How to create links and bookmarks in PDF

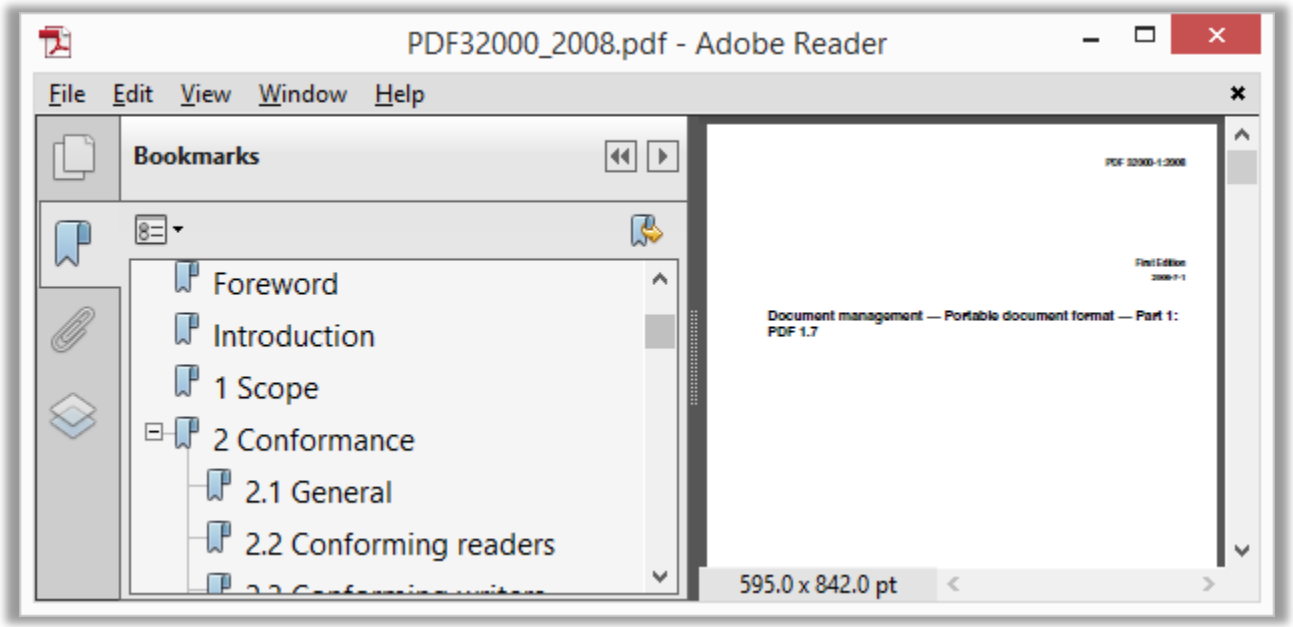
Written by Apitron Documentation Team

1. Links and bookmarks in PDF

While these terms seem familiar, we think there still needs to be a brief intro, describing their particular properties taking into account the PDF basis. Links in PDF are made by link annotations, special kind of “annotation” objects which, in general, can be added to any page being a helpful addition to page content. There are lots of annotation types defined in PDF specification, just take a look at the section *12.5.6 Annotation Types*.

Link annotations - work as “hot” area markers and navigate user to the desired location within the document, or to external destination. Seems similar to HTML, but the key difference is that link annotations are totally independent from the content of PDF document. It’s also possible to assign “actions” to link annotations instead of pure `Destination` objects, e.g. `JavaScriptAction`, `GoToAction`, `SoundAction` etc. Section *12.6.4 Actions Types* of the specification has a complete list. We will focus on simple destinations in this post, leaving the actions for near future.

Bookmarks - can be seen on the left pane of PDF reader if present, see the image below:



Pic. 1 Bookmarks panel

PDF Bookmarks work similar to links, having the same set of features: one can be navigated using `Destination` objects, or the specific action can be triggered by clicking a bookmark. In addition, they can form hierarchical structures showing the outline of the document.

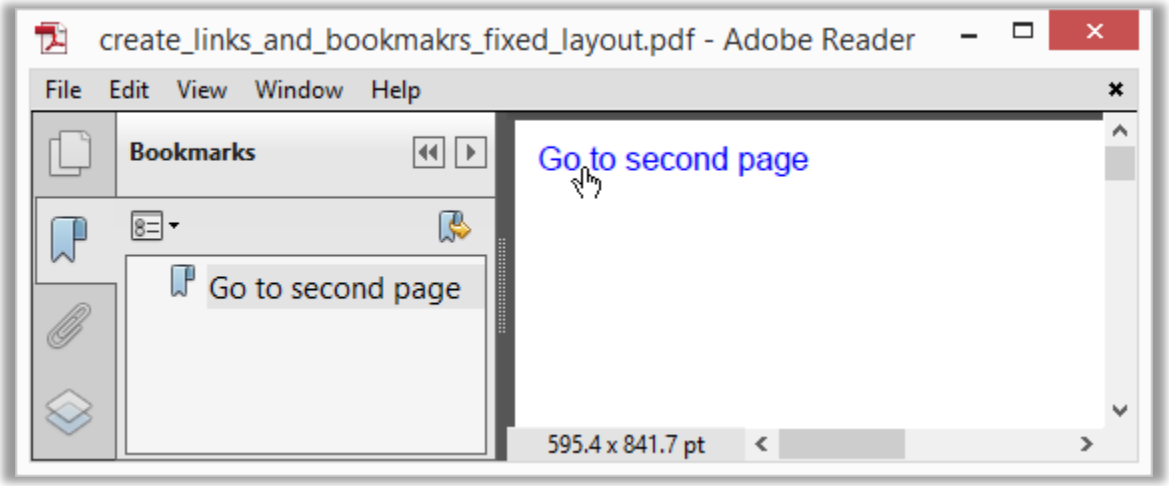
2. Fixed layout approach to navigation

Fixed layout API from Apitron PDF Kit for .NET provides an easy and straightforward way to create links and bookmarks in pdf documents. All necessary objects are described in the pdf specification and have very close mapping to their library counterparts, therefore, one can quickly start creating PDF documents by referring to a standard specification.

See the c# code sample below, it creates a link and bookmark pointing to the second page:

```
public void CreateLinksAndBookmarks()
{
    // create pdf document
    FixedDocument doc = new FixedDocument();
    // add a bookmark for second page
    doc.Bookmarks.AddFirst(new Bookmark(new Destination(1),"Go to second page"));

    // add first page with text link
    Page firstPage = new Page();

    // add link text
    TextObject linkTextObject = new TextObject();
    linkTextObject.AppendTextLine("Go to second page");

    firstPage.Content.SaveGraphicsState();
    firstPage.Content.SetDeviceNonStrokingColor(RgbColors.Blue.Components);
    firstPage.Content.Translate(10, 820);
    firstPage.Content.AppendText(linkTextObject);
    firstPage.Content.RestoreGraphicsState();

    // add link annotation covering the text
    LinkAnnotation link = new LinkAnnotation(new Boundary(10,815,135,835));
    link.BorderStyle = new AnnotationBorderStyle(0);
    link.Destination = new Destination(1);
    firstPage.Annotations.Add(link);

    // add destination page
    Page secondPage = new Page();

    TextObject textObject = new TextObject();
    textObject.AppendTextLine("Second page");

    secondPage.Content.Translate(10,820);
    secondPage.Content.AppendText(textObject);

    // append pages
    doc.Pages.Add(firstPage);
    doc.Pages.Add(secondPage);

    // save document
    using (Stream stream = File.Create("create_links_and_bookmarks_fixed_layout.pdf"))
    {
        doc.Save(stream);
    }
}
```

The resulting document is shown on the image below:



Pic. 2 Creating pdf link and bookmark usign fixed layout API

As it can be seen from this image, the link and bookmark both navigate the user to the second page of the document. We used `Destination` object constructed using page index which set up the target for navigation. Note that links on page are independent from content and are simply “hot” areas which can cover any desired part of the pdf page.

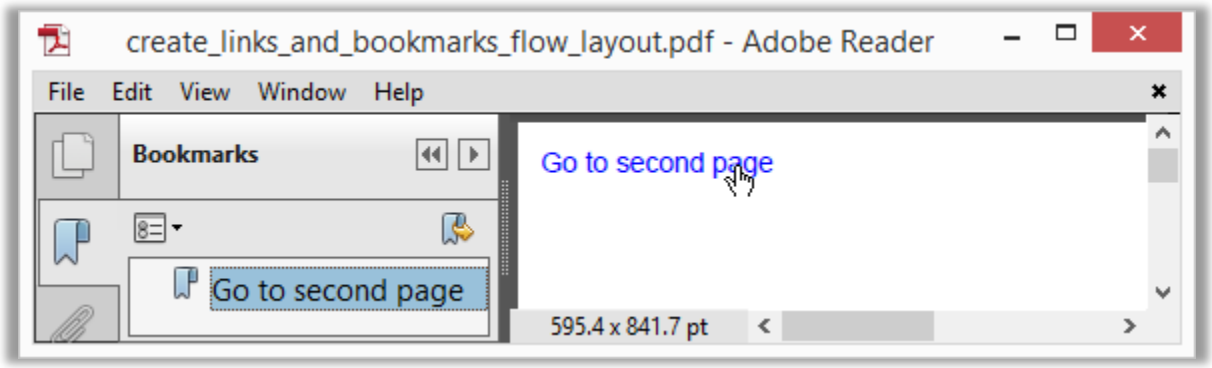
3. Flow layout approach to navigation

If you were to create a flow layout document and add links or bookmarks to it, you would have to use `Link` or `Bookmark` property that each `ContentElement` provides. We use a declarative approach here – each content element indicates whether it is a link, pointing to other element or external resource, or it needs a bookmark pointing to it, or both. The rest is being processed automatically.

Consider the code:

```
public void CreateLinksAndBookmarksFlowLayout()
{
    // create pdf document
    FlowDocument doc = new FlowDocument(){Margin = new Thickness(10)};

    // create first text block and make it a link
    doc.Add(new TextBlock("Go to second page"){ Color = RgbColors.Blue,
        Link = new CrossReference("page2")});
    // force 2nd page creation
    doc.Add(new PageBreak());

    // add second text block, indicate that it needs a bookmark
    doc.Add(new TextBlock("Second page") { Id = "page2",
        Bookmark = new BookmarkEntry("Go to second page") });

    // save document
    using (Stream stream = File.Create("create_links_and_bookmarks_flow_layout.pdf"))
    {
        doc.Write(stream, new ResourceManager());
    }
}
```

And the result:

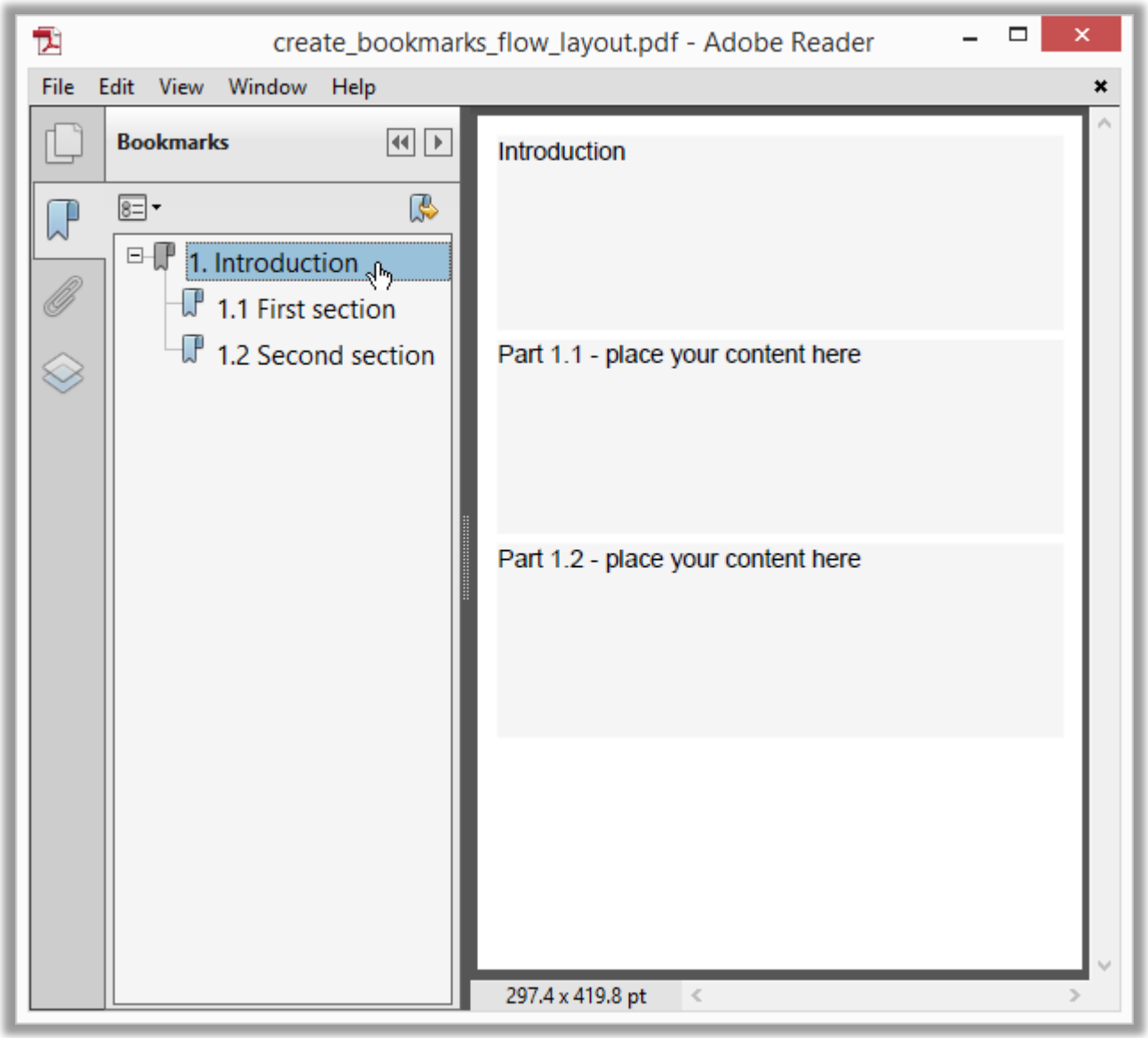


Pic. 3 Create pdf links and bookmarks using flow layout API

It looks the same as document created using fixed layout API. Notice that it took us less than 10 lines of code to produce this PDF document.

3.1 Advanced bookmarking sample

In this example we will create a document outline using flow layout API, you can use this technique to create table of contents or similar structural descriptions. See the image below, it shows the document having a set of bookmarks and each of them corresponds to a content section on PDF page.



Pic. 4 Hierarchical bookmarks

The code generating this document is provided below, it uses a style matched via the type selector for textblock elements styling; other parts are similar to the code from previous example.

Code sample:

```
public void CreateLinksAndBookmarksFlowLayoutAdvanced()
{
    // create pdf document
    FlowDocument doc = new FlowDocument() { Margin = new Thickness(10) };

    // create style matching each textblock, making it small article
    doc.StyleManager.RegisterStyle("Textblock", new Style(){Display = Display.Block,
        Height = 100,
        Background = RgbColors.WhiteSmoke, Margin = new Thickness(0,0,0,5)});

    // create a section with a root bookmark
    Section page1 = new Section(){Bookmark = new BookmarkEntry("1. Introduction")};
    page1.Add(new TextBlock("Introduction"));

    // add textblocks and indicate they need a bookmark
    page1.Add(new TextBlock("Part 1.1 - place your content here")
        { Bookmark = new BookmarkEntry("1.1 First section")});
    page1.Add(new TextBlock("Part 1.2 - place your content here")
        { Bookmark = new BookmarkEntry("1.2 Second section") });

    // add section to the document
    doc.Add(page1);

    // save document
    using (Stream stream = File.Create("create_bookmarks_flow_layout.pdf"))
    {
        doc.Write(stream, new ResourceManager(), new PageBoundary(Boundaries.A6));
    }
}
```

You can see that we used a `Section` object here to create the bookmarks hierarchy. The bookmark assigned to the section gets created as the parent bookmark for bookmarks generated for nested elements.

4. Conclusion

In this post we have shown how to create links and bookmarks in PDF documents. You can read more about PDF processing in our free book available for download here. Future posts will continue to explore various aspects of PDF and our API.

The Apiton PDF Kit for .NET is a powerful and easy to use pdf library available for all modern platforms. Develop applications for Xamarin (iOS, Android, Mac) and cross-platform MONO apps. Create Windows .NET applications targeting desktop, phone or tablet users. It doesn't matter whether you use WinForms, WPF, or develop for Windows Store or Windows Phone Silverlight; it always provides you with an ability to create perfect PDF documents. This library can be used to create web apps or web services based on ASP.NET MVC, WebForms, Azure Web or Worker roles.

Contact us if you have any questions, we're always open for any feedback.