Sage 300 Web Screens SDK

Html Helpers

July 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 4](#_Toc76557728)

[2. Composite Extensions 5](#_Toc76557729)

[2.1 SgNumericFor 6](#_Toc76557730)

[2.1.1 Interface 7](#_Toc76557731)

[2.1.2 Parameters 7](#_Toc76557732)

[2.1.3 Example 9](#_Toc76557733)

[2.2 SgTextFor 10](#_Toc76557734)

[2.2.1 Interface 11](#_Toc76557735)

[2.2.2 Parameters 11](#_Toc76557736)

[2.2.3 Example 13](#_Toc76557737)

[2.3 SgFinderFor 14](#_Toc76557738)

[2.3.1 Interface 15](#_Toc76557739)

[2.3.2 Parameters 15](#_Toc76557740)

[2.3.3 Example 18](#_Toc76557741)

[2.4 SgPasswordFor 19](#_Toc76557742)

[2.4.1 Interface 20](#_Toc76557743)

[2.4.2 Parameters 20](#_Toc76557744)

[2.4.3 Example 22](#_Toc76557745)

[2.5 SgDropdownFor 23](#_Toc76557746)

[2.5.1 Interface 24](#_Toc76557747)

[2.5.2 Parameters 24](#_Toc76557748)

[2.5.3 Example 26](#_Toc76557749)

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. 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. Example

<div class="numeric-group">

@Html.SgNumericFor(model => model.Data.StoredQuantity,

new { sagevalue = "Data.StoredQuantity", sagedisable = "true" },

new { id = "nbrStoredQuantity" })

</div>

* 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. 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. Example

<div class="input-group">

@Html.SgTextFor(model => model.Data.BillingCurrency,

new { sagevalue = "Data.BillingCurrency", sagedisable = "true" },

new { id = "txtExtendedBillingAmountCurrency" }, size: "xsmall",

labelHtmlAttrs: new { @id = "lblExtendedBillingAmountCurrency" })

</div>

* 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. 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. Example

<div class="search-group">

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

new { @sagevalue = "Data.Project", @sagedisable = "IsProjectDisabled" },

new { @id = "txtProject" }, labelText: Model.ProjectLabel,

buttonDataAttrs: new { @sagedisable = "IsProjectDisabled" })

</div>

* 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. 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. Example

<div class="input-group">

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

new { @sagevalue = "Data.Password", @sagedisable = "PasswordDisabled" },

new { @id = "txtPassword", @class = "txt-upper", @maxlength = 64 }, size: "large",

labelHtmlAttrs: new { @class = "required" })

</div>

* 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 textbox will be named “Data\_” + propertyName.
  + It can be overridden with the “id” html attribute.
    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. Example

<div class="dropdown-group">

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

new { @options = "JobRoles", @value = "Data.JobRole" },

new { @id = "Data\_JobRole", @class = "single-select" }, size: "medium")

</div>