Plugin Development: Sending Email

Use our mailer class to make sending email through your plugin simpler.

Sending an email with our mailer class requires using an existing email template or setting up a new custom template. All emails are queued and sent when the 'Process email queue' scheduled task next runs. There are three methods available in our mailer class depending on the situation you need to send an email for.

Ticket Email

The sendTicketMail method lets you send an email related to a ticket to either the ticket user or the relevant operators. This method ensures that the ticket merge fields are available, as well as selecting the correct From name and email address based on the ticket department. The first parameter is the template ID, the second is the Ticket model, and the third is an array of additional data. This array must contain 'email_type' => EmailTemplate::OPERATOR if the email is being sent to operators, and can be left out if it is for the ticket user.

```
AppModulesCoreControllersMailerMailer::sendTicketMail(
    $templateId,    $ticket,    [        'email_type'    => AppModulesCore
ModelsEmailTemplate::OPERATOR,        'activity_log' => [
            'type'       => AppModulesCoreModelsActivityLog::
SYSTEM,            'rel_name'    => $ticket->number,
            'rel_id'     => $ticket->id,            'rel_route'  =>
 'ticket.operator.ticket.show',            'section'     => 'ticket.ticket',
            'event_name' => 'sent_email_to_operators'        ]    ]);
```

Operator Email

If you need to send a non-ticket related email to operators, you can use the sendOperatorMail method. The first parameter is the template ID, the second is a collection of operator (User) models, and the third is an array of additional data. If you have only a single operator to email, you can use collect([$operator]) to generate a valid collection.

```
AppModulesCoreControllersMailerMailer::
sendOperatorMail(    $templateId,    $operators,    [
        'activity_log' => [            'type'       => AppModulesCoreModels
ActivityLog::SYSTEM,            'event_name' =>
 'sent_email_to_operators'        ]    ]);
```

User Email

If you need to send a non-ticket related email to users, you can use the sendUserMail method. The first parameter is the template ID, the second is a collection of User models, and the third is an array of additional data. If you have only a single user to email, you can use collect([$user]) to generate a valid collection.

```
AppModulesCoreControllersMailerMailer::sendUserMail(
    $templateId,    collect([ $user ]),    [        'activity_log' => [
            'type'       => AppModulesCoreModelsActivityLog::
SYSTEM,            'rel_name'   => $user->formatted_name,
            'rel_id'     => $user->id,            'rel_route'  =>
 'user.operator.user.edit',            'section'    => 'user.user',
            'event_name' => 'sent_template_email_to'        ]    ]);
```

Operator or User Group Email

If you wish to send an email to an operator or user group, this can be done by fetching the users in that group and then calling either of the two functions above depending on who you are emailing.

```
$group = AppModulesUserModelsUserGroup::with('users')->find($id);
$users = $group->users;
```

Activity Logging

As you may have seen in our examples, it is possible to add an activity log message when an email is sent in addition to the email log entry it automatically adds. This is achieved by adding an array for the 'activity_log' key in our additional data array. Below are a list of available event_name values that you can use.

Event Name
sent_template_mail_to
sent_ticket_email_to_user
sent_email_to_operators
sent_ticket_email_to_operators

Related Values

Below is a list of the related values to use depending on the context.

Key	Ticket
rel_name	
rel_id	
rel_route	
section	

Online URL: https://docs.supportpro.vn/article/plugin-development-sending-email-227.html