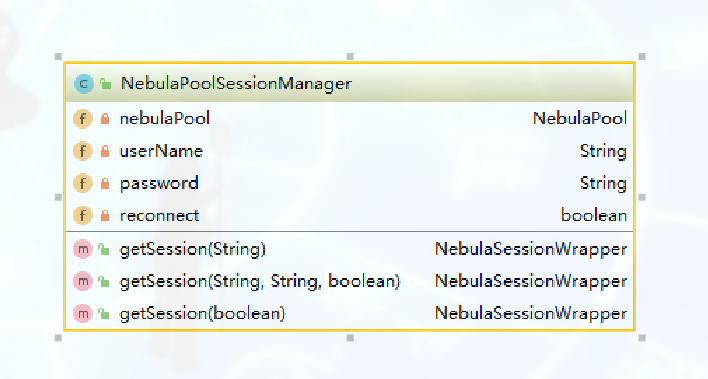
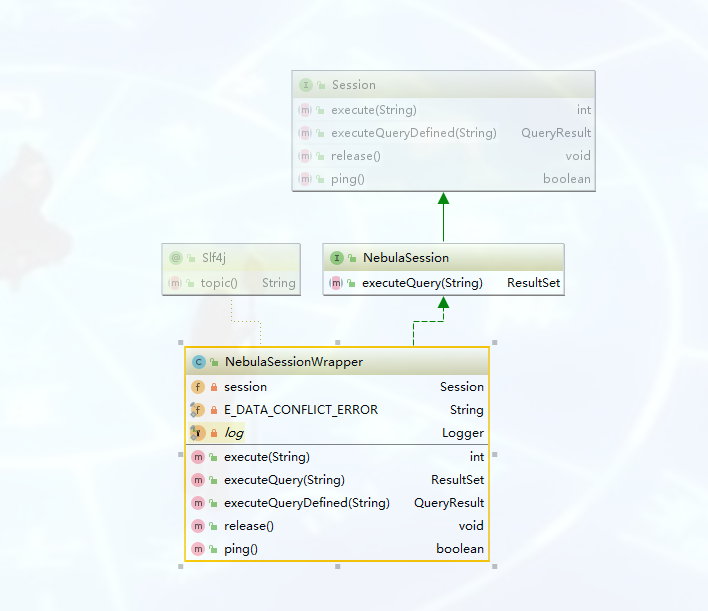
**Nebula-java-ORM设计文档**

# Session管理

由于nebula-java-client中Session是由NebulaPool所产生，且com.vesoft.nebula.client.graph.net.NebulaPool#getSession每次获取Session都需要传入用户名、密码等不变配置，所以想要加强Session则需要同步加强NebulaPool类





加强的连接池管理和session管理有如下优点：

1、获取session时不用每次指定用户名、密码和重连次数，交由加强类管理；

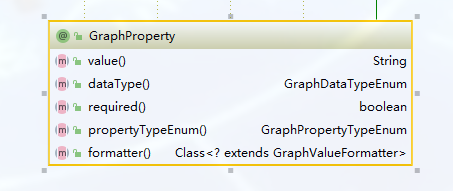
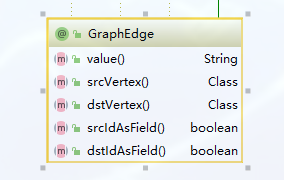
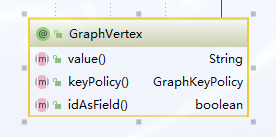
2、微服务时代，每个应用通常只对应了一个图空间（space），加强之后不必每次指定space。

3、加强类区分了查询和更新操作，并且查询封装了自己的返回POJO，为实体类的转换打下基础。

# 注解定义

本套ORM框架是使用注解将实体映射到数据库元素的，有点仿JPA的意思。

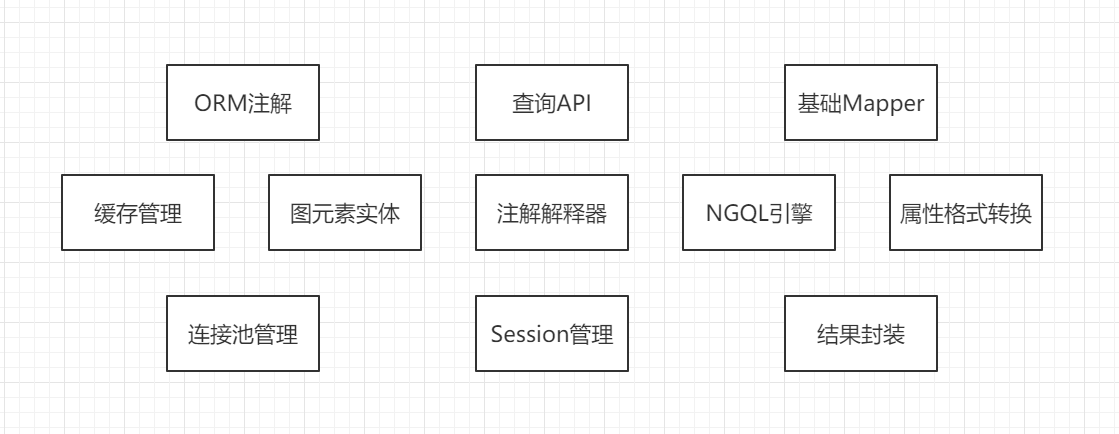
注解包括：com.qihoo.finance.graph.annotation.GraphVertex顶点Tag，com.qihoo.finance.graph.annotation.GraphEdge边类型，com.qihoo.finance.graph.annotation.GraphProperty图属性



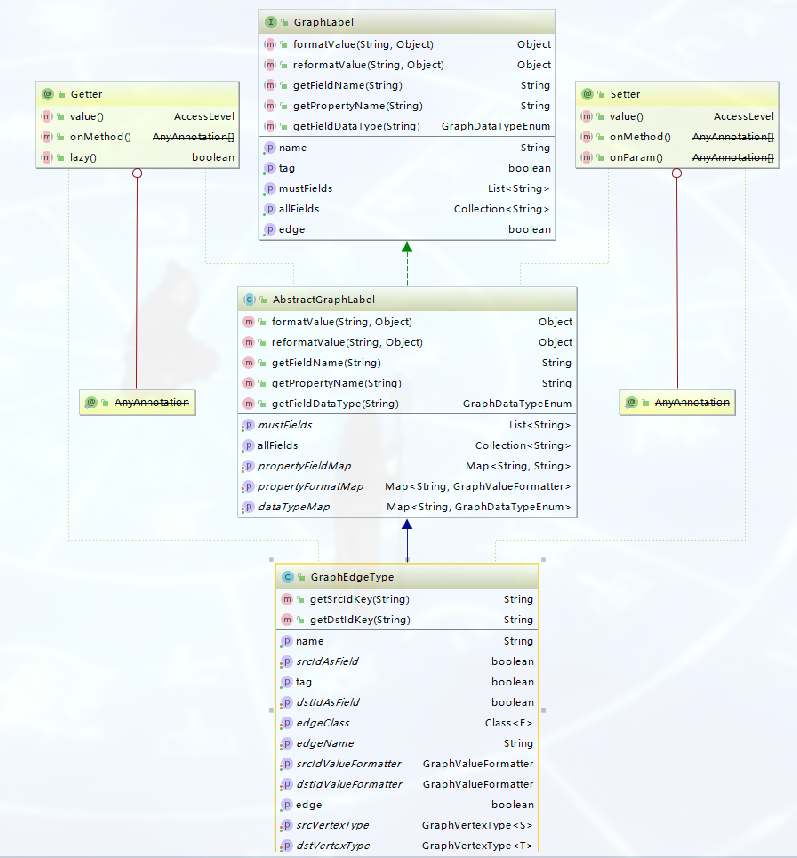
其中包括主键策略（因为没有图空间实体映射，主键策略放在了顶点TAG的注解上），id是否作为属性，属性格式工厂等设置。

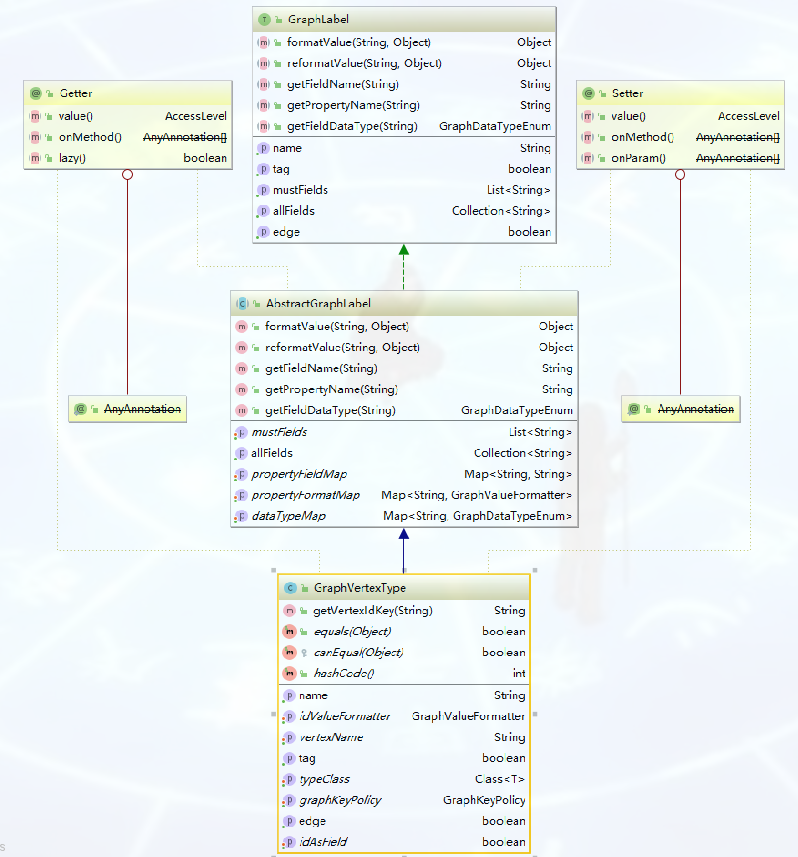
# 整体架构

精炼之后的整体骨架设计图如下：

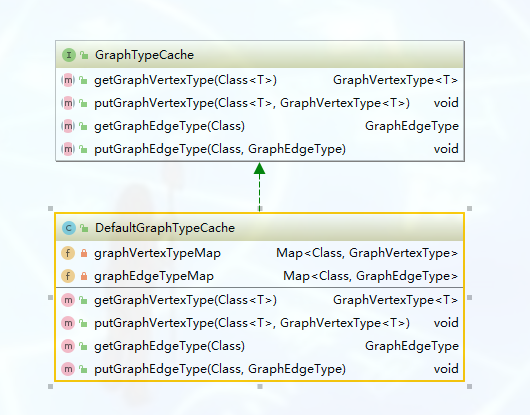


有了上面的注解定义，于是应该有注解解释器，将注解标注的实体映射成图空间中的元素，而元素也是一种抽象概念，它有两种形式：TAG和EDGE，这两种有很多共同的特性。

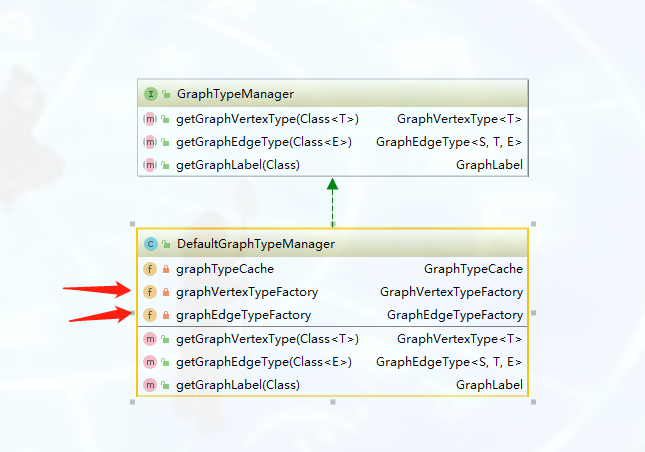


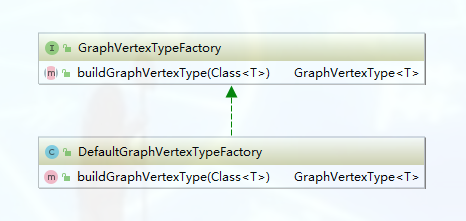


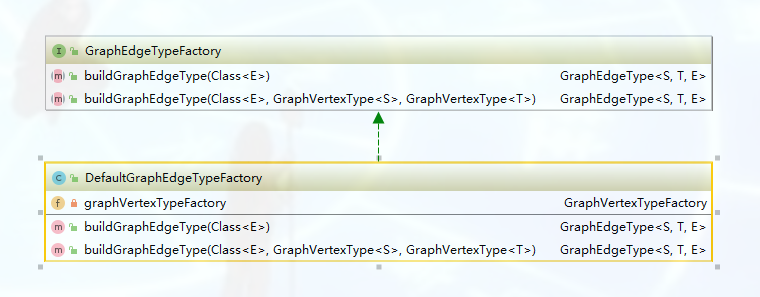
注解映射成抽象元素的过程，一般不常变，所以每次应用第一次解析成抽象实体之后，都应该缓存起来，所以有了缓存管理。

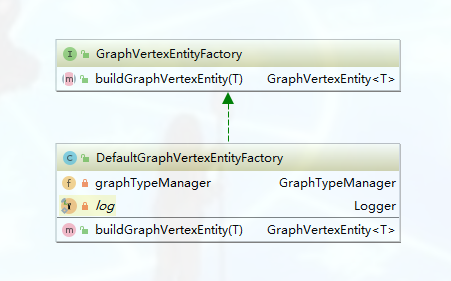


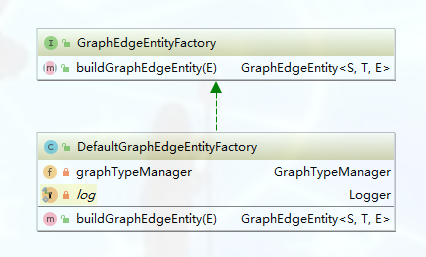
将实体解析映射成抽象图元素以及抽象图元素的实体是一个复杂的解析过程，并且Tag和Edege的解析略有区别，所以区别对待。



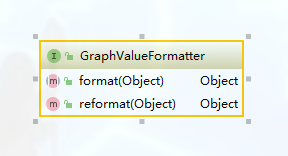






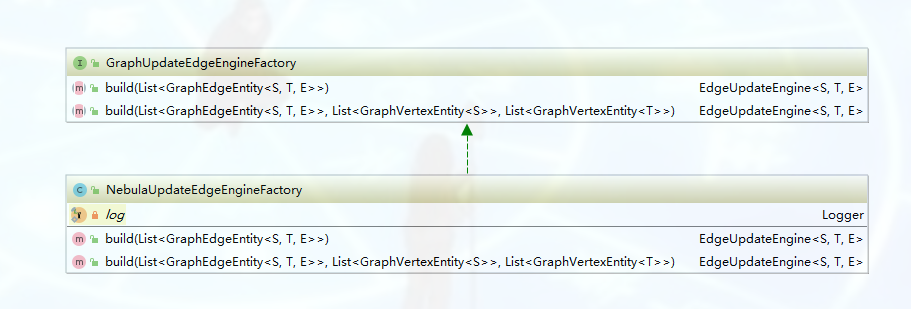


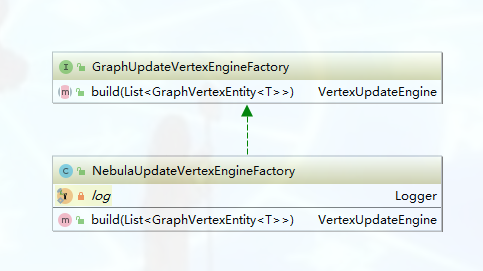
POJO中的属性值和nebula中的字段值不一定是一致的，所以需要向用户提供开放接口。

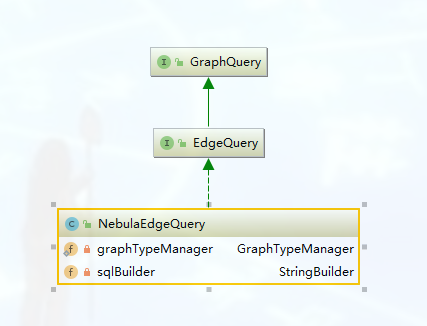


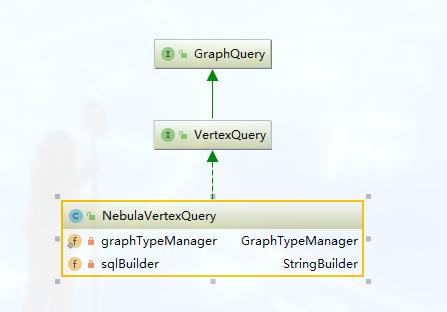
在基础Mapper中封装了写和查询的方法，但写和查的引擎又是分开的。



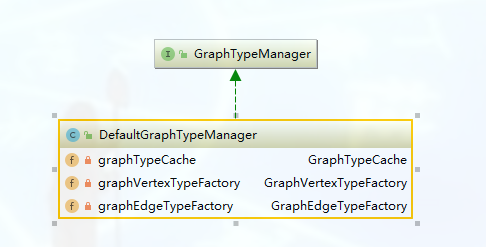




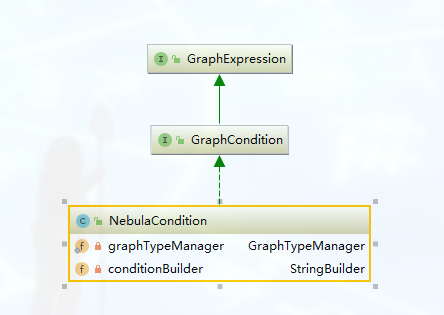


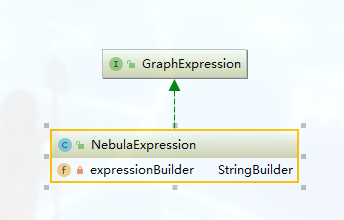


更新和查询都会运用缓存中的抽象元素实体，所以缓存中的元素统一由GraphTypeManager管理



查询中可能会有一些比较复杂的语句，比如条件过滤，case when等，所以引入了GraphExpression和GraphCondition两种概念，而GraphCondition又继承了GraphExpression





结果返回封装类：com.qihoo.finance.graph.domain.impl.QueryResult

上面的设计完成之后，再辅助一些工具类和异常管理，整个ORM框架即呼之欲出。