MVC Maintenance Exercise

Student Information System (SIS)

You have been hired at the SG College to help complete their SIS system. It has been under development, but has an outstanding list of tasks that need completed. Your job is to familiarize yourself with the application as it exists currently and evaluate, plan, and implement the additional features specified for the initial release of the application.

Table of Contents

ADMIN SCREENS **1**
- MAJORS ADMINISTRATION 1
- STATES ADMINISTRATION 2
  - *Add State Link to Admin Menu* *2*
  - *Setup admin controller for States List* *2*
  - *Create the States List View* *2*
  - *Setup admin controller for Adding State* *3*
  - *Create the AddState View* *3*
  - *Setup admin controller for Editing State* *4*
  - *Create the EditState View* *4*
  - *Setup the admin controller for deleting states* *5*
  - *Create the DeleteState View* *5*
- COURSES ADMINISTRATION 5

STUDENT INFORMATION **5**
- STUDENT LIST VIEW 6
- STUDENT ADD VIEW 6
- STUDENT EDIT VIEW 6
- STUDENT DELETE VIEW 6

Admin Screens

Majors Administration

Administration of college majors is completed. You can review this code to understand the workflow of administration pages.

States Administration

You must complete the administration screens for managing the states that we offer services in. We are currently in Ohio, Kentucky and Minnesota, but expansion as we expand we have to be able to manage the states.

The repository for states is complete. The remaining tasks are detailed below.

Add State Link to Admin Menu

Locate the **_Layout.cshtml** page in the **Views -> Shared** and create a link in the main menu under the admin section that will load the State list page.

Setup admin controller for States List

Create a method in the Admin Controller for States. This method should return a view to lists all the existing states.

1. The return type for this method should be `ActionResult`.
2. This method should be created as a `HttpGet` (this is the default so an annotation denoting this is optional).
3. The method should get a list of existing states from the repository and pass it as a parameter to the view returned by the method

Create the States List View

1. Right click on the controller method name and choose **Add View**.
2. Select **Empty (without model)**
3. Leave the default view name in the dialog. It should match the name of the method you are creating it from.
4. Leave the **Use a layout page** checkbox as checked.
5. Click the **Add** button.

   *NOTE – Once the view is created, it will appear in the **Views -> Admin** folder under your MVC project in the solution explorer. By convention MVC looks for your view in a folder under Views that matches the name of the controller from which it was created.*

6. Add the correct declaration at the top of the view to indicate the type of the model that will be passed to the view.

7. In the view, create some HTML that contains a table to display all states currently stored in our data repository.

    a. There should be a column header for State Abbreviation and one for State Name.

    b. Use the razor syntax in the <tbody></tbody> of the table to create a foreach loop that will display each of the states.

8. Add a button/link to the States list view that will call the `AddState` method in the Admin controller.

Setup admin controller for Adding State

1. Create a `GET` method in the Admin controller for adding states (`AddState`).

    a. The method will return a view with an empty State model passed to it.

2. Create the `POST` method in the Admin controller for adding states (AddState).

    a. The method should redirect the user to the States list view upon successful completion of the Add operation.

Create the AddState View

1. Right click on the controller method name and choose **Add View**.

2. Select **Empty (without model)**

3. Leave the default view name in the dialog. It should match the name of the method you are creating it from.

4. Leave the **Use a layout page** checkbox as checked.

5. Click the **Add** button.

6. Add the correct declaration at the top of the view to indicate the type of the model that will be passed to the view.

7. In the view create a form that will collect the `StateAbbreviation` and `StateName` from the user and POST it back to the `AddState HttpPost` method in the Admin controller.

8. Add a link to the States list view table (one link per row of data) that will call the `EditState` method in the Admin controller passing the `StateAbbreviation` of the specific row to the controller to identify which one to edit.

Setup admin controller for Editing State

1. Add a `GET` method in the Admin controller for EditState.

    a. The method should take a string parameter for `StateAbbreviation`

2. Create the `POST` method in the Admin controller for editing states (EditState).

    a. The method should redirect the user to the States list view upon successful completion of the Edit operation.

Create the EditState View

1. The view creation processes should be similar to those specified in previous steps.

2. Once the view is created, add a form similar to the one specified in the AddState view that will post the data back to the server.

3. Add a link to the States list view table (one link per row of data) that will call the `DeleteState` method in the Admin controller passing the `StateAbbreviation` of the specific row to the controller to identify which one to delete.

Setup the admin controller for deleting states

1. Add a `GET` method in the Admin controller for DeleteState.
    a. The method should take a string parameter for `StateAbbreviation`
2. Create the `POST` method in the Admin controller for deleting states (DeleteState).
    a. The method should redirect the user to the States list view upon successful completion of the Delete operation.

Create the DeleteState View

1. The view creation processes should be similar to those specified in previous steps.
2. Once the view is created, add a form that will contain a hidden field for StateAbbreviation that will post the data back to the server.
3. The view should warn the user of the impending deletion and allow them to proceed with the delete or cancel.
4. If the user chooses to cancel, redirect them to the States list view.
5. If the user chooses to proceed with the delete, post the form to the DeleteState Post method in the Admin controller.

Courses Administration

There is a need for administration screens to support Listing, Adding, Editing and Deleting courses offered. Implement these screens in a similar fashion to the existing admin screens.

Student Information

The Student Information section of the site is partially complete.

Student List View

The student list view is completed with the exception of adding links to each row of data in the table to allow for editing and deleting students.

Student Add View

The student add view is complete but should be manually tested to ensure proper functionality.

Student Edit View

A view will need added for editing student data. This will require additions to the Student controller as well.

The Student Edit view should contain a form for editing the personal details as created in the student add view. There should be an additional form along side the personal details that will allow for saving the address of the student.

Student Delete View

A view will need added for confirming the user wishes to delete the specified student.