

Contents

- [Integrating VT with ColdBox](#)
 - [The ColdBox Interceptor](#)
 - [i18n Support](#)
 - [The ColdBox Plugin \(Deprecated\)](#)

Integrating VT with ColdBox

The ColdBox Interceptor

ValidateThis 0.99 and above, ships with a ColdBox Interceptor which you can find in the extras/coldbox directory. The interceptor has been written to work with ColdBox 2.6.4 and ColdBox 3.0.0 (which is currently a Release Candidate).

To set up ColdBox to use the interceptor, add the definition to your ColdBox config file. The following is a simple example of using the interceptor with a ColdBox 3.0.0 application.

```
function configure() {
    ...your ColdBox config settings...

    interceptors = [
        //ValidateThis
        {
            class= "validatethis.extras.coldbox.ColdBoxValidateThisInterceptor"
        }
    ];
}
```

The interceptor will load and configure an instance of ValidateThis and store it in the ColdBox cache. To access ValidateThis from within your application you can use the DSL property injection syntax:

```
<--- Coldbox 3.0.0 syntax --->
<cfproperty name="ValidationService" inject="ocm:ValidateThis" >
<--- Coldbox 2.6.4 syntax --->
<cfproperty name="ValidationService" type="ocm:ValidateThis" >
```

Alternatively you can get ValidateThis from the cache by using:

```
ValidationService = getColdboxOCM().get( "ValidateThis" );
```

To pass in configuration settings to ValidateThis, you need to define them as properties. For example:

```
function configure() {
    ...your ColdBox config settings...

    interceptors = [
        // Configure ValidateThis
        {
            class= "validatethis.extras.coldbox.ColdBoxValidateThisInterceptor" ,
            properties = {
                // Tell VT that I will include Javascript libraries myself (optional)
                JSIncludes = false,
                // Tell VT that I want to use a customised version of ValidateThis.core.BOValidator (optional)
                boValidatorPath = "model.validation.BOValidator"
            }
        }
    ];
}
```

By default the ColdBox interceptor will store ValidateThis in the ColdBox cache with the key name "ValidateThis", if you would prefer to use a different name, then add a ValidateThisCacheKey property to your interceptor config and specify the key name you want to use.

i18n Support

ValidateThis supports i18n for multi-language sites. If you are using the ColdBox i18n resource bundles, then the ColdBox interceptor will recognise this and configure ValidateThis to use the ColdBox resource bundles.

The ColdBox Plugin (Deprecated)

Previous version of ValidateThis shipped with ColdBox Plugins which could be used to load and configure ValidateThis. The plugins have been deprecated in preference of the ColdBox Interceptor (see above). If you are using the plugin, then it may not be compatible with future versions of ValidateThis.

Unless you are using the ColdBox External Plugin location setting, you will need to copy the ValidateThisCB3Plugin.cfc which ships with ValidateThis into the plugins directory of your ColdBox application.

To configure the ValidateThis ColdBox plugin, define the ValidateThis config keys and values in the custom settings part of your ColdBox config file. for example:

```
function configure(){  
    ...your ColdBox config settings...  
  
    // custom settings  
    settings = {  
        ValidateThisConfig = {  
            JSIncludes = false,  
            defaultFormName = 'vtform'  
        }  
    }  
}
```

To access the ValidateThis ColdBox Plugin from within your application you can use the DSL property injection syntax:

```
<!-- Coldbox 3.0.0 syntax --->  
<cfproperty name="ValidationService" inject="coldbox:myPlugin:ValidateThisCB3Plugin" >  
<!-- Coldbox 2.6.4 syntax --->  
<cfproperty name="ValidationService" type="coldbox:myPlugin:ValidateThisCB3Plugin" >
```

Alternatively you can get the ValidateThis ColdBox Plugin using:

```
ValidationService = getMyPlugin( "ValidateThisCB3Plugin" );
```