# Zend Form Zend_form code generator

1) Creating Form object

Zend form can be created by creating an object of ZendForm class.

```
$zendform = new ZendForm("ContactUS", "POST", "", $formattributes);
```

Params:

1) Class name: Name of the class that will extend Zend_form you need to create object of this class when using it in Zend framework
2) Form Method: Http method of the form i.e. either POST or GET
3) Form Action: Form action i.e. Name of controller/action on which will be posted
4) Form attributes: An array of form's attributes like form name, form id, and enctype if form contains file upload field e.g. `<form name='contactus' id='contctusid'>` etc

2) Setting form attributes

Form attributes can be set using an array of key value pairs and passing that array as 4 argument of the form constructor e.g.

```
$formattributes = array('id' =>'profilesubmission','enctype' =>
'multipart/form-data');
$zendform = new ZendForm("ContactUS", "POST", "", $formattributes);
```

3) Adding text field to the form

Text filed can be added to the form using addTextField() function e.g.

```
$zendform->addTextField("yourname", "Your Name:",
$fieldattributes,$filters, $namevalidators);
```

Params:

Field name: $fieldname represents id,name html preperty of the submit button i.e <input type="text" id="buttonid">

Field Label: Field label that should appear to the right of the field

Attributes: An array of key value pairs containing field attributes e.g.

```
$fieldattributes = array('id' =>'yourname','class' => 'textfield');
```

Filters: An array of filters that should be applied to the value of this filed when form is posted. All possible filters are defined in Zend_form_base_class.php. An array of filters can be created as

```
$filters = array(HTML_ENTETIES_FILTER,STRING_TRIM,STRIP_TAGS);
```

Validators: An array of validators those should be applied to this filed when form is posted. All possible validators are defined in Zend_form_base_class.php e.g.

```php
$notemptyvalidator = array('name' => NOT_EMPTY_VALIDATOR,'message' =>
"Please provide your First Name",'stopexecution' => "true");

$alphabetsvalidator = array('name' =>
ALPHABETS_VALIDATOR,'allowhitespaces' => 'true','message' => "First name
can only contain Alphabets",'stopexecution' => "true");

/// Note order is important. Validators will run in the same order as
pushed in the Array

$namevalidators = array($notemptyvalidator,$alphabetsvalidator);
```

Now this array $namevalidators can be passed as 5th argument to addTextField function and these validators will be applied in the sequence in which they are added in the array.

4) Creating validators

A validator can be created as array with fixed key names. E.g. Not empty validator can be created as

```php
$notemptyvalidator = array('name' => NOT_EMPTY_VALIDATOR,'message' =>
"Please provide your First Name",'stopexecution' => "true");
```

Keys:

1) name: Name of the validator.
2) message: Message that should appear when validation fails.
3) stopexecution: If set to true then next validator will not be executed if more than one validators are applied to same field. E.g. if we apply two validators i.e. NOT_EMPTY_VALIDATOR and ALPHABETS_VALIDATOR to name field and “stopexecution” is set to true for NOT_EMPTY_VALIDATOR then ALPHABETS_VALIDATOR will not be executed if user has left name field blank and submitted the form. However if user has filled value in the name field but if value does not contain on characters then ALPHABETS_VALIDATOR validator will be executed.
   If stopexecution is set to false then next validator will be executed and error message for both validators will be displayed on the form.

Finally you need to push all validators for a field to an array and pass that array to the create field function. Validators will be executed in the order in which they are pushed in the array.

Here are the few examples of creating individual validators

```php
$emailvalidator = array('name' => ENAILADDRESS_VALIDATOR,"message" =>
"Please provide a valid Email Address",'stopexecution' => "true" );

$digitsvalidator = array('name' => DIGITS_VALIDATOR , 'message' =>
'Phone number can only contain digits','stopexecution' => "true");
```

```
/// Value array must be in "Value shown to user" => "Backend value"
format
$inarrayvalidator = array('name' => IN_ARRAY_VALIDATOR,'haystack' =>
array('USA' => 'usa' ,'Canada' => 'cancada','United Kingdom' =>
'uk','Pakistan' => 'pakistan'),'strict' => 'false','message'=>'Invalid
value for Country','stopexecution' => "true");

$stringlengthvalidator =
array('name'=>STRING_LENGTH_VALIDATOR,'message'=>"Comments can have
minimum lenght of length of 10 characters and maximum lenght of 60
characters.",'max'=>60,'min' => 10,'stopexecution' => "true");

/// list of supported locales can be viewed at
http://framework.zend.com/manual/1.12/en/zend.locale.appendix.html
$dobvalidator = array('name' => DATE_VALIDATOR,'format'
=>'dd/mm/YYYY','locale' => 'en_US', 'message' => 'Please provide a valid
date','stopexecution' => "true");
```

5) Creating Filters

Filters are easy to create all you need to do is push all validators in single array and pass that array to respective field. Here are few examples of creating filters

```
$filters =
array(STRING_TRIM,HTML_ENTETIES_FILTER,STRING_TRIM,STRIP_TAGS);
```

6) Adding Drop down field

Adding drop down field is easy by using addSelectField() function. You need to pass an array of keys and values to the value argument of the function. Values array must be in "Backend value" => "Value shown to user" format as shown in the example below.

```
$values = array('' => 'Select Country', 'usa' => 'USA','cancada' =>
'Canada','uk' => 'United Kingdom');

$zendform->addSelectField("country", "Select country", $values,
array(),$filters, $countryvalidators);
```

Adding other form fields and validators are shown in the example files.