LearnCubiByExamplesC5
Learn Cubi By Examples Chapter 5 - Ticket Comments and History

Updated Jan 17, 2012 by jack@openbiz.me

Learn Cubi By Examples Chapter 5 - Ticket Comments and History

At the end of previous chapter, we enhanced ticket list form by adding table join, as well as edit form by binding data to form elements. This chapter will add user comments and change history in ticket detail page.

Have a Separate Ticket Detail Page

By far, all ticket forms are organized in one page - the ticket list view. In real world, people like to have a separate page displaying individual ticket information so that this page can be bookmarked and forwarded in email.

Create Ticket Detail View

To make a new view, we simply copy trac/view/TicketListView.xml to trac/view/TicketDetailView.xml. Then edit the content to

```
<?xml version="1.0" standalone="no"?>
<EasyView Name="TicketDetailView" Description="Ticket details" Class="EasyView" Tab="" TemplateEngine="Smarty" TemplateFil
   <FormReferences>
       <Reference Name="trac.ticket.form.TicketDetailForm"/>
   </FormReferences>
</EasyView>
```

Then change navigation links in ticket list form to the ticket detail view. We modify the "fld_Id" and "fld_summary" element in trac/ticket/form/TicketListForm.xml

```
<Element Name="fld_Id" Class="ColumnText" FieldName="Id" Label="Id" Sortable="Y" AllowURLParam="N"/>
<Element Name="fld_summary" Class="ColumnText" FieldName="summary" Link="{@home:url}/trac/ticket_detail/{@:Elem[fld_Id].Va
```

Here we add a link attribute to summary element and remove the onclick action on Id element.

Now if you reload the ticket list view, then click on the summary link, you would see a separate ticket detail page with url like http://host/cubi/index.php/trac/ticket_detail/1

Add Ticket Change History

One of the key requirements of tracking the ticket change is to record and display all ticket change history. Whenever a change is made to a ticket, we want to record

- When and Who made the change
- Which fields were change. What are the old and new values
- Any comment was added

As mentioned in Chapter 1, table "trac_comments" is used to store ticket's comments and changes that are put in a serialized string in 'comments' column.

Again, we can use "gen_meta" script to generate comments metadata.

```
php gen_meta.php Default trac_comments trac.comments
```

Because ticket comments doesn't need a separate page, the view creation, access control and mod.xml change are skipped in metadata generation wizard. Then open trac/comments/do/CommentsDO.xml and edit it as

```
<?xml version="1.0" standalone="no"?>
<BizDataObj Name="CommentsDO" Description="" Class="BizDataObj" DBName="Default" Table="trac_comments" SearchRule="" SortI
    <BizFieldList>
        <BizField Name="Id" Column="id" Type="Number"/>
        <BizField Name="parent_id" Column="parent_id" Length="30" Required="Y" Type="Text"/>
        <BizField Name="time" Column="create_time" Required="N" Type="Datetime" ValueOnCreate="{date('Y-m-d H:i:s')}"/>
        <BizField Name="author_id" Column="author_id" Type="Number" ValueOnCreate="{@profile:Id}"/>
        <BizField Name="author" Join="user" Column="username" Type="Text" Required="N"/>
        <BizField Name="comments" Column="comments" Required="N" Type="Text"/>
    </BizFieldList>
    <TableJoins>
        <Join Name="user" Table="user" Column="id" ColumnRef="author_id" JoinType="LEFT JOIN"/></TableJoins>
    <ObjReferences>
    </ObjReferences>
</BizDataObj>
```

Please note that a table join is added so that the comment data object can have author username.

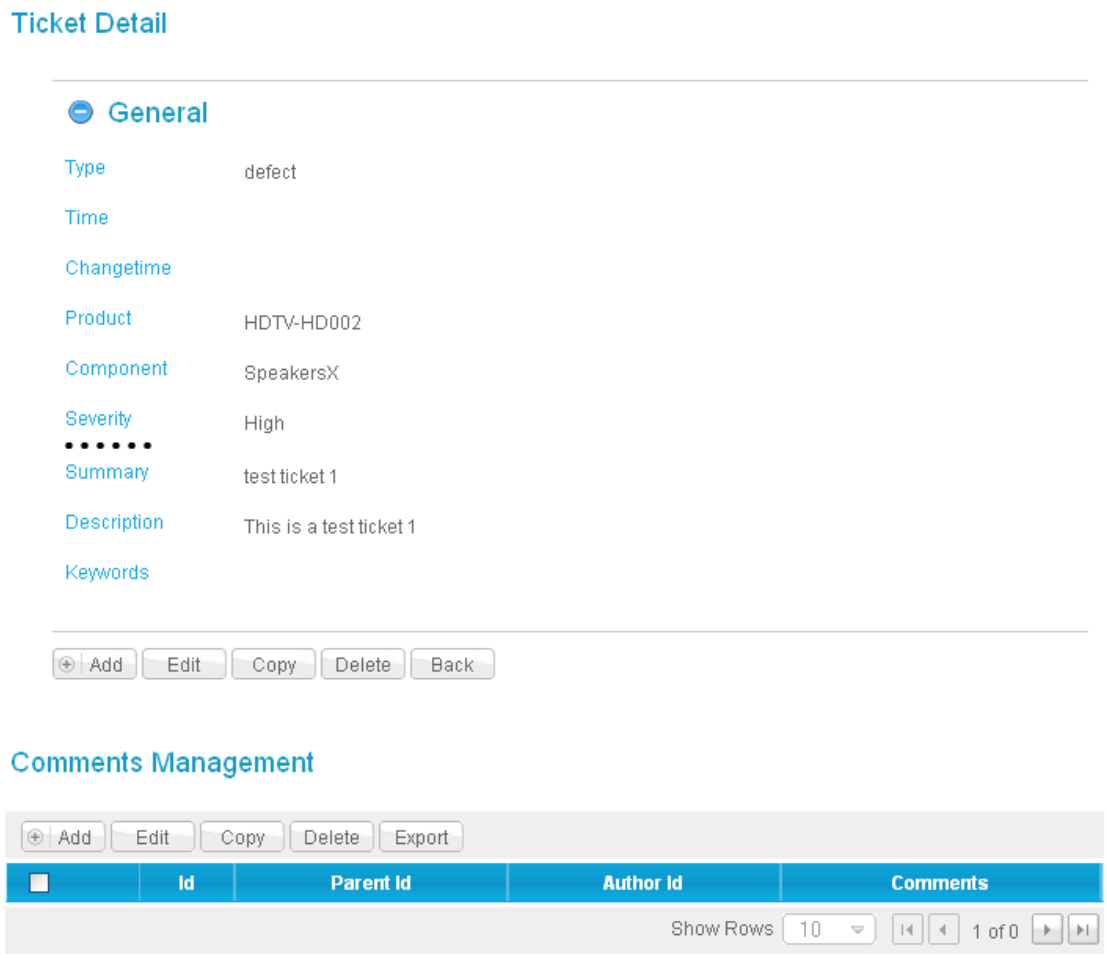
Now we can add the newly added comments list form to ticket detail view.

```
<?xml version="1.0" standalone="no"?>
<EasyView Name="TicketDetailView" Description="Ticket details" Class="EasyView" Tab="" TemplateEngine="Smarty" TemplateFil
   <FormReferences>
       <Reference Name="trac.ticket.form.TicketDetailForm"/>
       <Reference Name="trac.comments.form.CommentsListForm"/>
   </FormReferences>
</EasyView>
```

And change CommentsListForm.xml's access attribute as same as TicketDetailView.xml to "trac.ticket.Access " like below code

```
<EasyForm Name="CommentsListForm" Class="EasyForm" FormType="List" jsClass="jbForm" Title="Comments Management" Descriptic
```

The ticket detail looks like



Customize Form Logic

The just created comment list form obviously is not what we expect to see in ticket change history. We want to resolve two things:

1. How to save the change history from ticket record change into a serialized string in comments table.
2. How to display the serialize comment string into an easy understandable format.

Also we want to send a notification email of ticket creation or change to ticket owners.

These logic cannot be covered in core Openbiz-Cubi EasyForm class. Thus Cubi recommends implementing a custom class extending from EasyForm class.

Ticket Form

To have an user-defined class (TicketForm class), we simply do

- make a new file called TicketForm.php under /trac/ticket/form/, the same directory of TicketListForm.xml.
- extend TicketForm from EasyForm class.
- change the "Class" attribute in TicketEditForm.xml and TicketDetailForm.xml from "EasyForm" to "TicketForm".
- then fill in the custom logic in methods.

As there is special logic in ticket "save" actions, we override *doInsert and* doUpdate methods in TicketForm.

```php
<?php
include_once (MODULE_PATH.'/trac/email/TicketEmailService.php');

class TicketForm extends EasyForm
{
        protected $commentsDOName = "trac.comments.do.CommentsDO";
        protected $versionDOName = "trac.version.do.VersionDO";
        protected $milestoneDOName = "trac.milestone.do.MilestoneDO";
        protected $productDOName = "trac.product.do.ProductDO";
        protected $componentDOName = "trac.component.do.ComponentDO";
        protected $userDOName = "system.do.UserDO";

        protected $ticketEmailSvc = "trac.email.TicketEmailService";

        protected function _doInsert($inputRecord)
        {
        $emailSvc = BizSystem::getObject($this->ticketEmailSvc);

        parent::_doInsert($inputRecord);

        $ticketRec = $this->getActiveRecord($ticket_id);

        // send email to ticket owner and cc
        $emailSvc->notifyNewTicket($ticketRec);

        return true;
        }

        protected function _doUpdate($inputRecord, $currentRecord)
        {
        $emailSvc = BizSystem::getObject($this->ticketEmailSvc);

        parent::_doUpdate($inputRecord, $currentRecord);

        // get audit dataobj
        $commentsDO = BizSystem::getObject($this->commentsDOName);
        if (!$commentsDO) {
            return;
        }
        $ticket_id = $currentRecord['Id'];

        // compose the comments data in serialized array field=>(oldval, newval)
        $data = array();

        // reform the record so that the history doesn't include change of field_id
        $this->reformRecord($inputRecord, $currentRecord);

        foreach ($inputRecord as $fldName=>$fldVal)
        {
            $oldVal = $currentRecord[$fldName];
            if ($oldVal == $fldVal)
                continue;

            $data[$fldName] = array('old'=>$oldVal, 'new'=>$fldVal);
        }
        $comment = BizSystem::clientProxy()->getFormInputs("fld_comments");
        if (!empty($comment))
            $data['comment'] = $comment;

        if (empty($data))
            return;

        // save to comment do
        $dataRec = new DataRecord(null, $commentsDO);
        $dataRec['parent_id'] = $ticket_id;
        $dataRec['comments'] = serialize($data);
        try {
            $dataRec->save();
        }
        catch (BDOException $e)
        {
            $this->processBDOException($e);
            return;
        }

        $ticketRec = $this->getActiveRecord($ticket_id);

        $this->runEventLog();

        // send email to ticket owner and cc
        $commentRec['author_id'] = $dataRec['author_id'];
        $commentRec['time'] = $dataRec['time'];
        $commentRec['changes'] = $data;
        $emailSvc->notifyChangeTicket($ticketRec, $commentRec);

        return true;
        }

        // special logic for version_id, product_id, component_id, milestone_id, owner_id
        // get the field name value and append it in the array
        protected function reformRecord(&$inputRecord, $currentRecord)
        {
        // version_id.
        if ($inputRecord['version_id']!=$currentRecord['version_id']) {
            $versionDO = BizSystem::getObject($this->versionDOName);
            $version = $versionDO->fetchById($inputRecord['version_id']);
            $inputRecord['version'] = $version['name'];
            unset($inputRecord['version_id']);
        }

        // milestone_id
        if ($inputRecord['milestone_id']!=$currentRecord['milestone_id']) {
```

```
            $milestoneDO = BizSystem::getObject($this->milestoneDOName);
            $milestone = $milestoneDO->fetchById($inputRecord['milestone_id']);
            $inputRecord['milestone'] = $milestone['name'];
            unset($inputRecord['milestone_id']);
        }

        // product_id
        if ($inputRecord['product_id']!=$currentRecord['product_id']) {
            $productDO = BizSystem::getObject($this->productDOName);
            $product = $productDO->fetchById($inputRecord['product_id']);
            $inputRecord['product'] = $product['name'];
            unset($inputRecord['product_id']);
        }

        // component_id
        if ($inputRecord['component_id']!=$currentRecord['component_id']) {
            $componentDO = BizSystem::getObject($this->componentDOName);
            $component = $componentDO->fetchById($inputRecord['component_id']);
            $inputRecord['component'] = $component['name'];
            unset($inputRecord['component_id']);
        }

        // owner_id
        unset($inputRecord['owner_id']);
    }
}
?>
```

In the second line, TicketEmailService.php is included. This service, will be discussed shortly, is used to send ticket emails.

Ticket Change History Form Template

In display of ticket change history, a special list is needed instead of the standard Cubi list form look. In the list, each change history record needs to show:

- When did the change happen
- Who did make the change
- List changes of each field with old and new values
- Show comments added

Open and edit the trac/comments/CommentsListForm.xml as

```
<?xml version="1.0" encoding="UTF-8"?>
<EasyForm Name="CommentsListForm" Class="EasyForm" FormType="List" jsClass="Openbiz.Form" Title="Change history" Descripti
    <DataPanel>
        <Element Name="fld_Id" Class="ColumnText" FieldName="Id" Label="Id" Sortable="Y"/>
        <Element Name="fld_parent_id" Class="ColumnText" FieldName="parent_id" Label="Parent Id"  Sortable="Y"/>
        <Element Name="fld_time" Class="ColumnText" FieldName="time" Label="Time"  Sortable="Y"/>
        <Element Name="fld_author" Class="ColumnText" FieldName="author" Label="Author"  Sortable="Y"/>
        <Element Name="fld_comments" Class="RawData" FieldName="comments" Label="Comments" UnSerialize="Y" Sortable="Y"/>
    </DataPanel>
    <ActionPanel>
    </ActionPanel>
    <NavPanel>
    </NavPanel>
    <SearchPanel>
    </SearchPanel>
</EasyForm>
```

Note in the form xml, we

- assign a new template file as "ticket_history.tpl"
- change the Class of comments field to "RawData" and set UnSerialize attribute to "Y".

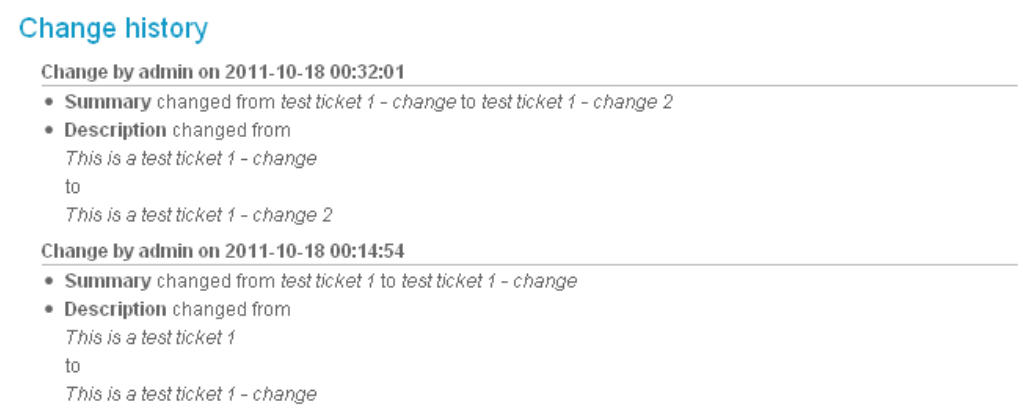
The ticket_history.tpl (under /trac/template/) template file actually quite simple. It understands the array un-serialized from the "comments" field.

```
<form id='{$form.name}' name='{$form.name}'>
<div style="padding-left:25px;padding-right:40px;">
    <div>
    {if $form.icon !='' }
    <div class="form_icon"><img  src="{$image_url}/{$form.icon}" border="0" /></div>
    {/if}
    <h2>
    {$form.title}
    </h2>
    {if $form.description !='' }
    <p class="form_desc">{$form.description}</p>
    {/if}
    </div>

<!-- table start -->
<div style="margin-left:15px;padding-bottom:10px">
{foreach item=row from=$dataPanel.data}
    <div style="font-weight:bold;border-bottom:1px solid #BBB;width:90%">
    Change by {$row.fld_author} on {$row.fld_time}
    </div>
    {foreach item=fld key=fldname from=$row.fld_comments}
        {if $fldname != 'comment'}
        <li style="margin-left:15px;"><b>{$fldname|capitalize}</b> changed from <i>{$fld.old}</i> to <i>{$fld.new}</i>
        {/if}
    {/foreach}
    <p style="margin-left:15px;padding-bottom:5px;">{$row.fld_comments.comment}</p>
{/foreach}
</div>
```

```
</div>
</form>
```

The ticket change history now has a new look.



There is a small bug here. When the save button is clicked on the ticket edit form, the form switches to ticket detail form. The change is saved, but the history is not update. You'd have to reload the page to see the new history. This is because "Save" button only switch the ticket edit form, not refresh the comments form. To fix it, we make the RedirectPage of the save button in TicketEditForm.xml.

```
    <ActionPanel>
        <Element Name="btn_save" Class="Button" Text="Save" CssClass="button_gray_m">
            <EventHandler Name="save_onclick" Event="onclick" EventLogMsg="" Function="UpdateRecord()" RedirectPage="{@ho
        </Element>
        <Element Name="btn_cancel" Class="Button" Text="Cancel" CssClass="button_gray_m">
            <EventHandler Name="btn_cancel_onclick" Event="onclick" Function="SwitchForm()" ShortcutKey="Escape" ContextM
        </Element>
    </ActionPanel>
```

Ticket Email Service

Typically pure backend logic (like email sending) will be put into a service class. A service class can be put in any folder that makes logic sense to developers. Openbiz Framework Service Chapter provides more information of creating a service.

In the service metadata file, it defines two template files - new ticket email and ticket change email.

```
<?xml version="1.0" standalone="no"?>
<PluginService Name="TicketEmailService" Package="service" Class="TicketEmailService"
               BizDataObj="email.do.EmailQueueDO"
               SendtoQueue="Y" >

    <Template   Name="NewTicketEmail"
               Title="Ticket #{$id} [New]: {$summary}"
               EmailAccount="TicketNotifier"
               Template="newticket_email.tpl" />

    <Template   Name="ChangeTicketEmail"
               Title="Ticket #{$id} [Change]: {$summary}"
               EmailAccount="TicketNotifier"
               Template="changeticket_email.tpl" />

</PluginService>
```

Also the email account "TicketNotifier" needs to be added into /cubi/modules/service/emailService.xml.

```
<?xml version="1.0" standalone="no"?>
<PluginService Name="emailService" Package="service" Class="emailService">
   <Accounts>
        <!-- use SMTP server -->
        <Account Name="System" Host="smtp.yourcompany.com" FromName="admin" FromEmail="admin@yourcompany.com" IsSMTP="Y" S
        <!-- use default unix sendmail function -->
        <Account Name="SystemNotifier" Host="" FromName="System Notification" FromEmail="notify@yourcompany.com" IsSMTP="N
        <Account Name="TicketNotifier" Host="" FromName="Ticket Notification" FromEmail="ticket@yourcompany.com" IsSMTP="N
   </Accounts>
   <Logging Type="DB" Object="email.do.EmailLogDO" Enabled="Y" />
</PluginService>
```

Here is the changeticket_email.tpl

```
<html>
{$ticket.url}

{$comment.author} <{$comment.author_email}> changed:

<table>
<tr>
        <th>what</th>
        <th>Removed</th>
        <th>Added</th>
```

```
</tr>
{foreach item=item key=itemName from=$comment.changes}
{if $itemName != 'comment'}
<tr>
        <td>{$itemName|capitalize}</td>
        <td>{$item.old}</td>
        <td>{$item.new}</td>
</tr>
{/if}
{/foreach}
</table>

--- Comment from {$comment.author} <{$comment.author_email}> {$comment.time} ---
{$comment.changes.comment}

------- You are receiving this mail because: ------- You are the {$trac_role} for the bug.
</html>
```

Here is the ticket email service code.

```
<?php

include_once(MODULE_PATH."/service/userEmailService.php");

class TicketEmailService extends userEmailService
{
    public function notifyNewTicket($ticketRec)
    {
        // ticket url
        $ticketRec['url'] = SITE_URL.APP_INDEX.'/ticket/detail/'.$ticketRec['Id'];

        //init email info
        $template = $this->m_Tempaltes["NewTicketEmail"]["TEMPLATE"];
        $subject  = $this->m_Tempaltes["NewTicketEmail"]["TITLE"];
        $sender   = $this->m_Tempaltes["NewTicketEmail"]["EMAILACCOUNT"];
        $subject = str_replace('{$id}',$ticketRec['Id'],$subject);
        $subject = str_replace('{$summary}',$ticketRec['summary'],$subject);

        $template = BizSystem::getTplFileWithPath($template, "trac");

        $userObj = BizSystem::getObject("system.do.UserDO");
        // send to owner
        $user_id = $ticketRec['owner_id'];
        $data = $userObj->directFetch("[Id]='".$user_id."'", 1);
        if(count($data)) {
            $userData = $data[0];
            $data = array("ticket"=>$ticketRec, "trac_role"=>'owner');

            //render the email tempalte
            $content = $this->RenderEmail($data, $template);

            //prepare recipient info
            $recipient['email'] = $userData['email'];
            $recipient['name']  = $userData['username'];

            //send it to the queue
            $result = $this->sendEmail($sender,$recipient,$subject,$content);
        }
        //email to reporter
        $user_id = $ticketRec['reporter_id'];
        $data = $userObj->directFetch("[Id]='".$user_id."'", 1);
        if(count($data)) {
            $userData = $data[0];
            $data = array("ticket"=>$ticketRec, "trac_role"=>'reporter');

            //render the email tempalte
            $content = $this->RenderEmail($data, $template);

            //prepare recipient info
            $recipient['email'] = $userData['email'];
            $recipient['name']  = $userData['username'];

            //send it to the queue
            $result = $this->sendEmail($sender,$recipient,$subject,$content);
        }
        // email to cc
        if(!empty($ticketRec['cc'])) {
            $data = array("ticket"=>$ticketRec, "trac_role"=>'watcher');

            //render the email tempalte
            $content = $this->RenderEmail($data, $template);

            //prepare recipient info
            $recipient['email'] = $ticketRec['cc'];
            $recipient['name']  = $recipient['email'];

            //send it to the queue
            $result = $this->sendEmail($sender,$recipient,$subject,$content);
        }
    }

    public function notifyChangeTicket($ticketRec, $commentRec)
    {
        // ticket url
        $ticketRec['url'] = SITE_URL.APP_INDEX.'/ticket/detail/'.$ticketRec['Id'];

        //init email info
        $template = $this->m_Tempaltes["ChangeTicketEmail"]["TEMPLATE"];
        $subject  = $this->m_Tempaltes["ChangeTicketEmail"]["TITLE"];
        $sender   = $this->m_Tempaltes["ChangeTicketEmail"]["EMAILACCOUNT"];
        $subject = str_replace('{$id}',$ticketRec['Id'],$subject);
```

```
        $subject = str_replace('{$summary}',$ticketRec['summary'],$subject);

        $template = BizSystem::getTplFileWithPath($template, "trac");

        $userObj = BizSystem::getObject("system.do.UserDO");

        // get comment author record
        $user_id = $commentRec['author_id'];
        $data = $userObj->directFetch("[Id]='".$user_id."'", 1);
        if(count($data)) {
            $commentRec['author'] = $data[0]['username'];
            $commentRec['author_email'] = $data[0]['email'];
        }

        // send to owner
        $user_id = $ticketRec['owner_id'];
        $data = $userObj->directFetch("[Id]='".$user_id."'", 1);
        if(count($data)) {
            $userData = $data[0];
            $data = array("ticket"=>$ticketRec, "comment"=>$commentRec, "trac_role"=>'owner');

            //render the email tempalte
            $content = $this->RenderEmail($data, $template);

            //prepare recipient info
            $recipient['email'] = $userData['email'];
            $recipient['name']  = $userData['username'];

            //send it to the queue
            $result = $this->sendEmail($sender,$recipient,$subject,$content);
        }
        //email to reporter
        $user_id = $ticketRec['reporter_id'];
        $data = $userObj->directFetch("[Id]='".$user_id."'", 1);
        if(count($data)) {
            $userData = $data[0];
            $data = array("ticket"=>$ticketRec, "comment"=>$commentRec, "trac_role"=>'reporter');

            //render the email tempalte
            $content = $this->RenderEmail($data, $template);

            //prepare recipient info
            $recipient['email'] = $userData['email'];
            $recipient['name']  = $userData['username'];

            //send it to the queue
            $result = $this->sendEmail($sender,$recipient,$subject,$content);
        }
        // email to cc
        if(!empty($ticketRec['cc'])) {
            $data = array("ticket"=>$ticketRec, "comment"=>$commentRec, "trac_role"=>'watcher');

            //render the email tempalte
            $content = $this->RenderEmail($data, $template);

            //prepare recipient info
            $recipient['email'] = $ticketRec['cc'];
            $recipient['name']  = $recipient['email'];

            //send it to the queue
            $result = $this->sendEmail($sender,$recipient,$subject,$content);
        }

        // need to sent email to comment author?

    }
}
?>
```

Now that we have the ticket change history created, we will add attachment to ticket in the next chapter Add Ticket Attachment

Comment by chitwar...@gmail.com, Oct 22, 2011

---

1. cannot save any tickets reporting error on javascripts files and undefined variables(same with other components)

2. error in module start (named it document rather than any other thing) 3. three errors in firebug{CKEDITOR is not DEfined} this one is being Generated on all the forms that have textarea field

   elements defined for them

4. The Calendar is also not working on the fields that require us to have Date inputs, also this fail with the error

   (showCalendar is not Defined)

5. on the third lecture of "Add Components", o follow untill when we have to change the metadata in all the other

   components under the trac directory(trac/), but i cannot find any metadata to i.e a mod.xml for all the components all i have is trac/component_directory/DO && Form directories no mod.xml - what do i do

Comment by project member rockys...@gmail.com, Oct 22, 2011

Please get the source code from https://openbiz-cubi.googlecode.com/svn/trunk/docs/learnbyexamples. Don't use the trac module source https://openbiz-cubi.googlecode.com/svn/trunk/apps/trac in this book

Comment by chitwar...@gmail.com, Oct 23, 2011

thanks, let me try it out

Comment by thekenya...@gmail.com, Aug 20, 2012

This is very well done.

► Sign in to add a comment

Terms - Privacy - Project Hosting Help

Powered by Google Project Hosting