Contents

Using the Custom Validation Type

The custom validation type is used when you want to write arbitrary CFML code to enforce a validation. The other validation types work by interrogating the values of properties in the object itself. For example, the *required* validation type checks the value of the property in question to ensure that it is not empty.

Sometimes you need to use data in a validation that does not reside in the object itself. For example, you may want to ensure that a *userName* property contains a unique value. In that case you'd need to look in the database to see if the userName has already been assigned to another record, so you'd need to execute a database query and check the number of records returned. The custom validation type allows you to do this.

How It Works

In order to use the custom validation type in a rule you must do two things:

1. define a rule with a type of *custom* for your object
2. have a method in your object that performs the validation

The framework will then call that method when performing validations on your object, and the result of the method will determine whether the validation passed or not.

Defining a Custom Rule

Creating a Rule Definition

You define a custom rule for a property of an object in the same manner as any rule, you add a *rule* element to your xml file inside the given *property* element. For example, to specify a custom rule for the *userName* property, your xml would look something like this:

```
<property  name= "userName" >
 <rule  type= "custom"  failureMessage=  "That UserName is already taken. Please try a different one."       >
  <param  name= "methodname"  value= "validateDuplicateUserName"     />
  <param  name= "remoteURL"  value= "validateDuplicateUserName.cfm"     />
 </rule>
</property>
```

Parameters

A custom rule definition has one required and one optional parameter:

- *methodName* The name of the method in the object that performs the validation
- *remoteURL (optional)* A URL that can be called via AJAX which will result in a response of either *true* or *false*

Defining the Custom Method

As discussed above, the *methodName* parameter of the custom rule will point to a method in the object being validated. That method can do anything you like. For example, it could perform a database query, it could check for the existence of a file on disk, it could check against a value in a persistent scope, such as the *application* or *session* scope.

A good best practice is to have the method call a method in a service object, as ideally you wouldn't want to have database queries or any of the other logic mentioned above inside your business object, but the choice is yours. The method can do absolutely anything that can be done with CFML. The only requirement is that it must return either:

1. a boolean, or
2. a struct with a particular set of keys

The Return Value

Your method can simply return a boolean, in which case the validation will consider to have passed if the method returns *true* and to have failed if the method returns *false*.

If you want to be able to define the failure message returned from within your method then you can opt to return a struct. If returning a struct it must contain following keys:

- *isSuccess* Should be *true* if the validation passed and *false* if the validation failed
- *failureMessage* A message to display to the user if the validation failed. Because this is generated by the method in the object, you can make this message as dynamic as you wish.

A Sample Custom Method

Continuing with our example of a custom validation that checks for a duplicate userName, here's an example of a custom method that uses a composed service object to do the check:

```
<cffunction  name= "validateDuplicateUserName"    access= "public"  output= "false"  returntype= "struct"  hint= "Checks for a duplicate userName."
 <cfset  local.returnStruct = {isSuccess =     false ,failureMessage =   "That UserName has already been used. Try to be more original!"       } />
 <cfif  NOT getUserService().isDuplicateUserName(getUserId(),getUserName())       >
  <cfset  local.returnStruct.isSuccess =     true  />
 </cfif>
 <cfreturn  local.returnStruct   />
</cffunction>
```

This method starts by creating a default struct that will be returned if the validation fails. It then asks a userService object that has been composed into the business object to check for a duplicate userName, passing in the userId and userName properties of the current object. If the service reports that it is not a duplicate, then the method changes the isSuccess key in the struct to *true*. Finally the method returns the struct.

Setting Up a Remote URL

As mentioned above, the *remoteURL* parameter allows client-side validations to make use of the custom rule type, even though the validation rule is coded in CFML. It does this by using the *remote* feature of the jQuery validation plugin.

Note that the request to the URL must return a simple string with a value of either *true* or *false*.