亚星指出的:

ItemView改名为ListView // 遵循MFC等主流UI框架的命名

DataModelInterface 的纯虚析构方法改成虚方法 // 纯虚析构方法在场景”这个类必须是抽象类, 但是我没有合适的虚方法”时是有用的, 但是此处没有必要.

宽字符和utf8互相转换时不必要的操作 // 清理一下

SetListener 的参数从含义不明的UserData改成接口类 // 覆盖listener时可以通过销毁接口类指针销毁listener对象, 如果只是DWORD类型的UserData会造成内存泄露

实现类要保留虚函数关键字 // 为了代码阅读方便, 不引起混淆

lua封装类接口~~const~~ // 这样做会限制实现类的行为, 封装类不应如此

XmlDataModel :: m\_playlist 管理 // loader持有playlist的引用是为了方便加卸载, playlist的创建和销毁应该留在XmlDataModel里处理

lua function ref/unref // function的ref和unref要成对出现

用playlist.clear() 代替while循环销毁playlist中的对象 // 容器中的对象有指针成员且指针指向对象需要被销毁时, 还是要用while循环销毁, 容器中的对象只有对象成员时可以用clear()

GetItemAtIndex(void \*\*itemData) 的内存泄露 // 对于string, push到lua环境中后要释放掉为此开辟的新内存

春哥指出的:

TreeView在scroll range发生变化时scrollbar的thumb button大小不会发生变化, 要动一下才会变化 // 在setScrollRange时更新一下

**主题:** 答复: ItemViewDemo及文档

关于c++层的建议

1. 不要在dllmain里面做实际工作，如果是独立dll，那么初始化和反初始化要么有明确的导出接口，要么在内部第一次使用的地方初始化

<?> 改dll. 只是注册lua实例和lua封装类的时机和由谁调用不是很清楚.

1. 不要使用c++的异常处理，我们自己的代码都不应该跑出异常，公共库所有接口都不会抛出异常

检查代替异常

1. 编码转换使用公共库的transcode辅助类

使用辅助类

1. lua\_pushnil(luaState); return 1;  返回值无意义的话就不需要返回值了

去掉无意义返回值. 在lua方法期待返回值的时候, 如果失败也会放一个nil, 否则lua会得到栈上无关的数据会被pop作为返回值吧?

1. LuaDataModelClass使用模板，有些重型，如果只是为了对指定接口进行lua封装，使用接口会更方便和优雅
2. 如果LuaDataModelClass操作接口基类DataModelInterface, XmlDataModel实现DataModelInterface, 这种实现我在注册lua类和lua静态对象时遇到问题; (lua调用时加类名作为参数, Factory:CreaetInstance()和LuaDataModelClass实现静态方法的时候都先从luaState里取出DataModelClass类名参数?)

<?>你指的是那种?

1. AttachSingleDataReadyListener会覆盖老的listener，命名成SetxxxListener会更好，不容易误解；或者内部使用事件队列用以管理多个事件(这里没必要)

改方法名

7、XmlDataModel应该是数组管理核心，数据存储和管理应该在此层去做；xmlparser更多的应该只是解析，输出结果给datamodel，而本身应该是无状态

8、xmparser里面做了解析xml和加载数据(bitmap)，所以更多的应该是叫DataLoader，而xmlparser应该是DataLoader的更下面一层，DataLoader使用xmlparser来解析xml，并且负责加载图片这些，DataLoader和DataModel交互

         7. 8. 条, 增加DataLoader类, 移动解析/加载工作. 改死了.

9、核心数据结构使用了重型的mutex，首先单进程更多的应该选择使用关键代码段，criticalsection，效率高很多

10、核心数据结构最好设计成无锁的，子线程加载完毕后，投递结果到主线程，主线程再合并

         9. 10条, 改用critical section处理加载/卸载队列, 核心数据结构playlist去掉锁, 子线程对playlist只读不写, 结果post message到主线程自己处理.

11、UIThreadDll这个工程有些多余，之直接当成一个辅助类就ok，没必要使用一个单独的dll，作为一个单独dll太重型

         UIThreadDll的类和方法搬到DataModel里直接用了.

关于lua层的建议

1. 控件提供的方法有些少，很多查询方法都没有，比如
   1. 查询控件当前item数量，正在显示的item索引等等，虽然数据模型可以直接得知，但是这也是listview基础的接口
   2. 查询当前整体视窗大小，可显式视窗大小等，这个可以考虑增加

增加GetDataCount(), GetColumnCount(), FirstVisibleItemIndex(), GetScrollSize(), GetViewportSize()

1. 控件的一些核心事件还是需要，listview的使用者可能需要监听，比如下面
   1. 滚动了位置
   2. 增加了数据
   3. 某些数据需要显示/隐藏了

增加事件VerticalScrollPosChanged(oldpos, newpos), HorizontalScrollPosChanged(oldpos, newpos), VisibleItemChanged(newfrom, newto)

1. 如何删除和中间插入一个条目？是数据模型提供相应的事件，还是直接调用listview接口来完成？ 貌似放在datamodal比较合适~

认为这个应该是DataModel提供的. ListView要监听DataModel datachanged事件

         在listview的实现中增加:

         dataModelCallbackTable.AttachDataInsertedListener(function onDataInserted(index) if index is already loaded then …end)

         dataModelCallbackTable.AttachDataRemovedListener(function onDataRemoved(index) if index is already loaded then … end)

         dataModelCallbackTable.AttachDataChangedListener(function onDataChanged(index) if index is already loaded then … end)

         或者干脆

         dataModelCallbackTable.AttachDataRangeChangedListener(function onDataChanged(fromIndex, toIndex)… end)

         在dataModelCallbackTable中增加Attach…Listener方法, 如果使用者打算改变数据的话

1. scrollbar实例化指定的id，要避免重复，最好能是控件id+内部id形式，试想一下假如界面上放两个listview情况

if attr.ScrollBarClassNameV then

                   local objFactory = XLGetObject("Xunlei.UIEngine.ObjectFactory")

                   local scrollObjV = objFactory:CreateUIObject("scrollbarV", attr.ScrollBarClassNameV)

                   self:SetScrollBarV(scrollObjV)

         end

         改一下id = 控件id+ list view id

1. 、ItemView没有提供单元格被选中, Hover 或者 Click的事件. 因为由使用者提供的Item Factory来创建单元格对象, 使用者有机会在返回对象之前绑定想要监听的事件.

这个应该是用户自己的单元格对象可以自定义各种消息，因为用户自定义item控件的zorder比较高，所以可以优先接收鼠标和键盘消息

         这个是改文档对吗.

1. 、滚轮消息的处理

listview理论上可以在任意地方滚滚轮，listview都可以处理，所以这里就是难点，因为如果焦点在内部item上时候，listview以及scrollbar无法收到滚轮消息，所以可以以下两点：

1. 引擎考虑增加OnControlMouseWheel事件，在里面可以监听所有子对象上的mousewheel事件，这样处理最方便
2. 子控件收到mousewheel后，显式的把消息路由给listview，listview再传到给scrollbar，这样处理比较复杂

这个问题我也要考虑下

         除了MouseWheel, mouseDown + MouseMove + mouseUp也会滚动滚轮.

         我认为子控件应该有处理鼠标事件的机会. 子控件检查过不需要处理再routeToFather就可以了. 在特定的情况下, 子控件可能不想让滚轮滚动.

先想到这么多，文档写的很详细~  还需要花点时间慢慢看，有问题再详细讨论