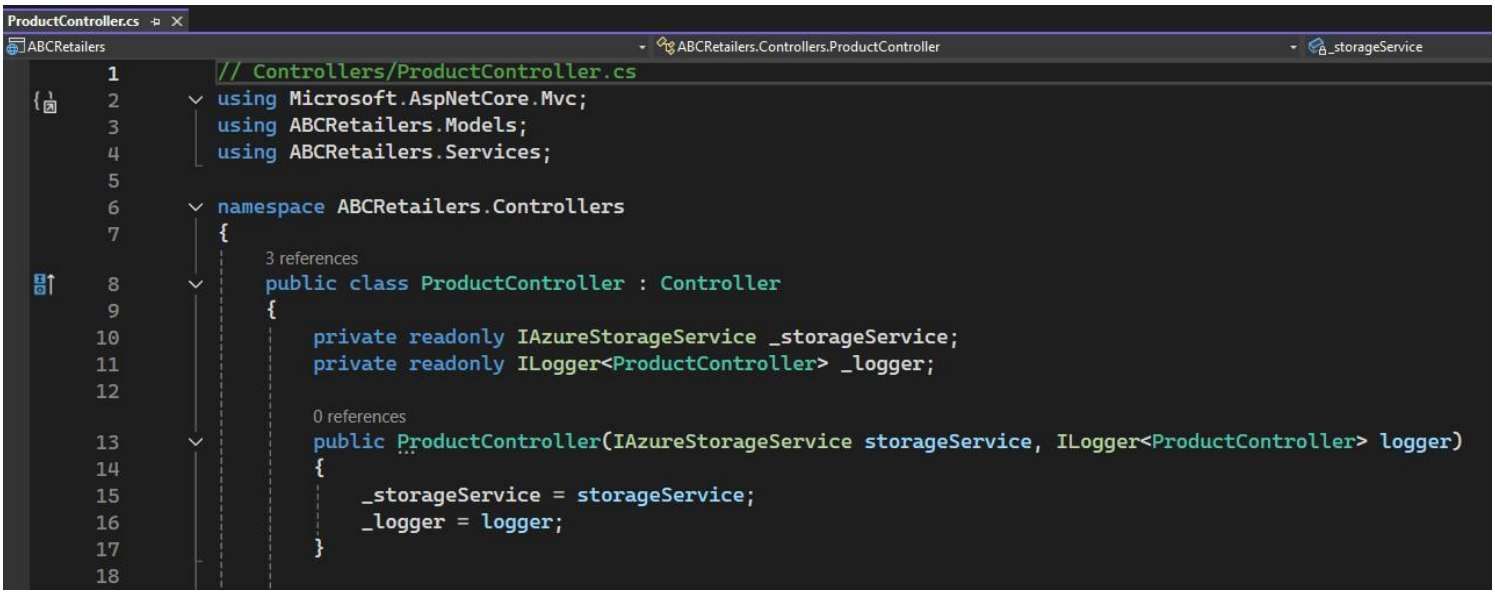
ABCRetailers Web Application - ProductController Code Snapshots

ProductController.cs (Full Code Snapshot) - Difficult to take Full Code Snapshot – Watch Video Recording for Start and End of Code – The Start and End of Code is also captured in the First and Last Snapshot Respectively)

ProductController.cs (IAzureStorageService Action Code Snapshot - NOTE THAT THE PREAMBLE PART OF THE CODE IS CAPTURED HERE)



ProductController.cs (Index Action Code Snapshot)

```
3 references
public async Task<IActionResult> Index()
{
    var products = await _storageService.GetAllEntitiesAsync<Product>();
    return View(products);
}
```

ProductController.cs (Create GET Action Code Snapshot)

```
0 references
public IActionResult Create()
{
    return View();
}
```

ProductController.cs (Create POST Action Code Snapshot)

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create(Product product, IFormFile? imageFile)
{
    // Manual price parsing to fix binding issue
    if (Request.Form.TryGetValue("Price", out var priceFormValue))
    {
        _logger.LogInformation("Raw price from form: '{PriceFormValue}'", priceFormValue.ToString());

        if (decimal.TryParse(priceFormValue, out var parsedPrice))
        {
            product.Price = parsedPrice;
            _logger.LogInformation("Successfully parsed price: {Price}", parsedPrice);
        }
        else
        {
            _logger.LogWarning("Failed to parse price: {PriceFormValue}", priceFormValue.ToString());
        }
    }

    _logger.LogInformation("Final product price: {Price}", product.Price);

    if (ModelState.IsValid)
    {
        try
        {
            if (product.Price <= 0)
            {
                ModelState.AddModelError("Price", "Price must be greater than $0.00");
                return View(product);
            }

            // Upload image if provided
            if (imageFile != null && imageFile.Length > 0)
            {
                var imageUrl = await _storageService.UploadImageAsync(imageFile, "product-images");
                product.ImageUrl = imageUrl;
            }

            await _storageService.AddEntityAsync(product);
            TempData["Success"] = $"Product '{product.ProductName}' created successfully with price {product.Price:C}!";
            return RedirectToAction(nameof(Index));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error creating product");
            ModelState.AddModelError("", $"Error creating product: {ex.Message}");
        }
    }

    return View(product);
}
```

ProductController.cs (Edit GET Action Code Snapshot)

```
0 references
public async Task<IActionResult> Edit(string id)
{
    if (string.IsNullOrEmpty(id))
    {
        return NotFound();
    }

    var product = await _storageService.GetEntityAsync<Product>("Product", id);
    if (product == null)
    {
        return NotFound();
    }

    return View(product);
}
```

ProductController.cs (Edit POST Action Code Snapshot)

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Edit(Product product, IFormFile? imageFile)
{
    // Manual price parsing for edit too
    if (Request.Form.TryGetValue("Price", out var priceFormValue))
    {
        if (decimal.TryParse(priceFormValue, out var parsedPrice))
        {
            product.Price = parsedPrice;
            _logger.LogInformation("Edit: Successfully parsed price: {Price}", parsedPrice);
        }
    }

    if (ModelState.IsValid)
    {
        try
        {
            // Get the original product to preserve ETag
            var originalProduct = await _storageService.GetEntityAsync<Product>("Product", product.RowKey);
            if (originalProduct == null)
            {
                return NotFound();
            }

            // Update properties but keep the original ETag
            originalProduct.ProductName = product.ProductName;
            originalProduct.Description = product.Description;
            originalProduct.Price = product.Price;
            originalProduct.StockAvailable = product.StockAvailable;

            // Upload new image if provided
            if (imageFile != null && imageFile.Length > 0)
            {
                var imageUrl = await _storageService.UploadImageAsync(imageFile, "product-images");
                originalProduct.ImageUrl = imageUrl;
            }

            await _storageService.UpdateEntityAsync(originalProduct);
            TempData["Success"] = "Product updated successfully!";
            return RedirectToAction(nameof(Index));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error updating product: {Message}", ex.Message);
            ModelState.AddModelError("", $"Error updating product: {ex.Message}");
        }
    }
    return View(product);
}
```

ProductController.cs (Delete Action Code Snapshot – NOTE THAT THIS PART OF THE CODE CAPTURES THE TWO CLOSING BRACES)

```
        }

        [HttpPost]
        0 references
        public async Task<IActionResult> Delete(string id)
        {
            try
            {
                await _storageService.DeleteEntityAsync<Product>("Product", id);
                TempData["Success"] = "Product deleted successfully!";
            }
            catch (Exception ex)
            {
                TempData["Error"] = $"Error deleting product: {ex.Message}";
            }

            return RedirectToAction(nameof(Index));
        }
    }
}
```