Sage 300 Web Screens SDK

Html Helpers

August 2021

The MIT License (MIT)

Copyright © 2021 The Sage Group plc or its licensors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

[1. Introduction 5](#_Toc77242488)

[2. Composite Extensions 6](#_Toc77242489)

[2.1 SgNumericFor 7](#_Toc77242490)

[2.1.1 Flow 7](#_Toc77242491)

[2.1.2 Interface 8](#_Toc77242492)

[2.1.3 Parameters 8](#_Toc77242493)

[2.1.4 Individual Helpers Example 10](#_Toc77242494)

[2.1.5 New Composite Helper Example 10](#_Toc77242495)

[2.2 SgTextFor 11](#_Toc77242496)

[2.2.1 Flow 11](#_Toc77242497)

[2.2.2 Interface 12](#_Toc77242498)

[2.2.3 Parameters 12](#_Toc77242499)

[2.2.4 Individual Helpers Example 14](#_Toc77242500)

[2.2.5 New Composite Helper Example 14](#_Toc77242501)

[2.3 SgFinderFor 15](#_Toc77242502)

[2.3.1 Flow 16](#_Toc77242503)

[2.3.2 Interface 17](#_Toc77242504)

[2.3.3 Parameters 17](#_Toc77242505)

[2.3.4 Individual Helpers Example 20](#_Toc77242506)

[2.3.5 New Composite Helper Example 20](#_Toc77242507)

[2.4 SgPasswordFor 22](#_Toc77242508)

[2.4.1 Flow 22](#_Toc77242509)

[2.4.2 Interface 23](#_Toc77242510)

[2.4.3 Parameters 23](#_Toc77242511)

[2.4.4 Individual Helpers Example 25](#_Toc77242512)

[2.4.5 New Composite Helper Example 25](#_Toc77242513)

[2.5 SgDropdownFor 26](#_Toc77242514)

[2.5.1 Flow 26](#_Toc77242515)

[2.5.2 Interface 27](#_Toc77242516)

[2.5.3 Parameters 27](#_Toc77242517)

[2.5.4 Individual Helpers Example 29](#_Toc77242518)

[2.5.5 New Composite Helper Example 29](#_Toc77242519)

[2.6 SgDatepickerFor 31](#_Toc77242520)

[2.6.1 Flow 31](#_Toc77242521)

[2.6.2 Interface 32](#_Toc77242522)

[2.6.3 Parameters 32](#_Toc77242523)

[2.6.4 Individual Helpers Example 34](#_Toc77242524)

[2.6.5 New Composite Helper Example 34](#_Toc77242525)

[2.7 SgTimepickerFor 35](#_Toc77242526)

[2.7.1 Flow 35](#_Toc77242527)

[2.7.2 Interface 36](#_Toc77242528)

[2.7.3 Parameters 36](#_Toc77242529)

[2.7.4 Individual Helpers Example 38](#_Toc77242530)

[2.7.5 New Composite Helper Example 38](#_Toc77242531)

[2.8 SgCheckboxFor 39](#_Toc77242532)

[2.8.1 Flow 39](#_Toc77242533)

[2.8.2 Interface 40](#_Toc77242534)

[2.8.3 Parameters 40](#_Toc77242535)

[2.8.4 Individual Helpers Example 41](#_Toc77242536)

[2.8.5 New Composite Helper Example 41](#_Toc77242537)

[2.9 SgRadioButtonFor 42](#_Toc77242538)

[2.9.1 Flow 42](#_Toc77242539)

[2.9.2 Interface 43](#_Toc77242540)

[2.9.3 Parameters 43](#_Toc77242541)

[2.9.4 Individual Helpers Example 44](#_Toc77242542)

[2.9.5 New Composite Helper Example 44](#_Toc77242543)

1. Introduction

The Sage 300 web screens use HTML Helpers in the ASP.NET MVC razor views.

Always use helpers instead of writing HTML tags manually.

Sage provides numerous extensions to the HTML Helper class and the usages and interfaces are explained in this document.

1. Composite Extensions

This section will describe the various composite extensions that are available in the Sage 300 Web Screens.

A composite extension is one that creates multiple controls or widgets.

For example, here is an existing snippet that creates a Label, a Numeric Textbox, and a Validation Message:

@Html.SageLabelFor(model => model.Data.Quantity, new { @id = "lblQuantity", @class = "" })

@Html.KoSageNumericBoxFor(model => model.Data.Quantity, new { @value = "Data.Quantity", @sagevalue = "Data.Quantity", @sagedisable = "IsQuantityDisabled" }, new { @id = "nbrQuantity", @class = "numeric default" })

@Html.ValidationMessageFor(model => model.Data.Quantity, null, new { @class = "" })

The three lines above can be replaced with a single composite extension as follows:

@Html.SgNumericFor(model => model.Data.Quantity, new { sagevalue = "Data.Quantity", sagedisable = " IsQuantityDisabled " })

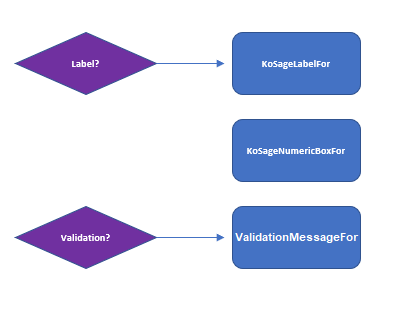
The purpose of the composite extensions is to:

* Create additional controls if specified while creating the main control.
* Internally implement functionality for the developer instead of requiring some information to be passed to the helper.
* Name controls based upon standards while allowing this convention to be overridden.
* Simplification!
  1. SgNumericFor

The ***SgNumericFor*** helper will create a numeric textbox, an optional label, and an optional validation.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that a validation message will be created.
* The default is that the size will default to ‘default’.
* The default is that the numeric textbox will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label, a numeric textbox, and an optional validation

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the textbox</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for the textbox, otherwise null</param>

/// <param name="format">Format for the input otherwise empty</param>

/// <param name="size">The size of the numeric textbox otherwise 'default'</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for the label, otherwise null</param>

/// <param name="includeValidation">True to add validation otherwise false (no validation)</param>

/// <param name="validationText">Validation override message otherwise null</param>

/// <param name="validationHtmlAttrs">HTML attributes for the validation, otherwise null</param>

/// <returns>HTML input elements for a textbox and other optional controls (label, validation)</returns>

public static MvcHtmlString SgNumericFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, string format = "", string size = "default",

bool includeLabel = true, string labelText = null, object labelDataAttrs = null, object labelHtmlAttrs = null,

bool includeValidation = true, string validationText = null, object validationHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.Quantity |
| dataAttrs | ***Required***. The knockout data attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { sagevalue = "Quantity", sagedisable = "IsQuantityDisabled" }  Note: If specified, the sagevalue will be copied to the value attribute.  Note: If specified, the sagedisable attribute will be copied to the disable attribute. |
| htmlAttrs | ***Optional***. The html attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “numeric {size}” and therefore does not need to be assigned externally. | htmlAttrs: new { id = "nbrQuantity", @class = "data-nav" } |
| format | ***Optional****.* The format for the input if required to be different from default masking | format: “0” |
| size | ***Optional.*** The size for the numeric textbox. Defaults to “default”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “small” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Quantity |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsQuantityDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "Line\_Number", @class = “required” } |
| includeValidation | ***Optional.*** True to add validation otherwise no validation will be added. Defaults to true. | includeValidation: false |
| validationText | ***Optional.*** If a validation is included, the validation text will be assigned from the attribute of the expression’s property unless being overridden by this parameter | validationText: ReceiptResx.ValidCurrency |
| validationHtmlAttrs | ***Optional***. The html attributes for the validation specified in an anonymous object or a RouteValueDictionary. | validationHtmlAttrs: new { @class = “vlidation-bottom” } |

* + 1. Individual Helpers Example

<div class="numeric-group">

@Html.SageLabel("lblUnitWeight",ICCommonResx.UnitWeight)

@Html.KoSageNumericBoxFor(m => m.Data.UnitWeight,

new { @sagevalue = "Data.UnitWeight"},

new { @id = "txtUnitWeight", @class = "numeric default" })

</div>

* + 1. New Composite Helper Example

<div class="numeric-group">

@Html.SgNumericFor(m => m.Data.UnitWeight,

new { sagevalue = "Data.UnitWeight" },

new { id = "txtUnitWeight" },

labelText: ICCommonResx.UnitWeight, includeValidation: false)

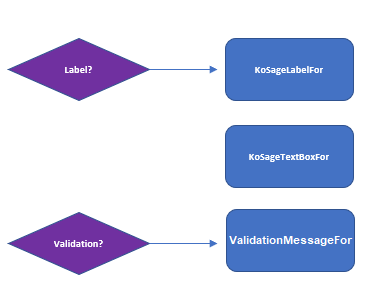
</div>

* Do not need to specify label id attribute as helper defaults to these values.
* Do not need to specify **numeric** or **default** values for class attribute as helper defaults to these values.
* Need to specify no to validation message as helper defaults to including validation message.
  1. SgTextFor

The ***SgTextFor*** helper will create a text textbox, an optional label, and an optional validation.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that a validation message will be created.
* The default is that the size will default to ‘default’.
* The default is that the textbox will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label, a text textbox, and an optional validation

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the textbox</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for the textbox, otherwise null</param>

/// <param name="size">The size of the numeric textbox otherwise 'default'</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for the label, otherwise null</param>

/// <param name="includeValidation">True to add validation otherwise false (no validation)</param>

/// <param name="validationText">Validation override message otherwise null</param>

/// <param name="validationHtmlAttrs">HTML attributes for the validation, otherwise null</param>

/// <returns>HTML input elements for a textbox and other optional controls (label, validation)</returns>

public static MvcHtmlString SgTextFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, string size = "default",

bool includeLabel = true, string labelText = null, object labelDataAttrs = null, object labelHtmlAttrs = null,

bool includeValidation = true, string validationText = null, object validationHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.Description |
| dataAttrs | ***Required***. The knockout data attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { sagevalue = "Data.Description", sagedisable = "IsDescriptionDisabled" }  Note: If specified, the sagevalue will be copied to the value attribute.  Note: If specified, the sagedisable attribute will be copied to the disable attribute.  Note: valueUpdate will default to “’input’” and does not need to be specified unless another value for valueUpdate is required. |
| htmlAttrs | ***Optional***. The html attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “ {size}” and therefore does not need to be assigned externally. | htmlAttrs: new { id = "txtDescription"} |
| size | ***Optional.*** The size for the textbox. Defaults to “default”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “large” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsDescriptionDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "lblDescription", @class = “required” } |
| includeValidation | ***Optional.*** True to add validation otherwise no validation will be added. Defaults to true. | includeValidation: false |
| validationText | ***Optional.*** If a validation is included, the validation text will be assigned from the attribute of the expression’s property unless being overridden by this parameter | validationText: ReceiptResx.ValidCurrency |
| validationHtmlAttrs | ***Optional***. The html attributes for the validation specified in an anonymous object or a RouteValueDictionary. | validationHtmlAttrs: new { @class = “vlidation-bottom” } |

* + 1. Individual Helpers Example

<div class="input-group">

@Html.SageLabel("Address\_ShortName", VendorResx.ShortName)

@Html.KoSageTextBoxFor(m => m.Data.ShortName,

new { @sagevalue = "Data.ShortName", @valueUpdate = "'input'" },

new { @id = "txtShortName", @maxlength = "10", @class = "default" })

</div>

* + 1. New Composite Helper Example

<div class="input-group">

@Html.SgTextFor(m => m.Data.ShortName,

new { @sagevalue = "Data.ShortName" }, new { @id = "txtShortName", @maxlength = "10" },

labelHtmlAttrs: new { @id = "Address\_ShortName" }, labelText: VendorResx.ShortName,

includeValidation: false)

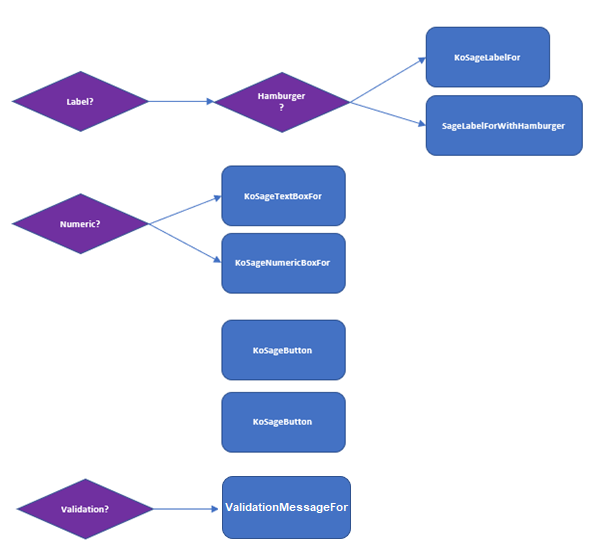
</div>

* Need to specify label id attribute to override helper default.
* Do not need to specify **default** value for class attribute and **‘input’** for valueUpdate attribute as helper defaults to these values.
* Need to specify no to validation message as helper defaults to including validation message.
  1. SgFinderFor

The ***SgFinderFor*** helper will create a numeric or text textbox, an optional label or hamburger label, a “go” button, a ‘finder” button, and an optional validation.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that a hamburger label will not be created.
  + If will be named “lnk” + propertyName
  + It can be overridden with the “id” html attribute.
* The default is that the textbox will be for text.
  + It can be made to be numeric by specifying the isNumeric parameter.
* The default is that a validation message will be created.
* The default is that the size will default to ‘default’.
* The default is that the textbox will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that the Go button will be named “btnLoad” + propertyName.
  + It can be overridden by specifying the goButtonId parameter.
* The default is that the Find button will be named “btnFinder” + propertyName.
  + It can be overridden by specifying the findButtonId parameter.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label, a finder (textbox, go button, finder button), and an optional validation

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the textbox</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for the textbox, otherwise null</param>

/// <param name="size">The size of the numeric textbox otherwise 'default'</param>

/// <param name="isNumeric">True if finder is numeric otherwise false (text)</param>

/// <param name="goButtonId">Go button id otherwise defaults to btnLoad... </param>

/// <param name="findButtonId">Find button id otherwise defaults to btnFinder...</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for the label, otherwise null</param>

/// <param name="labelWithHamburger">True to add a hamburger to the label otherwise false</param>

/// <param name="hamburgerUrl">The text to use for the hamburger menu's HREF (NOT USED)</param>

/// <param name="hamburgerValue">The inner text of the hamburger menu</param>

/// <param name="buttonDataAttrs">Data attributes for the button, otherwise null</param>

/// <param name="includeValidation">True to add validation otherwise false (no validation)</param>

/// <param name="validationText">Validation override message otherwise null</param>

/// <param name="validationHtmlAttrs">HTML attributes for the validation, otherwise null</param>

/// <returns>HTML input elements for a finder and other optional controls (label, validation)</returns>

public static MvcHtmlString SgFinderFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, string size = "default", bool isNumeric = false,

string goButtonId = null, string findButtonId = null,

bool includeLabel = true, string labelText = null,

object labelDataAttrs = null, object labelHtmlAttrs = null,

bool labelWithHamburger = false, string hamburgerUrl = null, string hamburgerValue = null,

object buttonDataAttrs = null,

bool includeValidation = true, string validationText = null, object validationHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.Project |
| dataAttrs | ***Required***. The knockout data attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { sagevalue = "Data.Project", sagedisable = "IsProjectDisabled" }  Note: If specified, the sagevalue will be copied to the value attribute.  Note: If specified, the sagedisable attribute will be copied to the disable attribute.  Note: For text texboxes only, valueUpdate will default to “’input’” and does not need to be specified unless another value for valueUpdate is required. |
| htmlAttrs | ***Optional***. The html attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “ {size}” or “numeric {size}” and therefore does not need to be assigned externally. | htmlAttrs: new { id = "txtProject"} |
| size | ***Optional.*** The size for the textbox. Defaults to “default”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “small” |
| isNumeric | ***Optional.*** True for a numeric textbox otherwise false for a text textbox. Defaults to false. | isNumeric: true |
| goButtonId | ***Optional.*** The name of the Go Button. Defaults to null and will be created internally as “btnLoad” + propertyName. | goButtonId: “btnGo” |
| findButtonId | ***Optional.*** The name of the Finder Button. Defaults to null and will be created internally as “btnFinder” + propertyName | findButtonId: “btnFinder” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsProjectDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. * If labelWithHamburger is true,   + The “id” attribute can be used when the default of “lnk” + propertyName requires overriding.   + The “title” attribute may be specified to provide a title for the hamburger menu | labelHtmlAttrs: new { id = "lblProject", @class = “required” }  labelHtmlAttrs: new { id = "lnkProject", @class = “required”, @title = APCommonResx.Details } |
| labelWithHamburger | ***Optional.*** True to add a hamburger to the label otherwise false. Defaults to false.  Note: If specifying a hamburger, the labelHtmlAttrs parameter to be used to specify an id, title and other attributes. | labelWithHamburger: true |
| hamburgerUrl | ***Optional.*** The text to use for the hamburger’s HREF.  Note: Not used at present time |  |
| hamburgerValue | ***Optional.*** The inner text of the hamburger menu. |  |
| ­­buttonDataAttrs | ***Optional***. The knockout data attributes for the buttons specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable   Note: Separate attributes maybe required for scenarios where the buttons may or may not be disabled like the textbox. This allows the developer to set these attributes independently for the textbox and the buttons | buttonDataAttrs: new { sagedisable = "IsProjectDisabled" } |
| includeValidation | ***Optional.*** True to add validation otherwise no validation will be added. Defaults to true. | includeValidation: false |
| validationText | ***Optional.*** If a validation is included, the validation text will be assigned from the attribute of the expression’s property unless being overridden by this parameter | validationText: ReceiptResx.ValidCurrency |
| validationHtmlAttrs | ***Optional***. The html attributes for the validation specified in an anonymous object or a RouteValueDictionary. | validationHtmlAttrs: new { @class = “vlidation-bottom” } |

* + 1. Individual Helpers Example

<div class="search-group">

@Html.SageLabel("TermsCode", APCommonResx.TermsCode, new { @class = "required" })

@Html.KoSageTextBoxFor(m => m.Data.TermsCode, new { @value = "Data.TermsCode",

@valueUpdate = "'input'" }, new { @maxlength = "6", @class = "txt-upper default",

@formatTextbox = "alphaNumeric" })

@Html.KoSageButton("btnLoadTermsCode",

new { @value = "Data.TermsCode", @disable = "false" },

new { @id = "btnLoadTermsCode", @tabindex = -1, @class = "icon btn-go" })

@Html.KoSageButton("btnTermsFinder", new { },

new { @class = "icon btn-search positionTM1", @id = "btnTermsFinder", @tabindex = -1 })

@Html.ValidationMessageFor(m => m.Data.TermsCode)

</div>

* + 1. New Composite Helper Example

<div class="search-group">

@Html.SgFinderFor(m => m.Data.TermsCode,

new { @value = "Data.TermsCode" },

new { @maxlength = "6", @class = "txt-upper", @formatTextbox = "alphaNumeric" },

findButtonId: "btnTermsFinder",

labelHtmlAttrs: new { @id = "TermsCode", @class = "required" },

labelText: APCommonResx.TermsCode)

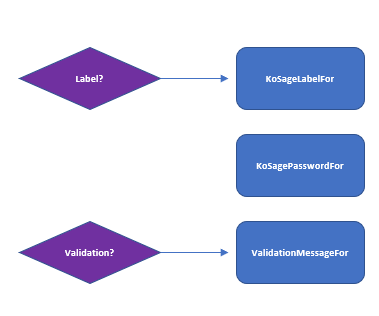
</div>

* Need to specify label id attribute to override helper default.
* Need to specify **labelText** value since an override is requested.
* Need to specify button name to override helper default.
* Do not need to specify **icon**, **btn-go**, **btn-search**, and **default** for class attributes as helper defaults these values.
* Do not need to specify **tabindex** attribute as helper defaults this value
  1. SgPasswordFor

The ***SgPasswordFor*** helper will create a password textbox, an optional label, and an optional validation.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that a validation message will be created.
* The default is that the size will default to ‘default’.
* The default is that the textbox will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label, a password textbox, and an optional validation

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the textbox</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for the textbox, otherwise null</param>

/// <param name="size">The size of the numeric textbox otherwise 'default'</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for the label, otherwise null</param>

/// <param name="includeValidation">True to add validation otherwise false (no validation)</param>

/// <param name="validationText">Validation override message otherwise null</param>

/// <param name="validationHtmlAttrs">HTML attributes for the validation, otherwise null</param>

/// <returns>HTML input elements for a textbox and other optional controls (label, validation)</returns>

public static MvcHtmlString SgPasswordFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, string size = "default",

bool includeLabel = true, string labelText = null, object labelDataAttrs = null, object labelHtmlAttrs = null,

bool includeValidation = true, string validationText = null, object validationHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.Password |
| dataAttrs | ***Required***. The knockout data attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { sagevalue = "Data.Password", sagedisable = "IsPasswordDisabled" }  Note: If specified, the sagevalue will be copied to the value attribute.  Note: If specified, the sagedisable attribute will be copied to the disable attribute.  Note: valueUpdate will default to “’input’” and does not need to be specified unless another value for valueUpdate is required. |
| htmlAttrs | ***Optional***. The html attributes for the textbox specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “ login {size}” and therefore does not need to be assigned externally. | htmlAttrs: new { id = "txtPassword"}  Note: autocomplete will default to “off” and does not need to be specified unless another value for autocomplete is required.  Note: data\_val will default to “true” and does not need to be specified unless another value for data\_val is required.  Note: placeholder will default to “” and does not need to be specified unless another value for placeholder is required. |
| size | ***Optional.*** The size for the textbox. Defaults to “default”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “large” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsPasswordDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "lblDescription", @class = “required” } |
| includeValidation | ***Optional.*** True to add validation otherwise no validation will be added. Defaults to true. | includeValidation: false |
| validationText | ***Optional.*** If a validation is included, the validation text will be assigned from the attribute of the expression’s property unless being overridden by this parameter | validationText: UserResx.Password |
| validationHtmlAttrs | ***Optional***. The html attributes for the validation specified in an anonymous object or a RouteValueDictionary. | validationHtmlAttrs: new { @class = “vlidation-bottom” } |

* + 1. Individual Helpers Example

<div class="input-group">

@Html.SageLabel("CompanyEmail\_Password", CompanyProfileResx.Password)

@Html.KoSagePasswordFor(m => m.Data.CompanyProfileEmail.Password,

new { @value = "Data.CompanyProfileEmail.Password", @valueUpdate = "'input'" },

new { @class = "medium" })

</div>

* + 1. New Composite Helper Example

<div class="input-group">

@Html.SgPasswordFor(m => m.Data.CompanyProfileEmail.Password,

new { @value = "Data.CompanyProfileEmail.Password" }, size: "medium",

labelHtmlAttrs: new { @id = " CompanyEmail\_Password " },

labelText: CompanyProfileResx.Password, includeValidation: false)

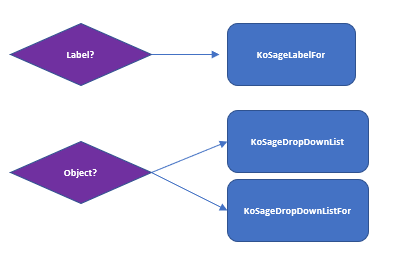
</div>

* Need to specify label id attribute to override helper default.
* Need to specify **labelText** value since an override is requested.
* Do not need to specify the **valueUpdate** attribute as the helper defaults this value.
* Do not need to specify **medium** for class attribute.
* Need to specify **size** parameter to override helper default.
* Need to specify **includeValidation** to override helper default.
  1. SgDropdownFor

The ***SgDropdownFor*** helper will create a dropdown list and an optional label.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that the size will default to ‘default’.
* The default is that the dropdown will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label and a dropdown list

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the dropdown</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class), otherwise null</param>

/// <param name="selectList">A collection of objects, otherwise null /> </param>

/// <param name="optionLabel">The text for a default empty item, otherwise null</param>

/// <param name="size">The size of the dropdown otherwise 'default'</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class), otherwise null</param>

/// <returns>HTML input elements for a dropdown and optional control (label)</returns>

public static MvcHtmlString SgDropdownFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, IEnumerable<CustomSelectList> selectList = null,

string optionLabel = null, string size = "default",

bool includeLabel = true, string labelText = null, object labelDataAttrs = null,

object labelHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.JobRoles |
| dataAttrs | ***Required***. The knockout data attributes for the dropdown specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { @options = "JobRoles", @value = "Data.JobRole", @sagedisable = "IsJobRoleDisabled" }  Note: **options** is required to specify a method name to populate the dropdown otherwise **selectList** will be specified and **options** can be omitted.  Note: If specified, the sagedisable attribute will be copied to the disable attribute.  Note: If specified, the sagevalue attribute will be copied to the value attribute.  Note: optionsText will default to “’Text’” and does not need to be specified unless another value for valueUpdate is required.  Note: optionsValue will default to “’Value’” and does not need to be specified unless another value for valueUpdate is required. |
| htmlAttrs | ***Optional***. The html attributes for the dropdown specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “ {size}” and therefore does not need to be assigned externally. | new { @id = "Data\_JobRole", @class = "single-select" }  Note: Specify single-select when single select behavior is required |
| selectList | ***Optional:*** A collection of CustomSelectList objects to be used to populate the dropdown | selectList: Model.GetRoles  Note: **selectList** is required when **options** is omitted from the dataAttrs parameter. |
| optionLabel | ***Optional:*** the text for a default empty item | optionLabel: “None” |
| size | ***Optional.*** The size for the textbox. Defaults to “default”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “large” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsPasswordDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "lblDescription", @class = “required” } |

* + 1. Individual Helpers Example

<div class="dropdown-group">

@Html.SageLabelFor(m => m.Data.AgedByTypes)

@Html.KoSageDropDownList("Data\_DueDateOrInvoiceDate",

new { @options = "AgedBys", @value = "Data.AgedByTypes",

@optionsText = "'Text'", @optionsValue = "'Value'" },

new { @id = "Data\_DueDateOrInvoiceDate", @class="small" })

</div>

<div class="dropdown-group">

@Html.SageLabel("CalculateBaseforDiscountwithTax", TermCodeResx.CalcBaseforDiscountwithTax,

new { @class = "" })

@Html.KoSageDropDownListFor(m => m.Data.CalcBaseforDiscountwithTax,

Model.BaseForDiscountWithTax, null,

new { @optionsText = "'Text'", @optionsValue = "'Value'" }, new { @class = "default" })

</div>

* + 1. New Composite Helper Example

<div class="dropdown-group">

@Html.SgDropdownFor(m => m.Data.AgedByTypes,

new { @options = "AgedBys", @value = "Data.AgedByTypes" },

new { @id = "Data\_DueDateOrInvoiceDate"}, size: "small" )

</div>

<div class="dropdown-group">

@Html.SgDropdownFor(m => m.Data.CalcBaseforDiscountwithTax,

new { @value = "Data.CalcBaseforDiscountwithTax" },

labelHtmlAttrs: new { @id = " CalculateBaseforDiscountwithTax" },

labelText: TermCodeResx.CalcBaseforDiscountwithTax, selectList: Model.BaseForDiscountWithTax)

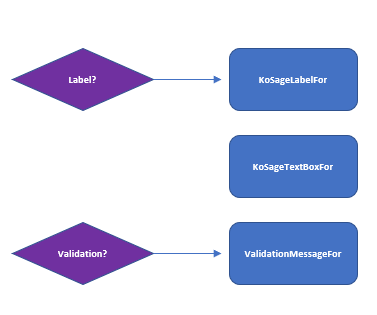
</div>

* Do not need to specify the **optionsText** or **optionsValue** attributes as the helper defaults these values.
* Do not need to specify **small** or **default** for class attribute as helper defaults to **default.**
* Need to specify **size** parameter (for **small**) to override helper default.
* Need to specify **id** attribute to override label default.
* Need to specify **labelText** parameter to override helper default.
* Need to specify **selectList** parameter for list to use to populate dropdown.
  1. SgDatepickerFor

The ***SgDatepickerFor*** helper will create a date picker, an optional label, and an optional validation.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that a validation message will be created.
* The default is that the size will default to ‘small’.
* The default is that the datepicker will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label, a date picker, and an optional validation

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the date</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for date, otherwise null</param>

/// <param name="size">The size of the date otherwise 'small'</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for label, otherwise null</param>

/// <param name="includeValidation">True to add validation otherwise false </param>

/// <param name="validationText">Validation override message otherwise null</param>

/// <param name="validationHtmlAttrs">HTML attributes for the validation, otherwise null</param>

/// <returns>HTML input elements for a date picker and optional label and validation</returns>

public static MvcHtmlString SgDatepickerFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, string size = "small",

bool includeLabel = true, string labelText = null, object labelDataAttrs = null,

object labelHtmlAttrs = null,

bool includeValidation = true, string validationText = null,

object validationHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.StartDate |
| dataAttrs | ***Required***. The knockout data attributes for the date specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { @sageDatePicker = "Data.StartDate" }  Note: If specified, the sagedisable attribute will be copied to the disable attribute. |
| htmlAttrs | ***Optional***. The html attributes for the date specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “ datepicker {size}” and therefore does not need to be assigned externally. | htmlAttrs: new { id = "txtStartDate"} |
| size | ***Optional.*** The size for the dropdown. Defaults to “small”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “default” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsDateDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "Address\_StartDate", @class = “required” } |
| includeValidation | ***Optional.*** True to add validation otherwise no validation will be added. Defaults to true. | includeValidation: false |
| validationText | ***Optional.*** If a validation is included, the validation text will be assigned from the attribute of the expression’s property unless being overridden by this parameter | validationText: ReceiptResx.ValidCurrency |
| validationHtmlAttrs | ***Optional***. The html attributes for the validation specified in an anonymous object or a RouteValueDictionary. | validationHtmlAttrs: new { @class = “vlidation-bottom” } |

* + 1. Individual Helpers Example

<div class="datepicker-group">

@Html.SageLabelFor(m => m.Data.BatchDate)

@Html.KoSageTextBoxFor(m => m.Data.BatchDate,

new { @sageDatePicker = "Data.BatchDate", @sagedisable = "Data.IsDisableBatchHeader" },

new { @class = "datepicker default" })

@Html.ValidationMessageFor(m => m.Data.BatchDate

</div>

* + 1. New Composite Helper Example

<div class="datepicker-group">

@Html.SgDatepickerFor(m => m.Data.BatchDate,

new { @sageDatePicker = "Data.BatchDate", @sagedisable = "Data.IsDisableBatchHeader" },

size: "default"

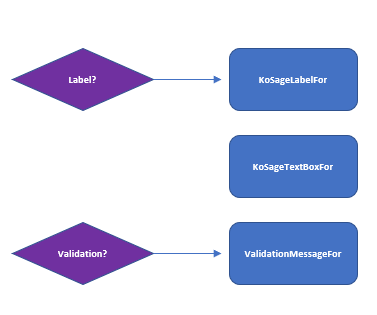
</div>

* Need to specify **size** parameter (for **default**) to override helper default.
* Do not need to specify **datapicker** for class attribute as helper defaults this value.
  1. SgTimepickerFor

The ***SgTimepickerFor*** helper will create a time picker, an optional label, and an optional validation.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that a validation message will be created.
* The default is that the size will default to ‘small’.
* The default is that the timepicker will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label, a time picker, and an optional validation

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <typeparam name="TProperty">The type of the property</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the time</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for the time, otherwise null</param>

/// <param name="size">The size of the time otherwise 'small'</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for label, otherwise null</param>

/// <param name="includeValidation">True to add validation otherwise false (no validation)</param>

/// <param name="validationText">Validation override message otherwise null</param>

/// <param name="validationHtmlAttrs">HTML attributes for the validation, otherwise null</param>

/// <returns>HTML input elements for a time picker and optional controls (label, validation)</returns>

public static MvcHtmlString SgTimepickerFor<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, TProperty>> expression, object dataAttrs,

object htmlAttrs = null, string size = "small",

bool includeLabel = true, string labelText = null, object labelDataAttrs = null,

object labelHtmlAttrs = null,

bool includeValidation = true, string validationText = null, object validationHtmlAttrs = null)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.TimeBegin |
| dataAttrs | ***Required***. The knockout data attributes for the time specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { @sagevalue = "Data.TimeBeginTimeSpan", @sagedisable = "ValidTimesSectionDisabled" }  Note: If specified, the sagedisable attribute will be copied to the disable attribute.  Note: valueUpdate will default to “’input’” and does not need to be specified unless another value for valueUpdate is required. |
| htmlAttrs | ***Optional***. The html attributes for the time specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. * “@class” will internally assign “ timepicker {size}” and therefore does not need to be assigned externally. | htmlAttrs: new { id = "txtTimeBegin"} |
| size | ***Optional.*** The size for the dropdown. Defaults to “small”.   * default, small, smaller, xsmall, large, larger, xlarge, medium, mediumlarge | size: “default” |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { sagedisable = "IsTimeDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "Address\_StartTime", @class = “required” } |
| includeValidation | ***Optional.*** True to add validation otherwise no validation will be added. Defaults to true. | includeValidation: false |
| validationText | ***Optional.*** If a validation is included, the validation text will be assigned from the attribute of the expression’s property unless being overridden by this parameter | validationText: ReceiptResx.ValidCurrency |
| validationHtmlAttrs | ***Optional***. The html attributes for the validation specified in an anonymous object or a RouteValueDictionary. | validationHtmlAttrs: new { @class = “vlidation-bottom” } |

* + 1. Individual Helpers Example

<div class="timepicker-group">

@Html.KoSagetextBoxFor(m => m.Data.TimeBegin,

new { @sagevalue = "Data.TimeBeginTimeSpan", @valueUpdate = "'input'",

@sagedisable = "ValidTimesSectionDisabled" },

new { @id = "txtTimeBegin", @class = "timepicker small" })

@Html.ValidationMessageFor(m => m.Data.TimeBegin)

</div>

* + 1. New Composite Helper Example

<div class="timepicker-group">

@Html.SgTimepickerFor(m => m.Data.TimeBegin,

new { @sagevalue = "Data.TimeBeginTimeSpan", @sagedisable = "ValidTimesSectionDisabled" },

new { @id = "txtTimeBegin" }, includeLabel: false)

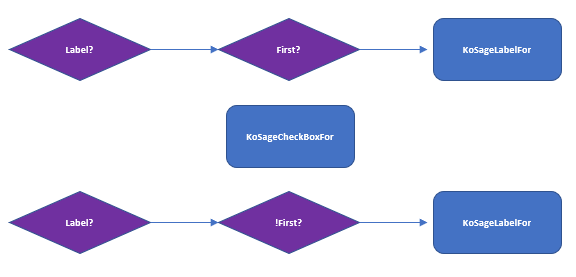
</div>

* Do not need to **small** for class attribute as helper defaults to this value in the size parameter.
* Do not need to specify **timepicker** for class attribute as helper defaults this value.
* Do not need to specify valueUpdate attribute as helper defaults this value.
* Need to specify **includeLabel** parameter to override default value of helper.
  1. SgCheckboxFor

The ***SgCheckboxFor*** helper will create a checkbox and an optional label.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that the checkbox will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that the label will be displayed after the checkbox.
  + 1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label and a checkbox

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="dataAttrs">Data attributes for the checkbox</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for the checkbox, otherwise null</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for label, otherwise null</param>

/// <param name="labelFirst">True to display label first otherwise display last</param>

/// <returns>HTML input elements for a dropdown and optional control (label)</returns>

public static MvcHtmlString SgCheckboxFor<TModel>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, bool>> expression, object dataAttrs,

object htmlAttrs = null, bool includeLabel = true, string labelText = null,

object labelDataAttrs = null, object labelHtmlAttrs = null, bool labelFirst = false)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.Sunday |
| dataAttrs | ***Required***. The knockout data attributes for the checkbox specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { @sagechecked = "Data.Sunday", @sagedisable = "ValidTimesSectionDisabled" }  Note: If specified, the sagedisable attribute will be copied to the disable attribute. |
| htmlAttrs | ***Optional***. The html attributes for the checkbox specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. | new { @id = "chkSunday" }  Note: name will default to the id of the checkbox and does not need to be specified unless another value for name is required. |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { @sagedisable = "ValidTimesSectionDisabled" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "lblDescription", @class = “required” } |
| labelFirst | ***Optional.*** True to add label before checkbox otherwise adds label after checkbox. Defaults to false. | labelFirst: true |

* + 1. Individual Helpers Example

<div class="ctrl-group">

<div class="child">

@Html.KoSageCheckBoxFor(m => m.Data.SortByTransactionType,

new { @sagechecked = "Data.SortByTransactionType",

@sagedisable = "HideShowOptionsSortByTransactionTypeChk" },

new { @id = "Data\_SortByTransactionType", @autofocus = "autofocus" })

@Html.SageLabelFor(m => m.Data.SortByTransactionType, new { @class = "" })

</div>

</div>

* + 1. New Composite Helper Example

<div class="ctrl-group">

<div class="child">

@Html.SgCheckboxFor(m => m.Data.SortByTransactionType,

new { @sagechecked = "Data.SortByTransactionType",

@sagedisable = "HideShowOptionsSortByTransactionTypeChk" })

</div>

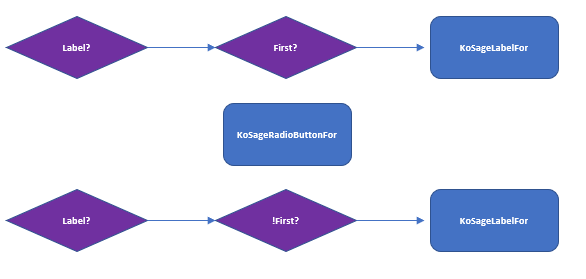
</div>

* Do not need to **id** for as helper defaults to this value.
* Note: autofocus properly is widely overused for checkboxes and in most cases is not required, but if it is required, it can be specified in the html attributes
  1. SgRadioButtonFor

The ***SgRadioButtonFor*** helper will create a radio button and an optional label.

The interface requires a minimal number of parameters to be specified for default behaviour:

* The default is that a label will be created.
  + It will be named “lbl” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that the radio button be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
* The default is that the label will be displayed after the radio button.
  + 1. Flow



* + 1. Interface

/// <summary>

/// Returns an optional label and a radio button

/// </summary>

/// <typeparam name="TModel">The type of the model</typeparam>

/// <param name="htmlHelper">The HTML helper instance that this method extends</param>

/// <param name="expression">An expression that identifies the properties to render.</param>

/// <param name="value">The value of the selected radio button</param>

/// <param name="dataAttrs">Data attributes for the radio button</param>

/// <param name="htmlAttrs">HTML attributes (i.e., id, class) for button, otherwise null</param>

/// <param name="includeLabel">True to add label otherwise false (no label)</param>

/// <param name="labelText">Label text (non-bound) otherwise null</param>

/// <param name="labelDataAttrs">Data attributes for the label, otherwise null</param>

/// <param name="labelHtmlAttrs">HTML attributes (i.e., id, class) for label, otherwise null</param>

/// <param name="labelFirst">True to display label first otherwise display last</param>

/// <returns>HTML input elements for a radio button and optional control (label)</returns>

public static MvcHtmlString SgRadioButtonFor<TModel>(this HtmlHelper<TModel> htmlHelper,

Expression<Func<TModel, object>> expression, object value, object dataAttrs,

object htmlAttrs = null, bool includeLabel = true, string labelText = null,

object labelDataAttrs = null, object labelHtmlAttrs = null, bool labelFirst = false)

* + 1. Parameters

|  |  |  |
| --- | --- | --- |
| parameter | Description | Example |
| expression | ***Required***. The expression. | model => model.Data.TransactionTypeChecked |
| dataAttrs | ***Required***. The knockout data attributes for the button specified in an anonymous object or a RouteValueDictionary.   * sagedisable, sageenable, sageDataPicker, sagetext, sagevalue, SagekendoGrid, Amount, sagechecked, sagevisible, sageoverridden, value, valueUpdate, visible­, disable | new { @sagechecked = "Data.TransactionTypeChecked" }  Note: If specified, the sagedisable attribute will be copied to the disable attribute. |
| Value | ***Required.*** The value of the selected radio button | 1 |
| htmlAttrs | ***Optional***. The html attributes for the checkbox specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “Data\_”+property requires overriding. | new { @id = " OptMaterialUsage " }  Note: name will default to the id of the radio button and does not need to be specified unless another value for name is required. |
| includeLabel | ***Optional.*** True to add label otherwise no label will be added. Defaults to true. | includeLabel: false |
| labelText | ***Optional.*** If a label is included, the label text will be assigned from the expression’s property unless being overridden by this parameter | labelText: ReceiptResx.Currency |
| labelDataAttrs | ***Optional***. The knockout data attributes for the label specified in an anonymous object or a RouteValueDictionary. See dataAttrs for value possibilities. | htmlAttrs: new { @sagedisable = "Data.IsMaterialUsageDisable" } |
| labelHtmlAttrs | ***Optional***. The html attributes for the label specified in an anonymous object or a RouteValueDictionary.   * “id” can be specified when the default of “lbl”+property requires overriding. | labelHtmlAttrs: new { id = "lblDescription", @class = “required” } |
| labelFirst | ***Optional.*** True to add label before checkbox otherwise adds label after checkbox. Defaults to false. | labelFirst: true |

* + 1. Individual Helpers Example

<div class="child block">

@Html.KoSageRadioButtonFor(m => m.Data.BatchTypeSwitch, (int)PostBatchTypeSwitch.Invoice,

new { @sagechecked = "Data.BatchTypeSwitch", @sagedisable = "isInvoiceDisable" },

new { @id = "radInvoice" })

@Html.SageLabel("BatchListing\_Invoice1",APCommonResx.Invoice1,

new { @class = "", @for = "radInvoice" })

</div>

<div class="child block">

@Html.KoSageRadioButtonFor(m => m.Data.BatchTypeSwitch, (int)PostBatchTypeSwitch.Payment,

new { @sagechecked = "Data.BatchTypeSwitch", @sagedisable = "isPaymentDisable" },

new { @id = "repayment" })

@Html.SageLabel("BatchListing\_Payment",APCommonResx.Payment,

new { @class = "", @for = "radPayment" })

</div>

* + 1. New Composite Helper Example

<div class="child block">

@Html.SgRadioButtonFor(m => m.Data.BatchTypeSwitch, (int)PostBatchTypeSwitch.Invoice,

new { @sagechecked = "Data.BatchTypeSwitch", @sagedisable = "isInvoiceDisable" },

new { @id = "radInvoice" },

labelHtmlAttrs: new { @id = "BatchListing\_Invoice1" }, labelText: APCommonResx.Invoice1)

</div>

<div class="child block">

@Html.SgRadioButtonFor(m => m.Data.BatchTypeSwitch, (int)PostBatchTypeSwitch.Payment,

new { @sagechecked = "Data.BatchTypeSwitch", @sagedisable = " isPaymentDisable" },

new { @id = "repayment " },

labelHtmlAttrs: new { @id = “BatchListing\_Payment" }, labelText: APCommonResx.Payment)

</div>

* Need **id** for label and checkbox to override helper defaults.
* No need to specify for attribute as helper defaults this value.