Software Cloud Computing

isobar

Index

Week #5 – Lesson #5

Review last lesson

isobar

Apex

- Object-oriented programming;
- Add business logic to system events;
- Build complex business processes;
- Customized user interfaces;
- Customize the prebuilt applications;
- and integrations with third-party systems.

Apex

- Anonymous Blocks;
- Triggers;
- Asynchronous Apex: `Queueable`, `Scheduled`, `Batch`, `Future;`
- Methods as SOAP Web Services;
- Classes as REST Web Services;
- Apex Email Service;
- Visualforce Classes;
- JavaScript Remoting;
- Apex in AJAX.

isobar

Apex

- Version
- Data Types
- Variables
- Collections
- Conditionals
- Loops
- Classes

Visualforce Pages

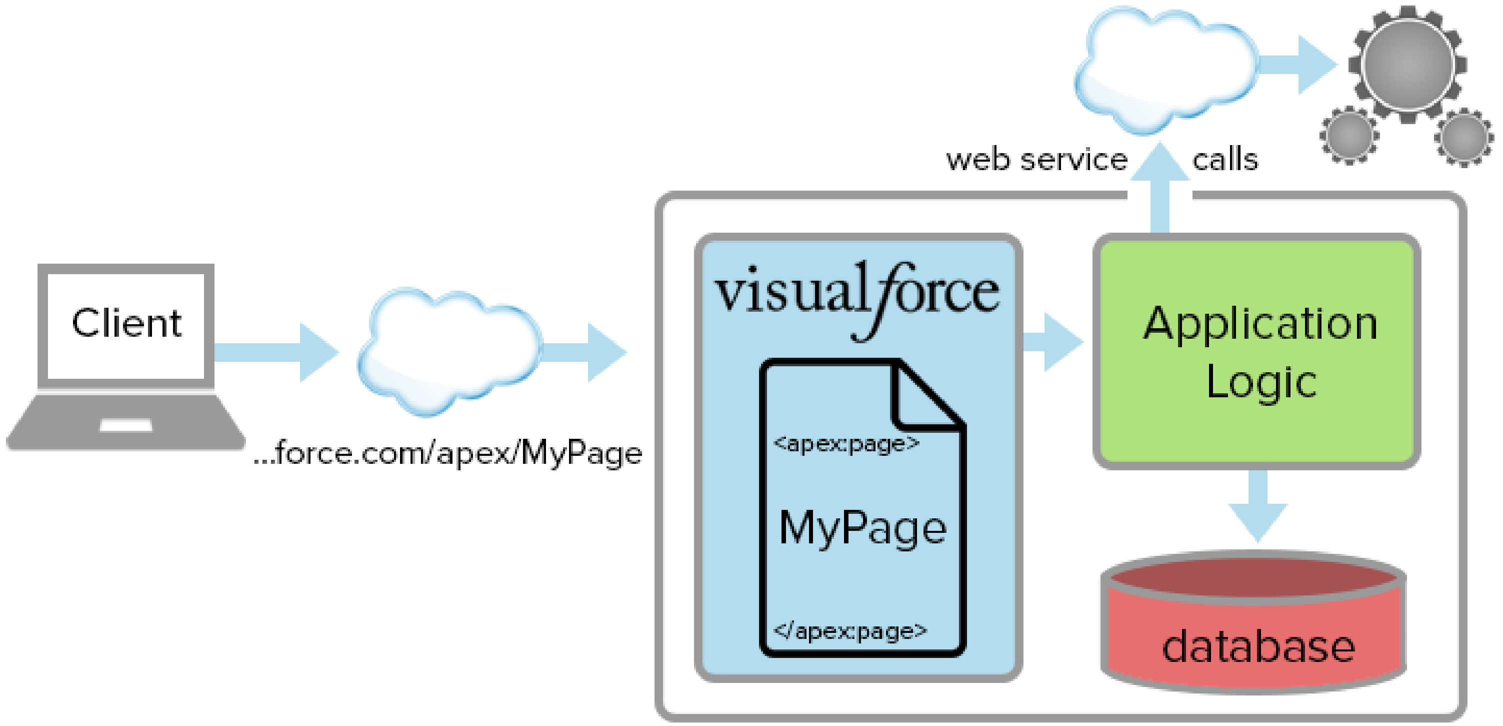
Visualforce Page

- is a web development framework that enables developers to build sophisticated, custom user interfaces for mobile and desktop;
- to build apps that align with the styling of Lightning Experience, as well as your own completely custom interface;
- to extend Salesforce’s built-in features, replace them with new functionality, and build completely new apps;
- built-in standard controller features,or write your own custom business logic in Apex;

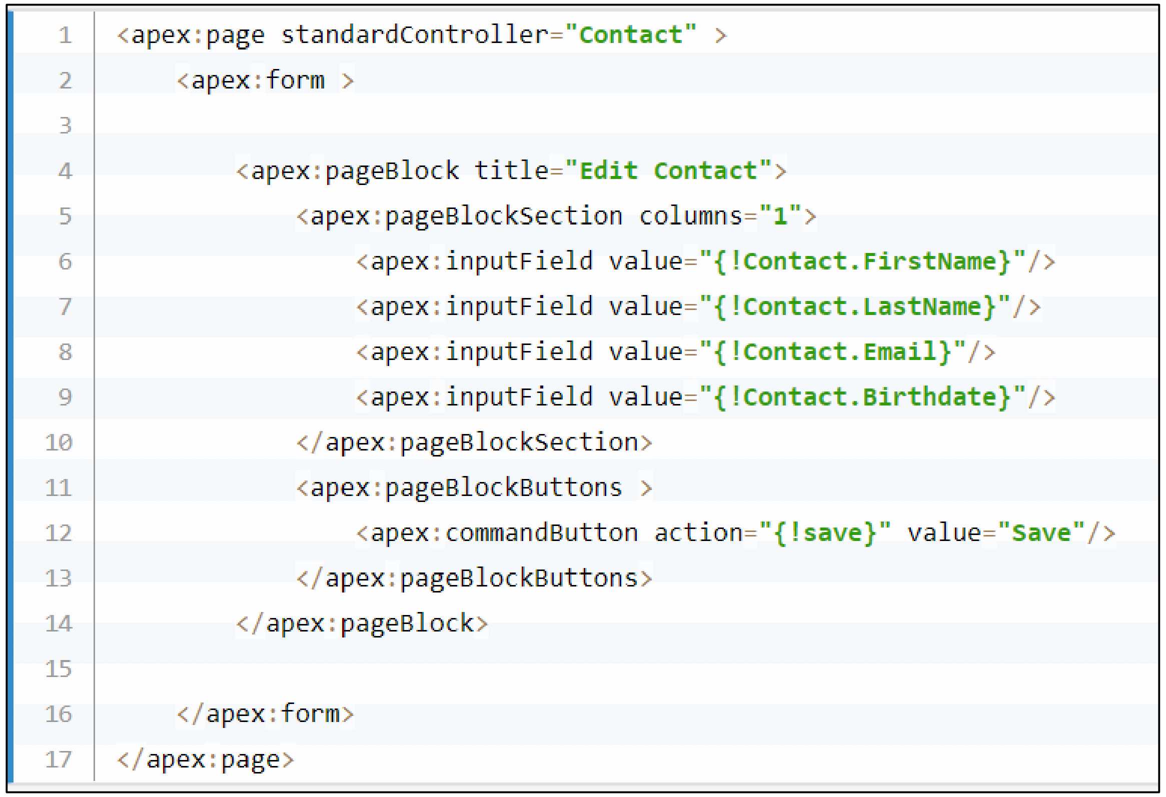
Visualforce Page

- create Visualforce pages by composing components, HTML, and optional styling elements;
- can integrate with any standard web technology or JavaScript framework to allow for a more animated and rich user interface;
- each page is accessible by a unique URL;
- the server performs any data processing required by the page, renders the page into HTML, and returns the results to the browser for display.

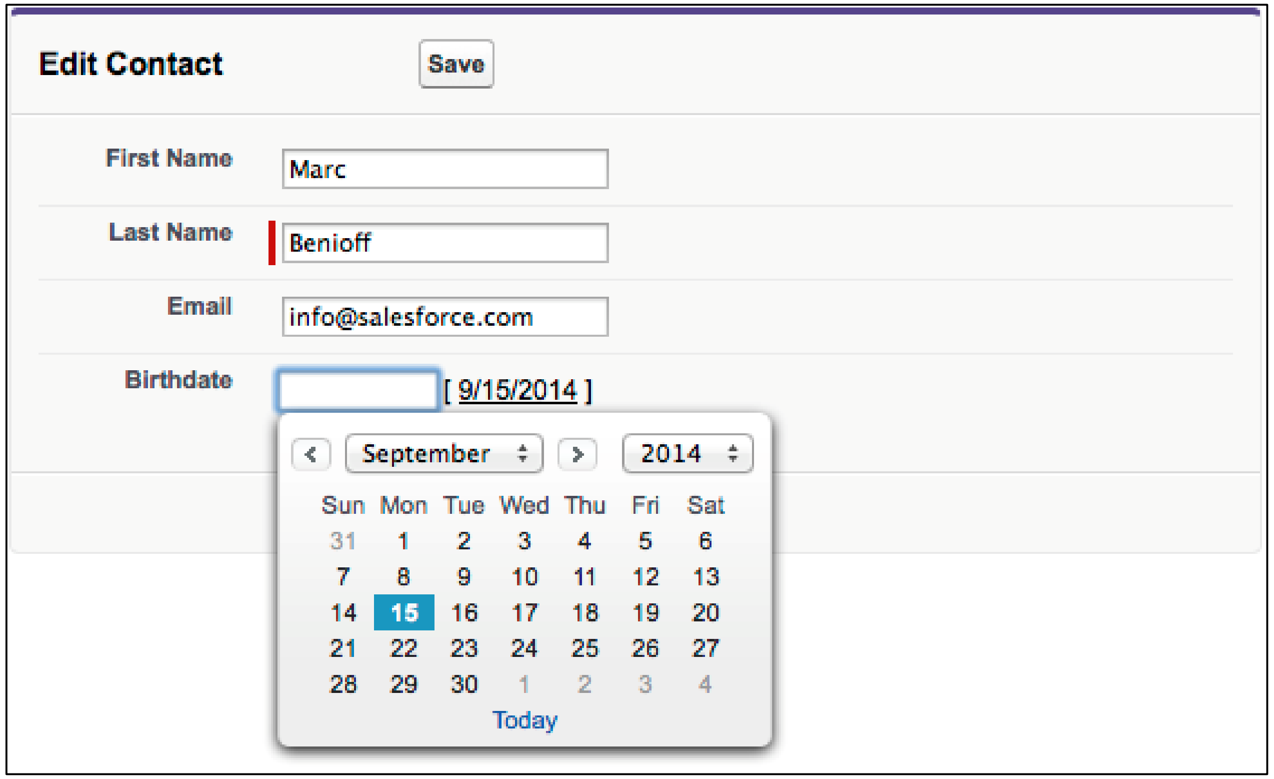
Visualforce Page



Visualforce Page



```
<apex:page standardController="Contact" >
    <apex:form >

        <apex:pageBlock title="Edit Contact">
            <apex:pageBlockSection columns="1">
                <apex:inputField value="{!Contact.FirstName}"/>
                <apex:inputField value="{!Contact.LastName}"/>
                <apex:inputField value="{!Contact.Email}"/>
                <apex:inputField value="{!Contact.Birthdate}"/>
            </apex:pageBlockSection>
            <apex:pageBlockButtons >
                <apex:commandButton action="{!save}" value="Save"/>
            </apex:pageBlockButtons>
        </apex:pageBlock>

    </apex:form>
</apex:page>
```



Edit Contact

Save

First Name: Marc

Last Name: Benioff

Email: info@salesforce.com

Birthdate: [9/15/2014]

isobar

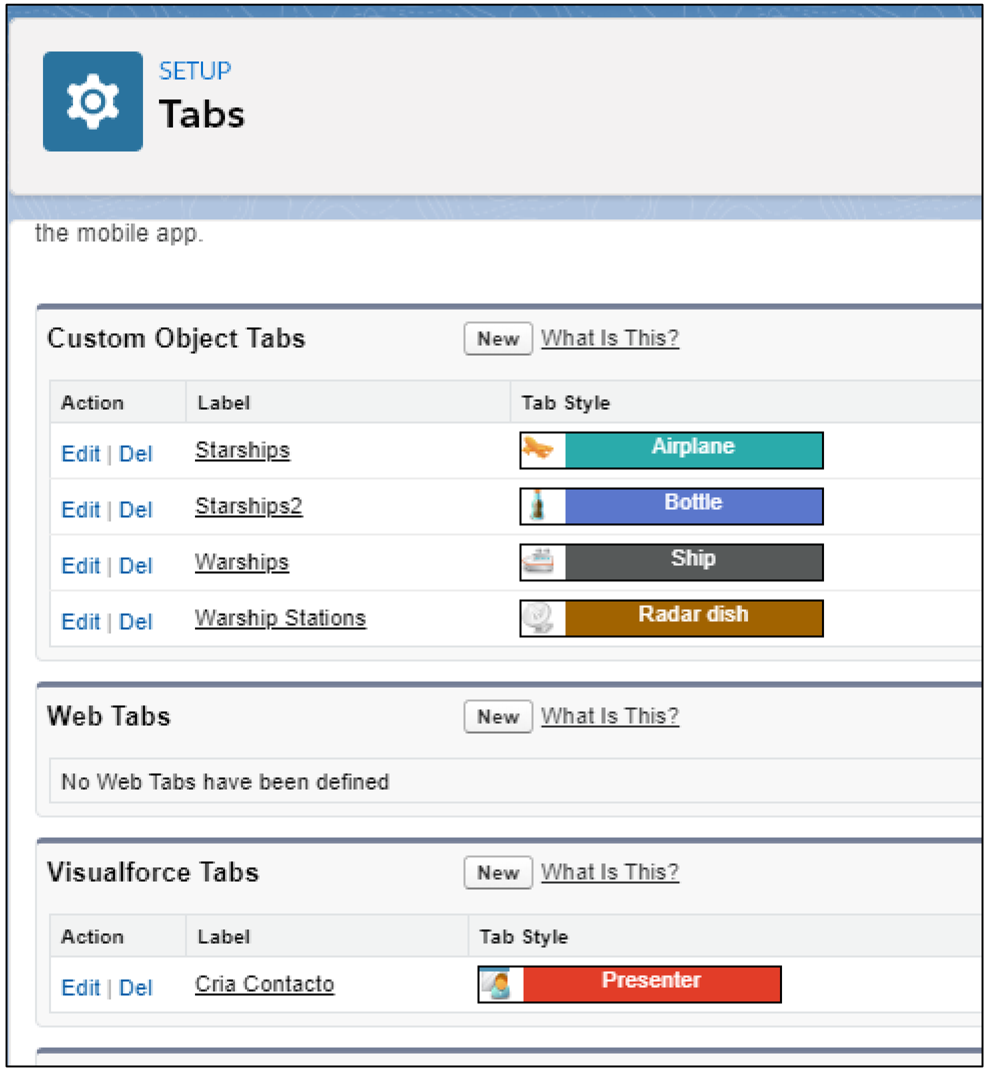
Visualforce Page

- It connects to the Visualforce standard controller, a part of the Visualforce framework that provides automatic data access and modification, standard actions, and more.
- When accessed **without** a record ID, the page displays a blank data entry form. When you click **Save**, a **new record** is created from the form data.
- When accessed **with** a record ID, the page **looks up** the data for that contact record and displays it in an editable form. When you click **Save**, your changes for the contact are **saved back** to the database.
- Each input field is smart about how it presents its value.
    - The email field knows what a valid email address looks like, and displays an error if an invalid email is entered.
    - The date field displays a date widget when you click into the field to make entering a date easier.
- The **Save** button calls the save action method;

Visualforce Page

Where You Can Use Visualforce (VF)

- **Open a Visualforce Page from the App Launcher** – need to create a new "Visualforce Tab" (in Setup → Tabs);



SETUP
Tabs

the mobile app.

Custom Object Tabs New What Is This?

Action	Label	Tab Style
Edit | Del	Starships	Airplane
Edit | Del	Starships2	Bottle
Edit | Del	Warships	Ship
Edit | Del	Warship Stations	Radar dish

Web Tabs New What Is This?

No Web Tabs have been defined

Visualforce Tabs New What Is This?

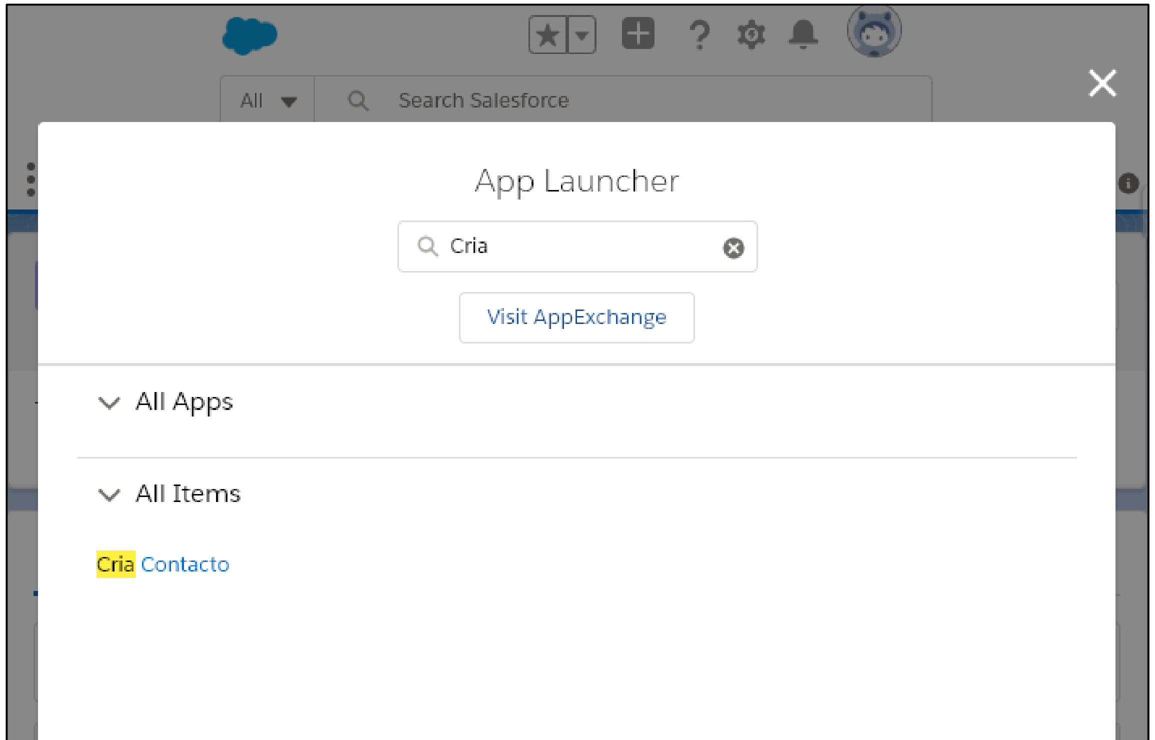
Action	Label	Tab Style
Edit | Del	Cria Contacto	Presenter

isobar

Visualforce Page

Where You Can Use Visualforce (VF)

- **Open a Visualforce Page from the App Launcher**

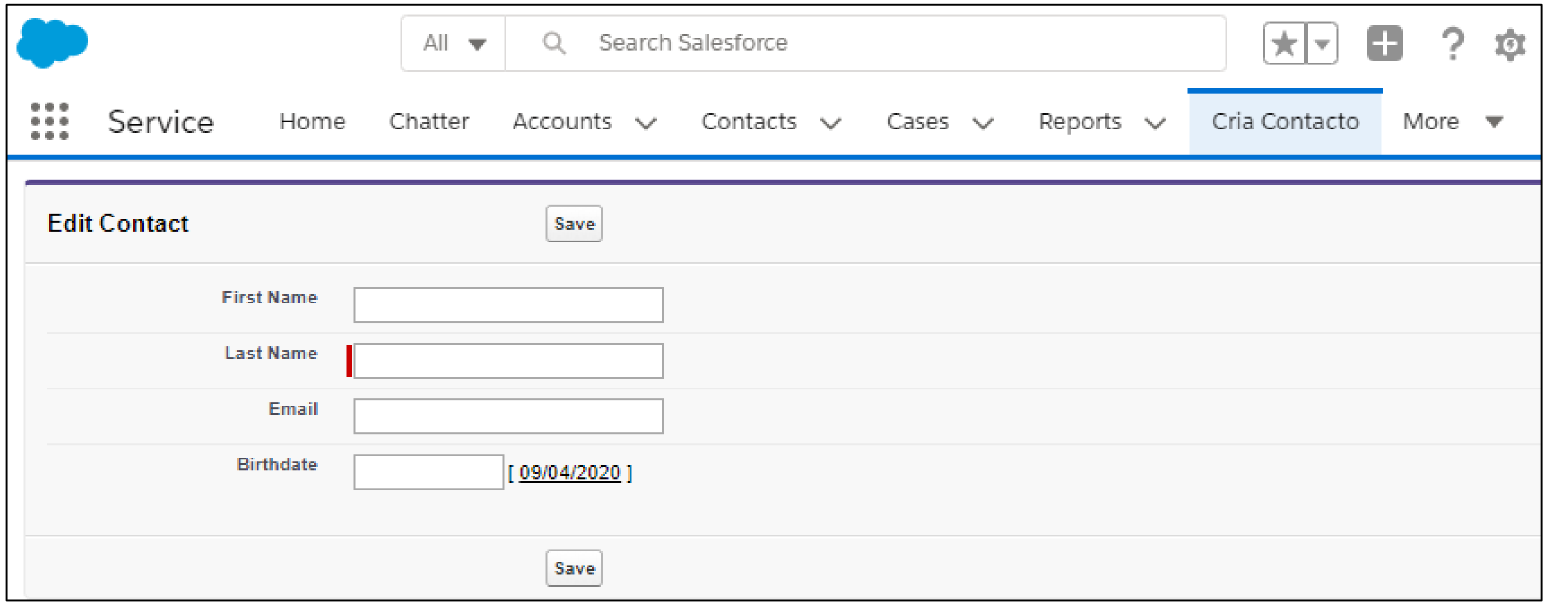


isobar

Visualforce Page

Where You Can Use Visualforce (VF)

- **Add a Visualforce Page to the Navigation Bar**: you can add Visualforce tabs to an app;



Edit Contact

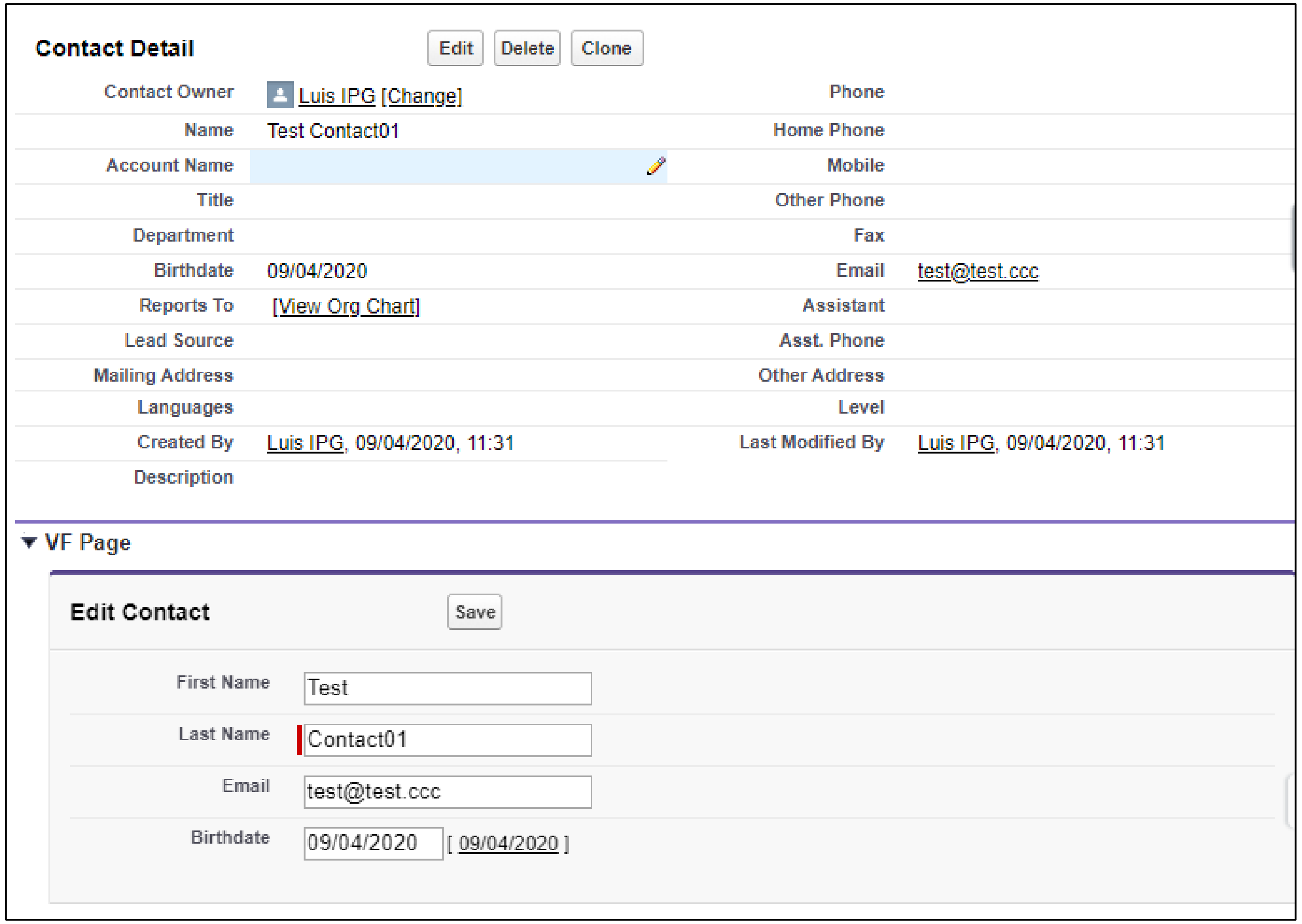
First Name	
Last Name	
Email	
Birthdate	[09/04/2020]

isbar

Visualforce Page

Where You Can Use Visualforce (VF)

- **Display a VF Page within a Standard Page Layout**: Extend your page layouts by embedding Visualforce pages;



Contact Detail

Contact Owner	Luis IPG [Change]	Phone	
Name	Test Contact01	Home Phone	
Account Name		Mobile	
Title		Other Phone	
Department		Fax	
Birthdate	09/04/2020	Email	test@test.ccc
Reports To	[View Org Chart]	Assistant	
Lead Source		Asst. Phone	
Mailing Address		Other Address	
Languages		Level	
Created By	Luis IPG, 09/04/2020, 11:31	Last Modified By	Luis IPG, 09/04/2020, 11:31
Description			

VF Page

Edit Contact

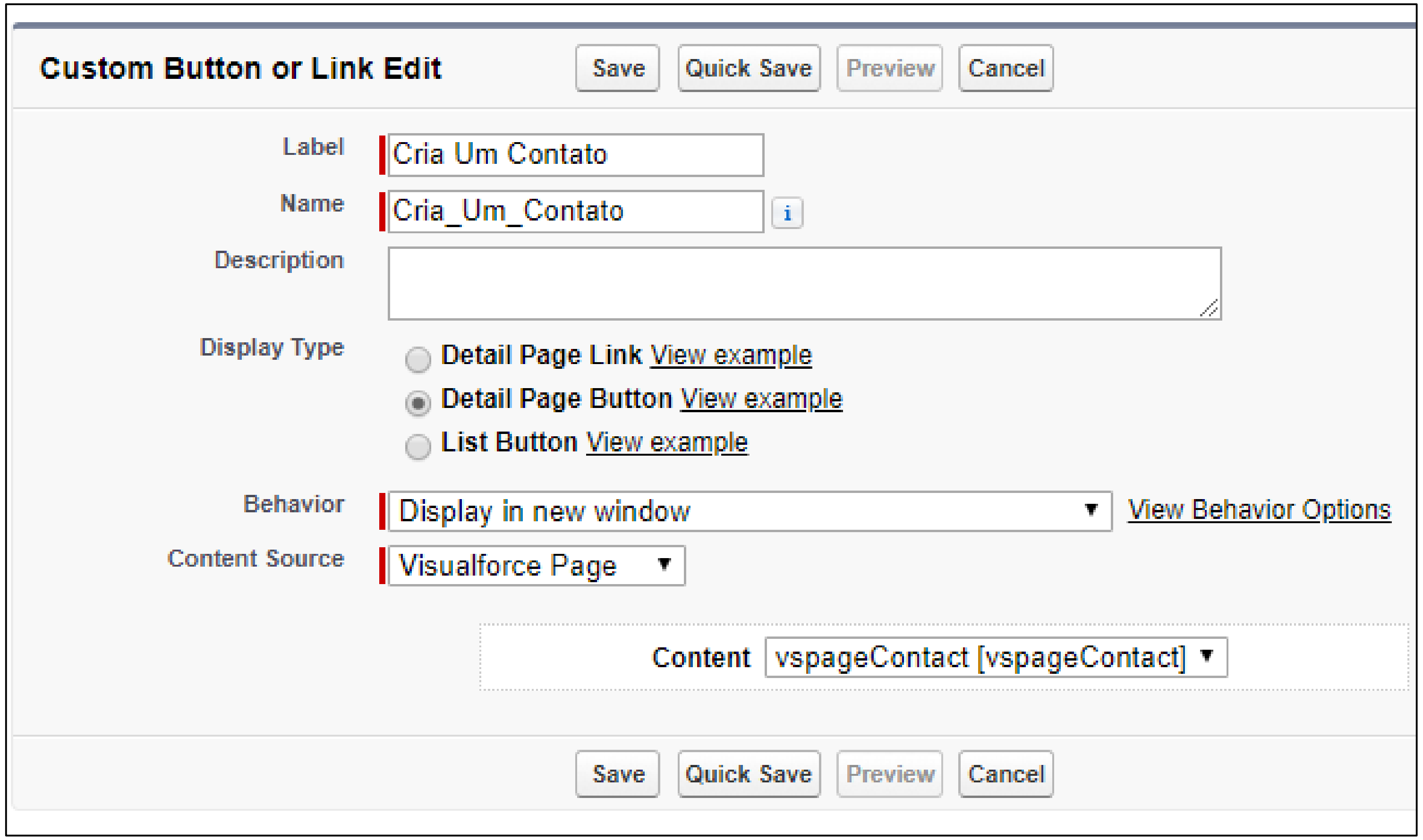
First Name	Test
Last Name	Contact01
Email	test@test.ccc
Birthdate	09/04/2020 [09/04/2020]

isobar

Visualforce Page

Where You Can Use Visualforce (VF)

- **Launch a Visualforce Page as a Quick Action**: Create a new Quick Action and add to the Page Layout;

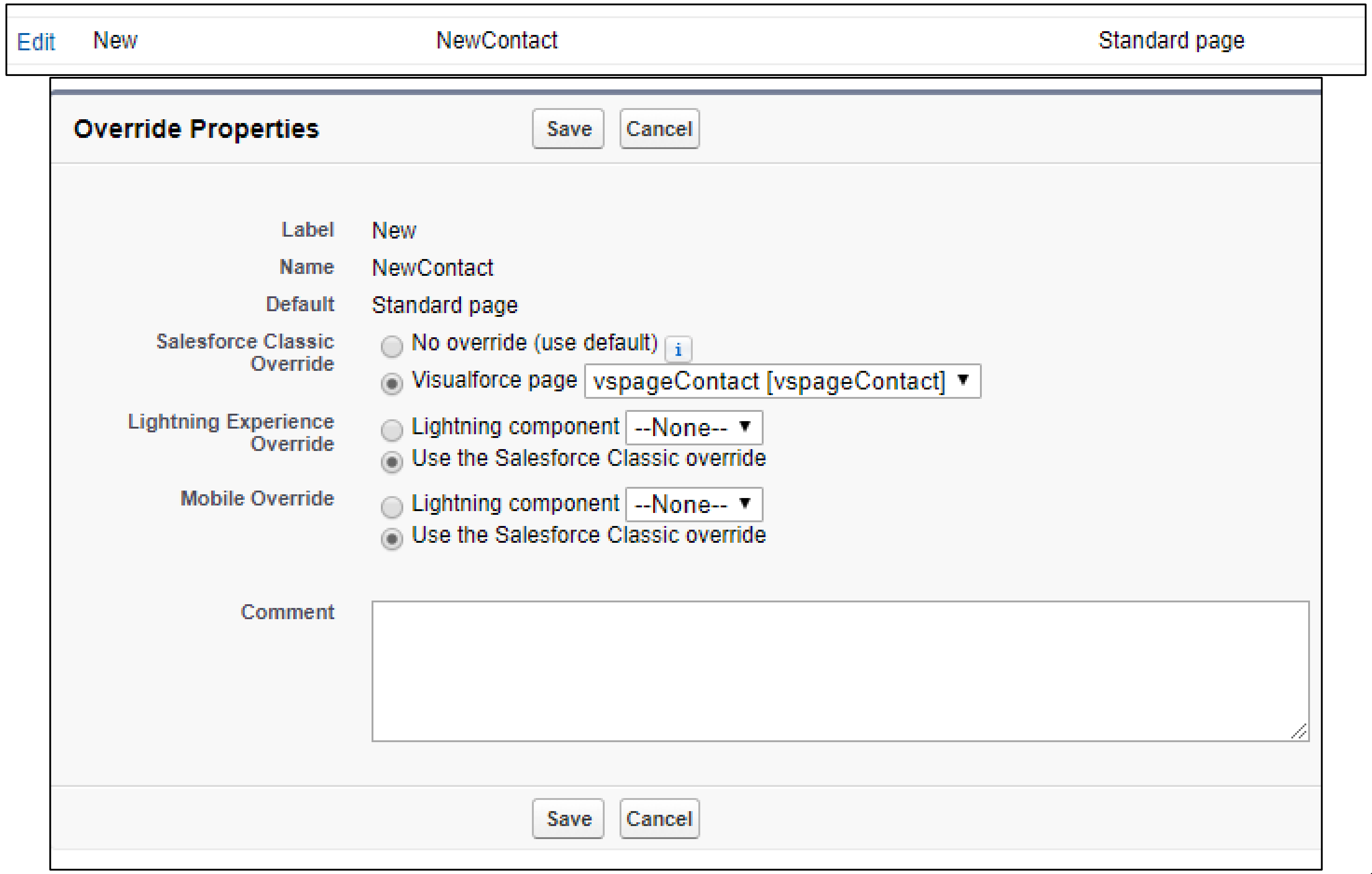


Label	Cria Um Contato
Name	Cria_Um_Contato
Description	
Display Type	( ) Detail Page Link View example (● Detail Page Button) View example ( ) List Button View example
Behavior	Display in new window
Content Source	Visualforce Page
Content	vspageContact [vspageContact]

Visualforce Page

Where You Can Use Visualforce (VF)

- **Overriding Standard Buttons or Links**: You can override the actions available on an object with a Visualforce page;



Edit	New	NewContact	Standard page

Override Properties Save Cancel

Label	New
Name	NewContact
Default	Standard page
Salesforce Classic Override	No override (use default) [✓ Visualforce page] vspageContact [vspageContact]
Lightning Experience Override	Lightning component --None-- [✓ Use the Salesforce Classic override]
Mobile Override	Lightning component --None-- [✓ Use the Salesforce Classic override]
Comment	

Save Cancel

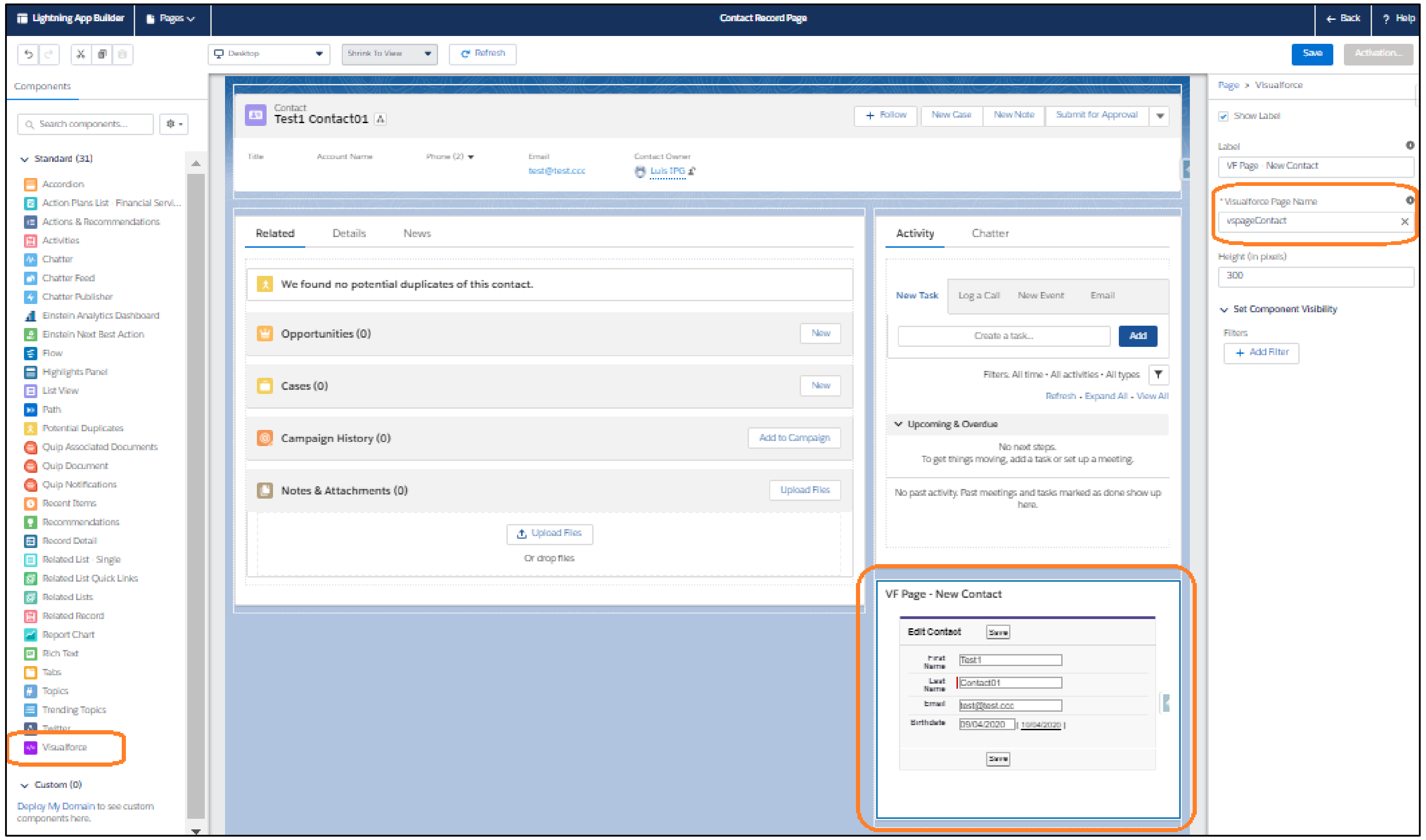
isobar

Visualforce Page

Where You Can Use

Visualforce (VF)

- **Component in the Lightning App Builder**: in the Lightning App Builder, you can add a Visualforce page to the page by using the Visualforce component;



Show Label ✓

Label: VF Page - New Contact

* Visualforce Page Name: vspageContact

Height (in pixels): 300

Set Component Visibility

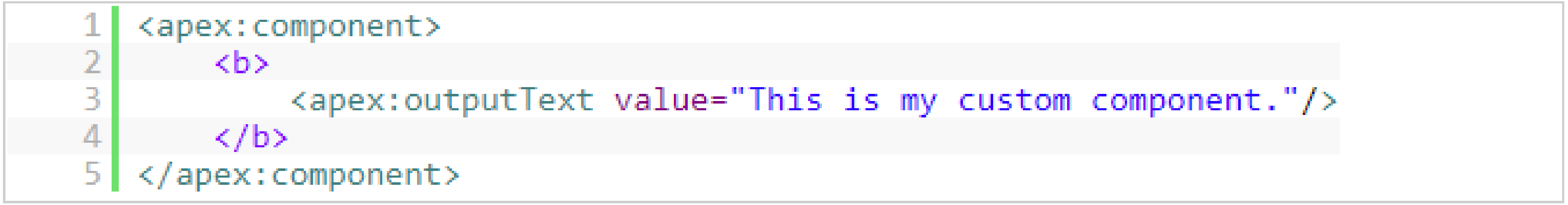
Filters

Add Filter

isbar

Visualforce component

- Reuse a component several times in one or more Visualforce pages (like you can reuse a piece of code in a method or in a program);
- Allow developers to define attributes that can be passed in to each component;
- The value of an attribute can then change the way the markup is displayed on the final page;
- Is defined within an **`<apex:component>`** tag. This tag must be the top-level tag in a custom component definition



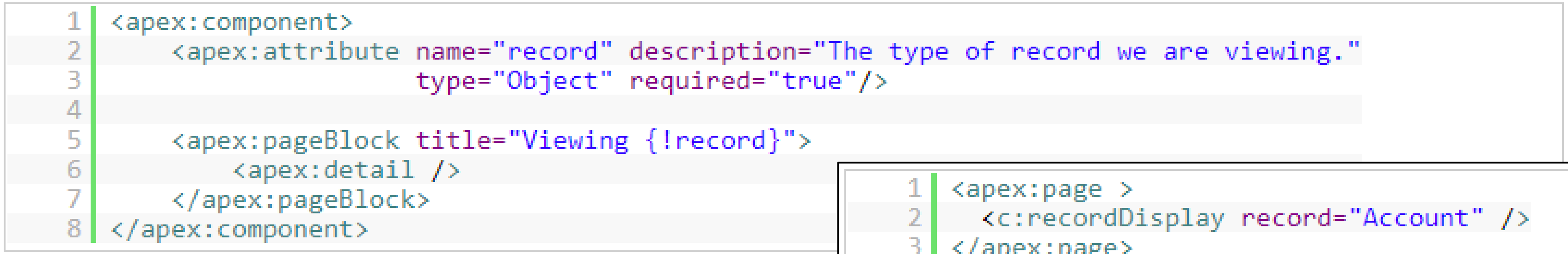
```
<apex:component>
    <b>
        <apex:outputText value="This is my custom component."/>
    </b>
</apex:component>
```

- In the Visualforce page you can insert the component like this:

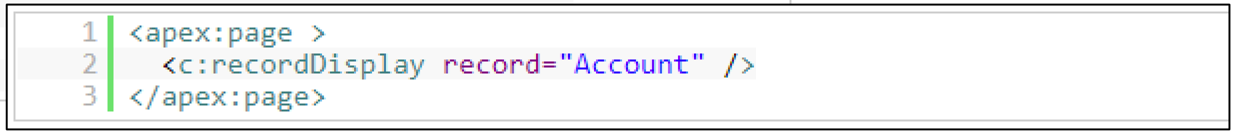
```
<c:Name_VF_Component />
```

Visualforce component

- The body of an `<apex:component>` tag can also specify the attributes that can be passed in to the custom component when it's used in a Visualforce page;
- An `<apex:attribute>` tag requires values for the **name**, **description**, and **type** attributes;
    - The **name** attribute defines how the custom attribute can be referenced in Visualforce pages (must be unique);
    - The **description** attribute defines the help text for the attribute that appears in the component;
    - The **type** attribute defines the Apex data type of the attribute;



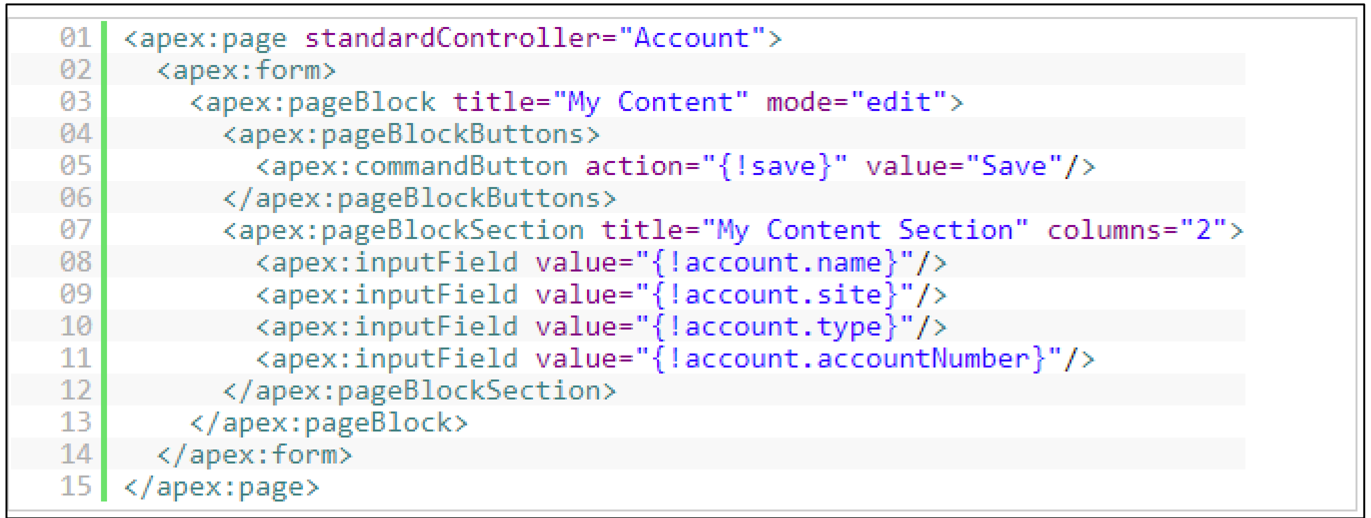
```
<apex:component>
    <apex:attribute name="record" description="The type of record we are viewing."
                    type="Object" required="true"/>

    <apex:pageBlock title="Viewing {!record}">
        <apex:detail />
    </apex:pageBlock>
</apex:component>
```



```
<apex:page >
  <c:recordDisplay record="Account" />
</apex:page>
```

Visualforce: Controllers

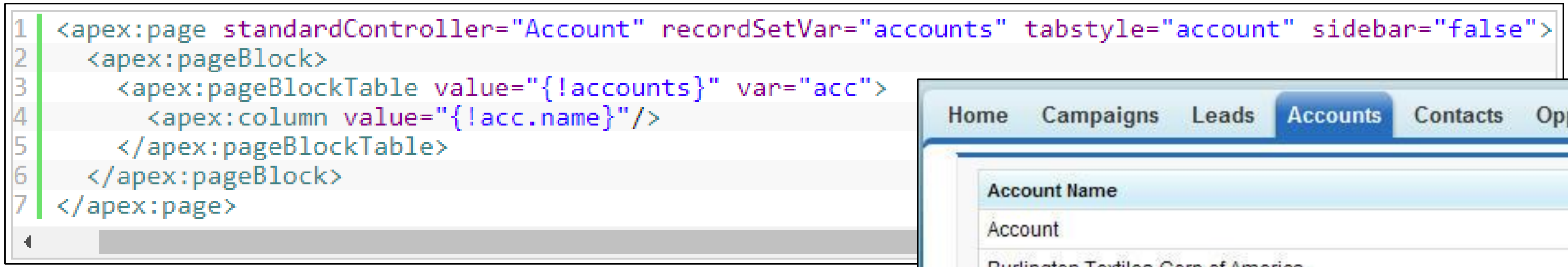
- **Standard Controllers**
  - Is a set of instructions that specify what happens when a user interacts with the components specified in associated Visualforce markup;
  - Controllers also provide access to the data that should be displayed in a page, and can modify component behavior;
  - Exists for every Salesforce object that can be queried using the Lightning Platform API;
  - Example: Save, Update, Delete, etc;



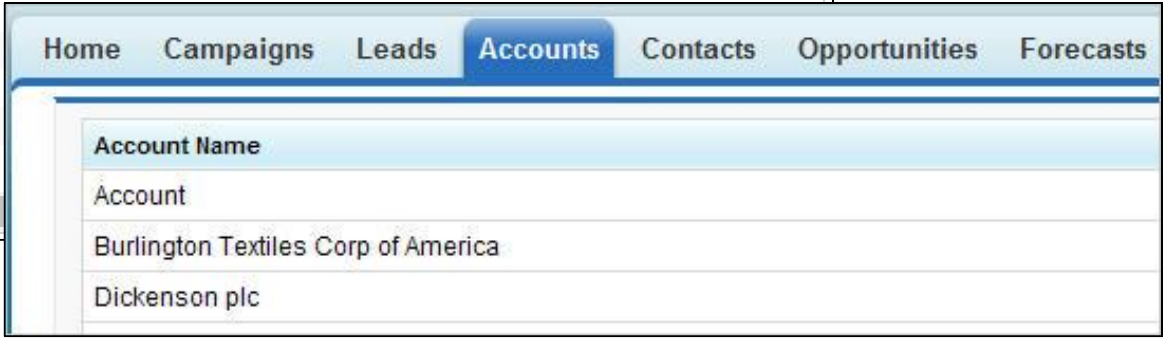
```
<apex:page standardController="Account">
  <apex:form>
    <apex:pageBlock title="My Content" mode="edit">
      <apex:pageBlockButtons>
        <apex:commandButton action="{!save}" value="Save"/>
      </apex:pageBlockButtons>
      <apex:pageBlockSection title="My Content Section" columns="2">
        <apex:inputField value="{!account.name}"/>
        <apex:inputField value="{!account.site}"/>
        <apex:inputField value="{!account.type}"/>
        <apex:inputField value="{!account.accountNumber}"/>
      </apex:pageBlockSection>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

Visualforce: Controllers

- **Standard List Controllers**
  - Allow you to create Visualforce pages that can display or act on a set of records;
  - work with a set of records include list pages, related lists, and mass action pages;
  - Standard list controllers can be used with the following objects:
    - Account; Asset; Campaign; Case; Contact; Contract; Idea; Lead; Opportunity; Order; Product2; Solution; User; Custom objects.



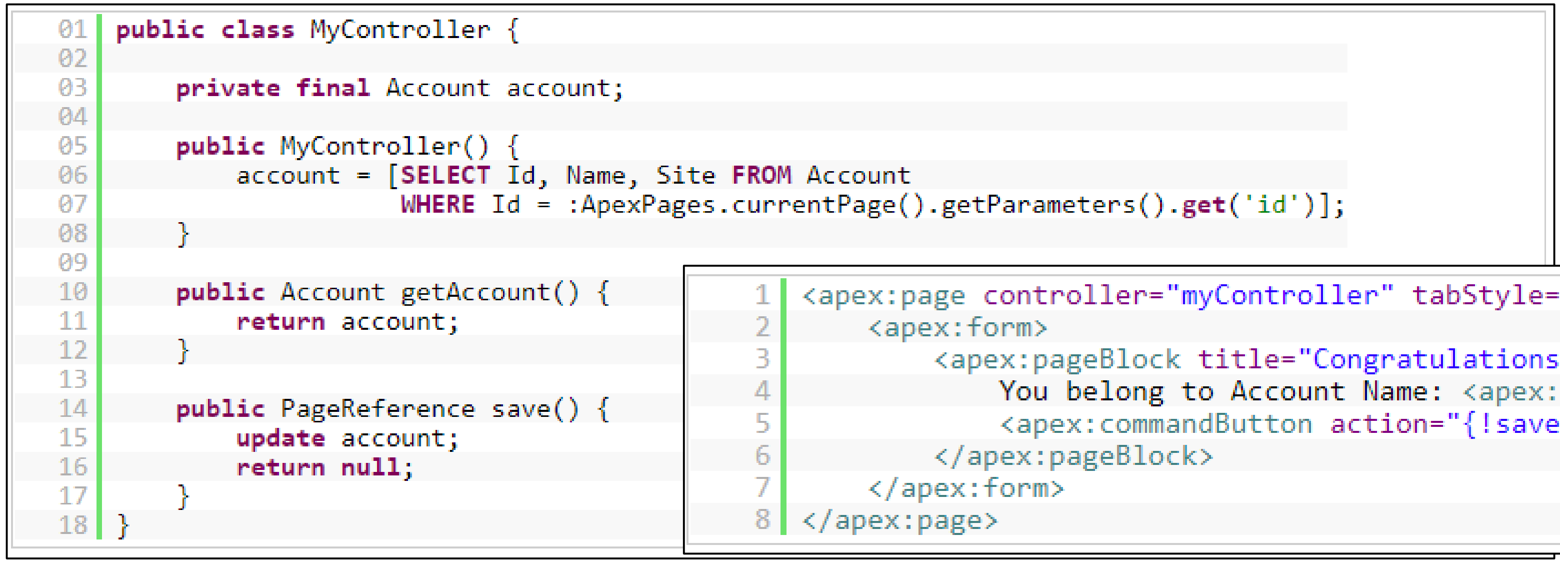
```
<apex:page standardController="Account" recordSetVar="accounts" tabstyle="account" sidebar="false">
  <apex:pageBlock>
    <apex:pageBlockTable value="{!accounts}" var="acc">
      <apex:column value="{!acc.name}"/>
    </apex:pageBlockTable>
  </apex:pageBlock>
</apex:page>
```



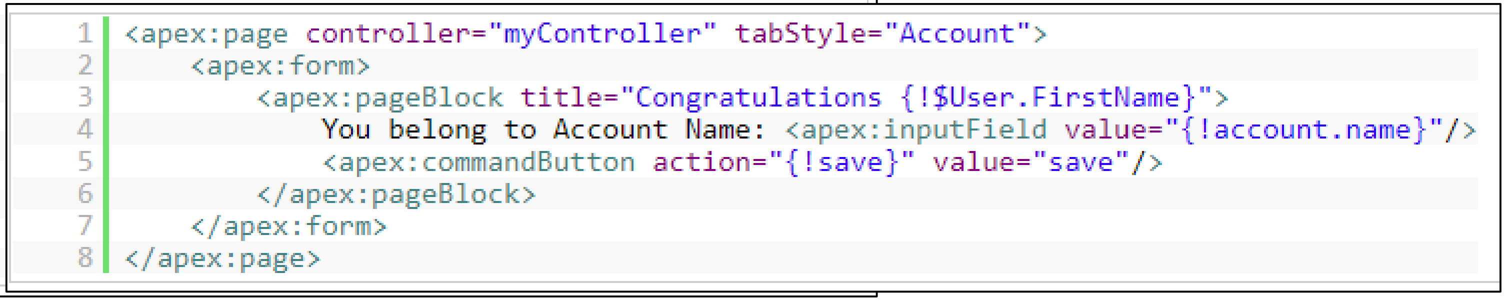
Account Name
Account
Burlington Textiles Corp of America
Dickenson plc

Visualforce: Controllers

- **Custom Controllers**
  - Implements all of the logic for a page without leveraging a standard controller;
  - If you want to override existing functionality;
  - Customize the navigation through an application;
  - Or if you need finer control for how information is accessed for your page;



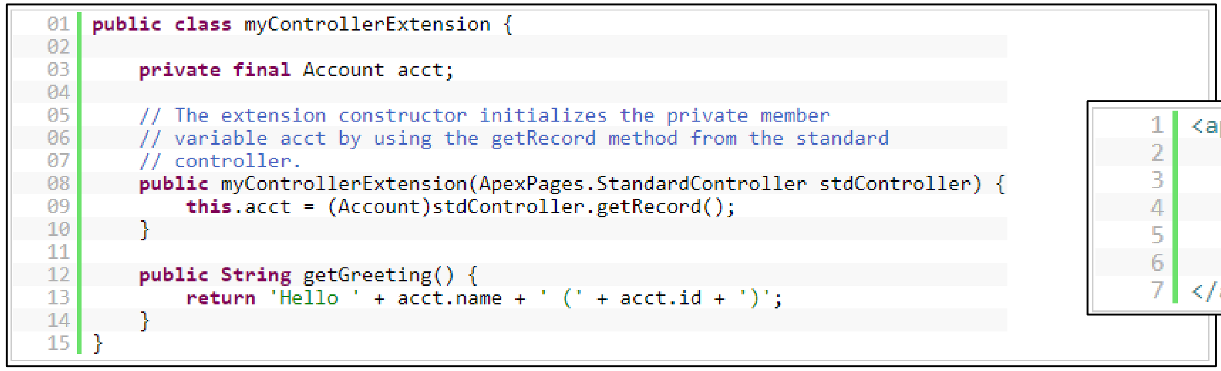
```
public class MyController {

    private final Account account;

    public MyController() {
        account = [SELECT Id, Name, Site FROM Account
                   WHERE Id = :ApexPages.currentPage().getParameters().get('id')];
    }

    public Account getAccount() {
        return account;
    }

    public PageReference save() {
        update account;
        return null;
    }
}
```



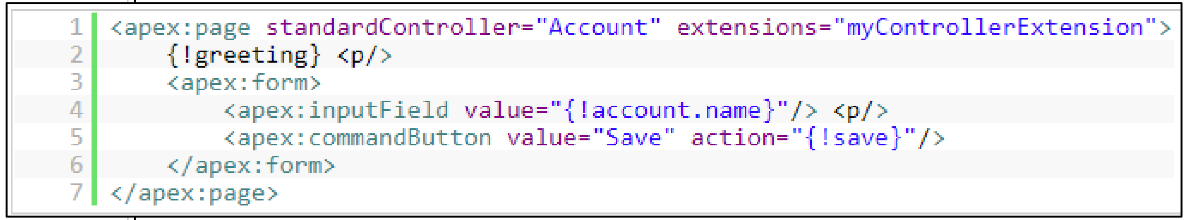
```
<apex:page controller="myController" tabStyle="Account">
    <apex:form>
        <apex:pageBlock title="Congratulations {!$User.FirstName}">
            You belong to Account Name: <apex:inputField value="{!account.name}"/>
            <apex:commandButton action="{!save}" value="save"/>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

Visualforce: Controllers

- **Controller Extension**
  - An Apex class that extends the functionality of a standard or custom controller;
  - You want to leverage the built-in functionality of a standard controller but override one or more actions, such as edit, view, save, or delete;
  - You want to add new actions;
  - You want to build a Visualforce page that respects user permissions:
    - it executes in user mode, in which permissions, field-level security, and sharing rules of the current user apply;



```
public class myControllerExtension {

    private final Account acct;

    // The extension constructor initializes the private member
    // variable acct by using the getRecord method from the standard
    // controller.
    public myControllerExtension(ApexPages.StandardController stdController) {
        this.acct = (Account)stdController.getRecord();
    }

    public String getGreeting() {
        return 'Hello ' + acct.name + ' (' + acct.id + ')';
    }
}
```



```
<apex:page standardController="Account" extensions="myControllerExtension">
    {!greeting} <p/>
    <apex:form>
        <apex:inputField value="{!account.name}"/> <p/>
        <apex:commandButton value="Save" action="{!save}"/>
    </apex:form>
</apex:page>
```

References

Visualforce Basics:
https://trailhead.salesforce.com/content/learn/modules/visualforce_fundamentals

Quick Start: Visualforce:
https://trailhead.salesforce.com/content/learn/projects/quickstart-visualforce

Visualforce & Lightning Experience
https://trailhead.salesforce.com/en/content/learn/modules/lex_dev_visualforce

Visualfoce Developer Guide
https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_intro.htm

Visualforce Training for Beginners
https://www.youtube.com/watch?v=YXYbZkSuEkU&list=PLdYQMTciVWO-_J9HB2NB-TApIornE33SI

Aura / LWC Overview

isobar

Lightning Components or Visualforce?

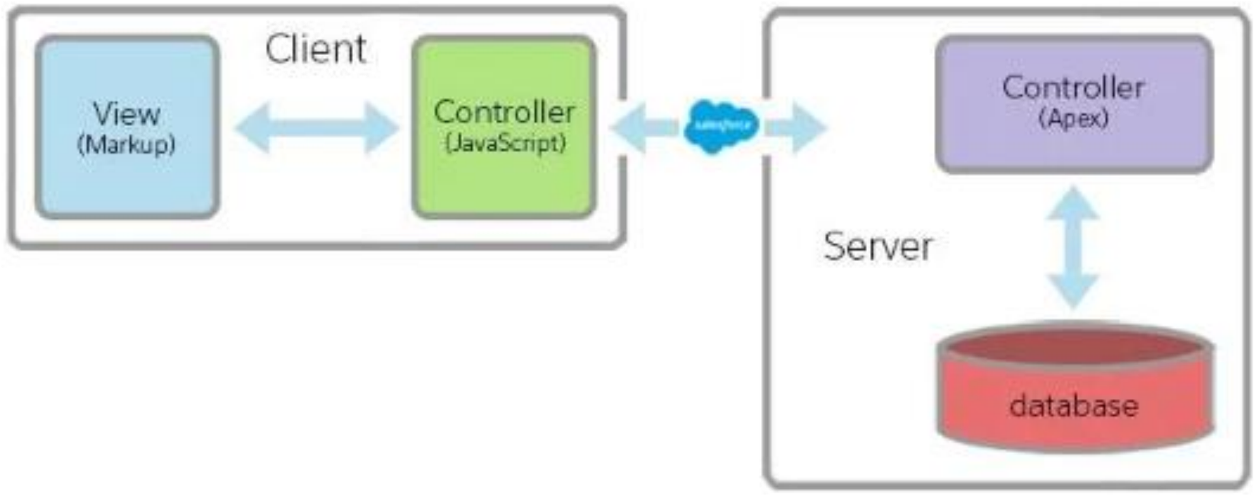
Visualforce and Lightning Components each have their strengths.

Visualforce Pages: Known as Salesforce Classic, is na example of **page-centric** web application model. It’s great for basic functionality, but it’s challenging to deliver the new, more dynamic experience that users expect, this is because it relies on the server to generate a new page every time you interact with the application;

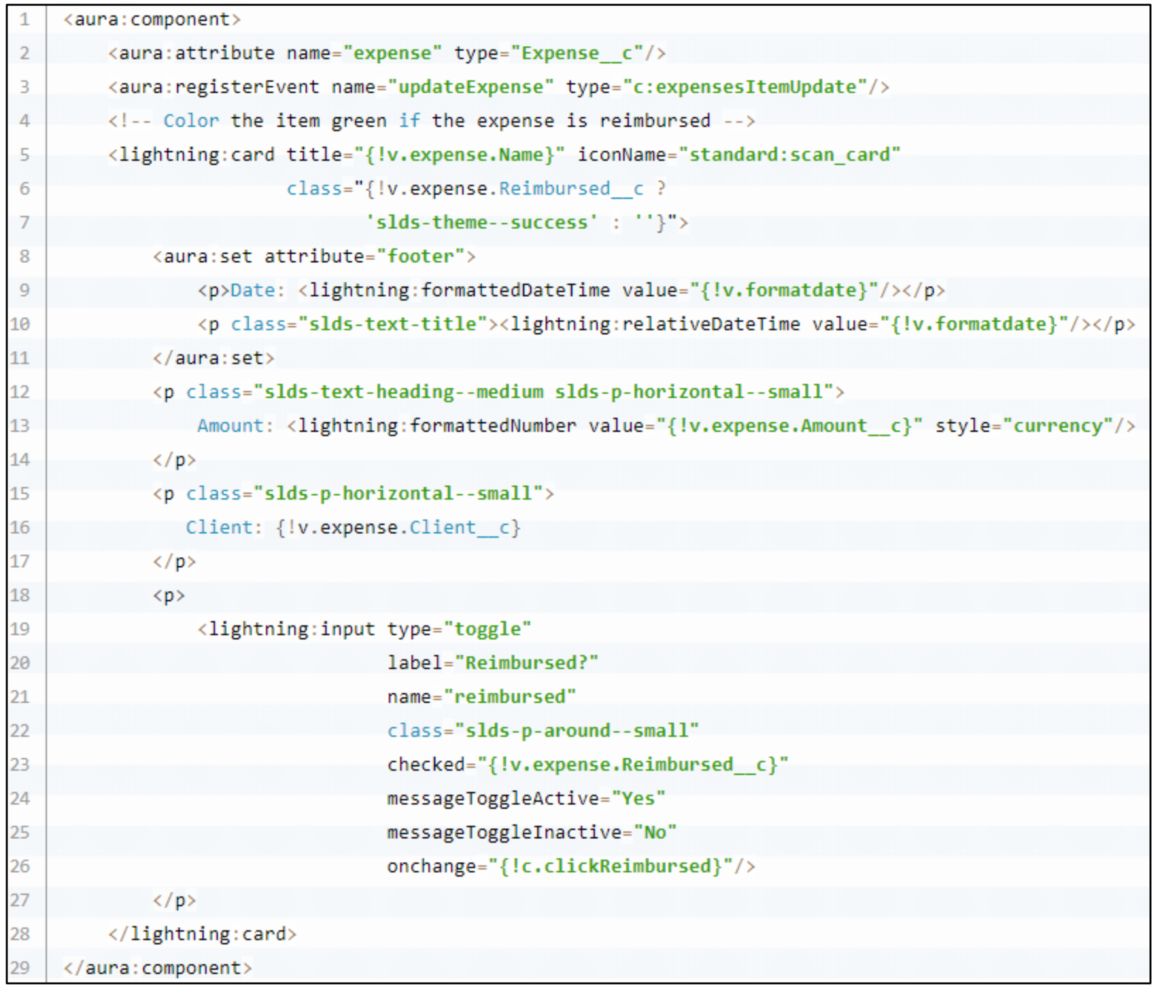
Lightning Components: To deliver a more interactive experience, you need help from JavaScript on the client-side. In this new **app-centric** model, JavaScript is used to create, modify, transform, and animate the user interface rather than completely replacing it a page at a time.

Aura Components (Lightning Components)

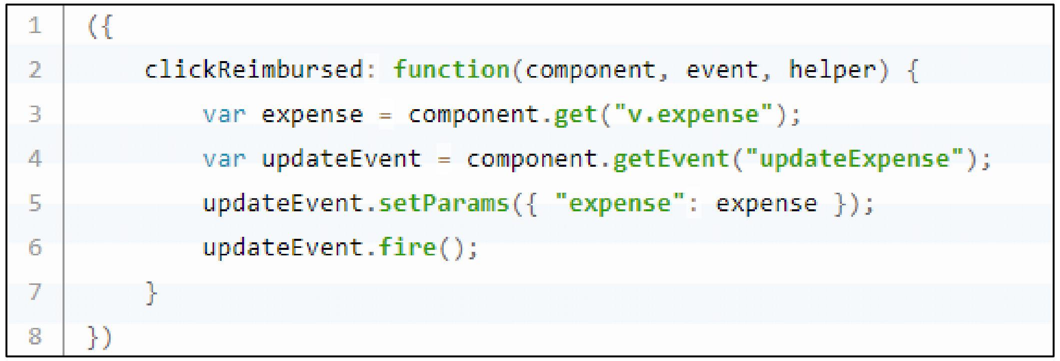
- The **Lightning Component** framework is a UI framework for developing web apps for mobile and desktop devices;
- It’s a modern framework for building single-page applications with dynamic, responsive user interfaces for Lightning Platform apps;
- It uses `JavaScript` on the *client side* and `Apex` on the *server side*.



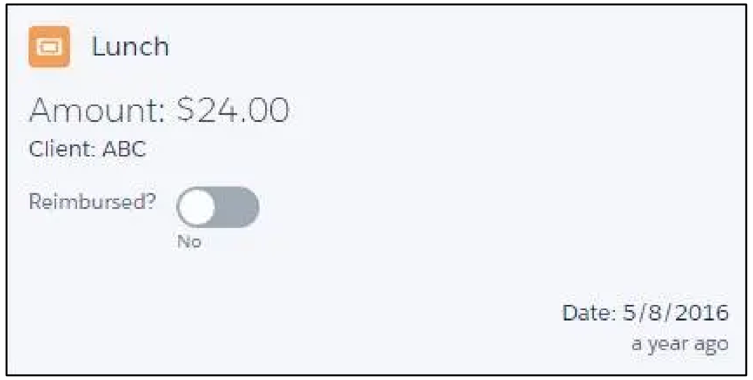
Aura Components (Lightning Components)



```
<aura:component>
    <aura:attribute name="expense" type="Expense__c"/>
    <aura:registerEvent name="updateExpense" type="c:expensesItemUpdate"/>
    <!-- Color the item green if the expense is reimbursed -->
    <lightning:card title="{!v.expense.Name}" iconName="standard:scan_card"
                    class="{!v.expense.Reimbursed__c ?
                           'slds-theme--success' : ''}">
        <aura:set attribute="footer">
            <p>Date: <lightning:formattedDateTime value="{!v.formatdate}"/></p>
            <p class="slds-text-title"><lightning:relativeDateTime value="{!v.formatdate}"/></p>
        </aura:set>
        <p class="slds-text-heading--medium slds-p-horizontal--small">
            Amount: <lightning:formattedNumber value="{!v.expense.Amount__c}" style="currency"/>
        </p>
        <p class="slds-p-horizontal--small">
           Client: {!v.expense.Client__c}
        </p>
        <p>
            <lightning:input type="toggle"
                             label="Reimbursed?"
                             name="reimbursed"
                             class="slds-p-around--small"
                             checked="{!v.expense.Reimbursed__c}"
                             messageToggleActive="Yes"
                             messageToggleInactive="No"
                             onchange="{!c.clickReimbursed}"/>
        </p>
    </lightning:card>
</aura:component>
```



```
({
    clickReimbursed: function(component, event, helper) {
        var expense = component.get("v.expense");
        var updateEvent = component.getEvent("updateExpense");
        updateEvent.setParams({ "expense": expense });
        updateEvent.fire();
    }
})
```



Lunch

Amount: $24.00

Client: ABC

Reimbursed? No

Date: 5/8/2016
a year ago

Aura Components (Lightning Components)

Where You Can Use Lightning Components

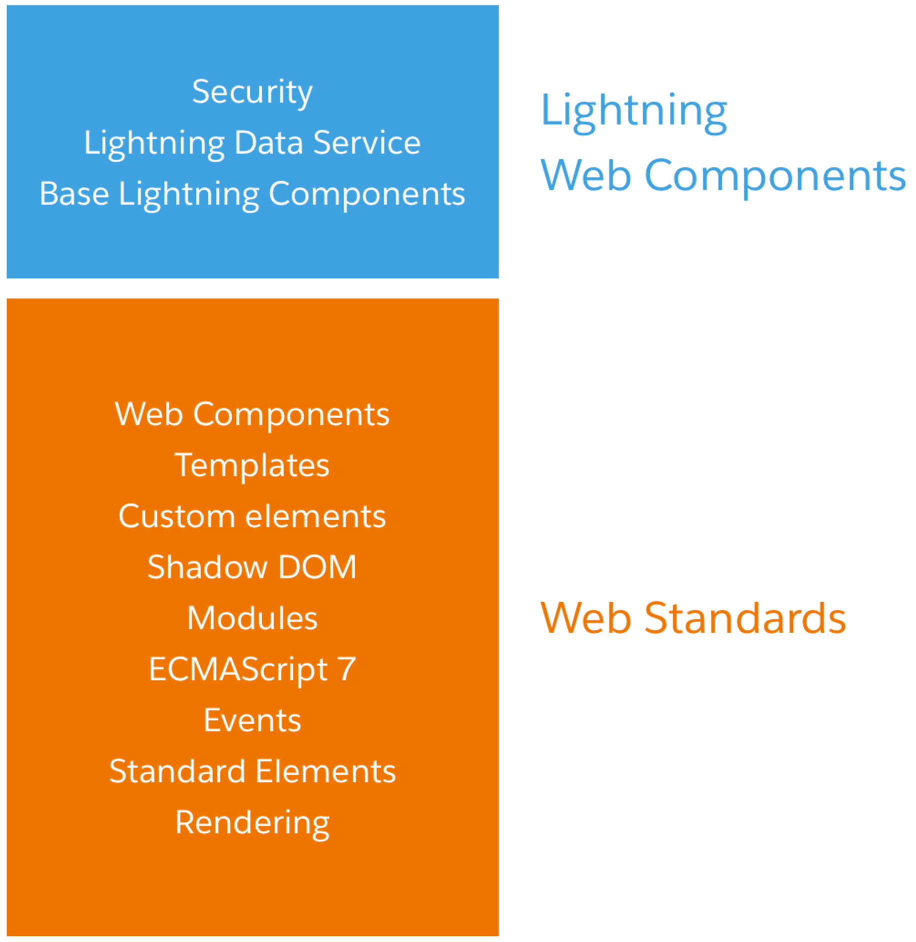
- Add Apps to the Lightning Experience App Launcher;
- Add Apps to Lightning Experience and Salesforce App Navigation;
- Create Drag-and-Drop Components for Lightning App Builder and Experience Builder;
- Add Lightning Components to Lightning Pages;
- Add Lightning Components to Lightning Experience Record Pages;
- Launch a Lightning Component as a Quick Action;
- Override Standard Actions with Lightning Components;
- Create Stand-Alone Apps;
- Run Lightning Components Apps Inside Visualforce Pages;
- Run Lightning Components Apps on Other Platforms with Lightning Out;
- Customize Flow Screens

Lightning Web Components

Open Door to Programming with Web Standards

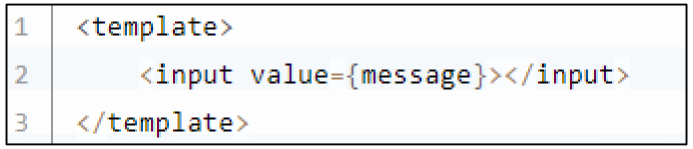
- Use of standard technologies like HTML, JavaScript, and CSS to build the next generation of Salesforce apps;
- Is focused on both the developer and user experience;
- Uses core *Web Components* standards and provides only what's necessary to perform well in browsers supported by Salesforce;
- Is lightweight and delivers exceptional performance;
- Most of the code you write is standard JavaScript and HTML;
- Lightning Web Components and Aura Components do work together, Aura components can contain Lightning web components, though not vice-versa.

Lightning Web Components



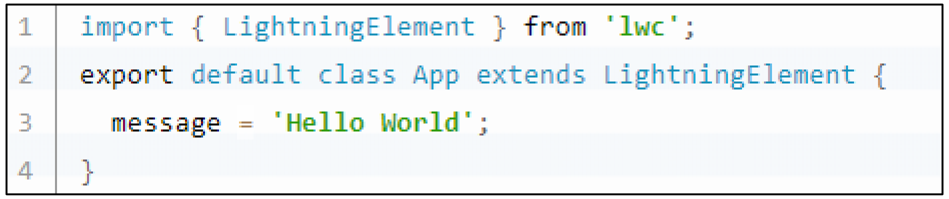
Lightning Web Components

HTML



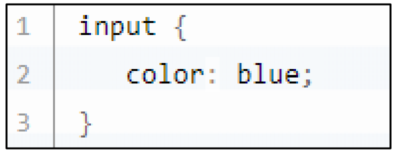
```html
<template>
    <input value={message}></input>
</template>
```

JavaScript



```javascript
import { LightningElement } from 'lwc';
export default class App extends LightningElement {
  message = 'Hello World';
}
```

CSS



```css
input {
   color: blue;
}
```

Lightning Web Components Playground:
https://developer.salesforce.com/docs/component-library/tools/playground

References

Aura Components Basics
https://trailhead.salesforce.com/content/learn/modules/lex_dev_lc_basics

Quick Start: Aura Components
https://trailhead.salesforce.com/en/content/learn/projects/quickstart-lightning-components

Aura Components Core Concepts
https://trailhead.salesforce.com/en/content/learn/modules/lex_dev_lc_vf_concepts

Build Flexible Apps with Aura Components
https://trailhead.salesforce.com/en/content/learn/projects/workshop-lightning-programmatic

Lightning Aura Components Developer Guide
https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/intro_framework.htm

References

Lightning Web Components for Aura Developers
https://trailhead.salesforce.com/en/content/learn/modules/lightning-web-components-for-aura-developers

Lightning Web Components Basics
https://trailhead.salesforce.com/en/content/learn/modules/lightning-web-components-basics

Quick Start: Lightning Web Components
https://trailhead.salesforce.com/en/content/learn/projects/quick-start-lightning-web-components

Build Lightning Web Components
https://trailhead.salesforce.com/en/content/learn/trails/build-lightning-web-components

Lightning : Sample Gallery
https://trailhead.salesforce.com/sample-gallery

Best Practices

Best Practices: Apex

- **Bulkify your Code**: refers to the concept of making sure the code properly handles more than one record at a time;
- **Avoid SOQL Queries or DML statements inside FOR Loops**: If you need to query, query once, retrieve all the necessary data in a single query, then iterate over the results. If you need to modify the data, batch up data into a list and invoke your DML once on that list of data.
- **Bulkify your Helper Methods**: any utility or helper methods are efficiently written to handle collections of records.
- **Using Collections, Streamlining Queries, and Efficient For Loops**: It is important to use Apex Collections to efficiently query data and store the data in memory. A combination of using collections and streamlining SOQL queries can substantially help writing efficient Apex code and avoid governor limits.
- **Avoid Hardcoding IDs**: When deploying Apex code between sandbox and production environments, or installing Force.com AppExchange packages, it is essential to avoid hardcoding IDs in the Apex code.

Best Practices: Apex

- **Streamlining Multiple Triggers on the Same Object**: to avoid redundancies and inefficiencies when deploying multiple triggers on the same object;
- **Querying Large Data Sets**: The total number of records that can be returned by SOQL queries in a request is 50,000. If returning a large set of queries causes you to exceed your heap limit, then a SOQL query **`for`** loop must be used instead.
- **Use of the Limits Apex Methods to Avoid Hitting Governor Limits**: Apex has a System class called **`Limits`** that lets you output debug messages for each governor limit (ex: `Limit.getLimitQueries()`).
- **Use `@future` Appropriately**: Apex written within an asynchronous method gets its own independent set of higher governor limits. No more than 10 `@future` methods can be invoke.d within a single Apex transaction.
- **Writing Test Methods to Verify Large Datasets**: Since Apex code executes in bulk, it is essential to have test scenarios to verify that the Apex being tested is designed to handle large datasets and not just single records.

isobar

Best Practices: Visualforce

- **Improving Visualforce Performance**: Check if the problem is only for a single user, the Load Time (large page sizes), the View State Size, Multiple Concurrent Requests, Queries and Security, Preventing Field Values from Dropping Off the Page;
- **Accessing Component IDs**: Use the **`$Component`** global variable to simplify referencing the DOM ID that is generated for a Visualforce component. (`<apex:form id="theForm"><apex:pageBlock id="theBlock">…`).
- **Static Resources**: Displaying the Content of a Static Resource with the action Attribute on `<apex:page>`.
- **Controllers and Controller Extensions**: Enforcing Sharing Rules in Controllers, You can do that by using the **`with sharing`** keywords in the class definition.
- **Rendering PDF Files**: reference static image and style sheet resources through the **`$Resource`** global variable.

References

Apex Code Best Practices:
https://developer.salesforce.com/page/Apex_Code_Best_Practices

Visualforce – Best Practices:
https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_compref_additional_best_practices.htm

Visualforce & Lightning Experience
https://trailhead.salesforce.com/en/content/learn/modules/lex_dev_visualforce

Visualfoce Developer Guide
https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_intro.htm

Visualforce Training for Beginners
https://www.youtube.com/watch?v=YXYbZkSuEkU&list=PLdYQMTciVWO-_J9HB2NB-TApIornE33Sl