Reporting in Delphi threads

Sometimes we need create many reports in multiple threads simultaneously. This may be the development of a web service, or output information from an existing multithreaded application in a certain needed document format (PDF as example).

I noted that FastReport VCL library is a better solution for document generation in multiple formats. This component library is easy to use and has a convenient report designer which allows you to easily connect to different data sources, among which may be internal application data - arrays, sets of parameters, etc.

Traditional use of FastReport does not give any difficulties, but now we have to use this report generator in a multithreaded application. The output file format will be PDF.

TfrxReport class has a description of several properties which need to be set immediately after the creation of the object.

You need to remember that the object must work in a thread without the creation of dialogs, progress bars and other visual information.

Here is an example of creating and setting an object of class TfrxReport before the execution of the report:

```
// create report
FReport := TfrxReport.Create(nil);
// disable all messages
FReport.EngineOptions.SilentMode := True;
// enable safe work in threads
FReport.EngineOptions.EnableThreadSafe := True;
// disable cache
FReport.EngineOptions.UseFileCache := false;
// disable progress bar
FReport.ShowProgress := False;
```

Some reports have integrated dialog forms, and showing them should be banned for obvious reasons. We need to override an event handler TfrxReport.Engine.OnRunDialog by the procedure ShowReportDialog for to any dialogs.

```
// handle all dialogs by ShowReportDialog
FReport.Engine.OnRunDialog := ShowReportDialog;
```

Our procedure will be executed instead of showing each report dialog. We can change the state of any control in a dialog, but we will leave this procedure empty.

```
procedure TTestThread.ShowReportDialog(Page: TfrxDialogPage);
begin
  // empty
end;
```

Then we create an object of TfrxPDFExport and disable the showing of dialog window and progress bar.

```
  PDF := TfrxPDFExport.Create(nil);
  PDF.ShowDialog := False;
  PDF.ShowProgress := False;
```

All operations on the creation and exporting of report objects can be done in the constructor of the thread. The destructor of the thread should look like:

```
destructor TTestThread.Destroy;
begin
  // destroy all created objects
  PDF.Free;
  FReport.Free;
  inherited;
end;
```

Necessary objects are created and configured. Now you can load the report template from a file and run a report in the implementation of the main thread procedure Execute. There, it will also be exported to the desired format.

```
// load report template
FReport.LoadFromFile(FFileName);
// set report variables
FReport.Variables['ThreadID'] := QuotedStr(FId);
// run report
if FReport.PrepareReport then
begin
  // save result in PDF
  PDF.FileName := FOutPath + '\report_'+ FId +
     '_' + FormatDateTime('YYYYYMMDDHHMMSS', Now) + '.pdf';
  FReport.Export(PDF);
end;
```

Try building reports without using RichText objects because by so doing you can get an unstable application.

Do not forget to include ActiveX controls in uses module, and add a call to CoInitialize (nil); in the procedure Execute before creating a report if the report is connected to ADO. Call to CoUninitialize at the end of the thread procedure.

```
// thread function
procedure TTestThread.Execute;
begin
    // initialize COM library in current thread
    CoInitialize(nil);
    try
      // load report template from the file
      FReport.LoadFromFile(FFileName);
      ...
      ...
      ...
    finally
      // Uninitialize COM
      CoUninitialize;
    end;
  end;
end;
```

You can see an attached example with the application which creates 10 reports in multiple threads and write many PDF files.