

[Dragon Age][Toolset]在單人遊戲使用自製的遊戲道具

來源 [BIOWARE 社群的 Custom Player Items In Single Player](http://social.bioware.com/5339/blog/576/)

<http://social.bioware.com/5339/blog/576/>

作者 [weriKK](#)

翻譯:台灣奇幻遊戲社區網的騎士 Yu

<http://www.pcf.it.idv.tw/nwn2/>

經由一步一步的教學，我將會教你建立兩個道具：長劍和盾，並導入到你的角色包包裡。完成品就如同 Bonus DLC 的 package。要完成此教學你必須安裝 Dragon Age：Origins 和 Dragon Age Toolset。並且要將你系統裡的非 Unicode 語言改成 English。完成此教學，你就是一個模組新人。

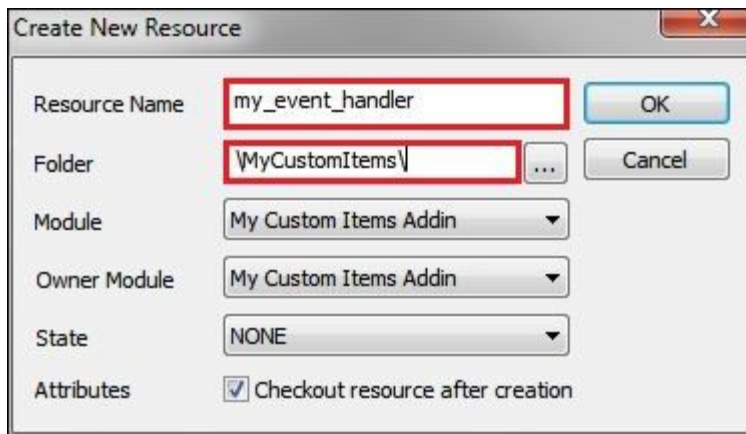
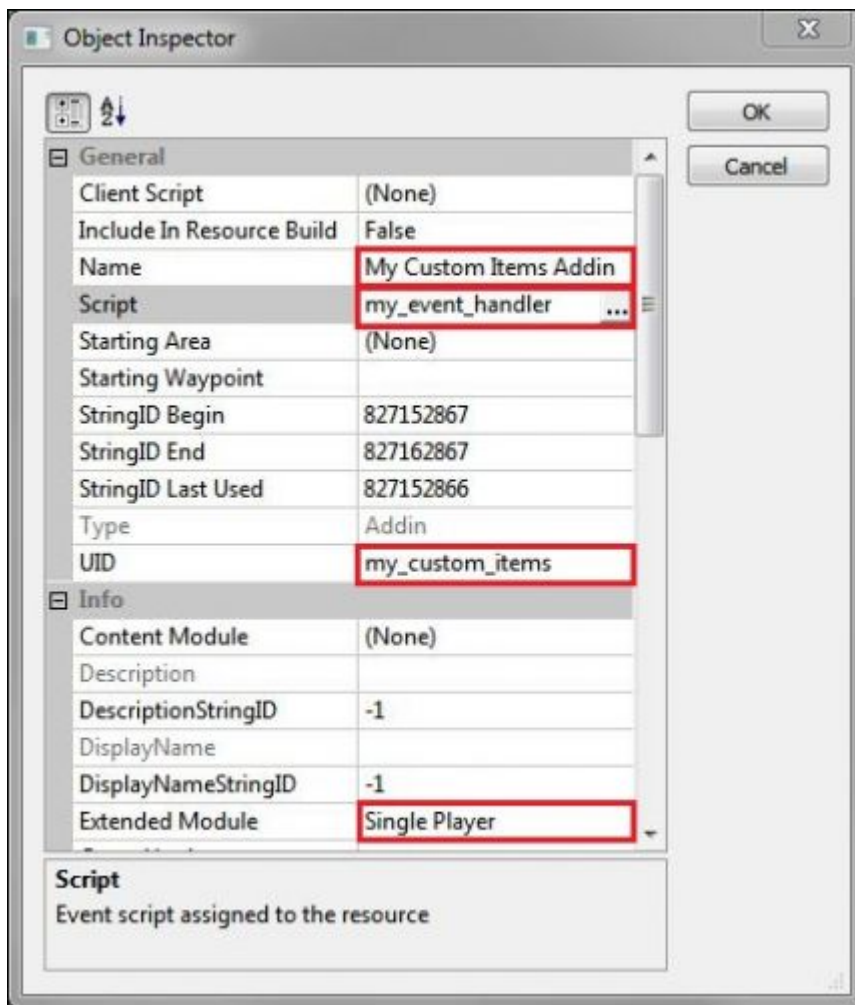
原始碼、封裝成包、最終成果都放在最後面。

這是一個開始，讓我們進入深深的水裡學習。

建立一個新模組

首先我們要在 **Toolset** 裡建立一個新模組，我們可以在以後的遊戲加載：

- 點擊 **File -> Manage Modules**，然後點擊 **New** 就可以看到 Object Inspector 出現
- 在 **Name** 區域，為你的模組命名：**"My Custom Items Addin "** (在此使用 **My Custom Items Addin** 作為我的模組名稱)，它將會顯示在 DLC 列表。
- 在 **UID** 區，為你的模組定義識別字，以我為例是使用 **"my_custom_items "** 作為識別字。它會以此關鍵字建立目錄並將插件導入。
- 往下拉會看到 **Extended Module**，在此選擇 **Single Player**，因為我們要添加額外的內容到單人遊戲裡（即遊戲道具要放到單人遊戲裡使用）
- 點擊 **OK** 然後關閉 **Manage Modules**。
- 開啓模組。
- 返回 **Manage Modules** 視窗。先點新建立的模組然後點擊 **Hierarchy**，勾選 **Single Player**。然後按 **OK**。
- 返回到模組的 **Properties**。
- 找到 **Script** 欄位，點擊 ... 等待 **Resource Open/Save** 的對話出現。
- 點擊選項 **New** 並且在 **Resource Name** 欄位輸入你的 **Script**（腳本）名稱，它是用來處理遊戲中你要使用的所有事件。在此我稱呼此腳本為 **"my_event_handler"**。
- 在 **Folder** 欄位，我們可以自訂我的模組位於系統的哪個區域，以我為例我是以 Windows 系統的根目錄，用 **"\MyCustomItems\"** 作為模組的使用區域然後點選 **OK**。
- 再次開啓模組。



在第三步定義識別字是很重要的，你定義的識別字必須其他模組都沒有使用過。識別字是 **Toolset** 用來指向你的模組。

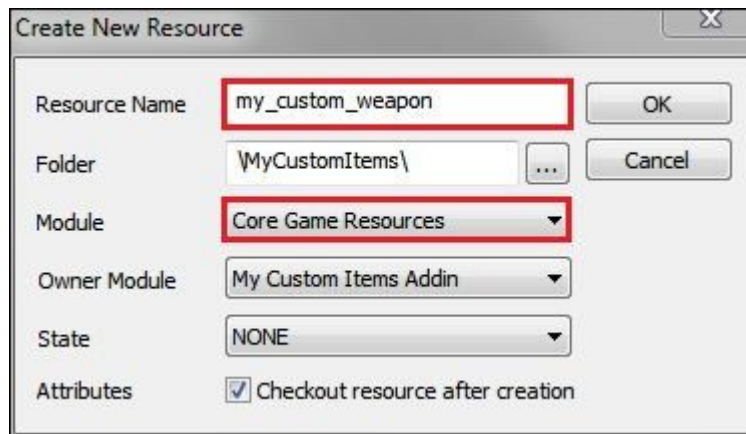
建立你的事件處理腳本是極為重要的事。忽略此步驟會在遊戲裡發生重大錯誤，而且也會妨礙其他模組的運作。唯一的例外就是你不使用任何的事件腳本，此時你要自列表中點選 (*None*)。

在模組裡建立自訂的道具

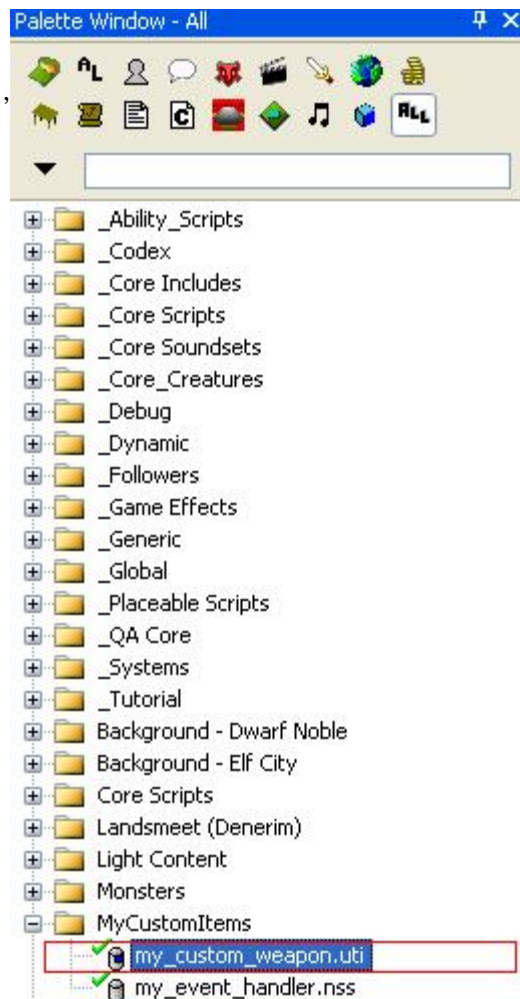
一旦建立了我們的模組，我們就可以開始製作物品。對於很多物件，Toolset 提供了樣板，故製作物品直接利用樣板製作。使用樣板製作物品只要修改樣板的屬性就能成為我們獨一無二的物品。

要利用樣板建立物品，我們必須先開啓 **Palette** 視窗。如果你找不到，就點選 **View -> Palette**。

- 點擊 **Palette** 視窗頂端的 **"ALL"** 圖標。會提供所有可以使用的東西。
- 找到你先前建立的 **MyCustomItems** 目錄。對它按滑鼠右鍵並選擇 **New -> Item**。Create New Resource 視窗會彈出來。
- 在 **Resource Name** 欄位，為你的物品輸入識別字。我使用 **"my_custom_weapon"**。
- 因為我們要讓物品可以在遊戲裡使用，所以我們必須確保資源是核心資源（core game resources）的一部分。；按 **Module** 下移選單，選 **Core Game Resources**。
- 在 **Owner Modul** 欄位必須是寫我們的模組名稱，以我為例為：**My Custom Items Addin**。
- 完成如下圖，按 **OK**，關閉視窗並等待 toolset 的訊息。



如果你是遵從此教學，
現在你會看到我們新建物品的 properties 視窗，
如果你找不到 properties 視窗，
就點選右圖的紅色方塊兩次。



讓我們利用表單裡重要的欄位建立一把不錯的長劍

- **Base Item Type:** 選擇物品的類型。我們要長劍，所以在下移選單裡選擇 **Weapon - Longsword**。
- **Description:** 此欄位是用來描述遊戲裡物品的說明，以我為例 "Mighty Longsword of my right hand!"
- **Icon:** 點 ... 開啓檔案，開啓對話並看圖標，選擇你想要的，在此我使用 **ico_longsword.dds** 作為遊戲的小圖標。(所有圖標都是以 **ico_** 開頭！)。
- **Inventory Subgroup:** 長劍使用 **205000**。此欄位是指明我們的物品是道具列表的第幾種。所有的長劍都是 **205000**。(譯者：查詢物品得的 subgroup <http://social.bioware.com/wiki/dataooset/index.php/Inventory>)
- **Item Variation:** 這是用來選遊戲裡物品實際的模型，在此我使用 **Longsword 5**，你可以選你喜歡的。
- **MaterialProgression & Material Type:** 這是用來選擇物品的材質與類型。首先我不知道 **MaterialProgression** 是幹麼的，而 **Material Type** 會影響外型 and 此物品實際能力，就以武器來說會影響傷害、重擊機率等等，在此我兩格都選擇 **Weapon, Dragonbone**。
- **Name:** 遊戲裡顯示得物品名稱，如 **"Unstoppable Force"**。
- **Tag:** 此欄位的內容是用來指向我們的物品。在此我使用先前所定義的資源名稱 (Resource Name) : **"my_custom_weapon"**。
- **Cost:** 此物品的價格，以我為例是 100 銀幣。
- **Item Properties:** 為你的物品選擇屬性如增加傷害、降低力量等等。
- **OnHit Effect & OnHit Power:** 如同項目所述，是攻擊時的額外效果 **Effect** 是選擇對龍或是對精靈等的額外傷害或是火焰等的特殊攻擊，**Power** 則是傷害值。在此我使用 +10 extra damage vs. darkspawn.

如果你想要更為瞭解 **Material Type**，以及使用哪種類型所帶來的效果，或是計算使用混合材質的最終屬性可以參考此：[The Secret Behind Item Statistics](http://social.bioware.com/5339/blog/651/) <http://social.bioware.com/5339/blog/651/>

如果你想瞭解，還有一個簡單方法就是開啓 *Palette* 視窗然後點選 *View the resources*。

Activated Ability	(None)
Base Item Type	Weapon - Longsword
Comments	
Crafting Recipe Type	Mabari Crunch
Description	Mighty Longsword of my right hand
DescriptionRequiresReTranslation	True
Heraldry	(None)
Icon	 ico_longsword.dds
Inventory Subgroup	205000
Item Variation	Longsword 5
MaterialProgression	Weapon, Dragonbone
MaterialType	Weapon, Dragonbone
Name	Unstoppable Force
NameRequiresReTranslation	True
Resource Name	my_custom_weapon
Tag	my_custom_weapon

好了！我們已經完成自製的劍。我們不用在修改屬性所以關掉它。在 **palette** 視窗，對你的物品按滑鼠右鍵並選擇 **check in**。此時物品會載入至資料庫。

現在依照之前得教學製作一個盾：

- **Resource Name:** "my_custom_shield"
- **Base Item Type:** Shield - Large Round
- **Icon:** ico_shield_largeroundmetal.dds
- **Inventory Subgroup:** 342000
- **Item Variation:** Large Darkspawn Shield
- **MaterialProgression:** Shield, Dragonbone
- **MaterialType:** Shield, Dragonbone
- **Name:** "Immovable Object"
- **Tag:** "my_custom_shield"

一旦完成盾後，一樣點擊 **check in**。

Activated Ability	(None)
Base Item Type	Shield - Large Round
Comments	
Crafting Recipe Type	Mabari Crunch
Description	Mighty Shield of my left hand!
DescriptionRequiresReTranslation	True
Heraldry	(None)
Icon	 ico_shield_largeroundmetal.dds
Inventory Subgroup	342000
Item Variation	Large Darkspawn Shield
MaterialProgression	Shield, Dragonbone
MaterialType	Shield, Dragonbone
Name	Immovable Object
NameRequiresReTranslation	True
Resource Name	my_custom_shield
Tag	my_custom_shield

將物品添加到角色背包裡

接下來是困難的部份。我們在模組裡有自製物品，但是遊戲裡不會出現，我們必須溝通遊戲讓遊戲裡的單人遊戲出現自製物品。而 **Script**（腳本）就是負責溝通

先前在我們編輯模組屬性時已經建立事件處理腳本(event handler script)。現在該是寫腳本的時候。在 **Palette** 視窗，找尋我們的腳本：**my_event_handler.nss**。對它連點兩次就會開啓。如果你看到腳本編輯畫面的頂端是紅條，就對腳本按滑鼠右鍵選擇 **check out**。現在就讓我們開始編輯巴。

Dragon Age 的 Toolset 使用的腳本語言很像 C 語言，所以對於新手來說有點難度。此份教學是點到為止，但會提供給各位一個方便製作自製物的腳本。

在我們開始教導腳本之前，先瞭解 (`///`)，此符號表示註解，有 `//` 開頭的那一列都不會視為腳本。我會利用註釋廣泛的講解腳本。

每一次被呼叫的腳本，都能看到 **main() function** 在開頭。不用去計較有多少個腳本檔案，其執行的第一個函數 (**function**) 總是 **main function** 。正因如此，我們也從這裡開始：

```
void main()  
{  
}
```

Dragon Age: Origins 有事件驅動腳本系統(event driven scripting system)。每當遊戲裡的模組要發生事件，遊戲會發送通知，當模組收到事件訊息就會呼叫他們的事件處理腳本處理訊息。邏輯上來說，第一件我們要學習的就是檢索此訊息。

```
void main()  
{  
    event ev = GetCurrentEvent();  
    int nEventType = GetEventType(ev);  
}
```

一旦我們成功地檢索到事件，我們必須檢查此事件是否為我們要尋找的。我們要讓我們的英雄角色只有在模組第一次讀取時才會拿到物品，所以我們要尋找一個事件叫做 **EVENT_TYPE_MODULE_LOAD**。

```
void main()  
{  
    event ev = GetCurrentEvent();  
    int nEventType = GetEventType(ev);  
  
    // 我們會關注每一個事件的類型，  
    // 一旦出現我們需要的事件就會視為特殊案子來處理，  
    // 並忽略其他事件。  
    switch ( nEventType )  
    {  
        // 此事件是只要每次模組讀取就會出現。  
        // 它通常發生在開始遊戲或是讀取遊戲的紀錄檔。  
        case EVENT_TYPE_MODULE_LOAD:  
        {  
  
            // 我們要處理正在等待的事件，  
            // 在此時我們要停止尋找其他事件。  
            break;  
        }  
        default:
```

```

        break;
    }
}

```

現在我們可以為我們的腳本建立模組讀取事件，但它還不會作任何有趣的事。
現在我們要告訴腳本，當模組讀取時就要在角色背包裡添加我們的物品。

這是一個特殊的 **utility function**，會添加資源以及其數量到玩家的背包裡。此函數叫 **UT_AddItemToInventory**，而且它只有兩個參數：資源要添加到背包裡和要添加的數量。

但在我們使用此函數前，我們要讓 **Toolset** 知道到哪找此函數。要使 **Toolset** 知道此函數是很簡單的，此函數是位於一個名叫 "utility_h" 的腳本檔案裡。故我們將 載入資訊放在 **main** 函數前，這樣就能使用 **UT_AddItemToInventory**。

```

#include "utility_h" //載入腳本utility_h

void main()
{
    ...
}

```

我們可以延伸我們的腳本了

```

// ---- 腳本開始 ----

#include "utility_h"

void main()
{
    event ev = GetCurrentEvent();
    int nEventType = GetEventType(ev);

    // 我們會關注每一個事件的類型，
    // 一旦出現我們需要的事件就會視為特殊案子來處理，
    // 並忽略其他事件。
    switch ( nEventType )
    {
        // 此事件是只要每次模組讀取就會出現。
        // 它通常發生在開始遊戲或是讀取遊戲的紀錄檔。
        case EVENT_TYPE_MODULE_LOAD:
        {
            // UT_AddItemToInventory 函數會添加多樣的資源到
            // 生物的背包裡。在此我們添加武器和盾牌。
            UT_AddItemToInventory(R"my_custom_weapon.uti", 1);
            UT_AddItemToInventory(R"my_custom_shield.uti", 1);

            // 我們要處理正在等待的事件，

```

```

        // 在此時我們要停止尋找其他事件。
        break;
    }
    default:
        break;
}
}
// ---- 腳本結束 ----

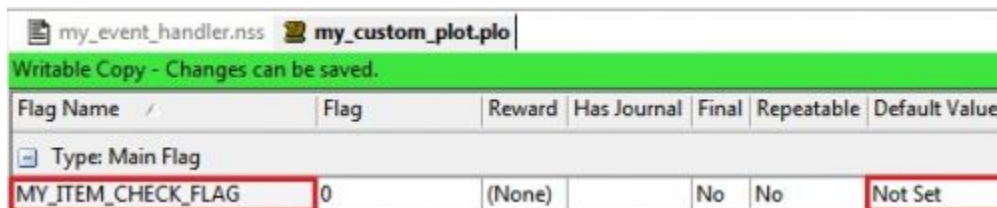
```

還不壞。但仍然有問題存在。儘管此腳本會正確地給予角色物品，但只要每次讀取遊戲的紀錄檔就會在執行一次，使角色又拿到重複物品。只要一段時間，我們就會獲得大量的相同裝備。

我們要如何避免這一點？我們必須找到方法來追蹤角色是已經拿到物品還是尚未拿到。有很多方法，在此採用 **Plot flag**，因為這樣比較簡單，也使你的大腦能夠記住腳本不至於遺忘。

我們稍後再來處理腳本，現在先存檔，將注意力集中到 **Palette** 視窗，如同以前所講，對 **MyCustomItems** 按滑鼠右鍵並選 **new ->plot**

- **Resource Name** 欄位輸入 "my_custom_plot" 然後點擊 **OK**。
- 當 **plot editor** 開好後，對任意一處按滑鼠右鍵選 **Insert -> Main Flag**。
- 對 **flag** 的名字點選兩下 (也許他是顯示 **FLAG_0** 或是其他類似的) 將它更名為 **MY_ITEM_CHECK_FLAG**。
- 確保 **Default Value** 的行是寫 **Not Set**。
- 保存檔案並且將它 **Check in**。完成 **plot** 了。



返回我們的腳本檔案！在我們放入新製作的 **plot flag** 前，我們必須要讓 **Toolset** 知道如何找到它。我們要添加額外的行在 **main** 函數前：

```

#include "utility_h"
#include "plt_my_custom_plot" //注意"plt_"

void main()
{
    ...
}

```

注意必須在 **plot** 檔案名稱前加入 **"plt_"**！它是非常重要的。它會告知 **Toolset** 這是 **plot** 而不是其他的腳本檔案。

有兩件事我們必須要放在腳本裡面，設定 **plot flag** 何時給角色物品以及檢查物品是否以給。

事不宜遲，下面是完成的腳本： 注意 **toolset** 不能保存中文 故請輸入最後一面的英文版


```

// ---- 腳本開始----

// 在此腳本，我們會用到 UT_AddItemToInventory() 函數

// 來添加物品到角色的背包裡，
// 但在使用此函數前，我們必須告知 Toolset 去哪找。
// 此函數是在 "utility_h" 腳本檔案裡
// 根據 _Core 引導，我們必須在 main 函數前將 utility_h 腳本檔案引入

#include "utility_h"
#include "plt_my_custom_plot"

void main()
{
    // 如果 plot flag 設為 TRUE，
    // 就表示已經將物品給予玩家，
    // 所以不需要運行此腳本。
    if ( WR_GetPlotFlag( PLT_MY_CUSTOM_PLOT, MY_ITEM_CHECK_FLAG ) ==
TRUE )
        return;

    event ev = GetCurrentEvent();
    int nEventType = GetEventType(ev);

    // 我們會關注每一個事件的類型，
    // 一旦出現我們需要的事件就會視為特殊案子來處理，
    // 並忽略其他事件。
    switch ( nEventType )
    {
        // 此事件是只要每次模組讀取就會出現。
        // 它通常發生在開始遊戲或是讀取遊戲的紀錄檔。
        case EVENT_TYPE_MODULE_LOAD:
        {
            // UT_AddItemToInventory 函數會添加多樣的資源到
            // 生物的背包裡。在此我們添加武器和盾牌。
            UT_AddItemToInventory(R"my_custom_weapon.uti", 1);
            UT_AddItemToInventory(R"my_custom_shield.uti", 1);

            // 設定 plot flag 為 TRUE，則下一次腳本運作時
            // 就不會在放多的物品到角色背包裡。
            WR_SetPlotFlag( PLT_MY_CUSTOM_PLOT, MY_ITEM_CHECK_FLAG,
TRUE );

            // 我們要處理正在等待的事件，

            // 在此時我們要停止尋找其他事件。
            break;
        }
    }
}

```

```

        default:
            break;
    }
}
// ---- 到此腳本節束 ----

```

腳本完成後就按滑鼠右鍵選擇 **check in**。我們終於完成。剩下一件事就是輸出我們的模組到 Dragon Age: Origins。

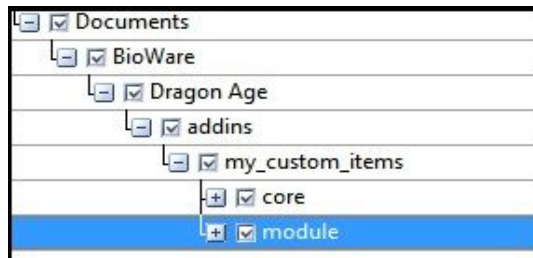
- 找到 **MyCustomItems** 資料夾，裡面有所有你的檔案。對 **MyCustomItems** 按滑鼠右鍵。
- 在彈出的視窗選 **Export->Export without dependent resources**。會使用此的原因是我們需要的資源已經在核心封包（**core package**）裡了。
- 反覆上一步，**Generate Module XML & Generate Manifest XML**。
- 不幸的是，因為 Toolset 有個 bug，必須作一道步驟來處理。儘管我們明確地表示輸出不需要依賴資源，但顯然地 Toolset 仍然會輸出一些不需要的檔案。如果這些多餘的檔案不處理，它們會使遊戲嚴重損害。所以要刪除這些檔案，到 "**MyDocuments\Bioware\Dragon Age\packages\core\override\toolsetexport**" 並刪除此目錄裡的所有單一檔案。

請確保你在正確的目錄！

此時，如果你啟動 Dragon Age: Origins，你會看到你的模組出現在 downloadable content 區域，一旦開始遊戲，你的角色背包裡會出現你製作的長劍和盾牌。

如果你要分享你的模組給其他人，就必須封裝它。簡單方法如下：

- 在工具列上點擊 **Tools->Builder->Builder To Player Package**。它會詢問你的 manifest 檔案在哪。以我為例，它位於：My Documents\BioWare\Dragon Age\AddIns\my_custom_items\module\override
- 之後會出現一個巨大的樹狀圖。你可以將 **packages** 上的勾去掉。我們只需要下方的 my_custom_items。
- 點擊 OK 並且保存你的第一個 Dragon Age: Origins 封包（package）檔案。
- 喝一杯慶祝巴！



你可以在此下載[完成品](http://social.bioware.com/project/422/) 並研究它。

<http://social.bioware.com/project/422/>

請使用此討論區 談論任何關於製作插件的問題，不要在部落格談論，這樣不好管理。

<http://social.bioware.com/project/422/&v=discussions>

下圖是模組實際的範例圖片。



以下是原文的腳本原始碼 因為 **toolset** 不能保存中文 所以請改成英文

```
// ---- SCRIPT STARTS HERE ----

// Later on in this script, we will use the UT_AddItemToInventory()
// function to add an item to the player's inventory.
// But before we could use it, we have to tell the toolset where
// to look for it. The function is in the "utility_h" script file
// under _Core Includes, we will include this file at the top of
// our script file, above the main function.

#include "utility_h"
#include "plt_my_custom_plot"

void main()
{
    // If our plot flag is set to TRUE, that means we have already
    // given the items to the player, there is no need to continue
    // running this script.
    if ( WR_GetPlotFlag( PLT_MY_CUSTOM_PLOT, MY_ITEM_CHECK_FLAG ) ==
TRUE )
        return;

    event ev = GetCurrentEvent();
    int nEventType = GetEventType(ev);

    // We will watch for every event type and if the one we need
```

```

// appears we will handle it as a special case.
// We will ignore the rest of the events
switch ( nEventType )
{
    // This event happens every time the module loads
    // This usually happens when creating a new game
    // or loading a savegame
    case EVENT_TYPE_MODULE_LOAD:
    {
        // The UT_AddItemToInventory function
        // adds various resources to a
        // creature's inventory.
        //Here we add one weapon and one shield.
        UT_AddItemToInventory(R"my_custom_weapon.uti", 1);
        UT_AddItemToInventory(R"my_custom_shield.uti", 1);

        // Set our plot flag to TRUE,
        //so the next time this script tries
        // to run it will not add extra items
        //to the player's inventory
        WR_SetPlotFlag( PLT_MY_CUSTOM_PLOT, MY_ITEM_CHECK_FLAG,
TRUE );

        // We have dealt with the event
        // we were waiting for.
        // At this point we can stop
        //looking for other events
        break;
    }
    default:
        break;
}
}
// ---- SCRIPT ENDS HERE ----

```