What's New in ValidateThis 0.98 - Part II - New Validation Types

Posted At : January 3, 2011 5:08 PM | Posted By : Bob Silverberg
Related Categories: ValidateThis

In addition to a bunch of neat new features, version 0.98 of ValidateThis also included a number of new validation types. One of the most important aspects of the framework, to me anyway, is the fact that it is incredibly extensible and allows you to easily create your own validation types, which is precisely what a number of folks including Adam Drew, Marc Esher and Chris Blackwell did. They were also kind enough to contribute those validation types to the framework so we can all benefit from them. In this post I'll describe each of the types that were added for version 0.98.

Each validation type accepts certain parameters, all of which have been documented in the ValidateThis Online Documentation, so I won't clutter up this post with that information. You can click on the name of each type in this post and be taken directly to its description in the docs if you wish. I'll just concentrate on describing what the validation type tests and suggestions for ways in which it might be useful.

New Validation Types

CollectionSize

The *CollectionSize* type ensures that the contents of a property is of a specific size. On the server this can be a list, struct or array. A client version is still being developed. You can validate whether a collection is of a particular size and/or whether it is at least or at most a particular number of items. For example, a customer could have between 1 and 3 addresses.

DateRange

The *DateRange* type ensures that the contents of a property is a valid date that falls between two dates. Note that you can use the new *expression* parameter type to make the *from* and *to* dates dynamic (e.g., set one to *now()*).

DoesNotContainOtherProperties

The *DoesNotContainOtherProperties* type ensures that the contents of a property does not contain values found in other properties. This can be used, for example, to ensure that a *password* property does not include the user's first or last names.

False

The *False* type ensures that the contents of a property is a value that can be interpreted as *false*. This includes the values *false*,*no* and *0*.

FutureDate

The *FutureDate* type ensures that the contents of a property is a valid date that falls after a particular date. By default it compares the contents of the property to the current date, but you can pass in an optional *after* parameter to validate whether the contents of the property falls after a specific date.

InList

The *InList* type ensures that the contents of a property is one of the values specified in a list. You can pass in the delimiter for the list as an optional parameter.

IsValidObject

The *IsValidObject* type is used to validate properties which contain other objects. For example, say you have a *User* object which has an *address* property which contains an *Address* object. If you have validation rules defined for the *Address* object then you can add a rule to the *address* property of your *User* object of type *IsValidObject* and when you ask ValidateThis to perform server-side validations on your *User* object it will also automatically validate the composed *Address* object and return all of the failures for both objects to you in the Result.

NoHTML

The *NoHTML* type ensures that the contents of a property does not contain HTML.

NotInList

The *NotInList* type ensures that the contents of a property is not one of the values specified in a list. You can pass in the delimiter for the list as an optional parameter.

The *PastDate* type ensures that the contents of a property is a valid date that falls before a particular date. By default it compares the contents of the property to the current date, but you can pass in an optional *before* parameter to validate whether the contents of the property falls before a specific date.

True

The *True* type ensures that the contents of a property is a value that can be interpreted as *true*. This includes the values *true*,*yes* and any number other than *0*.

URL

The *URL* type ensures that the contents of a property is a valid (properly formed) URL.

Expression

The *Expression* type ensures that a cfml expression evaluates to true. This allows you to define any arbitrary cfml expression and have it evaluated in the context of the object being validated.

Creating Your Own Validation Types

If you're interested in creating your own validation types, please check out the Creating New Validation Types page in the online documentation.

As always, the latest code is available from the ValidateThis RIAForge site, and if you have any questions about the framework, or suggestions for enhancements, please send them to theValidateThis Google Group.