Java 最常见的 208 道面试题：第十三模块答案

Java团长 4月6日

十三、Mybatis

125. mybatis 中 #{}和 ${}的区别是什么？

- #{}是预编译处理，${}是字符串替换；
- Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；
- Mybatis在处理${}时，就是把${}替换成变量的值；
- 使用#{}可以有效的防止SQL注入，提高系统安全性。

126. mybatis 有几种分页方式？

a. 数组分页
b. sql分页
c. 拦截器分页
d. RowBounds分页

128. mybatis 逻辑分页和物理分页的区别是什么？

- 物理分页速度上并不一定快于逻辑分页，逻辑分页速度上也并不一定快于物理分页。
- 物理分页总是优于逻辑分页：没有必要将属于数据库端的压力加诸到应用端来，就算速度上存在优势,然而其它性能上的优点足以弥补这个缺点。

129. mybatis 是否支持延迟加载？延迟加载的原理是什么？

Mybatis仅支持association关联对象和collection关联集合对象的延迟加载，association指的就是一对一，collection指的就是一对多查询。在Mybatis配置文件中，可以配置是否启用延迟加载lazyLoadingEnabled=true|false。

它的原理是，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。

当然了，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。

130. 说一下 mybatis 的一级缓存和二级缓存？

一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当 Session flush 或 close 之后，该 Session 中的所有 Cache 就将清空，默认打开一级缓存。

二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap 存储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如 Ehcache。默认不打开二级缓存，要开启二级缓存，使用二级缓存属性类需要实现 Serializable序列化接口(可用来保存对象的状态),可在它的映射文件中配置 <cache/> ；

对于缓存数据更新机制，当某一个作用域(一级缓存 Session/二级缓存 Namespaces)的进行了C/U/D 操作后，默认该作用域下所有 select 中的缓存将被 clear。

131. mybatis 和 hibernate 的区别有哪些？

（1）Mybatis和hibernate不同，它不完全是一个ORM框架，因为MyBatis需要程序员自己编写Sql语句。

（2）Mybatis直接编写原生态sql，可以严格控制sql执行性能，灵活度高，非常适合对关系数据模型要求不高的软件开发，因为这类软件需求变化频繁，一但需求变化要求迅速输出成果。但是灵活的前提是mybatis无法做到数据库无关性，如果需要实现支持多种数据库的软件，则需要自定义多套sql映射文件，工作量大。

（3）Hibernate对象/关系映射能力强，数据库无关性好，对于关系模型要求高的

软件，如果用hibernate开发可以节省很多代码，提高效率。

132. mybatis 有哪些执行器（Executor）？

Mybatis有三种基本的执行器（Executor）：

1. **SimpleExecutor**：每执行一次update或select，就开启一个Statement对象，用完立刻关闭Statement对象。
2. **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内，供下一次使用。简言之，就是重复使用Statement对象。
3. **BatchExecutor**：执行update（没有select，JDBC批处理不支持select），将所有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。与JDBC批处理相同。

133. mybatis 分页插件的实现原理是什么？

分页插件的基本原理是使用Mybatis提供的插件接口，实现自定义插件，在插件的拦截方法内拦截待执行的sql，然后重写sql，根据dialect方言，添加对应的物理分页语句和物理分页参数。

134. mybatis 如何编写一个自定义插件？

转自：blog.csdn.net/qq_30051265/article/details/80266434

Mybatis自定义插件针对Mybatis四大对象（Executor、StatementHandler 、ParameterHandler 、ResultSetHandler ）进行拦截，具体拦截方式为：

- Executor：拦截执行器的方法(log记录)
- StatementHandler ：拦截Sql语法构建的处理
- ParameterHandler ：拦截参数的处理
- ResultSetHandler ：拦截结果集的处理

Mybatis自定义插件必须实现Interceptor接口：

```
public interface Interceptor {
    Object intercept(Invocation invocation) throws Throwable;
    Object plugin(Object target);
    void setProperties(Properties properties);
}
```

> intercept方法：拦截器具体处理逻辑方法
> plugin方法：根据签名signatureMap生成动态代理对象
> setProperties方法：设置Properties属性

自定义插件demo：

```
// ExamplePlugin.java
@Intercepts({@Signature(
  type= Executor.class,
  method = "update",
  args = {MappedStatement.class,Object.class})})
public class ExamplePlugin implements Interceptor {
  public Object intercept(Invocation invocation) throws Throwable {
  Object target = invocation.getTarget(); //被代理对象
  Method method = invocation.getMethod(); //代理方法
  Object[] args = invocation.getArgs(); //方法参数
  // do something ...... 方法拦截前执行代码块
  Object result = invocation.proceed();
  // do something .......方法拦截后执行代码块
  return result;
  }
  public Object plugin(Object target) {
    return Plugin.wrap(target, this);
  }
  public void setProperties(Properties properties) {
  }
}
```

一个@Intercepts可以配置多个@Signature，@Signature中的参数定义如下：

- type：表示拦截的类，这里是Executor的实现类；

- method：表示拦截的方法，这里是拦截Executor的update方法；
- args：表示方法参数。

（完）