**最详细的ionic cordova 插件配置文件 Plugin.xml 注解**

[ 星辰大海\_王](https://www.jianshu.com/u/57f8807cac77) 关注

0.2 2018.07.22 23:29\* 字数 2675 阅读 1139评论 0喜欢 2

**PS:目前利用前端技术开发跨平台App很火爆，最火的当属使用cordova框架开发App，另外最近很火的前端开发框架ionic访问原生的能力也是来自于cordova(对cordova进行了封装)。插件是它们的核心所在，无论我们要使用ionic还是直接使用cordova去开发一个App，都会用到框架提供的插件或自定义插件，所以掌握Plugin.xml的编写还是很重要滴~。**

**plugin.xml文件作用**

plugin.xml文件定义插件所需的结构和设置，它通过几个元素来提供有关插件的详细信息。

[cordova Plugin.xml 官网](http://cordova.axuer.com/docs/zh-cn/latest/reference/cordova-plugin-file/index.html)

现在开始详细说一下这些元素（标签）~~~

**顶级元素plugin**

该plugin元素是插件清单的顶级元素。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| xmlns(string) | 必需的，插件命名空间<http://apache.org/cordova/ns/plugins/1.0>。如果文档包含来自其他命名空间的XML，例如AndroidManifest.xml在Android的情况下要添加到文件中的标记，那么这些命名空间也应该包含在元素中。 |
| id(string) | 必需的，插件的npm样式标识符。 |
| version(string) | *必需的，* 插件的版本号。支持[Semver](http://semver.org/)语法。 |

例：

<plugin xmlns="http://apache.org/cordova/ns/plugins/1.0"

xmlns:android="http://schemas.android.com/apk/res/android"

id="my-plugin-id"

version="1.0.2">

**engines and engine**

plugin元素的子元素<engines>指定此插件支持的基于Apache Cordova的框架的版本。对于目标项目不满足引擎约束的任何插件，CLI将使用非零代码进行中止。如果不指定了标记，则CLI会尝试盲目地安装到指定的cordova项目目录中。

注意：在**Cordova 6.1.0+中**，指定平台，插件和CLI依赖项的推荐位置在插件的package.json中。有关 更多信息，请参阅[指定Cordova依赖项](https://cordova.apache.org/docs/en/latest/guide/hybrid/plugins/index.html#specifying-cordova-dependencies)。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| name(string) | Required, 引擎名称。以下是支持的默认引擎：cordova; cordova-plugman;cordova-android; cordova-ios; cordova-windows; cordova-osx; windows-os; android-sdk （返回安装的最高Android api级别）; windows-sdk （返回本机Windows SDK版本）; apple-xcode （返回xcode版本）; apple-ios （返回安装的最高iOS版本）; apple-osx （返回OSX版本）。除默认框架外，您还可以指定自定义框架。 |
| version(string) | Required，您的框架必须具有的版本才能安装。支持Semver语法。 |
| scriptSrc(string) | \*仅适用于自定义框架 \*, Required ,此脚本文件告诉plugman自定义框架的版本。理想情况下，此文件应位于插件目录的顶级目录中。 |
| platform(string) | *仅适用于自定义框架* Required ,您的框架支持的平台。您可以使用通配符\*来表示支持所有平台，使用管道字符指定多个，如`android。 |

例如:

<engines>

<engine name="cordova-android" version="=1.8.0" />

</engines>

引擎元素还可以使用“>”，“> =”等指定模糊匹配以避免重复，并在更新基础平台时减少维护。

<engines>

<engine name="cordova-android" version=">=1.8.0" />

</engines>

该<engine>标签默认支持所有的已经存在的主要的cordova平台。指定cordova引擎标记意味着任何平台上的所有Cordova版本都必须满足引擎版本属性。您还可以列出特定平台及其版本，以覆盖笼统的cordova引擎：

<engines>

<engine name="cordova" version=">=1.7.0" />

<engine name="cordova-android" version=">=1.8.0" />

<engine name="cordova-ios" version=">=1.7.1" />

</engines>

**自定义框架的例子：**

<engines>

<engine name="my\_custom\_framework" version="1.0.0" platform="android" scriptSrc="path\_to\_my\_custom\_framework\_version"/>

<engine name="another\_framework" version=">0.2.0" platform="ios|android" scriptSrc="path\_to\_another\_framework\_version"/>

<engine name="even\_more\_framework" version=">=2.2.0" platform="\*" scriptSrc="path\_to\_even\_more\_framework\_version"/>

</engines>

**name**

该name元素用于指定插件的名称。此元素（尚未）处理本地化。

例如：

<name>Foo</name>

**description**

该description元素用于指定插件的描述。此元素（尚未）处理本地化。

例如：

<description>Foo plugin description</description>

**author**

author元素的内容包含插件作者的名称。

例如：

<author>Foo plugin author</author>

**keywords**

keywords元素的内容包含逗号分隔的关键字以描述插件。

例子：

<keywords>foo,bar</keywords>

**license（许可、执照）**

此元素用于指定插件的许可证。

例如:

<license>Apache 2.0 License</license>

**asset**

此元素用于列出要复制到Cordova应用程序www目录中的文件或目录。任何嵌套在<platform>元素中的<asset>元素都指定特定于平台的web资源。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| src(string) | Required，相对于plugin.xml文档,文件或目录位于插件包中的位置。如果指定的src位置不存在文件，CLI将停止并撤消安装过程，发出有关冲突的通知，并使用非零代码退出。 |
| target(string) | Required，文件或目录应位于Cordova应用程序中相对于www目录的位置。如果目标位置已存在文件，CLI将停止并撤消安装过程，发出有关冲突的通知，并以非零代码退出。 |

例子：

<!-- a single file, to be copied in the root directory -->

<asset src="www/foo.js" target="foo.js" />

<!-- a directory, also to be copied in the root directory -->

<asset src="www/foo" target="foo" />

资源也可以指定到子目录。这将在www目录中创建js/experimental目录，除非已存在，并复制该new-foo.js文件并将其重命名为foo.js。

<asset src="www/new-foo.js" target="js/experimental/foo.js" />

**js-module**

大多数插件都包含一个或多个JavaScript文件。每个<js-module>标记对应一个JavaScript文件，并阻止插件的使用者必须为每个文件添加<script>标记。不要使用cordova.define包装文件，因为它是自动添加的。该模块包含在一个闭包中，带有模块，导出，以及要求在范围内，就像普通的AMD模块一样。嵌套在<platform>元素中的<js-module>元素声明特定平台绑定的JavaScript模块。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| src(string) | Required，引用插件目录中相对于plugin.xml文件的文件。如果src未解析为现有文件，CLI将停止并撤消安装，发出问题通知，并以非零代码退出。 |
| name(string) | Required，提供模块名称的最后一部分。它通常可以是你喜欢的任何东西，只有你想在你的JavaScript代码中使用cordova.require导入插件的其他部分才有意义。一个<js-module>中的模块名称<js-module>是您的插件的id。 |

例如：

使用下面的示例安装插件时，将socket.js复制到www/plugins/my-plugin-id/socket.js并添加为条目www/cordova\_plugins.js。在加载时，代码cordova.js使用XHR读取每个文件并将<script>标记注入HTML。

<js-module src="socket.js" name="Socket">

</js-module>

同样对于此示例，如果插件ID为chrome-socket，则模块名称将为chrome-socket.Socket。

**clobbers（覆盖~个人理解）**

<js-module>允许元素内使用。用于指定window插入module.exports的对象下的命名空间。你可以拥有任意多的<clobbers>，相应的window上创建的对象将不可用。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| target(string) | 插入module.exports的命名空间。 |

例：

<js-module src="socket.js" name="Socket">

<clobbers target="chrome.socket" />

</js-module>

这里module.exports作为window.chrome.socket被插入到window对象中。

**merges**

<js-module>允许元素内使用。用来指定在哪里module.exports获取与任何现有的价值合并window对象的命名空间。如果已经存在，模块的版本取代原来的。你可以有很多的merges只要你喜欢。创建在window上的任何对象不可用。

| **属性(类型)** | **描述** |

| --- | --- |

| target(string) | module.exports合并到的命名空间。 |

例：

<js-module src="socket.js" name="Socket">

<merges target="chrome.socket" />

</js-module>

这里module.exports与任何现有值合并在一起window.chrome.socket。

**runs**

<js-module>允许元素内使用。它意味着您的代码应该使用cordova.require，但不能安装在window对象上。这在初始化模块，附加事件处理程序或其他方面很有用。您最多只能有一个<runs/>标签。需要注意的是，包括一个<runs/>带有<clobbers/>或者<merges/>是多余的，因为它们也cordova.require你的模块。

例如：

<js-module src="socket.js" name="Socket">

<runs/>

</js-module>

**dependency**

该<dependency>标签允许你指定在其当前插件依赖其他插件。插件由其唯一的npm id或github url引用。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| id(string) | 提供插件的ID。 |
| url(string) | 插件的URL。这应该引用一个git存储库，CLI试图克隆它。 |
| commit(string) | 这是git通过git checkout以下方式引用：分支或标记名称（例如master，0.3.1），或提交哈希（例如975ddb228af811dd8bb37ed1dfd092a3d05295f9）。 |
| subdir(string) | 指定目标插件依赖项作为git存储库的子目录存在。这很有用，因为它允许存储库包含几个相关的插件，每个插件都单独指定。如果你设置url一个的<dependency>标签"."，并提供一个subdir，依赖插件是从同一个本地或远程的Git仓库作为指定父插件安装<dependency>标签。请注意，subdir始终指定相对于git存储库根目录的路径，而不是父插件。即使您使用本地路径直接安装插件，也是如此.CLI找到git存储库的根目录，然后从那里找到另一个插件。 |
| version(string) | 该插件的版本取决于。支持Semver语法。 |

例:

<dependency id="cordova-plugin-someplugin" url="https://github.com/myuser/someplugin" commit="428931ada3891801" subdir="some/path/here" />

<dependency id="cordova-plugin-someplugin" version="1.0.1">

**platform**

标识具有关联原生代码或需要修改其配置文件的平台。使用此规范的工具可以识别支持的平台并将代码安装到Cordova项目中。没有<platform>标签的插件被假定为仅仅是JavaScript，因此可以安装在任何和所有平台上。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| name(string) | Required， 允许值：ios，android，windows，browser，osx 标识支持的平台，将元素的子项与该平台相关联。 |

例如：

<platform name="android">

<!-- android-specific elements -->

</platform>

**source-file**

标识应安装到项目中的可执行源代码。

|  |  |  |
| --- | --- | --- |
| **属性(类型)** | | **描述** |
| src(string) | | Required， 位置相对于plugin.xml文件。如果找不到src文件，CLI将停止并撤消安装，发出有关问题的通知，并以非零代码退出。 |
| target-dir(string) | | Required， 允许值：ios，android，windows，browser，osx 标识支持的平台，将元素的子项与该平台相关联。 |
| framework(boolean)苹果平台 | | 默认值：false |
| 如果设置为true，则还将指定的文件作为框架添加到项目中。 |  | |
| compiler-flags(string) 苹果平台 | | 如果设置，则为特定源文件指定编译器标志。 |

例子：

<!-- android -->

<source-file src="src/android/Foo.java" target-dir="src/com/alunny/foo" />

<!-- ios -->

<source-file src="src/ios/CDVFoo.m" />

<source-file src="src/ios/someLib.a" framework="true" />

<source-file src="src/ios/someLib.a" compiler-flags="-fno-objc-arc" />

**header-file**

这类似于<source-file>元素，但专门针对iOS和Android等平台，用于区分源文件，头文件和资源。Windows不支持此功能。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| src(string) | Required， 位置相对于plugin.xml文件。如果找不到src文件，CLI将停止并撤消安装，发出有关问题的通知，并以非零代码退出。 |
| target-dir(string) | 应该复制文件的目录，相对于Cordova项目的根目录。 |

例：

<header-file src="CDVFoo.h" />

**resource-file**

这类似于<source-file>元素，但专门用于区分源文件，头文件和资源的iOS和Android等平台。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| src(string) | Required， 位置相对于plugin.xml文件。如果找不到src文件，CLI将停止并撤消安装，发出有关问题的通知，并以非零代码退出。 |
| target(string) | 你的将要被复制的目录中的文件路径。 |
| arch(string) windows平台 | 允许值：x86，x64或ARM。指示仅在为指定体系结构构建时才包含该文件。 |
| device-target windows平台 | 设备对象 允许值:( win或windows），phone或all。指示仅在为指定的目标设备类型构建时才包含该文件。 |
| versions windows平台 | 指示仅在为与指定版本字符串匹配的版本构建时才应包含该文件。值可以是任何有效的节点语义版本范围字符串。 |
| reference windows平台 | 指示应从src引用该文件，而不是将其复制到目标目标。该文件将显示在Visual Studio中，文件名由target指定，但是将指向相应的src，具体取决于体系结构。 |

例子：

For Android:

<resource-file src="FooPluginStrings.xml" target="res/values/FooPluginStrings.xml" />

For Windows:

<resource-file src="src/windows/win81/MobServices.pri" target="win81\MobServices.pri" device-target="windows" versions="8.1" arch="x64"/>

<!-- Example of referencing -->

<resource-file src="x86/foo.dll" target="foo.dll" arch="x86" reference="true" />

<resource-file src="x64/foo.dll" target="foo.dll" arch="x64" reference="true" />

**注意：**target应使用反斜杠以避免在Visual Studio中DEP2100部署错误。

**config-file**

标识要修改的基于XML的配置文件，即：该文档中应该被修改的位置，以及应修改的内容。已使用此元素测试修改的两种文件类型是xml和plist文件。该config-file元素仅允许您将新子项附加到XML文档树。子项是要插入目标文档的XML literals。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| target(string) | 要修改的文件以及相对于Cordova项目根目录的路径。如果指定的文件不存在，该工具将忽略配置更改并继续安装。target可以包含wildcard（\*）元素。在这种情况下，CLI递归搜索项目目录结构并使用第一个匹配项。在iOS上，配置文件相对于项目目录root的位置未知，因此指定config.xml解析为的目标cordova-ios-project/MyAppName/config.xml。 |
| after(string) | 接受兄弟姐妹的优先列表之后，添加 XML 片段。这是有用的在需要指定像XML元素的严格排序的文件中比如：[this](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff769509%28v=vs.105%29.aspx#BKMK_EXTENSIONSelement)。 |
| device-target(string) | windows。 允许值：win，phone，all。适用于在影meta-name package.appxmanifest时，此属性指示仅在为指定的目标设备类型构建时才应修改该文件。 |
| versions (string) | windows。 适用于在影meta-name package.appxmanifest时，此属性表示只应针对与指定版本字符串匹配的版本，更改特定Windows版本的应用程序清单。值可以是任何有效的节点语义版本范围的字符串。 |

例如:

For XML（Android等）:

<config-file target="AndroidManifest.xml" parent="/manifest/application">

<activity android:name="com.foo.Foo" android:label="@string/app\_name">

<intent-filter>

</intent-filter>

</activity>

</config-file>

For plist（iOS）:

<config-file target="\*-Info.plist" parent="CFBundleURLTypes">

<array>

<dict>

<key>PackageName</key>

<string>$PACKAGE\_NAME</string>

</dict>

</array>

</config-file>

For windows-specific attributes:

<config-file target="package.appxmanifest" parent="/Package/Capabilities" versions="<8.1.0">

<Capability Name="picturesLibrary" />

<DeviceCapability Name="webcam" />

</config-file>

<config-file target="package.appxmanifest" parent="/Package/Capabilities" versions=">=8.1.0" device-target="phone">

<DeviceCapability Name="webcam" />

</config-file>

上面的示例将设置8.1之前的平台（特别是Windows 8）以要求webcam设备功能和picturesLibrary常规功能，并仅将webcam设备功能应用于为Windows Phone构建的Windows 8.1项目。Windows桌面8.1系统未经修改。

**plugins-plist**

指定要附加到iOS Cordova项目中AppInfo.plist 文件的键和值。**这已经过时**，因为它仅适用于cordova-ios 2.2.0及更低版本。将**使用<config-file>标签**用于较新版本的Cordova。

**lib-file**

类似source, resource, and header files，但特别适用于使用用户生成的库的BlackBerry 10等平台。对于Windows平台，在生成的Windows项目文件中该<lib-file>元素允许包含an <SDKReference>。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| src(string) | Required， 位置相对于plugin.xml文件。如果找不到src文件，CLI将停止并撤消安装，发出有关问题的通知，并以非零代码退出。对于Windows，它指示要包含的SDK的名称（将用作Include生成<SDKReference>元素的属性的值）。 |
| arch(string) | 真机或者simulator，构建已经被bulid 过的.so文件。对于Windows，它表示<SDKReference>仅包含在编译特定的包时。支持的值是x86，x64或ARM。 |
| device-target(string) | windows。 允许值：win，phone，all。它表示<SDKReference>仅包含在编译特定目标设备的包时。 |
| versions (string) | windows。 它表示<SDKReference>仅包含在编译特定版本的包时。值可以是任何有效的节点语义版本范围的字符串。 |

例如：

<lib-file src="src/BlackBerry10/native/device/libfoo.so" arch="device" />

<lib-file src="src/BlackBerry10/native/simulator/libfoo.so" arch="simulator" />

For Windows:

<lib-file src="Microsoft.WinJS.2.0, Version=1.0" arch="x86" />

<lib-file src="Microsoft.WinJS.2.0, Version=1.0" versions=">=8.1" />

<lib-file src="Microsoft.WinJS.2.0, Version=1.0" target="phone" />

<lib-file src="Microsoft.WinJS.2.0, Version=1.0" target="win" versions="8.0" arch="x86" />

**framework**

标识插件所依赖的框架（通常是OS /平台的一部分）。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| src(string) | Required， 位置相对于plugin.xml文件。如果找不到src文件，CLI将停止并撤消安装，发出有关问题的通知，并以非零代码退出。对于Windows，它指示要包含的SDK的名称（将用作Include生成<SDKReference>元素的属性的值）。 |
| src(string) | Required，系统框架的名称或被包含的作为插件的一部分的系统框架的相对路径。 |
| custom(boolean) | 指示框架是否包含在插件文件中。 |
| weak(boolean) | Default: false ， 指示框架是否应该弱链接。 |
| type(string) | Default: .； 设置包含要添加引用的子项目的目录的相对路径。默认值“.”表示应用程序项目。 |
| parent(string) | 指示要添加的框架类型。 |
| arch(string) | windows 。允许值：x86，x64或ARM。指仅在编译特定包时才包含框架。 |
| device-target(string) | windows。 允许值：( win或windows），phone或all。指仅在编译特定设备的包时才包含框架。 |
| versions (string) | windows。 它表示框架仅包含在编译特定版本的包时。值可以是任何有效的节点语义版本范围的字符串。 |
| target-dir (string) | 表示框架中的应该被复制子目录。实际上，在不同的芯片架构或设备目标中包含不同的框架版本但具有相同名称时，这一点非常重要。这允许您为每个框架版本指定不同的子文件夹，以便它们不会相互重叠。 |

例如：

For iOS:

<framework src="libsqlite3.dylib" />

<framework src="social.framework" weak="true" />

<framework src="relative/path/to/my.framework" custom="true" />

在Android上（截至cordova-android@4.0.0），框架标记用于包含Maven依赖项，或包含捆绑的库项目。

<!-- Depend on latest version of GCM from play services -->

<framework src="com.google.android.gms:play-services-gcm:+" />

<!-- Depend on v21 of appcompat-v7 support library -->

<framework src="com.android.support:appcompat-v7:21+" />

<!-- Depend on library project included in plugin -->

<framework src="relative/path/FeedbackLib" custom="true" />

Framework还可用于将自定义.gradle文件包含在主项目的.gradle文件中：

<framework src="relative/path/rules.gradle" custom="true" type="gradleReference" />

在Windows下使用，custom='true'并且type='projectReference'将添加项目引用，将被添加到这些cordova项目的编译+链接步骤。这基本上是当前“自定义”框架可以针对多个体系结构的唯一方式，因为它们显式构建为依赖通过cordova应用程序的引用。

<framework src="path/to/project/LibProj.csproj" custom="true" type="projectReference"/>

使用这些Windows特定属性的示例：

<framework src="src/windows/example.dll" arch="x64" />

<framework src="src/windows/example.dll" versions=">=8.0" />

<framework src="src/windows/example.vcxproj" type="projectReference" target="win" />

<framework src="src/windows/example.vcxproj" type="projectReference" target="all" versions="8.1" arch="x86" />

<framework src="src/windows/example.dll" target-dir="bin/x64" arch="x64" custom="true"/>

**info**

向用户提供的其他信息。当您需要无法轻松实现或超出CLI范围的额外步骤时，这非常有用。CLI安装插件时会打印出此标记的内容。

例：

<info>

You need to install \_\_Google Play Services\_\_ from the `Android Extras` section using the Android SDK manager (run `android`).

You need to add the following line to the `local.properties`:

android.library.reference.1=PATH\_TO\_ANDROID\_SDK/sdk/extras/google/google\_play\_services/libproject/google-play-services\_lib

</info>

**hook**

表示您在发生特定操作时将由Cordova调用的自定义脚本（例如，在添加插件或调用平台准备逻辑之后）。当您需要扩展默认的Cordova功能时，这非常有用。有关详细信息，请参阅 [Hooks Guide](http://cordova.axuer.com/docs/zh-cn/latest/guide/appdev/hooks/index.html)。

例：

<hook type="after\_plugin\_install" src="scripts/afterPluginInstall.js" />

**uses-permission**

在某些情况下，插件可能需要根据目标应用程序进行配置更改。例如，要在Android上注册C2DM，其包ID为my-app-id的应用程序将需要一个权限，比如：

<uses-permission android:name="my-app-id.permission.C2D\_MESSAGE"/>

在这种情况下，从plugin.xml文件中插入的内容未提前知道，变量可以用美元符号表示，后跟一系列大写字母，数字或下划线。对于上面的示例，该plugin.xml文件将包含此标记：

<uses-permission android:name="$PACKAGE\_NAME.permission.C2D\_MESSAGE"/>

CLI使用指定值替换变量引用，如果未找到则替换空字符串。可以检测变量引用的值（在这种情况下，从AndroidManifest.xml文件中）或由工具的用户指定; 确切的过程取决于特定的工具。

Plugman可以请求用户指定插件的必需变量。例如，可以将C2M和Google Maps的API密钥指定为命令行参数：

plugman --platform android --project /path/to/project --plugin name|git-url|path --variable API\_KEY=!@CFATGWE%^WGSFDGSDFW$%^#$%YTHGsdfhsfhyer56734

应保留某些变量名称，例如$PACKAGE\_NAME。这是包的反向域样式唯一标识符，对应CFBundleIdentifier于iOS上或package属性的顶级manifest元素在AndroidManifest.xml文件中。

**preference**

如上一节所示，有时插件可能要求用户指定其变量的值。要使这些变量成为必需项，<platform>标记需要包含<preference>标记。CLI检查是否传入了这些必需的首选项。如果没有，它应该警告用户如何传入变量并使用非零代码退出。

|  |  |
| --- | --- |
| **属性(类型)** | **描述** |
| name(string) | Required，变量的名称。 |
| default(string) | 变量的默认值。如果存在，将使用其值，并且如果用户未输入任何值，则不会发出错误。 |

例如:

<preference name="API\_KEY" default="default-value" />

\*好啦，到这里终于把Plugin.xml文件给解析了一遍，希望对读者有帮助，下一节我们将在ionic项目中创建一个[cordova自定义插件](https://www.jianshu.com/p/c10593d41b8a)。