OldW设计与使用规范

## 监测数据格式规范

### 点测点与线测点

在Revit中，测点分为“点测点”与“线测点”，其区别在于一个测点中是否有多个子节点。这里的节点并不一定是物理上的不同位移的子节点，而是广义上的在一个测点中可以监测的多项数据。比如测斜管、墙顶位移就属性线测点，而地表隆沉就属性点测点；

在Excel中，没点也分为“点测点”与“线测点”。其中点测点的监测数据保存在工作表的某个字段下，而线测点的监测数据保存在一整张工作表中，其中的字段即代表线测点中的每一个子节点。

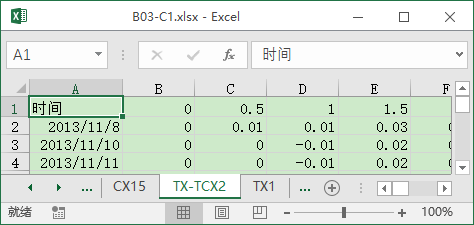
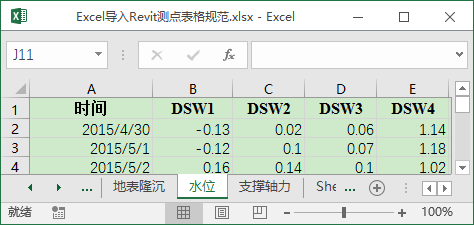
### Revit与Excel数据库测点类型映射

对于不同的监测点类型，在Revit中，通过对应的族Family的名称来区分；

Excel数据库中，每一个工作表的第一个字段名称必须是“时间”；其余的字段名称支持一般字符与数值，特殊符号可以包括“-”，但是不要包含“#”。

在Excel中，对于监测数据的区分稍复杂，具体列表如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | Revit族名称（默认编号） | 图标 | Excel数据库匹配准则 |
| 1 | 墙体测斜（CX01） | F:\Software\Revit\RevitDevelop\OldW\Resources\icons\MonitorIncli_32.png | 每一个测斜点的数据用一个工作表来保存；  每个墙体测斜工作表的名称必须以“CX”开头；工作表中的数值字段名的允许格式有：03、03.2、03Dot2 |
| 2 | 土体测斜（TX01） |  | 每一个测斜点的数据用一个工作表来保存；  每个土体测斜工作表的名称必须以“TX”开头； |
| 3 | 墙顶位移（QD01） |  | 墙顶位移在Revit中是按线测点进行归类的，因为在此一个测点上相当于有两个子节点，分别代表水平与垂直的位移。但是在Excel中，考虑到一般的习惯，它是按水平位移与垂直位移这两个点测点的数据表格式存储在两个工作表中的。在二者进行交互的过程中，会为墙顶位移测点的Import与Export方法进行重写。  墙顶位移的监测数据包括水平位移与垂直位移，在Excel中通过两张表“墙顶水平位移”与“墙顶垂直位移”来保存。 |
| 4 | 地表隆沉（DB01） | F:\Software\Revit\RevitDevelop\OldW\Resources\icons\MonitorSet_32.png | Excel工作表名称以“地表隆沉”开头 |
| 5 | 立柱隆沉（LZ01） |  | Excel工作表名称以“立柱隆沉”开头 |
| 6 | 支撑轴力（ZL01） | F:\Software\Revit\RevitDevelop\OldW\Resources\icons\MonitorForce_32.png | Excel工作表名称以“支撑轴力”开头 |
| 7 | 水位（SW01） |  | Excel工作表名称以“水位”开头 |
| 8 | 其他点测点（PM01） |  | 其他未在上面标记过的测点类型，其每一个点测点的监测数据都保存在工作表中的某个字段下。  Excel工作表名称以“PM”开头 |
| 9 | 其他线测点（LM01） |  | 其他未在上面标记过的测点类型，其每一个线测点中有多个子节点（类似于测斜管），一个测点的监测数据保存在一张工作表，而表中的每一个字段代表此线测点中的一个子节点。  Excel工作表名称以“LM”开头 |

格式范例： 

### 测点名称

在Excel数据库中，不论是线测点（工作表名）还是点测点（工作表中的字段名），其名称支持DB12以及DB2-7这两种模式，但是不推荐小数点。

对于线测点中每一个子节点的字段名称，支持小数点。

## 基坑开挖建模规范

### 模型土体的筛选

### 开挖土体的筛选

开挖土体必须位于模型土体所在的组Group中，其类型为FamilyInstance，类别为场地OST\_Site。

## OldW设计

### Transaction作为修改Revit时的虚假形参

在要对Revit文档进行修改的方法中，必须要定义一个Transaction类型的形参，但是这个参数在方法内部是不进行任何使用的。

它的存在只是为了确保在调用此方法之前，以及打开了相应的事务。而对此事务的开启、提交或者回滚都应该是在此方法外部实现的。

### Revit参数的交互

对于Revit文档Element中的每一个相关参数，都要设置对应的Get与Set方法，并且设计一个private变量来作为此参数值的临时变量。

临时变量的目的是避免高频地与Revit文档进行数据交互，而是直接通过面向对象的内存数据作为纽带。

对于一般的值类型数据，可以通过Nullable<值类型>来作为临时变量，这样的话在Get参数数据时，就可以通过对此临时变量是否为null的判断来决定是否要从Revit中提取数据了。

|  |
| --- |
| /// <summary> 临时变量的目的是避免高频地与Revit文档进行数据交互，而是直接通过面向对象的内存数据作为纽带。 </summary>  private string \_monitorName; |
| /// <summary> 提取测点的名称 </summary>  public string GetMonitorName()  {  return \_monitorName ?? Monitor.get\_Parameter(Constants.SP\_MonitorName\_Guid).AsString();  } |
| /// <summary> 设置测点的名称 </summary>  public void SetMonitorName(Transaction tran, string MonitorName)  {  Parameter para = Monitor.get\_Parameter(Constants.SP\_MonitorName\_Guid);  para.Set(MonitorName);  // store its name in the private variable.  this.\_monitorName = MonitorName;  } |

## 报错与解决

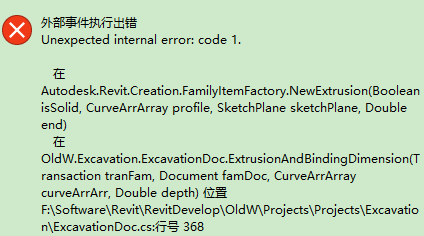
### 创建拉伸实体时出现internal error

Extrusion extru = familyCreation.NewExtrusion(

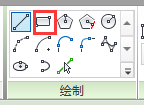
isSolid: true, profile: curveArrArr,

sketchPlane: sp, end: depth); // 创建拉伸实体

在绘制模型土体或者开挖土体时，在创建拉伸实体操作中出现如下报错：



具体原因：NewExtrusion函数说明：If creation was successful the new Extrusion is returned, otherwise an exception with failure information will be thrown. 比如，profile中的Curve不能依次连接成封闭的CurveLoop（即使在几何上，这些Curve是满足要求的封闭曲线），则会出现如下InternalException报错：Unexpected internal error.

具体的原因还未找到，但是在测试中，发现如果通过绘制出来的四边形矩形在创建土体时就会出现如上报错。但是通过其他的工具（比如绘制线条、多边形、圆形等）绘制的轮廓就不会报错。

其可能是原因是：上面的NewExtrusion方法中，对于profile中曲线集合的顺序有要求，可以需要其按连续的顺序来存放。

# OldW 帮助文档