**基本命名原則（Naming Policy）**

1.   不論是何種物件（類別、屬性、方法、事件、函數、委派或其他物件等），只要是非專屬於迴圈（loops）中使用的，一律採明確名稱方式命名，其名稱需要明白的表明用途，非必要不可使用無意義的名稱。

2.   若非必要，不得使用匈牙利命名法（Hungarian notation），例如lngAmount、iDataCount、szCompanyName等，而需以明確用途的命名方式處理。

3.   名稱排列以單字或單詞為原則，盡量不要超過三個單字（word），例如MyName、StockQuantity、ArticleID等，但若業界或慣例中有相對應的縮寫時，可以使用縮寫。例如Quantity（數量）可以縮為Qty，Amount（金額）可縮寫為Amt等。

4.   不限制一定要使用Pascal命名法，但命名要可以讓開發或維護的人員很快的就了解其意義，若物件仍難以直接由名稱了解時，則必須要在宣告處加上註解以協助閱讀。

5.   命名物件時，除非特定場合，不要使用底線（underline，”\_”）來連接兩個字彙。

說明：

使用底線的話，在非 IDE 工具的場合中，容易會因為漏 key 而發生錯誤。

且如果單字多的話，底線會變的很多，可讀性不會比沒有好。

**類別命名與開發原則（Naming and Development Policy for Class）**

1.   一般類別（General Class）應依實際代表的角色來命名，例如Product代表產品；Customer代表客戶；Ad或Advertise代表廣告等。抽象類別（Abstract Class）與一般類別的命名相同。

2.   集合類別（Collection Class）除了依一般類別命名外，需要在字尾加上Collection，例如ProductCollection代表產品集合；CustomerCollection代表客戶集合。

3.   基底類別（Base Class）除了依一般類別命名外，需要在字尾加上Base，例如ProductBase代表產品的基底類別。

4.   類別不使用任何前綴字元（像是”C”）來標示。

說明：

只要有夠清楚的規範定義的話，類別前面加上 C 反而會變成一種累贅。

不過由於介面和類別性質很像，所以介面仍保留 "I" 這個前綴字。

5.   在類別中的私有成員變數（private member variables），命名時需在名稱前加入底線識別，例如 \_customerID、\_isPaid等，並且在使用時要透過this（或Me）來存取。

6.   私有成員變數必須要以屬性（Property）方式開放給外部存取，不可以單獨的宣告為public。

7.   類別中的方法（methods）名稱應以它要執行的動作來命名，可以是動詞或是動作類的名詞。例如Copy()、Create()、FindArticle()或是UpdateAuthorization()等等，若動作的對象是複數時（例如回傳一份清單或是預期會多於一個的物件），則應以對象的複數方式來表示之。 例如GetProducts()、MoveFiles()或是LookupCustomerList()。

8.   方法應以執行單一動作（single action）為原則，除非必要，不得用會跨越不同類別的方式來發展方法。

9.   針對提取資料的方法，建議以Get開頭來命名提取單一資料的方法，以Lookup開頭命名提取多筆資料的方法，以Search或Find開頭命名搜尋資料的方法。

10. 類別中的事件（events）名稱儘量以時態表示法來命名，像是Deleted、Clicked、Changing、Loaded、PreRender這類的詞彙。

11. 若類別只是用來裝載資料（data-only class），則除了必要的資料物件外，應額外加入針對資料正確性檢查的程式碼或規則，以確保資料的正確性。

12. 位於抽象類別的方法，應該是以其衍生類別必須要實作的方法為主，而基底類別或一般類別若有預期其衍生類別會改變原有方法的行為時，請務必將它宣告為虛擬方法（virtual method）。

**介面命名與開發原則（Naming and Development Policy for Interface）**

1.   介面（interface）應作為定義系統的執行規範來使用，像是模組或元件之介接規範、資料規範或是方法實作規範等。

2.   介面應以前綴字”I”來表示，像是IHostingModule、IDataFactory、IConnectionProvider等。

3.   介面所定義的任何屬性、方法或事件命名與類別相同。

**工具類別命名與開發原則（Naming and Development Policy for Utility Class）**

1.   工具類別（Utility Class）命名應以工具名稱、工具性質或輔助性質方式命名，例如FileUtility或是HttpUtility等。

2.   工具類別中的方法應是以static方式宣告，以達到多物件共用的目的，其方法的命名與類別的方法相同，基本原則亦相同。

3.   工具類別應不會保留呼叫端物件的任何狀態，也不會保留其工作本身的狀態，以確保工具類別和實際處理的類別間的低藕合性。

**全域物件開發原則（Development Policy for Global Objects）**

1.   全域物件（Global Objects）若非必要應避免使用，只有在特定情況下才可以運用，並且要配合ASP.NET（或PHP）中的應用程式層次（application level）物件一併使用。

2.   若要使用全域物件，則應建立一完善的控管機制以處理生成（instantiate）、存取與釋放（release）等工作，以避免因控管失誤造成的應用程式穩定性問題。

**列舉物件的使用（Usage of enumerations）**

1.   列舉物件（enumeration）是相當有用的值域限制工具，可以避免儲存或使用到錯誤範圍的值，可在程式中需要的部份（例如檔案類型或是角色類型）中使用，但它只適合用於整數型態的資料。

2.   列舉物件的命名應以列舉本身代表的資料命名，並在字尾加上Enum，像是ProductTypeEnum、RoleTypeEnum或是FileStateEnum等等。

**常數的使用（Usage of constants）**

1.  常數（constants）可以包裝在工具類別或全域類別中，以指示特定的數值，其命名應以常數本身代表的意義為主，例如RoleTypeAdministrator = 0x00、RoleTypeUser = 0x01等。

**命名空間原則（Namespace Policy）**

1.   若開發的是基礎類別庫（Framework）中的組件時，其命名空間必須要以指定的根命名空間開始，然後以元件的大類（應用範圍）來命名，接著再以元件的小類（或專案名稱）給定。

2.   若開發的是專案，則請以組織名稱作為根命名空間，再依Framework命名階層為藍本來命名專案的命名空間。

**控制項使用的命名原則（Naming Policy for Controls）**

1.   命名控制項變數時，原則以業界常用的命名縮寫為慣例，例如：

|  |  |  |
| --- | --- | --- |
| **Web Control** | **縮寫** | **範例** |
| Label | label | labelMyName |
| TextBox | txt | txtMyName |
| Button | cmd | cmdOK |
| LinkButton | lcmd | lcmdEdit |
| ImgeButton | imgcmd | imgcmdOK |
| HyperLink | link | linkSite |
| DropDownList | cbo | cboGender |
| ListBox | lst | lstFile |
| CheckBox | chk | chkFilter |
| CheckBoxList | chkl | chklItems |
| RadioButton | opt | optByDate |
| RadioButtonList | optl | optlFilter |
| Image | img | imgPhoto |
| ImageMap | imgmap | imgmapMap |
| Table | tb | tbResult |
| BulletedList | bl | blBulletin |
| HiddenField | hidden | hiddenRate |
| Literal | lit | litScriptSpace |
| Calendar | cal | calBirthday |
| AdRotator | ad | adTop |
| FileUpload | fileup | fileupVersion |
| Wizard | wiz | wizTask |
| Xml | xml | xmlData |
| MultiView | mv | mvTask |
| Panel | panel | panelDataView |
| PlaceHolder | ph | phControls |
| View | view | viewControl |
| Substitution | substi | substiDate |
| Localize | loc | locPage |

|  |  |  |
| --- | --- | --- |
| **ADO.NET控制項** | **縮寫** | **範例** |
| GridView | gv | gvEmployee |
| DataList | dl | dlPhotoAlbum |
| DetailView | dtv | dtvProfile |
| FormView | fv | fvForm |
| Repeater | rep | repDataView |
| SqlDataSrouce | sqlds | sqldsEmployee |
| AccessDataSource | mdbds | mdbdsProducts |
| ObjectDataSource | objds | objdsWS |
| XmlDataSource | xmlds | xmldsProfile |
| SiteMapDataSource | sitemapds | sitemapdsMySite |

**程式碼編寫原則（Coding Policy）**

1.   程式碼可適當使用#region…#endregion來表示不同功用的程式區段。

2.   多使用註解（comments）來標示變數、方法、參數等的用途，也可以幫助使用Visual Studio的XML文件自動產生功能。

3.   程式段落應使用縮排（quote）來提升可讀性。

**錯誤處理基本原則（Error Handling Policy）**

1.   在可能會發出錯誤的區塊，使用try…catch…來捕捉錯誤。

2.   使用不同的catch區塊來處理不同的錯誤。

3.   可組織可能會發生的錯誤類型，回報具親和力的錯誤訊息，或由另一個工具類別來處理錯誤訊息輸出的工作。

**資料庫與資料庫物件命名原則（Naming Policy for Database and Database Objects）**

1.   資料庫命名若是獨立專案，應使用組織名稱來命名，在部份特殊案例中，則可使用不同的命名方式處理。

2.   資料表命名應以名詞，複數性質的字彙或詞彙命名，例如Products代表產品資料表，Customers代表客戶資料表，Orders代表訂單資料表等。

3.   若非需求規格中明訂，任何資料庫與資料物件（表格、檢視表、預存程序、觸發程序或使用者函數等）都不得使用中文命名。

4.   資料庫物件可參考下列前綴字元來命名，或是沿用類別方法的原則，但資料庫物件可以適量的使用底線來明確標示其所屬的資料表：

檢視表：v或vw。

預存程序：usp。

使用者函數：udf或fn。

觸發程序：tr或tp。

5.  資料庫物件應以代表其用途的方式命名，例如檢視表可以其顯露的資料性質命名（如vwSalePersons\_GroupByAmounts），預存程序以動作來命名（如uspOrders\_UpdateOrder），使用者函數則可依本身的功能來命名等。

6.  資料表的欄位（column），請依實際儲存的資料性質來命名，例如PhotoID代表相片代碼、Caption代表名稱、PhotoData代表相片資料、PhotoType代表相片類型等。

**資料庫與資料庫物件設計與存取原則（Design and Access Policy for Database and Database Objects）**

1.   資料表結構應避免直接開放給外部應用程式，而應該以檢視表（View）的方式來顯露結構，並以預存程序（Stored Procedure）來更新資料庫。外部程式碼僅可存取檢視表、預存程序或使用者函數（user defined function）等物件，而不應直接存取資料表結構。

說明：

檢視表或預存程序都可以用來隱藏資料表結構，不過：

預存程序適合用來封裝規則與邏輯。

檢視表適合用來組織資料結構。

2.  只要是針對資料表寫入的作業（即新增、刪除與更新三種作業），都必須要使用交易（Transaction）來處理，不論是在預存程序中或是由外部程式來執行皆然，並且將作業記錄在稽核檔（獨立的資料表）中備查。

說明：

若系統沒有嚴謹到必須要用稽核檔來監控資料異動的話，在只輸入一筆資料的情況下可以免用交易模式。

3.   外部應用程式針對資料庫的作業，絕對禁止使用字串聯結（string append）的方式來組建帶有參數的SQL指令，並且也不得直接存取資料表，而應該使用檢視表來取得資料，並用預存程序來寫入資料。任何要傳遞進資料庫執行的參數，都必須要用參數物件（Parameter Objects）來包裝以確保資料庫的安全。

說明：

若有絕對的把握，在使用 Dynamic SQL 時不會被人抓到漏洞的話，適當使用字串聯結是可以的，但代入參數一定要求使用參數方式，不可使用字串聯結，以防止 SQL Injection 的發生機率（只要用的是參數，那 SQL Injection 的機率就幾乎等於零了）。

4.   外部應用程式不得使用高權限的帳戶（例如sa或root）存取資料庫，而應另設帳戶存取，此原則對在內部開發之專案或部署至用戶環境時皆然。

5.   除了檢視表以外，外部應用程式只可在必要的時候（或資料量可控在一定範圍內）使用無WHERE過濾指令的SQL指令來撈取資料，若需要分頁時，應使用分頁查詢技巧來查詢資料，不應只依賴前端的自動分頁功能。

6.   資料表欄位的型別若是文字，除對長度有要求或有固定長度需求的資料可用nchar型別外，原則上應使用nvarchar型別。

7.   若欄位的資料型別是BLOB（大型二進位物件）時，SQL Server 2000需使用image（二進位資料）或是ntext（大型文字資料），SQL Server 2005以後版本則使用varbinary(max)或是nvarchar(max)作為型別，Oracle 使用 CLOB 或 BLOB。

8.   若資料表中會儲存密碼（Password）時，應盡可能使用強度較高的雜湊演算法（Hashing Algorithm）處理（.NET Framework至少要使用SHA256以上），並儲存雜湊後的字串，且在應用程式中加入重設密碼的功能。