# *Web Programming VI (420-H60-HR)*

# *Assignment 1 – Online Shopping Experience*

Date Assigned: Friday, August 30

Due Date: **Friday, September 13, 23h50**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Create and user a .NET Core MVC application
* Use Data-first Entity Framework to generate the classes for the data model and database
* Use Code-first Entity Framework to create and manipulate the data model and database
* ~~Use LINQ queries to manipulate the data in the database~~

**Setup**

Create a folder called initialsH60A01; this is the name of the folder you will zip and hand in. Inside this folder create a solution called *initials*H60Store. NOTE THE NAMING IS IMPORTANT HERE SO FOLLOW THE INSTRUCTIONS. The solution uses the MVC template for a Web application. This will create a folder in the A01 folder with the solution name.

You also need to figure out what kind of products you want to sell in your online store. The type of products must be able to be broken into at least 5 different categories and you must have at least 4 products per category. Keep things in good taste.

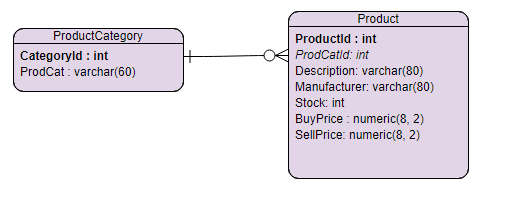
**The Business Case**

You have been hired to create a web application for a company that is going to start selling products online. Initially (this assignment) you are going to prepare the basic store ("warehouse") functionality which allows the owners to add and remove products. You will also set up the remaining tables in the database that will be used when the customer functionality is added.

In the upcoming assignments you will set up and use web services for all of store and client functionality to create a functioning shopping experience for both the store and the user.

Part A: Adding Products and Categories

1. You will create a database on cssql SQL Server called H60AssignmentDB\_*yourinitials*. To do this:
   1. get the H60AssignmentDB\_template.sql file.
   2. logon to the cssql server using SQL Server Management Studio and your local, personal Id and password.
   3. Copy in the H60AssignmentDB\_template.sql file.
   4. Rename occurrences of H60AssignmentDB\_template to H60AssignmentDB\_*yourinitials.*
   5. Run the script to create H60AssignmentDB\_*yourinitials.* This is your personal database that you’ll use for your assignment.
2. It contains two empty tables, one for ProductCategory and one called Product.

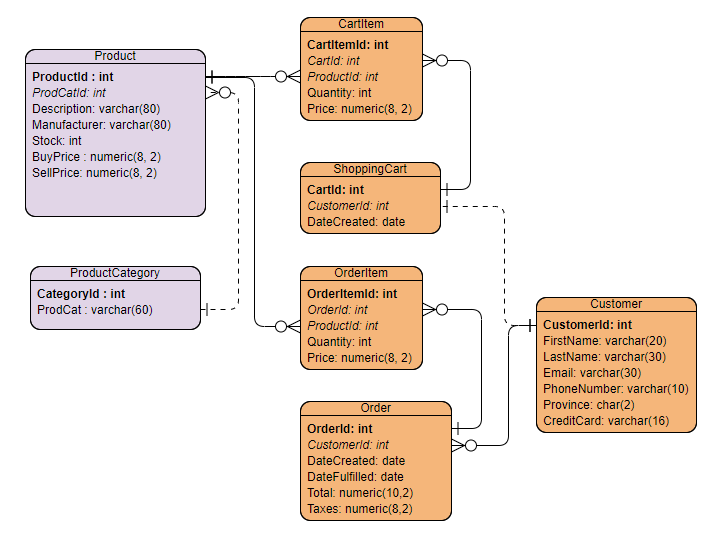


Create a connection to the database using Entity Framework database scaffolding. This will create your initial classes.

1. In your models create functions to perform CRUD operations on the ProductCategory table and Product table.
   1. It is critical to the remaining assignments and your sanity that these functions be singular in function and easily removable/replaceable, as in the upcoming assignments you will be replacing them with calls to ReST functions you will be creating.
   2. Keep ALL the business level functionality in the model.
   3. This is no DB access directly in the Controller. i.e no execution of dbContext in the controller.
2. In the Store solution provide views to access the model (and thereby the database) and perform the following operations:
3. Display a list of all product categories sorted by category
4. Display a list of all products (include description, stock and sell price) sorted by product
5. Update just the stock on a given product
6. Update the BuyPrice and SellPrice for a given product.
7. When updating the stock on a product:
8. The amount the stock value is to be changed by is passed as an integer parameter.
9. If the parameter is a negative number, the stock value is reduced by that value.
10. If the parameter is a positive number, the stock value is increased by that value.
11. The parameter is required.
12. If the passed value would make the stock value less than 0, throw an exception and do not change the stock value.
13. When updating the Buy or Sell price of a product
14. Price can have a maximum of 2 decimals.
15. If price is more than 2-decimals it is rounded.
16. If a price has less than 2-decimals, it is brought to a 2-decimal number.
17. If price is not a number an arithmetic exception is thrown.
18. If the new price is a negative number an argument exception is thrown.
19. If the Sell price is less than the Buy price an exception is thrown. (I added this rule, because the testing requirements here were so weak).
20. In the Store solution provide an interface to use the functions to access the database and perform the following operations:
21. CRUD on product categories (as above)
22. CRUD on products (as above)
23. Get a list of all products by product category (include same fields as above) sorted by category and product within category
24. Get a list of all products for a given product category sorted by product. This gets the abbreviated product information including: Description and SellPrice
25. Get the details for a given product (display all the details including the product category as a string not id).

**Part B – Prepping for Customers**

1. You are going to change the data model to prepare to implement the client side of the application. For this you will use code first Entity Framework Modelling. The new data model is:



1. These MUST be created using code first techniques. Notice, a customer can only have one shopping cart at a time.
   1. Add classes to the Models to represent the new tables.
   2. Use database seeding in Entity Framework to provide initial values to the tables. The new tables MUST have at least 3 customers, at least one customer MUST have a completed order with at least 3 products and at least one customer must have a shopping cart with at least 2 products, one of which has more than one ordered (quantity > 1).
   3. Create a migration to update the database and add the tables. Make sure this works, as I will start out by copying your assignment database and running the migration myself.
   4. DO NOT ADD ANY VIEWS/FORMS TO ADD THE CUSTOMER INFORMATION.

Implementation notes:

1. All queries must use LINQ (your choice as to which syntax: query, method).
2. Make it professional and make it work on mobile. This is your final ever web assignment; do something you will be proud of.
3. Make it homogeneous and real. That is, if you want an online grocery store, make that the items. If you want an online gaming shop, that’s fine. Make the categories and products make sense. The next assignment will build on this.

**Final Five**

Completing the above receives a maximum of 95%. The final 5% is available if you complete at least three of the following extra pieces of functionality.

1. Allow filtering on partial product name.
2. Allow filtering on price (equal to, less than and greater than).
3. “Highlight” or otherwise indicate items that are out of stock in the list of items.
4. Add an image for a category (database change needed).
5. Add employee notes per product (database change).
6. Add a minimum/maximum limit of stock with an indication for re-stock limit (database change).
7. Allow sorting by price, amount of stock or markup (selectable)

**Marking**

|  |  |  |
| --- | --- | --- |
| **Web Programming VI (420-H60-HR) Assignment 1 (eStore)** |  | **Mark** |
|  |  | 0% |
|  |  |  |
|  | **Mark** | **Out Of** |
| **Functionality Store** |  |  |
| Setting up Classes using scaffolding |  | 2 |
| Get a list of all product categories sorted |  | 3 |
| Get a list of all products sorted |  | 4 |
| Update stock on a product |  | 5 |
| Update buy and sell price on product |  | 5 |
| CRUD for Product Categories |  | 8 |
| CRUD for Products |  | 9 |
| List of products by category |  | 3 |
| List of product for specific category |  | 4 |
| Details of product |  | 2 |
| Migration for new tables |  | 5 |
| Seeding performed on new tables |  | 4 |
| **Functionality** |  |  |
| Interface design and flow, professionalism, proper english |  | 10 |
| Handling errors returned |  | 4 |
| **Design** |  |  |
| Skinny controller |  | 6 |
| Distinct functions for DB access |  | 8 |
| Code design/efficiency/SRP |  | 10 |
| **Work Period** |  |  |
| Used given class time to work effectively |  | 5 |
| **Final Five** |  |  |
| Completed Three of the Following: - Allow filtering on partial product name. - Allow filtering on price (equal to, less than and greater than). - Indicate items that are out of stock in the list of items. - Add an image for a category - Add employee notes per product - Add a minimum/maximum limit of stock with an indication for re-stock limit - Allow sorting by price, amount of stock or markup |  | 5 |
|  |  |  |
| **Total** | 0 | 100 |

**To submit**

When you have completed the assignment, submit a zip file (***initials*H60A01.zip**) with all the files from the document folder to the Moodle page for this course.

Note: Here’s how I test, I advise you also make sure this sequence works:

1. Copy in your solution
2. Change the connection string to my own database with my own username, password.
3. Run “Update-Database” to setup my own database.
4. Run the solution. It should just work with seeded data.