Foxit


# DEVELOPER GUIDE

## Foxit® WebPDF Viewer

Microsoft® Partner
Gold Independent Software Vendor (ISV)

# Table of Contents

# 1 Foxit WebPDF Viewer 概述

通过 Foxit WebPDF Viewer, 开发人员可以部署和定制 WebPDF 阅读器以实现在网页浏览器上快速显示、注释、填写、签署和管理 PDF 文档。将 WebPDF Viewer 集成到 Web 应用程序中，终端用户可在桌面和移动设备上阅读 PDF 文档，而无需在本地额外下载任何软件。

## 1.1 为什么选择 Foxit WebPDF Viewer

Foxit 是亚马逊投资的领先软件解决方案供应商，专注于 PDF 显示、编辑、创建、管理以及安全方面。WebPDF Viewer 是一款跨平台的在线 PDF 阅读解决方案，其已被全球众多知名公司选择并集成到他们的解决方案中。选择 Foxit WebPDF Viewer 的几大理由：

**灵活定制**
程序开发人员可轻松地设计 WebPDF Viewer 界面所需要的风格，并使其与他们的 Web 应用程序保持一致。

**易于集成**
程序开发人员可轻松地通过创建 REST API 的方式对接 WebPDF Viewer，从而获取和导出 Web 应用程序中的文档。

**集成服务器组件**
安装包中已包含 Web 容器和数据库来支持集群环境的快速部署。

**标准化和一致性的注释数据**

WebPDF Viewer 上的注释数据能在其他标准的 PDF 应用中显示和编辑。

**全面管控文档**
用户无需下载即可打开保存在服务器中的原始文档，且文档拥有者对文档持有全面的控制权。还支持从第三方系统导入用户权限，为不同的用户设置不同的文档权限。

**基于福昕高保真的 PDF 渲染引擎**

Foxit WebPDF Viewer 的核心技术是基于众多知名企业所信赖的福昕 PDF 引擎。福昕强大的 PDF 引擎可快速、稳定地显示文档，不受环境的约束。

另外，用户可以选择购买产品相关的维护服务，福昕专业的技术支持团队将竭诚为您服务。我们的产品将定期地进行更新改进。因此，若您需要开发一款能够管控文档分发的跨平台的 PDF 文档阅读解决方案，选择高性价比的 Foxit WebPDF Viewer 实为明智之举。

## 1.2 文档读者和范畴

该文档主要适用于需要将 Foxit WebPDF Viewer 集成到其 Web 应用程序中的开发人员。它涵盖了用于集成的 REST API 的定义和使用，以及用于自定义的前端 API。在集成开发开始之前，请先参阅部署指南。

## 1.3 你的 Web 应用程序

Foxit WebPDF Viewer 允许 Web 应用程序无缝地显示 PDF，而无需安装任何插件或者本地软件。在使用 WebPDF Viewer 之前，开发人员需要准备一台存放 PDF 的主机服务器。

## 1.4 功能

Foxit WebPDF Viewer 提供了常见的 PDF 功能以及允许开发人员将强大的 PDF 技术集成到其应用程序中，如显示、打印、注释 PDF 文档，文本搜索以及表单填写。

程序开发人员可使用包中的 sample Reader 默认的界面或者自定义界面。

| **功能** | |
|---|---|
| **PDF 显示** | 页面跳转，缩放，适应宽度，书签，缩略图，打印，旋转 |
| **PDF 文本** | 文本选择和复制，文本搜索 |
| **注释** | 显示所有的注释<br>15 种注释工具 (高亮，下划线，波浪线，删除线，备注，钢笔，打字机，注释框，矩形，椭圆，直线，箭头，折线，多边形，图章) |

| | |
|---|---|
| **数字签名** | 为开发人员提供 REST API 定义用来集成第三方数字签名，包括图片和手写签名(JR 引擎不支持) |
| **Acroform 表单** | 填表，导出/导入 PDF 表单数据 |
| **安全** | 支持密码保护，水印 |
| **渲染引擎** | 两种渲染引擎。<br>SR，服务器端渲染，在服务器端将 PDF 文件转换成浏览器可识别的文件(HTML 文件)。<br>JR，JavaScript 渲染，在本地浏览器中使用 JavaScript 渲染 PDF。 |

## 1.5 评估

用户可申请下载 Foxit WebPDF Viewer 的试用版本进行试用评估。试用版除了有 30 天的免费试用以及生成的 PDF 页面上会有试用水印以外，其他都和标准版一样。当试用期到期后，用户需联系福昕销售团队并购买 licenses 以便继续使用 Foxit WebPDF Viewer.

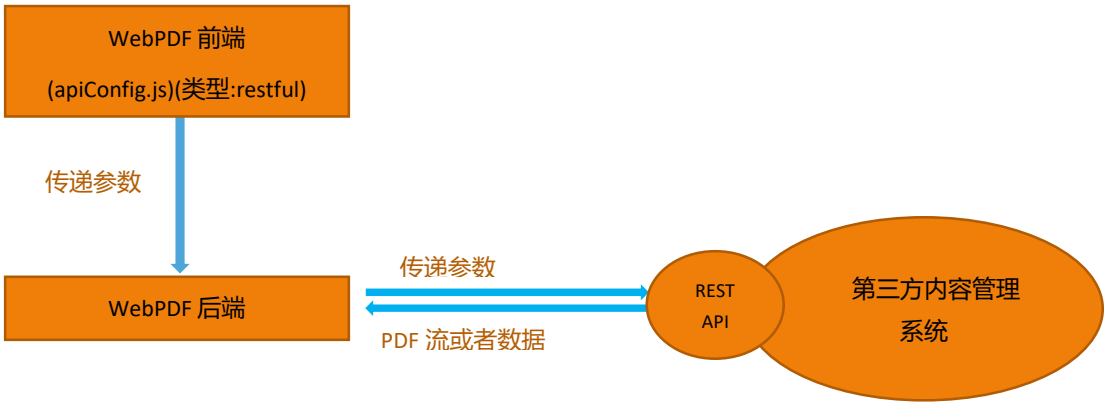
## 1.6 License

程序开发人员需购买 licenses 才能在其解决方案中使用 Foxit WebPDF Viewer。 Licenses 授予用户发布基于 WebPDF Viewer 开发的应用程序的权限。然而，在未经福昕软件公司授权下，用户不能将 Foxit WebPDF Viewer 包中的任何文档、示例代码以及源代码分发给任何第三方机构。

# 2 REST API

WebPDF Viewer 定义了一系列的接口 (名称，参数以及返回值) ，以便第三方内容管理系统可参考其并创建 REST APIs 进行 WebPDF Viewer 的集成。内容管理系统的开发人员可通过多种开发语言包括 PHP，.NET，Java 等来实现 REST APIs。Foxit WebPDF Viewer 提供了一个用 Java 来实现的示例工程。

WebPDF Viewer 将会给 REST APIs 发送带参数的请求，然后使用返回的对象 (例如，返回一个 PDF 对象) 在 WebPDF 中继续该功能。当开发人员开始使用 REST API 时，需要在 apiConfig.js 中去掉对应 API 的注释。




## 2.1 Sample REST API (Java) 工程

福昕提供了一个用 Java 实现的 REST API 示例工程以供参考。该示例工程放在 ..foxitsoftware/webpdf/sample 目录下，包含 13 个 REST APIs，请按照以下方法运行该工程。

a. getDocument()
b. exportDocument()
c. getReadablePages()
d. getSignature()
e. getContents()
f. exportSignedDocument()
g. getSignedDocument()
h. importForm()
i. exportForm()

j. getUserInfo()

k. getUserPermission()

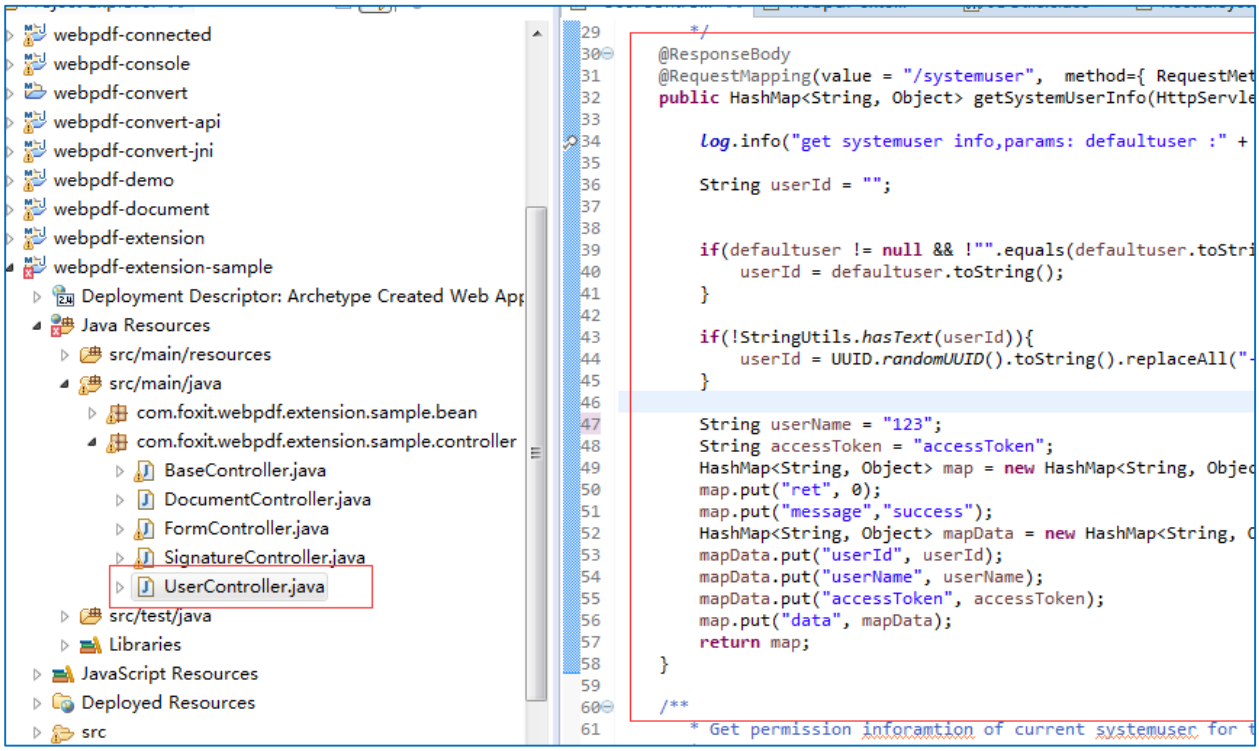
l. getPrintCount()

m. getCopyCount()

### 2.1.1 如何使用示例工程

#### *2.1.1.1 开始 Java 工程*

开始 webpdf-extension-sample 工程。请提前准备一台 Web 服务器，用来运行该工程。(如 Tomcat)




请尝试打开 localhost:8083/webpdf-extension-sample/systemuser, 如果工程运行成功，则打开后会出现如下所示的信息。(请根据你的环境修改 web 服务端口)

注意：如果需要支持中文，则需要在工程开始前将编码设置为 UTF-8。例如，在 Tomcat 的 server.xml 文件中添加 URIEncoding="UTF-8"。

```
        <Connector port="8083" protocol="HTTP/1.1"
                   connectionTimeout="20000"
                   redirectPort="8443"  URIEncoding="UTF-8"/>
          )
```

#### *2.1.1.2 apiConfig.js 配置*

apiConfig.js 是一个配置文件，用来判别 WebPDF 是否使用了 REST API 流程。默认情况下，所有 REST APIs 的配置都在 WebPDF Viewer 的前端中被注释了。开启 REST API，则需要在 apiConfig.js 中删除对应 API 的注释。以"getUserInfo" API 为例：

```
"getUserInfo": function(){
   return{
     "type": "restful",
     "url": "http://localhost:8083/webpdf-extension-sample/systemuser",
     "method": "get",
     "heads": {
               "defaultuser": getCookie('default_user')
              },
              "params":{}
          }
                   },
```

#### *2.1.1.3 验证对接 WebPDF Viewer 的 REST API*

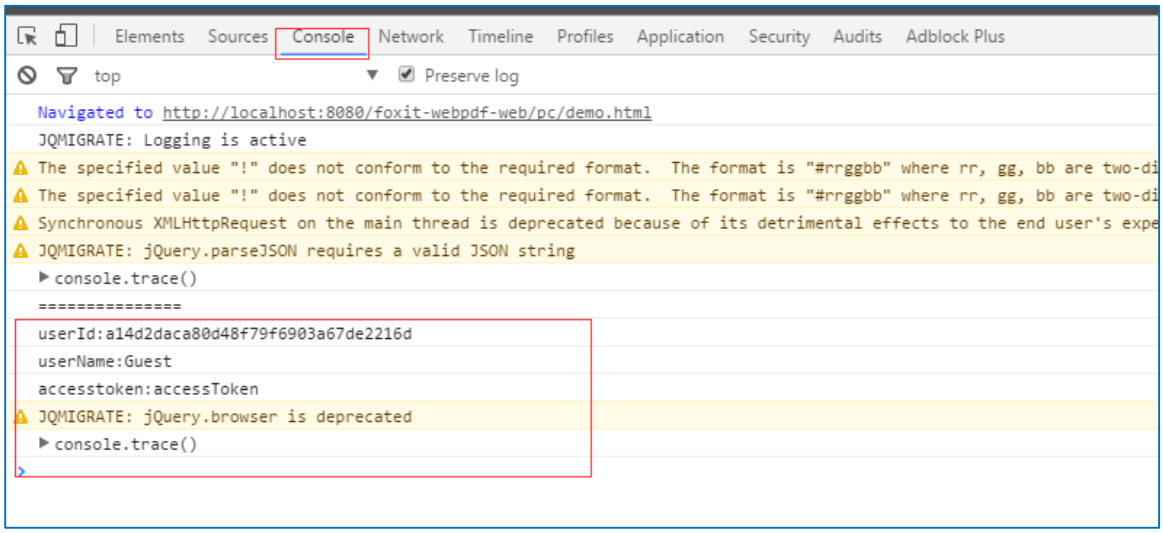
请按照以下步骤来验证 Java 示例工程中的 REST API 是否成功对接到 WebPDF。

以"getUserInfo" API 为例，开发人员可以使用 WebPDF 对应的 JavaScript API "WebPDF.ViewerInstance.updateCurrentUserInfo()" 向 REST API 发送一个请求，然后验证其返回值是否正确。WebPDF 前端示例代码在 viewer/webapp/scripts/control/pc/demo.js 文件中。

```
WebPDF.ready(docViewerId, optionsParams).then(function(data){
      … …
      return WebPDF.ViewerInstance.updateCurrentUserInfo();
}).then(function(data) {
      WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event,
      data){
             console.log("userId:"+WebPDF.AccountInstance.getUserId());
             console.log("userName:"+WebPDF.AccountInstance.getUserName());
             console.log("accesstoken:"+WebPDF.AccountInstance.getAccessToken()});
           }
});
```

如果返回值如下图所示，则表明 REST API 对接成功。

该接口需要和 REST API getUserInfo()配合使用。




## 2.2 apiConfig.js 简介

apiConfig.js 文件存放在 ...\webpdf\viewer\webapp\scripts\config\apiConfig.js. 该文件是一个配置文件，用来判别 WebPDF 是否使用了 REST API 流程。默认情况下，所有的 REST APIs 都被注释了。

```
{
type: “restful” //restful api tag
url:    //The url of the REST API
method: //The http request method, Values maybe: Get, Post, Put.
heads:  // The head information when request
params: //The parameters when request
}
```

**通常的响应格式**

REST API 通常的返回格式如下：

```
{
ret:… //the error code of return value. 0 refers to a success.
Message:.. //the detailed message information of the return result.
data:{//the data of the return result.
}
}
```

**错误码列表**

| 错误码 | 错误信息 | 描述 |
|---|---|---|
| -1 | SERVER ERROR | Interface exception |
| 0 | SUCCESS | Return success |
| 10001 | FILE_IS_NOT_EXIST | The file does not exist. |
| 10002 | SINATURE_CERT_IS_NOT_EXIST | The certification does not exist. |
| 10003 | PARSE_SINATURE_CERT_ERROR | Cannot parse the certification. |
| 10004 | SIGNATURE_FILTER_NOT_SUPPORT | Cannot support the signature filter. |
| 10005 | SIGNATURE_SUBFILTER_NOT_SUPPORT | Cannot support the signature subfilter. |

## 2.3 示例 REST API (Java) 细节

本节以 “webpdf-extension-sample” 工程为例来学习所有的 REST API 接口。

### 2.3.1 文档接口

通常，开发人员将 WebPDF Viewer 作为一种在线的 PDF 阅读解决方案。因此，WebPDF Viewer 需要提供方法允许从第三方内容管理系统获取文档或者导出 PDFs 文档。

#### *2.3.1.1 获取文档*

从第三方内容管理系统获取 PDF 文档

```
*/
     "getDocument": function(){
       return {
                 "type": "restful",
                  "url": "http://localhost:8083/webpdf-extension-sample/document",
                  "method": "get",
                  "heads": { },
                  "params": { }
                }
        }
```

**Response:**

Response body: PDF 文档的二进制流

Response Headers:

```
   {"Last-Modified": Wed, 31 Aug 2016 06:26:14 GMT}
```

#### *2.3.1.2 导出文档*

x 从 WebPDF 导出 PDF 文档和标注数据到第三方内容管理系统

```
   /*
   "exportDocument": function(){
      return {
           "type": "restful",
           "url": "http://localhost:8083/webpdf-extension-sample/document/export",
           "method": "post",
           "heads": {},
           "params": {}
         }
      }
```

**Response:**

Response body:

In common status, return

```
   {"ret":"0 ","message":”SUCCESS”}
```

如果存在错误，则返回 API 定义的错误码。

#### *2.3.1.3 获取可阅读的页数*

获取当前用户对当前文档可阅读的页数。

```
"getReadablePages": function(){
    return {
         "type": "restful",
         "url": "http://localhost:8083/webpdf-extension-
sample/document/readablepages",
         "method": "get",
         "heads": {},
         "params": {}
          };*/
         }
```

Response:

Response body:

In common status, return

```
{
  ret:0,
  message:"success",
  data:{
  readablepages:0
  }
}
```

如果存在错误，则返回 API 定义的错误码。

### 2.3.2 表单接口

当开发一个在线表单填写功能时，为了便于集中管理，你可能想要将用户填写的所有数据保存到一个数据库中。WebPDF Viewer 支持从远程服务器导入或导出表单数据。

#### *2.3.2.1 向 WebPDF 导入表单数据(XML 格式)*

从远程服务器(url)导入表单数据(XML 格式)到 WebPDF

```
/*
"importForm": function(){
      return {
            "type": "restful",
            "url": "http://localhost:8083/webpdf-extension-sample/form/import",
            "method": "get",
            "heads": {},
            "params": {}
            }
    }
```

**Response**:

Response body:

In common status, return

```
{"ret":"0 ","message":"SUCCESS"}
```

如果存在错误，则返回 API 定义的错误码。

#### *2.3.2.2 导出表单数据(XML 格式) 到第三方内容管理系统*

导出表单数据(XML 格式)到远程服务器(url)

```
/*
  exportForm: function () {
      return {
          "type": "restful",
          "url": "http://localhost:8083/webpdf-extension-sample/form/export",
          "method": "post",
          "heads": {},
          "params": {}
           }
    }
```

**Response**:

Response body

In common status, return

```
{"ret":"0 ","message":"SUCCESS"}
```

如果存在错误，则返回 API 定义的错误码。

### 2.3.3 签名接口

WebPDF Viewer 支持数字证书的集成，允许终端用户对 PDF 进行签名。APIs 定义了 WebPDF 如何从系统中获取所需的信息(签名外观图片，证书) 。

#### *2.3.3.1 getSignature*

获取签名的证书信息

```
/*
   getSignature: function() {
     return {
              "type":  "restful",
              "url": "http://localhost:8083/webpdf-extension-sample/signature",
              "method":"get",
              "heads": {},
              "params":{}
             }
      }
```

**Response**

Response body:

In common status, return

```
{"ret":"0 ",
"message":”SUCCESS”,
"data":{
    “imgData”:” ...”,  //image base64 data
     “Signer”:””,
     “dn”:””,
     “contactInfo”:””
}}
```

如果存在错误，则返回 API 定义的错误码。

#### *2.3.3.2 getContents*

获取签名的内容

```
/*
  getContents: function() {
     return {
      "type": "restful",
      "url": "http://localhost:8083/webpdf-extension-sample/signature/content",
      "method": "get",
      "heads":{},
      "params":{}
    }
```

**Response:**

Response body:

In common status, return

```
{"ret":"0 ",
"message":”SUCCESS”,
"data":{
    “content”:”...” // Hex string of signature content
}}
```

如果存在错误，则返回 API 定义的错误码。

#### *2.3.3.3 getSignedDocument*

获取签名后的 PDF 文档

```
/*
"getSignedDocument": function(){
   return{
     "type": "restful",
     "url": "http://localhost:8083/webpdf-extension-sample/signature/document",
     "method":"get",
     "head": {},
     "params": {}
   }
 }
```

**Response:**

Response body:

In common status, return the binary stream of the PDF.

#### *2.3.3.4 exportSignedDocument*

导出签名后的 PDF 文档

```
/*
exportSignedDocument:function() {
return {
 "type": "restful",
 "url": "http://localhost:8083/webpdf-extension-sample/signature/document/export",
 "method": "post",
 "heads":{},
 "params":{}
       }
   }
```

**Response:**

Response body:

In common status, return

```
   {"ret":"0 ",
   "message":”SUCCESS”,
   "data":{
      “fileName”:”...”
   }}
```

如果存在错误，则返回 API 定义的错误码。

### 2.3.4 用户接口

集成第三方内容管理系统的用户账号到 WebPDF Viewer 将允许用户保存其来自 WebPDF Viewer 特定的数据，例如注释，手写签名模板， 表单数据等。同时也支持从文档管理系统导入用户权限 – WebPDF Viewer 可以根据每个用户的权限设置提供不同的功能。

#### *2.3.4.1 getUserInfo*

获取用户信息，如用户 id, 用户名等。

```
*/
"getUserInfo": function(){
     return{
        "type":   "restful",
         "url":   "http://localhost:8083/webpdf-extension-sample/systemuser",
         "method":"post",
         "heads": {
                    "defaultuser": getCookie('default_user')
                  },
         "params":{}
        }
    }
```

如果开发人员不想通过 web service 来设置用户信息，则也可以在前

端 ..\webpdf\viewer\webapp\scripts\control\pc\demo.js 中设置。

```
WebPDF.ready(docViewerId, optionsParams).then(function(data) {
        var viewToolBar = ViewToolBar.getViewToolBar(WebPDF.optionParameters);
        viewToolBar.init();
       var param = {
            accessToken:"testAccessToken",
            userName:"testUserName",
            userId:"testUserId"
        };
        return WebPDF.ViewerInstance.updateCurrentUserInfo(param);
    })
```

**Response:**

Response body:

In common status, return

```
{"ret":"0 ",
"message":”SUCCESS”,
"data":{
    “userId”:””,
     “userName”:””,
     “accessToken”:””
}}
```

如果存在错误，则返回 API 定义的错误码。

#### *2.3.4.2 getUserPermission*

获取当前用户对当前文档的权限

```
/*
Get the permission of current user for current document
*/
   "getUserPermission": function(){
      return{
         "type":    "restful",
         "url":     "http://localhost:8083/webpdf-extension-
                    sample/systemuser/permission",
          "method": "post",
          "heads":  {},
           "params":{}
      }
  }
```

**Response:**

Response body:

In common status, return

```
{"ret":"0 ",
"message":”SUCCESS”,
"data":{
    “permission”:-1,  // -1 for all permission
                       0 for no permissions including view
                       1 for view permission,
                       2 for login permission,
                       4 for extract content permission,
                       8 for print permission,
                       16 for download permission,
                       32 for comment permission,
                       64 for signature permission,
                       128 for form filling permission.
}}
```

如果存在错误，则返回 API 定义的错误码。

### 2.3.5 打印控制接口

在某些情况下，文档所有者想要控制打印的页面以此来限制未付费用户的使用。

#### *2.3.5.1 getPrintCount*

获取当前用户对当前文档可打印的页面范围，以及剩余可打印的页面数。

```
"getPrintCount": function ()
{
     /* return{
     "type": "restful",
     "url": "http://localhost:8083/webpdf-extension-sample/api/print/count",
     "method": "get",
     "heads": {},
     "params":{}
     }*/
}
```

**Response:**

Response body:

In common status, return

```
{
      ret:0,
      message:"success",
      data:{
      printRange:1,2,3 //The printable page range at this time, can be set 1,2-
      5
      remainingPrintCount:3 //The reminding printable page count
           }
`
```

如果存在错误，则返回 API 定义的错误码。

### 2.3.6 文本复制控制接口

在某些情况下，文档所有者想要控制可复制的文本字符数以此来限制未付费用户的使用。

#### *2.3.6.1 getCopyCount*

获取允许复制的文本字符数以及剩下可复制的字符数

```
"getPrintCount": function ()
{
     /* return{
     "type": "restful",
     "url": "http://localhost:8083/webpdf-extension-sample/api/print/count",
     "method": "get",
     "heads": {},
     "params":{}
     }*/
}
```

**Response:**

Response body:

In common status, return

```
{
      ret:0,
      message:"success",
      data:{
      copyCount:3 // the text count that allow to copy at this time
      remainingCount:0 // the remaining count
           }
}
```

如果存在错误，则返回 API 定义的错误码。

## 2.4 在 WebPDF Viewer 的前端调用 REST API

本章节将详细介绍如何在 WebPDF Viewer 前端使用这些 REST APIs. 你可以通过日志信息来验证其的使用结果。

首先，在{webapp}/webpdf-extension-sample/WEB-INF/classes/log4j.properties 文件中修改登录级别为"INFO"， 然后重启该 web 工程。

```
#log4j.rootLogger=ERROR,FILE
log4j.rootLogger=INFO, FILE

#CONSOLE
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d-[WebReader] %p

#FILE
log4j.appender.FILE=org.apache.log4j.DailyRollingFileAppender
log4j.appender.FILE.File=${webapp.root}/logs/webpdf.log
```

在{webapp}/webpdfd-extension-sample/log/webpdf.log 文件中查看日志文件，记录内容如下所示：

```
INFO localhost-startStop-1 (DocumentController.java:48) - the EXPORT_HOME is:/D:/webpdf_home_webpdf/.metadata/.p
INFO localhost-startStop-1 (FormController.java:39) - the EXPORT_HOME is:/D:/webpdf_home_webpdf/.metadata/.plugi
INFO http-bio-8083-exec-3 (SystemUserController.java:34) - get systemuser info,params: defaultuser :a14d2daca80d
INFO http-bio-8083-exec-4 (DocumentController.java:67) - get document, params: docUri:http://localhost:8080/foxi
INFO http-bio-8083-exec-5 (SystemUserController.java:88) - get systemuser permission,params: accessToken :access
```

### 2.4.1 文档

#### *2.4.1.1 导出文档*

WebPDF Viewer 提供了一个前端接口去调用 REST API exportDocument():

```
WebPDF.ViewerInstance.exportDocumentToUrl (exportFileName,params,
successCallback, failCallback)
```

前端示例:

```
   $("#btnExportPDFRemote").click(function(){
   WebPDF.ViewerInstance.exportDocumentToUrl("exportname.pdf",null,function(){aler
t("success")}, function(){alert("fail")});
   });
```

验证:

移除 exportDocument()的注释，点击 test 按钮，如果在 web 服务器的特定文件夹下会生成一个 PDF 文件，并且日志如下，则说明 REST API 调用成功。

```
INFO localhost-startStop-1 (DocumentController.java:48) - the EXPORT_HOME is:/D:/webpdf_home_webpdf/.metadata/.plugins/org.eclipse.wst.server.core/tmp2
INFO localhost-startStop-1 (FormController.java:39) - the EXPORT_HOME is:/D:/webpdf_home_webpdf/.metadata/.plugins/org.eclipse.wst.server.core/tmp2/wtp
INFO http-bio-8083-exec-3 (SystemUserController.java:34) - get systemuser info,params: defaultuser :a14d2daca80d48f79f6903a67de2216d
INFO http-bio-8083-exec-4 (DocumentController.java:67) - get document, params: docUri:http://localhost:8080/foxit-webpdf-web/docs/sample/butterflies.pd
INFO http-bio-8083-exec-5 (SystemUserController.java:88) - get systemuser permission,params: accessToken :accessToken,userId :6e5d18bdfa684c8e8e5e279fc
INFO http-bio-8083-exec-6 (DocumentController.java:156) - export document (Method: POST)
```

#### *2.4.1.2 获取文档*

WebPDF Viewer 没有提供直接调用 REST API getDocument() 的前端接口，开发人员可以在 ..\webpdf\viewer\webapp\scripts\control\pc\demo.js 中使用 openFileByUri ()

```
   WebPDF.ViewerInstance.openFileByUri()
```

验证:

移除 getDocument() 的注释，点击 test 按钮，如果在 web 服务器的特定文件夹下生成一个 PDF 文件，并且日志如下，则说明 REST API 调用成功。

```
INFO localhost-startStop-1 (DocumentController.java:48) - the EXPORT_HOME is:/D:/webpdf_home_webpdf/.metadata/.plugins/org.eclipse.wst.server.core/tmp2/wtpwe
INFO localhost-startStop-1 (FormController.java:39) - the EXPORT_HOME is:/D:/webpdf_home_webpdf/.metadata/.plugins/org.eclipse.wst.server.core/tmp2/wtpwebapp
INFO http-bio-8083-exec-3 (SystemUserController.java:34) - get systemuser info,params: defaultuser :a14d2daca80d48f79f6903a67de2216d
INFO http-bio-8083-exec-4 (DocumentController.java:67) - get document, params: docUri:http://localhost:8080/foxit-webpdf-web/docs/sample/butterfiles.pdf
INFO http-bio-8083-exec-5 (SystemUserController.java:88) - get systemuser permission,params: accessToken :accessToken,userId :6e5d18bdfa684c8e8e5e279fc63a7f9
```

#### *2.4.1.3 获取可阅读的页数*

WebPDF Viewer 没有提供直接调用 REST API getReadablePages() 的前端接口。当 viewer 的前端发出图片、文本和标注的请求时，WebPDF 将会调用 getReadablePages()，并且触发 WebPDF.EventList.PAGE_SHOW_LIMIT 事件。

验证:

移除 getReadablePages() 的注释，然后重新打开一个 PDF 文档。如果文档可以成功打开，并且日志(当请求图片时)如下，则说明 REST API 调用成功。

```
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a822ef94318841152afd508e
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a822ef94318841152afd508e2
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a822ef94318841152afd508e2
```

同时，WebPDF.EventList.PAGE_SHOW_LIMIT 将会被触发，可以从结果数据中进行检测。

```
viewer.on(WebPDF.EventList.PAGE_SHOW_LIMIT, function(event, data)
{
alert('pageIndex:'+ data.pageIndex);
});
```

### 2.4.2 表单数据导入/导出

#### *2.4.2.1 向 WebPDF 导入表单数据(XML 格式)*

WebPDF Viewer 提供了一个前端接口去调用 REST API importForm():

```
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).importXMLFromUrl
(params, successCallback, failCallback)
```

前端示例:

```
$("#btnImportFormRemote").click(function(){
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).importXMLFromUrl(nul
l,function(){alert("success")}, function(){alert("fail")});
});
```

验证:

移除 importForm() 的注释。点击 test 按钮，如果表单域的值被导入的 XML 文件的数据填充，并且日志如下，则说明 REST API 调用成功。

```
] INFO http-bio-8083-exec-3 (FormController.java:60) - export form params (Method: POST), fileName:export.xml
] INFO http-bio-8083-exec-2 (FormController.java:132) - import form
```

请注意 importForm() 和 exportForm() 接口定义在 webpdf-extension-sample 工程中的 FormController.java 类中。它们使用 {project}/WEB-INF/classes/export/ 作为默认的路径。开发人员可以相应地修改文件路径。

#### *2.4.2.2 导出表单数据 (XML 格式) 到第三方内容管理系统*

WebPDF Viewer 提供了一个前端接口去调用 REST API exportForm():

```
   WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).exportXMLFromUrl
(filename, params, successCallback, failCallback)
```

前端示例:

```
   $("#btnImportFormRemote").click(function(){
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName).exportXMLFromUrl("ex
port.xml", null,function()
   {alert("success")}
   , function()
   {alert("fail")}
   );
});
```

验证:

移除 exportForm()的注释，点击 test 按钮，如果在 web 服务器的特定文件夹下会生成一个 XML 文件，并且日志如下，则说明 REST API 调用成功。

```
INFO http-bio-8083-exec-3 (FormController.java:60) - export form params (Method: POST), fileName:export.xml
```

### 2.4.3 签名

WebPDF Viewer 没有提供直接调用签名相关 REST API 的前端接口。

验证:

a. 移除 getSignature()、getContents()、exportDocument() 和 getSignedDocument() 的注释。如果在 Sample Reader 中带数字证书功能的普通签名可以工作，并且日志如下，则说明 REST API 调用成功。

```
Controller.java:60) - export form params (Method: POST), fileName:export.xml
Controller.java:132) - import form
atureController.java:135) - get signature, params : imgUrl:http://localhost:8080/foxit-webpdf-web/images/signature/Sample%20Software%20Incorporation(Cir).pnc
atureController.java:202) - get signature content , params : hash :ZGZE7vIFV96ZFCiqRnizA70RK8Q=,cert:Sample Software Incorporated.pfx,filter:Adobe.PPKLite,su
atureController.java:478) - export signature document(Method: POST)
atureController.java:390) - save signature document,params: docuri :7FormField1_signed.pdf
emUserController.java:88) - get systemuser permission,params: accessToken :accessToken,userId :321e076d043a48169fb7bfdca1d4f698,uri :7FormField1_signed.pdf,v
```

如果需要支持中文签名，则需要将编码设置为 UTF-8。例如，在 server.xml 文件中添加 URIEncoding=”UTF-8”。

```
      <Connector port="8083" protocol="HTTP/1.1"
                 connectionTimeout="20000"
                 redirectPort="8443"  URIEncoding="UTF-8"/>
        )
```

### 2.4.4 用户接口

#### *2.4.4.1 getUserInfo*

WebPDF Viewer 提供了一个前端接口去调用 REST API getUserInfo():

```
WebPDF.ViewerInstance.updateCurrentUserInfo(param)
```

在 Sample Reader 中，参考 viewer/webapp/scripts/control/pc/demo.js 中的示例代码。

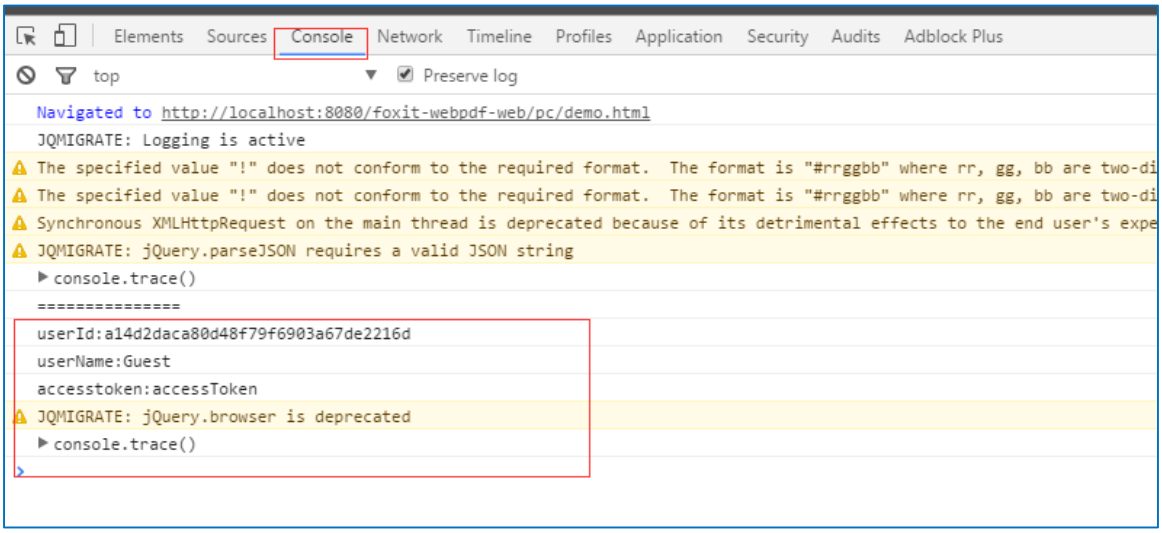
```
  WebPDF.ready(docViewerId, optionsParams).then(function(data) {
        …
        return WebPDF.ViewerInstance.updateCurrentUserInfo();
    })
```

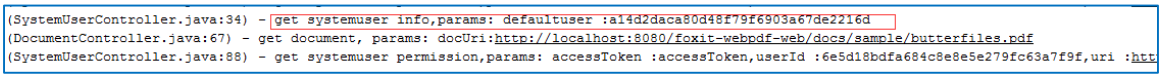
验证:

移除 getUserInfo()的注释。将下列代码添加到 webapp/scripts/control/pc/demo.js 中，然后在浏览器控制台验证输出的信息。

```
WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event, data)
{
        console.log("userId:"+WebPDF.AccountInstance.getUserId());
        console.log("userName:"+WebPDF.AccountInstance.getUserAccount());
        console.log("accesstoken:"+WebPDF.AccountInstance.getAccessToken());
   });
```

在浏览器控制台查看输出信息。如果输出信息和日志如下，则说明 REST API 调用成功。



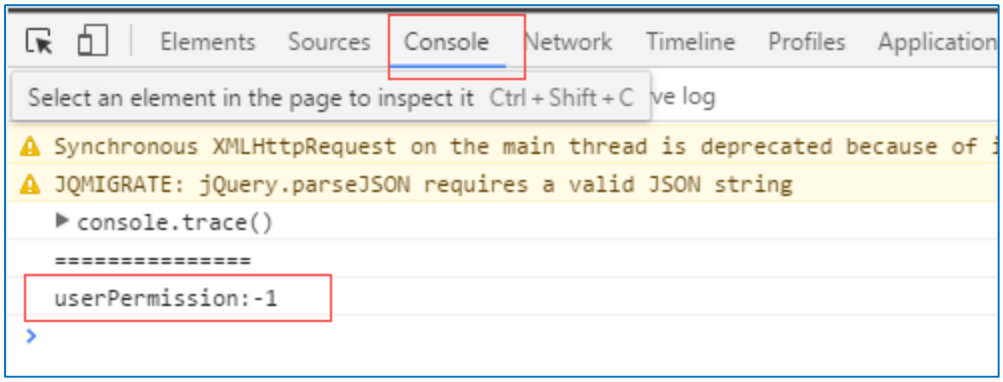

#### *2.4.4.2 getUserPermission*

验证:

移除 getUserPermission() 的注释。将下列代码添加到 webapp/scripts/control/pc/demo.js，

```
WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event, data)
{
        console.log("userPermission:"+WebPDF.ViewerInstance.getUserPermission());
   });
```

在浏览器控制台查看输出信息。如果输出信息和日志如下，则说明 REST API 调用成功。




```
(SystemUserController.java:34) - get systemuser info,params: defaultuser :a14d2daca80d48f79f6903a67de2216d
(DocumentController.java:67) - get document, params: docUri:http://localhost:8080/foxit-webpdf-web/docs/sample/butterfiles.pdf
(SystemUserController.java:88) - get systemuser permission,params: accessToken :accessToken,userId :6e5d18bdfa684c8e8e5e279fc63a7f9f,uri :ht
```

### 2.4.5 打印控制接口

#### *2.4.5.1 getPrintCount*

WebPDF Viewer 没有提供直接调用 REST API getPrintCount() 的前端接口。当用户在打印对话框中设置打印页面范围，开始打印时，WebPDF 将会调用 getPrintCount()，并且触发 WebPDF.EventList.PAGES_Prepare_PRINT 事件。

验证:

移除 getPrintCount() 的注释，然后尝试打印。如果打印成功，并且日志如下，则说明 REST API 调用成功。

```
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a822ef9431884115
(PrintController.java:41) - preprint: fileId = 99bb7a822ef9431884115 2afd508e2d7, userId = 78f9b4060cd54c279d25368e6b87514d ,printRange = 1-8
```

同时，WebPDF.EventList.PAGES_PREPARE_PRINT 将会被触发，可以从结果数据中进行检测。

```
WebPDF.viewerInstance.on(WebPDF.EventList.PAGES_PREPARE_PRINT, function(event,
  data){
WebPDF.alert(null,null,"printRange " + data.printRange + ",can print page
  count:" + data.remainingPrintCount + "");
```

### 2.4.6 文本复制控制接口

#### *2.4.6.1 getCopyCount*

WebPDF Viewer 没有提供直接调用 REST API getCopyCount() 的前端接口。当用户复制文本时，WebPDF 将会调用 getCopyCount()。

验证:

移除 getCopyCount() 的注释，然后尝试进行文本复制。如果复制成功，并且日志如下，则说明 REST API 调用成功。

```
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a8
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a8
 (DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a8
(DocumentController.java:278) - Get the readable pages(Method: PUT/POST/PUT), params: userId:78f9b4060cd54c279d25368e6b87514d,fileId :99bb7a8
(PrintController.java:41) - preprint: fileId = 99bb7a822ef94318841152afd508e2d7, userId = 78f9b4060cd54c279d25368e6b87514d ,printRange = 1-8
(TextController.java:42) - get rest count params (Method: GET), userId:78f9b4060cd54c279d25368e6b87514d
(TextController.java:43) - get rest count params (Method: GET), docId:99bb7a822ef94318841152afd508e2d7
(TextController.java:44) - get rest count params (Method: GET), copyCount:248
```

同时，回调函数将会被触发，如下所示：

```
   WebPDF.ViewerInstance.on(WebPDF.EventList.DOCUMENT_LOADED, function(event,
data) {
   var textSelectionHandler =
WebPDF.ViewerInstance.getToolHandlerByName(WebPDF.Tools.TOOL_NAME_SELECTTT;
   var callback = function(copyInfo, remainingCount){
   console.log("current selected text: " +
copyInfo+"remainingCount:"+remainingCount);
```

# 3 前端 API

Foxit WebPDF Viewer 为开发人员提供了一系列的接口用来创建定制的 WebPDF Viewer。创建一个 HTML 页面，然后在其中创建一个 DIV 标签作为 viewer 对象的容器。

创建一个名为“myviewer”的 HTML 页面 (可选择自己喜欢的名字)，将其放到安装目录 ..\webpdf\viewer\webapp\pc 下。

在 HTML 的<body>部分创建一个 标签作为 WebPDF Viewer 的容器，ID 设为“docViewer”。

```
<div id="docViewer" style=“backgroud:#ABABAB;"></div>
```

在 HTML 的<head>部分为 viewer 元素添加一个样式，使其只占用一个合理大小的区域。

```
<style>
        #docViewer {
            height:700px;
            width:800px;
        }
    </style>
```

在<head>部分添加 css 样式链接

```
<link rel="stylesheet" type="text/css" href="../styles/reader/pc/webpdf.frontend.mini.css" />
<link rel="stylesheet" type="text/css" href="../styles/reader/webpdf.mini.css"/>
```

在 HTML 的 body 部分靠近(</body>)处添加下面的脚本来创建一个新的 viewer 实例

```
   <script type="text/javascript" src="../scripts/jquery-1.10.2.min.js"></script>
   <script type="text/javascript" src="../scripts/jquery-migrate-1.2.1.js"></script>
   <script type="text/javascript"
src="../scripts/release_websdk/webpdf.tools.mini.js?v=20160920"></script>
   <script type="text/javascript" src="../scripts/control/common/common.js"></script>
   <script type="text/javascript" src="../scripts/config/config.js"></script>
   <script type="text/javascript" src="../scripts/config/apiConfig.js"></script>
   <script type="text/javascript" src="../scripts/release_websdk/webpdf.mini.js"></script>
   <script type="text/javascript">
       var docViewerId = 'docViewer';
       $(document).ready(function() {
           var optionsParams = {
               language: getLanguage(),
               customizedUrl: getBaseUrl(),
               baseUrl: getBaseUrl()
           };
           WebPDF.ready(docViewerId,optionsParams).then(function(data) {
               return WebPDF.ViewerInstance.updateCurrentUserInfo();
           }).then(function(data) {
               var openFileParams = {
                   url: getBaseUrl() + 'docs/sample/butterfiles.pdf'
               };
               WebPDF.ViewerInstance.openFileByUri(openFileParams);
           });
           $(window).resize(function() {
               if(WebPDF == null || WebPDF.ViewerInstance == null){
                   return;
               }
               var $docViewer = $("#docViewer");
               var viewWidth = $docViewer[0] && $docViewer[0].offsetWidth ?
$docViewer[0].offsetWidth + $docViewer[0].offsetLeft: $(window).width();
               var viewHeight = $docViewer[0] && $docViewer[0].offsetHeight ?
$docViewer[0].offsetHeight : $(window).height();
               WebPDF.ViewerInstance.updateLayout(viewWidth, viewHeight);
           });
       });
   </script>
```

通过访问 http://localhost:8080/viewer/pc/myviewer.html 打开创建的 viewer。

## 3.1 PC 端 WebPDF Viewer

本章节将会介绍如何在 PC 端使用 WebPDF Viewer APIs.

下列文件包括了本章使用的主要参考代码。

```
File 1: ..\webpdf\viewer\webapp\pc\index.html
File 2: ..\webpdf\viewer\webapp\scripts\control\common
File 3: ..\webpdf\viewer\webapp\scripts
control\pc (All the JavaScript files of PC view)
File 4: ..\webpdf\viewer\webapp\styles\reader\pc (css files of pc view)
File 5: FoxitWebPDF_APIREFERENCE (API file)
```

### 3.1.1 打开一个 PDF 文件

一旦 集成 REST API getDocument()，通过使用 openFile() 传送 PDF URL 到 WebPDF Viewer 可以打开网络中的 PDF 文件。

在 ..\webpdf\viewer\webapp\scripts\control\pc\index.js 中提供了详细的代码，以供参考。

Viewer 的常用 APIs

| API Name | Description |
|---|---|
| closeFile() | Close current PDF file |
| exportDocument() | Export documents to local browsers |
| exportDocumentToUrl() | Export current document to a remote server |
| exportAnnotsToFDF() | Export annotations of the document into a FDF file |
| exportAnnotsToXFDF() | Export annotations of current document into a XFDF file |
| exportAnnotsToXFDFStream() | Export annotations of current document into a XFDF stream |
| exportDocumentStream() | Extract document to stream |
| Fire() | Fires an event with the given name and data |
| focus() | Focus the viewport so it can be natively scrolled with the keyboard |
| generateDocId() | Generate assert ID of current PDF document. Before open a PDF file, the viewer needs to load the PDF content by the 3rd document loader. |
| getCurToolHandlerName() | Get the current tool handler name. |
| getCurZoomLevel() | Get the current zoom level. |
| getFileID() | Get the ID of current open document |
| getFileName() | Ge the file name of the current PDF document. |
| getOptions() | Get the configuration options. |
| getPageAnnots() | Get the annotation data of specific page in json format |

| | |
|---|---|
| getPageAnnotation() | Get the annotation data of specific page in json format. |
| getPageCount() | Get the page counts of current document. |
| getPluginByName() | Get an instance of plug-in object according to its name |
| getThumbnail() | Get the URL of page thumbnail by the specified page index. |
| getToolHandlerByName() | Get the instance of tool handler according to its name |
| getUserPermission() | Get user permission. |
| getViewMode() | Get current view mode. |
| getZoomLevels() | Get the list of available zoom levels |
| gotoNextPage() | Go to the next page. |
| gotoPage() | Jump to a page |
| gotoPageByDestination() | Jump to a page by the specified page index and the offset to be scrolled to |
| gotoPrevPage() | Go to the previous page |
| hasForm() | Detect whether the document has form fields |
| HideAllAnnots() | Hide all annotations |
| hideAnnots() | Hide annotations |
| highlightText() | Highlight an area by rectangle |
| importAnnotsFromFDF() | Import annotation to current document from FDF |
| importAnnotsFromXFDF() | Import annotations to current document from XFDF |
| isDocModified() | Check whether current document has been modified |
| isFitWidth() | Check whether the current zoom level is WebPDF.ZOOM_FIT_WIDTH |
| isJREngineReady() | Check whether JavaScript engine is ready |
| load() | Load current document assets |
| off() | Removes an event handler from a given event If the handler is not provided; remove all handlers of the given type |
| on() | Adds a new event handler for a particular type of event |
| openFileByStream() | Open a PDF file from file stream |
| print() | Open the Print dialog |
| rotate() | Rotate the page view. |
| save() | Save user data of current PDF to server |
| searchAllText() | Full-text search |

| | |
|---|---|
| | This function obtains the data in an asynchronous way. The data will be returned in the callback SEARCH_ALL_FINISHED, Please refer to the event WebPDF.EventList.SEARCH_ALL_FINISHED |
| searchText() | Search text |
| setCurrentToolByName() | Set the current tool by its name |
| setFullScreenFlag | Set the flag to indicate current viewer whether show in full screen mode or not |
| setLayoutShowMode() | Change the page layout mode of the viewer |
| showAllAnnots() | Show all annotations |
| showAnnots() | Show annotations |
| showFullPage() | Set to display the specified page in full screen mode. |
| updateCurrentUserInfo() | Update current login user information. |
| updateLayout() | Update the layout of current viewer instance |
| updateUUID() | Update current UUID after user login. |
| zoomTo() | Zoom to the given value |

### 3.1.2 Annotation

一个 annotation 关联一个对象，如 PDF 文档中某一页某个位置的下划线、高亮或者备注。它提供了一种通过鼠标和键盘与用户进行交互的方式。Foxit WebPDF Viewer 支持 annotation 的类型包括备注、下划线、波浪线、删除线、高亮、打字机、铅笔、直线、折线、箭头、椭圆、矩形、多边形、注释框、图章。WebPDF Viewer 提供了创建、访问和删除 annotations 的 APIs. 调用 “WebPDF.ViewerInstance.setCurrentToolByName()” 接口可以在不同的 annotation 工具中进行切换。请参考 “toolbar.js” 中的具体代码。

**示例：设置当前的 annotation 工具为高亮**

```
WebPDF.ViewerInstance.setCurrentToolByName(WebPDF.Tools.TOOL_NAME_COMMENT_HIGHTLIGHT)
```

当在文档中对 annotations 进行编辑后，用户需要保存其修改。调用 “ViewerInstance.save()” 接口来保存 annotations 和用户数据。请参考 “toolbar.js” 中的具体代码。

**示例：点击 Save 按钮**

```
$("#btnSave").off('click').click(function(){
   WebPDF.ViewerInstance.save();
});
```

### 3.1.3 Rotation

当文档布局显示非常规时，用户可能需要对页面视图进行旋转。旋转页面视图，调用“WebPDF.ViewerInstance.rotate()” 接口。请参考 “toolbar.js” 中的具体代码。

**示例：点击 Rotate Left 菜单**

```
$("#btnLeftRotate").off("click").click(function(){
            WebPDF.ViewerInstance.rotate(WebPDF.ROTATE_LEFT);
    });
```

### 3.1.4 Download

用户可能需要将 WebPDF Viewer 中的 PDF 文档下载到本地磁盘以备进一步使用。下载 PDF 文档，调用 “WebPDF.ViewerInstance.exportDocument()” 接口。请参考 “toolbar.js” 中的具体代码。

**示例：点击 Download 按钮**

```
$("#btnExportPDF").off("click").click(function(){
    WebPDF.ViewerInstance.exportDocument(null);
    return false;
});
```

### 3.1.5 Print

用户可以调用 “WebPDF.ViewerInstance.print()” 接口进行 PDF 打印。请参考 “toolbar.js” 中的具体代码。

**示例：点击 Print 按钮**

```
  $("#btnPrint").click(function(){
    WebPDF.ViewerInstance.print();
 });
```

### 3.1.6 Account

开发人员可能想获取当前用户的账户信息以备进一步使用。前端 API 提供了与账号相关的一系列接口。在 demo 中，用户账号是随机生成的。开发人员可以使用 REST API 将自己系统的账户信息集成到 WebPDF Viewer 中。请参考 “common.js” 中的具体代码。

Account 的常用 APIs

| API name | Description |
|---|---|
| getAccessToken() | Get access token of current user |
| getUserAccount() | Get account name of current user |
| getUserconfig() | Get configuration data of current user |
| getUserId() | Get user ID of current user |
| getWatermarkInfo() | Get the information of user watermark |
| initUserConfig() | Initialize to request current user configuration data after user login |
| isLogin() | Check whether user has login |
| setAccessToken() | Set access token of current user |
| setLoginState() | Set login status of current user |
| setUserAccount() | Set account of current user |
| setUserId() | Set ID of current user |
| setWatermarkInfo() | Set the user watermark information |

### 3.1.7 Form

从 1.2 版本开始，WebPDF Viewer 支持 Acroform 表单填写，以及导入和导出表单数据。调用导入/导出表单数据 REST API 将表单数据保存到数据库中。请参考 “form.js” 和 “toolbar.js” 中的具体代码。

导出或者导入 XML 格式的表单数据

```
   $("#btnExportForm").removeClass("disabled");
                    $("#btnExportForm").off("click").on("click", function () {
                        var formPlugin =
WebPDF.ViewerInstance.getPluginByName(WebPDF.FormPluginName);
                        if(formPlugin) {
                            formPlugin.exportXML();
                        }
                    });
```

Form 的常用接口

| API Name | Description |
|---|---|
| exportToXML() | Export form data to XML |
| exportToXMLStream() | Export form data to XML stream |
| exportXMLToUrl() | Export form data (XML format) to a remote server (url). |
| highlight() | Highlight all form fields. |
| importFromXML() | Import form data (XML format) from local machine |
| importFromXMLStream() | Import from data from XML stream |
| importXMLFromUrl() | Import form data (XML format) from a remote server (url). |
| isHighlightBarVisible() | Check the highlight bar visible or not. |
| showHighlightBar() | Show or hide current highlight bar. |

### 3.1.8 Watermark

WebPDF Viewer 支持两种类型的水印设置。一种是在管理员页面的全局水印设置，其将会作用到系统中的所有文档(请查看部署手册)。另一种是水印 API，其定义了每一个文档中每个水印的内容。这两种类型的水印是独立的，并且不会互相影响。

水印 API 提供了两个接口，“setWatermarkInfo()” 和 “getWatermarkInfo()”. 在 demo 中，水印数据默认保存在 cookies 中 (如果 cookies 被清除后，其将失效)。当打开一个带水印的 PDF 文档时，WebPDF Viewer 将会处理来自 cookies 的水印数据，然后使用 “setWatermarkInfo()” 将其应用到文档中。

## 3.2 移动端 WebPDF Viewer

本章节将会介绍如何在移动端使用 WebPDF Viewer APIs. 下列文件包括了本章使用的主要参考代码。

```
File 1: ..\webpdf\viewer\webapp\mobile\index.html
File 2: ..\webpdf\viewer\webapp\scripts
Control\common (All the common JavaScript files)
File 3: ..\webpdf\viewer\webapp\scripts
control\common\mobile (All the JavaScript files of Mobile view)
File 4: ..\webpdf\viewer\webapp\styles\reader\mobile (css files of Mobile view)
File 5: FoxitWebPDF_APIREFERENCE (API file)
```

### 3.2.1 打开一个 PDF 文件

当开发人员实现 REST API getDocument() 后，在网络中的 PDF 文件的通过 URL 就能在 Viewer Viewer 中打开。

在 ..\webpdf\modules\reader\webapp\scripts\control\mobile\index.js 中提供了详细的代码，以供参考。

查看 Viewer 的常用接口.

### 3.2.2 Annotation

查看 PC 端 Annotation.

请参考 “index.html” 和 “readercontrol.js” 中的具体代码。

### 3.2.3 Save Annotation

查看 PC 端 Annotation 保存 .

请参考 “index.html” 和 “readercontrol.js” 中的具体代码。

**示例：点击 Save 按钮**

```
/*
 * handle tap event of save button.
 */
$("#fwrm-main-btnSave").on("tap", function(event) {
      if (_isDocModified) {
            WebPDF.ViewerInstance.saveAnnots();
```

### 3.2.4 Rotation

查看 PC 端的 Rotation .

请参考 “index.html” 和 “more.js” 中的具体代码。

**示例：点击 Rotate 按钮**

```
$("#rotateLeft").off("tap").on("tap", function () {
      viewerInstance.rotate(WebPDF.ROTATE_LEFT);
      event.stopPropagation();
      event.preventDefault();
      return false;
});
/*
 * bind tap event on rotate right button.
 */
$("#rotateRight").off("tap").on("tap", function () {
      viewerInstance.rotate(WebPDF.ROTATE_RIGHT);
      event.stopPropagation();
      event.preventDefault();
      return false;
});
```

### 3.2.5 Account

查看 PC 端 Account .

请参考 “common.js” 中的具体代码。

### 3.2.6 Web PDF Form

查看 PC 端的 Web PDF 表单 .

请参考 “index.html” 和 “readercontrol.js” 中的具体代码。

### 3.2.7 Watermark

查看 PC 端的水印 .

# 4 Demo

从 2.2 版本开始，WebPDF Viewer 为开发人员提供了可供参考的 demos.

在 ..\webpdf\sample\webpdf-extension-sample\src\main\webapp\sample 目录下，提供了五个 demos:

- Form: PDF 表单数据处理。
- Hide-show-specific-user-annotation: 隐藏和显示特定用户的 annotation。
- Navigation-panel: 创建一个面板，然后加载一个 html 页面。
- Sharing-annotation: 加载多个用户的 annotation，以及多用户协作。
- User-permission: 处理不同类型的用户权限。

# 5 FAQ

1. **WebPDF Viewer 对 PDF 文件大小有什么限制吗？**

   福昕 WebPDF Viewer 支持 200M 以下的 PDF 文件。

2. **为什么用之前的 URL 不能打开 PDF 文件？**

   当用户用 WebPDF Viewer 打开一个 PDF 文件时，实际上 URL 是指向一个数据库的。因此，如果该 PDF 文件在数据库中的缓存被清除了，则 URL 将会失效。在这种情况下，用户需要通过文件系统重新打开 PDF 文件去激活缓存。

3. **为什么在我的浏览器中 WebPDF Viewer 显示的 PDF 发生损坏？**

   如果你正在使用 IE 浏览器，请查看该浏览器是否兼容。WebPDF Viewer 不支持旧的 IE 版本，比如 IE 6 or IE 7。因此，如果使用这些版本的 IE，PDF 的显示可能会发生损坏。

4. **如何查看 WebPDF Viewer 的 log 信息？**

   请在安装根目录下的 log 文件夹下查看 WebPDF Viewer 的 log 信息。当该文件夹存储满时，您可以清除 log 文件。

5. **当打印文档时，为什么输出文件的页数比在 WebPDF Viewer 中原始文件的页面要多？怎么解决？**

   不同的浏览器的打印配置是不同的。因此，建议在打印前设置正确的布局、纸张大小和页边距。

6. **为什么使用正确的密码不能打开 PDF 文档？**

   WebPDF Viewer 需要对用户会话进行缓存，所以在部署 WebPDF Viewer 时请确保开启 web 容器的会话功能。

7. **为什么运行 demo 的时候没有看到签名功能？**

运行 demo 的时候没有看到签名功能可能是渲染引擎在控制台被设置为 JavaScript 引擎。在当前 demo 中，JavaScript 渲染引擎不支持任何签名功能。

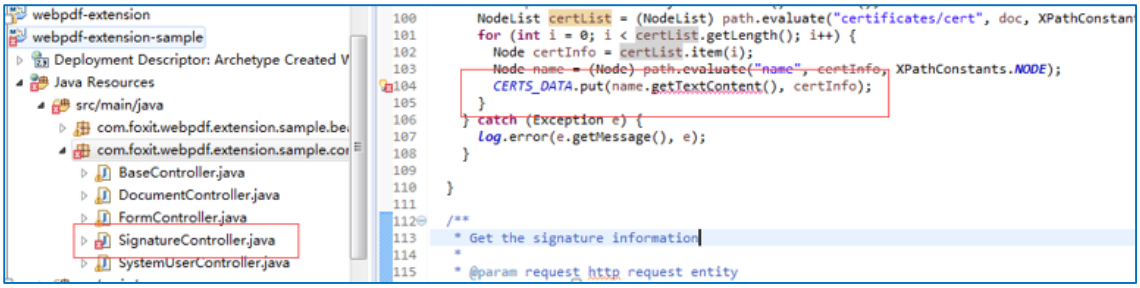
8. **如果 WebPDF Viewer 的前端和工程后台不在同一个域中，怎么使 WebPDF Viewer 成功运行？**

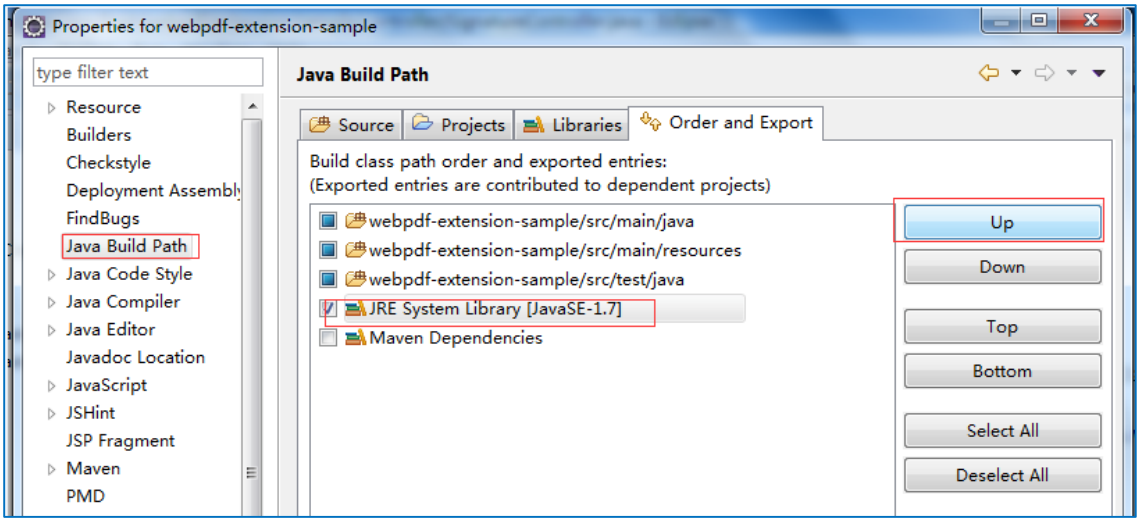
如果需要集成 WebPDF Viewer 的工程和 WebPDF Viewer 前端不在一个域中，你需要将 WebPDF Viewer 的所有静态文件(前端)拷贝到你的工程 webapp 文件夹下。同时，在 ../viewer/webapp/scripts/control/pc/demo.js 中初始化 Viewer 时，需要修改 customizedUrl 的参数。

```
   $(document).ready(function() {
     var optionsParams = {
            language: getLanguage(),
            serverBaseUrl:"http://127.0.0.1:8080/viewer/",//change the ip of Url
to the one your webpdf viewer install
           baseUrl: getBaseUrl()
     };
     WebPDF.ready(docViewerId, optionsParams).then(function(data) {
   … …
     });
});
```

9. **当运行 webpdf-extension-sample 工程时，为什么 Eclipse 总是提示在 org.w3c.dom.Node 类中 “The method getTextContent() is undefined” ？**

该工程使用 dom4j->xml-apis，其从 dom4j 调用 node 类，而不是从 JDK。因此，会提示 node 类错误。为了解决该问题，右击工程的 properties，选择 builder path，然后将 JDK 移到 maven dependencies 的上面。这样工程就会从 JDK 调用 node 类，上述的错误就也不存在了。






## 10. 是否可以禁用矩形文本选择？

一些用户更倾向于按行进行文本选择，而希望禁用矩形文本选择。从 2.2 版本开始，WebPDF Viewer 支持禁用矩形文本选择。在 config.js 中，设置如下的参数为 false 即可。

```
RectangularTextSelection:false,
```

# Support

**Foxit Support:**

http://www.foxitsoftware.com/support/

**Sales Contact:**

Phone: 1-866-680-3668

Email: sales@foxitsoftware.com

**Support & General:**

Phone: 1-866-MYFOXIT or 1-866-693-6948

Email: support@foxitsoftware.com