CocoAuto：组件脚手架与自动构建发布

1. 自动化构建

Coco输入法组件开发基于CocoAuto自动化构建发布项目，CocoAuto包含三部分：

1.1. CocoAutoCreator

CocoAutoCreator负责项目创建，其用法如下：

```
python CocoAutoCreator.py [UserNameForIcode] [ProjectName] [ModuleName]
```

该命令会从Icode上clone脚手架工程 `CocoScaffold` ，并基于该工程，创建相应的开发工程。所有Coco、输入法、基础库等相关的依赖都会自动配置好，这部分开发者无需关心。

创建完成后的工程目录结构如下：

```
ProjectName // 工程名
    devapp // 开发&测试APP
    bpisdk // sdk library module
    moduleName // 需要开发的组件
```

- `devapp` 是输入法项目代码，初期可以考虑用完整的输入法项目替代
- `bpisdk` 是放置所有bpi文件的项目，该module会自动化生成与更新，开发者无需关心也不要修改。
- `moduleName` 是真正需要开发的业务组件。

1.2. CocoAutoUpgrader

CocoAutoUpgrader负责对组件脚手架依赖包进行升级，其用法如下：

`python CocoAutoUpgrader.py [UserNameForIcode] [ProjectName] [ModuleName]`

该命令会从Icode上clone脚手架工程 `CocoScaffold` ，并基于该工程，将脚手架中的基础依赖库版本进行自动化更新，以确保组件可以用到最新的依赖库版本，并与整个项目保持一致。

1.3. CocoAutoDeployer

CocoAutoDeployer用于将测试完成的组件项目发布到maven仓库，其用法如下：

`python CocoAutoDeployer.py upload`

2. 组件开发流程

总体流程：

1. 基于CocoAutoCreator构建项目 ->
2. 在devapp中引入需要依赖的其它业务组件aar ->
3. Gradle Sync ->
4. 业务组件开发 ->
5. 开发完成提测 ->
6. Jenkins编译出相应aar与apk，交由QA测试 ->
7. 测试通过 ->
8. Jenkins通过CocoAutoDeployer发布对应aar到maven仓库

业务组件开发流程：

1. 详设，确认接口定义，总体框架 ->
2. 编写 `.bpi` 接口文件，生成接口文档 ->
3. 在devapp中调用相应接口完成接入 ->
4. 编写接口单元测试 ->
5. 编写业务组件整体框架代码 ->
6. 实现内部逻辑 ->
7. 接口若存在问题需要修改回到2重新开始