SERVER VALIDATION

FORM VALIDATION, REPOPULATION and VALIDATORS

FORM VALIDATION

YAML VALIDATION FILE

To validate the form data, create a YAML validation file with the **same name of the action** called by the form in the **validate directory** of the module. This file contain the name of fields that need to be validated and the **validators**.

validation file sample:
/<app_name>/modules/<module_name>/validate/send.yml

```
fillin:
  enabled:      true

validators:
  myStringValidator:
    class: sfStringValidator
    param:
      min:          2
      min_error: This field is too short (2 characters minimum)
      max:          100
      max_error: This field is too long (100 characters maximum)

methods:          [post]     # This is the default setting

fields:
  name:
    required:
      msg:        The name field cannot be left blank
    myStringValidator:
```

E.g.: To validate the form data on the call to the **send action**, a configuration file called **send.yml** must be created

ACTION MODIFICATION

By default, symfony looks for a **handleError<name_of_action>()** method in the action class whenever the validation process fails, or displays the **<name_of_action>Error.php** template if the method doesn't exist.
To display the form again with an error message in it, override the default **handleError<name_of_action>()** method for the form handling action and end it with a redirection to the action with display the form. E.g.:

```
class ContactActions extends sfActions{
   ...
   public function handleErrorSend() {
        $this->forward('contact', 'index');
   }
}
```

You can add an error manually with the setError() method of the sfRequest:

```
$this->getRequest()->setError('name', 'The name field cannot be left blank');
```

TEMPLATE MODIFICATION

You can detect whether the form has errors by calling the ->hasErrors() method of the sfRequest object. To get the list of the error messages,use the method ->getErrors(). So you should add the following lines at the top of the template:

```
<?php if ($sf_request->hasErrors()): ?>
   <p>The data you entered seems to be incorrect.
   Please correct the following errors and resubmit:</p>
   <ul>
     <?php foreach($sf_request->getErrors() as $error): ?>
          <li><?php echo $error ?></li>
     <?php endforeach ?>
  </ul>
<?php endif ?>
```

To show the error message next to the field with error, simply add the following line to every field:

```
<?php if ($sf_request->hasError('<name_of_the_field>')): ?>
      <?php echo $sf_request->getError('<name_of_the_field>') ?>
<?php endif ?><br />
```

FORM REPOPULATION

If you want your form to be filled in with the values previously entered by the user, simply add these lines to your YAML validation file:

```
fillin:
  enabled: true      # activate repopulation
  param:
    name: test       # Form name, not needed if there is
                       only one form in the page
    skip_fields: [email]  # Do not repopulate these fields
    exclude_types: [hidden, password]
                     # Do not repopulate these field types
    check_types: [text, checkbox, radio] # Do repopulate
    converters:      # Converters to apply
      htmlentities:     [first_name, comments]
      htmlspecialchars: [comments]
```

By default, the automatic repopulation works for:

- text inputs, check boxes, radio buttons, text areas and select components (simple and multiple)

The fillin feature doesn't repopulate:

- file tags

VALIDATION HELPERS

<?php echo use_helper('Validation') ?>

```
form_has_error($param)
form_error($param, $options=array(), $catalogue= 'messages')
```

VALIDATORS

The **validators** can be found in the **symfony lib validator directory**.
Each validator is a particular class that can have certain parameters.
You can easily create new ones.

sfStringValidator

apply string-related constraints to a parameter

```
sfStringValidator:
  values:        [foo, bar]
  values_error: The only accepted values are foo and bar
  insensitive:    false  # If true, comparison w/ values is case insensitive
  min:            2
  min_error:      Please enter at least 2 characters
  max:            100
  max_error:      Please enter less than 100 characters
```

sfNumberValidator

verifies if a parameter is a number and allows you to apply size constraints

```
sfNumberValidator:
  nan_error:    Please enter an integer
  min:          0
  min_error:    The value must be at least zero
  max:          100
  max_error:    The value must be less than or equal to 100
```

sfRegexValidator

allows you to match a value against a regular expression pattern

```
sfRegexValidator:
  match:         No
  match_error:   Posts containing more than one URL are considered as spam
  pattern:       /http.*http/si
```

*The **match param** determines if the request parameter must matchthe pattern to be valid (value Yes) or match the pattern to be invalid (value No)*

sfCompareValidator

checks the equality of two different request parameters; very useful for password check

```
fields:
  password1:
    required:
      msg:      Please enter a password
  password2:
    required:
      msg:      Please retype the password
    sfCompareValidator:
      check:    password1
      compare_error: The two passwords do not match
```

*The **check param** contains the name of the field that the current field must match to be valid.*

sfPropelUniqueValidator

validates that the value of a request parameter doesn't already exist in your database. Useful for primary keys.

```
fields:
  nickname:
    sfPropelUniqueValidator:
      class:           User
      column:          login
      unique_error: This login already exists. Please choose another one.
```

In this example, the validator will look in the database for a record of class User where the login column has the same value as the field to validate.

sfEmailValidator

verifies if a parameter contains a value that qualifies as an email

```
sfEmailValidator:
  strict:       true
  email_error:  This email address is invalid
```

sfFileValidator

applies format (an array of mime types) and size constraints to file upload fields

```
fields:
  image:
    required:
      msg:      Please upload an image file
      file:     True
    sfFileValidator:
      mime_types:
        - 'image/jpeg'
        - 'image/png'
        - 'image/x-png'
        - 'image/pjpeg'
      mime_types_error: Only PNG and JPEG images are allowed
      max_size:          512000
      max_size_error:    Max size is 512Kb
```

sfUrlValidator

verifies a parameter contains a value that qualifies as a valid URL.

```
sfUrlValidator:
  url_error:    This URL is invalid
```

sfDateValidator

verifies a parameter is of a date format.