i-net
Clear Reports
2016

.NET Programming Guide

i-net software

1 Overview 2
2 Installation 3
2.1 Prerequisites 3
2.2 Internet Information Server (IIS) 3
2.2.1 API 3
2.2.2 Proxy 3
2.3 Global Assembly Cache 4
3 Samples 5
3.1 Engine 5
3.2 Cache 6
3.3 Printing 7
3.4 .NET Viewer Example 8

1 Overview

i-net Clear Reports .NET API is a fully-featured reporting solution. With i-net Clear Reports .NET API you have the possibility to embed report rendering and printing into your existing .NET application. This guide gives a few examples on how to do this.

For further information have a look at the API for Clear Reports .NET

Alternatively you can only embedded a viewer in your .NET application which connect to a i-net Clear Reports server.

2 Installation

In order to embed i-net Clear Reports .NET API into your application, you first need to install a custom setup. Select the .NET API in the setup. You can download the setup file from our website.

2.1 Prerequisites

- .NET framework 3.0 or higher
- Windows XP to Windows 10
- Windows 2003 and newer
- 32-bit and 64-bit compatible
- about 250MB free disk space

2.2 Internet Information Server (IIS)

This is for the case that you are using the IIS for your applications. You can invoke the Clear Reports API from within you web appliction. Additionally you could use the Proxy example if you need a proxy.

2.2.1 API

For using the API for your web projects that run on the IIS it is recommended to install i-net Clear Reports in the Global Assembly Cache (GAC). This improves the startup speed and reduces the needed resources of the IIS. It can also eliminate some problems with security restrictions.

2.2.2 Proxy

There is an example proxy that can be used with the IIS. You can find this sample in the subfolder `\documentation\developer\samples\asp.net` of the i-net Clear Reports installation folder. There is a small *.aspx file that forwards the client requests to the i-net Clear Reports service. If you call this script in your aspx application then you can control the report generation with the URL parameters.

2.3 Global Assembly Cache

The setup of i-net Clear Reports Server .NET installs all available assemblies to the subfolder 'dotnet'. Instead of linking the dll files from the installation folder, you can add them to the Global Assembly Cache (GAC). This is done by simply copying the required files into the folder '%SystemRoot%\\assembly'.

Using the Global Assembly Cache has the following advantages:

- The loading from GAC needs less overhead and your application will always load the correct versions
- The assemblies run with full trust by default and therefore will work in every web application

3 Samples

The following samples are a selection of what is possible with your installation of i-net Clear Reports .NET API. They only cover the basics to fulfill the given scenario.

3.1 Engine

This examples shows how to use the Engine to execute an existing report file as PDF and retrieve the page data.

```
// Create an Engine object
Engine engine = new Engine(Engine.EXPORT_PDF);

// Define the report
// Note: The report "mySampleReport.rpt" must exist in the current
// working directory.
engine.setReport("mySampleReport.rpt");

// Now execute the report
engine.execute();

// Return the number of pages.
int pageCount = engine.getPageCount();

// Now iterate through the pages and request the data for
// each page
for (int pageNumber = 1; pageNumber <= pageCount; pageNumber++)
{
    byte[] pageData = engine.getPageData(pageNumber);
    // Now this page can be processed further ...
}
```

You can find more samples for the usage of the Cache API in the Java code samples directory: `<install>\documentation\developer\samples\`.

3.2 Cache

Reports, rendered by i-net Clear Reports, are stored in a cache mechanism which can be administrated in the Configuration Manager. The available cache types depend on the purchased license.

For some output formats like the HTML the server will send the rendering result in multiple steps. Therefore it is necessary to cache the results between different requests. Also, with the cache it is possible to share the rendered results between multiple users. Please have a look at the Cache settings in the Confiugration Manager to check your current settings.

This examples shows how to use the cache to execute an existing report file as PDF and retrieve the page data. For further details on the possible cache features, have a look at the Cache API documentation.

```
// Get the current cache instance according to the
// administrated type
Cache cache = Cache.getCache();

// Define the report and optional properties for rendering.
// Note: The report "mySampleReport.rpt" must exist in the current
// working directory.
Properties reportProperties = new Properties();
reportProperties.put("report", "mySampleReport.rpt");

// Adding the PDF export property as the target format
reportProperties.put("export_fmt", "pdf");

// Create a new cache key to the provided properties
ReportCacheKey cacheKey = cache.getKey(reportProperties);

// Now execute the report and return the number of pages. If the
// report is already cached,
// the method will return immediately.
int pageCount = cache.getPageCountAndWait(cacheKey);

// Now iterate through the pages and request the data for
// each page
for (int pageNumber = 1; pageNumber <= pageCount; pageNumber++)
{
    byte[] pageData = cache.getPageAndWait(cacheKey, pageNumber);
    // Now this page can be processed further ...
}
```

You can find more samples for the usage of the Cache API in the Java code samples directory: `<install>\documentation\developer\samples\`.

3.3 Printing

i-net Clear Reports allows you to embed the printing of a report from your server without showing the viewer window. You need to specify an URL to a running i-net Clear Reports Server with the report parameter set.

This example shows how to create the render data instance, configure the printer and start the printing process. A "RenderData" object is created from an HTTP report URL to a report on a i-net Clear Reports Server. The i-net Clear Reports Server is running on the same machine, listening on port 9000. To print the report directly, the default printer of the system is used. The paper size, orientation and margins are retrieved from the report settings.

```
// The CR-Viewer.dll assembly is required for this sample.
// create RenderData for the requested report
RenderData data = new URLRenderData
                      ("http://localhost:9000/?report=sample.rpt");

HashPrintRequestAttributeSet attributeSet =
                                new HashPrintRequestAttributeSet();
```

```
PrinterJob job = PrinterJob.getPrinterJob();

// use default print service
job.setPrintService(PrintServiceLookup.lookupDefaultPrintService());

// create the viewer print job and start it
PrinterJobProgress progress = new PrinterJobProgress
                                 (null, job, attributeSet, data);
progress.StartProgress();
```

The following code sets additional print attributes like paper format, printable area and orientation. There are more attributes available, than currently used. Each attribute is an instance of the class `javax.print.attribute.PrintRequestAttribute`:

```

// selecting a paper type
attributeSet.add(MediaSizeName.NA_LETTER);

// force form print (ignores margins and resize), should be used
// for form reports
attributeSet.add(new FormPrint());

// Changing the margins of the selected paper
// (to 1 inch on all sides)
MediaSize size = MediaSize.getMediaSizeForName
                                       (MediaSizeName.NA_LETTER);
float width = size.getX(MediaPrintableArea.INCH);
float height = size.getY(MediaPrintableArea.INCH);
MediaPrintableArea area = new MediaPrintableArea(1, 1, width - 2,
                               height - 2, MediaPrintableArea.INCH);

// switching from portrait to landscape
attributeSet.add(OrientationRequested.LANDSCAPE);
```

3.4 .NET Viewer Example

The .NET Viewer shows an example of how you could visualize a report in you .NET application. Internally it is using a Web-control to show the reports. Find this example in the samples folder. This example comes with source files.