#### Hibernate和mybatis的区别

**hibernate:**是一个标准的ORM框架（对象关系映射）。入门门槛较高，不需要写sql，sql语句自动生成，对sql语句进行优化、修改比较困难。

**mybatis：**专注sql本身，需要程序员自己编写sql语句，sql修改、优化比较方便。mybatis是一个不完全的ORM框架，虽然程序员自己写sql，mybatis也可以实现映射（输入映射，输出映射）

## 1.1 Hibernate 简介

Hibernate对数据库结构提供了较为完整的封装，Hibernate的O/R Mapping实现了POJO 和数据库表之间的映射，以及SQL 的自动生成和执行。程序员往往只需定义好了POJO 到数据库表的映射关系，即可通过Hibernate 提供的方法完成持久层操作。程序员甚至不需要对SQL 的熟练掌握， Hibernate/OJB 会根据制定的存储逻辑，自动生成对应的SQL 并调用JDBC 接口加以执行。

## 1.2 MyBatis简介

iBATIS 的着力点，则在于POJO 与SQL之间的映射关系。然后通过映射配置文件，将SQL所需的参数，以及返回的结果字段映射到指定POJO。 相对Hibernate“O/R”而言，iBATIS 是一种“Sql Mapping”的ORM实现。

## Hibernate的调优方案

1. 制定合理的缓存策略；
2. 尽量使用延迟加载特性；
3. 采用合理的Session管理机制；
4. 使用批量抓取，设定合理的批处理参数（batch\_size）;
5. 进行合理的O/R映射设计

## Mybatis调优方案

MyBatis在Session方面和Hibernate的Session生命周期是一致的，同样需要合理的Session管理机制。MyBatis同样具有二级缓存机制。 MyBatis可以进行详细的SQL优化设计。

## SQL优化方面

Hibernate的查询会将表中的所有字段查询出来，这一点会有性能消耗。Hibernate也可以自己写SQL来指定需要查询的字段，但这样就破坏了Hibernate开发的简洁性。而Mybatis的SQL是手动编写的，所以可以按需求指定查询的字段。

Hibernate HQL语句的调优需要将SQL打印出来，而Hibernate的SQL被很多人嫌弃因为太丑了。MyBatis的SQL是自己手动写的所以调整方便。但Hibernate具有自己的日志统计。Mybatis本身不带日志统计，使用Log4j进行日志记录。

#### hibernate的缓存机制

Hibernate缓存包括两大类：Hibernate一级缓存和Hibernate二级缓存Hibernate一级缓存又称为“Session的缓存”，它是内置的，不能被卸载（不能被卸载的意思就是这种缓存不具有可选性，必须有的功能，不可以取消session缓存）。由于Session对象的生命周期通常对应一个数据库事务或者一个应用事务，因此它的缓存是事务范围的缓存。第一级缓存是必需的，不允许而且事实上也无法卸除。在第一级缓存中，持久化类的每个实例都具有唯一的OID。 Hibernate二级缓存又称为“SessionFactory的缓存”，由于SessionFactory对象的生命周期和应用程序的整个过程对应，因此Hibernate二级缓存是进程范围或者集群范围的缓存，有可能出现并发问题，因此需要采用适当的并发访问策略，该策略为被缓存的数据提供了事务隔离级别。第二级缓存是可选的，是一个可配置的插件，在默认情况下，SessionFactory不会启用这个插件。

#### cookie和session的区别

　　1 session保存在服务器，客户端不知道其中的信息；cookie保存在客户端，服务端可以知道其中的信息

　　2 session中保存的是对象，cookie中保存的是字符串

　　3 session不能区分路径，同一个用户在访问一个网站期间，所有的session在任何一个地方都可以访问到；而cookie中如果设置了路径参数，那么同一个网站中不同路径下的cookie互相是访问不道德

### session的原理：

　　http是无状态的协议，客户每次读取web页面时，服务器都打开新的会话，而且服务器也不会自动维护客户的上下文信息。session就是一种保存上下文信息的机制，她是针对每一个用户的，session的内容在服务器端，通过sessionId来区分不同的客户，session是以cookie或url重写为基础的，默认用cookie来实现，系统会创造一个JSESSIONID的输出cookie，我们成为session cookie，以区分persistent coookies，注意session cookie是存储于浏览器内存中的，并不是写到硬盘上的；我们通常是看不见JSESSIONID的，但是当我们禁用浏览器的cookie后，web服务器会采用url重写的方式传递sessionid，我们就可以在浏览器看到sessionid=HJHADKSFHKAJSHFJ之类的字符串；session cookie针对某一次会话而言，会话结束session cookie也就消失了

### session与cookie的联系：

　　session是需要借助cookie才能正常工作的，如果客户端完全禁止cookie，session将失效

### session和cookie的应用：

如何利用实现自动登录  
　　当用户在某个网站注册后，就会收到一个惟一用户ID的cookie。客户后来重新连接时，这个用户ID会自动返回，服务器对它进行检查，确定它是否为注册用户且选择了自动登录，从而使用户无需给出明确的用户名和密码，就可以访问服务器上的资源

会话跟踪：

通常session cookie是不能跨窗口使用的，当你新开了一个浏览器窗口进入相同页面时，系统会赋予你一个新的sessionid，这样我们信息共享的目的就达不到了，此时我们可以先把sessionid保存在persistent cookie中，然后在新窗口中读出来，就可以得到上一个窗口SessionID了，这样通过session cookie和persistent cookie的结合我们就实现了跨窗口的session tracking

**1、内联接**（典型的联接运算，使用像 =  或 <> 之类的比较运算符）。包括相等联接和自然联接。       
内联接使用比较运算符根据每个表共有的列的值匹配两个表中的行。例如，检索 students和courses表中学生标识号相同的所有行。     
**2、外联接。**外联接可以是左向外联接、右向外联接或完整外部联接。       
在 FROM子句中指定外联接时，可以由下列几组关键字中的一组指定：       
1）LEFT  JOIN或LEFT OUTER JOIN       
左向外联接的结果集包括  LEFT OUTER子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值。         
2）RIGHT  JOIN 或 RIGHT  OUTER  JOIN       
右向外联接是左向外联接的反向联接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。         
3）FULL  JOIN 或 FULL OUTER JOIN  
完整外部联接返回左表和右表中的所有行。当某行在另一个表中没有匹配行时，则另一个表的选择列表列包含空值。如果表之间有匹配行，则整个结果集行包含基表的数据值。     
  
**3、交叉联接**交叉联接返回左表中的所有行，左表中的每一行与右表中的所有行组合。交叉联接也称作笛卡尔积。      
FROM 子句中的表或视图可通过内联接或完整外部联接按任意顺序指定；但是，用左或右向外联接指定表或视图时，表或视图的顺序很重要。有关使用左或右向外联接排列表的更多信息，请参见使用外联接。

#### Spring的AOP面向切面编程

1、AOP概念：

面向切面编程，指扩展功能不修改源代码，将功能代码从业务逻辑代码中分离出来。

       主要功能：日志记录，性能统计，安全控制，事务处理，异常处理等等。

       主要意图：将日志记录，性能统计，安全控制，事务处理，异常处理等代码从业务逻辑代码中划分出来，通过对这些行为的分离，我们希望可以将它们独立到非指导业务逻辑的方法中，进而改变这些行为的时候不影响业务逻辑的代码。

2、AOP特点：

采用横向抽取机制，取代了传统纵向继承体系重复性代码。

3、AOP底层实现：

AOP底层使用动态代理实现。包括两种方式：

      （1）使用JDK动态代理实现。

      （2）使用cglib来实现