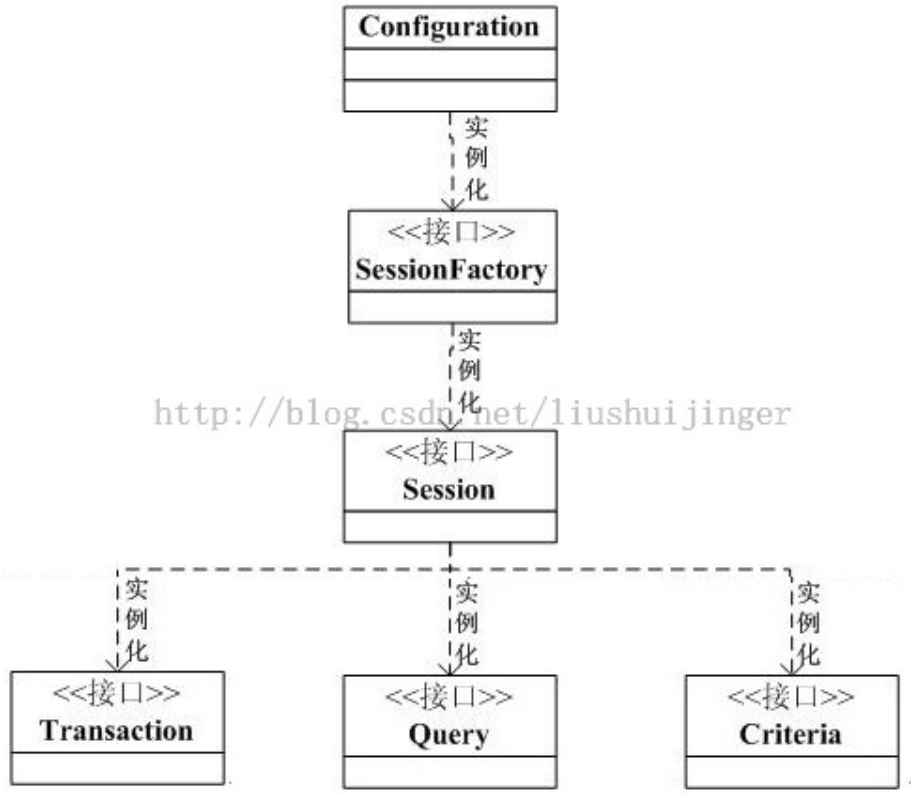
在使用Hibernate的时候，我们通常都会用的Configuration、SessionFactory、Session、Transaction、Query和Criteria等接口。

通过这些接口可以，不仅可以存储与取出持久化对象，还可以对事务进行管理。
下面对着几个接口一一介绍：
几个接口之间的层次关系如下图：



Configuration：
Configuration是Hibernate的入口，负责将配置文件信息加载到内存，并创建一个SessionFactory对象，把读入的配置信息加载到SessionFactory对象的内存里。
特点：
Configuration对象的作用是除了有读取配置文件的功能，还能创建SessionFactory对象。
Configuration对象只存在于系统的初始化阶段，然后所有的持久化操作都能通过这个SessionFactory实例来进行。
Configuration对象只有在Hibernate 进行初始化的时候才需要创建，当使用Configuration对象的实例创建了SessionFactory对象的实例后，其配置信息已经绑定在他返回的SessionFactory对象实例中。因此，一般情况下，得到SessionFactory对象后，Configuration对象的使命就结束了。
用法：
属性文件（hibernate.properties）：Configuration cfg = new Configuration();
Xml文件（hibernate.cfg.xml）：Configuration cfg = new Configuration().configure();

SessionFactory：

SessionFactory负责创建Session实例，每个SessionFactory实例对应一个数据库。SessionFactory是重量级的，占用缓存较大，所以每个数据库只需创建一个SessionFactory实例，当需要操作多个数据库时，再为每一个数据库指定一个SessionFactory实例。

特点：

- 1线程安全，同一个实例可以被应用的多个线程共享
- 2重量级，不能随意创建和销毁他的实例，一个数据库，只需要创建一个SessionFactory的实例。
- 3以后对Configuration对象势力作出的修改都不会影响已经创建好的SessionFactory实例，如果需要使用基于改动后的Configuration实例的SessionFactory，需要从Configuration对象中重新创建新的SessionFactory实例。

用法：

```
Configuration config = new Configuration();
ServiceRegistry sr = new
ServiceRegistryBuilder().applySettings(config.getProperties()).buildServiceRegistry();
SessionFactory sessionFactory = config.buildSessionFactory(sr);
```

Session：

Session是Hibernate持久化操作的基础，负责管理所有与持久化有关的操作，Session与SessionFactory不同，它是轻量级的，也是非线程安全的。创建和销毁不会消耗太多资源，可以为每一个请求分配一个Session。

特点：

- 1不是线程安全的，应该避免多个线程共享同一个Session实例。
- 2Session实例是轻量级的。
- 3Session对象内部有一个缓存，被称为Hibernate第一缓存，他存放被当前工作单元中加载的对象，每个Session实例都有自己的缓存。

用法：

Session session = sessionFactory.openSession();常用方法：
session.save();session.update();session.saveOrUpdate();session.delete();

Transaction：

Transaction负责Hibernate的数据库事务，其实Hibernate本身并不具备事务管理的能力，只是对底层事务接口进行了封装，这样有利于在不同环境或容器中移植，也可以直接访问底层的事务接口。

用法：

Transaction tx = session.beginTransaction();

Query和Criteria：

Query和Criteria负责Hibernate的查询操作。

Query实例封装了一个HQL（Hibernate Query Language）查询语句，HQL与SQL有些类似，只是HQL是面向对象的，它操作的是持久化类的类名和属性名，而SQL操作的是表名和字段名。

Criteria实例完全封装了字符串形式的查询语句，它比Query实例更加面向对象，更适合执行动态查询。

本文只是对这几个接口的一个简单介绍，它们还有很多需要我们去学习跟了解的地方，这几个接口有一个共同的目的，就是让我们用更加面向对象的方式去编程。