INT422 Assignment 9

Continue using security principles. Rich text handling. Working with non-text media types.

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

Due date

Section A & B: ~~Wednesday, April 5~~ April 6, 2017, at 11:59pm ET

Section C: ~~Thursday, April 6~~ April 7, 2017, at 11:59pm ET

Grade value: 9% of your final course grade

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

Objective(s)

There are several objectives for this Assignment 9:

- Continue using security principles
- Working with internet media types
- Rich text handling
- Post to a public web server

Introduction and overview

We continue working with the *music business* problem domain in this assignment.

The web app needs to be enhanced, to support security concepts, as well as richer data. The richer data will include rich text (i.e. HTML markup), and internet media types (e.g. image, audio, video).

The teacher team has posted a working sample solution to Azure:

https://wsong18-wa2017a9.azurewebsites.net/

It has the functionality needed for the interim/progress submission on Thursday, Mar 30 for section A & B or Friday, Mar 31 for section C. Then, it will be updated to include the media handling capabilities.

Tasks that should be done for the interim/progress submission

You will be able to do these tasks now:

1. Modify the data model, to add a class for role claims
2. Modify the Register() methods in the Account controller to use the new class
3. Modify the data model, to accommodate lengthy text fields in the Artist and Album entity classes
4. Implement "add new" for these entity classes, to support rich text entry
5. Implement "get…" views to display objects that have rich text

Tasks that can be done after the interim/progress submission

Then, after the remaining topics are introduced, you will be able to do these tasks:

1. Modify the data model, to accommodate non-text media types in the Artist and Track entity classes
2. Support add and display of the richer-looking objects

Please note that you will also implement security-related content while implementing these tasks. This includes 1) protection of some actions to specific users/groups, and 2) conditionally rendering content.

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
- Improve the security claims configuration
- Support rich text entry, editing, and display
- Support some data service operations (add, deliver) for non-text media types

- Publish the app to the web

Here is a brief work plan sequence:

1. Create the project
2. Customize the app's appearance
3. Create the design model classes
4. Create view models and mappers that cover the initial use cases
5. Configure the security settings for the app
6. Add methods to the Manager class that handle the use cases, and load initial data
7. Implement rich text editing, and media item handling
8. Publish to Azure

Create the project, based on the project template

Create a new web app, named **Assignment9**. It MUST use the new "**Web app project A9**" project template. Get this new project template from the course website, and install it into your Visual Studio configuration.

After creating the web app, customize the home page. Change the large "**Learn more >>**" button to **"Assignment 9 on Azure**" and set the button link to the URL of the assignment 9 on Azure.

Build/compile the app, to refresh the packages. Do NOT run the app yet.

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.

Fix these problems in the template

Unintentionally, the project template for Assignment 9 has an error, and a setting that can cause problems in some situations. Fix them before continuing.

Attribute routing fix

Open the App_Start > RouteConfig.cs source code file.

Move the statement that activates attribute routing up, so that it is *before* the "routes.MapRoute…" statement.

Temporarily disable error-handling for HTTP errors 500 and higher

Open the Global.asax.cs source code file.

In the Application_EndRequest() method, comment out the “if (code >= 500)…” statement.

Partial view Upload.cshtml

Go to the Shared folder. You may need to add the folder and file **EditorTempletes/Upload.cshtml** if they do not exist.

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.

Improve the role claims configuration

As you know, the Register() methods in the Account controller have hard-coded role claim values.

Not good, for an in-production real-world app.

Therefore, we will fix that now.

In the DesignModelClasses.cs source code file, create a **RoleClaim** class, with an identifier property, and a string property “Name” (required, string length 100). At a minimum, it needs these properties. You can add more, if you wish.

Add a DbSet<TEntity> property to the DataContext class.

In the Manager class, use the LoadData() method to create and save some new role claims. Let’s continue to use the idea that we had in Assignment 8, with role claims for

Executive, **Coordinator**, **Clerk**, and **Staff**. Then, add two others, and you decide what they will be. Therefore, there will be **six (6) role claims** in total.

Also in the Manager class, add a **RoleClaimGetAllStrings**() method that returns an ordered collection of strings from the RoleClaim entity collection. It will be used by the Register() method in the Account controller to get data it needs for the user interface.

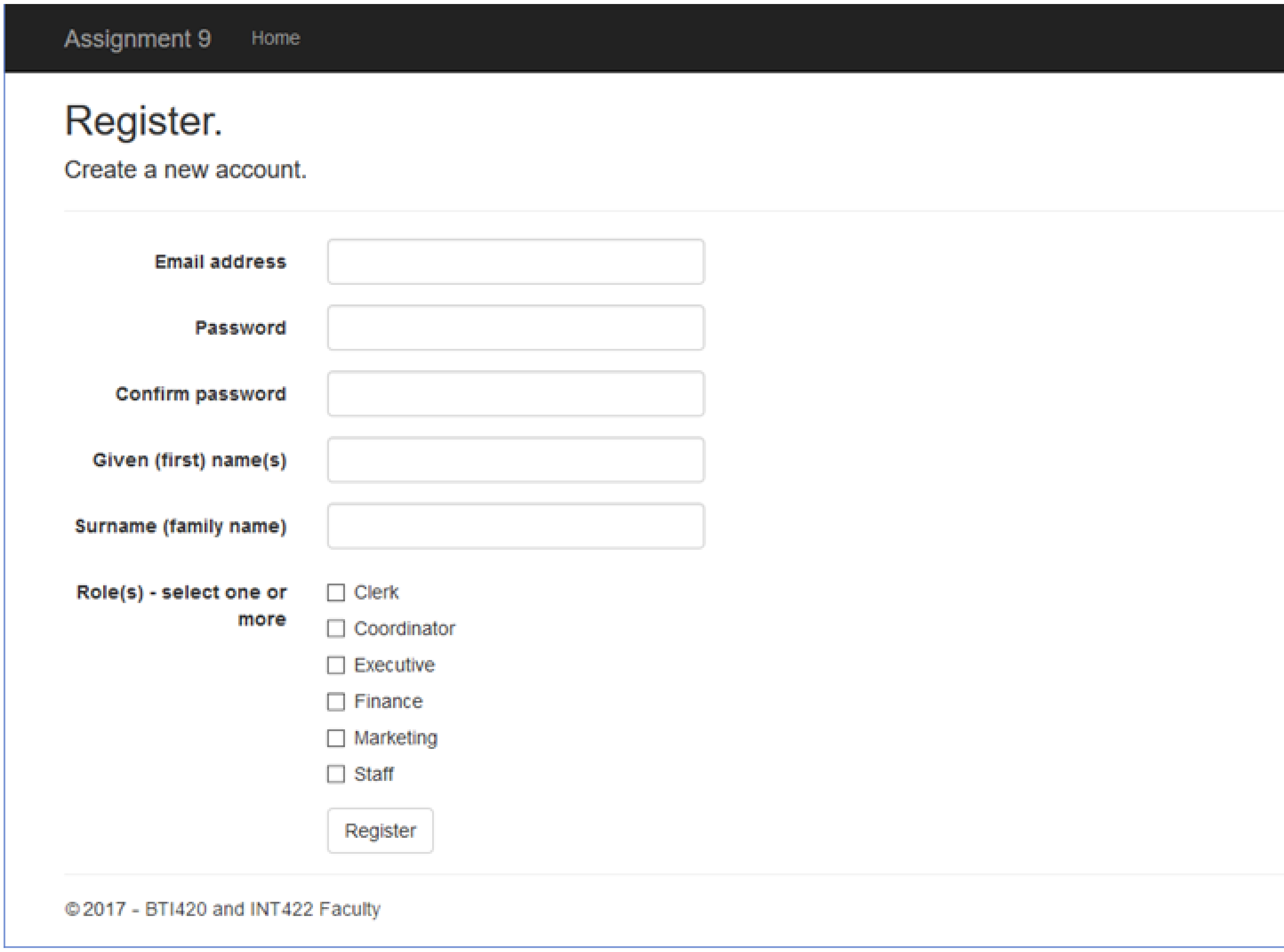
Like you probably did before, use the LoadData controller, and call the manager object's LoadData() method.

Now, edit the Account controller. In both Register() methods, you must change all the code that references the hard-coded role claim collection, so that it fetches and uses the role claim string collection from the manager object.

Test your work

Test your work by attempting to register a new user. Before you do that, you must run the load data task (to load the role claims created above). The new role claim list should appear on the "register" page, and enable the correct configuration of a new user.

Click the image to open it full-size in a new tab/window.



Register.

Create a new account.

Email address	
Password	
Confirm password	
Given (first) name(s)	
Surname (family name)	
Role(s) - select one or more	☐ Clerk ☐ Coordinator ☐ Executive ☐ Finance ☐ Marketing ☐ Staff

© 2017 - BTI420 and INT422 Faculty

Data storage preparation tasks

Next, enable migrations. That will create a starting point (snapshot) of the database structure.

Prepare for security-aware content

We will continue to use the user account security ideas that were implemented in Assignment 8. In other words:

- A user with an Executive role claim is allowed to add, edit, and delete an artist
- A user with a Coordinator role claim is allowed to add, edit, and delete an album

- A user with a Clerk role claim is allowed to add, edit, and delete a track
- All other kinds of authenticated users can read artist, album, and track data
- Anonymous users cannot work with artist, album, and track data

Add these kinds of users to your app, using your updated Register page.

Recently, you learned how to conditionally enable functionality, based on information in the security principal (e.g. authenticated or not, user info, role claim info). Content containers could be visible, or not, depending upon the user.

In this assignment, you have the opportunity to implement these ideas. In general:

- Menu items – whether individual or dropdown – should appear (or not), based on the user
- Content containers – including static text and data from the model, and links/buttons – should appear (or not), based on the user

As you make progress when coding a feature, pause for a moment, and ask whether security influences its presence in the user interface. Use common sense. If you’re not sure, ask a friend who knows, or ask your professor.

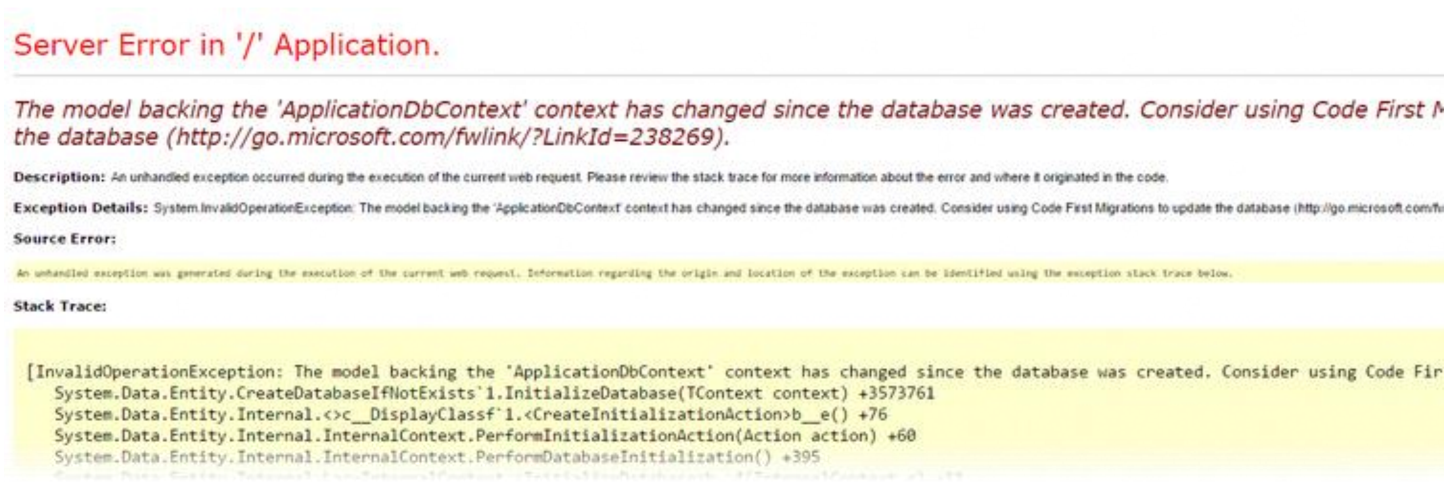
Preparing for rich text editing

The Artist and Album design model classes need another string property, to hold rich text.

In the **Artist** class, add a new string property named “**Portrayal**”. Not required, and lengthy. Its intention is to capture content about the history, biographical data, and musical style of the artist.

In the **Album** class, add a new string property named “**Depiction**”. Not required, and lengthy. Its intention is to capture content about the theme, style, content, and assembly of the album.

Build/compile the app, and run/load it in a browser. Uh oh, an error:



```
Server Error in '/' Application.

The model backing the 'ApplicationDbContext' context has changed since the database was created. Consider using Code First M
the database (http://go.microsoft.com/fwlink/?LinkId=238269).

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.InvalidOperationException: The model backing the 'ApplicationDbContext' context has changed since the database was created. Consider using Code First Migrations to update the database (http://go.microsoft.com/fw

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

[InvalidOperationException: The model backing the 'ApplicationDbContext' context has changed since the database was created. Consider using Code Fir
   System.Data.Entity.CreateDatabaseIfNotExists`1.InitializeDatabase(TContext context) +3573761
   System.Data.Entity.Internal.<>c__DisplayClassf`1.<CreateInitializationAction>b__e() +76
   System.Data.Entity.Internal.InternalContext.PerformInitializationAction(Action action) +60
   System.Data.Entity.Internal.InternalContext.PerformDatabaseInitialization() +395
```

As it states, the data model has changed.

To fix this, review the coverage of the Migrations feature in the week 8 Lecture Notes:

1. add-migration *descriptive-name-for-migration*
2. update-database

Later, remember this “recipe” for enabling rich text editing and display:

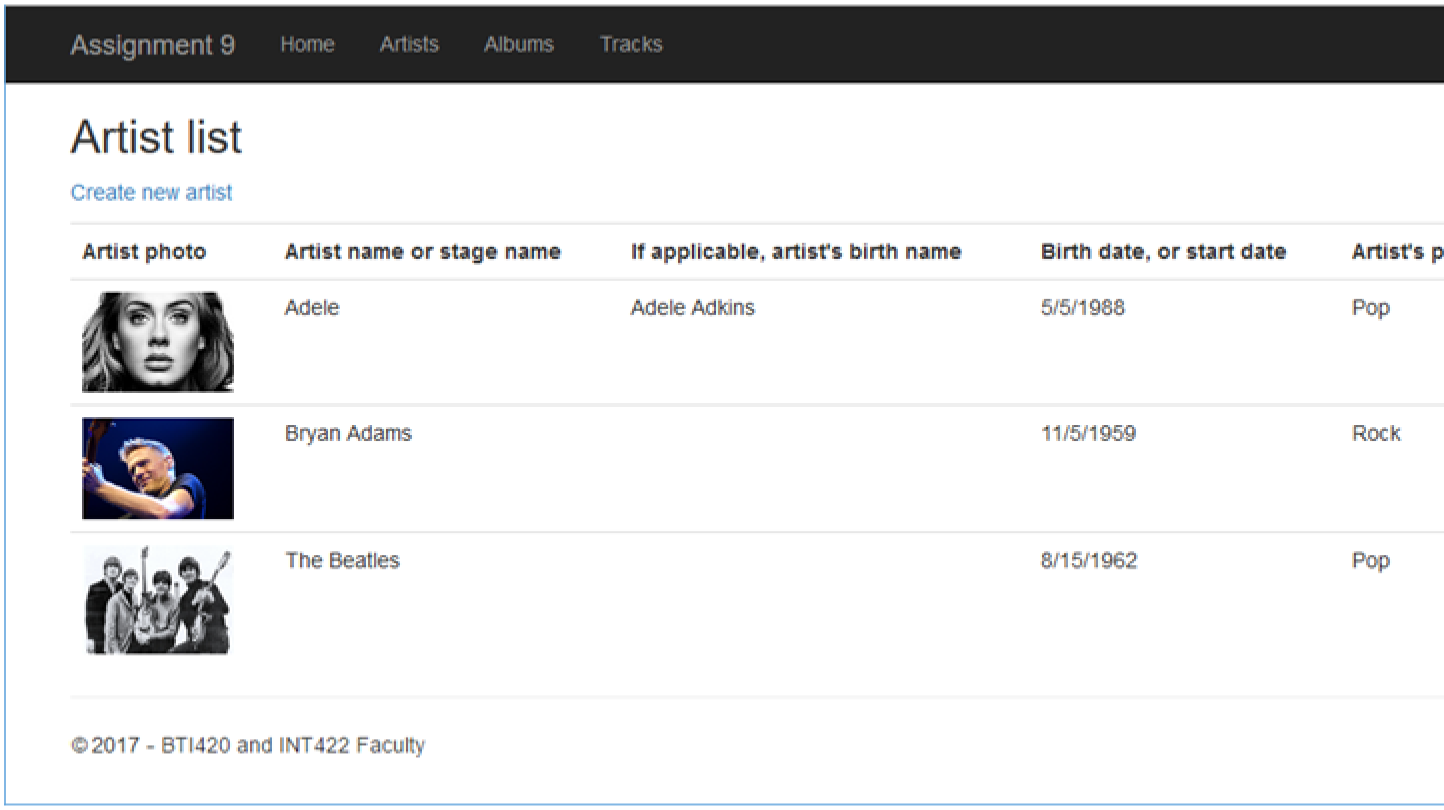
1. Add **[DataType(DataType.MultiLineText)]** to the string property in the view model class
2. In the view, before the code that renders the string property, add a reference to the rich text editor app
3. After the code that renders the string property, run the command to convert the <textarea> into a rich text editor rectangle
4. In the controller method that handles the data submitted by the user (the POST method), add **[ValidateInput(false)]**
5. When displaying rich text, use the **Raw()** HTML Helper

Implement “get all” for Artist and Album

We want to be able to get to “add new” Artist and Album as soon as we can. Before doing that, it makes sense to implement “get all” for them, so that you have views that shows lists of existing artists or albums.

Use your memory, the class notes, or the patterns summary document that you were supposed to create (and submit to your teacher) two months ago (and documented on the week 8 Lecture Notes page), and implement this functionality.

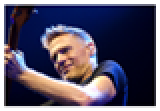
Maybe your artist list looks like the following. Click the image to open it full-size in a new tab/window.



Assignment 9 Home Artists Albums Tracks

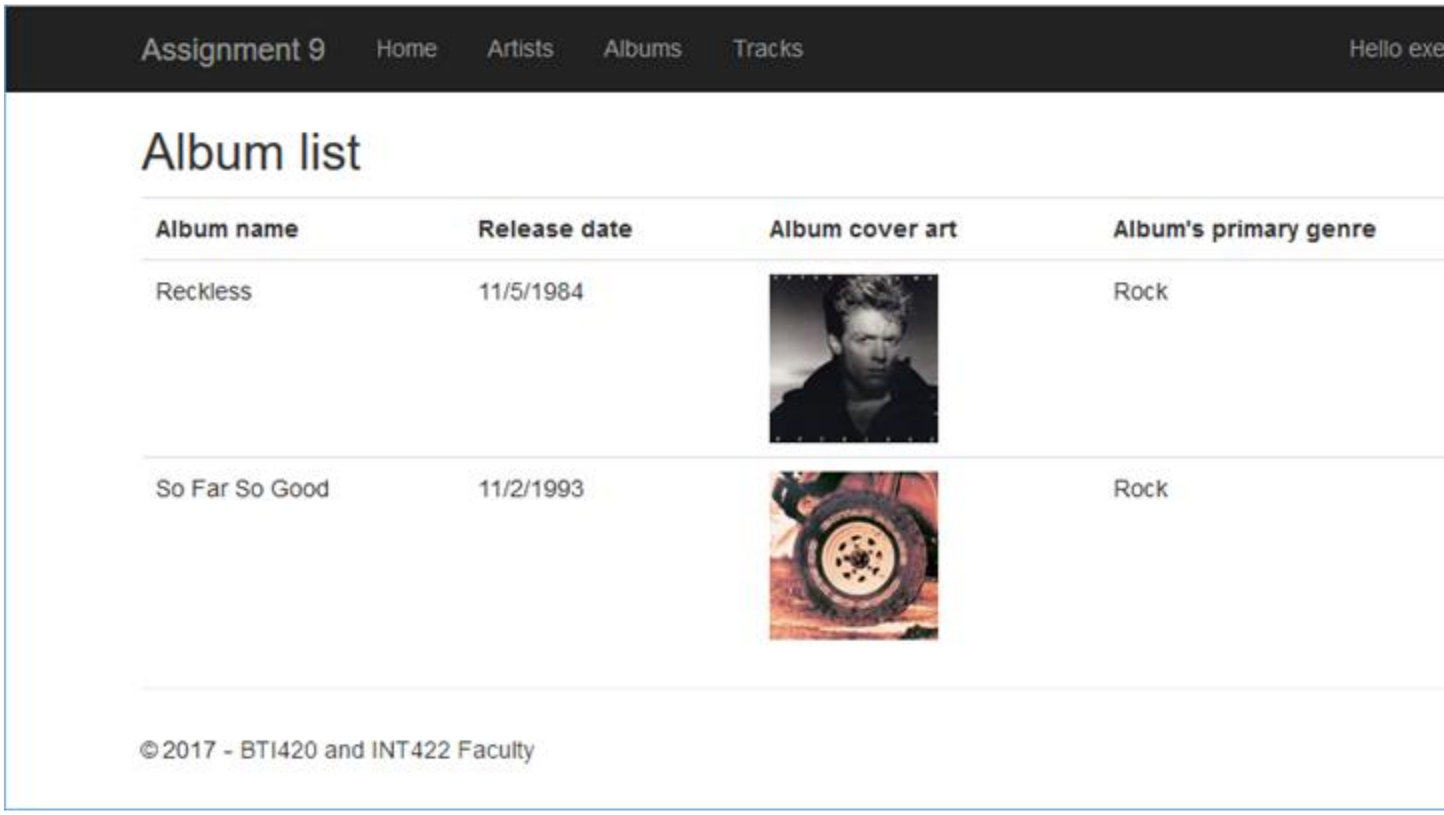
Artist list

Create new artist

Artist photo	Artist name or stage name	If applicable, artist's birth name	Birth date, or start date	Artist's p
	Adele	Adele Adkins	5/5/1988	Pop
	Bryan Adams		11/5/1959	Rock
	The Beatles		8/15/1962	Pop

©2017 - BTI420 and INT422 Faculty

Maybe your album list looks like the following. Click the image to open it full-size in a new tab/window.



Album list

Album name	Release date	Album cover art	Album's primary genre
Reckless	11/5/1984		Rock
So Far So Good	11/2/1993		Rock

© 2017 - BTI420 and INT422 Faculty

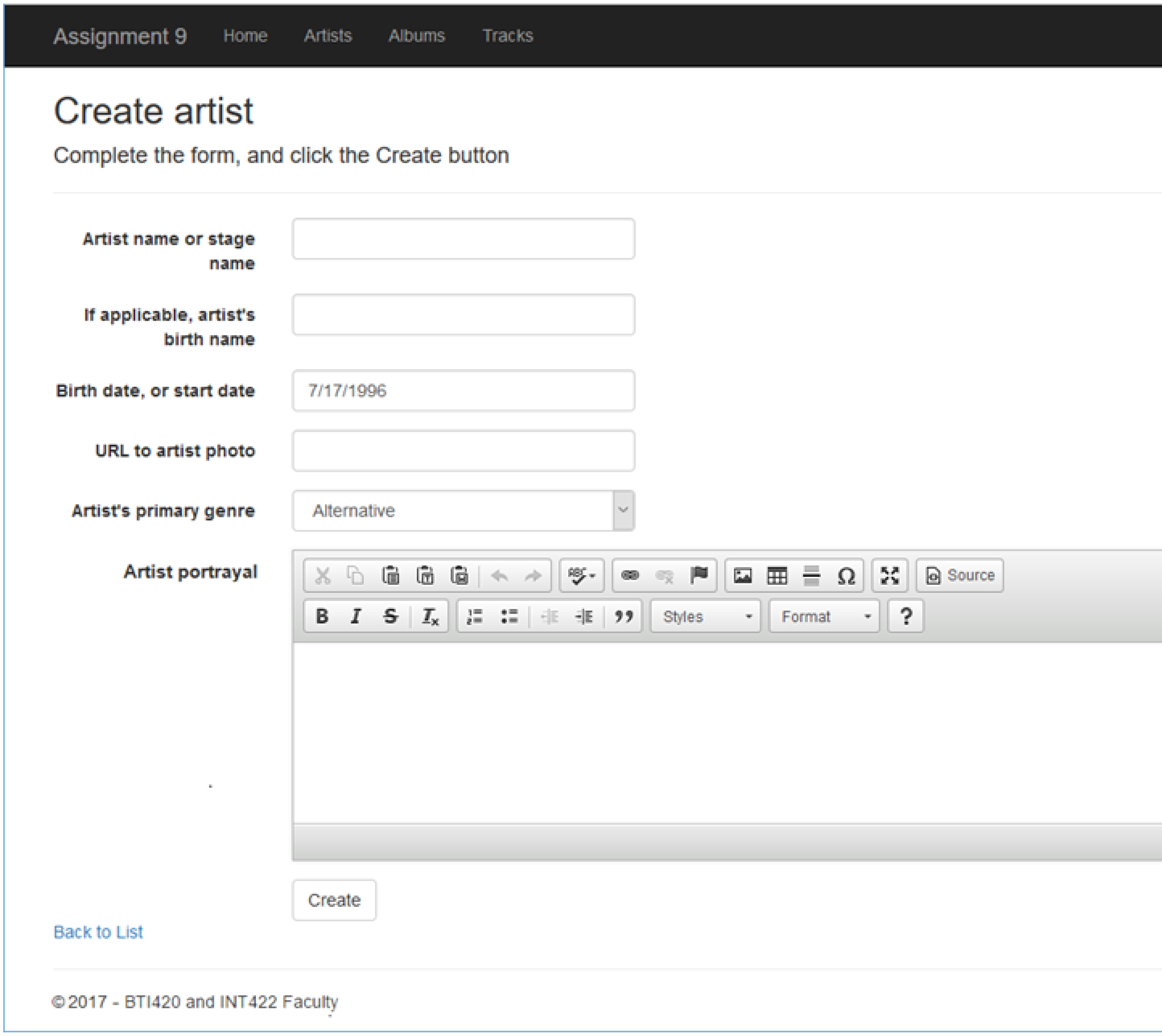
Implement "add new" for Artist

This task will be similar to the "add new" Artist task that you coded in Assignment 8. However, the updated Artist design model class has a new "**Portrayal**" string property, which will hold rich text.

Implement this task. Again, use your memory, class notes, or the patterns summary document to complete this task. Make sure that you implement the recently learned *rich text editing* tasks (also documented above). It may be possible to re-use some of your design and code from Assignment 8.

When you're testing, you can typically get **portrayal** information from Wikipedia (or other web resources).

Maybe your "add new" Artist view will look like the following. Click the image to open it full-size in a new tab/window.


Assignment 9 Home Artists Albums Tracks

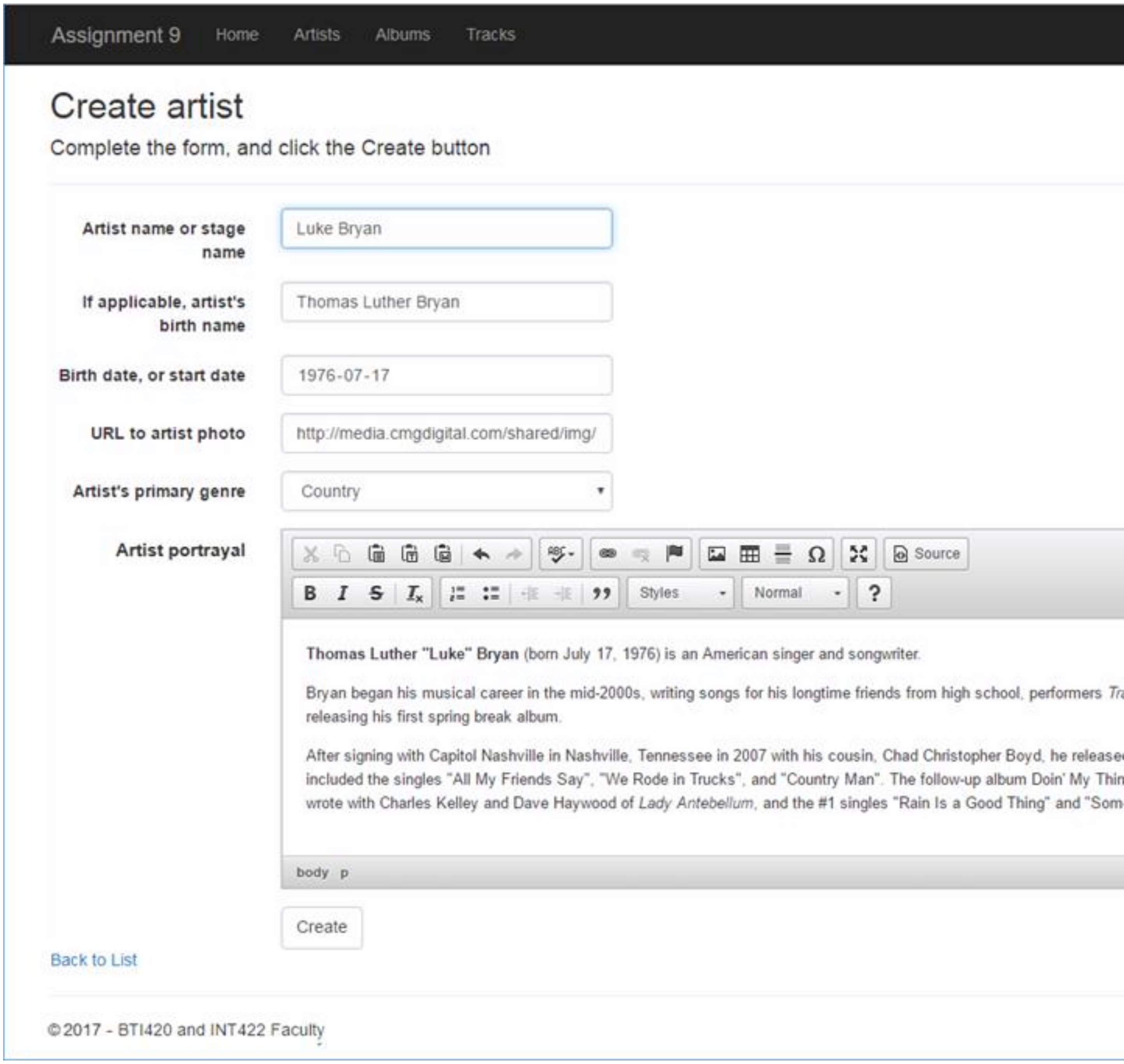
Create artist

Complete the form, and click the Create button

Artist name or stage name	
If applicable, artist's birth name	
Birth date, or start date	7/17/1996
URL to artist photo	
Artist's primary genre	Alternative
Artist portrayal	

Back to List

© 2017 - BTI420 and INT422 Faculty



Create artist

Complete the form, and click the Create button

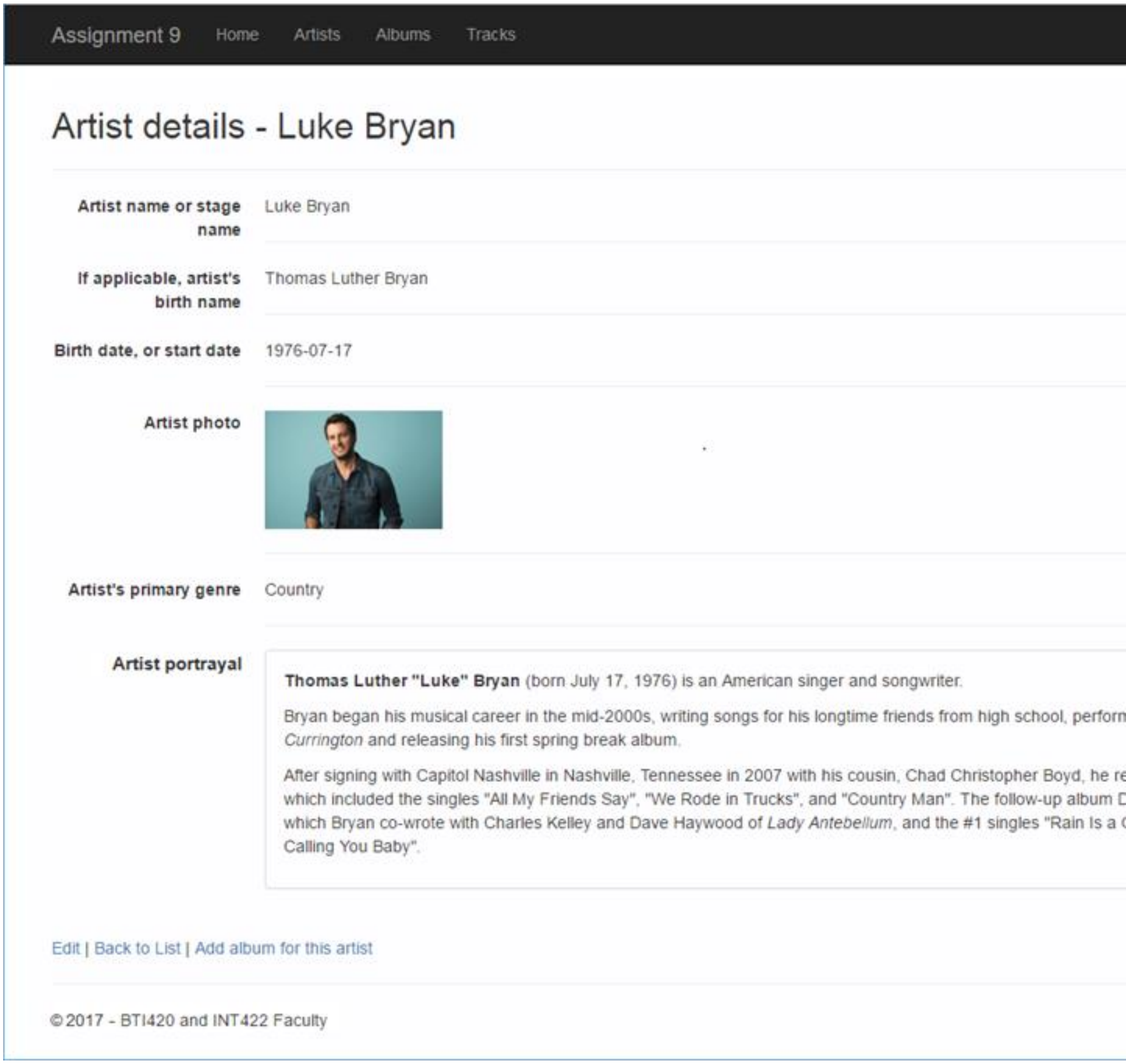
Artist name or stage name	Luke Bryan
If applicable, artist's birth name	Thomas Luther Bryan
Birth date, or start date	1976-07-17
URL to artist photo	http://media.cmgdigital.com/shared/img/
Artist's primary genre	Country
Artist portrayal	

Thomas Luther "Luke" Bryan (born July 17, 1976) is an American singer and songwriter.

Bryan began his musical career in the mid-2000s, writing songs for his longtime friends from high school, performers *Tr*
releasing his first spring break album.

After signing with Capitol Nashville in Nashville, Tennessee in 2007 with his cousin, Chad Christopher Boyd, he release
included the singles "All My Friends Say", "We Rode in Trucks", and "Country Man". The follow-up album Doin' My Thin
wrote with Charles Kelley and Dave Haywood of *Lady Antebellum*, and the #1 singles "Rain Is a Good Thing" and "Som

©2017 - BTI420 and INT422 Faculty



Artist details - Luke Bryan

Artist name or stage name	Luke Bryan
If applicable, artist's birth name	Thomas Luther Bryan
Birth date, or start date	1976-07-17
Artist photo	
Artist's primary genre	Country
Artist portrayal	**Thomas Luther "Luke" Bryan** (born July 17, 1976) is an American singer and songwriter. Bryan began his musical career in the mid-2000s, writing songs for his longtime friends from high school, perform *Currington* and releasing his first spring break album. After signing with Capitol Nashville in Nashville, Tennessee in 2007 with his cousin, Chad Christopher Boyd, he re which included the singles "All My Friends Say", "We Rode in Trucks", and "Country Man". The follow-up album D which Bryan co-wrote with Charles Kelley and Dave Haywood of *Lady Antebellum*, and the #1 singles "Rain Is a C Calling You Baby".

Edit | Back to List | Add album for this artist

© 2017 - BTI420 and INT422 Faculty

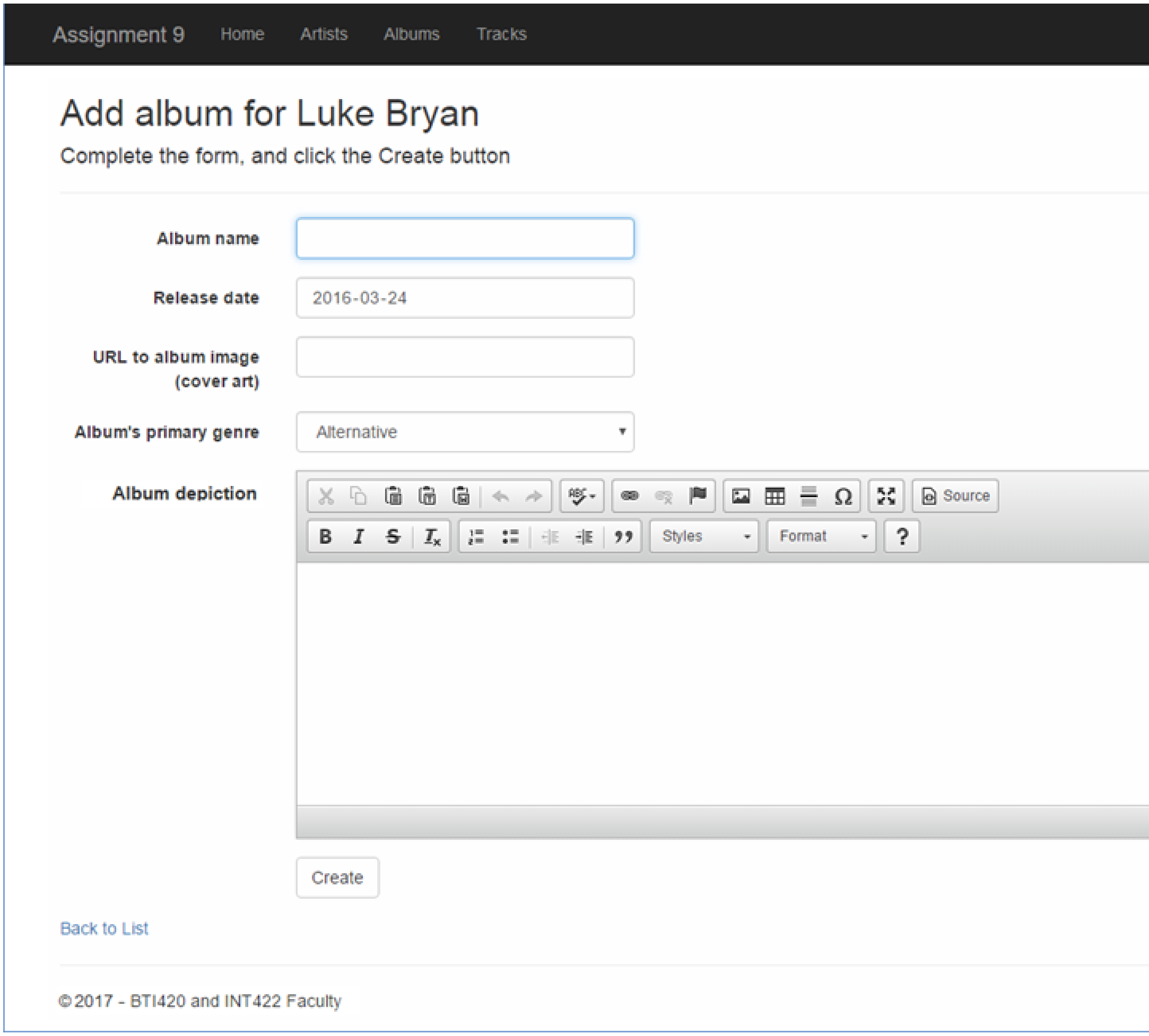
Implement “add new” for Album

This task is **NOT** exactly the same as the Assignment 8 “add new” Album task.

This task will be *much simpler*. It will enable the browser user to add a new album *for a specific known artist*. You will NOT display a (checkbox) list of all artists, or a (checkbox) list of all tracks. As a result, it will be more similar to Assignment 8's "add new" Track task, or others from the past (e.g. "add new" Vehicle task, for a specific known manufacturer).

Therefore, on a "get one" details view for Artist, add a link to enable "Add new album for this artist". Then, in the Artists controller, implement the "add new" Album functionality.

Maybe your "add new" Album view will look like the following. Click the image to open it full-size in a new tab/window.

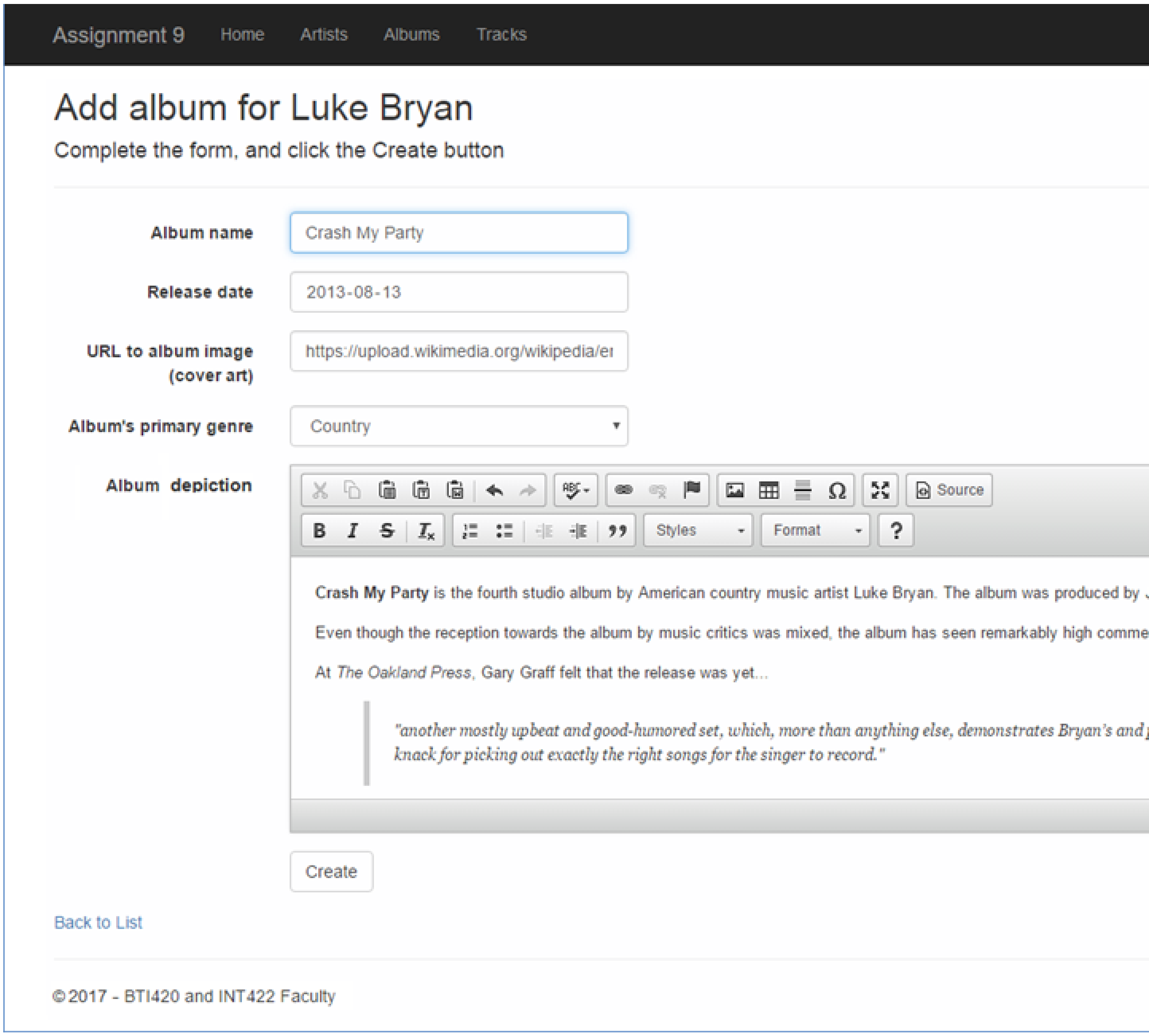


Add album for Luke Bryan

Complete the form, and click the Create button

Album name	
Release date	2016-03-24
URL to album image (cover art)	
Album's primary genre	Alternative
Album depiction	

©2017 - BTI420 and INT422 Faculty



Add album for Luke Bryan

Complete the form, and click the Create button

Album name	Crash My Party
Release date	2013-08-13
URL to album image (cover art)	https://upload.wikimedia.org/wikipedia/er
Album's primary genre	Country
Album depiction	

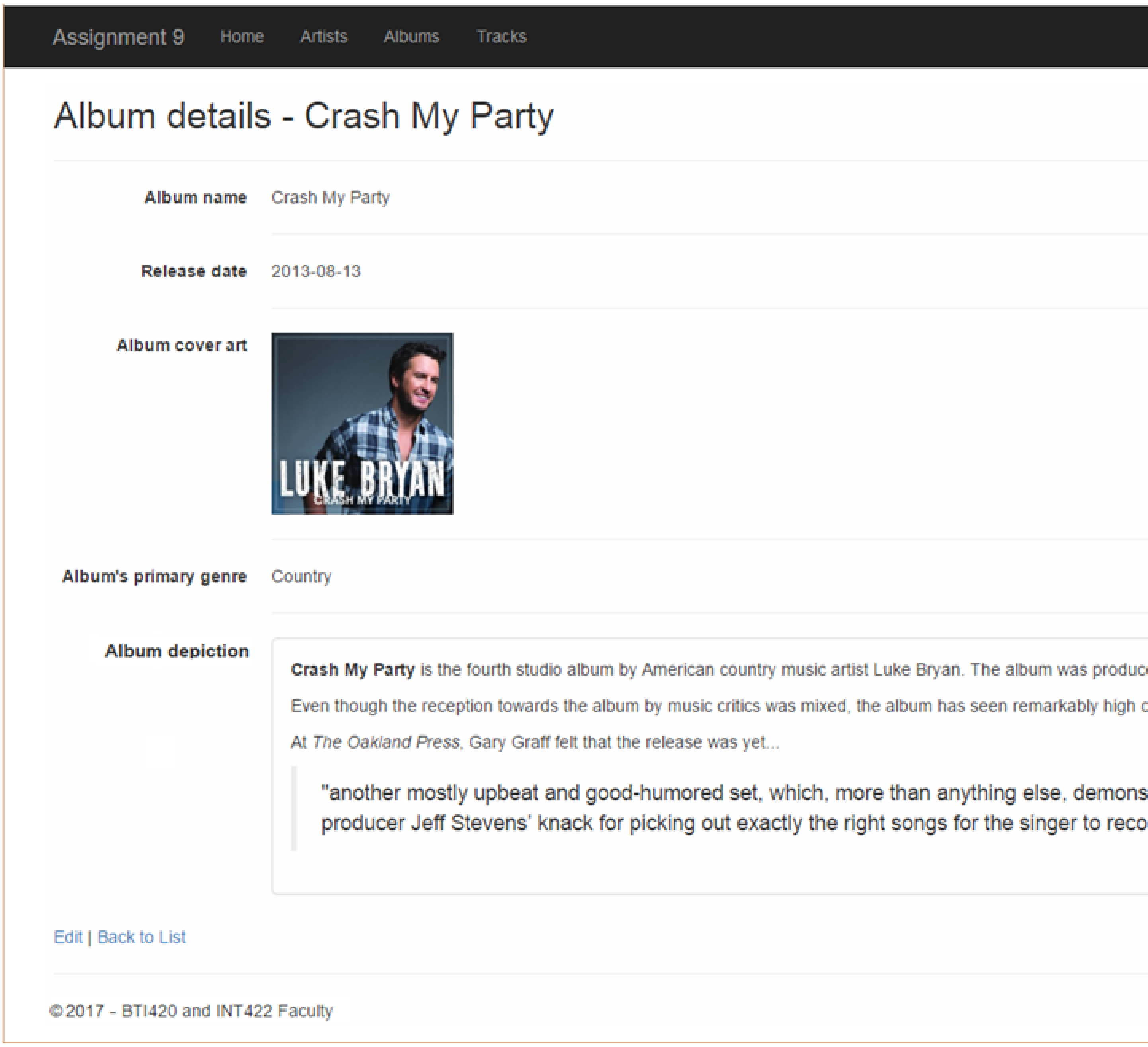
Crash My Party is the fourth studio album by American country music artist Luke Bryan. The album was produced by

Even though the reception towards the album by music critics was mixed, the album has seen remarkably high comme

At *The Oakland Press*, Gary Graff felt that the release was yet...

> *"another mostly upbeat and good-humored set, which, more than anything else, demonstrates Bryan's and*
> *knack for picking out exactly the right songs for the singer to record."*

© 2017 - BTI420 and INT422 Faculty



Album details - Crash My Party

Album name Crash My Party

Release date 2013-08-13

Album cover art

Album's primary genre Country

Album depiction

Crash My Party is the fourth studio album by American country music artist Luke Bryan. The album was produc

Even though the reception towards the album by music critics was mixed, the album has seen remarkably high c

At *The Oakland Press*, Gary Graff felt that the release was yet...

> "another mostly upbeat and good-humored set, which, more than anything else, demons
> producer Jeff Stevens' knack for picking out exactly the right songs for the singer to reco

Edit | Back to List

© 2017 - BTI420 and INT422 Faculty

Progress checkpoint

At this point, your project will be ready for the "progress" submission.

Upload it to the "**Assignment 9 progress**" area in Blackboard before the due date.

Suggestion – deploy to Azure

Although not required, your teacher team believes that it will be a really good idea to deploy your app to Azure. That way you know it will work. Later, when you add more functionality (media item handling), you will deploy again, and the online/public app will be updated.

The name of the web app should have "**a9**" in it. For example, if your Microsoft Account user name is in the format of "wsong18-wa2164", then the web app name should be "**wsong18-wa2017a9**".

The name of the sql database should also have "**a9**" in it. For example, we suggest "**A9Store**", or "**Assign9Store**".

Prepare for non-text media types

In this app, two scenarios will be implemented for non-text media type handling.

In the **simple** scenario, the existing Track entity will be modified to handle an audio media item.

In the **more complex** scenario, a new **ArtistMedia** entity will be created. This new entity will be dedicated to the description and storage of media items. The existing Artist entity will be modified to handle a collection of **ArtistMedia** objects.

Media type – add audio to the Track entity

As noted above, this is the **simple** scenario. The existing Track entity will be modified to handle an audio media item.

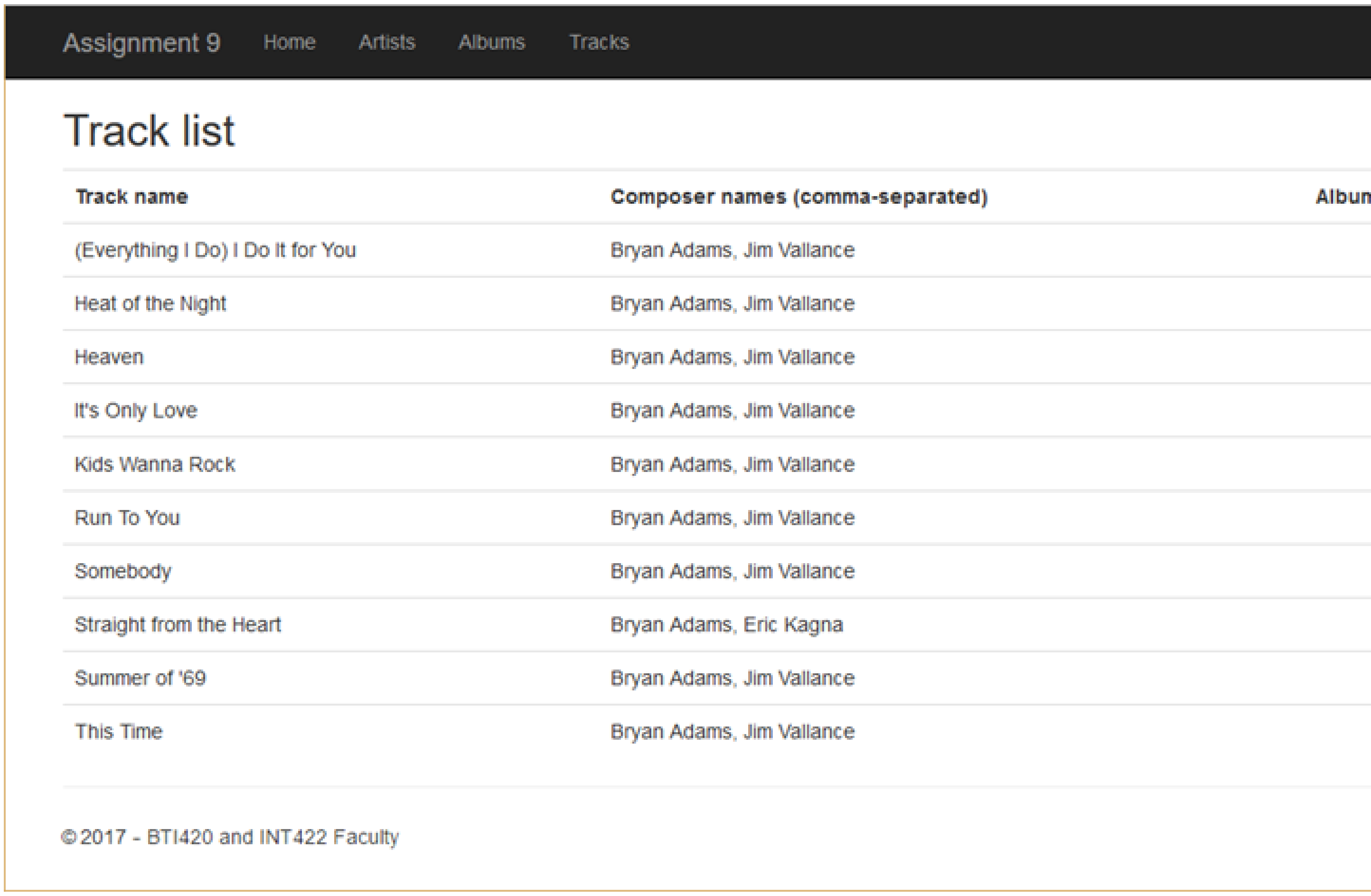
Do that now. Follow the guidance in the week 10 Lecture Notes, and in the PhotoProperty code example.

When you build/compile, and the load/run in a browser, the “model… changed” error appears. Again, add a migration, and update the database.

Implement “get all” and “get one” track

Now it’s time to implement these use cases. You will do work in the Manager class, a Tracks controller, and some views. It may be possible to re-use some of your design and code from Assignment 8.

Your track list may look like the following. Click the image to open it full-size in a new tab/window.



Track list

Track name	Composer names (comma-separated)	Albun
(Everything I Do) I Do It for You	Bryan Adams, Jim Vallance	
Heat of the Night	Bryan Adams, Jim Vallance	
Heaven	Bryan Adams, Jim Vallance	
It's Only Love	Bryan Adams, Jim Vallance	
Kids Wanna Rock	Bryan Adams, Jim Vallance	
Run To You	Bryan Adams, Jim Vallance	
Somebody	Bryan Adams, Jim Vallance	
Straight from the Heart	Bryan Adams, Eric Kagna	
Summer of '69	Bryan Adams, Jim Vallance	
This Time	Bryan Adams, Jim Vallance	

© 2017 - BTI420 and INT422 Faculty

The details view for the “get one” use case should include an HTML audio player for the track’s sample clip.

<audio src="/clip/@Model.Id" controls="controls"></audio>

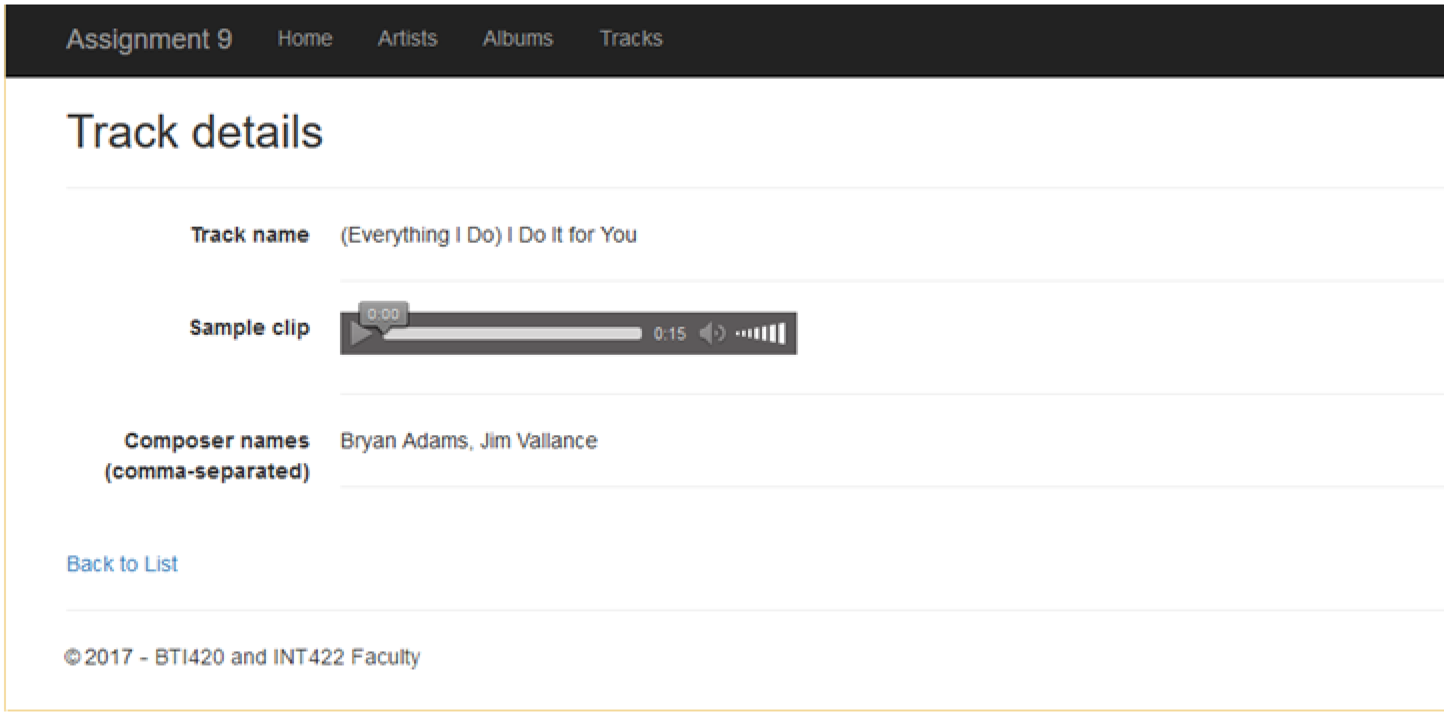
As you can see, the value of the *src* attribute is a path like this:

- **/clip/123**

That means that we need an action/method – somewhere – to fetch the clip. The PhotoProperty code example used a dedicated controller for that. You can do that if you wish.

Alternatively, you can simply add a method to the Tracks controller that will work with the manager object to fetch and deliver the clip. Your choice. Whatever you do, make sure you use attribute routing, for a clean URL.

Your track details view may look like the following. Notice the disabled state of the audio player, because it could not find the audio clip data. (We'll fix that soon.) Click the image to open it full-size in a new tab/window.



Track details

Track name	(Everything I Do) I Do It for You
Sample clip	
Composer names (comma-separated)	Bryan Adams, Jim Vallance

Back to List

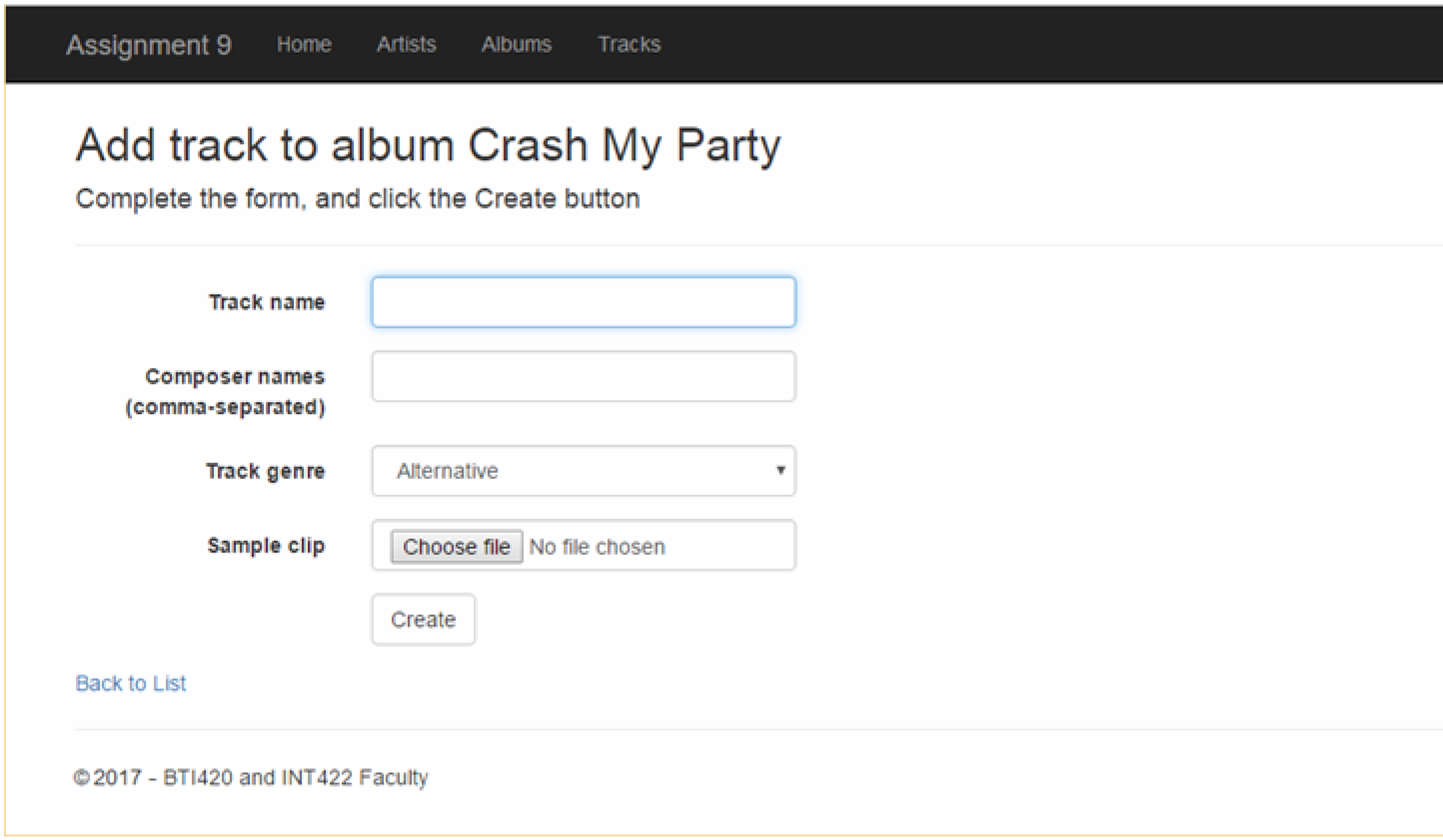
© 2017 - BTI420 and INT422 Faculty

Implement "add new" track

This will work almost exactly like it did in Assignment 8: A browser user can add a track *for a specific and known album*. Again, add a hyperlink on the album details view to "Add a new track to this album". Locate the "add new" track actions/methods in the Albums controller.

To this base, you will add the ability to upload a sample clip of the track. Again, follow the guidance in the week 10 Lecture Notes, and in the PhotoProperty code example.

Your track create view may look like the following. Click the image to open it full-size in a new tab/window.



Assignment 9 Home Artists Albums Tracks

Add track to album Crash My Party

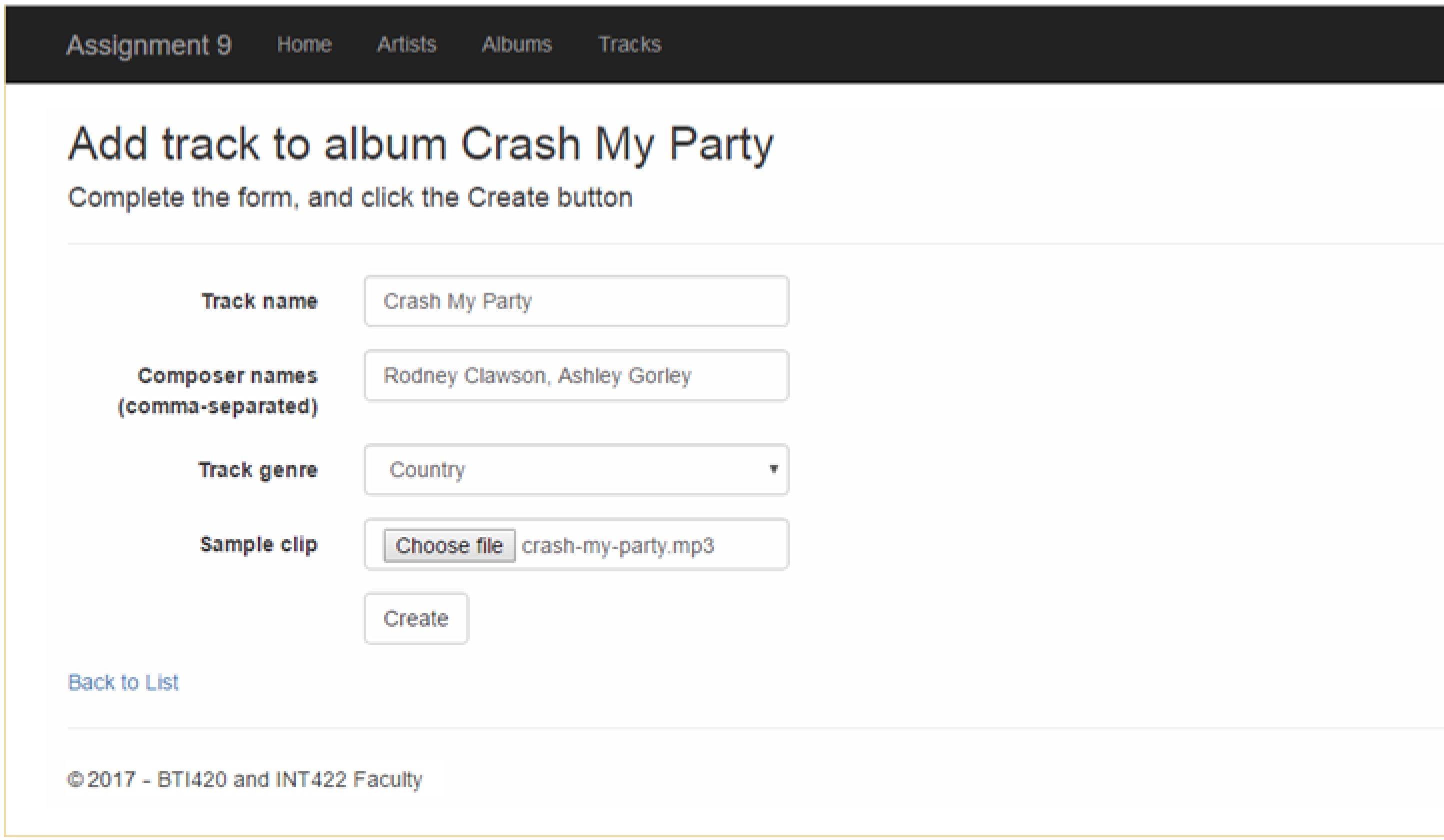
Complete the form, and click the Create button

Track name	
Composer names (comma-separated)	
Track genre	Alternative
Sample clip	Choose file No file chosen
	Create

Back to List

© 2017 - BTI420 and INT422 Faculty

After completing the form, it look like the following. Click the image to open it full-size in a new tab/window.


Assignment 9 Home Artists Albums Tracks

Add track to album Crash My Party

Complete the form, and click the Create button

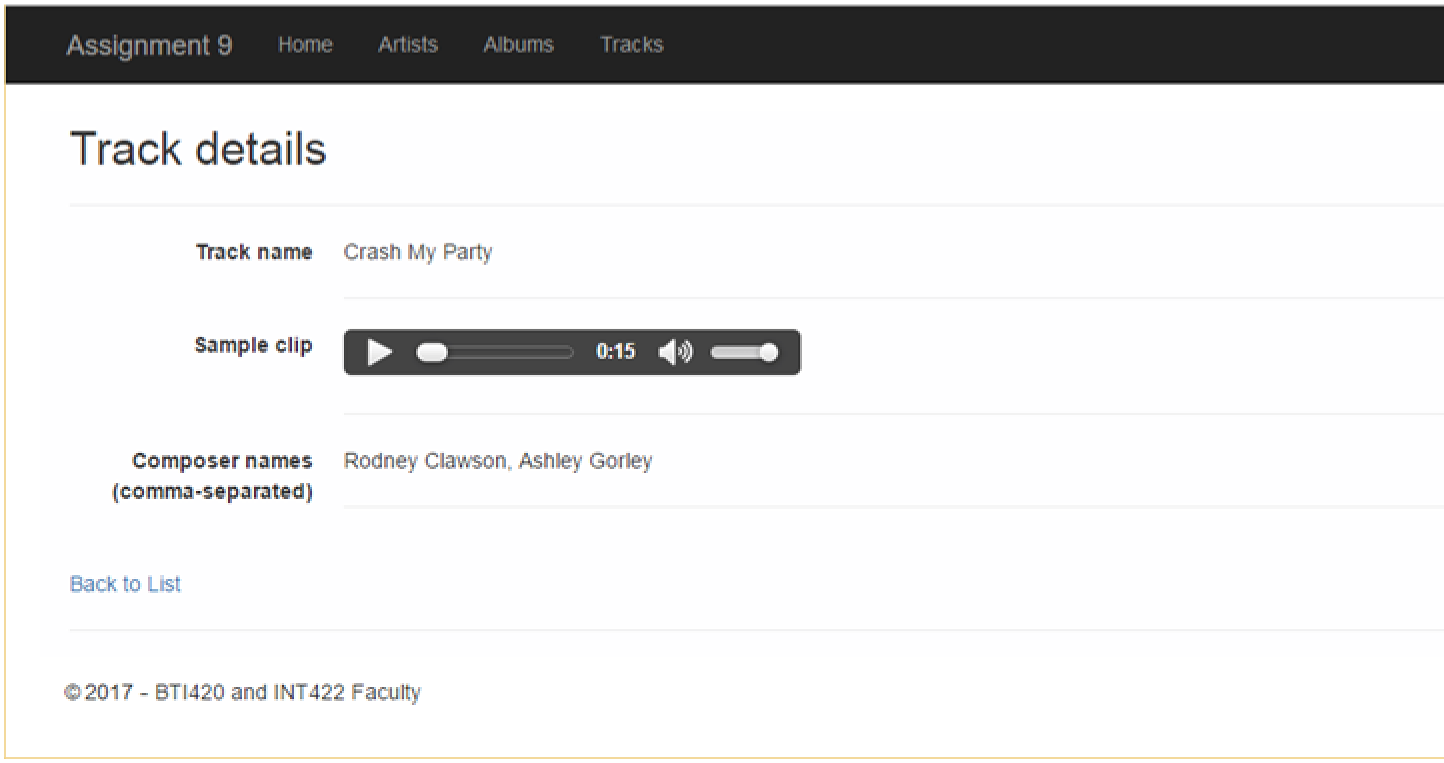
Track name	Crash My Party
Composer names (comma-separated)	Rodney Clawson, Ashley Gorley
Track genre	Country
Sample clip	Choose file crash-my-party.mp3

Create

Back to List

© 2017 - BTI420 and INT422 Faculty

Finally, after a successful save, it look like the following. Notice the enabled state of the audio player, with controls that can be used. Click the image to open it full-size in a new tab/window.



Track details

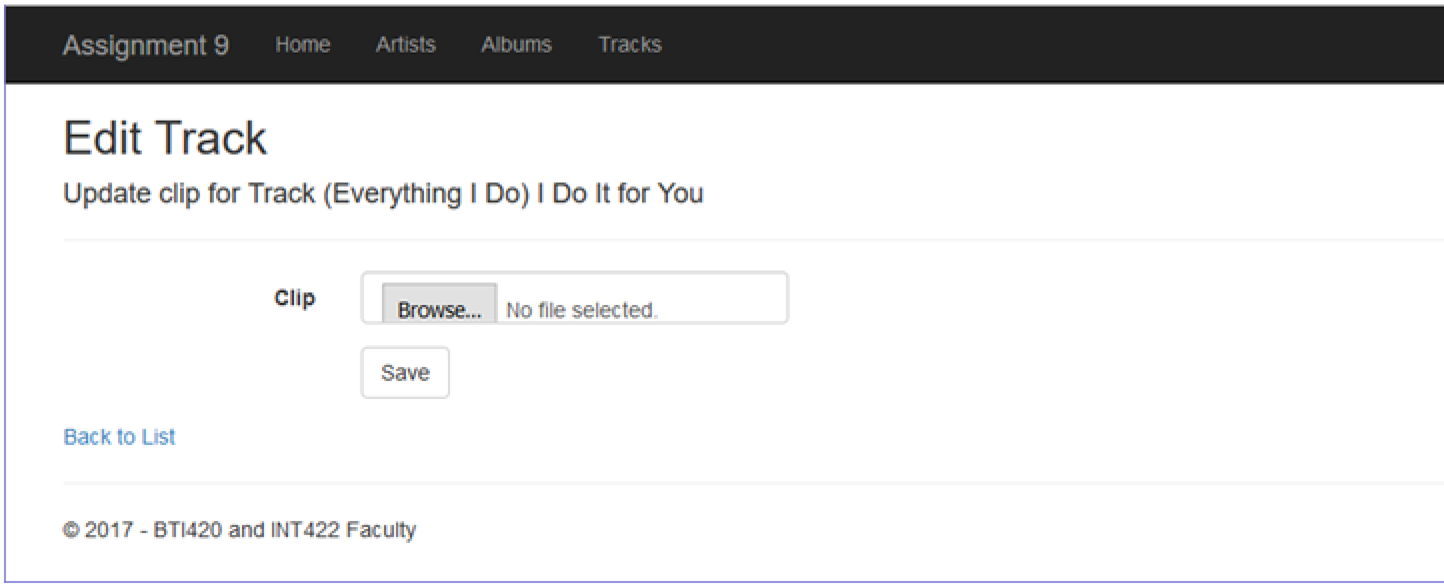
Track name	Crash My Party
Sample clip	0:15
Composer names (comma-separated)	Rodney Clawson, Ashley Gorley

Back to List

© 2017 - BTI420 and INT422 Faculty

Implement track “Edit existing” and “Delete item”

Among the entities of Artist, Album and Track, only Track need to implement “Edit existing” and “Delete item”. A browser user can edit or update clip for an existing track. All tracks created using initial data, should have clips updated. Click the image to open it full-size in a new tab/window.



Edit Track

Update clip for Track (Everything I Do) I Do It for You

Clip Browse... No file selected.

Save

Back to List

© 2017 - BTI420 and INT422 Faculty

You will need track clips. Where can you get some? Here are the links: clip1.mp3, clip2.mp3, clip3.mp3. Each are about 15 seconds in length.

Please do not use full-length tracks as clips. They're too big. Multi-megabyte. Big problem when submitting your work on Blackboard.

Instead, limit your clip length to ten to fifteen seconds.

Do you want to create your own clips?

There are many ways to do this. Here is one way.

Use the Audacity software. File > Open an existing audio file that's on your computer.

Click a location about 15 seconds into the track.

Press Shift+End (or Edit > Select > Cursor to track end). Press Delete.

If you want a nice fade-out, click a location about 2 seconds from the end of the track. Shift+End again. Then Effects > Fade Out. (This works too for Fade In at the beginning of the track.)

Then File > Export Audio

Media type – add a media collection to the Artist entity

As noted above, this is the **more complex** scenario. A new ArtistMedia entity will be created, and the existing Artist entity will be modified to handle a collection of ArtistMedia objects.

In this scenario, the design and coding approach will be similar to the one in the recent *PhotoEntity* code example.

The one difference is that we will permit *any kind* of media item to be associated with an artist object. In other words, a photo, some audio, or video, or even a digital document like a PDF or Word-or-Excel document.

What's different here

During rendering, in the view code, we’ll make decisions about how to render a media item:

If image, then render an HTML <img> element.

If audio, then render an HTML <audio> element.

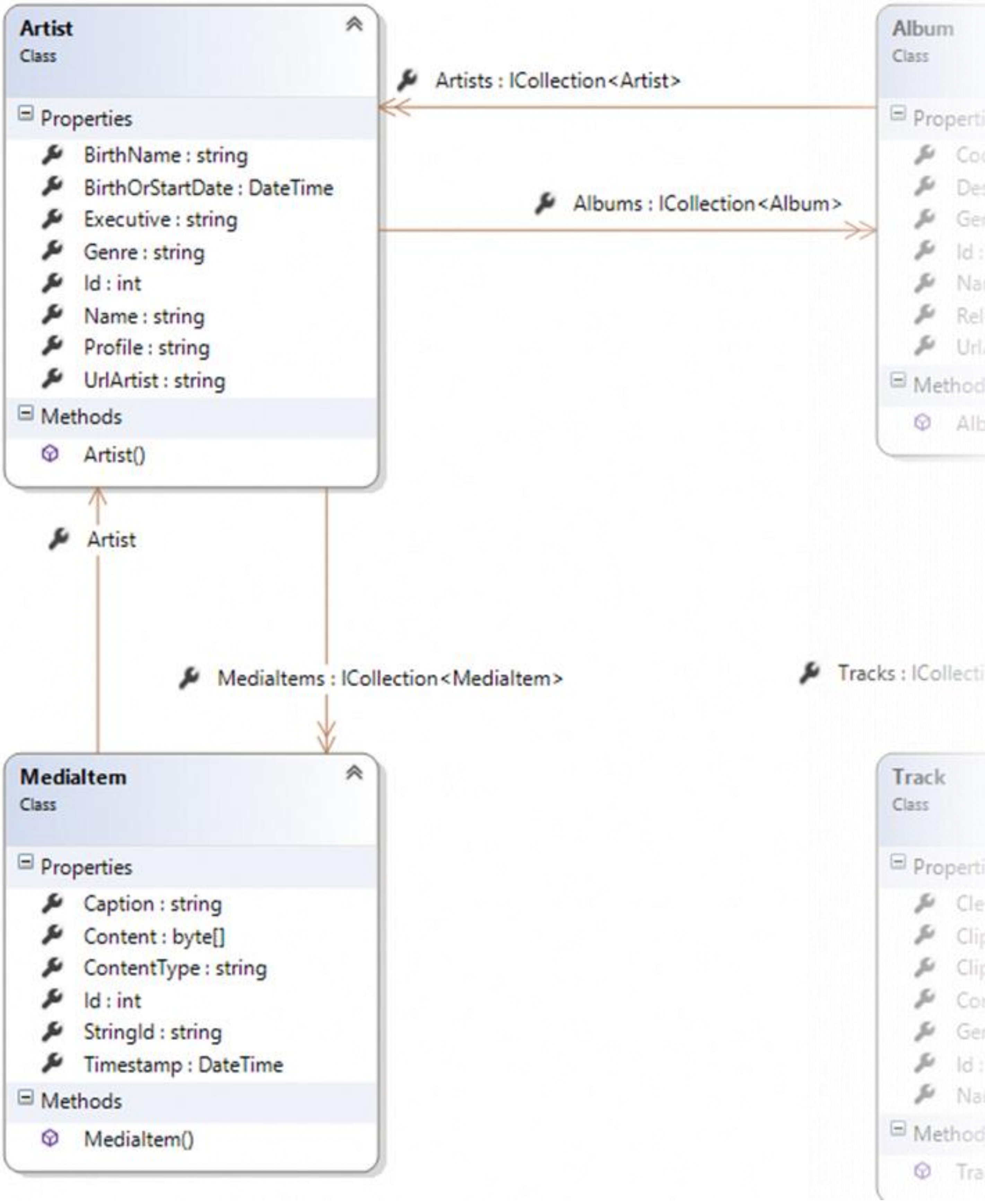
If video, then render an HTML <video> element.

If PDF, or Word, or Excel (etc.), then render an HTML <a> element, which uses the download-and-save workflow.

Implementing Artist object – MediaItem collection

Add a MediaItem design model class, probably with the same properties and constructor design that are in the *PhotoEntity* code example’s dedicated media item class.

It will be associated with Artist, which has a one-to-many association with MediaItem.


Artist
Class
Properties
BirthName : string
BirthOrStartDate : DateTime
Executive : string
Genre : string
Id : int
Name : string
Profile : string
UrlArtist : string
Methods
Artist()
Artists : ICollection<Artist>
Albums : ICollection<Album>
Album
Class
Properti
Methods
Artist
MediaItems : ICollection<MediaItem>
Tracks : ICollecti
MediaItem
Class
Properties
Caption : string
Content : byte[]
ContentType : string
Id : int
StringId : string
Timestamp : DateTime
Methods
MediaItem()
Track
Class
Properti

Remember to add the DbSet<TEntity> property in the data context class.

Build/compile, and run in a browser. It will show an error, because the "model backing the 'ApplicationDbContext' context has changed since the database was created." Add another migration, and update the database, before continuing.

View model classes – MediaItem

A "…Base" class, with identifying and descriptive properties, is needed.

Hint and tip:

Include the ContentType property. It will be useful later.

A "…Content" class, for the digital content of the media item, is needed.

Next, we need to write the view model classes that handle the "add media item for a known/existing artist" workflow.

An "…AddForm" class is needed. As you have learned, it MUST have an artist identifier, and it SHOULD have some descriptive information about the artist, to display on the HTML Form.

An "…Add" class is needed. It MUST have the artist identifier, and the properties that capture information and data for the media item.

Remember to add AutoMapper maps to cover the use cases.

Manager class method for "add media item for artist"

The design and coding approach will be similar to any one-to-many scenario where you are adding a new object for a known/existing object. This approach has been implemented many times before:

- Add a new Product for an existing Supplier
- Add a new Vehicle for an existing Manufacturer
- Add a new Album for an existing Artist
- Add a new Track for an existing Album

Here, we are adding a new MediaItem for an existing Artist.

Therefore, follow this well-known coding pattern. Add in the media-handling code, to extract and save the media item data from the HttpPostedFileBase object.

Artists controller "add new" media item handling

As you have learned, when you're adding an item to a dependent collection, you start with the dependent object.

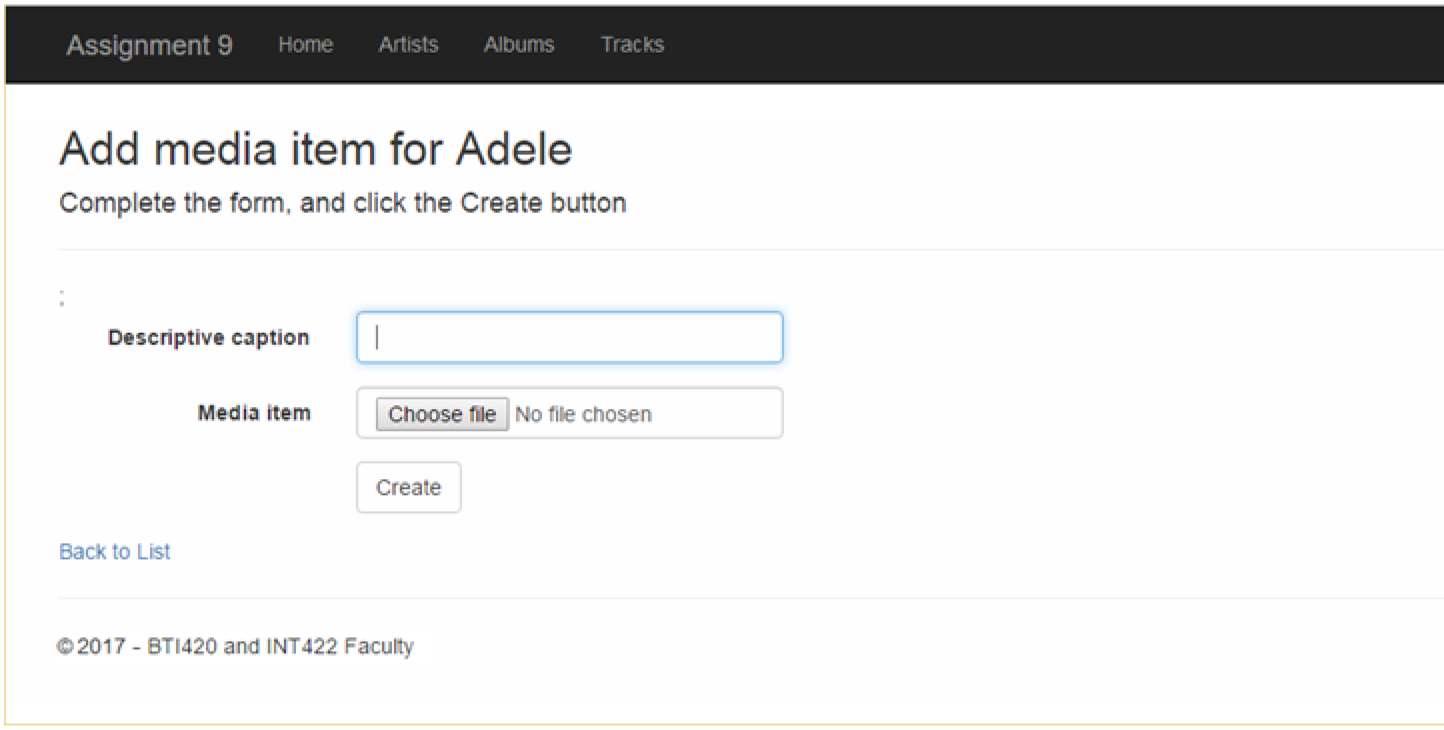
In other words, we work with the Artists controller, and code the "add new" media item there. You have already done that with "add new" album, so do the same for "add new" media item. Yes, the Coordinator role claim will be needed to complete this task.

Create the GET method. Use attribute routing.

Generate the view. Remember to:

1. Change the BeginForm() constructor to specify the correct "enctype"
2. Add the hidden ArtistId information

At this point, your view may look something like the following. Click the image to open it full-size in a new tab/window.



Add media item for Adele

Complete the form, and click the Create button

Descriptive caption	
Media item	Choose file No file chosen

© 2017 - BTI420 and INT422 Faculty

Next, create the POST method. Test your work.

Add the dedicated media item delivery controller and Manager method

Next, add the dedicated media item delivery controller.

From the PhotoEntity code example, you should also include the download-and-save functionality.

Add a Manager class method that will deliver a media item object.

Prepare to display the media items

Let’s modify the existing artist details view, and all its bits and pieces.

We will create a new view model class, ArtistWithMediaInfo (in the style similar to the *PhotoEntity* code example). Remember to add an AutoMapper map.

In the Manager class, create another “get one” method (copy/paste), but it will fetech/include the media items, and return an object of the new type (above).

In the Artists controller, call that new manager method. You’ll have to change the model type in the view too.

At this point in time, build/compile, and run in a browser. The app should continue to work.

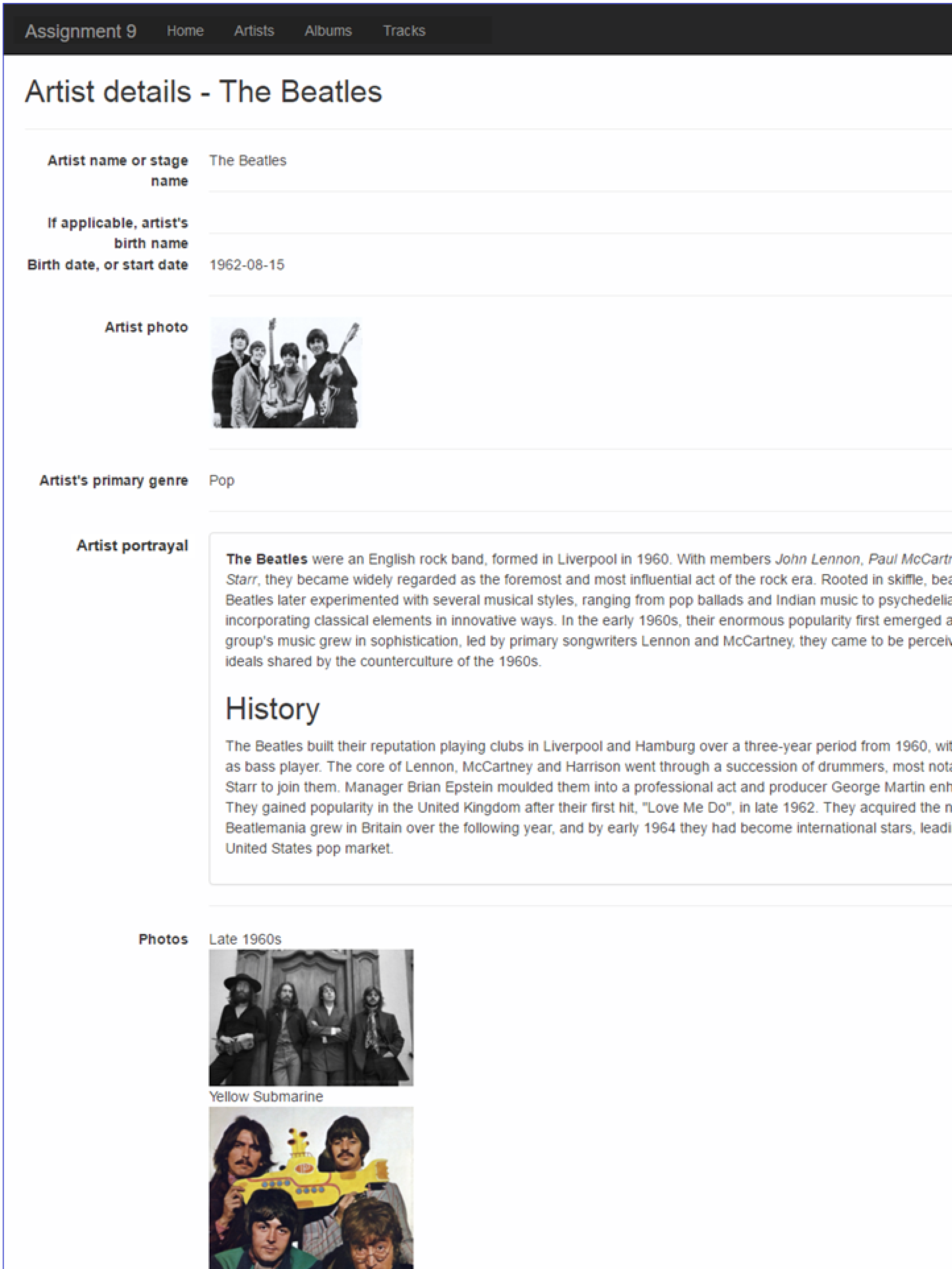
Displaying or rendering media items

There are many ways to display or render media items. Images and sounds can use the built-in HTML elements.

For other content – digital documents for example – the content can be rendered as a hyperlink that references the media item. For best results, use the save-to-download feature for their URLs.

If you want to render a text-based hyperlink, do it. Alternatively, you can include icons in your app, and render those as the hyperlink. Here are some icons that you can save and use:

In the teacher's sample solution, we chose to render the media items in groups. For example, we first rendered all the photos. Then the sounds. And then the digital documents. Click the image to open it full-size in a new tab/window.



Assignment 9 Home Artists Albums Tracks

Artist details - The Beatles

Artist name or stage name	The Beatles
If applicable, artist's birth name	
Birth date, or start date	1962-08-15
Artist photo	
Artist's primary genre	Pop

Artist portrayal

The Beatles were an English rock band, formed in Liverpool in 1960. With members *John Lennon, Paul McCartr*
Starr, they became widely regarded as the foremost and most influential act of the rock era. Rooted in skiffle, bea
Beatles later experimented with several musical styles, ranging from pop ballads and Indian music to psychedelia
incorporating classical elements in innovative ways. In the early 1960s, their enormous popularity first emerged a
group's music grew in sophistication, led by primary songwriters Lennon and McCartney, they came to be percei
ideals shared by the counterculture of the 1960s.

History

The Beatles built their reputation playing clubs in Liverpool and Hamburg over a three-year period from 1960, wit
as bass player. The core of Lennon, McCartney and Harrison went through a succession of drummers, most nota
Starr to join them. Manager Brian Epstein moulded them into a professional act and producer George Martin enh
They gained popularity in the United Kingdom after their first hit, "Love Me Do", in late 1962. They acquired the n
Beatlemania grew in Britain over the following year, and by early 1964 they had become international stars, leadi
United States pop market.

Photos

Late 1960s

Yellow Submarine

Each group was rendered with “foreach” code, but using only the desired media type. For example:

```
<dt>

  Photos

</dt>

<dd>

  @foreach (var item in Model.MediaItems.Where(m =>
m.ContentType.Contains("image/")))

  {

    <div>

      <span>@item.Caption</span><br>

      <img src="~/media/@item.StringId" alt="@item.Caption"
title="@item.Caption" width="200">

    </div>

  }

  <hr>

</dd>
```

Publish to Azure

Follow the guidance in this document to deploy/publish your web app to Azure:

https://scs.senecac.on.ca/~wei.song/int422/assignments/INT422-Assignment7.html

Please note that you ~~must NOT~~ may have to create a new database server. And you need to create a new SQL Database.

You may need to delete the items that is longer needed, e.g. if you have your assignment 7 marked already, you can remove the web app server, database server and database used for assignment 7 on Azure.

Suggested names, assuming that your Microsoft Account name uses the recommended format “wsong18-wa2017”:

Web app: wsong18-wa2017a9

Database server: wsong18-ds2017a9

Database: Assign9Store (or A9Store)

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.

Important Notes

You must double check your assignment 9 web app:

- You MUST use the provided “Web app project A9” project template and AutoMapper static API for your assignment.
- The lengthy string properties of Artist class and Album classes for holding rich-text descriptive info must be named as specified.

Fail to do so will lead to a major penalty for the assignment.

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*.

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.

v