MileStone – OLTP Planning -48 hrs to do this in GIT

Only 1 Issue – creating a document for implementing transaction form

-space[space ]spacc text - issue comment

* Close issue after you done (you can learn more about markdown go to markdowntutorial.com

Issue#1Demonstare the components for documenting the OLTP for this course

* [] Creating milestone
* [] Creating Issue
* - Review mark down
* Discuss the opening

Issue #2 Title – Demonstration the issue documentation for OLTP Planning

Issue (comment)Eg – This task 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 millstone. The tasks that are counted for the milestone.

-[] create Data models

- Create Query Models (gives bullate dot)

- [] Track\_FetchTracksBy (gives check box)

-[] PlaylistTrack\_FetchPlaylist

-Create Commend Models

-[] PlaylistTrackTRXInput

-[] Create Transactional Services(methods)

-[] Add a Track

-[] Remove tracks from Playlist

-[]Reorganize playlist tracks

-[] Create front end webpage (little dot – convert to issue, it will become link to issue number 3) and you can write comment on it like other issue

- [] task 1

-[] task 2

Second commend box (you can found in tutorial for dummies)

Comment 1# space Expected transactional form (attached playlist management form)

(Another box)Comment 2 -# space Chinook ERD for Transactional Form (attached) (convert Word through snip and stretched tool)

CQRS – commend Query Responsibility Segregation

Register (rules)

1.data coming in - Commend classes

2.Data going out -Query classes

3.dta models (CQRS)

The data models are the c# classes that will be coded in the clas library project/application (folder.View Models) that holds the data classes in our solution.These classes represent the query class models and the commend class models(CQRS)

Attached CQRS summery image (do not put image in excersize)

---- (4 dashes solid line)

## Query models

### Artist and Album Tracks fetch(Track\_FetchTrackBy)

Type (back tick csharp‘’’csharp

Public class TrackSelection

{

Public int TrackId {get; set;}

Public string SongName {get; set;}

Public string AlbumTitle{get; set;}

Public string Milliseconds {get; set;}

Public decimal Price{get;set;}

}

‘’’

### Current PlaylistTrackInfo (playlist\_FetchPlaylist)

Type (back tick csharp‘’’Csharp

Public class TrackSelection

{

Public int TrackId {get; set;}

Public string Tracknumber {get; set;}

Public string SongName{get; set;}

Public string Milliseconds {get; set;}

}

‘’’

## command Models

### Add Tracks

\*\* NO model class, individual parameters\*\* (bolded)

### Remove Tracks

>space individual parameters for username (security) and Playliist name.Requires a list of trackids.List<int>

### Move Tracks (re-sequence playlist)

Type (back tick csharp‘’’csharp

Public class PlayTrackMove

{

Public int TrackId {get; set;}

Public string TrackNumber{get; set;}

Public bool SelectedTrack{get; set;}

Public string TrackInput {get; set;}

}

‘’’

For Data put into the database we have View model will have Query classes and commend classes

Query class (objects)send data out to your form

Playlist Management example (on moodle)

Query

Query does not belong to only 1 entity so we need anonymous data set bet we don’t use that in C# so we need Strongly typed dataset

**QUERY CLASSES**

**Public Class SongItem**

{

Public string song{get;set;}

Public string Album

Public string Artist

Public double length

Public decimal price

Public int TrackID

}

(TrackId (hidden) with the button)

Second table on the right

Trk#, Song, Length, hidden trackID – data coming Out from Database

Reorg Trk data coming in

2 query class

**Public class PlyaListItem**

{

Public int Track Number

Public string Song

Public double length

Public Int TrackID

}

**COMMEND CLASSES** (pass in playlist name and trackID)

Add Track

No class needed

Public Void AddTrack record (\_\_\_ , \_\_\_)

Remove Track (need playlist name and List<int>)

Move Tracks (need playlist ,

(more than one field coming in so need a class)

Public class reorgTrackItem

{

Public Int Track Id

Public Int current TrackNumber

Public Int Reorg TrackNumber

}

A no playlist – check if playlist exists (query)

If No

-create a new playlist

(staged) -Add to table(playlist)

-set track number = 1

B playlist

If Yes

* Check to see if track already on playlist (query)
* Yes(on) an error throw
* No (new)

-query for current Max (tracknumber)

(staged) -Add tracknumber

* Add the track to the playlist track
* Commit (save changes)

It could be new employee or employeeskills to the existing employee

In exercise only one transaction)

New connection so we can use entity frame work.

Add connection – entity framework core (multi- provider)

Add server name

Database name

…

…