链滴

XPage 系列 | 这次升级后终于是全自动化注册了!

作者：xuexiangjys
原文链接：https://ld246.com/article/1610303112921
来源网站：链滴

前言

作为 X-Library系列框架 的灵魂所在，XPage 开源两年以来，一直致力于降低Fragment使用的难度努力实现一个Activity多Fragment的Android开发模式。

就在前不久，我就整理了XPage开源这几年来的使用情况，写了一篇《史上最方便的Android页面框XPage使用指南》 ,并且还录了几期视频单独讲解了XPage的使用 ，让越来越多地人看到了XPage使的便捷性。

但就在前几天，在交流群里突然有人问我下面几个问题：

- 1.我如果想在多个module中使用XPage，我该怎么办呀?
- 2.为什么我使用XPage之后，一直找不到 AppPageConfig这个类啊?

上面的问题让我突然认识到一点：并不是所有人都对APT技术有所了解的。

看来我之前参考 ARouter实现的自动注册功能可能并没有完善，难怪 ARouter后来会写一个 arouter-egister插件来实现自动注册的功能。

于是乎，为了能够让XPage的自动注册功能更加完美，我加班加点开发，于是就有了XPage的3.1.1版--彻底的自动化注册 。

升级后有什么变化

在感受全自动化页面注册带来的便利之前，让我们先来感受一下之前版本的使用。

3.1.1之前版本

在3.1.1之前版本，在使用自动注册功能的时候，还是需要实现 PageConfiguration接口，并调用编时自动生成的页面配置类“moduleName”+PageConfig 的getPages()方法返回注册的页面。

```
PageConfig.getInstance()
        .setPageConfiguration(new PageConfiguration() { //页面注册
            @Override
            public List<PageInfo> registerPages(Context context) {
                //自动注册页面,是编译时自动生成的，build一下就出来了。如果你还没使用@Page的话暂时是不会生成的。
                return AppPageConfig.getInstance().getPages(); //自动注册页面
            }
        })
        .debug("PageLog")       //开启调试
        .setContainActivityClazz(XPageActivity.class) //设置默认的容器Activity
        .enableWatcher(false)   //设置是否开启内存泄露监测
        .init(this);
```

可以看到，这里的自动注册还是需要一部分手动配合才能完成的。如果说你当前只有一个module的，可能还好说。但是如果你使用了多个module之后，你就需要把多个module生成的配置类像上面那一个一个地加进去，这样用起来会让人感觉非常的不方便，这明显违背了我写 XPage框架的初衷！

不仅如此，这样写死还会带来其他很多问题：

- 1.如果module名变了，还需要对应地去修改配置类的类名。
- 2.如果当前module没有使用 @Page注解修饰Fragment的话，配置类也不会自动生成，这样会很多初次使用者非常疑惑。
- 3.项目要是没有编译过的话，配置类是不会自动生成的，这样代码就会报错说类找不到，然后很多手就懵逼了。

3.1.1之后版本

为了能够解决以上的问题，我实现了一个自动注册的配置类AutoPageConfiguration 。那么下面就一下最新版本的XPage是如何注册的吧：

```
PageConfig.getInstance()
        .debug("PageLog")       //开启调试
        .setContainActivityClazz(XPageActivity.class) //设置默认的容器Activity，按需设置（非必须

        .init(this);            //初始化页面配置
```

是的，你没有看错，这里没有手动实现 PageConfiguration接口的部分了，可以说是真正实现了全自页面注册，是不是非常方便呀？

如何实现注册的自动化

看到上面的变化，你是不是非常想知道我是如何实现彻底的自动化注册的？

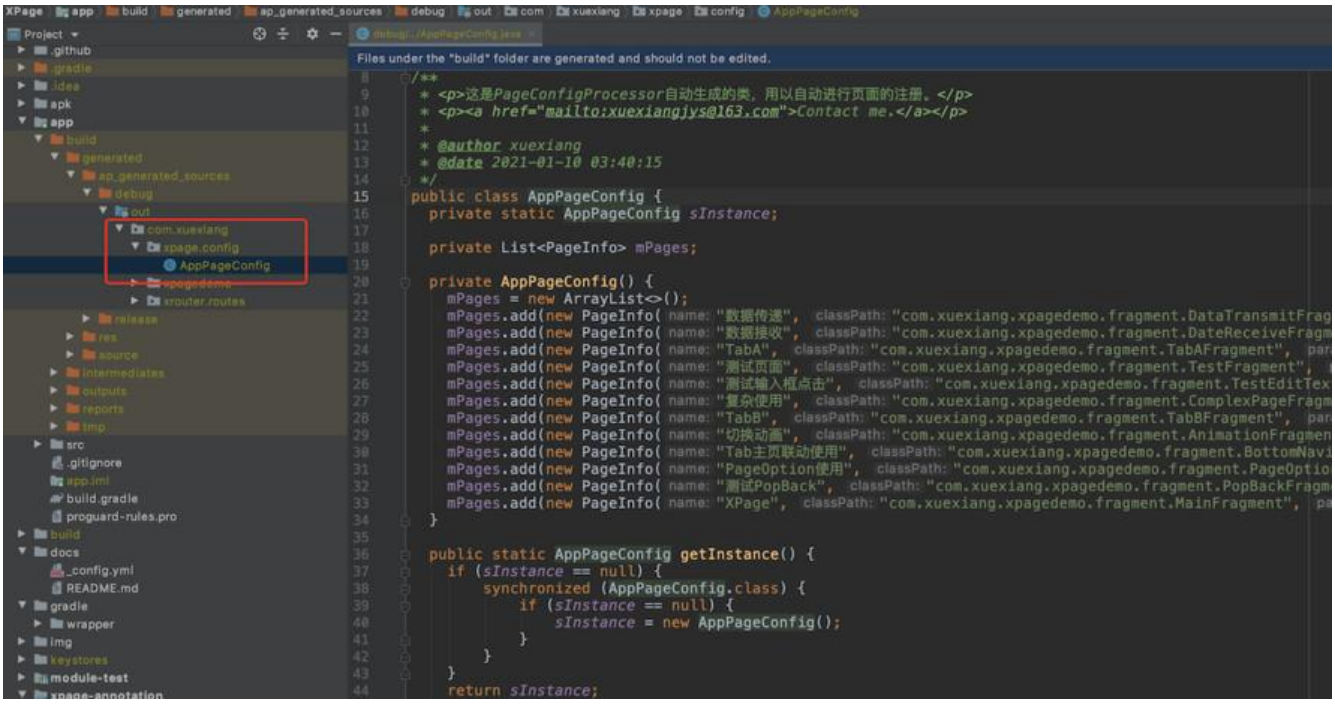
想要回答这个问题，还是让我们先看一看这个编译时自动生成的配置类是如何实现的。

APT技术实现页面配置类的自动生成

其实当初实现页面配置类的自动生成的方案，也是我研读了ARouter源码之后，受到了APT技术的启后完成的。

因为XPage实现路由跳转主要就是靠 PageInfo 和 Fragment 建立起来的映射关系。当时的思路就是用APT技术，利用 @Page注解去标识需要注册的Fragment，然后在编译的时候通过APT技术去扫描所有使用了 @Page注解标识了的Fragment，将注解信息转化为 PageInfo, 并按module生成对应的面配置类，在这个配置类里面存放了该module下所有标注了 @Page的页面信息 PageInfo。

下面是自动生成的一个简单的配置类例子：



可以看到，自动生成的配置类都会存放在 com.xuexiang.xpage.config包下，类名都是以 PageConfig为结尾。注意这里非常关键，它是我后面实现自动化注册的关键。

详细的实现细节，可以参见XPage的页面配置自动生成器PageConfigProcessor的源码 。

运行时扫描指定包下的配置类反射实现自动注册

上面我们在了解页面配置类是如何自动生成的时候发现一个规律：

自动生成的配置类都会存放在 com.xuexiang.xpage.config包下，类名都是以 PageConfig作为结尾。

那么我们可不可以在运行的时候，直接扫描 com.xuexiang.xpage.config包下的所有类，然后找到以ageConfig作为结尾的配置类，然后反射它的 getPages方法直接获取到所有的配置信息，然后注册去?

下面是我根据上面的猜想，实现的AutoPageConfiguration类源码：

```
public class AutoPageConfiguration implements PageConfiguration {
    /**
     * 页面配置所在的包名
     */
    private static final String PAGE_CONFIG_PACKAGE_NAME = "com.xuexiang.xpage.config";
    /**
     * 页面配置生成类的类后缀名
     */
    private static final String PAGE_CONFIG_CLASS_NAME_SUFFIX = "PageConfig";

    @Override
    public List<PageInfo> registerPages(Context context) {
        List<PageInfo> pageInfos = new ArrayList<>();
        Set<String> classSet = null;
        try {
```

```
            classSet = ClassUtils.getClassNames(context, PAGE_CONFIG_PACKAGE_NAME);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (classSet != null) {
            for (String className : classSet) {
                if (className != null && className.endsWith(PAGE_CONFIG_CLASS_NAME_SUFFIX
) {
                    try {
                        pageInfos.addAll(getPagesByClass(Class.forName(className)));
                    } catch (Exception e) {
                        PageLog.e(e);
                    }
                }
            }
        }
        return pageInfos;
    }

    private List<PageInfo> getPagesByClass(Class<?> clazz) throws Exception {
        // 获取单例对象
        Method getInstanceMethod = clazz.getDeclaredMethod("getInstance");
        getInstanceMethod.setAccessible(true);
        Object instance = getInstanceMethod.invoke(null);
        // 获取页面信息
        Method getPagesMethod = clazz.getDeclaredMethod("getPages");
        getPagesMethod.setAccessible(true);
        return (List<PageInfo>) getPagesMethod.invoke(instance);
    }
}
```

从源码中我们可以看到，我是这样做的：

- 1.使用 ClassUtils.getClassNames获取到com.xuexiang.xpage.config包下的所有类的类名。这的ClassUtils也是我借鉴了ARouter里面的源码。
- 2.遍历这个类名集合，并根据类名结尾是否是 PageConfig筛选出所有的配置类。
- 3.调用 getPagesByClass方法，反射获取到配置类的所有页面信息，然后加入到页面集合中，最返回所有module配置页面的信息。

有了 AutoPageConfiguration之后，下面就非常简单啦，我们只需要将mPageConfiguration默认置成 AutoPageConfiguration，这样就可以实现自动化注册啦！

```
/**
 * 初始化页面信息
 *
 * @param context 上下文
 */
private void initPages(Context context) {
    if (mPageConfiguration == null) {
        // 没有设置的话，就使用自动注册配置
        mPageConfiguration = new AutoPageConfiguration();
    }
    registerPageInfos(mPageConfiguration.registerPages(context));
```

```
    CoreConfig.init(context, getPages());
}
```

增加混淆配置

你以为到这儿就结束了？没那么简单！可以发现，上面的实现方案主要是依赖于扫描类并进行反射注的。所以如果代码做了混淆了之后，该方案就会失效了，所以我们还需要在混淆配置清单中增加如下配置来避免混淆：

```
-keep class com.xuexiang.xpage.config.** { *; }
```

我们要保证 com.xuexiang.xpage.config包下的类不能被混淆。

到这儿，自动注册的实现算是真正的讲完了，下面让我们来瞧瞧XPage的新版本还有那些地方更新了！

其他更新

去除LeakCanary依赖

在此之前，XPage一直依赖了 LeakCanary，主要原因还是 LeakCanary在2.0版本之前的使用还是相不方便的，于是我就做了一下默认集成以方便使用。

但是当 LeakCanary升级到2.0以上版本的时候，这个问题似乎就没了。因为进行了重新的设计，Leak

anary的使用变得没那么具有侵入性，因此我就考虑去除了 LeakCanary的依赖。

优化了页面点击的键盘处理

之前在XPageActivity里面做了简单的页面点击处理：当用户点击到非输入框区域就隐藏键盘。

但是这样做了之后发现效果并不是很好，因为有些页面可能并不需要这个功能，如果把这个写到Activiy里面的话，那么在这个Activity下的所有Fragment都将拥有这个功能，这样非常不灵活。

除此之外，使用者可能也想自定义屏幕的触摸事件，因此我对此做了重新设计，将触摸事件的处理下至每个Fragment之中，由Activity调用指定的方法进行处理。

相关链接

- 史上最方便的Android页面框架XPage使用指南
- Navigation和XPage框架相比谁更香
- XPage项目地址：https://github.com/xuexiangjys/XPage

最后

非常感谢大家对XPage 的支持，喜欢的小伙伴可以到项目的Github主页：https://github.com/xuexiangjys/XPage 点击star支持一下哦！

更多资讯内容，欢迎微信搜索公众号:「我的Android开源之旅」