COMBODYN2

The COMBODYN control is the dynamic counterpart of the COMBOFIX control. Whereas the selection options inside the COMBOFIX control are defined in a fixed way inside the page definition, the COMBODYN2 control offers the possibility to derive the selection options dynamically from adapter properties.

The following topics are covered below:

- Example
- Typical Problems with COMBODYN2
- Properties

Example

The XML layout definition looks as follows:

```
<rowarea name="ComboDyn">
    <itr>
        <label name="Cost Center" width="120">
        </label>
        <combodyn2 valueprop="costCenter" validvaluesprop="validCostCenters"
                   width="200" size="1">
        </combodyn2>
    </itr>
</rowarea>
```

The definition of the COMBODYN2 control refers to a `valueprop` property: this is the property of the adapter class in which the selection is actually passed. In addition, the definition refers to a `validvaluesprop` property: this is the property from which the options are taken.

The code of the corresponding adapter class looks as follows:

```
import com.softwareag.cis.server.Adapter;
import com.softwareag.cis.server.util.COMBODYNValidValues;

// This class is a generated one.

public class ComboFixAdapter
    extends Adapter
{
    // property >costCenter<
    String m_costCenter;
    public String getCostCenter() { return m_costCenter; }
    public void setCostCenter(String value) { m_costCenter = value; }
```

```
    // property >validCostCenters<
    COMBODYNValidValues m_validCostCenters = new COMBODYNValidValues();
    public COMBODYNValidValues getValidCostCenters() { return m_validCostCenters; }

    /** initialisation - called when creating this instance*/
    public void init()
    {
        m_validCostCenters.addValidValue("0001","Marketing");
        m_validCostCenters.addValidValue("0002","Sales");
        m_validCostCenters.addValidValue("0003","Development");
    }
}
```

Typical Problems with COMBODYN2

The rendering problems with the internally used HTML control SELECT also apply for the COMBODYN2 control. See the corresponding information in the section *Typical Problems with COMBOFIX*.

For this reason, COMBODYN2 offers the property `renderasfield`: when switched to "true", the rendering is not done by using the HTML control SELECT, but by using the normal Application Designer FIELD with valid value support. Rendering as FIELD has the following advantages:

- There are no overlapping conflicts anymore.
- Valid values are brought to the client at the point of time when the user requests value help.

But there is also a disadvantage:

- When selecting a value from the valid value list, the value is displayed with its ID - not with its description.

Properties

Basic			
valueprop	Server side property representation of the control.	Obligatory	
validvaluesprop	Adapter property that provides for the valid values that are available as selectable options. The adapter property must be of type "COMBODYNValidValues".	Obligatory	

width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	
Appearance			
width	(already explained above)		
size	Number of rows that are displayed inside the control. If specified as "1" (default) then the control is rendered as combo box - if ">1" then the control is rendered as multi line selection.	Optional	
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control’s contained text.	Optional	left center right

valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control two span over more than one columns. The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
renderasfield	If set to "true" then the combo box is rendered like a FIELD control that offers valid value support. Default is "false". The normal translation of COMBODYN2 into HTML renders an HTML-select control. This control has certain limitations inside Internet Explorer: it only offers a very reduced set of styles to manipulate its look and feel and - much worse: it always occupies z-index "0" i.e. if you other areas overlapping the COMBODYN2 area then COMBODYN2 is always on the top. This is quite ugly if e.g. a menu is opened and parts of the menu overlap a COMBODYN2 control.	Optional	true false
allowmultiselection	If set to true then multiple selections are allowed.	Optional	true false

combostyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
valueprop	(already explained above)		
validvaluesprop	(already explained above)		
displayprop	Name of adapter property that controls whether the field is displayonly(true) or not (false). By using this property you can dynamically control the "display"-status of the control by your adapter object.	Optional	

statusprop	Name of the adapter property that dynamically passes information how the field should be rendered and how it should act.	Optional	
titleprop	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchonization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representaion directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit method to be called when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
titleprop	(already explained above)		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	