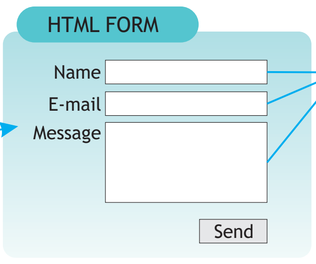
Forms - sfForm (Part I)

A symfony form (**sfForm**) is a class made of fields. Each field has a name, a validator and a widget.

HTML FORM



Form fields are configured in the **configure()** method using the **setValidators()** and **setWidgets()** methods

1 SYMFONY FORM: Define Widgets and Validators

```
class ContactForm extends sfForm {
  public function configure() {
    $this->setWidgets(array(
      'name'    => new sfWidgetFormInputText(),
      'email'   => new sfWidgetFormInputText(),
      'message' => new sfWidgetFormTextarea(),
    ));

    $this->setValidators(array(
      'email'   => new sfValidatorEmail(),
      'message' => new sfValidatorString(array('max_length' => 255)),
    ));
  }
}
```

lib/form/ContactForm.class.php

sfWidgetSchema render the form in XHTML by default. To switch to HTML use: sfWidget::setXhtml(false);

sfValidatorSchema validate input raw data and clean/convert the data to validated and consistent data.

2 ACTION

```
function executeIndex($request){
  $this->form = new ContactForm(array(/* initial values */));

  if ($request->isMethod('post')) {
    $this->form->bind($request->getParameter('contact'));
    if ($this->form->isValid()) {
      // do something with cleaned values
      $values = $this->form->getValues();
      $this->redirect('@somewhere');
    }
  }
}
```

apps/frontend/modules/mymodule/actions/actions.class.php

- `$this->form = new ContactForm(...)`: Creates a form and passes it to the view (using $this->).
- `if ($request->isMethod('post'))`: Checks if there is a POST request.
- `$this->form->bind(...)`: Bound to the request data and triggers the validation. The validation itself is managed by the validator schema, not the form.
- `if ($this->form->isValid())`: Checks if the form is valid.

3 TEMPLATE

```
<?php echo $form->renderFormTag('foo/contact') ?>
  <?php echo $form->renderHiddenFields() ?>
  <?php echo $form->renderGlobalErrors() ?>
  <table>
    <?php echo $form['name']->renderRow() ?>
    <?php echo $form['email']->renderRow() ?>
    <?php echo $form['message']->renderRow() ?>
    <tr>
      <td colspan="2">
        <input type="submit" />
      </td>
    </tr>
  </table>
</form>
```

apps/frontend/modules/mymodule/templates/indexSuccess.php

LOCATION OF THE FORM CLASSES

```
lib
  form       { your custom classes and propel classes (generated
               using the task: $ symfony propel:build-forms)
    doctrine { doctrine classes (generated using the task:
               $ php symfony doctrine:build --forms)
```

WHAT CAN I DO IN THE configure() METHOD?

UNSET FIELDS: unset

```
unset($this['created_at'], $this['updated_at'], $this['expires_at'] );
```

LIST THE FIELDS YOU WANT TO DISPLAY: useFields

The useFields() method does two things automatically: adds the hidden fields and the array of fields is used to change the fields order.

```
$this->useFields(array('id', 'type', 'description', 'token', 'is_public', 'email'));
```

DEFINE THE POSITION OF THE FIELDS: setPositions

```
$this->widgetSchema->setPositions(array('bodywork_id', 'mark_id',
'price', 'fuel_id', 'model_id'));
```

DEFINE WIDGETS: setWidget and setWidgets

```
$this->setWidgets(array('password' => new sfWidgetFormInputPassword()));
```

DEFINE A HELP MESSAGE FOR FIELDS: setHelp and setHelps

```
$this->widgetSchema->setHelp(
   'phone', 'Only numbers'
);
```

or

```
$this->widgetSchema->setHelps(array(
  'phone'     => 'Only numbers',
  'is_public' => 'Public?',
));
```

EMBED I18N OBJECTS `doctrine` `propel`

```
$this->embedI18n(array('pt', 'en', 'es'));
```

VALIDATE FORM FIELDS: setValidator and setValidators

```
$this->setValidator('filename', new sfValidatorFile(array('mime_types'=>'web_images')));
```

DEFINE CUSTOM LABEL FOR FIELDS: setLabel and setLabels

```
$this->widgetSchema->setLabel(
  'pt', 'Português'
);
```

or

```
$this->widgetSchema->setLabels(array(
  'category_id'    => 'Category',
  'is_public'      => 'Public?',
  'how_to_apply'   => 'How to apply?',
));
```

DEFINE DEFAULT VALUES FOR FIELDS: setDefault and setDeafults

```
$this->setDefault(
    'email', 'Your Email Here'
);
```

or

```
$this->setDefaults(array(
    'email' => 'Your Email Here',
    'name' => 'Your Name Here'
));
```

MODIFY THE NAME HTML ATTRIBUTE FOR ALL WIDGETS: setNameFormat

```
$this->widgetSchema->setNameFormat('search[%s]');
```

EMBED RELATIONS: embedRelation `doctrine` `propel`

```
$this->embedRelation('Games');
```

EMBED OTHER FORMS INTO THE CURRENT FORM: embedForm

```
$authorForm = new AuthorForm($this->getObject()->getAuthor());
$this->embedForm('Author', $authorForm);
```

MERGE FORMS: mergeForm

```
$this->mergeForm(new RegisterForm());
```