Trisha Echual

Hugo Garcia

CECS 475

Professor Phuong Nguyen

May 10, 2018

Lab Write-Up

**Part 1 – Added Code**

*Underlined code signify added or updated code.*

|  |
| --- |
| Namespace FlickrViewer |
|  | { |
|  | public partial class FlickrViewerForm : Form |
|  | { |
|  | // Flickr API key |
|  | private const string KEY = "5d07e5f58a358cd51c325e0a518af32c"; |
|  |  |
|  | // Create an WebClient object to invoke Flickr web service |
|  | private WebClient flickrClient = new WebClient(); |
|  |  |
|  | // Initialize a Task<String> object called flickrTask to null |
|  | // Task<string> queries Flickr |
|  | Task<string> flickrTask = null; |
|  |  |
|  | /// <summary> |
|  | /// Constructor initializes viewer form |
|  | /// </summary> |
|  | public FlickrViewerForm() |
|  | { |
|  | InitializeComponent(); |
|  | } |
|  |  |
|  | /// <summary> |
|  | /// Initiates the asynchronous Flickr search query and displayes results when query completes |
|  | /// </summary> |
|  | /// <param name="sender"></param> |
|  | /// <param name="e"></param> |
|  | private async void searchButton\_Click( object sender, EventArgs e ) |
|  | { |
|  | // Check whether user started a search previously |
|  | if ( flickrTask != null && flickrTask.Status != TaskStatus.RanToCompletion ) |
|  | { |
|  | var result = MessageBox.Show( |
|  | "Cancel the current Flickr search?", |
|  | "Are you sure?", MessageBoxButtons.YesNo, |
|  | MessageBoxIcon.Question ); |
|  |  |
|  | // If user clicks No, the event handler returns |
|  | if (result == DialogResult.No) |
|  | return; |
|  | else |
|  | flickrClient.CancelAsync(); // cancel current search |
|  | } |
|  |  |
|  | // Create a URL for invoking the Flickr web service's method flickr.photos.search |
|  | var flickrURL = string.Format("https://api.flickr.com/services" + |
|  | "/rest/?method=flickr.photos.search&api\_key={0}&tags={1}" + |
|  | "&tag\_mode=all&per\_page=500&privacy\_filter=1", KEY, |
|  | inputTextBox.Text.Replace(" ", ",")); |
|  |  |
|  | imagesListBox.DataSource = null; |
|  | imagesListBox.Items.Clear(); // clear imagesListBox |
|  | pictureBox.Image = null; // clear pictureBox |
|  | imagesListBox.Items.Add( "Loading..." ); // display Loading... |
|  |  |
|  | try |
|  | { |
|  | // Call WebClient's DownloadStringTaskAsync method to request information |
|  | flickrTask = flickrClient.DownloadStringTaskAsync(flickrURL); |
|  |  |
|  | // await fickrTask then parse results with XDocument |
|  | XDocument flickrXML = XDocument.Parse(await flickrTask); |
|  |  |
|  | // Gather from each photo element in the XML the id, title, secret, server, and farm attributes |
|  | var flickrPhotos = |
|  | from photo in flickrXML.Descendants( "photo" ) |
|  | let id = photo.Attribute("id").Value |
|  | let title = photo.Attribute("title").Value |
|  | let secret = photo.Attribute("secret").Value |
|  | let server = photo.Attribute("server").Value |
|  | let farm = photo.Attribute("farm").Value |
|  | select new FlickrResult |
|  | { |
|  | Title = title, |
|  | URL = string.Format("http://farm{0}.staticflickr.com/{1}/{2}\_{3}.jpg", |
|  | farm, server, id, secret) |
|  | }; |
|  | imagesListBox.Items.Clear(); |
|  |  |
|  | // set ListBox properties only if results were found |
|  | if ( flickrPhotos.Any() ) |
|  | { |
|  | imagesListBox.DataSource = flickrPhotos.ToList(); |
|  | imagesListBox.DisplayMember = "Title"; |
|  | } |
|  | else // no matches were found |
|  | imagesListBox.Items.Add( "No matches" ); |
|  | } |
|  | catch ( WebException ) |
|  | { |
|  | // check whether Task failed |
|  | if ( flickrTask.Status == TaskStatus.Faulted ) |
|  | MessageBox.Show( "Unable to get results from Flickr", |
|  | "Flickr Error", MessageBoxButtons.OK, |
|  | MessageBoxIcon.Error ); |
|  | imagesListBox.Items.Clear(); // clear imagesListBox |
|  | imagesListBox.Items.Add( "Error occurred" ); |
|  | } |
|  | } |
|  |  |
|  | /// <summary> |
|  | /// Display selected image |
|  | /// </summary> |
|  | /// <param name="sender"></param> |
|  | /// <param name="e"></param> |
|  | private async void imagesListBox\_SelectedIndexChanged(object sender, EventArgs e) |
|  | { |
|  | if ( imagesListBox.SelectedItem != null ) |
|  | { |
|  | string selectedURL = ((FlickrResult)imagesListBox.SelectedItem).URL; |
|  |  |
|  | // use WebClient to get selected image's bytes asynchronously |
|  | WebClient imageClient = new WebClient(); |
|  |  |
|  | byte[] imageBytes = await imageClient.DownloadDataTaskAsync(selectedURL); |
|  |  |
|  | MemoryStream memoryStream = new MemoryStream(imageBytes); |
|  |  |
|  | pictureBox.Image = Image.FromStream(memoryStream); |
|  | } |
|  | } |
|  | } |
|  | } |

**Part 2 – Reflections**

We both have had introductory experiences with ASP.NET MVC through our senior project, so many aspects of the material in the tutorial were familiar. The tutorial was straight-forward and fairly easy. The only hiccups occurred when configuring SQL Server LocalDB and remembering to bind data in a View after its Model and Controller were updated. Other than that, the tutorial was actually helpful in allowing us to understand top-down the MVC pattern. The most value we received from this tutorial is gaining a better understanding of how to utilize LocalDB and seed data, as well as using data annotations to engage basic validation in a project.