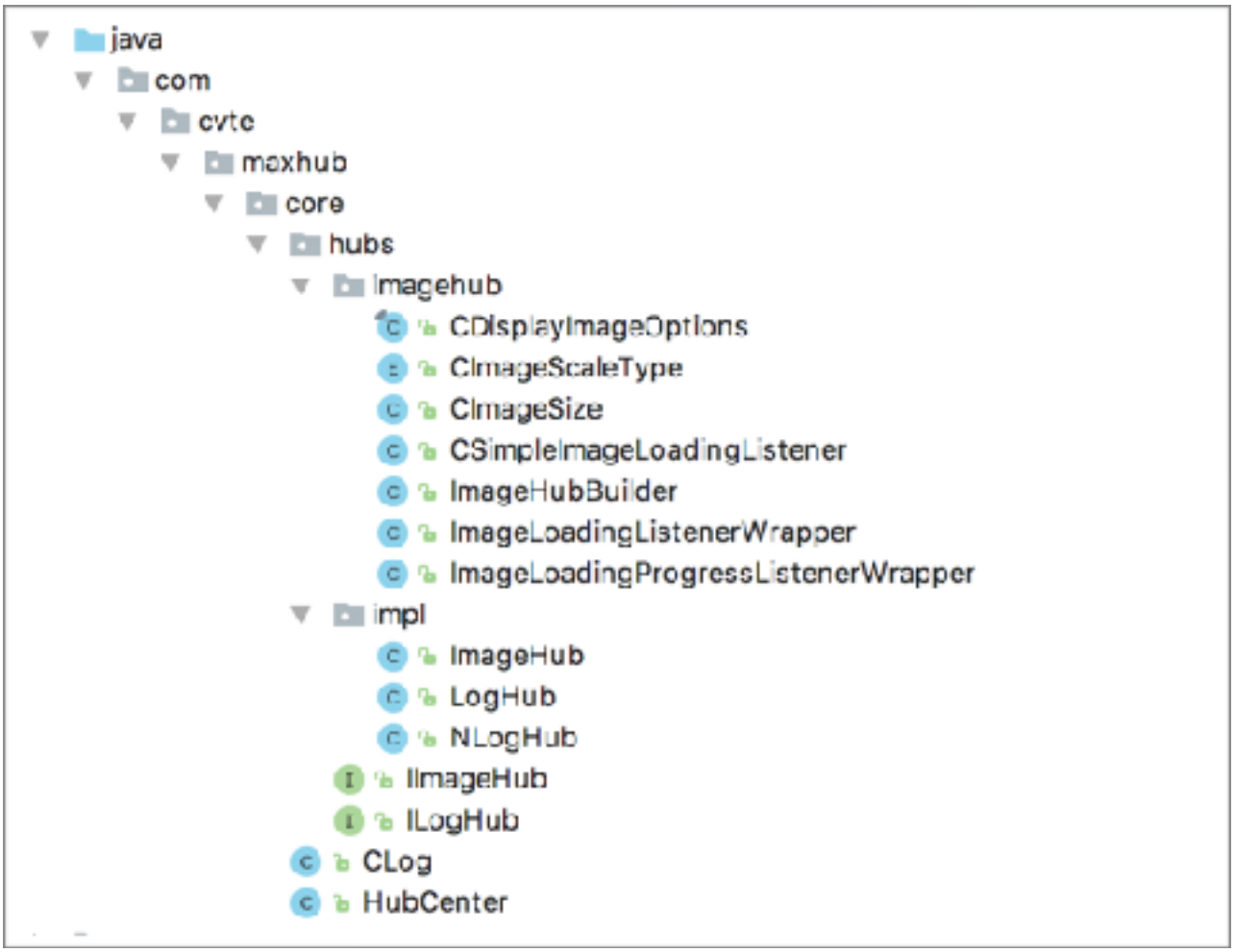
Core层代码设计

1、包结构说明



如上：

core

- 对外核心对象。如：CLog、HubCenter。

core.hubs

- 对外暴露接口。如：ILogHub、IImageHub

core.hubs.impl

- 接口实现类。如：LogHub（使用Log实现）、NLogHub（使用NLog实现）

core.hubs.imagehub / core.hubs.loghub

- hub模块包。如：IImageHub接口对外的数据结构、Listener、Builder等等。

2、初始化

2.1、HubCenter初始化：

```java
    HubCenter.init(Application application);
```

之所以引入HubCenter初始化，是因为Context太常用，部分Hub模块需要依赖Context。如果每个模块都单独传入Context初始化，就限定了使用模块前必须初始化。

2.2、默认配置：

```java
    HubCenter.image();
```

调用时，如果没有实例，会立即构建默认的Hub实例。如果需要提前初始化，可以如此操作。

2.3、自定义配置：

```java
    HubCenter.imageHubBuilder()
            .debug(true)
            .displayImageOptions(CDisplayImageOptions.createSimple());
```

虽然是用时初始化，但需要自定义配置时，需要先构建对应的Builder进行配置，此时构建了Builder，但Hub对象还没初始化。直到真正调用`HubCenter.image()`时，才会使用Builder构建对应的实例。（一旦已经实例化，Builder就不再生效，所以Builder要尽快配置）

3、Hub使用

```java
    HubCenter.image().displayImage(uri);
```

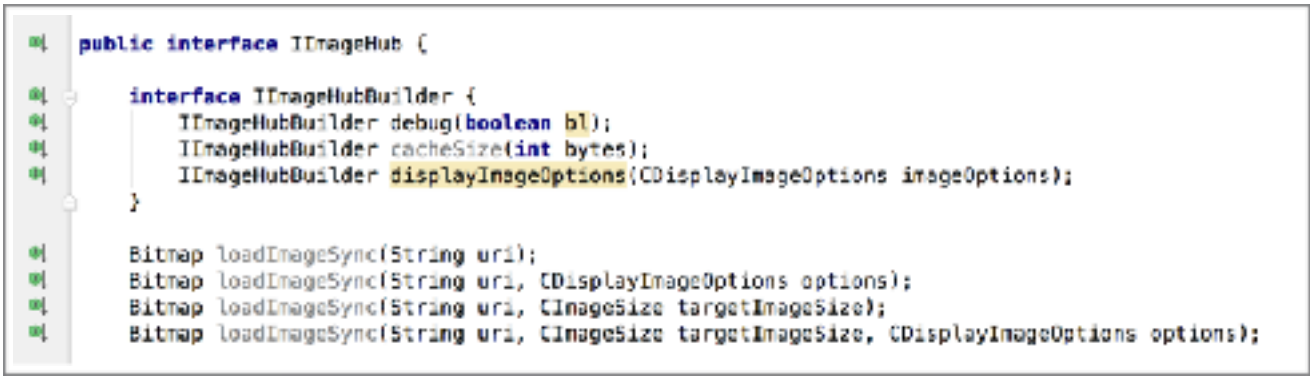
通过HubCenter直接使用即可。

4、CLog的说明

初稿的时候，有同学指出了Log应该为最底层的服务，所以单独抽离成CLog。

5、Builder的说明

通过`HubCenter`，如果看到XXXHubBuilder，说明这个Hub可以被配置。同样，如果Hub需要被配置，则需要提供构建Builder，但不应该提供build()接口，因为仅供内部使用。



```
public interface IImageHub {

    interface IImageHubBuilder {
        IImageHubBuilder debug(boolean bl);
        IImageHubBuilder cacheSize(int bytes);
        IImageHubBuilder displayImageOptions(CDisplayImageOptions imageOptions);
    }

    Bitmap loadImageSync(String uri);
    Bitmap loadImageSync(String uri, CDisplayImageOptions options);
    Bitmap loadImageSync(String uri, CImageSize targetImageSize);
    Bitmap loadImageSync(String uri, CImageSize targetImageSize, CDisplayImageOptions options);
```

6、C对象的说明

由于是对第三方库的一些封装，为了区分，以C开头表示本地的对象，而不是第三方对象。

demo：git@gitlab.gz.cvte.cn:kuangzukai/Whiteboard.git