Code

Models

Order

```
public class Order
{
    public int OrderId { get; set; }

    public string UserId { get; set; }

    public DateTime CreatedTime { get; set; }

    public Address DeliveryAddress { get; set; } = new Address();

    public List<Pizza> Pizzas { get; set; } = new List<Pizza>();

    public decimal GetTotalPrice() => Pizzas.Sum(p => p.GetTotalPrice());

    public string GetFormattedTotalPrice() => GetTotalPrice().ToString("0.00");
}
```

Address

```
public class Address
{
    public int Id { get; set; }
    [Required,MinLength(3,ErrorMessage = "长度应大于3"),MaxLength(100,ErrorMessage = "长度应小于100")]
    public string Name { get; set; }
    [Required, MinLength(5, ErrorMessage = "长度应大于5"), MaxLength(100, ErrorMessage = "长度应小于100")]
    public string Line1 { get; set; }
    [MaxLength(100, ErrorMessage = "长度应小于100")]
    public string Line2 { get; set; }
    [Required,MinLength(3, ErrorMessage = "长度应大于3"),MaxLength(50, ErrorMessage = "长度应小于50")]
    public string City { get; set; }
    [Required, MinLength(3, ErrorMessage = "长度应大于3"), MaxLength(20, ErrorMessage = "长度应小于20")]
    public string Region { get; set; }
    [Required,RegularExpression(@"^([0-9]{5})$",ErrorMessage = "邮政编码为5位数字")]
    public string PostalCode { get; set; }
}
```

OrderWithStatus

```
public class OrderWithStatus
{
    //模拟的准备时间和外送时间
    public readonly static TimeSpan PreparationDuration = TimeSpan.FromSeconds(10);
    public readonly static TimeSpan DeliveryDuration = TimeSpan.FromMinutes(1);
    //Unrealistic, but more interesting to watch

    public Order Order { get; set; }
    public string StatusText { get; set; }

    public bool IsDelivered => StatusText == "Delivered";

    public static OrderWithStatus FromOrder(Order order)
    {
        // To simulate a real backend process, we fake status updates based on the amount
        // of time since the order was placed
        string statusText;
        var dispatchTime = order.CreatedTime.Add(PreparationDuration);

        if (DateTime.Now < dispatchTime)
        {
            statusText = "Preparing";
        }
        else if (DateTime.Now < dispatchTime + DeliveryDuration)
        {
            statusText = "Out for delivery";
        }
        else
        {
            statusText = "Delivered";
        }

        return new OrderWithStatus
        {
            Order = order,
            StatusText = statusText
        };
    }


}
```

Pizza

```
/// <summary>
/// Represents a customized pizza as part of an order
/// </summary>
public class Pizza
{
    //public const int DefaultSize = 12;
    public int DefaultSize { get; set; } = 12;
    public const int MinimumSize = 9;
    public const int MaximumSize = 17;

    public int Id { get; set; }

    public int OrderId { get; set; }

    public PizzaSpecial Special { get; set; }

    public int SpecialId { get; set; }

    public int Size { get; set; }
    //一对多导航属性
    public List<PizzaTopping> Toppings { get; set; }

    public decimal GetBasePrice()
    {
        return ((decimal)Size / (decimal)DefaultSize) * Special.BasePrice;
    }

    public decimal GetTotalPrice()
    {
        return GetBasePrice();
    }

    public string GetFormattedTotalPrice()
    {
        return GetTotalPrice().ToString("0.00");
    }
}
```

PizzaSpecial

```
/// <summary>
/// Represents a pre-configured template for a pizza a user can order
/// </summary>
public class PizzaSpecial
{
    public int Id { get; set; }

    public string Name { get; set; }

    public decimal BasePrice { get; set; }

    public string Description { get; set; }

    public string ImageUrl { get; set; }

    public string GetFormattedBasePrice() => BasePrice.ToString("0.00");
}
```

PizzaTopping

```
public class PizzaTopping
{
    public Topping Topping { get; set; }

    public int ToppingId { get; set; }

    public int PizzaId { get; set; }
}
```

Topping

```
public class Topping
{
    public int Id { get; set; }

    public string Name { get; set; }

    public decimal Price { get; set; }

    public string GetFormattedPrice() => Price.ToString("0.00");
}
```

UserInfo

```
public class UserInfo
{
    public bool IsAuthenticated { get; set; }

    public string Name { get; set; }
}
```

Data

PizzaStoreContext

```
/// <summary>
/// 此类创建可用于注册数据库服务的数据库上下文。
/// 该上下文还让我们拥有将访问数据库的控制器。
/// </summary>
public class PizzaStoreContext:DbContext
{
    public PizzaStoreContext(DbContextOptions options) : base(options)
    {
    }
    public DbSet<PizzaSpecial> Specials { get; set; }

    public DbSet<Order> Orders { get; set; }
    public DbSet<Pizza> Pizzas { get; set; }
    public DbSet<Topping> Toppings { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        // Configuring a many-to-many special -> topping relationship that is friendly for serialization
        modelBuilder.Entity<PizzaTopping>().HasKey(pst => new { pst.PizzaId, pst.ToppingId });
        modelBuilder.Entity<PizzaTopping>().HasOne<Pizza>().WithMany(ps => ps.Toppings);
        modelBuilder.Entity<PizzaTopping>().HasOne(pst => pst.Topping).WithMany();
    }

}
```

SeedData

```
public static class SeedData
{
    public static void Initialize(PizzaStoreContext db)
    {
        var specials = new PizzaSpecial[]
        {
            new PizzaSpecial()
            {
                Name = "Basic Cheese Pizza",
                Description = "It's cheesy and delicious. Why wouldn't you want one?",
                BasePrice = 9.99m,
                ImageUrl = "img/pizzas/cheese.jpg",
            },
            new PizzaSpecial()
            {
                Id = 2,
                Name = "The Baconatorizor",
                Description = "It has EVERY kind of bacon",
                BasePrice = 11.99m,
                ImageUrl = "img/pizzas/bacon.jpg",
            },
            new PizzaSpecial()
            {
                Id = 3,
                Name = "Classic pepperoni",
                Description = "It's the pizza you grew up with, but Blazing hot!",
                BasePrice = 10.50m,
                ImageUrl = "img/pizzas/pepperoni.jpg",
            },
            new PizzaSpecial()
            {
                Id = 4,
                Name = "Buffalo chicken",
                Description = "Spicy chicken, hot sauce and bleu cheese, guaranteed to warm you up",
                BasePrice = 12.75m,
                ImageUrl = "img/pizzas/meaty.jpg",
            },
            new PizzaSpecial()
            {
                Id = 5,
                Name = "Mushroom Lovers",
                Description = "It has mushrooms. Isn't that obvious?",
                BasePrice = 11.00m,
                ImageUrl = "img/pizzas/mushroom.jpg",
            },
            new PizzaSpecial()
            {
                Id = 7,
                Name = "Veggie Delight",
                Description = "It's like salad, but on a pizza",
                BasePrice = 11.50m,
                ImageUrl = "img/pizzas/salad.jpg",
            },
            new PizzaSpecial()
            {
                Id = 8,
                Name = "Margherita",
                Description = "Traditional Italian pizza with tomatoes and basil",
                BasePrice = 9.99m,
                ImageUrl = "img/pizzas/margherita.jpg",
            },
            new PizzaSpecial()
            {
                Id = 9,
                Name = "Margherita Family Size",
                Description = "Only 24\" of pure tomatoes and basil",
                BasePrice = 12.99m,
                ImageUrl = "img/pizzas/margherita.jpg",
            },
        };
        db.Specials.AddRange(specials);
        db.SaveChanges();
    }
}
```

Controller

OrdersController

```
/// <summary>
/// 允许应用获取所有当前订单并下订单。
/// [Route("orders")] Blazor 属性允许此类处理对 /orders 和 /orders/{orderId} 的传入 HTTP 请求。
/// </summary>
[Route("orders")]
[ApiController]
public class OrdersController : Controller
```

```
{
    private readonly PizzaStoreContext _db;

    public OrdersController(PizzaStoreContext db)
    {
        _db = db;
    }

    [HttpGet]
    public async Task<ActionResult<List<OrderWithStatus>>> GetOrders()
    {
        var orders = await _db.Orders
            .Include(o => o.Pizzas).ThenInclude(p => p.Special)
            .Include(o => o.Pizzas).ThenInclude(p => p.Toppings).ThenInclude(t => t.Topping)
            .OrderByDescending(o => o.CreatedTime)
            .ToListAsync();

        return orders.Select(o => OrderWithStatus.FromOrder(o)).ToList();
    }

    [HttpPost]
    public async Task<ActionResult<int>> PlaceOrder(Order order)
    {
        order.CreatedTime = DateTime.Now;

        // Enforce existence of Pizza.SpecialId and Topping.ToppingId
        // in the database - prevent the submitter from making up
        // new specials and toppings
        foreach (var pizza in order.Pizzas)
        {
            pizza.SpecialId = pizza.Special.Id;
            pizza.Special = null;
        }

        _db.Orders.Attach(order);
        await _db.SaveChangesAsync();

        return order.OrderId;
    }

    //获取订单的状态
    [HttpGet("{orderId}")]
    public async Task<ActionResult<OrderWithStatus>> GetOrderWithStatus(int orderId)
    {
        var order = await _db.Orders
            .Where(o => o.OrderId == orderId)
            .Include(o => o.Pizzas).ThenInclude(p => p.Special)
            .Include(o => o.Pizzas).ThenInclude(p => p.Toppings).ThenInclude(t => t.Topping)
            .SingleOrDefaultAsync();

        if (order == null)
        {
            return NotFound();
        }

        return OrderWithStatus.FromOrder(order);
    }
}
```

SpecialsController

```
/// <summary>
/// 此类创建一个控制器，让我们能够在数据库中查询披萨特价商品，
/// 并在 (http://localhost:5000/specials) URL 以 JSON 的形式返回这些内容。
/// </summary>
```

```
[Route("specials")]
[ApiController]
public class SpecialsController : Controller
{
    private readonly PizzaStoreContext _db;

    public SpecialsController(PizzaStoreContext db)
    {
        _db = db;
    }

    [HttpGet]
    public async Task<ActionResult<List<PizzaSpecial>>> GetSpecials()
    {
        return (await _db.Specials.ToListAsync()).OrderByDescending(s => s.BasePrice).ToList();
    }
}
```

Services

OrderState

```
/// <summary>
/// 新的订单状态容器服务
/// </summary>
public class OrderState
{
    public bool ShowingConfigureDialog { get; private set; }
    public Pizza ConfiguringPizza { get; private set; }
    public Order Order { get; private set; } = new Order();

    public void ShowConfigurePizzaDialog(PizzaSpecial special)
    {
        ConfiguringPizza = new Pizza()
        {
            Special = special,
            SpecialId = special.Id,
            //Size = Pizza.DefaultSize,
            Toppings = new List<PizzaTopping>(),
        };
        ConfiguringPizza.Size=ConfiguringPizza.DefaultSize;
        ShowingConfigureDialog = true;
    }

    public void CancelConfigurePizzaDialog()
    {
        ConfiguringPizza = null;

        ShowingConfigureDialog = false;
    }

    public void ConfirmConfigurePizzaDialog()
    {
        Order.Pizzas.Add(ConfiguringPizza);
        ConfiguringPizza = null;

        ShowingConfigureDialog = false;
    }

    public void RemoveConfiguredPizza(Pizza pizza)
    {
        Order.Pizzas.Remove(pizza);
    }
    public void ResetOrder()
    {
        Order = new Order();
    }
}
```