Salesforce : développement ‘par clic’, ou via Apex.

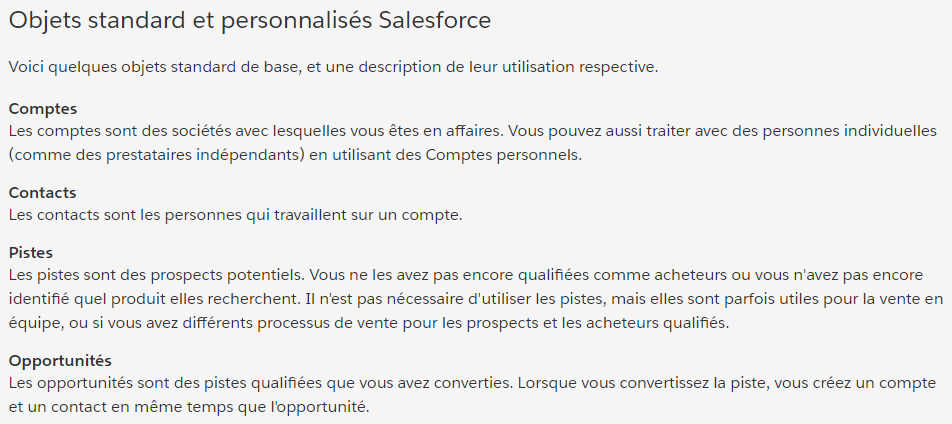
Emilie Waille : emilie.waille@ei-technologies.com

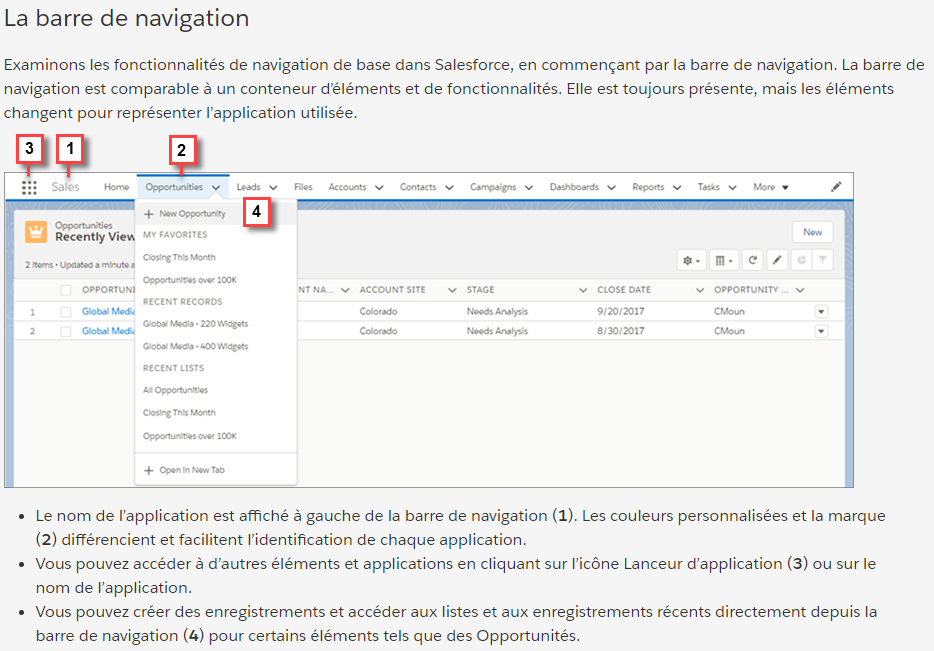
Salesforce lightning

Code wifi : EITECH922 (sales Einstein et bâtiment)

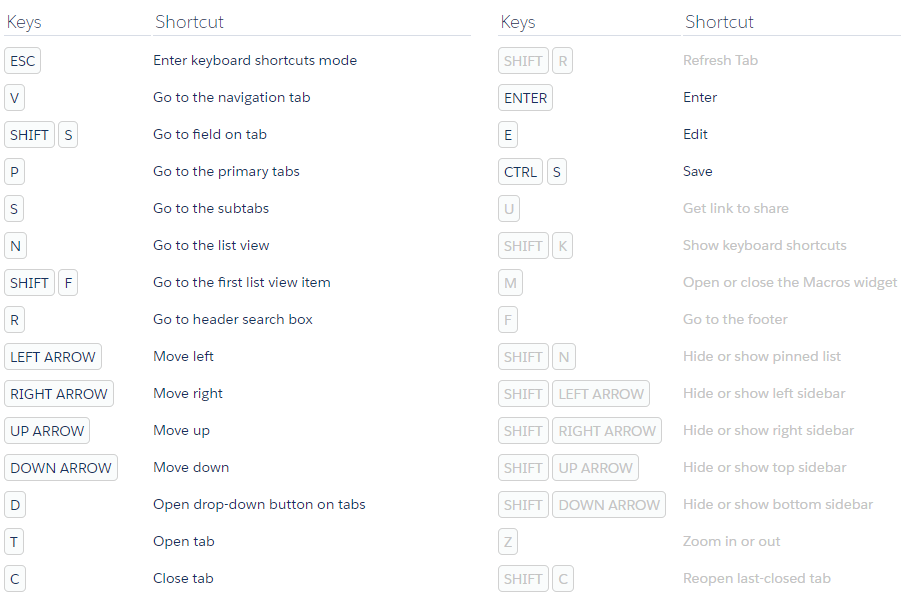
CRM [clic](https://fr.wikipedia.org/wiki/Gestion_de_la_relation_client)

[DEV450 : mon trail](https://trailhead.salesforce.com/en/users/00550000007ZpJRAA0/trailmixes/dev-450-alexis/)





Les raccourcis dans Salesforce



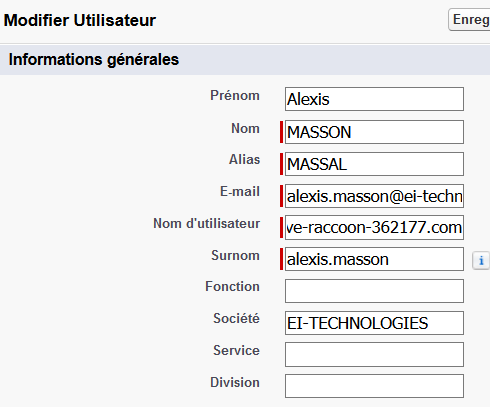
Les raccourcis dans la console developpeur



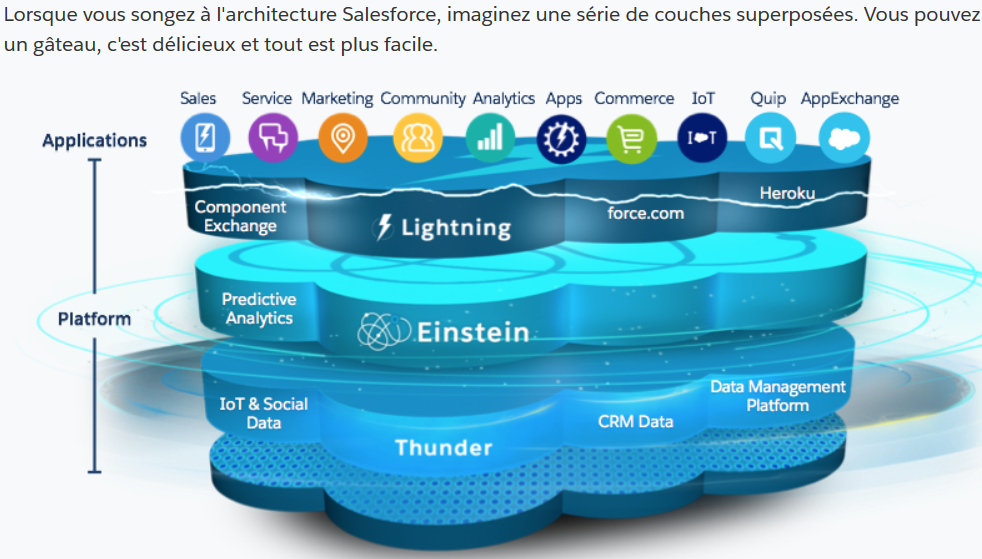
La gestion de la relation client (GRC), ou gestion des relations avec les clients, en anglais customer relationship management (CRM), est l'ensemble des outils et techniques destinés à capter, traiter, analyser les informations relatives aux clients et aux prospects, dans le but de les fidéliser en leur offrant ou proposant des services.

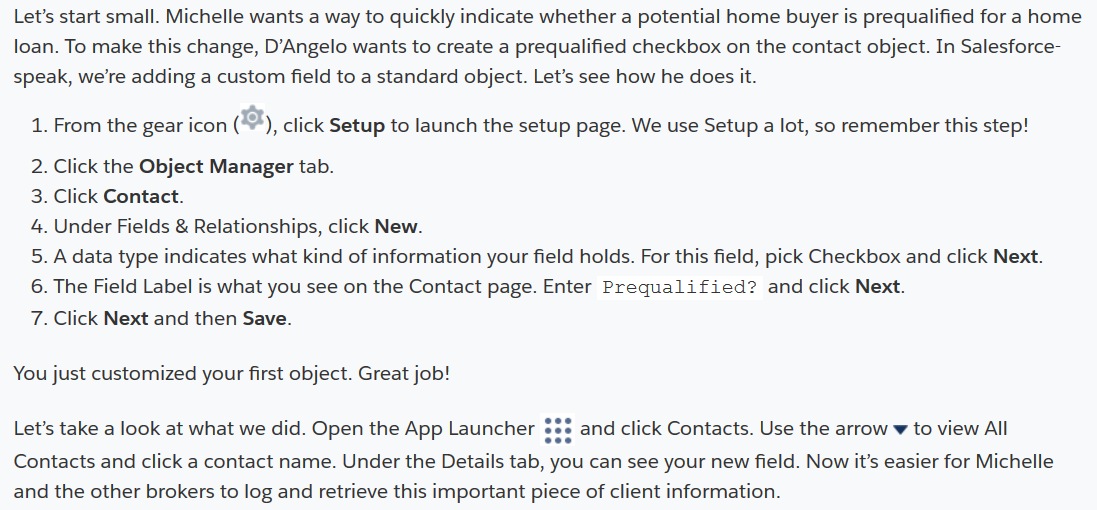
En ce qui concerne les applications informatiques, il s'agit notamment des progiciels qui permettent de traiter directement avec le client, que ce soit sur le plan de la vente, du marketing ou du service, et que l'on regroupe souvent sous le terme de « front-office », ceci par opposition aux outils de « back-office » que sont les progiciels de gestion intégrés (PGI / ERP).

Trailhead : plus de 200 modules sur l’ensemble de l’offre de Salesforce, trailhead permet de s’entrainer pour la maitrise de Salesforce.

  
[alexis.masson@brave-raccoon-362177.com](mailto:alexis.masson@brave-raccoon-362177.com)  
[alexis.masson@ei-technologies.com](mailto:alexis.masson@ei-technologies.com)  
MASSAL

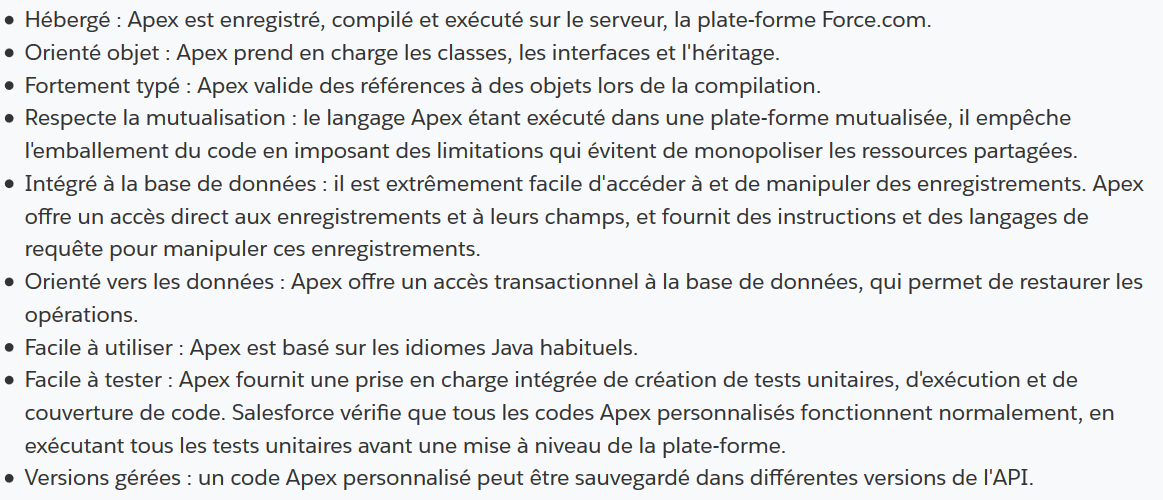
Dans SalesForce, privilégier les listes plutôt que les tableaux.

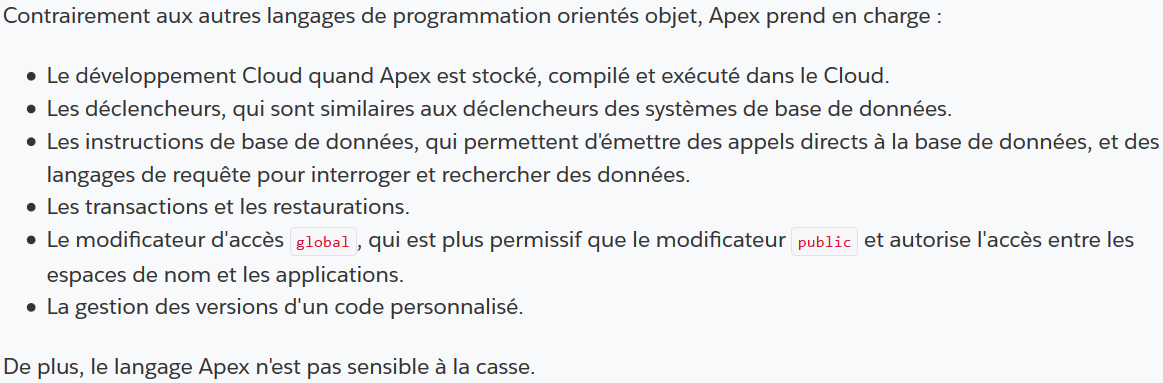


Ajout d’un bail :  


SaaS : software as a service

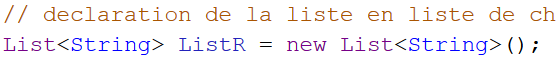
Bases du code [cours SalesForces](https://trailhead.salesforce.com/fr/modules/platform_dev_basics/units/platform_dev_basics_code)  
  
Les composants Lightning utilisent JavaScript côté client et Apex côté serveur.



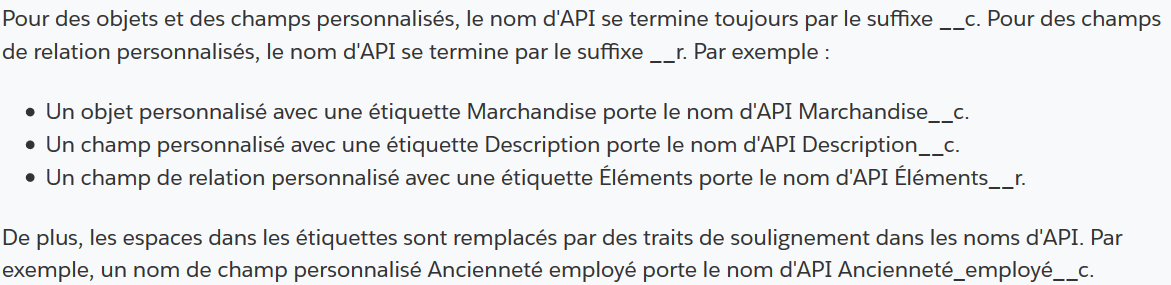


Mot cléf ‘global’ plus permissif que ‘public’.

Utile :  
‘Les méthodes statiques sont plus faciles à appeler que les méthodes d'instance, car il n'est pas nécessaire de les appeler dans une instance de la classe. Elles sont appelées directement dans le nom de la classe.’  
  
Pas besoin donc d’instancier un objet  
**Object obj = new Object() ;  
obj.methode() ;**  
  
On appelle directement :  
**NomClasse.methode()**

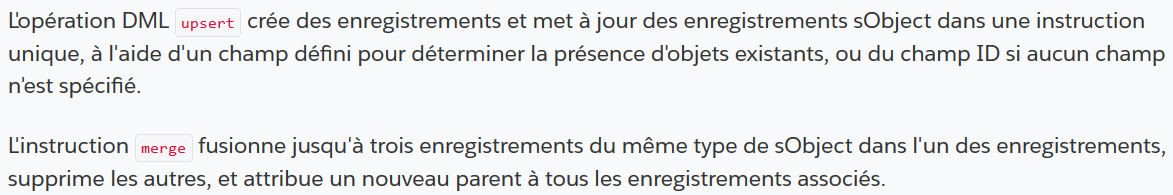
**Attention** à la façon de déclarer une liste :  


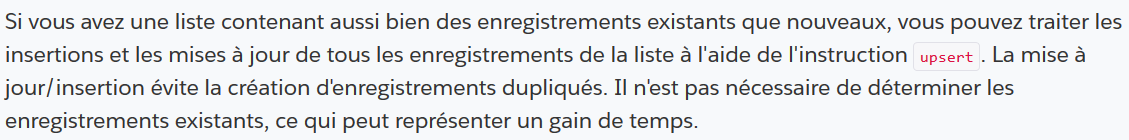
De plus, pour l’ajout d’éléments à la liste, on utilise :  

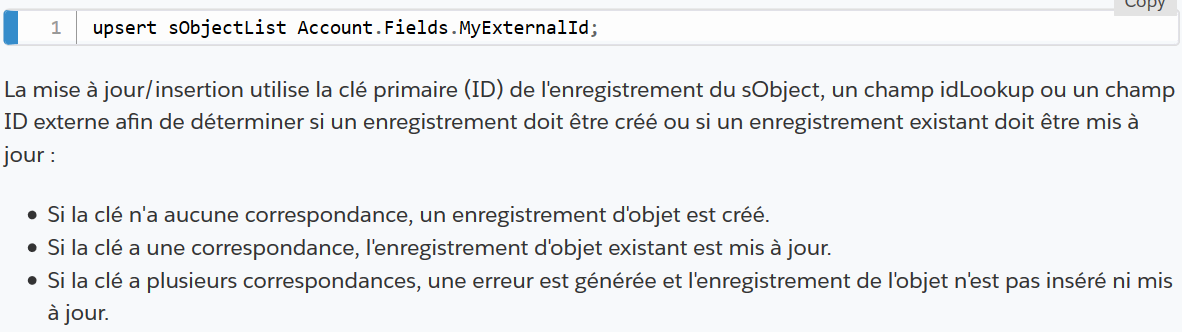



Sauvegarder puis manipuler un objet : DML 🡪 Data Manipulation Language.  
Récupérer un enregistrer : **SOQL** 🡪 Salesforces Object Query Language  
  
Chaque enregistrement dans Salesforce est représenté nativement en tant que sObject dans Apex.

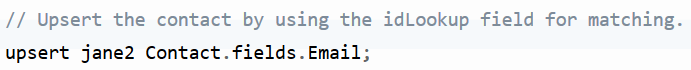


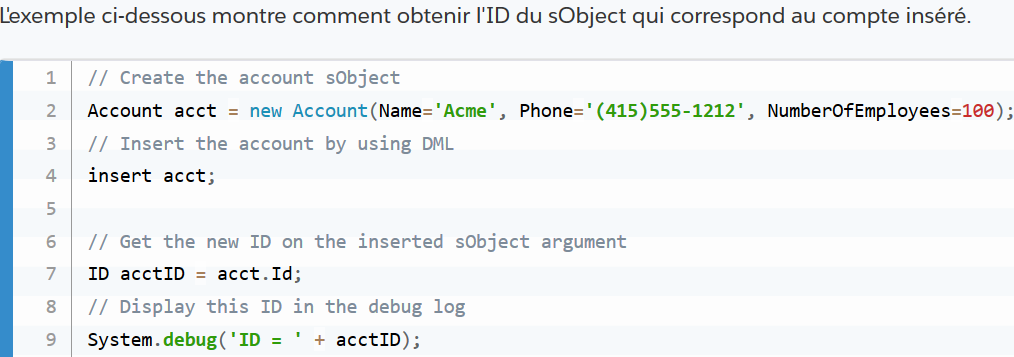
  
Quant à l’upsert :

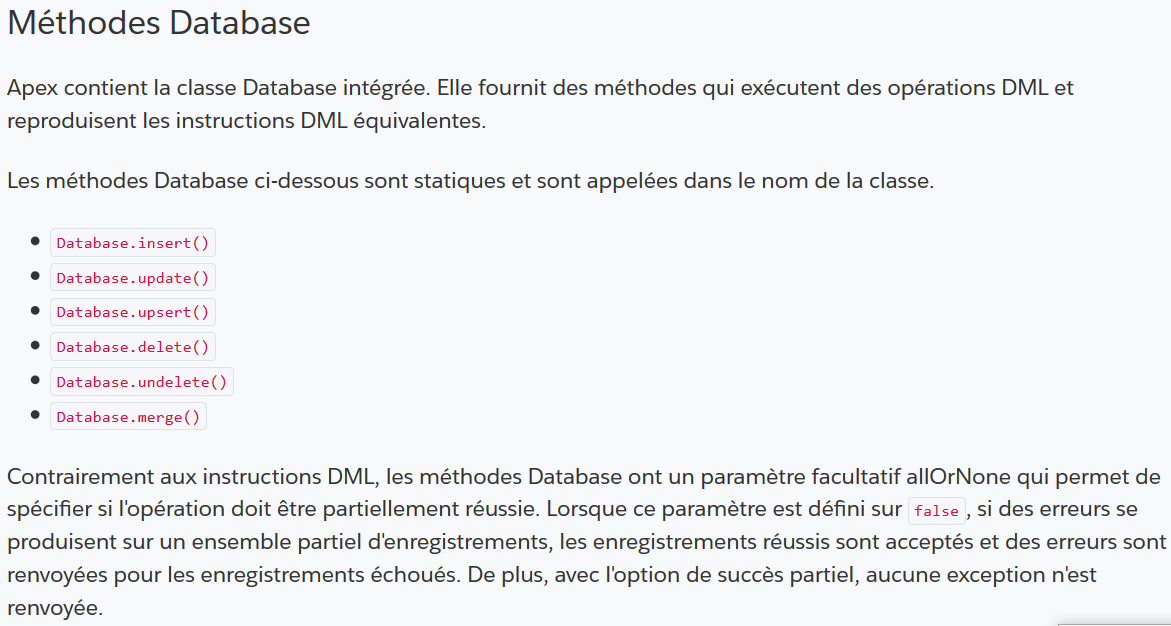
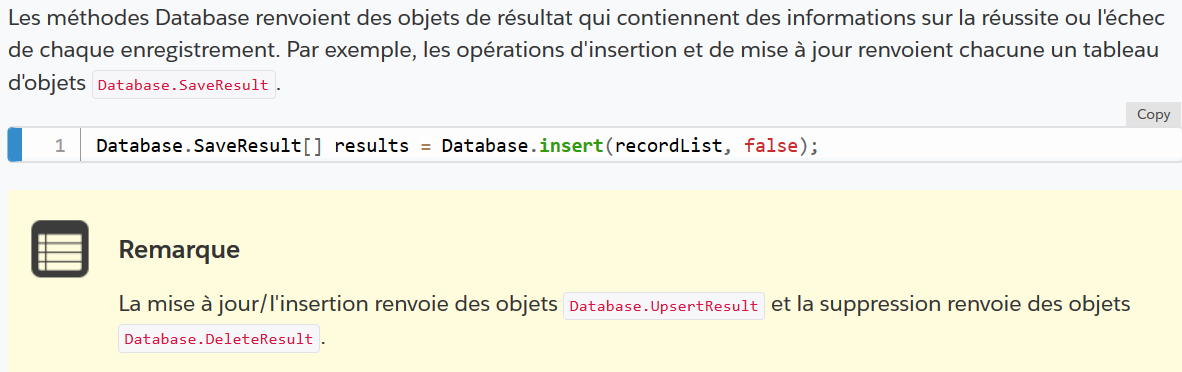


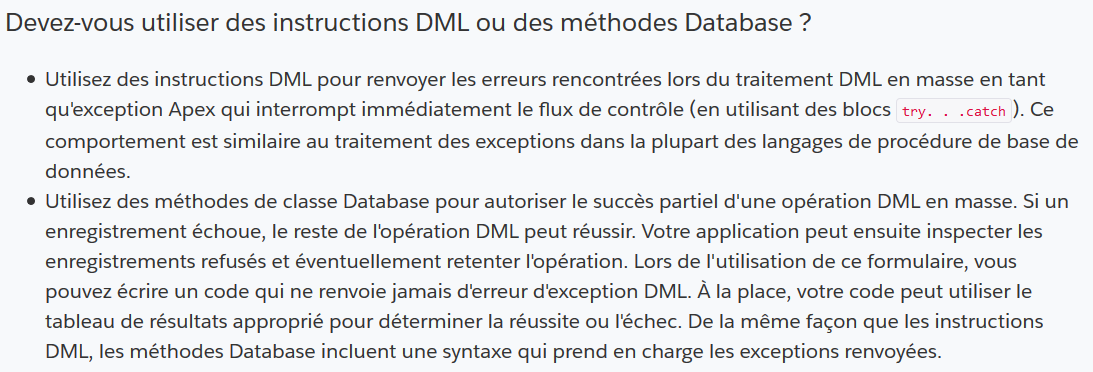


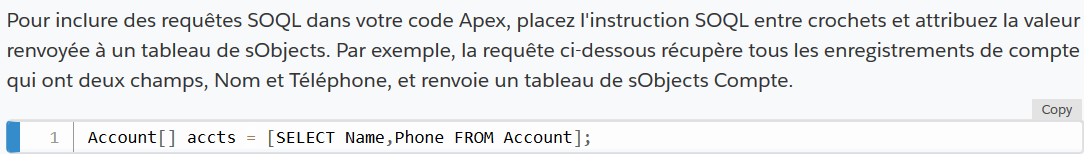
La ligne suivante permet d’ajouter le contact jane2, si son mail n’est pas déjà assimilé à un autre contact.



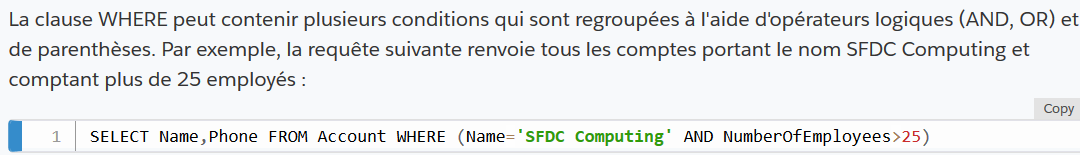
Récupérer l’ID d’un objet tout juste inséré :  


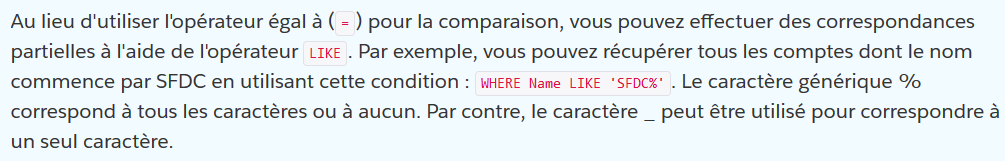
  




SOQL - Salesforces Object Query Language  
  


Contrairement aux autres langages SQL, vous ne pouvez pas spécifier d'astérisque (\*) pour tous les champs. Vous devez spécifier explicitement chaque champ souhaité.  
Ceci parce qu’exécuté sur le cloud, ne dois pas être trop lourd.



Recherche d’une correspondance incomplète :  


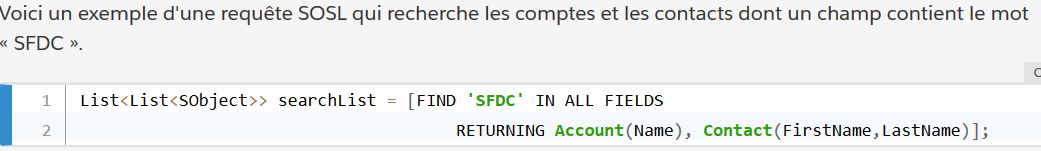
|  |  |  |
| --- | --- | --- |
| Recherche de caractère(s) (Excel) | Equivalent Saleforce | objectif |
| \* | % | Tous les caractères |
| ? | \_ | Un caractère |

**Order by name desc** marche très bien.  
  
Vous pouvez limiter le nombre d'enregistrements renvoyés à une valeur arbitraire en ajoutant la clause **LIMIT n**, où n est le nombre d'enregistrements renvoyés voulu :  

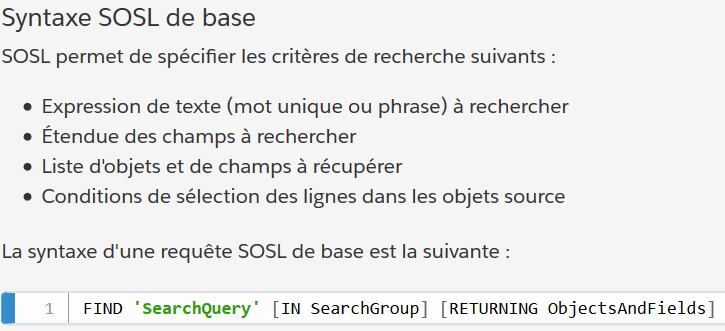


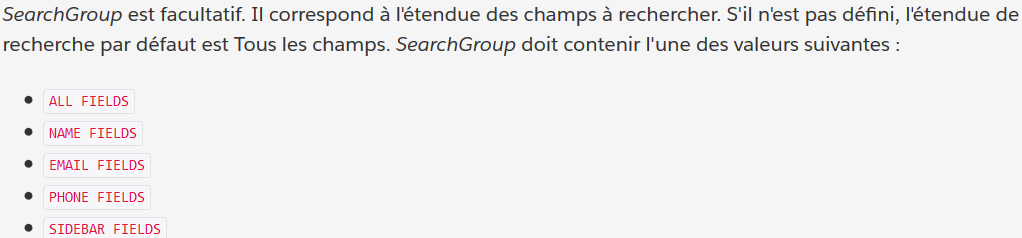

On peut remplir direct une liste avec la query :  
Contact[] conList = [query SOQL] ;

**SOSL – Salesforce Object Search Language :**

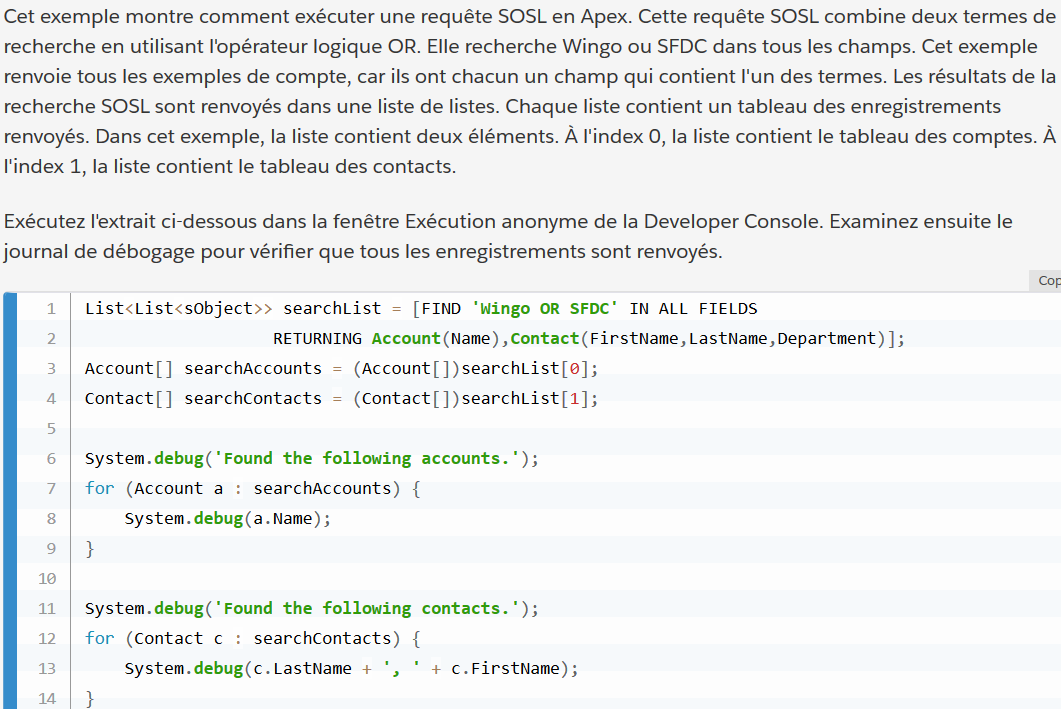
Salesforce Object Search Language (SOSL) est un langage de recherche Salesforce utilisé pour effectuer des recherches de texte dans des enregistrements.  
L'ajout de requêtes SOSL à Apex est simple : vous pouvez incorporer des requêtes SOSL directement dans votre code Apex. Lorsqu'il est incorporé dans Apex, SOSL est appelé SOSL en ligne.  


SOSL mappe les champs à l'aide d'une correspondance sur un mot, alors que SOQL effectue une correspondance exacte par défaut (quand aucun caractère générique n'est utilisé).  
Par exemple, la recherche de « Numérique » dans SOSL renvoie les enregistrements qui contiennent les valeurs de champ « Numérique »

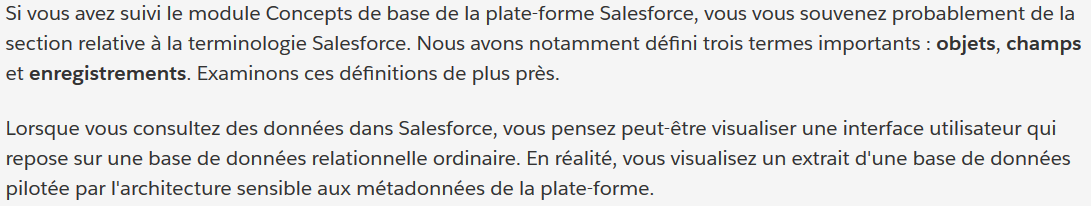


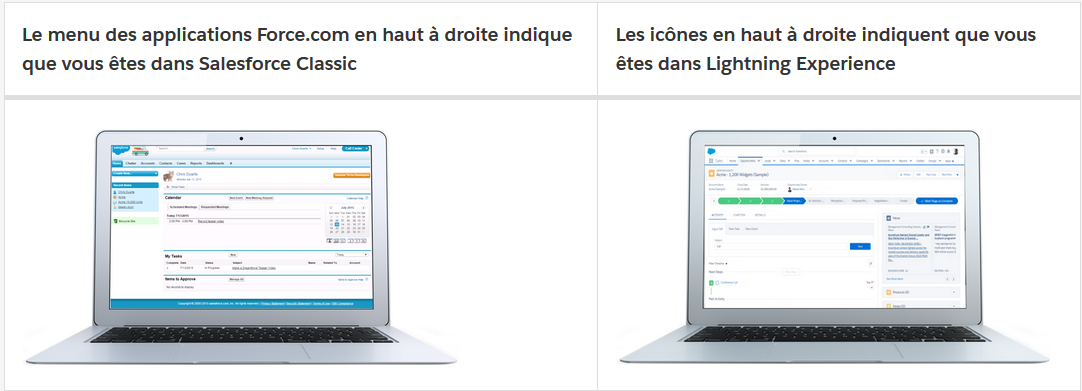


Exemple de requête :

Récuperation puis un cast pour bien ordonner le résultat, qui est une liste de liste (de sObjects)  


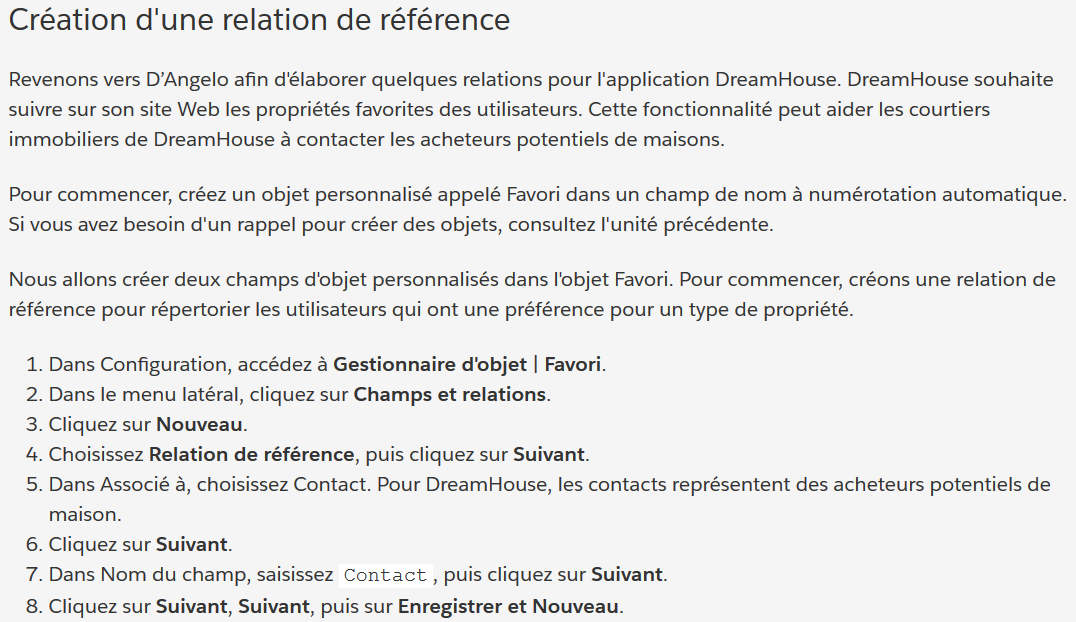
Quant aux métadonnées :

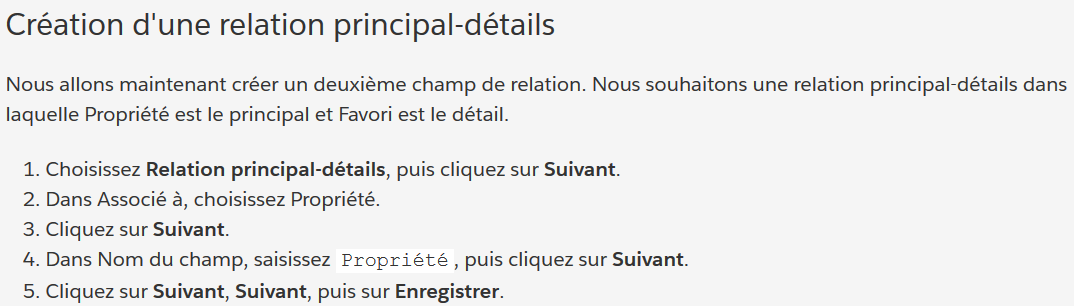


Différences Salesforce Classic 🡪 Salesforce Lightning Experience  


Créer des objets personnalisés



Création d’une relation principal-détail :  


Création d’une relation principal-détails :  


Reprise le 25 avril 2018

A bosser :  
Student guide - Building Applications with Force.com  
SFDEV-401V35-SGA4

Exercice guide - Building Applications with Force.com  
SFDEV-401V35-EGA4

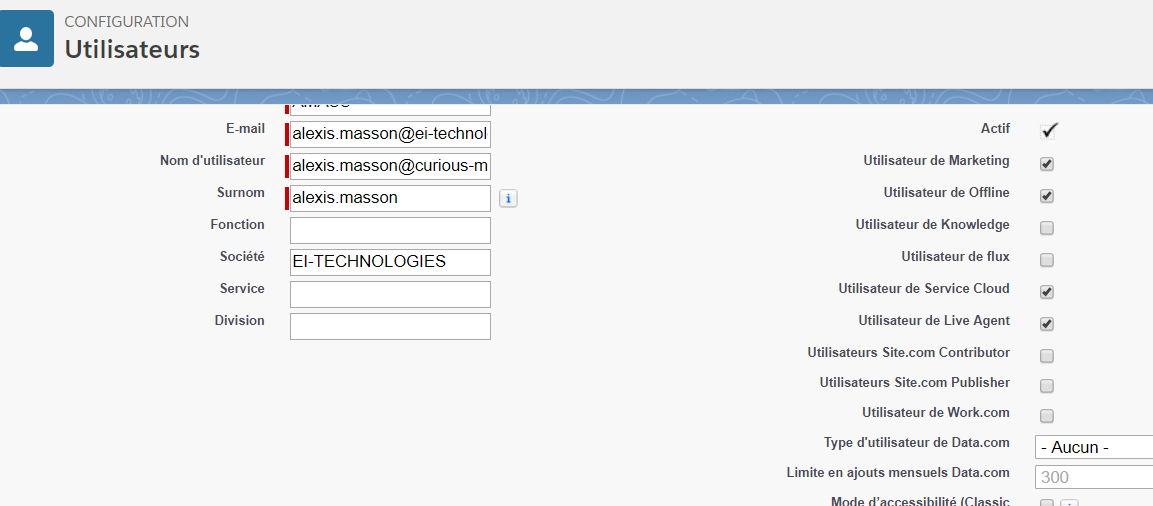
Student guide – DEV450 : Programmatic Development Using Apex and Visualforce  
SFDEV-450V3-SGA4

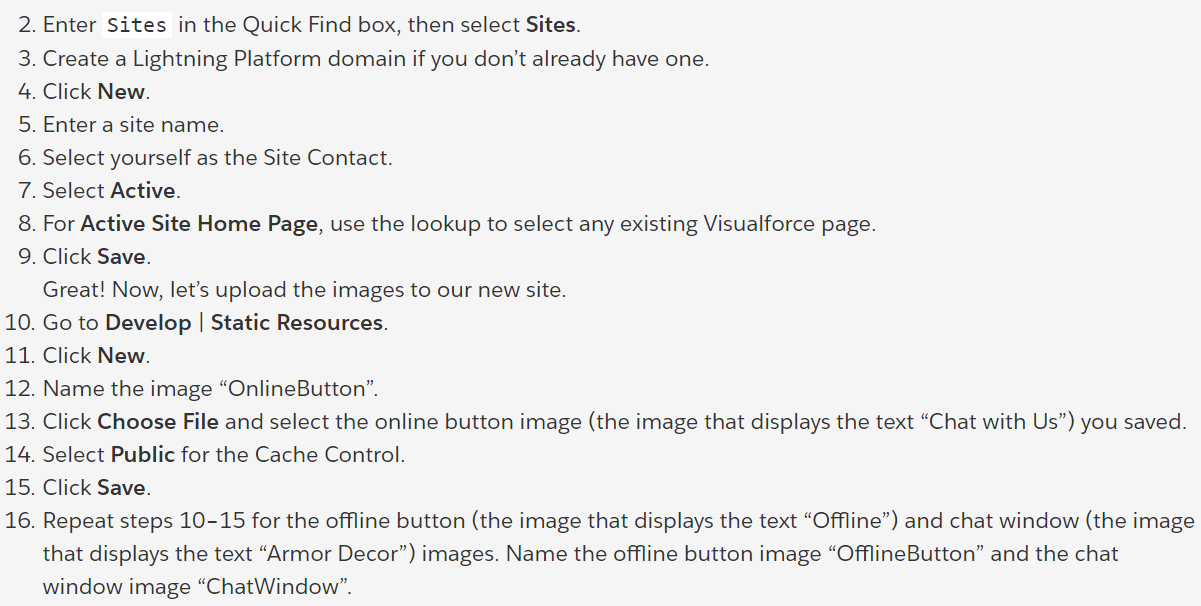
Exercice guide – DEV450 : Programmatic Development Using Apex and Visualforce  
SFDEV-450V3-EGA4

Live Agent Administration [exercice trailhead ici](https://trailhead.salesforce.com/fr/projects/build-a-community-with-knowledge-and-chat/steps/enable-live-agent-and-communities), [pdf ici](http://resources.docs.salesforce.com/212/17/en-us/sfdc/pdf/live_agent_administrator.pdf)

ATTENTION en fait ça commence [ici](https://trailhead.salesforce.com/fr/projects/build_branded_chat/steps/build_branded_chat_enable)

Capture Utilisateur (vue « modifier ») pour pouvoir cliquer « utilisateur de Live Agent »

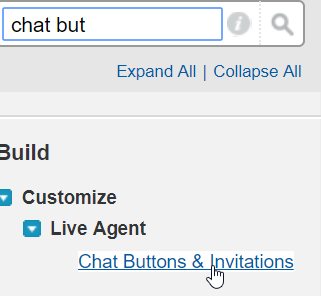
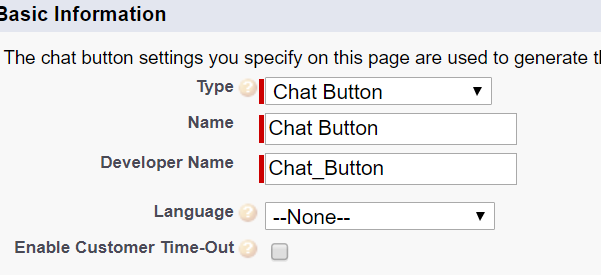


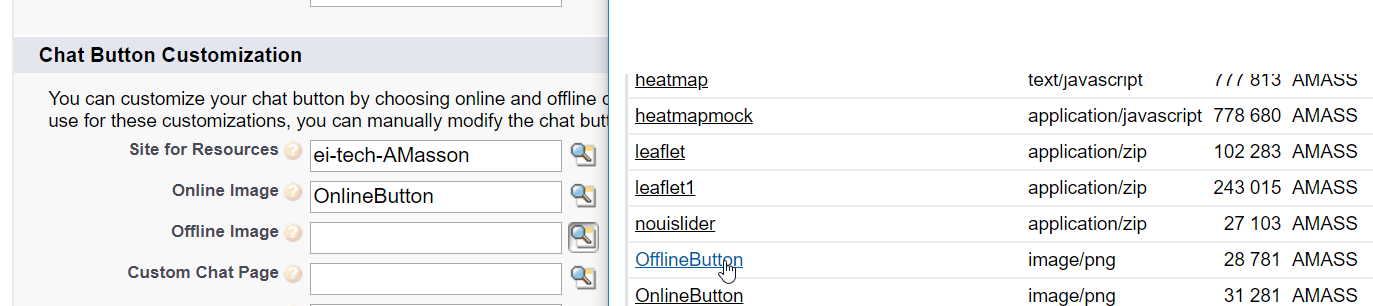
Création d’un site (quick view : site), possible en Salesforce Classic au moins.  


Ajout des ressources en quickview : Static Resource (étape 10) :

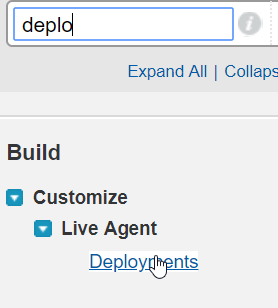


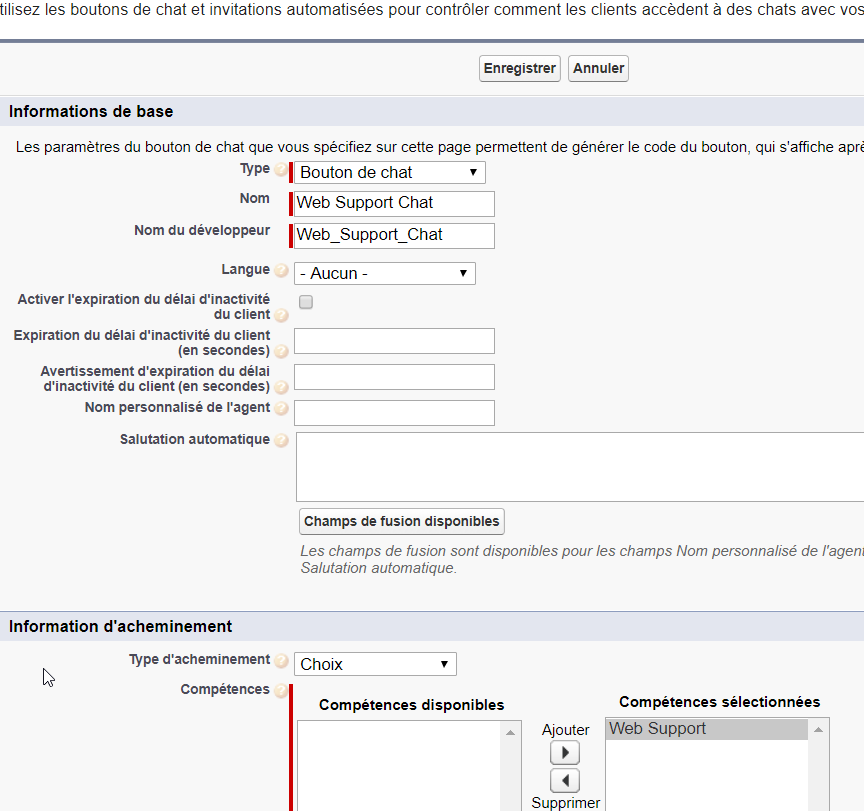
Chat button & invitations :

 puis :

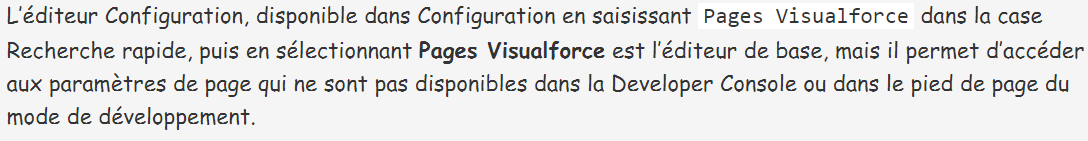


Ensuite déploiement :

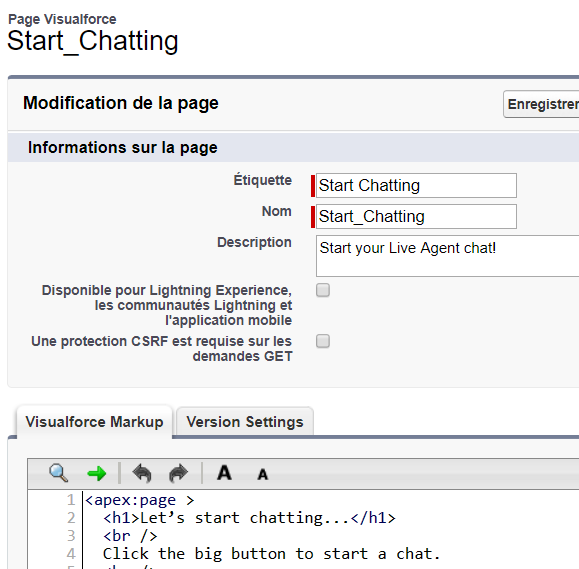




Pour modifier efficacement certains paramètres d’une page, il est plus intéressant de passer par la Recherche Rapide de Salesforce :

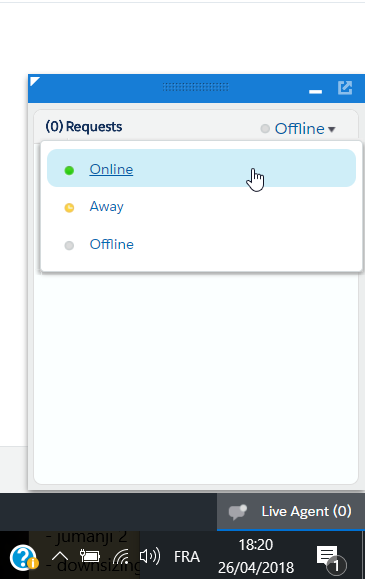


On accède ainsi à cette interface :



*A cette étape, soucis actuel pour moi,* [*voir ici*](https://developer.salesforce.com/forums/?id=9060G000000Xc4vQAC) *forum d’aide*

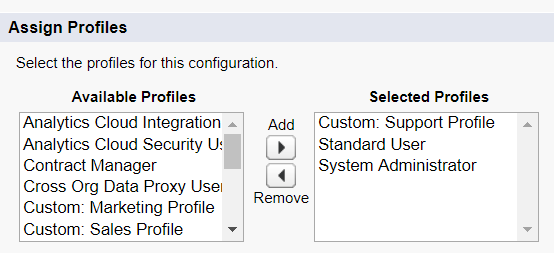
Live Agent : statut online du chat

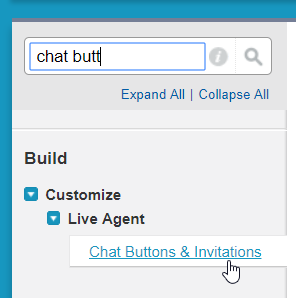


27 avril 2018

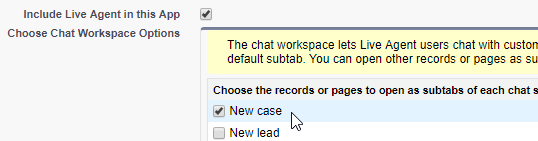
**Enable live agent & communities**Tout ça se fait sur l’interface Salesforce Classic !

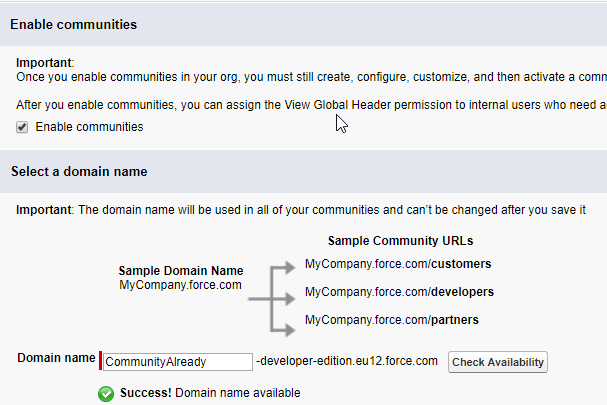
Etapes :  
  
1. Enable le live agent : live agent settings, enable, save.  
2. Assigner des live agent permission : rechercher Users, edit, live agent users, save  
3. Création d’une skill : skill, new, nom de la skill, ajouter des profils qui la possèdent, save.  
4. Création d’une config Live Agent : Live Agent Configurations, new, ajouter des profils qui la possèdent, save.  
5. Création du chat bouton : « chat buttons » dans la recherche, sélection de chat button & invitations, new, chat button, donner son nom, et dans Routing Information sélectionner la skill et l’ajouter, save.  
6. Création du live agent Deployment : Deployment dans la recherche, new, donner un nom, un titre  
7. Ajout de Live Agent au Sample Console : rechercher Apps, puis Edit a côté du Sample Console, on coche Include Live Agent in this App, on coche « new case » (cf capture d’écran), puis on coche visible dans Assign to Profiles, pour l’admin par exemple.  
8. Enable Communities : Communities settings, enable, on entre un nom, on save, click sur New Community, on sélectionne un template (ex : Customer Service), get started, nom, create !

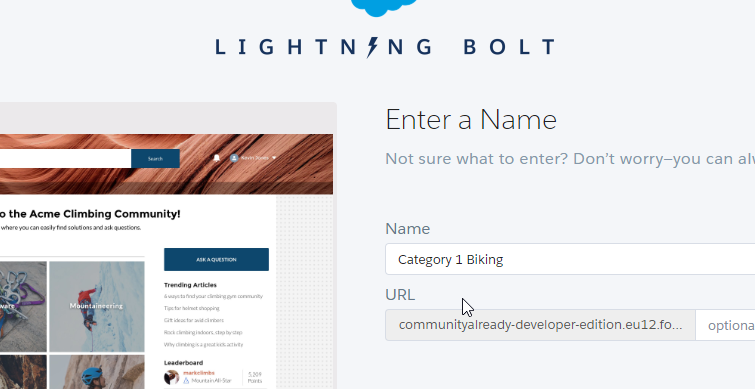
4. Capture :  
 

5. Capture :  
 

7. Capture :  

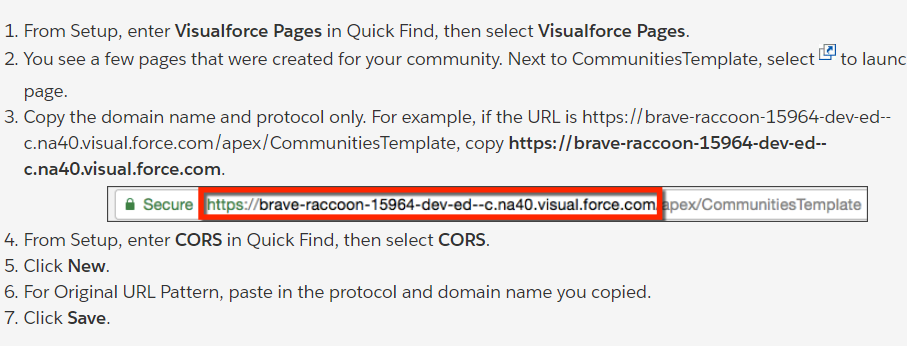

Puis :  
  
nb :  
“The chat workspace lets Live Agent users chat with customers, find existing records related to chats, and create new records based on incoming chats. Each new chat session opens as a primary tab in a console with the Details page as a default subtab. You can open other records or pages as subtabs. »

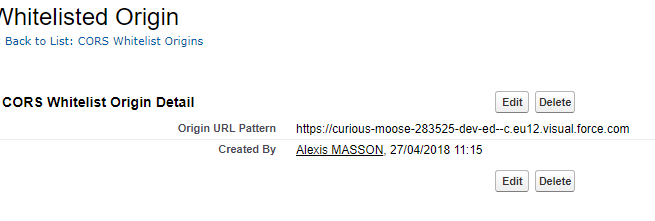
8. Capture :  


Nouvelle communauté !  


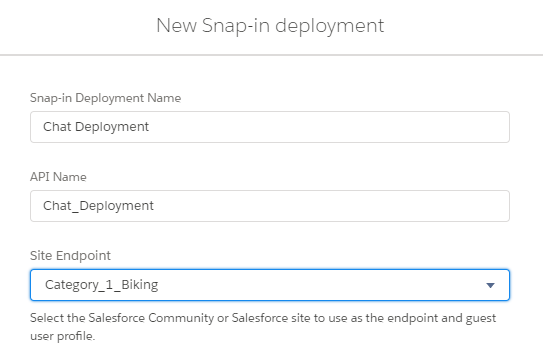
Ajout à la Whitelisted de la page du chat, via son l’URL :

On récupère l’URL d’une Visual Force Page, celle de template **CommunitiesTemplate** :

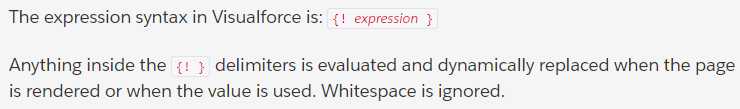


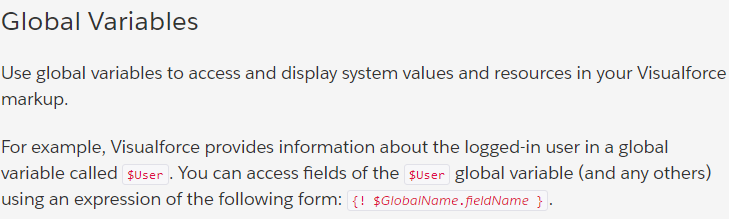


New Snap-in



Lamba expression – Visualforce expressions

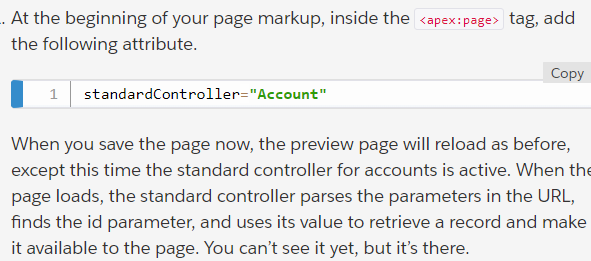




Voici la liste de toutes les variables globales de VisualForce : [clic](https://developer.salesforce.com/docs/atlas.en-us.212.0.pages.meta/pages/pages_variables_global.htm)

Module ici : [clic](https://trailhead.salesforce.com/modules/visualforce_fundamentals)

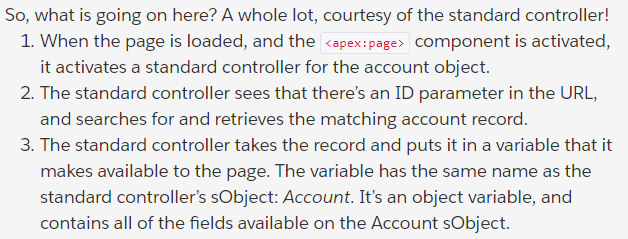
Ajout d’attribut dans le ApexPage :



Ensuite on peut passer en paramètre d’une requête :

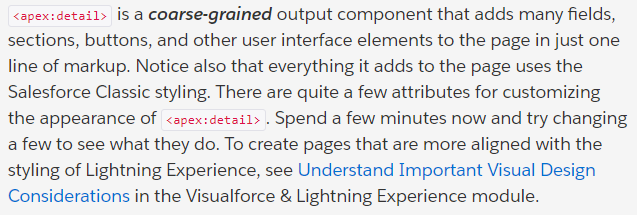
&id=0011r00001m36crAAA

Ou l’id est celle d’un object Account,  
Et la il y a interprétation par le navigateur et récupération de l’objet, pour pouvoir l’utiliser.

Explication par Trailhead :   


On peut utiliser <apex:detail /> qui rend directement toute la page d’un contact dont le paramètre est passé en &id=sonId dans le get de l’URL !!

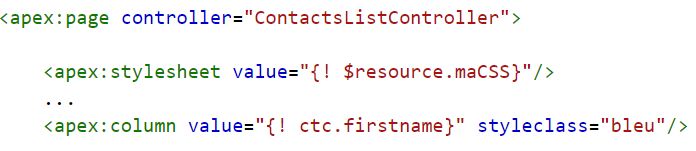
On peut aussi sélectionner des informations selon le besoin (passage en attribut dans la balise).

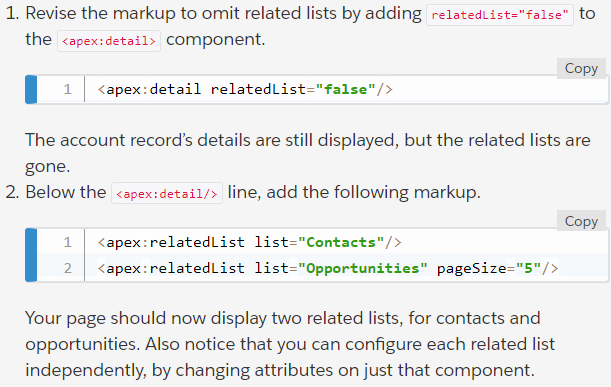


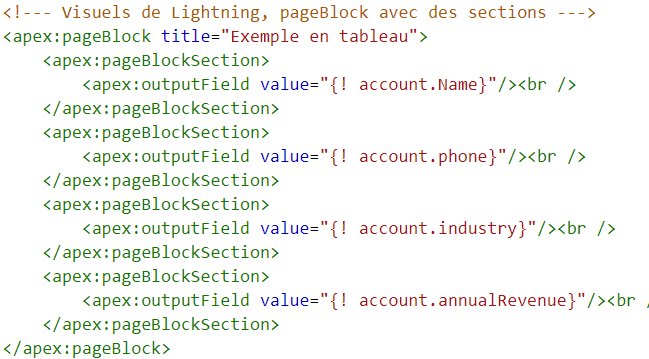
Ajout de CSS en VisualForce

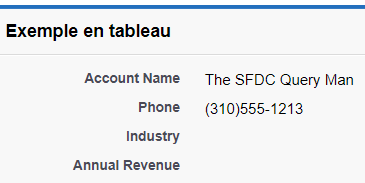




Ceci marche très bien par exemple :  


Attribut relatedList permet de montrer ou pas les objets liés :  
<apex:detail relatedList="false"/>  
  


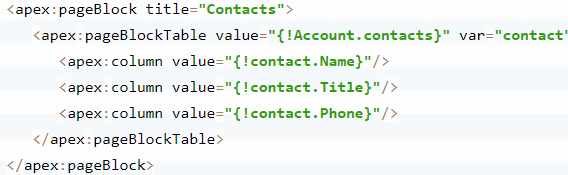


Résultat :   


Apex : outputField est adaptable automatiquement, son format sa place dans la page, c’est responsive sans qu’on intervienne pour caricaturer.

La table

Permet de choisir les colonnes

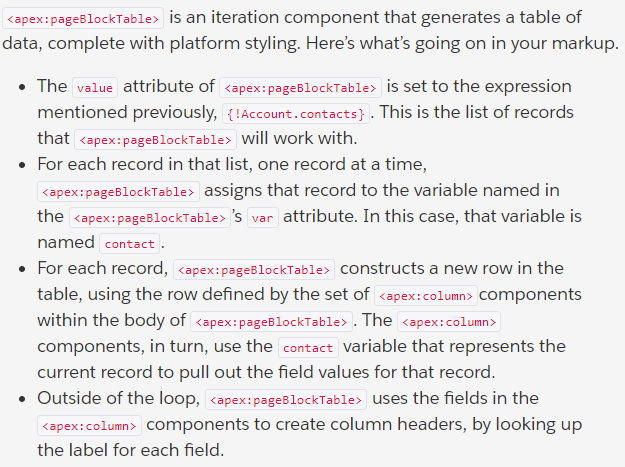


Données de test :  
Liste des accounts : <https://curious-moose-283525-dev-ed.lightning.force.com/lightning/o/Account/list?filterName=Recent>

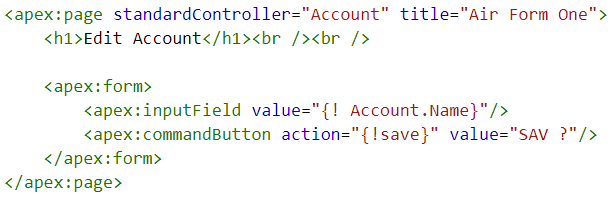
ID d’un account : ?&id=0011r00001oPW6KAAW  
ID account possédé par plusieurs contacts : ?&id=0011r00001m31ovAAA

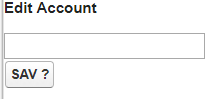
ID de Contacts : <https://curious-moose-283525-dev-ed.lightning.force.com/lightning/o/Contact/list?filterName=Recent>

ID du contact Carol Ruiz : ?&id=0031r000023g2UfAAI

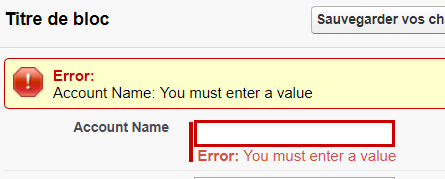


Un FORM simple :





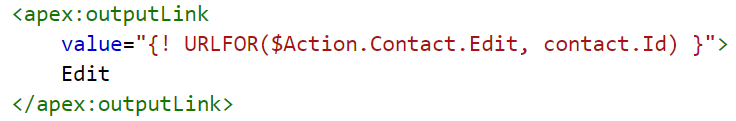
Ajout de messages d’erreur : 

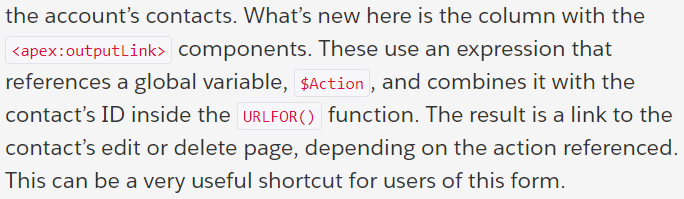
Effet, message récoltant toutes les erreurs en en tête :  


Rajout d’un bloc :



Il permet de récupérer les contacts liés au compte, d’ajouter des boutons d’edit et suppression

La boucle sur les contacts est faite par :  
  
  
Les boutons sont ajoutés par :  


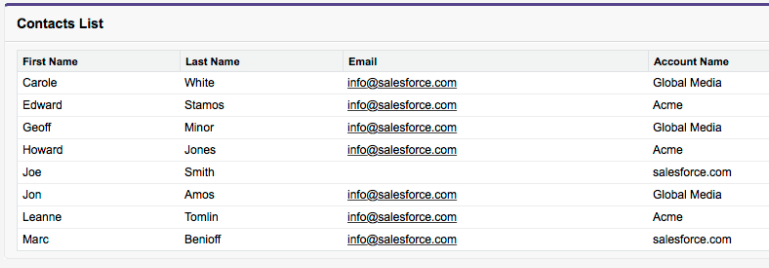
A ce propos :  


<apex :input… > sont utiles, à voir, lorsque le inputField n’est plus suffisant (champs perso etc)  
<apex :select …>pour des listes, radiobuttons etc.

Standard list controller

Listes de listes



Display la liste de nos contacts :  


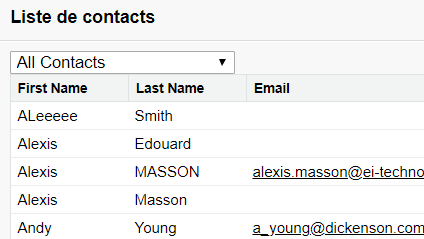
En gros, c’est une double boucle rien de fou !

On peut ajouter un id="contact"  
Cela donne au pageBlock un "name" qu'on va pouvoir utiliser pour du Ajax

Voici le bloc filtre ajouté (ci dessous) ;  
Quand on change le choix dans la liste, il faut aller voir dans actionSupport :   
event="onChange" qui rerender la partie dont l'attribut est reRender="idAttribut"



Il permet d'afficher un choix, et la liste (contacts List) s'actualise immédiatement !



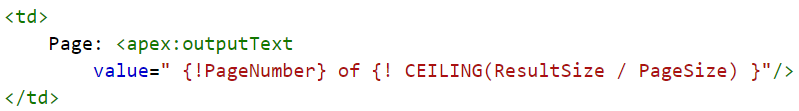
Le code complet est donc :



ReRender="ct2\_list" pour remettre à jour immédiatement (ajax)

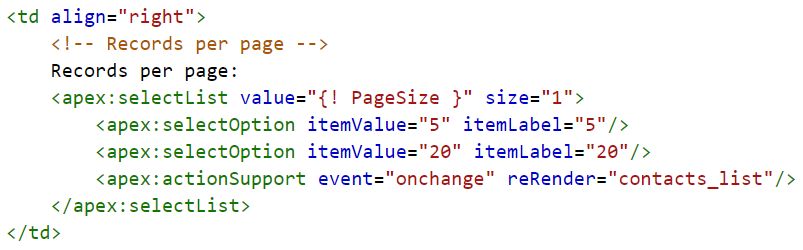
Ajax ne met à jour que la partie de la page dont l'attribut est ct2\_list !

**Pagination**

Page n sur M :  


Boutons PREVIOUS et NEXT  

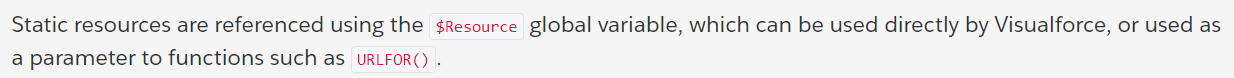

Choix d'affichage de combien par page

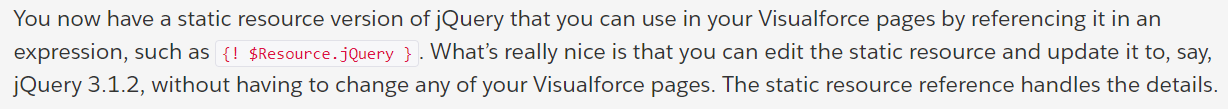


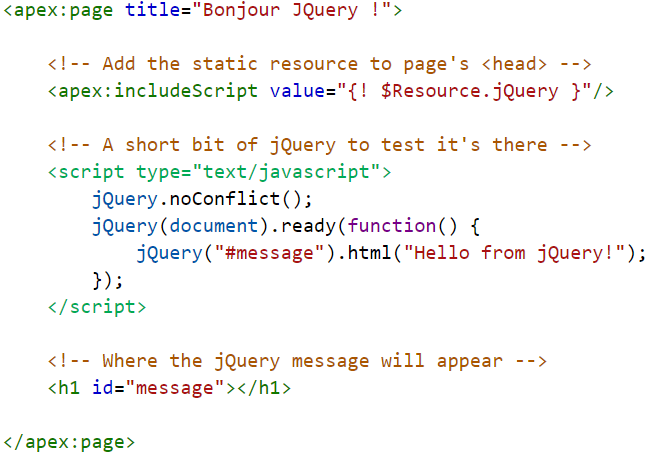
On peut boucler avec repeat, on repete sur value et on prend la variable a pour interroger dans la boucle.

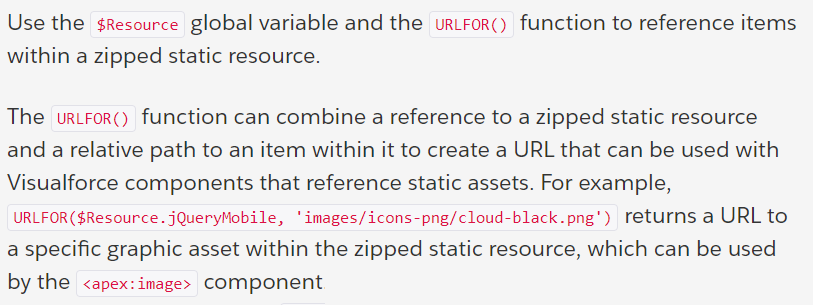


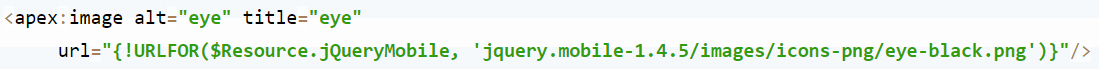
Static Resources



On quick find -> Static Resource -> New -> add + fill des bails, puis on peut accéder :  


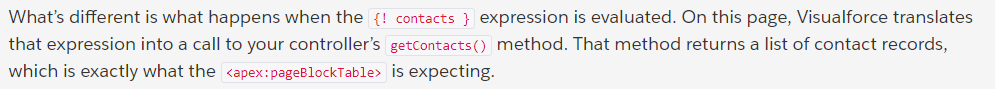
Premier script :  
  


Utilisation de URLFOR, explication :   


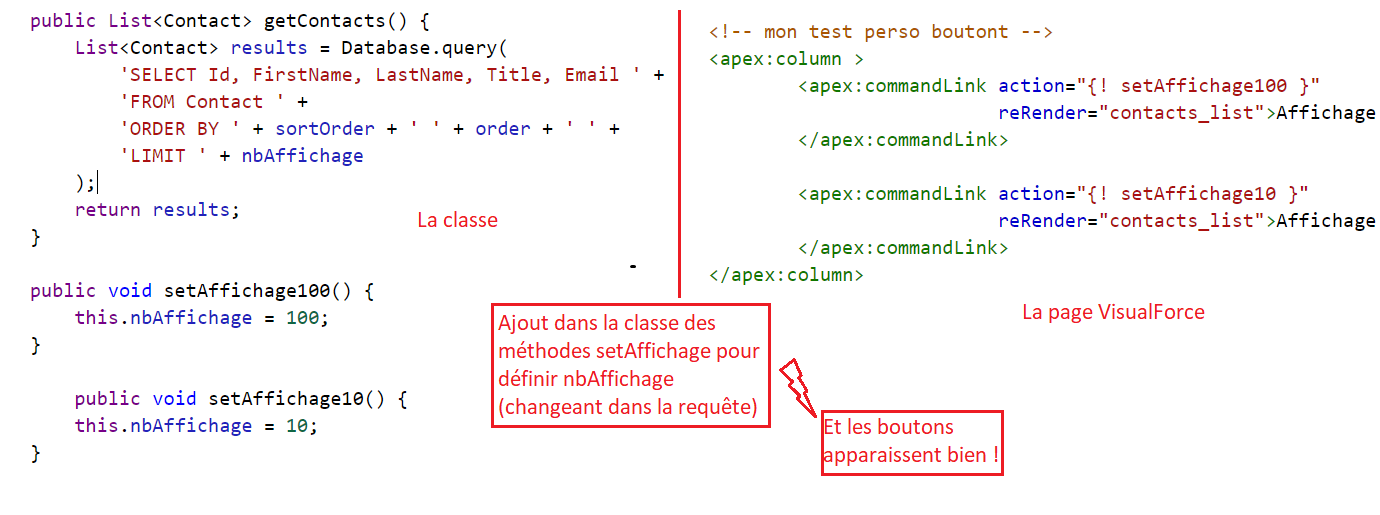
Iclure image :  


Custom Controllers

En gros, on met un controller="ControllerPerso", qui est une classe Apex qui contient un getContacts()  
Quand on touche aux contacts (dans le pageBlockTable), c'est ce getContacts qui est automatiquement appelé.



Voici par exemple des boutons que j'ai ajouté :



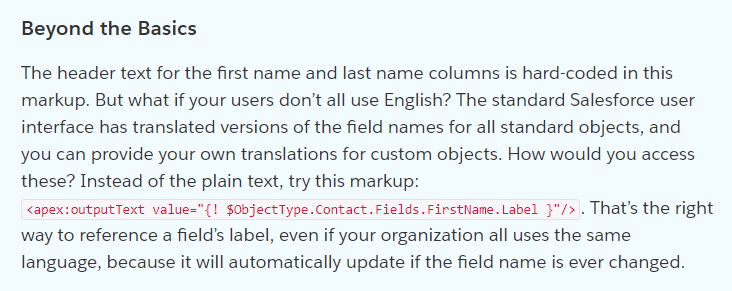
Du coup en fait, on a :  
<apex:commandLink action={! maMethode }  
 reRender="plagedontcestlID">NomBouton  
</apex:commandLink>

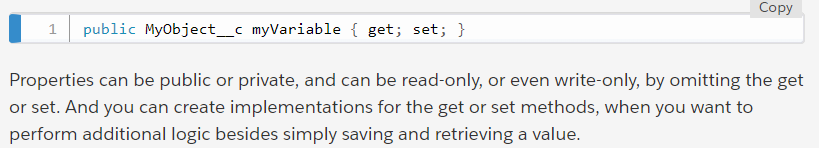
Voir :



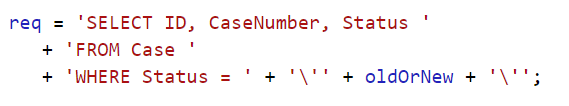
Il est moins verbeux de rajouter un attribut "value" associé à Last Name, histoire de pouvoir faire un commandLink autofermant.

Intéressant : au niveau du bouton, on peut le nommer non pas en dur, mais en adaptable à la langue de l'user.

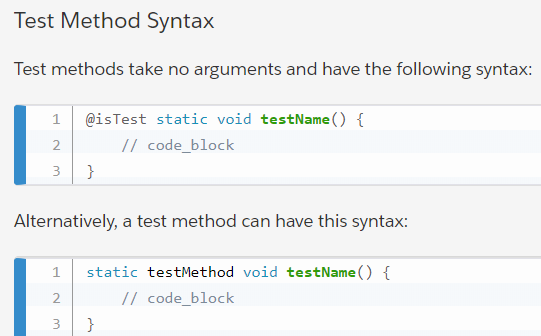


Properties  


Requêtes avec un WITH : il faut échapper un caractère 🡪



Tests : module [clic](https://trailhead.salesforce.com/modules/apex_testing/units/apex_testing_intro)

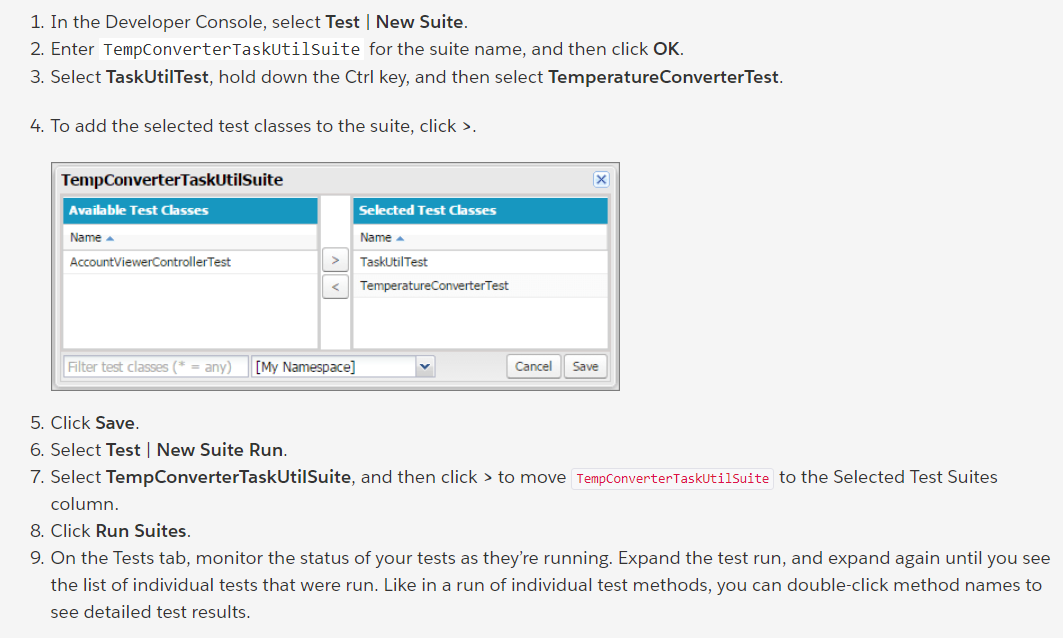


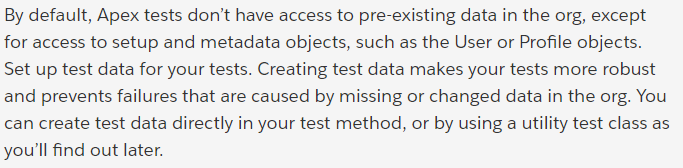
@isTest peut prendre des paramètres, mieux.

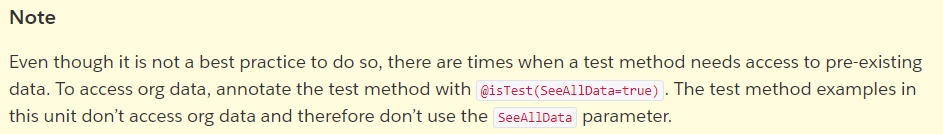
System.assertEquals(info1, info2) :  
The verifications are done by calling the System.assertEquals() method, which takes two parameters: the first is the expected value, and the second is the actual value.

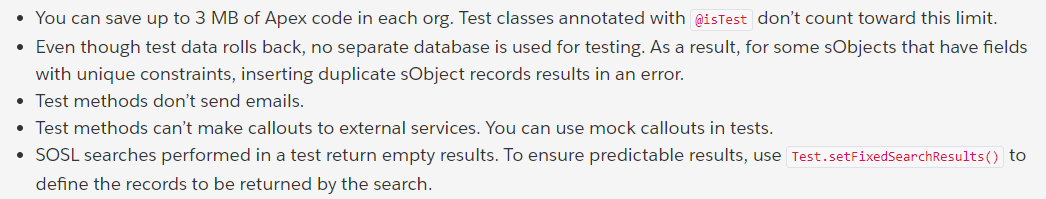
There is another version of this method that takes a third parameter—a string that describes the comparison being done, which is used in testBoilingPoint(). This optional string is logged if the assertion fails.



Méthode d'ajout d'une batterie de tests (test suite) :  


IMPORTANT :  
Les données créées lors de tests, ne sont pas ajoutées à la base !!  
Il y a un rollback à la fin.  
  
De plus, il faut créer des données de test. Les test ne peuvent utiliser les données préexistantes, histoire de ne pas faire de conneries.  
  


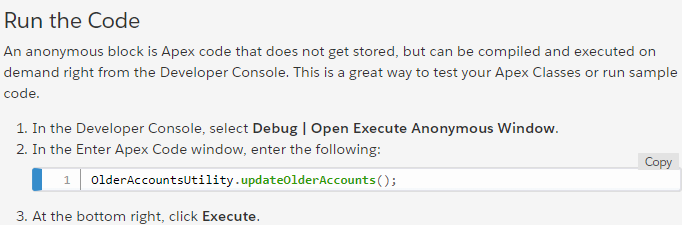
Cela dit, on peut forcer à l'utilisation des données :  




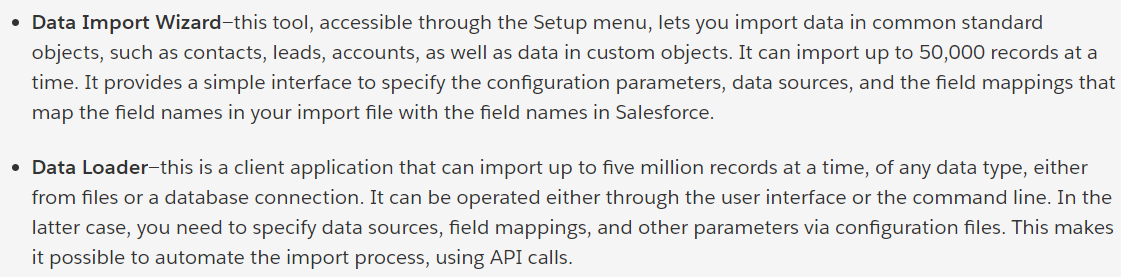
Attention donc si requêtes SOSL à bien relire le pavé ci-dessus.

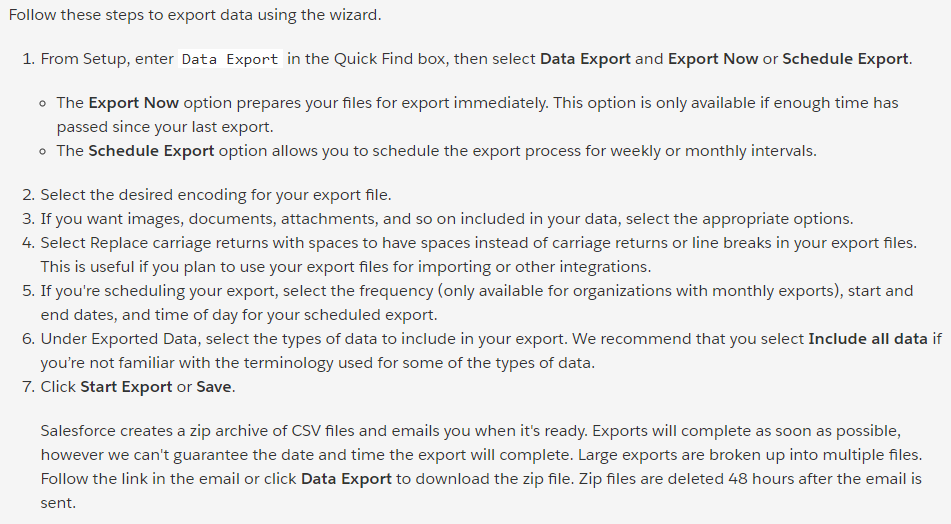
7 mai 2018

Lancer un code pour le tester : on est sur [ce module](https://trailhead.salesforce.com/projects/quickstart-apex/steps/quickstart-apex-3?trailmix_creator_id=005500000060WkUAAU&trailmix_id=diving-into-apex-development)

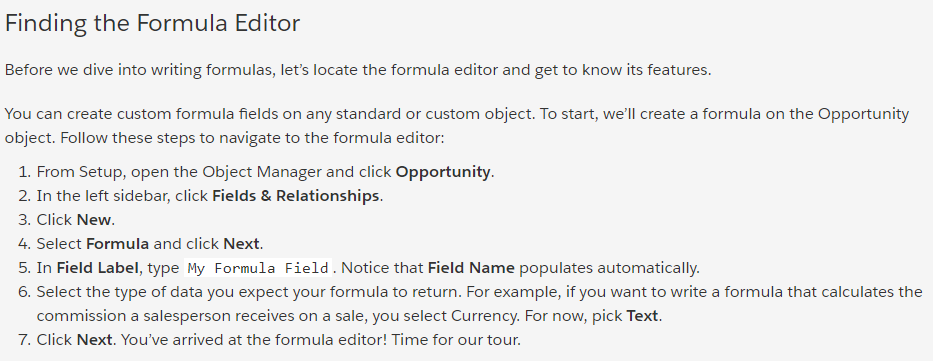


Data import Wizard & Data loader

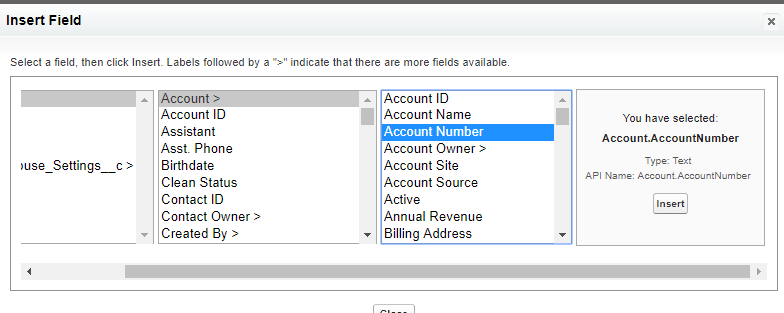


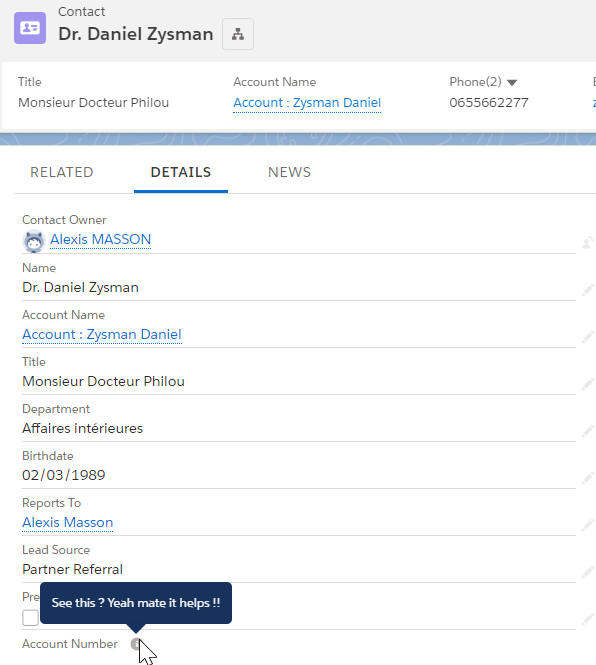
Export data :  


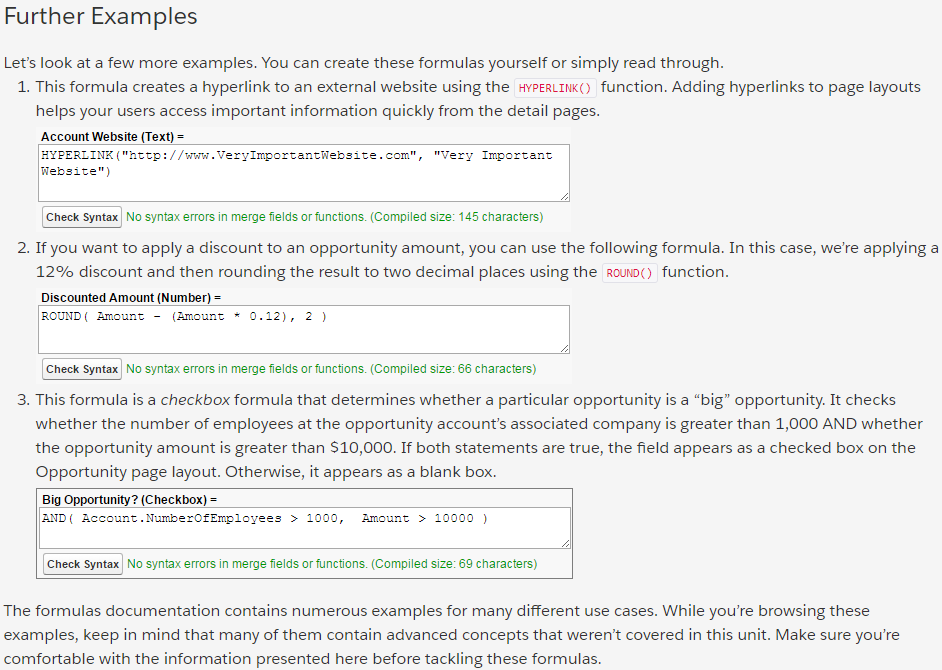
Formules – module [clic](https://trailhead.salesforce.com/trails/force_com_dev_beginner/modules/point_click_business_logic/units/formula_fields)



Retenir qu’on fait Setup 🡪 Object Manager 🡪 Contact 🡪 Fields & Relationship !

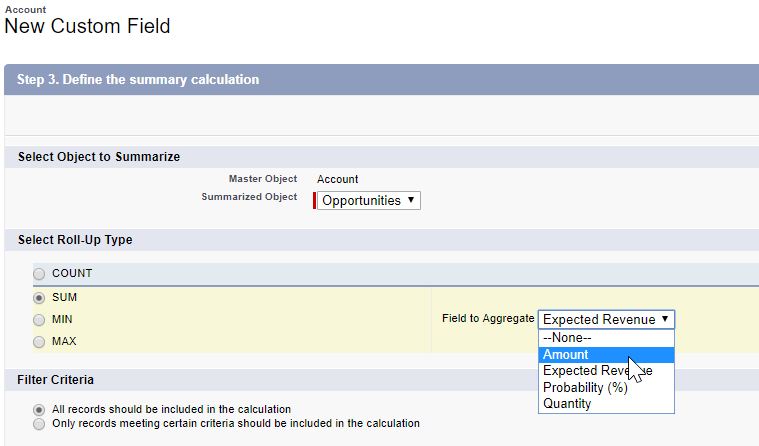


Résultat visible ici :  
 

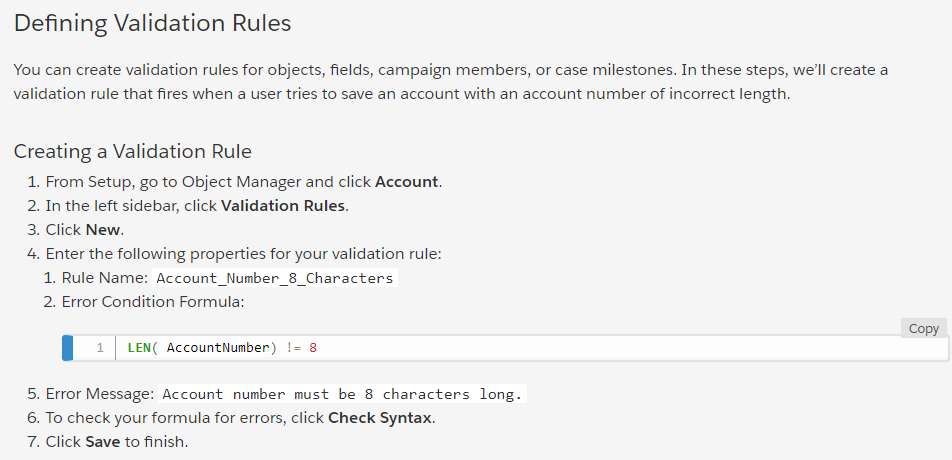
Quelques exemples de fonctions :  


Roll-up summary  

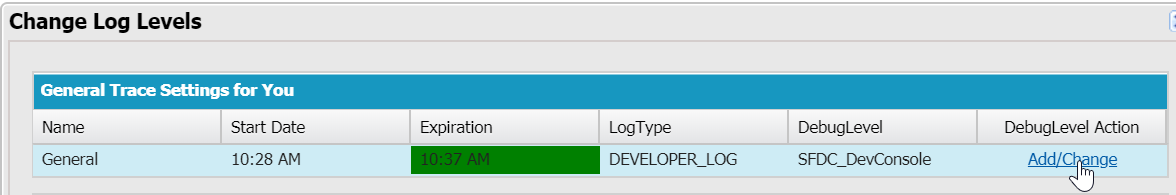

« A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list. »

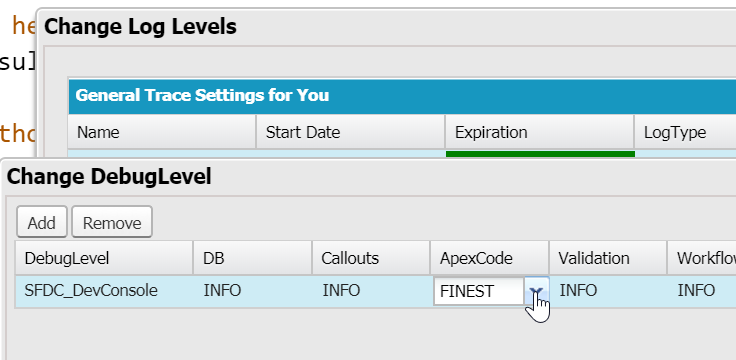


Validation rules – Validation de données

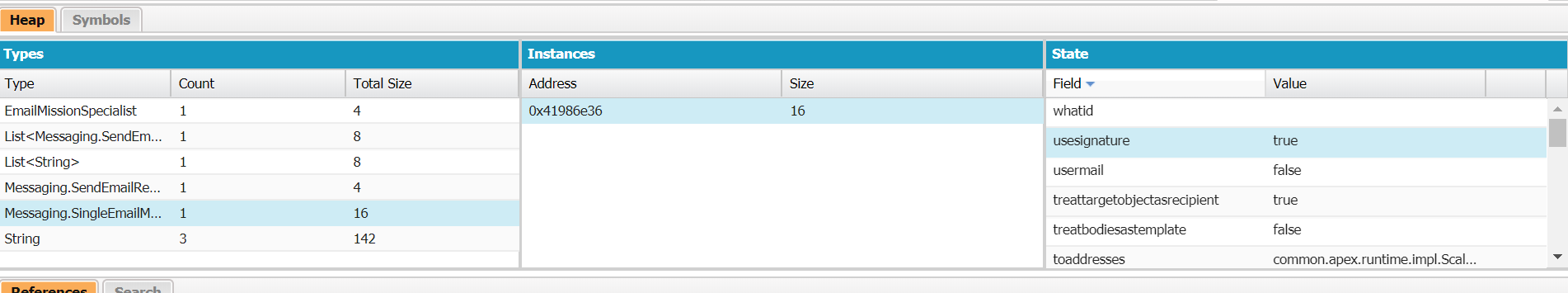


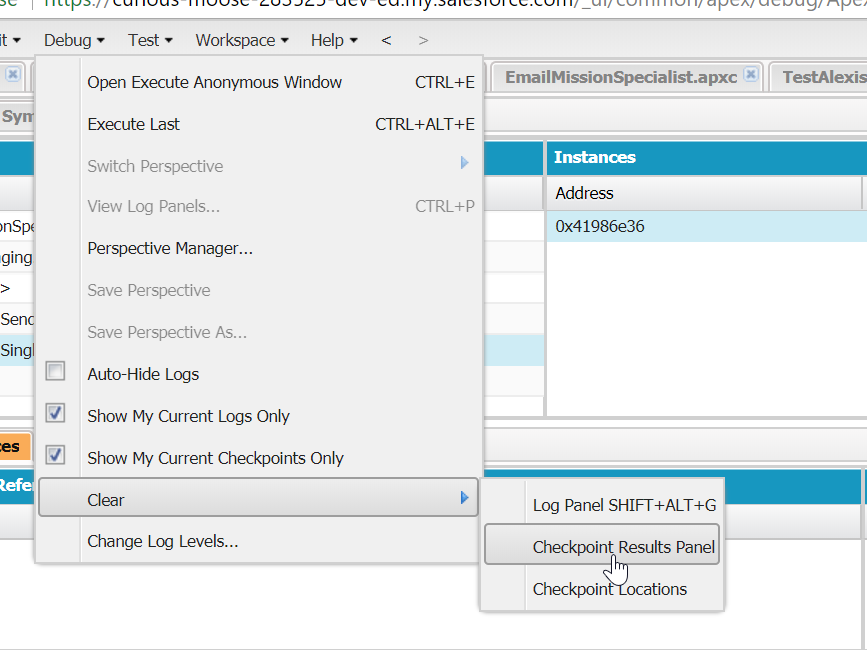
A propos des logs : module [ici](https://trailhead.salesforce.com/trails/force_com_dev_beginner/modules/developer_console/units/developer_console_logs)

Changement du level d'un log, clic Debug -> change Log Levels puis Add/Change  




Heap & Symbols tabs

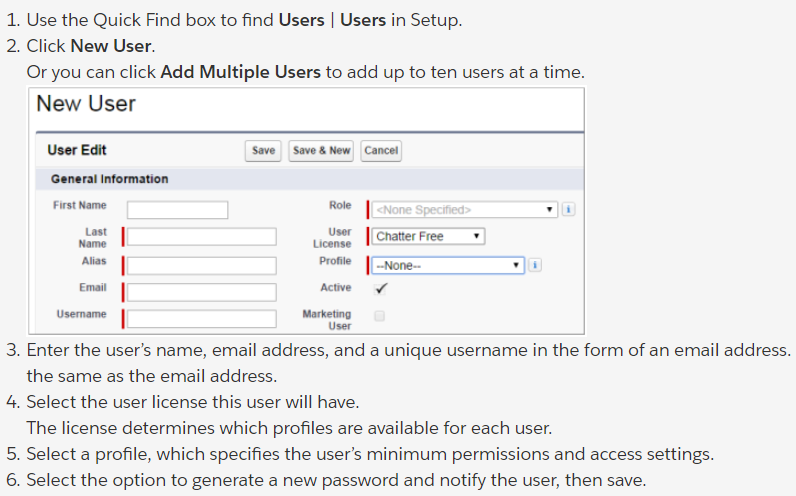


Effacer la console :  


Module sur la data security : [clic](https://trailhead.salesforce.com/trails/force_com_dev_beginner/modules/data_security/units/data_security_overview)

Petit quizz : [clic](https://quizlet.com/132187490/salesforce-app-builder-certification-practice-2-flash-cards/)

On va sur USER pour créer des profils & les paramétrer.



Trois levels de sécurité :  
- objects  
- fields  
- records

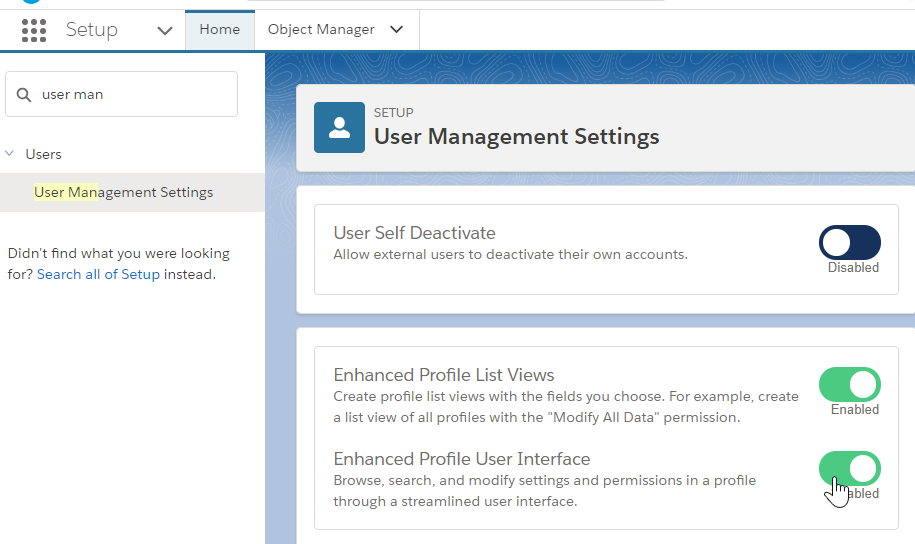
A profile is a collection of settings and permissions (paramètres et autorisations) 🡪 ce que l'user peut voir (les datas), et ce qu'il peut faire à propos de ce qu'il peut voir (datas).

Standard profiles : entre autre  
- read only  
- contract manager  
- marketing user  
- standard user  
- system administrator 🡪 view all datas, and modify all datas

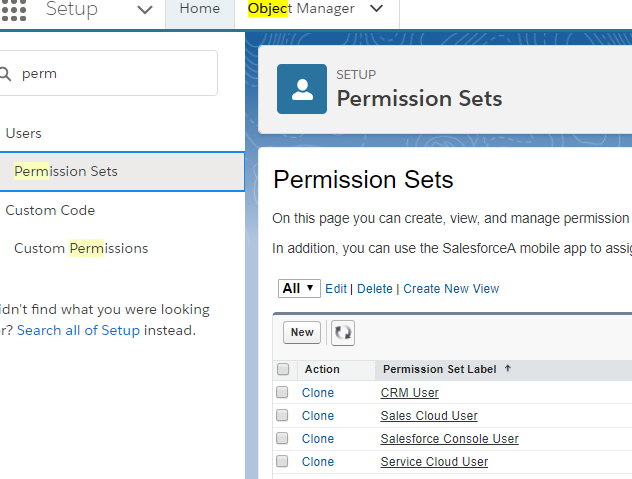
On peut cloner les profils pour les adapter (on ne peut modifier les permissions des profils de base).

Create a profile by cloning : quick find Setud 🡪 Enhanced Profile User Interface

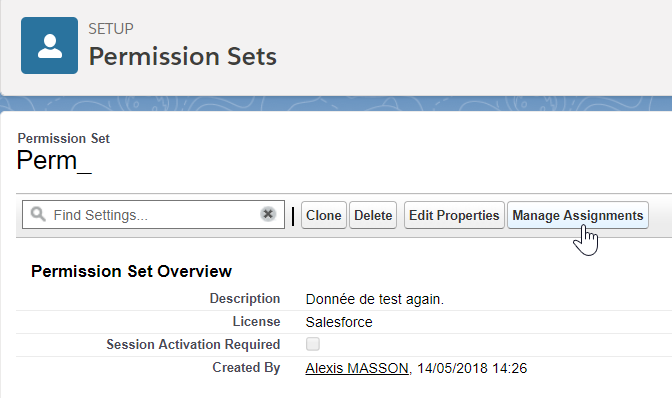
Ensuite dans Profiles 🡪 clone.

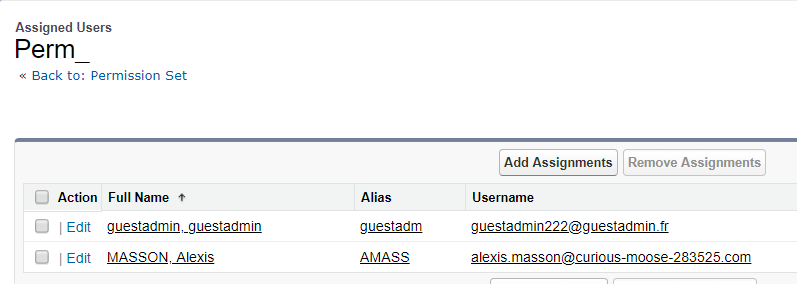


Quant aux **permissions** **:** Permission Sets



Création d'une donnée test (Permission Sets, New ) :  
on peut ensuite Manage Assignments

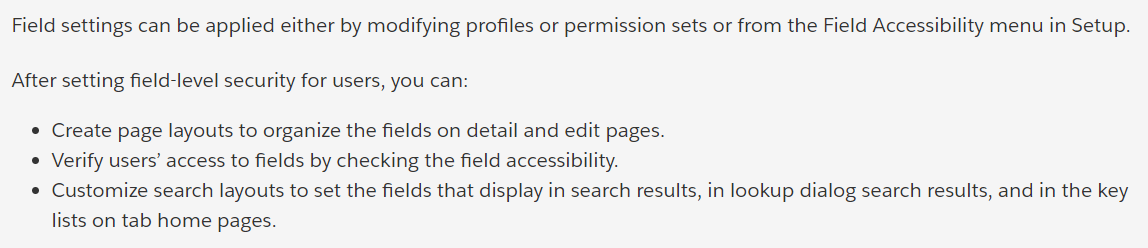


Et voila, on peut add des profils qui bénéficient de cette permission ! (ici guestadmin et MASSON)  


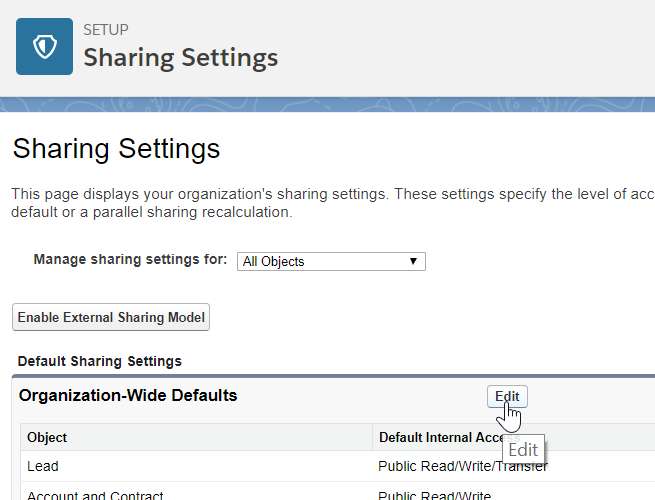
Field Accessibility

Permet de restreindre l'accès à un champs particulier : par exemple le SSN (Social Security Number) que tous n'ont pas besoin de connaitre, ou les salaires.

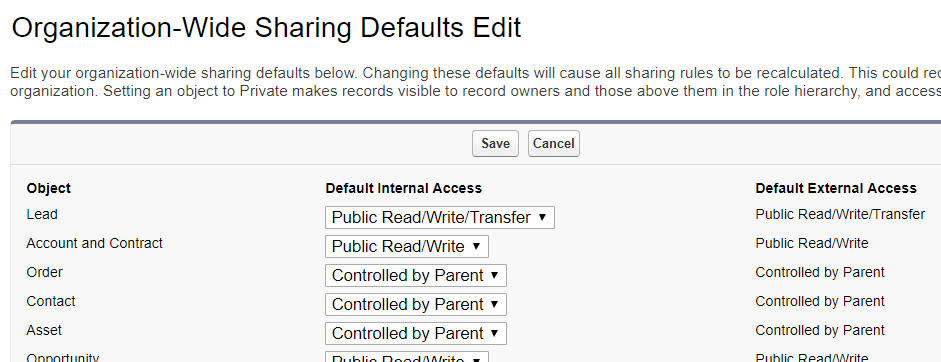
"Field settings can be applied either by modifying profiles or permission sets or from the Field Accessibility menu in Setup."



On peut **configurer les settings de Default Sharing Settings pour l'organisation (Organization-Wide Defaults)** en allant :  
Quick Find 🡪 Sharing Settings

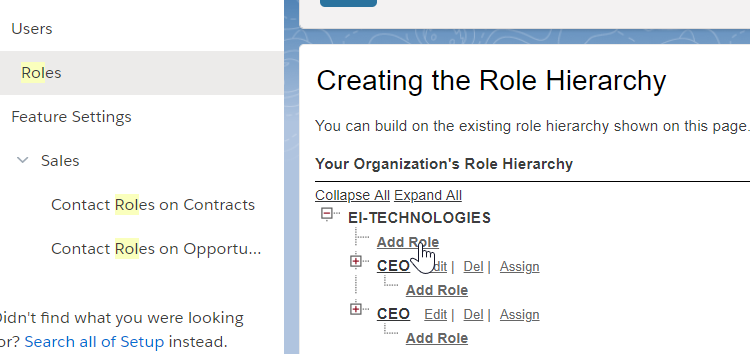


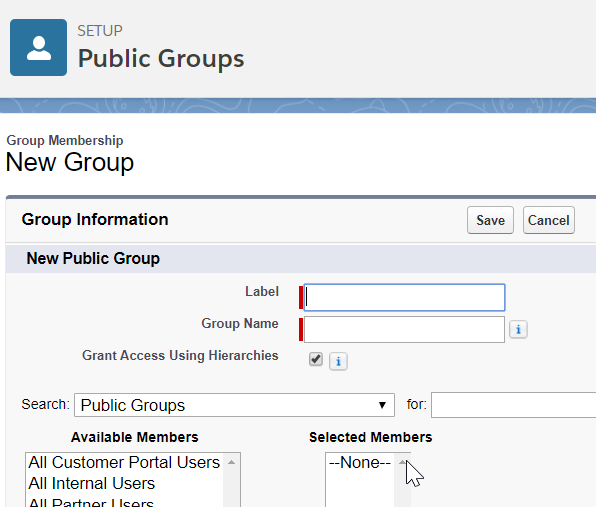
Edition :



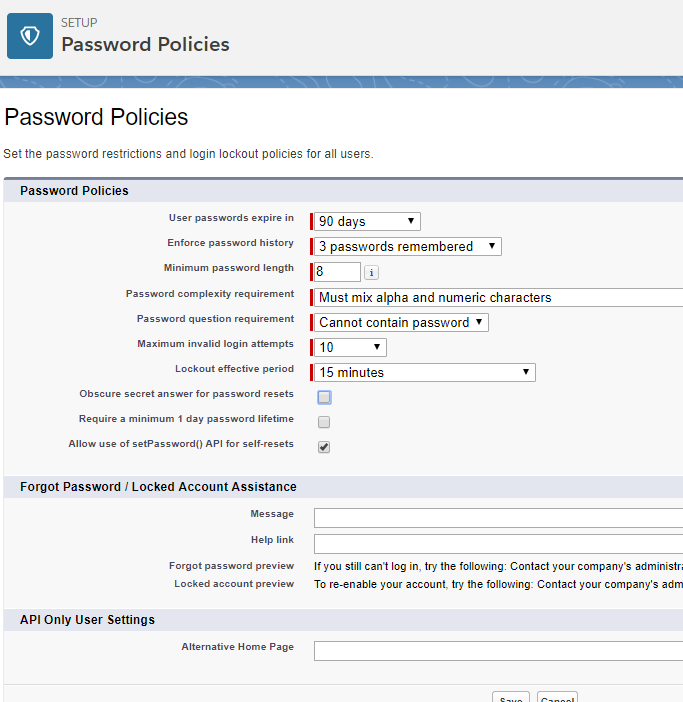
Définir un Role Hierarchy

Setup 🡪 Roles

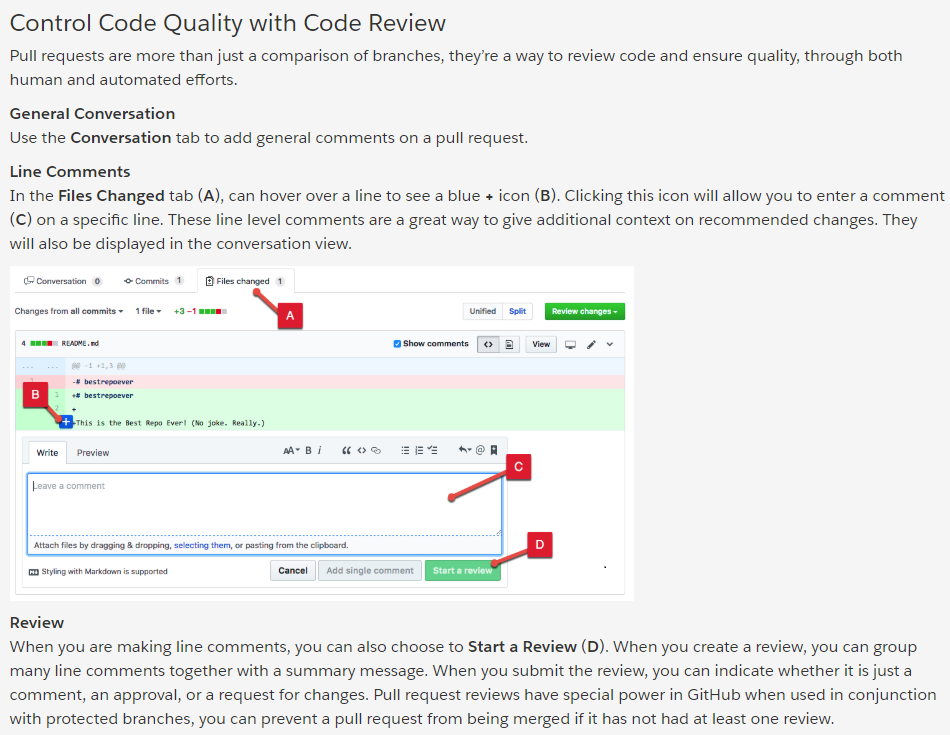


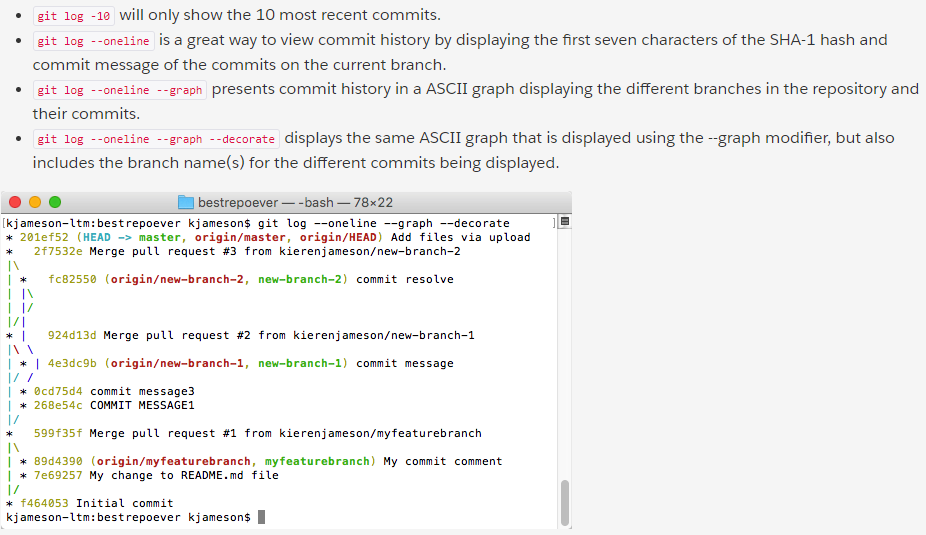
Public Group pour des règles générales à un groupe qui partage une même fonction :  


Password policies



Module Git bien fait : [clic](https://trailhead.salesforce.com/trails/sfdx_get_started/modules/git-and-git-hub-basics/units/work-with-teams-in-git-hub)





**FORMATION DEV 450**

Id d'un objet : deux codes, soit le code 15 soit le code 18 (nombre de caractères).  
Le 15 est case sensitive, le 18 ne l'est pas. On compare ce qui est comparable : on ne peut pas comparer du 15 avec du 18.

L'apex renvoie du 18 ! On peut enlever les 3 derniers caractères pour récupérer du 15.

[admin@dev450.804125.com](mailto:admin@dev450.804125.com)

mdp : salesforce1salesforce1

**MODULE 5 - SOQL**

Tous les objets sont requêtables (99%, sauf certains très rares).  
Exécuté directement par la plateforme, permet **uniquement de faire de la requête**. Sinon on utilise le DML.

C'est un langage **gouverné**, qui empêche de faire du SELECT \* par exemple : on doit préciser les champs que l'on veut récupérer.

100 requêtes en une fois, 300 sous requêtes au total (SELECT imbriqués), remontée possible sur 5 niveaux, descente sur 1 niveau, 50.000 résultats renvoyés.

Intérêt : on peut interroger avec des account.name etc.

SELECT id, CaseNumber, etc  
FROM  
WHERE  
LIMIT  
ORDER BY

L'id est forcément récupéré, de façon implicite même si non déclaré.  
On doit définir lequel on veut récupérer, donc il **faut** utiliser un ORDER BY.

Une requête SOQL renvoie :  
- sObject  
- List<sObject>  
- Integer  
- Aggregat Result  
- Liste d'agregat Result

Where CloseDate != LAST\_N\_DAYS:365

Il est recommandé de toujours utiliser des listes pour récupérer les résultats de requêtes.  
Bonne syntaxe aussi : List <Case> cList = new List <Case> ([requête SOQL])

Schema : permet de récupérer les métadatas, en map.  
Schema.sObjectType.Case.getRecordTypeInfosByName().get('Product Support').getRecordTypeId();

* Permet de récupérer des objets de manière dynamique, sans utiliser de requêtes.

On est en revanche limité sur Schema.

Cela permet un code **dynamique**.

**MODULE 6**

Relation custom : \_\_r  
Relation standard

On peut **descendre à 1** niveau en SOQL, mais on peut **remonter à 5**.  
Eviter les 1-n, ça monte vite.

Limité à 300 sous-requêtes dans une requête.

Si l'on veut faire des requêtes avec jointures (externes ou internes), on sous requête avec du IN + sous requête.

Le FROM sur une relation, permet la jointure.

**MODULE 7 – DML ESSENTIALS**

DML pour la persistence.

Standalone DML 🡪 Si une erreur, terminé avec renvoie d'exception.  
Database.method(sObject List, false) 🡪 traitement partiel, AllorNone passé à faux.

Avec SalesForce les relations se font avec l'ID, il est donc important de savoir le manipuler.

**DEV 450 - Troisième journée**

Résumé de la deuxième journée : before update, after update, before undelete (mais pas le after), before insert, after insert,

Traverser du fils vers le père : si c'est un custom champs (relation), on peut remonter en prenant nomChamps\_\_r (r pour relation), on peut ainsi remonter de 5 niveau.

Pour descendre, on fait une sous-requête dans la clause SELECT ; on la fait sur la relation (nomObjetFils) au pluriel.

On peut descendre puis remonter (toujours 5 objets max pour la remontada).

Outer Join dans la clause WHERE pour vérifier que l'ID n'est pas présent (est présent, pour inner join).

On peut récupérer 50.000 éléments, on peut faire 300 sous requêtes, on peut faire 100 requêtes.

Requêtes dynamiques dans la version sans crochets (database.method()) de la requête.

Avant d'insérer, on doit vérifier :  
- la cohérence des données  
- l'existence de données  
- la non nullité des données (null = pas bon)

150 requêtes max dans une transaction, 10.000 max d'éléments.

Niveau trigger :  
On ne peut pas faire un insert ET un delete en même temps (même sur différents objets)

Trigger est bulkifié par défaut. En insérant 302 éléments, il y aura un pack de 200, puis un autre de 102. C'est différent d'une transaction

PROF : Salesforce est une base Oracle, qui est commune pour tous.  
Salesforce a mis tous les records de tous les objets de tous les clients, dans UNE table Oracle (en vrai une 15aine, mais c'est l'idée).

Salesforce garanti un temps de réaction à moins de 300 ms !  
Comment gère t-elle ?

Avec l'ID de l'instance, l'ID de Company Profile.  
Chaque objet a un id, que l'on retrouve dans l'URL : a03 pour un objet custom.  
On peut faire jusqu'à 800 champs pour un objet. Un record fait 2 ko quoiqu'il arrive.

Personal account : compte + contact, qui fait donc 4 ko.  
*Taches : fait partie de Activity, séparé en tache & événement (hs).*

Il y a une autre table : la table des métadatas !  
Organisée sensiblement de la même façon, et enregistrant notamment les types.

Les champs formules sont interprétés à l'affichage.

Les trigger se déclenchent sur la table de datas. La formule n'ayant pas changée pour le jour, le trigger n'a pas été déclanché !

**Prof : si on fait un UPDATE, ça fait :**

On fait un UPDATE (IHM), voici les étapes dans l'ordre :

Etapes :

1. D'abord, chargement du **old**

2. ...puis du **new**.

3. Déclanchement des validations système !

4. Déclanchement des règles de validation

5. Trigger : beforeUpdate, si plusieurs ils se déclanchent ... On ne sait pas quand. DONC BONNE PRATIQUE pas plusieurs trigger sur le même événement.

6. Validation systeme + Validation Rules

7. En base, on fait des calculs puis retourne l'id (permet la manipulation immédiate).

... autres règles que le prof nous épargne, puis on passe au

9. Tous les after update

... autres règles que le prof nous épargne, puis on passe au

11. Evaluation des workflow : il évalue la formule, et si elle est vraie il dit "on va faire des actions plus tard sur cela), ce qu'il fait en 12 (ordre qu'on ne connait pas)

12. Fait les actions (qui sont soit envoie d'un email, envoie d'un message SOAP, fieldupdate ou création d'un évé)

13. si Field Update (FU) 3+5+7+9

14. PB

15. Actions sur PB

16. si Field Update 3+5+7+9

... autres règles que le prof nous épargne, puis on passe au

18. Rollup

Si rollup, ça remet le bordel

CA c'est à maitriser. On fait des classes de test, pour gérer ça.  
On utilise des APEX uniquement si besoin.

La transaction commence au SAVE  
L'APEX Transaction commence étape 5 (All Before Update).

List, map, set : [clic](https://beginnersbook.com/2015/01/difference-between-list-set-and-map-in-java/)

**DEV 450 - Quatrième journée**

Résumé de la journée précédente

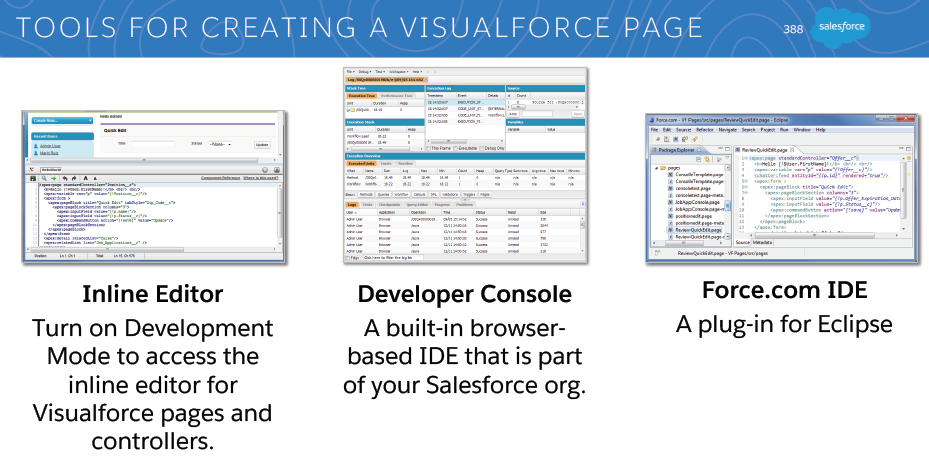
Tests : simuler ce que fait l'utilisateur, créer des méthodes privées, @TestSetup qui permet de lancer un truc qui se lance avant chaque méthode (déclarée par @isTest), startTest et StopTest pour réinitialiser les compteurs (remettre dans les conditions utilisateur).

On ne travaille pas sur la vrai base, rollback à la fin ; mauvaise méthode d'utiliser le paramètre de @isTest qui récupère les données de la base.

Bonnes pratiques : jamais de DML SOQL dans boucles ou fonctions réccursives, mettre des clauses WHERE pour récupérer uniquement ce dont on a besoin, utiliser les relations pour remonter les descendences et pas doubler les requêtes, FOR loop pour éviter de tuer la heap size, et dans un soucis de performance faire des clauses WHERE sur les champs indexés.  
Les trigger : on ne crée pas plusieurs trigger sur un événement, car on n'en maitrise jamais l'ordre. Corps du traitement trigger dans une classe (pour lisibilité et rendre factorisable un traitement).  
Les classes ne font que du traitement (pas obligatoire mais préférable, pour les triggers).  
On ne prend en paramètre que des listes car les triggers sont bulkifiés.

Penser aussi à vérifier qu'une appli n'est pas déjà développée dans le salesforce store répondant à notre besoin, avant d'envisager d'en créer une par nous-mêmes.

Création d'une page Visual Foce



Professeur Abdel-Nor conseille la version Inline.

Adresse URL de l'édition : c.notreInstance.visualforce.com (c pour custom), car les serveurs d'hébergement de Salesforce ne sont pas forcément les mêmes.

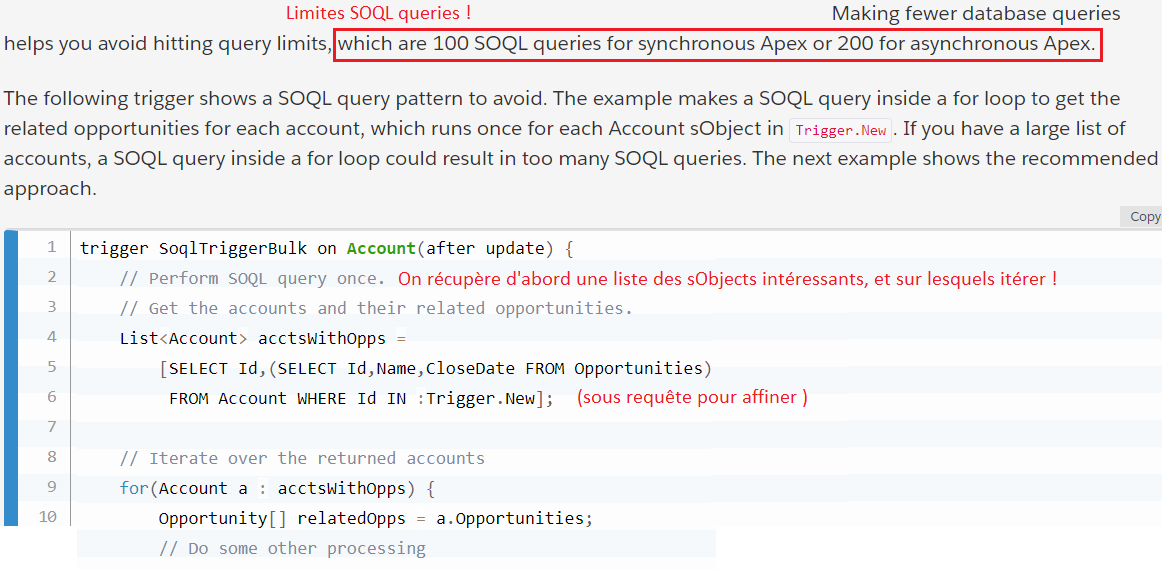
Cocher développeur mode, puis mettre dans l'URL /apex/notreUrl (à voir en exercice)

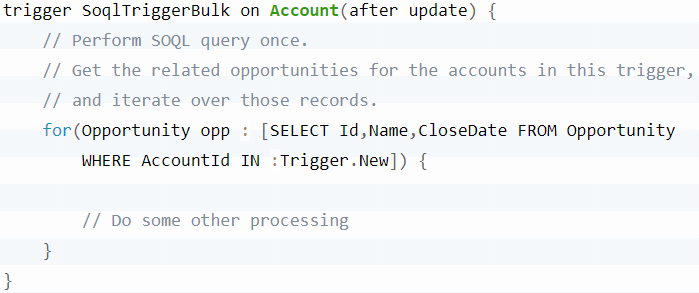
TRIGGERS

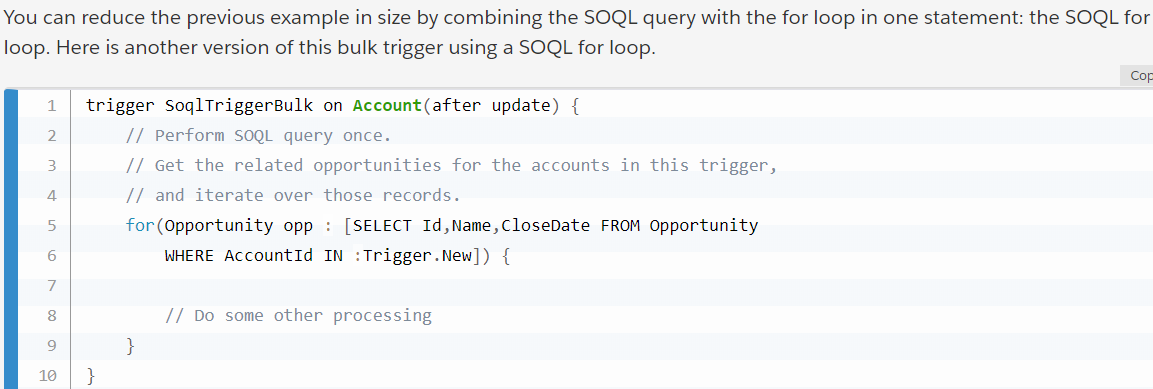
Bonne pratique : faire opérer sur un record set plutôt que sur un record, de plus bonne pratique : toujours partir de l'idée qu'on bosse sur un set de données, plutôt que sur une seule (marche toujours, et plus efficace).

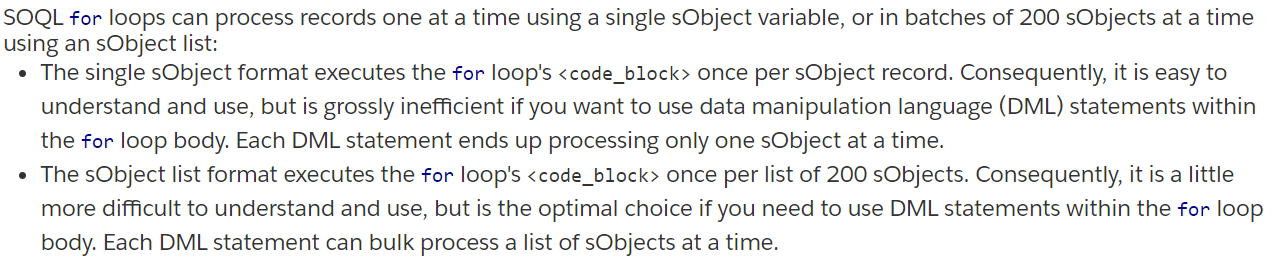
Faire le moins de requêtes SOQL (limites : 100 SOQL en synchrone, et 200 en asynchrone).  
*Making fewer database queries helps you avoid hitting query limits, which are 100 SOQL queries for synchronous Apex or 200 for asynchronous Apex.*

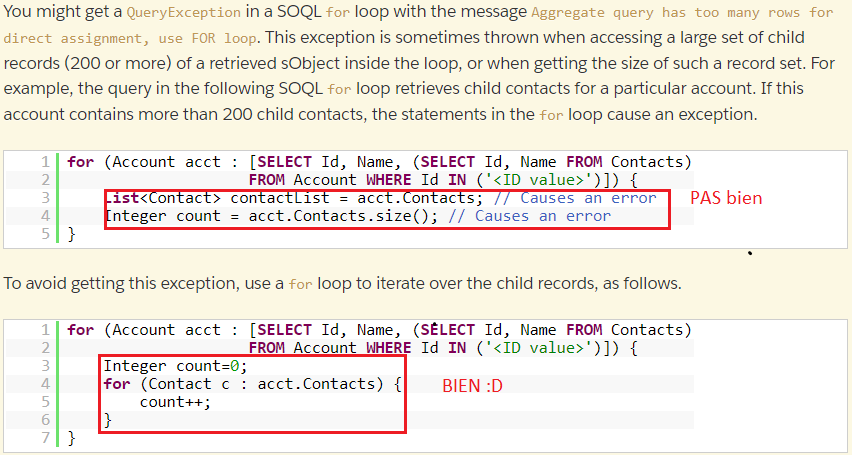
Par exemple ne **jamais** mettre de requête SOQL dans une boucle.



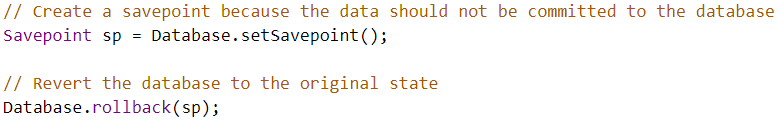
En condensé :  




Niveau bonnes pratiques ([clic](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_loops_for_SOQL.htm))  




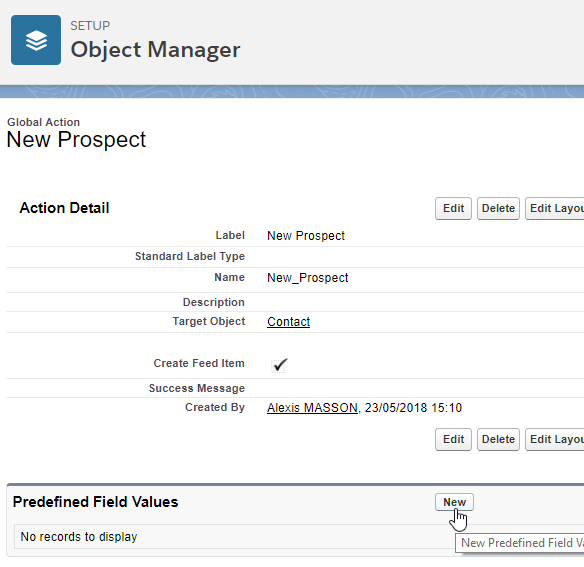
Savepoint



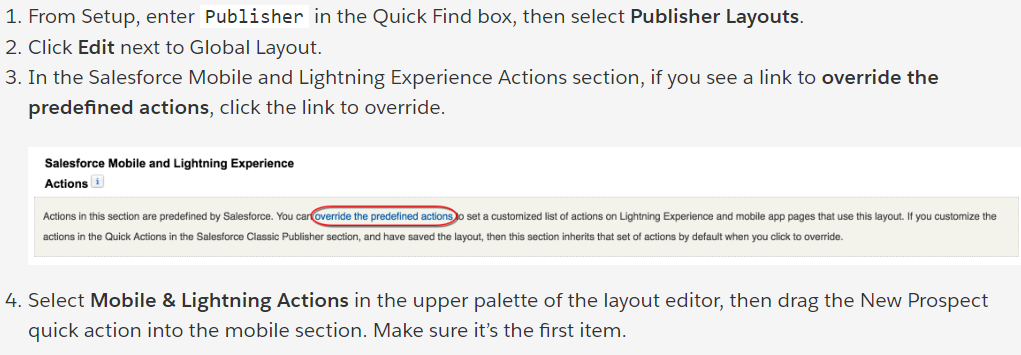
BONNE PRATIQUE SOQL :  
The Apex runtime allows up to 150 DML calls in one transaction.

Rollout : développement (application)

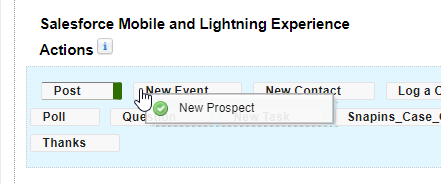
Ajouter un champs et lui donner une valeur prédéfinie :



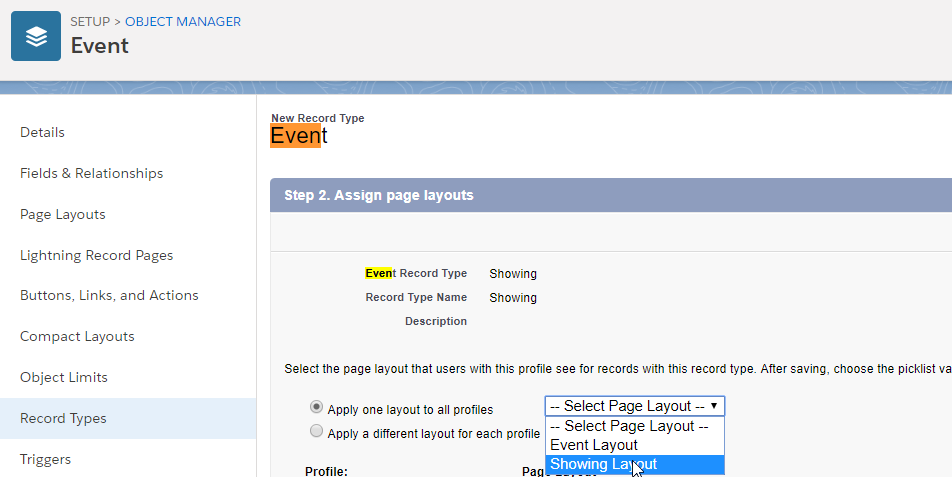
**Add an action to the Global Publisher Layout**



Global Actions, publisher Layout 🡪 on fait glisser le Prospect (qu'on vient de créer) pour l'ajouter au Mobile & Lighthing Xperience Actions

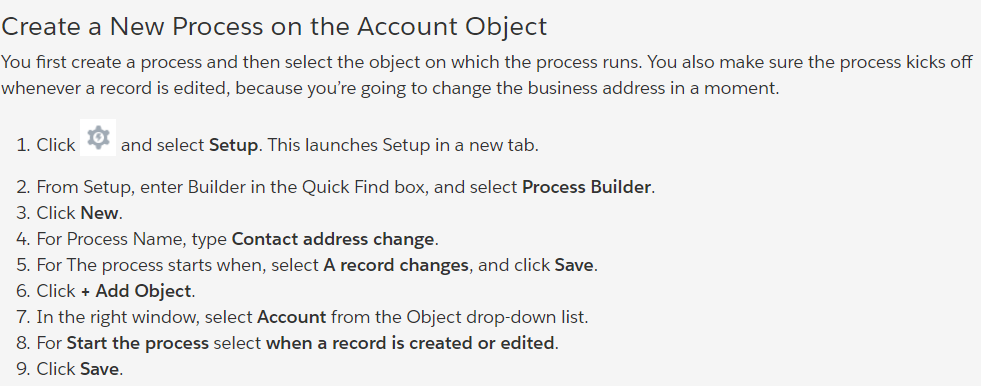


Ajout d'une merde au layout event je sais pas quoi



Clic ici pour le [tuto trailhead](https://trailhead.salesforce.com/trails/force_com_dev_beginner/modules/salesforce1_mobile_app/units/salesforce1_mobile_app_actions_objectspecific)

PROCESS BUILDER

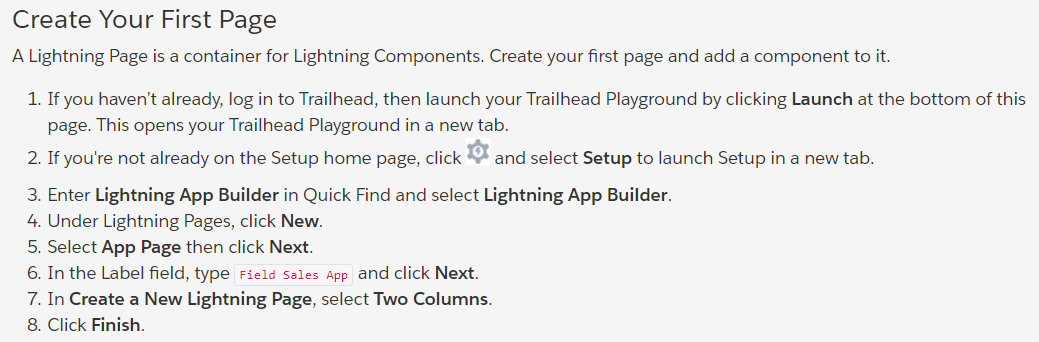


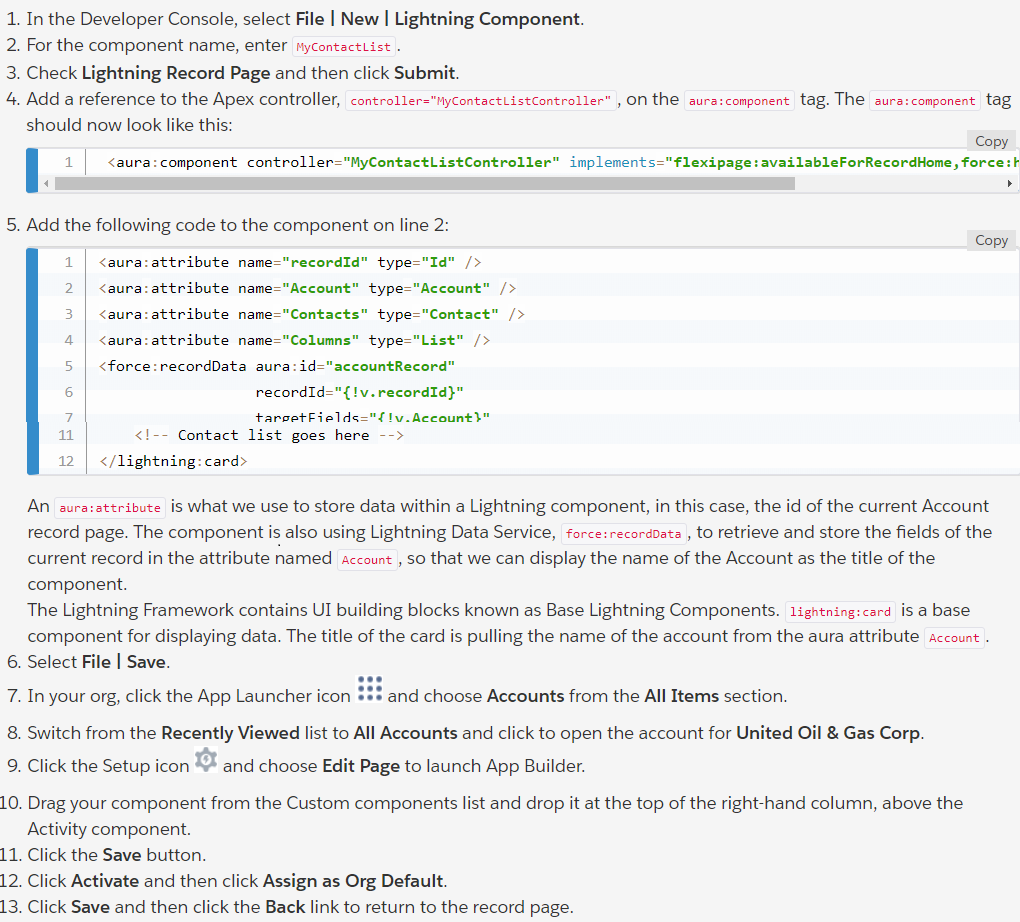
Once a process is activated, it cannot be edited. If you need to make edits to an activated process, follow these steps: (1) Clone the process as a version of the current process; (2) make changes to the cloned process; (3) activate the new version.

Module fait à l'arrache, éventuellement à refaire : [clic](https://trailhead.salesforce.com/projects/customize-a-sales-path-for-your-team)

Permet la création d'un path de sales, en gros du object manager avec création de relations etc.

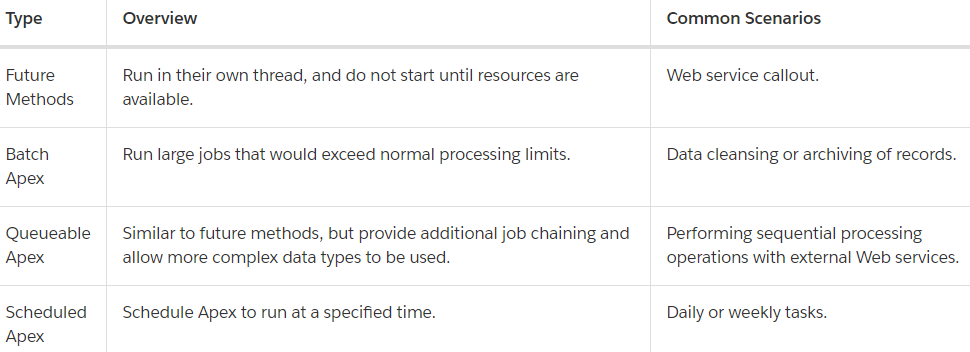
Créer une app Lightning :



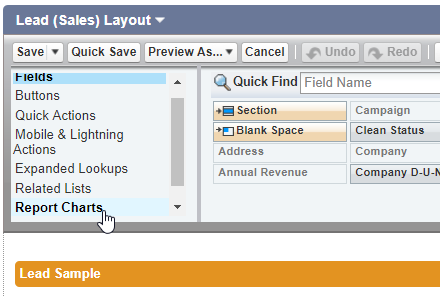
Crée Lightining Component :

Asynchronous Processing Basics [clic](https://trailhead.salesforce.com/modules/asynchronous_apex/units/async_apex_introduction)

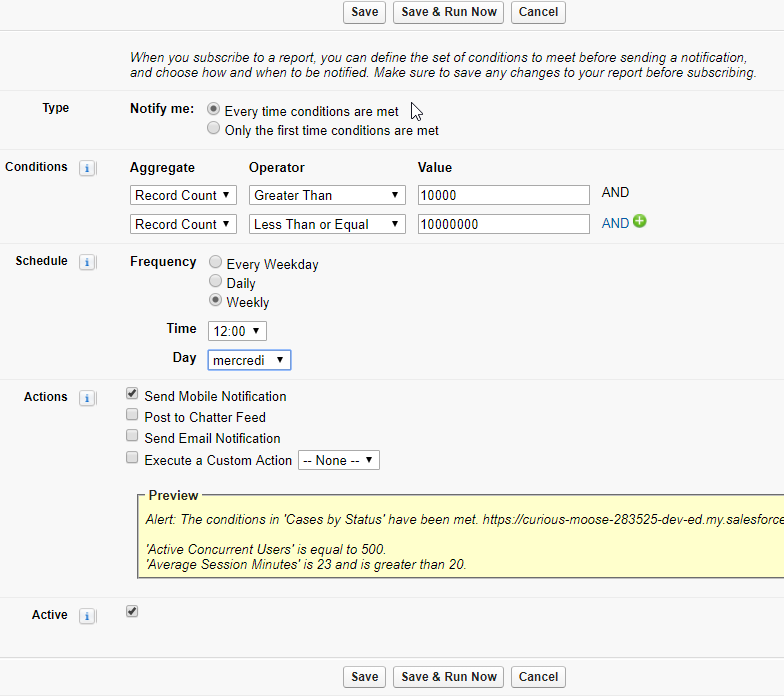
Utiliser du Apex asynchrone permet d'augmenter :  
- user efficiency  
- scalability  
- higher limits



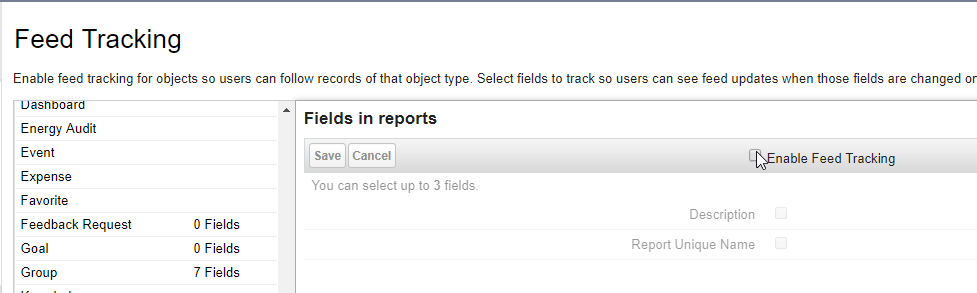
ADMIN :

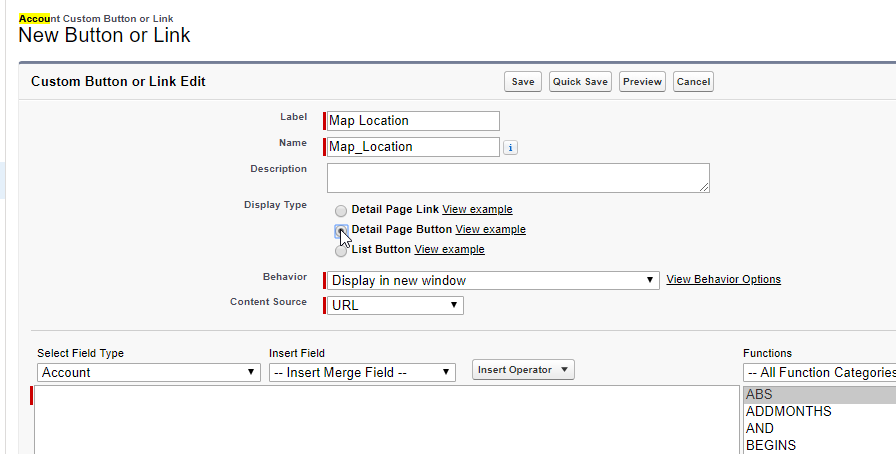
Ajouter un grpahique :  
Build 🡪 Customize 🡪 (nomObject) 🡪 Page Layout  
Puis on édit le layout désiré et :  


Activer les notifications : subscribe



Autoriser Feed Tracking pour suivre les rapports !

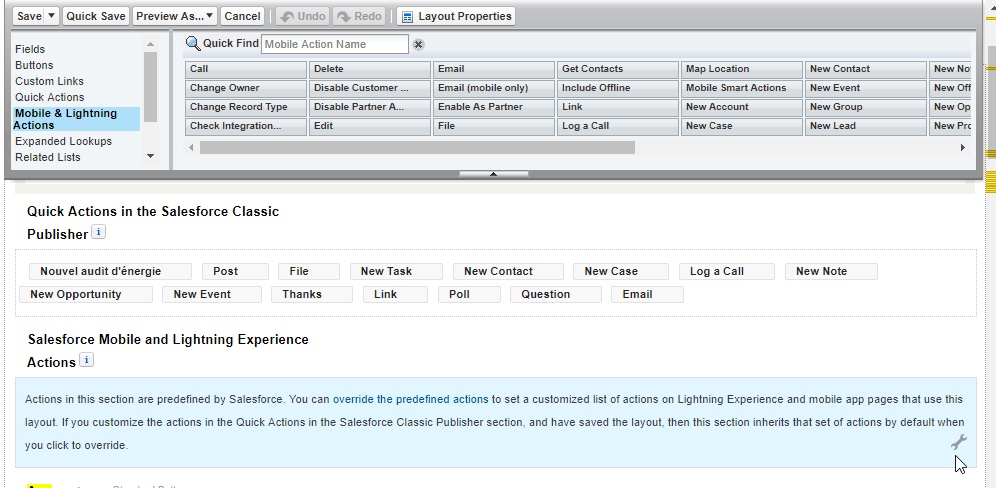


Custom boutons ! [clic](https://trailhead.salesforce.com/modules/lex_customization/units/lex_customization_buttons_links)  
La création d'un bouton passe par, dans l'objet (après object manager) sa définition (dans Buttons, Links etc qui est sous Fields & Relationship) :  


Restera ensuite à l'assigner à l'objet pour le faire apparaitre dans son layout.

Formule permettant de go direct à la location de l'account !  
[http://maps.google.com/maps?q={!Account\_BillingStreet}%20{!Account\_BillingCity}%20{!Account\_BillingState}%20{!Account\_BillingPostalCode}](http://maps.google.com/maps?q=%7b!Account_BillingStreet%7d%20%7b!Account_BillingCity%7d%20%7b!Account_BillingState%7d%20%7b!Account_BillingPostalCode%7d)

Nb : le %20 signifie un espace (32 enISO-latin, soit 20 en hexadécimal)

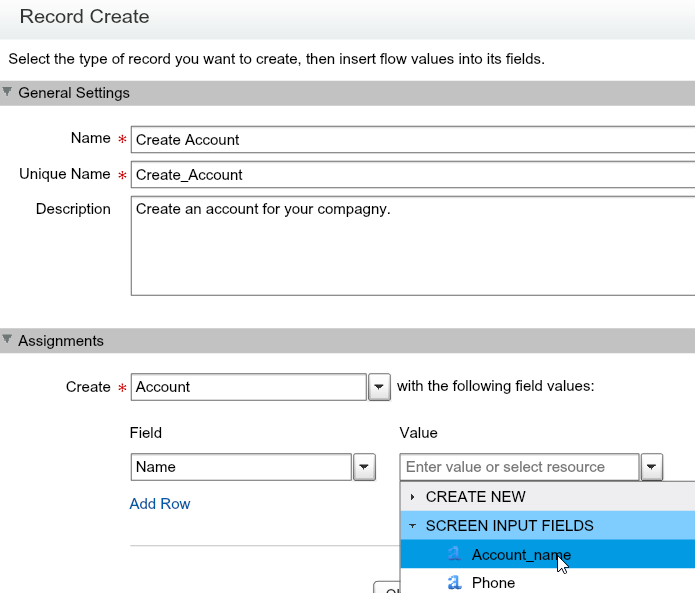
Pour rajouter une action sur Lightning, il faut passer par une customisation (bouton de réparation) :  


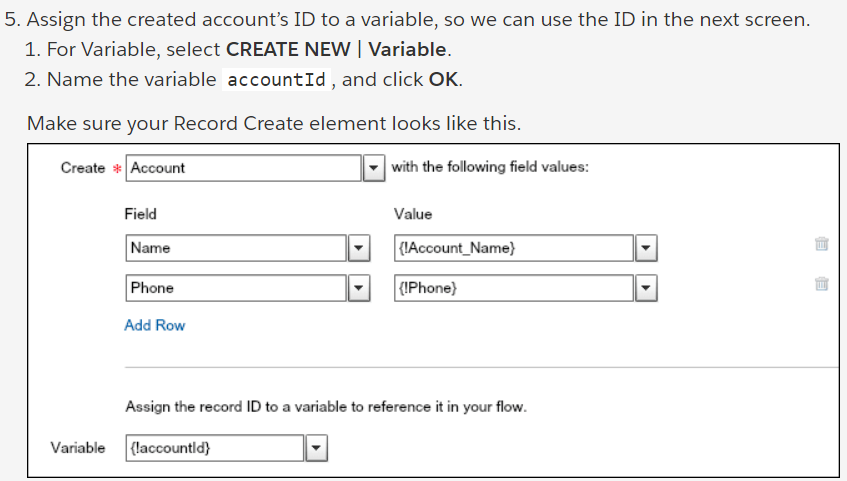
**11 juin 2018** **– Lightning Flows**

Configuration 🡪 flows

Trail [ici](https://curious-moose-283525-dev-ed.lightning.force.com/one/one.app#eyJjb21wb25lbnREZWYiOiJvbmU6YWxvaGFQYWdlIiwiYXR0cmlidXRlcyI6eyJhZGRyZXNzIjoiL2Rlc2lnbmVyL2Rlc2lnbmVyLmFwZXhwIn0sInN0YXRlIjp7fX0%3D)

On a créé un élément Screen avec Phone et Nom accout,  
On map ensuite dans le nouvel élément qu'on crée ensuite (Record Create) avec les données du Screen :

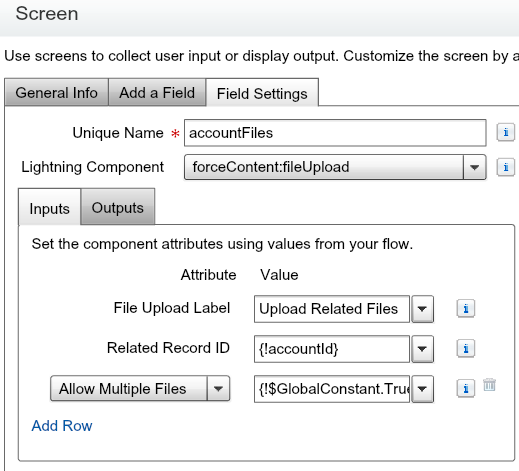


On assigne le compte juste créé à une variable pour pouvoir utiliser son Id :  


Création d'un screen d'upload de files : astuce trailhead 🡪   
" Under Navigation Actions, **deselect Allow Previous**.  
Otherwise, if users navigate back to the first screen, multiple accounts will be accidentally created."

Dans le deuxième screen, on paramètre en liant avec l'id de l'account créé (screen précédent) ;  
De plus on allow la multi sélection.

Résultat: " As configured, the component lets users upload more than one file at a time to the created account."



Pour ajouter le flow à une page :

Setup 🡪 Lightning App Builder 🡪 new Home Page 🡪 clone de la Salesforce Default Page

On ajoute Flow au layout, et on peut paramétrer à droite.

Petite vidéo récapitulative 3 min [ici](http://play.vidyard.com/WYDiNNK0SVJH6Kae42t5Mw) (présente une dropdown list)

**Apex synchrone & asynchrone** [**clic**](https://trailhead.salesforce.com/fr/modules/asynchronous_apex/units/async_apex_introduction)

Le nombre de requêtes SOQL est doublé. Il passe de 100 à 200 lors de l’utilisation d’appels asynchrones

**Les requêtes futures @future**

