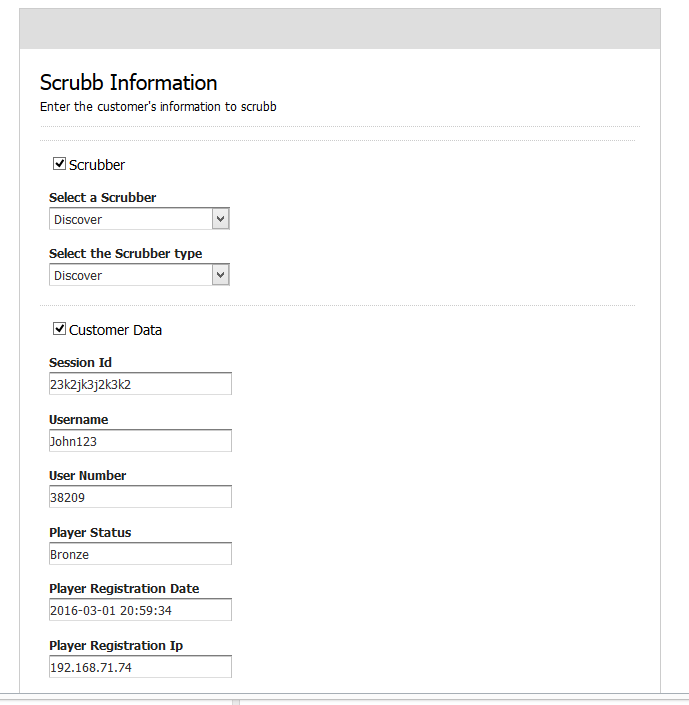
**Build a Laravel class that is available   
throughout the cycle of an HTTP web request**

**General requirements:**

* Do not use Session variables
* the classes must be instantiated once
* the created object must exist throughout the request, must be accessible whole application
* use Orchestra xml parser (https://github.com/orchestral/parser)
* strong OOP, MVC, Laravel practices, and clear reliable code.

<http://www.scrubber_tests.ag/main/form.html>



This form has an ajax call to the mini-app you are doing.

The form action is:

http://scrubber.ag/scrubb/bccc1fe7-e745-4343-b53b-6145993f177e

This is the route being invoked in our mini-app:

Route::*post*('scrubb/{id}', 'ScrubberController@Index');

For the purposes of this mini-project:

1. Option selected at “Select an action” is: “New”
2. Option selected at “Select the direction” is: “Request” or “Response”
3. Option selected at “Select a Scrubber” is: “AcuityTec”
4. Option selected at “Select the Scrubber type” is: “Full”

The controller is:

\app\Http\Controllers\ScrubberController.php

This controller is in charge of instancing the new class which based on the received parameters above.

*The parameters above must be set in a way they’re accessible from anywhere during the request. These values will not always be available from the HTTP POST request, so, for purposes of this mini-project, they must be properly set into a global object for the request. Maybe they can be set into a little class called ConfigSettings which is instanced in the controller and made available throughout the request.*

**ScrubberController**

This **creates** and sets a global object with initial settings:

* token
* action
* direction
* scrubber
* scrubberType

**Creates** a global “Scrubber” object which on initialization:

**Creates** a requestDataHandler object (if direction = “request”). This object is **visible locally within current Scrubber instance** only:

**Initialization:**

Instantiates a member object “objScrubberField” of type “ScrubberField”, which loads and parses XML ***conf\_fields\_[scrubberType]\_[action]\_[direction].xml***  by a class extending Orchestra parser (ScrubberXmlParser).

**ScrubberField functions:**

**filteredArray($refAttribute, $rawArray):**  
returns an associative array which keys exist in *conf\_fields\_[scrubberType]\_[action]\_[direction].xml*

**getVariableToSysArray((bool)$requiredOnly)**  
returns array where the key is the “variable” attribute the xml

**getSysToVariableArray((bool)$requiredOnly)**  
returns array where the key is the “sys” attribute the xml

**getAttributeOnSys($sysName, $attrName)**  
returns the value of an attribute by providing “sys” value of a field.

**getAttributeOnVariable($variableName, $attrName)**  
returns the value of an attribute by providing “variable” value of a field.

**getValue($params)**  
 *"params" => array(  
 "source" => "sys|variable",  
 "item" => "sysSessionId|session\_id"  
 )*  
returns the value of the Post Parameter (post parameter names are the ones defined in the "sys" attributes)

**getItemId($params):**  
example 1:  
*"params" => array(  
 "source" => "sys",  
 "destination" => "variable",  
 "item" => "sysSessionId"  
 )*returns the "variable" for given "sys" value  
  
example2:  
*"params" => array(  
 "source" => "sys",  
 "destination" => "variable",  
 "item" => "sysSessionId"  
 )*returns the "variable" for given "sys" value

All data from HTTP POST Request is loaded into an associative array, just a clone of the $\_POST object, except that only parameters included in the ***conf\_fields\_[scrubberType]\_[action]\_[direction].xml*** are present. This is done by calling objScrubberField->filteredArray(“sys”, $\_POST)

For “direction” = “request”, the name of the $\_POST parameters are the same as the “sys” attribute in the xml (example: *conf\_fields\_full\_new\_request.xml*).

For “direction” = “response”, the name of the $\_POST parameters are the same as the “variable” attribute in the xml (example: *conf\_fields\_full\_new\_response.xml*), and the call would be: objScrubberField->filteredArray(“variable”, $\_POST)

Throughout all XMLs, the attribute “sys” correspond to the name of an item in our local system, and “variable” corresponds to the name of an item in remote system being invoked or in a response being received from a remote system.

**How to determine which XMLs it’s gonna work on?**

This is based on the “Scrubber” value in “ConfigSettings” object. It will find the XMLs in the following location:

***\app\Scrubber\plugins\[Scrubber]***

*Example: \app\Scrubber\plugins\AcuityTec\*

**Function: getRequestParams()**

This function will iterate through all “field” elements in objScrubberField and:  
- if field is optional and is not contained in the $arrayPost, then it will add entry to $arrayErrors (ex: Missing field “[attribute: name]”)  
- if field is required and is not contained in the $arrayPost, then it will add entry to $arrayErrors (ex: Missing field “[attribute: name]”)  
- if field is contained in $arrayPost, then the data is validated according to these attributes: “type”, “validator”, “length”, if these data are passed and a “format” attribute is provided, the data is formatted accordingly, if validation or formtting failes an entry to $arrayErrors is added (ex: Wrong Field “[attribute: name]”).  
- fields that pass successfully will be added to $arrayPostFinal as key => value pairs (where “key” is the “variable” attribute (we are converting the original “sys” => “value” pairs to “variable” => “value” pairs with the value being validated and formatted.

**Validation Note:**  
- the validator is in the format: “x@y”, where “x” is the function that validates the value, “y” is a regular expression and is only present when function = “reg”

**Function: getErrors()**

Returns the $arrayErrors

**Function: getWarnings()**

Returns the $arrayWarnings

**Function: getErrorCount()**

Returns the count of elements in $arrayErrors

**Function: hasErrors()**

Returns TRUE if count of elements in $arrayErrors > 0

According to the "load" parameter it will use (if already created) or create (if not instantiated yet), a class in "\app\Scrubber"

According to the "scrubber" parameter and the "load" parameter, the new class will load and parse an xml in \app\Scrubber\plugins\[Scrubber], for example, if [Scrubber] = "AcuityTec" and "load" is "base":

- will instantiate \app\Scrubber\ScrubberConfBase if not exist or use if already exist

- will load and \app\Scrubber\plugins\AcuityTec\conf\_base.xml

**If information is OK, there are no errors and we are ready to make the remote HTTP request then, then we call makeRequest within “Scrubber” which makes a call to the remote API and returns ResponseDataHandler Object**

**Initialization:**

When the remote API system is called, we receive a response, which is then passed to the ResponseDataHandler constructor for its parsing.  
\_\_Construct($response) {  
 $objScrubberField = new ScrubberField(false)  
 $arrayResponse ($response data is parsed)  
 getResponse Params()  
}

**Function: getResponseParams()**

Sets $arrayResponseFinal, changing the keys of the associative array or object, by converting the "variable" based key to a "sys" based key (according to conf\_fields\_[scrubb type]\_[action]\_response.xml) the value is converted as well according to the provided converting classes and xmls (ex: ScrubberConfSuggest, ScrubberConfScore, etc.), values are formatted to provided “format” attribute in xml.

**Function: getData($params)**

This function will access data from the different classes converting data using the XMLs:  
- ScrubberConfSuggest  
- ScrubberConfErrors  
- ScrubberConfRreasons  
- ScrubberConfScore  
-ScrubberConfStatus

EXAMPLE of $params:  
*$params = array(  
 "load" => "base|errors|reasons|score|status|suggest",  
 "function" => "getConvert|getDescription|..."  
 "params" => array(  
 "value" => "xxxx"  
 )  
)*

**functions of each class:**

**ScrubberConfBase:**

- getSettingValue($setting)

returns the value for that setting.

- getScrubbTypes()

return simple array of supported (value="true" && type="boolean") scrubbTypes

- getCredentials()

return associative array of credentials (setting => value)

**ScrubberConfDecision:**

- getConvert($params) ($contains for example: array("token" => "bccc1fe7-e745-4343-b53b-6145993f177e", "score" => 2.5) ):

returns "convert" attribute for given token (if token not found returns from default)

>> if type = range it must validate $score according to "value" range

**ScrubberConfErrors:**

- getDescription($error)

returns "description" for a given error code

- getConvert($error)

return "convert" for the given $error value

**ScrubberConfReasons:**

- getDescription($reason)

returns the "description" for a given reason code

- getConvert($reason)

return "convert" for the given $reason value

**ScrubberConfScore:**

- getConvert($score) (example: "score" => 2.5):

returns the "convert" attribute for the given score

>> if type = range it must validate the provided $score falls in valid "value" range

**ScrubberConfStatus:**

- getConvert($status)

returns "convert" value for the provided $status

**ScrubberConfSuggest:**

- getConvert($suggest)

returns the "convert" value for the provided $suggest

**Get final result:**

Once the Response has been parsed and converted, a Result object is generated and returned.

This functions relies on some decision making and the “ScrubberConfDecision” class.