



# XLibEditor User Guide & License

Wing Xu E-mail: [eawtest@163.com](mailto:eawtest@163.com)

## Features:

XLibEditor is a general-use database editor for easy use. You can customize the template for database easily, and create the database for game or other applications quickly, convenient for mass data input that in different type, supporting general PLIST file format and Cocoa NSKeyedArchiver file.

XLibEditor supporting multiple data types, and you can make reference of resources in different class, auto preview images by path, multiple language string, etc.

## Use Flow:

- 1.Create template for your database in Template Editor.
- 2.Create and edit datas in main interface.

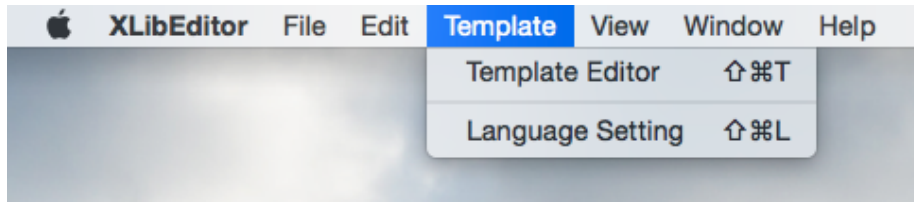
## Example:

Create a template, add types for “Item” & “Monster” , create properties for these types like “Name” “ATK”. Create resource named “Iron Sword” under “Item” type in main interface and edit the property, save the database file and import it into your game.

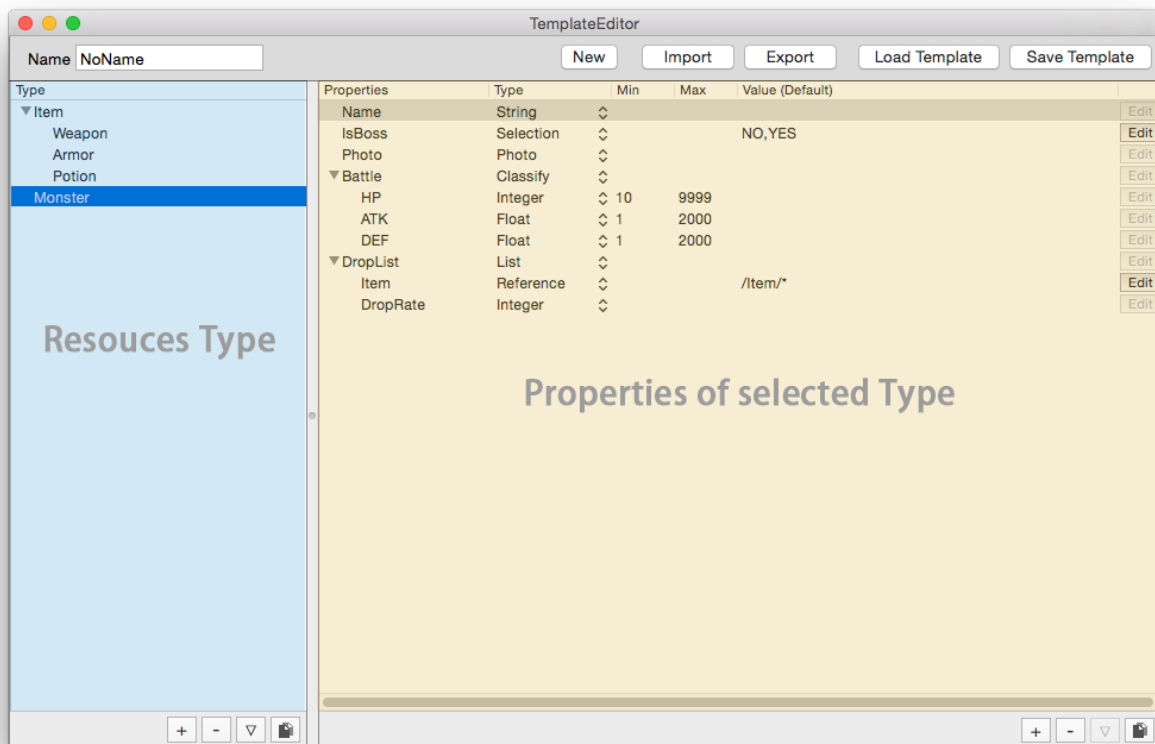
(This document contains these parts: Template Edit、 Database Edit、 File Format、 License)

## ===== Template Edit =====

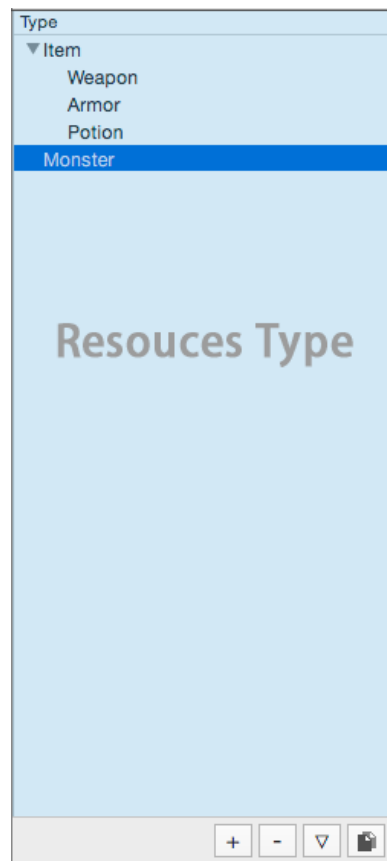
Because of you can create the template once and use it repeat, and most time of developing is edit the database, so the Template Editor is hide by default, you can open it in application menu:



Main interface of Template Editor:

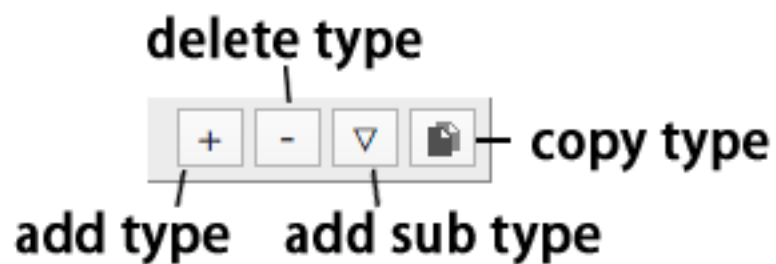


## 1. Resource type edit zone



Resource Type managed by tree diagram.

Features of buttons at bottom:



\*You can change the type position in tree diagram by drag.

## 2.Resource properties edit zone

Properties	Type	Min	Max	Value (Default)	
Name	String	↕			Edit
IsBoss	Selection	↕		NO,YES	Edit
Photo	Photo	↕			Edit
▼ Battle	Classify	↕			Edit
HP	Integer	↕ 10	9999		Edit
ATK	Float	↕ 1	2000		Edit
DEF	Float	↕ 1	2000		Edit
▼ DropList	List	↕			Edit
Item	Reference	↕		/Item/*	Edit
DropRate	Integer	↕			Edit

### Features:

Properties: Properties of this Resource Type (name)

Type: data type of this property

- Integer: Same as integer in C language, store integers.
- Float: Same as float in C language, store digits with decimal.
- Keyword: Store text. Not supporting multi-language.
- String: Store text. Supporting multi-language.
- photo: Will popup a file select panel to help you select photos, and supporting image preview. Supporting multi-language.

- Selection: You can create a selection menu and define items, the data store in database will be the index of selected item  
( Example: Create a property named “Element” for equipments, and create a selection menu for it, prevent wrong input ).
- Reference: Set a reference to a resource in different type, database will store the full path of referenced resource (/parentType/childType.../lastChildType.resourceName, refer to the image below).
- Classify: Allowed to add sub properties, use to classify properties.
- List: Refer to the image below, use to create a list that has repeat properties.

#### Template:

▼ DropList	List	↕		Edit
Item	Reference	↕	/Item/*	Edit
DropRate	Integer	↕		Edit

#### Database:

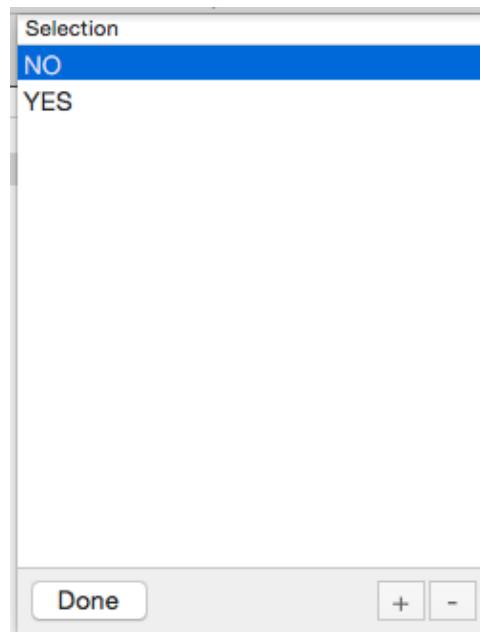
▼ DropList		↕		Edit
▼ 0		↕		Edit
Item		↕	/Item/Weapon.Wood Sword	Edit
DropRate		↕	5	Edit
▼ 1		↕		Edit
Item		↕	/Item/Armor.Iron Armor	Edit
DropRate		↕	2	Edit
▼ 2		↕		Edit
Item		↕	/Item/Potion.Apple	Edit
DropRate		↕	15	Edit

- Min: Minimum value of this property (when the property type set to “List”, this means the minimum count of the list)
- Max: Maximum value of this property (when the property type set to “List”, this means the maximum count of the list)
- Value: When the property type set to numeral types or “String”, this cell means default vale. When the property type set to “Selection”, this cell show items of selection menu. When the property type set to “Reference” show the path of referenced type.

\*You can change the value by double click the cell or push the edit button at right. Buttons at bottom are similar to resource type edit zone. They are used for add property, delete property, add sub property, copy property.

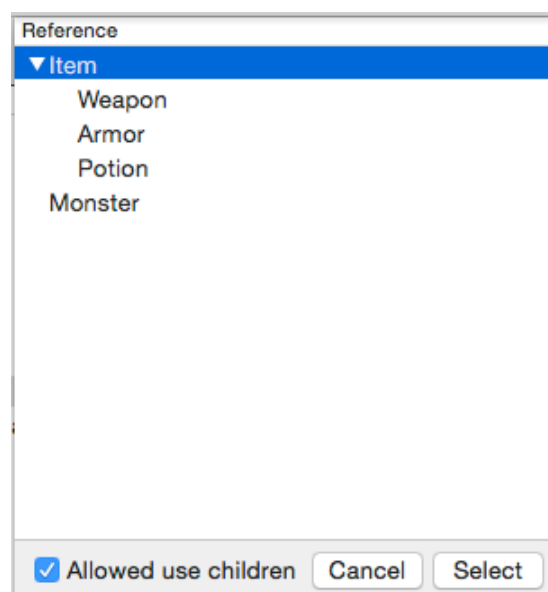
\*Selection menu edit:

(You can add&remove items with + - button, and change the item name by double click the cell)

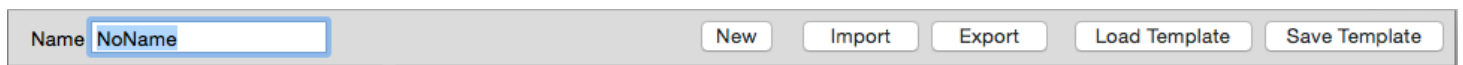


\*Reference edit:

(Database can only reference the resource under selected type by uncheck "allowed use children", can reference to resource under selected type and children types by check)



### 3.Template file manage zone



The screenshot shows a horizontal toolbar with a 'Name' label followed by a text input field containing 'NoName'. To the right of the input field are five buttons: 'New', 'Import', 'Export', 'Load Template', and 'Save Template'.

Name: Set name for editing template.

New: Delete editing template, create an empty template.

Import: Import template from file.

Export: Export template to file.

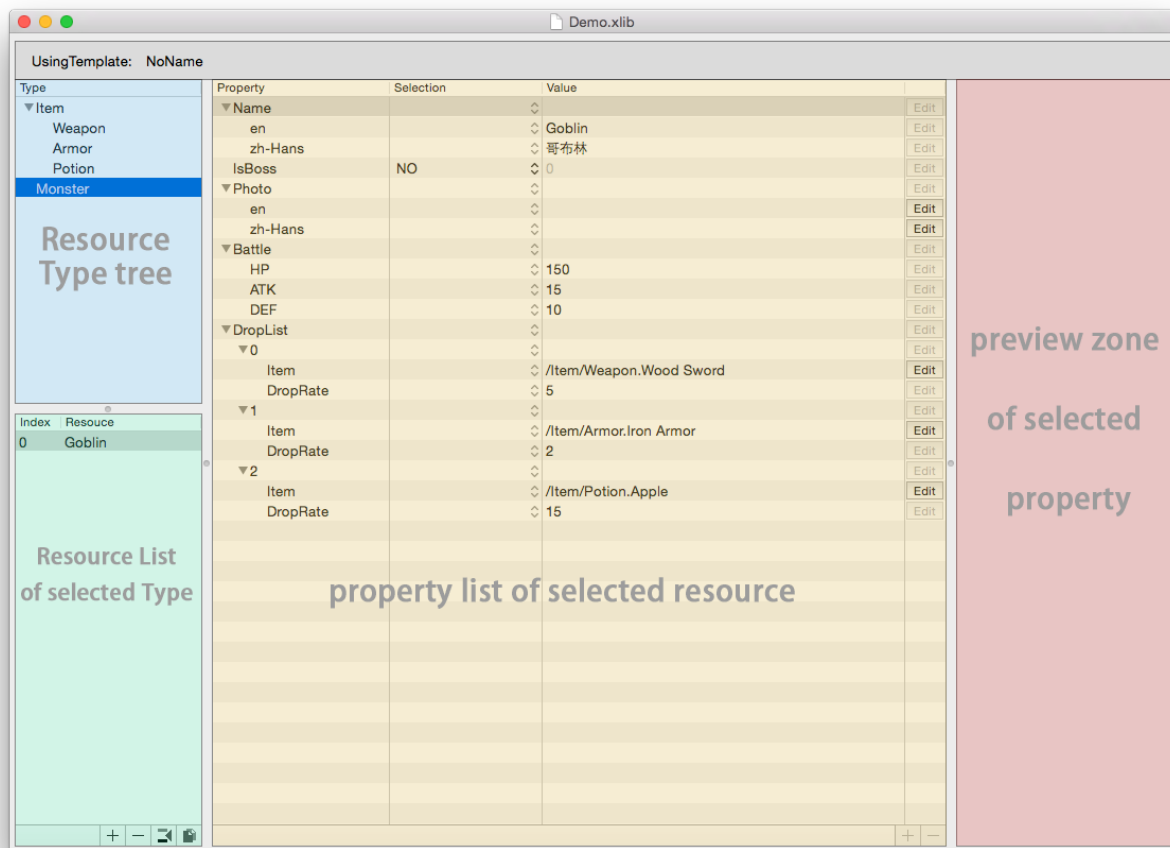
Load Template: Load template from editor.

Save Template: Save template into editor, and change template for all opened database.

\*The template that editor using is store in User Defaults. Because of time and energy (and lazy), XLibEditor does not support multiple template yet, when database is not match with template, it will delete the datas that not exist in new template, also add new datas and set to default value. Maybe add more flexible template manager later. When you load a wrong template, just don't save the template or database file and quit.

# ===== Database Edit =====

Main interface:



## 1. Resource tree zone at up left

There is no add, delete or drag action for this zone, this zone showing the tree diagram of your database, you can edit resources of a type by select it.



## 2.Resource list

Index	Resource
0	Goblin

Index: Index of resource in it's type.

Resource: show the name of resource.

Buttons at bottom use for add resource, delete resource, insert resource, copy resource. Drag to change the position of resource, double click the name to edit.

\*No worry about references when change position or name for resource, reference will track to the resource and change the reference path automatically. But if you change the name of type in template, reference will be invalid.

### 3.Properties edit zone

[illegible]

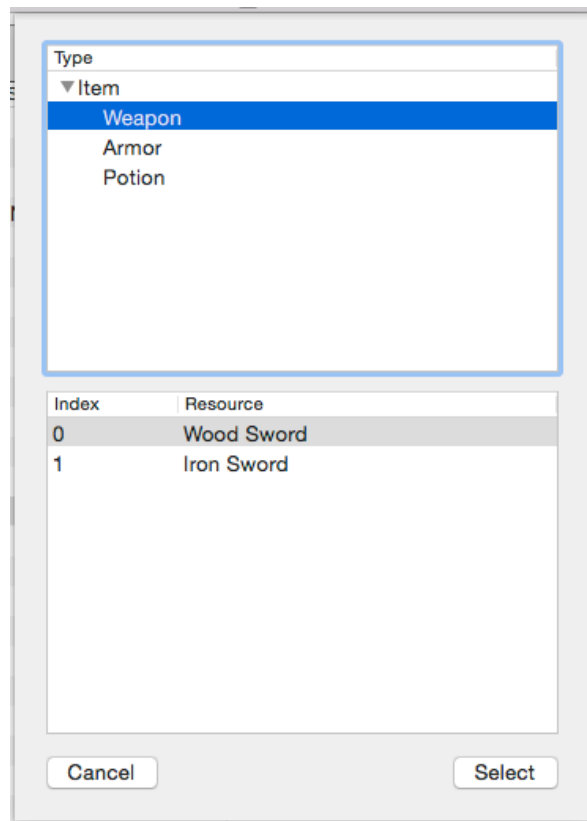
Property: show all properties of selected resource.

Selection: Show selection menu create in template, “value” cell at right will show selection index after change. Will disable if this property isn’t “Selection”.

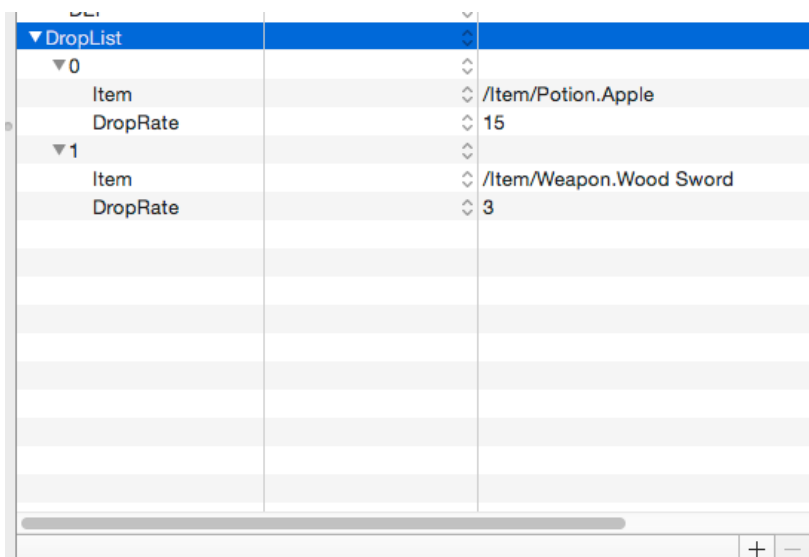
zh-Hans		哥
IsBoss	✓ NO	0
▼ Photo	YES	
en		/G
zh-Hans		

Value: Show value of the property, you can edit it by double click. You can just delete the value of “photo” or “reference” if you don’t want it.

Edit: For “Photo” & “Reference”, will popup a panel to help you select.

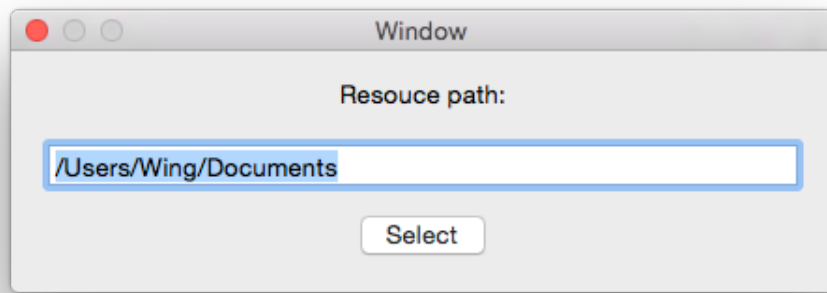


+ & – buttons at right bottom can add and delete “List” item.





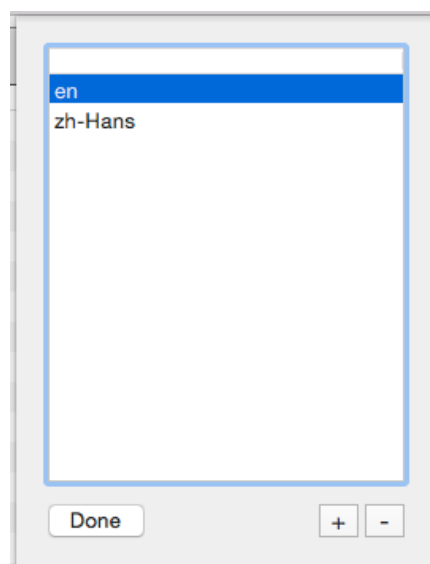
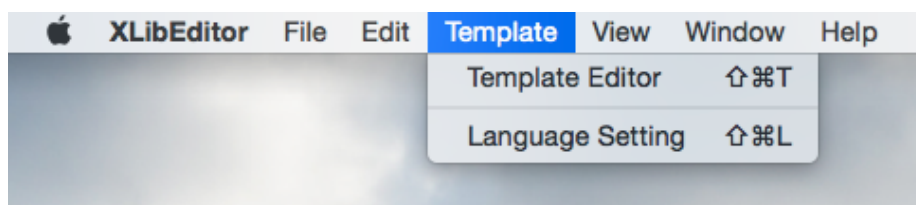




\*Considered that a game/app may develop for different language, XLibEditor added multi-language support, "String" and "Photo" property will automatically create sub properties base on language list:

▼ Name		
en		Goblin
zh-Hans		哥布林
zh-Hant		哥布林

You can open Language Settings in application menu ( I think you already know how to edit this : ) :



## ===== File Format =====

XLibEditor now supporting two types of file formats , one is NSKeyedArchiver format, the other is PLIST (just XML in fact) .Because XLibEditor is an open source software, so you can also add file format support by yourself.

Inside NSKeyedArchiver format, it is a nested structure with NSArray & NSDictionary, it's supported by Cocoa directly. You can create a small database (there's already a demo.xlib), import it into your application, use NSLog to show the stuff inside it. Also you can save it as PLIST file, and use PLIST/XML editor to check it out.

You can use these codes to import the file:

```
NSString *dataPath = @"your file path";
NSData *data = [NSData dataWithContentsOfFile:dataPath];
NSKeyedUnarchiver *Unarchiver = [[NSKeyedUnarchiver alloc]
initWithReadingWithData:data];

NSString *templateName = [Unarchiver decodeObjectForKey:@"templateName"];
NSArray *languageList = [Unarchiver decodeObjectForKey:@"languageList"];
NSArray *dataArchive = [Unarchiver decodeObjectForKey:@"data"];
[Unarchiver finishDecoding];
```

Also you can extract read/write methods from source code of XLibEditor. in this way, you can read the file and restore it back to objective-c object structure. Depends on your demand, you can simplify them or use them directly.

Class of Resource is BEResource, class of Property is BEProperty, and methods for restoring are in BEDataController. But in this way, it needs template to load, if you don't need template in your application, it have to take some time to clear it, so maybe use original NSArray & NSDictionary will be easier.

NSKeyedArchiver format have two types of suffix, depends on what kind of file it is, template is .template, and database is .xlib. This is for prevent wrong use.

The second file format is PLIST, it's also supported by Cocoa directly. On other platforms, you can find parsing libraries for PLIST / XML.

Both template file and database file can save as PLIST, but because of the inner structure is different, so if you open files wrong, there will be no problem.

You can use these codes to import the file with Cocoa (you get a same structure as NSKeyedArchiver format) :

```
NSString *dataPath = @"your file path";
NSData *data = [NSData dataWithContentsOfFile:dataPath];

NSPropertyListFormat format;

NSDictionary *dictionary = [NSPropertyListSerialization
    dictionaryWithData:data
    options:NSPropertyListMutableContainersAndLeaves
    format:&format
    error:nil];

NSString *templateName = [dictionary objectForKey:@"TemplateName"];
NSArray *languageList = [dictionary objectForKey:@"languageList"];
NSArray *dataArchive = [dictionary objectForKey:@"data"];
```

This project including AES256 encryption class extension for NSData made by Apple (NSData+AES256), but I can't sure the encrypted file can be decrypt on other platforms, and because of time and energy (and lazy), so I didn't use it in XLibEditor. If you need, you can use these codes where save and load documents:



```
NSData *EncryptedData = [data AES256EncryptWithKey:key];
```

and:

```
NSData *decryptedData = [data AES256DecryptWithKey:key];
```

Then you can encrypt and decrypt your files (well, maybe these stuffs are basics) .

And I used path for Resource Reference, because it's can adapt multilevel types and track referenced resources easily. You can parsing the string to find what resource it's pointing. There's codes inside the source code of XLibEditor. Also don't forget, do not use "/" and "." symbol in name of types and resources.

===== License =====

Copyright (c) 2015, Wing Xu  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED

WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.