##### 技术专题讨论第二期总结：如何对 JPA 或者 MyBatis 进行技术选型

schedule**2017-12-26 23:25:18.0** face**26214** comment**18** favorite\_border**38**添加至喜欢删除

技术专题讨论历史：

* [《技术专题讨论第一期总结：微服务中网关（API Gateway）的技术选型》](http://www.spring4all.com/article/336" \o "《技术专题讨论第一期总结：微服务中网关（API Gateway）的技术选型》) <http://www.spring4all.com/article/336>
* [《技术专题讨论第二期总结：如何对 JPA 或者 MyBatis 进行技术选型》](http://www.spring4all.com/article/391" \o "《技术专题讨论第二期总结：如何对 JPA 或者 MyBatis 进行技术选型》) <http://www.spring4all.com/article/391>

在我们平时的项目中，大家都知道可以使用 JPA 或者 Mybatis 作为 ORM 层。对 JPA 和 Mybatis 如何进行技术选型？

下面看看大精华总结如下：

## 最佳回答

首先表达个人观点，JPA必然是首选的。

个人认为仅仅讨论两者使用起来有何区别，何者更加方便，不足以真正的比较这两个框架。要评判出更加优秀的方案，我觉得可以从软件设计的角度来评判。个人对 mybatis 并不熟悉，但 JPA 规范和 springdata 的实现，设计理念绝对是超前的。软件开发复杂性的一个解决手段是遵循 DDD（DDD 只是一种手段，但不是唯一手段），而我着重几点来聊聊 JPA 的设计中是如何体现领域驱动设计思想的，抛砖引玉。

### 聚合根和值对象

领域驱动设计中有两个广为大家熟知的概念，entity（实体）和 value object（值对象）。entity 的特点是具有生命周期的，有标识的，而值对象是起到一个修饰的作用，其具有不可变性，无标识。在 JPA中 ，需要为数据库的实体类添加 [@Entity](https://github.com/Entity" \o "@Entity) 注解，相信大家也注意到了，这并不是巧合。

@Entity@Table(name = "t\_order")public class Order {

@Id

private String oid;

@Embedded

private CustomerVo customer;

@OneToMany(cascade = {CascadeType.ALL}, orphanRemoval = true, fetch = FetchType.LAZY, mappedBy = "order")

private List<OrderItem> orderItems;

}

如上述的代码，Order 便是 DDD 中的实体，而 CustomerVo，OrderItem 则是值对象。程序设计者无需关心数据库如何映射这些字段，因为在 DDD 中，需要做的工作是领域建模，而不是数据建模。实体和值对象的意义不在此展开讨论，但通过此可以初见端倪，JPA 的内涵绝不仅仅是一个 ORM 框架。

### 仓储

Repository 模式是领域驱动设计中另一个经典的模式。在早期，我们常常将数据访问层命名为：DAO，而在 SpringData JPA 中，其称之为 Repository（仓储），这也不是巧合，而是设计者有意为之。

熟悉 SpringData JPA 的朋友都知道当一个接口继承 JpaRepository 接口之后便自动具备了 一系列常用的数据操作方法，findAll， findOne ，save等。

public interface OrderRepository extends JpaRepository<Order, String>{

}

那么仓储和DAO到底有什么区别呢？这就要提到一些遗留问题，以及一些软件设计方面的因素。在这次SpringForAll 的议题中我能够预想到有很多会强调 SpringData JPA 具有方便可扩展的 API，像下面这样：

public interface OrderRepository extends JpaRepository<Order, String>{

findByOrderNoAndXxxx(String orderNo,Xxx xx);

@Transactional

@Modifying(clearAutomatically = true)

@Query("update t\_order set order\_status =?1 where id=?2")

int updateOrderStatusById(String orderStatus, String id);

}

但我要强调的是，这是 SpringData JPA 的妥协，其支持这一特性，并不代表其建议使用。因为这并不符合领域驱动设计的理念。注意对比，SpringData JPA 的设计理念是将 Repository 作为数据仓库，而不是一系列数据库脚本的集合，findByOrderNoAndXxxx 方法可以由下面一节要提到的JpaSpecificationExecutor代替，而 updateOrderStatusById 方法则可以由 findOne + save 代替，不要觉得这变得复杂了，试想一下真正的业务场景，修改操作一般不会只涉及一个字段的修改， findOne + save 可以帮助你完成更加复杂业务操作，而不必关心我们该如何编写 SQL 语句，真正做到了面向领域开发，而不是面向数据库 SQL 开发，面向对象的拥趸者也必然会觉得，这更加的 OO。

### Specification

上面提到 SpringData JPA 可以借助 Specification 模式代替复杂的 findByOrderNoAndXxxx 一类 SQL 脚本的查询。试想一下，业务不停在变，你怎么知道将来的查询会不会多一个条件 变成 findByOrderNoAndXxxxAndXxxxAndXxxx.... 。SpringData JPA 为了实现领域驱动设计中的 Specification 模式，提供了一些列的 Specification 接口，其中最常用的便是 ：JpaSpecificationExecutor

public interface OrderRepository extends JpaRepository<Order,String>,JpaSpecificationExecutor<Order>{

}

使用 SpringData JPA 构建复杂查询（join操作，聚集操作等等）都是依赖于 JpaSpecificationExecutor 构建的 Specification 。例子就不介绍，有点长。

请注意，上述代码并不是一个例子，在真正遵循 DDD 设计规范的系统中，OrderRepository 接口中就应该是干干净净的，没有任何代码，只需要继承 JpaRepository （负责基础CRUD）以及 JpaSpecificationExecutor （负责Specification 查询）即可。当然， SpringData JPA 也提供了其他一系列的接口，根据特定业务场景继承即可。

### 乐观锁

为了解决数据并发问题，JPA 中提供了 [@Version](https://github.com/Version" \o "@Version) ，一般在 Entity 中 添加一个 Long version 字段，配合 [@Version](https://github.com/Version" \o "@Version) 注解，SpringData JPA 也考虑到了这一点。这一点侧面体现出，JPA 设计的理念和 SpringData 作为一个工程解决方案的双剑合璧，造就出了一个伟大的设计方案。

### 复杂的多表查询

很多人青睐 Mybatis ，原因是其提供了便利的 SQL 操作，自由度高，封装性好……SpringData JPA对复杂 SQL 的支持不好，没有实体关联的两个表要做 join ，的确要花不少功夫。但 SpringData JPA 并不把这个当做一个问题。为什么？因为现代微服务的架构，各个服务之间的数据库是隔离的，跨越很多张表的 join 操作本就不应该交给单一的业务数据库去完成。解决方案是：使用 elasticSearch做视图查询 或者 mongodb 一类的Nosql 去完成。问题本不是问题。

### 总结

真正走进 JPA，真正走进 SpringData 会发现，我们并不是在解决一个数据库查询问题，并不是在使用一个 ORM 框架，而是真正地在实践领域驱动设计。

（再次补充：DDD 只是一种手段，但不是唯一手段）

## 第二名回答

lexburne 兄说的也很不错了，不过我还想在补充2点，来消除大家对使用spring data jpa 的误解  
spring data jpa 的好处我相信大家都了解，就是开发速度很快，很方便，大家不愿意使用spring data jpa 的地方通常是因为sql 不是自己写的，不可控，复杂查询不好搞，那么下面我要说的就是其实对于这种需求，spring data jpa 是完全支持的！！

第一种方式:[@query](https://github.com/query" \o "@query) 注解指定nativeQuery,这样就可以使用原生sql查询了,示例代码来自官方文档:

public interface UserRepository extends JpaRepository<User, Long> {

[@Query](https://github.com/Query" \o "@Query)(value = "SELECT \* FROM USERS WHERE EMAIL\_ADDRESS = ?1", nativeQuery = true)User findByEmailAddress(String emailAddress);

}

2.如果单靠sql搞不定怎么办？必须要写逻辑怎么办?可以使用官方文档3.6.1 节：Customizing individual repositories 提供的功能去实现，先看官方文档的代码:

interface CustomizedUserRepository {void someCustomMethod(User user);

}class CustomizedUserRepositoryImpl implements CustomizedUserRepository {

public void someCustomMethod(User user) {

*// Your custom implementation*

}

}interface UserRepository extends CrudRepository<User, Long>, CustomizedUserRepository {

*// Declare query methods here*

}

我来解释下上面的代码，如果搞不定的查询方法，可以自定义接口，例如CustomizedUserRepository ，和他的实现了类，然后在这个实现类里用你自己喜欢的dao 框架，比如说mybatis ,jdbcTemplate ,都随意，最后在用UserRepository 去继承CustomizedUserRepository接口，就实现了和其他dao 框架的组合使用！！

那么下面我在总结1下，有了上面介绍的2种功能，你还在担心，使用spring data jpa 会有局限么，他只会加速你的开发速度，并允许你组合使用其他框架，只有好处，没有坏处。。  
最后再说1点，我最近在看es ,然后看了下spring data es 的文档，大概扫了1下，我发现，学会spring data 其中某1个系列以后，在看其他的，我发现我都不用花时间学。。直接就可以用，对就是这么神奇～～

## 第三

工作以来一直是使用 hibernate 和 mybatis，总结下来一般传统公司、个人开发（可能只是我）喜欢用jpa，互联网公司更青睐于 mybatis  
原因：，而mybatis 更加灵活。开发迭代模式决定的，传统公司需求迭代速度慢，项目改动小，hibernate可以帮他们做到一劳永逸；而互联网公司追求快速迭代，需求快速变更，灵活的 mybatis 修改起来更加方便，而且一般每一次的改动不会带来性能上的下降，hibernate经常因为添加关联关系或者开发者不了解优化导致项目越来越糟糕（本来开始也是性能很好的）

1、mybatis官方文档就说了他是一个半自动化的持久层框架，相对于全自动化的 hibernate 他更加的灵活、可控  
2、mybatis 的学习成本低于 hibernate。hibernate 使用需要对他有深入的理解，尤其是缓存方面，作为一个持久层框架，性能依然是第一位的。

hibernate 它有着三级缓存，一级缓存是默认开启的，二级缓存需要手动开启以及配置优化，三级缓存可以整合业界流行的缓存技术 redis，ecache 等等一起去实现  
hibernate 使用中的优化点：  
1、缓存的优化  
2、关联查询的懒加载（在开发中，还是不建议过多使用外键去关联操作）

jpa（Java Persistence API） 与 hibernate 的关系：  
Jpa是一种规范，hibernate 也是遵从他的规范的。  
springDataJpa 是对 repository 的封装,简化了 repository 的操作

## 第四

* 数据分析型的OLAP应用适合用MyBatis，事务处理型OLTP应用适合用JPA。
* 越是复杂的业务，越需要领域建模，建模用JPA实现最方便灵活。但是JPA想用好，门槛比较高，不懂DDD的话，就会沦为增删改查了。
* 复杂的查询应该是通过CQRS模式，通过异步队列建立合适查询的视图，通过视图避免复杂的Join，而不是直接查询领域模型。
* 从目前的趋势来看OLAP交给NoSQL数据库可能更合适

## 第五

使用了一段时间jpa，而mybatis是之前一直在用的，不说区别是啥，因为有很多人比较这两个框架了！  
从国内开源的应用框架来看，国内使用jpa做orm的人还是比较少，如果换成hibernate还会多一些，所以面临的风险可能就是你会用，和你合作的人不一定会用，如果要多方协作，肯定要考虑这个问题！  
灵活性方面，jpa更灵活，包括基本的增删改查、数据关系以及数据库的切换上都比mybatis灵活，但是jpa门槛较高，另外就是更新数据需要先将数据查出来才能进行更新，数据量大的时候，jpa效率会低一些，这时候需要做一些额外的工作去处理！  
现在结合Springboot有Springdata jpa给到，很多东西都简化了，感兴趣并且有能力可以考虑在公司内部和圈子里推广！我的博客里有一些简单使用jpa的示例，<https://github.com/icnws/spring-data-jpa-demo>

## 第六

1.相对来说，jpa的学习成本比mybatis略高  
2.公司业务需求频繁变更导致表结构复杂，此处使用mybatis比jpa更灵活  
3.就方言来讲，一般公司选定数据库后再变更微乎其微，所以此处方言的优势可以忽略