manual para desarrollo de aplicacions en avantgarde studio

# introduccion

Este documento tiene el objetivo de ser una guía para el desarrollo de aplicaciones con la metodología de la empresa de Avantgarde Studio. Su propósito es servir a los empleados como guía para desarrollar aplicaciones dentro de la Empresa, así como para entender la estructura de aplicaciones existentes dentro de la empresa.

# Creación de una base de datos

La base de datos es la parte principal en el desarrollo de la aplicación y por eso hacemos especial enfoque en el desarrollo de la misma. Normalmente se tiende a menospreciar o relegar a un segundo plano el desarrollo de la misma por parecernos menos interesante que el desarrollo de la interface o de la lógica de la Aplicación, pero, en realidad es el centro de la aplicación y debe ponerse especial atención al desarrollo de la misma.

Para agilizar el proceso de aprendizaje vamos a descargar una base de datos de ejemplo que luego podremos modificar para su uso dentro de la aplicación. Utilizaremos la base de datos Northwind que viene ya con contenido y es un buen ejemplo de una aplicación de facturación.

Podemos descargarla desde la URL “https://binaryworld.net/blogs/northwind-database-creation-script-for-sql-server/” y desde allí utilizar el enlace “Install-Northwind-Script-SQL-Server” que nos descarga un archivo ZIP con el script para generar la base de datos.

Desde el SQL Server Management Studio abrimos el script desde la base de datos Master de nuestro servidor Local.

Antes de ejecutar el script debemos comentar las líneas 24 y 25 donde dice:

exec sp\_dboption 'Northwind','trunc. log on chkpt.','true'

exec sp\_dboption 'Northwind','select into/bulkcopy','true'

Ejecutamos el Script para que cree la base de datos. A continuación en el Explorador de Objetos refrescamos el Nodo de Bases de Datos para ver que efectivamente está la base de datos Northwind.

Probamos que está creada correctamente creando una nueva consulta y ejecutando:

SELECT \* FROM Orders O INNER JOIN [Order Details] D ON O.OrderID=D.OrderID INNER JOIN Products P on P.ProductID = D.ProductID

Si está todo correcto debemos empezar a hacer modificaciones en la base de datos para ajustarla a la nomenclatura de Avantgarde Studio. Es importante esta adaptación para que todos trabajemos de la misma manera, evitemos errores del tipo CASE SENSITIVE y encontremos enlaces de datos mas fácilmente.

Lo primero que vemos es que la tabla “Order Details” tiene un espacio de separación entre las dos palabras que demos corregir eliminando ese espacio pues las tablas no deben tener espacios para evitar tener que utilizar corchetes al a hora de nombrarla en las consultas.

Para realizar el cambio usamos el comando:

exec sp\_rename 'dbo.[Order Details]', 'OrderDetails'

Una vez ejecutado realizaremos la consulta anterior epr en este caso:

SELECT \* FROM Orders O INNER JOIN OrderDetails D ON O.OrderID=D.OrderID INNER JOIN Products P on P.ProductID = D.ProductID

Otro cambio que debemos tener en cuenta es que nuestras tablas deben tener el prefijo tbl y por ello debemos renombrar todas las tablas utilizando el mismo. Como ejemplo volvemos a renombrar la tabla anterior:

exec sp\_rename 'OrderDetails', 'tblOrderDetails'

y luego haremos lo mismo con todas las demás, tras lo cual, la consulta debería quedar como:

SELECT \* FROM tblOrders O INNER JOIN tblOrderDetails D ON O.OrderID=D.OrderID INNER JOIN tblProducts P on P.ProductID = D.ProductID

# T-SQL y XML

Hemos visto que la consulta anterior genera el Dataset típico pero en Avantgarde Studio utilizamos XML constantemente para luego convertirlo en la interface de la aplicación mediante XSLT.

Si quisiéramos que los datos nos sean mostrados en formato XML alteraríamos la consulta anterior de la siguiente manera.

SELECT \* FROM tblOrders O INNER JOIN tblOrderDetails D ON O.OrderID=D.OrderID INNER JOIN tblProducts P on P.ProductID = D.ProductID

FOR XML PATH ('Order'), ROOT ('Orders')

Donde “Order” define cada nodo particular y “Orders” es la raíz del documento XML.

<Orders>

<Order>

<OrderID>10248</OrderID>

<CustomerID>VINET</CustomerID>

<EmployeeID>5</EmployeeID>

<OrderDate>1996-07-04T00:00:00</OrderDate>

<RequiredDate>1996-08-01T00:00:00</RequiredDate>

<ShippedDate>1996-07-16T00:00:00</ShippedDate>

<ShipVia>3</ShipVia>

<Freight>32.3800</Freight>

<ShipName>Vins et alcools Chevalier</ShipName>

<ShipAddress>59 rue de l'Abbaye</ShipAddress>

<ShipCity>Reims</ShipCity>

<ShipPostalCode>51100</ShipPostalCode>

<ShipCountry>France</ShipCountry>

<OrderID>10248</OrderID>

<ProductID>11</ProductID>

<UnitPrice>14.0000</UnitPrice>

<Quantity>12</Quantity>

<Discount>0.0000000e+000</Discount>

<ProductID>11</ProductID>

<ProductName>Queso Cabrales</ProductName>

<SupplierID>5</SupplierID>

<CategoryID>4</CategoryID>

<QuantityPerUnit>1 kg pkg.</QuantityPerUnit>

<UnitPrice>21.0000</UnitPrice>

<UnitsInStock>22</UnitsInStock>

<UnitsOnOrder>30</UnitsOnOrder>

<ReorderLevel>30</ReorderLevel>

<Discontinued>0</Discontinued>

</Order>

</Orders>

Para estructurarlo mejor, cambiamos la consulta y así nos dará un documento mas estructurado y fácil de leer.

SELECT \*

,(SELECT \*

,(SELECT ProductName FROM tblProducts P

WHERE P.ProductID = D.ProductID

FOR XML PATH ('Product'),TYPE)

FROM tblOrderDetails D

WHERE O.OrderID=D.OrderID

FOR XML PATH ('Detail'),TYPE) AS 'Details'

FROM tblOrders O

FOR XML PATH ('Order'), ROOT ('Orders')

Resultado:

<Orders>

<Order>

<OrderID>10248</OrderID>

<CustomerID>VINET</CustomerID>

<EmployeeID>5</EmployeeID>

<OrderDate>1996-07-04T00:00:00</OrderDate>

<RequiredDate>1996-08-01T00:00:00</RequiredDate>

<ShippedDate>1996-07-16T00:00:00</ShippedDate>

<ShipVia>3</ShipVia>

<Freight>32.3800</Freight>

<ShipName>Vins et alcools Chevalier</ShipName>

<ShipAddress>59 rue de l'Abbaye</ShipAddress>

<ShipCity>Reims</ShipCity>

<ShipPostalCode>51100</ShipPostalCode>

<ShipCountry>France</ShipCountry>

<Details>

<Detail>

<OrderID>10248</OrderID>

<ProductID>11</ProductID>

<UnitPrice>14.0000</UnitPrice>

<Quantity>12</Quantity>

<Discount>0.0000000e+000</Discount>

<Product>

<ProductName>Queso Cabrales</ProductName>

</Product>

</Detail>

<Detail>

<OrderID>10248</OrderID>

<ProductID>42</ProductID>

<UnitPrice>9.8000</UnitPrice>

<Quantity>10</Quantity>

<Discount>0.0000000e+000</Discount>

<Product>

<ProductName>Singaporean Hokkien Fried Mee</ProductName>

</Product>

</Detail>

Podemos usar una variable para filtrar sólo el pedido que nos interesa:

DECLARE @OrderID INT = 10249

SELECT \*

,(SELECT \*

,(SELECT ProductName FROM tblProducts P

WHERE P.ProductID = D.ProductID

FOR XML PATH ('Product'),TYPE)

FROM tblOrderDetails D

WHERE O.OrderID=D.OrderID

FOR XML PATH ('Detail'),TYPE) AS 'Details'

FROM tblOrders O WHERE O.OrderID = @OrderID

FOR XML PATH ('Order'), ROOT ('Orders')

Hemos utilizado la variable “@OrderID” de tipo entero para poder sacar sólo el pedido que deseamos.

Ahora podemos crear un Procedimiento almacenado basado en la consulta anterior:

CREATE PROCEDURE OrderGet

@OrderID INT = 0

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

SELECT \*

,(SELECT \*

,(SELECT ProductName FROM tblProducts P

WHERE P.ProductID = D.ProductID

FOR XML PATH ('Product'),TYPE)

FROM tblOrderDetails D

WHERE O.OrderID=D.OrderID

FOR XML PATH ('Detail'),TYPE) AS 'Details'

FROM tblOrders O WHERE O.OrderID = @OrderID

FOR XML PATH ('Order'), ROOT ('Orders')

END

GO

Y luego podemos probar su resultado ejecutando:

exec OrderGet 10249

Vemos que el resultado es el mismo. Podemos ir mas allá modificando la línea del WHERE principal

FROM tblOrders O WHERE O.OrderID = @OrderID por FROM tblOrders O WHERE O.OrderID = @OrderID OR @OrderID = 0

Y ejecutar ahora tan solo:

exec OrderGet 0

Para que todos los documentos tengan una raíz única debemos crear una tabla unificadora que llamaremos tblSite.

CREATE TABLE tblSite(

SiteID int IDENTITY NOT NULL,

SiteName varchar(50) Not Null

)

INSERT tblSite(SiteName)VALUES('Northwind')

Luego debemos crear el campo Site\_ID en todas las demás tablas para vincularlas con su raíz. Por ejemplo en la tabla tblOrders:

ALTER TABLE tblOrders ADD Site\_ID INT NOT NULL DEFAULT 1

Haremos lo mismo con las demás tablas. Observar que el campo Site\_ID al ser de una tabla que hereda ese índice lleva un UNDERSCORE entre el nombre y el ID. Lo utilizamos para abreviarlo a la hora de crear los where en las consultas.

Modificamos entonces el Procedimiento almacenado:

ALTER PROCEDURE [dbo].[OrderGet]

@OrderID INT = 0

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

SELECT \*

,(SELECT \*

,(SELECT \*

,(SELECT ProductName FROM tblProducts P

WHERE P.ProductID = D.ProductID

FOR XML PATH ('Product'),TYPE)

FROM tblOrderDetails D

WHERE O.OrderID=D.OrderID

FOR XML PATH ('Detail'),TYPE) AS 'Details'

FROM tblOrders O WHERE O.Site\_ID = tblSite.SiteID AND (O.OrderID = @OrderID OR @OrderID = 0)

FOR XML PATH ('Order'), TYPE) AS 'Orders'

FROM tblSite

FOR XML PATH ('Site'), ROOT ('Doc')

END

Y ejecutamos de nuevo:

exec OrderGet 10249

Continuando con el ejemplo creamos un Procedimiento almacenado que nos de los datos y productos de los proveedores:

CREATE PROCEDURE [dbo].[SupplierGet]

@SupplierID INT = 0

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

SELECT \*

,(SELECT \*

,(SELECT \* FROM tblProducts P

WHERE P.SupplierID = S.SupplierID

FOR XML PATH ('Product'),TYPE) as 'Products'

FROM tblSuppliers S WHERE S.Site\_ID = tblSite.SiteID AND (S.SupplierID = @SupplierID OR @SupplierID = 0)

FOR XML PATH ('Supplier'), TYPE) AS 'Suppliers'

FROM tblSite

FOR XML PATH ('Site'), ROOT ('Doc')

END

A continuación creamos las tablas tblInvoice y tblInvoiceDetails a semejanza de tblOrders y tblOrderDetails respectivamente

CREATE TABLE [dbo].[tblInvoices](

InvoiceID int IDENTITY(1,1) NOT NULL,

Customer\_ID nchar(5) NULL,

Employee\_ID int NULL,

OrderDate datetime NULL,

RequiredDate datetime NULL,

ShippedDate datetime NULL,

ShipVia int NULL,

Freight money NULL,

ShipName nvarchar(40) NULL,

ShipAddress nvarchar(60) NULL,

ShipCity nvarchar(15) NULL,

ShipRegion nvarchar(15) NULL,

ShipPostalCode nvarchar(10) NULL,

ShipCountry nvarchar(15) NULL,

Site\_ID int NOT NULL,

)

CREATE TABLE tblInvoiceDetails(

InvoiceDetailID INT IDENTITY,

Invoice\_ID int NOT NULL,

Product\_ID int NOT NULL,

UnitPrice money NOT NULL,

Quantity smallint NOT NULL,

Discount real NOT NULL,

Site\_ID int NOT NULL

)

Hemos añadido un campo IDENTITY a la tabla de detalles para poder identificarlo de manera única. Haremos lo mismo con la tabla de detalle de “Orders”

ALTER TABLE tblOrderDetails ADD OrderDetailID INT IDENTITY

# Aplicación en visual studio

En Visual Studio, preferiblemente Comunity Edition, creamos un nuevo proyecto al que le daremos el nombre de Northwind. Debe ser un proyecto vacío en el que usaremos Webforms. En principio la versión de .NET es indiferente aunque preferiremos la mas avanzada.

Creamos un página ASPX que llamaremos Default.aspx que cuelgue de una Master.

En esta página insertaremos un control XML que utilizaremos mas tarde.

Creamos en la carpeta App\_Data un archivo XML en el copiaremos el resultado de ejecutar:

exec OrderGet 0

Por otro lado crearemos en la carpeta App\_Data otro archivo, esta vez del tipo XSLT.

En la página ASPX anterior utilizaremos estos archivos para realizar la transformación de los datos con el control XML que hemos insertado en la página.

Modificaremos el archivo XSLT para que muestre los datos en una tabla.

Para finalizar usaremos Bootstrap para dar estilo a la página y a la tabla en concreto.