ValidateThis! 0.75 - Now Supports Internationalization (i18n)

Posted At : June 1, 2009 9:49 PM | Posted By : Bob Silverberg
Related Categories: OO Design, ColdFusion, i18n, ValidateThis

A developer named Mischa Sameli asked a question about supporting internationalization (i18n) in ValidateThis!, my validation framework for ColdFusion objects, on the **ValidateThis Google Group** a few weeks ago. His questions got me thinking about how I might do that, and those thoughts turned into design ideas, which turned into code. So I'm happy to announce that the latest version of VT, just released to **RIAForge**, has full support for i18n. I have created a **new i18n demo** which shows off this new feature in all of its glory. If you visit that demo page you'll see that you can switch languages between English and French.

So, what does i18n support in VT mean? In a nutshell, it means that all validation failure messages, both on the client side and on the server side, can now appear in multiple languages within a single application. A developer still only has to specify the metadata for each validation rule once (which is the primary objective of the framework), with the translations being performed by a translation layer.

What the heck's a translation layer, you may ask? Well, I knew that I wanted to provide a means of using resource bundles, which are a standard in Java, and, I believe, in web applications in general. However, I also recognized that not everyone might choose to use resource bundles to store their translation metadata, so I didn't want to lock the framework into that single implementation. So I created a Translator object, which is used to perform all translations, and which can be easily overridden by a developer.

I actually created two Translator objects. The BaseTranslator actually does nothing. It accepts a term for translation and returns the identical term. That translator will be used by the framework when i18n is not required. The second Translator object, which extends the BaseTranslator is called the RBTranslator. It uses resource bundles to provide translations for terms. Because it is simple to override either of these Translator objects with your own object, if you want to provide translations in another format (e.g., from a database table), all you have to do it write a new Translator and tell the framework to use your Translator instead.

The only documentation of how to implement this feature is the demo application mentioned above. It is available as part of the **download from RIAForge**. I do hope to provide some more documentation on how to use all aspects of the framework, probably in the form of a wiki, but as long as I'm the only one using it (as far as I know), that remains a low priority.