ValidateThis 0.99 - Goodies for ColdBox, Debugging and More

Posted At : February 25, 2011 11:33 AM | Posted By : Bob Silverberg
Related Categories: ValidateThis

I promised John Whish that we'd have another version of ValidateThis, an awesome validation framework for ColdFusion, out in time for his presentation at Scotch on the Rocks on March 4th, 2011, and I've never been one to disappoint. So today I announce the release of version 0.99 of ValidateThis.

The lion's share of the work on this release was done by John, Adam Drew and myself. A lot of the work in this release is actually building towards something very cool that will come out later in the year, but there are still some significant enhancements including:

- Goodies for ColdBoxers including a ColdBox interceptor and three new sample applications.
- New debugging features to make it easier to figure out what's going wrong when things aren't working as you expect.
- The ability to ignore a validation client-side.
- Support for onMissingMethod in your objects.
- JavaScript assets are now retrieved from a CDN or written to the browser so you don't need to place JS files in a web accessible folder.
- Updates to the addFailure() method.
- A couple of new validation types were added.

As always, the latest version can be downloaded from the ValidateThis RIAForge site. Details of the enhancements follow:

ColdBox Goodies

Not knowing much about ColdBox myself, I've left this area in the capable hands of John and Adam and they've come up with some pretty cool stuff.

The framework now ships with a ColdBox interceptor that allows you to incorporate VT into your CB app. It will make use of resource bundles that you've already defined in your CB app to do i18n for validations. Demos are available for a simple ColdBox app, a ColdBox app with i18n support, and a ColdBox app that uses a module. Documentation on using VT and CB together can be found on the VT wiki. Note that the ColdBox interceptor deprecates the ColdBox plugins that used to ship with the framework.

Debugging Features

Debugging features were requested by some VT users and we thought they were a good idea so they've been added.

There is a new key in the ValidateThisConfig struct called debuggingMode, which can be set to one of three values:

- *none*, which is the default does not enable any debugging behaviour.
- *info*, turns on debugging for server-side validations and also will log an error if your xml file does not validate against the ValidateThis xsd.
- *strict*, turns on debugging for server-side validations and also will throw an exception if your xml file does not validate against the ValidateThis xsd.

If you use the *info* or *strict* modes, then after performing server-side validations you can see the debugging info by calling the *getDebugging()* method on the *Result* object.

Ignoring Form Fields

A couple of times the issue has come up where a user would like to have some hidden form fields not be validated. We discussed a number of different approaches to this and Adam came up with a fantastic solution. Now, if you want VT to ignore a field on your form you simply add the class *ignore* to it and it will not get validated.

Support for onMissingMethod in Your Objects

The framework can now deal with the use of onMissingMethod in your business objects. You can use onMM for getters, as well as for methods to satisfy a custom validation type or a dynamic parameter.

No Need to Worry About JavaScript Assets

In the past you had to copy certain assets into a web-accessible folder in order to load them into the browser. We are now retrieving jQuery and the jQuery Validation plugin from a CDN, and streaming some additional js that was in a file directly to the browser, so you don't need to copy any files anywhere anymore.

Updates to the API of the addFailure() Method

You can manually add a validation failure in your code by calling the *addFailure()* method on the *Result* object. This method accepted a single argument which was a struct that contained keys to copy into the failure. We've made the API more specific now, so it accepts the following arguments, all of which are optional:

- *failure*, a struct which is used as a basis for the failure. This was kept as the first argument to maintain backwards compatibility. Any values in this struct will override any values set specifically.
- *propertyName*, the name of the property that caused the failure.
- *clientFieldName*, the name of the form field that caused the failure, defaults to the value of *propertyName*.
- *type*, the type of validation that caused the failure.
- *theObject*, the object that was being validated.
- *objectType*, the type of object that was being validated.

In addition to updating the API, when you call *addFailure()* it automatically sets success=false on the *Result* object.

New Validation Types

MinPatternsMatch

The *MinPatternsMatch* type (which was renamed from *Patterns* and was contributed by Marc Esher) ensures that the contents of a property matches a minimum number of patterns. This can be used, for example, to ensure that a password property conforms to certain standards. For example, a password could require at least 3 of the following:

- An uppercase character
- A lowercase character
- A number
- A punctuation character

Time

The *Time* type ensures that the contents of a property is a valid time that falls between 00:00 and 23:59.

Once again, 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. I'd also like to again thank Adam and John for their contributions to the framework.