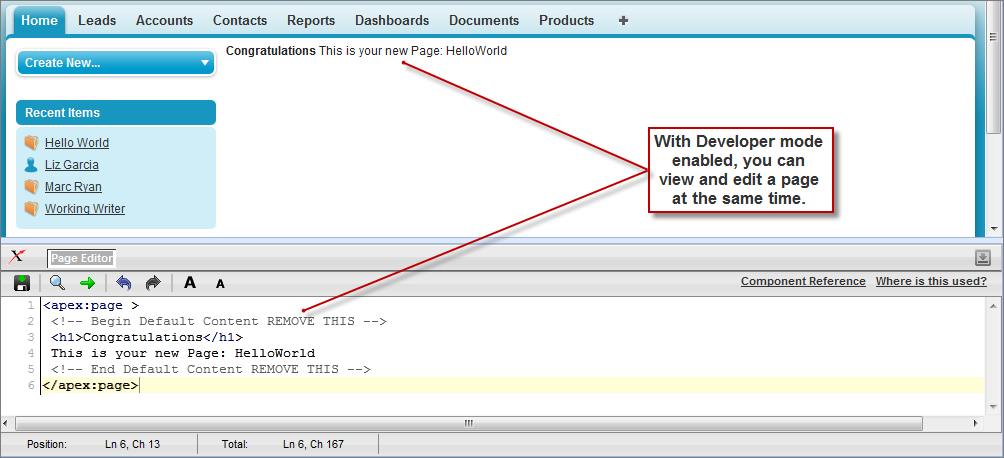
**Creating Your First Page**

**A New Visualforce Page**

You now have a Visualforce page that includes default text. To edit your new page, click the **Page Editor** bar that appears at the bottom of the browser. It expands to show you the following Visualforce markup:

|  |  |
| --- | --- |
| 1 | <apex:page> |
| 2 | <!-- Begin Default Content REMOVE THIS --> | |

|  |  |
| --- | --- |
| 3 | <h1>Congratulations</h1> |
| 4 | This is your new Apex Page: HelloWorld | |

|  |  |  |
| --- | --- | --- |
| 5 | <!-- End Default Content REMOVE THIS --> | |
| 6 | </apex:page> |

This default markup includes the only required tag for any page— the <apex:page> tag that begins and ends any page markup. Embedded within the start and close <apex:page> tags is plain text, some of which is formatted with a standard HTML tag, <h1>.

As long as you keep the required <apex:page> tag you can add as much plain text or valid HTML to this page as you want. For example, after entering the following code and clicking **Save** in the Page Editor, the page displays the text “Hello World!” in bold:

|  |  |
| --- | --- |
| 1 | <apex:page> |
| 2 | <b>Hello World!</b> | |

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

**Displaying Field Values with Visualforce**

Visualforce pages use the same expression language as formulas—that is, anything inside {! } is evaluated as an expression that can access values from records that are currently in context. For example, you can display the current user's first name by adding the {!$User.FirstName} expression to a page:

|  |  |
| --- | --- |
| 1 | <apex:page> |
| 2 | Hello {!$User.FirstName}! | |

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

$User is a global variable that always represents the current user record. All global variables are referenced with a $ symbol. For a list of global variables that you can use in Visualforce, see [Global Variables](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_variables_global.htm).

To access fields from a record that is not globally available, like a specific account, contact, or custom object record, you need to associate your page with a *controller*. Controllers provide pages with the data and business logic that make your application run, including the logic that specifies how to access a particular object's records. While you can define a custom controller for any page with Apex, Salesforce includes standard controllers for every standard and custom object.

For example, to use the standard controller for accounts, add the standardController attribute to the <apex:page> tag, and assign it the name of the account object:

|  |  |  |
| --- | --- | --- |
| 1 | <apex:page standardController="Account"> | |
| 2 | Hello {!$User.FirstName}! |

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

**Using the Visualforce Component Library**

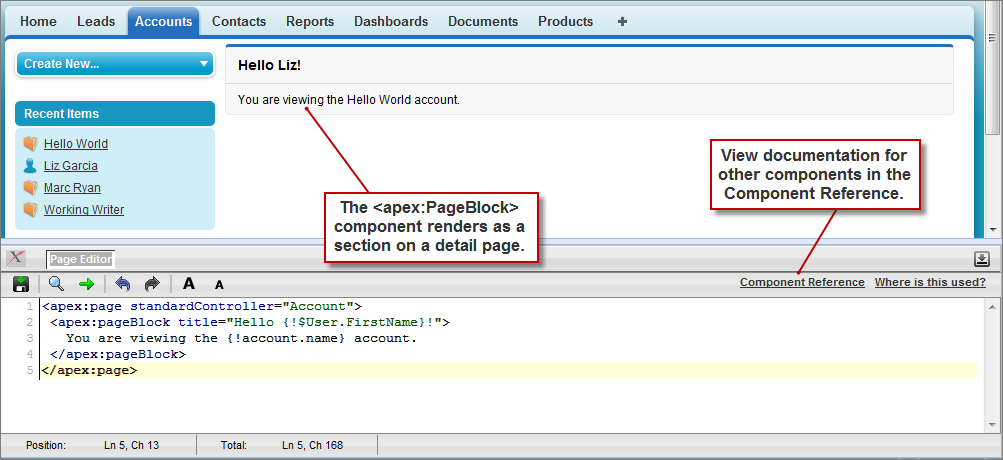
Up to this point, the only Visualforce tag that has been used in the examples is the mandatory <apex:page> tag that must be placed at the start and end of all Visualforce markup. However, just as you can insert images or tables into an HTML document with the <img> or <table> tags, respectively, you can add user interface components to your Visualforce pages using tags that are defined in the Visualforce component library.

For example, to add a component that looks like a section on a detail page, use the <apex:pageBlock> component tag:

|  |  |
| --- | --- |
| 1 | <apex:page standardController="Account"> |
| 2 | <apex:pageBlock title="Hello {!$User.FirstName}!"> | |

|  |  |  |
| --- | --- | --- |
| 3 | You are viewing the {!account.name} account. | |
| 4 | </apex:pageBlock> |

|  |  |
| --- | --- |
| 5 | </apex:page> |

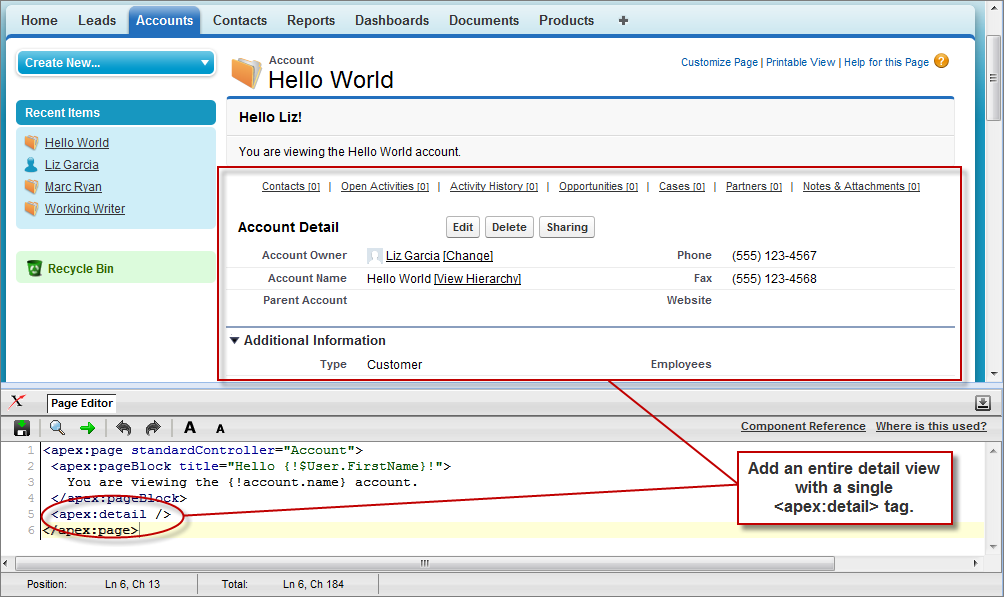
**The <apex:pageBlock> Component**

Tags also exist for other common Salesforce interface components, such as related lists, detail pages, and input fields. For example, to add the content of a detail page, use the <apex:detail> component tag:

|  |  |
| --- | --- |
| 1 | <apex:page standardController="Account"> |
| 2 | <apex:pageBlock title="Hello {!$User.FirstName}!"> | |

|  |  |  |
| --- | --- | --- |
| 3 | You are viewing the {!account.name} account. | |
| 4 | </apex:pageBlock> |

|  |  |  |
| --- | --- | --- |
| 5 | <apex:detail/> | |
| 6 | </apex:page> |

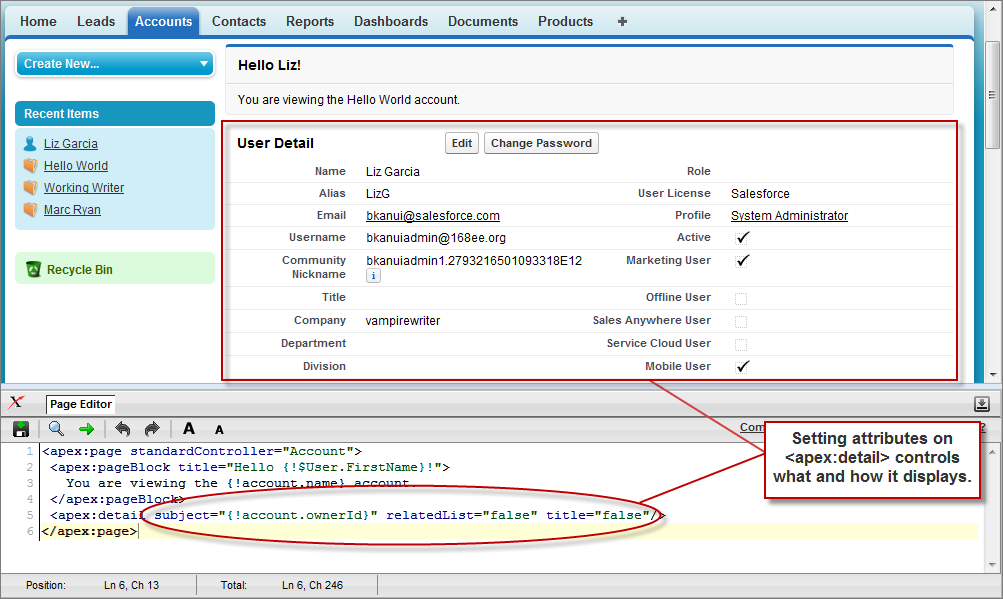
**The <apex:detail> Component Without Attributes**

Without any specified attributes on the tag, <apex:detail> displays the complete detail view for the context record. If you want to modify properties such as which record details are displayed, or whether related lists or the title appear, you can use attributes on the tag. For example, the following markup displays the details of the context account's owner, without related lists or a colored title bar:

|  |  |
| --- | --- |
| 1 | <apex:page standardController="Account"> |
| 2 | <apex:pageBlock title="Hello {!$User.FirstName}!"> | |

|  |  |  |
| --- | --- | --- |
| 3 | You are viewing the {!account.name} account. | |
| 4 | </apex:pageBlock> |

|  |  |  |
| --- | --- | --- |
| 5 | <apex:detail subject="{!account.ownerId}" relatedList="false" title="false"/> | |
| 6 | </apex:page> |

**The <apex:detail> Component Without Related List or Title Elements**

If a component is updated or edited, the Visualforce page that references it is also updated.