DESIGN

http://www.i3design.co.uk

ConstructorMDRAD

Advanced Software Construction Toolkit for .NET

Optimized for

PETSHOP v3 APPLICATION Part 2

Constructor PetShop Application

Introduction

In **Part 1**, the Microsoft PetShop v3 application was converted as much as possible like-for-like to a Constructor/MDRAD-based Model-Driven application. A relatively small amount of code was changed or introduced and approx. 1,350 lines of code were saved. The database schemas (for both MSPetShop and MSPetShopOrders) were purposefully left unchanged.

Code count comparison

	Microsoft version	Constructor version
Presentation Layer	1,822	1,852
Model and Business Logic Layers	559	701
Data Access Layer	1,538	No user code required
Total Lines of Code	**3,919**	**2,553**

In this, Part 2, we built on the change to a Constructor/MDRAD approach by expanding the Model, updating the database schema and reworking some of the code to take advantage of these changes.

Aims

This example of a 'Constructor' version of the PetShop v3 (for .NET v1 and VS2002/3) was designed to mimic the existing internal operations of the original as much as possible, and demonstrates the use of Constructor/MDRAD to modify a database schema, enhance a UML Model, and make greater use of defined relationship paths to reduce or simplify code. The business logic still follows the same process.

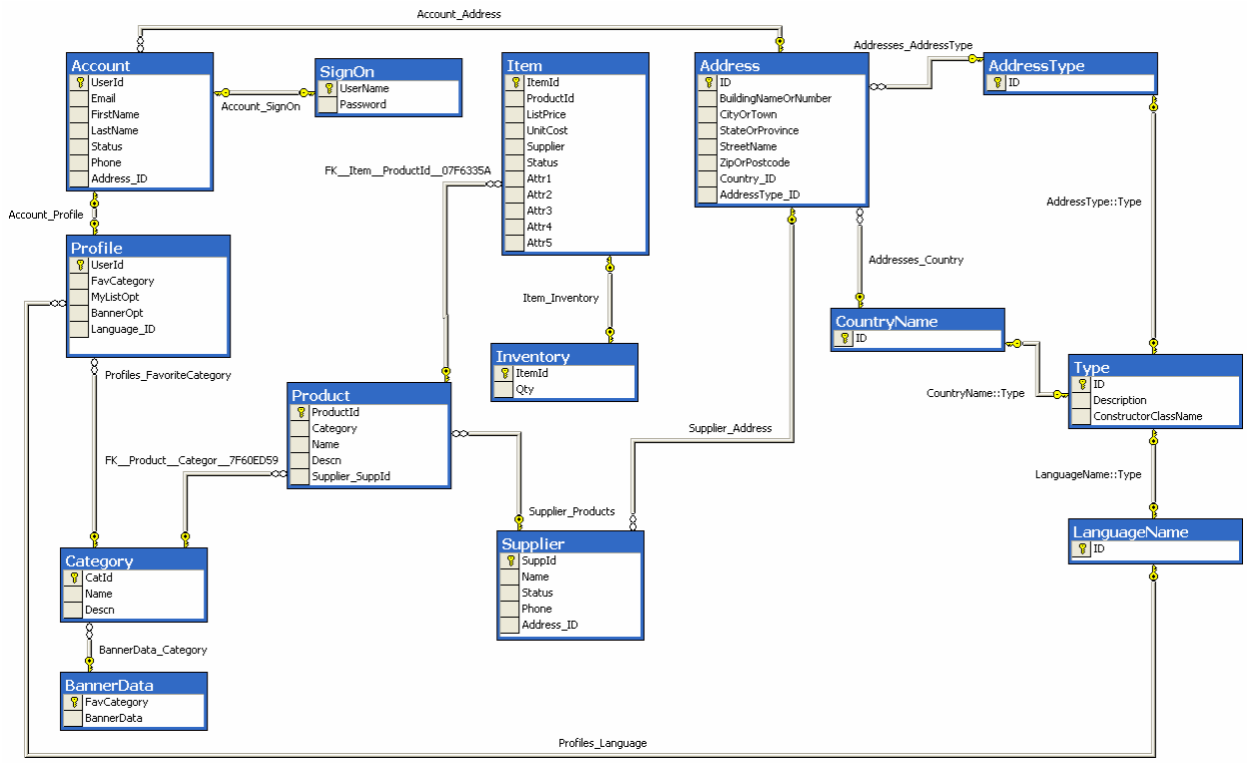
The user interface is unchanged but some of the property names in html within <% %> tags were replaced to match the Attribute names in the Model, and others were left unchanged to demonstrate the fact that an Attribute in a Model can be named differently than the underlying column.

Also demonstrated: Model and Database Inheritance, non-persistent Attributes (extension properties) and use of virtual references.

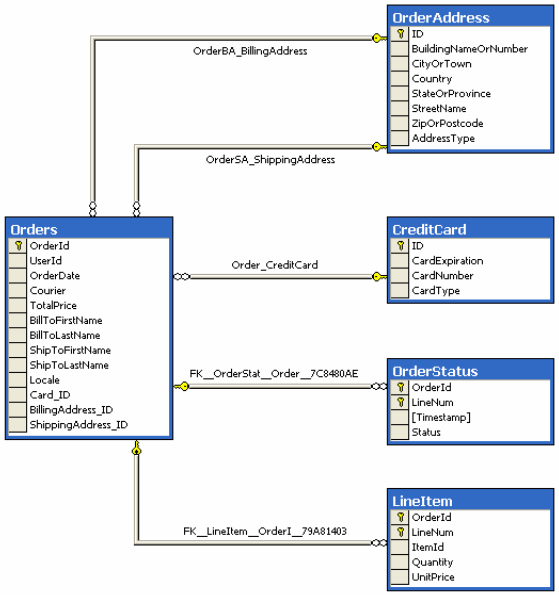
Revised Database Schema

The enhanced PetShop uses two databases with revised schemas as below:

MSPetShop



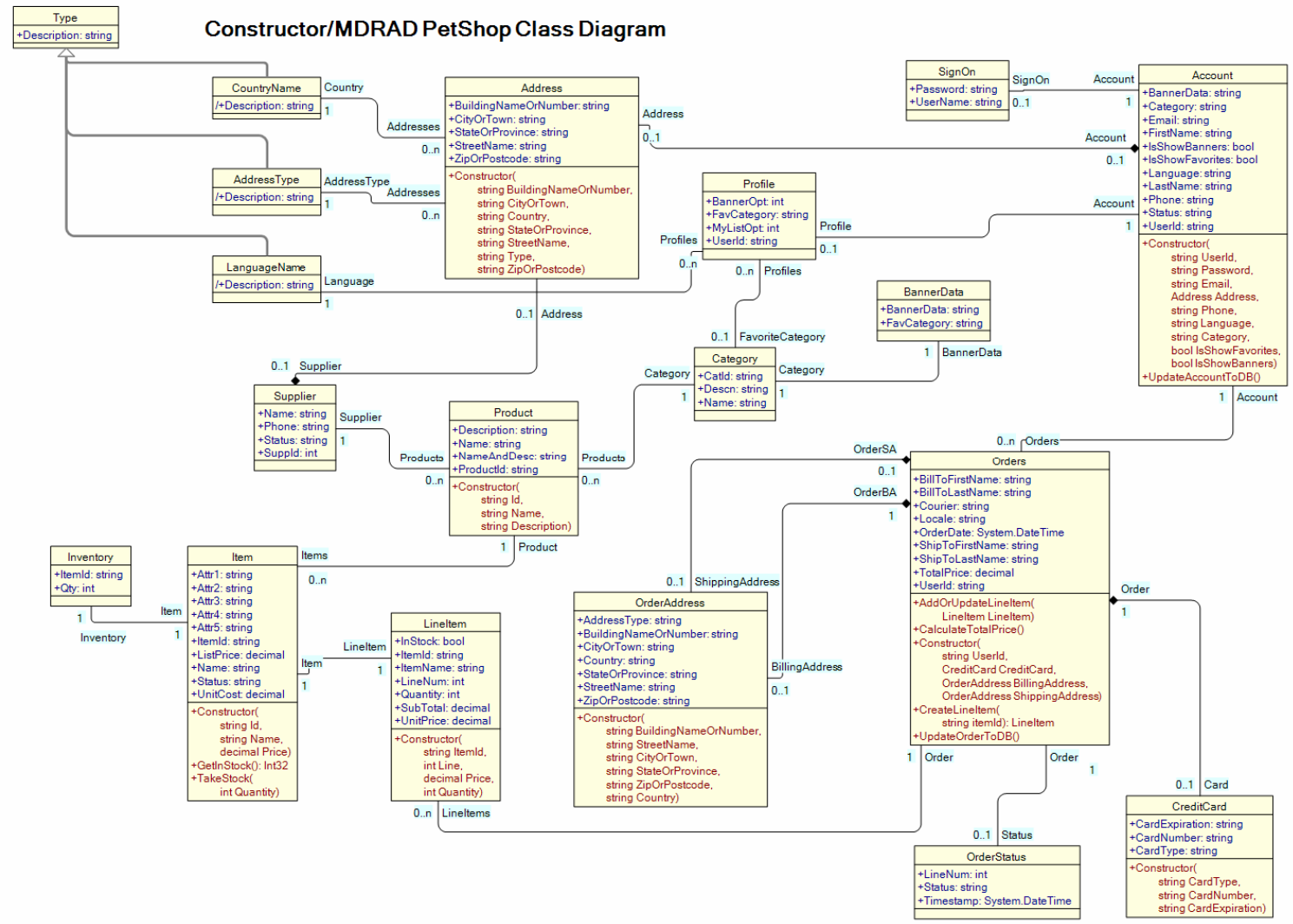
MSPetShopOrders



NOTE that there is no relationship between Account and Orders, because they are in separate databases. BUT in Constructor code you can still navigate from Account to Orders or vice-versa in order to automatically load the related data. This is a virtual relationship in action.

Revised Class Diagram

A UML Class Diagram representing the enhanced Model is shown below. Note the ‘Account/Orders’ link, and the 3 classes ‘derived’ from ‘Type’.



Overview of steps taken to enhance PetShop project

Add Classes and Relationships in the Model

As described in Part 1, the PetShop database schema has some relationships defined, but not as many as there should be. This was probably designed-in (or out) on purpose, to reduce the amount of code that would have had to be added to cater for more collections, additional SQL statements and additional class definitions.

Examples of the need for more code references included:

- When accessing an Account. It may be desirable to access past order information (*Account* should reference *Orders*)
- When defining 'address' objects.
    - The *Orders* table holds data on a 'billing' and a 'shipping' address. There should have been an *OrderAddress* table related to *Orders* and an *Order* would then (via its *Account* reference) have been able to reference and copy the original address detail to *OrderAddress* (in case the original address data changes in future).
    - The *Account* table holds address data.
    - The *Supplier* table holds address data.
    - The suggested *Address* class should have a reference to *Country* (new table and class required – *CountryName*) and to *Type* indicating whether a billing, shipping, account or supplier address (new table and class required – *AddressType*)
- When defining order 'credit card' objects. The *Orders* table holds basic credit card data. There should have been an *CreditCard* table related to *Orders*.
- When checking Inventory (e.g. quantity in stock) for a product Item (*Item* and *LineItem* should reference *Inventory*)
- When retrieving order data. It may be desirable to see the category, product or supplier name for a *LineItem* (*LineItem* should reference *Item*)
- The existing relationship between *Supplier* and *Item* should in fact be one between *Supplier* and *Product*.
- When determining the user's preferences (in *Profile*). These should be accessed using a reference from *Account*. In addition to this, *Profile* should have references to *LanguagePreference* (new table and class required – *LanguageName*) and *FavoriteCategory* (using *Category*).
- For signing in purposes. Either *SignOn* should be amalgamated into *Account*, or it should be referenced by *Account*.
- For general usage, a 'Helper' class (non-persistent) is required to provide access to general-purpose methods to create objects when there is no appropriate object to delegate to in order to obtain a result or perform a process.

These new Classes and References were added to the Constructor Model. New Classes were created first, then all the new relationships added by dragging relevant referencing classes to the classes to be referenced.

NOTE
When an 'Association' (a relationship between two classes) is defined in a Constructor Model, a new Data Column definition is added to the mapped referenced table (to hold the Foreign Key value of the referencing table). Constructor does not assume that an existing column of the same name as the Primary Key column name of the referencing table should be used. So, before synchronizing the database schema to the amended Model, you need to check that the appropriate *referencing* and *referenced* column names are as they should be, and change them if not. Typically you will need to change the referenced column name to an already existing column, and then delete the column added by Constructor. You may also need to use the 'swap' option to determine which table a Foreign Key column should go in for tables related on a '1 to 1' basis. **The data types and facets of each column in a relationship should match (for example FavCategory in Profile did not match CatId in Category).**

Classes Added to Model

Class Name	Database Mapping	Description
Address	MSPetShop	To hold address data for Account and Supplier Related to Account Related to AddressType Related to CountryName Related to Supplier
AddressType	MSPetShop	To hold ID of Description (from Type) when relating AddressType to Address Related to Address
CountryName	MSPetShop	To hold ID of Description (from Type) when relating Country to Address Related to Address
CreditCard	MSPetShopOrders	To hold payment card details for an Order Related to Orders
Helper	N/a NON-PERSISTENT	Provides several globally accessible methods to retrieve Constructor Objects/ObjectSets, and to perform processes such as Sign-in.
LanguageName	MSPetShop	To hold ID of Description (from Type) when relating Language to Profile Related to Profile
OrderAddress	MSPetShopOrders	To hold a copy of the Address used at time of Order Related to Orders
Type	MSPetShop	To hold basic descriptions to be used as 'lookup' records Base class for AddressType, CountryName, LanguageName

Classes Related in Model

Class Name	Database Mapping	Description
Account	MSPetShop	To relate to Address To relate to Orders To relate to Profile To relate to SignOn
BannerData	MSPetShop	To relate to Category
Category	MSPetShop	To relate to BannerData To relate to Profile (already related to Product)
Inventory	MSPetShop	To relate to Item
Item	MSPetShop	To relate to Inventory To relate to LineItem To relate to Product
LineItem	MSPetShopOrders	To relate to Item (already related to Orders)
Orders	MSPetShopOrders	To relate to Account To relate to OrderAddress To relate to CreditCard (already related to LineItem) (already related to OrderStatus)
Product	MSPetShop	To relate to Supplier (already related to Category) (already related to Item)
Profile	MSPetShop	To relate to Account To relate to Category To relate to LanguageName
SignOn	MSPetShop	To relate to Account
Supplier	MSPetShop	To relate to Address To relate to Product

Attributes Removed in Model

Class Name	Database Mapping	Attribute Name
Account	MSPetShop	Addr1
		Addr2
		City
		Country
		State
		Zip
Orders	MSPetShopOrders	BillAddr1
		BillAddr2
		BillCity
		BillCountry
		BillState
		BillZip
		ShipAddr1
		ShipAddr2
		ShipCity
		ShipCountry
		ShipState
		ShipZip
		CardType
		CreditCard
		ExprDate
Supplier	MSPetShop	Addr1
		Addr2
		City
		State
		Zip

Associations Removed in Model

Database Mapping	Association Name
MSPetShop	FK__Item__Supplier__08EA5793 (References between Item and Supplier)

Add, Modify and Remove Data in the Databases

Three steps needed to be taken to modify the PetShop databases to suit the changes made in the Model.

*NOTE – WE USED DUPLICATES OF THE ORIGINAL DATABASES CALLED **MSPETSHOPNEW** AND **MSPETSHOPORDERSNEW.***

1. Additional tables, columns and relationships to be defined in the DB schemas.

   This step could be performed by the DB schema synchronize feature of Constructor/MDRAD. See the later listing.

2. Data to be added to new tables, and moved between existing and new tables, in addition to Foreign Key values (such as for Country in Address) to be set for existing data.

   This step required SQL statements to be executed manually (see later listing).

3. Now-redundant columns to be removed from database schema.

   This step required SQL statements to be executed manually (see later listing).

Relationships Removed

Relation Name	Database	Description
FK__Item__Supplier__08EA5793	MSPetShop	Relation between *Item* and *Supplier* No longer required *Supplier* now related to *Product* (*Product* is related to *Item*)

Data Created in Tables

Table Name	Database	Description
AddressType	MSPetShop	Types of Address i.e. Billing, Shipping, Account, Supplier
CountryName	MSPetShop	Names of Countries e.g. United States, United Kingdom, Japan
LanguageName	MSPetShop	Language names, e.g. English, Japanese, French
Type	MSPetShop	All descriptions, identified by derived class name

Columns Removed

Table Name	Database	Column Name	Description
Item (data deleted)	MSPetShop	Supplier	Foreign Key of Supplier record (Relationship also dropped from DB)
Account (data moved to Address table)	MSPetShop	Addr1	
		Addr2	
		City	
		Country	
		State	
		Zip	
Orders (data moved to OrderAddress table and CreditCard table)	MSPetShopOrders	BillAddr1	
		BillAddr2	
		BillCity	
		BillCountry	
		BillState	
		BillZip	
		ShipAddr1	
		ShipAddr2	
		ShipCity	
		ShipCountry	
		ShipState	
		ShipZip	
		CardType	
		CreditCard	
		ExprDate	
Supplier (data moved to Address table)	MSPetShop	Addr1	
		Addr2	
		City	
		State	
		Zip	

APPENDIX A – TSQL LISTINGS

Additional tables, columns and relationships defined in the DB schemas.

This step performed by the DB schema synchronize feature of Constructor/MDRAD.

ACTION TSQL for MSPetShop

```
USE [MSPetShopNew]

ALTER TABLE dbo.[Account] WITH NOCHECK
ADD
    [Address_ID] int NULL

CREATE TABLE dbo.[Address]
(
    [ID] int IDENTITY (1 , 1) NOT NULL ,
    [BuildingNameOrNumber] varchar (20) NULL ,
    [CityOrTown] varchar (30) NULL ,
    [StateOrProvince] varchar (20) NULL ,
    [StreetName] varchar (40) NULL ,
    [ZipOrPostcode] varchar (10) NULL ,
    [Country_ID] int NULL ,
    [AddressType_ID] int NULL
)

ALTER TABLE dbo.[Address] WITH NOCHECK
    ADD CONSTRAINT [Address_pk]
    PRIMARY KEY ([ID])

CREATE TABLE dbo.[AddressType]
(
    [ID] int NOT NULL
)

ALTER TABLE dbo.[AddressType] WITH NOCHECK
    ADD CONSTRAINT [AddressType_pk]
    PRIMARY KEY ([ID])

CREATE TABLE dbo.[CountryName]
(
    [ID] int NOT NULL
)
```

UNDO TSQL for MSPetShop

```
ALTER TABLE dbo.[Supplier]
    DROP CONSTRAINT [Supplier_Address]

ALTER TABLE dbo.[SignOn]
    DROP CONSTRAINT [Account_SignOn]

ALTER TABLE dbo.[Profile]
    DROP CONSTRAINT [Profiles_Language]

ALTER TABLE dbo.[Profile]
    DROP CONSTRAINT [Profiles_FavoriteCategory]

ALTER TABLE dbo.[Profile]
    DROP CONSTRAINT [Account_Profile]

ALTER TABLE dbo.[Product]
    DROP CONSTRAINT [Supplier_Products]

ALTER TABLE dbo.[LanguageName]
    DROP CONSTRAINT [LanguageName::Type]

ALTER TABLE dbo.[Item]
    DROP CONSTRAINT [Item_Inventory]

ALTER TABLE dbo.[CountryName]
    DROP CONSTRAINT [CountryName::Type]

ALTER TABLE dbo.[Category]
    DROP CONSTRAINT [BannerData_Category]

ALTER TABLE dbo.[AddressType]
    DROP CONSTRAINT [AddressType::Type]

ALTER TABLE dbo.[Address]
    DROP CONSTRAINT [Addresses_Country]

ALTER TABLE dbo.[Address]
    DROP CONSTRAINT [Addresses_AddressType]
```

```
ALTER TABLE dbo.[CountryName] WITH NOCHECK
    ADD CONSTRAINT [CountryName_pk]
    PRIMARY KEY ([ID])

CREATE TABLE dbo.[LanguageName]
(
    [ID] int NOT NULL
)

ALTER TABLE dbo.[LanguageName] WITH NOCHECK
    ADD CONSTRAINT [LanguageName_pk]
    PRIMARY KEY ([ID])

ALTER TABLE dbo.[Product] WITH NOCHECK
ADD
    [Supplier_SuppId] int NULL

ALTER TABLE dbo.[Profile] WITH NOCHECK
ADD
    [Language_ID] int NULL ,
   [FavoriteCategory_CatId] varchar (10) NULL

ALTER TABLE dbo.[Supplier] WITH NOCHECK
ADD
    [Address_ID] int NULL

CREATE TABLE dbo.[Type]
(
    [ID] int IDENTITY (1 , 1) NOT NULL ,
    [Description] varchar (255) NULL ,
   [ConstructorClassName] varchar (100) NULL
)

ALTER TABLE dbo.[Type] WITH NOCHECK
    ADD CONSTRAINT [Type_pk]
    PRIMARY KEY ([ID])

ALTER TABLE dbo.[Account] WITH NOCHECK ADD
    CONSTRAINT [Account_Address]
    FOREIGN KEY ([Address_ID])
    REFERENCES [Address] ([ID])

ALTER TABLE dbo.[Address] WITH NOCHECK ADD
    CONSTRAINT [Addresses_AddressType]
    FOREIGN KEY ([AddressType_ID])
    REFERENCES [AddressType] ([ID]),
    CONSTRAINT [Addresses_Country]
    FOREIGN KEY ([Country_ID])
    REFERENCES [CountryName] ([ID])
```

```
ALTER TABLE dbo.[Account]
    DROP CONSTRAINT [Account_Address]

ALTER TABLE dbo.[Type]
    DROP CONSTRAINT [Type_pk]

DROP TABLE dbo.[Type]

ALTER TABLE dbo.[Supplier]
    DROP COLUMN [Address_ID]

ALTER TABLE dbo.[Profile]
   DROP COLUMN [Language_ID],[FavoriteCategory_CatId]

ALTER TABLE dbo.[Product]
    DROP COLUMN [Supplier_SuppId]

ALTER TABLE dbo.[LanguageName]
    DROP CONSTRAINT [LanguageName_pk]

DROP TABLE dbo.[LanguageName]

ALTER TABLE dbo.[CountryName]
    DROP CONSTRAINT [CountryName_pk]

DROP TABLE dbo.[CountryName]

ALTER TABLE dbo.[AddressType]
    DROP CONSTRAINT [AddressType_pk]

DROP TABLE dbo.[AddressType]

ALTER TABLE dbo.[Address]
    DROP CONSTRAINT [Address_pk]

DROP TABLE dbo.[Address]

ALTER TABLE dbo.[Account]
    DROP COLUMN [Address_ID]
```

```
ALTER TABLE dbo.[AddressType] WITH NOCHECK ADD
    CONSTRAINT [AddressType::Type]
    FOREIGN KEY ([ID])
    REFERENCES [Type] ([ID])

ALTER TABLE dbo.[Category] WITH NOCHECK ADD
    CONSTRAINT [BannerData_Category]
    FOREIGN KEY ([Name])
    REFERENCES [BannerData] ([FavCategory])

ALTER TABLE dbo.[CountryName] WITH NOCHECK ADD
    CONSTRAINT [CountryName::Type]
    FOREIGN KEY ([ID])
    REFERENCES [Type] ([ID])

ALTER TABLE dbo.[Item] WITH NOCHECK ADD
    CONSTRAINT [Item_Inventory]
    FOREIGN KEY ([ItemId])
    REFERENCES [Inventory] ([ItemId])

ALTER TABLE dbo.[LanguageName] WITH NOCHECK ADD
    CONSTRAINT [LanguageName::Type]
    FOREIGN KEY ([ID])
    REFERENCES [Type] ([ID])

ALTER TABLE dbo.[Product] WITH NOCHECK ADD
    CONSTRAINT [Supplier_Products]
    FOREIGN KEY ([Supplier_SuppId])
    REFERENCES [Supplier] ([SuppId])

ALTER TABLE dbo.[Profile] WITH NOCHECK ADD
    CONSTRAINT [Account_Profile]
    FOREIGN KEY ([UserId])
    REFERENCES [Account] ([UserId]),
    CONSTRAINT [Profiles_FavoriteCategory]
    FOREIGN KEY ([FavCategory])
    REFERENCES [Category] ([CatId]),
    CONSTRAINT [Profiles_Language]
    FOREIGN KEY ([Language_ID])
    REFERENCES [LanguageName] ([ID])

ALTER TABLE dbo.[SignOn] WITH NOCHECK ADD
    CONSTRAINT [Account_SignOn]
    FOREIGN KEY ([UserName])
    REFERENCES [Account] ([UserId])

ALTER TABLE dbo.[Supplier] WITH NOCHECK ADD
    CONSTRAINT [Supplier_Address]
    FOREIGN KEY ([Address_ID])
    REFERENCES [Address] ([ID])
```

ACTION TSQL for MSPetShopOrders

```
CREATE TABLE dbo.[CreditCard]
(
    [ID] int IDENTITY (1 , 1) NOT NULL ,
    [CardExpiration] varchar (255) NULL ,
    [CardNumber] varchar (255) NULL ,
    [CardType] varchar (255) NULL
)

ALTER TABLE dbo.[CreditCard] WITH NOCHECK
    ADD CONSTRAINT [CreditCard_pk]
    PRIMARY KEY ([ID])

CREATE TABLE dbo.[OrderAddress]
(
    [ID] int IDENTITY (1 , 1) NOT NULL ,
    [BuildingNameOrNumber] varchar (20) NULL ,
    [CityOrTown] varchar (30) NULL ,
    [Country] varchar (50) NULL ,
    [StateOrProvince] varchar (20) NULL ,
    [StreetName] varchar (40) NULL ,
    [ZipOrPostcode] varchar (10) NULL ,
    [AddressType] varchar (20) NULL
)

ALTER TABLE dbo.[OrderAddress] WITH NOCHECK
    ADD CONSTRAINT [OrderAddress_pk]
    PRIMARY KEY ([ID])

ALTER TABLE dbo.[Orders] WITH NOCHECK
ADD
    [Card_ID] int NULL ,
    [BillingAddress_ID] int NULL ,
    [ShippingAddress_ID] int NULL

ALTER TABLE dbo.[Orders] WITH NOCHECK ADD
    CONSTRAINT [Order_CreditCard]
    FOREIGN KEY ([Card_ID])
    REFERENCES [CreditCard] ([ID]),
    CONSTRAINT [OrderBA_BillingAddress]
    FOREIGN KEY ([BillingAddress_ID])
    REFERENCES [OrderAddress] ([ID]),
    CONSTRAINT [OrderSA_ShippingAddress]
    FOREIGN KEY ([ShippingAddress_ID])
    REFERENCES [OrderAddress] ([ID])
```

UNDO TSQL for MSPetShopOrders

```
ALTER TABLE dbo.[Orders]
   DROP CONSTRAINT [OrderSA_ShippingAddress]

ALTER TABLE dbo.[Orders]
    DROP CONSTRAINT [OrderBA_BillingAddress]

ALTER TABLE dbo.[Orders]
    DROP CONSTRAINT [Order_CreditCard]

ALTER TABLE dbo.[Orders]
    DROP COLUMN [Card_ID],[BillingAddress_ID],[ShippingAddress_ID]

ALTER TABLE dbo.[OrderAddress]
    DROP CONSTRAINT [OrderAddress_pk]

DROP TABLE dbo.[OrderAddress]

ALTER TABLE dbo.[CreditCard]
    DROP CONSTRAINT [CreditCard_pk]

DROP TABLE dbo.[CreditCard]
```

Data added to new tables, moved between existing and new tables, Foreign Key values set.

TSQL for MSPetShop.Address
This copies address data from Account and sets the PK of the record in Address as the FK value in Account

```
declare @pk varchar(20), @accountpk varchar(20)

create table #tempTable(AccountID varchar(20),
                        Addr1 varchar(80) NULL, Addr2 varchar(80) NULL, City varchar(80) NULL,
                        State varchar(80) NULL, Zip varchar(20) NULL, Country int, Type int)

INSERT INTO #tempTable(AccountID, Addr1, Addr2, City, State, Zip, Country, Type)
       SELECT UserId, Addr1, Addr2, City, State, Zip,
       (SELECT [ID] FROM Type WHERE ConstructorClassName = 'CountryName' AND [Description] = 'United States'),
       (SELECT [ID] FROM Type WHERE ConstructorClassName = 'AddressType' AND [Description] = 'Account')
        FROM Account
       ORDER BY UserId

WHILE (SELECT COUNT(AccountID) FROM #tempTable) > 0
BEGIN
       SELECT TOP 1 @accountpk = AccountID FROM #tempTable ORDER BY AccountID;
       INSERT INTO Address(BuildingNameOrNumber, StreetName, CityOrTown, StateOrProvince, ZipOrPostcode, Country_ID, AddressType_ID)
             SELECT Addr1, Addr2, City, State, Zip, Country, Type FROM #tempTable
              WHERE AccountId = @accountpk;
       SET @pk = SCOPE_IDENTITY();
       UPDATE Account SET Address_ID = @pk WHERE UserID = @accountpk
       DELETE FROM #temptable WHERE AccountID = @accountpk
END

DROP table #tempTable
```

TSQL for MSPetShop.AddressType (and Type)
This adds records to Type table and then to AddressType table

```
declare @pk int

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('Billing', 'AddressType');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO AddressType ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('Shipping', 'AddressType');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO AddressType ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('Account', 'AddressType');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO AddressType ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('Supplier', 'AddressType');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO AddressType ([ID]) VALUES (@pk)
```

TSQL for MSPetShop.CountryName (and Type)
This adds records to Type table and then to CountryName table

```
declare @pk int

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('United States', 'CountryName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO CountryName ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('United Kingdom', 'CountryName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO CountryName ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('Japan', 'CountryName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO CountryName ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('France', 'CountryName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO CountryName ([ID]) VALUES (@pk)
```

TSQL for MSPetShop.LanguageName (and Type)
This adds records to Type table and then to LanguageName table

```
declare @pk int

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('English', 'LanguageName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO LanguageName ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('French', 'LanguageName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO LanguageName ([ID]) VALUES (@pk)

INSERT INTO Type ([Description], [ConstructorClassName])
VALUES ('Japanese', 'LanguageName');
SELECT @pk = SCOPE_IDENTITY();
INSERT INTO LanguageName ([ID]) VALUES (@pk)
```

TSQL for MSPetShop.Product
This sets the Foreign Key value to a default Supplier

```
UPDATE Product
SET Supplier_SuppId = 1
```

TSQL for MSPetShop.Profile
This sets the Foreign Key value to a default Language Preference

```
UPDATE Profile
SET Language_ID = (SELECT [ID] FROM Type WHERE ConstructorClassName = 'LanguageName' AND [Description] = 'English')
```

TSQL for MSPetShop.Supplier
This copies address data from Supplier and sets the PK of the record in Address as the FK value in Supplier

```
declare @pk int, @supplierpk int

create table #tempTable(SupplierID int,
                        Addr1 varchar(80) NULL, Addr2 varchar(80) NULL, City varchar(80) NULL,
                        State varchar(80) NULL, Zip varchar(20) NULL, Country int, Type int)

INSERT INTO #tempTable(SupplierID, Addr1, Addr2, City, State, Zip, Country, Type)
       SELECT SuppId, Addr1, Addr2, City, State, Zip,
       (SELECT [ID] FROM Type WHERE ConstructorClassName = 'CountryName' AND [Description] = 'United States'),
       (SELECT [ID] FROM Type WHERE ConstructorClassName = 'AddressType' AND [Description] = 'Supplier')
        FROM Supplier
       ORDER BY SuppId

WHILE (SELECT COUNT(SupplierID) FROM #tempTable) > 0
BEGIN
       SELECT TOP 1 @supplierpk = SupplierID FROM #tempTable ORDER BY SupplierID;
       INSERT INTO Address(BuildingNameOrNumber, StreetName, CityOrTown, StateOrProvince, ZipOrPostcode, Country_ID, AddressType_ID)
              SELECT Addr1, Addr2, City, State, Zip, Country, Type FROM #tempTable
               WHERE SupplierID = @supplierpk;
       SET @pk = SCOPE_IDENTITY();
       UPDATE Supplier SET Address_ID = @pk WHERE SuppId = @supplierpk
       DELETE FROM #temptable WHERE SupplierID = @supplierpk
END

DROP table #tempTable
```

TSQL for MSPetShopOrders.CreditCard
This copies card data from Orders and sets the PK of the record in CreditCard as the FK value in Orders

```
DECLARE @pk int, @orderpk int

CREATE TABLE #tempTable(OrderID int, CardExpiration varchar(7) NULL, CardNumber varchar(20) NULL, CardType varchar(40) NULL)

INSERT INTO #tempTable(OrderID, CardExpiration, CardNumber, CardType)
       SELECT OrderId, ExprDate, CreditCard, CardType FROM Orders
       ORDER BY OrderId

WHILE (SELECT COUNT(OrderID) FROM #tempTable) > 0
BEGIN
       SELECT TOP 1 @orderpk = OrderID FROM #tempTable ORDER BY OrderID;
       INSERT INTO CreditCard( CardExpiration, CardNumber, CardType)
             SELECT CardExpiration, CardNumber, CardType FROM #tempTable
              WHERE OrderID = @orderpk;
       SET @pk = SCOPE_IDENTITY();
       UPDATE Orders SET Card_ID = @pk WHERE OrderID = @orderpk
       DELETE FROM #temptable WHERE OrderID = @orderpk
END

DROP TABLE #tempTable
```

TSQL for MSPetShopOrders.OrderAddress
This copies address data from Orders and sets the PK of the record in OrderAddress as the FK value in Orders

```
declare @pk int, @orderpk int, @type varchar(20)

create table #tempTable(OrderID int, Type varchar(20), Addr1 varchar(80) NULL, Addr2 varchar(80) NULL, City varchar(80) NULL,
                        State varchar(80) NULL, Zip varchar(20) NULL, Country varchar(20) NULL)

INSERT INTO #tempTable(OrderID, Type, Addr1, Addr2, City, State, Zip, Country)
       SELECT OrderId, 'Billing', BillAddr1, BillAddr2, BillCity, BillState, BillZip, BillCountry FROM Orders
       ORDER BY OrderId
INSERT INTO #tempTable(OrderID, Type, Addr1, Addr2, City, State, Zip, Country)
       SELECT OrderId, 'Shipping', ShipAddr1, ShipAddr2, ShipCity, ShipState, ShipZip, ShipCountry FROM Orders
       ORDER BY OrderId

WHILE (SELECT COUNT(OrderID) FROM #tempTable) > 0
BEGIN
       SELECT top 1 @orderpk = OrderID, @type = Type FROM #tempTable ORDER BY OrderID;
       INSERT INTO OrderAddress( AddressType, BuildingNameOrNumber, StreetName, CityOrTown, StateOrProvince, ZipOrPostcode, Country)
             SELECT Type, Addr1, Addr2, City, State, Zip, Country FROM #tempTable
              WHERE OrderID = @orderpk;
       SET @pk = SCOPE_IDENTITY();
       IF (@type = 'Billing')
              UPDATE Orders SET BillingAddress_ID = @pk WHERE OrderID = @orderpk
       ELSE
              UPDATE Orders SET ShippingAddress_ID = @pk WHERE OrderID = @orderpk

       DELETE FROM #temptable WHERE OrderID = @orderpk AND Type = @type
END

DROP table #tempTable
```

Redundant columns to be removed from database schema.

TSQL for MSPetShop.Address
This removes address columns/data from Account

```
ALTER TABLE [Account]
DROP COLUMN
     Addr1, Addr2, City, State, Zip, Country
```

TSQL for MSPetShop.Item
This removes redundant relationship to Supplier table

```
ALTER TABLE [Item]
DROP CONSTRAINT
     FK__Item__Supplier__08EA5793

ALTER TABLE [Item]
DROP COLUMN Supplier
```

TSQL for MSPetShop.Profile
This removes the language preference column/data from Profile

```
ALTER TABLE [Profile]
DROP COLUMN LangPref
```

TSQL for MSPetShop.Supplier
This removes address columns/data from Supplier (did not have a Country column)

```
ALTER TABLE [Supplier]
DROP COLUMN
     Addr1, Addr2, City, State, Zip
```

TSQL for MSPetShopOrders.OrderAddress
This removes card and address columns/data from Orders

```
ALTER TABLE [Orders]
DROP COLUMN
     BillAddr1, BillAddr2, BillCity, BillState, BillZip, BillCountry,
     ShipAddr1, ShipAddr2, ShipCity, ShipState, ShipZip, ShipCountry,
     ExprDate, CreditCard, CardType
```

APPENDIX B – New Code Count comparison

The code count works on the same principle as mentioned previously, that is:

Files counted were:
- *.VB, .CS, .ASPX, .ASCX, .ASPX.VB, .ASPX.CS, .ASCX.VB, and .ASCX.CS*

Excluded from the count were:
- *blank lines*
- *comment lines*
- *import declarations*
- *namespace declarations*
- *lines only containing { } braces*
- *html directives*
- *HTML/HEAD/TITLE/LINK/BODY tags*

and the same count methodology was applied to the M/Soft and Constructor versions.

CODE COUNT (REVISED)	Microsoft version	Constructor version part 1	Constructor version part 2
Presentation Layer	1,822	1,852	1,600
Model and Business Logic Layers	559	701	680
Data Access Layer	1,538	No user code required	No user code required
Total Lines of Code	**3,919**	**2,553**	**2,280**

Although the further reduction in lines of code is less significant than in ‘Part 1’ at approx. 270 lines, note that:

- The lines saved are the more complex ones related to class creation and data access
- The saving is in the context of a more complex set of relationships (doubled at 20) between the existing and the several new entities. Without Constructor, more complex collection code and data access statements would have been required, which could easily have amounted to hundreds of extra lines of code.
- The end result is more built-in run-time functionality, thanks to the ObjectFactory and the Run Time Model