Menubutton

Table of Contents

Description

The MenuButton is an extremely versatile control and offers a variety of options for usability and skinning, which includes **either**:

- An embedded spin control to select from a list of menu choices, **or**
- A modal dialog to select from a list of menu choices

The implementation of MenuButton consumes both SpinControl and DialogMenu. There is a lot of flexibility in the presentation of the button text and currently selected menu value. The button label may be static text, may include static text and the currently selected menu value, or just the value itself. This capability allows us to use the MenuButton in place of existing controls that do things like select facade views (of course the skin designer may choose to preserve the current/legacy behavior or do something new).

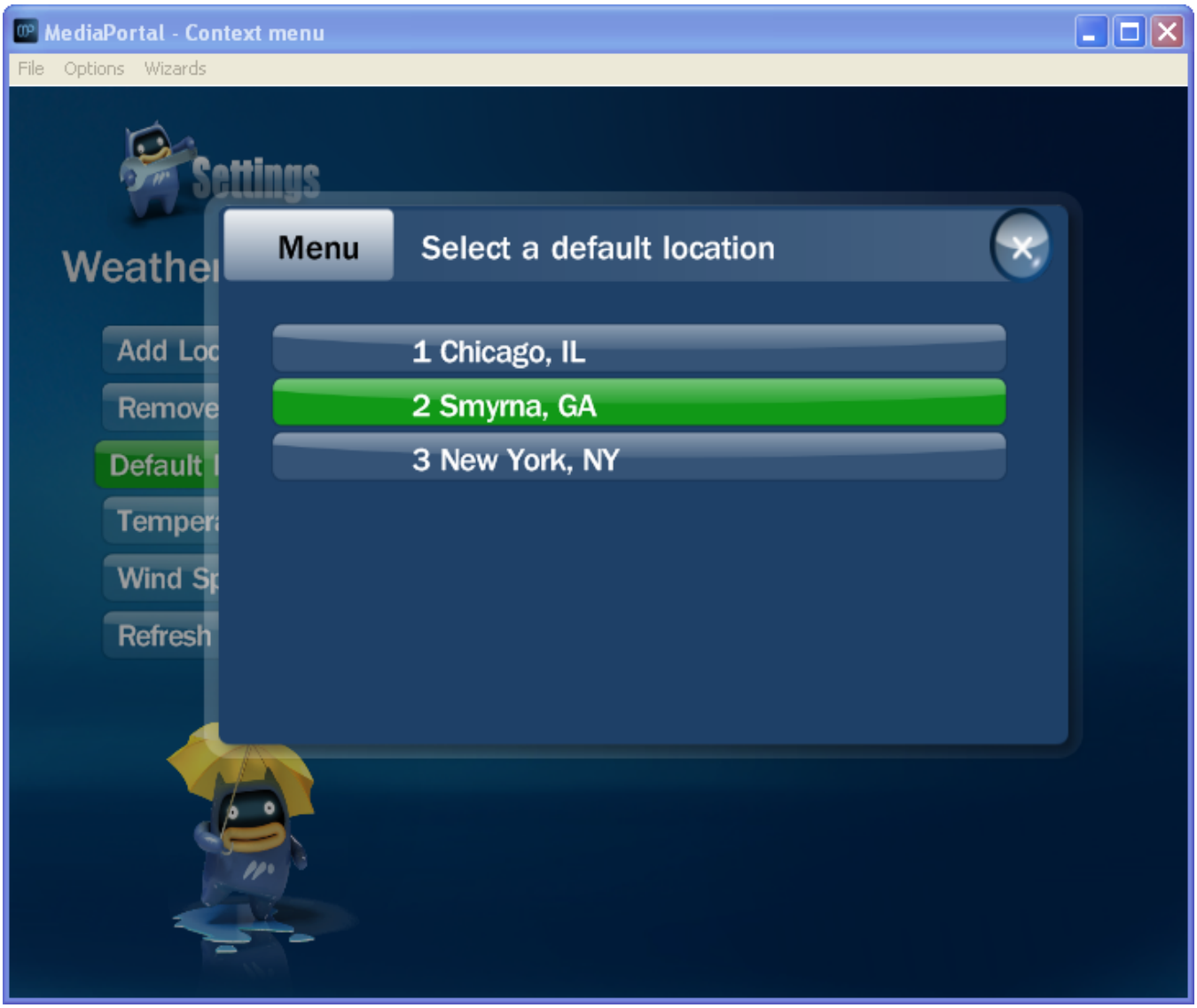
Changelog

Change	Date	Version
MenuButton in MP plugins	2011/12/17	1.2.0 to 1.3.0
MenuButton Control	2011/12/17	1.2.0 to 1.3.0
Enhance textXOff in Buttons	2011/12/17	1.2.0 to 1.3.0
Text Padding	2013/01/27	1.2.0 to 1.3.0

Screenshots

On this weather settings page the last four buttons (Default location, Temperature Units, Wind Speed Units, and Refresh Interval) are all MenuButtons. Default location is <mode>dialoglist</mode>, the others are <mode>spinlist</mode>.


MediaPortal - Settings->Weather
File
Options
Wizards
Settings
Weather
Add Location
Remove Location
Default location: Smyrna, GA
Temperature Units
Farenheit
Wind Speed Units
mph
Refresh Interval
42 mins



Tags

MenuButton (GUIMenuButton)

Element Name	Data Type	Description
textureFocus [border, position, textureRepeat, textureRotate, texture, colorKey, corners, cornerRotate, mask, tileFill, mask]	String	The texture to display when the button has the focus/is selected.
border	String	With this feature you have the ability to add borders composed from textures that you identify. See Borders for more information.
position	Border Position	Specifies the position of the border relative to the image or control rectangle edges. Valid values are OutsideImage, InsideImage, CenterImage, OutsideControl, InsideControl, CenterControl. The default value is "OutsideImage". Example: <border position="CenterControl">10</border>.
textureRepeat	Boolean	Specifies whether the texture used for the border should repeat or stretch inside each of the four rectangles that compose the overall border. The default value is "no". Example: <border textureRepeat="yes">10</border>
textureRotate	Boolean	

		Specifies whether or not the texture used for the border should be rotated for each of the border rectangles. If the texture should rotate then the texture will be rotated 90 deg for the right border rectangle, 180 deg for the bottom, and 270 deg for the left. The default value is "no". Example: <border textureRotate="yes">10</border>
texture	String	Specifies the tetxure filename for the border rectangles. A single file is used for all four of the border rectangles. Based on the value of textureRepeat, the entire texture extent is either stretched (scaled up/down) to fill the border rectangles or is scaled (up/down) to fit inside the border rectangle at its native aspect ratio and repeatedly drawn until the border rectangle is filled. The default value is "image_border.png". This texture file must exist in the skin media directory otherwise no border will be drawn. Example: <border texture="my_border.png">10</border>
colorKey	Long	Specfies the color key for the border texture. The default value is 0xFFFFFFFF. Example: <border colorKey="0x66FFFFFF">10</border>
corners	Boolean	Specifies that the border should use corner rendering logic. The default value is false. Example: <border corners="yes">10</border>. Implies the use of a default texture file named image_border_corner.png.
cornerRotate	Boolean	Specifies that the border corner should be rotated for each corner. Does not imply that the corner rendering logic should be used. The default value is true. Example: <border corners="yes" cornerRotate="yes">10</border>. Again, implies the use of a default texture file named image_border_corner.png.
tileFill	Boolean	Specifies that the border texture should tile fill the border rectangles rather than stretch to fill the rectangles.
mask	String	Specifies an image texture filename for a mask. For example, ff the image is black with rounded corners (transparent corners) then the textureFocus image will be clipped to have round corners. The masking operation is not a clipping function; if the mask image is not black then image blending occurs. See Image Masks for more information
textureNoFocus [border, position, textureRepeat, textureRotate, texture, colorKey, corners, cornerRotate, mask, tileFill]	String	The texture to display when the button does not have the focus/is not selected.
border	String	With this feature you have the ability to add borders composed from textures that you identify. See Borders for more information.
position	Border Position	Specifies the position of the border relative to the image or control rectangle edges. Valid values are OutsideImage, InsideImage, CenterImage, OutsideControl, InsideControl, CenterControl. The default value is "OutsideImage". See Borders for a more detailled description.
textureRepeat	Boolean	Specifies whether the texture used for the border should repeat or stretch inside each of the four rectangles that compose the overall border. The default value is "no". Example: <border textureRepeat="yes">10</border>.
textureRotate	Boolean	Specifies whether or not the texture used for the border should be rotated for each of the border rectangles. If the texture should rotate then the texture will be rotated 90 deg for the right border rectangle, 180 deg for the bottom, and 270 deg for the left. The default value is "no". Example: <border textureRotate="yes">10</border>
texture	String	Specifies the tetxure filename for the border rectangles. A single file is used for all four of the border rectangles. Based on the value of textureRepeat, the entire texture extent is either stretched (scaled up/down) to fill the border rectangles or is scaled (up/down) to fit inside the border rectangle at its native aspect ratio and repeatedly drawn until the border rectangle is filled. The default value is "image_border.png". This texture file must exist in the skin media directory otherwise no border will be drawn. Example: <border texture="my_border.png">10</border>
colorKey	Long	Specfies the color key for the border texture. The default value is 0xFFFFFFFF. Example: <border colorKey="0x66FFFFFF">10</border>
corners	Boolean	Specifies that the border should use corner rendering logic. The default value is false. Example: <border corners="yes">10</border>. Implies the use of a default texture file named image_border_corner.png.
cornerRotate	Boolean	Specifies that the border corner should be rotated for each corner. Does not imply that the corner rendering logic should be used. The default value is true. Example: <border corners="yes" cornerRotate="yes">10</border>. Again, implies the use of a default texture file named image_border_corner.png.
tileFill	Boolean	Specifies that the border texture should tile fill the border rectangles rather than stretch to fill the rectangles.
mask	String	Specifies an image texture filename for a mask. For example, ff the image is black with rounded corners (transparent corners) then the textureFocus image will be clipped to have round corners. The masking operation is not a clipping function; if the mask image is not black then image blending occurs. See Image Masks for more information
hover [border, position, textureRepeat, textureRotate, texture, colorKey, corners, cornerRotate, mask, tileFill]	String	The texture to display when the button is hovered over.
border	String	With this feature you have the ability to add borders composed from textures that you identify. See Borders for more information.

position	Border Position	Specifies the position of the border relative to the image or control rectangle edges. Valid values are OutsideImage, InsideImage, CenterImage, OutsideControl, InsideControl, CenterControl. The default value is "OutsideImage". See Borders for a more detailled description.
textureRepeat	Boolean	Specifies whether the texture used for the border should repeat or stretch inside each of the four rectangles that compose the overall border. The default value is "no". Example: <border textureRepeat="yes">10</border>.
textureRotate	Boolean	Specifies whether or not the texture used for the border should be rotated for each of the border rectangles. If the texture should rotate then the texture will be rotated 90 deg for the right border rectangle, 180 deg for the bottom, and 270 deg for the left. The default value is "no". Example: <border textureRotate="yes">10</border>
texture	String	Specifies the tetxure filename for the border rectangles. A single file is used for all four of the border rectangles. Based on the value of textureRepeat, the entire texture extent is either stretched (scaled up/down) to fill the border rectangles or is scaled (up/down) to fit inside the border rectangle at its native aspect ratio and repeatedly drawn until the border rectangle is filled. The default value is "image_border.png". This texture file must exist in the skin media directory otherwise no border will be drawn. Example: <border texture="my_border.png">10</border>
colorKey	Long	Specfies the color key for the border texture. The default value is 0xFFFFFFFF. Example: <border colorKey="0x66FFFFFF">10</border>
corners	Boolean	Specifies that the border should use corner rendering logic. The default value is false. Example: <border corners="yes">10</border>. Implies the use of a default texture file named image_border_corner.png.
cornerRotate	Boolean	Specifies that the border corner should be rotated for each corner. Does not imply that the corner rendering logic should be used. The default value is true. Example: <border corners="yes" cornerRotate="yes">10</border>. Again, implies the use of a default texture file named image_border_corner.png.
tileFill	Boolean	Specifies that the border texture should tile fill the border rectangles rather than stretch to fill the rectangles.
mask	String	Specifies an image texture filename for a mask. For example, ff the image is black with rounded corners (transparent corners) then the textureFocus image will be clipped to have round corners. The masking operation is not a clipping function; if the mask image is not black then image blending occurs. See Image Masks for more information.
hoverX	Integer	The x position of the hover image.
hoverY	Integer	The y position of the hover image.
hoverWidth	Integer	The width of the hover image.
hoverHeight	Integer	The height of the hover image.
font	String	The font to use to display the button text.
label	String	The button text, property or a number that corresponds to an id in the strings.xml file.
textcolor	Long	The color of the text when the button has the focus/is selected.
textcolorNoFocus	Long	The color of the text when the button is not selected.
disabledcolor	Long	The color of the text when the button is disabled.
textXOff [hasMargin]	Integer	The number of pixels the text label is offset from the left edge of the button image.
hasMargin	Boolean	If true, textXOff is used to provide horizontal space before and after the button label.
textYOff	Integer	The number of pixels the text label is offset from the top edge of the button image.
textpadding	Integer	[Since 1.3] provides "space" inside the label text to prevent overlap with graphics that follow on the right.
mode	Button Mode	The mode in which the button will operate; dialoglist or spinlist.
textureUp	String	(applies only when mode=spinlist) The texture for the up arrow when the up arrow is not in focus.
textureDown	String	(applies only when mode=spinlist)The texture for the up arrow when the up arrow is not in focus.
textureUpFocus	String	(applies only when mode=spinlist) The texture for the up arrow when the up arrow is in focus.
textureDown	String	(applies only when mode=spinlist) The texture for the down arrow when the down arrow is not in focus.
textureDownFocus	String	(applies only when mode=spinlist) The texture for the down arrow when the down arrow is not in focus.
spinWidth	Integer	(applies only when mode=spinlist) The width of the embedded spin control.
spinHeight	Integer	(applies only when mode=spinlist) The height of the embedded spin control.
spinXOff	Integer	(applies only when mode=spinlist) The x position of the spin control relative to the button x position.
spinYOff	Integer	(applies only when mode=spinlist) The y position of the spin control relative to the button y position.

spinalign	Alignment	(applies only when mode=spinlist) The horizontal alignment of the spin control relative to the button; left, center, right.
spinvalign	VAlignment	(applies only when mode=spinlist) The vertical alignment of the spin control relative to the button; bottom, middle, top.
spinTextXOff	Integer	(applies only when mode=spinlist) The horizontal offset
spinTextYOff	Integer	(applies only when mode=spinlist)
showrange	Boolean	(applies only when mode=spinlist)
digits	Integer	(applies only when mode=spinlist)
reverse	Boolean	(applies only when mode=spinlist)
cycleItems	Boolean	(applies only when mode=spinlist)
spinType	SpinTyle	(applies only when mode=spinlist)
orientation	Orientation	(applies only when mode=spinlist)
dialogTitle	String	(applies only when mode=dialoglist)
dialogShowNumbers	Boolean	(applies only when mode=dialoglist)
onclick	String	Executes a MediaPortal skin function when the button is clicked. See Skin Settings and Skin Expressions for more information.
binding	String	Associates the value of a skin variable with the value of the button. The string value may be any valid Skin Expression. For example <binding>#Skin.CurrentTheme</binding> sets the value of the control to the value of #Skin.CurrentTheme. If the #Skin.CurrentTheme changes, even if the control is already rendered to the screen, then the button will be updated in real-time.
valueTextInButton [align]	Boolean	If true then the value of the control is displayed in the label portion of the button.
align	Alignment	The horizontal alignment of the spin control relative to the button; left, center, right.
valuePrefixText [join]	String	Prefix text to be concatenated with the value of the control.
join	String	Text used to join the prefix and control value strings. Useful for joining a localized string with the value.
valueSuffixText [join]	String	Suffix text to be concatenated with the value of the control.
join	String	Text used to join the suffix and control value strings. Useful for joining a localized string with the value.
scrollStartDelaySec	Integer	The amount of time, after the control gains focus, that the control will wait before horizontal scrolling of text begins. Value of -1 prevents scrolling.
scrollWrapString	String	A string used to concatenate the end of control label with the beginning of the control label when the text is scrolling.
shadowAngle	Integer	The integral angle, in degrees, of the shadow text. Zero degrees is along the x-axis, increasing positive values from zero will rotate the shadow clockwise.
shadowDistance	Integer	The number of pixels the shadow is offset from the normal (foreground) text.
shadowColor	Long	The color of the shadow.
textalign	Alignment	The horizontal alignmet of the text in the button; left, center, right.
textvalign	VAlignment	The vertical alignmet of the text in the button; bottom, middle, top.

Inherited by GUIControl

See GUIControl for the full documentation of this control.

Element Name	Data Type	Description
id	Integer	The id of the control. The id will couple the skin file to the code, so if we later on want to check that a user pressed a button, the id will be required and must be unique. For controls that will never be referenced in the code it is safe to set it to "1"
description	String	An optional description of the control for your reference

type	String	The type of the control, for instance "button", "label", "textbox" and all other controls.
posX	Integer	The X-position on the window for this control
posY	Integer	The Y-position on the window for this control
width	Integer	The width of this control
height	Integer	The height of this control
onleft	Integer	The control id to move the focus to when the user moves left. If not specified (or zero) MediaPortal will find the closest control in that direction to move to. As of **v1.7.0** Skin Settings and Skin Expressions are also supported.
onright	Integer	The control id to move the focus to when the user moves right. If not specified (or zero) MediaPortal will find the closest control in that direction to move to. As of **v1.7.0** Skin Settings and Skin Expressions are also supported.
onup	Integer	The control id to move the focus to when the user moves up. If not specified (or zero) MediaPortal will find the closest control in that direction to move to. As of **v1.7.0** Skin Settings and Skin Expressions are also supported.
ondown	Integer	The control id to move the focus to when the user moves down. If not specified (or zero) MediaPortal will find the closest control in that direction to move to. As of **v1.7.0** Skin Settings and Skin Expressions are also supported.
colordiffuse	Long	Allows you to mix a color & a graphics texture. E.g. If you have a graphics texture like a blue button you can mix it with a yellow color diffuse and the end result will be green. Defaults to 0xFFFFFFFF
dimColor	Integer	Color for a control when it is not focussed. Defaults to half transparent (0x60ffffff)
onfocus	String	[Since 1.3] Executes a MediaPortal skin function when the control gains focus. See Skin Settings for more information.

XML samples

```
<control>
    <description>default menubutton</description>
    <type>menubutton</type>
    <id>0</id>
    <mode>spinlist</mode>
    <width>260</width>
    <height>45</height>
    <textureUp>arrow_round_up_nofocus.png</textureUp>
    <textureUpFocus>arrow_round_up_focus.png</textureUpFocus>
    <textureDown>arrow_round_down_nofocus.png</textureDown>
    <textureDownFocus>arrow_round_down_focus.png</textureDownFocus>
    <spinWidth>24</spinWidth>
    <spinHeight>24</spinHeight>
    <spinXOff>13</spinXOff>
    <spinTextXOff>7</spinTextXOff>
    <spinTextYOff>2</spinTextYOff>
    <spinalign>right</spinalign>
    <spinvalign>middle</spinvalign>
    <spintype>text</spintype>
    <showrange>no</showrange>
    <reverse>no</reverse>
    <cycleItems>yes</cycleItems>
    <dialogTitle></dialogTitle>
    <dialogShowNumbers>no</dialogShowNumbers>
    <font>font12</font>
    <textcolor>ffffffff</textcolor>
    <colordiffuse>ffffffff</colordiffuse>
    <disabledcolor>ff808080</disabledcolor>
    <textcolorNoFocus>ffa9d0f7</textcolorNoFocus>
    <dimColor>ff000000</dimColor>
    <shadowAngle>45</shadowAngle>
    <shadowDistance>3</shadowDistance>
    <shadowColor>ff000000</shadowColor>
    <textXOff>17</textXOff>
    <textYOff>6</textYOff>
    <textureFocus>button_focus.png</textureFocus>
    <textureNoFocus>button_nofocus.png</textureNoFocus>
    <animation effect="zoom" start="100,100" end="105,105" time="50">focus</animation>
    <animation effect="zoom" start="105,105" end="100,100" time="0">unfocus</animation>
  </control>
```

```
  <control>
    <description>View-As</description>
    <type>menubutton</type>
    <id>2</id>
    <label></label>
    <textureFocus>hiddenmenu_item_selected.png</textureFocus>
    <textureNoFocus>-</textureNoFocus>
    <width>497</width>
    <height>64</height>
    <textXOff>58</textXOff>
    <textYOff>14</textYOff>
    <onright>50</onright>
    <onleft>50</onleft>
    <onup>66614</onup>
    <mode>dialoglist</mode>
    <dialogTitle>792</dialogTitle>
    <valueTextInButton>yes</valueTextInButton>
    <valuePrefixText>95</valuePrefixText>
  </control>
```

In the following example we combine several features to manage the display of an unknown set of values, the names of available skin themes. The key attributes are <valueTextInButton>, <onclick>, <binding>, and <subitems>. The value of the control is displayed in the label portion of the button. Because of the <binding> this allows the button to reflect the name of the currently selected skin theme. The values available in the menu are defined by the content of the <subitems>. In this case, MediaPortal maintains #Skin.Themes as a CSV string. The content of each (comma separated) element is presented as a selectable menu option. You may choose to add multiple, hardcoded <subitem> entries to fill the menu as well or in combination with a CSV entry. When the menu button is clicked the <onclick> skin function is executed. In this example, the function Skin.SetTheme() is called to set the theme to the value of this control. The notation for retrieving the value of this control is #selectedlabel<id>, where <id> is the control id. Concatenating the control id to #selectedlabel enables multple menu buttons to work on the same screen.

```
<control>
    <description>theme</description>
    <type>menubutton</type>
    <id>15</id>
    <width>471</width>
    <height>52</height>
    <mode>spinlist</mode>
    <valueTextInButton>yes</valueTextInButton>
    <valuePrefixText>94</valuePrefixText>
    <onclick>Skin.SetTheme(#selectedlabel15)</onclick>
    <binding>#Skin.CurrentTheme</binding>
    <subitems>
      <subitem>#Skin.Themes</subitem>
    </subitems>
    <spinTextXOff>7</spinTextXOff>
    <spinTextYOff>2</spinTextYOff>
  </control>
```

Position text using textXOff while preserving the width of the label text by setting the textXOff property hasMargin to "no";

```
<textXOff hasMargin="no">20</textXOff>
```

Menubutton and pre-defined controls:

```
<control>
  <description>"List rows" button</description>
  <type>menubutton</type>
  <id>33</id>
  <mode>dialoglist</mode>
  <dialogTitle>List rows</dialogTitle>
  <valuePrefixText>List rows: </valuePrefixText>
  <valueTextInButton>yes</valueTextInButton>
  <onclick>#(skin.setstring('#skin.list.rows',#selectedlabel33))</onclick>
```

```
  <binding>#skin.list.rows</binding>
  <subitems>
    <subitem>#(3)</subitem>
    <subitem>#(4)</subitem>
    <subitem>#(5)</subitem>
    <subitem>#(6)</subitem>
    <subitem>#(7)</subitem>
    <subitem>#(8)</subitem>
    <subitem>#(9)</subitem>
    <subitem>#(10)</subitem>
    <subitem>#(11)</subitem>
    <subitem>#(12)</subitem>
    <subitem>#(13)</subitem>
  </subitems>
  <onup>9</onup>
  <ondown>6</ondown>
  <onleft>10</onleft>
  <onright>10</onright>
</control>
```

In this code:

- **onclick** specifies the skin setting to be set with the value chosen by the user.
- **binding** specifies which skin setting contains the value to be used to highlight the current value when the menubutton list is displayed.
- **subitem** lists the possible values for this skin setting. In this example the user can choose any value in the range 3 to 13. Note that in this particular case I have to use expression notation to prevent the skin engine interpreting the value as a reference to a string in the language file.
- The control id is arbitrary -- you just need to choose a value that is not used by any of the pre-defined controls on that panel.