go deploy

Screenshots

# Module 7: Using Entity Framework Core in ASP.NET Core

## Lab: Using Entity Framework Core in ASP.NET Core

**Scenario**

Your company is planning to write a web application for managing cupcakes and bakeries. To connect the application to a database, your development team has decided to use Entity Framework Core. You have been asked to create a class that derives from a DbContext class, and then use the class to retrieve data from the database and store data in the database. The application will enable users to store uploaded cupcakes, edit their properties, view their details, and delete them.

## Exercise 1: Adding Entity Framework Core

**Scenario**

In this exercise, you will first add the Cupcake model and the Bakery model to the Cupcake web application. You will then add an Entity Framework context class named CupcakeContext to the web application. After that you will configure the CupcakeContext class to connect to a SQLite database. Finally, you will use data seeding to populate the database with sample data when the database is created.

The main tasks for this exercise are as follows:

- Create model classes
- Create a class that derives from DbContext
- Set up Entity Framework Core to use SQLite
- Use OnModelCreating to populate the database

## Task 1: Create model classes

1. Go to **D:\Allfiles\Mod07\Labfiles\01_Cupcakes_begin**, and then double-click **Cupcakes.sln**.

> **Note**: If a **Security Warning for Cupcakes** dialog box appears, verify that the **Ask me for every project in this solution** check box is cleared, and then click OK.

2. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point

go deploy

3. In the **Add New Item - Cupcakes** dialog box, in the **Name:** textbox, type Cupcake, and then click **Add**.

4. In the **Cupcake.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
[Key]
public int CupcakeId { get; set; }

[Required(ErrorMessage = "Please select a cupcake type")]
[Display(Name = "Cupcake Type:")]
public CupcakeType? CupcakeType { get; set; }

[Required(ErrorMessage = "Please enter a cupcake description")]
[Display(Name = "Description:")]
public string Description { get; set; }

[Display(Name = "Gluten Free:")]
public bool GlutenFree { get; set; }

[Range(1, 15)]
[Required(ErrorMessage = "Please enter a cupcake price")]
[DataType(DataType.Currency)]
[Display(Name = "Price:")]
public double? Price { get; set; }

[NotMapped]
[Display(Name = "Cupcake Picture:")]
public IFormFile PhotoAvatar { get; set; }

public string ImageName { get; set; }

public byte[] PhotoFile { get; set; }

public string ImageMimeType { get; set; }

[Required(ErrorMessage = "Please select a bakery")]
public int? BakeryId { get; set; }

public virtual Bakery Bakery { get; set; }
```

5. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **Class...**.

6. In the **Add New Item - Cupcakes** dialog box, in the **Name** box, type Bakery, and then click **Add**.

7. In the **Bakery.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
[StringLength(50, MinimumLength = 4)]
public string BakeryName { get; set; }

[Range(1, 40)]
public int Quantity { get; set; }

[StringLength(50, MinimumLength = 4)]
public string Address { get; set; }

public virtual ICollection<Cupcake> Cupcakes { get; set; }
```

## Task 2: Create a class that derives from DbContext

1. In **Solution Explorer**, right-click **Cupcakes**, point to **Add**, and then click **New Folder**.
2. In the **NewFolder** box, type **Data**, and then press Enter.
3. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Data**, point to **Add**, and then click **Class...**.
4. In the **Add New Item - Cupcakes** dialog box, in the **NameL** box, type **CupcakeContext**, and then click **Add**.
5. In the **CupcakeContext.cs** code window, locate the following code:

```
public class CupcakeContext
```

6. Add a base class **DbContext** so that the code looks like the following:

```
public class CupcakeContext : DbContext
```

7. In the **CupcakeContext.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
public CupcakeContext(DbContextOptions<CupcakeContext> options) : base(options)
{
}

public DbSet<Cupcake> Cupcakes { get; set; }
public DbSet<Bakery> Bakeries { get; set; }
```

## Task 3: Set up Entity Framework Core to use SQLite

1. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Cupcakes**, and then select **Manage NuGet Packages**.

go deploy

3. In the **Search** box, type **Microsoft.EntityFrameworkCore.Sqlite**, and then press Enter.
4. In the search results list, choose **Microsoft.EntityFrameworkCore.Sqlite**, and in the right-hand panel select the latest stable version, and then click **Install**.
5. If a **Preview Changes** dialog box appears, click **OK**.
6. If a **License Acceptance** dialog box appears, click **I Accept**.
7. Close the **NuGet: Cupcakes** window.
8. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, click **Program.cs**.
9. In the **Program.cs** code window, find the line:

```
var app = builder.Build();
```

10. **Before** this line, add the following code:

```
builder.Services.AddDbContext<CupcakeContext>(options =>
        options.UseSqlite("Data Source=cupcake.db"));
```

11. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, in the **Data** folder, click on **CupcakeContext.cs**.
12. In **CupcakeContext.cs**, in the constructor body, add the statement **Database.EnsureCreated();**, so that the constructor looks like the following:

```
public CupcakeContext(DbContextOptions<CupcakeContext> options) : base(options)
{
    Database.EnsureCreated();
}
```

> This will create a database file for you automatically when the application first runs.

## Task 4: Use OnModelCreating to populate the database

1. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, under **Data**, click **CupcakeContext.cs**.
2. In the **CupcakeContext.cs** code window, locate the following code:

```
public DbSet<Cupcake> Cupcakes { get; set; }
public DbSet<Bakery> Bakeries { get; set; }
```

go deploy

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
}
```

4. In the **OnModelCreating** method code block, type the following code:

```
modelBuilder.Entity<Bakery>().HasData(
new Bakery
{
    BakeryId = 1,
    BakeryName = "Gluteus Free",
    Address = "635 Brighton Circle Road",
    Quantity = 8
},
new Bakery
{
    BakeryId = 2,
    BakeryName = "Cupcakes Break",
    Address = "4323 Jerome Avenue",
    Quantity = 22
},
new Bakery
{
    BakeryId = 3,
    BakeryName = "Cupcakes Ahead",
    Address = "2553 Pin Oak Drive",
    Quantity = 18
},
new Bakery
{
    BakeryId = 4,
    BakeryName = "Sugar",
    Address = "1608 Charles Street",
    Quantity = 30
}
);
```

5. In the **OnModelCreating** method code block, immediately after the code you just added, press Enter, and then type the following code:

```
            {
                CupcakeId = 1,
                CupcakeType = CupcakeType.Birthday,
                Description = "Vanilla cupcake with coconut cream",
                GlutenFree = true,
                Price = 2.5,
                BakeryId = 1,
                ImageMimeType = "image/jpeg",
                ImageName = "birthday-cupcake.jpg"
            },
            new Cupcake
            {
                CupcakeId = 2,
                CupcakeType = CupcakeType.Chocolate,
                Description = "Chocolate cupcake with caramel filling and chocolate butter
                GlutenFree = false,
                Price = 3.2,
                BakeryId = 2,
                ImageMimeType = "image/jpeg",
                ImageName = "chocolate-cupcake.jpg"
            },
            new Cupcake
            {
                CupcakeId = 3,
                CupcakeType = CupcakeType.Strawberry,
                Description = "Chocolate cupcake with straberry cream filling",
                GlutenFree = false,
                Price = 4,
                BakeryId = 3,
                ImageMimeType = "image/jpeg",
                ImageName = "pink-cupcake.jpg"
            },
            new Cupcake
            {
                CupcakeId = 4,
                CupcakeType = CupcakeType.Turquoise,
                Description = "Vanilla cupcake with butter cream",
                GlutenFree = true,
                Price = 1.5,
                BakeryId = 4,
                ImageMimeType = "image/jpeg",
                ImageName = "turquoise-cupcake.jpg"
            }
        );
```

- [ ] 6. In the Visual Studio, from the menu, choose **File** and then **Save All**, to save your changes.

✓ **Results**: After completing this exercise, you have added Entity Framework Core to the Cupcake

go deploy

## Exercise 2: Use Entity Framework Core to Retrieve and Store Data

**Scenario**

In this exercise, you will first create a repository for the web application. The repository will access an SQLite database by using Entity Framework Core. You will then use a dependency injection to inject the service to a controller. You will use the repository in the controller to access the database. In the controller, you will retrieve the cupcakes and bakeries data, and then you will manipulate the data.

The main tasks for this exercise are as follows:

- Create a repository
- Update a controller to use a repository
- Use Entity Framework Core to retrieve data
- Manipulate data by using Entity Framework Core
- Run the application

### Task 1: Create a repository

1. In **Solution Explorer**, right-click **Cupcakes**, point to **Add**, and then click **New Folder**.
2. In the **NewFolder** box, type **Repositories**, and then press Enter.
3. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Repositories**, point to **Add**, and then click **New Item...**.
4. In the **Add New Item - Cupcakes** dialog, select **Interface**.
5. In the **Name:** textbox, type **ICupcakeRepository**, and then click **Add**.
6. In the **ICupcakeRepository.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
IEnumerable<Cupcake> GetCupcakes();
Cupcake GetCupcakeById(int id);
void CreateCupcake(Cupcake cupcake);
void DeleteCupcake(int id);
void SaveChanges();
IQueryable<Bakery> PopulateBakeriesDropDownList();
```

7. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Repositories**, point to **Add**, and then click **Class...**.

go deploy

9. In the **CupcakeRepository.cs** code window, locate the following code:

```
public class CupcakeRepository
```

10. Add the **ICupcakeRepository** interface to class, so that it looks like the following code:

```
public class CupcakeRepository : ICupcakeRepository
```

11. In the **CupcakeRepository.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
private CupcakeContext _context;

public CupcakeRepository(CupcakeContext context)
{
    _context = context;
}
```

12. Ensure that the cursor is at the end of the **constructor** code block, press Enter two times, and then type the following code:

```
public IEnumerable<Cupcake> GetCupcakes()
{
    return _context.Cupcakes.ToList();
}
```

13. Ensure that the cursor is at the end of the **GetCupcakes** method code block, press Enter two times, and then type the following code:

```
public Cupcake GetCupcakeById(int id)
{
    return _context.Cupcakes.Include(b => b.Bakery)
        .SingleOrDefault(c => c.CupcakeId == id);
}
```

14. Ensure that the cursor is at the end of the **GetCupcakeById** method code block, press Enter two times, and then type the following code:

go deploy

```
        if (cupcake.PhotoAvatar != null && cupcake.PhotoAvatar.Length > 0)
        {
            cupcake.ImageMimeType = cupcake.PhotoAvatar.ContentType;
            cupcake.ImageName = Path.GetFileName(cupcake.PhotoAvatar.FileName);
            using (var memoryStream = new MemoryStream())
            {
                cupcake.PhotoAvatar.CopyTo(memoryStream);
                cupcake.PhotoFile = memoryStream.ToArray();
            }
            _context.Add(cupcake);
            _context.SaveChanges();
        }
    }
```

- [ ] 15. Ensure that the cursor is at the end of the **CreateCupcake** method code block, press Enter two times, and then type the following code:

```
public void DeleteCupcake(int id)
{
    var cupcake = _context.Cupcakes.SingleOrDefault(c => c.CupcakeId == id);
    _context.Cupcakes.Remove(cupcake);
    _context.SaveChanges();
}
```

- [ ] 16. Ensure that the cursor is at the end of the **DeleteCupcake** method code block, press Enter two times, and then type the following code:

```
public void SaveChanges()
{
    _context.SaveChanges();
}
```

- [ ] 17. Ensure that the cursor is at the end of the **SaveChanges** method code block, press Enter two times, and then type the following code:

```
public IQueryable<Bakery> PopulateBakeriesDropDownList()
{
    var bakeriesQuery = from b in _context.Bakeries
                        orderby b.BakeryName
                        select b;
    return bakeriesQuery;
}
```

- [ ] 18. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, click **Program.cs**.
- [ ] 19. In the **Program.cs** code window, find the line:

go deploy

- [ ] 20. **Before** this line, add the following code:

```
builder.Services.AddTransient<ICupcakeRepository, CupcakeRepository>();
```

- [ ] 21. In the **Cupcakes - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.

## Task 2: Update a controller to use a repository

- [ ] 1. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Controllers**, and then click **CupcakeController.cs**.
- [ ] 2. In the **CupcakeController.cs** code window, locate the following code:

```
using Microsoft.AspNetCore.Mvc;
```

- [ ] 3. Ensure that the cursor is at the end of the **Microsoft.AspNetCore.Mvc** namespace, press Enter, and then type the following code:

```
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
```

- [ ] 4. In the **CupcakeController.cs** code block, locate the following code:

```
public IActionResult Index()
{
    return View();
}
```

- [ ] 5. Place the cursor before the located code, press Enter, press the Up arrow key, type the following code, and then press Enter.

```
private ICupcakeRepository _repository;
private IWebHostEnvironment _environment;

public CupcakeController(ICupcakeRepository repository, IWebHostEnvironment environ
{
    _repository = repository;
    _environment = environment;
}
```

## Task 3: Use Entity Framework Core to retrieve data

- [ ] 1. In the **CupcakeController.cs** code block, in the **Index** action code block, select the following code:

```
return View();
```

go deploy

```
return View(_repository.GetCupcakes());
```

3. Ensure that the cursor is at the end of the **Index** action code block, press Enter two times, and then type the following code:

```
public IActionResult Details(int id)
{
    var cupcake = _repository.GetCupcakeById(id);
    if (cupcake == null)
    {
        return NotFound();
    }
    return View(cupcake);
}
```

4. Ensure that the cursor is at the end of the **Details** action code block, press Enter two times, and then type the following code:

```
private void PopulateBakeriesDropDownList(int? selectedBakery = null)
{
    var bakeries = _repository.PopulateBakeriesDropDownList();
    ViewBag.BakeryID = new SelectList(bakeries.AsNoTracking(), "BakeryId", "BakeryN
}
```

5. In the **Cupcakes - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.

## Task 4: Manipulate data by using Entity Framework Core

1. In the **CupcakeController.cs** code block, locate the following code:

```
public IActionResult Details(int id)
{
        var cupcake = _repository.GetCupcakeById(id);
        if (cupcake == null)
        {
            return NotFound();
        }
        return View(cupcake);
}
```

2. Place the cursor at the end of the located code, press Enter, type the following code, and then press Enter two times.

go deploy

```
{
    PopulateBakeriesDropDownList();
    return View();
}
```

- [ ] 3. In the **Create** action code block, type the following code:

- [ ] 4. Ensure that the cursor is at the end of the **Create** action code block, press Enter two times, and then type the following code:

```
[HttpPost, ActionName("Create")]
public IActionResult CreatePost(Cupcake cupcake)
{
    if (ModelState.IsValid)
    {
        _repository.CreateCupcake(cupcake);
        return RedirectToAction(nameof(Index));
    }
    PopulateBakeriesDropDownList(cupcake.BakeryId);
    return View(cupcake);
}
```

- [ ] 5. Ensure that the cursor is at the end of the **CreatePost** method code block, press Enter two times, and then type the following code:

```
[HttpGet]
public IActionResult Edit(int id)
{
    Cupcake cupcake = _repository.GetCupcakeById(id);
    if (cupcake == null)
    {
        return NotFound();
    }
    PopulateBakeriesDropDownList(cupcake.BakeryId);
    return View(cupcake);
}
```

- [ ] 6. Ensure that the cursor is at the end of the **Edit** action code block, press Enter two times, and then type the following code:

```
{
    var cupcakeToUpdate = _repository.GetCupcakeById(id);
    bool isUpdated = await TryUpdateModelAsync<Cupcake>(
                            cupcakeToUpdate,
                            "",
                            c => c.BakeryId,
                            c => c.CupcakeType,
                            c => c.Description,
                            c => c.GlutenFree,
                            c => c.Price);
    if (isUpdated == true)
    {
        _repository.SaveChanges();
        return RedirectToAction(nameof(Index));
    }
    PopulateBakeriesDropDownList(cupcakeToUpdate.BakeryId);
    return View(cupcakeToUpdate);
}
```

- [ ] 7. Ensure that the cursor is at the end of the **EditPost** action code block, press Enter two times, and then type the following code:

```
[HttpGet]
public IActionResult Delete(int id)
{
    var cupcake = _repository.GetCupcakeById(id);
    if (cupcake == null)
    {
        return NotFound();
    }
    return View(cupcake);
}
```

- [ ] 8. Ensure that the cursor is at the end of the **Delete** action code block, press Enter two times, and then type the following code:

```
[HttpPost, ActionName("Delete")]
public IActionResult DeleteConfirmed(int id)
{
    _repository.DeleteCupcake(id);
    return RedirectToAction(nameof(Index));
}
```

## Task 5: Run the application

- [ ] 1. In the **Cupcakes - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.
- [ ] 2. on the **Debug** menu, click **Start Without Debugging**.

go deploy

**Note**: The browser displays the **Create** action inside the **CupcakesController**.

- [ ] 4. On the **Add a Cupcake to the Shop** page, in the **Bakery** list, choose any value.
- [ ] 5. In the **Cupcake Type** list, choose any value.
- [ ] 6. In the **Description** box, type **A delicious cupcake**.
- [ ] 7. In the **Price** box, type **3.50**.
- [ ] 8. In the **Cupcake Picture** box, import the image from **D:\Allfiles\Mod07\Labfiles\Image\strawberry-cupcake.jpg**, and then click **Submit**.
- [ ] 9. On the **Welcome to our Cupcakes Shop** page, verify the newly submitted cupcake details are present.
- [ ] 10. Select a cupcake of your choice, and then click **Details**.
- [ ] 11. View the cupcake details, and then click **Back to List**.
- [ ] 12. On the **Welcome to our Cupcakes Shop** page, select a cupcake of your choice, and then click **Edit**.
- [ ] 13. On the **Edit a Cupcake** page, in the **Price** box, type **9.35**, and then click **Save**.
- [ ] 14. On the **Welcome to our Cupcakes Shop** page, select the cupcake you edited, and then click **Details**.
- [ ] 15. On the **Cupcake's Details** page, verify the newly edited cupcake details, and then click **Back to List**.
- [ ] 16. On the **Welcome to our Cupcakes Shop** page, select a cupcake of your choice, and then click **Delete**.
- [ ] 17. On the **Are you sure you want to delete this?** page, click the **Delete** button.
- [ ] 18. On the **Welcome to our Cupcakes Shop** page, verify that the cupcake is deleted.
- [ ] 19. In Microsoft Edge, click **Close**.
- [ ] 20. In the **Cupcakes - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**.
- [ ] 21. In Microsoft Edge, verify that the changes you made have been persisted.
- [ ] 22. In Microsoft Edge, click **Close**.

go deploy

solution folder. The next time you run the application it will be re-created for you.

✓ **Results**: After completing this exercise, you will be able to use Entity Framework Core to retrieve and store data through a repository in the **CupcakeController**.

## Exercise 3: Use Entity Framework Core to Connect to Microsoft SQL Server

**Scenario**

In this exercise, you will first configure the Cupcakes Shop web application to connect to a SQL Server database instead of connecting to an SQLite database. You will then store the connection string which is used to connect to the database in a configuration file. After that, you will use Migrations to create the database. Finally, you will add a property to an entity, and use Migrations to update the database schema.

The main tasks for this exercise are as follows:

- Connect to a SQL Database
- Specify a connection string in a configuration file
- Use Migrations to create a database
- Run the application
- Use Migrations to update the database schema
- Run the application

## Task 1: Connect to a SQL Server Database

1. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, expand the **Data** folder and click **CupcakeContext.cs**.

2. In the **CupcakeContext.cs** code window, delete the following code:

```
cupcakeContext.Database.EnsureCreated();
```

3. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, click **Program.cs**.

4. In the **Program.cs** code window, select the following code:

```
services.AddDbContext<CupcakeContext>(options =>
                options.UseSqlite("Data Source=cupcake.db"));
```

5. Replace the selected code with the following code:

go deploy

```
                    options.UseSqlServer(connectionString));
```

## Task 2: Specify a connection string in a configuration file

1. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Cupcakes**, point to **Add**, and then click **New Item...**.
2. In the **Add New Item - Cupcakes** dialog box, in the navigation pane, under **Installed**, click **ASP.NET Core**. In the result pane, click **App Settings File**, and then click **Add**.
3. In the **appsettings.json** code window, select the following code:

```
"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=_CHANGE_ME;Trusted_Co
```

4. Replace the selected code with the following code:

```
"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=BakeriesDb;Trusted_Co
```

5. In the **Program.cs** code window, select the following code:

```
string connectionString = "Server=(localdb)\\MSSQLLocalDB;Database=BakeriesDb;Trust
services.AddDbContext<CupcakeContext>(options =>
                    options.UseSqlServer(connectionString));
```

6. Replace the selected code with the following code:

```
builder.Services.AddDbContext<CupcakeContext>(options =>
                    options.UseSqlServer(builder.Configuration.GetConnectionString("Default
```

## Task 3: Use Migrations to create a database

1. In the **Cupcakes - Microsoft Visual Studio** window, on the **Tools** menu, point to **NuGet Package Manager**, and then click **Package Manager Console**.
2. In **Package Manager Console** tab, type the following command, and then press Enter.

```
Add-Migration InitialCreate
```

> ⚠ **Note**: In **Solution Explorer**, verify if a new folder named **Migrations** is created with multiple files. If the command gives an error, run the command **Install-Package Microsoft.EntityFrameworkCore.Tools** to ensure that the tools are installed.

3. In the **Package Manager Console** tab, type the following command, and then press Enter (it may complain that the database already exists if you have done this exercise before).

go deploy

- [ ] 4. In the **Cupcakes - Microsoft Visual Studio** window, on the **View** menu, click **SQL Server Object Explorer**.
- [ ] 5. In **SQL Server Object Explorer**, expand **(localdb)\MSSQLLocalDB**, expand **Databases**, and then expand **BakeriesDb**.

> **Note**: There should be several tables in **BakeriesDb**, including a dbo.Bakeries and a dbo.Cupcakes.

## Task 4: Run the application

- [ ] 1. In the **Cupcakes - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.
- [ ] 2. on the **Debug** menu, click **Start Without Debugging**.
- [ ] 3. In Microsoft Edge, click **Add Cupcake**.

> **Note**: The browser displays the **Create** action inside the **CupcakesController**.

- [ ] 4. On the **Add a Cupcake to the Shop** page, in the **Bakery** list, choose any value.
- [ ] 5. In the **Cupcake Type** list, choose any value.
- [ ] 6. In the **Description** box, type **A delicious cupcake**.
- [ ] 7. In the **Price** box, type **3.78**.
- [ ] 8. In the **Cupcake Picture** box, import an image from **D:\Allfiles\Mod07\Labfiles\Image\strawberry-cupcake.jpg**, and then click **Submit**.
- [ ] 9. On the **Welcome to our Cupcakes Shop** page, verify the newly submitted cupcake details.
- [ ] 10. Select a cupcake of your choice, and then click **Details**.
- [ ] 11. View the cupcake details, and then click **Back to List**.
- [ ] 12. On the **Welcome to our Cupcakes Shop** page, select a cupcake of your choice, and then click **Edit**.
- [ ] 13. On the **Edit a Cupcake** page, in the **Price** box, type **2.99** , and then click **Save**.
- [ ] 14. On the **Welcome to our Cupcakes Shop** page, select the cupcake you edited, and then click **Details**.
- [ ] 15. On the **Cupcake's Details** page, verify the newly edited cupcake details, and then click **Back to**

go deploy

16. On the **Welcome to our Cupcakes Shop** page, select a cupcake of your choice, and then click **Delete**.
17. On the **Are you sure you want to delete?** page, click **Delete**.
18. On the **Welcome to our Cupcakes Shop** page, verify that the cupcake is deleted.
19. In Microsoft Edge, click **Close**.
20. In the **Cupcakes - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**.
21. In Microsoft Edge, verify that the changes you made have been persisted in the database.
22. In Microsoft Edge, click **Close**.

## Task 5: Use Migrations to update the database schema

1. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, under **Models**, click **Cupcake.cs**.
2. In the **Cupcake.cs** code window, locate the following code:

```
[Display(Name = "Gluten Free:")]
public bool GlutenFree { get; set; }
```

3. Place the cursor at the end of the located code, press Enter two times, and then type the following code:

```
[Display(Name = "Calorific Value:")]
public int CalorificValue { get; set; }
```

4. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, under **Data**, click **CupcakeContext.cs**.
5. In the **CupcakeContext.cs** code window, in the **OnModelCreating** method code block, select the following code:

```
ImageName = "birthday-cupcake.jpg"
```

6. Replace the selected code with the following code:

```
ImageName = "birthday-cupcake.jpg",
CalorificValue = 355
```

7. In the **OnModelCreating** method code block, select the following code:

go deploy

8. Replace the selected code with the following code:

```
ImageName = "chocolate-cupcake.jpg",
CalorificValue = 195
```

9. In the **OnModelCreating** method code block, select the following code:

```
ImageName = "pink-cupcake.jpg"
```

10. Replace the selected code with the following code:

```
ImageName = "pink-cupcake.jpg",
CalorificValue = 295
```

11. In the **OnModelCreating** method code block, select the following code:

```
ImageName = "turquoise-cupcake.jpg"
```

12. Replace the selected code with the following code:

```
ImageName = "turquoise-cupcake.jpg",
CalorificValue = 360
```

13. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Views**, expand **Cupcake**, and then click **Details.cshtml**.

14. In the **Details.cshtml** code window, locate the following code:

```
<div>
    <p class="display-label">
            @Html.DisplayNameFor(model => model.Price)
    </p>
    <p class="display-field">
            @Html.DisplayFor(model => model.Price)
    </p>
</div>
```

15. Place the cursor after the **>** (greater than) sign of the ** ** tag, press Enter, and then type the following code:

go deploy

```
                @Html.DisplayNameFor(model => model.CalorificValue)
        </p>
        <p class="display-field">
                @Html.DisplayFor(model => model.CalorificValue)
        </p>
    </div>
```

16. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, under **Views**, under **Cupcake**, click **Edit.cshtml**.

17. In the **Edit.cshtml** code window, locate the following code:

```
<div class="form-field">
    <label asp-for="Price"></label>
    <input asp-for="Price" />
    <span asp-validation-for="Price"></span>
</div>
```

18. Place the cursor after the **>** (greater than) sign of the ** ** tag, press Enter, and then type the following code:

```
<div class="form-field">
    <label asp-for="CalorificValue"></label>
    <input asp-for="CalorificValue" />
    <span asp-validation-for="CalorificValue"></span>
</div>
```

19. In the **Cupcakes - Microsoft Visual Studio** window, in **Solution Explorer**, under **Views**, under **Cupcake**, click **Create.cshtml**.

20. In the **Create.cshtml** code window, locate the following code:

```
<div class="form-field">
    <label asp-for="Price"></label>
    <input asp-for="Price" />
    <span asp-validation-for="Price"></span>
</div>
```

21. Place the cursor after the **>** (greater than) sign of the ** ** tag, press Enter, and then type the following code:

```
<div class="form-field">
    <label asp-for="CalorificValue"></label>
    <input asp-for="CalorificValue" />
    <span asp-validation-for="CalorificValue"></span>
</div>
```

go deploy

- [ ] 23. In the **Package Manager Console** tab, type the following command, and then press Enter.

```
Add-Migration AddCupcakeCaloricificValue
```

> **Note**: In **Solution Explorer**, under **Migrations**, verify that a new migration file is created.

- [ ] 24. In the **Package Manager Console** panel, type the following command, and then press Enter.

```
Update-Database
```

## Task 6: Run the application

- [ ] 1. In the **Cupcakes - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.
- [ ] 2. on the **Debug** menu, click **Start Without Debugging**.
- [ ] 3. Select a cupcake of your choice, and then click **Details**.

> **Note**: The browser displays the Calorific value of the cupcake. If we had not done the database migration, we would get an error.

- [ ] 4. View the cupcake details, and then click **Back to List**.
- [ ] 5. In Microsoft Edge, click **Close**.
- [ ] 6. In the **Cupcakes - Microsoft Visual Studio** window, on the **File** menu, click **Exit**.

> ✓ **Results**: After completing this exercise, you should have created a cupcakes shop application in which users can add a new cupcake, edit a cupcake, delete a cupcake and view a cupcake's details.