1. 数据库表结构调整
   1. 表合并

独立的信息表合并到主对象表中，TransfromM合并到区域中，DepPos合并到DevInfo中。

* 1. 表分离

关联关系从主对象表中分类，单独做一个关联表。

1. 相应的传输类(WCF)分离

后续数据库表结构还可能进行调整，不能每次数据库调整，客户端都要相应的修改。

通过分离EF类和WCF类，根据EF类创建相应的WCF类供传输数据用，让数据库和WCF服务解耦，分别独立演变。

WCF这边可以按需传输需要的数据，把数据库表的内容按需传输给前端。

1. 在WCF服务端进行数据库类(EF)和传输类转换(WCF)

最早其实也有考虑分离的，但是那时觉得要维护两个类库比较麻烦，而且还要写相应的类对象转换代码。TModel这个类库很早就创建了，一直没用。

现在看来还是有必要的，然后最基本的方式就是两边维护两个相同的类，在WCF服务端进行转换处理。

为了方便创建新类（新表）写了个自动生成转换代码的工具，也是这次改造中为了提高速度，路径在：

<https://192.168.1.188/svn/IOM/LocationSystem/trunk/Location/Tools/ModelConvertCodeGenerator>

打开工具后，左侧是TModel的类，右侧是DbModel的类。

选择一个TModel的类，右侧会自动选中同名的DbModel类，不同命情况用[DbClass()]特性标注关联关系。

点击Generate则自动生成转换代码，拷贝添加到ModelConvertHelper中就行了。

1. 传输类说明

基本和原来的接口一致，比如：数据库中虽然DepPos和DevInfo合并了，在TModel中还是分开的两个类对象，DevPos还是DevInfo的一个属性。

部分类在数据库中改名了，在TModel中还是原来的名称，如Tag（标签）在数据库中改名为LocationCard（定位卡）了，在TModel中还是Tag。

先尽量维持原来的接口，后续需要改动可以客户端开发人员自己调整TModel中的类。

1. 现在3d客户端要做的工作
   1. 更新代码
   2. 删除数据库，两个都删除
   3. 运行WebSite重新生成数据库
   4. 运行Server，启动服务
   5. 打开3d项目，更新服务引用
   6. 测试客户端功能正确性

组织机构和区域列表测试过了，其他的没有测试完整。

* 1. 合并本地新代码

有问题找我和小楼。

现在还比较麻烦的是如何导入设备类型到新表中，已经当前的那个信息的设备类型表是否够用。这个王锴可以找我和小楼讨论一下。

1. 因为早期设计的不完善，增加了额外的工作量。这周前端主要把当前的功能合并完整，并和后端做数据对接，主要有两票、巡检；设备数据；告警数据。