INT422 Assignment 6

Working with a many-to-many association, “edit existing” use case.

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

Due date

Section A: Saturday, Oct 22, 2016, at 11:59**am** ET

Section B: Friday, Oct 21, 2016, at 11:59**am** 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)

Work with many-to-many associated data, with “edit existing” functionality. Your web app will enable users to work with Playlist objects.

Introduction to the problem to be solved

We need an app that will enable a browser user to view and edit playlists.

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 appropriate menu items
- Displays lists (“get all”) of Playlist objects
- Enables “edit existing” Playlist 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 initial use cases
4. Add methods to the Manager class that handle the use cases
5. Add controller(s), with code to work with the manager object
6. For the playlist entity, implement the “get all” and “get one” use cases; including controller code, and views

7. For the playlist entity, implement the “edit existing” use case; 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.

Create the project, based on the project template

Create a new web app, named Assignment6.

It MUST use the “Web app project v2” project template. Download this new project template from the course website, and install it into your Visual Studio configuration.

Warning: Your teachers believe that the best way to work through this assignment is to do incrementally. Get one thing working, before moving on to the next. Test each part.

Customize the app’s appearance

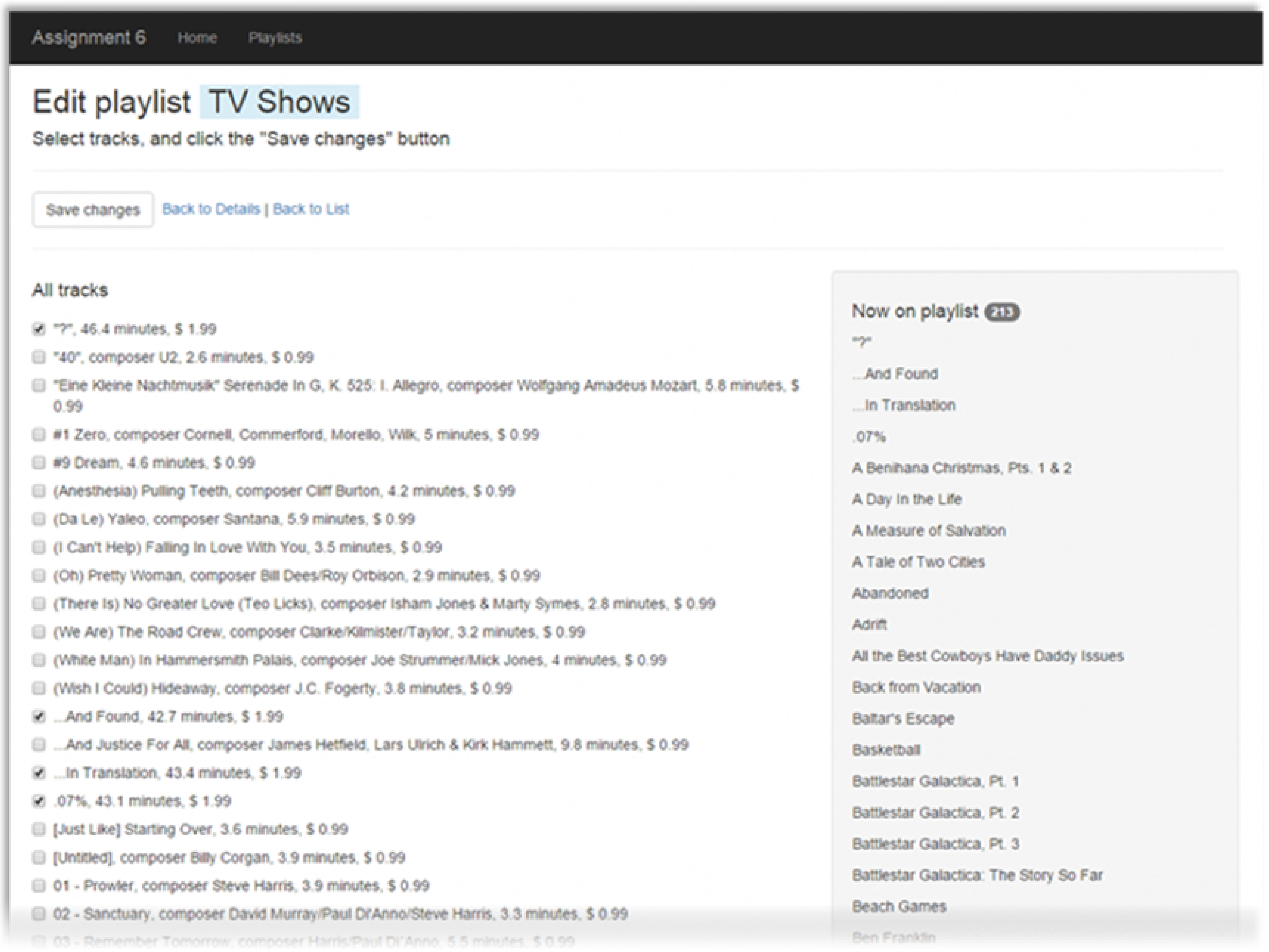
You will customize the appearance all of your web apps and assignments. Never submit an assignment that has the generic auto-generated text content. Make the time to customize the web app’s appearance.

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.

Overall goal: Enable basic editing

The overall goal is to enable basic editing of a playlist. The following image shows what you may end up with at the end of this assignment. Click to open the image full-size in a new tab/window.



Edit playlist TV Shows

Select tracks, and click the "Save changes" button

All tracks

- [✓] "?", 46.4 minutes, $ 1.99
- [ ] "40", composer U2, 2.6 minutes, $ 0.99
- [ ] "Eine Kleine Nachtmusik" Serenade In G, K. 525: I. Allegro, composer Wolfgang Amadeus Mozart, 5.8 minutes, $ 0.99
- [ ] #1 Zero, composer Cornell, Commerford, Morello, Wilk, 5 minutes, $ 0.99
- [ ] #9 Dream, 4.6 minutes, $ 0.99
- [ ] (Anesthesia) Pulling Teeth, composer Cliff Burton, 4.2 minutes, $ 0.99
- [ ] (Da Le) Yaleo, composer Santana, 5.9 minutes, $ 0.99
- [ ] (I Can't Help) Falling In Love With You, 3.5 minutes, $ 0.99
- [ ] (Oh) Pretty Woman, composer Bill Dees/Roy Orbison, 2.9 minutes, $ 0.99
- [ ] (There Is) No Greater Love (Teo Licks), composer Isham Jones & Marty Symes, 2.8 minutes, $ 0.99
- [ ] (We Are) The Road Crew, composer Clarke/Kilmister/Taylor, 3.2 minutes, $ 0.99
- [ ] (White Man) In Hammersmith Palais, composer Joe Strummer/Mick Jones, 4 minutes, $ 0.99
- [ ] (Wish I Could) Hideaway, composer J.C. Fogerty, 3.8 minutes, $ 0.99
- [✓] ...And Found, 42.7 minutes, $ 1.99
- [ ] ...And Justice For All, composer James Hetfield, Lars Ulrich & Kirk Hammett, 9.8 minutes, $ 0.99
- [✓] ...In Translation, 43.4 minutes, $ 1.99
- [✓] .07%, 43.1 minutes, $ 1.99
- [ ] [Just Like] Starting Over, 3.6 minutes, $ 0.99
- [ ] [Untitled], composer Billy Corgan, 3.9 minutes, $ 0.99
- [ ] 01 - Prowler, composer Steve Harris, 3.9 minutes, $ 0.99
- [ ] 02 - Sanctuary, composer David Murray/Paul Di'Anno/Steve Harris, 3.3 minutes, $ 0.99
- [ ] 03 - Remember Tomorrow, composer Harris/Paul Di'Anno, 5.5 minutes, $ 0.99

Now on playlist 213

"?"
...And Found
...In Translation
.07%
A Benihana Christmas, Pts. 1 & 2
A Day In the Life
A Measure of Salvation
A Tale of Two Cities
Abandoned
Adrift
All the Best Cowboys Have Daddy Issues
Back from Vacation
Baltar's Escape
Basketball
Battlestar Galactica, Pt. 1
Battlestar Galactica, Pt. 2
Battlestar Galactica, Pt. 3
Battlestar Galactica: The Story So Far
Beach Games
Ben Franklin

Study the image. It shows a list of existing tracks on the playlist on the right side of the page. On the left, it shows a list of ALL tracks, whether on the playlist or not. The browser user can use the checkboxes to edit the content of the playlist.

To render this view, it will need one select list object, and a collection:

- The select list will be used to render the checkboxes
- The collection will hold the track objects on the existing playlist

Create view models and mappers that cover the initial use cases

The most logical starting point for the app will be to display a list of playlist objects.

Therefore, you will need a PlaylistBase view model class.

Next, you will need another view model class to hold details, and the most important details is a collection of track objects.

This in turn implies that you will need a TrackBase class.

The “edit existing” task will need another two view model classes:

1. A view model class named (maybe) PlaylistEditTracksForm, to hold the data that is needed to render the HTML Form
2. A view model class named (maybe) PlaylistEditTracks, to hold the data submitted by the browser user

As noted in the previous section, the “…Form” view model class will yes, need a select list property (for multiple selection). It will *also* need another property, to hold the collection of tracks that are on the current playlist.

As you have learned, the “…Form” view model class needs identification information. The object identifier, for sure, and obviously. And text to display to the browser user.

Remember, you will also need mappers.

Add methods to the Manager class that handle the use cases

Methods to handle “get all” playlists, and “get one” playlist, are needed.

As you have seen with other associated-item scenarios, the “get one” method will fetch the associated object(s). Therefore, “get one” playlist must fetch its track objects.

In addition, the “edit existing” playlist method will be needed. It will use the standard pattern for editing a to-many association of objects, as seen in the *AssocManyToMany* code example, and in the recent class notes.

Now, stop for a moment, and think about the “edit playlist” task. Presumably, you will create a page that will show or identify the playlist-being-edited. To edit/select the tracks on the playlist, the page will also have to *show a list of tracks*, in an item-selection element.

Therefore, in the Manager class, you will also need a “get all” tracks method.

Please fix the Manager class constructor

Please add the following highlighted code to the Manager class constructor, to fix a performance problem. Your teacher team decided that we would not revise the web app project template, which would have been another way to fix the problem. Instead, please fix it yourself. Thanks.

```
public Manager()
{
  // Turn off the Entity Framework (EF) proxy creation features
  // We do NOT want the EF to track changes - we'll do that ourselves
  ds.Configuration.ProxyCreationEnabled = false;

  // Also, turn off lazy loading...
  // We want to retain control over fetching related objects
  ds.Configuration.LazyLoadingEnabled = false;
}
```

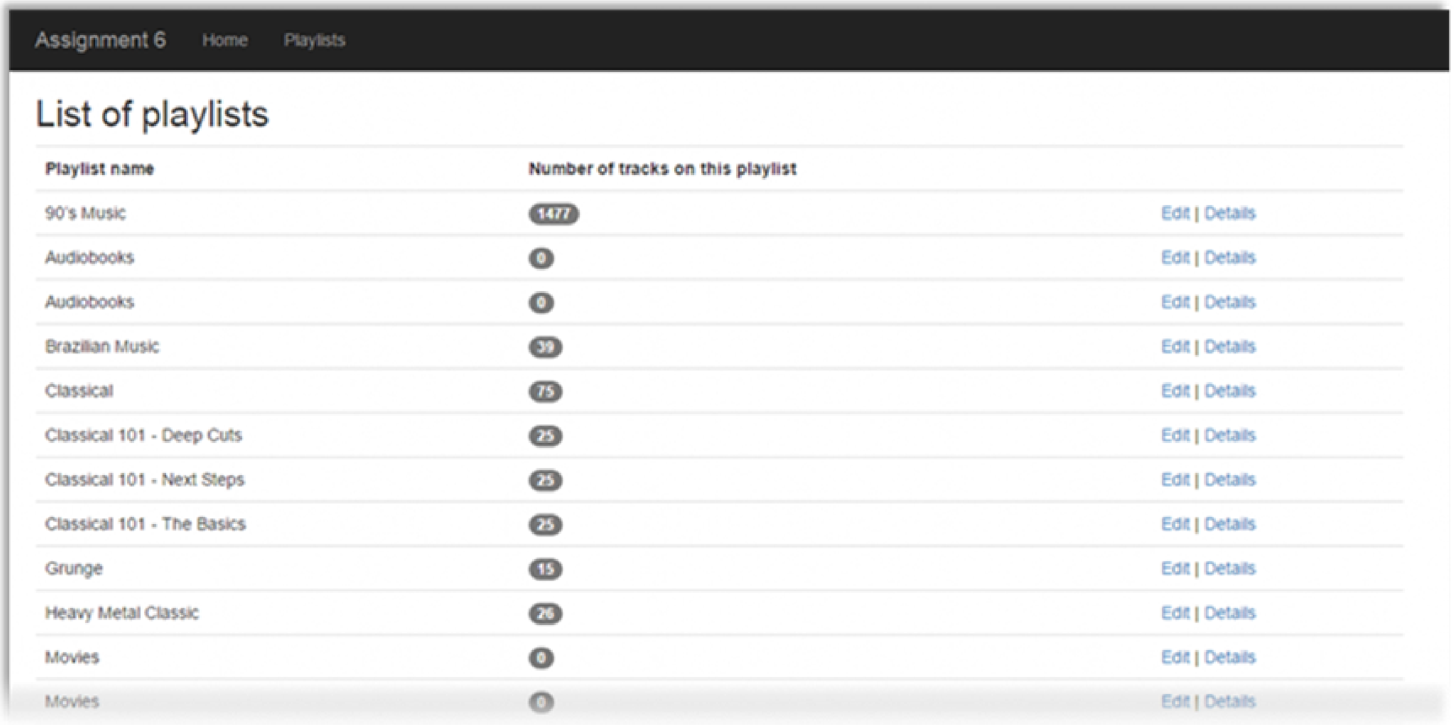
Add controller(s), with code to work with the manager object

Add a controller for the playlist entity.

For the playlist entity, implement the “get all” and “get one” use cases; including controller code, and views

In the playlist controller, add the standard methods for “get all” and “get one”.

Generate the “get all” view. It should look something like the following. Click the image to open it in a new tab/window.



List of playlists

Playlist name	Number of tracks on this playlist	
90's Music	1477	Edit | Details
Audiobooks	0	Edit | Details
Audiobooks	0	Edit | Details
Brazilian Music	39	Edit | Details
Classical	75	Edit | Details
Classical 101 - Deep Cuts	25	Edit | Details
Classical 101 - Next Steps	25	Edit | Details
Classical 101 - The Basics	25	Edit | Details
Grunge	15	Edit | Details
Heavy Metal Classic	26	Edit | Details
Movies	0	Edit | Details
Movies	0	Edit | Details

❝How did the middle column – Number of tracks on this playlist – get generated?

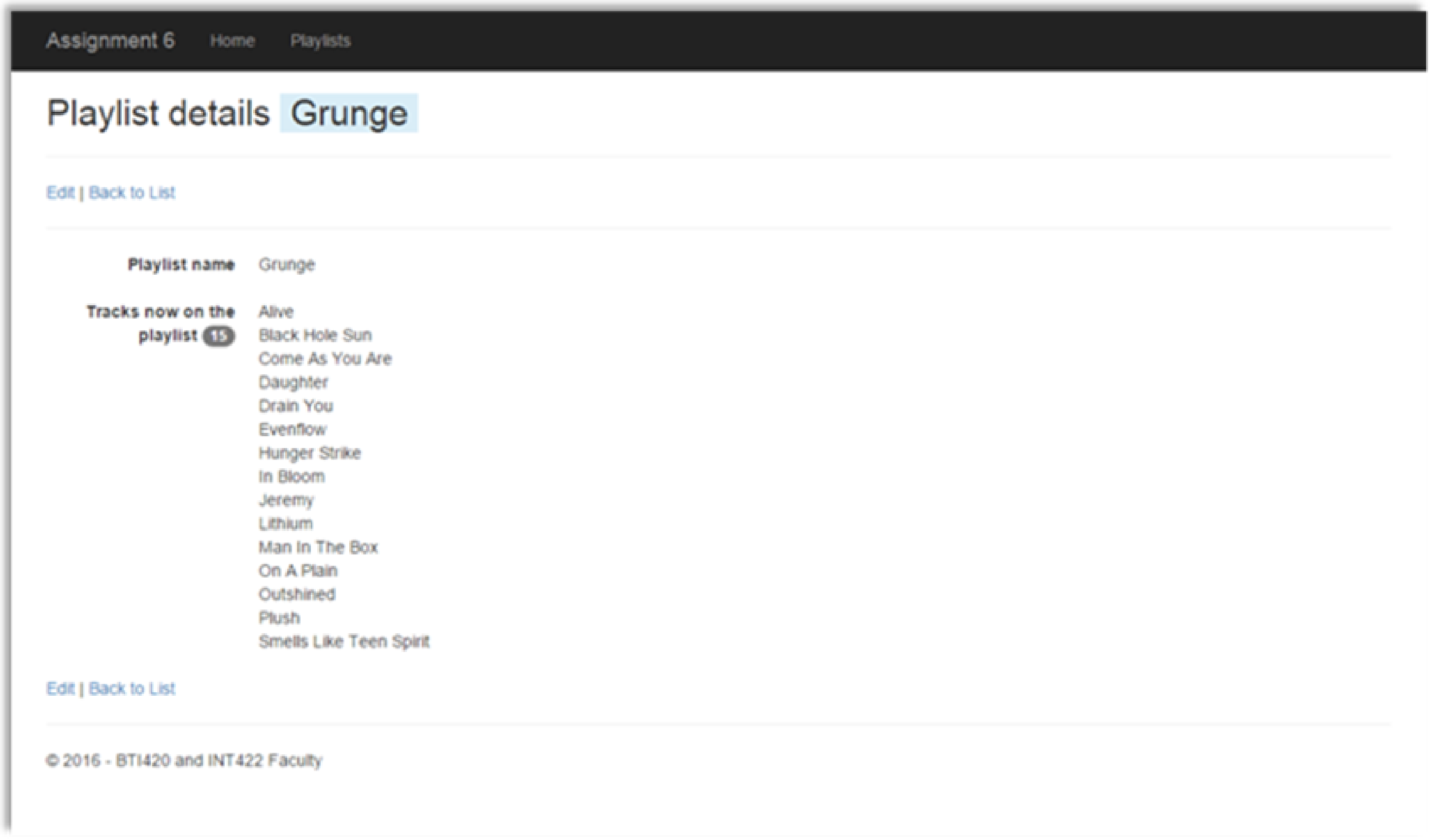
Two changes – one in the view model class, and one in the Manager class.

In the PlaylistBase class, add an int property named “TracksCount”. This name is important. It is a *composed name*. For a specific playlist object, which has a navigation property named “Tracks”, it will fetch/render the “Count” property of the “Tracks” object. Wait – there’s no property named “Count” in the design model class – where is that defined? Well, “Count” is a property of the collection.

Then, in the Manager class, edit the “get all” method, so that it includes the “Tracks” property in the fetch.

Finally, render this column in the view. If you want the number to show up as a Bootstrap badge, you can do that too.

Generate the “get one” view. It will need hand-editing, to render the collection of Track objects (wrapped in <p> or <span> elements, your choice). It should look something like the following. Click the image to open it in a new tab/window.



Playlist details Grunge

Edit | Back to List

Playlist name Grunge

Tracks now on the playlist 15
Alive
Black Hole Sun
Come As You Are
Daughter
Drain You
Evenflow
Hunger Strike
In Bloom
Jeremy
Lithium
Man In The Box
On A Plain
Outshined
Plush
Smells Like Teen Spirit

Edit | Back to List

© 2016 - BTI420 and INT422 Faculty

For the playlist entity, implement the "edit existing" use case; including controller code, and view

Now, you're ready to implement the "edit existing" use case. If you wish, you can use the pair of method stubs in the controller that the scaffolder created for the edit task, to get started. Use the familiar pattern to code the rest of the methods.

You should know by now that the purpose of the GET method is to prepare and configure the "…Form" view model object, and send it to the view, so that an HTML Form can be created.

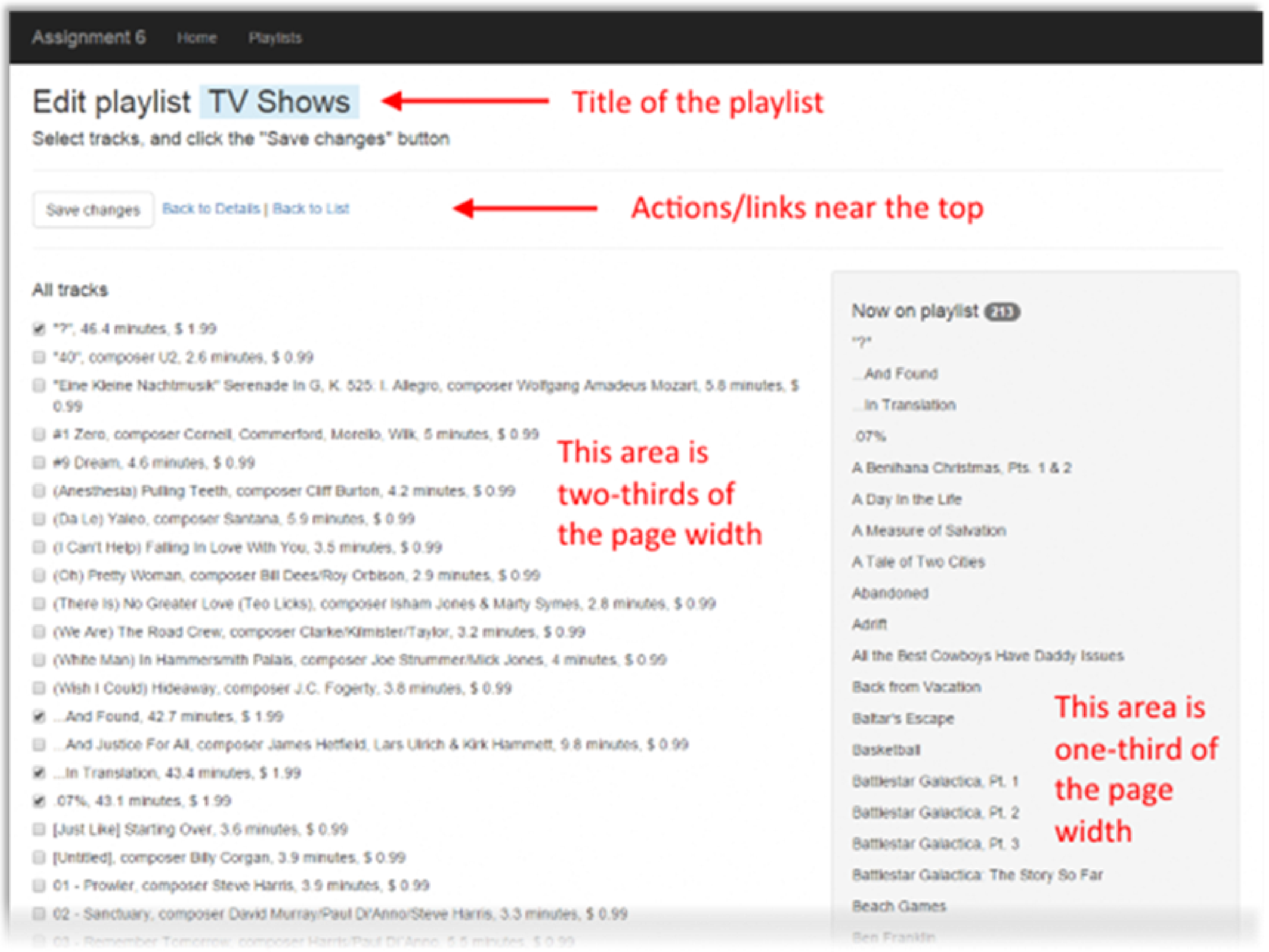
Create a "…Form" object. Configure it. In addition to identification property values, it will need:

- A select list object, configured with 1) all possible tracks, and 2) the identifiers for the tracks that are currently associated with the playlist
- A TrackBase collection, configured with the tracks that are currently associated with the playlist

Generate and edit the view

Generate the view. As you have learned, the scaffolder does not render collections or item-selection elements.

At the end of this task, the plan is to end up with a view that was first shown above. Here's how. First, notice how the page is designed:


Assignment 6
Home
Playlists
Title of the playlist
Actions/links near the top
This area is two-thirds of the page width
This area is one-third of the page width
Save changes

Edit playlist TV Shows

Select tracks, and click the "Save changes" button

Back to Details | Back to List

All tracks

- ☑ "?", 46.4 minutes, $ 1.99
- ☐ "40", composer U2, 2.6 minutes, $ 0.99
- ☐ "Eine Kleine Nachtmusik" Serenade In G, K. 525: I. Allegro, composer Wolfgang Amadeus Mozart, 5.8 minutes, $ 0.99
- ☐ #1 Zero, composer Cornell, Commerford, Morello, Wilk, 5 minutes, $ 0.99
- ☐ #9 Dream, 4.6 minutes, $ 0.99
- ☐ (Anesthesia) Pulling Teeth, composer Cliff Burton, 4.2 minutes, $ 0.99
- ☐ (Da Le) Yaleo, composer Santana, 5.9 minutes, $ 0.99
- ☐ (I Can't Help) Falling In Love With You, 3.5 minutes, $ 0.99
- ☐ (Oh) Pretty Woman, composer Bill Dees/Roy Orbison, 2.9 minutes, $ 0.99
- ☐ (There Is) No Greater Love (Teo Licks), composer Isham Jones & Marty Symes, 2.8 minutes, $ 0.99
- ☐ (We Are) The Road Crew, composer Clarke/Kilmister/Taylor, 3.2 minutes, $ 0.99
- ☐ (White Man) In Hammersmith Palais, composer Joe Strummer/Mick Jones, 4 minutes, $ 0.99
- ☐ (Wish I Could) Hideaway, composer J.C. Fogerty, 3.8 minutes, $ 0.99
- ☑ ...And Found, 42.7 minutes, $ 1.99
- ☐ ...And Justice For All, composer James Hetfield, Lars Ulrich & Kirk Hammett, 9.8 minutes, $ 0.99
- ☑ ...In Translation, 43.4 minutes, $ 1.99
- ☑ .07%, 43.1 minutes, $ 1.99
- ☐ [Just Like] Starting Over, 3.6 minutes, $ 0.99
- ☐ [Untitled], composer Billy Corgan, 3.9 minutes, $ 0.99
- ☐ 01 - Prowler, composer Steve Harris, 3.9 minutes, $ 0.99
- ☐ 02 - Sanctuary, composer David Murray/Paul Di'Anno/Steve Harris, 3.3 minutes, $ 0.99
- ☐ 03 - Remember Tomorrow, composer Harris/Paul Di'Anno, 5.5 minutes, $ 0.99

Now on playlist 213

"?"
...And Found
...In Translation
.07%
A Benihana Christmas, Pts. 1 & 2
A Day In the Life
A Measure of Salvation
A Tale of Two Cities
Abandoned
Adrift
All the Best Cowboys Have Daddy Issues
Back from Vacation
Baltar's Escape
Basketball
Battlestar Galactica, Pt. 1
Battlestar Galactica, Pt. 2
Battlestar Galactica, Pt. 3
Battlestar Galactica: The Story So Far
Beach Games
Ben Franklin

If you want the playlist name to appear as you see it above (with a light blue background), then surround it with a <span> element that uses one of the Bootstrap contextual backgrounds.

How do we get a two-column layout? With the Bootstrap *.row* and *.col-xx-xx classes*. Here's how to make the container:

1. Add a <div> element with class=row
2. Inside this container, add two more <div> elements
3. The left side will take two-thirds of the width, or 8/12 of the grid system layout, therefore, add class=col-md-8 to the first (left-side) container
4. Then, add class=col-md-4 to the second (right-side) container

In the left-side container, add code for the checkbox group that displays all possible tracks. (The image is showing the result of standard <input type=checkbox… elements.)

How did we render the track name, composer, track time/length, and unit price? By adding a read-only property (which is a property that does not have a "set" accessor), and then using that in the select list object's constructor. Here's an example of how you can add a read-only property in a view model class:

```
// Composed read-only property
public string NameFull
{
  get
  {
    var ms = Math.Round((((double)Milliseconds / 1000) / 60), 1);

    var composer = string.IsNullOrEmpty(Composer) ? "" : ", composer " +
Composer;
    var trackLength = (ms > 0) ? ", " + ms.ToString() + " minutes" : "";
    var unitPrice = (UnitPrice > 0) ? ", $ " + UnitPrice.ToString() : "";

    return string.Format("{0}{1}{2}{3}", Name, composer, trackLength,
unitPrice);
  }
}
```

The right-side container will have code for the <p> elements that hold the track names now on the playlist.

If you want the right-side container to have a grey background with rounded corners, simply add one of the Bootstrap well classes.

Code the method that processes the data submitted by the browser user

The incoming data will include the playlist identifier, and an int collection, with the identifiers of the tracks that will be on the playlist. Pass them on to the manager method. A successful result will redirect to the "details" view.

Testing your work

While designing and coding your web app, use the Visual Studio debugger to test your algorithms, and inspect the data that you are working with.

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.