|  |
| --- |
| ClassLibrary建立 |
| |  | | --- | | **建立檔案** | | 檔案 =>專案 =>新增 =>類別庫(.NET Framework) | |

|  |
| --- |
| ClassLibrary宣告使用 |
| |  | | --- | | **建立檔案** | | 1. ClassLibrary專案先重建方案(可建立D: \SQL\_ClassLibrary\SQL\_ClassLibrary\bin\Debug\SQL\_ClassLibrary.dll)  2.需要使用SQL\_ClassLibrary的專案中  參考 =>加入參考 => 瀏覽 =>D: \SQL\_ClassLibrary\SQL\_ClassLibrary\bin\Debug\SQL\_ClassLibrary.dll  3.在程式中宣告  using SQL\_ClassLibrary;  public Class1 class1 = new Class1(); // Class1 是SQL\_ClassLibrary中 namespace的名稱 | |

|  |
| --- |
| ClassLibrary專案內碼更新 |
| |  | | --- | | **建立檔案** | | 1. ClassLibrary專案先重建方案(可建立D: \SQL\_ClassLibrary\SQL\_ClassLibrary\bin\Debug\SQL\_ClassLibrary.dll)  2.需要使用SQL\_ClassLibrary的專案中  參考 =>加入參考 => 瀏覽 =>D: \SQL\_ClassLibrary\SQL\_ClassLibrary\bin\Debug\SQL\_ClassLibrary.dll | |

|  |
| --- |
| SQL資料查詢 |
| |  | | --- | | **呼叫SQL資料查詢語法** | | public List<SelectListItem> Types\_list()  {  string sql = "SELECT type\_id AS Value, name AS Text FROM API.dbo.Types WITH(NOLOCK)";  return SQL\_select\_exce (sql);  }  private List<SelectListItem> GetSelectListItems(string sqlQuery)  {  List<SelectListItem> return\_list = new List<SelectListItem>();  // 查詢資料  DataTable reader = class1.SQL\_select\_exce(sqlQuery);  // 將資料讀取並轉換為 SelectListItem  for (int a = 0; a < reader.Rows.Count; a++)  {  return\_list.Add(new SelectListItem  {  Text = reader.Rows[a]["Text"].ToString(),  Value = reader.Rows[a]["Value"].ToString()  });  }  return return\_list;  } | | **SQL資料查詢FUNCTION** | | public DataTable SQL\_select\_exce(string sqlstring)  {  string connectionString = "連資料庫路徑";  SqlDataReader reader;  DataTable data = new DataTable();  using (SqlConnection conn = new SqlConnection(connectionString))  {  conn.Open();  using (SqlCommand cmd = new SqlCommand(sqlstring, conn))  {  reader = cmd.ExecuteReader();//讀取回傳的資料  data.Load(reader);  }  }  return data;  } | |

|  |
| --- |
| SQL資料新增修改 |
| |  | | --- | | **呼叫SQL資料新增修改語法** | | public void Insert\_update\_Products(List<Production> input\_list)  {  string sql\_list = string.Empty;  sql\_list = $@"  DECLARE @NewID INT;  SET @NewID = (SELECT ISNULL(MAX(CAST(SUBSTRING(product\_id, 2, 4) AS INT)), 0) FROM API.dbo.Production);  IF EXISTS(select 1 from API.dbo.Production where product\_id=@product\_id)  BEGIN  update API.dbo.Production  Set  type\_id=TRIM(@type\_id),  name=TRIM(@name),  number=@number,  price=@price,  dt\_date=SYSDATETIME()  where product\_id=@product\_id  END  ELSE  BEGIN  insert into API.dbo.Production  (  product\_id,  type\_id,  name,  number,  price,  dr\_date  )  values  (  'P' + RIGHT('000' + CAST(@NewID + 1 AS VARCHAR(4)), 4), -- 生成唯一的 OrderDetail ID  TRIM(@type\_id),  TRIM(@name),  @number,  @price,  SYSDATETIME()  )  END";  DataTable data = new DataTable();  data.Columns.Add("type\_id", typeof(string));//編號名稱  data.Columns.Add("type\_name", typeof(string));//編號名稱  data.Columns.Add("product\_id", typeof(string));//編號  data.Columns.Add("name", typeof(string));//商品姓名  data.Columns.Add("number", typeof(int));//庫存數量  data.Columns.Add("price", typeof(int));//售貨價格  // 將 List 內的數據逐一加到 DataTable  foreach (var product in input\_list)  {  if (product.type\_id != null && product.name != null && product.price != 0)  {  DataRow row = data.NewRow();  row["type\_id"] = product.type\_id;  row["type\_name"] = product.type\_name;  row["product\_id"] = product.product\_id;  row["name"] = product.name;  row["number"] = product.number;  row["price"] = product.price;  data.Rows.Add(row);  }  }  class1.SQL\_insert\_update(sql\_list, data);  } | | **SQL資料新增修改FUNCTION** | | public void SQL\_insert\_update(string sqlstring, DataTable data)  {  string connectionString = "連資料庫路徑";  using (SqlConnection conn = new SqlConnection(connectionString))  {  conn.Open();  for (int a = 0; a < data.Rows.Count; a++)  {  using (SqlCommand cmd = new SqlCommand(sqlstring, conn))  {  try  {  cmd.Parameters.AddWithValue("@type\_id", SqlDbType.VarChar).Value = data.Rows[a]["type\_id"];  cmd.Parameters.AddWithValue("@product\_id", SqlDbType.VarChar).Value = data.Rows[a]["product\_id"];  cmd.Parameters.AddWithValue("@name", SqlDbType.VarChar).Value = data.Rows[a]["name"];  cmd.Parameters.AddWithValue("@number", SqlDbType.Int).Value = data.Rows[a]["number"];  cmd.Parameters.AddWithValue("@price", SqlDbType.Int).Value = data.Rows[a]["price"];  cmd.ExecuteNonQuery(); // 執行  }  catch (Exception ex)  {  cmd.Parameters.AddWithValue("@inout\_id", SqlDbType.Int).Value = data.Rows[a]["inout\_id"];  cmd.Parameters.AddWithValue("@type\_id", SqlDbType.VarChar).Value = data.Rows[a]["type\_id"];  cmd.Parameters.AddWithValue("@product\_id", SqlDbType.VarChar).Value = data.Rows[a]["product\_id"];  cmd.Parameters.AddWithValue("@number", SqlDbType.Int).Value = data.Rows[a]["number"];  cmd.Parameters.AddWithValue("@price", SqlDbType.Int).Value = data.Rows[a]["price"];  cmd.ExecuteNonQuery(); // 執行  }  }  }  }  } | |

|  |
| --- |
| SQL資料刪除 |
| |  | | --- | | **呼叫SQL資料刪除語法** | | public void delete\_ProductionDetail(List<ProductionDetail> production)  {  string sql\_list = string.Empty;  foreach (var alls in production)  {  if (alls.inout\_id != null && alls.product\_id != null && alls.number != 0 && alls.price != 0)  {  sql\_list = $@" delete from API.dbo.ProductionDetail WHERE product\_id ='{alls. productiondetail\_id }' ";  class1. SQL\_delete (sql\_list);  }  }  } | | **SQL資料刪除FUNCTION** | | public void SQL\_delete(string sqlstring)  {  string connectionString = "連資料庫路徑";  using (SqlConnection conn = new SqlConnection(connectionString))  {  conn.Open();  using (SqlCommand cmd = new SqlCommand(sqlstring, conn))  {  cmd.ExecuteNonQuery(); // 執行  }  }  } | |

|  |
| --- |
| SQL 商品入庫及報銷(預存程式－方法一) |
| |  | | --- | | **SQL語法** | | USE [API]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[production\_insert\_update] \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[production\_insert\_update]  @inout\_id INT,  @product\_id VARCHAR(50),  @number INT,  @price INT  AS  BEGIN -- 將數據新增入 ProductionDetail 表  INSERT INTO API.dbo.ProductionDetail  (  inout\_id,  product\_id,  number,  price  )  values(  @inout\_id,  @product\_id,  @number,  @price  )  IF @inout\_id = 0  BEGIN -- 商品入庫(@inout\_id = 0)  UPDATE API.dbo.Production  SET number = number + @number  WHERE product\_id = @product\_id;  END  ELSE  BEGIN -- 商品報銷(@inout\_id = 1)  UPDATE API.dbo.Production  SET number = number - @number  WHERE product\_id = @product\_id;  END  END; | | **呼叫SQL預存程式語法** | | public List<ProductionDetail> EXCE\_ProductionDetail(List<ProductionDetail> production)  {  List<ProductionDetail> return\_list = new List<ProductionDetail>();  string sql\_list = string.Empty;  foreach (var alls in production)  {  if (alls.inout\_id != null && alls.product\_id != null && alls.number != 0 && alls.price != 0)  {  sql\_list = $@"exec [API].dbo.[production\_insert\_update]  @inout\_id={alls.inout\_id},  @product\_id='{alls.product\_id}',  @number={alls.number},  @price={alls.price}";  DataTable reader = class1.SQL\_select\_exce(sql\_list);  }  }  return return\_list;  } | | **SQL預存程式FUNCTION** | | public DataTable SQL\_select\_exce(string sqlstring)  {  string connectionString = "xxxx";  SqlDataReader reader;  DataTable data = new DataTable();  using (SqlConnection conn = new SqlConnection(connectionString))  {  conn.Open();  using (SqlCommand cmd = new SqlCommand(sqlstring, conn))  {  reader = cmd.ExecuteReader();  data.Load(reader);  }  }  return data;  } | |

|  |
| --- |
| SQL 下訂單(預存程式－方法二) |
| |  | | --- | | **SQL語法** | | /\*定義　<使用者資料表類型>\*/  USE [API]  GO  /\*\*\*\*\*\* Object: UserDefinedTableType [dbo].[OrderDetailType2] \*\*\*\*\*\*/  CREATE TYPE [dbo].[OrderDetailType2] AS TABLE(  [product\_id] [varchar](50) NULL,  [number] [int] NULL,  [price] [int] NULL  )  GO | | USE [API]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[production\_insert\_update] \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER PROCEDURE [dbo].[production\_insert\_update]  @inout\_id INT,  @type\_id VARCHAR(50),  @product\_id VARCHAR(50),  @number INT,  @price INT  AS  BEGIN -- 將數據新增入 ProductionDetail 表  INSERT INTO API.dbo.ProductionDetail  (  inout\_id,  type\_id,  product\_id,  number,  price  )  values(  @inout\_id,  @type\_id,  @product\_id,  @number,  @price  )  IF @inout\_id = 0  BEGIN -- 商品入庫(@inout\_id = 0)  UPDATE API.dbo.Production  SET number = number + @number  WHERE product\_id = @product\_id;  END  ELSE  BEGIN -- 商品報銷(@inout\_id = 1)  UPDATE API.dbo.Production  SET number = number - @number  WHERE product\_id = @product\_id;  END  END; | | **呼叫SQL預存程式語法** | | public string EXCE\_OrderDetail(New\_Order OrderDetail)  {  string return\_list = "";  string sql\_list = string.Empty;  DataTable data = new DataTable();  data.Columns.Add("product\_id", typeof(string));  data.Columns.Add("number", typeof(int));  data.Columns.Add("price", typeof(int));  foreach (var alls in OrderDetail.OrderDetial\_list)  {  if (alls.product\_id != null)  {  DataRow row = data.NewRow();  row["product\_id"] = alls.product\_id;  row["number"] = alls.number;  row["price"] = alls.price;  data.Rows.Add(row);  }  }  if (OrderDetail.name != null && OrderDetail.factory\_name != null)  {  try  {  //EXEC InsertOrdersProcedure @Orders = @orderDetails, @name = 'John Doe', @price = 30.50;  sql\_list = $@"exec [API].dbo.[InsertOrdersData]  @Orders=@OrdersDate,  @customer\_name='{OrderDetail.name}',  @factory\_name='{OrderDetail.factory\_name}'";  return\_list= class1.SQL\_exce\_spectail(sql\_list, data, OrderDetail.name,OrderDetail.factory\_name);  } catch (Exception ex)  {  return\_list = ex.Message;  }  }  else  {  return\_list = "值有錯誤";  }  return return\_list;  } | | **SQL預存程式FUNCTION** | | public string SQL\_exce\_spectail(string sqlstring, DataTable data, string name, string factory\_name)  {  string connectionString = "連資料庫路徑";  SqlDataReader reader;  string datas = "";  try  {  using (SqlConnection conn = new SqlConnection(connectionString))  {  // 使用參數化查詢呼叫預存程序  using (SqlCommand cmd = new SqlCommand("[API].dbo.[InsertOrdersData]", conn))  {  // 指定使用預存程序  cmd.CommandType = CommandType.StoredProcedure;  // 設定表值參數  SqlParameter ordersParam = cmd.Parameters.AddWithValue("@Orders", data);  ordersParam.SqlDbType = SqlDbType.Structured; // 指定為表值參數  ordersParam.TypeName = "dbo.OrderDetailType2"; // 指定表值參數的型別  // 設定其他參數  cmd.Parameters.AddWithValue("@customer\_name", name);  cmd.Parameters.AddWithValue("@factory\_name", factory\_name);  // 打開連接  conn.Open();  // 執行命令  //cmd.ExecuteNonQuery();  reader = cmd.ExecuteReader();  }  }  }  catch (Exception ex)  {  datas = ex.Message;  }  return datas;  } | |