1.
(1) Mybatis 的动态 Sql 用于根据不同的条件动态生产 Sql
(2) if / choose / when / otherwise / trim / where / set / foreach / script / bind
2.
(1)支持懒加载
(2) Mybatis 仅支持 association 和 collection 关联集合对象的延迟加载并且不能是多表连接
Configuration.lazyLoadingEnabled 懒加载开关
Configuration.aggressiveLazyLoading 当开启时，任何方法的调用都会加载该对象的所有属性
ResultMapping.lazy 会优先 Configuration.lazyLoadingEnabled
懒加载时，会使用 JavassistProxyFactory 动态生成代理对象，
当 aggressiveLazyLoading=true 时，调用任何方法都会执行所有的懒加载
当 aggressiveLazyLoading=false 时，当调用具体的懒加载对象的 get 方法时才加载
懒加载的对象保存在 ResultLoaderMap.loaderMap 中，当加载完成之后会删除
3.
(1) SimpleExecutor、ReuseExecutor、BatchExecutor、CachingExecutor
SimpleExecutor：每次开启一个 Statement 对象，用完立刻关闭 Statement 对象
ReuseExecutor：每次开始读或写操作，优先从缓存中获取对应的 Statement 对象。如果不存在，才进行创建。执行完成后，不关闭该 Statement 对象。
BatchExecutor：批量执行的 Executor 实现类
CachingExecutor：支持二级缓存的 Executor 的实现类
4.
(1)一级缓存：
使用 PerpetualCache，其内部为 HashMap
和 SqlSession 生命周期相同
update/delete/insert 操作会清空缓存
Sqlsessiond 调用 close/clearCache/commit/rollback 时也会清空
(2) 二级缓存
默认使用 PerpetualCache，也可以自定义（实现 Cache 接口）
全局缓存，跨多个 SqlSession
insert、update 和 delete 语句会刷新缓存
LRU 算法会回收缓存
只有当 SqlSession 执行 Commit 时才会把 TransactionalCache.entriesToAddOnCommit 放入缓存
5.
在四大组件 Executor/StatementHandler/ParameterHandler/ResultSetHandler 提供了插件机制
创建对应的对象之后，调用 interceptorChain.pluginAll 方法，其内部调用 interceptor.plugin 方法进行代理，最后返回代理后的对象
实现 Interceptor 接口，然后添加注解@Intercepts，最后配置插件到配置文件

```
@Intercepts({
        @Signature(type= StatementHandler.class,
                method = "prepare",
                args = {Connection.class,Integer.class})
})
public class MyPlugin implements Interceptor {
```

```
}
```

```
    <plugins>
        <plugin interceptor="com.lagou.plugin.MyPlugin">
            <property name="name" value="tom"/>
        </plugin>
    </plugins>
```