SBNDF&CMS

User Guide

How to create Simple Contact Form

By SBND Technologies

fcms.sbnd.net

1) Requirements

Let's say we need to make “contact us“ functionality. We will also need to save user contacts in database and send notification email to the administrator.

2) Create a component for Control Panel

First, we will create a component for Control Panel. With this component admin user will be able to manage the user contacts. New component can be created in directory root/cmp, but it is a good practice to create directories /back and /front in /cmp.

2.1) Create a php file root/cmp/back/ContactUs.cmp.php. The sub extension “cmp” is important, it tells the system that this php file is a component.

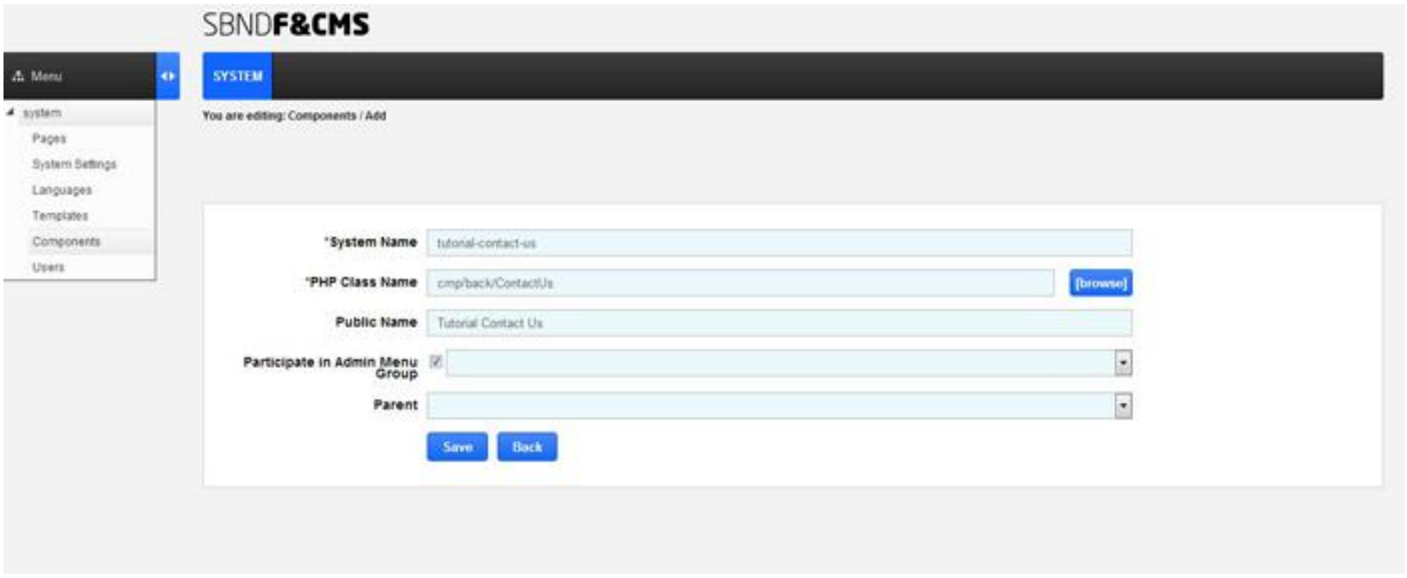
2.2) Declare a class ContactUs in the new file (the name of the class has to be the same as the file's name) that extends CmsComponent.

```php
class ContactUs extends CmsComponent{
  // define the name of the database table.
  public $base = 'totorial_contact_us';
  }

```

2.3) Override the method main() and declare two component fields. For more information about component fields see*component-fields-options.txt* in the documentation.

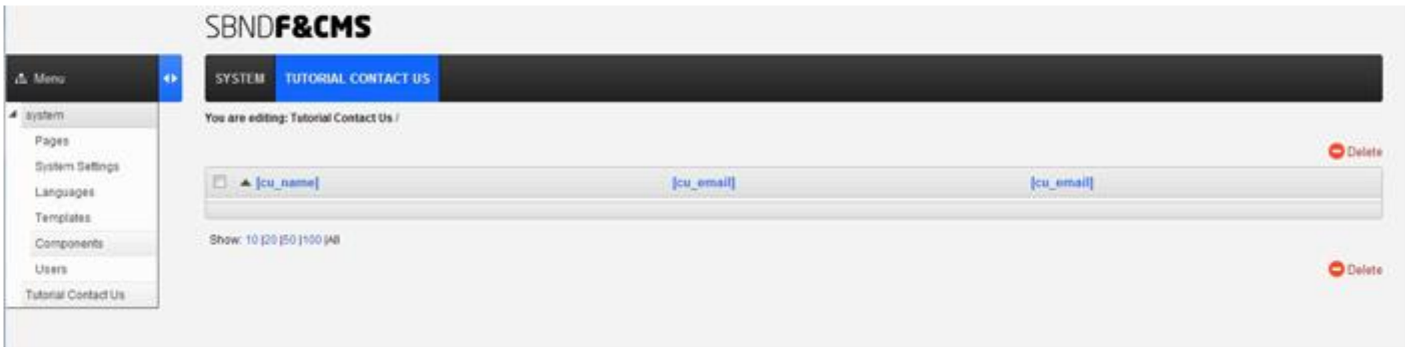
```
function main(){
  // call parent method for save main functionality.
  parent::main();

  $this->setField('name', array(
  'text' => BASIC_LANGUAGE::init()->get('cu_name'),
  'perm' => '*'
  ));
  $this->setField('email', array(
      'text' => BASIC_LANGUAGE::init()->get('cu_email'),
      'perm' => '*'
      ));
      $this->setField('body', array(
      'text' => BASIC_LANGUAGE::init()->get('cu_body'),
      'perm' => '*',
      'formtype' => ‘textarea',
      'dbtype' => 'text'
      ));
      }

```

2.4) Add the component in the control panel. Go to *Control Panel(cp) -> System -> Components -> Add* in Control Panel and fill in the fields.



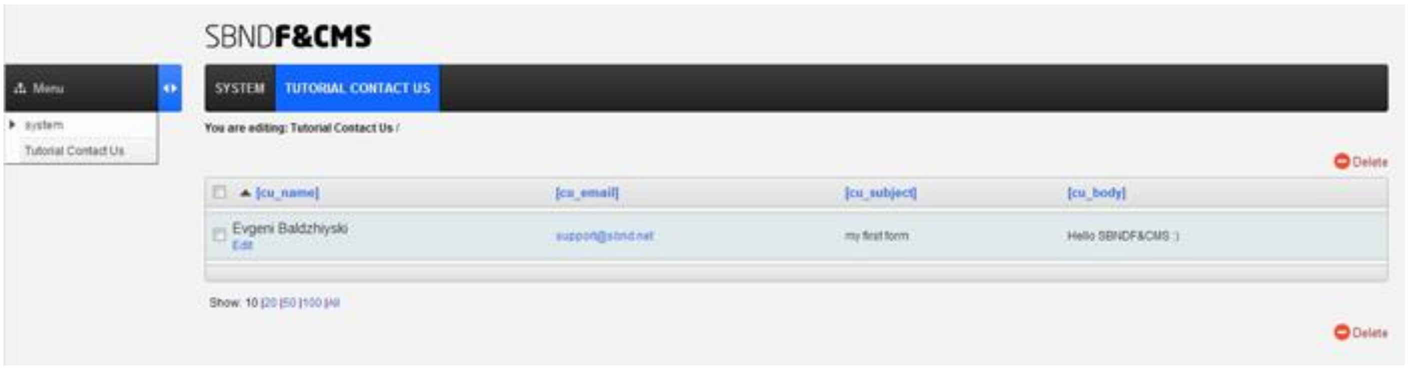
You are editing: Components / Add

*System Name	tutorial-contact-us
*PHP Class Name	cmp/back/ContactUs
Public Name	Tutorial Contact Us
Participate in Admin Menu Group	✓
Parent	



2.5) Default system List view shows columns for all components fields. We can change this.

```
function ActionList(){
  // do not need the admin to add new records
$this->delAction('add');
 // make the form only for read
$this->delAction('save');
// make sorting support
  $this->sorting = new BasicSorting('name', false, $this->prefix);

  $this->map('name', BASIC_LANGUAGE::init()->get('cu_name'), 'formater');
      $this->map('email', BASIC_LANGUAGE::init()->get('cu_email'), 'formater');
      $this->map('body', BASIC_LANGUAGE::init()->get('cu_email'), 'cu_body');

      return parent::ActionList();
      }
      function formater($val, $name, $row){
       // make click email write support
      if($name == 'email'){
      return '<a href="emilto:'.$val.'">'.$val.'</a>';
      }
      return $val;
      }
```



You are editing: Tutorial Contact Us /

	[cu_name]	[cu_email]	[cu_subject]	[cu_body]
☐	Evgeni Baldzhiyski	support@sbnd.net	my first form	Hello SBNDF&CMS :)

Done. We have finished the administrative part setup.

3) Create a component for client side

First, we have to create a component that extends the ContactUs (the previously created administrative component). We then have to create a template file with specific for the task formatting. Finally, we need to create a page where the Contact Us functionality will be presented.

3.1) Create a php file root/cmp/front/ContactUsFront.cmp.php

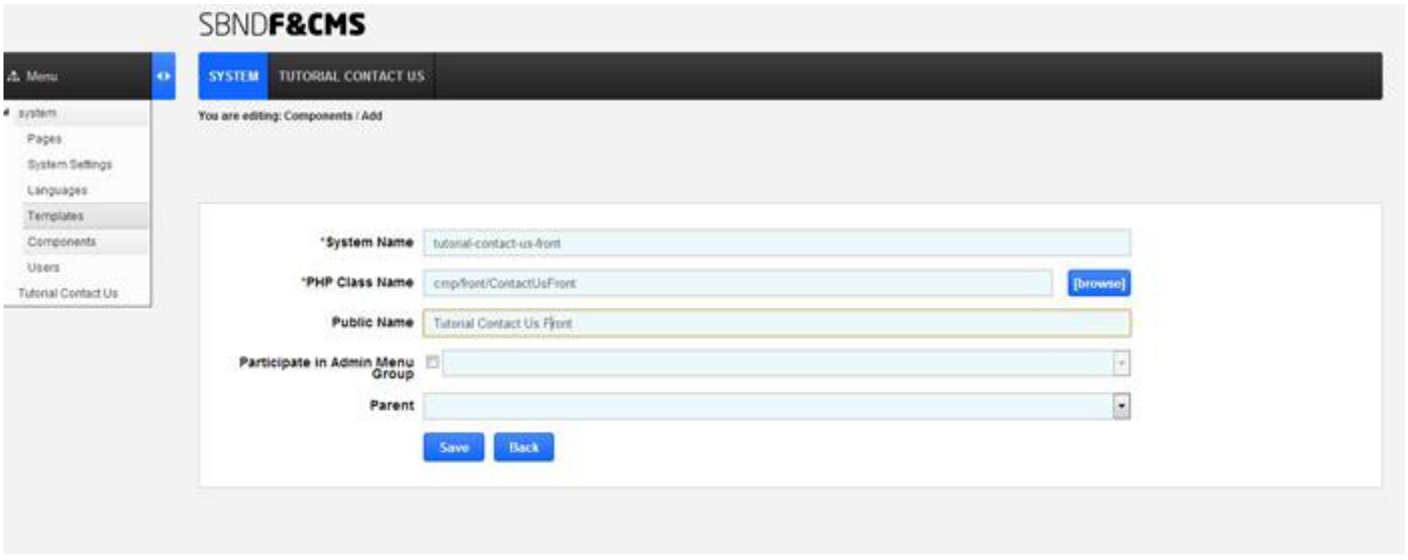
```
  BASIC::init()->imported("ContactUs", "cmp/back")
  class ContactUsFront extends ContactUs {
}
```

3.2) Configure the component to make form by default

```
function main(){
  parent::main();

  // make the form's name space. This will save form requ
est if in the same page have
  // more forms (login, search, other components form).
  $this->prefix = 'cu';

  // redirect default action to form creator.
  $this->updateAction('list', 'ActionFormAdd');

```

```
        // (optional) change text of the action cancel and use it like reset button.
        $this->updateAction('cancel', null, BASIC_LANGUAGE::init()->get('cu_reset'));

        // for security delete edit and delete actions
        $this->delAction('edit');
        $this->delAction('delete');

        // by default the system use action edit when exist error. Because we stop edit
        // action will change to action add.
        $this->errorAction = 'add';
        }

```

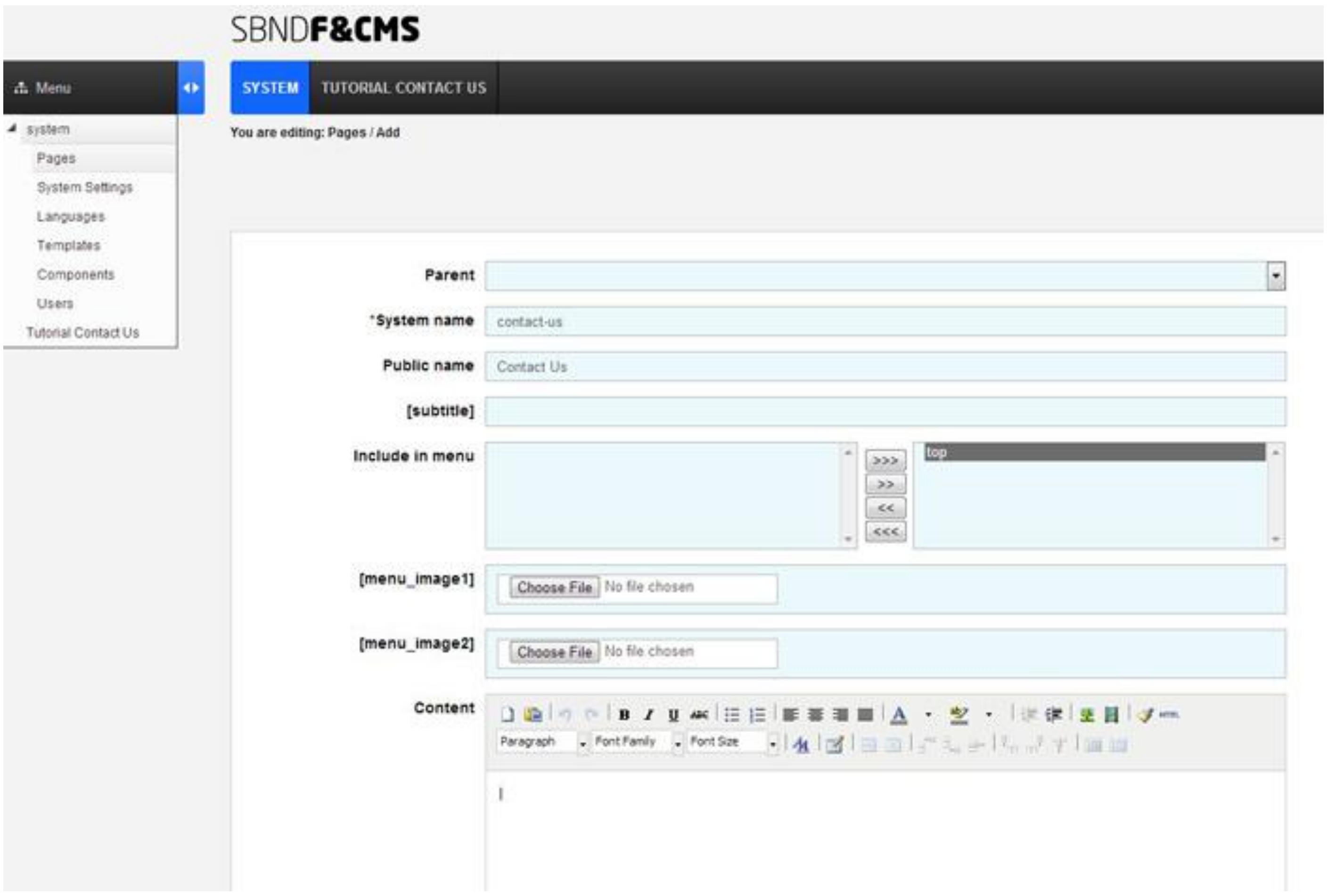
3.3) Open *Control Panel(cp) -> System -> Components -> Add* in Control Panel and fill the fields



You are editing: Components / Add

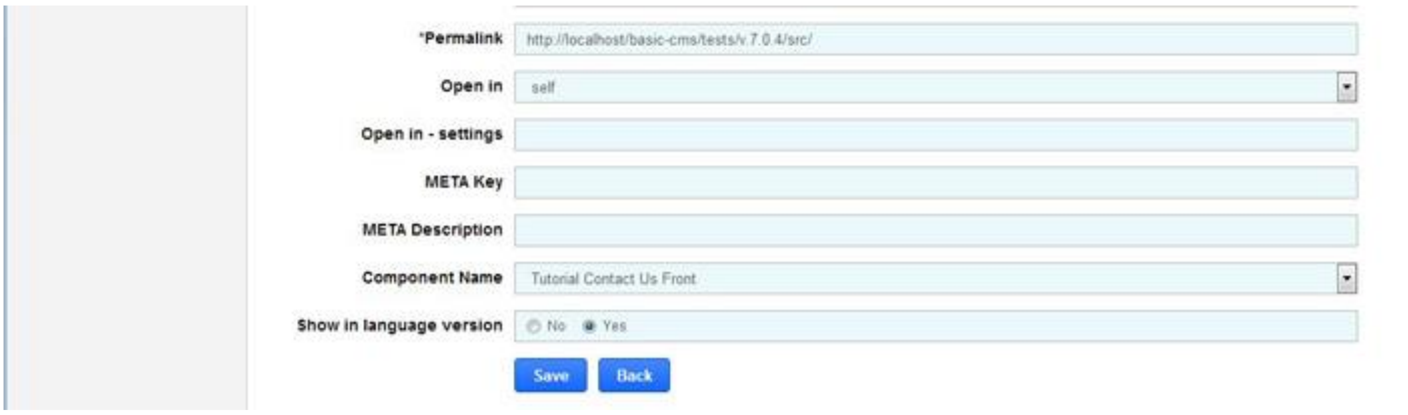
*System Name	tutorial-contact-us-front
*PHP Class Name	cmp/front/ContactUsFront
Public Name	Tutorial Contact Us Front
Participate in Admin Menu Group	
Parent	

3.4) Open *Control Panel(cp) -> System -> Pages -> Add* in Control Panel and create new page

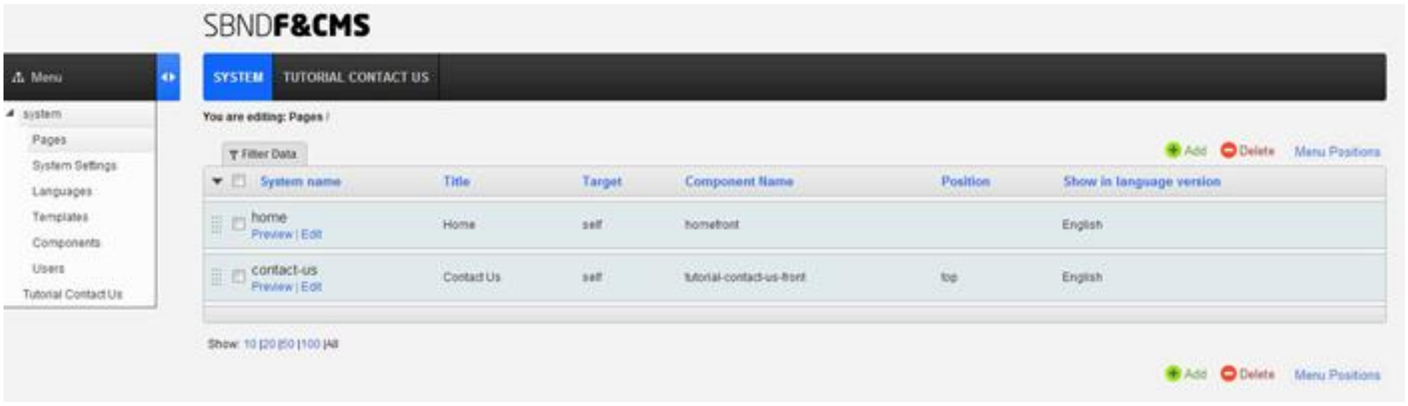


You are editing: Pages / Add

Parent	
*System name	contact-us
Public name	Contact Us
[subtitle]	
Include in menu	top
[menu_image1]	No file chosen
[menu_image2]	No file chosen
Content	



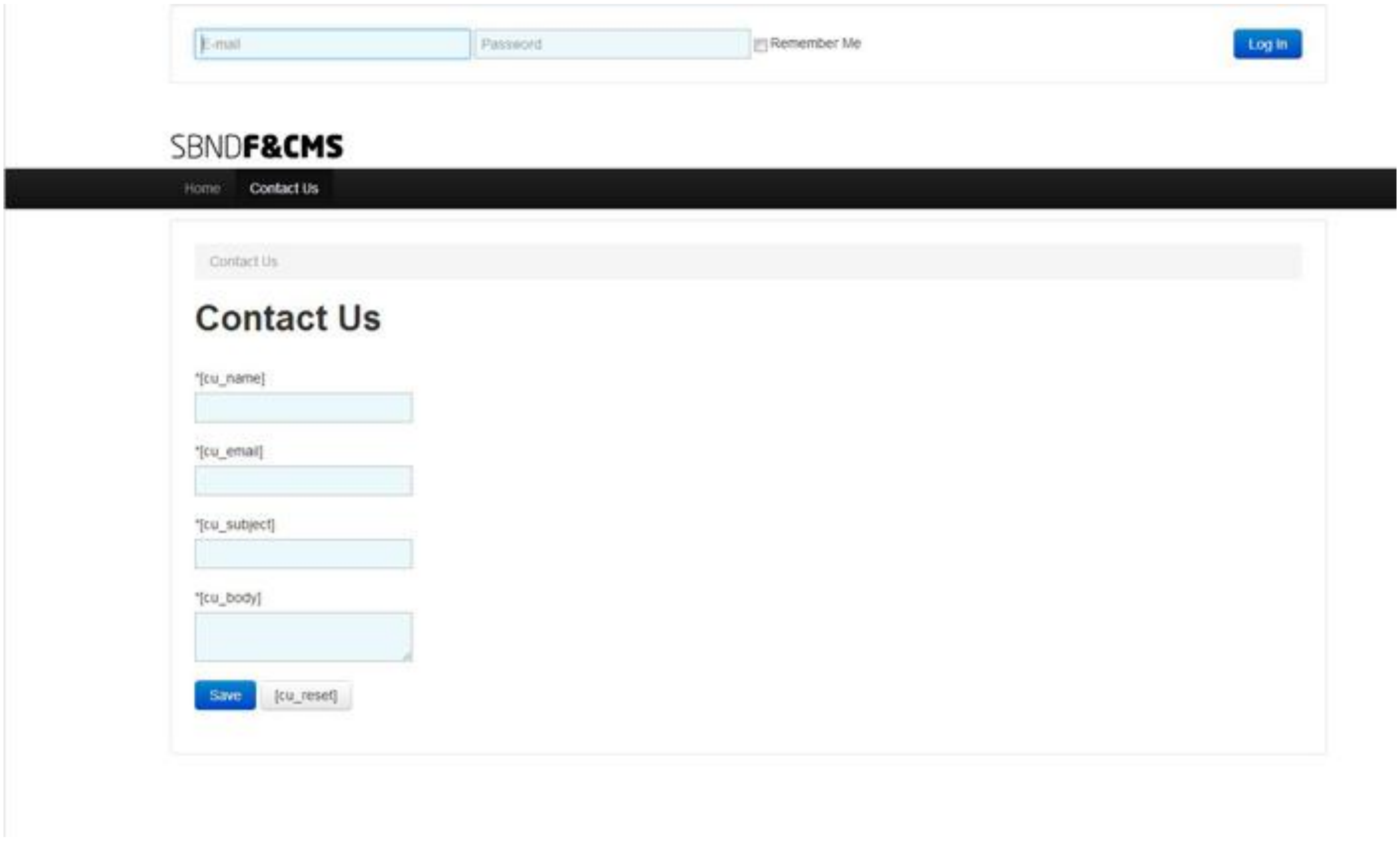
*Permalink	http://localhost/basic-cms/tests/v.7.0.4/src/
Open in	self
Open in - settings	
META Key	
META Description	
Component Name	Tutorial Contact Us Front
Show in language version	No [✓ Yes]



You are editing: Pages /

System name	Title	Target	Component Name	Position	Show in language version
home Preview | Edit	Home	self	homefront		English
contact-us Preview | Edit	Contact Us	self	tutorial-contact-us-front	top	English

Show: 10 \|20 \|50 \|100 \|All



Contact Us

*[cu_name]

*[cu_email]

*[cu_subject]

*[cu_body]

3.5) We now have two security issues - spam and email validation.

3.5.1) Spam

We will use capcha functionality in the form. In method main() add the following code:

```
// make new field in the form.
  $this->setField('spam', array(
  'text' => BASIC_LANGUAGE::init()->get('cu_spam'),
  'formtype' => 'capcha',
  'dbtype' => 'none',
  'messages' => array(
  2 => BASIC_LANGUAGE::init()->get('invalid_sec_code')
  )
  ));

      // show method that will call after the standart va
lidator.
```

```
      $this->specialTest = 'validator';

      Make new method:
    function validator(){
      if(
      strtolower(BASIC_GENERATOR::init()->getControl('capcha')->
      code($this->prefix.'spam'))
      !=
      strtolower($this->getDataBuffer('spam'))
      ){
      $this->setMessage('spam', 2);
      }
      }

```

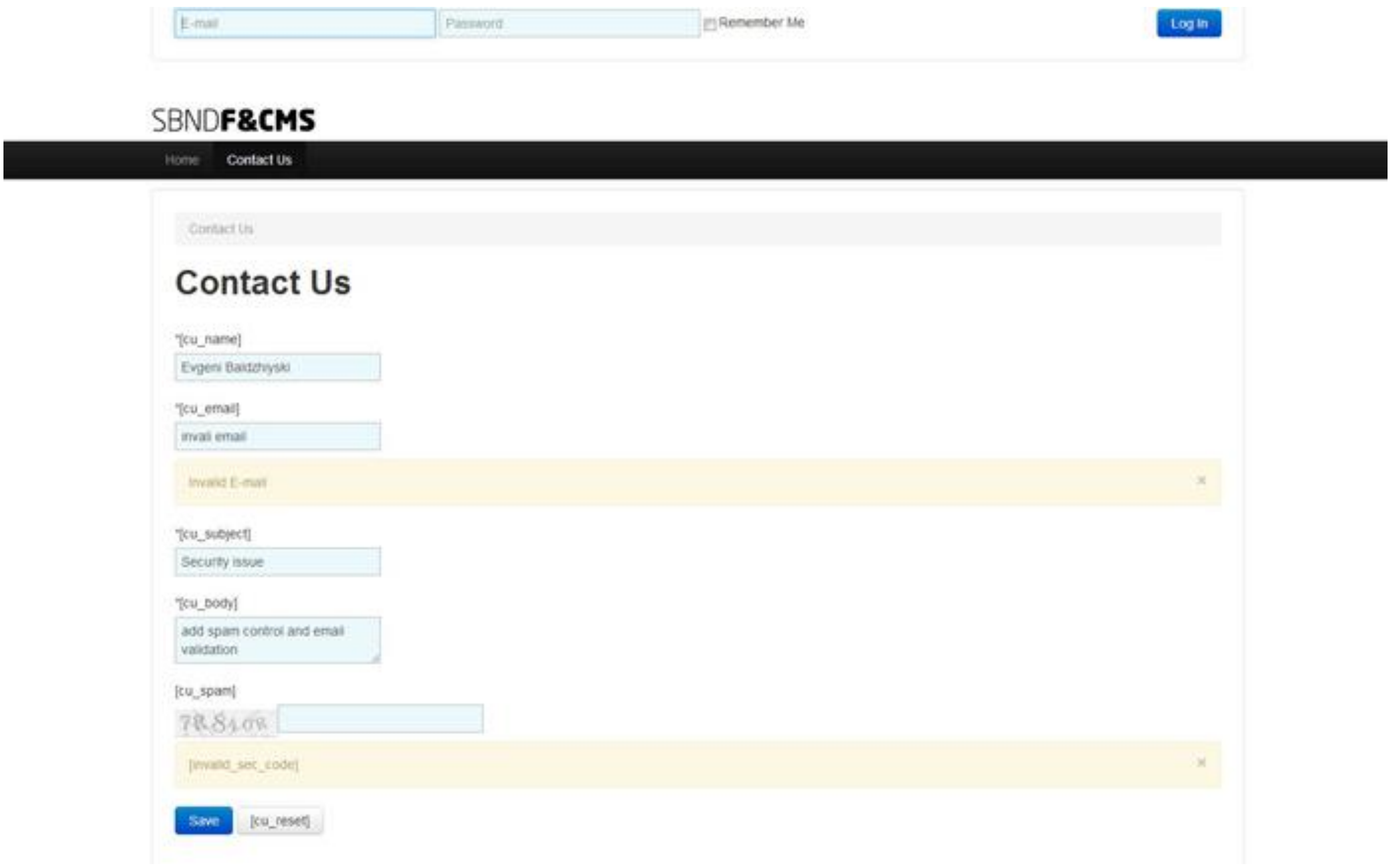
3.5.2) Email

We will add message in the email field. In method main() add the following code:

```
// add error message for invalid email
 $this->updateField('email', array(
 'messages' => array(
 2 => BASIC_LANGUAGE::init()->get('invalid_email')
 )
 ));
```

In the method "validator" add this code:

```
if(!BASIC::init()->validEmail($this->getDataBuffer('email'))){
$this->setMessage('email', 2);
}
```


E-mail | Password | Remember Me | Log In

SBNDF&CMS

Home | Contact Us

Contact Us

Contact Us

*[cu_name]
Evgeni Baldzhiyski

*[cu_email]
invali email

Invalid E-mail ×

*[cu_subject]
Security issue

*[cu_body]
add spam control and email validation

[cu_spam]
7R84c8

[invalid_sec_code] ×

Save | [cu_reset]

3.5.3) Finally, we have to send notification email to administrator

```php
// extend the action save handler with send email functionality.
  function ActionSave(){
  // import spam mode.
  BASIC::init()->imported('spam.mod');

  // make email from form request
  $mail = new BasicMail(
  $this->getDataBuffer('email'),
  $this->getDataBuffer('name'),
      array(
      'subject' => $this->getDataBuffer('subject'),
      'body' => $this->getDataBuffer('body')
      )
      );
      // send to support email.
      $mail->send('support@sbnd.net');
```

```

    // make session flag thet will show message.
    BASIC_SESSION::init()->set('contact_sended', 1);

    return parent::ActionSave();
    }
    // extend the form creator to support show message after save success.
    function ActionFormAdd(){
    $this->delAction('cancel');

    if(BASIC_SESSION::init()->get('contact_sended')){
    BASIC_SESSION::init()->un('contact_sended');

    BASIC_ERROR::init()->setMessage(
    BASIC_LANGUAGE::init()->get('contact_send_success'));
    }
    return parent::ActionFormAdd();
    }
    }

```