



友虹版式阅读软件 集成说明 SC-Reader V2.0

友虹（北京）科技有限公司
2024 年

目录

第 1 章 文档说明	2
第 2 章 产品简介	3
2.1. 产品形态	3
2.1.1. 集成方式	3
2.1.2. 方式区别	3
2.2. 产品结构	4
2.3. 前置要求	4
第 3 章 产品集成	5
3.1. 纯前端集成	5
3.2. 服务端集成	6
第 4 章 附录	9
4.1. PDF 预览	9
4.2. 服务端示例	9



第 1 章 文档说明

友虹版式阅读软件（简称：SC-Reader V2.0）支持 OFD 和 PDF 版式文件的在线阅读及 OFD 版式文件的处理且无需安装任何插件。兼容谷歌 Chrome、火狐 Firefox、360 等主流浏览器以及奇安信、红莲花等国产浏览器。也支持使用 h5 方式嵌入至移动端使用。本文档主要描述产品在不同形态下的集成过程。

第 2 章 产品简介

2.1. 产品形态

2.1.1. 集成方式

产品分为纯前端以及服务端两种集成方式，用户可按照实际业务场景选择。

➤ 纯前端集成

将提供的 JS 以及图像等静态资源文件放置于前端项目中合适位置，在业务页面使用<script>标签引入即可使用。

➤ 服务端集成

在前端集成的基础上，同时将提供 JAVA 后端 SDK 包集成至后端业务系统，并实现相关后端处理接口。

2.1.2. 方式区别

➤ 纯前端

支持 url、blob、unit8array、file 方式打开文件。此模式下时会一次性将文件所有数据加载至浏览器内存而后渲染，打开 url 文件时先通过网络获取文件数据后阅读。当目标文件体积超大时浏览器内存压力会增大，极端情况下可能会造成卡顿或浏览器假死现象。故此方式适用于阅读较小文件且网络条件尚可的场景。其集成过程较简单，仅需引入静态文件即可实现文件预览。

➤ 服务端

支持打开服务端磁盘 OFD 文件，此模式下文件直接在服务端解析，基于 OFD 文件结构实现文件的按需加载，按照前端显示需求仅将部分内容响应至前端减少前端渲染压力，完成文档的快速打开预览，同时也支持在阅读前通过后台接口控制页面显示权限或页面指定区域位置遮盖等功能。适用于超大



文件阅读以及有阅读权限控制需求的相关场景。需在集成前端的基础上再集成 JAVA 后端 SDK 并实现指定的接口。

2.2. 产品结构

➤ 前端

包含阅读器核心文件“sc-reader.zip”、“pdfjs.zip” pdf 预览插件及阅读器集成示例“demo.html”文件。

➤ 服务端

纯前端模式下集成时可忽略此目录，包含“SC-Reader-Server-1.0.0.jar”后端 SDK、“reader-demo.zip”后台示例工程（可直接导入 idea 使用）

2.3. 前置要求

➤ 前端运行环境

基于浏览器实现文件的在线预览，浏览器环境需支持 wasm，如火狐（53+）、谷歌（57+）、Edge（16+）、Safari（11+）等。

➤ 服务后端

后台 SDK 为 java 编写，建议 64 位 JDK1.8，SDK 包中相关接口及处理逻辑基于 Spring 注解实现，用户在使用此模式时须具备 Spring 框架且版本 4.0 及以上，本文档以 SpringBoot 作为集成示例项目。

第 3 章 产品集成

3.1. 纯前端集成

将静态文件放置于合适位置，通过 script 标签引入关联后即可使用（ps：不支持直接使用静态 html 打开预览，需使用 http 服务通过 url 方式访问 html 业务页面查看预览），步骤如下：

➤ 静态文件集成

将“前端”目录下压缩包“sc-reader.zip”解压并放置前端项目中合适位置（ps：目录结构不可变更，包含 fonts、img、js、wasm 等静态文件）

➤ 阅读器实例化

在业务页面按照静态文件实际位置引入 SCOFDReader.umd.min.js 并创建阅读器实例（不同时期 js 名称可能稍有不同，参考压缩包中 demo.html 实例化代码片段），示例代码片段如下：

<!--1、引入 js 文件-->

```
<script src="./sc-reader/SCOFDReader.umd.min.js"/>
```

<!--2、构建阅读器展示容器（必须定义容器宽高）-->

```
<div id="app" style="width:100%;height:100%"></div>
```

<!--3、实例化阅读器对象，设置授权码（无授权码不影响功能调用）-->

```
let ofd = new SCOFDReader(window.app, "授权码",{showMenu:true})
```

➤ 打开文件

调用打开文件接口，如 ofd.openFile(“http://192.168.1.123/file/文件.ofd”)

（参考压缩包中 demo.html 打开文件代码片段）

<!--4、打开 ofd 文件，支持 File、blob、unit8Array、url，此处示例为 url-->

```
ofd.openFile("http://192.168.1.27:9000/file/ofd.ofd");
```

➤ 其它功能

集成后可使用阅读器其它的相关功能接口，如文档信息获取、注释绘制等，详见《SC-Reader 版式阅读软件 V2.0-接口文档.pdf》

3.2. 服务端集成

服务端模式支持打开服务器所在磁盘文件，在集成了前端的基础上再集成后台 SDK 包，SDK 包中包含与前端交互的 HTTP 接口以及对应文件处理逻辑，用户需将该 SDK 包集成至自身业务系统并对指定包路径扫描注入，而后实现指定接口类以及配置文件完成集成（ps: Spring 版本要求见[章节 2.3 前置要求](#)），步骤如下：

➤ 添加 SDK 包依赖

将“服务端/SDK”目录下“SC-Reader-Server-1.0.0.jar”放置项目中并添加依赖（不同时期 SDK 包版本号可能稍有区别），而后将第三方的 maven 坐标依赖添加至 pom.xml 文件中并更新索引，见示例工程中“pom.xml”文件。（ps: 非 maven 项目可手动下载相关依赖包添加。项目中如已存在相同包在版本相差不大情况下可使用自身依赖包）

➤ 添加授权配置

服务端模式下 license 配置通过@Value 注解加载，以 springboot 项目为例，一般放置于 application.yml 或 properties 文件中，license.licensePath 为授权文件的绝对路径地址。（参考示例工程中“application.yml”文件配置）

➤ 实现 FileAccessInterface 接口

编写该接口实现类并实现所有方法，该接口共有 9 个方法，方法说明参考示例工程中“FileAccessInterfaceImpl.java”实现类。

➤ 包扫描及实现类注入

SDK 包中包含了必要的 HTTP 接口以及相关处理功能，其通过 Spring

依赖注入完成对象的实例化，故需对指定包路径添加扫描（com.yh.scofd），并将接口实现类注入（参考示例工程中“ReaderTestApplication.java”类）。

➤ 项目启动

以上步骤完成后启动项目，初次启动时将校验 license，提供前端业务域名或 ip 以及端口由我方制作授权文件，而后将授权文件放置于合适位置，在配置项 licensePath 中指定。下图：

```
2024-10-15 13:50:25.616 [main] INFO org.apache.catalina.core.AprLifecycleListener - The APR based Apache Tomcat Native library which allows optimal performance i
2024-10-15 13:50:25.676 [main] INFO o.a.c.core.ContainerBase.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
2024-10-15 13:50:25.676 [main] INFO org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization completed in 474 ms
2024-10-15 13:50:25.707 [main] INFO o.s.scheduling.concurrent.ThreadPoolTaskScheduler - Initializing ExecutorService 'a'
2024-10-15 13:50:25.730 [main] INFO com.yh.scofd.m5.y1 - start asyncServiceExecutor
2024-10-15 13:50:25.730 [main] INFO com.yh.scofd.m5.y2 - Initializing ExecutorService
2024-10-15 13:50:25.731 [main] INFO com.yh.scofd.m5.y2 - Initializing ExecutorService 'asyncServiceExecutor'
2024-10-15 13:50:25.858 [main] INFO o.s.b.a.web.servlet.WelcomePageHandlerMapping - Adding welcome page: class path resource [static/index.html]
2024-10-15 13:50:25.890 [main] INFO com.yh.scofd.license.y3 - ===== 正在安装产品证书 =====
2024-10-15 13:50:25.894 [main] WARN java.util.prefs - Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x80000002. Windows RegCreateKeyEx(..
2024-10-15 13:50:25.900 [main] ERROR com.yh.scofd.license.y6 - 证书校验失败，请联系产品提供商，提供机器码业务域名或ip以及端口获取最新授权文件！
java.nio.file.NoSuchFileException Create breakpoint : D:\licensetest\license.lic1
    at sun.nio.fs.WindowsException.translateToIOException(WindowsException.java:75)
    at sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:97)
    at sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:102)
```

再次重启项目后如提示授权验证通过并打印授权时间则代表集成通过（ps：不配置授权亦可正常使用，仅在预览界面时左上角有未授权标记以及水印背景，但不影响功能以及接口调用），下图：

```
Debugger
Frames Console
2024-10-15 13:51:13.907 [main] INFO com.example.demo.ReaderTestApplication - No active profile set, falling back to default profiles: default
2024-10-15 13:51:13.783 [main] INFO o.s.boot.web.embedded.tomcat.TomcatWebServer - Tomcat initialized with port(s): 80 (http)
2024-10-15 13:51:13.788 [main] INFO org.apache.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-80"]
2024-10-15 13:51:13.791 [main] INFO org.apache.catalina.core.StandardService - Starting service [Tomcat]
2024-10-15 13:51:13.791 [main] INFO org.apache.catalina.core.StandardEngine - Starting Servlet Engine: Apache Tomcat/9.0.13
2024-10-15 13:51:13.794 [main] INFO org.apache.catalina.core.AprLifecycleListener - The APR based Apache Tomcat Native library which allows optimal performance in produ
2024-10-15 13:51:13.837 [main] INFO o.a.c.core.ContainerBase.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
2024-10-15 13:51:13.837 [main] INFO org.springframework.web.context.ContextLoader - Root WebApplicationContext: initialization completed in 482 ms
2024-10-15 13:51:13.863 [main] INFO o.s.scheduling.concurrent.ThreadPoolTaskScheduler - Initializing ExecutorService 'a'
2024-10-15 13:51:13.882 [main] INFO com.yh.scofd.m5.y1 - start asyncServiceExecutor
2024-10-15 13:51:13.882 [main] INFO com.yh.scofd.m5.y2 - Initializing ExecutorService
2024-10-15 13:51:13.882 [main] INFO com.yh.scofd.m5.y2 - Initializing ExecutorService 'asyncServiceExecutor'
2024-10-15 13:51:14.017 [main] INFO o.s.b.a.web.servlet.WelcomePageHandlerMapping - Adding welcome page: class path resource [static/index.html]
2024-10-15 13:51:14.047 [main] INFO com.yh.scofd.license.y3 - ===== 正在安装产品证书 =====
2024-10-15 13:51:14.050 [main] WARN java.util.prefs - Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x80000002. Windows RegCreateKeyEx(...) retu
2024-10-15 13:51:14.424 [main] INFO com.yh.scofd.license.y6 - 证书安装成功，证书有效期：2024-09-30 00:00:00 - 2025-01-01 23:59:59
2024-10-15 13:51:14.424 [main] INFO com.yh.scofd.license.y3 - ===== 产品证书安装结束 =====
```

➤ 打开文件

服务端模式下静态文件的集成以及阅读器对象的实例化均与纯前端相同，仅需在调用打开文件接口前先指定后端服务的 http 地址，而后再调用打开文件方法打开服务端文件，此时阅读器将向服务端发起请求并将标识传递服务端而后实现类中可接收到该文件标记，用户结合自身业务使用该标记打开指定的 OFD 文件，部分示例代码如下：



<!--此处省略引入 js 及实例化部分代码-->

....

<!--指定服务端实际地址，通过服务端打开文件，固定为 http://ip:port/fs/-->

ofd.setFileServUrl("http://192.168.1.27:8082/fs/");

<!--调用 openFile 接口打开文件，参数为业务标识。调用此方法后服务端 FileAccessInterface 实现类的将收到该参数，此时依据业务返回对应的 OFD 文件磁盘地址或文件流即可-->

ofd.openFile("378271728233");

➤ 跨域配置（可选）

由于在服务端模式下通过访问后端地址打开 ofd 文件，所以当前端访问的业务域名与后端 sdk 所在项目的域名（即调用 setFileServUrl 接口的地址）不一致时候可能出现跨域访问导致前端报错，如 Access to XMLHttpRequest at 'http://localhost:999/xxx' from origin 'http://192.168.1.23/fs' has been blocked by CORS xx...。此时需服务后台配置允许跨域访问，可按照自身业务实际需求参考示例工程中“ReaderTestApplication.java”类中配置跨域请求。

第 4 章 附录

4.1. PDF 预览

产品默认支持 OFD 文件预览,PDF 文件使用 pdf.js 来实现 PDF 文件的预览,其步骤如下:

➤ pdfjs 引入

在前端项目根目录下新建“pdfjs”目录,将“pdfjs.zip”压缩包中所有文件放置于该目录下即可。(ps: 如用户自身项目有目录要求,也可更换至其它目录下,更换后做目录访问映射,保证“<http://ip:port/pdfjs/web/viewer.html>”链接能正常访问)

➤ 打开 pdf 文件

<!--此处省略引入 js 及实例化部分代码-->

....

<!--调用 openPDF 接口打开 PDF 文件,此处示例为 url 地址-->

ofd.openPDF("http://192.168.1.27/file/pdf/demo.pdf");

4.2. 服务端示例

“服务端/示例工程”目录下“reader-demo.zip”为服务端后台的 springboot 示例工程,将其解压后可直接导入 idea 使用。默认端口 8082。工程中“resources”目录下包含阅读器的静态文件以及 html 调用示例,启动后打开浏览器访问 url “<http://localhost:8082/>”可查看