**创建事件**

接受除了id，EventID之外所有的字段，存入数据库中。在relation表中记录事件id，用户id，Create。

**参加一个事件**

传入 Userid, EventID, showcontact，Message

首先确认， 用户id和事件id的关系，若是member，则返回 @已经在活动中了，无需参加， 若是Checking，则返回 @正在审核中，无需重复提交审核。 若是Create，则返回@不可参加自己创建的事件。

在确认用户和事件没有关系后，获取事件的NeedPermission属性，

若为否，判断 showContact是否为真，若showcontact为真，则将用户，事件id，Member，用户表中的Contact以Key：Value;记录到数据库中，返回@参加成功， 若showcontact为假，则将用户，事件id，member记录到数据库中，返回@参加成功。

若needPermission为真，则重复showcontact的判断流程，

记录用户，事件ID，Checking，和message到数据库中。

**获取一个事件的信息**

**接受事件id，userid**

**添加属性，PersonCount=0，isMember=false，Members=[ ],**

**首先选中relation表中所有事件id的记录，若为Member，则在PersonCount+1，在Members内加入用户ID所对应的名字,若有Message则用|分割一并加入，即Members对应一项中的内容应该为\*（Name|Message）。如果Member信息中有和传入的UserID相同的，则将isMember设为true**

**若为Create，同上，Checking则不做要求。**

**返回所有事件**

**按照返回一个事件的标准，返回所有事件。**

**返回和一个用户有关的所有事件**

接受Userid

返回的值将被分为四个部分，分别储存在一个数组的0,1,2,3位置

其中分别代表了 用户创建的事件， 用户参加的事件， 用户申请加入的事件， 其他用户发来的申请。

首先读取Relation表中UserId所对应的所有事件。按照Type字段的值有分别处理。若是Member, 则放在数组的第二位，若是Checking，则放在数组的第一位。若是Create， 则放在数组的零一位。而对于每一个关系为Create的事件，获取事件id所对应的所有Type为Checking的用户Id，对于每个id，分别把事件存储到数组的第四位，并在存储的时候将这个Userid存入额外添加的inUser属性中。