通过解析方法名创建查询

通过前面的例子，读者基本上对解析方法名创建查询的方式有了一个大致的了解，这也是 Spring Data JPA 吸引开发者的一个很重要的因素。该功能其实并非 Spring Data JPA 首创，而是源自一个开源的 JPA 框架 Hades，该框架的作者 Oliver Gierke 本身又是 Spring Data JPA 项目的 Leader，所以把 Hades 的优势引入到 Spring Data JPA 也就是顺理成章的了。

框架在进行方法名解析时，会先把方法名多余的前缀截取掉，比如 find、findBy、read、readBy、get、getBy，然后对剩下部分进行解析。并且如果方法的最后一个参数是 Sort 或者 Pageable 类型，也会提取相关的信息，以便按规则进行排序或者分页查询。

在创建查询时，我们通过在方法名中使用属性名称来表达，比如 findByUserAddressZip ()。框架在解析该方法时，首先剔除 findBy，然后对剩下的属性进行解析，详细规则如下（此处假设该方法针对的域对象为 AccountInfo 类型）：

先判断 userAddressZip （根据 POJO 规范，首字母变为小写，下同）是否为 AccountInfo 的一个属性，如果是，则表示根据该属性进行查询；如果没有该属性，继续第二步；

从右往左截取第一个大写字母开头的字符串（此处为 Zip），然后检查剩下的字符串是否为 AccountInfo 的一个属性，如果是，则表示根据该属性进行查询；如果没有该属性，则重复第二步，继续从右往左截取；最后假设 user 为 AccountInfo 的一个属性；

接着处理剩下部分（ AddressZip ），先判断 user 所对应的类型是否有 addressZip 属性，如果有，则表示该方法最终是根据 "AccountInfo.user.addressZip" 的取值进行查询；否则继续按照步骤 2 的规则从右往左截取，最终表示根据 "AccountInfo.user.address.zip" 的值进行查询。

可能会存在一种特殊情况，比如 AccountInfo 包含一个 user 的属性，也有一个 userAddress 属性，此时会存在混淆。读者可以明确在属性之间加上 "\_" 以显式表达意图，比如 "findByUser\_AddressZip()" 或者 "findByUserAddress\_Zip()"。

在查询时，通常需要同时根据多个属性进行查询，且查询的条件也格式各样（大于某个值、在某个范围等等），Spring Data JPA 为此提供了一些表达条件查询的关键字，大致如下：

And --- 等价于 SQL 中的 and 关键字，比如 findByUsernameAndPassword(String user, Striang pwd)；

Or --- 等价于 SQL 中的 or 关键字，比如 findByUsernameOrAddress(String user, String addr)；

Between --- 等价于 SQL 中的 between 关键字，比如 findBySalaryBetween(int max, int min)；

LessThan --- 等价于 SQL 中的 "<"，比如 findBySalaryLessThan(int max)；

GreaterThan --- 等价于 SQL 中的">"，比如 findBySalaryGreaterThan(int min)；

IsNull --- 等价于 SQL 中的 "is null"，比如 findByUsernameIsNull()；

IsNotNull --- 等价于 SQL 中的 "is not null"，比如 findByUsernameIsNotNull()；

NotNull --- 与 IsNotNull 等价；

Like --- 等价于 SQL 中的 "like"，比如 findByUsernameLike(String user)；

NotLike --- 等价于 SQL 中的 "not like"，比如 findByUsernameNotLike(String user)；

OrderBy --- 等价于 SQL 中的 "order by"，比如 findByUsernameOrderBySalaryAsc(String user)；

Not --- 等价于 SQL 中的 "！ ="，比如 findByUsernameNot(String user)；

In --- 等价于 SQL 中的 "in"，比如 findByUsernameIn(Collection<String> userList) ，方法的参数可以是 Collection 类型，也可以是数组或者不定长参数；

NotIn --- 等价于 SQL 中的 "not in"，比如 findByUsernameNotIn(Collection<String> userList) ，方法的参数可以是 Collection 类型，也可以是数组或者不定长参数；