数据表实体类3种：

对应于数据表的，---hibernate的对应；

非对应于数据表的，---实体类和数据表字段不需要对应；实体类扩展主数据表，加入关联字段；

---见ibatis中的实现；

继承实体类；---主实体类对应于数据表字段；继承实体类继承主实体类，然后加入关联字段；

---见mytatis中的实现；

~

实际使用，1和3，2的方式废弃；

实体类2种：

跟数据库相关的实体类；

跟数据库无关的实体类；---自定义的用于封装多参数，传递的实体类； | 命名：以bean结尾；

//////////////////////////////////////////////////////////////////////////////

类内容：

变量；

变量的get/set方法；

~

构造器：全参数，无参，有限参数；

应用1：

普通实体类；

应用2：---可以据需要加入json使用；

实体类包括实体类；

实体类包括列表；

应用3：

实体类-映射文件，见hibernate下的文档；

见ibatis，ibatis下文档；

应用4：

实体类-json，见json下文档；

应用5：

List,map等操作，见24数据操作-list等类型文档；

应用6：

实体类中包括一些的业务逻辑；

/////////////////////////////////////////////////////////////////////////////////////////////////////////

普通实体类：

**public** **class** LuceneStatus **implements** java.io.Serializable {

**private** **long** bigtable1Id;

**private** Integer lsManageType;

**private** Byte lsStatus;

**private** Date lsManageTime;

//无参数；

**public** LuceneStatus() {

}

//有限参数；

**public** LuceneStatus(**long** bigtable1Id) {

**this**.bigtable1Id = bigtable1Id;

}

//全参数；

**public** LuceneStatus(**long** bigtable1Id, Integer lsManageType, Byte lsStatus,

Date lsManageTime) {

**this**.bigtable1Id = bigtable1Id;

**this**.lsManageType = lsManageType;

**this**.lsStatus = lsStatus;

**this**.lsManageTime = lsManageTime;

}

实体类包括实体类：

1. ，普通实体类，
2. ，主实体类，

**public** **class** MobieMessageJson<H, B> {

**private** H head;// 泛型类

**private** B body;// 泛型类

填充：

MobieMessageJson<MobileHeadJson,AppStore> appJson = **new** MobieMessageJson<MobileHeadJson,AppStore>();

//AppStore appStore = new AppStore();

//appStore.setName("微信");

//appJson.setBody(appStore);

MobileHeadJson mobileHeadJson = **new** MobileHeadJson();

mobileHeadJson.setMethod(10011);

appJson.setHead(mobileHeadJson);

取值：

MobieMessageJson<MobileHeadJson,AppStore> appJsonGet = **new** MobieMessageJson<MobileHeadJson,AppStore>();

~

appJsonGet.get.get；

实体类包括实体类列表：

**public** **class** MobieMessageJsonT<H, B> {

**private** H head;// 泛型类

**private** List<B> body;// 泛型集合

列表，

//获取数据库表信息，转换存储到少字段的list类中；

MobieMessageJsonT<MobileHeadJson,AppStore> appJson = **new** MobieMessageJsonT<MobileHeadJson,AppStore>();

//数据库列表

List<TAppStore> list = *tAppStoreService*.queryRecordAll();

//转换结果List

List<AppStore> newList = **new** ArrayList();

TAppStore tAppStore2 = **new** TAppStore();

//转换操作

**if**(list.size()>0){

**for**(**int** i=0;i<list.size();i++){

tAppStore2 = (TAppStore) list.get(i);

//这个实体类要放在里面；

AppStore appStore = **new** AppStore(); appStore.setAppId(tAppStore2.getAppId().toString());

appStore.setApkUrl(tAppStore2.getApkurl());

appStore.setDescription(tAppStore2.getDescription());

appStore.setIconURl(tAppStore2.getIconurl());

appStore.setImageUrl(tAppStore2.getImageurl());

appStore.setName(tAppStore2.getName());

appStore.setPackageName(tAppStore2.getPackagename());

appStore.setSize(tAppStore2.getSize().toString());

newList.add(appStore);

}

}

appJson.setBody(newList);

MobileHeadJson mobileHeadJson = **new** MobileHeadJson();

mobileHeadJson.setMethod(10011);

mobileHeadJson.setStatus(1);

mobileHeadJson.setMessage("");

appJson.setHead(mobileHeadJson);

取值：

MobieMessageJsonT<MobileHeadJson,AppStore> appJsonGet = **new** MobieMessageJsonT<MobileHeadJson,AppStore>();

~

appJsonGet.get(0).get;

应用6：

实体类中，据需要可以添加字段；---实体类和表不需要对应；

·如果是查询的，只需要声明get方法；

如：

/\*\* 进度 进行计算\*\*/

**private** String scheduleStr;

**public** String getScheduleStr() {

**if** (**null** != **super**.getAccountYes()) {

scheduleStr = *df*.format(**super**.getAccountYes().multiply(**new** BigDecimal(100)).divide(**super**.getAccountReal(), 2, BigDecimal.***ROUND\_DOWN***));

} **else** {

scheduleStr = "0";

}

**return** scheduleStr;

}

//////////////////////////////////////////////

实体类中赋值-初始值：---实际测试下意义；

private Integer createBy = 0;

private Integer status = 0;

private Date createTime = new Date();

应该的，就可以在添加时，不用给int和date等类型的表字段赋初值，

框架、系统等会自动赋予初值；