

WPS加载项快速上手说明

WPS加载项快速上手说明

[Demo怎么能快速跑起来](#)

[WPS加载项的运行原理](#)

[WPS加载项技术架构](#)

[OA助手——WPS加载项的一种应用](#)

[WPS加载项正常运行时WPS客户端的必要配置](#)

[WPS加载项（本Demo）代码结构说明](#)

[服务端代码结构说明](#)

[WPS加载项代码结构说明](#)

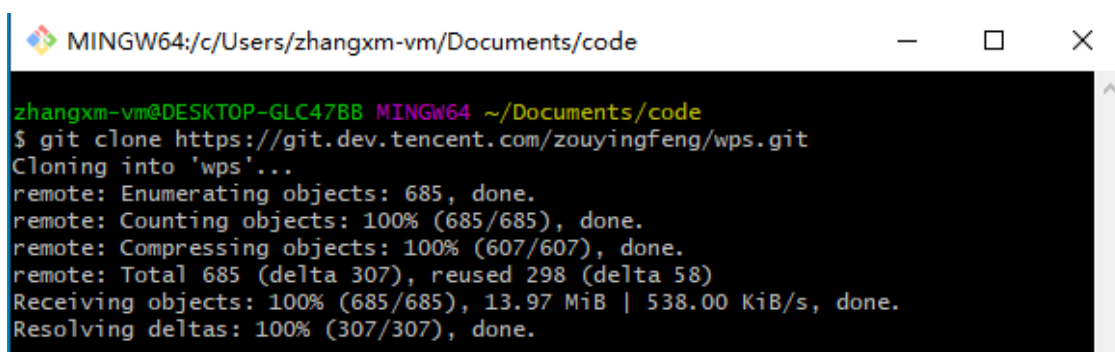
[WPS加载项调试](#)

[FAQ](#)

Demo怎么能快速跑起来

1. 下载[git](#)和[nodejs](#)，执行默认安装即可；
2. 获取代码到本地，新建一个文件夹，并在此文件夹下通过git客户端执行clone命令，如下图所示：

```
1 | git clone https://e.coding.net/zouyingfeng/wps.git
```



```
MINGW64/c:/Users/zhangxm-vm/Documents/code
zhangxm-vm@DESKTOP-GLC47BB MINGW64 ~/Documents/code
$ git clone https://git.dev.tencent.com/zouyingfeng/wps.git
Cloning into 'wps'...
remote: Enumerating objects: 685, done.
remote: Counting objects: 100% (685/685), done.
remote: Compressing objects: 100% (607/607), done.
remote: Total 685 (delta 307), reused 298 (delta 58)
Receiving objects: 100% (685/685), 13.97 MiB | 538.00 KiB/s, done.
Resolving deltas: 100% (307/307), done.
```







此电脑 > 文档 > code >

名称	修改日期
wps	2020/2/10 12:01

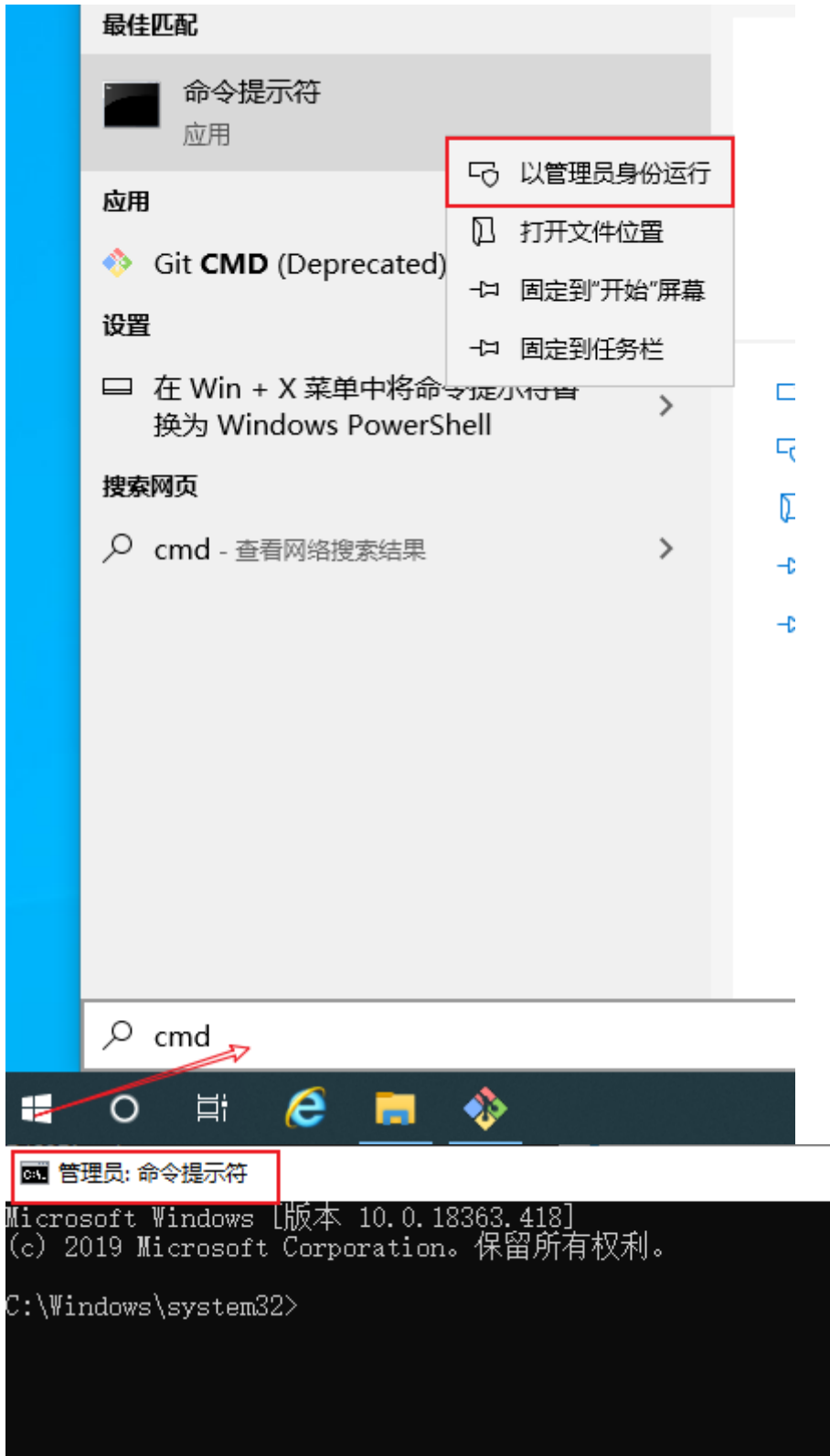
此电脑 > 文档 > code > wps >

名称	修改日期
cpp	2020/2/10 12:01
java	2020/2/10 12:01
np-example	2020/2/10 12:01
oaassist	2020/2/10 12:01

3. JSAPI的Demo所在文件夹是 `oaassist`，进入后的代码结构如下图所示：

名称	修改日期
 EtOAAssist	2020/2/10 12:01
 server	2020/2/10 12:01
 WppOAAssist	2020/2/10 12:01
 WpsOAAssist	2020/2/10 12:01
 demo.html	2020/2/10 12:01
 README.md	2020/2/10 12:01

4. **以管理员的身份**启动命令行工具，操作步骤如下图：



切换到SAPI的Demo文件夹的 server 目录下，执行如下命令（文档例子中的目录是：

C:\Users\zhangxm-vm\Documents\code\wps\oaassist\server :

1 | `cd C:\Users\zhangxm-vm\Documents\code\wps\oaassist\server`

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.18363.418]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Windows\system32>cd C:\Users\zhangxm-vm\Documents\code\wps\oaassist\server

C:\Users\zhangxm-vm\Documents\code\wps\oaassist\server>dir
驱动器 C 中的卷没有标签。
卷的序列号是 5E94-9455

C:\Users\zhangxm-vm\Documents\code\wps\oaassist\server 的目录
2020/02/10 12:01 <DIR>          .
2020/02/10 12:01 <DIR>          ..
2020/02/10 12:01             41 .gitignore
2020/02/10 12:01            466 package.json
2020/02/10 12:01           6,711 StartupServer.js
2020/02/10 12:01 <DIR>          wwwroot
                3 个文件          7,218 字节
                3 个目录 43,473,178,624 可用字节

C:\Users\zhangxm-vm\Documents\code\wps\oaassist\server>_
```

5. 执行 `npm install` 命令，初始化Demo的模拟服务端：

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.18363.418]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Windows\system32>cd c:\Users\zhangxm-vm\Documents\code\wps\oaassist\server

c:\Users\zhangxm-vm\Documents\code\wps\oaassist\server>npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN oaassist_demo@1.0.0 No repository field.

added 64 packages from 43 contributors and audited 143 packages in 6.886s
found 0 vulnerabilities

c:\Users\zhangxm-vm\Documents\code\wps\oaassist\server>
```

6. 执行 `node StartupServer.js` 启动服务端：

```
管理员: 命令提示符 - node StartupServer.js
Microsoft Windows [版本 10.0.18363.418]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Windows\system32>cd c:\Users\zhangxm-vm\Documents\code\wps\oaassist\server

c:\Users\zhangxm-vm\Documents\code\wps\oaassist\server>npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN oaassist_demo@1.0.0 No repository field.

added 64 packages from 43 contributors and audited 143 packages in 6.886s
found 0 vulnerabilities

c:\Users\zhangxm-vm\Documents\code\wps\oaassist\server>node StartupServer.js
2020年2月10日 14:6:50 启动本地web服务(http://127.0.0.1:3888)成功!
```

7. 打开 `oaassist` 目录下的 `demo.html` 文件，开始基础环境检测并体验Demo：

打开 `demo.html` 文件后，默认开始检测服务端运行情况，之后检测本地WPS客户端的安装情况，以Windows为例，安装企业版或个人版，皆可体验。



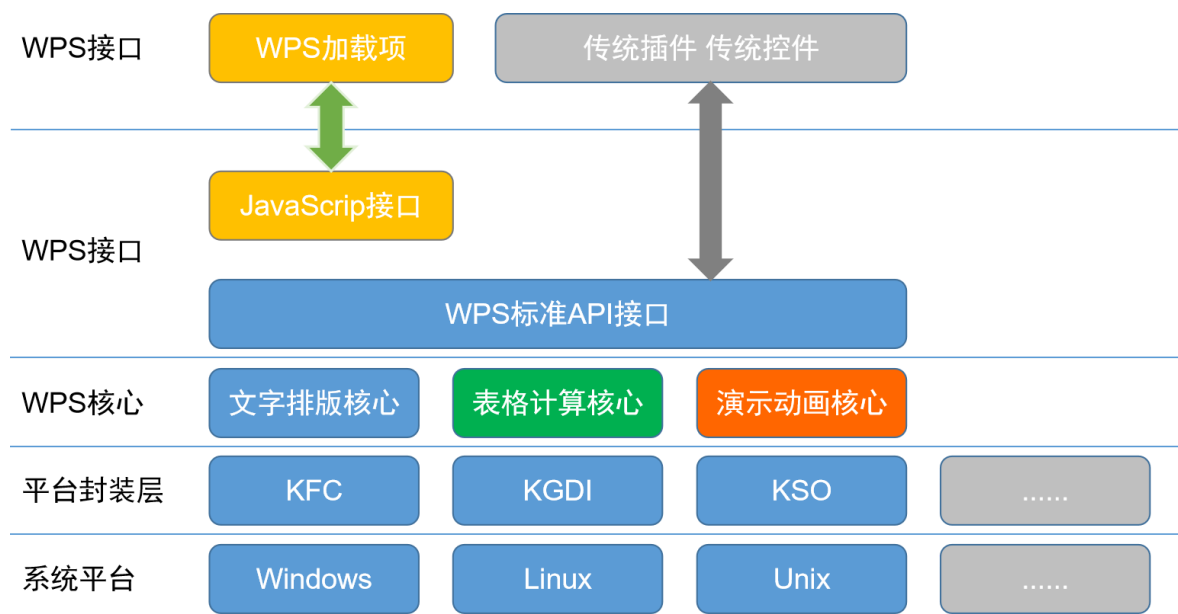
8. 点击开始体验后，跳转到的界面即是模拟的业务系统操作页面，此页面的功能，可以模拟通过前端页面调起本地WPS客户端，在WPS客户端完成文档编辑后，通过WPS加载项将数据与服务端同步的使用场景。



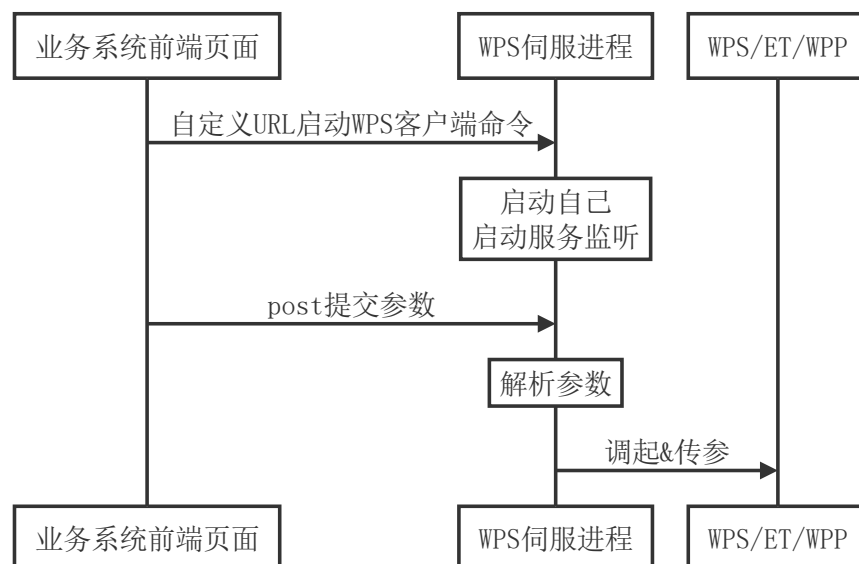
WPS加载项的运行原理

WPS加载项技术架构

- WPS加载项通过WPS JavaScript接口使用WPS标准API接口
- WPS JavaScript接口是基于WPS标准API接口，可有效降低开发人员学习成本
- WPS标准API接口全面兼容Office接口



- 前端页面启动WPS客户端并加载OA助手的过程

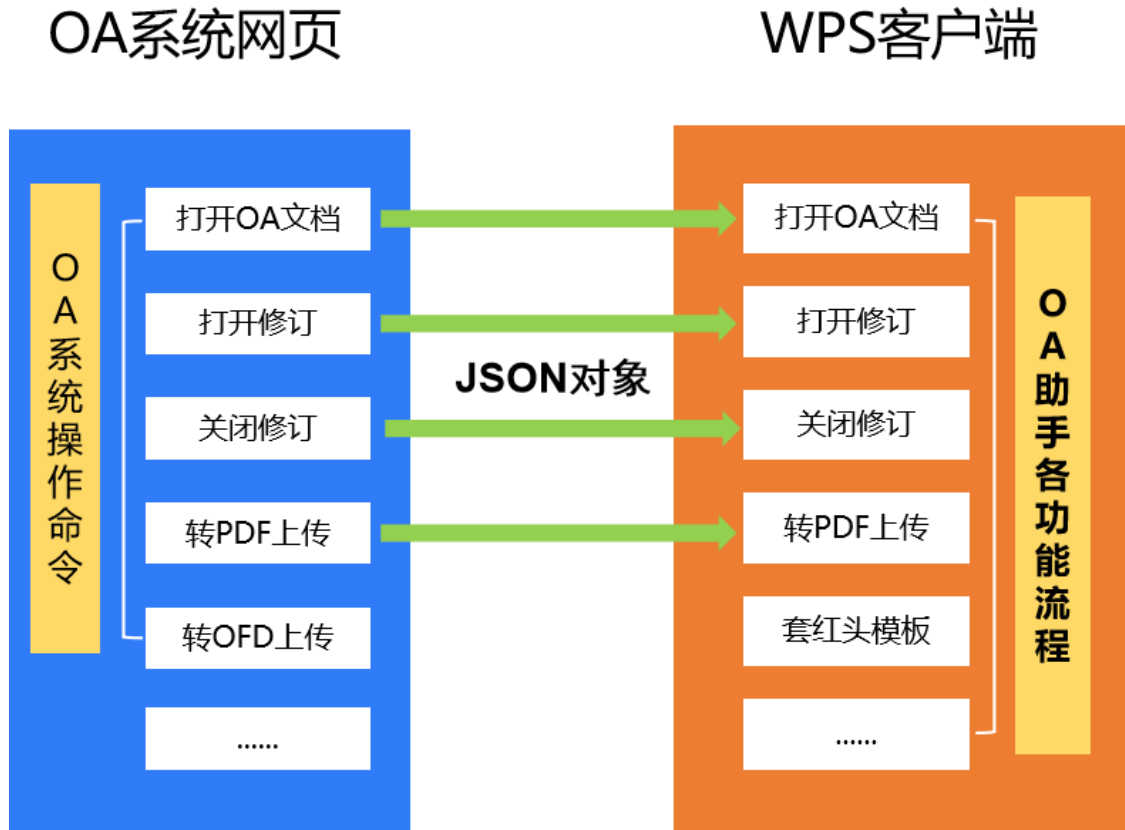


- WPS伺服进程当前机制是活动的进程有且仅有一个
- WPS伺服进程提供HTTP和HTTPS监听服务，分别监听58890和58891端口，根据业务系统的调用协议，选用不同的端口
- WPS伺服进程名称：wpsoffice

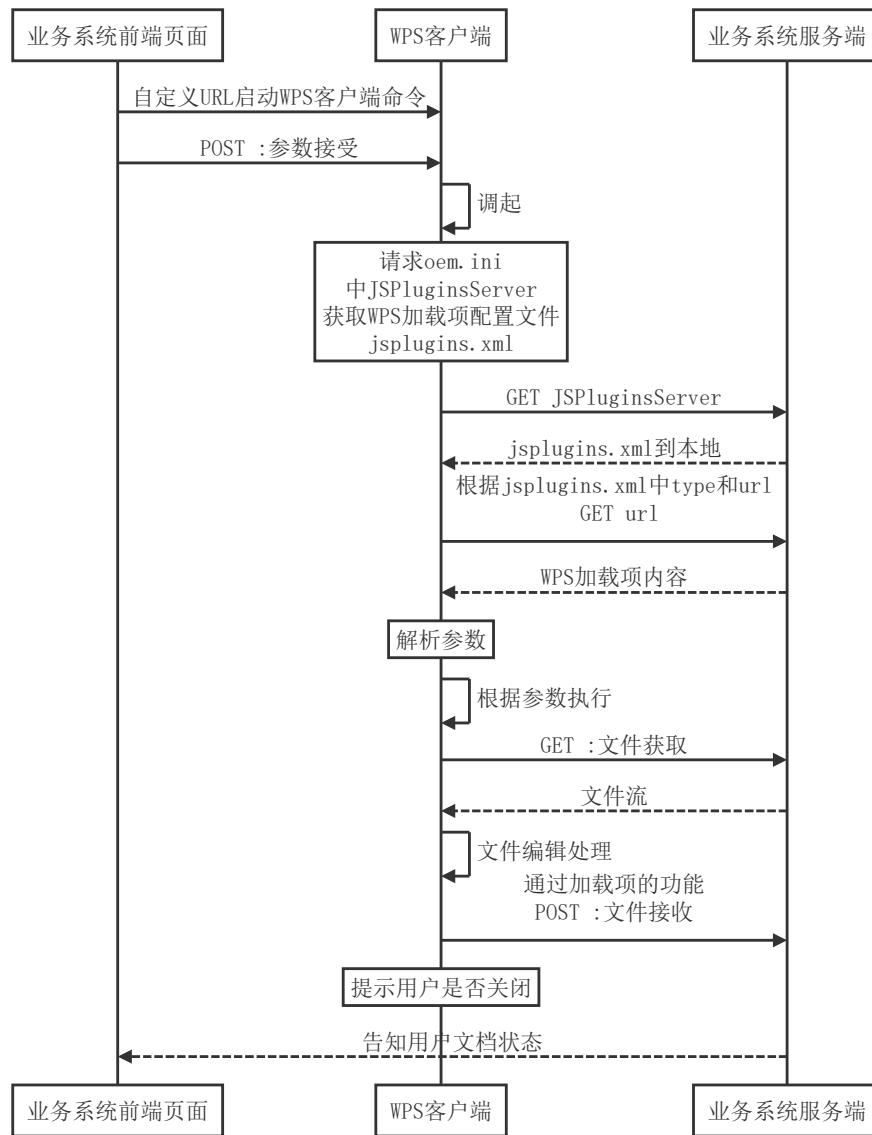
OA助手——WPS加载项的一种应用

- OA助手是OA系统与WPS客户端进行协同的桥梁
- 本Demo提供的样例代码中，实现了OA系统中与Office客户端交互的常用业务逻辑处理流程，开发者可基于此自由扩展
- OA助手不需要依赖浏览器控件，具有跨平台、跨浏览器、可扩展性强、系统健壮和部署升级便利的特点

- OA助手实现的OA系统网页和WPS客户端交互原理是利用自定义URL协议，如下图：



- 所有浏览器都支持自定义URL协议
- OA系统与OA助手通过自定义URL协议交互
- OA系统页面与WPS客户端交换JSON数据
- JSON对象的内容由OA系统与OA助手协商
- WPS客户端根据JSON数据加载OA助手相应功能
- 业务系统集成WPS加载项时，业务系统和WPS客户端的典型交互关系



WPS加载项正常运行时WPS客户端的必要配置

1. oem.ini文件

- 这是WPS客户端的配置文件，所在位置在WPS客户端安装目录的 `office6\cfgs` 下，以Windows客户端举例，WPS客户端安装在 `C:\Program Files (x86)\kingsoft\WPS Office\11.8.2.8748`，则 `oem.ini` 的路径即是 `C:\Program Files (x86)\kingsoft\WPS Office\11.8.2.8748\office6\cfgs\oem.ini`
- 在Demo中，通过在本机运行node服务端，快捷的修改了本机的WPS客户端 `oem.ini` 文件；**在生产环境中，此 `oem.ini` 需将 `JSPluginsServer` 按照生产环境的配置好后再做WPS客户端打包**
- 此文件对WPS加载项运行必须设置的配置项如下：

```

1  [Support]
2  JsApiPlugin = true
3  JsApiShowWebDebugger = true
4
5  [Server]
6  JSPluginsserver = http://127.0.0.1:3888/jsplugins.xml
  
```

- JsApiPlugin, 设置为 `true` 表示启用WPS加载项

- JsApiShowWebDebugger, 设置为 true 表示在WPS加载项的工具栏上显示 打开JS调试器 的按钮, 如下图。推荐在开发阶段此配置设置为true, 正式生产环境中设置为false



- JSPluginsServer, OA系统通过自定义浏览器协议调起WPS客户端时, WPS客户端会通过此配置项获取WPS加载项的配置文件

2. jsplugins.xml文件

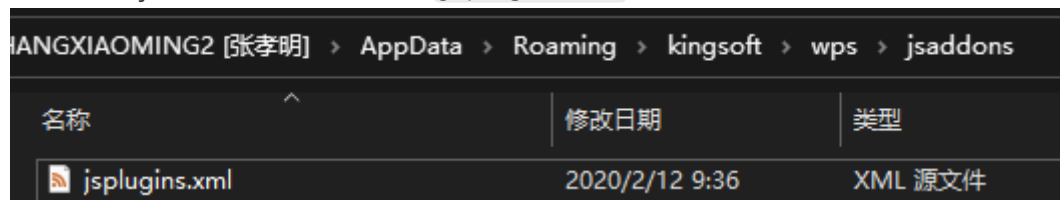
- 这是WPS加载项在客户端的配置文件, 以Windows环境距离, 位置在 %appdata%\kingsoft\wps\jsaddons 下
- WPS加载项运行有两种方式, 在线和离线。在线方式表示自定义功能区、任务窗格和Web对话框的数据文件存储在服务端, WPS客户端启动后通过网络请求获取; 离线方式则表示这些数据会在第一次加载时将数据拉取到本地, 之后每一次启动将对比本地和xml文件中定义版本号, 不一致将进行更新。
- 在线方式配置

```

1 <jsplugins>
2   <jspluginonline name="EtOAAssist" type="et"
   url="http://127.0.0.1:3888/plugin/et"/>
3   <jspluginonline name="wpsOAAssist" type="wps"
   url="http://127.0.0.1:3888/plugin/wps"/>
4   <jspluginonline name="wppOAAssist" type="wpp"
   url="http://127.0.0.1:3888/plugin/wpp"/>
5 </jsplugins>

```

此配置WPS客户端每次启动时, 通过 oem.ini 中 JSPluginsServer 配置项进行更新。
在线方式的jsaddons文件夹下, 只有 jsplugins.xml 文件, 如图:



- 离线方式配置

```

1 <jsplugins>
2   <jsplugin name="EtOAAssist" type="et"
   url="http://127.0.0.1:3888/plugins/v0.1/EtOAAssist.7z" version="0.1"
   />
3   <jsplugin name="wpsOAAssist" type="wps"
   url="http://127.0.0.1:3888/plugins/v0.1/wpsOAAssist.7z"
   version="0.1" />
4   <jsplugin name="wppOAAssist" type="wpp"
   url="http://127.0.0.1:3888/plugins/v0.1/wppOAAssist.7z"
   version="0.1" />
5 </jsplugins>

```


此配置WPS客户端每次启动时，通过 `oem.ini` 中 `JSPluginsServer` 配置项进行更新。
比在线配置多了 `version` 配置项，WPS客户端每次启动后，将对比此配置中 `version` 和本地文件的版本，不一致时将执行更新。



名称	修改日期	类型	大小
WpsOAAssist_0.1	2020/2/12 10:23	文件夹	
jsplugins.xml	2020/2/12 10:23	XML 源文件	1 KB

如上图，WPS加载项的离线文件夹是 `wpsOAAssist_0.1`，此规则为xml文件中，`name_version`，如果破坏此规则，会造成WPS加载项不可正常加载。

重要规则

- 在线方式和离线方式不可共存
- `type` 值对应着WPS三个组件，取值为 `et`，`wps` 和 `wpp`，不可出现两个相同的 `type` 配置

WPS加载项（本Demo）代码结构说明

服务端代码结构说明

- 此服务端模拟了OA业务系统与Office的交互场景，采用的Node.js做得实现，开发者亦可选择其他语言实现相应的功能。
- `StartupServer.js`：此文件是Node服务端入口文件，实现的主要功能包括
 - 以下请求，是支持WPS加载项运行时的必须实现的：
 - `/plugin/et/`、`/plugin/wps/`、`/plugin/wpp/`：在线方式下，获取加载项内容
 - `/Upload`：保存文件到服务端，需按照表单数据格式进行解析
 - `/Download`：从服务端获取指定文件
 - 以下请求，非必须实现：
 - `/WpsSetupTest`：WPS客户端环境检测
- `jsplugins.xml`：此文件是服务端的WPS加载项的配置文件，WPS客户端的 `oem.ini` 的 `JSPluginsServer` 需指向可获取此文件，每次WPS客户端启动都会获取此件来更新本地文件
- `resource/wps.html(et.html,wpp.html)`：此三个文件是OA助手应用场景示例网页
- `resource/js/wps.js(et.js,wpp.js)`：此三个文件是OA助手应用场景示例网页对应的js代码，开发者可将此文件的代码引入到自己的项目代码
 - `wpsStartup.Startup` 方法中，插件名称(name)必须与 `jsplugins.xml` 文件中的 `name` 保持一致，否则将无法加载WPS加载项
 - `wpsStartup.Startup` 方法中，加载项方法入口(func)必须与 `js/common/func_oastarter.js` 所定义的方法一致
- `resource/js/wpsstartup.js`：此文件封装了前端网页启动WPS客户端的自定URL协议的公用方法，开发者可将此文件引入到自己的项目工程中

WPS加载项代码结构说明

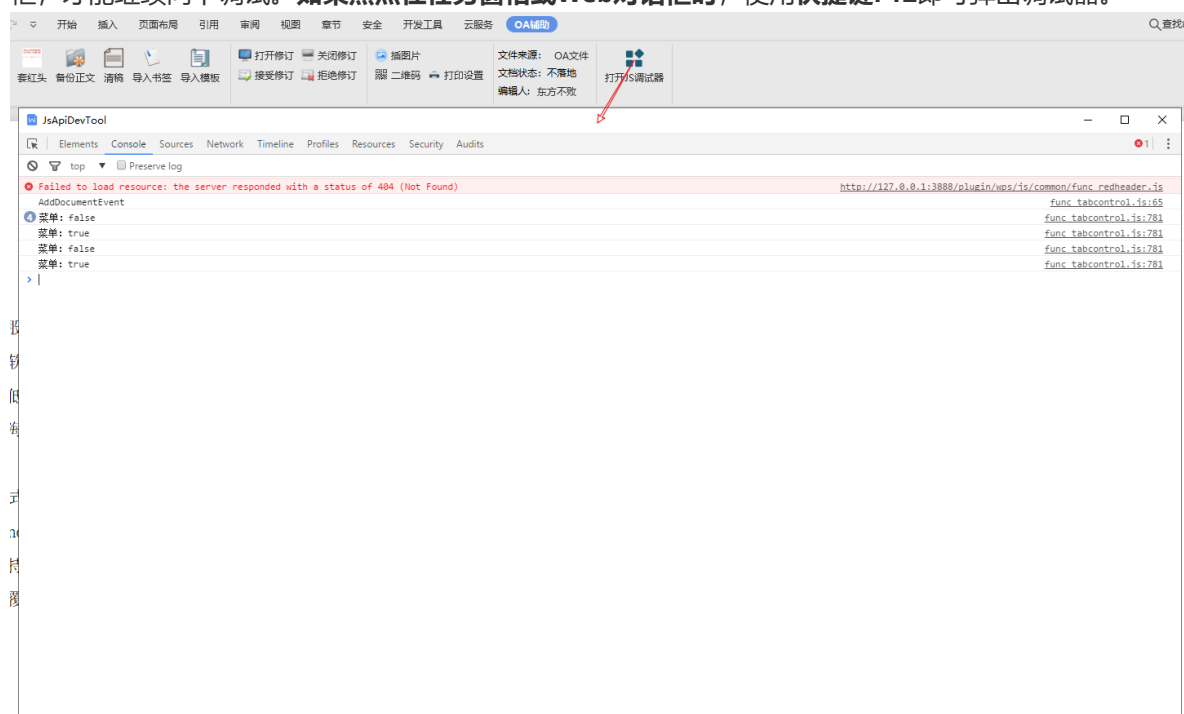
以WPsOAAssit为例说明

- `ribbon.xml`：此文件是自定义功能区的配置文件，按照MSO标准的规则定义自定义功能区的 `tabs`、`tab`、`group`、`box`、`button`这些常用的元素；通过command来控制WPS客户端的功能，例如保存，另存，输出为PDF，输出为OFD等，控制这些功能的显隐和自定义回调。

- index.html：加载项的前端入口文件，没有具体需要显示的内容，最关键的是引入 `main.js` 文件。
- js/main.js：引入所有的外部 JavaScript 文件。在这些 JavaScript 文件中通常包含了一系列用 JavaScript 实现的函数，这些函数与自定义功能区的功能一一对应，我们称之为接口函数。
- js/common/common.js：加载项常用的基础方法封装；如定义 `onShowDialog` 方法实现弹出 Web 对话框操作；
- js/common/enum.js：WPS API 常用枚举值，本加载项中自定义的枚举值；
- js/common/func_oastarter.js：集中处理来自 OA 的传入参数，web 页面调用 WPS 的方法入口也在此定义，如 `dispatcher` 这个方法的定义；
- js/common/func_tabcontrol.js：自定义功能区按钮所对应的功能实现；
- js/common/func_docProcess.js：针对文档原子操作的封装，暴露给 `func_tabcontrol` 做调用；
- js/common/func_docEvents.js：针对文档事件的功能实现，如文档保存前、后，文档打印后，文档新建后；暴露给 `func_tabcontrol` 做注册和卸载；通过 WPS 加载项事件能够对 WPS 应用程序发出的事件添加 JavaScript 方法进行处理。在通知型事件中，可以接收已经发生变化，比如通过 `WindowActivate` 事件，可以对文档的切换做一些功能性处理；在询问型事件中，可以控制是否继续执行当前操作，比如通过 `WorkbookBeforeClose` 事件，可以取消文档的关闭。
- taskpane.html：任务窗格的前端页面和处理逻辑；提供更丰富的内容展示；
- redhead.html：Web 对话框页面代表。结合事件监听，实现自由交互。

WPS加载项调试

WPS 加载项调试是对其中的一个网页单独进行的调试。调试时会弹出一个独立调试器对话框，除此之外和网页调试基本一致。可以在调试器的 Console 中直接查看任意的 API 属性和调用 API 方法。调试自动生成的 index.html 网页，使用快捷键 **ALT + F12**。注意调试过程中需要先关闭 alert 或其它同步弹框，才能继续向下调试。**如果焦点在任务窗格或 Web 对话框时，使用快捷键 F12 即可弹出调试器。**



FAQ

- `wps` 对象是在 WPS 客户端渲染网页时，将此对象注册到 Dom 树的 Windows 节点下，所有在 Web 对话框或任务窗格中，可以直接调用 `wps` 对象，无需声明。
- 在通过 WPS 宏编辑器进行了动作录制后，可以结合 `wps.chm` 的 API 手册，进行方法的查询。
- 通过 API 手册查询出相同的方法时，优先选择对象树层级少的方法，可以提高效率。
- 查看 WPS 接口中关于公文域的：`wps.WpsApplication().ActiveDocument.DocumentFields`

- 在需要调用wps接口的代码工程目录下，通过 `npm init` 建一个 `package.json`，然后通过执行一下 `npm install --save-dev wps-jsapi`，这样就在写代码时，关于wps相关接口就会有代码提示了，在此Demo中，到wps这个文件夹下，直接执行 `npm install` 即可安装代码提示插件。