# Contents



# Specifying the objectType on a Method Call

When using the ValidateThis Facade Object, for most method calls it is necessary to let the framework know which objectType you are working with. This objectType corresponds to the filename of your Rules Definition File.

You inform the framework of the objectType in one of two ways:

1. You pass a value for *objectType* as an argument to the method.
2. You pass in the object, and the framework determines the *objectType* for you.

### Passing in the objectType Argument

This is the most straightforward method, although also the most verbose in some situations. You simply pass the name of the *objectType* into the method, like so:

```
<cfset Result = application.ValidateThis.validate(theObject=myObject,objectType='') />
```

This explicitly tells the framework that the Rules Definition File is called *User.xml* (or *User.xml.cfm*).

### Allowing the Framework to Determine the objectType

In this scenario you pass the object itself to the framework, and the framework then attempts to determine the *objectType*. The framework will do two things to make this determination:

1. If the object contains a method called *getobjectType()*, the framework will call that method and use the returned value as the *objectType*. This allows you to define the objectType within the business object itself, using CFML code.
2. Otherwise the framework will look at the metadata of the object and use the final part of its dot delimited name as the *objectType*.

#### Example 1:

I have a component called UserObject.cfc, which I instantiate to create a User object. That cfc contains a method called *getObjectType()* with the following code:

```
<cffunction name="getObjectType" access="public" output="false" returntype="any">
 <cfreturn "myUser" />
</cffunction>
```

If I call *validate()* passing in only this object, like so:

```
<cfset Result = application.ValidateThis.validate(theObject=User) />
```

The framework will look for a Rules Definition File called *myUser.xml*, because *myUser* is the value returned from *getObjectType()*.

#### Example 2:

I have a component called UserObject.cfc, which I instantiate to create a User object. That cfc **does not contain** a method called *getObjectType()*.

If I call *validate()* passing in only this object, like so:

```
<cfset Result = application.ValidateThis.validate(theObject=User) />
```

The framework will look for a Rules Definition File called *UserObject.xml*, because *UserObject* is the filename of the cfc.

### Keeping It Simple

The above may seem very complicated, and the options only exist to provide a developer with maximum flexibility. To keep things simple it is recommended that one adopts one of the above approaches and sticks with it throughout their application code.

To summarize, the following three options exist for telling the framework the name of your Rules Definition File.

1. Pass it into each method using the *objectType* argument.
2. Create a *getObjectType()* method in your business objects that returns the *objectType* value.
3. Use the same name for your business object cfcs as you do for your Rules Definition File, and the framework will match them up.