# Lesson 13: Jasper Reports, E-mail Attachments

Created by Jacques Marais, last modified on Sep 13, 2018

> *As a Farmer I want to receive an invoice for each purchase. This is to be sent as an E-mail attachment. In addition the E-mail should make use of a custom template.*



## Lesson Outcomes

After this lesson you should:

- Be able to add a Jasper report to a Helium project where the intention is to e-mail the report
- Be able to use a Helium built-in function to send a mail with a Jasper report as attachment
- Know how to include custom e-mail templates in a DSL application
- Use a custom e-mail template when sending an e-mail

## App Use Case Scenario

After a farmer has made a purchase, the app should send an invoice in the form of a Jasper report by e-mail. The content of the e-mail should be customised for the specific farmer. This includes the file name of the attachment which should contain the farmer name and a timestamp for the purchase. The e-mail should be formatted using a custom e-mail template that has been provided as part of the app source.

## New & Modified App Files

**`./builtin-reports/farmer_purchase_invoice_report/FarmerPurchaseInvoice.jrxml`**

**`./web-app/presenters/farmer_ops/FarmerPurchases.mez`**

**`./web-app/email-templates/email_template.html`**

**`./web-app/images/mz-190x61.png`**

## Creating a Jasper Report for the Purpose of Sending by E-mail

A discussion of Jasper report development is out of the scope of this project. There are, however, several mechanisms used in Helium to integrate Jasper reports with Helium applications. Note that this discussion focuses on reports that will be sent from application logic as e-mail attachments. Helium also provides functionality for Jasper reports on the frontend. This, however, is discussed separately in Lesson 17.

- Any variable within local scope can be referenced as a parameter in the report

- Helium provides parameters for the database schema name and the id of the currently logged in app user through the `schemaName` and `USER_APP_ROLE_ID` parameters respectively.

In order for us to add our report to the project, we need to create a folder called **`builtin-reports`** under our project root directory. For each report we will have a subfolder with the appropriate jrxml file and any other resources, such as images, required by the report. In our case this means creating a subfolder called **`farmer_purchase_invoice_report`** and simply placing our **`FarmerPurchaseInvoice.jrxml`** file in it.

## Sending a Jasper Report as an E-mail Attachment

Now that we have a report in our project, we can add a call to the `Mez:email` built-in function in the `makePurchase` function of our `FarmerPurchases` unit. Note that since we reference a parameter called `farmerPurchaseId` in our report, we will have to create a variable with the same name in the same scope in which the `Mez:email` function is called:

```
uuid farmerPurchaseId = farmerPurchase._id;

Mez:email(
    farmer.emailAddress,
    "messaging.email.farmer_purchase_invoice_descriptic
    "messaging.email.farmer_purchase_invoice_subject",
    "messaging.email.farmer_purchase_invoice_body",
    "farmer_purchase_invoice_report/FarmerPurchaseInvoi
);
```

Note how similar the call is to what we used in Lesson 10. The only addition is the relative path to our Jasper report.

Even though it is not necessary for our use case, it is also possible to send multiple report attachments per e-mail. To do this simply add the relative paths of the additional reports as arguments to the `Mez:email` function.

```
Mez:email(
    farmer.emailAddress,
    "messaging.email.farmer_purchase_invoice_descriptic
    "messaging.email.farmer_purchase_invoice_subject",
    "messaging.email.farmer_purchase_invoice_body",
    "farmer_purchase_invoice_report/FarmerPurchaseInvoi
    "additional_farmer_report_1/AdditionalFarmerReport1
    "additional_farmer_report_2/AdditionalFarmerReport2
);
```

## Using a Custom Attachment Name

In order to use a custom name for the report attachment, we can provide a function that will be used to generate the file name and reference it as part of the arguments for the

`Mez:emailAttach` function. Our code therefore changes to use the following in order to send the mail:

```
Mez:emailAttach(
    farmer.emailAddress,
    "messaging.email.farmer_purchase_invoice_descriptic
    "messaging.email.farmer_purchase_invoice_subject",
    "messaging.email.farmer_purchase_invoice_body",
    {"farmer_purchase_invoice_report/FarmerPurchaseInvc
);
```

## Using a Custom E-mail Template

To change the layout, images or colours of the e-mail for e.g. custom branding purposes, a final enum parameter can be added to the `Mez:email` or `Mez:emailAttach` built-in functions to specify a custom (html) e-mail template. The enumerated type is `EMAIL_TEMPLATES`, with the specified enum member the html file name (without the extension), as included by the developer under the **`web-app/email-templates`** directory. The snippet below shows the usage of `Mez:emailAttach` with a custom template.

```
Mez:emailAttach(
    farmer.emailAddress,
    "messaging.email.farmer_purchase_invoice_description",
    "messaging.email.farmer_purchase_invoice_subject",
    "messaging.email.farmer_purchase_invoice_body",
    {"farmer_purchase_invoice_report/FarmerPurchaseInvoice.j
    EMAIL_TEMPLATES.email_template
);
```

Images for this template is to be included under **`web-app/images`**, and prefixed by **`cid`** in the html:

```
<img alt="logo" width="190" height="61" src="cid:myCustomLog
```

Please note that it is not mandatory to have the email-template folder. If one is not included the original default email template in service will be used to send messages.

To override this behaviour you can include just one template called **`default.html`** which will then be used for all the mails instead of the original default template. Just including the file will enable the behaviour without you having to explicitly mention the template when the email syntax is used. More customization will require you to indicate the template you want to use via above mentioned enumeration.

```
└── web-app
    ├── email-templates
    │   └── myEmailTemplate.html
    └── images
        └── myCustomLogo.png
```

Please refer to the default e-mail template as an example or starting point for your custom template: email_template.html. Note that it contains `${subject}`, `${body}` and `${applicationInstanceName}` placeholders that are replaced at runtime. In essence these placeholders should always be standard and included in any variation of custom template you introduce.

## Lesson Source Code

Lesson 13.zip

No labels