Contents

TRIGGERS

1.What is a trigger?

A trigger in Salesforce is a piece of Apex code that executes automatically in response to certain events on a particular object. Triggers allow you to perform custom actions, such as updating related records, enforcing complex business logic, or automating processes when data changes in Salesforce.
Types of Triggers
1. Before Trigger: Executes before a record is inserted, updated, or deleted. Used to validate or modify data before it is saved to the database.
2. After Trigger: Executes after a record is inserted, updated, or deleted. Used for actions that require access to the record's database ID or related objects.

2.various trigger context variables?

1. Trigger.new
Contains a list of new versions of the records that are being inserted or updated.
Available in: Insert, Update, Undelete triggers.

2. Trigger.old
Contains a list of old versions of the records before they were updated or deleted.
Available in: Update, Delete triggers.

3. Trigger.newMap
A map of record IDs to their new versions (records being inserted or updated).
Available in: Insert, Update, Undelete triggers.

4. Trigger.oldMap
A map of record IDs to their old versions (records before they were updated or deleted).
Available in: Update, Delete triggers.

5. Trigger.isInsert
Returns true if the trigger was fired due to an insert operation.

6. Trigger.isUpdate
Returns true if the trigger was fired due to an update operation.

7. Trigger.isDelete
Returns true if the trigger was fired due to a delete operation.

8. Trigger.isBefore
Returns true if the trigger was fired before the record was saved to the database.

9. Trigger.isAfter
Returns true if the trigger was fired after the record was saved to the database.

3.Difference between before and after triggers?

Before and After triggers in Salesforce are differentiated by when they execute in relation to the DML operation (e.g., INSERT, UPDATE, DELETE) and their purpose:

1. Timing

Before Trigger: Executes before the record is saved to the database.
After Trigger: Executes after the record is saved to the database.

2. Purpose

Before Trigger:
Primarily used to validate or modify data before it is committed to the database.
Example: Ensuring that a field is populated or updating a field value based on certain conditions.
Modifications made in Trigger.new are automatically saved; no explicit DML operation is required.
After Trigger:
Used for actions that require access to the record's database ID or related objects, which are only available after the record is saved.
Example: Creating related records, sending email notifications, or logging data to external systems.

4. Best Practices for Writing Triggers

Here are some key best practices for writing efficient and maintainable triggers in Salesforce:

1. **One Trigger Per Object**:
   - Write a single trigger per object to avoid conflicts and better manage logic.
2. **Use Context Variables**:
   - Use context variables like Trigger.new, Trigger.old, Trigger.isInsert, Trigger.isUpdate, etc., to ensure appropriate behavior.
3. **Avoid Hardcoding**:
   - Use custom settings, custom labels, or metadata to avoid hardcoding values.
4. **Bulkify Your Trigger**:
   - Ensure the trigger handles bulk records by processing lists and collections instead of single records.
5. **Avoid SOQL/DML in Loops**:
   - Move SOQL and DML operations outside loops to avoid hitting governor limits.
6. **Use Helper Classes**:
   - Delegate complex logic to Apex helper classes for modularity and reusability.
7. **Consider Trigger Order of Execution**:
   - Ensure triggers do not unintentionally conflict with workflows, processes, or other automations.
8. **Error Handling**:
   - Implement try-catch blocks for exception handling, and provide meaningful error messages.

9. **Test Coverage**:
    - Write comprehensive test classes to ensure at least 75% coverage and test for bulk operations and edge cases.
10. **Use @future or Asynchronous Apex**:
    - For long-running operations, move the logic to a future method to improve performance.
11. **Document the Code**:
    - Use comments to explain the trigger's purpose and logic for future reference.
12. **Disable Triggers for Specific Contexts**:
    - Add logic to disable triggers for specific scenarios (e.g., data migrations).

---

5. Order of Execution in Salesforce

The Salesforce order of execution defines the sequence of events that occur when a record is created, updated, deleted, or undeleted:

1. **System Validation Rules**:
    - Validates system constraints (e.g., required fields, field formats).
2. **Before Triggers**:
    - Executes before insert or before update triggers.
3. **Custom Validation Rules**:
    - Executes any user-defined validation rules.
4. **Duplicate Rules**:
    - Executes duplicate management rules (if configured).
5. **After Triggers**:
    - Executes after insert, after update, or after delete triggers.
6. **Assignment Rules**:
    - Executes assignment rules (for leads or cases).
7. **Auto-Response Rules**:
    - Executes auto-response rules (for leads or cases).
8. **Workflow Rules**:
    - Evaluates and executes workflow rules.
    - Updates from field updates trigger before and after update triggers again.
9. **Process Builder/Flows**:
    - Executes processes and flows if configured.
10. **Escalation Rules**:
    - Executes escalation rules (for cases).
11. **Entitlement Rules**:
    - Applies entitlement rules (for cases).
12. **Roll-Up Summary Fields**:
    - Updates roll-up summary fields (for parent objects).
13. **Parent Triggers**:
    - Executes triggers on parent objects (if changes occur due to roll-up fields).
14. **Post-Commit Logic**:
    - Executes post-commit actions like email alerts and asynchronous Apex (@future, batch jobs).

APEX

---

1. Best Practices in Apex

- Write **bulkified code** to handle multiple records (e.g., use for loops carefully).
- Avoid **SOQL/DML inside loops** to prevent governor limit issues.
- Use **collection types** (e.g., maps, lists, sets) for efficient operations.
- Implement proper **exception handling**.
- Use **@future** and other asynchronous operations wisely for heavy processing.
- Write **test classes** with 75%+ code coverage and meaningful assertions.

---

2. What Are Governor Limits?

Governor limits are Salesforce-enforced limits that ensure efficient resource usage in a multitenant environment. Examples include:

- **SOQL Queries**: Max 100 queries per transaction.
- **DML Statements**: Max 150 per transaction.
- **Heap Size**: 6 MB for synchronous and 12 MB for asynchronous Apex.

---

3. Various DML Operations

- **Insert**: Add new records.
- **Update**: Modify existing records.
- **Delete**: Remove records.
- **Undelete**: Restore deleted records.
- **Upsert**: Insert or update based on record existence.
- **Merge**: Combine up to 3 records of the same type.

---

4. Apex Syntax

- Similar to Java.
- Classes: public class MyClass {}
- Methods: public void myMethod() {}
- Variables: String name = 'Salesforce';
- Conditional: if (condition) {}

---

5. List Methods

- add(element): Adds an element.
- get(index): Returns the element at a specified index.
- size(): Returns the size of the list.
- remove(index): Removes the element at a specific index.
- set(index, element): Replaces an element.

---

6. Apex Class & Method Syntax

```
public class MyClass {
   public void myMethod() {
      System.debug('Hello World');
   }
}
```

---

7. Share Apex Class Using Virtual Keyword

Use the virtual keyword to allow a class to be extended.

```
public virtual class ParentClass {
   public virtual void displayMessage() {
      System.debug('Parent Message');
   }
}
```

---

8. Various Access Modifiers

- **Private**: Visible only within the same class.
- **Public**: Accessible from any class.
- **Protected**: Accessible in the same class or subclasses.
- **Global**: Accessible across namespaces.

---

9. Difference Between With & Without Sharing

- **With Sharing**: Enforces user sharing rules.
- **Without Sharing**: Ignores user sharing rules.

---

10. How to Delete Apex Class in Production?

- Classes in production cannot be deleted directly.
- First, remove dependencies (e.g., triggers, references).
- Deploy changes through a sandbox using deployment tools like **Change Sets** or **ANT**.

---

11. What Is a Test Class?

- Test classes validate the functionality of Apex code.
- Required to achieve minimum 75% code coverage for deployment.
- Written using @isTest annotation.

---

12. What Is Static & Final Keyword?

- **Static**: Shared across all instances of the class (e.g., static Integer count;).
- **Final**: Prevents a variable's value or method from being overridden.

---

13. What Are Test Clusters?

Likely a misinterpretation; in Salesforce, this term might not be standard. Clarify if this relates to testing concepts or batch testing groups.

---

14. OOP in Apex

- **Encapsulation**: Grouping code and data in classes.
- **Inheritance**: Extending functionality of parent classes.
- **Polymorphism**: Overloading and overriding methods.
- **Abstraction**: Hiding implementation details.

---

15. Data Binding & Method Overloading

- **Data Binding**: Linking a variable to a UI element or another variable.
- **Method Overloading**: Defining multiple methods with the same name but different parameters.

---

16. Asynchronous Apex & Exception Handling

- **Asynchronous Apex**:
    - **@future**: Processes in the background.

    - **Batch Apex**: Handles large data sets.
    - **Queueable Apex**: Chainable background jobs.
    - **Scheduled Apex**: Executes at specific times.
- **Exception Handling**: Use try-catch blocks to handle runtime errors gracefully.

LWC

---

1. What Are Decorators?

- Decorators are special symbols used to annotate and modify the behavior of classes, properties, methods, or parameters in LWC.
- Common decorators:
    - @api: Makes a property or method publicly accessible to the parent component.
    - @track: Tracks changes to an object or array for reactivity.
    - @wire: Used to connect a component to Salesforce data or a custom Apex method.

---

2. What Are Life Cycle Hooks?

Lifecycle hooks are methods in LWC that allow developers to tap into the component's lifecycle for specific logic. Common hooks include:

- **constructor()**: Invoked when the component is created.
- **connectedCallback()**: Executes when the component is inserted into the DOM.
- **renderedCallback()**: Runs after the component's template is rendered.
- **disconnectedCallback()**: Executes when the component is removed from the DOM.
- **errorCallback(error, stack)**: Captures and handles errors.

---

3. @api and @track

- **@api**:
    - Exposes a property or method to the parent component.
    - Enables parent-child communication.
- @api propertyName;
- **@track**:
    - Tracks changes to an object or array for reactivity.
    - Automatically updates the UI when changes occur.
- @track myArray = [];

---

4. What Are Annotations?

Annotations are metadata markers used in Apex to define the behavior of classes and methods. Common annotations:

- **@AuraEnabled**: Exposes Apex methods to LWC and Aura.
- **@isTest**: Marks a class or method as a test.
- **@future**: Runs a method asynchronously.
- **@InvocableMethod**: Enables methods to be called from flows.
- **@TestSetup**: Creates test data reusable across multiple test methods.

---

5. Different Types of Events in LWC

Events facilitate communication between components. Types include:

1. **Standard DOM Events**: Predefined events like click, change, and mouseover.
    - Example: <button onclick={handleClick}>Click Me</button>
2. **Custom Events**: Created using CustomEvent for custom communication.
3. this.dispatchEvent(new CustomEvent('mycustomevent', { detail: { key: 'value' } }));
4. **Application Events**: Handled at the application level for broader communication.

5. **System Events**: Events triggered by the system (e.g., load, error).

Salesforce

1. Difference Between Standard & Custom Objects

- **Standard Objects**: Predefined by Salesforce (e.g., Account, Contact, Opportunity).
- **Custom Objects**: Created by users to store specific data related to their business needs.

---

2. Types of Relationships

- **One-to-One**: Rare, implemented using unique lookup fields.
- **One-to-Many**: Standard (e.g., Account to Contact).
- **Many-to-Many**: Achieved using a **junction object**.

---

3. What Is a Junction Object?

- A custom object used to establish a many-to-many relationship between two objects.
- Example: A "Course Enrollment" object linking "Student" and "Course."

---

4. Security Settings for Junction Object

- Inherit security settings from its parent objects (via Master-Detail relationships).
- Use profiles and permission sets to control access.

---

5. Difference Between Master-Detail & Lookup

- **Master-Detail**:
  - Parent controls child's record ownership and security.
  - Cascade delete and roll-up summaries are available.
- **Lookup**:
  - More loosely coupled; independent records.
  - No cascade delete or roll-up summaries.

---

6. Convert Lookup to Master-Detail

- Possible if all child records have a parent.

---

7. Relationships Between Standard Objects

- **Account ↔ Contact**: One-to-Many.
- **Account ↔ Opportunity**: One-to-Many.
- **Account ↔ Case**: One-to-Many.

---

8. Ways to Share a Record

- Role hierarchy
- Manual sharing
- Sharing rules
- Public groups
- Permission sets

---

9. Can a User Be Deleted?

- No, users cannot be deleted; they can only be deactivated.

---

10. Difference Between Freeze & Deactivate User

- **Freeze**: Prevents login temporarily.
- **Deactivate**: Permanently disables login and removes the user from roles.

---

11. What Is Manual Sharing?

- Allows record owners or admins to share specific records with other users.

12. What Are Sharing Settings?

- Define how records are shared at the organization, object, or record level.

13. What Are OWD Settings?

- **Organization-Wide Defaults (OWD)**: Set baseline record access (Public Read/Write, Public Read-Only, Private).

14. Difference Between Role, Profile & User

- **Role**: Controls data access (e.g., hierarchy-based sharing).
- **Profile**: Controls permissions and functionality.
- **User**: Actual individual accessing the system.

15. What Is a Profile?

- A collection of permissions controlling object, field, and application access.

16. What Is Role Hierarchy?

- Allows record sharing up the hierarchy (e.g., manager accessing subordinates' data).

17. Difference Between Flows, Process Builder & Workflow

- **Workflow**: Limited to simple automation.
- **Process Builder**: More advanced; can call flows.
- **Flows**: Most powerful; supports complex logic and screens.

18. Record Access Levels

- **Private**: Only owner and admins can access.
- **Public Read/Write**: All users can view and edit.
- **Public Read-Only**: All users can view but not edit.

19. Difference Between Data Import Wizard & Data Loader

- **Data Import Wizard**: Simple tool; supports limited objects.
- **Data Loader**: Advanced tool for bulk operations; supports all objects.

20. SOQL vs. SOSL

- **SOQL**: Retrieves records using conditions.
- **SOSL**: Searches across multiple objects and fields.

21. Developer Console Not Visible

- Possible reasons: Profile permissions, browser cache, or network issues.

22. What Is a Permission Set?

- Provides additional permissions to users without modifying their profiles.

23. What Is Field-Level Security?

- Controls visibility of specific fields for profiles or permission sets.

24. What Is a Change Set?

- Tool for deploying customizations between Salesforce orgs.

25. What Are Custom Settings?

- Customizable data stored at org or profile level, used in formulas and logic.

26. What Is Outbound Message?

- Sends SOAP messages to external systems based on a workflow or process.

27. Cascading Delete in Master-Detail

- When the parent record is deleted, all child records are also deleted.

---

28. Types of Reports & Dashboards

- **Reports**: Tabular, Summary, Matrix, Joined.
- **Dashboards**: Visual summaries using charts, graphs, or tables.

---

29. Difference Between Set & List

- **Set**: Unique, unordered collection.
- **List**: Ordered collection, can have duplicates.

---

30. Controlling Field Types

- **Picklist** or **Checkbox** fields can be controlling fields.

---

31. View All & Modify All Permissions

- **View All**: Read access to all records of an object.
- **Modify All**: Read, edit, and delete all records of an object.

---

32. Various Licenses in Salesforce

- **Salesforce**
- **Salesforce Platform**
- **Community**
- **Service Cloud**
- **Chatter Free**

---

33. What Is a Queue?

- A collection of records shared among a group of users.

---

34. What Is Feed Tracking?

- Tracks changes to fields and posts updates to the Chatter feed.

---

35. Types of Sandboxes

- **Developer**
- **Developer Pro**
- **Partial Copy**
- **Full Copy**

PFIZHER

Interviewer: *Can you explain your role in the Pfizer Pharma project?*

You:
"In the Pfizer Pharma project, I worked as a Salesforce Developer/Administrator, contributing to the design, implementation, and optimization of several key processes to support Pfizer's business needs. My role involved:

1. **Custom Object and Data Model Design**:
    - I developed and managed custom objects like **Healthcare Professional (HCP)**, **Drug Sample Distribution**, **Sales Territory**, and **Compliance Audit** to align with Pfizer's business requirements.
    - I also implemented relationships between objects, such as Master-Detail and Lookup, to ensure data integrity and effective reporting.

2. **Automation and Business Logic**:
    - I automated processes using **Flows**, **Process Builder**, and **Apex Triggers**. For example, I implemented a **lead assignment flow** based on zip codes to ensure leads were automatically routed to the correct sales reps.
    - Created validation rules, such as ensuring HCP licenses were valid during record creation.
3. **Custom Development**:
    - Developed **Apex Classes and Triggers** to handle complex logic, such as bulk updates to drug sample requests and territory assignments.
    - Integrated third-party systems using APIs to enable seamless data exchange, such as automating drug inventory updates.
4. **Security and Access Control**:
    - Configured sharing rules, role hierarchies, and OWD settings to ensure data confidentiality and access control, critical for compliance with healthcare regulations.
    - Managed **profiles**, **permission sets**, and **field-level security** to tailor access for various user roles, such as sales reps, marketers, and compliance officers.
5. **Reports and Dashboards**:
    - Designed custom **reports** and **dashboards** for sales and compliance teams to monitor KPIs, such as drug sample distribution efficiency and sales team performance.
6. **Collaboration with Stakeholders**:
    - Worked closely with Pfizer's sales, marketing, and compliance teams to gather requirements, deliver solutions, and train end-users.

Through my contributions, I helped Pfizer improve their operational efficiency, enhance regulatory compliance, and streamline data management processes within Salesforce."

HR

1. Tell me about yourself.

Answer:
“I am a Salesforce Developer with [X years] of experience specializing in developing scalable and efficient solutions on the Salesforce platform. I have expertise in Apex, Lightning Web Components, Visualforce, and integrations with REST and SOAP APIs. I’ve worked on automating business processes using Flows, Process Builder, and triggers while adhering to Salesforce best practices. My recent project involved implementing a Salesforce solution for Pfizer, where I customized objects like Healthcare Professional, Compliance Audit, and Sales Call to meet the business requirements.”

---

2. What are your strengths and weaknesses?

Answer:
“My strengths include my ability to analyze business requirements and deliver optimized Salesforce solutions using Apex and declarative tools. I excel in integrating Salesforce with external systems and automating workflows. My weakness is that I sometimes focus too much on technical details, but I’m actively working on balancing my technical focus with a broader project perspective.”

---

3. Why should we hire you?

Answer:
“You should hire me because I bring a blend of technical expertise and business understanding. I have a proven track record of developing robust Salesforce solutions that align with business goals. My ability to integrate Salesforce with external systems and implement security models makes me a strong candidate for this role.”

---

4. Where do you see yourself in five years?

Answer:
“In five years, I see myself taking on more responsibilities as a Salesforce Technical Architect. I aim to deepen my expertise in advanced Salesforce features like Einstein Analytics and lead strategic projects that deliver measurable value to the organization.”

---

5. Describe a challenging situation at work and how you handled it.

Answer:
“During the Pfizer project, we encountered a challenge with integrating an external HCP license validation API. The API frequently timed out due to large data volumes. I resolved this by implementing an asynchronous callout using Future methods, which ensured smooth API communication without impacting system performance. The client appreciated the solution as it met their compliance needs.”

---

6. Why are you leaving your current job?

Answer:
“I’m seeking opportunities to work on more complex Salesforce implementations and grow my skills in areas like advanced Lightning development and AI features. While I’ve gained valuable experience in my current role, I’m excited to take on new challenges and contribute to your organization’s success.”

---

7. How do you handle stress and pressure?

Answer:
“I handle stress by staying organized and prioritizing tasks effectively. For example, I use Agile methodologies to break down complex requirements into manageable sprints and track progress with tools like Jira. Open communication with stakeholders helps me address potential challenges early.”

---

8. Can you give an example of a time when you had to learn something quickly?

Answer:
“In a previous project, I had to quickly learn Salesforce CPQ as it was a key client requirement. I took Trailhead courses and practiced setting up product rules, pricing rules, and quote templates. Within two weeks, I successfully implemented a CPQ solution that streamlined the client’s sales process.”

---

9. How do you prioritize your work?

Answer:
"I prioritize work based on deadlines, business impact, and dependencies. I break tasks into smaller, actionable steps and use Agile practices to plan sprints. For example, in the Pfizer project, I prioritized tasks related to compliance audit tracking first because they were critical for regulatory approval."

---

10. What makes you unique?

Answer:
"My combination of strong technical skills and a focus on business outcomes makes me unique. For instance, during the Pfizer project, I not only delivered technical solutions but also ensured they aligned with the company's compliance and sales goals. My proactive communication and ability to understand business needs set me apart."

---

11. How do you stay current with industry trends?

Answer:
"I stay updated by completing Trailhead modules, attending Salesforce webinars, and participating in developer forums. I also follow Salesforce release notes and apply new features in my projects whenever relevant."

---

12. Describe a time when you had to work with a difficult team member.

Answer:
"I worked with a team member who was resistant to adopting Lightning Web Components. I organized a knowledge-sharing session to demonstrate its advantages and provided hands-on guidance during development. This collaborative approach helped the team member adapt, and we successfully delivered the project."

---

13. Why do you want to work here?

Answer:
"I'm impressed by your organization's innovative use of Salesforce and the impact it has on your industry. I'm excited about the opportunity to contribute my skills to challenging projects and grow within a company that values cutting-edge solutions."

---

14. What do you think you could improve in your work style?

Answer:
"I'm always looking to improve my time management skills. While I complete tasks on time, I'm working on better estimating time for complex tasks by breaking them into smaller milestones. This helps me set clearer expectations for stakeholders."

---

15. How would your previous coworkers describe you?

Answer:
"My coworkers would describe me as collaborative, reliable, and technically proficient. They often appreciated my ability to troubleshoot complex issues and my proactive approach to resolving challenges in team projects."

These answers are tailored for a Salesforce Developer role, emphasizing technical expertise, problem-solving, and collaboration.