100 XP

Exercise - Access data from a Blazor component

5 minutes

The current hard-coded pizzas in the app need to be replaced with a database. You can use the Microsoft Entity Framework to add connections to data sources. In this app, we'll use a SQLite database to store the pizzas.

In this exercise, you'll add packages to support database functionality, connect classes to a back-end database, and add a helper class to preload data for the company's pizzas.

Add packages to support database access

1. Stop the app if it's still running.
2. In Visual Studio Code, select **Terminal** > **New Terminal**.
3. In the new terminal, set your location to the *BlazingPizza* directory.

```
PowerShell

cd BlazingPizza
```

4. Run these commands to add the **Microsoft.EntityFrameworkCore**, **Microsoft.EntityFrameworkCore.Sqlite** and **System.Net.Http.Json** packages:

```
PowerShell

dotnet add package Microsoft.EntityFrameworkCore --version 6.0.8
dotnet add package Microsoft.EntityFrameworkCore.Sqlite --version 6.0.8
dotnet add package System.Net.Http.Json --version 6.0.0
```

These commands add package references to your BlazingPizza.csproj file:

```
XML

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="6.0.8" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="6.0.8" />
    <PackageReference Include="System.Net.Http.Json" Version="6.0.0" />
  </ItemGroup>
```

Add a database context

1. In Visual Studio Code, create a new folder in the *BlazingPizza* folder. Name it *Data*.
2. Create a new file in the *Data* folder. Name it *PizzaStoreContext.cs*.
3. Enter this code for the class:

C#

```
using Microsoft.EntityFrameworkCore;

namespace BlazingPizza.Data;

public class PizzaStoreContext : DbContext
{
    public PizzaStoreContext(DbContextOptions options) : base(options)
    {
    }

    public DbSet<PizzaSpecial> Specials { get; set; }
}
```

This class creates a database context we can use to register a database service. The context also allows us to have a controller that accesses the database.

4. Save your changes.

Add a controller

1. Create a new folder in the *BlazingPizza* folder. Name it *Controllers*.

2. Create a new file in the *Controllers* folder. Name it *SpecialsController.cs*.

3. Enter this code for the class:

C#

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using BlazingPizza.Data;

namespace BlazingPizza.Controllers;

[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();
    }
}
```

This class creates a controller that allows us to query the database for pizza specials and returns them as JSON at the `(http://localhost:5000/specials)` URL.

4. Save your changes.

Load data into the database

The app checks to see if there's an existing SQLite database and creates one with some premade pizzas.

1. Create a new file in the *Data* directory. Name it *SeedData.cs*.

2. Enter this code for the class:

C#

```
namespace BlazingPizza.Data;

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",
            },
        };
        db.Specials.AddRange(specials);
        db.SaveChanges();
    }
}
```

The class uses a passed database context, creates some `PizzaSpecial` objects in an array, and then saves them.

3. In the file explorer, select **Program.cs**.

4. At the top, add a reference to a new `PizzaStoreContext`:

```
C#

using BlazingPizza.Data;
```

This statement allows the app to use the new service.

5. Insert this segment above the `app.Run();` method:

```
C#

...
// Initialize the database
var scopeFactory = app.Services.GetRequiredService<IServiceScopeFactory>();
using (var scope = scopeFactory.CreateScope())
{
    var db = scope.ServiceProvider.GetRequiredService<PizzaStoreContext>();
    if (db.Database.EnsureCreated())
    {
        SeedData.Initialize(db);
    }
}

app.Run();
```

This change creates a database scope with the `PizzaStoreContext`. If there isn't a database already created, it calls the `SeedData` static class to create one.

6. At the moment, the app doesn't work because we haven't initialized the `PizzaStoreContext`. In the `Add Services to the container` section higher in the *Program.cs* file, add this code under the current services (the lines that start `builder.Services.`):

```
C#

  builder.Services.AddHttpClient();
  builder.Services.AddSqlite<PizzaStoreContext>("Data Source=pizza.db");
```

This code registers two services. The first `AddHttpClient` statement allows the app to access HTTP commands. The app uses an HttpClient to get the JSON for pizza specials. The second statement registers the new `PizzaStoreContext` and provides the filename for the SQLite database.

Use the database to display pizzas

We can now replace the hard-coded pizza in the **Index.razor** page.

1. In the file explorer, select **Index.razor**.
2. Replace the existing `OnInitialized()` method with:

```
C#

protected override async Task OnInitializedAsync()
{
    specials = await HttpClient.GetFromJsonAsync<List<PizzaSpecial>>(NavigationManager.BaseUri + "specials");
}
```

> **Note**
>
> This code has replaced `OnInitialized()` with `OnInitializedAsync()`. Specials are now going to be returned as JSON from the app asynchronously.

3. There are some errors that you need to fix. Add these `@inject` statements under the `@page` directive:

```
razor

@inject HttpClient HttpClient
@inject NavigationManager NavigationManager
```

4. Save all your changes, and then select `F5` or select **Run**. Then select **Start Debugging**.

   There's a runtime error when you run the app. The **JsonReader** raised an exception.

5. Remember that the app should be creating JSON at `(http://localhost:5000/specials)`. Go to that URL.

   The app doesn't know how to route this request. You'll learn about routing in the module on Blazor routing. Let's fix the error now.

6. Select `Shift` + `F5`, or select **Stop Debugging**.
7. In the file explorer, select **Program.cs**.
8. Around the middle of the file, after the lines that start `app.`, add this endpoint:

```
C#
```

```
app.MapControllerRoute("default", "{controller=Home}/{action=Index}/{id?}");
```

The code should now be:

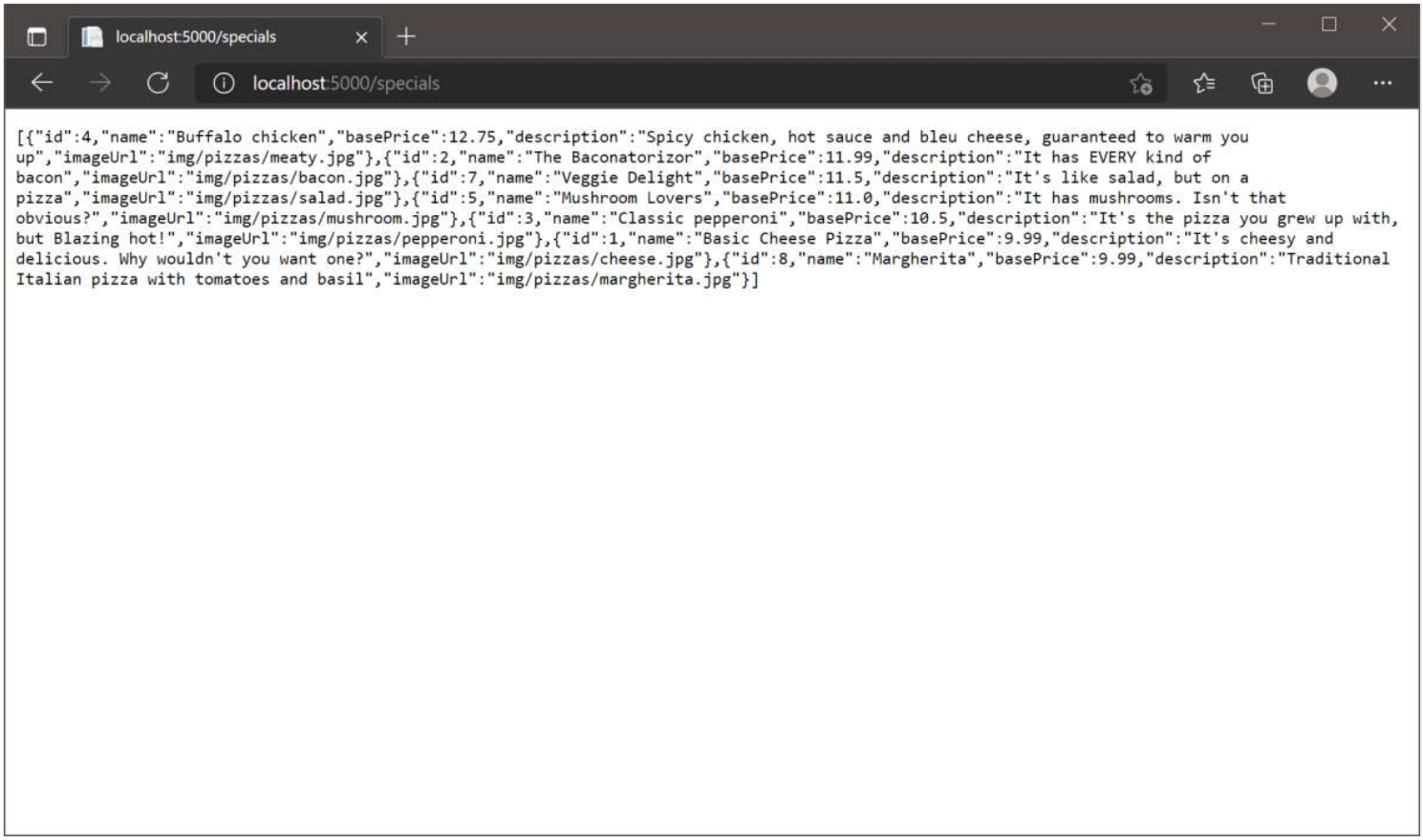
```
C#

...
app.MapRazorPages();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");
app.MapControllerRoute("default", "{controller=Home}/{action=Index}/{id?}");
...
```

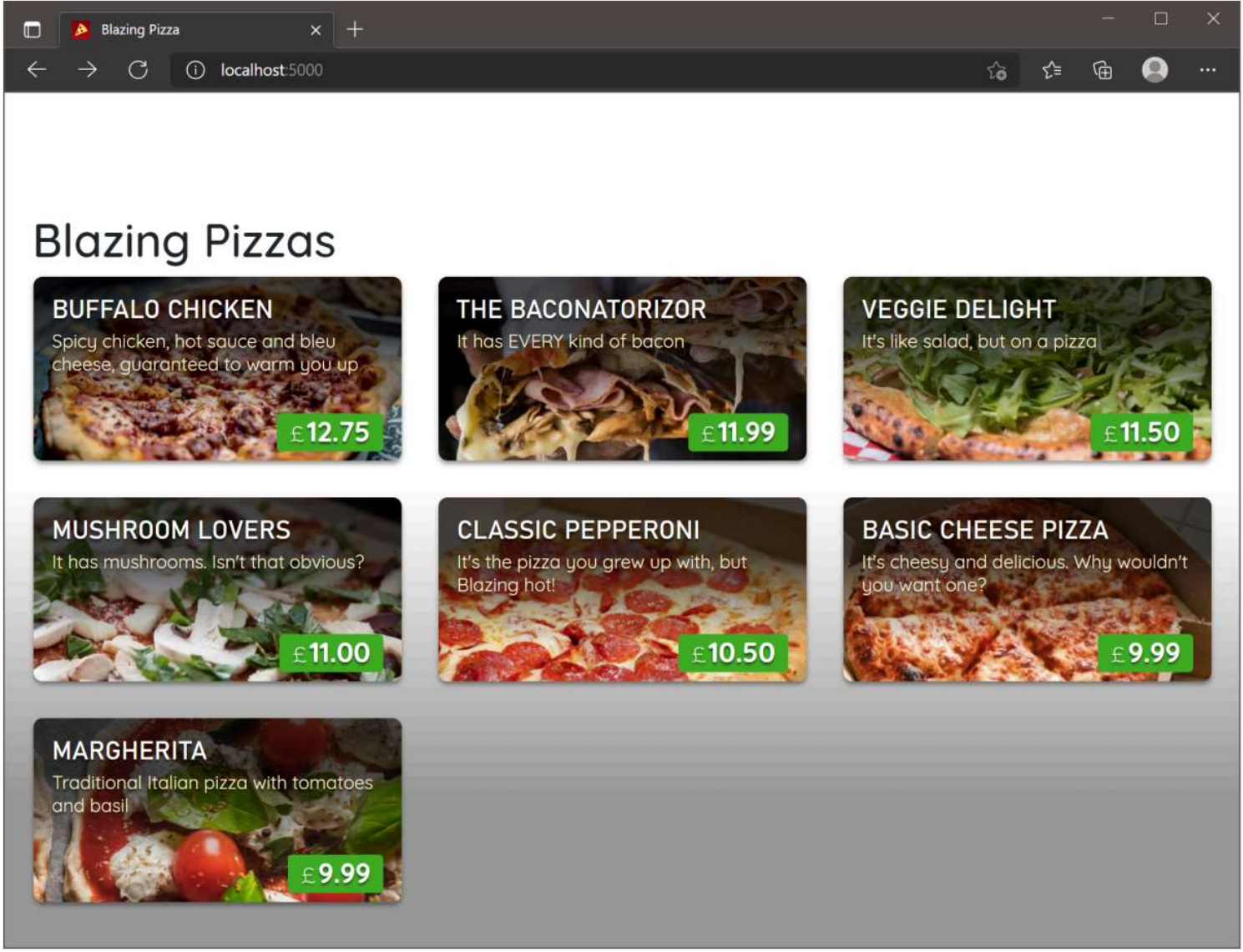
9. Select F5 or select **Run**. Then select **Start Debugging**.

   The app should now work, but let's check that the JSON is being created correctly.

10. Go to `(http://localhost:5000/specials)` to see:



The JSON has the pizzas listed in descending order of price as specified in the special pizza controller.



Next unit: Share data in Blazor applications

Continue >