iOS13简单适配

yuyanglk 关注
发表于 2019-10-17 08:28:51 字数 1,437 阅读 12,230

iOS13的beta版本已经出来了，APP适配也快来了，这篇文章主要介绍一些适配时候的一些工作。

Note:iOS13 及更新版本的主线程中

Q：我不想让用户进入Dark模式，怎不可以？
A：当然可以，请往下看

在iOS13，为 UIUserInterfaceStyle 和 UITrait 扩展了一个新的API：overrideUserInterfaceStyle，使用方法，官方文档大致意思是这样的：

> 通过设置 overrideUserInterfaceStyle 属性以请求视图及其子视图将特定的 UserInterfaceStyle，但如果想要获取当前的 UserInterfaceStyle，需要改用 traitCollection.userInterfaceStyle。
>
> 在可能值的 UIUserInterface 上的 overrideUserInterfaceStyle 属性，仅在以下时候使用此属性：
>
> - 在单个视图或一组视图之间切换主题或使用特定样式。
> - 在希望为某个 UIWindow 及其视图控制器和模态表单设置 overrideUserInterfaceStyle 时。这不是安装强制更改整个应用程序表现样式。（比如说强制希望整个应用程序表现暗样式，那不要使用这，而是在info.plist中设置 UIUserInterfaceStyle 键。）
>
> 1. 当设置改变通知 UITrait 之时
>
> - 此属性仅影响此视图及其子视图的样式。
> - 它不会影响任何视图控制器或其他视图控制器的子视图。
>
> 2. 在 UIWindow 之设置时：
>
> - 此属性会影响 rootViewController 以及此窗口中视图控制器和视图层次结构。
> - 它还会影响通过 present 呈现的视图控制器。

最近可见，overrideUserInterfaceStyle 不仅会影响自己，还会影响自己的子视图，所谓 window 就会影响整个 window 中的所有视图及视图控制器，但是根据苹果的说法和测试结果，overrideUserInterfaceStyle 不会影响自己的父视图。

所以，这种特性带来了，你可以设置整个应用程序的主题和样式，就算大部分应用公司已经 info.plist 配置了 User Interface Style，并且支持Dark Light Dark。

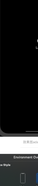
info.plist 应用程序配置

适配主要涉及的方面：

- 模拟器测试(simulator debug)
- 图片(assets)
- 颜色(color)
- 状态栏(status bar)
- 模态展示(modal present)-2019.10.3更新
- ...

模拟器测试

1、运行项目
2、在左Xcode底部菜单栏中 Environment Overrides
3、开启Interface Style，就可以切换了。



切换Dark&Light模式

模拟器打开Overrides.png

图片适配

图片适配，主要是我们未他图片资源适配，种他图片的话，还是比较繁琐的，目前我们还没有，只做了本地图片适配。
图片适配还为了便利而更通过 Assets.xcassets 进行图片管理。

1、进入一个Image set，重命名为"adaptimage"，并中添加image set;
2、选中Attributes Inspector;
3、将Appearances的"None"改为"Any,Dark"。
4、不同模式下设置不同图片即可，mode 设置会自动选择不同的图片

Assets.png

当然图片资源，你也可以直接使用种独自适应的mode的方式进行设置，就跟个人用基不喜欢那些麻烦，因为这要要求多种模式的图片，最终UIImageAssets中应该会 initWithCatalogName:_;_;_;我们有电子设计师支持：

```
/** Track environments expose a trait collection that defines the environment's
 */ public protocol UITraitEnvironment : NSObjectProtocol {
    // MainThreadOnly API 8.0.
    var traitCollection: UITraitCollection { get }
    // This has overridden as needed to provide custom behavior when the environment's
    // traits change.
    func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?)
}
```

所以，我们只需要在改变系统mode的时候，重写方法：

```
func updateDisplayMode() {
    let image = UIImageCollection.useInterfaceStyle == .light ? UIImage(named: "dark-ic
    tagsBtn.setImage = image
}
// override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?)
// func traitCollectionDidChange(previousTraitCollection)
    updateDisplayMode()
}
```

代码还配置了，还有一种简单的方式，这种方式有需要在Assets.xcassets中同时添加Dark/Any两种模式下图片，那就那样设置这部分手动设置assetsImage图片一样。

```
let image = UIImage(named: "ic")!imageAsset!.imageForUI(traitCollection)
tagsBtn.setImage(image, image)
```

颜色适配

像图片适配一样，颜色适配有三种方式，
而第一一是通过 Assets.xcassets 添加一个 Color Set，目前系统是iOS11.0

```
extension UIColor {
    // MainThreadOnly API 8.0.
    public class func color(name: String) -> UIColor? // Looks first main bundle.
    // MainThreadOnly API 11.0.0.
    public class func color(name: String, in bundle: Bundle?, compatibleWith
    }
```

color.set.png

方法二：代码创建动态颜色 init(dynamicProvider: Resolving (UITraitCollection) -> UIColor)，目前系统支持iOS 13.0

```
// iOS 13.0
let dynamicColor = UIColor.init(dynamicProvider: { (trait) -> UIColor in
    return trait.userInterfaceStyle == .light ? UIColor.black : UIColor.white
})
let myCollection(traitCollection, for: nsmap)
```

方法三：像图片一样，这种模式同样支重写 traitCollectionDidChange(_:)方法，不推荐

状态栏(StatusBar)

目前状态栏也增加了一种模式，自之前的两种，变成了三种，默认 default 自之前的黑色内容，变成了白色或黑色模式，自动跟随当前模式 light/Lumen 还是 darkLumen。

```
public class UIStatusBarStyle : NSEnum {
    // default 7.0.0.0
```