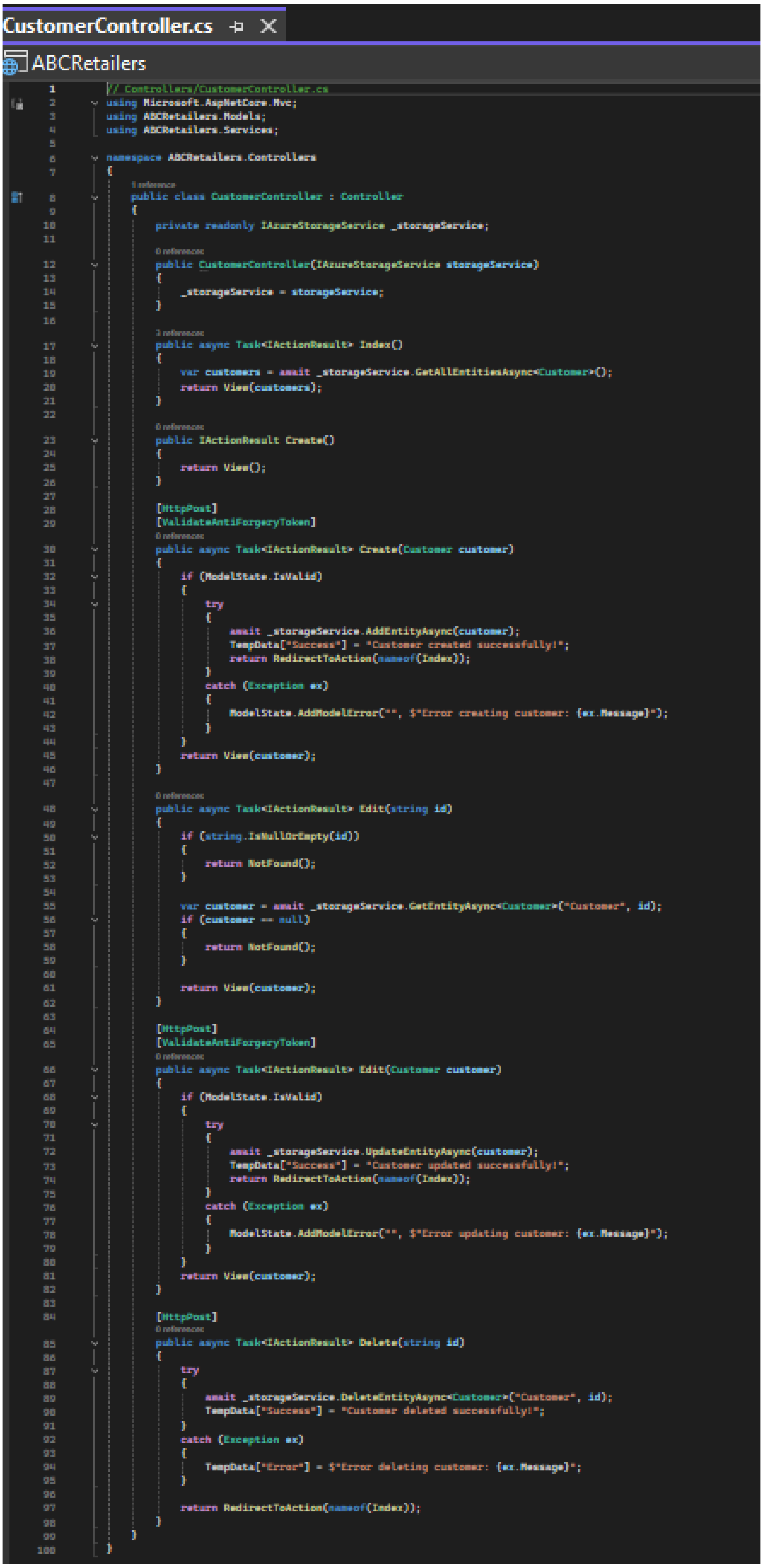
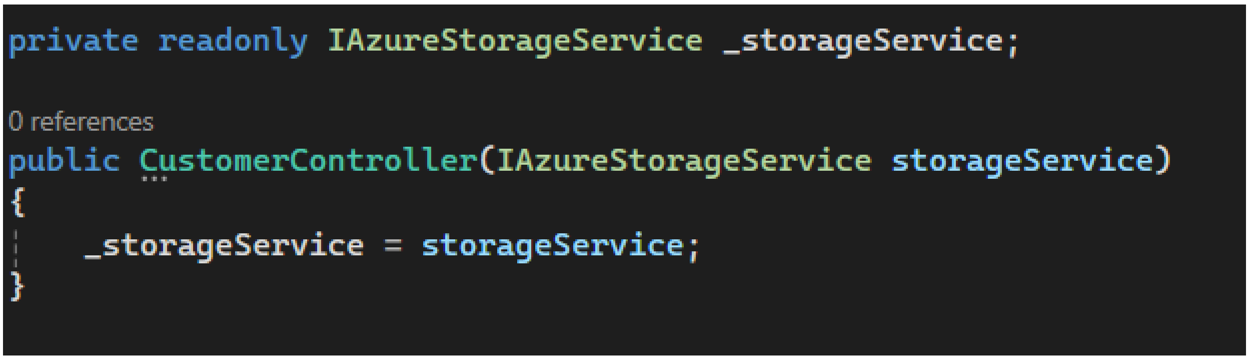
ABCRetailers Web Application - CustomerController Code Snapshots

CustomerController.cs (Full Code Snapshot)



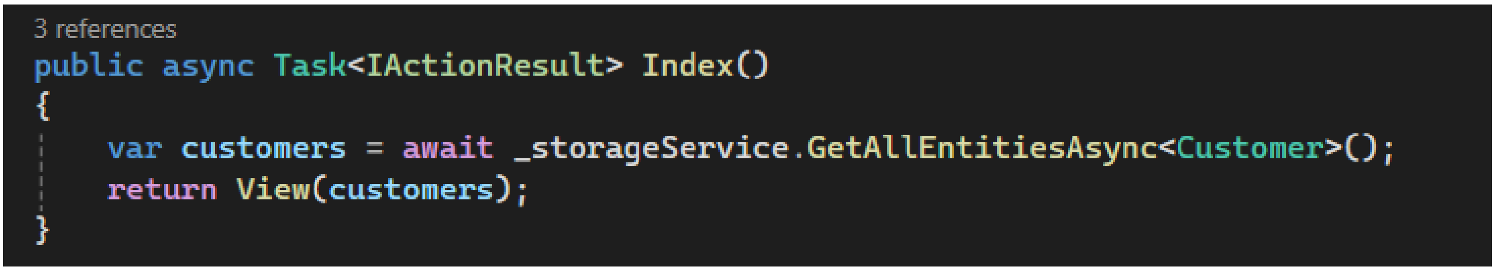
```csharp
// Controllers/CustomerController.cs
using Microsoft.AspNetCore.Mvc;
using ABCRetailers.Models;
using ABCRetailers.Services;

namespace ABCRetailers.Controllers
{
    1 reference
    public class CustomerController : Controller
    {
        private readonly IAzureStorageService _storageService;

        0 references
        public CustomerController(IAzureStorageService storageService)
        {
            _storageService = storageService;
        }

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

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

        [HttpPost]
        [ValidateAntiForgeryToken]
        0 references
        public async Task<IActionResult> Create(Customer customer)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    await _storageService.AddEntityAsync(customer);
                    TempData["Success"] = "Customer created successfully!";
                    return RedirectToAction(nameof(Index));
                }
                catch (Exception ex)
                {
                    ModelState.AddModelError("", $"Error creating customer: {ex.Message}");
                }
            }
            return View(customer);
        }

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

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

            return View(customer);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        0 references
        public async Task<IActionResult> Edit(Customer customer)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    await _storageService.UpdateEntityAsync(customer);
                    TempData["Success"] = "Customer updated successfully!";
                    return RedirectToAction(nameof(Index));
                }
                catch (Exception ex)
                {
                    ModelState.AddModelError("", $"Error updating customer: {ex.Message}");
                }
            }
            return View(customer);
        }

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

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

CustomerController.cs (IAzureStorageService Action Code Snapshot)



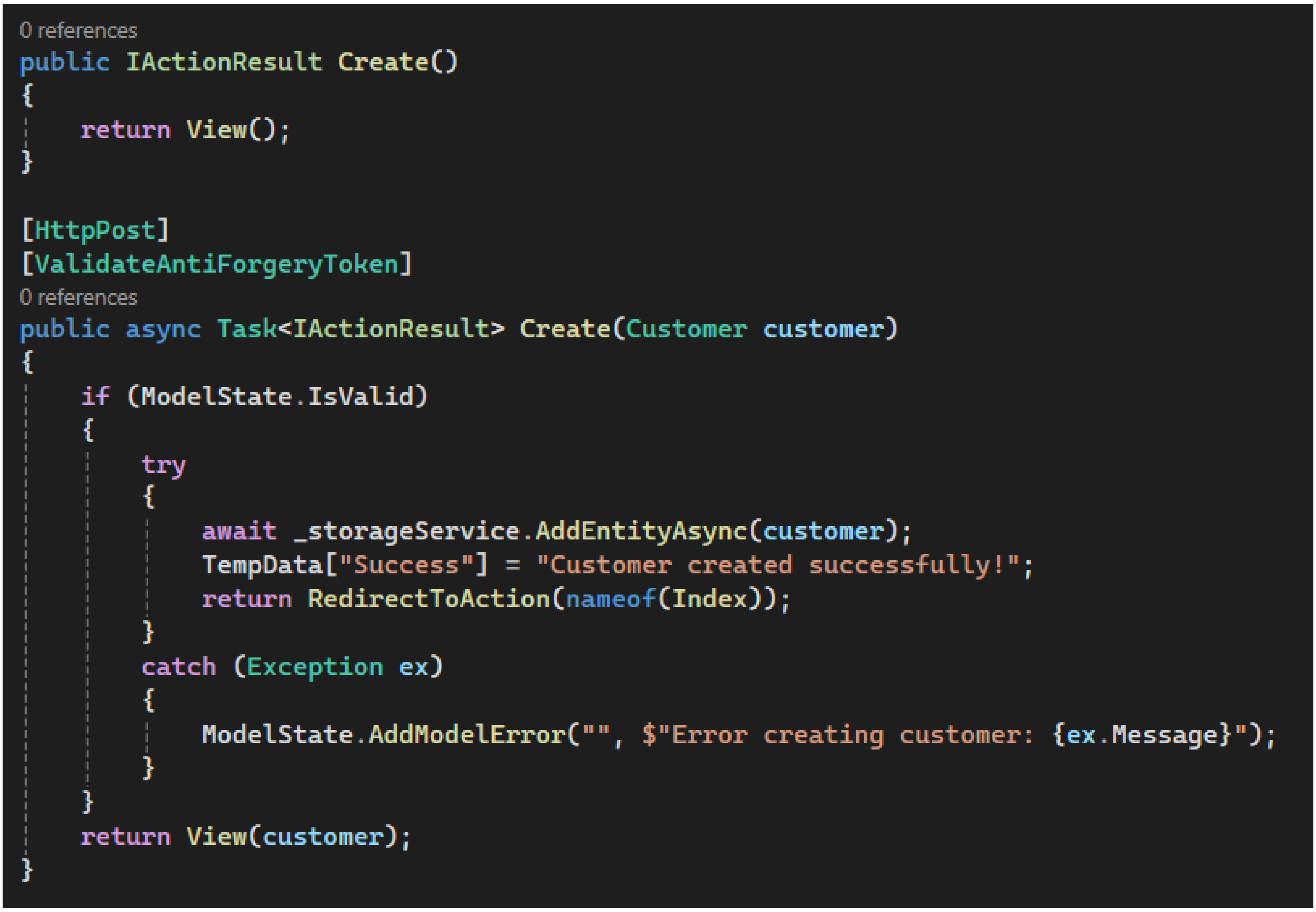
```
private readonly IAzureStorageService _storageService;

0 references
public CustomerController(IAzureStorageService storageService)
{
    _storageService = storageService;
}
```

CustomerController.cs (Index Action Code Snapshot)



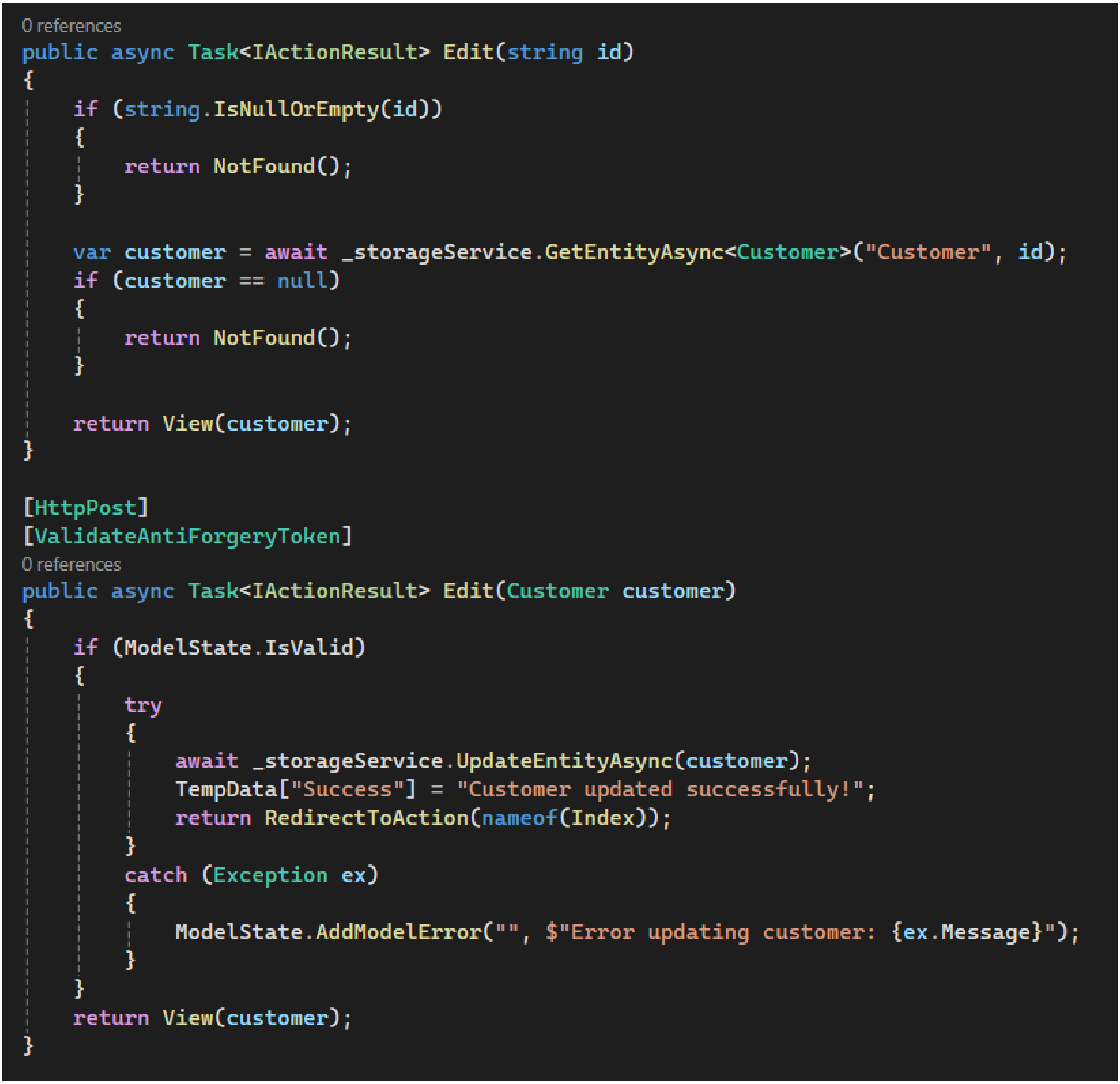
```
3 references
public async Task<IActionResult> Index()
{
    var customers = await _storageService.GetAllEntitiesAsync<Customer>();
    return View(customers);
}
```

CustomerController.cs (Create Action Code Snapshot)



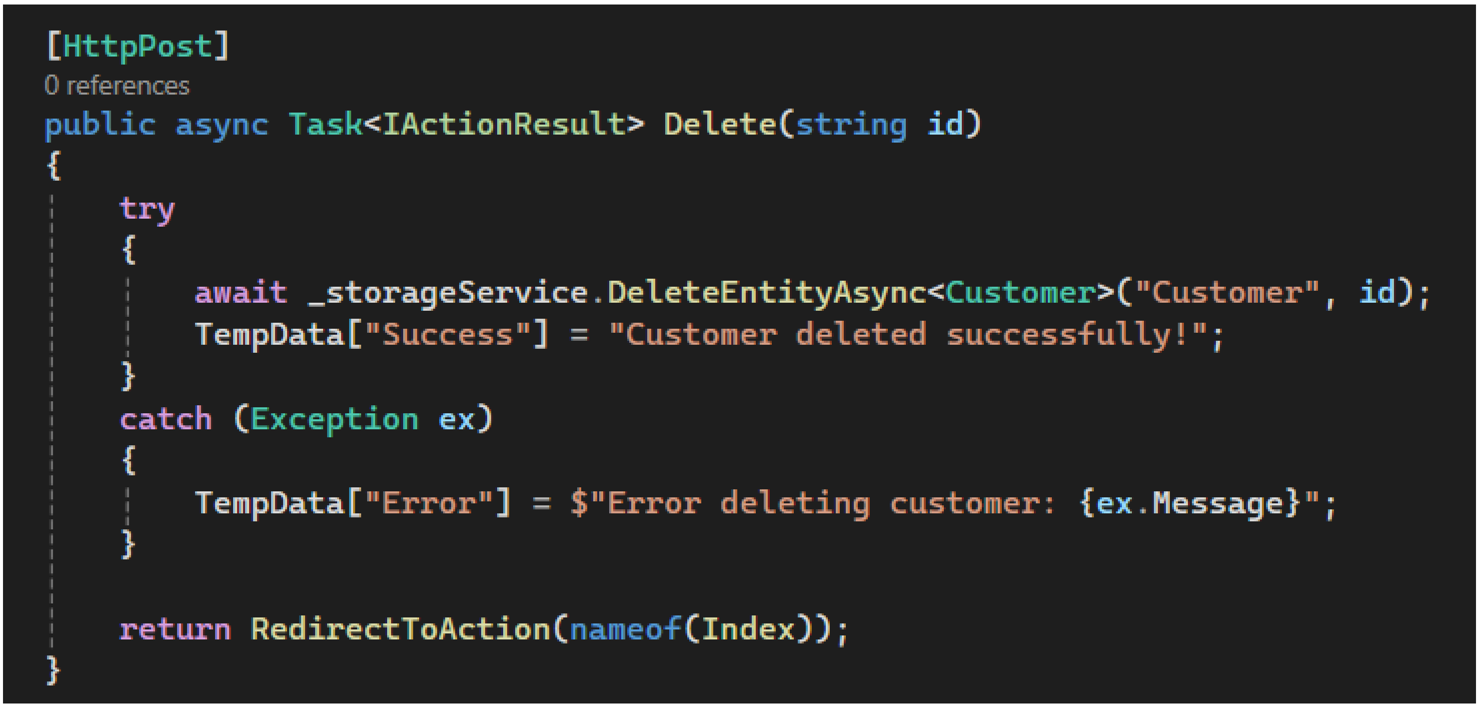
```
0 references
public IActionResult Create()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create(Customer customer)
{
    if (ModelState.IsValid)
    {
        try
        {
            await _storageService.AddEntityAsync(customer);
            TempData["Success"] = "Customer created successfully!";
            return RedirectToAction(nameof(Index));
        }
        catch (Exception ex)
        {
            ModelState.AddModelError("", $"Error creating customer: {ex.Message}");
        }
    }
    return View(customer);
}
```

CustomerController.cs (Edit Action Code Snapshot)



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

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

    return View(customer);
}

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Edit(Customer customer)
{
    if (ModelState.IsValid)
    {
        try
        {
            await _storageService.UpdateEntityAsync(customer);
            TempData["Success"] = "Customer updated successfully!";
            return RedirectToAction(nameof(Index));
        }
        catch (Exception ex)
        {
            ModelState.AddModelError("", $"Error updating customer: {ex.Message}");
        }
    }
    return View(customer);
}
```

CustomerController.cs (Delete Action Code Snapshot)



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

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