WebradioManager

1.0

Generated by Doxygen 1.8.5

Tue May 27 2014 15:38:59

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Package WebradioManager

Classes

- class Ad

  An ad audio file.
- class AdminController

  A controller for handling admin view.
- class AdminView

  An admin view.
- class AudioFile

  An audio file. Abstract class.
- class Bdd

  A bdd connection.
- class BddControls
- class CalendarEvent

  A calendar event.
- struct DayWeek

  A week with boolean values for each day. Uses for store which day is selected for an CalendarEvent.
- class EventAppointment

  An event appointment. Add 2 properties to the original Appointment class (from Calendar library).
- interface IController

  Interface for controller.
- class Music

  A music.
- class Playlist

  A playlist. Abstract class.
- class PlaylistAd

  A playlist ad.
- class PlaylistMusic

  A playlist music.
- class **Program**
- class SelectionController

  A controller for SelectionView.
- class SelectionView

  A selection view.

- class TranscoderAacPlus

  A transcoder aac plus.
- class TranscoderMp3

  A transcoder mp3.
- class Webradio

  A webradio.
- class WebradioCalendar

  A webradio calendar.
- class WebradioListener

  A webradio listener.
- class WebradioServer

  A shoutcast webradio server.
- class WebradioServerStats

  A webradio server statistics.
- class WebradioTranscoder

  A webradio transcoder.
- class WMModel

  A data Model for the WebradioManager project.

Enumerations

- enum AudioType { **Music** = 2, **Ad** = 1 }

  Values that represent AudioType's id. Defined in DB.
- enum DayValue {
  Monday = 2, **Tuesday** = 4, **Wednesday** = 8, **Thursday** = 16,
  Friday = 32, **Saturday** = 64, **Sunday** = 1 }

  Values that represent DayValue for calendar event. http://wiki.winamp.com/wiki/SHOUTcast_-Calendar_Event_XML_File_Specification#Calendar_Tag.
- enum StreamType { **MP3** = 1, **AACPlus** = 2 }

  Values that represent StreamType's id. Defined in DB.

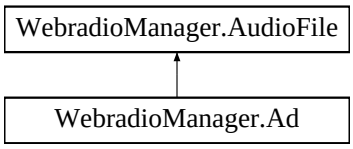
Chapter 6

Class Documentation

6.1 WebradioManager.Ad Class Reference

An ad audio file.

Inheritance diagram for WebradioManager.Ad:



Public Member Functions

- Ad (int id, string filename, string title, string artist, string album, int year, string label, TimeSpan duration, string gender)

  Constructor.
- Ad (string filename, string title, string artist, string album, int year, string label, TimeSpan duration, string gender)

  Constructor.

Additional Inherited Members

6.1.1 Detailed Description

An ad audio file.

Author

Simon Menetrey

Date

23.05.2014

Definition at line 20 of file Ad.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 public WebradioManager.Ad.Ad (int *id*, string *filename*, string *title*, string *artist*, string *album*, int *year*, string *label*, TimeSpan *duration*, string *gender*)

Constructor.

Author

Simon Menetrey

Date

23.05.2014

Parameters

id	The identifier.
filename	Filename of the audio file.
title	The title.
artist	The artist.
album	The album.
year	The year.
label	The label.
duration	The duration.
gender	The gender.

Definition at line 42 of file Ad.cs.

6.1.2.2 public WebradioManager.Ad.Ad (string *filename*, string *title*, string *artist*, string *album*, int *year*, string *label*, TimeSpan *duration*, string *gender*)

Constructor.

Author

Simon Menetrey

Date

23.05.2014

Parameters

filename	Filename of the audio file.
title	The title.
artist	The artist.
album	The album.
year	The year.
label	The label.
duration	The duration.
gender	The gender.

Definition at line 66 of file Ad.cs.

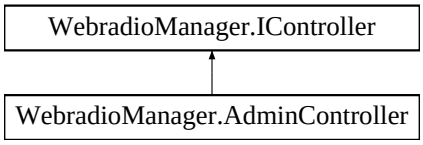
The documentation for this class was generated from the following file:

- Ad.cs

6.2 WebradioManager.AdminController Class Reference

A controller for handling admin view.

Inheritance diagram for WebradioManager.AdminController:



Public Member Functions

- AdminController (int id, WMModel model)

  Constructor.
- void UpdateView ()

  Updates the view.
- Webradio GetWebradio (int id)

  Gets a webradio.
- List< AudioFile > GetLibrary ()

  Gets the library.
- List< string > GetGenders ()

  Gets the genders.
- void CheckFolders (int webradioId)

  Check folders.
- void FormClose ()

  Form close. Remove observer.
- bool ImportFilesToLibrary (string[] filenames, AudioType type)

  Import files to library.
- bool DeleteAudioFile (int id, string filename)

  Deletes the audio file.
- bool CreatePlaylist (string name, string webradioName, int webradioId, AudioType type)

  Creates a playlist.
- bool DeletePlaylist (Playlist playlist, int webradioId)

  Deletes the playlist.
- bool AddToPlaylist (Playlist playlist, Dictionary< int, string > audioFiles)

  Adds to the playlist.
- bool RemoveFromPlaylist (Playlist playlist, Dictionary< int, string > audioFiles)

  Removes from playlist.
- List< AudioFile > GetPlaylistContent (Playlist playlist)

  Gets playlist content.
- bool GeneratePlaylist (string name, TimeSpan duration, AudioType type, string gender, int webradioId, string webradioName)

  Generates a playlist.
- bool CreateEvent (CalendarEvent newEvent, int webradioId)

  Creates an event.
- bool UpdateEvent (CalendarEvent aEvent, int webradioId)

  Updates the event.
- bool DeleteEvent (CalendarEvent aEvent, int webradioId)

  Deletes the event.

- bool CreateTranscoder (string name, StreamType st, int sampleRate, int bitrate, string url, IPAddress ip, int port, int adminport, string password, int webradioId)

  Creates a transcoder.
- bool DeleteTranscoder (WebradioTranscoder transcoder, int webradioId)

  Deletes the transcoder.
- bool UpdateTranscoder (WebradioTranscoder transcoder, bool debug, int webradioId)

  Updates the transcoder.
- bool StartTranscoder (WebradioTranscoder transcoder, bool debug, int webradioId)

  Starts a transcoder.
- bool StopTranscoder (WebradioTranscoder transcoder, int webradioId)

  Stops a transcoder.
- bool StopAllTranscoders (int webradioId)

  Stops all transcoders.
- void GenerateAllConfigs (int webradioId)

  Generates all configs.
- bool UpdateServer (bool debug, int port, string password, string adminPassword, int maxListener, int webradioId)

  Updates the server.
- bool StartServer (int webradioId, bool debug)

  Starts a server.
- bool StopServer (int webradioId)

  Stops a server.
- void ShowServerWebInterface (int webradioId)

  Shows the server web interface.
- void ShowServerWebAdmin (int webradioId)

  Shows the server web admin.
- bool TranscoderNextTrack (WebradioTranscoder transcoder)

  Transcoder next track.
- bool ClearHistory (int transcoderId)

  Clears the transcoder's history.
- bool GenerateHistory (int webradioId, string transcoderName, int transcoderId, string outputFilename)

  Generates a history.
- bool ModifyWebradioName (string name, int webradioId)

  Modify webradio's name.
- AudioFile GetAudioFileByFilename (string filename)

  Gets audio file by filename.
- bool UpdateAudioFile (AudioFile file)

  Updates the audio file described by file.
- bool TranscoderCapture (bool active, string device, WebradioTranscoder transcoder, int webradioId)

  Transcoder capture mode set.
- List< WebradioListener > GetServerListeners (int webradioId)

  Get server's listeners.
- bool UpdateServerStats (int webradioId)

  Updates the server statistics.
- bool CheckLibrary ()

  Check library integrity.
- int GetSimilarViewCount (int webradioId)

  Gets similar view count.

Properties

- AdminView View `[get, set]`
  Gets or sets the view.
- WMModel Model `[get, set]`
  Gets or sets the model.

6.2.1 Detailed Description

A controller for handling admin view.

Author

Simon Menetrey

Date

23.05.2014

Definition at line 22 of file AdminController.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 public WebradioManager.AdminController.AdminController (int *id*, WMModel *model*)

Constructor.

Author

Simon Menetrey

Date

23.05.2014

Parameters

id	The webradio identifier.
model	The model.

Definition at line 76 of file AdminController.cs.

6.2.3 Member Function Documentation

6.2.3.1 public bool WebradioManager.AdminController.AddToPlaylist (Playlist *playlist*, Dictionary< int, string > *audioFiles*)

Adds to the playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

playlist	The playlist.
audioFiles	The audio files.

Returns

true if it succeeds, false if it fails.

Definition at line 271 of file AdminController.cs.

6.2.3.2 public void WebradioManager.AdminController.CheckFolders (int *webradioId*)

Check folders.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Definition at line 159 of file AdminController.cs.

6.2.3.3 public bool WebradioManager.AdminController.CheckLibrary ( )

Check library integrity.

Author

Simon Menetrey

Date

23.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 807 of file AdminController.cs.

6.2.3.4 public bool WebradioManager.AdminController.ClearHistory (int *transcoderId*)

Clears the transcoder's history.

Author

Simon Menetrey

Date

23.05.2014

Parameters

transcoderId	Identifier for the transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 658 of file AdminController.cs.

6.2.3.5 public bool WebradioManager.AdminController.CreateEvent (CalendarEvent *newEvent,* int *webradioId*)

Creates an event.

Author

Simon Menetrey

Date

23.05.2014

Parameters

newEvent	The new event.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 353 of file AdminController.cs.

6.2.3.6 public bool WebradioManager.AdminController.CreatePlaylist (string *name,* string *webradioName,* int *webradioId,* AudioType *type*)

Creates a playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

name	The playlis's name.
webradioName	Name of the webradio.
webradioId	Identifier of the webradio.
type	The type.

Returns

true if it succeeds, false if it fails.

Definition at line 232 of file AdminController.cs.

6.2.3.7 public bool WebradioManager.AdminController.CreateTranscoder (string *name,* StreamType *st,* int *sampleRate,* int *bitrate,* string *url,* IPAddress *ip,* int *port,* int *adminport,* string *password,* int *webradioId*)

Creates a transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

name	The name.
st	The streamtype.
sampleRate	The sample rate.
bitrate	The bitrate.
url	URL of the stream.
ip	The IP.
port	The port.
adminport	The adminport.
password	The password.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 418 of file AdminController.cs.

6.2.3.8 public bool WebradioManager.AdminController.DeleteAudioFile (int *id,* string *filename*)

Deletes the audio file.

Author

Simon Menetrey

Date

23.05.2014

Parameters

id	The identifier.
filename	Filename of the file.

Returns

true if it succeeds, false if it fails.

Definition at line 211 of file AdminController.cs.

6.2.3.9 public bool WebradioManager.AdminController.DeleteEvent (CalendarEvent *aEvent,* int *webradioId*)

Deletes the event.

Author

Simon Menetrey

Date

23.05.2014

Parameters

aEvent	The event.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 391 of file AdminController.cs.

6.2.3.10 public bool WebradioManager.AdminController.DeletePlaylist (Playlist *playlist,* int *webradioId*)

Deletes the playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

playlist	The playlist.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 252 of file AdminController.cs.

6.2.3.11 public bool WebradioManager.AdminController.DeleteTranscoder (WebradioTranscoder *transcoder,* int *webradioId*)

Deletes the transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

transcoder	The transcoder.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 437 of file AdminController.cs.

6.2.3.12 public void WebradioManager.AdminController.FormClose ( )

Form close. Remove observer.

Author

Simon Menetrey

Date

23.05.2014

Definition at line 173 of file AdminController.cs.

6.2.3.13 public void WebradioManager.AdminController.GenerateAllConfigs (int *webradioId*)

Generates all configs.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Definition at line 530 of file AdminController.cs.

6.2.3.14 public bool WebradioManager.AdminController.GenerateHistory (int *webradioId,* string *transcoderName,* int *transcoderId,* string *outputFilename*)

Generates a history.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.
transcoderName	Name of the transcoder.
transcoderId	Identifier of the transcoder.
outputFilename	Filename of the output file.

Returns

true if it succeeds, false if it fails.

Definition at line 679 of file AdminController.cs.

6.2.3.15 public bool WebradioManager.AdminController.GeneratePlaylist (string *name*, TimeSpan *duration*, AudioType *type*, string *gender*, int *webradioId*, string *webradioName*)

Generates a playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

name	The name.
duration	The duration.
type	The type.
gender	The gender.
webradioId	Identifier of the webradio.
webradioName	Name of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 334 of file AdminController.cs.

6.2.3.16 public AudioFile WebradioManager.AdminController.GetAudioFileByFilename (string *filename*)

Gets audio file by filename.

Author

Simon Menetrey

Date

23.05.2014

Parameters

filename	Filename of the file.

Returns

The audio file by filename.

Definition at line 716 of file AdminController.cs.

6.2.3.17 public List< string > WebradioManager.AdminController.GetGenders ( )

Gets the genders.

Author

Simon Menetrey

Date

23.05.2014

Returns

The genders.

Definition at line 143 of file AdminController.cs.

6.2.3.18 public List< AudioFile > WebradioManager.AdminController.GetLibrary ( )

Gets the library.

Author

Simon Menetrey

Date

23.05.2014

Returns

The library.

Definition at line 127 of file AdminController.cs.

6.2.3.19 public List< AudioFile > WebradioManager.AdminController.GetPlaylistContent (Playlist *playlist*)

Gets playlist content.

Author

Simon Menetrey

Date

23.05.2014

Parameters

playlist	The playlist.

Returns

The playlist content.

Definition at line 311 of file AdminController.cs.

6.2.3.20 public List< WebradioListener > WebradioManager.AdminController.GetServerListeners (int *webradioId*)

Get server's listeners.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Returns

A List<WebradioListener>

Definition at line 773 of file AdminController.cs.

6.2.3.21 public int WebradioManager.AdminController.GetSimilarViewCount (int *webradioId*)

Gets similar view count.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Returns

The similar view count.

Definition at line 825 of file AdminController.cs.

6.2.3.22 public Webradio WebradioManager.AdminController.GetWebradio (int *id*)

Gets a webradio.

Author

Simon Menetrey

Date

23.05.2014

Parameters

id	The webradio identifier.

Returns

The webradio.

Definition at line 111 of file AdminController.cs.

6.2.3.23 public bool WebradioManager.AdminController.ImportFilesToLibrary (string[] *filenames,* AudioType *type*)

Import files to library.

Author

Simon Menetrey

Date

23.05.2014

Parameters

filenames	The filenames.
type	The type.

Returns

true if it succeeds, false if it fails.

Definition at line 192 of file AdminController.cs.

6.2.3.24 public bool WebradioManager.AdminController.ModifyWebradioName (string *name,* int *webradioId*)

Modify webradio's name.

Author

Simon Menetrey

Date

23.05.2014

Parameters

name	The name.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 698 of file AdminController.cs.

6.2.3.25 public bool WebradioManager.AdminController.RemoveFromPlaylist (Playlist *playlist,* Dictionary< int, string > *audioFiles*)

Removes from playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

playlist	The playlist.
audioFiles	The audio files.

Returns

true if it succeeds, false if it fails.

Definition at line 293 of file AdminController.cs.

6.2.3.26 public void WebradioManager.AdminController.ShowServerWebAdmin (int *webradioId*)

Shows the server web admin.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Definition at line 622 of file AdminController.cs.

6.2.3.27 public void WebradioManager.AdminController.ShowServerWebInterface (int *webradioId*)

Shows the server web interface.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Definition at line 606 of file AdminController.cs.

6.2.3.28 public bool WebradioManager.AdminController.StartServer (int *webradioId,* bool *debug*)

Starts a server.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.
debug	Debug or not.

Returns

true if it succeeds, false if it fails.

Definition at line 572 of file AdminController.cs.

6.2.3.29 public bool WebradioManager.AdminController.StartTranscoder (WebradioTranscoder *transcoder,* bool *debug,* int *webradioId*)

Starts a transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

transcoder	The transcoder.
debug	Debug or not.

webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 477 of file AdminController.cs.

6.2.3.30 public bool WebradioManager.AdminController.StopAllTranscoders (int *webradioId*)

Stops all transcoders.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 514 of file AdminController.cs.

6.2.3.31 public bool WebradioManager.AdminController.StopServer (int *webradioId*)

Stops a server.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 590 of file AdminController.cs.

6.2.3.32 public bool WebradioManager.AdminController.StopTranscoder (WebradioTranscoder *transcoder,* int *webradioId*)

Stops a transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

transcoder	The transcoder.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 496 of file AdminController.cs.

6.2.3.33 public bool WebradioManager.AdminController.TranscoderCapture (bool *active,* string *device,* WebradioTranscoder *transcoder,* int *webradioId*)

Transcoder capture mode set.

Author

Simon Menetrey

Date

23.05.2014

Parameters

active	true to active.
device	The device's name.
transcoder	The transcoder.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 755 of file AdminController.cs.

6.2.3.34 public bool WebradioManager.AdminController.TranscoderNextTrack (WebradioTranscoder *transcoder*)

Transcoder next track.

Author

Simon Menetrey

Date

23.05.2014

Parameters

transcoder	The transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 640 of file AdminController.cs.

6.2.3.35 public bool WebradioManager.AdminController.UpdateAudioFile (AudioFile *file*)

Updates the audio file described by file.

Author

Simon Menetrey

Date

23.05.2014

Parameters

file	The file.

Returns

true if it succeeds, false if it fails.

Definition at line 734 of file AdminController.cs.

6.2.3.36 public bool WebradioManager.AdminController.UpdateEvent (CalendarEvent *aEvent,* int *webradioId*)

Updates the event.

Author

Simon Menetrey

Date

23.05.2014

Parameters

aEvent	The event.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 372 of file AdminController.cs.

6.2.3.37 public bool WebradioManager.AdminController.UpdateServer (bool *debug*, int *port*, string *password*, string *adminPassword*, int *maxListener*, int *webradioId*)

Updates the server.

Author

Simon Menetrey

Date

23.05.2014

Parameters

debug	Debug or not.
port	The port.
password	The password.
adminPassword	The admin password.
maxListener	The maximum listener.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 553 of file AdminController.cs.

6.2.3.38 public bool WebradioManager.AdminController.UpdateServerStats (int *webradioId*)

Updates the server statistics.

Author

Simon Menetrey

Date

23.05.2014

Parameters

webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 791 of file AdminController.cs.

6.2.3.39 public bool WebradioManager.AdminController.UpdateTranscoder (WebradioTranscoder *transcoder*, bool *debug*, int *webradioId*)

Updates the transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

transcoder	The transcoder.
debug	Debug or not.
webradioId	Identifier of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 457 of file AdminController.cs.

6.2.3.40 public void WebradioManager.AdminController.UpdateView ( )

Updates the view.

Author

Simon Menetrey

Date

23.05.2014

Implements WebradioManager.IController.

Definition at line 93 of file AdminController.cs.

6.2.4 Property Documentation

6.2.4.1 public WMModel WebradioManager.AdminController.Model [get], [set]

Gets or sets the model.

Returns

The model.

Definition at line 56 of file AdminController.cs.

6.2.4.2 public AdminView WebradioManager.AdminController.View [get], [set]

Gets or sets the view.

Returns

The view.

Definition at line 42 of file AdminController.cs.

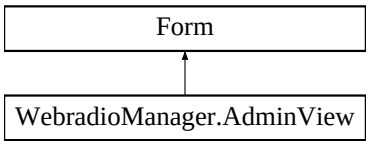
The documentation for this class was generated from the following file:

- AdminController.cs

6.3 WebradioManager.AdminView Class Reference

An admin view.

Inheritance diagram for WebradioManager.AdminView:



Public Member Functions

- AdminView (int idWebradio, AdminController controller)
  Constructor.
- void UpdateView ()
  Updates the view.

Properties

- string NameWebradio `[get, set]`
  Gets or sets the webradio's name.
- List< EventAppointment > EventsCalendar `[get, set]`
  Gets or sets the events calendar.
- int IdWebradio `[get, set]`
  Gets or sets the identifier of the current webradio.
- AdminController Controller `[get, set]`
  Gets or sets the controller.

6.3.1 Detailed Description

An admin view.

Author

Simon Menetrey

Date

23.05.2014

Definition at line 29 of file AdminView.cs.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 public WebradioManager.AdminView.AdminView (int *idWebradio*, AdminController *controller*)

Constructor.

Author

Simon Menetrey

Date

23.05.2014

Parameters

idWebradio	The webradio's identifier.
controller	The controller.

Definition at line 121 of file AdminView.cs.

6.3.3 Member Function Documentation

6.3.3.1 public void WebradioManager.AdminView.UpdateView ( )

Updates the view.

Author

Simon Menetrey

Date

23.05.2014

Definition at line 139 of file AdminView.cs.

6.3.4 Property Documentation

6.3.4.1 public AdminController WebradioManager.AdminView.Controller [get], [set]

Gets or sets the controller.

Returns

The controller.

Definition at line 102 of file AdminView.cs.

6.3.4.2 public List< EventAppointment > WebradioManager.AdminView.EventsCalendar [get], [set]

Gets or sets the events calendar.

Returns

The events calendar.

Definition at line 74 of file AdminView.cs.

6.3.4.3 public int WebradioManager.AdminView.IdWebradio [get], [set]

Gets or sets the identifier of the current webradio.

Returns

The identifier webradio.

Definition at line 88 of file AdminView.cs.

6.3.4.4 public string WebradioManager.AdminView.NameWebradio [get], [set]

Gets or sets the webradio's name.

Returns

The name webradio.

Definition at line 60 of file AdminView.cs.

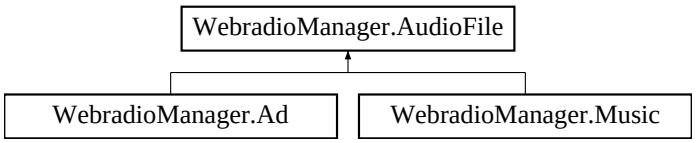
The documentation for this class was generated from the following file:

- AdminView.cs

6.4 WebradioManager.AudioFile Class Reference

An audio file. Abstract class.

Inheritance diagram for WebradioManager.AudioFile:



Public Member Functions

- AudioFile (int id, string filename, string title, string artist, string album, int year, string label, TimeSpan duration, string gender, AudioType audiotype)

  Constructor.
- AudioFile (string filename, string title, string artist, string album, int year, string label, TimeSpan duration, string gender, AudioType audiotype)

  Constructor.
- string[] **GetInfosArray** ()

Properties

- int Id [get, set]

  Gets or sets the identifier.
- string Filename [get, set]

  Gets or sets the filename of the audiofile.
- string Title [get, set]

  Gets or sets the title.
- string Artist [get, set]

  Gets or sets the artist.
- string Album [get, set]

  Gets or sets the album.
- int Year [get, set]

  Gets or sets the year.
- string Label [get, set]

  Gets or sets the label.
- TimeSpan Duration [get, set]

Gets or sets the duration.

- string Gender `[get, set]`

  Gets or sets the gender.

- AudioType Type `[get, set]`

  Gets or sets the type.

6.4.1 Detailed Description

An audio file. Abstract class.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file AudioFile.cs.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 public WebradioManager.AudioFile.AudioFile (int *id*, string *filename*, string *title*, string *artist*, string *album*, int *year*, string *label*, TimeSpan *duration*, string *gender*, AudioType *audiotype*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
filename	Filename of the audiofile.
title	The title.
artist	The artist.
album	The album.
year	The year.
label	The label.
duration	The duration.
gender	The gender.
audiotype	The audiotype.

Definition at line 218 of file AudioFile.cs.

6.4.2.2 public WebradioManager.AudioFile.AudioFile (string *filename*, string *title*, string *artist*, string *album*, int *year*, string *label*, TimeSpan *duration*, string *gender*, AudioType *audiotype*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

filename	Filename of the audiofile.
title	The title.
artist	The artist.
album	The album.
year	The year.
label	The label.
duration	The duration.
gender	The gender.
audiotype	The audiotype.

Definition at line 251 of file AudioFile.cs.

6.4.3 Property Documentation

6.4.3.1 public string WebradioManager.AudioFile.Album [get], [set]

Gets or sets the album.

Returns

The album.

Definition at line 119 of file AudioFile.cs.

6.4.3.2 public string WebradioManager.AudioFile.Artist [get], [set]

Gets or sets the artist.

Returns

The artist.

Definition at line 105 of file AudioFile.cs.

6.4.3.3 public TimeSpan WebradioManager.AudioFile.Duration [get], [set]

Gets or sets the duration.

Returns

The duration.

Definition at line 161 of file AudioFile.cs.

6.4.3.4 public string WebradioManager.AudioFile.Filename [get], [set]

Gets or sets the filename of the audiofile.

Returns

The filename.

Definition at line 77 of file AudioFile.cs.

6.4.3.5 public string WebradioManager.AudioFile.Gender [get], [set]

Gets or sets the gender.

Returns

The gender.

Definition at line 175 of file AudioFile.cs.

6.4.3.6 public int WebradioManager.AudioFile.Id [get], [set]

Gets or sets the identifier.

Returns

The identifier.

Definition at line 63 of file AudioFile.cs.

6.4.3.7 public string WebradioManager.AudioFile.Label [get], [set]

Gets or sets the label.

Returns

The label.

Definition at line 147 of file AudioFile.cs.

6.4.3.8 public string WebradioManager.AudioFile.Title [get], [set]

Gets or sets the title.

Returns

The title.

Definition at line 91 of file AudioFile.cs.

6.4.3.9 public AudioType WebradioManager.AudioFile.Type [get], [set]

Gets or sets the type.

Returns

The type.

Definition at line 189 of file AudioFile.cs.

6.4.3.10 public int WebradioManager.AudioFile.Year [get], [set]

Gets or sets the year.

Returns

The year.

Definition at line 133 of file AudioFile.cs.

The documentation for this class was generated from the following file:

- AudioFile.cs

6.5 WebradioManager.Bdd Class Reference

A bdd connection.

Public Member Functions

- Bdd ()
  Default constructor.
- Dictionary< int, Webradio > LoadWebradios ()
  Loads the webradios from db. All webradio's elements are loaded too.
- List< AudioFile > LoadLibrary ()
  Loads the library from db.
- int AddWebradio (Webradio webradio)
  Adds a webradio to db. Create server and calendar entries for this new webradio.
- bool WebradioExist (string name)
  Webradio exists in the db.
- bool DeleteWebradio (int id)
  Deletes the webradio described by ID.
- List< string > GetGenders ()
  Gets genders list.
- int GetGenderId (string gender)
  Gets gender identifier.
- int AddGender (string gender)
  Adds a gender.
- int AddAudioFile (AudioFile file)
  Adds an audio file.
- bool UpdateAudioFile (AudioFile file) hh\:mm\:ss"
  Updates the audio file's values.
- bool AudioFileExist (string filename)
  Audio file exists.
- bool DeleteAudioFile (int id)
  Deletes the audio file described by ID.
- int CreatePlaylist (Playlist playlist, int webradioId)
  Creates a playlist.
- bool DeletePlaylist (int id)
  Deletes the playlist described by ID.
- bool AddToPlaylist (int idAudioFile, int idPlaylist)
  Adds an audio file to a playlist.

- bool RemoveFromPlaylist (int idAudioFile, int idPlaylist)

  Removes an audio file from a playlist.
- int AddGeneratedPlaylist (Playlist playlist, List< int > audioFilesId, int webradioId)

  Adds a generated playlist.
- int AddEvent (CalendarEvent newEvent, int calendarId, int playlistId)

  Adds an event to a calendar.
- bool EventExist (CalendarEvent aEvent, int calendarId)

  Event exists in the db.
- bool UpdateEvent (CalendarEvent aEvent)

  Updates the event's value.
- bool DeleteEvent (CalendarEvent aEvent)

  Deletes the event described by aEvent.
- bool TranscoderExist (string name, int webradioId)

  Transcoder exists.
- int AddTranscoder (WebradioTranscoder transcoder, int webradioId)

  Adds a transcoder to a webradio.
- bool DeleteTranscoder (int transcoderId)

  Deletes the transcoder described by transcoder's id.
- bool UpdateTranscoder (WebradioTranscoder transcoder)

  Updates the transcoder with transcoder param's values.
- bool UpdateServer (int port, string password, string adminPassword, int maxListener, int webradioId)

  Updates the server configuration.
- bool AddToHistory (int transcoderId, DateTime date, string filename)

  Adds audiofile's filename to the transcoder's history.
- Dictionary< string, string > GetHistory (int transcoderId)

  Gets a transcoder's history.
- bool ClearHistory (int transcoderId)

  Clears the transcoder's history.
- bool ModifyWebradioName (string name, int webradioId)

  Modify webradio's name.
- bool UpdateFilenames (string oldName, string newName, Webradio webradio)

  Updates the filenames with new webradio's name.

Public Attributes

- const int **ERROR** = -1

Properties

- BddControls Controls `[get, set]`

  Gets or sets the bdd controls.

6.5.1 Detailed Description

A bdd connection.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 23 of file Bdd.cs.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 public WebradioManager.Bdd.Bdd ( )

Default constructor.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 63 of file Bdd.cs.

6.5.3 Member Function Documentation

6.5.3.1 public int WebradioManager.Bdd.AddAudioFile (AudioFile *file*)

Adds an audio file.

Author

Simon Menetrey

Date

26.05.2014

Parameters

file	The audiofile.

Returns

The new audio file's id.

Definition at line 447 of file Bdd.cs.

6.5.3.2 public int WebradioManager.Bdd.AddEvent (CalendarEvent *newEvent,* int *calendarId,* int *playlistId*)

Adds an event to a calendar.

Author

Simon Menetrey

Date

26.05.2014

Parameters

newEvent	The new event.
calendarId	Identifier for the calendar.
playlistId	Identifier for the playlist.

Returns

The new event's id.

Definition at line 754 of file Bdd.cs.

6.5.3.3 public int WebradioManager.Bdd.AddGender (string *gender*)

Adds a gender.

Author

Simon Menetrey

Date

26.05.2014

Parameters

gender	The gender's name.

Returns

The new gender's id.

Definition at line 421 of file Bdd.cs.

6.5.3.4 public int WebradioManager.Bdd.AddGeneratedPlaylist (Playlist *playlist*, List< int > *audioFilesId*, int *webradioId*)

Adds a generated playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

playlist	The playlist.
audioFilesId	Identifier for the audio files.
webradioId	Identifier for the webradio.

Returns

AThe new generated playlist's id.

Definition at line 728 of file Bdd.cs.

6.5.3.5 public bool WebradioManager.Bdd.AddToHistory (int *transcoderId,* DateTime *date,* string *filename*)

Adds audiofile's filename to the transcoder's history.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoderId	Identifier for the transcoder.
date	The date Date/Time.
filename	Filename of the audiofile.

Returns

true if it succeeds, false if it fails.

Definition at line 1051 of file Bdd.cs.

6.5.3.6 public bool WebradioManager.Bdd.AddToPlaylist (int *idAudioFile,* int *idPlaylist*)

Adds an audio file to a playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

idAudioFile	The identifier audio file.
idPlaylist	The identifier playlist.

Returns

true if it succeeds, false if it fails.

Definition at line 647 of file Bdd.cs.

6.5.3.7 public int WebradioManager.Bdd.AddTranscoder (WebradioTranscoder *transcoder,* int *webradioId*)

Adds a transcoder to a webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.
webradioId	Identifier for the webradio.

Returns

The new transcoder's id.

Definition at line 900 of file Bdd.cs.

6.5.3.8 public int WebradioManager.Bdd.AddWebradio (Webradio *webradio*)

Adds a webradio to db. Create server and calendar entries for this new webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradio	The webradio.

Returns

Id of created webradio. or ERROR.

Definition at line 257 of file Bdd.cs.

6.5.3.9 public bool WebradioManager.Bdd.AudioFileExist (string *filename*)

Audio file exists.

Author

Simon Menetrey

Date

26.05.2014

Parameters

filename	Filename of the audio file.

Returns

true if it exists, false if it doesn't exist.

Definition at line 530 of file Bdd.cs.

6.5.3.10 public bool WebradioManager.Bdd.ClearHistory (int *transcoderId*)

Clears the transcoder's history.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoderId	Identifier for the transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 1103 of file Bdd.cs.

6.5.3.11 public int WebradioManager.Bdd.CreatePlaylist (Playlist *playlist,* int *webradioid*)

Creates a playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

playlist	The playlist.
webradioid	The webradio's id.

Returns

The new playlist's id or error code.

Definition at line 588 of file Bdd.cs.

6.5.3.12 public bool WebradioManager.Bdd.DeleteAudioFile (int *id*)

Deletes the audio file described by ID.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 560 of file Bdd.cs.

6.5.3.13 public bool WebradioManager.Bdd.DeleteEvent (CalendarEvent *aEvent*)

Deletes the event described by aEvent.

Author

Simon Menetrey

Date

26.05.2014

Parameters

aEvent	The event.

Returns

true if it succeeds, false if it fails.

Definition at line 843 of file Bdd.cs.

6.5.3.14 public bool WebradioManager.Bdd.DeletePlaylist (int *id*)

Deletes the playlist described by ID.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 620 of file Bdd.cs.

6.5.3.15 public bool WebradioManager.Bdd.DeleteTranscoder (int *transcoderId*)

Deletes the transcoder described by transcoder's id.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoderId	Identifier for the transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 951 of file Bdd.cs.

6.5.3.16 public bool WebradioManager.Bdd.DeleteWebradio (int *id*)

Deletes the webradio described by ID.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 346 of file Bdd.cs.

6.5.3.17 public bool WebradioManager.Bdd.EventExist (CalendarEvent *aEvent,* int *calendarId*)

Event exists in the db.

Author

Simon Menetrey

Date

26.05.2014

Parameters

aEvent	The event.
calendarId	Identifier for the calendar.

Returns

true if it exists, false if it doesn't exists.

Definition at line 790 of file Bdd.cs.

6.5.3.18 public int WebradioManager.Bdd.GetGenderId (string *gender*)

Gets gender identifier.

Author

Simon Menetrey

Date

26.05.2014

Parameters

gender	The gender's name.

Returns

The gender identifier.

Definition at line 395 of file Bdd.cs.

6.5.3.19 public List< string > WebradioManager.Bdd.GetGenders ( )

Gets genders list.

Author

Simon Menetrey

Date

26.05.2014

Returns

The genders list.

Definition at line 370 of file Bdd.cs.

6.5.3.20 public Dictionary< string, string > WebradioManager.Bdd.GetHistory (int *transcoderId*)

Gets a transcoder's history.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoderId	Identifier for the transcoder.

Returns

The history (date,filename).

Definition at line 1078 of file Bdd.cs.

6.5.3.21 public List< AudioFile > WebradioManager.Bdd.LoadLibrary ( )

Loads the library from db.

Author

Simon Menetrey

Date

26.05.2014

Returns

The library list.

Definition at line 206 of file Bdd.cs.

6.5.3.22 public Dictionary< int, Webradio > WebradioManager.Bdd.LoadWebradios ( )

Loads the webradios from db. All webradio's elements are loaded too.

Author

Simon Menetrey

Date

26.05.2014

Returns

The webradios list.

Definition at line 79 of file Bdd.cs.

6.5.3.23 public bool WebradioManager.Bdd.ModifyWebradioName (string *name,* int *webradioId*)

Modify webradio's name.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails or already exist.

Definition at line 1130 of file Bdd.cs.

6.5.3.24 public bool WebradioManager.Bdd.RemoveFromPlaylist (int *idAudioFile,* int *idPlaylist*)

Removes an audio file from a playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

idAudioFile	The identifier audio file.
idPlaylist	The identifier playlist.

Returns

true if it succeeds, false if it fails.

Definition at line 677 of file Bdd.cs.

6.5.3.25 public bool WebradioManager.Bdd.TranscoderExist (string *name,* int *webradioId*)

Transcoder exists.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
webradioId	Identifier for the webradio.

Returns

true if it exists, false if it doesn't exist.

Definition at line 870 of file Bdd.cs.

6.5.3.26 public bool WebradioManager.Bdd.UpdateAudioFile (AudioFile *file*) hh\:mm\:ss"

Updates the audio file's values.

Author

Simon Menetrey

Date

26.05.2014

Parameters

file	The audio file.

Returns

true if it succeeds, false if it fails.

Definition at line 491 of file Bdd.cs.

6.5.3.27 public bool WebradioManager.Bdd.UpdateEvent (CalendarEvent *aEvent*)

Updates the event's value.

Author

Simon Menetrey

Date

26.05.2014

Parameters

aEvent	The event.

Returns

true if it succeeds, false if it fails.

Definition at line 812 of file Bdd.cs.

6.5.3.28 public bool WebradioManager.Bdd.UpdateFilenames (string *oldName,* string *newName,* Webradio *webradio*)

Updates the filenames with new webradio's name.

Author

Simon Menetrey

Date

26.05.2014

Parameters

oldName	Webradio's old name.
newName	Webradio's new name.
webradio	The webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1162 of file Bdd.cs.

6.5.3.29 public bool WebradioManager.Bdd.UpdateServer (int *port,* string *password,* string *adminPassword,* int *maxListener,* int *webradioId*)

Updates the server configuration.

Author

Simon Menetrey

Date

26.05.2014

Parameters

port	The port.
password	The password.
adminPassword	The admin password.
maxListener	The number of maximum listener.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1018 of file Bdd.cs.

6.5.3.30 public bool WebradioManager.Bdd.UpdateTranscoder (WebradioTranscoder *transcoder*)

Updates the transcoder with transcoder param's values.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 977 of file Bdd.cs.

6.5.3.31 public bool WebradioManager.Bdd.WebradioExist (string *name*)

Webradio exists in the db.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The webradio's name.

Returns

true if it exists, false if it not exists.

Definition at line 315 of file Bdd.cs.

6.5.4 Property Documentation

6.5.4.1 public BddControls WebradioManager.Bdd.Controls [get], [set]

Gets or sets the bdd controls.

Returns

The controls.

Definition at line 46 of file Bdd.cs.

The documentation for this class was generated from the following file:

- Bdd.cs

6.6 WebradioManager.BddControls Class Reference

Public Member Functions

- DataTable GetDataTable (string sql)

  Allows the programmer to run a query against the Database.
- int ExecuteNonQuery (string sql)

  Allows the programmer to interact with the database for purposes other than a query.
- SQLiteDataReader ExecuteDataReader (string sql)
- bool Update (String tableName, Dictionary< String, String > data, String where)

  Allows the programmer to easily update rows in the DB.
- bool Delete (String tableName, String where)

  Allows the programmer to easily delete rows from the DB.
- bool Insert (String tableName, Dictionary< String, String > data)

  Allows the programmer to easily insert into the DB
- bool ClearDB ()

  Allows the programmer to easily delete all data from the DB.
- bool ClearTable (String table)

  Allows the user to easily clear all data from a specific table.

Static Public Member Functions

- static string ExecuteScalar (string sql)

  Allows the programmer to retrieve single items from the DB.

6.6.1 Detailed Description

Definition at line 13 of file BddControls.cs.

6.6.2 Member Function Documentation

6.6.2.1 bool WebradioManager.BddControls.ClearDB ( )

Allows the programmer to easily delete all data from the DB.

Returns

A boolean true or false to signify success or failure.

Definition at line 186 of file BddControls.cs.

6.6.2.2 bool WebradioManager.BddControls.ClearTable (String *table*)

Allows the user to easily clear all data from a specific table.

Parameters

table	The name of the table to clear.

Returns

A boolean true or false to signify success or failure.

Definition at line 209 of file BddControls.cs.

6.6.2.3 bool WebradioManager.BddControls.Delete (String *tableName,* String *where*)

Allows the programmer to easily delete rows from the DB.

Parameters

tableName	The table from which to delete.
where	The where clause for the delete.

Returns

A boolean true or false to signify success or failure.

Definition at line 137 of file BddControls.cs.

6.6.2.4 SQLiteDataReader WebradioManager.BddControls.ExecuteDataReader (string *sql*)

Parameters

sql	

Definition at line 89 of file BddControls.cs.

6.6.2.5 int WebradioManager.BddControls.ExecuteNonQuery (string *sql*)

Allows the programmer to interact with the database for purposes other than a query.

Parameters

sql	The SQL to be run.

Returns

An Integer containing the number of rows updated.

Definition at line 54 of file BddControls.cs.

6.6.2.6 static string WebradioManager.BddControls.ExecuteScalar (string *sql*) [static]

Allows the programmer to retrieve single items from the DB.

Parameters

sql	The query to run.

Returns

A string.

Definition at line 70 of file BddControls.cs.

6.6.2.7 DataTable WebradioManager.BddControls.GetDataTable (string *sql*)

Allows the programmer to run a query against the Database.

Parameters

sql	The SQL to run

Returns

A DataTable containing the result set.

Definition at line 28 of file BddControls.cs.

6.6.2.8 bool WebradioManager.BddControls.Insert (String *tableName,* Dictionary< String, String > *data*)

Allows the programmer to easily insert into the DB

Parameters

tableName	The table into which we insert the data.
data	A dictionary containing the column names and data for the insert.

Returns

A boolean true or false to signify success or failure.

Definition at line 158 of file BddControls.cs.

6.6.2.9 bool WebradioManager.BddControls.Update (String *tableName*, Dictionary< String, String > *data*, String *where*)

Allows the programmer to easily update rows in the DB.

Parameters

tableName	The table to update.
data	A dictionary containing Column names and their new values.
where	The where clause for the update statement.

Returns

A boolean true or false to signify success or failure.

Definition at line 108 of file BddControls.cs.

The documentation for this class was generated from the following file:

- BddControls.cs

6.7 WebradioManager.CalendarEvent Class Reference

A calendar event.

Public Member Functions

- CalendarEvent (string name, TimeSpan starttime, TimeSpan duration, int repeat, int priority, bool shuffle, bool loopatend, Playlist playlist)

  Constructor.
- CalendarEvent (int id, string name, TimeSpan starttime, TimeSpan duration, int repeat, int priority, bool shuffle, bool loopatend, Playlist playlist)

  Constructor.
- DayWeek GetSelectedDays ()

  Gets selected days from repest value. `http://wiki.winamp.com/wiki/SHOUTcast_Calendar_Event_XML_File_Specification#Calendar_Tag`.

Properties

- int Id `[get, set]`

  Gets or sets the identifier.
- Playlist Playlist `[get, set]`

  Gets or sets the playlist.
- string Name `[get, set]`

Gets or sets the name.

- TimeSpan StartTime `[get, set]`

  Gets or sets the start time.

- TimeSpan Duration `[get, set]`

  Gets or sets the duration.

- int Repeat `[get, set]`

  Gets or sets the repeat's value.

- bool Shuffle `[get, set]`

  Gets or sets a value indicating whether the shuffle.

- bool Loopatend `[get, set]`

  Gets or sets a value indicating whether the loopatend.

- int Priority `[get, set]`

  Gets or sets the priority.

6.7.1 Detailed Description

A calendar event.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 44 of file CalendarEvent.cs.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 public WebradioManager.CalendarEvent.CalendarEvent (string *name*, TimeSpan *starttime*, TimeSpan *duration*, int *repeat*, int *priority*, bool *shuffle*, bool *loopatend*, Playlist *playlist*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
starttime	The starttime.
duration	The duration.
repeat	The repeat.
priority	The priority.

shuffle	true to shuffle.
loopatend	true to loopatend.
playlist	The playlist.

Definition at line 233 of file CalendarEvent.cs.

6.7.2.2 public WebradioManager.CalendarEvent.CalendarEvent (int *id,* string *name,* TimeSpan *starttime,* TimeSpan *duration,* int *repeat,* int *priority,* bool *shuffle,* bool *loopatend,* Playlist *playlist*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
starttime	The starttime.
duration	The duration.
repeat	The repeat.
priority	The priority.
shuffle	true to shuffle.
loopatend	true to loopatend.
playlist	The playlist.

Definition at line 258 of file CalendarEvent.cs.

6.7.3 Member Function Documentation

6.7.3.1 public DayWeek WebradioManager.CalendarEvent.GetSelectedDays ( )

Gets selected days from repest value. http://wiki.winamp.com/wiki/SHOUTcast_Calendar_-Event_XML_File_Specification#Calendar_Tag.

Author

Simon Menetrey

Date

26.05.2014

Returns

The selected days in DayWeek structure.

Definition at line 283 of file CalendarEvent.cs.

6.7.4 Property Documentation

6.7.4.1 public TimeSpan WebradioManager.CalendarEvent.Duration [get], [set]

Gets or sets the duration.

Returns

The duration.

Definition at line 152 of file CalendarEvent.cs.

6.7.4.2 public int WebradioManager.CalendarEvent.Id [get], [set]

Gets or sets the identifier.

Returns

The identifier.

Definition at line 96 of file CalendarEvent.cs.

6.7.4.3 public bool WebradioManager.CalendarEvent.Loopatend [get], [set]

Gets or sets a value indicating whether the loopatend.

Returns

true if loopatend, false if not.

Definition at line 194 of file CalendarEvent.cs.

6.7.4.4 public string WebradioManager.CalendarEvent.Name [get], [set]

Gets or sets the name.

Returns

The name.

Definition at line 124 of file CalendarEvent.cs.

6.7.4.5 public Playlist WebradioManager.CalendarEvent.Playlist [get], [set]

Gets or sets the playlist.

Returns

The playlist.

Definition at line 110 of file CalendarEvent.cs.

6.7.4.6 public int WebradioManager.CalendarEvent.Priority [get], [set]

Gets or sets the priority.

Returns

The priority.

Definition at line 208 of file CalendarEvent.cs.

6.7.4.7 public int WebradioManager.CalendarEvent.Repeat [get], [set]

Gets or sets the repeat's value.

Returns

The repeat.

Definition at line 166 of file CalendarEvent.cs.

6.7.4.8 public bool WebradioManager.CalendarEvent.Shuffle [get], [set]

Gets or sets a value indicating whether the shuffle.

Returns

true if shuffle, false if not.

Definition at line 180 of file CalendarEvent.cs.

6.7.4.9 public TimeSpan WebradioManager.CalendarEvent.StartTime [get], [set]

Gets or sets the start time.

Returns

The start time.

Definition at line 138 of file CalendarEvent.cs.

The documentation for this class was generated from the following file:

- CalendarEvent.cs

6.8 WebradioManager.DayWeek Struct Reference

A week with boolean values for each day. Uses for store which day is selected for an CalendarEvent.

Public Member Functions

- bool[] **ToArray** ()

Public Attributes

- const int **DAYS_COUNT** = 7

Properties

- bool Monday [get, set]

  Gets or sets a value indicating whether the monday.
- bool Tuesday [get, set]

  Gets or sets a value indicating whether the tuesday.
- bool Wednesday [get, set]

  Gets or sets a value indicating whether the wednesday.
- bool Thursday [get, set]

  Gets or sets a value indicating whether the thursday.
- bool Friday [get, set]

  Gets or sets a value indicating whether the friday.
- bool Saturday [get, set]

  Gets or sets a value indicating whether the saturday.
- bool Sunday [get, set]

  Gets or sets a value indicating whether the sunday.

6.8.1 Detailed Description

A week with boolean values for each day. Uses for store which day is selected for an CalendarEvent.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 18 of file DayWeek.cs.

6.8.2 Property Documentation

6.8.2.1 public bool WebradioManager.DayWeek.Friday [get],[set]

Gets or sets a value indicating whether the friday.

Returns

true if friday, false if not.

Definition at line 109 of file DayWeek.cs.

6.8.2.2 public bool WebradioManager.DayWeek.Monday [get],[set]

Gets or sets a value indicating whether the monday.

Returns

true if monday, false if not.

Definition at line 53 of file DayWeek.cs.

6.8.2.3 public bool WebradioManager.DayWeek.Saturday [get], [set]

Gets or sets a value indicating whether the saturday.

Returns

true if saturday, false if not.

Definition at line 123 of file DayWeek.cs.

6.8.2.4 public bool WebradioManager.DayWeek.Sunday [get], [set]

Gets or sets a value indicating whether the sunday.

Returns

true if sunday, false if not.

Definition at line 137 of file DayWeek.cs.

6.8.2.5 public bool WebradioManager.DayWeek.Thursday [get], [set]

Gets or sets a value indicating whether the thursday.

Returns

true if thursday, false if not.

Definition at line 95 of file DayWeek.cs.

6.8.2.6 public bool WebradioManager.DayWeek.Tuesday [get], [set]

Gets or sets a value indicating whether the tuesday.

Returns

true if tuesday, false if not.

Definition at line 67 of file DayWeek.cs.

6.8.2.7 public bool WebradioManager.DayWeek.Wednesday [get], [set]

Gets or sets a value indicating whether the wednesday.

Returns

true if wednesday, false if not.

Definition at line 81 of file DayWeek.cs.

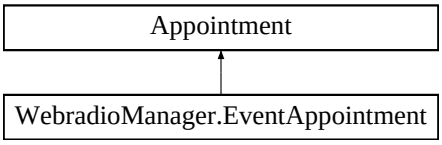
The documentation for this struct was generated from the following file:

- DayWeek.cs

6.9 WebradioManager.EventAppointment Class Reference

An event appointment. Add 2 properties to the original Appointment class (from Calendar library).

Inheritance diagram for WebradioManager.EventAppointment:



Public Member Functions

- EventAppointment ()
  Default constructor.

Properties

- CalendarEvent EventObject `[get, set]`
  Gets or sets the event object.
- Playlist Playlist `[get, set]`
  Gets or sets the playlist.

6.9.1 Detailed Description

An event appointment. Add 2 properties to the original Appointment class (from Calendar library).

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file EventAppointment.cs.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 public WebradioManager.EventAppointment.EventAppointment ( )

Default constructor.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 69 of file EventAppointment.cs.

6.9.3 Property Documentation

6.9.3.1 public CalendarEvent WebradioManager.EventAppointment.EventObject [get], [set]

Gets or sets the event object.

Returns

The event object.

Definition at line 39 of file EventAppointment.cs.

6.9.3.2 public Playlist WebradioManager.EventAppointment.Playlist [get], [set]

Gets or sets the playlist.

Returns

The playlist.

Definition at line 53 of file EventAppointment.cs.

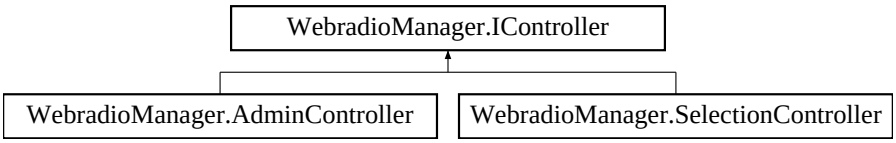
The documentation for this class was generated from the following file:

- EventAppointment.cs

6.10 WebradioManager.IController Interface Reference

Interface for controller.

Inheritance diagram for WebradioManager.IController:



Public Member Functions

- void UpdateView ()

  Updates the view.

6.10.1 Detailed Description

Interface for controller.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 19 of file IController.cs.

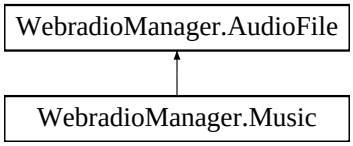
The documentation for this interface was generated from the following file:

- IController.cs

6.11 WebradioManager.Music Class Reference

A music.

Inheritance diagram for WebradioManager.Music:



Public Member Functions

- Music (int id, string filename, string title, string artist, string album, int year, string label, TimeSpan duration, string gender)

  Constructor.
- Music (string filename, string title, string artist, string album, int year, string label, TimeSpan duration, string gender)

  Constructor.

Additional Inherited Members

6.11.1 Detailed Description

A music.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file Music.cs.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 public WebradioManager.Music.Music (int *id,* string *filename,* string *title,* string *artist,* string *album,* int *year,* string *label,* TimeSpan *duration,* string *gender*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
filename	Filename of the audiofile.
title	The title.
artist	The artist.
album	The album.
year	The year.
label	The label.
duration	The duration.
gender	The gender.

Definition at line 42 of file Music.cs.

6.11.2.2 public WebradioManager.Music.Music (string *filename*, string *title*, string *artist*, string *album*, int *year*, string *label*, TimeSpan *duration*, string *gender*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

filename	Filename of the audiofile.
title	The title.
artist	The artist.
album	The album.
year	The year.
label	The label.
duration	The duration.
gender	The gender.

Definition at line 66 of file Music.cs.

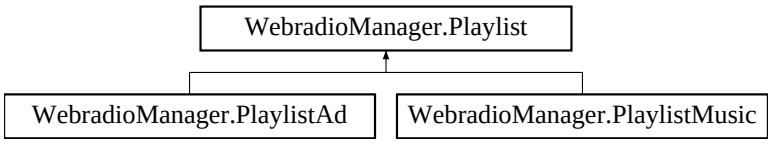
The documentation for this class was generated from the following file:

- Music.cs

6.12 WebradioManager.Playlist Class Reference

A playlist. Abstract class.

Inheritance diagram for WebradioManager.Playlist:



Public Member Functions

- Playlist (string name, string filename, AudioType type)

  Constructor.
- Playlist (int id, string name, string filename, AudioType type)

  Constructor.
- void GenerateConfigFile ()

  Generates a configuration file.
- override string ToString ()

  Convert this object into a string representation.

Properties

- List< string > AudioFileList `[get, set]`

  Gets or sets a list of audio files's filename.
- AudioType Type `[get, set]`

  Gets or sets the type.
- int Id `[get, set]`

  Gets or sets the identifier.
- string Filename `[get, set]`

  Gets or sets the filename of the file.
- string Name `[get, set]`

  Gets or sets the name.

6.12.1 Detailed Description

A playlist. Abstract class.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 21 of file Playlist.cs.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 public WebradioManager.Playlist.Playlist (string *name*, string *filename*, AudioType *type*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
filename	Filename of the file.
type	The type.

Definition at line 129 of file Playlist.cs.

6.12.2.2 public WebradioManager.Playlist.Playlist (int *id,* string *name,* string *filename,* AudioType *type*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
filename	Filename of the file.
type	The type.

Definition at line 148 of file Playlist.cs.

6.12.3 Member Function Documentation

6.12.3.1 public void WebradioManager.Playlist.GenerateConfigFile ( )

Generates a configuration file.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 166 of file Playlist.cs.

6.12.3.2 public override string WebradioManager.Playlist.ToString ( )

Convert this object into a string representation.

Author

Simon Menetrey

Date

26.05.2014

Returns

Chaîne qui représente l'objet actif.

summary Retourne une chaîne qui représente l'objet actif.

Definition at line 191 of file Playlist.cs.

6.12.4 Property Documentation

6.12.4.1 public List< string > WebradioManager.Playlist.AudioFileList [get], [set]

Gets or sets a list of audio files's filename.

Returns

A List of audio files.

Definition at line 52 of file Playlist.cs.

6.12.4.2 public string WebradioManager.Playlist.Filename [get], [set]

Gets or sets the filename of the file.

Returns

The filename.

Definition at line 94 of file Playlist.cs.

6.12.4.3 public int WebradioManager.Playlist.Id [get], [set]

Gets or sets the identifier.

Returns

The identifier.

Definition at line 80 of file Playlist.cs.

6.12.4.4 public string WebradioManager.Playlist.Name [get], [set]

Gets or sets the name.

Returns

The name.

Definition at line 108 of file Playlist.cs.

6.12.4.5 public AudioType WebradioManager.Playlist.Type [get], [set]

Gets or sets the type.

Returns

The type.

Definition at line 66 of file Playlist.cs.

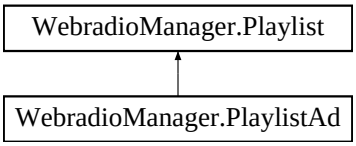
The documentation for this class was generated from the following file:

- Playlist.cs

6.13 WebradioManager.PlaylistAd Class Reference

A playlist ad.

Inheritance diagram for WebradioManager.PlaylistAd:



Public Member Functions

- PlaylistAd (int id, string name, string filename)
  Constructor.
- PlaylistAd (string name, string filename)
  Constructor.

Additional Inherited Members

6.13.1 Detailed Description

A playlist ad.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 18 of file PlaylistAd.cs.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 public WebradioManager.PlaylistAd.PlaylistAd (int *id*, string *name*, string *filename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
filename	Filename of the file.

Definition at line 34 of file PlaylistAd.cs.

6.13.2.2 public WebradioManager.PlaylistAd.PlaylistAd (string *name,* string *filename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
filename	Filename of the file.

Definition at line 51 of file PlaylistAd.cs.

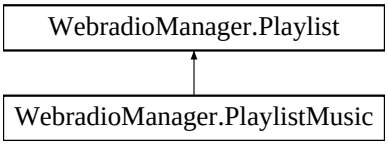
The documentation for this class was generated from the following file:

- PlaylistAd.cs

6.14 WebradioManager.PlaylistMusic Class Reference

A playlist music.

Inheritance diagram for WebradioManager.PlaylistMusic:



Public Member Functions

- PlaylistMusic (int id, string name, string filename)
  Constructor.
- PlaylistMusic (string name, string filename)
  Constructor.

Additional Inherited Members

6.14.1 Detailed Description

A playlist music.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 18 of file PlaylistMusic.cs.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 public WebradioManager.PlaylistMusic.PlaylistMusic (int *id*, string *name*, string *filename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
filename	Filename of the file.

Definition at line 34 of file PlaylistMusic.cs.

6.14.2.2 public WebradioManager.PlaylistMusic.PlaylistMusic (string *name*, string *filename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.

filename	Filename of the file.

Definition at line 52 of file PlaylistMusic.cs.

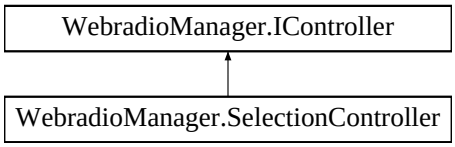
The documentation for this class was generated from the following file:

- PlaylistMusic.cs

6.15 WebradioManager.SelectionController Class Reference

A controller for SelectionView.

Inheritance diagram for WebradioManager.SelectionController:



Public Member Functions

- SelectionController (SelectionView view)

  Constructor.
- void LoadLibrary ()

  Loads the library.
- void LoadWebradios ()

  Loads the webradios.
- void UpdateView ()

  Updates the view.
- List< Webradio > GetWebradios ()

  Gets the webradios list.
- bool CreateWebradio (string name)

  Creates a webradio.
- bool DeleteWebradio (int id)

  Deletes the webradio described by ID.
- bool DuplicateWebradio (int id)

  Duplicate webradio described by id.
- void OpenWebradio (int id)

  Opens a webradio described by id in an AdminView.
- bool StopAllProcess ()

  Stops all process.

Properties

- WMModel Model `[get, set]`

  Gets or sets the model.
- SelectionView View `[get, set]`

  Gets or sets the view.

6.15.1 Detailed Description

A controller for SelectionView.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file SelectionController.cs.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 public WebradioManager.SelectionController.SelectionController (SelectionView *view*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

view	The view.

Definition at line 73 of file SelectionController.cs.

6.15.3 Member Function Documentation

6.15.3.1 public bool WebradioManager.SelectionController.CreateWebradio (string *name*)

Creates a webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The webradio's name.

Returns

true if it succeeds, false if it fails.

Definition at line 151 of file SelectionController.cs.

6.15.3.2 public bool WebradioManager.SelectionController.DeleteWebradio (int *id*)

Deletes the webradio described by ID.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 169 of file SelectionController.cs.

6.15.3.3 public bool WebradioManager.SelectionController.DuplicateWebradio (int *id*)

Duplicate webradio described by id.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 187 of file SelectionController.cs.

6.15.3.4 public List< **Webradio** > WebradioManager.SelectionController.GetWebradios ( )

Gets the webradios list.

Author

Simon Menetrey

Date

26.05.2014

Returns

The webradios list.

Definition at line 133 of file SelectionController.cs.

6.15.3.5 public void WebradioManager.SelectionController.LoadLibrary ( )

Loads the library.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 89 of file SelectionController.cs.

6.15.3.6 public void WebradioManager.SelectionController.LoadWebradios ( )

Loads the webradios.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 103 of file SelectionController.cs.

6.15.3.7 public void WebradioManager.SelectionController.OpenWebradio (int *id*)

Opens a webradio described by id in an AdminView.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Definition at line 203 of file SelectionController.cs.

6.15.3.8 public bool WebradioManager.SelectionController.StopAllProcess ( )

Stops all process.

Author

Simon Menetrey

Date

26.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 221 of file SelectionController.cs.

6.15.3.9 public void WebradioManager.SelectionController.UpdateView ( )

Updates the view.

Author

Simon Menetrey

Date

26.05.2014

Implements WebradioManager.IController.

Definition at line 117 of file SelectionController.cs.

6.15.4 Property Documentation

6.15.4.1 public WMModel WebradioManager.SelectionController.Model [get], [set]

Gets or sets the model.

Returns

The model.

Definition at line 40 of file SelectionController.cs.

6.15.4.2 public SelectionView WebradioManager.SelectionController.View [get], [set]

Gets or sets the view.

Returns

The view.

Definition at line 54 of file SelectionController.cs.

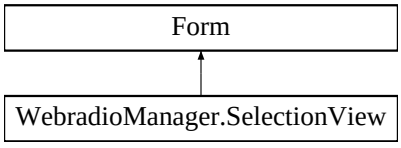
The documentation for this class was generated from the following file:

- SelectionController.cs

6.16 WebradioManager.SelectionView Class Reference

A selection view.

Inheritance diagram for WebradioManager.SelectionView:



Public Member Functions

- SelectionView ()

  Default constructor.
- void UpdateView ()

  Updates the view.

Protected Member Functions

- override void Dispose (bool disposing)

  Clean up any resources being used.

Properties

- SelectionController Controller `[get, set]`

  Gets or sets the controller.

6.16.1 Detailed Description

A selection view.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 22 of file SelectionView.cs.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 public WebradioManager.SelectionView.SelectionView ( )

Default constructor.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 62 of file SelectionView.cs.

6.16.3 Member Function Documentation

6.16.3.1 override void WebradioManager.SelectionView.Dispose (bool *disposing*) `[protected]`

Clean up any resources being used.

Parameters

disposing	true if managed resources should be disposed; otherwise, false.

Definition at line 14 of file SelectionView.Designer.cs.

6.16.3.2 public void WebradioManager.SelectionView.UpdateView ( )

Updates the view.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 82 of file SelectionView.cs.

6.16.4 Property Documentation

6.16.4.1 public SelectionController WebradioManager.SelectionView.Controller [get], [set]

Gets or sets the controller.

Returns

The controller.

Definition at line 45 of file SelectionView.cs.

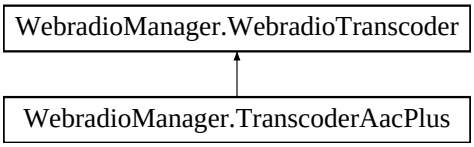
The documentation for this class was generated from the following files:

- SelectionView.cs
- SelectionView.Designer.cs

6.17 WebradioManager.TranscoderAacPlus Class Reference

A transcoder aac plus.

Inheritance diagram for WebradioManager.TranscoderAacPlus:



Public Member Functions

- TranscoderAacPlus (int id, string name, int bitrate, int sampleRate, IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename)

  Constructor.
- TranscoderAacPlus (string name, int bitrate, int sampleRate, IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename)

  Constructor.

Additional Inherited Members

6.17.1 Detailed Description

A transcoder aac plus.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file TranscoderAacPlus.cs.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 public WebradioManager.TranscoderAacPlus.TranscoderAacPlus (int *id*, string *name*, int *bitrate*, int *sampleRate*, IPAddress *ip*, int *port*, int *adminport*, string *url*, string *password*, string *configFilename*, string *logFilename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
bitrate	The bitrate.
sampleRate	The sample rate.
ip	The IP.
port	The port.
adminport	The administration port.
url	URL of the stream.
password	The password.
configFilename	Filename of the configuration file.
logFilename	Filename of the log file.

Definition at line 44 of file TranscoderAacPlus.cs.

6.17.2.2 public WebradioManager.TranscoderAacPlus.TranscoderAacPlus (string *name*, int *bitrate*, int *sampleRate*, IPAddress *ip*, int *port*, int *adminport*, string *url*, string *password*, string *configFilename*, string *logFilename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
bitrate	The bitrate.
sampleRate	The sample rate.
ip	The IP.
port	The port.
adminport	The administration port.
url	URL of the stream.
password	The password.
configFilename	Filename of the configuration file.
logFilename	Filename of the log file.

Definition at line 70 of file TranscoderAacPlus.cs.

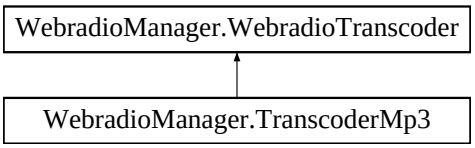
The documentation for this class was generated from the following file:

- TranscoderAacPlus.cs

6.18 WebradioManager.TranscoderMp3 Class Reference

A transcoder mp3.

Inheritance diagram for WebradioManager.TranscoderMp3:



Public Member Functions

- TranscoderMp3 (int id, string name, int bitrate, int sampleRate, IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename)

  Constructor.
- TranscoderMp3 (string name, int bitrate, int sampleRate, IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename)

  Constructor.

Additional Inherited Members

6.18.1 Detailed Description

A transcoder mp3.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file TranscoderMp3.cs.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 public WebradioManager.TranscoderMp3.TranscoderMp3 (int *id,* string *name,* int *bitrate,* int *sampleRate,* IPAddress *ip,* int *port,* int *adminport,* string *url,* string *password,* string *configFilename,* string *logFilename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
bitrate	The bitrate.
sampleRate	The sample rate.
ip	The IP.
port	The port.
adminport	The administration port.
url	URL of the stream.
password	The password.
configFilename	Filename of the configuration file.
logFilename	Filename of the log file.

Definition at line 44 of file TranscoderMp3.cs.

6.18.2.2 public WebradioManager.TranscoderMp3.TranscoderMp3 (string *name,* int *bitrate,* int *sampleRate,* IPAddress *ip,* int *port,* int *adminport,* string *url,* string *password,* string *configFilename,* string *logFilename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
bitrate	The bitrate.
sampleRate	The sample rate.
ip	The IP.
port	The port.
adminport	The administration port.
url	URL of the stream.

password	The password.
configFilename	Filename of the configuration file.
logFilename	Filename of the log file.

Definition at line 70 of file TranscoderMp3.cs.

The documentation for this class was generated from the following file:

- TranscoderMp3.cs

6.19 WebradioManager.Webradio Class Reference

A webradio.

Public Member Functions

- Webradio (string name, int id)
  Constructor.
- Webradio (string name)
  Constructor.
- void GenerateConfigFiles ()
  Generates a configuration files.
- override string ToString ()
  Convert this object into a string representation.

Properties

- List< WebradioTranscoder > Transcoders `[get, set]`
  Gets or sets the transcoders list.
- WebradioServer Server `[get, set]`
  Gets or sets the server.
- string Name `[get, set]`
  Gets or sets the name.
- WebradioCalendar Calendar `[get, set]`
  Gets or sets the calendar.
- List< Playlist > Playlists `[get, set]`
  Gets or sets the playlists list.
- int Id `[get, set]`
  Gets or sets the identifier.

6.19.1 Detailed Description

A webradio.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file Webradio.cs.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 public WebradioManager.Webradio.Webradio (string *name*, int *id*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
id	The identifier.

Definition at line 143 of file Webradio.cs.

6.19.2.2 public WebradioManager.Webradio.Webradio (string *name*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.

Definition at line 162 of file Webradio.cs.

6.19.3 Member Function Documentation

6.19.3.1 public void WebradioManager.Webradio.GenerateConfigFiles ( )

Generates a configuration files.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 176 of file Webradio.cs.

6.19.3.2 public override string WebradioManager.Webradio.ToString ( )

Convert this object into a string representation.

Author

Simon Menetrey

Date

26.05.2014

Returns

Chaîne qui représente l'objet actif.

summary Retourne une chaîne qui représente l'objet actif.

Definition at line 203 of file Webradio.cs.

6.19.4 Property Documentation

6.19.4.1 public WebradioCalendar WebradioManager.Webradio.Calendar [get], [set]

Gets or sets the calendar.

Returns

The calendar.

Definition at line 95 of file Webradio.cs.

6.19.4.2 public int WebradioManager.Webradio.Id [get], [set]

Gets or sets the identifier.

Returns

The identifier.

Definition at line 123 of file Webradio.cs.

6.19.4.3 public string WebradioManager.Webradio.Name [get], [set]

Gets or sets the name.

Returns

The name.

Definition at line 81 of file Webradio.cs.

6.19.4.4 public List< Playlist > WebradioManager.Webradio.Playlists [get], [set]

Gets or sets the playlists list.

Returns

The playlists.

Definition at line 109 of file Webradio.cs.

6.19.4.5 public WebradioServer WebradioManager.Webradio.Server [get], [set]

Gets or sets the server.

Returns

The server.

Definition at line 67 of file Webradio.cs.

6.19.4.6 public List< WebradioTranscoder > WebradioManager.Webradio.Transcoders [get], [set]

Gets or sets the transcoders list.

Returns

The transcoders.

Definition at line 53 of file Webradio.cs.

The documentation for this class was generated from the following file:

- Webradio.cs

6.20 WebradioManager.WebradioCalendar Class Reference

A webradio calendar.

Public Member Functions

- WebradioCalendar (int id, string filename)
  Constructor.
- WebradioCalendar (string filename)
  Constructor.
- void GenerateConfigFile ()
  Generates a configuration file.

Properties

- int Id [get, set]
  Gets or sets the identifier.
- string Filename [get, set]
  Gets or sets the filename of the calendar's file.
- List< CalendarEvent > Events [get, set]
  Gets or sets the events.

6.20.1 Detailed Description

A webradio calendar.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 22 of file WebradioCalendar.cs.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 public WebradioManager.WebradioCalendar.WebradioCalendar (int *id,* string *filename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
filename	Filename of the file.

Definition at line 92 of file WebradioCalendar.cs.

6.20.2.2 public WebradioManager.WebradioCalendar.WebradioCalendar (string *filename*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

filename	Filename of the file.

Definition at line 110 of file WebradioCalendar.cs.

6.20.3 Member Function Documentation

6.20.3.1 public void WebradioManager.WebradioCalendar.GenerateConfigFile ( )

Generates a configuration file.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 125 of file WebradioCalendar.cs.

6.20.4 Property Documentation

6.20.4.1 public List< CalendarEvent > WebradioManager.WebradioCalendar.Events [get], [set]

Gets or sets the events.

Returns

The events.

Definition at line 72 of file WebradioCalendar.cs.

6.20.4.2 public string WebradioManager.WebradioCalendar.Filename [get], [set]

Gets or sets the filename of the calendar's file.

Returns

The filename.

Definition at line 58 of file WebradioCalendar.cs.

6.20.4.3 public int WebradioManager.WebradioCalendar.Id [get], [set]

Gets or sets the identifier.

Returns

The identifier.

Definition at line 44 of file WebradioCalendar.cs.

The documentation for this class was generated from the following file:

- WebradioCalendar.cs

6.21 WebradioManager.WebradioListener Class Reference

A webradio listener.

Public Member Functions

- WebradioListener (string hostname, string useragent, uint connectiontime, int uid)
  Constructor.

Properties

- int Uid [get, set]
  Gets or sets the UID.
- uint ConnectionTime [get, set]
  Gets or sets the connection time.
- string Useragent [get, set]
  Gets or sets the useragent.
- string Hostname [get, set]
  Gets or sets the hostname.

6.21.1 Detailed Description

A webradio listener.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 19 of file WebradioListener.cs.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 public WebradioManager.WebradioListener.WebradioListener (string *hostname,* string *useragent,* uint *connectiontime,* int *uid*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

hostname	The hostname.
useragent	The useragent.
connectiontime	The connection time.
uid	The UID.

Definition at line 108 of file WebradioListener.cs.

6.21.3 Property Documentation

6.21.3.1 public uint WebradioManager.WebradioListener.ConnectionTime [get],[set]

Gets or sets the connection time.

Returns

The connection time.

Definition at line 57 of file WebradioListener.cs.

6.21.3.2 public string WebradioManager.WebradioListener.Hostname [get],[set]

Gets or sets the hostname.

Returns

The hostname.

Definition at line 85 of file WebradioListener.cs.

6.21.3.3 public int WebradioManager.WebradioListener.Uid [get], [set]

Gets or sets the UID.

Returns

The UID.

Definition at line 43 of file WebradioListener.cs.

6.21.3.4 public string WebradioManager.WebradioListener.Useragent [get], [set]

Gets or sets the useragent.

Returns

The useragent.

Definition at line 71 of file WebradioListener.cs.

The documentation for this class was generated from the following file:

- WebradioListener.cs

6.22 WebradioManager.WebradioServer Class Reference

A shoutcast webradio server.

Public Member Functions

- WebradioServer (int port, string logFilename, string configFilename, string password, string adminPassword, int maxListener)

  Constructor.
- void GenerateConfigFile ()

  Generates the configuration file.
- bool IsRunning ()

  Query if the process is running.
- bool Start (bool debug)

  Starts the process.
- bool Stop ()

  Stops the process.
- bool UpdateStats ()

  Updates the statistics.
- List< WebradioListener > GetListeners ()

  Gets the list of listeners.

Public Attributes

- const string **DEFAULT_ADMIN_LOGIN** = "admin"

Properties

- string WebInterfaceUrl `[get]`

  Gets URL of the web interface.
- string WebAdminUrl `[get]`

  Gets URL of the web admin.
- int MaxListener `[get, set]`

  Gets or sets the number of maximum listener.
- Process Process `[get, set]`

  Gets or sets the process.
- string LogFilename `[get, set]`

  Gets or sets the filename of the log file.
- string ConfigFilename `[get, set]`

  Gets or sets the filename of the configuration file.
- string Password `[get, set]`

  Gets or sets the password.
- string AdminPassword `[get, set]`

  Gets or sets the admin password.
- int Port `[get, set]`

  Gets or sets the port.

6.22.1 Detailed Description

A shoutcast webradio server.

Author

Simon Menetrey

Date

21.05.2014

Definition at line 26 of file WebradioServer.cs.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 public WebradioManager.WebradioServer.WebradioServer (int *port*, string *logfilename*, string *configfilename*, string *password*, string *adminPassword*, int *maxlistener*)

Constructor.

Author

Simon Menetrey

Date

21.05.2014

Parameters

port	The port.
logfilename	The filename of the log file.
configfilename	The filename of the configuration file.
password	The password.
adminPassword	The admin password.
maxlistener	The number of max listener.

Definition at line 218 of file WebradioServer.cs.

6.22.3 Member Function Documentation

6.22.3.1 public void WebradioManager.WebradioServer.GenerateConfigFile ( )

Generates the configuration file.

Author

Simon Menetrey

Date

21.05.2014

Definition at line 238 of file WebradioServer.cs.

6.22.3.2 public List< WebradioListener > WebradioManager.WebradioServer.GetListeners ( )

Gets the list of listeners.

Author

Simon Menetrey

Date

21.05.2014

Returns

The list of listeners.

Definition at line 421 of file WebradioServer.cs.

6.22.3.3 public bool WebradioManager.WebradioServer.IsRunning ( )

Query if the process is running.

Author

Simon Menetrey

Date

21.05.2014

Returns

true if running, false if not.

Definition at line 265 of file WebradioServer.cs.

6.22.3.4 public bool WebradioManager.WebradioServer.Start (bool *debug*)

Starts the process.

Author

Simon Menetrey

Date

21.05.2014

Parameters

debug	True to debug mode.

Returns

true if it succeeds, false if it fails.

Definition at line 298 of file WebradioServer.cs.

6.22.3.5 public bool WebradioManager.WebradioServer.Stop ( )

Stops the process.

Author

Simon Menetrey

Date

21.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 330 of file WebradioServer.cs.

6.22.3.6 public bool WebradioManager.WebradioServer.UpdateStats ( )

Updates the statistics.

Author

Simon Menetrey

Date

21.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 385 of file WebradioServer.cs.

6.22.4 Property Documentation

6.22.4.1 public string WebradioManager.WebradioServer.AdminPassword [get], [set]

Gets or sets the admin password.

Returns

The admin password.

Definition at line 180 of file WebradioServer.cs.

6.22.4.2 public string WebradioManager.WebradioServer.ConfigFilename [get], [set]

Gets or sets the filename of the configuration file.

Returns

The filename of the configuration file.

Definition at line 152 of file WebradioServer.cs.

6.22.4.3 public string WebradioManager.WebradioServer.LogFilename [get], [set]

Gets or sets the filename of the log file.

Returns

The filename of the log file.

Definition at line 138 of file WebradioServer.cs.

6.22.4.4 public int WebradioManager.WebradioServer.MaxListener [get], [set]

Gets or sets the number of maximum listener.

Returns

The maximum listener.

Definition at line 110 of file WebradioServer.cs.

6.22.4.5 public string WebradioManager.WebradioServer.Password [get], [set]

Gets or sets the password.

Returns

The password.

Definition at line 166 of file WebradioServer.cs.

6.22.4.6 public int WebradioManager.WebradioServer.Port [get], [set]

Gets or sets the port.

Returns

The port.

Definition at line 194 of file WebradioServer.cs.

6.22.4.7 public Process WebradioManager.WebradioServer.Process [get], [set]

Gets or sets the process.

Returns

The process.

Definition at line 124 of file WebradioServer.cs.

6.22.4.8 public string WebradioManager.WebradioServer.WebAdminUrl [get]

Gets URL of the web admin.

Returns

The web admin URL.

Definition at line 94 of file WebradioServer.cs.

6.22.4.9 public string WebradioManager.WebradioServer.WebInterfaceUrl [get]

Gets URL of the web interface.

Returns

The web interface URL.

Definition at line 78 of file WebradioServer.cs.

The documentation for this class was generated from the following file:

- WebradioServer.cs

6.23 WebradioManager.WebradioServerStats Class Reference

A webradio server statistics.

Public Member Functions

- WebradioServerStats ()

  Default constructor.

Properties

- TimeSpan AverageTime `[get, set]`

  Gets or sets the average time listening.
- int PeakListeners `[get, set]`

  Gets or sets the peak listeners count.
- int UniqueListeners `[get, set]`

  Gets or sets the unique listeners count.
- int CurrentListeners `[get, set]`

  Gets or sets the current listeners count.

6.23.1 Detailed Description

A webradio server statistics.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 20 of file WebradioServerStats.cs.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 public WebradioManager.WebradioServerStats.WebradioServerStats ( )

Default constructor.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 103 of file WebradioServerStats.cs.

6.23.3 Property Documentation

6.23.3.1 public TimeSpan WebradioManager.WebradioServerStats.AverageTime `[get],[set]`

Gets or sets the average time listening.

Returns

The average time.

Definition at line 44 of file WebradioServerStats.cs.

6.23.3.2 public int WebradioManager.WebradioServerStats.CurrentListeners [get], [set]

Gets or sets the current listeners count.

Returns

The current listeners.

Definition at line 86 of file WebradioServerStats.cs.

6.23.3.3 public int WebradioManager.WebradioServerStats.PeakListeners [get], [set]

Gets or sets the peak listeners count.

Returns

The peak listeners.

Definition at line 58 of file WebradioServerStats.cs.

6.23.3.4 public int WebradioManager.WebradioServerStats.UniqueListeners [get], [set]

Gets or sets the unique listeners count.

Returns

The unique listeners.

Definition at line 72 of file WebradioServerStats.cs.

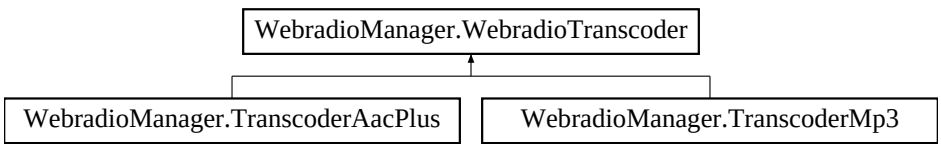
The documentation for this class was generated from the following file:

- WebradioServerStats.cs

6.24 WebradioManager.WebradioTranscoder Class Reference

A webradio transcoder.

Inheritance diagram for WebradioManager.WebradioTranscoder:



Public Member Functions

- WebradioTranscoder (string name, int bitrate, int sampleRate, IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename, StreamType st)

  Constructor.
- WebradioTranscoder (int id, string name, int bitrate, int sampleRate, IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename, StreamType st)

  Constructor.

- void GenerateConfigFile (List< Playlist > playlists)

  Generates a configuration file.
- bool IsRunning ()

  Query if this transcoder's process is running.
- bool Start (bool debug)

  Starts the transcoder's process.
- bool Stop ()

  Stops the transcoder's process.
- string GetStatus ()

  Gets the transcoder's status. `http://wiki.winamp.com/wiki/SHOUTcast_Transcoder_AJAX_-api_Specification#GetStatus`.
- void SetCaptureMode (bool active, string device)

  Sets capture mode.
- void NextTrack ()

  Go to next track.
- override string ToString ()

  Convert this object into a string representation.

Public Attributes

- const string **DEFAULT_CONFIG_EXTENSION** = ".config"
- const string **DEFAULT_LOG_EXTENSION** = ".log"
- const string **DEFAULT_ADMIN** = "admin"
- const string **DEFAULT_ADMIN_PASSWORD** = "admin"

Properties

- bool Capture `[get, set]`

  Gets or sets a value indicating whether the capture.
- string CurrentTrack `[get, set]`

  Gets or sets the current track's filename.
- int AdminPort `[get, set]`

  Gets or sets the admin port.
- string CalendarFile `[get, set]`

  Gets or sets the calendar file.
- Process Process `[get, set]`

  Gets or sets the process.
- static int[] **AvaliableSampleRates** `[get, set]`
- static int[] **AvaliableBitrates** `[get, set]`
- int Id `[get, set]`

  Gets or sets the identifier.
- int Birate `[get, set]`

  Gets or sets the birate.
- int SampleRate `[get, set]`

  Gets or sets the sample rate.
- string Name `[get, set]`

  Gets or sets the name.
- string Url `[get, set]`

  Gets or sets URL of the document.
- IPAddress Ip `[get, set]`

  Gets or sets the IP.

- int Port [get, set]

  Gets or sets the port.
- string Password [get, set]

  Gets or sets the password.
- string ConfigFilename [get, set]

  Gets or sets the filename of the configuration file.
- string LogFilename [get, set]

  Gets or sets the filename of the log file.
- StreamType StreamType [get, set]

  Gets or sets the type of the stream.

6.24.1 Detailed Description

A webradio transcoder.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 24 of file WebradioTranscoder.cs.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 public WebradioManager.WebradioTranscoder.WebradioTranscoder (string *name*, int *bitrate*, int *sampleRate*, IPAddress *ip*, int *port*, int *adminport*, string *url*, string *password*, string *configFilename*, string *logFilename*, StreamType *st*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
bitrate	The bitrate.
sampleRate	The sample rate.
ip	The IP.
port	The port.
adminport	The adminport.
url	URL of the document.

password	The password.
configFilename	Filename of the configuration file.
logFilename	Filename of the log file.
st	The st.

Definition at line 363 of file WebradioTranscoder.cs.

6.24.2.2 public WebradioManager.WebradioTranscoder.WebradioTranscoder (int *id,* string *name,* int *bitrate,* int *sampleRate,* IPAddress *ip,* int *port,* int *adminport,* string *url,* string *password,* string *configFilename,* string *logFilename,* StreamType *st*)

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
name	The name.
bitrate	The bitrate.
sampleRate	The sample rate.
ip	The IP.
port	The port.
adminport	The adminport.
url	URL of the document.
password	The password.
configFilename	Filename of the configuration file.
logFilename	Filename of the log file.
st	The st.

Definition at line 391 of file WebradioTranscoder.cs.

6.24.3 Member Function Documentation

6.24.3.1 public void WebradioManager.WebradioTranscoder.GenerateConfigFile (List< Playlist > *playlists*)

Generates a configuration file.

Author

Simon Menetrey

Date

26.05.2014

Parameters

playlists	The playlists.

Definition at line 421 of file WebradioTranscoder.cs.

6.24.3.2 public string WebradioManager.WebradioTranscoder.GetStatus ( )

Gets the transcoder's status. http://wiki.winamp.com/wiki/SHOUTcast_Transcoder_AJAX_-api_Specification#GetStatus.

Author

Simon Menetrey

Date

26.05.2014

Returns

The status (XML).

Definition at line 567 of file WebradioTranscoder.cs.

6.24.3.3 public bool WebradioManager.WebradioTranscoder.IsRunning ( )

Query if this transcoder's process is running.

Author

Simon Menetrey

Date

26.05.2014

Returns

true if running, false if not.

Definition at line 467 of file WebradioTranscoder.cs.

6.24.3.4 public void WebradioManager.WebradioTranscoder.NextTrack ( )

Go to next track.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 626 of file WebradioTranscoder.cs.

6.24.3.5 public void WebradioManager.WebradioTranscoder.SetCaptureMode (bool *active,* string *device*)

Sets capture mode.

Author

Simon Menetrey

Date

26.05.2014

Parameters

active	true to active.
device	The device name.

Definition at line 597 of file WebradioTranscoder.cs.

6.24.3.6 public bool WebradioManager.WebradioTranscoder.Start (bool *debug*)

Starts the transcoder's process.

Author

Simon Menetrey

Date

26.05.2014

Parameters

debug	true to debug.

Returns

true if it succeeds, false if it fails.

Definition at line 503 of file WebradioTranscoder.cs.

6.24.3.7 public bool WebradioManager.WebradioTranscoder.Stop ( )

Stops the transcoder's process.

Author

Simon Menetrey

Date

26.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 537 of file WebradioTranscoder.cs.

6.24.3.8 public override string WebradioManager.WebradioTranscoder.ToString ( )

Convert this object into a string representation.

Author

Simon Menetrey

Date

26.05.2014

Returns

Chaîne qui représente l'objet actif.

summary Retourne une chaîne qui représente l'objet actif.

Definition at line 649 of file WebradioTranscoder.cs.

6.24.4 Property Documentation

6.24.4.1 public int WebradioManager.WebradioTranscoder.AdminPort [get], [set]

Gets or sets the admin port.

Returns

The admin port.

Definition at line 124 of file WebradioTranscoder.cs.

6.24.4.2 public int WebradioManager.WebradioTranscoder.Birate [get], [set]

Gets or sets the birate.

Returns

The birate.

Definition at line 208 of file WebradioTranscoder.cs.

6.24.4.3 public string WebradioManager.WebradioTranscoder.CalendarFile [get], [set]

Gets or sets the calendar file.

Returns

The calendar file.

Definition at line 138 of file WebradioTranscoder.cs.

6.24.4.4 public bool WebradioManager.WebradioTranscoder.Capture [get], [set]

Gets or sets a value indicating whether the capture.

Returns

true if capture, false if not.

Definition at line 96 of file WebradioTranscoder.cs.

6.24.4.5 public string WebradioManager.WebradioTranscoder.ConfigFilename [get], [set]

Gets or sets the filename of the configuration file.

Returns

The filename of the configuration file.

Definition at line 306 of file WebradioTranscoder.cs.

6.24.4.6 public string WebradioManager.WebradioTranscoder.CurrentTrack [get], [set]

Gets or sets the current track's filename.

Returns

The current track.

Definition at line 110 of file WebradioTranscoder.cs.

6.24.4.7 public int WebradioManager.WebradioTranscoder.Id [get], [set]

Gets or sets the identifier.

Returns

The identifier.

Definition at line 194 of file WebradioTranscoder.cs.

6.24.4.8 public IPAddress WebradioManager.WebradioTranscoder.Ip [get], [set]

Gets or sets the IP.

Returns

The IP.

Definition at line 264 of file WebradioTranscoder.cs.

6.24.4.9 public string WebradioManager.WebradioTranscoder.LogFilename [get], [set]

Gets or sets the filename of the log file.

Returns

The filename of the log file.

Definition at line 320 of file WebradioTranscoder.cs.

6.24.4.10 public string WebradioManager.WebradioTranscoder.Name [get], [set]

Gets or sets the name.

Returns

The name.

Definition at line 236 of file WebradioTranscoder.cs.

6.24.4.11 public string WebradioManager.WebradioTranscoder.Password [get], [set]

Gets or sets the password.

Returns

The password.

Definition at line 292 of file WebradioTranscoder.cs.

6.24.4.12 public int WebradioManager.WebradioTranscoder.Port [get], [set]

Gets or sets the port.

Returns

The port.

Definition at line 278 of file WebradioTranscoder.cs.

6.24.4.13 public Process WebradioManager.WebradioTranscoder.Process [get], [set]

Gets or sets the process.

Returns

The process.

Definition at line 152 of file WebradioTranscoder.cs.

6.24.4.14 public int WebradioManager.WebradioTranscoder.SampleRate [get], [set]

Gets or sets the sample rate.

Returns

The sample rate.

Definition at line 222 of file WebradioTranscoder.cs.

6.24.4.15 public StreamType WebradioManager.WebradioTranscoder.StreamType [get], [set]

Gets or sets the type of the stream.

Returns

The type of the stream.

Definition at line 334 of file WebradioTranscoder.cs.

6.24.4.16 public string WebradioManager.WebradioTranscoder.Url [get], [set]

Gets or sets URL of the document.

Returns

The URL.

Definition at line 250 of file WebradioTranscoder.cs.

The documentation for this class was generated from the following file:

- WebradioTranscoder.cs

6.25 WebradioManager.WMModel Class Reference

A data Model for the WebradioManager project.

Public Member Functions

- WMModel ()
  Default constructor.
- void AddObserver (IController observer)
  Adds an observer.
- void RemoveObserver (IController observer)
  Removes the observer described by observer.
- int GetSimiliarViewCount (int webradioId)
  Gets similiar view count.
- void LoadLibrary ()
  Loads the library.
- void CheckFolders (int webradioId)
  Check folders.
- void LoadWebradios ()
  Loads the webradios.
- Webradio GetWebradio (int id)
  Gets a webradio.
- Webradio GetWebradioByName (string name)
  Gets webradio by name.
- List< Webradio > GetWebradios ()
  Gets the webradios list.
- bool CreateWebradio (string name)
  Creates a webradio.
- bool DeleteWebradio (int id)
  Deletes the webradio described by ID.
- bool DuplicateWebradio (int id)
  Duplicate webradio.
- List< AudioFile > GetLibrary ()
  Gets the library.
- List< string > GetGenders ()
  Gets the genders list.
- bool ImportFilesToLibrary (string[] filenames, AudioType type)
  Import files to library.
- bool DeleteAudioFile (int id, string audioFilename)
  Deletes the audio file.
- bool UpdateAudioFile (AudioFile file)

Updates the audio file with param one's value. Retag file.

- bool CreatePlaylist (string name, string webradioName, int webradioId, AudioType type, out Playlist newPlaylist)

  Creates a playlist.
- bool DeletePlaylist (Playlist playlist, int webradioId)

  Deletes the playlist.
- bool AddToPlaylist (Playlist playlist, Dictionary< int, string > audioFiles)

  Adds to the playlist.
- bool RemoveFromPlaylist (Dictionary< int, string > audioFiles, Playlist playlist)

  Removes from playlist.
- List< AudioFile > GetPlaylistContent (Playlist playlist)

  Gets playlist content.
- bool GeneratePlaylist (string name, TimeSpan duration, AudioType type, string gender, int webradioId, string webradioName)

  Generates a playlist.
- bool CreateEvent (CalendarEvent newEvent, int webradioId)

  Creates an event.
- bool UpdateEvent (CalendarEvent aEvent, int webradioId)

  Updates the event with param one's values.
- bool DeleteEvent (CalendarEvent aEvent, int webradioId)

  Deletes the event.
- bool CreateTranscoder (string name, StreamType st, int sampleRate, int bitrate, string url, IPAddress ip, int port, int adminport, string password, int webradioId)

  Creates a transcoder.
- bool DeleteTranscoder (WebradioTranscoder transcoder, int webradioId)

  Deletes the transcoder.
- bool UpdateTranscoder (WebradioTranscoder transcoder, bool debug, int webradioId)

  Updates the transcoder.
- bool StartTranscoder (WebradioTranscoder transcoder, bool debug, int webradioId)

  Starts a transcoder.
- bool StopTranscoder (WebradioTranscoder transcoder, int webradioId)

  Stops a transcoder.
- bool StopAllProcess (int webradioId)

  Stops all process of a webradio.
- bool StopAllProcess ()

  Stops all process of the program.
- void GenerateConfigFiles (int webradioId)

  Generates a configuration files.
- bool UpdateServer (bool debug, int port, string password, string adminPassword, int maxListener, int webradioId)

  Updates the server.
- bool StartServer (int webradioId, bool debug)

  Starts a server.
- bool StopServer (int webradioId)

  Stops a server.
- void ShowServerWebInterface (int webradioId)

  Shows the server web interface.
- void ShowServerWebAdmin (int webradioId)

  Shows the server web admin.
- bool TranscoderNextTrack (WebradioTranscoder transcoder)

  Transcoder goes next track.

- bool ClearHistory (int transcoderId)

  Clears the transcoder's history described by transcoderId.
- bool GenerateHistory (int webradioId, string transcoderName, int transcoderId, string outputFilename)

  Generates a transcoder's history.
- AudioFile GetAudioFileByFilename (string filename)

  Gets audio file by filename.
- bool ModifyWebradioName (string newName, int webradioId)

  Modify webradio name.
- bool TranscoderCapture (bool active, string device, WebradioTranscoder transcoder, int webradioId)

  Transcoder capture set.
- List< WebradioListener > UpdateServerListeners (int webradioId)

  Updates the webradio's server listeners.
- bool UpdateServerStats (int webradioId)

  Updates the webradio's server statistics.
- bool CheckLibrary ()

  Check if audio files from library exists on the disc.

Public Attributes

- const string **DEFAULT_WEBRADIOS_FOLDER** = "webradios/"
- const string **DEFAULT_SHOUTCAST_FOLDER** = "shoutcast/"
- const string **DEFAULT_CALENDAR_FILENAME** = "calendar.xml"

Properties

- List< WebradioServer > ActiveServers `[get, set]`

  Gets or sets the active servers list.
- List< WebradioTranscoder > ActiveTranscoders `[get, set]`

  Gets or sets the active transcoders list.
- System.Windows.Forms.Timer ProcessWatcher `[get, set]`

  Gets or sets the process watcher.
- Bdd Bdd `[get, set]`

  Gets or sets the bdd.
- List< IController > Observers `[get, set]`

  Gets or sets the observers.
- Dictionary< int, Webradio > Webradios `[get, set]`

  Gets or sets the webradios.
- List< AudioFile > Library `[get, set]`

  Gets or sets the library.

6.25.1 Detailed Description

A data Model for the WebradioManager project.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 28 of file WMModel.cs.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 public WebradioManager.WMModel.WMModel ( )

Default constructor.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 199 of file WMModel.cs.

6.25.3 Member Function Documentation

6.25.3.1 public void WebradioManager.WMModel.AddObserver (IController *observer*)

Adds an observer.

Author

Simon Menetrey

Date

26.05.2014

Parameters

observer	The observer.

Definition at line 309 of file WMModel.cs.

6.25.3.2 public bool WebradioManager.WMModel.AddToPlaylist (Playlist *playlist,* Dictionary< int, string > *audioFiles*)

Adds to the playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

playlist	The playlist.
audioFiles	The audio files.

Returns

true if it succeeds, false if it fails.

Definition at line 1025 of file WMModel.cs.

6.25.3.3 public void WebradioManager.WMModel.CheckFolders (int *webradioId*)

Check folders.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Definition at line 422 of file WMModel.cs.

6.25.3.4 public bool WebradioManager.WMModel.CheckLibrary ( )

Check if audio files from library exists on the disc.

Author

Simon Menetrey

Date

26.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 1930 of file WMModel.cs.

6.25.3.5 public bool WebradioManager.WMModel.ClearHistory (int *transcoderId*)

Clears the transcoder's history described by transcoderId.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoderId	Identifier for the transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 1725 of file WMModel.cs.

6.25.3.6 public bool WebradioManager.WMModel.CreateEvent (CalendarEvent *newEvent,* int *webradioId*)

Creates an event.

Author

Simon Menetrey

Date

26.05.2014

Parameters

newEvent	The new event.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1194 of file WMModel.cs.

6.25.3.7 public bool WebradioManager.WMModel.CreatePlaylist (string *name,* string *webradioName,* int *webradioId,* AudioType *type,* out Playlist *newPlaylist*)

Creates a playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

	name	The name.
	webradioName	Name of the webradio.
	webradioId	Identifier for the webradio.
	type	The type.
out	*newPlaylist*	The new playlist.

Returns

true if it succeeds, false if it fails.

Definition at line 944 of file WMModel.cs.

6.25.3.8 public bool WebradioManager.WMModel.CreateTranscoder (string *name,* StreamType *st,* int *sampleRate,* int *bitrate,* string *url,* IPAddress *ip,* int *port,* int *adminport,* string *password,* int *webradioId*)

Creates a transcoder.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
st	The st.
sampleRate	The sample rate.
bitrate	The bitrate.
url	URL of the document.
ip	The IP.
port	The port.
adminport	The administration port.
password	The password.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1292 of file WMModel.cs.

6.25.3.9 public bool WebradioManager.WMModel.CreateWebradio (string *name*)

Creates a webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.

Returns

true if it succeeds, false if it fails.

Definition at line 526 of file WMModel.cs.

6.25.3.10 public bool WebradioManager.WMModel.DeleteAudioFile (int *id,* string *audioFilename*)

Deletes the audio file.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.
audioFilename	Filename of the audio file.

Returns

true if it succeeds, false if it fails.

Definition at line 802 of file WMModel.cs.

6.25.3.11 public bool WebradioManager.WMModel.DeleteEvent (CalendarEvent *aEvent*, int *webradioId*)

Deletes the event.

Author

Simon Menetrey

Date

26.05.2014

Parameters

aEvent	The event.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1257 of file WMModel.cs.

6.25.3.12 public bool WebradioManager.WMModel.DeletePlaylist (Playlist *playlist*, int *webradioId*)

Deletes the playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

playlist	The playlist.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 997 of file WMModel.cs.

6.25.3.13 public bool WebradioManager.WMModel.DeleteTranscoder (WebradioTranscoder *transcoder,* int *webradioId*)

Deletes the transcoder.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1347 of file WMModel.cs.

6.25.3.14 public bool WebradioManager.WMModel.DeleteWebradio (int *id*)

Deletes the webradio described by ID.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 574 of file WMModel.cs.

6.25.3.15 public bool WebradioManager.WMModel.DuplicateWebradio (int *id*)

Duplicate webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

true if it succeeds, false if it fails.

Definition at line 648 of file WMModel.cs.

6.25.3.16 public void WebradioManager.WMModel.GenerateConfigFiles (int *webradioId*)

Generates a configuration files.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Definition at line 1546 of file WMModel.cs.

6.25.3.17 public bool WebradioManager.WMModel.GenerateHistory (int *webradioId,* string *transcoderName,* int *transcoderId,* string *outputFilename*)

Generates a transcoder's history.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.
transcoderName	Name of the transcoder.
transcoderId	Identifier for the transcoder.
outputFilename	Filename of the output file.

Returns

true if it succeeds, false if it fails.

Definition at line 1746 of file WMModel.cs.

6.25.3.18 public bool WebradioManager.WMModel.GeneratePlaylist (string *name,* TimeSpan *duration,* AudioType *type,* string *gender,* int *webradioId,* string *webradioName*)

Generates a playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.
duration	The duration.
type	The type.
gender	The gender.
webradioId	Identifier for the webradio.
webradioName	Name of the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1127 of file WMModel.cs.

6.25.3.19 public AudioFile WebradioManager.WMModel.GetAudioFileByFilename (string *filename*)

Gets audio file by filename.

Author

Simon Menetrey

Date

26.05.2014

Parameters

filename	Filename of the file.

Returns

The audio file by filename.

Definition at line 1791 of file WMModel.cs.

6.25.3.20 public List< string > WebradioManager.WMModel.GetGenders ( )

Gets the genders list.

Author

Simon Menetrey

Date

26.05.2014

Returns

The genders list.

Definition at line 724 of file WMModel.cs.

6.25.3.21 public List< AudioFile > WebradioManager.WMModel.GetLibrary ( )

Gets the library.

Author

Simon Menetrey

Date

26.05.2014

Returns

The library.

Definition at line 708 of file WMModel.cs.

6.25.3.22 public List< AudioFile > WebradioManager.WMModel.GetPlaylistContent (Playlist *playlist*)

Gets playlist content.

Author

Simon Menetrey

Date

26.05.2014

Parameters

playlist	The playlist.

Returns

The playlist content (AudioFile list).

Definition at line 1095 of file WMModel.cs.

6.25.3.23 public int WebradioManager.WMModel.GetSimiliarViewCount (int *webradioId*)

Gets similiar view count.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Returns

The similiar view count.

Definition at line 343 of file WMModel.cs.

6.25.3.24 public Webradio WebradioManager.WMModel.GetWebradio (int *id*)

Gets a webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

id	The identifier.

Returns

The webradio.

Definition at line 459 of file WMModel.cs.

6.25.3.25 public Webradio WebradioManager.WMModel.GetWebradioByName (string *name*)

Gets webradio by name.

Author

Simon Menetrey

Date

26.05.2014

Parameters

name	The name.

Returns

The webradio by name.

Definition at line 477 of file WMModel.cs.

6.25.3.26 public List< Webradio > WebradioManager.WMModel.GetWebradios ( )

Gets the webradios list.

Author

Simon Menetrey

Date

26.05.2014

Returns

The webradios list.

Definition at line 502 of file WMModel.cs.

6.25.3.27 public bool WebradioManager.WMModel.ImportFilesToLibrary (string[] *filenames*, AudioType *type*)

Import files to library.

Author

Simon Menetrey

Date

26.05.2014

Parameters

filenames	The filenames array.
type	The type.

Returns

true if it succeeds, false if it fails.

Definition at line 743 of file WMModel.cs.

6.25.3.28 public void WebradioManager.WMModel.LoadLibrary ( )

Loads the library.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 406 of file WMModel.cs.

6.25.3.29 public void WebradioManager.WMModel.LoadWebradios ( )

Loads the webradios.

Author

Simon Menetrey

Date

26.05.2014

Definition at line 437 of file WMModel.cs.

6.25.3.30 public bool WebradioManager.WMModel.ModifyWebradioName (string *newName,* int *webradioId*)

Modify webradio name.

Author

Simon Menetrey

Date

26.05.2014

Parameters

newName	Name of the new webradio.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1819 of file WMModel.cs.

6.25.3.31 public bool WebradioManager.WMModel.RemoveFromPlaylist (Dictionary< int, string > *audioFiles,* Playlist *playlist*)

Removes from playlist.

Author

Simon Menetrey

Date

26.05.2014

Parameters

audioFiles	The audio files.
playlist	The playlist.

Returns

true if it succeeds, false if it fails.

Definition at line 1061 of file WMModel.cs.

6.25.3.32 public void WebradioManager.WMModel.RemoveObserver (IController *observer*)

Removes the observer described by observer.

Author

Simon Menetrey

Date

26.05.2014

Parameters

observer	The observer.

Definition at line 325 of file WMModel.cs.

6.25.3.33 public void WebradioManager.WMModel.ShowServerWebAdmin (int *webradioId*)

Shows the server web admin.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Definition at line 1681 of file WMModel.cs.

6.25.3.34 public void WebradioManager.WMModel.ShowServerWebInterface (int *webradioId*)

Shows the server web interface.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Definition at line 1665 of file WMModel.cs.

6.25.3.35 public bool WebradioManager.WMModel.StartServer (int *webradioId,* bool *debug*)

Starts a server.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.
debug	true to debug.

Returns

true if it succeeds, false if it fails.

Definition at line 1617 of file WMModel.cs.

6.25.3.36 public bool WebradioManager.WMModel.StartTranscoder (WebradioTranscoder *transcoder,* bool *debug,* int *webradioId*)

Starts a transcoder.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.
debug	true to debug.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1419 of file WMModel.cs.

6.25.3.37 public bool WebradioManager.WMModel.StopAllProcess (int *webradioId*)

Stops all process of a webradio.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1487 of file WMModel.cs.

6.25.3.38 public bool WebradioManager.WMModel.StopAllProcess ( )

Stops all process of the program.

Author

Simon Menetrey

Date

26.05.2014

Returns

true if it succeeds, false if it fails.

Definition at line 1515 of file WMModel.cs.

6.25.3.39 public bool WebradioManager.WMModel.StopServer (int *webradioId*)

Stops a server.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1642 of file WMModel.cs.

6.25.3.40 public bool WebradioManager.WMModel.StopTranscoder (WebradioTranscoder *transcoder,* int *webradioId*)

Stops a transcoder.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1454 of file WMModel.cs.

6.25.3.41 public bool WebradioManager.WMModel.TranscoderCapture (bool *active,* string *device,* WebradioTranscoder *transcoder,* int *webradioId*)

Transcoder capture set.

Author

Simon Menetrey

Date

26.05.2014

Parameters

active	true to active.
device	The device.
transcoder	The transcoder.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1866 of file WMModel.cs.

6.25.3.42 public bool WebradioManager.WMModel.TranscoderNextTrack (WebradioTranscoder *transcoder*)

Transcoder goes next track.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.

Returns

true if it succeeds, false if it fails.

Definition at line 1699 of file WMModel.cs.

6.25.3.43 public bool WebradioManager.WMModel.UpdateAudioFile (AudioFile *file*)

Updates the audio file with param one's value. Retag file.

Author

Simon Menetrey

Date

26.05.2014

Parameters

file	The file.

Returns

true if it succeeds, false if it fails.

Definition at line 843 of file WMModel.cs.

6.25.3.44 public bool WebradioManager.WMModel.UpdateEvent (CalendarEvent *aEvent,* int *webradioId*)

Updates the event with param one's values.

Author

Simon Menetrey

Date

26.05.2014

Parameters

aEvent	The event.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1221 of file WMModel.cs.

6.25.3.45 public bool WebradioManager.WMModel.UpdateServer (bool *debug,* int *port,* string *password,* string *adminPassword,* int *maxListener,* int *webradioId*)

Updates the server.

Author

Simon Menetrey

Date

26.05.2014

Parameters

debug	true to debug.
port	The port.
password	The password.
adminPassword	The admin password.
maxListener	The maximum listener.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1569 of file WMModel.cs.

6.25.3.46 public List< WebradioListener > WebradioManager.WMModel.UpdateServerListeners (int *webradioId*)

Updates the webradio's server listeners.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Returns

A List<WebradioListener>

Definition at line 1893 of file WMModel.cs.

6.25.3.47 public bool WebradioManager.WMModel.UpdateServerStats (int *webradioId*)

Updates the webradio's server statistics.

Author

Simon Menetrey

Date

26.05.2014

Parameters

webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1911 of file WMModel.cs.

6.25.3.48 public bool WebradioManager.WMModel.UpdateTranscoder (WebradioTranscoder *transcoder,* bool *debug,* int *webradioId*)

Updates the transcoder.

Author

Simon Menetrey

Date

26.05.2014

Parameters

transcoder	The transcoder.
debug	true to debug.
webradioId	Identifier for the webradio.

Returns

true if it succeeds, false if it fails.

Definition at line 1380 of file WMModel.cs.

6.25.4 Property Documentation

6.25.4.1 public List< WebradioServer > WebradioManager.WMModel.ActiveServers [get], [set]

Gets or sets the active servers list.

Returns

The active servers.

Definition at line 98 of file WMModel.cs.

6.25.4.2 public List< WebradioTranscoder > WebradioManager.WMModel.ActiveTranscoders [get], [set]

Gets or sets the active transcoders list.

Returns

The active transcoders.

Definition at line 112 of file WMModel.cs.

6.25.4.3 public Bdd WebradioManager.WMModel.Bdd [get], [set]

Gets or sets the bdd.

Returns

The bdd.

Definition at line 140 of file WMModel.cs.

6.25.4.4 public List< AudioFile > WebradioManager.WMModel.Library [get], [set]

Gets or sets the library.

Returns

The library.

Definition at line 182 of file WMModel.cs.

6.25.4.5 public List< IController > WebradioManager.WMModel.Observers [get], [set]

Gets or sets the observers.

Returns

The observers.

Definition at line 154 of file WMModel.cs.

6.25.4.6 public System Windows Forms Timer WebradioManager.WMModel.ProcessWatcher [get], [set]

Gets or sets the process watcher.

Returns

The process watcher.

Definition at line 126 of file WMModel.cs.

6.25.4.7 public Dictionary< int, Webradio > WebradioManager.WMModel.Webradios [get], [set]

Gets or sets the webradios.

Returns

The webradios.

Definition at line 168 of file WMModel.cs.

The documentation for this class was generated from the following file:

- WMModel.cs

Chapter 7

File Documentation

7.1 Ad.cs File Reference

Implements the ad class.

Classes

- class WebradioManager.Ad

  An ad audio file.

Namespaces

- package WebradioManager

7.1.1 Detailed Description

Implements the ad class.

Definition in file Ad.cs.

7.2 Ad.cs

```
00001
00007 using System;
00008
00009 namespace WebradioManager
00010 {
00020     public class Ad : AudioFile
00021     {
00022         #region Methods
00023
00042         public Ad(int id, string filename, string title, string artist, string album, int year, string
      label, TimeSpan duration, string gender):
00043             base(id,filename,title,artist,album,year,label,duration,gender,
      AudioType.Ad)
00044         {
00045
00046         }
00047
00066         public Ad(string filename, string title, string artist, string album, int year, string label,
      TimeSpan duration, string gender) :
00067             base(filename, title, artist, album, year, label, duration, gender,
      AudioType.Ad)
00068         {
00069
00070         }
00071         #endregion
```

```
00072     }
00073 }
```

7.3 AdminController.cs File Reference

Implements the admin controller class.

Classes

- class WebradioManager.AdminController

  A controller for handling admin view.

Namespaces

- package WebradioManager

7.3.1 Detailed Description

Implements the admin controller class.

Definition in file AdminController.cs.

7.4 AdminController.cs

```
00001
00007 using System;
00008 using System.Collections.Generic;
00009 using System.Net;
00010
00011 namespace WebradioManager
00012 {
00022     public class AdminController : IController
00023     {
00024         #region Fields
00025         // \brief The model.
00026         private WMModel _model;
00027         // \brief The view.
00028         private AdminView _view;
00029         #endregion
00030
00031         #region Properties
00032
00041         public AdminView View
00042         {
00043             get { return _view; }
00044             set { _view = value; }
00045         }
00046
00055         public WMModel Model
00056         {
00057             get { return _model; }
00058             set { _model = value; }
00059         }
00060         #endregion
00061
00062         #region Methods
00063
00076         public AdminController(int id, WMModel model)
00077         {
00078             this.View = new AdminView(id,this);
00079             this.Model = model;
00080             this.UpdateView();
00081             this.View.Show();
00082         }
00083
00093         public void UpdateView()
00094         {
00095             this.View.UpdateView();
```

```
00096         }
00097
00111         public Webradio GetWebradio(int id)
00112         {
00113             return this.Model.GetWebradio(id);
00114         }
00115
00127         public List<AudioFile> GetLibrary()
00128         {
00129             return this.Model.GetLibrary();
00130         }
00131
00143         public List<string> GetGenders()
00144         {
00145             return this.Model.GetGenders();
00146         }
00147
00159         public void CheckFolders(int webradioId)
00160         {
00161             this.Model.CheckFolders(webradioId);
00162         }
00163
00173         public void FormClose()
00174         {
00175             this.Model.RemoveObserver(this);
00176         }
00177
00192         public bool ImportFilesToLibrary(string[] filenames,
      AudioType type)
00193         {
00194             return this.Model.ImportFilesToLibrary(filenames, type);
00195         }
00196
00211         public bool DeleteAudioFile(int id,string filename)
00212         {
00213             return this.Model.DeleteAudioFile(id,filename);
00214         }
00215
00232         public bool CreatePlaylist(string name, string webradioName, int webradioId,
      AudioType type)
00233         {
00234             Playlist newPlaylist;
00235             return this.Model.CreatePlaylist(name,webradioName,webradioId,type, out newPlaylist);
00236         }
00237
00252         public bool DeletePlaylist(Playlist playlist, int webradioId)
00253         {
00254             return this.Model.DeletePlaylist(playlist, webradioId);
00255         }
00256
00271         public bool AddToPlaylist(Playlist playlist, Dictionary<int, string>
      audioFiles)
00272         {
00273             if (playlist != null)
00274                 return this.Model.AddToPlaylist(playlist, audioFiles);
00275             else
00276                 return false;
00277         }
00278
00293         public bool RemoveFromPlaylist(Playlist playlist, Dictionary<int,string>
      audioFiles)
00294         {
00295             return this.Model.RemoveFromPlaylist(audioFiles, playlist);
00296         }
00297
00311         public List<AudioFile> GetPlaylistContent(Playlist playlist)
00312         {
00313             return this.Model.GetPlaylistContent(playlist);
00314         }
00315
00334         public bool GeneratePlaylist(string name, TimeSpan duration,
      AudioType type, string gender, int webradioId, string webradioName)
00335         {
00336             return this.Model.GeneratePlaylist(name, duration, type, gender, webradioId, webradioName);
00337         }
00338
00353         public bool CreateEvent(CalendarEvent newEvent, int webradioId)
00354         {
00355             return this.Model.CreateEvent(newEvent, webradioId);
00356         }
00357
00372         public bool UpdateEvent(CalendarEvent aEvent, int webradioId)
00373         {
00374             return this.Model.UpdateEvent(aEvent,webradioId);
00375         }
00376
00391         public bool DeleteEvent(CalendarEvent aEvent, int webradioId)
```

```
00392         {
00393             return this.Model.DeleteEvent(aEvent, webradioId);
00394         }
00395
00418         public bool CreateTranscoder(string name, StreamType st, int sampleRate,
      int bitrate, string url, IPAddress ip, int port, int adminport, string password, int webradioId)
00419         {
00420             return this.Model.CreateTranscoder(name, st, sampleRate, bitrate, url, ip, port, adminport,
      password, webradioId);
00421         }
00422
00437         public bool DeleteTranscoder(WebradioTranscoder transcoder, int
      webradioId)
00438         {
00439             return this.Model.DeleteTranscoder(transcoder, webradioId);
00440         }
00441
00457         public bool UpdateTranscoder(WebradioTranscoder transcoder, bool
      debug, int webradioId)
00458         {
00459             return this.Model.UpdateTranscoder(transcoder, debug, webradioId);
00460         }
00461
00477         public bool StartTranscoder(WebradioTranscoder transcoder,bool
      debug, int webradioId)
00478         {
00479             return this.Model.StartTranscoder(transcoder, debug, webradioId);
00480         }
00481
00496         public bool StopTranscoder(WebradioTranscoder transcoder, int
      webradioId)
00497         {
00498             return this.Model.StopTranscoder(transcoder,webradioId);
00499         }
00500
00514         public bool StopAllTranscoders(int webradioId)
00515         {
00516             return this.Model.StopAllProcess(webradioId);
00517         }
00518
00530         public void GenerateAllConfigs(int webradioId)
00531         {
00532             this.Model.GenerateConfigFiles(webradioId);
00533         }
00534
00553         public bool UpdateServer(bool debug, int port, string password, string adminPassword,
      int maxListener, int webradioId)
00554         {
00555             return this.Model.UpdateServer(debug, port, password, adminPassword, maxListener, webradioId);
00556         }
00557
00572         public bool StartServer(int webradioId, bool debug)
00573         {
00574             return this.Model.StartServer(webradioId, debug);
00575         }
00576
00590         public bool StopServer(int webradioId)
00591         {
00592             return this.Model.StopServer(webradioId);
00593         }
00594
00606         public void ShowServerWebInterface(int webradioId)
00607         {
00608             this.Model.ShowServerWebInterface(webradioId);
00609         }
00610
00622         public void ShowServerWebAdmin(int webradioId)
00623         {
00624             this.Model.ShowServerWebAdmin(webradioId);
00625         }
00626
00640         public bool TranscoderNextTrack(WebradioTranscoder transcoder)
00641         {
00642             return this.Model.TranscoderNextTrack(transcoder);
00643         }
00644
00658         public bool ClearHistory(int transcoderId)
00659         {
00660             return this.Model.ClearHistory(transcoderId);
00661         }
00662
00679         public bool GenerateHistory(int webradioId, string transcoderName, int transcoderId,
       string outputFilename)
00680         {
00681             return this.Model.GenerateHistory(webradioId, transcoderName, transcoderId, outputFilename);
00682         }
00683
```

```
00698         public bool ModifyWebradioName(string name, int webradioId)
00699         {
00700             return this.Model.ModifyWebradioName(name, webradioId);
00701         }
00702 
00716         public AudioFile GetAudioFileByFilename(string filename)
00717         {
00718             return this.Model.GetAudioFileByFilename(filename);
00719         }
00720 
00734         public bool UpdateAudioFile(AudioFile file)
00735         {
00736             return this.Model.UpdateAudioFile(file);
00737         }
00738 
00755         public bool TranscoderCapture(bool active, string device,
      WebradioTranscoder transcoder,int webradioId)
00756         {
00757             return this.Model.TranscoderCapture(active, device, transcoder, webradioId);
00758         }
00759 
00773         public List<WebradioListener> GetServerListeners(int webradioId)
00774         {
00775             return this.Model.UpdateServerListeners(webradioId);
00776         }
00777 
00791         public bool UpdateServerStats(int webradioId)
00792         {
00793             return this.Model.UpdateServerStats(webradioId);
00794         }
00795 
00807         public bool CheckLibrary()
00808         {
00809             return this.Model.CheckLibrary();
00810         }
00811 
00825         public int GetSimilarViewCount(int webradioId)
00826         {
00827             return this.Model.GetSimiliarViewCount(webradioId);
00828         }
00829 
00830         #endregion
00831 
00832 
00833     }
00834 }
```

7.5 AdminView.cs File Reference

Implements the admin view class.

Classes

- class WebradioManager.AdminView

  An admin view.

Namespaces

- package WebradioManager

7.5.1 Detailed Description

Implements the admin view class.

Definition in file AdminView.cs.

7.6 AdminView.cs

```
00001
00007 using Calendar;
00008 using System;
00009 using System.Collections.Generic;
00010 using System.Diagnostics;
00011 using System.Drawing;
00012 using System.IO;
00013 using System.Management;
00014 using System.Net;
00015 using System.Net.Sockets;
00016 using System.Windows.Forms;
00017
00018 namespace WebradioManager
00019 {
00029     public partial class AdminView : Form
00030     {
00031         #region Const
00032         // \brief The default search string.
00033         const string DEFAULT_SEARCH_STRING = "Search...";
00034         // \brief The maximum name length.
00035         const int MAX_NAME_LENGTH = 255;
00036         #endregion
00037
00038         #region Fields
00039         // \brief The controller.
00040         private AdminController _controller;
00041         // \brief The identifier of the current webradio.
00042         private int _idWebradio;
00043         // \brief The webradio's name.
00044         private string _nameWebradio;
00045         // \brief The events for calendar.
00046         List<EventAppointment> _events;
00047         #endregion
00048
00049         #region Properties
00050
00059         public string NameWebradio
00060         {
00061             get { return _nameWebradio; }
00062             set { _nameWebradio = value; }
00063         }
00064
00073         public List<EventAppointment> EventsCalendar
00074         {
00075             get { return _events; }
00076             set { _events = value; }
00077         }
00078
00087         public int IdWebradio
00088         {
00089             get { return _idWebradio; }
00090             set { _idWebradio = value; }
00091         }
00092
00101         public AdminController Controller
00102         {
00103             get { return _controller; }
00104             set { _controller = value; }
00105         }
00106         #endregion
00107
00108         #region Methods
00109
00121         public AdminView(int idWebradio, AdminController controller)
00122         {
00123             InitializeComponent();
00124             this.Controller = controller;
00125             this.IdWebradio = idWebradio;
00126             this.EventsCalendar = new List<EventAppointment>();
00127             this.UpdateAudioDevices();
00128         }
00129
00139         public void UpdateView()
00140         {
00141             int index;
00142             Webradio webradio = this.Controller.GetWebradio(this.IdWebradio);
00143             this.NameWebradio = webradio.Name;
00144             this.Text = "WebradioManager - " + this.NameWebradio;
00145             txbWebradioName.Text = this.NameWebradio;
00146             lblWebradioTitle.Text = this.NameWebradio;
00147             cmbTypePlaylist.SelectedIndex = 0;
00148             cmbTypePlaylistGenerate.SelectedIndex = 0;
00149
00150             //LIBRARY
```

```
00151             dgvMusics.Rows.Clear();
00152             dgvAds.Rows.Clear();
00153             List<AudioFile> audiofiles = this.Controller.GetLibrary();
00154             foreach (AudioFile file in audiofiles)
00155             {
00156                 if (file is Music)
00157                     dgvMusics.Rows.Add(file.GetInfosArray());
00158                 else if (file is Ad)
00159                     dgvAds.Rows.Add(file.GetInfosArray());
00160             }
00161             txbSearchAd.Text = (txbSearchAd.Text != DEFAULT_SEARCH_STRING) ? txbSearchAd.Text :
      DEFAULT_SEARCH_STRING;
00162             txbSearchMusic.Text = (txbSearchMusic.Text != DEFAULT_SEARCH_STRING) ? txbSearchMusic.Text :
      DEFAULT_SEARCH_STRING;
00163             this.txbSearchTextChanged(txbSearchAd, new EventArgs());
00164             this.txbSearchTextChanged(txbSearchMusic, new EventArgs());
00165             //----
00166             //PLAYLIST
00167             lblPlaylistDuration.Text = "";
00168             cmbPlaylistsMusic.Items.Clear();
00169             cmbPlaylistsAd.Items.Clear();
00170             cmbPlaylistEvent.Items.Clear();
00171             lsbPlaylistsAd.Items.Clear();
00172             lsbPlaylistsMusic.Items.Clear();
00173             dgvPlaylistContent.Rows.Clear();
00174 
00175             cmbPlaylistEvent.Items.AddRange(webradio.Playlists.ToArray());
00176             foreach (Playlist playlist in webradio.Playlists)
00177             {
00178                 if (playlist is PlaylistMusic)
00179                 {
00180                     cmbPlaylistsMusic.Items.Add(playlist);
00181                     lsbPlaylistsMusic.Items.Add(playlist);
00182                 }
00183                 else if (playlist is PlaylistAd)
00184                 {
00185                     cmbPlaylistsAd.Items.Add(playlist);
00186                     lsbPlaylistsAd.Items.Add(playlist);
00187                 }
00188             }
00189             if (cmbPlaylistsAd.Items.Count > 0)
00190                 cmbPlaylistsAd.SelectedIndex = 0;
00191             if (cmbPlaylistsMusic.Items.Count > 0)
00192                 cmbPlaylistsMusic.SelectedIndex = 0;
00193             if (cmbPlaylistEvent.Items.Count > 0)
00194                 cmbPlaylistEvent.SelectedIndex = 0;
00195             lsbPlaylistsAd.SelectedIndex = -1;
00196             lsbPlaylistsMusic.SelectedIndex = -1;
00197             //----
00198             //GENDER
00199             List<string> gender = this.Controller.GetGenders();
00200             cmbGenderGenerate.Items.Clear();
00201             cmbGenderGenerate.Items.AddRange(gender.ToArray());
00202             if (cmbGenderGenerate.Items.Count > 0)
00203                 cmbGenderGenerate.SelectedIndex = 0;
00204             //----
00205             //EVENTS
00206             this.EventsCalendar.Clear();
00207             dvwTimetable.Refresh();
00208             dvwTimetable.Invalidate();
00209             foreach (CalendarEvent ev in webradio.Calendar.Events)
00210             {
00211                 DayWeek dw = ev.GetSelectedDays();
00212                 bool[] days = dw.ToArray();
00213                 for (int i = 0; i < 7; i++)
00214                 {
00215                     if (days[i])
00216                     {
00217                         EventAppointment m_Appointment = new
      EventAppointment();
00218 
00219                         m_Appointment.StartDate = new DateTime(dvwTimetable.StartDate.Year, dvwTimetable.
      StartDate.Month, (i + 1), ev.StartTime.Hours, ev.StartTime.Minutes, ev.
      StartTime.Seconds);
00220                         m_Appointment.EndDate = m_Appointment.StartDate.Add(ev.Duration);
00221                         m_Appointment.Title = ev.Name + "(" + ev.Priority.ToString() + ")\r\n " + ev.
      Playlist.Name + "\r\n Shuffle : " + ((ev.Shuffle) ? "True" : "False");
00222                         if (ev.Playlist.Type == AudioType.Music)
00223                         {
00224                             m_Appointment.BorderColor = Color.Blue;
00225                             //m_Appointment.Color = Color.Blue;
00226                         }
00227                         else
00228                         {
00229                             m_Appointment.BorderColor = Color.Red;
00230                             //m_Appointment.Color = Color.Red;
00231                         }
```

```
00232                         m_Appointment.Playlist = ev.Playlist;
00233                         m_Appointment.EventObject = ev;
00234                         this.EventsCalendar.Add(m_Appointment);
00235                     }
00236                 }
00237             }
00238             //----
00239             //TRANSCODERS
00240             index = lsbTranscoders.SelectedIndex;
00241             lsbTranscoders.Items.Clear();
00242             lsbTranscoders.Items.AddRange(webradio.Transcoders.ToArray());
00243             lsbTranscoders.SelectedIndex = (index >= lsbTranscoders.Items.Count)?-1:index;
00244             cmbBitrate.Items.Clear();
00245             index = cmbBitrateEdit.SelectedIndex;
00246             cmbBitrateEdit.Items.Clear();
00247             foreach (int bitrate in WebradioTranscoder.AvaliableBitrates)
00248             {
00249                 cmbBitrate.Items.Add((bitrate / 1000));
00250                 cmbBitrateEdit.Items.Add((bitrate / 1000));
00251             }
00252             cmbBitrate.SelectedIndex = 0;
00253             cmbBitrateEdit.SelectedIndex = index;
00254 
00255             cmbSampleRate.Items.Clear();
00256             index = cmbSampleRateEdit.SelectedIndex;
00257             cmbSampleRateEdit.Items.Clear();
00258             foreach (int samplerate in WebradioTranscoder.AvaliableSampleRates)
00259             {
00260                 cmbSampleRate.Items.Add(samplerate.ToString());
00261                 cmbSampleRateEdit.Items.Add(samplerate.ToString());
00262             }
00263             cmbSampleRate.SelectedIndex = 0;
00264             cmbSampleRateEdit.SelectedIndex = index;
00265 
00266             cmbEncoder.SelectedIndex = 0;
00267 
00268             lsbStatus.Items.Clear();
00269             dgvCurrentTracks.Rows.Clear();
00270             foreach (WebradioTranscoder transcoder in webradio.Transcoders)
00271             {
00272                 lsbStatus.Items.Add(transcoder.Name + " : " + ((transcoder.IsRunning()) ? "On" : "Off"));
00273                 if (transcoder.IsRunning())
00274                 {
00275                     AudioFile file = this.Controller.GetAudioFileByFilename(
      transcoder.CurrentTrack);
00276                     if (file != null)
00277                     {
00278                         string[] values = new string[] { transcoder.Name, file.Title, file.Artist,
      file.Album };
00279                         dgvCurrentTracks.Rows.Add(values);
00280                     }
00281                 }
00282             }
00283 
00284             //----
00285             //SERVER
00286             nudPortServer.Value = webradio.Server.Port;
00287             txbLocalServerPassword.Text = webradio.Server.Password;
00288             txbLocalServerAdminPassword.Text = webradio.Server.AdminPassword;
00289             nudMaxListener.Value = webradio.Server.MaxListener;
00290             bool running = webradio.Server.IsRunning();
00291             lblStatusServer.Text = (running) ? "On" : "Off";
00292             lblStatusServer.ForeColor = (running) ? Color.Green : Color.Red;
00293             btnStartServer.Enabled = (running) ? false : true;
00294             btnStopServer.Enabled = (running) ? true : false;
00295 
00296             if (webradio.Server.Stats != null)
00297                 this.ShowServerStats(webradio.Server.Stats);
00298             //----
00299         }
00300 
00312         private void ShowServerStats(WebradioServerStats stats)
00313         {
00314             lblNumberListeners.Text = stats.CurrentListeners.ToString();
00315             lblPeakListeners.Text = stats.PeakListeners.ToString();
00316             lblUniqueListeners.Text = stats.UniqueListeners.ToString();
00317             lblAverageTime.Text = stats.AverageTime.ToString();
00318         }
00319 
00329         private void UpdateAudioDevices()
00330         {
00331             ManagementObjectSearcher objSearcher = new ManagementObjectSearcher("SELECT * FROM
       Win32_SoundDevice");
00332             ManagementObjectCollection objCollection = objSearcher.Get();
00333             cmbAudioDevice.Items.Clear();
00334             foreach (ManagementObject obj in objCollection)
00335             {
```

```
00336                 foreach (PropertyData property in obj.Properties)
00337                 {
00338                     if (property.Name == "Caption")
00339                         cmbAudioDevice.Items.Add(property.Value);
00340                 }
00341             }
00342 
00343             if (cmbAudioDevice.Items.Count > 0)
00344                 cmbAudioDevice.SelectedIndex = 0;
00345         }
00346 
00359         void dvwTimetable_SelectionChanged(object sender, EventArgs e)
00360         {
00361             ckbMonday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Monday) ? true : false;
00362             ckbTuesday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Tuesday) ? true :
      false;
00363             ckbWednesday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Wednesday) ? true :
      false;
00364             ckbThursday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Thursday) ? true :
      false;
00365             ckbFriday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Friday) ? true : false;
00366             ckbSaturday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Saturday) ? true :
      false;
00367             ckbSunday.Checked = (dvwTimetable.SelectionStart.DayOfWeek == DayOfWeek.Sunday) ? true : false;
00368 
00369             TimeSpan start = new TimeSpan();
00370             TimeSpan end = new TimeSpan();
00371             if (dvwTimetable.Selection == SelectionType.Appointment)
00372             {
00373                 start = new TimeSpan(dvwTimetable.SelectedAppointment.StartDate.Hour, dvwTimetable.
      SelectedAppointment.StartDate.Minute, dvwTimetable.SelectedAppointment.StartDate.Second);
00374                 end = new TimeSpan(dvwTimetable.SelectedAppointment.EndDate.Hour, dvwTimetable.
      SelectedAppointment.EndDate.Minute, dvwTimetable.SelectedAppointment.EndDate.Second);
00375             }
00376             else if (dvwTimetable.Selection == SelectionType.DateRange)
00377             {
00378                 start = new TimeSpan(dvwTimetable.SelectionStart.Hour, dvwTimetable.SelectionStart.Minute,
      dvwTimetable.SelectionStart.Second);
00379                 end = new TimeSpan(dvwTimetable.SelectionEnd.Hour, dvwTimetable.SelectionEnd.Minute,
      dvwTimetable.SelectionEnd.Second);
00380             }
00381 
00382 
00383             mtbStartTime.Text = start.ToString();
00384             mtbDuration.Text = (end - start).ToString();
00385         }
00386 
00399         private void dvwTimetable_ResolveAppointments(object sender, ResolveAppointmentsEventArgs args)
00400         {
00401             List<Appointment> m_Apps = new List<Appointment>();
00402             foreach (Appointment m_App in this.EventsCalendar)
00403                 m_Apps.Add(m_App);
00404 
00405             args.Appointments = m_Apps;
00406 
00407         }
00408 
00421         private void AdminView_FormClosing(object sender, FormClosingEventArgs e)
00422         {
00423             if (this.Controller.GetSimilarViewCount(this.IdWebradio) == 1)
00424             {
00425                 if (MessageBox.Show("All transcoders and the server will be shuting down. Are you sure ?",
      "Close", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
00426                 {
00427                     if (this.Controller.StopAllTranscoders(this.IdWebradio))
00428                         this.Controller.FormClose();
00429                     else
00430                     {
00431                         MessageBox.Show("An error has occured", "Error");
00432                         e.Cancel = true;
00433                     }
00434                 }
00435                 else
00436                     e.Cancel = true;
00437             }
00438             else
00439             {
00440                 this.Controller.FormClose();
00441             }
00442          }
00443 
00456         private void ImportFolder_Click(object sender, EventArgs e)
00457         {
00458             AudioType type;
00459             string[] filenames;
00460             if ((sender as Button).Tag.ToString() == AudioType.Music.ToString())
00461                 type = AudioType.Music;
```

```
00462             else
00463                 type = AudioType.Ad;
00464             if (FBD.ShowDialog() == System.Windows.Forms.DialogResult.OK)
00465             {
00466                 //Recusively
00467                 if (MessageBox.Show("Do you want to search recursively ?", "Recursively", MessageBoxButtons
      .YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
00468                     filenames = Directory.GetFiles(FBD.SelectedPath, "*.mp3", SearchOption.AllDirectories);
00469                 else
00470                     filenames = Directory.GetFiles(FBD.SelectedPath, "*.mp3", SearchOption.TopDirectoryOnly
      );
00471
00472                 if (this.Controller.ImportFilesToLibrary(filenames, type))
00473                     MessageBox.Show("Importation completed", "Success");
00474                 else
00475                     MessageBox.Show("An error occured", "Error");
00476             }
00477         }
00478
00491         private void ImportFiles_Click(object sender, EventArgs e)
00492         {
00493             AudioType type;
00494             if ((sender as Button).Tag.ToString() == AudioType.Music.ToString())
00495                 type = AudioType.Music;
00496             else
00497                 type = AudioType.Ad;
00498             if (OFD.ShowDialog() == System.Windows.Forms.DialogResult.OK)
00499             {
00500                 if (this.Controller.ImportFilesToLibrary(OFD.FileNames, type)
      )
00501                     MessageBox.Show("Importation completed", "Success");
00502                 else
00503                     MessageBox.Show("An error occured", "Error");
00504             }
00505         }
00506
00519         private void txbSearchEnter(object sender, EventArgs e)
00520         {
00521             TextBox txb = (sender as TextBox);
00522             if (txb.Text == DEFAULT_SEARCH_STRING)
00523                 txb.Text = "";
00524         }
00525
00538         private void txbSearchLeave(object sender, EventArgs e)
00539         {
00540             TextBox txb = (sender as TextBox);
00541             if (!string.IsNullOrEmpty(txb.Text))
00542                 txb.Text = "Search...";
00543         }
00544
00558         private void btnDeleteLibrary_Click(object sender, EventArgs e)
00559         {
00560             AudioType type;
00561             bool state = true;
00562             Button btn = (sender as Button);
00563             if (btn.Tag.ToString() == AudioType.Music.ToString())
00564                 type = AudioType.Music;
00565             else
00566                 type = AudioType.Ad;
00567
00568             DataGridViewSelectedRowCollection rows = (btn.Tag.ToString() ==
      AudioType.Music.ToString()) ? dgvMusics.SelectedRows : dgvAds.SelectedRows;
00569             foreach (DataGridViewRow row in rows)
00570             {
00571                 //Simple check if selected row is not the last empty row
00572                 if (row.Cells[0].Value != null)
00573                     if (!this.Controller.DeleteAudioFile(int.Parse(row.Cells["
      colId" + type.ToString()].Value.ToString()), row.Cells["colPath" + type.ToString()].Value.ToString()))
00574                         state = false;
00575             }
00576
00577             if (state)
00578                 this.UpdateView();
00579             else
00580                 MessageBox.Show("An error occured");
00581         }
00582
00595         private void txbSearchTextChanged(object sender, EventArgs e)
00596         {
00597             AudioType type;
00598             bool valid = false;
00599             string searchString = "";
00600             TextBox txb = sender as TextBox;
00601             if (txb.Tag.ToString() == AudioType.Music.ToString())
00602                 type = AudioType.Music;
00603             else
00604                 type = AudioType.Ad;
```

```
00605
00606
00607             if (!string.IsNullOrEmpty(txb.Text) && txb.Text != DEFAULT_SEARCH_STRING)
00608             {
00609                 searchString = txb.Text.ToLower();
00610                 foreach (DataGridViewRow row in ((type == AudioType.Music) ? dgvMusics.Rows :
      dgvAds.Rows))
00611                 {
00612                     foreach (DataGridViewCell cell in row.Cells)
00613                     {
00614                         if (cell.Value.ToString().ToLower().Contains(searchString))
00615                         {
00616                             valid = true;
00617                             break;
00618                         }
00619                     }
00620                     row.Visible = (valid) ? true : false;
00621                     valid = false;
00622                 }
00623             }
00624         }
00625
00639         private void btnCreatePlaylist_Click(object sender, EventArgs e)
00640         {
00641             if (!string.IsNullOrEmpty(txbPlaylistName.Text.Trim()) && txbPlaylistName.Text.Length <=
      MAX_NAME_LENGTH)
00642             {
00643                 if (!this.Controller.CreatePlaylist(txbPlaylistName.Text, this.
      NameWebradio, this.IdWebradio, (cmbTypePlaylist.SelectedItem.ToString() == "Music") ?
      AudioType.Music : AudioType.Ad))
00644                     MessageBox.Show("Playlist already exist or the name is invalid", "Error");
00645             }
00646             else
00647                 MessageBox.Show("Please enter a valid playlist's name. (1-" + MAX_NAME_LENGTH + "
       characters)", "Empty name");
00648         }
00649
00663         private void btnDeletePlaylistClick(object sender, EventArgs e)
00664         {
00665             AudioType type;
00666             type = ((sender as Button).Tag.ToString() == AudioType.Music.ToString()) ? AudioType.Music :
      AudioType.Ad;
00667
00668             if (((type == AudioType.Music) ? lsbPlaylistsMusic.SelectedIndex :
      lsbPlaylistsAd.SelectedIndex) >= 0)
00669                 this.Controller.DeletePlaylist((Playlist)((type == AudioType.Music) ?
      lsbPlaylistsMusic.SelectedItem : lsbPlaylistsAd.SelectedItem), this.IdWebradio);
00670             else
00671                 MessageBox.Show("Please select a playlist to delete.", "No playlist selected");
00672         }
00673
00687         private void btnAddToClick(object sender, EventArgs e)
00688         {
00689             AudioType type = ((sender as Button).Tag.ToString() == AudioType.Music.ToString()) ?
      AudioType.Music : AudioType.Ad;
00690             Dictionary<int, string> audioFiles = new Dictionary<int, string>();
00691             DataGridViewSelectedRowCollection rows = ((type == AudioType.Music) ? dgvMusics.SelectedRows :
      dgvAds.SelectedRows);
00692             this.Cursor = Cursors.WaitCursor;
00693             foreach (DataGridViewRow row in rows)
00694             {
00695                 //Simple check if selected row is not the last empty row
00696                 if (row.Cells[0].Value != null)
00697                     audioFiles.Add(int.Parse(row.Cells["colId" + type.ToString()].Value.ToString()), row.
      Cells["colPath" + type.ToString()].Value.ToString());
00698
00699             }
00700             this.Cursor = Cursors.Default;
00701             if (this.Controller.AddToPlaylist((Playlist)((type ==
      AudioType.Music) ? cmbPlaylistsMusic.SelectedItem : cmbPlaylistsAd.SelectedItem),
00702                 audioFiles))
00703                 this.UpdateView();
00704
00705         }
00706
00719         private void lsbPlaylistsSelectedIndexChanged(object sender, EventArgs e)
00720         {
00721             ListBox lsb = sender as ListBox;
00722             AudioType type = (lsb.Tag.ToString() == AudioType.Music.ToString()) ?
      AudioType.Music : AudioType.Ad;
00723             btnRemoveFromPlaylist.Tag = type;
00724             if (lsb.SelectedIndex >= 0)
00725             {
00726                 this.GetPlaylistContent((Playlist)lsb.SelectedItem);
00727             }
00728         }
00729
```

```
00741         private void GetPlaylistContent(Playlist playlist)
00742         {
00743             dgvPlaylistContent.Rows.Clear();
00744             TimeSpan totalDuration = new TimeSpan(0, 0, 0);
00745             List<AudioFile> audioFiles = this.Controller.GetPlaylistContent(playlist);
00746             foreach (AudioFile af in audioFiles)
00747             {
00748                 totalDuration += af.Duration;
00749                 dgvPlaylistContent.Rows.Add(af.GetInfosArray());
00750             }
00751             lblPlaylistDuration.Text = "Duration : " + totalDuration.ToString();
00752         }
00753 
00767         private void btnRemoveFromPlaylist_Click(object sender, EventArgs e)
00768         {
00769             if (btnRemoveFromPlaylist.Tag == null)
00770             {
00771                 MessageBox.Show("No playlist selected", "No playlist");
00772                 return;
00773             }
00774             AudioType type = (AudioType)btnRemoveFromPlaylist.Tag;
00775             Dictionary<int, string> audioFiles = new Dictionary<int, string>();
00776             this.Cursor = Cursors.WaitCursor;
00777             foreach (DataGridViewRow row in dgvPlaylistContent.SelectedRows)
00778             {
00779                 //Simple check if selected row is not the last empty row
00780                 if (row.Cells[0].Value != null)
00781                     audioFiles.Add(int.Parse(row.Cells["colIdPlaylist"].Value.ToString()), row.Cells["
      colPathPlaylist"].Value.ToString());
00782 
00783             }
00784             this.Cursor = Cursors.Default;
00785             Playlist selectedPlaylist = (Playlist)((type == AudioType.Music) ?
      lsbPlaylistsMusic.SelectedItem : lsbPlaylistsAd.SelectedItem);
00786             if (!this.Controller.RemoveFromPlaylist(selectedPlaylist,
      audioFiles))
00787                 MessageBox.Show("An error occured", "Error");
00788             else
00789             {
00790                 this.GetPlaylistContent(selectedPlaylist);
00791             }
00792 
00793         }
00794 
00807         private void txbSearchPlaylistContent_TextChanged(object sender, EventArgs e)
00808         {
00809             string searchString = "";
00810             bool valid = false;
00811             if (!string.IsNullOrEmpty(txbSearchPlaylistContent.Text) && txbSearchPlaylistContent.Text !=
      DEFAULT_SEARCH_STRING)
00812             {
00813                 searchString = txbSearchPlaylistContent.Text.ToLower();
00814                 foreach (DataGridViewRow row in dgvPlaylistContent.Rows)
00815                 {
00816                     foreach (DataGridViewCell cell in row.Cells)
00817                     {
00818                         if (cell.Value.ToString().ToLower().Contains(searchString))
00819                         {
00820                             valid = true;
00821                             break;
00822                         }
00823                     }
00824                     row.Visible = (valid) ? true : false;
00825                     valid = false;
00826                 }
00827             }
00828         }
00829 
00843         private void cmbTypePlaylistGenerate_SelectedIndexChanged(object sender, EventArgs e)
00844         {
00845             if (cmbTypePlaylistGenerate.SelectedItem.ToString() == AudioType.Ad.ToString())
00846                 cmbGenderGenerate.Enabled = false;
00847             else
00848                 cmbGenderGenerate.Enabled = true;
00849         }
00850 
00863         private void btnGeneratePlaylist_Click(object sender, EventArgs e)
00864         {
00865             if (!string.IsNullOrEmpty(txbPlaylistNameGenerate.Text.Trim()) && txbPlaylistNameGenerate.Text.
      Length <= MAX_NAME_LENGTH)
00866             {
00867                 TimeSpan duration = new TimeSpan(0, (int)nudDurationGenerate.Value, 0);
00868                 if (!this.Controller.GeneratePlaylist(txbPlaylistNameGenerate.
      Text, duration,
00869                     (cmbTypePlaylistGenerate.SelectedItem.ToString() == AudioType.Music.ToString()) ?
      AudioType.Music : AudioType.Ad,
00870                     cmbGenderGenerate.SelectedItem.ToString(), this.IdWebradio, this.
```

```
      NameWebradio))
00871                 {
00872                     MessageBox.Show("Impossible to generate a playlist with these parameters. Some issues
       possible :\n - A playlist with this name and this type already exists\n - The given duration is too short for
       this gender", "Error");
00873                 }
00874             }
00875             else
00876                 MessageBox.Show("Please enter a valid playlist's name. (1-" + MAX_NAME_LENGTH + "
       characters)", "Error");
00877         }
00878
00891         private void ckbCheckedChanged(object sender, EventArgs e)
00892         {
00893             if ((sender as CheckBox).Name == "ckbAll")
00894             {
00895                 ckbMonday.Checked = ckbAll.Checked;
00896                 ckbTuesday.Checked = ckbAll.Checked;
00897                 ckbWednesday.Checked = ckbAll.Checked;
00898                 ckbThursday.Checked = ckbAll.Checked;
00899                 ckbFriday.Checked = ckbAll.Checked;
00900                 ckbSaturday.Checked = ckbAll.Checked;
00901                 ckbSunday.Checked = ckbAll.Checked;
00902             }
00903
00904         }
00905
00918         private void btnCreateEvent_Click(object sender, EventArgs e)
00919         {
00920             if (!string.IsNullOrEmpty(txbEventName.Text.Trim()) && txbEventName.Text.Length <=
      MAX_NAME_LENGTH)
00921             {
00922                 TimeSpan start = new TimeSpan();
00923                 if (!TimeSpan.TryParse(mtbStartTime.Text, out start))
00924                 {
00925                     MessageBox.Show("Start time format is not correct.", "Error");
00926                     return;
00927                 }
00928                 TimeSpan duration = new TimeSpan();
00929                 if (!TimeSpan.TryParse(mtbDuration.Text, out duration))
00930                 {
00931                     MessageBox.Show("Duration time format is not correct.", "Error");
00932                     return;
00933                 }
00934                 TimeSpan minimumDuration = new TimeSpan(0, 1, 0);
00935                 if (duration >= minimumDuration)
00936                 {
00937                     int repeat = this.GetRepeatValue();
00938                     if (repeat > 0)
00939                     {
00940                         CalendarEvent newEvent = new CalendarEvent(txbEventName.Text, start, duration,
      repeat, (int)nudPriority.Value, ckbShuffle.Checked, true, (Playlist)cmbPlaylistEvent.SelectedItem);
00941                         if (!this.Controller.CreateEvent(newEvent, this.IdWebradio))
00942                             MessageBox.Show("Event already exists (with this name)", "Error");
00943                     }
00944                     else
00945                         MessageBox.Show("Please select a day", "Error");
00946
00947                 }
00948                 else
00949                     MessageBox.Show("Duration must be longer or equal to " + minimumDuration.ToString());
00950             }
00951             else
00952                 MessageBox.Show("Please enter a valid event's name. (1-" + MAX_NAME_LENGTH + " characters)"
      , "Error");
00953         }
00954
00967         private void dvwTimetable_MouseUp(object sender, MouseEventArgs e)
00968         {
00969             if (dvwTimetable.Selection == SelectionType.Appointment)
00970             {
00971                 EventAppointment app = (EventAppointment)dvwTimetable.SelectedAppointment;
00972                 if (this.CheckMovePossible(app))
00973                 {
00974                     CalendarEvent aEvent = app.EventObject;
00975                     TimeSpan start = new TimeSpan(app.StartDate.Hour, app.StartDate.Minute, app.StartDate.
      Second);
00976                     TimeSpan end = new TimeSpan(app.EndDate.Hour, app.EndDate.Minute, app.EndDate.Second);
00977                     aEvent.StartTime = start;
00978                     aEvent.Duration = end - start;
00979
00980                     aEvent.Repeat = this.GetRepeatValueFromAppointement(this.GetAllRelatedAppointment(app))
      ;
00981                     if (!this.Controller.UpdateEvent(aEvent, this.IdWebradio))
00982                         MessageBox.Show("An error occured", "Error");
00983                 }
00984                 else
```

```
00985                 {
00986                     MessageBox.Show("The same event can't be in the same day more than once.", "Error");
00987                     this.UpdateView();
00988                 }
00989             }
00990         }
00991 
01003         private int GetRepeatValue()
01004         {
01005             int repeat = 0;
01006             repeat += (ckbMonday.Checked) ? (int)DayValue.Monday : 0;
01007             repeat += (ckbTuesday.Checked) ? (int)DayValue.Tuesday : 0;
01008             repeat += (ckbWednesday.Checked) ? (int)DayValue.Wednesday : 0;
01009             repeat += (ckbThursday.Checked) ? (int)DayValue.Thursday : 0;
01010             repeat += (ckbFriday.Checked) ? (int)DayValue.Friday : 0;
01011             repeat += (ckbSaturday.Checked) ? (int)DayValue.Saturday : 0;
01012             repeat += (ckbSunday.Checked) ? (int)DayValue.Sunday : 0;
01013             return repeat;
01014         }
01015 
01029         private List<EventAppointment> GetAllRelatedAppointment(EventAppointment app)
01030         {
01031             List<EventAppointment> eventList = new List<EventAppointment>();
01032             foreach (EventAppointment tmp in this.EventsCalendar)
01033             {
01034                 if (tmp.EventObject.Id == app.EventObject.Id)
01035                     eventList.Add(tmp);
01036             }
01037 
01038             return eventList;
01039         }
01040 
01054         private int GetRepeatValueFromAppointement(List<EventAppointment> eventList)
01055         {
01056             int repeat = 0;
01057             foreach (EventAppointment ev in eventList)
01058             {
01059                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Monday) ? (int)DayValue.Monday : 0;
01060                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Tuesday) ? (int)DayValue.Tuesday : 0;
01061                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Wednesday) ? (int)DayValue.Wednesday : 0;
01062                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Thursday) ? (int)DayValue.Thursday : 0;
01063                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Friday) ? (int)DayValue.Friday : 0;
01064                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Saturday) ? (int)DayValue.Saturday : 0;
01065                 repeat += (ev.StartDate.DayOfWeek == DayOfWeek.Sunday) ? (int)DayValue.Sunday : 0;
01066             }
01067             return repeat;
01068         }
01069 
01083         private bool CheckMovePossible(EventAppointment app)
01084         {
01085             bool result = true;
01086             foreach (EventAppointment ev in this.EventsCalendar)
01087             {
01088                 if (ev.EventObject.Id == app.EventObject.Id && ev != app)
01089                 {
01090                     if (ev.StartDate.DayOfWeek == app.StartDate.DayOfWeek)
01091                     {
01092                         result = false;
01093                         break;
01094                     }
01095                 }
01096             }
01097             return result;
01098         }
01099 
01112         private void dvwTimetable_MouseClick(object sender, MouseEventArgs e)
01113         {
01114             if (e.Button == System.Windows.Forms.MouseButtons.Right)
01115             {
01116                 if (dvwTimetable.Selection == SelectionType.Appointment)
01117                 {
01118                     if (MessageBox.Show("Do you really want to delete this event ?", "Delete event",
      MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
01119                     {
01120                         if (!this.Controller.DeleteEvent((dvwTimetable.
      SelectedAppointment as EventAppointment).EventObject, this.IdWebradio))
01121                             MessageBox.Show("An error occured", "Error");
01122                     }
01123                 }
01124             }
01125         }
01126 
01139         private void btnCreateTranscoder_Click(object sender, EventArgs e)
01140         {
01141             if (!string.IsNullOrEmpty(txbStreamName.Text.Trim()) && !string.IsNullOrEmpty(txbServerPassword
      .Text.Trim()) && txbStreamName.Text.Length <= MAX_NAME_LENGTH)
01142             {
```

```
01143                 IPAddress ip;
01144                 if (IPAddress.TryParse(txbServerIP.Text, out ip))
01145                 {
01146                     if (cmbEncoder.SelectedItem.ToString() == StreamType.MP3.ToString())
01147                         MessageBox.Show("You have selected MP3. MP3 need a licence to broadcast. Currently
       the transcoder will not work. WIP");
01148                     bool result = this.Controller.CreateTranscoder(txbStreamName.Text,
01149                         (cmbEncoder.SelectedItem.ToString() == StreamType.MP3.ToString()) ?
      StreamType.MP3 : StreamType.AACPlus,
01150                         int.Parse(cmbSampleRate.SelectedItem.ToString()), int.Parse(cmbBitrate.SelectedItem
      .ToString()), txbStreamUrl.Text, ip, (int)nudPort.Value, (int)nudAdminPort.Value, txbServerPassword.Text,
      this.IdWebradio);
01151                     if (!result)
01152                         MessageBox.Show("Transcoder already exist or name is invalid", "Error");
01153                 }
01154                 else
01155                     MessageBox.Show("IP Address is invalid", "Error");
01156             }
01157             else
01158                 MessageBox.Show("Please enter a valid stream's name and a server's password.(1-" +
      MAX_NAME_LENGTH + " characters)", "Error");
01159         }
01160 
01173         private void btnDeleteTranscoder_Click(object sender, EventArgs e)
01174         {
01175             if (lsbTranscoders.SelectedIndex >= 0)
01176             {
01177                 if (!this.Controller.DeleteTranscoder((WebradioTranscoder)
      lsbTranscoders.SelectedItem, this.IdWebradio))
01178                     MessageBox.Show("An error occured.", "Error");
01179             }
01180             else
01181                 MessageBox.Show("Please select a transcoder to delete", "Error");
01182         }
01183 
01196         private void lsbTranscoders_SelectedIndexChanged(object sender, EventArgs e)
01197         {
01198             this.ShowTranscoderInfos((WebradioTranscoder)lsbTranscoders.SelectedItem);
01199         }
01200 
01212         private void ShowTranscoderInfos(WebradioTranscoder transcoder)
01213         {
01214             if (transcoder != null)
01215             {
01216                 txbStreamNameEdit.Text = transcoder.Name;
01217                 txbStreamUrlEdit.Text = transcoder.Url;
01218                 txbServerIPEdit.Text = transcoder.Ip.ToString();
01219                 txbServerPasswordEdit.Text = transcoder.Password;
01220                 nudPortEdit.Value = transcoder.Port;
01221                 nudAdminPortEdit.Value = transcoder.AdminPort;
01222                 cmbSampleRateEdit.SelectedItem = transcoder.SampleRate.ToString();
01223                 cmbBitrateEdit.SelectedItem = transcoder.Birate;
01224                 cmbEncoderEdit.SelectedItem = (transcoder.StreamType == StreamType.MP3) ? "MP3" : "AAC+";
01225 
01226                 bool running = transcoder.IsRunning();
01227                 lblStatusTranscoder.Text = (running) ? "On" : "Off";
01228                 lblStatusTranscoder.ForeColor = (running) ? Color.Green : Color.Red;
01229                 btnStartTranscoder.Enabled = !running;
01230                 btnStopTranscoder.Enabled = running;
01231                 btnNextTrack.Enabled = running;
01232 
01233                 btnStartCapture.Enabled = !transcoder.Capture;
01234                 btnStopCapture.Enabled = transcoder.Capture;
01235 
01236             }
01237         }
01238 
01251         private void btnUpdate_Click(object sender, EventArgs e)
01252         {
01253             if (lsbTranscoders.SelectedIndex >= 0)
01254             {
01255                 if (MessageBox.Show("Transcoder will restart if it's running after update. Do you really
       want to update ?", "Update", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
      System.Windows.Forms.DialogResult.Yes)
01256                 {
01257                     if (!string.IsNullOrEmpty(txbStreamNameEdit.Text.Trim()) && !string.IsNullOrEmpty(
      txbServerPasswordEdit.Text.Trim()))
01258                     {
01259                         IPAddress ip;
01260                         if (IPAddress.TryParse(txbServerIPEdit.Text, out ip))
01261                         {
01262                             WebradioTranscoder transcoder = (WebradioTranscoder)lsbTranscoders.SelectedItem
      ;
01263                             transcoder.Name = txbStreamNameEdit.Text;
01264                             transcoder.Url = txbStreamUrlEdit.Text;
01265                             transcoder.Password = txbServerPasswordEdit.Text;
01266                             transcoder.Ip = ip;
```

```
01267                           transcoder.Port = (int)nudPortEdit.Value;
01268                           transcoder.AdminPort = (int)nudAdminPortEdit.Value;
01269                           transcoder.Birate = int.Parse(cmbBitrateEdit.SelectedItem.ToString());
01270                           transcoder.SampleRate = int.Parse(cmbSampleRateEdit.SelectedItem.ToString());
01271                           transcoder.StreamType = (cmbEncoderEdit.SelectedItem.ToString() ==
    StreamType.MP3.ToString()) ? StreamType.MP3 : StreamType.AACPlus;
01272                           if (!this.Controller.UpdateTranscoder(transcoder,
    ckbTranscoderDebug.Checked, this.IdWebradio))
01273                               MessageBox.Show("An error has occured", "Error");
01274                           else
01275                           {
01276                               this.ShowTranscoderInfos(transcoder);
01277                               MessageBox.Show("Update successful", "Succes");
01278                           }
01279                       }
01280                       else
01281                           MessageBox.Show("IP address is invalid", "Error");
01282                   }
01283                   else
01284                       MessageBox.Show("Please enter a name and a password", "Error");
01285               }
01286           }
01287           else
01288               MessageBox.Show("Please select a transcoder", "Error");
01289       }
01290
01303       private void btnStartTranscoder_Click(object sender, EventArgs e)
01304       {
01305           if (lsbTranscoders.SelectedIndex >= 0)
01306           {
01307               WebradioTranscoder transcoder = (WebradioTranscoder)lsbTranscoders.SelectedItem;
01308               if (this.Controller.StartTranscoder(transcoder, ckbTranscoderDebug
    .Checked, this.IdWebradio))
01309               {
01310                   this.ShowTranscoderInfos(transcoder);
01311
01312               }
01313               else
01314                   MessageBox.Show("An error has occured. Please terminate sc_trans process.", "Error");
01315           }
01316           else
01317               MessageBox.Show("Please select a transcoder", "Error");
01318       }
01319
01332       private void btnStopTranscoder_Click(object sender, EventArgs e)
01333       {
01334           if (lsbTranscoders.SelectedIndex >= 0)
01335           {
01336               WebradioTranscoder transcoder = (WebradioTranscoder)lsbTranscoders.SelectedItem;
01337               if (this.Controller.StopTranscoder(transcoder, this.IdWebradio))
01338               {
01339                   this.ShowTranscoderInfos(transcoder);
01340
01341               }
01342               else
01343                   MessageBox.Show("An error has occured.\n- Please terminate sc_trans process.\n- Please
     check that sc_trans.exe is in the shoutcast folder.", "Error");
01344           }
01345           else
01346               MessageBox.Show("Please select a transcoder", "Error");
01347
01348       }
01349
01362       private void btnShowTranscoderLog_Click(object sender, EventArgs e)
01363       {
01364           if (lsbTranscoders.SelectedIndex >= 0)
01365           {
01366               try
01367               {
01368                   Process.Start("notepad", Directory.GetCurrentDirectory() + "\\" + ((WebradioTranscoder)
    lsbTranscoders.SelectedItem).LogFilename.Replace("/", "\\"));
01369               }
01370               catch
01371               {
01372                   MessageBox.Show("Impossible to access logfile", "Error");
01373               }
01374           }
01375       }
01376
01389       private void generateAllConfigsToolStripMenuItem_Click(object sender, EventArgs e)
01390       {
01391           this.Controller.GenerateAllConfigs(this.IdWebradio);
01392       }
01393
01406       private void btnSaveServer_Click(object sender, EventArgs e)
01407       {
01408           if (MessageBox.Show("Server will restart if it's running after update. Do you really want to
```

```
       update ?", "Update", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
      System.Windows.Forms.DialogResult.Yes)
01409             {
01410                 if (!string.IsNullOrEmpty(txbLocalServerPassword.Text.Trim()) && !string.IsNullOrEmpty(
      txbLocalServerAdminPassword.Text.Trim()))
01411                 {
01412                     if (txbLocalServerPassword.Text != txbLocalServerAdminPassword.Text)
01413                     {
01414                         if (this.Controller.UpdateServer(ckbServerDebug.Checked, (int
      )nudPortServer.Value, txbLocalServerPassword.Text, txbLocalServerAdminPassword.Text, (int)nudMaxListener.
      Value, this.IdWebradio))
01415                             MessageBox.Show("Server informations updated.", "Success");
01416                         else
01417                             MessageBox.Show("An error has occured", "Error");
01418                     }
01419                     else
01420                         MessageBox.Show("Password must be different", "Error");
01421                 }
01422                 else
01423                     MessageBox.Show("Please enter a password and an admin password.", "Error");
01424             }
01425         }
01426
01439         private void btnStartServer_Click(object sender, EventArgs e)
01440         {
01441             if (!this.Controller.StartServer(this.IdWebradio, ckbServerDebug.Checked))
01442                 MessageBox.Show("An error has occured.\n- Please terminate sc_server process.\n- Please
       check that sc_serv.exe is in the shoutcast folder.", "Error");
01443         }
01444
01457         private void btnStopServer_Click(object sender, EventArgs e)
01458         {
01459             if (!this.Controller.StopServer(this.IdWebradio))
01460                 MessageBox.Show("An error has occured. Please terminate sc_server process.", "Error");
01461         }
01462
01475         private void btnShowWebInterface_Click(object sender, EventArgs e)
01476         {
01477             this.Controller.ShowServerWebInterface(this.IdWebradio);
01478         }
01479
01492         private void btnShowWebAdministration_Click(object sender, EventArgs e)
01493         {
01494             this.Controller.ShowServerWebAdmin(this.IdWebradio);
01495         }
01496
01509         private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
01510         {
01511             MessageBox.Show("WebradioManager V0.1\nSimon Menetrey\nTravail de diplôme 2014", "About",
      MessageBoxButtons.OK, MessageBoxIcon.Information);
01512         }
01513
01526         private void btnShowServerLog_Click(object sender, EventArgs e)
01527         {
01528             try
01529             {
01530                 Process.Start("notepad", Directory.GetCurrentDirectory() + "\\" + this.
      Controller.GetWebradio(this.IdWebradio).Server.LogFilename.Replace("/", "\\"));
01531             }
01532             catch
01533             {
01534                 MessageBox.Show("Impossible to access logfile", "Error");
01535             }
01536         }
01537
01550         private void btnResolve_Click(object sender, EventArgs e)
01551         {
01552             bool editSection = false;
01553             if ((sender as Button).Tag != null)
01554                 editSection = true;
01555             IPAddress ip;
01556             if (!GetResolvedConnecionIPAddress((editSection) ? txbServerIPEdit.Text : txbServerIP.Text, out
       ip))
01557                 MessageBox.Show("Impossible to resolve this dns name", "Error");
01558             else
01559             {
01560                 if (editSection)
01561                     txbServerIPEdit.Text = ip.ToString();
01562                 else
01563                     txbServerIP.Text = ip.ToString();
01564             }
01565
01566         }
01567
01583         private static bool GetResolvedConnecionIPAddress(string serverNameOrURL,
01584                    out IPAddress resolvedIPAddress)
01585         {
```

```
01586             bool isResolved = false;
01587             IPHostEntry hostEntry = null;
01588             IPAddress resolvIP = null;
01589             try
01590             {
01591                 if (!IPAddress.TryParse(serverNameOrURL, out resolvIP))
01592                 {
01593                     hostEntry = Dns.GetHostEntry(serverNameOrURL);
01594 
01595                     if (hostEntry != null && hostEntry.AddressList != null
01596                                 && hostEntry.AddressList.Length > 0)
01597                     {
01598                         if (hostEntry.AddressList.Length == 1)
01599                         {
01600                             resolvIP = hostEntry.AddressList[0];
01601                             isResolved = true;
01602                         }
01603                         else
01604                         {
01605                             foreach (IPAddress var in hostEntry.AddressList)
01606                             {
01607                                 if (var.AddressFamily == AddressFamily.InterNetwork)
01608                                 {
01609                                     resolvIP = var;
01610                                     isResolved = true;
01611                                     break;
01612                                 }
01613                             }
01614                         }
01615                     }
01616                 }
01617                 else
01618                 {
01619                     isResolved = true;
01620                 }
01621             }
01622             catch
01623             {
01624                 isResolved = false;
01625                 resolvIP = null;
01626             }
01627             finally
01628             {
01629                 resolvedIPAddress = resolvIP;
01630             }
01631 
01632             return isResolved;
01633         }
01634 
01647         private void btnNextTrack_Click(object sender, EventArgs e)
01648         {
01649             if (lsbTranscoders.SelectedIndex >= 0)
01650             {
01651                 if (!this.Controller.TranscoderNextTrack((WebradioTranscoder)
    lsbTranscoders.SelectedItem))
01652                     MessageBox.Show("An error occured.", "Error");
01653             }
01654             else
01655                 MessageBox.Show("Please select a running transcoder.", "Error");
01656         }
01657 
01670         private void btnClearHistory_Click(object sender, EventArgs e)
01671         {
01672             if (lsbTranscoders.SelectedIndex >= 0)
01673             {
01674                 if (this.Controller.ClearHistory(((WebradioTranscoder)lsbTranscoders.
    SelectedItem).Id))
01675                     MessageBox.Show("History cleared", "Success");
01676                 else
01677                     MessageBox.Show("An error has occured", "Error");
01678             }
01679             else
01680                 MessageBox.Show("Please select a transcoder.", "Error");
01681         }
01682 
01695         private void btnGenerateHistory_Click(object sender, EventArgs e)
01696         {
01697             if (lsbTranscoders.SelectedIndex >= 0)
01698             {
01699                 SaveFileDialog SFD = new SaveFileDialog();
01700                 SFD.Filter = ".pdf|PDF";
01701                 SFD.FileName = "history.pdf";
01702                 if (SFD.ShowDialog() == System.Windows.Forms.DialogResult.OK)
01703                 {
01704                     if (this.Controller.GenerateHistory(this.IdWebradio, ((
    WebradioTranscoder)lsbTranscoders.SelectedItem).Name, ((WebradioTranscoder)lsbTranscoders.SelectedItem).Id, SFD.
    FileName))
```

```
01705                     {
01706                         if (MessageBox.Show("Do you want to view the pdf file ?", "Success",
    MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
01707                             Process.Start(SFD.FileName);
01708                     }
01709                     else
01710                         MessageBox.Show("An error has occured", "Error");
01711                 }
01712             }
01713             else
01714                 MessageBox.Show("Please select a transcoder.", "Error");
01715         }
01716 
01729         private void btnModifyName_Click(object sender, EventArgs e)
01730         {
01731             if (MessageBox.Show("To rename this webradio, all running process (transcoder and server) will
     be shutting down. Are you sure ?", "Process", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
    System.Windows.Forms.DialogResult.Yes)
01732             {
01733                 if (!string.IsNullOrEmpty(txbWebradioName.Name.Trim()) && txbWebradioName.Name.Length <=
    MAX_NAME_LENGTH)
01734                 {
01735                     if (this.Controller.ModifyWebradioName(txbWebradioName.Text
    , this.IdWebradio))
01736                         MessageBox.Show("Modification completed", "Success");
01737                     else
01738                         MessageBox.Show("This name is already used", "Error");
01739                 }
01740                 else
01741                     MessageBox.Show("Please enter a valid name. (1-" + MAX_NAME_LENGTH + " characters)", "
    Error");
01742             }
01743         }
01744 
01757         private void dgvCellEndEdit(object sender, DataGridViewCellEventArgs e)
01758         {
01759             DataGridView dgv = (sender as DataGridView);
01760             DataGridViewRow row = dgv.Rows[e.RowIndex];
01761             AudioFile file;
01762             if (dgv.Tag.ToString() == AudioType.Music.ToString())
01763             {
01764                 file = new Music(int.Parse(row.Cells["colId" + dgv.Tag.ToString()].Value.ToString()),
01765                     row.Cells["colPath" + dgv.Tag.ToString()].Value.ToString(),
01766                     row.Cells["colTitle" + dgv.Tag.ToString()].Value.ToString(),
01767                     row.Cells["colArtist" + dgv.Tag.ToString()].Value.ToString(),
01768                     row.Cells["colAlbum" + dgv.Tag.ToString()].Value.ToString(),
01769                     int.Parse(row.Cells["colYear" + dgv.Tag.ToString()].Value.ToString()),
01770                     row.Cells["colLabel" + dgv.Tag.ToString()].Value.ToString(),
01771                     new TimeSpan(),
01772                     row.Cells["colGender" + dgv.Tag.ToString()].Value.ToString());
01773             }
01774             else
01775             {
01776                 file = new Ad(int.Parse(row.Cells["colId" + dgv.Tag.ToString()].Value.ToString()),
01777                     row.Cells["colPath" + dgv.Tag.ToString()].Value.ToString(),
01778                 row.Cells["colTitle" + dgv.Tag.ToString()].Value.ToString(),
01779                 row.Cells["colArtist" + dgv.Tag.ToString()].Value.ToString(),
01780                 row.Cells["colAlbum" + dgv.Tag.ToString()].Value.ToString(),
01781                 int.Parse(row.Cells["colYear" + dgv.Tag.ToString()].Value.ToString()),
01782                 row.Cells["colLabel" + dgv.Tag.ToString()].Value.ToString(),
01783                 new TimeSpan(),
01784                 row.Cells["colGender" + dgv.Tag.ToString()].Value.ToString());
01785             }
01786             if (!this.Controller.UpdateAudioFile(file))
01787                 MessageBox.Show("An error has occured", "Error");
01788         }
01789 
01802         private void btnUpdateAudioDevice_Click(object sender, EventArgs e)
01803         {
01804             this.UpdateAudioDevices();
01805         }
01806 
01819         private void btnStartCapture_Click(object sender, EventArgs e)
01820         {
01821             if (lsbTranscoders.SelectedIndex >= 0)
01822             {
01823                 this.Controller.TranscoderCapture(true, cmbAudioDevice.SelectedItem.ToString(), (
    WebradioTranscoder)lsbTranscoders.SelectedItem, this.IdWebradio);
01824             }
01825             else
01826                 MessageBox.Show("Please select a transcoder.", "Error");
01827         }
01828 
01841         private void btnStopCapture_Click(object sender, EventArgs e)
01842         {
01843             if (lsbTranscoders.SelectedIndex >= 0)
01844             {
```

```
01845                 this.Controller.TranscoderCapture(false, cmbAudioDevice.SelectedItem.ToString(), (
      WebradioTranscoder)lsbTranscoders.SelectedItem, this.IdWebradio);
01846             }
01847             else
01848                 MessageBox.Show("Please select a transcoder.", "Error");
01849         }
01850 
01863         private void btnUpdateListeners_Click(object sender, EventArgs e)
01864         {
01865             List<WebradioListener> listeners = this.Controller.GetServerListeners(this.IdWebradio);
01866             dgvServerListeners.Rows.Clear();
01867             foreach (WebradioListener listener in listeners)
01868             {
01869                 string[] infos = new string[] { listener.Hostname, listener.Useragent,
      listener.ConnectionTime.ToString(), listener.Uid.ToString() };
01870                 dgvServerListeners.Rows.Add(infos);
01871             }
01872             this.Controller.UpdateServerStats(this.IdWebradio);
01873         }
01874 
01887         private void checkLibraryToolStripMenuItem_Click(object sender, EventArgs e)
01888         {
01889             if (this.Controller.CheckLibrary())
01890                 MessageBox.Show("Library has been checked and invalids files has been deleted.", "Success")
      ;
01891             else
01892                 MessageBox.Show("An error occured", "Error");
01893         }
01894 
01895         #endregion
01896 
01897     }
01898 }
```

7.7 AudioFile.cs File Reference

Implements the audio file class.

Classes

- class WebradioManager.AudioFile

  An audio file. Abstract class.

Namespaces

- package WebradioManager

7.7.1 Detailed Description

Implements the audio file class.

Definition in file AudioFile.cs.

7.8 AudioFile.cs

```
00001 
00007 using System;
00008 
00009 namespace WebradioManager
00010 {
00020     public abstract class AudioFile
00021     {
00022         #region Const
00023         // \brief Number of elements (for an audiofile).
00024         const int NUMBER_OF_ELEMENTS = 9;
00025         // \brief The default identifier.
00026         const int DEFAULT_ID = 0;
00027         #endregion
```

```
00028
00029         #region Fields
00030         // \brief The identifier.
00031         private int _id;
00032         // \brief Filename of the audiofile.
00033         private string _filename;
00034         // \brief The title.
00035         private string _title;
00036         // \brief The artist.
00037         private string _artist;
00038         // \brief The album.
00039         private string _album;
00040         // \brief The year.
00041         private int _year;
00042         // \brief The label.
00043         private string _label;
00044         // \brief The duration.
00045         private TimeSpan _duration;
00046         // \brief The gender.
00047         private string _gender;
00048         // \brief The type.
00049         private AudioType _type;
00050         #endregion
00051
00052         #region Properties
00053
00062         public int Id
00063         {
00064             get { return _id; }
00065             set { _id = value; }
00066         }
00067
00076         public string Filename
00077         {
00078             get { return _filename; }
00079             set { _filename = value; }
00080         }
00081
00090         public string Title
00091         {
00092             get { return _title; }
00093             set { _title = value; }
00094         }
00095
00104         public string Artist
00105         {
00106             get { return _artist; }
00107             set { _artist = value; }
00108         }
00109
00118         public string Album
00119         {
00120             get { return _album; }
00121             set { _album = value; }
00122         }
00123
00132         public int Year
00133         {
00134             get { return _year; }
00135             set { _year = value; }
00136         }
00137
00146         public string Label
00147         {
00148             get { return _label; }
00149             set { _label = value; }
00150         }
00151
00160         public TimeSpan Duration
00161         {
00162             get { return _duration; }
00163             set { _duration = value; }
00164         }
00165
00174         public string Gender
00175         {
00176             get { return _gender; }
00177             set { _gender = value; }
00178         }
00179
00188         public AudioType Type
00189         {
00190             get { return _type; }
00191             set { _type = value; }
00192         }
00193
00194         #endregion
```

```
00195
00196         #region Methods
00197
00218         public AudioFile(int id, string filename, string title, string artist, string album, int
      year, string label, TimeSpan duration, string gender, AudioType audiotype)
00219         {
00220             this.Id = id;
00221             this.Filename = filename;
00222             this.Title = title;
00223             this.Artist = artist;
00224             this.Album = album;
00225             this.Year = year;
00226             this.Label = label;
00227             this.Duration = duration;
00228             this.Gender = gender;
00229             this.Type = audiotype;
00230         }
00231
00251         public AudioFile(string filename, string title, string artist, string album, int year,
      string label, TimeSpan duration, string gender, AudioType audiotype)
00252             :this(DEFAULT_ID,filename, title, artist, album, year, label, duration, gender, audiotype)
00253         {
00254             //NO CODE
00255         }
00256
00268         public string[] GetInfosArray()
00269         {
00270             string[] infos = new string[NUMBER_OF_ELEMENTS];
00271
00272             infos[0] = this.Id.ToString();
00273             infos[1] = this.Title;
00274             infos[2] = this.Artist;
00275             infos[3] = this.Album;
00276             infos[4] = this.Year.ToString();
00277             infos[5] = this.Label;
00278             infos[6] = this.Duration.ToString(@"\:mm\:ss");
00279             infos[7] = this.Gender;
00280             infos[8] = this.Filename;
00281
00282             return infos;
00283         }
00284         #endregion
00285     }
00286 }
```

7.9 AudioType.cs File Reference

Implements the audio type enum.

Namespaces

- package WebradioManager

Enumerations

- enum WebradioManager.AudioType { **Music** = 2, **Ad** = 1 }

  Values that represent AudioType's id. Defined in DB.

7.9.1 Detailed Description

Implements the audio type enum.

Definition in file AudioType.cs.

7.10 AudioType.cs

```
00001
00007 namespace WebradioManager
```

```
00008 {
00015     public enum AudioType
00016     {
00017         //< An enum constant representing the music option
00018         Music = 2,
00019         //< An enum constant representing the ad option
00020         Ad = 1,
00021     }
00022 }
```

7.11 Bdd.cs File Reference

Implements the bdd class.

Classes

- class WebradioManager.Bdd

  A bdd connection.

Namespaces

- package WebradioManager

7.11.1 Detailed Description

Implements the bdd class.

Definition in file Bdd.cs.

7.12 Bdd.cs

```
00001
00007 using System;
00008 using System.Collections.Generic;
00009 using System.Data.SQLite;
00010 using System.Net;
00011
00012 namespace WebradioManager
00013 {
00023     public class Bdd
00024     {
00025         #region Const
00026         // \brief The error code.
00027         public const int ERROR = -1;
00028         #endregion
00029
00030         #region Fields
00031         // \brief The bdd controls.
00032         private BddControls _controls;
00033         #endregion
00034
00035         #region Properties
00036
00045         public BddControls Controls
00046         {
00047             get { return _controls; }
00048             set { _controls = value; }
00049         }
00050         #endregion
00051
00052         #region Methods
00053
00063         public Bdd()
00064         {
00065             this.Controls = new BddControls();
00066         }
00067
00079         public Dictionary<int,Webradio> LoadWebradios()
```

```
00080         {
00081             Dictionary<int, Webradio> webradios = new Dictionary<int, Webradio>();
00082             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT * FROM twebradio");
00083             while(reader.Read())
00084             {
00085                 webradios.Add( int.Parse(reader["id"].ToString()), new Webradio(reader["name"].
      ToString(), int.Parse(reader["id"].ToString())));
00086             }
00087             reader.Close();
00088 
00089             foreach(KeyValuePair<int,Webradio> pair in webradios)
00090             {
00091                 Webradio wr = pair.Value;
00092                 //Server
00093                 int id = wr.Id;
00094                 reader = this.Controls.ExecuteDataReader("SELECT * FROM tserver WHERE webradioid = " +
      id.ToString());
00095                 reader.Read();
00096                 wr.Server = new WebradioServer(int.Parse(reader["port"].ToString()),
00097                     reader["logfilename"].ToString(),
00098                     reader["configfilename"].ToString(),
00099                     reader["password"].ToString(),
00100                     reader["adminpassword"].ToString(),
00101                     int.Parse(reader["maxlistener"].ToString()));
00102                 reader.Close();
00103                 //----
00104 
00105                 //Calendar
00106                 reader = this.Controls.ExecuteDataReader("SELECT id, filename FROM tcalendar WHERE
       webradioid = " + id.ToString());
00107                 reader.Read();
00108                 wr.Calendar = new WebradioCalendar(int.Parse(reader["id"].ToString()),
      reader["filename"].ToString());
00109                 reader.Close();
00110                 reader = this.Controls.ExecuteDataReader("SELECT ce.id AS EventId, ce.playlistid,
       ce.starttime, ce.duration, ce.name AS EventName, ce.repeat, ce.priority, ce.shuffle, ce.loopatend, p.name  AS
       PlaylistName, p.filename AS PlaylistFilename, at.name AS AudiotypeName FROM tcalendarevent ce, tplaylist p,
       taudiotype at WHERE ce.calendarid = " + wr.Calendar.Id + " AND ce.playlistid = p.id AND p.typeid = at.id");
00111                 while(reader.Read())
00112                 {
00113                     string[] time = reader["starttime"].ToString().Split(':');
00114                     TimeSpan start = new TimeSpan(int.Parse(time[0]),int.Parse(time[1]),int.Parse(time[2]))
      ;
00115                     time = reader["duration"].ToString().Split(':');
00116                     TimeSpan duration = new TimeSpan(int.Parse(time[0]),int.Parse(time[1]),int.Parse(time[2
      ]));
00117                     Playlist playlist;
00118                     if (reader["AudiotypeName"].ToString() == AudioType.Music.ToString())
00119                         playlist = new PlaylistMusic(reader["PlaylistName"].ToString(), reader
      ["PlaylistFilename"].ToString());
00120                     else
00121                         playlist = new PlaylistAd(reader["PlaylistName"].ToString(), reader["
      PlaylistFilename"].ToString());
00122                     wr.Calendar.Events.Add(new CalendarEvent(int.Parse(reader["EventId"].
      ToString()), reader["EventName"].ToString(),
00123                         start,
00124                         duration,
00125                         int.Parse(reader["repeat"].ToString()),
00126                         int.Parse(reader["priority"].ToString()),
00127                         Convert.ToBoolean(reader["shuffle"].ToString()),
00128                         Convert.ToBoolean(reader["loopatend"].ToString()),
00129                         playlist));
00130                 }
00131                 reader.Close();
00132                 //---
00133 
00134                 //Playlists
00135                 reader = this.Controls.ExecuteDataReader("SELECT p.id, p.name AS PlaylistName, p.filename,
       t.name AS AudioType FROM tplaylist p, taudiotype t WHERE p.typeid = t.id AND webradioid = " + id.ToString())
      ;
00136                 while(reader.Read())
00137                 {
00138                     Playlist p = null;
00139                     if(reader["AudioType"].ToString() == AudioType.Ad.ToString())
00140                         p = new PlaylistAd(int.Parse(reader["id"].ToString()), reader["
      PlaylistName"].ToString(), reader["filename"].ToString());
00141                     else if(reader["AudioType"].ToString() == AudioType.Music.ToString())
00142                         p = new PlaylistMusic(int.Parse(reader["id"].ToString()), reader["
      PlaylistName"].ToString(), reader["filename"].ToString());
00143                     wr.Playlists.Add(p);
00144                 }
00145                 reader.Close();
00146                 foreach(Playlist playlist in wr.Playlists)
00147                 {
00148                     reader = this.Controls.ExecuteDataReader("SELECT m.filename FROM tmusic m,
       tplaylist_has_music pm WHERE pm.playlistid = " + playlist.Id + " AND m.id = pm.musicid");
00149                     while(reader.Read())
```

```
00150                     {
00151                         playlist.AudioFileList.Add(reader["filename"].ToString());
00152                     }
00153                 }
00154                 reader.Close();
00155                 //Transcoders
00156                 reader = this.Controls.ExecuteDataReader("SELECT c.filename AS CalendarFilename, tr.id,
      tr.name AS TransName, tr.bitrate, tr.samplerate, tr.url, tr.ip, tr.port, tr.adminport, tr.password,
      tr.configfilename, tr.logfilename, st.name AS StreamName FROM tcalendar c, ttranscoder tr, tstreamtype st WHERE
      tr.webradioid = " + id.ToString() + " AND tr.streamtypeid = st.id AND c.webradioid = " + id.ToString());
00157                 while(reader.Read())
00158                 {
00159                     WebradioTranscoder trans = null;
00160                     if(reader["StreamName"].ToString() == StreamType.MP3.ToString())
00161                         trans = new TranscoderMp3(int.Parse(reader["id"].ToString()),
00162                             reader["TransName"].ToString(),
00163                             int.Parse(reader["bitrate"].ToString()),
00164                             int.Parse(reader["samplerate"].ToString()),
00165                             IPAddress.Parse(reader["ip"].ToString()),
00166                             int.Parse(reader["port"].ToString()),
00167                             int.Parse(reader["adminport"].ToString()),
00168                             reader["url"].ToString(),
00169                             reader["password"].ToString(),
00170                             reader["configfilename"].ToString(),
00171                             reader["logfilename"].ToString());
00172                     else
00173                         trans = new TranscoderAacPlus(int.Parse(reader["id"].ToString()),
00174                             reader["TransName"].ToString(),
00175                             int.Parse(reader["bitrate"].ToString()),
00176                             int.Parse(reader["samplerate"].ToString()),
00177                             IPAddress.Parse(reader["ip"].ToString()),
00178                             int.Parse(reader["port"].ToString()),
00179                             int.Parse(reader["adminport"].ToString()),
00180                             reader["url"].ToString(),
00181                             reader["password"].ToString(),
00182                             reader["configfilename"].ToString(),
00183                             reader["logfilename"].ToString());
00184                     trans.CalendarFile = reader["CalendarFilename"].ToString();
00185                     wr.Transcoders.Add(trans);
00186                 }
00187                 reader.Close();
00188                 //---
00189 
00190             }
00191 
00192             return webradios;
00193         }
00194 
00206         public List<AudioFile> LoadLibrary()
00207         {
00208             List<AudioFile> audios = new List<AudioFile>();
00209             //Music
00210             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT m.id, m.filename, m.title,
      m.artist, m.album, m.year, m.label, m.duration, t.name AS AudioType, g.name AS GenderName FROM tmusic m,
      taudiotype t, tgender g WHERE m.typeid = t.id AND m.genderid = g.id");
00211             while (reader.Read())
00212             {
00213                 string[] time = reader["duration"].ToString().Split(':');
00214                 TimeSpan duration = new TimeSpan(int.Parse(time[0]), int.Parse(time[1]), int.Parse(time[2])
      );
00215                 AudioFile af = null;
00216                 if (reader["AudioType"].ToString() == AudioType.Ad.ToString())
00217                     af = new Ad(int.Parse(reader["id"].ToString()),
00218                         reader["filename"].ToString(),
00219                         reader["title"].ToString(),
00220                         reader["artist"].ToString(),
00221                         reader["album"].ToString(),
00222                         int.Parse(reader["year"].ToString()),
00223                         reader["label"].ToString(),
00224                         duration,
00225                         reader["GenderName"].ToString());
00226                 else if (reader["AudioType"].ToString() == AudioType.Music.ToString())
00227                     af = new Music(int.Parse(reader["id"].ToString()),
00228                         reader["filename"].ToString(),
00229                         reader["title"].ToString(),
00230                         reader["artist"].ToString(),
00231                         reader["album"].ToString(),
00232                         int.Parse(reader["year"].ToString()),
00233                         reader["label"].ToString(),
00234                         duration,
00235                         reader["GenderName"].ToString());
00236 
00237                 audios.Add(af);
00238             }
00239             reader.Close();
00240 
00241             return audios;
```

```
00242         }
00243
00257         public int AddWebradio(Webradio webradio)
00258         {
00259             //Webradio name must be unique !
00260             if (this.WebradioExist(webradio.Name))
00261                 return ERROR;
00262             Dictionary<string, string> data = new Dictionary<string, string>();
00263             data.Add("name", webradio.Name);
00264             try
00265             {
00266                 //Webradio
00267                 this.Controls.Insert("twebradio", data);
00268                 SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM twebradio WHERE
      name = '" + webradio.Name + "'");
00269                 reader.Read();
00270                 int id = int.Parse(reader["id"].ToString());
00271                 reader.Close();
00272                 data.Clear();
00273                 //Server creation
00274                 data.Add("webradioid", id.ToString());
00275                 data.Add("port", webradio.Server.Port.ToString());
00276                 data.Add("logfilename", webradio.Server.LogFilename);
00277                 data.Add("configfilename", webradio.Server.ConfigFilename);
00278                 data.Add("password", webradio.Server.Password);
00279                 data.Add("adminpassword", webradio.Server.AdminPassword);
00280                 data.Add("maxlistener", webradio.Server.MaxListener.ToString());
00281                 this.Controls.Insert("tserver", data);
00282                 data.Clear();
00283                 //----
00284                 //Calendar creation
00285                 data.Add("filename", webradio.Calendar.Filename);
00286                 data.Add("webradioid", id.ToString());
00287                 this.Controls.Insert("tcalendar", data);
00288                 data.Clear();
00289                 reader = this.Controls.ExecuteDataReader("SELECT id FROM tcalendar WHERE webradioid = " +
     id.ToString());
00290                 reader.Read();
00291                 webradio.Calendar.Id = int.Parse(reader["id"].ToString());
00292                 reader.Close();
00293                 //----
00294                 return id;
00295             }
00296             catch
00297             {
00298                 return ERROR;
00299             }
00300         }
00301
00315         public bool WebradioExist(string name)
00316         {
00317             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT COUNT(*) AS Count FROM
      twebradio WHERE name = '"+ name +"'");
00318             reader.Read();
00319             if (int.Parse(reader["Count"].ToString()) == 0)
00320             {
00321                 reader.Close();
00322                 return false;
00323             }
00324             else
00325             {
00326                 reader.Close();
00327                 return true;
00328             }
00329
00330
00331         }
00332
00346         public bool DeleteWebradio(int id)
00347         {
00348             try
00349             {
00350                 this.Controls.Delete("twebradio", "id = " + id.ToString());
00351                 return true;
00352             }
00353             catch
00354             {
00355                 return false;
00356             }
00357         }
00358
00370         public List<string> GetGenders()
00371         {
00372             List<string> genders = new List<string>();
00373             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT * FROM tgender");
00374             while(reader.Read())
00375             {
```

```
00376                genders.Add(reader["name"].ToString());
00377            }
00378            reader.Close();
00379            return genders;
00380        }
00381
00395        public int GetGenderId(string gender)
00396        {
00397            SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM tgender WHERE name =
       '"+gender+"'");
00398            reader.Read();
00399            int id;
00400            if (reader.HasRows)
00401                id = int.Parse(reader["id"].ToString());
00402            else
00403                id = ERROR;
00404            reader.Close();
00405            return id;
00406        }
00407
00421        public int AddGender(string gender)
00422        {
00423            Dictionary<string, string> data = new Dictionary<string, string>();
00424            data.Add("name", gender);
00425            this.Controls.Insert("tgender", data);
00426
00427            SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM tgender WHERE name =
       '"+ gender +"'");
00428            reader.Read();
00429            int id = int.Parse(reader["id"].ToString());
00430            reader.Close();
00431            return id;
00432        }
00433
00447        public int AddAudioFile(AudioFile file)
00448        {
00449            if(this.AudioFileExist(file.Filename))
00450                return ERROR;
00451
00452            int genderId = this.GetGenderId(file.Gender);
00453            //If return error, gender doesn't exist in DB, so add it
00454            if (genderId == ERROR)
00455                //Get the new id
00456                genderId = AddGender(file.Gender);
00457
00458            Dictionary<string, string> data = new Dictionary<string, string>();
00459            data.Add("filename", file.Filename.Replace('\'', ' '));
00460            data.Add("title", file.Title);
00461            data.Add("artist", file.Artist);
00462            data.Add("album", file.Album);
00463            data.Add("year", file.Year.ToString());
00464            data.Add("label", file.Label);
00465            data.Add("duration", file.Duration.ToString(@"hh\:mm\:ss"));
00466            data.Add("genderid", genderId.ToString());
00467            data.Add("typeid", ((int)file.Type).ToString());
00468            this.Controls.Insert("tmusic", data);
00469
00470            //Get the new id
00471            SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM tmusic WHERE filename
       = '" + file.Filename.Replace('\'', ' ') + "'");
00472            reader.Read();
00473            int id = int.Parse(reader["id"].ToString());
00474            reader.Close();
00475            return id;
00476        }
00477
00491        public bool UpdateAudioFile(AudioFile file)
00492        {
00493            try
00494            {
00495                int genderId = this.GetGenderId(file.Gender);
00496                //If return error, gender doesn't exist in DB, so add it
00497                if (genderId == ERROR)
00498                    //Get the new id
00499                    genderId = AddGender(file.Gender);
00500                Dictionary<string, string> data = new Dictionary<string, string>();
00501                data.Add("title", file.Title);
00502                data.Add("artist", file.Artist);
00503                data.Add("album", file.Album);
00504                data.Add("year", file.Year.ToString());
00505                data.Add("label", file.Label);
00506                data.Add("genderid", genderId.ToString());
00507                data.Add("typeid", ((int)file.Type).ToString());
00508                this.Controls.Update("tmusic", data, "id = " + file.Id);
00509                return true;
00510            }
00511            catch
```

```
00512             {
00513                 return false;
00514             }
00515         }
00516 
00530         public bool AudioFileExist(string filename)
00531         {
00532             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT COUNT(*) AS Count FROM tmusic
      WHERE filename = '" + filename.Replace('\'', ' ') + "'");
00533             reader.Read();
00534             if (reader["Count"].ToString() == "0")
00535             {
00536                 reader.Close();
00537                 return false;
00538             }
00539             else
00540             {
00541                 reader.Close();
00542                 return true;
00543             }
00544 
00545         }
00546 
00560         public bool DeleteAudioFile(int id)
00561         {
00562             try
00563             {
00564                 this.Controls.Delete("tmusic", "id = " + id.ToString());
00565                 //this.Controls.Delete("tplaylist_has_music", "idmusic = " + id.ToString());
00566                 return true;
00567             }
00568             catch
00569             {
00570                 return false;
00571             }
00572         }
00573 
00588         public int CreatePlaylist(Playlist playlist, int webradioId)
00589         {
00590             if(this.PlaylistExist(playlist,webradioId))
00591                 return ERROR;
00592 
00593             Dictionary<string,string> data = new Dictionary<string,string>();
00594             data.Add("name", playlist.Name);
00595             data.Add("filename", playlist.Filename);
00596             data.Add("webradioid", webradioId.ToString());
00597             data.Add("typeid", ((int)playlist.Type).ToString());
00598             this.Controls.Insert("tplaylist", data);
00599 
00600             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM tplaylist WHERE name
      = '" + playlist.Name + "' AND webradioid = " + webradioId.ToString() + " AND typeid = " + ((int)
     playlist.Type).ToString());
00601             reader.Read();
00602             int id = int.Parse(reader["id"].ToString());
00603             reader.Close();
00604             return id;
00605         }
00606 
00620         public bool DeletePlaylist(int id)
00621         {
00622             try
00623             {
00624                 this.Controls.Delete("tplaylist", "id = " + id.ToString());
00625                 return true;
00626             }
00627             catch
00628             {
00629                 return false;
00630             }
00631         }
00632 
00647         public bool AddToPlaylist(int idAudioFile, int idPlaylist)
00648         {
00649             try
00650             {
00651                 Dictionary<string,string> data = new Dictionary<string,string>();
00652                 data.Add("playlistid", idPlaylist.ToString());
00653                 data.Add("musicid", idAudioFile.ToString());
00654                 this.Controls.Insert("tplaylist_has_music", data);
00655                 return true;
00656             }
00657             catch
00658             {
00659                 return false;
00660             }
00661         }
00662 
```

```
00677         public bool RemoveFromPlaylist(int idAudioFile, int idPlaylist)
00678         {
00679             try
00680             {
00681                 this.Controls.Delete("tplaylist_has_music", "musicid = " + idAudioFile.ToString() + " AND
      playlistid = " + idPlaylist.ToString());
00682                 return true;
00683             }
00684             catch
00685             {
00686                 return false;
00687             }
00688         }
00689 
00704         private bool PlaylistExist(Playlist playlist, int webradioId)
00705         {
00706             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT COUNT(*) AS Count FROM
      tplaylist WHERE name = '"+playlist.Name+"' AND webradioid = "+webradioId.ToString()+" AND typeid = " + ((int)
     playlist.Type).ToString());
00707             reader.Read();
00708             bool result = Convert.ToBoolean(int.Parse(reader["Count"].ToString()));
00709             reader.Close();
00710             return result;
00711         }
00712 
00728         public int AddGeneratedPlaylist(Playlist playlist, List<int>
     audioFilesId, int webradioId)
00729         {
00730             int idPlaylist = this.CreatePlaylist(playlist, webradioId);
00731             if (idPlaylist == ERROR)
00732                 return idPlaylist;
00733             foreach(int audioFileId in audioFilesId)
00734                 this.AddToPlaylist(audioFileId, idPlaylist);
00735             return idPlaylist;
00736 
00737         }
00738 
00754         public int AddEvent(CalendarEvent newEvent, int calendarId, int playlistId)
00755         {
00756             Dictionary<string, string> data = new Dictionary<string, string>();
00757             data.Add("name", newEvent.Name);
00758             data.Add("starttime", newEvent.StartTime.ToString());
00759             data.Add("duration", newEvent.Duration.ToString());
00760             data.Add("repeat", newEvent.Repeat.ToString());
00761             data.Add("priority", newEvent.Priority.ToString());
00762             data.Add("shuffle", (newEvent.Shuffle) ? "TRUE" : "FALSE");
00763             data.Add("loopatend", "TRUE");
00764             data.Add("calendarid", calendarId.ToString());
00765             data.Add("playlistid", playlistId.ToString());
00766 
00767             this.Controls.Insert("tcalendarevent", data);
00768 
00769             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM tcalendarevent WHERE
      name = '"+ newEvent.Name +"' AND calendarid = " + calendarId.ToString());
00770             reader.Read();
00771             int id = int.Parse(reader["id"].ToString());
00772             reader.Close();
00773             return id;
00774         }
00775 
00790         public bool EventExist(CalendarEvent aEvent, int calendarId)
00791         {
00792             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT COUNT(*) AS Count FROM
      tcalendarevent WHERE name = '" + aEvent.Name + "' AND calendarid = " + calendarId.ToString());
00793             reader.Read();
00794             bool result = (reader["Count"].ToString() == "0")?false:true;
00795             reader.Close();
00796             return result;
00797         }
00798 
00812         public bool UpdateEvent(CalendarEvent aEvent)
00813         {
00814             Dictionary<string,string> data = new Dictionary<string,string>();
00815             //Only change starttime and duration for the moment
00816             data.Add("starttime", aEvent.StartTime.ToString());
00817             data.Add("duration", aEvent.Duration.ToString());
00818             data.Add("repeat", aEvent.Repeat.ToString());
00819             try
00820             {
00821                 this.Controls.Update("tcalendarevent", data, "id = " + aEvent.Id);
00822                 return true;
00823             }
00824             catch
00825             {
00826                 return false;
00827             }
00828         }
```

```
00829
00843         public bool DeleteEvent(CalendarEvent aEvent)
00844         {
00845             try
00846             {
00847                 this.Controls.Delete("tcalendarevent", "id = " + aEvent.Id.ToString());
00848                 return true;
00849             }
00850             catch
00851             {
00852                 return false;
00853             }
00854         }
00855
00870         public bool TranscoderExist(string name, int webradioId)
00871         {
00872             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT COUNT(*) AS Count FROM
      ttranscoder WHERE webradioid = " + webradioId.ToString() + " AND name = '"+ name +"'");
00873             reader.Read();
00874             if (reader["Count"].ToString() == "0")
00875             {
00876                 reader.Close();
00877                 return false;
00878             }
00879             else
00880             {
00881                 reader.Close();
00882                 return true;
00883             }
00884         }
00885
00900         public int AddTranscoder(WebradioTranscoder transcoder, int
      webradioId)
00901         {
00902             if (this.TranscoderExist(transcoder.Name, webradioId))
00903                 return ERROR;
00904             try
00905             {
00906                 Dictionary<string, string> data = new Dictionary<string, string>();
00907                 data.Add("webradioid", webradioId.ToString());
00908                 data.Add("streamtypeid", ((int)transcoder.StreamType).ToString());
00909                 data.Add("bitrate", transcoder.Birate.ToString());
00910                 data.Add("samplerate", transcoder.SampleRate.ToString());
00911                 data.Add("name", transcoder.Name);
00912                 data.Add("url", transcoder.Url);
00913                 data.Add("port", transcoder.Port.ToString());
00914                 data.Add("adminport", transcoder.AdminPort.ToString());
00915                 data.Add("ip", transcoder.Ip.ToString());
00916                 data.Add("password", transcoder.Password);
00917                 data.Add("configfilename", transcoder.ConfigFilename);
00918                 data.Add("logfilename", transcoder.LogFilename);
00919                 this.Controls.Insert("ttranscoder", data);
00920
00921                 SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT id FROM ttranscoder WHERE
      webradioid = " + webradioId.ToString() + " AND name = '" + transcoder.Name + "'");
00922                 reader.Read();
00923                 int id = int.Parse(reader["id"].ToString());
00924                 reader.Close();
00925                 data.Clear();
00926                 data.Add("configfilename", transcoder.ConfigFilename + id.ToString() + ".config");
00927                 data.Add("logfilename", transcoder.LogFilename + "/" + id.ToString() + ".log");
00928                 this.Controls.Update("ttranscoder", data, "webradioid = " + webradioId.ToString() + " AND
      name = '" + transcoder.Name + "'");
00929                 return id;
00930             }
00931             catch
00932             {
00933                 return ERROR;
00934             }
00935
00936         }
00937
00951         public bool DeleteTranscoder(int transcoderId)
00952         {
00953             try
00954             {
00955                 this.Controls.Delete("ttranscoder", "id = " + transcoderId.ToString());
00956                 return true;
00957             }
00958             catch
00959             {
00960                 return false;
00961             }
00962         }
00963
00977         public bool UpdateTranscoder(WebradioTranscoder transcoder)
00978         {
```

```
00979             try
00980             {
00981                 Dictionary<string, string> data = new Dictionary<string, string>();
00982                 data.Add("streamtypeid", ((int)transcoder.StreamType).ToString());
00983                 data.Add("bitrate", transcoder.Birate.ToString());
00984                 data.Add("samplerate", transcoder.SampleRate.ToString());
00985                 data.Add("name", transcoder.Name);
00986                 data.Add("url", transcoder.Url);
00987                 data.Add("port", transcoder.Port.ToString());
00988                 data.Add("adminport", transcoder.AdminPort.ToString());
00989                 data.Add("ip", transcoder.Ip.ToString());
00990                 data.Add("password", transcoder.Password);
00991 
00992                 this.Controls.Update("ttranscoder", data, "id = " + transcoder.Id.ToString());
00993                 return true;
00994             }
00995             catch
00996             {
00997                 return false;
00998             }
00999         }
01000 
01018         public bool UpdateServer(int port, string password, string adminPassword, int
      maxListener, int webradioId)
01019         {
01020             try
01021             {
01022                 Dictionary<string,string> data = new Dictionary<string,string>();
01023                 data.Add("port", port.ToString());
01024                 data.Add("password", password);
01025                 data.Add("adminpassword", adminPassword);
01026                 data.Add("maxlistener", maxListener.ToString());
01027                 this.Controls.Update("tserver", data, "webradioid = " + webradioId.ToString());
01028                 return true;
01029             }
01030             catch
01031             {
01032                 return false;
01033             }
01034         }
01035 
01051         public bool AddToHistory(int transcoderId, DateTime date, string filename)
01052         {
01053             Dictionary<string, string> data = new Dictionary<string, string>();
01054             data.Add("date", date.ToString());
01055             data.Add("filename", filename);
01056             data.Add("transcoderid", transcoderId.ToString());
01057 
01058             if (this.Controls.Insert("thistory", data))
01059                 return true;
01060             else
01061                 return false;
01062 
01063         }
01064 
01078         public Dictionary<string,string> GetHistory(int transcoderId)
01079         {
01080             Dictionary<string,string> filenames = new Dictionary<string,string>();
01081             SQLiteDataReader reader = this.Controls.ExecuteDataReader("SELECT date, filename FROM thistory
       WHERE transcoderid = " + transcoderId.ToString());
01082             while(reader.Read())
01083             {
01084                 filenames.Add(reader["date"].ToString(), reader["filename"].ToString());
01085             }
01086             reader.Close();
01087             return filenames;
01088         }
01089 
01103         public bool ClearHistory(int transcoderId)
01104         {
01105             try
01106             {
01107                 this.Controls.Delete("thistory", "transcoderid = " + transcoderId.ToString());
01108                 return true;
01109             }
01110             catch
01111             {
01112                 return false;
01113             }
01114         }
01115 
01130         public bool ModifyWebradioName(string name,int webradioId)
01131         {
01132             if (this.WebradioExist(name))
01133                 return false;
01134             try
01135             {
```

```
01136                 Dictionary<string,string> data = new Dictionary<string,string>();
01137                 data.Add("name", name);
01138                 this.Controls.Update("twebradio", data, "id = " + webradioId);
01139                 return true;
01140             }
01141             catch
01142             {
01143                 return false;
01144             }
01145         }
01146 
01162         public bool UpdateFilenames(string oldName, string newName,
      Webradio webradio)
01163         {
01164             try
01165             {
01166                 Dictionary<string, string> data = new Dictionary<string, string>();
01167                 foreach (WebradioTranscoder transcoder in webradio.
      Transcoders)
01168                 {
01169                     data.Clear();
01170                     data.Add("configfilename", transcoder.ConfigFilename.Replace(oldName, newName));
01171                     data.Add("logfilename", transcoder.LogFilename.Replace(oldName, newName));
01172                     this.Controls.Update("ttranscoder", data, "webradioid = " + webradio.Id);
01173                 }
01174                 data.Clear();
01175                 data.Add("configfilename", webradio.Server.ConfigFilename.Replace(oldName, newName));
01176                 data.Add("logfilename", webradio.Server.LogFilename.Replace(oldName, newName));
01177                 this.Controls.Update("tserver", data, "webradioid = " + webradio.Id);
01178                 foreach (Playlist playlist in webradio.Playlists)
01179                 {
01180                     data.Clear();
01181                     data.Add("filename", playlist.Filename.Replace(oldName, newName));
01182                     this.Controls.Update("tplaylist", data, "webradioid = " + webradio.Id);
01183                 }
01184                 data.Clear();
01185                 data.Add("filename", webradio.Calendar.Filename.Replace(oldName, newName));
01186                 this.Controls.Update("tcalendar", data, "webradioid = " + webradio.Id);
01187                 return true;
01188             }
01189             catch
01190             {
01191                 return false;
01192             }
01193         }
01194         #endregion
01195     }
01196 }
```

7.13 CalendarEvent.cs File Reference

Implements the calendar event class.

Classes

- class WebradioManager.CalendarEvent

  A calendar event.

Namespaces

- package WebradioManager

Enumerations

- enum WebradioManager.DayValue {
  Monday = 2, **Tuesday** = 4, **Wednesday** = 8, **Thursday** = 16,
  Friday = 32, **Saturday** = 64, **Sunday** = 1 }

  Values that represent DayValue for calendar event. http://wiki.winamp.com/wiki/SHOUTcast_-Calendar_Event_XML_File_Specification#Calendar_Tag.

7.13.1 Detailed Description

Implements the calendar event class.

Definition in file CalendarEvent.cs.

7.14 CalendarEvent.cs

```
00001
00007 using System;
00008
00009 namespace WebradioManager
00010 {
00017     public enum DayValue
00018     {
00019         //< An enum constant representing the monday option
00020         Monday = 2,
00021         //< An enum constant representing the tuesday option
00022         Tuesday = 4,
00023         //< An enum constant representing the wednesday option
00024         Wednesday = 8,
00025         //< An enum constant representing the thursday option
00026         Thursday = 16,
00027         //< An enum constant representing the friday option
00028         Friday = 32,
00029         //< An enum constant representing the saturday option
00030         Saturday = 64,
00031         //< An enum constant representing the sunday option
00032         Sunday = 1,
00033     }
00034
00044     public class CalendarEvent
00045     {
00046         #region Const
00047         // \brief The monday mask.
00048         const int MONDAY_MASK = 2;
00049         // \brief The tuesday mask.
00050         const int TUESDAY_MASK = 4;
00051         // \brief The wednesday mask.
00052         const int WEDNESDAY_MASK = 8;
00053         // \brief The thursday mask.
00054         const int THURSDAY_MASK = 16;
00055         // \brief The friday mask.
00056         const int FRIDAY_MASK = 32;
00057         // \brief The saturday mask.
00058         const int SATURDAY_MASK = 64;
00059         // \brief The sunday mask.
00060         const int SUNDAY_MASK = 1;
00061         #endregion
00062
00063         #region Fields
00064         // \brief The name.
00065         private string _name;
00066         // \brief The start time.
00067         private TimeSpan _startTime;
00068         // \brief The duration.
00069         private TimeSpan _duration;
00070         // \brief The repeat.
00071         private int _repeat;
00072         // \brief true to shuffle.
00073         private bool _shuffle;
00074         // \brief true to loopatend.
00075         private bool _loopatend;
00076         // \brief The priority number.
00077         private int _priority;
00078         // \brief The playlist.
00079         private Playlist _playlist;
00080         // \brief The identifier.
00081         private int _id;
00082
00083         #endregion
00084
00085         #region Properties
00086
00095         public int Id
00096         {
00097             get { return _id; }
00098             set { _id = value; }
00099         }
00100
00109         public Playlist Playlist
00110         {
```

```
00111             get { return _playlist; }
00112             set { _playlist = value; }
00113         }
00114
00123         public string Name
00124         {
00125             get { return _name; }
00126             set { _name = value; }
00127         }
00128
00137         public TimeSpan StartTime
00138         {
00139             get { return _startTime; }
00140             set { _startTime = value; }
00141         }
00142
00151         public TimeSpan Duration
00152         {
00153             get { return _duration; }
00154             set { _duration = value; }
00155         }
00156
00165         public int Repeat
00166         {
00167             get { return _repeat; }
00168             set { _repeat = value; }
00169         }
00170
00179         public bool Shuffle
00180         {
00181             get { return _shuffle; }
00182             set { _shuffle = value; }
00183         }
00184
00193         public bool Loopatend
00194         {
00195             get { return _loopatend; }
00196             set { _loopatend = value; }
00197         }
00198
00207         public int Priority
00208         {
00209             get { return _priority; }
00210             set { _priority = value; }
00211         }
00212         #endregion
00213
00214         #region Methods
00215
00233         public CalendarEvent(string name, TimeSpan starttime, TimeSpan duration, int repeat,
    int priority, bool shuffle, bool loopatend, Playlist playlist)
00234             :this(0,name,starttime,duration,repeat,priority,shuffle,loopatend,playlist)
00235         {
00236             //NO CODE
00237         }
00238
00258         public CalendarEvent(int id, string name, TimeSpan starttime, TimeSpan duration, int
    repeat, int priority, bool shuffle, bool loopatend, Playlist playlist)
00259         {
00260             this.Id = id;
00261             this.Name = name;
00262             this.StartTime = starttime;
00263             this.Duration = duration;
00264             this.Repeat = repeat;
00265             this.Priority = priority;
00266             this.Shuffle = shuffle;
00267             this.Loopatend = loopatend;
00268             this.Playlist = playlist;
00269         }
00270
00283         public DayWeek GetSelectedDays()
00284         {
00285             DayWeek dow = new DayWeek();
00286             dow.Monday = Convert.ToBoolean(this.Repeat & MONDAY_MASK);
00287             dow.Tuesday = Convert.ToBoolean(this.Repeat & TUESDAY_MASK);
00288             dow.Wednesday = Convert.ToBoolean(this.Repeat & WEDNESDAY_MASK);
00289             dow.Thursday = Convert.ToBoolean(this.Repeat & THURSDAY_MASK);
00290             dow.Friday = Convert.ToBoolean(this.Repeat & FRIDAY_MASK);
00291             dow.Saturday = Convert.ToBoolean(this.Repeat & SATURDAY_MASK);
00292             dow.Sunday = Convert.ToBoolean(this.Repeat & SUNDAY_MASK);
00293             return dow;
00294         }
00295         #endregion
00296     }
00297 }
```

7.15 DayWeek.cs File Reference

Implements the day week struct.

Classes

- struct WebradioManager.DayWeek

  A week with boolean values for each day. Uses for store which day is selected for an CalendarEvent.

Namespaces

- package WebradioManager

7.15.1 Detailed Description

Implements the day week struct.

Definition in file DayWeek.cs.

7.16 DayWeek.cs

```
00001
00007 namespace WebradioManager
00008 {
00018     public struct DayWeek
00019     {
00020         #region Const
00021         // \brief Number of days.
00022         const int DAYS_COUNT = 7;
00023         #endregion
00024
00025         #region Fields
00026         // \brief true to monday.
00027         private bool _monday;
00028         // \brief true to tuesday.
00029         private bool _tuesday;
00030         // \brief true to wednesday.
00031         private bool _wednesday;
00032         // \brief true to thursday.
00033         private bool _thursday;
00034         // \brief true to friday.
00035         private bool _friday;
00036         // \brief true to saturday.
00037         private bool _saturday;
00038         // \brief true to sunday.
00039         private bool _sunday;
00040         #endregion
00041
00042         #region Properties
00043
00052         public bool Monday
00053         {
00054             get { return _monday; }
00055             set { _monday = value; }
00056         }
00057
00066         public bool Tuesday
00067         {
00068             get { return _tuesday; }
00069             set { _tuesday = value; }
00070         }
00071
00080         public bool Wednesday
00081         {
00082             get { return _wednesday; }
00083             set { _wednesday = value; }
00084         }
00085
00094         public bool Thursday
00095         {
00096             get { return _thursday; }
```

```
00097             set { _thursday = value; }
00098         }
00099
00108         public bool Friday
00109         {
00110             get { return _friday; }
00111             set { _friday = value; }
00112         }
00113
00122         public bool Saturday
00123         {
00124             get { return _saturday; }
00125             set { _saturday = value; }
00126         }
00127
00136         public bool Sunday
00137         {
00138             get { return _sunday; }
00139             set { _sunday = value; }
00140         }
00141         #endregion
00142
00143         #region Methods
00144
00156         public bool[] ToArray()
00157         {
00158             bool[] array = new bool[DAYS_COUNT];
00159             array[0] = this.Monday;
00160             array[1] = this.Tuesday;
00161             array[2] = this.Wednesday;
00162             array[3] = this.Thursday;
00163             array[4] = this.Friday;
00164             array[5] = this.Saturday;
00165             array[6] = this.Sunday;
00166             return array;
00167         }
00168         #endregion
00169     }
00170 }
```

7.17 EventAppointment.cs File Reference

Implements the event appointment class.

Classes

- class WebradioManager.EventAppointment

  An event appointment. Add 2 properties to the original Appointment class (from Calendar library).

Namespaces

- package WebradioManager

7.17.1 Detailed Description

Implements the event appointment class.

Definition in file EventAppointment.cs.

7.18 EventAppointment.cs

```
00001
00007 using Calendar;
00008
00009 namespace WebradioManager
00010 {
00020     public class EventAppointment : Appointment
00021     {
```

```
00022         #region Fields
00023         // \brief The playlist.
00024         private Playlist _playlist;
00025         // \brief The event object.
00026         private CalendarEvent _eventObject;
00027         #endregion
00028 
00029         #region Properties
00030 
00038         public CalendarEvent EventObject
00039         {
00040             get { return _eventObject; }
00041             set { _eventObject = value; }
00042         }
00043 
00052         public Playlist Playlist
00053         {
00054             get { return _playlist; }
00055             set { _playlist = value; }
00056         }
00057         #endregion
00058 
00059         #region Methods
00060 
00069         public EventAppointment():base()
00070         {
00071             //NO CODE
00072 
00073         }
00074         #endregion
00075     }
00076 }
```

7.19 IController.cs File Reference

Declares the IController interface.

Classes

- interface WebradioManager.IController

  Interface for controller.

Namespaces

- package WebradioManager

7.19.1 Detailed Description

Declares the IController interface.

Definition in file IController.cs.

7.20 IController.cs

```
00001 
00008 namespace WebradioManager
00009 {
00019     public interface IController
00020     {
00027         void UpdateView();
00028     }
00029 }
```

7.21 Music.cs File Reference

Implements the music class.

Classes

- class WebradioManager.Music

  A music.

Namespaces

- package WebradioManager

7.21.1 Detailed Description

Implements the music class.

Definition in file Music.cs.

7.22 Music.cs

```
00001
00007 using System;
00008
00009 namespace WebradioManager
00010 {
00020     public class Music : AudioFile
00021     {
00022         #region Methods
00023
00042         public Music(int id, string filename, string title, string artist, string album, int year,
      string label, TimeSpan duration, string gender):
00043             base(id,filename,title,artist,album,year,label,duration,gender,
      AudioType.Music)
00044         {
00045
00046         }
00047
00066         public Music(string filename, string title, string artist, string album, int year, string
      label, TimeSpan duration, string gender) :
00067             base(filename, title, artist, album, year, label, duration, gender,
      AudioType.Music)
00068         {
00069
00070         }
00071         #endregion
00072     }
00073 }
```

7.23 Playlist.cs File Reference

Implements the playlist class.

Classes

- class WebradioManager.Playlist

  A playlist. Abstract class.

Namespaces

- package WebradioManager

7.23.1 Detailed Description

Implements the playlist class.

Definition in file Playlist.cs.

7.24 Playlist.cs

```
00001
00007 using System.Collections.Generic;
00008 using System.IO;
00009
00010 namespace WebradioManager
00011 {
00021     public abstract class Playlist
00022     {
00023         #region Const
00024         // \brief The default identifier.
00025         const int DEFAULT_ID = 0;
00026         #endregion
00027
00028         #region Fields
00029         // \brief The name.
00030         private string _name;
00031         // \brief The identifier.
00032         private int _id;
00033         // \brief Filename of the file.
00034         private string _filename;
00035         // \brief The type.
00036         private AudioType _type;
00037         // \brief List of audio files's filename.
00038         private List<string> _audioFileList;
00039         #endregion
00040
00041         #region Properties
00042
00051         public List<string> AudioFileList
00052         {
00053             get { return _audioFileList; }
00054             set { _audioFileList = value; }
00055         }
00056
00065         public AudioType Type
00066         {
00067             get { return _type; }
00068             set { _type = value; }
00069         }
00070
00079         public int Id
00080         {
00081             get { return _id; }
00082             set { _id = value; }
00083         }
00084
00093         public string Filename
00094         {
00095             get { return _filename; }
00096             set { _filename = value; }
00097         }
00098
00107         public string Name
00108         {
00109             get { return _name; }
00110             set { _name = value; }
00111         }
00112         #endregion
00113
00114         #region Methods
00115
00129         public Playlist(string name, string filename, AudioType type):this(DEFAULT_ID,name
      ,filename,type)
00130         {
00131             //NO CODE
00132         }
00133
```

```
00148         public Playlist(int id, string name, string filename, AudioType type)
00149         {
00150             this.Id = id;
00151             this.Name = name;
00152             this.Filename = filename;
00153             this.Type = type;
00154             this.AudioFileList = new List<string>();
00155         }
00156 
00166         public void GenerateConfigFile()
00167         {
00168             string output = "";
00169             if (File.Exists(this.Filename))
00170                 File.Delete(this.Filename);
00171             foreach(string filename in this.AudioFileList)
00172             {
00173                 output += (filename + "\n");
00174             }
00175             File.WriteAllText(this.Filename, output);
00176         }
00177 
00191         public override string ToString()
00192         {
00193             return this.Name;
00194         }
00195         #endregion
00196 
00197     }
00198 }
```

7.25 PlaylistAd.cs File Reference

Implements the playlist ad class.

Classes

- class WebradioManager.PlaylistAd

  A playlist ad.

Namespaces

- package WebradioManager

7.25.1 Detailed Description

Implements the playlist ad class.

Definition in file PlaylistAd.cs.

7.26 PlaylistAd.cs

```
00001 
00007 namespace WebradioManager
00008 {
00018     public class PlaylistAd : Playlist
00019     {
00020         #region Methods
00021 
00034         public PlaylistAd(int id, string name, string filename):base(id,name,filename,
      AudioType.Ad)
00035         {
00036 
00037         }
00038 
00051         public PlaylistAd(string name, string filename)
00052             : base(name, filename, AudioType.Ad)
00053         {
00054 
```

```
00055         }
00056         #endregion
00057     }
00058 }
```

7.27 PlaylistMusic.cs File Reference

Implements the playlist music class.

Classes

- class WebradioManager.PlaylistMusic

  A playlist music.

Namespaces

- package WebradioManager

7.27.1 Detailed Description

Implements the playlist music class.

Definition in file PlaylistMusic.cs.

7.28 PlaylistMusic.cs

```
00001
00007 namespace WebradioManager
00008 {
00018     public class PlaylistMusic : Playlist
00019     {
00020         #region Methods
00021
00034         public PlaylistMusic(int id, string name, string filename)
00035             : base(id, name, filename, AudioType.Music)
00036         {
00037
00038         }
00039
00052         public PlaylistMusic(string name, string filename)
00053             : base(name, filename, AudioType.Music)
00054         {
00055
00056         }
00057         #endregion
00058     }
00059 }
```

7.29 SelectionController.cs File Reference

Implements the selection controller class.

Classes

- class WebradioManager.SelectionController

  A controller for SelectionView.

Namespaces

- package WebradioManager

7.29.1 Detailed Description

Implements the selection controller class.

Definition in file SelectionController.cs.

7.30 SelectionController.cs

```
00001
00007 using System.Collections.Generic;
00008
00009 namespace WebradioManager
00010 {
00020     public class SelectionController : IController
00021     {
00022         #region Fields
00023         // \brief The view.
00024         private SelectionView _view;
00025         // \brief The model.
00026         private WMModel _model;
00027         #endregion
00028
00029         #region Properties
00030
00039         public WMModel Model
00040         {
00041             get { return _model; }
00042             set { _model = value; }
00043         }
00044
00053         public SelectionView View
00054         {
00055             get { return _view; }
00056             set { _view = value; }
00057         }
00058         #endregion
00059
00060         #region Methods
00061
00073         public SelectionController(SelectionView view)
00074         {
00075             this.View = view;
00076             this.Model = new WMModel();
00077             this.Model.AddObserver(this);
00078         }
00079
00089         public void LoadLibrary()
00090         {
00091             this.Model.LoadLibrary();
00092         }
00093
00103         public void LoadWebradios()
00104         {
00105             this.Model.LoadWebradios();
00106         }
00107
00117         public void UpdateView()
00118         {
00119             this.View.UpdateView();
00120         }
00121
00133         public List<Webradio> GetWebradios()
00134         {
00135             return this.Model.GetWebradios();
00136         }
00137
00151         public bool CreateWebradio(string name)
00152         {
00153             return this.Model.CreateWebradio(name);
00154         }
00155
00169         public bool DeleteWebradio(int id)
00170         {
00171             return this.Model.DeleteWebradio(id);
00172         }
```

```
00173
00187         public bool DuplicateWebradio(int id)
00188         {
00189             return this.Model.DuplicateWebradio(id);
00190         }
00191
00203         public void OpenWebradio(int id)
00204         {
00205
00206             AdminController admincontroller = new AdminController(id, this.
      Model);
00207             this.Model.AddObserver(admincontroller);
00208         }
00209
00221         public bool StopAllProcess()
00222         {
00223             return this.Model.StopAllProcess();
00224         }
00225         #endregion
00226
00227     }
00228 }
```

7.31 SelectionView.cs File Reference

Implements the selection view class.

Classes

- class WebradioManager.SelectionView

  A selection view.

Namespaces

- package WebradioManager

7.31.1 Detailed Description

Implements the selection view class.

Definition in file SelectionView.cs.

7.32 SelectionView.cs

```
00001
00007 using System;
00008 using System.Collections.Generic;
00009 using System.Windows.Forms;
00010
00011 namespace WebradioManager
00012 {
00022     public partial class SelectionView : Form
00023     {
00024         #region Const
00025         // \brief The maximum name length.
00026         const int MAX_NAME_LENGTH = 255;
00027         #endregion
00028
00029         #region Fields
00030         // \brief The controller.
00031         private SelectionController _controller;
00032         #endregion
00033
00034         #region Properties
00035
00044         public SelectionController Controller
00045         {
00046             get { return _controller; }
```

```
00047             set { _controller = value; }
00048         }
00049         #endregion
00050
00051         #region Methods
00052
00062         public SelectionView()
00063         {
00064             InitializeComponent();
00065             this.Cursor = Cursors.WaitCursor;
00066             this.Controller = new SelectionController(this);
00067             this.Controller.LoadWebradios();
00068             this.Controller.LoadLibrary();
00069             this.UpdateView();
00070             this.Cursor = Cursors.Default;
00071         }
00072
00082         public void UpdateView()
00083         {
00084             lsbSelection.Items.Clear();
00085             List<Webradio> webradios = this.Controller.GetWebradios();
00086             foreach(Webradio wr in webradios)
00087             {
00088                 lsbSelection.Items.Add(wr);
00089             }
00090         }
00091
00105         private void btnNew_Click(object sender, EventArgs e)
00106         {
00107             if (!string.IsNullOrEmpty(txbName.Text.Trim()) && txbName.Text.Length <= MAX_NAME_LENGTH)
00108             {
00109                 if (this.Controller.CreateWebradio(txbName.Text))
00110                     MessageBox.Show("Webradio created !");
00111                 else
00112                     MessageBox.Show("An error occured. (Invalid name or cannot create folders and files.)",
      "Error");
00113                 this.UpdateView();
00114             }
00115             else
00116                 MessageBox.Show("Please enter a valid webradio's name. (1-255 characters)", "Error");
00117         }
00118
00132         private void btnDelete_Click(object sender, EventArgs e)
00133         {
00134             if (this.lsbSelection.SelectedIndex >= 0)
00135             {
00136                 int id = ((Webradio)this.lsbSelection.SelectedItem).Id;
00137                 if (!this.Controller.DeleteWebradio(id))
00138                     MessageBox.Show("An error occured.", "Error");
00139                 this.UpdateView();
00140             }
00141             else
00142                 MessageBox.Show("Please select a webradio to delete.", "No webradio selected",
      MessageBoxButtons.OK, MessageBoxIcon.Error);
00143         }
00144
00158         private void btnDuplicate_Click(object sender, EventArgs e)
00159         {
00160             if(this.lsbSelection.SelectedIndex >= 0)
00161             {
00162                 int id = ((Webradio)this.lsbSelection.SelectedItem).Id;
00163                 if (!this.Controller.DuplicateWebradio(id))
00164                     MessageBox.Show("An error occured", "Error");
00165             }
00166             else
00167                 MessageBox.Show("Please select a webradio to duplicate.", "No webradio selected",
      MessageBoxButtons.OK, MessageBoxIcon.Error);
00168         }
00169
00183         private void btnOpen_Click(object sender, EventArgs e)
00184         {
00185             if(this.lsbSelection.SelectedIndex >= 0)
00186                 this.Controller.OpenWebradio(((Webradio)this.lsbSelection.SelectedItem).Id);
00187
00188         }
00189
00202         private void SelectionView_FormClosing(object sender, FormClosingEventArgs e)
00203         {
00204             if (MessageBox.Show("All transcoders and all servers will be shuting down. Are you sure ?", "
      Close", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
00205             {
00206                 if (!this.Controller.StopAllProcess())
00207                 {
00208                     MessageBox.Show("An error has occured", "Error");
00209                     e.Cancel = true;
00210                 }
00211             }
```

```
00212             else
00213                 e.Cancel = true;
00214         }
00215         #endregion
00216     }
00217 }
```

7.33 StreamType.cs File Reference

Implements the stream type class.

Namespaces

- package WebradioManager

Enumerations

- enum WebradioManager.StreamType { **MP3** = 1, **AACPlus** = 2 }

  Values that represent StreamType's id. Defined in DB.

7.33.1 Detailed Description

Implements the stream type class.

Definition in file StreamType.cs.

7.34 StreamType.cs

```
00001
00007 namespace WebradioManager
00008 {
00015     public enum StreamType
00016     {
00017         //< An enum constant representing the mp3 option
00018         MP3 = 1,
00019         //< An enum constant representing the aac plus option
00020         AACPlus = 2,
00021     }
00022 }
```

7.35 TranscoderAacPlus.cs File Reference

Implements the transcoder aac plus class.

Classes

- class WebradioManager.TranscoderAacPlus

  A transcoder aac plus.

Namespaces

- package WebradioManager

7.35.1 Detailed Description

Implements the transcoder aac plus class.

Definition in file TranscoderAacPlus.cs.

7.36 TranscoderAacPlus.cs

```
00001
00007 using System.Net;
00008
00009 namespace WebradioManager
00010 {
00020     public class TranscoderAacPlus : WebradioTranscoder
00021     {
00022         #region Methods
00023
00044         public TranscoderAacPlus(int id, string name, int bitrate, int sampleRate,
      IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename)
00045             :base(id,name,bitrate,sampleRate,ip,port,adminport, url,password,configFilename,logFilename,
      StreamType.AACPlus)
00046         {
00047
00048         }
00049
00070         public TranscoderAacPlus(string name, int bitrate, int sampleRate, IPAddress ip,
      int port, int adminport, string url, string password, string configFilename, string logFilename)
00071             : base(name, bitrate, sampleRate, ip, port, adminport, url, password, configFilename,
      logFilename, StreamType.AACPlus)
00072         {
00073
00074         }
00075         #endregion
00076     }
00077 }
```

7.37 TranscoderMp3.cs File Reference

Implements the transcoder mp 3 class.

Classes

- class WebradioManager.TranscoderMp3

  A transcoder mp3.

Namespaces

- package WebradioManager

7.37.1 Detailed Description

Implements the transcoder mp 3 class.

Definition in file TranscoderMp3.cs.

7.38 TranscoderMp3.cs

```
00001
00007 using System.Net;
00008
00009 namespace WebradioManager
00010 {
```

```
00020     public class TranscoderMp3 : WebradioTranscoder
00021     {
00022         #region Methods
00023 
00044         public TranscoderMp3(int id, string name, int bitrate, int sampleRate, IPAddress ip,
      int port, int adminport, string url, string password, string configFilename, string logFilename)
00045             :base(id,name,bitrate,sampleRate,ip,port, adminport, url,password,configFilename,logFilename,
      StreamType.MP3)
00046         {
00047 
00048         }
00049 
00070         public TranscoderMp3(string name, int bitrate, int sampleRate, IPAddress ip, int port,
       int adminport, string url, string password, string configFilename, string logFilename)
00071             : base(name, bitrate, sampleRate, ip, port, adminport, url, password, configFilename,
      logFilename, StreamType.MP3)
00072         {
00073 
00074         }
00075         #endregion
00076     }
00077 }
```

7.39 Webradio.cs File Reference

Implements the webradio class.

Classes

- class WebradioManager.Webradio

  A webradio.

Namespaces

- package WebradioManager

7.39.1 Detailed Description

Implements the webradio class.

Definition in file Webradio.cs.

7.40 Webradio.cs

```
00001 
00007 using System.Collections.Generic;
00008 
00009 namespace WebradioManager
00010 {
00020     public class Webradio
00021     {
00022         #region Const
00023         // \brief The default identifier.
00024         const int DEFAULT_ID = 0;
00025         #endregion
00026 
00027         #region Fields
00028         // \brief The playlists list.
00029         private List<Playlist> _playlists;
00030         // \brief The calendar.
00031         private WebradioCalendar _calendar;
00032         // \brief The name.
00033         private string _name;
00034         // \brief The server.
00035         private WebradioServer _server;
00036         // \brief The identifier.
00037         private int _id;
00038         // \brief The transcoders list.
```

```
00039         private List<WebradioTranscoder> _transcoders;
00040         #endregion
00041
00042         #region Properties
00043
00052         public List<WebradioTranscoder> Transcoders
00053         {
00054             get { return _transcoders; }
00055             set { _transcoders = value; }
00056         }
00057
00066         public WebradioServer Server
00067         {
00068             get { return _server; }
00069             set { _server = value; }
00070         }
00071
00080         public string Name
00081         {
00082             get { return _name; }
00083             set { _name = value; }
00084         }
00085
00094         public WebradioCalendar Calendar
00095         {
00096             get { return _calendar; }
00097             set { _calendar = value; }
00098         }
00099
00108         public List<Playlist> Playlists
00109         {
00110             get { return _playlists; }
00111             set { _playlists = value; }
00112         }
00113
00122         public int Id
00123         {
00124             get { return _id; }
00125             set { _id = value; }
00126         }
00127         #endregion
00128
00129         #region Methods
00130
00143         public Webradio(string name, int id)
00144         {
00145             this.Name = name;
00146             this.Id = id;
00147             this.Playlists = new List<Playlist>();
00148             this.Transcoders = new List<WebradioTranscoder>();
00149         }
00150
00162         public Webradio(string name):this(name,DEFAULT_ID)
00163         {
00164             //NO CODE
00165         }
00166
00176         public void GenerateConfigFiles()
00177         {
00178             foreach (Playlist playlist in this.Playlists)
00179             {
00180                 playlist.GenerateConfigFile();
00181             }
00182             this.Calendar.GenerateConfigFile();
00183             this.Server.GenerateConfigFile();
00184             foreach (WebradioTranscoder transcoder in this.
      Transcoders)
00185             {
00186                 transcoder.GenerateConfigFile(this.Playlists);
00187             }
00188         }
00189
00203         public override string ToString()
00204         {
00205             return this.Name + " | ID = " + this.Id.ToString();
00206         }
00207         #endregion
00208     }
00209 }
```

7.41 WebradioCalendar.cs File Reference

Implements the webradio calendar class.

Classes

- class WebradioManager.WebradioCalendar

  A webradio calendar.

Namespaces

- package WebradioManager

7.41.1 Detailed Description

Implements the webradio calendar class.

Definition in file WebradioCalendar.cs.

7.42 WebradioCalendar.cs

```
00001
00007 using System.Collections.Generic;
00008 using System.IO;
00009 using System.Xml;
00010
00011 namespace WebradioManager
00012 {
00022     public class WebradioCalendar
00023     {
00024         #region Fields
00025         // \brief The events.
00026         private List<CalendarEvent> _events;
00027         // \brief Filename of the calendar's file.
00028         private string _filename;
00029         // \brief The identifier.
00030         private int _id;
00031         #endregion
00032
00033         #region Properties
00034
00043         public int Id
00044         {
00045           get { return _id; }
00046           set { _id = value; }
00047         }
00048
00057         public string Filename
00058         {
00059             get { return _filename; }
00060             set { _filename = value; }
00061         }
00062
00071         public List<CalendarEvent> Events
00072         {
00073             get { return _events; }
00074             set { _events = value; }
00075         }
00076         #endregion
00077
00078         #region Methods
00079
00092         public WebradioCalendar(int id, string filename)
00093         {
00094             this.Id = id;
00095             this.Filename = filename;
00096             this.Events = new List<CalendarEvent>();
00097         }
00098
00110         public WebradioCalendar(string filename)
00111         {
00112             this.Filename = filename;
00113             this.Events = new List<CalendarEvent>();
00114         }
00115
00125         public void GenerateConfigFile()
00126         {
00127             if (File.Exists(this.Filename))
00128                 File.Delete(this.Filename);
```

```
00129             XmlDocument document = new XmlDocument();
00130             XmlElement root = document.CreateElement("eventlist");
00131             foreach(CalendarEvent ev in this.Events)
00132             {
00133                 XmlElement eventelement = document.CreateElement("event");
00134                 eventelement.SetAttribute("type", "playlist");
00135                 XmlElement playlist = document.CreateElement("playlist");
00136                 playlist.SetAttribute("loopatend", (ev.Loopatend)?"1":"0");
00137                 playlist.SetAttribute("shuffle", (ev.Shuffle) ? "1" : "0");
00138                 playlist.SetAttribute("priority", ev.Priority.ToString());
00139                 playlist.InnerText = ev.Playlist.Name;
00140                 eventelement.AppendChild(playlist);
00141 
00142                 XmlElement calendar = document.CreateElement("calendar");
00143                 calendar.SetAttribute("starttime", ev.StartTime.ToString(@"hh\:mm\:ss"));
00144                 calendar.SetAttribute("duration", ev.Duration.ToString(@"hh\:mm\:ss"));
00145                 calendar.SetAttribute("repeat", ev.Repeat.ToString());
00146                 eventelement.AppendChild(calendar);
00147                 root.AppendChild(eventelement);
00148             }
00149 
00150             document.AppendChild(root);
00151             document.Save(this.Filename);
00152         }
00153         #endregion
00154     }
00155 }
```

7.43 WebradioListener.cs File Reference

Implements the webradio listener class.

Classes

- class WebradioManager.WebradioListener

  A webradio listener.

Namespaces

- package WebradioManager

7.43.1 Detailed Description

Implements the webradio listener class.

Definition in file WebradioListener.cs.

7.44 WebradioListener.cs

```
00001 
00008 namespace WebradioManager
00009 {
00019     public class WebradioListener
00020     {
00021         #region Fields
00022         // \brief The hostname.
00023         private string _hostname;
00024         // \brief The useragent.
00025         private string _useragent;
00026         // \brief The connection time.
00027         private uint _connectionTime;
00028         // \brief The UID.
00029         private int _uid;
00030         #endregion
00031 
00032         #region Properties
00033 
00042         public int Uid
```

```
00043         {
00044             get { return _uid; }
00045             set { _uid = value; }
00046         }
00047 
00056         public uint ConnectionTime
00057         {
00058             get { return _connectionTime; }
00059             set { _connectionTime = value; }
00060         }
00061 
00070         public string Useragent
00071         {
00072             get { return _useragent; }
00073             set { _useragent = value; }
00074         }
00075 
00084         public string Hostname
00085         {
00086             get { return _hostname; }
00087             set { _hostname = value; }
00088         }
00089 
00090         #endregion
00091 
00092         #region Methods
00093 
00108         public WebradioListener(string hostname, string useragent, uint connectiontime, int
      uid)
00109         {
00110             this.Hostname = hostname;
00111             this.Useragent = useragent;
00112             this.ConnectionTime = connectiontime;
00113             this.Uid = uid;
00114         }
00115         #endregion
00116     }
00117 }
```

7.45 WebradioServer.cs File Reference

Implements the webradioserver class.

Classes

- class WebradioManager.WebradioServer

  A shoutcast webradio server.

Namespaces

- package WebradioManager

7.45.1 Detailed Description

Implements the webradioserver class.

Definition in file WebradioServer.cs.

7.46 WebradioServer.cs

```
00001 
00007 using System;
00008 using System.Collections.Generic;
00009 using System.Diagnostics;
00010 using System.IO;
00011 using System.Net;
00012 using System.Net.Sockets;
00013 using System.Xml;
```

```
00014
00015 namespace WebradioManager
00016 {
00026     public class WebradioServer
00027     {
00028         #region Fields
00029         // \brief Filename of the server file.
00030         const string SC_SERVER_FILENAME = "\\shoutcast\\sc_serv.exe";
00031         // \brief The default admin login.
00032         public const string DEFAULT_ADMIN_LOGIN = "admin";
00033
00034         // \brief The port.
00035         private int _port;
00036         // \brief Filename of the log file.
00037         private string _logFilename;
00038         // \brief Filename of the configuration file.
00039         private string _configFilename;
00040         // \brief The password.
00041         private string _password;
00042         // \brief The admin password.
00043         private string _adminPassword;
00044         // \brief The process.
00045         private Process _process;
00046         // \brief The number of maximum listener.
00047         private int _maxListener;
00048         // \brief The statistics.
00049         private WebradioServerStats _stats;
00050
00051         #endregion
00052
00053         #region Properties
00054
00063         internal WebradioServerStats Stats
00064         {
00065             get { return _stats; }
00066             set { _stats = value; }
00067         }
00068
00077         public string WebInterfaceUrl
00078         {
00079             get
00080             {
00081                 return "http://" + this.GetLocalIPAddress() + ":" + this.Port;
00082             }
00083         }
00084
00093         public string WebAdminUrl
00094         {
00095             get
00096             {
00097                 return "http://" + this.GetLocalIPAddress() + ":" + this.Port + "/admin.cgi";
00098             }
00099         }
00100
00109         public int MaxListener
00110         {
00111             get { return _maxListener; }
00112             set { _maxListener = value; }
00113         }
00114
00123         public Process Process
00124         {
00125             get { return _process; }
00126             set { _process = value; }
00127         }
00128
00137         public string LogFilename
00138         {
00139             get { return _logFilename; }
00140             set { _logFilename = value; }
00141         }
00142
00151         public string ConfigFilename
00152         {
00153             get { return _configFilename; }
00154             set { _configFilename = value; }
00155         }
00156
00165         public string Password
00166         {
00167             get { return _password; }
00168             set { _password = value; }
00169         }
00170
00179         public string AdminPassword
00180         {
00181             get { return _adminPassword; }
```

```
00182             set { _adminPassword = value; }
00183         }
00184
00193         public int Port
00194         {
00195             get { return _port; }
00196             set { _port = value; }
00197         }
00198
00199         #endregion
00200
00201         #region Methods
00202
00218         public WebradioServer(int port, string logFilename, string configFilename, string
      password, string adminPassword, int maxListener)
00219         {
00220             this.Port = port;
00221             this.LogFilename = logFilename;
00222             this.ConfigFilename = configFilename;
00223             this.Password = password;
00224             this.AdminPassword = adminPassword;
00225             this.MaxListener = maxListener;
00226             this.Stats = new WebradioServerStats();
00227         }
00228
00238         public void GenerateConfigFile()
00239         {
00240             if (File.Exists(this.ConfigFilename))
00241                 File.Delete(this.ConfigFilename);
00242             string output = "";
00243             output += "logfile=" + Directory.GetCurrentDirectory() + "\\" + this.
      LogFilename.Replace('/', '\\') + "\n";
00244             output += "portbase=" + this.Port.ToString() + "\n";
00245             output += "password=" + this.Password + "\n";
00246             output += "adminpassword=" + this.AdminPassword + "\n";
00247             output += "publicserver=always" + "\n";
00248             output += "maxuser=" + this.MaxListener.ToString() + "\n";
00249             //Don't kick idle source
00250             output += "autodumpsourcetime=0\n";
00251             File.WriteAllText(this.ConfigFilename, output);
00252         }
00253
00265         public bool IsRunning()
00266         {
00267             bool result = false;
00268             try
00269             {
00270                 foreach (Process prc in Process.GetProcesses())
00271                 {
00272                     if (prc.Id == this.Process.Id)
00273                     {
00274                         result = true;
00275                     }
00276                 }
00277                 return result;
00278             }
00279             catch
00280             {
00281                 return false;
00282             }
00283         }
00284
00298         public bool Start(bool debug)
00299         {
00300             ProcessStartInfo StartInfo = new ProcessStartInfo(Directory.GetCurrentDirectory() +
      SC_SERVER_FILENAME)
00301             {
00302                 CreateNoWindow = true,
00303                 WindowStyle = (debug)?ProcessWindowStyle.Minimized:ProcessWindowStyle.Hidden,
00304                 Arguments = "\"" + Directory.GetCurrentDirectory() + "\\" + this.ConfigFilename.Replace('/'
      , '\\') + "\""
00305             };
00306             if (this.IsRunning())
00307                 this.Process.Kill();
00308             try
00309             {
00310                 this.Process = Process.Start(StartInfo);
00311                 return true;
00312             }
00313             catch
00314             {
00315                 return false;
00316             }
00317         }
00318
00330         public bool Stop()
00331         {
```

```
00332             if (this.IsRunning() && this.Process.Responding)
00333             {
00334                 try
00335                 {
00336                     this.Process.Kill();
00337                     return true;
00338                 }
00339                 catch
00340                 {
00341                     return false;
00342                 }
00343             }
00344             return true;
00345         }
00346 
00358         private string GetLocalIPAddress()
00359         {
00360             IPHostEntry host;
00361             string localIP = "";
00362             host = Dns.GetHostEntry(Dns.GetHostName());
00363             foreach (IPAddress ip in host.AddressList)
00364             {
00365                 if (ip.AddressFamily == AddressFamily.InterNetwork)
00366                 {
00367                     localIP = ip.ToString();
00368                     break;
00369                 }
00370             }
00371             return localIP;
00372         }
00373 
00385         public bool UpdateStats()
00386         {
00387             try
00388             {
00389                 WebClient wc = new WebClient();
00390                 wc.Credentials = new NetworkCredential(DEFAULT_ADMIN_LOGIN, this.
      AdminPassword);
00391                 string response = wc.DownloadString("http://127.0.0.1:" + this.Port + "
      /admin.cgi?mode=viewxml&page=1&sid=1");
00392                 XmlDocument doc = new XmlDocument();
00393                 doc.LoadXml(response);
00394                 XmlNodeList nodes = doc.SelectNodes("/SHOUTCASTSERVER");
00395                 foreach(XmlNode xn in nodes)
00396                 {
00397                     this.Stats.CurrentListeners = int.Parse(xn["CURRENTLISTENERS"].InnerText);
00398                     this.Stats.PeakListeners = int.Parse(xn["PEAKLISTENERS"].InnerText);
00399                     this.Stats.UniqueListeners = int.Parse(xn["UNIQUELISTENERS"].InnerText);
00400                     this.Stats.AverageTime = TimeSpan.FromSeconds(double.Parse(xn["AVERAGETIME"].InnerText)
      );
00401                 }
00402                 return true;
00403             }
00404             catch
00405             {
00406                 return false;
00407             }
00408         }
00409 
00421         public List<WebradioListener> GetListeners()
00422         {
00423             List<WebradioListener> list = new List<WebradioListener>();
00424             if (this.IsRunning())
00425             {
00426                 WebClient wc = new WebClient();
00427                 wc.Credentials = new NetworkCredential(DEFAULT_ADMIN_LOGIN, this.
      AdminPassword);
00428                 string response = wc.DownloadString("http://127.0.0.1:" + this.Port + "
      /admin.cgi?mode=viewxml&page=3&sid=1");
00429 
00430                 XmlDocument doc = new XmlDocument();
00431                 doc.LoadXml(response);
00432                 XmlNodeList nodes = doc.SelectNodes("/SHOUTCASTSERVER/LISTENERS/LISTENER");
00433                 if (nodes.Count > 0)
00434                 {
00435                     foreach (XmlNode xn in nodes)
00436                     {
00437                         WebradioListener listener = new
      WebradioListener(xn["HOSTNAME"].InnerText, xn["USERAGENT"].InnerText, uint.Parse(xn["
      CONNECTTIME"].InnerText), int.Parse(xn["UID"].InnerText));
00438                         list.Add(listener);
00439                     }
00440                 }
00441             }
00442             return list;
00443         }
00444         #endregion
```

```
00445     }
00446 }
```

7.47 WebradioServerStats.cs File Reference

Implements the webradio server statistics class.

Classes

- class WebradioManager.WebradioServerStats

  A webradio server statistics.

Namespaces

- package WebradioManager

7.47.1 Detailed Description

Implements the webradio server statistics class.

Definition in file WebradioServerStats.cs.

7.48 WebradioServerStats.cs

```
00001
00007 using System;
00008
00009 namespace WebradioManager
00010 {
00020     public class WebradioServerStats
00021     {
00022         #region Fields
00023         // \brief The current listeners count.
00024         private int _currentListeners;
00025         // \brief The unique listeners count.
00026         private int _uniqueListeners;
00027         // \brief The peak listeners count.
00028         private int _peakListeners;
00029         // \brief The average time listening.
00030         private TimeSpan _averageTime;
00031         #endregion
00032
00033         #region Properties
00034
00043         public TimeSpan AverageTime
00044         {
00045             get { return _averageTime; }
00046             set { _averageTime = value; }
00047         }
00048
00057         public int PeakListeners
00058         {
00059             get { return _peakListeners; }
00060             set { _peakListeners = value; }
00061         }
00062
00071         public int UniqueListeners
00072         {
00073             get { return _uniqueListeners; }
00074             set { _uniqueListeners = value; }
00075         }
00076
00085         public int CurrentListeners
00086         {
00087             get { return _currentListeners; }
00088             set { _currentListeners = value; }
00089         }
00090         #endregion
```

```
00091
00092         #region Methods
00093
00103         public WebradioServerStats()
00104         {
00105             this.CurrentListeners = 0;
00106             this.UniqueListeners = 0;
00107             this.PeakListeners = 0;
00108             this.AverageTime = new TimeSpan(0,0,0);
00109         }
00110         #endregion
00111     }
00112 }
```

7.49 WebradioTranscoder.cs File Reference

Implements the webradio transcoder class.

Classes

- class WebradioManager.WebradioTranscoder

  A webradio transcoder.

Namespaces

- package WebradioManager

7.49.1 Detailed Description

Implements the webradio transcoder class.

Definition in file WebradioTranscoder.cs.

7.50 WebradioTranscoder.cs

```
00001
00007 using System.Collections.Generic;
00008 using System.Collections.Specialized;
00009 using System.Diagnostics;
00010 using System.IO;
00011 using System.Net;
00012
00013 namespace WebradioManager
00014 {
00024     public abstract class WebradioTranscoder
00025     {
00026         #region Const
00027         // \brief The default identifier.
00028         const int DEFAULT_ID = 0;
00029         // \brief Filename of the transcoder executable file.
00030         const string SC_TRANS_FILENAME = "\\shoutcast\\sc_trans.exe";
00031         // \brief The default configuration extension.
00032         public const string DEFAULT_CONFIG_EXTENSION = ".config";
00033         // \brief The default log extension.
00034         public const string DEFAULT_LOG_EXTENSION = ".log";
00035         // \brief The default admin port.
00036         const int DEFAULT_ADMIN_PORT = 9000;
00037         // \brief The default admin login.
00038         public const string DEFAULT_ADMIN = "admin";
00039         // \brief The default admin password.
00040         public const string DEFAULT_ADMIN_PASSWORD = "admin";
00041         // \brief http://wiki.winamp.com/wiki/SHOUTcast_DNAS_Transcoder_2#Network_Options.
00042         const int PROTOCOL_VALUE = 3;
00043         #endregion
00044
00045         #region Fields
00046         // \brief The identifier.
00047         private int _id;
```

```
00048         // \brief The birate.
00049         private int _birate;
00050         // \brief The sample rate.
00051         private int _sampleRate;
00052         // \brief The name.
00053         private string _name;
00054         // \brief _URL of the document.
00055         private string _url;
00056         // \brief The IP.
00057         private IPAddress _ip;
00058         // \brief The port.
00059         private int _port;
00060         // \brief The admin port.
00061         private int _adminPort;
00062         // \brief The password.
00063         private string _password;
00064         // \brief Filename of the configuration file.
00065         private string _configFilename;
00066         // \brief Filename of the log file.
00067         private string _logFilename;
00068         // \brief The calendar file.
00069         private string _calendarFile;
00070         // \brief The current track's filename.
00071         private string _currentTrack;
00072         // \brief true to live capture.
00073         private bool _capture;
00074         // \brief Type of the stream.
00075         private StreamType _streamType;
00076         // \brief The process.
00077         private Process _process;
00078         // \brief The avaliable bitrates.
00079         private static int[] _avaliableBitrates = { 64000, 96000, 128000, 256000, 320000 };
00080         // \brief The avaliable sample rates.
00081         private static int[] _avaliableSampleRates = { 44100 };
00082         #endregion
00083
00084
00085         #region Properties
00086
00095         public bool Capture
00096         {
00097             get { return _capture; }
00098             set { _capture = value; }
00099         }
00100
00109         public string CurrentTrack
00110         {
00111             get { return _currentTrack; }
00112             set { _currentTrack = value; }
00113         }
00114
00123         public int AdminPort
00124         {
00125             get { return _adminPort; }
00126             set { _adminPort = value; }
00127         }
00128
00137         public string CalendarFile
00138         {
00139             get { return _calendarFile; }
00140             set { _calendarFile = value; }
00141         }
00142
00151         public Process Process
00152         {
00153             get { return _process; }
00154             set { _process = value; }
00155         }
00156
00165         public static int[] AvaliableSampleRates
00166         {
00167             get { return WebradioTranscoder._avaliableSampleRates; }
00168             set { WebradioTranscoder._avaliableSampleRates = value; }
00169         }
00170
00179         public static int[] AvaliableBitrates
00180         {
00181             get { return WebradioTranscoder._avaliableBitrates; }
00182             set { WebradioTranscoder._avaliableBitrates = value; }
00183         }
00184
00193         public int Id
00194         {
00195             get { return _id; }
00196             set { _id = value; }
00197         }
00198
```

```
00207         public int Birate
00208         {
00209             get { return _birate; }
00210             set { _birate = value; }
00211         }
00212 
00221         public int SampleRate
00222         {
00223             get { return _sampleRate; }
00224             set { _sampleRate = value; }
00225         }
00226 
00235         public string Name
00236         {
00237             get { return _name; }
00238             set { _name = value; }
00239         }
00240 
00249         public string Url
00250         {
00251             get { return _url; }
00252             set { _url = value; }
00253         }
00254 
00263         public IPAddress Ip
00264         {
00265             get { return _ip; }
00266             set { _ip = value; }
00267         }
00268 
00277         public int Port
00278         {
00279             get { return _port; }
00280             set { _port = value; }
00281         }
00282 
00291         public string Password
00292         {
00293             get { return _password; }
00294             set { _password = value; }
00295         }
00296 
00305         public string ConfigFilename
00306         {
00307             get { return _configFilename; }
00308             set { _configFilename = value; }
00309         }
00310 
00319         public string LogFilename
00320         {
00321             get { return _logFilename; }
00322             set { _logFilename = value; }
00323         }
00324 
00333         public StreamType StreamType
00334         {
00335             get { return _streamType; }
00336             set { _streamType = value; }
00337         }
00338         #endregion
00339 
00340         #region Methods
00341 
00363         public WebradioTranscoder(string name, int bitrate, int sampleRate, IPAddress ip,
       int port, int adminport, string url, string password, string configFilename, string logFilename,
      StreamType st)
00364             : this(DEFAULT_ID, name, bitrate, sampleRate, ip, port, adminport, url, password,
      configFilename, logFilename, st)
00365         {
00366             //NO CODE
00367         }
00368 
00391         public WebradioTranscoder(int id, string name, int bitrate, int sampleRate,
      IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename,
      StreamType st)
00392         {
00393             this.Id = id;
00394             this.Name = name;
00395             this.Birate = bitrate;
00396             this.SampleRate = sampleRate;
00397             this.Ip = ip;
00398             this.Port = port;
00399             this.AdminPort = adminport;
00400             this.Url = url;
00401             this.Password = password;
00402             this.LogFilename = logFilename;
00403             this.ConfigFilename = configFilename;
```

```
00404             this.StreamType = st;
00405             this.Capture = false;
00406             this.Process = new Process();
00407             this.CurrentTrack = "";
00408         }
00409 
00421         public void GenerateConfigFile(List<Playlist> playlists)
00422         {
00423             if (File.Exists(this.ConfigFilename))
00424                 File.Delete(this.ConfigFilename);
00425             string output = "";
00426             output += "logfile=" + Directory.GetCurrentDirectory() + "\\" + this.
      LogFilename.Replace('/', '\\') + "\n";
00427             output += "encoder_1=" + ((this.StreamType == WebradioManager.StreamType.AACPlus) ? "aacp" : "
      mp3") + "\n";
00428             output += "bitrate_1=" + (this.Birate * 1000) + "\n";
00429             output += "adminport=" + this.AdminPort + "\n";
00430             output += "adminuser=" + DEFAULT_ADMIN + "\n";
00431             output += "adminpassword=" + DEFAULT_ADMIN_PASSWORD + "\n";
00432             output += "capturedebug=1\n";
00433 
00434             output += "outprotocol_1=" + PROTOCOL_VALUE + "\n";
00435             output += "serverip_1=" + this.Ip + "\n";
00436             output += "serverport_1=" + this.Port + "\n";
00437             output += "password_1=" + this.Password + "\n";
00438             output += "streamid_1=1\n";
00439 
00440             output += "streamtitle=" + this.Name + "\n";
00441             output += "streamurl=" + this.Url + "\n";
00442             output += "genre=Misc\n";
00443 
00444             output += "calendarfile=" + Directory.GetCurrentDirectory() + "\\" + this.
      CalendarFile.Replace('/', '\\') + "\n";
00445             int index = 0;
00446             foreach (Playlist playlist in playlists)
00447             {
00448                 index++;
00449                 output += "playlistfilename_" + index.ToString() + "=" + playlist.
      Name + "\n";
00450                 output += "playlistfilepath_" + index.ToString() + "=" + Directory.GetCurrentDirectory() +
      "\\" + playlist.Filename.Replace("/", "\\") + "\n";
00451             }
00452 
00453             File.WriteAllText(this.ConfigFilename, output);
00454         }
00455 
00467         public bool IsRunning()
00468         {
00469             bool result = false;
00470             try
00471             {
00472                 foreach (Process prc in Process.GetProcesses())
00473                 {
00474                     if (prc.Id == this.Process.Id)
00475                     {
00476                         result = true;
00477                     }
00478                 }
00479                 if(this.Process.HasExited || !this.Process.Responding)
00480                     result = false;
00481                 return result;
00482             }
00483             catch
00484             {
00485                 return false;
00486             }
00487 
00488         }
00489 
00503         public bool Start(bool debug)
00504         {
00505 
00506             ProcessStartInfo StartInfo = new ProcessStartInfo(Directory.GetCurrentDirectory() +
      SC_TRANS_FILENAME)
00507               {
00508                   CreateNoWindow = true,
00509                   WindowStyle = (debug)?ProcessWindowStyle.Minimized:ProcessWindowStyle.Hidden,
00510                   Arguments = "\"" + Directory.GetCurrentDirectory() + "\\" + this.ConfigFilename.Replace(
      '/', '\\') + "\""
00511               };
00512             if (this.IsRunning())
00513                 this.Process.Kill();
00514             try
00515             {
00516                 this.Process = Process.Start(StartInfo);
00517                 return true;
00518             }
```

```
00519             catch
00520             {
00521                 return false;
00522             }
00523 
00524         }
00525 
00537         public bool Stop()
00538         {
00539 
00540             if (this.IsRunning() && this.Process.Responding)
00541             {
00542                 try
00543                 {
00544                     this.Process.Kill();
00545                 }
00546                 catch
00547                 {
00548                     return false;
00549                 }
00550             }
00551             return true;
00552 
00553         }
00554 
00567         public string GetStatus()
00568         {
00569             WebClient wb = new WebClient();
00570             var data = new NameValueCollection();
00571             data["op"] = "getstatus";
00572             data["seq"] = "45";
00573             wb.Credentials = new NetworkCredential(DEFAULT_ADMIN, DEFAULT_ADMIN_PASSWORD);
00574             try
00575             {
00576                 var response = wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data)
    ;
00577                 return System.Text.Encoding.UTF8.GetString(response);
00578             }
00579             catch
00580             {
00581                 return string.Empty;
00582             }
00583         }
00584 
00597         public void SetCaptureMode(bool active, string device)
00598         {
00599             this.Capture = active;
00600 
00601             WebClient wb = new WebClient();
00602             var data = new NameValueCollection();
00603             data["op"] = "setoptions";
00604             data["seq"] = "45";
00605             data["capturedevice"] = "Contrôleur audio haute définition";
00606             wb.Credentials = new NetworkCredential(DEFAULT_ADMIN, DEFAULT_ADMIN_PASSWORD);
00607             wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00608 
00609             data = new NameValueCollection();
00610             data["op"] = "capture";
00611             data["seq"] = "45";
00612             data["state"] = (active)?"on":"off";
00613             wb.Credentials = new NetworkCredential(WebradioTranscoder.DEFAULT_ADMIN,
    WebradioTranscoder.DEFAULT_ADMIN_PASSWORD);
00614             wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00615         }
00616 
00626         public void NextTrack()
00627         {
00628             WebClient wb = new WebClient();
00629             var data = new NameValueCollection();
00630             data["op"] = "nexttrack";
00631             data["seq"] = "45";
00632             wb.Credentials = new NetworkCredential(WebradioTranscoder.DEFAULT_ADMIN,
    WebradioTranscoder.DEFAULT_ADMIN_PASSWORD);
00633             wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00634         }
00635 
00649         public override string ToString()
00650         {
00651             return this.Name;
00652         }
00653 
00654         #endregion
00655     }
00656 }
```

7.51 WMModel.cs File Reference

Implements the wm model class.

Classes

- class WebradioManager.WMModel

  A data Model for the WebradioManager project.

Namespaces

- package WebradioManager

7.51.1 Detailed Description

Implements the wm model class.

Definition in file WMModel.cs.

7.52 WMModel.cs

```
00001
00007 using iTextSharp.text;
00008 using iTextSharp.text.pdf;
00009 using System;
00010 using System.Collections.Generic;
00011 using System.Diagnostics;
00012 using System.IO;
00013 using System.Net;
00014 using System.Threading;
00015 using System.Xml;
00016
00017 namespace WebradioManager
00018 {
00028     public class WMModel
00029     {
00030         #region Const
00031         //Default
00032         // \brief Defaults webradio's folder.
00033         public const string DEFAULT_WEBRADIOS_FOLDER = "webradios/";
00034         // \brief The default shoutcast folder.
00035         public const string DEFAULT_SHOUTCAST_FOLDER = "shoutcast/";
00036         // \brief The default logfilename.
00037         const string DEFAULT_LOGFILENAME = "log.txt";
00038         // \brief The default configfilename.
00039         const string DEFAULT_CONFIGFILENAME = "config.config";
00040         // \brief The default password.
00041         const string DEFAULT_PASSWORD = "1234";
00042         // \brief The maximum name length.
00043         const int MAX_NAME_LENGTH = 255;
00044
00045         //Server
00046         // \brief Default server folder
00047         const string DEFAULT_SERVER_FOLDER = "server/";
00048         // \brief The default server port.
00049         const int DEFAULT_SERVER_PORT = 8000;
00050         // \brief The default maximum listener.
00051         const int DEFAULT_MAX_LISTENER = 32;
00052         // \brief The default server password.
00053         const string DEFAULT_SERVER_PASSWORD = "1234";
00054
00055         //Calendar
00056         // \brief Default calendar's filename
00057         public const string DEFAULT_CALENDAR_FILENAME = "calendar.xml";
00058
00059         //Playlist
00060         // \brief Default playlist folder
00061         const string DEFAULT_PLAYLISTS_FOLDER = "playlists/";
00062         // \brief The maximum try generate.
00063         const int MAX_TRY_GENERATE = 10;
00064
```

```
00065         //Transcoder
00066         // \brief Default transcoders folder
00067         const string DEFAULT_TRANSCODERS_FOLDER = "transcoders/";
00068         #endregion
00069
00070         #region Fields
00071         // \brief The webradios list (id,webradio object).
00072         private Dictionary<int, Webradio> _webradios;
00073         // \brief The observers list.
00074         private List<IController> _observers;
00075         // \brief The bdd.
00076         private Bdd _bdd;
00077         // \brief The library.
00078         private List<AudioFile> _library;
00079         // \brief The process watcher.
00080         private System.Windows.Forms.Timer _processWatcher;
00081         // \brief The active transcoders list.
00082         private List<WebradioTranscoder> _activeTranscoders;
00083         // \brief The active servers list.
00084         private List<WebradioServer> _activeServers;
00085         #endregion
00086
00087         #region Properties
00088
00097         public List<WebradioServer> ActiveServers
00098         {
00099             get { return _activeServers; }
00100             set { _activeServers = value; }
00101         }
00102
00111         public List<WebradioTranscoder> ActiveTranscoders
00112         {
00113             get { return _activeTranscoders; }
00114             set { _activeTranscoders = value; }
00115         }
00116
00125         public System.Windows.Forms.Timer ProcessWatcher
00126         {
00127             get { return _processWatcher; }
00128             set { _processWatcher = value; }
00129         }
00130
00139         public Bdd Bdd
00140         {
00141             get { return _bdd; }
00142             set { _bdd = value; }
00143         }
00144
00153         public List<IController> Observers
00154         {
00155             get { return _observers; }
00156             set { _observers = value; }
00157         }
00158
00167         public Dictionary<int, Webradio> Webradios
00168         {
00169             get { return _webradios; }
00170             set { _webradios = value; }
00171         }
00172
00181         public List<AudioFile> Library
00182         {
00183             get { return _library; }
00184             set { _library = value; }
00185         }
00186         #endregion
00187
00188         #region Methods
00189
00199         public WMModel()
00200         {
00201             this.Webradios = new Dictionary<int, Webradio>();
00202             this.Observers = new List<IController>();
00203             this.Bdd = new Bdd();
00204             this.Library = new List<AudioFile>();
00205             this.ActiveTranscoders = new List<WebradioTranscoder>();
00206             this.ActiveServers = new List<WebradioServer>();
00207             this.ProcessWatcher = new System.Windows.Forms.Timer();
00208             this.ProcessWatcher.Tick += ProcessWatcher_Tick;
00209             this.ProcessWatcher.Interval = 1000;
00210             this.ProcessWatcher.Start();
00211         }
00212
00227         private string GetCurrentTrackFromXML(string xml)
00228         {
00229             if (!string.IsNullOrEmpty(xml))
00230             {
```

```
00231                 string currentTrack = "";
00232                 XmlDocument document = new XmlDocument();
00233                 document.LoadXml(xml);
00234                 XmlNodeList nodes = document.GetElementsByTagName("activesource");
00235                 foreach (XmlNode node in nodes[0].ChildNodes)
00236                 {
00237                     if (node.Name == "currenttrack")
00238                     {
00239                         currentTrack = node.InnerText;
00240                         break;
00241                     }
00242                 }
00243                 return currentTrack;
00244             }
00245             else
00246                 return xml;
00247         }
00248 
00262         void ProcessWatcher_Tick(object sender, EventArgs e)
00263         {
00264             bool needUpdate = false;
00265             for (int i = 0; i < this.ActiveTranscoders.Count; i++)
00266             {
00267                 if (!this.ActiveTranscoders[i].IsRunning())
00268                 {
00269                     this.ActiveTranscoders.RemoveAt(i);
00270                     needUpdate = true;
00271                     i--;
00272                 }
00273                 else
00274                 {
00275                     string currentTrack = this.GetCurrentTrackFromXML(this.ActiveTranscoders[i].GetStatus()
    );
00276                     if (currentTrack != this.ActiveTranscoders[i].CurrentTrack)
00277                     {
00278                         this.ActiveTranscoders[i].CurrentTrack = currentTrack;
00279                         needUpdate = true;
00280                         if (!string.IsNullOrEmpty(currentTrack.Trim()))
00281                             this.Bdd.AddToHistory(this.ActiveTranscoders[i].Id, DateTime.Now, currentTrack)
    ;
00282                     }
00283                 }
00284             }
00285             for (int i = 0; i < this.ActiveServers.Count; i++)
00286             {
00287                 if (!this.ActiveServers[i].IsRunning())
00288                 {
00289                     this.ActiveServers.RemoveAt(i);
00290                     needUpdate = true;
00291                     i--;
00292                 }
00293             }
00294             if (needUpdate)
00295                 this.UpdateObservers();
00296         }
00297 
00309         public void AddObserver(IController observer)
00310         {
00311             this.Observers.Add(observer);
00312         }
00313 
00325         public void RemoveObserver(IController observer)
00326         {
00327             this.Observers.Remove(observer);
00328         }
00329 
00343         public int GetSimiliarViewCount(int webradioId)
00344         {
00345             int ret = 0;
00346             foreach (IController controler in this.Observers)
00347             {
00348                 if (controler is AdminController)
00349                 {
00350                     if ((controler as AdminController).View.IdWebradio == webradioId)
00351                         ret++;
00352                 }
00353             }
00354             return ret;
00355         }
00356 
00368         private void UpdateObservers(int webradioId)
00369         {
00370             foreach (IController controller in this.Observers)
00371             {
00372                 if (controller is AdminController)
00373                 {
00374                     if ((controller as AdminController).View.IdWebradio == webradioId)
```

```
00375                         controller.UpdateView();
00376                 }
00377             }
00378         }
00379 
00389         private void UpdateObservers()
00390         {
00391             foreach (IController controller in this.Observers)
00392             {
00393                 controller.UpdateView();
00394             }
00395         }
00396 
00406         public void LoadLibrary()
00407         {
00408             this.Library = this.Bdd.LoadLibrary();
00409         }
00410 
00422         public void CheckFolders(int webradioId)
00423         {
00424             if (!Directory.Exists(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name))
00425                 Directory.CreateDirectory(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name);
00426         }
00427 
00437         public void LoadWebradios()
00438         {
00439             if (!Directory.Exists(DEFAULT_WEBRADIOS_FOLDER))
00440                 Directory.CreateDirectory(DEFAULT_WEBRADIOS_FOLDER);
00441             if (!Directory.Exists(DEFAULT_SHOUTCAST_FOLDER))
00442                 Directory.CreateDirectory(DEFAULT_SHOUTCAST_FOLDER);
00443             this.Webradios = this.Bdd.LoadWebradios();
00444         }
00445 
00459         public Webradio GetWebradio(int id)
00460         {
00461             return this.Webradios[id];
00462         }
00463 
00477         public Webradio GetWebradioByName(string name)
00478         {
00479             Webradio ret = null;
00480             foreach (KeyValuePair<int, Webradio> webradio in this.Webradios)
00481             {
00482                 if (webradio.Value.Name == name)
00483                 {
00484                     ret = webradio.Value;
00485                     break;
00486                 }
00487             }
00488             return ret;
00489         }
00490 
00502         public List<Webradio> GetWebradios()
00503         {
00504             //Get only webradios with its name and id and without useless stuffs for SelectionView
00505             List<Webradio> list = new List<Webradio>();
00506             foreach (KeyValuePair<int, Webradio> wr in this.Webradios)
00507             {
00508                 list.Add(new Webradio(wr.Value.Name, wr.Value.Id));
00509             }
00510             return list;
00511         }
00512 
00526         public bool CreateWebradio(string name)
00527         {
00528             if (this.IsValidPath(name))
00529             {
00530                 string webradioFilename = DEFAULT_WEBRADIOS_FOLDER + name + "/";
00531                 Webradio wr = new Webradio(name);
00532                 WebradioServer server = new WebradioServer(DEFAULT_SERVER_PORT,
00533                     webradioFilename + DEFAULT_SERVER_FOLDER + DEFAULT_LOGFILENAME,
00534                     webradioFilename + DEFAULT_SERVER_FOLDER + DEFAULT_CONFIGFILENAME,
    DEFAULT_SERVER_PASSWORD, WebradioServer.DEFAULT_ADMIN_LOGIN, DEFAULT_MAX_LISTENER);
00535                 wr.Server = server;
00536                 wr.Playlists = new List<Playlist>();
00537                 wr.Calendar = new WebradioCalendar(webradioFilename +
    DEFAULT_CALENDAR_FILENAME);
00538                 wr.Transcoders = new List<WebradioTranscoder>();
00539                 try
00540                 {
00541                     wr.Id = this.Bdd.AddWebradio(wr);
00542                     this.Webradios.Add(wr.Id, wr);
00543                     //Directory creation
00544                     Directory.CreateDirectory(webradioFilename);
00545                     Directory.CreateDirectory(webradioFilename + DEFAULT_SERVER_FOLDER);
00546                     Directory.CreateDirectory(webradioFilename + DEFAULT_PLAYLISTS_FOLDER);
00547                     Directory.CreateDirectory(webradioFilename + DEFAULT_TRANSCODERS_FOLDER);
```

```
00548                     Thread.Sleep(100);
00549                     wr.GenerateConfigFiles();
00550                 }
00551                 catch
00552                 {
00553                     return false;
00554                 }
00555                 return true;
00556             }
00557             else
00558                 return false;
00559         }
00560 
00574         public bool DeleteWebradio(int id)
00575         {
00576             bool output = false;
00577             output = this.Bdd.DeleteWebradio(id);
00578             Directory.Delete(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[id].Name, true);
00579             //Delete webradio from model
00580             try
00581             {
00582                 for (int i = 0; i < this.Observers.Count; i++)
00583                 {
00584                     AdminController ac = null;
00585                     if (this.Observers[i] is AdminController)
00586                     {
00587                         ac = this.Observers[i] as AdminController;
00588                         if (ac.View.IdWebradio == id)
00589                         {
00590                             ac.View.Close();
00591                             this.RemoveObserver(ac);
00592                             i--;
00593                         }
00594                     }
00595                 }
00596                 this.Webradios.Remove(id);
00597                 output = true;
00598             }
00599             catch
00600             {
00601                 output = false;
00602             }
00603 
00604             return output;
00605         }
00606 
00621         private Playlist GetPlaylistByName(string name, int webradioId)
00622         {
00623             Playlist playlist = null;
00624             foreach (Playlist p in this.Webradios[webradioId].Playlists)
00625             {
00626                 if (p.Name == name)
00627                 {
00628                     playlist = p;
00629                     break;
00630                 }
00631             }
00632             return playlist;
00633         }
00634 
00648         public bool DuplicateWebradio(int id)
00649         {
00650             Webradio webradio = this.Webradios[id];
00651             string name = "Copy of " + webradio.Name;
00652             if (this.IsValidPath(name))
00653             {
00654                 try
00655                 {
00656                     this.CreateWebradio(name);
00657                     Webradio newWebradio = this.GetWebradioByName(name);
00658                     //SERVER CONFIGURATION - Put server config to the clone
00659                     this.UpdateServer(true, webradio.Server.Port, webradio.Server.Password,
    webradio.Server.AdminPassword, webradio.Server.MaxListener, newWebradio.Id);
00660                     //PLAYLIST CONFIGURATION
00661                     foreach (Playlist playlist in webradio.Playlists)
00662                     {
00663                         Playlist newPlaylist;
00664                         if (this.CreatePlaylist(playlist.Name, newWebradio.Name,
    newWebradio.Id, playlist.Type, out newPlaylist))
00665                         {
00666                             newPlaylist.AudioFileList = new List<string>(playlist.AudioFileList);
00667                         }
00668                     }
00669                     //CALENDAR CONFIGURATION
00670                     foreach (CalendarEvent ev in webradio.Calendar.
    Events)
00671                     {
```

```
00672                           Playlist newPlaylistEvent = this.GetPlaylistByName(ev.Playlist.Name,
      newWebradio.Id);
00673                           if (newPlaylistEvent != null)
00674                           {
00675                               CalendarEvent newEvent = new
      CalendarEvent(ev.Name, ev.StartTime, ev.Duration, ev.
      Repeat, ev.Priority, ev.Shuffle, ev.Loopatend, newPlaylistEvent);
00676                               this.CreateEvent(newEvent, newWebradio.Id);
00677                           }
00678                       }
00679                       //TRANSCODER CONFIGURATION
00680                       foreach (WebradioTranscoder transcoder in webradio.
      Transcoders)
00681                       {
00682                           this.CreateTranscoder(transcoder.Name,
      transcoder.StreamType, transcoder.SampleRate, transcoder.Birate, transcoder.Url,
      transcoder.Ip, transcoder.Port, transcoder.AdminPort, transcoder.Password, newWebradio.Id);
00683                       }
00684                       this.UpdateObservers();
00685                       return true;
00686                   }
00687                   catch
00688                   {
00689                       return false;
00690                   }
00691               }
00692               else
00693                   return false;
00694
00695           }
00696
00708           public List<AudioFile> GetLibrary()
00709           {
00710               return this.Library;
00711           }
00712
00724           public List<string> GetGenders()
00725           {
00726               return this.Bdd.GetGenders();
00727           }
00728
00743           public bool ImportFilesToLibrary(string[] filenames,
      AudioType type)
00744           {
00745               AudioFile file;
00746               bool state = true;
00747               foreach (string filename in filenames)
00748               {
00749                   try
00750                   {
00751                       TagLib.File tagFile = TagLib.File.Create(filename);
00752                       if (type == AudioType.Music)
00753                           file = new Music(filename,
00754                               tagFile.Tag.Title,
00755                               tagFile.Tag.FirstPerformer,
00756                               tagFile.Tag.Album,
00757                               (int)tagFile.Tag.Year,
00758                               tagFile.Tag.Copyright,
00759                               tagFile.Properties.Duration,
00760                               tagFile.Tag.FirstGenre);
00761                       else
00762                           file = new Ad(filename,
00763                               tagFile.Tag.Title,
00764                               tagFile.Tag.FirstPerformer,
00765                               tagFile.Tag.Album,
00766                               (int)tagFile.Tag.Year,
00767                               tagFile.Tag.Copyright,
00768                               tagFile.Properties.Duration,
00769                               tagFile.Tag.FirstGenre);
00770
00771                       int id = this.Bdd.AddAudioFile(file);
00772                       if (id != Bdd.ERROR)
00773                       {
00774                           file.Id = id;
00775                           this.Library.Add(file);
00776                           this.UpdateObservers();
00777                       }
00778                       state = true;
00779                   }
00780                   catch
00781                   {
00782
00783                   }
00784               }
00785               return state;
00786           }
00787
```

```
00802         public bool DeleteAudioFile(int id, string audioFilename)
00803         {
00804             foreach (KeyValuePair<int, Webradio> webradio in this.Webradios)
00805             {
00806                 foreach (Playlist playlist in webradio.Value.Playlists)
00807                 {
00808                     for (int i = 0; i < playlist.AudioFileList.Count; i++)
00809                     {
00810                         if (playlist.AudioFileList[i] == audioFilename)
00811                         {
00812                             playlist.AudioFileList.Remove(playlist.AudioFileList[i]);
00813                             i--;
00814                         }
00815                     }
00816                 }
00817             }
00818             foreach (AudioFile file in this.Library)
00819             {
00820                 if (file.Id == id)
00821                 {
00822                     this.Library.Remove(file);
00823                     break;
00824                 }
00825             }
00826             return this.Bdd.DeleteAudioFile(id);
00827         }
00828 
00843         public bool UpdateAudioFile(AudioFile file)
00844         {
00845             if (this.Bdd.UpdateAudioFile(file))
00846             {
00847                 try
00848                 {
00849                     TagLib.File tagFile = TagLib.File.Create(file.Filename);
00850                     tagFile.Tag.Title = file.Title;
00851                     tagFile.Tag.Performers = new string[] { file.Artist };
00852                     tagFile.Tag.Album = file.Album;
00853                     tagFile.Tag.Year = (uint)file.Year;
00854                     tagFile.Tag.Copyright = file.Label;
00855                     tagFile.Tag.Genres = new string[] { file.Gender };
00856                     tagFile.Save();
00857 
00858                     foreach (AudioFile af in this.Library)
00859                     {
00860                         if (af.Id == file.Id)
00861                         {
00862                             af.Title = file.Title;
00863                             af.Artist = file.Artist;
00864                             af.Album = file.Album;
00865                             af.Year = file.Year;
00866                             af.Label = file.Label;
00867                             af.Gender = file.Gender;
00868                             break;
00869                         }
00870                     }
00871                     this.UpdateObservers();
00872                     return true;
00873                 }
00874                 catch
00875                 {
00876                     return false;
00877                 }
00878             }
00879             else
00880                 return false;
00881         }
00882 
00896         private bool IsValidFilename(string testName)
00897         {
00898             char[] invalidFileChars = Path.GetInvalidFileNameChars();
00899             if (testName.IndexOfAny(invalidFileChars) == -1)
00900                 return true;
00901             else
00902                 return false;
00903         }
00904 
00918         private bool IsValidPath(string testName)
00919         {
00920             char[] invalidFileChars = Path.GetInvalidPathChars();
00921             if (testName.IndexOfAny(invalidFileChars) == -1)
00922                 return true;
00923             else
00924                 return false;
00925         }
00926 
00944         public bool CreatePlaylist(string name, string webradioName, int webradioId,
      AudioType type, out Playlist newPlaylist)
```

```
00945         {
00946             Webradio selectedWebradio = this.Webradios[webradioId];
00947             int id = Bdd.ERROR;
00948             newPlaylist = null;
00949             if (this.IsValidFilename(name))
00950             {
00951                 string filename = DEFAULT_WEBRADIOS_FOLDER + webradioName + "/" + DEFAULT_PLAYLISTS_FOLDER
      + name + ".lst";
00952                 if (type == AudioType.Music)
00953                     newPlaylist = new PlaylistMusic(name, filename);
00954                 else
00955                     newPlaylist = new PlaylistAd(name, filename);
00956                 id = this.Bdd.CreatePlaylist(newPlaylist, webradioId);
00957                 if (id == Bdd.ERROR)
00958                     return false;
00959                 else
00960                 {
00961                     try
00962                     {
00963                         newPlaylist.Id = id;
00964                         newPlaylist.GenerateConfigFile();
00965                         selectedWebradio.Playlists.Add(newPlaylist);
00966                         this.UpdateObservers(webradioId);
00967                         return true;
00968                     }
00969                     catch
00970                     {
00971                         return false;
00972                     }
00973 
00974                 }
00975             }
00976             else
00977             {
00978                 return false;
00979             }
00980 
00981         }
00982 
00997         public bool DeletePlaylist(Playlist playlist, int webradioId)
00998         {
00999             if (this.Bdd.DeletePlaylist(playlist.Id))
01000             {
01001                 this.Webradios[webradioId].Playlists.Remove(playlist);
01002                 System.IO.File.Delete(playlist.Filename);
01003                 this.UpdateObservers(webradioId);
01004                 return true;
01005             }
01006             else
01007                 return false;
01008 
01009         }
01010 
01025         public bool AddToPlaylist(Playlist playlist, Dictionary<int, string>
      audioFiles)
01026         {
01027             bool state = true;
01028             foreach (KeyValuePair<int, string> audioFile in audioFiles)
01029             {
01030                 if (this.Bdd.AddToPlaylist(audioFile.Key, playlist.
      Id))
01031                 {
01032                     playlist.AudioFileList.Add(audioFile.Value);
01033                     state = true;
01034                 }
01035                 else
01036                 {
01037                     state = false;
01038                     break;
01039                 }
01040 
01041             }
01042             playlist.GenerateConfigFile();
01043             return state;
01044 
01045         }
01046 
01061         public bool RemoveFromPlaylist(Dictionary<int, string> audioFiles,
      Playlist playlist)
01062         {
01063             bool state = true;
01064             foreach (KeyValuePair<int, string> audioFile in audioFiles)
01065             {
01066                 if (this.Bdd.RemoveFromPlaylist(audioFile.Key, playlist.
      Id))
01067                 {
01068                     playlist.AudioFileList.Remove(audioFile.Value);
```

```
01069                   state = true;
01070               }
01071               else
01072               {
01073                   state = false;
01074                   break;
01075               }
01076
01077           }
01078           playlist.GenerateConfigFile();
01079           return state;
01080       }
01081
01095       public List<AudioFile> GetPlaylistContent(Playlist playlist)
01096       {
01097           List<AudioFile> audioFiles = new List<AudioFile>();
01098           foreach (string filename in playlist.AudioFileList)
01099           {
01100               foreach (AudioFile af in this.Library)
01101               {
01102                   if (af.Filename == filename)
01103                       audioFiles.Add(af);
01104               }
01105           }
01106           return audioFiles;
01107       }
01108
01127       public bool GeneratePlaylist(string name, TimeSpan duration,
      AudioType type, string gender, int webradioId, string webradioName)
01128       {
01129           Playlist newPlaylist;
01130           string filename = DEFAULT_WEBRADIOS_FOLDER + webradioName + "/" + DEFAULT_PLAYLISTS_FOLDER +
      name + ".lst";
01131           if (type == AudioType.Music)
01132               newPlaylist = new PlaylistMusic(name, filename);
01133           else
01134               newPlaylist = new PlaylistAd(name, filename);
01135           TimeSpan tmpDuration = new TimeSpan();
01136           Random random = new Random();
01137           AudioFile audioFile;
01138           int retries = 0;
01139           List<int> audioFilesId = new List<int>();
01140           while (tmpDuration <= duration)
01141           {
01142               audioFile = this.Library[random.Next(0, this.Library.Count - 1)];
01143               if (audioFile.Type == type)
01144               {
01145                   if (retries > MAX_TRY_GENERATE)
01146                       break;
01147                   if ((tmpDuration + audioFile.Duration) > duration)
01148                   {
01149                       retries++;
01150                       continue;
01151                   }
01152                   else
01153                       retries = 0;
01154
01155                   if ((audioFile.Type == AudioType.Ad) || (audioFile.Type == AudioType.Music
       && audioFile.Gender == gender))
01156                   {
01157                       tmpDuration += audioFile.Duration;
01158                       newPlaylist.AudioFileList.Add(audioFile.Filename);
01159                       audioFilesId.Add(audioFile.Id);
01160                   }
01161               }
01162           }
01163
01164           //Impossible to create a playlist
01165           if (newPlaylist.AudioFileList.Count == 0)
01166               return false;
01167
01168           //Add to database
01169           int idPlaylist = this.Bdd.AddGeneratedPlaylist(newPlaylist, audioFilesId, webradioId);
01170           if (idPlaylist == Bdd.ERROR)
01171               return false;
01172           newPlaylist.Id = idPlaylist;
01173           //Add to model
01174           this.Webradios[webradioId].Playlists.Add(newPlaylist);
01175           newPlaylist.GenerateConfigFile();
01176           this.UpdateObservers(webradioId);
01177           return true;
01178       }
01179
01194       public bool CreateEvent(CalendarEvent newEvent, int webradioId)
01195       {
01196           if (this.Bdd.EventExist(newEvent, this.Webradios[webradioId].Calendar.
      Id))
```

```
01197                 return false;
01198
01199             int id = this.Bdd.AddEvent(newEvent, this.Webradios[webradioId].Calendar.Id,
      newEvent.Playlist.Id);
01200             newEvent.Id = id;
01201             this.Webradios[webradioId].Calendar.Events.Add(newEvent);
01202             this.Webradios[webradioId].Calendar.GenerateConfigFile();
01203             this.UpdateObservers(webradioId);
01204             return true;
01205         }
01206
01221         public bool UpdateEvent(CalendarEvent aEvent, int webradioId)
01222         {
01223             if (this.Bdd.UpdateEvent(aEvent))
01224             {
01225                 foreach (CalendarEvent ce in this.Webradios[webradioId].Calendar.Events)
01226                 {
01227                     CalendarEvent tmp = ce;
01228                     if (ce.Id == aEvent.Id)
01229                     {
01230                         tmp.StartTime = aEvent.StartTime;
01231                         tmp.Duration = aEvent.Duration;
01232                         break;
01233                     }
01234                 }
01235                 this.Webradios[webradioId].Calendar.GenerateConfigFile();
01236                 this.UpdateObservers(webradioId);
01237                 return true;
01238             }
01239             else
01240                 return false;
01241         }
01242
01257         public bool DeleteEvent(CalendarEvent aEvent, int webradioId)
01258         {
01259             if (this.Bdd.DeleteEvent(aEvent))
01260             {
01261                 this.Webradios[webradioId].Calendar.Events.Remove(aEvent);
01262                 this.Webradios[webradioId].Calendar.GenerateConfigFile();
01263                 this.UpdateObservers(webradioId);
01264                 return true;
01265             }
01266             else
01267                 return false;
01268         }
01269
01292         public bool CreateTranscoder(string name, StreamType st, int sampleRate,
      int bitrate, string url, IPAddress ip, int port, int adminport, string password, int webradioId)
01293         {
01294             string filename = DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name + "/" +
      DEFAULT_TRANSCODERS_FOLDER;
01295             WebradioTranscoder transcoder;
01296             if (st == StreamType.AACPlus)
01297                 transcoder = new TranscoderAacPlus(name,
01298                 bitrate,
01299                 sampleRate,
01300                 ip,
01301                 port,
01302                 adminport,
01303                 url,
01304                 password,
01305                 filename,
01306                 filename);
01307             else
01308                 transcoder = new TranscoderMp3(name,
01309                  bitrate,
01310                  sampleRate,
01311                  ip,
01312                  port,
01313                  adminport,
01314                  url,
01315                  password,
01316                  filename,
01317                  filename);
01318
01319             transcoder.CalendarFile = DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name + "/" +
      DEFAULT_CALENDAR_FILENAME;
01320             int id = this.Bdd.AddTranscoder(transcoder, webradioId);
01321             if (id == Bdd.ERROR)
01322                 return false;
01323             transcoder.Id = id;
01324             transcoder.ConfigFilename = filename + id.ToString() + ".config";
01325             transcoder.LogFilename = filename + id.ToString() + ".log";
01326             this.Webradios[webradioId].Transcoders.Add(transcoder);
01327             transcoder.GenerateConfigFile(this.Webradios[webradioId].Playlists);
01328             this.Webradios[webradioId].GenerateConfigFiles();
01329             this.UpdateObservers();
```

```
01330             return true;
01331         }
01332
01347         public bool DeleteTranscoder(WebradioTranscoder transcoder, int
      webradioId)
01348         {
01349             if (transcoder.IsRunning())
01350                 transcoder.Stop();
01351
01352             if (this.Bdd.DeleteTranscoder(transcoder.Id))
01353             {
01354                 System.IO.File.Delete(transcoder.ConfigFilename);
01355                 if (System.IO.File.Exists(transcoder.LogFilename))
01356                     System.IO.File.Delete(transcoder.LogFilename);
01357                 this.Webradios[webradioId].Transcoders.Remove(transcoder);
01358                 this.UpdateObservers(webradioId);
01359                 return true;
01360             }
01361             else
01362                 return false;
01363         }
01364
01380         public bool UpdateTranscoder(WebradioTranscoder transcoder, bool
      debug, int webradioId)
01381         {
01382             try
01383             {
01384                 bool wasRunning = false;
01385                 if (transcoder.IsRunning())
01386                 {
01387                     wasRunning = true;
01388                     transcoder.Process.Kill();
01389                 }
01390
01391                 this.Bdd.UpdateTranscoder(transcoder);
01392                 transcoder.GenerateConfigFile(this.Webradios[webradioId].Playlists);
01393                 if (wasRunning)
01394                     transcoder.Start(debug);
01395                 this.UpdateObservers(webradioId);
01396                 return true;
01397             }
01398             catch
01399             {
01400                 return false;
01401             }
01402         }
01403
01419         public bool StartTranscoder(WebradioTranscoder transcoder, bool
      debug, int webradioId)
01420         {
01421             try
01422             {
01423                 if (transcoder.Start(debug))
01424                 {
01425                     this.ActiveTranscoders.Add(transcoder);
01426                     UpdateObservers(webradioId);
01427                     return true;
01428                 }
01429                 else
01430                     return false;
01431
01432             }
01433             catch
01434             {
01435                 return false;
01436             }
01437
01438         }
01439
01454         public bool StopTranscoder(WebradioTranscoder transcoder, int
      webradioId)
01455         {
01456             try
01457             {
01458                 if (transcoder.Stop())
01459                 {
01460                     this.ActiveTranscoders.Remove(transcoder);
01461                     this.Webradios[webradioId].Calendar.GenerateConfigFile();
01462                     this.UpdateObservers(webradioId);
01463                     return true;
01464                 }
01465                 else
01466                     return false;
01467             }
01468             catch
01469             {
01470                 return false;
```

```
01471             }
01472         }
01473
01487         public bool StopAllProcess(int webradioId)
01488         {
01489             try
01490             {
01491                 foreach (WebradioTranscoder transcoder in this.Webradios[webradioId].
      Transcoders)
01492                 {
01493                     transcoder.Stop();
01494                 }
01495                 this.Webradios[webradioId].Server.Stop();
01496                 return true;
01497             }
01498             catch
01499             {
01500                 return false;
01501             }
01502         }
01503
01515         public bool StopAllProcess()
01516         {
01517             try
01518             {
01519                 foreach (KeyValuePair<int, Webradio> webradio in this.Webradios)
01520                 {
01521                     foreach (WebradioTranscoder transcoder in webradio.Value.Transcoders)
01522                     {
01523                         transcoder.Stop();
01524                     }
01525                     webradio.Value.Server.Stop();
01526                 }
01527                 return true;
01528             }
01529             catch
01530             {
01531                 return false;
01532             }
01533         }
01534
01546         public void GenerateConfigFiles(int webradioId)
01547         {
01548             this.Webradios[webradioId].GenerateConfigFiles();
01549         }
01550
01569         public bool UpdateServer(bool debug, int port, string password, string adminPassword,
      int maxListener, int webradioId)
01570         {
01571             try
01572             {
01573                 WebradioServer server = this.Webradios[webradioId].Server;
01574                 bool wasRunning = false;
01575                 if (server.IsRunning())
01576                 {
01577                     wasRunning = true;
01578                     server.Process.Kill();
01579                 }
01580
01581                 if (!this.Bdd.UpdateServer(port, password, adminPassword, maxListener,
      webradioId))
01582                     return false;
01583
01584                 server.AdminPassword = adminPassword;
01585                 server.Password = password;
01586                 server.Port = port;
01587                 server.MaxListener = maxListener;
01588                 server.GenerateConfigFile();
01589                 if (wasRunning)
01590                     server.Start(debug);
01591
01592
01593                 this.UpdateObservers(webradioId);
01594                 return true;
01595             }
01596             catch
01597             {
01598                 return false;
01599             }
01600
01601         }
01602
01617         public bool StartServer(int webradioId, bool debug)
01618         {
01619             if (this.Webradios[webradioId].Server.Start(debug))
01620             {
01621                 this.ActiveServers.Add(this.Webradios[webradioId].Server);
```

```
01622                 this.UpdateObservers(webradioId);
01623                 return true;
01624             }
01625             else
01626                 return false;
01627         }
01628
01642         public bool StopServer(int webradioId)
01643         {
01644             if (this.Webradios[webradioId].Server.Stop())
01645             {
01646                 this.ActiveServers.Remove(this.Webradios[webradioId].Server);
01647                 this.UpdateObservers(webradioId);
01648                 return true;
01649             }
01650             else
01651                 return false;
01652         }
01653
01665         public void ShowServerWebInterface(int webradioId)
01666         {
01667             Process.Start(this.Webradios[webradioId].Server.WebInterfaceUrl);
01668         }
01669
01681         public void ShowServerWebAdmin(int webradioId)
01682         {
01683             Process.Start(this.Webradios[webradioId].Server.WebAdminUrl);
01684         }
01685
01699         public bool TranscoderNextTrack(WebradioTranscoder transcoder)
01700         {
01701             try
01702             {
01703                 transcoder.NextTrack();
01704                 return true;
01705             }
01706             catch
01707             {
01708                 return false;
01709             }
01710         }
01711
01725         public bool ClearHistory(int transcoderId)
01726         {
01727             return this.Bdd.ClearHistory(transcoderId);
01728         }
01729
01746         public bool GenerateHistory(int webradioId, string transcoderName, int transcoderId,
      string outputFilename)
01747         {
01748             Document document = new Document(PageSize.A4);
01749             PdfWriter.GetInstance(document, new FileStream(outputFilename, FileMode.Create));
01750             document.Open();
01751             iTextSharp.text.Font fontTitle = FontFactory.GetFont(FontFactory.HELVETICA, 20,
     iTextSharp.text.Font.NORMAL);
01752             iTextSharp.text.Font fontText = FontFactory.GetFont(FontFactory.HELVETICA, 12,
     iTextSharp.text.Font.NORMAL);
01753             iTextSharp.text.Font fontTextError = FontFactory.GetFont(FontFactory.HELVETICA, 12,
     BaseColor.RED);
01754             document.Add(new Paragraph(new Chunk("Webradio's name : " + this.Webradios[webradioId].Name,
     fontTitle)));
01755             document.Add(new Paragraph(new Chunk("Transcoder's name : " + transcoderName + "\n\n",
     fontTitle)));
01756             foreach (KeyValuePair<string, string> filename in this.Bdd.
     GetHistory(transcoderId))
01757             {
01758                 string line = "";
01759                 AudioFile file = this.GetAudioFileByFilename(filename.Value);
01760                 if (file != null)
01761                 {
01762                     line += "Title : " + file.Title + "\n";
01763                     line += "Artist : " + file.Artist + "\n";
01764                     line += "Album : " + file.Album + "\n";
01765                     line += "Date : " + filename.Key + "\n\n";
01766                     document.Add(new Paragraph(new Chunk(line, fontText)));
01767                 }
01768                 else
01769                 {
01770                     document.Add(new Paragraph(new Chunk("File not found", fontTextError)));
01771                 }
01772             }
01773
01774             document.Close();
01775             return true;
01776         }
01777
01791         public AudioFile GetAudioFileByFilename(string filename)
```

```
01792         {
01793             AudioFile result = null;
01794             foreach (AudioFile file in this.Library)
01795             {
01796                 if (file.Filename == filename)
01797                 {
01798                     result = file;
01799                     break;
01800                 }
01801             }
01802             return result;
01803         }
01804 
01819         public bool ModifyWebradioName(string newName, int webradioId)
01820         {
01821             Webradio selectedWebradio = this.Webradios[webradioId];
01822             this.StopAllProcess(webradioId);
01823             string oldName = selectedWebradio.Name;
01824             if (this.Bdd.ModifyWebradioName(newName, webradioId) &&
      this.Bdd.UpdateFilenames(selectedWebradio.Name, newName, selectedWebradio))
01825             {
01826                 foreach (WebradioTranscoder transcoder in selectedWebradio.
      Transcoders)
01827                 {
01828                     transcoder.ConfigFilename = transcoder.ConfigFilename.Replace(oldName, newName);
01829                     transcoder.LogFilename = transcoder.LogFilename.Replace(oldName, newName);
01830 
01831                 }
01832                 selectedWebradio.Server.ConfigFilename = selectedWebradio.Server.ConfigFilename.Replace(
      oldName, newName);
01833                 selectedWebradio.Server.LogFilename = selectedWebradio.Server.LogFilename.Replace(oldName,
      newName);
01834                 foreach (Playlist playlist in selectedWebradio.Playlists)
01835                 {
01836                     playlist.Filename = playlist.Filename.Replace(oldName, newName);
01837                 }
01838                 selectedWebradio.Calendar.Filename = selectedWebradio.Calendar.Filename.Replace(oldName,
      newName);
01839 
01840                 Directory.Move(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name,
      DEFAULT_WEBRADIOS_FOLDER + newName);
01841                 selectedWebradio.Name = newName;
01842                 selectedWebradio.GenerateConfigFiles();
01843                 this.UpdateObservers(webradioId);
01844                 return true;
01845             }
01846             else
01847                 return false;
01848         }
01849 
01866         public bool TranscoderCapture(bool active, string device,
      WebradioTranscoder transcoder, int webradioId)
01867         {
01868             try
01869             {
01870                 transcoder.SetCaptureMode(active, device);
01871                 this.UpdateObservers(webradioId);
01872                 return true;
01873             }
01874             catch
01875             {
01876                 return false;
01877             }
01878         }
01879 
01893         public List<WebradioListener> UpdateServerListeners(int webradioId)
01894         {
01895             return this.Webradios[webradioId].Server.GetListeners();
01896         }
01897 
01911         public bool UpdateServerStats(int webradioId)
01912         {
01913             WebradioServer server = this.Webradios[webradioId].Server;
01914             server.UpdateStats();
01915             this.UpdateObservers(webradioId);
01916             return true;
01917         }
01918 
01930         public bool CheckLibrary()
01931         {
01932             try
01933             {
01934                 for (int i = 0; i < this.Library.Count; i++)
01935                 {
01936                     AudioFile file = this.Library[i];
01937                     if (!System.IO.File.Exists(file.Filename))
01938                     {
```

```
01939                         this.Bdd.DeleteAudioFile(file.Id);
01940                         this.Library.Remove(file);
01941                         i--;
01942                     }
01943                 }
01944                 this.UpdateObservers();
01945                 return true;
01946             }
01947             catch
01948             {
01949                 return false;
01950             }
01951         }
01952         #endregion
01953
01954     }
01955 }
```

Index