Kuwait University

College of Computing Sciences and Engineering

Department of Information Science

COMP 2139 Web Application Development with C# .NET

Lab 10
The Quarterly Sales Application

Table of Contents

Laboratory #10 – The Quarterly Sales Application

1. Laboratory Objective

The goal of this lab is to build application that validates the data a user enters in a data-driven web application.

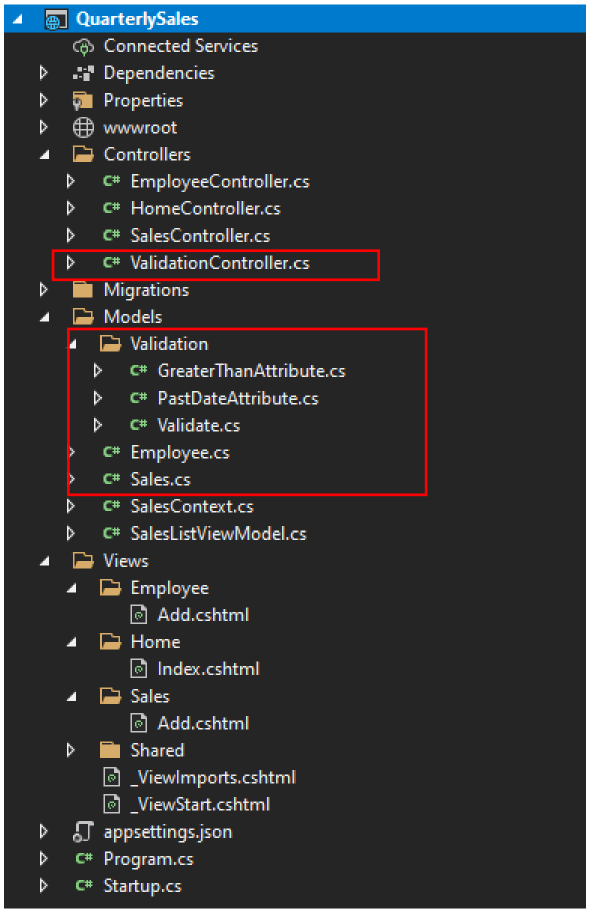
2. Laboratory Learning Outcomes: After conducting this laboratory students will be able to:

a. Create and maintain a Git repository project
b. Create the Quarterly Sales Application using C#.NET
c. Build a multi-page application, that validates users data.

3. Laboratory Instructions

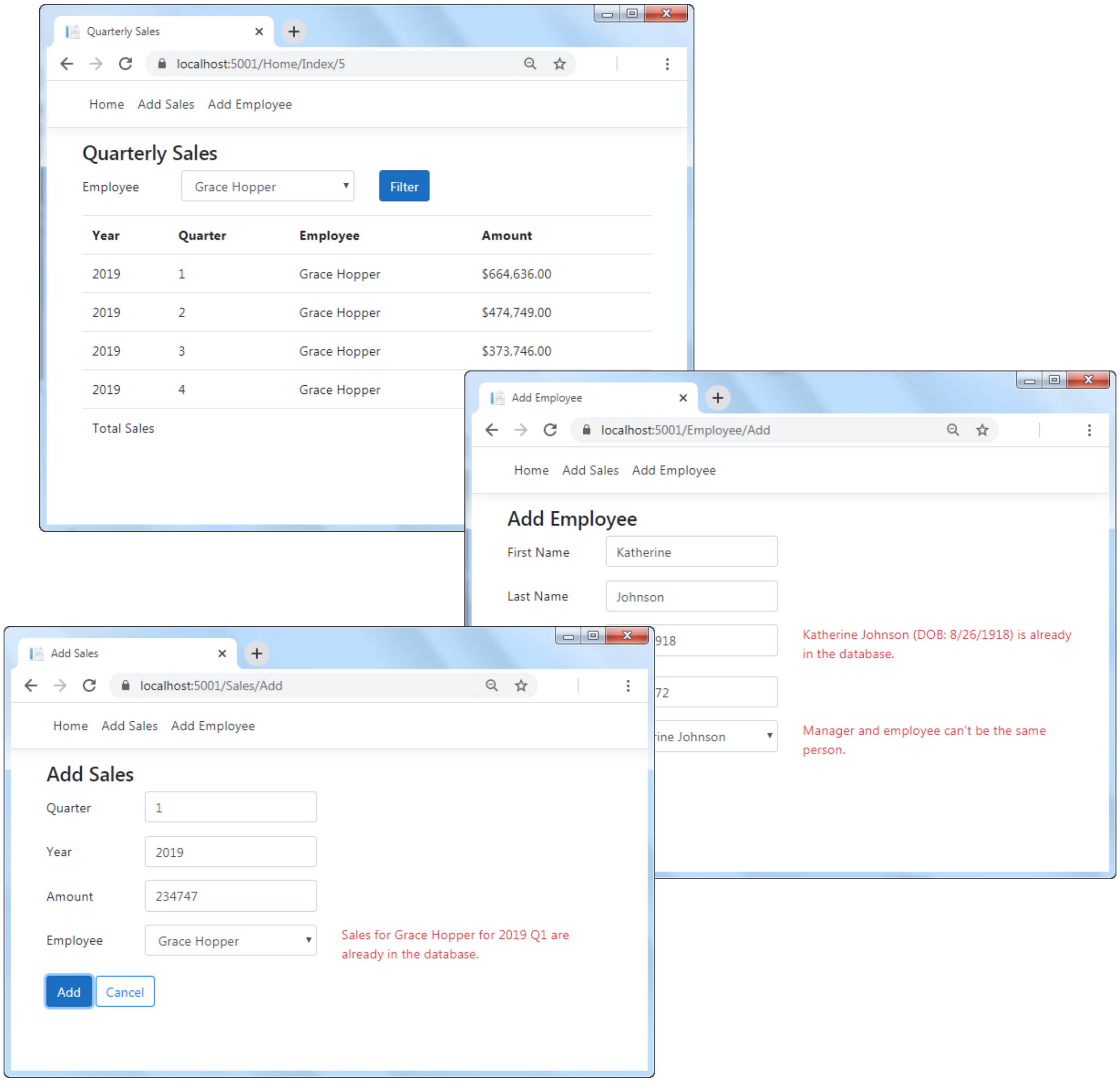
For this lab, we will build a multi-page, web application that will have a structure similar to the one presented below:

The Quarterly Sales Application



```
QuarterlySales
  Connected Services
  Dependencies
  Properties
  wwwroot
  Controllers
    EmployeeController.cs
    HomeController.cs
    SalesController.cs
    ValidationController.cs
  Migrations
  Models
    Validation
      GreaterThanAttribute.cs
      PastDateAttribute.cs
      Validate.cs
    Employee.cs
    Sales.cs
    SalesContext.cs
    SalesListViewModel.cs
  Views
    Employee
      Add.cshtml
    Home
      Index.cshtml
    Sales
      Add.cshtml
    Shared
    _ViewImports.cshtml
    _ViewStart.cshtml
  appsettings.json
  Program.cs
  Startup.cs
```

The Quarterly Sales Application



Quarterly Sales

localhost:5001/Home/Index/5

Home Add Sales Add Employee

Quarterly Sales

Employee: Grace Hopper — Filter

Year	Quarter	Employee	Amount
2019	1	Grace Hopper	$664,636.00
2019	2	Grace Hopper	$474,749.00
2019	3	Grace Hopper	$373,746.00
2019	4	Grace Hopper	
Total Sales			

Add Employee

localhost:5001/Employee/Add

Home Add Sales Add Employee

Add Employee

First Name	Katherine	
Last Name	Johnson	
	918	Katherine Johnson (DOB: 8/26/1918) is already in the database.
	72	
	ine Johnson	Manager and employee can't be the same person.

Add Sales

localhost:5001/Sales/Add

Home Add Sales Add Employee

Add Sales

Quarter	1	
Year	2019	
Amount	234747	
Employee	Grace Hopper	Sales for Grace Hopper for 2019 Q1 are already in the database.

Add Cancel

Specifications

- When the application starts, it should display quarterly sales data and provide a drop-down that allows a user to filter by employee.

- The web application should have a page to enter a new employee, and a page to enter new sales data. To keep things simple, this application doesn't provide edit or delete capabilities.

New Employee Specifications

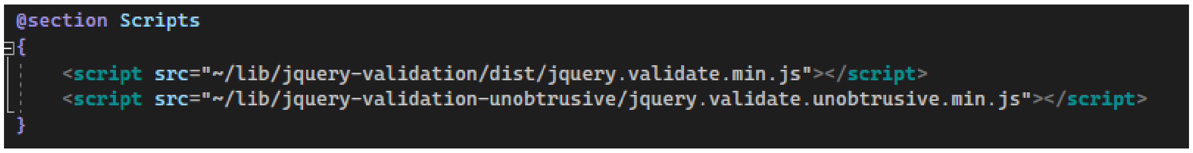
- Here are the requirements for valid data for a new employee:

    - First name, last name, date of birth, date of hire and manager are required
    - Date of birth and date of hire must be valid date (not in the past)
    - Date of hire must not be before 1/1/1995, the data the company was founded
    - A new employee may not have the same first name, last name and date of birth of an employee that already in the database
    - A new employee may not have a manager with the same first name, last name, and date of birth. That is, an employee and their manager may not be the same person.

New Sales Specifications

- Here are the requirements for valid data for a new sales amount:
    - Quarter, year, amount and employee are required
    - Quarter must be between 1 and 4
    - Year must be after the year 2000
    - Amount must be greater than 0
    - New sales data may not have the same quarter, year and employee as sales data that already in the database

- The web application should include the jQuery validation libraries so validation happens on the client, as a convenience to users.



```
@section Scripts
{
    <script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
    <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>
}
```

- For security, all validation should also take place on the server as well. To keep things simple, make the custom validation attributes only validate on the server. Thus, they will only fire after all client-side validation has passed.

- To make it possible to enter a new employee, you can edit the context file so it provides some seed data so the database includes at least one employee that you can select as a manager. For example

```
new Employee
{
  EmployeeId = 1,
  Firstname  = "Bruce",
  Lastname   = "Wayne" ,
  DOB        = new DateTime(1985, 12, 10),
  DateOfHire = new DateTime(1995, 1, 1),
  ManagerId  = 0
}
```

Creating Git Repository (Optional / Review)

1. Refer to past lab instructions on creating new repositories and clone them into your local workspace.

Creating the Web Application (Review)

1. Within Visual Studio select **File→New→Project**, select **ASP.NET Core Web Application**.
   a. Fill in your desired project details for Project Name, and Location, then select create
   b. From the resultant page, select **Web-Application (Model-View-Controller),** then **Create**
   c. You should then be presented with your workspace to start creating your project.
   d. Clean-up (delete/remove) all auto-generated **controllers**, **views**, **models** from the project

Create the Quarterly Sales Model

1. Based on the screenshot provided for the Quarterly Sales, create the application model classes, based on the following design:

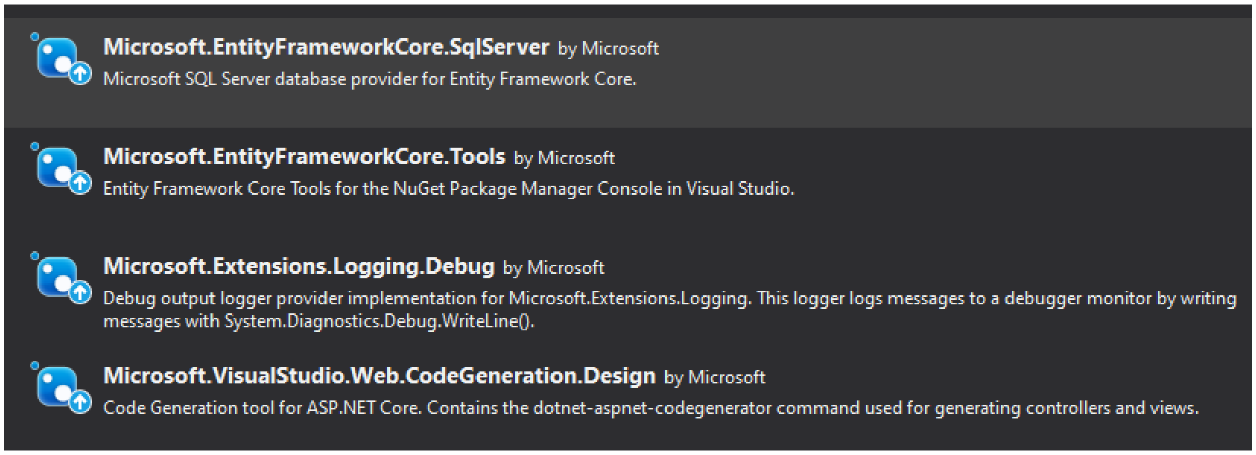
Employee

Attribute	**Type**	**Validation/Other**
EmployeeId	Integer	Primary Key
Firstname	string	Required StringLength Display(*Name*)
Lastname	String	Required StringLength Display(*Name*)
DOB	DateTime	Required PastDate (**custom**) Remote Display(*Name*)
DateOfHire	DateTime	Required PastDate (**custom**) GreaterThan (**custom**) Display(*Name*)
ManagerId	Integer	GreaterThan (**custom**) Remote Display(*Name*)
Method	Return Type	Description
Fullname	String	*Firstname + Lastname*

Sales

Attribute	Type	Validation/Other
SalesId	Integer	Primary Key
Quarter	Integer	Required Range
Year	Integer	Required GreaterThan (**custom**)
Amount	Double	Required GreaterThan (**custom**)
EmployeeId	Integer	GreaterThan (**custom**) Remote Display(*Name*)
Employee	Employee	None

Install Packages (NuGet)



Microsoft.EntityFrameworkCore.SqlServer by Microsoft
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by Microsoft
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Microsoft.Extensions.Logging.Debug by Microsoft
Debug output logger provider implementation for Microsoft.Extensions.Logging. This logger logs messages to a debugger monitor by writing messages with System.Diagnostics.Debug.WriteLine().

Microsoft.VisualStudio.Web.CodeGeneration.Design by Microsoft
Code Generation tool for ASP.NET Core. Contains the dotnet-aspnet-codegenerator command used for generating controllers and views.

Create the Context

1. Since these is a database required for this lab, you will need to create a context file, populating seed data for your Employee and Sales data.

Update appsettings.json

1. Configure your connection string to the database.

Update Program.cs

1. Default routing is sufficient
2. DbContext needs to be configured.

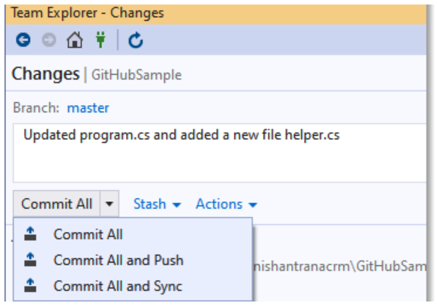
Create the ValidationController

1. The ValidationController requires the following action methods:
   i. JsonResult **CheckEmployee()**
   ii. JsonResult **CheckManager()**
   iii. JsonResult **CheckSales()**

Run and test the Application, then Commit

1. Run the application and test to make sure it functions as expected (correct any errors).

2. If all is well, commit the code to your code GitHub repository. To commit the code, navigate to the git changes tab, and select **commit all and push**



Updated program.cs and added a new file helper.cs

nishantranacrm\GitHubSam

Home Work

1. The HomeController requires the following action methods:
    - IActionResult **Index(Int id)** – [*HttpGet*]
    - RedirectToActionResult **Index(Employee)** – [*HttpPost*]

2. The EmployeeController requires the following action methods:
    - IActionResult **Index()**
    - IActionResult Add**()** – [*HttpGet*]
    - IActionResult Add**(Employee)** – [*HttpPost*]

3. The SalesController requires the following action methods:
    - IActionResult **Index()**
    - IActionResult **Add()** – [*HttpGet*]
    - IActionResult **Add(Sales)** – [*HttpPost*]

4. Based on the wireframe/screenshots provided, create the following views for the project namely:
    - Home/Index.cshtml
    - Employee/Add.cshtml
    - Sales/Add.cshtml
    - _Layout.cshtml