

Demonstration the issue documentation for OLTP Planning #2

New issue

Open

5 of 9 tasks

dwelchnait opened this issue 6 days ago · 4 comments

Assignees

Labels

documentation

Milestone

Sample desired OL...

dwelchnait commented 6 days ago · edited

Collaborator

This task list area will be completed once the implementation plan has been outlined. This area is where one creates the task list that is associated with the milestone. The tasks that are outlined in this area, are the tasks that are counted for the milestone.

☒ Create Data Models

- ☐ Create Query Models
 - ☒ Track_FetchTracksBy
 - ☒ PlaylistTrack_FetchPlaylist
- ☐ Create Command Models
 - ☒ PlaylistTrackTRXInput

☐ Create Transactional Services (methods)

- ☒ Add a Track
- ☐ Remove tracks from playlist
- ☐ Reorganize playlist tracks

☐ [Create form web page #3](#)

dwelchnait added the documentation label 6 days ago

dwelchnait added this to the Sample desired OLTP design documentation for 2018 course milestone 6 days ago

 dwelchnait self-assigned this 6 days ago

dwelchnait commented 6 days ago

CollaboratorAuthor

Expected Transactional Form

Playlist Management

This page has been created to demonstrate an UI interface for bulk maintenance of data on the database, commonly referred to as **transactional processing**. The page will allow the user to create or alter a list of tracks (Add, reorgnaize, and remove) of a playlist.

Several UI techniques will be employeed on this page. This is not the only way the interface could have been done. Special note should be directed to the display and collection of data from the Playlist table. This table will demonstrate an implementation of using CQRS data models. The error list display demonstrates the display of having multiple errors returned from a business service.

Search pattern

deep

Artist

Album

Tracks ?

	Song	Album	Artist	Length	Price
<div>+</div>	A Castle Full Of Rascals	Purpendicular	Deep Purple	05:11	0.99
<div>+</div>	A Touch Away	Purpendicular	Deep Purple	04:36	0.99
<div>+</div>	A Twist In The Tail	The Battle Rages On	Deep Purple	04:17	0.99
<div>+</div>	Anya	The Battle Rages On	Deep Purple	06:32	0.99
<div>+</div>	Anyone's Daughter	Fireball	Deep Purple	04:44	0.99

<First> <Prev> 1 2 3 4 5 <Next> <Last>

Enter playlist name:

hansenb1

Fetch List

Remove Tracks

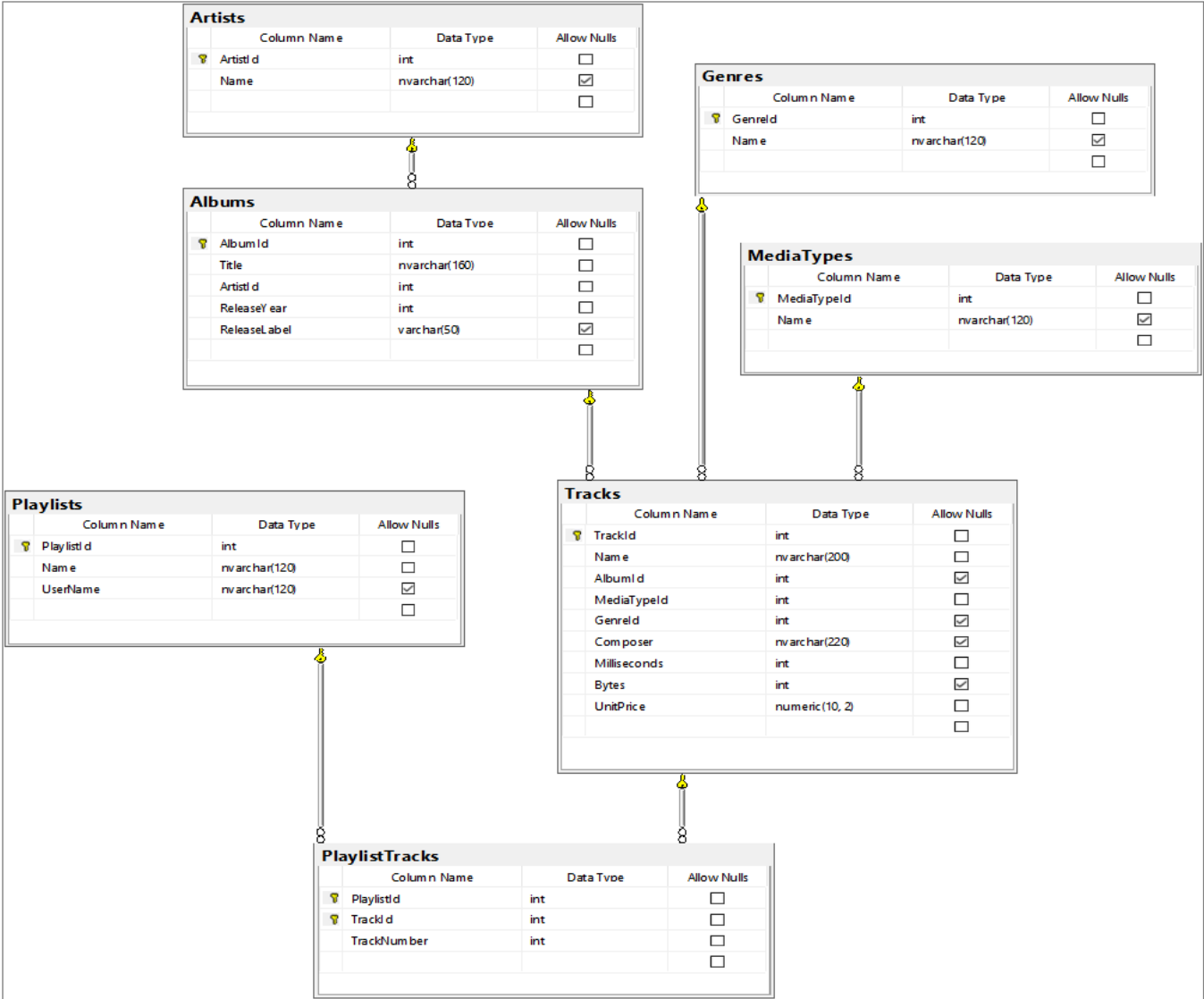
Re-Org Tracks

Trk #	Song	length	Reorg Trk#
<div><input type="checkbox"/></div> 1	Always Be All Right	02:08	<div></div>
<div><input type="checkbox"/></div> 2	A Tarde	04:26	<div></div>
<div><input type="checkbox"/></div> 3	As Dores do Mundo	04:15	<div></div>
<div><input type="checkbox"/></div> 4	Há Quanto Tempo	04:30	<div></div>
<div><input type="checkbox"/></div> 5	Vou Pra Ai	05:00	<div></div>
<div><input type="checkbox"/></div> 6	Sem Sentido	04:10	<div></div>
<div><input type="checkbox"/></div> 7	Encontrar Alguém	03:44	<div></div>
<div><input type="checkbox"/></div> 8	My Brother	04:13	<div></div>
<div><input type="checkbox"/></div> 9	Dance Enquanto é Tempo	03:49	<div></div>
<div><input type="checkbox"/></div> 10	Vício	04:29	<div></div>
<div><input type="checkbox"/></div> 11	Rapidamente	04:12	<div></div>
<div><input type="checkbox"/></div> 12	Onibusfobia	05:15	<div></div>

dwelchnait commented 6 days ago

CollaboratorAuthor

Chinook ERD for Transactional Form

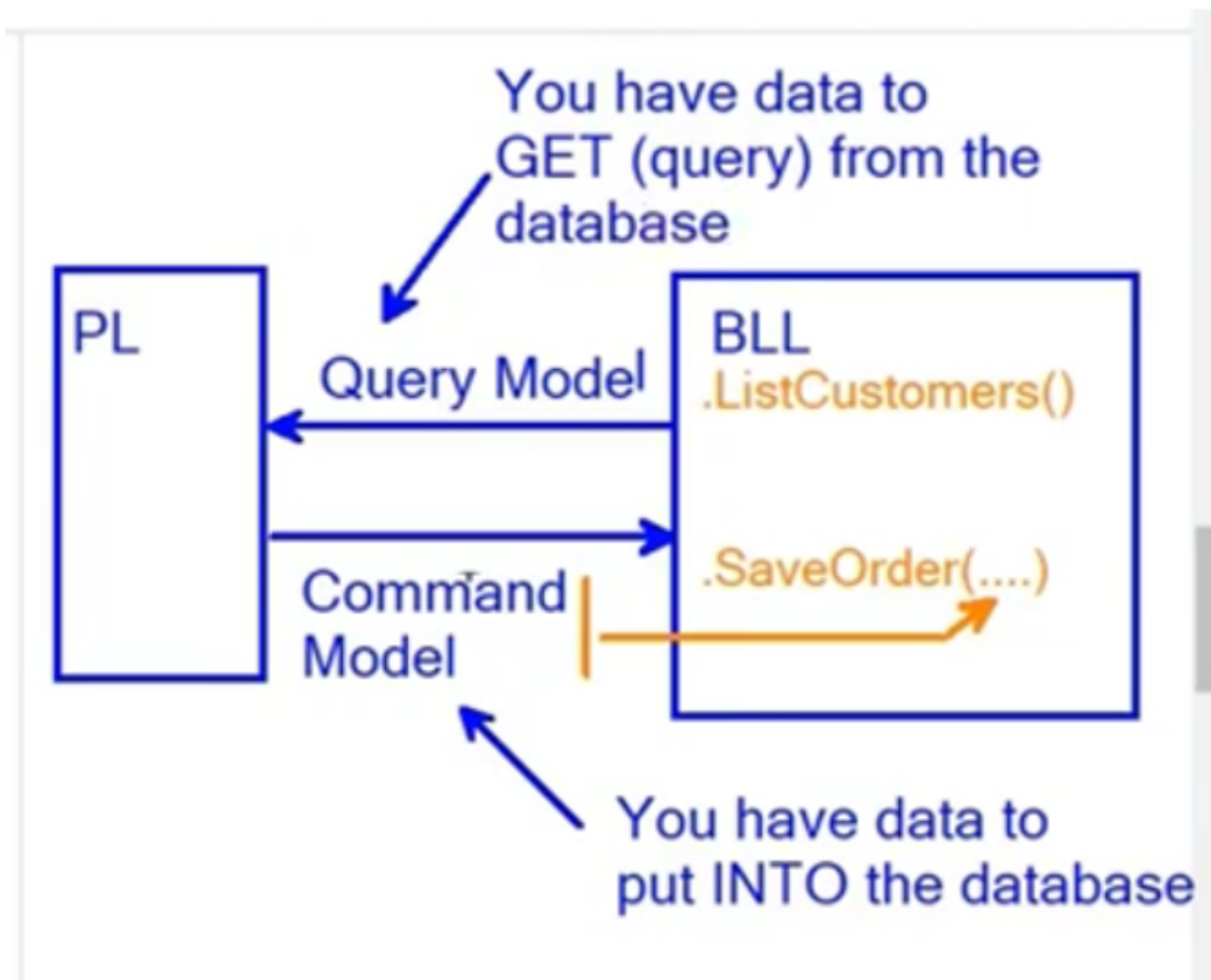


dwelchnait commented 6 days ago • edited ▾

CollaboratorAuthor

Data Models (CQRS)

The data models are the C# classes that will be coded in the class library project/application (folder: ViewModels) that holds the public data classes in our solution. These classes represent the query class models and the command class models (CQRS)



Query Models

Artist and Album Tracks fetch (Track_FetchTracksBy)

```
public class TrackSelection
{
    public int TrackId {get; set;}
    public string SongName {get; set;}
    public string AlbumTitle{get; set;}
    public string ArtistName{get; set;}
    public int Milliseconds {get; set;}
    public decimal Price {get; set;}
}
```

Current Playlist fetch (PlaylistTrack_FetchPlaylist)

```
public class PlaylistTrackInfo
{
    public int TrackId {get; set;}
    public int TrackNumber {get; set;}
    public string SongName {get; set;}
    public int Milliseconds {get; set;}
}
```

Command Models

Add Track

No model class, individual parameters

Remove/Move Tracks (shared command model due to the web page handling of BindProperty)

Remove Tracks

for the Remove Tracks functionality, the SelectedTrack and TrackId will be required for transactional processing

Move Tracks (re-sequence playlist)

for the Move Tracks functionality, the TrackId, TrackNumber and TrackInput will be required for transactional processing

```
public class PlaylistTrackTRX
{
    public bool SelectedTrack {get; set;}
    public int TrackId {get; set;}
    public int TrackNumber {get; set;}
    public int TrackInput {get; set;}
}
```

dwelchnait commented 2 days ago • edited ▾

Collaborator

Author

Business Processing Requirements

This comment will describe the various methods that will be used for Commands (CQRS) that alter the database. Queries are read only and require on average no complex processing. You *may* in another comment outline your query methods. However, commands require **business rules, data validation and manipulations of one or more tables and/or records**. Therefore, the processing of commands could required extensive logic to be outlined.

include the *method signature* and a *bullet list of processing (pseudo-code)*

Add Track

`void PlaylistTrack_AddTrack(string playlistname, string username, int trackid)`

- check that the incoming data exists
 - if a problem exists, throw an `ArgumentNullException` for missing incoming value
- check track exists
 - does not exist, throw an `ArgumentException` on trackid
- check to see if playlist exists
 - no (new playlist)
 - create a new playlist record
 - set track number to 1
 - yes (appending to existing playlist)
 - check if track already on the playlist (B/R)
 - yes
 - throw an `Exception` B/R
 - no
 - determine the next track number
- add track to playlist tracks
- check for any errors
 - yes: throw list of all collected exceptions
 - no: save all work to database

Assignees



dwelchnait

Labels

documentation

Projects

None yet

Milestone

Sample desired OLTP design documentat...

Development

No branches or pull requests

1 participant

