System.Web.Mvc Namespace

Reference

Classes

AcceptVerbsAttribute	Represents an attribute that specifies which HTTP verbs an action method will respond to.
ActionDescriptor	Provides information about an action method, such as its name, controller, parameters, attributes, and filters.
ActionExecutedContext	Provides the context for the ActionExecuted method of the ActionFilterAttribute class.
ActionExecutingContext	Provides the context for the ActionExecuting method of the ActionFilterAttribute class.
ActionFilterAttribute	Represents the base class for filter attributes.
ActionMethodSelectorAttribute	Represents an attribute that is used to influence the selection of an action method.
ActionNameAttribute	Represents an attribute that is used for the name of an action.
ActionNameSelectorAttribute	Represents an attribute that affects the selection of an action method.
ActionResult	Represents the result of an action method.
AdditionalMetadataAttribute	Provides a class that implements the IMetadataAware interface in order to support additional metadata.
AjaxHelper	Represents support for rendering HTML in AJAX scenarios within a view.
AjaxHelper<TModel>	Represents support for rendering HTML in AJAX scenarios within a strongly typed view.
AjaxRequestExtensions	Represents a class that extends the HttpRequestBase class by adding the ability to determine whether an HTTP request is an AJAX request.
AllowAnonymousAttribute	Represents an attribute that marks controllers and actions to skip the AuthorizeAttribute during authorization.
AllowHtmlAttribute	Allows a request to include HTML markup during model binding by skipping request validation for the property. (It is strongly recommended that your application explicitly check all models where you disable request validation in order to prevent script exploits.)
AreaRegistration	Provides a way to register one or more areas in an ASP.NET MVC application.
AreaRegistrationContext	Encapsulates the information that is required in order to register an area within an ASP.NET MVC application.

AssociatedMetadataProvider	Provides an abstract class to implement a metadata provider.
AssociatedValidatorProvider	Provides an abstract class for classes that implement a validation provider.
AsyncController	Provided for backward compatibility with ASP.NET MVC 3.
AsyncTimeoutAttribute	Represents an attribute that is used to set the timeout value, in milliseconds, for an asynchronous method.
AuthorizationContext	Encapsulates the information that is required for using an AuthorizeAttribute attribute.
AuthorizeAttribute	Specifies that access to a controller or action method is restricted to users who meet the authorization requirement.
BindAttribute	Represents an attribute that is used to provide details about how model binding to a parameter should occur.
BuildManagerCompiledView	Represents the base class for views that are compiled by the BuildManager class before being rendered by a view engine.
BuildManagerViewEngine	Provides a base class for view engines.
ByteArrayModelBinder	Maps a browser request to a byte array.
CachedAssociatedMetadataProvider<TModelMetadata>	Provides an abstract class to implement a cached metadata provider.
CachedDataAnnotationsMetadataAttributes	Provides a container to cache System.ComponentModel.DataAnnotations attributes.
CachedDataAnnotationsModelMetadata	Provides a container to cache DataAnnotationsModelMetadata.
CachedDataAnnotationsModelMetadataProvider	Implements the default cached model metadata provider for ASP.NET MVC.
CachedModelMetadata<TPrototypeCache>	Provides a container for cached metadata.
CancellationTokenModelBinder	Provides a mechanism to propagates notification that model binder operations should be canceled.
ChildActionOnlyAttribute	Represents an attribute that is used to indicate that an action method should be called only as a child action.
ChildActionValueProvider	Represents a value provider for values from child actions.
ChildActionValueProviderFactory	Represents a factory for creating value provider objects for child actions.
ClientDataTypeModelValidatorProvider	Returns the client data-type model validators.
CompareAttribute	Provides an attribute that compares two properties of a model.

ContentResult	Represents a user-defined content type that is the result of an action method.
Controller	Provides methods that respond to HTTP requests that are made to an ASP.NET MVC Web site.
ControllerActionInvoker	Represents a class that is responsible for invoking the action methods of a controller.
ControllerBase	Represents the base class for all MVC controllers.
ControllerBuilder	Represents a class that is responsible for dynamically building a controller.
ControllerContext	Encapsulates information about an HTTP request that matches specified RouteBase and ControllerBase instances.
ControllerDescriptor	Encapsulates information that describes a controller, such as its name, type, and actions.
ControllerInstanceFilterProvider	Adds the controller to the FilterProviderCollection instance.
CustomModelBinderAttribute	Represents an attribute that invokes a custom model binder.
DataAnnotationsModelMetadata	Provides a container for common metadata, for the DataAnnotationsModelMetadataProvider class, and for the DataAnnotationsModelValidator class for a data model.
DataAnnotationsModelMetadata Provider	Implements the default model metadata provider for ASP.NET MVC.
DataAnnotationsModelValidator	Provides a model validator.
DataAnnotationsModel Validator<TAttribute>	Provides a model validator for a specified validation type.
DataAnnotationsModelValidator Provider	Implements the default validation provider for ASP.NET MVC.
DataErrorInfoModelValidator Provider	Provides a container for the error-information model validator.
DefaultControllerFactory	Represents the controller factory that is registered by default.
DefaultModelBinder	Maps a browser request to a data object. This class provides a concrete implementation of a model binder.
DefaultViewLocationCache	Represents a memory cache for view locations.
DependencyResolver	Provides a registration point for dependency resolvers that implement IDependencyResolver or the Common Service Locator IServiceLocator interface.
DependencyResolverExtensions	Provides a type-safe implementation of GetService(Type) and GetServices(Type).

DictionaryValueProvider<TValue>	Represents the base class for value providers whose values come from a collection that implements the IDictionary<TKey,TValue> interface.
EmptyModelMetadataProvider	Provides an empty metadata provider for data models that do not require metadata.
EmptyModelValidatorProvider	Provides an empty validation provider for models that do not require a validator.
EmptyResult	Represents a result that does nothing, such as a controller action method that returns nothing.
ExceptionContext	Provides the context for using the HandleErrorAttribute class.
ExpressionHelper	Provides a helper class to get the model name from an expression.
FieldValidationMetadata	Provides a container for client-side field validation metadata.
FileContentResult	Sends the contents of a binary file to the response.
FilePathResult	Sends the contents of a file to the response.
FileResult	Represents a base class that is used to send binary file content to the response.
FileStreamResult	Sends binary content to the response by using a Stream instance.
Filter	Represents a metadata class that contains a reference to the implementation of one or more of the filter interfaces, the filter's order, and the filter's scope.
FilterAttribute	Represents the base class for action and result filter attributes.
FilterAttributeFilterProvider	Defines a filter provider for filter attributes.
FilterInfo	Encapsulates information about the available action filters.
FilterProviderCollection	Represents the collection of filter providers for the application.
FilterProviders	Provides a registration point for filters.
FormCollection	Contains the form value providers for the application.
FormContext	Encapsulates information that is required in order to validate and process the input data from an HTML form.
FormValueProvider	Represents a value provider for form values that are contained in a NameValueCollection object.
FormValueProviderFactory	Represents a class that is responsible for creating a new instance of a form-value provider object.
GlobalFilterCollection	Represents a class that contains all the global filters.
GlobalFilters	Represents the global filter collection.

HandleErrorAttribute	Represents an attribute that is used to handle an exception that is thrown by an action method.
HandleErrorInfo	Encapsulates information for handling an error that was thrown by an action method.
HiddenInputAttribute	Represents an attribute that is used to indicate whether a property or field value should be rendered as a hidden input element.
HtmlHelper	Supports the rendering of HTML controls in a view.
HtmlHelper<TModel>	Represents support for rendering HTML controls in a strongly typed view.
HttpAntiForgeryException	This type/member supports the .NET Framework infrastructure and is not intended to be used directly from your code.
HttpDeleteAttribute	Represents an attribute that is used to restrict an action method so that the method handles only HTTP DELETE requests.
HttpFileCollectionValueProvider	Represents a value provider to use with values that come from a collection of HTTP files.
HttpFileCollectionValueProviderFactory	Represents a class that is responsible for creating a new instance of an HTTP file collection value provider object.
HttpGetAttribute	Represents an attribute that is used to restrict an action method so that the method handles only HTTP GET requests.
HttpHeadAttribute	Specifies that the HTTP request must be the HTTP HEAD method.
HttpNotFoundResult	Defines an object that is used to indicate that the requested resource was not found.
HttpOptionsAttribute	Represents an attribute that is used to restrict an action method so that the method handles only HTTP OPTIONS requests.
HttpPatchAttribute	Represents an attribute that is used to restrict an action method so that the method handles only HTTP PATCH requests.
HttpPostAttribute	Represents an attribute that is used to restrict an action method so that the method handles only HTTP POST requests.
HttpPostedFileBaseModelBinder	Binds a model to a posted file.
HttpPutAttribute	Represents an attribute that is used to restrict an action method so that the method handles only HTTP PUT requests.
HttpRequestExtensions	Extends the HttpRequestBase class that contains the HTTP values that were sent by a client during a Web request.
HttpStatusCodeResult	Provides a way to return an action result with a specific HTTP response status code and description.
HttpUnauthorizedResult	Represents the result of an unauthorized HTTP request.

JavaScriptResult	Sends JavaScript content to the response.
JQueryFormValueProvider	The JQuery Form Value provider is used to handle JQuery formatted data in request Forms.
JQueryFormValueProviderFactory	Provides the necessary ValueProvider to handle JQuery Form data.
JsonResult	Represents a class that is used to send JSON-formatted content to the response.
JsonValueProviderFactory	Enables action methods to send and receive JSON-formatted text and to model-bind the JSON text to parameters of action methods.
LinqBinaryModelBinder	Maps a browser request to a LINQ Binary object.
MaxLengthAttributeAdapter	Provides an adapter for the MaxLengthAttribute attribute.
MinLengthAttributeAdapter	Provides an adapter for the MinLengthAttribute attribute.
ModelBinderAttribute	Represents an attribute that is used to associate a model type to a model-builder type.
ModelBinderDictionary	Represents a class that contains all model binders for the application, listed by binder type.
ModelBinderProviderCollection	No content here will be updated; please do not add material here.
ModelBinderProviders	Provides a container for model binder providers.
ModelBinders	Provides global access to the model binders for the application.
ModelBindingContext	Provides the context in which a model binder functions.
ModelClientValidationEqualToRule	This type/member supports the .NET Framework infrastructure and is not intended to be used directly from your code.
ModelClientValidationRangeRule	This type/member supports the .NET Framework infrastructure and is not intended to be used directly from your code.
ModelClientValidationRegexRule	This type/member supports the .NET Framework infrastructure and is not intended to be used directly from your code.
ModelClientValidationRemoteRule	Represents the remote rule for the validation of the model client.
ModelClientValidationRequiredRule	Represents the required rule for the validation of the model client.
ModelClientValidationRule	This type/member supports the .NET Framework infrastructure and is not intended to be used directly from your code.
ModelClientValidationStringLengthRule	This type/member supports the .NET Framework infrastructure and is not intended to be used directly from your code.Represents a length of the validation rule of the model client.
ModelError	Represents an error that occurs during model binding.

ModelErrorCollection	A collection of ModelError instances.
ModelMetadata	Provides a container for common metadata, for the ModelMetadataProvider class, and for the ModelValidator class for a data model.
ModelMetadataProvider	Provides an abstract base class for a custom metadata provider.
ModelMetadataProviders	Provides a container for the current ModelMetadataProvider instance.
ModelState	Encapsulates the state of model binding to a property of an action-method argument, or to the argument itself.
ModelStateDictionary	Represents the state of an attempt to bind a posted form to an action method, which includes validation information.
ModelValidationResult	Provides a container for a validation result.
ModelValidator	Provides a base class for implementing validation logic.
ModelValidatorProvider	Provides a list of validators for a model.
ModelValidatorProviderCollection	No content here will be updated; please do not add material here.
ModelValidatorProviders	Provides a container for the current validation provider.
MultiSelectList	Represents a list of items that users can select more than one item from.
MvcFilter	When implemented in a derived class, provides a metadata class that contains a reference to the implementation of one or more of the filter interfaces, the filter's order, and the filter's scope.
MvcHandler	Selects the controller that will handle an HTTP request.
MvcHtmlString	Represents an HTML-encoded string that should not be encoded again.
MvcHttpHandler	Verifies and processes an HTTP request.
MvcRouteHandler	Creates an object that implements the IHttpHandler interface and passes the request context to it.
MvcWebRazorHostFactory	Creates instances of System.Web.Mvc.MvcWebPageRazorHost files.
NameValueCollectionExtensions	Extends a NameValueCollection object so that the collection can be copied to a specified dictionary.
NameValueCollectionValue Provider	Represents the base class for value providers whose values come from a NameValueCollection object.
NoAsyncTimeoutAttribute	Provides a convenience wrapper for the AsyncTimeoutAttribute attribute.
NonActionAttribute	Represents an attribute that is used to indicate that a controller method is not an action method.
OutputCacheAttribute	Represents an attribute that is used to mark an action method whose

	output will be cached.
OverrideActionFiltersAttribute	Represents the attributes associated with the override filter.
OverrideAuthenticationAttribute	Represents the attributes associated with the authentication.
OverrideAuthorizationAttribute	Represents the attributes associated with the authorization.
OverrideExceptionFiltersAttribute	Represents the attributes associated with the exception filter.
OverrideResultFiltersAttribute	Represents the attributes associated with the result filter.
ParameterBindingInfo	Encapsulates information for binding action-method parameters to a data model.
ParameterDescriptor	Contains information that describes a parameter.
PartialViewResult	Represents a base class that is used to send a partial view to the response.
PreApplicationStartCode	Provides a registration point for ASP.NET Razor pre-application start code.
QueryStringValueProvider	Represents a value provider for query strings that are contained in a NameValueCollection object.
QueryStringValueProviderFactory	Represents a class that is responsible for creating a new instance of a query-string value-provider object.
RangeAttributeAdapter	Provides an adapter for the RangeAttribute attribute.
RazorView	Represents the class used to create views that have Razor syntax.
RazorViewEngine	Represents a view engine that is used to render a Web page that uses the ASP.NET Razor syntax.
RedirectResult	Controls the processing of application actions by redirecting to a specified URI.
RedirectToRouteResult	Represents a result that performs a redirection by using the specified route values dictionary.
ReflectedActionDescriptor	Contains information that describes a reflected action method.
ReflectedControllerDescriptor	Contains information that describes a reflected controller.
ReflectedParameterDescriptor	Contains information that describes a reflected action-method parameter.
RegularExpressionAttributeAdapter	Provides an adapter for the RegularExpressionAttribute attribute.
RemoteAttribute	Provides an attribute that uses the jQuery validation plug-in remote validator.
RequiredAttributeAdapter	Provides an adapter for the RequiredAttributeAttribute attribute.

RequireHttpsAttribute	Represents an attribute that forces an unsecured HTTP request to be re-sent over HTTPS.
ResultExecutedContext	Provides the context for the OnResultExecuted(ResultExecutedContext) method of the ActionFilterAttribute class.
ResultExecutingContext	Provides the context for the OnResultExecuting(ResultExecutingContext) method of the ActionFilterAttribute class.
RouteAreaAttribute	Defines the area to set for all the routes defined in this controller.
RouteAttribute	Place on a controller or action to expose it directly via a route. When placed on a controller, it applies to actions that do not have any System.Web.Mvc.RouteAttribute's on them.
RouteCollectionAttributeRoutingExtensions	Provides routing extensions for route collection attribute.
RouteCollectionExtensions	Extends a RouteCollection object for MVC routing.
RouteDataValueProvider	Represents a value provider for route data that is contained in an object that implements the IDictionary<TKey,TValue> interface.
RouteDataValueProviderFactory	Represents a factory for creating route-data value provider objects.
RoutePrefixAttribute	Annotates a controller with a route prefix that applies to all actions within the controller.
SelectList	Represents a list that lets users select one item.
SelectListGroup	Represents the optgroup HTML element and its attributes. In a select list, multiple groups with the same name are supported. They are compared with reference equality.
SelectListItem	Represents the selected item in an instance of the SelectList class.
SessionStateAttribute	Specifies the session state of the controller.
SessionStateTempDataProvider	Provides session-state data to the current TempDataDictionary object.
StringLengthAttributeAdapter	Provides an adapter for the StringLengthAttribute attribute.
TagBuilder	Contains classes and properties that are used to create HTML elements. This class is used to write helpers, such as those found in the System.Web.Helpers namespace.
TempDataDictionary	Represents a set of data that persists only from one request to the next.
TemplateInfo	Encapsulates information about the current template context.
UrlHelper	Contains methods to build URLs for ASP.NET MVC within an application.
UrlParameter	Represents an optional parameter that is used by the MvcHandler class during routing.
ValidatableObjectAdapter	Provides an object adapter that can be validated.

ValidateAntiForgeryTokenAttribute	Represents an attribute that is used to prevent forgery of a request.
ValidateInputAttribute	Represents an attribute that is used to mark action methods whose input must be validated.
ValueProviderCollection	Represents the collection of value-provider objects for the application.
ValueProviderDictionary	Note: This API is now obsolete.Represents a dictionary of value providers for the application.
ValueProviderFactories	Represents a container for value-provider factory objects.
ValueProviderFactory	Represents a factory for creating value-provider objects.
ValueProviderFactoryCollection	Represents the collection of value-provider factories for the application.
ValueProviderResult	Represents the result of binding a value (such as from a form post or query string) to an action-method argument property, or to the argument itself.
ViewContext	Encapsulates information that is related to rendering a view.
ViewDataDictionary	Represents a container that is used to pass data between a controller and a view.
ViewDataDictionary<TModel>	Represents a container that is used to pass strongly typed data between a controller and a view.
ViewDataInfo	Encapsulates information about the current template content that is used to develop templates and about HTML helpers that interact with templates.
ViewEngineCollection	Represents a collection of view engines that are available to the application.
ViewEngineResult	Represents the result of locating a view engine.
ViewEngines	Represents a collection of view engines that are available to the application.
ViewMasterPage	Represents the information that is needed to build a master view page.
ViewMasterPage<TModel>	Represents the information that is required in order to build a strongly typed master view page.
ViewPage	Represents the properties and methods that are needed to render a view as a Web Forms page.
ViewPage<TModel>	Represents the information that is required in order to render a strongly typed view as a Web Forms page.
ViewResult	Represents a class that is used to render a view by using an IView instance that is returned by an IViewEngine object.
ViewResultBase	Represents a base class that is used to provide the model to the view and then render the view to the response.

ViewStartPage	Provides an abstract class that can be used to implement a view start (master) page.
ViewTemplateUserControl	Provides a container for TemplateInfo objects.
ViewTemplateUser Control<TModel>	Provides a container for TemplateInfo objects.
ViewType	Represents the type of a view.
ViewUserControl	Represents the information that is needed to build a user control.
ViewUserControl<TModel>	Represents the information that is required in order to build a strongly typed user control.
VirtualPathProviderViewEngine	Represents an abstract base-class implementation of the IViewEngine interface.
WebFormView	Represents the information that is needed to build a Web Forms page in ASP.NET MVC.
WebFormViewEngine	Represents a view engine that is used to render a Web Forms page to the response.
WebViewPage	Represents the properties and methods that are needed in order to render a view that uses ASP.NET Razor syntax.
WebViewPage<TModel>	Represents the properties and methods that are needed in order to render a view that uses ASP.NET Razor syntax.

Interfaces

IActionFilter	Defines the methods that are used in an action filter.
IActionInvoker	Defines the contract for an action invoker, which is used to invoke an action in response to an HTTP request.
IActionInvokerFactory	Used to create an IActionInvoker instance for the current request.
IAuthorizationFilter	Defines the methods that are required for an authorization filter.
IClientValidatable	Provides a way for the ASP.NET MVC validation framework to discover at run time whether a validator has support for client validation.
IController	Defines the methods that are required for a controller.
IControllerActivator	Provides fine-grained control over how controllers are instantiated using dependency injection.
IControllerFactory	Defines the methods that are required for a controller factory.
IDependencyResolver	Defines the methods that simplify service location and dependency resolution.

IEnumerableValueProvider	Represents a special IValueProvider that has the ability to be enumerable.
IExceptionFilter	Defines the methods that are required for an exception filter.
IFilterProvider	Provides an interface for finding filters.
IMetadataAware	Provides an interface for exposing attributes to the AssociatedMetadataProvider class.
IMethodInfoActionDescriptor	An optional interface for ActionDescriptor types which provide a MethodInfo.
IModelBinder	Defines the methods that are required for a model binder.
IModelBinderProvider	Defines methods that enable dynamic implementations of model binding for classes that implement the IModelBinder interface.
IMvcFilter	Defines members that specify the order of filters and whether multiple filters are allowed.
IResultFilter	Defines the methods that are required for a result filter.
IRouteWithArea	Associates a route with an area in an ASP.NET MVC application.
ITempDataProvider	Defines the contract for temporary-data providers that store data that is viewed on the next request.
ITempDataProviderFactory	Used to create an ITempDataProvider instance for the controller.
IUnvalidatedValueProvider	Represents an IValueProvider interface that can skip request validation.
IValueProvider	Defines the methods that are required for a value provider in ASP.NET MVC.
IView	Defines the methods that are required for a view.
IViewDataContainer	Defines the methods that are required for a view data dictionary.
IViewEngine	Defines the methods that are required for a view engine.
IViewLocationCache	Defines the methods that are required in order to cache view locations in memory.
IViewPageActivator	Provides fine-grained control over how view pages are created using dependency injection.

Enums

AreaReference	Controls interpretation of a controller name when constructing a RemoteAttribute.
FilterScope	Defines values that specify the order in which ASP.NET MVC filters run within the same filter type and filter order.

FormMethod	Enumerates the HTTP request types for a form.
Html5DateRenderingMode	Enumerates the date rendering mode for HTML5.
HttpVerbs	Enumerates the HTTP verbs.
InputType	Enumerates the types of input controls.
JsonRequestBehavior	Specifies whether HTTP GET requests from the client are allowed.
TagRenderMode	Enumerates the modes that are available for rendering HTML tags.

Delegates

ActionSelector	Represents a delegate that contains the logic for selecting an action method.
DataAnnotationsModelValidationFactory	Represents the method that creates a DataAnnotationsModelValidatorProvider instance.
DataAnnotationsValidatableObjectAdapterFactory	Provides a factory for validators that are based on IValidatableObject.

Feedback

Was this page helpful? Yes No

ValidationExtensions.ValidationSummary Method

Reference

Definition

Namespace: System.Web.Mvc.Html

Assembly: System.Web.Mvc.dll

Package: Microsoft.AspNet.Mvc v5.2.6

Overloads

ValidationSummary(HtmlHelper, String, Object, String)	
ValidationSummary(HtmlHelper, String, IDictionary<String,Object>, String)	
ValidationSummary(HtmlHelper, Boolean, String, String)	
ValidationSummary(HtmlHelper, Boolean, String, Object)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.
ValidationSummary(HtmlHelper, Boolean, String, IDictionary<String,Object>)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.
ValidationSummary(HtmlHelper, String, String)	
ValidationSummary(HtmlHelper, String, Object)	Returns an unordered list (ul element) of validation messages in the ModelStateDictionary object.
ValidationSummary(HtmlHelper, String, IDictionary<String,Object>)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.
ValidationSummary(HtmlHelper, Boolean, String)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.
ValidationSummary(HtmlHelper, String)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.
ValidationSummary(HtmlHelper, Boolean)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-

	level errors.
ValidationSummary(HtmlHelper)	Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.
ValidationSummary(HtmlHelper, Boolean, String, IDictionary<String,Object>, String)	
ValidationSummary(HtmlHelper, Boolean, String, Object, String)	

ValidationSummary(HtmlHelper, String, Object, String)

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, string message, object htmlAttributes,
string headingTag);
```

Parameters

htmlHelper HtmlHelper

message String

htmlAttributes Object

headingTag String

Returns

MvcHtmlString

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, String, IDictionary<String,Object>, String)

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, string message,
System.Collections.Generic.IDictionary<string,object> htmlAttributes, string
headingTag);
```

Parameters

htmlHelper HtmlHelper

message String

htmlAttributes IDictionary<String,Object>

headingTag String

Returns

MvcHtmlString

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean, String, String)

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors, string message,
string headingTag);
```

Parameters

htmlHelper HtmlHelper

excludePropertyErrors Boolean

message String

headingTag String

Returns

MvcHtmlString

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean, String, Object)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors, string message,
object htmlAttributes);
```

Parameters

htmlHelper HtmlHelper

The HTML helper instance that this method extends.

excludePropertyErrors Boolean

true to have the summary display model-level errors only, or false to have the summary display all errors.

message String

The message to display with the validation summary.

htmlAttributes Object

An object that contains the HTML attributes for the element.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean, String, IDictionary<String,Object>)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors, string message,
System.Collections.Generic.IDictionary<string,object> htmlAttributes);
```

Parameters

htmlHelper HtmlHelper

The HTML helper instance that this method extends.

excludePropertyErrors Boolean

true to have the summary display model-level errors only, or false to have the summary display all errors.

message String

The message to display with the validation summary.

htmlAttributes IDictionary<String,Object>

A dictionary that contains the HTML attributes for the element.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, String, String)

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, string message, string headingTag);
```

Parameters

`htmlHelper` HtmlHelper

`message` String

`headingTag` String

Returns

MvcHtmlString

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, String, Object)

Returns an unordered list (ul element) of validation messages in the ModelStateDictionary object.

```
C#

public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, string message, object htmlAttributes);
```

Parameters

`htmlHelper` HtmlHelper

The HTML helper instance that this method extends.

`message` String

The message to display if the specified field contains an error.

`htmlAttributes` Object

An object that contains the HTML attributes for the element.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, String, IDictionary<String,Object>)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

```
C#

public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, string message,
System.Collections.Generic.IDictionary<string,object> htmlAttributes);
```

Parameters

`htmlHelper` HtmlHelper

The HTML helper instance that this method extends.

`message` String

The message to display if the specified field contains an error.

`htmlAttributes` IDictionary<String,Object>

A dictionary that contains the HTML attributes for the element.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean, String)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors, string message);
```

Parameters

`htmlHelper` HtmlHelper

The HTML helper instance that this method extends.

`excludePropertyErrors` Boolean

true to have the summary display model-level errors only, or false to have the summary display all errors.

message String

The message to display with the validation summary.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, String)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, string message);
```

Parameters

htmlHelper HtmlHelper

The HMTL helper instance that this method extends.

message String

The message to display if the specified field contains an error.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object and optionally displays only model-level errors.

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors);
```

Parameters

`htmlHelper` HtmlHelper

The HTML helper instance that this method extends.

`excludePropertyErrors` Boolean

true to have the summary display model-level errors only, or false to have the summary display all errors.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper)

Returns an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper);
```

Parameters

htmlHelper HtmlHelper

The HTML helper instance that this method extends.

Returns

MvcHtmlString

A string that contains an unordered list (ul element) of validation messages.

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean, String, IDictionary<String,Object>, String)

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors, string message,
System.Collections.Generic.IDictionary<string,object> htmlAttributes, string
headingTag);
```

Parameters

htmlHelper HtmlHelper

excludePropertyErrors Boolean

message String

htmlAttributes IDictionary<String,Object>

headingTag String

Returns

MvcHtmlString

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

ValidationSummary(HtmlHelper, Boolean, String, Object, String)

C#

```
public static System.Web.Mvc.MvcHtmlString ValidationSummary (this
System.Web.Mvc.HtmlHelper htmlHelper, bool excludePropertyErrors, string message,
object htmlAttributes, string headingTag);
```

Parameters

htmlHelper HtmlHelper

excludePropertyErrors Boolean

message String

htmlAttributes Object

headingTag String

Returns

MvcHtmlString

Applies to

▼ ASP.NET MVC 5.2

Product	Versions
ASP.NET MVC	5.2

MVC

Preparación ejemplo ModelState

ViewModels / Home / AddUserVM.cs

```
public class AddUserVM
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string EmailAddress { get; set; }
}
```

Vistas / Inicio / Add.cshtml

```
@model ModelStateDemo.ViewModels.Home.AddUserVM
<h2>Add</h2>

@using(Html.BeginForm())
{
    <div>
        <div>
            @Html.TextBoxFor(x => x.FirstName)
        </div>
        <div>
            @Html.TextBoxFor(x => x.LastName)
        </div>
        <div>
            @Html.TextBoxFor(x => x.EmailAddress)
        </div>
        <div>
            <input type="submit" value="Save" />
        </div>
    </div>
}
```

Controladores / HomeController.cs

```
...
[HttpGet]
public ActionResult Add()
{
    AddUserVM model = new AddUserVM();
    return View(model);
}

[HttpPost]
public ActionResult Add(AddUserVM model)
{
    if(!ModelState.IsValid)
    {
        return View(model);
    }
    return RedirectToAction("Index");
}
```

Cuando enviemos el formulario a la acción POST, todos los valores que ingresamos se mostrarán en la instancia AddUserVM.

Last modified: Monday, 1 April 2019, 8:21 AM

MVC

La clase ModelStateDictionary

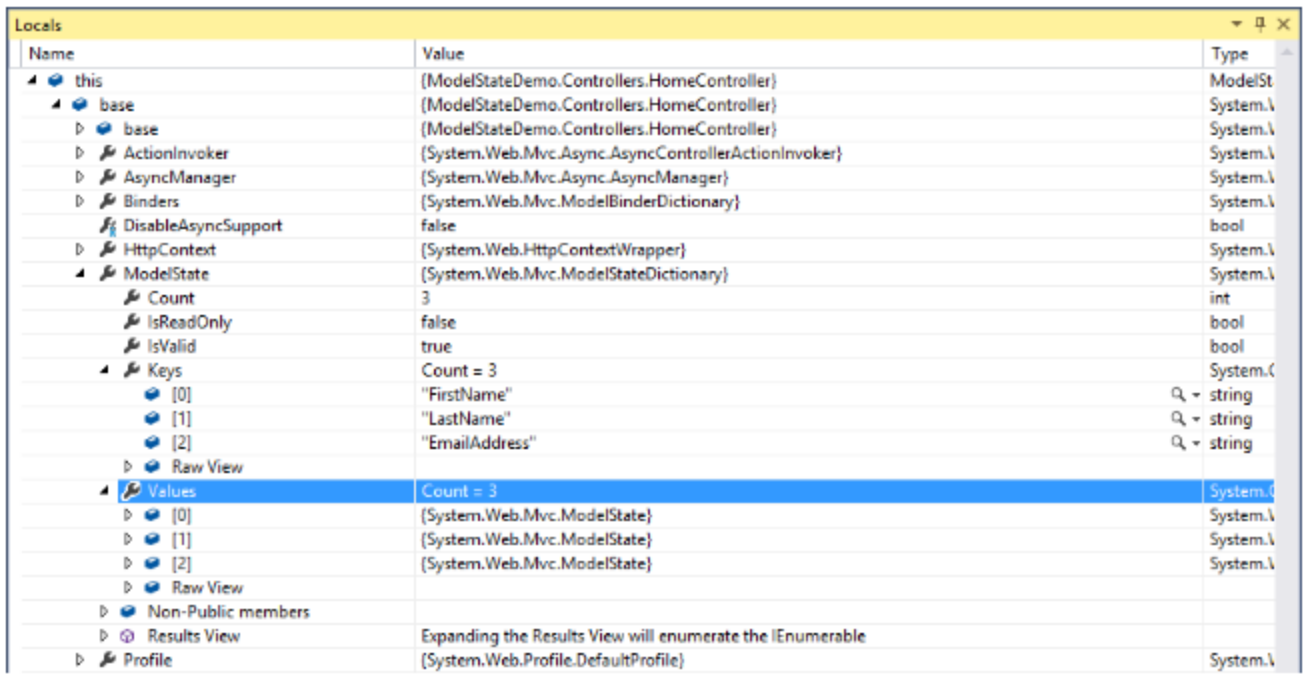
Veamos el formulario HTML representado para la página Agregar:

```
<form action="/Home/Add" method="post">
    <div>
        <div>
            <label for="FirstName">First Name:</label>
            <input id="FirstName" name="FirstName" type="text" value="">
        </div>
        <div>
            <label for="LastName">Last Name:</label>
            <input id="LastName" name="LastName" type="text" value="">
        </div>
        <div>
            <label for="EmailAddress">Email Address:</label>
            <input id="EmailAddress" name="EmailAddress" type="text" value="">
        </div>
        <div>
            <input type="submit" value="Save">
        </div>
    </div>
</form>
```

En un POST, todos los valores de las `<input>` etiquetas se envían al servidor como pares clave-valor.

Cuando MVC recibe un POST, toma todos los parámetros de publicación y los agrega a una instancia de ModelStateDictionary .

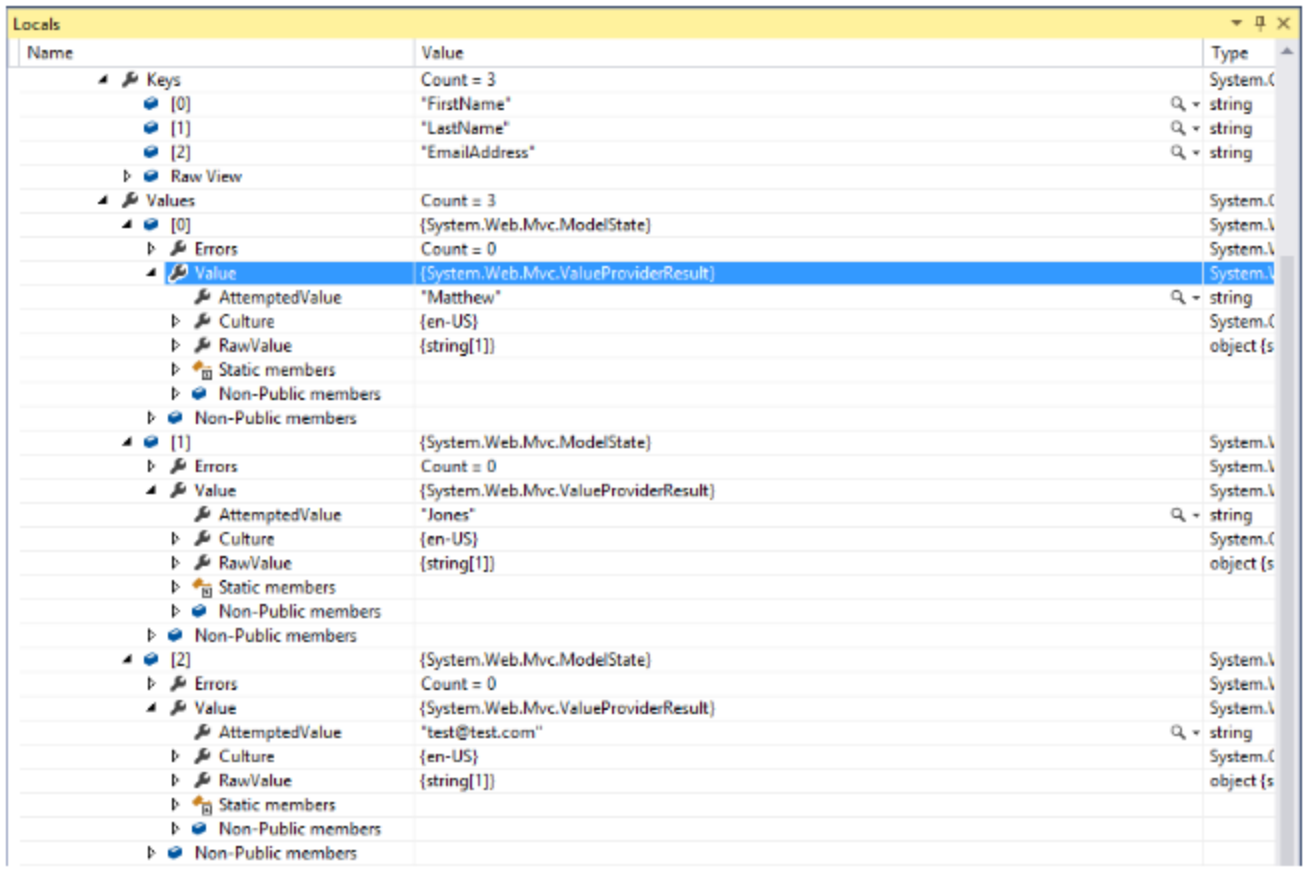
Al depurar la acción POST del controlador en Visual Studio, podemos usar la ventana Locales para investigar este diccionario:



La propiedad Valores del ModelStateDictionary contiene instancias que son del tipo System.Web.Mvc.ModelState . ¿Qué contiene realmente un ModelState?

¿Qué hay en un ModelState?

Aquí es cómo se ven esos valores, desde la misma sesión del depurador:



Cada una de las propiedades tiene una instancia de ValueProviderResult que contiene los valores reales enviados al servidor.

MVC crea todas estas instancias automáticamente para nosotros cuando enviamos un POST con datos, y la acción POST tiene entradas que se asignan a los valores enviados. Esencialmente, MVC está envolviendo las entradas del usuario en clases más amigables con el servidor (ModelState y ValueProviderResult) para un uso más sencillo.

Sin embargo, todavía hay dos propiedades importantes que no hemos discutido: la propiedad ModelState.Errors y la propiedad ModelStateDictionary.IsValid. Se utilizan para la segunda función de ModelState: para almacenar los errores encontrados en los valores enviados.

Last modified: Monday, 1 April 2019, 8:29 AM

◄ Preparación ejemplo ModelState

Jump to...

Errores de validación en ModelState ►

You are logged in as Leandro Sandoval (Log out)
MVC01
Data retention summary
Get the mobile app

ModelStateDictionary Class

Reference

Definition

Namespace: System.Web.Mvc

Assembly: System.Web.Mvc.dll

Package: Microsoft.AspNet.Mvc v5.2.6

Represents the state of an attempt to bind a posted form to an action method, which includes validation information.

C#

```
[System.Serializable]
public class ModelStateDictionary :
System.Collections.Generic.ICollection<System.Collections.Generic.KeyValuePa
ir<string,System.Web.Mvc.ModelState>>,
System.Collections.Generic.IDictionary<string,System.Web.Mvc.ModelState>,
System.Collections.Generic.IEnumerable<System.Collections.Generic.KeyValuePa
ir<string,System.Web.Mvc.ModelState>>
```

Inheritance Object → ModelStateDictionary

Attributes SerializableAttribute

Implements ICollection<KeyValuePair<String,ModelState>>, ICollection<KeyValuePair<TKey,TValue>>, IDictionary<String,ModelState>, IEnumerable<KeyValuePair<String,ModelState>>, IEnumerable<KeyValuePair<TKey,TValue>>, IEnumerable<T>, IEnumerable

Constructors

ModelStateDictionary()	Initializes a new instance of the ModelStateDictionary class.
ModelStateDictionary(ModelStateDictionary)	Initializes a new instance of the ModelStateDictionary class by using values that are copied from the specified model-state

	dictionary.

Properties

Count	Gets the number of key/value pairs in the collection.
IsReadOnly	Gets a value that indicates whether the collection is read-only.
IsValid	Gets a value that indicates whether this instance of the model-state dictionary is valid.
Item[String]	Gets or sets the value that is associated with the specified key.
Keys	Gets a collection that contains the keys in the dictionary.
Values	Gets a collection that contains the values in the dictionary.

Methods

Add(KeyValuePair<String,ModelState>)	Adds the specified item to the model-state dictionary.
Add(String, ModelState)	Adds an element that has the specified key and value to the model-state dictionary.
AddModelError(String, Exception)	Adds the specified model error to the errors collection for the model-state dictionary that is associated with the specified key.
AddModelError(String, String)	Adds the specified error message to the errors collection for the model-state dictionary that is associated with the specified key.
Clear()	Removes all items from the model-state dictionary.
Contains(KeyValuePair<String,ModelState>)	Determines whether the model-state dictionary contains a specific value.
ContainsKey(String)	Determines whether the model-state dictionary contains the specified key.
CopyTo(KeyValuePair<String,ModelState>[], Int32)	Copies the elements of the model-state dictionary to an array, starting at a specified index.
GetEnumerator()	Returns an enumerator that can be used to iterate through the collection.
IsValidField(String)	Determines whether there are any ModelError objects that are associated with or prefixed with the specified key.

Merge(ModelStateDictionary)	Copies the values from the specified ModelStateDictionary object into this dictionary, overwriting existing values if keys are the same.
Remove(KeyValuePair<String,ModelState>)	Removes the first occurrence of the specified object from the model-state dictionary.
Remove(String)	Removes the element that has the specified key from the model-state dictionary.
SetModelValue(String, ValueProviderResult)	Sets the value for the specified key by using the specified value provider dictionary.
TryGetValue(String, ModelState)	Attempts to gets the value that is associated with the specified key.

Explicit Interface Implementations

IEnumerable.GetEnumerator()	Returns an enumerator that can be used to iterate through the collection.

Applies to

Product	Versions
ASP.NET MVC	5.2

Feedback

Was this page helpful? Yes No

MVC

Errores de validación en ModelState

Vamos a cambiar nuestra clase AddUserVM:

```
public class AddUserVM
{
    [Required(ErrorMessage = "Please enter the user's first name.")]
    [StringLength(50, ErrorMessage = "The First Name must be less than {1} characters.")]
    [Display(Name = "First Name:")]
    public string FirstName { get; set; }

    [Required(ErrorMessage = "Please enter the user's last name.")]
    [StringLength(50, ErrorMessage = "The Last Name must be less than {1} characters.")]
    [Display(Name = "Last Name:")]
    public string LastName { get; set; }

    [EmailAddress(ErrorMessage = "The Email Address is not valid")]
    [Required(ErrorMessage = "Please enter an email address.")]
    [Display(Name = "Email Address:")]
    public string EmailAddress { get; set; }
}
```

Hemos agregado atributos de validación, específicamente Requerido , StringLength y EmailAddress .

También hemos establecido los mensajes de error que se mostrarán si se producen los errores de validación correspondientes.

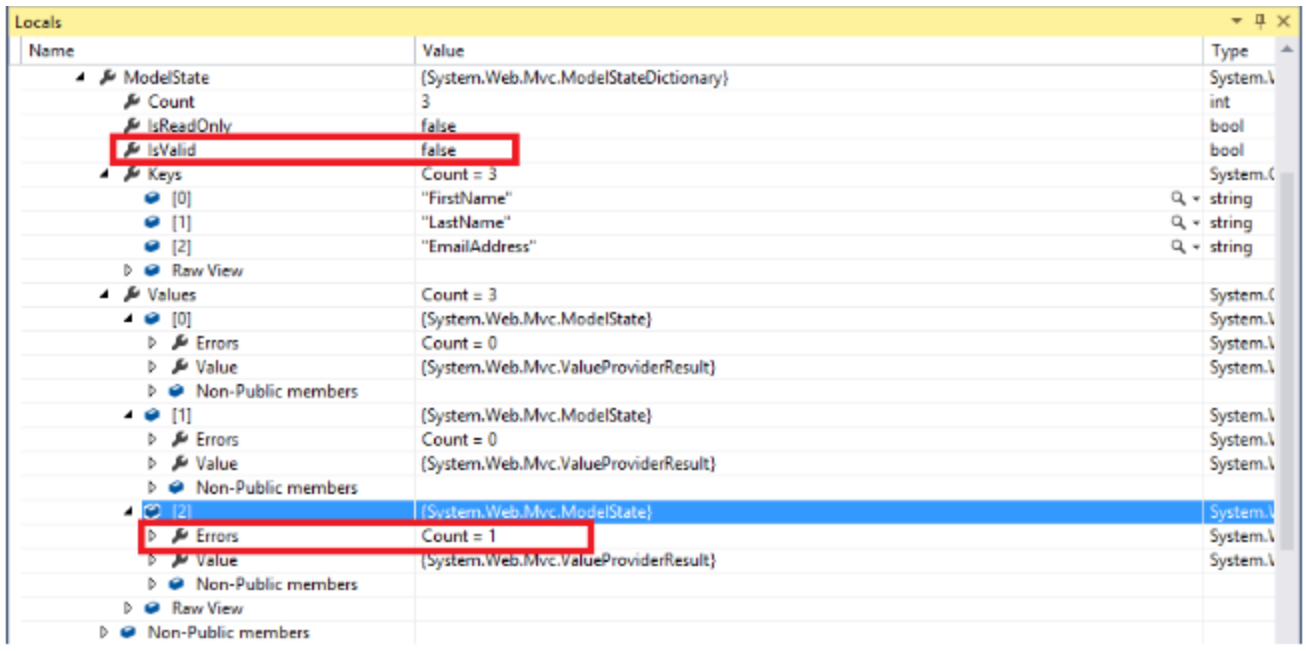
Con los cambios anteriores en su lugar, modifiquemos la vista Agregar para mostrar los mensajes de error si ocurren:

```
@model ModelStateDemo.ViewModels.Home.AddUserVM

<h2>Add</h2>

@using(Html.BeginForm())
{
    @Html.ValidationSummary()
    <div>
        <div>
            @Html.LabelFor(x => x.FirstName)
            @Html.TextBoxFor(x => x.FirstName)
            @Html.ValidationMessageFor(x => x.FirstName)
        </div>
        <div>
            @Html.LabelFor(x => x.LastName)
            @Html.TextBoxFor(x => x.LastName)
            @Html.ValidationMessageFor(x => x.LastName)
        </div>
        <div>
            @Html.LabelFor(x => x.EmailAddress)
            @Html.TextBoxFor(x => x.EmailAddress)
            @Html.ValidationMessageFor(x => x.EmailAddress)
        </div>
        <div>
            <input type="submit" value="Save" />
        </div>
    </div>
}
```

Observe los dos ayudantes que estamos usando ahora, ValidationSummary y ValidationMessageFor .

- ValidationSummary lee todos los errores del estado del modelo y los muestra en una lista con viñetas.
- ValidationMessageFor muestra solo errores para la propiedad especificada.

Veamos qué sucede cuando intentamos enviar un POST no válido que falta la dirección de correo electrónico. Cuando llegamos a la acción POST durante la depuración, tenemos los siguientes valores en nuestro ModelStateDictionary:



Tenga en cuenta que la instancia de ModelState para la dirección de correo electrónico ahora tiene un error en la colección de Errores.

Cuando MVC crea el estado del modelo para las propiedades enviadas, también pasa por cada propiedad en el Modelo de Vista y valida la propiedad usando atributos asociados a ella. Si se encuentra algún error, se agrega a la colección de Errores en el ModelState de la propiedad.

También tenga en cuenta que IsValid es falso ahora. Eso es porque existe un error; IsValid es falso si alguna de las propiedades enviadas tiene algún mensaje de error adjunto.

Lo que todo esto significa es que al configurar la validación de esta manera, permitimos que MVC funcione de la forma en que fue diseñado.

ModelState almacena los valores enviados, permite que se asignen a propiedades de clase (o simplemente como parámetros a la acción) y mantiene una colección de mensajes de error para cada propiedad. En escenarios simples, esto es todo lo que necesitamos, ¡y todo esto está sucediendo detrás de escena!

Last modified: Monday, 1 April 2019, 8:32 AM

◄ La clase ModelStateDictionary

Jump to...

Validación personalizada ►

You are logged in as Leandro Sandoval (Log out)
MVC01
Data retention summary
Get the mobile app

MVC

Validación personalizada

De hecho, podemos añadir errores al estado a través del modelo de AddModelError método de ModelStateDictionary:

```
[HttpPost]
public ActionResult Add(AddUserVM model)
{
    if(model.FirstName == model.LastName)
    {
        ModelState.AddModelError("LastName", "The last name cannot be the same as the first name.");
    }
    if(!ModelState.IsValid)
    {
        return View(model);
    }
    return RedirectToAction("Index");
}
```

El primer parámetro del método AddModelError es el nombre de la propiedad a la que se aplica el error. En este caso, lo configuramos como Apellido. También puede establecerlo en nada (o en un nombre falso) si solo desea que aparezca en el resumen de validación y no en un mensaje de validación.

Last modified: Monday, 1 April 2019, 8:34 AM

◄ Errores de validación en ModelState

Jump to...

ModelState.Clear() ¿Que es? ►

MVC

ModelState.Clear() ¿Que es?

Si obtiene su modelo de un formulario y desea manipular los datos que provienen del formulario del cliente y escribirlos en una vista, debe llamar a ModelState.Clear () para limpiar los valores de ModelState.

El motivo es que, normalmente, desea devolver al cliente el formulario con todos los errores. Entonces, cuando devuelve el parámetro que contiene su modelo a la vista que se devolverá, este usará el valor de ModelState.

Entonces, por ejemplo, si cambio una propiedad y la envío al cliente:

```
[HttpPost]
public ActionResult Edit(MyObject objModel)
{
  objModel.Property1 = "NEW VALUE";
  //...
  return View(objModel)
}
```

Esto no pondrá en la interfaz de usuario el nuevo valor para la propiedad1 **porque los valores de ModelState no contienen este valor sino el que ingresó el usuario.**

Para poder anular el estado del modelo, debe eliminar todos los datos del mismo.

```
[HttpPost]
public ActionResult Edit(MyObject objModel)
{
  objModel.Property1 = "NEW VALUE";
  //...
  ModelState.Clear();
  return View(objModel)
}
```

Para borrar la memoria del estado del modelo, debe usar **ModelState.Clear ()** .También puede eliminar solo el campo deseado utilizando el método de ModelState.

```
ModelState.Remove("Property1");
```

Además, si siempre desea no usar ModelState, es posible que no quiera usar HtmlHelper y directamente use el modelo con código Html.

```
My Property: <input type="text" name="Property1" value="@Model.Property?"
```

En cualquier situación, lo que debe recordar es que ModelState es el mecanismo predeterminado y, por defecto, será el que se usará para mostrar la información al formulario.

Last modified: Monday, 1 April 2019, 1:38 PM

MVC

Resumen

1) El ModelState representa los valores enviados y los errores en dichos valores durante un POST.

2) El proceso de validación respeta los atributos como Required y EmailAddress (y otros), y podemos agregar errores personalizados a la validación si así lo deseamos.

3) Utilizamos las clases **ValidationSummary** and **ValidationMessage** para leer directamente desde ModelState para mostrar los errores al usuario.

4) Cuando MVC crea el estado del modelo para las propiedades enviadas, también pasa por cada propiedad en el Modelo de Vista y valida la propiedad usando atributos asociados a ella. Si se encuentra algún error, se agrega a la colección de Errores en el ModelState de la propiedad.

5) Cada una de las propiedades tiene una instancia de ValueProviderResult que contiene los valores reales enviados al servidor. MVC crea todas estas instancias automáticamente para nosotros cuando enviamos un POST con datos, y la acción POST tiene entradas que se asignan a los valores enviados. Esencialmente, **MVC está envolviendo las entradas del usuario en clases más amigables con el servidor**(ModelState y ValueProviderResult) para un uso más sencillo.

6) Si obtiene su modelo de un formulario y desea manipular los datos que provienen del formulario del cliente y escribirlos en una vista, debe llamar a ModelState.Clear () para limpiar los valores de ModelState.

7) Html.ActionLink puede ser utilizado para navegar a diferentes controladores, con y sin parámetros.

Last modified: Thursday, 4 April 2019, 1:57 PM

◄ ModelState.Clear() ¿Que es?

Jump to...

View() vs RedirectToAction() vs Redirect() ►