

WebradioManager

Travail de diplôme 2014

Simon Menetrey – T.IN E2A – V1.2

28/05/2014



1 RÉSUMÉ

1.1 FRANÇAIS

Cette documentation décrit mon projet « WebradioManager » qui a pour but l’élaboration d’un logiciel C#/.NET permettant la gestion complète de multiples webradios. Le cahier des charges a été écrit en collaboration avec l’entreprise KTFM.

L’utilisateur peut gérer plusieurs webradios différentes à la fois. L’application contient une bibliothèque de musiques commune à toutes les webradios créées. Cette bibliothèque est remplie par l’utilisateur avec ses fichiers musicaux. Pour chacune des webradios, il est possible de créer des listes de lectures basées sur les musiques de la bibliothèque et gérer une grille horaire sur une semaine. Le but final est de générer un flux audio qui sera envoyé sur un serveur de diffusion (ce dernier distribuera le flux aux différents auditeurs, il peut être local ou distant). Chaque webradio peut créer différents flux pour différent serveur en se basant sur leur propre grille horaire et leurs propres listes de lecture. Enfin, chaque webradio dispose d’un serveur de diffusion local intégré.

Une base de données SQLite est utilisée pour sauvegarder les données du programme. La création de flux (transcodeur) et la création de serveurs de diffusion sont effectuées à l’aide d’outils nommés « ShoutCAST » qui sont développés par Nullsoft. Ces outils nécessitent des fichiers de configuration. Le programme s’occupe donc de générer ces fichiers à partir des données dont il dispose pour, ensuite, lancer les exécutables ShoutCAST et gérer leur fonctionnement.

1.2 ENGLISH

This documentation describes my project « WebradioManager » which aims to develop a C#/.NET software which allows a complete management of multiple webradios. The specifications were written in collaboration with the KTFM company.

The user can manage several different radios at once. The application contains a library common to all radios. This library is filled with music files by the user. For each of the webradios , it is possible to create playlists based on music library and manage a schedule over a week. The ultimate goal is to generate an audio stream to be sent to a streaming/broadcast server (the latter distributes the stream to different listeners it can be local or remote). Each webradio can create different streams for different servers based on their own schedule and their own playlists. Finally , each webradio has an integrated local broadcast server.

A SQLite database is used to store program data. The creation of stream (transcoder) and the creation of streaming server are performed using tools called « ShoutCAST » that are developed by Nullsoft. These tools require configuration files. The program generates these files from the data available (playlists, schedule and configurations) then run the ShoutCAST’s executables and manage their operations.

2 CONTENU

1	Résumé	1
1.1	Français.....	1
1.2	English.....	1
2	Contenu	2
3	Introduction.....	9
3.1	Mise en situation	9
3.2	Sujet.....	9
3.3	Qu'est-ce qu'une webradio ?	9
3.4	Pourquoi ce sujet ?	10
4	Cahier des charges.....	11
5	Analyse fonctionnelle	13
5.1	Principe.....	13
5.2	Schéma de l'application	13
5.3	Interface principale	14
5.3.1	Fenêtre d'administration.....	14
5.3.2	Menu principal	14
5.4	Gestion de plusieurs webradios	15
5.4.1	Création d'une webradio.....	16
5.4.2	Sélection d'une webradio.....	16
5.4.3	Fermeture d'une webradio	16
5.5	Serveur de diffusion interne et externe	17
5.6	Onglet « Status »	18
5.7	Gestion des musiques/pubs	19
5.7.1	Affichage.....	19
5.7.2	Importer et indexer	19
5.7.3	Ajout à une playlist	20

5.7.4	Modifier	20
5.7.5	Supprimer	20
5.8	Gestion des listes de lecture	21
5.8.1	Listes de lecture musicales	21
5.8.2	Listes de lecture publicitaires	21
5.8.3	Création	21
5.8.4	Génération automatique	21
5.8.5	Affichage du contenu d'une playlist	22
5.8.6	Retirer d'une playlist	22
5.8.7	Recherche	22
5.9	Gestion des horaires	23
5.9.1	Événement	23
5.9.2	Événement périodique	24
5.9.3	Rémplissage manuel	24
5.9.4	Modification d'un événement	24
5.9.5	Suppression d'un événement	25
5.10	Gestion des transcodeurs	26
5.10.1	Création	26
5.10.2	Affichage	27
5.10.3	Modification	27
5.10.4	Contrôles	27
5.10.5	Capture live	27
5.10.6	Historique de diffusion	28
5.11	Gestion du serveur de diffusion	28
5.11.1	Contrôles	28
5.11.2	Log	29
5.11.3	Configuration	29

5.11.4	Interface web.....	29
5.11.5	Administration web	29
6	Analyse organique	31
6.1	Environnement.....	31
6.2	Diagramme de classes	31
6.2.1	<i>Model</i>	31
6.2.2	MVC	31
6.2.3	Observateurs/Sujet	32
6.2.4	AudioType et StreamType	34
6.2.5	Stockage des webradios	35
6.3	Base de données.....	36
6.3.1	SQLite.....	36
6.3.2	Utilisation	36
6.3.3	Suppression en cascade.....	37
6.3.4	Schéma	38
6.3.5	Twebradio.....	39
6.3.6	Tserver	39
6.3.8	Tcalendar	40
6.3.9	Tcalendarevent.....	40
6.3.10	Tplaylist.....	41
6.3.11	Taudiotype.....	41
6.3.12	Tmusic.....	41
6.3.13	Tgender.....	42
6.3.14	Tplaylist_has_music.....	42
6.3.15	Thistory.....	43
6.3.16	Ttranscoder	43
6.4	Schéma de diffusion	45

6.4.1	Principe de base	45
6.4.2	Infomaniak.....	45
6.5	ShoutCast.....	46
6.5.1	Présentation	46
6.5.2	Pourquoi cet outil ?	46
6.5.3	Serveur	47
6.5.4	Transcoder.....	47
6.5.5	Schéma de fonctionnement résumé	48
6.6	Structures des dossiers/fichiers	49
6.6.1	Schéma	49
6.6.2	Exécutables Shoutcast.....	50
6.7	Initialisation de l'application	51
6.8	Webradio.....	53
6.8.1	Classes associées	53
6.8.2	Affichage des webradios disponibles	53
6.8.3	Création	53
6.8.4	Chargement	55
6.8.5	Duplication	56
6.8.6	Suppression	57
6.8.7	Changement de nom.....	58
6.8.8	Génération des configurations.....	58
6.9	Bibliothèque	60
6.9.1	Classes utilisées	60
6.9.2	MP3	60
6.9.3	Importation	60
6.9.4	Tags ID3	62
6.9.5	Analyse des tags	62

6.9.6	Indexation.....	63
6.9.7	Modification	64
6.9.8	Suppression	64
6.9.9	Vérification des données.....	65
6.9.10	Recherche.....	65
6.10	Listes de lecture.....	67
6.10.1	Classes utilisées	67
6.10.2	Génération de configuration	67
6.10.3	Création d'une playlist.....	68
6.10.4	Génération automatique.....	69
6.10.5	Ajout à une playlist	70
6.10.6	Retirer d'une playlist	71
6.10.7	Affichage du contenu	71
6.10.8	Suppression d'une playlist.....	72
6.11	Grille horaire.....	73
6.11.1	Classes utilisées	73
6.11.2	Outil utilisé	74
6.11.3	Gestion du calendrier par ShoutCAST	74
6.11.4	Génération de configuration	75
6.11.5	Affichage du calendrier	76
6.11.6	Sélection depuis le calendrier	78
6.11.7	Création d'un événement.....	78
6.11.8	Modification d'un événement.....	79
6.11.9	Suppression d'un événement.....	81
6.12	Transcoders	82
6.12.1	Classes utilisées	82
6.12.2	Outil utilisé	83

6.12.3	Définition des bitrates, taux d'échantillonnage et type d'encoder	84
6.12.4	Fichier de configuration et log.....	84
6.12.5	Licence MP3.....	87
6.12.6	Création d'un transcoder	87
6.12.7	Résolution de nom	88
6.12.8	Affichage d'un transcoder	88
6.12.9	Modification d'un transcoder.....	88
6.12.10	Suppression d'un transcoder.....	89
6.12.11	Exécution et fermeture	89
6.12.12	Administration web (weblet)	90
6.12.13	Capture live.....	90
6.12.14	Historique	91
6.13	Serveur de diffusion interne.....	95
6.13.1	Classes utilisées	95
6.13.2	Outil utilisé	96
6.13.3	Configuration.....	96
6.13.4	Mise à jour de la configuration	97
6.13.5	Exécution et fermeture	97
6.13.6	Statistiques et auditeurs.....	98
6.13.7	Affichage des interfaces web	100
6.14	Gestion des processus.....	101
6.14.1	IsRunning	101
6.14.2	Détection des crashes/fermetures externes.....	101
6.14.3	Fermeture automatique	102
7	Tests.....	103
7.1	Webradios	103
7.2	Status.....	103

7.3	Bibliothèque	103
7.4	Listes de lecture.....	104
7.5	Grille horaire.....	104
7.6	Transcoders	105
7.7	Serveur de diffusion	106
7.8	Processus.....	106
7.9	Autres	106
8	Conclusion	107
8.1	Bilan personnel.....	107
8.2	Améliorations possibles.....	108
9	Remerciements	108
10	Apports personnels	108
11	Glossaire	110
12	Illustrations.....	110
13	Références.....	112
14	Annexes	112

3 INTRODUCTION

3.1 MISE EN SITUATION

Je m'appelle Simon Menetrey, j'ai 20 ans et je suis actuellement en dernière année de formation technicien ES en informatique. J'ai effectué un CFC d'informaticien en 4 ans avant de commencer ma formation actuelle.

Cette documentation concerne mon travail de diplôme réalisé pour ma dernière année en tant que technicien ES en informatique. Ce travail a pour sujet la création d'un gestionnaire de webradio.

Mon professeur de diplôme, Monsieur Garcia, m'a mis en contact avec une entreprise (KTFM) qui recherche un programme de gestion pour leur webradio. En effet, actuellement, c'est un tiers qui s'occupe de la diffusion de leur webradio via des émissions préalablement enregistrées dans leurs studios. Dans l'état actuel, l'entreprise n'a pas un contrôle direct sur la diffusion de son contenu et elle désirerait pouvoir gérer elle-même l'intégralité de leur webradio. Ainsi, elle supprimera un intermédiaire et aura pleinement contrôle de la diffusion.

En plus de cela, KTFM a aussi besoin d'avoir une traçabilité des morceaux qu'elle diffuse avec des informations précises afin de faciliter le paiement des droits d'auteur à la Suisa¹.

J'ai donc réalisé le cahier des charges avec KTFM lors de deux rendez-vous afin de répondre au mieux à leurs besoins.



3.2 SUJET

Gestionnaire de webradio :

- Permettre de diffuser directement depuis le logiciel
- Gérer les morceaux à diffuser/listes de lecture
- Gérer les plages horaires
- Historique de diffusion

3.3 QU'EST-CE QU'UNE WEBRADIO ?

Une webradio est une radio diffusée sur internet via la technologie de lecture en continu. Cette technologie fournit ce que l'on appelle un « flux » que les auditeurs écoutent via leur lecteur multimédia préféré ou via un site web.

¹ SUISA est la coopérative des auteurs et éditeurs de musique <http://www.suisa.ch/fr/>

3.4 POURQUOI CE SUJET ?

Je suis passionné de musique. J'ai aussi toujours recherché un outil simple et gratuit pour gérer une webradio et sa diffusion. J'ai donc imaginé une application, tout-en-un, remplissant ce besoin. J'ai donc l'espoir que mon projet me sera autant utile à moi qu'à l'entreprise KTFM ainsi que de potentielles futures entreprises intéressées.

4 CAHIER DES CHARGES

Ce projet a pour but la création d'un gestionnaire de webradio (de type shoutCAST²) complet. Les principales fonctionnalités sont les suivantes :

- Possibilité de gérer plusieurs webradios indépendamment
- Gestion des playlists, horaires et pubs
 - Génération automatique de playlist
 - Génération au format XML
- Gestion des serveurs de diffusions distants et transcoder³ interne.
 - Diffusion de la webradio
- Indexation des fichiers musicaux (tags, chemin sur le disque dur)
- Historique des morceaux joués
 - Génération d'un compte-rendu afin de faciliter la gestion des droits d'auteurs
- Serveurs de diffusion interne (local)

Options :

- Si serveur de diffusion interne activé : Serveur WEB interne contenant un mini-site
- modifier les informations des musiques indexées.

Plus de détails dans le résumé (1).

² Voir le chapitre concernant [shoutCAST](#) dans l'analyse organique.

³ Permet de créer un flux audio vers un serveur de diffusions

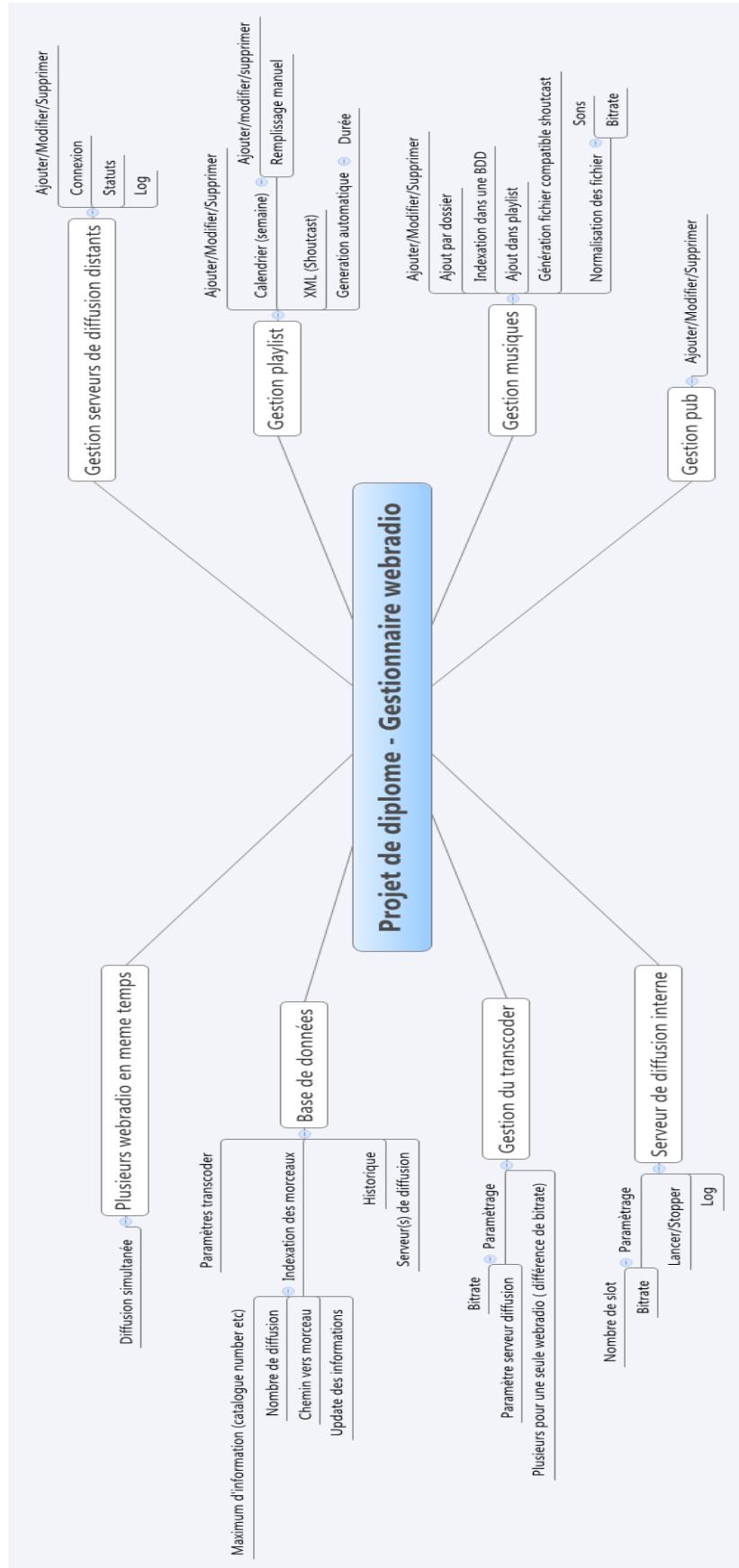


Figure 1 - Mindmap (discussion avec KTMAnalyses)

5 ANALYSE FONCTIONNELLE

5.1 PRINCIPE

Pour le principe de base d'une webradio, rendez-vous au chapitre concernant [ce sujet](#) dans l'analyse organique.

L'application a pour but de créer un flux audio qui sera récupéré par un serveur de diffusion et ce dernier s'occupera de diffuser le flux aux différents auditeurs. Un schéma sous forme d'une affiche est disponible en [annexe](#).

5.2 SCHÉMA DE L'APPLICATION

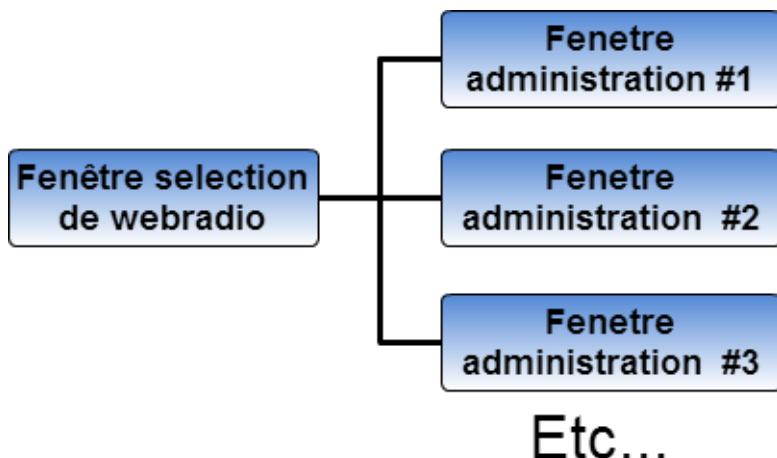


Figure 2 - Schéma application

Lors du lancement de l'application, la fenêtre de sélection de webradios s'affiche. L'utilisateur choisit la webradio qu'il veut gérer, puis une fenêtre d'administration s'ouvre avec les informations de la webradio sélectionnée. Il peut à tout moment réafficher la fenêtre de sélection et ouvrir une nouvelle fenêtre d'administration avec une autre webradio à gérer.

La diffusion de webradio est possible via des serveurs distants ou alors via le serveur interne (local) à l'application. Il y en a 1 par webradio.

5.3 INTERFACE PRINCIPALE

5.3.1 FENÊTRE D'ADMINISTRATION



Figure 3 - Interface principale AdminView

Cette fenêtre est l'interface principale de l'application. Elle est associée à une webradio. Comme montré dans le schéma de l'application (Figure 2), il est possible d'avoir plusieurs fenêtres d'administration ouvertes simultanément, une pour chaque webradio lancée. Les différents onglets sont décrits dans les chapitres suivants.

5.3.2 MENU PRINCIPAL

- File
 - Generate all configs : Génère la configuration de chaque élément de la webradio
 - Check library : Vérifie chaque fichier de la bibliothèque (si le fichier n'existe plus à l'emplacement spécifié, son enregistrement est supprimé).
- About : Affiche les informations sur le logiciel et son auteur

5.4 GESTION DE PLUSIEURS WEBRADIOS

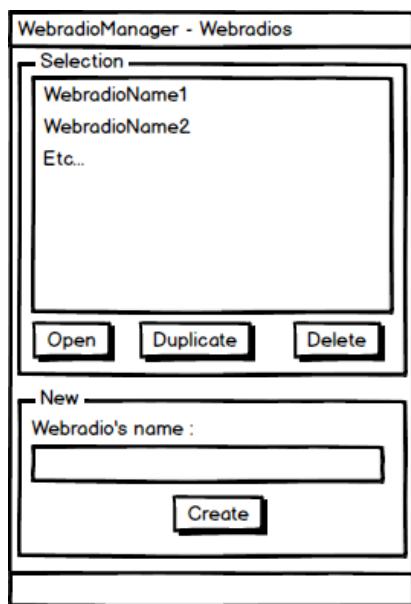


Figure 4 - Interface gestion des webradios SelectionView

La fenêtre de gestion des webradios s'affiche au démarrage de l'application pour sélectionner ou créer une webradio. Lorsque l'utilisateur clique sur « Open », la fenêtre principale s'ouvre avec les données de la webradio sélectionnée. Il peut aussi utiliser un double clic pour ouvrir une webradio.

Le bouton « Duplicate » crée une copie exacte de la webradio sélectionnée avec un préfix « copy of » à son nom. Enfin, le bouton « Delete » supprime la webradio sélectionnée.

La fenêtre n'est pas redimensionnable et ne peut être maximisée.

Concernant la création, la partie inférieure de la fenêtre propose la création d'une nouvelle webradio. L'utilisateur lui donne un nom, puis clique sur le bouton « Create ». Attention : Les webradios doivent avoir des noms différents. Un message d'erreur notifie l'utilisateur si une webradio a déjà le même nom.

L'utilisateur peut gérer autant de webradios en même temps qu'il le souhaite, en effet, à tout moment, il peut ouvrir la fenêtre de gestion des webradios et en sélectionner une autre. Cela ouvrira une nouvelle fenêtre principale, sans pour autant fermer les autres déjà ouvertes, avec les informations de la webradio fraîchement sélectionnée.

Chaque webradio possède ses propres listes de lecture, sa propre grille horaire ainsi que ses propres transcodeurs. Un transcodeur est un outil qui permet d'envoyer un flux musical à un serveur de diffusion. Pour plus d'informations, rendez-vous au chapitre concernant la [diffusion](#).

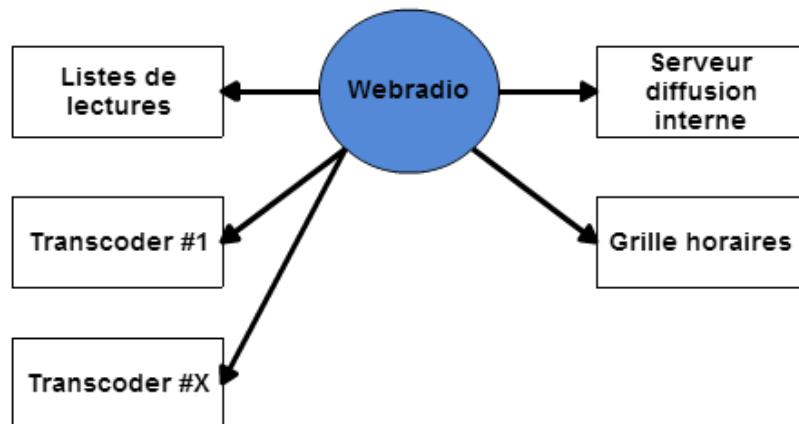
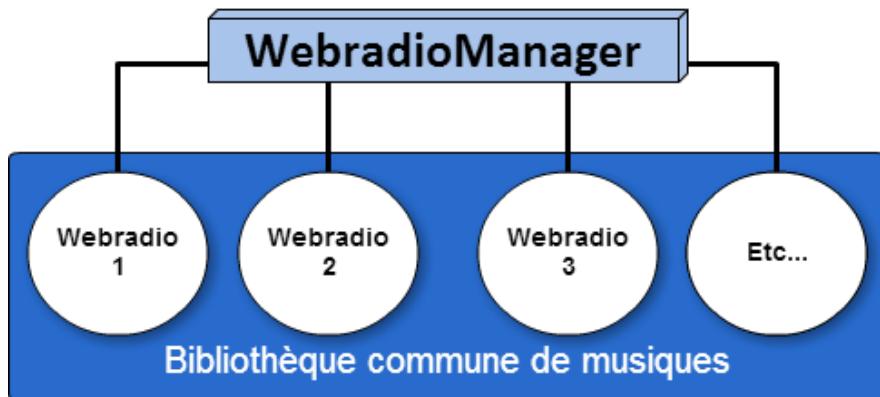


Figure 5 - Schéma webradios

5.4.1 CRÉATION D'UNE WEBRADIO

L'utilisateur entre le nom de la future webradio dans le champ correspondant, puis clique sur « create » pour créer la webradio et ouvrir la fenêtre d'administration liée à la nouvelle webradio fraîchement créée. Le nom d'une webradio ne peut pas dépasser 255 caractères.

5.4.2 SÉLECTION D'UNE WEBRADIO

L'utilisateur sélectionne une webradio parmi la liste proposée, puis clique sur « open » pour ouvrir une fenêtre d'administration liée à la webradio sélectionnée.

5.4.3 FERMETURE D'UNE WEBRADIO

Lors de la fermeture d'une fenêtre d'administration de webradio, l'utilisateur doit confirmer s'il est sûr de vouloir fermer la fenêtre en sachant que cela arrêtera le serveur de diffusion et les transcodeurs. Seule exception, si une autre *vue* qui pointe sur la même webradio est aussi affichée, l'utilisateur n'a pas besoin de confirmer, car les transcodeurs ne seront pas arrêtés dans ce cas. Ils sont seulement interrompus si la fenêtre fermée est la seule pointant sur la webradio en question.

5.5 SERVEUR DE DIFFUSION INTERNE ET EXTERNE

Il faut bien différencier les serveurs de diffusion qui sont externes et ceux internes. Les externes sont des serveurs hébergés par un provider⁴ (exemple : infomaniak⁵ pour KTFM). L'application va donc se servir de ces serveurs pour diffuser la webradio.

Les serveurs internes sont des serveurs de diffusion hébergés dans l'application elle-même (en local sur l'ordinateur).

⁴ Fournisseur de services internet

⁵ <http://www.infomaniak.com/>

5.6 ONGLET « STATUS »

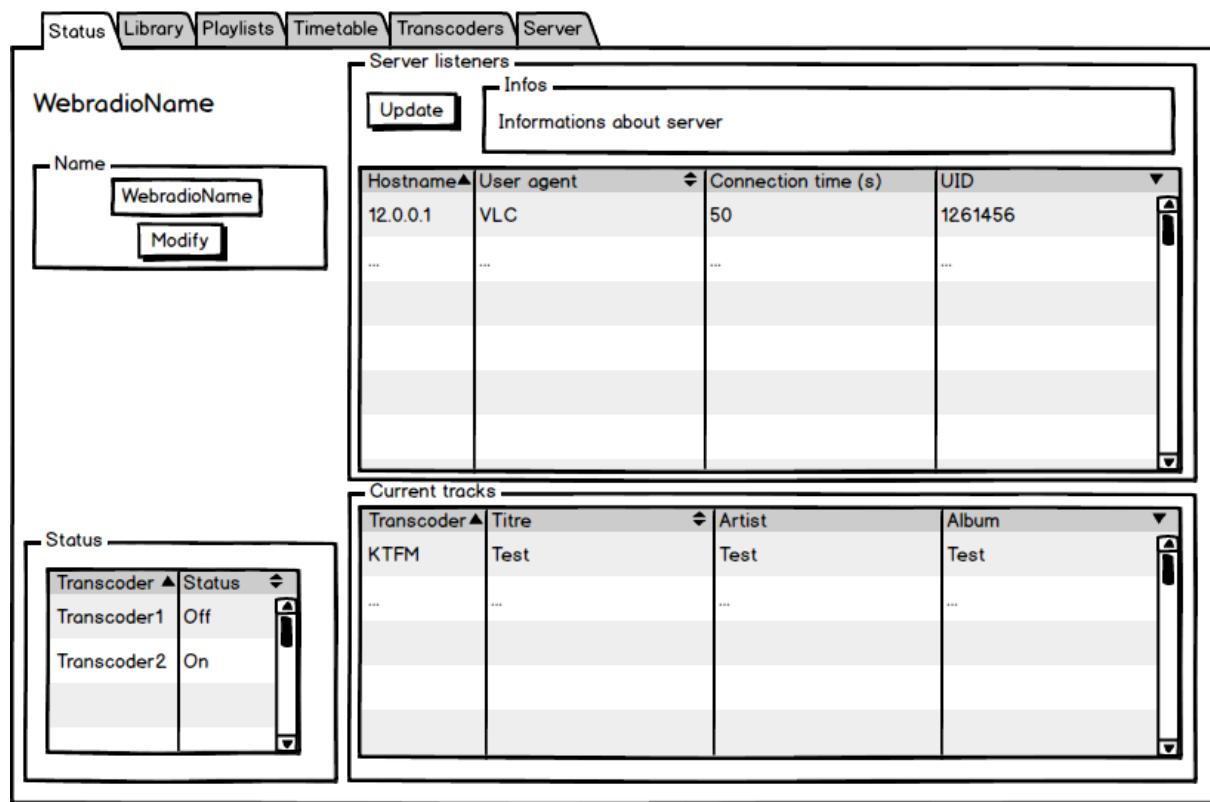


Figure 6 - Onglet "status"

L'onglet de statut est la page d'accueil de la webradio. L'utilisateur peut y modifier le nom de la webradio via le champ en dessous du nom. Si l'utilisateur change le nom, tous les transcodeurs et le serveur seront éteints.

Un tableau affiche l'état (allumé ou éteint) des différents transcodeurs de la webradio actuelle.

Dans la partie de droite, 2 tableaux :

- « Server listeners » : Affiche la liste des auditeurs/clients connectés au serveur local. Au-dessus, les informations concernant le nombre de connectés, le nombre maximum de connectés et le temps moyen de connexion sont affichés.
- « Current tracks » : Affiche le morceau actuellement joué par chaque transcodeur en fonctionnement.

5.7 GESTION DES MUSIQUES/PUBS

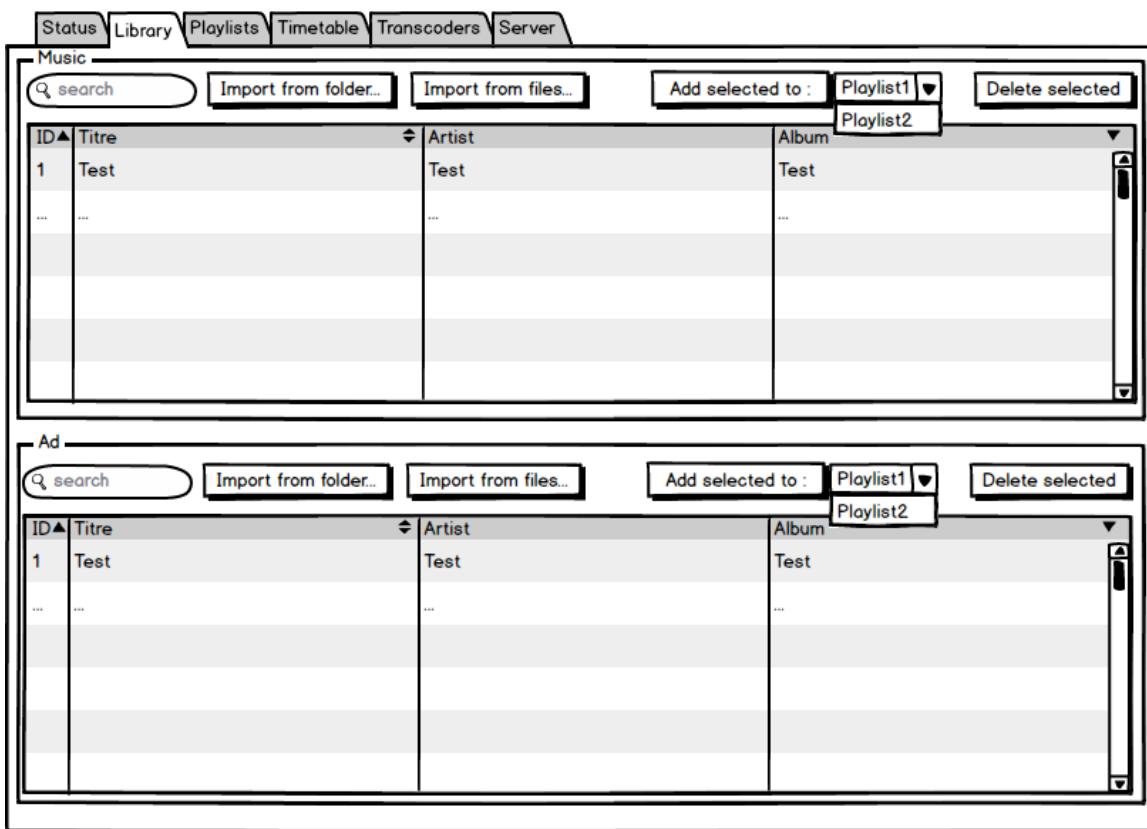


Figure 7 - Bibliothèque de musiques

Les musiques sont indexées pour former une bibliothèque commune dans l'application. Elle est commune, car elle est accessible depuis n'importe quelle webradio créée. L'interface est doublée : une partie pour les musiques et une autre pour les pubs. Elles sont identiques.

5.7.1 AFFICHAGE

La partie principale de cette interface est l'affichage du contenu de la bibliothèque dans la partie inférieure. Il est affiché sous la forme d'un tableau. Il est possible de sélectionner un ou plusieurs éléments.

5.7.2 IMPORTER ET INDEXER

L'utilisateur peut importer des fichiers musicaux dans sa bibliothèque (à l'heure actuelle, seuls les fichiers MP3 sont importables). Le bouton « Import from folder... » ouvre une boîte de dialogue où l'utilisateur peut sélectionner le dossier à analyser afin d'importer les fichiers musicaux qui y sont présents. C'est la partie dite « d'indexation ». Cette indexation peut être récursive, c'est-à-dire que les sous-dossiers du dossier sélectionné vont être analysés aussi. Une boîte de dialogue demande donc à l'utilisateur s'il veut effectuer une analyse récursive.

Le bouton « Import from files... » permet d'importer un ou plusieurs fichiers sélectionnés manuellement. La sélection s'effectue via une boîte de dialogue Windows standard.

À gauche de ces 2 boutons, une barre de recherche permet d'effectuer une recherche dans la bibliothèque de l'application.

Le bouton de droite « Delete selected » va supprimer les éléments sélectionnés dans la liste de morceaux affichés en dessous.

5.7.3 AJOUT À UNE PLAYLIST

Pour ajouter des morceaux à une playlist, l'utilisateur peut en sélectionner autant qu'il le souhaite dans la liste d'affichage puis sélectionner une playlist via le menu déroulant situé à droite du bouton « Add selected to ». Ce dernier permet donc de confirmer l'ajout à une playlist.

5.7.4 MODIFIER

L'utilisateur peut modifier les informations enregistrées dans la bibliothèque et, automatiquement, ces informations seront écrites dans le fichier MP3 lié. Les champs « id », « duration » et « path » ne sont pas modifiables. Pour modifier un champ (entrer en mode modification), l'utilisateur a plusieurs possibilités :

- Cliquer sur la cellule du champ qu'il désire modifier, puis commencer à écrire les modifications.
- Sélectionner une ligne, puis double cliquer sur la cellule à modifier.
- Sélectionner une cellule et appuyer sur la touche F2.

Dans tous les cas, l'utilisateur doit appuyer sur la touche « enter » pour valider.

Si, par exemple, le fichier musical n'existe plus à l'endroit indiqué, l'utilisateur en est averti.

5.7.5 SUPPRIMER

L'utilisateur a la possibilité de supprimer une ou plusieurs musiques/pubs d'un seul coup grâce à la sélection multiple.

5.8 GESTION DES LISTES DE LECTURE

La gestion des différentes listes de lecture se fait dans l'onglet « playlists ».

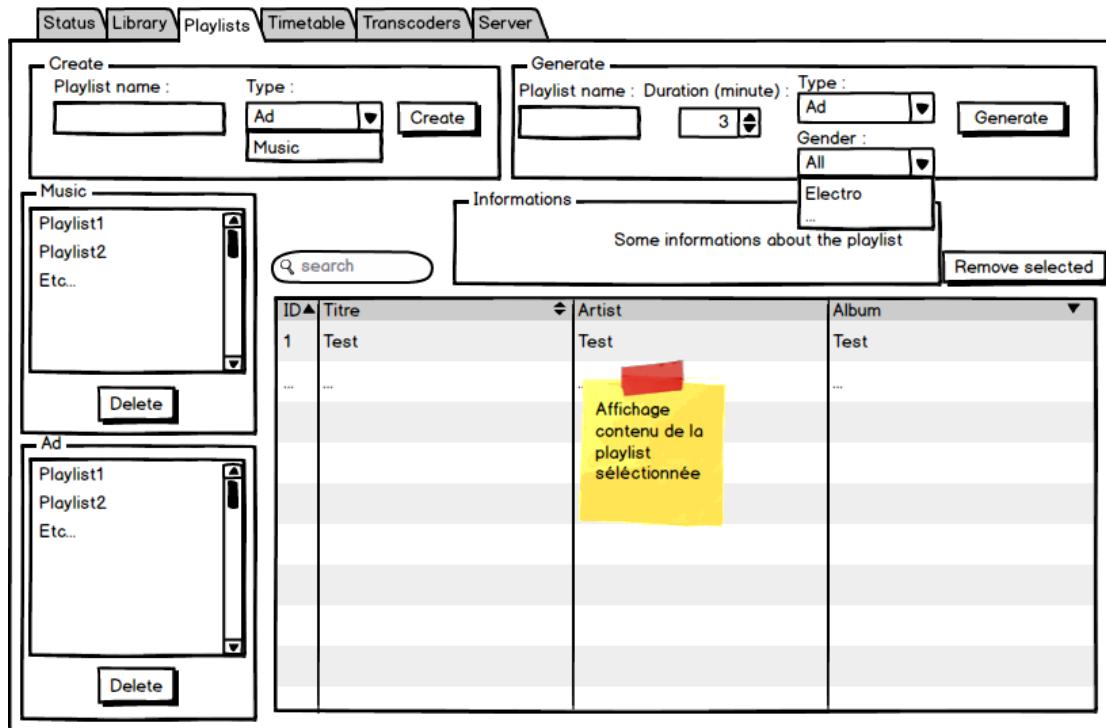


Figure 8 - Onglet "playlists"

5.8.1 LISTES DE LECTURE MUSICALES

Une liste de lecture musicale contient exclusivement des musiques. Son type est « music ».

5.8.2 LISTES DE LECTURE PUBLICITAIRES

Une liste de lecture publicitaire contient des pubs, des annonces ou des jingles. Son type est « ad ».

5.8.3 CRÉATION

La création d'une playlist se fait via le cadre situé en haut à gauche. Il lui faut un nom et un type. La playlist fraîchement créée est ensuite ajoutée à l'une des 2 listes situées à gauche en fonction de son type. L'utilisateur ne peut pas créer de playlist avec le même nom par type de playlist (« Music » ou « Ad ») et par webradio.

5.8.4 GÉNÉRATION AUTOMATIQUE

L'utilisateur a la possibilité de générer automatiquement une playlist en définissant une durée (minimum 1 minute et maximum 500 minutes), un nom, un genre musical et un type. Si le type « Ad » est choisi, le menu déroulant « Gender » est désactivé (une pub ne peut pas avoir de genre musical).

La liste de genre affiche les genres disponibles dans la bibliothèque de l'application. Le logiciel va prendre différentes musiques ou publicité afin de remplir le temps voulu. L'algorithme fait en sorte de s'approcher le plus possible de la durée demandée, mais plus celle-ci est longue, plus cela sera possible d'être précis.

5.8.5 AFFICHAGE DU CONTENU D'UNE PLAYLIST

Quand une playlist est sélectionnée (musique ou pub), son contenu est affiché dans la partie centrale de la *vue*. Les informations des morceaux sont affichées de même manière que pour l'onglet « [Library](#) ». Au-dessus de cette partie, des informations concernant la playlist sont affichées telles que le temps total de lecture, le nombre de morceaux présents, etc.

5.8.6 RETIRER D'UNE PLAYLIST

L'utilisateur peut sélectionner un ou plusieurs morceaux dans la partie affichant le contenu de la playlist afin de les retirer de cette dernière. Une fois la sélection faite, le bouton « Remove selected » supprime le/les morceau(x) et le nouveau contenu de la playlist est affiché.

5.8.7 RECHERCHE

L'utilisateur a la possibilité de recherche parmi les morceaux d'une playlist sélectionnée via le champ de recherche au-dessus de l'affichage du contenu de la playlist.

5.9 GESTION DES HORAIRES

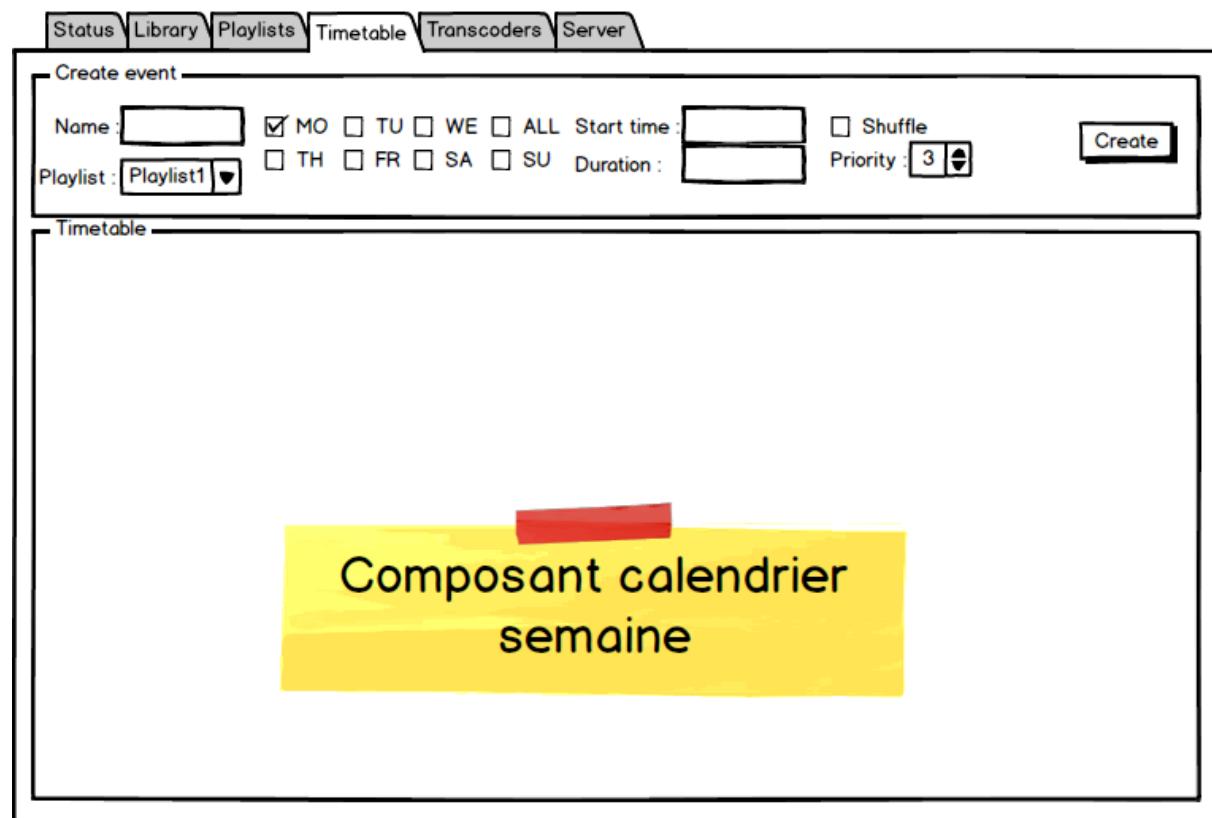


Figure 9 - Onglet "timetable"

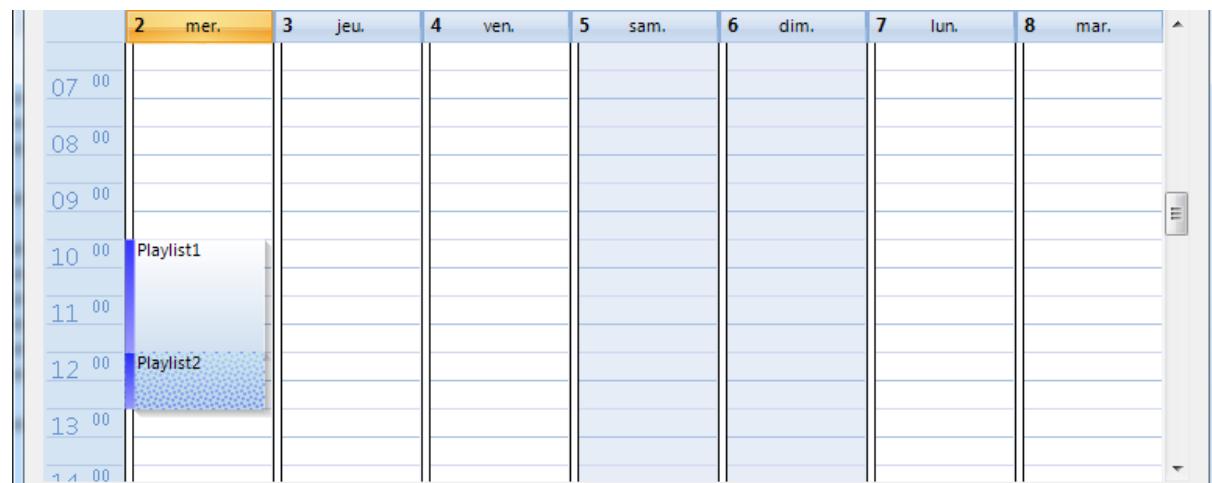


Figure 10 - Calendrier

5.9.1 ÉVÉNEMENT

Un événement est un élément du calendrier. Il est composé des informations suivantes :

- Un nom
- Une playlist (nom de la playlist qui sera jouée à l'événement)
- Un ou plusieurs jours où la playlist sera jouée

- Une heure de début (identique pour chaque jour sélectionné)
- Une durée de lecture (la playlist est jouée en boucle pendant le temps défini)
- Une option pour jouer la playlist en mode aléatoire ou non
- Une priorité (dans le cas où plusieurs événements se superposent, celui avec la plus grande priorité sera joué). De 0 à 100.

5.9.2 ÉVÉNEMENT PÉRIODIQUE

Un événement périodique se produit à intervalles réguliers.

Cette fonctionnalité n'est à l'heure actuellement pas encore implémentée.

http://wiki.winamp.com/wiki/SHOUTcast_Calendar_Event_XML_File_Specification#Time_Periodic_Events

5.9.3 REMPLISSAGE MANUEL

L'utilisateur peut utiliser le formulaire de création d'événements en le remplissant à la main ou alors il a la possibilité de sélectionner sur le calendrier, la plage horaire où il désire créer un événement.

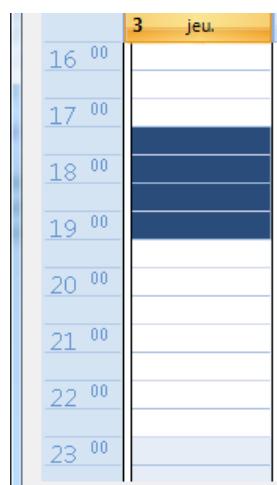


Figure 11 - Sélection multiple calendrier

La partie bleu foncé est la sélection faite par l'utilisateur. Dans ce cas, l'heure de début sera 17h30 et sa durée sera 2h, car la sélection finit à 19h30. Ces informations sont automatiquement affichées dans le formulaire de création. Évidemment, le reste des informations demandées sera à remplir à la main par l'utilisateur.

L'utilisateur peut aussi remplir manuellement les informations concernant le début et la durée d'un événement. La case à cocher « All » permet de cocher tous les jours de la semaine ou au contraire de les décocher. La case « shuffle » permet de choisir si l'événement lit la liste de lecture de façon aléatoire. « Priority » définit le niveau de priorité de l'événement par rapport à un autre qui serait prévu au même moment.

5.9.4 MODIFICATION D'UN ÉVÉNEMENT

	1 lundi	2 mardi	3 mercredi	4 jeudi	5 vendredi	6 samedi	7 dimanche
...							
06 00							
07 00							
08 00	Événement					Élément	
09 00	EuphoricTime(0) Euphoric Shuffle : False						
10 00							
11 00							
12 00							
13 00							

Figure 12 - Définition d'un événement avec des éléments

L'utilisateur peut modifier le début, la durée et les jours d'un événement. Pour se faire, il peut déplacer les « zones » (éléments) d'un événement pour le placer où bon lui semble. Il existe malgré cela des règles :

- Il ne peut pas y avoir plus d'un élément d'un événement dans le même jour.
- Tous les éléments commencent et finissent à la même heure. Si un élément est changé dans sa durée, cela sera appliqué à tous les autres éléments de cet événement.

L'utilisateur doit redémarrer ses transcodeurs pour que la nouvelle configuration soit prise en compte. Le programme en cours est donc interrompu et reprendra selon le nouveau calendrier après redémarrage.

5.9.5 SUPPRESSION D'UN ÉVÉNEMENT

Pour supprimer un événement, l'utilisateur doit faire clic droit sur l'événement en question. Une boîte de dialogue lui demande s'il est sûr de vouloir supprimer cet événement. La suppression d'un élément de l'événement entraîne la suppression complète de ce dernier.

5.10 GESTION DES TRANSCODEURS

Figure 13 - Onglet "Transcoders"

Cet onglet permet de gérer les différents transcodeurs d'une webradio. Ces derniers servent à diffuser le flux audio vers un serveur de diffusion. Ils utilisent les playlists et le calendrier configuré par l'utilisateur.

5.10.1 CRÉATION

La partie de gauche offre la possibilité de créer un nouveau transcodeur avec ses différents réglages. Pour des mesures de simplicité, le minimum requis est demandé à l'utilisateur. Les informations suivantes sont demandées :

- Le type de flux (mp3 ou ACC+)
- Le bitrate⁶ du flux
- Le sample rate⁷
- Le nom du flux (qui sert de nom pour le transcodeur)
- L'URL du flux (par exemple : site web de la webradio) : pas obligatoire
- Un port d'administration

⁶ Débit binaire : une mesure de la quantité de données numériques transmises par unité de temps.

⁷ Taux d'échantillonnage

- Adresse IP du serveur (ou le nom dns via bouton « resolve »)
- Port du serveur
- Mot de passe du serveur

Le bouton « resolve » permet à l'utilisateur d'entrer le nom de domaine du serveur puis de cliquer sur ce bouton pour trouver l'adresse IP liée. Si elle est trouvée, elle sera affichée dans le champ à la place du nom DNS.

5.10.2 AFFICHAGE

Dans la partie droite, la liste des transcodeurs de la webradio est affichée. L'utilisateur peut cliquer et sélectionner un des transcodeurs afin de la supprimer à l'aide du bouton « Delete ». Les informations du transcodeur sélectionné sont affichées à droite de la liste, dans les champs prévus à cet effet. Le statut (on ou off) est affiché en dessous de ces dernières. Tout en bas, le log du transcodeur est affiché et peut être effacé à l'aide du bouton « Clear ».

5.10.3 MODIFICATION

Les informations affichées dans les champs à droite peuvent être modifiées puis enregistrées (bouton « Update ») afin de mettre à jour les paramètres du transcodeur sélectionné.

5.10.4 CONTRÔLES

La partie « status » permet de contrôler le transcodeur sélectionné. La case à cocher « Debug » permet de lancer le serveur avec son log dans une fenêtre console.

Si une erreur est détectée lors du lancement du transcodeur, l'utilisateur en est informé. Il est possible que le transcodeur en question soit déjà lancé dans un autre processus ou alors que le fichier exécutable ne soit plus présent.

Le bouton « next » (qui est noté « next track » sur l'interface finale) permet à l'utilisateur de dire au transcodeur de passer à la musique suivante dans la playlist qui est actuellement jouée par le transcodeur sélectionné.

Attention, il est possible que la configuration du transcodeur demande à être régénérée pour bien fonctionner. L'utilisateur ne doit pas hésiter à utiliser le bouton « update » pour régénérer la configuration du transcodeur sélectionné.

5.10.5 CAPTURE LIVE

L'utilisateur peut choisir de couper la lecture du calendrier (et donc des playlists) pour capturer le son, en provenance de son micro ou de sa carte son, et le diffuser via le transcodeur en question.

Dans la partie « Live capture », la liste des différents périphériques audio de l'ordinateur est affichée. Le bouton à sa droite permet de rafraîchir la liste. Une fois un périphérique sélectionné et le transcodeur lancé, l'utilisateur peut démarrer la capture avec le bouton « Start » et la stopper avec « Stop ».

À savoir que cette fonctionnalité n'est pas encore tout à fait au point et peut présenter des bugs.
Vérifier le log lors de l'activation.

5.10.6 HISTORIQUE DE DIFFUSION

L'historique de diffusion est enregistré (quelle musique est passée à quelle heure) et l'utilisateur peut générer un fichier de type PDF avec le bouton « generate ». Il lui sera demandé l'emplacement désiré pour l'enregistrement du fichier. Il est aussi possible de vider l'historique avec le bouton « clear ».

5.11 GESTION DU SERVEUR DE DIFFUSION

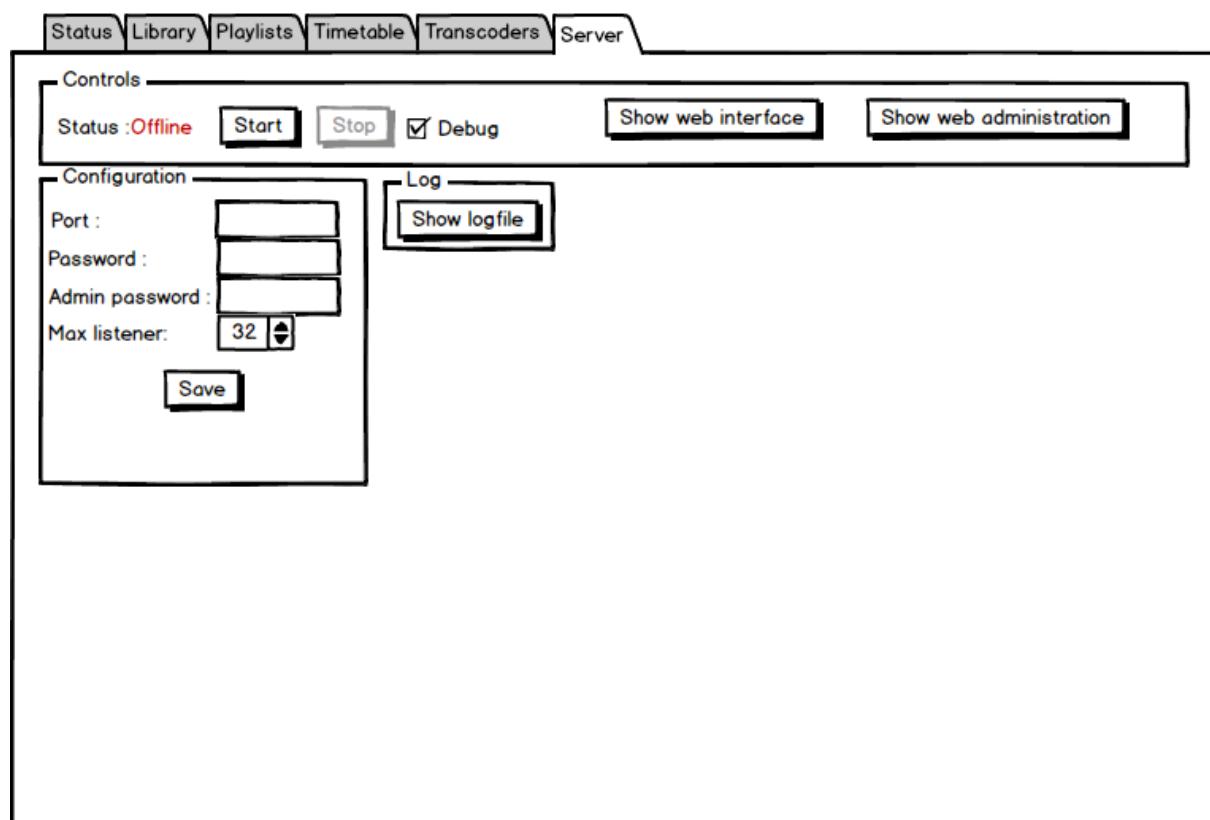


Figure 14 - Onglet "server"

L'onglet serveur propose un serveur de diffusion interne (local) pour la webradio. Bien entendu, l'utilisateur doit créer un transcodeur qui s'y connectera afin de diffuser le flux audio.

Le serveur de diffusion permet de distribuer un flux audio à des auditeurs.

5.11.1 CONTRÔLES

La partie « controls » permet de démarrer ou arrêter le serveur de la webradio via les boutons « start » et « stop ». La case à cocher « Debug » permet de lancer le serveur avec son log dans une fenêtre console. À sa droite, un bouton permet de lancer l'interface web et un autre l'administration web via le navigateur par défaut de l'utilisateur.

Si une erreur est détectée lors du lancement du serveur, l'utilisateur en est informé. Il est possible que le serveur soit déjà lancé dans un autre processus ou alors que le fichier exécutable ne soit plus présent.

5.11.2 LOG

Si l'utilisateur lance le serveur en mode « debug », il peut avoir un log sous forme de programme console qui apparaît dans une fenêtre minimisée. Le bouton « Show logfile » permet d'afficher le fichier de log du serveur dans le bloc note.

5.11.3 CONFIGURATION

Différents éléments sont configurables pour le serveur de diffusion :

- Port : le port de connexion au serveur pour le transcoder (Défaut : 8000)
- Password : Le mot de passe de connexion au serveur pour le transcoder
- Admin password : Le mot de passe pour l'accès à l'administration web du serveur
- Max listener : Le nombre maximum de connexions au serveur (connexion client/auditeur).
Par défaut : 1. Maximum : 2000.

Attention : Les 2 mots de passe doivent être différents !

5.11.4 INTERFACE WEB

Une interface web est fournie avec le serveur de diffusion. L'outil utilisé pour créer le serveur génère tout seul cette interface ainsi que les informations qui y figurent. Cette interface permet de voir différentes informations sur le serveur et le stream. Il est aussi possible de télécharger le fichier permettant d'écouter le flux directement sur un lecteur multimédia.

SHOUTcast Stream Status

The screenshot shows the SHOUTcast Stream Status interface. At the top, there is a navigation bar with links: Status, Song History, Listen, Stream URL, Admin Login, and Server Login. The title above the main content area is "SHOUTcast Server v2.2.1.109/win64". Below the title, a dark header bar says "Current Stream Information". Underneath, there is a table of stream details:

Server Status:	Server is currently up and private
Stream Status:	Stream is up at 56 kbps with 0 of 32 listeners
Stream Name:	My Test Server
Content Type:	audio/aacp
Stream Genre:	Misc
Stream URL:	http://www.shoutcast.com

Figure 15 - Interface web ShoutCAST serveur

Un historique des morceaux joué par le serveur est aussi disponible.

5.11.5 ADMINISTRATION WEB

Via l'interface web décrite précédemment, il est possible d'accéder à une section d'administration. Pour se faire, il faut cliquer sur le lien « Admin login » puis entrer le nom d'utilisateur « admin » et le mot de passe configuré pour le serveur.

The screenshot shows the SHOUTcast Listener and Status web interface. At the top, it displays 'Uptime: 14 minutes 29 seconds' and 'SHOUTcast Server v2.2.1.109/win64'. Below this is a navigation bar with links: 'Listeners' (selected), 'Log (Tailing | Save)', 'Song History', 'Ban List', 'Reserved List', 'Admin Logout', and 'Server Login'. A dark header bar says 'Current Stream Information'. On the left, there's a 'Stream Details' panel containing log file information ('Log file: sc_serv.log', 'Configuration file: examples/sc_serv_basic.conf', 'Intro file is empty', 'Backup file is empty'), idle timeouts ('Idle timeouts are 30s'), source connection type ('Source connection type: v2'), and artwork status ('Stream artwork not available', 'Playing artwork not available'). To the right, detailed stream information is listed:

Server Status:	Server is currently up and private
Stream Status:	Stream is up at 56 kbps with 0 of 32 listeners
Stream Name:	My Test Server
Content Type:	audio/aacp
Stream Genre:	Misc
Stream URL:	http://www.shoutcast.com
Stream Source:	127.0.0.1 [kick]
Stream Uptime:	22 seconds

Figure 16 - Administration web ShoutCAST serveur

Pour l'administrateur, il est possible de :

- Voir la liste des « listeners » (clients qui écoutent la webradio depuis ce serveur)
- Voir le log du serveur
- Gérer une liste d'adresses IP bannies
- Gérer une liste d'adresses IP qui disposent d'un accès réservé (si une adresse réservée désire écouter la webradio, mais que le serveur est plein, un client sera éjecté du serveur pour laisser une place au client disposant d'une adresse réservée)

Tous ces services sont fournis par l'interface d'administration web.

6 ANALYSE ORGANIQUE

6.1 ENVIRONNEMENT

- Langage : C#/.NET
- IDE : Visual Studio 2013
- OS : Windows 7 64 bits

6.2 DIAGRAMME DE CLASSES

6.2.1 MODEL

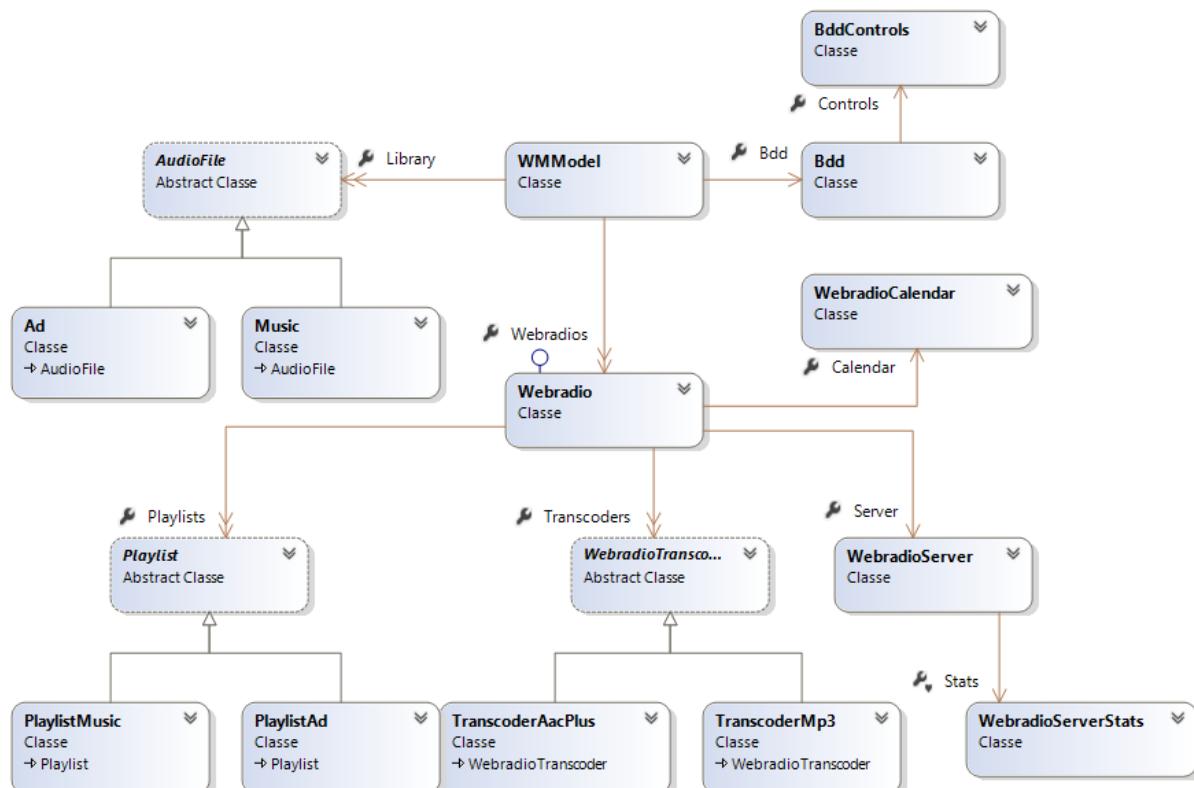


Figure 17 - Diagramme model

6.2.2 MVC

L'application est codée en suivant le *modèle* (patron) MVC (<http://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur>).

MVC Architecture (basic)

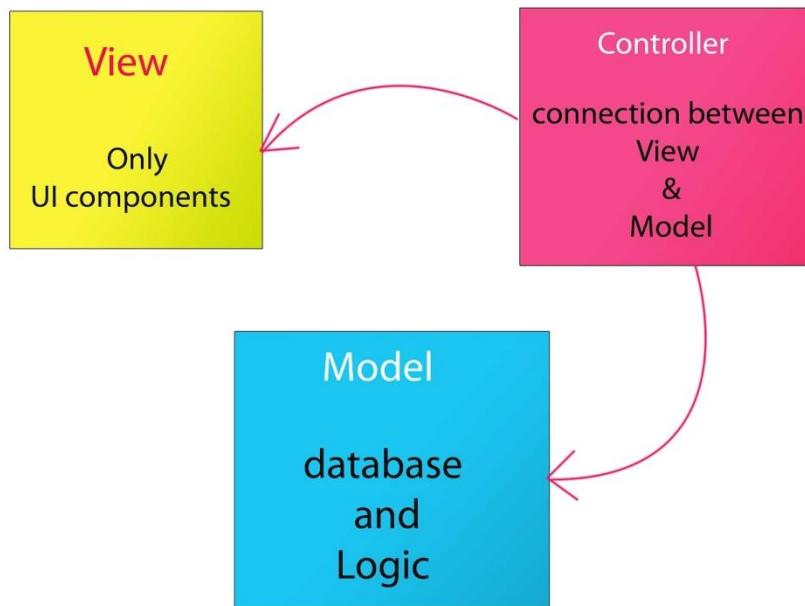


Figure 18 - MVC

L'application dispose de 2 vues :

- AdminView : Vue d'administration de webradio
- SelectionView : Vue de gestion des webradios

Chacune des vues à son controller : AdminController et SelectionController.

Le *model* est décrit dans le chapitre précédent. C'est la classe nommée « WMModel ». Elle contient toute la logique et les différentes données du programme.

6.2.3 OBSERVATEURS/SUJET

Le patron de conception observateur/observable est utilisé en programmation pour envoyer un signal à des modules qui jouent le rôle d'observateur. En cas de notification, les observateurs effectuent alors l'action adéquate en fonction des informations qui parviennent depuis les modules qu'ils observent (les « observables »).

Source : Wikipédia

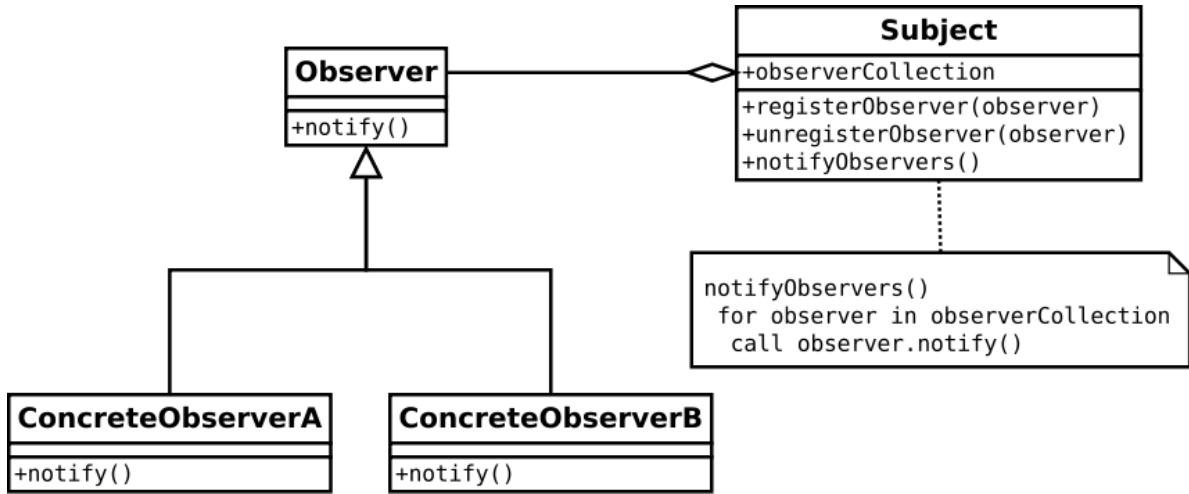


Figure 19 - Pattern observateurs/sujet

Le schéma ci-dessus explique le principe de ce patron de conception. Le sujet dispose d'une liste d'observateurs qui s'inscrivent via la méthode « registerObserver » et se désinscrivent avec « unregisterObserver ». La méthode « notifyObservers » parcourt la liste et appelle la méthode « notify » de chaque observateur. Les observateurs peuvent être de différent type, c'est pour cela qu'une interface « Observer » est créée afin que la méthode « notifyObservers » puisse appeler « notify » sans prendre en compte le type réel de l'observateur.

Dans ce projet, le sujet est le *model* et les observateurs sont les *controllers* des différentes *vues* du projet. Ces derniers vont ensuite appeler la méthode *UpdateView* de leur *vue* respective. L'interface *IController* est égale à « *Observer* » dans le schéma précédent. La méthode « *UpdateView* » correspond à « *notify* ». Dans le *model*, c'est la méthode « *UpdateObservers* » qui s'occupe d'appeler la méthode « *UpdateView* » de chaque contrôleur.

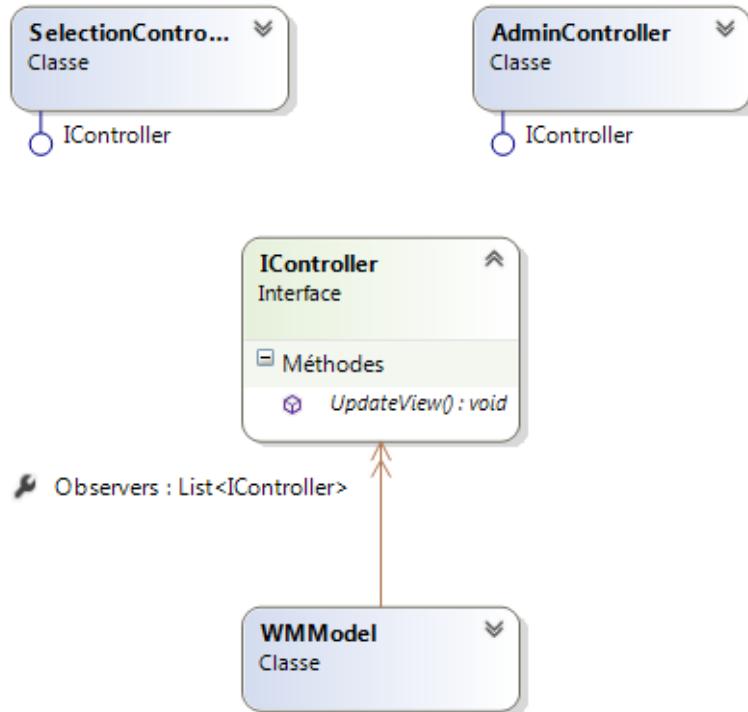


Figure 20 - Observateurs/sujet dans WebradioManager

Il faut noter que le *model* dispose de 2 méthodes « `UpdateObservers` ». L'un n'a aucun paramètre et l'autre prend l'identifiant d'une webradio. La deuxième permet de mettre à jour seulement les observateurs qui sont liés à la webradio dont l'identifiant est donné en paramètre.

J'ai utilisé ce système, car il permet de lancer plusieurs fenêtres d'administration en même temps (qu'elles soient liées à la même webradio ou non). De ce fait, lorsque l'une apporte des modifications au *model*, les autres sont informées et se mettent à jour.

6.2.4 AUDIOTYPE ET STREAMTYPE

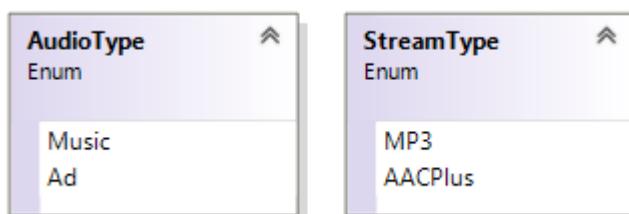


Figure 21 - AudioType et StreamType enum

Le mot clé `enum` est utilisé pour déclarer une énumération, c'est-à-dire un type distinct constitué d'un ensemble de constantes nommées que l'on appelle « liste d'énumérateurs ».

Source : MSDN

Ces 2 enums contiennent comme valeur, l'identifiant statique de leur valeur dans la base de données. C'est-à-dire, par exemple, que la valeur de la constante « Music » dans l'enum « AudioType » est égale à l'identifiant correspond dans la table « taudiotype » de la base de données.

6.2.5 STOCKAGE DES WEBRADIOS

Les données des webradios contenues dans la base de données sont chargées dans le *model* au lancement de l'application.

Ces données sont stockées sous la forme d'un dictionnaire (Dictionary<int,Webradio>). Le fait d'utiliser un dictionnaire avec comme valeur clé, une entier, permet d'y stocker l'identifiant de la webradio référencée comme valeur. Ainsi, la recherche de données dans le *model* est facilitée et accélérée.

6.3 BASE DE DONNÉES

La base de données est présente pour sauvegarder les diverses informations de l'application. La plupart des paramètres sont stockés dans des fichiers texte sous forme de configuration utilisable par les différents transcodeurs et serveurs.

6.3.1 SQLITE



Figure 22 - Logo SQLite

SQLite est une bibliothèque écrite en C qui propose un moteur de base de données relationnelle accessible par le langage SQL. SQLite implémente en grande partie le standard SQL-92 et des propriétés ACID (voir glossaire 9).

Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel client-serveur, mais d'être directement intégrée aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

Source : Wikipédia (<http://fr.wikipedia.org/wiki/SQLite>)

Pour mon projet, j'utilise le logiciel SQLite Studio pour éditer ma base de données et y entrer des données de test : <http://sqlitestudio.pl/>

J'ai choisi SQLite, car il est léger et rapide. Il permet aussi d'avoir une base de données locale simple d'utilisation et multiplateforme.

6.3.2 UTILISATION

Une bibliothèque sous forme d'une DLL⁸ est disponible pour .NET (C#) ici :

<https://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki>

Une classe fournit par le site web suivant : <http://www.dreamincode.net/forums/topic/157830-using-sqlite-with-c%23/#/> permet l'interaction avec un fichier SQLite. Cette classe prendra la place de BddControls dans le diagramme de classe de l'application. Le fichier de base de données « database.db » est à la racine du logiciel. Une copie « vierge » (sans données à l'intérieur, mise à

⁸ Dynamic Link Library

part les tables taudiotype et tstreamtype : voir 0) de la base de données est stockée dans les ressources du logiciel afin d'avoir une BDD de base.

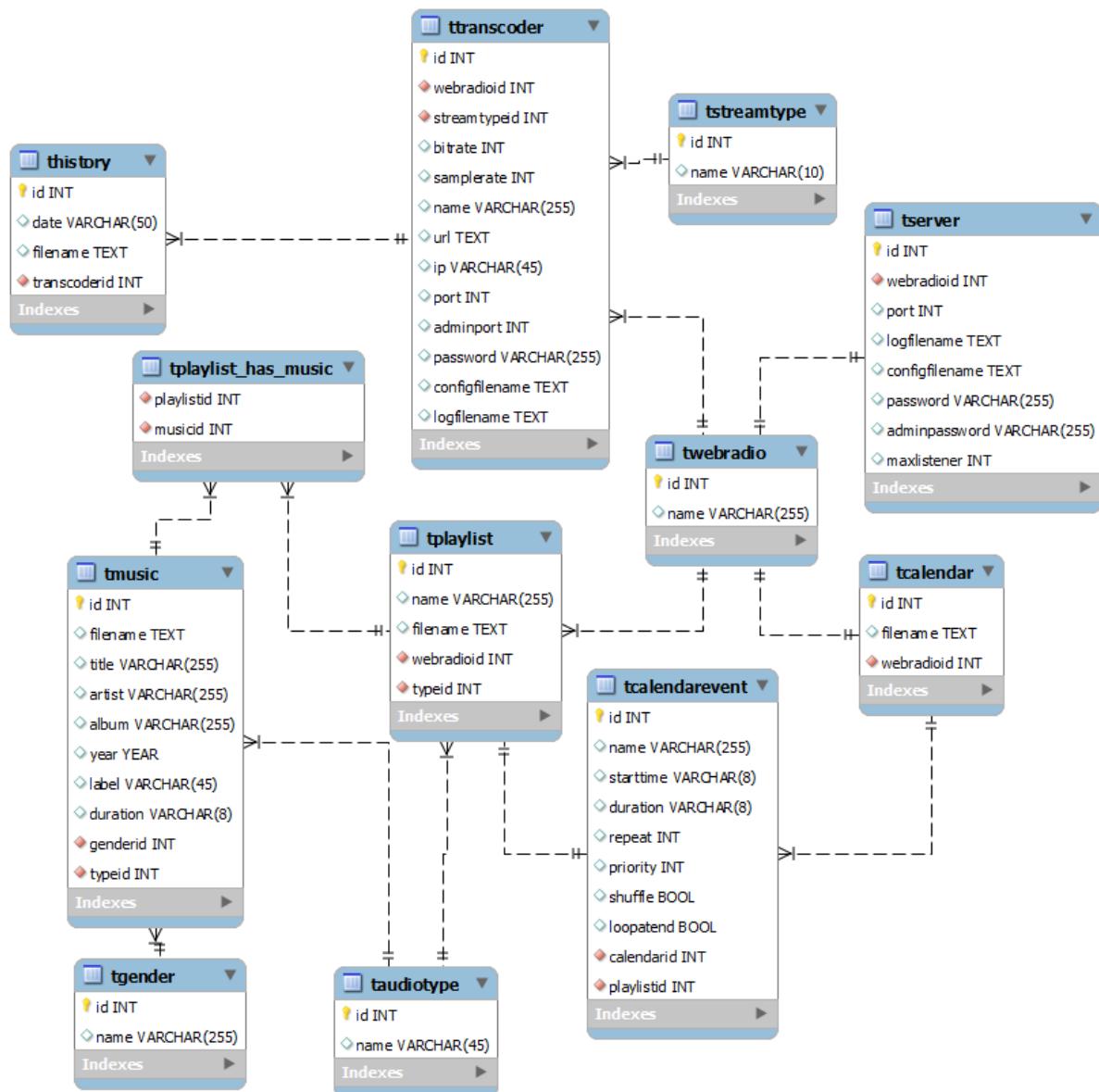
La classe Bdd utilise BddControls (voir diagramme de classe **Erreur ! Source du renvoi introuvable.**) Afin d'effectuer des requêtes sur la base de données. Bdd propose des méthodes simples comme par exemple « AddWebradio ». Ainsi la partie traitement des données se fait dans Bdd et la partie exécution dans BddControls.

La base de données sert principalement à la sauvegarde des données qui sont récupérées dans le *model* à chaque démarrage de l'application.

6.3.3 SUPPRESSION EN CASCADE

SQLite permet la suppression en cascade. C'est-à-dire, lorsqu'un enregistrement est supprimé, toutes les références à ce dernier via des clés étrangères sont supprimées automatiquement.

6.3.4 SCHÉMA



6.3.5 TWEBRADIO

Cette table contient la liste des webradios qui est composée de leur nom.

Nom	Type	Description
Id	Int	Identifiant unique d'une webradio
Name	Varchar(255)	Nom de la webradio

6.3.6 TSERVER

Cette table contient les informations concernant les différents serveurs. Un enregistrement est lié à une webradio via une clé étrangère.

Nom	Type	Description
Id	Int	Identifiant unique d'un serveur
Webradiooid	Int	Clé étrangère (radio possédant ce serveur)
Port	Int	Numéro du port d'écoute du serveur0
Logfilename	Text	Chemin vers le fichier de log du serveur
Configfilename	Text	Chemin vers le fichier de configuration du serveur
Password	Varchar(255)	Mot de passe de connexion au serveur (pour une source)
Adminpassword	Varchar(255)	Mot de passe de l'administration web du serveur
Maxlistener	Int	Le nombre maximal d'auditeurs sur le serveur

6.3.8 TCALENDAR

Cette table contient les informations pour un calendrier. Un enregistrement est lié à une webradio via une clé étrangère.

Nom	Type	Description
Id	Int	Identifiant unique d'un calendrier
Filename	Text	Chemin vers le fichier XML du calendrier
Webradiooid	Int	Clé étrangère (radio possédant ce calendrier)

6.3.9 TCALENDAREVENT

Cette table contient les différents événements d'un calendrier. Un événement est lié à un calendrier via une clé étrangère et contient une playlist. Le lien vers cette playlist est aussi effectué avec une clé étrangère vers la table tplaylist.

Nom	Type	Description
Id	Int	Identifiant unique d'un événement
Name	Varchar(45)	Nom de l'événement
Starttime	Varchar(8)	Heure du commencement de l'événement
Duration	Varchar(8)	Durée de l'événement
Repeat	Int	Valeur de répétition de l'événement (voir chapitre Grille horaire)
Priority	Int	Priorité de l'événement
Shuffle	Bool	Défini si l'événement lit la playlist de façon aléatoire
Loopatend	Bool	Défini si la playlist recommence une fois que tous les morceaux sont écoutés
Calendarid	int	Clé étrangère (Calendrier possédant cet événement)
Playlistid	Int	Clé étrangère (Playlist jouée par cet événement)

6.3.10 TPLAYLIST

Cette table contient les informations sur une playlist (mais pas les fichiers audio qui lui sont liés). Un enregistrement est lié à une webradio via une clé étrangère.

Nom	Type	Description
Id	Int	Identifiant unique d'une playlist
Name	Varchar(255)	Nom de la playlist
Filename	Text	Chemin vers le fichier de la playlist
Webradiooid	Int	Clé étrangère (webradio possédant cette playlist)
Typeid	Int	Clé étrangère (le type de la playlist)

6.3.11 TAUDIOTYPE

Cette table est liée à l'enum « AudioType », car les valeurs qui y sont enregistrées sont codées en « dur » dans l'enum. Elle définit les types de fichier audio possible (musique ou publicitaire à l'heure actuelle).

Nom	Type	Description
Id	Int	Identifiant unique d'un type audio
Name	Varchar(45)	Nom du type audio

6.3.12 TMUSIC

Cette table contient la bibliothèque de fichiers musicaux indexés. Un enregistrement est lié à un type de fichier audio via une clé étrangère ainsi qu'à un genre de la table tgender.

Nom	Type	Description
Id	Int	Identifiant unique d'une musique
Filename	Text	Chemin vers le fichier musical
Title	Varchar(255)	Titre du morceau
Artist	Varchar(255)	Artiste du morceau
Album	Varchar(255)	Album du morceau

Year	Year	Année du morceau
Label	Varchar(45)	Label du morceau
Duration	Varchar(8)	Durée du morceau
Genderid	Int	Clé étrangère (genre du morceau)
Typeid	Int	Clé étrangère (type du morceau)

6.3.13 TGENDER

Cette table contient la liste des genres des fichiers musicaux contenues dans la table tmusic.

Nom	Type	Description
Id	Int	Identifiant unique d'un genre musical
Name	Varchar(255)	Nom du genre

6.3.14 TPLAYLIST_HAS_MUSIC

Cette table permet de lier un fichier audio à une playlist via les identifiants de leur enregistrement.

Nom	Type	Description
Playlistid	Int	Clé étrangère (Playlist concernée)
Musicid	Int	Clé étrangère (Musique concernée, qui fait partie de la playlist ayant l'identifiant Playlistid)

6.3.15 THISTORY

Cette table contient l'historique (sous forme de nom de fichier) d'un transcoder (dont la liaison est effectuée avec une clé étrangère).

Nom	Type	Description
Id	Int	Identifiant unique d'un élément d'historique
Date	Varchar(50)	Date de l'événement dans l'historique
Filename	Text	Chemin vers le fichier joué
Transcoderid	Int	Clé étrangère (Transcodeur ayant joué cette musique)

6.3.16 TTRANSCODER

Cette table contient les informations sur les différents transcodeurs. Un transcoder est lié à une webradio via une clé étrangère. Une autre clé étrangère permet de définir le type de stream (table tstreamtype). À l'heure actuelle, il y a 2 types : MP3 et AAC+.

Nom	Type	Description
Id	Int	Identifiant unique d'un transcoder
Webradiooid	Int	Clé étrangère (Webradio possédant ce transcoder)
Streamtypeid	Int	Clé étrangère (Type du transcodeur MP3 ou autre)
Bitrate	Int	Débit binaire du flux (en bits/s)
Samplerate	Int	Taux d'échantillonnage du flux
Name	Varchar(255)	Nom du flux
Url	Text	URL concernant le flux (site web par exemple)
Ip	Varchar(45)	Adresse IP du serveur de diffusion
Port	Int	Port du serveur de diffusion
Adminport	Int	Port d'administration du transcoder

Password	Varchar(255)	Mot de passe du serveur de diffusion
Configfilename	Text	Chemin vers le fichier de configuration du transcodeur
Logfilename	Text	Chemin vers le fichier de log du transcodeur

6.4 SCHÉMA DE DIFFUSION

6.4.1 PRINCIPE DE BASE

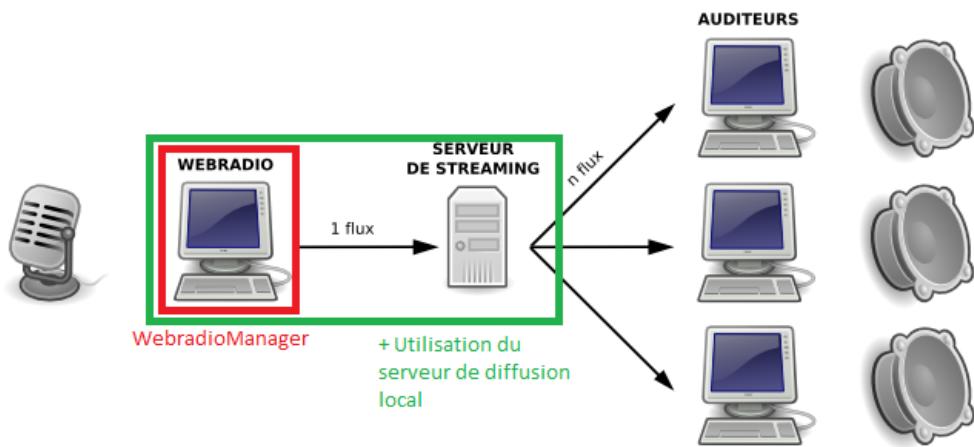


Figure 23 - Principe de base de diffusion

Ce type de diffusion est appelé « client-serveur ». C'est le principe de base de streaming audio/vidéo. Dans WebradioManager, la diffusion est possible sur des serveurs distants ou sur un serveur interne (local).

6.4.2 INFOMANIAK

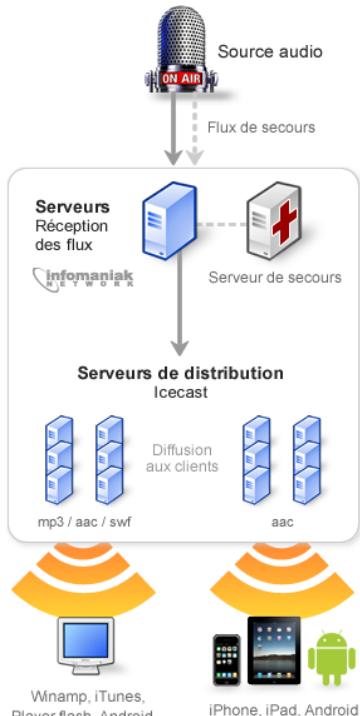


Schéma de fonctionnement
du streaming radio live Infomaniak Network SA

Figure 24 - Schéma de diffusion

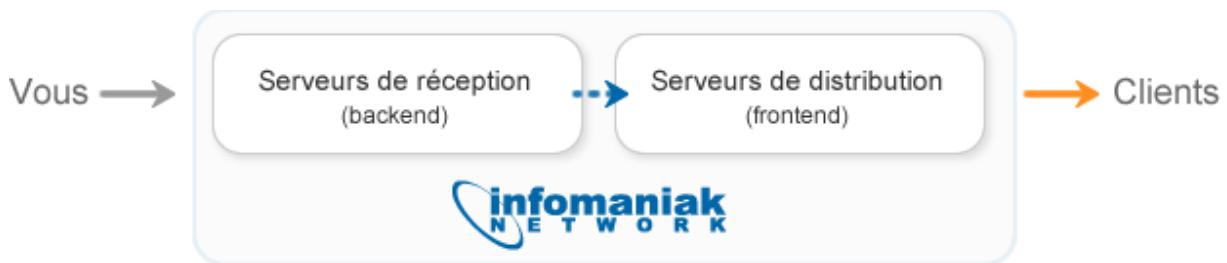


Figure 25 - Schéma de diffusion 2

Ces 2 schémas montrent le fonctionnement de la diffusion des webradios via infomaniak. Le logiciel avec les transcodeurs est la « source audio ». Infomaniak demande seulement un flux audio, peut importer le logiciel utilisé pour le diffuser. C'est ensuite leurs serveurs qui s'occuperont de diffuser le flux aux différents auditeurs.

6.5 SHOUTCAST



Figure 26 - Logo Shoutcast

6.5.1 PRÉSENTATION

SHOUTcast est le nom d'un protocole et d'un serveur de diffusion pour webradio ou pour webtv. Il a été créé par la société Nullsoft en même temps que le logiciel client Winamp pour l'écoute. Le protocole s'appuie sur deux protocoles, HTTP et ICY pour supporter les ID tag (« title streaming »).

Source : Wikipédia (<http://fr.wikipedia.org/wiki/SHOUTcast>)

SHOUTcast a été racheté récemment par la société Radionomy (<http://www.pcworld.fr/business/actualites,societe-belge-radionomy-confirme-rachat-winamp-shoutcast,545553,1.htm>). Cela a pour répercussion que les fichiers ne sont plus disponibles sur le site web de Shoutcast. Mais malgré cela, les fichiers de la version 2 sont toujours disponibles sur le forum de Winamp⁹. Ce sont donc ces derniers qui seront utilisés pour le projet.

6.5.2 POURQUOI CET OUTIL ?

⁹ Lecteur multimédia développé par Nullsoft

J'ai choisi Shoutcast, car il propose des outils en ligne de commande ainsi qu'une gestion facilitée via des fichiers XML¹⁰ ou texte basiques. Cela permet d'interagir plus facilement avec des applications externes telles que la mienne.

6.5.3 SERVEUR

Shoutcast, parmi ses outils, propose un serveur en ligne de commande qui sera utilisée dans ce projet. Il permet de diffuser un flux qu'il reçoit et qui est envoyé par, par exemple, un [transcodeur](#). Ce flux est distribué aux clients (auditeurs dans ce cas) qui désirent écouter la webradio. Ce serveur propose aussi une interface web pour visualiser le statut (données sur le flux actuel, musique en cours, etc.) et d'administrer (il faut être authentifié en tant qu'administrateur avec le mot de passe défini dans la configuration du serveur) le serveur. Ces détails sont expliqués dans [l'analyse fonctionnelle \(5\)](#).

Plus de détails dans la partie consacrée au [serveur interne de diffusion](#). (6.13)

6.5.4 TRANSCODER

Tout comme le serveur, le transcoder est un outil fourni par Shoutcast en ligne de commande. Il permet de diffuser un flux sur un serveur de diffusion. Ce flux peut-être en MP3 ou AAC+. Il donne aussi la possibilité de créer des playlists et de les agencer dans un calendrier XML. Les détails concernant ce système sont expliqués dans la suite de cette analyse organique.

Il gère de façon indépendante les horaires, les priorités entre les playlists et la lecture des fichiers musicaux. Le programme de ce projet va s'occuper de générer les fichiers nécessaires au transcoder en fonction des paramètres définis par l'utilisateur ainsi que d'afficher ces informations de façon visuelle (exemple : calendrier) afin de faciliter la manipulation et la configuration.

¹⁰ eXtensible Markup Language : Langage de balisage extensible

6.5.5 SCHÉMA DE FONCTIONNEMENT RÉSUMÉ

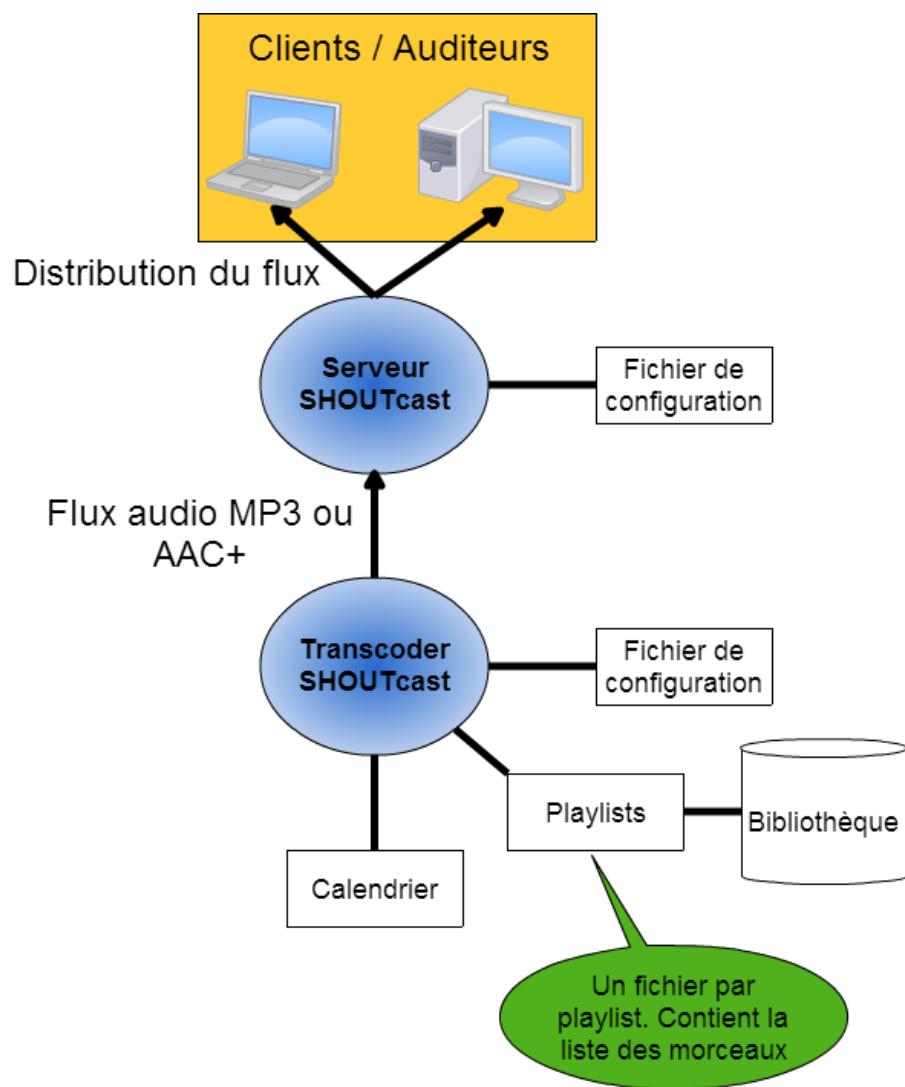


Figure 27 - Schéma shoutcast

6.6 STRUCTURES DES DOSSIERS/FICHIERS

6.6.1 SCHÉMA

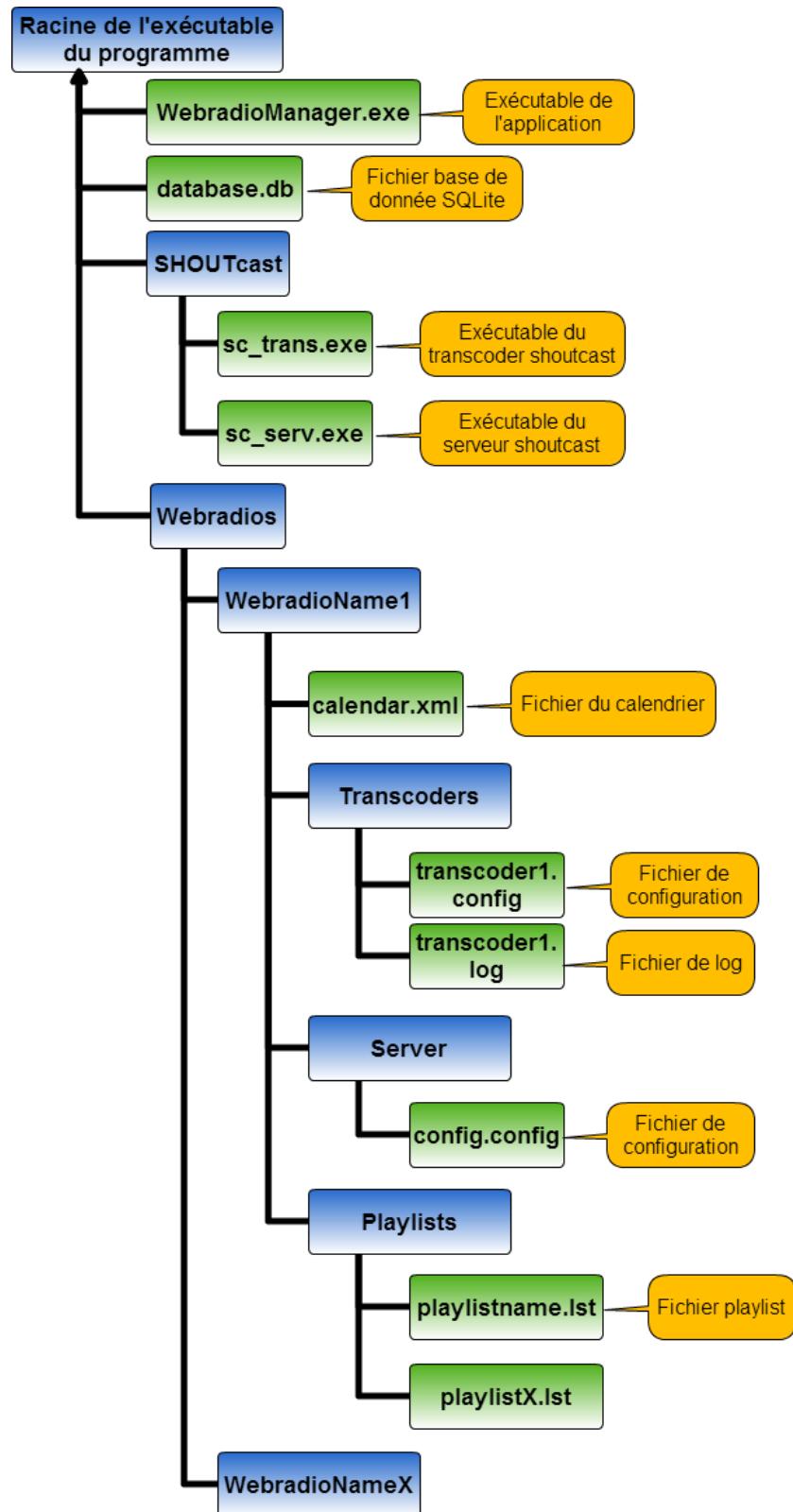


Figure 28 - Schéma structure des fichiers/dossiers

Ce schéma décrit l'organisation des fichiers dans le dossier de l'application. La base de données contient les informations pour les transcodeurs et les serveurs ainsi que le chemin vers les différents fichiers de ces dernières (configuration, calendrier, etc.). Cela permet à l'application de savoir où trouver les fichiers lors des traitements.

6.6.2 EXÉCUTABLES SHOUTCAST

Attention : Il n'y a pas un exécutable de transcodeur par transcoder de webradio ni un exécutable de serveur par webradio. Il existe seulement un seul et unique exécutable transcodeur et serveur dans le dossier « shoutcast ». Ces exécutables peuvent être lancés avec le chemin vers un fichier de configuration en paramètre. Par exemple : À chaque fois qu'un transcodeur devra se lancer, une nouvelle instance de l'exécutable sc_trans.exe présent dans le dossier « shoutcast » sera lancée dans un nouveau processus et utilisera le fichier de configuration du transcodeur en question. Cela a été décidé, car si une mise à jour des exécutables doit être faite, seuls les 2 exécutables du dossier « shoutcast » seront mise à jour.

Voici ce principe illustré :

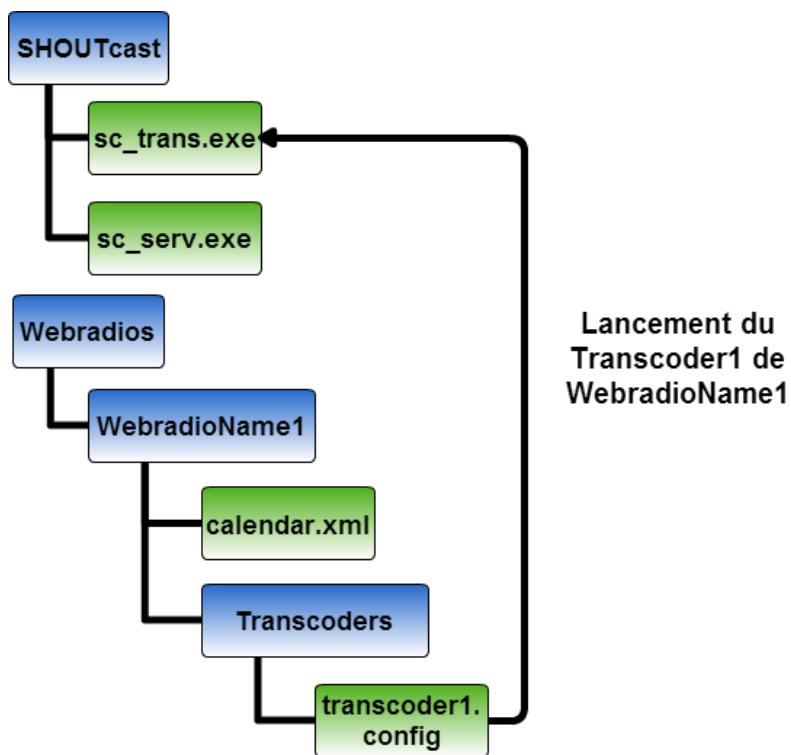


Figure 29 - Exemple lancement transcoder

6.7 INITIALISATION DE L'APPLICATION

La première fenêtre à se lancer est la SelectionView. C'est elle qui va appeler les différentes méthodes du *model* (via son *controller*) servant à l'initialisation.

C'est la classe Bdd qui s'occupe du traitement des données pour les passer ensuite au *model* pour que ce dernier remplisse ses champs « webradios » et « library ». Le diagramme de séquence suivant explique le déroulement de façon simplifiée.

■

Figure 30 - Diagramme de séquence initialisation application

À la fin de ce diagramme, SelectionView est affichée à l'utilisateur avec les informations recueillies avec UpdateView(). Cette méthode met à jour la *vue* avec les informations disponibles dans le *model* la concernant.

Le *model* vérifie avant tout que les dossiers de « base » (webradios, shoutcast) sont présents. Si ce n'est pas le cas, il les crée. Cette vérification est effectuée dans la méthode « LoadWebradios » du *model*.

Le *model* est donc rempli au démarrage de l'application. Toutes les informations contenues dans la base de données sont récupérées, traitées et ajoutées au *model*. Cela permet d'éviter un nombre de requêtes inutiles vers la base de données et de travailler avec les informations stockées en mémoire sous forme d'instances de classes dans le *model*.

Les 2 méthodes « LoadWebradios() » et « LoadLibrary() » de la classe Bdd s'occupent du traitement des informations. La première charge toutes les informations de façon hiérarchique (une webradio a un calendrier, qui lui dispose d'événement, etc.) pour chaque webradio de la base de données. La 2^e charge les musiques présentes dans la bibliothèque ainsi que les playlists qui leur sont associées.

6.8 WEBRADIO

6.8.1 CLASSES ASSOCIÉES

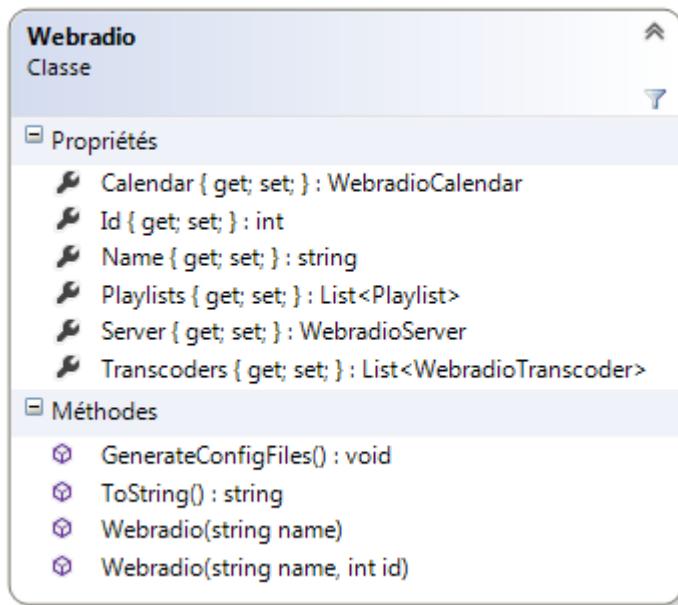


Figure 31 - Classe "Webradio"

La méthode « `ToString` » écrase (override) la méthode du même nom implémentée dans la classe parente de Webradio (Object). Elle retourne le nom de la webradio avec son identifiant. Cela permet lors de l'ajout d'objets de type Webradio à un ListBox, par exemple, d'afficher le nom de cette dernière, car les composants de type ListBox, ComboBox ou autres utilisent la méthode « `ToString` » des objets qu'ils contiennent pour afficher leur nom.

6.8.2 AFFICHAGE DES WEBRADIOS DISPONIBLES

Les webradios disponibles sont affichées dans la ListBox centrale de la fenêtre SelectionView. Pour la remplir, cela est effectué dans la méthode « `UpdateView()` » qui va récupérer les webradios du *model* et remplir la liste « `d'items11` » avec les objets de type Webradio obtenus. La ListBox appelle la méthode « `ToString` » des objets qu'elle contient pour afficher sa liste. Dans ce cas, la méthode « `ToString` » des objets Webradio sera appelée pour chacun d'entre eux.

6.8.3 CRÉATION

Une webradio ne peut pas avoir un nom qui dépasse 255 caractères. Pour se faire, la limitation est directement configurée dans la propriété « `MaxLength` » du TextBox permettant à l'utilisateur de nommer sa webradio.

¹¹ Voir [glossaire](#)

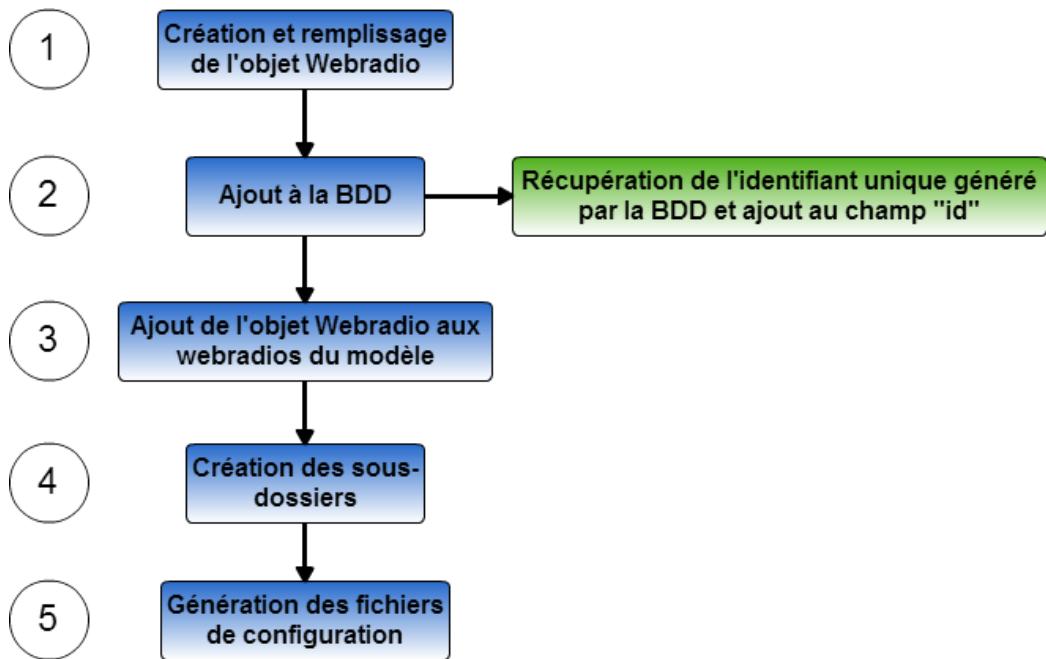


Figure 32 - Schéma création webradio

Comme présenté dans le schéma ci-dessus, la création se divise en 5 étapes, ces dernières s'effectuent dans le *model* et la méthode « CreateWebradio » :

- 1 : Instanciation¹² d'une classe Webradio avec le nom donné en paramètre à la méthode « CreateWebradio() ». Instanciation des classes nécessaires à la webradio (WebradioServer, WebradioCalendar et les différentes listes d'objet tel que la propriété Playlists). Seule la propriété « id » de l'objet webradio n'est pas remplie, car il s'agit de son identifiant dans la base de données, il sera donc rempli par la suite.
- 2 : L'objet créé est passé à la classe Bdd qui s'occupe de l'ajout de toutes ces informations dans la base de données via sa méthode « AddWebradio() » qui retourne l'identifiant qui a été attribué à cette nouvelle webradio par la base de données. Cet id est récupéré dans la méthode « CreateWebradio() » précédente et ajouté à l'objet webradio.
- 3 : L'objet webradio final est ajouté à la liste d'objets de type « Webradio » du *model*.
- 4 : Création des différents sous-dossiers pour stocker les fichiers de la nouvelle webradio.
- 5 : La méthode « GenerateConfigFiles() » est appelée afin de créer les fichiers de configuration nécessaires à la webradio. Plus d'informations sur cette méthode dans [ce chapitre](#). (6.8.8)

¹² Voir [glossaire](#)

6.8.4 CHARGEMENT

Une webradio est chargée lorsqu'elle est sélectionnée via SelectionView. L'identifiant de la webradio à ouvrir est passé au *controller* de SelectionView. Ce dernier instancie un nouvel AdminController avec l'identifiant. Une fenêtre de type AdminView est ensuite ouverte avec les informations de la webradio sélectionnée. Ces informations viennent du *model*. Voici le diagramme de séquence pour la création d'une nouvelle instance d'une AdminView :

15

Figure 33 - Diagramme de séquence instantiation AdminView

Après cette initialisation, le nouvel AdminController est ajouté à la liste d'observateurs du *model*.

À la création de la fenêtre, cette dernière appelle la méthode « CheckFolders » de son *controller*. Cette méthode va faire de même sur le *model*. En fin de compte, cette méthode vérifie la présence de tous les dossiers nécessaires à la webradio lancée avec AdminView (serveur, playlists et transcodeurs). Si un d'eux n'est pas présent, il est créé.

Comme montrée dans le diagramme, la méthode « UpdateView » de AdminView est appelée afin de charger les informations dans ses différents composants. La *vue* contient l'id de la webradio qui lui est attribuée. C'est avec cet id qu'elle va pouvoir effectuer des actions sur le *model* via le *controller*.

En ce qui concerne le remplissage des différentes ComboBox et ListBox, leurs Items ne seront pas de simples chaînes de caractères, mais des objets complets. Par exemple, les ComboBox affichant les

playlists disponibles comme dans l'onglet « Library », sont remplis avec des objets de type Playlist. Il est important que les classes ajoutées à des composants de ces types implémentent une méthode « ToString », car c'est celle qui est appelée par le composant lorsqu'il affiche, sous forme de chaîne de caractères, les éléments de sa liste. Cette méthode « override¹³ » celle héritée par la classe parente « Object¹⁴ ». De cette façon, il est facile de personnaliser les informations renvoyées par classe quand un composant utilise sa méthode ToString.

Le fait d'utiliser directement des objets dans les ListBox ou ComboBox permet de garder un pointeur sur les objets présents dans le *model*. Cela permet de manipuler un objet et ses modifications seront répercutées partout où il est utilisé.

Le calendrier est un composant spécial et il est aussi mis à jour lors de l'UpdateView. Pour plus d'information sur son fonctionnement, rendez-vous [au chapitre le concernant](#).

6.8.5 DUPLICATION

La duplication s'effectue en plusieurs étapes :

- Création du nom de la webradio « clone » : « Copy of » + nom de la webradio qui est clonée.
- Utilisation de la méthode du *model* nommée « [CreateWebradio](#) » puis récupération de la nouvelle webradio fraîchement ajoutée au *model* avec la méthode « [GetWebradioByName](#) ».
- Copie des informations (configuration) des différents éléments (serveur, playlists, calendrier et transcoder) de la webradio clonée, vers la webradio « clone ». Cette copie s'effectue en fait en utilisant les différentes méthodes de création d'éléments déjà présentes dans le *model*.
- UpdateObservers quand l'objet « clone » est à jour.

Plus de précision concernant la copie des informations. Exemple avec les transcoders :

```
foreach(WebradioTranscoder transcoder in webradio.Transcoders)
{
    this.CreateTranscoder(transcoder.Name, transcoder.StreamType, transcoder.SampleRate,
    transcoder.Birate, transcoder.Url, transcoder.Ip, transcoder.Port,
    transcoder.AdminPort, transcoder.Password, newWebradio.Id);
}
```

[WMMModel.cs](#)

Le foreach parcourt les transcodeurs de la webradio clonée et appelle la méthode création de transcoder avec les paramètres du transcoder courant, mais utilise l'identifiant de la nouvelle

¹³ Le modificateur override est nécessaire pour étendre ou modifier l'implémentation abstraite ou virtuelle d'une méthode, d'une propriété, d'un indexeur ou d'un événement hérité(e).

¹⁴ Voir [glossaire](#)

webradio. De ce fait, le nouveau transcoder sera créé pour la nouvelle webradio, mais avec les paramètres de la webradio clonée.

Concernant les playlists, par exemple, il y a une instance d'objet qui s'ajoute. La méthode « CreatePlaylist » utilise un paramètre avec le mot clé « out » (<http://msdn.microsoft.com/fr-fr/library/ee332485.aspx>). Cela permet, comme un pointeur, de modifier la valeur de la variable qui a été donnée en paramètre. Ce paramètre est de type Playlist. C'est-à-dire que l'objet de type Playlist donné en paramètre à la fonction sera rempli par la Playlist créée dans « CreatePlaylist » :

```
Playlist newPlaylist;

if(this.CreatePlaylist(playlist.Name,newWebradio.Name,newWebradio.Id,playlist.Type,
out newPlaylist))

{
    newPlaylist.AudioFileList = new List<string>(playlist.AudioFileList);
}
```

WMMModel.cs

6.8.6 SUPPRESSION

La suppression d'une webradio s'effectue via son identifiant. La méthode « DeleteWebradio() » du *model* va en premier temps supprimer la webradio de la base de données, puis de sa liste (dictionnaire) de webradios. La suppression dans la liste se fait via la méthode « Remove » proposée par les listes de type Dictionary qui prend l'identifiant de la webradio à supprimer.

6.8.7 CHANGEMENT DE NOM

Le schéma de traitement principal est le suivant :

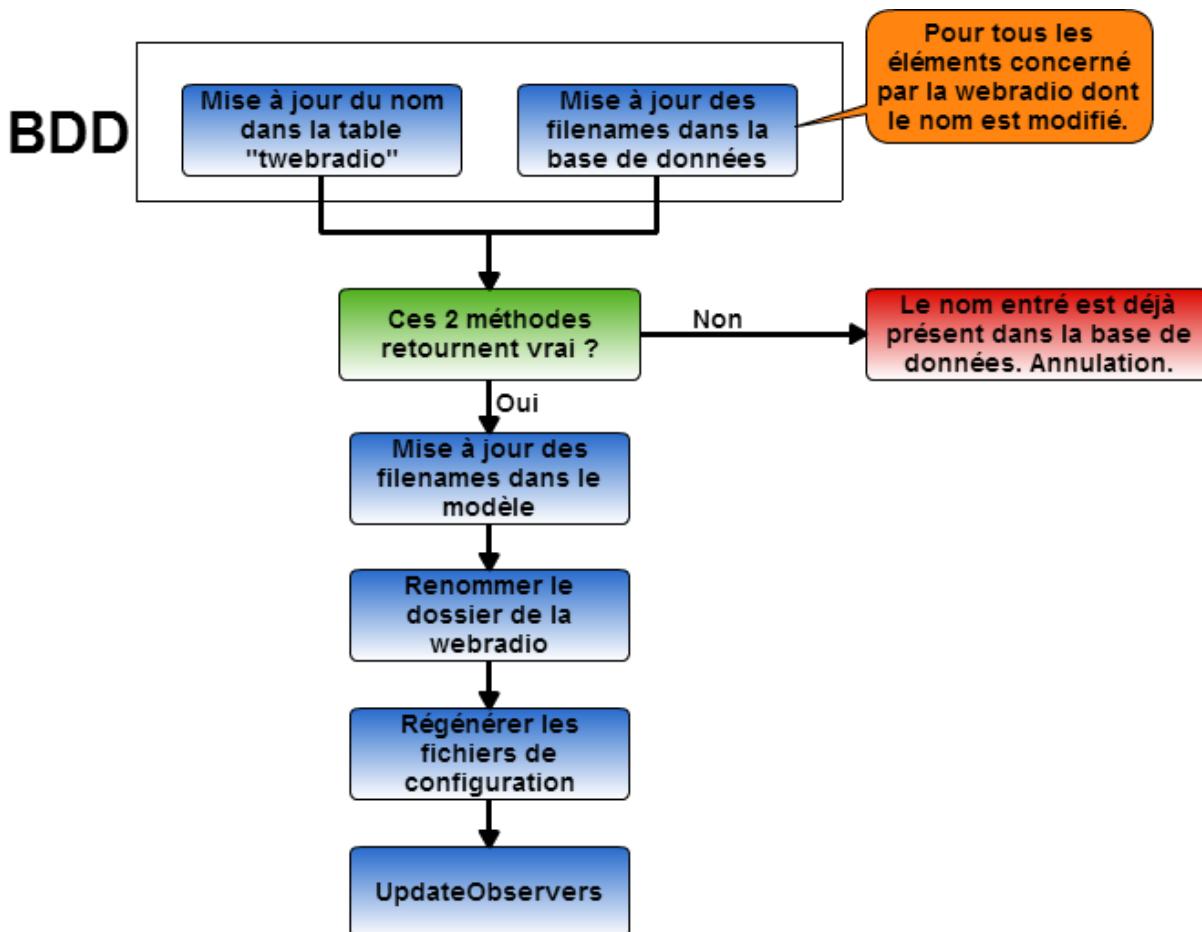


Figure 34 - Schema changement nom webradio

Les filenames (chemin vers les différents fichiers de configuration et de log des éléments d'une webradio) sont mis à jour, car ils réfèrent à des fichiers qui se situent dans un dossier qui porte le nom de la webradio. Ce dossier est renommé, les anciens filenames seront donc corrompus. La mise à jour se fait simplement avec un replace :

```
transcoder.ConfigFilename = transcoder.ConfigFilename.Replace(oldName, newName);
```

WMMModel.cs

Lors d'une modification de nom, tous les processus de la webradio sont arrêtés, car renommer un dossier est impossible si les fichiers à l'intérieur sont utilisés.

6.8.8 GÉNÉRATION DES CONFIGURATIONS

La classe Webradio dispose d'une méthode « GenerateConfigFiles() » qui appelle la méthode « GenerateConfigFile() » de chacun de ses membres (Server, Playlists, etc.) ayant besoin d'un fichier de configuration.

Ces méthodes suppriment le fichier de configuration existant (s'il y en a un) et en créent un nouveau avec les informations contenues dans les champs de la classe en question. En fonction de la classe, le type de fichier de configuration sera différent (fichier texte ou fichier XML).

La génération de configuration est appelée lors de la sauvegarde ou le changement d'informations depuis l'AdminView, mais encore lors de la création d'une nouvelle webradio. Plus d'informations pour la génération de configuration de chaque composant du programme dans la suite de la documentation.

6.9 BIBLIOTHÈQUE

6.9.1 CLASSES UTILISÉES

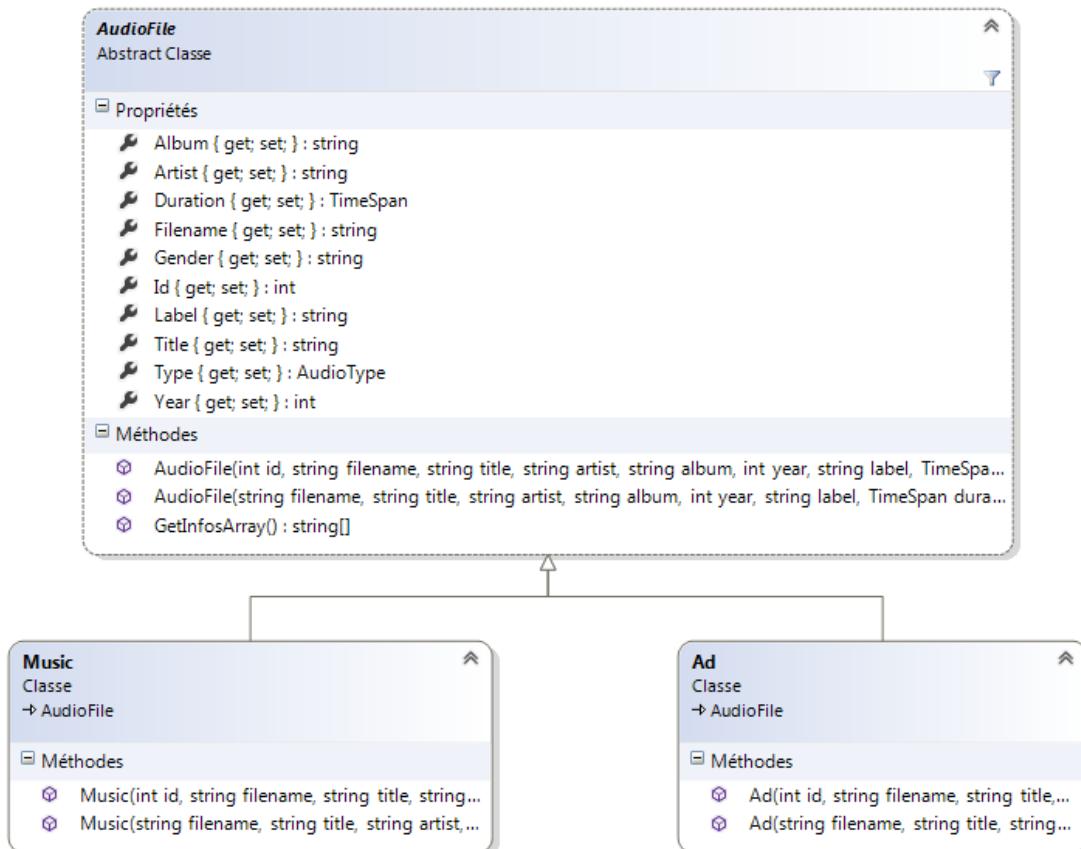


Figure 35 - Classes bibliothèque

6.9.2 MP3

À l'heure actuelle, seuls les fichiers MP3 peuvent être importés. La raison principale est que le flux de sortie de la webradio sera de toute façon compressé donc il n'y a pas d'intérêt à importer des fichiers de meilleure qualité (WAV par exemple). Des fichiers MP3 encodés en 320 kbits/s convient très bien et offrent une très bonne qualité pour une taille de fichier restreinte. De plus, le mp3 est une norme courante et utilisée par tout le monde.

Plus d'informations sur la technique de codage des fichiers MP3 : http://fr.wikipedia.org/wiki/MPEG-1/_2_Audio_Layer_3#Technique_de_codage

6.9.3 IMPORTATION

Depuis l'onglet « library », l'utilisateur choisit s'il veut importer un dossier ou des fichiers à la section « music » ou « ad »(publicité, top-horaire, etc.). Chaque bouton contient une valeur dans sa propriété « tag » qui est soit « Music » ou soit « Ad », car les événements « OnClick » des boutons (regroupés par type d'importation, c'est-à-dire, par dossier ou fichiers) pointent sur une même

méthode et cette propriété « tag » permet de différencier si le bouton cliqué est dans la section « music » ou « ad ».

Dans le cas d'un dossier, le traitement va rechercher tous les fichiers MP3 contenus dans le dossier sélectionné. Si l'utilisateur a désiré d'importer de façon récursive, les sous-dossiers du dossier sélectionné seront aussi analysés.

La méthode statique « GetFiles » de la classe .NET « Directory » permet de récupérer un tableau de string contenant les « filename » (chemin absolu vers les fichiers) des fichiers correspondants au pattern donné en paramètre dans le dossier spécifié. Ce pattern se présente sous la forme : « *.mp3 » pour récupérer seulement les fichiers dont l'extension est « mp3 ». Une option permet de faire cette recherche de façon récursive. Voici un exemple de code pour des fichiers mp3 et récursivement :

```
Directory.GetFiles(FBD.SelectedPath, "*.mp3", SearchOption.AllDirectories);
```

Pour une recherche non récursive, l'option « SearchOption » doit être changée en :

```
SearchOption.TopDirectoryOnly.
```

La *vue* va ensuite passer le tableau de string au *model* via son *controller*. Le *model* va effectuer le traitement (analyse de tags ID3¹⁵ (version 1 et 2) et ajout à la base de donnée/*model*) avec sa méthode « ImportFilesToLibrary ». Cette méthode est générique, il suffit de lui envoyer un tableau de filenames pour fonctionner. Cela permet qu'une importation par fichiers ou dossier puisse utiliser la même méthode. Voici le schéma récapitulatif :

¹⁵ Voir chapitre concernant [les tags ID3](#)

Modèle

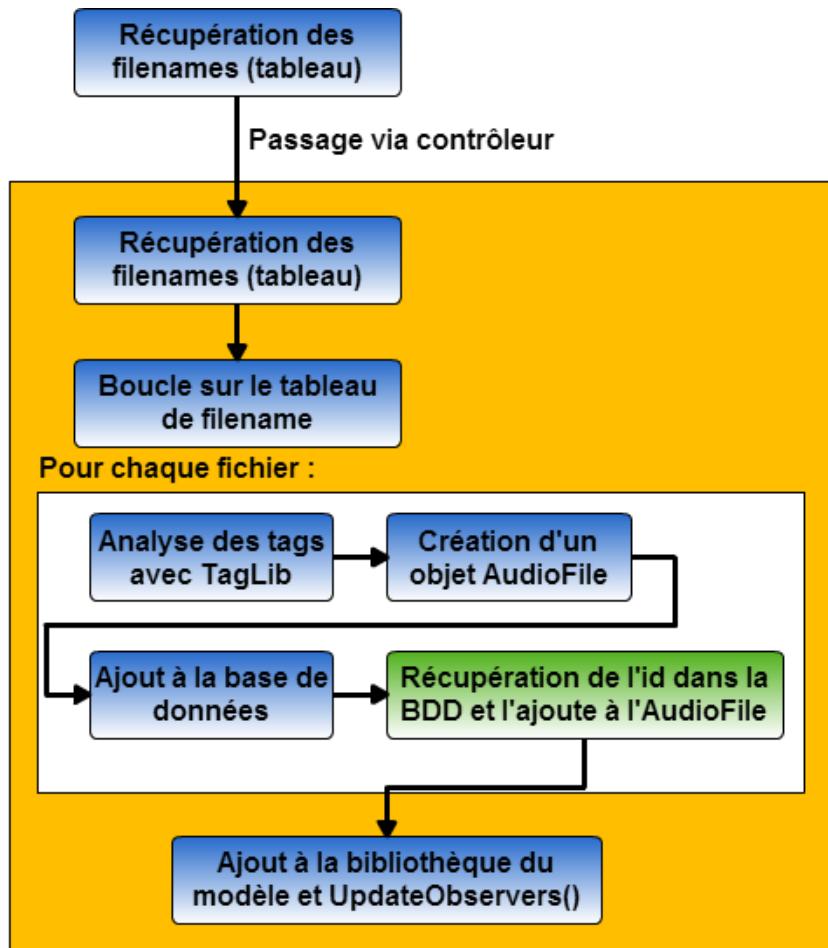


Figure 36 - Schéma importation fichier

La partie « analyse de tag » est expliquée plus précisément [ici](#). Concernant l'ajout à la base de données, [ce chapitre](#) décrit le procédé.

6.9.4 TAGS ID3

ID3 est le nom des métadonnées pouvant être insérées dans un fichier audio comme MP3. Ces métadonnées¹⁶ permettent d'avoir des informations sur le contenu du fichier comme le titre, le nom de l'interprète, les commentaires, ou encore la date de sortie.

Source : Wikipédia

6.9.5 ANALYSE DES TAGS

La récupération des tags ID3 est effectuée à l'aide de la bibliothèque « TagLib-Sharp » :

<https://github.com/mono/taglib-sharp>

¹⁶ Une métad donnée est une donnée servant à définir ou décrire une autre donnée quel que soit son support (papier ou électronique).

Le fonctionnement de cette bibliothèque est simple. Pour analyser et récupérer les tags ID3 d'un fichier, elle fournit une classe File (à ne pas confondre avec la classe File fournie par .NET) qui propose une méthode statique¹⁷ « Create » qui prend un filename en paramètre. Cette dernière retourne donc un objet instancié de type « TagLib.File » qui contient toutes les informations sur le fichier donné.

2 types d'informations importantes se distinguent et qui seront utilisées :

- La propriété « Tag » qui contient des sous-propriétés comme « Title », « Year », etc. Elles sont les tags à proprement parler et donnent des informations au sujet du contenu musical.
- La propriété « Properties » qui contient des sous-propriétés comme « Duration », « BitRate », etc. Ce sont les informations concernant le fichier en lui-même.

6.9.6 INDEXATION

Indexation est la partie qui consiste à enregistrer les informations des musiques de la bibliothèque dans la base de données. Pour se faire, quelques règles sont établies :

- Il ne peut y avoir qu'une occurrence par fichier. C'est-à-dire, pas de doublon. Plusieurs musiques peuvent avoir les mêmes informations dans les tags, mais c'est le nom de fichier qui fait foi.
- Chaque genre musical à son enregistrement dans la table « tgender ». Si lors de l'ajout d'une musique/pub à la base de données, son genre n'est pas encore dans cette table, il y est ajouté et son identifiant est récupéré. Dans l'autre cas, si le genre est déjà présent dans la table, son identifiant est juste trouvé dans la table. Si aucun genre n'est trouvé dans le tag, un genre vide lui est attribué.

Voici ces 2 règles représentées sous forme de code :

```
if(this.AudioFileExist(file.Filename))  
    return ERROR;  
  
int genderId = this.GetGenderId(file.Gender);  
//If return error, gender doesn't exist in DB, so add it  
if (genderId == ERROR)  
    //Get the new id  
    genderId = AddGender(file.Gender);
```

Bdd.cs

¹⁷ Méthode qui ne nécessite pas que sa classe soit instanciée pour être utilisée.

Ensuite, l'ajout s'effectue avec un nouvel enregistrement dans la table « tmusic » puis l'identifiant de ce nouvel enregistrement est retourné à la fin de la méthode.

6.9.7 MODIFICATION

Les colonnes « id », « duration » et « path » sont en lecture seule (propriété `ReadOnly`).

Lorsque l'utilisateur valide ses modifications, l'événement « `CellEndEdit` » du `DataGridView` en question est appelé. Un objet de type `Music` ou `Ad` (en fonction du `DataGridView` qui appelle l'événement) est créé avec les informations de la ligne modifiée (modifications comprises), puis la méthode « `UpdateAudioFile` » du *controller* est appelée. C'est ensuite dans le *model* que les modifications sont appliquées à la base de données, au fichier MP3 (tags) et à la bibliothèque contenus dans le *model* lui-même. Les étapes de modification s'effectuent comme ceci :

1. Mise à jour dans la base de données. Si la méthode « `UpdateAudioFile` » de la classe `Bdd` retourne vrai, passage à l'étape suivante.
 - a. La méthode située dans la classe `Bdd` effectue la même vérification que pour l'ajout de fichier audio concernant le genre. C'est-à-dire que si le genre inscrit n'existe pas, il est ajouté à la table `tgender`.
2. Modification des tags du fichier MP3 à l'aide de la bibliothèque C# `TagLib`.
3. Modification des informations dans le *model*
4. Mise à jour des observateurs

La mise à jour via `TagLib` s'effectue comme ceci :

```
TagLib.File tagFile = TagLib.File.Create(file.Filename);
tagFile.Tag.Title = file.Title;
tagFile.Tag.Performers = new string[] { file.Artist };
tagFile.Tag.Album = file.Album;
tagFile.Tag.Year = (uint)file.Year;
tagFile.Tag.Copyright = file.Label;
tagFile.Tag.Genres = new string[] { file.Gender };
tagFile.Save();
```

WMMModel.cs

6.9.8 SUPPRESSION

Pour la suppression, le même principe que l'importation par rapport à la différenciation entre le bouton « `Delete selected` » de la section « `Music` » et « `Ad` » avec la propriété « `Tag` ».

L'utilisateur peut supprimer plusieurs occurrences d'un seul coup, pour cela il faut parcourir la propriété « `SelectedRows` » du composant « `DataGridView` » en question. Voici la boucle :

```

foreach(DataGridViewRow row in ((type ==
AudioType.Music)?dgvMusics.SelectedRows:dgvAds.SelectedRows))
{
    if (!this.Controller.DeleteAudioFile(int.Parse(row.Cells[0].Value.ToString()),
row.Cells[row.Cells.Count - 1].Value.ToString()))
        state = false;
}

```

AdminView.cs

La présence d'un test ternaire¹⁸ dans le partie « in » du foreach permet de sélectionner le bon composant DataGridView en fonction de la section (Music ou Ad) dans laquelle le bouton de suppression a été pressé.

Pour chaque occurrence, la musique/pub va être supprimée de la base de données ainsi que du *model* et des playlists la concernant. Pour se faire, toutes les playlists de toutes les webradios du programme vont être bouclées et dans la liste de filename de chacune, les occurrences du filename de la musique/pub supprimée seront enlevées.

Ensuite, la suppression s'effectue dans la bibliothèque du *model* puis dans la base de données.

6.9.9 VÉRIFICATION DES DONNÉES

Dans le menu « File », le bouton « Check library » permet de vérifier l'intégrité de la bibliothèque. C'est-à-dire, supprimer les références vers des fichiers musicaux qui n'existeraient plus à l'emplacement auquel ils sont référencés. La méthode du *model* « CheckLibrary » parcourt toute la bibliothèque et vérifie que le filename stocké dans l'objet AudioFile référence bien un fichier existant. Si ce n'est pas le cas, l'enregistrement est supprimé de la base de données et du *model*.

6.9.10 RECHERCHE

La recherche consiste à afficher seulement les lignes ayant au moins une correspondance (n'importe quel champ de la ligne) avec la chaîne de recherche entrée par l'utilisateur. Cette chaîne est en premier temps mise en minuscule afin de ne pas prendre la casse en compte.

Une boucle parcourt les lignes du DataGridView et pour chacune, une autre boucle parcourt les cellules. Ensuite, pour chaque cellule, la valeur de cette dernière est prise, un « ToString » suivit d'un « ToLower » y est appliqué (pas de casse¹⁹) et enfin la méthode « Contains » va retourner un booléen si elle trouve une occurrence dans la valeur de la cellule. Si c'est le cas, cette ligne est valide et pourra être affichée. La propriété « Visible » de la ligne est donc modifiée à *true* mais dans le cas contraire elle sera *false*. Et ainsi de suite pour chaque ligne.

¹⁸ Un test ternaire utilise l'expression ternaire pour effectuer un test : test ? expression1 : expression2.

¹⁹ Pas de différence entre lettres majuscules et minuscules.

```
searchString = (sender as TextBox).Text.ToLower();

foreach(DataGridViewRow row in ((type == AudioType.Music)?dgvMusics.Rows:dgvAds.Rows))
{
    foreach(DataGridViewCell cell in row.Cells)
    {
        if (cell.Value.ToString().ToLower().Contains(searchString))
        {
            valid = true;
            break;
        }
    }
    row.Visible = (valid) ? true : false;
    valid = false;
}
```

AdminView.cs

6.10 LISTES DE LECTURE

Les listes de lecture sont utilisées par le calendrier, qui lui, est utilisé par un transcodér.

6.10.1 CLASSES UTILISÉES

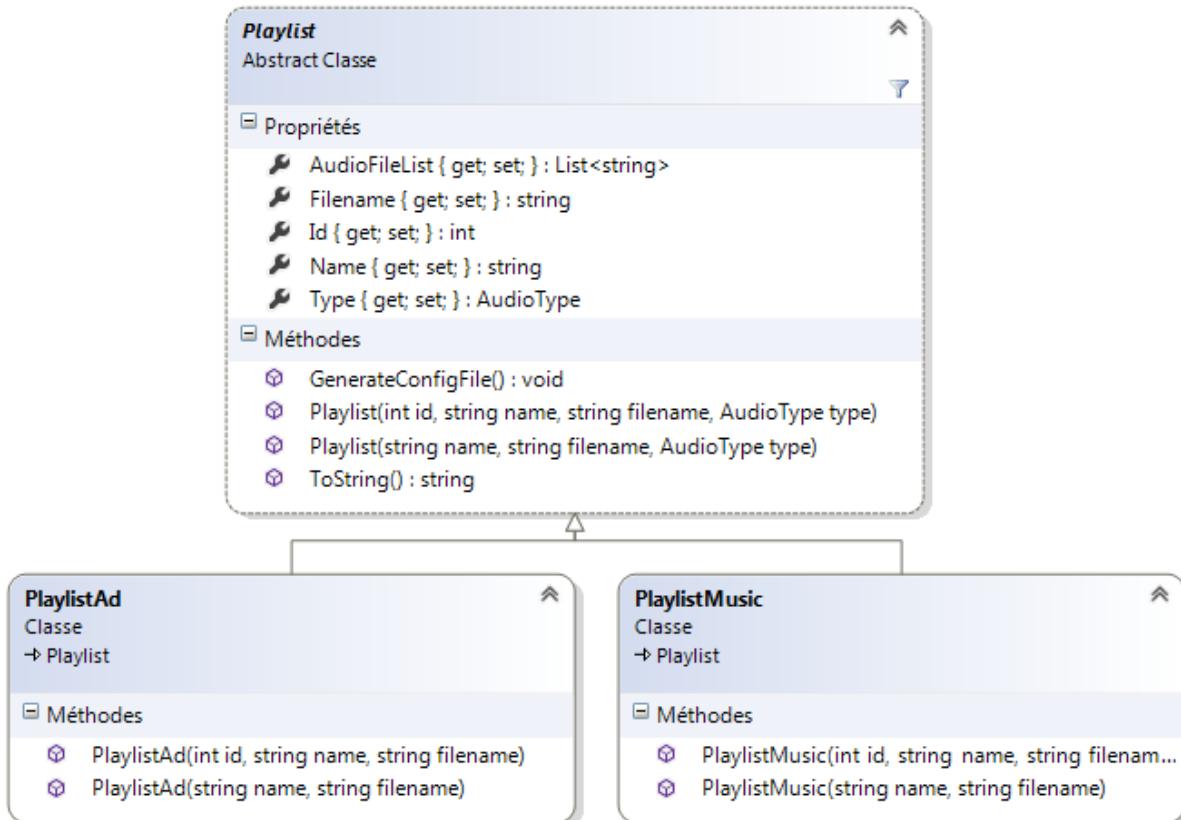


Figure 37 - Classes Playlist

6.10.2 GÉNÉRATION DE CONFIGURATION

Une liste de lecture comprend un fichier de configuration dont la forme est une simple liste des fichiers à utiliser. Elle est stockée dans un fichier texte avec l'extension « .lst ». Voici un exemple de fichier :

```

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Those_Were_The_Days_Original_Instrumental_Mix-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Everlasting_Outro-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz -
Wistfulness\Wasted_Penguinz_and_Toneshifterz-Together_Extended_Version-ToffMusic.mp3

C:\Users\MENETREYS_INFO\Music\test\Wasted Penguinz - Wistfulness\Wasted_Penguinz-
Falling_Extended_Version-ToffMusic.mp3

```

```
C:\Users\MENETREYS_INFO\Music\test\Wasted_Penguinz - Wistfulness\Wasted_Penguinz-  
Endless_Extended_Version-ToffMusic.mp3
```

Playlisttest.lst

Chaque ligne du fichier correspond au « filename » (chemin de fichier) vers la musique/pub de la playlist. C'est grâce à cela que le transcoder pourra aller chercher les musiques/pubs pour les jouer et les diffuser.

Concernant la génération, c'est le champ « AudioFileList » de la classe « Playlist » qui contient les filenames des musiques/pubs de la playlist, qui est parcouru afin de générer le fichier.

6.10.3 CRÉATION D'UNE PLAYLIST

Pour la création d'une nouvelle playlist, une règle importante a été établie :

- Le nom d'une playlist doit être unique (au sein de la même webradio et avec le même type (Music ou Ad)).
- Par exemple : Il peut y avoir 2 playlists qui se nomment « Test » au sein de la même webradio si chacune d'entre elles a un type différent.
- Le nom ne doit pas contenir de caractères Windows invalides. C'est-à-dire, des caractères interdits pour des noms de fichiers Windows.

Concernant la création à proprement parler, elle se décompose en plusieurs étapes qui se déroulent dans la méthode « CreatePlaylist » du *model* :

1. Création du futur filename du fichier de configuration de la playlist sous cette forme :
`DEFAULT_WEBRADIOS_FOLDER + webradioName + "/" + DEFAULT_PLAYLISTS_FOLDER + name + ".lst";`
2. Instanciation d'une classes PlaylistMusic ou PlaylistAd en fonction du type de playlist créée.
3. Insertion dans la base de données. La méthode d'ajout à la BDD retourne l'identifiant dans la BDD de la nouvelle playlist. Si cet id correspond « ERROR » (constante définie), une erreur est survenue lors de l'ajout. Dans ce cas, la méthode de création retourne directement « *false* ». Si un identifiant valide a été retourné, il est configuré à l'objet de type playlist instancié précédemment.
4. La méthode « [GenerateConfigFile](#) » est appelée sur l'objet Playlist afin de créer son fichier de configuration (bien que vide pour le moment).
5. L'objet est ajouté au *model* (dans la propriété « Playlists » de la webradio à laquelle la playlist est ajoutée).
6. « *UpdateObservers* » est appelé afin de mettre à jour toutes les fenêtres concernées.

6.10.4 GÉNÉRATION AUTOMATIQUE

Le principe de la génération automatique est de créer une playlist d'une durée donnée en la remplissant (aléatoirement) de musiques/pubs du genre donné. Dans le cas de pubs, le genre n'est pas pris en compte.

Il faut savoir que plus la durée demandée grande, plus il sera possible de s'en rapprocher le plus possible en utilisant les morceaux disponibles dans la bibliothèque.

Concernant l'algorithme, il consiste à parcourir la bibliothèque du *model* de façon aléatoire (avec l'objet de type Random) dans une boucle de type while. La condition de cette dernière est que la durée de la playlist doit rester plus petite que la durée demandée. Le but étant de se rapprocher le plus possible de la durée demandée sans la dépasser.

Il est déterminé que l'algorithme essaie un nombre de fois (consécutives), défini par la constante « MAX_TRY_GENERATE », de remplir la liste de lecture. Si ce nombre de fois est dépassé, il est considéré qu'il n'est plus possible de remplir sans dépasser la limite de durée et la boucle est donc arrêtée.

À chaque tour de boucle, une musique/pub est « piochée » et il est testé si le temps actuel de la playlist + le temps de la musique sélectionnée dépasse la durée demandée. Si c'est le cas, le nombre d'essais s'incrémente de 1 et la boucle refait un tour avec l'instruction « continue ». Si ce n'est pas le cas, le compteur d'essais est remis à zéro et la boucle continue son traitement sans interruption. Ensuite, un test vérifie si le type est Ad « ou » si le type est Music « et » du genre demandé, c'est en passant ce test que la musique/pub est ajoutée à la playlist ainsi que sa durée qui est additionnée à la durée actuelle de la playlist. Une liste d'entier est aussi utilisée pour stocker les identifiants des morceaux ajoutés à la playlist afin de pouvoir les utiliser lors de l'ajout à la base de données.

Voici un schéma récapitulatif de la méthode présente dans le *model* :

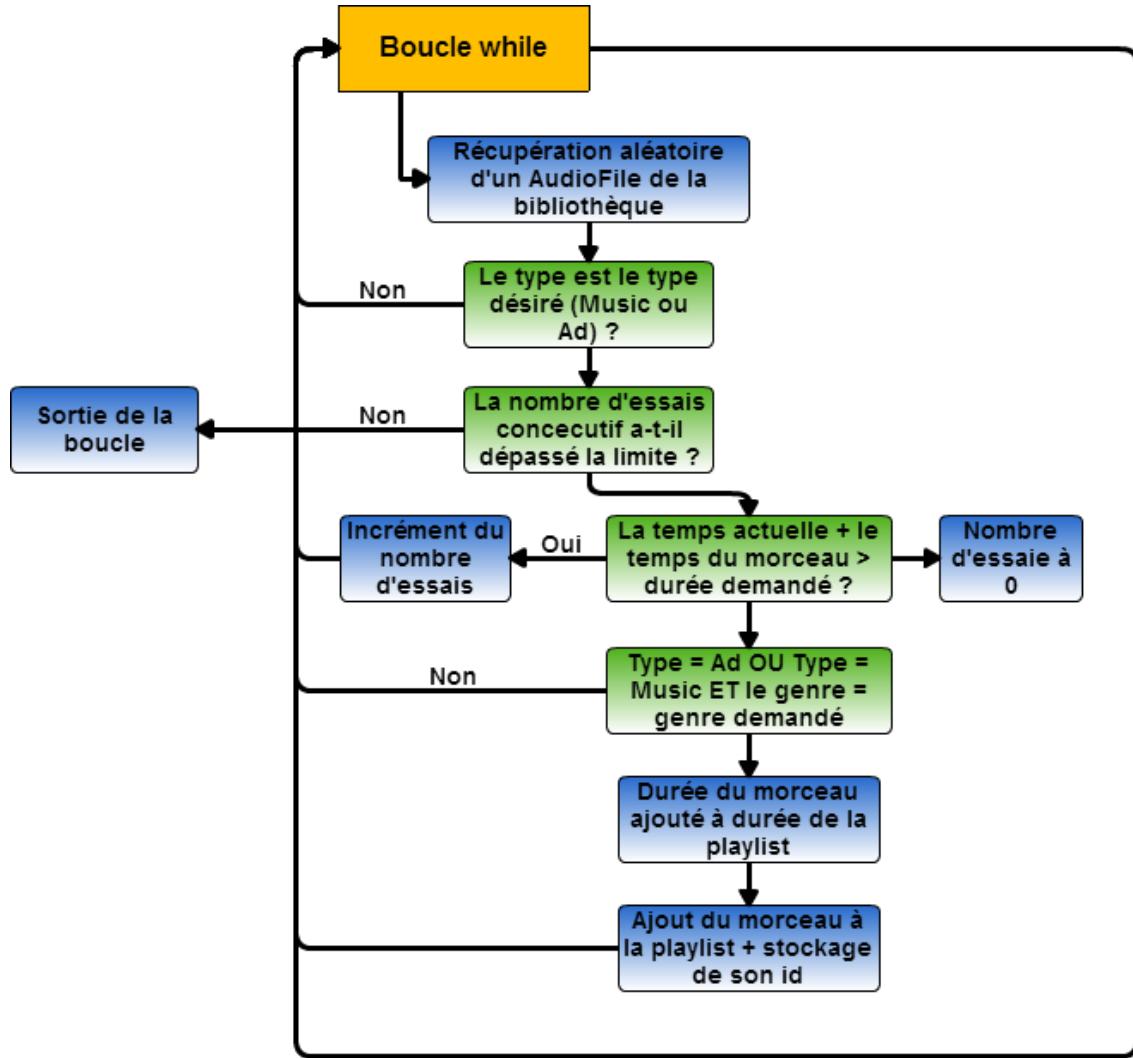


Figure 38 - Algorithme génération playlist

Au final, après la boucle, un test vérifie qu'il y a bien des musiques/pubs présentent dans la playlist. Si n'est pas le cas (l'algorithme n'a pas réussi à combler la durée malgré les essais), la méthode retourne *false*. Dans le cas contraire, la playlist est ajoutée à la base de données et au *model*. Pour finir, le fichier de configuration de la playlist est généré et la méthode « *UpdateObservers* » est appelée.

6.10.5 AJOUT À UNE PLAYLIST

L'ajout de musique/pub s'effectue depuis l'onglet « Library ».

Les 2 boutons d'ajout (celui dans la section Music ou la section Ad) appellent la même méthode d'événement. Pour différencier quel bouton appelle la méthode, la valeur « Music » ou « Ad » est inscrite dans la propriété « Tag » des boutons. Un test est effectué pour détecter quel bouton a été appuyé.

En premier temps, la *vue* va générer une liste de type *Dictionnaire<int,string>*. La clé (int) correspond à l'identifiant de la musique sélectionnée et la valeur (string) correspond à son filename. Ce

dictionnaire va permettre au *model* d'ajouter les différents éléments dans le *model* et la base de données. Il est généré à l'aide de la propriété « SelectedRows » du composant *dataGridView*. Cette propriété donne la liste des lignes sélectionnées par l'utilisateur.

Par la suite, le dictionnaire est envoyé au *model* via le *controller*. Il est parcouru par la méthode « AddToPlaylist » qui prend en paramètre un objet Playlist (correspondant à la playlist sélectionnée) et le dictionnaire. Pour chacun des éléments, il est ajouté à la base de données via la méthode de la classe Bdd « AddToPlaylist » qui prend en paramètre : la clé de l'événement (la valeur int qui correspond à l'id du morceau) et l'id de l'objet Playlist. Si cette méthode retourne une erreur, la boucle est arrêtée et le *model* retourne *false*. Sinon, la boucle se finit et la playlist génère sa configuration (GenerateConfigFile sur l'objet playlist).

6.10.6 RETIRER D'UNE PLAYLIST

La suppression d'éléments d'une playlist se déroule exactement comme l'ajout (dictionnaire avec les morceaux sélectionnés par l'utilisateur qui est parcouru par le *model*) à l'exception que les éléments seront supprimés de la base de données (suppression du lien dans la table « tplaylist_has_music ») et du *model*. À la fin, la nouvelle configuration est générée.

6.10.7 AFFICHAGE DU CONTENU

Lorsque l'utilisateur choisit un élément dans un des ListBox (section Music ou Ad), l'événement « SelectedIndexChanged » est appelé. Comme pour les autres éléments, chacun des ListBox à son type dans sa propriété Tag afin de différencier dans la méthode de l'événement en question.

La méthode privée de la vue « GetPlaylistContent » qui prend un objet de type Playlist en paramètre permet de vider le DataGridView servant d'affichage au contenu des playlists, puis de le remplir avec le contenu de la playlist voulue. Pour se faire, elle récupère une liste d'objet de type AudioFile en provenance du *model* via son *controller*. Le *model* va simplement parcourir sa bibliothèque et prendre les AudioFile dont le nom de fichier (filename) correspond à un des noms de fichier présent dans la playlist :

```
List<AudioFile> audioFiles = new List<AudioFile>();  
  
foreach(string filename in playlist.AudioFileList)  
{  
    foreach(AudioFile af in this.Library)  
    {  
        if (af.Filename == filename)  
            audioFiles.Add(af);  
    }  
}  
  
return audioFiles;
```

[AdminView.cs](#)

Ensuite, la *vue* va parcourir cette liste afin de remplir le DataGridView d'affichage.

Les classes Playlist disposent d'une méthode « GetInfosArray » qui retourne un tableau avec les informations de leurs champs. Cela permet de donner un tableau à la méthode d'ajout de ligne au dataGridView : `dgvPlaylistContent.Rows.Add(af.GetInfosArray());`
Par la même occasion, la durée de chaque morceau ajouté est additionnée à une variable de type TimeSpan afin de calculer la durée totale de la playlist et de l'afficher.

6.10.8 SUPPRESSION D'UNE PLAYLIST

La suppression de playlist va supprimer l'enregistrement correspondant dans la base de données, la variable dans le *model* ainsi que le fichier de configuration enregistré sur le disque. Les *vues* sont ensuite mises à jour via `UpdateObservers`.

6.11 GRILLE HORAIRE

6.11.1 CLASSES UTILISÉES

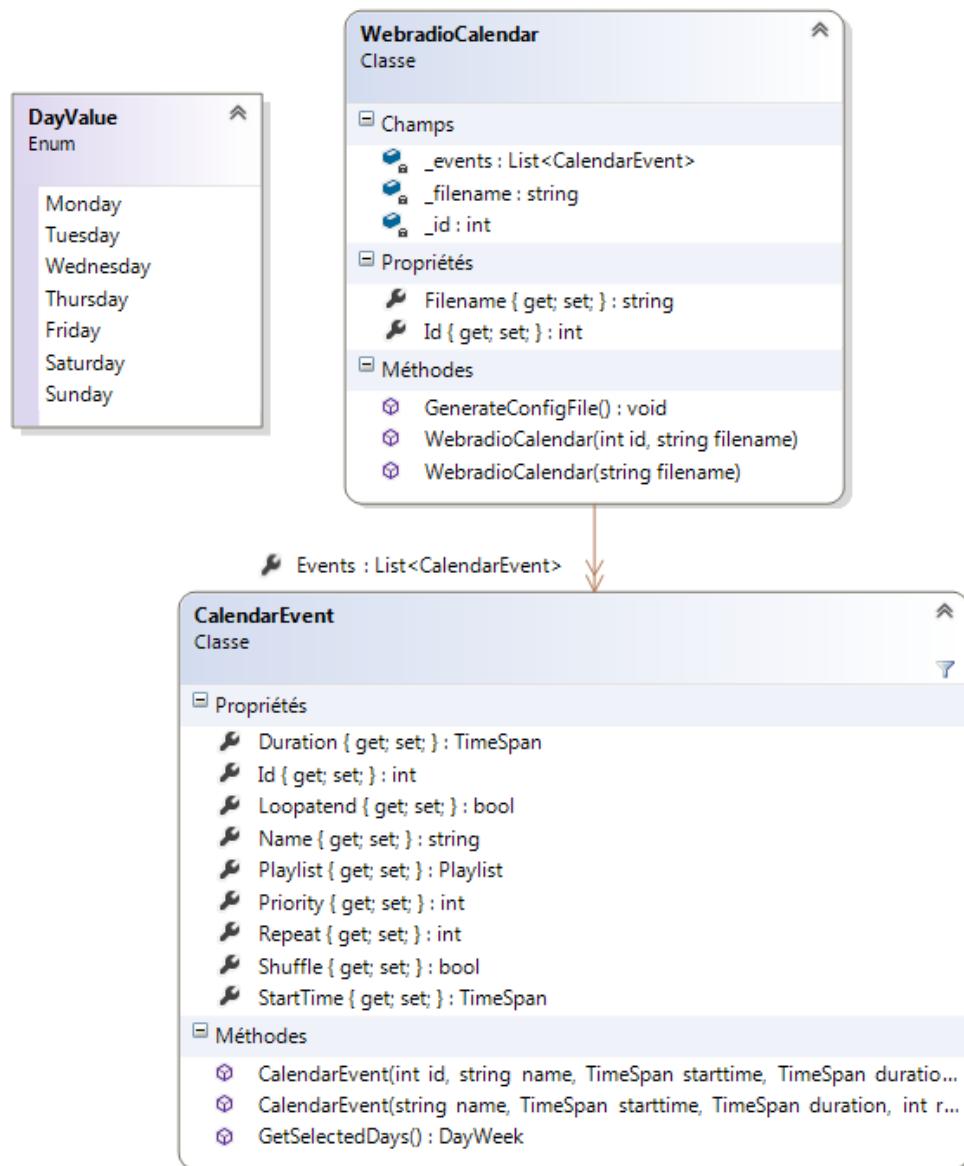


Figure 39 - Classes timetable

L'enum « DayValue » contient des constantes définies par ShoutCAST transcoder :

http://wiki.winamp.com/wiki/SHOUTcast_Calendar_Event_XML_File_Specification#Calendar_Tag

6.11.2 OUTIL UTILISÉ

Pour l'affichage et la gestion de la timetable²⁰, un composant tiers est utilisé :

<http://calendar.codeplex.com/>

Il s'agit de « Day View Calendar ». Ce composant a été choisi, car il propose une vue sous forme de jour et il est entièrement personnalisable (nombre de jours affichés, découpage des heures, etc.).

Voici un exemple d'utilisation :

	1 lundi	2 mardi	3 mercredi	4 jeudi	5 vendredi	6 samedi	7 dimanche
00 00							
01 00							
02 00	Pub(3) Pub migros Shuffle : False	Pub(3) Pub migros Shuffle : False	aaaaa (0) Comme	Pub(3) Pub migros Shuffle	Pub(3) Pub migros Shuffle : False	Pub(3) Pub migros Shuffle : False	Pub(3) Pub migros Shuffle : False
03 00							
04 00							
05 00							
06 00				feewe(1) HardLol Shuffle : True	feewe(1) HardLol Shuffle : True		
07 00							
08 00							

Figure 40 - Day View Calendar

6.11.3 GESTION DU CALENDRIER PAR SHOUTCAST

C'est le transcoder fournit par ShoutCAST qui se charge de prendre en compte le calendrier et jouer les playlists aux bons moments. Ce calendrier est sous forme XML et il est composé d'événements (event). Chaque événement est composé des propriétés suivantes (celles présentées sont celles utilisées dans l'application. Pour une description complète, rendez-vous si le site officiel :

http://wiki.winamp.com/wiki/SHOUTcast_Calendar_Event_XML_File_Specification) :

- Un type (playlist ou DJ) : Dans le cas de l'application, le type playlist sera toujours utilisé
- Une playlist : La lecture de la playlist dispose de plusieurs paramètres :
 - « loopatend » est un booléen qui définit si la playlist doit être rejouée quand tous les morceaux qui s'y trouvent ont été joués. Dans le cas de l'application, cette valeur est toujours « true ».
 - « shuffle » est un booléen qui définit si les morceaux de la playlist doivent être joués de façon aléatoire.
 - « priority » est une valeur entière non signée qui définit la hauteur de la priorité de l'événement. C'est-à-dire, si 2 événements sont prévus au même moment, celui avec la plus haute priorité est joué.
 - Le nom de la playlist jouée (nom du fichier .lst)
- Un horaire : L'horaire définit les paramètres suivants :

²⁰ Table du temps ou grille horaire en anglais.

- « starttime » est l'heure (format : hh:mm:ss) de début de l'événement.
- « duration » est la durée (format : hh:mm:ss) de lecture de la playlist.
- « repeat » est une valeur entière non signée servant au transcoder pour savoir quel jour doit être lancé cet événement aux horaires donnés. Chaque jour de la semaine a une valeur et l'addition des valeurs des jours sélectionnés donne la valeur de « repeat ». Voici les valeurs définies par ShoutCAST :
 - 1 - Every Sunday
 - 2 - Every Monday
 - 4 - Every Tuesday
 - 8 - Every Wednesday
 - 16 - Every Thursday
 - 32 - Every Friday
 - 64 - Every Saturday
 - 128 - Time periodic : Cette dernière n'est pas encore implémentée. Pour plus d'information, rendez-vous au lien ci-dessus (lien vers le site officiel).

6.11.4 GÉNÉRATION DE CONFIGURATION

ShoutCAST (transcoder) utilise un fichier XML pour la gestion de son calendrier. La génération d'un fichier XML sous C# se fait simplement avec un objet XmlDocument et des XmlElement. Voici un exemple de création de fichier de configuration XML pour un calendrier :

```
XmlDocument document = new XmlDocument();

XmlElement root = document.CreateElement("eventlist");

foreach(CalendarEvent ev in this.Events)

{

    XmlElement eventelement = document.CreateElement("event");

    eventelement.SetAttribute("type", "playlist");

    XmlElement playlist = document.CreateElement("playlist");

    playlist.SetAttribute("loopatend", (ev.Loopatend)? "1" : "0");

    playlist.SetAttribute("shuffle", (ev.Shuffle) ? "1" : "0");

    playlist.SetAttribute("priority", ev.Priority.ToString());

    playlist.InnerText = ev.Playlist;

    eventelement.AppendChild(playlist);

    XmlElement calendar = document.CreateElement("calendar");

    calendar.SetAttribute("starttime", ev.StartTime.ToString("hh:mm:ss"));

    calendar.SetAttribute("duration", ev.Duration.ToString("hh:mm:ss"));

    calendar.SetAttribute("repeat", ev.Repeat.ToString());
```

```

eventelement.AppendChild(calendar);

root.AppendChild(eventelement);

}

document.appendChild(root);

document.Save(this.Filename);

```

WebradioCalendar.cs

Voici le XML une fois généré :

```

<eventlist>
  <event type="playlist">
    <playlist loopatend="1" shuffle="1" priority="1">HardLol</playlist>
    <calendar starttime="06:00:00" duration="02:00:00" repeat="48" />
  </event>
  <event type="playlist">
    <playlist loopatend="0" shuffle="0" priority="0">Pub migros</playlist>
    <calendar starttime="01:00:00" duration="00:30:00" repeat="0" />
  </event>
  <event type="playlist">
    <playlist loopatend="0" shuffle="0" priority="0">Commercial</playlist>
    <calendar starttime="02:00:00" duration="01:30:00" repeat="8" />
  </event>
</eventlist>

```

Figure 41 - Exemple XML calendrier

6.11.5 AFFICHAGE DU CALENDRIER

L'affichage du calendrier se fait avec le composant décrit dans le chapitre concernant [l'outil utilisé](#) (0). Il a été configuré pour afficher 7 jours (une semaine) ainsi qu'un quadrillage basé sur les demi-heures.

Les événements sont affichés selon la convention suivante :

- Événement de type « Music » :
 - Couleur de bordure et de fond : Bleue
 - Contenu :
 - « *NomDeLevenement (HauteurPriorité)*
 - NomDeLaPlaylist*
 - Shuffle : *TrueOuFalse* »
- Événement de type « Ad » :
 - Couleur de bordure et de fond : Rouge
 - Contenu :
 - « *NomDeLevenement (HauteurPriorité)*
 - NomDeLaPlaylist*
 - Shuffle : *TrueOuFalse* »

C'est lors de l'appel de la méthode « UpdateView » que le calendrier sera rempli. Ce composant a besoin de disposer d'une variable de type `List<Appointment>` qu'il utilisera afin de se remplir. Cette liste doit donc être remplie lors d'UpdateView et elle est globale à la vue AdminView. C'est en réalité l'événement « `ResolveAppointments` » du composant qui va par la suite lire cette variable et remplir ce dernier :

```
List<Appointment> m_Apps = new List<Appointment>();
foreach (Appointment m_App in this.EventsCalendar)
    m_Apps.Add(m_App);

args.Appointments = m_Apps;
```

AdminView.cs

L'événement est appelé lorsque le composant doit se rafraîchir. Il permet de personnaliser la façon dont on veut remplir le composant.

Dans le cadre du projet, une classe nommée « `EventAppointment` » a été créée. Elle hérite de la classe « `Appointment` » proposée par le composant, mais elle propose une propriété de type `Playlist` afin de stocker l'objet `Playlist` lié à l'événement ainsi qu'une autre propriété de type `CalendarEvent` qui stocke l'objet `CalendarEvent` de l'événement. Cela est utile pour la suppression et modification. Cette classe reprend donc à l'identique la forme de sa classe parente, mais seules 2 propriétés sont ajoutées.

Comme décrit précédemment, ShoutCAST utilise une valeur nommée « `repeat` » pour stocker quels sont les jours où l'événement doit se jouer. Calculer cette valeur se fait à l'aide des valeurs données, dans la documentation, pour chaque jour. Mais lors de l'affichage dans le calendrier et la création des `EventAppointment`, il faut décoder cette valeur afin d'en ressortir les jours sélectionnés. Pour se faire, une méthode utilisant des masques binaires est utilisée. La classe `CalendarEvent` propose une méthode « `GetSelectedDays` » qui retourne une structure « `DayWeek` » qui contient tous les jours de la semaine sous forme de booléen. Voici un exemple d'utilisation d'un masque :

```
DayWeek dow = new DayWeek();
dow.Monday = Convert.ToBoolean(this.Repeat & MONDAY_MASK);
```

CalendarEvent.cs

La structure ainsi remplie va être récupérée lors d'UpdateView et utilisée de cette manière :

```
DayWeek dw = ev.GetSelectedDays();
bool[] days = dw.ToArray();
for(int i = 0; i < 7;i++)
{
    if(days[i])
```

```
{  
//Création d'un EventAppointment et ajout à la liste utilisée par le composant  
Calendar  
}  
}
```

AdminView.cs

Le principe est le suivant : une boucle est programmée pour tourner 7 fois (pour chaque jour de la semaine). Pour chaque tour (chaque jour), le programme teste si le jour actuel a une valeur booléenne vraie dans le tableau retourné par la structure DayWeek rempli à l'aide de la classe de l'événement courant.

Si la condition est vraie, un nouvel EventAppointment peut être créé, rempli et ajouté à la variable globale contenant les événements du calendrier.

6.11.6 SÉLECTION DEPUIS LE CALENDRIER

Le composant permet de faire une sélection (d'une heure à une autre heure) avec la souris. L'événement « SelectionChanged » est appelé si tel est le cas. Les informations du formulaire de création d'événements sont automatiquement mises à jour en fonction de la sélection de l'utilisateur.

6.11.7 CRÉATION D'UN ÉVÉNEMENT

Le formulaire de création est composé de 2 MaskedTextBox pour les champs « start time » et « duration ». Celle-ci permet de garantir que le format d'entrée de ces 2 propriétés sera juste afin de créer des TimeSpan. Le masque utilisé est le suivant « 00:00:00 ». Afin de récupérer ces valeurs, un autre test doit être effectué : vérifier que le format de l'heure est réaliste (par exemple, pas de cette façon : 89:20:67. C'est une heure qui n'est pas possible). La méthode statique « TryParse » de la classe TimeSpan permet cette vérification :

```
TimeSpan start = new TimeSpan();  
  
if (!TimeSpan.TryParse(mtbStartTime.Text, out start))  
{  
    MessageBox.Show("Start time format is not correct.", "Error");  
    return;  
}  
}
```

AdminView.cs

Si le test passe, la méthode remplira la variable « start » avec les valeurs du MaskedTextBox.

Une durée minimum de 1 minute est exigée pour un événement. Aussi, le nom de l'événement doit être unique.

La génération de la valeur « repeat » s'effectue en fonction des cases cochées (jours de la semaine) par l'utilisateur. La méthode « GetRepeatValue » présente dans la *vue* va calculer et retourner la valeur. L'enum DayValue fournit les valeurs de chaque jour (défini par ShoutCAST et défini [précédemment](#)) :

```
int repeat = 0;  
repeat += (ckbMonday.Checked) ? (int)DayValue.Monday : 0;
```

Et ainsi de suite pour chaque jour. Si la valeur rentrée est 0, c'est que l'utilisateur n'a sélectionné aucun jour. Un message d'erreur apparaît. La valeur « loopatend » est toujours vraie.

Ensuite, l'ajout à la base de données et au *model* s'effectue de la même façon que pour les autres éléments du logiciel (ajout à la bdd, récupération du nouvel identifiant, ajout à l'objet puis ajout au *model*). Pour finir, UpdateObservers pour mettre à jour les *vues*. La méthode GenerateConfigFile est aussi appelée pour mettre à jour le fichier XML du calendrier avec les nouvelles valeurs.

6.11.8 MODIFICATION D'UN ÉVÉNEMENT

Le composant Calendar permet de manipuler ses éléments avec la souris afin de les déplacer ou modifier leur longueur. Pour le moment, seuls la modification de la longueur, le commencement et les jours d'un événement sont modifiables (voir le [chapitre fonctionnel](#)). Les règles qui y sont décrites ont été faites pour correspondre le mieux possible avec le système de calendrier de ShoutCAST.

Lorsque l'utilisateur déplace un élément de l'événement et le change de jour, l'événement « MouseUp » du composant est appelé. En premier temps, il est testé s'il y a bien un élément (EventAppointment) sélectionné dans le composant. Puis, le traitement suivant est appliqué :

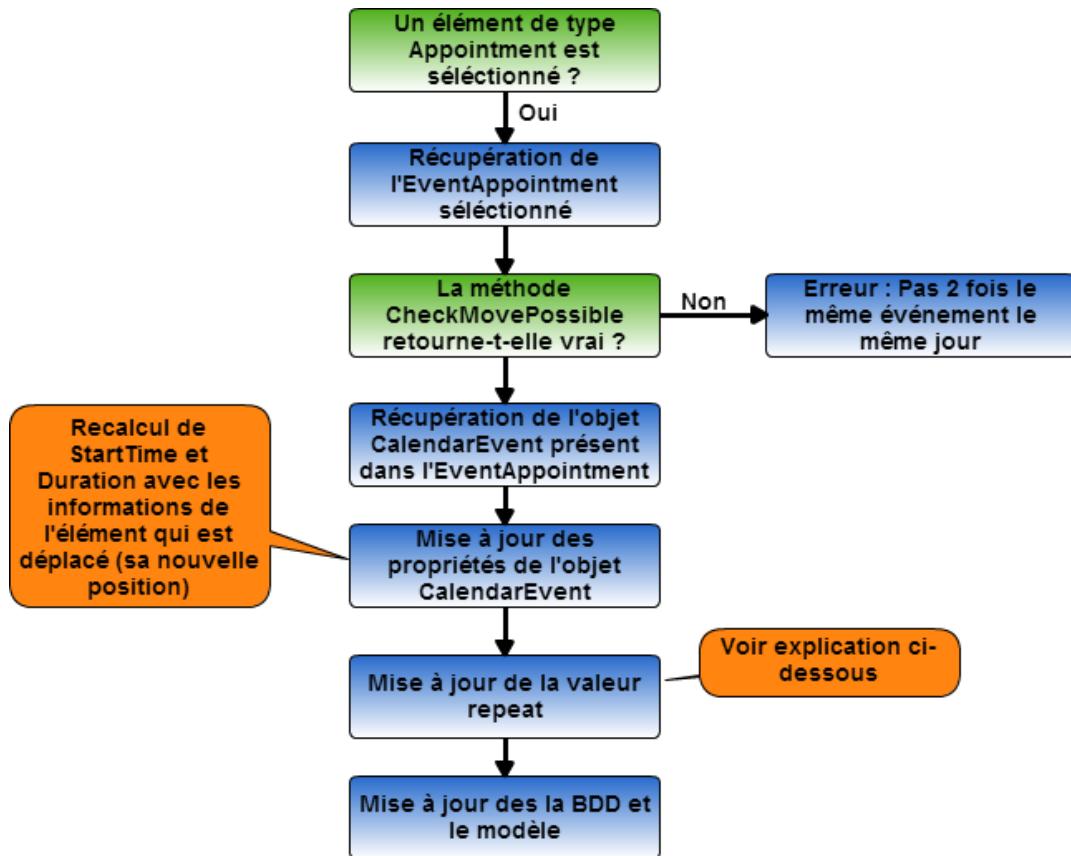


Figure 42 - Modification d'un événement

La méthode « CheckMovePossible » vérifie qu'il n'y a pas déjà un EventAppointment du même événement dans le jour où l'élément a été déplacé. Pour se faire, tous les éléments (EventAppointment) du composant sont vérifiés un par un. Si un d'entre eux à la même date de début (propriété DayOfWeek de celle-ci) et le même identifiant d'événement (stocké dans l'objet CalendarEvent de l'élément), le mouvement sera alors impossible.

Concernant la mise à jour de la valeur « repeat », elle est recalculée par rapport à l'emplacement des différents éléments d'un événement dans le composant. Pour se faire, plusieurs étapes sont exécutées :

1. Récupération de tous les éléments (EventAppointment) du composant qui sont concernés par l'événement en question (l'événement déplacé). C'est la méthode « GetAllRelatedApppointment » de la vue qui va rechercher dans la liste d'« EventAppointment » utilisée par le composant.
2. Ensuite, la méthode « GetRepeatValueFromAppointment » va calculer et retourner la valeur de repeat en fonction des « EventAppointment » qui lui sont donnés en paramètre. Pour chaque EventAppointment, il est vérifié (pour chaque jour de la semaine) si le « DayOfWeek » de sa propriété « StartDate » est égal au jour en question. De cette façon : `repeat += (ev.StartDate.DayOfWeek==DayOfWeek.Monday)?(int)DayValue.Monday : 0;`

Pour finir, les configurations sont régénérées.

6.11.9 SUPPRESSION D'UN ÉVÉNEMENT

Un clic droit sur un élément appelle l'événement « MouseClick » du composant. Il est vérifié que le bouton cliqué est bien le droit et si l'élément de type Appointment est bien sélectionné sur le composant. Une boîte de dialogue demande confirmation à l'utilisateur puis la méthode « DeleteEvent » du *controller* est appelé. L'objet CalendarEvent stocké dans l'EventAppointment est donné en paramètre. Cette méthode ira appeler la même méthode dans le *model* qui s'occupera, lui, de supprimer l'événement dans la BDD et ses propriétés. UpdateObservers est ensuite appelé ainsi que la régénération de configuration.

6.12 TRANSCODERS

6.12.1 CLASSES UTILISÉES

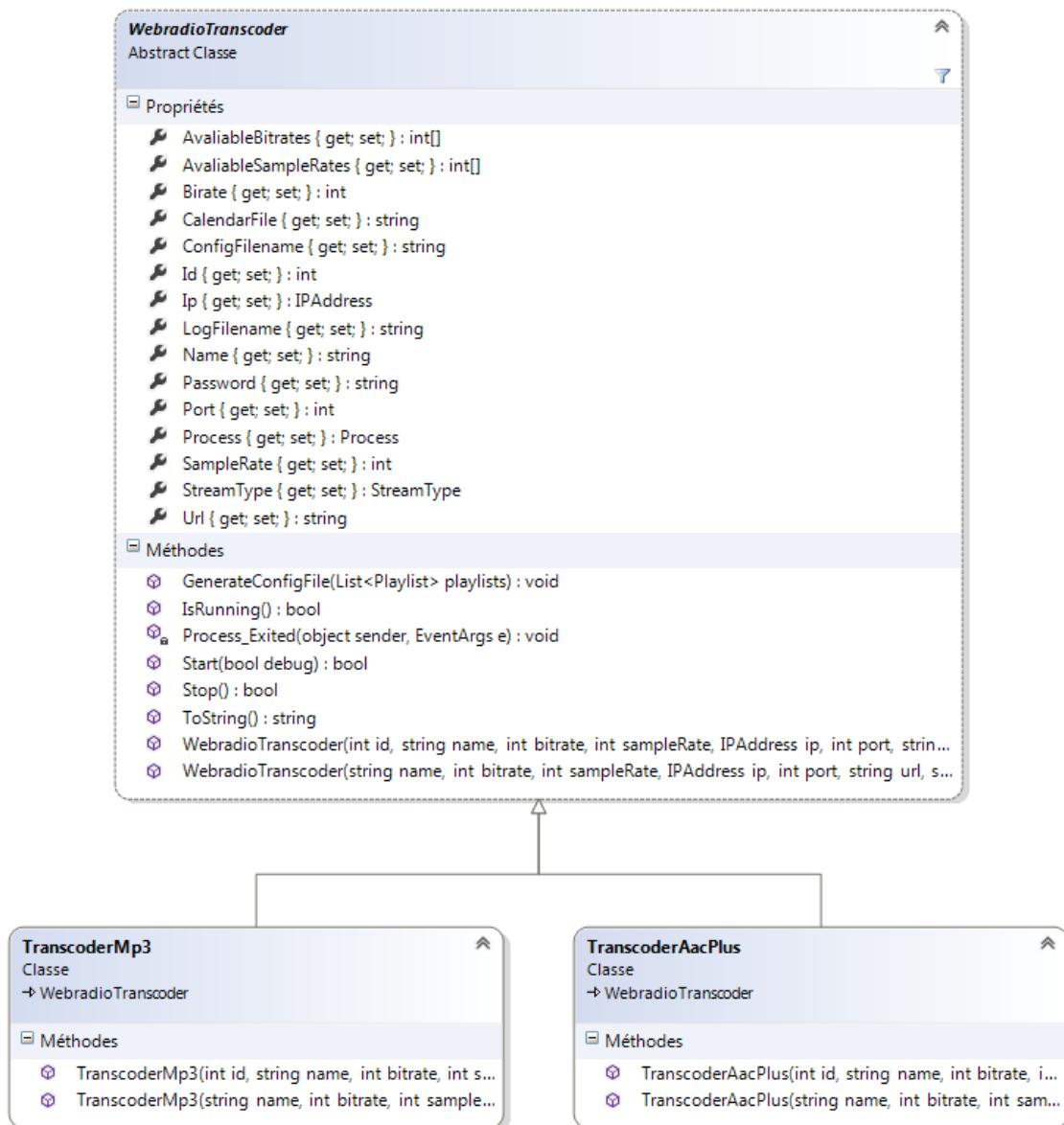


Figure 43 - Classes transcodeurs

6.12.2 OUTIL UTILISÉ

ShoutCAST propose un outil en lignes de commandes nommées « TransCAST » qui est un transcodeur :

Le transcodage, en vidéo ou en audio, est le fait de changer le format de codage d'un média (voir aussi codage et codec) utilisé pour comprimer ou encapsuler un média audio ou vidéo dans un fichier ; ou transporter un signal analogique ou numérique. On notera qu'il ne s'agit pas d'un codage au sens strict du terme, car le plus souvent la transformation comporte des pertes.

Source : Wikipédia

Dans le cas d'une webradio, le transcodeur sert à créer un flux à partir de fichiers audio afin de l'envoyer sur un serveur (serveur de diffusion) qui s'occupera de la diffusion aux clients/auditeurs. Un transcodeur peut disposer de playlists (listes de lecture) ainsi que d'un calendrier définissant des événements sur une semaine. Ces éléments ont été documentés précédemment. Le logiciel TransCAST fourni par ShoutCAST prend donc en charge toute la gestion de playlists, d'horaires et de lecture de musiques ainsi que le transcodage et la création du flux final.

Le lancement de cet outil se fait de la manière suivante :

```
sc_trans.exe myconfig.config
```

La première partie est le chemin vers l'exécutable. La seconde est le chemin vers le fichier de configuration. Voici la console du logiciel au lancement

```

2014-05-12 13:30:01 I msg:*****
2014-05-12 13:30:01 I msg:*** TRANScast Distributed Network Audio Content Provider
2014-05-12 13:30:01 I msg:*** Copyright (C) 2000-2011 Nullsoft, Inc. All Rights Reserved.
2014-05-12 13:30:01 I msg:*** Use "sc_trans filename.conf" to specify a config file.
2014-05-12 13:30:01 I msg:*****
2014-05-12 13:30:01 I msg:[MAIN] TRANSCast/win64 v2.0.0.54 (Oct 7 2011) starting...
2014-05-12 13:30:01 I msg:[MAIN] PID: 10688
2014-05-12 13:30:01 I msg:[MAIN] Loaded config from C:\Users\..._INFO\Documents\GitHub\WebradioManager\WebradioManager\bin\Debug\webradios\Gorilo\transcoders\8.config
2014-05-12 13:30:01 I msg:[MAIN] TimeMultiplier = 1, TimeShift = 0
2014-05-12 13:30:01 I msg:[CALENDARMGR] Adding playlist event: m_playlist=HardLol m_loopAtEnd=0 m_priority=0 m_shuffle=0 m_id=1 m_startDate=year:0,mon:0,day:0,hour:8,min:0,sec:0, isdst:0,wday:0,yday:0 m_endDate=year:0,mon:0,day:0,hour:0,min:0,sec:0, isdst:0,wday:0,yday:0 m_duration=28800 m_timeOffset=year:0,mon:0,day:0,hour:0,min:0,sec:0, isdst:0,wday:0,yday:0 m_hasStartDate=0 m_hasEndDate=0 m_hasDuration=1 m_hasTimeOffset=1 m_repeat=127
2014-05-12 13:30:01 I msg:[VUPUSH] 0 VU images loaded for left channel
2014-05-12 13:30:01 I msg:[VUPUSH] 0 VU images loaded for right channel
2014-05-12 13:30:01 I msg:[PLAYLISTMGR] Playlist load from file
2014-05-12 13:30:01 E msg:[PLAYLISTMGR] Could not activate playlist ` because no playlist file found
2014-05-12 13:30:01 I msg:[MAIN] Streaming thread starting
2014-05-12 13:30:01 I msg:[RESAMPLER] Deactivated 44100/2 == 44100/2
2014-05-12 13:30:01 I msg:[SOURCEANDENDPOINTMANAGER] Encoder thread 1 starting

```

Figure 44 - ShoutCAST transcoder console

Cet outil propose une API²¹ web accessible via le port défini dans le [fichier de configuration](#) (adminport). Certaines informations peuvent être récupérées et modifiées. Plus d'informations sur le site officiel : http://wiki.winamp.com/wiki/SHOUTcast_Transcoder_AJAX_api_Specification

Pour le projet, le protocole Ultravox 2.1 a été sélectionné, car il s'agit de la dernière version et la plus stable. Il est évidemment possible de changer cette valeur. Le transcoder ShoutCAST propose 3 protocoles différents. Chacun a une valeur à configurer dans le fichier de configuration :

```
1 = SHOUTcast 1 (Legacy)  
2 = Ultravox (Ultravox 2.0)  
3 = SHOUTcast 2 (Ultravox 2.1)
```

Plus d'informations sur cette page :

[http://wiki.winamp.com/wiki/SHOUTcast_2_\(Ultravox_2.1\)_Protocol_Details](http://wiki.winamp.com/wiki/SHOUTcast_2_(Ultravox_2.1)_Protocol_Details)

6.12.3 DÉFINITION DES BITRATES, TAUX D'ÉCHANTILLONNAGE ET TYPE D'ENCODER

Les types d'encodeur sont définis par un énumérateur « StreamType » :

```
public enum StreamType  
{  
    MP3 = 1,  
    AACplus = 2,  
}
```

StreamType.cs

Les valeurs assignées sont statiques et sont identiques aux identifiants de ces valeurs dans la base de données (table tstreamtype).

Concernant les bitrates et les taux d'échantillonnage, 2 tableaux statiques sont définis dans la classe « WebradioTranscoder » :

```
private static int[] _availableBitrates = { 64000, 96000, 128000, 256000 };  
private static int[] _availableSampleRates = { 44100 };
```

WebradioTranscoder.cs

6.12.4 FICHIER DE CONFIGURATION ET LOG

²¹ Application Programming Interface (interface de programmation) : un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

Voici la liste des paramètres utilisés et configurés dans le fichier de configuration d'un transcoder:

- Logfile : Le chemin vers le fichier de log
- Encoder_1 : Le type d'encodeur (mp3 ou aacp)
- Bitrate_1 : Le bitrate de l'encoder
- Adminimport : Le port pour accéder au transcoder (pour envoi de requêtes Ajax²²). Cette valeur est définie avec la valeur constante « DEFAULT_ADMIN_PASSWORD ». Actuellement, elle vaut 9000.
- Adminuser : Nom d'utilisateur pour les requêtes
- Adminpassword : Mot de passe pour les requêtes
- Outprotocol_1 : Sélection de protocoles utilisés pour le flux. Toujours configuré à 3, car cela correspond au protocole le plus récent : SHOUTcast 2 (Ultravox 2.1). Le serveur distant doit prendre en charge ce protocole
- Serverip_1 : Adresse IP du serveur de diffusion
- Serverport_1 : Port du serveur de diffusion
- Password_1 : Mot de passe du serveur de diffusion
- Streamid_1 : défini l'id du flux
- Streamtitle : Le nom du flux
- Streamurl : L'adresse du flux (l'adresse du site web de la webradio par exemple)
- Genre : Le genre de musique du flux. Toujours configuré à « Misc ».
- Calendarfile : Le chemin vers le fichier de calendrier de la webradio (tous les transcodeurs d'une même webradio pointent sur le même calendrier)

À savoir que le numéro écrit comme ceci « _X » après certains paramètres est un numéro qui permet de regrouper tous les paramètres à l'encoder de ce numéro (défini par le paramètre « streamid »). Car un transcoder peut, dans certains cas, avoir différents encodeurs et serveur de diffusion. Dans le cas de WebradioManager, il a été décidé qu'un transcoder avait un seul serveur de diffusion.

En plus de ces paramètres, il faut que le fichier de configuration contienne les playlists utilisées par le calendrier défini. Ces informations sont par couple de 2 paramètres :

- Playlistfilename_X : Le nom de la playlist (le nom qui est inscrit [dans le calendrier](#))
- Playlistfilepath_X : Chemin vers le fichier de cette playlist

Le X correspond à un numéro qui permet de retrouver les 2 paramètres qui vont de pair. C'est-à-dire que chacun des 2 paramètres doit avoir le même numéro. Pour générer ces paramètres, un paramètre de type « List<Playlist> » est passé en paramètre à la méthode de génération de configuration.

Les fichiers de configuration sont stockés dans le dossier « transcoders » de la webradio (voir le chapitre concernant [les dossiers/fichiers \(0\)](#)).

²² Voir [glossaire](#)

Chaque fichier est nommé « IdentifiantDuTranscoder.config » (identifiant dans la base de données). La même chose pour son fichier de log, mais avec l'extension « .log »

Exemple de configuration :

```
logfile=C:\Users\MENETREYS_INFO\Documents\GitHub\WebradioManager\WebradioManager\Web  
radioManager\bin\Debug\webradios\Gorilo\transcoders\7.log  
  
encoder_1=aacp  
  
bitrate_1=256000  
  
adminport=9000  
  
adminuser=admin  
  
adminpassword=admin  
  
outprotocol_1=3  
  
serverip_1=127.0.0.1  
  
serverport_1=8000  
  
password_1=lol  
  
streamid_1=1  
  
streamtitle=Gorilo 256  
  
streamurl=www.hardstylefm.ch  
  
genre=Misc  
  
calendarfile=C:\Users\MENETREYS_INFO\Documents\GitHub\WebradioManager\WebradioManager  
\WebradioManager\bin\Debug\webradios\Gorilo\calendar.xml  
  
playlistfilename_1=Pub migros  
  
playlistfilepath_1=C:\Users\MENETREYS_INFO\Documents\GitHub\WebradioManager\WebradioMa  
nager\WebradioManager\bin\Debug\webradios\Gorilo\playlists\Pub migros.lst  
  
playlistfilename_2=Commercial  
  
playlistfilepath_2=C:\Users\MENETREYS_INFO\Documents\GitHub\WebradioManager\WebradioMa  
nager\WebradioManager\bin\Debug\webradios\Gorilo\playlists\Commercial.lst  
  
playlistfilename_3=HardLol  
  
playlistfilepath_3=C:\Users\MENETREYS_INFO\Documents\GitHub\WebradioManager\WebradioMa  
nager\WebradioManager\bin\Debug\webradios\Gorilo\playlists\HardLol.lst  
  
playlistfilename_4=Test  
  
playlistfilepath_4=C:\Users\MENETREYS_INFO\Documents\GitHub\WebradioManager\WebradioMa  
nager\WebradioManager\bin\Debug\webradios\Gorilo\playlists\Test.lst
```

6.12.5 LICENCE MP3

Pour diffuser en mp3, l'utilisateur doit obtenir une licence comme expliquée dans la documentation du transcodeur ShoutCAST :

http://wiki.winamp.com/wiki/SHOUTcast_DNAS_Transcoder_2#Registering_for_MP3_Stream_Encoding

Malheureusement, les licences ne sont plus distribuées pour le moment. L'utilisateur peut tout de même sélectionner MP3, mais doit entrer manuellement sa licence dans le fichier de configuration si il en possède une. De ce fait, la diffusion en MP3 n'est pas fonctionnelle à 100%.

6.12.6 CRÉATION D'UN TRANSCODER

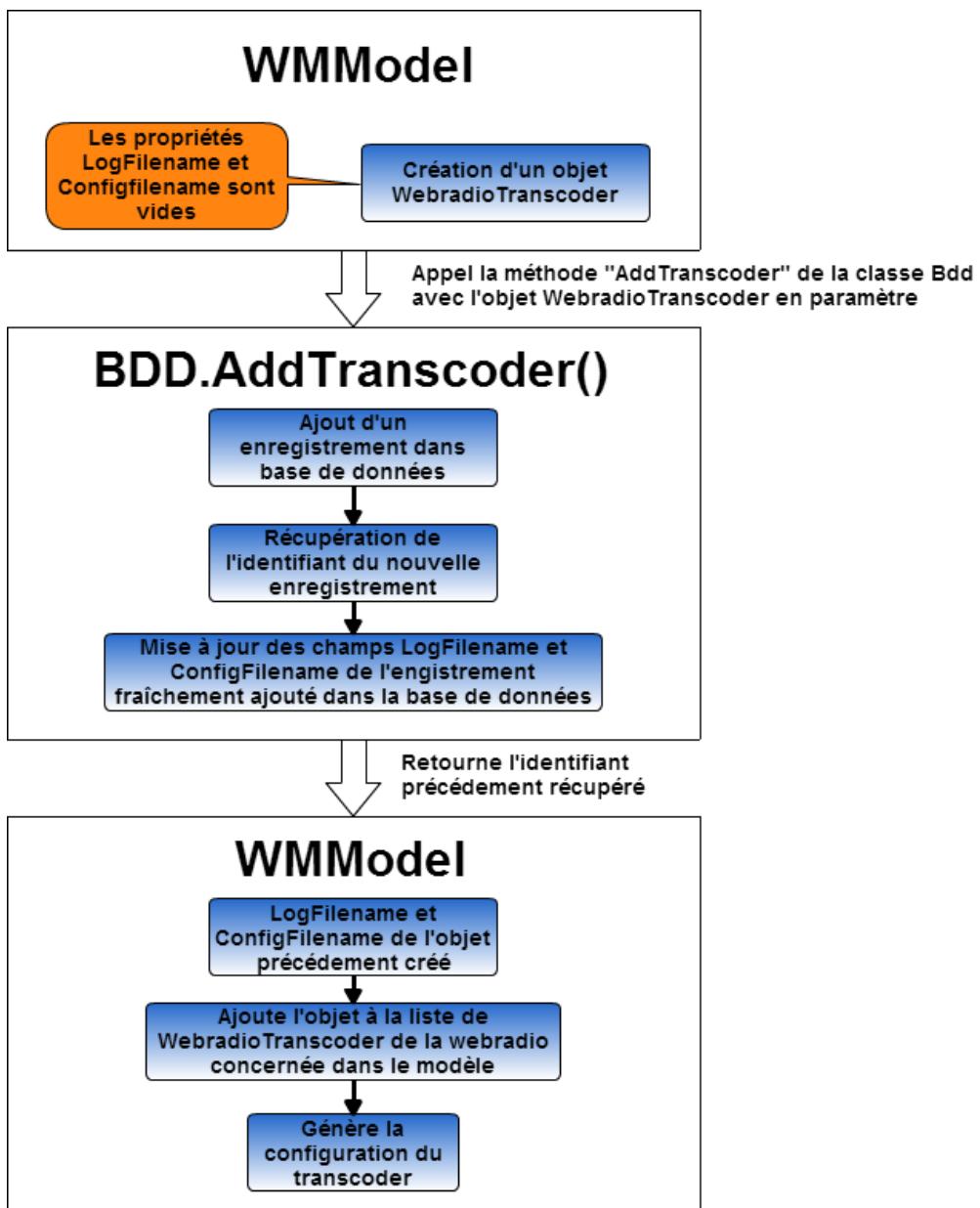


Figure 45 - Schéma création d'un transcodér

Le schéma ci-dessus explique le fonctionnement global de l'ajout d'un nouveau transcoder à une webradio au niveau du *model*. En amont, certaines vérifications sont effectuées au niveau de la *vue*. Pour commencer, les champs concernant le nom du flux et le mot de passe du serveur de diffusion doivent être remplis. L'adresse IP rentrée doit être correcte et elle est vérifiée avec la méthode « TryParse » de la classe IPAddress. L'utilisateur peut aussi entrer le nom DNS du serveur. Dans ce cas, le TryParse va retourner *false*.

Ensuite, dans la méthode « AddTranscoder » de la classe Bdd décrite dans le schéma, un premier test fondamental est effectué avant toute manipulation. Le transcodeur créé doit avoir un nom unique (nom du flux) pour cette webradio. La méthode « TranscoderExists » de la même classe retourne un booléen. Si le nom du flux existe déjà, la méthode retourne directement la valeur constante « ERROR ». De son côté, le *model* va recevoir l'identifiant du transcoder qui a été sensé être ajouté. Il vérifie donc que l'id reçu n'est pas égale à la valeur constante « ERROR » de la classe Bdd. Si elle est égale, le *model* retourne « *false* » et la *vue*, dans ce cas, affiche un message d'erreur à l'utilisateur.

6.12.7 RÉSOLUTION DE NOM

Les 2 boutons « resolve » permettent de résoudre le nom DNS donné par l'utilisateur dans le champ « Server IP ». La méthode « GetResolvedConnectionIPAddress » de la *vue* AdminView (trouvée sur le site : <http://www.codeproject.com/Tips/440861/Resolving-a-hostname-in-Csharp-and-retrieving-IP-v>) va retourner un booléen si l'opération a réussi ou non. La variable « *out* » de type IPAddress sera remplie avec une nouvelle instance contenant l'adresse trouvée.

6.12.8 AFFICHAGE D'UN TRANSCODER

Lors de la sélection d'un transcoder dans la liste proposée située dans l'onglet « transcoders », la méthode de la *vue* nommée « ShowTranscoderInfos » va se charger de remplir la partie d'édition de transcodeur (partie de droite de l'onglet) avec les informations du transcoder sélectionné.

Le reste des informations sont emplies de façon standard. La partie « status » affichant l'état du transcoder sélectionné est aussi remplie :

```
bool running = transcoder.IsRunning();

lblStatusTranscoder.Text = (running) ? "On" : "Off";
lblStatusTranscoder.ForeColor = (running) ? Color.Green : Color.Red;
btnStartTranscoder.Enabled = (running) ? false : true;
btnStopTranscoder.Enabled = (running) ? true : false;
```

AdminView.cs

6.12.9 MODIFICATION D'UN TRANSCODER

Pour la mise à jour des informations d'un transcodeur, la procédure ressemble à celle de l'ajout. Les champs concernant le nom et le mot de passe du serveur de diffusion doivent être remplis et l'adresse IP rentrée doit être valide. Ensuite, le transcodeur sélectionné dans le ListBox est récupéré

et les informations contenues dans ses propriétés sont mises à jour avec celles du formulaire de modification. L'objet modifié est ensuite envoyé au *model* via le *controller* afin d'apporter les modifications dans la base de données et les données du *model*.

Si le transcoder était en fonctionnement, il est arrêté et sera redémarré après la mise à jour des informations et la régénération de la configuration.

6.12.10 SUPPRESSION D'UN TRANSCODER

Lors de la suppression d'un transcoder, celui sélectionné dans le ListBox (affichant les transcodeurs disponibles) est envoyé au *model* via le *controller*. Les informations concernant le transcodeur sont supprimées de la base de données, puis, les fichiers (configuration et log) sont supprimés et pour finir, l'objet est supprimé du *model*.

6.12.11 EXÉCUTION ET FERMETURE

La classe WebradioTranscoder propose 3 méthodes concernant la gestion du processus du transcodeur qu'elle représente :

- Start : Lance le processus
- Stop : Arrête le processus
- IsRunning : Retourne un booléen. « *True* » si le processus est en fonctionnement et « *false* » dans le cas contraire.

Le lancement d'un transcoder peut se faire en mode « debug ». C'est le paramètre booléen de la méthode « start » qui définit si ce mode est activé ou non. Si il est actif, le processus se lance avec une fenêtre console affichant le log du transcoder qui est directement minimisée dans la barre de tâche. Sans ce mode, aucune fenêtre ne s'affiche. Voici la préparation au lancement du processus grâce à la classe ProcessStartInfo (voir aussi le chapitre concernant le [serveur](#)) :

```
ProcessStartInfo StartInfo = new ProcessStartInfo(Directory.GetCurrentDirectory() +  
SC_SERVER_FILENAME)  
{  
    CreateNoWindow = true,  
    WindowStyle =  
(debug)?ProcessWindowStyle.Minimized:ProcessWindowStyle.Hidden,  
    Arguments = Directory.GetCurrentDirectory() + "\\\" +  
this.ConfigFilename.Replace('/', '\\')  
};
```

WebradioTranscoder.cs

Pour l'arrêt d'un processus, la méthode « stop » va en premier voir si le processus fonctionne et répond toujours en tâche de fond. Si c'est le cas, le processus est arrêté. Le calendrier est à nouveau généré, car il a été remarqué que ce dernier était modifié par le transcoder lui-même lors de l'arrêt du processus.

Deux cas d'erreur au lancement sont possibles :

- Un processus « fantôme » de ce transcodeur est encore dans les processus système et le programme n'arrive pas à l'arrêter. Dans ce cas, l'utilisateur doit aller arrêter le processus lui-même (via le gestionnaire de tâche windows).
- Le fichier exécutable ShoutCAST « sc_trans.exe » n'est plus disponible.

Pour plus d'informations concernant la gestion des différents processus, rendez-vous au chapitre « [gestion des processus](#) ».

6.12.12 ADMINISTRATION WEB (WEBLET)

La fonctionnalité « next track » permet de passer à la musique suivante dans la playlist qui est jouée par le transcodeur sélectionné. Cela est possible grâce à l'API Ajax proposée par TransCAST. Voici un exemple de création d'une requête vers l'API :

```
 WebClient wb = new WebClient();

var data = new NameValueCollection();

data["op"] = "nexttrack";

data["seq"] = "45";

wb.Credentials = new NetworkCredential(WebradioTranscoder.DEFAULT_ADMIN,
WebradioTranscoder.DEFAULT_ADMIN_PASSWORD );

var response = wb.UploadValues("http://127.0.0.1:"+this.AdminPort+"/api", "POST",
data);
```

WebradioTranscoder.cs

Chaque action possible sur l'API se compose de la même façon. C'est une requête de type POST sur l'adresse de l'API du transcodeur (IP + PortAdministration /api). Cette requête se compose de 2 valeurs : « op » qui correspond au nom de l'opération et « seq » qui correspond à une valeur (parfois utile pour l'action et parfois non). Il faut être authentifié pour utiliser l'API, c'est pour cela que la requête est effectuée avec des *credentials*²³.

L'exemple ci-dessus permet de dire au transcodeur de passer à la musique suivante de la playlist actuellement jouée.

6.12.13 CAPTURE LIVE

Afin d'effectuer une capture live, le transcodeur ShoutCAST a besoin de savoir quel périphérique audio utiliser pour la capture. La méthode « UpdateAudioDevices » de la *vue* permet de lister les périphériques audio de l'ordinateur dans le ComboBox prévu à cet effet :

²³ Un couple nom d'utilisateur/mot de passe servant à accéder à une ressource réseau.

```
ManagementObjectSearcher objSearcher = new ManagementObjectSearcher("SELECT * FROM Win32_SoundDevice");

ManagementObjectCollection objCollection = objSearcher.Get();

cmbAudioDevice.Items.Clear();

foreach (ManagementObject obj in objCollection)

{

    foreach (PropertyData property in obj.Properties)

    {

        if (property.Name == "Caption")

            cmbAudioDevice.Items.Add(property.Value);

    }

}

}
```

AdminView.cs

Ce code a été trouvé ici : <http://stackoverflow.com/questions/1525320/how-to-enumerate-audio-out-devices-in-c-sharp>

L'API Ajax du transcoder permet d'activer ou de désactiver la capture pendant l'exécution de ce dernier ainsi que de configurer le périphérique audio. La classe WebradioTranscoder dispose d'une méthode « SetCaptureMode » prenant un booléen qui détermine si cette fonctionnalité doit être activée ou non, ainsi qu'une chaîne de caractères contenant le nom du périphérique cible, en paramètres.

Plus d'informations sur l'activation via l'API :

http://wiki.winamp.com/wiki/SHOUTcast_Transcoder_AJAX_api_Specification#Capture

Cette fonctionnalité ne semble pas tout à fait bien fonctionner. En effet, il se trouve que sur l'ordinateur de développement, malgré le listing des périphériques audio, le transcoder ne trouve pas ce dernier lors de l'activation de la capture.

6.12.14 HISTORIQUE

L'historique de chaque transcodeur est enregistré dans la base de données (table thistory). L'historique référence quelle musique a été jouée par quel transcodeur à quelle date et heure.

Le serveur ShoutCAST propose un historique des morceaux joués (les 10 derniers maximum), mais dans le cas de ce projet, il est préférable de regarder au niveau du transcodeur, quelles musiques ont été jouées, car dans le cas d'une diffusion vers un serveur de diffusion distant, le logiciel n'a aucun contrôle sur ce dernier (que ce soit pour l'accès aux données ou alors le fonctionnement du serveur). Malheureusement, les transcodeurs ShoutCAST ne proposent pas d'historique. Une solution alternative a été mise en place afin de détecter un changement de fichier audio sur un transcodeur afin de noter ses informations dans la BDD.

La méthode appelée par le timer servant de surveillance des processus (voir le chapitre concernant la [Gestion des processus](#) 6.14) est utilisée afin de vérifier le statut de chaque transcoder. Le statut de ces derniers est récupéré à l'aide de l'API Ajax (voir le [chapitre précédent](#) pour l'envoi de commande) avec la commande « getstatus ». Cette dernière retourne les informations suivantes (XML) :

```
<status>

    <activesource source="playlist|capture|dj|relay">

        <currenttrack/>

        <nexttrack/>

        <name/>

        <file/>

        <ip/>

        <port/>

        <url/>

        <sourcetype/>

        <bitrate/>

        <device/>

        <input/>

        <samplerate/>

        <channels/>

    </activesource>

    <endpointlist>

        <endpoint>

            <name/>

            <bytessent/>

            <status/>

        </endpoint>

        <endpoint>

            <name/>

            <bytessent/>

            <status/>

        </endpoint>

    </endpointlist>

</status>
```

```

</endpoint>
</endpointlist>
</status>

```

Les différents champs sont décrits dans la documentation officielle :

http://wiki.winamp.com/wiki/SHOUTcast_Transcoder_AJAX_api_Specification#GetStatus

Dans le cas de l'historique, la valeur indiquée par « currenttrack » est récupérée. Il s'agit du chemin vers le fichier audio de la musique jouée actuellement par le transcodér en question. La classe WebradioManager contient une propriété « CurrentTrack » qui contient, justement, la valeur de la musique jouée. Afin de détecter lorsque le transcodér a changé de musique, un test vérifie si la valeur « currenttrack » renournée par l'API est différente de la valeur stockée dans la propriété « CurrentTrack » du transcodér. Si c'est le cas, le transcodér a donc changé de musique et cette dernière doit être ajoutée à l'historique.

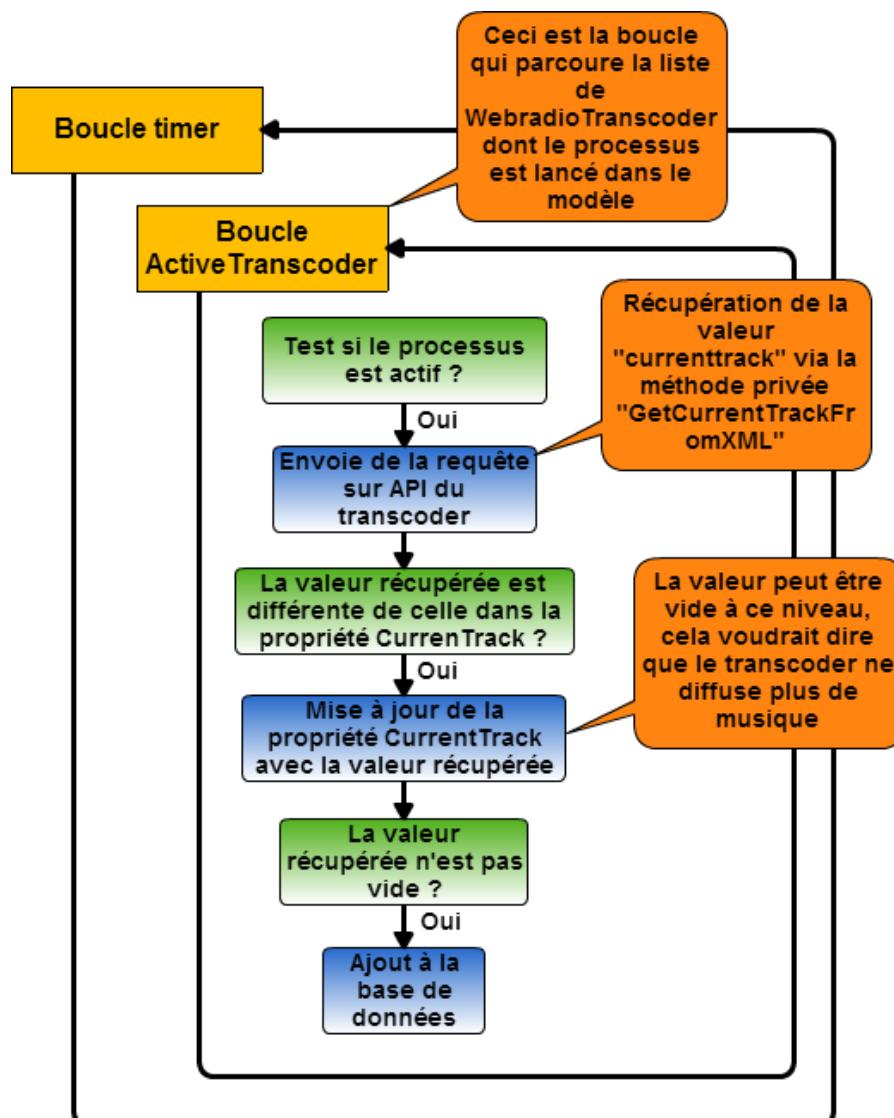


Figure 46 - Schéma gestion de l'historique

L'utilisateur peut ensuite générer un fichier PDF contenant l'historique d'un transcoder. La bibliothèque .NET « iTextSharp » (<https://github.com/itext/itextsharp>) est utilisée pour générer ce document. La méthode « GenerateHistory » du *model* va générer le fichier à l'emplacement sélectionné par l'utilisateur précédemment. L'historique stocké dans la base de données est récupéré sous la forme d'un dictionnaire (string,string) dont la clé correspond à la date et la valeur correspond au chemin vers le fichier joué.

6.13 SERVEUR DE DIFFUSION INTERNE

6.13.1 CLASSES UTILISÉES

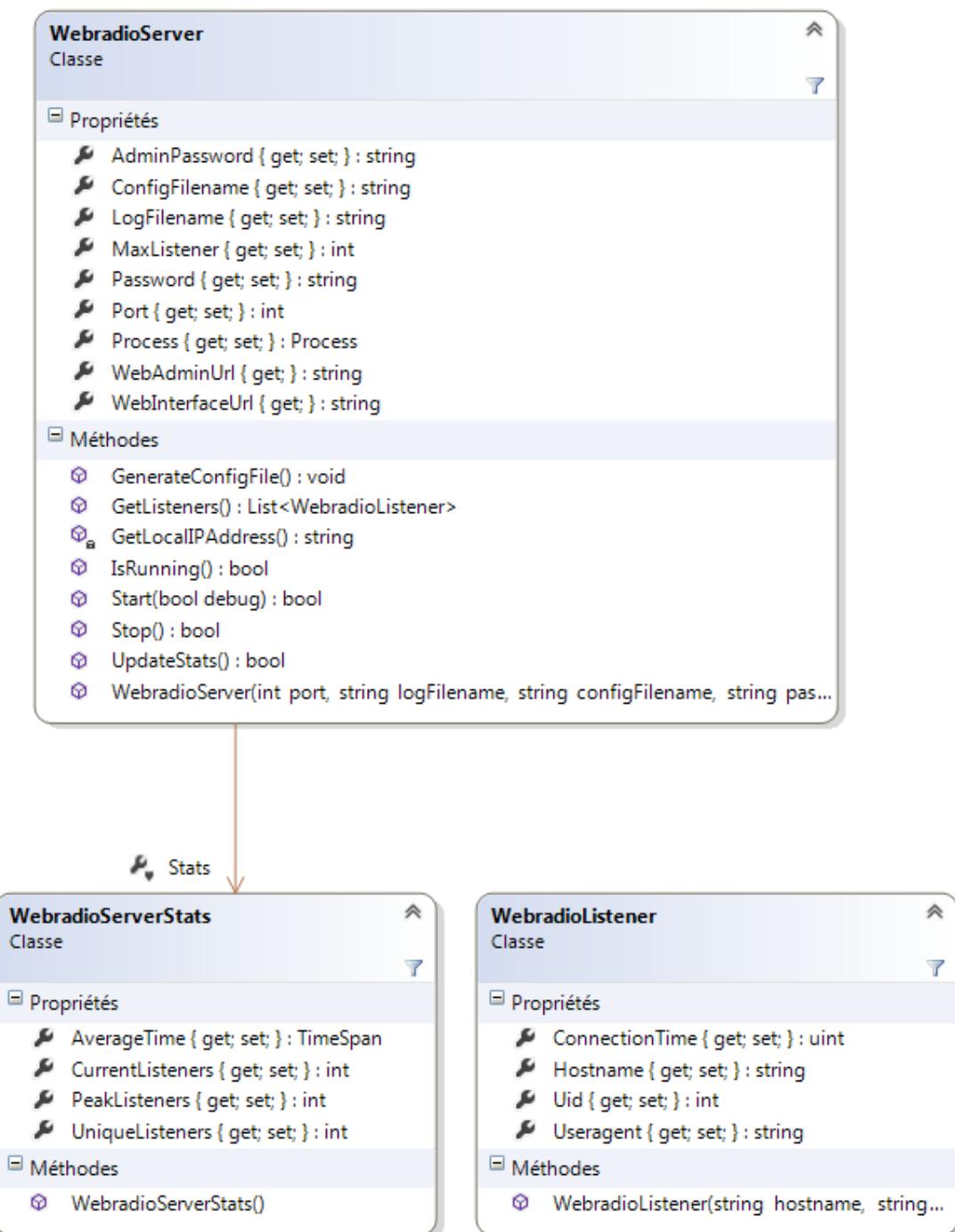


Figure 47 - Classe WebradioServer

La classe « `WebradioListener` » permet de transporter les informations concernant un auditeur connecté au serveur afin que la vue puisse afficher ses informations.

6.13.2 OUTIL UTILISÉ

Shoutcast est avant tout un serveur de diffusion de flux audio ou vidéo (Shoutcast DNAS server 2). L'outil propose un serveur en ligne de commande qui sera utilisée dans ce projet. Le serveur fonctionne avec un fichier de configuration qui lui est donnée en paramètre lorsque l'on l'exécute :

```
sc_serv.exe myconfig.config
```

```
C:\Users\MENETREYS_INFO\Documents\GitHub\WebRadioManager\WebRadioManager\WebRadio... 2014-05-08 09:28:06     WARN  [CONFIG] Deprecated statement found on line 7 of C:\Users\MENETREYS_INFO\Documents\GitHub\WebRadioManager\WebRadioManager\WebRadioManager\bin\Debug\webradios\Gorilo\server\config.config -> change autodumpstartime_1=0 to autodumpertime_1=0 2014-05-08 09:28:06     INFO  2014-05-08 09:28:06     INFO  ***** 2014-05-08 09:28:06     INFO  ** SHOUTcast Distributed Network Audio Server (DNAS) 2014-05-08 09:28:06     INFO  ** Copyright (C) 1999-2013 Nullsoft, Inc. All Rights Reserved. 2014-05-08 09:28:06     INFO  ***** 2014-05-08 09:28:06     INFO  [MAIN] SHOUTcast DNAS/win64 v2.2.1.109 (Nov 29 2013) 2014-05-08 09:28:06     INFO  [MAIN] PID: 10896 2014-05-08 09:28:06     INFO  [MAIN] Loaded config from C:\Users\MENETREYS_INFO\Documents\GitHub\WebRadioManager\WebRadioManager\WebRadioManager\bin\Debug\webradios\Gorilo\server\config.config 2014-05-08 09:28:06     INFO  [MAIN] Calculated CPU count is 8 -> using all available CPUs 2014-05-08 09:28:06     INFO  [MAIN] Starting 8 network threads 2014-05-08 09:28:06     INFO  [MICROSERVER] Listening for source and client connections on port 8000 2014-05-08 09:28:06     INFO  [MICROSERVER] Listening for legacy source connections on port 8001 2014-05-08 09:28:06     INFO  [MICROSERVER] Flash policy file server not enabled
```

Figure 48 - ShoutCAST serveur

La documentation concernant la configuration du serveur via le fichier est expliquée ici :

http://wiki.winamp.com/wiki/SHOUTcast_DNAS_Server_2. Dans le cadre de ce projet, seules quelques options seront utilisées et configurées. Une partie est configurable par l'utilisateur via l'interface de l'onglet «server» et l'autre partie est configurée par défaut par l'application :

- L'emplacement du fichier de log
- L'emplacement du fichier de configuration lui-même

Pour plus de détails sur les emplacements des différents fichiers, rendez-vous au chapitre concernant [la structure des dossiers/fichiers](#) 0.

6.13.3 CONFIGURATION

Voici la liste des paramètres configurés dans le fichier de configuration d'un serveur dans WebradioManager :

- Logfile : Le chemin vers le fichier de log
- Portbase : Le numéro du port du serveur (sur lequel le transcoder se connecte)
- Password : Le mot de passe de connexion pour une source (un transcoder)

- Adminpassword : Le mot de passe d'administration de l'interface web. Attention, il doit être différent du mot de passe de connexion pour les sources.
- Publicserver : Cette valeur sert à définir sur la webradio sera indexée dans le site de référencement de ShoutCAST. Il n'est pas d'actualité donc la valeur par défaut « always » est définie.
- Maxuser : Le nombre maximum de clients/auditeurs connectés
- Autodumpsourcetime : Nombre de secondes à attendre avant de déconnecter une source (transcoder) si son flux est vide. Dans le cas du projet, la valeur 0 est entrée par défaut (cela correspond à désactiver cette fonction), car il n'est pas voulu que le serveur ferme les connexions vides.

Exemple de fichier de configuration :

```
logfile=C:\Users\NETREYS_INFO\Documents\GitHub\WebradioManager\WebradioManager\WebradioManager\bin\Debug\webradios\Gorilo\server\log.txt

portbase=8000

password=lol

adminpassword=admin

publicserver=always

maxuser=32

autodumpsourcetime=0
```

6.13.4 MISE À JOUR DE LA CONFIGURATION

Lors d'une mise à jour de la configuration du serveur, il est d'abord arrêté puis modifié et ensuite redémarré (seulement dans le cas où il était démarré lors de la sauvegarde de la nouvelle configuration).

Les informations du formulaire sont passées au *model* via le *controller* de la *vue*. Il est testé si le serveur est allumé, si c'est le cas il est éteint et une variable booléenne est enregistrée à « *true* » afin de savoir, à la fin du traitement, si le serveur doit être redémarré.

Les informations sont modifiées dans la base de données, puis si la modification a réussi, elles sont modifiées dans le *model*. En fin de traitement, la méthode « *UpdateObservers* » est appelée.

6.13.5 EXÉCUTION ET FERMETURE

L'exécutable du serveur ShoutCAST est lancé depuis la classe « *WebradioServer* » et sa méthode « *Start* ». Cette dernière utilise la classe « *Process* » afin de créer un processus où l'exécutable sera lancé. Une classe nommée « *ProcessStartInfo* » permet de donner des paramètres d'exécution du processus :

```
ProcessStartInfo StartInfo = new ProcessStartInfo(Directory.GetCurrentDirectory() + SC_SERVER_FILENAME)
```

```
{  
    CreateNoWindow = true,  
    WindowStyle =  
(debug)?ProcessWindowStyle.Minimized:ProcessWindowStyle.Hidden,  
    Arguments = Directory.GetCurrentDirectory() + "\\\" +  
this.ConfigFilename.Replace('/', '\\')  
};
```

WebradioServer.cs

Il y a 2 modes d'exécution pour le serveur :

- Avec debug : Ouverture de la fenêtre console de l'application serveur ShoutCAST
- Sans debug : Pas d'ouverture de l'application console

Ce mode est choisi par l'utilisateur grâce à la case à cocher qui se trouve dans l'onglet « server ». Dans le code ci-dessus, la variable booléenne « debug » va définir si la propriété « WindowStyle » doit être « minimized » (ouverte, mais minimisée) ou « hidden » (pas de fenêtre apparente). Dans le premier cas, le debug est activé. Le processus est ensuite lancé via Process.Start(StartInfo) qui retourne un objet Process instancié et lancé qui sera enregistré dans la propriété « Process » de la classe « WebradioServer ».

2 cas d'erreurs sont possibles :

- L'exécutable ShoutCAST serveur (sc_serv.exe) n'est pas présent.
- Une instance précédente de ce processus (en cas de crash de l'application par exemple) est en tâche de fond et doit être fermée manuellement par l'utilisateur.

Pour plus d'informations concernant la gestion des différents processus, rendez-vous au chapitre [gestion des processus](#) 6.14.

6.13.6 STATISTIQUES ET AUDITEURS

Les statistiques du serveur sont récupérées via l'API web du serveur ShoutCAST (http://wiki.winamp.com/wiki/SHOUTcast_DNAS_Server_2_XML_Reponses#General_Server_Summary) et contiennent les informations suivantes :

- CurrentListeners : Le nombre d'auditeurs connectés (même avec doublons d'adresses IP)
- UniqueListeners : Le nombre d'auditeurs connectés (Adresses IP différentes)
- PeakListeners : Le record d'auditeurs connectés simultanément sur le serveur durant cette session d'exécution.
- AverageTime : Le temps moyen de connexion par auditeur.

La classe « WebradioServerStats » permet de stocker ces informations. La classe « WebradioServer » contient un champ « Stats » de ce type ainsi qu'une méthode « UpdateStats » qui permet de mettre à jour les données contenues dans le champ « Stats ».

Concernant les auditeurs, il est aussi possible, via l'API du serveur, de lister les auditeurs connectés et de récupérer les informations suivantes :

- Hostname : Adresse IP ou nom de domaine de l'auditeur
- Useragent : Décris le type de lecteur audio utilisé pour écouter la webradio
- ConnectionTime : Le temps de connexion en secondes.
- Uid : Un identifiant unique pour l'auditeur, distribué automatiquement par le serveur.

La classe « WebradioListener » permet de stocker ces informations. Dans la classe « WebradioServer », la méthode « GetListeners » retourne une liste d'objet de type « WebradioListener » avec les informations récupérées.

Depuis la *vue*, le bouton situé dans l'onglet « status » et nommé « Update » permet de lancer cette mise à jour de statistiques et de la liste d'auditeurs. La *vue* appelle 2 méthodes (de son *controller*) différentes et effectue donc les 2 mises à jour séparément :

- UpdateListeners : La *vue* va recevoir la liste d'objet « WebradioListener » en provenance de la classe « WebradioServer » et les afficher dans le DataGridView.
- UpdateServerStats : Va appeler la méthode « UpdateStats » de la classe « WebradioServer » puis le *vue* sera mise à jour via « UpdateObservers » depuis le *model*. Dans sa méthode « UpdateView », il est prévu que les informations des statistiques du serveur soient affichées.

6.13.7 AFFICHAGE DES INTERFACES WEB

Les 2 boutons qui permettent l'affichage de l'interface web et l'administration web du serveur appellent le contrôleur qui lui appellera le *model*. Ce dernier utilise la classe Process et sa méthode statique « Start » pour lancer les URL dans le navigateur par défaut de l'utilisateur. L'adresse de chacune des interfaces est récupérable via la propriété « WebInterfaceUrl » et « WebAdminUrl » :

```
Process.Start(this.Webradios[webradioId].Server.WebInterfaceUrl);

//Dans la classe WebradioServer :
public string WebInterfaceUrl

{
    get
    {
        return "http://" + this.LocalIPAddress() + ":" + this.Port;
    }
}
```

WebradioServer.cs

6.14 GESTION DES PROCESSUS

Les processus lancés par le programme sont instancié à l'aide de la classe .NET « Process » :

Fournit l'accès à des processus locaux ainsi que distants, et vous permet de démarrer et d'arrêter des processus système locaux.

Source : MSDN Developer Network²⁴

6.14.1 ISRUNNING

Chaque classe comportant une priorité de type Process (WebradioTranscoder et WebradioServer) dispose de 3 méthodes : Start, Stop et IsRunning. Voici le fonctionnement de la méthode :

```
foreach (Process prc in Process.GetProcesses())
{
    if (prc.ProcessName.Contains(this.Process.ProcessName))
    {
        result = true;
    }
}

if(this.Process.HasExited || !this.Process.Responding)
    result = false;

return result;
```

WebradioTranscoder.cs et WebradioServer.cs

En premier temps, il est vérifié si le processus est présent dans la liste des processus Windows. Si c'est le cas, un booléen est mis à vrai. Ensuite, un test vérifié si le processus a été fermé et ne répond plus. Si c'est le cas, cela veut dire que le processus présent dans la liste de processus Windows est un ancien processus qui ne correspond plus à celui présent dans la classe. Dans ce cas, le booléen est mis à faux.

6.14.2 DÉTECTION DES CRASHS/FERMETURES EXTERNES

Il est possible que les processus lancés par le logiciel soient fermés par une application externe ou qu'ils soient arrêtés par l'utilisateur en passant par le gestionnaire de tâches par exemple.

L'application doit donc pouvoir détecter quand cela arrive afin de mettre à jour les informations affichées à l'écran pour l'utilisateur.

²⁴ [http://msdn.microsoft.com/fr-fr/library/system.diagnostics.process\(v=vs.110\).aspx](http://msdn.microsoft.com/fr-fr/library/system.diagnostics.process(v=vs.110).aspx)

Pour ce faire, le *model* dispose de 2 propriétés de type « List<> ». Une contient des WebradioServer et l'autre des WebradioTranscoder. Ces 2 listes vont être remplies par les objets (référence vers ces objets) dont le processus est censé être en fonctionnement. Ensuite, un timer (une minuterie à laquelle est attaché une méthode qui est appelée toutes les X millisecondes) va, toutes les secondes, vérifier l'état de chacun des processus présents dans les classes contenues dans les listes. Il exécute la méthode « IsRunning » sur chaque processus. Si il ne « run » plus, il est enlevé de la liste. Si au moins une liste a été mise à jour, la méthode UpdateObservers est appelée à la fin de la méthode.

6.14.3 FERMETURE AUTOMATIQUE

Tous les processus en cours sont fermés à la fermeture volontaire de l'application par l'utilisateur. Il doit confirmer qu'il désire réellement fermer l'application.

7 TESTS

7.1 WEBRADIOS

N°	Description	Résultat	Commentaire
1	Affichage des webradios disponibles	OK	
2	Création d'une webradio	OK	
3	Suppression d'une webradio	OK	
4	Duplication d'une webradio	OK	Amélioration
5	Ouvrir une webradio	OK	
6	Ouverture de plusieurs webradios simultanément (différentes ou non)	OK	
7	Applications des modifications à toutes les webradios ouvertes	OK	Mise à jour des informations affichées (observateurs/sujet)
8	Changement du nom de la webradio	OK	Mise à jour dans la liste de webradios et l'administration de la webradio en question

7.2 STATUS

Corresponds à l'onglet « status » disponible dans chaque AdminView liée à une webradio.

N°	Description	Résultat	Commentaire
9	Affichage des statuts des transcodeurs de la webradio	OK	
10	Affichage des auditeurs + statistiques du serveur de la webradio	OK	
11	Mise à jour de l'affichage précédent	OK	Ne crash pas lorsque le serveur est arrêté
12	Affichage des musiques actuellement jouées sur chaque transcoder de la webradio	OK	

7.3 BIBLIOTHÈQUE

N°	Description	Résultat	Commentaire
13	Affichage des musiques/publicités dans les tableaux	OK	

14	Recherche dans les 2 tableaux	OK	Par n'importe quel critère
15	Importer des morceaux par dossier	OK	
16	Importer des morceaux par dossier de façon récursive	OK	
17	Importer des morceaux par fichiers directs	OK	
18	Ajouter des morceaux à une playlist	OK	Ajout multiple ou simple
19	Suppression de morceaux	OK	Multiple ou simple

7.4 LISTES DE LECTURE

N°	Description	Résultat	Commentaire
20	Création de playlists simple	OK	
21	Création de playlists générée	OK	Selon durée et genre (pour les playlists musicales)
22	Affichage des playlists disponibles	OK	Musicales ou publicitaires
23	Affichage du contenu d'une playlist	OK	
24	Affichage du temps total de la playlist	OK	
25	Recherche dans le contenu d'une playlist	OK	N'importe quel critère
26	Retirer morceaux de la playlist	OK	Sélection multiple ou simple
27	Suppression de playlist	OK	

7.5 GRILLE HORAIRE

N°	Description	Résultat	Commentaire
28	Affichage des événements du calendrier	OK	Dans le composant : rouge = publicités et bleu = musiques
29	Création d'événements	OK	Avec choix de playlist, etc.
30	Remplissage du formulaire de création automatique	OK	Remplissage en fonction de la sélection faite sur le composant Calendar (début, durée et

			jour)
31	Déplacement d'un événement	OK	Empêcher d'avoir 2 fois le même événement le même jour. Amélioration.

7.6 TRANSCODERS

N°	Description	Résultat	Commentaire
32	Création de transcodeurs	OK	Pas 2 fois le même nom par webradio
33	Affichage des transcodeurs disponibles	OK	
34	Affichage de la configuration d'un transcodeur	OK	
35	Modification de la configuration d'un transcodeur	OK	Nécessite le redémarrage du transcodeur
36	Suppression de transcodeurs	OK	
37	Démarrage/arrêt de transcodeur	OK	Mode debug ou non
38	Affichage du statut d'un transcodeur	OK	Allumé ou arrêté
39	Passage au morceau suivant sur le transcodeur	OK	Amélioration
40	Génération et affichage de l'historique	OK	Format PDF
41	Effacement de l'historique	OK	Dans la base de données
42	Affichage du fichier de log	OK	Éditeur par défaut
43	Capture live – Listing des périphériques audio	~OK	Périphérique bien listé, mais pas sûr que la bonne information soit récupérée (nom). Amélioration
44	Capture live - Lancement	OK	Informations bien configurées dans le transcodeur. Amélioration
45	Capture live - fonctionnelle	KO	Le transcodeur n'arrive pas à trouver le périphérique audio qui lui est configuré. Amélioration

7.7 SERVEUR DE DIFFUSION

N°	Description	Résultat	Commentaire
46	Affichage des paramètres du serveur	OK	
47	Modification des paramètres du serveur	OK	
48	Démarrage/arrêt du serveur	OK	Mode debug ou non
49	Affichage du statut du serveur	OK	
50	Affichage des interfaces web via bouton	OK	
51	Affichage du fichier de log	OK	

7.8 PROCESSUS

N°	Description	Résultat	Commentaire
52	Détection des processus crachés ou arrêtés extérieurement à l'application	OK	
53	Fermeture des processus liés à une webradio lors de la fermeture de sa dernière fenêtre d'administration ouverte	OK	
54	Fermeture de tous les processus à la fermeture de l'application	OK	
55	Détection des processus en cours	OK	IsRunning

7.9 AUTRES

N°	Description	Résultat	Commentaire
56	Régénération de toutes les configurations via le menu	OK	
57	Vérification de l'intégrité de la bibliothèque via le menu	OK	

8 CONCLUSION

8.1 BILAN PERSONNEL

Le projet WebradioManager consistait à concevoir un gestionnaire de webradios et cet objectif a été atteint. Le cahier des charges défini avec l'entreprise KTFM a été rempli. Le projet ayant avancé rapidement durant la durée du travail, des améliorations ont pu être discutées et ajoutées :

- La modification des informations de la bibliothèque (avec tag des fichiers musicaux)
- Ajout de la capture live (expérimentale)
- Duplication (clonage) d'une webradio
- Affichage des auditeurs et musiques en cours

Concernant les problèmes rencontrés, ils furent surtout présents dans la partie « gestion de processus ». C'est-à-dire, faire en sorte que mon application garde un plein contrôle sur les processus qu'elle lance (transcodeurs et serveurs). J'ai dû mettre en place un moyen de « surveiller » le processus afin de toujours savoir leur état et en informer l'utilisateur. J'ai aussi eu quelques soucis à utiliser l'API Ajax des transcodeurs, mais après quelques recherches, j'ai pu maîtriser cet outil. Le dernier problème concerne l'ajout de l'amélioration concernant la capture en live qui ne fonctionne pas parfaitement à cause d'un mauvais listing des périphériques son.

Pour le reste de l'application, les fonctionnalités se sont facilement ajoutées. J'avais la chance d'avoir un cahier des charges bien défini et d'avoir le fonctionnement de l'application au clair dans ma tête. Celle m'a permis de faire une analyse rapide et de pouvoir faire fonctionner toutes les fonctionnalités en « harmonie ». Au final, le projet a pu avancer rapidement et des améliorations ont pu être envisagées.

Je suis satisfait de mon travail, car il est fonctionnel et il m'a permis de réaliser un outil dont j'ai toujours rêvé afin de créer une webradio facilement. C'était une bonne expérience de réaliser un projet en collaboration avec une vraie entreprise possédant elle-même une webradio. Nous avons pu échanger et discuter. En plus de cela, c'est un univers qui m'intéresse beaucoup et j'ai appris de nombreuses choses.

J'ai pu appliquer mes connaissances en C# dans le cadre d'un projet important ainsi que de les consolider pour la suite de mon avenir professionnel. J'ai remarqué qu'avec les temps, mes habitudes de codage s'améliorent de mieux en mieux et cela me permet de faire moins d'erreurs et par conséquent, d'aller plus vite lors du développement.

Pour conclure, ce fut un superbe projet que j'ai pris du plaisir à réaliser dans le cadre de mon travail diplôme. Je compte poursuivre son développement afin d'implémenter les améliorations possibles au projet (elles sont nombreuses).

8.2 AMÉLIORATIONS POSSIBLES

- Multiserveur de diffusion par transcodeur
- Évènements périodiques (+ top horaire)
- DJ
- Gestion de semaines « type » (gestion multi semaine)
- Ajouter les exécutables ShoutCAST dans les ressources du logiciel.
- Srialisation
- Génération de playlist : durée maximum et minimum des morceaux choisis
- Multiplateformes

9 REMERCIEMENTS

Je remercie mon professeur Francisco Garcia pour m'avoir suivi et aidé pendant tout mon travail. Je le remercie aussi de m'avoir mis en contact avec l'entreprise KTFM que je remercie également pour avoir pris le temps de prendre part à mon projet.

Je remercie également toute ma classe pour leur soutien et la bonne ambiance durant le travail. Je suis avec la plupart d'entre eux depuis ma première année de CFC.

10 APPORTS PERSONNELS

Élément	Apport (%)	Commentaire
Patron MVC et observateurs/sujet	100	
Interaction directe avec la base de données	10	Classe BddControls trouvée sur un tutoriel. J'y ai apporté quelques modifications.
Interfaces	100	
CRUD²⁵ des webradios	100	
Renommer webradio	100	
Duplication webradio	100	
CRUD des morceaux musicaux et publicitaires	100	
CRUD des playlists	100	
CRUD des événements dans le calendrier	100	

²⁵ CRUD (pour Create, Read, Update, Delete) désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.

Gestion visuel d'un calendrier hebdomadaire	0	Composant trouvé sur internet
CRUD des transcodeurs	100	
Résolution d'adresse IP	0	Code trouvé sur un tutoriel
CRUD serveurs de diffusion	100	
Affichage des différentes informations dans la vue	100	
Génération des configurations de chaque élément	100	
Gestion des divers fichiers sur le disque	90	Sauf différents fichiers temporaires créés par le transcodeur et le serveur
Transcodeur	0	Exécutable transcodeur fournit par ShoutCAST
Serveur de diffusion	0	Exécutable serveur de diffusion fournit par ShoutCAST
Gestion des processus	100	Avec l'aide de la classe .NET « Process ». Regroupe le démarrage/arrêt/surveillance des processus.
Gestion de l'historique	80	Utilisation de l'API Ajax du transcoder pour récupérer des informations
Capture live	25	Capture live gérée par le transcodeur, mais configuration par moi-même

11 GLOSSAIRE

- Playlist : Liste de lecture contenant plusieurs fichiers musicaux.
- Stream : Flux
- Log : Journal des événements
- Item : Objet/élément
- Live : En direct
- Instanciation : (Anglicisme) Action d'instancier, d'initialiser en programmation, à partir d'un espace mémoire réservé, un objet à partir d'un ensemble de caractéristiques, appelé «classe». Les objets sont obtenus à partir d'une instanciation de classe.
- Object : Classe qui est à la base de toutes les classes .NET.
- SQL-92 : La 3e révision du langage SQL.
- ACID : les propriétés ACID (atomicité, cohérence, isolation et durabilité) sont un ensemble de propriétés qui garantissent qu'une transaction informatique est exécutée de façon fiable.
- Ajax : (acronyme d'Asynchronous JavaScript and XML) permet de construire des applications Web et des sites web dynamiques interactifs sur le poste client en se servant de différentes technologies ajoutées aux navigateurs web entre 1995 et 2005.

12 ILLUSTRATIONS

Figure 1 - Mindmap (discussion avec KTF	12
Figure 2 - Schéma application	13
Figure 3 - Interface principale AdminView.....	14
Figure 4 - Interface gestion des webradios SelectionView	15
Figure 5 - Schéma webradios	16
Figure 6 - Onglet "status"	18
Figure 7 - Bibliothèque de musiques.....	19
Figure 8 - Onglet "playlists"	21
Figure 9 - Onglet "timetable"	23
Figure 10 - Calendrier	23
Figure 11 - Sélection multiple calendrier	24
Figure 12 - Définition d'un événement avec des éléments.....	25
Figure 13 - Onglet "Transcoders"	26
Figure 14 - Onglet "server"	28
Figure 15 - Interface web ShoutCAST serveur.....	29

Figure 16 - Administration web ShoutCAST serveur	30
Figure 17 - Diagramme <i>model</i>	31
Figure 18 - MVC	32
Figure 19 - Pattern observateurs/sujet	33
Figure 20 - Observateurs/sujet dans WebradioManager.....	34
Figure 21 - AudioType et StreamType enum.....	34
Figure 22 - Logo SQLite.....	36
Figure 23 - Principe de base de diffusion	45
Figure 24 - Schéma de diffusion	45
Figure 25 - Schéma de diffusion 2	46
Figure 26 - Logo Schoutcast.....	46
Figure 27 - Schéma shoutcast.....	48
Figure 28 - Schéma structure des fichiers/dossiers	49
Figure 29 - Exemple lancement transcoder	50
Figure 30 - Diagramme de séquence initialisation application	52
Figure 31 - Classe "Webradio".....	53
Figure 32 - Schéma création webradio.....	54
Figure 33 - Diagramme de séquence instanciation AdminView	55
Figure 34 - Schema changement nom webradio	58
Figure 35 - Classes bibliothèque	60
Figure 36 - Schéma importation fichier	62
Figure 37 - Classes Playlist	67
Figure 38 - Algorithme génération playlist.....	70
Figure 39 - Classes timetable.....	73
Figure 40 - Day View Calendar	74
Figure 41 - Exemple XML calendrier.....	76
Figure 42 - Modification d'un événement.....	80

Figure 43 - Classes transcodeurs	82
Figure 44 - ShoutCAST transcoder console	83
Figure 45 - Schéma création d'un transcoder	87
Figure 46 - Schéma gestion de l'historique	93
Figure 47 - Classe WebradioServer	95
Figure 48 - ShoutCAST serveur	96

13 RÉFÉRENCES

- <https://cacoo.com> : Création de diagrammes en ligne
- <http://balsamiq.com/> : Création de « mokup »
- <http://calendar.codeplex.com/> : Composant C# pour l'affichage du calendrier sur une semaine
- <http://sqlitestudio.pl/> : Logiciel de gestion de base de données SQLite
- <https://github.com/itext/itextsharp> : Bibliothèque .NET pour la génération de document (PDF, etc.)
- <http://www.shoutcast2.com/> : Logiciel de diffusion pour webradio

14 ANNEXES

- Plannings
- Documentation et listing du code
- Poster format A4

WebradioManager

1.0

Generated by Doxygen 1.8.5

Tue May 27 2014 15:38:59

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	Package WebradioManager	9
6	Class Documentation	11
6.1	WebradioManager.Ad Class Reference	11
6.1.1	Detailed Description	11
6.1.2	Constructor & Destructor Documentation	12
6.1.2.1	Ad	12
6.1.2.2	Ad	12
6.2	WebradioManagerAdminController Class Reference	13
6.2.1	Detailed Description	15
6.2.2	Constructor & Destructor Documentation	15
6.2.2.1	AdminController	15
6.2.3	Member Function Documentation	15
6.2.3.1	AddToPlaylist	15
6.2.3.2	CheckFolders	16
6.2.3.3	CheckLibrary	16
6.2.3.4	ClearHistory	16
6.2.3.5	CreateEvent	17
6.2.3.6	CreatePlaylist	17
6.2.3.7	CreateTranscoder	18

6.2.3.8	DeleteAudioFile	18
6.2.3.9	DeleteEvent	19
6.2.3.10	DeletePlaylist	19
6.2.3.11	DeleteTranscoder	19
6.2.3.12	FormClose	20
6.2.3.13	GenerateAllConfigs	20
6.2.3.14	GenerateHistory	20
6.2.3.15	GeneratePlaylist	21
6.2.3.16	GetAudioFileByFilename	21
6.2.3.17	GetGenders	22
6.2.3.18	GetLibrary	22
6.2.3.19	GetPlaylistContent	22
6.2.3.20	GetServerListeners	23
6.2.3.21	GetSimilarViewCount	23
6.2.3.22	GetWebradio	24
6.2.3.23	ImportFilesToLibrary	24
6.2.3.24	ModifyWebradioName	24
6.2.3.25	RemoveFromPlaylist	25
6.2.3.26	ShowServerWebAdmin	25
6.2.3.27	ShowServerWebInterface	25
6.2.3.28	StartServer	26
6.2.3.29	StartTranscoder	26
6.2.3.30	StopAllTranscoders	27
6.2.3.31	StopServer	27
6.2.3.32	StopTranscoder	28
6.2.3.33	TranscoderCapture	28
6.2.3.34	TranscoderNextTrack	28
6.2.3.35	UpdateAudioFile	29
6.2.3.36	UpdateEvent	29
6.2.3.37	UpdateServer	30
6.2.3.38	UpdateServerStats	30
6.2.3.39	UpdateTranscoder	30
6.2.3.40	UpdateView	31
6.2.4	Property Documentation	31
6.2.4.1	Model	31
6.2.4.2	View	31
6.3	WebradioManager.AdminView Class Reference	32
6.3.1	Detailed Description	32
6.3.2	Constructor & Destructor Documentation	32
6.3.2.1	AdminView	32

6.3.3	Member Function Documentation	33
6.3.3.1	UpdateView	33
6.3.4	Property Documentation	33
6.3.4.1	Controller	33
6.3.4.2	EventsCalendar	33
6.3.4.3	IdWebradio	33
6.3.4.4	NameWebradio	34
6.4	WebradioManager.AudioFile Class Reference	34
6.4.1	Detailed Description	35
6.4.2	Constructor & Destructor Documentation	35
6.4.2.1	AudioFile	35
6.4.2.2	AudioFile	35
6.4.3	Property Documentation	36
6.4.3.1	Album	36
6.4.3.2	Artist	36
6.4.3.3	Duration	36
6.4.3.4	Filename	37
6.4.3.5	Gender	37
6.4.3.6	Id	37
6.4.3.7	Label	37
6.4.3.8	Title	37
6.4.3.9	Type	37
6.4.3.10	Year	38
6.5	WebradioManager.Bdd Class Reference	38
6.5.1	Detailed Description	39
6.5.2	Constructor & Destructor Documentation	40
6.5.2.1	Bdd	40
6.5.3	Member Function Documentation	40
6.5.3.1	AddAudioFile	40
6.5.3.2	AddEvent	40
6.5.3.3	AddGender	41
6.5.3.4	AddGeneratedPlaylist	41
6.5.3.5	AddToHistory	42
6.5.3.6	AddToPlaylist	42
6.5.3.7	AddTranscoder	42
6.5.3.8	AddWebradio	43
6.5.3.9	AudioFileExist	43
6.5.3.10	ClearHistory	44
6.5.3.11	CreatePlaylist	44
6.5.3.12	DeleteAudioFile	44

6.5.3.13	DeleteEvent	45
6.5.3.14	DeletePlaylist	45
6.5.3.15	DeleteTranscoder	46
6.5.3.16	DeleteWebradio	46
6.5.3.17	EventExist	46
6.5.3.18	GetGenderId	47
6.5.3.19	GetGenders	47
6.5.3.20	GetHistory	47
6.5.3.21	LoadLibrary	48
6.5.3.22	LoadWebradios	48
6.5.3.23	ModifyWebradioName	49
6.5.3.24	RemoveFromPlaylist	49
6.5.3.25	TranscoderExist	49
6.5.3.26	UpdateAudioFile	50
6.5.3.27	UpdateEvent	50
6.5.3.28	UpdateFilenames	51
6.5.3.29	UpdateServer	51
6.5.3.30	UpdateTranscoder	51
6.5.3.31	WebradioExist	52
6.5.4	Property Documentation	52
6.5.4.1	Controls	52
6.6	WebradioManager.BddControls Class Reference	53
6.6.1	Detailed Description	53
6.6.2	Member Function Documentation	53
6.6.2.1	ClearDB	53
6.6.2.2	ClearTable	53
6.6.2.3	Delete	54
6.6.2.4	ExecuteDataReader	55
6.6.2.5	ExecuteNonQuery	55
6.6.2.6	ExecuteScalar	55
6.6.2.7	GetDataTable	55
6.6.2.8	Insert	56
6.6.2.9	Update	57
6.7	WebradioManager.CalendarEvent Class Reference	57
6.7.1	Detailed Description	58
6.7.2	Constructor & Destructor Documentation	58
6.7.2.1	CalendarEvent	58
6.7.2.2	CalendarEvent	59
6.7.3	Member Function Documentation	59
6.7.3.1	GetSelectedDays	59

6.7.4	Property Documentation	60
6.7.4.1	Duration	60
6.7.4.2	Id	60
6.7.4.3	Loopatend	60
6.7.4.4	Name	60
6.7.4.5	Playlist	60
6.7.4.6	Priority	61
6.7.4.7	Repeat	61
6.7.4.8	Shuffle	61
6.7.4.9	StartTime	61
6.8	WebradioManager.DayWeek Struct Reference	61
6.8.1	Detailed Description	62
6.8.2	Property Documentation	62
6.8.2.1	Friday	62
6.8.2.2	Monday	62
6.8.2.3	Saturday	63
6.8.2.4	Sunday	63
6.8.2.5	Thursday	63
6.8.2.6	Tuesday	63
6.8.2.7	Wednesday	63
6.9	WebradioManager.EventAppointment Class Reference	64
6.9.1	Detailed Description	64
6.9.2	Constructor & Destructor Documentation	64
6.9.2.1	EventAppointment	64
6.9.3	Property Documentation	65
6.9.3.1	EventObject	65
6.9.3.2	Playlist	65
6.10	WebradioManager.IController Interface Reference	65
6.10.1	Detailed Description	65
6.11	WebradioManager.Music Class Reference	66
6.11.1	Detailed Description	66
6.11.2	Constructor & Destructor Documentation	66
6.11.2.1	Music	66
6.11.2.2	Music	67
6.12	WebradioManager.Playlist Class Reference	67
6.12.1	Detailed Description	68
6.12.2	Constructor & Destructor Documentation	68
6.12.2.1	Playlist	68
6.12.2.2	Playlist	69
6.12.3	Member Function Documentation	69

6.12.3.1 GenerateConfigFile	69
6.12.3.2 ToString	69
6.12.4 Property Documentation	70
6.12.4.1 AudioFileList	70
6.12.4.2 Filename	70
6.12.4.3 Id	70
6.12.4.4 Name	70
6.12.4.5 Type	71
6.13 WebradioManager.PlaylistAd Class Reference	71
6.13.1 Detailed Description	71
6.13.2 Constructor & Destructor Documentation	71
6.13.2.1 PlaylistAd	71
6.13.2.2 PlaylistAd	72
6.14 WebradioManager.PlaylistMusic Class Reference	72
6.14.1 Detailed Description	73
6.14.2 Constructor & Destructor Documentation	73
6.14.2.1 PlaylistMusic	73
6.14.2.2 PlaylistMusic	73
6.15 WebradioManager.SelectionController Class Reference	74
6.15.1 Detailed Description	75
6.15.2 Constructor & Destructor Documentation	75
6.15.2.1 SelectionController	75
6.15.3 Member Function Documentation	75
6.15.3.1 CreateWebradio	75
6.15.3.2 DeleteWebradio	76
6.15.3.3 DuplicateWebradio	76
6.15.3.4 GetWebradios	76
6.15.3.5 LoadLibrary	77
6.15.3.6 LoadWebradios	77
6.15.3.7 OpenWebradio	77
6.15.3.8 StopAllProcess	77
6.15.3.9 UpdateView	78
6.15.4 Property Documentation	78
6.15.4.1 Model	78
6.15.4.2 View	78
6.16 WebradioManager.SelectionView Class Reference	78
6.16.1 Detailed Description	79
6.16.2 Constructor & Destructor Documentation	79
6.16.2.1 SelectionView	79
6.16.3 Member Function Documentation	79

6.16.3.1	Dispose	79
6.16.3.2	UpdateView	80
6.16.4	Property Documentation	80
6.16.4.1	Controller	80
6.17	WebradioManager.TranscoderAacPlus Class Reference	80
6.17.1	Detailed Description	81
6.17.2	Constructor & Destructor Documentation	81
6.17.2.1	TranscoderAacPlus	81
6.17.2.2	TranscoderAacPlus	81
6.18	WebradioManager.TranscoderMp3 Class Reference	82
6.18.1	Detailed Description	82
6.18.2	Constructor & Destructor Documentation	83
6.18.2.1	TranscoderMp3	83
6.18.2.2	TranscoderMp3	83
6.19	WebradioManager.Webradio Class Reference	84
6.19.1	Detailed Description	84
6.19.2	Constructor & Destructor Documentation	85
6.19.2.1	Webradio	85
6.19.2.2	Webradio	85
6.19.3	Member Function Documentation	85
6.19.3.1	GenerateConfigFiles	85
6.19.3.2	ToString	86
6.19.4	Property Documentation	86
6.19.4.1	Calendar	86
6.19.4.2	Id	86
6.19.4.3	Name	86
6.19.4.4	Playlists	86
6.19.4.5	Server	87
6.19.4.6	Transcoders	87
6.20	WebradioManager.WebradioCalendar Class Reference	87
6.20.1	Detailed Description	87
6.20.2	Constructor & Destructor Documentation	88
6.20.2.1	WebradioCalendar	88
6.20.2.2	WebradioCalendar	88
6.20.3	Member Function Documentation	88
6.20.3.1	GenerateConfigFile	88
6.20.4	Property Documentation	89
6.20.4.1	Events	89
6.20.4.2	Filename	89
6.20.4.3	Id	89

6.21 WebradioManager.WebradioListener Class Reference	89
6.21.1 Detailed Description	90
6.21.2 Constructor & Destructor Documentation	90
6.21.2.1 WebradioListener	90
6.21.3 Property Documentation	90
6.21.3.1 ConnectionTime	90
6.21.3.2 Hostname	90
6.21.3.3 Uid	91
6.21.3.4 Useragent	91
6.22 WebradioManager.WebradioServer Class Reference	91
6.22.1 Detailed Description	92
6.22.2 Constructor & Destructor Documentation	92
6.22.2.1 WebradioServer	92
6.22.3 Member Function Documentation	93
6.22.3.1 GenerateConfigFile	93
6.22.3.2 GetListeners	93
6.22.3.3 IsRunning	93
6.22.3.4 Start	94
6.22.3.5 Stop	94
6.22.3.6 UpdateStats	94
6.22.4 Property Documentation	95
6.22.4.1 AdminPassword	95
6.22.4.2 ConfigFilename	95
6.22.4.3 LogFilename	95
6.22.4.4 MaxListener	95
6.22.4.5 Password	95
6.22.4.6 Port	96
6.22.4.7 Process	96
6.22.4.8 WebAdminUrl	96
6.22.4.9 WebInterfaceUrl	96
6.23 WebradioManager.WebradioServerStats Class Reference	96
6.23.1 Detailed Description	97
6.23.2 Constructor & Destructor Documentation	97
6.23.2.1 WebradioServerStats	97
6.23.3 Property Documentation	97
6.23.3.1 AverageTime	97
6.23.3.2 CurrentListeners	98
6.23.3.3 PeakListeners	98
6.23.3.4 UniqueListeners	98
6.24 WebradioManager.WebradioTranscoder Class Reference	98

6.24.1	Detailed Description	100
6.24.2	Constructor & Destructor Documentation	100
6.24.2.1	WebradioTranscoder	100
6.24.2.2	WebradioTranscoder	101
6.24.3	Member Function Documentation	101
6.24.3.1	GenerateConfigFile	101
6.24.3.2	GetStatus	102
6.24.3.3	IsRunning	102
6.24.3.4	NextTrack	102
6.24.3.5	SetCaptureMode	103
6.24.3.6	Start	103
6.24.3.7	Stop	103
6.24.3.8	ToString	104
6.24.4	Property Documentation	104
6.24.4.1	AdminPort	104
6.24.4.2	Birate	104
6.24.4.3	CalendarFile	104
6.24.4.4	Capture	104
6.24.4.5	ConfigFilename	105
6.24.4.6	CurrentTrack	105
6.24.4.7	Id	105
6.24.4.8	Ip	105
6.24.4.9	LogFilename	105
6.24.4.10	Name	105
6.24.4.11	Password	106
6.24.4.12	Port	106
6.24.4.13	Process	106
6.24.4.14	SampleRate	106
6.24.4.15	StreamType	106
6.24.4.16	Url	106
6.25	WebradioManager.WMMModel Class Reference	107
6.25.1	Detailed Description	109
6.25.2	Constructor & Destructor Documentation	110
6.25.2.1	WMMModel	110
6.25.3	Member Function Documentation	110
6.25.3.1	AddObserver	110
6.25.3.2	AddToPlaylist	110
6.25.3.3	CheckFolders	111
6.25.3.4	CheckLibrary	111
6.25.3.5	ClearHistory	111

6.25.3.6 CreateEvent	112
6.25.3.7 CreatePlaylist	112
6.25.3.8 CreateTranscoder	112
6.25.3.9 CreateWebradio	113
6.25.3.10 DeleteAudioFile	113
6.25.3.11 DeleteEvent	114
6.25.3.12 DeletePlaylist	114
6.25.3.13 DeleteTranscoder	115
6.25.3.14 DeleteWebradio	115
6.25.3.15 DuplicateWebradio	115
6.25.3.16 GenerateConfigFiles	116
6.25.3.17 GenerateHistory	116
6.25.3.18 GeneratePlaylist	117
6.25.3.19 GetAudioFileByFilename	117
6.25.3.20 GetGenders	117
6.25.3.21 GetLibrary	118
6.25.3.22 GetPlaylistContent	118
6.25.3.23 GetSimiliarViewCount	118
6.25.3.24 GetWebradio	119
6.25.3.25 GetWebradioByName	119
6.25.3.26 GetWebradios	120
6.25.3.27 ImportFilesToLibrary	120
6.25.3.28 LoadLibrary	120
6.25.3.29 LoadWebradios	121
6.25.3.30 ModifyWebradioName	121
6.25.3.31 RemoveFromPlaylist	121
6.25.3.32 RemoveObserver	122
6.25.3.33 ShowServerWebAdmin	122
6.25.3.34 ShowServerWebInterface	122
6.25.3.35 StartServer	123
6.25.3.36 StartTranscoder	123
6.25.3.37 StopAllProcess	124
6.25.3.38 StopAllProcess	124
6.25.3.39 StopServer	124
6.25.3.40 StopTranscoder	125
6.25.3.41 TranscoderCapture	125
6.25.3.42 TranscoderNextTrack	126
6.25.3.43 UpdateAudioFile	126
6.25.3.44 UpdateEvent	126
6.25.3.45 UpdateServer	127

6.25.3.46 UpdateServerListeners	127
6.25.3.47 UpdateServerStats	128
6.25.3.48 UpdateTranscoder	128
6.25.4 Property Documentation	129
6.25.4.1 ActiveServers	129
6.25.4.2 ActiveTranscoders	129
6.25.4.3 Bdd	129
6.25.4.4 Library	129
6.25.4.5 Observers	129
6.25.4.6 ProcessWatcher	129
6.25.4.7 Webradios	130
7 File Documentation	131
7.1 Ad.cs File Reference	131
7.1.1 Detailed Description	131
7.2 Ad.cs	131
7.3 AdminController.cs File Reference	132
7.3.1 Detailed Description	132
7.4 AdminController.cs	132
7.5 AdminView.cs File Reference	135
7.5.1 Detailed Description	135
7.6 AdminView.cs	136
7.7 AudioFile.cs File Reference	150
7.7.1 Detailed Description	150
7.8 AudioFile.cs	150
7.9 AudioType.cs File Reference	152
7.9.1 Detailed Description	152
7.10 AudioType.cs	152
7.11 Bdd.cs File Reference	153
7.11.1 Detailed Description	153
7.12 Bdd.cs	153
7.13 CalendarEvent.cs File Reference	162
7.13.1 Detailed Description	163
7.14 CalendarEvent.cs	163
7.15 DayWeek.cs File Reference	165
7.15.1 Detailed Description	165
7.16 DayWeek.cs	165
7.17 EventAppointment.cs File Reference	166
7.17.1 Detailed Description	166
7.18 EventAppointment.cs	166

7.19 IController.cs File Reference	167
7.19.1 Detailed Description	167
7.20 IController.cs	167
7.21 Music.cs File Reference	168
7.21.1 Detailed Description	168
7.22 Music.cs	168
7.23 Playlist.cs File Reference	168
7.23.1 Detailed Description	169
7.24 Playlist.cs	169
7.25 PlaylistAd.cs File Reference	170
7.25.1 Detailed Description	170
7.26 PlaylistAd.cs	170
7.27 PlaylistMusic.cs File Reference	171
7.27.1 Detailed Description	171
7.28 PlaylistMusic.cs	171
7.29 SelectionController.cs File Reference	171
7.29.1 Detailed Description	172
7.30 SelectionController.cs	172
7.31 SelectionView.cs File Reference	173
7.31.1 Detailed Description	173
7.32 SelectionView.cs	173
7.33 StreamType.cs File Reference	175
7.33.1 Detailed Description	175
7.34 StreamType.cs	175
7.35 TranscoderAacPlus.cs File Reference	175
7.35.1 Detailed Description	176
7.36 TranscoderAacPlus.cs	176
7.37 TranscoderMp3.cs File Reference	176
7.37.1 Detailed Description	176
7.38 TranscoderMp3.cs	176
7.39 Webradio.cs File Reference	177
7.39.1 Detailed Description	177
7.40 Webradio.cs	177
7.41 WebradioCalendar.cs File Reference	178
7.41.1 Detailed Description	179
7.42 WebradioCalendar.cs	179
7.43 WebradioListener.cs File Reference	180
7.43.1 Detailed Description	180
7.44 WebradioListener.cs	180
7.45 WebradioServer.cs File Reference	181

7.45.1 Detailed Description	181
7.46 WebradioServer.cs	181
7.47 WebradioServerStats.cs File Reference	185
7.47.1 Detailed Description	185
7.48 WebradioServerStats.cs	185
7.49 WebradioTranscoder.cs File Reference	186
7.49.1 Detailed Description	186
7.50 WebradioTranscoder.cs	186
7.51 WMMModel.cs File Reference	191
7.51.1 Detailed Description	191
7.52 WMMModel.cs	191
 Index	 206

Chapter 1

Namespace Index

1.1 Packages

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

WebradioManager	9
-----------------	-------	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Appointment	
WebradioManager.EventAppointment	64
WebradioManager.AudioFile	34
WebradioManager.Ad	11
WebradioManager.Music	66
WebradioManager.Bdd	38
WebradioManager.BddControls	53
WebradioManager.CalendarEvent	57
WebradioManager.DayWeek	61
Form	
WebradioManager.AdminView	32
WebradioManager.SelectionView	78
WebradioManager.IController	65
WebradioManagerAdminController	13
WebradioManager.SelectionController	74
WebradioManager.Playlist	67
WebradioManager.PlaylistAd	71
WebradioManager.PlaylistMusic	72
WebradioManager.Webradio	84
WebradioManager.WebradioCalendar	87
WebradioManager.WebradioListener	89
WebradioManager.WebradioServer	91
WebradioManager.WebradioServerStats	96
WebradioManager.WebradioTranscoder	98
WebradioManager.TranscoderAacPlus	80
WebradioManager.TranscoderMp3	82
WebradioManager.WMModel	107

Chapter 3

Class Index

3.1 Class List

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

WebradioManager.Ad	An ad audio file	11
WebradioManagerAdminController	A controller for handling admin view	13
WebradioManager.AdminView	An admin view	32
WebradioManager.AudioFile	An audio file. Abstract class	34
WebradioManager.Bdd	A bdd connection	38
WebradioManager.BddControls	53
WebradioManager.CalendarEvent	A calendar event	57
WebradioManager.DayWeek	A week with boolean values for each day. Uses for store which day is selected for an CalendarEvent	61
WebradioManager.EventAppointment	An event appointment. Add 2 properties to the original Appointment class (from Calendar library)	64
WebradioManager.IController	Interface for controller	65
WebradioManager.Music	A music	66
WebradioManager.Playlist	A playlist. Abstract class	67
WebradioManager.PlaylistAd	A playlist ad	71
WebradioManager.PlaylistMusic	A playlist music	72
WebradioManager.SelectionController	A controller for SelectionView	74
WebradioManager.SelectionView	A selection view	78
WebradioManager.TranscoderAacPlus	A transcoder aac plus	80
WebradioManager.TranscoderMp3	A transcoder mp3	82
WebradioManager.Webradio	A webradio	84

WebradioManager.WebradioCalendar	87
A webradio calendar	
WebradioManager.WebradioListener	89
A webradio listener	
WebradioManager.WebradioServer	91
A shoutcast webradio server	
WebradioManager.WebradioServerStats	96
A webradio server statistics	
WebradioManager.WebradioTranscoder	98
A webradio transcoder	
WebradioManager.WMModel	107
A data Model for the WebradioManager project	

Chapter 4

File Index

4.1 File List

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

Ad.cs	Implements the ad class	131
AdminController.cs	Implements the admin controller class	132
AdminView.cs	Implements the admin view class	135
AudioFile.cs	Implements the audio file class	150
AudioType.cs	Implements the audio type enum	152
Bdd.cs	Implements the bdd class	153
BddControls.cs	??
CalendarEvent.cs	Implements the calendar event class	162
DayWeek.cs	Implements the day week struct	165
EventAppointment.cs	Implements the event appointment class	166
IController.cs	Declares the IController interface	167
Music.cs	Implements the music class	168
Playlist.cs	Implements the playlist class	168
PlaylistAd.cs	Implements the playlist ad class	170
PlaylistMusic.cs	Implements the playlist music class	171
Program.cs	??
SelectionController.cs	Implements the selection controller class	171
SelectionView.cs	Implements the selection view class	173
SelectionView.Designer.cs	??
StreamType.cs	Implements the stream type class	175

TranscoderAacPlus.cs	Implements the transcoder aac plus class	175
TranscoderMp3.cs	Implements the transcoder mp 3 class	176
Webradio.cs	Implements the webradio class	177
WebradioCalendar.cs	Implements the webradio calendar class	178
WebradioListener.cs	Implements the webradio listener class	180
WebradioServer.cs	Implements the webradioserver class	181
WebradioServerStats.cs	Implements the webradio server statistics class	185
WebradioTranscoder.cs	Implements the webradio transcoder class	186
WMModel.cs	Implements the wm model class	191

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.

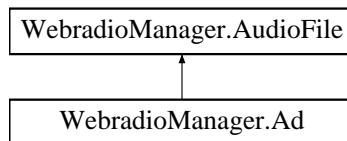
Chapter 6

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

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

<i>filename</i>	Filename of the audio file.
<i>title</i>	The title.
<i>artist</i>	The artist.
<i>album</i>	The album.
<i>year</i>	The year.
<i>label</i>	The label.
<i>duration</i>	The duration.
<i>gender</i>	The gender.

Definition at line 66 of file [Ad.cs](#).

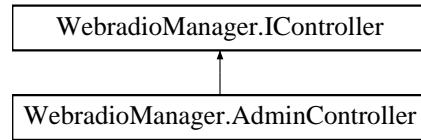
The documentation for this class was generated from the following file:

- [Ad.cs](#)

6.2 WebradioManagerAdminController Class Reference

A controller for handling admin view.

Inheritance diagram for WebradioManagerAdminController:



Public Member Functions

- **AdminController** (int id, [WMMModel](#) 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.
- **WMMModel 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 WebradioManagerAdminController(int id, WMMModel model)

Constructor.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>id</i>	The webradio identifier.
<i>model</i>	The model.

Definition at line 76 of file [AdminController.cs](#).

6.2.3 Member Function Documentation

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

Adds to the playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>playlist</i>	The playlist.
<i>audioFiles</i>	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 WebradioManagerAdminController.CheckFolders (int webradioid)

Check folders.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	Identifier of the webradio.
-------------------	-----------------------------

Definition at line 159 of file [AdminController.cs](#).

6.2.3.3 public bool WebradioManagerAdminController.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 WebradioManagerAdminController.ClearHistory (int transcoderId)

Clears the transcoder's history.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>transcoderId</i>	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 WebradioManagerAdminController.CreateEvent (CalendarEvent newEvent, int webradioid)

Creates an event.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>newEvent</i>	The new event.
<i>webradioid</i>	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 WebradioManagerAdminController.CreatePlaylist (string name, string webradioName, int webradioid, AudioType type)

Creates a playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>name</i>	The playlis's name.
<i>webradioName</i>	Name of the webradio.
<i>webradioid</i>	Identifier of the webradio.
<i>type</i>	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 WebradioManagerAdminController.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

<i>name</i>	The name.
<i>st</i>	The streamtype.
<i>sampleRate</i>	The sample rate.
<i>bitrate</i>	The bitrate.
<i>url</i>	URL of the stream.
<i>ip</i>	The IP.
<i>port</i>	The port.
<i>adminport</i>	The adminport.
<i>password</i>	The password.
<i>webradioid</i>	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 WebradioManagerAdminController.DeleteAudioFile (int *id*, string *filename*)

Deletes the audio file.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>id</i>	The identifier.
<i>filename</i>	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 WebradioManagerAdminController.DeleteEvent (*CalendarEvent aEvent*, int *webradioid*)

Deletes the event.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>aEvent</i>	The event.
<i>webradioid</i>	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 WebradioManagerAdminController.DeletePlaylist (*Playlist playlist*, int *webradioid*)

Deletes the playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>playlist</i>	The playlist.
<i>webradioid</i>	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 WebradioManagerAdminController.DeleteTranscoder (*WebradioTranscoder transcoder*, int *webradioid*)

Deletes the transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>transcoder</i>	The transcoder.
<i>webradioId</i>	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 WebradioManagerAdminController.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 WebradioManagerAdminController.GenerateAllConfigs(int webradioId)

Generates all configs.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioId</i>	Identifier of the webradio.
-------------------	-----------------------------

Definition at line 530 of file [AdminController.cs](#).

6.2.3.14 public bool WebradioManagerAdminController.GenerateHistory(int webradioId, string transcoderName, int transcoderId, string outputfilename)

Generates a history.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioId</i>	Identifier of the webradio.
<i>transcoderName</i>	Name of the transcoder.
<i>transcoderId</i>	Identifier of the transcoder.
<i>outputFilename</i>	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 WebradioManagerAdminController.GeneratePlaylist (string *name*, TimeSpan *duration*, AudioType *type*, string *gender*, int *webradioId*, string *webradioName*)

Generates a playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>name</i>	The name.
<i>duration</i>	The duration.
<i>type</i>	The type.
<i>gender</i>	The gender.
<i>webradioId</i>	Identifier of the webradio.
<i>webradioName</i>	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 WebradioManagerAdminController.GetAudioFileByFilename (string *filename*)

Gets audio file by filename.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>filename</i>	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

<i>playlist</i>	The playlist.
-----------------	---------------

Returns

The playlist content.

Definition at line 311 of file [AdminController.cs](#).

6.2.3.20 public List< WebradioListener > WebradioManagerAdminController.GetServerListeners (int webradioid)

Get server's listeners.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	Identifier of the webradio.
-------------------	-----------------------------

Returns

A List<[WebradioListener](#)>

Definition at line 773 of file [AdminController.cs](#).

6.2.3.21 public int WebradioManagerAdminController.GetSimilarViewCount (int webradioid)

Gets similar view count.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	Identifier of the webradio.
-------------------	-----------------------------

Returns

The similar view count.

Definition at line 825 of file [AdminController.cs](#).

6.2.3.22 public Webradio WebradioManagerAdminController.GetWebradio (int *id*)

Gets a webradio.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>id</i>	The webradio identifier.
-----------	--------------------------

Returns

The webradio.

Definition at line 111 of file [AdminController.cs](#).

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

Import files to library.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>filenames</i>	The filenames.
<i>type</i>	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 WebradioManagerAdminController.ModifyWebradioName (string *name*, int *webradioid*)

Modify webradio's name.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>name</i>	The name.
<i>webradioid</i>	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 WebradioManagerAdminController.RemoveFromPlaylist (Playlist *playlist*, Dictionary< int, string > *audioFiles*)

Removes from playlist.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>playlist</i>	The playlist.
<i>audioFiles</i>	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 WebradioManagerAdminController.ShowServerWebAdmin (int *webradioid*)

Shows the server web admin.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	Identifier of the webradio.
-------------------	-----------------------------

Definition at line 622 of file [AdminController.cs](#).

6.2.3.27 public void WebradioManagerAdminController.ShowServerWebInterface (int *webradioid*)

Shows the server web interface.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	Identifier of the webradio.
-------------------	-----------------------------

Definition at line [606](#) of file [AdminController.cs](#).

6.2.3.28 public bool WebradioManagerAdminController.StartServer (int *webradioid*, bool *debug*)

Starts a server.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	Identifier of the webradio.
<i>debug</i>	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 WebradioManagerAdminController.StartTranscoder (WebradioTranscoder *transcoder*, bool *debug*, int *webradioid*)

Starts a transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>transcoder</i>	The transcoder.
<i>debug</i>	Debug or not.

<i>webradioid</i>	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 WebradioManagerAdminController.StopAllTranscoders (int *webradioid*)

Stops all transcoders.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	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 WebradioManagerAdminController.StopServer (int *webradioid*)

Stops a server.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	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 WebradioManagerAdminController.StopTranscoder (WebradioTranscoder *transcoder*, int *webradioid*)

Stops a transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>transcoder</i>	The transcoder.
<i>webradioid</i>	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 WebradioManagerAdminController.TranscoderCapture (bool *active*, string *device*, WebradioTranscoder *transcoder*, int *webradioid*)

Transcoder capture mode set.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>active</i>	true to active.
<i>device</i>	The device's name.
<i>transcoder</i>	The transcoder.
<i>webradioid</i>	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 WebradioManagerAdminController.TranscoderNextTrack (WebradioTranscoder *transcoder*)

Transcoder next track.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>transcoder</i>	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 WebradioManagerAdminController.UpdateAudioFile (*AudioFile file*)

Updates the audio file described by file.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>file</i>	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 WebradioManagerAdminController.UpdateEvent (*CalendarEvent aEvent*, int *webradioid*)

Updates the event.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>aEvent</i>	The event.
<i>webradioid</i>	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 WebradioManagerAdminController.UpdateServer (bool *debug*, int *port*, string *password*, string *adminPassword*, int *maxListener*, int *webradioid*)

Updates the server.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>debug</i>	Debug or not.
<i>port</i>	The port.
<i>password</i>	The password.
<i>adminPassword</i>	The admin password.
<i>maxListener</i>	The maximum listener.
<i>webradioid</i>	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 WebradioManagerAdminController.UpdateServerStats (int *webradioid*)

Updates the server statistics.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>webradioid</i>	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 WebradioManagerAdminController.UpdateTranscoder (WebradioTranscoder *transcoder*, bool *debug*, int *webradioid*)

Updates the transcoder.

Author

Simon Menetrey

Date

23.05.2014

Parameters

<i>transcoder</i>	The transcoder.
<i>debug</i>	Debug or not.
<i>webradioid</i>	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 WebradioManagerAdminController.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 WebradioManagerAdminController.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 WebradioManagerAdminController.View [get], [set]

Gets or sets the view.

Returns

The view.

Definition at line 42 of file [AdminController.cs](#).

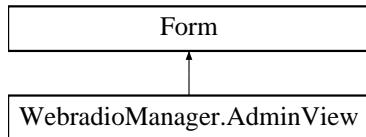
The documentation for this class was generated from the following file:

- [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

<i>idWebradio</i>	The webradio's identifier.
<i>controller</i>	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](#).

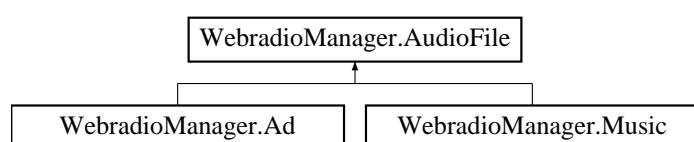
The documentation for this class was generated from the following file:

- [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

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

<i>filename</i>	Filename of the audiofile.
<i>title</i>	The title.
<i>artist</i>	The artist.
<i>album</i>	The album.
<i>year</i>	The year.
<i>label</i>	The label.
<i>duration</i>	The duration.
<i>gender</i>	The gender.
<i>audiotype</i>	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

<i>file</i>	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

<i>newEvent</i>	The new event.
<i>calendarId</i>	Identifier for the calendar.
<i>playlistId</i>	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

<i>gender</i>	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

<i>playlist</i>	The playlist.
<i>audioFilesId</i>	Identifier for the audio files.
<i>webradioid</i>	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

<i>transcoderId</i>	Identifier for the transcoder.
<i>date</i>	The date Date/Time.
<i>filename</i>	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

<i>idAudioFile</i>	The identifier audio file.
<i>idPlaylist</i>	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

<i>transcoder</i>	The transcoder.
<i>webradioid</i>	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

<i>webradio</i>	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

<i>filename</i>	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

<i>transcoderId</i>	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

<i>playlist</i>	The playlist.
<i>webradioId</i>	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

<i>id</i>	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

<i>aEvent</i>	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

<i>id</i>	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

<i>transcoderId</i>	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

<i>id</i>	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

<i>aEvent</i>	The event.
<i>calendarId</i>	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

<i>gender</i>	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

<i>transcoderId</i>	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

<i>name</i>	The name.
<i>webradioid</i>	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

<i>idAudioFile</i>	The identifier audio file.
<i>idPlaylist</i>	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

<i>name</i>	The name.
<i>webradioid</i>	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

<i>file</i>	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

<i>aEvent</i>	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

<i>oldName</i>	Webradio's old name.
<i>newName</i>	Webradio's new name.
<i>webradio</i>	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

<i>port</i>	The port.
<i>password</i>	The password.
<i>adminPassword</i>	The admin password.
<i>maxListener</i>	The number of maximum listener.
<i>webradioid</i>	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

<i>transcoder</i>	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

<i>name</i>	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

<code>table</code>	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

<i>tableName</i>	The table from which to delete.
<i>where</i>	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.ExecuteReader (string *sql*)**Parameters**

<i>sql</i>

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

<i>sql</i>	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

<i>sql</i>	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

<i>sql</i>	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

<i>tableName</i>	The table into which we insert the data.
<i>data</i>	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

<i>tableName</i>	The table to update.
<i>data</i>	A dictionary containing Column names and their new values.
<i>where</i>	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 repeat 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

<i>name</i>	The name.
<i>starttime</i>	The starttime.
<i>duration</i>	The duration.
<i>repeat</i>	The repeat.
<i>priority</i>	The priority.

<i>shuffle</i>	true to shuffle.
<i>loopatend</i>	true to loopatend.
<i>playlist</i>	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

<i>id</i>	The identifier.
<i>name</i>	The name.
<i>starttime</i>	The starttime.
<i>duration</i>	The duration.
<i>repeat</i>	The repeat.
<i>priority</i>	The priority.
<i>shuffle</i>	true to shuffle.
<i>loopatend</i>	true to loopatend.
<i>playlist</i>	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 repeat value. http://wiki.winamp.com/wiki/SOCAST_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](#).

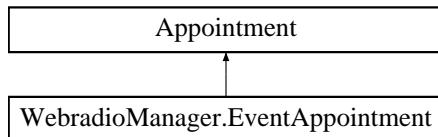
The documentation for this struct was generated from the following file:

- [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](#).

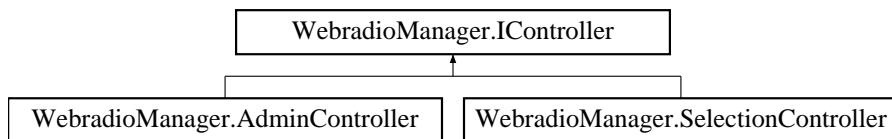
The documentation for this class was generated from the following file:

- [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](#).

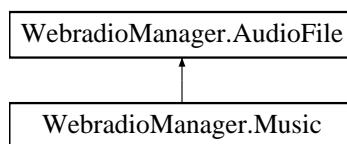
The documentation for this interface was generated from the following file:

- [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

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

<i>filename</i>	Filename of the audiofile.
<i>title</i>	The title.
<i>artist</i>	The artist.
<i>album</i>	The album.
<i>year</i>	The year.
<i>label</i>	The label.
<i>duration</i>	The duration.
<i>gender</i>	The gender.

Definition at line 66 of file [Music.cs](#).

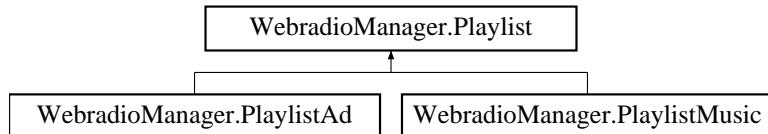
The documentation for this class was generated from the following file:

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

Constructor.

Author

Simon Menetrey

Date

26.05.2014

Parameters

<i>name</i>	The name.
<i>filename</i>	Filename of the file.
<i>type</i>	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

<i>id</i>	The identifier.
<i>name</i>	The name.
<i>filename</i>	Filename of the file.
<i>type</i>	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](#).

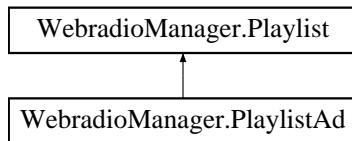
The documentation for this class was generated from the following file:

- [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

<i>id</i>	The identifier.
<i>name</i>	The name.
<i>filename</i>	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

<i>name</i>	The name.
<i>filename</i>	Filenome of the file.

Definition at line 51 of file [PlaylistAd.cs](#).

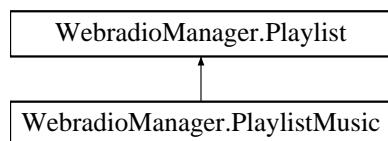
The documentation for this class was generated from the following file:

- [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

<i>id</i>	The identifier.
<i>name</i>	The name.
<i>filename</i>	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

<i>name</i>	The name.
-------------	-----------

<code>filename</code>	Filename of the file.
-----------------------	-----------------------

Definition at line 52 of file [PlaylistMusic.cs](#).

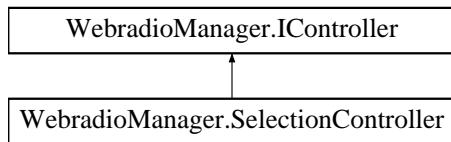
The documentation for this class was generated from the following file:

- [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

<code>view</code>	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

<code>name</code>	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

<i>id</i>	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

<i>id</i>	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

<i>id</i>	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 WMMModel 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](#).

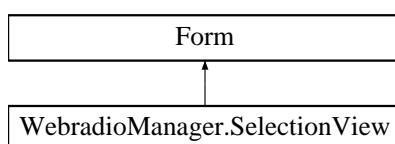
The documentation for this class was generated from the following file:

- [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

<i>disposing</i>	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](#).

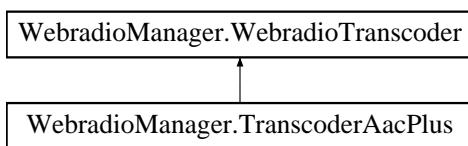
The documentation for this class was generated from the following files:

- [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

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

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

Definition at line 70 of file [TranscoderAacPlus.cs](#).

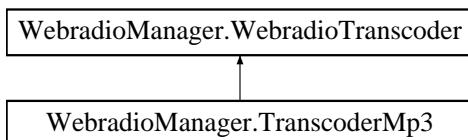
The documentation for this class was generated from the following file:

- [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

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

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

<i>password</i>	The password.
<i>configFilename</i>	Filename of the configuration file.
<i>logFilename</i>	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

<i>name</i>	The name.
<i>id</i>	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

<i>name</i>	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.

Returns

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

<i>id</i>	The identifier.
<i>filename</i>	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

<i>filename</i>	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

<code><i>hostname</i></code>	The hostname.
<code><i>useragent</i></code>	The useragent.
<code><i>connectiontime</i></code>	The connection time.
<code><i>uid</i></code>	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

<i>port</i>	The port.
<i>logfilename</i>	The filename of the log file.
<i>configfilename</i>	The filename of the configuration file.
<i>password</i>	The password.
<i>adminPassword</i>	The admin password.
<i>maxlistener</i>	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

<i>debug</i>	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](#).

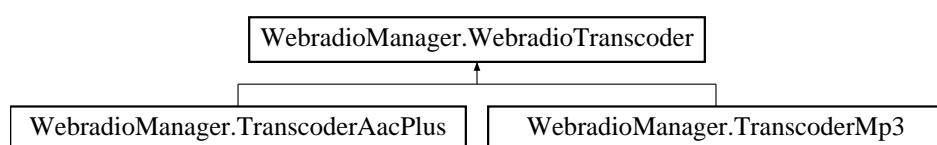
The documentation for this class was generated from the following file:

- [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[] **AvailableSampleRates** [get, set]
- static int[] **AvailableBitrates** [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

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

<i>password</i>	The password.
<i>configFilename</i>	Filename of the configuration file.
<i>logFilename</i>	Filename of the log file.
<i>st</i>	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

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

<i>playlists</i>	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

<i>active</i>	true to active.
<i>device</i>	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

<i>debug</i>	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

<i>observer</i>	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

<i>playlist</i>	The playlist.
<i>audioFiles</i>	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

<i>webradioid</i>	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

<i>transcoderId</i>	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

<i>newEvent</i>	The new event.
<i>webradioid</i>	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

	<i>name</i>	The name.
	<i>webradioName</i>	Name of the webradio.
	<i>webradioid</i>	Identifier for the webradio.
	<i>type</i>	The type.
<i>out</i>	<i>newPlaylist</i>	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

<i>name</i>	The name.
<i>st</i>	The st.
<i>sampleRate</i>	The sample rate.
<i>bitrate</i>	The bitrate.
<i>url</i>	URL of the document.
<i>ip</i>	The IP.
<i>port</i>	The port.
<i>adminport</i>	The administration port.
<i>password</i>	The password.
<i>webradioid</i>	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

<i>name</i>	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

<i>id</i>	The identifier.
<i>audioFilename</i>	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

<i>aEvent</i>	The event.
<i>webradioid</i>	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

<i>playlist</i>	The playlist.
<i>webradioid</i>	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

<i>transcoder</i>	The transcoder.
<i>webradioid</i>	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

<i>id</i>	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

<i>id</i>	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

<i>webradioid</i>	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

<i>webradioid</i>	Identifier for the webradio.
<i>transcoderName</i>	Name of the transcoder.
<i>transcoderId</i>	Identifier for the transcoder.
<i>outputFilename</i>	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

<i>name</i>	The name.
<i>duration</i>	The duration.
<i>type</i>	The type.
<i>gender</i>	The gender.
<i>webradioid</i>	Identifier for the webradio.
<i>webradioName</i>	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

<i>filename</i>	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

<i>playlist</i>	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

<i>webradioId</i>	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

<i>id</i>	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

<i>name</i>	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

<i>filenames</i>	The filenames array.
<i>type</i>	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

<i>newName</i>	Name of the new webradio.
<i>webradioid</i>	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

<i>audioFiles</i>	The audio files.
<i>playlist</i>	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

<i>observer</i>	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

<i>webradioid</i>	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

<i>webradioid</i>	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

<i>webradioid</i>	Identifier for the webradio.
<i>debug</i>	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

<i>transcoder</i>	The transcoder.
<i>debug</i>	true to debug.
<i>webradioid</i>	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

<i>webradioid</i>	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

<i>webradioid</i>	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

<i>transcoder</i>	The transcoder.
<i>webradioid</i>	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

<i>active</i>	true to active.
<i>device</i>	The device.
<i>transcoder</i>	The transcoder.
<i>webradioid</i>	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

<i>transcoder</i>	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

<i>file</i>	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

<i>aEvent</i>	The event.
<i>webradioid</i>	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

<i>debug</i>	true to debug.
<i>port</i>	The port.
<i>password</i>	The password.
<i>adminPassword</i>	The admin password.
<i>maxListener</i>	The maximum listener.
<i>webradioid</i>	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

<code>webradioid</code>	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

<code>webradioid</code>	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

<code>transcoder</code>	The transcoder.
<code>debug</code>	true to debug.
<code>webradioid</code>	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,
00044                 AudioType.Ad)
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,
00068                 AudioType.Ad)
00069         {
00070         }
00071     #endregion
```

```
00072     }
00073 }
```

7.3 AdminController.cs File Reference

Implements the admin controller class.

Classes

- class [WebradioManagerAdminController](#)
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,
00193          AudioType type)
00194      {
00195          return this.Model.ImportFilesToLibrary(filenames, type);
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,
00233          AudioType type)
00234      {
00235          Playlist newPlaylist;
00236          return this.Model.CreatePlaylist(name, webradioName, webradioId, type, out newPlaylist);
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>
00272          audioFiles)
00273      {
00274          if (playlist != null)
00275              return this.Model.AddToPlaylist(playlist, audioFiles);
00276          else
00277              return false;
00278      }
00293      public bool RemoveFromPlaylist(Playlist playlist, Dictionary<int, string>
00294          audioFiles)
00295      {
00296          return this.Model.RemoveFromPlaylist(audioFiles, playlist);
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,
00335          AudioType type, string gender, int webradioId, string webradioName)
00336      {
00337          return this.Model.GeneratePlaylist(name, duration, type, gender, webradioId, webradioName);
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,
00419             int bitrate, string url, IPAddress ip, int port, int adminport, string password, int webradioId)
00420         {
00421             return this.Model.CreateTranscoder(name, st, sampleRate, bitrate, url, ip, port, adminport,
00422                 password, webradioId);
00423         }
00424
00437         public bool DeleteTranscoder(WebradioTranscoder transcoder, int
00438             webradioId)
00439         {
00440             return this.Model.DeleteTranscoder(transcoder, webradioId);
00441         }
00442
00457         public bool UpdateTranscoder(WebradioTranscoder transcoder, bool
00458             debug, int webradioId)
00459         {
00460             return this.Model.UpdateTranscoder(transcoder, debug, webradioId);
00461         }
00477         public bool StartTranscoder(WebradioTranscoder transcoder, bool
00478             debug, int webradioId)
00479         {
00480             return this.Model.StartTranscoder(transcoder, debug, webradioId);
00481         }
00496         public bool StopTranscoder(WebradioTranscoder transcoder, int
00497             webradioId)
00498         {
00499             return this.Model.StopTranscoder(transcoder, webradioId);
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,
00554             int maxListener, int webradioId)
00555         {
00556             return this.Model.UpdateServer(debug, port, password, adminPassword, maxListener, webradioId);
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,
00680             string outputfilename)
00681         {
00682             return this.Model.GenerateHistory(webradioId, transcoderName, transcoderId, outputfilename);
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,
00756         WebradioTranscoder transcoder, int webradioId)
00757     {
00758         return this.Model.TranscoderCapture(active, device, transcoder, webradioId);
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 {
00020     public partial class AdminView : Form
00021     {
00022         #region Const
00023         // \brief The default search string.
00024         const string DEFAULT_SEARCH_STRING = "Search...";
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 AdminController _controller;
00032         // \brief The identifier of the current webradio.
00033         private int _idWebradio;
00034         // \brief The webradio's name.
00035         private string _nameWebradio;
00036         // \brief The events for calendar.
00037         List<EventAppointment> _events;
00038         #endregion
00039
00040         #region Properties
00041
00042         public string NameWebradio
00043         {
00044             get { return _nameWebradio; }
00045             set { _nameWebradio = value; }
00046         }
00047
00048         public List<EventAppointment> EventsCalendar
00049         {
00050             get { return _events; }
00051             set { _events = value; }
00052         }
00053
00054         public int IdWebradio
00055         {
00056             get { return _idWebradio; }
00057             set { _idWebradio = value; }
00058         }
00059
00060         public AdminController Controller
00061         {
00062             get { return _controller; }
00063             set { _controller = value; }
00064         }
00065         #endregion
00066
00067         #region Methods
00068
00069         public AdminView(int idWebradio, AdminController controller)
00070         {
00071             InitializeComponent();
00072             this.Controller = controller;
00073             this.IdWebradio = idWebradio;
00074             this.EventsCalendar = new List<EventAppointment>();
00075             this.UpdateAudioDevices();
00076         }
00077
00078         public void UpdateView()
00079         {
00080             int index;
00081             Webradio webradio = this.Controller.GetWebradio(this.IdWebradio);
00082             this.NameWebradio = webradio.Name;
00083             this.Text = "WebradioManager - " + this.NameWebradio;
00084             txbWebradioName.Text = this.NameWebradio;
00085             lblWebradioTitle.Text = this.NameWebradio;
00086             cmbTypePlaylist.SelectedIndex = 0;
00087             cmbTypePlaylistGenerate.SelectedIndex = 0;
00088             //LIBRARY
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150

```

```

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 :
00162     DEFAULT_SEARCH_STRING;
00163     txbSearchMusic.Text = (txbSearchMusic.Text != DEFAULT_SEARCH_STRING) ? txbSearchMusic.Text :
00164     DEFAULT_SEARCH_STRING;
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(webadio.Playlists.ToArray());
00176     foreach (Playlist playlist in webadio.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 webadio.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
00218                     EventAppointment();
00219
00220                 m_Appointment.StartDate = new DateTime(dvwTimetable.StartDate.Year, dvwTimetable.
00221                     StartDate.Month, (i + 1), ev.StartTime.Hours, ev.StartTime.Minutes, ev.
00222                     StartTime.Seconds);
00223                 m_Appointment.EndDate = m_Appointment.StartDate.Add(ev.Duration);
00224                 m_Appointment.Title = ev.Name + "(" + ev.Priority.ToString() + ")\r\n" + ev.
00225                     Playlist.Name + "\r\n" Shuffle : " + ((ev.Shuffle) ? "True" : "False");
00226                 if (ev.Playlist.Type == AudioType.Music)
00227                 {
00228                     m_Appointment.BorderColor = Color.Blue;
00229                     //m_Appointment.Color = Color.Blue;
00230                 }
00231                 else
00232                 {
00233                     m_Appointment.BorderColor = Color.Red;
00234                     //m_Appointment.Color = Color.Red;
00235                 }
00236             }
00237         }
00238     }

```

```

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(webadio.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.AvailableBitrates)
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.AvailableSampleRates)
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 webadio.Transcoders)
00271 {
00272     lsbStatus.Items.Add(transcoder.Name + " : " + ((transcoder.IsRunning()) ? "On" : "Off"));
00273     if (transcoder.IsRunning())
00274     {
00275         AudioFile file = this.Controller.GetAudioFileByFilename(
00276             transcoder.CurrentTrack);
00277         if (file != null)
00278         {
00279             string[] values = new string[] { transcoder.Name, file.Title, file.Artist,
00280                 file.Album };
00281             dgvCurrentTracks.Rows.Add(values);
00282         }
00283     }
00284 }
00285 //---
00286 //SERVER
00287 nudPortServer.Value = webadio.Server.Port;
00288 txbLocalServerPassword.Text = webadio.Server.Password;
00289 txbLocalServerAdminPassword.Text = webadio.Server.AdminPassword;
00290 nudMaxListener.Value = webadio.Server.MaxListener;
00291 bool running = webadio.Server.IsRunning();
00292 lblStatusServer.Text = (running) ? "On" : "Off";
00293 lblStatusServer.ForeColor = (running) ? Color.Green : Color.Red;
00294 btnStartServer.Enabled = (running) ? false : true;
00295 btnStopServer.Enabled = (running) ? true : false;
00296
00297 if (webadio.Server.Stats != null)
00298     this.ShowServerStats(webadio.Server.Stats);
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 }
00329 private void UpdateAudioDevices()
00330 {
00331     ManagementObjectSearcher objSearcher = new ManagementObjectSearcher("SELECT * FROM
00332 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
00347 void dwvTimetable_SelectionChanged(object sender, EventArgs e)
00348 {
00349     ckbMonday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Monday) ? true : false;
00350     ckbTuesday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Tuesday) ? true :
00351         false;
00352     ckbWednesday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Wednesday) ? true :
00353         false;
00354     ckbThursday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Thursday) ? true :
00355         false;
00356     ckbFriday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Friday) ? true : false;
00357     ckbSaturday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Saturday) ? true :
00358         false;
00359     ckbSunday.Checked = (dwvTimetable.SelectionStart.DayOfWeek == DayOfWeek.Sunday) ? true : false;
00360
00361     TimeSpan start = new TimeSpan();
00362     TimeSpan end = new TimeSpan();
00363     if (dwvTimetable.Selection == SelectionType.Appointment)
00364     {
00365         start = new TimeSpan(dwvTimetable.SelectedAppointment.StartDate.Hour, dwvTimetable.
00366             SelectedAppointment.StartDate.Minute, dwvTimetable.SelectedAppointment.StartDate.Second);
00367         end = new TimeSpan(dwvTimetable.SelectedAppointment.EndDate.Hour, dwvTimetable.
00368             SelectedAppointment.EndDate.Minute, dwvTimetable.SelectedAppointment.EndDate.Second);
00369     }
00370     else if (dwvTimetable.Selection == SelectionType.DateRange)
00371     {
00372         start = new TimeSpan(dwvTimetable.SelectionStart.Hour, dwvTimetable.SelectionStart.Minute,
00373             dwvTimetable.SelectionStart.Second);
00374         end = new TimeSpan(dwvTimetable.SelectionEnd.Hour, dwvTimetable.SelectionEnd.Minute,
00375             dwvTimetable.SelectionEnd.Second);
00376     }
00377
00378     mtbStartTime.Text = start.ToString();
00379     mtbDuration.Text = (end - start).ToString();
00380 }
00381
00382
00383 private void dwvTimetable_ResolveAppointments(object sender, ResolveAppointmentsEventArgs args)
00384 {
00385     List<Appointment> m_Apps = new List<Appointment>();
00386     foreach (Appointment m_App in this.EventsCalendar)
00387         m_Apps.Add(m_App);
00388
00389     args.Appointments = m_Apps;
00390 }
00391
00392 private void AdminView_FormClosing(object sender, FormClosingEventArgs e)
00393 {
00394     if (this.Controller.GetSimilarViewCount(this.IdWebradio) == 1)
00395     {
00396         if (MessageBox.Show("All transcoders and the server will be shutting down. Are you sure ?", "Close", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
00397         {
00398             if (this.Controller.StopAllTranscoders(this.IdWebradio))
00399                 this.Controller.FormClose();
00400             else
00401             {
00402                 MessageBox.Show("An error has occurred", "Error");
00403                 e.Cancel = true;
00404             }
00405         }
00406         else
00407             e.Cancel = true;
00408     }
00409 }
00410
00411 private void ImportFolder_Click(object sender, EventArgs e)
00412 {
00413     AudioType type;
00414     string[] filenames;
00415     if ((sender as Button).Tag.ToString() == AudioType.Music.ToString())
00416         type = AudioType.Music;
00417 }
```

```

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 occurred", "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 occurred", "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() ==
00569     AudioType.Music.ToString()) ? dgvMusics.SelectedRows : dgvAds.SelectedRows;
00570     foreach (DataGridViewRow row in rows)
00571     {
00572         //Simple check if selected row is not the last empty row
00573         if (row.Cells[0].Value != null)
00574             if (!this.Controller.DeleteAudioFile(int.Parse(row.Cells["
00575     colId" + type.ToString()].Value.ToString()), row.Cells["colPath" + type.ToString()].Value.ToString()))
00576                 state = false;
00577
00578         if (state)
00579             this.UpdateView();
00580         else
00581             MessageBox.Show("An error occurred");
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 :
00611             dgvAds.Rows))
00612         {
00613             foreach (DataGridViewCell cell in row.Cells)
00614             {
00615                 if (cell.Value.ToString().ToLower().Contains(searchString))
00616                 {
00617                     valid = true;
00618                     break;
00619                 }
00620                 row.Visible = (valid) ? true : false;
00621                 valid = false;
00622             }
00623         }
00624     }
00625
00626     private void btnCreatePlaylist_Click(object sender, EventArgs e)
00627     {
00628         if (!string.IsNullOrEmpty(txbPlaylistName.Text.Trim()) && txbPlaylistName.Text.Length <=
00629             MAX_NAME_LENGTH)
00630         {
00631             if (!this.Controller.CreatePlaylist(txbPlaylistName.Text, this.
00632                 NameWebradio, this.IdWebradio, (cmbTypePlaylist.SelectedItem.ToString() == "Music") ?
00633                 AudioType.Music : AudioType.Ad))
00634                 MessageBox.Show("Playlist already exist or the name is invalid", "Error");
00635             }
00636             else
00637                 MessageBox.Show("Please enter a valid playlist's name. (1-" + MAX_NAME_LENGTH + "
00638                     characters)", "Empty name");
00639         }
00640
00641     private void btnDeletePlaylistClick(object sender, EventArgs e)
00642     {
00643         AudioType type;
00644         type = ((sender as Button).Tag.ToString() == AudioType.Music.ToString()) ? AudioType.Music :
00645             AudioType.Ad;
00646
00647         if (((type == AudioType.Music) ? lsbPlaylistsMusic.SelectedIndex :
00648             lsbPlaylistsAd.SelectedIndex) >= 0)
00649             this.Controller.DeletePlaylist((Playlist)((type == AudioType.Music) ?
00650                 lsbPlaylistsMusic.SelectedItem : lsbPlaylistsAd.SelectedItem), this.IdWebradio);
00651             else
00652                 MessageBox.Show("Please select a playlist to delete.", "No playlist selected");
00653         }
00654
00655     private void btnAddToClick(object sender, EventArgs e)
00656     {
00657         AudioType type = ((sender as Button).Tag.ToString() == AudioType.Music.ToString()) ?
00658             AudioType.Music : AudioType.Ad;
00659         Dictionary<int, string> audioFiles = new Dictionary<int, string>();
00660         DataGridViewSelectedRowCollection rows = ((type == AudioType.Music) ? dgvMusics.SelectedRows :
00661             dgvAds.SelectedRows);
00662         this.Cursor = Cursors.WaitCursor;
00663         foreach (DataGridViewRow row in rows)
00664         {
00665             //Simple check if selected row is not the last empty row
00666             if (row.Cells[0].Value != null)
00667                 audioFiles.Add(int.Parse(row.Cells["colId" + type.ToString()].Value.ToString()), row.
00668                     Cells["colPath" + type.ToString()].Value.ToString());
00669             }
00670             this.Cursor = Cursors.Default;
00671             if (this.Controller.AddToPlaylist((Playlist)((type ==
00672                 AudioType.Music) ? cmbPlaylistsMusic.SelectedItem : cmbPlaylistsAd.SelectedItem),
00673                     audioFiles))
00674                 this.UpdateView();
00675             }
00676
00677     private void lsbPlaylistsSelectedIndexChanged(object sender, EventArgs e)
00678     {
00679         ListBox lsb = sender as ListBox;
00680         AudioType type = (lsb.Tag.ToString() == AudioType.Music.ToString()) ?
00681             AudioType.Music : AudioType.Ad;
00682         btnRemoveFromPlaylist.Tag = type;
00683         if (lsb.SelectedIndex >= 0)
00684         {
00685             this.GetPlaylistContent((Playlist)lsb.SelectedItem);
00686         }
00687     }
00688 }
00689

```

```

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
00754     private void btnRemoveFromPlaylist_Click(object sender, EventArgs e)
00755     {
00756         if (btnRemoveFromPlaylist.Tag == null)
00757         {
00758             MessageBox.Show("No playlist selected", "No playlist");
00759             return;
00760         }
00761         AudioType type = (AudioType)btnRemoveFromPlaylist.Tag;
00762         Dictionary<int, string> audioFiles = new Dictionary<int, string>();
00763         this.Cursor = Cursors.WaitCursor;
00764         foreach (DataGridViewRow row in dgvPlaylistContent.SelectedRows)
00765         {
00766             //Simple check if selected row is not the last empty row
00767             if (row.Cells[0].Value != null)
00768                 audioFiles.Add(int.Parse(row.Cells["colIdPlaylist"].Value.ToString()), row.Cells["colPathPlaylist"].Value.ToString());
00769         }
00770         this.Cursor = Cursors.Default;
00771         Playlist selectedPlaylist = (Playlist)((type == AudioType.Music) ?
00772             lsbPlaylistsMusic.SelectedItem : lsbPlaylistsAd.SelectedItem);
00773         if (!this.Controller.RemoveFromPlaylist(selectedPlaylist,
00774             audioFiles))
00775             MessageBox.Show("An error occurred", "Error");
00776         else
00777         {
00778             this.GetPlaylistContent(selectedPlaylist);
00779         }
00780     }
00781
00782     private void txbSearchPlaylistContent_TextChanged(object sender, EventArgs e)
00783     {
00784         string searchString = "";
00785         bool valid = false;
00786         if (!string.IsNullOrEmpty(txbSearchPlaylistContent.Text) && txbSearchPlaylistContent.Text != DEFAULT_SEARCH_STRING)
00787         {
00788             searchString = txbSearchPlaylistContent.Text.ToLower();
00789             foreach (DataGridViewRow row in dgvPlaylistContent.Rows)
00790             {
00791                 foreach (DataGridViewCell cell in row.Cells)
00792                 {
00793                     if (cell.Value.ToString().ToLower().Contains(searchString))
00794                     {
00795                         valid = true;
00796                         break;
00797                     }
00798                 }
00799                 row.Visible = (valid) ? true : false;
00800                 valid = false;
00801             }
00802         }
00803     }
00804
00805     private void cmbTypePlaylistGenerate_SelectedIndexChanged(object sender, EventArgs e)
00806     {
00807         if (cmbTypePlaylistGenerate.SelectedItem.ToString() == AudioType.Ad.ToString())
00808             cmbGenderGenerate.Enabled = false;
00809         else
00810             cmbGenderGenerate.Enabled = true;
00811     }
00812
00813     private void btnGeneratePlaylist_Click(object sender, EventArgs e)
00814     {
00815         if (!string.IsNullOrEmpty(txbPlaylistNameGenerate.Text.Trim()) && txbPlaylistNameGenerate.Text.
00816             Length <= MAX_NAME_LENGTH)
00817         {
00818             TimeSpan duration = new TimeSpan(0, (int)nudDurationGenerate.Value, 0);
00819             if (!this.Controller.GeneratePlaylist(txbPlaylistNameGenerate.
00820             Text, duration,
00821                 (cmbTypePlaylistGenerate.SelectedItem.ToString() == AudioType.Music.ToString()) ?
00822                 AudioType.Music : AudioType.Ad,
00823                 cmbGenderGenerate.SelectedItem.ToString(), this.IdWebradio, this.
00824             );
00825         }
00826     }

```

```

        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
00879     private void ckbCheckedChanged(object sender, EventArgs e)
00880     {
00881         if ((sender as CheckBox).Name == "ckbAll")
00882         {
00883             ckbMonday.Checked = ckbAll.Checked;
00884             ckbTuesday.Checked = ckbAll.Checked;
00885             ckbWednesday.Checked = ckbAll.Checked;
00886             ckbThursday.Checked = ckbAll.Checked;
00887             ckbFriday.Checked = ckbAll.Checked;
00888             ckbSaturday.Checked = ckbAll.Checked;
00889             ckbSunday.Checked = ckbAll.Checked;
00890         }
00891     }
00892
00893     private void btnCreateEvent_Click(object sender, EventArgs e)
00894     {
00895         if (!string.IsNullOrEmpty(txbEventName.Text.Trim()) && txbEventName.Text.Length <=
MAX_NAME_LENGTH)
00896         {
00897             TimeSpan start = new TimeSpan();
00898             if (!TimeSpan.TryParse(mtbStartTime.Text, out start))
00899             {
00900                 MessageBox.Show("Start time format is not correct.", "Error");
00901                 return;
00902             }
00903             TimeSpan duration = new TimeSpan();
00904             if (!TimeSpan.TryParse(mtbDuration.Text, out duration))
00905             {
00906                 MessageBox.Show("Duration time format is not correct.", "Error");
00907                 return;
00908             }
00909             TimeSpan minimumDuration = new TimeSpan(0, 1, 0);
00910             if (duration >= minimumDuration)
00911             {
00912                 int repeat = this.GetRepeatValue();
00913                 if (repeat > 0)
00914                 {
00915                     CalendarEvent newEvent = new CalendarEvent(txbEventName.Text, start, duration,
repeat, (int)nudPriority.Value, ckbShuffle.Checked, true, (Playlist)cmbPlaylistEvent.SelectedItem);
00916                     if (!this.Controller.CreateEvent(newEvent, this.IdWebradio))
00917                         MessageBox.Show("Event already exists (with this name)", "Error");
00918                     else
00919                         MessageBox.Show("Please select a day", "Error");
00920                 }
00921             }
00922             else
00923                 MessageBox.Show("Duration must be longer or equal to " + minimumDuration.ToString());
00924         }
00925         else
00926             MessageBox.Show("Please enter a valid event's name. (1-" + MAX_NAME_LENGTH + " characters)"
, "Error");
00927     }
00928
00929     private void dvwTimetable_MouseUp(object sender, MouseEventArgs e)
00930     {
00931         if (dvwTimetable.Selection == SelectionType.Appointment)
00932         {
00933             EventAppointment app = (EventAppointment)dvwTimetable.SelectedAppointment;
00934             if (this.CheckMovePossible(app))
00935             {
00936                 CalendarEvent aEvent = app.EventObject;
00937                 TimeSpan start = new TimeSpan(app.StartDate.Hour, app.StartDate.Minute, app.StartDate.
Second);
00938                 TimeSpan end = new TimeSpan(app.EndDate.Hour, app.EndDate.Minute, app.EndDate.Second);
00939                 aEvent.StartTime = start;
00940                 aEvent.Duration = end - start;
00941                 aEvent.Repeat = this.GetRepeatValueFromAppointement(this.GetAllRelatedAppointment(app))
00942 ;
00943                 if (!this.Controller.UpdateEvent(aEvent, this.IdWebradio))
00944                     MessageBox.Show("An error occurred", "Error");
00945             }
00946         }
00947     }

```

```

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",
01119                         MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
01120                 {
01121                     if (!this.Controller.DeleteEvent((dvwTimetable.
01122                         SelectedAppointment as EventAppointment).EventObject, this.IdWebradio))
01123                         MessageBox.Show("An error occurred", "Error");
01124                 }
01125             }
01126
01139             private void btnCreateTranscoder_Click(object sender, EventArgs e)
01140             {
01141                 if (!string.IsNullOrEmpty(txbStreamName.Text.Trim()) && !string.IsNullOrEmpty(txbsServerPassword
01142 .Text.Trim()) && txbStreamName.Text.Length <= MAX_NAME_LENGTH)
01143                 {

```

```

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()) ?
01150 StreamType.MP3 : StreamType.AACPlus,
01151                     int.Parse(cmbSampleRate.SelectedItem.ToString()), int.Parse(cmbBitrate.SelectedItem
.ToString()), txbStreamUrl.Text, ip, (int)nudPort.Value, (int)nudAdminPort.Value, txbServerPassword.Text,
this.IdWebradio);
01152                     if (!result)
01153                         MessageBox.Show("Transcoder already exist or name is invalid", "Error");
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 occurred.", "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() ==
01272             StreamType.MP3.ToString()) ? StreamType.MP3 : StreamType.AACplus;
01273             if (!this.Controller.UpdateTranscoder(transcoder,
01274                 ckbTranscoderDebug.Checked, this.IdWebradio))
01275                 MessageBox.Show("An error has occured", "Error");
01276             else
01277                 {
01278                     this.ShowTranscoderInfos(transcoder);
01279                     MessageBox.Show("Update successful", "Succes");
01280                 }
01281             else
01282                 MessageBox.Show("IP address is invalid", "Error");
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     }
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
01309 .Checked, this.IdWebradio))
01310                 {
01311                     this.ShowTranscoderInfos(transcoder);
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     }
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                 }
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     }
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)
01369 lsbTranscoders.SelectedItem).LogFilename.Replace("/", "\\"));
01370             }
01371             catch
01372             {
01373                 MessageBox.Show("Impossible to access logfile", "Error");
01374             }
01375         }
01389         private void generateAllConfigsToolStripMenuItem_Click(object sender, EventArgs e)
01390         {
01391             this.Controller.GenerateAllConfigs(this.IdWebradio);
01392         }
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) ==
01409         {
01410             if (!string.IsNullOrEmpty(txbLocalServerPassword.Text.Trim()) && !string.IsNullOrEmpty(
01411                 txbLocalServerAdminPassword.Text.Trim()))
01412                 {
01413                     if (txbLocalServerPassword.Text != txbLocalServerAdminPassword.Text)
01414                         if (this.Controller.UpdateServer(ckbServerDebug.Checked, (int
01415 )nudPortServer.Value, txbLocalServerPassword.Text, txbLocalServerAdminPassword.Text, (int)nudMaxListener.
01416 Value, this.IdWebradio))
01417                             MessageBox.Show("Server informations updated.", "Success");
01418                         else
01419                             MessageBox.Show("An error has occurred", "Error");
01420                     else
01421                         MessageBox.Show("Password must be different", "Error");
01422                 }
01423             else
01424                 MessageBox.Show("Please enter a password and an admin password.", "Error");
01425         }
01426
01427     private void btnStartServer_Click(object sender, EventArgs e)
01428     {
01429         if (!this.Controller.StartServer(this.IdWebradio, ckbServerDebug.Checked))
01430             MessageBox.Show("An error has occurred.\n- Please terminate sc_server process.\n- Please
check that sc_serv.exe is in the shoutcast folder.", "Error");
01431     }
01432
01433     private void btnStopServer_Click(object sender, EventArgs e)
01434     {
01435         if (!this.Controller.StopServer(this.IdWebradio))
01436             MessageBox.Show("An error has occurred. Please terminate sc_server process.", "Error");
01437     }
01438
01439     private void btnShowWebInterface_Click(object sender, EventArgs e)
01440     {
01441         this.Controller.ShowServerWebInterface(this.IdWebradio);
01442     }
01443
01444     private void btnShowWebAdministration_Click(object sender, EventArgs e)
01445     {
01446         this.Controller.ShowServerWebAdmin(this.IdWebradio);
01447     }
01448
01449     private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
01450     {
01451         MessageBox.Show("WebradioManager V0.1\nSimon Menetrey\nTravail de diplôme 2014", "About",
01452             MessageBoxButtons.OK, MessageBoxIcon.Information);
01453     }
01454
01455     private void btnShowServerLog_Click(object sender, EventArgs e)
01456     {
01457         try
01458         {
01459             Process.Start("notepad", Directory.GetCurrentDirectory() + "\\\" + this.
01460 Controller.GetWebradio(this.IdWebradio).Server.LogFilename.Replace("/", "\\"));
01461         }
01462         catch
01463         {
01464             MessageBox.Show("Impossible to access logfile", "Error");
01465         }
01466     }
01467
01468     private void btnResolve_Click(object sender, EventArgs e)
01469     {
01470         bool editSection = false;
01471         if ((sender as Button).Tag != null)
01472             editSection = true;
01473         IPAddress ip;
01474         if (!GetResolvedConnexionIPAddress((editSection) ? txbServerIPEdit.Text : txbServerIP.Text, out
01475 ip))
01476             MessageBox.Show("Impossible to resolve this dns name", "Error");
01477         else
01478         {
01479             if (editSection)
01480                 txbServerIPEdit.Text = ip.ToString();
01481             else
01482                 txbServerIP.Text = ip.ToString();
01483         }
01484     }
01485
01486     private static bool GetResolvedConnexionIPAddress(string serverNameOrURL,
01487             out IPAddress resolvedIPAddress)
01488     {

```

```

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)
01652                 lsbTranscoders.SelectedItem))
01653                 MessageBox.Show("An error occurred.", "Error");
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.
01675                 SelectedItem).Id))
01676                 MessageBox.Show("History cleared", "Success");
01677             else
01678                 MessageBox.Show("An error has occurred", "Error");
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, ((
01705                     WebradioTranscoder)lsbTranscoders.SelectedItem).Name, ((WebradioTranscoder)lsbTranscoders.SelectedItem).Id, SFD.
01706                     FileName))

```

```

01705             {
01706                 if (MessageBox.Show("Do you want to view the pdf file ?", "Success",
01707                     MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
01708                     Process.Start(SFD.FileName);
01709                 else
01710                     MessageBox.Show("An error has occurred", "Error");
01711             }
01712         }
01713     }
01714     else
01715         MessageBox.Show("Please select a transcoder.", "Error");
01716     }
01717
01718     private void btnModifyName_Click(object sender, EventArgs e)
01719     {
01720         if (MessageBox.Show("To rename this webradio, all running process (transcoder and server) will
01721 be shutting down. Are you sure ?", "Process", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
01722 System.Windows.Forms.DialogResult.Yes)
01723         {
01724             if (!string.IsNullOrEmpty(txbWebradioName.Name.Trim()) && txbWebradioName.Name.Length <=
01725 MAX_NAME_LENGTH)
01726             {
01727                 if (this.Controller.ModifyWebradioName(txbWebradioName.Text
01728 , this.IdWebradio))
01729                     MessageBox.Show("Modification completed", "Success");
01730                 else
01731                     MessageBox.Show("This name is already used", "Error");
01732             }
01733             else
01734                 MessageBox.Show("Please enter a valid name. (1-" + MAX_NAME_LENGTH + " characters)", "Error");
01735         }
01736     }
01737
01738     private void dgvCellEndEdit(object sender, DataGridViewCellEventArgs e)
01739     {
01740         DataGridView dgv = (sender as DataGridView);
01741         DataGridViewRow row = dgv.Rows[e.RowIndex];
01742         AudioFile file;
01743         if (dgv.Tag.ToString() == AudioType.Music.ToString())
01744         {
01745             file = new Music(int.Parse(row.Cells["colId" + dgv.Tag.ToString()].Value.ToString()),
01746                             row.Cells["colPath" + dgv.Tag.ToString()].Value.ToString(),
01747                             row.Cells["colTitle" + dgv.Tag.ToString()].Value.ToString(),
01748                             row.Cells["colArtist" + dgv.Tag.ToString()].Value.ToString(),
01749                             row.Cells["colAlbum" + dgv.Tag.ToString()].Value.ToString(),
01750                             int.Parse(row.Cells["colYear" + dgv.Tag.ToString()].Value.ToString()),
01751                             row.Cells["colLabel" + dgv.Tag.ToString()].Value.ToString(),
01752                             new TimeSpan(),
01753                             row.Cells["colGender" + dgv.Tag.ToString()].Value.ToString());
01754         }
01755         else
01756         {
01757             file = new Ad(int.Parse(row.Cells["colId" + dgv.Tag.ToString()].Value.ToString()),
01758                             row.Cells["colPath" + dgv.Tag.ToString()].Value.ToString(),
01759                             row.Cells["colTitle" + dgv.Tag.ToString()].Value.ToString(),
01760                             row.Cells["colArtist" + dgv.Tag.ToString()].Value.ToString(),
01761                             row.Cells["colAlbum" + dgv.Tag.ToString()].Value.ToString(),
01762                             int.Parse(row.Cells["colYear" + dgv.Tag.ToString()].Value.ToString()),
01763                             row.Cells["colLabel" + dgv.Tag.ToString()].Value.ToString(),
01764                             new TimeSpan(),
01765                             row.Cells["colGender" + dgv.Tag.ToString()].Value.ToString());
01766         }
01767         if (!this.Controller.UpdateAudioFile(file))
01768             MessageBox.Show("An error has occurred", "Error");
01769     }
01770
01771     private void btnUpdateAudioDevice_Click(object sender, EventArgs e)
01772     {
01773         this.UpdateAudioDevices();
01774     }
01775
01776     private void btnStartCapture_Click(object sender, EventArgs e)
01777     {
01778         if (lsbTranscoders.SelectedIndex >= 0)
01779         {
01780             this.Controller.TranscoderCapture(true, cmbAudioDevice.SelectedItem.ToString(), (
01781 WebradioTranscoder)lsbTranscoders.SelectedItem, this.IdWebradio);
01782         }
01783         else
01784             MessageBox.Show("Please select a transcoder.", "Error");
01785     }
01786
01787     private void btnStopCapture_Click(object sender, EventArgs e)
01788     {
01789         if (lsbTranscoders.SelectedIndex >= 0)
01790         {
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02350
02351
02352
02353
02354
02355
02356
02357
02358
02359
02360
02361
02362
02363
02364
02365
02366
02367
02368
02369
02370
02371
02372
02373
02374
02375
02376
02377
02378
02379
02380
02381
02382
02383
02384
02385
02386
02387
02388
02389
02390
02391
02392
02393
02394
02395
02396
02397
02398
02399
02400
02401
02402
02403
02404
02405
02406
02407
02408
02409
02410
02411
02412
02413
02414
02415
02416
02417
02418
02419
02420
02421
02422
02423
02424
02425
02426
02427
02428
02429
02430
02431
02432
02433
02434
02435
02436
02437
02438
02439
02440
02441
02442
02443
02444
02445
02446
02447
02448
02449
02450
02451
02452
02453
02454
02455
02456
02457
02458
02459
02460
02461
02462
02463
02464
02465
02466
02467
02468
02469
02470
02471
02472
02473
02474
02475
02476
02477
02478
02479
02480
02481
02482
02483
02484
02485
02486
02487
02488
02489
02490
02491
02492
02493
02494
02495
02496
02497
02498
02499
02500
02501
02502
02503
02504
02505
02506
02507
02508
02509
02510
02511
02512
02513
02514
02515
02516
02517
02518
02519
02520
02521
02522
02523
02524
02525
02526
02527
02528
02529
02530
02531
02532
02533
02534
02535
02536
02537
02538
02539
02540
02541
02542
02543
02544
02545
02546
02547
02548
02549
02550
02551
02552
02553
02554
02555
02556
02557
02558
02559
02560
02561
02562
02563
02564
02565
02566
02567
02568
02569
02570
02571
02572
02573
02574
02575
02576
02577
02578
02579
02580
02581
02582
02583
02584
02585
02586
02587
02588
02589
02590
02591
02592
02593
02594
02595
02596
02597
02598
02599
02600
02601
02602
02603
02604
02605
02606
02607
02608
02609
02610
02611
02612
02613
02614
02615
02616
02617
02618
02619
02620
02621
02622
02623
02624
02625
02626
02627
02628
02629
02630
02631
02632
02633
02634
02635
02636
02637
02638
02639
02640
02641
02642
02643
02644
02645
02646
02647
02648
02649
02650
02651
02652
02653
02654
02655
02656
02657
02658
02659
02660
02661
02662
02663
02664
02665
02666
02667
02668
02669
02670
02671
02672
02673
02674
02675
02676
02677
02678
02679
02680
02681
02682
02683
02684
02685
02686
02687
02688
02689
02690
02691
02692
02693
02694
02695
02696
02697
02698
02699
02700
02701
02702
02703
02704
02705
02706
02707
02708
02709
02710
02711
02712
02713
02714
02715
02716
02717
02718
02719
02720
02721
02722
02723
02724
02725
02726
02727
02728
02729
02730
02731
02732
02733
02734
02735
02736
02737
02738
02739
02740
02741
02742
02743
02744
02745
02746
02747
02748
02749
02750
02751
02752
02753
02754
02755
02756
02757
02758
02759
02760
02761
02762
02763
02764
02765
02766
02767
02768
02769
02770
02771
02772
02773
02774
02775
02776
02777
02778
02779
02780
02781
02782
02783
02784
02785
02786
02787
02788
02789
02790
02791
02792
02793
02794
02795
02796
02797
02798
02799
02800
02801
02802
02803
02804
02805
02806
02807
02808
02809
02810
02811
02812
02813
02814
02815
02816
02817
02818
02819
02820
02821
02822
02823
02824
02825
02826
02827
02828
02829
02830
02831
02832
02833
02834
02835
02836
02837
02838
02839
02840
02841
02842
02843
02844
02845
02846
02847
02848
02849
02850
02851
02852
02853
02854
02855
02856
02857
02858
02859
02860
02861
02862
02863
02864
02865
02866
02867
02868
02869
02870
02871
02872
02873
02874
02875
02876
02877
02878
02879
02880
02881
02882
02883
02884
02885
02886
02887
02888
02889
02890
02891
02892
02893
02894
02895
02896
02897
02898
02899
02900
02901
02902
02903
02904
02905
02906
02907
02908
02909
02910
02911
02912
02913
02914
02915
02916
02917
02918
02919
02920
02921
02922
02923
02924
02925
02926
02927
02928
02929
02930
02931
02932
02933
02934
02935
02936
02937
02938
02939
02940
02941
02942
02943
02944
02945
02946
02947
02948
02949
02950
02951
02952
02953
02954
02955
02956
02957
02958
02959
02960
02961
02962
02963
02964
02965
02966
02967
02968
02969
02970
02971
02972
02973
02974
02975
02976
02977
02978
02979
02980
02981
02982
02983
02984
02985
02986
02987
02988
02989
02990
02991
02992
02993
02994
02995
02996
02997
02998
02999
03000
03001
03002
03003
03004
03005
03006
03007
03008
03009
03010
03011
03012
03013
03014
03015
03016
03017
03018
03019
03020
03021
03022
03023
03024
03025
03026
03027
03028
03029
03030
03031
03032
03033
03034
03035
03036
03037
03038
03039
03040
03041
03042
03043
03044
03045
03046
03047
03048
03049
03050
03051
03052
03053
03054
03055
03056
03057
03058
03059
03060
03061
03062
03063
03064
03065
03066
03067
03068
03069
03070
03071
03072
03073
03074
03075
03076
03077
03078
03079
03080
03081
03082
03083
03084
03085
03086
03087
03088
03089
03090
03091
03092
03093
03094
03095
03096
03097
03098
03099
03100
03101
03102
03103
03104
03105
03106
03107
03108
03109
03110
03111
03112
03113
03114
03115
03116
03117
03118
03119
03120
03121
03122
03123
03124
03125
03126
03127
03128
03129
03130
03131
03132
03133
03134
03135
03136
03137
03138
03139
03140
03141
03142
03143
03144
03145
03146
03147
03148
03149
03150
03151
03152
03153
03154
03155
03156
03157
03158
03159
03160
03161
03162
03163
03164
03165
03166
03167
03168
03169
03170
03171
03172
03173
03174
03175
03176
03177
03178
03179
03180
03181
03182
03183
03184
03185
03186
03187
03188
03189
03190
03191
03192
03193
03194
03195
03196
03197
03198
03199
03200
03201
03202
03203
03204
03205
03206
03207
03208
03209
03210
03211
03212
03213
03214
03215
03216
03217
03218
03219
03220
03221
03222
03223
03224
03225
03226
03227
03228
03229
03230
03231
03232
03233
03234
03235
03236
03237
03238
03239
03240
03241
03242
03243
03244
03245
03246
03247
03248
03249
03250
03251
03252
03253
03254
03255
03256
03257
03258
03259
03260
03261
03262
03263
03264
03265
03266
03267
03268
03269
03270
03271
03272
03273
03274
03275
03276
03277
03278
03279
03280
03281
03282
03283
03284
03285
03286
03287
03288
03289
03290
03291
03292
03293
03294
03295
03296
03297
03298
03299
03300
03301
03302
03303
03304
03305
03306
03307
03308
03309
03310
03311
03312
03313
03314
03315
03316
03317
03318
03319
03320
03321
03322
03323
03324
03325
03326
03327
03328
03329
03330
03331
03332
03333
03334
03335
03336
03337
03338
03339
03340
03341
03342
03343
03344
03345
03346
03347
03348
03349
03350
03351
03352
03353
03354
03355
03356
03357
03358
03359
03360
03361
03362
03363
03364
03365
03366
03367
03368
03369
03370
03371
03372
03373
03374
03375
03376
03377
03378
03379
03380
03381
03382
03383
03384
03385
03386
03387
03388
03389
03390
03391
03392
03393
03394
03395
03396
03397
03398
03399
03400
03401
03402
03403
03404
03405
03406
03407
03408
03409
03410
03411
03412
03413
03414
03415
03416
03417
03418
03419
03420
03421
03422
03423
03424
03425
03426
03427
03428
03429
03430
03431
03432
03433
03434
03435
03436
03437
03438

```

```

01845             this.Controller.TranscoderCapture(false, cmbAudioDevice.SelectedItem.ToString(), (
01846                 WebradioTranscoder)lsbTranscoders.SelectedItem, this.IdWebradio);
01847             }
01848         else
01849             MessageBox.Show("Please select a transcoder.", "Error");
01850     }
01851 
01852     private void btnUpdateListeners_Click(object sender, EventArgs e)
01853     {
01854         List<WebradioListener> listeners = this.Controller.GetServerListeners(this.IdWebradio);
01855         dgvServerListeners.Rows.Clear();
01856         foreach (WebradioListener listener in listeners)
01857         {
01858             string[] infos = new string[] { listener.Hostname, listener.Useragent,
01859             listener.ConnectionTime.ToString(), listener.Uid.ToString() };
01860             dgvServerListeners.Rows.Add(infos);
01861         }
01862         this.Controller.UpdateServerStats(this.IdWebradio);
01863     }
01864 
01865     private void checkLibraryToolStripMenuItem_Click(object sender, EventArgs e)
01866     {
01867         if (this.Controller.CheckLibrary())
01868             MessageBox.Show("Library has been checked and invalids files has been deleted.", "Success");
01869         else
01870             MessageBox.Show("An error occured", "Error");
01871     }
01872 
01873 #endregion
01874
01875 }
01876 }
01877 }
01878 }
```

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
00002 using System;
00003
00004 namespace WebradioManager
00005 {
00006     public abstract class AudioFile
00007     {
00008         #region Const
00009         // \brief Number of elements (for an audiofile).
00010         const int NUMBER_OF_ELEMENTS = 9;
00011         // \brief The default identifier.
00012         const int DEFAULT_ID = 0;
00013     #endregion
00014 }
```

```
00028      #region Fields
00029      // \brief The identifier.
00030      private int _id;
00031      // \brief Filename of the audiofile.
00032      private string _filename;
00033      // \brief The title.
00034      private string _title;
00035      // \brief The artist.
00036      private string _artist;
00037      // \brief The album.
00038      private string _album;
00039      // \brief The year.
00040      private int _year;
00041      // \brief The label.
00042      private string _label;
00043      // \brief The duration.
00044      private TimeSpan _duration;
00045      // \brief The gender.
00046      private string _gender;
00047      // \brief The type.
00048      private AudioType _type;
00049      #endregion
00050
00051      #region Properties
00052
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
00198      public AudioFile(int id, string filename, string title, string artist, string album, int
00199          year, string label, TimeSpan duration, string gender, AudioType audiotype)
00200      {
00201          this.Id = id;
00202          this.Filename = filename;
00203          this.Title = title;
00204          this.Artist = artist;
00205          this.Album = album;
00206          this.Year = year;
00207          this.Label = label;
00208          this.Duration = duration;
00209          this.Gender = gender;
00210          this.Type = audiotype;
00211      }
00212
00213      public AudioFile(string filename, string title, string artist, string album, int year,
00214          string label, TimeSpan duration, string gender, AudioType audiotype)
00215          :this(DEFAULT_ID,filename, title, artist, album, year, label, duration, gender, audiotype)
00216      {
00217          //NO CODE
00218      }
00219
00220      public string[] GetInfosArray()
00221      {
00222          string[] infos = new string[NUMBER_OF_ELEMENTS];
00223
00224          infos[0] = this.Id.ToString();
00225          infos[1] = this.Title;
00226          infos[2] = this.Artist;
00227          infos[3] = this.Album;
00228          infos[4] = this.Year.ToString();
00229          infos[5] = this.Label;
00230          infos[6] = this.Duration.ToString(@"\:mm\:ss");
00231          infos[7] = this.Gender;
00232          infos[8] = this.Filename;
00233
00234          return infos;
00235      }
00236  #endregion
00237 }
00238 }
```

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
00002  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         }
00079         public Dictionary<int,Webradio> LoadWebradios()
```

```

00080      {
00081          Dictionary<int, Webradio> webradios = new Dictionary<int, Webradio>();
00082          SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT * FROM twebradio");
00083          while(reader.Read())
00084          {
00085              webradios.Add( int.Parse(reader["id"].ToString()), new Webradio(reader["name"]).
00086              ToString(), int.Parse(reader["id"].ToString()));
00087          }
00088          reader.Close();
00089
00090          foreach(KeyValuePair<int,Webradio> pair in webradios)
00091          {
00092              Webradio wr = pair.Value;
00093              //Server
00094              int id = wr.Id;
00095              reader = this.Controls.ExecuteReader("SELECT * FROM tserver WHERE webradiooid = " +
00096                  id.ToString());
00097              reader.Read();
00098              wr.Server = new WebradioServer(int.Parse(reader["port"].ToString()),
00099                  reader["logfilename"].ToString(),
00100                  reader["configfilename"].ToString(),
00101                  reader["password"].ToString(),
00102                  reader["adminpassword"].ToString(),
00103                  int.Parse(reader["maxlistener"].ToString()));
00104              reader.Close();
00105              //----
00106
00107              //Calendar
00108              reader = this.Controls.ExecuteReader("SELECT id, filename FROM tcalendar WHERE
00109                  webradiooid = " + id.ToString());
00110              reader.Read();
00111              wr.Calendar = new WebradioCalendar(int.Parse(reader["id"].ToString()),
00112                  reader["filename"].ToString());
00113              reader.Close();
00114              reader = this.Controls.ExecuteReader("SELECT ce.id AS EventId, ce.playlistid,
00115                  cestarttime, ce.duration, ce.name AS EventName, ce.repeat, ce.priority, ce.shuffle, ce.loopatend, p.name AS
00116                  PlaylistName, p.filename AS PlaylistFilename, at.name AS AudiotypeName FROM tcalendarevent ce, tplaylist p,
00117                  audiotype at WHERE ce.calendarid = " + wr.Calendar.Id + " AND ce.playlistid = p.id AND p.typeid = at.id");
00118              while(reader.Read())
00119              {
00120                  string[] time = reader["starttime"].ToString().Split(':');
00121                  TimeSpan start = new TimeSpan(int.Parse(time[0]),int.Parse(time[1]),int.Parse(time[2]));
00122
00123                  time = reader["duration"].ToString().Split(':');
00124                  TimeSpan duration = new TimeSpan(int.Parse(time[0]),int.Parse(time[1]),int.Parse(time[2]));
00125
00126                  Playlist playlist;
00127                  if (reader["AudiotypeName"].ToString() == AudioType.Music.ToString())
00128                      playlist = new PlaylistMusic(reader["PlaylistName"].ToString(), reader[
00129                          "PlaylistFilename"].ToString());
00130                  else
00131                      playlist = new PlaylistAd(reader["PlaylistName"].ToString(), reader[""
00132                          PlaylistFilename"].ToString());
00133
00134                  wr.Calendar.Events.Add(new CalendarEvent(int.Parse(reader["EventId"]).
00135                      ToString()), reader["EventName"].ToString(),
00136                      start,
00137                      duration,
00138                      int.Parse(reader["repeat"].ToString()),
00139                      int.Parse(reader["priority"].ToString()),
00140                      Convert.ToBoolean(reader["shuffle"].ToString()),
00141                      Convert.ToBoolean(reader["loopatend"].ToString()),
00142                      playlist));
00143
00144              }
00145              reader.Close();
00146              foreach(Playlist playlist in wr.Playlists)
00147              {
00148                  reader = this.Controls.ExecuteReader("SELECT m.filename FROM tmusic m,
00149                      tplaylist_has_music pm WHERE pm.playlistid = " + playlist.Id + " AND m.id = pm.musicid");
00150                  while(reader.Read())
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759

```

```

00150             {
00151                 playlist.AudioFileList.Add(reader["filename"].ToString());
00152             }
00153         }
00154         reader.Close();
00155         //Transcoders
00156         reader = this.Controls.ExecuteReader("SELECT c.filename AS CalendarFilename, tr.id,
00157         tr.name AS TransName, tr.bitrate, tr.samplerate, tr.url, tr.ip, tr.port, tr.adminport, tr.password,
00158         tr.configfilename, tr.logfilename, st.name AS StreamName FROM tcalendar c, ttranscoder tr, tstreamtype st WHERE
00159         tr.webradiooid = " + id.ToString() + " AND tr.streamtypeid = st.id AND c.webradiooid = " + id.ToString());
00160         while(reader.Read())
00161         {
00162             WebradioTranscoder trans = null;
00163             if(reader["StreamName"].ToString() == StreamType.MP3.ToString())
00164                 trans = new TranscoderMp3(int.Parse(reader["id"].ToString()),
00165                     reader["TransName"].ToString(),
00166                     int.Parse(reader["bitrate"].ToString()),
00167                     int.Parse(reader["samplerate"].ToString()),
00168                     IPAddress.Parse(reader["ip"].ToString()),
00169                     int.Parse(reader["port"].ToString()),
00170                     int.Parse(reader["adminport"].ToString()),
00171                     reader["url"].ToString(),
00172                     reader["password"].ToString(),
00173                     reader["configfilename"].ToString(),
00174                     reader["logfilename"].ToString());
00175             else
00176                 trans = new TranscoderAacPlus(int.Parse(reader["id"].ToString()),
00177                     reader["TransName"].ToString(),
00178                     int.Parse(reader["bitrate"].ToString()),
00179                     int.Parse(reader["samplerate"].ToString()),
00180                     IPAddress.Parse(reader["ip"].ToString()),
00181                     int.Parse(reader["port"].ToString()),
00182                     int.Parse(reader["adminport"].ToString()),
00183                     reader["url"].ToString(),
00184                     reader["password"].ToString(),
00185                     reader["configfilename"].ToString(),
00186                     reader["logfilename"].ToString());
00187             trans.CalendarFile = reader["CalendarFilename"].ToString();
00188             wr.Transcoders.Add(trans);
00189         }
00190     }
00191     return webradios;
00192 }
00193
00194 public List<AudioFile> LoadLibrary()
00195 {
00196     List<AudioFile> audios = new List<AudioFile>();
00197     //Music
00198     SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT m.id, m.filename, m.title,
00199     m.artist, m.album, m.year, m.label, m.duration, t.name AS AudioType, g.name AS GenderName FROM tmusic m,
00200     taudiotype t, tgender g WHERE m.typeid = t.id AND m.genderid = g.id");
00201     while (reader.Read())
00202     {
00203         string[] time = reader["duration"].ToString().Split(':');
00204         TimeSpan duration = new TimeSpan(int.Parse(time[0]), int.Parse(time[1]), int.Parse(time[2]));
00205         ;
00206         AudioFile af = null;
00207         if (reader["AudioType"].ToString() == AudioType.Ad.ToString())
00208             af = new Ad(int.Parse(reader["id"].ToString()),
00209                 reader["filename"].ToString(),
00210                 reader["title"].ToString(),
00211                 reader["artist"].ToString(),
00212                 reader["album"].ToString(),
00213                 int.Parse(reader["year"].ToString()),
00214                 reader["label"].ToString(),
00215                 duration,
00216                 reader["GenderName"].ToString());
00217         else if (reader["AudioType"].ToString() == AudioType.Music.ToString())
00218             af = new Music(int.Parse(reader["id"].ToString()),
00219                 reader["filename"].ToString(),
00220                 reader["title"].ToString(),
00221                 reader["artist"].ToString(),
00222                 reader["album"].ToString(),
00223                 int.Parse(reader["year"].ToString()),
00224                 reader["label"].ToString(),
00225                 duration,
00226                 reader["GenderName"].ToString());
00227         audios.Add(af);
00228     }
00229     reader.Close();
00230
00231     return audios;
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241

```

```

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.ExecuteReader("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("webradiooid", 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("webradiooid", id.ToString());
00287             this.Controls.Insert("tcalendar", data);
00288             data.Clear();
00289             reader = this.Controls.ExecuteReader("SELECT id FROM tcalendar WHERE webradiooid = " +
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.ExecuteReader("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.ExecuteReader("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.ExecuteReader("SELECT id FROM tgender WHERE name =
00398             '" + gender + "');");
00399         reader.Read();
00400         int id;
00401         if (reader.HasRows)
00402             id = int.Parse(reader["id"].ToString());
00403         else
00404             id = ERROR;
00405         reader.Close();
00406         return id;
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.ExecuteReader("SELECT id FROM tgender WHERE name =
00428             '" + gender + "');");
00429         reader.Read();
00430         int id = int.Parse(reader["id"].ToString());
00431         reader.Close();
00432         return id;
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.ExecuteReader("SELECT id FROM tmusic WHERE filename
00472             = '" + file.Filename.Replace('\\', ' ') + "');");
00473         reader.Read();
00474         int id = int.Parse(reader["id"].ToString());
00475         reader.Close();
00476         return id;
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
00517     public bool AudioFileExist(string filename)
00518     {
00519         SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT COUNT(*) AS Count FROM tmusic
00520 WHERE filename = '" + filename.Replace('\\', ' ') + "'");
00521         reader.Read();
00522         if (reader["Count"].ToString() == "0")
00523         {
00524             reader.Close();
00525             return false;
00526         }
00527         else
00528         {
00529             reader.Close();
00530             return true;
00531         }
00532     }
00533
00534     public bool DeleteAudioFile(int id)
00535     {
00536         try
00537         {
00538             this.Controls.Delete("tmusic", "id = " + id.ToString());
00539             //this.Controls.Delete("tplaylist_has_music", "idmusic = " + id.ToString());
00540             return true;
00541         }
00542         catch
00543         {
00544             return false;
00545         }
00546     }
00547
00548     public int CreatePlaylist(Playlist playlist, int webradioId)
00549     {
00550         if(this.PlaylistExist(playlist,webradioId))
00551             return ERROR;
00552
00553         Dictionary<string,string> data = new Dictionary<string,string>();
00554         data.Add("name", playlist.Name);
00555         data.Add("filename", playlist.Filename);
00556         data.Add("webradioid", webradioId.ToString());
00557         data.Add("typeid", ((int)playlist.Type).ToString());
00558         this.Controls.Insert("tplaylist", data);
00559
00560         SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT id FROM tplaylist WHERE name
00561         = '" + playlist.Name + "' AND webradioid = " + webradioId.ToString() + " AND typeid = " + ((int)
00562         playlist.Type).ToString());
00563         reader.Read();
00564         int id = int.Parse(reader["id"].ToString());
00565         reader.Close();
00566         return id;
00567     }
00568
00569     public bool DeletePlaylist(int id)
00570     {
00571         try
00572         {
00573             this.Controls.Delete("tplaylist", "id = " + id.ToString());
00574             return true;
00575         }
00576         catch
00577         {
00578             return false;
00579         }
00580     }
00581
00582     public bool AddToPlaylist(int idAudioFile, int idPlaylist)
00583     {
00584         try
00585         {
00586             Dictionary<string,string> data = new Dictionary<string,string>();
00587             data.Add("playlistid", idPlaylist.ToString());
00588             data.Add("musicid", idAudioFile.ToString());
00589             this.Controls.Insert("tplaylist_has_music", data);
00590             return true;
00591         }
00592         catch
00593         {
00594             return false;
00595         }
00596     }
00597 
```

```

00677     public bool RemoveFromPlaylist(int idAudioFile, int idPlaylist)
00678     {
00679         try
00680         {
00681             this.Controls.Delete("tplaylist_has_music", "musicid = " + idAudioFile.ToString() + " AND
00682             playlistid = " + idPlaylist.ToString());
00683         }
00684         catch
00685         {
00686             return false;
00687         }
00688     }
00689
00704     private bool PlaylistExist(Playlist playlist, int webradioId)
00705     {
00706         SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT COUNT(*) AS Count FROM
00707             tplaylist WHERE name = '" + playlist.Name + "' AND webradiooid = " + webradioId.ToString() + " AND typeid = " + ((int)
00708             playlist.Type).ToString());
00709         reader.Read();
00710         bool result = Convert.ToBoolean(int.Parse(reader["Count"].ToString()));
00711         reader.Close();
00712         return result;
00713     }
00728     public int AddGeneratedPlaylist(Playlist playlist, List<int>
00729         audioFilesId, int webradioId)
00730     {
00731         int idPlaylist = this.CreatePlaylist(playlist, webradioId);
00732         if (idPlaylist == ERROR)
00733             return idPlaylist;
00734         foreach(int audioFileDialog in audioFilesId)
00735             this.AddToPlaylist(audioFileDialog, idPlaylist);
00736         return idPlaylist;
00737     }
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.ExecuteReader("SELECT id FROM tcalendarevent WHERE
00770             name = '" + newEvent.Name + "' AND calendarid = " + calendarId.ToString());
00771         reader.Read();
00772         int id = int.Parse(reader["id"].ToString());
00773         reader.Close();
00774         return id;
00775     }
00790     public bool EventExist(CalendarEvent aEvent, int calendarId)
00791     {
00792         SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT COUNT(*) AS Count FROM
00793             tcalendarevent WHERE name = '" + aEvent.Name + "' AND calendarid = " + calendarId.ToString());
00794         reader.Read();
00795         bool result = (reader["Count"].ToString() == "0")?false:true;
00796         reader.Close();
00797         return result;
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
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
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
00856
00857     public bool TranscoderExist(string name, int webradioId)
00858     {
00859         SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT COUNT(*) AS Count FROM
00860         ttranscoder WHERE webradioId = " + webradioId.ToString() + " AND name = '" + name + "'");
00861         reader.Read();
00862         if (reader["Count"].ToString() == "0")
00863         {
00864             reader.Close();
00865             return false;
00866         }
00867         else
00868         {
00869             reader.Close();
00870             return true;
00871         }
00872     }
00873
00874     public int AddTranscoder(WebradioTranscoder transcoder, int
00875         webradioId)
00876     {
00877         if (this.TranscoderExist(transcoder.Name, webradioId))
00878             return ERROR;
00879         try
00880         {
00881             Dictionary<string, string> data = new Dictionary<string, string>();
00882             data.Add("webradioId", webradioId.ToString());
00883             data.Add("streamtypeid", ((int)transcoder.StreamType).ToString());
00884             data.Add("bitrate", transcoder.Birate.ToString());
00885             data.Add("sampleRate", transcoder.SampleRate.ToString());
00886             data.Add("name", transcoder.Name);
00887             data.Add("url", transcoder.Url);
00888             data.Add("port", transcoder.Port.ToString());
00889             data.Add("adminport", transcoder.AdminPort.ToString());
00890             data.Add("ip", transcoder.Ip.ToString());
00891             data.Add("password", transcoder.Password);
00892             data.Add("configfilename", transcoder.ConfigFilename);
00893             data.Add("logfilename", transcoder.LogFilename);
00894             this.Controls.Insert("ttranscoder", data);
00895
00896             SQLiteDataReader reader = this.Controls.ExecuteReader("SELECT id FROM ttranscoder WHERE
00897             webradioId = " + webradioId.ToString() + " AND name = '" + transcoder.Name + "'");
00898             reader.Read();
00899             int id = int.Parse(reader["id"].ToString());
00900             reader.Close();
00901             data.Clear();
00902             data.Add("configfilename", transcoder.ConfigFilename + id.ToString() + ".config");
00903             data.Add("logfilename", transcoder.LogFilename + "/" + id.ToString() + ".log");
00904             this.Controls.Update("ttranscoder", data, "webradioId = " + webradioId.ToString() + " AND
00905             name = '" + transcoder.Name + "'");
00906             return id;
00907         }
00908         catch
00909         {
00910             return ERROR;
00911         }
00912     }
00913
00914     public bool DeleteTranscoder(int transcoderId)
00915     {
00916         try
00917         {
00918             this.Controls.Delete("ttranscoder", "id = " + transcoderId.ToString());
00919             return true;
00920         }
00921         catch
00922         {
00923             return false;
00924         }
00925     }
00926
00927     public bool UpdateTranscoder(WebradioTranscoder transcoder)
00928     {

```

```
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("amplerate", 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
01078     public Dictionary<string, string> GetHistory(int transcoderId)
01079     {
01080         Dictionary<string, string> filenames = new Dictionary<string, string>();
01081         SQLiteDataReader reader = this.Controls.ExecuteReader("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,
01163     Webradio webradio)
01164     {
01165         try
01166         {
01167             Dictionary<string, string> data = new Dictionary<string, string>();
01168             foreach (WebradioTranscoder transcoder in webradio.
01169             Transcoders)
01170             {
01171                 data.Clear();
01172                 data.Add("configfilename", transcoder.ConfigFilename.Replace(oldName, newName));
01173                 data.Add("logfilename", transcoder.LogFilename.Replace(oldName, newName));
01174                 this.Controls.Update("ttranscoder", data, "webradiooid = " + webradio.Id);
01175             }
01176             data.Clear();
01177             data.Add("configfilename", webradio.Server.ConfigFilename.Replace(oldName, newName));
01178             data.Add("logfilename", webradio.Server.LogFilename.Replace(oldName, newName));
01179             this.Controls.Update("tserver", data, "webradiooid = " + webradio.Id);
01180             foreach (Playlist playlist in webradio.Playlists)
01181             {
01182                 data.Clear();
01183                 data.Add("filename", playlist.Filename.Replace(oldName, newName));
01184                 this.Controls.Update("tpplaylist", data, "webradiooid = " + webradio.Id);
01185             }
01186             data.Clear();
01187             data.Add("filename", webradio.Calendar.Filename.Replace(oldName, newName));
01188             this.Controls.Update("tcalendar", data, "webradiooid = " + webradio.Id);
01189             return true;
01190         }
01191         catch
01192         {
01193             return false;
01194         }
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
00115     public string Name
00116     {
00117         get { return _name; }
00118         set { _name = value; }
00119     }
00120
00121     public TimeSpan StartTime
00122     {
00123         get { return _startTime; }
00124         set { _startTime = value; }
00125     }
00126
00127     public TimeSpan Duration
00128     {
00129         get { return _duration; }
00130         set { _duration = value; }
00131     }
00132
00133     public int Repeat
00134     {
00135         get { return _repeat; }
00136         set { _repeat = value; }
00137     }
00138
00139     public bool Shuffle
00140     {
00141         get { return _shuffle; }
00142         set { _shuffle = value; }
00143     }
00144
00145     public bool Loopatend
00146     {
00147         get { return _loopatend; }
00148         set { _loopatend = value; }
00149     }
00150
00151     public int Priority
00152     {
00153         get { return _priority; }
00154         set { _priority = value; }
00155     }
00156
00157 #endregion
00158
00159 #region Methods
00160
00161     public CalendarEvent(string name, TimeSpan starttime, TimeSpan duration, int repeat,
00162     int priority, bool shuffle, bool loopatend, Playlist playlist)
00163     :this(0,name,starttime,duration,repeat,priority,shuffle,loopatend,playlist)
00164     {
00165         //NO CODE
00166     }
00167
00168     public CalendarEvent(int id, string name, TimeSpan starttime, TimeSpan duration, int
00169     repeat, int priority, bool shuffle, bool loopatend, Playlist playlist)
00170     {
00171         this.Id = id;
00172         this.Name = name;
00173         this.StartTime = starttime;
00174         this.Duration = duration;
00175         this.Repeat = repeat;
00176         this.Priority = priority;
00177         this.Shuffle = shuffle;
00178         this.Loopatend = loopatend;
00179         this.Playlist = playlist;
00180     }
00181
00182     public DayWeek GetSelectedDays()
00183     {
00184         DayWeek dow = new DayWeek();
00185         dow.Monday = Convert.ToBoolean(this.Repeat & MONDAY_MASK);
00186         dow.Tuesday = Convert.ToBoolean(this.Repeat & TUESDAY_MASK);
00187         dow.Wednesday = Convert.ToBoolean(this.Repeat & WEDNESDAY_MASK);
00188         dow.Thursday = Convert.ToBoolean(this.Repeat & THURSDAY_MASK);
00189         dow.Friday = Convert.ToBoolean(this.Repeat & FRIDAY_MASK);
00190         dow.Saturday = Convert.ToBoolean(this.Repeat & SATURDAY_MASK);
00191         dow.Sunday = Convert.ToBoolean(this.Repeat & SUNDAY_MASK);
00192         return dow;
00193     }
00194 }
00195 #endregion
00196 }
00197 }
```

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
0010 {
0020     public class EventAppointment : Appointment
0021     {
```

```

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     #endregion
00074 }
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
0010 {
0020     public class Music : AudioFile
0021     {
0022         #region Methods
0023
0042         public Music(int id, string filename, string title, string artist, string album, int year,
0043             string label, TimeSpan duration, string gender) :
0044             base(id,filename,title,artist,album,year,label,duration,gender,
0045                 AudioType.Music)
0046         {
0047
0066         public Music(string filename, string title, string artist, string album, int year, string
0067             label, TimeSpan duration, string gender) :
0068             base(filename, title, artist, album, year, label, duration, gender,
0069                 AudioType.Music)
0070         {
0071             #endregion
0072         }
0073 }
```

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
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,
00035             AudioType.Ad)
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
00174     public bool DuplicateWebradio(int id)
00175     {
00176         return this.Model.DuplicateWebradio(id);
00177     }
00178
00179     public void OpenWebradio(int id)
00180     {
00181
00182         AdminController admincontroller = new AdminController(id, this.
00183             Model);
00184         this.Model.AddObserver(admincontroller);
00185     }
00186
00187     public bool StopAllProcess()
00188     {
00189         return this.Model.StopAllProcess();
00190     }
00191
00192     #endregion
00193
00194 }
00195 }
```

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
00002 using System;
00003 using System.Collections.Generic;
00004 using System.Windows.Forms;
00005
00006 namespace WebradioManager
00007 {
00008     public partial class SelectionView : Form
00009     {
00010         #region Const
00011         // \brief The maximum name length.
00012         const int MAX_NAME_LENGTH = 255;
00013         #endregion
00014
00015         #region Fields
00016         // \brief The controller.
00017         private SelectionController _controller;
00018         #endregion
00019
00020         #region Properties
00021
00022         public SelectionController Controller
00023         {
00024             get { return _controller; }
00025         }
00026
00027     }
00028 }
```

```

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 occurred. (Invalid name or cannot create folders and files.)",
00113 "Error");
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 occurred.", "Error");
00139         this.UpdateView();
00140     }
00141     else
00142         MessageBox.Show("Please select a webradio to delete.", "No webradio selected",
00143 MessageBoxButtons.OK, MessageBoxIcon.Error);
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 occurred", "Error");
00165     }
00166     else
00167         MessageBox.Show("Please select a webradio to duplicate.", "No webradio selected",
00168 MessageBoxButtons.OK, MessageBoxIcon.Error);
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
00202 private void SelectionView_FormClosing(object sender, FormClosingEventArgs e)
00203 {
00204     if (MessageBox.Show("All transcoders and all servers will be shutting down. Are you sure ?", "Close",
00205 MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
00206     {
00207         if (!this.Controller.StopAllProcess())
00208         {
00209             MessageBox.Show("An error has occurred", "Error");
00210             e.Cancel = true;
00211         }
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,
00045             IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename)
00046             :base(id, name, bitrate, sampleRate, ip, port, adminport, url, password, configFilename, logFilename,
00047                 StreamType.AACPlus)
00048         {
00049
00070         public TranscoderAacPlus(string name, int bitrate, int sampleRate, IPAddress ip,
00071             int port, int adminport, string url, string password, string configFilename, string logFilename)
00072             : base(name, bitrate, sampleRate, ip, port, adminport, url, password, configFilename,
00073                 logFilename, StreamType.AACPlus)
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,
00045             int port, int adminport, string url, string password, string configFilename, string logFilename)
00046             :base(id, name, bitrate, sampleRate, ip, port, adminport, url, password, configFilename, logFilename,
00047                 StreamType.MP3)
00048         {
00049
00070         public TranscoderMp3(string name, int bitrate, int sampleRate, IPAddress ip, int port,
00071             int adminport, string url, string password, string configFilename, string logFilename)
00072             :base(name, bitrate, sampleRate, ip, port, adminport, url, password, configFilename,
00073                 logFilename, StreamType.MP3)
00074         {
00075         }
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
0010 {
0020     public class Webradio
0021     {
0022         #region Const
0023         // \brief The default identifier.
0024         const int DEFAULT_ID = 0;
0025         #endregion
0026
0027         #region Fields
0028         // \brief The playlists list.
0029         private List<Playlist> _playlists;
0030         // \brief The calendar.
0031         private WebradioCalendar _calendar;
0032         // \brief The name.
0033         private string _name;
0034         // \brief The server.
0035         private WebradioServer _server;
0036         // \brief The identifier.
0037         private int _id;
0038         // \brief The transcoders list.
```

```

00039     private List<WebradioTranscoder> _transcoders;
00040     #endregion
00041
00042     #region Properties
00043
00044     public List<WebradioTranscoder> Transcoders
00045     {
00046         get { return _transcoders; }
00047         set { _transcoders = value; }
00048     }
00049
00050     public WebradioServer Server
00051     {
00052         get { return _server; }
00053         set { _server = value; }
00054     }
00055
00056     public string Name
00057     {
00058         get { return _name; }
00059         set { _name = value; }
00060     }
00061
00062     public WebradioCalendar Calendar
00063     {
00064         get { return _calendar; }
00065         set { _calendar = value; }
00066     }
00067
00068     public List<Playlist> Playlists
00069     {
00070         get { return _playlists; }
00071         set { _playlists = value; }
00072     }
00073
00074     public int Id
00075     {
00076         get { return _id; }
00077         set { _id = value; }
00078     }
00079     #endregion
00080
00081     #region Methods
00082
00083     public Webradio(string name, int id)
00084     {
00085         this.Name = name;
00086         this.Id = id;
00087         this.Playlists = new List<Playlist>();
00088         this.Transcoders = new List<WebradioTranscoder>();
00089     }
00090
00091     public Webradio(string name):this(name,DEFAULT_ID)
00092     {
00093         //NO CODE
00094     }
00095
00096     public void GenerateConfigFiles()
00097     {
00098         foreach (Playlist playlist in this.Playlists)
00099         {
00100             playlist.GenerateConfigFile();
00101         }
00102         this.Calendar.GenerateConfigFile();
00103         this.Server.GenerateConfigFile();
00104         foreach (WebradioTranscoder transcoder in this.
00105             Transcoders)
00106         {
00107             transcoder.GenerateConfigFile(this.Playlists);
00108         }
00109     }
00110
00111     public override string ToString()
00112     {
00113         return this.Name + " | ID = " + this.Id.ToString();
00114     }
00115     #endregion
00116 }
00117 }
```

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 {
0019     public class WebradioListener
0020     {
0021         #region Fields
0022         // \brief The hostname.
0023         private string _hostname;
0024         // \brief The useragent.
0025         private string _useragent;
0026         // \brief The connection time.
0027         private uint _connectionTime;
0028         // \brief The UID.
0029         private int _uid;
0030     #endregion
0031
0032     #region Properties
0033
0042         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         }
00093         public string WebAdminUrl
00094         {
00095             get
00096             {
00097                 return "http://" + this.GetLocalIPAddress() + ":" + this.Port + "/admin.cgi";
00098             }
00099         }
00109         public int MaxListener
00110         {
00111             get { return _maxListener; }
00112             set { _maxListener = value; }
00113         }
00123         public Process Process
00124         {
00125             get { return _process; }
00126             set { _process = value; }
00127         }
00137         public string LogFilename
00138         {
00139             get { return _logFilename; }
00140             set { _logFilename = value; }
00141         }
00151         public string ConfigFilename
00152         {
00153             get { return _configFilename; }
00154             set { _configFilename = value; }
00155         }
00165         public string Password
00166         {
00167             get { return _password; }
00168             set { _password = value; }
00169         }
00179         public string AdminPassword
00180         {
00181             get { return _adminPassword; }

```

```

00182         set { _adminPassword = value; }
00183     }
00184
00185     public int Port
00186     {
00187         get { return _port; }
00188         set { _port = value; }
00189     }
00190
00191     #endregion
00192
00193     #region Methods
00194
00195     public WebradioServer(int port, string logFilename, string configFilename, string
00196     password, string adminPassword, int maxListener)
00197     {
00198         this.Port = port;
00199         this.LogFilename = logFilename;
00200         this.ConfigFilename = configFilename;
00201         this.Password = password;
00202         this.AdminPassword = adminPassword;
00203         this.MaxListener = maxListener;
00204         this.Stats = new WebradioServerStats();
00205     }
00206
00207     public void GenerateConfigFile()
00208     {
00209         if (File.Exists(this.ConfigFilename))
00210             File.Delete(this.ConfigFilename);
00211         string output = "";
00212         output += "logfile=" + Directory.GetCurrentDirectory() + "\\\" + this.
00213 LogFilename.Replace('/', '\\') + "\\n";
00214         output += "portbase=" + this.Port.ToString() + "\\n";
00215         output += "password=" + this.Password + "\\n";
00216         output += "adminpassword=" + this.AdminPassword + "\\n";
00217         output += "publicserver=always" + "\\n";
00218         output += "maxuser=" + this.MaxListener.ToString() + "\\n";
00219         //Don't kick idle source
00220         output += "autodumpsourcetime=0\\n";
00221         File.WriteAllText(this.ConfigFilename, output);
00222     }
00223
00224     public bool IsRunning()
00225     {
00226         bool result = false;
00227         try
00228         {
00229             foreach (Process prc in Process.GetProcesses())
00230             {
00231                 if (prc.Id == this.Process.Id)
00232                 {
00233                     result = true;
00234                 }
00235             }
00236             return result;
00237         }
00238         catch
00239         {
00240             return false;
00241         }
00242     }
00243
00244     public bool Start(bool debug)
00245     {
00246         ProcessStartInfo StartInfo = new ProcessStartInfo(Directory.GetCurrentDirectory() +
00247 SC_SERVER_FILENAME)
00248         {
00249             CreateNoWindow = true,
00250             WindowStyle = (debug)?ProcessWindowStyle.Minimized:ProcessWindowStyle.Hidden,
00251             Arguments = "\"" + Directory.GetCurrentDirectory() + "\\\" + this.ConfigFilename.Replace("//"
00252 , '\\') + "\\\"";
00253         };
00254         if (this.IsRunning())
00255             this.Process.Kill();
00256         try
00257         {
00258             this.Process = Process.Start(StartInfo);
00259             return true;
00260         }
00261         catch
00262         {
00263             return false;
00264         }
00265     }
00266
00267     public bool Stop()
00268     {

```

```

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
0010 {
0020     public class WebradioServerStats
0021     {
0022         #region Fields
0023         // \brief The current listeners count.
0024         private int _currentListeners;
0025         // \brief The unique listeners count.
0026         private int _uniqueListeners;
0027         // \brief The peak listeners count.
0028         private int _peakListeners;
0029         // \brief The average time listening.
0030         private TimeSpan _averageTime;
0031         #endregion
0032
0033         #region Properties
0034
0043         public TimeSpan AverageTime
0044         {
0045             get { return _averageTime; }
0046             set { _averageTime = value; }
0047         }
0048
0057         public int PeakListeners
0058         {
0059             get { return _peakListeners; }
0060             set { _peakListeners = value; }
0061         }
0062
0071         public int UniqueListeners
0072         {
0073             get { return _uniqueListeners; }
0074             set { _uniqueListeners = value; }
0075         }
0076
0085         public int CurrentListeners
0086         {
0087             get { return _currentListeners; }
0088             set { _currentListeners = value; }
0089         }
0090     #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 available bitrates.
00079     private static int[] _availableBitrates = { 64000, 96000, 128000, 256000, 320000 };
00080     // \brief The available sample rates.
00081     private static int[] _availableSampleRates = { 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[] AvailableSampleRates
00166     {
00167         get { return WebradioTranscoder._availableSampleRates; }
00168         set { WebradioTranscoder._availableSampleRates = value; }
00169     }
00170
00179     public static int[] AvailableBitrates
00180     {
00181         get { return WebradioTranscoder._availableBitrates; }
00182         set { WebradioTranscoder._availableBitrates = 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
00213     public int SampleRate
00214     {
00215         get { return _sampleRate; }
00216         set { _sampleRate = value; }
00217     }
00218
00219     public string Name
00220     {
00221         get { return _name; }
00222         set { _name = value; }
00223     }
00224
00225     public string Url
00226     {
00227         get { return _url; }
00228         set { _url = value; }
00229     }
00230
00231     public IPAddress Ip
00232     {
00233         get { return _ip; }
00234         set { _ip = value; }
00235     }
00236
00237     public int Port
00238     {
00239         get { return _port; }
00240         set { _port = value; }
00241     }
00242
00243     public string Password
00244     {
00245         get { return _password; }
00246         set { _password = value; }
00247     }
00248
00249     public string ConfigFilename
00250     {
00251         get { return _configFilename; }
00252         set { _configFilename = value; }
00253     }
00254
00255     public string LogFilename
00256     {
00257         get { return _logFilename; }
00258         set { _logFilename = value; }
00259     }
00260
00261     public StreamType StreamType
00262     {
00263         get { return _streamType; }
00264         set { _streamType = value; }
00265     }
00266
00267 #endregion
00268
00269 #region Methods
00270
00271     public WebradioTranscoder(string name, int bitrate, int sampleRate, IPAddress ip,
00272         int port, int adminport, string url, string password, string configFilename, string logFilename,
00273         StreamType st)
00274         : this(DEFAULT_ID, name, bitrate, sampleRate, ip, port, adminport, url, password,
00275             configFilename, logFilename, st)
00276     {
00277         //NO CODE
00278     }
00279
00280     public WebradioTranscoder(int id, string name, int bitrate, int sampleRate,
00281         IPAddress ip, int port, int adminport, string url, string password, string configFilename, string logFilename,
00282         StreamType st)
00283     {
00284         this.Id = id;
00285         this.Name = name;
00286         this.Birate = bitrate;
00287         this.SampleRate = sampleRate;
00288         this.Ip = ip;
00289         this.Port = port;
00290         this.AdminPort = adminport;
00291         this.Url = url;
00292         this.Password = password;
00293         this.LogFilename = logFilename;
00294         this.ConfigFilename = configFilename;
00295     }

```

```

00404         this.StreamType = st;
00405         this.Capture = false;
00406         this.Process = new Process();
00407         this.CurrentTrack = "";
00408     }
00409
00410     public void GenerateConfigFile(List<Playlist> playlists)
00411     {
00412         if (File.Exists(this.ConfigFilename))
00413             File.Delete(this.ConfigFilename);
00414         string output = "";
00415         output += "logfile=" + Directory.GetCurrentDirectory() + "\\\" + this.
00416 LogFilename.Replace('/', '\\') + "\n";
00417         output += "encoder_1=" + ((this.StreamType == WebradioManager.StreamType.AACPlus) ? "aacp" : "
00418 mp3") + "\n";
00419         output += "bitrate_1=" + (this.Birate * 1000) + "\n";
00420         output += "adminport=" + this.AdminPort + "\n";
00421         output += "adminuser=" + DEFAULT_ADMIN + "\n";
00422         output += "adminpassword=" + DEFAULT_ADMIN_PASSWORD + "\n";
00423         output += "captureddebug=1\n";
00424
00425         output += "outprotocol_1=" + PROTOCOL_VALUE + "\n";
00426         output += "serverip_1=" + this.Ip + "\n";
00427         output += "serverport_1=" + this.Port + "\n";
00428         output += "password_1=" + this.Password + "\n";
00429         output += "streamid_1=1\n";
00430
00431         output += "streamtitle=" + this.Name + "\n";
00432         output += "streamurl=" + this.Url + "\n";
00433         output += "genre=Misc\n";
00434
00435         output += "calendarfile=" + Directory.GetCurrentDirectory() + "\\\" + this.
00436 CalendarFile.Replace('/', '\\') + "\n";
00437         int index = 0;
00438         foreach (Playlist playlist in playlists)
00439         {
00440             index++;
00441             output += "playlistfilename_" + index.ToString() + "=" + playlist.
00442 Name + "\n";
00443             output += "playlistfilepath_" + index.ToString() + "=" + Directory.GetCurrentDirectory() +
00444 "\\\" + playlist.Filename.Replace("/", "\\") + "\n";
00445         }
00446
00447         File.WriteAllText(this.ConfigFilename, output);
00448     }
00449
00450     public bool IsRunning()
00451     {
00452         bool result = false;
00453         try
00454         {
00455             foreach (Process prc in Process.GetProcesses())
00456             {
00457                 if (prc.Id == this.Process.Id)
00458                 {
00459                     result = true;
00460                 }
00461                 if (this.Process.HasExited || !this.Process.Responding)
00462                     result = false;
00463             }
00464         }
00465         catch
00466         {
00467             return false;
00468         }
00469     }
00470
00471     public bool Start(bool debug)
00472     {
00473
00474         ProcessStartInfo StartInfo = new ProcessStartInfo(Directory.GetCurrentDirectory() +
00475 SC_TRANS_FILENAME)
00476         {
00477             CreateNoWindow = true,
00478             WindowStyle = (debug)?ProcessWindowStyle.Minimized:ProcessWindowStyle.Hidden,
00479             Arguments = "\\\" + Directory.GetCurrentDirectory() + "\\\" + this.ConfigFilename.Replace(
00480 '/', '\\') + "\n";
00481             if (this.IsRunning())
00482                 this.Process.Kill();
00483             try
00484             {
00485                 this.Process = Process.Start(StartInfo);
00486                 return true;
00487             }
00488         }
00489     }

```

```

00519         catch
00520     {
00521         return false;
00522     }
00523 }
00524 }
00525
00526 public bool Stop()
00527 {
00528
00529     if (this.IsRunning() && this.Process.Responding)
00530     {
00531         try
00532         {
00533             this.Process.Kill();
00534         }
00535         catch
00536         {
00537             return false;
00538         }
00539     }
00540     return true;
00541 }
00542 }
00543
00544 public string GetStatus()
00545 {
00546     WebClient wb = new WebClient();
00547     var data = new NameValueCollection();
00548     data["op"] = "getstatus";
00549     data["seq"] = "45";
00550     wb.Credentials = new NetworkCredential(DEFAULT_ADMIN, DEFAULT_ADMIN_PASSWORD);
00551     try
00552     {
00553         var response = wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00554         return System.Text.Encoding.UTF8.GetString(response);
00555     }
00556     catch
00557     {
00558         return string.Empty;
00559     }
00560 }
00561
00562 public void SetCaptureMode(bool active, string device)
00563 {
00564     this.Capture = active;
00565
00566     WebClient wb = new WebClient();
00567     var data = new NameValueCollection();
00568     data["op"] = "setoptions";
00569     data["seq"] = "45";
00570     data["capturedevice"] = "Contrôleur audio haute définition";
00571     wb.Credentials = new NetworkCredential(DEFAULT_ADMIN, DEFAULT_ADMIN_PASSWORD);
00572     wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00573
00574     data = new NameValueCollection();
00575     data["op"] = "capture";
00576     data["seq"] = "45";
00577     data["state"] = (active)?"on":"off";
00578     wb.Credentials = new NetworkCredential(WebradioTranscoder.DEFAULT_ADMIN,
00579     WebradioTranscoder.DEFAULT_ADMIN_PASSWORD);
00580     wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00581 }
00582
00583
00584 public void NextTrack()
00585 {
00586     WebClient wb = new WebClient();
00587     var data = new NameValueCollection();
00588     data["op"] = "nextrack";
00589     data["seq"] = "45";
00590     wb.Credentials = new NetworkCredential(WebradioTranscoder.DEFAULT_ADMIN,
00591     WebradioTranscoder.DEFAULT_ADMIN_PASSWORD);
00592     wb.UploadValues("http://127.0.0.1:" + this.AdminPort + "/api", "POST", data);
00593 }
00594
00595
00596 public override string ToString()
00597 {
00598     return this.Name;
00599 }
00600
00601 #endregion
00602
00603 }
```

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
00089     public List<WebradioServer> ActiveServers
00090     {
00091         get { return _activeServers; }
00092         set { _activeServers = value; }
00093     }
00094
00095     public List<WebradioTranscoder> ActiveTranscoders
00096     {
00097         get { return _activeTranscoders; }
00098         set { _activeTranscoders = value; }
00099     }
00100
00101     public System.Windows.Forms.Timer ProcessWatcher
00102     {
00103         get { return _processWatcher; }
00104         set { _processWatcher = value; }
00105     }
00106
00107     public Bdd Bdd
00108     {
00109         get { return _bdd; }
00110         set { _bdd = value; }
00111     }
00112
00113     public List<IController> Observers
00114     {
00115         get { return _observers; }
00116         set { _observers = value; }
00117     }
00118
00119     public Dictionary<int, Webradio> Webradios
00120     {
00121         get { return _webradios; }
00122         set { _webradios = value; }
00123     }
00124
00125     public List<AudioFile> Library
00126     {
00127         get { return _library; }
00128         set { _library = value; }
00129     }
00130
00131     #endregion
00132
00133     #region Methods
00134
00135     public WMModel()
00136     {
00137         this.Webradios = new Dictionary<int, Webradio>();
00138         this.Observers = new List<IController>();
00139         this.Bdd = new Bdd();
00140         this.Library = new List<AudioFile>();
00141         this.ActiveTranscoders = new List<WebradioTranscoder>();
00142         this.ActiveServers = new List<WebradioServer>();
00143         this.ProcessWatcher = new System.Windows.Forms.Timer();
00144         this.ProcessWatcher.Tick += ProcessWatcher_Tick;
00145         this.ProcessWatcher.Interval = 1000;
00146         this.ProcessWatcher.Start();
00147     }
00148
00149     private string GetCurrentTrackFromXML(string xml)
00150     {
00151         if (!string.IsNullOrEmpty(xml))
00152         {
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
010010
010011
010012
010013
010014
010015
010016
010017
010018
010019
010020
010021
010022
010023
010024
010025
010026
010027
010028
010029
010030
010031
010032
010033
010034
010035
010036
010037
010038
010039
010040
010041
010042
010043
010044
010045
010046
010047
010048
010049
010050
010051
010052
010053
010054
010055
010056
010057
010058
010059
010060
010061
010062
010063
010064
010065
010066
010067
010068
010069
010070
010071
010072
010073
010074
010075
010076
010077
010078
010079
010080
010081
010082
010083
010084
010085
010086
010087
010088
010089
010090
010091
010092
010093
010094
010095
010096
010097
010098
010099
0100100
0100101
0100102
0100103
0100104
0100105
0100106
0100107
0100108
0100109
0100110
0100111
0100112
0100113
0100114
0100115
0100116
0100117
0100118
0100119
0100120
0100121
0100122
0100123
0100124
0100125
0100126
0100127
0100128
0100129
0100130
0100131
0100132
0100133
0100134
0100135
0100136
0100137
0100138
0100139
0100140
0100141
0100142
0100143
0100144
0100145
0100146
0100147
0100148
0100149
0100150
0100151
0100152
0100153
0100154
0100155
0100156
0100157
0100158
0100159
0100160
0100161
0100162
0100163
0100164
0100165
0100166
0100167
0100168
0100169
0100170
0100171
0100172
0100173
0100174
0100175
0100176
0100177
0100178
0100179
0100180
0100181
0100182
0100183
0100184
0100185
0100186
0100187
0100188
0100189
0100190
0100191
0100192
0100193
0100194
0100195
0100196
0100197
0100198
0100199
0100200
0100201
0100202
0100203
0100204
0100205
0100206
0100207
0100208
0100209
0100210
0100211
0100212
0100213
0100214
0100215
0100216
0100217
0100218
0100219
0100220
0100221
0100222
0100223
0100224
0100225
0100226
0100227
0100228
0100229
0100230
0100231
0100232
0100233
0100234
0100235
0100236
0100237
0100238
0100239
0100240
0100241
0100242
0100243
0100244
0100245
0100246
0100247
0100248
0100249
0100250
0100251
0100252
0100253
0100254
0100255
0100256
0100257
0100258
0100259
0100260
0100261
0100262
0100263
0100264
0100265
0100266
0100267
0100268
0100269
0100270
0100271
0100272
0100273
0100274
0100275
0100276
0100277
0100278
0100279
0100280
0100281
0100282
0100283
0100284
0100285
0100286
0100287
0100288
0100289
0100290
0100291
0100292
0100293
0100294
0100295
0100296
0100297
0100298
0100299
0100300
0100301
0100302
0100303
0100304
0100305
0100306
0100307
0100308
0100309
0100310
0100311
0100312
0100313
0100314
0100315
0100316
0100317
0100318
0100319
0100320
0100321
0100322
0100323
0100324
0100325
0100326
0100327
0100328
0100329
0100330
0100331
0100332
0100333
0100334
0100335
0100336
0100337
0100338
0100339
0100340
0100341
0100342
0100343
0100344
0100345
0100346
0100347
0100348
0100349
0100350
0100351
0100352
0100353
0100354
0100355
0100356
0100357
0100358
0100359
0100360
0100361
0100362
0100363
0100364
0100365
0100366
0100367
0100368
0100369
0100370
0100371
0100372
0100373
0100374
0100375
0100376
0100377
0100378
0100379
0100380
0100381
0100382
0100383
0100384
0100385
0100386
0100387
0100388
0100389
0100390
0100391
0100392
0100393
0100394
0100395
0100396
0100397
0100398
0100399
0100400
0100401
0100402
0100403
0100404
0100405
0100406
0100407
0100408
0100409
0100410
0100411
0100412
0100413
0100414
0100415
0100416
0100417
0100418
0100419
0100420
0100421
0100422
0100423
0100424
0100425
0100426
0100427
0100428
0100429
0100430
0100431
0100432
0100433
0100434
0100435
0100436
0100437
0100438
0100439
0100440
0100441
0100442
0100443
0100444
0100445
0100446
0100447
0100448
0100449
0100450
0100451
0100452
0100453
0100454
0100455
0100456
0100457
0100458
0100459
0100460
0100461
0100462
0100463
0100464
0100465
0100466
0100467
0100468
0100469
0100470
0100471
0100472
0100473
0100474
0100475
0100476
0100477
0100478
0100479
0100480
0100481
0100482
0100483
0100484
0100485
0100486
0100487
0100488
0100489
0100490
0100491
0100492
0100493
0100494
0100495
0100496
0100497
0100498
0100499
0100500
0100501
0100502
0100503
0100504
0100505
0100506
0100507
0100508
0100509
0100510
0100511
0100512
0100513
0100514
0100515
0100516
0100517
0100518
0100519
0100520
0100521
0100522
0100523
0100524
0100525
0100526
0100527
0100528
0100529
0100530
0100531
0100532
0100533
0100534
0100535
0100536
0100537
0100538
0100539
0100540
0100541
0100542
0100543
0100544
0100545
0100546
0100547
0100548
0100549
0100550
0100551
0100552
0100553
0100554
0100555
0100556
0100557
0100558
0100559
0100560
0100561
0100562
0100563
0100564
0100565
0100566
0100567
0100568
0100569
0100570
0100571
0100572
0100573
0100574
0100575
0100576
0100577
0100578
0100579
0100580
0100581
0100582
0100583
0100584
0100585
0100586
0100587
0100588
0100589
0100590
0100591
0100592
0100593
0100594
0100595
0100596
0100597
0100598
0100599
0100600
0100601
0100602
0100603
0100604
0100605
0100606
0100607
0100608
0100609
0100610
0100611
0100612
0100613
0100614
0100615
0100616
0100617
0100618
0100619
0100620
0100621
0100622
0100623
0100624
0100625
0100626
0100627
0100628

```

```

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
00249 void ProcessWatcher_Tick(object sender, EventArgs e)
00250 {
00251     bool needUpdate = false;
00252     for (int i = 0; i < this.ActiveTranscoders.Count; i++)
00253     {
00254         if (!this.ActiveTranscoders[i].IsRunning())
00255         {
00256             this.ActiveTranscoders.RemoveAt(i);
00257             needUpdate = true;
00258             i--;
00259         }
00260         else
00261         {
00262             string currentTrack = this.GetCurrentTrackFromXML(this.ActiveTranscoders[i].GetStatus());
00263         }
00264         if (currentTrack != this.ActiveTranscoders[i].CurrentTrack)
00265         {
00266             this.ActiveTranscoders[i].CurrentTrack = currentTrack;
00267             needUpdate = true;
00268             if (!string.IsNullOrEmpty(currentTrack.Trim()))
00269                 this.Bdd.AddToHistory(this.ActiveTranscoders[i].Id, DateTime.Now, currentTrack);
00270         }
00271     }
00272     for (int i = 0; i < this.ActiveServers.Count; i++)
00273     {
00274         if (!this.ActiveServers[i].IsRunning())
00275         {
00276             this.ActiveServers.RemoveAt(i);
00277             needUpdate = true;
00278             i--;
00279         }
00280     }
00281     if (needUpdate)
00282         this.UpdateObservers();
00283 }
00284
00285 public void AddObserver(IController observer)
00286 {
00287     this.Observers.Add(observer);
00288 }
00289
00290 public void RemoveObserver(IController observer)
00291 {
00292     this.Observers.Remove(observer);
00293 }
00294
00295 public int GetSimiliarViewCount(int webradioId)
00296 {
00297     int ret = 0;
00298     foreach (IController controller in this.Observers)
00299     {
00300         if (controller is AdminController)
00301         {
00302             if ((controller as AdminController).View.IdWebradio == webradioId)
00303                 ret++;
00304         }
00305     }
00306     return ret;
00307 }
00308
00309 private void UpdateObservers(int webradioId)
00310 {
00311     foreach (IController controller in this.Observers)
00312     {
00313         if (controller is AdminController)
00314         {
00315             if ((controller as AdminController).View.IdWebradio == webradioId)
00316

```

```

00375             controller.UpdateView();
00376         }
00377     }
00378 }
00379
00380     private void UpdateObservers()
00381 {
00382     foreach (IController controller in this.Observers)
00383     {
00384         controller.UpdateView();
00385     }
00386 }
00387
00388     public void LoadLibrary()
00389 {
00390     this.Library = this.Bdd.LoadLibrary();
00391 }
00392
00393     public void CheckFolders(int webradioId)
00394 {
00395     if (!Directory.Exists(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name))
00396         Directory.CreateDirectory(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name);
00397 }
00398
00399     public void LoadWebradios()
00400 {
00401     if (!Directory.Exists(DEFAULT_WEBRADIOS_FOLDER))
00402         Directory.CreateDirectory(DEFAULT_WEBRADIOS_FOLDER);
00403     if (!Directory.Exists(DEFAULT_SHOUTCAST_FOLDER))
00404         Directory.CreateDirectory(DEFAULT_SHOUTCAST_FOLDER);
00405     this.Webradios = this.Bdd.LoadWebradios();
00406 }
00407
00408     public Webradio GetWebradio(int id)
00409 {
00410     return this.Webradios[id];
00411 }
00412
00413     public Webradio GetWebradioByName(string name)
00414 {
00415     Webradio ret = null;
00416     foreach (KeyValuePair<int, Webradio> webradio in this.Webradios)
00417     {
00418         if (webradio.Value.Name == name)
00419         {
00420             ret = webradio.Value;
00421             break;
00422         }
00423     }
00424     return ret;
00425 }
00426
00427     public List<Webradio> GetWebradios()
00428 {
00429     //Get only webradios with its name and id and without useless stuffs for SelectionView
00430     List<Webradio> list = new List<Webradio>();
00431     foreach (KeyValuePair<int, Webradio> wr in this.Webradios)
00432     {
00433         list.Add(new Webradio(wr.Value.Name, wr.Value.Id));
00434     }
00435     return list;
00436 }
00437
00438     public bool CreateWebradio(string name)
00439 {
00440     if (this.IsValidPath(name))
00441     {
00442         string webradioFilename = DEFAULT_WEBRADIOS_FOLDER + name + "/";
00443         Webradio wr = new Webradio(name);
00444         WebradioServer server = new WebradioServer(DEFAULT_SERVER_PORT,
00445             webradioFilename + DEFAULT_SERVER_FOLDER + DEFAULT_LOGFILENAME,
00446             webradioFilename + DEFAULT_SERVER_FOLDER + DEFAULT_CONFIGFILENAME,
00447             DEFAULT_SERVER_PASSWORD, WebradioServer.DEFAULT_ADMIN_LOGIN, DEFAULT_MAX_LISTENER);
00448         wr.Server = server;
00449         wr.Playlists = new List<Playlist>();
00450         wr.Calendar = new WebradioCalendar(webradioFilename +
00451             DEFAULT_CALENDAR_FILENAME);
00452         wr.Transcoders = new List<WebradioTranscoder>();
00453         try
00454         {
00455             wr.Id = this.Bdd.AddWebradio(wr);
00456             this.Webradios.Add(wr.Id, wr);
00457             //Directory creation
00458             Directory.CreateDirectory(webradioFilename);
00459             Directory.CreateDirectory(webradioFilename + DEFAULT_SERVER_FOLDER);
00460             Directory.CreateDirectory(webradioFilename + DEFAULT_PLAYLISTS_FOLDER);
00461             Directory.CreateDirectory(webradioFilename + DEFAULT_TRANSCODERS_FOLDER);
00462         }
00463     }
00464 }
00465
00466     public void UpdateWebradio(Webradio webradio)
00467 {
00468     if (webradio != null)
00469     {
00470         this.Bdd.UpdateWebradio(webradio);
00471         this.Webradios[webradio.Id] = webradio;
00472     }
00473 }
00474
00475     public void DeleteWebradio(int id)
00476 {
00477     if (id > 0)
00478     {
00479         Webradio wr = this.GetWebradio(id);
00480         if (wr != null)
00481         {
00482             this.Bdd.DeleteWebradio(wr);
00483             this.Webradios.Remove(wr.Id);
00484         }
00485     }
00486 }
00487
00488     public void UpdateWebradioList()
00489 {
00490     this.Webradios = this.Bdd.LoadWebradios();
00491 }
00492
00493     public void AddWebradio(Webradio webradio)
00494 {
00495     if (webradio != null)
00496     {
00497         this.Bdd.AddWebradio(webradio);
00498         this.Webradios.Add(webradio.Id, webradio);
00499     }
00500 }
00501
00502     public void RemoveWebradio(int id)
00503 {
00504     if (id > 0)
00505     {
00506         Webradio wr = this.GetWebradio(id);
00507         if (wr != null)
00508         {
00509             this.Bdd.DeleteWebradio(wr);
00510             this.Webradios.Remove(wr.Id);
00511         }
00512     }
00513 }
00514
00515     public void UpdateWebradio(Webradio webradio)
00516 {
00517     if (webradio != null)
00518     {
00519         this.Bdd.UpdateWebradio(webradio);
00520         this.Webradios[webradio.Id] = webradio;
00521     }
00522 }
00523
00524     public void DeleteWebradio(int id)
00525 {
00526     if (id > 0)
00527     {
00528         Webradio wr = this.GetWebradio(id);
00529         if (wr != null)
00530         {
00531             this.Bdd.DeleteWebradio(wr);
00532             this.Webradios.Remove(wr.Id);
00533         }
00534     }
00535 }
00536
00537     public void UpdateWebradioList()
00538 {
00539     this.Webradios = this.Bdd.LoadWebradios();
00540 }
00541
00542     public void AddWebradio(Webradio webradio)
00543 {
00544     if (webradio != null)
00545     {
00546         this.Bdd.AddWebradio(webradio);
00547         this.Webradios.Add(webradio.Id, webradio);
00548     }
00549 }
00550
00551     public void RemoveWebradio(int id)
00552 {
00553     if (id > 0)
00554     {
00555         Webradio wr = this.GetWebradio(id);
00556         if (wr != null)
00557         {
00558             this.Bdd.DeleteWebradio(wr);
00559             this.Webradios.Remove(wr.Id);
00560         }
00561     }
00562 }
00563
00564     public void UpdateWebradioList()
00565 {
00566     this.Webradios = this.Bdd.LoadWebradios();
00567 }
00568
00569     public void AddWebradio(Webradio webradio)
00570 {
00571     if (webradio != null)
00572     {
00573         this.Bdd.AddWebradio(webradio);
00574         this.Webradios.Add(webradio.Id, webradio);
00575     }
00576 }
00577
00578     public void RemoveWebradio(int id)
00579 {
00580     if (id > 0)
00581     {
00582         Webradio wr = this.GetWebradio(id);
00583         if (wr != null)
00584         {
00585             this.Bdd.DeleteWebradio(wr);
00586             this.Webradios.Remove(wr.Id);
00587         }
00588     }
00589 }
00590
00591     public void UpdateWebradioList()
00592 {
00593     this.Webradios = this.Bdd.LoadWebradios();
00594 }
00595
00596     public void AddWebradio(Webradio webradio)
00597 {
00598     if (webradio != null)
00599     {
00600         this.Bdd.AddWebradio(webradio);
00601         this.Webradios.Add(webradio.Id, webradio);
00602     }
00603 }
00604
00605     public void RemoveWebradio(int id)
00606 {
00607     if (id > 0)
00608     {
00609         Webradio wr = this.GetWebradio(id);
00610         if (wr != null)
00611         {
00612             this.Bdd.DeleteWebradio(wr);
00613             this.Webradios.Remove(wr.Id);
00614         }
00615     }
00616 }
00617
00618     public void UpdateWebradioList()
00619 {
00620     this.Webradios = this.Bdd.LoadWebradios();
00621 }
00622
00623     public void AddWebradio(Webradio webradio)
00624 {
00625     if (webradio != null)
00626     {
00627         this.Bdd.AddWebradio(webradio);
00628         this.Webradios.Add(webradio.Id, webradio);
00629     }
00630 }
00631
00632     public void RemoveWebradio(int id)
00633 {
00634     if (id > 0)
00635     {
00636         Webradio wr = this.GetWebradio(id);
00637         if (wr != null)
00638         {
00639             this.Bdd.DeleteWebradio(wr);
00640             this.Webradios.Remove(wr.Id);
00641         }
00642     }
00643 }
00644
00645     public void UpdateWebradioList()
00646 {
00647     this.Webradios = this.Bdd.LoadWebradios();
00648 }
00649
00650     public void AddWebradio(Webradio webradio)
00651 {
00652     if (webradio != null)
00653     {
00654         this.Bdd.AddWebradio(webradio);
00655         this.Webradios.Add(webradio.Id, webradio);
00656     }
00657 }
00658
00659     public void RemoveWebradio(int id)
00660 {
00661     if (id > 0)
00662     {
00663         Webradio wr = this.GetWebradio(id);
00664         if (wr != null)
00665         {
00666             this.Bdd.DeleteWebradio(wr);
00667             this.Webradios.Remove(wr.Id);
00668         }
00669     }
00670 }
00671
00672     public void UpdateWebradioList()
00673 {
00674     this.Webradios = this.Bdd.LoadWebradios();
00675 }
00676
00677     public void AddWebradio(Webradio webradio)
00678 {
00679     if (webradio != null)
00680     {
00681         this.Bdd.AddWebradio(webradio);
00682         this.Webradios.Add(webradio.Id, webradio);
00683     }
00684 }
00685
00686     public void RemoveWebradio(int id)
00687 {
00688     if (id > 0)
00689     {
00690         Webradio wr = this.GetWebradio(id);
00691         if (wr != null)
00692         {
00693             this.Bdd.DeleteWebradio(wr);
00694             this.Webradios.Remove(wr.Id);
00695         }
00696     }
00697 }
00698
00699     public void UpdateWebradioList()
00700 {
00701     this.Webradios = this.Bdd.LoadWebradios();
00702 }
00703
00704     public void AddWebradio(Webradio webradio)
00705 {
00706     if (webradio != null)
00707     {
00708         this.Bdd.AddWebradio(webradio);
00709         this.Webradios.Add(webradio.Id, webradio);
00710     }
00711 }
00712
00713     public void RemoveWebradio(int id)
00714 {
00715     if (id > 0)
00716     {
00717         Webradio wr = this.GetWebradio(id);
00718         if (wr != null)
00719         {
00720             this.Bdd.DeleteWebradio(wr);
00721             this.Webradios.Remove(wr.Id);
00722         }
00723     }
00724 }
00725
00726     public void UpdateWebradioList()
00727 {
00728     this.Webradios = this.Bdd.LoadWebradios();
00729 }
00730
00731     public void AddWebradio(Webradio webradio)
00732 {
00733     if (webradio != null)
00734     {
00735         this.Bdd.AddWebradio(webradio);
00736         this.Webradios.Add(webradio.Id, webradio);
00737     }
00738 }
00739
00740     public void RemoveWebradio(int id)
00741 {
00742     if (id > 0)
00743     {
00744         Webradio wr = this.GetWebradio(id);
00745         if (wr != null)
00746         {
00747             this.Bdd.DeleteWebradio(wr);
00748             this.Webradios.Remove(wr.Id);
00749         }
00750     }
00751 }
00752
00753     public void UpdateWebradioList()
00754 {
00755     this.Webradios = this.Bdd.LoadWebradios();
00756 }
00757
00758     public void AddWebradio(Webradio webradio)
00759 {
00760     if (webradio != null)
00761     {
00762         this.Bdd.AddWebradio(webradio);
00763         this.Webradios.Add(webradio.Id, webradio);
00764     }
00765 }
00766
00767     public void RemoveWebradio(int id)
00768 {
00769     if (id > 0)
00770     {
00771         Webradio wr = this.GetWebradio(id);
00772         if (wr != null)
00773         {
00774             this.Bdd.DeleteWebradio(wr);
00775             this.Webradios.Remove(wr.Id);
00776         }
00777     }
00778 }
00779
00780     public void UpdateWebradioList()
00781 {
00782     this.Webradios = this.Bdd.LoadWebradios();
00783 }
00784
00785     public void AddWebradio(Webradio webradio)
00786 {
00787     if (webradio != null)
00788     {
00789         this.Bdd.AddWebradio(webradio);
00790         this.Webradios.Add(webradio.Id, webradio);
00791     }
00792 }
00793
00794     public void RemoveWebradio(int id)
00795 {
00796     if (id > 0)
00797     {
00798         Webradio wr = this.GetWebradio(id);
00799         if (wr != null)
00800         {
00801             this.Bdd.DeleteWebradio(wr);
00802             this.Webradios.Remove(wr.Id);
00803         }
00804     }
00805 }
00806
00807     public void UpdateWebradioList()
00808 {
00809     this.Webradios = this.Bdd.LoadWebradios();
00810 }
00811
00812     public void AddWebradio(Webradio webradio)
00813 {
00814     if (webradio != null)
00815     {
00816         this.Bdd.AddWebradio(webradio);
00817         this.Webradios.Add(webradio.Id, webradio);
00818     }
00819 }
00820
00821     public void RemoveWebradio(int id)
00822 {
00823     if (id > 0)
00824     {
00825         Webradio wr = this.GetWebradio(id);
00826         if (wr != null)
00827         {
00828             this.Bdd.DeleteWebradio(wr);
00829             this.Webradios.Remove(wr.Id);
00830         }
00831     }
00832 }
00833
00834     public void UpdateWebradioList()
00835 {
00836     this.Webradios = this.Bdd.LoadWebradios();
00837 }
00838
00839     public void AddWebradio(Webradio webradio)
00840 {
00841     if (webradio != null)
00842     {
00843         this.Bdd.AddWebradio(webradio);
00844         this.Webradios.Add(webradio.Id, webradio);
00845     }
00846 }
00847
00848     public void RemoveWebradio(int id)
00849 {
00850     if (id > 0)
00851     {
00852         Webradio wr = this.GetWebradio(id);
00853         if (wr != null)
00854         {
00855             this.Bdd.DeleteWebradio(wr);
00856             this.Webradios.Remove(wr.Id);
00857         }
00858     }
00859 }
00860
00861     public void UpdateWebradioList()
00862 {
00863     this.Webradios = this.Bdd.LoadWebradios();
00864 }
00865
00866     public void AddWebradio(Webradio webradio)
00867 {
00868     if (webradio != null)
00869     {
00870         this.Bdd.AddWebradio(webradio);
00871         this.Webradios.Add(webradio.Id, webradio);
00872     }
00873 }
00874
00875     public void RemoveWebradio(int id)
00876 {
00877     if (id > 0)
00878     {
00879         Webradio wr = this.GetWebradio(id);
00880         if (wr != null)
00881         {
00882             this.Bdd.DeleteWebradio(wr);
00883             this.Webradios.Remove(wr.Id);
00884         }
00885     }
00886 }
00887
00888     public void UpdateWebradioList()
00889 {
00890     this.Webradios = this.Bdd.LoadWebradios();
00891 }
00892
00893     public void AddWebradio(Webradio webradio)
00894 {
00895     if (webradio != null)
00896     {
00897         this.Bdd.AddWebradio(webradio);
00898         this.Webradios.Add(webradio.Id, webradio);
00899     }
00900 }
00901
00902     public void RemoveWebradio(int id)
00903 {
00904     if (id > 0)
00905     {
00906         Webradio wr = this.GetWebradio(id);
00907         if (wr != null)
00908         {
00909             this.Bdd.DeleteWebradio(wr);
00910             this.Webradios.Remove(wr.Id);
00911         }
00912     }
00913 }
00914
00915     public void UpdateWebradioList()
00916 {
00917     this.Webradios = this.Bdd.LoadWebradios();
00918 }
00919
00920     public void AddWebradio(Webradio webradio)
00921 {
00922     if (webradio != null)
00923     {
00924         this.Bdd.AddWebradio(webradio);
00925         this.Webradios.Add(webradio.Id, webradio);
00926     }
00927 }
00928
00929     public void RemoveWebradio(int id)
00930 {
00931     if (id > 0)
00932     {
00933         Webradio wr = this.GetWebradio(id);
00934         if (wr != null)
00935         {
00936             this.Bdd.DeleteWebradio(wr);
00937             this.Webradios.Remove(wr.Id);
00938         }
00939     }
00940 }
00941
00942     public void UpdateWebradioList()
00943 {
00944     this.Webradios = this.Bdd.LoadWebradios();
00945 }
00946
00947     public void AddWebradio(Webradio webradio)
00948 {
00949     if (webradio != null)
00950     {
00951         this.Bdd.AddWebradio(webradio);
00952         this.Webradios.Add(webradio.Id, webradio);
00953     }
00954 }
00955
00956     public void RemoveWebradio(int id)
00957 {
00958     if (id > 0)
00959     {
00960         Webradio wr = this.GetWebradio(id);
00961         if (wr != null)
00962         {
00963             this.Bdd.DeleteWebradio(wr);
00964             this.Webradios.Remove(wr.Id);
00965         }
00966     }
00967 }
00968
00969     public void UpdateWebradioList()
00970 {
00971     this.Webradios = this.Bdd.LoadWebradios();
00972 }
00973
00974     public void AddWebradio(Webradio webradio)
00975 {
00976     if (webradio != null)
00977     {
00978         this.Bdd.AddWebradio(webradio);
00979         this.Webradios.Add(webradio.Id, webradio);
00980     }
00981 }
00982
00983     public void RemoveWebradio(int id)
00984 {
00985     if (id > 0)
00986     {
00987         Webradio wr = this.GetWebradio(id);
00988         if (wr != null)
00989         {
00990             this.Bdd.DeleteWebradio(wr);
00991             this.Webradios.Remove(wr.Id);
00992         }
00993     }
00994 }
00995
00996     public void UpdateWebradioList()
00997 {
00998     this.Webradios = this.Bdd.LoadWebradios();
00999 }
01000
01001     public void AddWebradio(Webradio webradio)
01002 {
01003     if (webradio != null)
01004     {
01005         this.Bdd.AddWebradio(webradio);
01006         this.Webradios.Add(webradio.Id, webradio);
01007     }
01008 }
01009
01010     public void RemoveWebradio(int id)
01011 {
01012     if (id > 0)
01013     {
01014         Webradio wr = this.GetWebradio(id);
01015         if (wr != null)
01016         {
01017             this.Bdd.DeleteWebradio(wr);
01018             this.Webradios.Remove(wr.Id);
01019         }
01020     }
01021 }
01022
01023     public void UpdateWebradioList()
01024 {
01025     this.Webradios = this.Bdd.LoadWebradios();
01026 }
01027
01028     public void AddWebradio(Webradio webradio)
01029 {
01030     if (webradio != null)
01031     {
01032         this.Bdd.AddWebradio(webradio);
01033         this.Webradios.Add(webradio.Id, webradio);
01034     }
01035 }
01036
01037     public void RemoveWebradio(int id)
01038 {
01039     if (id > 0)
01040     {
01041         Webradio wr = this.GetWebradio(id);
01042         if (wr != null)
01043         {
01044             this.Bdd.DeleteWebradio(wr);
01045             this.Webradios.Remove(wr.Id);
01046         }
01047     }
01048 }
01049
01050     public void UpdateWebradioList()
01051 {
01052     this.Webradios = this.Bdd.LoadWebradios();
01053 }
01054
01055     public void AddWebradio(Webradio webradio)
01056 {
01057     if (webradio != null)
01058     {
01059         this.Bdd.AddWebradio(webradio);
01060         this.Webradios.Add(webradio.Id, webradio);
01061     }
01062 }
01063
01064     public void RemoveWebradio(int id)
01065 {
01066     if (id > 0)
01067     {
01068         Webradio wr = this.GetWebradio(id);
01069         if (wr != null)
01070         {
01071             this.Bdd.DeleteWebradio(wr);
01072             this.Webradios.Remove(wr.Id);
01073         }
01074     }
01075 }
01076
01077     public void UpdateWebradioList()
01078 {
01079     this.Webradios = this.Bdd.LoadWebradios();
01080 }
01081
01082     public void AddWebradio(Webradio webradio)
01083 {
01084     if (webradio != null)
01085     {
01086         this.Bdd.AddWebradio(webradio);
01087         this.Webradios.Add(webradio.Id, webradio);
01088     }
01089 }
01090
01091     public void RemoveWebradio(int id)
01092 {
01093     if (id > 0)
01094     {
01095         Webradio wr = this.GetWebradio(id);
01096         if (wr != null)
01097         {
01098             this.Bdd.DeleteWebradio(wr);
01099             this.Webradios.Remove(wr.Id);
01100         }
01101     }
01102 }
01103
01104     public void UpdateWebradioList()
01105 {
01106     this.Webradios = this.Bdd.LoadWebradios();
01107 }
01108
01109     public void AddWebradio(Webradio webradio)
01110 {
01111     if (webradio != null)
01112     {
01113         this.Bdd.AddWebradio(webradio);
01114         this.Webradios.Add(webradio.Id, webradio);
01115     }
01116 }
01117
01118     public void RemoveWebradio(int id)
01119 {
01120     if (id > 0)
01121     {
01122         Webradio wr = this.GetWebradio(id);
01123         if (wr != null)
01124         {
01125             this.Bdd.DeleteWebradio(wr);
01126             this.Webradios.Remove(wr.Id);
01127         }
01128     }
01129 }
01130
01131     public void UpdateWebradioList()
01132 {
01133     this.Webradios = this.Bdd.LoadWebradios();
01134 }
01135
01136     public void AddWebradio(Webradio webradio)
01137 {
01138     if (webradio != null)
01139     {
01140         this.Bdd.AddWebradio(webradio);
01141         this.Webradios.Add(webradio.Id, webradio);
01142     }
01143 }
01144
01145     public void RemoveWebradio(int id)
01146 {
01147     if (id > 0)
01148     {
01149         Webradio wr = this.GetWebradio(id);
01150         if (wr != null)
01151         {
01152             this.Bdd.DeleteWebradio(wr);
01153             this.Webradios.Remove(wr.Id);
01154         }
01155     }
01156 }
01157
01158     public void UpdateWebradioList()
01159 {
01160     this.Webradios = this.Bdd.LoadWebradios();
01161 }
01162
01163     public void AddWebradio(Webradio webradio)
01164 {
01165     if (webradio != null)
01166     {
01167         this.Bdd.AddWebradio(webradio);
01168         this.Webradios.Add(webradio.Id, webradio);
01169     }
01170 }
01171
01172     public void RemoveWebradio(int id)
01173 {
01174     if (id > 0)
01175     {
01176         Webradio wr = this.GetWebradio(id);
01177         if (wr != null)
01178         {
01179             this.Bdd.DeleteWebradio(wr);
01180             this.Webradios.Remove(wr.Id);
01181         }
01182     }
01183 }
01184
01185     public void UpdateWebradioList()
01186 {
01187     this.Webradios = this.Bdd.LoadWebradios();
01188 }
01189
01190     public void AddWebradio(Webradio webradio)
01191 {
01192     if (webradio != null)
01193     {
01194         this.Bdd.AddWebradio(webradio);
01195         this.Webradios.Add(webradio.Id, webradio);
01196     }
01197 }
01198
01199     public void RemoveWebradio(int id)
01200 {
01201     if (id > 0)
01202     {
01203         Webradio wr = this.GetWebradio(id);
01204         if (wr != null)
01205         {
01206             this.Bdd.DeleteWebradio(wr);
01207             this.Webradios.Remove(wr.Id);
01208         }
01209     }
01210 }
01211
01212     public void UpdateWebradioList()
01213 {
01214     this.Webradios = this.Bdd.LoadWebradios();
01215 }
01216
01217     public void AddWebradio(Webradio webradio)
01218 {
01219     if (webradio != null)
01220     {
01221         this.Bdd.AddWebradio(webradio);
01222         this.Webradios.Add(webradio.Id, webradio);
01223     }
01224 }
01225
01226     public void RemoveWebradio(int id)
01227 {
01228     if (id > 0)
01229     {
01230         Webradio wr = this.GetWebradio(id);
01231         if (wr != null)
01232         {
01233             this.Bdd.DeleteWebradio(wr);
01234             this.Webradios.Remove(wr.Id);
01235         }
01236     }
01237 }
01238
01239     public void UpdateWebradioList()
01240 {
01241     this.Webradios = this.Bdd.LoadWebradios();
01242 }
01243
01244     public void AddWebradio(Webradio webradio)
01245 {
01246     if (webradio != null)
01247     {
01248         this.Bdd.AddWebradio(webradio);
01249         this.Webradios.Add(webradio.Id, webradio);
01250     }
01251 }
01252
01253     public void RemoveWebradio(int id)
01254 {
01255     if (id > 0)
01256     {
01257         Webradio wr = this.GetWebradio(id);
01258         if (wr != null)
01259         {
01260             this.Bdd.DeleteWebradio(wr);
01261             this.Webradios.Remove(wr.Id);
01262         }
01263     }
01264 }
01265
01266     public void UpdateWebradioList()
01267 {
01268     this.Webradios = this.Bdd.LoadWebradios();
01269 }
01270
01271     public void AddWebradio(Webradio webradio)
01272 {
01273     if (webradio != null)
01274     {
01275         this.Bdd.AddWebradio(webradio);
01276         this.Webradios.Add(webradio.Id, webradio);
01277     }
01278 }
01279
01280     public void RemoveWebradio(int id)
01281 {
01282     if (id > 0)
01283     {
01284         Webradio wr = this.GetWebradio(id);
01285         if (wr != null)
01286         {
01287             this.Bdd.DeleteWebradio(wr);
01288             this.Webradios.Remove(wr.Id);
01289         }
01290     }
01291 }
01292
01293     public void UpdateWebradioList()
01294 {
01295     this.Webradios = this.Bdd.LoadWebradios();
01296 }
01297
01298     public void AddWebradio(Webradio webradio)
01299 {
01300     if (webradio != null)
01301     {
01302         this.Bdd.AddWebradio(webradio);
01303         this.Webradios.Add(webradio.Id, webradio);
01304     }
01305 }
01306
01307     public void RemoveWebradio(int id)
01308 {
01309     if (id > 0)
01310     {
01311         Webradio wr = this.GetWebradio(id);
01312         if (wr != null)
01313         {
01314             this.Bdd.DeleteWebradio(wr);
01315             this.Webradios.Remove(wr.Id);
01316         }
01317     }
01318 }
01319
01320     public void UpdateWebradioList()
01321 {
01322     this.Webradios = this.Bdd.LoadWebradios();
01323 }
01324
01325     public void AddWebradio(Webradio webradio)
01326 {
01327     if (webradio != null)
01328     {
01329         this.Bdd.AddWebradio(webradio);
01330         this.Webradios.Add(webradio.Id, webradio);
01331     }
01332 }
01333
01334     public void RemoveWebradio(int id)
01335 {
01336     if (id > 0)
01337     {
01338         Webradio wr = this.GetWebradio(id);
01339         if (wr != null)
01340         {
01341             this.Bdd.DeleteWebradio(wr);
01342             this.Webradios.Remove(wr.Id);
01343         }
01344     }
01345 }
01346
01347     public void UpdateWebradioList()
01348 {
01349     this.Webradios = this.Bdd.LoadWebradios();
01350 }
01351
01352     public void AddWebradio(Webradio webradio)
01353 {
01354     if (webradio != null)
01355     {
01356         this.Bdd.AddWebradio(webradio);
01357         this.Webradios.Add(webradio.Id, webradio);
01358     }
01359 }
01360
01361     public void RemoveWebradio(int id)
01362 {
01363     if (id > 0)
01364     {
01365         Webradio wr = this.GetWebradio(id);
01366         if (wr != null)
01367         {
01368             this.Bdd.DeleteWebradio(wr);
01369             this.Webradios.Remove(wr.Id);
01370
```

```

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,
00673                 newWebradio.Id);
00674             if (newPlaylistEvent != null)
00675             {
00676                 CalendarEvent newEvent = new
00677                     CalendarEvent(ev.Name, ev.StartTime, ev.Duration, ev.
00678                     Repeat, ev.Priority, ev.Shuffle, ev.Loopatend, newPlaylistEvent);
00679                     this.CreateEvent(newEvent, newWebradio.Id);
00680             }
00681         }
00682     }
00683     //TRANSCODER CONFIGURATION
00684     foreach (WebradioTranscoder transcoder in webradio.
00685         Transcoders)
00686     {
00687         this.CreateTranscoder(transcoder.Name,
00688             transcoder.StreamType, transcoder.SampleRate, transcoder.Birate, transcoder.Url,
00689             transcoder.Ip, transcoder.Port, transcoder.AdminPort, transcoder.Password, newWebradio.Id);
00690     }
00691     this.UpdateObservers();
00692     return true;
00693 }
00694 }
00695 }
00696
00708     public List<AudioFile> GetLibrary()
00709     {
00710         return this.Library;
00711     }
00724     public List<string> GetGenders()
00725     {
00726         return this.Bdd.GetGenders();
00727     }
00743     public bool ImportFilesToLibrary(string[] filenames,
00744         AudioType type)
00745     {
00746         AudioFile file;
00747         bool state = true;
00748         foreach (string filename in filenames)
00749         {
00750             try
00751             {
00752                 TagLib.File tagFile = TagLib.File.Create(filename);
00753                 if (type == AudioType.Music)
00754                     file = new Music(filename,
00755                         tagFile.Tag.Title,
00756                         tagFile.Tag.FirstPerformer,
00757                         tagFile.Tag.Album,
00758                         (int)tagFile.Tag.Year,
00759                         tagFile.Tag.Copyright,
00760                         tagFile.Properties.Duration,
00761                         tagFile.Tag.FirstGenre);
00762                 else
00763                     file = new Ad(filename,
00764                         tagFile.Tag.Title,
00765                         tagFile.Tag.FirstPerformer,
00766                         tagFile.Tag.Album,
00767                         (int)tagFile.Tag.Year,
00768                         tagFile.Tag.Copyright,
00769                         tagFile.Properties.Duration,
00770                         tagFile.Tag.FirstGenre);
00771
00772                 int id = this.Bdd.AddAudioFile(file);
00773                 if (id != Bdd.ERROR)
00774                 {
00775                     file.Id = id;
00776                     this.Library.Add(file);
00777                     this.UpdateObservers();
00778                 }
00779                 state = true;
00780             }
00781             catch
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
00829     public bool UpdateAudiofile(Audiofile file)
00830     {
00831         if (this.Bdd.UpdateAudiofile(file))
00832         {
00833             try
00834             {
00835                 TagLib.File tagFile = TagLib.File.Create(file.Filename);
00836                 tagFile.Tag.Title = file.Title;
00837                 tagFile.Tag.Performers = new string[] { file.Artist };
00838                 tagFile.Tag.Album = file.Album;
00839                 tagFile.Tag.Year = (uint)file.Year;
00840                 tagFile.Tag.Copyright = file.Label;
00841                 tagFile.Tag.Genres = new string[] { file.Gender };
00842                 tagFile.Save();
00843
00844                 foreach (Audiofile af in this.Library)
00845                 {
00846                     if (af.Id == file.Id)
00847                     {
00848                         af.Title = file.Title;
00849                         af.Artist = file.Artist;
00850                         af.Album = file.Album;
00851                         af.Year = file.Year;
00852                         af.Label = file.Label;
00853                         af.Gender = file.Gender;
00854                         break;
00855                     }
00856                 }
00857                 this.UpdateObservers();
00858                 return true;
00859             }
00860             catch
00861             {
00862                 return false;
00863             }
00864         }
00865         else
00866             return false;
00867     }
00868 }
00869
00870     private bool IsValidfilename(string testName)
00871     {
00872         char[] invalidFileChars = Path.GetInvalidfilenameChars();
00873         if (testName.IndexOfAny(invalidFileChars) == -1)
00874             return true;
00875         else
00876             return false;
00877     }
00878
00879     private bool IsValidPath(string testName)
00880     {
00881         char[] invalidPathChars = Path.GetInvalidPathChars();
00882         if (testName.IndexOfAny(invalidPathChars) == -1)
00883             return true;
00884         else
00885             return false;
00886     }
00887
00888     public bool CreatePlaylist(string name, string webradioName, int webradioId,
00889     AudioType type, out Playlist newPlaylist)
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02349
02350
02351
02352
02353
02354
02355
02356
02357
02358
02359
02359
02360
02361
02362
02363
02364
02365
02366
02367
02368
02369
02369
02370
02371
02372
02373
02374
02375
02376
02377
02378
02379
02379
02380
02381
02382
02383
02384
02385
02386
02387
02388
02389
02389
02390
02391
02392
02393
02394
02395
02396
02397
02398
02399
02399
02400
02401
02402
02403
02404
02405
02406
02407
02408
02409
02409
02410
02411
02412
02413
02414
02415
02416
02417
02418
02419
02419
02420
02421
02422
02423
02424
02425
02426
02427
02428
02429
02429
02430
02431
02432
02433
02434
02435
02436
02437
02438
02439
02439
02440
02441
02442
02443
02444
02445
02446
02447
02448
02449
02449
02450
02451
02452
02453
02454
02455
02456
02457
02458
02459
02459
02460
02461
02462
02463
02464
02465
02466
02467
02468
02469
02469
02470
02471
02472
02473
02474
02475
02476
02477
02478
02479
02479
02480
02481
02482
02483
02484
02485
02486
02487
02488
02489
02489
02490
02491
02492
02493
02494
02495
02496
02497
02498
02499
02499
02500
02501
02502
02503
02504
02505
02506
02507
02508
02509
02509
02510
02511
02512
02513
02514
02515
02516
02517
02518
02519
02519
02520
02521
0
```

```

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
00952              + name + ".lst";
00953              if (type == AudioType.Music)
00954                  newPlaylist = new PlaylistMusic(name, filename);
00955              else
00956                  newPlaylist = new PlaylistAd(name, filename);
00957              id = this.Bdd.CreatePlaylist(newPlaylist, webradioId);
00958              if (id == Bdd.ERROR)
00959                  return false;
00960              else
00961              {
00962                  try
00963                  {
00964                      newPlaylist.Id = id;
00965                      newPlaylist.GenerateConfigFile();
00966                      selectedWebradio.Playlists.Add(newPlaylist);
00967                      this.UpdateObservers(webradioId);
00968                      return true;
00969                  }
00970                  catch
00971                  {
00972                      return false;
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.
01031              Id))
01032          {
01033              playlist.AudioFileList.Add(audioFile.Value);
01034              state = true;
01035          }
01036          else
01037          {
01038              state = false;
01039              break;
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.
01067              Id))
01068          {
01069              playlist.AudioFileList.Remove(audioFile.Value);
01070          }
01071      }
01072  }

```

```

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,
01128         AudioType type, string gender, int webradioId, string webradioName)
01129     {
01130         Playlist newPlaylist;
01131         string filename = DEFAULT_WEBRADIOS_FOLDER + webradioName + "/" + DEFAULT_PLAYLISTS_FOLDER +
01132             name + ".lst";
01133         if (type == AudioType.Music)
01134             newPlaylist = new PlaylistMusic(name, filename);
01135         else
01136             newPlaylist = new PlaylistAd(name, filename);
01137         TimeSpan tmpDuration = new TimeSpan();
01138         Random random = new Random();
01139         AudioFile audioFile;
01140         int retries = 0;
01141         List<int> audioFilesId = new List<int>();
01142         while (tmpDuration <= duration)
01143         {
01144             audioFile = this.Library[random.Next(0, this.Library.Count - 1)];
01145             if (audioFile.Type == type)
01146             {
01147                 if (retries > MAX_TRY_GENERATE)
01148                     break;
01149                 if ((tmpDuration + audioFile.Duration) > duration)
01150                 {
01151                     retries++;
01152                     continue;
01153                 }
01154                 retries = 0;
01155                 if ((audioFile.Type == AudioType.Ad) || (audioFile.Type == AudioType.Music
01156                 && audioFile.Gender == gender))
01157                 {
01158                     tmpDuration += audioFile.Duration;
01159                     newPlaylist.AudioFileList.Add(audioFile.Filename);
01160                     audioFilesId.Add(audioFile.Id);
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.
01197             Id))
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02349
02350
02351
02352
02353
02354
02355
02356
02357
02358
02359
02359
02360
02361
02362
02363
02364
02365
02366
02367
02368
02369
02369
02370
02371
02372
02373
02374
02375
02376
02377
02378
02379
02379
02380
02381
02382
02383
02384
02385
02386
02387
02388
02389
02389
02390
02391
02392
02393
02394
02395
02396
02397
02398
02399
02399
02400
02401
02402
02403
02404
02405
02406
02407
02408
02409
02409
02410
02411
02412
02413
02414
02415
02416
02417
02418
02419
02419
02420
02421
02422
02423
02424
02425
02426
02427
02428
02429
02429
02430
02431
02432
02433
02434
02435
02436
02437
02438
02439
02439
02440
02441
02442
02443
02444
02445
02446
02447
02448
02449
02449
02450
02451
02452
02453
02454
02455
02456
02457
02458
02459
02459
02460
02461
02462
02463
02464
02465
02466
02467
02468
02469
02469
02470
02471
02472
02473
02474
02475
02476
02477
02478
02479
02479
02480
02481
02482
02483
02484
02485
02486
02487
02488
02489
02489
02490
02491
02492
02493
02494
02495
02496
02497
02498
02499
02499
02500
02501
02502
02503
02504
02505
02506
02507
02508
02509
02509
02510
02511
02512
02513
02514
02515
02516
02517
02518
02519
02519
02520
02521
02522
02523
02524
02525
02526
02527
02528
02529
02529
02530
02531
02532
02533
02534
02535
02536
02537
02538
02539
02539
02540
02541
02542
02543
02544
02545
02546
02547
02548
02549
02549
02550
02551
02552
02553
02554
02555
02556
02557
02558
02559
02559
02560
02561
02562
02563
02564
02565
02566
02567
02568
02569
02569
02570
02571
02572
02573
02574
02575
02576
02577
02578
02579
02579
02580
02581
02582
02583
02584
02585
02586
02587
02588
02589
02589
02590
02591
02592
02593
02594
02595
02596
02597
02598
02599
02599
02600
02601
02602
02603
02604
02605
02606
02607
02608
02609
02609
02610
02611
02612
02613
02614
02615
02616
02617
02618
02619
02619
02620
02621
02622
02623
02624
02625
02626
02627
02628
02629
02629
02630
02631
02632
02633
02634
02635
02636
02637
02638
02639
02639
02640
02641
02642
02643
02644
02645
02646
02647
02648
02649
02649
02650
02651
02652
02653
02654
02655
02656
02657
02658
02659
02659
02660
02661
02662
02663
02664
02665
02666
02667
02668
02669
02669
02670
02671
02672
02673
02674
02675
02676
02677
02678
02679
02679
02680
02681
02682
02683
02684
02685
02686
02687
02688
02689
02689
02690
02691
02692
02693
02694
02695
02696
02697
02698
02699
02699
02700
02701
02702
02703
02704
02705
02706
02707
02708
02709
02709
02710
02711
02712
02713
02714
02715
02716
02717
02718
02719
02719
02720
02721
02722
02723
02724
02725
02726
02727
02728
02729
02729
02730
02731
02732
02733
02734
02735
02736
02737
02738
02739
02739
02740
02741
02742
02743
02744
02745
02746
02747
02748
02749
02749
02750
02751
02752
02753
02754
02755
02756
02757
02758
02759
02759
02760
02761
02762
02763
02764
02765
02766
02767
02768
02769
02769
02770
02771
02772
02773
02774
02775
02776
02777
02778
02779
02779
02780
02781
02782
02783
02784
02785
02786
02787
02788
02789
02789
02790
02791
02792
02793
02794
027
```

```

01197             return false;
01198
01199         int id = this.Bdd.AddEvent(newEvent, this.Webradios[webradioId].Calendar.Id,
01200             newEvent.Playlist.Id);
01201         newEvent.Id = id;
01202         this.Webradios[webradioId].Calendar.Events.Add(newEvent);
01203         this.Webradios[webradioId].Calendar.GenerateConfigFile();
01204         this.UpdateObservers(webradioId);
01205         return true;
01206     }
01207
01208     public bool UpdateEvent(CalendarEvent aEvent, int webradioId)
01209     {
01210         if (this.Bdd.UpdateEvent(aEvent))
01211         {
01212             foreach (CalendarEvent ce in this.Webradios[webradioId].Calendar.Events)
01213             {
01214                 CalendarEvent tmp = ce;
01215                 if (ce.Id == aEvent.Id)
01216                 {
01217                     tmp.StartTime = aEvent.StartTime;
01218                     tmp.Duration = aEvent.Duration;
01219                     break;
01220                 }
01221             }
01222             this.Webradios[webradioId].Calendar.GenerateConfigFile();
01223             this.UpdateObservers(webradioId);
01224             return true;
01225         }
01226         else
01227             return false;
01228     }
01229
01230     public bool DeleteEvent(CalendarEvent aEvent, int webradioId)
01231     {
01232         if (this.Bdd.DeleteEvent(aEvent))
01233         {
01234             this.Webradios[webradioId].Calendar.Events.Remove(aEvent);
01235             this.Webradios[webradioId].Calendar.GenerateConfigFile();
01236             this.UpdateObservers(webradioId);
01237             return true;
01238         }
01239         else
01240             return false;
01241     }
01242
01243     public bool CreateTranscoder(string name, StreamType st, int sampleRate,
01244         int bitrate, string url, IPAddress ip, int port, int adminport, string password, int webradioId)
01245     {
01246         string filename = DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name + "/" +
01247             DEFAULT_TRANSCODERS_FOLDER;
01248         WebradioTranscoder transcoder;
01249         if (st == StreamType.AACPlus)
01250             transcoder = new TranscoderAacPlus(name,
01251                 bitrate,
01252                 sampleRate,
01253                 ip,
01254                 port,
01255                 adminport,
01256                 url,
01257                 password,
01258                 filename,
01259                 filename);
01260         else
01261             transcoder = new TranscoderMp3(name,
01262                 bitrate,
01263                 sampleRate,
01264                 ip,
01265                 port,
01266                 adminport,
01267                 url,
01268                 password,
01269                 filename,
01270                 filename);
01271
01272         transcoder.CalendarFile = DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name + "/" +
01273             DEFAULT_CALENDAR_FILENAME;
01274         int id = this.Bdd.AddTranscoder(transcoder, webradioId);
01275         if (id == Bdd.ERROR)
01276             return false;
01277         transcoder.Id = id;
01278         transcoder.ConfigFilename = filename + id.ToString() + ".config";
01279         transcoder.LogFilename = filename + id.ToString() + ".log";
01280         this.Webradios[webradioId].Transcoders.Add(transcoder);
01281         transcoder.GenerateConfigFile(this.Webradios[webradioId].Playlists);
01282         this.Webradios[webradioId].GenerateConfigFiles();
01283         this.UpdateObservers();
01284     }

```

```
01330         return true;
01331     }
01332 
01347     public bool DeleteTranscoder(WebradioTranscoder transcoder, int
01348     webradioId)
01349     {
01350         if (transcoder.IsRunning())
01351             transcoder.Stop();
01352 
01353         if (this.Bdd.DeleteTranscoder(transcoder.Id))
01354         {
01355             System.IO.File.Delete(transcoder.ConfigFilename);
01356             if (System.IO.File.Exists(transcoder.LogFilename))
01357                 System.IO.File.Delete(transcoder.LogFilename);
01358             this.Webradios[webradioId].Transcoders.Remove(transcoder);
01359             this.UpdateObservers(webradioId);
01360             return true;
01361         }
01362         else
01363             return false;
01364     }
01365 
01380     public bool UpdateTranscoder(WebradioTranscoder transcoder, bool
01381     debug, int webradioId)
01382     {
01383         try
01384         {
01385             bool wasRunning = false;
01386             if (transcoder.IsRunning())
01387             {
01388                 wasRunning = true;
01389                 transcoder.Process.Kill();
01390             }
01391 
01392             this.Bdd.UpdateTranscoder(transcoder);
01393             transcoder.GenerateConfigFile(this.Webradios[webradioId].Playlists);
01394             if (wasRunning)
01395                 transcoder.Start(debug);
01396             this.UpdateObservers(webradioId);
01397             return true;
01398         }
01399         catch
01400         {
01401             return false;
01402         }
01403     }
01419     public bool StartTranscoder(WebradioTranscoder transcoder, bool
01420     debug, int webradioId)
01421     {
01422         try
01423         {
01424             if (transcoder.Start(debug))
01425             {
01426                 this.ActiveTranscoders.Add(transcoder);
01427                 UpdateObservers(webradioId);
01428                 return true;
01429             }
01430             else
01431                 return false;
01432         }
01433         catch
01434         {
01435             return false;
01436         }
01437     }
01438 }
01439 
01454     public bool StopTranscoder(WebradioTranscoder transcoder, int
01455     webradioId)
01456     {
01457         try
01458         {
01459             if (transcoder.Stop())
01460             {
01461                 this.ActiveTranscoders.Remove(transcoder);
01462                 this.Webradios[webradioId].Calendar.GenerateConfigFile();
01463                 this.UpdateObservers(webradioId);
01464                 return true;
01465             }
01466             else
01467                 return false;
01468         }
01469         catch
01470         {
01471             return false;
01472         }
01473     }
```

```

01471         }
01472     }
01473
01474     public bool StopAllProcess(int webradioId)
01475     {
01476         try
01477         {
01478             foreach (WebradioTranscoder transcoder in this.Webradios[webradioId].
01479                 Transcoders)
01480             {
01481                 transcoder.Stop();
01482             }
01483             this.Webradios[webradioId].Server.Stop();
01484             return true;
01485         }
01486         catch
01487         {
01488             return false;
01489         }
01490     }
01491
01492     public bool StopAllProcess()
01493     {
01494         try
01495         {
01496             foreach (KeyValuePair<int, Webradio> webradio in this.Webradios)
01497             {
01498                 foreach (WebradioTranscoder transcoder in webradio.Value.Transcoders)
01499                 {
01500                     transcoder.Stop();
01501                 }
01502                 webradio.Value.Server.Stop();
01503             }
01504             return true;
01505         }
01506         catch
01507         {
01508             return false;
01509         }
01510     }
01511
01512     public void GenerateConfigFiles(int webradioId)
01513     {
01514         this.Webradios[webradioId].GenerateConfigFiles();
01515     }
01516
01517     public bool UpdateServer(bool debug, int port, string password, string adminPassword,
01518     int maxListener, int webradioId)
01519     {
01520         try
01521         {
01522             WebradioServer server = this.Webradios[webradioId].Server;
01523             bool wasRunning = false;
01524             if (server.IsRunning())
01525             {
01526                 wasRunning = true;
01527                 server.Process.Kill();
01528             }
01529             if (!this.Bdd.UpdateServer(port, password, adminPassword, maxListener,
01530             webradioId))
01531                 return false;
01532
01533             server.AdminPassword = adminPassword;
01534             server.Password = password;
01535             server.Port = port;
01536             server.MaxListener = maxListener;
01537             server.GenerateConfigFile();
01538             if (wasRunning)
01539                 server.Start(debug);
01540
01541             this.UpdateObservers(webradioId);
01542             return true;
01543         }
01544         catch
01545         {
01546             return false;
01547         }
01548     }
01549
01550     public bool StartServer(int webradioId, bool debug)
01551     {
01552         if (this.Webradios[webradioId].Server.Start(debug))
01553         {
01554             this.ActiveServers.Add(this.Webradios[webradioId].Server);
01555         }
01556     }
01557
01558     public void StopServer(int webradioId)
01559     {
01560         if (this.Webradios[webradioId].Server.Stop())
01561         {
01562             this.ActiveServers.Remove(this.Webradios[webradioId].Server);
01563         }
01564     }
01565
01566     public void StopAllServers()
01567     {
01568         foreach (Webradio webradio in this.Webradios)
01569         {
01570             if (webradio.Server != null)
01571                 webradio.Server.Stop();
01572         }
01573     }
01574
01575     public void StopAllServers(bool debug)
01576     {
01577         foreach (Webradio webradio in this.Webradios)
01578         {
01579             if (webradio.Server != null)
01580                 webradio.Server.Stop(debug);
01581         }
01582     }
01583
01584     public void StartAllServers()
01585     {
01586         foreach (Webradio webradio in this.Webradios)
01587         {
01588             if (webradio.Server != null)
01589                 webradio.Server.Start();
01590         }
01591     }
01592
01593     public void StopAllServers()
01594     {
01595         foreach (Webradio webradio in this.Webradios)
01596         {
01597             if (webradio.Server != null)
01598                 webradio.Server.Stop();
01599         }
01600     }
01601
01602     public void StartAllServers()
01603     {
01604         foreach (Webradio webradio in this.Webradios)
01605         {
01606             if (webradio.Server != null)
01607                 webradio.Server.Start();
01608         }
01609     }
01610
01611     public void StopAllServers(bool debug)
01612     {
01613         foreach (Webradio webradio in this.Webradios)
01614         {
01615             if (webradio.Server != null)
01616                 webradio.Server.Stop(debug);
01617         }
01618     }
01619
01620     public void StartAllServers(bool debug)
01621     {
01622         foreach (Webradio webradio in this.Webradios)
01623         {
01624             if (webradio.Server != null)
01625                 webradio.Server.Start(debug);
01626         }
01627     }
01628
01629     public void StopAllServers()
01630     {
01631         foreach (Webradio webradio in this.Webradios)
01632         {
01633             if (webradio.Server != null)
01634                 webradio.Server.Stop();
01635         }
01636     }
01637
01638     public void StartAllServers()
01639     {
01640         foreach (Webradio webradio in this.Webradios)
01641         {
01642             if (webradio.Server != null)
01643                 webradio.Server.Start();
01644         }
01645     }
01646
01647     public void StopAllServers(bool debug)
01648     {
01649         foreach (Webradio webradio in this.Webradios)
01650         {
01651             if (webradio.Server != null)
01652                 webradio.Server.Stop(debug);
01653         }
01654     }
01655
01656     public void StartAllServers(bool debug)
01657     {
01658         foreach (Webradio webradio in this.Webradios)
01659         {
01660             if (webradio.Server != null)
01661                 webradio.Server.Start(debug);
01662         }
01663     }
01664
01665     public void StopAllServers()
01666     {
01667         foreach (Webradio webradio in this.Webradios)
01668         {
01669             if (webradio.Server != null)
01670                 webradio.Server.Stop();
01671         }
01672     }
01673
01674     public void StartAllServers()
01675     {
01676         foreach (Webradio webradio in this.Webradios)
01677         {
01678             if (webradio.Server != null)
01679                 webradio.Server.Start();
01680         }
01681     }
01682
01683     public void StopAllServers(bool debug)
01684     {
01685         foreach (Webradio webradio in this.Webradios)
01686         {
01687             if (webradio.Server != null)
01688                 webradio.Server.Stop(debug);
01689         }
01690     }
01691
01692     public void StartAllServers(bool debug)
01693     {
01694         foreach (Webradio webradio in this.Webradios)
01695         {
01696             if (webradio.Server != null)
01697                 webradio.Server.Start(debug);
01698         }
01699     }
01700
01701     public void StopAllServers()
01702     {
01703         foreach (Webradio webradio in this.Webradios)
01704         {
01705             if (webradio.Server != null)
01706                 webradio.Server.Stop();
01707         }
01708     }
01709
01710     public void StartAllServers()
01711     {
01712         foreach (Webradio webradio in this.Webradios)
01713         {
01714             if (webradio.Server != null)
01715                 webradio.Server.Start();
01716         }
01717     }
01718
01719     public void StopAllServers(bool debug)
01720     {
01721         foreach (Webradio webradio in this.Webradios)
01722         {
01723             if (webradio.Server != null)
01724                 webradio.Server.Stop(debug);
01725         }
01726     }
01727
01728     public void StartAllServers(bool debug)
01729     {
01730         foreach (Webradio webradio in this.Webradios)
01731         {
01732             if (webradio.Server != null)
01733                 webradio.Server.Start(debug);
01734         }
01735     }
01736
01737     public void StopAllServers()
01738     {
01739         foreach (Webradio webradio in this.Webradios)
01740         {
01741             if (webradio.Server != null)
01742                 webradio.Server.Stop();
01743         }
01744     }
01745
01746     public void StartAllServers()
01747     {
01748         foreach (Webradio webradio in this.Webradios)
01749         {
01750             if (webradio.Server != null)
01751                 webradio.Server.Start();
01752         }
01753     }
01754
01755     public void StopAllServers(bool debug)
01756     {
01757         foreach (Webradio webradio in this.Webradios)
01758         {
01759             if (webradio.Server != null)
01760                 webradio.Server.Stop(debug);
01761         }
01762     }
01763
01764     public void StartAllServers(bool debug)
01765     {
01766         foreach (Webradio webradio in this.Webradios)
01767         {
01768             if (webradio.Server != null)
01769                 webradio.Server.Start(debug);
01770         }
01771     }
01772
01773     public void StopAllServers()
01774     {
01775         foreach (Webradio webradio in this.Webradios)
01776         {
01777             if (webradio.Server != null)
01778                 webradio.Server.Stop();
01779         }
01780     }
01781
01782     public void StartAllServers()
01783     {
01784         foreach (Webradio webradio in this.Webradios)
01785         {
01786             if (webradio.Server != null)
01787                 webradio.Server.Start();
01788         }
01789     }
01790
01791     public void StopAllServers(bool debug)
01792     {
01793         foreach (Webradio webradio in this.Webradios)
01794         {
01795             if (webradio.Server != null)
01796                 webradio.Server.Stop(debug);
01797         }
01798     }
01799
01800     public void StartAllServers(bool debug)
01801     {
01802         foreach (Webradio webradio in this.Webradios)
01803         {
01804             if (webradio.Server != null)
01805                 webradio.Server.Start(debug);
01806         }
01807     }
01808
01809     public void StopAllServers()
01810     {
01811         foreach (Webradio webradio in this.Webradios)
01812         {
01813             if (webradio.Server != null)
01814                 webradio.Server.Stop();
01815         }
01816     }
01817
01818     public void StartAllServers()
01819     {
01820         foreach (Webradio webradio in this.Webradios)
01821         {
01822             if (webradio.Server != null)
01823                 webradio.Server.Start();
01824         }
01825     }
01826
01827     public void StopAllServers(bool debug)
01828     {
01829         foreach (Webradio webradio in this.Webradios)
01830         {
01831             if (webradio.Server != null)
01832                 webradio.Server.Stop(debug);
01833         }
01834     }
01835
01836     public void StartAllServers(bool debug)
01837     {
01838         foreach (Webradio webradio in this.Webradios)
01839         {
01840             if (webradio.Server != null)
01841                 webradio.Server.Start(debug);
01842         }
01843     }
01844
01845     public void StopAllServers()
01846     {
01847         foreach (Webradio webradio in this.Webradios)
01848         {
01849             if (webradio.Server != null)
01850                 webradio.Server.Stop();
01851         }
01852     }
01853
01854     public void StartAllServers()
01855     {
01856         foreach (Webradio webradio in this.Webradios)
01857         {
01858             if (webradio.Server != null)
01859                 webradio.Server.Start();
01860         }
01861     }
01862
01863     public void StopAllServers(bool debug)
01864     {
01865         foreach (Webradio webradio in this.Webradios)
01866         {
01867             if (webradio.Server != null)
01868                 webradio.Server.Stop(debug);
01869         }
01870     }
01871
01872     public void StartAllServers(bool debug)
01873     {
01874         foreach (Webradio webradio in this.Webradios)
01875         {
01876             if (webradio.Server != null)
01877                 webradio.Server.Start(debug);
01878         }
01879     }
01880
01881     public void StopAllServers()
01882     {
01883         foreach (Webradio webradio in this.Webradios)
01884         {
01885             if (webradio.Server != null)
01886                 webradio.Server.Stop();
01887         }
01888     }
01889
01890     public void StartAllServers()
01891     {
01892         foreach (Webradio webradio in this.Webradios)
01893         {
01894             if (webradio.Server != null)
01895                 webradio.Server.Start();
01896         }
01897     }
01898
01899     public void StopAllServers(bool debug)
01900     {
01901         foreach (Webradio webradio in this.Webradios)
01902         {
01903             if (webradio.Server != null)
01904                 webradio.Server.Stop(debug);
01905         }
01906     }
01907
01908     public void StartAllServers(bool debug)
01909     {
01910         foreach (Webradio webradio in this.Webradios)
01911         {
01912             if (webradio.Server != null)
01913                 webradio.Server.Start(debug);
01914         }
01915     }
01916
01917     public void StopAllServers()
01918     {
01919         foreach (Webradio webradio in this.Webradios)
01920         {
01921             if (webradio.Server != null)
01922                 webradio.Server.Stop();
01923         }
01924     }
01925
01926     public void StartAllServers()
01927     {
01928         foreach (Webradio webradio in this.Webradios)
01929         {
01930             if (webradio.Server != null)
01931                 webradio.Server.Start();
01932         }
01933     }
01934
01935     public void StopAllServers(bool debug)
01936     {
01937         foreach (Webradio webradio in this.Webradios)
01938         {
01939             if (webradio.Server != null)
01940                 webradio.Server.Stop(debug);
01941         }
01942     }
01943
01944     public void StartAllServers(bool debug)
01945     {
01946         foreach (Webradio webradio in this.Webradios)
01947         {
01948             if (webradio.Server != null)
01949                 webradio.Server.Start(debug);
01950         }
01951     }
01952
01953     public void StopAllServers()
01954     {
01955         foreach (Webradio webradio in this.Webradios)
01956         {
01957             if (webradio.Server != null)
01958                 webradio.Server.Stop();
01959         }
01960     }
01961
01962     public void StartAllServers()
01963     {
01964         foreach (Webradio webradio in this.Webradios)
01965         {
01966             if (webradio.Server != null)
01967                 webradio.Server.Start();
01968         }
01969     }
01970
01971     public void StopAllServers(bool debug)
01972     {
01973         foreach (Webradio webradio in this.Webradios)
01974         {
01975             if (webradio.Server != null)
01976                 webradio.Server.Stop(debug);
01977         }
01978     }
01979
01980     public void StartAllServers(bool debug)
01981     {
01982         foreach (Webradio webradio in this.Webradios)
01983         {
01984             if (webradio.Server != null)
01985                 webradio.Server.Start(debug);
01986         }
01987     }
01988
01989     public void StopAllServers()
01990     {
01991         foreach (Webradio webradio in this.Webradios)
01992         {
01993             if (webradio.Server != null)
01994                 webradio.Server.Stop();
01995         }
01996     }
01997
01998     public void StartAllServers()
01999     {
02000         foreach (Webradio webradio in this.Webradios)
02001         {
02002             if (webradio.Server != null)
02003                 webradio.Server.Start();
02004         }
02005     }
02006
02007     public void StopAllServers(bool debug)
02008     {
02009         foreach (Webradio webradio in this.Webradios)
02010         {
02011             if (webradio.Server != null)
02012                 webradio.Server.Stop(debug);
02013         }
02014     }
02015
02016     public void StartAllServers(bool debug)
02017     {
02018         foreach (Webradio webradio in this.Webradios)
02019         {
02020             if (webradio.Server != null)
02021                 webradio.Server.Start(debug);
02022         }
02023     }
02024
02025     public void StopAllServers()
02026     {
02027         foreach (Webradio webradio in this.Webradios)
02028         {
02029             if (webradio.Server != null)
02030                 webradio.Server.Stop();
02031         }
02032     }
02033
02034     public void StartAllServers()
02035     {
02036         foreach (Webradio webradio in this.Webradios)
02037         {
02038             if (webradio.Server != null)
02039                 webradio.Server.Start();
02040         }
02041     }
02042
02043     public void StopAllServers(bool debug)
02044     {
02045         foreach (Webradio webradio in this.Webradios)
02046         {
02047             if (webradio.Server != null)
02048                 webradio.Server.Stop(debug);
02049         }
02050     }
02051
02052     public void StartAllServers(bool debug)
02053     {
02054         foreach (Webradio webradio in this.Webradios)
02055         {
02056             if (webradio.Server != null)
02057                 webradio.Server.Start(debug);
02058         }
02059     }
02060
02061     public void StopAllServers()
02062     {
02063         foreach (Webradio webradio in this.Webradios)
02064         {
02065             if (webradio.Server != null)
02066                 webradio.Server.Stop();
02067         }
02068     }
02069
02070     public void StartAllServers()
02071     {
02072         foreach (Webradio webradio in this.Webradios)
02073         {
02074             if (webradio.Server != null)
02075                 webradio.Server.Start();
02076         }
02077     }
02078
02079     public void StopAllServers(bool debug)
02080     {
02081         foreach (Webradio webradio in this.Webradios)
02082         {
02083             if (webradio.Server != null)
02084                 webradio.Server.Stop(debug);
02085         }
02086     }
02087
02088     public void StartAllServers(bool debug)
02089     {
02090         foreach (Webradio webradio in this.Webradios)
02091         {
02092             if (webradio.Server != null)
02093                 webradio.Server.Start(debug);
02094         }
02095     }
02096
02097     public void StopAllServers()
02098     {
02099         foreach (Webradio webradio in this.Webradios)
02100         {
02101             if (webradio.Server != null)
02102                 webradio.Server.Stop();
02103         }
02104     }
02105
02106     public void StartAllServers()
02107     {
02108         foreach (Webradio webradio in this.Webradios)
02109         {
02110             if (webradio.Server != null)
02111                 webradio.Server.Start();
02112         }
02113     }
02114
02115     public void StopAllServers(bool debug)
02116     {
02117         foreach (Webradio webradio in this.Webradios)
02118         {
02119             if (webradio.Server != null)
02120                 webradio.Server.Stop(debug);
02121         }
02122     }
02123
02124     public void StartAllServers(bool debug)
02125     {
02126         foreach (Webradio webradio in this.Webradios)
02127         {
02128             if (webradio.Server != null)
02129                 webradio.Server.Start(debug);
02130         }
02131     }
02132
02133     public void StopAllServers()
02134     {
02135         foreach (Webradio webradio in this.Webradios)
02136         {
02137             if (webradio.Server != null)
02138                 webradio.Server.Stop();
02139         }
02140     }
02141
02142     public void StartAllServers()
02143     {
02144         foreach (Webradio webradio in this.Webradios)
02145         {
02146             if (webradio.Server != null)
02147                 webradio.Server.Start();
02148         }
02149     }
02150
02151     public void StopAllServers(bool debug)
02152     {
02153         foreach (Webradio webradio in this.Webradios)
02154         {
02155             if (webradio.Server != null)
02156                 webradio.Server.Stop(debug);
02157         }
02158     }
02159
02160     public void StartAllServers(bool debug)
02161     {
02162         foreach (Webradio webradio in this.Webradios)
02163         {
02164             if (webradio.Server != null)
02165                 webradio.Server.Start(debug);
02166         }
02167     }
02168
02169     public void StopAllServers()
02170     {
02171         foreach (Webradio webradio in this.Webradios)
02172         {
02173             if (webradio.Server != null)
02174                 webradio.Server.Stop();
02175         }
02176     }
02177
02178     public void StartAllServers()
02179     {
02180         foreach (Webradio webradio in this.Webradios)
02181         {
02182             if (webradio.Server != null)
02183                 webradio.Server.Start();
02184         }
02185     }
02186
02187     public void StopAllServers(bool debug)
02188     {
02189         foreach (Webradio webradio in this.Webradios)
02190         {
02191             if (webradio.Server != null)
02192                 webradio.Server.Stop(debug);
02193         }
02194     }
02195
02196     public void StartAllServers(bool debug)
02197     {
02198         foreach (Webradio webradio in this.Webradios)
02199         {
02200             if (webradio.Server != null)
02201                 webradio.Server.Start(debug);
02202         }
02203     }
02204
02205     public void StopAllServers()
02206     {
02207         foreach (Webradio webradio in this.Webradios)
02208         {
02209             if (webradio.Server != null)
02210                 webradio.Server.Stop();
02211         }
02212     }
02213
02214     public void StartAllServers()
02215     {
02216         foreach (Webradio webradio in this.Webradios)
02217         {
02218             if (webradio.Server != null)
02219                 webradio.Server.Start();
02220         }
02221     }
02222
02223     public void StopAllServers(bool debug)
02224     {
02225         foreach (Webradio webradio in this.Webradios)
02226         {
02227             if (webradio.Server != null)
02228                 webradio.Server.Stop(debug);
02229         }
02230     }
02231
02232     public void StartAllServers(bool debug)
02233     {
02234         foreach (Webradio webradio in this.Webradios)
02235         {
02236             if (webradio.Server != null)
02237                 webradio.Server.Start(debug);
02238         }
02239     }
02240
02241     public void StopAllServers()
02242     {
02243         foreach (Webradio webradio in this.Webradios)
02244         {
02245             if (webradio.Server != null)
02246                 webradio.Server.Stop();
02247         }
02248     }
02249
02250     public void StartAllServers()
02251     {
02252         foreach (Webradio webradio in this.Webradios)
02253         {
02254             if (webradio.Server != null)
02255                 webradio.Server.Start();
02256         }
02257     }
02258
02259     public void StopAllServers(bool debug)
02260     {
02261         foreach (Webradio webradio in this.Webradios)
02262         {
02263             if (webradio.Server != null)
02264                 webradio.Server.Stop(debug);
02265         }
02266     }
02267
02268     public void StartAllServers(bool debug)
02269     {
02270         foreach (Webradio webradio in this.Webradios)
02271         {
02272             if (webradio.Server != null)
02273                 webradio.Server.Start(debug);
02274         }
02275     }
02276
02277     public void StopAllServers()
02278     {
02279         foreach (Webradio webradio in this.Webradios)
02280         {
02281             if (webradio.Server != null)
02282                 webradio.Server.Stop();
02283         }
02284     }
02285
02286     public void StartAllServers()
02287     {
02288         foreach (Webradio webradio in this.Webradios)
02289         {
02290             if (webradio.Server != null)
02291                 webradio.Server.Start();
02292         }
02293     }
02294
02295     public void StopAllServers(bool debug)
02296     {
02297         foreach (Webradio webradio in this.Webradios)
02298         {
02299             if (webradio.Server != null)
02300                 webradio.Server.Stop(debug);
02301         }
02302     }
02303
02304     public void StartAllServers(bool debug)
02305     {
02306         foreach (Webradio webradio in this.Webradios)
02307         {
02308             if (webradio.Server != null)
02309                 webradio.Server.Start(debug);
02310         }
02311     }
02312
02313     public void StopAllServers()
02314     {
02315         foreach (Webradio webradio in this.Webradios)
02316         {
02317             if (webradio.Server != null)
02318                 webradio.Server.Stop();
02319         }
02320     }
02321
02322     public void StartAllServers()
02323     {
02324         foreach (Webradio webradio in this.Webradios)
02325         {
02326             if (webradio.Server != null)
02327                 webradio.Server.Start();
02328         }
02329     }
02330
02331     public void StopAllServers(bool debug)
02332     {
02333         foreach (Webradio webradio in this.Webradios)
02334         {
02335             if (webradio.Server != null)
02336                 webradio.Server.Stop(debug);
02337         }
02338     }
02339
02340     public void StartAllServers(bool debug)
02341     {
02342         foreach (Webradio webradio in this.Webradios)
02343         {
02344             if (webradio.Server != null)
02345                 webradio.Server.Start(debug);
02346         }
02347     }
02348
02349     public void StopAllServers()
02350     {
02351         foreach (Webradio webradio in this.Webradios)
02352         {
02353             if (webradio.Server != null)
02354                 webradio.Server.Stop();
02355         }
02356     }
02357
02358     public void StartAllServers()
02359     {
02360         foreach (Webradio webradio in this.Webradios)
02361         {
02362             if (webradio.Server != null)
02363                 webradio.Server.Start();
02364         }
02365     }
02366
02367     public void StopAllServers(bool debug)
02368     {
02369         foreach (Webradio webradio in this.Webradios)
02370         {
02371             if (webradio.Server != null)
02372                 webradio.Server.Stop(debug);
02373         }
02374     }
02375
02376     public void StartAllServers(bool debug)
02377     {
02378         foreach (Webradio webradio in this.Webradios)
02379         {
02380             if (webradio.Server != null)
02381                 webradio.Server.Start(debug);
02382         }
02383     }
02384
02385     public void StopAllServers()
02386     {
02387         foreach (Webradio webradio in this.Webradios)
02388         {
02389             if (webradio.Server != null)
02390                 webradio.Server.Stop();
02391         }
02392     }
02393
02394     public void StartAllServers()
02395     {
02396         foreach (Webradio webradio in this.Webradios)
02397         {
02398             if (webradio.Server != null)
02399                 webradio.Server.Start();
02400         }
02401     }
02402
02403     public void StopAllServers(bool debug)
02404     {
02405         foreach (Webradio webradio in this.Webradios)
02406         {
02407             if (webradio.Server != null)
02408                 webradio.Server.Stop(debug);
02409         }
02410     }
02411
02412     public void StartAllServers(bool debug)
02413     {
02414         foreach (Webradio webradio in this.Webradios)
02415         {
02416             if (webradio.Server != null)
02417                 webradio.Server.Start(debug);
02418         }
02419     }
02420
02421     public void StopAllServers()
02422     {
02423         foreach (Webradio webradio in this.Webradios)
02424         {
02425             if (webradio.Server != null)
02426                 webradio.Server.Stop();
02427         }
02428     }
02429
02430     public void StartAllServers()
02431     {
02432         foreach (Webradio webradio in this.Webradios)
02433         {
02434             if (webradio.Server != null)
02435                 webradio.Server.Start();
02436         }
02437     }
02438
02439     public void StopAllServers(bool debug)
02440     {
02441         foreach (Webradio webradio in this.Webradios)
02442         {
02443             if (webradio.Server != null)
02444                 webradio.Server.Stop(debug);
02445         }
02446     }
02447
02448     public void StartAllServers(bool debug)
02449     {
02450         foreach (Webradio webradio in this.Webradios)
02451         {
02452             if (webradio.Server != null)
02453                 webradio.Server.Start(debug);
02454         }
02455     }
02456
02457     public void StopAllServers()
02458     {
02459         foreach (Webradio webradio in this.Webradios)
02460         {
02461             if (webradio.Server != null)
02462                 webradio.Server.Stop();
02463         }
02464     }
02465
02466     public void StartAllServers()
02467     {
02468         foreach (Webradio webradio in this.Webradios)
02469         {
02470             if (webradio.Server != null)
02471                 webradio.Server.Start();
02472         }
02473     }
02474
02475     public void StopAllServers(bool debug)
02476     {
02477         foreach (Webradio webradio in this.Webradios)
02478         {
02479             if (webradio.Server != null)
02480                 webradio.Server.Stop(debug);
02481         }
02482     }
02483
02484     public void StartAllServers(bool debug)
02485     {
02486         foreach (Webradio webradio in this.Webradios)
02487         {
02488             if (webradio.Server != null)
02489                 webradio.Server.Start(debug);
02490         }
02491     }
02492
02493     public void StopAllServers()
02494     {
02495         foreach (Webradio webradio in this.Webradios)
02496         {
02497             if (webradio.Server != null)
02498                 webradio.Server.Stop();
02499         }
02500     }
02501
02502     public void StartAllServers()
02503     {
02504         foreach (Webradio webradio in this.Webradios)
02505         {
02506             if (webradio.Server != null)
02507                 webradio.Server.Start();
02508         }
02509     }
02510
02511     public void StopAllServers(bool debug)
02512     {
02513         foreach (Webradio webradio in this.Webradios)
02514         {
02515             if (webradio.Server != null)

```

```

01622         this.UpdateObservers(webadioId);
01623         return true;
01624     }
01625     else
01626         return false;
01627 }
01628
01629 public bool StopServer(int webadioId)
01630 {
01631     if (this.Webradios[webadioId].Server.Stop())
01632     {
01633         this.ActiveServers.Remove(this.Webradios[webadioId].Server);
01634         this.UpdateObservers(webadioId);
01635         return true;
01636     }
01637     else
01638         return false;
01639 }
01640
01641 public void ShowServerWebInterface(int webadioId)
01642 {
01643     Process.Start(this.Webradios[webadioId].Server.WebInterfaceUrl);
01644 }
01645
01646 public void ShowServerWebAdmin(int webadioId)
01647 {
01648     Process.Start(this.Webradios[webadioId].Server.WebAdminUrl);
01649 }
01650
01651 public bool TranscoderNextTrack(WebradioTranscoder transcoder)
01652 {
01653     try
01654     {
01655         transcoder.NextTrack();
01656         return true;
01657     }
01658     catch
01659     {
01660         return false;
01661     }
01662 }
01663
01664 public bool ClearHistory(int transcoderId)
01665 {
01666     return this.Bdd.ClearHistory(transcoderId);
01667 }
01668
01669 public bool GenerateHistory(int webadioId, string transcoderName, int transcoderId,
01670     string outputfilename)
01671 {
01672     Document document = new Document(PageSize.A4);
01673     PdfWriter.GetInstance(document, new FileStream(outputfilename, FileMode.Create));
01674     document.Open();
01675     iTextSharp.text.Font fontTitle = FontFactory.GetFont(FontFactory.HELVETICA, 20,
01676     iTextSharp.text.Font.NORMAL);
01677     iTextSharp.text.Font fontText = FontFactory.GetFont(FontFactory.HELVETICA, 12,
01678     iTextSharp.text.Font.NORMAL);
01679     iTextSharp.text.Font fontTextError = FontFactory.GetFont(FontFactory.HELVETICA, 12,
01680     BaseColor.RED);
01681     document.Add(new Paragraph(new Chunk("Webradio's name : " + this.Webradios[webadioId].Name,
01682     fontTitle)));
01683     document.Add(new Paragraph(new Chunk("Transcoder's name : " + transcoderName + "\n\n",
01684     fontTitle)));
01685     foreach (KeyValuePair<string, string> filename in this.Bdd.
01686     GetHistory(transcoderId))
01687     {
01688         string line = "";
01689         AudioFile file = this.GetAudioFileByFilename(filename.Value);
01690         if (file != null)
01691         {
01692             line += "Title : " + file.Title + "\n";
01693             line += "Artist : " + file.Artist + "\n";
01694             line += "Album : " + file.Album + "\n";
01695             line += "Date : " + filename.Key + "\n\n";
01696             document.Add(new Paragraph(new Chunk(line, fontText)));
01697         }
01698         else
01699         {
01700             document.Add(new Paragraph(new Chunk("File not found", fontTextError)));
01701         }
01702     }
01703     document.Close();
01704     return true;
01705 }
01706
01707 public AudioFile GetAudioFileByFilename(string filename)
01708 
```

```

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) &&
01825             this.Bdd.UpdateFilenames(selectedWebradio.Name, newName, selectedWebradio))
01826             {
01827                 foreach (WebradioTranscoder transcoder in selectedWebradio.
01828                     Transcoders)
01829                 {
01830                     transcoder.ConfigFilename = transcoder.ConfigFilename.Replace(oldName, newName);
01831                     transcoder.LogFilename = transcoder.LogFilename.Replace(oldName, newName);
01832                 }
01833                 selectedWebradio.Server.ConfigFilename = selectedWebradio.Server.ConfigFilename.Replace(
01834                     oldName, newName);
01835                 selectedWebradio.Server.LogFilename = selectedWebradio.Server.LogFilename.Replace(oldName,
01836                     newName);
01837                 foreach (Playlist playlist in selectedWebradio.Playlists)
01838                 {
01839                     playlist.Filename = playlist.Filename.Replace(oldName, newName);
01840                 }
01841                 selectedWebradio.Calendar.Filename = selectedWebradio.Calendar.Filename.Replace(oldName,
01842                     newName);
01843                 Directory.Move(DEFAULT_WEBRADIOS_FOLDER + this.Webradios[webradioId].Name,
01844                     DEFAULT_WEBRADIOS_FOLDER + newName);
01845                 selectedWebradio.Name = newName;
01846                 selectedWebradio.GenerateConfigFiles();
01847                 this.UpdateObservers(webradioId);
01848                 return true;
01849             }
01850         else
01851             return false;
01852     }
01866     public bool TranscoderCapture(bool active, string device,
01867         WebradioTranscoder transcoder, int webradioId)
01868     {
01869         try
01870         {
01871             transcoder.SetCaptureMode(active, device);
01872             this.UpdateObservers(webradioId);
01873             return true;
01874         }
01875         catch
01876         {
01877             return false;
01878         }
01879     }
01893     public List<WebradioListener> UpdateServerListeners(int webradioId)
01894     {
01895         return this.Webradios[webradioId].Server.GetListeners();
01896     }
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     }
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

ActiveServers
 WebradioManager::WMModel, 129

ActiveTranscoders
 WebradioManager::WMModel, 129

Ad
 WebradioManager::Ad, 12

Ad.cs, 131

AddAudioFile
 WebradioManager::Bdd, 40

AddEvent
 WebradioManager::Bdd, 40

AddGender
 WebradioManager::Bdd, 41

AddGeneratedPlaylist
 WebradioManager::Bdd, 41

AddObserver
 WebradioManager::WMModel, 110

AddToHistory
 WebradioManager::Bdd, 41

AddToPlaylist
 WebradioManager::AdminController, 15
 WebradioManager::Bdd, 42
 WebradioManager::WMModel, 110

AddTranscoder
 WebradioManager::Bdd, 42

AddWebradio
 WebradioManager::Bdd, 43

AdminController
 WebradioManager::AdminController, 15

AdminController.cs, 132

AdminPassword
 WebradioManager::WebradioServer, 95

AdminPort
 WebradioManager::WebradioTranscoder, 104

AdminView
 WebradioManager::AdminView, 32

AdminView.cs, 135

Album
 WebradioManager::AudioFile, 36

Artist
 WebradioManager::AudioFile, 36

AudioFile
 WebradioManager::AudioFile, 35

AudioFile.cs, 150

AudioFileExist
 WebradioManager::Bdd, 43

AudioFileList
 WebradioManager::Playlist, 70

AudioType.cs, 152

AverageTime
 WebradioManager::WebradioServerStats, 97

Bdd
 WebradioManager::Bdd, 40
 WebradioManager::WMModel, 129

Bdd.cs, 153

Birate
 WebradioManager::WebradioTranscoder, 104

Calendar
 WebradioManager::Webradio, 86

CalendarEvent
 WebradioManager::CalendarEvent, 58, 59

CalendarEvent.cs, 162

CalendarFile
 WebradioManager::WebradioTranscoder, 104

Capture
 WebradioManager::WebradioTranscoder, 104

CheckFolders
 WebradioManager::AdminController, 16
 WebradioManager::WMModel, 110

CheckLibrary
 WebradioManager::AdminController, 16
 WebradioManager::WMModel, 111

ClearDB
 WebradioManager::BddControls, 53

ClearHistory
 WebradioManager::AdminController, 16
 WebradioManager::Bdd, 43
 WebradioManager::WMModel, 111

ClearTable
 WebradioManager::BddControls, 53

ConfigFilename
 WebradioManager::WebradioServer, 95
 WebradioManager::WebradioTranscoder, 104

ConnectionTime
 WebradioManager::WebradioListener, 90

Controller
 WebradioManager::AdminView, 33
 WebradioManager::SelectionView, 80

Controls
 WebradioManager::Bdd, 52

CreateEvent
 WebradioManager::AdminController, 17
 WebradioManager::WMModel, 111

CreatePlaylist
 WebradioManager::AdminController, 17
 WebradioManager::Bdd, 44
 WebradioManager::WMModel, 112

CreateTranscoder
 WebradioManager::AdminController, 17
 WebradioManager::WMModel, 112

CreateWebradio
 WebradioManager::SelectionController, 75
 WebradioManager::WMModel, 113

CurrentListeners
 WebradioManager::WebradioServerStats, 97

CurrentTrack
 WebradioManager::WebradioTranscoder, 105

DayWeek.cs, 165

Delete
 WebradioManager::BddControls, 53

DeleteAudioFile
 WebradioManager::AdminController, 18
 WebradioManager::Bdd, 44
 WebradioManager::WMModel, 113

DeleteEvent
 WebradioManager::AdminController, 18
 WebradioManager::Bdd, 45
 WebradioManager::WMModel, 114

DeletePlaylist
 WebradioManager::AdminController, 19
 WebradioManager::Bdd, 45
 WebradioManager::WMModel, 114

DeleteTranscoder
 WebradioManager::AdminController, 19
 WebradioManager::Bdd, 45
 WebradioManager::WMModel, 114

DeleteWebradio
 WebradioManager::Bdd, 46
 WebradioManager::SelectionController, 75
 WebradioManager::WMModel, 115

Dispose
 WebradioManager::SelectionView, 79

DuplicateWebradio
 WebradioManager::SelectionController, 76
 WebradioManager::WMModel, 115

Duration
 WebradioManager::AudioFile, 36
 WebradioManager::CalendarEvent, 60

EventAppointment
 WebradioManager::EventAppointment, 64

EventAppointment.cs, 166

EventExist
 WebradioManager::Bdd, 46

EventObject
 WebradioManager::EventAppointment, 65

Events
 WebradioManager::WebradioCalendar, 89

EventsCalendar
 WebradioManager::AdminView, 33

ExecuteDataReader
 WebradioManager::BddControls, 55

ExecuteNonQuery
 WebradioManager::BddControls, 55

ExecuteScalar
 WebradioManager::BddControls, 55

File
 WebradioManager::BddControls, 55

Filename
 WebradioManager::AudioFile, 36
 WebradioManager::Playlist, 70
 WebradioManager::WebradioCalendar, 89

FormClose
 WebradioManager::AdminController, 20

Friday
 WebradioManager::DayWeek, 62

Gender
 WebradioManager::AudioFile, 37

GenerateAllConfigs
 WebradioManager::AdminController, 20

GenerateConfigFile
 WebradioManager::Playlist, 69
 WebradioManager::WebradioCalendar, 88
 WebradioManager::WebradioServer, 93
 WebradioManager::WebradioTranscoder, 101

GenerateConfigFiles
 WebradioManager::Webradio, 85
 WebradioManager::WMModel, 116

GenerateHistory
 WebradioManager::AdminController, 20
 WebradioManager::WMModel, 116

GeneratePlaylist
 WebradioManager::AdminController, 21
 WebradioManager::WMModel, 116

GetAudioFileByFilename
 WebradioManager::AdminController, 21
 WebradioManager::WMModel, 117

GetDataTable
 WebradioManager::BddControls, 55

GetGenderId
 WebradioManager::Bdd, 47

GetGenders
 WebradioManager::AdminController, 22
 WebradioManager::Bdd, 47
 WebradioManager::WMModel, 117

GetHistory
 WebradioManager::Bdd, 47

GetLibrary
 WebradioManager::AdminController, 22
 WebradioManager::WMModel, 118

GetListeners
 WebradioManager::WebradioServer, 93

GetPlaylistContent
 WebradioManager::AdminController, 22
 WebradioManager::WMModel, 118

GetSelectedDays
 WebradioManager::CalendarEvent, 59

GetServerListeners
 WebradioManager::AdminController, 23

GetSimilarViewCount
 WebradioManager::AdminController, 23

GetSimiliarViewCount
 WebradioManager::WMModel, 118

Status
 WebradioManager::BddControls, 55

WebradioManager::WebradioTranscoder, 102
GetWebradio
 WebradioManager::AdminController, 23
 WebradioManager::WMModel, 119
GetWebradioByName
 WebradioManager::WMModel, 119
GetWebradios
 WebradioManager::SelectionController, 76
 WebradioManager::WMModel, 119
Hostname
 WebradioManager::WebradioListener, 90
IController.cs, 167
Id
 WebradioManager::AudioFile, 37
 WebradioManager::CalendarEvent, 60
 WebradioManager::Playlist, 70
 WebradioManager::Webradio, 86
 WebradioManager::WebradioCalendar, 89
 WebradioManager::WebradioTranscoder, 105
IdWebradio
 WebradioManager::AdminView, 33
ImportFilesToLibrary
 WebradioManager::AdminController, 24
 WebradioManager::WMModel, 120
Insert
 WebradioManager::BddControls, 55
Ip
 WebradioManager::WebradioTranscoder, 105
IsRunning
 WebradioManager::WebradioServer, 93
 WebradioManager::WebradioTranscoder, 102
Label
 WebradioManager::AudioFile, 37
Library
 WebradioManager::WMModel, 129
LoadLibrary
 WebradioManager::Bdd, 48
 WebradioManager::SelectionController, 76
 WebradioManager::WMModel, 120
LoadWebradios
 WebradioManager::Bdd, 48
 WebradioManager::SelectionController, 77
 WebradioManager::WMModel, 120
LogFilename
 WebradioManager::WebradioServer, 95
 WebradioManager::WebradioTranscoder, 105
Loopatend
 WebradioManager::CalendarEvent, 60
MaxListener
 WebradioManager::WebradioServer, 95
Model
 WebradioManager::AdminController, 31
 WebradioManager::SelectionController, 78
ModifyWebradioName
 WebradioManager::AdminController, 24
 WebradioManager::Bdd, 48
 WebradioManager::WMModel, 121
Monday
 WebradioManager::DayWeek, 62
Music
 WebradioManager::Music, 66, 67
Music.cs, 168
Name
 WebradioManager::CalendarEvent, 60
 WebradioManager::Playlist, 70
 WebradioManager::Webradio, 86
 WebradioManager::WebradioTranscoder, 105
NameWebradio
 WebradioManager::AdminView, 33
NextTrack
 WebradioManager::WebradioTranscoder, 102
Observers
 WebradioManager::WMModel, 129
OpenWebradio
 WebradioManager::SelectionController, 77
Password
 WebradioManager::WebradioServer, 95
 WebradioManager::WebradioTranscoder, 105
PeakListeners
 WebradioManager::WebradioServerStats, 98
Playlist
 WebradioManager::CalendarEvent, 60
 WebradioManager::EventAppointment, 65
 WebradioManager::Playlist, 68, 69
Playlist.cs, 168
PlaylistAd
 WebradioManager::PlaylistAd, 71, 72
PlaylistAd.cs, 170
PlaylistMusic
 WebradioManager::PlaylistMusic, 73
PlaylistMusic.cs, 171
Playlists
 WebradioManager::Webradio, 86
Port
 WebradioManager::WebradioServer, 95
 WebradioManager::WebradioTranscoder, 106
Priority
 WebradioManager::CalendarEvent, 60
Process
 WebradioManager::WebradioServer, 96
 WebradioManager::WebradioTranscoder, 106
ProcessWatcher
 WebradioManager::WMModel, 129
RemoveFromPlaylist
 WebradioManager::AdminController, 25
 WebradioManager::Bdd, 49
 WebradioManager::WMModel, 121
RemoveObserver
 WebradioManager::WMModel, 122
Repeat

WebradioManager::CalendarEvent, 61
SampleRate
 WebradioManager::WebradioTranscoder, 106
Saturday
 WebradioManager::DayWeek, 62
SelectionController
 WebradioManager::SelectionController, 75
SelectionController.cs, 171
SelectionView
 WebradioManager::SelectionView, 79
SelectionView.cs, 173
Server
 WebradioManager::Webradio, 86
SetCaptureMode
 WebradioManager::WebradioTranscoder, 102
ShowServerWebAdmin
 WebradioManager::AdminController, 25
 WebradioManager::WMModel, 122
ShowServerWebInterface
 WebradioManager::AdminController, 25
 WebradioManager::WMModel, 122
Shuffle
 WebradioManager::CalendarEvent, 61
Start
 WebradioManager::WebradioServer, 93
 WebradioManager::WebradioTranscoder, 103
StartServer
 WebradioManager::AdminController, 26
 WebradioManager::WMModel, 123
StartTime
 WebradioManager::CalendarEvent, 61
StartTranscoder
 WebradioManager::AdminController, 26
 WebradioManager::WMModel, 123
Stop
 WebradioManager::WebradioServer, 94
 WebradioManager::WebradioTranscoder, 103
StopAllProcess
 WebradioManager::SelectionController, 77
 WebradioManager::WMModel, 123, 124
StopAllTranscoders
 WebradioManager::AdminController, 27
StopServer
 WebradioManager::AdminController, 27
 WebradioManager::WMModel, 124
StopTranscoder
 WebradioManager::AdminController, 27
 WebradioManager::WMModel, 125
StreamType
 WebradioManager::WebradioTranscoder, 106
StreamType.cs, 175
Sunday
 WebradioManager::DayWeek, 63
Thursday
 WebradioManager::DayWeek, 63
Title
 WebradioManager::AudioFile, 37
ToString
 WebradioManager::Playlist, 69
 WebradioManager::Webradio, 85
 WebradioManager::WebradioTranscoder, 103
TranscoderAacPlus
 WebradioManager::TranscoderAacPlus, 81
TranscoderAacPlus.cs, 175
TranscoderCapture
 WebradioManager::AdminController, 28
 WebradioManager::WMModel, 125
TranscoderExist
 WebradioManager::Bdd, 49
TranscoderMp3
 WebradioManager::TranscoderMp3, 83
TranscoderMp3.cs, 176
TranscoderNextTrack
 WebradioManager::AdminController, 28
 WebradioManager::WMModel, 125
Transcoders
 WebradioManager::Webradio, 87
Tuesday
 WebradioManager::DayWeek, 63
Type
 WebradioManager::AudioFile, 37
 WebradioManager::Playlist, 70
Uid
 WebradioManager::WebradioListener, 90
UniqueListeners
 WebradioManager::WebradioServerStats, 98
Update
 WebradioManager::BddControls, 57
UpdateAudioFile
 WebradioManager::AdminController, 29
 WebradioManager::Bdd, 50
 WebradioManager::WMModel, 126
UpdateEvent
 WebradioManager::AdminController, 29
 WebradioManager::Bdd, 50
 WebradioManager::WMModel, 126
UpdateFilenames
 WebradioManager::Bdd, 50
UpdateServer
 WebradioManager::AdminController, 29
 WebradioManager::Bdd, 51
 WebradioManager::WMModel, 127
UpdateServerListeners
 WebradioManager::WMModel, 127
UpdateServerStats
 WebradioManager::AdminController, 30
 WebradioManager::WMModel, 128
UpdateStats
 WebradioManager::WebradioServer, 94
UpdateTranscoder
 WebradioManager::AdminController, 30
 WebradioManager::Bdd, 51
 WebradioManager::WMModel, 128
UpdateView
 WebradioManager::AdminController, 31

WebradioManager::AdminView, 33
 WebradioManager::SelectionController, 77
 WebradioManager::SelectionView, 80
Url
 WebradioManager::WebradioTranscoder, 106
Useragent
 WebradioManager::WebradioListener, 91
View
 WebradioManager::AdminController, 31
 WebradioManager::SelectionController, 78
WMMModel
 WebradioManager::WMMModel, 110
WMMModel.cs, 191
WebAdminUrl
 WebradioManager::WebradioServer, 96
WebInterfaceUrl
 WebradioManager::WebradioServer, 96
Webradio
 WebradioManager::Webradio, 85
Webradio.cs, 177
WebradioCalendar
 WebradioManager::WebradioCalendar, 88
WebradioCalendar.cs, 178
WebradioExist
 WebradioManager::Bdd, 52
WebradioListener
 WebradioManager::WebradioListener, 90
WebradioListener.cs, 180
WebradioManager, 9
WebradioManager.Ad, 11
WebradioManager.AdminController, 13
WebradioManager.AdminView, 32
WebradioManager.AudioFile, 34
WebradioManager.Bdd, 38
WebradioManager.BddControls, 53
WebradioManager.CalendarEvent, 57
WebradioManager.DayWeek, 61
WebradioManager.EventAppointment, 64
WebradioManager.IContainer, 65
WebradioManager.Music, 66
WebradioManager.Playlist, 67
WebradioManager.PlaylistAd, 71
WebradioManager.PlaylistMusic, 72
WebradioManager.SelectionController, 74
WebradioManager.SelectionView, 78
WebradioManager.TranscoderAacPlus, 80
WebradioManager.TranscoderMp3, 82
WebradioManager.WMMModel, 107
WebradioManager.Webradio, 84
WebradioManager.WebradioCalendar, 87
WebradioManager.WebradioListener, 89
WebradioManager.WebradioServer, 91
WebradioManager.WebradioServerStats, 96
WebradioManager.WebradioTranscoder, 98
WebradioManager::Ad
 Ad, 12
WebradioManager::AdminController
AddToPlaylist, 15
AdminController, 15
CheckFolders, 16
CheckLibrary, 16
ClearHistory, 16
CreateEvent, 17
CreatePlaylist, 17
CreateTranscoder, 17
DeleteAudioFile, 18
DeleteEvent, 18
DeletePlaylist, 19
DeleteTranscoder, 19
FormClose, 20
GenerateAllConfigs, 20
GenerateHistory, 20
GeneratePlaylist, 21
GetAudioFileByFilename, 21
GetGenders, 22
GetLibrary, 22
GetPlaylistContent, 22
GetServerListeners, 23
GetSimilarViewCount, 23
GetWebradio, 23
ImportFilesToLibrary, 24
Model, 31
ModifyWebradioName, 24
RemoveFromPlaylist, 25
ShowServerWebAdmin, 25
ShowServerWebInterface, 25
StartServer, 26
StartTranscoder, 26
StopAllTranscoders, 27
StopServer, 27
StopTranscoder, 27
TranscoderCapture, 28
TranscoderNextTrack, 28
UpdateAudioFile, 29
UpdateEvent, 29
UpdateServer, 29
UpdateServerStats, 30
UpdateTranscoder, 30
UpdateView, 31
View, 31
WebradioManager::AdminView
 AdminView, 32
 Controller, 33
 EventsCalendar, 33
 IdWebradio, 33
 NameWebradio, 33
 UpdateView, 33
WebradioManager::AudioFile
 Album, 36
 Artist, 36
 AudioFile, 35
 Duration, 36
 Filename, 36
 Gender, 37
 Id, 37

Label, 37
Title, 37
Type, 37
Year, 37
WebradioManager::Bdd
 AddAudioFile, 40
 AddEvent, 40
 AddGender, 41
 AddGeneratedPlaylist, 41
 AddToHistory, 41
 AddToPlaylist, 42
 AddTranscoder, 42
 AddWebradio, 43
 AudioFileExist, 43
 Bdd, 40
 ClearHistory, 43
 Controls, 52
 CreatePlaylist, 44
 DeleteAudioFile, 44
 DeleteEvent, 45
 DeletePlaylist, 45
 DeleteTranscoder, 45
 DeleteWebradio, 46
 EventExist, 46
 GetGenderId, 47
 GetGenders, 47
 GetHistory, 47
 LoadLibrary, 48
 LoadWebradios, 48
 ModifyWebradioName, 48
 RemoveFromPlaylist, 49
 TranscoderExist, 49
 UpdateAudioFile, 50
 UpdateEvent, 50
 UpdateFilenames, 50
 UpdateServer, 51
 UpdateTranscoder, 51
 WebradioExist, 52
WebradioManager::BddControls
 ClearDB, 53
 ClearTable, 53
 Delete, 53
 ExecuteDataReader, 55
 ExecuteNonQuery, 55
 ExecuteScalar, 55
 GetDataTable, 55
 Insert, 55
 Update, 57
WebradioManager::CalendarEvent
 CalendarEvent, 58, 59
 Duration, 60
 GetSelectedDays, 59
 Id, 60
 Loopatend, 60
 Name, 60
 Playlist, 60
 Priority, 60
 Repeat, 61
 Shuffle, 61
 StartTime, 61
WebradioManager::DayWeek
 Friday, 62
 Monday, 62
 Saturday, 62
 Sunday, 63
 Thursday, 63
 Tuesday, 63
 Wednesday, 63
WebradioManager::EventAppointment
 EventAppointment, 64
 EventObject, 65
 Playlist, 65
WebradioManager::Music
 Music, 66, 67
WebradioManager::Playlist
 AudioFileList, 70
 Filename, 70
 GenerateConfigFile, 69
 Id, 70
 Name, 70
 Playlist, 68, 69
 ToString, 69
 Type, 70
WebradioManager::PlaylistAd
 PlaylistAd, 71, 72
WebradioManager::PlaylistMusic
 PlaylistMusic, 73
WebradioManager::SelectionController
 CreateWebradio, 75
 DeleteWebradio, 75
 DuplicateWebradio, 76
 GetWebradios, 76
 LoadLibrary, 76
 LoadWebradios, 77
 Model, 78
 OpenWebradio, 77
 SelectionController, 75
 StopAllProcess, 77
 UpdateView, 77
 View, 78
WebradioManager::SelectionView
 Controller, 80
 Dispose, 79
 SelectionView, 79
 UpdateView, 80
WebradioManager::TranscoderAacPlus
 TranscoderAacPlus, 81
WebradioManager::TranscoderMp3
 TranscoderMp3, 83
WebradioManager::WMModel
 ActiveServers, 129
 ActiveTranscoders, 129
 AddObserver, 110
 AddToPlaylist, 110
 Bdd, 129
 CheckFolders, 110

CheckLibrary, 111
 ClearHistory, 111
 CreateEvent, 111
 CreatePlaylist, 112
 CreateTranscoder, 112
 CreateWebradio, 113
 DeleteAudioFile, 113
 DeleteEvent, 114
 DeletePlaylist, 114
 DeleteTranscoder, 114
 DeleteWebradio, 115
 DuplicateWebradio, 115
 GenerateConfigFiles, 116
 GenerateHistory, 116
 GeneratePlaylist, 116
 GetAudioFileByFilename, 117
 GetGenders, 117
 GetLibrary, 118
 GetPlaylistContent, 118
 GetSimiliarViewCount, 118
 GetWebradio, 119
 GetWebradioByName, 119
 GetWebradios, 119
 ImportFilesToLibrary, 120
 Library, 129
 LoadLibrary, 120
 LoadWebradios, 120
 ModifyWebradioName, 121
 Observers, 129
 ProcessWatcher, 129
 RemoveFromPlaylist, 121
 RemoveObserver, 122
 ShowServerWebAdmin, 122
 ShowServerWebInterface, 122
 StartServer, 123
 StartTranscoder, 123
 StopAllProcess, 123, 124
 StopServer, 124
 StopTranscoder, 125
 TranscoderCapture, 125
 TranscoderNextTrack, 125
 UpdateAudioFile, 126
 UpdateEvent, 126
 UpdateServer, 127
 UpdateServerListeners, 127
 UpdateServerStats, 128
 UpdateTranscoder, 128
 WMModel, 110
 Webradios, 129
WebradioManager::Webradio
 Calendar, 86
 GenerateConfigFiles, 85
 Id, 86
 Name, 86
 Playlists, 86
 Server, 86
 ToString, 85
 Transcoders, 87
Webradio, 85
WebradioManager::WebradioCalendar
 Events, 89
 Filename, 89
 GenerateConfigFile, 88
 Id, 89
 WebradioCalendar, 88
WebradioManager::WebradioListener
 ConnectionTime, 90
 Hostname, 90
 Uid, 90
 Useragent, 91
 WebradioListener, 90
WebradioManager::WebradioServer
 AdminPassword, 95
 ConfigFilename, 95
 GenerateConfigFile, 93
 GetListeners, 93
 IsRunning, 93
 LogFilename, 95
 MaxListener, 95
 Password, 95
 Port, 95
 Process, 96
 Start, 93
 Stop, 94
 UpdateStats, 94
 WebAdminUrl, 96
 WebInterfaceUrl, 96
 WebradioServer, 92
WebradioManager::WebradioServerStats
 AverageTime, 97
 CurrentListeners, 97
 PeakListeners, 98
 UniqueListeners, 98
 WebradioServerStats, 97
WebradioManager::WebradioTranscoder
 AdminPort, 104
 Birate, 104
 CalendarFile, 104
 Capture, 104
 ConfigFilename, 104
 CurrentTrack, 105
 GenerateConfigFile, 101
 GetStatus, 102
 Id, 105
 Ip, 105
 IsRunning, 102
 LogFilename, 105
 Name, 105
 NextTrack, 102
 Password, 105
 Port, 106
 Process, 106
 SampleRate, 106
 SetCaptureMode, 102
 Start, 103
 Stop, 103

StreamType, [106](#)
ToString, [103](#)
Url, [106](#)
WebradioTranscoder, [100, 101](#)
WebradioServer
 WebradioManager::WebradioServer, [92](#)
WebradioServer.cs, [181](#)
WebradioServerStats
 WebradioManager::WebradioServerStats, [97](#)
WebradioServerStats.cs, [185](#)
WebradioTranscoder
 WebradioManager::WebradioTranscoder, [100, 101](#)
WebradioTranscoder.cs, [186](#)
Webradios
 WebradioManager::WMModel, [129](#)
Wednesday
 WebradioManager::DayWeek, [63](#)
Year
 WebradioManager::AudioFile, [37](#)