

INTRODUCCIÓN AL DESARROLLO DE APLICACIONES CON PL/SQL

TEMA 5

CONECTIVIDAD CON PL/SQL

ORACLE®

"Si trabajas duro y no tienes éxito, entonces debes
aumentar la dosis."

Julio Chai, empresario

Autor: Pablo Matías Garramone Ramírez ©

Tema 5 Conectividad con PL/SQL

RESUMEN

En este último tema vamos a explicar como podemos conectar PL/SQL con tres de las plataformas más comunes de desarrollo, pero vamos si se quiere extrapolar a cualquier otra la diferencia es mínima, de modo que si sabes realizarla con una tecnología, no es excesivamente difícil pasar a cualquier otra, con los matices y diferencias de cada una obviamente.

PABLO TÍO, QUIERO IR AL GRANO

Bueno en este caso vosotros elegís que tecnología queréis probar, he planteado 6 conexiones a nuestra aplicación, lo bonito sería que hicierais las 6, pero os voy a pedir un mínimo de 2 de la tecnología elegida, si queréis hacer más o luego hacer más de otras plataformas, perfecto, pero al menos un mínimo de 2 con una de ellas. Yo os voy a dar hecha una de las conexiones de cada plataforma, y además os dejaré una segunda prácticamente hecha de cada una también, de manera que si quieres ir al grano, completa esta que te dejo incompleta y haz una extra tú.

INDICE

1. INTRODUCCIÓN CONEXIÓN CON PL/SQL.....	3
2. PREPARACIÓN ENTORNO DE DESARROLLO.....	3
2.1 WAMP con PHP	3
2.2 XAMPP con PHP	7
2.3 VISUAL STUDIO. C#	8
2.4 NETBEANS. JAVA.....	9
3. ELECCIÓN LIBRERIA Y ESTRAGIA DE CONEXIÓN CON PL/SQL.....	10
3.1 Elección de Librería.....	10
3.2 Estrategia de Conexión.....	10
3.2.1 Devolución de valores escalares	11
3.2.2 Devolver una fila de datos.....	11
3.2.3 Devolución de múltiples datos	11
4. CONECTAR Y TRABAJAR APLICACIONES CON PL/SQL CON OCI	12
4.1 Ejemplos Conexión con cada Plataforma.....	12
4.1.1 FUNCIÓN DESPLEGABLE.....	12
4.1.2 PROCEDURE MOSTRAR JUGADORES	12
4.1.3 FUNCIÓN PHP	12
4.1.4 PROCEDURE PHP	14
4.1.5 FUNCIÓN C#	15
4.1.6 PROCEDURE C#	16
4.1.7 FUNCIÓN JAVA.....	17
4.1.8 PROCEDURE JAVA.....	18
4.2 Conexión a ORACLE.....	19
4.2.1 Conectar PHP.....	19
4.2.2 Conectar C#	19
4.2.3 Conectar JAVA.....	20
4.2.4 Desconectar PHP	20
4.2.5 Desconectar C#	21

4.2.6 Desconectar JAVA.....	21
4.3 Llamar a procedimientos almacenados	21
4.3.1 Llamar con PHP.....	21
4.3.2 Llamar con C#.....	23
4.3.3 Llamar con JAVA.....	25
4.4 Trabajar con los datos obtenidos.....	26
4.4.1 Trabajar con PHP.....	26
4.4.2 Trabajar con C#.....	26
4.4.3 Trabaja con JAVA	27

1. INTRODUCCIÓN CONEXIÓN CON PL/SQL

Como siempre les explico a mis pupilos, puedes ser un enorme programador de PL/SQL y hacer los mejores procedimientos almacenados, con su complejidad máxima y todo lo que quieras, pero si no hay nada por encima que lo utilice, el PL/SQL por si solo está limitado a procesos de la base de datos.

Para realmente esta pensado este lenguaje es para que haya una plataforma por encima que haga uso de él, y que haga de interfaz e interlocutor con el usuario a traves de botones, links, desplegables etc.

De manera que debemos tener nociones de algún lenguaje de programación capaz de desarrollar interfaces amigables y que pueda conectarse con nuestro procedimientos almacenados. De manera que un curso de PL/SQL no puede quedar completo, si no vemos como podemos llamar a esos desarrollos que tan difícil nos ha costado construir, aquí tenemos en este tema como vamos a hacerlo.

2. PREPARACIÓN ENTORNO DE DESARROLLO

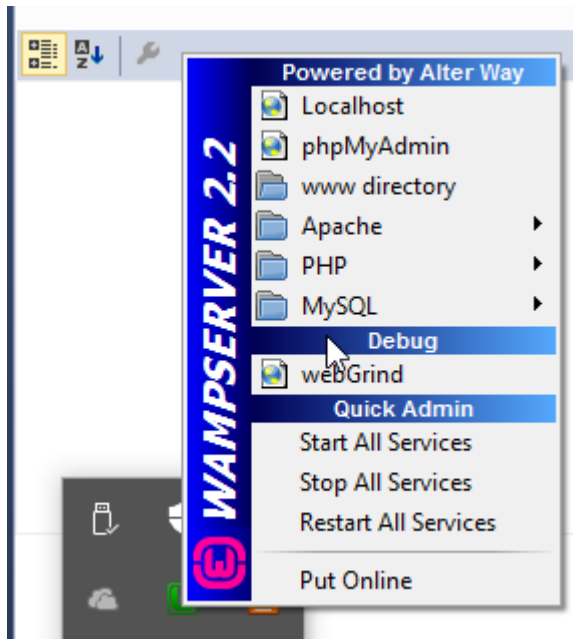
En el tema 1 ya vimos como instalar las 3 plataformas, de manera que si aún no lo habéis realizado, iros al tema 1, e instalar bien el WAMP/XAMPP/LAMP si vais a usar PHP, Visual Studio si vais a desarrollar en C# o netBeans si queréis finalizar este curso con JAVA.

Pero aunque ya los hayáis instalado, queda alguna cosita por hacer, yo como os voy a pasar ya la aplicación "semihecha" esto no es necesario que lo hagáis. No obstante si hubiera algún problema podéis mirar que esto este bien o volverlo hacer en vuestra máquina virtual.

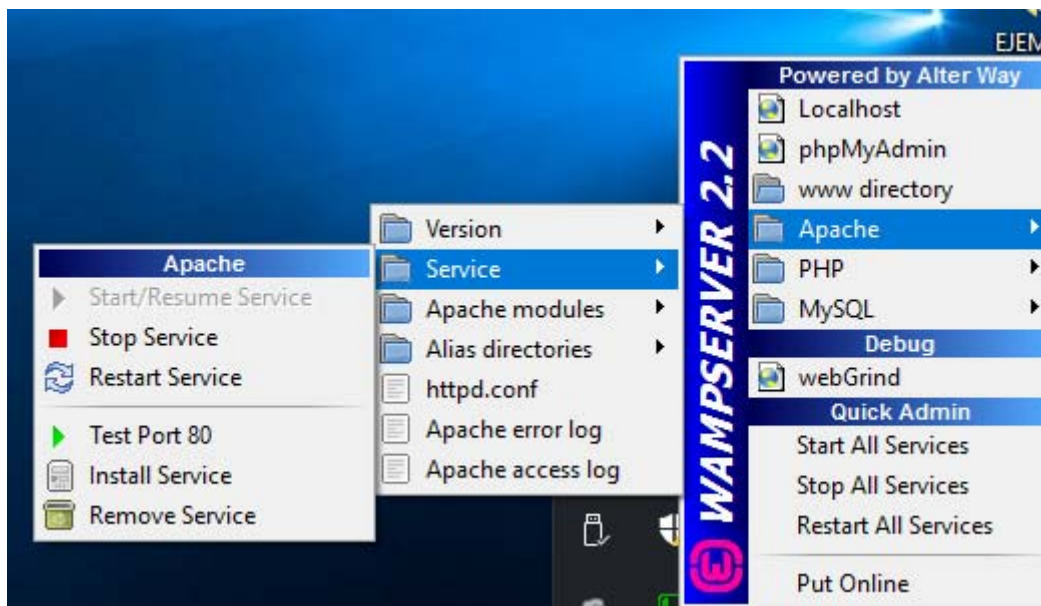
2.1 WAMP con PHP

Instalar el WAMP no tiene misterio y salvo el problema de pueda aparecer al darle a localhost, que ya os detalle como resolverlo en el tema 1, no debería tener ningún problema.

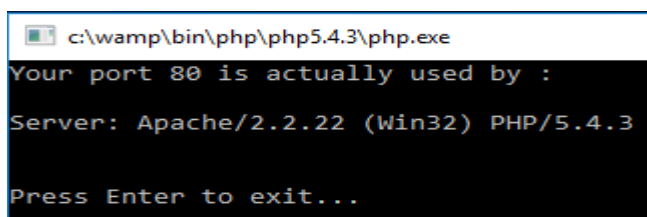
Debéis grabar todos los ficheros que os paso directamente en la carpeta c:/WAMP/WWW, que es la carpeta raíz que hace de servidor Apache. SE puede llegar directamente desde la W que os aparecerá al lado del reloj.



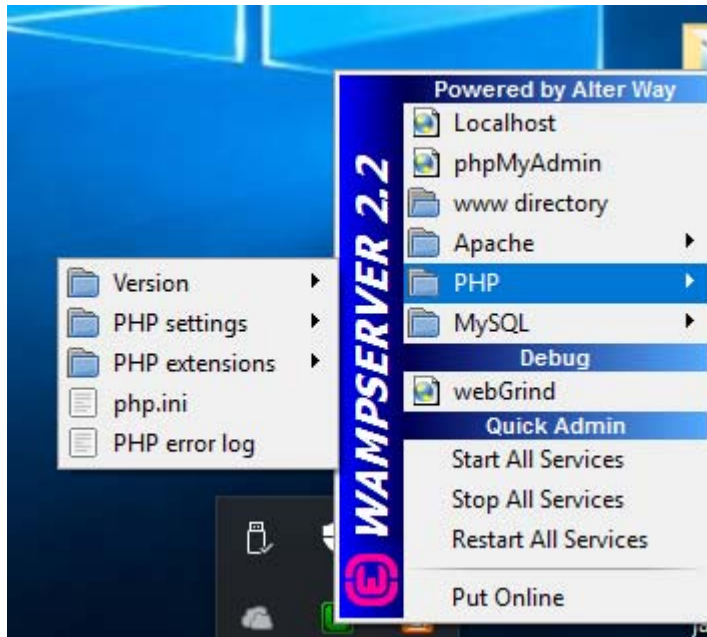
Lo primero que debemos asegurarnos es que la W está en color verde, es como un semáforo, primero está en Rojo (que no está conectado), Amarillo (En proceso de conexión o si permanece que hay algún warning) y verde (conectado). Si comprobamos que no se pone en verde, sería interesante comprobar si tenemos el puerto 80 ocupado, recordemos que por defecto el WAMP funciona en dicho puerto, para comprobarlo, nos vamos donde pone Apache y luego a Service y testear el puerto 80:



Y debemos estar usándolo nosotros con nuestro WAMP, en caso que haya otra cosa, ya sabemos que es lo que tenemos que desconectar:



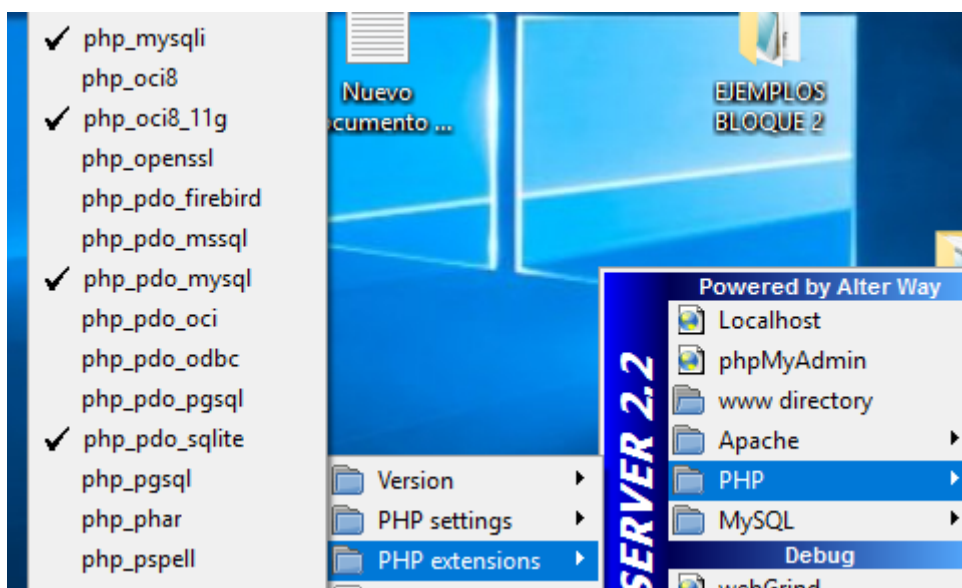
Una vez en verde, debemos configurar la conexión de PHP con ORACLE, para ello vamos a la carpetita que vemos que dice "PHP", y de ahí vamos a donde pone php.ini:



Y debemos tener descomentada la línea:


```
extension=php_mysql.dll  
;extension=php_oci8.dll  
extension=php_oci8_11g.dll  
;extension=php_oci8_12c.dll
```

Es interesante comprobar, que donde pone en este mismo sitio PHP extensions, está marcado el oci8_11g, si solo nos aparece oci8_12c, es que por defecto no está instalada en esa versión del WAMP el dll para la versión 11 y deberíamos instalarla, o buscar otra versión de WAMP que sí la traiga:



Finalmente, desde la misma podéis llegar a la carpeta www directory y también darle a Localhost para ver la aplicación desde el navegador web predeterminado.

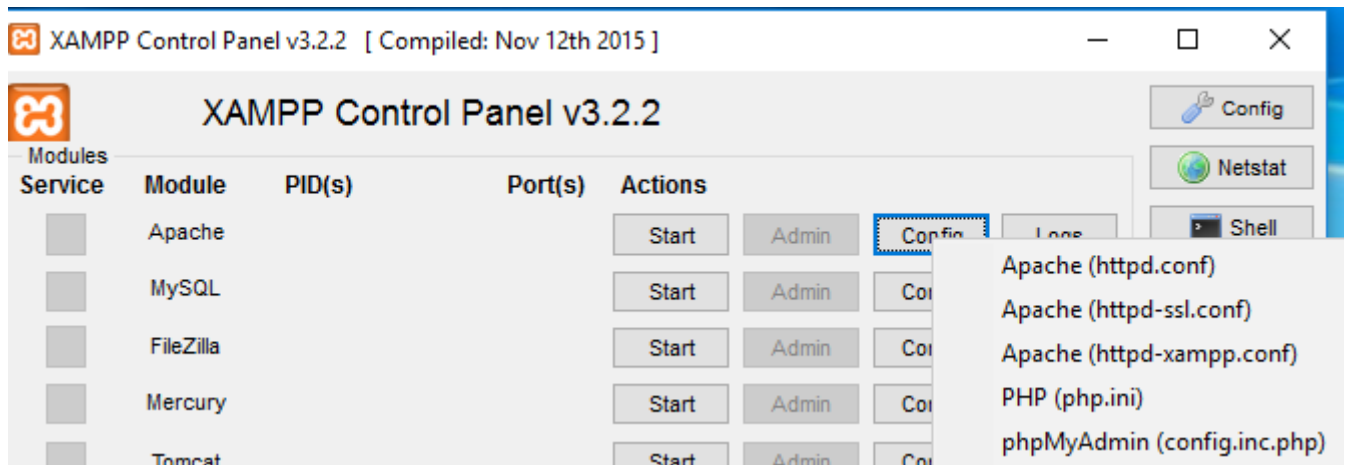
Una vez hayáis cargado todos los ficheros, si le dáis a localhost y váis al fichero principal.php os deberá salir la pantalla inicial de la aplicación si todo ha ido correctamente:

<p>5.1) Cargar Datos Mundiales</p> <p>Fichero a Cargar <input type="text"/> <input type="button" value="Cargar Datos"/></p>	<p>Ponente: PABLO GARRAMONE</p> <p>PROYECTO MUNDIAL</p> <p>PL/SQL</p> <p>ORACLE</p> 
<p>5.2) El Consultor</p> <p><input type="button" value="v"/> <input type="button" value="v"/> <input type="button" value="v"/> <input type="button" value="Consultar"/></p>	
<p>5.3) Gestionar Jugadores</p> <p>Selección: <input type="text"/> <input type="button" value="Gestionar Jugadores"/></p>	
<p>5.4) Buscar Partidos</p> <p>Año Mundial: <input type="text"/></p> <p>Equipo Local: <input type="text"/> <input type="button" value="Buscar Partido"/></p> <p>Equipo Visitante <input type="text"/></p> <p>Estadio <input type="text"/></p>	
<p>5.5) Clasificación Mundial</p> <p>Mundial (Déjalo en blanco si quieres todos): <input type="text"/> <input type="button" value="Clasificación"/></p>	

Si al darle al localhost presenta algún problema, pero poniendo 127.0.0.1 funciona, en el tema 1 os detallé como resolverlo.

2.2 XAMPP con PHP

Si os decantáis por el XAMPP es prácticamente lo mismo, una vez instalado os saldrá el Panel de Control de XAMPP, en él en la parte de Apache, debéis pulsar sobre config:

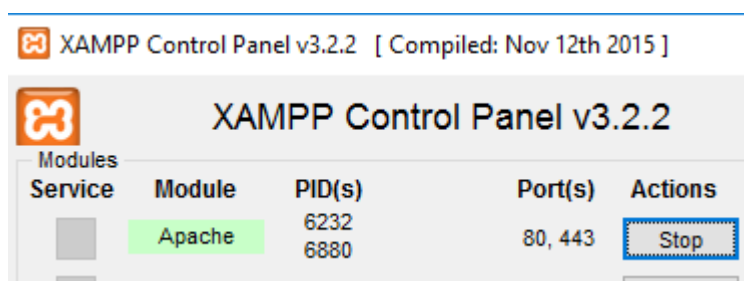


Y dentro del desplegable que os sale pulsar sobre PHP (php.ini) y hacer exactamente la misma operación que en el caso anterior:

```
extension=php_mbstring.dll
extension=php_exif.dll ; Must be after mbstring as it depends on it
extension=php_mysql.dll
extension=php_mysqli.dll
extension=php_oci8.dll ; Use with Oracle 10gR2 Instant Client
extension=php_oci8_11g.dll ; Use with Oracle 11gR2 Instant Client
:extension=php_openssl.dll
```

En el XAMPP la única cuestión que debemos tener en cuenta con diferencia al WAMP, es que la carpeta del servidor donde se deben alojar los ficheros es C:\xampp\htdocs\.

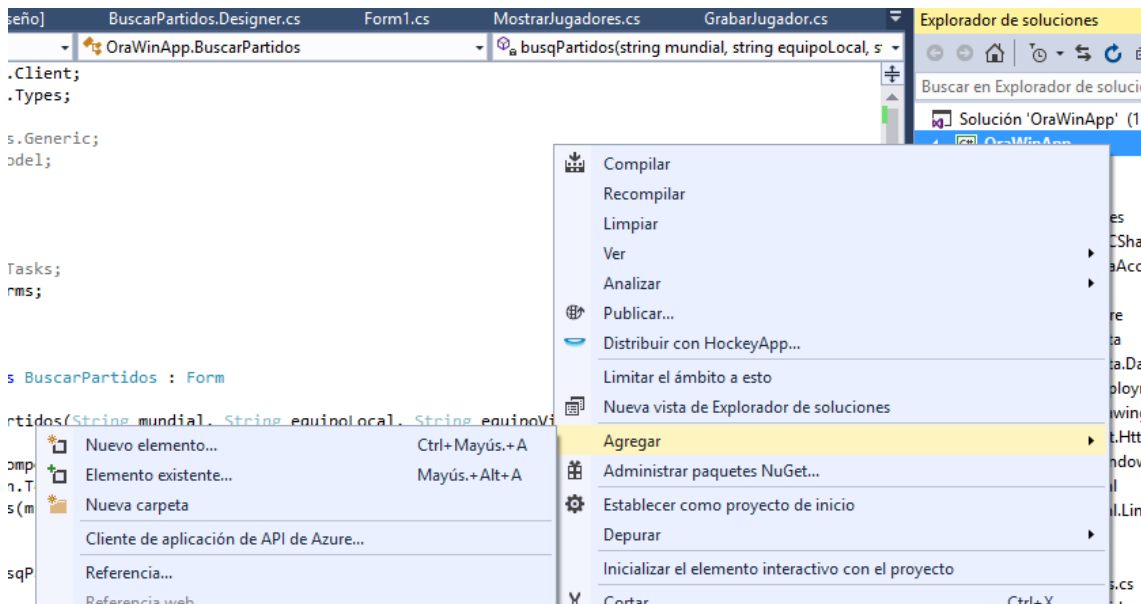
Finalmente nos queda arrancar el Apache, para ellos debemos pulsar Start en el Apache, y al cabo de unos segundos, se pone en color amarillo, y finalmente si no hay problemas se pondrá en verde y nos marcará los puertos a los que se conecta:



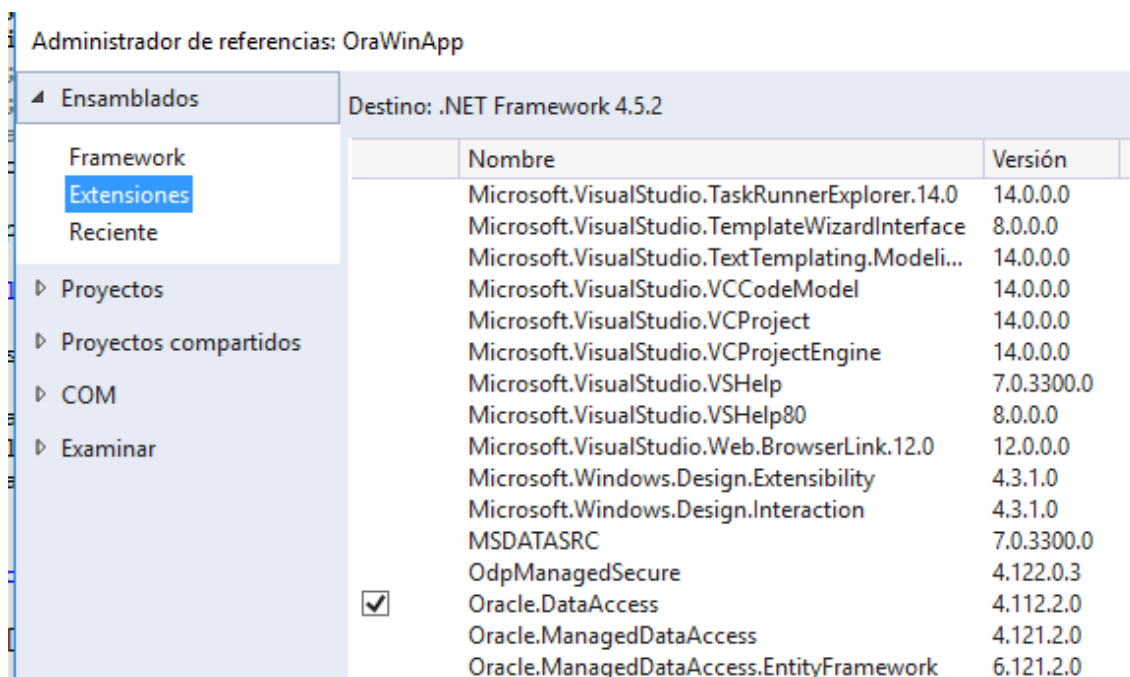
Una vez tengamos todos los pasos hechos, si nos vamos a localhost y la carpeta de la aplicación nos debe arrancar sin problemas.

2.3 VISUAL STUDIO. C#

Una vez tengáis instalado el Visual Studio para poder trabajar con ORACLE, se debe añadir las referencias de ORACLE para poder usar las librerías de conexión y gestión de la Base de Datos Oracle. Para ello una vez tengamos creado el proyecto sobre el que vamos a trabajar se debe añadir dicha referencia. Como ya os he dicho esta referencia ya está añadida en el proyecto que os paso, luego no deberías tener que hacer nada, pero si hacéis uno nuevo algún día, os explico, pulsáis botón derecho en vuestro proyecto y añadís una nueva referencia:



Y en la referencia hay que añadir el que os dejo aquí marcado Oracle.DataAccess:

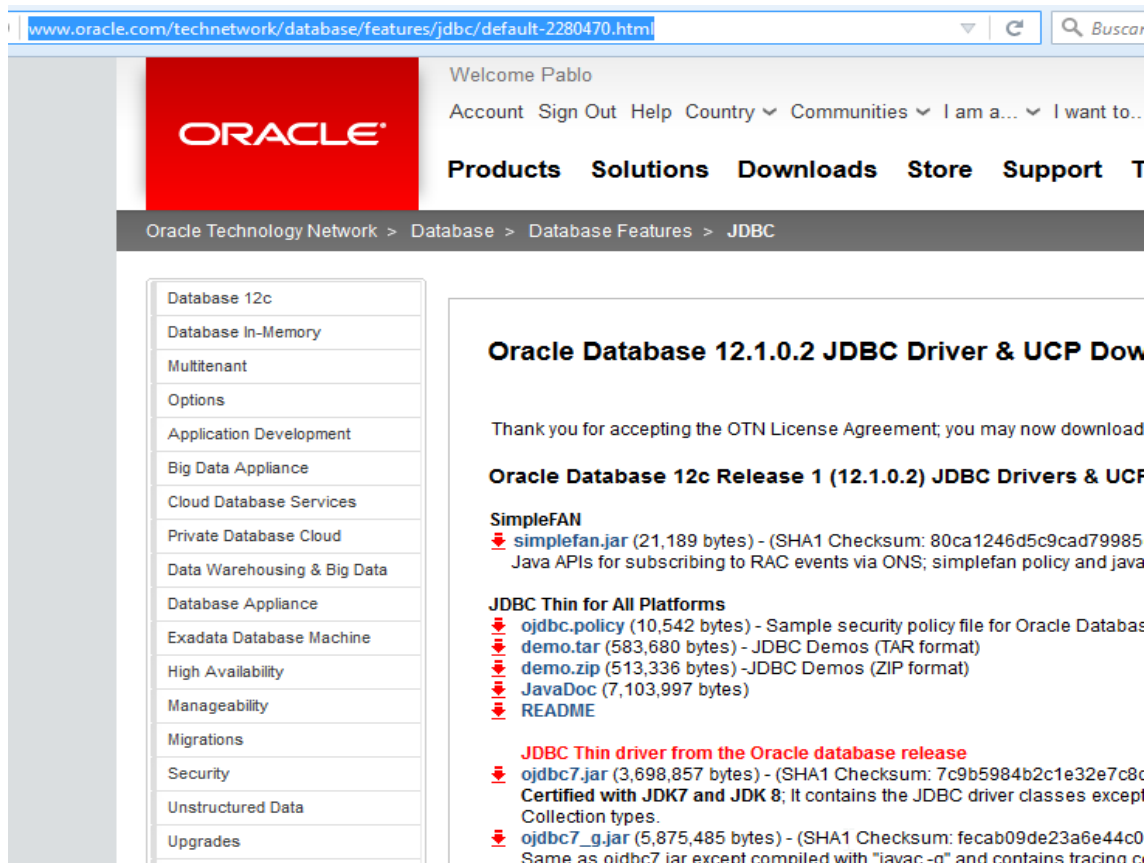


En cualquier caso aquí os dejo una dirección de referencia para más detalles:
<http://www.oracle.com/technetwork/articles/cook-vs08-088541.html>

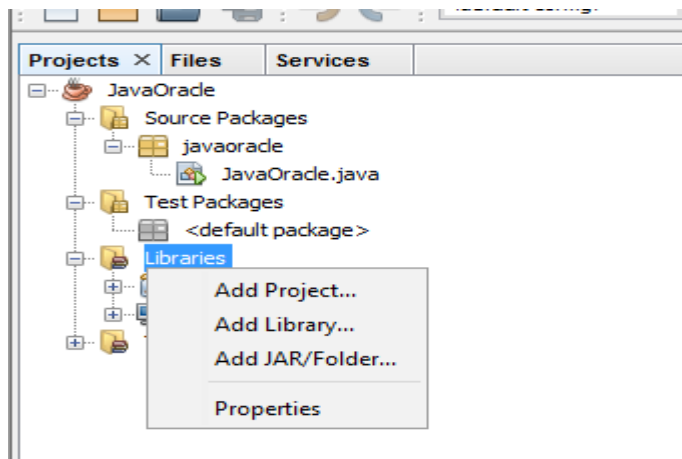
2.4 NETBEANS. JAVA

Para poder usar ORACLE con JAVA, concretamente la librería JDBC para ORACLE, en este caso yo he instalado la versión 7, en teoría no debéis hacer nada, pero si hacéis un nuevo proyecto os comento como hacerlo.

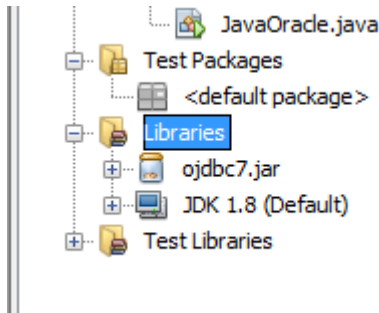
Primero se debe bajar el fichero de la web de ORACLE:



Posteriormente os vais a vuestro proyecto en netBeans, y añadís esta librería, del siguiente modo, os váis a vuestro proyecto a la carpeta librerías:



Y añadís el .JAR del JDBC:



3. ELECCIÓN LIBRERÍA Y ESTRAGIA DE CONEXIÓN CON PL/SQL

3.1 Elección de Librería

Disponemos de varias librerías para poder conectarnos con ORACLE, la elección de la librería puede ser importante, y va depende mucho de la Base de datos con la que trabajemos y las posibilidades de que en un futuro se modifiquen o no.

Si trabajamos con ORACLE es complicado que en un futuro se modifique la idea de negocio tecnológico de la empresa y se decida cambiar a otro sistema gestor y por lo tanto migrar todo a otra base de datos. Pero nada es imposible.

Si buscamos una librería que esté asociada a ORACLE y se adapte a la perfección al sistema gestor, sus tipos y sus características es evidente que debemos OCI, que tiene sus versiones para cada lenguaje, la librería OCI8 para PHP, su proveedor ODAC (Oracle Data Provider) para C# o la versión OJDBC para Oracle de JAVA, sin embargo si no estamos seguro de que sistema gestor vamos a usar, y bien ahora es Oracle pero en el futuro tiene muchas opciones de modificarse, podríamos pensar en un ADODB o ODBC (si no nos importa estar atados a Microsoft), que son librerías independientes del sistema gestor, con las que he trabajado y van muy bien.

En este curso, por cuestión de tiempo, y de practicidad, y porque es la específica para ORACLE, vamos a dar la versión OCI exclusiva de Oracle que es la más común para conectarnos a este sistema gestor.

3.2 Estrategia de Conexión

Una vez estemos conectados a ORACLE con la libreria OCI, tenemos que decidir como vas a transportar los datos de la ORACLE a la aplicación, y aquí es un mundo, no hay una forma concreta y todo depende del gusto, necesidades y habilidades del diseñador de la arquitectura.

La idea que supongo que cualquier tiene apriori de la estregia de traer datos del Sistema gestor es através de una select sobre la tabla en cuestión y esos datos ponerlos en un record set, pasarlo a un array de la aplicación y recorrerlo. Pero cuando no trabajamos solo con tablas, si no también

tenemos procedimientos almacenados y lo que devuelven estos no necesariamente deben ser de una tabla concreta de la base de datos, puede ser de múltiples de ellas e incluso muchas veces son datos calculados que no están almacenados en ningún sitio por lo tanto no existe una tabla concreta que disponga de esos datos.

Particularmente yo he ideado una forma concreta para realizar esta transmisión de datos, pero debe quedar claro que no es "la forma", es solo una forma más que he decidido, que puede haber otras muchas otras.

3.2.1 Devolución de valores escalares

Tendremos la opción de devolver valores escalares, es decir, una función de nuestros PACKAGE que devuelvan un solo valor, bien un número, un alfanumérico o una fecha simplemente, esto no conlleva problema ninguno, se almacenará en un tipo equivalente en la aplicación y no tiene más cuestión.

3.2.2 Devolver una fila de datos

Puede ser que tengamos que devolver varios campos de una misma cosa, como puede ser los datos de un jugador para modificarlos, en este caso, he decidido que usaremos un ARRAY, ya que las tres plataformas disponen de un tipo de dato equivalente al ARRAY de PL/SQL y podemos hacer la conversión directa.

3.2.3 Devolución de múltiples datos

Aquí es la cuestión más habitual, y la que vamos a utilizar, en este caso vamos a utilizar como medio de transporte el REF CURSOR, que son cursores (pseudo tablas) que almacenan todas las filas que se le meta. Como ya he mencionado en anteriores temas, utilizaremos en la parte de PL/SQL dos diferentes cursores, uno que tiene una tabla por detrás y otro que tenga lo que le echen, es decir, que no está vinculado a ninguna tabla. En la parte de la aplicación no tiene diferencia ambos serán un REF CURSOR.

La ventaja del REF CURSOR es su versatilidad, ya que no necesariamente se forma con una tabla concreta, se contruye con una select, y una select puede abarcar diferentes tablas. Pero claro el problema que muchas veces tenemos, es que no siempre podremos realizar una operación en PL/SQL que se realice con una simple select (por más compleja que esta sea) y que requiera de ir realizando cálculos en varias fases, y debemos ir almacenando cada parte para al final devolverlo todo, ¿como hacer esto?, aquí es donde se me ocurrió utilizar las tablas TEMPORARY TABLES, que son tablas que desaparecen los datos por cada sesión, de modo que podemos usarla durante nuestra sesión ir metiendo en ella todas las cosas que vayamos haciendo, y al final crear un REF CURSOR que devuelva dicha tabla a la aplicación. Es por ello que durante el curso os he pedido que metáis los resultados en nuestra famosa tabla TMP_ESTRUCTURA.

De modo, que cuando podamos realizar las cosas con una select, meteremos el REF CURSOR con la tabla concreta, y si esto no es posible, porque necesitamos varias fases en el proceso de nuestro PL/SQL entonces usaremos la TEMPORARY TABLE.

4. CONECTAR Y TRABAJAR APLICACIONES CON PL/SQL CON OCI

4.1 Ejemplos Conexión con cada Plataforma

Voy a ponerlos dos ejemplos que os doy resuelto de cada plataforma, y os lo detallaré, luego los explicaré por partes.

4.1.1 FUNCIÓN DESPLEGABLE

En este caso vamos a mostrar como se llama a una función en ORACLE, esta función recibirá 3 varchar2 y 1 number, es decir, el nombre de una tabla, porque columna debe ordenar y opcionalmente recibirá un where para filtrar, que serán los 2 varchar2 por defecto. Y devolverá un REF CURSOR con todas las filas que devuelva la select con lo que se llenará el desplegable con el que se llamó a esta función:

```
FUNCTION LLenar_Desplegables(tabla varchar2, posicionCampo number,
colWhere varchar2 default null, valWhere varchar2 default null)
return SYS_REFCURSOR
IS
    desplegable SYS_REFCURSOR;
    cadenaSql varchar2(32000);
BEGIN
    if colWhere is null then
        cadenaSql:='select * from '||tabla||' order by '||posicionCampo;
    else
        cadenaSql:='select * from '||tabla||
        ' where '|| colWhere ||' = '||valWhere
        ||'' order by '||posicionCampo;
    end if;
    OPEN desplegable FOR cadenaSql;

    return desplegable;
exception when others then
    raise_application_error(-20100,'Error inesperado: '||sqlerrm);
END LLenar_Desplegables;
```

4.1.2 PROCEDURE MOSTRAR JUGADORES

El procedure que os pondré como ejemplo, es el que dado el nombre de un equipo, debéis devolver todos los jugadores del mismo, este procedure os sirve como modelo para todos los demás.

El código lo tenéis detalladamente explicado en el video que os pasé de como preparar un procedure.

4.1.3 FUNCIÓN PHP

Una función es un procedimiento almacenado que devuelve datos, en este caso será un ref cursor:

```

Selección:
<select size="1" name="seleccion">
1)   <option value=""></option>
    <?php
2)   $conexion = oci_connect('MUNDIAL', '1234', 'localhost/orcl');
        if (!$conexion) {
            $e = oci_error();
            trigger_error(htmlentities($e['message']), E_USER_ERROR);
        }
3)   $arrayEquipos = array();
        $indice=0;
        // AMBAS FUNCIONAN!!!
4)   // $stid = oci_parse($conexion, 'begin :refCurResult := prueba.LLenar_Desplegables(:tabla,
:posicion); end;');
        $stid = oci_parse($conexion, 'call prueba.LLenar_Desplegables(:tabla, :posicion) into
:refCurResult');
5)   $deplegableEquipos = oci_new_cursor($conexion);
        $tabla="EQUIPOS";
        $posicion=1;
6)   oci_bind_by_name($stid,":tabla",$tabla);
        oci_bind_by_name($stid,":posicion",$posicion);
7)   oci_bind_by_name($stid,":refCurResult",$deplegableEquipos,-1,OCI_B_CURSOR);
8)   oci_execute($stid);
        oci_execute($deplegableEquipos);
        if (!$stid)
            echo "error";
        else {
9)   while ($fila = oci_fetch_array($deplegableEquipos, OCI_ASSOC))
            {
                $arrayEquipos[$indice]=$fila['EQUIPO'];
                $indice=$indice+1;
            }
            <option value="<?php echo $fila['EQUIPO']?>"><?php echo $fila['EQUIPO']?></option>
10)  <?php }}      oci_free_statement($stid); oci_close($conexion);
        ?>
</select>

```

1) Se pone un option vacío para que por defecto aparezca una posición vacía.

2) Función que se conecta a ORACLE, se requiere el usuario de la BD, su contraseña y identificador de Oracle url/Sid de la BD. Luego haremos un condicional por si la conexión falla que nos diga qué ha pasado.

3) En este caso para no hacer varias llamadas a ORACLE para rellenar el mismo desplegable, aprovecho esta llamada para llenar un array y el resto de desplegables de EQUIPOS los lleno con el Array.

4) oci_parse es la función que prepara el procedimiento almacenado, se debe indicar el id conexión que devuelve oci_connect y luego la llamada al procedimiento almacenado con sus parámetros. En este caso como es una función debe devolver algo. Como veremos hay dos forma de llamarlo con BEGIN y END y lo llamamos como si fuera un bloque anónimo o si usamos CALL usamos INTO como si fuera un select into. Todos los parámetros sean IN, OUT o de devolución de una función se ponen con el carácter ":" y el nombre que quieras.

5) Como vamos a usar un REF CURSOR y es un parámetro especial, nos tenemos que crear una variable en PHP que va a asumir dicho REF CURSOR y para que sea compatible debemos usar la función oci_new_cursor.

6) oci_bind_by_name: Se trata de una función que "bindea" los parámetros y asocia el parámetro que hemos puesto con el carácter ":" en la llamada al procedimiento a una variable de PHP.

7) Como este parámetro es de devolución o tipo OUT requieren más parámetros, en este caso es para asociar el REF CURSOR a la variable PHP que hemos creado antes para el REF CURSOR.

8) oci_execute, ejecuta el procedimiento almacenado, una vez están listos los parámetros y el procedimiento se puede lanzar, fijaros que hay 2 oci_execute, el segundo sirve para rellenar la variable PHP con el REF CURSOR

9) oci_fech_array, es una función que convierte las filas que devuelve el REF CURSOR o una tabla en un array de PHP. Cada vez que se llama a esta función devuelve la siguiente fila, es como un FETCH de un cursor, por lo tanto lo ponemos en un bucle y volvamos el resultado en los <option> del desplegable.

10) Finalmente cerramos la conexión a la base de datos y liberamos la memoria.

4.1.4 PROCEDURE PHP

```
$conexion = oci_connect('MUNDIAL', '1234', 'localhost/orcl');
try
{
    $stid = oci_parse($conexion, 'call prueba.mostrarEquipo(:eq,:rsPart)');
    $partidos = oci_new_cursor($conexion);
    oci_bind_by_name($stid,":eq",$equipo);
    oci_bind_by_name($stid,":rsPart",$partidos,-1,OCI_B_CURSOR);
    oci_execute($stid);
    oci_execute($partidos);
    ?>
    <table border=1>
    <?php
    while ($fila = oci_fetch_array($partidos, OCI_ASSOC))
    {?>
        <tr><td>
            <?php echo $fila['N1']. ' - ' . $fila['C2']. ' - ' . $fila['C3']. ' - ' . $fila['C4'];?>
            <A href="grabarJugador.php?jugador=<?php echo $fila['C3'];?>&seleccion=<?php echo $equipo; ?>">
            <input type="button"></A>
        </td></tr>
    <?php
    }
    oci_free_statement($stid);
    oci_close($conexion);
} catch (exception $e) { print_r($e); }
```

No hay ninguna diferencia, salvo que el procedure no devuelve nada, pero tenemos el REF CURSOR de tipo out que es prácticamente lo mismo.

4.1.5 FUNCIÓN C#

```

public void llenarDesplegable(String tabla, ComboBox desplegable, Int32 posicionCampoMostrar, Int32
posicionCampoGrabar, String colWhere, String valWhere)
{
1)      libreriasOracle obconection = new libreriasOracle();
      OracleConnection conn = obconection.Conectar();
      OracleCommand objSelectCmd = new OracleCommand();
      OracleDataAdapter objAdapter = new OracleDataAdapter();
      DataTable dt = new DataTable();

2)      objSelectCmd.Connection = conn;
      objSelectCmd.CommandText = "prueba.LLenar_Desplegables";
      objSelectCmd.CommandType = CommandType.StoredProcedure;

3)      objSelectCmd.Parameters.Add("tabla", OracleDbType.Varchar2, 30).Value = tabla;
      objSelectCmd.Parameters.Add("posicionCampo", OracleDbType.Int32, 100).Value =
posicionCampoMostrar;
4)      if (colWhere!=null)
      {
          objSelectCmd.Parameters.Add("colWhere", OracleDbType.Varchar2, 100).Value = colWhere;
          objSelectCmd.Parameters.Add("valWhere", OracleDbType.Varchar2, 100).Value = valWhere;
      }
5)      objSelectCmd.Parameters.Add("estructura", OracleDbType.RefCursor).Direction =
ParameterDirection.ReturnValue;
6)      objSelectCmd.BindByName = true;

      objAdapter.SelectCommand = objSelectCmd;
7)      objAdapter.Fill(dt);

8)      DataRow row = dt.NewRow();
      row[posicionCampoGrabar - 1] = null;
      row[posicionCampoMostrar - 1] = null;
      dt.Rows.InsertAt(row, 0);

9)      desplegable.DataSource = dt;
      desplegable.DisplayMember = dt.Columns[posicionCampoMostrar-1].ToString();
      desplegable.ValueMember = dt.Columns[posicionCampoGrabar-1].ToString();
10)     conn.Close();
}

```

1) Llamamos a nuestra clase genérica libreriasOracle que contiene el método Conectar que es el que realiza la conexión a ORACLE, la explico luego. Seguidamente tenemos 3 variables que se necesitan para poder hacer la llamada al PL/SQL y recorrer sus datos. OracleCommand sirve para hacer la llamada al procedimiento almacenado y gestionar sus parámetros. objAdapter se usa para ejecutar el OracleCommand y rellenar un dataSet o un dataTable.

2) oracleCommand debemos rellenar sus variables Connection con la conexión, CommandText con el nombre del procedimiento almacenado que vamos a usar, y commmandType para decirle que es un procedimiento almacenado PL/SQL.

3) Ahora añadimos los parámetros uno a uno, en este caso vemos los de tipo IN, que hay que darles un nombre, tipo de dato y dimensión. Es importante que el nombre que ponemos como primer parámetro coincida con el que está en la llamada de PL/SQL, y para todos los que sean de tipo IN ponemos como value la variable de C# que queramos enviar al procedimiento.

4) Recordar que en este caso teníamos parámetros por defecto, por eso está este IF, si no vienen estos parámetros no enviamos nada así coge los NULL por defecto.

5) Este parámetro es el de devolución, por eso no ponemos el value, ponemos el DIRECTION indicándole que será un Return Value, este es el REF CURSOR.

6) Importante indicarle que "bindee" las variables.

7) oracleAdapter ejecuta el oracleCommand que acabamos de formar y con Fill, al ser un REF CURSOR lo que devuelve el procedimiento rellena el dataTable

8) Creamos la primera fila en blanco.

9) Llenamos el desplegable con el dataTable y le indicamos que campos son para que se muestre y para que sirva de valor a enviar.

10) Cerramos la conexión a la base de datos.

4.1.6 PROCEDURE C#

```

private void Obtener(String equipo)
{
    libreriasOracle obconexion = new libreriasOracle();
    OracleConnection conn = obconexion.Conectar();
    OracleDataAdapter objAdapter = new OracleDataAdapter();
    DataTable dt = new DataTable();
    OracleCommand objSelectCmd = new OracleCommand();
    objSelectCmd.Connection = conn;
    objSelectCmd.CommandText = "prueba.mostrarEquipo";
    objSelectCmd.CommandType = CommandType.StoredProcedure;
    objSelectCmd.Parameters.Add("p_equipo", OracleDbType.Varchar2, 20).Value = equipo;
1) objSelectCmd.Parameters.Add("estructura", OracleDbType.RefCursor).Direction =
    ParameterDirection.Output;
    objAdapter.SelectCommand = objSelectCmd;
    objAdapter.Fill(dt);

2) dataGridView1.DataSource = dt;
    dataGridView1.Columns["ORIGEN_DATOS"].Visible = false;
    dataGridView1.Columns["ID_TRANSACCION"].Visible = false;
    dataGridView1.Columns["C5"].Visible = false;
    dataGridView1.Columns["C6"].Visible = false;
    dataGridView1.Columns["C7"].Visible = false;
    dataGridView1.Columns["C1"].Visible = false;
    dataGridView1.Columns["C9"].Visible = false;
    dataGridView1.Columns["C10"].Visible = false;
    dataGridView1.Columns["N2"].Visible = false;
    dataGridView1.Columns["N3"].Visible = false;
    dataGridView1.Columns["N4"].Visible = false;
    dataGridView1.Columns["N5"].Visible = false;
    dataGridView1.Columns["N6"].Visible = false;
    dataGridView1.Columns["N7"].Visible = false;
    dataGridView1.Columns["N8"].Visible = false;
    dataGridView1.Columns["N9"].Visible = false;
    dataGridView1.Columns["N10"].Visible = false;

3) dataGridView1.Columns["N1"].HeaderText = "DORSAL";
    dataGridView1.Columns["C2"].HeaderText = "DEMARCACIÓN";
    dataGridView1.Columns["C3"].HeaderText = "NOMBRE";
    dataGridView1.Columns["C4"].HeaderText = "MUNDIALES JUGADOS";
    dataGridView1.Columns["C4"].Width = 200;
    conn.Close();
}

```

- 1) En este caso no es un valor de retorno, es de salir, por lo que se cambia el direction y se pone parameterDirection.out, pero tiene el mismo efecto.
- 2) Como vamos a meterlo en un dataGridView, metemos directamente el DataTable obtenido, y tenemos que ocultar todas las columnas que no vayamos a usar.
- 3) Y las que sí vamos a usar le cambiamos el nombre de la cabecera para que sean más entendibles.

4.1.7 FUNCIÓN JAVA

```

public void llenarDesplegable(String tabla, JComboBox desplegable, int posicionCampo, String
colWhere, String valWhere)
{
    try {
1)      Connection con = null;
        ResultSet rs = null;
        CallableStatement cs = null;

2)      JavaOracle obconexcion=new JavaOracle();
        con=obconexcion.Conectar();

3)      String sql;
        if(colWhere==null)
            sql="{?=call prueba.Llenar_Desplegables(?,?)}"
        else
            sql="{?=call prueba.Llenar_Desplegables(?,?,?,?)}"
        cs = con.prepareCall(sql);

4)      int pos = 0;
        // Cargamos los parametros de entrada IN
        cs.registerOutParameter(++pos, OracleTypes.CURSOR);
        cs.setString(++pos, tabla);
        cs.setInt(++pos, posicionCampo);
        if(colWhere!=null)
        {
            cs.setString(++pos, colWhere);
            cs.setString(++pos, valWhere);
        }
        // Ejecutamos
5)      cs.execute();

6)      rs = (ResultSet) cs.getObject(1);    // Nuestro cursor, convertido en ResultSet
7)      desplegable.removeAllItems();
8)      desplegable.addItem(null);
9)      while(rs.next())
        desplegable.addItem(rs.getString(posicionCampo));
10)     SqlTools.close(rs, cs, con);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

- 1) Declaramos las variables que vamos a usar para llamar al procedimiento almacenado, la conexión, un ResultSet para recoger el resultado y CallableStatement que es el que recoge que procedimiento almacenado se llama y gestiona sus parámetros y "bindeo".
- 2) Llamamos a nuestra Clase genérica donde tenemos la conexión a ORACLE, luego veremos.
- 3) Montamos el procedimiento almacenado llamamos a prepareCall. Cada parámetro se pone un interrogante, y si es una función ponemos el interrogante asignándolo a la función.
- 4) Ponemos todos los parámetros, dentro de nuestro CallableStatement, y este tiene un método para cada tipo de dato si es de tipo IN, y si es de tipo OUT debemos poner registerOutParameter y poner el tipo de dato, en este caso un OracleTypes.CURSOR, si fuera un NUMBER usaríamos OracleTypes.DECIMAL.
- 5) Para ejecutar llamamos siempre a execute()
- 6) Con el CallableStatement podemos ir a cualquier parámetro y con el método get de su tipo de dato y su posición en los parámetros, en este caso es un REF CURSOR que JAVA lo trata como un Object y como es el que se devuelve está en la posición 1, por lo tanto, cs.getObject(1) y le pasamos un casting de ResultSet y lo metemos en la variable.
- 7) Vaciamos el desplegable antes de rellenarlo.
- 8) Ponemos la primera línea vacía.
- 9) Recorremos el resultSet al ser varias filas, en un bucle, con next, y en este caso, vamos añadiendo al desplegable cada resultado.
- 10) Cerramos la conexión a la BD y liberamos toda la memoria. SqlTools son funciones para ello que os he hecho.

4.1.8 PROCEDURE JAVA

```

private void MostrarJugadores(String equipo)
{
    Connection con = null;
    ResultSet rs = null;
    CallableStatement cs = null;
    try
    {
        JavaOracle obconexion=new JavaOracle();
        con=obconexion.Conectar();
        String sql="{call prueba.MOSTRAREQUIPO (?,?)}";
        cs = con.prepareCall(sql);

        int pos = 0;
        // Cargamos los parametros de entrada IN
        cs.setString(++pos, equipo);
        // Registramos los parametro de salida OUT
        cs.registerOutParameter(++pos, OracleTypes.CURSOR);
        // Ejecutamos
        cs.execute();
        // Cosechamos los parametros de salida OUT

1)  rs = (ResultSet) cs.getObject(2);  // Nuestro cursor, convertido en ResultSet
2)  String Titulos[]={ " DORSAL ", " PUESTO ", " NOMBRE ", " MUNDIALES JUGADOS ", " CANTIDAD " };
    String fila[]=new String[5];
    jLabel13.setText("JUGADORES DE "+equipo);
3)  DefaultTableModel modelo = new DefaultTableModel(null, Titulos);
4)  while (rs.next())
    {
        fila[0] = rs.getString("N1");
        fila[1] = rs.getString("C2");
        fila[2] = rs.getString("C3");
        fila[3] = rs.getString("C4");
        fila[4] = rs.getString("N2");
        modelo.addRow(fila);
    }
5)  jTable1.setModel(modelo);

    }
    catch (SQLException ge)
    {
        JOptionPane.showMessageDialog(null, ge.getMessage());
    }
    finally {
        SqlTools.close(rs, cs, con);
    }
}

```

1) En este caso no es de devolución si no com o tipo OUT, pero como podemos observar se resuelve exactamente igual, en este caso está en la posición 2 de los parámetros.

2) Para hace un jTable de JAVA (el equivalente de dataGridView de C#) usaremos 2 array uno para los títulos y otro para los campos. Y los rellenamos.

3) Creamos un objeto TableModel que es el que jTable y le asignamos la fila del título.

4) Los valores los rellenamos con el ResultSet (el REF CURSOR). Usando next y getString, y lo vamos añadiendo fila a fila al TableModel.

5) Finalmente llenamos el jTable con el objeto Tablemodel.

Ahora vamos a ir explicando cada una de las funciones de la librería con un ejemplo.

4.2 Conexión a ORACLE

Desde luego lo primero que debemos hacer es conseguir conectarnos a ORACLE, parece lo más obvio, pero no siempre lo más sencillo, en el momento que tengamos esto ya estamos cerca de conseguir traer los datos.

Cualquiera de las plataformas dispone de una función para conexión que básicamente se basa en darle los datos de la conexión y que nos devuelva un identificador de sesión para que podamos trabajar con el resto de funciones, evidentemente sin este identificador no se puede hacer nada en la Base de Datos.

4.2.1 Conectar PHP

Para conectar desde PHP es muy sencillo, con una línea sobra, y es conveniente dejar la conexión siempre en un fichero a parte para no tener que modificar en todos los sitios cada vez que sea necesario hacer modificar alguna cosa, también decir que una empresa real, los datos de contraseña y usuario irían encriptados y no se dejarían en texto plano.

La función es OCI_CONNECT(<base de datos>, <contraseña>, <usuario>) y devuelve un entero que es el id de conexión.

Ejemplo:

```
$conexion = oci_connect('MUNDIAL', '1234', 'localhost/orcl')
```

4.2.2 Conectar C#

En este caso C# trabaja con objetos y deberemos crear un objeto OracleConnection que recibirá como parámetro en su constructor la cadena de conexión, con la particularidad que debe ser igual que la que tenemos en la conexión del tnames de nuestra conexión de ORACLE.

Como siempre es conveniente tenerlo en una función a parte, en el ejemplo de nuestra aplicación la tenéis en la clase libreriasOracle.cs en su método Conectar, y devolverá un objeto de conexión:

```
public OracleConnection Conectar()
{
    string oradb = "Data Source=(DESCRIPTION=(ADDRESS_LIST="
+ "(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521)))"
+ "(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCL)))";
+ "User Id=MUNDIAL;Password=1234;pooling=false";

    conexion = new OracleConnection(oradb); // C#

    conexion.Open(); // C#

    return conexion;
}
```

Fijaros que en el caso de C# añadido además del Data Source (la base de datos), el User Id y el Password, añadido un comando extra "pooling=false", esto lo hago para que no mantenga cacheado el pool de las sesiones conectadas. Resulta que C# lo tiene por defecto a true el pool, esto lo que hace es que le dice a ORACLE que cachee las sesiones durante un tiempo aunque estas se cierren, mantienen activos un tiempo para agilizar las nuevas conexiones, de manera que nuevas sesiones no abre nuevas sino reutiliza antiguas, esto hace que sea más rápido en caso que la base de datos tenga mucha concurrencia, pero esto a nosotros nos daría problemas con la tabla global temporary ya que si no elimina la sesión correríamos el riesgo de que datos de una sesión antigua se mezclaran con la nueva, por lo tanto no haremos esto del pool, en una caso real donde tuviéramos mucha concurrencia, entonces tendríamos que cambiar nuestra estrategia con la TMP y ponerle un id de transacción para no tener problemas de cruce de datos con otras sesiones.

4.2.3 Conectar JAVA

Con JAVA usaremos OJDBC, y al igual que ocurre con C# también usará un objeto, veremos que entre JAVA y C# hay bastante similitudes, más que con PHP aunque tampoco es que sean muy notables las diferencias con este. También en este caso debemos ponerlo en una función, en este caso lo tenéis en la clase JavaOracle.java en la función Conectar().

```
public Connection Conectar()
{
    try{
        Class.forName("oracle.jdbc.driver.OracleDriver");
        String BaseDeDatos = "jdbc:oracle:thin:@localhost:1521:ORCL";
        conexion= DriverManager.getConnection(BaseDeDatos,"MUNDIAL","1234");
        if(conexion!=null)
            System.out.println("Conexión realizada con éxito a MUNDIAL");
        else
            System.out.println("Conexión fallida");
    }
    catch(Exception e)
    { System.out.println("FALLOOOOOO EXCEPCION!!!"); e.printStackTrace(); }
    return conexion;
}
```

4.2.4 Desconectar PHP

Es importante recordad que debemos desconectarnos una vez acabamos de hacer cosas con ORACLE para no gastar recursos ni memoria y para que se cierre la sesión y limpie nuestra TMP.

Tendremos dos funciones, la primera para liberar todos los elementos y estructuras que se usaron en la llamada a los procedimientos almacenados y luego la función de cierre de sesión con ORACLE.

oci_free_statement(<id de la llamada a ORACLE>) y oci_close(<id de la conexión>)

Ejemplo:

```
oci_free_statement($stid);
oci_close($conexion);
```

4.2.5 Desconectar C#

En C# será llamar al método Close() del objeto de la conexión.

Ejemplo:

```
conn.Close()
```

4.2.6 Desconectar JAVA

En JAVA igual que en C# tendremos el método Close() en el objeto de conexión, pero también podemos cerrar el objeto recorset y el objeto statement que se usen para recibir datos de procedimientos almacenados, os he creado una clase en JAVA en el fichero sqlTools.java que lo hace:

SqlTools.close(rs, cs, con); Que lo único que hace es recoger los 3 objetos y llamar a sus métodos close(), si no los habéis usados los 3, podéis cerrarlo individualmente.

4.3 Llamar a procedimientos almacenados

4.3.1 Llamar con PHP

A un procedimiento almacenado lo podemos llamar de dos maneras:

1. Como si estuviéramos en un bloque anónimo, es decir, anteponiendo el BEGIN...END;
2. Con la cláusula CALL.

Ejemplo:

Si es un procedure:

1. call prueba.mostrarEquipo(:eq,:rsPart)
2. BEGIN prueba.mostrarEquipo(:eq,:rsPart); END;

Si es una función:

1. call prueba.LLenar_Desplegables(:tabla, :posicion) into :refCurResult
2. begin :refCurResult := prueba.LLenar_Desplegables(:tabla, :posicion); end;

Teniendo esto claro, ahora veremos las funciones OCI que debemos utilizar para conseguir llamar y recoger los datos que nos ofrece:

<id de procedimiento> = oci_parse (<id de conexión>, <llamada al procedimientos almacenado>): Es la función que prepara la llamada al procedimiento almacenado, debemos darle el id de conexión que nos ha devuelto la función de conexión para saber la sesión que esta pidiendo llamar a ORACLE y qué procedimiento almacenado deseamos llamar, todo los parámetros y si es una función lo que devuelve se debe poner el carácter ":" delante de un nombre que queramos que tenga referencia al parámetro que corresponda en el PL/SQL, incluso si usamos el mismo mejor. Pero podemos poner cualquier cosa, a veces se ponen números.
Devuelve un identificador de dicho procedimiento, para que lo usen otras funciones.

oci_bind_by_name (<id de procedimiento>, <parámetro pl/sql>, <variable php>, <max longitud>, <tipo>): Su propio nombre lo indica, bindea una variable a un parámetro del procedimiento almacenado, es decir, vincula la variable de PHP al parámetro de sustitución de variable vinculada de Oracle, entonces hace un doble juego, si el parámetro es de tipo IN entonces la variable de PHP le da el valor al parámetro del PL/SQL, y si este es de tipo OUT entonces es al revés el valor que resulta al acabar el procedimiento en el PL/SQL se lo pasa a la variable de PHP.

Los valores <max longitud> y <tipo>: Son optativos para tipo IN. <max longitud> Establece la longitud máxima para los datos, y es obligatoria para las variables que sean de tipo OUT. <tipo> es el tipo de dato.

Ejemplo:

```
$stid = oci_parse($conexion, 'call prueba.LLenar_Desplegables(:tabla, :posicion) into :refCurResult');
oci_bind_by_name($stid,":tabla",$tabla);
oci_bind_by_name($stid,":posicion",$posicion);
oci_bind_by_name($stid,":refCurResult",$deplegableEquipos,-1,OCI_B_CURSOR);
```

Y si es un tipo OUT o de devolución escalar, por ejemplo una función que retorna un NUMBER, sería:

```
oci_bind_by_name($stid,":cantidad",$cantidad,20);
```

Donde 20 sería la longitud máxima del número a devolver. Podemos poner -1, entonces ORACLE establecerá la longitud actual de la variable como máxima.

También tenermos una modalidad de esta función específica para Arrays, que usaremos para cuando queramos devolver desde PL/SQL un Array en vez de un tipo convencional, cosa que hacemos al obtener un jugador (lo podéis ver en el ejemplo resuelto):

En este caso tendremos más parámetros, como vemos en el ejemplo, le decimos, cuantas posiciones se esperan, la longitud máxima que tiene cada posición y el tipo de dato, y de este modo tan sencillo tendremos un Array de PL/SQL en un Array de PHP:

```
oci_bind_array_by_name($stid, ":c1", $fila, 5, 100, SQLT_CHR)
```

Nos habremos percatado, en ejemplo anterior, que una de las variables es un REF CURSOR. Es un tipo especial que requiere de una función especial para preparar a PHP para ella:

oci_new_cursor(<id de conexion>): Devuelve una variable en PHP preparada para recibir un cursor.

Ejemplo:

```
$deplegableEquipos = oci_new_cursor($conexion)
```

Existen otras funciones OCI para tipos complejos, a parte del CURSOR, como por ejemplo para un BLOB, como podréis ver en el ejemplo que os hago de grabar un jugador, allí necesitaremos trabajar en PHP con un BLOB, para ello en ves de oci_new_cursor usaremos OCINewDescriptor que funciona igual:

```
$foto = OCINewDescriptor($conexion, OCI_D_LOB)
```

Finalmente necesitamos la función que ejecuta todo:

oci_execute(<id a ejecutar>, <modo>): El id a ejecutar suelo ser un id de procedimiento, lo que devuelve oci_parse, y así se ejecuta el procedimiento almacenado, pero también puede ser la variable cursor, para que se llene con el cursor de PL/SQL.

El <modo> es para el famoso auto commit, por defecto es auto commit, es decir si el procedimiento acaba correctamente se hace commit, pero podemos hacer que no haya auto commit, basta con poner OCI_DEFAULT como modo. Esto es algo que usaré en PHP para grabar la foto del BLOB, que el commit, lo hará el PHP y entonces quitaré el autocommit.

Ejemplo:

```
oci_execute($stid);  
oci_execute($deplegableEquipos);
```

Ejemplo completo que llama a llenar el desplegable:

```
$stid = oci_parse($conexion, 'call prueba.LLenar_Desplegables(: tabla, : posicion) into : refCurResult');  
$deplegableEquipos = oci_new_cursor($conexion);  
$tabla="EQUIPOS";  
$posicion=1;  
oci_bind_by_name($stid,": tabla",$tabla);  
oci_bind_by_name($stid,": posicion",$posicion);  
oci_bind_by_name($stid,": refCurResult",$deplegableEquipos,-1,OCI_B_CURSOR);  
oci_execute($stid);  
oci_execute($deplegableEquipos);
```

4.3.2 Llamar con C#

Aunque al final es siempre lo mismo, hay ciertas diferencias, además con C# tenemos más versatilidad y podemos devolver los datos no solo en un record set como hace PHP si no también en un tabla o otros tipo de C#, aunque prácticamente es lo mismo.

Para seguir el mismo ejemplo que con PHP, veremos que variables y tipos necesitaremos:

- Un oracleCommand, que es lo mismo que el OCI_PARSE de PHP
- Un OracleDataAdapter que nos permitirá ejecutar el procedimiento y convertirlo en un tipo que C# puede almacenar para recorrer después.
- Un DataTable, esta es una de las posibilidades, almacenarlo en una tabla, que facilmente podemos recorrer. O meter directamente en un dataGridView.
- Hay otras opciones, como meter cualquier parámetro de tipo OUT en tipos concretos en C# jugando con los parámetros del OracleCommand, como veréis que hago en grabarJugador.cs donde obtento un Array y un BLOB, por no complicar los apuntes no lo desarrollo aquí, pero en ese fichero podéis ver como obtenerlos.

```
OracleCommand objSelectCmd = new OracleCommand();  
OracleDataAdapter objAdapter = new OracleDataAdapter();  
DataTable dt = new DataTable();
```

Una vez declaradas las variables, empezamos a preparar el procedimiento almacenado, le damos la conexión con

```
objSelectCmd.Connection = conn;
```

Le decimos como se llama el procedimiento almacenado:

```
objSelectCmd.CommandText = "prueba.LLenar_Desplegables";
```

Le decimos el tipo que es:

```
objSelectCmd.CommandType = CommandType.StoredProcedure;
```

Y vamos poniendo en el mismo orden y con el mismo nombre que tienen en el PL/SQL cada uno de los parámetros. Y tendremos 3 tipos:

- Entrada: Entonces usaremos VALUE para mandarse el valor.
- Salida: Entonces usaremos DIRECTION y pondremos que es de tipo OUT
- Return: Si es una función y devuelve algo, se pondrá en Direction también, y diremos que es un returnvalue.

En este ejemplo son 2 parámetros de entrada y uno REF CURSOR que se devuelve por return:

```
objSelectCmd.Parameters.Add("tabla", OracleDbType.Varchar2, 30).Value = tabla
objSelectCmd.Parameters.Add("posicionCampo", OracleDbType.Int32, 100).Value =
posicionCampoMostrar
objSelectCmd.Parameters.Add("estructura", OracleDbType.RefCursor).Direction =
ParameterDirection.ReturnValue
```

Si en vez de ser devuelto por return, hubiera sido un procedimiento con el REF CURSOR como parámetro de salida (OUT) hubiera sido así:

```
objSelectCmd.Parameters.Add("estructura", OracleDbType.RefCursor).Direction =
ParameterDirection.Output
```

Pero también podemos querer obtener su valor en una variable en el caso que sea un tipo escalar y no un REF CURSOR, por ejemplo si fuera un NUMBER, no usaríamos Direction:

```
var numDevuelto = objSelectCmd.Parameters.Add("num", OracleDbType.Int16,
ParameterDirection.ReturnValue);

OracleDecimal numero = (OracleDecimal) numDevuelto.Value;
```

Una vez ya tienes todos los parámetros para que estos queden "bindeados" con PL/SQL es conveniente usar:

```
objSelectCmd.BindByName = true;
```

Y finalmente solo queda ejecutar, tenemos varias formas, dependiendo donde queramos guardar los resultados, como en este caso ese en una tabla, podemos usar el objeto oracleDataAdapter que transforme el resultado en una tabla C# llenándola:

```
objAdapter.SelectCommand = objSelectCmd;
objAdapter.Fill(dt);
```

Otra forma de ejecutar es con el método del objeto OracleCommand que ejecuta el procedimiento parecido a OCI_EXECUTE de PHP:

```
objSelectCmd.ExecuteNonQuery();
```

Esto lo haremos cuando no quedemos hacer una tabla de C# si no recuperar datos en variables de salir directamente, como un BLOB, un array o un dato escalar, en grabarJugadores tenéis el ejemplo.

4.3.3 Llamar con JAVA

Como era de esperar, tiene muchas similitudes con C#, nos declararemos 3 variables:

```
Connection con = null
ResultSet rs = null
CallableStatement cs = null
```

La conexión, el ResultSet que haría de table de C# y el CallableStatement que es lo mismo que el OracleCommand de C# o el OCI_PARSE de PHP, será donde preparamos la llamada al procedimiento almacenado y sus parámetros.

Desde luego es el que más me gusta de los 3, lo veo más claro. En este caso debes poner cada parámetro o valor a devolver como un interrogante, y después en orden le vas diciendo cada uno qué es:

```
String sql = "{?=call prueba.Llenar_Desplegables(?,?)}"
cs = con.prepareCall(sql)
```

Y luego pones los parámetros, cada uno tiene una posición que debes poner con un número bien ordenado:

```
int pos = 0
cs.registerOutParameter(++pos, OracleTypes.CURSOR)
cs.setString(++pos, tabla)

cs.setInt(++pos, posicionCampo)
```

Vemos que vamos poniendo cada parámetro, y tendremos de dos tipos:

- De salida o devolución: Entonces ponemos registerOutParameter
- De entrada: Pondremos el tipo de dato del parámetro, setString, setInt....

Finalmente para ejecutar es simplemente llamar al método Execute, y después podemos obtener cualquier de los valores de los parámetros simplemente accediendo con su tipo de dato a la posición concreta del mismo:

```
// Ejecutamos
cs.execute();
rs = (ResultSet) cs.getObject(1);
```

Y este siempre se hace así veamos el ejemplo que veamos, sea del tipo que sea, siempre del mismo modo, lo considero bastante atractivo.

Si queremos obtener un dato escalar sería exactamente igual (cambiando el tipo de dato del get), suponiendo que el número fuera el primer interrogante del procedimiento almacenado:

```
int numero = cs.getInt(1)
```

4.4 Trabajar con los datos obtenidos

Lo último una vez has llamado y rellenado las variables de la aplicación simplemente es recorrerlas e ir poniendo los datos donde quieras.

4.4.1 Trabajar con PHP

Si es un valor escalar el que se devuelve o un array pero con 1 sola fila entonces ya hemos visto antes como se recupera (`oci_bind_by_name`), se mete en una variable PHP y se puede usar normalmente.

Cuando se devuelven varias fila como con un REF CURSOR, para poder recorrerlo tenemos la función **`oci_fetch_array`**, que se le debe pasar la variable PHP que guarda el REF CURSOR y le decimos si queremos un array asociativo, numérico o ambos (el mejor es asociativo) de manera que nos sea fácil detectar que columna es. Y esta funci

```
while ($fila = oci_fetch_array($partidos, OCI_ASSOC))
{
    <tr><td>
        <?php echo $fila['N1']. ' - ' . $fila['C2']. ' - ' . $fila['C3']. ' - ' . $fila['C4'];?>
        <A href="grabarJugador.php?jugador=<?php echo $fila['C3'];?>&seleccion=<?php echo
        $equipo;?>"> <input type="button"> </A>
    </td></tr><?php
}
```

4.4.2 Trabajar con C#

En C# ya vimos que si tenemos que trabajar con algo escalar que nos devuelvan, debemos asignar una variable al parámetro `returnvalue` o `out` y luego usamos una función de conversión al tipo deseado.

En cambio si son varias filas las que se devuelven, lo mejor es usar un `DataTable` y este lo podemos meter en un `dataGridView`.

```
objAdapter.SelectCommand = objSelectCmd
objAdapter.Fill(dt)
dataGridView1.DataSource = dt
```

Y ya en el `dataGridView` podemos hacer que campos no salgan o se le cambie el título de cabecera:

```
--Para ocultar columnas:
dataGridView1.Columns["ORIGEN_DATOS"].Visible = false
.....
-- Para cambiar el título de cabecera
dataGridView1.Columns["N1"].HeaderText = "DORSAL"
```

4.4.3 Trabaja con JAVA

En JAVA para conseguir datos tanto escalares como complejos de tipo OUT o RETURNO, es con la función get del tipo de dato que corresponda:

Número:

```
int num = cs.getInt(1);
```

REF CURSOR:

```
rs = (ResultSet) cs.getObject(6);
```

Y para recorrer un ResulSet se utilizar la función NEXT y lo más común es meterlo en un JTABLE, donde también podemos cambiar el nombre a las cabeceras:

```
String Titulos[]={" DORSAL "," PUESTO "," NOMBRE "," MUNDIALES JUGADOS "," CANTIDAD "};
String fila[]=new String[5];
while (rs.next()) {
    fila[0] = rs.getString("N1");
    fila[1] = rs.getString("C2");
    fila[2] = rs.getString("C3");
    fila[3] = rs.getString("C4");
    fila[4] = rs.getString("N2");
    modelo.addRow(fila);
}
jTable1.setModel(modelo);
```