INT422 Assignment 3

Common interaction patterns in a web app that uses a persistent store.

Read/skim all of this document before you begin work.

Due date

Section A: Thursday, Sept 29, 2016, at 11:59pm ET

Section B: Wednesday, Sept 28, 2016, at 11:59pm ET

Grade value: 4% of your final course grade

If you wish to submit the lab before the due date and time, you can do that.

Objective(s)

Implement some of the common interaction patterns in a web app that uses a persistent store.

Introduction to the problem to be solved

In the previous assignment, you implemented some of the common interaction patterns. In this assignment, you will implement the remaining patterns.

Specifications overview and work plan

Here's a brief list of specifications that you must implement:

- Follows best practices
- Implements the recommended system design guidance
- Customized appearance, with an added menu item
- Enables the "edit existing" use case for employee objects
- Implements LINQ-enabled filtering and sorting for track objects

Here is a brief work plan sequence:

1. Create the project, based on the project template
2. Customize the app's appearance
3. Create view models and mappers that cover the use cases
4. Add methods to the Manager class that handle the use cases
5. Add controllers, with code to work with the manager object
6. For the employee entity, implement the "get all" use case; including controller code, and view
7. Implement the "edit existing" use case; including controller code, and view

8. For the track entity, implement some “get all” filtered/sorted use cases; including controller code, and view

During the class/session, your professor will help you *get started* and *make progress* on this assignment.

Every week, in the computer-lab class/session, your teacher will record a grade when you complete a specific small portion of the assignment. We call this “*in-class grading*“.

The *in-class grading* will be announced in-class by your professor.

Getting started

Create a new web app, named Assignment3. It MUST use the “Web app project v2” project template.

If you see an error during project creation, asking you to upgrade the database, then use the previously-learned procedure to edit the connection string in the project’s Web.config file, changing “v11.0” to “MSSQLLocalDB”.

Build/compile, and run the app, to ensure that you are starting with a working error-free base. Then, as you write code, build/compile frequently.

Customize the app’s appearance

You will customize the appearance all of your web apps and assignments. Never submit an assignment that has the generic auto-generated text content. Make the time to customize the web app’s appearance.

For this assignment, you can defer this customization work until later. Come back to it at any time, and complete it before you submit your work.

Follow the guidance from Assignment 1 to customize the app’s appearance.

Create view models and mappers that cover the use cases

We will be working with the employee and track entities.

Tip:

> Study the DesignModelClasses.cd class diagram that’s in the Models folder.
>
> It will help you visualize where the Employee and Track entities are located in the design model.

As noted above, these use cases need view models:

- Employee – “get all”
- Employee – “edit existing”
- Track – “get all”

Go ahead and write those view model classes.

As with Assignment 2 (and the code examples), you can ignore navigation properties, and the Employee.ReportsTo property in the Employee class.

The Track design model class has an attribute named “Column”. It cannot be used in a view model class, so do NOT include it in your track base view model class.

All view models require suitable data annotations. Study each property, and determine which data annotations should be added. No, we will not tell you which ones to add.

Your “edit existing” use case will permit the editing of the following employee properties:

- Address, City, State, Country, PostalCode, Phone, Fax, Email

In addition, your “edit existing” view models will support the editing of some “extra” properties (which, of course, will not be saved – they will exist only in the view model and HTML Form). The extra properties:

- Password, text, should be masked as dots on the HTML Form
- Number of weeks vacation, integer, must be between 2 and 6

As you have recently learned, the typical pattern for “edit existing” is to write two view model classes. One will carry data to the HTML Form, and the other describes the data package that is posted back to the controller method from the browser user.

Mappers

Define the maps that these use cases will need. At this point in time, you should have enough experience to know which maps are required. Ask if you need help.

Add methods to the Manager class that handle the use cases

The class notes and code examples have all you need to implement this part of the work plan.

For the “get all” use cases on the track entity, several methods will be needed, and each will use LINQ query expressions to filter and sort the results. All will return an IEnumerable<TrackBase>, as you would expect. The suggested methods:

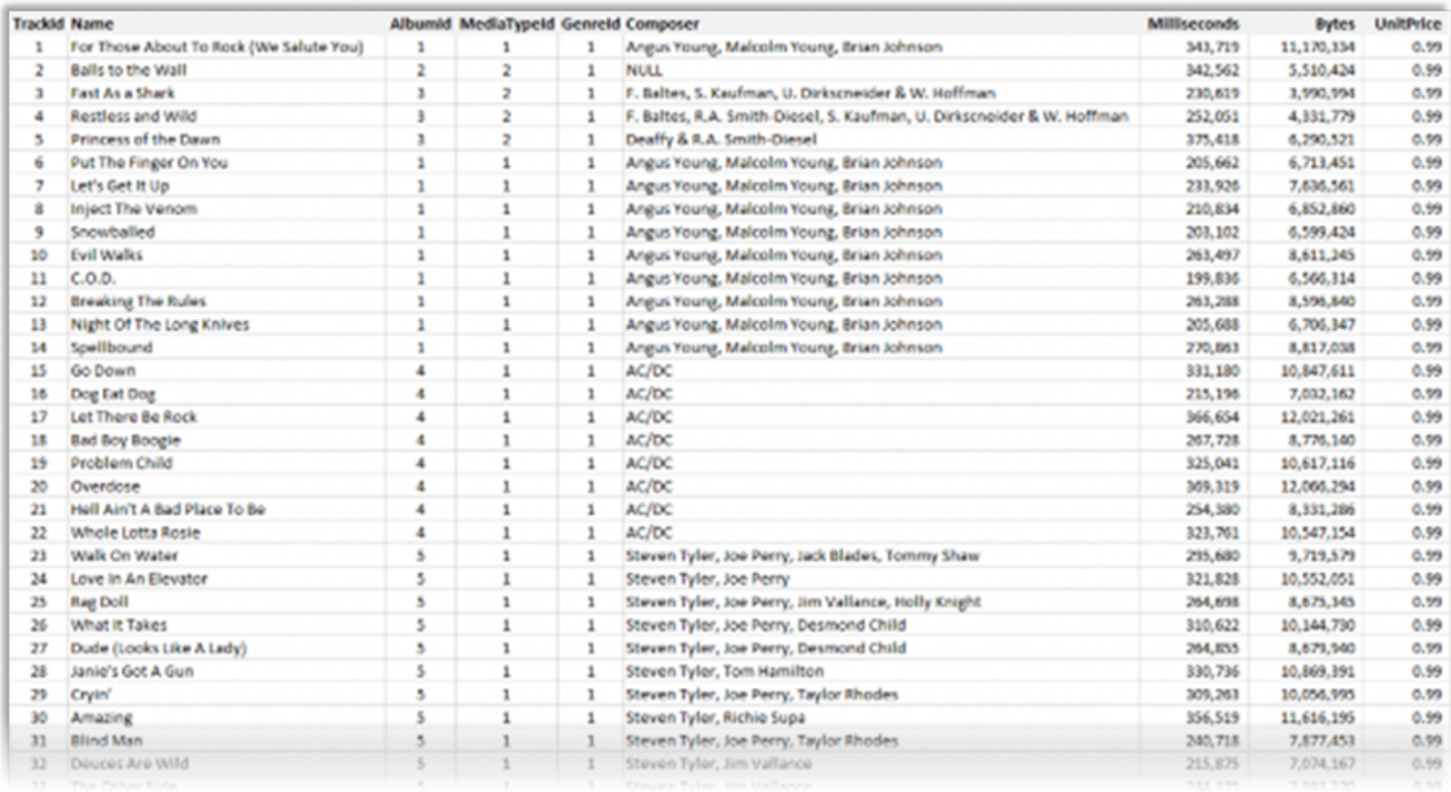
TrackGetAll – Typical “get all” method, but you should sort the results (on two properties) in a way that makes sense to you.

TrackGetAllPop – GenreId is 9, sorted ascending by track Name

TrackGetAllDeepPurple – Composer contains “Jon Lord”, sorted ascending by TrackId

TrackGetAllTop100Longest – Sorted descending by Milliseconds; use the Take() method to limit the results to 100 items only

Data for the track entity looks like the following.



TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1	For Those About To Rock (We Salute You)	1	1	1	Angus Young, Malcolm Young, Brian Johnson	343,719	11,170,334	0.99
2	Balls to the Wall	2	2	1	NULL	342,562	5,510,424	0.99
3	Fast As a Shark	3	2	1	F. Baltes, S. Kaufman, U. Dirkscneider & W. Hoffman	230,619	3,990,994	0.99
4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Diesel, S. Kaufman, U. Dirkscneider & W. Hoffman	252,051	4,331,779	0.99
5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375,418	6,290,521	0.99
6	Put The Finger On You	1	1	1	Angus Young, Malcolm Young, Brian Johnson	205,662	6,713,451	0.99
7	Let's Get It Up	1	1	1	Angus Young, Malcolm Young, Brian Johnson	233,926	7,636,561	0.99
8	Inject The Venom	1	1	1	Angus Young, Malcolm Young, Brian Johnson	210,834	6,852,860	0.99
9	Snowballed	1	1	1	Angus Young, Malcolm Young, Brian Johnson	203,102	6,599,424	0.99
10	Evil Walks	1	1	1	Angus Young, Malcolm Young, Brian Johnson	263,497	8,611,245	0.99
11	C.O.D.	1	1	1	Angus Young, Malcolm Young, Brian Johnson	199,836	6,566,314	0.99
12	Breaking The Rules	1	1	1	Angus Young, Malcolm Young, Brian Johnson	263,288	8,596,840	0.99
13	Night Of The Long Knives	1	1	1	Angus Young, Malcolm Young, Brian Johnson	205,688	6,706,347	0.99
14	Spellbound	1	1	1	Angus Young, Malcolm Young, Brian Johnson	270,863	8,817,038	0.99
15	Go Down	4	1	1	AC/DC	331,180	10,847,611	0.99
16	Dog Eat Dog	4	1	1	AC/DC	215,196	7,032,162	0.99
17	Let There Be Rock	4	1	1	AC/DC	366,654	12,021,261	0.99
18	Bad Boy Boogie	4	1	1	AC/DC	267,728	8,776,140	0.99
19	Problem Child	4	1	1	AC/DC	325,041	10,617,116	0.99
20	Overdose	4	1	1	AC/DC	369,319	12,066,294	0.99
21	Hell Ain't A Bad Place To Be	4	1	1	AC/DC	254,380	8,331,286	0.99
22	Whole Lotta Rosie	4	1	1	AC/DC	323,761	10,547,154	0.99
23	Walk On Water	5	1	1	Steven Tyler, Joe Perry, Jack Blades, Tommy Shaw	295,680	9,719,579	0.99
24	Love In An Elevator	5	1	1	Steven Tyler, Joe Perry	321,828	10,552,051	0.99
25	Rag Doll	5	1	1	Steven Tyler, Joe Perry, Jim Vallance, Holly Knight	264,698	8,675,345	0.99
26	What It Takes	5	1	1	Steven Tyler, Joe Perry, Desmond Child	310,622	10,144,730	0.99
27	Dude (Looks Like A Lady)	5	1	1	Steven Tyler, Joe Perry, Desmond Child	264,855	8,679,940	0.99
28	Janie's Got A Gun	5	1	1	Steven Tyler, Tom Hamilton	330,736	10,869,391	0.99
29	Cryin'	5	1	1	Steven Tyler, Joe Perry, Taylor Rhodes	309,263	10,056,995	0.99
30	Amazing	5	1	1	Steven Tyler, Richie Supa	356,519	11,616,195	0.99
31	Blind Man	5	1	1	Steven Tyler, Joe Perry, Taylor Rhodes	240,718	7,877,453	0.99
32	Deuces Are Wild	5	1	1	Steven Tyler, Jim Vallance	215,875	7,074,167	0.99

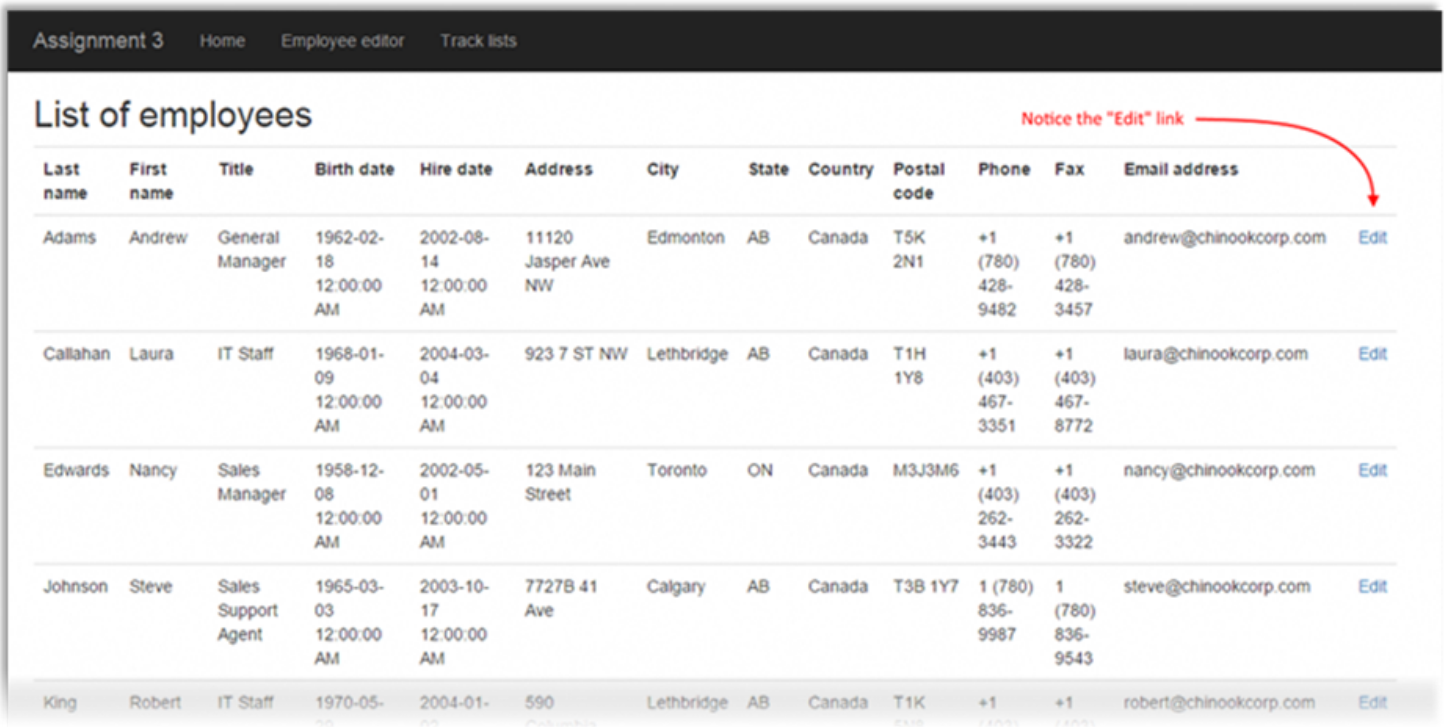
Add controllers, with code to work with the manager object

Two controllers are needed, right? One for Employees, and one for Tracks.

For the employee entity, implement the “get all” use case; including controller code, and view

The “get all” use case will show the default view when working with employee objects. Each employee will have an “Edit” link at the right side. This view will also be the destination after a successful “edit existing” task.

Here’s an example screen capture.



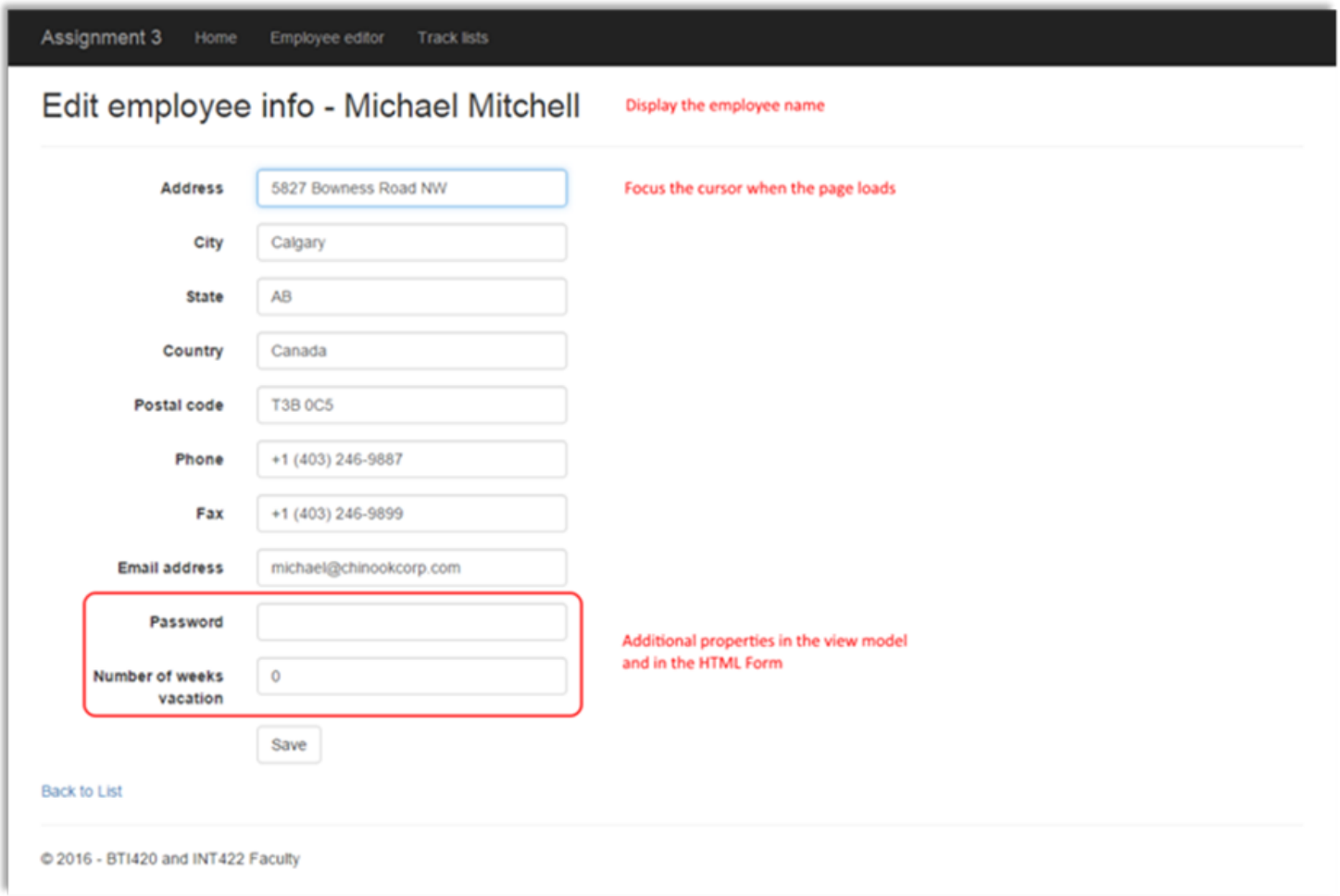
Implement the “edit existing” use case; including controller code, and view

As noted above, the class notes and code examples have all you need to implement this part of the work plan.

Don’t worry about the “extra” properties that are part of the object that is posted by the browser user from the HTML Form. They will not match any of the design model object properties, so they will be ignored.

The destination after a successful “edit existing” task must be a redirect-to-action to the Index action.

Here’s an example screen capture.

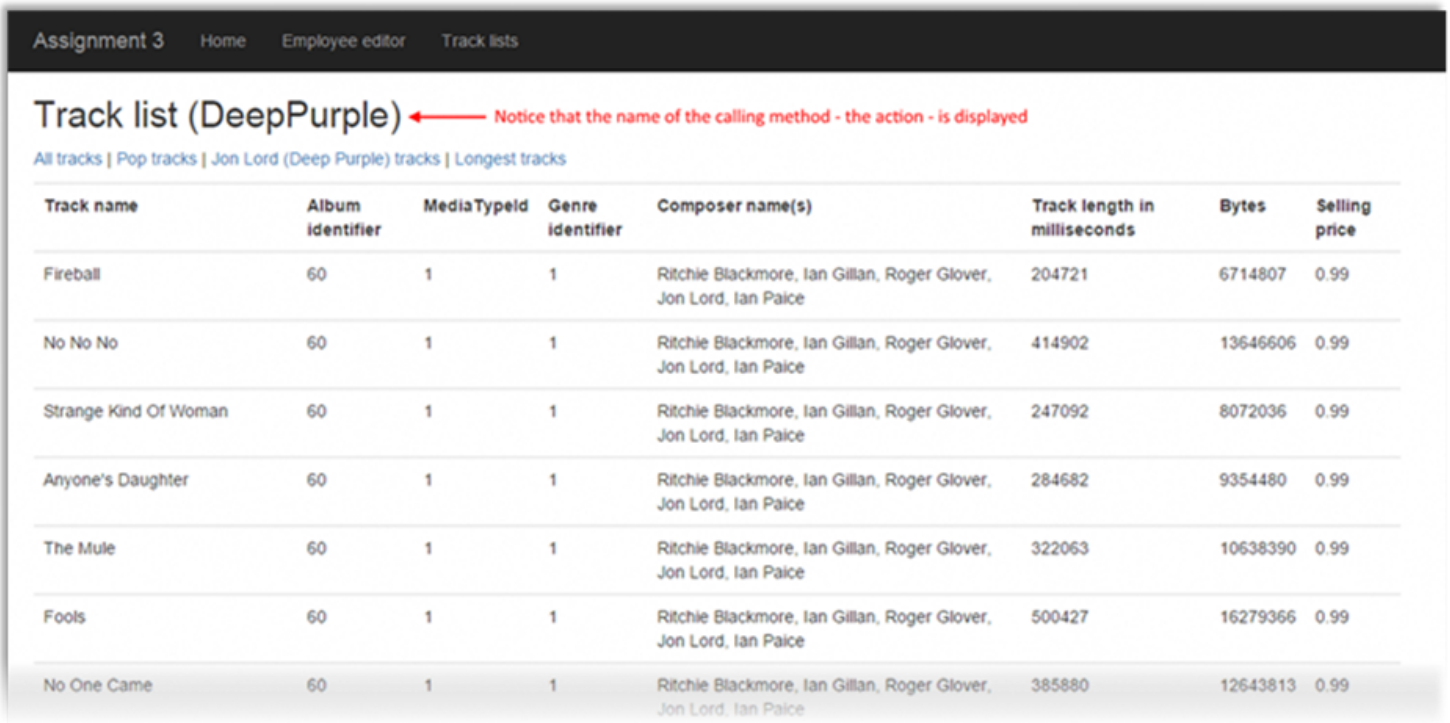


For the track entity, implement some “get all” filtered/sorted use cases; including controller code, and view

In this part of the assignment, you will get some experience with some basic LINQ functionality.

The list-of-tracks view – which by default, is scaffolded to the “Index” view – will host this functionality, and work with some new methods in the controller. This will be done by adding some links – as ActionLink HTML Helpers – near the top of the Index view.

Here’s an example screen capture.



The scaffolded TracksController class has an Index() method. It will call the typical “get all” method, and return in the typical manner.

Think carefully about the other method names, because they will become part of the URL. Don’t include the word “get” in the method name.

Each method will call a method in the manager object. Then, each method will return the Index view, and pass on the fetched collection.

Tip:

> If you want the name of the method (action) to display on the view, try the following. In the Index.cshtml view source code, edit the <h2> header element near the top, so that it looks like this:

```
<h2>Track list (@ViewContext.RouteData.GetRequiredString("action"))</h2>
```

Modifying the Index.cshtml view

As noted above, you will add some links – as ActionLink HTML Helpers – near the top of the Index view. Each link will call the appropriate action/method in the TracksController.

Make sure that you include a fourth link, which simply will call the Index action/method, to show all tracks.

Testing your work

In a browser, test your work, by doing tasks that fulfill the use cases in the specifications.

Reminder about academic honesty

You must comply with the College's academic honesty policy. Although you may interact and collaborate with others, *you must submit your own work*.

Important note

You MUST use the provided "Web app project v2" project template and AutoMapper instance API for your assignment. Fail to do so will result huge penalty for the assignment.

Submitting your work

Here's how to submit your work, before the due date and time:

1. Locate the folder that holds your solution files. In Solution Explorer, right-click the "Solution" item, and choose "Open Folder in File Explorer". It has three (or more) items: a Visual Studio Solution file, a folder that has your project's source code, and a "packages" folder. Go UP one level.

2. Make a copy of the folder. This is the version that you will be uploading.

3. Remove the "packages" folder from the copied folder; also, remove the "bin" and "obj" folders.

4. Compress/zip the copied folder. The zip file SHOULD be about 2MB or less in size. If it isn't, you haven't followed the instructions properly.

5. Login to My.Seneca/Blackboard. Open the Web Programming on Windows course area. Click the "Assignments" link on the left-side navigator. Follow the link for this lab. Submit/upload your zip file. The page will accept three submissions, so if you upload, then decide to fix something and upload again, you can do so.