# Java项目开发中的3个小问题

2020-02-26

**1. mysql需要id自增主键吗**

有些时候我们会使用具有具体含义的字段作为主键，比如用用户名username而不是无意义的自增id。这样做的好处之一是可以利用主键不可重复的特性，保证username的惟一。如果试图写入重复的数据，数据库会抛出”Duplicate entry * for key PRIMARY”的错误。

利用这个“好处”的问题是，我们能否依赖以及是否需要依赖主键的默认特性（字段不重复）来实现我们业务上的需求（字段不重复），而事实上“不重复”是惟一索引的特性，并不是主键的特性。主键字段会自动加上惟一索引。

同时，利用这个“好处”的不便之处之一是，数据表缺少一个表示“数据行序列”的字段。oracle还有rownum这样的东西，mysql则没有，得用脚本实现。如果需要分段遍历表的话，没有行的概念就很难操作。

《阿里巴巴Java开发手册》明确要求表必备id字段，他们不一定是对的，但我相信他们的理由一定足够充分。

**2. Map还是Bean**

用mybatis查询数据时，返回值类型一种是List，一种是List。这里的Bean指数据对象（DO）。

使用Bean的好处显而易见，输入对象名再按下一个点，idea会自动弹出bean的getter和setter，表包含的属性一目了然。无论取值还是赋值，都对使用者友好。

使用Bean有可能存在的问题之一是，如果表结构有改动，比如删除一个字段，Bean类中对应删除了一个属性，同时删除了相应的getter和setter，那么只要是程序中用到了这个bean这个字段的地方，都需要逐一修改，如果不改，程序将无法通过编译。

我们经常在开发过程中遇到这种情况，调试A类，改了某个公用的COM类，编译的时候BCDEF…都报错了，这很让人抓狂，我只想要也迫切的只需要确认A类的问题是不是由COM类引起，却由于这种依赖关系需要尝试其他方法。

如果面临系统改造、整合、重构之类的问题，Bean也会暴露出同样的问题。我在最近接触的项目中，为了尽可能多复用原系统，尽可能少做改动，只好用含义不太相同的Bean.username储存实际上内容为Bean.phoneNumber的值。原系统的各种工具类（加解密、本地缓存之类）全都依赖Bean，涉及到Bean的改动之后，要么大批量的全改，要么全不改。

如果使用的是Map，Bean带来的问题可能会得到稍微的缓解。尤其是在合理处理取值操作后，比如`String.valueOf(map.get(""))`可以避免伟大的空指针异常，程序取值为空也可以正常启动，需要变动的地方随心所欲的get、set，不再受Bean类getter、setter方法的限制。

另外，其实Map和Bean都是对属性的一种封装。

**3. 注解还是xml**

Spring的IOC支持注解也支持xml，mybatis的sql同样支持注解也支持xml。

在以前的时代，Java项目往往会打包成war或者jar，放在容器tomcat或者resin中运行。容器在运行的时候会将包自动解压成class文件和资源文件，资源文件就包括xml。使用xml配置的好处之一是，线上环境可以直接修改xml重启项目就完成操作，不需要再把java文件编译成class文件，打包解压替换，走一遍项目上线的流程，拉分支、开发、测试、编译、部署、合并基线，一步步发邮件。

我想现在时代变了，有持续集成，有自动化运维，有properties，有yml，xml似乎显得不是那么重要。

对于mybatis的sql来说，内部有一个xml的解析器，xml相当于一种dsl，通过配置来实现动态sql语句之类的需求。如果使用注解的方式，把sql写在provider里，我们就失去了xml解析器的功能。不过失去能力的同时，我们也会获得自由。面对字符串形式的sql语句，我们完全可以自己封装一个parser。

一个有趣的现象是，国外的开发者似乎更喜欢注解的方式，国内的开发者倾向于使用xml，国外的书籍很多作者使用注解做案例，国内的博客文章则大多使用xml配置做教程。这一点有些类似于前端框架react和vue的状况，技术实力较强的企业、开发者更加青睐react，因为jsx更像是编程语言，有更多的发挥空间，vue则受到技术实力较弱的企业和开发者喜爱，因为上手简单，开发简单。如果你用过vue就应该有体会，用那玩意儿开发不叫编程，它只是软件工业化的产物。