Trabajando con bases de datos. ClaseOracleBD

Índice de contenido

Archivo de configuración

Podemos almacenar la configuración de nuestra aplicación web dentro de un archivo que se llama **web.config**. En este capítulo lo utilizaremos para guardar la cadena de conexión a la base de datos.

Se puede editar directamente o utilizar las herramientas que nos proporciona Visual Studio para ello, con lo que será más fácil y evitaremos errores.

Agregar una cadena de conexión a Web.config con un control de origen de datos

Utlizando un Control SQLDataSource

Vamos a utilizar un control ***SqlDataSource*** para que Visual Studio nos modifique automáticamente el fichero **web.config**. Además de está forma podremos probar la conexión antes de añadirla.

Abra una página .aspx en la vista Diseño en Visual Studio.

En el Cuadro de herramientas, en la carpeta Datos, arrastre un control de origen de datos, por ejemplo un control **SqlDataSource**, hasta la superficie de diseño.

Haga clic con el botón secundario del ratón en el control y, a continuación, haga clic en **Mostrar etiqueta inteligente**.

En el panel de etiquetas inteligentes, haga clic en **Configurar origen de datos**. (*Ilustración 1*)

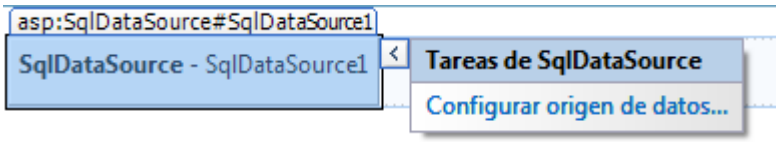


Ilustración 1: Configurar origen de datos

En el panel **Elegir una conexión de datos**, haga clic en **Nueva conexión**.

Seleccione *Base de datos de Oracle* de la lista en el cuadro de diálogo **Elegir una conexión de datos**, y luego haga clic en **Aceptar**. (*Ilustración 2*)

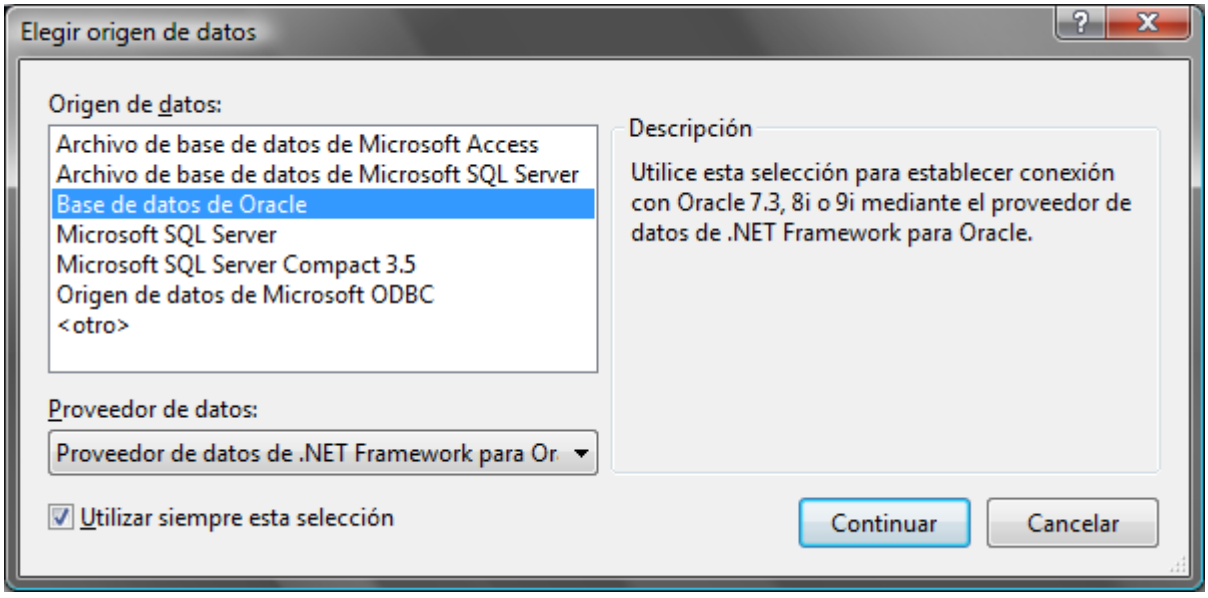


Ilustración 2: Elegir una conexión de datos

Indique el nombre de servidor, nombre de usuario y contraseña correctos en el cuadro de diálogo **Agregar conexión**, y después haga clic en **Aceptar**. (*Ilustración 3*).

Regresa al cuadro de diálogo **Configurar origen de datos** con un resumen de los detalles de su conexión.

Haga clic en **Siguiente**, y luego en **Sí** para guardar su cadena de conexión en el archivo **Web.config**. Elegiremos un nombre para nuestra cadena de conexión.

La cadena de conexión ahora se almacena en el archivo **Web.config** y puede configurar los detalles de la consulta para su control

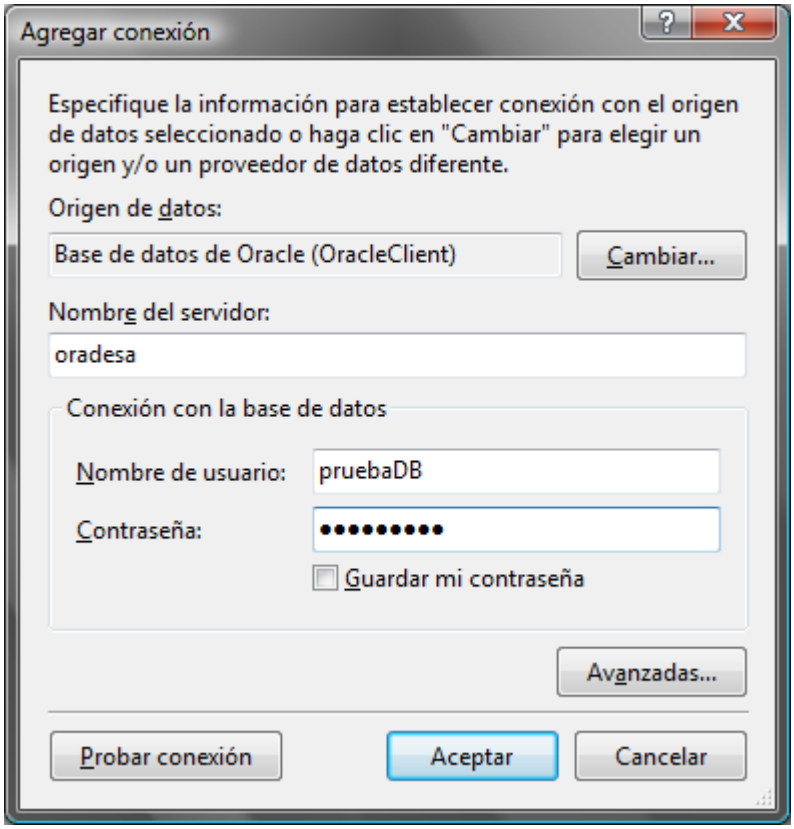


Ilustración 3: Agregar conexión

Si abre el archivo **web.config** podrá ver que se ha agregado una entrada dentro del grupo **ConnectionStrings** con la conexión creada.

Manualmente

Para crear una cadena de conexión en el archivo **web.config** añadiremos las siguientes etiquetas:

```
<connectionStrings>
  <add name="ConnectionString1"
   connectionString=
      "Data Source=oradesa;User ID=pruebaDB;Password=xxxx;Unicode=True"
   providerName="System.Data.OracleClient"/>
</connectionStrings>
```

Utilización de la cadena de conexión

Una vez creada la cadena de conexión podemos utilizarla en cualquier aspx de nuestra aplicación mediante la clase **ConfigurationManager**, que proporciona acceso a los archivos de configuración. Con el método **ConnectionStrings** accedemos a la sección **connectionStrings** de web.config. Tendremos que especificar el nombre de nuestra conexión entre corchetes y luego usar alguno de sus métodos o propiedades.

Por ejemplo, con el método **ConnectionString** se obtiene o establece la cadena de conexión:

```
cadenaConexion = ConfigurationManager.ConnectionStrings["ConnectionString1"].ConnectionString;
```

Para poder utilizar la clase **ConfigurationManager** tendremos que incluir la librería **Configuration**.

```
using System.Configuration;
```

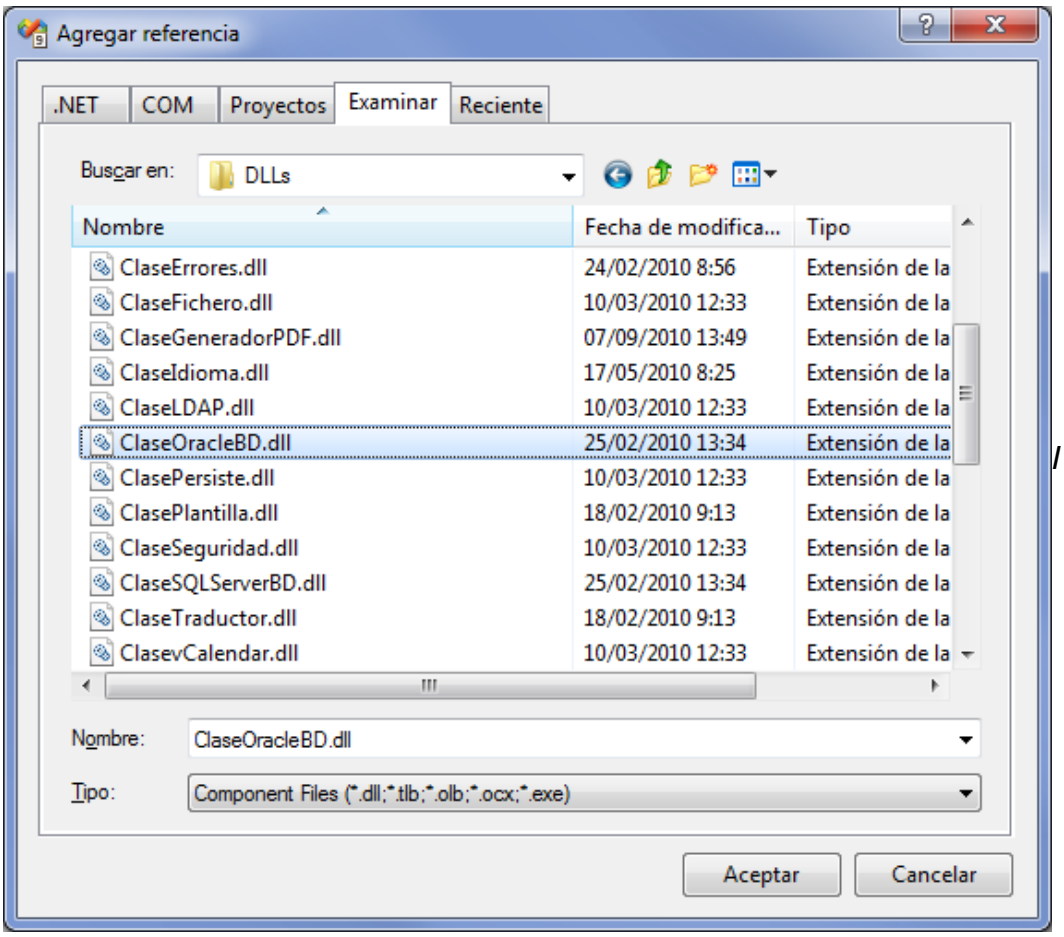
Integración de ClaseOracleBD

Vamos a ver como incluir la clase ClaseOracleBD en nuestro proyecto.

En el **Explorador de soluciones** haga clic con el botón derecho en el proyecto y seleccione **Agregar referencia...**

Todas las DLL's comunes se guardan en la carpeta \ToolsNet\DLLs del servidor

En la ventana que nos aparece (*Ilustración 4*) seleccionaremos la pestaña **Examinar** y buscaremos dentro de la carpeta anterior el fichero **ClaseOracleBD.dll**



lustración 4: Referenciar ClaseOracleBD

En nuestro aspx tendremos que incluir la correspondiente clausula using.

```
using ua;
```

También tendremos que agregar una referencia a OracleClient para trabajar con Oracle. Para ello pinchamos con el botón derecho en el proyecto y elegimos **Agregar referencia.** En la pestaña .NET escogemos System.Data.OracleClient y pulsamos Aceptar. (*Ilustración 5*).

En nuestro código incluiremos la clausula using correspondiente:

```
using System.Data.OracleClient;
```

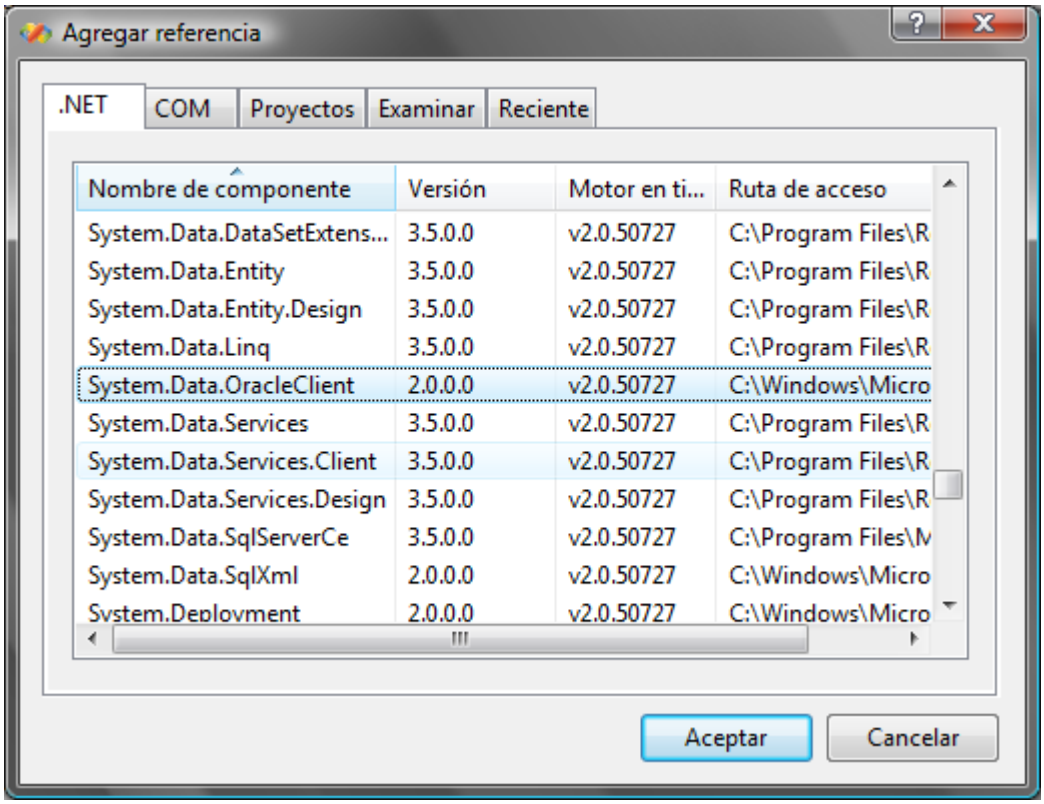


Ilustración 5: Referenciar OracleClient

Creación del objeto

Para definir una nueva instancia de la clase primera la declaramos y luego la creamos

```
ClaseOracleBD baseDatos;
baseDatos = new ClaseOracleBD();
```

O en una misma línea:

```
ClaseOracleBD baseDatos = new ClaseOracleBD();
```

Asociar una cadena de conexión

Para asociar la cadena de conexión definida en el web.config utilizaremos el método **CadenaConexion**.

Por ejemplo, para la cadena de conexión antes creada:

```
baseDatos.CadenaConexion =
ConfigurationManager.ConnectionStrings["ConnectionString1"].ConnectionString;
```

Ejecutar sentencias SQL

Especificaremos con el método **TextoComando** el comando sql que queremos ejecutar.

```
baseDatos.TextoComando = "SELECT descloc FROM localizaciones";
```

Ahora mediante la propiedad rs de cOracleBD podremos recorrer los valores obtenidos en la consulta. Esta propiedad nos devuelve un objeto DataReader del que utilizaremos el método HasRows para comprobar si hemos obtenido algún resultado y Read para obtener los datos. Accederemos al valor de cada columna de la consulta especificando su nombre o posición entre corchetes.

```
if (baseDatos.Rs.HasRows) {
    while (baseDatos.Rs.Read()) {
        Label1.Text += baseDatos.Rs["descloc"] + "<br/>";
    }
}
```

Un ejemplo completo de ejecutar una consulta sería:

```
...
using ua;
using System.Data.OracleClient;

public partial class _Default : System.Web.UI.Page
{
  ClaseOracleBD baseDatos;

  protected void Page_Load(object sender, EventArgs e) {
    baseDatos = new ClaseOracleBD();

    baseDatos.CadenaConexion =
ConfigurationManager.ConnectionStrings["ConnectionString1"].ConnectionString;


    baseDatos.TextoComando = "SELECT descloc FROM ctlsrv_localizaciones";

    if (baseDatos.Rs.HasRows)
    {
      Label1.Text = "";
      while (baseDatos.Rs.Read()) {
          Label1.Text += baseDatos.Rs["descloc"] + "<br/>";
      }
    }
    baseDatos.Close();
  }
}
```

Ejecutar sentencias SQL con Bind Variables

Para ejecutar sentencias que se ejecutan muchas veces utilizaremos bind variables para mejorar el rendimiento.
Primero insertaremos en la sentencia las bind variables

```
String cadSQL;
casSQL = "SELECT descloc FROM localizaciones WHERE codloc=:pCodLoc";
baseDatos.TextoComando = cadSQL;
```

Ahora tendremos que darle valor a las bind variables que hemos puesto en la sentencia anterior.
Para esto utilizaremos el método **CrearParametro**. Esté método tiene cuatro parámetros:

1. Nombre de la bind variable
2. Tipo, Definidos en *OracleType*
3. Dirección. Definidos en *System.Data.ParameterDirection*
4. Tamaño
5. Valor

```
baseDatos.crearParametro("pCodLoc",
   OracleType.Number,
   System.Data.ParameterDirection.Input,
   0,
   codLoc);
```

Para variables numéricas el tamaño será 0, si la variable es de texto sería:

```
baseDatos.crearParametro("pCaca",
         OracleType.VarChar,
         System.Data.ParameterDirection.Input,
         sCaca.Length,
         sCaca);
```

Ahora accederemos a los datos de la misma forma que en la consulta sin bind variables. En este ejemplo utilizaremos otro objeto DataReader

Declaramos el objeto DataReader

```
OracleDataReader reader;
```

Ejecutamos la consulta y nos devuelve el DataReader

```
reader = baseDatos.rs;
```

Comprobamos que tiene datos

```
if (reader.HasRows) {
    do {
         Label1.Text += reader["descloc"] + "<br />";
       } while (reader.Read());
}
```

Cerrar los objetos

Es importante cerrar los objetos de base de datos al terminar de utilizarlos. Si no cerramos los objetos nos podemos encontrar con conexiones que se mantienen abiertas. Utilizaremos el método Close(), de DataReader y ClaseOracleBD

Por ejemplo, para cerrar los objetos antes creados

```
reader.Close();
oBD.Close();
```

Ejecución de procedimientos

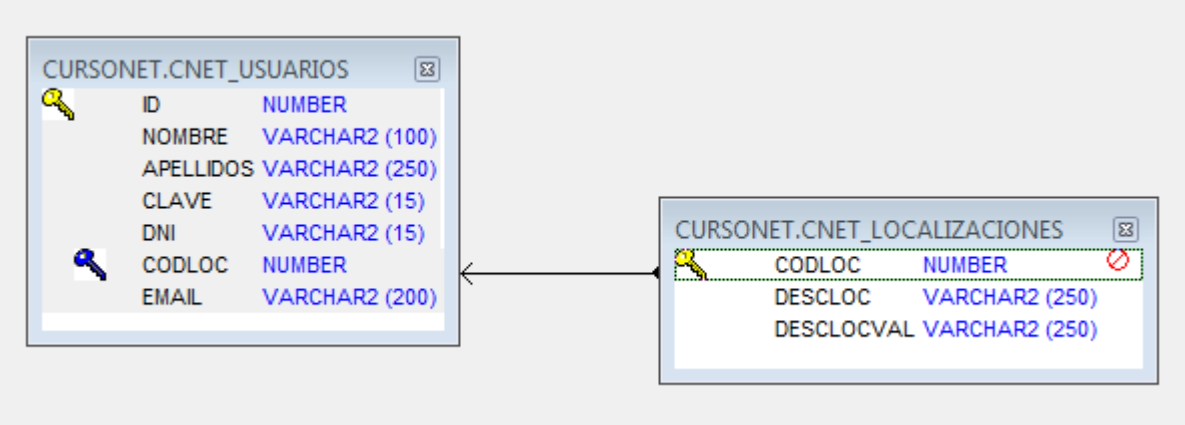
Para ejecutar procedimientos almacenados primero definimos el tipo de comando, luego declaramos el procedimiento sin parámetros y vamos añadiendo los parámetros como hacíamos con las bind variables. Al final ejecutaremos con el método **Ejecuta**.
En el siguiente ejemplo se puede ver todo el proceso:

```
// Definimos el tipo de comando
baseDatos.TipoComando= CommandType.StoredProcedure;
// Declaramos el procedimiento sin parámetros
baseDatos.TextoComando= "PRUEBA_PK.Almacenar";
// Declaramos los parámetros, no importa el orden
baseDatos.CrearParametro("param1",
   OracleType.Number,
   System.Data.ParameterDirection.Input, 0, valorParam1);
baseDatos.CrearParametro("param2",
   OracleType.VarChar,
   System.Data.ParameterDirection.Input, valorParam2.Length, valorParam2);
//Ejecutamos el procedimiento
oBD.Ejecuta();
```

De base de datos a clases

El transformar nuestras tablas de base de datos en un buen esquema de clases nos puede facilitar la implementación de nuestras aplicaciones, aumentando la sencillez, legibilidad y reutilización.

Por ejemplo, con un esquema como este:



Nos crearíamos las siguientes tablas, aunque depende también de como usen los datos la aplicación:

- **ClaseUsuario**: Para tratar los datos de los usuarios
- **ClaseLocalizacion**: Para tratar los datos de las localizaciones
- **ClaseUsuarios**: Clase estática. Para filtrar usuarios

Las clases ClaseUsuario y ClaseLocalizacion tendrán variables miembro y propiedades almacenar y utilizar los campos de las tablas de la base de datos que necesitemos en la aplicación.

Tanto la clase usuario como localizaciones tendrán métodos Carga y Guarda, para leer y modificar o insertar datos en la base de datos.

Para recuperar mas de un elemento de la base de datos utilizaremos clases estáticas que devuelven listas, como la ClaseUsuarios. Que devuelve listas de ClaseUsuario. El nombre será el mismo que la clase que trata pero en plural.

Por ejemplo si necesitamos que nuestra aplicación nos devuelva varias localizaciones nos crearemos una nueva clase ClaseLocalizaciones para filtrar objetos de ClaseLocalizacion

El diagrama de clases sería: