1 The Classes

1.1 Basics

The class createForm needs 4 variables.

```
createForm::getInstance($genFromArray,$icon,$formTagparams,
                                        $moreFormparams)->printForm();
```

The first is mandatory, it is the configuration array for the html string to be generated. The other three have default values.

- $icon is for required fields,
- the $formTagparams is for the attributes of the form tag,
- $moreFormparams to set classes or ids for the submit and reset button.

The public static function toString($formtagpara=FALSE, $moreformpara=FALSE) creates the html string based on the

```
$genFromArray =
```

```
array('Record Info//id=recinfo//class=fieldset01//l_id=lid//l_class=lclass'
    => array('Search' =>
         array('input' =>
                  array(   'type'=>'search',
                           'name'=>'search',
                           //'value'=> '',
                           //'required'=>'required',
                           'placeholder' =>'serach term',
                           'class'=>'theclass',
                           '@\#db' => 'VARCHAR( 80 ) NOT NULL',
                  ),
         ),
...
'Key Generator' =>
         array('keygen' =>
                  array(   'name'=>'keygen',
                  //'value'=>'wertyu56789',
                  'required'=>'required',
                  '@#db' => 'VARCHAR( 255 ) NOT NULL DEFAULT 0',
                  )
         ),
    ),
);
```

The function createTable() of the class dbOperations generates the MySQL code for generating a table for the form with a field for every form field but the buttons.

1.2 The Config Array Requirements

Top level of the array is the fieldset title, with classes or ids for the fieldset and the label, seperated by
// double slash. The variable is $fieldsetSep, which can be set within the class definition. All checkboxes, radio buttons or select fields are configured with arrays. See the example form which has an example for any field type available to day. To preselect a value or check a box use the separator @#= with 'checked' or 'selected' appended. The separator can be changed thru the variable, $separator. Every key value pair of the array is put into the string, except the @#db key. When no fieldset is wanted, put integers on the top level. The example above, replace 'Record Info' with an integer. It can't be left out, since the configuration array needs that structure. By replacing titles with integers or vice versa, fieldsets can be taken out of or put into the string.