

Original: [How to take pictures, record videos and play music using MMAPI](#) Também publicado em: [Java Me effort](#)

A **Mobile Media API** (MMAPI) (JSR 135) é uma poderosa e flexível API que permite-nos renderizar e capturar dados de áudio e vídeo. Esta simples API possibilita aos programadores de aplicações Java ME ganhar acesso aos serviços nativos de multimídia disponíveis em um determinado dispositivo. Este artigo descreve como usar a MMAPI para tirar fotos, gravar vídeos e executar áudio. Para mais informações você pode acessar [aqui](#).

Contents

- [1 Aviso](#)
- [2 Classe úteis](#)
- [3 Tirando fotos](#)
- [4 Gravando Vídeo](#)
- [5 Tocando uma música](#)

Aviso

Os exemplos abaixo podem não funcionar nos emuladores.

Classe úteis

Neste exemplo, nós iremos usar as classes **Manager** e **Player** do pacote de áudio [javax.microedition.media](#) e conceitos de *thread*.

A classe **Manager** é o ponto de acesso para obter os recursos do sistema tais como *players* para processamento multimídia. Um objeto **Player** é usado para controlar e renderizar mídia de acordo com o *content type* do conteúdo. *Content types* identificam o tipo da mídia. Por exemplo, a seguir são apresentados alguns exemplos de *content types* extraídos de [class Manager](#)

```
* Arquivos de áudio Wave: audio/x-wav
* Arquivos de áudio AU: audio/basic
* Arquivos de áudio MP3: audio/mpeg
* Arquivos MIDI: audio/midi
* Sequência de tons: audio/x-tone-seq
```

Se você quiser saber mais sobre os *content types* clique [aqui](#).

Uma vez que um objeto **Player** tenha sido criado, a aplicação pode requisitar objetos **Control** que encapsulam os comportamentos para controlar a execução, captura e outras ações da mídia.

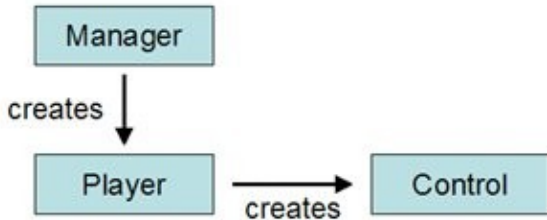
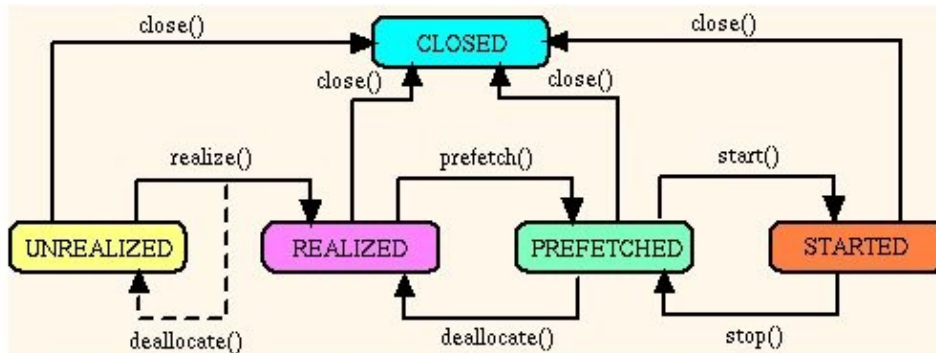


Figura 1. Manager, Player, Control: Relacionamento das classes e interfaces do *framework* usado para gerenciar as mídias usando a MMAPI.

Um objeto **Player** tem cinco estados: UNREALIZED, REALIZED, PREFETCHED, STARTED, e CLOSED. A figura a seguir apresenta os cinco estados e os métodos de transição. Para mais informações sobre os estados dos objetos **Player** clique [aqui](#)

A **Figura 2** apresenta o ciclo de vida de um objeto **Player**



Tirando fotos

Neste exemplo nós tiramos uma foto e a apresentamos em um objeto **Form**. Antes de tirarmos a foto, podemos ver na tela que a câmera está capturando e escolhendo a melhor posição para tirar a foto.

Tirar fotos requer uma instância de **Player**, uma instância de **VideoControl** e uma instância de **Display**. A instância de **VideoControl** é usada para controlar o vídeo proveniente da câmera e exibi-lo em **Canvas/Form**. Primeiramente nós criamos uma instância de **Player** e o colocamos no estado **REALIZE**.

```
// ?capture://image? is used to indicate that we want a image.  
// Se colocarmos outra string aqui, não poderemos utilizar o método getSnapshot()  
// em VideoControl .  
Player mPlayer = Manager.createPlayer("capture://image");  
mPlayer.realize();
```

Como tirar fotos, gravar vídeos, e tocar música usando a API MMAPI

Então nós inicializamos uma instância de **VideoControl** e adicionamos o vídeo ao **Canvas/Form**, fazendo isso o vídeo torna-se visível.

```
VideoControl mVideoControl=(VideoControl) mPlayer.getControl("VideoControl");
Form form = new Form("Camera form");
Item item = (Item) mVideoControl.initDisplayMode(
GUIControl.USE_GUI_PRIMITIVE, null); form.append(item);
```

Nós usamos o método **getSnapshot()** de **VideoControl** para capturar imagens. Este método retorna um array de bytes correspondendo a nossa imagem. Exibir essa imagem requer uma instância de **Image** para adicioná-lo a tela. Para finalizar ajustamos o estado do **Player** para **CLOSE** e setamos para null nossa instância de **VideoControl**.

```
// Obtém a imagem
byte[] raw = mVideoControl.getSnapshot(null);
Image image = Image.createImage(raw, 0, raw.length);

// Adiciona a imagem no Form principal.
if (this.size() > 0 && this.get(0) instanceof StringItem)
    this.delete(0);

append(image);

// Retorna ao Form principal
mDisplay.setCurrent(this);
// Finaliza os objetos 'Player' e 'VideoControl'
mPlayer.close();
mPlayer = null;
mVideoControl = null;
```

Gravando Vídeo

Neste exemplo podemos gravar um vídeo e em seguida executá-lo. De forma semelhante a seção anterior, gravar vídeos requer uma instância de **Player** e uma instância de **VideoControl**. Mas nesse exemplo nós podemos alterar a string dentro do método **createPlayer**. Entretanto, nesse caso nós podemos alterar a string dentro do método **createPlayer**. `?capture://video?`.

```
Player myPlayer = Manager.createPlayer("capture://video");
myPlayer.realize();
videoControl = (VideoControl) myPlayer.getControl("VideoControl");
```

Para exibir o vídeo que será gravado na tela é necessário criar uma instância da classe **VideoCanvas**. Nossa **MIDlet** ainda não está realizando gravação do vídeo que a câmera está capturando.

Em **VideoCanvas** nós adicionamos um comando chamado **cmd_capture**. Quando nós chamamos este comando a **thread RecordCamera** é iniciada. Esta **thread** é responsável por gravar o vídeo capturado pela câmera e armazenar o vídeo em um array de bytes.

```
class RecordCamera extends Thread { // GRAVANDO!!!!....
    RecordControl rc;
    public void run() {
        try {
```

Como tirar fotos, gravar vídeos, e tocar música usando a API MMAPI

```
rc = (RecordControl) myPlayer.getControl("RecordControl");
if (rc == null) {
    return;
}
output = new ByteArrayOutputStream();
rc.setRecordStream(output);
rc.startRecord();

} catch (Exception e) {
    e.printStackTrace();
}
}

//Método invocado quando o usuário finaliza a gravação
public void StopRecord() {
    try {
        if (rc != null) {
            rc.stopRecord();
            rc.commit();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

Para exibir o vídeo após gravado é necessário realizar as seguintes tarefas:

- Recuperar o array de bytes onde o vídeo é armazenado
- Alterar o estado do *player* para **REALIZE**
- Recuperar uma nova instância de **VideoControl** a partir do método **getControl**
- Iniciar a exibição do vídeo a partir do método **start** do objeto *player*

```
ByteArrayInputStream bis = new ByteArrayInputStream(output.toByteArray());
myPlayer = Manager.createPlayer(bis, "video/3gpp");
myPlayer.realize();
videoControl = (VideoControl) myPlayer.getControl("VideoControl");
if (videoCanvas != null) {
    videoCanvas.initControls(videoControl, myPlayer);
    display.setCurrent(videoCanvas);
    myPlayer.start();
}
```

Tocando uma música

Nós iremos agora executar um arquivo mp3 que está armazenado no diretório **res** da aplicação. Para executar um arquivo de áudio é necessário recuperar o arquivo através de um objeto da classe **InputStream** e uma instância de **Player**. Nossa classe terá de implementar a interface **Runnable** de forma que a execução da música possa ser realizada em uma outra *thread*.

A *string* **audio/mpeg** passada como parâmetro no método **createPlayer** indica ao *player* o formato da codificação do arquivo de áudio

```
InputStream in = getClass().getResourceAsStream("/123.mp3");
player = Manager.createPlayer(in, "audio/mpeg");
```

Como tirar fotos, gravar vídeos, e tocar música usando a API MMAPI

```
player.prefetch();  
player.start();
```