## Fonction boucle

Activation de la fonction boucle à l’aide du bouton : 

**Classe AAP**

Lors de l’évènement click sur ce bouton, la variable « isLoop » passe à « True » est permet donc de boucler dans un morceau de cette manière (ligne 970 à 991) :

/\*\*

\* Thread qui synchronise la musique et la barre de musique + boucle

\*/

**private** Runnable progressUpdater = **new** Runnable() {

@Override

**public** **void** run() {

**if**(*isPlay* && !onTouchSeekBarMusic)

{

seekBar\_Music.setProgress(*mPlayer*.getCurrentPosition());

((TextView)findViewById(R.id.*position*)).setText(heureToString(*mPlayer*.getCurrentPosition()));

**if**(isLoop)//Si la fonction boucle est activé

{

// si la progression de la musique a dépassé la fin de la boucle

**if**(*mPlayer*.getCurrentPosition() >= seekBar\_fin.getProgress())

{

// on replace la lecture au début de la boucle

*mPlayer*.seekTo(seekBar\_debut.getProgress());

}

}

mHandler.postDelayed(**this**, 1000);

}

}

};

## Sauvegarde d’une boucle

Sauvegarde de la boucle courante en appuyant sur le bouton : 

Ensuite l’utilisateur rentre un nom de boucle et en validant, cette nouvelle boucle va être sauvegardée dans le fichier XML **loopConfg.xml à la racine de la carte SD**.

Appel de la méthode **public** **static** **void** addLoop(Loop loop) ligne 594 (AAP .java).

Structure du fichier XML :



* <chanson titre= « when\_the\_rain »> correspond à la chanson titrée « when\_the\_rain »
* <loop> correspond à la boucle
* <nom> correspond au nom de la boucle
* <debut> correspond au début de la boucle en milliseconde
* <fin> correspond à la fin de la boucle en milliseconde

**Classe ExtractLoopConfig**

La méthode **public** **static** **void** addLoop(Loop loop) se charge de parcourir le fichier XML pour sauvegarder la boucle « loop ».

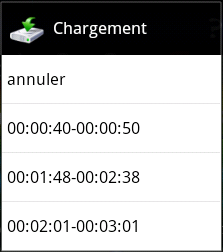
Premièrement, la méthode vérifie si le nœud <chanson> correspondant a déjà était créé.

Si oui, la méthode rajoute un nœud <loop> au nœud <chanson> correspondant.

Si non, la méthode crée un nœud <chanson> correspondant à la chanson de la boucle, et crée à l’intérieur de ce nœud <chanson> un nœud <loop>.

## Chargement d’un boucle

Chargement de la boucle en appuyant sur le bouton : 

Ensuite l’utilisateur choisit une boucle déjà enregistrée pour la chanson en cours de lecture.

Appel de la méthode **public** **static** ArrayList<Loop> getLoops(String titre) ligne 549 (AAP.java).

Puis appel de la méthode **public** **static** Loop getLoop(String nomLoop, String titre) ligne 566 (AAP.java) pour récupérer dans le fichier XML la boucle choisit dans la boite de dialogue ci-dessus.

**Classe ExtractLoopConfig**

La méthode **public** **static** ArrayList<Loop> getLoops(String titre) se charge de chercher les boucles de la chanson en cours de lecture pour remplir la boite de dialogue ci-dessus

La méthode **public** **static** Loop getLoop(String nomLoop, String titre) se charge de récupérer la boucle choisit dans la boite de dialogue.