Milestone 6: Cart Controller and Views

CPS 353: Internet Programming
Web Development Project
Due Sunday, November 22, 2015 by 11pm

Introduction

The goal of this milestone is to developer the shopping cart view. I made it smaller since I am giving it to you late and I extended the deadline.

We will need to hook up the zip code field to allow users to enter a zip code. With this information, we will be able to show the cheaper prices for qualifying products. This will expose you to working with cookies. Also in this milestone we will lay the ground work for making ajax calls to the controllers. If there is time in the course, we will come back to one or two of our ajax calls and make them more efficient.

We will also update the featured product view and the items in the cart so that they are aware of SalePrice. This means we will need to find promotions that are active and apply to this product in this zip code. Our UI will be adjusted to show the retail price crossed out with the sale price next to it.

We also need to make some tweaks to our data structure for the promotions.

The details are below.

Specifications

Add data and database changes

Update the Promotions table to have an optional ProductId column which is a foreign key to the Product table. Add a column to CartPromotions called CartId. This cannot be null and should be an integer and a foreign key to the Cart table. Drop the ProductId column from this table. Update your edmx file and models and make sure the new columns are part of the table definitions in the sql files you turn in.

Create sql files to insert data for at least 2 categories with 10 products each. Include these in the zip file that you turn in. Add two promotions record that is only valid in the 01984 zip code. One record should be for one of your products, the other record for a different product. For one of them make the date range from November 1, 2015 – December 1st 2015. For the other set the range to be from October 1st to October 31st. Enter other data for the promotions as needed.

Create controllers and service

If you do not have one, create a ProductViewModel and a mapping method that maps a product model to a product viewModel. Update this product view model to have a SalePrice property. If you have not, create a mapper function that maps a product model to a product view model.

Add an action to the home controller called UpdateZip that saves the user entered zip code to session. You can add this by creating an HttpCookie in the controller action and calling SetCookie on the Response object. Add a call to this action by using a js ajax call like we saw in class. This js code will need to be added to a js file which you can call site.js. The view returned from the action can be any since we are ignoring the actual html returned. By setting the cookie in the response, that will create it on the client but still be available in future requests.

Add a product service to your solution that follows the unit of work pattern (or update yours to follow this pattern if you have not) discussed in class to access the repositories. This service should have the following method:

- GetProductSalePrice – this takes a product id and a zipcode and returns 0 if there was no valid promo or the sale price if there was an active promo for this product in this zip code for today.

Update the product viewmodel mapper code such it calls the GetProductSalePrice method in the product service and returns a list of valid promotions. Set the Product viewmodel SalePrice property to the promotion price. And show this new price in the UI (if it is not zero) and put a strikethrough style on the retail price. This should be done in the featured product as well as in the shopping cart. Any calls to this method in the service should get the zip code from the Requset.Cookies collection. If it is not there, you do not need to call the service method.

Add a user service that uses the unit of work pattern to access the repositories it needs. Add the following methods to this service:

- CreateUser – this takes an option first and last name and an optional email address and creates a user object in the db. It returns this object with the user id. If no name or email are provided and if your schema requires one, use default values.

Add a cart service that uses the unit of work pattern to access the repositories. To this service add the following methods:

- GetCart – this takes a status (new, abandoned or purchased) and a user id and returns the cart or null if one is not found.
- CreateCart – this takes a userid and an optional zip code and creates a cart object. If User is -1, call the user service's CreateUser to get a valid user id and use this when creating the cart.
- UpdateCart – this takes a CartId, a product id and a quantity (defaulting to 1) and adds the item to the cart. If quantity is zero, remove the item from the cart.

Add a cart controller that has the following actions:

- AddToCart – takes a product id
    - If there is a cookie with the user id, then look up the cart for that user with status new by calling the GetCart method on the cart service passing the user id and the "new" status. If a cart is found, call the UpdateCart, passing in the cart id, the product id and the quantity of 1.
    - If there is no cookie with the user id, then call CreateUser on the user service, With the response that is returned, add the user id to a cookie and add the cookie to the response for future use. Then create a cart and add the item to the cart.
    - AddToCart should redirect to the cart controller and that will render the cart view. You can call the AddToCart via ajax if you want and when you get the response back, redirect to the new shopping cart page. Or you can try to do a Response.Redirect call on the server to move to the new page

Add and update view files

Featured Product

This should now show the sale price if there is one. Also add an "Add To cart" button. That calls the AddToCart action on the cart controller.

Shopping cart view
Create a view that implements the mock up you created. Make sure that the delete and quantity change options work. This will be returned by the cart controller. Update the navigation so that when I click the shopping cart icon, I go to the cart. Remember that the cart page needs to handle the case for when there are no products and when there is no user record yet.

Required files
Zip up and turn in your entire solution along with the sql files. Also, include sql for inserting product and category data to make testing easier.

This assignment is due November 22nd by 11pm.