Kuwait University

College of Computing Sciences and Engineering

Department of Information Science

COMP 2139
Web Application Development with C# .NET

Lab 7
Trip Log Application

Table of Contents

Laboratory #7 – Trip Log Application

1. Laboratory Objective

The goal of this lab is to build a data-driven application that uses models to pass data from controllers to views, the ViewBag object to pass data from views to the layout, and TempData to persist data between requests.

2. Laboratory Learning Outcomes:

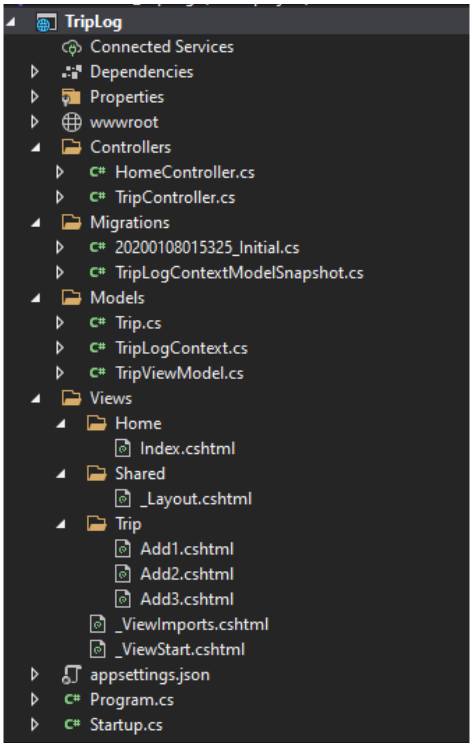
After conducting this laboratory students will be able to:

a. Create and maintain a Git repository project
b. Create a Trip Log Application using C#.NET
c. Build a multipage web application, with multiple views, nested layouts, layout sections, utilizing TempData and ViewModel.

3. Laboratory Instructions

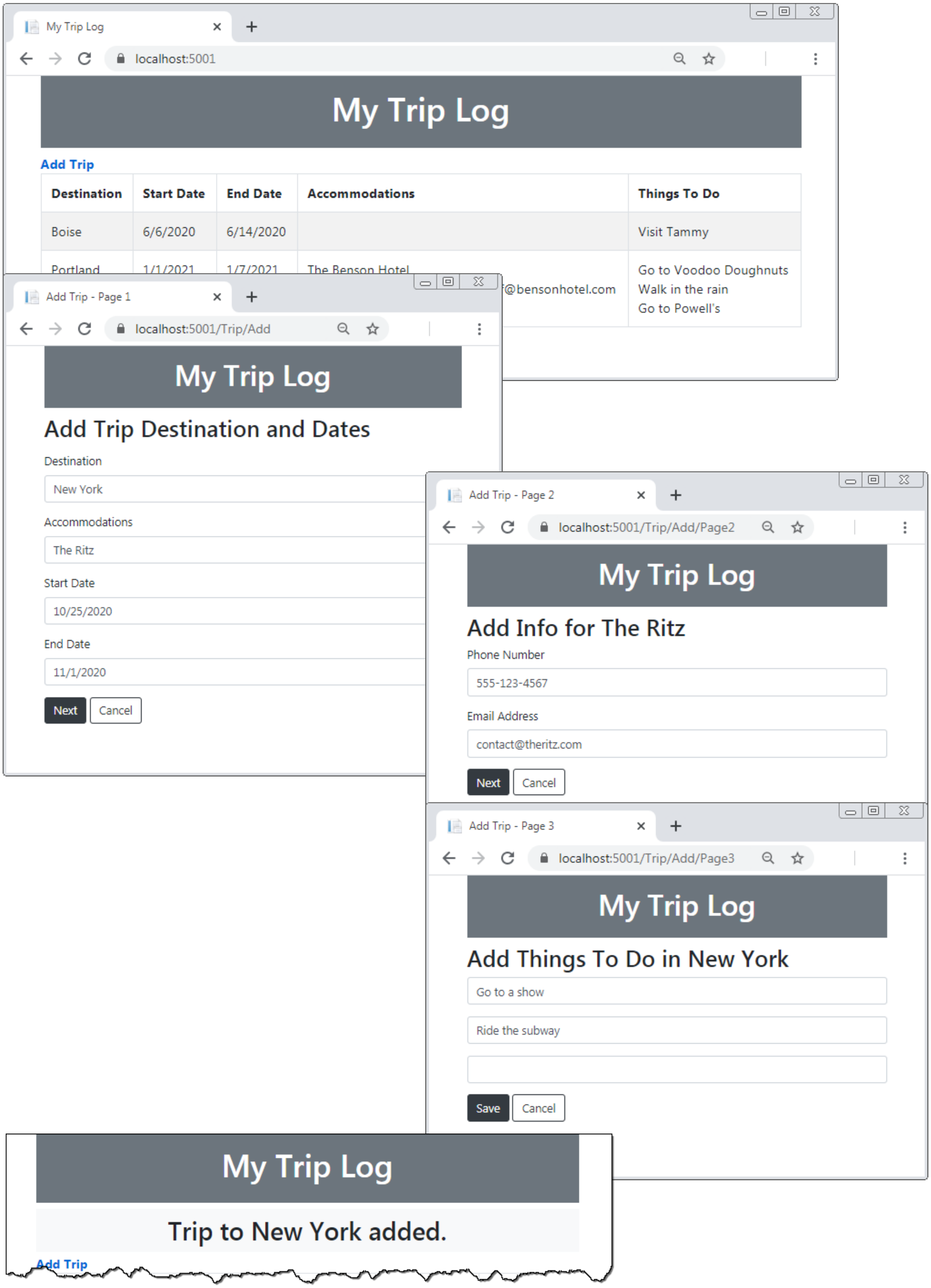
For this lab, we will build a multi-page, web application that will have a structure similar to the one presented below:

My Trip Log



```
TripLog
  Connected Services
  Dependencies
  Properties
  wwwroot
  Controllers
    HomeController.cs
    TripController.cs
  Migrations
    20200108015325_Initial.cs
    TripLogContextModelSnapshot.cs
  Models
    Trip.cs
    TripLogContext.cs
    TripViewModel.cs
  Views
    Home
      Index.cshtml
    Shared
      _Layout.cshtml
    Trip
      Add1.cshtml
      Add2.cshtml
      Add3.cshtml
    _ViewImports.cshtml
    _ViewStart.cshtml
  appsettings.json
  Program.cs
  Startup.cs
```

The Trip Log Application



My Trip Log

Add Trip

Destination	Start Date	End Date	Accommodations	Things To Do
Boise	6/6/2020	6/14/2020		Visit Tammy
Portland	1/1/2021	1/7/2021	The Benson Hotel @bensonhotel.com	Go to Voodoo Doughnuts Walk in the rain Go to Powell's

My Trip Log

Add Trip Destination and Dates

Destination: New York

Accommodations: The Ritz

Start Date: 10/25/2020

End Date: 11/1/2020

My Trip Log

Add Info for The Ritz

Phone Number: 555-123-4567

Email Address: contact@theritz.com

My Trip Log

Add Things To Do in New York

Go to a show

Ride the subway

My Trip Log

Trip to New York added.

Add Trip

Specifications

- When the application starts, it should display trip data and an Add Trip link.

- The layout for this web application should display a banner at the top of the page, and a subheading below it. However, the subheading should only display if there's text in the *ViewData* property for it.

- When the user clicks the Add Trip link, a three-page data entry process should start.

- **On the first page**, the user should enter data about the trip destination and dates. The Accommodations field no this page should be optional, and the rest should be required.

- **The second page** should only display if the user entered a value for the Accommodations field on the first page. On this page, the user should enter data about the accommodations. The accommodations value the user entered should display in the subheading, and the fields should be optional.

- **On the third page**, the user should enter data about things to do on their trip. The destination the user entered on the first page should display in the subhead, and the fields should be optional.

- When the user clicks the Next button on the first or second page, the web application should save the data posted from the page(s) in *TempData*. Use this data to get the user entries to display in the subheadings as needed, but make sure any data that needs to be saved to the database persists in *TempData*.

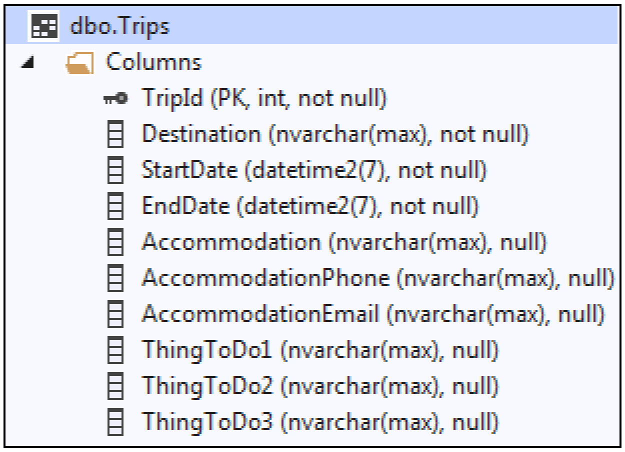
  - TempData. ASP.NET Core exposes the Razor Pages TempData or Controller TempData. This property **stores data until it's read in another request**. The Keep(String) and Peek(string) methods can be used to examine the data without deletion at the end of the request.
  - Keep(String): Marks a key string (in the dictionary) for retention
  - Peek(String): The key of the element to return

- When the user clicks the Save button on the third page, the web application should save the data posted from the page and the data in *TempData* to the database. Then the Home page should display with a temporary message that the trip has been added.

- When the user clicks the Cancel button on any of the *Add* pages, the data in *TempData* should be cleared and the Home page should display.

  You can use this statement to clear the data: **TempData.Clear();**

- To keep things simple for this lab, keep all fields for a trip in a single table, a screenshot has been provided to simplify the process:



dbo.Trips

Columns

- TripId (PK, int, not null)
- Destination (nvarchar(max), not null)
- StartDate (datetime2(7), not null)
- EndDate (datetime2(7), not null)
- Accommodation (nvarchar(max), null)
- AccommodationPhone (nvarchar(max), null)
- AccommodationEmail (nvarchar(max), null)
- ThingToDo1 (nvarchar(max), null)
- ThingToDo2 (nvarchar(max), null)
- ThingToDo3 (nvarchar(max), null)

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 Trip Model

1. Based on the screenshot provided in create the trip model screenshot, create the trip model with the following attributes:
   - TripId
   - Destination
   - StartTime
   - EndDate
   - Accommodation
   - AccommodationPhone
   - AccommodationEmail
   - ThingToDo1
   - ThingToDo2
   - ThingToDo3

Create the TripLogContext

1. Create the **TripLogContext** which will be utilized to provide access to the Trip Log database.

Create the TripViewModel

As mentioned in lecture, a *ViewModel* is used to shape multiple entities from one or more models into a single object. The conversation into a single model provides better optimization for those views that require data from more than one model.

1. Since each of our Add pages (*Add1*, *Add2*, *Add3*) display Trip information and a page number lets create *ViewModel* called *TripViewModel* that make this information available for those views.

   The act of creating a *ViewModel,* is the same process as creating a regular model (class .Net) just with the *ViewModel* term appended to the end of the class name.

   The *TripViewModel* only require a Trip (accessor/mutator) and Page Number (int, accessor/mutator)

Update appsettings.json and Program.cs

1. Update the web applications **appsettings.json** and **Program.cs** file with the database connection string for the project.

Create the HomeController

1. The HomeController for the TripLog Application, should have only a single index action method, simple return the result set for querying the complete TripLog database to then be used to render the results on screen. In order to achieve this, make sure you have injected your TripLogContext into you HomeControllers constructor.

Create the TripController

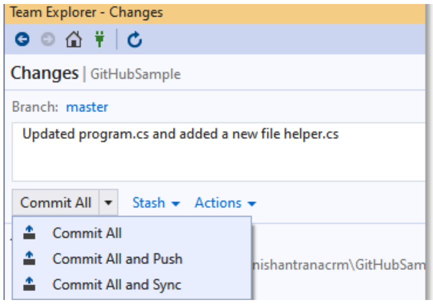
1. The TripController can be dissected as follows:

   - Require a TripLogContext courtesy of constructor dependency injection
   - **Index()** method which redirects the user to then HomeControllers Index() action method
   - **Add(string)** method that services the **HTTP GET** request and returns a ViewResult
   - **Add(TripViewModel)** method that services the **HTTP POST returns** IActionResult
   - **Cancel()** method which redirects the user to the Index() action in the HomeController.

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**



Branch: master

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

Home Work

1. Based on the wireframe/screenshots provided, create the following views for the project namely:
    - Index.cshtml
    - Add1.cshtml
    - Add2.chtml
    - Add3.cshtml
    - _Layout.cshtml