**AthletesUP project form validation check**

The purpose of this document is to unify the validation logic and clarify how errors will be returned.

**How to write the validation logic?**

Example:

**Team Controller:**

public function action\_post\_add()

{

**// GET THE POST VALUES**

$this->payloadDesc = "Add a new team";

$org\_sport\_link\_id = (int)trim($this->request->post('org\_sport\_link\_id'));

$orgs\_id = (int)trim($this->request->post('orgs\_id'));

$sports\_id = (int)trim($this->request->post('sports\_id'));

$complevels\_id = (int)trim($this->request->post('complevels\_id'));

$seasons\_id = (int)trim($this->request->post('seasons\_id'));

**// ADD THESE TO AN ARRAY OF ARGUMENTS TO PASS TO MODEL**

$args['org\_sport\_link\_id'] = $org\_sport\_link\_id;

$args['orgs\_id'] = $orgs\_id;

$args['sports\_id'] = $sports\_id;

$args['complevels\_id'] = $complevels\_id;

$args['seasons\_id'] = $seasons\_id;

**// PASS TO MODEL METHOD FOR SAVING AND GET RESULT OBJECT**

$result = $this->mainModel->addTeam($args);

**//Check for success / error**

if(get\_class($result) == get\_class($this->mainModel)){

return $result;

}

elseif(get\_class($result) == 'ORM\_Validation\_Exception'){

**// Parse error and add to error array. The second param is the path to the error path.**

$this->processValidationError($result,$this->mainModel->error\_message\_path);

return false;

}

}

**Team Model:**

**Rules Method:**

public function rules()

{

return array(

'seasons\_id' => array(

array('not\_equals', array(':value', 0))

),

'complevels\_id' => array(

array('not\_equals', array(':value', 0))

),

'year' => array(

array('not\_empty'),

array('digit'),

array('exact\_length', array(':value', 4)),

),

'mascot' => array(

array('alpha')

),

'unique\_ident' => array(

array('alpha')

)

);

}

**The Save Method:**

public function addTeam($post\_values = array()){

extract($post\_values);

if (isset($org\_sport\_link\_id))

$this->org\_sport\_link\_id = $org\_sport\_link\_id;

if (isset($complevels\_id))

$this->complevels\_id = $complevels\_id;

if (isset($seasons\_id))

$this->seasons\_id = $seasons\_id;

if (isset($year))

$this->year = $year;

if (isset($mascot))

$this->mascot = $mascot;

if (isset($unique\_ident))

$this->unique\_ident = $unique\_ident;

if (!isset($orgs\_id))

$post\_values['orgs\_id'] = 0;

if (!isset($sports\_id))

$post\_values['sports\_id'] = 0;

if ($org\_sport\_link\_id == ""){

**// ADD EXTERNAL VALIDATION RULES**

$external\_validate = Validation::factory($post\_values)

->rule('orgs\_id', 'not\_equals', array(':value', 0))

->rule('sports\_id', 'not\_equals', array(':value', 0));

//if check pass,add org\_id,sports\_id to db,generate org\_sport\_id for use

if ($external\_validate->check()){

//check org\_sport in db.

$org\_sport\_link\_model = ORM::factory("Sportorg\_Orgsportlink");

$result = $org\_sport\_link\_model->getOrgSportId($orgs\_id, $sports\_id);

if(!$result->loaded())

{

unset($org\_sport\_link\_model);

$org\_sport\_link\_model = ORM::factory("Sportorg\_Orgsportlink");

//Insert new row to org\_sport\_link

$org\_sport\_link\_model->orgs\_id = $orgs\_id;//already extracted

$org\_sport\_link\_model->sports\_id = $sports\_id;//already extracted

$org\_sport\_link\_model->save();

$org\_sport\_pk = $org\_sport\_link\_model->pk();

$this->org\_sport\_link\_id = $org\_sport\_pk;

}else{

$this->org\_sport\_link\_id = $result->id;

}

}

}else{

$external\_validate = Validation::factory($this->as\_array())

->rule('org\_sport\_link\_id', 'check\_org\_sport\_id\_exist');

}

**// ATTEMPT TO SAVE AND CATCH EXCEPTION IF VALIDATION FAILS.**

try {

$this->save($external\_validate);

return $this;

} catch(ORM\_Validation\_Exception $e){

**// RETURN THE ERROR OBJECT**

return $e;

}

}

**Where to put custom validate rules?**

/application/classes/Valid.php

Any validation methods that will be used more than once should go in Valid.php. Methods that will only be used once can go in the model the rule is for.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Method  (Get,Post,  Put,Delete) | Form rules | Expect Result | Validtion |
|  |  |  |  |  |
| User/add | Post | email | Not null | not\_empty,  email,  unique\_email |
|  |  | first\_name | Not null | not\_empty, alpha |
|  |  | last\_name | Not null | not\_empty, alpha |
|  |  | password | Not null | not\_empty |
|  |  | facebook\_id | null | N/A |
|  |  |  |  |  |
| user/addteam | Post | teams\_id | Not null | Not\_empty,  teams\_id\_exist, |
|  |  | orgs\_id | Not null | Not\_empty,  orgs\_id\_exist |
|  |  | sports\_id | Not null | Not\_empty,  sports\_id\_exist |
|  |  | complevels\_id | Not null | Not\_empty,  complevels\_id\_exist |
|  |  | seasons\_id | Not null | not\_empty, seasons\_id\_exist |
|  |  | seasons\_arr (Todo) |  |  |
| user/addsport | Post | sports\_id | Not null | Note empty,  sports\_id\_exist |
|  |  |  |  |  |
| user/addrole/ | Not do yet | roles\_id | Not null | Note empty,  roles\_id\_exist |
| user/addidentity | Not do yet |  |  |  |
| user/position | Not do yet |  |  |  |
|  |  |  |  |  |
| Org | Post | Name | Required | not\_empty |
|  |  | Single sport | bool–Only accept 0 or 1 | not\_empty,  in\_array(0, 1) |
|  |  | season\_profiles\_id | Not null | not\_equals |
|  |  | complevel\_profiles\_id | Not null | not\_equals |
|  |  | leagues\_id | Not null | not\_equals |
|  |  | divisions\_id | Not null | not\_equals |
| org/addsport | Not do yet |  |  |  |
|  |  |  |  |  |
| position/add/ |  | name | Required | not\_empty |
|  |  | sports\_id | Not null | not\_equals |
|  |  | stattab\_id | Not null | not\_empty, stat\_tab\_id\_exist |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Error Messages**

Universal messages are kept in /application/config/error\_messages.php

Any messages that will be used over and over can be put in here. If an error message is added, open a ticket and assign it to Mike to check the wording of the message.

Those error messages can then be referenced in the error message file which exist per model. These error messages go in /application/messages/\* where \* is the model path. Each model can define a property $this->error\_message\_path which holds the path to the error messages.