Identifier Validation Plugins

See also: Writing Registry Plugins and Identifier Validation

Plugin Requirements

1. The name of the Plugin should match the format `FooIdentifierValidator`.
2. The Plugin should implement a `FooIdentifierValidator` model and a corresponding controller (`FooIdentifierValidatorsController`)
    a. This model should `belongTo CoIdentifierValidator`.
    b. The controller should extend `SIVController`.
3. The Plugin must declare whether or not it must be instantiated, as described below. For Plugins that are instantiated:
    a. When a new CO Identifier Validator is created, a skeletal row in the corresponding `foo_identifier_validators` table will be created. There is no `add` operation or view required. The skeletal row will point to the parent CO Identifier Validator.
    b. When a CO Identifier Validator is edited, the entry point to the Plugin will be `foo_identifier_validator /foo_identifier_validators/edit/#`. This will be called immediately after the parent CO Identifier Validator is created.
    c. Note `CoIdentifierValidator` has a `hasOne` (ie: 1 to 1) relationship with `FooIdentifierValidator`.
    d. The table `foo_identifier_validators` should include a foreign key to `co_identifier_validators:id`.
        i. Other tables used by the plugin should reference `foo_identifier_validators:id`.
4. The Plugin must implement a validation routine in the FooIdentifierValidator model, as described below.

Instantiation

Identifier Validation Plugins may determine whether or not they should be *instantiated*. An instantiated Plugin requires configuration information, such as an LDAP server to query. A non-instantiated Plugin does not, such as a Plugin that determines if there are any non-ASCII characters in the identifier.

A plugin declares whether or not it needs to be instantiated in the FooIdentifierValidator model:

```
public $cmPluginInstantiate = true;
```

If a Plugin is instantiated, then

Validation

The Plugin must implement a validation routine in the `FooIdentifierValidator` model, with the following signature:

```
public function validate($identifier, $coIdentifierValidator, $coExtendedType, $pluginCfg=null)
```

where

- **identifier**: The identifier (or Email Address) to be validated.
- **coIdentifierValidator**: An array of configuration information about this validator, corresponding to cm_co_identifier_validators.
- **coExtendedType**: An array of extended type information describing the `$identifier` type. Corresponds to cm_co_extended_types, but for Email Addresses and Identifiers only.
- **pluginCfg**: If this plugin is instantiated, an array of configuration for this plugin.

Return Values

If `validate` determines the `$identifier` to be acceptable/valid, it returns `true`. Otherwise, the function should throw an exception in accordance with the type of error.

- **InvalidArgumentException**: The identifier is not in an acceptable format. (eg: The identifier has non alpha-numeric characters.)
- **OverflowException**: The identifier is already in use. (eg: The identifier is already a mail alias.)
- **RuntimeException**: Any other type of error.

The message passed with the exception should be a localized string suitable for display to the user.