Subject: Validation plug-ins
Posted by janalwin on Thu, 07 Sep 2006 12:57:56 GMT
View Forum Message <> Reply to Message

Hallo,

I'd like to integrate my own validations in the validation class. For example for checking postcodes, telephonenumber formats etc. These are validations that occur over and over in projects and it's a waste to reimplement them in every database class.

It would by nice if you could plug-in validators into the standard validation object.

Maybe it should work this way:
- define in the subtype which validator is to be loaded: subtype="dutch_postcode"
- load the class dutch_postcode-validator in the validation class.
- do the validation with this class

We could setup some kind of repositery for exchanging validator classes between developers.

Maybe you think it's not a good idea to mess with the current validation class. It still would be nice to have a standard way to implement validators in an other way.

--
Jan Alwin de Jong

Subject: Re: Validation plug-ins
Posted by AJM on Thu, 07 Sep 2006 13:34:47 GMT
View Forum Message <> Reply to Message

I don't really want to mess with the current validation class as it is totally generic, but the need for custom validation rules which can be applied after the generic rules does seem like a reasonable request.

I will have a think about how this may be done to see what the possibilities are, then I will get back to you.

Off the top of my head something like an extra field in the data dictionary to identify a validation method for that field, at at runtime I could instantiate an object from a copy of 'custom_validation_class' which exists in your project directory, then pass that field's value to that method name. How does that sound?

Subject: Re: Validation plug-ins
Posted by janalwin on Thu, 07 Sep 2006 13:56:56 GMT
View Forum Message <> Reply to Message

Sounds good to me!

Subject: Re: Validation plug-ins
Posted by janalwin on Thu, 07 Sep 2006 14:49:30 GMT
View Forum Message <> Reply to Message

Just been thinking it over.

If you call a method in a custom validator class by the name defined in the data dictonary, it will be hard to exchange the validator-methods with others, or to put it in other projects without altering working code.

Wouldn't it be a better idea to load a class with a standard interface from a directory with validators.

For example
=== Valdidator class ====

```
class dutch_postcode  {
     var $errors=Array();

     function validate($fieldname) {
          //.. put custom validation code here
   }

   function getErrors() {
        return $errors
  }
}
```

==== validator call code ====

```
if (isset($fieldspec[$fieldname]['custom_validator'])) {


   require_once($path_to_validators .  $fieldspec[$fieldname]['custom_validator']) . ".class.inc");

   $validator =&  singleton::getInstance($fieldspec[$fieldname]['custom_valida tor']);

    $validator->validate();

    $this->errors=array_merge($this->errors,$validator->getErrors())

} // end if
```

Subject: Re: Validation plug-ins
Posted by AJM on Thu, 07 Sep 2006 15:28:03 GMT
View Forum Message <> Reply to Message

Before I can activate a method on a class there are 3 things I must know:
(a) the name of the file containing the class definition
(b) the class name
(c) the method name

I can either have (a) and (b) taken care of by providing a blank custom validation class with a fixed name to which you just add your own methods, or I can allow you to create your own class(es) with your own name(s).

The first option means that all you have to supply in the data dictionary is the method name, while the second option means that you have to supply the file name, the class name and the method name.

The first option means that you can have a separate copy of the custom validation class in your own application subdirectory so that it does not clash with anything being used in any other subdirectories, but it does mean that you can only have one custom validation class per subdirectory. This would mean that merging several different classes into one class would be a manual exercise.

The second option would be more flexible as you could have a range of different methods available in a range of different classes, and you would not have to touch any code. All you would have to do is specify in the data dictionary for each field the name of the file/class/method to be used for validation.

How does that sound?

Subject: Re: Validation plug-ins
Posted by janalwin on Thu, 07 Sep 2006 16:05:05 GMT
View Forum Message <> Reply to Message

I like option 2 the best. But is it really needed to have more than one validation method in a custom validation class? You could put every custom validation in a seperate class, load this class and call e standard method that all custom validation classes should have. (for example valdidate())

Every validator could extent this class for example:

```
 class custom_validator {

        var $errors

        function validate($fieldvalue) {

             return;
         } // end method

        function getErrors() {
            return $this->errors
         } // end method
} // end class
```

Examples of emplementations

```
class  dutch_postcode_validator extends validator {

        function validate($fieldvalue) {
                // check
                if (eregi.....    {
                        $this->errors[]="The postcode may only contain... "

                 }   //end if

         } // end method

} // end class

class  dutch_phonenumber_validator extends validator {

        function validate($fieldvalue) {
                // check
                if (eregi.....    {
                        $this->errors[]="A phonenumber can only contain numbers"

                 }   // end if

                if(strlen($fieldvalue)!=10) {
                        $this->errors[]="A phonenumber must contain 10 characters"
                 } // end if
         } // end method
```

} // end class

An other way to do it would be to define an interface, but it's only possible in PHP5 so that not an option I think.

I'm relatively new to object oriented programming. But this seems the most flexible method to me.

The indents in the code seem to get lost when I'm posting. Sorry

--
Jan Alwin

---

Subject: Re: Validation plug-ins
Posted by AJM on Thu, 07 Sep 2006 16:18:51 GMT
View Forum Message <> Reply to Message

I think the most flexible method is to allow you to use whatever file names you like, whatever class names you like, and whatever method names you like. You then have the choice of putting several related methods in the same class, or into separate classes. I do not see the point of restricting it to one method per class, with a fixed method name of validate(). This smells of the infamous "convention over configuration" that RubyOnRails is famous for, and I wish to avoid that if I can.

You are correct in saying that I do not want to use any options that only work in PHP 5.

---

Subject: Re: Validation plug-ins
Posted by janalwin on Fri, 08 Sep 2006 08:31:29 GMT
View Forum Message <> Reply to Message

OK.

I understand your point.

Jan Alwin

---

Subject: Re: Validation plug-ins
Posted by AJM on Mon, 11 Sep 2006 16:36:46 GMT

The code for this has been made available in version 1.15.0.

Refer to http://www.radicore.org/viewarticle.php?article_id=84 for documentation.