魔兽插件编写普及教程

ADDONS 编写普及

作者：VVER

前言
22 号基本写完。比预计的要快

这个不是数学书或者语文书。我不想也没有那个能力写成那样.....
所以请抱着看小说的态度来看。

因为 ADDONS 的编写是一个整体。我实在无法分清哪个是要先说 哪个是要后说
所以有看不懂的地方就跳过去。接着看下面

前几层楼写的相对详细些。后几层楼更多的是提示。我还是希望看官能自己分析，自己动手研究

最后。短短的 6 层楼包含了我半年的心血，而且我语文课真的是没好好去上
所以如果一时看不懂。请多读几遍 或者回帖指出，我会尽力解释的

基础

如果你完全的彻底的不了解什么是 ADDONS，请先参考 CAMUS 老大的帖子的 2 楼：
插件，宏 常见问题解答

任何一个 ADDONS 都是由 3 种文件组成
LUA XML TOC

TOC 是 ADDONS 的标签，就好比是一本书的封面
他将告诉 WOW 我们编写的 ADDONS 的名字。作者。版本号。注释。主要文件，依赖文件，保存文件
所以任何一个 ADDONS 中 TOC 文件都是必不可少的

举个例子
我们编写如下信息保存为 firs.TOC 文件

```
## Interface: 1600
## Title: 我是一个插件
## Notes: 虽然我没有内容，但是我真的是插件
```

然后我们进入游戏。会发现在插件列表中真的出现了我们的这个 ADDONS。
虽然没有任何作用。但是我们还真的迈开了第一步

XML 是定义 ADDONS 组成的框体（或者用元素这个词更贴切些。不过我更喜欢框体这个词）
我们的 ADDONS 有多少个按扭。多少个文字。每个按扭长什么样子。具体在什么位置 等等等等
都是在 XML 文件中定义的。我会用大量的篇幅介绍 XML 文件的写法。敬请期待

如果 XML 是定义 ADDONS 静态的框体。那么 LUA 就是把这些框体动态化
比如说按扭在什么时候弹出。血条在多少的时候改变颜色。进入战斗的时候把人物血条移动到屏幕中间
等等等等。都是在 LUA 中定义

总之，LUA 就是把死的框体 真正的发挥出作用来
当然 LUA 还可以使得我们的代码简化，本地化。等等等等
一个 ADDONS 中 LUA 并不是不可或缺的。但是一个 ADDONS 编写的好与坏。就看 LUA 的了

StatusBar

在之前我们大概知道了 ADDONS 的一些基本概念

那么现在。在各种类型的框体中挑一个 StatusBar 来说说

基本概念
StatusBar：是 WOW 中用来定义类似施法条。进度条之类的一种框体，
说白了就是可以根据某个数值，实时的改变条条的长短

一个 StatusBar 有 3 个重要的参数
1。最大长度 maxValue
2。最小长度 minValue
3。当前长度 Value

要动态的改变 StatusBar 就需要用脚本（Scripts）中的事件实时的设置当前长度（Value）
（这句话可能有点饶口。不过我就这点语文水平了。。见谅见谅）

准备工作

首先根据我们之前的概念。一个 TOC 文件是必不可少的，编个 first.toc
```
## Interface: 1600
## Title: 我的第一插件
## Notes: 真的是我的第一插件哦
```

然后我们当然得用 XML 来定义 StatusBar 这个框体，那么编写个 first.xml
不过 WOW 并不知道我们写了 first.xml 这个文件。我们得告诉他。所以在 first.toc 中加一句（红色的部分）
```
## Interface: 1600
## Title: 我的第一插件
## Notes: 真的是我的第一插件哦
first.xml
```

好了 正式开始编写 first.xml

按照基本的 XML 文件格式 先写好基本的嵌套
```
<StatusBar>
</StatusBar>
```

起个好名字
```
<StatusBar name="haomingzi">
</StatusBar>
```

注意红色的那句。现在我们名叫 haomingzi 的 StatusBar 会根据变量 SVALUE 自动改变长短了
（这里我们用到了脚本。后面会详细解释的。先记得<Scripts>是脚本就可以了）
```
<StatusBar name="haomingzi">
                    <Scripts>
                            <OnUpdate>
                                    haomingzi:SetValue(SVALUE);
                            </OnUpdate>
```

```
                </Scripts>
</StatusBar>
```

现在我们把变量 SVALUE 设置为玩家的血量
那么：

```
<StatusBar name="haomingzi">
                <Scripts>
                        <OnUpdate>
                              haomingzi:SetValue( UnitHealth("player") );
                        </OnUpdate>
                </Scripts>
</StatusBar>
```

OK。一个玩家的 HP 条就写出来了

当然仅仅这几行还远远不够，继续完善下
先把最大和最小长度定义好

```
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <Scripts>
                        <OnUpdate>
                                haomingzi:SetValue( UnitHealth("player") );
                        </OnUpdate>
                </Scripts>
</StatusBar>
```

这里我们设置的是 0 到 100。显然玩家的血量肯定不会在 0 到 100 之内的。
那么我们就要把他转换为百分比

```
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <Scripts>
                        <OnUpdate>
                              local SVALUE=( UnitHealth("player") / UnitHealthMax("player") )*100
                                haomingzi:SetValue(SVALUE);
                        </OnUpdate>
                </Scripts>
</StatusBar>
```

设定下他的位置，比方把他放在屏幕的中间
（具体的如何设置位置。在楼下会讲。现在只要知道红色部分的代码是设定位置就可以了）

```
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <Anchors>
                        <Anchor point="center" relativeTo="UIparent" relativePoint="center"/>
                </Anchors>
                <Scripts>
                        <OnUpdate>
                                local SVALUE=( UnitHealth("player") / UnitHealthMax("player") )*100
                                haomingzi:SetValue(SVALUE);
                        </OnUpdate>
                </Scripts>
</StatusBar>
```

设定一下大小

```
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <Size>
                        <AbsDimension x="70" y="8"/>
                </Size>
                <Anchors>
                        <Anchor point="center" relativeTo="UIparent" relativePoint="center"/>
                </Anchors>
                <Scripts>
```

```
                        <OnUpdate>
                                local SVALUE=( UnitHealth("player") / UnitHealthMax("player") )*100
                                haomingzi:SetValue(SVALUE);
                        </OnUpdate>
                </Scripts>
</StatusBar>
```

当然他长什么样子我们还没弄呢~

```
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <BarTexture file="Interface\TargetingFrame\UI-StatusBar"/>
                  <Size>
                        <AbsDimension x="70" y="8"/>
                  </Size>
                  <Anchors>
                        <Anchor point="center" relativeTo="UIparent" relativePoint="center"/>
                  </Anchors>
                  <Scripts>
                        <OnUpdate>
                                local SVALUE=( UnitHealth("player") / UnitHealthMax("player") )*100
                                haomingzi:SetValue(SVALUE);
                        </OnUpdate>
                  </Scripts>
</StatusBar>
```

最后给头尾加上最基本的<ui></ui>嵌套。就大功告成了

```
<ui>
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <BarTexture file="Interface\TargetingFrame\UI-StatusBar"/>
                  <Size>
                        <AbsDimension x="70" y="8"/>
                  </Size>
                  <Anchors>
                        <Anchor point="center" relativeTo="UIparent" relativePoint="center"/>
                  </Anchors>
                  <Scripts>
                        <OnUpdate>
                                local SVALUE=( UnitHealth("player") / UnitHealthMax("player") )*100
                                haomingzi:SetValue(SVALUE);
                        </OnUpdate>
                  </Scripts>
</StatusBar>
</ui>
```

CODE:

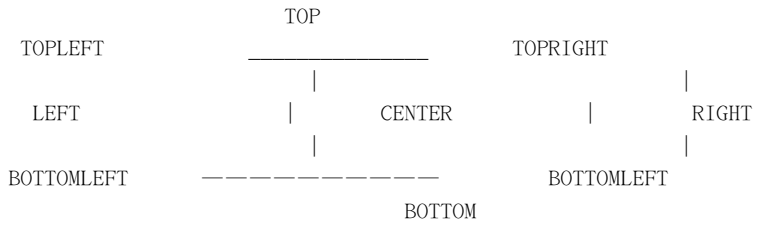
```
<Anchors>
    <Anchor point="CENTER" relativeTo="Minimap" relativePoint="CENTER">
        <Offset>
            <AbsDimension x="55" y="-55"/>
        </Offset>
    </Anchor>
</Anchors>
```

这样的代码就是用来定义的位置的

数学课和物理课都上过把？（虽然我很讨厌我们过去的数学老师 ><）
想知道任何一个物体的位置。只要有一个固定的参考物。再有与参考物体的相对坐标。就可以了

先来点基本概念

我们用一个方框来表示框体，那么



应该有点头绪了把？
之前那段代码的意思就是 把位置定义在
小地图的中心点和我们的框体的中心点 X 坐标为 55 Y 坐标为 55 的地方

如果看不明白这句话　不要紧 我们一句一句的来分析

具体分析

头一句和最后一句

```
<Anchors>
</Anchors>
```

这个嵌套是告诉 WOW：中间的代码是定义位置

第二句和倒数第二句

```
    <Anchor point="CENTER" relativeTo="Minimap" relativePoint="CENTER">
     </Anchor>
```

这个嵌套就是告诉 WOW 我们开始定义位置了。

先看第二句。

`point="CENTER"` 参考之前我画的那张很丑陋的图。
意思就是：要定义位置的框体（为了描述方便。以下简称为框体 A）的中心点（CENTER）作为定义点。
（定义点这个名词是我杜撰的。如果不明白。先接着往下看）

`relativeTo="Minimap"`
意思就是：给我们的框体 A 设置一个参考物（为了描述方便。以下把参考物简称为框体 B）
在这里。就是把框体 B 设置为小地图（Minimap）

`relativePoint="CENTER"`
同样的。也得给我们的框体 B 设置一个定义点
在这里。就设置为中心点（CENTER）

现在解释一下定义点这个我杜撰的名词
为什么要有定义点这个概念呢？
因为所有的框体都不是一个点。而是一个平面。而相对坐标只能是点与点的坐标。所以就必须在框体上找一个点来定义坐标
而这个点。就是我所谓的定义点
至于一个框体的定义点可以设置为那些，参考上面我画的那张丑陋的图

搞明白了以上的概念。那么中间的那段代码

CODE:
```
            <Offset>
                <AbsDimension x="55" y="-55"/>
            </Offset>
```
也就不难理解是什么意思了。这正是设置2个定义点之间的相对坐标

进阶

为什么一向崇尚操作简单的暴雪要把位置的定义弄的这么复杂呢？

似乎我们只要变换2个定义点之间的相对坐标。那么无论我们把定义点怎么设置 都可以达到同样的效果其实。是因为框体的大小有时候是不固定的。

比如我想实现这样的效果：

在玩家血条的左边显示HP的具体数值

如果这么定义位置：

CODE:
```
<FontString name="HPText">
<Anchors>
    <Anchor point="CENTER" relativeTo="PlayerFrame" relativePoint="CENTER">
        <Offset>
            <AbsDimension x="55" y="0"/>
        </Offset>
    </Anchor>
</Anchors>
```
。

。

。

。

似乎可以到达效果，

可是HP有多有少。

当HP为3位数的时候。数值是在血条的左边。但是但HP为4为数的时候。数值就超过了左边挡主了部分血条。

所以。得这么写

CODE:
```
<FontString name="HPText">
<Anchors>
    <Anchor point="RIGHT" relativeTo="PlayerFrame" relativePoint="LEFT">
        <Offset>
            <AbsDimension x="0" y="0"/>
        </Offset>
    </Anchor>
</Anchors>
```
。

。

。

。

这样。无论 HP 为多少。数值的右侧永远都和血条的坐侧对齐

PS：

当相对坐标为 0，0 的时候。代码可以简化

比如刚才的代码可以简化为

```
CODE: <FontString name="HPText">
<Anchors>
     <Anchor point="RIGHT" relativeTo="PlayerFrame" relativePoint="LEFT"/>
</Anchors>
```

注意：别漏看了第三句最后的那个反斜杠

脚本

看到这里。我们对框体的定义应该有了很大的了解。现在定义一个自己的框体应该没什么难度了把？

在 STATUSBAR 部分我们提到了脚本。

脚本我个人觉得是 ADDONS 的精髓

弄懂了脚本部分。那么去他的 FLEXBAR 去他的 DAB 去他的 DUF 我们不需要了。我们自己就可以来做了

基本概念

什么是脚本。通俗的说：脚本就是告诉框体在什么时候执行什么命令

同样的

我们用<Scripts></Scripts>这样的嵌套来表示代码中脚本的部分

具体的举个例子

还记得 FLEXBAR 或者 DAB 一个很实用的功能把？当鼠标进入按扭的区域 按纽显示 离开则隐藏

现在我们直接在 ADDONS 中写（为了描述方便起见 以下只写出代码中我们需要注意的部分）

```
<BUTTON name="button_1">
        <Scripts>
                <OnEnter>
                        This:Show();
                </OnEnter>
                <OnLeave>
                        This:Hide();
                </OnLeave>
        </Scripts>
</BUTTON>
```

<OnEnter>这个嵌套就是鼠标进入框体的区域需要执行的命令

<OnLeave>则是鼠标离开框体的区域需要执行的命令

如何？是不是很简单呢？

接着来。

FLEXBAR 或者 DAB 还有很多神奇的功能。比如根据条件自动改变按扭的位置 透明度 缩放 等等等

如果我们直接在 ADDONS 中编写也很方便

比如进入战斗状态 自动出现按扭 反之隐藏

```
<BUTTON name="button_1">
        <Scripts>
                <OnLoad>
                        this:RegisterEvent("PLAYER_ENTER_COMBAT");
                        this:RegisterEvent("PLAYER_LEAVE_COMBAT");
                </OnLoad>
                <OnEvent>
                        if (event == "PLAYER_ENTER_COMBAT") then
                                this:Show();
                        elseif (event == "PLAYER_LEAVE_COMBAT") then
                                this:Hide();
                        end
                </OnEvent>
        </Scripts>
</BUTTON>
```

<OnLoad>是框体被加载的时候需要执行的命令

这里。我们给 button_1 这个框体注册了 2 个事件：玩家进入战斗和玩家离开战斗

<OnEvent>是注册的事件发生的时候需要执行的命令

这里。我们用了一个判断语句。

当事件为玩家进入战斗的时候 显示按扭 1 当事件为玩家离开战斗的时候 隐藏按扭 1

同样的

脚本和框体一样 不可能仅仅只有我上面所说的那几个。

更多的脚本需要你自己去发现。。我不想罗嗦了。

更多的惊喜

以上 2 个只是很简单的例子。脚本中执行的命令还可以是相互调用。相互依存的来实现更多更复杂的功能

这时候。仅仅一个 XML 文件已经不能满足我们的需要了。我们得在 LUA 文件中来编写

LUA

如果你有耐心看完了上面的全部内容 并且亲手去实验了

那么如何编写一个 XML 文件应该了然与胸了把

当然一个精巧的 ADDONS 不可能仅仅只有 XML 文件而已。他还需要 LUA 文件

LUA 文件当然就是用 LUA 格式写的

具体的 LUA 的语法 限与帖子的篇幅，不能详尽说明。好在现在网上的资料很多的

我只说几个个人觉得很有用的部分

1。引用 LUA 和定义函数

先回头看 6 楼的最后那段代码

```
<BUTTON name="button_1">
        <Scripts>
                <OnLoad>
                        this:RegisterEvent("PLAYER_ENTER_COMBAT");
                        this:RegisterEvent("PLAYER_LEAVE_COMBAT");
                </OnLoad>
                </OnEvent>
                        if (event == "PLAYER_ENTER_COMBAT") then
                                this:Show();
                        elseif (event == "PLAYER_LEAVE_COMBAT") then
                                this:Hide();
                        end
                </OnEvent>
        </Scripts>
</BUTTON>
```

我们可以把脚本的部分放到 LUA 中来写。

首先我们新建一个 BUTTON1.LUA 文件

然后在 XML 文件里面要告诉 WOW 我们写了 BUTTON1.LUA 文件

```
<Script file="BUTTON1.lua"/>
<BUTTON name="button_1">
        <Scripts>
                <OnLoad>
                        this:RegisterEvent("PLAYER_ENTER_COMBAT");
                        this:RegisterEvent("PLAYER_LEAVE_COMBAT");
                </OnLoad>
                <OnEvent>
                        if (event == "PLAYER_ENTER_COMBAT") then
                                this:Show();
                        elseif (event == "PLAYER_LEAVE_COMBAT") then
                                this:Hide();
```

```
                        end
                </OnEvent>
        </Scripts>
</BUTTON>
```

定义函数的LUA命令是function
现在我们就把<OnLoad>和<OnEvent>这2个部分的命令定义为函数

```
function button_1_onload()
                        this:RegisterEvent("PLAYER_ENTER_COMBAT");
                        this:RegisterEvent("PLAYER_LEAVE_COMBAT");
end

function button_1_onevent(event)
                        if (event == "PLAYER_ENTER_COMBAT") then
                                this:Show();
                        elseif (event == "PLAYER_LEAVE_COMBAT") then
                                this:Hide();
                        end
end
```

然后在XML文件中引用这2个函数

```
<Script file="BUTTON1.lua"/>
<BUTTON name="button_1">
        <Scripts>
                <OnLoad>
                        button_1_onload();
                </OnLoad>
                <OnEvent>
                        button_1_onevent(event);
                </OnEvent>
        </Scripts>
</BUTTON>
```

这样原来的一个XML文件就被我们分成了2个文件LUA和XML
也许就上面的那段简单的代码 我们还觉得这样做并没有什么太大的意义
不过。当你写了一段很复杂 很麻烦的代码的时候。这么做显然有助与你简化代码和理清思路

2。代码的本地化

因为WOW有很多国家的版本。所以一些变量的设置需要本地的语言。
比如能在中国使用的ADDONS，有时候并不能在美国使用。这时候我们就需要做一些本地化的工作

怎么做？
我们把所有的需要使用当地语言的变量集中起来 在一个LUA文件中定义
（这个LUA文件。我们一般起名叫：localization.lua）

而且。WOW 还提供了自动判断语种的功能
这些都很简单。随便打开一个 ADDONS 的 localization.lua 自己看一看就明白　不罗嗦了

当然 LUA 的作用远远不止这些
毕竟 LUA 是一个很成熟的语言。熟练的运用将大大简化我们的工作量
比如 LUA 的数组功能。字符串的判断
更多的细节。可以在自己动手写 ADDONS 的过程中慢慢摸索。

继承

WOW 已经帮我定义好了很多有用的框体
所以很多的时候。我们并不需要自己完全的重新定义
直接引用他们就可以了
这里就要用到继承这个概念

如何做？
继承的命令是 inherits

比如我想定义一个文字框体。他的样子和显示玩家的名字的文字的样子是一样的
那么：

```
<FontString name="TEXT_FRAME" inherits="GameFontNormalSmall" >
<FontString/>
```

这样 我们简单的用了 inherits="GameFontNormalSmall"命令
就把 TEXT_FRAME 框体的大小 颜色 透明度 字体等等等等属性全部搞定了

如果有不满意的地方 还可以重新定义。
比如改变一下颜色

```
<FontString name="TEXT_FRAME" inherits="GameFontNormalSmall">
        <Color r="0" g="1" b="0"/>
<FontString/>
```

现在他就是绿色的咯

当然我们也可以定义自己的
这将大大有助于简化我们的代码。
还记得我以前写的那个 OPENDOOR 吗？
我在里面一共定义了 7 个框体
其中有 6 个框体是按扭。并且他们很多部分都是公共的。

所以。如果我现在再来写那个 OPENDOOR 我会先写一个父框 把 6 个按扭公共的部分全部写进去
然后在一个一个的继承就 OK 了。~

具体的父框的定义 不罗嗦了
大家可以随便打开一个 ADDONS 找到名字后面为 Template 的框体。那多半就是父框了。
动手分析一下把
（提示一点：在父框中的$parent 就是要被继承的子框的名字）

进阶

StatusBar 除了 SetValue 这个重要的命令以外。还有个 SetStatusBarColor 命令。是用来改变颜色的

比如还是上面的例子
我们现在想当 HP 超过 50%的时候为绿色 低于 50%的时候为红色
那么先定义一个函数 就叫 haomingzi_OnUpdate 把 用来实现上面的功能

```
function haomingzi_OnUpdate()
local SVALUE=( UnitHealth("player") / UnitHealthMax("player") )*100;
   if SVALUE > 50 then
       haomingzi:SetStatusBarColor(1,0,0);
   end
end
```

然后我们在脚本中调用这个函数就可以了

```
<ui>
<StatusBar name="haomingzi" minValue="0" maxValue="100">
                <BarTexture file="Interface\TargetingFrame\UI-StatusBar"/>
                 <Size>
                        <AbsDimension x="70" y="8"/>
                 </Size>
                 <Anchors>
                        <Anchor point="center" relativeTo="UIparent" relativePoint="center"/>
                 </Anchors>
                 <Scripts>
                        <OnUpdate>
                                haomingzi_OnUpdate() ;
                        </OnUpdate>
                 </Scripts>
</StatusBar>
</ui>
```

更多的框体

当然框体绝对不仅仅只有<StatusBar>这一中。还有诸如<Button> <Frame> <Texture> <FontString>等等等等
这里就不一一解释了，你随便打开一个写好的 ADDONS 都可以发现他们的身影。自己分析一下把

这里将说说 如何定义一个框体的位置

我们打开任意的一个编写好的 ADDONS 的 XML 文件。多半会发现形如这样的代码