VALIDAÇÃO NO SERVIDOR

Validação e repopulação de Forms e Validadores

VALIDAÇÃO DE UM FORM

ARQUIVO DE VALIDAÇÃO YAML

Para validar os dados de um form, crie um arquivo de validação YAML com o **mesmo nome da ação** chamada pelo form no **diretório validate** do módulo. Este arquivo contém o nome dos campos que precisam ser validados e os **validadores**.

exemplo de um arquivo de validação:
/<app_name>/modules/<module_name>/validate/send.yml

Ex.: para validar os dados de um form na chamada para a **ação send**, deve-se criar um arquivo de configuração chamado **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:
```

MODIFICAÇÃO DA ACTION

Por padrão, o symfony procura por um método **handleError<nome_da_ação>()** na classe action sempre que o processo de validação falhar, ou exibe o template **<nome_da_ação>Error.php** se o método não existir. Para exibir o form novamente com uma mensagem de erro, sobrescreva o método default **handleError<nome_da_ação>()** para a ação chamada pelo form e finalize com um redirecionamento para a ação que exibe o form. Ex.:

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

É possível adicionar um erro manualmente com o método **setError()** do sfRequest:

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

MODIFICAÇÃO NO TEMPLATE

Você pode descobrir se o form possui erros chamando o método **->hasErrors()** do objeto sfRequest. Para obter a lista das mensagens de erro, utilize o método **->getErrors()**. Então, você deve adicionar as seguintes linhas no topo do 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 ?>
```

Para exibir a mensagem de erro próxima ao campo com erro, simplesmente acrescente a seguinte linha para cada campo:

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

REPOPULAR UM FORM

Se você deseja preencher seu form com os valores previamente informados pelo usuário, simplesmente acrescente estas linhas ao seu arquivo de validação YAML:

```
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]
```

Por padrão, a repopulação automática funciona com:

- text inputs, check boxes, radio buttons, text areas e componentes select (simples e múltiplos)

A característica de fillin não repopula:

- tags do tipo file

HELPERS PARA VALIDAÇÃO <?php echo use_helper('Validation') ?>

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

VALIDADORES

Os **validadores** são encontrados no **diretório lib validator do symfony.** Cada validador é uma classe particular que pode possuir determinados parâmetros. **Você pode facilmente criar novos validadores.**

sfStringValidator

aplica restrições relacionadas a strings para um parâmetro

```
sfStringValidator:
  values:         [foo, bar]
  values_error: The only accepted values are foo and bar
  insensitive:    false  # Se true, comparação com valores é case insensitive
  min:            2
  min_error:      Please enter at least 2 characters
  max:            100
  max_error:      Please enter less than 100 characters
```

sfNumberValidator

verifica se um parâmetro é um número e permite aplicar restrições de tamanho

```
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

permite equiparar um valor com um padrão de expressão regular

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

*O **parâmetro match** determina se o parâmetro do request deve ser igual ao padrão para ser válido (valor Yes) ou igual ao padrão para ser inválido (valor No)*

sfCompareValidator

verifica a igualdade de dois parâmetros de request diferentes; muito útil para checar senhas

```
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
```

*O **parâmetro check** contém o nome do campo que deve ser igual ao campo atual para ser válido.*

sfPropelUniqueValidator

valida se o valor do parâmetro do request já não existe em seu banco de dados. Útil para chaves primárias.

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

Neste exemplo, o validador procurará no banco de dados por um registro da classe User onde a coluna login possui o mesmo valor informado no campo para validar.

sfEmailValidator

verifica se um parâmetro contém um valor que qualifica como um e-mail

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

sfFileValidator

aplica restrições de formato (um array de tipos mime) e de tamanho para campos de upload de arquivos

```
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

verifica se um parâmetro contém um valor que qualifica como uma URL válida

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

sfDateValidator

verifica se um parâmetro está em um formato de data