**Creating a Custom Controller with the Messaging Class**

At minimum, a custom controller that uses the Apex Messaging namespace needs a subject, a body, and a recipient for the email. You will need a page that acts as a form to fill out the subject and body and deliver the email.

Create a new page called sendEmailPage and use the following code:

|  |  |  |
| --- | --- | --- |
| 01 | <apex:page controller="sendEmail"> | |
| 02 | <apex:messages /> |

|  |  |  |
| --- | --- | --- |
| 03 | <apex:pageBlock title="Send an Email to Your | |
| 04 | {!account.name} Representatives"> |

|  |  |  |
| --- | --- | --- |
| 05 | <p>Fill out the fields below to test how you might send an email to a user.</p> | |
| 06 | <br /> |

|  |  |  |
| --- | --- | --- |
| 07 | <apex:dataTable value="{!account.Contacts}" var="contact" border="1"> | |
| 08 | <apex:column > |

|  |  |  |
| --- | --- | --- |
| 09 | <apex:facet name="header">Name</apex:facet> | |
| 10 | {!contact.Name} |

|  |  |  |
| --- | --- | --- |
| 11 | </apex:column> | |
| 12 | <apex:column > |

|  |  |  |
| --- | --- | --- |
| 13 | <apex:facet name="header">Email</apex:facet> | |
| 14 | {!contact.Email} |

|  |  |
| --- | --- |
| 15 | </apex:column> |
| 16 | </apex:dataTable> | |

|  |  |
| --- | --- |
| 17 |  |
| 18 | <apex:form > | |

|  |  |
| --- | --- |
| 19 | <br /><br /> |
| 20 | <apex:outputLabel value="Subject" for="Subject"/>:<br /> | |

|  |  |  |
| --- | --- | --- |
| 21 | <apex:inputText value="{!subject}" id="Subject" maxlength="80"/> | |
| 22 | <br /><br /> |

|  |  |
| --- | --- |
| 23 | <apex:outputLabel value="Body" for="Body"/>:<br /> |
| 24 | <apex:inputTextarea value="{!body}" id="Body" rows="10" cols="80"/> | |

|  |  |
| --- | --- |
| 25 | <br /><br /><br /> |
| 26 | <apex:commandButton value="Send Email" action="{!send}" /> | |

|  |  |
| --- | --- |
| 27 | </apex:form> |
| 28 | </apex:pageBlock> | |

|  |  |
| --- | --- |
| 29 | </apex:page> |

Notice in the page markup that the account ID is retrieved from the URL of the page. For this example to render properly, you must associate the Visualforce page with a valid account record in the URL. For example, if 001D000000IRt53 is the account ID, the resulting URL should be:

|  |  |
| --- | --- |
| 1 | [https://](NULL)Salesforce\_instance/apex/sendEmailPage?id=001D000000IRt53 |

[Displaying Field Values with Visualforce](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_quick_start_display_field_values.htm) has more information about retrieving the ID of a record.

The following code creates a controller named sendEmail that implements the Messaging.SingleEmailMessage class, and uses the contacts related to an account as recipients:

|  |  |
| --- | --- |
| 01 | public class sendEmail { |
| 02 | public String subject { get; set; } | |

|  |  |  |
| --- | --- | --- |
| 03 | public String body { get; set; } | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | private final Account account; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | // Create a constructor that populates the Account object | |
| 08 | public sendEmail() { |

|  |  |  |
| --- | --- | --- |
| 09 | account = [select Name, (SELECT Contact.Name, Contact.Email FROM Account.Contacts) | |
| 10 | from Account where id = :ApexPages.currentPage().getParameters().get('id')]; |

|  |  |  |
| --- | --- | --- |
| 11 | } | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | public Account getAccount() { | |
| 14 | return account; |

|  |  |  |
| --- | --- | --- |
| 15 | } | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | public PageReference send() { | |
| 18 | // Define the email |

|  |  |  |
| --- | --- | --- |
| 19 | Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage(); | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | String addresses; |
| 22 | if (account.Contacts[0].Email != null) | |

|  |  |
| --- | --- |
| 23 | { |
| 24 | addresses = account.Contacts[0].Email; | |

|  |  |  |
| --- | --- | --- |
| 25 | // Loop through the whole list of contacts and their emails | |
| 26 | for (Integer i = 1; i < account.Contacts.size(); i++) |

|  |  |
| --- | --- |
| 27 | { |
| 28 | if (account.Contacts[i].Email != null) | |

|  |  |
| --- | --- |
| 29 | { |
| 30 | addresses += ':' + account.Contacts[i].Email; | |

|  |  |  |
| --- | --- | --- |
| 31 | } | |
| 32 | } |

|  |  |  |
| --- | --- | --- |
| 33 | } | |
| 34 |  |

|  |  |  |
| --- | --- | --- |
| 35 | String[] toAddresses = addresses.split(':', 0); | |
| 36 |  |

|  |  |  |
| --- | --- | --- |
| 37 | // Sets the paramaters of the email | |
| 38 | email.setSubject( subject ); |

|  |  |  |
| --- | --- | --- |
| 39 | email.setToAddresses( toAddresses ); | |
| 40 | email.setPlainTextBody( body ); |

|  |  |
| --- | --- |
| 41 |  |
| 42 | // Sends the email | |

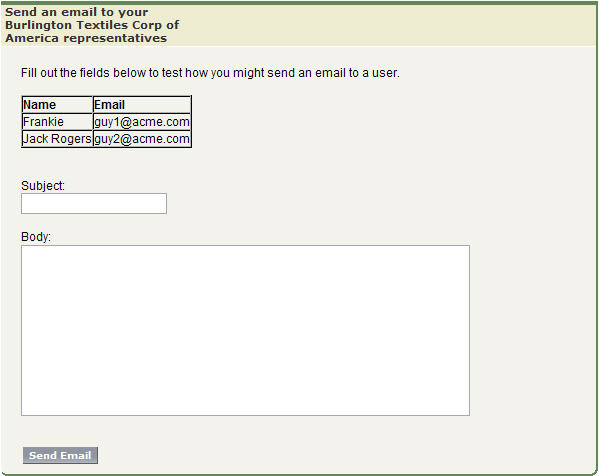
|  |  |
| --- | --- |
| 43 | Messaging.SendEmailResult [] r = |
| 44 | Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email}); | |

|  |  |
| --- | --- |
| 45 |  |
| 46 | return null; | |

|  |  |  |
| --- | --- | --- |
| 47 | } | |
| 48 | } |

Notice in the controller that:

* The subject and body of the email are set through a separate Visualforce page and passed into the controller.
* The method that sends the email is called send(). This name must match the name of the action for the Visualforce button that sends the email.
* The recipients of the email, that is, the email addresses stored in toAddresses[], come from the addresses of the contacts available in an associated account. When compiling a list of recipients from contacts, leads, or other records, it is a good practice to loop through all the records to verify that an email address is defined for each. The account ID is retrieved from the URL of the page.

**Example of the Form on sendEmailPage**

# Creating an Email Attachment

If you want to add an attachment to your email, you will need to add only a few lines of code to your custom controller. Email attachments are Blob file types. To create an attachment, you need to use the Apex Messaging.EmailFileAttachmentclass. You must define both the file name and the content of an EmailFileAttachment object.

## Adding a PDF Attachment

The following example demonstrates how to transform a PageReference to a Visualforce page rendered as a PDF into an email attachment. First, create a page called attachmentPDF:

|  |  |  |
| --- | --- | --- |
| 01 | <apex:page standardController="Account" renderAs="PDF"> | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | <h1>Account Details</h1> | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | <apex:panelGrid columns="2"> | |
| 06 |  |

|  |  |
| --- | --- |
| 07 | <apex:outputLabel for="Name" value="Name"/> |
| 08 | <apex:outputText id="Name" value="{!account.Name}"/> | |

|  |  |
| --- | --- |
| 09 |  |
| 10 | <apex:outputLabel for="Owner" value="Account Owner"/> | |

|  |  |  |
| --- | --- | --- |
| 11 | <apex:outputText id="Owner" value="{!account.Owner.Name}"/> | |
| 12 |  |

|  |  |
| --- | --- |
| 13 | <apex:outputLabel for="AnnualRevenue" value="Annual Revenue"/> |
| 14 | <apex:outputText id="AnnualRevenue" value="{0,number,currency}"> | |

|  |  |  |
| --- | --- | --- |
| 15 | <apex:param value="{!account.AnnualRevenue}"/> | |
| 16 | </apex:outputText> |

|  |  |
| --- | --- |
| 17 |  |
| 18 | <apex:outputLabel for="NumberOfEmployees" value="Employees"/> | |

|  |  |  |
| --- | --- | --- |
| 19 | <apex:outputText id="NumberOfEmployees" value="{!account.NumberOfEmployees}"/> | |
| 20 |  |

|  |  |  |
| --- | --- | --- |
| 21 | </apex:panelGrid> | |
| 22 |  |

|  |  |
| --- | --- |
| 23 | </apex:page> |

Note

See [Best Practices for Rendering PDF Files](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_compref_additional_render_pdf.htm#pages_compref_additional_render_pdf) for details of which components are recommended for use in PDF attachments.

Next, create the EmailFileAttachment object in the send() method of your custom controller. The following examples must be placed before calling Messaging.sendEmail:

|  |  |  |
| --- | --- | --- |
| 01 | // Reference the attachment page, pass in the account ID | |
| 02 | PageReference pdf = Page.attachmentPDF; |

|  |  |  |
| --- | --- | --- |
| 03 | pdf.getParameters().put('id',(String)account.id); | |
| 04 | pdf.setRedirect(true); |

|  |  |
| --- | --- |
| 05 |  |
| 06 | // Take the PDF content | |

|  |  |  |
| --- | --- | --- |
| 07 | Blob b = pdf.getContent(); | |
| 08 |  |

|  |  |
| --- | --- |
| 09 | // Create the email attachment |
| 10 | Messaging.EmailFileAttachment efa = new Messaging.EmailFileAttachment(); | |

|  |  |  |
| --- | --- | --- |
| 11 | efa.setFileName('attachment.pdf'); | |
| 12 | efa.setBody(b); |

If your SingleEmailMessage object is named email, then you associate the attachment like this:

|  |  |
| --- | --- |
| 1 | email.setFileAttachments(new Messaging.EmailFileAttachment[] {efa}); |

## Defining a Custom Component as an Attachment

By creating a custom component and using it on the Visualforce email form and to render the PDF for the email, users can see a preview of the content they are trying to send.

The following markup defines a custom component named attachment that represents the attachment for the email:

|  |  |  |
| --- | --- | --- |
| 01 | <apex:component access="global"> | |
| 02 | <h1>Account Details</h1> |

|  |  |
| --- | --- |
| 03 |  |
| 04 | <apex:panelGrid columns="2"> | |

|  |  |
| --- | --- |
| 05 |  |
| 06 | <apex:outputLabel for="Name" value="Name"/> | |

|  |  |  |
| --- | --- | --- |
| 07 | <apex:outputText id="Name" value="{!account.Name}"/> | |
| 08 |  |

|  |  |
| --- | --- |
| 09 | <apex:outputLabel for="Owner" value="Account Owner"/> |
| 10 | <apex:outputText id="Owner" value="{!account.Owner.Name}"/> | |

|  |  |
| --- | --- |
| 11 |  |
| 12 | <apex:outputLabel for="AnnualRevenue" value="Annual Revenue"/> | |

|  |  |  |
| --- | --- | --- |
| 13 | <apex:outputText id="AnnualRevenue" value="{0,number,currency}"> | |
| 14 | <apex:param value="{!account.AnnualRevenue}"/> |

|  |  |  |
| --- | --- | --- |
| 15 | </apex:outputText> | |
| 16 |  |

|  |  |
| --- | --- |
| 17 | <apex:outputLabel for="NumberOfEmployees" value="Employees"/> |
| 18 | <apex:outputText id="NumberOfEmployees" value="{!account.NumberOfEmployees}"/> | |

|  |  |
| --- | --- |
| 19 |  |
| 20 | </apex:panelGrid> | |

|  |  |
| --- | --- |
| 21 | </apex:component> |

Replace your attachmentPDF page like this:

|  |  |  |
| --- | --- | --- |
| 1 | <apex:page standardController="account" renderAs="PDF"> | |
| 2 | <c:attachment/> |

|  |  |
| --- | --- |
| 3 | </apex:page> |

Then add the custom component to render at the bottom of your previous sendEmailPage:

|  |  |  |
| --- | --- | --- |
| 1 | <apex:pageBlock title="Preview the Attachment for {!account.name}"> | |
| 2 | <c:attachment/> |

|  |  |
| --- | --- |
| 3 | </apex:pageBlock> |

If you want to make changes to both the attachment and the preview, the attachment custom component needs to be modified in only one location.

## Example: Sending an Email with an Attachment

The following example shows the [previous sendEmail example](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_email_intro.htm) with a custom component that adds a Visualforce page as an attachment. First, the controller:

|  |  |
| --- | --- |
| 01 | public class sendEmail { |
| 02 | public String subject { get; set; } | |

|  |  |  |
| --- | --- | --- |
| 03 | public String body { get; set; } | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | private final Account account; | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | // Create a constructor that populates the Account object | |
| 08 | public sendEmail() { |

|  |  |
| --- | --- |
| 09 | account = [SELECT Name, |
| 10 | (SELECT Contact.Name, Contact.Email FROM Account.Contacts) | |

|  |  |
| --- | --- |
| 11 | FROM Account |
| 12 | WHERE Id = :ApexPages.currentPage().getParameters().get('id')]; | |

|  |  |  |
| --- | --- | --- |
| 13 | } | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | public Account getAccount() { | |
| 16 | return account; |

|  |  |  |
| --- | --- | --- |
| 17 | } | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | public PageReference send() { | |
| 20 | // Define the email |

|  |  |  |
| --- | --- | --- |
| 21 | Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage(); | |
| 22 |  |

|  |  |  |
| --- | --- | --- |
| 23 | // Reference the attachment page and pass in the account ID | |
| 24 | PageReference pdf =  Page.attachmentPDF; |

|  |  |  |
| --- | --- | --- |
| 25 | pdf.getParameters().put('id',(String)account.id); | |
| 26 | pdf.setRedirect(true); |

|  |  |
| --- | --- |
| 27 |  |
| 28 | // Take the PDF content | |

|  |  |  |
| --- | --- | --- |
| 29 | Blob b = pdf.getContent(); | |
| 30 |  |

|  |  |
| --- | --- |
| 31 | // Create the email attachment |
| 32 | Messaging.EmailFileAttachment efa = new Messaging.EmailFileAttachment(); | |

|  |  |  |
| --- | --- | --- |
| 33 | efa.setFileName('attachment.pdf'); | |
| 34 | efa.setBody(b); |

|  |  |
| --- | --- |
| 35 |  |
| 36 | String addresses; | |

|  |  |
| --- | --- |
| 37 | if (account.Contacts[0].Email != null) { |
| 38 | addresses = account.Contacts[0].Email; | |

|  |  |  |
| --- | --- | --- |
| 39 | // Loop through the whole list of contacts and their emails | |
| 40 | for (Integer i = 1; i < account.Contacts.size(); i++) { |

|  |  |
| --- | --- |
| 41 | if (account.Contacts[i].Email != null) { |
| 42 | addresses += ':' + account.Contacts[i].Email; | |

|  |  |  |
| --- | --- | --- |
| 43 | } | |
| 44 | } |

|  |  |  |
| --- | --- | --- |
| 45 | } | |
| 46 |  |

|  |  |  |
| --- | --- | --- |
| 47 | String[] toAddresses = addresses.split(':', 0); | |
| 48 |  |

|  |  |  |
| --- | --- | --- |
| 49 | // Sets the paramaters of the email | |
| 50 | email.setSubject( subject ); |

|  |  |  |
| --- | --- | --- |
| 51 | email.setToAddresses( toAddresses ); | |
| 52 | email.setPlainTextBody( body ); |

|  |  |
| --- | --- |
| 53 |  |
| 54 | email.setFileAttachments(new Messaging.EmailFileAttachment[] {efa}); | |

|  |  |
| --- | --- |
| 55 |  |
| 56 | // Sends the email | |

|  |  |
| --- | --- |
| 57 | Messaging.SendEmailResult [] r = |
| 58 | Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email}); | |

|  |  |
| --- | --- |
| 59 |  |
| 60 | return null; | |

|  |  |  |
| --- | --- | --- |
| 61 | } | |
| 62 | } |

Next, the Visualforce page that sends the email:

[view source](https://developer.salesforce.com/docs/#viewSource)

[print](https://developer.salesforce.com/docs/#printSource)[?](https://developer.salesforce.com/docs/#about)

|  |  |  |
| --- | --- | --- |
| 01 | <apex:page controller="sendEmail"> | |
| 02 | <apex:messages/> |

|  |  |  |
| --- | --- | --- |
| 03 | <apex:pageBlock title="Send an Email to Your {!account.name} Representatives"> | |
| 04 | <p>Fill out the fields below to test how you might send an email to a user.</p> |

|  |  |
| --- | --- |
| 05 |  |
| 06 | <apex:dataTable value="{!account.Contacts}" var="contact" border="1"> | |

|  |  |
| --- | --- |
| 07 | <apex:column> |
| 08 | <apex:facet name="header">Name</apex:facet> | |

|  |  |  |
| --- | --- | --- |
| 09 | {!contact.Name} | |
| 10 | </apex:column> |

|  |  |
| --- | --- |
| 11 | <apex:column> |
| 12 | <apex:facet name="header">Email</apex:facet> | |

|  |  |  |
| --- | --- | --- |
| 13 | {!contact.Email} | |
| 14 | </apex:column> |

|  |  |  |
| --- | --- | --- |
| 15 | </apex:dataTable> | |
| 16 |  |

|  |  |
| --- | --- |
| 17 | <apex:form><br/><br/> |
| 18 | <apex:outputLabel value="Subject" for="Subject"/>: <br/> | |

|  |  |  |
| --- | --- | --- |
| 19 | <apex:inputText value="{!subject}" id="Subject" maxlength="80"/> | |
| 20 | <br/><br/> |

|  |  |
| --- | --- |
| 21 |  |
| 22 | <apex:outputLabel value="Body" for="Body"/>: <br/> | |

|  |  |  |
| --- | --- | --- |
| 23 | <apex:inputTextarea value="{!body}" id="Body" rows="10" cols="80"/> | |
| 24 | <br/><br/> |

|  |  |
| --- | --- |
| 25 |  |
| 26 | <apex:commandButton value="Send Email" action="{!send}"/> | |

|  |  |
| --- | --- |
| 27 | </apex:form> |
| 28 | </apex:pageBlock> | |

|  |  |
| --- | --- |
| 29 |  |
| 30 | <apex:pageBlock title="Preview the Attachment for {!account.name}"> | |

|  |  |
| --- | --- |
| 31 | <c:attachment/> |
| 32 | </apex:pageBlock> | |

|  |  |
| --- | --- |
| 33 | </apex:page> |