

User Guide

# Creating a Simple Contact Form

By SBND Technologies

## Description and Requirements

This example shows how to develop a "Contact Us" functionality. The functionality must be able to store user contacts in the database and send notification emails to the administrator.

## Creating a Component for the Administration Panel

First, you must create a new component for the administration panel. By using this component, the administrator will manage the user contacts. New components can be created directly in the directory *root/cmp*. However, we recommend that you create sub-directories */back* and */front* in the */cmp* directory. In this example the *root/cmp/back* and *root/cmp/front* directories are used.

1. Create a PHP file *ContactUs.cmp.php* in the *root/cmp/back* directory. The sub-extension *cmp* is important, as it tells the system that this PHP file is a component.
2. In the file declare a class named *ContactUs* (the name of the class has to be the same as the file's name). That class extends the *CmsComponent* class:

```
1. class ContactUs extends CmsComponent{
2.     // define the name of the database table.
3.     public $base = 'tutorial_contact_us';
4. }
```

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

```
1. function main(){
2.     // call parent method for save main functionality.
3.     parent::main();
4.
5.     $this->setField('name', array(
6.         'text' => BASIC_LANGUAGE::init()->get('cu_name'),
7.         'perm' => '*'
8.     ));
9.     $this->setField('email', array(
10.        'text' => BASIC_LANGUAGE::init()->get('cu_email'),
11.        'perm' => '*'
12.    ));
13.    $this->setField('body', array(
14.        'text' => BASIC_LANGUAGE::init()->get('cu_body'),
15.        'perm' => '*',
16.        'formtype' => 'textarea',
17.        'dbtype' => 'text'
18.    ));
19. }
```

4. Add the component in the administration panel. To do that, in the administration panel go to *System* → *Components*, click the *Add* button and enter the required information:

SBND **F&CMS**

Menu: system, Pages, System Settings, Languages, Templates, Components, Users

SYSTEM

You are editing: Components / Add

\*System Name: tutorial-contact-us

\*PHP Class Name: cmp/back/ContactUs [browse]

Public Name: Tutorial Contact Us

Participate in Admin Menu: ☒ Group: [dropdown]

Parent: [dropdown]

[Save] [Back]

SBND **F&CMS**

Menu: system, Pages, System Settings, Languages, Templates, Components, Users, Tutorial Contact Us

SYSTEM TUTORIAL CONTACT US

You are editing: Tutorial Contact Us /

	[cu_name]	[cu_email]	[cu_email]
Show: 10 [20] [50] [100] [All]			

5. The default system *List* view shows columns for all components fields. You can change this as shown below:

```

1. function ActionList(){
2.     // do not need the admin to add new records
3.     $this->delAction('add');
4.     // make the form only for read
5.     $this->delAction('save');
6.     // make sorting support
7.     $this->sorting = new BasicSorting('name', false, $this->prefix);
8.
9.     $this->map('name', BASIC_LANGUAGE::init()->get('cu_name'), 'formater');
10.    $this->map('email', BASIC_LANGUAGE::init()-
11.    >get('cu_email'), 'formater');
12.    $this->map('body', BASIC_LANGUAGE::init()-
13.    >get('cu_email'), 'cu_body');
14.
15.    return parent::ActionList();
16. }
17. function formater($val, $name, $row){
18.     // make click email write support
19.     if($name == 'email'){
20.         return '<a href="mailto:'.$val.'">'.$val.'</a>';
21.     }
22.     return $val;
23. }

```

## Creating a Component for the Client Side

First, you create a component that extends the *ContactUs* class (the previously created administrative component). After that you create a template file with specific formatting for the task. Finally, create a page where the *Contact Us* functionality will be presented.

1. Create a PHP file *ContactUsFront.cmp.php* in the *root/cmp/front* directory:

```
1. BASIC::init()->imported("ContactUs.cmp", "cmp/back")
2. class ContactUsFront extends ContactUs {
3. }
```

2. Configure the component to create a form by default:

```
1. function main(){
2.     parent::main();
3.
4.     // make the form's name space. This will save form request if in the same page have
5.     // more forms (login, search, other components form).
6.     $this->prefix = 'cu';
7.
8.     // redirect default action to form creator.
9.     $this->updateAction('list', 'ActionFormAdd');
10.
11.    // (optional) change text of the action cancel and use it like reset button.
12.    $this->updateAction('cancel', null, BASIC_LANGUAGE::init()->get('cu_reset'));
13.
14.    // for security delete edit and delete actions
15.    $this->delAction('edit');
16.    $this->delAction('delete');
17.
18.    // by default the system use action edit when exist error. Because we stop edit
19.    // action will change to action add.
20.    $this->errorAction = 'add';
21. }
```

3. In the administration panel go to *System* → *Components*, click the *Add* button and enter the required information:

SBND F&CMS

Menu

- system
- Pages
- System Settings
- Languages
- Templates
- Components
- Users
- Tutorial Contact Us

SYSTEM TUTORIAL CONTACT US

You are editing: Components / Add

\*System Name: tutorial-contact-us-front

\*PHP Class Name: cmp/front/ContactUsFront [browse]

Public Name: Tutorial Contact Us Front

Participate in Admin Menu Group: ☐

Parent: [dropdown]

Save Back

4. Create a new page for the *Contact Us* form. To do this, in the administration panel go to *System* → *Pages*, click the *Add* button and enter the required information for the new page:

SBND F&CMS

Menu

- system
- Pages
- System Settings
- Languages
- Templates
- Components
- Users
- Tutorial Contact Us

SYSTEM TUTORIAL CONTACT US

You are editing: Pages / Add

Parent: [dropdown]

\*System name: contact-us

Public name: Contact Us

[subtitle]: [text area]

Include in menu: [checkbox] [dropdown] [top]

[menu\_image1]: [Choose File] No file chosen

[menu\_image2]: [Choose File] No file chosen

Content: [Rich Text Editor]

\*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

Save

Back

SBNDF&CMS

Menu

SYSTEM

TUTORIAL CONTACT US

system

Pages

System Settings

Languages

Templates

Components

Users

Tutorial Contact Us

You are editing: Pages /

Filter Date

System name

Title

Target

Component Name

Position

Show in language version

home

Home

self

homefront

English

contact-us

Contact Us

self

tutorial-contact-us-front

top

English

Show: 10 | 20 | 50 | 100 | All

Add

Delete

Menu

Email

Password

Remember Me

Log In

SBNDF&CMS

Home

Contact Us

Contact Us

Contact Us

\*[cu\_name]

\*[cu\_email]

\*[cu\_subject]

\*[cu\_body]

Save

[cu\_reset]

## Security Issues

There are two security issues that must be addressed - spam and email validation.

## Spam

To add captcha functionality in the form, add the following code in the *main()* method:

```
1. // make new field in the form.
2. $this->setField('spam', array(
3.     'text' => BASIC_LANGUAGE::init()->get('cu_spam'),
4.     'formtype' => 'captcha',
5.     'dbtype' => 'none',
6.     'messages' => array(
7.         2 => BASIC_LANGUAGE::init()->get('invalid_sec_code')
8.     )
9. ));
10.
11.     // show method that will call after the standart validator.
12.     $this->specialTest = 'validator';
13.
14.     // Make new method:
15.     function validator(){
16.         if(
17.             strtolower(BASIC_GENERATOR::init()->getControl('captcha')->
18.                 code($this->prefix.'spam'))
19.             !=
20.             strtolower($this->getDataBuffer('spam'))
21.         ){
22.             $this->setMessage('spam', 2);
23.         }
24.     }
```

## Email Validation

To add an error message for invalid emails, add the following code in the *main()* method:

```
1. // add error message for invalid email
2. $this->updateField('email', array(
3.     'messages' => array(
4.         2 => BASIC_LANGUAGE::init()->get('invalid_email')
5.     )
6. ));
```

In the method *validator()* add this code:

```
1. if(!BASIC::init()->validEmail($this->getDataBuffer('email'))){
2.     $this->setMessage('email', 2);
3. }
```

☐ Remember Me

**SBND F&CMS**

[Home](#)
[Contact Us](#)

Contact Us

## Contact Us

\*[cu\_name]

\*[cu\_email]

invalid E-mail

\*[cu\_subject]

\*[cu\_body]

[cu\_spam]

[invalid\_sec\_code]

## Sending Notification Emails to the Administrator

```

1. // extend the action save handler with send email functionality.
2. function ActionSave(){
3.     // import spam mode.
4.     BASIC::init()->imported('spam.mod');
5.
6.     // make email from form request
7.     $mail = new BasicMail(
8.         $this->getDataBuffer('email'),
9.         $this->getDataBuffer('name'),
10.         array(
11.             'subject' => $this->getDataBuffer('subject'),
12.             'body' => $this->getDataBuffer('body')
13.         )
14.     );
15.     // send to support email.
16.     $mail->send('support@sbnd.net');
17.
18.     // make session flag that will show message.
19.     BASIC_SESSION::init()->set('contact_sended', 1);
20.
21.     return parent::ActionSave();
22. }
23. // extend the form creator to support show message after save success.
24. function ActionFormAdd(){
25.     $this->delAction('cancel');

```



```
26.  
27.     if(BASIC_SESSION::init()->get('contact_sended')){  
28.         BASIC_SESSION::init()->un('contact_sended');  
29.  
30.         BASIC_ERROR::init()->setMessage(  
31.             BASIC_LANGUAGE::init()->get('contact_send_success'));  
32.     }  
33.     return parent::ActionFormAdd();  
34. }  
35. }
```