Salesforce GROW

Module : Dynamic Apex and Describe Calls

Dynamic Apex and Describe Calls

Dynamic Apex

Dynamic Apex enables developers to create more flexible applications by providing them with the ability to:

Access sObject and field describe information

Describe information provides metadata information about sObject and field properties. For example, the describe information for an sObject includes whether that type of sObject supports operations like create or undelete, the sObject's name and label, the sObject's fields and child objects, and so on. The describe information for a field includes whether the field has a default value, whether it is a calculated field, the type of the field, and so on.

Write dynamic SOQL queries, dynamic SOSL queries and dynamic DML

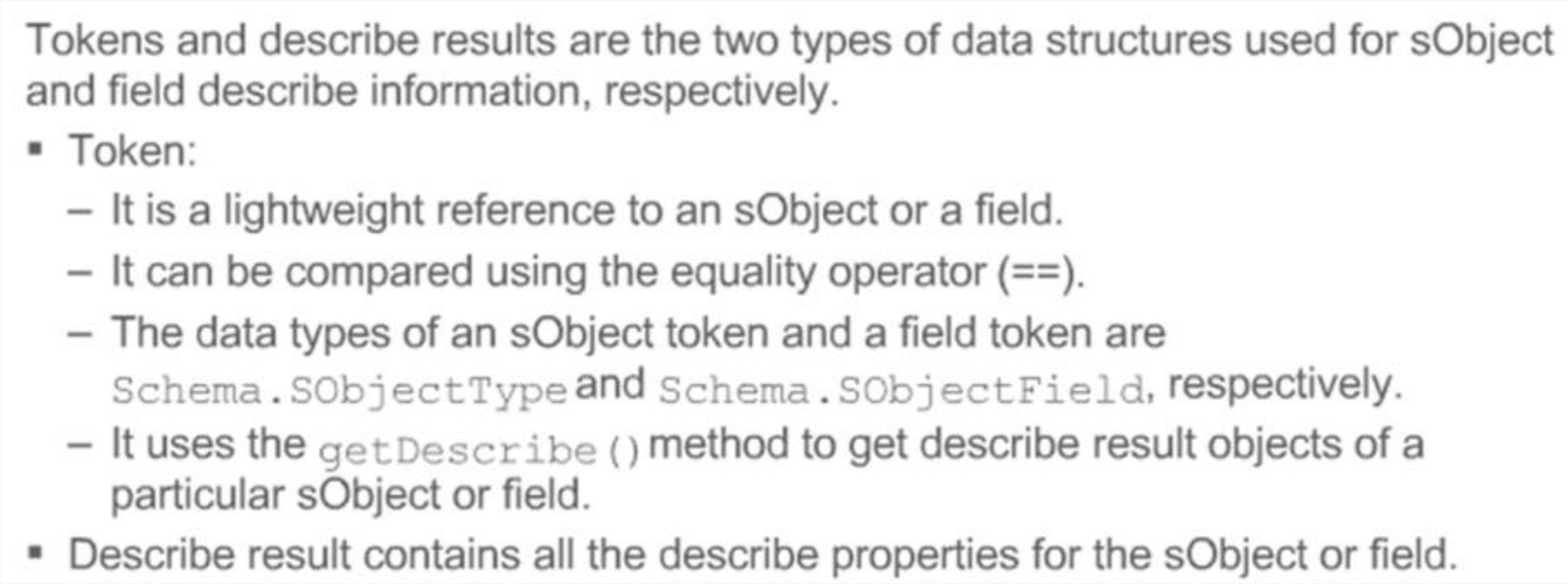
Dynamic SOQL and SOSL queries provide the ability to execute SOQL or SOSL as a string at runtime, while dynamic DMLprovides the ability to create a record dynamically and then insert it into the database using DML. This can be useful for applications that are installed from Force.com AppExchange.

Access Salesforce app information

Obtain describe information for standard and custom apps available in the Salesforce user interface. Each app corresponds to a collection of tabs. Describe information for an app includes the app’s label, namespace, and tabs. Describe information for a tab includes the sObject associated with the tab, tab icons and colors.

Access sObject and field describe information

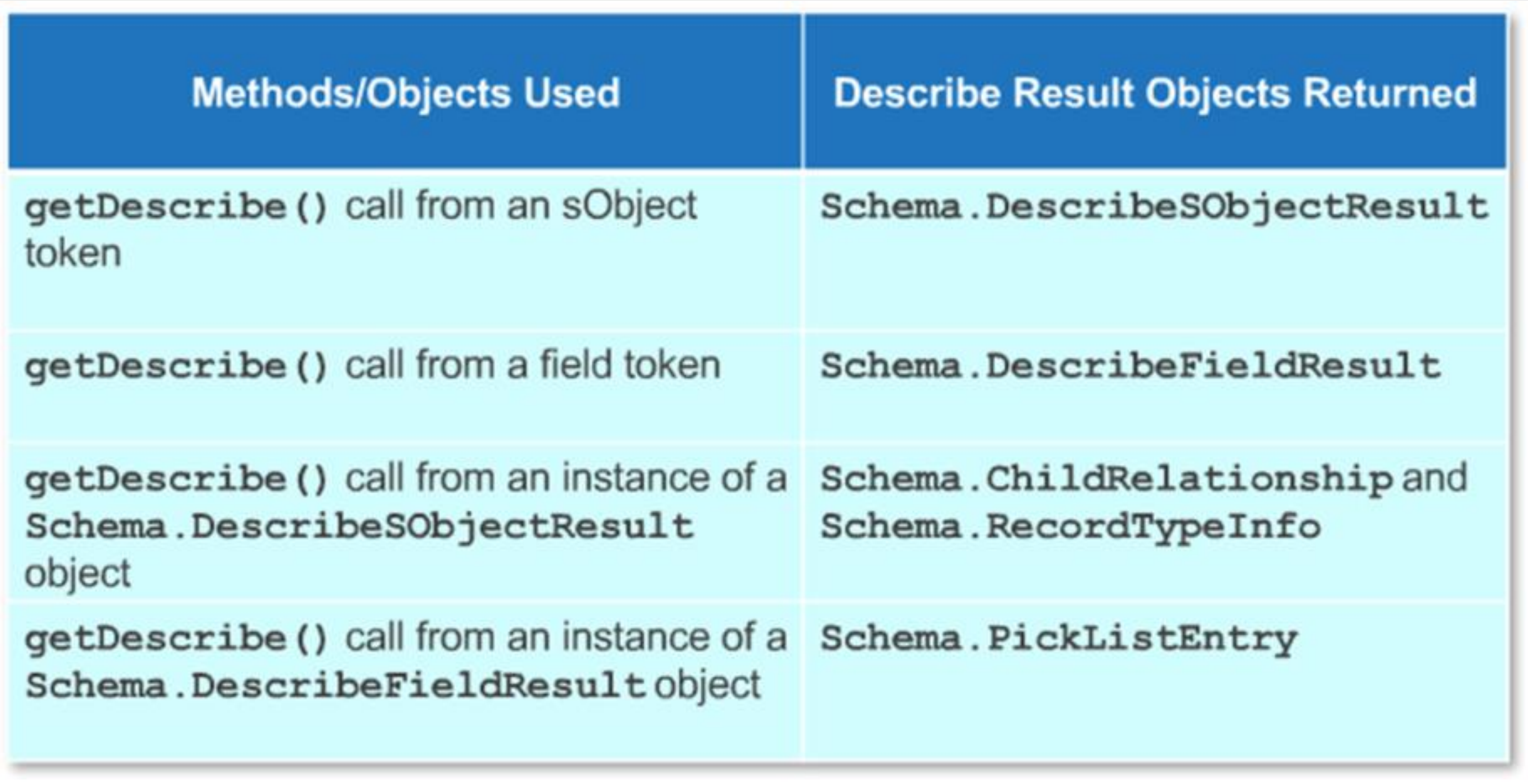
Apex Describe Structures



Tokens and describe results are the two types of data structures used for sObject and field describe information, respectively.

- Token:
  - It is a lightweight reference to an sObject or a field.
  - It can be compared using the equality operator (==).
  - The data types of an sObject token and a field token are `Schema.SObjectType` and `Schema.SObjectField`, respectively.
  - It uses the `getDescribe()` method to get describe result objects of a particular sObject or field.
- Describe result contains all the describe properties for the sObject or field.

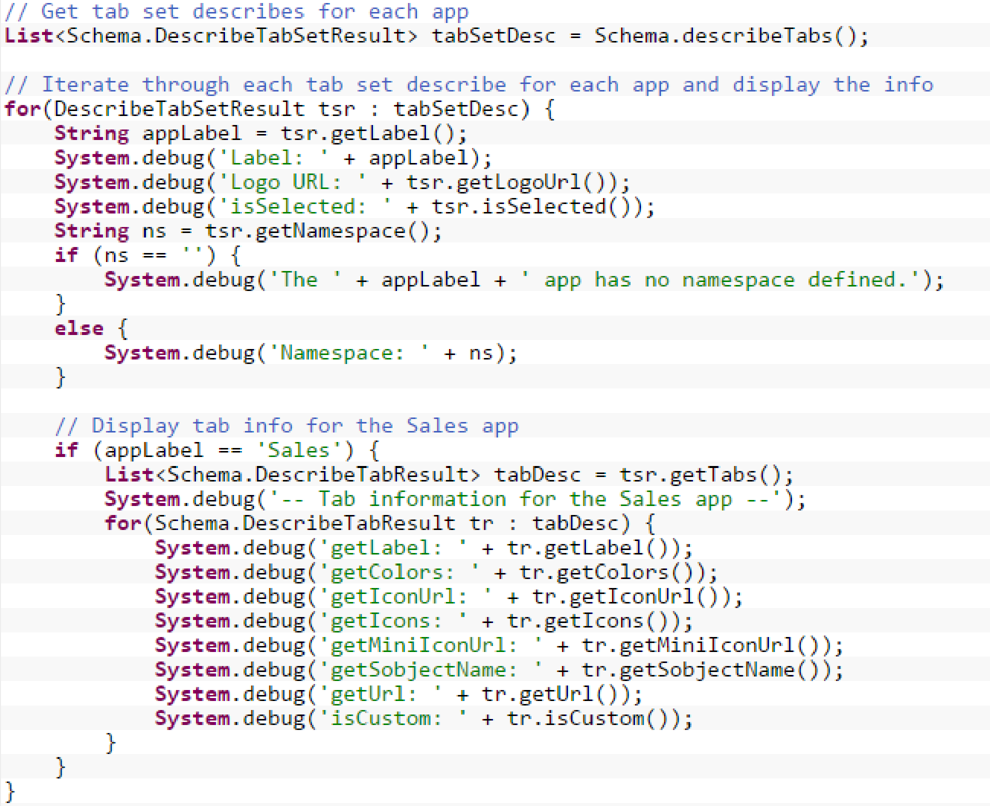
Schema Describe Result Objects



Methods/Objects Used	Describe Result Objects Returned
`getDescribe()` call from an sObject token	`Schema.DescribeSObjectResult`
`getDescribe()` call from a field token	`Schema.DescribeFieldResult`
`getDescribe()` call from an instance of a `Schema.DescribeSObjectResult` object	`Schema.ChildRelationship` and `Schema.RecordTypeInfo`
`getDescribe()` call from an instance of a `Schema.DescribeFieldResult` object	`Schema.PickListEntry`

Access Salesforce App Information: Describing Tabs

Get metadata information about the apps and tabs available in the Salesforce user interface by executing a describe call in Apex. The **describeTabs** Schema method and the **getTabs** method in Schema. DescribeTabResult, respectively.



```
// Get tab set describes for each app
List<Schema.DescribeTabSetResult> tabSetDesc = Schema.describeTabs();

// Iterate through each tab set describe for each app and display the info
for(DescribeTabSetResult tsr : tabSetDesc) {
    String appLabel = tsr.getLabel();
    System.debug('Label: ' + appLabel);
    System.debug('Logo URL: ' + tsr.getLogoUrl());
    System.debug('isSelected: ' + tsr.isSelected());
    String ns = tsr.getNamespace();
    if (ns == '') {
        System.debug('The ' + appLabel + ' app has no namespace defined.');
    }
    else {
        System.debug('Namespace: ' + ns);
    }

    // Display tab info for the Sales app
    if (appLabel == 'Sales') {
        List<Schema.DescribeTabResult> tabDesc = tsr.getTabs();
        System.debug('-- Tab information for the Sales app --');
        for(Schema.DescribeTabResult tr : tabDesc) {
            System.debug('getLabel: ' + tr.getLabel());
            System.debug('getColors: ' + tr.getColors());
            System.debug('getIconUrl: ' + tr.getIconUrl());
            System.debug('getIcons: ' + tr.getIcons());
            System.debug('getMiniIconUrl: ' + tr.getMiniIconUrl());
            System.debug('getSobjectName: ' + tr.getSobjectName());
            System.debug('getUrl: ' + tr.getUrl());
            System.debug('isCustom: ' + tr.isCustom());
        }
    }
}
```

Write Dynamic SOQL

Dynamic SOQL refers to the creation of a SOQL string at runtime with Apex code.

To create a dynamic SOQL query at runtime, use the database query method, in one of the following ways:

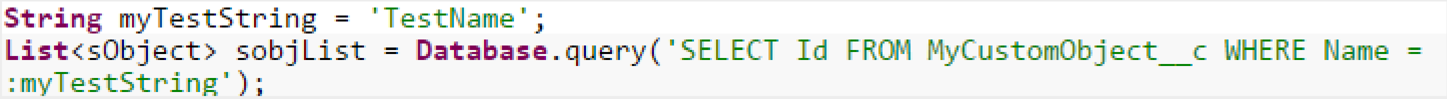
- Return a single sObject when the query returns a single record:

  sObject s = Database.query(string_limit_1);

- Return a list of sObjects when the query returns more than a single record:

  List<sObject> sobjList = Database.query(string);

<u>Example</u>:



```
String myTestString = 'TestName';
List<sObject> sobjList = Database.query('SELECT Id FROM MyCustomObject__c WHERE Name =
:myTestString');
```

Dynamic SOQL results can be specified as concrete sObjects, such as Account or MyCustomObject__c, or as the generic sObject data type. At runtime, the system validates that the type of the query matches the declared type of the variable. If the query does not return the correct sObject type, a runtime error is thrown.

Dynamic SOQL queries have the same governor limits as static queries.

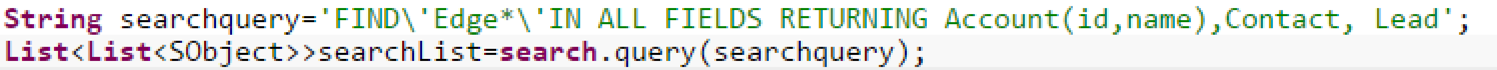
Write Dynamic SOSL

Dynamic SOSL refers to the creation of a SOSL string at run time with Apex code.

To create a dynamic SOSL query at run time, use the search query method.

Syntax:

List<List <sObject>> myQuery = search.quer



```
String searchquery='FIND\'Edge*\'IN ALL FIELDS RETURNING Account(id,name),Contact, Lead';
List<List<SObject>>searchList=search.query(searchquery);
```

Example:

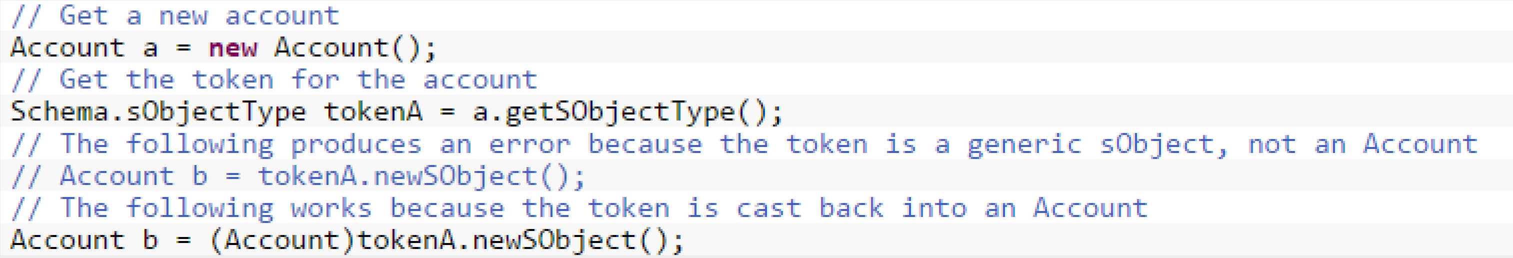
Dynamic SOSL statements evaluate to a list of lists of sObjects, where each list contains the search results for a particular sObject type. The result lists are always returned in the same order as they were specified in the dynamic SOSL query.

Dynamic SOSL queries have the same governor limits as static queries.

Write Dynamic DML

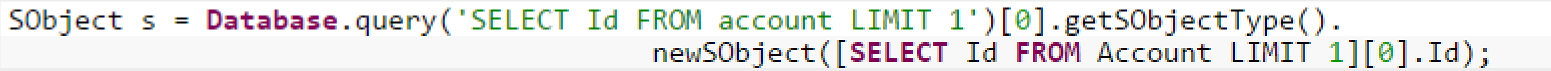
In addition to querying describe information and building SOQL queries at runtime, we can create sObjects dynamically and insert them into the database using DML.

To create a new sObject of a given type, use the newSObject method on an sObject token. Note that the token must be cast into a concrete sObject type (such as Account).



```
// Get a new account
Account a = new Account();
// Get the token for the account
Schema.sObjectType tokenA = a.getSObjectType();
// The following produces an error because the token is a generic sObject, not an Account
// Account b = tokenA.newSObject();
// The following works because the token is cast back into an Account
Account b = (Account)tokenA.newSObject();
```

We can also specify an ID with newSObject to create an sObject that references an existing record that can be updated later.



```
SObject s = Database.query('SELECT Id FROM account LIMIT 1')[0].getSObjectType().
                                       newSObject([SELECT Id FROM Account LIMIT 1][0].Id);
```

Hands-On Exercise

Apex: Advanced Topics
Exercise 10-1: Searching with Dynamic SQL

https://lms.cfs-api.com/v1/content/68ccef73-165d-46a1-a290-50ddd30a0201/presentation_content/external_files/AdvancedTopicsExerciseGuide.pdf

Stop at Page 7

Understanding Apex Describe Information:

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dynamic_describe_objects_understanding.htm

Describing Tabs Using Schema Methods:

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dynamic_describeTabs.htm

Dynamic SOQL,SOSL and DML:

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dynamic_soql.htm

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dynamic_sosl.htm

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dynamic_dml.htm