

[Back to Main Extension Provider Tutorial](#)

Writing an extension - Part 3

After we have created the provider and created the form, we now want to validate the form so that one can check for different circumstances in the form editing. So whether combinations of fields are allowed only or similar other checks.

For this you can use the method called `validate` of the `install.cfc` component. In order to validate the forms for mango blog I created the following method:

```

name="validate" returnType="void" output="no" hint="called to validate the entered">
  name="error" type="struct">
    name="path" type="string">
    name="config" type="struct">
    name="step" type="numeric">

  var mixed = config.mixed>
  arguments.step eq 1>
    not structKeyExists(config, "datasource_select") or len(trim(mixed.datasource_select)) eq 0>
      error.fields.datasource_select="Please choose an existing datasource or create a new one">

    mixed.datasource_select eq "new">
      len(trim(mixed.datasource_new)) eq 0>
        error.fields.datasource_new = "The datasource name must not be empty">

      not structKeyExists(config, "dbtype_new") OR len(trim(mixed.dbtype_new)) eq 0>
        error.fields.dbtype_new = "Please select a database type">

      len(trim(mixed.server_new)) eq 0>
        error.fields.server_new = "The server name must not be empty">

      len(trim(mixed.database_new)) eq 0>
        error.fields.database_new = "The database name must not be empty">

      len(trim(mixed.username_new)) eq 0>
        error.fields.username_new = "The username must not be empty">

      len(trim(mixed.password_new)) eq 0>
        error.fields.password_new = "The password must not be empty">

  arguments.step eq 2>
    not structKeyExists(config, "isblognew") or len(trim(mixed.isblognew)) eq 0>
      error.fields.isblognew = "Please choose whether the blog is new or it will be imported from either BlogCFC or Wordpress">

    mixed.isblognew eq "new">
      len(trim(mixed.name)) eq 0>
        error.fields.name = "The name of the blog owner">

      len(trim(mixed.username)) eq 0>
        error.fields.username = "The login username must not be empty">

      len(trim(mixed.password)) eq 0>
        error.fields.password = "The login password must not be empty">

      len(trim(mixed.email)) eq 0>
        error.fields.email = "The blog owner email must not be empty">

      len(trim(mixed.blog_title)) eq 0>
        error.fields.blog_title = "The blog title must not be empty">

      len(trim(mixed.blog_address)) eq 0>
        error.fields.blog_address = "The blog address must not be empty">

```

So simply by setting keys (named after their corresponding form fields) in the `error.fields` struct you can validate your form. If the `error.fields` struct contains no key, the form validation is completed successfully. As you can see above the method gets passed the following attributes: `error`, `path`, `config` and `step`.

By adding several steps to the `config.xml` you can create a kind of wizard which you can use for filling the necessary variables for the configuration of your installer. In the `validate` method you receive an argument that contains the current step to validate. After the last step has been validated, the `install` method is called which we will have a look at next. The `install` method simply executes whatever actions you require it to do. So it's mostly up to you to define what is going to happen there. You can unzip the archive, create mappings, create tables and datasources

and even other things like add jars to the local context or create CFX tags.

[Back to Main Extension Provider Tutorial](#)