在 WebView 中显示本地 PDF

创建自定义 WebView 控件

通过继承 WebView 的方式创建自定义的 PDFWebView 控件，代码如下所示：

```
using Xamarin.Forms;

namespace XamarinForm.Views
{
    public class PDFWebView : WebView
    {
        protected BindableProperty UrlProperty = BindableProperty.Create("Url",
typeof(string), typeof(PDFWebView), defaultValue: default(string));

        public string Url
        {
            get { return (string)GetValue(UrlProperty); }
            set { SetValue(UrlProperty, value); }
        }
    }
}
```

使用自定义控件

代码如下：

```
using Xamarin.Forms;
using XamarinForm.Views;

namespace XamarinForm.Pages.Control
{
    public class TestPDFWebViewPage:ContentPage
    {
        public TestPDFWebViewPage()
        {
            Content = new StackLayout
            {
                Children = {
                    new PDFWebView {
                        Url= "spec.pdf",
```

```
                        HorizontalOptions = LayoutOptions.FillAndExpand,
                        VerticalOptions = LayoutOptions.FillAndExpand
                    }
                }
            };
        }
    }
}
```

为每个平台创建自定义渲染器

在 IOS 上创建自定义渲染器

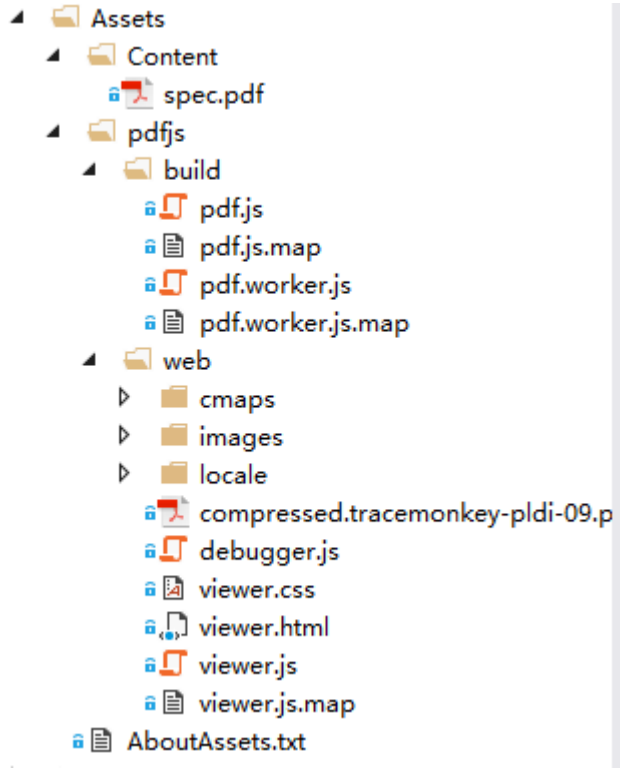
创建 PDFWebViewRenderer 类，继承 ViewRenderer<T1, T2>，并重写 OnElementChanged 方法，代码如下所示：

```
using Foundation;
using System.IO;
using System.Net;
using UIKit;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;
using XamarinForm.iOS.CustomRenderer;
using XamarinForm.Views;

[assembly:ExportRenderer(typeof(PDFWebView),typeof(PDFWebViewRenderer))]
namespace XamarinForm.iOS.CustomRenderer
{
    class PDFWebViewRenderer: ViewRenderer<PDFWebView, UIWebView>
    {
        protected override void OnElementChanged(ElementChangedEventArgs<PDFWebView>
e)
        {
            base.OnElementChanged(e);
            if (Control == null)
            {
                SetNativeControl(new UIWebView());
            }

            if (e.OldElement == null)
            {
```

```
                //Cleanup
            }
            if (e.NewElement != null)
            {
                var pdfWebView = Element as PDFWebView;
                string fileName = Path.Combine(NSBundle.MainBundle.BundlePath,
string.Format("Content/{0}", WebUtility.UrlEncode(pdfWebView.Url)));
                Control.LoadRequest(new NSUrlRequest(new NSUrl(fileName,false)));
                Control.ScalesPageToFit = true;
            }
        }
    }
}
```

在 Android 上创建自定义渲染器

添加 `pdf.js` 到 Android 项目的 `Assets` 文件中，文件结构如下：



然后创建 `PDFWebViewRenderer` 类，继承 `WebViewRenderer`，并重写 `OnElementChanged` 方法，代码如下所示：

```
using System.Net;
using Android.Content;
using XamarinForm.Views;
using XamarinForm.Droid.CustomRenderer;
```

```
using Xamarin.Forms;
using Xamarin.Forms.Platform.Android;

[assembly:ExportRenderer(typeof(PDFWebView), typeof(PDFWebViewRenderer))]
namespace XamarinForm.Droid.CustomRenderer
{
    public class PDFWebViewRenderer : WebViewRenderer
    {
        public PDFWebViewRenderer(Context context) : base(context)
        {
        }

        protected override void OnElementChanged(ElementChangedEventArgs<WebView> e)
        {
            base.OnElementChanged(e);
            var pdfWebView = Element as PDFWebView;
            Control.Settings.AllowUniversalAccessFromFileURLs = true;

Control.LoadUrl(string.Format("file:///android_asset/pdfjs/web/viewer.html?file={0}"
                , string.Format("file:///android_asset/Content/{0}",
WebUtility.UrlEncode(pdfWebView.Url))));
        }
    }
}
```

注意：如果 API 等级低于 19 无法通过`?file=`请求字符串在视图中加载本地 PDF 文档，必须通过修改`Assets/pdfjs/web/viewer.js`中的`DEFAULT_URL`的值来加载本地 PDF 文档。