Rules Definition File

Each business object will have its own corresponding rules definition file, in which all of the validation rules for the object are described. Currently this file can be in either XML or JSON format.

There is an *xml schema document* (xsd) which comes with the framework, located at /ValidateThis/core/validateThis.xsd. There is also an online version available at http://www.validatethis.org/validateThis.xsd. If you will be using XML files, it is **highly recommended** that you validate your files against this xml schema.

Annotated examples of both an XML and a JSON Rules Definition File are available on the Sample Rules Definition File page.

The following section describes all of the metadata that can be specified to describe your validation rules. It describes the metadata in terms of xml elements and attributes, but the conversion to JSON is straightforward. Please refer to the Sample Rules Definition File page for an example of the JSON format.

Items in *italics* denote an element attribute.

Elements

- **validateThis**
  Top level element for the xml file.
  - **conditions** (optional)
    Specify any conditions that are to be assigned to rules below.
    - **condition**
      A condition that will be used in one or more rule.
      name A unique name for the condition.
      serverTest A ColdFusion expression that will evaluate to either true or false in the context of a Business Object.
      clientTest (optional) A JavaScript expression that will evaluate to either true or false in the context of a form.
      desc (optional) A description of the condition, used to generate validation failure messages.
  - **contexts** (optional)
    Specify any contexts for which you wish to define formNames. These are used to target a particular form when generating client-side validations.
    Note that you **do not** have to specify contexts here in order to use contexts in your rules. The only purpose of this element is to allow contexts to be assigned form names.
    - **context**
      A context that is targeted at a specific form.
      name A unique name for the context.
      formName The name of the form that corresponds to the context. Used to support generating client-side validations for multiple forms on a single page.
  - **objectProperties**
    This is a container for all of the properties of the Business Object which need to be defined to the framework.

- **property**
  A property element must be included for any property that either:
  1. Has any rules defined for it.
  2. Needs a description recorded for it. E.g., if it relates to another property with a rule (e.g., via an equalTo rule type).

  name The name of the property in your Business Object. The Business Object must have a corresponding getter, either explicitly or implicity.
  desc (optional) A descriptive name for the property, used to generate validation failure messages. Defaults to the value of @name.
  clientfieldname (optional) The fieldname (id, in fact) on a form that corresponds to the property. Defaults to the value of @name.
  Note that *clientfieldname* need only be used if the name of your property differs from the form fieldname. For example, a User Object has a UserGroup, and therefore has a UserGroup property, but because UserGroup contains an object, we need to tell the framework the form fieldname (probably UserGroupId) in order to allow client-side validations to be generated for the field.
  - **rule**
    A property may have any number of rules defined, including zero.
    Note that defining multiple rules **of the same type** for the same property may cause problems for certain client-side validation routines.
    type The type of validation to perform (e.g., required, email, custom, etc.).
    contexts (optional) A comma delimited list of contexts in which the validation should be performed.
    E.g., a User object might have a Register context, a PasswordChange context and an AddressChange context.
    Note that either leaving the contexts attribute missing, or specifying a value of "" or "*" will cause the rule to be added to all contexts.
    Note also that contexts are not required at all. A simple object might not need any contexts.
    failureMessage (optional) A message to display to the user when a validation failure occrus. If none is specified the framework will generate a failure message.
    condition (optional) A condition that must evaluate to *true* in order for this validation rule to be applied.
    Note that the value of the *condition* attribute must be the name of a condition that has been defined in the conditions element above.
    - **param**
      A rule may have any number of params defined, including zero.
      Certain rule types require that certain params be defined. E.g., the MaxLength rule type requires a MaxLength param.
      name The name of the parameter (e.g., minLength, maxLength, methodName, etc.).
      value The value of the parameter.
      type (optional) The type of the parameter. This can be one of either value, expression or property.
      Value is the default, which simply uses the parameter value as is.
      Expression means that ColdFusion is to evaluate the value of the parameter. For example, specifying a type of 'expression' and a value of 'Now()' will cause the actual value of the paremeter to be the current date/time.
      Property means that the actual value of the parameter should be the

contents of a property of the object. For example, specifying a type of 'property' and a value of 'firstName' will cause the actual value of the parameter to be the contents of the firstName property.

Categories:

- Configuration