Bitmap Conversion and Scaling

Purpose

The purpose of this wiki is used to convert a Jpeg Image to a Bitmap. The converted Bitmap is then scaled according to the requirement.

Header Files

```
#include <ImageConversion.h> //CImageDecoder
#include <Fbs.h> //CFbsBitmap
#include <BitmapTransforms.h> //CBitmapScaler
```

Link against

```
LIBRARY mgfetch.lib
LIBRARY bafl.lib
LIBRARY fbscli.lib
LIBRARY efsrv.lib
LIBRARY imageconversion.lib
LIBRARY bitmaptransforms.lib
```

Capabilities

```
capability ReadUserData // MgFetch::RunL()
```

Example code

As the API's for conversion and scaling are Asynchronous requests,
and the requirement is for making this Synchronous as scaling should be started only after conversion is done,
we make use of the "CActiveSchedulerWait" class which makes it possible.

The following code explains its usage.

```
CActiveSchedulerWait* iScheduler = new (ELeave) CActiveSchedulerWait;
//Starts a new wait loop under the control of the current active scheduler.
iScheduler->Start();
//within RunL() call the following code once the request is accomplished
// by checking 'iStatus'
if(iStatus==KErrNone)
{
//Stops the scheduling loop owned by this object
iActiveSchedulerWait.AsyncStop();
}
```

The following code is used to convert a Jpeg Image to Bitmap.

"CImageDecoder" class has a API "Convert()" will start decoding an image frame asynchronously.

As its an Asynchronous call, we must derive our class from "CActive" class and call the Convert() API.

Initializing and Issuing Request code

```
//File session for the imagedecoder
//Open a connection to the File Server
RFs iFileSession;
iFileSession.Connect();
//Imagedecoder object
CImageDecoder* iDecoder;

//We pass the FileSession instance and the name of the file to be decoded
// It creates a decoder for the image in the named file.
iDecoder=CImageDecoder::FileNewL(iFileSession,iImageFolder);

//Retrieve the frame info for a specified frame of the image.
TFrameInfo bmpInfo=iDecoder->FrameInfo();

//CFbsBitmap object
CFbsBitmap* iBitmap;
iBitmap=new(ELeave) CFbsBitmap;
```


(C) Copyright Nokia 2013. All rights reserved.

```
//Creates a bitmap with the specified size and display mode.
iBitmap->Create(bmpInfo.iOverallSizeInPixels,bmpInfo.iFrameDisplayMode);//EGray256

//start conversion to bitmap
iDecoder->Convert(&iStatus,*iBitmap);
SetActive();
```

code to be handled after Issuing Request

```
if(iStatus==KErrNone)
{
     if(iDecoder)
     {
        delete iDecoder;
        iDecoder = NULL;
     }
}
switch( iState )
{
     case EDecoding:
     {
          if( iStatus == KErrNone )
          {
              iState = EIdle;
              iActiveSchedulerWait.AsyncStop();
              break;
          }
          else if(iStatus == KErrUnderflow )
          {
              iDecoder->ContinueConvert( &iStatus );
              SetActive();
              break;
          }
          else if ( iStatus == KErrCorrupt )
          {
             iState = EIdle;
             //Inform about the error
             break;
          }
          else
          {
             // Unknown error
             //Inform about the error...
             break;
          }
     }
     default:
        break;
}
```

The following code is used to Scale the Bitmap

`CBitmapScaler` class has a API `Scale()` which is used for Scaling the Bitmap.

Its an Asynchronous call, so we must derive our class from `CActive` class.

Initializing and Issuing Request code

```
//CFbsBitmap object
CFbsBitmap* iSrcBitmap;
//CBitmapScaler object
CBitmapScaler* iBmpScale;
iBmpScale = CBitmapScaler::NewL();
//This begins the bitmap re-scaling operation.
//iSrcBitmap, The bitmap to be re-scaled.
//This reference is also the target location for the re-scaled bitmap.
//TSize(100,100), the requested target size for the re-scaled bitmap.
iBmpScale->Scale(&iStatus, *iSrcBitmap, TSize(100,100));
SetActive();
//Finally the iSrcBitmap contains the scaled Bitmap.
```

Example project

- BitmapScale.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.