CIS 3325
Assignment # 6 (25 points)

Due Date: Beginning of class on Thursday, November 14, 2019

To be done by each student individually

Introduction:

This is a completely new assignment. It is not related to the SportsPro application you have been working on for the last several projects. Assignment # 6 comprises of making a data request to a Web REST API and displaying the results in a client form.

This application must be designed to be compliant with the MVC design paradigm, using **Object Data** sources, class Library projects to hold all business and data access controllers and business entity classes, as well as Newtonsoft JSON class library.

Requirements for Assignment # 6:

1. ***Required reading:*** Class Notes based on class demos developed to showcase how we consume JSON data in a WinForms application.
2. You will need to download and install the NewtonSoft's JSON class library. You can accomplish this via the NuGet Package Manager tool (refer to class notes or external resources posted in TRACS).
3. It will be helpful to use JSON online editor to see the structure of the raw JSON stream returned to pattern your C# business entity classes and manage deserialization. JSON2CSharp and JSON Utilities are other websites to help generate business classes to pattern JSON structure returned.
4. **ALL NAMING CONVENTIONS MUST BE FOLLOWED.**

Naming Requirements

1. Solution: Assignment6-YourName   e.g. Assignment6-MayurMehta
2. User Interface Project: WebApiUserInterface
3. Business Class Library Project: WebApiBusinessClassLibrary
4. Data Access Class Library Project: WebApiDataAccessClassLibrary
5. Please follow **ALL NAMING CONVENTIONS.**

Submission Requirements:

1. The assignment is to be completed individually by each student. This is a requirement. Collaborative submissions and work will be treated as academic dishonesty and handled accordingly.
2. Your **complete and fully working Solution** should be available at the ***root level of your USB drive. Non-working projects (meaning your solution contains any build or runtime errors) should not be submitted for grading.***
3. Submit the USB drive with your completed and running solution as well as the following printouts **arranged per sequence below**:
   a. A cover page with your name, project name, and submission date.
   b. A screenshot of the Brewery Form showing brewery data returned by the Web Api for brewery type **brewpub** in the state of **Arizona**.
   c. Printout of the controller code page for FrmBreweryInfo.

d. Printouts of all business and database classes listed in section B above. Please arrange them in the order listed there (printouts of Brewery entity class followed by that of BreweryBL, and then BreweryDA).

e. Staple the printouts in the above order and **hook** in your USB drive with a paper clip or a string into the staple.

4. Place the USB drive and your printouts on the instructor's desk **before the start of class on the due date. Once the instructor starts class, the assignment is considered late and will not be accepted for grading.**

A. User Interface Requirements:

Assignment # 6 will require you to access a Web REST API that manages a repository of data on breweries in the USA. This data is made available for testing purposes and following sample illustrates the type of data available on each brewery.

- **id**: 2,
- **name**: "Avondale Brewing Co",
- **brewery_type**: "micro",
- **street**: "201 41st St S",
- **city**: "Birmingham",
- **state**: "Alabama",
- **postal_code**: "35222-1932",
- **country**: "United States",
- **longitude**: "-86.774322",
- **latitude**: "33.524521",
- **phone**: "2057775456",
- **website_url**: "http://www.avondalebrewing.com"

The Application will comprise of only one form named "**FrmBreweryInfo**". The form will comprise of two text boxes and two command buttons, and one datagridview control. The form and the DataGridView must be sized to display the brewery information completely, without displaying horizontal scroll bars in the DataGridView.

The two textboxes will be used to enter two search criteria – brewery type and state name to look for breweries of specified type in the specified state. One of the two command buttons will initiate the call to get the data from the Web API per the MVC design paradigm while the other command button will enable the user to exit the application. The datagridview will display the information on just the **id, name, street, city, state, postal_code, phone number, and website_url** for the breweries of specified type (micro, regional, brewpub, large, planning, bar, contract, and proprietor) and specified state. Please note that search parameter values are case sensitive. So, by_state = New_York is not the same as by_state = new_york. Also, brewery types and state names in the search parameter values are all in lower case.

The Web API's endpoint is

https://api.openbrewerydb.org/breweries?by_type=XXX&by_state=YYY

where XXX is the brewery type (one in the list provided above) while YYY is the name of the specified state. The type and name search parameter values must be in **ALL lower case**. If the state name is comprised of two words e.g. New York then separate the words with an underscore i.e. new_york (Again, note that state names are in all lower case). Your program needs to make sure that whatever is entered into the two textboxes must be converted to all lower case before inserting those values in the above URL string.

The website **https://api.openbrewerydb.org** provides ample documentation on how to form the URL request using different search criteria. Use the Documentation link from the landing page of this website to access examples.

For this project, you will design the form on your own so use your creative skills to design a visually appealing and information appropriate form. Significant part of your grade will be based on the overall design (form, classes, methods within classes, etc.)

The DataGridView **MUST** be ***bound to an object data source*** based on the business entity class that will be derived from the data stream received from the Web API in JSON Format. The DataGridView should be populated when the user clicks on the command button to retrieve breweries of specified type in the specified state.

The form must be centered in the screen. The DataGridView should be anchored within the form so that it sizes proportionately with the form size.

This UI will make use of several business and data access classes to display the needed information. These classes are describe below:

B. Business Class Library (WebApiBusinessClassLibrary)

The following business classes will be used in this project.

Class Name	Description
Brewery	A business entity class that represents a single brewery. The class is derived from the JSON data stream returned by the Web API.
BreweryBL	A business class that contains method(s) to invoke the Data Access class and format the data for return to the UI.

C. DataAccess Class Library (WebApiDataAccessClassLibrary)

Class Name	Description
BreweryDA	A data access class that contains method(s) to access the Web API and manage data received from the Web API. The endpoint URL will be constructed in a method within this class by concatenating the brewery type and state name passed in as arguments.

Within each BreweryBL and BreweryDA class, you will determine needed methods to include, the data type each method should return and arguments to be passed in to each method. This is where you bring in your own thoughts, analysis and creative work into play. The instructor will provide only minimal instructions and assistance on this project. You will have to figure out the complete logic based on MVC architecture and instructions provided here. Remember, you **must** implement an MVC architecture. Therefore, the UI can only interact with the BL classes; BL classes can only interact with the DA classes; and DA class can interact only with the Web API endpoint.

All Exceptions, including JSON, Web, and general, must be properly and correctly handled. This means displaying *customized, user-oriented messages for JSON and Web exceptions.*

D. **Processing Tasks**

Operation

- The **FrmBreweryInfo** should be displayed in the **center of the screen** when the application is started.
- The user will enter the brewery type and state name in their respective text boxes and click the command button to request brewery info from the Web API endpoint via the BrewryBL and BreweryDA class methods.
- The DataGridView should be populated with the received list of breweries, showing the data for items indicated earlier. The DataGridView control **must be bound to an object source** that will provide values for each brewery returned in the list.
- User clicks on the Exit button to close the form and exit the application.

Specifications

- DataGridView on the FrmBreweryInfo **must be data bound to an object data source**. The data will be displayed by setting the datasource property of the BindingSource, and not that of the DataGridView.
- **You must check all received data to ensure that it contains data before processing the received data. Your program must handle situations appropriately where request did not result in any data being received.**
- The object data source will be associated with business entity class Brewery.
- The Brewery class is derived from the returned JSON stream.
- You will need to evaluate the returned JSON Stream to determine its data structure and define a business entity class to hold data once the stream is deserialized. Use NewtonSoft's JSON class library to download the data and accomplish deserialization into business objects.
- Before you can use the classes in class libraries, you must compile (Or build) the library and add a reference to the project where its classes are used. You should also add a **using** directive for the library where needed.

Academic Dishonesty:

1. **This is an individual assignment. Each individual is to do his/her own work.**
2. **Any collusion or sharing of work among individuals will be considered an academic dishonesty and will be handled in accordance with the Texas State's Honor Code.**
3. **This instructor is serious about enforcing this policy to the fullest extent possible.**