1. mybatis的一级、二级缓存
2. 一级缓存：基于PerpetualCache的HashMap本地缓存（key为hashcode+statementId+sql语句，Value为查询出来的结果集映射成的java对象），其作用域为session，当session flush或者close后，该session中的所有cache就会清空，默认打开一级缓存
3. 二级缓存：机制与一级缓存相同，不同在于其作用域为Mapper（Namespace），并且可自定义存储源，默认不打开
4. 对于缓存数据更新机制，当某一个作用域进行了C/U/D操作后，默认该作用域下所有select中的缓存将被clear；若开启了二级缓存，则只根据配置判断是否clear
5. Mybatis都有哪些Executor执行器，它们之间的区别是什么？

Executor有两个实现类：BaseExecutor和CachingExecutor；

BaseExecutor是个抽象类，是Exectutor的默认实现，它有三个子类，分别是SimpleExecutor、ReuseExecutor 、BatchExecutor

SimpleExecutor：没执行一次update或select，就开启一个statement对象，用完立刻关闭statement对象

ReuseExecutor：执行update或select，以sql作为key查找statement对象，存在就使用，不存在就创建；用完后，不关闭statement对象，而是放置于Map<String,Statement>内，供下次使用，可重复使用

BatchExecutor：执行update（没有select，JDBC批处理不支持select），将所有的sql都添加到批处理中（addBatch（）），等待统一执行（executeBatch（）），它缓存了多个statement对象，每个statement对象都是addBatch（）完毕后，等待逐一执行executeBatch（）批处理

CacheExecutor：缓存执行器，先从缓存中查询结果，如果存在就返回之前的结果，如果不存在，再委托给Executor delegate去数据库中取，delegate可以是上面任何一个执行器

作用范围：Executor的这些特点，都严格限制在sqlsession生命周期范围内

1. Executor执行的具体过程

（1）当有一个查询请求访问的时候，首先会经过 Executor 的实现类 CachingExecutor ，先从缓存中查询 SQL 是否是第一次执行，如果是第一次执行的话，那么就直接执行 SQL 语句，并创建缓存，如果第二次访问相同的 SQL 语句的话，那么就会直接从缓存中提取。

（2）如果没有的话，就再重新创建 Executor 执行器执行 SQL 语句， 创建我们上面提到的三种执行器

（3）到这里，执行器所做的工作就完事了，Executor 会把后续的工作交给 StatementHandler 继续执行。

1. mybatis是否支持延迟加载？实现原理是什么？

Mybatis仅支持association关联对象和collection关联对象集合对象得延迟加载，association就是一对一，collection就是一对多查询；可在配置文件中配置lazyLoadingEnable=true|false

实现原理：使用CGLIB创建目标对象的代理对象，当调用目标方法时，进入拦截器方法；比如调用a.getB().getName()，拦截器invoke()方法发现a.getB()是null值，那么就会单独发送事先保存好的查询关联B对象的sql，把B查询上来，然后调用a.setB(b),于是a的对象b属性就有值了，接着完成a.getB().getName()方法的调用，这就是延迟加载的基本原理。不光是mybatis，几乎所有的包括hibernate，支持延迟加载的原理都是一样的。

1. mybatis的插件运行原理以及如何编写一个插件？

Mybatis 仅可以编写针对 ParameterHandler、ResultSetHandler、 StatementHandler、Executor 这 4 种接口的插件，Mybatis 使用 JDK 的动态代理，为需要拦截的接口生成代理对象以实现接口方法拦截功能，每当执行这 4 种接口对象的方法时，就会进入拦截方法，具体就是 InvocationHandler 的 invoke()方法，当然，只会拦截那些你指定需要拦截的方法。  
编写插件：实现 Mybatis 的 Interceptor 接口并复写 intercept()方法，然后在给插件编写注解，指定要拦截哪一个接口的哪些方法即可，记住，别忘了在配置文件中配置你编写的插件。