LF

LearnFrenzy

Apex Interview Cheat Sheet

This content document is related to the Apex Cheat Sheet.
It covers the most important **real-time interview Q&A, scenarios, and code examples** every Salesforce Developer should know in 2026.

1. What they asked me:

What are Governor Limits in Salesforce? Can you give real examples?

What I said:
Governor limits are runtime restrictions Salesforce enforces to ensure no single transaction monopolizes resources. Examples:

- Max 100 SOQL queries per transaction.
- Max 150 DML statements.
- Max CPU time: 10,000 ms synchronous / 60,000 ms async.
- Heap size: 6 MB synchronous / 12 MB async.

Real Scenario:
In a trigger handling OpportunityLineItem updates, querying inside a loop causes **SOQL 101 error**. Solution: use bulkified queries.

```
// Wrong: Query inside loop
for(OpportunityLineItem oli : Trigger.new){
    Account acc = [SELECT Id, Name FROM Account WHERE Id=:oli.AccountId];
}

// Correct: Bulkified
Set<Id> accIds = new Set<Id>();
for(OpportunityLineItem oli : Trigger.new){
    accIds.add(oli.AccountId);
}
Map<Id, Account> accMap = new Map<Id,
    Account>( [SELECT Id, Name FROM Account WHERE Id
    IN :accIds]
);
```

Tips:
Always mention you **design bulk-safe code** to respect governor limits.

---

2. What they asked me:

LF

Explain the difference between Before and After triggers with an example.

What I said:

- **Before Trigger**: Modify record values before saving to DB. Example: set default field values.
- **After Trigger**: Access record IDs and related records after saving. Example: create child records.

Scenario:
When Case is inserted, set priority before saving:

```
trigger CaseTrigger on Case (before insert, after insert)
   { if(Trigger.isBefore){
      for(Case c : Trigger.new){
         if(c.Priority == null) c.Priority = 'Medium';
      }
   }
   if(Trigger.isAfter){
      for(Case c : Trigger.new){
         Task t = new Task(Subject='Follow up', WhatId=c.Id);
         insert t;
      }
   }
}
```

Tips:

- Use *before* for validation/defaults.
- Use *after* for creating related records (child objects).

---

3. What they asked me:

Difference between Future, Queueable, and Batch Apex?

What I said:

- **Future**: Lightweight async, no chaining, no complex return.
- **Queueable**: Supports chaining, allows complex objects, monitoring in Apex Jobs.
- **Batch**: Best for millions of records, runs in chunks of 200.

Scenario:
Processing 5M records → use **Batch Apex**.

LF

For chained async jobs (callouts, retry) → use **Queueable**.
For simple tasks (update flags) → use **Future**.

Tips:
Interviewers love when you say:
"I choose the async type based on **volume, complexity, and chaining needs**."

---

4. What they asked me:

How do you handle SOQL 101 error (too many queries)?

What I said:

- Avoid SOQL inside loops.
- Use Maps/Sets for bulk queries.
- Pre-fetch data.

Scenario:
Updating Contacts with Account Industry:

```
trigger ContactTrigger on Contact (before insert, before update)
   { Set<Id> accIds = new Set<Id>();
   for(Contact c : Trigger.new){
      if(c.AccountId != null) accIds.add(c.AccountId);
   }
   Map<Id, Account> accMap = new Map<Id,
      Account>( [SELECT Id, Industry FROM Account WHERE Id
      IN :accIds]
   );
   for(Contact c :
      Trigger.new){ if(accMap.containsKey(c.AccountId)){
         c.Industry__c = accMap.get(c.AccountId).Industry;
      }
   }
}
```

Tips:
Always explain with **bulk-safe example**.

---

5. What they asked me:

How do you prevent recursion in triggers?

What I said:
Use static variables in a helper class.

```
public class TriggerHelper {
   public static Boolean isFirstRun = true;
}
trigger AccountTrigger on Account (after update)
   { if(TriggerHelper.isFirstRun){
      TriggerHelper.isFirstRun = false;
      // business logic
   }
}
```

Tips:
Say: *"I design triggers using handler frameworks and static flags to avoid recursion."*

---

6. What they asked me:

What's the difference between With Sharing, Without Sharing, and Inherited Sharing?

What I said:

- **With Sharing** → Enforces sharing rules.
- **Without Sharing** → Ignores sharing rules.
- **Inherited Sharing** → Uses context from caller (best practice for service classes).

Tips:
Always say: *"For service classes, I use Inherited Sharing to respect security."*

---

7. What they asked me:

How do you write a Test Class? Best practices?

What I said:

- Use @isTest.
- Create test data (don't rely on org data).
- Cover positive + negative scenarios.
- Assert results.

Example:

```
@isTest
private class AccountTriggerTest
   { @isTest static void
   testAccountInsert(){
      Account acc = new Account(Name='Test Acc');
      insert acc;

      // Fetch inserted
      Account result = [SELECT Id, Name FROM Account WHERE Id=:acc.Id];
      System.assertEquals('Test Acc', result.Name);
   }
}
```

Tips:
Mention:
"I always keep >75% coverage."
"I write tests for both happy and edge cases."

---

8. What they asked me:

What's the difference between SOQL and SOSL?

What I said:

- **SOQL**: Query specific objects, structured. (e.g., `SELECT Id, Name FROM Account`).
- **SOSL**: Search across multiple objects, full-text search.

Tips:
Say: *"I use SOSL when searching multiple objects in global search use cases."*

---

9. What they asked me:

How do you make Apex trigger bulk-safe?

What I said:

- Process all records in `Trigger.new`.
- Collect IDs in Set.
- Query once, store in Map.
- Perform DML outside loop.

LF

Tips:
Interviewers expect **best practices explanation with code**.

---

10. What they asked me:

How do you design a REST API callout in Apex?

What I said:

```
public with sharing class FXRateService {
    public static Decimal getRate(String from, String
        to){ Http h = new Http();
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:Frankfurter/api/latest?from='+from+'&to='+to);
        req.setMethod('GET');
        HttpResponse res = h.send(req);
        if(res.getStatusCode()==200){
            Map<String,Object> result = (Map<String,Object>) JSON.deserializeUntyped(res.getBody());
            return Decimal.valueOf(result.get('rates').get(to));
        }
        return 0;
    }
}
```

Tips:
Always mention:
Use **Named Credentials** for security.
Wrap callouts with **try-catch**.