# Sencha Cmd 6升级指南

本指南旨在帮助开发人员使用Sencha Cmd从Ext JS 5.x或Sencha Touch 2.4.x迁移到Ext JS 6.0.x。尽管在这个版本中有一些重要的变化，但是我们已经尽力使升级过程尽可能的平滑。

在深入讨论之前，需要提醒的是，这个指南是在如下前提下的。

* 您的应用程序是使用Ext JS 5.x或Sencha Touch 2.4.x构建的
* 您的应用程序是使用我们推荐的MVC模式布局的
* 你的应用目前正在用Sencha Cmd构建

我们认识到并不是所有的客户都有能力运行最新版本的框架和Sencha Cmd。

时间限制、最后上线期限和支持许可常常决定企业中使用的版本。

如果您的应用程序是基于Sencha框架的旧版本，那么您可以参考与这些发行版相关的升级指南，作为深入学习的起点。

例如，Sencha Cmd 5升级指南。

## 安装程序的变化

使用Sencha Cmd 6，我们引入了新的安装程序，允许非管理员/root用户安装应用程序。

我们还在安装程序中包含了Sencha Cmd运行需要的Java运行时环境(JRE)，所以你不需要单独下载和安装。

如果您已安装了Sencha Cmd的早期版本，那么您可能会在升级中遇到一些问题。

### Windows PATH

使用Sencha Cmd的老版本的Windows用户的PATH变化的一个副作用是，您将看不到新版本的Sencha在您的PATH前面。

这是很自然的，因为前面的老版本路径 ，会覆盖掉后面的新版本的路径。

要管理你的路径，请使用控制面板:

（译者注：在环境变量栏里，选择系统级和用户级的Path环境变量。系统级的环境变量会在用户级的环境变量前面被查寻。）

在PATH环境变量的最终值中，系统Path字符串出现在用户路径字符串之前。

以前版本的Sencha Cmd安装在系统路径中，因此它们将“隐藏”新版本的路径。

注：操作系统会按照先寻找系统路径，再找用户路径的方式寻找Sencha Cmd。所以，在系统路径中找到了就不再再继续向下找了。

### Mac OS X / Linux

在非windows平台上，安装程序现在通过修改/.bashprofile只在Path上添加一个位置，

export PATH=~/bin/Sencha/Cmd:$PATH

在以前的安装程序中，您可能会发现多个入口，例如:

export PATH=~/bin/Sencha/Cmd/5.1.1.39:$PATH

export PATH=~/bin/Sencha/Cmd/5.1.2.52:$PATH

export PATH=~/bin/Sencha/Cmd/5.1.3.61:$PATH

新的安装程序现在维护了一个符号链接“/bin/sencha/cmd/sencha”，它指向最新安装的版本。

# 迁移过程

## 在开始之前

### 清除状态

迁移过程的第一步是确保在您的源代码控制系统中没有未提交的更改。

不建议在未提交更改时启动升级。

这将让你更容易地回顾Sencha Cmd的变化，这样你就可以确定你所做的任何更改未受影响。

### 安装合并工具(推荐)

在升级过程中，Sencha Cmd可能需要对一些您可能已经编辑过的文件进行更改。

就像任何这样的场景一样，Sencha Cmd需要更新和您可能已经编辑了的代码相同，由此产生了合并冲突。

好消息是，就像版本控制一样，有一些工具可以帮助解决这些合并冲突。

Sencha Cmd可以使用任何可以从命令行运行的可视化合并工具(这几乎是所有的)。

这一步是可选的，但高度推荐，因为它将使处理升级变得更加简单。

下面是一些流行的选择(一些免费的，一些商业的):

* [p4merge](http://www.perforce.com/product/components/perforce-visual-merge-and-diff-tools)
* [SourceGear](http://www.sourcegear.com/diffmerge/index.html)
* [kdiff3](http://sourceforge.net/projects/kdiff3/files/kdiff3/)
* [Syntevo SmartSynchronize 3](http://www.syntevo.com/smartsynchronize/)
* [TortoiseMerge](http://tortoisesvn.net) - (part of TortoiseSVN)
* [AraxisMerge](http://www.araxis.com/merge/)

在下一步中，我们将配置Sencha Cmd使用您的首选合并工具。

### 升级Sencha Cmd

接下来，您需要获取Sencha Cmd 6的最新版本。

您可以从Sencha网站下载Sencha Cmd 6（http://www.sencha.com/products/sencha-cmd/）。

安装Sencha Cmd并重新启动您的终端。

如果您有一个较老版本的Sencha Cmd，您可能需要纠正您的PATH环境变量(见上文)。

注意:如果你安装了以前的Cmd，它不会取代它，但是在运行Cmd软件时它会优先考虑旧版本。

有关详细信息,请参阅上面。

### 配置合并工具

如果你选择了一个合并工具，我们需要配置Sencha Cmd来使用这个合并工具。

要这样做，您需要向配置文件中添加两个属性:

cmd.merge.tool

cmd.merge.tool.args

我们可以把这些在“sencha.cfg”文件中进行配置，该文件位于Sencha Cmd安装目录。

另外，这两个属性可以以独立于版本的方式设置，这样您的配置参数将适用于Sencha Cmd的所有版本。

详情请见“sencha.cfg” 文件的尾部。

### Appl Upgrade

我们已经准备好开始升级了。

只需从应用程序的根文件夹中运行该命令，就可以完成升级操作:

sencha app upgrade path/to/ext6

您应该会看到少量的绿色文本，让您知道您的应用程序已经成功地升级了。

如果您正在升级Ext JS 4.x，您应该参考Sencha Cmd 5升级指南的微加载器（Microloader）和构建属性（Building Properties）部分。

同样地，如果你在使用Cordova或PhoneGap，你也应该参考这些部分。

### 检查源代码

一旦应用程序升级完成，并且您已经解决了报告的任何可能的合并冲突，现在可能是通过源代码控制查看代码差异的好时机。

如果一切看起来都正常，那就最好提交代码，重新回到一个整洁的状态。

### 清洁和构建

下一步是从应用程序根目录快速构建一个应用:

sencha app build --clean development

请注意，--clean开关将从构建输出文件夹中删除所有文件。

这将确保在以前的构建中没有任何的缓存文件，但是为了安全起见，我们希望在继续操作之前确保所有的更改都是提交的。

您可能需要做进一步的工作来使应用程序可构建，特别是当您使用Sencha Touch时。

如果是这样，上面的构建将显示错误。

一旦所有的构建错误被纠正，构建命令就会成功。

### 检查

一旦所有的构建错误都得到了解决，就该提交代码更改了。

Cmd的升级阶段到此完成。由于框架的升级，可能需要更多的更改，但是最好从一个可构建的应用程序开始。

### 监视

要使用Sencha Fashion(我们的新主题编译器)实现的新的实时更新功能，应用程序必须由app watch托管。

使用watch，您现在可以在自己本地主机上的web服务器的或在http://localhost:1841上查看应用程序，这是我们自动为您定制的服务器。

这个web服务器是app watch默认的服务器。

在Sencha Cmd 5之前，您必须单独运行Sencha web start来使用Sencha Cmd web服务器。

您可以使用如下这些属性调整web服务器配置(它们的默认值如下所示):

build.web.port=1841

build.web.root=${workspace.dir}

## 应用程序更新

当使用Sencha Cmd生成Ext JS 6应用程序时，下面的命令将生成一个通用应用程序(一个同时使用经典和现代工具包的应用):

sencha -sdk path/to/ext6 generate app YourAppName ./AppFolder

要为Ext JS 6生成一个经典的工具类型的应用程序，添加“—classic“参数:

sencha -sdk path/to/ext6 generate app --classic YourAppName ./AppFolder

类似地，生成一个仅使用现代工具箱的应用程序使用如下命令:

sencha -sdk path/to/ext6 generate app --modern YourAppName ./AppFolder

### app.json

使用Ext JS 6生成的应用程序拥有一个更完整的app.json文件。

app.json文件的某些部分可能需要合并到现有应用程序的app.json文件中。

### “sass”对象

该对象包含控制样式编译的属性。下面的代码片段将在新生成的应用程序中应用。

如果您的app.json没有一个“sass”对象，那么您可以直接粘贴以下内容。

/\*\*

\* Sass configuration properties.

\*/

"sass": {

/\*\*

\* The root namespace to use when mapping \*.scss files to classes in the

\* sass/src and sass/var directories. For example, "App.view.Foo" would

\* map to "sass/src/view/Foo.scss". If we changed this to "App.view" then

\* it would map to "sass/src/Foo.scss". To style classes outside the app's

\* root namespace, change this to "". Doing so would change the mapping of

\* "App.view.Foo" to "sass/src/App/view/Foo.scss".

\*/

"namespace": "App",

/\*\*

\* Comma-separated list of files or folders containing extra Sass. These

\* files are automatically included in the Sass compilation. By default this

\* is just "etc/all.scss" to allow import directives to control the order

\* other files are included.

\*

\* All "etc" files are included at the top of the Sass compilation in their

\* dependency order:

\*

\* +-------+---------+

\* | | base |

\* | theme +---------+

\* | | derived |

\* +-------+---------+

\* | packages | (in package dependency order)

\* +-----------------+

\* | application |

\* +-----------------+

\*/

"etc": [

"sass/etc/all.scss",

"${toolkit.name}/sass/etc/all.scss"

],

/\*\*

\* Comma-separated list of folders containing Sass variable definitions

\* files. These file can also define Sass mixins for use by components.

\*

\* All "var" files are included after "etc" files in the Sass compilation in

\* dependency order:

\*

\* +-------+---------+

\* | | base |

\* | theme +---------+

\* | | derived |

\* +-------+---------+

\* | packages | (in package dependency order)

\* +-----------------+

\* | application |

\* +-----------------+

\*

\* The "sass/var/all.scss" file is always included at the start of the var

\* block before any files associated with JavaScript classes.

\*/

"var": [

"sass/var/all.scss",

"sass/var",

"${toolkit.name}/sass/var"

],

/\*\*

\* Comma-separated list of folders containing Sass rule files.

\*

\* All "src" files are included after "var" files in the Sass compilation in

\* dependency order (the same order as "etc"):

\*

\* +-------+---------+

\* | | base |

\* | theme +---------+

\* | | derived |

\* +-------+---------+

\* | packages | (in package dependency order)

\* +-----------------+

\* | application |

\* +-----------------+

\*/

"src": [

"sass/src",

"${toolkit.name}/sass/src"

],

/\*\*

\* File used to save sass variables edited via Sencha Inspector. This file

\* content will override all other variables.

\*/

// "save": "sass/save.scss"

},

上面的属性将替换掉”sencha/app/sencha.cfg”文件中的 “apps.sass.\*”属性。

尽管Fashion并不是Sass的一个实现，但我们在上下文中保留了“sass”这个术语，因为它在历史上一直是这些路径设置的名称。

### “css”数组

以前，应用程序都有这些配置(在Ext JS 5或Sencha Con发行版中):

{

"path": "bootstrap.css",

"bootstrap": true

},

{

"path": "${build.out.css.path}"

"bundle": true

}

这些值将会被如下值取代：

我们添加的exclude属性确保了CSS文件在启用Fashion时跳过，因为它将从.scss中构建CSS。

## 工作空间（workspace）更新

有了Sencha Cmd 6，现在就有了一个工作区。

将json文件添加到工作区根文件夹中。

虽然该文件将在需要时生成，但在升级过程中会得到一些注意。

workspace.json

这个文件在默认情况下是这样的: