Contents

Defining Validation Rules

There are currently three means of defining the validation rules for your objects, in an external *Rules Definition File*, using annotations on property tags and via ColdFusion code.

Using a Rules Definition File

Two formats of rules definition files are supported: XML and JSON. Some advantages to recording your rule metadata in an external file (i.e., not inside your business object), include:

- The definitions of the validation rules are kept separate from the code of the object itself. This provides a couple of benefits:
  - One can change validation rules without having to open up the object code, which removes the chance of bugs being introduced into the code.
  - One can change one's approach to validations for an object (by switching to a different validation framework, for example) without having to make any changes to the object's code.
- The xml schema provides a very concise way of describing the validation rules for an object. There is absolutely no duplication of information required.
- If you are using ValidateThis without objects (i.e., validating a structure), this is currently the only option.

Details of the metadata that is recorded in a rules definition file can be found in the Rules Definition File reference.

Details of all of the validation types available can be found in the Validation Types Supported By ValidateThis section.

Sample rules definition files, in XML and JSON format, can be found on the Sample Rules Definition File page.

More details on where to place your rules definition files can be found in the How ValidateThis Finds Your Rules Definition Files section.

Using Property Annotations

You can add annotations to properties of your objects to define validation rules. These annotations can be in one of three formats: XML, JSON and ValidateThis Markup Language (VTML). More documentation on defining rules using annotations is in the works.

The addRule() Method

The *addRule()* method allows you to add a rule via ColdFusion code. Currently this technique is only supported if you are injecting the ValidateThis framework into your business objects, which will then give your object access to this method. We are looking at ways of making it simpler to define validation rules via CFML, which will eliminate that requirement.

The *addRule()* method accepts the following arguments:

- *propertyName* The name of the property for which the rule is being defined.
- *valType* The validation type.
- *clientFieldName (optional)* The id of the form field that corresponds to the property. Defaults to propertyName.
- *propertyDesc (optional)* A descriptive name of the property, used in validation failure messages. Defaults to propertyName.
- *condition (optional)* A structure that contains keys for *name*, *serverTest*, *clientTest (optional)* and *desc (optional)*.
- *parameters (optional)* A structure that contains one key for each parameter.
- *contexts (optional)* A list of contexts to which the rule should be added.
- *failureMessage (optional)* A custom message to be displayed on validation failure.
- *formName (optional)* The name of the form to which the rules applies.

Please see the Rules Definition File reference for a more in-depth description of the above arguments.

Future Enhancements

The framework has been designed to allow for additional ways in which to supply the validation metadata. The addition of the JSON and VTML formats makes use of these extensibility features. If you are interested in using an alternate form of metadata, please contact us and we'd be happy to help you achieve your goal.

We are currently looking at ways of specifying validation metadata dynamically via CFML.