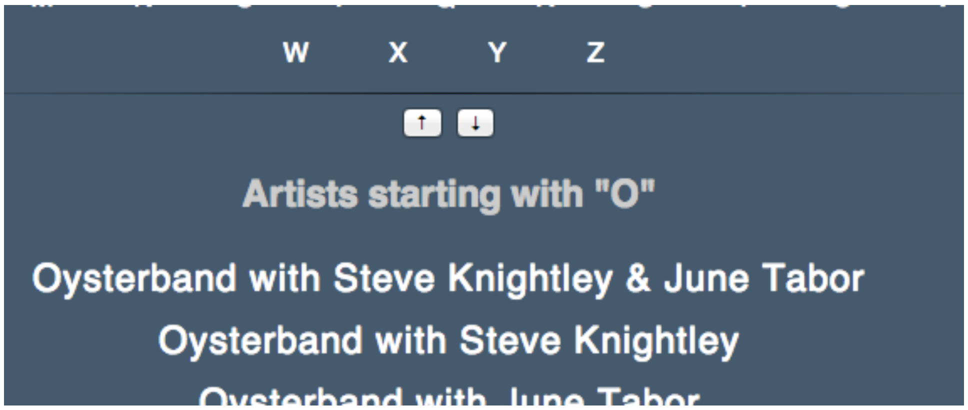
Session 3

OVERVIEW

In this session, you will add sort capabilities to the Music browser's artists view:

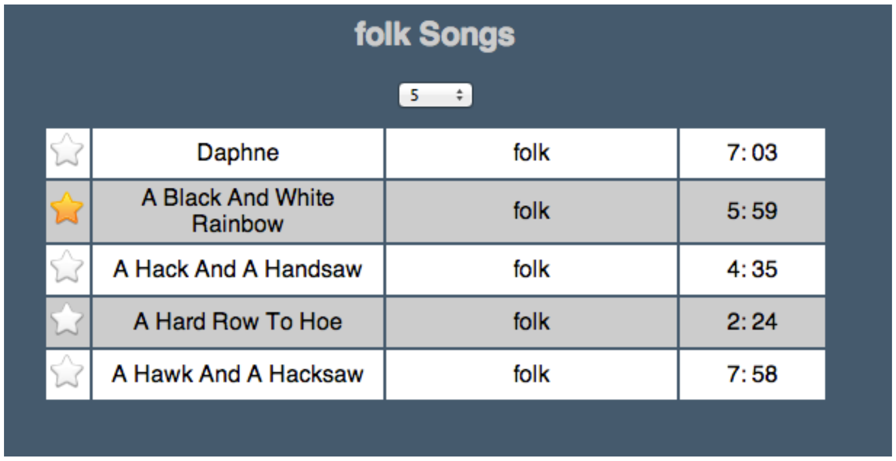


Artists starting with "O"

Oysterband with Steve Knightley & June Tabor

Oysterband with Steve Knightley

You'll also add the ability to tag the Hot tracks with a star, and limit the number of rows that you see when browsing by Genre.



☆	Daphne	folk	7:03
★	A Black And White Rainbow	folk	5:59
☆	A Hack And A Handsaw	folk	4:35
☆	A Hard Row To Hoe	folk	2:24
☆	A Hawk And A Hacksaw	folk	7:58

We've added a new table to the music database to allow you to retrieve a track by track_id. To set this up, refer to the Setup section below. The TracksDAO class has been updated to reflect this. In addition, various UI elements have been added for you. You'll simply need to implement functionality in the TracksDAO and ArtistDAO classes.

Setup

You may need to adjust the paths to match your directory structure. Notice that this table only uses one column for the Primary Key, which means that the Primary Key is equivalent to the Partition Key. Each partition therefore has a very small number of cells. This is known as a *Narrow Table*.

```
create table track_by_id (track text, artist text, track_id
UUID, track_length_in_seconds int, genre text,music_file
text, primary key (track_id));

copy track_by_id (track_id, genre, artist, track,
track_length_in_seconds, music_file) FROM 'songs.csv' WITH
DELIMITER = '|'  AND HEADER=true;
```

If you would like to wipe out your database and start from scratch:

We've provided a script to re-create the keyspace and tables from scratch. Wipe out everything you've done by dropping the keyspace. You can't be "using" the keyspace that you're dropping, so either use the system keyspace:

```
use system;
```

or restart cqlsh.

To drop the keyspace:

```
drop keyspace playlist;
```

To recreate everything, there is a script in the “scripts” directory of the project. You can run the script on Unix machines with the SOURCE command. For example, if you run it from the scripts directory:

```
source ‘playlist.cql’;
```

Double-check the paths in playlist.cql to the .csv files in the COPY commands, and ensure that the copy command will read in the csv files.

Load the application in Eclipse as you have done in the previous sessions, and run it accordingly.

EXERCISE #1: Adding an “Order By” Clause to a Select Statement

1. If you look at the screenshot of an “artists” page above, you’ll noticed the up and down arrow buttons. Pressing this button pass a new Boolean parameter called “desc” to the ArtistsDAO. listArtistByLetter method:

```
public static List<String> listArtistByLetter(String
first_letter, boolean desc)
```

If the desc parameter contains true, return results sorted in descending order, otherwise the results should appear in ascending order. Use Cassandra functionality to handle the ordering of data.

EXERCISE #2: Choose the Number of Tracks to Return

1. Because there may be a large number of tracks for a given genre, we’ve added the ability to allow the user to restrict the number of tracks returned. To use this, navigate to the Song Database in the application, and choose a genre. Notice the drop-down list. If a user has not selected a number, we default to 25.

2. The default is controlled the Controller Class TrackServlet with this line using a Ternary Conditional operator.

```
  howmany = howmany == null ? "25" : howmany;
```

In addition, further down in the doGet() method, if the user choose “All” or we otherwise get something that is not a number, we use exception processing to set the num_tracks to 100,000, our version of “All”:

```
// If what comes in is not a number or is null,
// default to 100,000

      try {
        num_tracks = Integer.parseInt(howmany);
      } catch (NumberFormatException e) {
        num_tracks = 100000;
      }
```

3. The method TracksDAO.listSongsBygenre takes a parameter called "num_track" to restrict the number Tracks returned.

```
listSongsByGenre(String genre, int num_tracks)
```

4. Add code to this method to ensure that only the correct number of tracks are returned. In other words you want to limit the size of the result set.

EXERCISE #3: Marking Tracks as being "Hot Tracks"

5. In the music browser, you can click the star next to a track to mark it as a "Hot" track. Add a Boolean field to the necessary tables in the data model to store the fact a track has been starred: For example, you can use the ALTER TABLE statement to add additional columns to a table:

```
alter table track_by_artist add <something here> ;
```

6. Do you need to add the field to any additional tables?
7. The TracksDAO has a constructor called TracksDAO(Row row) which takes a single Java Driver Row object and constructs a TracksDAO – the object that represents a single track. We've added a new field to this object called starred. Fix this constructor method to set the starred field to the value of the newly added column. For example, if you used a Boolean, consider the row.getBool() method.
8. There is a non-static method called star() which is called when someone stars an object. It needs to persist this setting. In other words, set the new Boolean field(s) to true for this object.
9. Try it out – click the star on a track, and ensure that it becomes yellow. Navigate away, and re-visit the track.
10. Check the table(s) with cqlsh. Try selecting the new field in cqlsh to ensure the application sets it to true.

How the Star mechanism works – We do a read before we write here.

When you click the star, we post only the track_id to the doPost() method in the TrackController. This method calls TracksDAO.getTrackById() which returns a single track. We then call the star() method on that track.

Here's the code:

```
if (star != null) {
   TracksDAO.getTrackById(UUID.fromString(star)).star();
```

Notice that because we need to pass a UUID to getTrackByID, but the variable "star" which comes from the web page is a string, we use the static method UUID.fromString() to create the UUID object.

EXERCISE #4: Keep Hot Tracks Hot for only 30 seconds

11. The music business is extremely fickle, so songs don't stay hot for very long. Modify the star() method of the TracksDAO class to keep songs starred for only 30 seconds. Consider using features that are built-into Cassandra to accomplish this.