**Learning Sales Force, the Basics** 

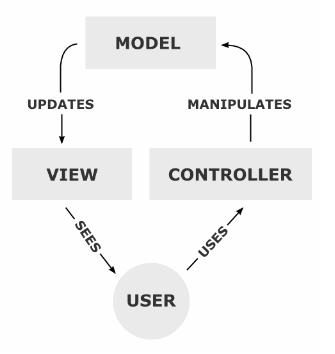
**By: Gabriel Moreira**

**The Idea**

* **Salesforce**
  + Tool to manage all the data of sales team of an organization.
  + Salesforce is a customer relationship management tool (CRM), which means it is used to keep track of and strengthen a company’s relationship with its existing and potential clients.
  + Salesforce is an account-centric CRM, so contacts and opportunities are always linked to an account.

**The Layer**

* **VisualForce**
  + VF is the layout and APEX is the work behind it.
    - VisualForce is the interactive layout, with bottoms and etc. APEX is the language doing the work underneath this layout that allows VF to run
* **APEX**
  + Apex provides a complete set of features for building business applications
    - Including data models and objects to manage data, a workflow engine for managing collaboration of that data between users, a user interface model to handle forms and other interactions, and a SOAP API for programmatic access and integration.
  + Apex Code extends the Force.com platform by introducing the ability to write code that runs on salesforce.com servers.
  + The language enables a new class of applications and features to be developed and deployed entirely on demand.
    - These applications make existing Force.com apps “smarter” by providing the ability to capture business logic and rules
      * E.g. – such as data validation – and make entirely new kinds of apps possible on demand – such as complex inventory checking and order fulfillment.
  + Sales force language, very similar to Java and some database SQL implementations
  + You can write your own Controller’s (or extensions) and Model Classes using Apex
* **MVC**
  + A software architecture pattern which separates the representation of information from the user’s interaction with it.
  + In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.



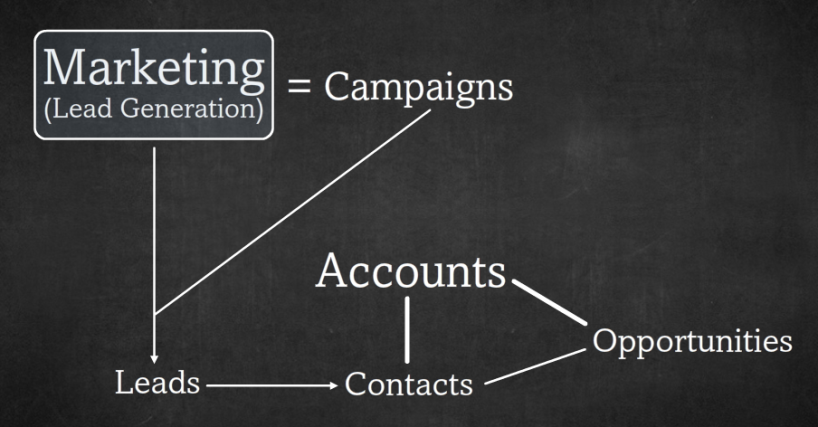
* **Controller**
  + Can send commands to its associated view to change the view’s presentation of the model
    - e.g., by scrolling through a document.
  + It can also send commands to the model to update the model’s state
    - e.g., editing a document.
  + How the interface actions. Controllers are used to perform the actions whenever users interact with visual force.
* **Model**
  + Notifies its associated views and controllers when there has been a change in its state.
    - This notification allows the views to produce updated output, and the controllers to change the available set of commands.
    - A *passive* implementation of MVC omits these notifications
  + What schema and data does salesforce uses to represent the system completely. In salesforce.
    - We can say that sObjects are the model as every entity in salesforce is mapped to some sObject.
* **View**
  + Requests from the model the information that it needs to generate an output representation.
  + How the schema and data is represented. Visualforce is used to present the data to users.
* VF pages = **Visual Force**
  + Each is associated with a Controller
    - You can write your own Controller with APEX
  + Uses the traditional model-view-controller (MVC) paradigm, with the option to use auto-generated controllers for database objects, providing simple and tight integration with the database
* **SFDC**
  + Visual Force pages, Page Layouts, Tabs comes under View Layer of Model View controller
  + Workflows, Apex Classes, Triggers, validation rules comes under Controller part in Model View controller
  + Objects, Fields, Relationships comes under Model Layer of Model View Controller

**Terminology**

* **Instance**
  + A complete set of systems, network and storage infrastructure, both shared and non-shared, that provides the salesforce.com service to a subset of our customers.
    - E.g, na14.salesforce.com is an instance.
* **Superpod**
  + A set of systems, network and storage infrastructure, including outbound proxy servers, load balancers, mail servers, SAN fabric and other infrastructure supporting multiple instances.
  + Superpods provide service isolation within a datacenter so that problems with shared or complex components cannot impact every instance in that datacenter.
* **Org/Organization**
  + A single customer of the Salesforce application.
  + An org is highly customizable and can have distinct security settings, record visibility and sharing settings, UI look and feel, workflows, triggers, custom objects, custom fields on standard salesforce.com CRM objects, and even custom REST APIs.
  + An org can support anywhere from one to millions of licensed individual user accounts, portal user accounts and Force.com Sites user accounts.
* **Sandbox**
  + An instance of the salesforce.com service that hosts full copies of production orgs for customer application development purposes.
* **SaaS (Software as a Service)**
  + Is a way of delivering applications over the Internet—as a service.
    - Instead of installing and maintaining software, you simply access it via the Internet
* **Workflow Rules**
  + Give you the ability to enforce key business process easily without needing to write any code
  + Automated processes that take place
* **Creating Workflow Rules**
  + *Setup* 🡪 *Build* 🡪*Create* 🡪*Workflow & Approvals* 🡪 *Workflow Rules* 🡪click on *New Rule* button
    - Select the Object 🡪 Next 🡪 enter rule name, description, and the fields 🡪 Save & Next 🡪 Add Workflow Action 🡪 Enter the description from the action 🡪 Save 🡪 Done

**Salesforce Objects**

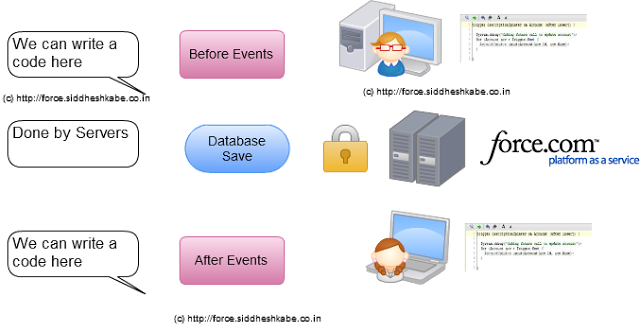
* **Objects** 
  + Are database tables that allow us to store data specific to organization in salesforce.
    - Note that this is not the same definition of regular programming languages Object
* **Five main objects**
  + Accounts, contact, opportunities, leads, and campaigns.
* **Standard Objects**
  + The objects provided by salesforce.com
    - E.g: accounts, contacts, opportunities, Leads, products, campaigns, cases, users, contracts, Report and dashboards etc…
* **Custom Objects**
  + Objects created by us are called custom objects.
  + Store information that is unique and important to your organization.
  + Are the heart of any application.
  + Provide a structure for sharing data.
    - Custom objects have properties such as: Custom fields, Relationship to other object, Page Layouts, A custom user interface tab
* **To create a Custom Object**
  + *Setup* 🡪 *Build* 🡪 *Create* 🡪 *Objects* 🡪 Click on *new custom object* button and enter label name, plural label and object name.
  + And also enter Record Name according to the data type. There are two data types available to create record name, those are
    - Text
    - Auto Number
  + Allow Reports
    - If we check this check box then only these objects are available to create reports.
  + Allow Activities
    - If we check this chis check box then we are able to create activities on this object.
  + Track Field History
    - If we check this check box then only we are to track fields. We can track up to 20 fields for single object. And also we following deployment Status
  + In Development
    - If we check this check box, this object is still in development mode. This object is not available for deployments.
  + Deployed
    - After selecting this check box then only it will be available for deployment.
  + After completing all the details click on save.
  + If we does not select “Launch new custom tab wizard” from object creation page, the object will save without tab appearance. In this case we have to create tab for this object.
    - If we select this check box the object will save and tab will be created and appeared.
* **Tabs**
  + Is a User interface to create records for object and to view the records in objects.
  + Salesforce has three types of tabs
    - Custom Object Tab
    - Web Tab
    - Visualforce Tab
  + **Create Custom Tabs**
    - *Setup* 🡪 *Build* 🡪*Create* 🡪*Tab* 🡪click on *New tab* and enter the details to complete the tab creation process.
* **Accounts**
  + Are the companies you deal with
* **Contacts**
  + Are the people at those companies
* **Opportunities**
  + Are the deals you’re making with those companies to sell them your product or service.
  + Salesforce is an account-centric CRM, so contacts and opportunities are always linked to an account.
* **Leads**
  + They're the people who have shown some interest in your company, but not enough to take the next step in the sales process.
  + Leads are generated by your company's marketing efforts, like speaking engagements, webinars, etc.
* **Campaigns**
  + Salesforce's way of keeping track of marketing efforts.
    - Say Mark goes to a conference and gives a talk; he gets a list of all the people who attended. I could create a campaign in Salesforce for that specific conference and talk, and when I imported those attendees as leads, they would be linked to that campaign. Campaigns are a good way to track where your leads are coming from, and how your marketing efforts are doing.



* **Marketing efforts**
  + Are linked to campaigns
  + Generate leads
    - Once a lead shows the requisite level of interest, depending on the company and sales process, it's ready to become a contact, and it is converted.
      * During the conversion process, the new contact is linked to a new or existing account and an opportunity may be created.

**Triggers**

* Ever run into workflow limitations?
  + Your workflow can’t create or update a separate object or you’re not allowed to reference certain fields, like Owner.Name (why Salesforce, why!)
* Imagine triggers to be like workflows without limitations.
  + Like workflows, you define entry criteria and actions.
  + Unlike workflows, you can make each of these do or reference anything you want.
* There exists **two types of Triggers**
  + Before Triggers
    - Are used to update or validate record values before they are saved to the database.
    - A before insert trigger can change the value in the field.
    - We won't get before undelete trigger.
  + After Triggers
    - Can be used for a much more complex purposes, like updating a value of some other object when this is updated.
    - Or it can also be used to trigger some other event
      * like workflow and approval process
    - The events that run behind the scenes are as follows:



* So choose the object, choose the event and finally choose the type of trigger.
* Context Variable
  + When we write a trigger, the Force.com server automatically sends us the value that was operated on in a variable called Trigger, this is called Context Variable
* **Writing Triggers**
  + *Problem*: When the User is entering the Opportunity, check for the Opportunity Amount. If the Opportunity Amount is greater than 50,000. Mark the Parent Account as 'Featured'.
  + *Solution*: We will be modifying the parent Account Type field to 'Featured' and select value from Opportunity Amount**.**
  + *Trigger:*
    - trigger SampleTrigger on Opportunity (after insert) {

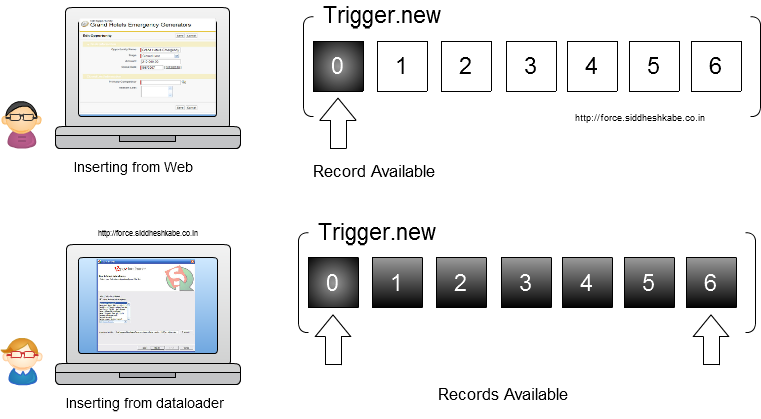
Opportunity TheOpportunity = trigger.new[0]; if(TheOpportunity.amount>50000){

Account theAccount= [Select Type, Amount, name from Account where id:=theOpportunity.Accountid]; theAccount.type='Featured'; update theAccount;

}

}

* + *Line by line explanation*
  + trigger SampleTrigger on Opportunity (after insert) {
    - Trigger Keyword suggest this is a trigger
    - the SampleTrigger is the name of the trigger.
    - The trigger is written on the Opportunity Object.
    - The Event that the code runs is written inside the bracket.
      * The following operations are possible in a trigger.
        + Insert, update, delete, merge, upsert, undelete
  + Suppose I am updating the Opportunity named 'Opportunity1' on the opportunity edit page.
    - In the before update opportunity trigger, Force.com will send all the values that were existing before the record was updated in the list called Trigger.Old and the new values that the user added in the list called Trigger.New.
    - If the record is updated from the web browser, Salesforce application we will find always find the value in Trigger.new[0], if the record is updated using a data loader, we need to loop through Trigger.new one record at a time as shown in the figure below.
  + Opportunity TheOpportunity = trigger.new[0];



* + if(TheOpportunity.amount>50000) { --------- code here----}
    - We select the Amount field and check if it is greater than 50,000. If it is, we can fetch the parent Account on the Opportunity using the AccountId field.

**Formulas**

* Objects persisted in the database tend to have field types that store data, for example numbers or text. They can also have a formula field type, which is calculated at run-time much like formulas in a spreadsheet.
* A formula is similar to an equation that is executed at run time.
  + Depending on the context of the formula, it can make use of various data and operations to perform the calculation.
  + Besides their use as a field type, formulas are used to determine when certain actions should occur on the platform
    - E.g. when a workflow rule should fire, and in validation rules that determine when data is bona fide.
* Formulas can make use of data and operations to calculate a new value of some type.
  + E.g. 
* **Syntax**
  + A formula can contain references to the values of standard or custom fields, operators, functions, literal values, or even other formulas.
  + **Literal Values**
    - A text string or number you enter that is not calculated or changed.
      * E.g, 2.0 or 'Hello World'
  + **Field References**
    - Reference the value of another field using a merge field. The syntax for a merge field is field\_name for a standard field or field\_name\_\_c for a custom field.
      * E.g., My\_Field\_\_c.
  + **Global Variables**
    - Reference the value of a system or environment variable,
    - They start with a ‘$’ on the front
    - such as information about the logged in user or the organization settings.
      * E.g., $User.Department.
  + **Functions**
    - A collection of operations and conditional operations that when used within a formula field allow you to perform such things as math calculations, if/and/or and string manipulation.
    - Functions are grouped into four primary categories: Date & Time, Logical, Math and Text.
    - In addition, there are a number of advanced functions including regular expressions, and vlookup ability.
      * These are currently only available in some formula contexts, not all.
    - Like a method
  + **Operator**
    - Operators fall into two categories;
      * Math operators that specify the type of calculation to perform or the order in which to do it,
        + e.g. math operators include +,-,\*
      * Logical operators that evaluate if a value is greater than, less than, equivalent or true/false.
        + E.g. logical operators include >, <, &&, == and so on.
* **Formula Field**
  + Is a read-only field that derives its value from a formula expression you define
  + A formula field looks similar to an equation made up of different functions and source fields, and is automatically updated when any of the source fields change
  + These fields are read only, and are also accessible via the Force.com SOAP API
  + The values are calculated every time they are needed; they are not stored.
  + Formula fields also let you set dynamic default values for new records.
    - Say that your company has established a sales process, which requires that all new opportunities must be followed up within 5 days of being created in the system. The first thing to do is to create a new custom Date field on the Opportunity object called FollowUpDate. On the same step where you define the field name and label, you will have the option to define a Default Value. In this case a default value of TODAY() + 5 will give the desired effect - a value that is dynamic (it varies depending on the day that the record is created!).

**Resources**

Model View Controller (MVC): <http://www.salesforcetutorial.com/model-view-controller-mvc/>

Sales Force Architecture: <http://highscalability.com/blog/2013/9/23/salesforce-architecture-how-they-handle-13-billion-transacti.html>

Sales Force Objects: <http://www.salesforcetutorial.com/salesforce-objects/>

Salesforce for Beginners: <http://www.newfangled.com/an_introduction_to_salesforce>

Intro to Formulas: <https://developer.salesforce.com/page/An_Introduction_to_Formulas>

Triggers: <http://www.sfdc99.com/2013/05/09/what-is-a-salesforce-trigger/>

Writing basic trigger: <http://force.siddheshkabe.co.in/2012/01/basic-of-writing-apex-trigger.html>

What is APEX: https://developer.salesforce.com/page/Apex\_Code:\_The\_World's\_First\_On-Demand\_Programming\_Language