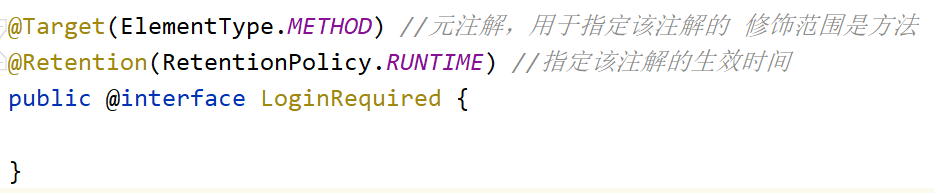
## 注解是怎么实现的？

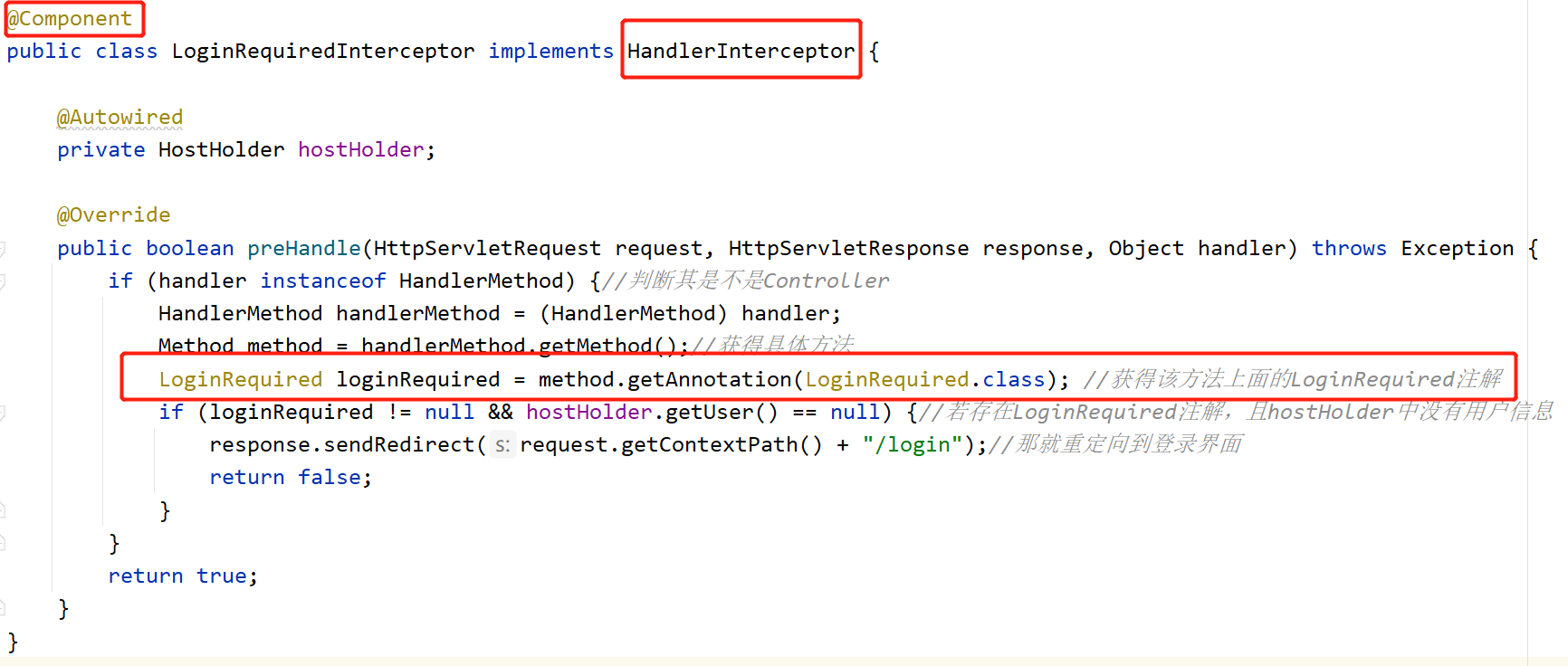
注解其实是代码里面的特殊标记，这些标记可以在编译、类加载、运行时被读取，并执行相应的处理。

注解是一个接口，程序可以通过反射来获取制定程序元素的Annotation对象，以此来取得注解里面的元数据。

自定义注解：



拦截器，拦击带有LoginRequired注解的controller方法。带有此注解意味着需要登录检查。



## ThreadLoca+Redis替代Session，在线程内保存用户信息



对一个请求的处理是在一个线程内实现的。

因此可以使用HostHolder保存用户信息，使用拦截器，在访问到达controller之前就根据请求中的cookie中的sessionid查找redis中对应的用户信息（包含用户名、过期时间等等），保存在Hostholder中，因此不论是在controller还是service处需要使用用户信息的时候，就直接注入HostHolder对象即可。

且该对象内的成员变量是ThreadLocal，因此，对应不同的线程都是不同的用户。

这样就避免了每次需要用户信息就去Redis中查找了。也不需要想Service中单独传入用户信息或者sessionid了。

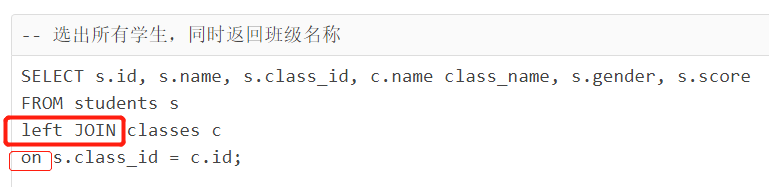
## 连接查询：

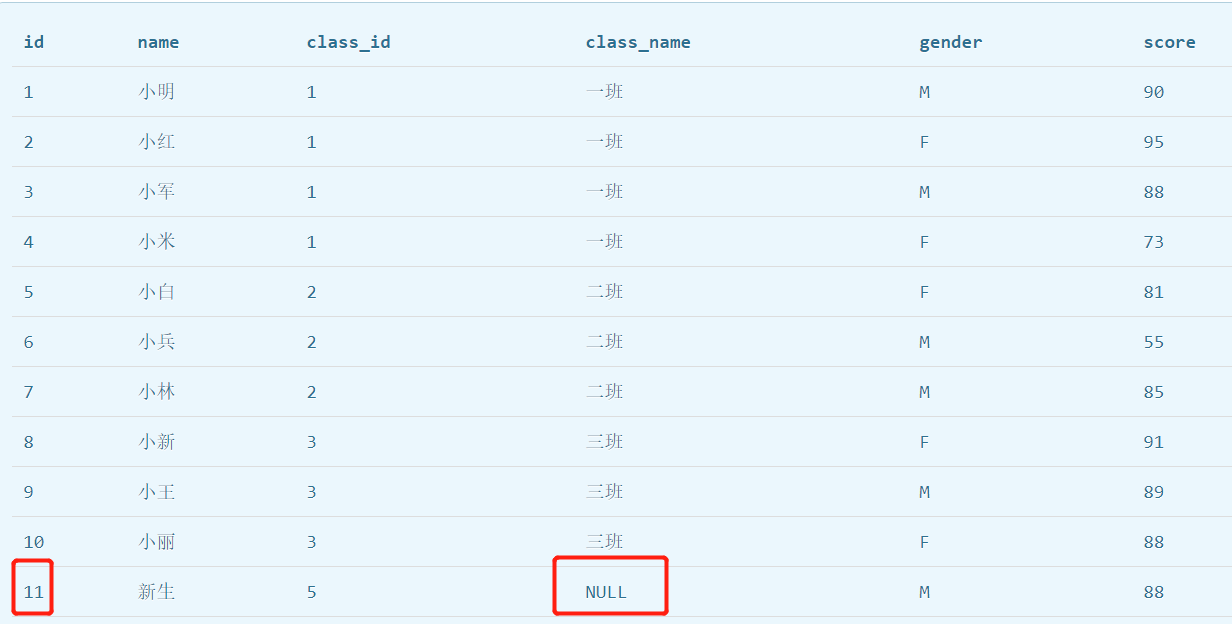
https://www.liaoxuefeng.com/wiki/1177760294764384/1179610888796448

**SELECT** ... **FROM** tableA ??? **JOIN** tableB **ON** tableA.column1 = tableB.column2

注意：区分以下两个查询语句的区别：

1、

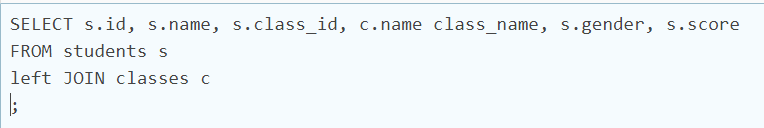




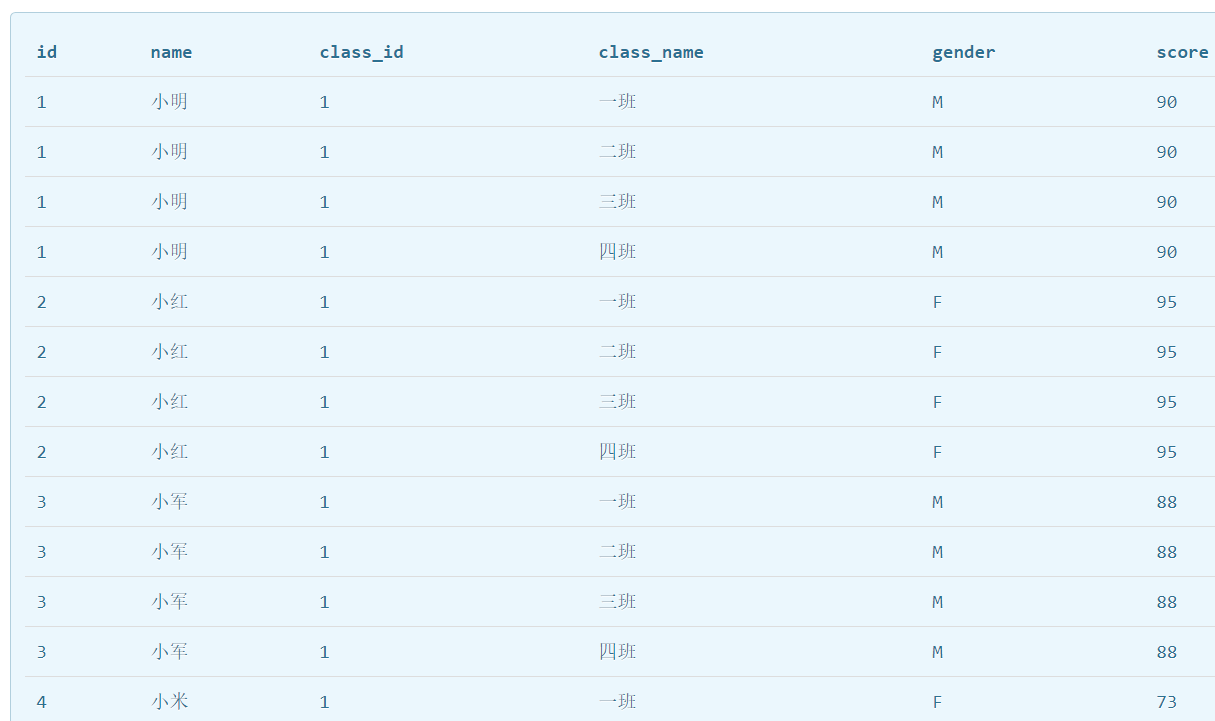
语句意义为：以s.class\_id=c.lcass\_id为条件，将表c左连接至表s当中。

显然，包含左表的全部数据。注意第11条，其class\_name列为空，意味着在表c中，没查到符合s.class\_id=c.lcass\_id条件的。就没有记录可以接到s此行的右侧。

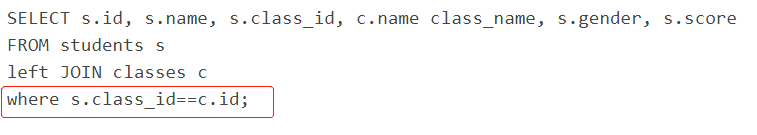
2、



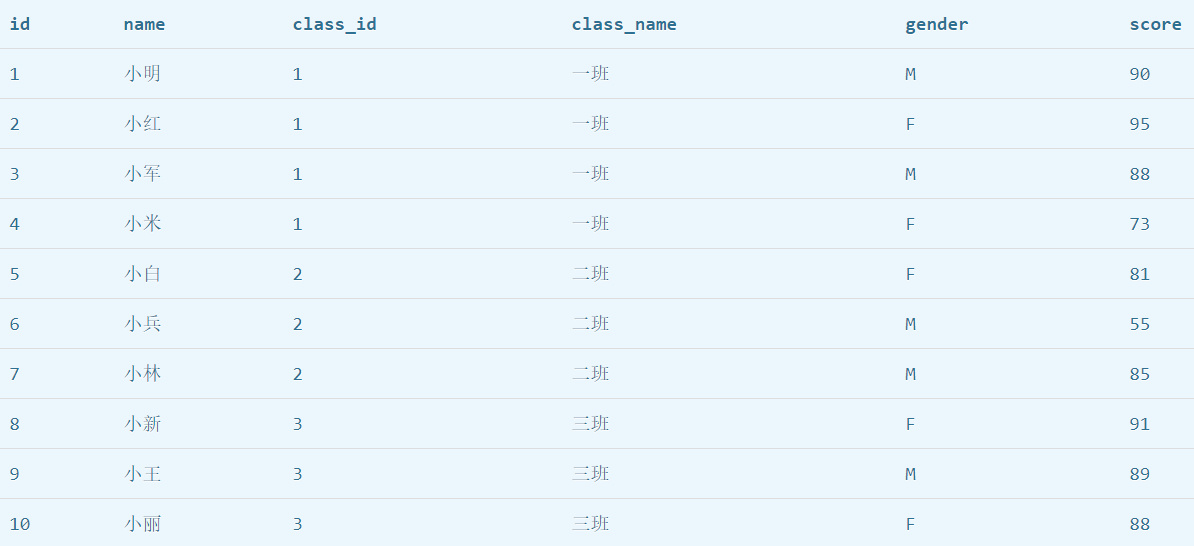
无条件进行左连接，即笛卡尔积。得到结果如下：

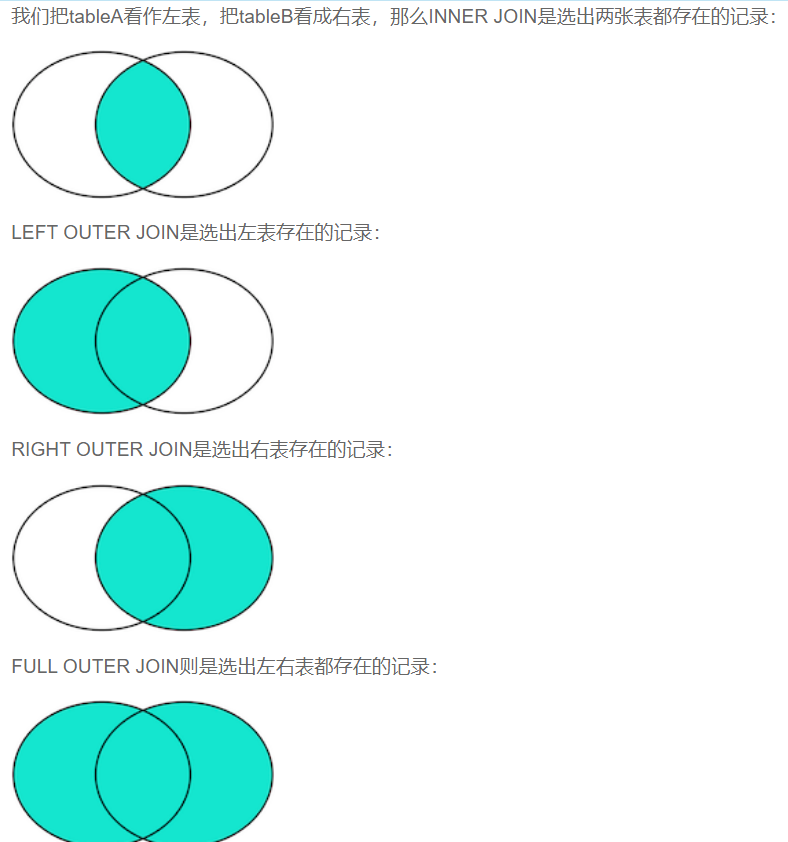


3、



在刚才笛卡尔积的基础上，进行了条件查询。由于在新生那一行的class\_id != c.id，因此此时，不包含第十一行。且**查询效率低于直接on条件**。





## 创建线程的方式

<http://www.cyc2018.xyz/Java/Java%20%E5%B9%B6%E5%8F%91.html#%E5%AE%9E%E7%8E%B0-callable-%E6%8E%A5%E5%8F%A3>

4种方式：

1. 继承Thread
2. 实现Runable
3. 实现Callable
4. 线程池

实现 Runnable 和 Callable 接口的类只能当做一个可以在线程中运行的任务，不是真正意义上的线程，因此最后还需要通过 Thread 来调用。可以理解为任务是通过线程驱动从而执行的。







实现接口 VS 继承 Thread

实现接口会更好一些，因为：

* Java 不支持多重继承，因此继承了 Thread 类就无法继承其它类，但是可以实现多个接口；
* 类可能只要求可执行就行，继承整个 Thread 类开销过大。