How to listen to radio in Java ME

This code example explains the options for accessing radio content with Java ME, and provides a more detailed implementation of how to implement progressive download with Java ME within the context of an online radio application.

17 Mar 2013

Overview

There are a number of approaches for accessing radio content on Series 40 using Java ME:

- Progressive download over HTTP (retrieve media content from network)
- Accessing the radio tuner on the device

At time of writing progressive download is the only validated method for accessing radio content (the radio tuner is inaccessible on most devices at time of writing and RTSP streaming is only supported for video.)

HTTP progressive download

This section shows a HTTP progressive download radioMIDlet implementation.

The media types associated with progressive download must be checked prior to implementing this functionality. This is achieved as follows:

```
String [] list = Manager.getSupportedContentTypes("http");
...
```

It is quite important to find a link to an online radio that streams the media in a supported device format. The example code sample used above was tested by streaming audio mp3 to Nokia Asha 306.

Below is is a simple MIDlet that only has a play command appended to the **Options** Menu (at the top left corner) and a stop Command appended to Action Button 1 (at the top right corner).

```
Command playInputStream = new Command("Over Http InputStream", Command.OK, 2);
Command exitCommand = new Command("Exit", Command.EXIT, 0);
Command stopCommand = new Command("Stop", Command.OK, 0);
```

When the play button is selected from the Options Menu, a thread is started:

```
if(c == playInputStream) {
  thread = new Thread(this);
  thread.start();
}
```

These are the operations that perform the playback inside the thread:

```
HttpConnection connection = (HttpConnection) Connector.open("http://broadcast.infomaniak.ch/radiojazz-high.mp3");
InputStream inputStream = connection.openInputStream();
player = Manager.createPlayer(inputStream, "audio/mp3");
```

As seen above, an `HttpConnection` is opened to the given url and an input stream is read out of it. Then a player instance is created by providing as arguments the input stream opened from the radio's link, and the content type the player is expected to read.

The player is then realized, pre-fetched and started:

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

When stopping the player, we need to remember to de-allocate its instance:

```
if(c == stopCommand) {
            if(player != null) {
                try {

                    player.stop();
                    player.deallocate();
                } catch (MediaException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
```

Finally, in order to enable progressive download, we need to change the MIDlet's Application Descriptor by adding the following JAD attribute:

```
progressive_download: enabled
```


(C) Copyright Nokia 2013. All rights reserved.

Accessing the radio tuner

On devices that have an MMAPI implementation which supports the "radio" capture locator, the following code can be used to listen to a radio in Java ME:

```
Player player = Manager.createPlayer("capture://radio?f=98.3&st=stereo");
// This will tune to 98.3 FM frequency in stereo mode
```

The *Asha software platform 1.0* supports the *radio* capture locator, allowing Java apps to access and control the device AM/FM tuner via the TunerControl interface. Full details about tuner support on the *Asha software platform* are available here: Implementing an AM/FM radio in Java ME.

Warning: This locator is available on a number of older Series 40 phones. Testing shows that it does not work on Series 40 Developer Platform 2.0 devices including Asha 311, Asha 202, and Asha 201. It also does not work on Symbian devices.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.