Chinook PlayList.

# ERD for UI/UX process



The playlists will contain purchased tracks and non-purchased tracks. Only tracks that have been purchased would play (this is functionality of another subsystem).

The manage playlist OLTP sample will not include generation of Invoices for playlists. This would be another button (Purchase) on the Possible UI/UX Interface.

When creating your prototype, you can expect some changes as you create your system and review it with the client. A prototype is a starting point.

# Possible UX Interface



|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| ArtistArg (**1**) (TextBox) | | None / None |
| **BBL Class(es) and Method (s)** | | |
|  |  | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
|  | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| MediaTypeDDL (**2**) | | None / wired via ObjectDataSource |
| **BBL Class(es) and Method (s)** | | |
| MediaTypeController | List<SelectionList> List\_MediaTypeNames()  Retrieve a list of media type names and media type id for DDL | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| MediaTypes (R) | | SelectionList (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| GenreDDL (**3**) | | None / wired via ObjectDataSource |
| **BBL Class(es) and Method (s)** | | |
| GenreController | List<SelectionList> List\_GenreNames()  Retrieve a list of genre names and genre id for DDL | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Genres (R) | | SelectionList (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| AlbumArg (**4**) (TextBox) | | None / None |
| **BBL Class(es) and Method (s)** | | |
|  |  | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
|  | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Fetch (Artist, MediaType (autopostback), Genre, Album) (**5**),  Tracks (**6**), TracksBy (**7**), SearchArgID (**8**) | | OnClick / wired via ObjectDataSource   * Move selection value to TracksBy which identifies DDL/TextBox picked, (hidden field) * DDL/TextBox id to SearchArgID; (hidden field) * Bind data to Tracks |
| **BBL Class(es) and Method (s)** | | |
| TracksController | List<TrackList> List\_TracksForPlaylistSelection(string tracksby, string arg)  Retrieve a list of tracks for supplied selection type and argument strings. Return trackid, song, title, artist, media, genre, composer, timelength, size, unitprice. Display in Tracks. | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Tracks (R) Artist (R) Album (R) | | TrackList (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Fetch (**9**), Playlist (**10**), PlayListName (**11**) | | Fetch OnClick   * Validate data present; * call BLL; * Bind returned data to PlayList. |
| **BBL Class(es) and Method (s)** | | |
| PlaylistTracksController | List< UserPlaylistTrack > List\_TracksForPlaylist(string playlistname, string username)  Retrieve a list of tracks for supplied playlist name and username. Return trackid, title, timelength, unitprice, tracknumber. Display in PlayList. | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| PlayList (R), PlaylistTracks(R), Track (R) | | UserPlaylistTrack (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Tracks: Plus Sign (**6**), PlayListName (**11**) | | OnCommand   * Collect track information (trackid) and playlist name and validate; * send to BLL for adding track to playlist; * refresh playlist. |
| **BBL Class(es) and Method (s)** | | |
| PlaylistTracksController | void Add\_TrackToPlaylist(string playlistname, string username, int trackid)  TRX   * create playlist if needed, set tracknumber to 1, * playlist exists, find current highest tracknumber to increment by 1 * verify desired track not already on list, if so reject * add new track to playlist tracks | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| PlayList(C opt), PlaylistTracks(C) | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Up/Down (**12**), PlayListName (**11**), PlayList (**10**) | | OnCLick   * Collect track id, track number, playlist name and validate (only 1 track can move); * send to BLL for moving track in playlist; * refresh playlist. |
| **BBL Class(es) and Method (s)** | | |
| PlaylistTracksController | void Move\_TrackInPlaylist(string playlistname, string username, int trackid, ~~int tracknumber~~, string direction)  TRX(determine if move necessary, swap tracks depending on direction)   * if already a top and moving up, reject move * if already a bottom and moving down, reject move * move up, swap with track above by altering tracknumbers * move down, swap with track below by altering tracknumbers. | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Playlist(R), PlaylistTracks(U) | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Delete (**12**), PlayListName (**11**), PlayList (**10**) | | OnCLick   * Collect track ids, ~~track number~~ and playlist name and validate (at least one track selected); * Confirmation * send to BLL for removing track from playlist; * refresh playlist. |
| **BBL Class(es) and Method (s)** | | |
| PlaylistTracksController | void Remove\_TrackFromPlaylist(string playlistname, string username, List<int> trackstodelete ~~, List<int> tracknumber~~)  TRX   * find tracks, * remove tracks, * shift remaining tracknumbers up by renumbering tracks. | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Playlist(R), PlaylistTracks(U/D) | |  |

