100 XP

Exercise - Bind controls to data in Blazor applications

5 minutes

The Blazing Pizza app needs to update the interface when customers amend pizzas and add them to their orders. Blazor allows you to bind HTML controls to C# properties to update when their values change.

Customers should see what pizzas they're ordering and how the size they choose affects the price they'll pay.

In this exercise, you'll get the Blazing Pizza app in a position where orders can be updated and edited. You'll see how to bind controls to the properties of a pizza and recalculate prices on these changes.

Display a customer's pizza order

You're going to add a sidebar that will display all the pizzas that a customer has added to their order.

1. Stop the app if it's still running.
2. In Visual Studio Code, in the file explorer, expand **Pages** and then select **Index.razor**.
3. Between the `@if` and `@code` blocks, add this code:

razor

```razor
<div class="sidebar">
    @if (order.Pizzas.Any())
    {
        <div class="order-contents">
            <h2>Your order</h2>

            @foreach (var configuredPizza in order.Pizzas)
            {
              <div class="cart-item">
                  <div class="title">@(configuredPizza.Size)" @configuredPizza.Special.Name</div>
                  <div class="item-price">
                      @configuredPizza.GetFormattedTotalPrice()
                  </div>
              </div>
            }
        </div>
    }
    else
    {
        <div class="empty-cart">Choose a pizza<br>to get started</div>
    }

    <div class="order-total @(order.Pizzas.Any() ? "" : "hidden")">
        Total:
        <span class="total-price">@order.GetFormattedTotalPrice()</span>
        <a href="checkout" class="@(OrderState.Order.Pizzas.Count == 0 ? "btn btn-warning disabled" : "btn
btn-warning")">
            Order >
        </a>
    </div>
</div>
```

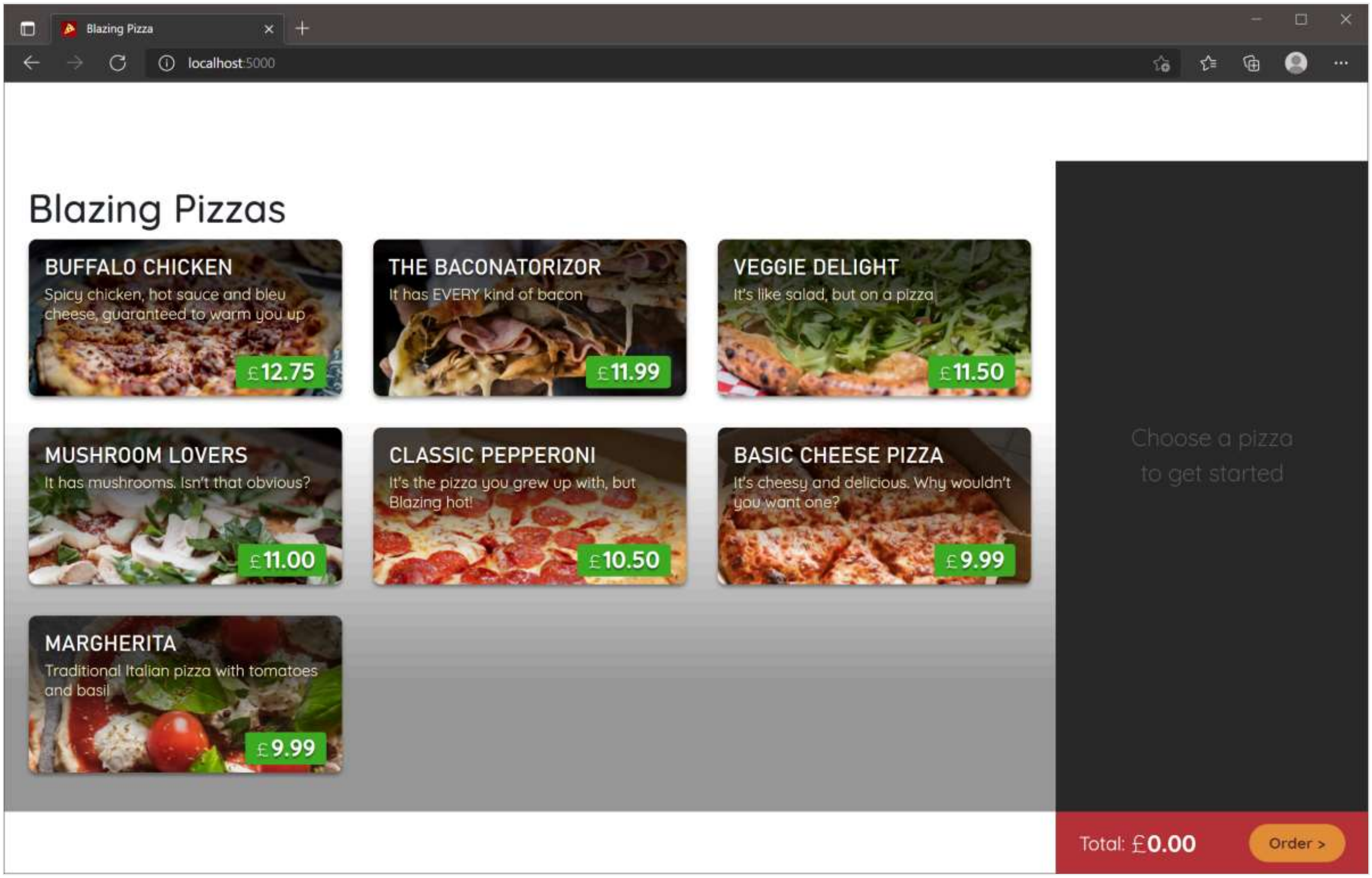
This HTML adds a sidebar to the page. If there are any pizzas in `OrderState.Order`, it displays them. If there are no orders, customers are prompted to add some.

You'll see some errors because the component doesn't know what the orders are.

4. In the `@code` block under `List<PizzaSpecial> specials = new();`, add this code:

```C#
Order order => OrderState.Order;
```

5. Select `F5` or select **Run**. Then select **Start Debugging**.



Blazing Pizza
localhost:5000
Blazing Pizzas
BUFFALO CHICKEN
Spicy chicken, hot sauce and bleu cheese, guaranteed to warm you up
£12.75
THE BACONATORIZOR
It has EVERY kind of bacon
£11.99
VEGGIE DELIGHT
It's like salad, but on a pizza
£11.50
MUSHROOM LOVERS
It has mushrooms. Isn't that obvious?
£11.00
CLASSIC PEPPERONI
It's the pizza you grew up with, but Blazing hot!
£10.50
BASIC CHEESE PIZZA
It's cheesy and delicious. Why wouldn't you want one?
£9.99
MARGHERITA
Traditional Italian pizza with tomatoes and basil
£9.99
Choose a pizza to get started
Total: £0.00
Order >

Try ordering some pizzas and canceling some. You'll see they get added to the basket and the order total updates.

6. Select `Shift`+`F5` or select **Stop Debugging**.

Remove a pizza from a customer's order

You might have noticed that there's no way to remove a configured pizza from the customer's shopping basket. Let's add that functionality.

The first step is to update the `OrderState` service so that it can provide a method to remove pizzas from an order.

1. In the file explorer, select **Services/OrderState.cs**.

2. At the bottom of the class, add this method:

```C#
public void RemoveConfiguredPizza(Pizza pizza)
{
    Order.Pizzas.Remove(pizza);
}
```

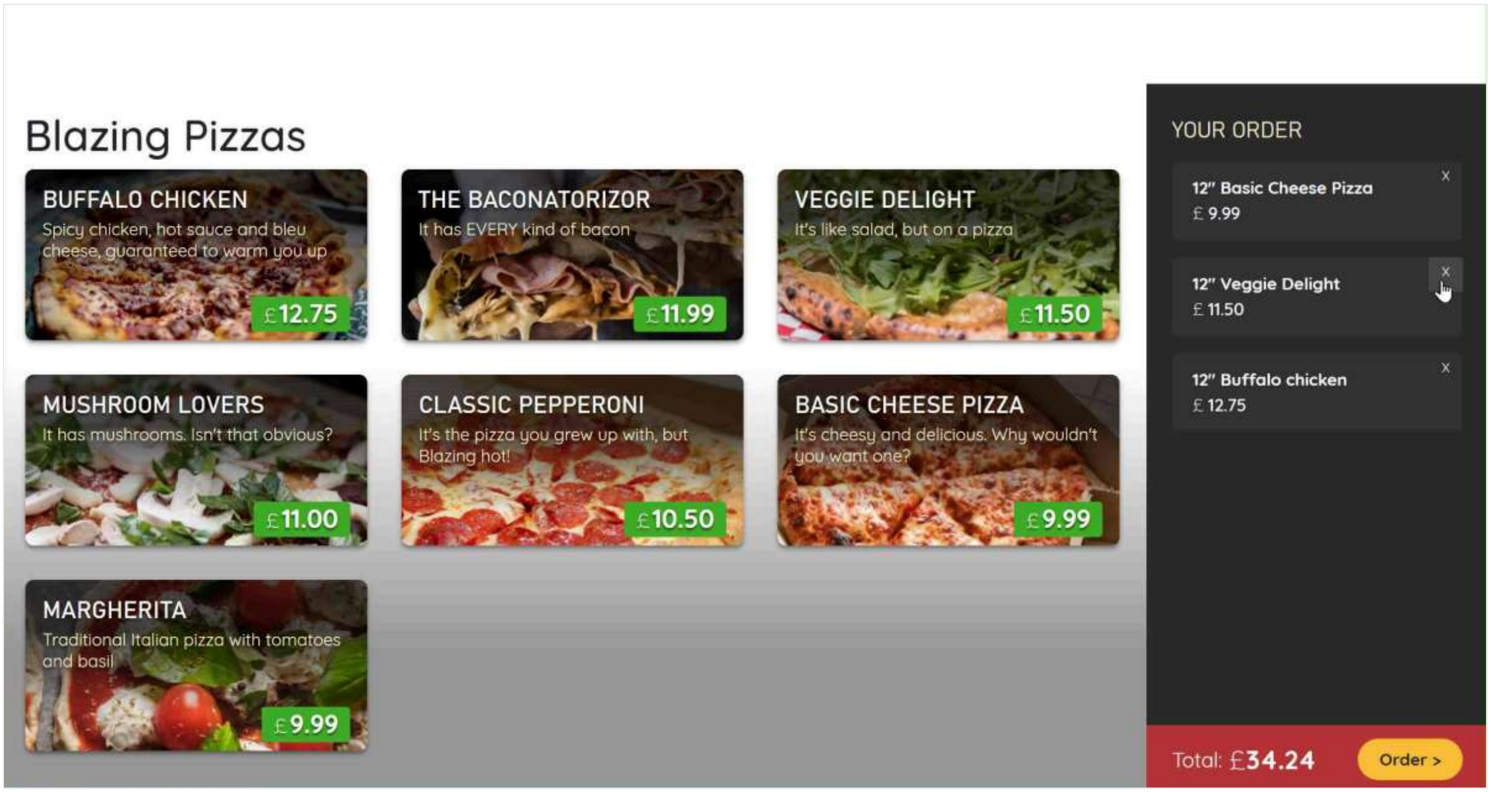
3. In the file explorer, expand **Pages** and then select **Index.razor**.

4. In `<div class="cart-item">`, add an `<a>` tag before the closing `</div>` to create a remove button:

```razor
<a @onclick="@(() => OrderState.RemoveConfiguredPizza(configuredPizza))" class="delete-item">x</a>
```

This tag adds an `X` to each pizza in the order sidebar. When it's selected, it calls the `RemoveConfiguredPizza` method in the `OrderState` service.

5. Select `F5` or select **Run**. Then select **Start Debugging**.


Blazing Pizzas
BUFFALO CHICKEN
Spicy chicken, hot sauce and bleu cheese, guaranteed to warm you up
£12.75
THE BACONATORIZOR
It has EVERY kind of bacon
£11.99
VEGGIE DELIGHT
It's like salad, but on a pizza
£11.50
MUSHROOM LOVERS
It has mushrooms. Isn't that obvious?
£11.00
CLASSIC PEPPERONI
It's the pizza you grew up with, but Blazing hot!
£10.50
BASIC CHEESE PIZZA
It's cheesy and delicious. Why wouldn't you want one?
£9.99
MARGHERITA
Traditional Italian pizza with tomatoes and basil
£9.99
YOUR ORDER
12" Basic Cheese Pizza
£ 9.99
x
12" Veggie Delight
£ 11.50
x
12" Buffalo chicken
£ 12.75
x
Total: £34.24
Order >

6. Select `Shift`+`F5` or select **Stop Debugging**.

Configure a pizza size dynamically

The pizza configuration dialog doesn't update when the size slider is changed. The component needs a way to update the pizza and the size, and then recalculate the price.

1. In the file explorer, expand **Shared** and then select **ConfigurePizzaDialog.razor**.

2. Add code to the `input` HTML control to bind its value to the pizza size:

razor

```
<input type="range" min="@Pizza.MinimumSize" max="@Pizza.MaximumSize" step="1" @bind="Pizza.Size"/>
```

3. Select F5 or select **Run**. Then select **Start Debugging**.

   Use the updated dialog to add some different-sized pizzas to your order. Click the slider bar instead of dragging. Observe that the pizza size updates on the mouse-up event on the control.

   If you drag the slider, the size won't change until you release the mouse. Let's fix that.

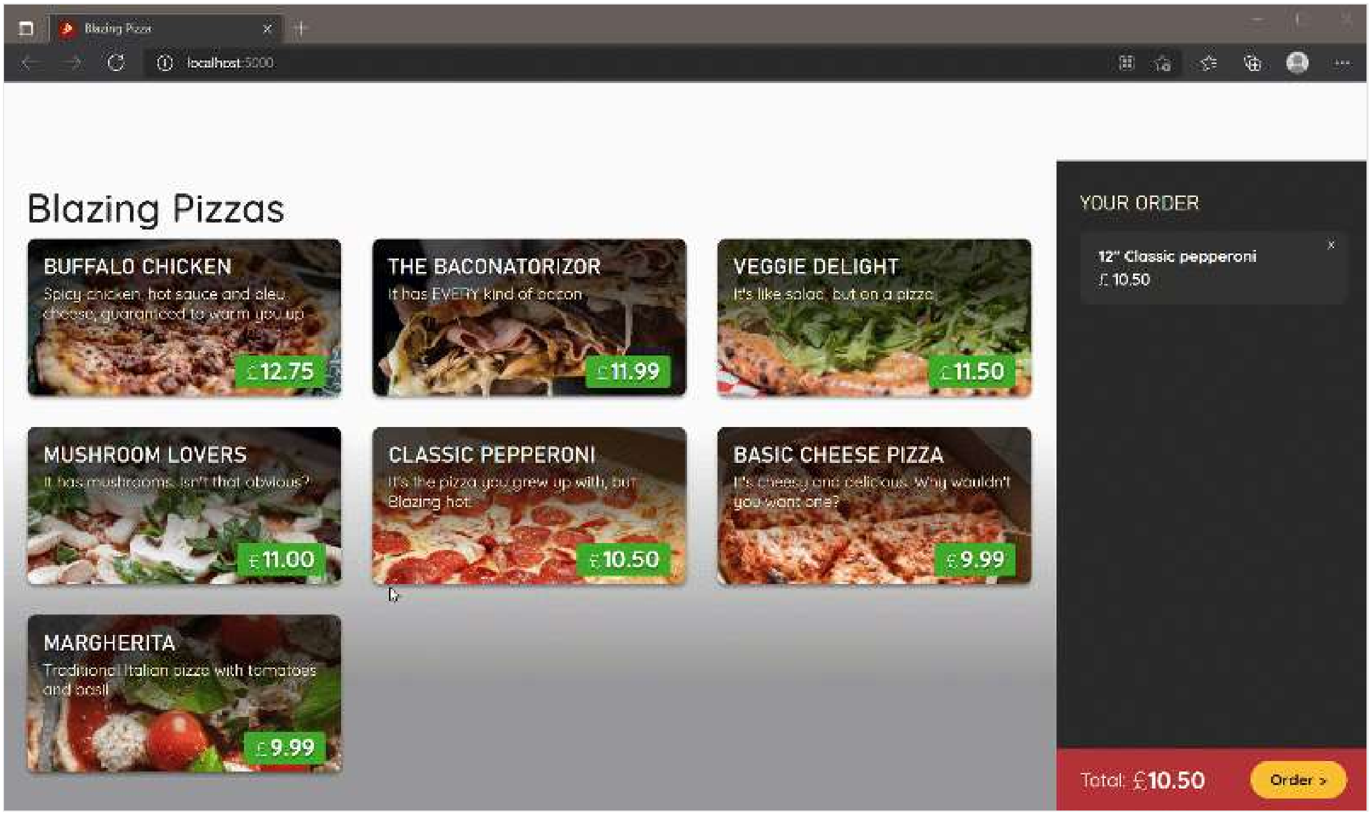
4. Select Shift + F5 or select **Stop Debugging**.

5. Add the event the control should be bound to:

   razor

```
<input type="range" min="@Pizza.MinimumSize" max="@Pizza.MaximumSize" step="1" @bind="Pizza.Size"
@bind:event="oninput" />
```

6. Select F5 or select **Run**. Then select **Start Debugging**.


Blazing Pizza
localhost:5000
Blazing Pizzas
BUFFALO CHICKEN
Spicy chicken, hot sauce and bleu cheese, guaranteed to warm you up
£12.75
THE BACONATORIZOR
It has EVERY kind of bacon
£11.99
VEGGIE DELIGHT
It's like salad, but on a pizza
£11.50
MUSHROOM LOVERS
It has mushrooms. Isn't that obvious?
£11.00
CLASSIC PEPPERONI
It's the pizza you grew up with, but Blazing hot
£10.50
BASIC CHEESE PIZZA
It's cheesy and delicious. Why wouldn't you want one?
£9.99
MARGHERITA
Traditional Italian pizza with tomatoes and basil
£9.99
YOUR ORDER
12" Classic pepperoni
£10.50
Total: £10.50
Order >

How did adding the `@bind="Pizza.Size"` code provide so much functionality? If you examine the whole of the **ConfigurePizzaDialog.razor** code, you'll see that your team had already connected the other elements to the pizza's properties.

For example, the price updates because of this code:

razor

```
Price: <span class="price">@(Pizza.GetFormattedTotalPrice())</span>
```

The price updates as the pizza size changes because the method on the pizza `GetFormattedTotalPrice()` uses the pizza size to calculate the total price.

You've made progress on the Blazing Pizza app. If you want to continue to improve it, complete the next module in this learning path.

Next unit: Summary

Continue >