cfUniForm v2.6 - Global Plugin Configs + ValidateThis! Integration

Posted At : November 13, 2008 4:17AM | Posted By : Matt Quackenbush
Related Categories: ColdFusion, Uni-Form Tag Library

The latest release of the cfUniForm tag library focuses on a new configuration feature, as well as integration with the ValidateThis! validation framework.

New Feature: Global jQuery Plugin Configuration

cfUniForm has built-in integration with several jQuery plugins, such as the date picker, time entry, validation, and textarea resizing plugins. While the default behavior of each plugin is quite useful in many cases, sometimes you need to alter the behavior for your form(s). While the v2.5 release added support for setting these behaviors within each form, Dan Wilson suggested being able to pass in those plugin configs via the config struct. Now you can!

Here is a quick overview:

- You can create a GlobalConfig struct in CFML, or if you are utilizing a DI framework such as ColdSpring, you can create your config inside of it. Here are the valid keys (all keys are optional):
  - **jQuery** (path to jquery.js)
  - **renderer** (path to the renderValidationErrors.cfm custom tag)
  - **uniformCSS** (path to the uni-form-styles.css)
  - **uniformJS** (path to the uni-form.jquery.js)
  - **validationJS** (path to the jQuery validation js file)
  - **dateCSS** (path to ui.datepicker.css)
  - **dateJS** (path to ui.datepicker.js)
  - **dateSetup** (struct of var:value pairs to config the datepicker plugin)
  - **timeCSS** (path to timeenter.css)
  - **timeJS** (path to timeentry.js)
  - **timeSetup** (struct of var:value pairs to config the timeentry plugin)
  - **maskJS** (path to maskedinput.js)
  - **textareaJS** (path to prettyComments.js)
  - **textareaSetup** (struct of var:value pairs to config the prettyComments plugin)
- JavaScript is cAsE-sEnSiTiVe. Therefore, when creating config keys for your plugin setups, you *must* utilize array notation. If you don't, ColdFusion will automatically render the key in UPPERCASE. So, instead of

  <cfset myStruct.dateSetup.yearRange = "'2008:2040'" />

  you would write it as

  <cfset myStruct.dateSetup['yearRange'] = "'2008:2040'" />

- Certain plugin setup rules require single quotes (e.g. strings). Others require no quotes (e.g. true|false). The cfUniForm tags assume that you know what you're doing and will output exactly what you've written. If the JavaScript breaks, it's likely because of quotes not being where they should be, or vice-versa.

- Even if you've supplied plugin setup rules in your GlobalConfig struct, you can still override them on an individual form by utilizing the attributes provided on the form tag (e.g. @dateSetup or @timeSetup). You may also override them on an individual field by utilizing the 'pluginSetup' attribute on the field tag.

To see it in action, visit the demo page .

ValidateThis! Integration

Validation is a PITA. Obviously, it's a necessary thing, but it's really no fun to code all of the validation for each form and action in your application. There's client-side validation, but it came from the client, so you can't really trust it. As a result, that means you have to write server-side validation. What if there was a way to have all of that generated for you?

One of the really, really, really, really, *really* cool things that has come along for cfUniForm is Bob Silverberg 's ValidateThis! validation framework. Thanks to a couple of code submissions by Bob, cfUniForm now integrates fully with ValidateThis!. As a matter of fact, the ValidateThis! demo page is powered by cfUniForm.

I would go into great detail explaining ValidateThis!, but I could not do it justice. For more information I would recommend checking out Bob's blog as well as the new ValidateThis! mailing list on Google Groups. There have been some pretty intense discussions going on over there the last few days. Drop by and jump on in.

In case you missed it, you can find the cfUniForm demos here .

Thanks again to Dan and Bob for their contributions to both cfUniForm and the ColdFusion community at large.