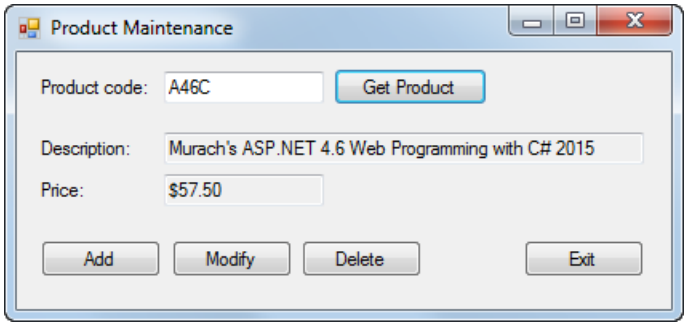
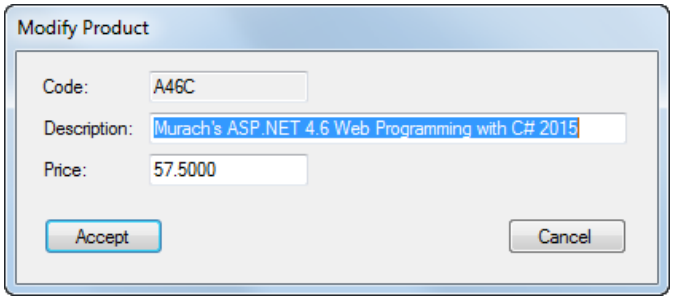
CS2563 : Week 8 Assignment – Product Maintenance application

In this exercise, you'll use ADO.NET to write the data access code for an application that lets the user add, modify, and delete products.





Open the project and add a class that gets a connection to the database

1. Open the ProductMaintenance that is attached. This project contains the two forms for the application, the Product and Validator classes, and the MMABooks.mdf database file.
2. Add a public class named MMABooksDB to the project. Then, add a static method named GetConnection that creates an SqlConnection object for the MMABooks database and then returns that connection. For this to work, you'll need to add a using directive for the System.Data.SqlClient namespace at the beginning of the class.

Write the code to retrieve a product

3. Add another public class named ProductDB to the project, and add using directives for the System.Data and System.Data.SqlClient namespaces to this class.
4. Add a static method named GetProduct to the ProductDB class. This method should receive the product code of the product to be retrieved, and it should return a Product object for that product. If a product with the product code isn't found, this method should return null. Place the code that works with the database in the try block of a try-catch statement, include a catch block that catches and then throws any SqlException that occurs, and include a finally block that closes the connection.
5. Display the code for the Product Maintenance form, and add a statement to the GetProduct method that calls the GetProduct method of the ProductDB class to get the product with the product code the user enters. Assign the product that's returned to the product variable.
6. Test the application to be sure that the data for a product is displayed when you enter a product code and click the Get Product button.

Write the code to update a product

7. Add a static method named UpdateProduct to the ProductDB class. This method should receive two Product objects. The first one should contain the original product data and should be used to provide for optimistic concurrency. The second one should contain the new product data and should be used to update the product row. This method should also return a Boolean value that indicates if the update was successful. Like the GetProduct method, this method should include a try-catch statement with a catch block that catches and then throws any SqlException that occurs, and a finally block that closes the connection.
8. Display the code for the Add/Modify Product form, and locate the event handler for the Click event of the Accept button. Add code to this event handler that calls the UpdateProduct method of the ProductDB class. If this method returns a true value, the event handler should assign the newProduct object to the product object and then set the DialogResult property of the form to DialogResult.OK. Otherwise, it should display an error message indicating that another user has updated or deleted the product, and then set the DialogResult property to DialogResult.Retry.
9. Test the application to be sure the modify operation works correctly.

Write the code to add a product

10. Add a static method named AddProduct to the ProductDB class. This method should receive a Product object with the data for the new product, and it should return a Boolean value that indicates if the add operation was successful. Be sure to include a try-catch statement with a catch block that catches and throws an SqlException and a finally block that closes the connection.
11. Display the code for the Add/Modify Product form, and add code to the event handler for the Click event of the Accept button that calls the AddProduct method of the ProductDB class. If this method returns a true value, the event handler should set the DialogResult property of the form to DialogResult.OK. Otherwise, it should display an error message indicating that another product with the same code already exists, and then set the DialogResult property to DialogResult.Retry.
12. Test the application to be sure the add operation works correctly.

Write the code to delete a product

13. Add a static method named DeleteProduct to the ProductDB class. This method should receive a Product object with the data for the product to be deleted, and it should return a Boolean value that indicates if the delete operation was successful. The Product object should be used to provide for optimistic concurrency. Include a try-catch statement like the ones in the other methods of this class.
14. Display the code for the Product Maintenance form, and add code to the event handler for the Click event of the Delete button that calls the DeleteProduct method of the ProductDB class. If this method returns a true value, the event handler should call the ClearControls method of the form. Otherwise, it should display a message indicating that another user has updated or deleted the product. Then, it should call the GetProduct method of the form to determine if the product has been deleted. If it hasn't, the event handler should call the DisplayProduct method of the form. Otherwise, it should call the ClearControls method of the form.

15. Test the application to be sure the delete operation works correctly. Note, however, that you won't be able to delete a product if it's associated with one or more rows in the InvoiceLineItems table.