Defines

Table of Contents

Overview

<Define> tags allow skins to define a value to be used in multiple places in a skin xml (or in any xml imported or included by a window xml).

Defines

<Define> tags normally appear at the top of a skin xml for easy reference. They allow a basic form of parameter to be used so that imported files can be as generic as possible. A <define> tag is a simple name:value pairing.

- <define> tags can be used in any xml file that has a window ID.
- <define> tags are entered in the window elements control at the top of the xml, before the <controls> tag.
- The name used in the <define> tag must begin with the hash character "#", and the name is case sensitive ("#abc" and "#ABC" are two different names).
- Global <define> tags can also be used in the file "references.xml" to define global values.

Sample

The following is an example of how this feature can be used:

```
<window>
  <!-- include the common window features here -->
  <define>#header.label:134</define>
  <define>#header.image:videos_logo.png</define>
  <define>#header.hover:hover_my videos.png</define>
  <controls>
    <!-- include the header logo, text and other common elements -->
    <import>common.window.xml</import>
    <control>
    ...
```

The value 134 in the above example references a text string in the corresponding language file. For English, this would be the file "strings_en.xml" in the *la nguage* folder in the MediaPortal application folder.

Any occurrence of the token represented by the <define> tag's name will be replaced with the appropriate value in any imported markup. The following is an example of how an imported file might look:

```
<window>
  <control>
    <type>image</type>
    <id>1</id>
    <posX>60</posX>
    <posY>20</posY>
    <texture>#header.image</texture>
  </control>
  <control>
    <type>label</type>
    <id>1</id>
    <posX>250</posX>
    <posY>70</posY>
    <label>#header.label</label>
    <font>font16</font>
    <textcolor>ffffffff</textcolor>
  </control>
  ...
```

Enhanced Defines

With effect from MediaPortal version 1.3.0, <define> tags can be promoted to become skin properties, to allow them to contain skin expressions, and be added to the window definition through the use of the <include> tag.

By promoting <define> tags so that they are tracked inside MediaPortal as properties, skin designers can use defines to transfer information from one page to another, and to use those values in skin expressions and functions. This is very useful for skin capabilities that provide for user controlled layout changes on a "skin settings" page. Enabling this behavior significantly reduces the need for skin developers to develop and distribute an MP plugin with their skin (where this decision logic has historically resided). An example would be modifying the EPG length or home page layout and content. The following points should be noted:

- **Multiple <define> tags with the same name** - As skin xml files are loaded, the <define> tags in those windows are loaded and saved as properties. Property values have global scope and are available to be used on any other MediaPortal window. If a window loads and defines a property "#A", and a subsequent window is loaded that also defines "#A", the second value of "#A" overwrites the first value. This implementation was chosen in preference to ignoring the second definition of "#A" in the expectation that the window information loaded most recently is more likely to be correct than window information loaded in the past.
- **<define> tags in conflict with built-in property names** - If a window is loaded with a <define> that redefines a built-in property name, the <define> in the skin xml overwrites the current value of the built-in property. However, this should be used with caution, as the skin designer does not have any control over when the built-in property value is written.
- **Property/<define> namespaces** - Because <define> tags can be managed as properties, it makes good sense for skins to maintain <define> tags using a namespace that will not be in conflict with built-in properties. This is easily done by adopting a simple naming convention for the names used in <define> tags. The recommendation is to preface all names in <define> tags with the word "#Skin". This convention is safe because MediaPortal does not maintain any properties that begin with "#Skin". For example, use the name "#Skin.tvguide.rows.is_short" rather then simply "#tvguide.rows.is_short". Note that names are *case sensitive*, so "#skin" and "#Skin" represent two different values.

Defines can be added to a skin xml file using the <include> tag. This behavior allows for the skin designer to create skin files that contain a common set of values that may be used throughout the skin. For example, you can create a skin file named "colors.xml" with the content shown below. Each skin xml file that includes "colors.xml" (by using: <include>colors.xml</include>) is able to use the <define> tags that it contains.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<window>
  <define>#red:ffff0000</define>
  <define>#green:ff00ff00</define>
  <define>#blue:ff0000ff</define>
</window>
```

Syntax and use

The syntax for a <define> supports the following attributes:

- **property** - a boolean value; either "true" or "false". If "true", the definition is promoted to become a MediaPortal property, and can be used just as any other property is used (in any MediaPortal window).
- **evaluateNow** - a boolean value; either "true" or "false. This specifies whether the definition (which would normally be a skin expression in this case) should be evaluated when it is loaded ("true"), or evaluated later by its consumer ("false"). The consumer must provide expression evaluation if the value is "false".

In the following, at the moment the definition is read in (on window load), the definition is promoted to be become a MediaPortal property (via GUIPropertyManager.SetProperty()). Expressions in the definition's value are not evaluated when the window loads; they are evaluated later, when the control that consumes the define/property is loaded.

```
<define property="true">#name:value</define>
```

In the following, at the moment the definition is read in (on window load), the definition is evaluated and any expressions are resolved immediately (via GUIPropertyManager.Parse()). The definition is not promoted to become a MediaPortal property.

```
<define evaluateNow="true">#name:value</define>
```

In the following, the expression is evaluated on window load and the result is promoted to become a MediaPortal property.

```
<define property="true" evaluateNow="true">#name:value</define>
```

The following alternatives implement the legacy behavior of a <define> tag, and are all functionally equivalent.

```
<define>#name:value</define>
<define property="false">#name:value</define>
<define evaluateNow="false">#name:value</define>
<define property="false" evaluateNow="false">#name:value</define>
```

Example

This example shows how you can modify the number of TV EPG rows via a GUI user setting using <define> tags and skin settings. In a settings xml skin file, the following user control is defined to control the length of the EPG (short or long). The <selected> and <onclick> tags manage a skin setting that determines whether the EPG length should be short or long (using the skin setting "#Skin.tvguide.rows.is_short"). When the checkbutton control is clicked, the boolean setting is toggled. Using the skin setting capability provides persistence for the setting (the setting is written to a skin configuration file).

```
<control>
  <description>Guide Rows</description>
  <type>checkbutton</type>
  <id>14</id>
  <label>Display Less Rows In Guide</label>
  <font>font12</font>
  <width>471</width>
  <height>52</height>
  <textXOff>28</textXOff>
  <textYOff>8</textYOff>
  <markalign>right</markalign>
  <markvalign>top</markvalign>
  <markWidth>33</markWidth>
  <markHeight>21</markHeight>
  <markXOff>30</markXOff>
  <markYOff>10</markYOff>
  <textureFocus>settings_button_focus.png</textureFocus>
  <textureNoFocus>settings_button_nofocus.png</textureNoFocus>
  <textureCheckmark>settings_checkmark_checked.png</textureCheckmark>
  <textureCheckmarkNoFocus>settings_checkmark_unchecked.png</textureCheckmarkNoFocus>
  <textcolor>ffffffff</textcolor>
  <textcolorNoFocus>ffffffff</textcolorNoFocus>
  <onleft>#defaultcontrol.onleft</onleft>
  <selected>skin.hassetting(#Skin.tvguide.rows.is_short)</selected>
  <onclick>skin.togglesetting(#Skin.tvguide.rows.is_short)</onclick>
</control>
```

Rendering with the setting

In the skin xml file that renders the EPG, the following <define> tags are used to configure the individual controls that make up the EPG. Notice that each <define> contains as its value a skin expression that is calculated when the <define> is read in by the skin engine. Since each <define> is resolved completely at window load time, and in the order in which it appears in the skin xml file, it is possible to build relationships between <define> tags to lessen the complexity of a compound expression.

In this example each <define> consumes the value of the skin setting that is managed via the checkbutton control described above ("#Skin.tvguide.rows.is_short"). Since "#Skin.tvguide.rows.is_short" is a boolean, it may be used in the skin expression *iif(condition, true part, false part)* as the *condition*. In each case the appropriate value is selected (either the *true part* or *false part*) and assigned to the <define> name at the moment the <define> is read in by the skin engine (during window load).

```
<define>#Skin.tvguide.viewport.height:#(iif(#Skin.tvguide.rows.is_short, 101, 789))</define>
<define>#Skin.tvguide.viewport.texture:#(iif(#Skin.tvguide.rows.is_short, 'tvguide_timeline_bg.
png', 'viewport_tv_guide_10rows.png'))</define>
<define>#Skin.tvguide.panel.height:#(iif(#Skin.tvguide.rows.is_short, 566, 666))</define>
<define>#Skin.tvguide.channel_template.height:#(iif(#Skin.tvguide.rows.is_short, 64, 66))<
/define>
<define>#Skin.tvguide.current_program_title.posy:#(iif(#Skin.tvguide.rows.is_short, 816, 950))<
/define>
<define>#Skin.tvguide.current_program_time.posy:#(iif(#Skin.tvguide.rows.is_short, 868, 1006))<
/define>
<define>#Skin.tvguide.description1.visible:#(iif(#Skin.tvguide.rows.is_short, 'yes', 'no'))<
/define>
<define>#Skin.tvguide.horizontal_scrollbar.posy:#(iif(#Skin.tvguide.rows.is_short, 905, 505))<
/define>
<define>#Skin.tvguide.vertical_scrollbar.posx:#(iif(#Skin.tvguide.rows.is_short, 1829, 1329))<
/define>
<define>#Skin.tvguide.tv_group_button.height:#(iif(#Skin.tvguide.rows.is_short, 533, 665))<
/define>
<define>#Skin.tvguide.tv_group_label.posy:#(iif(#Skin.tvguide.rows.is_short, 733, 860))<
/define>
```

In the body of the skin xml that renders the EPG, the <define> tags above are consumed by GUI controls. Simply use the appropriate <define> in place of a hard-coded value. Example:

```
<control>
  <description>VIEWPORT</description>
  <type>image</type>
  <id>0</id>
  <posX>0</posX>
  <posY>162</posY>
  <width>1893</width>
  <height>#Skin.tvguide.viewport.height</height>
  <texture>#Skin.tvguide.viewport.texture</texture>
  <shouldCache>true</shouldCache>
</control>
```

See also: Skin Expressions

Related

- Skin Architecture