Migrating from PDFreactor 8 to PDFreactor 9

UPDATED DEFAULT HTML STYLES

With PDFreactor 9 some of the standard HTML styles have been edited to be more in line with the HTML5 specification and modern browsers. While most changes are cosmetic, some will influence the layouts of existing documents. When these have a negative impact on existing documents and adapting the print styles accordingly is to complex, the following snippets can be used to undo the most significant changes:

Body Margin

In previous versions of PDFreactor the default margins of body elements were 8px, in accordance to the HTML specification. As these default margins are only useful in non-paged environments they were dropped in PDFreactor 9.

To Undo

```
body {
    margin: 8px;
}
```

For this to take effect in multi-page documents the following snippet has to be applied as well.

Box Decoration Break

The default value of the property box-decoration-break is now slice, as per the specification. The margin, padding and border-width values around page break will be treated as 0.

To Undo

```
* {
    box-decoration-break: clone;
}
```

Rounding Mode

The default value of the proprietary property -ro-rounding-mode is now floor. This avoids rare cases where accumulated rounding imprecisions could lead to unexpected layout results, but may lead to different line and page-breaks.

To Undo

```
html {
    -ro-rounding-mode: round;
}
```

First Page Side

The default side of first pages has changed from left to recto (i.e. right, unless the document direction is right-to-left), as per the specification.

To Undo

```
@-ro-preferences {
    first-page-side: left;
}
```

API CHANGES

The package of the `ExceedingContent` class has been changed from

```
com.realobjects.pdfreactor.exceedingcontent
```

to

```
com.realobjects.pdfreactor.contentobserver
```

DEPRECATED API

Java and Java Wrapper

Constructors of inner `Configuration` classes (except for the `KeyValuePair` class) that take multiple arguments have been deprecated and will be removed in a future version. Use the no-argument constructor in conjunction with individual setters instead. An exception is the `KeyValuePair` class whose multi-argument constructor is not deprecated. Setters now return

an instance of the class, thus allowing you to chain multiple subsequent setter calls. The following example demonstrates using no-argument constructors and chainable setters by adding a user script and a user style sheet to a `Configuration` instance:

Old API

```
config.setUserScripts(new ScriptResource(null, "http://myScript.js", false));
config.setUserStyleSheets(new Resource("p { color: red; }", null));
```

New API

```
config
    .setUserScripts(new ScriptResource().setUri("http://myScript.js"))
    .setUserStyleSheets(new Resource().setContent("p { color: red; }"));
```

.NET Wrapper

Multi-argument constructors are now obsolete in favor of object initializers.

Old API

```
config.UserScripts.Add(new ScriptResource(null, "http://myScript.js", false));
config.UserStyleSheets.Add(new Resource("p { color: red; }", null));
```

New API

```
config.UserScripts.Add(new ScriptResource { Uri = "http://myScript.js" });
config.UserStyleSheets.Add(new Resource { Content = "p { color: red; }" });
```

Migrating from PDFreactor 8.0 to PDFreactor 8.1

JAVA AND .NET WRAPPERS

When using either the Java or .NET wrapper APIs, your integration has to be adjusted. These wrapper APIs are now based on the REST API rather than the SOAP API. This was done to provide an API that is more in line with the other wrapper APIs.

PDFREACTOR WEB SERVICE SETTINGS

When using the PDFreactor Web Service with custom settings in the "pdfreactorwebservice.vmoptions" file, you have to migrate these settings to the "start.ini" located in the "PDFreactor/jetty" directory.

If your "pdfreactorwebservice.vmoptions" looked like this

```
-Xmx1024m
-Djava.awt.headless=true
```

you have to add the lines from your "pdfreactorwebservice.vmoptions" to the end of your "start.ini" file

```
--exec
-Dorg.apache.cxf.Logger=org.apache.cxf.common.logging.Slf4jLogger
-Dcom.realobjects.interceptConsoleOutput=true
# old pdfreactorwebservice.vmoptions settings
-Djava.awt.headless=true
-Xmx1024m
```

Migrating from PDFreactor 7- to PDFreactor 8+

With PDFreactor 8 we are introducing the first major API change since PDFreactor 2. One major benefit of this change is that the new Java API is identical (with a few additions) to the newly introduced Java web service client API.

USING THE LEGACY JAVA API

Java integrators can still use the old API, however we highly recommend to migrate to the new API as soon as possible, since the old one will be removed in a future release of PDFreactor. To continue using the old legacy API, just change the package from

```
com.realobjects.pdfreactor
```

to

```
com.realobjects.pdfreactor.legacy
```

MIGRATING TO THE NEW API

The Configuration Object

The most obvious change is the introduction of the `Configuration` object. In previous versions of PDFreactor, you invoked all API methods on the PDFreactor instance, however this had some disadvantages like making the PDFreactor instance non-reusable. In PDFreactor 8, you just have to create one instance of PDFreactor. The instance only has a few API methods, like `convert`.

All settings and options are properties of the configuration and have to be set there. While most methods are the same as in previous versions of PDFreactor, some have changed. For example, instead of add-methods, getters are used to retrieve lists on which new entries can be added. Make sure to consult the API documentation.

Converting

To create a PDF or image, you now just have to call the `convert` method with the configuration as a single parameter, which also specifies the input document. PDFreactor automatically detects if you are converting an HTML string, a URL or binary data.

Retrieving the Result

The new `convert(Configuration)` method no longer returns the PDF as binary directly. It returns a `Result` object which not only contains the PDF as binary, but also other useful data such as the log.

There are additional convert methods, such as `convert(Configuration, OutputStream)` which writes the PDF directly in the specified OutputStream instead of returning it or `convertAsBinary(Configuration)` which returns the binary data directly instead of a Result object. Please make sure to read the API documentation and the PDFreactor manual (Chapter "Integration").

EXAMPLES

Below are simple examples in different programming languages that show how PDFreactor was used previously and how it is used now.

Java

Old API

```
PDFreactor pdfReactor = new PDFreactor();

// simple settings
pdfReactor.setAddBookmarks(true);
pdfReactor.setAddLinks(true);

// adding a user style sheet
pdfReactor.addUserStyleSheet("p { color: red }", null, null, null);

// create PDF and specify the document
byte[] pdf = pdfReactor.renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
PDFreactor pdfReactor = new PDFreactor();
Configuration config = new Configuration();

// specify the document
config.setDocument("http://www.realobjects.com");

// simple settings
config.setAddBookmarks(true);
config.setAddLinks(true);

// adding a user style sheet
config.getUserStyleSheets().add(new Resource("p { color: red }", null));

// create PDF
Result result = pdfReactor.convert(config);
byte[] pdf = result.getDocument();
```

.NET

Old API

```
PDFreactor pdfReactor = new PDFreactor();

// simple settings
pdfReactor.SetAddBookmarks(true);
pdfReactor.SetAddLinks(true);

// adding a user style sheet
pdfReactor.AddUserStyleSheet("p { color: red }", "", "", "");

// create PDF and specify the document
byte[] pdf = pdfReactor.RenderDocumentFromURL("http://www.realobjects.com");
```

New API *(Changed in version 8.1)*

```
PDFreactor pdfReactor = new PDFreactor();
Configuration config = Configuration();

// specify the document
config.Document = "http://www.realobjects.com";

// simple settings
config.AddBookmarks = true;
config.AddLinks = true;

// adding a user style sheet
config.UserStyleSheets = new List<Resource> {new Resource("p { color: red }", "")};

// create PDF
Result result = pdfReactor.Convert(config);
byte[] pdf = result.Document;
```

PHP

Old API

```
$pdfReactor = new PDFreactor();

// simple settings
$pdfReactor->setAddBookmarks(true);
$pdfReactor->setAddLinks(true);

// adding a user style sheet
$pdfReactor->addUserStyleSheet("p { color: red }", "", "", "");

// create PDF and specify the document
$result = $pdfReactor->renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
$pdfReactor = new PDFreactor();
$config = array(
    // specify the document
    "document" => "http://www.realobjects.com",

    // simple settings
    "addBookmarks" => true,
    "addLinks" => true,

    // adding a user style sheet
    "userStyleSheets" => array(
        array(
            "content"=> "p { color: red }"
        )
    )
);

// create PDF
// ...as base64 encoded String
$result = $pdfReactor->convert($config);
$pdf = $result->document;

// ...as binary
$pdf = $pdfReactor->convertAsBinary($config);
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
echo base64_decode($result->document);
```

Python

Old API

```
pdfReactor = PDFreactor()

# simple settings
pdfReactor.setAddBookmarks(True)
pdfReactor.setAddLinks(True)

# adding a user style sheet
pdfReactor.addUserStyleSheet("p { color: red }", "", "", "")

# create PDF and specify the document
result = pdfReactor.renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
pdfReactor = PDFreactor()
config = {
    # specify the document
    'document': "http://www.realobjects.com",

    # simple settings
    'addBookmarks': True,
    'addLinks': True,

    # adding a user style sheet
    'userStyleSheets': [
        {
            'content': "p { color: red }"
        }
    ]
}

# create PDF
# ...as base64 encoded String
result = pdfReactor.convert(config)
pdf = result['document']

# ...as binary
pdf = pdfReactor.convertAsBinary(config)
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
import base64
print(base64.b64decode(result['document']))
```

Ruby

Old API

```
pdfReactor = PDFreactor.new();

# simple settings
pdfReactor.setAddBookmarks(true)
pdfReactor.setAddLinks(true)

# adding a user style sheet
pdfReactor.addUserStyleSheet("p { color: red }", "", "", "")

# create PDF and specify the document
result = pdfReactor.renderDocumentFromURL("http://www.realobjects.com")
```

New API

```
pdfReactor = PDFreactor.new()
config = {
    # specify the document
    document: "http://www.realobjects.com",

    # simple settings
    addBookmarks: true,
    addLinks: true,

    # adding a user style sheet
    userStyleSheets: [
        {
            content: "p { color: red }"
        }
    ]
}

# create PDF
# ...as base64 encoded String
result = pdfReactor.convert(config)
pdf = result["document"]

# ...as binary
pdf = pdfReactor.convertAsBinary(config)
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
require "base64"
print Base64.decode64(result["document"])
```

Perl

Old API

```
my $pdfReactor = PDFreactor -> new();

# simple settings
$pdfReactor -> setAddBookmarks('true');
$pdfReactor -> setAddLinks('true');

# adding a user style sheet
$pdfReactor -> addUserStyleSheet("p { color: red }", "", "", "");

# create PDF and specify the document
$result = pdfReactor -> renderDocumentFromURL("http://www.realobjects.com");
```

New API

```
my $pdfReactor = PDFreactor -> new();
$config = {
    # specify the document
    'document' => "http://www.realobjects.com",

    # simple settings
    'addBookmarks' => 'true',
    'addLinks' => 'true',

    # adding a user style sheet
    'userStyleSheets' => [
        {
            'content' => "p { color: red }"
        }
    ]
};

# create PDF
# ...as base64 encoded String
$result = $pdfReactor -> convert($config);
$pdf = $result->{'document'};

# ...as binary
$pdf = $pdfReactor -> convertAsBinary($config);
```

Note: To convert the base64 encoded document into binary data, you can do the following:

```
use MIME::Base64 ();
print MIME::Base64::decode($result->{'document'});
```