

Lập trình web

LẬP TRÌNH VỚI ASP.NET MVC P5

GV: Nguyễn Huy Cường

Email: nh.cuong@hutech.edu.vn

Nội dung: Một số bài toán trong ASP.NET Core MVC

 **Tìm kiếm**

 **Phân trang**

 **Thực hiện giỏ hàng: Đặt hàng, Thanh toán**

 **Xử lý Exception**

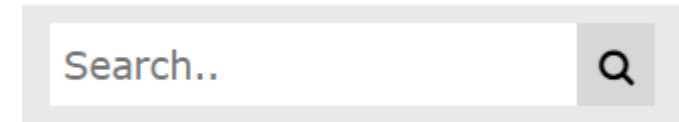
 **Log**

 **Đa ngôn ngữ**

 **Triển khai Website**

1- Bài toán tìm kiếm

- Tìm kiếm giúp nâng cao trải nghiệm người dùng
- Ý tưởng thực hiện :

A search bar with a light gray border. Inside, the text "Search.." is written in a light gray font. To the right of the text is a magnifying glass icon inside a small gray square button.

- 1 – Tạo giao diện **View** cho **Search**
- 2 – Tạo **Action** thực hiện **Search** ở controller: Action này trả về 1 View kết quả
- 3 – Trả về kết quả **search** (searchModel)



Thêm search vào Products

1 – Tạo Search View

```
@using (Html.BeginForm("SearchProducts", "Products", FormMethod.Get))
{
    <input type="text" name="query" placeholder="Tìm Sản phẩm" />
    <input type="submit" value="Search" />
}
```

```
<form action="/Products/SearchProducts" method="get">
    <input type="text" name="query" placeholder="Tìm Sản phẩm" />
    <input type="submit" value="Search" />
</form>
```

Để search có thể đẹp hơn, chúng ta lấy 1 số mẫu khác (VD: button được thay bằng hình) ...

2- Tạo Action *SearchProducts* trong Controller

```
[HttpGet]
0 references
public IActionResult SearchProducts(string query)
{
    try
    {
        if (string.IsNullOrEmpty(query))
            return BadRequest("Search query is required.");
        var result = _context.Products.Where(
            p => p.ProductName.Contains(query) ||
            (p.Description != null && p.Description.Contains(query))).ToList();
        return View("Index", result);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

3 – Trả về View từ kết quả

2- Bài toán phân trang (pagination)

- Phân trang: tăng trải nghiệm người dùng trong việc tìm hiểu thông tin...Việc chia nhỏ thông tin bằng việc phân trang (Pagination) cũng giúp hạn chế quá tải thông tin trên cùng một site.
- Ý tưởng thực hiện:

Sử dụng **Skip**, **Take** để lấy danh sách sản phẩm ở trang **pageNumber**

Với **PageSize** là số lượng item/trang

```
public async Task<IActionResult> PagingNoLibrary(int pageNumber)
{
    int pageSize = 10;
    IQueryable<Product> productsQuery = _context.Products.Include(p => p.Category);
    var pagedProducts = await productsQuery.Skip((pageNumber - 1) * pageSize)
                                           .Take(pageSize)
                                           .ToListAsync();

    return View(pagedProducts);
}
```

ProductName	Image	Description	Price	Category	
Name25		Description25	25.00	1	Edit Details Delete
Name26		Description26	26.00	1	Edit Details Delete
Name27		Description27	27.00	1	Edit Details Delete
Name28		Description28	28.00	1	Edit Details Delete
Name29		Description29	29.00	1	Edit Details Delete
Name3		Description3	3.00	1	Edit Details Delete
Name30		Description30	30.00	1	Edit Details Delete
Name31		Description31	31.00	1	Edit Details Delete
Name32		Description32	32.00	1	Edit Details Delete
Name33		Description33	33.00	1	Edit Details Delete

Phân trang với PaginatedList

- Không sử dụng thư viện

<https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/sort-filter-page?view=aspnetcore-8.0>

1- Tạo lớp **PaginatedList** sử dụng take, skip và tính PageIndex, TotalPages

PageIndex (vị trí trang)

pageSize (số item/trang)

`TotalPages = (int)Math.Ceiling(count / (double)pageSize;`

2- Thêm Paging trong Action

```
IQueryable<Product> productsQuery = _context.Products.Include(p => p.Category);
var paginatedProducts = await PaginatedList<Product>.CreateAsync(productsQuery, pageNumber, pageSize);
return View(paginatedProducts);
```

3- Thêm Paging trong View & tạo view chuyển trang

@model **PaginatedList**<DemoST6.Models.Product>

```
<a asp-action="Index"
    asp-route-pageNumber="@((Model.PageIndex - 1))" class="btn btn-
default @prevDisabled">
    Previous
</a>
<a asp-action="Index"
    asp-route-pageNumber="@((Model.PageIndex + 1))"
    class="btn btn-default @nextDisabled">
    Next
</a>
```



phân trang PaginatedList

1- Tạo lớp **PaginatedList** sử dụng take, skip và tính **PageIndex**, **TotalPages**

2- Thêm Paging trong Action

```
public class PaginatedList<T> : List<T>
{
    public int PageIndex { get; private set; }
    public int TotalPages { get; private set; }
    public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)
    {
        PageIndex = pageIndex;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);

        this.AddRange(items);
    }

    public bool HasPreviousPage => PageIndex > 1;

    public bool HasNextPage => PageIndex < TotalPages;

    public static async Task<PaginatedList<T>> CreateAsync(IQueryable<T> source,
int pageIndex, int pageSize)
    {
        var count = await source.CountAsync();
        var items = await source.Skip((pageIndex - 1) *
pageSize).Take(pageSize).ToListAsync();
        return new PaginatedList<T>(items, count, pageIndex, pageSize);
    }
}
```

```
public async Task<IActionResult> Index(int pageNumber = 1)
{
    int pageSize = 10;
    IQueryable<Product> productsQuery = _context.Products.Include(p => p.Category);
    var paginatedProducts = await PaginatedList<Product>.CreateAsync(productsQuery,
pageNumber, pageSize);
    return View(paginatedProducts);
}
```



phân trang PaginatedList

3- Điều chỉnh view

❑ Thay `IEnumerable<Product>` bằng
`PaginatedList<Product>` ở `@model`

❑ Thêm phần hiển thị chuyển trang
(previous / next)

```
<a asp-action="Index"
  asp-route-pageNumber="@((Model.PageIndex - 1))" class="btn btn-default
@prevDisabled">
  Previous
</a>
<a asp-action="Index"
  asp-route-pageNumber="@((Model.PageIndex + 1))"
class="btn btn-default @nextDisabled">
  Next
</a>
```



Previous Next

```
@model PaginatedList<DemoST6.Models.Product>
```

```
<div>
  <nav aria-label="Page navigation">
    <ul class="pagination">
      @for (int i = 1; i <= @Model.TotalPages; i++)
      {
        <li class="page-item @(i == @Model.PageIndex ?
"active" : "")">
          <a class="page-link"
href="@Url.Action("Index", new { pageNumber = i })">@i</a>
        </li>
      }
    </ul>
  </nav>
</div>
```



1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	----	----

Phân trang sử dụng thư viện: X.PagedList.Mvc.Core

- Tham khảo:

3- Thực hiện giỏ hàng

- Giỏ hàng chính là cầu nối trung gian giữa trang sản phẩm và quy trình thanh toán. Do đó, khách hàng có đi đến quyết định thanh toán và hoàn thành đơn hàng hay không phụ thuộc vào sự tối ưu của tính năng giỏ hàng.
- Ý tưởng thực hiện:
 - (1) Tạo **data model** tương ứng cho giỏ hàng (**CartItem**)
 - (2) Tạo **Controller** cho giỏ hàng (**ShoppingCartController**).
 - Sử dụng **Session** để lưu **List<CartItem>**
 - Thực hiện các **Action**:
 - ❑ **AddToCart**: Thêm vào giỏ hàng
 - ❑ **Index**: Lấy ds **CartItem** trong giỏ hàng
 - ❑ **UpdateCart**: Cập nhật số lượng, tổng số lượng, số tiền
 - ❑ **DeleteCart**: Xóa 1 CartItem trong giỏ hàng
 - ❑ **DeleteAllCart**: Xóa giỏ hàng (remove session)

```
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

.....
app.UseSession();
```

DEMO Tạo CartItem

(1) Tạo model **CartItem** tương ứng cho giỏ hàng để lưu trữ thông tin sản phẩm và số lượng đặt hàng

```
public class CartItem
{
    //Product Info
    public required string ProductId { get; set; }

    public string ProductName { get; set; }

    public string? Image { get; set; }

    public decimal Price { get; set; }

    //Quantity
    public int Quantity { get; set; }
}
```



Tạo ShoppingCart Controller

(2) Tạo **ShoppingCartController** để thực hiện

2.1 Sử dụng **Session** để lấy/lưu danh sách **CartItem** => Viết vào 1 hàm để tái sử dụng

```
List<CartItem>? GetCartItems()
{
    var session = HttpContext.Session;
    string jsoncart = session.GetString("cart");
    if (jsoncart != null)
    {
        return JsonConvert.DeserializeObject<List< CartItem >>(jsoncart);
    }
    return new List<CartItem>();
}

void SaveCartSession(List<CartItem> ls)
{
    string jsoncart = JsonConvert.SerializeObject(ls);
    HttpContext.Session.SetString("cart", jsoncart);
}
```



Thực hiện AddToCart: Thêm vào giỏ hàng

(2.1) Viết AddToCart: thêm 1 sản phẩm vào giỏ hàng và redirect tới giỏ hàng. Gọi đặt hàng để tới AddToCart ở danh sách sản phẩm

Ý tưởng: Nếu sản phẩm chưa có trong giỏ hàng -> Add vào giỏ hàng.

Ngược lại sẽ chỉ tăng số lượng

localhost:7136/products

ProductName	Image	Description	Price	Category	
prduct 2	11	22	33.00	2	Edit Details Delete Đặt hàng
1111	22	33	111.00	1	Edit Details Delete Đặt hàng
Name1		Description1	1.00	1	Edit Details Delete Đặt hàng
Name10		Description10	10.00	1	Edit Details Delete Đặt hàng
Name100		Description100	100.00	1	Edit Details Delete Đặt hàng

```

public ActionResult AddToCart(string id)
{
    Product? itemProduct = _context.Products.FirstOrDefault(p=>p.ProductId == id);
    if (itemProduct == null)
        return BadRequest("Sản phẩm không tồn tại");
    var carts = GetCartItems();
    var findCartItem = carts.FirstOrDefault(p=>p.ProductId.Equals(id));
    if (findCartItem == null)
    {
        //Th thêm mới vào giỏ hàng
        findCartItem = new CartItem()
        {
            ProductId = itemProduct.ProductId,
            ProductName = itemProduct.ProductName,
            Image = itemProduct.Image,
            Price = itemProduct.Price,
            Quantity = 1
        };
        carts.Add(findCartItem);
    }
    else
        findCartItem.Quantity++;
    return RedirectToAction("Index");
}

```




Thực hiện Index – Xem giỏ hàng

(2.2) Viết Action **Index** để Lấy giỏ hàng

Ý tưởng: Lấy giỏ hàng (từ session), tính tổng số lượng, đơn giá tương ứng và tới view giỏ hàng

```
public ActionResult Index()
{
    var carts = GetCartItems();
    return View(carts);
}
```

Giỏ hàng

ProductName	Image	Quantity	Price	Total			
Laptop		1	33.00	33	Chi tiết	Xóa	Cập nhật



Điều chỉnh Index – Xem giỏ hàng (lấy tổng số lượng, giá)- đặt hàng

(2.2) Điều chỉnh thêm tổng số lượng và tổng tiền

THÔNG TIN GIỎ HÀNG

Tên Sản phẩm	Hình ảnh	Số lượng	Đơn giá	Thành tiền	
Laptop		<input type="text" value="1"/>	33	33	Chi tiết Xóa Cập nhật
				Tổng Số: 1	Tổng tiền: 33 VNĐ
					Xóa Giỏ Hàng
Đặt hàng					

```
public ActionResult Index()
{
    var carts = GetCartItems();
    ViewBag.TongTien = carts.Sum(p=>p.Price * p.Quantity);
    ViewBag.TongSoLuong = carts.Sum(p => p.Quantity);
    return View(carts);
}
```



View giỏ hàng

```
@model IEnumerable<DemoST6.Models.CartItem>
@{
```

```
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<h2 style="text-align:center">THÔNG TIN GIỎ HÀNG</h2>
```

```
<table class="table" align="center" border="1">
```

```
    <thead>
```

```
        <tr style="text-align:center; font-weight:bold">
```

```
            <td> Tên Sản phẩm </td>
```

```
            <td> Hình ảnh </td>
```

```
            <td> Số lượng </td>
```

```
            <td> Đơn giá </td>
```

```
            <td> Thành tiền </td>
```

```
            <td width="50px"></td>
```

```
            <td width="50px"></td>
```

```
            <td width="50px"></td>
```

```
        </tr>
```

```
    </thead>
```

```
    <tbody>
```

```
@foreach (var item in Model) {
```

```
    <tr style="text-align:center; font-weight:bold">
```

```
        <td> @Html.DisplayFor(modelItem => item.ProductName) </td>
```

```
        <td>  </td>
```

```
        <td> <input type="number" min="1" name="txtQuantity" value="@item.Quantity" style="background-color:yellow" /> </td>
```

```
        <td> @string.Format("{0:0,0}", item.Price) </td>
```

```
        <td> @string.Format("{0:0,0}", item.Price * item.Quantity) </td>
```

```
        <td> <a asp-action="Details" asp-route-id="@item.ProductId">Chi tiết</a> </td>
```

```
        <td> <a asp-action="DeleteCart" asp-route-id="@item.ProductId">Xóa</a></td>
```

```
        <td> <a asp-action="UpdateCart" asp-route-id="@item.ProductId" asp-route-quantity="@item.Quantity">Cập nhật</a></td>
```

```
    </tr>
```

```
}
```

```
<tr style="font-weight:bold; text-align:right; color:red">
```

```
    <td colspan="5"> Tổng Số: @ViewBag.TongSoLuong </td>
```

```
    <td colspan="5"> Tổng tiền: @String.Format("{0:0,0}",
```

```
    ViewBag.TongTien) VNĐ</td>
```

```
</tr>
```

```
<tr style="font-weight:bold; color:blue; text-align:right">
```

```
    <td colspan="9"> <a asp-action="Delete">Xóa Giỏ Hàng</a>
```

```
</td>
```

```
</tr>
```

```
<tr style="font-weight:bold; color:blue; text-align:right">
```

```
    <td colspan="9" align="center"><a asp-action="Order">Đặt
```

```
hàng</a> </td>
```

```
</tr>
```

```
</tbody>
```

```
</table>
```





UpdateCart– Cập nhật giỏ hàng

(2.3) Viết Action **UpdateCart** để cập nhật thay đổi giỏ hàng: số lượng, thành tiền

Ý tưởng:

(1) Khi cập nhật: Lấy thông tin mã, số lượng và gọi về Controller thực hiện

THÔNG TIN GIỎ HÀNG

Tên Sản phẩm	Hình ảnh	Số lượng	Đơn giá	Thành tiền	
Laptop		<div><div>100</div><div></div></div>	33	33	Chi tiết Xóa Cập nhật
				Tổng Số: 1	Tổng tiền: 33 VNĐ
					Xóa Giỏ Hàng
Đặt hàng					

`<td> Cập nhật</td>`

`<td> <input type="number" min="1" id="txtQuantity_@item.ProductId" value="@item.Quantity" style="background-color:yellow" /> </td>`

```
public ActionResult UpdateCart(string id, int quantity)
{
    var carts = GetCartItems();
    var findCartItem = carts.FirstOrDefault(p=>p.ProductId == id);
    if(findCartItem!= null)
    {
        findCartItem.Quantity = quantity;
        SaveCartSession(carts);
    }
    return RedirectToAction("Index");
}
```

```
<script>
    function updateCart(productId) {
        var quantity = document.getElementById('txtQuantity_' +
productId).value;
        window.location.href = '/ShoppingCart/UpdateCart?id=' + productId +
'&quantity=' + quantity;
    }
</script>
```

(2.4) Viết Action **DeleteCart** để xóa mục trong giỏ hàng

Ý tưởng:

 <td> <a asp-action="DeleteCart" asp-route-id="@item.ProductId">Xóa</td> |

THÔNG TIN GIỎ HÀNG

Tên Sản phẩm	Hình ảnh	Số lượng	Đơn giá	Thành tiền	
Laptop		<input type="text" value="1"/>	33	33	Chi tiết Xóa Cập nhật
1111		<input type="text" value="1"/>	111	111	Chi tiết Xóa Cập nhật
				Tổng Số: 2	Tổng tiền: 144 VND
Xóa Giỏ Hàng					
Đặt hàng					

```
public ActionResult DeleteCart(string id)
{
    var carts = GetCartItems();
    var findCartItem = carts.FirstOrDefault(p => p.ProductId == id);
    if(findCartItem != null)
    {
        carts.Remove(findCartItem);
        SaveCartSession(carts);
    }
    return RedirectToAction("Index");
}
```

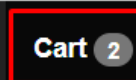


Lấy số lượng cart trong giỏ hàng trên layout

(2.5) Sử dụng **ViewComponent** để tạo **CartSummary** giỏ hàng từ Layout

Tham khảo: <https://cazton.com/blogs/technical/view-components-in-asp-net-core>

Kết quả :



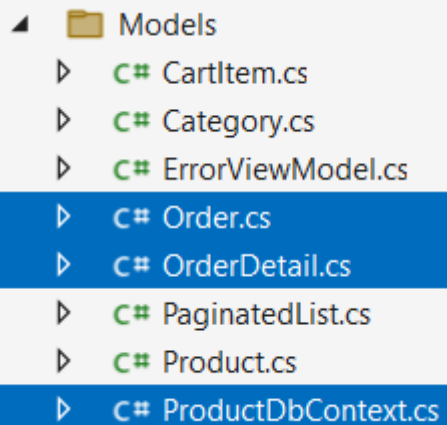
Hello member1@gmail.com!

Log off

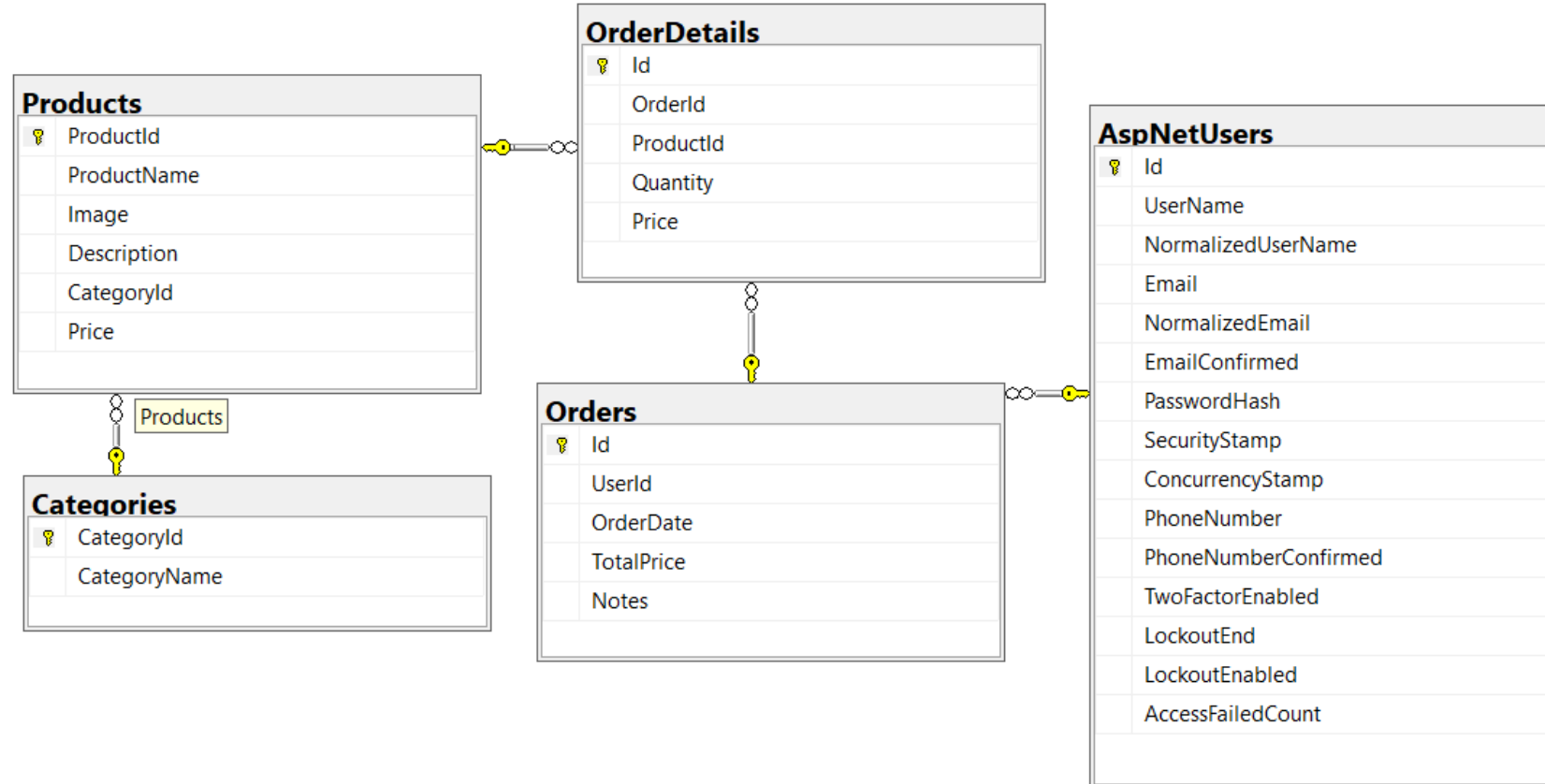
2.6 Đặt hàng, thanh toán

❑ Thêm Đơn hàng “**Order**”, và chi tiết đơn hàng “**OrderDetail**” để lưu trữ việc đặt hàng và chi tiết đơn hàng

⇒ cập nhật lại **Models** và **CSDL** tương ứng (code first -> DB)



- Models
 - ▶ C# CartItem.cs
 - ▶ C# Category.cs
 - ▶ C# ErrorViewModel.cs
 - ▶ C# Order.cs
 - ▶ C# OrderDetail.cs
 - ▶ C# PaginatedList.cs
 - ▶ C# Product.cs
 - ▶ C# ProductDbContext.cs



Cập nhật CSDL

- ❑ Thêm Order
- ❑ Thêm OrderDetail
- ❑ Thêm DbContext
- ❑ Chạy PMC

add-migration versionOrders

update-database

```
public class Order
{
    public int Id { get; set; }
    public string? UserId { get; set; }
    public DateTime OrderDate { get; set; }
    public decimal TotalPrice { get; set; }
    public string? Notes { get; set; }
    public IdentityUser? User { get; set; }
    public List<OrderDetail>? OrderDetails { get; set; }
}
```

```
public class OrderDetail
{
    public int Id { get; set; }
    public int OrderId { get; set; }
    public int ProductId { get; set; }
    public int Quantity { get; set; }
    public decimal Price { get; set; }
    public required Order Order { get; set; }
    public Product Product { get; set; }
}
```

```
public partial class ProductDbContext : IdentityDbContext
{
    0 references
    public ProductDbContext()
    {
    }

    0 references
    public ProductDbContext(DbContextOptions<ProductDbContext> options)
        : base(options)
    {
    }

    8 references
    public virtual DbSet<Category> Categories { get; set; }

    17 references
    public virtual DbSet<Product> Products { get; set; }

    0 references
    public virtual DbSet<Order> Orders { get; set; }

    0 references
    public virtual DbSet<OrderDetail> OrderDetails { get; set; }
}
```

Thực hiện ShoppingCartController – Đặt hàng (Order)

Thực hiện Order - Đặt hàng

Ý tưởng: Thêm mới vào 2 bảng **Order**, **OrderDetail**

THÔNG TIN GIỎ HÀNG

Tên Sản phẩm	Hình ảnh	Số lượng	Đơn giá	Thành tiền	
Name1		<input type="text" value="10"/>	01	10	Chi tiết Xóa Cập nhật
Name10		<input type="text" value="2"/>	10	20	Chi tiết Xóa Cập nhật
				Tổng Số: 12	Tổng tiền: 30 VND
				Xóa Giỏ Hàng	
				Đặt hàng	

```
public async Task<IActionResult> Order()
{
    try
    {
        var carts = GetCartItems();
        var user = await _userManager.GetUserAsync(User);

        var order = new Order
        {
            UserId = user != null ? user.Id : null,
            OrderDate = DateTime.UtcNow,
            TotalPrice = carts.Sum(i => i.Price * i.Quantity),
            OrderDetails = carts.Select(item => new OrderDetail
            {
                ProductId = item.ProductId,
                Quantity = item.Quantity,
                Price = item.Price
            }).ToList()
        };
        _context.Orders.Add(order);
        await _context.SaveChangesAsync();

        //xóa session giỏ hàng
        HttpContext.Session.Remove(CARTKEY);

        return View("OrderCompleted", order);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

```
OrderCompleted.cshtml
1 @model DemoST6.Models.Order
2 @{
3     ViewData["Title"] = "Order";
4     Layout = "~/Views/Shared/_Layout.cshtml";
5 }
6 <h2>Order Completed</h2>
7 <p>Your order with ID @Model.Id has been placed successfully.</p>
8 <div>
9     <a asp-action="Index" asp-controller="Products">Về trang chủ</a>
10 </div>
```



Thanh toán

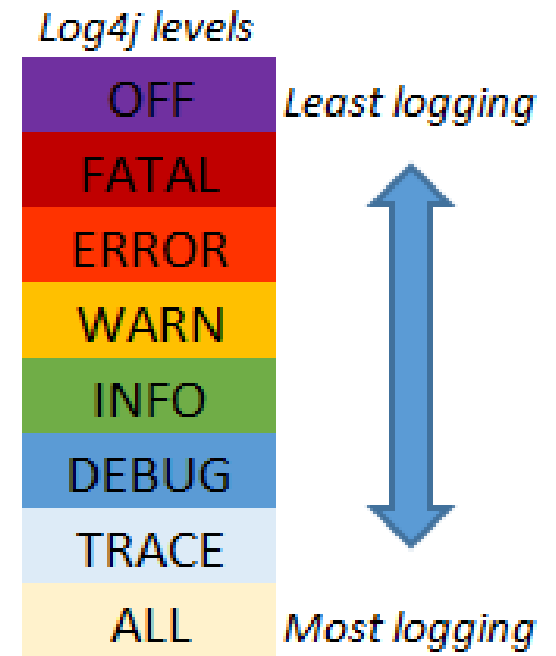
Tham khảo: Thanh toán bằng **Momo** qua file gửi kèm

4- Quản lý Exception

- Tham khảo: <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/error-handling?view=aspnetcore-8.0>
- Một số gợi ý
 - ❑ Có thể xử lý lỗi bằng code như các khối lệnh **try { ... } catch { ... }**.
 - ❑

5- Log

- Tại sao phải ghi Log?
- Giải pháp trong ASP.NET Core





Sử dụng log tích hợp vào website

- Các mode log

```
<rules>
```

```
<logger name="*" minlevel="DEBUG" writeTo="logfile" />
```

```
</rules>
```

```

148 System.Threading.ThreadAbortException: Thread was being aborted.
149   at System.Threading.Thread.AbortInternal()
150   at System.Threading.Thread.Abort(Object stateInfo)
151   at System.Web.HttpResponse.AbortCurrentThread()
152   at System.Web.HttpResponse.End()
153   at System.Web.HttpServerUtility.Transfer(String path, Boolean preserveForm)
154   at System.Web.HttpServerUtility.Transfer(String path)
155   at FTSS.Web.Admin.DownloadCustomer.Page_Load(Object sender, EventArgs e) in E:\EX_FTSS\Source\FTSS.Web\DownloadCustomer.a
156 2018-12-24 11:18:47,150 [FTSS.Utilities.Common] [1] DEBUG FTSS.Utilities.Common - Init log4net completed
157 2018-12-24 11:18:47,163 [FTSS.Utilities.Common] [1] DEBUG FTSS.Utilities.Common - InitLog4Net completed
158 2018-12-24 11:18:49,935 [FTSS.Web.Admin.Upload] [6] ERROR FTSS.Web.Admin.Upload - Error Page_Load
159 System.Threading.ThreadAbortException: Thread was being aborted.
160   at System.Threading.Thread.AbortInternal()
161   at System.Threading.Thread.Abort(Object stateInfo)
162   at System.Web.HttpResponse.AbortCurrentThread()
163   at System.Web.HttpResponse.End()
164   at System.Web.HttpServerUtility.Transfer(String path, Boolean preserveForm)
165   at System.Web.HttpServerUtility.Transfer(String path)
166   at FTSS.Web.Admin.Upload.Page_Load(Object sender, EventArgs e) in E:\EX_FTSS\Source\FTSS.Web\Admin\Upload.aspx.cs:line 79
167 2018-12-24 11:18:58,955 [FTSS.Web.Admin.Upload] [6] INFO FTSS.Web.Admin.Upload - end page load
168 2018-12-24 11:19:37,886 [FTSS.Web.Admin.Upload] [7] INFO FTSS.Web.Admin.Upload - Request.Files file
169 2018-12-24 11:19:37,891 [FTSS.Web.Admin.Upload] [7] DEBUG FTSS.Web.Admin.Upload - postedFile contentLength =1477486. postedF

```

6- Đa ngôn ngữ (Multiple language)

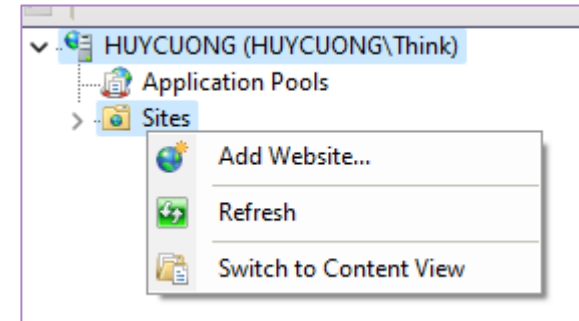
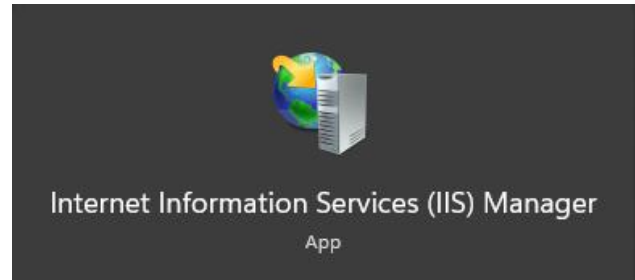
- Tạo các file resources tương ứng với ngôn ngữ: *Resource.resx*, *Resource.Ja.resx*, *Resource.en.resx*...
- <https://www.youtube.com/watch?v=G1uqsvh2JNo>

7 – Triển khai website

- Triển khai Website

- **Localhost**

- **IIS manager**
- **Sites/ Add website**
- **Nhập các thông số cấu hình: siteName, Physical path, Port...**



- Trên môi trường **Internet**: Azure/ các host miễn phí (someeee)

chú ý: thông tin web.config trở đúng csdl

ASM 7: (BTVN) Thực hiện trong BookDB

❑ Thêm CSDL bảng Đơn hàng “**Order**”, và chi tiết đơn hàng “**OrderDetail**”

Thực hiện các yêu cầu quản lý sách (tiếp bài 6) có các chức năng:

1 – Tìm kiếm sách (theo tác giả, tựa sách)

2 – Phân trang (10 sách / trang)

3 – Đặt hàng , thanh toán

có chức năng: cập nhật, tiếp tục mua hàng, xóa mục trong giỏ hàng, xóa giỏ hàng, thanh toán

4 – Thanh toán bằng Momo

5– Triển khai website trên internet