框架中常用的类命名规则总结

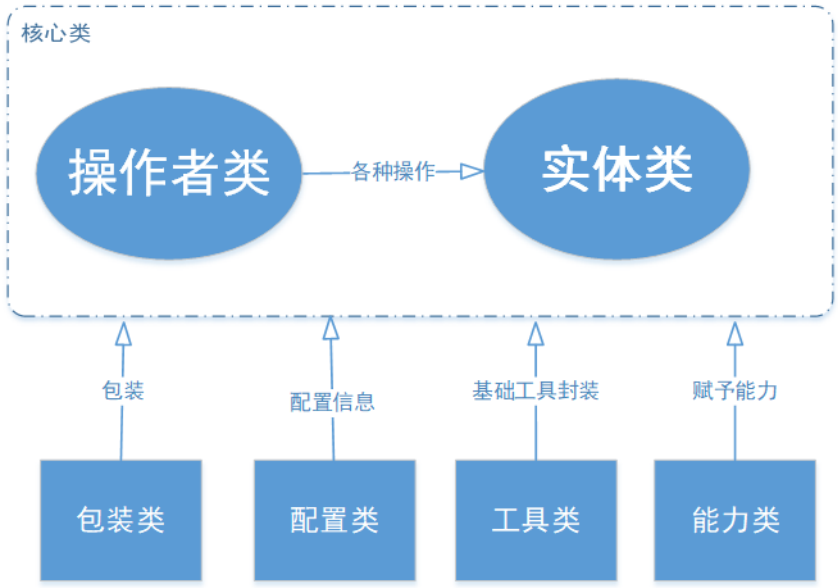
by 汪良文

一. 类或接口的分类

一个Java框架，是一些具有特定功能的类【或者接口】的集合。要对一个框架进行源码分析，不可避免要分析框架中的大量类。当大量的类扑面而来，如果不能快速了解各个类的大概功能，就会陷入类爆炸的漩涡之中。本文是在阅读源码（apache-commons，httpComponent,spring等）过程中，根据类的实际作用对类和接口进行大致的分类。当看到一个类的名字，可以根据其名字大致知道其功能。

1. **实体类**：最核心的概念，代表系统中的组成成分，对应现实世界中的某个实体。
2. **操作者类**：实体类的操作者类，代表系统中的某种操作或操作合集。对应现实世界中的一个动作行为。这是种类最多的类，可以根据操作的类型和目的进一步分类：
    - 监听器Listener：代表某个事件发生后的动作。
    - 解析器Parser：代表对某些文本进行解析的动作。
    - 策略类strategy：代表某些动作的不同的处理策略。
    - 工厂类Factory：代表类的创建动作。
    - 构建类Builder:代表类的构建动作。
    - .....
3. **包装类**：Wrapper或Holder，相当于实体类或操作者类的代理类，在原类的基础上添加一定的功能。
4. **能力类**：表示实体对象或操作类具有的一些能力，比如Closable，Callable,Runnable,Poolable.
5. **配置类**：将实体类和操作类的起配置作用的部分独立出来，形成配置类。
6. **工具类**：某些比较通用的操作可以独立出来，组合成工具类。

各种类型的类或接口的关系如下：



一个框架（或者一个系统）的核心实体及相互的互动构成了框架的基本功能，其他的类大部分都是和核心实体有关联的。可以说，实体类就是一个框架的骨骼，识别出一个框架的核心实体及其关联关系是深入了解一个框架的重中之重。识别出来之后，其他类就有了依附。当我们看到一个类时，就要把它和核心实体类联系起来，研究它和核心实体的关系。这样才能逐渐形成一个整体框架，不会在大量的类中迷失。

常见的一种实体类的命名方式：形容词 + 实体类，形容词表示实体类拥有的能力、特性、实现方法等，如HashMap表示用hash算法实现的map,TreeMap表示用tree实现的map。

常见的一种操作者类的命名方式：形容词 + 实体类 + 操作者，如AbstractBeanDefinitionParser,GenericApplicationListener等。其中，操作者尽量用有明确动作含义的单词，比如Builder,Verifier,Listener,Formatter等，尽量少用一些通用的操作单词，如Handler,Processor,Service等。

二. 框架中常用的一些类命名和后缀的含义

Builder：创建者模式，用来创建对象，使用户只需制定目标对象的规则，不需要直接参与创建过程。一般有Builder.setXXX().setXXX()....build()；

Factory：工厂模式，用来创建对象，使用户只需要对象接口和工厂接口，不需要直接依赖具体的对象。一般有Factory.create()或Factory.getObject()方法；

NameValuePair：键值对

Route & Router：路径，路由

Entry：条目，一般表示键值对。

Parser & Formatter：Parser将一段字符串解析成多个不同含义的元素 String -> Object；Formatter将多个不同含义的元素组装成一段字符串 Object -> String。

Cursor：游标，当进行容器遍历，字符串解析等操作时，需要一个游标来指示当前处理的位置。

Iterator：迭代器，Enumerate的替代品，用来逐个访问容器的元素，一般有hasNext，next两个方法。

Callback：回调，准备好一个回调方法，等到特定事件发生时让对方来触发调用。

Processor&Interceptor：处理器和拦截器，责任链模式中处理器包含有一个拦截器列表，依次进行拦截处理。

Mapper ：匹配者，一般用在服务器中对请求 url 找到一个Handler，一般有lookup()方法。

Resolver ：解析器，和Mapper类似，将一个东西解析成处理这个东西的处理器，比如把url解析到匹配的Controller，把view解析成具体的jsp文件等；

Handler：处理器，一般用在服务器中对请求进行处理。一般一个处理器处理一类请求。a routine for processing requests

Executor：执行器，用来执行某一个任务，一般有execute()方法

Verifier：验证者，用来验证对象是否符合要求，一般有verify()方法。

Generator：生成器，用来生成一个符合要求的对象。

Context：上下文，用来让一系列请求拥有一个共同的环境，能够相互传递消息，一般有getAttribute()和setAttribute方法。

Strategy：策略，指在一个处理过程中，需要采取的一种处理策略。

Utils：工具类。用来处理某一类比较通用的问题。比如TextUtils，EncodingUtils

Assert：断言，表示必须满足某个条件，不满足就需要抛出异常。

Pool：对象的缓存池，用来存储创建耗时费的对象。一般有borrowObject()和returnObject()，或者lease()和release()

Pool & Pooling & Poolable：Pool结尾代表自身是一个池，Pooling 代表自己内部拥有一个池，Poolable代表自己是一个池子里的对象。

Manager是一个管理者，可以管理对象的创建和池化，生命周期，配置项等多方面的内容，是Pool功能，工厂模式，Context上下文等各方面的综合。

Service是一个服务，对用户而言，这是提供某种服务的API。他把内部的很多功能包装成良好的接口提供给用户。

Aware：本意是知道，对...有兴趣的。比如GirlPhoneAware表示想知道GirlPhone的人，一般有setGirlPhone方法，这样等Girl办理好Phone之后，可以调用setGirlPhone，让他实现想知道GirlPhone的愿望。用在框架中，在框架准备好context之前，表达想拥有某资源的引用的愿望，这样框架在资源准备好之后，就可以调用Aware的set方法让他拥有相关的引用；而在框架准备好context之后，用户就知道此Aware就是知道了某个信息的对象了。XXXAware一般有setXXX方法。在Spring框架中使用广泛，Aware 接口 会在Bean初始化的时候注入对应的实例，比如BeanNameAware、ApplicationContextAware、ResourceLoaderAware、ServletContextAware等。如果把框架当做管理机构，对象当做被管理者，一般情况下，是管理机构指挥被管理者，被管理者是无权知道管理机构的信息的。而用了Aware之后，相当于被管理者知道了管理机构的信息，是一种对“好莱坞原则”的破坏。

Cancellable：代表某些可以被取消的过程或操作，一般拥有cancel方法。

Closeable:代表某些可以被关闭的实体，一般拥有close方法。

Scheme：解决方案，如AuthScheme：认证方案。

Wrapper： 对原始对象的包装

Holder：对原始对象的持有