http://blog.csdn.net/u012138706/article/details/47335315

转自：http://t.51jdy.cn/thread-259-1-1.html

持久层 是一个项目 后台 最重要的部分。他直接 决定了 数据读写的性能，业务编写的复杂度，数据结构(对象结构)等问题。

因此 架构师在考虑 使用那个持久层框架的时候 要考虑清楚。  
选择的 标准：  
**1，项目的场景。  
2，团队的技能掌握情况。  
3，开发周期(开发效率)。**  
  
**传统的 业务系统，通常业务都比较复杂，懂业务的运维人员 对sql查询工具都比较熟悉。**  
这种以 数据库 为主的 业务场景 使用 以sql为主的持久层框架 例如：ibatis，mybatis  
这种 性能优化的时候 sql 语句调整较为方便，甚至 业务人员 会直接提供 业务对于的 sql给开发人员。  
  
**对于快速迭代的中小型 新项目，适合使用hibernate(JPA + hibernate驱动)。**  
快速迭代 适合 使用面向对象的 持久层框架。在开发过程中 对象的变更非常频繁。  
这种 持久层框架 适合 快速 重构 ，提高开发效率。  
但是 这种框架 易学难精，用的不好就容易造成性能问题，用的好性能并不输于ibatis框架。  
在做性能优化方面 可借助 缓存实现。  
  
**以上说说到了 项目场景 和 开发效率。**  
再说说 团队技能掌握情况，这个情况 有的时候 可以忽略，但有时缺不能忽视。  
这个情况也和上面说的两个情况 对比着 分析，需要做一个平衡。  
分以下几个情况：  
**老团队新项目**：这个要考虑 是重用公司已有框架资源，持续积累改进，还是放弃旧的，开发新的。  
                        这个时候，大部分开发人员会怂恿技术主管或架构师 使用新的框架，尝试新的技术；不关心新技术带来的风险，项目的进度，已用积累的放弃。  
                        作为架构师/技术主管，需要考虑项目的场景，周期，核心点，公司技术的可持续积累，风险性，团队技术的培训提升 等方面综合考虑。  
                        结论：结合项目场景适度引进新技术(这也是提升团队技术的一个机会，但风险与机会并存，进度紧张的时候不建议更换）。  
**新项目新团队**：这个要区分地方，大城市好招人，以项目场景为主，选择框架，招聘精通此类框架的人才即可。  
                        小城市 招聘人才困难，如果 没有 技术牛人引导使用，建议 还是要以团队技术人员掌握的技术为主。  
                        一群人 用一个不熟悉的框架，不如 大部分用一个熟悉的框架。(技术培训 是需要一个周期的，如果没有牛人不建议现学现用)。  
                        这里 可能有另外一个声音：大家一起学，边学边用。在项目非紧急的情况下 可以考虑。  
                        但本人不太建议拿项目当试验；稳定，高效，靠谱，好扩展，这是作为一个架构师 要肩负的责任。  
**以技术为核心的项目**：要招聘一些技术牛人，使用符合项目场景的框架，而不是复合团队的框架，但团队技术问题 要通过培训，牛人带路等方式解决。  
  
**以上是抽象的说法，以下是我 选取的框架对比资料,选来选取 其实就围绕在两个方向：  
是面向对象 还是面向数据结构；  
是hibernate 还是ibatis；**  
  
**何时使用iBATIS**  
  
iBATIS最好是用在你需要全面地控制SQL的时候，在需要对SQL查询做微调的时候也很有用。  
当你在应用和数据库设计两方面都有完全的控制权的时候，就不应该使用iBATIS，  
因为在这样的情况下，应用可能会做出修改以适应数据库，或是反过来。  
在这种情形中，你可以构建一个完全的对象-关系应用，其他的ORM工具更适于使用，  
因为iBATIS较为以SQL为中心，其通常被称作反转的——功能齐全的ORM工具生成SQL，  
而iBATIS直接使用SQL。iBATIS也不适合于非关系型的数据库，  
因为这类数据库不支持事务和其他iBATIS用到的键特性。  
  
简单是iBATIS最大的优势，因为它提供一个简单的映射和用于构建数据访问代码的API层。  
在这一框架中，数据模型和对象模型不需要做精确的彼此映射。  
这是因为iBATIS使用了一个数据映射器（data mapper），  
其经由一个XML描述符而不是元数据映射器把对象映射到存储过程、SQL语句或是ResultSet上，  
而元数据映射器起的是把领域中的对象映射到数据库中的表上的作用。  
因此，iBATIS能够使得数据模型和对象模型彼此独立，互不相干。  
  
**为什么我们需要Hibernate?**  
  
传统上用于对象-关系映射的实体bean（entity bean）非常难以理解和维护，Hibernate使得对象-关系映射变得简单起来，它的方法是在一个XML文件中映射元数据，该文件定义了需要映射到某个特定类上的数据库中的表。  
在其他的持久性框架中，你需要修改应用类来实现对象-关系映射；而在Hibernate中则不需要这样做。  
使用了Hibernate后，你就无需担心数据库的改变，因为手工修改SQL脚本文件的工作已被免除。  
如果你需要不时改变应用使用到的数据库的话，也可以通过修改配置文件中的dialet属性来很容易地解决这一问题。  
Hibernate提供了全部的SQL功能，其中的有些是早先的商业ORM框架一直没有提供的。  
Hibernate也支持许多的数据库，其中包括MySQL、Oracle、Sybase、Derby和PostgreSQL等，而且也能够与基于简单Java对象（plain old Java object，POJO）的模型配合得很好。  
Hibernate基于所选择的底层数据库来生产JDBC代码，因此省去了编写JDBC代码的麻烦，它还支持连接的池化。Hibernate使用的API很简单也很容易学习，只有很少SQL知识的开发者也能够使用Hibernate，因为它减轻了编写SQL查询的负担。  
Hibernate架构  
  
就内部来说，Hibernate用到了JDBC，JDBC提供了数据库的一个抽象层，它同时也采用了Java Transaction API（JTA）和JNDI来集成其他应用。Hibernate需要用来与数据库交互的连接信息由JDBC连接池提供，这需要做配置。  
Hibernate的架构主要由两个接口——Session和Transaction组成——以及一个Query接口，该接口位于应用的持久层中。定义于应用的业务层中的类通过Hibernate持久层的独立元数据来进行交互，持久层转而使用某些JDBC API来与数据库层对话。此外，Hibernate还用到了其他的配置接口，其中主要是有着适当命名的Configuration类。Hibernate还使用回调接口和一些用于扩展映射功能的可选接口。  
  
  
Hibernate组成部分的主要编程接口：  
Ÿ           org.hibernate.SessionFactory基本上是用来获取一个session实例，并且可看作是连接池化机制的一个模拟。这是线程安全的，因为所有的应用线程都使用单一的SessionFactory（只要Hibernate只使用一个数据库）。该接口通过配置文件来配置，配置文件决定了要加载的映射文件。  
Ÿ           org.hibernate.Session提供了一个单独的线程，该线程确定应用和数据库之间的对话。这是对一个特定（单个）连接的模拟。该接口是非常轻量级的，而且是非线程安全的。  
Ÿ           org.hibernate.Transaction提供了一个单线程对象，其横跨整个应用并确定原子工作单元。其基本上抽象了JDBC、JTA和CORBA事务。  
Ÿ           org.hibernate.Query被用来执行查询，或以HQL的形式或是以底层数据库的SQL方言形式。Query实例是轻量级的，需要提到的很重要的一点是，它不能用在创建它的session的外部。

[51筋斗云](http://t.51jdy.cn/thread-259-1-1.html)（微信公众号：cn51jdy ），欢迎分享本文！