CS 3340 – HW 4 – Album Search

Contents

[1 Version C (75 points) 1](#_Toc130212962)

[2 Version B (85 points) 2](#_Toc130212963)

[3 Version A (100 points) 2](#_Toc130212964)

[Appendix 1 getAlbums Method 3](#_Toc130212965)

[Appendix 2 Json Support in .NET 3](#_Toc130212966)

# Version C (75 points)

An *Album* has a title, artist, and year released. Write a page that uses Ajax to allow a user to search for albums on the server by title and display the results. For example (yours should say HW 4):



Requirements:

1. Name your search page: *DefaultC.aspx.*
2. Add a class named, *Album* with: *Title, Artist,* and *Year* (int) properties. Hint: this is a very simple class.
3. Name your search service page: *AlbumService.aspx.* This is the page your Ajax will send the request to.
4. The *getAlbums* method found in [Appendix 1](#getAlbumsMethod) builds a list of albums that serves as the album in-memory database. Place this method in *AlbumService.aspx*. This builds a list of Albums that will be searched.
5. Must use Ajax via jQuery (recommended) or JavaScript. Must NOT use the built in Ajax controls in .NET.
6. Must return the data from *AlbumService* as a Json array. Instructions for enabling and using Json in C# provided in [Appendix 2](#JsonSupportinNET).

Hint:

1. This assignment is similar to *AjaxJQueryJson.aspx* which uses *MartianService.aspx.cs*.
2. The client code that processes the return must *parse* the Json, which turns it into a JS array. See the Json Notes from earlier in the semester. Those notes also show how to traverse the array.
3. The search does partial matching starting from the beginning of the title of the album.

# Version B (85 points)

Same as Version C, except allow searching by Artist (partial match) and Year (exact match). For example (yours should say HW 4):



Hints:

1. Do Version C, then modify.
2. Define a click event for each button. I.e. copy the one from Version C and rename some things.
3. You’ll have to modify your *AlbumService* code to accommodate different types of searches.

# Version A (100 points)

Same as Version B, except (a) use radio buttons to select the search type, (b) one text box. For example:



Hints:

1. The *RadioButtonList* should be named, *rblSearchCriteria*; oriented horizontally; and set both the *Text* and *Value* for each of the three items: “Title”, “Artist”, “Year”, respectively. If you do this, then your button event handler that does the Ajax can see which radio button is selected with this code:

var searchType = $("input[name='rblSearchCriteria']:checked").val();

1. You’ll need to send two pieces of information to the server: the *search type* (*i.e. Title, Artist,* or *Year*) and the *search criteria* (*i.e.* what the user typed in). Thus, in the *data* section of the jQuery ajax call, it can be specified like this:

data: {

 type: searchType,

 criteria: searchCriteria

},

 Appendix

1. getAlbums Method

The method below generates a list of albums that will be searched. You can call it with a line of code like this:

List<Album> albums = getAlbums();

private List<Album> getAlbums()

{

 List<Album> albums = new List<Album>();

 //An arbitrary array of names that are used as suggestions.

 string[] titles = { "Ace", "Blood on the Tracks", "I Saw the Light", "Have You Seen Me Lately",

 "Old Memories", "Hot Dawg", "Pickin' the Blues", "Dave's Picks Volume 4",

 "High, Low and In Between", "Find Your Angel" };

 string[] artists = { "Bob Weir", "Bob Dylan", "Bill Monroe", "Carly Simon",

 "Del McCoury", "David Grisman", "Doc Watson", "Grateful Dead",

 "Townes Van Zandt", "Verlon Thompson" };

 int[] years = { 1972, 1975, 1959, 1990,

 2012, 1978, 1985, 2012,

 1971, 2014 };

 for(int i=0; i<titles.Length; i++)

 {

 albums.Add(new Album(titles[i], artists[i], years[i]));

 }

 return albums;

}

1. Json Support in .NET

To configure Json in .NET do the following:

1. Right-click the solution in the Solution Explorer and choose: Manage NuGet Packages
2. Choose from the top: Browse
3. In the search bar, type: json.net
4. Choose the (probably the second item) one named: Newtonsoft.Json
5. Choose: Install
6. In *RecordService.aspx.cs* add this using statement: using Newtonsoft.Json;

At the end of your code, you will use this command to turn your list of matched albums into a Json string:

string json = JsonConvert.SerializeObject(albumMatches);

// Send the response

Response.Write(json);

// End the page lifecycle and send immediately.

Response.End();

I used a list to hold the albums that match:

List<Album> albumMatches = new List<Album>();