蚂蚁集团
ANT GROUP

蚂蚁科技产品手册

前端框架与 UI 组件

产品版本：V20201130

文档版本：V20201130

蚂蚁科技技术文档

商标声明

及其他蚂蚁科技服务相关的商标均为蚂蚁科技所有。
本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

由于产品版本升级、调整或其他原因，本文档内容有可能变更。蚂蚁科技保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在蚂蚁科技授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过蚂蚁科技授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

目录

1 基于 H5 框架 - Kylin 框架

1.1 Kylin 简介

Kylin 是 mPaaS H5 容器的无线前端解决方案，具有高效的运行时、一致的开发体验、丰富的研发支撑、完善的 UI 组件等诸多优点，解决移动 Hybrid 开发中遇到的前端打包、浏览器兼容性、Mock 接口等问题。

Kylin 仅提供基于 Vue 2.0 的视图层框架，以极小的 JS 加载开销实现高效的 DOM 更新。使用 Kylin 前请先学习 Vue 官方文档。

Kylin 作为无线前端解决方案只提供 Safari、UC Webview 和 Chrome 浏览器兼容性保障。

说明：mPaaS 已经提供了更新和更稳定的小程序组件，并作为今后主要维护和发展方向。相比于Kylin 方案，小程序组件有着更高的易用性和稳定性，能够适配多种场景。因此建议新用户接入小程序组件，已经接入了 Kylin 框架的用户在遇到新增功能需求时，建议您也转用小程序组件。更多详情，请参见 小程序 。

1.2 快速开始

1.2.1 搭建前端开发环境

前端开发环境需安装 NodeJS 和 cnpm。本文档将引导您针对不同操作系统下的环境搭建过程。如果您使用 Windows 操作系统，需要先完成windows 用户配置，操作方法参考 Windows 用户配置 。

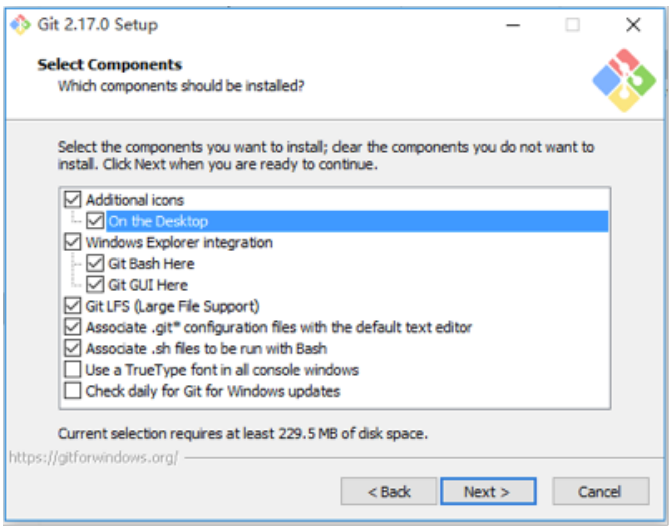
Windows 用户

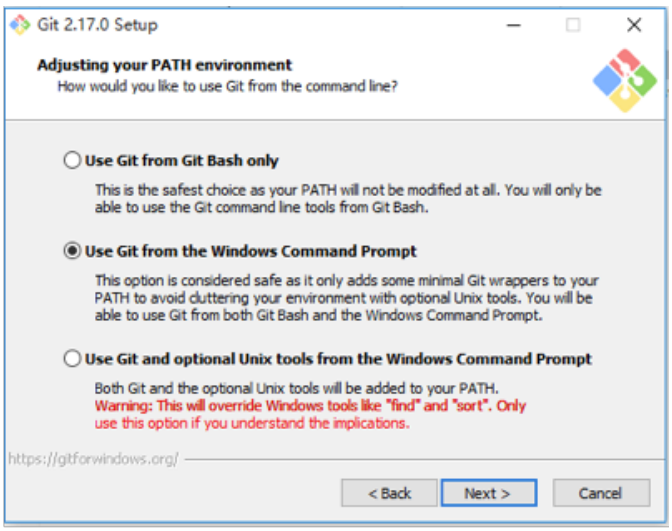
若您使用 Windows 操作系统，则需完成 Windows 用户配置 ，再安装 NodeJS 和 cnpm。

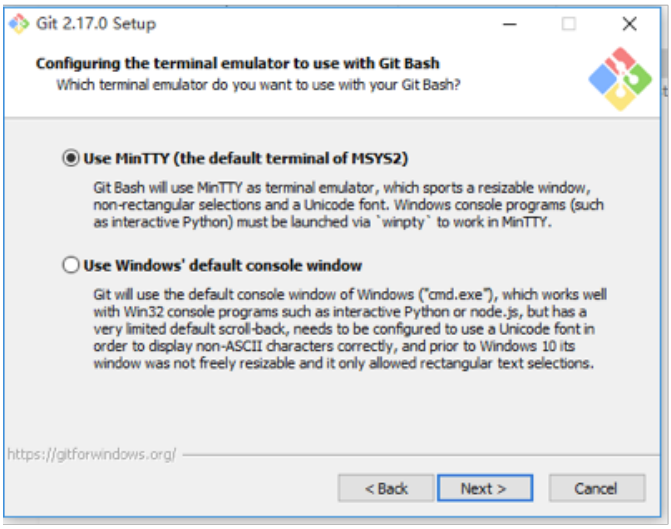
Windows 用户配置

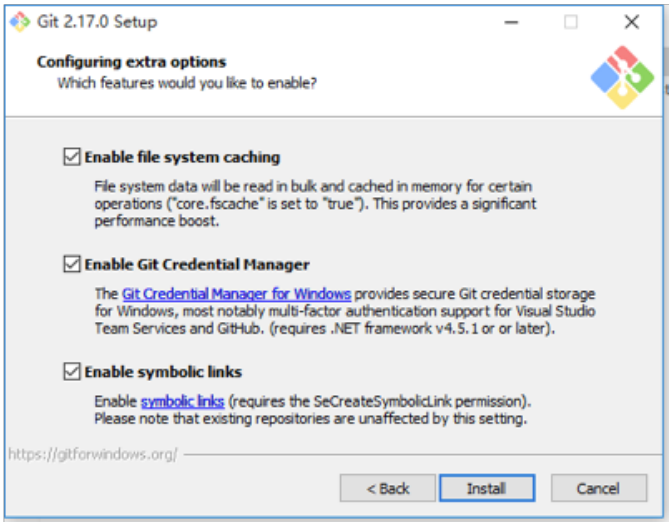
安装 mingw 命令行环境，下载地址：git-scm。

安装步骤如下图所示：









安装 NodeJS

下载 并安装 NodeJS v8 版本。

安装 cnpm

在 **mingw** 或 **终端** 中，执行以下命令，安装 cnpm：

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

重要：请勿使用添加 npm 参数的 alias 方式安装 cnpm。有关 cnpm 的介绍信息，参见 淘宝 NPM 镜像。

安装完成后，执行以下命令，检查是否已成功安装：

```
cnpm -v
cnpm@6.1.0 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\lib\parse_argv.js)
npm@6.11.3 (C:\Users\wb-
```

```
wly538545\AppData\Roaming\npm\node_modules\cnpm\node_modules\npm\lib\npm.js)
node@8.11.1 (D:\Program Files (x86)\nodejs\node.exe)
npminstall@3.23.0 (C:\Users\wb-
wly538545\AppData\Roaming\npm\node_modules\cnpm\node_modules\npminstall\lib\index.js)
prefix=C:\Users\wb-wly538545\AppData\Roaming\npm
win32 ia32 10.0.17134
registry=https://r.npm.taobao.org
```

MacOS 用户

安装 NodeJS

下载 并安装 NodeJS v8 版本。

安装 cnpm

操作方法参见 Windows 用户 > 安装 cnpm 。

相关链接

参见 获取代码示例 获取 Kylin Demo。

1.2.2 开发调试

进行开发调试，需要完成以下步骤：

1. 安装依赖
2. 开发调试
3. 构建生产

点击 代码示例，获取 Kylin Demo，并完成以下操作。

安装依赖

进入项目根目录，使用 cnpm 安装 npm 依赖：

```
# 安装npm依赖
cnpm install
```

开发调试

安装完环境后，可以通过以下一些命令来启动开发模式：

```
cnpm run dev
```

上述命令完成了以下操作：

- 运行 kylin build --dev，以 dev 模式开始构建：
  - 不使用 compress 压缩 css/js

- 自动 watch 代码变动
- 支持代码热更新

- 在 http://localhost:8090/ 启动一个服务器。

构建生产

可以通过以下命令来启动构建模式：

```
cnpm run build
```

上述命令完成了以下操作：

- 运行 kylin build 命令，将工程的源代码进行编译。
- 编译完成后将产物输出到./www/目录，以备后续打包。

1.3 项目结构

1.3.1 脚手架简介

本文介绍项目初始化结构以及每个部分的详细说明。

初始化结构

项目初始化结构如下:

```
project
├── mock
│  ├── mock.config.js
│  └── rpc
│  └── test.js
├── package.json
├── www
└── src
├── common
│  ├── components
│  ├── css
│  │  └── base.less
│  ├── img
│  └── js
├── layout
│  ├── index.html
│  └── layout.html
└── pages
└── index
├── components
├── index.js
└── store
```

子目录

- mock
- package.json
- www
- src/common
- src/layout
- src/pages
- 常用参数

mock

该目录提供了一种数据 mock 方式，即使用 cnpm run dev:mock 启动时，会自动加载其中的 rpc 目录和 jsapi 目录的对应数据接口。

package.json

在 package.json 文件中的 kylinApp 字段包含了项目配置的元信息，主要有 pages、output、devPort、plugins、dirAlias。

简单举例如下:

```
{
"kylinApp": {
"output":"www",
"pages": {
"index": {...}
},
"devPort": 8090,
"dirAlias": {
"common":"./src/common/",
"pages":"./src/pages/"
},
"plugins": [

]
}
}
```

www

执行 cnpm run build 后，会自动将构建产物输出到 www 目录中。

src/common

用以放置项目中使用的 css、js、img 文件。

src/layout

对应着 ./src/pages/${pageName} 的各个页面，可以在 package.json 中配置对应页面使用的 html 模板路径。支持

nujuncks 语法。

src/pages

这个目录是用来放各个页面的，各个页面分别放在 ./src/pages/${pageName}/ 目录下。分别包含了 components，store 和 index.js 。

- components 目录中，每个组件都是 Vue 组件，具体编写规范请参考 组件规范 。
- store 目录中，有一个 Vuex.Store 实例，具体使用请参考 状态注入 。
- index.js 为当前 **page** 的主入口，这里的 **page** 页面最后会生成一个特定的 ${pageName}.html 页面。

常用参数

下表的常用参数是指 kylinApp 下一级中，除了 pages、options、plugins 之外的所有键值。pages、options、plugins 将在下面单独展开。

参数名	类型	默认值	备注
output	String	dist	输出相对目录。
devPort	Number	8090	dev 模式监听的 IPv4 端口号(0.0.0.0:devPort)。
dirAlias	Record	{}	等同于 webpack.resolve.alias 中使用**相对路径**。
pageTemplate	String	-	公共 nunjucks 模板。

pages

此处列举 pages 键值对下的配置项，示例中的 home 表示以下配置均为对 pageName 为 home 的页面生效。

```
{
"kylinApp": {
"pages": {
"home": {
... // 这里的字段
}
}
}
}
```

字段名	类型	默认值	备注
entry	String	-	相对路径，指向当前页面的 JS 打包入口。
template	String	-	相对路径，指向当前页面的 HTML 打包路径，如果为空，会寻找 kylinApp.pageTemplate 字段值。

plugins

kylinApp.plugins 字段，是一个数组，支持按需加载各个插件。

```
{
kylinApp: {
plugins: [
"xxxx",
["yyyy",{ a: 1 }],
"zzzz",
["6666",{ b: 1 }]
]
}
}
```

支持传入的形式有 2 种，分别是 **默认配置** 和 **扩展配置** 方式，在上述的示例中，引入了 4 个插件。

- @ali/kylin-plugin-xxxx，以默认配置加载。
- @ali/kylin-plugin-yyyy，以 {a:1} 选项加载。
- @ali/kylin-plugin-zzzz，以默认配置加载。
- @ali/kylin-plugin-6666，以 {b:1} 选项加载。

已有插件

目前，支持配置的插件有 mock、resource，分别见如下文档：

- mock
- resource

1.3.2 页面

Page 是一个 Webview 的逻辑抽象层，同时也是组件挂载的根节点。

代码引入

```
import { Page } from '@ali/kylin-framework';
```

页面声明结构

一个Page包含的接口在 页面接口 中声明，提供了对 Vue 实例的完整控制能力，简易的 Page 使用如下，initOptions 负责处理额外的 Vue 配置选项。

```
import { Page } from '@ali/kylin-framework';
import IndexComponent from './indexComponent.vue';

class IndexPage extends Page {

initOptions() {
return {}
}

render(h) {
return <IndexComponent></IndexComponent>
```

```
}

}

new IndexPage('#app');
```

页面接口

本部分介绍页面接口的命名空间及 API。

命名空间

ES6 通过如下方式引入：

```
import { Page } from '@ali/kylin-framework';
```

API

目前 Page 提供如下成员方法以供派生：

- initOptions
- render

initOptions

```
function initOptions(): VueOptions
```

返回值

返回结果要求是一个合法的 Vue 入参。一般来说，不建议在 Page 层引入过于复杂的配置，涉及到的逻辑都可以放到 Component 中来维护。

render

该函数要求是一个合法的 Vue 的 render 函数。

```
function render(): VNode
```

返回值

返回结果要求是合法 VNode 元素，请按照 JSX 规范进行书写。

1.3.3 组件

Component 扩充自 Vue 的组件, 提供了 Vue 组件对等的输入参数能力。在代码书写时提供类 class 的装饰器 Decorator 风格。

代码引入

```
import { Component, Watch } from '@ali/kylin-framework';
```

组件声明结构

一个组件可以包含数据、JSX 渲染函数、模板、挂载元素、方法、生命周期等 Vue 的 options 选项的对等配置。组件声明包括以下几部分, 分别使用 @Component 和 @Watch 两种不同装饰器进行包装：

- class 类声明, 使用装饰器 @Component。
- 类成员声明, 不使用装饰器。
- 类成员方法声明, 一般不装饰器, 除非该方法需要 watch 另外一个已声明的变量。

下面是简单的组件开发演示，具体声明参数参见 组件接口 了解详情。

***.vue 单文件组件**

```
<!-- Hello.vue -->

<template>
<div>hello {{name}}
<Child></Child>
</div>
</template>

<style>
/* put style here */
</style>

<component default="Child"src="./child.vue"/>

<script>
import { Component } from '@ali/kylin-framework';

@Component
class Hello {
data = {
name: 'world'
}
}

export default Hello;
</script>
```

关于 <component> 标签式的组件依赖，参见 组件接口 了解详情。

组件接口

跟 vue 基本一致，组件定义写在 .vue 文件内，以下是一个简单的例子:

```
<template>
<div>
```

```
<AButton @click="onClick">点击</AButton>
</div>
</template>

<style lang="less"rel="stylesheet/less">
/* less */
</style>

<dependency component="{ AButton }"src="@alipay/antui-vue"lazy/>

<script type="text/javascript">
import { Component } from '@ali/kylin-framework';
@Component
export default class IndexView {
props = {}
data = {}
get comput() { return this.data.c * 2 }
onClick() {}
mounted() {}
}
</script>
```

上述例子中，有 4 个顶级标签，除了与 Vue 相同的 <template>、<style>、<script> 之外，还有一个 <dependency> 标签。

下文将对这 4 个标签的具体作用分别进行阐述。其中 <template>、 <style> 与 vue 中定义一致。

script

class 结构

定义一个 Component ，在代码结构上，使用类 class 的装饰器 Decorator 风格。其中装饰器有 @Component 和 @Watch 2 种，通过以下方式引入。

```
import { Component, Watch } from '@ali/kylin-framework';

@Component
export default class Hello {

}
```

方法类型

组件以 class 形式声明，必须对该 class 进行装饰器修饰。在 class 内部，成员变量是不需要被手动处理的，在构建过程中通过 babel 插件自动进行处理，而成员函数一般不需要装饰器挂载，除非是使用 @Watch 的场景,其中 @Component 会处理的属性如下表所示。

成员类型	名称	功能
get/set property	*	用以转换成 Vue 的 computed 计算属性，可以直接通过 this[varName] 调用。
beforeCreate	生命周期	生命周期方法，与 Vue 对等。
created	生命周期	生命周期方法，与 Vue 对等。
beforeMount	生命周期	生命周期方法，与 Vue 对等。

mounted	生命周期	生命周期方法，与 Vue 对等。
beforeUpdate	生命周期	生命周期方法，与 Vue 对等。
updated	生命周期	生命周期方法，与 Vue 对等。
beforeDestroy	生命周期	生命周期方法，与 Vue 对等。
destroyed	生命周期	生命周期方法，与 Vue 对等。
method	*	普通成员方法, 用以转换成 Vue 的 methods 方法列表。

getter/setter属性

```
@Component
export default class Hello {
get computName() {
// to sth
}
}
```

上述 getter 声明，等价于如下 vue 配置

```
HelloOption = {
computed: {
computName: {
get: computName() {
// to sth
}
}
}
}
```

同理，setter 也会被提取，如果同时存在 getter 和 setter 则会被一起提取。

生命周期函数

```
@Component
export default class Hello {
created() {}
mounted() {}
}
```

上述 created 和 mounted 生命周期函数，会被提取为数组。

```
TestOption = {
created: [function created(){}],
mounted: [function mounted(){}],
}
```

支持的生命周期方法名如下，beforeCreate、created、beforeMount、mounted、beforeUpdate、updated、beforeDestroy、destroyed。

Watch

该装饰器的出现，只是因为 watch 需要有以下几个要素：

- 被监听的变量名
- 监听选项
- 触发函数

用法

完整的 @Watch 接口如下表所示。

```
function Watch( key: string [, option: Object = {} ] ): PropertyDecorator
```

参数名	类型	用途	
key		string	监听的参数名，来自 computed、data、props 三者之一。
option	deep	boolean	若监听的是复杂对象，其内层数据变更是否触发，默认为 false。
	immediate	boolean	立即以表达式的当前值触发回调，默认为 false

示例

- 对于 @Watch 装饰的成员函数，支持对成员函数配置多个变量的监听，如下同时对 a 和 c 的变化进行了监听，如果任何一个发生变化，会触发 OnChangeA 成员方法。
- 如下，OnChangeA 本质是成员方法，所以他也会和其他成员方法一起被提取到methods块中，那么必须保证没有与其他方法重名。
- 如果对Watch有额外配置项，请按 @Watch('a', {deep: false})的方法传入。配置项请参考 watch配置项。

```
@Component
class WTest {

data = {
a: {
b: 2
},
c: 3
}

@Watch('c')
@Watch('a', {deep: false})
OnChangeA(newVal, oldVal) {

}
}
```

注意: 以上对 data.a 的监听，会转换成如下形式，需要注意的是，如果没开启 deep: true 选项，当 data.a.b 发生

变动的时候，不会触发该 OnChangeA 监听。

属性类型

构建工具会自动对成员变量应用了 @Component.Property 装饰器，不需要用户手动填写，最终的合并策略取决于被装饰的 成员变量 的标识符名称，框架内置了以下几种。如果不在下表中，会透传至 VueComponent 的 options 对象中。

成员类型	名称	功能
property	props	vue的props属性
property	data	vue的data属性，**会被转换成函数形式, 支持 this**，请勿直接写 data(){} 函数
property	*	其他未知属性，直接复制到 Vue 的 options 中的对应属性上

props

```
@Component
export default class Hello {

props = {
name: {
type: String,
default: 'haha'
},
num: Number
}
}
```

上述 props 成员变量定义，会被直接转换成到 options 中对应的 props。

```
HelloOption = {
props: {
name: {
type: String,
default: 'haha'
},
num: Number
}
}
```

具体完整定义结构请参见 Vue 文档 API-props。

data

```
@Component
export default class Hello {
props = {
name: {
type: Number,
default: 1
},
```

```
}
data = {
hello: this.props.name + 2
}
}
```

上述 data 成员变量定义，**会被转换成 data 函数形式**，您无需手动编写 data 函数。

```
TestOption = {
props: {
name: {
type: Number,
default: 1
},
},
data: function data() {
return {
hello: this.props.name + 2
}
}
}
```

dependency

上述 <script> 定义中，没有说明组件依赖的使用方式，在 .vue 文件中，推荐使用以下写法来标记组件依赖，即 <dependency> 标签，下面示例中即引用了 ./child.vue 组件。

```
<template>
<child></child>
</template>

<dependency component="Child"src="./child.vue"/>
```

标签属性

default 导入

针对 ES6 Module 的 default 导出或者 commonjs Module 对象的导出，可使用如下方式引入。

属性	类型	默认值	备注
component	string	必填	引入到 options.components 的标识符名。
src	string	必填	组件来源，同require(src)。
lazy	boolean	false	是否对该组件启用懒加载（当且仅当被 Vue 使用到时再进行 require 加载模块）。
style	string	undefined	默认不启用，如果设置了字符串，会根据 ${src}/${style} 的路径来加载组件对应样式文件。

如下示例:

```
<dependency component="name"src="source"lazy />
```

member 导入

针对 ES6 Module 的命名导出，可使用如下方式引入：

属性	类型	默认值	备注
component	string	必填	引入到 options.components 的多个命名标识符, 必须以花括号 { , } 包括，否则会识别为 default 引入。
src	string	必填	组件来源，同 require(src)。
lazy	boolean	false	是否对该组组件启用懒加载（当且仅当被 Vue 使用到时再进行 require 加载模块）。

如下示例:

```
<dependency component="{ List, ListItem, AButton }"src="@alipay/antui-vue"lazy />
```

默认对 @alipay/antui-vue 组件库支持 babel-plugin-import 按需加载。

template

模板的内容结构与 vue 一致。

```
<template>
<div>Hello World</div>
</template>
```

style

```
<style lang="less"rel="stylesheet/less">
/* less */
</style>
```

其中，可以通过添加 scoped 属性标记来使得该样式只对当前组件生效。

```
<style lang="less"rel="stylesheet/less"scoped>
/* less */
</style>
```

状态注入

对于 Kylin 组件， 如果需要使用到 Store 中的属性，使用计算属性把 $store 对象中的属性透传出来是一种不推荐的写法，如下所示：

```
@Component
```

```
class Hello {
// 通过计算属性来关联store中的状态
get hello() {
return this.$store.state.hello
}
}
```

推荐使用下面的 connect 机制来透传 $store 数据：

- mapStateToProps
- mapActionsToMethods
- mapMethods

接口声明

```
@Component({
mapStateToProps: Object|Array,
mapActionsToMethods: Object|Array,
mapMethods: Array|Boolean,
mapEvents: Array
})
class Hello {

}
```

mapStateToProps

把 state 中的特定键值映射到当前组件的 props 中，其接收参数等价于 Vuex 提供的 mapState 辅助函数。

有以下 3 种方式可实现上述功能：

函数方式

说明：把 $store.state 中的名为 bbb 的数据，映射到名为 aaa 的 props 上。

```
{
mapStateToProps: {
aaa: (state, getters) => state.bbb
}
}
```

字符串键值对方式

说明：把 $store.state 中名为 bbb 的数据，映射到名为 aaa 的 props 上。

```
{
mapStateToProps: {
aaa: 'bbb'
}
}
```

字符串数组方式

说明：

- 把 $store.state 中名为 aaa 的数据，映射到名为 aaa 的 props 上。
- 把 $store.state 中的名为 bbb 的数据，映射到名为 bbb 的props 上。

```
{
mapStateToProps: ['aaa', 'bbb']
}
```

mapActionsToMethods

与 Vuex 中 mapActions 入参一致，支持使用对象方式（名称映射）、数组方式（名称）把在全局 $store 下配置的 actions 注入到当前组件的 methods 中。

```
@Component({
mapActionsToMethods: ['a', 'b']
})
class IndexView {
async doSomeAction() {
const ret = await this.a(123);
// 等价于调用
// const ret = await this.$store.dispatch('a', 123);
}
}
```

mapMethods

通过在父子组件之间加一层中间层组件的方式来具体实现 connect 机制。当父组件调用子组件中特定的 method 方法时，无法直接访问子组件（实际访问到的是中间层组件），需要通过以下配置实现访问。

```
@Component({
mapMethods: true
})
export default class Child {
a() {}
}


<template>
<div>
this is parent
<child ref="child"></child>
</div>
</template>
<script>
@Component
export default class Parent {
b() {
// 此处可以访问
this.$refs.child.a();
}
}
```

```
</script>
```

1.3.4 命令行工具

初始化

当工程脚手架初始化完后，如果需要新增页面，除了单纯的复制粘贴以外，提供了以下命令来添加页面定义和组件定义：

- init-page
- init-component

init-page

命令格式

```
kylin init-page <pageName>
```

注意事项

- 上述命令中 pageName 为必选参数 ，指新创建页面的英文名称。
- 如果当前 cwd 下有 package.json 并且存在 kylinApp 字段，则会自动往 kylinApp.pages 添加新增的 page。

init-component

命令格式

```
kylin init-component <componentName>
```

注意事项

上述命令中 componentName 为必选参数 ，指新创建组件的英文名称。

如果当前 cwd 下有 package.json 并且 kylinApp.pages 大于 1 个 page，会提示选择在哪个 page/components 目录下创建。

构建

本部分介绍工具构建的命令格式、公共资源包注入的构建提示。

命令格式

```
kylin build # ... args
```

项目

常用使用

```
kylin build --dev # dev构建及静态服务器
kylin build --server --no-prod --hot # dev构建及静态服务器及启用热更新
kylin build --server # prod构建及静态服务器
kylin build --no-prod --watch # dev构建及监听文件变化
```

命令行入参

参数名	类型	备注
—dev	boolean	同老 buildtool 一致，使用 dev 的 conf 并开启 server。开启该选项会强制设置 prod=false,server=true,hot=true
—no-prod	boolean	prod 为 true 时使用 prod 的 conf 编译，为 false 时使用 dev 的 conf 编译，同理设置 NODE_ENV
—server	boolean	只开启静态服务器，开启该选项会强制设置 watch=true
—verbose	boolean	webpack 输出明细
—watch	boolean	是否检测文件变化
—no-compress	boolean	关闭压缩，默认启用压缩
—no-common	boolean	关闭 CommonsChunkPlugin，默认开启 common
—hot	boolean	开启热更新，默认关闭，只能在 prod=false 且 server=true 时使用
—open [entry]	boolean, String	只能在 --server 时有效，会打开 entry 指定的入口 URL，只 --open 但未明确指定 entry 时会处理第一个
—mock	boolean, String	开启 mock 插件读取 ./mock/mock.config.js

kylinApp配置选项

参数名	类型	备注
devPort	Number	默认监听 IPv4 的 0.0.0.0:8090 端口
pageTemplate	String	页面模板路径
output	String	输出相对目录
options	Object	额外选项，如下
dirAlias	Object	等同于 webpack.resolve.alias，如 { common:’ ./src/common/ ‘ }

构建提示

公共资源包注入

对于以下 require / import 的包路径，会自动注入对应 <script> / <link> 标签到对应 HTML 中。

包名	映射全局对象	映射路径
fastclick	FastClick	as.alipayobjects.com/g/luna-component/luna-fastclick/0.1.0/index.js

vue	Vue	a.alipayobjects.com/g/h5-lib/vue/2.1.6/vue.min.js
es6-promise	Promise	as.alipayobjects.com/g/component/es6-promise/3.2.2/es6-promise.min.js
fetch	fetch	as.alipayobjects.com/g/component/fetch/1.0.0/fetch.min.js
zepto	Zepto	a.alipayobjects.com/amui/zepto/1.1.3/zepto.js

1.4 插件

1.4.1 mock

Kylin-plugin-mock 插件是针对在桌面浏览器（Chrome）中调试 JSAPI 的需要而开发的数据 mock 插件。

开启插件

在脚手架工程中，执行如下语句即可，其等价于运行命令时添加 --mock：

```
cnpm run dev:mock
```

使用插件

在项目的 ./mock/mock.config.js 文件中，有如下配置项：

```
const config = {};

// 用户自定义mock
config.call = {
// mock rpc 接口
rpc: function (opts, callback) {
var type = opts.operationType;
var rpc = require('./rpc/' + type);
var data = typeof rpc === 'function' ? rpc(opts) : rpc;
// 防止在业务逻辑中对传入的对象进行了修改
data = Object.assign({}, data);
// 模拟服务端/网络接口延迟，此时会发现打了 2 次 log，一次是请求，一次包含返回结果
setTimeout(() => {
callback && callback(data);
}, 2000);
},
}

window.lunaMockConfig = config;
```

上述配置将 ./mock/rpc/*.js 中的接口进行数据映射。更多详细配置，可 获取代码示例 后查看。

示例

在执行 cnpm run dev:mock 后，会进入 mock 模式。该模式下在浏览器内执行 AlipayJSBridge.call('abc') ，会去 ./mock/jsapi/abc.js 寻找模拟接口数据。

1.4.2 resource

Kylin-plugin-resource 插件是针对 mPaaS 平台下的全局离线资源包设计的一种资源拦截机制。

使用插件

在脚手架的 package.json 中，可以看到如下配置：

```
["resource",
{
"map": {
"vue": {
"external":"Vue",
"js":"https://gw.alipayobjects.com/as/g/h5-lib/vue/2.5.13/vue.min.js"
},
"fastclick": {
"external":"FastClick",
"js":"https://as.alipayobjects.com/g/luna-component/luna-fastclick/0.1.0/index.js"
}
}
}
]
```

上述配置项表示当代码中出现如下的依赖语句，会进行一定处理：

```
import xxx from 'vue';
var xxx = require('vue');
```

上述对 vue 的依赖使用，会做如下处理:

1. 在生成的 HTML 模板中注入 <script src="https://gw.alipayobjects.com/as/g/h5-lib/vue/2.5.13/vue.min.js"></script> 脚本资源。
2. 把上述 vue 依赖重定向为 window.Vue 的值。

1.4.3 扩展能力

Kylin 框架支持扩展 webpack，babel，express 配置。

操作步骤

完成以下步骤进行扩展：

1. 在工程根目录创建 plugin.js。
2. 在 package.json 中的 kylinApp.plugins 中添加如下配置项：

```
// 在 package.json 中
{
"kylinApp": {
```

```
"plugins": [
"module:./plugin.js"
]
}
}
```

3. 在 plugin.js 中编写如下代码。在如下 modifyWebpackConfig 和 modifyBabelConfig 两个函数中，修改原有的 webpack 和 babel 配置：

```
// 在 plugin.js 中
exports.pluginName = '@ali/kylin-plugin-custom';
exports.default = function () {
return {
webpack: function modifyWebpackConfig(webpackConfig) {
console.log('webpackConfig', webpackConfig);
return webpackConfig;
},
babel: function modifyBabelConfig(babelConfig) {
console.log('babelConfig', babelConfig);
return babelConfig;
},
express: function modifyExpress(expressInstance) {
expressInstance.use(/* some middleware */);
}
}
}
```

示例

接入 httpProxy 插件

重要： 依赖 build@ 0.1.49+ 及以上版本。

1. 运行以下命令安装 http-proxy-middleware：

```
cnpm install --save-dev http-proxy-middleware
```

2. 修改 **httpProxy** 插件的 express 过程：

```
// 在 plugin.js 中
var proxy = require('http-proxy-middleware');
exports.pluginName = '@ali/kylin-plugin-custom';
exports.default = function () {
return {
express: function modifyExpress(expressInstance) {
expressInstance.use(
proxy('/api', {target: 'http://www.baidu.com'})
// 更多详细配置参阅 http-proxy-middleware 模块文档
);
```

```
}
}
}
```

接入 px2rem 插件

1. 运行以下命令安装 postcss-px2rem：

```
cnpm install --save-dev postcss-px2rem
```

2. 修改 **px2rem** 插件的 Webpack 过程。

```
// 在 plugin.js 中
var px2rem = require('postcss-px2rem');
exports.pluginName = '@ali/kylin-plugin-custom';
exports.default = function () {
return {
webpack: function modifyWebpackConfig(webpackConfig) {
webpackConfig.vue.postcss.push(
px2rem({
// 该参数和模版 html 中的 font-size 配置需要根据具体项目的 UI 交互稿修改
remUnit: 100
})
);
return webpackConfig;
}
}
}
```

2 基于 H5 框架 - UI 组件库

2.1 UI 组件库简介

AntUI 前端组件库，是基于蚂蚁金服支付宝 App 视觉交互规范，以 vue 为视图框架封装的一套组件库。AntUI 前端组件库是在基于原有 AntUI 无线 H5 样式库的样式下，经过交互逻辑完善，配置项抽象后的组件库。

主要有以下组成部分：

- @alipay/antui 样式库
- @alipay/antui-vue 基于上述样式库的组件库

相关链接

AntUI 无线 H5 样式库

2.2 使用指南

在 Kylin 工程下，可以看到 package.json 中已经存在了 @alipay/antui-vue 的依赖。要使用 Ant UI 前端组件，您可以通过以下方式之一：

- 在 Kylin 工程中使用该组件，按照 dependency 的方式导入，因为 Kylin 工程脚手架包含了这该组件的依赖。具体方式，参见本文的 Kylin 部分。
- 在别的工程中使用，通过 ESModule 的方式来导入组件。具体方式，参见本文的 ESModule 部分。

此外，每个组件还会提供 API 文档，因为对于一个标准的 Kylin 组件， prop、slot、method、event 是提供的接口。例如，要使用 AButton，可以定制 props 中的 size，来选择这个按钮是大是小，可以配置 slots 中的 icon 来自定义按钮样式，可以通过 events 中的 click 事件来获得点击事件。具体信息，参考每个组件的 API 文档部分。

Kylin

在 Kylin 工程的 .vue 组件文件下，支持特殊语法 <dependency> ，可以使用如下方式注册组件依赖：

```
<dependency component="{ Notice }"src="@alipay/antui-vue"></dependency>
```

上述语法表示，在当前 .vue 文件内，可以在 template 模板中使用 Notice 组件。

ESModule

在其他工程结构中，需要确保以下几件事：

loader

开发者需要能够加载 .vue 文件，比如在 webpack 下可以通过 vue-loader。

js

在支持 .vue 文件加载后，就像普通组件一样使用：

```
import { Notice } from '@alipay/antui-vue';

Vue.component(Notice.name, Notice);
```

2.3 基本组件

2.3.1 AButton 按钮

AButton 按钮文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式来导入组件。
- API 文档 提供了 props、slots、events 的接口信息。

此外，要查看该组件的视觉效果及示例代码，请参考本文 Demo 。

Kylin

```
<dependency component="{ AButton }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { AButton } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
type	按钮类型，可选值为 blue , white , warn , bottom , page-reuslt (仅用于 PageResult 下)。	String	blue
size	按钮大小，可选值为 default , tiny , 当设置为 tiny 时，type 仅支持white , blue , ([type=page-result] 不支持)。	String	default
disabled	设置禁用, ([type=page-result] 不支持)。	boolean	false
loading	设置按钮icon为loading，仅适用于 size=default 时, ([type=page-result] 不支持)。	boolean	false
success	设置按钮icon为success，仅适用于 size=default 时, ([type=page-result] 不支持)。	boolean	false
nativeType	DOM上的 type 属性。	String	button
href	如果不为空，使用a标签渲染，否则默认使用 button 渲染.。	String	-
inactiveLoading	设为 true 时，当 loading 为 true 时不会触发click事件，且不会改变 button 的样式；设为 false 时，保持 loading 原有行为。	boolean	false

slots

name	说明	scope
-	默认slot,填充按钮文本	-
icon	用于填充icon的自定义节点，在设置了 loading/success 时会填充默认值	-

events

name	说明	函数
click	当点击按下时	Function(event: Event): void

Demo

- 按钮

- 辅助按钮
- 加载
- 警示

按钮

截图

代码

```
<template>

<div style="padding: 10px;">
<AButton >主按钮</AButton>
<AButton type="white">次按钮</AButton>
<AButton disabled>按钮不可点</AButton>

<AButton href="#">a标签 主按钮</AButton>
<AButton href="#"type="white">a标签 次按钮</AButton>
<AButton href="#"disabled>a标签 按钮不可点</AButton>
</div>

</template>

<style scoped>
.am-button {
margin-bottom: 20px;
}
</style>
```

辅助按钮

截图

代码

```
<template>

<div style="padding: 10px;">

<AButton size="tiny"type="white">辅助按钮</AButton>
<AButton size="tiny"type="white"disabled>辅助按钮</AButton>
```

```
<AButton href="#"size="tiny"type="white">辅助按钮</AButton>
<AButton href="#"size="tiny"type="white"disabled>辅助按钮</AButton>

<AButton size="tiny">辅助按钮</AButton>
<AButton size="tiny"disabled>辅助按钮</AButton>

<AButton href="#"size="tiny">辅助按钮</AButton>
<AButton href="#"size="tiny"disabled>辅助按钮</AButton>

</div>

</template>

<style scoped>
.am-button {
margin-bottom: 20px;
}
</style>
```

加载

截图

代码

```
<template>

<div style="padding: 10px;">
<AButton loading>加载中...</AButton>
<AButton success>付款成功...</AButton>

<AButton loading type="blue">加载中...</AButton>
<AButton loading type="white">加载中...</AButton>
<AButton loading type="warn">加载中...</AButton>
<AButton loading type="bottom">加载中...</AButton>
<AButton loading type="blue"size="tiny">加载中...</AButton>
<AButton loading type="white"size="tiny">加载中...</AButton>
</div>

</template>

<style scoped>
.am-button {
margin-bottom: 20px;
}
</style>
```

警示

截图

代码

```
<template>

<div style="padding: 10px 0;">
<AButton type="warn">警示按钮</AButton>
<AButton type="warn"disbaled>警示按钮</AButton>
</div>

</template>

<style scoped>
.am-button {
margin-bottom: 20px;
}
</style>
```

2.3.2 Flexbox 弹性盒子

Flexbox 弹性盒子文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，需通过 ESModule 的方式来导入组件。
- API 文档 提供了 props、slots 的接口信息。

此外，要查看该组件的视觉效果及示例代码，请参考本文 Demo 。

Kylin

```
<dependency component="{ Flexbox, FlexboxItem }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Flexbox, FlexboxItem } from '@alipay/antui-vue';
```

API 文档

Flexbox

props

属性	说明	类型	默认值
direction	子元素的排列方式，可选row,row-reverse,column,column-reverse	String	'row'
wrap	子元素的换行方式，可选nowrap,wrap,wrap-reverse	String	'nowrap'

justify	子元素在主轴上的对齐方式，可选start,end,center,between,around	String	'start'
align	子元素在交叉轴上的对齐方式，可选start,center,end,baseline,stretch	String	'center'
alignContent	有多根轴线时的对齐方式，可选start,end,center,between,around,stretch	String	'stretch'

slots

name	说明	scope
-	默认 slot, 填充布局内部内容	-

FlexboxItem

props

FlexboxItem 组件默认加上了样式flex: 1,保证所有 item 平均分宽度, Flexbox 容器的 children 不一定是 FlexboxItem

slots

name	说明	scope
-	默认 slot, 填充元素内部内容	-

Demo

基础样式

截图

代码

```
<template>
<div>
<Flexbox>
<FlexboxItem>2列</FlexboxItem>
<FlexboxItem>2列</FlexboxItem>
</Flexbox>
<Flexbox>
<FlexboxItem>3列</FlexboxItem>
<FlexboxItem>3列</FlexboxItem>
<FlexboxItem>3列</FlexboxItem>
</Flexbox>
<Flexbox>
<div class="placeholder">定宽 70px</div>
<FlexboxItem>自适应</FlexboxItem>
</Flexbox>
<Flexbox>
```

```
<FlexboxItem>3</FlexboxItem>
<FlexboxItem class="am-ft-ellipsis">自定义个性化</FlexboxItem>
<FlexboxItem>3</FlexboxItem>
</Flexbox>
</div>
</template>

<style lang="less"rel="stylesheet/less"scoped>
.am-flexbox {
margin: 10px 0;
}

.demo-item {
padding: 6px 0;
background: #4A89DC;
border-radius: 4px;
text-align: center;
color: #fff;
}

.am-flexbox-item {
.demo-item;
}

.placeholder {
.demo-item;
width: 70px;
font-size: 12px;
margin-left: 8px;
margin-right: 8px;
}

</style>
```

排列方向

截图

代码

```
<template>
<div>
<Flexbox>
<div class="placeholder">1</div>
<div class="placeholder">2</div>
<div class="placeholder">3</div>
</Flexbox>

<Flexbox direction="row-reverse">
<div class="placeholder">1</div>
```

```
<div class="placeholder">2</div>
<div class="placeholder">3</div>
</Flexbox>
</div>
</template>

<style lang="less"rel="stylesheet/less"scoped>
.am-flexbox {
margin: 10px 0;
}

.demo-item {
padding: 6px 0;
background: #4A89DC;
border-radius: 4px;
text-align: center;
color: #fff;
}

.am-flexbox-item {
.demo-item;
}

.placeholder {
.demo-item;
width: 70px;
font-size: 12px;
margin-left: 8px;
margin-right: 8px;
}

</style>
```

元素换行

截图

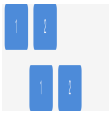
代码

```
<template>
<div>
<Flexbox>
<div class="placeholder">1</div>
<div class="placeholder">2</div>
<div class="placeholder">3</div>
<div class="placeholder">4</div>
<div class="placeholder">5</div>
<div class="placeholder">6</div>
<div class="placeholder">7</div>
<div class="placeholder">8</div>
```

```
<div class="placeholder">9</div>
</Flexbox>

<Flexbox wrap="wrap">
<div class="placeholder">1</div>
<div class="placeholder">2</div>
<div class="placeholder">3</div>
<div class="placeholder">4</div>
<div class="placeholder">5</div>
<div class="placeholder">6</div>
<div class="placeholder">7</div>
<div class="placeholder">8</div>
<div class="placeholder">9</div>
</Flexbox>

<Flexbox wrap="wrap-reverse">
<div class="placeholder">1</div>
<div class="placeholder">2</div>
<div class="placeholder">3</div>
<div class="placeholder">4</div>
<div class="placeholder">5</div>
<div class="placeholder">6</div>
<div class="placeholder">7</div>
<div class="placeholder">8</div>
<div class="placeholder">9</div>
</Flexbox>
</div>
</template>

<style lang="less"rel="stylesheet/less"scoped>
.am-flexbox {
margin: 10px 0;
}

.demo-item {
padding: 6px 0;
background: #4A89DC;
border-radius: 4px;
text-align: center;
color: #fff;
}

.am-flexbox-item {
.demo-item;
}

.placeholder {
.demo-item;
width: 70px;
font-size: 12px;
margin: 8px;
}

</style>
```

对齐方式

截图

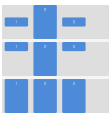


代码

```
<template>
<div>
<Flexbox>
<div class="placeholder">1</div>
<div class="placeholder">2</div>
</Flexbox>

<Flexbox justify="center">
<div class="placeholder">1</div>
<div class="placeholder">2</div>
</Flexbox>
</div>
</template>

<style lang="less"rel="stylesheet/less"scoped>
.am-flexbox {
margin: 10px 0;
}

.demo-item {
padding: 6px 0;
background: #4A89DC;
border-radius: 4px;
text-align: center;
color: #fff;
}

.am-flexbox-item {
.demo-item;
}

.placeholder {
.demo-item;
width: 70px;
font-size: 12px;
margin-left: 8px;
margin-right: 8px;
}

</style>
```

交叉轴对齐方式

截图



代码

```
<template>
<div>
<Flexbox>
<div class="placeholder">1</div>
<div class="placeholder high">2</div>
<div class="placeholder">3</div>
</Flexbox>

<Flexbox align="start">
<div class="placeholder">1</div>
<div class="placeholder high">2</div>
<div class="placeholder">3</div>
</Flexbox>

<Flexbox align="stretch">
<div class="placeholder">1</div>
<div class="placeholder high">2</div>
<div class="placeholder">3</div>
</Flexbox>
</div>
</template>

<style lang="less"rel="stylesheet/less"scoped>
.am-flexbox {
margin: 10px 0;
background-color: #dedede;
}

.demo-item {
padding: 6px 0;
background: #4A89DC;
border-radius: 4px;
text-align: center;
color: #fff;
}

.am-flexbox-item {
.demo-item;
}

.placeholder {
.demo-item;
width: 70px;
font-size: 12px;
margin-left: 8px;
margin-right: 8px;
}

.high {
height: 100px;
}

</style>
```

多轴对齐方式

截图

代码

```
<template>
<div>
<Flexbox wrap="wrap">
<div class="placeholder">1</div>
<div class="placeholder">2</div>
<div class="placeholder">3</div>
<div class="placeholder">4</div>
<div class="placeholder">5</div>
<div class="placeholder">6</div>
<div class="placeholder">7</div>
<div class="placeholder">8</div>
<div class="placeholder">9</div>
</Flexbox>

<Flexbox wrap="wrap"align-content="start">
<div class="placeholder">1</div>
<div class="placeholder">2</div>
<div class="placeholder">3</div>
<div class="placeholder">4</div>
<div class="placeholder">5</div>
<div class="placeholder">6</div>
<div class="placeholder">7</div>
<div class="placeholder">8</div>
<div class="placeholder">9</div>
</Flexbox>
</div>
</template>

<style lang="less"rel="stylesheet/less"scoped>
.am-flexbox {
margin: 10px 0;
height: 200px;
background-color: #dedede;
}

.demo-item {
padding: 6px 0;
background: #4A89DC;
border-radius: 4px;
text-align: center;
color: #fff;
}

.am-flexbox-item {
.demo-item;
}
```

```
.placeholder {
.demo-item;
width: 70px;
font-size: 12px;
margin: 8px;
}

</style>
```

2.4 选项卡组件

2.4.1 Tab 选项卡

Tab 选项卡文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，请通过 ESModule 的方式导入组件。
- API 说明 提供了 props、slots、events 的接口信息。

此外，如需查看该组件的视觉效果及示例代码，请参考本文的 Demo 。

Kylin

```
<dependency component="{ Tab, TabItem }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Tab, TabItem } from '@alipay/antui-vue';
```

API 说明

- Tab
- TabItem

Tab

props

属性	说明	类型	默认值
scroll	是否支持滚动	boolean	false
value	选中的tab-item的id，支持v-model	Number, String	null
initial-value	初始选中的tab-item的id，支持非v-model场景	Number, String	null
resistant	滑动超出屏幕范围时的速度比率 (px/s)	Number	0.2
reverseTime	超出屏幕范围回弹时间 (s)	Number	0.15
slideTime	滑动后的惯性时间 (s)	Number	0.3

slideRate	滑动后的惯性距离比例 (distance = slideRate * speed)	Number	0.1

slots

name	说明	scope
-	用于填充 TabItem	-

events

name	说明	函数
input	改变选中的 item 时触发	Function (value: [string, number]): void

TabItem

props

属性	说明	类型	默认值
id	标识每个 item 的 id	String, Number	-

slots

name	说明	scope
-	填充内容的占位	—-

Demo

Tab

效果示例

代码示例

```
<template>
<Tab v-model="selected">
<TabItem v-for="i in 3"v-bind:key="i":id="i">选项 {{ i }}</TabItem>
</Tab>
</template>

<script type="text/javascript">
export default {
data() {
return {
selected: 1,
};
```

```
},
};
</script>
```

2.4.2 TabPanel 标签页面

TabPanel 标签页面文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，请通过 ESModule 的方式导入组件。
- API 文档 提供了 props、slots、events 的接口信息。

此外，如需查看该组件的视觉效果及示例代码，参考本文的 Demo 。

Kylin

```
<dependency component="{ TabPanel, TabPanelItem }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { TabPanel, TabPanelItem } from '@alipay/antui-vue';
```

API 文档

- TabPanel
- TabPanelItem

TabPanel

props

属性	说明	类型	默认值
scroll	是否支持滚动	boolean	true
labels	顶部显示标签	Array	[]
value	选中的 tab item 的 index，用来支持v-model绑定	Number	0
initial-value	初始选中的 tab item 的 index，用来支持不需要v-model绑定的场景（性能更好）	Number	0
resistant	滑动超出屏幕范围时的速度比率 (px/s)	Number	0.2
reverseTime	超出屏幕范围回弹时间 (s)	Number	0.15
reverseTimingFunction	超出屏幕范围回弹缓动时间	String	-
slideTime	滑动后的惯性时间 (s)	Number	0.3

slideTimingFunction	滑动后的缓动函数	String	cubic-bezier(.86,0,.07,1)
slideRate	滑动后的惯性距离比例 (distance = slideRate * speed)	Number	0.1

slots

name	说明	scope
-	用于填充 TabPanelItem	-

events

name	说明	函数
input	选中 item 时触发 index 为选中 item 的索引 (再次选中当前 item 时依然会触发, 该问题在0.4.8-open02 版本已经修复)	Function (index: number): void

TabPanelItem

slots

name	说明	scope
-	填充内容的占位	—-

Demo

Tab-panel

截图

代码

```
<template>
<TabPanel
:labels="['标签1', '标签2', '标签3', '标签4', '标签5', '标签6', '标签7']"
v-model="selected"
>
<TabPanelItem
v-for="i in 7"
style="text-align: center; height: 100px; background: white"
>
内容{{ i }}
</TabPanelItem>
</TabPanel>
```

```
</template>

<script>
export default {
data() {
return {
selected: 0,
};
},
};
</script>
```

2.5 弹窗组件

2.5.1 ActionSheet 选项卡

ActionSheet 选项卡文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。
- 使用 Service 命令式调用 。 Service 文档 提供了 static methods、show options 的说明。

API 文档 提供了 props、slots、events 的接口信息。

此外，如需查看该组件的视觉效果及示例代码，参考本文的 Demo 。

Kylin

```
<dependency component="{ ActionSheet }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { ActionSheet } from '@alipay/antui-vue';
```

Service 命令式调用

可以直接使用 ActionSheet.show('显示内容'); 的方式调用命令，不需要写在模板中。会自动在 document.body 上插入对应 DOM。

```
ActionSheet.show({
title: '标题',
items: ['123','321'],
cancelButtonText: '取消'
}, function (index) {

});
```

Service 文档
static methods

ActionSheet 提供以下静态方法：

name	说明	函数
show	创建一个 ActionSheet 实例并显示，创建的参数如下	Function(option: Object|string, callback: Function): vm

show options

创建实例时，接受参数如下：

属性	说明	类型	默认值
title	面板上方显示的说明文字	string	-
items	选项的文字数组	array	-
cancelButtonText	取消按钮的文字	string	'取消'
destructiveBtnIndex	特殊操作的索引值	number	-
transitionDuration	渐变持续时间	string	'0.3s'
zIndex	设置弹层的 zIndex 值	number	9999

API 文档

props

属性	说明	类型	默认值
show	是否显示面板	boolean	false
title	面板上方显示的说明文字	string	-
items	选项的文字数组	array	-
cancelButtonText	取消按钮的文字	string	'取消'
destructiveBtnIndex	特殊操作的索引值	number	-
transitionDuration	渐变持续时间	string	'0.3s'

events

name	说明	函数
click	点击选项时触发，点击 mask 或取消按钮时为 -1	Function(index: number): void

slots

name	说明	作用域
label	用于对 props.items 进行 DOM 扩展的方式	{item: String}

Demo

1.show

截图

代码

```
<template>
<div>
<AButton @click="show = true">点击</AButton>
<ActionSheet
:show="show"
title="这是提供一行或二行注释, 通过信息澄清的方式避免产生用户疑问。"
:items="['选项1', '选项2', '选项3', '瞎搞']"
cancelButtonText="取消2"
:destructiveBtnIndex="3"
transitionDuration="0.3s"
@click="onClick"
/>
</div>
</template>

<script>
export default {
data() {
return {
show: false,
};
},
methods: {
onClick(index) {
if (index === -1) {
this.show = false;
}
},
},
};
</script>
```

2.5.2 ADialog 弹窗

ADialog 弹窗文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。

此外，如需查看该组件的视觉效果及示例代码，参考本文的 Demo 。

Kylin

```
<dependency component="{ ADialog }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { ADialog } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
animation	是否开启transition	boolean	true
type	弹框类型，可选值为text,image	string	text
value	弹框的显示隐藏状态，支持v-model	boolean	false
closable	是否露出关闭的x按钮	boolean	false
title	弹框标题	string	-
content	弹框内容	string	-
buttons	弹框按钮组, 要求数组元素为{ [key]:string, [text]:string, [disabled]: boolean }	array<{ key:string, text:string, disabled:boolean }>	[]
selection	选项卡对话框，设置按钮垂直分布	boolean	-
preventMove	是否在显示弹框/mask时阻止document的touchmove事件	boolean	true

slots

name	说明	scope
image	用于填充图像的区域	—-
-	默认占位，弹框内容，如果需要支持自定义DOM	—-
button	用于按钮组的区域	buttons

events

name	说明	函数
buttonClick	当点击按钮时触发	Function(event: event, buttonKey: string): void
maskClick	当点击mask时触发	Function(event: event): void
input	当value发生变化时触发	Function(value: boolean): void
show	当value变为true时触发	Function(): void
hide	当value变为false时触发	Function(): void

Demo

- 标题内容

- 无标题文本
- 标题+题图+内容
- 单行动按钮：标题+题图+内容
- 单行动按钮： 标题+内容
- 不可点按钮：标题+内容
- 选项卡
- 发送成功

标题内容

截图

代码

```
<template>
<ADialog
title="标题文字"
:value="true"
content="辅助说明文字"
:buttons="buttons"
@buttonClick="buttonClick"
></ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'cancel',
text: '取消',
}, {
key: 'ok',
text: '确定',
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

无标题文本

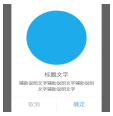
截图

代码

```
<template>
<ADialog
:value="true"
content="辅助说明文字"
:buttons="buttons"
@buttonClick="buttonClick"
></ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'cancel',
text: '取消',
}, {
key: 'ok',
text: '确定',
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

标题+题图+内容

截图



代码

```
<template>
```

```
<ADialog
title="标题文字"
type="image"
:value="true"
content="辅助说明文字"
:buttons="buttons"
@buttonClick="buttonClick"
>
<img slot="image"src="https://os.alipayobjects.com/rmsportal/QnFHZkFYDZUJqBD.png"alt="图片描述"/>
</ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'cancel',
text: '取消',
}, {
key: 'ok',
text: '确定',
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

单行动按钮：标题+题图+内容

截图

代码

```
<template>
<ADialog
title="标题文字"
type="image"
:value="true"
content="辅助说明文字"
:buttons="buttons"
@buttonClick="buttonClick"
>
```

```
<img slot="image"src="https://os.alipayobjects.com/rmsportal/QnFHZkFYDZUJqBD.png"alt="图片描述"/>
</ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'action',
text: '行动按钮',
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

单行动按钮： 标题+内容

截图

代码

```
<template>
<ADialog
title="标题文字"
:value="true"
content="辅助说明文字"
:buttons="buttons"
@buttonClick="buttonClick"
></ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'ok',
text: '确定',
}],
```

```
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

不可点按钮：标题+内容

截图

代码

```
<template>
<ADialog
title="标题文字"
:value="true"
content="辅助说明文字"
:buttons="buttons"
@buttonClick="buttonClick"
></ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'cancel',
text: '取消',
}, {
key: 'ok',
text: '确定',
disabled: true,
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

选项卡

截图

代码

```
<template>
<ADialog
:value="true"
content="内容说明当前状态、提示用户解决方案，最好不要超过两行。"
:selection="true"
:buttons="buttons"
@buttonClick="buttonClick"
></ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'action1',
text: '选项1',
}, {
key: 'action2',
text: '选项2',
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

发送成功

截图

代码

```
<template>
<ADialog
:value="true"
:selection="true"
:buttons="buttons"
@buttonClick="buttonClick"
>
<div class="am-dialog-send-img">
<img src="https://zos.alipayobjects.com/rmsportal/hkzwYezdQOqZfseGofgP.png"width="45px"height="29px">
</div>
<p class="am-dialog-brief">已发送</p>
</ADialog>
</template>

<script type="text/javascript">
import Toast from '../../../lib/toast';

export default {
data() {
return {
buttons: [{
key: 'taobao',
text: '返回手机淘宝',
}, {
key: 'alipay',
text: '留在支付宝',
}],
};
},
methods: {
buttonClick(evt, key) {
Toast.show(`您点击了${key}`);
},
},
};
</script>
```

2.5.3 Filter 筛选

Filter 筛选文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。
此外，如需查看该组件的视觉效果及示例代码，参考本文的 Demo 。

Kylin

```
<dependency component="{ AFilter, FilterList, FilterItem }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { AFilter, FilterList, FilterItem } from '@alipay/antui-vue';
```

API 文档

AFilter

type 为 dialog 时（默认）

props

属性	说明	类型	默认值
value	是否显示	boolean	false
animation	是否有打开动画	boolean	true

slots

name	说明	scope
-	默认 slot, 填充布局内部内容	-

events

name	说明	函数
input	点击阴影部分时触发，值为 false	Function(value: boolean): void

type 为 page 时

props

属性	说明	类型	默认值
value	是否显示	boolean	false
animation	是否有打开动画	boolean	true

slots

name	说明	scope
-	默认 slot, 填充布局内部内容	-
footer	用于改变底部按钮	-

events

name	说明	函数
reset	点击重置按钮时触发	Function(): void
confirm	点击确认按钮时触发	Function(): void

FilterList

props

属性	说明	类型	默认值
recommend	推荐选项	boolean	false
title	选项分组标题	string	""

slots

name	说明	scope
-	默认 slot, 填充元素内部内容	-

FilterItem

属性	说明	类型	默认值
selected	是否选中，值为 null 时，根据 option 与 value 进行判断	boolean, null	null
option	该选项的标识值	string, number	-
value	当前选中的选项的标识值，若与 option 相同则为选中，selected 不为 null 时优先判断 selected	string, number	-
disabled	是否被禁用	boolean	false

slots

name	说明	scope
-	默认 slot, 填充元素内部内容	-

events

name	说明	函数
input	选择选项时触发，disabled 时不触发	Function(option: [string, number]): void

Demo

- 筛选框
- 多分类筛选
- 全屏筛选

筛选框

截图

代码

```
<template>
<div style="min-height: 200px;">
<AFilter type="dialog"v-model="show">
<FilterList>
<FilterItem v-for="i in 8":option="i"v-model="selected">
{{ i }}
</FilterItem>
</FilterList>
</AFilter>
<div class="button-wrap">
<button class="am-button white"@click="show = true">点击显示筛选框</button>
</div>
</div>
</template>

<style lang="less"scoped>
.button-wrap {
padding: 15px;
}
</style>

<script>
export default {
data() {
return {
show: true,
selected: null,
};
},
};

</script>
```

多分类筛选

截图

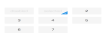
代码

```
<template>
<div style="min-height: 400px;">
<AFilter v-model="show">
<FilterList recommend>
```

```
<FilterItem v-for="i in 3":option="'recommend-' + i"v-model="selected">
{{ i }}
</FilterItem>
</FilterList>
<FilterList title="分类名称">
<FilterItem v-for="i in 5":option="'a-' + i"v-model="selected">
{{ i }}
</FilterItem>
</FilterList>
<FilterList title="分类名称">
<FilterItem v-for="i in 6":option="'b-' + i"v-model="selected">
{{ i }}
</FilterItem>
</FilterList>
</AFilter>
<div class="button-wrap">
<button class="am-button white"@click="show = true">点击显示筛选框</button>
</div>
</div>
</template>

<style lang="less"scoped>
.button-wrap {
padding: 15px;
}
</style>

<script>
export default {
data() {
return {
show: false,
selected: null,
};
},
};

</script>
```

全屏筛选

截图



代码

```
<template>
<div>
<div class="button-wrap">
<button class="am-button white"@click="show = true">点击显示筛选框</button>
</div>
```

```
<AFilter type="page"v-model="show">
<FilterList>
<FilterItem option="disabled":selected="!!~selected.indexOf('disabled')"@input="onSelect"disabled>
disabled
</FilterItem>
<FilterItem option="selected":selected="!!~selected.indexOf('selected')"@input="onSelect"disabled selected>
selected
</FilterItem>
<FilterItem v-for="i in 6":option="i":selected="!!~selected.indexOf(i)"@input="onSelect">
{{ i + 1 }}
</FilterItem>
</FilterList>
</AFilter>
</div>
</template>

<style lang="less"scoped>
.button-wrap {
padding: 15px;
}
</style>


<script>
export default {
data() {
return {
selected: [],
show: true
};
},
methods: {
onSelect(v) {
if (~this.selected.indexOf(v)) {
this.selected = this.selected.filter(i => i !== v);
} else {
this.selected.push(v);
}
},
},
};

</script>
```

2.5.4 Toast 提示

Toast 提示文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。
- 使用 Service 命令式调用 。 Service 文档 提供了 static methods、show options 的说明。

API 文档 提供了 props、slots、events、methods 的接口信息。

Kylin

```
<dependency component="{ Toast }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Toast } from '@alipay/antui-vue';
```

Service 命令式调用

直接使用 Toast.show('显示内容'); 的方式调用命令，不需要写在模板中。会自动在 document.body 上插入对应 DOM。

```
Toast.show({
type: 'text',
text: '显示的消息'
});
```

Service 文档

static methods

Toast 提供以下静态方法：

name	说明	函数
show	创建一个 Toast 实例并显示，创建的参数如下（也可以直接传字符串当做 text）	Function(option: Object| string): vm
closeAll	关闭当前所有未关闭的 Toast 实例	Function(void): void

show options

创建实例时，接受参数如下：

属性	说明	类型	默认值
type	Toast类型，可选值为text,loading, warn, success, network, fail	string	text
text	纯字符串文本	string	-
duration	超时自动隐藏并销毁实例	number	3000
zIndex	设置弹层的 zIndex 值	number	9999

instance methods

Toast.show方法返回了一个vm实例，其对外暴露以下方法：

name	说明	函数
hide	在 duration 未到时主动隐藏并销毁实例	Function(void): void

API 文档

props

属性	说明	类型	默认值
type	Toast类型，可选值为text,loading, warn, success, network, fail	string	text
value	Toast的显示隐藏状态，支持v-model，用于定时隐藏的触发	boolean	false
duration	超时自动触发$emit('input',false)，如果设置为0则不触发	number	3000
text	纯字符串文本，如需DOM请使用slot	string	-
onClose	当隐藏时会触发	Function	-
multiline	true时修改默认样式支持多行显示文本	Boolean	false

slots

name	说明	scope
-	默认占位，内容，如果需要支持自定义 DOM	—-

events

name	说明	函数
input	适配 v-model	Function(newValue: boolean): void

methods

name	说明	函数
hide	主动隐藏当前Toast	Function(void): void

Demo

- 文本
- 加载中
- 警告
- 成功

文本

截图

代码

```
<template>
```

```
<div style="padding: 10px;">
<AButton @click="trigger">{{show?'关闭':'显示'}}Toast</AButton>
<Toast v-model="show">自定义业务文案最多14个字符</Toast>
</div>

</template>

<script>
import { Toast, AButton } from"@alipay/antui-vue";
export default {
data() {
return { show: true };
},
methods: {
trigger() {
this.show = !this.show;
},
},
components: {
Toast,
AButton,
},
};
</script>
```

加载中

截图

代码

```
<template>

<div style="padding: 10px;">
<AButton @click="trigger">{{show?'关闭':'显示'}}Toast</AButton>
<Toast type="loading"v-model="show">加载中...</Toast>
</div>

</template>

<script>
import { Toast, AButton } from"@alipay/antui-vue";
export default {
data() {
return { show: true };
},
methods: {
trigger() {
this.show = !this.show;
},
```

```
},
components: {
Toast,
AButton,
},
};
</script>
```

警告

截图

代码

```
<template>

<div style="padding: 10px;">
<AButton @click="trigger">{{show?'关闭':'显示'}}Toast</AButton>
<Toast type="warn"v-model="show">警示</Toast>
</div>

</template>

<script>
import { Toast, AButton } from"@alipay/antui-vue";
export default {
data() {
return { show: true };
},
methods: {
trigger() {
this.show = !this.show;
},
},
components: {
Toast,
AButton,
},
};
</script>
```

成功

截图

代码

```
<template>

<div style="padding: 10px;">
<AButton @click="trigger">{{show?'关闭':'显示'}}Toast</AButton>
<Toast type="success"v-model="show">成功</Toast>
</div>

</template>

<script>
import { Toast, AButton } from"@alipay/antui-vue";
export default {
data() {
return { show: true };
},
methods: {
trigger() {
this.show = !this.show;
},
},
components: {
Toast,
AButton,
},
};
</script>
```

2.6 条目组件

2.6.1 List 列表

List 列表文档介绍了该组件的不同使用方式以及其 API 。

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用需要通过 ESModule 的方式导入组件。
- API 提供了 props、slots、events 的接口信息。

此外，如需查看该组件的视觉效果及示例代码，请参考本文的 Demo 。

列表项由于结构复杂，包含单行或多行文字列表、表单、校验码以及 checkbox、radio、switch 等表单元素。分别使用 List、ListCell、ListItem 来进行功能拆分。

- List：列表外包裹，等价于am-list，支持类型见 API 。
- ListItem：一行列表项目，能满足一般的列表布局，支持类型见 API 。
- ListCell：列表项中的各种块，便于灵活搭配结构，满足复杂布局，支持类型见 API 。

Kylin

```
<dependency component="{ List, ListItem, ListCell }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { List, ListItem, ListCell } from '@alipay/antui-vue';
```

API

List

props

属性	说明	类型	默认值
type	列表类型，可选值为”form”，”twoline”，”twoline-text”，”twoline-side”，”moreline”，”ptext”，”users”，”users-sm”，”users-lg”，”bank”，”info”，”delete”	String	-

slots

name	说明	scope
header	用于填充列表头，推荐使用ListCell	-
footer	用于填充列表尾，推荐使用ListCell	-

ListCell

props

属性	说明	类型	默认值
type	列表元素类型，可选值为”content”，”brief”，”extra”，”arrow”，”thumb”，”title”，”brief”，”sti”，“header”，”footer”，”header-sp”	String	“content”
noFlex	本单元格是否清除 flex	boolean	false
noEllips	单元格内容是否不需要单行	boolean	false
alignTop	单元格内容是否居上	boolean	false
twoColumn	单元格内容左右两列显示	boolean	false
title	注入ListCell[type=title]节点，填充文本	String	-
brief	注入ListCell[type=brief]节点，填充文本	String	-
titleEllips	对title注入节点是否开启am-ft-ellipsis省略	boolean	false
briefEllips	对brief注入节点是否开启am-ft-ellipsis省略	boolean	false

slots

name	说明	scope
-	用于填充子节点	-

events

name	说明	函数
click	当点击时触发	Function(event: event): void

ListItem

props

属性	说明	类型	默认值
type	列表行类型，可选值为 “” ，” twoline” ，” moreline” ，” check” ，” radio” ，” more” ，” part” ，” delete”	String	“”
arrow	是否显示右侧箭头, 如果配置了 href 属性会默认启用(当 arrow=true 时 href 属性不可用)	boolean	false
href	是否需要点击即跳转页面，设置后 DOM 为 A 标签，不设置为 DIV 标签	String	-
content	默认占位符中仅填充 ListCell[type=content] 的字符串时，可以用此简写	Stirng	-
extra	右侧占位文本	String	-
reddot	右侧 extra 部分显示小红点	boolean	-
deletable	是否显示列表左侧删除按钮	boolean	-
contentNoFlex	对 content 自动加入 ListCell[noFlex]属性	boolean	false
extraNoFlex	对 extra 自动加入 ListCell[noFlex]属性	boolean	false
contentNoEllips	对 content 自动加入 ListCell[noEllips]属性	boolean	false
extraNoEllips	对 extra 自动加入 ListCell[noEllips]属性	boolean	false
cancelText	当 [type=delete] 时，显示 “取消” 文案	String	“取消”
deleteText	当 [type=delete] 时，显示 “删除” 文案	String	“删除”
indentLine	下划线的缩进类型，可选值为 “” 、” thumb” 、” twoline”	String	“”

slots

name	说明	scope
thumb	左侧略缩图占位	-
-	默认占位，如果需要快速填充 ListCell[type=content] 节点，可直接使用 content 属性	-
extra	右侧占位如果不满足于字符串文本，可使用 DOM 节点代替	-

events

name	说明	函数
click	当点击时触发	Function(event: event): void
cancel	当 [type=delete] 时点击取消	Function(): void
delete	当 [type=delete] 时点击删除	Function(): void

Demo

- 表头表尾
- 单行文字列表
- 单元格超长
- 双行纯文字
- 图文信息
- 信息列表
- 左右双行

表头表尾

截图

代码

```
<template>
<List>
<ListCell type="header"slot="header">单行列表头纯文字</ListCell>
<ListItem content="标签文本"></ListItem>
<ListItem content="标签文本"extra="辅助标签三四五六七八九十"></ListItem>
<ListItem content="可点链接"extra="辅助标签三四五六七八九十"href="#"arrow></ListItem>
<ListItem content="标签文本"arrow></ListItem>
<ListCell type="footer"slot="footer">单行列表尾纯文字</ListCell>
</List>

</template>
```

单行文字列表

截图

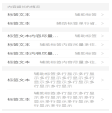
代码

```
<template>

<div>
<List>
<ListItem content="标签文本">
<AButton slot="extra"size="tiny"type="white">辅助按钮</AButton>
</ListItem>
</List>

<List>
<ListItem content="标签文本"extra="详细信息"arrow></ListItem>
</List>

<List>
<ListItem deletable content="删除列表"arrow></ListItem>
<ListItem deletable content="删除列表"arrow></ListItem>
</List>

<List>
<ListItem content="标签文本"reddot arrow></ListItem>
<ListItem content="标签文本"reddot arrow></ListItem>
</List>

<List>
<ListCell type="header"slot="header">单行列表头纯文字</ListCell>
<ListItem content="受控选中状态">
<ASwitch slot="extra"v-model="ctrl"></ASwitch>
</ListItem>
<ListItem content="选中状态">
<ASwitch slot="extra":value="true"></ASwitch>
</ListItem>
<ListItem content="未选中状态">
<ASwitch slot="extra":value="false"></ASwitch>
</ListItem>
<ListItem content="受控选中状态">
<ASwitch platform="android"slot="extra"v-model="ctrl"></ASwitch>
</ListItem>
<ListItem content="安卓样式开">
<ASwitch platform="android"slot="extra":value="true"></ASwitch>
</ListItem>
<ListItem content="安卓样式关">
<ASwitch platform="android"slot="extra":value="false"></ASwitch>
</ListItem>
<ListCell type="footer"slot="footer">单行列表尾纯文字</ListCell>
</List>

</div>
```

```
</template>

<script>
export default {
data() {
return {
ctrl: true
};
}
};
</script>
```

单元格超长

截图



代码

```
<template>

<div>

<List>
<ListCell type="header"slot="header">内容超长的情况</ListCell>
<ListItem content="标签文本"extra="辅助标签"arrow></ListItem>
<ListItem content="标签文本"extra="辅助标签单行省略号显示"arrow></ListItem>
</List>

<List>
<ListItem content="标签文本内容尽量多往右显示"extra="辅助标签"extra-no-flex arrow></ListItem>
<ListItem content="标签文本"content-no-flex extra="辅助标签内容尽量多往左显示"arrow></ListItem>

<ListItem content="标签文本内容尽量多往右显示"arrow>
<ListCell slot="extra"type="extra"no-flex>辅助标签</ListCell>
</ListItem>
<ListItem extra="辅助标签内容尽量多往左显示"arrow>
<ListCell type="content"no-flex>标签文本</ListCell>
</ListItem>
</List>

<List>
<ListItem
content="标签文本"content-no-flex
extra="辅助标签多行显示"extra-no-ellips
arrow
></ListItem>

<ListItem arrow>
<ListCell type="content"no-flex>标签文本</ListCell>
```

```
<ListCell slot="extra"type="extra"no-ellips>辅助标签多行显示</ListCell>
</ListItem>

</List>

</div>

</template>
```

双行纯文字

截图

代码

```
<template>

<List type="twoline-text">
<ListItem >
<ListCell type="content"title="标签文本"brief="内容文本内容文本内容文本"></ListCell>
</ListItem>
<ListItem arrow>
<ListCell type="content"title="标签文本"brief="内容文本内容文本内容文本"></ListCell>
</ListItem>
<ListItem arrow>
<ListCell
type="content"
title="标题文本标题文本标题文本标题文本标题文本标题文本标题文本"title-ellips
brief="标题文本标题文本标题文本标题文本标题文本标题文本标题文本"brief-ellips
></ListCell>
</ListItem>
</List>

</template>
```

图文信息

截图

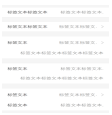
代码

```
<template>
<div>
<List type="ptext">
<ListCell type="header-sp"slot="header">图文列表组合, 附加来源</ListCell>
```

```
<ListItem content="标签文本">
<ListCell type="thumb"slot="thumb"src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
<ListCell
title="标题"
brief="“全球未来机场计划”是指未来游客在海外机场，通过支付宝使用航班提醒等服务。"
>
<ListCell type="sti"slot="after">
<span>来源 时间 | 其他信息</span>
</ListCell>
</ListCell>
</ListItem>
</List>

<List type="ptext">
<ListCell type="header-sp"slot="header">图文列表组合, 附加来源</ListCell>
<ListItem content="标签文本">
<ListCell type="thumb"slot="thumb"src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
<ListCell
title="标题"
brief="“全球未来机场计划”是指未来游客在海外机场，通过支付宝使用航班提醒等服务。"
/>
</ListItem>
<ListItem content="标签文本">
<ListCell type="thumb"slot="thumb"src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
<ListCell
title="标题"
brief="“全球未来机场计划”是指未来游客在海外机场，通过支付宝使用航班提醒等服务。"
/>
</ListItem>
<ListItem type="more"slot="footer">查看更多</ListCell>
</List>
</div>

</template>
```

信息列表

截图



代码

```
<template>
<div>

<List type="info">
<ListItem type="oneline"content="标签文本标签文本"extra="标签文本标签文本标签文本标签文本"/>
</List>

<List type="info">
<ListItem type="oneline"content="标签文本标签文本"extra="标签文本标签文本标签文本标签文本"arrow/>
```

```
</List>

<List type="info">
<ListItem type="more">
<ListItem type="part"content="标签文本"extra="标签文本标签文本标签文本标签文本"arrow/>
<ListItem type="part":content="false"extra="标签文本标签文本标签文本标签文本"/>
</ListItem>
</List>

<List type="info">
<ListItem>
<ListItem type="part"content="标签文本"extra="标签文本标签文本标签文本标签文本"/>
<ListItem type="part":content="false"extra="标签文本标签文本标签文本标签文本"/>
</ListItem>
</List>

<List type="info">
<ListItem type="more">
<ListItem type="part"content="标签文本"extra="标签文本标签文本标签文本标签文本"arrow/>
<ListItem type="part"content="标签文本"extra="标签文本标签文本标签文本标签文本"/>
</ListItem>
</List>
</div>

</template>
```

左右双行

截图

代码

```
<template>
<div>

<List type="bank">
<ListCell type="header"slot="header">银行卡列表(圆图 72 x 72(ios), 108 x 108(ios))</ListCell>
<ListItem arrow>
<ListCell type="thumb"src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"slot="thumb"/>
<ListCell title="招商银行"brief="尾号7785"/>
</ListItem>
</List>

<List type="twoline-side">
<ListCell type="header-sp"slot="header">左右文字组合列表</ListCell>
<ListItem >
<ListCell title="标题一"brief="内容一"/>
<ListCell type="extra"title="标题二"brief="内容二"slot="extra"/>
</ListItem>
<ListItem >
```

```
<ListCell title="标题一"brief="内容一"/>
<ListCell type="extra"title="标题二"brief="内容二"slot="extra"/>
</ListItem>

<ListItem >
<ListCell type="thumb"src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
<ListCell title="标题一"brief="内容一"/>
<ListCell type="extra"title="标题二"brief="内容二"slot="extra"/>
</ListItem>
<ListItem >
<ListCell type="thumb"src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
<ListCell title="标题一"brief="内容一"/>
<ListCell type="extra"title="标题二"slot="extra"/>
</ListItem>

</List>

</div>

</template>
```

2.7 输入组件

2.7.1 Checkbox 选择框

Checkbox 选择框文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。
- API 文档 提供了 props、slots、events 的接口信息。

Kylin

```
<dependency component="{ ListItemCheckbox }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { ListItemCheckbox } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
type	选择框类型, 可选 checkbox,radio,agreement	String	checkbox
value	对于 radio 框，返回的是选中的 tag 字段，对于 checkbox/agreement 返回的是 true/false	String, boolean	-
tag	对于 radio 框，同一个 name 下面不同的选项值	String	-

label	选择框的标签，如果需要自定义 DOM，请使用 slot[name=label]	string	-
id	input 的 id 属性	String	-
name	input 的 name 属性	String	-
disabled	是否禁用选择框	boolean	false
brief	用于辅助标签，agreement 类型不可用	String	-
thumb	显示图片的 url，agreement 类型不可用	String	-
thumbAlt	图片的 alt 属性，agreement 类型不可用	String	-

slots

name	说明	scope
label	可将 props.label 从字符串扩展为 DOM 节点，如果自定义 label ，需要将 label[for] 字段填充为 props.id	{id:String}

events

name	说明	函数
input	适配 v-model，如果类型为 radio 返回选中 tag 的字符串，否则返回是否选中的布尔值	Function(value: [string(radio), boolean(其它类型)]): void

Demo

- 多选框
- 单选框
- 协议复选框

多选框

不带图片

截图

代码

```
<template>
<List>
<ListItemCheckbox
id="test1"
label="表单项多选框——未选中标签"
v-model="c1"
/>
```

```
<ListItemCheckbox
id="test2"
label="表单项多选框——选中标签"
v-model="c2"
/>
<ListItemCheckbox
id="test3"
label="表单项多选框——未选中标签"
v-model="c3"
/>
<ListItemCheckbox id="test4"label="表单项多选框——禁用"disabled />
</List>
</template>

<script>
export default {
data() {
return {
c1: false,
c2: true,
c3: false,
};
},
};
</script>
```

带图片

截图

代码

```
<template>
<div>
<List>
<ListItemCheckbox
id="test1"
label="表单项多选框——选中标签"
thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
thumbAlt="图片描述"
/>
<ListItemCheckbox
id="test2"
label="表单项多选框——禁用"
thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
thumbAlt="图片描述"
disabled
/>
</List>
<List class="towline">
```

```
<ListItemCheckbox
id="test3"
label="表单项多选框——选中标签"
brief="辅助说明"
thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
thumbAlt="图片描述"
/>
<ListItemCheckbox
id="test4"
label="表单项多选框——禁用"
brief="辅助说明"
thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
thumbAlt="图片描述"
disabled
/>
</List>
</div>
</template>
```

单选框

截图

代码

```
<template>
<div>
<List>
<ListCell type="header"slot="header">单选-自控制</ListCell>
<ListItemCheckbox
:id="'test'+i"
type="radio"
thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
v-for="i in 3"
:label="'标签文本'+i"
:tag="i + ''"
name="demo"
@input="onChange"
/>
</List>
<List>
<ListCell type="header"slot="header">单选-受控</ListCell>
<ListItemCheckbox
:id="'test'+i"
type="radio"
thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
v-for="i in 3"
:label="'标签文本'+i"
:tag="i + ''"
name="demo2"
```

```
v-model="checked"
/>
<AButton @click="checked = '2'">
选择2
</AButton>
</List>
</div>
</template>

<script>
export default {
data() {
return {
checked: '1',
};
},
methods: {
onChange(v) {
console.log(v);
},
},
};
</script>
```

协议复选框

截图

代码

```
<template>
<List>
<ListItemCheckbox
type="agreement"
id="agree"
label
v-model="checked"
>
<template slot="label">
<label class="am-ft-md"for="agree">同意</label>
<a href="#">《信用支付服务合同》</a>
</template>
</ListItemCheckbox>
<ListItemCheckbox
id="test2"
type="agreement"
label="表单项多选框——禁用"
disabled
/>
</List>
```

```
</template>

<script>
export default {
data() {
return {
checked: false,
};
},
};
</script>
```

2.7.2 Input 输入框

Input 输入框文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。

> 注意：使用时一般需要搭配List[type='form']一起。

Kylin

```
<dependency component="{ ListItemInput }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { ListItemInput } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
value	输入框的值，支持v-model绑定	string	-
label	输入框左侧的标签，为空不显示	string	-
labelWidth	输入框左侧的标签的width样式，支持输入数字(按照em处理)，字符串(自行制定单位)	string, number	auto
placeholder	输入框placeholder	string	-
inputType	输入框type, 用于浏览器识别键盘类型	string	text
clear	输入框清除按钮，当且仅当输入框值不为空且clear为true且未disabled时显示按钮	boolean	false
labelId	输入框左侧标签id，用于无障碍	string	-

inputId	输入框标签id，用于无障碍	string	-
disabled	是否禁用输入框	boolean	false
error	表明当前input框输入值为不合理	boolean	false
button	为空时不显示，右侧按钮文本	string	-
buttonDisabled	右侧按钮是否禁用	boolean	false

slots

name	说明	scope
-	可将props.label从字符串扩展为DOM节点	-

events

name	说明	函数	备注
input	当value变化时触发input组件事件；当清除value时也触发, 支持v-model	Function(value: boolean): void	
click	当点击时触发	Function(event: event): void	
errorClick	当配置error选项时，点击右侧红色圆圈触发	Function(void): void	
buttonClick	当配置button选项且右侧button未禁用时，点击右侧button 触发	Function(void): void	Since 0.4.8-open02

Demo

常规

截图

代码

```
<template>

<div>

<List type="form">
<ListCell type="header"slot="header">常规类</ListCell>
<ListItemInput label="标签"placeholder="内容"clear></ListItemInput>
</List>

<List type="form">
<ListItemInput label="标签"placeholder="内容"v-model="syncText"clear></ListItemInput>
<ListItemInput label="标签"placeholder="内容"v-model="syncText"clear></ListItemInput>
```

```
</List>

<List type="form">
<ListItemInput label="标签"placeholder="内容"value="初始值不更新"></ListItemInput>
</List>

</div>

</template>

<script type="text/javascript">
export default {
data() {
return {
syncText: '同步内容修改',
};
},
};
</script>
```

标签

截图

代码

```
<template>

<div>

<List type="form">
<ListCell type="header"slot="header">常见表单</ListCell>
<ListItemInput label=""placeholder="无标签, 暗提示"clear></ListItemInput>
<ListItemInput label="优惠券名称"placeholder="请输入名称"clear></ListItemInput>
</List>

<List type="form">
<ListCell type="header"slot="header">固定5个字</ListCell>
<ListItemInput label="优惠券"label-width="5em"placeholder="暗提示"clear></ListItemInput>
<ListItemInput label="优惠券代码"label-width="5em"placeholder="暗提示"clear></ListItemInput>
</List>

<List type="form">
<ListCell type="header"slot="header">固定6个字</ListCell>
<ListItemInput label="优惠券":label-width="6"placeholder="暗提示"clear></ListItemInput>
<ListItemInput label="优惠券有效期":label-width="6"value="永久可用"></ListItemInput>
</List>

</div>
```

```
</template>

<script type="text/javascript">
export default {
data() {
return {
syncText: '同步内容修改',
};
},
};
</script>
```

表单错误

截图

代码

```
<template>

<div>

<List type="form">
<ListCell type="header"slot="header">表单中某列数据输入错误的情况</ListCell>
<ListItemInput label="身份证号"value="33010220170101001XX"error @error-click="onClick"></ListItemInput>
<ListCell type="footer"slot="footer">注意：表单出错时，首次自动出toast提示错误信息，2s后toast消失；点击右边icon再次toast提示。 </ListCell>
</List>

</div>

</template>

<script>
import Toast from '../../../lib/toast';

export default {
methods: {
onClick() {
Toast.show('填写出错');
},
},
};
</script>
```

2.8 加载组件

2.8.1 Loading 加载指示

Loading 加载指示文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots 的接口信息。

Kylin

```
<dependency component="{ Loading, LoadingIndicator }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Loading, LoadingIndicator } from '@alipay/antui-vue';
```

API 文档

Loading

封装过的Loading，包括全页面加载，页脚加载等

props

属性	说明	类型	默认值
type	Loading类型，可选值为"","page","refresh","nomore"	string	""

slots

name	说明	scope
-	默认占位，内容，如果需要支持自定义DOM	—-

LoadingIndicator

独立的Loading三方块动画

props

属性	说明	类型	默认值
white	是否显示为白色方块，默认蓝色	boolean	false
tiny	是否显示小尺寸	boolean	false

Demo

独立加载状态

截图

代码

```
<template>

<div style="text-align: center; background: #999;">
<LoadingIndicator />
<br/>
<LoadingIndicator white />
<br/>
<LoadingIndicator tiny />
<br/>
<LoadingIndicator tiny white />
</div>

</template>
```

页面中加载

截图

代码

```
<template>
<div style="min-height: 300px;">
<Loading type="page">加载中</Loading>
</div>
</template>
```

页面顶部加载

截图

代码

```
<template>

<Loading type="refresh">上次更新 2016-06-23 11:47</Loading>

</template>
```

页面底部加载

截图

代码

```
<template>

<div>
<div style="color:#F4333C;background-color: #fff; text-align: center; height: 300px;">内容显示区域</div>
<Loading >加载中...</Loading>
</div>

</template>
```

没有更多

截图

代码

```
<template>

<div>
<div style="color:#F4333C;background-color: #fff; text-align: center; height: 300px;">内容显示区域</div>
<Loading type="nomore">没有更多了</Loading>
</div>

</template>
```

2.9 结果页组件

2.9.1 Message 信息状态

Message 信息状态文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。

Kylin

```
<dependency component="{ Message }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Message } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
type	按钮类型，可选值为result,multi,week,''(空字符串)	string	''
icon	Icon类型，可选值为success, pay, fail, error, warn, info, wait, question，也可以传入一个class来自定义Icon	string	success
main	纯文本提示主内容,当指定为false时，该占位的DOM不展示	string,boolean	-
sub	纯文本提示副内容,当指定为false时，该占位的DOM不展示	string,boolean	-

slots

name	说明	scope
main	用于满足props.main需要填充DOM的场景	-
sub	用于满足props.sub需要填充DOM的场景	-
-	用于在sub下面填充内容	-

events

name	说明	函数

Demo

成功状态

截图

代码

```
<template>

<div>
<Message type="result"icon="success"main="成功"sub="成功副提示"></Message>
<Message type="multi"icon="success"main="主提示"sub="副提示"></Message>
<Message type=""icon="success"main="支付成功":sub="false"></Message>
</div>

</template>
```

成功结果页

截图

代码

```
<template>

<div>
<Message type="result"icon="success"main="成功">
<template slot="sub">
内容详情，根据实际文案安排<br />如果换行不超过两行
</template>
</Message>
<div class="am-button-wrap">
<AButton type="blue">主操作</AButton>
<AButton type="white">辅助操作</AButton>
</div>
</div>

</template>
```

2.9.2 PageResult 结果页

PageResult 结果页文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。

Kylin

```
<dependency component="{ PageResult, AButton }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { PageResult, AButton } from '@alipay/antui-vue';
```

Service命令式调用

直接使用PageResult.show('系统繁忙');的方式，调用命令，不需要写在模板中。会自动在document.body上插入对应DOM

```
PageResult.show({
type: 'text',
text: '显示的消息'
});
```

Service 文档

static methods

PageResult提供以下静态方法

name	说明	函数	
show	创建一个PageResult实例并显示（多次调用会先销毁上次的），创建的参数如下（也可以直接传字符串当做content）	Function(option : Object	string) : vm

show options

创建实例时，接受参数如下，

属性	说明	类型	默认值
type	PageResult类型，可选值为error,empty, nofound, network, busy	string	busy
content	纯字符串文本	string	系统繁忙，请稍后再试
title	标题文本	string	''
btns	显示的按钮数组	Array<{ text: string, click: Function }>	[{text: '重新加载' ,click: () => window.location.reload()}]
zIndex	设置弹层的zIndex值	number	9000

API 文档

props

属性	说明	类型	默认值
type	图标类型，可选值error,empty, nofound, network, busy	string	-
title	业务自定义文案最多14个字符	string	-

brief	业务自定义文案最多两行，可不要辅助文案	string	-

slots

name	说明	scope
-	默认slot，用来放置一个或多个按钮	-
title	用来满足需要自定义DOM的需求	-
brief	用来满足需要自定义DOM的需求	-

events

name	说明	函数

Demo

网络超时

截图

代码

```
<template>

<PageResult type="network"
title="网络不给力"
brief="世界上最遥远的距离莫过于此"
>
<AButton type="page-result">刷新</AButton>
</PageResult>

</template>
```

系统错误

截图

代码

```
<template>

<PageResult type="error"
```

```
title="系统错误正在排查"
brief="耽误你的时间，我们深表歉意"
>
<AButton type="page-result">刷新</AButton>
</PageResult>

</template>
```

empty

截图

代码

```
<template>

<PageResult type="empty"
title="标题文案"
brief="辅助文案"
>
<AButton type="page-result">操作选项</AButton>
</PageResult>

</template>
```

busy

截图

代码

```
<template>

<PageResult type="busy"
title="系统繁忙"
brief="系统繁忙文案"
>
<AButton type="page-result">刷新</AButton>
</PageResult>

</template>
```

2.10 通知组件

2.10.1 Inform 临时通知

Inform 临时通知文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。

Kylin

```
<dependency component="{ Inform }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Inform } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
operation	通知的可用操作，可选go,button	string	null
buttonText	operation为button时按钮显示的文字	string	'知道了'
href	operation为go时有效，点击后跳转位置	string	null

slots

name	说明	scope
-	通知内容	-
button	当operation为button时，可定制buttonText内容为DOM	-

events

name	说明	函数
buttonClick	点击按钮时触发	Function(): void

Demo

基础样式

截图

代码

```
<template>
<div>
<Inform
:style="{visibility: visible ? 'visible' : 'hidden'}"
operation="button"
buttonText="知道了"
@buttonClick="visible = false"
>
防欺诈盗号，请勿泄露支付密码
</Inform>
<Inform
operation="go"
href="https://alipay.com/"
>
防欺诈盗号，请勿泄露支付密码
</Inform>
<Inform>
防欺诈盗号，请勿泄露支付密码
</Inform>
</div>
</template>

<script>
export default {
data() {
return {
visible: true,
};
},
};
</script>
```

2.10.2 Notice 通知

Notice 通知文档介绍了使用该组件的不同方式以及 API 文档：

- 在 Kylin 工程中使用该组件。
- 在其他工程中使用，通过 ESModule 的方式导入组件。

API 文档 提供了 props、slots、events 的接口信息。

Kylin

```
<dependency component="{ Notice }"src="@alipay/antui-vue"></dependency>
```

ESModule

```
import { Notice } from '@alipay/antui-vue';
```

API 文档

props

属性	说明	类型	默认值
operation	公告的可用操作，可选go,close	string	null
href	operation为go时有效，点击后跳转位置	string	null

slots

name	说明	scope
-	公告内容	-

events

name	说明	函数
close	operation为close时点击关闭按钮触发	Function(): void

Demo

基础样式

截图

代码

```
<template>
<div>
<Notice>
因全国公民身份系统升级，添加银行卡
</Notice>
<Notice operation="go"href="https://www.alipay.com/">
因全国公民身份系统升级，添加银行卡
</Notice>
<Notice operation="close"@close="show = false"v-if="show">
因全国公民身份系统升级，添加银行卡
</Notice>
</div>

</template>
```

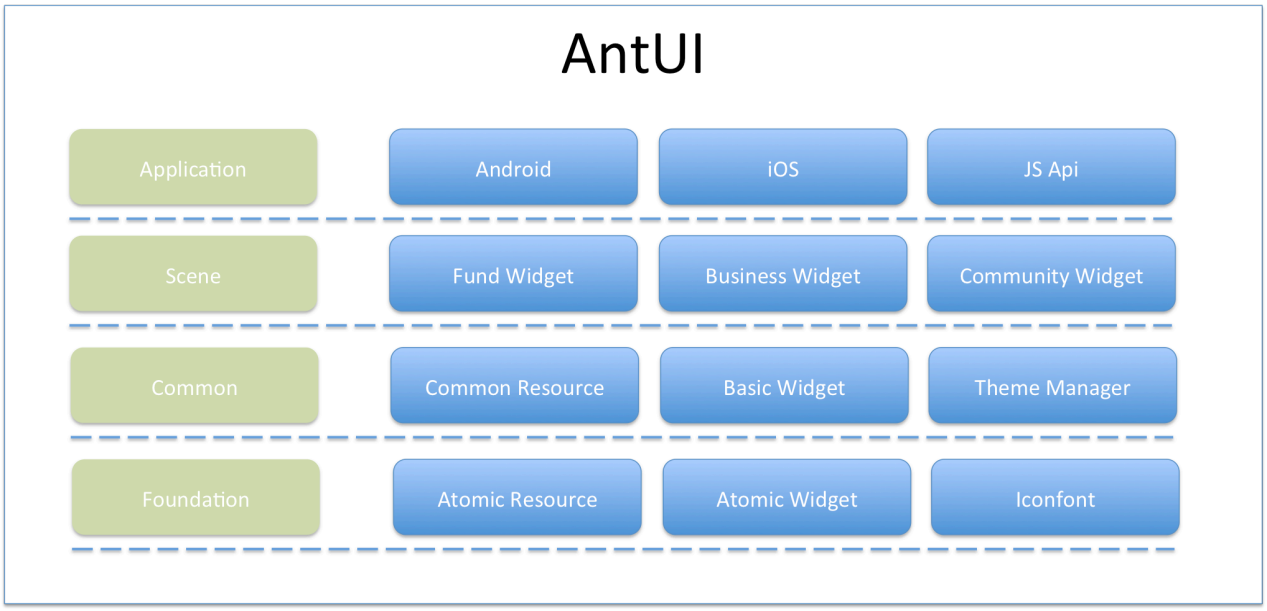
```
<script>
export default {
data() {
return {
show: true,
};
},
};
</script>
```

3 Native 框架简介

mPaaS 统一组件库（AntUI）以标准化的视觉规范为基础，将抽象的视觉规范概念转化为控件实体。作为开发人员，通过使用统一组件库，可以在接入控件时，实现客户端视觉规范的统一。

统一组件库架构

AntUI 的整体架构类似于积木搭建，由下至上构建出 AntUI 统一控件体系：



由下而上的构架层级及其描述如下表所示：

构架层级	描述
基础层（Foundation）	视觉规范单元化的体现，构建 AntUI 体系的基础，主要包含原子资源、原子控件和 Iconfont 图标。 基础层是由视觉规范最小的单元构建。
通用层（Common）	AntUI 的核心统一模块，即业务方最常用的统一控件模块，包含通用资源、基础控件和样式管理器。 通过对基础层的组合和视觉化应用而构建出通用层，通用层可以应用在客户端所有常见的场景。
场景层（Scene）	按照分场景的方式，构建具有场景特点的控件集合，比如资金控件、商家控件、社交控件等。 由于 mPaaS 是一个超级 App，其体量决定了很多业务需要有自己的个性化处理。因此，统一组件库搭建了场景层，按照这些场景在通用层的基础上构建处理业务的个性化控件。
应用层（Application）	应用层提供平台差异化处理和 H5 容器支持等能力，解决了统一和平台个性化之间的矛盾点。 原子、组合和场景成为 AntUI 构建的基础，但在实际应用中，需要同时兼顾到 Android、iOS 和 H5 三个方面的需求。因此，统一组件库构建出一些平台个性化和差异化的接口，即应用层。

基础层

基础层是视觉规范单元化的体现，构建 AntUI 体系的基础，主要包含原子资源、原子控件和 Iconfont 图标。

原子资源 将控件使用的颜色、大小、间距等资源进行原子化定义，保证其唯一性，比如颜色红黄蓝，字号 123 等。

原子控件 是将平台框架自带控件进行包装，构建一套基本的原子控件库。

Iconfont 图标 收集常用场景的图标，并构建 Iconfont 格式，提供一套可用的控件图标库。

通用层

通用层是 AntUI 的核心统一模块，即业务方最常用的统一控件模块，包含通用资源、基础控件和样式管理器。

通用资源 是将原子化资源按照使用场景做二次定义，比如标题颜色、内容颜色、链接色等。

基础控件 对视觉稿定义的控件进行一对一的视觉还原，保持 Android 和 iOS 两个平台的命名和实现一致性，便于客户端开发使用。

样式管理器 对样式进行抽象定义，并统一在管理器内部实现对其的管理，可以实现特定控件在多套皮肤间更换。样式抽象通过增量定义的方式实现，所以只需要关注业务需要的部分元素样式。

场景层

场景层按照分场景的方式，构建具有场景特点的控件集合，比如资金控件、商家控件、社交控件等。

应用层

应用层提供平台差异化处理和 H5 容器支持等能力，解决了统一和平台个性化之间的矛盾点。

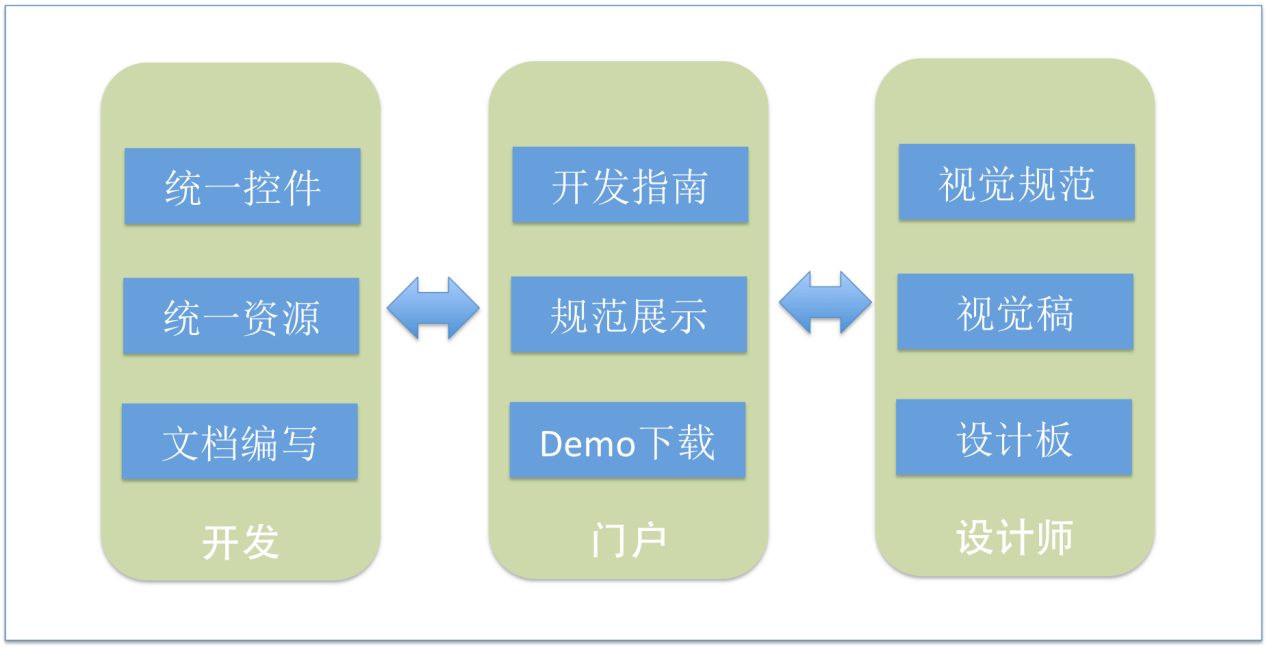
Android 和 iOS 平台在视觉规范上存在差异，以 actionsheet 为例，AntUI 根据平台对其做了不同处理：

- 对 iOS 平台，保持底部浮出。
- 对 Android 平台，则采用中间列表弹窗方式处理。

H5 会有很多差异的场景出现比如弹窗、标题栏等。为了让 H5 部分在视觉体验上保持平台特点，统一组件库对 H5 容器定义了统一的 JSAPI，方便唤起对应的平台控件，实现 H5 页面在Android 和 iOS 平台上的差异化处理。

联动

为减少设计和开发人员之间的沟通成本、避免重复的控件开发工作和视觉设计，统一组件库（AntUI）聚合完成了开发和视觉工作。



设计师制定规范，开发解释规范成为控件，完整的开发指南便于开发实现，形成一站式的控件体系。

通过统一的命名实现开发和设计的统一认知，更多关于命名规范。参见本文的 组件规范和原则 。

通过设计板实现设计人员对已有控件的认知，仅需要拖拽就可以搭建出页面基本结构。

利用门户聚合开发文档和视觉规范，并提供 Demo 下载，可更加直观地查看控件视觉效果。

组件规范和原则

命名风格

Android 和 iOS 两个平台的同类控件命名需完全一样，控件命名以 AU 为前缀，控件自定义属性全部采用驼峰命名。

注意：某些组件可能存在平台差异，一个平台需要实现，而另外一个平台不需要实现。

基础控件与视觉/交互规范匹配

规范中没有的控件不能放入标准控件中。

规范中没有但已经在多处使用的控件应放入候选控件集合中。

不强制某一规范必须实现为单个控件，例如标题栏规范。

易用性

与 commonui 不同的是，不对系统控件再做简单封装（如 APImageView，APTextView），需要用系统控件时，推荐使用原生控件。

命名一定要准确，无二意性。

类似功能在不同控件中应保持一致。

尊重用户习惯。

扩展性

控件功能中不要使用硬编码，比如切换标签个数支持动态更改。

部分控件要提供外部修改布局功能， 如一些对话框，导航条等。

新颖性

可尝试使用最新的平台功能，如 Android 的 recycleview。

4 基于 Native 框架 - Android 组件库

4.1 快速开始

AntUI 支持 **原生 AAR 接入**、**mPaaS Inside 接入** 和 **组件化接入** 三种接入方式。

前置条件

- 若采用原生 AAR 方式接入，需先完成 将 mPaaS 添加到您的项目中 的前提条件和后续相关步骤。
- 若采用 mPaaS Inside 方式接入，需先完成 mPaaS Inside 接入流程 。
- 若采用组件化方式接入，需先完成 组件化接入流程 。

添加 SDK

原生 AAR 方式

参考 AAR 组件管理 ，通过 **组件管理（AAR）** 在工程中安装 **AntUI** 组件。

mPaaS Inside 方式

在工程中通过 **组件管理** 安装 **AntUI** 组件。
更多信息，参考 管理组件依赖 。

组件化方式

在 Portal 和 Bundle 工程中通过 **组件管理** 安装 **AntUI** 组件。

更多信息，参考 管理组件依赖 。

代码示例

参考 获取代码示例 获取 Android Demo，并查看 AntUI 工程代码。

4.2 弹窗组件

4.2.1 卡片菜单

AUCardMenu 组件用于在用户点击 mPaaS 客户端首页上的卡片时弹出选择菜单，实质为弹窗（Dialog），类似 popupwindow。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* show dialog with default width
* @param view
* @param popItems
*/
public void showDrop(View view, ArrayList<MessagePopItem> popItems)

/**
* show dialog with given width
* @param view
* @param popItems
* @param width
*/
public void showDrop(View view, ArrayList<MessagePopItem> popItems, int width) {
int defaultMarginRight = mContext.getResources().getDimensionPixelSize(R.dimen.AU_SPACE5)/2;
showDrop(view, popItems, width, defaultMarginRight);
}

/**
* show dialog with given width & marginRight
* @param view
* @param popItems
* @param width
*/
public void showDrop(View view, ArrayList<MessagePopItem> popItems, int width, int marginRight)

/**
* show dialog with given ViewLoc
* @param location
* @param popItems
*/
public void showDropWithLocation(ViewLoc location, ArrayList<MessagePopItem> popItems)

如果有网络链接的图片，需要自己下载图片
public void setOnLoadImageListener(OnLoadImageListener onLoadImageListener)

public interface OnLoadImageListener {

/**
* show dialog with given width & marginRight
* @param url 图片的 URL
* @param imageView 图片的目标 View
* @param defaultDrawable 默认图片
*/
public void loadImage(String url, AUImageView imageView ,Drawable defaultDrawable);
}

```

自定义属性

无，不支持 XML 布局。

代码示例

```
ArrayList<MessagePopItem> menuList = new ArrayList<MessagePopItem>();

MessagePopItem item1 = new MessagePopItem();
IconInfo info = new IconInfo();
info.type = IconInfo.TYPE_DRAWABLE;
info.drawable = getResources().getDrawable(R.drawable.menu_del_reject);
item1.icon = info;
item1.title ="tes1的文案就是这样";
menuList.add(item1);


MessagePopItem item2 = new MessagePopItem();
IconInfo info2 = new IconInfo();
info2.type = IconInfo.TYPE_DRAWABLE;
info2.drawable = getResources().getDrawable(R.drawable.menu_delete);
item2.icon = info2;
item2.title ="阿斯顿发斯蒂芬";
menuList.add(item2);

MessagePopItem item3 = new MessagePopItem();
IconInfo info3 = new IconInfo();
info3.type = IconInfo.TYPE_DRAWABLE;
info3.drawable = getResources().getDrawable(R.drawable.menu_ignore);
item3.icon = info3;
item3.title ="高速费毒黄瓜东方红";
menuList.add(item3);

MessagePopItem item4 = new MessagePopItem();
IconInfo info4 = new IconInfo();
info4.type = IconInfo.TYPE_DRAWABLE;
info4.drawable = getResources().getDrawable(R.drawable.menu_reject);
item4.icon = info4;
item4.title ="加工费胡椒粉工行卡规范";
menuList.add(item4);

MessagePopItem item5 = new MessagePopItem();
IconInfo info5 = new IconInfo();
info5.type = IconInfo.TYPE_DRAWABLE;
info5.drawable = getResources().getDrawable(R.drawable.menu_report);
item5.icon = info5;
item5.title ="考虑过黄金矿工法国红酒发";
menuList.add(item5);

final AUCardMenu popMenu = new AUCardMenu(CardMenuActivity.this);
int id = v.getId();
if(id == R.id.showCardMenu1) {
popMenu.showDrop(textView1,menuList);
}else if(id == R.id.showCardMenu2) {
popMenu.showDrop(textView2,menuList);
}
```

4.2.2 级联选择器

AUCascadePicker 提供一个多级级联选择器，最多支持三级联动选择。

效果图

接口说明

```
/**
* 设置选中的列表
*/
public void setDateData(List<PickerDataModel> strList)


/**
* 设置选择启动选中项
* @param model
*/
public void setSelectedItem(PickerDataModel model)

/**
* 设置选择项监听
* @param model
*/
public void setOnLinkagePickerListener(OnLinkagePickerListener listener)
```

JSAPI 说明

接口

antUIGetCascadePicker

接口使用

```
AlipayJSBridge.call('antUIGetCascadePicker',
{
title: 'nihao',//级联选择标题
selectedList:[{"name":"杭州市",subList:[{"name":"上城区"}]}],
list: [
{
name:"杭州市",//条目名称
subList: [
{
name:"西湖区",
subList: [
{
name:"古翠街道"
```

```
},
{
name:"文新街道"
}
]
},
{
name:"上城区",
subList: [
{
name:"延安街道"
},
{
name:"龙翔桥街道"
}
]
}
]//级联子数据列表
}
]//级联数据列表
},
function(result){
console.log(result);
});
```

入参

名称	类型	描述	是否必选	默认值	版本
title	string	级联控件标题	NO	—	10.1.2
selectedList	json	选中态，指定选中的子项，格式与入参一致（[{“name”:”杭州市”,subList:[{“name”:”上城区”}]}]）	NO	—	10.1.2
list	json	选择器数据列表	YES	—	10.1.2
name (list 内的 name)	string	条目名称	YES	—	10.1.2
subList (list 内的 subList)	json	子条目列表	NO	—	10.1.2
fn	function	选择完成后的回调函数	NO	—	10.1.2

出参

名称	类型	描述	版本

success	bool	是否选择完成，取消返回 false	10.1.2
result	json	选择的结果，如 [{"name":"杭州市",subList:[{"name":"上城区"}]}]	10.1.2

代码示例

```
AUCascadePicker datePicker = new AUCascadePicker(PickerActivity.this);
datePicker.setDateData(datas);
datePicker.setOnLinkagePickerListener(new AUCascadePicker.OnLinkagePickerListener() {
@Override
public void onLinkagePicked(PickerDataModel msg) {
PickerDataModel model = msg;
AuiLogger.info("onLinkagePicked","onLinkagePicked:"+msg.name+ model);
StringBuilder sb = new StringBuilder();
while (msg != null){
sb.append(msg.name+"");
if(msg.subList != null && msg.subList.size() > 0) {
msg = msg.subList.get(0);
}else {
msg = null;
}
}
box3.getInputEdit().setText(sb);
}
});
datePicker.show();
```

4.2.3 日期

AUDatePicker 是一个日期选择控件，本质是一个弹窗。

效果图

依赖

参见 管理组件依赖 。

接口说明

```
/**
* Instantiates a new Date picker.
*
* @param activity the activity
* @param mode the mode
* @see #YEAR_MONTH_DAY #YEAR_MONTH_DAY#YEAR_MONTH_DAY
* @see #YEAR_MONTH #YEAR_MONTH#YEAR_MONTH
* @see #MONTH_DAY #MONTH_DAY#MONTH_DAY
*/
public AUDatePicker(Activity activity, @Mode int mode)

/**
* 设置日期范围
*
* @param startYear the start year
* @param endYear the end year
*/
public void setRange(int startYear, int endYear)

../**
* 选中指定的年月日
*
* @param year the year
* @param month the month
* @param day the day
*/
public void setSelectedItem(int year, int month, int day)

/**
* 选中指定的日期
*
* @param yearOrMonth the year or month
* @param monthOrDay the month or day
*/
public void setSelectedItem(int yearOrMonth, int monthOrDay)

/**
* 设置日期选中的监听
*
* @param listener the listener
*/
public void setOnDatePickListener(OnDatePickListener listener) {
this.onDatePickListener = listener;
}

/**
* The interface on year month day pick listener.
*/
public interface OnYearMonthDayPickListener extends OnDatePickListener {

/**
```

```
* On date picked.
*
* @param year the year
* @param month the month
* @param day the day
*/
void onDatePicked(String year, String month, String day);

}

/**
* The interface On year month pick listener.
*/
public interface OnYearMonthPickListener extends OnDatePickListener {

/**
* On date picked.
*
* @param year the year
* @param month the month
*/
void onDatePicked(String year, String month);

}

/**
* The interface On month day pick listener.
*/
public interface OnMonthDayPickListener extends OnDatePickListener {

/**
* On date picked.
*
* @param month the month
* @param day the day
*/
void onDatePicked(String month, String day);

}
```

自定义属性

无，不支持 XML 布局文件。

代码示例

```
AUDatePicker datePicker = new AUDatePicker(DatePickActivity.this,AUDatePicker.YEAR_MONTH_DAY);
datePicker.setRange(1949,2050);
datePicker.setOnDatePickListener(new AUDatePicker.OnYearMonthDayPickListener() {
@Override
public void onDatePicked(String year, String month, String day) {
Toast.makeText(DatePickActivity.this,year +"-"+ month +"-"+ day,Toast.LENGTH_LONG).show();
}
});
```

```
datePicker.show();
```

内嵌页面的 AUDatePicker 的使用：

```
AUDatePicker picker = new AUDatePicker(this);
picker.show();
picker.dismiss();
View view = picker.getOutterView();
LinearLayout layout = (LinearLayout) findViewById(R.id.layout);
layout.removeAllViews();
if(view != null) {
((ViewGroup) view.getParent()).removeAllViews();
layout.addView(view);
}
```

如有疑问，可 提交工单，联系技术支持人员获取帮助。

4.2.4 菜单

AUFloatMenu 组件提供一个包含图标、选项列表的菜单。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 构造方法
*
* @param context 包含 antui-build 依赖的 activity 上下文
*/
public AUFloatMenu(Context context)
/**
* 默认靠右显示
* @param view 基于显示的 view
* @param popItems 列表显示模型
*/
@Override
public void showDrop(View view, ArrayList<MessagePopItem> popItems);

/**
* 靠左显示
* @param view 基于显示的 view
* @param popItems 列表显示模型
*/
```

```
public void showAsDropDownLeft(View view, ArrayList<MessagePopItem> popItems);

/**
* 屏幕居中显示
* @param parent 基于显示的 view
* @param title 显示列表的标题
* @param popItems 列表显示模型
*/
public void showAsDropDownTitleCenter(View parent, String title, ArrayList<MessagePopItem> popItems);

/**
* 添加显示列表条目点击事件
* @param listener
*/
public void setOnClickListener(AdapterView.OnItemClickListener listener)
```

代码示例

```


ArrayList<MessagePopItem> menuList = new ArrayList<MessagePopItem>();

MessagePopItem item1 = new MessagePopItem();
IconInfo info = new IconInfo();
info.icon = getResources().getString(R.string.iconfont_add_user);
item1.icon = info;
item1.title ="添加朋友";
menuList.add(item1);


MessagePopItem item2 = new MessagePopItem();
IconInfo info2 = new IconInfo();
info2.icon = getResources().getString(R.string.iconfont_group_chat);
item2.icon = info2;
item2.title ="群聊";
menuList.add(item2);

MessagePopItem item3 = new MessagePopItem();
IconInfo info3 = new IconInfo();
info3.icon = getResources().getString(R.string.iconfont_scan);
item3.icon = info3;
item3.title ="扫一扫";
menuList.add(item3);

MessagePopItem item4 = new MessagePopItem();
IconInfo info4 = new IconInfo();
info4.icon = getResources().getString(R.string.iconfont_collect_money);
item4.icon = info4;
item4.title ="收付款";
menuList.add(item4);

MessagePopItem item5 = new MessagePopItem();
IconInfo info5 = new IconInfo();
info5.icon = getResources().getString(R.string.iconfont_help);
item5.icon = info5;
item5.title ="使用帮助";
```

```
menuList.add(item5);

final AUFloatMenu floatMenu = new AUFloatMenu(ScrollTitleBarActivity.this);
floatMenu.showDrop(v, menuList);
floatMenu.setOnClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
Toast.makeText(ScrollTitleBarActivity.this, String.valueOf(position), Toast.LENGTH_SHORT).show();
floatMenu.hideDrop();
}
});
```

4.2.5 图片弹窗

AUImageDialog（原 SalesPromotionLimitDialog）提供一个带顶部标题、三级文案、底部确认按钮或者底部左右按钮，且中间包含一个 ImageView 的对话框。该组件可用于限流的消息提示。

效果图

依赖

参见 快速开始 。

接口说明

```
public interface OnItemClickListener {
void onItemClick(int index);
}

/**
* 获取 AUImageDialog 实例
*
* @param context context 对象
* @return 返回一个 AUImageDialog 实例
*/
public static AUImageDialog getInstance(Context context)

/**
* 关闭监听
*
* @param mCloseBtnClickListener
*/
public void setCloseBtnClickListener(View.OnClickListener mCloseBtnClickListener)


/**
* 设置一级标题文案
*/
public void setTitle(CharSequence title)
```

```
/**
* 设置一级标题文案字体大小，单位为 sp
*
* @param size
*/
public void setTitleTextSize(float size)

/**
* 设置一级标题可见性
*
* @param visibility
*/
public void setTitleTextVisibility(int visibility)
}

/**
* 设置二级标题可见性
*
* @param visibility
*/
public void setSubTitleTextVisibility(int visibility)

/**
* 设置一级标题颜色
*
* @param color
*/
public void setTitleTextColor(int color)

/**
* 设置二级标题文案
*
* @param title
*/
public void setSubTitle(CharSequence title)

/**
* 设置二级标题字体大小，单位为 sp
*
* @param size
*/
public void setSubTitleTextSize(float size)

/**
* 设置二级标题文案颜色
*
* @param color
*/
public void setSubTitleTextColor(int color)

/**
* 设置三级标题文案
*
* @param text
*/
public void setThirdTitleText(String text)
```

```
/**
* 设置三级标题颜色
*
* @param color
*/
public void setThirdTitleTextColor(int color)

/**
* 设置 ImageView 的背景
*
* @param drawable
*/
public void setLogoBackground(Drawable drawable)

/**
* 设置 ImageView 的背景
*
* @param resid
*/
public void setLogoBackgroundResource(int resid)

/**
* 设置 ImageView 的背景颜色
*
* @param color
*/
public void setLogoBackgroundColor(int color)

/**
* 设置对话框的背景透明度
*
* @param alpha
*/
public void setBackgroundTransparency(float alpha)

/**
* 返回是否使用动画
*/
public boolean isUsdAnim()

/**
* 设置对话框显示、消失时是否使用动画，默认为 true
*
* @param usdAnim
*/
public void setUsdAnim(boolean usdAnim)

/**
* 设置关闭按钮是否可见
*
* @param visibility
*/
public void setCloseButtonVisibility(int visibility)

/**
```

```
* 设置确认按钮文案
*
* @param text
*/
public void setConfirmBtnText(String text)


/**
* 返回确认按钮
*/
public Button getConfirmBtn()

/**
* 设置确认按钮点击监听
*
* @param clickListener
*/
public void setOnConfirmBtnClickListener(View.OnClickListener clickListener)


/**
* 不带动画的显示对话框
*/
public void showWithoutAnim()

/**
* 设置倒计时
* @param seconds 倒计时秒
* @param tickColor
* @param action
* @param clickListener
* @param timerListener
*/
public void showWithTimer(int seconds, String tickColor, String action, View.OnClickListener clickListener,
TimerListener timerListener)

public void showWithTimer(int seconds, View.OnClickListener clickListener, TimerListener timerListener)

/**
* 获取默认的倒计时颜色
* @return
*/
public String getDefaultTimeColorStr()

/**
* 不带动画的 dismiss dialog
*/
public void dismissWithoutAnim()

@Override
public void dismiss()


public boolean isCanceledOnTouch() {
return canceledOnTouch;
}
```

```
/**
* 设置是否点击中间图片时对话框自动取消
*
* @param canceledOnTouch
*/
public void setCanceledOnTouch(boolean canceledOnTouch)


/**
* 设置列表按钮
* @param buttonListInfo
* @param listener
*/
public void setButtonListInfo(List<String> buttonListInfo, OnItemClickListener listener)

public ImageView getLogoImageView() {
return bgImageView;
}

public TextView getTitleTextView() {
return titleTextView_1;
}

public TextView getSubTitleTextView() {
return titleTextView_2;
}

public TextView getThirdTitleTextView() {
return titleTextView_3;
}

public ImageView getBottomLine() {
return bottomLine;
}
```

代码示例

```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.showWithTimer(5, null, null);
```

```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setCanceledOnTouch(true);
dialog.setTitle("标题单行");
```

```
dialog.setSubTitle("说明当前状态、提示用户解决方案，最好不要超过两行。");
dialog.setConfirmBtnText("行动按钮");
dialog.showWithoutAnim();
```

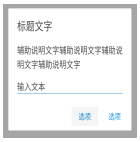
```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setCanceledOnTouch(true);
dialog.setTitle("一级文案");
dialog.setSubTitle("二级文案");
dialog.setThirdTitleText("同意xxx协议");
dialog.setConfirmBtnText("行动按钮");
dialog.showWithoutAnim();
```

```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setTitle("标题单行");
dialog.setSubTitle("描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。");
dialog.setButtonListInfo(getData(), new AUImageDialog.OnItemClickListener() {
@Override
public void onItemClick(int index) {

}
});
dialog.showWithoutAnim();
```

4.2.6 输入弹窗

AUInputDialog（原 APInputDialog）提供一个带标题、正文、确认和取消按钮以及一个输入框的对话框。

效果图



依赖

参见 快速开始 。

接口说明

```
/**
```

```
* 根据传入参数构造一个 AUInputDialog
*
* @param context context 对象
* @param title 标题
* @param msg 消息
* @param positiveString 确认按钮文案
* @param negativeString 取消按钮文案
* @param isAutoCancel 设置点击弹窗以外区域是否自动取消
*/
public AUInputDialog(Context context, String title, String msg, String positiveString,
String negativeString, boolean isAutoCancel)

/**
* 获取取消按钮
*/
public Button getCancelBtn();

/**
* 获取确认按钮
*/
public Button getEnsureBtn();

/**
* 获取标题 TextView
*/
public TextView getTitle();

/**
* 获取消息 TextView
*/
public TextView getMsg();

/**
* 获取底部按钮的 LinearLayout
*/
public LinearLayout getBottomLayout();

/**
* 获取弹窗最外层的 RelativeLayout
*/
public RelativeLayout getDialogBg();

/**
* 设置确认按钮监听
*/
public void setPositiveListener(OnClickPositiveListener listener);

/**
* 设置取消按钮监听
*/
public void setNegativeListener(OnClickNegativeListener listener);

/**
* 获取输入框 EditText
*/
public AUEditText getInputContent() {
```

```
return inputContent;
}

/**
* Starts and display the dialog.
*/
public void show();
```

代码示例

```
AUInputDialog dialog = new AUInputDialog(this,"标题文字","辅助说明文字",
"确认","取消", true);
dialog.show();
```

4.2.7 列表弹窗

AUListDialog（原 APListPopDialog）提供一个带标题、选项列表、确认、取消按钮的列表型对话框。每一个选项用 PopMenuItem 表示，包含图标、选项名称、选中状态等信息。

效果图

依赖

参见 快速开始 。

接口说明

```
public interface OnItemClickListener {
```

```
void onItemClick(int index);
}

/**
* 根据传入的列表数据创建 AUListDialog，item 只包含文字，无图片
*
* @param context context 对象
* @param list String 列表，纯 ItemName 属性，无图片
*/
public AUListDialog(Context context, ArrayList<String> list)

/**
* 根据传入的列表数据创建 AUListDialog
*
* @param list PopMenuItem 列表
* @param context context 对象
*/
public AUListDialog(ArrayList<PopMenuItem> list, Context context)

/**
* 根据传入的列表数据创建 AUListDialog
*
* @param title 标题
* @param list PopMenuItem 对象列表，可设置图标
* @param context context 对象
*/
public AUListDialog(String title, ArrayList<PopMenuItem> list, Context context)

/**
* 根据传入的列表数据创建 AUListDialog
*
* @param title 标题
* @param message 正文
* @param list PopMenuItem 对象列表，可设置图标
* @param context context 对象
*/
public AUListDialog(String title, String message, ArrayList<PopMenuItem> list, Context context)

/**
* 根据传入的列表数据创建 AUListDialog
*
* @param title 标题
* @param list PopMenuItem 列表
* @param showSelectionState 是否显示选项选中状态图标
* @param positiveString 确认按钮文案
* @param positiveListener 确认按钮监听器
* @param negativeString 取消按钮文案
* @param negativeListener 取消按钮监听器
* @param context context 对象
*/
public AUListDialog(String title, ArrayList<PopMenuItem> list, boolean showSelectionState,
String positiveString, View.OnClickListener positiveListener,
String negativeString, View.OnClickListener negativeListener, Context context)

/**
* 根据传入的列表数据创建AUListDialog
```

```
*
* @param title 标题
* @param message 正文
* @param list PopMenuItem 列表
* @param showSelectionState 是否显示选项选中状态图标
* @param positiveString 确认按钮文案
* @param positiveListener 确认按钮监听器
* @param negativeString 取消按钮文案
* @param negativeListener 取消按钮监听器
* @param context context 对象
*/
public AUListDialog(String title, String message, ArrayList<PopMenuItem> list, boolean showSelectionState,
String positiveString, View.OnClickListener positiveListener,
String negativeString, View.OnClickListener negativeListener, Context context)

/**
* 设置列表选项点击事件监听
*/
public void setOnItemClickListener(OnItemClickListener listener) {
this.listener = listener;
}


/**
* 动态数据刷新接口
*
* @param list
*/
public void updateData(ArrayList<PopMenuItem> list)
```

代码示例

纯列表弹窗


```
new AUListDialog(this, getData(7)).show();

private ArrayList<String> getData(int size){
ArrayList<String> data = new ArrayList<String>();
for (int i= 1 ; i<= size; i++){
data.add("选项文本"+ String.valueOf(i));
}
return data;
}
```

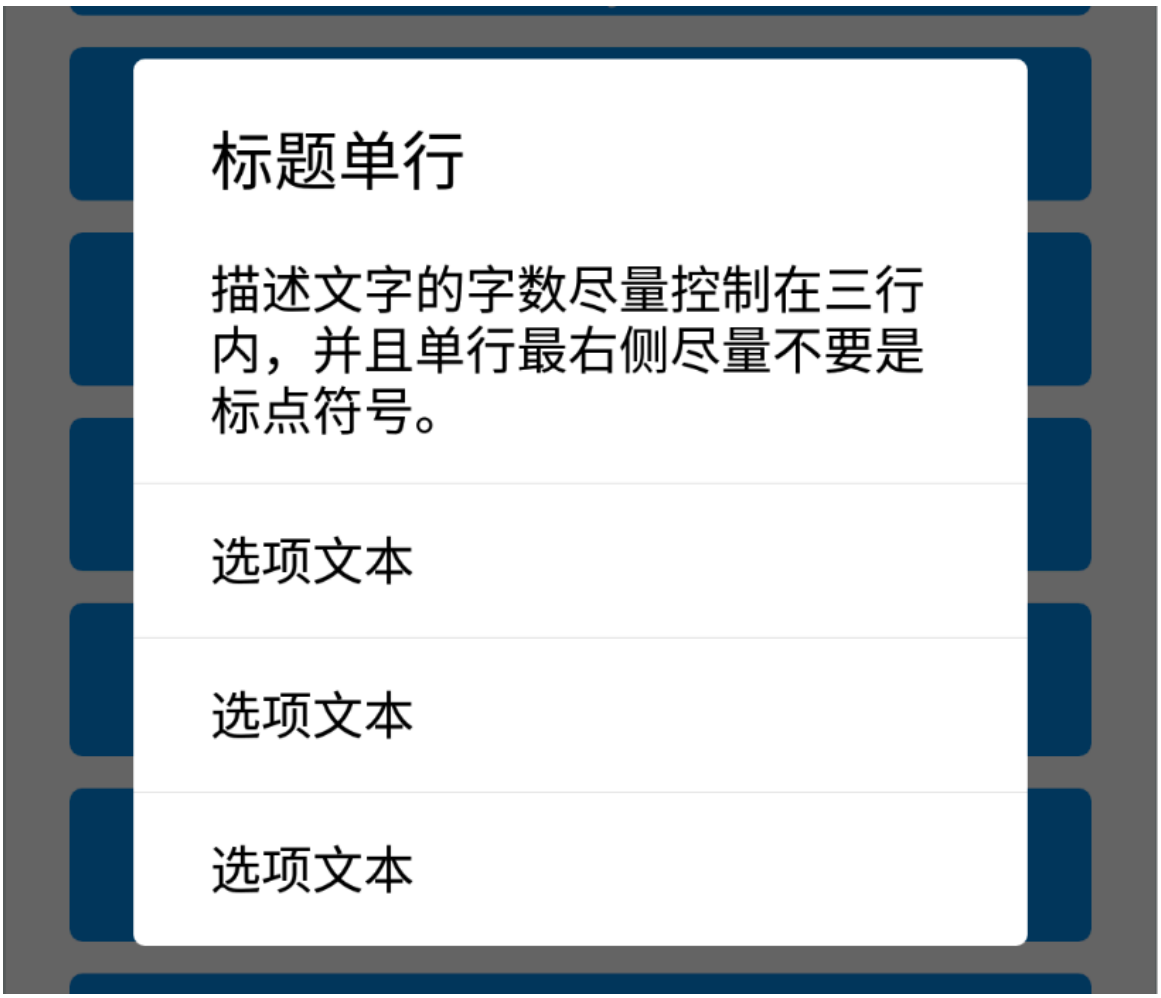
带标题的列表弹窗


```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
new AUListDialog("标题", items, this).show();
```

带说明文本的列表弹窗


```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
new AUListDialog("","描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。", items,
this).show();
```

带标题和说明文案的列表弹窗




```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
new AUListDialog("标题单行","描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。", items, this).show();
```

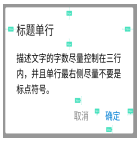
带勾选项的列表弹窗


```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
PopMenuItem item = new PopMenuItem("选项文本", null);
item.setType(AUCheckIcon.STATE_UNCHECKED);
items.add(item);
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
items.add(new PopMenuItem("选项文本", null));
new AUListDialog("标题文字", items, true,"确定", null,"取消", null, this).show();
```

4.2.8 消息弹窗

AUNoticeDialog （原 APNoticePopDialog ）提供一个带标题、正文、确认和取消按钮的对话框，支持常用的业务消息显示。

效果图




```
AUNoticeDialog dialog = new AUNoticeDialog(this,"标题单行",
"描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。",
"确定","取消", true);
dialog.show();
```

基本规则

- 弹窗有最小高度。
- 仅有标题或描述文字的时候，布局以最小高度进行纵向居中显示

- **确认**、**取消** 的按钮文字长度最好不要超出 4 个字，因为小屏手机（如 VIVO Y23L ）会显示不下。

依赖

参见 快速开始 。

接口

```
public AUNoticeDialog(Context context, CharSequence title, CharSequence msg,
String positiveString, String negativeString);

public AUNoticeDialog(Context context, CharSequence title, CharSequence msg,
String positiveString, String negativeString, boolean isAutoCancel) ;

/**
* 根据传入的参数创建一个 AUNoticeDialog
*
* @param context context 对象
* @param title 标题
* @param msg 消息
* @param positiveString 确认按钮文案
* @param negativeString 取消按钮文案
* @param isAutoCancel 设置点击弹窗以外区域是否自动取消
*/
public AUNoticeDialog(Context context, CharSequence title, CharSequence msg, String positiveString, String
negativeString, boolean isAutoCancel);
```

```
/**
* 设置确认按钮文案的颜色
*
* @param c 色值
*/
public void setPositiveTextColor(ColorStateList c);

/**
* 设置取消按钮文案的颜色
*
* @param c 色值
*/
public void setNegativeTextColor(ColorStateList c);

/**
* 获取取消按钮
*/
public Button getCancelBtn();

/**
* 获取确认按钮
*/
public Button getEnsureBtn();

/**
* 获取标题 TextView
*/
public TextView getTitle();

/**
* 获取消息 TextView
*/
public TextView getMsg();

/**
* 设置确认按钮点击监听
*
* @param listener
*/
public void setPositiveListener(OnClickPositiveListener listener);

/**
* 设置取消按钮点击监听
*
* @param listener
*/
public void setNegativeListener(OnClickNegativeListener listener);

/**
* 获取弹窗布局最外层的 RelativeLayout
*/
public RelativeLayout getDialogBg();

/**
* Start the dialog and display it on screen.
```

```
*/
public void show();
```

代码示例

```
// 不带标题的
AUNoticeDialog dialog = new AUNoticeDialog(this,"",
"描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。",
"确认","取消", true);
dialog.show();

// 不带描述信息的
AUNoticeDialog dialog = new AUNoticeDialog(this,"标题单行",
"",
"确认", null, true);
dialog.show();
```

4.2.9 社交_收银台结果页弹窗

AUOperationResultDialog 提供一个带标题文字、选项列表的弹窗，主要应用于社交和收银台的结果展示。

效果图

依赖

参见 管理组件依赖 。

接口说明

```
/**
* 根据传入的列表数据创建 AUListDialog
*
* @param title 标题
* @param list PopMenuItem 对象列表，可设置图标
* @param context context 对象
*/
public AUOperationResultDialog(Context context, String title, List<String> list)

/**
* 设置列表选项点击事件监听
*/
public void setOnItemClickListener(OnItemClickListener listener)

/**
* 动态数据刷新接口
*
```

```
* @param list
*/
public void updateData(ArrayList<PopMenuItem> list)

/**
* 获取 imageView
* @return
*/
public ImageView getIconView()

/**
* 设置分割线是否可见
* @param visibility
*/
public void setDivierViewVisibility(int visibility)
```

代码示例


```
public void clickAUOperationResultDialog(View view) {
AUOperationResultDialog dialog = new AUOperationResultDialog(this,"标题文字",getData());
dialog.getIconView().setImageDrawable(getResources().getDrawable(R.drawable.image));
dialog.show();
}
```

4.2.10 弹出菜单

AUPopMenu 组件提供导航栏选项卡点击弹出菜单，实质为 popupwindow。

AUPopMenu 与 AUFloatMenu 的区别：无底面蒙层，有外围边框，所有布局采用居中的形式。

基本功能

- 业务控制向上或向下弹出。
- 业务传入 string list 使用默认样式，或者直接传入 adapter。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 数据构造，使用默认样式
* @param context
* @param itemArrayList
*/
public AUPopMenu(Context context, ArrayList<MessagePopItem> itemArrayList)

/**
* adapter 构造，使用自定义样式
* @param context
* @param listAdapter
*/
public AUPopMenu(Context context, BaseAdapter listAdapter)

/**
* tip toast down
* @param anchorView
*/
public void showTipView(View anchorView)

/**
* tip toast with direction
* @param anchorView
* @param isDown
*/
public void showTipView(View anchorView, boolean isDown)

/**
* 窗口消失
*/
public void dismiss()


/**
* 设置选项点击监听
* @param listener
*/
public void setOnItemClickListener(AdapterView.OnItemClickListener listener)
```

自定义属性

无，不支持 XML 布局。

代码示例

```
final AUPopMenu popMenu = new AUPopMenu(ScrollTitleBarActivity.this, getItemList());
popMenu.showTipView(view);
popMenu.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

}
});
```

4.2.11 录音

AURecordFloatTip 组件用于显示 **正在录音** 状态的浮层，旨在给予用户更直接的录音体验。

效果图

依赖

参见 管理组件依赖 。

构造说明

```
public AURecordFloatTip(Activity activity) ;

public AURecordFloatTip(Activity activity, String tip);
```

接口说明

```
/**
* 显示浮层
*/
public void show();

/**
* 浮层消失
*/
public void dismiss();

/**
* 获取浮层文本 view
*
* @return
*/
public AUTextView getTipTextView();

/**
* 获取浮层图标 view
*
* @return
*/
public AUImageView getIconView();
```

代码示例

```
if (!isSHowAURecordFloatTip) {
((AUButton) view).setText("关闭AURecordFloatTip");
if (mAURecordFloatTip == null) {
mAURecordFloatTip = new AURecordFloatTip(this,"正在录音");
```

```
}
mAURecordFloatTip.show();
isSHowAURecordFloatTip = true;
} else {
((AUButton) view).setText("显示AURecordFloatTip");
if (mAURecordFloatTip != null) {
mAURecordFloatTip.dismiss();
}
isSHowAURecordFloatTip = false;
}
```

4.2.12 提示

使用 BaseFragmentActivity 和 BaseActivity 的 toast 方法时，mPaaS 框架会对弹窗提示统一进行修改。

对基于 BaseFragmentActivity 和 BaseActivity 开发的 activity，系统默认使用 AUToast。

效果图

AUToast

AUProgressDialog

依赖

参见 快速开始 。

接口说明

```
/**
* 实例化 Toast
*
* @param context 上下文，请使用当前页面的 actvity
* @param drawableId 图片资源
* @param tipSrcId 文字提示 ID
* @param duration 显示时间 Toast.Long/Toast.Short
* @return Toast
*/
public static Toast makeToast(Context context, int drawableId, int tipSrcId, int duration) {
CharSequence tipSrc = context.getResources().getText(tipSrcId);
return makeToast(context, drawableId, tipSrc, duration);
```

```
}

/**
* 创建 Toast
*
* @param context 上下文，请使用当前页面的 actvity
* @param tipSrcId 提示信息
* @param duration 时间
* @return toast
*/
public static Toast makeToast(Context context, int tipSrcId, int duration) {
CharSequence tipSrc = context.getResources().getText(tipSrcId);
return makeToast(context, 0, tipSrc, duration);
}

/**
* Make a toast that just contains a image view and a text view.
*
* @param context 上下文，请使用当前页面的 actvity
* @param drawableId image resourceid
* @param tipSrc The text to show. Can be formatted text.
* @param duration How long to display the message. Either or
* @return
*/
public static Toast makeToast(Context context, int drawableId, CharSequence tipSrc, int duration)
```

代码示例

```
//成功
AUToast.makeToast(ToastActivity.this, com.alipay.mobile.antui.R.drawable.toast_ok,"成功提示",
Toast.LENGTH_SHORT).show();


//失败
AUToast.makeToast(ToastActivity.this, com.alipay.mobile.antui.R.drawable.toast_false,"失败提示",
Toast.LENGTH_SHORT).show();
}

//警示
AUToast.makeToast(ToastActivity.this, com.alipay.mobile.antui.R.drawable.toast_warn,"警示提示",
Toast.LENGTH_SHORT).show();
}

//文本
AUToast.showToastWithSuper(ToastActivity.this, 0,"最长文案不超过14个字", Toast.LENGTH_SHORT);

//加载
AUProgressDialog dialog = new AUProgressDialog(this);
dialog.setMessage("加载中");
dialog.show();

}
```

4.3 输入组件

4.3.1 资金输入

AUAmountInputBox 组件提供金额输入框，输入框中的数字为特殊的数字字体。输入框包括编辑框（AUAmountEditText）和备注（AUAmountFootView）两个部分，其中 AUAmountFootView 有两种样式（可编辑的输入框和文本展示），可自由组合。

同时，该组件配套提供带特殊数字字体的展示用 AUAmountLabelText。

效果图

注意：金额输入规则如下图所示：

依赖

参见 管理组件依赖 。

接口说明

AUAmountInputBox

```
/**
* 获取编辑框
* @return
*/
public AUEditText getEditText()

/**
* 获取编辑框
* @return
*/
public AUAmountEditText getEditLayout()

/**
* 获取资金链的 footView
* @return
*/
public AUAmountFootView getFootView()

/**
* 获取输出框的标题栏
* @return
*/
public AUTextView getTitleView()
```

```
/**
* 设置 HeadView 的属性
* @param style EDIT_STYLE 、TEXT_STYLE
*/
public void setFootStyle(int style)

/**
* 设置 FootView 的编辑框提示
* @param hint
*/
public void setFootHint(String hint)


/**
* 设置FootView的text
* @param text
*/
public void setFootText(String text)
```

AUAmountEditText

```
/**
* 获取 EditText
* @return
*/
public AUEditText getEditText()


/**
* 获取输入框信息
* @return
*/
public Editable getEditTextEditable()

/**
* 设置分割线显示或隐藏
* @param visible
*/
public void setDividerVisible(boolean visible)

/**
* 设置提示
* @param hint
*/
public void setHint(String hint)

/**
* 设置是否展示删除按钮
* @param isShow
*/
public void isShowClearIcon(boolean isShow)

/**
* 增加 focus 监听
* @param listener
```

```
*/
public void addOnFocusChangeListeners(OnFocusChangeListener listener)

/**
* 绑定外部的 AUNumberKeyboardView ScrollView
* @param keyboardView
* @param scrollView
*/
public void setKeyBoardView(AUNumberKeyboardView keyboardView, ScrollView scrollView)

/**
* 绑定外部的 AUNumberKeyboardView
* @param keyboardView
*/
public void setKeyBoardView(AUNumberKeyboardView keyboardView)
```

自定义属性

属性名	说明	类型
footStyle	头部 view 的类型	editStyle，textStyle
amountTitleText	编辑框标题	string，reference
amountHintText	编辑框的提示	string，reference

代码示例

通用代码示例


```
<com.alipay.mobile.antui.amount.AUAmountEditText
android:id="@+id/edit_text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:amountHintText="可用余额500.00"/>

<com.alipay.mobile.antui.amount.AUAmountLabelText
android:id="@+id/label_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"/>

<com.alipay.mobile.antui.amount.AUAmountInputBox
android:id="@+id/amount_input_1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
app:amountTitleText="转账金额"/>

<com.alipay.mobile.antui.amount.AUAmountInputBox
```

```
android:id="@+id/amount_input_2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
app:amountTitleText="转账金额"
app:amountHintText="可用余额500.00"
app:footStyle="textStyle"/>

AUAmountInputBox inputBox1 = (AUAmountInputBox)findViewById(R.id.amount_input_1);
inputBox1.setFootHint("添加转账说明");

AUAmountInputBox inputBox2 = (AUAmountInputBox)findViewById(R.id.amount_input_2);
inputBox2.setFootText("不可输入");
```

带数字键盘的代码示例

```
<?xml version="1.0"encoding="utf-8"?>
<com.alipay.mobile.antui.basic.AULinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res/com.alipay.mobile.antui"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<com.alipay.mobile.antui.basic.AUScrollView
android:id="@+id/scroll"
android:layout_weight="1"
android:layout_width="match_parent"
android:layout_height="match_parent">

<com.alipay.mobile.antui.basic.AULinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<com.alipay.mobile.antui.amount.AUAmountInputBox
android:id="@+id/amount_input_1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
app:amountTitleText="转账金额"/>
</com.alipay.mobile.antui.basic.AULinearLayout>
</com.alipay.mobile.antui.basic.AUScrollView>

<com.alipay.mobile.antui.keyboard.AUNumberKeyboardView
android:id="@+id/keyboard"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:visibility="gone"/>
</com.alipay.mobile.antui.basic.AULinearLayout>



//初始化
keyboardView = (AUNumberKeyboardView) findViewById(R.id.keyboard);
inputBox1 = (AUAmountInputBox)findViewById(R.id.amount_input_1);
```

```
ScrollView scrollView = (ScrollView) findViewById(R.id.scroll);

//绑定键盘
inputBox1.getEditLayout().setKeyBoardView(keyboardView, scrollView);
```

4.3.2 输入框

下文分别介绍 mPaaS 提供的 AUInputBox 、 AUImageInputBox 和 AUTextCodeInputBox 三个输入框组件。其中，AUImageInputBox 和 AUTextCodeInputBox 继承 AUInputBox。

AUInputBox

AUInputBox 组件包含以下内容：

- 一个 AUEditText 的文本输入框
- 一个显示在输入框左侧的标签名
- 一个输入框获取焦点并且内容不为空时会显示的删除按钮

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 取得 UBB 编码后的字符串
*/
public String getUbbStr()

/***
* 设置 emoji 字体大小，单位为 px
*/
public void setEmojiSize(int emojiSize)
/***
* 设置是否支持 emoji
*/
public void setSupportEmoji(boolean isSupport) {
this.supportEmoji = isSupport;
}

/**
* 设置一个 Formatter 来对输入进行格式化
* 设置完成后对已经输入的文字不会立刻生效，需要等待输入才能有效果
*/
public void setTextFormatter(AUFormatter formatter)
```

```
/**
* 设置输入文字是否加粗显示
*
* @param isBold true 为加粗，false 为正常
*/
public void setApprerance(boolean isBold)

/**
* Set a special listener to be called when an action is performed on the
* text view. This will be called when the enter key is pressed, or when an
* action supplied to the IME is selected by the user. Setting this means
* that the normal hard key event will not insert a newline into the text
* view, even if it is multi-line; holding down the ALT modifier will,
* however, allow the user to insert a newline character.
*/
public void setOnEditorActionListener(OnEditorActionListener l)

/**
* Adds a TextWatcher to the list of those whose methods are called whenever
* this TextView's text changes.
*/
public void addTextChangedListener(TextWatcher watcher)


/**
* 输入框输入文字后会显示出删除按钮，此处设置删除按钮的点击事件接收对象
*/
public void setCleanButtonListener(View.OnClickListener listener)

/**
* 设置输入框文字内容
*/
public void setText(CharSequence inputContent)

/**
* 获取文字内容，若文字被格式化，需要调用方处理为需要格式
*/
public String getInputedText()

/**
* 获取输入框的 EditText 控件
*/
public AUEditText getInputEdit()

/**
* 设置标签文本
* @param title 输入的标签文本
*/
public void setInputName(String title)


/**
* 获取输入内容名称（标签名）的控件
*/
public AUTextView getInputName()
```

```
/**
* 输入内容名称字体大小，单位为 px
*/
public void setInputNameTextSize(float textSize)

/**
* 设置输入框字体大小，单位为 px
*/
public void setInputTextSize(float textSize)

/**
* 输入内容文字颜色
*/
public void setInputTextColor(int textColor)

/**
* 设置输入内容类型
*/
public void setInputType(int inputType)
/**
* 设置提示信息
*/
public void setHint(String hintString)

/**
* 设置左侧标签图标
*/
public void setInputImage(Drawable drawable)

/**
* 获取左侧标签图标
*/
public AUImageView getInputImage()

/**
* 设置提示信息颜色
*/
public void setHintTextColor(int textColor)

/**
* 设置输入框的最大输入长度
*
* @param maxlength 若参数 <=0，则不进行长度限制
*/
public void setMaxLength(int maxlength)

/**
* 获取清除按钮控件
*/
public AUIconView getClearButton()

/**
* 获取是否需要显示清除按钮
*/
public boolean isNeedShowClearButton() {
```

```
return isNeedShowClearButton;
}

/**
* 设置是否需要显示清除按钮，若设置 false，则任何情况下清除按钮都不会显示
*/
public void setNeedShowClearButton(boolean isNeedShowClearButton)

/**
* 设置控件的边框风格，包括上、中、下和独立
* 此方法为 AULineGroupItemInterface 接口中的方法
* 当控件结合 LineGroupView 使用时，这个方法会自动被调用到
*
* @param positionStyle，使用 AULineGroupItemInterface 中定义的变量
AULineGroupItemInterface.TOP，CENTER，BOTTOM，NORMAL，LINE，NONE
*/
@Override
public void setItemPositionStyle(int positionStyle)

/**
* 获取输入内容类型
*/
public int getInputType()
```

代码示例


```
<com.alipay.mobile.antui.input.AUInputBox
android:id="@+id/safeInputBox"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="top"
app:inputName="标签1"
app:inputType="textPassword"
app:inputHint="这个输入框会弹出安全键盘"/>

<com.alipay.mobile.antui.input.AUInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:inputName="标签2"
app:inputHint="按提示输入提示"/>

<com.alipay.mobile.antui.input.AUInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="bottom"
app:inputName="转入金额"
app:inputHint="按提示输入提示"/>
```

```
<com.alipay.mobile.antui.input.AUInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"
app:inputImage="@drawable/image"
app:inputName="转入金额"
app:inputHint="按提示输入提示"/>

<com.alipay.mobile.antui.input.AUInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"
app:inputHint="按提示输入提示"/>
```

AUImageInputBox

AUImageInputBox 继承 AUInputBox，包含：

- 一个显示在右侧的 IconView，可展示图标或 Unicode
- 一个显示在右侧的 TextView

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置最右侧功能按钮背景
* 若设置按钮背景为空，则不会显示功能按钮，保持与 AUInputBox 功能一致
*/
public void setLastImgDrawable(Drawable drawable)

/**
* 设置最右侧功能按钮背景
* @param unicode
*/
public void setLastImgUnicode(String unicode)

/**
* 设置最右侧图标是否可见
* @param visible
*/
public void setLastImgBtnVisible(boolean visible)

/**
```

```
* 设置最右侧功能按钮的监听
*/
public void setLastImgClickListener(View.OnClickListener l)

/**
* 设置最右侧的文本
* @param lastText
*/
public void setLastTextView(String lastText)

/**
* 获取最右边的 TextView
*
* @return 获取最右侧TextView
*/
public AUTextView getLastTextView()

/**
* 获取最右边的图标 View
* @return
*/
public AUIconView getLastImgBtn()
```

代码示例

```
<com.alipay.mobile.antui.input.AUImageInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"
app:listItemType="top"
app:inputName="姓名"
app:inputHint="收款人姓名"
app:input_rightIconUnicode="@string/iconfont_phone_contact"/>

<com.alipay.mobile.antui.input.AUImageInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:inputName="卡号"
app:inputHint="收款人储蓄卡号"/>

<com.alipay.mobile.antui.input.AUImageInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="bottom"
app:inputName="银行"
app:inputHint="请选择银行"
app:input_rightIconDrawable="@drawable/table_arrow"/>
```

```
<com.alipay.mobile.antui.input.AUImageInputBox
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"
app:inputName="金额"
app:inputHint="输入转账金额"
app:input_rightText="限额说明"/>
```

AUTextCodeInputBox

AUTextCodeInputBox 继承 AUInputBox，右侧包含一个发送短信验证码的文本按钮。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置发送按钮点击事件 callback
* @param callback 当用户点击发送按钮时，OnSendCallback.onSend() 方法将被回调
*/
public void setOnSendCallback(OnSendCallback callback)

/**
* 当前时间归 0
*/
public void currentSecond2Zero()

/**
* 设置当前时间
*/
public void setCurrentSecond(int current)

/**
* 获取当前时间
*/
public int getCurrentSecond()

/**
* 获取发送按钮
*/
public AUButton getSendCodeButton()

/**
* 供业务调用释放 timer
*/
```

```
public void releaseTimer()


/**
* 按钮开始倒计时
*/
public void scheduleTimer()

public interface SendButtonEnableChecker {
public boolean checkIsEnabled();
}

/**
* 此方法会设置检测 SendButton 是否可用的方法
* 若此方法检测按钮不可用，则调用 updateSendButtonEnableStatus 方法时按钮置灰
* 否则按照倒计时自己的逻辑，调用 updateSendButtonEnableStatus 方法时设置可用状态
* 总之，只有所有检查都可用，最后按钮才可用，否则都置灰
*/
public void setSendButtonEnableChecker(SendButtonEnableChecker checker)

/**
* 根据 SendButtonEnableChecker 和 CheckCodeSendBox 内部状态更新 SendButton 可用状态
* 只有所有检查都可用，最后按钮才可用，否则都置灰
*/
public void updateSendButtonEnableStatus()

/**
* 获取SendResultCallback
*/
public SendResultCallback getSendResultCallback()
```

代码示例

XML：

```
<com.alipay.mobile.antui.input.AUTextCodeInputBox
android:id="@+id/au_textcode_input"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"/>
```

Java：

```
final AUTextCodeInputBox textCodeInputBox = (AUTextCodeInputBox) findViewById(R.id.au_textcode_input);
textCodeInputBox.setOnSendCallback(new OnSendCallback() {
@Override
```

```
public void onSend(final SendResultCallback callback) {
// 这里rpc请求服务端发送验证码..
boolean resendSmsRpcSuccess = true;
if (resendSmsRpcSuccess) {
// 发送验证码成功，开始倒计时
callback.onSuccess();
// 收到验证码..
Toast.makeText(InputActivity.this,"收到验证码: 123456", Toast.LENGTH_SHORT)
.show();
textCodeInputBox.setText("123456");
Log.d(TAG,"输入的验证码为："+ textCodeInputBox.getInputedText());
} else {
// 发送验证码失败，重新enable发送按钮
callback.onFail();
}
}
});
```

自定义属性

下表所列的是以上三个组件的自定义属性参数。

属性名	说明	类型
inputName	输入内容名称	string，reference
inputHint	输入框提示内容	string，reference
maxLength	输入内容最大长度	integer
inputType	输入内容类型	enum，取值有 textNormal、textNumber、textDecimal、textPassword
inputImage	输入框左边的图片	reference
listItemType	条目的背景类型	enum，取值有 top、center、bottom、normal、line、none
input_rightIconUnicode	右侧图标	string，reference
input_rightIconDrawable	右侧图片	reference
input_rightText	右侧超链接文本	string，reference

4.3.3 数字键盘

AUNumberKeyboardView 提供三种状态的数字键盘。

使用说明

- 单独作为一个视图展示，如小程序。
- 与 AUAmountEditText 绑定使用，绑定工具为 AUNumberKeyBoardUtil，已经封装入 AUAmountEditText，具体可参考 AUAmountInputBox 文档 。
- 与普通的 Edittext 绑定使用，绑定工具为 AUNumberKeyBoardUtil，需自行调用。

效果图

依赖

参见 快速开始 。

接口说明

AUAmountEditText

```
/**
* 设置键盘的样式，默认为 STYLE_POINT
* @param style STYLE_POINT、STYLE_X、STYLE_NONE
*/
public void setStyle(int style)

/**
* 设置按钮监听
* @param listener
*/
public void setActionClickListener(OnActionClickListener listener)

/**
* 设置展示状态监听
* @param windowStateChangeListener
*/
public void setWindowStateChangeListener(WindowStateChangeListener windowStateChangeListener)

/**
* 展示
*/
public void show()

/**
* 消失
*/
public void hide()

/**
* 返回展示状态
* @return
*/
public boolean isShow()
```

AUNumberKeyBoardUtil

```
/**
* 传递入 EditText 以及 AUNumberKeyboardView
* @param context
* @param editText
* @param keyboardView
*/
public AUNumberKeyBoardUtil(Context context, EditText editText, AUNumberKeyboardView keyboardView)

/**
* 设置滚动 view
* @param view
*/
public void setScrollView(ScrollView view)

/**
* 显示数字键盘
*/
public void showKeyboard()

/**
* 隐藏数字键盘
*/
public void hideKeyboard()
```

代码示例

AUAmountEditText

```
AUNumberKeyboardView auNumberKeyboardView = new AUNumberKeyboardView(this,
AUNumberKeyboardView.STYLE_POINT, new AUNumberKeyboardView.OnActionClickListener() {
@Override
public void onNumClick(View view, CharSequence num) {

}

@Override
public void onDeleteClick(View view) {

}

@Override
public void onConfirmClick(View view) {

}

@Override
public void onCloseClick(View view) {

}
});
```

AUNumberKeyBoardUtil

XML：

```
<com.alipay.mobile.antui.basic.AULinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<com.alipay.mobile.antui.basic.AUScrollView
android:id="@+id/scroll"
android:layout_weight="1"
android:layout_width="match_parent"
android:layout_height="match_parent">

<com.alipay.mobile.antui.basic.AULinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<EditText
android:id="@+id/editText"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"/>
</com.alipay.mobile.antui.basic.AULinearLayout>
</com.alipay.mobile.antui.basic.AUScrollView>

<com.alipay.mobile.antui.keyboard.AUNumberKeyboardView
android:id="@+id/keyboard"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:visibility="gone"/>
</com.alipay.mobile.antui.basic.AULinearLayout>
```

Java：

```
keyBoardUtil = new AUNumberKeyBoardUtil(context, editText, keyboardView);
keyBoardUtil.setScrollView(scrollView);
```

4.3.4 搜索栏

AUSearchBar（原 APSocailSearchBar）提供包含返回按钮、搜索框和右侧搜索按钮的搜索标题栏。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置最大输入长度
*/
public void setInputMaxLength(int length);

/**
* 获取返回按钮
* @return
*/
public AUIconView getBackButton() ;

/**
* 获取删除按钮
* @return
*/
public AUIconView getClearButton();

/**
* 获取搜索输入框
* @return
*/
public AUEditText getSearchEditView();

/**
* 获取搜索按钮
* @return
*/
public AUIconView getSearchButton() ;

/**
* 获取搜索布局
* @return
*/
public AURelativeLayout getSearchRelativeLayout() ;

/**
* 获取语音搜索按钮
* @return
*/
public AUIconView getVoiceButton();

/**
* 增加编辑事件监听
*/
public void setEditChangedListener(TextWatcher watcher)
```

自定义属性

属性名	说明	类型
isShowSearchBtn	是否显示搜索按钮	boolean
isShowVoiceSearch	是否显示语音搜索	boolean

searchEditText	搜索框默认文本	string，reference
searchEditHint	搜索框默认提示内容	string，reference
searchButtonText	搜索按钮的文本	string，reference
inputMaxLength	搜索框的最长限制	integer，reference
hintIconUnicode	编辑框左侧图标的 Unicode	string，reference
hintIconDrawable	编辑框左侧图标的资源	reference
backIconUnicode	返回按钮的 Unicode	string，reference
backIconDrawable	返回按钮的资源	reference
editHintColor	编辑框内提示内容的颜色	color，reference
editTextColor	编辑框内文本的颜色	color，reference
editIconColor	编辑框内图标的颜色	color，reference

代码示例

```
<com.alipay.mobile.antui.basic.AUSearchBar
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
aui:searchEditText="输入文本"
aui:isShowSearchBtn="true"
aui:isShowVoiceSearch="true"/>

<com.alipay.mobile.antui.basic.AUSearchBar
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
aui:searchEditHint="暗文本提示"
aui:isShowSearchBtn="true"
aui:isShowVoiceSearch="true"/>
```

4.3.5 搜索框

AUSearchInputBox（原 APSocialTagSearchBar）提供包含搜索框、右侧搜索按钮的搜索标题栏。使用该组件时，需要设置 View 的高度。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置最大输入长度
*/
public void setInputMaxLength();

/**
* 获取删除按钮
* @return
*/
public AUIconView getClearButton();

/**
* 获取搜索输入框
* @return
*/
public AUEditText getSearchEditView();

/**
* 获取语音搜索按钮
* @return
*/
public AUIconView getVoiceButton();
```

自定义属性

属性名	说明	类型
isShowSearchBtn	是否显示搜索按钮	boolean
isShowVoiceSearch	是否显示语音搜索	boolean
searchEditText	搜索框默认文本	string，reference
searchEditHint	搜索框默认提示内容	string，reference
inputMaxLength	搜索框最长限制	integer，reference
hintIconUnicode	编辑框左侧图标的 Unicode	string，reference
hintIconDrawable	编辑框左侧图标的资源	reference
editHintColor	编辑框内提示内容的颜色	color，reference
editTextColor	编辑框内文本的颜色	color，reference
editIconColor	编辑框内图标的颜色	color，reference

代码示例

XML：

```
<com.alipay.mobile.antui.basic.AUSearchInputBox
android:layout_width="match_parent"
android:layout_height="52dp"
android:layout_marginTop="10dp"
app:searchEditHint="暗文本提示"/>


AUSearchInputBox inputBox = new AUSearchInputBox(this);
ViewGroup.LayoutParams layoutParams = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,300);
inputBox.setLayoutParams(layoutParams);

layout.addView(inputBox);
```

4.4 条目组件

4.4.1 辅助说明组件

AUAssistLabelView 是一个 TextView 组件，用来显示辅助说明文本。

效果图

依赖

参见 快速开始 。

代码示例

```
<com.alipay.mobile.antui.basic.AUAssistLabelView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="若关闭，当收到朋友消息时，通知提示将不显示发言人和内容摘要"/>


<com.alipay.mobile.antui.basic.AUAssistLabelView
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:isHead="true"
android:text="我是表头"/>
```

4.4.2 银行卡条目组件

AUBankCardItem 组件用于提供包含银行名称、银行 logo 和银行账号的银行卡条目。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 获取银行名称 view
* @return
*/
public AUEmptyGoneTextView getBankName();

/**
* 获取银行账号 view
* @return
*/
public AUEmptyGoneTextView getBankNumber();

/**
* 获取银行 logo view
* @return
*/
public AUCircleImageView getBankImage();

/**
* 同时设置银行名称和账号
* @param bankName
* @param bankNum
*/
public void setBankInfo(String bankName, String bankNum) ;
```

代码示例

```
AUBankCardItem cardItem = new AUBankCardItem(this);
cardItem.setBankInfo("银行名称","后4位尾数");
cardItem.getBankImage().setImageResource(R.drawable.image);
```

4.4.3 卡券条目组件

效果图

依赖

参见 快速开始 。

代码示例

```
AUCouponsItem couponsItem1 = new AUCouponsItem(this);
couponsItem1.setCouponsInfo("小标题","100元代金券","");
couponsItem1.getCouponsImage().setImageResource(R.drawable.image);

AUCouponsItem couponsItem2 = new AUCouponsItem(this);
couponsItem2.setCouponsInfo("","100元代金券","副标题非必填");

AUCouponsItem couponsItem3 = new AUCouponsItem(this);
couponsItem3.setCouponsInfo("小标题","100元代金券","");
couponsItem3.setCouponsAssitDes("483米有门店");
couponsItem3.getCouponsImage().setImageResource(R.drawable.image);
```

4.4.4 条目组件

AUListItem 是用于替换 APTableView 的系列控件。

原 APTableView 包括：

- APAbsTableView
- APTableView
- APExtTableView
- APMultiTextTableView
- APRadioTableView
- APLineTableView
- APListItemView
- APTableRowsView

以上控件可分别对应以下几种 ListItem 控件：

- AUSingleTitleListItem
- AUDoubleTitleListItem
- AUCheckBoxListItem
- AUSwitchListItem
- AUMultiListItem

- AUParallelTitleListItem
- AULineBreakListItem

效果图

AUSingleTitleListItem

AUDoubleTitleListItem

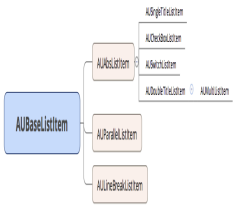
AUCheckBoxListItem

AUSwitchListItem

依赖

参见 快速开始 。

接口说明



基础接口

```
/**
* 设置 item 类型，上中下
*
```

```
* @param positionStyle AULineGroupItemInterface.NORMAL TOP BOTTOM CENTER LINE NONE
*/
public void setItemPositionStyle(int positionStyle)

/**
* 设置右侧箭头是否可见
* @param isVisible
*/
public void setArrowVisibility(boolean isVisible)
```

AUParallelListItem

```
/**
* 同时设置四个位置的 text
* @param leftText
* @param leftSubText
* @param rightText
* @param rightSubText
*/
public void setParallelText(String leftText, String leftSubText, String rightText, String rightSubText)

/**
* 设置左边的主文本
* @param leftText
*/
public void setLeftText(String leftText)

/**
* 设置右边的主文本
* @param rightText
*/
public void setRightText(String rightText)

/**
* 设置左边的副文本
* @param leftSubText
*/
public void setLeftSubText(String leftSubText)

/**
* 设置右边的副文本
* @param rightSubText
*/
public void setRightSubText(String rightSubText)
```

AULineBreakListItem

```
/**
* 设置左右文本
* @param left
* @param right
*/
public void setText(String left, String right)
```

```
/**
* 获取左边 TextView
* @return
*/
public AUTextView getLeftText()

/**
* 获取右边 TextView
* @return
*/
public AUTextView getRightText()
```

公共接口

```
/**
* 设置 icon 图片大小
*/
public void setIconSize(float width, float height)

/**
* 获取左侧主体文字
* @return
*/
public CharSequence getLeftText()

/**
* 设置左侧主体文字
* @param text
*/
public void setLeftText(CharSequence text)

/**
* 设置左侧主体文字颜色
* @param color
*/
public void setLeftTextColor(int color)

/**
* 获取左侧图片 view
* @return
*/
public AURoundImageView getLeftRoundImageView()
public AUImageView getLeftImageView()

/**
* 设置左侧图片
* @param resId
*/
public void setLeftImage(int resId)

/**
* 设置左侧图片
* @param drawable
*/
```

```
public void setLeftImage(Drawable drawable)

/**
* 由于已有接口已经有 setLeftImage 时对 Visibility 进行操作
* 所以此接口调用后再调用 setLeftImage 时系统将会重新设置 Visibility。
*
* @param vis View.GONE
*/
public void setLeftImageVisibility(int vis)

/**
* 获取左侧文本信息
* @return
*/
public AUTextView getLeftTextView()
}
```

AUSingleTitleListItem

```
/**
* 设置右侧选中按钮
* @param checked
*/
public void setItemChecked (boolean checked)


/**
* 设置右侧文本信息
* @param text
*/
public void setRightText(CharSequence text)

/**
* 设置右侧文本颜色
* @param color
*/
public void setRightTextColor(int color)

/**
* 设置右侧图片
*/
public void setRightImage(int resId)
public void setRightImage(Bitmap bitmap)
public void setRightImage(Drawable drawable)

/**
* 获取右侧文本 View
* @return
*/
public AUTextView getRightTextView()

/**
* 获取右侧图片 View
* @return
*/
```

```
public AUImageView getRightImageView()

/**
* 设置右侧文本信息
* @param text
*/
public void setRightButtonText(CharSequence text)

/**
* 获取 button
* @return
*/
public AUProcessButton getProcessButton()

/**
* 点击响应事件
* @param listener
*/
public void setButtonClickListener(OnClickListener listener)

/**
* 设置右侧样式
* @param type AUAbsListItem.TEXT_IMAGE AUAbsListItem.BUTTON
*/
public void setRightType(int type)
```

AUCheckBoxListItem

```
/**
* 获取左侧的勾选图标
* @return
*/
public AUCheckIcon getLeftCheckIcon()

/**
* 设置icon状态
* @param status AUCheckIcon.STATE_CHECKED|STATE_UNCHECKED|STATE_DISABLED
*/
public void setCheckstatus(int status)

/**
* 获取勾选状态
* @return
*/
public int getIconState()
```

AUSwitchListItem

```
/**
* 设置开关状态监听
* @param onCheckedChangeListener
*/
public void setOnSwitchListener (CompoundButton.OnCheckedChangeListener onCheckedChangeListener)
```

```
/**
* 获取开关
* @return
*/
public AUSwitch getSwitch()

/**
* 返回开关状态
* @return 开关是否打开
*/
public boolean isSwitchOn()

/**
* 设置开关状态
* @param status
*/
public void setSwitchStatus(boolean status)

/**
* 设置enable/disable
* @param enabled
*/
public void setSwitchEnabled(boolean enabled)
```

AUDoubleTitleListItem

```
/**
* 设置左侧副文本
* @param text
*/
public void setLeftSubText(CharSequence text)

/**
* 设置右侧文本
* @param text
*/
public void setRightText(CharSequence text)

/**
* 设置右侧文本字体颜色
* @param color
*/
public void setRightTextColor(int color)

/**
* 获取右侧文本 View
* @return
*/
public AUTextView getRightTextView()

/**
* 获取左侧副文本 View
* @return
*/
```

```
public AUTextView getLeftSubTextView()

/**
* 设置右侧文本信息
* @param text
*/
public void setRightButtonText(CharSequence text)

/**
* 获取button
* @return
*/
public AUProcessButton getProcessButton()

/**
* 点击响应事件
* @param listener
*/
public void setButtonClickListener(OnClickListener listener)

/**
* 设置右侧样式
* @param type AUAbsListItem.TEXT_IMAGE AUAbsListItem.BUTTON
*/
public void setRightType(int type)
```

AUMultiListItem

```
/**
* 左侧增加扩展 view
* @param view
*/
public void addLeftAssistantView(View view)

/**
* 设置左侧副文本
* @param text
*/
public void setLeftSubText(CharSequence text)

/**
* 获取副标题文本
* @return
*/
public AUEmptyGoneTextView getLeftSubTextView()
```

自定义属性

属性名	说明	类型
listItemType	设置位置样式	normal/top/bottom/center/line/none
listLeftText	左侧文案	string，reference
listLeftSubText	左侧副文案	string，reference

listLeftTextSize	左侧文案字号	dimension
listLeftSubTextSize	左侧副文案字号	dimension
listLeftTextColor	左侧文案颜色	color，reference
listLeftSubTextColor	左侧副文案颜色	color，reference
listLeftImage	左侧图标	reference
listLeftImageWidth	左侧图片宽度	dimension，reference
listLeftImageHeight	左侧图片高度	dimension，reference
listShowArrow	是否显示右侧箭头	boolean
listArrowType	箭头方向	arrow_right/arrow_down/arrow_up
listRightText	右侧文案	string，reference
listRightSubText	右侧副文案	string，reference
listRightType	右侧样式	text_image/button
listRightImage	右侧图片	string，reference
listShowCheck	右侧勾选图片	boolean

说明：

- 各控件支持使用的属性在下面 代码示例（XML）中展示。
- 如果需要按下背景变色效果，请加上属性 android:clickable="true"。
- 控件高度、左图片宽高由业务自定义。

代码示例

引入：xmlns:app="http://schemas.android.com/apk/res/com.alipay.mobile.antui"

AUParallelListItem

```
<com.alipay.mobile.antui.tablelist.AUParallelTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="top"
app:listLeftText="标题一"
app:listLeftSubText="内容一"
app:listRightText="标题二"
app:listRightSubText="内容二"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUParallelTitleListItem
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftText="标题一"
app:listLeftSubText="内容一"
app:listRightSubText="内容二"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUParallelTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftText="标题一"
app:listLeftSubText="内容一"
app:listRightText="标题二"
app:listShowArrow="false"/>
```

AULineBreakListItem

```
<com.alipay.mobile.antui.tablelist.AULineBreakListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="top"
app:listLeftText="主体信息"
app:listRightText="单行文字过多，换行与左侧文字间距保持30px"/>

<com.alipay.mobile.antui.tablelist.AULineBreakListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:listLeftText="单行文字过多，换行与右侧文字间距保持30px"
app:listRightText="详细信息"/>

<com.alipay.mobile.antui.tablelist.AULineBreakListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:listLeftText="单行文字"
app:listRightText="详细信息"/>

<com.alipay.mobile.antui.tablelist.AULineBreakListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="bottom"
app:listLeftText="单行文字过多，换行与右侧文字间距保持30px"
app:listRightText="单行文字过多，换行与右侧文字间距保持30px"/>
```

AUSingleTitleListItem

```
<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="top"
app:listLeftText="单行列表"
app:listRightText="详细内容"/>

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftText="单行文字过多，换行与右侧文字间距保持30px"
app:listRightText="详细信息"/>

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:listLeftText="单项选择列表"
app:listShowCheck="true"/>

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:listLeftImage="@drawable/image"
app:listLeftText="正常图片"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="center"
app:listLeftImage="@drawable/image"
app:listLeftImageSizeType="size_large"
app:listLeftText="大图片"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:hasRound="true"
app:listItemType="center"
app:listLeftImage="@drawable/image"
app:listLeftImageHeight="36dp"
app:listLeftImageWidth="36dp"
app:listLeftText="自定义图片大小"
```

```
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftText="标题"
app:listRightImage="@drawable/image"
app:listRightText="内容展示加长"/>


<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
android:id="@+id/button_item"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:listItemType="bottom"
app:listLeftImage="@drawable/image"
app:listLeftText="标题"
app:listRightText="试一试"
app:listRightType="button"/>
```

AUDoubleTitleListItem

```
<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftSubText="支付宝使用航班提醒等服务。"
app:listLeftText="标题一"
app:listRightText="10:30"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftImage="@drawable/testapp_icon"
app:listLeftSubText="说明文本"
app:listLeftText="正常图片"/>

<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftImage="@drawable/testapp_icon"
app:listLeftImageSizeType="size_large"
```

```
app:listLeftSubText="说明文本"
app:listLeftText="大图片"
app:listRightText="10:30"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="bottom"
app:listLeftImage="@drawable/testapp_icon"
app:listLeftImageSizeType="size_multi"
app:listLeftSubText=" “全球未来机场计划”是指未来游客在海外机场，支付宝使用航班提醒等服务。 "
app:listLeftText="图文列表图片"
app:listShowArrow="false"/>

<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="center"
app:listLeftImage="@drawable/image"
app:listLeftImageSizeType="size_large"
app:listLeftSubText="说明文本"
app:listLeftText="大图片"
app:listRightText="试一试"
app:listRightType="button"/>


<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
android:id="@+id/testLitItem"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginBottom="20dp"
android:layout_marginTop="10dp"
android:clickable="true"
app:listItemType="normal"
app:listLeftImage="@drawable/testapp_icon"
app:listLeftImageHeight="70dp"
app:listLeftImageWidth="70dp"
app:listLeftSubText="点击button设置type"
app:listLeftText="自定义图片大小"/>
```

AUCheckBoxListItem

```
<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:listItemType="top"
```

```
app:listLeftText="多项选择列表"/>

<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:checkIconState="checked"
app:listItemType="center"
app:listLeftText="多项选择列表"/>

<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:checkIconState="cannot_uncheck"
app:listItemType="bottom"
app:listLeftText="多项选择列表"/>

<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:clickable="true"
app:checkIconState="cannot_check"
app:listItemType="bottom"
app:listLeftText="多项选择列表"/>
```

AUSwitchListItem

```
<com.alipay.mobile.antui.tablelist.AUSwitchListItem
android:layout_width="match_parent"
android:layout_height="48dp"
app:listItemType="top"
app:listLeftText="标题"/>

<com.alipay.mobile.antui.tablelist.AUSwitchListItem
android:id="@+id/disable_switch_list_item"
android:layout_width="match_parent"
android:layout_height="48dp"
app:listItemType="bottom"
app:listLeftText="标题"/>
```

4.5 结果页组件

4.5.1 进度页

AUFlowResultView 支持显示带进度的结果页，用 FlowResult 表示一个节点，每个节点都可以设置不同的类型和辅助文案。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 清除所有的 FlowStepView
*/
public void clearFlows() {
removeAllViews();
}

/**
* 设置 FlowResult 列表，并生成对应的 FlowStepView
*
* @param flowResultList
*/
public void setFlows(List<FlowResult> flowResultList) {
```

FlowResult 接口

```
/**
* 构造一个 FlowResult
*
* @param resultStatus 节点状态，取值为 ResultConstant.RESULT_STATUS_ENUM_XX
* @param statusIcon 状态图标，类型为 ResultStatusIcon 枚举
* @param mainInfoText 主文案
* @param subTitles 次级文案列表
*/
public FlowResult(int resultStatus, ResultStatusIcon statusIcon, String mainInfoText,
List<String> subTitles);

/**
* 构造一个 FlowResult
*
* @param resultStatus 节点状态，取值为 ResultConstant.RESULT_STATUS_ENUM_XX
* @param statusIconId 状态 icon res id
* @param mainInfoText 主文案
* @param subTitles 次级文案列表
*/
public FlowResult(int resultStatus, int statusIconId, String mainInfoText,
List<String> subTitles);
```

代码示例

```
AUFlowResultView flowResultView = (AUFlowResultView) findViewById(R.id.flow_result_view);
List<FlowResult> flows = new ArrayList<FlowResult>();
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_OK, ResultStatusIcon.OK,
"支付成功", Arrays.asList("辅助说明文本","辅助说明文本")));
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_OK, ResultStatusIcon.PENDING,
"标签文本", Arrays.asList("辅助说明文本","辅助说明文本")));
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_NORMAL, ResultStatusIcon.PENDING,
"标签文本", Arrays.asList("辅助说明文本","辅助说明文本")));
flowResultView.setFlows(flows);
```

4.5.2 异常页

AUNetErrorView（原 APFlowTipView）提供一个网络异常空白页。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置简易模式
* @param isSimple
*/
public void setIsSimpleType(boolean isSimple);

/**
* 设置网络异常模式
* @param type
*/
public void resetFlowTipType(int type);

/**
* 设置按钮的属性
*
* @param text
* @param clickListener
*/
public void setAction(String text, OnClickListener clickListener) ;

/**
* 取消按钮
*/
public void setNoAction();
```

```
/**
* 设置提示信息
*
* @param text
*/
public void setTips(String text) ;

/**
* 设置辅助提示信息
* @param text
*/
public void setSubTips(String text) ;

/**
* 获取操作按钮
* @return
*/
public AUButton getActionButton();

/**
* 获取图片view
* @return
*/
public AUImageView getImageView() ;
```

自定义属性

属性名	说明	类型
netErrorType	网络异常的状态	signalError，empty，warning，overflow
isSimpleMode	是否简版	boolean

代码示例

```
<com.alipay.mobile.antui.basic.AUNetErrorView
android:id="@+id/net_error"
android:layout_width="match_parent"
android:layout_height="match_parent"
app:netErrorType="signalError"/>
```

4.5.3 二维码页面

AUQRCodeView 提供一个二维码页面。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置头像名称
* @param name
*/
public void setAvatarName(CharSequence name)

/**
* 设置二维码信息
* @param title
* @param description
*/
public void setCodeInfo(CharSequence title, CharSequence description)

/**
* 设置二维码标题
* @param title
*/
public void setCodeTitle(CharSequence title)

/**
* 设置二维码描述信息
* @param description
*/
public void setCodeDescription(CharSequence description)

/**
* 设置按钮信息 带吱口令图标
* @param title
* @param content
*/
public void setButtonInfo(CharSequence title, CharSequence content)

/**
* 设置按钮信息
* @param title
* @param content
* @param isToken 有无吱口令图标
*/
public void setButtonInfo(CharSequence title, CharSequence content, boolean isToken)

/**
* 设置按钮标题
* @param title
*/
public void setButtonTitle(CharSequence title)

/**
* 设置标题是否有图标
* @param isToken
*/
public void setButtonToken(boolean isToken)
```

```
/**
* 设置按钮是否可见
* @param isVisible
*/
public void setButtonVisibility(boolean isVisible)

/**
* 设置按钮内容信息
* @param content
*/
public void setButtonContent(CharSequence content)

/**
* 获取头像 imageView
* @return
*/
public AUImageView getAvatarImage()

/**
* 获取头像名称 View
* @return
*/
public AUTextView getAvatarName()

/**
* 获取二维码 imageView
* @return
*/
public AUImageView getCodeImage()

/**
* 获取二维码标题
* @return
*/
public AUTextView getCodeTitle()

/**
* 获取二维码描述信息
* @return
*/
public AUEmptyGoneTextView getCodeDescription()

/**
* 获取按钮
* @return
*/
public AULinearLayout getButton()

/**
* 获取按钮标题
* @return
*/
public AUTextView getButtonTitle()

/**
```

```
* 获取按钮内容信息
* @return
*/
public AUEmptyGoneTextView getButtonContent()
```

代码示例

```
AUQRCodeView codeView = new AUQRCodeView(this);
codeView.setAvartarName("生活号名称");
codeView.setCodeInfo("用支付宝扫二维码，加入该生活圈","该二维码将在2017年11月05日失效");
codeView.setButtonInfo("点击生成吱口令","推荐生活号給微信、QQ好友");
codeView.getCodeImage().setImageResource(R.drawable.qr_default);
```

4.5.4 结果页

AUResultView 提供一个带图标、三级文案的结果页。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置图标
*
* @param iconRes 图标资源 ID
*/
public void setIcon(@DrawableRes int iconRes);

/**
* 设置主标题文案
*
* @param text 文案内容
*/
public void setMainTitleText(CharSequence text);

/**
* 设置次标题文案
*
* @param text 文案内容
*/
public void setSubTitleText(CharSequence text);

/**
```

```
* 设置辅助标题文案
*
* @param text 文案内容
*/
public void setThirdTitleText(CharSequence text);

/**
* 设置辅助标题文案，带删除线效果
*
* @param text 文案内容
* @param strikeThrough 是否显示删除线
*/
public void setThirdTitleText(CharSequence text, boolean strikeThrough);
```

自定义属性

属性名	说明	类型
icon	图标	reference
mainTitleText	一级文案	string，reference
subTitleText	二级文案	string，reference
thirdTitleText	三级文案	string，reference

代码示例

XML：

```
<com.alipay.mobile.antui.status.AUResultView
android:id="@+id/result_view2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="20dp"
app:icon="@drawable/icon_result_alipay"
app:mainTitleText="支付成功"
app:subTitleText="998.00"
app:thirdTitleText="1098.00元"/>
```

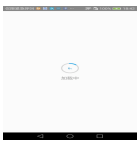
4.6 加载组件

AULoadingView 组件提供包含进度图案、加载进度、加载中文案等的加载页。

依赖

参见 快速开始 。

效果图



接口说明

AULoadingView

```
/**
* 构造方法
* @param context 包含 antu 依赖的页面上下文
*/
public AULoadingView(Context context)
/**
* 设置进度
* @param curentProgress 进度
*/
public void setCurrentProgress(int curentProgress)
```

AUPullLoadingView

```
/**
* 构造方法
* @param context 包含 antu 依赖的页面上下文
*/
public AUPullLoadingView(Context context)

/**
* 设置进度图案
* @param drawable
*/
public void setProgressDrawable(Drawable drawable)

/**
* 设置回弹图案
* @param mIndicatorUpDrawable
*/
public void setIndicatorUpDrawable

/**
* 设置加载中文案
* @param loadingText
*/
public void setLoadingText(String loadingText)

/**
* 设置拖拽中文案
* @param indicatorText
```

```
*/
public void setIndicatorText(String indicatorText)
```

AUDragLoadingView

```
/**
* 构造方法
* @param context 包含 antu 依赖的页面上下文
*/
public AUDragLoadingView(Context context)
/**
* 设置加载中文案
* @param text
*/
public void setLoadingText(CharSequence text)
```

代码示例

AULoadingView

```
private AULoadingView mAULoadingView mAULoadingView = (AULoadingView) findViewById(R.id.loadingView);
private Handler mHandler = new Handler() {
@Override
public void handleMessage(Message msg) {
super.handleMessage(msg);
mAULoadingView.setCurrentProgress(mCurrentProgress);
}
};
protected void onResume() {
super.onResume();
new Thread(new Runnable() {
@Override
public void run() {
while (mCurrentProgress < 100) {
try {
Thread.currentThread().sleep(500);
mCurrentProgress++;
mHandler.sendEmptyMessage(0);
} catch (Exception e) {
Log.e("EmptyPageLoadingActivity",e.getMessage());
}
}

}
}).start();
}
```

AUPullLoadingView

```
@Override
public AUPullLoadingView getOverView() {
```

```
mAUPullLoadingView2 = (AUPullLoadingView) LayoutInflater.from(getBaseContext())
.inflate(R.layout.au_framework_pullrefresh_overview, null);
return mAUPullLoadingView2;
}
```

AUDragLoadingView

```
mAUDragLoadingView = (AUDragLoadingView) findViewById(R.id.dragLoadingView);
findViewById(R.id.modifyLoadingText).setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
mAUDragLoadingView.setLoadingText("修改后的文案...");
}
});
```

4.7 导航组件

4.7.1 轮播组件

AUBannerView 轮播组件用于实现图片轮播效果。

效果图

默认提供白底的 AUTitleBar 控件：

依赖

参见 快速开始 。

代码示例

```
BannerView bannerView = new BannerView(this, 1000);
layout.addView(bannerView);

List<BannerView.BannerItem> items = new ArrayList<BannerView.BannerItem>();
items.add(new BannerView.BannerItem());
items.add(new BannerView.BannerItem());
items.add(new BannerView.BannerItem());
final List<String> list = new ArrayList<String>();
String color1 ="#111111";
String color2 ="#666666";
String color3 ="#eeeeee";
list.add(color1);
list.add(color2);
list.add(color3);
```

```
BannerView.BaseBannerPagerAdapter adapter = new BannerView.BaseBannerPagerAdapter(bannerView,items) {
@Override
public View getView(ViewGroup container, int position) {
TextView tv = new TextView(CarouselActivity.this);
tv.setBackgroundColor(Color.parseColor(list.get(position)));
container.addView(tv);
return tv;
}
};

bannerView.setAdapter(adapter);
```

4.7.2 列表组件

AUPinnedSectionListView 提供分组的 ListView，在滑动中固定每个分组的标题。

说明：若要使用此控件，数据模型需区分 type，若不区分，则是普通的 ListView。

效果图

依赖

参见 快速开始 。

代码示例

```
public class PinnedSectionActivity extends Activity{

private AUPullRefreshView pullRefreshView;
AUPullLoadingView mAUPullLoadingView;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.pinned_layout);
pullRefreshView = (AUPullRefreshView) findViewById(R.id.pull_refresh);
final AUPinnedSectionListView pinnedSectionListView = (AUPinnedSectionListView) findViewById(R.id.list_view);

TextView tv = new TextView(this);
tv.setText("nihao");
pinnedSectionListView.addHeaderView(tv);
pullRefreshView.setRefreshListener(new AUPullRefreshView.RefreshListener() {

@Override
public void onRefresh() {

pullRefreshView.autoRefresh();
```

```
pullRefreshView.postDelayed(new Runnable() {

@Override
public void run() {
pullRefreshView.refreshFinished();

}
}, 1000);

}

@Override
public AUPullLoadingView getOverView() {

mAUPullLoadingView = (AUPullLoadingView) LayoutInflater.from(getBaseContext())
.inflate(com.alipay.mobile.antui.R.layout.au_framework_pullrefresh_overview, null);
Date date = new Date(1466577757265L);
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
String dateString = formatter.format(date);
mAUPullLoadingView.setIndicatorText(dateString);
mAUPullLoadingView.setLoadingText(dateString);
return mAUPullLoadingView;
}

@Override
public boolean canRefresh() {
return true;
}
});


final SimpleAdapter adapter = new SimpleAdapter(this, android.R.layout.simple_list_item_1, android.R.id.text1);

pinnedSectionListView.setAdapter(adapter);

pinnedSectionListView.onFinishLoading(true);
pinnedSectionListView.setOnLoadMoreListener(new AUPinnedSectionListView.OnLoadMoreListener() {
@Override
public void onLoadMoreItems() {

pinnedSectionListView.postDelayed(new Runnable() {
@Override
public void run() {
pinnedSectionListView.onFinishLoading(false);
}
}, 3000);

}
});
}


static class SimpleAdapter extends ArrayAdapter<Item> implements
AUPinnedSectionListView.PinnedSectionListAdapter {
```

```
public SimpleAdapter(Context context, int resource, int textViewResourceId) {
super(context, resource, textViewResourceId);
generateDataset('A', 'Z', false);
}

public void generateDataset(char from, char to, boolean clear) {

if (clear) clear();

final int sectionsNumber = to - from + 1;
prepareSections(sectionsNumber);

int sectionPosition = 0, listPosition = 0;
for (char i=0; i<sectionsNumber; i++) {
Item section = new Item(Item.SECTION, String.valueOf((char)('A' + i)));
section.sectionPosition = sectionPosition;
section.listPosition = listPosition++;
onSectionAdded(section, sectionPosition);
add(section);

final int itemsNumber = (int) Math.abs((Math.cos(2f*Math.PI/3f * sectionsNumber / (i+1f)) * 25f));
for (int j=0;j<itemsNumber;j++) {
Item item = new Item(Item.ITEM, section.text.toUpperCase(Locale.ENGLISH) +" -" + j);
item.sectionPosition = sectionPosition;
item.listPosition = listPosition++;
add(item);
}

sectionPosition++;
}
}

protected void prepareSections(int sectionsNumber) { }
protected void onSectionAdded(Item section, int sectionPosition) { }

@Override public View getView(int position, View convertView, ViewGroup parent) {
TextView view = (TextView) super.getView(position, convertView, parent);
view.setTextColor(Color.DKGRAY);
view.setTag(""+ position);
Item item = getItem(position);
if (item.type == Item.SECTION) {
//view.setOnClickListener(PinnedSectionListActivity.this);
view.setBackgroundColor(Color.parseColor("#ff0000"));
}
return view;
}

@Override public int getViewTypeCount() {
return 2;
}

@Override public int getItemViewType(int position) {
return getItem(position).type;
}

@Override
```

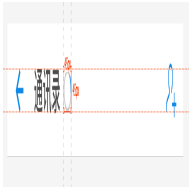
```
public boolean isItemViewTypePinned(int viewType) {
return viewType == Item.SECTION;
}

}

static class Item {

public static final int ITEM = 0;
public static final int SECTION = 1;

public final int type;
public final String text;

public int sectionPosition;
public int listPosition;

public Item(int type, String text) {
this.type = type;
this.text = text;
}

@Override public String toString() {
return text;
}

}
}
```

4.7.3 标题栏组件

AUTitleBar 提供包含返回按钮、标题文案、标题栏上的进度条、左按钮（文字和图标）、右按钮（文字和图标）的标题栏。

效果图

默认提供白底的 AUTitleBar 控件：



依赖

参见 快速开始 。

接口说明

```
/**
* 设置按钮的 drawable
* @param iconView
* @param resId
*/
public void setBtnImage(AUIconView iconView, int resId) ;

/**
* 设置按钮的大小和颜色
* @param iconView
* @param size
* @param color
*/
public void setIconFont(AUIconView iconView, int size, int color);


/**
* 获取返回按钮
* @return
*/
public AUIconView getBackButton() ;

/**
* 获取左按钮
* @return
*/
public AURelativeLayout getLeftButton() ;

/**
* 获取右按钮
* @return
*/
public AURelativeLayout getRightButton() ;

/**
* 获取进度菊花
* @return
*/
public AUProgressBar getProgressBar() ;

/**
* 获取标题文本view
* @return
*/
public AUTextView getTitleText() ;

/**
* 获取标题容器
* @return
*/
public AURelativeLayout getTitleContainer() ;
```

```
/**
* 获取标题栏区域
* @return
*/
public AURelativeLayout getTitleBarRelative() ;

@Override
public void setBackgroundDrawable(Drawable backgroundDrawable) ;

/**
* 设置进度的旋转资源
* @param progressDrawable
*/
public void setProgressBarDrawable(Drawable progressDrawable) ;

/**
* 设置标题的文字及样式，参数若使用默认值，则置 null 或 0
* @param text
* @param textSize
* @param textColor
*/
public void setTitleText(String text, int textSize, int textColor) ;

/**
* 设置标题的文字
* @param text
*/
public void setTitleText(String text) ;

/**
* 设置返回按钮的资源、大小、颜色，若为 null 或者 0，则保持默认值
* @param drawable
* @param size
* @param color
*/
public void setBackBtnInfo(Object drawable, int size, int color);

/**
* 设置返回按钮的资源
* @param drawable
*/
public void setBackBtnInfo(Object drawable);

/**
* 设置左按钮的资源、大小、颜色，若为 null 或者 0，则保持默认值
* @param drawable
* @param size
* @param color
* @param isText
*/
public void setLeftBtnInfo(Object drawable, int size, int color, boolean isText);

/**
* 设置左按钮的资源
* @param drawable
*/
```

```
public void setLeftButtonIcon(Drawable drawable);

public void setLeftButtonIcon(String unicode);

public void setLeftButtonText(String text);

/**
* 设置左按钮的颜色、大小
* @param size
* @param color
* @param isText
*/
public void setLeftButtonFont(int size, int color, boolean isText);

/**
* 设置右按钮的资源、大小、颜色，若为 null 或者 0，则保持默认值
* @param drawable
* @param size
* @param color
* @param isText
*/
public void setRightBtnInfo(Object drawable, int size, int color, boolean isText) ;

/**
* 设置右按钮的资源
* @param drawable
*/
public void setRightButtonIcon(Drawable drawable);

public void setRightButtonIcon(String unicode);

public void setRightButtonText(String text);

/**
* 设置右按钮的颜色、大小
* @param size
* @param color
* @param isText
*/
public void setRightButtonFont(int size, int color, boolean isText);

/**
* 进度条开始旋转
*/
public void startProgressBar();

/**
* 进度条停止并消失
*/
public void stopProgressBar() ;

/***
* 滑动渐变默认处理，totalHeight 使用默认高度
* @param currentHeight 当前高度
*/
public void handleScrollChange(int currentHeight);
```

```
/***
* 滑动渐变默认处理
*
* @param totalHeight 总高度
* @param currentHeight 当前高度
*/
public void handleScrollChange(int totalHeight, int currentHeight) ;

/**
* 设置使用透明底的图案颜色（白色）
*/
public void setColorWhiteStyle();

/**
* 设置使用白底的图案颜色
*/
public void setColorOriginalStyle();

/**
* 设置返回按钮消失
*/
public void setBackButtonGone();


/**
* 添加搜索框（不可进行输入），仅视觉调整
* @param search
*/
public void setTitle2Search(String search);

/**
* 搜索框转换为标题
*/
public void setSearch2Title();

/**
* 搜索框白底黑字
*/
public void setSearchColorOriginalStyle();

/**
* 搜索框透明底白字
*/
public void setSearchColorTransStyle();

/**
* 左边图标添加红点
* @param flagView
*/
public void attachFlagToLeftBtn(AUWidgetMsgFlag flagView);

/**
* 右边图标添加红点
* @param flagView
```

```
*/
public void attachFlagToRightBtn(AUWidgetMsgFlag flagView) ;

/**
* targetView 添加红点
* @param targetView
* @param flagView
*/
public void attachFlagView(AURelativeLayout container, View targetView, AUWidgetMsgFlag flagView;
```

自定义属性

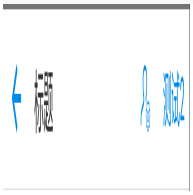
属性名	说明	类型
backgroundDrawable	标题栏的整个背景	reference
backIconColor	返回箭头颜色	color，reference
titleText	标题文案	string，reference
titleTextSize	标题的字体大小	dimension，reference
titleTextColor	标题的字体颜色	color，reference
leftIconResid	左图标的 PNG 或者 JPG ID	reference
leftIconUnicode	左图标的 Unicode	string，reference
leftIconColor	左图标的颜色	color，reference
leftIconSize	左图标的大小	dimension，reference
leftText	左文本文案	string，reference
leftTextColor	左文本的颜色	color，reference
leftTextSize	左文本的大小	dimension，reference
rightIconResid	右图标的 PNG 或者 JPG ID	reference
rightIconUnicode	右图标的 Unicode	string，reference
rightIconColor	右图标的颜色	color，reference
rightIconSize	右图标的大小	dimension，reference
rightText	右文本文案	string，reference
rightTextColor	右文本的颜色	color，reference
rightTextSize	右文本的大小	dimension，reference

代码示例

基础使用

```
<com.alipay.mobile.antui.basic.AUTitleBar
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
app:titleText="标题"
```

```
app:titleTextSize="@dimen/AU_TEXTSIZE2"
app:titleTextColor="#f64219"
app:leftIconUnicode="@string/iconfont_user_setting"
app:rightText="测试2"/>
```



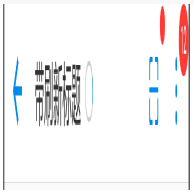
滚动透明

```
<com.alipay.mobile.antui.basic.AUTitleBar
android:id="@+id/title_bar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
app:rightIconUnicode="@string/iconfont_user_setting"
app:titleText="透明标题测试"/>


titleBar.handleScrollChange(testImg.getMeasuredHeight(), 0);
testScroll.setScrollViewListener(new AUScrollViewListener() {
@Override
public void onScrollChanged(ScrollView scrollView, int x, int y, int oldx, int oldy) {
titleBar.handleScrollChange(y);
}
});
```

红点标题

```
<com.alipay.mobile.antui.basic.AUTitleBar
android:id="@+id/progress_title"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
app:titleText="带刷新标题"
app:leftIconUnicode="@string/iconfont_scan"
app:rightIconUnicode="@string/iconfont_more"/>


AUTitleBar processBar = (AUTitleBar) findViewById(R.id.progress_title);

processBar.startProgressBar();
```

```
WidgetMsgFlag j = new WidgetMsgFlag(this);
j.showMsgFlag();
processBar.attachFlagToLeftBtn(j);

WidgetMsgFlag i = new WidgetMsgFlag(this);
i.showMsgFlag(12);
processBar.attachFlagToRightBtn(i);
```



4.8 其他组件

4.8.1 索引组件

AUBladeView 索引组件配合 ListView 使用，ListView 按照字母分类。在页面左侧或右侧的字母索引上，点击或者滑动到相应的字母，触发相应字母位置的事件。默认索引为字母 A - Z，顶部支持自定 1 或 2 个自定义的单个字符。

效果图

如下图所示，A 最上方的两个字符是自定义的，默认字符是 A - Z。

依赖

参见 快速开始 。

接口说明

```
/**
* 设置字母选中监听
*/
public void setOnItemClickListener(OnItemClickListener listener)

public interface OnItemClickListener {

/**
* 设置字母选中监听
* @param clickChar 点击或者选中的字母
*/
void onItemClick(String clickChar);

/**
```

```
* 手指抬起的事件，无特殊需求，无需关注此方法
*/
void onClickUp();
}
```

自定义属性

属性名	说明	类型
top1Text	自定义第一个文本字符	reference
top2Text	自定义第二个文本字符	reference
showSelectPop	是否显示滑动或者点击过程中中间弹出的浮层	boolean

代码示例

```
<com.alipay.mobile.antui.basic.AUBladeView
android:layout_width="24dp"
android:layout_height="wrap_content"
app:top1Text="⊙"
app:top2Text="才"/>
```

如 效果图 所示，top1Text、top2Text 默认都可以省略。

4.8.2 按钮组件

AUButton 组件用于提供拥有不同样式的按钮。

效果图

依赖

参见 快速开始 。

Style 接口

属性名	说明
mainButtonStyle	页面主按钮
subButtonStyle	页面次按钮
warnButtonStyle	警告按钮
assMainButtonStyle	辅助主按钮
assButtonStyle	辅助次按钮
listButtonStyle	列表按钮

代码示例

- **页面主按钮**

```
<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/mainButtonStyle"
android:layout_margin="12dp"
android:clickable="true"
android:text="页面主按钮 Normal"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/mainButtonStyle"
android:layout_margin="12dp"
android:enabled="false"
android:text="页面主按钮 Disable"
app:dynamicThemeDisable="true"/>
```

- **页面次按钮**

```
<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/subButtonStyle"
android:layout_margin="12dp"
android:clickable="true"
android:text="页面次要操作 Normal"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/subButtonStyle"
android:layout_margin="12dp"
android:enabled="false"/>
```

- **警告按钮**

```
<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/warnButtonStyle"
android:layout_margin="12dp"
android:clickable="true"
android:text="警告按钮 Normal"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/warnButtonStyle"
android:layout_margin="12dp"
android:enabled="false"
```

```
android:text="警告按钮 Disable"/>
```

- **辅助主按钮**

```
<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/assMainButtonStyle"
android:layout_margin="12dp"
android:clickable="true"
android:text="下载"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/assMainButtonStyle"
android:layout_margin="12dp"
android:enabled="false"
android:text="下载"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/assMainButtonStyle"
android:layout_margin="12dp"
android:text="辅助主按钮"/>
```

- **辅助次按钮**

```
<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/assButtonStyle"
android:layout_margin="12dp"
android:clickable="true"
android:text="下载"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/assButtonStyle"
android:layout_margin="12dp"
android:enabled="false"
android:text="下载"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/assButtonStyle"
android:layout_margin="12dp"
android:text="辅助按钮"/>
```

- **列表按钮**

```
<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/listButtonStyle"
android:layout_marginTop="12dp"
android:layout_marginBottom="12dp"
android:clickable="true"
android:text="取消关注"/>

<com.alipay.mobile.antui.basic.AUButton
style="@com.alipay.mobile.antui:style/listButtonStyle"
android:layout_marginTop="12dp"
android:layout_marginBottom="12dp"
android:clickable="true"
android:textColor="@com.alipay.mobile.antui:color/AU_COLOR_LINK"
android:text="更多服务"/>
```

4.8.3 操作条组件

AUCardOptionView 操作条组件用于实现点赞、评论、打赏，是一个组合的 View，继承 AULinearLayout，支持 XML 布局接入。

效果图

依赖

参见 快速开始 。

接口说明

```
/**
* 设置整个 view 的信息
* @param itemArrayList
* @param textVisible
*/
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, boolean textVisible)

/**
* 设置整个 view 的信息
* @param itemArrayList
* @param textType = CardOptionView.TEXT_NOT_CHANGE 则一直显示文字,不改变为数字
*/
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, String textType)
```

```
/**
* 设置整个 view 的信息
* @param itemArrayList
*/
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList)

/**
* 设置整个 view 的信息
* @param itemArrayList
* @param height
* @param textVisible
*/
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, int height, boolean textVisible)

/**
* 设置整个 view 的信息
* @param itemArrayList
* @param height
*/
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, int height)

/**
* 子 view 计数递增
* @param childView
*/
public void unitIncrease(View childView)

/**
* 子 view 计数递减
* @param childView
*/
public void unitDecrease(View childView)

/**
* 获取计数
* @param position
* @return
*/
public int getCount(int position)

/**
* 返回类型 View
* @param type
* @return
*/
public View getChildView(String type)

/**
* 设置监听
* @param cardOptionListner
*/
public void setCardOptionListner(CardOptionClickListner cardOptionListner) {
this.mListner = cardOptionListner;
}
```

自定义属性

普通的 ViewGroup，无新增自定义属性。

代码示例

```
AUCardOptionView.CardOptionItem optionItem1 = new AUCardOptionView.CardOptionItem();
optionItem1.type = AUCardOptionView.TYPE_PRAISE;
optionItem1.hasClicked = false;


AUCardOptionView.CardOptionItem optionItem2 = new AUCardOptionView.CardOptionItem();
optionItem2.type = AUCardOptionView.TYPE_REWARD;
optionItem2.hasClicked = false;


AUCardOptionView.CardOptionItem optionItem3 = new AUCardOptionView.CardOptionItem();
optionItem3.type = AUCardOptionView.TYPE_COMMENT;
optionItem3.hasClicked = false;


ArrayList<AUCardOptionView.CardOptionItem> optionItems = new
ArrayList<AUCardOptionView.CardOptionItem>();
optionItems.add(optionItem1);
optionItems.add(optionItem2);
optionItems.add(optionItem3);
mAUCardOptionView.setViewInfo(optionItems,AUCardOptionView.TEXT_NOT_CHANGE);
mAUCardOptionView.setCardOptionListner(new AUCardOptionView.CardOptionClickListner() {
@Override
public void onCardOptionClick(View v, AUCardOptionView.CardOptionItem optionItem, int position) {
mAUCardOptionView.unitIncrease(v);
}
});
```

4.8.4 勾选组件

AUCheckIcon 组件用于实现选择框的 IconView。

效果图

依赖

参见 快速开始 。

接口说明

```
/**选中状态*/
```

```
public static final int STATE_CHECKED = 0x01;
/**未选中状态*/
public static final int STATE_UNCHECKED = 0x02;
/**不可取消勾选状态*/
public static final int STATE_CANNOT_UNCHECKED = 0x03;
/**不可勾选状态*/
public static final int STATE_CANNOT_CHECKED = 0x04;

/**
* 设置 checkIcon 的状态
* @param state
*/
public void setIconState(int state);

/**
* 获取checkIcon的状态
* @return
*/
public int getIconState() ;
```

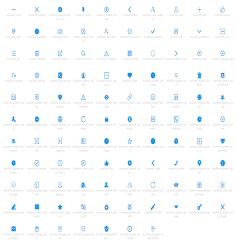
4.8.5 图标组件

AUIconView 为 iconfont 矢量图控件，可以同时实现 TextView 及 ImageView 的功能。

iconfont 图片控件（可当做 TextView 来使用）实际是通过 TextView 的 TTF 字体文件，定义特殊的 Unicode 码对应一类图片字体。也就是说，iconfont 相当于是加载了一个字体，一个字体对应了多张图片，每个图片有一个 Unicode 码。

每个 iconfont 集合实际就是一个 TTF 字体文件，因此可以加载多个 TTF 字体文件。每个 TTF 字体文件有一个名称，默认 AntUI 的 TTF 字体文件名称为 auiconfont。

效果图



依赖

参见 快速开始 。

图标资源

资源 ID	对应的示例名称
com.alipay.mobile.antui.R.string.iconfont_more	更多
com.alipay.mobile.antui.R.string.iconfont_cancel	取消
com.alipay.mobile.antui.R.string.iconfont_voice	语音
com.alipay.mobile.antui.R.string.iconfont_collect_money	收款

com.alipay.mobile.antui.R.string.iconfont_back	返回
com.alipay.mobile.antui.R.string.iconfont_user_setting	用户设置
com.alipay.mobile.antui.R.string.iconfont_user	用户
com.alipay.mobile.antui.R.string.iconfont_add	添加
com.alipay.mobile.antui.R.string.iconfont_praise	点赞
com.alipay.mobile.antui.R.string.iconfont_map	地图
com.alipay.mobile.antui.R.string.iconfont_checked	勾选
com.alipay.mobile.antui.R.string.iconfont_notice	公告
com.alipay.mobile.antui.R.string.iconfont_add_user	添加用户
com.alipay.mobile.antui.R.string.iconfont_comment	评论
com.alipay.mobile.antui.R.string.iconfont_selected	选择
com.alipay.mobile.antui.R.string.iconfont_bill	账单
com.alipay.mobile.antui.R.string.iconfont_pulldown	下拉
com.alipay.mobile.antui.R.string.iconfont_scan	扫描
com.alipay.mobile.antui.R.string.iconfont_list	列表
com.alipay.mobile.antui.R.string.iconfont_delete	删除
com.alipay.mobile.antui.R.string.iconfont_share	分享
com.alipay.mobile.antui.R.string.iconfont_search	搜索
com.alipay.mobile.antui.R.string.iconfont_complain	投诉
com.alipay.mobile.antui.R.string.iconfont_qrcode	二维码
com.alipay.mobile.antui.R.string.iconfont_unchecked	取消勾选
com.alipay.mobile.antui.R.string.iconfont_right_arrow	右箭头
com.alipay.mobile.antui.R.string.iconfont_help	帮助
com.alipay.mobile.antui.R.string.iconfont_group_chat	群聊
com.alipay.mobile.antui.R.string.iconfont_contacts	联系人
com.alipay.mobile.antui.R.string.iconfont_setting	设置
com.alipay.mobile.antui.R.string.iconfont_phone_book	通讯录
com.alipay.mobile.antui.R.string.iconfont_phone_contact	手机联系人

接口说明

```
/**
* 设置图片资源 ID
* @param resId
* @return
*/
@Override
public AUIconView setImageResource(int resId) {
if (resId == 0) {
```

```
return this;
}
clearView();
initImageView();
imageView.setImageResource(resId);
this.addView(imageView);
return this;
}

/**
* 设置图片资源 drawable
* @param drawable
* @return
*/
@Override
public IconfontInterface setImageDrawable(Drawable drawable)

/**
* 设置 iconfont 颜色
* @param color
* @return
*/
public AUIconView setIconfontColor(int color)

/**
* 设置 iconfont 颜色 ColorStateList
* @param color
* @return
*/
public AUIconView setIconfontColorStates(ColorStateList color)


/**
* 设置 view 的大小，单位 px
*
* @param size
*/
public AUIconView setIconfontSize(float size)


/**
* 设置 view 的 iconfont 资源或文本
* @param text
* @return
*/
@Override
public AUIconView setIconfontUnicode(String text)
```

代码示例

- 设置图标的信息：

```
AUIconView iconView = (AUIconView) convertView.findViewById(R.id.icon_view);
iconView.setIconfontUnicode(iconUnicode);
```

```
//例如
//iconView.setIconfontUnicode(getResources().getString(com.alipay.mobile.antui.R.string.iconfont_phone_contact));
```

- 设置图标的颜色：

```
<com.alipay.mobile.antui.iconfont.AUIconView
android:id="@+id/icon_view"
android:layout_width="@dimen/size"
android:layout_height="@dimen/size"
app:iconfontColor="@com.alipay.mobile.antui:color/AU_COLOR_APP_GREEN"
app:iconfontUnicode="@com.alipay.mobile.antui:string/iconfont_back"/>

//or:
iconView.setIconfontColor(color)
iconView.setIconfontColorStates(colorStateList)
```

4.8.6 刷新组件

AURefreshListView 刷新组件是包含下拉刷新及上拉加载的 ListView。

依赖

参见 快速开始 。

接口说明

```
/**
* 下拉刷新的状态监听
*
* @param onPullRefreshListener
*/
public void setOnPullRefreshListener(OnPullRefreshListener onPullRefreshListener)

/**
* 加载更多的状态监听
*
* @param onLoadMoreListener
*/
public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener)

/**
* 代码开启下拉刷新
*/
public void startRefresh()

/**
* 下拉刷新结束
*/
public void finishRefresh()

/**
```

```
* 底部加载更多状态更新
*
* @param isShowLoad
* @param hasMore
*/
public void updateLoadMore(boolean isShowLoad, boolean hasMore)
```

代码示例

```
<com.alipay.mobile.antui.load.AURefreshListView
android:id="@+id/refresh_list_view"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>


listView.setOnPullRefreshListener(new OnPullRefreshListener() {
@Override
public void onRefresh() {
listView.finishRefresh();
listView.updateLoadMore(true, true);
}

@Override
public void onRefreshFinished() {

}
});
listView.setOnLoadMoreListener(new OnLoadMoreListener() {
@Override
public void onLoadMore() {
for (int i = 0; i < 3; i++) {
Map<String, Object> map = new HashMap<String, Object>();
map.put("PIC","下接加载更多List");
map.put("TITLE","上拉加载更多");
contents.add(map);
}
adapter.notifyDataSetChanged();
if(contents.size() > 13) {
listView.updateLoadMore(true, false);
} else {
listView.updateLoadMore(true, true);
}
}

@Override
public void onLoadingFinished() {

}
});
```

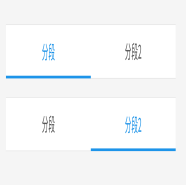
4.8.7 切换栏组件

AUSegment 用来替换 APSwitchTab 控件，对相关代码做了重构，并保留原有接口，可以平滑滚动。

10.0.20 版本之后，该组件支持可滚动 Tab 切换。每个 Tab 左右间距为 14dp：

- 当所有 Tab 超过初始设置宽度时，支持滚动切换。
- 当所有 Tab 小于初始宽度时，提供是否等分接口的选项，默认选择为等分。

效果图



依赖

参见 快速开始 。

接口说明

```
/**
* 重置 tab 视图
*/
public void resetTabView(String[] tabNameArray)

/**
* 调整底部选中线条的位置，一般在 viewPager 的 onPageScrolled 回调中，调用该方法。
*
* @param position 起始位置
* @param positionOffset 起始位置的偏移量（百分比）
*/
public void adjustLinePosition(int position, float positionOffset)

/**
* 选中 tab 但不调整底部线条位置， 适用于 viewpager 的 tab 切换，
* 底部选中线条通过在 viewpager 的 onPageScrolled 回调中调用 adjustLinePosition 来实现
*
* @param position 位置
*/
public void selectTab(int position)

/**
* 选中 tab 并调整底部线条的位置，<br>
* 用于非 viewpager 的 tab 切换场景，每个 tab 间隔之前的过场动画时间为 250ms
*
* @param position 目标选中的 tab
*/
public void selectTabAndAdjustLine(int position)

/**
* 选中 tab 并调整底部线条的位置，<br>
* 用于非 viewpager 的 tab 切换场景，每个 tab 间隔之前的过场动画时间由自己指定<br>
* 若上一个动画还未放完，启动下一个动画时，上一个动画立即结束，定位到上一个动画的最终位置后启动下一个动画。
*
```

```
* @param position 目标位置
* @param during 每个 tab 间隔之前的过场动画时间
*/
public void selectTabAndAdjustLine(int position, int during)

/**
* 设置 tab 切换监听器
*
* @param tabSwitchListener
*/
public void setTabSwitchListener(TabSwitchListener tabSwitchListener)

/**
* 加指定的位置加上红点
* @param view 红点 view
* @param position
*/
public void addTextRightView(View view, int position)

/**
* 加指定的位置加上红点
* @param view 红点
* @param params 红点的相对位置
* @param position
*/
public void addTextRightView(View view, RelativeLayout.LayoutParams params, int position)
```

Tab 滚动切换的接口说明

若要用滚动功能，需使用自定义属性参数 scroll 并将该参数设置为 true，如在布局文件中添加 app:scroll="true" 。可滚动 Tab 只支持以下四个接口，其他接口在可滚动 Tab 中无效。

```
/**
* 设置 Tab 切换监听器
* @param tabSwitchListener
*/
public void setTabSwitchListener(TabSwitchListener tabSwitchListener)

/**
* 设置数据源
* @param list
*/
public void init(List<ItemCategory> list)
/**
* 设置选中的 Tab
* @param position
*/
public void setCurrentSelTab(int position)

/**
* 每个 Tab 固定左右间距为 14dp，当所有 Tab 不足初始宽度时提供是否等分接口
* 默认等分，设置 false 禁止等分
* @param divideAutoSize
*/
```

```
public void setDivideAutoSize(boolean divideAutoSize)
```

自定义属性

10.0.20 及以后版本添加了 scroll 属性。

属性名	用途	类型
tabCount	Tab 数量	integer
tab1Text	Tab1 文案	string，reference
tab2Text	Tab1 文案	string，reference
tab3Text	Tab3 文案	string，reference
tab4Text	Tab4 文案	string，reference
tabTextArray	Tab 文本数组	string，reference
uniformlySpaced	是否自适应	boolean
tabTextColor	文字颜色	reference，color
tabTextSize	文字大小	dimension
buttomLineColor	底部线条颜色	color，reference
scroll	是否支持滚动	boolean

代码示例

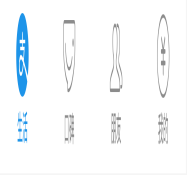
XML：

```
<com.alipay.mobile.antui.segement.AUSegment
android:id="@+id/switchtab_three"
android:layout_width="fill_parent"
android:layout_height="50dp"
android:layout_marginTop="10dp"
app:tab1Text="左边文字"
app:tab2Text="中间文字"
app:tab3Text="右边文字"
app:tabCount="3"/>
```

4.8.8 标签组件

AUTabBarItem 标签组件用于提供 mPaaS 框架 TabBar 里面的每一项。

效果图



依赖

参见 快速开始 。

自定义属性

属性名	说明	类型
topIconSid	图标	reference
topIconSize	图标大小	dimension
textColor	文字颜色	color，reference

代码示例

XML：

```
<com.alipay.mobile.antui.bar.AUTabBarItem
android:id="@+id/tab_2"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="口碑"
android:textSize="@dimen/AU_ICONSIZE2"
app:topIconSize="@dimen/AU_ICONSIZE2"
app:topIconSid="@drawable/tab_bar_alipay"
app:textColor="@color/tabbar_text_color1"/>
```

5 基于 Native 框架 - iOS 组件库

5.1 快速开始

mPaaS 对外提供统一组件库，满足用户对不同 Native 控件的需求。要实现客户端接入统一组件库，您必须先添加统一组件库的 SDK，然后在代码中调用 SDK 接口方法来添加控件。

前置条件

您已经接入工程到 mPaaS。更多信息，请参见以下内容：

- 基于 mPaaS 框架接入
- 基于已有工程且使用 mPaaS 插件接入
- 基于已有工程且使用 CocoaPods 接入

添加 SDK

根据您采用的接入方式，请选择相应的添加方式。

- 使用 mPaaS Xcode Extension。
  此方式适用于采用了 **基于 mPaaS 框架接入** 或 **基于已有工程且使用 mPaaS 插件接入** 的接入方式。

    - 点击 Xcode 菜单项 **Editor** > **mPaaS** > **编辑工程**，打开编辑工程页面。
    - 选择 **通用 UI**，保存后点击 **开始编辑**，即可完成添加。

- 使用 cocoapods-mPaaS 插件。此方式适用于采用了 **基于已有工程且使用 CocoaPods 接入** 的接入方式。
    - 在 Podfile 文件中，使用 mPaaS_pod"mPaaS_CommonUI" 添加通用 UI 组件依赖。
    - 执行 pod install 即可完成接入。

使用 SDK

请结合 通用 UI 官方 Demo 在 10.1.60 及以上版本的基线中使用通用 UI SDK。

5.2 基本组件

5.2.1 活动指示器基本类

- 活动指示器基本类 AUActivityIndicatorView 为 UIActivityIndicatorView 在 mPaaS 中的版本 。
- 为方便后续扩展，所有 mPaaS 应用都必须使用 AUActivityIndicatorView，而不是系统的 UIActivityIndicatorView。
- 由于目前 AUActivityIndicatorView 完全继承自 UIActivityIndicatorView，并未额外添加属性和方法，故此处不再对接口说明、代码示例等内容进行描述。

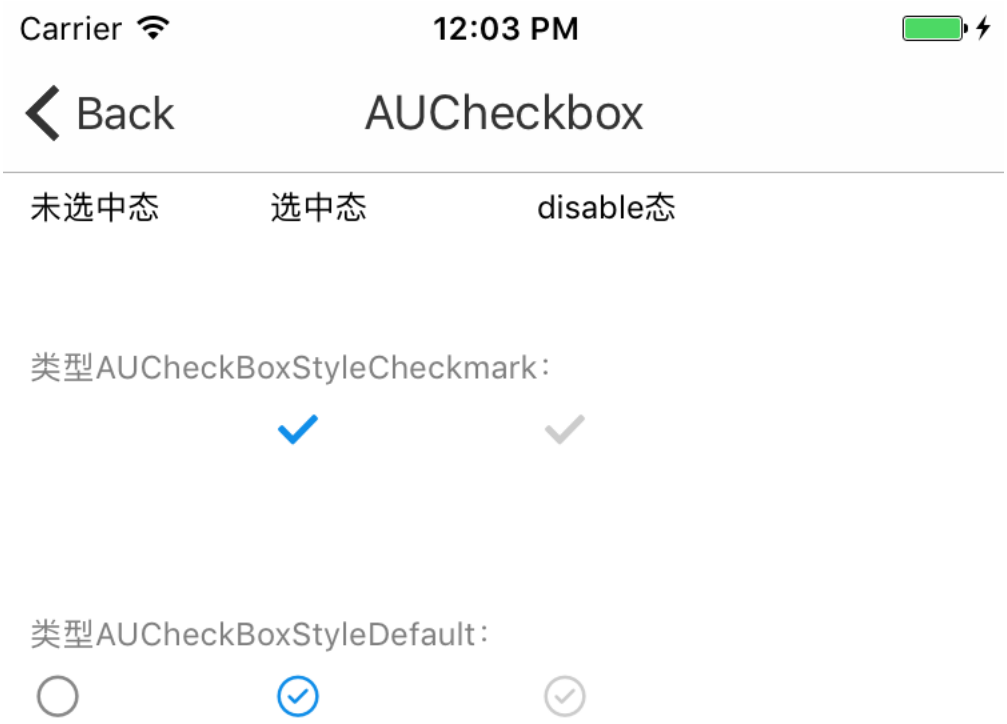
5.2.2 开关基本类

- 开关基本类 AUSwitch 为 UISwitch 在 mPaaS 中的版本 。
- 为方便后续扩展，所有 mPaaS 应用都必须使用 AUSwitch，而不是系统的 UISwitch。
- 由于目前开关基本类完全继承自 UISwitch，并未额外添加属性和方法，故此处不再对接口说明、代码示例等内容进行描述。

5.2.3 单选框控件

- AUCheckBox 为单选框控件 。
- AUCheckBox 迁移自 APCommonUI 的 APCheckbox，请使用最新的 AUCheckBox。

效果图



接口说明

```
/**
checkbox 类型

- AUCheckBoxStyleDefault: 默认样式，与 web 的 checkbox 类似
- AUCheckBoxStyleCheckmark: tableview 的 checkmark 样式
*/
typedef NS_ENUM(NSInteger, AUCheckBoxStyle) {
AUCheckBoxStyleDefault,
AUCheckBoxStyleCheckmark
};


/**
单选框控件
*/
@interface AUCheckBox : UIControl

/**
根据类型初始化 AUCheckBox 方法

@param style checkbox 类型

@return AUCheckBox
*/
- (instancetype)initWithStyle:(AUCheckBoxStyle)style;

/**
```

```
是否选中属性
*/
@property(nonatomic, assign, getter = isChecked) BOOL checked;

/**
是否 disabled 属性
*/
@property(nonatomic, assign, getter = isDisabled) BOOL disabled;

/**
checkbox 类型（只读，只能在初始化时设置）
*/
@property (nonatomic, assign, readonly) AUCheckBoxStyle style;


@end
```

代码示例

```
AUCheckBox *checkbox = [[AUCheckBox alloc] initWithStyle:AUCheckBoxStyleDefault];
checkbox.checked = YES;
checkbox.disabled = NO;
checkbox.origin = CGPointMake(100, 250);
[checkbox addTarget:self action:@selector(checkboxValueChanged:)
forControlEvents:UIControlEventValueChanged];
[self.view addSubview:checkbox];

- (void)checkboxValueChanged:(id)sender
{
AUCheckBox *checkbox = (AUCheckBox *)sender;
NSLog(@"%@", checkbox);
}
```

5.2.4 图像基本类

- 图像基本类 AUImage 为 UIImage 在 mPaaS 中的版本 。
- 为方便后续扩展，所有 mPaaS 应用都必须使用 AUImage，而不是系统的 UIImage。
- 由于目前图像基本类完全继承自 UIImage，并未额外添加属性和方法，故此处不再对接口说明、代码示例等内容进行描述。

5.2.5 标签基本类

- 标签基本类 AULabel 为 UILabel 在 mPaaS 中的版本，继承自 UILabel。
- 为方便后续扩展，所有 mPaaS 应用都必须使用 AULabel，而不是系统的 UILabel。
- 由于标签基本类目前完全继承自 UILabel，并未额外添加属性和方法，故此处不再对接口说明、代码示例等内容进行描述。

注意：对于复杂的 Label 设置需求，可以使用 TTTAttributedLabel（已经在 AntUI 中引入）。

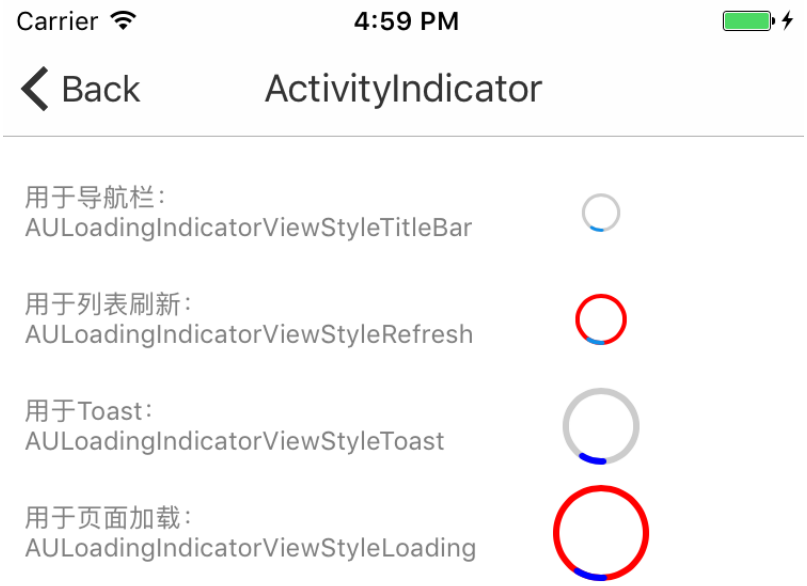
5.2.6 页脚基本类

文档解析出错,请参照金融科技上面的文档，文档编号为(71660),也请把括号中的文档编号发给我们查找问题,谢谢!

5.2.7 mPaaS 自定义加载控件

- AULoadingIndicatorView 为 mPaaS 自定义的加载控件。
- mPaaS 自定义加载控件迁移自 APCommonUI 的 APActivityIndicatorView，请使用最新的 AULoadingIndicatorView。

效果图



接口说明

```
typedef enum{
AULoadingIndicatorViewStyleTitleBar, //导航栏加载，直径：36px，环宽：3px
AULoadingIndicatorViewStyleRefresh, //列表刷新加载，直径：48px，环宽：4px
AULoadingIndicatorViewStyleToast, //toast 加载，直径：72px，环宽：6px
AULoadingIndicatorViewStyleLoading, //页面加载，直径：90px，环宽：6px
}AULoadingIndicatorViewStyle;



/**
mPaaS 自定义加载控件
*/
@interface AULoadingIndicatorView : UIView

@property (nonatomic, assign) BOOL hidesWhenStopped; //是否停止的时候隐藏掉
```

```
@property (nonatomic, strong) UIColor *trackColor; //圆环颜色
@property (nonatomic, strong) UIColor *progressColor; //指示器颜色
@property (nonatomic, assign) float progressWidth; //设置圆环的宽度，自定义圆圈大小时，默认为 2
@property (nonatomic, assign) CGFloat progress; //加载指示器的弧长与圆环的比值，默认为 0.1

/**
* 圆圈样式的 loading 框
* 说明：如果不使用默认 style，需要自定义圆圈的大小，请使用 initWithFrame: 初始化，此时圆环宽度默认为 2，可设置 progressWidth 调整
*
* @param style 当前 loading 类型。
*
*/
- (instancetype)initWithLoadingIndicatorStyle:(AULoadingIndicatorViewStyle)style;

/**
开始执行动画
*/
- (void)startAnimating;

/**
停止执行动画
*/
- (void)stopAnimating;

/**
是否正在执行动画

@return YES：动画执行中；NO：没有执行动画
*/
- (BOOL)isAnimating;


@end
```

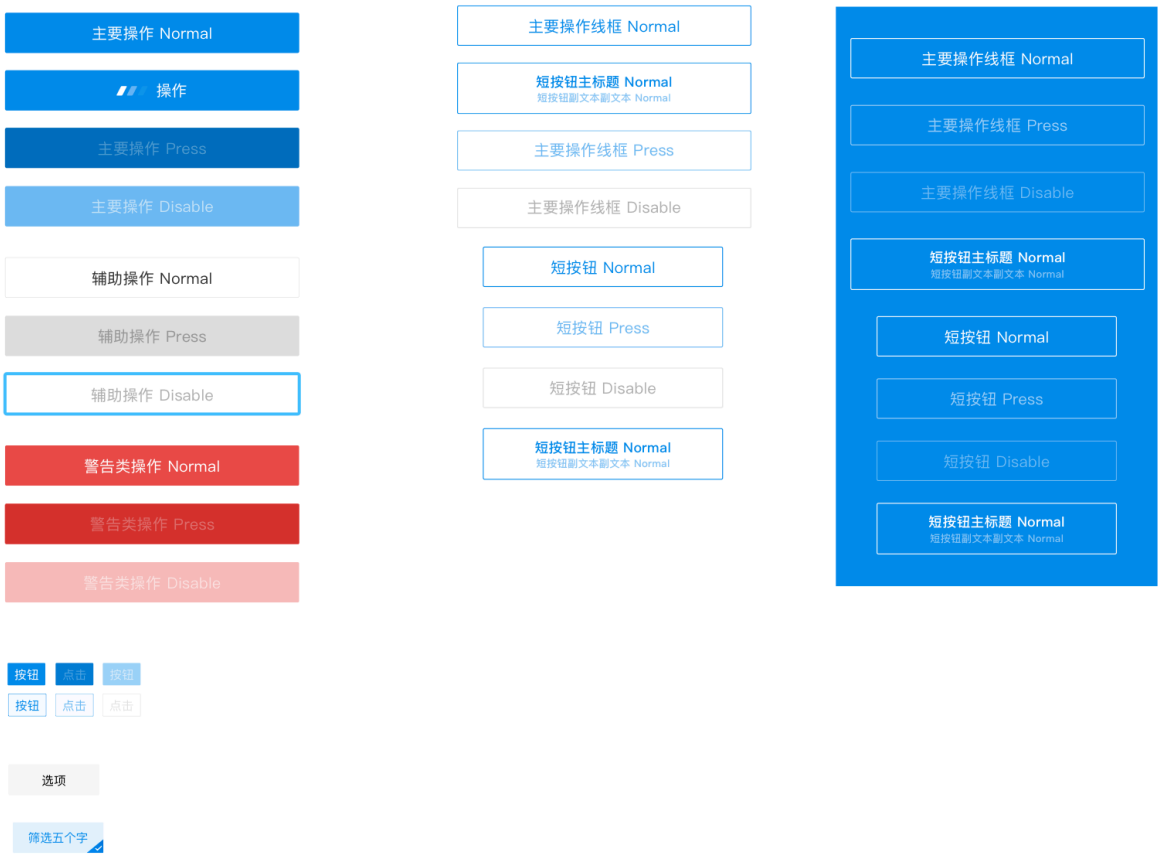
代码示例

```
AULoadingIndicatorView *view = [[AULoadingIndicatorView alloc]
initWithLoadingIndicatorStyle:AULoadingIndicatorViewStyleLoading];
view.hidesWhenStopped = YES;
view.center = CGPointMake(280, 250);
view.trackColor = [UIColor redColor];
view.progressColor = [UIColor blueColor];
[view startAnimating];
[self.view addSubview:view];
```

5.2.8 按钮基本类

- AUButton 遵照新的 UED 需求完成，目前包含两种样式，与 APCommonUI 中的 APButton 不能完全互通。
- 这两种样式不包括效果图中的警告类操作按钮。

效果图



依赖

AUButton 的依赖如下：

```
import <UIKit/UIKit.h>
```

接口说明

```
/**
初始化方法
@param style 样式
@return 创建的初始化对象
*/
+ (instancetype)buttonWithStyle:(AUButtonStyle)style;

/**
* 初始化的辅助方法，用于创建并初始化一个按钮的对象。
*
* @param buttonType 按钮类型，必须是定义在 AUButtonStyle 中的其中一个值。
* @param title 铵钮标题
* @param target 响应按钮点击事件的对象
* @param action 响应按钮点击事件的函数
*
```

```
* @return 新创建并经过初始化的按钮对象。
*
* 此方法初始化的对象 需要设置frame
*/
+ (instancetype)buttonWithStyle:(AUButtonStyle)style title:(NSString *)title target:(id)target action:(SEL)action;

/**
在按钮上展示菊花动画和文字，左菊花右文字，无文字时菊花居中

@param loadingTitle 展示菊花时候的文字，设置 nil 或者空串不展示，菊花居中
@param currentVC 当前 VC，为了 loading 结束的去掉遮罩
*/
- (void)startLoadingWithTitle:(NSString *)loadingTitle currentViewController:(UIViewController *)currentVC;


/**
停止转菊花
*/
- (void)stopLoading;
```

自定义属性

属性名	用途
AUButtonStyleNone	系统默认
AUButtonStyle1	蓝底，白字，无边框，大按钮样式
AUButtonStyle2	白底，黑字，浅灰色边框，大按钮样式
AUButtonStyle3	透明底，蓝字，蓝色边框，小按钮字样
AUButtonStyle4	白底，默认带上下分割线，字是红色的；使用场景（取消关注）等页面底部操作，默认高度 44 单位，宽度为屏幕宽度
AUButtonStyle5	白底，默认带上下分割线，字是蚂蚁蓝；使用场景（更多服务）等页面底部操作，默认高度 44 单位，宽度为屏幕宽度
AUButtonStyle6	红底白字，警告类操作，大按钮样式
AUButtonStyle7	白底，黑字，浅灰色边框，小按钮样式
AUButtonStyle8	蓝底，白字，无边框，小按钮样式

代码示例

```
AUButton *button = [AUButton buttonWithStyle:AUButtonStyle2 title:@"AUButtonStyle2"target:self
action:@selector(onButtonClicked:)];
button.frame = CGRectMake(XX, XX,XX, XX);

AUButton *buttonDisable = [AUButton buttonWithStyle:AUButtonStyle1];
buttonDisable.enabled = NO;
[buttonDisable setTitle:@"Style1disable"forState:UIControlStateNormal];
buttonDisable.frame = CGRectMake(XX, XX,XX, XX);

// button 上需要旋转菊花
[button startLoadingWithTitle:@"Loading"currentViewController:self];
```

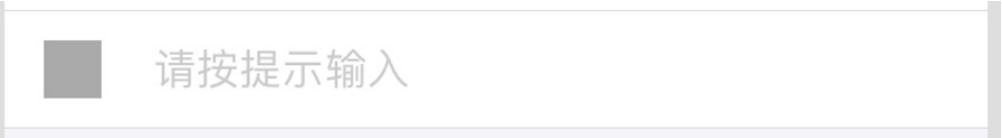
```
// button 菊花停止
[button stopLoading];
```

5.3 输入组件

5.3.1 带图输入框

AUImageInputBox 为左侧带图标的输入框，继承自 AUInputBox 。

效果图



接口说明

```
/**
左侧为图标的输入框样式
*/
@interface AUImageInputBox : AUInputBox

/**
左侧图标视图（只读）
*/
@property (nonatomic, strong, readonly) UIImageView *iconView;

/**
设置左侧图标图片

@param image 图标图片
*/
- (void)setIconImage:(UIImage *)image;
```

代码示例

```
AUImageInputBox *imageInputBox = [AUImageInputBox inputboxWithOriginY:startY
inputboxType:AUInputBoxTypeNone];
imageInputBox.textField.placeholder = @"请按提示输入";
[imageInputBox setIconImage:image];
[self.view addSubview:imageInputBox];
```

5.3.2 段落输入框

AUParagraphInputBox 为段落输入框控件，支持在业务中设置最大字数限制。

效果图

接口说明

```
// 段落输入框

@interface AUParagraphInputBox : UIView

@property (nonatomic, strong) UITextView *textView; // 输入框
@property (nonatomic, assign) NSInteger maxInputLen; // 设置最大输入字数（需要才设置）

// 初始化
- (instancetype)initWithFrame:(CGRect)frame placeHolder:(NSString *)placeHolder;

// 设置 placeHolder 文本
- (void)setPlaceHolder:(NSString *)placeHolder;

@end
```

代码示例

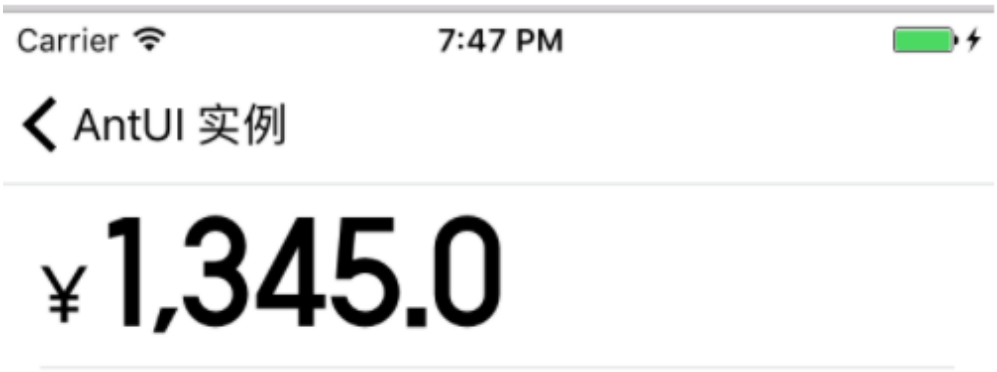
```
_paragraphInputBox = [[AUParagraphInputBox alloc] init];
_paragraphInputBox.frame = CGRectMake(0, startY, AUCommonUIGetScreenWidth(), 10);
_paragraphInputBox.maxInputLen = 1240;
_paragraphInputBox.textView.delegate = self;
[_paragraphInputBox setPlaceHolder:@"请输入你想表达的内容"];
[self.view addSubview:_paragraphInputBox];
```

5.3.3 简版金额输入框

AUAmountEditTextField

- 简版金额输入框 AUAmountEditTextField 可与金额显示组件 AUAmountLabelText 配套使用。
- 目前不含输入内容的校验与预处理逻辑，这些功能在业务中可通过设置 delegate 自行实现。

效果图



接口说明

```
NS_ASSUME_NONNULL_BEGIN

@interface AUAmountEditTextField : UITextField

@end


/**
带“¥”符号和下划线的简版金额输入组件。
输入内容字号大小会随内容长度缩放
*/
@interface AUAmountEditText : UIView


/**
金额输入框，可按需修改属性或设置 delegate.
clear 事件发生时，会调用 [amountTextField sendActionsForControlEvents:UIControlEventEditingChanged]
*/
@property(nonatomic,strong) AUAmountEditTextField *amountTextField;


/**
开放给 AUAmountLabelText，用于 inputText 长度变化时调整字号使用。
业务方请勿使用。

@param textLength inputText 长度
@return UIFont
*/
+ (UIFont *)resetFontSize:(NSUInteger) textLength;

@end
```

```
NS_ASSUME_NONNULL_END

// amountTextField 初始化设置:
_amountTextField.textColor = RGB(0x000000);
_amountTextField.backgroundColor = [UIColor clearColor];
_amountTextField.font = [UIFont fontWithCustomName:kAmountNumberFontName size:45.0];
_amountTextField.contentVerticalAlignment= UIControlContentVerticalAlignmentCenter;
_amountTextField.inputView = [AUNumKeyboards sharedKeyboardWithMode:AUNumKeyboardModeCommon];
_amountTextField.rightViewMode = UITextFieldViewModeWhileEditing;
_amountTextField.rightView = self.rightView;//clearButton 使用 rightView 实现
```

代码示例

```
field = [[AUAmountEditText alloc] init];//默认屏幕等宽，高度 70
field.amountTextField.delegate = self;
[view addSubview:field];
```

AUAmountLabelText

AUAmountLabelText 是与 AUAmountEditTextField 配套使用的金额显示组件。

效果图

接口说明

```
NS_ASSUME_NONNULL_BEGIN

/**
AUAmountEditTextField配套使用的金额显示组件.
*/
@interface AUAmountLabelText : UIView

@property (nonatomic, copy) NSString *amountText;//金额数字，不带半角符号，例如："80.01"

@end

NS_ASSUME_NONNULL_END
```

代码示例

```
label = [[AUAmountLabelText alloc] init];//默认屏幕等宽，高度 64
label.amountText = @"1,345.0";
[view addSubview:label];
```

5.3.4 金额输入框

AUAmountInputBox

- AUAmountInputBox 为带组合功能的金额输入框。
- 目前支持设置 title（纯文本），添加 footer（纯文本/输入框）的功能。
- 不包含输入内容的校验与预处理逻辑，在业务中可通过设置 delegate 自行实现。

效果图

接口说明

```
NS_ASSUME_NONNULL_BEGIN

/**
带组合功能的金额输入框。
目前支持设置 title（纯文本），添加 footer（纯文本/输入框）的功能。
不包含输入内容的校验与预处理逻辑，在业务中可通过设置 delegate 自行实现。
*/
@interface AUAmountInputBox : UIView

/**
AUAmountInputBox 初始化方法

@param views @[AUAmountInputField,AUAmountInputFieldFooterView]
@return AUAmountInputBox
*/
+ (AUAmountInputBox *)amountInputBoxWithViews:(NSArray *) views;

@end

NS_ASSUME_NONNULL_END
```

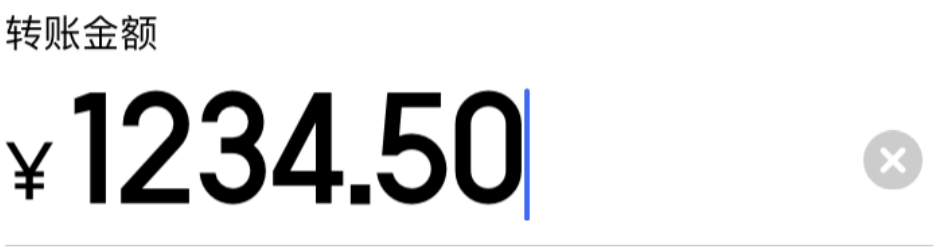
代码示例

```
AUAmountInputField *inputField = [AUAmountInputField amountInputWithTitle:@"转账金额"];
AUAmountInputFieldFooterView *footerView = [AUAmountInputFieldFooterView footerWithInput:@"添加备注(50字以内)"];
AUAmountInputBox *inputBox = [AUAmountInputBox amountInputBoxWithViews:[NSArray arrayWithObjects:inputField,footerView,nil]];
inputField.textField.delegate = self;
footerView.inputTextField.delegate = self;
[_scrollView addSubview:inputBox];
```

AUAmountInputField

基于 AUAmountEditText 的组合扩展，目前支持设置 title。

效果图



接口说明

```
NS_ASSUME_NONNULL_BEGIN


/**
基于 AUAmountEditText 的组合扩展，目前支持设置 title。
*/
@interface AUAmountInputField : UIView

- (AUAmountEditTextField *)textField;

+ (AUAmountInputField *)amountInputWithTitle:(NSString *) title;

@end

NS_ASSUME_NONNULL_END
```

代码示例

见 AUAmountInputBox 的 代码示例 。

AUAmountInputFieldFooterView

说明

AUAmountInputFieldFooterView 是 AUAmountInputBox 的 footerView，目前支持“纯文本”与“输入框”两种类型。

效果图

依赖

AUAmountInputFieldFooterView 的依赖如下：

```
pod 'AntUI'
```

接口说明

```
NS_ASSUME_NONNULL_BEGIN

@interface AUAmountInputFieldFooterView : UIView

@property (nonatomic, strong) UITextField * inputTextField;
@property (nonatomic, strong) UILabel * descTextLabel;

+ (AUAmountInputFieldFooterView *)footerWithInput:(nullable NSString *)placeholder;
+ (AUAmountInputFieldFooterView *)footerWithDesc:(nullable NSString *)text;

@end

NS_ASSUME_NONNULL_END
```

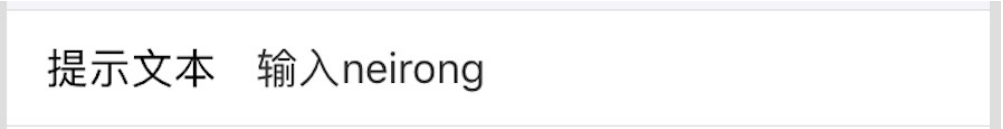
代码示例

见 AUAmountInputBox 的 代码示例 。

5.3.5 普通输入框

AUInputBox 为普通输入框，支持左侧标题以及右侧图像按钮。

效果图



接口说明

```
typedef NS_ENUM(NSInteger, AUInputBoxType)
{
AUInputBoxTypeMobileNumber, // 手机号码
AUInputBoxTypeCreditCard, // 信用卡
AUInputBoxTypeBankCard, // 借记卡
AUInputBoxTypeAmount, // 金额
AUInputBoxTypeIDNumber, // 身份证
AUInputBoxTypeNotEmpty, // 非空
AUInputBoxTypeAlipayAccount, // mPaaS 应用账号
AUInputBoxTypeNone // 不校验
};

typedef enum AUInputBoxStyle
{
AUInputBoxStyleNone, // 没有背景图片
AUInputBoxStyleiOS6, // 圆角的背景图片
AUInputBoxStyleiOS7 // 非圆角的背景图片
} AUInputBoxStyle;


/**
普通输入框，可带标题文字，按钮图片样式
*/
@interface AUInputBox : UIView

#pragma mark - AUInputBox 属性

// 文本输入框
@property(strong, nonatomic) AUTextField *textField;
@property(strong, nonatomic) NSString *textFieldText;
@property(strong, nonatomic) NSString *textFieldFormat;
@property(assign, nonatomic) CGFloat horizontalMargin;
@property(assign, nonatomic) CGFloat textFieldHorizontalMargin;


// 按钮
@property(strong, nonatomic) UIButton *iconButton;
@property(assign, nonatomic) BOOL hidesButtonWhileNotEmpty;
@property(assign, nonatomic) BOOL hidesButton;


// 显示在输入框左边的 lable
@property(nonatomic, readonly) UILabel *titleLabel;
@property(nonatomic, assign) CGFloat titleLabelWidth;


// 样式、验证器、背景图、文本框类型
@property(assign, nonatomic) AUInputBoxStyle style;
```

```
@property(readonly, nonatomic) UIImageView *backgroundImage;
@property(assign, nonatomic) AUInputBoxType inputBoxType;

#pragma mark - AUInputBox 静态方法
/**
* 创建输入框组件
* @param originY 输入框的 Y 坐标
* @param type 文本输入框的类型
* @return 输入框组件
*/
+ (instancetype)inputboxWithOriginY:(CGFloat)originY inputboxType:(AUInputBoxType)type;

/**
* 创建带图标按钮的输入框组件
* @param originY 输入框的 Y 坐标
* @param icon 按钮上的图标，44x44
* @param type 文本输入框的类型
* @return 带按钮的输入框组件
*/
+ (instancetype)inputboxWithOriginY:(CGFloat)originY buttonIcon:(UIImage *)icon
inputboxType:(AUInputBoxType)type;

/**
* @return 控件高度，默认值为 44，iPhone6 plus 为 47
*/
+ (float)heightOfControl;


#pragma mark - AUInputBox 实例方法

- (instancetype)initWithFrame:(CGRect)frame inputboxType:(AUInputBoxType)type;

- (void)buildIconButton:(UIImage *)icon;

/**
* 按照指定格式对文本添加空格
* @param text 文本内容
* @return 添加空格后的文本
*/
- (NSString *)formatText:(NSString *)text;

/**
* 对于没有在初始化时指定 icon 的 inputBox, 可以使用此方法添加
* @param icon 按钮上的图标
*
*/
- (void)setRightButtonIcon:(UIImage *)icon;

/**
* 检查输入的有效性.
*/
- (BOOL)checkInputValidity;

/**
* 过滤文本，只可输入数字，限定最大长度
* 参数为相应 delegate 参数，maxLength 为最大长度
```

```
*/
- (BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string withMaxLength:(int)maxLength;

/**
* 限定最大长度
* @maxLength 最大长度，不包括 format 的空格
*/
- (BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string
withFormatStringMaxLength:(int)maxLength;
```

代码示例

普通输入框：

```
AUInputBox *inputBox = [AUInputBox inputboxWithOriginY:startY inputboxType:AUInputBoxTypeNone];
inputBox.titleLabel.text = @"提示文本";
inputBox.textField.placeholder = @"请按提示输入";
[self.view addSubview:inputBox];
```

图片按钮：

```
AUInputBox *iconInputBox = [AUInputBox inputboxWithOriginY:startY buttonIcon:image
inputboxType:AUInputBoxTypeNone];
iconInputBox.titleLabel.text = @"提示文本";
iconInputBox.textField.placeholder = @"请按提示输入";
[self.view addSubview:iconInputBox];
```

5.3.6 搜索输入框

- AUSearchTitleView 为搜索栏点击入口控件。

  该组件类似搜索栏，但仅支持点击，提供如下三种样式：

  AUSearchTitleStyleDefault = 0：黑色文字，使用于浅色背景

  示例：mPaaS 全部应用页导航栏搜索入口样式。

  AUSearchTitleStyleMiddleAlign：黑色文字，使用于浅色背景（居中对齐）

  示例：联系人页面搜索入口样式。

  AUSearchTitleStyleContent：白色文字，适用于深色背景

  示例：mPaaS 应用首页导航栏搜索入口样式。

效果图

依赖

AUSearchTitleView 的依赖如下：

```
AntUI(iOS)
1.0.0.161108003457
APCommonUI(iOS)
1.2.0.161108102201
```

接口说明

```
typedef NS_ENUM(NSInteger, AUSearchTitleStyle) {
AUSearchTitleStyleDefault = 0, // 黑色文字，使用于浅色背景
AUSearchTitleStyleMiddleAlign, // 黑色文字，使用于浅色背景（居中对齐）
AUSearchTitleStyleContent, // 白色文字，适用于深色背景
};

@class AUSearchTitleView;

@protocol AUSearchTitleViewDelegate <NSObject>

@optional

// 点击搜索栏入口控件
- (void)didPressedTitleView:(AUSearchTitleView *)titleView;

// 点击搜索栏入口控件的 voice 按钮
- (void)didPressedVoiceButton:(AUSearchTitleView *)titleView;

@end
```

```
/**
搜索栏入口控件（默认宽度占据整个屏幕）
*/
@interface AUSearchTitleView : UIView

@property(nonatomic, assign)AUSearchTitleStyle style; //搜索背景样式，若不设置，默认为浅色背景

@property(nonatomic,strong) NSString *placeHolder; //搜索框 placeholder，默认为“搜索”
@property(nonatomic,strong) UIColor *placeHolderColor; //placeholder 的颜色

@property (nonatomic, weak) id<AUSearchTitleViewDelegate> delegate;

@property(nonatomic,strong) UIImage *searchIconImage; //搜索 icon
@property(nonatomic,strong) UIColor *normalBackgroundColor; //搜索框的背景颜色
@property(nonatomic,assign) BOOL isShowVoiceIcon; //是否显示语音搜索 icon，默认不显示


/**
* 搜索框距外层透明 View 的左右内边距，默认为 9。如业务需设置初始化的实例 View 与其他 View 的间距，请将内边距值考虑在内，否则视觉上会有误差
* 说明：将初始化的实例设为 navigationItem 的 titleview 时，系统会自适应布局 titlview 与左右 item 的间距，为满足视觉需求，设置了搜索框距外层透明 View 的内边距。
*
* 如有特殊需求，可重设此内边距
*
*/
@property(nonatomic,assign) CGFloat marginBetweenItem;

/**
* 获取实例的方法
*
* @param style 搜索框的style
*
* @return 获取的实例
*/
- (id)initWithSearchStyle:(AUSearchTitleStyle)style;

/**
* 默认调起全局搜索页面，若业务需自定义点击搜索框的事件，请在子类中重写此方法
*/
- (void)onClicked;

@end
```

代码示例

应用于导航栏：

```
AUSearchTitleView *titleView = [[AUSearchTitleView alloc] initWithSearchStyle:AUSearchTitleStyleDefault];
titleView.placeHolder = @"搜索栏入口样式";
titleView.placeHolderColor = [UIColor blackColor];
titleView.normalBackgroundColor = [UIColor orangeColor];
titleView.isShowVoiceIcon = YES;
titleView.delegate = self;
self.navigationItem.titleView = titleView;
```

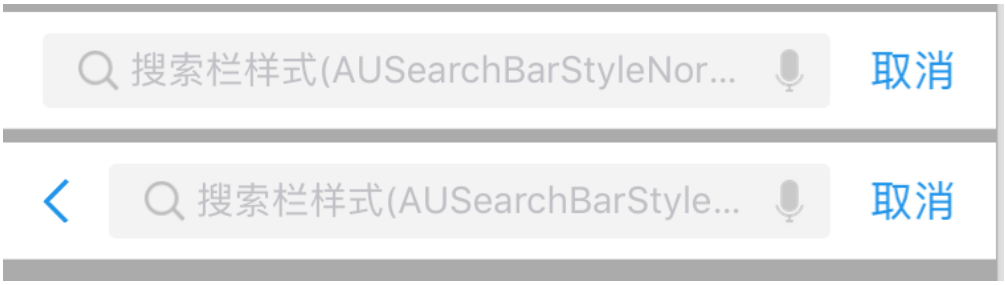
应用于普通视图

```
titleView = [[AUSearchTitleView alloc] initWithSearchStyle:AUSearchTitleStyleMiddleAlign];
titleView.placeHolder = @"AUSearchTitleStyleMiddleAlign样式";
titleView.isShowVoiceIcon = YES;
titleView.delegate = self;
[self.view addSubview:titleView];
```

5.3.7 搜索栏组件

- AUSearchBar 为 mPaaS 搜索栏控件 。
- 该组件提供两种显示样式：
    - AUSearchBarStyleNormal：带取消按钮的搜索栏，参考全局搜索主页搜索栏。
    - AUSearchBarStyleDetail：带取消和返回按钮的搜索栏，参考全局搜索二级页面搜索栏。

效果图



依赖

AUSearchBar 的依赖如下：

```
AntUI(iOS)
1.0.0.161108003457
APCommonUI(iOS)
1.2.0.161108102201
```

接口说明

```
@class AUSearchBar;

@protocol AUSearchBarDelegate <NSObject>

@optional

#pragma mark - 对应 UITextField 的代理方法
//
- (BOOL)searchBarTextShouldBeginEditing:(AUSearchBar *)searchBar;
//
- (BOOL)searchBarTextShouldEndEditing:(AUSearchBar *)searchBar;
```

```
// called when text starts editing
- (void)searchBarTextDidBeginEditing:(AUSearchBar *)searchBar;
// called when text ends editing
- (void)searchBarTextDidEndEditing:(AUSearchBar *)searchBar;
// called when text changes (including clear)
- (void)searchBar:(AUSearchBar *)searchBar textDidChange:(NSString *)searchText;
// called before text changes
- (BOOL)searchBar:(AUSearchBar *)searchBar shouldChangeTextInRange:(NSRange)range
replacementText:(NSString *)text;

- (BOOL)searchBarShouldClear:(AUSearchBar *)searchBar;

#pragma mark - 其他代理方法

// 键盘搜索按钮点击后的回调
- (void)searchBarSearchButtonClicked:(AUSearchBar *)searchBar;

// 取消按钮点击后的回调
- (void)searchBarCancelButtonClicked:(AUSearchBar *) searchBar;

// 返回按钮点击后的回调（AUSearchBarStyleDetail 有效）
- (void)searchBarBackButtonClicked:(AUSearchBar *)searchBar;

// voice按钮点击后的回调（shouldShowVoiceButton 为 YES 时有效）
- (void)searchBarOpenVoiceAssister:(AUSearchBar *)searchBar;

@end



typedef NS_ENUM(NSUInteger, AUSearchBarStyle) {
AUSearchBarStyleNormal = 0,//normal
AUSearchBarStyleDetail, //has back Button
};



/**
搜索栏控件（默认宽度和屏幕宽度一致，高度 44）
*/
@interface AUSearchBar : UIView

@property (nonatomic, strong) NSString *text; // 搜索框文本
@property (nonatomic, assign) BOOL isSupportHanziMode; // 是否支持汉字边输入边搜索模式，默认为 YES
@property (nonatomic, assign) AUSearchBarStyle style; // 搜索框样式
@property (nonatomic, assign) BOOL shouldShowVoiceButton; // 是否显示 voice 按钮，默认为 NO
@property (nonatomic, strong, readonly) UITextField *searchTextField; // 搜索框
@property (nonatomic, weak) id<AUSearchBarDelegate> delegate;

/**
初始化方法

@param style 搜索框样式

@return AUSearchBar实例
*/
```

```
- (instancetype)initWithStyle:(AUSearchBarStyle)style;

@end
```

代码示例

- 添加到导航栏：

```
AUSearchBar *searchBar = [[AUSearchBar alloc] initWithStyle:AUSearchBarStyleNormal];
searchBar.searchTextField.placeholder = @"搜索栏样式(AUSearchBarStyleNormal)";
searchBar.delegate = self;
searchBar.isSupportHanziMode = YES;
searchBar.shouldShowVoiceButton = YES;
self.navigationItem.titleView = searchBar;
self.navigationItem.leftBarButtonItem = nil; // 需要将左边导航按钮置空
self.navigationItem.rightBarButtonItem = nil; // 需要将右边导航按钮置空
self.navigationItem.hidesBackButton = YES; // 需要隐藏返回按钮
```

- 添加到普通视图：

```
searchBar = [[AUSearchBar alloc] initWithStyle:AUSearchBarStyleDetail];
searchBar.searchTextField.placeholder = @"搜索栏样式(AUSearchBarStyleDetail)";
searchBar.delegate = self;
searchBar.isSupportHanziMode = YES;
searchBar.shouldShowVoiceButton = YES;
[self.view addSubview:searchBar];
```

5.3.8 验证码输入框

AUTextCodeInputBox 为验证码输入控件。

效果图

接口说明

```
/**
短信验证码输入框，带倒计时按钮
*/
```

```
@interface AUTextCodeInputBox : AUSecurityCodeBox

/**
发送短信前的等待时间
*/
@property (nonatomic, assign) NSTimeInterval interval;

/**
* 创建短信验证码输入框
* @param frame 在父类的位置和大小
* @param interval 发送短信前的等待时间
* @return 短信验证码输入框
*/
- (AUTextCodeInputBox *)initWithFrame:(CGRect)frame interval:(NSTimeInterval)interval;

/**
* 创建短信验证码输入框
* @param originY 组件的 Y 坐标
* @param interval 发送短信前的等待时间
* @return 短信验证码输入框
*/
- (AUTextCodeInputBox *)initWithOriginY:(CGFloat)originY interval:(NSTimeInterval)interval;

/**
* 设置倒计时结束时执行的 block
* @param block 执行的 block
*/
- (void)setCountdownDidCompleteBlock:(void (^)(void))block;
```

代码示例

```
AUTextCodeInputBox *smsInputBox = [[AUTextCodeInputBox alloc] initWithOriginY:startY interval:60];
[smsInputBox.actionButton addTarget:self action:@selector(onSmsButtonClicked:)
forControlEvents:UIControlEventTouchUpInside]; // 处理右侧按钮的点击回调
[self.view addSubview:smsInputBox];
```

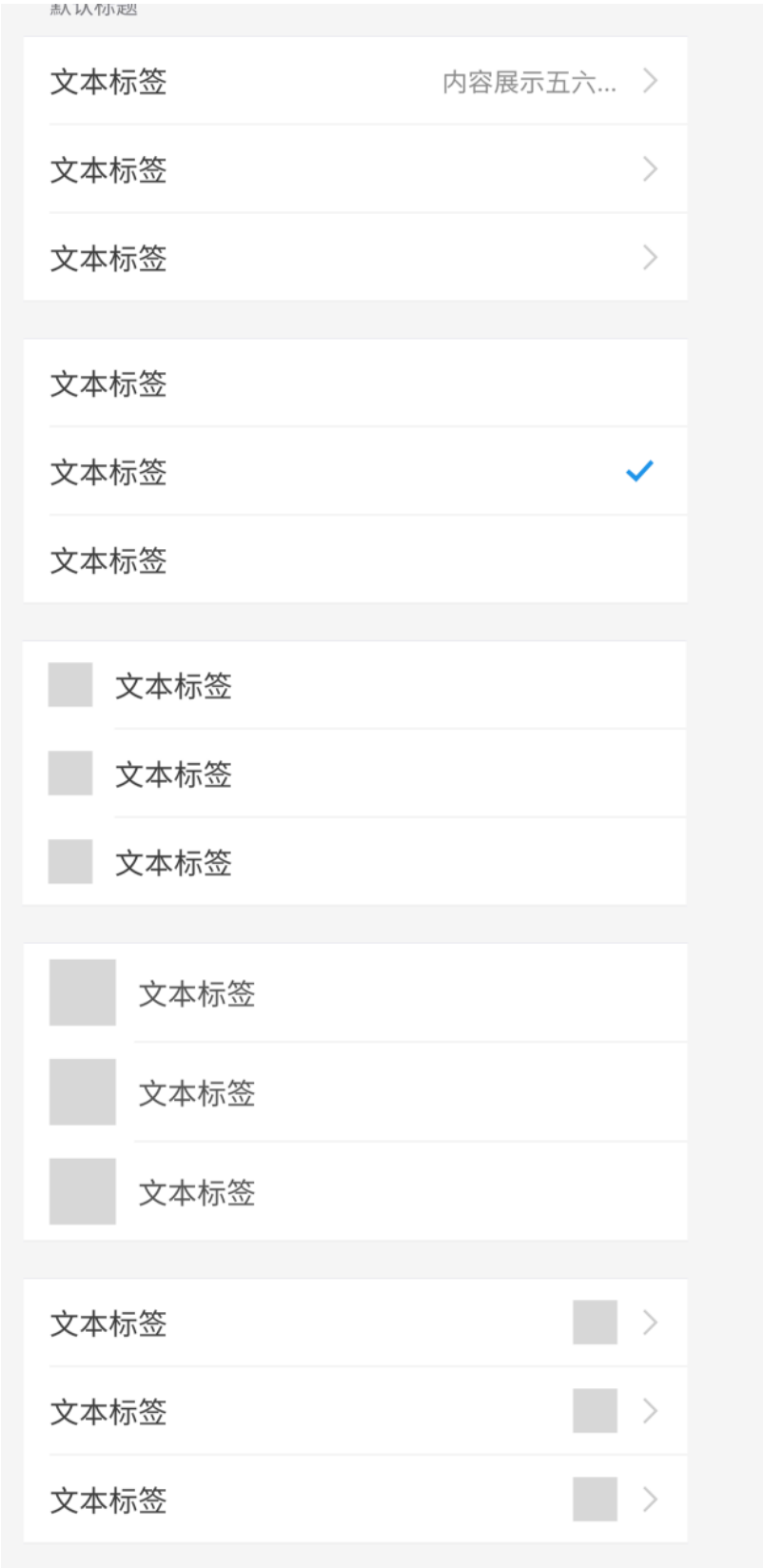
5.4 条目组件

- AUListItem 是根据 UED 新需求完成的控件系列，与原 APCommonUI 中的 APTableView 控件不互通。
- AUListItem 包括新的四种 ListItem，支持的元素如下表：

AUListItem	主标题	副标题	左图标	右图标	定制大小的左图标	选中时显示 checkmark	最右的辅助箭头图标
AUSingleTitleListItem	YES	YES	YES	YES	YES	YES	YES
AUDoubleTitleListItem	YES	YES	YES	—	YES	—	YES
AUCheckBoxListItem	YES	—	—	—	—	—	YES
AUSwitchListItem	YES	—	—	—	—	—	—

效果图

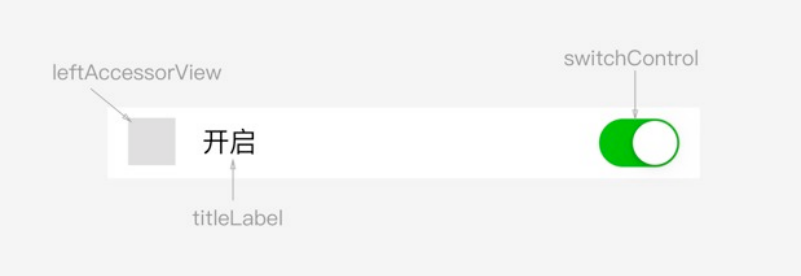
AUSingleTitleListItem



AUDoubleTitleListItem

AUCheckBoxListItem

AUSwitchListItem



依赖

AUListItem 的依赖如下：

```
import <UIKit/UIKit.h>
```

接口说明

共用接口

Model 层面：

设置多个 delegate，主要是为了规范外部业务方传参，有些元素不支持展示，外部也无法传参。比如，在 AUDoubleTitleListItem 中右边的图标不存在，则 AUDoubleTitleListItemModelDelegate 不会包含 rightImage 参数。

```
AUListItemProtocols.h
/**
AUSingleTitleListItem 可以设置和访问的数据项
*/
@protocol AUSingleTitleListItemModelDelegate <NSObject>

@property (nonatomic, copy) NSString *subtitle; //副标题
@property (nonatomic, strong) UIImage *leftImage; //左边图片
@property (nonatomic, strong) UIImage *rightImage; //右边文字前的图片
@property (nonatomic, strong) UIImage *rightAssistImage; //右边文字后的图片
@property (nonatomic, assign) CGSize leftimageSize; //左边图片大小可定制，不设置使用默认大小 22
@property (nonatomic, assign) CGSize rightAssistImageSize; //右边文字后的图片大小可定制，不设置使用默认大小 22
```

```
@end


/**
AUDoubleTitleListItem可以设置和访问的数据项

*/
@protocol AUDoubleTitleListItemModelDelegate <NSObject>

@property (nonatomic, copy) NSString *subtitle; //副标题
@property (nonatomic, strong) UIImage *leftImage; //左边图片
@property (nonatomic, assign) CGSize leftimageSize; //左边图片大小可定制，不设置使用默认大小
@property (nonatomic, copy) NSString *timeString; //右边时间
@property (nonatomic, copy) NSString *rightAssistString; //右边辅助信息，默认居中
@property (nonatomic, assign) NSInteger subtitleLines; //辅助标题行数，(业务方必须指定具体行数)
//@property (nonatomic, assign) BOOL showAccessory; //是否展示辅助图标

@end

/**
AUCheckBoxListItem可以设置和访问的数据项

*/
@protocol AUCheckBoxListItemModelDelegate <NSObject>
//@property (nonatomic, assign) BOOL showAccessory; //是否展示辅助图标

@end

/**
AUMultiListItemDelagate 可以设置和访问的数据项

*/
@protocol AUMultiListItemDelagate <NSObject>

@property (nonatomic, copy) NSString *subtitle; //副标题
@property (nonatomic, strong) UIImage *leftImage; //左边图片
//@property (nonatomic, assign) CGSize leftimageSize; //左边图片
@property (nonatomic, assign) BOOL showAccessory; //是否展示辅助图标
@property (nonatomic, assign) NSInteger subtitleLines; // 设置副标题的行数

@end


/**
AUMultiListBottomAssistDelagate 可以设置和访问的数据项

*/
@protocol AUMultiListBottomAssistDelagate <NSObject>

@property (nonatomic, strong) NSString *originalText; // 文字来源
@property (nonatomic, strong) NSString *timeDesc; // 时间信息描述
@property (nonatomic, strong) NSString *othersDesc; // 其他描述信息

@end
```

```
/**
AUParallelTitleListItem可以设置和访问的数据项

*/
@protocol AUParallelTitleListItemModelDelegate <NSObject>
@property (nonatomic, copy) NSString *subtitle; //标题二
@property (nonatomic, copy) NSString *describe; //描述一
@property (nonatomic, copy) NSString *subDescribe; //描述二

@end


/**
AULineBreakListItem可以设置和访问的数据项

*/
@protocol AULineBreakListItemModelDelegate <NSObject>
@property (nonatomic, copy) NSString *subtitle; //副标题
@end


/**
AUCouponsItemDelagate 可以设置和访问的数据项

*/
@protocol AUCouponsItemDelagate <NSObject>

@property (nonatomic, copy) NSString *subtitle; // 副标题
@property (nonatomic, strong) UIImage *leftImage; // 左边图片
@property (nonatomic, strong) UIImage *leftImageUrl; // 左边图片链接
@property (nonatomic, strong) NSString *assistDesc; // 文字辅助说明
@property (nonatomic, assign) NSInteger totalWidth; // 设置卡片宽度

@end


/**
TTTAttributeLabelDelagate 富文本协议

*/

@protocol TTTAttributeLabelDelagate <NSObject>

@property (nonatomic, copy) NSString *attributeText; // 富文本内容
@property (nonatomic, copy) NSString *linkText; // 富文本链接文案
@property (nonatomic, copy) NSString *linkURL; // 富文本跳转链接

@end


AUListItemModel.h
import"AUListItemProtocols.h"
@interface AUListItemModel : NSObject
@property (nonatomic, copy) NSString *title; //主标题
@property (nonatomic, assign) UIEdgeInsets separatorLineInset; //可设置分隔线离 cell 的左、右距离
```

```
@end
```

View 层面：

AUBaseListItem.h：

```
@interface AUBaseListItem : UITableViewCell
//以下开放出来为方便外部设置额外属性，如：主标题颜色
@property(nonatomic,strong) UILabel *titleLabel;
@property(nonatomic,strong) UIView *separatorLine;
/**
初始化函数
@param reuseIdentifier 重用标识
@param block 外部传入的 block，一般外部会在此 block 中设置 title、leftimage 等
@return 返回 self 实例
*/
- (instancetype)initWithReuseIdentifier:(NSString *)reuseIdentifier model:(void(^)(AUListItemModel*model))block;
/**
返回 cell 高度
@return 返回 cell 高度
*/
+ (CGFloat)cellHeight ;
@end

#ifdef AUBaseListItem_protected
//只对子类开放，在子类中 import AUBaseListItem 之前，设置 AUBaseListItem_protected = 1
@interface AUBaseListItem ()
@property (nonatomic,strong) AUListItemModel* baseModel;
@end

#endif
/**
业务方一般调用 AUBaseListItem 子类的【initWithReuseIdentifier:model:】方法即可满足需求
这里提供单独的 针对 title 等参数方法
除 title 外，都放在子类实现，访问隔断
*/
@interface AUBaseListItem (Extensions)
/**
设置主标题
@param title 主标题字符串
*/
- (void)setTitle:(NSString* )title;

/**
主标题 get 方法
@return 返回主标题字符串
*/
- (NSString*)title ;

/**
设置分割线距离 cell 左右的距离
@param separatorLineInset UIEdgeInsets 参数
*/
- (void)setSeparatorLineInset:(UIEdgeInsets)separatorLineInset;
```

```
/**
get分割线inset
@return 分割线的inset
*/
- (UIEdgeInsets)separatorLineInset;
```

AUSingleTitleListItem

```
typedef NS_ENUM(NSInteger, AUSingleTitleListItemStyle) {
AUSingleTitleListItemStyleDefault, // 高度 92，图标 58
AUSingleTitleListItemStyleValue1, // 高度 110，图标 72
};

@interface AUSingleTitleListItem : AUBaseListItem

@property(nonatomic,strong) UILabel *subtitleLabel;
@property(nonatomic,strong) UIImageView *leftImageView;
@property(nonatomic,strong) UIImageView *rightImageView;
@property(nonatomic,strong) UIImageView *rightAssistImageView;

/**重要
初始化函数

@param reuseIdentifier 重用标识
@param block 外部传入的 block，一般外部会在此 block 中设置 title、leftimage 等

@return 返回 self 实例
*/
- (instancetype)initWithReuseIdentifier:(NSString*)reuseIdentifier
model:(void(^)(AUListItemModel<AUSingleTitleListItemModelDelegate>*model))block __deprecated_msg("预计废弃
，请勿继续使用");



/**
设置 cell 展示所需的所有数据

@param block 传入的 block
*/
- (void)setModelBlock:(void(^)(AUListItemModel<AUSingleTitleListItemModelDelegate>*model))block;


/**
初始化函数

@param reuseIdentifier 标识
@param style 自定义的类型，详见 AUSingleTitleListItemStyle
@return 返回self实例
*/
- (instancetype)initWithReuseIdentifier:(NSString*)reuseIdentifier customStyle:(AUSingleTitleListItemStyle)style;



/**
根据不同 style 返回不同高度
```

```
@param style
@return 自定义类型，详见AUSingleTitleListItemStyle
*/
+ (CGFloat)cellHeightForStyle:(AUSingleTitleListItemStyle)style;

@end
```

AUDoubleTitleListItem

```
typedef NS_ENUM(NSInteger, AUDoubleTitleListItemStyle) {
AUDoubleTitleListItemStyleDefault, // 有左图，高度 120，图标 76 (单位：px)
AUDoubleTitleListItemStyleValue1, // 无左图，高度 120，(单位：px)
AUDoubleTitleListItemStyleValue2, // 有左图，高度 144，图标 88 (单位：px)
};

@interface AUDoubleTitleListItem : AUBaseListItem<AUDoubleTitleListItemModelDelegate,
TTTAttributeLabelDelagate>

@property(nonatomic,strong) UILabel *subtitleLabel;
@property(nonatomic,strong) UIImageView *leftImageView;
@property(nonatomic,strong) UILabel* timeLabel;
@property(nonatomic,strong) UILabel *rightAssistLabel;


/**
设置 cell 展示所需的所有数据

@param block 传入的 block
*/
- (void)setModelBlock:(void(^)(AUListItemModel<AUDoubleTitleListItemModelDelegate,
TTTAttributeLabelDelagate>*model))block;

/**
初始化函数

@param reuseIdentifier 标识
@param style 自定义的类型，详见 AUDoubleTitleListItemStyle
@return 返回 self 实例
*/
- (instancetype)initWithReuseIdentifier:(NSString*)reuseIdentifier customStyle:(AUDoubleTitleListItemStyle)style;


/**
根据不同 style 返回不同高度

@param style 自定义类型，详见AUDoubleTitleListItemStyle
@return 返回cell高度值
*/
+ (CGFloat)cellHeightForStyle:(AUDoubleTitleListItemStyle)style;



/**
根据不同 style 返回动态高度
```

```
@param style 自定义类型，详见 AUDoubleTitleListItemStyle
@param block 数据模型 详见 AUDoubleTitleListItemModelDelegate
需要注意：1. 该方法务必传入 model.accessoryType 的确切值；
2. 如果需要换行，请业务指定具体行数 subtitleLines
@return 返回 cell 高度值
*/
+ (CGFloat)cellHeightForStyle:(AUDoubleTitleListItemStyle)style
modelBlock:(void(^)(AUListItemModel<AUDoubleTitleListItemModelDelegate,
TTTAttributeLabelDelagate>*model))block;

@end
```

AUCheckBoxListItem

```
@protocol AUCheckBoxListItemDelegate <NSObject>

/**
checkbox 状态变化，给外部的回调

@param item checkbox 实例
*/
- (void)checkboxValueDidChanged:(AUCheckBox *)item;// 取 cell 的 tag 作为 item 的 tag

@end

@interface AUCheckBoxListItem : AUBaseListItem<AUCheckBoxListItemModelDelegate>

@property(nonatomic, assign, getter = isChecked) BOOL checked;//设置 checkbox 勾选状态
@property(nonatomic, assign, getter = isDisableCheck) BOOL disableCheck;//是否禁用 checkbox
@property(nonatomic, weak) id <AUCheckBoxListItemDelegate> delegate;

@end
```

AUSwitchListItem

```
@interface AUSwitchListItem : AUNBaseListItem

@property (nonatomic,strong) UISwitch *switchControl; // cell 中的 switch 控件

// 设置菊花为展示或者隐藏状态
- (void)showLoadingIndicator:(BOOL)show;

@end
```

自定义属性

属性名	用途	类型
title	主标题	NSString
titleLabel	主标题 Label	UILabel
subtitle	副标题	NSString

subtitleLabel	副标题 Label	UILabel
leftImage	左侧图标	UIImage
leftImageView	左侧图标 View	UIImageView
rightImage	右侧图标	UIImage
rightImageView	右侧图标 View	UIImageView
leftimageSize	左侧图标大小	CGSize
timeString	右侧时间字符串	NSString
timeLabel	右侧时间 Label	UILabel
showMarkWhenSelected	cell 被选中后，是否展示 checkmark	BOOL
showAccessory	是否展示辅助图标	BOOL
checked	设置 AUCheckBoxListItem 是否选中状态	BOOL
disableCheck	设置 AUCheckBoxListItem 是否 disable 状态	BOOL

注：各控件支持使用的属性在下面代码示例中展示。

代码示例

AUSingleTitleListItem

推荐用法：

```
AUSingleTitleListItem*cell = [[AUSingleTitleListItem alloc] initWithReuseIdentifier:identifierSingle1
model:^(AUListItemModel<AUSingleTitleListItemModelDelegate> *model) {
model.title = @"我是主标题";
model.subtitle = @"我是副副副标题";
model.showAccessory = YES;
model.XXX = XXXX;
//支持的属性包含在 AUListItemModel 以及 AUSingleTitleListItemDelegate 中，在上述接口说明中可查阅
}];
```

也可以单独设置某些提供的属性（与推荐用法中 model 中包含的保持一致）：

```
AUSingleTitleListItem*cell = [[AUSingleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"testsingle"];
cell.title = @"我是主标题";
```

支持控件上各个元素的设置：

```
AUSingleTitleListItem*cell = [[AUSingleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"testsingle"];
cell.titleLabel.backgroundColor = [UIColor redColor];
```

AUDoubleTitleListItem

推荐用法：

```
AUDoubleTitleListItem*cell = [[AUDoubleTitleListItem alloc] initWithReuseIdentifier:identifierDouble3
model:^(AUListItemModel<AUDoubleTitleListItemModelDelegate> *model) {
model.title = @"我不支持设置右边图像";
model.leftImage = [UIImage imageNamed:@"AntUI.bundle/ilustration_ap_expection_limit.png"];
model.leftimageSize = CGSizeMake(100, 100);
model.showAccessory = YES;

//支持的属性包含在 AUListItemModel 以及 AUDoubleTitleListItemDelegate 中，在上述接口说明中可查阅
}];
```

也可以单独设置提供的属性：

```
AUDoubleTitleListItem*cell = [[AUDoubleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"testdouble"];
cell.leftImage = [UIImage imageNamed:@"AntUI.bundle/ilustration_ap_expection_limit.png"];
```

支持控件上各个元素的设置：

```
AUDoubleTitleListItem*cell = [[AUDoubleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"testdouble"];
cell.leftImageView.image = [UIImage imageNamed:@"AntUI.bundle/ilustration_ap_expection_limit.png"];
```

AUCheckBoxListItem

推荐用法：

```
AUCheckBoxListItem* cell = [[AUCheckBoxListItem alloc] initWithReuseIdentifier:identifierChecbkox
model:^(AUListItemModel<AUCheckBoxListItemModelDelegate> *model) {
model.title = @"我默认被选中";
model.showAccessory = NO;
//只支持这两种设置
}];
cell.disableCheck = YES;//设置 check 按钮为 disable 状态
```

也可以单独设置提供的属性：

```
AUCheckBoxListItem*cell = [[AUCheckBoxListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"testcheck"];
cell.showAccessory =YES;
```

支持控件上各个元素的设置：

```
AUCheckBoxListItem*cell = [[AUDoubleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"testcheck"];
cell.titleLabel.text = @"我默认为被选中";
```

AUSwitchListItem

```
AUSwitchListItem *switchCell = [[AUSwitchListItem alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"switchCell"];
AUListItemModel *model = _datas[indexPath.row];
switchCell.titleLabel.text = model.title;
switchCell.leftAccessorView = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"certify.png"]];
switchCell.leftAccessorType = AUListItemLeftAccessorTypeIcon;
switchCell.switchControl.on = NO;
UISwitch *switchView = (UISwitch *)switchCell.accessoryView;
[switchView addTarget:self action:@selector(switchValueDidChanged:)
forControlEvents:UIControlEventValueChanged];
return switchCell;
```

5.5 弹窗组件

5.5.1 选项卡

AUActionSheet 迁移自 APActionSheet，样式稍有调整，支持普通带删除按钮外观以及普通选项按钮外观。

效果图

带删除按钮：

选项卡按钮：

红点：

依赖

AUActionSheet 的依赖如下：

```
AntUI(iOS)
1.0.0.161108003457
APCommonUI(iOS)
1.2.0.161108102201
```

接口说明

```
typedef NS_ENUM(NSInteger, AUActionSheetButtonType) {
```

```
AUActionSheetButtonTypeDefault = 0, // 默认
AUActionSheetButtonTypeDisabled, // 不可点击
AUActionSheetButtonTypeDestructive, // 红色删除性按钮
AUActionSheetButtonTypeCustom // 自定义
};

/**
AUAction Sheet，接口迁移自 APActionSheet，展示样式做了一些调整
*/
@interface AUActionSheet : UIView<UIAppearanceContainer>

/// 按钮高度 默认为 42
@property (nonatomic) CGFloat buttonHeight UI_APPEARANCE_SELECTOR;
/// 取消按钮高度
@property (nonatomic) CGFloat cancelButtonHeight UI_APPEARANCE_SELECTOR;
/// 分割线颜色 默认为 AU_COLOR_LINE
@property (strong, nonatomic) UIColor *separatorColor UI_APPEARANCE_SELECTOR;
/// 按钮点击背景色
@property (strong, nonatomic) UIColor *selectedBackgroundColor UI_APPEARANCE_SELECTOR;
// UI 组件的一些 Attributes
@property (copy, nonatomic) NSDictionary *titleTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *buttonTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *disabledButtonTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *destructiveButtonTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *cancelButtonTextAttributes UI_APPEARANCE_SELECTOR;

/// 显示隐藏动画时间，默认为 0.5
@property (nonatomic) NSTimeInterval animationDuration UI_APPEARANCE_SELECTOR;
/// 标题
@property(nonatomic,copy) NSString *title;
/// 是否已经展示
@property(nonatomic, readonly, getter=isVisible) BOOL visible;
/// 自定义按钮上方顶部视图
@property (strong, nonatomic) UIView *headerView;
/// ActionSheet 实例显示前的 keyWindow
@property (weak, nonatomic, readonly) UIWindow *previousKeyWindow;
/// 协议代理
@property(nonatomic,weak)id<UIActionSheetDelegate> delegate;
/// 取消按钮标题
@property (copy, nonatomic) NSString *cancelButtonTitle;
/// 按钮个数
@property(nonatomic, readonly) NSInteger numberOfButtons;
/// 取消按钮索引，默认为 -1
@property(nonatomic) NSInteger cancelButtonIndex;
/// 破坏性红色按钮索引值，默认为 -1，如果只有一个按钮则忽略。
@property(nonatomic) NSInteger destructiveButtonIndex;
/**
AUActionSheet 初始化方法

@param title 标题信息
@param delegate 代理对象
@param cancelButtonTitle 取消按钮标题
@param destructiveButtonTitle 破坏性按钮标题
@param otherButtonTitles 其他按钮标题参数列表
@return AUActionSheet 实例
*/
```

```
- (instancetype)initWithTitle:(NSString *)title delegate:(id<UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle destructiveButtonTitle:(NSString *)destructiveButtonTitle
otherButtonTitles:(NSString *)otherButtonTitles, ... NS_REQUIRES_NIL_TERMINATION;

/**
AUActionSheet 初始化方法

@param title 标题信息
@param delegate 代理对象
@param cancelButtonTitle 取消按钮标题
@param destructiveButtonTitle 破坏性按钮标题
@param items customOption数据列表（自定义 titleColor，红点）
@return AUActionSheet 实例
*/
- (instancetype)initWithTitle:(NSString *)title
delegate:(id<UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle
destructiveButtonTitle:(NSString *)destructiveButtonTitle
items:(NSArray<AUActionSheetItem *> *)items;

/**
增加一个按钮, 类型为默认类型

@param title 按钮标题
@return 按钮索引值，从 0 起
*/
- (NSInteger)addButtonWithTitle:(NSString *)title;

/**
增加一个按钮

@param title 按钮标题
@param type 按钮类型
@return 按钮索引值，从 0 起
*/
- (NSInteger)addButtonWithTitle:(NSString *)title type:(AUActionSheetButtonType)type;

/**
通过索引值获取按钮标题

@param buttonIndex 按钮索引值
@return 按钮标题
*/
- (NSString *)buttonTitleAtIndex:(NSInteger)buttonIndex;

/**
设置某个位置的按钮

@param item 封装信息后的按钮类型
@param index 需要替换的索引值，小于现有按钮个数
*/
- (void)setButton:(AUActionSheetItem *)item atIndex:(NSInteger)index;

/** ActionSheet 展示方法 */
- (void)show;
```

```
/**
手动调用隐藏方法

@param animate 隐藏是否带动画
*/
- (void)closeWithAnimate:(BOOL)animate;

/**
手动模拟按钮点击隐藏方法（会回调按钮点击相关的协议方法）

@param buttonIndex 按钮索引
@param animated 隐藏是否带动画
*/
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex animated:(BOOL)animated;

/**
* 动态添加 item
* 注意：请在 actionSheet show 出来后并显示在屏幕时调用；如需在 show 前 addButton，请使用 addButtonWithTitle
*
* @param item 自定义 item
* @param index 添加的位置
*/
- (void)addButton:(AUActionSheetItem *)item atIndex:(NSInteger)index;

// 设置后台模式，如果为 YES 或者 @(YES) 则隐藏所有已经展示的 ActionSheet。默认为 NO
+(void)setIsBackGroundMode:(BOOL)isBackGroundMode;
+(void)weakSetIsBackGroundMode:(id)isBackGroundMode;

- (void)showFromToolbar:(UIToolbar *)view;
- (void)showFromTabBar:(UITabBar *)view;
- (void)showFromBarButtonItem:(UIBarButtonItem *)item animated:(BOOL)animated NS_AVAILABLE_IOS(3_2);
- (void)showFromRect:(CGRect)rect inView:(UIView *)view animated:(BOOL)animated NS_AVAILABLE_IOS(3_2);
- (void)showInView:(UIView *)view;

@end


/** 封装后的 ActionSheet 按钮类 */
@interface AUActionSheetItem : NSObject
/// 按钮标题
@property (copy, nonatomic) NSString *title;
/// 按钮的类型
@property (nonatomic) AUActionSheetButtonType type;
/// 按钮标题颜色，如果设置该值，请手动将按钮类型调整为 AUActionSheetButtonTypeCustom
@property (strong,nonatomic) UIColor *titleColor;

/**
* 设置显示“红点”样式
*
* badgeValue: @"."显示红点
* @"new"显示 new
* @"数字"显示数字，大于 99 则显示图片 more（...）
* @"惠"/"hui"显示“惠”字
* @"xin"显示"新"字
* nil 清除当前显示
*/
```

```
@property (nonatomic, copy) NSString *badgeValue;

@end
```

代码示例

带删除按钮：

```
AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:@"这是提供一行或二行注释, 通过信息澄清的方式避免用户产生疑问"
delegate:self
cancelButtonTitle:@"取消"
destructiveButtonTitle:@"确认删除"
otherButtonTitles:nil];
[actionSheet show];
```

选项卡按钮：

```
AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:nil
delegate:self
cancelButtonTitle:@"取消"
destructiveButtonTitle:nil
otherButtonTitles:@"选项一",@"选项二",@"选项三", nil];
[actionSheet show];
```

设置某个选项添加红点：

```
AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:nil
delegate:self
cancelButtonTitle:@"取消"
destructiveButtonTitle:nil
otherButtonTitles:@"选项一",@"选项二",@"选项三", nil];
AUActionSheetItem *item = [[AUActionSheetItem alloc] init];
item.title = @"选项三";
item.type = AUActionSheetButtonTypeCustom;
item.badgeValue = @"new";
item.titleColor = [UIColor redColor];
[actionSheet setButton:item atIndex:2];

[actionSheet show];
```

5.5.2 日期组件

AUDatePicker 为日期选择控件。

效果图


运营商
下午9:20
AntUI 实例
AUDatePicker
类方法创建
成员方法创建
时间选择器1
时间选择器2
取消
请选择时间
完成
2016年 12月 18日
2017年 1月 19日
2018年 2月 20日
2019年 3月 21日
2020年 4月 22日
2021年 5月 23日
2022年 6月 24日


运营商
下午9:20
AntUI 实例
AUDatePicker
类方法创建
成员方法创建
时间选择器1
时间选择器2
取消
请选择时间
完成
2013年 7月 11日
2014年 8月 12日
2015年 9月 13日
2016年 10月 14日
2017年 11月 15日
2018年 12月 16日
2019年 1月 17日

接口说明

- AUDatePicker.h

```
//
// ALPPicketView.h
// TestCell
//
```

```
#import <UIKit/UIKit.h>

@class AUDatePicker;

@protocol AUDatePickerDelegate <UIPickerViewDataSource, UIPickerViewDelegate>

/*
* 点取消息时回调
*/
- (void)cancelPickerView:(AUDatePicker *)pickerView;

/*
* 点完成时回调，选中项可通过 pickerView/Users/zhuwei/ios-phone-antui/ANTUI/Sources/Views/pickerView/AUDatePicker.h selectedRowInComponent 返回
*/
- (void)selectedPickerView:(AUDatePicker *)pickerView;

@end
/* !
@class AUDatePicker
@abstract UIView
@discussion 原框架封装的选择器，在原来系统控件上加上的去掉和完成按钮
*/

@interface AUDatePicker : UIView

@property(nonatomic, strong) UIPickerView *pickerView; // 通用事务选择器
@property(nonatomic, strong) UIDatePicker *datePickerView; // 时间选择器

@property(nonatomic, assign) BOOL isDatePicker; // 当前是否是时间选择器，默认为 NO

@property(nonatomic, weak) id<AUDatePickerDelegate> delegate;


/*
* 创建组件
*
* @param title 标题，可为 nil
* @return 创建的组件，默认不显示，需调用 show
*/
+ (AUDatePicker *)pickerViewWithTitle:(NSString *)title;

/*
* 初始化对象
*
* @param frame 显示位置
* @param title 显示标题，不显示可设 nil
* @return 默认返回对象不显示，要显示需要调 show
*/
- (id)initWithFrame:(CGRect)frame withTitle:(NSString *)title;

/*
* 显示
*/
- (void)show;
```

```
/*
* 隐藏
*/
- (void)hide;

/**
* 重载数据
*/
- (void)reload;


/**
当 isDatePicker 为 YES 时，使用 datePickerView 选择时间

@param minDate 最小时间
@param maxDate 最大时间
*/
- (void) setTimeDateminDate:(NSDate *)minDate MaxDate:(NSDate *)maxDate;



/**
当 isDatePicker 为 YES 时，设置 datePickerView 的当前时间

@param currentDate 设置当前的时间
*/
- (void) setCurrentDate:(NSDate *) currentDate;


/**
当 isDatePicker 为 YES 时，设置时间选择器中选择的时间

@param date 选中的日期
@param animated 是否包含动画
*/
- (void)setAUDatePickerDate:(NSDate *)date animated:(BOOL)animated; // if animated is YES, animate the wheels of time to display the new date

@end
```

代码示例

```
//
// APPickerViewViewController.m
// UIDemo
//

#import"APPickerViewViewController.h"
#import"AUDatePicker.h"

@interface APPickerViewViewController ()<AUDatePickerDelegate,UIPickerViewDelegate,UIPickerViewDataSource>
@property(nonatomic,strong)AUDatePicker* apPickerView;
@property(nonatomic,strong)AUDatePicker* apPickerView2;
@property(nonatomic,strong)AUDatePicker* apPickerView3;
```

```
@property(nonatomic,strong)AUDatePicker* apPickerView4;

@property(nonatomic,strong)UILabel* textLabel;
@property(nonatomic,strong)NSArray* yearArray;
@property(nonatomic,strong)NSArray* monthArray;
@property(nonatomic,strong)NSArray* nameArray;
@end

@implementation APPickerViewViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
if (self) {
// Custom initialization
self.yearArray = @[@"2009",@"2010",@"2011",@"2012",@"2013",@"2014",@"2015",@"2016"];
self.monthArray = @[@"1",@"2",@"3",@"4",@"5",@"6",@"7",@"8",@"9",@"10",@"11",@"12"];
self.nameArray = @[@"赵一",@"钱二",@"孙三",@"李四",@"王五",@"张六",@"刘七"];
}
return self;
}

- (void)viewDidLoad
{
[super viewDidLoad];
// Do any additional setup after loading the view.
[self.view setBackgroundColor:[UIColor whiteColor]];

NSArray* items = @[@"类方法创建",@"成员方法创建",@"时间选择器1",@"时间选择器2"];
UISegmentedControl* segmentControl = [[UISegmentedControl alloc]initWithItems:items];
[segmentControl addTarget:self action:@selector(onClick:) forControlEvents:UIControlEventValueChanged];
segmentControl.selectedSegmentIndex = 0;
[segmentControl setFrame:CGRectMake(15, 70, AUCommonUIGetScreenWidth() - 30, 30)];
[self.view addSubview:segmentControl];

//label 用来显示 pickerView 选择的项目
self.textLabel = [[UILabel alloc]initWithFrame:CGRectMake(0, 110, 220, 50)];
self.textLabel.frame = CGRectOffset(self.textLabel.frame, (AUCommonUIGetScreenWidth()-
self.textLabel.frame.size.width)/2, 0);
self.textLabel.layer.cornerRadius = 12.f;
self.textLabel.lineBreakMode = NSLineBreakByWordWrapping;
self.textLabel.numberOfLines = 0;
self.textLabel.textAlignment = NSTextAlignmentCenter;
[self.view addSubview:self.textLabel];

//类方法创建的 pickerView
self.apPickerView = [AUDatePicker pickerViewWithTitle:nil];
self.apPickerView.delegate = self;
self.apPickerView.tag = 1000;
[self.view addSubview:self.apPickerView];
[self.apPickerView show];

//成员方法创建的 pickerView
_apPickerView2 = [[AUDatePicker alloc]initWithFrame:CGRectMake(0, 200, 200, 200) withTitle:nil];
_apPickerView2.delegate = self;
_apPickerView2.tag = 1001;
```

```
[self.view addSubview:_apPickerView2];

//时间选择器 1
self.apPickerView3 = [AUDatePicker pickerViewWithTitle:@"请选择时间"];
self.apPickerView3.tag = 1002;
self.apPickerView3.isDatePicker = YES;
NSDate * curretntDate = [NSDate date];
NSDate * minxDate = [NSDate dateWithTimeInterval:-(3600*24*3000) sinceDate:curretntDate];
NSDate * maxDate = [NSDate dateWithTimeInterval:3600*24*3000 sinceDate:curretntDate];
[self.apPickerView3 setTimeDateminDate:minxDate MaxDate:maxDate];
[self.apPickerView3 setCurrentDate:curretntDate];
[self.view addSubview:self.apPickerView3];

//时间选择器 2
self.apPickerView4 = [AUDatePicker pickerViewWithTitle:@"请选择时间"];
self.apPickerView4.tag = 1003;
self.apPickerView4.isDatePicker = YES;
[self.apPickerView4 setTimeDateminDate:minxDate MaxDate:maxDate];
[self.apPickerView4 setCurrentDate:curretntDate];
NSDate * selectDate =[NSDate dateWithTimeInterval:3600*24*888 sinceDate:curretntDate];
[self.apPickerView4 setAUDatePickerDate:selectDate animated:NO];
[self.view addSubview:self.apPickerView4];


// self.navigationItem.rightBarButtonItem = [APUtil getBarButtonWithTitle:RightBarButtonTitle target:self];
}

- (void)didReceiveMemoryWarning
{
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

#pragma mark - Button onClick
- (void)onBarButtonClick:(id)sender
{
}

- (void)onClick:(id)sender
{
[self.apPickerView hide];
[self.apPickerView2 hide];
[self.apPickerView3 hide];
[self.apPickerView4 hide];
UISegmentedControl* segmentControl = (UISegmentedControl*)sender;

switch (segmentControl.selectedSegmentIndex) {
case 0:
[self.apPickerView show];
break;
case 1:

[self.apPickerView2 show];
break;
case 2:
```

```
[self.apPickerView3 show];
break;
case 3:

[self.apPickerView4 show];
break;

default:
break;
}
}


#pragma APPickerDelegate delegate
- (void)cancelPickerView:(AUDatePicker *)pickerView
{
switch (pickerView.tag) {
case 1000:
[self.apPickerView hide];
break;
case 1001:
[self.apPickerView2 hide];
break;
case 1002:
[self.apPickerView3 hide];
break;
case 1003:
[self.apPickerView4 hide];
break;

default:
break;
}
[self.textLabel setText:@"点击“取消”按钮时的回调"];

}

- (void)selectedPickerView:(AUDatePicker *)pickerView
{
NSInteger index = [pickerView.pickerView selectedRowInComponent:0];
NSString *result = [self.yearArray objectAtIndex:index];

index = [pickerView.pickerView selectedRowInComponent:1];
result = [result stringByAppendingString:[NSString stringWithFormat:@"%@",[self.monthArray
objectAtIndex:index]]];

index = [pickerView.pickerView selectedRowInComponent:2];
result = [result stringByAppendingString:[NSString stringWithFormat:@"%@",[self.nameArray
objectAtIndex:index]]];

[self.textLabel setText:result];
}

#pragma UIPickerView delegate
- (NSString *)pickerView:(UIPickerView *)pickerView titleForRow:(NSInteger)row
forComponent:(NSInteger)component
```

```
{
if (component == 0) {
return [self.yearArray objectAtIndex:row];
} else if (component == 1){
return [self.monthArray objectAtIndex:row];
} else {
return [self.nameArray objectAtIndex:row];
}
}

- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView
{
return 3;
}

- (NSInteger)pickerView:(UIPickerView *)pickerView numberOfRowsInComponent:(NSInteger)component
{
if (component == 0) {
return [self.yearArray count];
} else if (component == 1){
return [self.monthArray count];
} else {
return [self.nameArray count];
}
}

@end
```

5.5.3 菜单组件

需要把原来的 APExtUI frameWork 中的（APNavPopview、APNavItemView）修改 为 AUFloatMenu 和 AUNavItemView，BEEBarPopMenu 也需要替换。

效果图

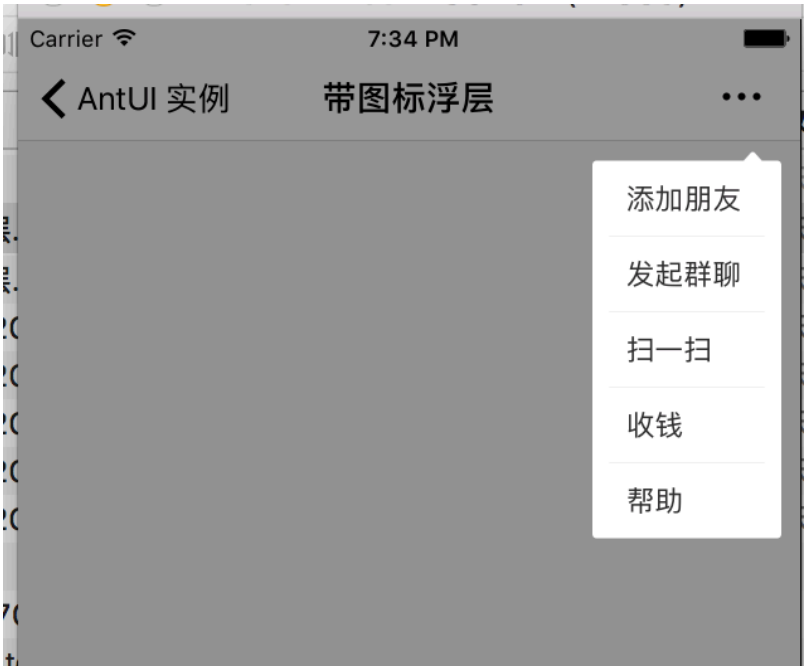
带红点浮层


AntUI 实例
带红点浮层
添加朋友 1
发起群聊 10
扫一扫
收钱 5
帮助

带图标浮层


Carrier
7:34 PM
AntUI 实例
带图标浮层
添加朋友
发起群聊
扫一扫
收钱
帮助

纯文字浮层



接口说明

- **AUFloatMenu.h**

```
//
// AUFloatMenu.h
// AntUI
//

#import <UIKit/UIKit.h>
/**popview 消失的通知*/
static NSString * const APExtUIPopViewDissmissedNotification = @"APExtUIPopViewDissmissedNotification";



@class AUNavItemView;
/* !
@class AUFloatMenu
@abstract UIView
@discussion floatViewMenu 浮层
*/
@interface AUFloatMenu : UIView<UIGestureRecognizerDelegate>
@property(nonatomic, assign) CGFloat marginToRight; //白色 popview 距离屏幕右侧的距离，不设置时默认为 10

/**
* 创建浮动菜单视图
*
* @param position 浮动菜单在屏幕上展示的位置
* @param items 展示的内容数组，一般为 APNavItemView 对象
*
* @return 浮动菜单视图
*/
+(AUFloatMenu *)showAtPostion:(CGPoint)position items:(NSArray<AUNavItemView *> *)items;
```

```
/**
* 创建浮动菜单视图
*
* @param position 浮动菜单在屏幕上展示的位置
* @param orignY 浮动菜单在屏幕上y坐标值
* @param items 展示的内容数组，一般为 APNavItemView 对象
*
* @return 浮动菜单视图
*/
+(AUFloatMenu *)showAtPostion:(CGPoint)position startOrignY:(CGFloat)orignY items:(NSArray<AUNavItemView
*> *)items;

/**
* 浮动菜单消失接口方法
*/
-(void)dismiss;

/**
* 菜单点开后发 RPC 加载动态下发菜单项，RPC 完成后调 update，完成旧 view 移出，添加新 view 的过程
*/
- (void)updateWithItems:(NSArray<AUNavItemView*> *)items;

@end
```

- **AUNavItemView.h**

```
//
// AUNavItemView.h
// AntUI
//

#import <UIKit/UIKit.h>

typedef NS_ENUM(NSInteger, AUCurrentTabType) {
AUCurrentTabTypeHome = 0,
AUCurrentTabTypeKouBei,
AUCurrentTabTypeFriend,
AUCurrentTabTypeWealth
};
/* !
@class AUNavItemView
@abstract UIView
@discussion floatMenu 浮层中每栏的 view
*/
@interface AUNavItemView : UIView
/**
* title
*/
@property(nonatomic,strong)NSString *itemTitle;

@property(nonatomic,strong,readonly)UIFont *titleFont;

/**
* 正常状态
```

```
*/
@property(nonatomic,strong)UIImage *nomarlStateIconImage;

/*
* iconFont Name 如果是 iconFont 的话，则调用这个接口，不用调上面的接口
*/
@property(nonatomic,strong)NSString *nomarlStateIconFontName;

/**
* 如果设置了 widgetId，就不需要设置 badgeNumber
*/
@property(nonatomic,strong)NSString *badgeNumber;

/**
* widgetId
*/
@property(nonatomic, copy) NSString *widgetId;

/**
*VoiceOver 需要的提示的文案，默认是 itemTitle，如果没有设置 itemTitle，需要手动设置此属性来支持 VoiceOver
*/
@property(nonatomic,strong)NSString *voiceOverText;

@property(nonatomic,assign)BOOL isNavigationItem;

@property(nonatomic,assign,readonly)CGFloat touchEventMargin;

@property(nonatomic,assign)AUCurrentTabType currentTabType;

@property(nonatomic,assign,readonly)CGFloat marginBetweenIconTitle;

@property(nonatomic,assign,readonly)CGFloat marginBetweenLeftIcon;

@property(nonatomic,assign,readonly)CGFloat badgeViewWidth;

/**
* 子类需重写此方法，然后处理点击的事件
*/
- (void)onClicked;

/**
返回Iconview的size

@return size
*/
- (CGSize) iconViewSize;

@end
```

代码示例

- 带红点示例：

```
//
```

```
// APNavPopViewViewController.m
// UIDemo
//

#import"APNavPopViewViewController.h"
#import"AUUtils.h"
#import"AUNavItemView.h"
#import"AUFloatMenu.h"
#import"AntUIShellObject.h"
#import"AUIconView.h"

@interface APNavPopViewViewController ()

@end

@implementation APNavPopViewViewController

- (void)viewDidLoad {
[super viewDidLoad];
self.view.backgroundColor = RGB(0xF5F5F9);

self.navigationItem.title = @"带红点浮层";
UIBarButtonItem *rightItem = [[UIBarButtonItem alloc]initWithImage:[UIImage
imageNamed:@"APCommonUI_ForDemo.bundle/more.png"] style:UIBarButtonItemStylePlain target:self
action:@selector(onClick:)];
self.navigationItem.rightBarButtonItem = rightItem;
[[AURegisterManager shareInstance] registerAUObject:[[AntUIShellObject alloc] init]];
}

- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

- (void)onClick:(id)sender
{
NSMutableArray *array = [[NSMutableArray alloc]initWithCapacity:4];

NSArray *items = @[@"添加朋友",@"发起群聊",@"扫一扫",@"收钱",@"帮助"];
int i = 0;
for (NSString *typeName in items) {
AUNavItemView *item = [[AUNavItemView alloc]initWithFrame:CGRectMake(20, 0, 0, 40)];
item.itemTitle = typeName;
item.isNavigationItem = NO;
//支持iconfont
// item.nomarlStateIconFontName = kICONFONT_USER_ADD;
if (i == 0 ) {
item.badgeNumber = @"1";
UIImage *image = [UIImage imageNamed:@"ap_add_friend.png"];
item.nomarlStateIconImage = image;
} else if(i == 1) {
item.badgeNumber = @"10";
UIImage *image = [UIImage imageNamed:@"ap_group_talk.png"];
item.nomarlStateIconImage = image;
} else if(i == 2) {
item.badgeNumber = @"100";
```

```
UIImage *image = [UIImage imageNamed:@"ap_scan.png"];
item.nomarlStateIconImage = image;
} else if(i == 3) {
item.badgeNumber = @"5";
UIImage *image = [UIImage imageNamed:@"ap_qrcode.png"];
item.nomarlStateIconImage = image;
} else if(i == 4) {
UIImage *image = [UIImage imageNamed:@"ap_help.png"];
item.nomarlStateIconImage = image;
}
i++;

[array addObject:item];
}

[AUFloatMenu showAtPostion:CGPointMake(0, 0) startOrignY:70 items:array];
}

- (void)onBarButtonClick:(id)sender
{
}


@end
```

- 带图标浮层示例：

```
//
// APNavPopViewViewController.m
// UIDemo
//

#import"APNavPopViewNoneRedViewController.h"
#import"AUUtils.h"
#import"AUNavItemView.h"
#import"AUFloatMenu.h"
#import"AntUIShellObject.h"
#import"AUIconView.h"

@interface APNavPopViewNoneRedViewController ()

@end

@implementation APNavPopViewNoneRedViewController

- (void)viewDidLoad {
[super viewDidLoad];
self.view.backgroundColor = RGB(0xF5F5F9);

self.navigationItem.title = @"带图标浮层";
UIBarButtonItem *rightItem = [[UIBarButtonItem alloc]initWithImage:[UIImage
imageNamed:@"APCommonUI_ForDemo.bundle/more.png"] style:UIBarButtonItemStylePlain target:self
action:@selector(onClick:)];
self.navigationItem.rightBarButtonItem = rightItem;
[[AURegisterManager shareInstance] registerAUObject:[[AntUIShellObject alloc] init]];
```

```
}

- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

- (void)onClick:(id)sender
{
NSMutableArray *array = [[NSMutableArray alloc]initWithCapacity:4];

NSArray *items = @[@"添加朋友",@"发起群聊",@"扫一扫",@"收钱",@"帮助"];
int i = 0;
for (NSString *typeName in items) {
AUNavItemView *item = [[AUNavItemView alloc]initWithFrame:CGRectMake(20, 0, 0, 40)];
item.itemTitle = typeName;
item.isNavigationItem = NO;
//支持iconfont
// item.nomarlStateIconFontName = kICONFONT_USER_ADD;
if (i == 0 ) {
// item.badgeNumber = @"1";
UIImage *image = [UIImage imageNamed:@"ap_add_friend.png"];
item.nomarlStateIconImage = image;
} else if(i == 1) {
// item.badgeNumber = @"10";
UIImage *image = [UIImage imageNamed:@"ap_group_talk.png"];
item.nomarlStateIconImage = image;
} else if(i == 2) {
// item.badgeNumber = @"100";
UIImage *image = [UIImage imageNamed:@"ap_scan.png"];
item.nomarlStateIconImage = image;
} else if(i == 3) {
// item.badgeNumber = @"5";
UIImage *image = [UIImage imageNamed:@"ap_qrcode.png"];
item.nomarlStateIconImage = image;
} else if(i == 4) {
UIImage *image = [UIImage imageNamed:@"ap_help.png"];
item.nomarlStateIconImage = image;
}
i++;

[array addObject:item];
}

[AUFloatMenu showAtPostion:CGPointMake(0, 0) startOrignY:70 items:array];
}

- (void)onBarButtonClick:(id)sender
{
}


@end
```

- 纯文字浮层示例：

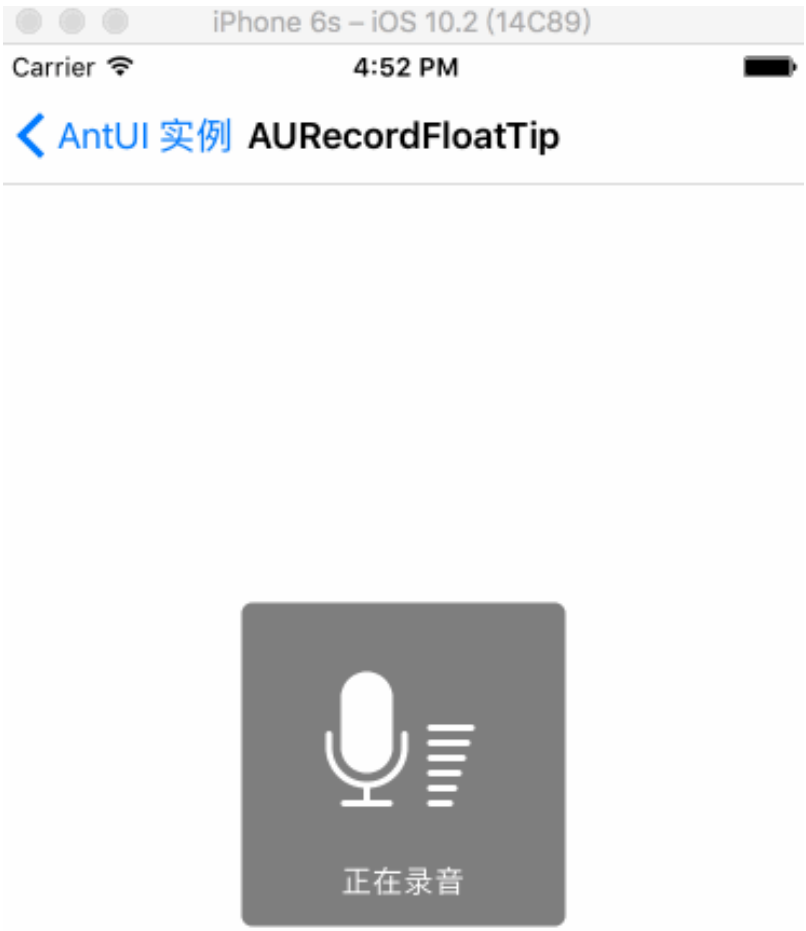
```
//
// APNavPopViewViewController.m
// UIDemo
//

#import"APNavPopViewOnlyViewController.h"
#import"AUUtils.h"
#import"AUNavItemView.h"
#import"AUFloatMenu.h"
#import"AntUIShellObject.h"
#import"AUIconView.h"

@interface APNavPopViewOnlyViewController ()

@end

@implementation APNavPopViewOnlyViewController

- (void)viewDidLoad {
[super viewDidLoad];
self.view.backgroundColor = RGB(0xF5F5F9);

self.navigationItem.title = @"纯文字浮层";
UIBarButtonItem *rightItem = [[UIBarButtonItem alloc]initWithImage:[UIImage
imageNamed:@"APCommonUI_ForDemo.bundle/more.png"] style:UIBarButtonItemStylePlain target:self
action:@selector(onClick:)];
self.navigationItem.rightBarButtonItem = rightItem;
[[AURegisterManager shareInstance] registerAUObject:[[AntUIShellObject alloc] init]];
}

- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

- (void)onClick:(id)sender
{
NSMutableArray *array = [[NSMutableArray alloc]initWithCapacity:4];

NSArray *items = @[@"添加朋友",@"发起群聊",@"扫一扫",@"收钱",@"帮助"];
int i = 0;
for (NSString *typeName in items) {
AUNavItemView *item = [[AUNavItemView alloc]initWithFrame:CGRectMake(0, 0, 0, 40)];
item.itemTitle = typeName;
item.isNavigationItem = NO;
//支持iconfont
// item.nomarlStateIconFontName = kICONFONT_USER_ADD;
// if (i == 0 ) {
//// item.badgeNumber = @"1";
// UIImage *image = [UIImage imageNamed:@"ap_add_friend.png"];
// item.nomarlStateIconImage = image;
// } else if(i == 1) {
//// item.badgeNumber = @"10";
// UIImage *image = [UIImage imageNamed:@"ap_group_talk.png"];
// item.nomarlStateIconImage = image;
```

```
// } else if(i == 2) {
//// item.badgeNumber = @"100";
// UIImage *image = [UIImage imageNamed:@"ap_scan.png"];
// item.nomarlStateIconImage = image;
// } else if(i == 3) {
//// item.badgeNumber = @"5";
// UIImage *image = [UIImage imageNamed:@"ap_qrcode.png"];
// item.nomarlStateIconImage = image;
// } else if(i == 4) {
// UIImage *image = [UIImage imageNamed:@"ap_help.png"];
// item.nomarlStateIconImage = image;
// }
i++;

[array addObject:item];
}

[AUFloatMenu showAtPostion:CGPointMake(0, 0) startOrignY:70 items:array];
}

- (void)onBarButtonClick:(id)sender
{
}


@end
```

5.5.4 录音状态浮层

AURecordFloatTip 为显示 **正在录音** 状态的浮层，用于给予用户更直接的录音体验。

效果图



接口说明

```
@interface AURecordFloatTip : UIView

@property (nonatomic, strong) UILabel *messageLabel; // 录音提示语，默认值为“正在录音”

// 浮层展示
- (void)showRecodingInView:(UIView *)view;
```

```
// 浮层消失
- (void)dismissRecordView;


@end
```

代码示例

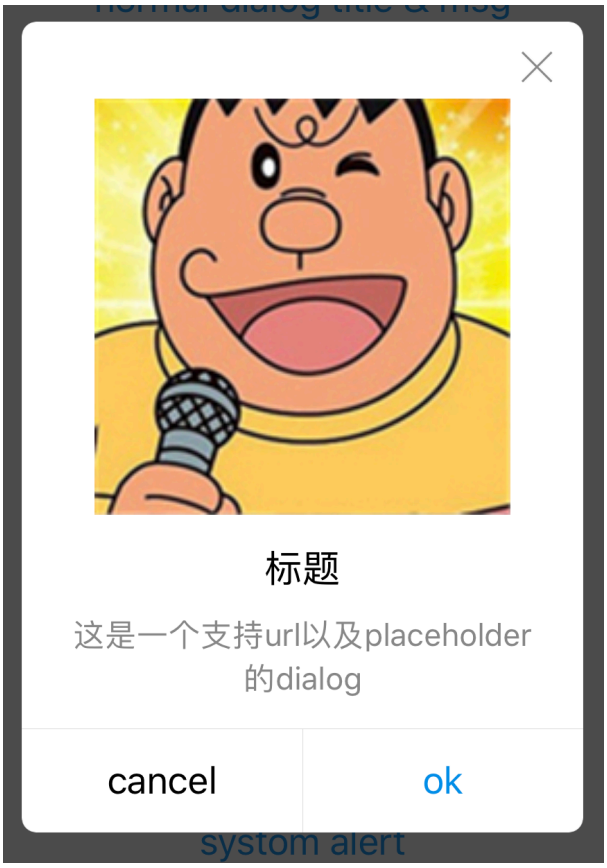
```
AURecordFloatTip *_tipView = [[AURecordFloatTip alloc] init];
[_tipView showRecodingInView:self.view];
```

5.5.5 图片弹窗

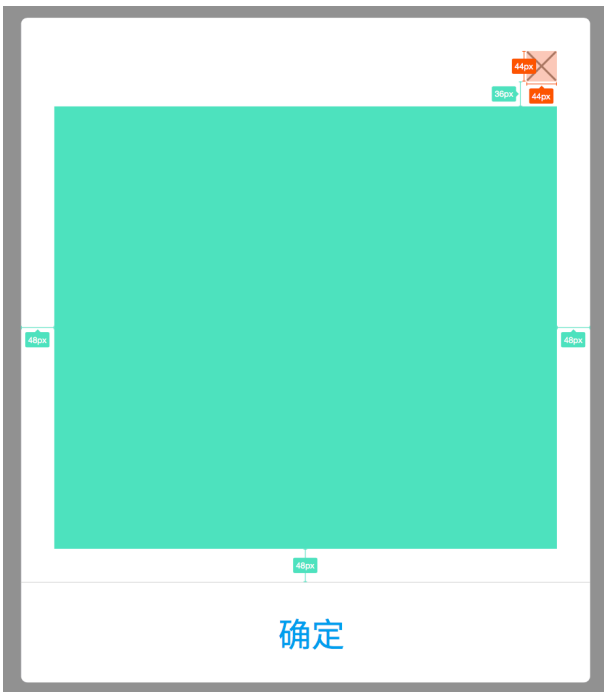
- AUImageDialog 为带图片样式的 Dialog，图片会做圆角处理，具体样式可自定义，如下面效果图展示。
- window 层级：self.windowLevel = UIWindowLevelAlert - 1

效果图

普通按钮样式



自定义样式



大图标样式

接口说明

```
// 按钮点击 Index 对应值
typedef NS_ENUM(NSInteger, AUImageDialogButtonIndex) {
AUImageDialogButtonIndex_Close = -2,
AUImageDialogButtonIndex_Link = -1,
```

```
AUImageDialogButtonIndex_Action = 0
};


/**
图片 Dialog，支持 UED 需求的一种特殊样式 Dialog，图片会做圆形显示。
有两种模式:
普通图片模式，添加按钮为普通按钮
行为按钮模式，可以添加一个行为按钮以及链接按钮，右上角会有 X 退出按钮样式。
两种模式不可添加对方模式的按钮，有 assert 校验。
*/
@interface AUImageDialog : AUDialogBaseView

/**
不带按钮标题的初始化方法。

@param image 图片
@param title 标题
@param message 消息内容
@param delegate 协议对象（遵循 AUDialogDelegate）
@return AUImageDialog 实例
*/
- (instancetype)initWithImage:(UIImage *)image
title:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate;

/**
带按钮标题的初始化方法。

@param image 图片
@param title 标题
@param message 消息内容
@param delegate 协议对象（遵循 AUDialogDelegate）
@param buttonTitle 按钮标题参数列表
@return AUImageDialog 实例
*/
- (instancetype)initWithImage:(UIImage *)image
title:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate
buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

/**
带蓝色行为按钮的初始化方法。

@param image 图片
@param title 标题
@param message 消息详情
@param delegate 协议对象（遵循 AUDialogDelegate）
@param actionTitle 行为按钮标题
@return AUImageDialog 实例
*/
- (instancetype)initWithImage:(UIImage *)image
title:(NSString *)title
message:(NSString *)message
```

```
delegate:(id<AUDialogDelegate>)delegate
actionButtonTitle:(NSString *)actionTitle;

/**
带蓝色行为按钮以及链接按钮的初始化方法。

@param image 图片
@param title 标题
@param message 消息详情
@param delegate 协议对象（遵循 AUDialogDelegate）
@param linkText 链接文本
@param actionTitle 行为按钮标题
@return AUImageDialog 实例
*/
- (instancetype)initWithImage:(UIImage *)image
title:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate
linkText:(NSString *)linkText
actionButtonTitle:(NSString *)actionTitle;

- (instancetype)init NS_UNAVAILABLE;

- (instancetype)initWithCustomView:(UIView *)customView; // 自定义内容区域，默认带右上角 X 按钮

/**
Dialog 展示方法。
*/
- (void)show;

/**
描述文本置为灰色，默认为 NO
*/
- (void)setGrayMessage:(BOOL)grayMessage;

/**
设置文本对齐

@param alignment 对齐参数
*/
- (void)setMessageAlignment:(NSTextAlignment)alignment;

/**
设置自定义图片尺寸, 宽度不许超过 Dialog 最大宽度 270, 默认 135x135
*/
- (void)configImageAreaSize:(CGSize)imageSize;

/**
添加普通按钮以及其回调方法（仅支持不带行为按钮和链接按钮情况下添加）。

@param buttonTitle 普通按钮标题
@param actionBlock 按钮回调
*/
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;

/**
```

```
添加行为按钮以及其回调方法。

@param actionTitle 行为按钮标题
@param actionBlock 行为按钮回调
*/
- (void)addActionButton:(NSString *)actionTitle actionBlock:(AUDialogActionBlock)actionBlock;

/**
添加链接按钮以及其回调方法。

@param linkText 链接文本
@param actionBlock 链接按钮回调
*/
- (void)addLinkButton:(NSString *)linkText actionBlock:(AUDialogActionBlock)actionBlock;

/**
隐藏右上角关闭按钮
*/
- (void)setCloseButtonHidden:(BOOL) hidden;
```

大图标样式接口说明

```
/**
图片 Dialog，支持 UED 需求的一种特殊样式 Dialog
* 样式：图片属于大图样式，图片高度固定为 312px，关闭按钮在图片右上角
* 关闭按钮是 iconfont，在大图样式下默认白色
*/

@interface AUImageDialog (largeImageStyle)

/**
不带按钮标题的初始化方法。

@param image 图片
@param title 标题
@param message 消息内容
@param delegate 协议对象（遵循 AUDialogDelegate）
@return AUImageDialog 实例
*/
- (instancetype)initWithLargeImage:(UIImage *)image
title:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate;

/**
* 设置右上角关闭按钮的色值，默认为白色
*/
- (void)resetCloseIconColor:(UIColor *)color;


@end
```

代码示例

- 普通按钮样式：

```
UIImage *image = [UIImage imageNamed:@"panghu.jpg"];
AUImageDialog *dialog = [[AUImageDialog alloc] initWithImage:image title:@"胖虎"message:@"严格匹配，规范中没有的不能放入标准控件中, 规范中未有但已经在多处使用的控件应放入候选控件集合中。另外不强制某一规范必须实现为单个控件，例如标题栏规范"delegate:self];
[dialog addButton:@"取消"actionBlock:nil];
[dialog addButton:@"确定"actionBlock:nil];
[dialog show];
```

- 自定义样式：

```
UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 240, 60)];
customView.backgroundColor = [UIColor greenColor];
AUImageDialog *dialog = [[AUImageDialog alloc] initWithCustomView:customView];
[dialog addButton:@"取消"actionBlock:nil];
[dialog addButton:@"确定"actionBlock:nil];
[dialog show];
```

- 大图标样式：

```
UIImage *image = [UIImage imageWithColor:[UIColor colorWithRGB:0xD8D8D8] size:CGSizeMake(100, 100)];
AUImageDialog *dialog = [[AUImageDialog alloc] initWithLargeImage:image title:@"标题单行"message:@"说明当前状态，提示用户解决方案，最好不要超过两行"delegate:self];
[dialog addButton:@"取消"actionBlock:nil];
[dialog addButton:@"确定"actionBlock:nil];
[dialog resetCloseIconColor:[UIColor redColor]];
[dialog show];
```

5.5.6 输入弹窗

- AUInputDialog 为带文本输入框的弹窗样式。
- window 层级：self.windowLevel = UIWindowLevelAlert - 1

效果图


中国联通
下午2:32
AntUI 实例
Dialog
normal dialog only title
标题
可能包含通知警报的声音图标和
按钮。这些可以
给朋友留言
取消
主操作
image dialog normal
image dialog url
我 你 在 这 是 一 不 有
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
空格
换行

接口说明

```
@interface AUInputDialog : AUDialogBaseView

/// 文本输入框
@property (nonatomic, strong, readonly) UITextField *textField;

/**
该实例是否在展示，适用于有指针指向该实例的情况。
如果有其他 dialog 盖住此 dialog，属性值也为 YES 不会发生变化。
*/
@property (nonatomic, assign, readonly) BOOL isDisplay;

/**
* 标题
*/
@property (nonatomic, strong) NSString *title;

/**
* 文本消息
*/
@property (nonatomic, strong) NSString *message;

/**
不带按钮标题的初始化方法。

@param title 标题
@param message 消息内容
@return AUInputDialog 实例
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message;


/**
```

```
AUInputDialog 实例化方法

@param title 标题
@param message 消息内容
@param placeholder 文本框的占位文字
@param delegate 代理对象
@param buttonTitle 按钮标题
@return AUInputDialog 实例
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message
placeholder:(NSString *)placeholder
delegate:(id<AUDialogDelegate>)delegate
buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

- (instancetype)initWithCustomView:(UIView *)customView; // 自定义内容区域

/// 禁用的初始化方法
- (instancetype)init NS_UNAVAILABLE;

/**
Dialog 展示方法。
*/
- (void)show;

/**
Dialog 消失方法, 如果监听 will/didDismissWithButtonIndex: 回调 index 值为默认的 0
*/
- (void)dismiss;

/**
隐藏 Dialog Window 上全部 dialog 视图
*/
+ (void)dismissAll;

/**
描述文本置为灰色，默认为 YES
*/
- (void)setGrayMessage:(BOOL)grayMessage;

/**
设置文本对齐

@param alignment 对齐参数
*/
- (void)setMessageAlignment:(NSTextAlignment)alignment;

/**
添加按钮以及其回调方法。

@param buttonTitle 按钮标题
@param actionBlock 按钮点击回调
*/
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;
```

代码示例

- 普通样式：

```
AUInputDialog *dialog = [[AUInputDialog alloc] initWithTitle:@"标题"message:@"可能包含通知警报的声音图标和按钮。这些可以"placeholder:@"给朋友留言"delegate:self buttonTitles:@"取消", @"主操作", nil];
[dialog show];
```

- 自定义样式：

```
UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 240, 60)];
customView.backgroundColor = [UIColor greenColor];
AUInputDialog *dialog = [[AUInputDialog alloc] initWithCustomView:customView];
[dialog addButton:@"取消"actionBlock:nil];
[dialog addButton:@"确定"actionBlock:nil];
[dialog show];
```

5.5.7 菜单组件

- AUPopMenu 与 AUFloatMenu 区别为：无底面蒙层，有外围边框，所有布局以居中的形式，分割线固定长度保持居中。
- 基本功能：业务控制向上或向下弹出，业务指定所弹出的位置。

效果图

接口说明

AUPopMenu.h

```
@protocol AUPopMenuDelegate <NSObject>

@optional
- (void)DidClickPopItemView:(AUPopItemModel *)viewModel;
```

```
@end

@interface AUPopMenu : UIView

@property (nonatomic, weak) id<AUPopMenuDelegate> delegate;

/* datas 是 AUPopItemModel 对象列表
* position 方向尖角所在位置
* superView 所在父 view
* isArchViewUp 方向角的朝向，默认朝下
*/
- (instancetype)initWithDatas:(NSArray *)datas
position:(CGPoint)position
superView:(UIView *)superView
isArchViewUp:(BOOL)isArchViewUp;

/* 默认带动画展示和隐藏
* position 指定方向角的起始位置
* superView 描述当前浮层展示在哪个父 view 上
*/
- (void)showMenu;

//
- (void)hideMenu;

@end
```

AUPopItemView.h

```

@interface AUPopItemView : AUPopItemBaseView

@property (nonatomic, strong) AUIconView *iconView; // 支持 iconfont 图标
//@property (nonatomic, strong) UIView *badgeView // 暂不支持红点

- (instancetype)initWithModel:(AUPopItemModel *)model position:(CGPoint )position;

@end
```

AUPopItemBaseView.h

```

//
@interface AUPopItemBaseView : UIControl

@property (nonatomic, strong) AULabel *titleLabel; //

@end
```

AUPopItemModel.h

```

// 对象模型
@interface AUPopItemModel : NSObject
```

```
@property (nonatomic, strong) NSString *titleString; // 主文案描述
@property (nonatomic, strong) id iconImage; // 左侧 icon，可以传 UIImage 对象或者 URL

@end
```

代码示例

```
_menu = [[AUPopMenu alloc] initWithDatas:array position:CGPointMake(CGRectGetMidX(button.frame),
CGRectGetMaxY(button.frame)+5) superView:self.view isArchViewUp:YES];
_menu.delegate = self;
[_menu showMenu];
```

5.5.8 弱提示组件

- AUToast 为 mPaaS 自定义 Toast 控件 。
- AUToast 迁移自 APCommonUI 的 APToast，请使用最新的 AUToast。

该组件主要包含两类 Toast：

- **普通 Toast**
- **模态 Toast**

模态 Toast 比普通 Toast 多了透明背景层，用户在背景层覆盖区域不可点击。

效果图


Carrier
5:40 PM
Back
Toast
AUToastIconNetFailure(系统繁忙)
系统繁忙
AUToastIconNetError(不传递文案)
网络无法连接
AUToastIconNetFailure(不传递文案)
网络不给力
AUToastIconSuccess
成功提示
AUToastIconFailure
失败提示
AUToastIconAlert
警示提示

接口说明

```
// log 输出函数声明，由外部设置
typedef void(*AUToastLogFunc)(NSString *tag, NSString *format, ...);
extern AUToastLogFunc g_ToastExternLogFunc; // log 输出函数全局变量，由外部设置
#define AUToastLog(fmt, ...) {if(g_ToastExternLogFunc)g_ToastExternLogFunc(@"@AUToast",fmt,##__VA_ARGS__);}


#define AUToast_Default_Duration 2.0 // AUToast 默认展示时间
#define AUToast_Strong_Duration 1.5 // AUToast 强提示展示时长
#define AUToast_Weak_Duration 1.0 // AUToast 弱提示展示时长

/**
* 添加新的 toastIcon 时，请向后添加，不要在中间插入，否则业务使用会有问题
*/
typedef enum{
AUToastIconNone = 0, // 无图标
AUToastIconSuccess, // 成功图标
AUToastIconFailure, // 失败图标
AUToastIconLoading, // 加载图标
AUToastIconNetFailure, // 网络失败
AUToastIconSecurityScan,// 安全扫描
AUToastIconNetError, // 网络错误，完全无法连接
AUToastIconProgress, // 加载图标，显示加载进度
AUToastIconAlert, // 警示图标
} AUToastIcon;

/**
* Toast控件
```

```
*/
@interface AUToast : UIView

@property (nonatomic, assign) CGFloat xOffset; // 设置相对父视图中心位置 X 轴方向的偏移量
@property (nonatomic, assign) CGFloat yOffset; // 设置相对父视图中心位置 Y 轴方向的偏移量

/*
* 模态显示提示，此时屏幕不响应用户操作（显示在 keywindow 上面），
* 需调用 dismissToast 方法使 Toast 消失
*
* @param text 显示文本，默认为 loading 加载
* @param logTag 日志标识
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentToastWithText:(NSString *)text
logTag:(NSString*)logTag;

/**
* 显示 Toast，需调用 dismissToast 方法使 Toast 消失
*
* @param superview 父视图
* @param text 显示文本
* @param logTag 日志标识
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentToastWithin:(UIView *)superview
text:(NSString *)text
logTag:(NSString*)logTag;

/**
* 显示 Toast，需调用 dismissToast 方法使 Toast 消失
*
* @param superview 父视图
* @param icon 图标类型
* @param text 显示文本
* @param logTag 日志标识
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
logTag:(NSString*)logTag;

/**
* 显示 Toast
*
* @param superview 父视图
* @param icon 图标类型
* @param text 显示文本
* @param duration 显示时长
* @param logTag 日志标识
*
* @return 返回显示的 Toast 对象
```

```
*/
+ (AUToast *)presentToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag;



/**
* 显示 Toast
*
* @param superview 要在其中显示 Toast 的视图
* @param icon 图标类型
* @param text 显示文本
* @param duration 显示时长
* @param logTag 日志标识
* @param completion Toast 自动消失后的回调
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag
completion:(void (^)())completion;


/**
* 显示 Toast
*
* @param superview 要在其中显示 Toast 的视图
* @param icon 图标类型
* @param text 显示文本
* @param duration 显示时长
* @param delay 延迟显示时长
* @param logTag 日志标识
* @param completion Toast 自动消失后的回调
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
delay:(NSTimeInterval)delay
logTag:(NSString*)logTag
completion:(void (^)())completion;



/*
* 模态 toast，需调用 dismissToast 方法使 Toast 消失
* 跟普通的 toast 区别是，会添加一个透明的背景层，防止用户屏幕点击
*
```

```
* @param superview 父视图
* @param text 显示文本
* @param logTag 日志标识
*
* @return 返回显示的Toast对象
*/
+ (AUToast *)presentModelToastWithin:(UIView *)superview
text:(NSString *)text
logTag:(NSString*)logTag;


/**
* 显示模态 Toast
* 跟普通的 toast 区别是，会添加一个透明的背景层，防止用户屏幕点击
*
* @param superview 要在其中显示 Toast 的视图
* @param icon 图标类型
* @param text 显示文本
* @param duration 显示时长
* @param logTag 日志标识
* @param completion Toast 自动消失后的回调
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentModalToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag
completion:(void (^)())completion;


/**
* 显示模态 Toast
* 跟普通的 toast 区别是，会添加一个透明的背景层，防止用户屏幕点击
*
* @param superview 要在其中显示 Toast 的视图
* @param icon 图标类型
* @param text 显示文本
* @param duration 显示时长
* @param delay 延迟显示时长
* @param logTag 日志标识
* @param completion Toast 自动消失后的回调
*
* @return 返回显示的 Toast 对象
*/
+ (AUToast *)presentModalToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
delay:(NSTimeInterval)delay
logTag:(NSString*)logTag
completion:(void (^)())completion;


/*
```

```
* 使 toast 消失
*/
- (void)dismissToast;

/**
* 设置进度的前缀文本，如果不设置，默认为“加载数据”
* 当 toast 类型为 AUToastIconProgress 时设置有效，否则忽略
*
* @param prefix 文本
*/
- (void)setProgressPrefix:(NSString*)prefix;

/**
* 显示当前加载数据的进度百分比
* 当 toast 类型为 AUToastIconProgress 时设置有效，否则忽略
*
* @param value 当前已加载的数据，范围为<0.0，1.0>
*
*/
- (void)setProgressText:(float)value;

@end
```

代码示例

```
[AUToast presentToastWithin:self.view withIcon:AUToastIconNetFailure text:@"系统繁忙"logTag:@"demo"];
[AUToast presentToastWithin:self.view withIcon:AUToastIconSuccess text:@"成功提示"logTag:@"demo"];
[AUToast presentToastWithin:self.view withIcon:AUToastIconFailure text:@"失败提示"logTag:@"demo"];
[AUToast presentToastWithin:self.view withIcon:AUToastIconAlert text:@"警示提示"logTag:@"demo"];

// 加载中
[AUToast presentToastWithin:self.view withIcon:AUToastIconLoading text:nil logTag:@"demo"];

// 显示进度场景
AUToast *toast = [AUToast presentToastWithin:self.view withIcon:AUToastIconProgress text:@"加载中
"logTag:@"demo"];
toast.origin = point;
[toast setProgressPrefix:@"~~~"];
[toast setProgressText:0.5];

// 模态Toast
[AUToast presentModalToastWithin:weakSelf.view withIcon:AUToastIconLoading text:@"模态toast，最长文案有这么
多，算不算多啊（三秒后消失）"duration:3 logTag:@"demo"completion:NULL];
[AUToast presentModalToastWithin:weakSelf.view withIcon:AUToastIconLoading text:@"模态toast，最长文案有这么
多，算不算多啊（三秒后消失）"duration:3 delay:2 logTag:@"demo"completion:NULL];
```

5.5.9 卡片菜单

在 iOS 中，beeviews:BEEPopMenuView 需要替换为 AUCardMenu.h。

效果图

弹出菜单/多行组合样式

弹出菜单/按压效果

弹出菜单/双行

弹出菜单+选择按钮

接口定义

- **AUCardMenu.h**

```
//
// AUCardMenu.h
// AntUI
//


@class AUMultiStyleCellView;
@class AUWindow;
/* !
@class AUCardMenu
@abstract AUWindow
@discussion 带 蒙层 + 方向角 的弹出菜单
*/

@interface AUCardMenu : AUWindow
{

}

/**
* 若需要响应点击事件，需要对 cellView 实现 AUMultiStyleCellDelegate 协议
* 在自己的 viewcontroller 中赋值 popMenuView.cellView.delegate = self;
*/
@property (nonatomic, strong) AUMultiStyleCellView *cellView;

/**
* 初始化方法（强烈推荐此方法）
*
* @param data 数组存放对象模型 CellDataModel
* @param location 方向角的基准点
* @param offset 方向角相对基准点的偏移量
*
* @return self
*/
```

```
- (instancetype)initWithData:(NSArray *)data
location:(CGPoint)location
offset:(CGFloat)offset;

/**
* 展示弹出菜单
*
* @param superView PopMenuView的superView
*/
- (void)showPopMenu:(UIView *)superView;

// 隐藏弹出菜单，最好在 dealloc 方法里也调用
- (void)hidePopMenu;

// 注意：带动画方式的展示或消失必须成对使用，即：动画出现必须动画消失，非动画出现必须非动画消失

// 带有动画方式的展示菜单
- (void)showPopMenu:(UIView *)superView animation:(BOOL) isAnimation;

// 带动画方式的消失菜单
- (void)hidePopMenuWithAnimation:(BOOL)isAnimation;

@end
```

- **AUCellDataModel.h**

```
//
// AUCellDataModel.h
// AntUI
//

#import <Foundation/Foundation.h>

/* !
@class AUMultiStyleCellView
@abstract UIView
@discussion menu中的子view
*/

@interface AUCellDataModel : NSObject

@property (nonatomic, strong) NSString *iconUrl;
@property (nonatomic, strong) NSString *titleText;
@property (nonatomic, strong) NSString *descText;
@property (nonatomic, strong) NSString *checkMarkUrl; // 对勾
@property (nonatomic, strong) NSString *indicatorUrl; // 右指示箭头
@property (nonatomic, strong) NSArray *buttonsArray; // NSArray<NSString>
@property (nonatomic, strong) NSDictionary *extendDic; // 给业务方使用的扩展字段
@property (nonatomic, assign) BOOL selectedState; // 当前model选中状态，默认为 NO，即不选中

@end
```

- **AUMultiStyleCellView.h**

```
//
// AUMultiStyleCellView.h
// AntUI
//

#import <UIKit/UIKit.h>
#import"AUCellDataModel.h"

@class AUMultiStyleCellView;
@protocol AUMultiStyleCellDelegate <NSObject>

@optional
/**
* 点击事件回调
*
* @param dataModel 点击的 view 对应的数据模型
* @param indexPath 点击的 view 在 CellDataModel 中的下标（若 CellDataModel.buttonsArray == nil，则 row 默认取值为 -1）
*/
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)indexPath;
- (void)DidClickCellButton:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)indexPath;
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)indexPath
cellView:(AUMultiStyleCellView *)cellView;
@end


/**
* 融合多样式的cellview
* 1. 图标 + 主标题
* 2. 图标 + 主标题 + 位于主标题下方的副标题
* 3. 图标 + 主标题 + 多行多列的带边框按钮控件
*/

@interface AUMultiStyleCellView : UIView

@property (nonatomic, weak) id<AUMultiStyleCellDelegate> delegate;
@property (nonatomic, strong) NSArray *cellDataArray;

// 如果 cellDataArray 为空等同于调用 initWithFrame 方法
- (instancetype)initWithFrame:(CGRect)frame
cellDataArray:(NSArray *)cellDataArray
isUpward:(BOOL)isUpward;

// 提供处理当前 cellView 选中与否的状态
- (void)updateSelectedState;


@end
```

代码示例

```
//
// cardMenuController.m
```

```
// AntUI
//

#import"cardMenuController.h"
#import"AUCardMenu.h"
#import"AUCellDataModel.h"
#import"AUMultiStyleCellView.h"

@interface cardMenuController ()<AUMultiStyleCellDelegate>

@property (nonatomic,strong) AUCardMenu * popMenuView;

@end

@implementation cardMenuController

- (void)viewDidLoad {
[super viewDidLoad];
// Do any additional setup after loading the view.
self.view.backgroundColor = RGB(0xF5F5F9);
UIButton * button = [UIButton buttonWithType:UIButtonTypeCustom];
[button setFrame:CGRectMake(0, 100, self.view.width, 100)];
[button setTitle:@"弹出菜单/多行组合样式"forState:UIControlStateNormal];
[button setTitleColor:RGB(0x888888) forState:UIControlStateNormal];
[button addTarget:self
action:@selector(handleButton:)
forControlEvents:UIControlEventTouchUpInside];
[button.titleLabel setTextAlignment:NSTextAlignmentLeft];
[button.titleLabel setFont:[UIFont systemFontOfSize:14]];
[button setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
[button setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];

[self.view addSubview:button];

UIButton * button2 = [UIButton buttonWithType:UIButtonTypeCustom];
[button2 setFrame:CGRectMake(0, 220, self.view.width, 100)];
[button2 setTitle:@"弹出菜单/按压效果"forState:UIControlStateNormal];
[button2 addTarget:self
action:@selector(handleButton2:)
forControlEvents:UIControlEventTouchUpInside];
[button2.titleLabel setTextAlignment:NSTextAlignmentLeft];
[button2 setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
[button2 setContentMode:UIViewContentModeLeft];
[button2 setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];

[button2 setTitleColor:RGB(0x888888) forState:UIControlStateNormal];
[button2.titleLabel setFont:[UIFont systemFontOfSize:14]];

[self.view addSubview:button2];

UIButton * button3 = [UIButton buttonWithType:UIButtonTypeCustom];
[button3 setFrame:CGRectMake(0, 320, self.view.width, 100)];
[button3 setTitle:@"弹出菜单/双行"forState:UIControlStateNormal];
[button3 addTarget:self
action:@selector(handleButton3:)
forControlEvents:UIControlEventTouchUpInside];
```

```
[button3.titleLabel setTextAlignment:NSTextAlignmentLeft];
[button3 setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
[button3 setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];
[button3.titleLabel setFont:[UIFont systemFontOfSize:14]];


[button3 setTitleColor:RGB(0x888888) forState:UIControlStateNormal];

[self.view addSubview:button3];


UIButton * button4 = [UIButton buttonWithType:UIButtonTypeCustom];
[button4 setFrame:CGRectMake(0, 420, self.view.width, 100)];
[button4 setTitle:@"弹出菜单+选择按钮"forState:UIControlStateNormal];
[button4 addTarget:self
action:@selector(handleButton4:)
forControlEvents:UIControlEventTouchUpInside];
[button4.titleLabel setTextAlignment:NSTextAlignmentLeft];
[button4 setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
[button4 setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];
[button4.titleLabel setFont:[UIFont systemFontOfSize:14]];

[button4 setTitleColor:RGB(0x888888) forState:UIControlStateNormal];

[self.view addSubview:button4];

}

- (void)handleButton4:(UIButton *)button
{
AUCellDataModel * model = [[AUCellDataModel alloc] init];
model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
model.titleText = @"我不感兴趣";
model.buttonsArray = @[@"过时",@"看过了",@"质量差"];
model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};


AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model] location:CGPointMake(button.width - 20,
button.centerY) offset:13];
tmpView.cellView.delegate=self;
[tmpView showPopMenu:button animation:YES];
self.popMenuView=tmpView;

}

- (void)handleButton3:(UIButton *)button
{
AUCellDataModel * model = [[AUCellDataModel alloc] init];
model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_ignore.png";
model.titleText = @"忽略";
// model.buttonsArray = @[@"你好",@"口吃吗",@"我不饿",@"你好吗",@"我很好"];
model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};

AUCellDataModel * model4 = [[AUCellDataModel alloc] init];
```

```
model4.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_reject.png";
model4.titleText = @"不再接受此类消息";
model4.descText = @"减少此类消息的接收";
model4.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};
AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model,model4] location:CGPointMake(button.width -
20, button.centerY) offset:13];
tmpView.cellView.delegate=self;
[tmpView showPopMenu:button animation:YES];
self.popMenuView=tmpView;

}

- (void)handleButton2:(UIButton *)button
{
AUCellDataModel * model = [[AUCellDataModel alloc] init];
model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_ignore.png";
model.titleText = @"忽略";
// model.buttonsArray = @[@"你好",@"口吃吗",@"我不饿",@"你好吗",@"我很好"];
model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};
AUCellDataModel * model2 = [[AUCellDataModel alloc] init];
model2.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
model2.titleText = @"我不感兴趣";
model2.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};
model2.highlightState = YES;
AUCellDataModel * model3 = [[AUCellDataModel alloc] init];
model3.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_inform.png";
model3.titleText = @"投诉";
model3.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};
AUCellDataModel * model4 = [[AUCellDataModel alloc] init];
model4.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_reject.png";
model4.titleText = @"不再接受此类消息";
model4.descText = @"减少此类消息的接收";
model4.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};
AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model,model2,model3,model4]
location:CGPointMake(button.width - 20, button.centerY) offset:13];
tmpView.cellView.delegate=self;
[tmpView showPopMenu:button animation:YES];
self.popMenuView=tmpView;

}
- (void)handleButton:(UIButton *)button
{
AUCellDataModel * model = [[AUCellDataModel alloc] init];
model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_ignore.png";
model.titleText = @"忽略";
// model.buttonsArray = @[@"你好",@"口吃吗",@"我不饿",@"你好吗",@"我很好"];
model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard":@""};
AUCellDataModel * model2 = [[AUCellDataModel alloc] init];
model2.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
model2.titleText = @"我不感兴趣";
```

```
model2.extendDic =
@{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069500009026 88",@"CCard":@""};
AUCellDataModel * model3 = [[AUCellDataModel alloc] init];
model3.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_inform.png";
model3.titleText = @"投诉";
model3.extendDic =
@{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069500009026 88",@"CCard":@""};
AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model,model2,model3]
location:CGPointMake(button.width - 20, button.centerY) offset:13];
tmpView.cellView.delegate=self;
[tmpView showPopMenu:button animation:YES];
self.popMenuView=tmpView;


}


- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

- (void)hidePopMenu
{
if (self.popMenuView) {
[self.popMenuView hidePopMenuWithAnimation:YES];
self.popMenuView.cellView.delegate = nil;
self.popMenuView = nil;

}
}


#pragma mark --- AUMultiStyleCellDelegate
/**
* 点击事件回调
*
* @param dataModel 点击的 view 对应的数据模型
* @param indexPath 点击的 view 在 CellDataModel 中的下标（若 CellDataModel.buttonsArray == nil，则 row 默认取值为 -1）
*/
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)indexPath
{
[self hidePopMenu];
}
- (void)DidClickCellButton:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)indexPath
{
[self hidePopMenu];
}
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)indexPath
cellView:(AUMultiStyleCellView *)cellView
{
[self hidePopMenu];
}
```

```
- (void)dealloc
{
self.popMenuView = nil;

}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
// Get the new view controller using [segue destinationViewController].
// Pass the selected object to the new view controller.
}
*/

@end
```

5.5.10 结果弹窗

AUOperationResultDialog 为带结果图片样式的 Dialog，图片默认大小为 90×58（单位：px），具体样式由 UED 提出需求，见效果图。

> **注意**：这类弹窗仅限于社交和收银台使用，其他业务请参考 AUImageDialog 。

- window 层级：self.windowLevel = UIWindowLevelAlert - 1。

效果图

接口说明

```
@interface AUOperationResultDialog : AUDialogBaseView

/**
该实例是否在展示，适用于有指针指向该实例的情况。
如果有其他 dialog 盖住此 dialog，属性值也为 YES 不会发生变化。
*/
@property (nonatomic, assign, readonly) BOOL isDisplay;

/**
* 描述文案
*/
@property (nonatomic, strong) NSString *describe;

/**
不带按钮标题的初始化方法。

@param image 图片
@param describe 消息描述
@param delegate 协议对象（遵循 AUDialogDelegate）
@return AUImageDialog 实例
*/
- (instancetype)initWithImage:(UIImage *)image
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate;

/**
带按钮标题的初始化方法。

@param image 图片
@param describe 消息描述
@param delegate 协议对象（遵循 AUDialogDelegate）
@param buttonTitle 按钮标题参数列表
@return AUImageDialog 实例
*/
- (instancetype)initWithImage:(UIImage *)image
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate
buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

/**
带下载链接的

@param imageUrl 图片链接
@param placeholder 占位图片
@param describe 消息描述
@param delegate 协议对象（遵循 AUDialogDelegate）
@return AUImageDialog 实例
*/
- (instancetype)initWithImageUrl:(NSString *)imageUrl
placeholder:(UIImage *)placeholder
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate;
```

```
/// 禁用的初始化方法
- (instancetype)init NS_UNAVAILABLE;

/**
Dialog 展示方法。
*/
- (void)show;

/**
Dialog 消失方法, 如果监听 will/didDismissWithButtonIndex: 回调 index 值为默认的 0
*/
- (void)dismiss;

/**
隐藏 Dialog Window 上全部 dialog 视图
*/
+ (void)dismissAll;

/**
添加普通按钮以及其回调方法(仅支持不带行为按钮情况下添加)。

@param buttonTitle 普通按钮标题
@param actionBlock 按钮回调
*/
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;

@end
```

代码示例

```
UIImage *image = [UIImage imageNamed:@"panghu.jpg"];
AUOperationResultDialog *dialog = [[AUOperationResultDialog alloc] initWithImage:image message:@"已发送
"delegate:self];
[dialog addButton:@"返回手机淘宝"actionBlock:nil];
[dialog addButton:@"留在支付宝"actionBlock:nil];
[dialog show];
```

5.5.11 级联选择器

AUCascadePicker 为多级级联选择器控件，最多支持三级。

效果图

接口说明

```
// 设置选择器的选中项
@interface AUCascadePickerSelectedRowItem : NSObject

@property (nonatomic, strong) NSString *selectedLeftTitle; // 当前第一子列表选中的 title
@property (nonatomic, strong) NSString *selectedMiddleTitle; // 当前第二子列表选中的 title
@property (nonatomic, strong) NSString *selectedRightTitle; // 当前第三子列表选中的 title

@end

@interface AUCascadePickerRowItemModel : NSObject

@property (nonatomic, strong) NSString *rowTitle;
@property (nonatomic, strong) NSArray<AUCascadePickerRowItemModel *> *rowSubList;

@end

// 联动效果所需要的数据模型
@interface AUCascadePickerModel : NSObject

@property (nonatomic,strong) AUCascadePickerSelectedRowItem *preSelected; // 业务方传进来的选中项
@property (nonatomic, strong) AUCascadePickerSelectedRowItem *selectedItem; // 当前组件内自行记录的选中数据列表
@property (nonatomic, strong) NSArray<AUCascadePickerRowItemModel *> *dataList ; // 数据列表
@property (nonatomic, strong) NSString *title; // 选择器标题

@end


@interface AUCascadePicker : AUPickerBaseView <UIPickerViewDataSource, UIPickerViewDelegate>
```

```
@property (nonatomic, strong) AUCascadePickerModel *dataModel;
@property (nonatomic, assign) NSInteger numberOfComponents;
@property (nonatomic, weak) id <AUCascadePickerDelegate> linkageDelegate;

- (instancetype)initWithPickerModel:(AUCascadePickerModel *)model;

@end

// 顶部“取消” & “完成” 的回调
@protocol AUCascadePickerDelegate <AUPickerBaseViewDelegate>
/*
* 点取消时回调
*/
- (void)cancelPickerView:(AUCustomDatePicker *)pickerView;

/*
* 点完成时回调，选中项可通过 selectedRowInComponent返回
*/
- (void)selectedPickerView:(AUCustomDatePicker *)pickerView

@end
```

JSAPI 说明

接口

antUIGetCascadePicker

使用

```
AlipayJSBridge.call('antUIGetCascadePicker',
{
title: 'nihao',//级联选择标题
selectedList:[{"name":"杭州市",subList:[{"name":"上城区"}]}],
list: [
{
name:"杭州市",//条目名称
subList: [
{
name:"西湖区",
subList: [
{
name:"古翠街道"
},
{
name:"文新街道"
}
]
},
{
name:"上城区",
subList: [
{
name:"延安街道"
```

```
},
{
name:"龙翔桥街道"
}
]
}
]//级联子数据列表
}
]//级联数据列表
},
function(result){
console.log(result);
});
```

入参

名称	类型	描述	必选	默认值	版本
title	string	级联控件标题	NO	—	10.1.2
selectedList	json	选中态，指定选中的子项，格式与入参一致 ([{“name”:”杭州市”,subList:[{“name”:”上城区”}]}])	NO	—	10.1.2
list	json	选择器数据列表	YES	—	10.1.2
name	string	条目名称，list 内的 name	YES	—	10.1.2
subList	json	子条目列表，list 内的 subList	NO	—	10.1.2
fn	function	选择完成后的回调函数	NO	—	10.1.2

出参

名称	类型	描述	版本
success	bool	是否选择完成，取消返回 false	10.1.2
result	json	选择的结果，如 [{“name”:”杭州市”,subList:[{“name”:”上城区”}]}]	10.1.2

代码示例

```
model = [[AULinkagePickerModel alloc] init];

NSMutableArray *modelList = [[NSMutableArray alloc] init];
for (int i=0; i<6; i++)
{
AULinkagePickerRowItemModel *item = [[AULinkagePickerRowItemModel alloc] init];
item.rowTitle = [NSString stringWithFormat:@"第一层的%d", i];
NSMutableArray *array = [[NSMutableArray alloc] init];
```

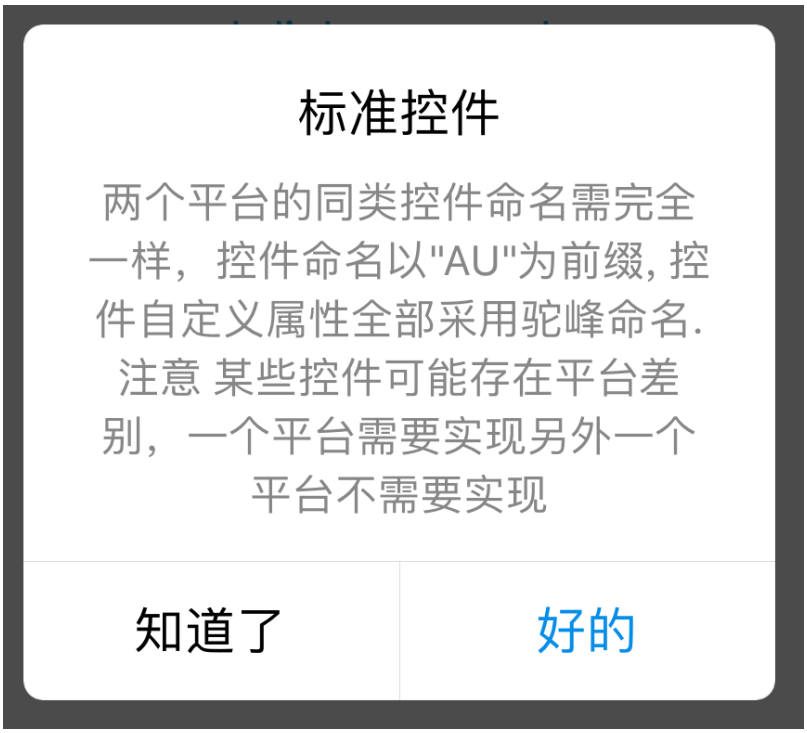
```
for (int j=0; j<7; j++)
{
if (i == 0)
{
break;
}
AULinkagePickerRowItemModel *item1 = [[AULinkagePickerRowItemModel alloc] init];
item1.rowTitle = [NSString stringWithFormat:@"第二层的%d", j];
NSMutableArray *array1 = [[NSMutableArray alloc] init];
for (int k=0; k<5; k++) {
AULinkagePickerRowItemModel *item2 = [[AULinkagePickerRowItemModel alloc] init];
item2.rowTitle = [NSString stringWithFormat:@"第三层的%d", k];
[array1 addObject:item2];
if (j == 1 || j== 2) {
break;
}
}
item1.rowSubList = array1;
[array addObject:item1];
if (i == 3 || i== 5) {
break;
}
}
item.rowSubList = array;
[modelList addObject:item];
}

model.dataList = modelList;


AULinkagePickerSelectedRowItem *item = [[AULinkagePickerSelectedRowItem alloc] init];
item.selectedLeftTitle = @"第一层的0";
item.selectedMiddleTitle = @"第二层的0";
item.selectedRightTitle = @"第三层的0";

model.selectedItem = item;

self.linkagePickerView = [[AULinkagePickerView alloc] initWithPickerModel:model];
self.linkagePickerView.linkageDelegate = self;
[self.linkagePickerView show];
```

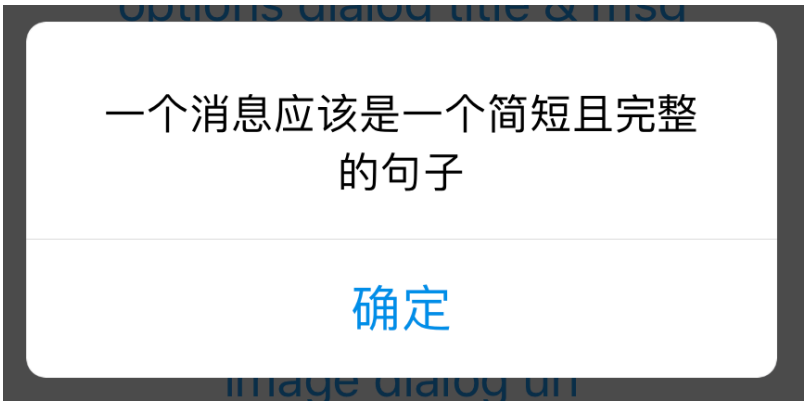
5.5.12 提示弹窗

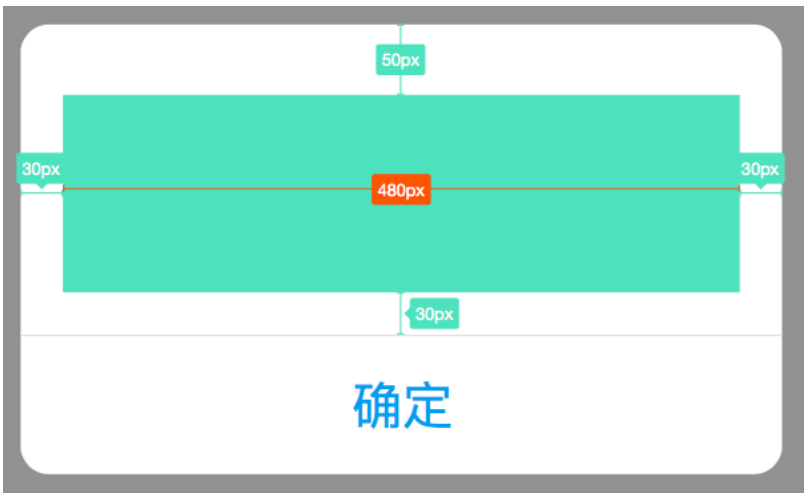
- AUNoticeDialog 为普通 Dialog 样式，参考自系统 AlertView 但是不带 blur 背景。
- Window 层级：self.windowLevel = UIWindowLevelAlert - 1。

效果图


标题
取消
确定


标准控件
两个平台的同类控件命名需完全一样，控件命名以"AU"为前缀，控件自定义属性全部采用驼峰命名.
注意 某些控件可能存在平台差别，一个平台需要实现另外一个平台不需要实现
知道了
好的


一个消息应该是一个简短且完整的句子
确定



接口说明

```
/**
普通 Dialog ，同系统样式不带 blur 背景
*/
@interface AUNoticeDialog : AUDialogBaseView

/**
不带按钮标题的初始化方法。

@param title 标题
@param message 消息内容
@return AUNoticeDialog 实例
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message;

/**
带按钮标题的初始化方法。

@param title 标题
@param message 消息内容
@param delegate 协议对象（遵循 AUDialogDelegate ）
@param buttonTitle 按钮标题列表
@return AUNoticeDialog 实例
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate
buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

- (instancetype)initWithCustomView:(UIView *)customView; // 自定义内容区域

- (instancetype)init NS_UNAVAILABLE;

/**
Dialog 展示方法。
*/
- (void)show;
```

```
/**
添加按钮以及其回调方法。

@param buttonTitle 按钮标题
@param actionBlock 按钮点击回调
*/
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;

/**
Dialog 消失方法,类似APAlertView的dismissWithClickedButtonIndex方法
*/
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex animated:(BOOL)animated;

/**
设置文本对齐
@param alignment 对齐参数
*/
- (void)setMessageAlignment:(NSTextAlignment)alignment;
```

全新接入

使用 block 添加 button 点击回调

```
AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithTitle:@"标题"message:@"内容"];
[dialog addButton:@"知道了"actionBlock:^{
NSLog(@"print pressed");
}];
[dialog addButton:@"好的"actionBlock:nil];
[dialog show];
```

使用 delegate 添加 button 点击回调

```
AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithTitle:@"标题"message:@"内容"delegate:delegate
buttonTitles:@"确定", nil];
[dialog show];

delegate协议为AUDialogDelegate（类似UIAlertViewDelegate）
```

简便方法接入

```
NS_INLINE AUNoticeDialog *AUNoticeDialogWithTitle(NSString *title)
NS_INLINE AUNoticeDialog *AUNoticeDialogWithTitleAndMessage(NSString *title, NSString *message)
```

APAlertView 与 UIAlertView 接入

以前主要为 APAlertView 和 UIAlertView，本节介绍如何更改为 AUNoticeDialog。

为了更简单的从 APAlertView 和 UIAlertView 接入 AUNoticeDialog，大部分接口均做了支持，因此大多数情况下只需要更改类名即可，具体如下：

- AUNoticeDialog 支持 APAlertView 的创建接口

```
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle
otherButtonTitles:(NSString *)otherButtonTitles, ... NS_REQUIRES_NIL_TERMINATION;
```

创建时，**更改类名即可**，只需将 [[APAlertView alloc] initWithxxxxxx] 改为 [[AUNoticeDialog alloc] initWithxxxxxx]。

- 使用如下方法创建 UIAlertView，**无需更改**，因为接口中已做了更改

```
NS_INLINE UIAlertView *UIAlertViewWithTitleAndMessage(NSString *title, NSString *message)
//
NS_INLINE UIAlertView *UIAlertViewWithTitle(NSString *title)
NS_INLINE UIAlertView *UIAlertViewWithMessage(NSString *message)
```

- 支持 APAlertView 的 addButtonWithTitle 接口，接入时 **无需更改**

```
- (NSInteger)addButtonWithTitle:(NSString *)title callback:(void (^)(int index, NSString *title))callback;

/**
@brief 添加取消 Button 和回调
@param title 按钮 title
@param callback 回调的 callback
*/
- (NSInteger)addCancelButtonWithTitle:(NSString *)title callback:(void (^)(int index, NSString *title))callback;

/**
@brief 添加 Button
@param title 按钮 title
*/
- (NSInteger)addButtonWithTitle:(NSString *)title;

/**
@brief 添加取消 button
@param title 按钮 title
*/
- (NSInteger)addCancelButtonWithTitle:(NSString *)title;

+(void)setBackgroundMode:(BOOL)isBackMode;
```

- 使用如下 UIAlertView 方法也 **无需更改**，AUNoticeDialog 有同名方法支持

```
/**
Dialog 消失方法，类似 APAlertView 的 dismissWithClickedButtonIndex 方法
*/
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex animated:(BOOL)animated
- (nullable NSString *)buttonTitleAtIndex:(NSInteger)buttonIndex;
/**
有多少个按钮（类似 APAlertView的numberOfButtons）
*/
```

```
@property(nonatomic,readonly) NSInteger numberOfButtons;

/**
取消按钮的 index ( 类似 APAlertView的cancelButtonIndex )
*/
@property(nonatomic) NSInteger cancelButtonIndex;
```

调用 APAlertView 的如下接口需要变更为其他方法，**只需更改方法名**：

将 showAlert 方法改为 show方法

如：[alertView showAlert] ==> [alertView show]

将 removeAllAlerviews 方法改为 dismissAll 方法

如：[APAlertView removeAllAlerviews] ==> [AUNoticeDialog dismissAll]

如果使用了 APAlertView 或者 UIAlertView 的输入框功能，请使用 AUInputDialog 替换，使用方法与 AUNoticeDialog 基本相同

注： 类文件为 AUInputDialog.h。

UIAlertController 接入

创建方法修改，如：

```
[UIAlertController alertControllerWithTitle:title message:message preferredStyle:UIAlertControllerStyleAlert]
修改为
[[AUNoticeDialog alloc] initWithTitle:@"标题"message:@"内容"]
```

添加button和事件修改：

```
[UIAlertAction actionWithTitle:title style:(UIAlertActionStyle)style handler:handler]
修改为
[dialog addButton:@"知道了"actionBlock:^{
NSLog(@"xxxx");
}]
```

代码示例

标准样式：

```
AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithTitle:@"标准控件"message:@"两个平台的同类控件命名需完全一样，控件命名以\"AU\"为前缀，控件自定义属性全部采用驼峰命名。注意：某些控件可能存在平台差别，一个平台需要实现另外一个平台不需要实现。"];
```

```
[dialog addButton:@"知道了"actionBlock:nil];
[dialog addButton:@"好的"actionBlock:nil];
[dialog show];
```

自定义样式：

```
UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 240, 60)];
customView.backgroundColor = [UIColor greenColor];

AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithCustomView:customView];
[dialog addButton:@"取消"actionBlock:nil];
[dialog addButton:@"确定"actionBlock:nil];
[dialog show];
```

5.5.13 自定义日期组件

AUCustomDatePicker 为自定义的日期选择控件，目前支持以下几种模式：

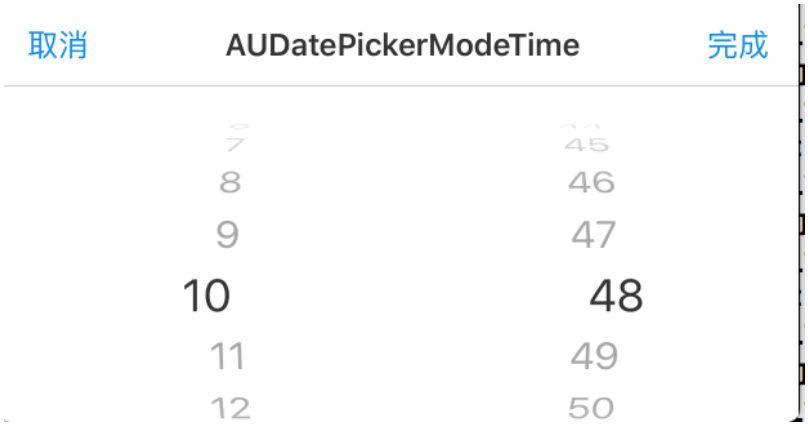
- AUDatePickerModeTime：小时/分，24小时制
- AUDatePickerModeDate：年/月/日
- AUDatePickerModeDateAndTime：月/日/星期/小时/分/，24小时制，

  注意：年是按照 minimumDate 定义，默认为 2000 年闰年，故存在2/29

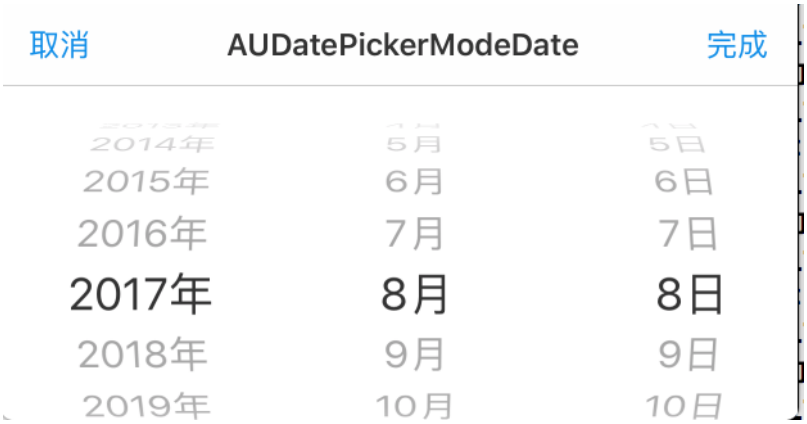
- AUDatePickerYear：年
- AUDatePickerYearMonth：年/月

效果图

AUDatePickerModeTime



AUDatePickerModeDate



AUDatePickerModeDateAndTime

AUDatePickerYear

AUDatePickerYearMonth

带自定义 BottomView

接口说明

AUCustomDatePicker.h

```
//自定义底部View
@property (nonatomic,strong) UIView *bottomView;


/**
* 创建 Picker，默认使用 AUDatePickerModeDate 模式
*
*/
+ (AUCustomDatePicker *)pickerViewWithTitle:(NSString *)title;

+ (AUCustomDatePicker *)pickerViewWithTitle:(NSString *)title pickerMode:(AUCustomDatePickerMode)mode;
```

```
/**
* 设定可选择的日期区间
@param minDate 最小时间，默认为 2000 年 1 月 1 日 00:00:00，闭
@param maxDate 最大时间，默认为 2050 年 12 月 31 日 23:59:59，闭
*/
- (void) setTimeDateminDate:(NSDate *)minDate MaxDate:(NSDate *)maxDate;



/**
@param currentDate 设置默认选中的时间
*/
- (void) setCurrentDate:(NSDate *) currentDate animated:(BOOL) animated;



/**
展示日期选择控件
*/
-(void) show;

/**
隐藏日期选择控件
*/
-(void) hide;
```

示例代码

创建：

```
self.apCustomDatePickerView = [AUCustomDatePicker
pickerViewWithTitle:@"AUDatePickerYearMonth"pickerMode:AUDatePickerYearMonth];

UIView *customBottomView = [[UIView alloc]initWithFrame:CGRectMake(0, 0,
AUCommonUIGetScreenWidth(), 40)];
customBottomView.backgroundColor = RGB(0x00AAEE);
self.apCustomDatePickerView.bottomView = customBottomView;

[self.apCustomDatePickerView setCurrentDate:[NSDate date] animated:NO];
self.apCustomDatePickerView.tag = 1004;
self.apCustomDatePickerView.delegate = self;
[self.view addSubview:self.apCustomDatePickerView];
```

展示 / 隐藏

```
[self.apCustomDatePickerView show];
[self.apCustomDatePickerView hide];
```

取值

```
- (void)cancelPickerView:(AUCustomDatePicker *)pickerView
{
[self.apCustomDatePickerView hide];
}

- (void)selectedPickerView:(AUCustomDatePicker *)pickerView
{

NSDate *selectedDate = picker.selectedDate;

NSDateFormatter *formatter = [[NSDateFormatter alloc]init];
formatter.dateFormat = @"YYYY-MM-dd HH:mm:ss";

[self.textLabel setText:[formatter stringFromDate:selectedDate]];

[pickerView hide];

}
```

5.6 加载组件

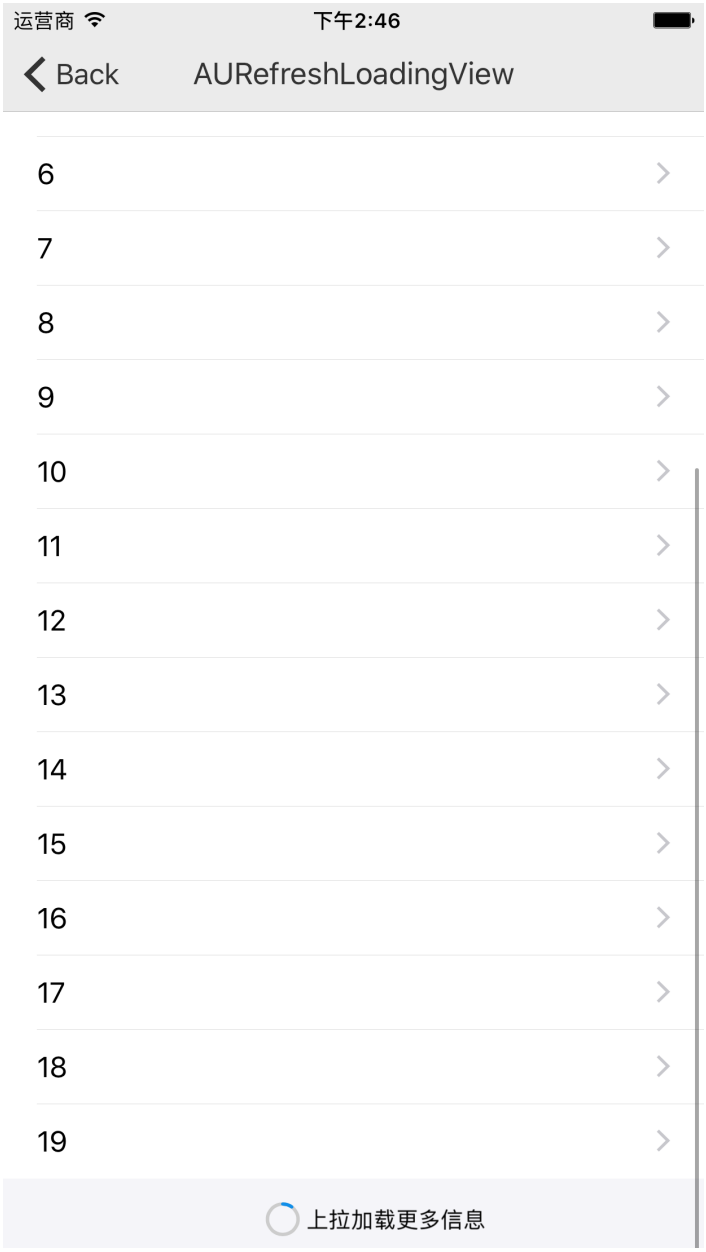
5.6.1 上拉刷新控件

- 需要对 iOS 中的 BeeViews：BEESmilePullRefreshView(BEESmileAnimationView、BEESmileStaticView) 首页单独刷新控件，由首页负责。

  以下非业务定制化的控件，都需要切换为下拉（AUPullLoadingView）或上拉（AUDragLoadingView）组件。

  CommonUI：ODRefreshControl、APCircleRefreshControl、EGORefreshTableHeaderView、APNextPagePullView

效果图



接口说明

AUDragLoadingView.h

```
//
// AUDragLoadingView.h
// AntUI
//

#import <AntUI/AntUI.h>

@interface AUDragLoadingView : AUPullLoadingView

@end
```

AUPullLoadingView.h

见 下拉刷新控件 。

代码示例

```
//
// APRefreshTableViewController.m
// UIDemo
//

#import"APRefreshTableViewController.h"
@interface APRefreshTableViewController ()
{
BOOL _headerReloading;
BOOL _footerReloading;
BOOL _isHeader;
}
@property(nonatomic,strong)AUPullLoadingView *refreshHeaderView;
@property(nonatomic,strong)AUDragLoadingView *refreshFooterView;
@property(nonatomic, strong) UITableView *tableView;
@property(nonatomic, strong) NSMutableArray* listArray;


@end

@implementation APRefreshTableViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
if (self) {
// Custom initialization
NSArray *array = @[@"0",
@"1",
@"2",
@"3",
@"4",
@"5",
@"6",
@"7",
@"8",
@"9",
@"10",
@"11",
@"12",
@"13",
@"14",
@"15",
@"16",
@"17",
@"18",
@"19"];
self.listArray = [NSMutableArray arrayWithArray:array];
```

```
}
return self;
}

- (void)viewDidLoad
{
[super viewDidLoad];
// Do any additional setup after loading the view.
self.edgesForExtendedLayout = UIRectEdgeNone;
// self.navigationItem.rightBarButtonItem = [APUtil getBarButtonWithTitle:RightBarButtonTitle target:self];

self.tableView = [[UITableView alloc]initWithFrame:self.view.bounds style:UITableViewStylePlain];
self.tableView.dataSource = self;
self.tableView.delegate = self;
self.tableView.backgroundColor = [UIColor colorWithRGB:0xf5f5f9];
self.tableView.separatorColor = [UIColor colorWithRGB:0xdddddd];
[self.view addSubview:self.tableView];

if (_refreshHeaderView == nil) {

AUPullLoadingView *view = [[AUPullLoadingView alloc] initWithFrame:CGRectMake(0.0f, 0.0f -
self.tableView.bounds.size.height, self.view.frame.size.width, self.tableView.bounds.size.height)];
view.delegate = self;
[view ShowLastPullDate:YES];
[view ShowStatusLabel:NO];

[self.tableView addSubview:view];
_refreshHeaderView = view;
}
[_refreshHeaderView refreshLastUpdatedDate];

if (_refreshFooterView == nil) {
AUDragLoadingView *view = [[AUDragLoadingView alloc] initWithFrame:CGRectMake(0, 0,
self.view.bounds.size.width, 48)];
view.delegate = self;
[view setPullUp:@"上拉加载更多信息"];
[view setRelease:@"放松"];
self.tableView.tableFooterView = view;
_refreshFooterView = view;
}

_isHeader = YES;
}

- (void)didReceiveMemoryWarning
{
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

#pragma tableview datasource
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return 1;
}
```

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
return _listArray.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
static NSString *CellIdentifier = @"RefreshCell";
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
if (nil == cell)
{
cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
}
cell.textLabel.text = _listArray[indexPath.row];
cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

return cell;
}

#pragma mark -
#pragma mark Data Source Loading / Reloading Methods

- (void)reloadHeaderTableViewDataSource{

// should be calling your tableviews data source model to reload
// put here just for demo
NSInteger first = [_listArray[0] integerValue] - 1;
[_listArray insertObject:[NSString stringWithFormat:@"%li",(long)first] atIndex:0];

_headerReloading = YES;
}

- (void)doneLoadingHeaderTableViewData{

// model should call this when its done loading
_headerReloading = NO;
[_refreshHeaderView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
[self.tableView reloadData];

}

- (void)reloadFooterTableViewDataSource{

// should be calling your tableviews data source model to reload
// put here just for demo
NSInteger count = [_listArray count];
NSInteger last = [_listArray[count-1] integerValue] + 1;
[_listArray addObject:[NSString stringWithFormat:@"%li",(long)last]];

_footerReloading = YES;
}

- (void)doneLoadingFooterTableViewData{

// model should call this when its done loading
```

```
_footerReloading = NO;
[_refreshFooterView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
[self.tableView reloadData];

}

#pragma mark -
#pragma mark UIScrollViewDelegate Methods

- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
if (scrollView.contentInset.top + scrollView.contentOffset.y < 0) {
_isHeader = YES;
} else {
_isHeader = NO;
}

if (_isHeader) {
[_refreshHeaderView egoRefreshScrollViewDidScroll:scrollView];
} else {
[_refreshFooterView egoRefreshScrollViewDidScroll:scrollView];
}
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
if (_isHeader) {
[_refreshHeaderView egoRefreshScrollViewDidEndDragging:scrollView];
} else {
[_refreshFooterView egoRefreshScrollViewDidEndDragging:scrollView];
}
}

#pragma mark -
#pragma mark EGORefreshTableHeaderDelegate Methods

- (void)egoRefreshTableHeaderDidTriggerRefresh:(AUPullLoadingView*)view
{
if (_isHeader) {
[self reloadHeaderTableViewDataSource];
[self performSelector:@selector(doneLoadingHeaderTableViewData) withObject:nil afterDelay:2.0];
} else {
[self reloadFooterTableViewDataSource];
[self performSelector:@selector(doneLoadingFooterTableViewData) withObject:nil afterDelay:2.0];
}

}

- (BOOL)egoRefreshTableHeaderDataSourceIsLoading:(AUPullLoadingView*)view
{
if (_isHeader) {
return _headerReloading;
} else {
return _footerReloading; // should return if data source model is reloading
}
```

```
}

- (NSDate*)egoRefreshTableHeaderDataSourceLastUpdated:(AUPullLoadingView*)view{

return [NSDate date]; // should return date data source was last changed

}

@end
```

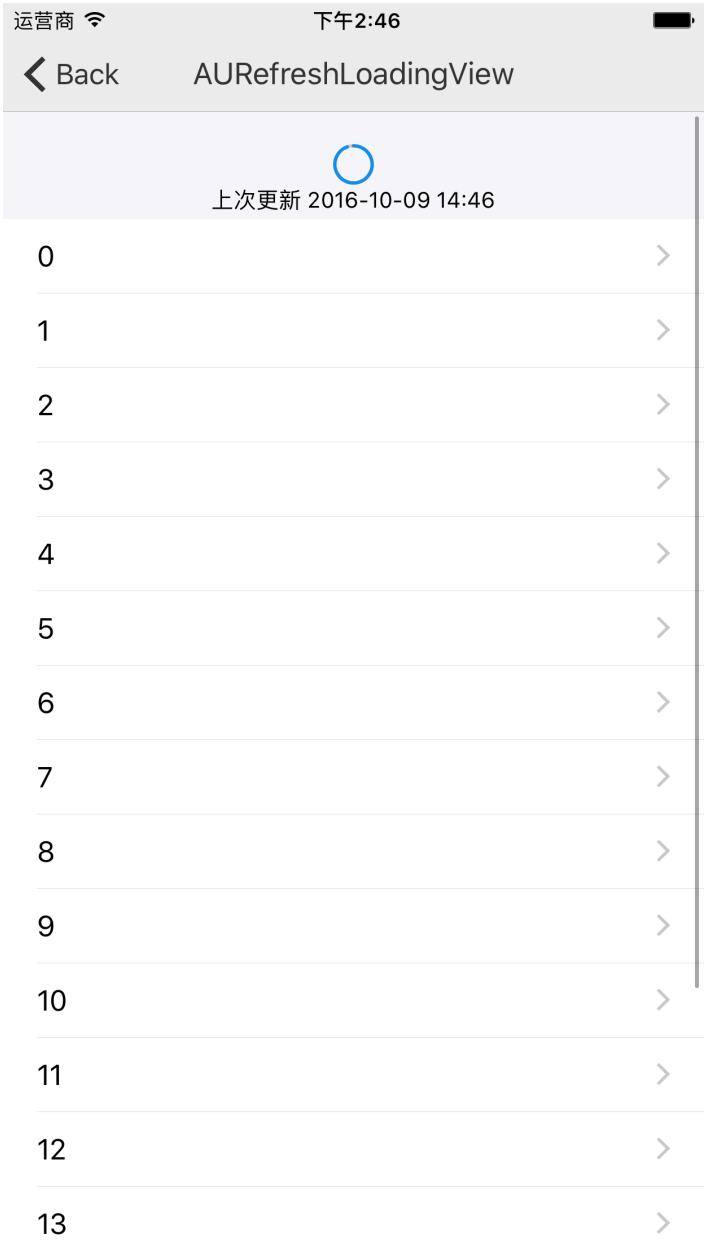
5.6.2 下拉刷新控件

- 需要对 iOS 中的 BeeViews：BEESmilePullRefreshView(BEESmileAnimationView、BEESmileStaticView) 首页单独刷新控件，由首页负责。

  以下非业务定制化的控件，都需要切换为下拉（AUPullLoadingView）或上拉（AUDragLoadingView）组件。

  CommonUI：ODRefreshControl、APCircleRefreshControl、EGORefreshTableHeaderView、APNextPagePullView

效果图



接口说明

AUPullLoadingView.h

```
//
// EGORefreshTableHeaderView.h
// Demo
//

#import <UIKit/UIKit.h>
#import <QuartzCore/QuartzCore.h>

typedef enum {
AUEGOPullingDown = 1000,
```

```
AUEGOPullingUp
} AUEGOPullDirection;

typedef enum{
AUEGOOPullRefreshPulling = 0,
AUEGOOPullRefreshNormal,
AUEGOOPullRefreshLoading,
} AUEGOPullRefreshState;

@class AULoadingIndicatorView;
@protocol AURefreshLoadingViewDelegate;
/* !
@class AURefreshLoadingView
@abstract UIView
@discussion 从第三方 EGORefreshTableHeaderView 迁移而来，功能下拉、上拉加载更多的 view
*/

@interface AUPullLoadingView : UIView {

__weak id _delegate;
AUEGOPullRefreshState _state;

UILabel *_lastUpdatedLabel;
UILabel *_statusLabel;
// APActivityIndicatorView *_activityView;
AUEGOPullDirection _pullDirection;

BOOL isAutoPullFlag;
}
@property(nonatomic, strong) AULoadingIndicatorView *activityView;

/**
* 上拉加载时，设置初始状态文案信息，默认为“上拉加载更多”，显示
*
* @param tip tips内容
*
*/
- (void)setPullUp:(NSString *)tip;

/**
* 下拉刷新时，设置初始状态文案信息，默认为“下拉刷新”。不显示
*
* @param tip tips内容
*
*/
- (void)setPullDown:(NSString *)tip;

/**
* 设置松手后 loading 过程中的文案信息，默认为“加载中”。
* 注意：默认下拉刷新时不显示，上拉加载时显示
*
* @param tip tips内容
*
*/
- (void)setLoading:(NSString *)tip;
```

```
/**
* 提示用户可以放手的文案信息，默认为“释放即可刷新”
*
* @param tip tips内容
*
*/
- (void)setRelease:(NSString *)tip;

/**
* 是否显示上次刷新信息的文案，默认不显示
*
* @param isOpen YES 表示显示
*
*/
- (void)ShowLastPullDate:(BOOL)isOpen;

/**
* 是否显示加载状态信息的文案。
*
* 默认：下拉刷新时不显示，上拉加载时，显示文案“加载中”
*
* @param isShow YES 表示显示
*
*/
- (void)ShowStatusLabel:(BOOL)isShow;

- (void)setDateFormat:(NSDateFormatter *)dateFromatter;

- (void)setAutoPull:(BOOL)isAutoPull;

@property(nonatomic,weak) id <AURefreshLoadingViewDelegate> delegate;

- (void)refreshLastUpdatedDate;
- (void)egoRefreshScrollViewDidScroll:(UIScrollView *)scrollView;
- (void)egoRefreshScrollViewDidEndDragging:(UIScrollView *)scrollView;
- (void)egoRefreshScrollViewDataSourceDidFinishedLoading:(UIScrollView *)scrollView;
- (void)egoRefreshScrollViewDataSourceDidFinishedLoadingWithoutUpdate:(UIScrollView *)scrollView;

- (void)autoUpdateScrollView:(UIScrollView *)scrollView;

#pragma Mark -- for LegacySystem not recommend
@property(nonatomic,assign) AUEGOPullRefreshState state;
@property(nonatomic,retain) NSString *statusText;
@property (nonatomic, retain) UILabel *lastUpdatedLabel;
@property (nonatomic, retain) UILabel *statusLabel;

- (void)setCurrentDate;

@end

@protocol AURefreshLoadingViewDelegate
- (void)egoRefreshTableHeaderDidTriggerRefresh:(AUPullLoadingView*)view;
- (BOOL)egoRefreshTableHeaderDataSourceIsLoading:(AUPullLoadingView*)view;
@optional
- (NSDate*)egoRefreshTableHeaderDataSourceLastUpdated:(AUPullLoadingView*)view;
@end
```

AUDragLoadingView.h

见 上拉刷新控件 。

代码示例

```
//
// APRefreshTableViewController.m
// UIDemo
//

#import"APRefreshTableViewController.h"
@interface APRefreshTableViewController ()
{
BOOL _headerReloading;
BOOL _footerReloading;
BOOL _isHeader;
}
@property(nonatomic,strong)AUPullLoadingView *refreshHeaderView;
@property(nonatomic,strong)AUDragLoadingView *refreshFooterView;
@property(nonatomic, strong) UITableView *tableView;
@property(nonatomic, strong) NSMutableArray* listArray;


@end

@implementation APRefreshTableViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
if (self) {
// Custom initialization
NSArray *array = @[@"0",
@"1",
@"2",
@"3",
@"4",
@"5",
@"6",
@"7",
@"8",
@"9",
@"10",
@"11",
@"12",
@"13",
@"14",
@"15",
@"16",
@"17",
@"18",
@"19"];
self.listArray = [NSMutableArray arrayWithArray:array];
```

```
}
return self;
}

- (void)viewDidLoad
{
[super viewDidLoad];
// Do any additional setup after loading the view.
self.edgesForExtendedLayout = UIRectEdgeNone;
// self.navigationItem.rightBarButtonItem = [APUtil getBarButtonWithTitle:RightBarButtonTitle target:self];

self.tableView = [[UITableView alloc]initWithFrame:self.view.bounds style:UITableViewStylePlain];
self.tableView.dataSource = self;
self.tableView.delegate = self;
self.tableView.backgroundColor = [UIColor colorWithRGB:0xf5f5f9];
self.tableView.separatorColor = [UIColor colorWithRGB:0xdddddd];
[self.view addSubview:self.tableView];

if (_refreshHeaderView == nil) {

AUPullLoadingView *view = [[AUPullLoadingView alloc] initWithFrame:CGRectMake(0.0f, 0.0f -
self.tableView.bounds.size.height, self.view.frame.size.width, self.tableView.bounds.size.height)];
view.delegate = self;
[view ShowLastPullDate:YES];
[view ShowStatusLabel:NO];

[self.tableView addSubview:view];
_refreshHeaderView = view;
}
[_refreshHeaderView refreshLastUpdatedDate];

if (_refreshFooterView == nil) {
AUDragLoadingView *view = [[AUDragLoadingView alloc] initWithFrame:CGRectMake(0, 0,
self.view.bounds.size.width, 48)];
view.delegate = self;
[view setPullUp:@"上拉加载更多信息"];
[view setRelease:@"放松"];
self.tableView.tableFooterView = view;
_refreshFooterView = view;
}

_isHeader = YES;
}

- (void)didReceiveMemoryWarning
{
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

#pragma tableview datasource
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return 1;
}
```

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
return _listArray.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
static NSString *CellIdentifier = @"RefreshCell";
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
if (nil == cell)
{
cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
}
cell.textLabel.text = _listArray[indexPath.row];
cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

return cell;
}

#pragma mark -
#pragma mark Data Source Loading / Reloading Methods

- (void)reloadHeaderTableViewDataSource{

// should be calling your tableviews data source model to reload
// put here just for demo
NSInteger first = [_listArray[0] integerValue] - 1;
[_listArray insertObject:[NSString stringWithFormat:@"%li",(long)first] atIndex:0];

_headerReloading = YES;
}

- (void)doneLoadingHeaderTableViewData{

// model should call this when its done loading
_headerReloading = NO;
[_refreshHeaderView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
[self.tableView reloadData];

}

- (void)reloadFooterTableViewDataSource{

// should be calling your tableviews data source model to reload
// put here just for demo
NSInteger count = [_listArray count];
NSInteger last = [_listArray[count-1] integerValue] + 1;
[_listArray addObject:[NSString stringWithFormat:@"%li",(long)last]];

_footerReloading = YES;
}

- (void)doneLoadingFooterTableViewData{

// model should call this when its done loading
```

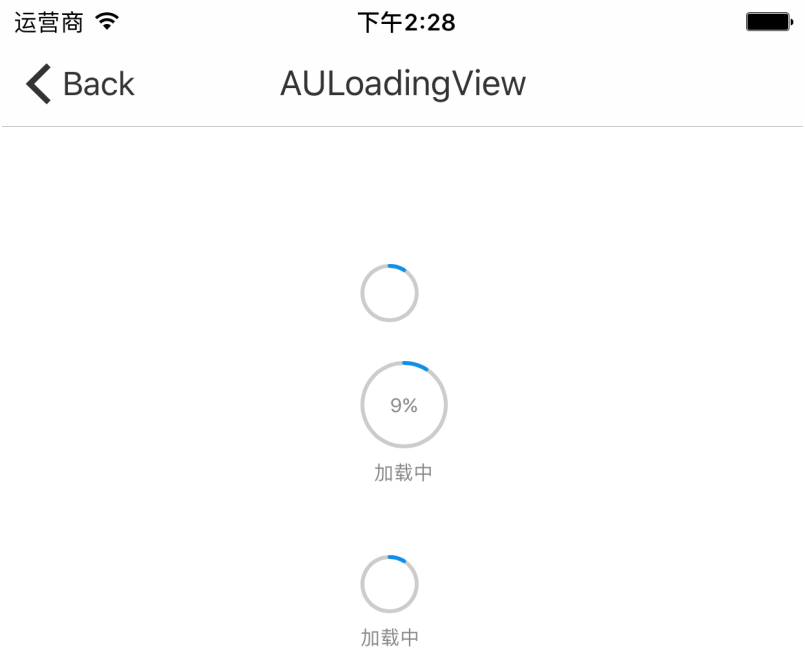
```
_footerReloading = NO;
[_refreshFooterView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
[self.tableView reloadData];

}

#pragma mark -
#pragma mark UIScrollViewDelegate Methods

- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
if (scrollView.contentInset.top + scrollView.contentOffset.y < 0) {
_isHeader = YES;
} else {
_isHeader = NO;
}

if (_isHeader) {
[_refreshHeaderView egoRefreshScrollViewDidScroll:scrollView];
} else {
[_refreshFooterView egoRefreshScrollViewDidScroll:scrollView];
}
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
if (_isHeader) {
[_refreshHeaderView egoRefreshScrollViewDidEndDragging:scrollView];
} else {
[_refreshFooterView egoRefreshScrollViewDidEndDragging:scrollView];
}
}

#pragma mark -
#pragma mark EGORefreshTableHeaderDelegate Methods

- (void)egoRefreshTableHeaderDidTriggerRefresh:(AUPullLoadingView*)view
{
if (_isHeader) {
[self reloadHeaderTableViewDataSource];
[self performSelector:@selector(doneLoadingHeaderTableViewData) withObject:nil afterDelay:2.0];
} else {
[self reloadFooterTableViewDataSource];
[self performSelector:@selector(doneLoadingFooterTableViewData) withObject:nil afterDelay:2.0];
}

}

- (BOOL)egoRefreshTableHeaderDataSourceIsLoading:(AUPullLoadingView*)view
{
if (_isHeader) {
return _headerReloading;
} else {
return _footerReloading; // should return if data source model is reloading
}
```

```
}

- (NSDate*)egoRefreshTableHeaderDataSourceLastUpdated:(AUPullLoadingView*)view{

return [NSDate date]; // should return date data source was last changed

}

@end
```

5.6.3 加载组件

AULoadingView 为新增的加载控件。

效果图



接口定义

AULoadingView.h

```
//
// AULoadingView.h
// AntUI
//

#import <UIKit/UIKit.h>

/**
中间加载的 loading 控件中间含数字
*/
```

```
@interface AULoadingView : UIView

@property (nonatomic,assign) BOOL isShowProgressPer; //是否显示进度百分比，默认为 NO
@property (nonatomic,assign) BOOL isShowLoadingText; //是否显示加载文案，默认为 NO


/**
设置进度百分比

@param progress 百分比的值
*/
- (void) setProgressPer:(CGFloat) progress;


@end
```

代码示例

```
//
// AULoadingViewController.m
// AntUI
//

#import"AULoadingViewController.h"
#import"AULoadingView.h"

@interface AULoadingViewController ()
@property (nonatomic,strong) AULoadingView * loadingView;
@property (nonatomic,strong) AULoadingView * loadingView2;
@property (nonatomic,strong) AULoadingView * loadingView3;

@property (nonatomic,assign) CGFloat progress;

@end

@implementation AULoadingViewController

- (void)viewDidLoad {
[super viewDidLoad];
self.view.backgroundColor = [UIColor whiteColor];
// Do any additional setup after loading the view.
self.loadingView = [[AULoadingView alloc] init];
self.loadingView.center = CGPointMake(200, 200);
self.loadingView.isShowProgressPer = YES;
self.loadingView.isShowLoadingText = YES;
[self.view addSubview:self.loadingView];

self.loadingView2 = [[AULoadingView alloc] init];
self.loadingView2.center = CGPointMake(200, 150);
// self.loadingView2.isShowProgressPer = YES;
// self.loadingView2.isShowLoadingText = YES;
[self.view addSubview:self.loadingView2];
```

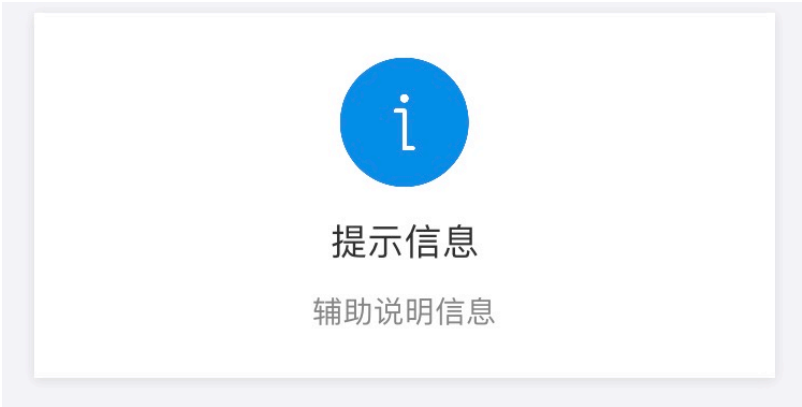
```
self.loadingView3 = [[AULoadingView alloc] init];
self.loadingView3.center = CGPointMake(200, 300);
// self.loadingView3.isShowProgressPer = YES;
self.loadingView3.isShowLoadingText = YES;
[self.view addSubview:self.loadingView3];


[NSTimer scheduledTimerWithTimeInterval:0.1
target:self
selector:@selector(loadingTimer:)
userInfo:nil
repeats:YES];

}


- (void) loadingTimer:(id)timer
{
self.progress += 0.01;
if ((int)(self.progress *100) > 100) {
self.progress = 0.0;
[timer invalidate];
return;
}
[self.loadingView setProgressPer:self.progress];
[self.loadingView2 setProgressPer:self.progress];
[self.loadingView3 setProgressPer:self.progress];

}


- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
// Get the new view controller using [segue destinationViewController].
// Pass the selected object to the new view controller.
}
*/

@end
```

5.7 结果页组件

5.7.1 结果页组件

AUResultView 用来展示一些带图片的状态视图。

效果图



接口说明

```
/**
显示状态的结果视图
*/
@interface AUResultView : UIView

/**
顶部图像
*/
@property (nonatomic, strong) UIImage *icon;

/**
文本区域顶部的黑色中尺寸标题
*/
@property (nonatomic, strong) NSString *mainTitleText;

/**
中间的黑色大尺寸标题
*/
@property (nonatomic, strong) NSString *midTitleText;

/**
底部的灰色消息
*/
```

```
@property (nonatomic, strong) NSString *bottomMessage;

/**
底部消息是否加贯穿线
*/
@property (nonatomic, assign) BOOL messageThrough;

/**
视图期望的高度，初始化完成即可获取
*/
@property (nonatomic, assign, readonly) CGFloat expectHeight;

/**
ResultView 实例化方法

@param icon 图像
@param mainTitleText 第一个标题
@param midTitleText 中间大标题
@param bottomMessage 底部灰色消息
@param messageThrough 是否加横线贯穿
@return AUResultView 实例
*/
- (instancetype)initWithIcon:(UIImage *)icon mainTitleText:(NSString *)mainTitleText midTitleText:(NSString
*)midTitleText bottomMessage:(NSString *)bottomMessage messageThrough:(BOOL)messageThrough;
```

代码示例

```
UIImage *image = AUBundleImage(@"alipay-60");
AUResultView *resultView = [[AUResultView alloc] initWithIcon:image
mainTitleText:@"支付成功"
midTitleText:@"998.00"
bottomMessage:@"1098.00元"
messageThrough:YES];
resultView.frame = CGRectMake(marginX, originY, AUCommonUIGetScreenWidth()-2*marginX,
resultView.expectHeight);
[self.view addSubview:resultView];
```

5.7.2 异常页组件

- AUNetErrorView 为空页面异常视图显示控件。
- AUNetErrorView 包括两种提示风格：
    - 简单版风格（默认），包含 5 种样式
    - 插图版风格，包含 5 种样式

两种风格的主要区别在于使用的提示图片不同，见效果图。

效果图

简单版本（半屏）风格

AUNetErrorTypeAlert

AUNetErrorTypeNetworkError

AUNetErrorTypeEmpty

AUNetErrorTypeLimit

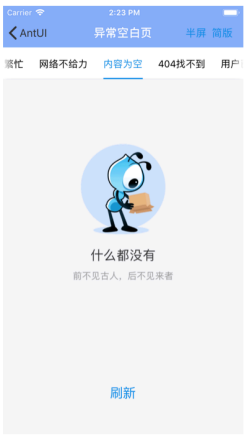
AUNetErrorTypeUserLogout

插图版本（全屏）风格

AUNetErrorTypeLimit

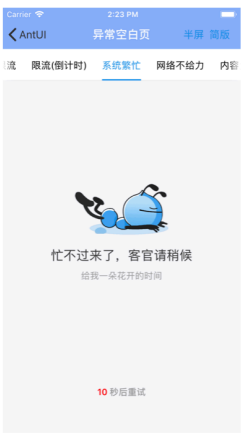
AUNetErrorTypeNetworkError

AUNetErrorTypeEmpty



AUNetErrorTypeUserLogout

AUNetErrorTypeAlert



接口说明

```
typedef NS_ENUM(NSInteger, AUNetErrorType) {

AUNetErrorTypeLimit, // 限流
AUNetErrorTypeAlert, // 系统繁忙（系统错误）、警示
AUNetErrorTypeNetworkError, // 网络不给力
```

```
AUNetErrorTypeEmpty, // 内容为空
AUNetErrorTypeNotFound, // 404 找不到（与 AUNetErrorTypeAlert 图片相同）
AUNetErrorTypeUserLogout, // 用户已注销

AUNetErrorTypeFailure __attribute__((deprecated)) = AUNetErrorTypeNetworkError,
AUNetErrorTypeError __attribute__((deprecated)) = AUNetErrorTypeNetworkError, //网络错误，完全无法连接
AUNetErrorTypeSystemBusy __attribute__((deprecated)) = AUNetErrorTypeAlert, //警示
APExceptionEnumNetworkError __attribute__((deprecated)) = AUNetErrorTypeNetworkError, //网络错误，完全无法连接
APExceptionEnumEmpty __attribute__((deprecated)) = AUNetErrorTypeEmpty, //内容为空
APExceptionEnumAlert __attribute__((deprecated)) = AUNetErrorTypeAlert, //警示
APExceptionEnumLimit __attribute__((deprecated)) = AUNetErrorTypeLimit, //限流，
APExceptionEnumNetworkFailure __attribute__((deprecated)) = AUNetErrorTypeNetworkError, //网络不给力
};


typedef NS_ENUM(NSInteger, AUNetErrorStyle) {
AUNetErrorStyleMinimalist, //简单版
AUNetErrorStyleIlustration, //插图版

APExceptionStyleIlustration __attribute__((deprecated)) = AUNetErrorStyleIlustration, //插图版
APExceptionStyleMinimalist __attribute__((deprecated)) = AUNetErrorStyleMinimalist //简单版
};



/**
空页面异常视图显示控件

包括两种提示风格：
1、简单版风格（默认），包含3种类型样式
2、插图版风格，包含7种类型样式

两种风格和类型主要是图片不一样。
*/
@interface AUNetErrorView : UIView

@property(nonatomic, strong, readonly) UIButton *actionButton; // 默认文案是刷新
@property(nonatomic, strong, readonly) UIImageView *iconImageView; // icon 视图
@property(nonatomic, strong, readonly) UILabel *infoLabel; // 主提示文案 Label
@property(nonatomic, strong, readonly) UILabel *detailLabel; // 详细提示文案 Label

@property(nonatomic, strong) NSString *infoTitle; // 主文案说明
@property(nonatomic, strong) NSString *detailTitle; // 辅助文案说明

/**
* 初始化异常view并设定异常风格和类型
* （target和action为空时，刷新按钮不显示）
*
* @param frame view 的坐标，必选
* @param style 异常的风格，插画版 or 极简版，必选
* @param type 异常类型，必选
* @param target 刷新事件处理对象
* @param action 刷新事件处理方法
*
* @return APExceptionView
*/
```

```
- (id)initWithFrame:(CGRect)frame
style:(AUNetErrorStyle)style
type:(AUNetErrorType)type
target:(id)target
action:(SEL)action;

/**
* 初始化异常视图并显示在指定的视图上
* （target 和 action 为空时，刷新按钮不显示）
*
* @param parent view 的 superView，必选
* @param style 异常的风格，插画版 or 极简版，必选
* @param type 异常类型，必选
* @param target 刷新事件处理对象
* @param action 刷新事件处理方法
*
* @return APExceptionView
*/
+ (id)showInView:(UIView *)parent
style:(AUNetErrorStyle)style
type:(AUNetErrorType)type
target:(id)target
action:(SEL)action;

/**
* 取消异常视图的显示
*/
- (void)dismiss;

/**
* 倒计时 仅限限流使用
* 如果 completeBlock == nil 且 业务没有设置 actionButton 的点击响应事件 则倒计时功能不生效；
* 如果 completeBlock != nil，倒计时结束直接执行 completeBlock，同时隐藏 actionButton
* 如果使用 getActionButton 来添加button的响应事件，要确保在该方法之前添加 actionButton 的响应事件
*/
- (void)setCountdownTimeInterval:(NSInteger)startTime // 倒计时起始时间
completeBlock:(void (^)(void))completeBlock; // 倒计时结束后

@end
```

代码示例

```
netErrorView = [[AUNetErrorView alloc] initWithFrame:CGRectMake(0, CGRectGetMaxY(label.frame) + 5,
self.view.width, 300) style:AUNetErrorStyleIllustration type:AUNetErrorTypeError target:self
action:@selector(pressedNetErrorView)];
netErrorView.detailTitle = @"类型是AUNetErrorTypeError";
[self.view addSubview:netErrorView];

// 设置倒计时
[netErrorView setCountdownTimeInterval:10 completeBlock:^{
NSLog(@"倒计时结束");
}];
```

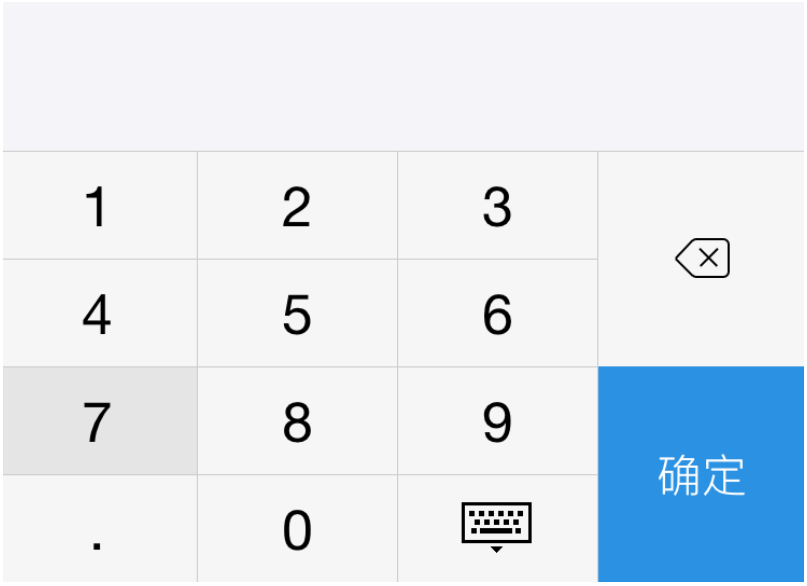
5.8 键盘组件

5.8.1 键盘组件

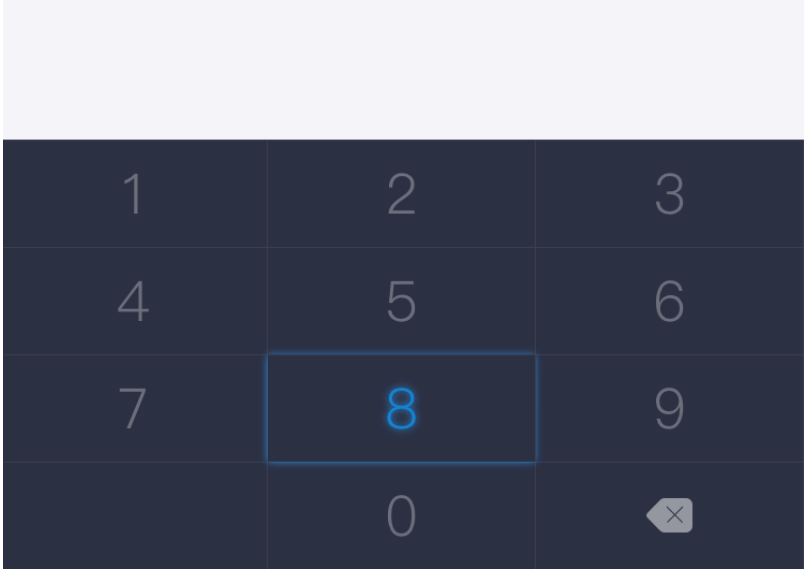
AUNumKeyboards 为自定义数字键盘。

效果图

Common Mode



Chat Mode



接口说明

```
typedef NS_ENUM(NSInteger, AUNumKeyboardMode) {
AUNumKeyboardModeCommon, //通用键盘
AUNumKeyboardModeChat, //聊天键盘
AUNumKeyboardModeInvalid //无效键盘，目前不可用
```

```
};


/**
自定义数字键盘
*/
@interface AUNumKeyboards : UIView

/**
* 创建键盘组件，默认为通用键盘
*
* @return 初始化的键盘组件
*/
+ (AUNumKeyboards *)sharedKeyboard;

/**
* 创建键盘组件
*
* @param mode 键盘模式
*
* @return 初始化的键盘组件
*/
+ (AUNumKeyboards *)sharedKeyboardWithMode:(AUNumKeyboardMode)mode;

/**
* 手动设置 textinput，外部需要设置 keyboard 的 Y 轴
*/
@property (nonatomic, weak) id<UITextInput> textInput;

/**
* 身份证 x
*/
@property (nonatomic, assign) BOOL idNumber;

/**
* 设置键盘模式
*/
@property (nonatomic, assign, readonly) AUNumKeyboardMode mode;

/**
* 小数点，是否隐藏
*/
@property (nonatomic, assign) BOOL dotHidden;

/**
* 是否收起键盘
*/
@property (nonatomic, assign) BOOL dismissHidden;

/**
* 提交按钮是否可点
*/
@property (nonatomic, assign) BOOL submitEnable;

/**
* 提交按钮文案
```

```
* 注意：根据视觉要求，此文案最多显示三个汉字，国际化时请注意英文文案长度
*/
@property (nonatomic, strong) NSString *submitText;
```

代码案例

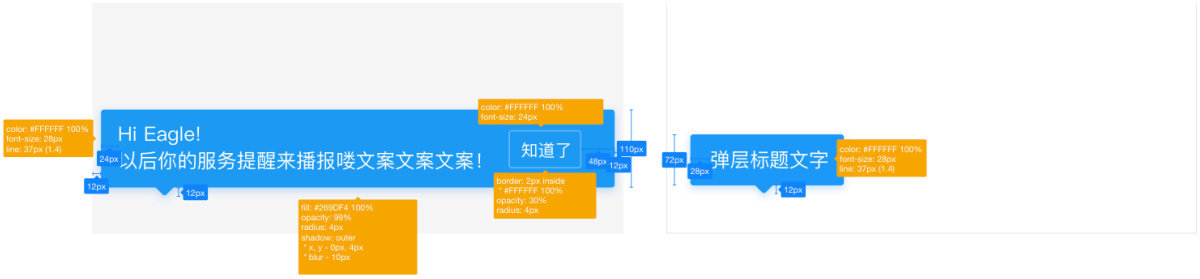
```
UITextField *numTextField = ...
numTextField.inputView = [AUNumKeyboards sharedKeyboardWithMode:AUNumKeyboardModeCommon] ; //Chat Mode 参数: AUNumKeyboardModeChat
[self.view addSubview:numTextField];
```

5.9 引导组件

5.9.1 引导提示组件

AUPopTipView 为引导提示组件。

效果图



接口说明

```
typedef NS_ENUM(NSInteger, AUPopViewIndicatorDirection) {
AUPopViewIndicatorDirectionUp,
AUPopViewIndicatorDirectionDown,
};

@interface AUPopTipView : AUPopDrawBoardView

AU_UNAVAILABLE_INIT

@property (nonatomic, assign) AUPopViewIndicatorDirection indicatorDirection;

- (void)dismiss:(BOOL)animated;

+ (instancetype)showFromView:(UIView *)fromView
fromPoint:(CGPoint)fromPoint
toView:(UIView *)toView
animated:(BOOL)animated
withText:(NSString *)text
buttonTitle:(NSString *)buttonTitle;
```

```
@end
```

代码示例

```
// 展示
AUPopTipView *popTipView = [AUPopTipView showFromView:button
fromPoint:CGPointZero
toView:self.view
animated:YES
withText:@"你好福建省"
buttonTitle:@"关闭"]; // 当 buttontitle 不传递时，右边按钮不显示

// 隐藏
[popTipView dismiss:YES];
```

5.9.2 引导浮层栏组件

AUPopBar 为引导浮层栏组件。

效果图

接口说明

```
@interface AUPopBar : AUView

AU_UNAVAILABLE_INIT

+ (instancetype)showInViewBottom:(UIView *)view
animated:(BOOL)animated
withText:(NSString *)text
icon:(UIImage *)icon
buttonTitle:(NSString *)buttonTitle
actionBlock:(BOOL(^)())actionBlock;

- (void)dismiss:(BOOL)animated;
```

```
@end
```

代码示例

```
// 展示
AUPopBar *popBar = [AUPopBar showInViewBottom:weakSelf.view animated:YES withText:@"把“城市服务”添加到首页"icon:[UIImage imageNamed:@"ap_scan"] buttonTitle:@"立即添加"actionBlock:^{
NSLog(@"点击了");
return YES;
}];

// 隐藏
[popBar dismiss:YES];
```

5.10 导航组件

5.10.1 纵向选择器

AUVerticalTabView 为纵向选择器组件。

效果图

依赖

AUVerticalTabView 的依赖如下：

```
AntUI
```

接口说明

```
#import <UIKit/UIKit.h>

@protocol AUVerticalTabViewDataProtocol <NSObject>

@required
- (NSString *) tabName;
```

```
@end

@class AUVerticalTabView;

typedef void (^AUVerticalTabSelectedCallback)(AUVerticalTabView *verticalTabView);

@interface AUVerticalTabView : UIView

/**
推荐初始化方法，布局参数为 AntDNA 规范:
AUVerticalTabView : width=110pt
TabCell : width=110pt,height=55pt

@param verticalTabViewDatas 设置 tab 数据
@param selectedCallback 设置点击回调
@param height AUVerticalTabView 高度
@param business 业务标示，如：GoldWord、BeeCityPicker
@return AUVerticalTabView
*/
+ (AUVerticalTabView *)verticalTabViewWithDatas:(NSArray <id<AUVerticalTabViewDataProtocol>>*)
verticalTabViewDatas
selectedCallback:(AUVerticalTabSelectedCallback)selectedCallback
height:(CGFloat)height
business:(NSString *)business;

@property(nonatomic, strong) NSArray <id<AUVerticalTabViewDataProtocol>>* verticalTabViewDatas;
@property(nonatomic, assign) NSUInteger selectedIndex;//default 0
@property(nonatomic, copy) AUVerticalTabSelectedCallback selectedCallback;

@end
```

代码示例

```
// 外部数据对象实现 AUVerticalTabViewDataProtocol，返回需要的 tabName
@interface DemoVerticalTabData : NSObject <AUVerticalTabViewDataProtocol>

- (NSString *)tabName;

@end


NSArray *datas = @[[DemoVerticalTabData new],
[DemoVerticalTabData new],
[DemoVerticalTabData new],
[DemoVerticalTabData new],
[DemoVerticalTabData new],
[DemoVerticalTabData new],
[DemoVerticalTabData new]];

AUVerticalTabView *tabView = [AUVerticalTabView verticalTabViewWithDatas:datas
selectedCallback:^(AUVerticalTabView *verticalTabView ){
NSUInteger selectedIndex = verticalTabView.selectedIndex;
id<AUVerticalTabViewDataProtocol> selectedData = [verticalTabView.verticalTabViewDatas
```

```
objectAtIndex:selectedIndex];
}
height:self.view.height
business:@"AntUI"];

[self.view addSubview:tabView];
```

5.10.2 双标题

AUDoubleTitleView 为导航栏两行标题的视图控件（主标题和副标题）。

效果图

接口说明

```
/**
包含两行的导航栏 titleView
*/
@interface AUDoubleTitleView : UIView

/**
* 创建上下两个标题的 titleView
*
* @param title 主标题
* @param detaileTitle 副标题
*
* @return 初始化后的 APTitleView 控件
*/
- (UIView *)initWithTitle:(NSString *)title detailTitle:(NSString *)detaileTitle;

/**
* 修改主标题的文案。
*
* @param title 主标题文案
*
*/
- (void)updateTitle:(NSString *)title;

/**
* 修改主标题的文案。
*
* @param detailTitle 主标题文案
*
*/
```

```
- (void)updateDetailTitle:(NSString *)detailTitle;


/**
修改主标题的 font

@param titleFont 主标题 font
*/
- (void)updateTitleFont:(UIFont *)titleFont;

/**
* 修改主标题的 font
*
* @param detailTitleFont 主标题 font
*
*/
- (void)updateDetailTitleFont:(UIFont *)detailTitleFont;

@end
```

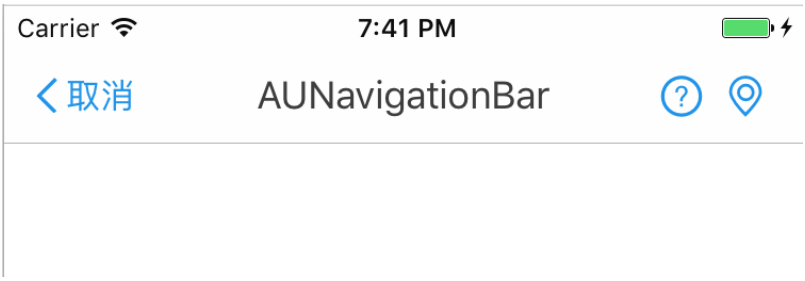
代码示例

```
self.navigationItem.titleView = [[AUDoubleTitleView alloc] initWithTitle:@"主标题"detailTitle:@"副标题"];
```

5.10.3 导航栏

- AUNavigationBar 为 mPaaS 导航栏控件，继承自 AUNavigationBar，并提供了 mPaaS 导航栏的默认样式等。
- 为方便后续扩展，所有 mPaaS 都必须使用 AUNavigationBar，而不是系统的 UINavigationBar。

效果图



接口说明

```
/**
mPaaS 导航栏控件（包括 mPaaS 样式等）
初始化：
UINavigationController *navBar = [[UINavigationController alloc]
initWithNavigationBarClass:NSClassFromString(@"AUNavigationBar") toolbarClass:nil];
*/
@interface AUNavigationBar : UINavigationBar
```

```
@end


/**
UINavigationBar 扩展，定义了 UINavigationBar 的默认样式
*/
@interface UINavigationBar (AUNavigationBarExtensions)

/**
* 返回框架导航栏默认 title 颜色，默认为 #000000
*
* @return
*/
+ (UIColor*)getNavigationBarTitleDefaultColor;

/**
* 返回框架导航栏上 item 颜色，默认为 #108EE9
*
* @return
*/
+ (UIColor*)getNavigationBarButtonItemDefaultColor;

/**
* 返回框架导航栏颜色，默认为 #fffff
*
* @return
*/
+ (UIColor*)getNavigationBarDefaultColor;

/**
* 获取导航栏底部横线的颜色，默认为 #e1e1e1
*
* @return
*/
+ (UIColor*)getNavigationBarBotLineColor;

/**
* 注意：
* 1、基类 DTViewController 在 ViewWillAppear 里设置了导航栏的默认样式；
* 2、业务可以通过系统接口或者下面提供的接口来修改导航栏的样式，一般在 ViewWillAppear 设置；
* 3、如果 VC 是 DTViewController 的子类必须在 ViewWillAppear 里设置，否则会被覆盖；
* 4、保证修改后在 ViewWillDisappear 时通过 setNavigationBarDefaultStyle 恢复默认样式；
* 5、如果 VC 是在 UITabBarController 容器的首页，不要做上面 4 的处理，否则切换 tab 时有覆盖问题。
*/

/**
*
* 设置默认的导航栏背景，默认设置背景色 #ffffff，底部横线 #e1e1e1
*
*/
- (void)setNavigationBarDefaultStyle;

/**
*
* 设置默认的导航栏标题样式
```

```
*
*/
- (void)setNavigationBarDefaultTitleTextAttributes;

/**
*
* 设置导航栏标题颜色，请在 ViewWillAppear 里设置，否则会被框架默认颜色覆盖
*
*/
- (void)setNavigationBarTitleTextAttributesWithTextColor:(UIColor *)textColor;

/**
*
* 设置导航栏透明样式
* 注意：此方法设置导航栏全透明后，返回的动画过程中会产生闪白问题，目前无解，业务请勿使用。如需使用，请评估影响是否可接受
*/
- (void)setNavigationBarTranslucentStyle;

/**
* 指定导航栏颜色，当 translucent 为 Yes 时，有毛玻璃效果
* 注意：调用此接口后，如有需要，请在此方法之后调用设置底部横线的接口，否则底部横线颜色会被默认颜色 #e1e1e1 覆盖
*
* @param color 显示颜色
* @param translucent 是否透明
*
*/
- (void)setNavigationBarStyleWithColor:(UIColor *)color translucent:(BOOL)translucent;

/**
* 导航栏下面可能有分割线，导致界面不符合一些UI的要求，使用这个方法设置
* 注意：若自定义了导航栏背景，（包括调用 setNavigationBarStyleWithColor：或重写 opaqueNavigationBarColor）
* 请在修改背景色方法之后调用此接口，否则底部横线颜色会被默认颜色 #e1e1e1 覆盖
*/
- (void)setNavigationBarBottomLineColor:(UIColor*)color;

/**
* 业务使用系统方法 setBarTintColor,setBackGroundImage,setBackgroundColor 设置导航栏颜色时，先调用此方法消除默认效果
* 否则默认颜色会与系统设置色叠加产生色差
*/
- (void)resetNavigationBarColor;

/**
*
* 屏蔽右滑返回取消时，导航栏闪烁的问题，业务方请勿调用
*/
- (void)setNavigationBarMaskLayerWithtColor:(UIColor *)color;

/**
* 返回导航栏当前的背景色
*
* @return 导航栏当前的背景色
*/
- (UIColor*)getNavigationBarCurrentColor;
```

```
@end
```

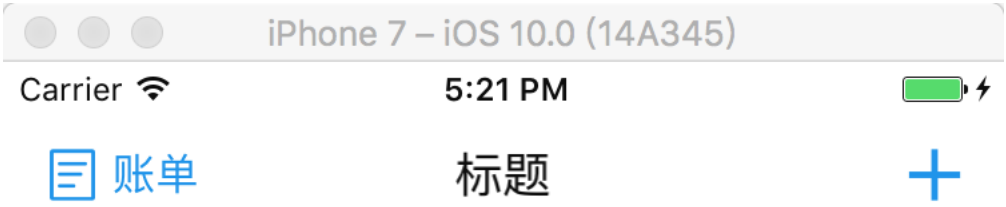
代码示例

```
// 初始化 UINavigationController
UINavigationController *navBar = [[UINavigationController alloc]
initWithNavigationBarClass:NSClassFromString(@"AUNavigationBar") toolbarClass:nil];

// 然后再 VC 中配置导航栏
AUBarButtonItem *cancelItem = [AUBarButtonItem backBarButtonItemWithTitle:@"返回"target:self
action:@selector(cancel)];
cancelItem.backButtonTitle = @"取消";
self.navigationItem.leftBarButtonItem = cancelItem;

UIImage *image1 = [AUIconView iconWithName:kICONFONT_MAP width:22 color:AU_COLOR_LINK];
UIImage *image2 = [AUIconView iconWithName:kICONFONT_HELP width:22 color:AU_COLOR_LINK];
AUBarButtonItem *rightItem1 = [[AUBarButtonItem alloc] initWithImage:image1 style:UIBarButtonItemStylePlain
target:self action:@selector(rightBarItemPressed)];
AUBarButtonItem *rightItem2 = [[AUBarButtonItem alloc] initWithImage:image2 style:UIBarButtonItemStylePlain
target:self action:@selector(rightBarItemPressed)];
self.navigationItem.rightBarButtonItems = @[rightItem1, rightItem2];
```

5.10.4 定制导航栏

- AUCustomNavigationBar 为 mPaaS 中专门为有需要导航栏透明定制的导航栏 。
- 原生的导航栏当从透明切换到不透明时会有视觉体验问题，故写此类。

效果图



接口说明

```
/**
自定义透明导航栏，主要用于导航栏需要透明的场景
由于用原生的导航栏切换会有视觉体验问题，故写此类
*/
@interface AUCustomNavigationBar : UIView

@property(nonatomic, strong) UIView *backgroundView; // 毛玻璃背景 view
```

```
@property(nonatomic, strong) NSString *backButtonTitle; // 返回按钮 title（默认无）
@property(nonatomic, strong) UIColor *backButtonTitleColor; // 返回按钮 title 颜色
@property(nonatomic, strong) UIImage *backButtonImage; // 返回按钮图片

@property(nonatomic, strong) NSString *title; // 标题
@property(nonatomic, strong) UIColor *titleColor; // 标题颜色
@property(nonatomic, strong) UIView *titleView; // 自定义 titleview

@property(nonatomic, strong) NSString *rightItemTitle; // 右侧 item title
@property(nonatomic, strong) UIColor *rightItemTitleColor; // 右侧 item title 颜色
@property(nonatomic, strong) UIImage *rightItemImage; // 右侧 item 图片

/**
* 右侧 item 的 VoiceOver 提示文案,
* 左侧 item 默认为“返回”
* 右侧 item 默认是 rightItemTitle，如果没有设置 rightItemTitle，需要手动设置此属性来支持 VoiceOver
*/
@property(nonatomic,strong) NSString *rightItemVoiceOverText;

@property(nonatomic,strong) NSString *leftItemVoiceOverText;


/**
* 创建指定透明的导航栏 View。
*
* （1）此导航栏默认在左侧显示返回箭头图片，不显示返回文本。若当前页面需设置与框架逻辑一致的返回文案，请在 VC 中重写 - (UIView *)customNavigationBar 方法
* （2）如需设置标题、右侧 item、毛玻璃背景，请调用相关接口
*
* @param currentVC 当前 VC
*
* @return 透明的导航栏 View
*/
+ (AUCustomNavigationBar *)navigationBarForCurrentVC:(UIViewController *)currentVC;

/**
* 设置毛玻璃背景 View，默认透明度为 0
*/
- (void)setNavigationBarBlurEffective;

/**
* 创建导航栏右侧 item
*
* @param rightItemTitle 显示的文本
* @param target target
* @param action action
*
*/
- (void)setNavigationBarRightItemWithTitle:(NSString *)rightItemTitle target:(id)target action:(SEL)action;

/**
* 创建导航栏右侧 item
*
* @param rightItemImage 显示的图片
* @param target target
* @param action action
```

```
*
*/
- (void)setNavigationBarRightItemWithImage:(UIImage *)rightItemImage target:(id)target action:(SEL)action;


/**
* 创建导航栏左侧 item
*
* @param leftItemTitle 显示的文本
* @param target target
* @param action action
*
*/
- (void)setNavigationBarLeftItemWithTitle:(NSString *)leftItemTitle target:(id)target action:(SEL)action;

/**
* 创建导航栏左侧 item
*
* @param leftItemTitle 显示的图片
* @param target target
* @param action action
*
*/
- (void)setNavigationBarLeftItemWithImage:(UIImage *)leftItemTitle target:(id)target action:(SEL)action;


@end
```

代码示例

```
AUCustomNavigationBar *navBar = [AUCustomNavigationBar navigationBarForCurrentVC:self];
[navBar setNavigationBarBlurEffective]; // 毛玻璃效果
[self.view addSubview:navBar];
navBar.title = @"标题";
navBar.backButtonImage = [AUIconView iconWithName:kICONFONT_BILL width:22 color:AU_COLOR_LINK];
navBar.backButtonTitle = @"账单";
navBar.rightItemImage = [AUIconView iconWithName:kICONFONT_ADD width:22 color:AU_COLOR_LINK];

// mPaaS 内使用，需覆盖父类的如下两个方法
- (BOOL)autohideNavigationBar
{
return YES;
}
- (UIView *)customNavigationBar
{
return self.navBar;
}
```

5.11 二维码组件

5.11.1 二维码组件

- AUQRCodeView 为支持多选项按钮的 Alert 视图。
- window 层级：self.windowLevel = UIWindowLevelAlert - 1

效果图

接口说明

```
// 数据模型对象
@interface QRDataModel : NSObject

@property (nonatomic, strong) id topLeftIcon; // 可以传 image 或者 url 或者 cloudID
@property (nonatomic, strong) NSString *topTitle; // 可以传 image 或者 url 或者 cloudID
@property (nonatomic, strong) id qrCodeIcon; // 二维码图
@property (nonatomic, strong) NSString *bottomTitle;
@property (nonatomic, strong) NSString *bottomMessage;
@property (nonatomic, strong) id actionButtonIcon; // 可以传 image 或者 url 或者 cloudID
@property (nonatomic, strong) NSString *actionButtonTitle; // 底部行动按钮主文案
@property (nonatomic, strong) NSString *actionButtonMessage; // 底部行动按钮辅助文案
```

```
@end

// 二维码底部行动按钮
@interface QRActionButton : UIControl

@end


// 二维码组件
@interface AUQRCodeView : UIView

@property (nonatomic, strong) UIView *maskView;
@property (nonatomic, strong) UIView *containerView; // 二维码容器
@property (nonatomic, strong) UIImageView *topLeftImageView; // 左上角图片
@property (nonatomic, strong) UILabel *topTitleLabel; // 顶部 title 描述文案
@property (nonatomic, strong) UIImageView *qrCodeView; // 二维码图
@property (nonatomic, strong) UILabel *bottomTitleLabel; // 底部主说明文案
@property (nonatomic, strong) UILabel *bottomMessageLabel; // 底部辅助说明文案
@property (nonatomic, strong) QRActionButton *actionButton; // 底部行为按钮

// frame 即控件 frame; block 初始化数据模型
- (instancetype)initWithFrame:(CGRect)frame model:(void(^)(QRDataModel *model))block;

// 转菊花
- (void)startLoading;

// 停止菊花
- (void)stopLoading;

@end
```

代码示例

- 标准样式

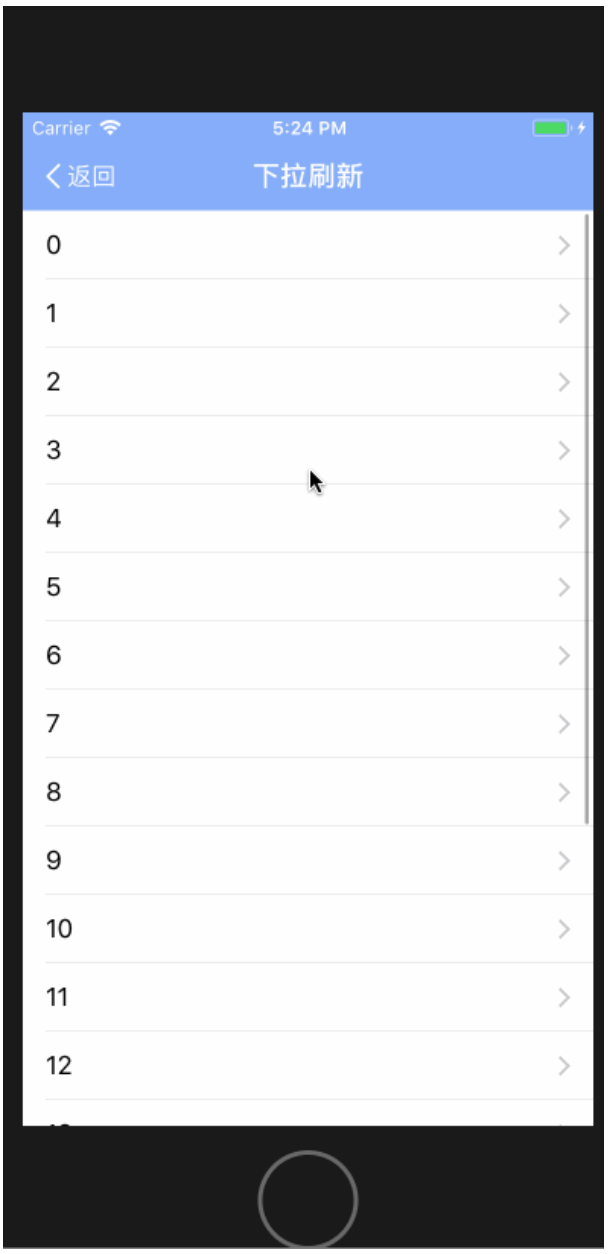
```
AUQRCodeView *qrCodeView = [[AUQRCodeView alloc] initWithFrame:frame model:^(QRDataModel *model) {
model.topLeftIcon = [UIImage imageWithColor:[UIColor colorWithRGB:0xbbbbbb] size:CGSizeMake(54, 54)];
model.topTitle = @"生活号名称";
model.bottomTitle = @"用支付宝二维码，关注生活号";
model.bottomMessage = @"该二维码将在 2017 年 11 月 05 日失效";
model.actionButtonTitle = @"保存到本地";
}];
[self.view addSubview:qrCodeView];
```

5.12 下拉刷新组件

5.12.1 下拉刷新组件

AURefreshView 为新版下拉刷新小蚂蚁样式，目前有两种色值，见效果图。

效果图



接口说明

```
typedef NS_ENUM(NSUInteger, AURefreshViewState) {
AURefreshViewStateNomal = 0, // 列表恢复为初始指定位置
AURefreshViewStateBeginPulling = 1, // 用户开始下拉
AURefreshViewStateLoading = 2, // 触发用户RPC加载，列表 contentInset 设置在默认地方
AURefreshViewStateFinishedLoading = 3, // RPC加载结束，列表 contentInset 即将恢复为原始默认的
AURefreshViewStateBeginResetting = 4, // 列表 contentInset 正在开始恢复原始默认 inset
};

typedef NS_ENUM(NSUInteger, AURefreshViewType) {
AURefreshViewDefault, // 页面内的刷新样式
AURefreshViewTypeFeature1 // 用在带有一定背景的 titlebar 如首页或者财富 tab
};

@protocol AURefreshViewDelegate;
```

```
/**
mPaaS 下拉刷新动画 View
*/

@interface AURefreshView : UIView

@property (nonatomic, readonly) AURefreshViewState state;

@property (nonatomic, weak) id <AURefreshViewDelegate> delegate;

/**
下拉刷新动画 Lottie 控件
*/
@property (nonatomic, strong) UIView /*LOTAnimationView */ *lottieAnimationView;

/* 指定下拉刷新所在的父 view，下拉刷新初始默认高度是 scrollView 的高度；默认将 refreshView 添加到父 scrollView 上
* 默认初始frame为(0, 0 - scrollView.height, scrollView.width, scrollView.height)) */
- (instancetype)initWithSuperView:(UIScrollView *)scrollView
type:(AURefreshViewType)type
bizType:(NSString *)bizType;

// 下拉刷新文案
- (void)setupLabelText:(NSString *)text;

// UIScrollView的delegate 里面回调以下方法
- (void)auRefreshScrollViewWillBeginDragging:(UIScrollView *)scrollView;
- (void)auRefreshScrollViewDidScroll:(UIScrollView *)scrollView;
- (void)auRefreshScrollViewDidEndDragging:(UIScrollView *)scrollView;

// 结束动画收起列表请调用以下方法
- (void)auRefreshScrollViewDidFinishedLoading:(UIScrollView *)scrollView;

// 需要业务方先滚动到初始位置，然后再调用自动下拉刷新，否则滚动会异常
- (void)autoPullRefreshScrollView:(UIScrollView *)scrollView;

//
- (void)pauseAnimation;

// 页面展开
- (void)resumeAnimation;

@end

@protocol AURefreshViewDelegate <NSObject>

@optional
// 刚刚下拉到默认位置即 (Lottie)View 的高度时触发该协议
- (void)auRefreshViewDidTriggerloading:(AURefreshView *)view;
// 下拉刷新完成复位操作
- (void)auRefreshViewDidDidFinishAnimation:(AURefreshView *)view;


@end
```

代码示例

标准样式如下：

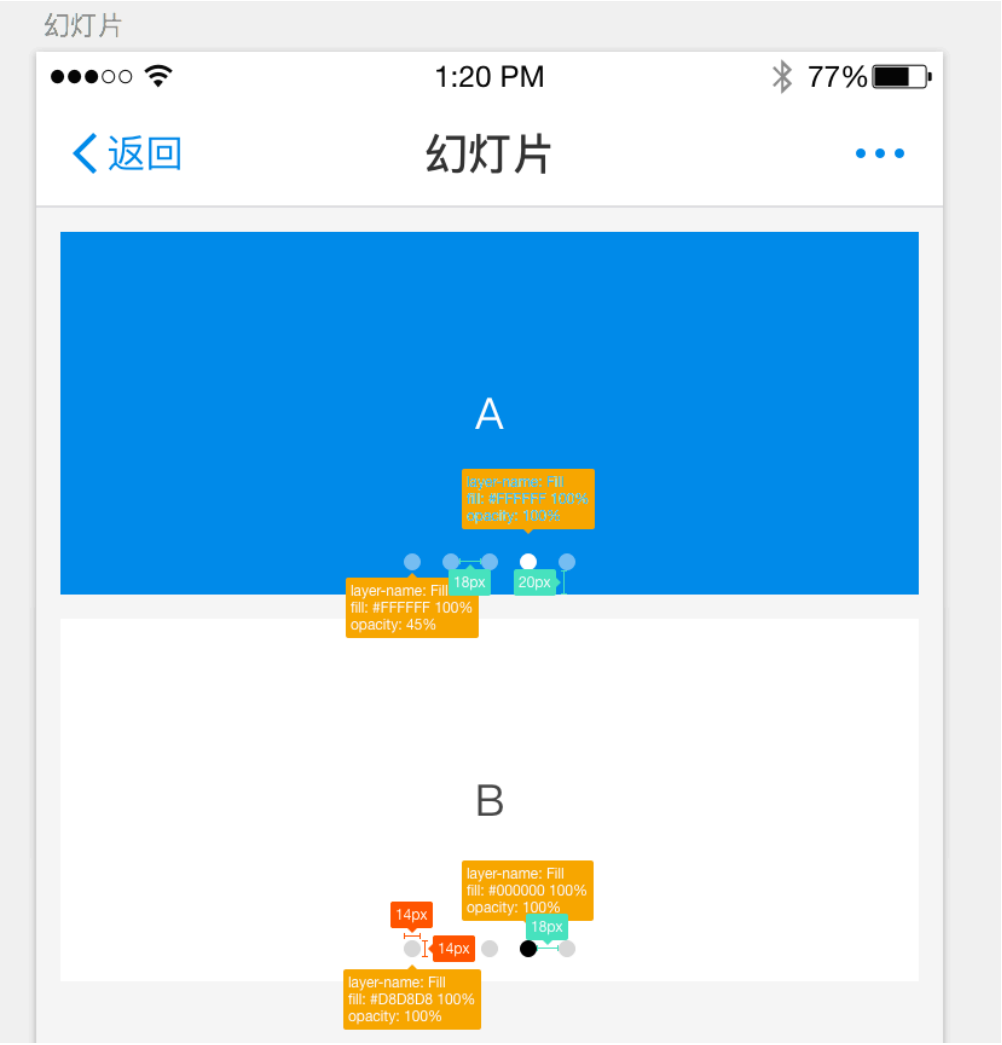
```
_refreshView = [[AURefreshView alloc] initWithSuperView:self.tableView type:AURefreshViewDefault
bizType:@"demo"];
[_refreshView setupLabelText:@"正在刷新"];
[self.tableView addSubview:_refreshView];

- (void)viewDidAppear:(BOOL)animated
{
[super viewDidAppear:animated];

[_refreshView resumeAnimation];
}

- (void)viewDidDisappear:(BOOL)animated
{
[super viewDidDisappear:animated];
[_refreshView pauseAnimation];
}

- (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView
{
[_refreshView auRefreshScrollViewWillBeginDragging:scrollView];
}

- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
[_refreshView auRefreshScrollViewDidScroll:scrollView];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
[_refreshView auRefreshScrollViewDidEndDragging:scrollView];
}
```

5.13 其他组件

5.13.1 轮播组件

AUBannerView 为图片轮播组件。

效果图



接口说明

```
typedef NS_ENUM(NSUInteger, AUBannerStyle) {
AUBannerStyleDeepColor, // 深色样式
AUBannerStyleLightColor // 浅色样式
};

@interface AUBannerViewConfig : NSObject

@property (nonatomic, assign) AUBannerStyle style; // 默认的样式
@property (nonatomic, strong) UIColor *pageControlNormalColor; // 默认色
@property (nonatomic, strong) UIColor *pageControlSelectedColor; // 选中色
@property (nonatomic, assign) CGFloat pageControlMarginBottom; // 分页标识距离底部的margin
@property (nonatomic, assign) BOOL pageControlDotTapEnabled; // 分页标识圆点是否支持点击（默认为NO）
@property (nonatomic, assign) UIEdgeInsets contentViewMargin; // 内容区边距
@property (nonatomic, assign) UIEdgeInsets contentViewPadding; // 内容空白区，滚动时会经过该区域
@property (nonatomic, assign) BOOL autoTurn; // 自动轮播（默认为 YES）
@property (nonatomic, assign) BOOL autoStartTurn; // 自动开始轮播
@property (nonatomic, assign) CGFloat duration; // 自动轮播间隔

@end
```

```
@class AUBannerView;
@protocol AUBannerViewDelegate <NSObject>

@required
- (NSInteger)numberOfItemsInBannerView:(AUBannerView *)bannerView;
- (UIView *)bannerView:(AUBannerView *)bannerView itemViewAtPos:(NSInteger)pos;

@optional
- (void)bannerView:(AUBannerView *)bannerView didTapedItemAtPos:(NSInteger)pos;
- (CGFloat)bannerView:(AUBannerView *)bannerView durationOfItemAtPos:(NSInteger)pos;

@end



@interface AUBannerView : UIView

AU_UNAVAILABLE_INIT

@property (nonatomic, readonly) UIView *contentView; // 内容区视图
@property (nonatomic, readonly) AUPageControl *pageControl; // 分页标识视图

@property (nonatomic, copy) NSString *bizType; // 业务标识
@property (nonatomic, assign) NSInteger currentPage; // 当前页码（从 0 开始）
@property (nonatomic, weak) id<AUBannerViewDelegate> delegate; // 数据源和事件代理


/**
创建banner视图

@param frame frame
@param bizType 业务标识（不能为空）
@param configOperation 配置 block
@return banner 视图
*/
- (instancetype)initWithFrame:(CGRect)frame
bizType:(NSString *)bizType
makeConfig:(void(^)(AUBannerViewConfig *config))configOperation;


/**
开始自动轮播（如果 autoStartTurn 设置为 NO，才需要调用这个方法）
*/
- (void)startTurning;


/**
重新加载 banner（数据源变更，需调用此方法重新加载数据）
*/
- (void)reloadData;

@end
```

```
//#################################
//####### UIImage ################
//#################################

@interface AUBannerView (Image)

/**
创建图片的banner视图
注意：需要保持 imageURLs 和 actionURLs 相等，否则创建失败

@param frame frame
@param bizType 业务标识（不能为空）
@param images 图片集合（可为图片链接字符串，或者 image 对象）
@param placeholder 图片占位图（UIImage 对象）
@param actionURLs 图片点击后的跳转链接（字符串，如果某张图不支持跳转，请设置 [NSNull null]）
@param configOperation banner 视图的配置参数
@return 轮播图片的 banner 视图
*/
+ (instancetype)bannerViewWithFrame:(CGRect)frame
bizType:(NSString *)bizType
images:(NSArray *)images
placeholder:(UIImage *)placeholder
actionURLs:(NSArray *)actionURLs
makeConfig:(void(^)(AUBannerViewConfig *config))configOperation;

@end



//#################################
//####### Extension ###############
//#################################


@interface AUBannerView (Extension)

/**
更新 bannerview 配置
会自动触发 reload 事件

@param update update 的 block
*/
- (void)updateConfigOperation:(void(^)(AUBannerViewConfig *config))update;

@end
```

代码示例

```
// 普通 banner（深色）
for (NSInteger i = 0; i < 1; i ++) {
```

```
CGRect rect = CGRectMake(10, 10 + (height + spaceY) * i, self.view.width - 20, height);
AUBannerView *bannerView = [[AUBannerView alloc] initWithFrame:rect
bizType:@"demo"
makeConfig:^(AUBannerViewConfig *config)
{
config.duration = 1.5;
// config.contentViewMargin = UIEdgeInsetsMake(5, 5, 10, 5);
// config.contentViewPadding = UIEdgeInsetsMake(0, 50, 0, 50);
config.style = AUBannerStyleDeepColor;
config.autoTurn = YES;
config.autoStartTurn = YES;
}];

bannerView.delegate = self;
bannerView.tag = 1;
bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1];
[self.view addSubview:bannerView];
}

// 普通 banner ( 浅色 )
for (NSInteger i = 1; i < 2; i ++) {
CGRect rect = CGRectMake(10, 10 + (height + spaceY) * i, self.view.width - 20, height);
AUBannerView *bannerView = [[AUBannerView alloc] initWithFrame:rect
bizType:@"demo"
makeConfig:^(AUBannerViewConfig *config)
{
config.duration = 1.5;
config.style = AUBannerStyleLightColor;
config.autoTurn = NO;
config.pageControlDotTapEnabled = YES;
}];

bannerView.delegate = self;
bannerView.tag = 2;
bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1];
[self.view addSubview:bannerView];
}

// 纯图片banner
for (NSInteger i = 2; i < 3; i ++) {
CGRect rect = CGRectMake(10, 10 + (height + spaceY) * i, self.view.width - 20, height);
NSMutableArray *images = [NSMutableArray array];
for (NSInteger j = 0; j < 5; j ++) {
UIImage *image = [UIImage imageNamed:[NSString stringWithFormat:@"%@.jpg", @(j + 1)]];
[images addObject:image];
}
AUBannerView *bannerView = [AUBannerView bannerViewWithFrame:rect
bizType:@"demo"
images:images
placeholder:nil
actionURLs:nil
makeConfig:NULL];
bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1];
[self.view addSubview:bannerView];
}
```

```
#pragma mark - AUBannerViewDelegate

- (NSInteger)numberOfItemsInBannerView:(AUBannerView *)bannerView
{
return bannerView.tag == 1 ? 2 : 4;
}

- (UIView *)bannerView:(AUBannerView *)bannerView itemViewAtPos:(NSInteger)pos
{
NSArray *array = nil;
// 深色
if (bannerView.tag == 1) {
array = @[RGB(0x108EE9), RGB_A(0x108EE9, 0.5), [UIColor blueColor], [UIColor yellowColor]];
}
// 浅色
else {
array = @[RGB(0xfFFFFF),RGB_A(0xeFFFFF, 0.7),RGB(0xcFFFFF),RGB_A(0xeFFFFF, 0.5),RGB_A(0xeFFFFF, 0.9)];
}

UIView *view = [[UIView alloc] init];
view.backgroundColor = array[pos];
return view;
}


- (void)bannerView:(AUBannerView *)bannerView didTapedItemAtPos:(NSInteger)pos
{
NSLog(@"didTapedItemAtPos %@", @(pos));
}

//- (CGFloat)bannerView:(AUBannerView *)bannerView durationOfItemAtPos:(NSInteger)pos
//{
// return 1;
//}
```

5.13.2 切换栏组件

AUSegment 遵照新的 UED 需求完成，与 APCommonUI 中的 APSegmentedControl 并不能完全互通，因为 APSegmentedControl 只是封装了系统 UISegmentedControl 而没有其余功能。

效果图

依赖

AUSegment 的依赖为：

```
import"AUSegmentedControlItem.h"
```

接口说明

```
@protocol AUSegmentedControlDelegate <UIScrollViewDelegate>
//AUSegment 点击事件回调
@optional

- (void)didSegmentValueChanged:(AUSegment*)segmentControl;
```

```
- (void)didSelectSegmentItemModel:(AUSegmentItemModel*)selectedItemModel;//

@end

// segment 默认高度
#define AUSegmentHeight AU_SPACE13

/**
分段切换组件
*/
@interface AUSegment : UIScrollView

/**
初始化函数

@param frame frame
@param titles 数组：包含所有标题字符串

@return 返回 AUSegment 实例
*/
- (instancetype)initWithFrame:(CGRect)frame titles:(NSArray<NSString*> *)titles;

/**
禁用 init 方法
*/
- (instancetype)init NS_UNAVAILABLE;

/**
禁用 initWithFrame 方法
*/
- (instancetype)initWithFrame:(CGRect)frame NS_UNAVAILABLE;

/**
AUSegmentedControlDelegate
*/
@property (nonatomic, weak) id <AUSegmentedControlDelegate> delegate;

/**/

/**
标题数组
*/
@property (nonatomic, strong) NSMutableArray *titles;

/**
* 菜单字体
*/
@property (nonatomic, copy) UIFont *titleFont;

/**
当前选中的 segment
*/
@property (nonatomic, assign) NSInteger selectedSegmentIndex;
```

```
/**
选中项的颜色（包括文字和滑块）
*/
@property (nonatomic, copy) UIColor *selecedColor;

/**
* 每个文字菜单水平方向的左右边距
* 默认为 21 像素
* 当为红点样式时，则 fixedItemWidth 不起作用，所有菜单不是固定宽度
*/
@property(nonatomic, assign) NSInteger textHorizontalPadding;

/**
* 是否使用固定菜单宽度
* 默认 YES，方便兼容老的菜单样式
* 当为 YES 时，则 horizontalPadding 不起作用，所有菜单固定同一宽度
*/
@property (nonatomic, assign) BOOL fixedItemWidth;

/**
* 是否自动滚动选中菜单项到合适位置（优先中间位置，不够位置时再靠边显示）
* 默认为 NO
*/
@property (nonatomic, assign) BOOL autoScroll;

/**
* 点击后是否自动更新指示条到当前选中 item 的下标
* 默认为 YES
*/
@property (nonatomic, assign) BOOL autoChangeSelectedIndex;

/*
* model 数组
*/
@property(nonatomic, strong) NSMutableArray<AUSegmentItemModel *> *itemModels;

/**
支持中间插入多项

@param array 插入的标题数组
@param indexes 插入的 indexs
*/
- (void)insertTitleArray:(NSArray<NSString*> *)array atIndexes:(NSIndexSet *)indexes;


/**
支持在末尾新增多项

@param array 新增的标题数组
*/
- (void)addTitleArray:(NSArray<NSString*>*)array;

/**
* 设置自动滚动到指定下标位置，注意：只滚动展示 item，选中指示条保持不变
* 默认与 selectedSegmentIndex（即选中项保持一致）
*/
```

```
- (void)autoScrollToIndex:(NSInteger)index;

- (BOOL)segmentItemIsInVisualAear:(NSInteger)index;

@end


@interface AUSegmentItemModel : NSObject

@property(nonatomic, copy) NSString *title;
@property(nonatomic, copy) UIImage *img;
@property(nonatomic, copy) NSString *imgId;
@property(nonatomic, copy) NSString *badgeNumber;
@property(nonatomic, copy) NSString *badgeWidgetId;
@property(nonatomic, assign) BOOL badgeReserved; // 设置当前 item 是否要预留红点位置，如果不预留则红点展示时界面可能有跳动感
@property(nonatomic, strong) NSDictionary *extendInfo; // 扩展字段

@end


@interface AUSegment (ItemModel)



/**
* 第二版的初始化函数
* @param frame frame
* @param menus item 数组
*/
- (instancetype) initWithFrame:(CGRect)frame menus:(NSArray<AUSegmentItemModel *>*)menus;

/**
支持更新控件 item 项

@param items 需要更新的 item 数组，主要用于增删或者全部更新已有 model 数据
*/
- (void)updateItems:(NSArray<AUSegmentItemModel *>*)items;

/**
支持更新控件item项

@param items 删除已有数据，重新替换为新的 item 数据
*/
- (void)updateItemModel:(AUSegmentItemModel *)model
atIndex:(NSInteger)index;

@end


// 右边显示行动点按钮，默认显示加号 +
@interface AUSegment (AUActionIcon)

- (void)showActionIcon:(BOOL)showIcon target:(id)target action:(SEL)action;

@end
```

自定义属性

属性名	用途	类型
titles	segment 的标题数组	NSArray
selectedSegmentIndex	当前选中的 segment	NSInteger
delegate	实现 AUSegmentedControlDelegate	id
autoScroll	是否自动滚动选中菜单项到合适位置（优先中间位置，不够位置时再靠边显示）	BOOL
fixedItemWidth	是否使用固定菜单宽度	BOOL
textHorizontalPadding	每个文字菜单水平方向的左右边距	BOOL
titleFont	自定义菜单字体	UIFont

代码示例

不带红点的分段控件：

```
NSArray *testArray1 = @[@"tab1",@"tab2",@"tab3",@"tab4",@"tab5",@"tab6",@"tab7",@"tab8"];
AUSegment *segment = [[AUSegment alloc] initWithFrame:CGRectMake(0, 300, self.view.width, 44)
titles:testArray1];
segment.delegate = self;
[self.view addSubview:segment];

//回调
- (void)didSegmentValueChanged:(AUSegment*)segmentControl {
NSLog(@"AUSegmented切换了");
}
```

带红点的分段控件：

```
NSMutableArray *array = [[NSMutableArray alloc] init];
for (int i=0; i<7; i++)
{
AUSegmentItemModel *model = [[AUSegmentItemModel alloc] init];
model.title = [NSString stringWithFormat:@"选项 %d", i];
if (i == 0)
{
model.badgeNumber = @".";
}
if (i == 1)
{
model.badgeNumber = @"new";
}
if (i == 6)
{
model.badgeNumber = @"6";
}
model.badgeReserved = YES;
[array addObject:model];
```

```
}
AUSegment *segment2 = [[AUSegment alloc] initWithFrame:CGRectMake(0, topMargin, self.view.width, 44)
menus:array];
[self.view addSubview:segment2];
[segment2 autoScrollToIndex:6];
segment2.backgroundColor = [UIColor whiteColor];
[segment2 showActionIcon:YES target:self action:@selector(clickActionIcon:)];
```

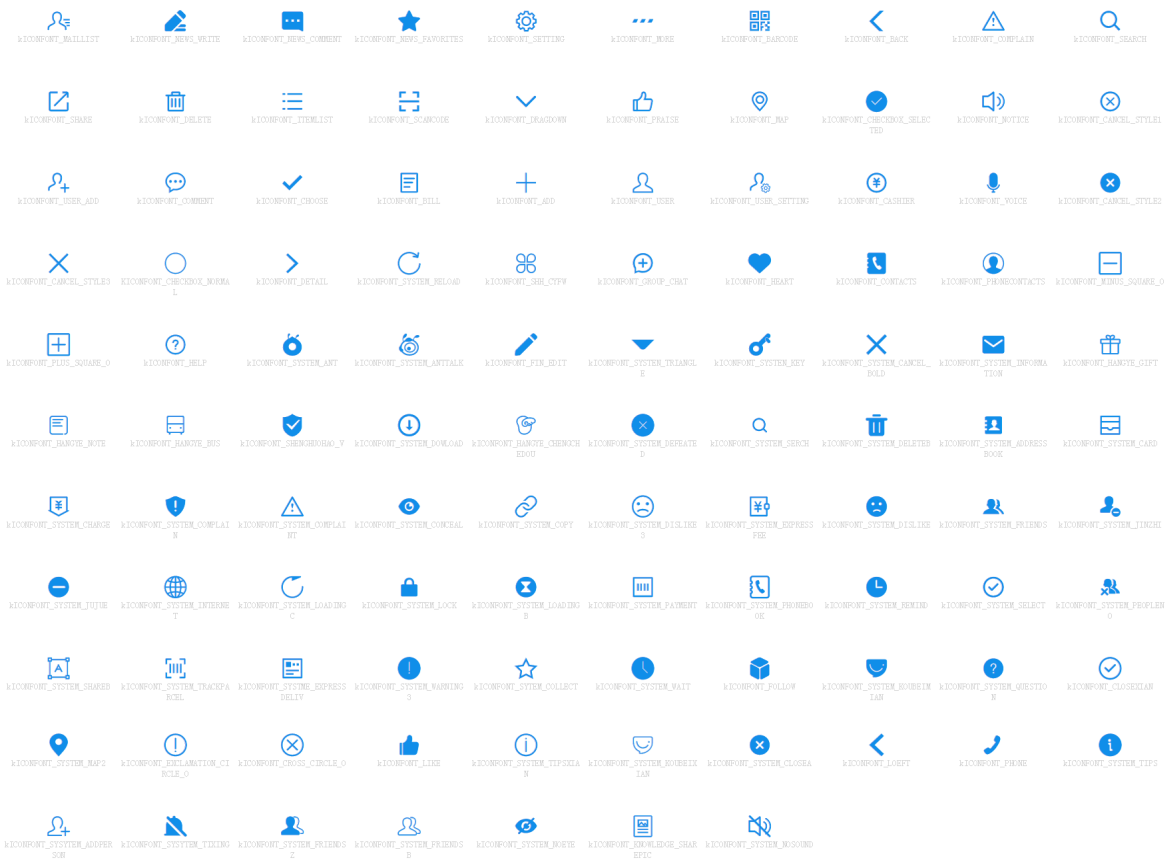
5.13.3 图标组件

- AUIconView 为 iconfont 矢量图控件，使用方式与 UIimage 类似。
- iconfont 图片控件（可以类似当做 imageview 来使用）实际是通过 string 的 drawrect 功能画出的 image 对象。

  注意：目前只支持正方形的矢量图。

- iconfont 相当于是加载了一个字体，一个字体对应了多张图片，每个图片有一个 unicode 码，所以可以设置 text 为对应的 unicode 码，并调用 string 的 drawInRect 方法将 iconfont 渲染出来。
- 每个 iconfont 集合实际就是一个 ttf 字体文件，因此可以加载多个 ttf 字体文件，每个 ttf 字体文件有一个名称，默认为 AntUI 的 ttf 字体，名称为 auiconfont。

效果图



接口说明

```
// AntUI 默认 iconfont 名称
#define kICONFONT_FONTNAME (@"auiconfont")
// AntUI 默认 iconfont 路径
#define kICONFONT_FONTPATH (@"APCommonUI.bundle/iconfont/auiconfont")



/**
iconfont 图片控件（可以类似当做 imageview 来使用）
实际是通过 string 的 drawrect 功能画出的 image 对象，
注意目前只支持正方形的矢量图

iconfont 相当于是加载了一个字体，一个字体对应了多张图片，每个图片有一个 unicode 码，
所以可以设置 text 为对应的 unicode 码，并调用 string 的 drawInRect 方法将 iconfont 渲染出来。

每个 iconfont 集合实际就是一个 ttf 字体文件，因此可以加载多个
ttf 字体文件，每个 ttf 字体文件有一个名称，默认为 AntUI 的 ttf 字体
名称为 auiconfont。
*/
@interface AUIconView : UIImageView

@property (nonatomic, strong) UIColor *color; // 矢量图颜色（默认蚂蚁蓝）
@property (nonatomic, strong) NSString *name; // 矢量图名称
@property (nonatomic, strong) NSString *fontName; // 矢量图字库名称


/**
初始化方法

@param frame 视图 frame
@param name iconfont 图片名称

@return AUIconView实例
*/
- (instancetype)initWithFrame:(CGRect)frame name:(NSString *)name;

/**
初始化方法
（如果该种 iconfont 字体已经加载过，则不需要传入 fontPath 也可以渲染）

@param frame 视图 frame
@param name iconfont 图片名称
@param fontName iconfont 字体名称

@return AUIconView 实例
*/
- (instancetype)initWithFrame:(CGRect)frame name:(NSString *)name fontName:(NSString *)fontName;


/**
获取 iconView 的 size

@return 如果是 iconfont，返回的是 iconfont 的 size，如果是普通的 imageview 返回的是 image 的 size
*/
- (CGSize)iconViewSize;
```

```
@end


@interface UIImage (AUIconFont)

/**
注册 iconfont（只需要调用一次）

@param fontName iconfont 字体名称
@param fontPath iconfont 路径（如 @"AntUI.bundle/iconfont/auiconfont"）
*/
+ (void)registerIconFont:(NSString *)fontName fontPath:(NSString *)fontPath;

/**
获取一个正方形矢量图（长和宽相等）

@param name 名称
@param width 大小
@param color 图像颜色，传 nil，默认为蚂蚁蓝

@return 正方形矢量图
*/
+ (UIImage *)iconWithName:(NSString *)name
width:(CGFloat)width
color:(UIColor *)color;


/**
获取一个正方形矢量图（长和宽相等）

@param name 名称
@param fontName 矢量字体名称
@param width 大小
@param color 图像颜色，传 nil，默认为蚂蚁蓝

@return 正方形矢量图
*/
+ (UIImage *)iconWithName:(NSString *)name
fontName:(NSString *)fontName
width:(CGFloat)width
color:(UIColor *)color;

@end
```

代码示例

```
// 使用 AUIconView
AUIconView *view = [[AUIconView alloc] initWithFrame:CGRectZero name:_array[indexPath.row]];
view.tag = 1;

view.size = CGSizeMake(30, 30);
view.origin = CGPointMake(100, 10);
view.color = RGB(0x2b91e2);
```

```
[cell.contentView addSubview:view];

// 单独使用 image 的扩展
self.image = [UIImage iconWithName:self.name fontName:self.fontName width:width color:self.color];
```

5.13.4 索引组件

效果图

接口说明

AUBladeView.h

```
//
```

```
// indexBar.h
//

#import <Foundation/Foundation.h>

#define kIndexSearchTitle @"搜"

@protocol AUBladeViewDelegate;
/* !
@class AUBladeView
@abstract UIView
@discussion 字母索引的 View
*/
@interface AUBladeView : UIView

- (id)init;
- (id)initWithFrame:(CGRect)frame;
- (void)clearIndex;

@property (nonatomic, weak) id<AUBladeViewDelegate> delegate;
@property (nonatomic, strong) UIColor *highlightedBackgroundColor;
@property (nonatomic, strong) UIColor *textColor;
@property (nonatomic, strong) UIFont *textFont;
@property (nonatomic, strong) NSArray * iconImageNames;
@property (nonatomic, strong) NSArray * iconTitles;
@property (nonatomic, assign) BOOL enableSearch;
@property (nonatomic, strong) NSArray * defaultIndexes;


- (void)updateIndexes;

@end

@protocol AUBladeViewDelegate<NSObject>
@optional
- (void)indexSelectionDidChange:(AUBladeView *)indexBar index:(NSInteger)index title:(NSString*)title;
@end
```

代码示例

```


//
// bladeViewController.m
// AntUI
//

#import"bladeViewController.h"
#import"AUBladeView.h"

@interface bladeViewController ()<AUBladeViewDelegate,UITableViewDelegate,UITableViewDataSource>
@property (nonatomic,strong) AUBladeView * bladeView;
@property (nonatomic,strong) NSArray * sectionArr;
@property (nonatomic,strong) NSArray * mainDataIndexChar;
@property (nonatomic,strong) UITableView * tableView;
@end
```

```
@implementation bladeViewController

- (void)viewDidLoad {
[super viewDidLoad];
self.title = @"城市选择";
// Do any additional setup after loading the view.
self.tableView = [[UITableView alloc] initWithFrame:CGRectMake(0, 0, self.view.frame.size.width,
self.view.frame.size.height) style:UITableViewStylePlain];
self.tableView.delegate = self;
self.tableView.dataSource = self;
[self.view addSubview:self.tableView];
self.bladeView = [[AUBladeView alloc] initWithFrame:CGRectMake(self.view.frame.size.width-16.0, 80, 16.0,
self.view.bounds.size.height-60)];
self.bladeView.delegate = self;

NSString * plistStr = [[NSBundle mainBundle]
pathForResource:@"APCommonUI_ForDemo.bundle/citydict"ofType:@"plist"];
NSDictionary * srcPlistDic = [NSDictionary dictionaryWithContentsOfFile:plistStr];
NSMutableArray * citysList = [[NSMutableArray alloc] initWithCapacity:27];
[citysList addObject:@{@"热门城市":@[@"上海",@"杭州",@"广州",@"北京",@"深圳"]}];

NSMutableArray * indexArrList = [[NSMutableArray alloc] initWithCapacity:27];
[indexArrList addObject:@"热"];
NSArray * keyList = [srcPlistDic allKeys];
NSArray * sortedList = [keyList sortedArrayUsingComparator:^NSComparisonResult(id _Nonnull obj1, id _Nonnull
obj2) {
return [(NSString *)obj1 compare:obj2];

}];
[sortedList enumerateObjectsUsingBlock:^(id _Nonnull obj, NSUInteger idx, BOOL * _Nonnull stop) {
if (obj) {

NSDictionary * tmpDic = [[NSDictionary alloc] initWithObjectsAndKeys:[srcPlistDic objectForKey:obj],obj, nil];
[citysList addObject:tmpDic];
[indexArrList addObject:obj];
}

}];
self.sectionArr = citysList;
self.mainDataIndexChar = indexArrList;
// self.bladeView.iconTitles = secondaryIndexsTitles;
// self.bladeView.iconImageNames = secondaryIndexsIcons;
self.bladeView.defaultIndexes = self.mainDataIndexChar;
[self.bladeView updateIndexes];
[self.view addSubview:self.bladeView];
self.tableView.showsVerticalScrollIndicator = NO;
self.tableView.showsHorizontalScrollIndicator = NO ;
[self.view bringSubviewToFront:self.bladeView];


}


#pragma mark -----UITableViewDelegate
// Display customization
```

```
// Variable height support

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
return 44;
}// Use the estimatedHeight methods to quickly calcuate guessed values which will allow for fast load times of the table.

- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section
{
return 35;
}
#pragma mark -----UITableViewDataSource
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
NSDictionary * test = [self.sectionArr objectAtIndex:section] ;
NSArray * valueArr = [test objectForKey:([[test allKeys] firstObject])];

return [valueArr count];

}

- (nullable NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section
{
NSDictionary * test = [self.sectionArr objectAtIndex:section] ;

return [[test allKeys] firstObject];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"BladeTableViewCell"];
if (!cell) {
cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"BladeTableViewCell"];
}
NSInteger section = [indexPath section];
NSInteger row = [indexPath row];
NSDictionary * test = [self.sectionArr objectAtIndex:section] ;
NSArray * valueArr = [test objectForKey:([[test allKeys] firstObject])];
NSString * text = [valueArr objectAtIndex:row];
cell.textLabel.text =text ;
return cell;

}
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return [self.sectionArr count];
}// Default is 1 if not implemented

- (void)dealloc
{
self.tableView.delegate = nil;
self.tableView.dataSource = nil;
self.bladeView.delegate = nil;
self.bladeView = nil;
```

```
}
- (void)didReceiveMemoryWarning {
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
// Get the new view controller using [segue destinationViewController].
// Pass the selected object to the new view controller.
}
*/

#pragma mark ---- AUBladeViewDelegate
- (void)indexSelectionDidChange:(AUBladeView *)indexBar index:(NSInteger)index title:(NSString*)title
{
if (self.tableView){
NSInteger ret = 0;
if ([title isEqualToString:kIndexSearchTitle]){
[self.tableView scrollToRowAtIndexPath:[NSIndexPath indexPathForRow:0 inSection:ret]
atScrollPosition:UITableViewScrollPositionBottom
animated:NO];
return;
}
else {
ret = [self findIndexSection:title];
if (ret != NSNotFound) {
[self.tableView scrollToRowAtIndexPath:[NSIndexPath indexPathForRow:0 inSection:ret]
atScrollPosition:UITableViewScrollPositionTop
animated:NO];
}
}
}

}

- (NSInteger)findIndexSection:(NSString *)title {
NSInteger ret = NSNotFound;
int beginIndex = 0;
/*
以下为自定义的 section 计算规则
beginIndex += self.customSectionCount;
if (self.secondarySectionCount > 0 ){
for (NSInteger i = 0 ; i < [self.secondarySectionTitles count]; i++) {
NSString * secondaryTitle = [self.secondarySectionTitles objectOrNilAtIndex:i];
if ([secondaryTitle isEqualToString:title]) {
ret = beginIndex + i;
return ret;
}
}
beginIndex += self.secondarySectionCount;
}
*/
```

```
if ([self.mainDataIndexChar count] > 0){
for(NSInteger i = 0; i < [self.mainDataIndexChar count]; i++) {
NSString * indexChar = [self.mainDataIndexChar objectAtIndex:i];
if ([indexChar isEqualToString:title]) {
ret = i;
break;
}
}
}
if (ret != NSNotFound) {
ret = ret + beginIndex;
}
return ret;
}

@end
```

5.13.5 导航栏切换组件

AUTitleBarSegment 只是封装系统 UISegmentedControl 简单修改其 UI 样式，提供默认高度以及每段默认宽度。

效果图

接口说明

```
/*
mPaaS 标准：该分段控件只能用于导航栏顶部的分段控件
具有默认色值，默认高度 52px
*/

#define AUTitleBarSegment_DefaultHeight 26 // 默认高度值为 26
#define AUTitleBarSegment_DefaultSegmentWidth 90 // 默认每段的宽度值为 90

@interface AUTitleBarSegment : UISegmentedControl

@end
```

代码示例

```
AUTitleBarSegment *titleBatSegment = [[AUTitleBarSegment alloc] initWithItems:@[@"标签", @"标签"]];
self.navigationItem.titleView = titleBatSegment;
```

5.13.6 导航按钮

- AUBarButtonItem 为 UIBarButtonItem 在 mPaaS 的版本（包括了预定义颜色和字体等）。
- 为方便后续扩展，所有 mPaaS 应用都必须使用 AUBarButtonItem 而不是系统的 UIBarButtonItem。
- 目前 AUBarButtonItem 完全继承自 AUSwitch，并未额外添加属性和方法。

接口说明

```
/**
AUBarButtonItem 为 UIBarButtonItem 在 mPaaS 的版本（包括了预定义颜色和字体等）。
为方便后续扩展，所有 mPaaS 应用都必须使用 AUBarButtonItem 而不是系统的 UIBarButtonItem。
需要和 AUNavigationBar 一起使用
*/
@interface AUBarButtonItem : UIBarButtonItem

@property(nonatomic, strong) NSString *backButtonTitle; // 返回按钮 title
@property(nonatomic, strong) UIImage *backButtonImage; // 返回按钮图片
@property(nonatomic, strong) UIColor *titleColor; // 返回按钮文本颜色

/**
* 设置按钮间的间距
*
* @return 返回 UIBarButtonSystemItemFlexibleSpace 风格的空按钮
*/
+ (AUBarButtonItem *)flexibleSpaceItem;

/**
* 创建默认的返回按钮样式
*
* @param title 显示文本
* @param target 点击接受者
* @param action 点击处理方法
*
* @return APBarButtonItem
*/
+ (AUBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title target:(id)target action:(SEL)action;

/**
* 创建默认的返回按钮样式
*
* @param title 显示文本
* @param count 最大显示文字数
* @param target 点击接受者
* @param action 点击处理方法
*
* @return APBarButtonItem
*/
+ (AUBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title maxWordsCount:(NSInteger)count
target:(id)target action:(SEL)action;

@end
```

代码示例

```
// 定义一个 backBarItem
// 默认包含了一个返回的 icon 图片
AUBarButtonItem *cancelItem = [AUBarButtonItem backBarButtonItemWithTitle:@"返回"target:self
action:@selector(cancel)];
cancelItem.backButtonTitle = @"取消";
self.navigationItem.leftBarButtonItem = cancelItem;

AUBarButtonItem *rightItem1 = [[AUBarButtonItem alloc] initWithImage:image1 style:UIBarButtonItemStylePlain
target:self action:@selector(rightBarItemPressed)];
```

5.13.7 适配依赖

作为 AntUI 的外壳，AntUIShell 主要用去实现 AntUI 中第三方的协议，可以用于内嵌入 mPaaS 应用使用，并且减少 AntUI 对外界的依赖关系。

接口说明

AntUIShellObject.h

```
//
// AntUIShellObject.h
// AntUIShell
//

#import <Foundation/Foundation.h>
#import <AntUI/AntUI.h>

@interface AntUIShellObject : NSObject<AUThirdPartyAdapter>

@end
```

代码示例

```
//
// AntUIShellObject.m
// AntUIShell
//

#import"AntUIShellObject.h"
#import <APMonitor/APMonitor.h>
#import <APMultimedia/APMultimedia.h>
#import <MPBadgeService/MPBadgeService.h>

@implementation AntUIShellObject


#pragma mark ----AUThirdPartyAdapter
/*************************************************************/
//图片协议 APMultimedia
```

```
/*
第三方适配下载图片接口
主要对多媒体接口进行包装，由第三方实现
*/
- (NSString *)thirdPartyGetImage:(NSString *)identifier
business:(NSString *)business
zoom:(CGSize)size
originalSize:(CGSize)originSize
progress:(void (^)(double percentage,long long partialBytes,long long totalBytes))progress
completion:(void (^)(UIImage *image, NSError *error))complete
{
return [[APImageManager manager] getImage:identifier business:business zoom:size originalSize:originSize
progress:progress completion:complete];

}

/*
第三方适配 uiimageview 下载图片接口
由第三方去实现。
*/
- (void)thirdPartypFromImageView:(UIImageView *)fromImgView
setImageWithKey:(NSString *)key
business:(NSString *)business
placeholderImage:(UIImage *)placeholder
zoom:(CGSize)zoom
originalSize:(CGSize)originalSize
progress:(void (^)(double percentage,long long partialBytes,long long totalBytes))progress
completion:(void (^)(UIImage *image, NSError *error))complete
{
if(fromImgView && [fromImgView isKindOfClass:[UIImageView class]]) {
[fromImgView setImageWithKey:key business:business placeholderImage:placeholder zoom:zoom
originalSize:originalSize progress:progress completion:complete];
}
}
/************************************************************/
//红点协议 MPBadgeService
/*
初始化红点 View
*/
- (UIView *) thirdPartyBadgeViewWithFrame:(CGRect)frame
{
return [[MPBadgeView alloc] initWithFrame:frame];
}

/*
红点设置 widgetId

*/
- (void) thirdPartyBadgeViewWith:(UIView *)badgeView
widgetId:(NSString *) widgetId
{
if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
tmpBadgeView.widgetId = widgetId;
}
```

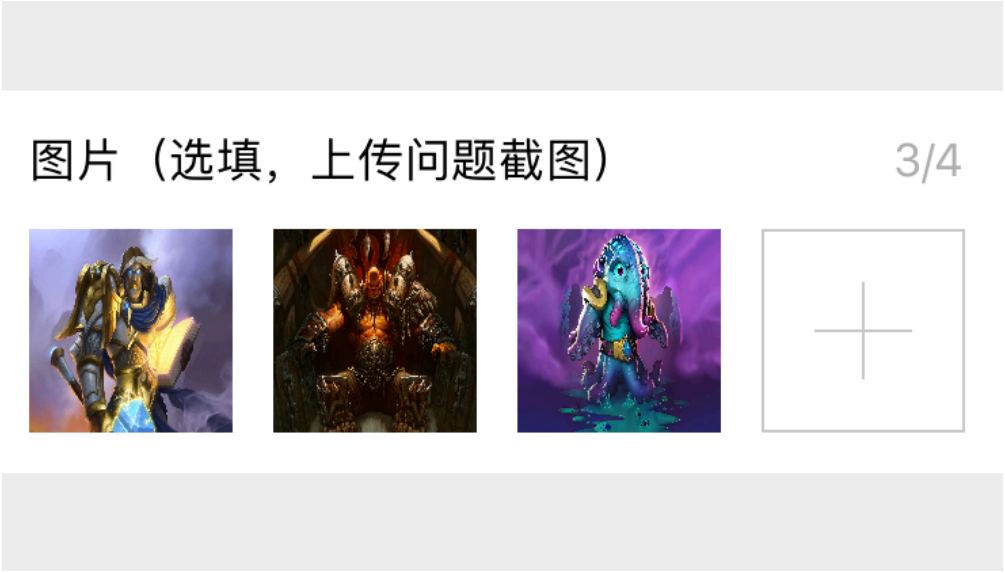
```
}
/*
注册红点 view 到 MPBadgeManager 管理者。
*/
- (void) thirdPartyBadgeViewReg:(UIView *)badgeView
{
if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
[[MPBadgeManager sharedInstance] registerBadgeView:tmpBadgeView];
}

}

/**
* 更新显示“红点”样式
* @param badgeView 红点 View
* @param badgeValue: @"."显示红点
* @"new"显示 new
* @"数字"显示数字，大于 99 则显示图片 more ( ... )
* @"惠"/"hui"显示“惠”字
* @"xin"显示"新"字
* nil 清除当前显示
*
* @return 无
*/
- (void) thirdPartyBadgeViewWith:(UIView *)badgeView
updateValue:(NSString *)badgeValue
{
if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
[tmpBadgeView updateBadgeValue:badgeValue];
}
}

/*
提供业务监控红点控件刷新接口。
widgetInfo 类型是 MPWidgetInfo
*/
- (void) thirdPartyBadgeViewWith:(UIView *)badgeView
updateBlock:(void(^)(id widgetInfo, BOOL isShow)) updateBlock
{
if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
if(updateBlock) {
tmpBadgeView.updateBlock = updateBlock;
}
}

}

/*
埋点协议 APMonitor
*/
//按钮的 actionName 的埋点协议
- (void) thirdPartySetButtonActionLog:(UIButton *)button
actionNameLog:(NSString *)actionName
```

```
{
if(button && [button isKindOfClass:[UIButton class]]) {
button.actionName = actionName;
}
}

/*
通知协议 (AUCardMenu/AUFloatMenu)
*/

/*
AUCardMenu 注册退出登录的通知，保证退出登录 AUCardMenu 能够及时销毁
*/
- (NSString *) thirdPartyCardMenuDismissNotiName
{
return @"SAAccountDidExitNotification";
}

/*
AUFloatMenu 注册 alerView kShareTokenAlertViewShownNotification
*/
- (NSString *) thirdPartyFloatMenuDismissFromAlertNotiName
{
return @"kShareTokenAlertViewShownNotification";
}

/*
AUFloatMenu 注册 alerView SALoginAppWillStartNotification
*/
- (NSString *) thirdPartyFloatMenuDismissFromLoginNotiName
{
return @"SALoginAppWillStartNotification";
}


@end
```

5.13.8 选图组件视觉与交互封装

- AUImagePickerSkeleton 是选图组件视觉与交互的封装，不包括调用相册、浏览、上传功能。
- 目前暂无完整的选图功能，后续会在 BEEViews 中增加继承了功能的组件。

效果图



依赖

暂未加入AntUI基线

接口说明

```
@protocol AUImagePickerDataProtocol <NSObject>

- (UIImage *)image;

@end

@interface AUImagePickerSkeleton : UIView

- (AUImagePickerSkeleton *)initWithTitle:(NSString *)title
maxNumberOfImages:(NSUInteger)maxNumberOfImages;

@property(nonatomic, assign, readonly) NSUInteger maxNumberOfImages;
@property(nonatomic, weak) id<AUImagePickerDelegate> delegate;
@property(nonatomic, strong, readonly) NSArray<id<AUImagePickerDataProtocol>> *imagePickerDatas;

- (void)updateImagePickerDatas:(NSArray <id<AUImagePickerDataProtocol>>*) datas;

@end

@protocol AUImagePickerDelegate <NSObject>

@required
- (void)imagePickerAddButtonClick:(AUImagePickerSkeleton *)imagePicker;

@optional
- (void)imagePickerImageClick:(AUImagePickerSkeleton *)imagePicker
clickData:(id<AUImagePickerDataProtocol>)clickData;
@end
```

代码示例

```
- (void)viewDidLoad {
[super viewDidLoad];
self.datas = [[NSMutableArray alloc] init];
self.picker = [[AUImagePickerSkeleton alloc] initWithTitle:@"图片（选填，上传问题截图）"maxNumberOfImages:4];
self.picker.top = 100;
self.picker.delegate = self;
[self.view addSubview:self.picker];
[self.view addSubview:self.button];
self.view.backgroundColor = RGB(0xEBEBEB);
}

- (void)imagePickerAddButtonClick:(AUImagePickerSkeleton *)imagePicker
{
AUImagePickerData *data = [AUImagePickerData new];
data.originalImage = [self getImageWithCount:[self.datas count]];
[self.datas addObject:data];
[self updatePickerAndResize];
}

-(void)imagePickerImageClick:(AUImagePickerSkeleton *)imagePicker
clickData:(id<AUImagePickerDataProtocol>)clickData
{
NSString *msg = @"";
if ([self.datas containsObject:clickData]) {
msg = [NSString stringWithFormat:@"点击第%d张图片",(int)[self.datas indexOfObject:clickData]+1];
}else{
msg = @"点击的图片发生异常";
}
[AUToast presentModalToastWithin:self.view
withIcon:AUToastIconNone
text:msg
duration:1
logTag:@"demo"
completion:NULL];

}
```


蚂蚁集团
ANT GROUP