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/bind.php**  $myeid 变量绑定到 :eidbv 绑定变量，因此重新执行该查询时，$myeid 的新值将传递到该查询。这使您可以再次执行该语句，而无需用新值对其进行重新分析，因此可以提高代码的性能。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**7. 使用存储过程**

PL/SQL 是 Oracle 对 SQL 的过程语言扩展。PL/SQL 存储过程和函数存储在数据库中，因此其访问速度非常快。使用 PL/SQL 存储过程允许所有数据库应用程序重用逻辑，无论应用程序以何种方式访问数据库。许多与数据相关的操作在 PL/SQL 中的执行速度比将数据提取到一个程序中（例如，PHP）然后再进行处理的速度快。

Oracle 允许 PL/SQL 和 Java 存储过程。在本教程中，您将创建一个 PL/SQL 存储过程并在一个 PHP 脚本中调用它。执行以下步骤：

|  |  |
| --- | --- |
| **1.** | 启动 SQL\*Plus，用以下命令创建一个新表 ptab：  **sqlplus hr/hrpwd@//localhost/orcl**  **create table ptab (mydata varchar(20), myid number);** |
| **2.** | 在 SQL\*Plus 中，使用以下命令创建一个存储过程 myproc，以将数据插入到 ptab 表中：  **create or replace procedure**  **myproc(d\_p in varchar2, i\_p in number) as**  **begin**  **insert into ptab (mydata, myid) values (d\_p, i\_p);**  **end;**  **/** |
| **3.** | 查看 **$HOME/public\_html** 目录的 [**proc.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/proc.php) 文件中包含的以下代码。查看以下代码：  **<?php**  **$c = oci\_connect('hr', 'hrpwd', '//localhost/orcl');**  **$s = oci\_parse($c, "call myproc('mydata', 123)");**  **oci\_execute($s);**  **echo "Completed";**  **?>** |
| **4.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/proc.php**  PHP 脚本通过调用存储过程 myproc 在 ptab 表中新建了一行。ptab 表新增了一行，值为“mydata”和 123。  切换到您的 SQL\*Plus 会话，查询表以显示新的行：    **select \* from ptab;** |
| **5.** | 将 proc.php 扩展为查询该表以检查数据是否已插入。将 **proc.php** 更改为如下内容：  **<?php**  **$c = oci\_connect('hr', 'hrpwd', '//localhost/orcl');**  **$s = oci\_parse($c, "call myproc('mydata', :bv)");**  **$v = 123;**  **oci\_bind\_by\_name($s, ":bv", $v);**  **oci\_execute($s);**  **echo "Completed";**  **?>**  使用 *oci\_bind\_by\_name()* 将一个 PHP 变量 $v 绑定到“:bv”，并尝试通过更改 $v 中的值来更改插入的值。  重新运行以下 URL：  **http://localhost/~phplab/proc.php**  再次查询该表以显示新的行：    **select \* from ptab;** |
| **6.** | 除了存储过程之外，通常还使用 PL/SQL 存储函数。在 SQL\*Plus 中，创建一个 PL/SQL 存储函数 myfunc()，以向 ptab 表中插入一行，并返回插入的 myid 值的两倍：  **create or replace function    myfunc(d\_p in varchar2, i\_p in number) return number as**  **begin**  **insert into ptab (mydata, myid) values (d\_p, i\_p);**  **return (i\_p \* 2);**  **end;**  **/** |
| **7.** | 查看 **$HOME/public\_html** 目录的 [**func.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/func.php) 文件中包含的以下代码。查看以下代码：  <?php     $c = oci\_connect('hr', 'hrpwd', '//localhost/orcl');     $s = oci\_parse($c, "begin :bv := myfunc('mydata', 123); end;");     oci\_bind\_by\_name($s, ":bv", $v, 10);     oci\_execute($s);     echo $v, "<br>\n";     echo "Completed";     ?>  由于要返回一个值，因此将 oci\_bind\_by\_name() 的可选长度参数设置为 10，这样 PHP 就可以分配能够存储 10 位的正确内存量了。  重新运行以下 URL：  **http://localhost/~phplab/func.php** |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**8. 使用集合**

PL/SQL 集合是一组相同类型（例如，数组类型）的有序元素。

要在 PHP 中使用 PL/SQL 集合，执行以下步骤：

|  |  |
| --- | --- |
| **1.** | 首先需要创建一个简单表和新过程 myproc()。该过程接受一个数组，并使用 Oracle 的快速批量插入“FORALL”语句插入数组中的所有元素。查看 **$HOME/public\_html** 目录的 [**proc2.sql**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/proc2.sql) 文件中的代码。  drop table ptab;  create table ptab(name varchar2(20));  create or replace package mypkg as  type arrtype is table of varchar2(20) index by pls\_integer;  procedure myproc(p1 in out arrtype);  end mypkg;  /  create or replace package body mypkg as  procedure myproc(p1 in out arrtype) is  begin  forall i in indices of p1  insert into ptab values (p1(i));  end myproc;  end mypkg;  /  在终端窗口中，执行以下命令：  **sqlplus hr/hrpwd@//localhost/orcl**  **@proc2** |
| **2.** | 查看 **$HOME/public\_html** 目录的 [**coll.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/coll.php) 文件中包含的以下代码。  <?php  function do\_query($conn)  {  echo "<pre>";  $stid = oci\_parse($conn, "select \* from ptab");  oci\_execute($stid, OCI\_DEFAULT);  oci\_fetch\_all($stid, $res);  var\_dump($res);  echo "</pre>";  }  for ($i = 0; $i < 10; $i++) {  $a[] = 'value '.$i;  }  $c = oci\_connect("hr", "hrpwd", "//localhost/orcl");  $s = oci\_parse($c, "BEGIN mypkg.myproc(:c1); END;");  oci\_bind\_array\_by\_name($s, ":c1", $a, count($a), -1, SQLT\_CHR);  oci\_execute($s);  do\_query($c)  ?>  这将在 $a 中创建一个字符串数组。然后，将该数组绑定到 PL/SQL 过程的参数。 |
| **3.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/coll.php**  从 PTAB 表查询这些值，以验证它们已插入。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**9. 错误处理**

使用 Oracle 数据库错误处理功能时，PHP 函数 oci\_error() 很有用。

如果没找到错误，oci\_error() 连接错误会返回 FALSE 并且无需传入参数。如果发生连接错误，oci\_error() 将 Oracle 错误作为一个关联数组返回。这适用于所有连接函数 (oci\_connect()、oci\_pconnect() 和 oci\_new\_connect())。

处理分析错误或执行错误时，传入 oci\_error() 的资源句柄。

要实践一些简单的错误处理，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 查看 **$HOME/public\_html** 目录的 [**errors.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/errors.php) 文件中包含的以下代码。  <?php //Create connection to Oracle $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl"); if (!$conn) {  // No argument needed for connection errors.  // To generate an error here, change the connection parameters to be invalid.  $e = oci\_error();  print "There was a database connection error: " . htmlentities($e['message']);  exit; } // To generate an error here, change the \* to an another character, such as %. $query = "select \* from departments"; $stid = oci\_parse($conn, $query); if (!$stid) {  // For parsing errors, pass the connection resource  $e = oci\_error($conn);  print "There was a statement parsing error: " . htmlentities($e['message']);  exit; } $r = oci\_execute($stid); if (!$r) {  // For execution and fetching errors, pass the statement resource  // To generate an error here, change $query to be an invalid query.  $e = oci\_error($stid);  echo "<p>";  print "There was a statement execution error: <strong>" . htmlentities($e['message']).  "</strong><br>";  print "The error is located at character " . htmlentities($e['offset']+1) ."  of the query:   <strong>". htmlentities($e['sqltext']). "</strong><br>";  echo "</p>"; exit; } // Fetch the results in an associative array  print '<table border="1">'; while ($row = oci\_fetch\_array($stid, OCI\_RETURN\_NULLS+OCI\_ASSOC)) {  print '<tr>';  foreach ($row as $item) {  print '<td>'.($item?htmlentities($item):'&nbsp;').'</td>';  }  print '</tr>'; } print '</table>';  // Close the Oracle connection oci\_close($conn);  ?> |
| **2.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/errors.php** |
| **3.** | 要生成一个连接错误，编辑 errors.php 将登录信息更改为一个将发生故障的连接字符串（例如，对 HR 用户无效的口令）。  **$conn = oci\_connect("hr", "hrxx", "//localhost/orcl");** |
| **4.** | 重新加载以下 URL：  **http://localhost/~phplab/errors.php**  连接错误处理代码会捕获连接错误，并在输出中显示错误。  注：第一个错误是由 PHP 生成的错误，可以通过关闭 php.ini 配置文件中的错误报告来禁止。 |
| **5.** | 编辑 **errors.php**，将登录信息更改为原始登录，这样登录和连接才能成功。 |
| **6.** | 要生成一个分析错误，将 $query 变量编辑成一个无效的查询结构，例如：  **$query = "select ' from departments";** |
| **7.** | 在您的浏览器中，重新加载以下 URL：  **http://localhost/~phplab/errors.php**  分析错误处理代码会捕获分析错误，并在输出中显示错误。 |
| **8.** | 要生成一个获取错误，将 $query 变量编辑成一个无效的查询，例如：  $query = "select \* from **sometable"**; |
| **9.** | 重新运行以下 URL：  **http://localhost/~phplab/errors.php**  获取错误处理代码会捕获获取错误，并在输出中显示错误。  offset 参数包含分析错误开始的字符位置，sqltext 参数包含导致该分析错误的 SQL 语句。 |
| **10.** | *@* 函数前缀可以禁止所有 PHP 错误。这与将 php.ini 文件设置为不显示错误一样，但仅与您在其上使用该前缀的函数相关。使用 @ 前缀可以消除前面的错误处理示例中显示的 PHP 错误。要演示此功能，将 oci\_execute() 更改为：  $r=@oci\_execute($stid); |
| **11.** | 重新运行以下 URL：  **http://localhost/~phplab/errors.php**  PHP 错误已禁止，但脚本中的错误处理代码仍然显示 Oracle 错误。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**10. 使用 LOB：上载和查询图像**

Oracle 字符大对象 (CLOB) 和二进制大对象 (BLOB) 列（以及 PL/SQL 变量）可以包含大量的数据。创建这些对象以优化 Oracle 存储的方法有多种。此外，还预先提供了一个程序包 DBMS\_LOB，通过它可以轻松地在 PL/SQL 中操作这些对象。

要创建一个小型应用程序以将图像加载并显示到数据库，执行以下步骤。

|  |  |
| --- | --- |
| **1.** | 在执行本部分之前，先创建一个表来存储 BLOB。在 SQL\*Plus 中，以 HR 用户身份登录，执行以下命令：  **create table btab (blobid number, blobdata blob);** |
| **2.** | 查看 **$HOME/public\_html** 目录的 [**blobins.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/blobins.php) 文件中包含的以下代码。  <?php $myblobid = 1; // should really be a unique id e.g. a sequence number if (!isset($\_FILES['lob\_upload'])) { ?> <form action="<?php echo $\_SERVER['PHP\_SELF']; ?>"  method="POST" enctype="multipart/form-data"> Image filename: <input type="file" name="lob\_upload"> <input type="submit" value="Upload"> </form> <?php } else {  $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl");  // Delete any existing BLOB  $query = 'DELETE FROM BTAB WHERE BLOBID = :MYBLOBID';  $stmt = oci\_parse ($conn, $query);  oci\_bind\_by\_name($stmt, ':MYBLOBID', $myblobid);  $e = oci\_execute($stmt, OCI\_COMMIT\_ON\_SUCCESS);  if (!$e) {  die;  }  else {  $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl");  // Delete any existing BLOB  $query = 'DELETE FROM BTAB WHERE BLOBID = :MYBLOBID';  $stmt = oci\_parse ($conn, $query);  oci\_bind\_by\_name($stmt, ':MYBLOBID', $myblobid);  $e = oci\_execute($stmt, OCI\_COMMIT\_ON\_SUCCESS);  if (!$e) {  die;  }  oci\_free\_statement($stmt);  // Insert the BLOB from PHP's temporary upload area  $lob = oci\_new\_descriptor($conn, OCI\_D\_LOB);  $stmt = oci\_parse($conn, 'INSERT INTO BTAB (BLOBID, BLOBDATA) '  .'VALUES(:MYBLOBID, EMPTY\_BLOB()) RETURNING BLOBDATA INTO :BLOBDATA');  oci\_bind\_by\_name($stmt, ':MYBLOBID', $myblobid);  oci\_bind\_by\_name($stmt, ':BLOBDATA', $lob, -1, OCI\_B\_BLOB);  oci\_execute($stmt, OCI\_DEFAULT);  if ($lob->savefile($\_FILES['lob\_upload']['tmp\_name'])) {  oci\_commit($conn);  echo "BLOB uploaded";  }  else {  echo "Couldn't upload BLOB\n";  }  $lob->free();  oci\_free\_statement($stmt); } } ?> |
| **3.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/blobins.php**  它显示一个包括 Browse 和 Upload 按钮的 Web 表单。单击 **Browse**。 |
| **4.** | 从 /home/oracle/public\_html 目录中选择 **oracle.jpg**，然后单击 **Open**。 |
| **5.** | 单击 **Upload**。  form action 再次调用该脚本，但现在设置特殊的变量 $\_FILES['lob\_upload'] 并执行echo 语句。  图像已上载到 Web 服务器。 |
| **6.** | 要显示图像，查看 **$HOME/public\_html** 目录的 [**blobview.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/blobview.php) 文件中包含的以下代码。  <?php $myblobid = 1; $conn = oci\_connect("hr", "hrpwd", "//localhost/orcl"); // Now query the uploaded BLOB and display it $query = 'SELECT BLOBDATA FROM BTAB WHERE BLOBID = :MYBLOBID'; $stmt = oci\_parse ($conn, $query); oci\_bind\_by\_name($stmt, ':MYBLOBID', $myblobid); oci\_execute($stmt); $arr = oci\_fetch\_assoc($stmt); $result = $arr['BLOBDATA']->load(); header("Content-type: image/JPEG"); echo $result; oci\_free\_statement($stmt); oci\_close($conn); ?> |
| **7.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/blobview.php**  确保脚本中没有 echo 语句且“<?php”前面没有任何空格，否则将发送错误的 HTTP 头，浏览器无法正确显示图像。如果有问题，将 header() 函数调用注释掉，看看会显示什么。 |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**11. 使用 XML**

PHP5 提供了出色的 XML 功能。本教程包括将 XML 数据从 Oracle 返回至 PHP 的基本操作。

|  |  |
| --- | --- |
| **1.** | 您可以将相关行作为 XML 获取。在本例中，您将使用 SQL XMLELEMENT 函数检索 Employees 表的姓名和 ID，其中 employee\_id < 115。查看 **$HOME/public\_html** 目录的 [**xml1.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/xml1.php) 文件中的代码。 |
| **2.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/xml1.php** |
| **3.** | 另一种创建 XML 表单关系数据的方法是使用 PL/SQL 程序包 DBMS\_XMLGEN()（返回一个 CLOB）。[**xml2.php**](http://www.oracle.com/technology/obe/11gr1_db/appdev/php/files/xml2.php) 文件中的代码将完成以下操作：  a) 检索部门内 ID 为 30 的员工名字，并将 XML 标记的输出存储在 $mylob 中  $q = "select dbms\_xmlgen.getxml('  select first\_name  from employees where department\_id= 30') xml from dual";  $s = oci\_parse($c, $q); oci\_execute($s); $res = oci\_fetch\_row($s); $mylob = $res[0]->load(); // treat as LOB descriptor  b) 显示 $mylob 的内容  echo htmlentities($mylob);  c) 使用 PHP 的 SmpleXML 函数将 CLOB 转换成一个 XML 数组。  $xml = (array) simplexml\_load\_string($mylob); |
| **4.** | 在 Web 浏览器中，输入以下 URL 以显示输出：  **http://localhost/~phplab/xml2.php** |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**总结**

在本教程中，您学习了如何：

|  |  |
| --- | --- |
|  | 创建连接 |
|  | 创建简单查询 |
|  | 创建持久连接 |
|  | 创建事务 |
|  | 获取数据函数 |
|  | 优化数据预取 |
|  | 使用绑定变量 |
|  | 使用 PL/SQL |
|  | 使用集合 |
|  | 实现错误处理 |
|  | 上载和查询图像 |
|  | 使用 XML |

[返回主题列表](http://www.oracle.com/technetwork/cn/tutorials/228991-zhs.htm#t)

**附录：PHP 入门**

PHP 是一种动态类型的脚本语言。它在 Web 应用程序中很常见，但可用于运行命令行脚本。基本的 PHP 语法简单易学。它具有熟悉的循环、测试和赋值结构。每行以分号结束。

字符串可以包含在单引号或双引号中：

'A string constant'

"another constant"

变量名以美元符号为前缀。类似双引号字符串内的变量将扩展为：

"A value appears here: $v1"

也可使用句点将字符串和变量连接在一起。

'Employee ' . $ename . ' is in department ' . $dept

变量不需要声明类型：

$count = 1;

$ename = 'Arnie';

数组可以具有数字索引或关联索引：

$a1[1] = 3.1415;

$a2['PI'] = 3.1415;

可以用 echo 或 print 语句显示字符串和变量。使用 printf() 还可以实现格式化输出。

echo 'Hello, World!';

echo $v, $x;

print 'Hello, World!';

printf("There is %d %s", $v1, $v2);

var\_dump() 函数对于调试很有帮助。

var\_dump($a2);

假定上面指定的值为 $a2，输出如下所示：

array(1) {

["PI"]=>

float(3.1415)

}

可以通过测试和循环来控制代码流。PHP 还具有一个 switch 语句。if/elseif/else 语句如下所示：

if ($sal > 900000) {

echo 'Salary is way too big';

}

elseif ($sal > 500000) {

echo 'Salary is huge';

}

else {

echo 'Salary might be OK';

}

这还会显示代码块是如何包含在括号中。

传统的循环为：

for ($i = 0; $i < 10; $i++) {

echo $i;

}

这将输出数字 0 到 9。$i 的值在每次迭代后递增。当测试条件值为 false 时，循环停止。您也可以使用 while 或 do while 结构进行循环。

foreach 命令对于数组迭代很有帮助：

$a3 = array('Aa', 'Bb', 'Cc');

foreach ($a3 as $v) {

echo $v;

}

这会依次将 $v 设置为数组中的每个元素。

可能会定义如下所示的函数：

function myfunc($p1, $p2) {

echo $p1, $p2;

return $p1 + $p2;

}

函数可能具有可变数量的参数，可能返回值，也可能不返回值。可以使用以下代码调用该函数：

$v3 = myfunc(1, 3);

函数调用可能会出现在函数定义之前。

可以使用 include() 或 require() 语句将子文件包括在 PHP 脚本中。

include("foo.php");

require("bar.php");

如果没找到该脚本，require() 将生成严重错误。

注释要么是一行：

// a short comment

要么是多行：

/\*

A longer comment

\*/

PHP 脚本包含在 *<?php* 和 *?>* 标记中。

<?php

echo 'Hello, World!';

?>

当 Web 服务器配置为通过 PHP 解释程序运行 PHP 文件时，在浏览器中加载脚本将导致执行 PHP 代码，所有输出将传输到浏览器。

PHP 代码块和 HTML 代码块可能是交替的。PHP 代码还可以显式输出 HTML 标记。

<?php

require('foo.php');

echo '<h3>';

echo 'Full Results';

echo '</h3>';

$output = bar(123);

?>

<table border="1">

<tr>

<td>

<?php echo $output ?>

</td>

</tr>

</table>

PHP 的许多方面由 php.ini 配置文件中的设置控制。文件的位置取决于系统。使用 phpinfo() 函数，可以找到其位置、  
加载的扩展名列表以及所有初始化设置的值：

<?php

phpinfo();

?>

可以通过编辑 phpl.ini 或使用 Zend Core for Oracle 控制台然后重新启动 Web 服务器来更改值。通过使用 ini\_set() 函数，可以在脚本中改变某些值。

各种 oci\_xxx 函数的列表包括：

|  |  |
| --- | --- |
| oci\_fetch\_all | 将结果数据的所有行获取到一个数组中 |
| oci\_fetch\_array | 将结果数据中的下一行作为关联数组和/或数字数组返回 |
| oci\_fetch\_assoc | 将结果数据中的下一行作为关联数组返回 |
| oci\_fetch\_object | 将结果数据中的下一行作为对象返回 |
| oci\_fetch\_row | 将结果数据中的下一行作为数字数组返回 |