Apex Language Overview

Apex is a strongly typed, object-oriented programming language that allows developers to execute flow and transaction control statements on the Salesforce platform server in conjunction with calls to the API.

Table of Contents

Introduction

Apex is a proprietary language developed by Salesforce.com. It is similar to Java and C# and is specially designed for building business logic within the Salesforce platform. Apex enables developers to add and interact with data in the Lightning Platform, and it can be used to create web services, write complex business processes, and much more.

Key Features

Feature	Description
Strongly Typed	Apex is a strongly typed language, requiring explicit declaration of variable types.
Object-Oriented	Supports classes, interfaces, and inheritance.
Data Manipulation	Apex provides built-in support for DML operations such as `insert`, `update`, `delete`, and `query` on Salesforce records.
Testing Framework	Apex includes a testing framework to write and execute tests that ensure code coverage and application stability.
Governor Limits	Enforces limits to ensure that Apex code does not monopolize shared resources.

Syntax and Structure

Apex syntax is similar to Java and uses dot notation to access object members. Classes, methods, and variables must be declared with explicit data types.

Classes and Methods

```
public class MyFirstClass {
    public String sayHello(String name) {
        return 'Hello, ' + name + '!';
```

```
    }
}
```

Variables and Data Types

```
Integer myInteger = 123;
String myString = 'OSF Digital';
List<Account> accounts = [SELECT Id, Name FROM Account LIMIT 10];
```

Sample Code

Below is a simple example of an Apex trigger that automatically populates the `Description` field of a new `Account` record with a custom message.

```
trigger AccountBeforeInsert on Account (before insert) {
    for (Account acc : Trigger.new) {
        acc.Description = 'This account was created on ' +
String.valueOf(Date.today()) + ' and is managed by OSF Digital.';
    }
}
```

Best Practices

- **Bulkify Your Code**: Ensure your code properly handles more than one record at a time.
- **Use Collections**: Leverage lists, sets, and maps to efficiently handle data.
- **Write Efficient SOQL Queries**: Avoid queries inside loops and limit the number of records returned.
- **Implement Error Handling**: Use try-catch blocks to gracefully handle exceptions.
- **Test Thoroughly**: Write comprehensive test methods to cover various scenarios and ensure at least 75% code coverage.

Resources

- Salesforce Developers Documentation
- Apex Developer Guide
- Trailhead Apex Modules

This markdown file serves as a brief introduction to the Apex programming language, its features, and some best practices for writing efficient Apex code.