8.1.0 – Transaction Overview

Introduction

In previous lessons we used LINQ queries to get data from the database and used ObjectDataSource controls to bind data to either a ListView or GridView control. Our BLL classes (i.e. ProductController) had code that was a simple transaction:

Listing 1: Sample Simple Transaction

```
[DataObjectMethod(DataObjectMethodType.Insert, true)]
public void AddProduct(Product productInfo)
{
    using (var context = new eStoreContext())
    {
        // What are the business rules? Add the code for them here.

        // ProductDescription is an optional field; i.e. can be NULL
        productInfo.ProductDescription =
string.IsNullOrEmpty(productInfo.ProductDescription) ? null :
productInfo.ProductDescription;

        // add and commit the changes
        context.Product.Add(productInfo);
        context.SaveChanges();
    }//end using
}//eom
```

In this lesson, you will see how transactions work.

Transactions

A transaction is a processing task that must be completely done successfully or not done (which we refer to as rolled back). This task is known as a Logic Unit of Work (LUW). The process of completing the work is called a Transaction.

Transactions that are used in business processes, such as Sales, Purchase Orders, and Inventory Control, all require the Transaction to interact with more than one database table, and sometimes more than one database. (Read 7_0_0_A_BusinessProcess.pdf.)

In our code, the entire transaction will occur in a code block like that below, which is in our BLL classes:

Listing 2: Transaction Code Block

```
// Setup transaction area
using (var context = new eStoreContext())
{
    // Entire Transaction will be here.
    context.SaveChanges();
}//end using
```

Types of Transactions

For businesses, there are several types of transactions. In this course, we will focus on:

- Shopping Cart and Sale
- Creating a Purchase Order
- Receiving an Order

Previously in this course:

- In lesson 4.2.0, you created the code and web form necessary to view all the products for a given category
- In lesson 5.2.0, you were able to select an item from a ListView
- In lesson 5.3.0, you could loop through a ListView, or GridView, to create a summary
- In lesson 6.1.0, you coded CRUD methods

Shopping Cart and Sale

There are many ways businesses design and code their online sales. Think of the last time you ordered something online. What steps did you do to order your item(s)? It is likely that you had the following steps (the next lesson will cover as many of the steps as time permits):

- Logging in to the store's web site
- View all products
- Search for a product
- Search products, either by category or name
- Add a product to a shopping cart
- Viewing the item(s) in your shopping cart
- Removing an item from your cart
- Updating the quantity of the item you added to your shopping cart
- Cancelling your order (not buying anything)
- Finishing your purchase (checkout); this also includes how you will pay for your item(s)

The figure below is a generic/basic Sequence Diagram for a shopping cart sale. You will notice that there are 2 Boundary classes, **WebSite-Shopping** and **WebSite-Checkout**. This design is typical of most online shopping cart sales. There will be one new technique to learn to be able to transfer data from one web form to another web form (covered in lesson 7.2.0):

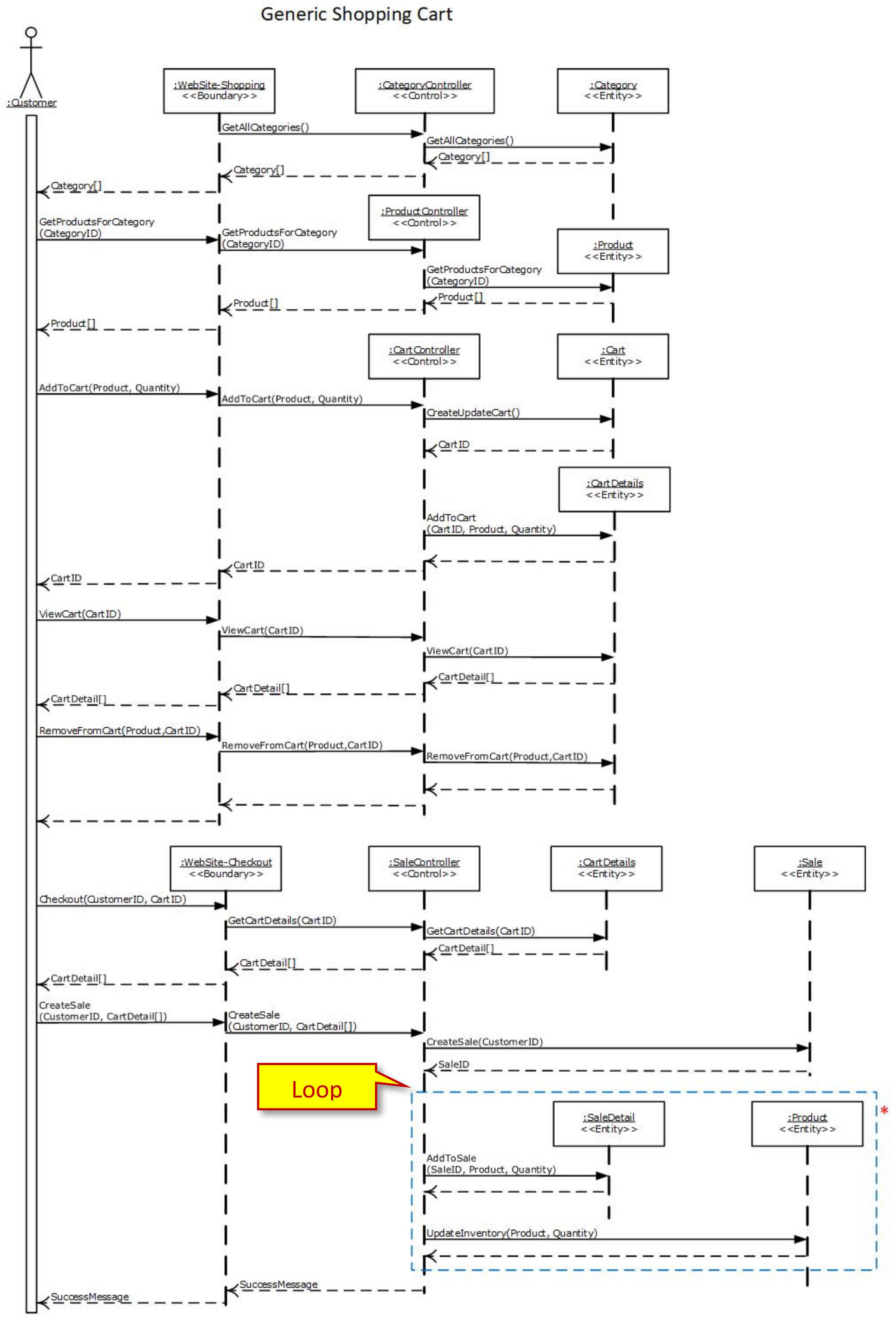
Generic Shopping Cart
:Customer
:WebSite-Shopping <<Boundary>>
:CategoryController <<Control>>
:Category <<Entity>>
GetAllCategories()
GetAllCategories()
Category[]
Category[]
Category[]
:Product Controller <<Control>>
:Product <<Entity>>
GetProductsForCategory (CategoryID)
GetProductsForCategory (CategoryID)
GetProductsForCategory (CategoryID)
Product[]
Product[]
Product[]
:Cart Controller <<Control>>
:Cart <<Entity>>
AddToCart(Product, Quantity)
AddToCart(Product, Quantity)
CreateUpdateCart()
Cart ID
:Cart Details <<Entity>>
AddToCart (Cart ID, Product, Quantity)
Cart ID
Cart ID
ViewCart(Cart ID)
ViewCart(Cart ID)
ViewCart(Cart ID)
Cart Detail[]
Cart Detail[]
Cart Detail[]
RemoveFromCart(Product, Cart ID)
RemoveFromCart(Product, Cart ID)
RemoveFromCart(Product, Cart ID)
:WebSite-Checkout <<Boundary>>
:SaleController <<Control>>
:Cart Details <<Entity>>
:Sale <<Entity>>
Checkout(CustomerID, Cart ID)
GetCartDetails(Cart ID)
GetCartDetails(Cart ID)
Cart Detail[]
Cart Detail[]
Cart Detail[]
CreateSale (CustomerID, CartDetail[])
CreateSale (CustomerID, CartDetail[])
CreateSale(CustomerID)
SaleID
Loop
*
:SaleDetail <<Entity>>
:Product <<Entity>>
AddToSale (SaleID, Product, Quantity)
UpdateInventory(Product, Quantity)
SuccessMessage
SuccessMessage

Figure 1: Generic Shopping Cart Sequence Diagram

Creating a Purchase Order

In DMIT2028, you were introduced to this business process. The basics are that businesses need products to sell and need to make sure they have enough products available for sale at all times. When the quantity available for one or more products goes below a predefined level, the business must order new products.

The sequence diagram for the generic creation of a purchase order is shown below. Notice this is a single web form:

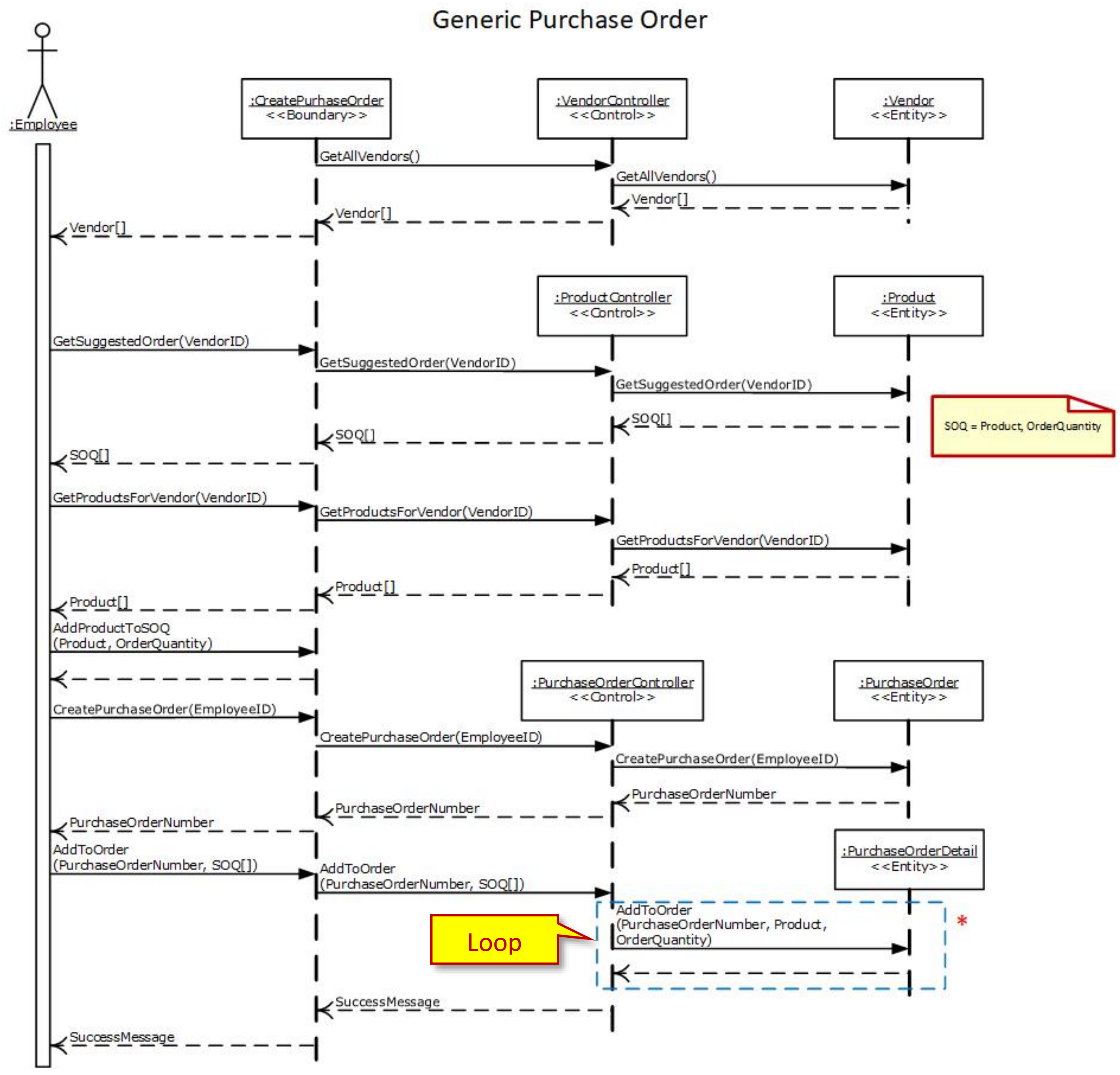
Generic Purchase Order
:Employee
:CreatePurhaseOrder <<Boundary>>
:VendorController <<Control>>
:Vendor <<Entity>>
GetAllVendors()
GetAllVendors()
Vendor[]
Vendor[]
Vendor[]
:ProductController <<Control>>
:Product <<Entity>>
GetSuggestedOrder(VendorID)
GetSuggestedOrder(VendorID)
GetSuggestedOrder(VendorID)
SOQ[]
SOQ[]
SOQ[]
SOQ = Product, OrderQuantity
GetProductsForVendor(VendorID)
GetProductsForVendor(VendorID)
GetProductsForVendor(VendorID)
Product[]
Product[]
Product[]
AddProductToSOQ (Product, OrderQuantity)
:PurchaseOrderController <<Control>>
:PurchaseOrder <<Entity>>
CreatePurchaseOrder(EmployeeID)
CreatePurchaseOrder(EmployeeID)
CreatePurchaseOrder(EmployeeID)
PurchaseOrderNumber
PurchaseOrderNumber
PurchaseOrderNumber
:PurchaseOrderDetail <<Entity>>
AddToOrder (PurchaseOrderNumber, SOQ[])
AddToOrder (PurchaseOrderNumber, SOQ[])
AddToOrder (PurchaseOrderNumber, Product, OrderQuantity)
Loop
*
SuccessMessage
SuccessMessage

Figure 2: Generic Create Purchase Order Sequence Diagram

Receive Purchase Order

In DMIT2028, you were introduced to this business process. The basics are that after sending out a purchase order to a vendor, the vendor will send you the products you ordered. When you receive this order, you need to add the products to your inventory and close the purchase order if all the items ordered are received.

The sequence diagram for the generic receiving of an order is shown below. Notice this is a single web form:

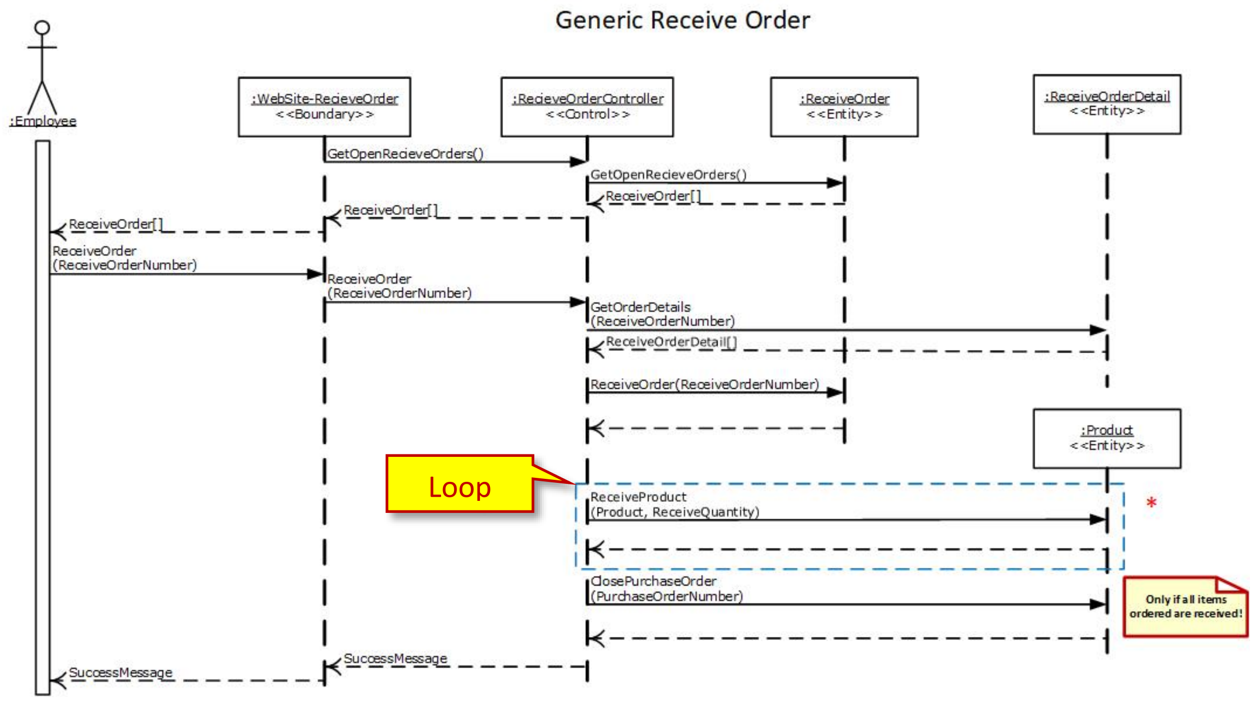

Figure 3: Generic Receive Order Sequence Diagram