Introduction

This article explains how you can start the shop website without having to write every thing of ASP.NET. It also explains some basic rules and regulations for the shop website which would help you build your website even more better.

Environment Requirement

You need to be having Microsoft WebMatrix or Visual Studio where you can build, compile and run this project. For beginners you can download Microsoft WebMatrix from the Microsoft's website (http://www.microsoft.com/web) and then run this project. When you download the software, Web Platform Installer would install all the required software for the project to run.

You need to copy/paste this extracted folder to the C:/Users/<username>/Documents/My WebSites/<here>

After this you will be able to run this project in WebMatrix.

Background

Many forums have been developed and many question and answer websites are provided but still developers who are new to development world mostly ask such questions related to this problem. "How can I develop my own eCommerce/shopping website in ASP.NET?". So for them I am here with an article post where I will explain the basics how they should/would start their journey and add some lines of codes to get it to work.

What is eCommerce

eCommerce is also known as Electronic Commerce, where you put your entire business on the internet and the customer can use the UI to trigger the transactions.

Most of the new developers get tricked by the name of this software "eCommerce" and believe it might be a very big project or it would require something that is very hard to build.

There is no such thing, even a developer with little knowledge of ASP.NET can develop a simple website for his small company.

Major points/Requirements

Like all other projects, eCommerce website has some functions that are required for it to work. Similarly eCommerce requires some basic functions that include the most importantly following points

1. A platform where the a user can view all the content of the website. Like the items to display in the web page.
2. This project also requires a Database, that must be set up for the process of transaction and user account. Most of the eCommerce professionals build a database and allow the users to set up their account. So that they can have a customer and so on.
3. Basic implementation of security in your website, to prevent hacking and so on.

4. Code for each type of item to be handled, and for each user. Guest and registered user must be handled as they deserve to be.
5. UI must be user friendly and user must feel free to browser your website. Provide him with less but efficient details.

These are some major requirements while developing eCommerce. The real nightmare comes when you have to learn the API of a third-party service to include the Checkout system. I don't have any API so I won't add any Checkout system right now, in future if I get access to any, I will post it here!

The entire project is just a simple website, with the basic implementation of security and selling. Where you sell your products to the users by showing them an image and some details about your product like its price, description etc. You can add as many details to it as you can to be more user friendly.

You can also allow the users to create their account on your website, although it is not required since the third party service they'll use for checkout would have this handled. For this you will require the database. Database is also required for the data related to the Products and Details etc. But giving a free account to the users is a good attemp to get more customers.

Database Design

Database design is a very basic and most important part in this shopping website. You put all of your content inside of the Database, and then extract it from there.

Designing a database as per your requirement is very essential step. You need to make an algorithm for the tables and the objects inside each table that you would need. For example, in a Product table you need to be having these following objects a lot

1. **ProductId**
   Which is used to distinguish among all the products. This is the ID that you assing to it. Its data type is int, since you would not be using ABC as the ID of the product.

2. **ProductName**
   This tells the user what is this product. Most of the products are understandable using their name. "Apple Pie" is a name for the product. It suggests the user what this product is. You can easily understand that this is a pie made up of apples.

3. **ProductDescription**
   Sometimes it is easy to fully describe your product and the help or support product would provide the user with. "Apple pie, made out of fresh apples". Nice description though, to explain that this pie is made up of fresh apples.

4. **ProductPrice**
   Used to tell the user, how much he would be required to pay for this product. Usually you pass this value in dollars but providing this in user friendly mode (in local currency) is a better attempt.

5. **AvailableItems (Optional)**
   This feature enables the users to know how much quantity of the product is available. This is not required, it is just optional where you can tell the user how much he can buy right now. Most of the major companies use this to tell their customers, where purchase is made in tens and hundreds, that he can right now buy only this much items or he must order for more.

The User table would be only created if you want to allow the users to have accounts on their website. Users table would be like this

1. **UserId**
   Same purpose, just to distinguish between users. There might be another user with the name "Afzaal" but no other user with ID of 1.

2. **Email**
   Used to contact the user to inform the user for his purchase.

3. **Name**
   To identify the user. Although this is not required you can use the email of his, but calling the user as "Hello, Afzaal!" is better than calling him as "Hello, example@example.com".

4. ...add more functions to this table if you need so.

Then comes the final table required, the purchases that are made through your website. You need to save these purchases, so that if any error occurs in transaction or the deal doesn't reach the destination you can track it in your website and guide the customer. It is as

1. **UserId**
   The user id of the user that made the purchase

2. **ProductId**
   The id of the product that was purched

3. **Time**
   Time of the purchase

Enough!

Why I wrote just 3 columns? Because when you use UserId and ProductId, you can extract the remaining content from other two tables. There is no need to save the UserName again, or the ProductName, ProductDescription, it would just take the amount of space on your HDD. So, using a UserId and ProductId you can extract the remaining content from the Database using that value.

The screenshots that I took were as the following, it would explain what might be the thing I was trying to explain.

Table - (bakery.sdf).Products

Id	Name	Description	Price	ImageName
1	Carrot Cake	A scrumptious mini-carrot cake encr	8.99	carrot_cake.jpg
2	Lemon Tart	A delicious lemon tart with fresh mei	9.99	lemon_tart.jpg
3	Cupcakes	Delectable vanilla and chocolate cup	5.99	cupcakes.jpg
4	Bread	Fresh baked French-style bread	1.49	bread.jpg
6	Pear Tart	A glazed pear tart topped with sliced	5.99	pear_tart.jpg
7	Chocolate Cake	Rich chocolate frosting cover this ch	8.99	chocolate_cake.jpg

In the above screenshot, ImageName column is extra that I didn't explain. That is because in this template, there is an image attached to each of the product. That is not required, but is a good attempt. Most of the images are related to the products that you want to show. You can either save their name in the Database or save them anywhere else as per your needs.

Table - (bakery.sdf).Purchase | Table - (bakery.sdf).Users × | Table - (bakery.sdf).Products

UserId	Name	Email	Address
1	Afzaal Ahmad Zeeshan	example@example.com	My home address!
2	Daniyal Ahmad Rizwan	example1@example.com	His home address

Table - (bakery.sdf).Purchase ×

UserId	ProductId	Time

This is the basic structure of the database for a shop website. You can add more features to your database tables and add more tables to your database if you need to do so.

Shopping Website

A shopping website is a subsite of an eCommerce website. In this website you will find almost all of the functionality of the eCommerce website but why is it called a shopping website?

Because eCommerce websites are generally for the major giants of the business. For example a banking website, an international wire transfer for money, a mall for shopping on internet. They require a very large framework to work on. They are called eCommerce websites for [particular_field].

A shopping website, is where **you** place all of your content of the products, such as the one explained in this, Apple Pie and all other food products. And you ask the customers to, ok, you can place an order here too if you can't reach me. This doesnot require a very big security check, or a large framework for business to be handled. You create the website, you let the users use it. Enough! The users that access you content are loyal to you (in most cases) and they'll place an order and you'll reach them at their home and give and take takes place. You give them their order they give you the money, which was the deal. This doesnot reflect eCommerce, where transactions take place, user send their money, order to withdraw the money. That is why, shopping website is a sub category of the eCommerce website.

That is why it is called shopping website. Shopping website is easy to develop as compared to eCommerce, because you only need to place the products in the Database and access them everytime the user surf your website. After this, you simply generate an order page where the user inputs his details and places an order which, you handle later.

Code in the Project

ASP.NET (Web Pages)

This project was developed by the ASP.NET team and is the best solution for any website owner to kick start his website. All he needs to know is how to edit ASP.NET and a very basic knowledge for HTML and CSS, including JavaScript.

You can learn ASP.NET here.

Hide Copy Code

```
var db = Database.Open("bakery");
var products = db.Query("SELECT * FROM PRODUCTS").ToList();
```

```
var featured = products[new Random().Next(products.Count)];
```

Th above code is from the main page. What the above code does is that it extracts the results from the Database and then gets a Random item.

This functionality is seen on many website, where you see a Random item on the main page header and all other content is displayed below it! You get a Random number from the limit and then you access the data for that object.

jQuery code

One thing in this website that I like is the live calculator for the total amount being charges. jQuery was used for this function. Following is the code for this function,

Hide Copy Code

```
<script type="text/javascript">
   $(function () {
       var price = parseFloat($("#orderPrice").text()).toFixed(2),
       total = $("#orderTotal"),
       orderQty = $("#orderQty");
       orderQty.change(function () {
          var quantity = parseInt(orderQty.val());
          if (!quantity || quantity < 1) {
              orderQty.val(1);
              quantity = 1;
          } else if (quantity.toString() !== orderQty.val()) {
                orderQty.val(quantity);
          }
          total.text("$" + (price * quantity).toFixed(2));
       });
    });
</script>
```

The above codes gets the value from the user and then counts the total amount he'll pay. After he submits the form, the website sends email to the user by getting his email address while making his purchase. And you're done! This is the entire shop website where you get an order from the user you process it, and inform the user. The total hierarchy is as

1. User enters
2. Places an order
3. You process it
4. Inform the user
5. Parcel the order

That is the entire logic for the shopping website. You don't need to hardcode anything, just google for the best methods available for such methods.

Simple wasn't it?

Database/SQL Server CE

A default website's database is also present inside the project zip folder. You can extract the project and run it inside the WebMatrix or Visual Studio. Database is provided to you and you can execute it. You can alter the database and use it.

Feel free to update the template as much as you can. You should always get a good result for your attempts. I don't find any other part that needs some explaination. If you need help, contact me or message/comment here I would

guide you.

Extended Feature in Project

I have developed one more feature in the site, that was lacked. It was the ability to create a new product using user interface. Although you can always go to the Database, add the content, come back and then copy/paste the image inside the folder where you want it to be. There are some CSS fixes in the code that enables the image to only span across the area allowed. If you upload the image more than 670px then the button to the right side was overriden. This issue was solved adding a line of code

Hide Copy Code

```
div#featuredProduct img {
   width: 670px;
}
```

Now the image doesn't expand more than 670px, but the height is not used here you should take care of that.

The new page added was: "NewProduct" in the Admin folder. You can add security to this page yourself or use any other method to hide/secure it.

The page include the following code,

Hide Shrink Copy Code

```
@{
    // page related codes
    Layout = "~/_SiteLayout.cshtml";
    Page.Title = "Add Product";
    var result = "";

    if(IsPost) {
        // Form submitted, save it!
        // initialize the variables.
        var name = Request.Form["name"];
        var desc = Request.Form["desc"];
        var price = Request.Form["price"];
        // get the image from the request.
        var image = WebImage.GetImageFromRequest();
        var fileName = "";

        // get if there is some image with the request. Otherwise,
        // remember, the image won't be shown and it is not a good UX
        // to see a broken link to image. But it will work!
        if(image != null) {
            // get the file name
            fileName = Path.GetFileName(image.FileName);
            var mapPath = Server.MapPath("~/Images/Products/" + fileName);
            var thumbPath = Server.MapPath("~/Images/Products/Thumbnails/" + fileName);
            // save the image
            image.Save(mapPath);
            image.Save(thumbPath);
        }

        // database events
        var db = Database.Open("bakery");
        // insert the data to the database
        db.Execute("INSERT INTO Products (Name, Description, Price, ImageName) VALUES (@0, @1, @2, @3)",
name, desc, price, fileName);
        // update the variable to show in the HTML markup.
        result = "Your product was saved. View it in your shop website!";
    }
}

<!DOCTYPE html>
```

```
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title></title>
        <!-- CSS styles on the page -->
        <style>
            label {
                margin: 0;
                display: block;
            }
            input[type="text"], textarea {
                padding: 2px;
                width: 300px;
                font-family: inherit;
            }
            textarea {
                height: 100px;
            }
        </style>
    </head>
    <body>
        @result
        <h4>Details</h4>
        <p>Add details for you product then save it to the database.</p>
        <form method="post" enctype="multipart/form-data">
            <label id="name">Product Name</label><br />
            <input type="text" name="name" id="name" /><br />
            <label id="price">Product Price</label><br />
            <input type="text" name="price" id="price" /><br />
            <label id="desc">Product Description</label><br />
            <textarea name="desc" id="desc"></textarea><br />
            <label id="name">Product Image</label><br />
            <input type="file" name="image" id="name" accept="Image/*" /><br />
            <p>Using this, you can add an image that will be shown to the user when he will open the
product page.</p>
            <input type="submit" value="Save" />
        </form>
    </body>
</html>
```

This includes a simple functionality to write the details and select a photo that would be saved across the website and a new entry for the product would also be added. You will get a message if everything goes Ok, otherwise you'll not see anything (simple as that!).

Points of Interest

I've learnt, "Helping others is a great thing to do!". Enough to learn. This was the first time, I just wrote the code from scratch and after doing so in the first attempt I executed the code and it worked! Well after a very long time I finally came to learn ASP.NET ^_^.