ESTUDIO DE LA VERSIÓN 4 de la Librería

1.-Introducción.

2.-Pasos a seguir.

3.- Seguimiento detallado de la versión LibreriaV4-Inicial.

4. Ejercicios.

1.- Introducción.

En el estudio de este tema, comenzamos haciendo una breve introducción, de la evolución de nuestra aplicación (versiones Libreria 1,2 y 3) hacia la gestión con bases de datos. Se eliminará el fichero de la versión 3 y se pondrá como almacén de datos una base de datos.

2.- Pasos a seguir.

1.- Se indica al alumno, que instale en XAMPP la base de datos que se encuentra en la carpeta **BBDD**.

2.- Seguidamente, instalamos el paquete **Nuget** necesario para poder conectarnos con las base de datos(**MySQL Data**). En este paquete estarán las clases necesarias para poder realizar la conexión con la base de datos. MYSQL Data, es el driver de conexión con MySQL. Haremos una semejanza con los drivers de Java. En Java, primero se sube el driver a memoria antes de ser utilizado. En C# , todo este proceso lo realiza el sistema.

Para la instalación de estos paquetes, hay que ver la documentación de los fichero **Manual-Configuración-1.doc** y **Manual-Configuración-2.doc,** que se encuentran en la carpeta **Documentacion.** Estos manuales, podemos encontrarlos en el manual completo **Documentacion-NET-FINAL.doc.**

Tendremos en cuenta el fichero **IncidenciasV4.doc** de la carpeta **Documentación**, por si se produjese un error de ese tipo.

3.- Una vez creada la base de datos y configurados los drivers, instalamos la aplicación **LibreriaV4-Inicio**, para estudiar los conceptos del acceso a datos en C#. Pasamos el depurador por todo el proyecto, explicando los conceptos encontrados. En primer lugar, el alumno debe entender el máximo código posible. Si disponemos de tiempo, también propondríamos realizar alguna mejora de la aplicación.

Hemos decidido explicar previamente esta aplicación antes de la final **LibreriaV4**, porque tiene menos complejidad al tener únicamente la clase **AccesoBD**. La gestión se realiza desde la capa de presentación utilizando directamente los métodos de la framework AccesoBD. En la versión final de LibreriaV4, utilizaremos una clase intermedia entre AccesoBD y la capa de presentación, llamada **AccesoLibro** , que permitirá abstraernos de los métodos primitivos de AccesoBD. Esta forma de trabajar, permitirá realizar proyectos más genéricos y fáciles de ampliar.

En esta versión inicial de LibreriaV4, hemos quitado complejidad, creando dos formas de montar las sentencias SQL. En la primera, creamos la sentencia SQL por partes, antes de enviarla a las órdenes de AccesoBD(**ejecutarUpdate() y ejecutarConsulta()).** En la segunda forma, utilizamos métodos de la clase **UtilSql**, que automatiza la creación de las sentencias. Esto hace que el programador de la capa de presentación realice aplicaciones más rápidamente y eficientes.

4.- Después de entender esta aplicación, pasamos a ver el proyecto final **LibreriaV4**, que es una evolución del anterior y donde podemos observar la arquitectura completa del proyecto. Aquí también se trazará el programa e iremos explicando los conceptos encontrados.

5.- Lectura del **MANUAL-V4**. En este manual, se realiza una explicación detallada de todos los conceptos utilizados en la versión 4.

6.- Para finalizar el estudio del tema, leer el fichero **modificaciones.doc** de la carpeta **DOCUMETACIÓN**, por si fuesen necesarias tener en cuenta sus anotaciones.

7.- El grupo de alumnos creará un único fichero de **INCIDENCIAS-V4.doc** compartido(válido para toda la clase), donde pondrán todas las incidencias encontradas y cualquier aportación que se haga al proyecto.

3.- Seguimiento detallado de la versión LibreriaV4-Inicial.

1.- Se produce un error que no está controlado, si la base de datos no está abierta. **Proponer al alumno arreglar dicho error.** En la llamada al método abrirConexión() de AccesoBD, se produce el error, que es enviado a la capa presentación, pero como en ésta no tenemos un try catch, no se puede controlar.

2.- Los Temas de la capa de presentación se cargan en el Diseñador. En próximas versiones, los temas estarán en una tabla.

3.- Podéis codificar la gestión centralizada de los **mensajes**, aunque en versiones posteriores se realiza.

4.- En el proyecto **NO** están contemplados todas las incidencias. Se deja que alumno, descubra el máximo número de ellas. Por ejemplo, si introduzco **títulos** de libro repetidos , producirá un error. Aunque la tabla tenga el campo **CodLibro**, no vamos a tener en cuenta su gestión. En la clase **Tlibro(**POJO**)** lo ponemos en comentarios.

5.- En el DTO(POJO) TLibro, no gestionamos los campos **CodLibro** y **Borrado**, para evitar complejidad al proyecto. Hay que tener cuidado en versiones posteriores, que CodLibro no esté como clave primaria. En la arquitectura habría que hacer algún cambio, para que pueda actuar CodLibro como clave primaria. Esto se indica, porque el proyecto fallaría en determinados casos por esta circunstancia. **Se puede dejar como tarea resolver este problema, aunque de momento no es necesario prestar atención a esta incidencia.**

6.- El alumno comprobará en este proyecto, que el montaje de las sentencias SQL, se gestionan de dos maneras. La primera, utilizando directamente la orden sql y la otra mediante la clase UtilSql.

7.- En esta versión, no se gestionan los errores en la capa de presentación.

8.- En esta versión, la sintaxis de las órdenes SQL es distinta a la de la versión final de LibreriaV4, pero el resultado es el mismo. De esta forma, aprendemos distintos métodos de montaje de sentencias SQL.

9.- En AccesoBD cuando se produce un error, no se gestiona en la clase, se envía mediante la orden **throw** , a la capa superior(Presentación).

10.- Si la sintaxis de algunas sentencias, parece algo compleja su lectura, se puede realizar en varios pasos para comprenderla. Un ejemplo :

```
comando = new MySqlCommand(sql, conexion);
return comando.ExecuteNonQuery() > 0;
```

Puedes descomponerla en varias órdenes para entenderla mejor.

```
comando = new MySqlCommand(sql, conexion);
Boolean estado=comando.ExecuteNonQuery() > 0;
return estado;
```

Ejercicios.

1.- Realizar un estudio detallado del método ejecutarConsulta() de AccesoBD. Este ejercicio **NO** se hará, mientras no se domine la arquitectura.

2.-Desarrollar algún método de la clase AccesoLibro **(proyectoLibreriaV4)**, partiendo de LibreriaV3(ficheros) y LibreriaV4-Inicial. Para realizar este ejercicio, debéis fijaros en la gestión de los dos proyectos. Se trata de quitar a la capa de presentación, la responsabilidad de conocer la gestión de las órdenes primitivas de base de datos y traspasar toda esa lógica a la nueva clase AccesoLibro**. Pondremos un ejemplo de cómo se haría con la lógica interna del método Form_Load().**

3.- Realizar una comparativa entre las dos versiones de Libreriav4 y enumerar las mejoras de la versión final de esta nueva arquitectura.

4.- Crear una clase **AccesoF** en la versión LibreriaV3, que contenga los métodos primitivos de acceso a ficheros y realizar la gestión del proyecto con esa arquitectura.