Salesforce.PDII-JPN.v2026-02-02.q144

試験コード：	PDII-JPN
試験名称：	
認証ベンダー：	Salesforce
無料問題の数：	144
バージョン：	v2026-02-02
ページの閲覧量：	106
問題集の閲覧量：	1550

https://www.jpnshiken.com/shiken/Salesforce.PDII-JPN.v2026-02-02.q144.html

質問: **1**

テスト用のコントローラー拡張機能を作成するには、上記の単体テストのセットアップの指定された場所に何を追加する必要がありますか?

A. ApexPages.Standardcontroller sc = 新しい ApexPages。標準コントローラー (acct.Id) ; AccountControllerExt 拡張子 = 新しい AccountControllerExt(sc) ; ああ

B. AccountControllerExt 拡張機能 = 新しい AccountControllerExt (acct);

C. ApexPages.StandardController sc = new ApexPages.StandardController(acct); AccountcontrollerExt 拡張子 = 新しい AccountControllerExt(sc);

D. AccountControllerExt 拡張子 = 新しい AccountControllerExt (acct.Id) ;

正解: **(**正解を表示します**)**

ユニットテストでコントローラ拡張を作成する場合、正しい設定は、Accountレコードを使用して新しいApexPages.StandardControllerをインスタンス化し、それを使用してAccountControllerExtをインスタンス化することです。これにより、指定されたAccountレコードのコンテキストでコン

トローラ拡張が設定されます。参考資料 :Apex開発者ガイド - カスタムコントローラとコントローラ拡張のテスト

質問: **2**
開発者が外部Webサービスへの呼び出しを必要とするコードを記述しています。非同期メソッドで呼び出しを行う必要があるシナリオはどれですか？
A. コールアウトが完了するまでに 60 秒以上かかる可能性があります。
B. コールアウトは REST API を使用して行われます。
C. コールアウトは Apex トリガーで実行されます。
D. 1 回のトランザクションで 10 回を超えるコールアウトが行われます。
正解: **C (**コメントを発表する**)**
Salesforceでは、Apexトリガーから直接コールアウトを行うことはできません（オプションC）。これはプラットフォームの根本的なアーキテクチャ上の制約です。トリガーはデータベーストランザクションの一部として実行されます。トリガーが同期コールアウトを実行できる場合、データベーストランザクション（および関連するすべての行ロック）は、外部システムからの応答を待機している間、オープン状態のままになります。その結果、深刻なパフォーマンスボトルネックやトランザクションタイムアウトが発生する可能性があります。
トリガー イベントに基づいてコールアウトを実行するには、開発者は、コールアウト ロジックを @future(callout=true) メソッド、Queueable クラスなどの非同期コンテキストに移動するか、プラットフォーム イベントを公開する必要があります。
選択肢Aは不正解です。非同期メソッドはCPU時間を長く使用できますが、個々のコールアウトの最大タイムアウト（120秒）は延長されません。選択肢Bは不正解です。RESTは単なるプロトコルであり、同期性を規定するものではないためです。選択肢Dは不正解です。単一トランザクションにおけるコールアウトの上限は100であるため、同期コンテキストでは10回のコールアウトが許容されます（トリガー内ではない場合）。
したがって、トリガーの存在は、プログラム的に非同期アプローチを義務付ける唯一の要素です。

質問: **3**
企業は、商談のレコード タイプに基づいて異なるロジックを実行したいと考えています。
このリクエストを処理し、ベスト プラクティスに従っているコード セグメントはどれですか?

A.

```
RecordType recType_New = [SELECT Id, Name FROM
RecordType WHERE SObjectType = 'Opportunity' AND
IsActive = True And DeveloperName = 'New' LIMIT
1];

RecordType recType_Renewal = [SELECT Id, Name FROM
RecordType WHERE SObjectType = 'Opportunity' AND
IsActive = True And DeveloperName = 'Renewal'
LIMIT 1];

for(Opportunity o : Trigger.new) {
if(o.RecordTypeId == recType_New ) {
    // do some logic Record Type 1
    }
else if (o.RecordTypeId == recType_Renewal ) {
    // do some logic for Record Type 2
    }
}
```

B.

```
List<RecordType> recTypes = [SELECT Id, Name FROM
RecordType WHERE SObjectType = 'Opportunity' AND
```

```
chema.SObjectType.Opportunity.getRecordTypeInfosByDe

d recType_Renewal =
chema.SObjectType.Opportunity.getRecordTypeInfosByDe

or(Opportunity o : Trigger.new) {
  if(o.RecordTypeId == recType_New ) {
  // do some logic Record Type 1
  }
lse if (o.RecordTypeId == recType_Renewal ) {
  // do some logic for Record Type 2
  }
```

C.

D.

```
for(Opportunity o : Trigger.new) {
   if(o.RecordType.Name == 'New') {
   // do some logic Record Type 1
   }
```

正解: **C** (コメントを発表する)

このリクエストを適切に処理し、ベストプラクティスに従うには、オプションCを使用するのが適切です。オプションCでは、サブクエリを含むSOQLクエリを使用して、レコードタイプで商談をフィルタリングし、その結果を反復処理します。これにより、開発者は環境や言語によって変化する可能性のあるレコードタイプ名やIDをハードコーディングする必要がなくなり、商談とレコードタイプ間のリレーション名を使用してレコードタイプ名にアクセスできます。また、開発者はレコードタイプの条件に一致する商談のみをクエリするため、ループ内でレコードタイプ名を確認する必要がなくなり、SOQLクエリの数と反復処理の回数も削減されます。オプションCは、バインド変数の使用、不要なフィールドの回避、選択的フィルタの使用など、SOQLクエリ記述のベストプラクティスにも従っています12。参考文献 :[SOQLおよびSOSLクエリ]、[SOQLリレーションシップ]、[SOQLベストプラクティス]

質問: **4**

Aura コンポーネントから呼び出され、クラスにラップされたデータを返す以下のコントローラーコードを考えてみましょう。

```
public class myServerSideController {
    @AuraEnabled
    public static MyDataWrapper getSomeData( String theType ) {
        Some_Object__c someObj = [
                SELECT ID, Name
                FROM Some_Object__c
                WHERE Type__c = :theType
                LIMIT 1
        ];

        Another_Object__c anotherObj = [
                SELECT ID, Option__c
                FROM Another_Object__c
                WHERE Some_Object__c = :someObj.Name
                LIMIT 1
        ];

        MyDataWrapper theData = new MyDataWrapper();

        theData.Name = someObj.Name;
        theData.Option = anotherObj.Option__c;
        return theData;
    }
```


開発者は、クエリがそれぞれ 1 つのレコードを返し、Aura コンポーネントにエラー処理があることを確認しましたが、コントローラー getSemeData を呼び出したときにコンポーネントが何も返さないことを確認しました。
'なにが問題ですか？
A. MyDatsWrapper などの Apex クラスのインスタンスを Lightning コンポーネントに返すことはできません。
B. メンバーの名前とオプションにはゲッターとセッターを含めることはできません。
C. メンバーの名前とオプションは公開宣言されるべきではありません。
D. メンバーのカメとクラス MyDataWrapper のオプションにも @AuraEnabled のアノテーションを付ける必要があります。
正解: **(**正解を表示します**)**
Auraコンポーネントにデータを渡すには、コンポーネントからアクセスされるApexクラスのすべてのプロパティに@AuraEnabledアノテーションを付与する必要があります。このアノテーションがないと、Auraフレームワークはプロパティをシリアル化してフロントエンドに送信できません。

参考資料: Aura コンポーネント開発者ガイド - AuraEnabled アノテーション

質問: **5**
企業は REST Web サービスを公開しており、Salesforce と REST Web サービスの間に双方向 SSL を確立したいと考えています。適切な認証局によって署名された証明書が開発者に提供されています。
Salesforce 側で必要な変更は何ですか? 2つ選んでください。)
A. HttpRequest.setClientCertificateName() を使用するようにコードを更新します。
B. HttpRequest.setHeaderQ を使用して Authorization ヘッダーを設定するようにコードを更新します。
C. 提供された証明書を使用して 2 要素認証を構成する
D. 証明書とキーの管理で証明書のエントリを作成します。
正解: **A,D (**コメントを発表する**)**

質問: **6**
Universal Containers は、Salesforce のソース駆動開発アプローチに従う開発チームを率いています。継続的インテグレーションおよびデリバリー (CL/CD) プロセスの一環として、サンドボックスや実稼働環境を含む複数の環境に変更を自動的にデプロイする必要があります。
「ソース駆動開発における CI/CD パイプラインを最もよくサポートするメカニズムまたはツールはどれですか?
A. Salesforce CLI と Salesforce DX
B. Visual Studio Code 用の Salesforce 拡張機能
C. 変更セット
D. Ant 移行ツール
正解: **(**正解を表示します**)**
Salesforce DX と連携した Salesforce CLI は、ソース駆動開発をサポートし、継続的インテグレーションおよびデリバリー (CI/CD) プロセス向けに設計されています。これにより、開発者はサンドボックス環境や本番環境を含む複数の環境へのデプロイを自動化できます。参考資料: Salesforce DX 開発者ガイド

質問: **7**
キューに入れられた apex ジョブの進行状況は System.enqueueJob メソッドを使用しており、監視する必要があります。
どのオプションが有効ですか? 2つ選んでください。)
A. 設定で Apex ジョブ ページを使用します。
B. AsyncApexJob レコードのクエリ
C. セットアップで [スケジュールされたジョブ] ページを使用します
D. Queueable Apex レコードのクエリ
正解: **(**正解を表示します**)**

質問: **8**
セーブポイントに関して正しい記述はどれですか?
A. 任意の順序で作成された任意のセーブポイント変数にロールバックできます。
B. 静的変数はロールバック中に元に戻されません。
C. セーブポイントは、DML ステートメント ガバナーの制限によって制限されません。
D. セーブポイントへの参照はトリガー呼び出しをまたぐことができます。
正解: **(**正解を表示します**)**
Salesforce Apex では、Database.Savepoint を使用することで、開発者はトランザクション内の特定のポイントを定義し、エラー発生時にそのポイントに戻ることで、データベースへの変更を事実上元に戻すことができます。ただし、ロールバックの影響を受けるものと受けないものを理解することが重要です。プラットフォーム仕様によると、データベースレコード (DML 操作) はセーブポイント設定時の状態に戻りますが、静的変数は元に戻りません (オプション B)。静的変数はロールバックの有無にかかわらずトランザクション全体にわたって保持されるため、トリガーが既に起動されたかどうかを追跡するための「再帰ガード」としてよく使用されます。
その他のオプションは、セーブポイントの厳密な「スタック」の性質上、正しくありません。セーブポイントは時系列の逆順にロールバックする必要があります。以前のセーブポイントにロールバックすると、それ以降に作成されたセーブポイントは無効になります (オプションA)。さらに、セーブポイント変数は、それが作成された特定の実行コンテキスト内でのみ有効です。セーブポイント参照を、異なるトリガー呼び出しや非同期境界を越えて渡すことはできません (オプションD)。最後に、ロールバック自体はDMLステートメントではありませんが、セーブポイントの設定からロールバックまでの間に実行される操作は、DMLおよびCPUガバナーの制限にカウントされます (オプションC)。
複雑なトランザクション ロジックをデバッグするには、データベースのロールバック中にメモリ状態 (静的変数) が変更されないことを理解することが大切です。

質問: **9**
企業のサポート プロセスでは、「修正できませんでした」というステータスでケースがクローズされるたびに、エンジニアリング レビューのカスタム オブジェクト レコードを作成し、ケース、連絡先、およびそれに関連付けられた製品からの情報を入力する必要があります。場合。Apex トリガを使用してこれを自動化する正しい方法は何ですか?
A. Engineering Review レコードを作成して挿入する更新前トリガー
B. Engineering Review レコードを作成して挿入する after update トリガー
C. Engineering Review レコードを作成して挿入する更新前トリガー
D. Engineering Review レコードを作成して挿入する after upsert トリガー
正解: **B (**コメントを発表する**)**

質問: **10**
ユニバーサル コンテナ (UC) に関する次の情報を考慮します。
* UC は顧客を Salesforce のアカウントとして表します。

* すべての顧客は、UC のシステム全体で一意の Customer__Number_c を持っています。
* UC には、外部システムから送信される請求書を表す、アカウントへのルックアップを備えたカスタムの Invoice c オブジェクトもあります。
UC は、請求書データを Salesforce に統合して、営業担当者が顧客がいつ期日通りに請求書を支払うかを確認できるようにしたいと考えています。
これを実装する最適な方法は何ですか?
A. 顧客番号が外部 ID であり、カスタム フィールドの請求書番号が外部 ID であることを確認し、請求書データを毎晩更新/挿入します。
B. Salesforce Connect と外部データ オブジェクトを使用して、カスタム コードを使用せずに請求書データを Salesforce にシームレスにインポートします。
C. 各顧客の Salesforce アカウント ID を使用してカスタム請求システムに相互参照テーブルを作成し、毎晩請求書データを挿入します。
D. 呼び出しごとに Account オブジェクトをクエリして請求書データを挿入し、請求書の顧客番号に対応する Salesforce ID を取得します。
正解: **(**正解を表示します**)**
この要件を実装する最適な方法は、カスタム請求システムに各顧客の Salesforce アカウント ID との相互参照テーブルを作成し、請求書データを毎晩挿入することです。このようにして、開発者は Salesforce アカウント ID を外部キーとして使用して、請求書データを Salesforce 内の正しいアカウント レコードに関連付け、各請求書レコードのアカウント オブジェクトをクエリする必要がなくなります。これにより、クエリの数と顧客番号項目への依存が減るため、データ統合のパフォーマンスと信頼性が向上します。顧客番号が外部 ID であり、カスタム項目の請求書番号が外部 ID であることを確認し、請求書データを毎晩 upsert することは最適ではありません。顧客番号項目と一致させるために、各請求書レコードのアカウント オブジェクトをクエリする必要があるためです。Salesforce Connect と外部データ オブジェクトを使用して、カスタム コードを使用せずに請求書データを Salesforce にシームレスにインポートすることは最適ではありません。外部システムで請求書データを OData サービスとして公開する必要があり、Salesforce Connect を使用するための追加コストも発生するためです。請求書データを挿入するたびにアカウントオブジェクトをクエリし、請求書の顧客番号に対応するSalesforce IDを取得する方法は、特に顧客番号フィールドが一意でなかったりインデックス化されていない場合、非効率でエラーが発生しやすいため、最適ではありません。参考 :[Salesforce Connect: データ統合プラットフォーム]、[外部IDフィールド]

質問**: 11**
コードの重複を避け、保守性を向上させるために、Universal Containers はコード再利用のための API 統合をどのように実装する必要がありますか?
A. API 統合用の再利用可能な Apex クラスを作成し、関連する Apex クラスから呼び出します。
B. 統合ロジックをカプセル化するには、API エンドポイントごとに個別の Apex クラスを使用します。
C. API 統合コードを静的リソースとして保存し、各 Apex クラスで参照します。

D. API 統合コードを、それを必要とする各 Apex クラスに直接含めます。
正解: **(**正解を表示します**)**
包括的かつ詳細な説明:1
Apex開発におけるDRY（Don't Repeat Yourself：同じことを繰り返さない）の基本原則は、複数の場所で使用されるロジックを一元管理することです。API連携の場合、これは通常、ユーティリティクラスまたはサービスクラス（オプションA）の作成を伴います。
このクラスは、統合の共通の「配管」を処理します。
* 名前付き資格情報を取得します。
* ヘッダーの設定 (Content-Type、Timeout)。
* エラー処理とログ記録の標準化。
* JSON ペイロードのシリアル化とデシリアル化。
この中心的なクラスを様々なトリガ、バッチジョブ、コントローラから呼び出すことで、開発者はAPIへの変更（バージョン更新やヘッダー変更など）を1か所でのみ実行できるようになります。オプションBは「クラスの無秩序な拡散」と定型コードの重複につながります。オプションCは、静的リソースに実行可能なApexコードを含めることができないため、誤りです。オプションDは、保守性が低いことを意味します。

質問: **12**
次のコード スニペットを考えてみましょう。

```
<c-selected-order>
<template for:each={orders.data} for:item="order">
    <c-order orderId={order.Id}></c-order>
</template>
</c-selected-order>
```


<e-orders> コンポーネントは、注文がユーザーによって選択されたことを <c-selected-orders コンポーネントにどのように伝達する必要がありますか?
A. コンポーネント イベントを作成して起動します。
B. アプリケーション イベントを作成して起動します。
C. 標準 DOM イベントを作成して起動します。
D. カスタム イベントを作成して送信します。
正解: **(**正解を表示します**)**
カスタムイベントを作成してディスパッチします。このアプローチにより、LWCコンポーネントはコンポーネントのカプセル化と再利用性を維持しながら、効率的に通信できます。参考資料：Lightning Webコンポーネント間の通信

質問: **13**
開発者がテストクラス内から組織データにアクセスしようとしています。

テストクラスに (seeAllData=true) アノテーションが必要なのはどの sObject タイプですか?
A. ユーザー
B. レコードタイプ
C. レポート
D. プロフィール
正解: **(**正解を表示します**)**
seeAllData=true アノテーションは、テストクラスが組織内のすべてのデータにアクセスできるようにするために使用されます。テストクラス内で作成されたテストデータやテスト設定メソッドを使用して作成されたテストデータを使用する代わりに、このアノテーションは慎重に使用し、絶対に必要な場合にのみ使用してください。テストクラスが組織内のデータに依存するようになり、予期しないエラーや信頼性の低い結果が発生する可能性があるためです。seeAllData=true アノテーションは、レポート、ダッシュボード、ドキュメント、添付ファイルなど、テストデータの作成をサポートしていない一部の sObject 型で必須です。したがって、テストクラスがレポート sObject にアクセスする必要がある場合は、seeAllData=true アノテーションを使用する必要があります。参考資料: [単体テストにおけるテストデータと組織データの分離]、[seeAllData アノテーションの使用]

質問: **14**
開発者は、注文品目の数量、単価、および合計を表示する Lightning Web コンポーネントを構築しています。合計は、数量に単価を掛けて動的に計算されます。

JavaScript:
```
import { LightningElement } from 'lwc';
export default class OrderLineItem extends LightningElement {
    @api quantity;
    @api unitPrice;
}
```

Template Markup:
```
<template>
    <div>
        Quantity: {quantity} <br />
        Unit Price: {unitPrice} <br />
    </div>
</template>
```


合計を表示するには何を追加する必要がありますか?
A. calculate Total() {return quantity * unitPrice;} を javaScript に追加し、Total : {calculate Total()} をテンプレートに追加します。
B. get total() { return quantity * unitPrice;} を JavaScript に追加し、Total: {total} をテンプレートに追加します。

C. 合計を追加します。{複数の数量、y 単価)} テンプレートで。
D. テンプレートに Total: {quantity * UnitPrice} を追加します。
正解: **(**正解を表示します**)**

質問: **15**
メッセージがページにレンダリングされています。メッセージを表示するには、どのコンポーネントを Visualforce ページに追加する必要がありますか?

A.
```
<apex:message for="info"/>
```

B.
```
<apex:pageMessage severity="info" />
```

C.
```
<apex:facet name="messages" />
```

D.
```
<apex:pageMessages />
```

正解: **(**正解を表示します**)**

質問: **16**
ビジネスルールでは、新しいアカウントが作成されると必ず連絡先も作成されるように設定されています。連絡先の作成に失敗した場合、アカウントが作成されないようにするためのカスタム画面を開発する際に、どのような機能を使用できますか？
A. try-catch ブロックで setSavePoint() と rollback() を使用します。
B. try-catch ブロックで Database Savepoint メソッドを使用します。
C. allOrNone を false に設定して Database.Insert メソッドを使用します。
D. 連絡先の挿入に失敗した場合は、Database.Delete メソッドを使用します。
正解: **A (**コメントを発表する**)**
この要件では「トランザクションのアトミック性」が求められます。つまり、両方のデータベース操作（アカウント作成と連絡先作成）が成功するか、どちらもデータベースにコミットされないかのいずれかです。Apexでは、セーブポイントで管理されていない限り、各DML文は通常、独立したトランザクションとして機能します。
正しいアプローチは、try-catchブロック内でDatabase.setSavepoint()とDatabase.rollback()を使用することです（オプションA）。開発者は、Accountが挿入される直前にセーブポイントを設定します。Accountは正常に作成されたものの、その後のContactの挿入が（検証ルール、トリガーエラー、

またはシステム例外によって）失敗した場合、コードはcatchブロックに入ります。catchブロック内で、開発者はDatabaseを実行します。
rollback(sp) は、アカウントが挿入される前の状態にデータベースを戻します。
オプションBは技術的には似ていますが、「A」は標準的なプログラムパターン名を提供します。オプションC（allOrNone=false）は、1回のDML呼び出しで単一のレコードリストにのみ適用され、取引先と取引先の成功を関連付けることはできません。オプションD（手動削除は信頼性の低い「クリーンアップ」戦略であり、システムがクラッシュした場合や、最初の挿入による二次的な副作用が発生した場合には失敗します。セーブポイントを使用することで、プラットフォームがロールバックを安全かつ完全に処理することが保証されます。

質問: **17**
Salesforce 管理者と Cloud Kicks は、米国内のすべての郵便番号とその郵便番号が属する Cloud Kicks 販売地域を保存するために、Region_c というカスタム オブジェクトを作成しました。

Cloud Kicks は、リードの郵便番号に基づいてリージョンを設定するためのリード上のトリガーを必要としています。
この要求を満たす最も効率的な方法はどのコード セグメントですか?

A.

```
Set<String> zips = new Set<String>();
for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          zips.add(l.PostalCode);
     }
}

for(Lead l : Trigger.new) {
     List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips];
     for(Region__c r :regions) {
          if(l.PostalCode == r.Zip_Code__c) {
               l.Region__c = r.Region_Name__c;
          }
     }
          }
     }
}
```

B.

```
Set<String> zips = new Set<String>();
for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          zips.add(l.PostalCode);
     }
}

List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips];

for(Lead l : Trigger.new) {
     for(Region__c r :regions) {
          if(l.PostalCode == r.Zip_Code__c) {
               l.Region__c = r.Region_Name__c;
          }
     }
}
```

C.

```
for(Lead l : Trigger.new) {
     Region__c reg = [SELECT Region_Name__c FROM Region__c WHERE Zip_Code__c = :l.PostalCode];
     l.Region__c = reg.Region_Name__c;
}
```

D.

```
Set<String> zips = new Set<String>();
for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          zips.add(l.PostalCode);
     }
}

List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips];

Map<String, String> zipMap = new Map<String, String>();
for(Region__c r :regions) {
     zipMap.put(r.Zip_Code__c, r.Region_Name__c);
}

for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          l.Region__c = zipMap.get(l.PostalCode);
     }
}
```


正解: **(**正解を表示します**)**

郵便番号に基づいてリードのRegion項目にデータを入力する最も効率的な方法は、Mapを使用して、Zip_Code__c項目をキーとするRegion__cレコードを保存することです。この方法により、開発者は各リードレコードに対してRegion__cオブジェクトをクエリする（SOQL制限例外が発生する）ことを回避できます。代わりに、開発者はRegion__cオブジェクトを一度クエリし、その結果をMapに保存します。その後、開発者はリードレコードをループ処理し、Mapを使用してZip_Code__c項目に基づいて対応するRegion__cレコードを取得します。このアプローチにより、SOQLクエリの回数が削減され、トリガーのパフォーマンスが向上します。

質問: **18**

以下の Aura コンポーネントを参照してください。

Component markup:

```
<aura:component>
    <aura:attribute name="contactInfo" type="Object"/>
    <aura:attribute name="showContactInfo" type="boolean" default="true"/>
    <aura:handler name="init" value="{!this}" action="{!c.init}"/>

    <!--- ... other code .... -->
    <aura:if isTrue="{!v.showContactInfo}">
        <c:contactInfo value="{!v.contactInfo}"/>
    </aura:if>
</aura:component>
```

Controller JS:

```
({
    init: function(cmp, helper) {
        // ...other code ...
        var show = helper.getShowContactInfo();
        cmp.set("v.showContactInfo", show);
    },
   // other code...
})
```


開発者は、コンポーネントの読み込みが遅いという苦情を受け取りました。
コンポーネントの動作を高速化するために、開発者はどの変更を実装できますか?

A. showContactInfo のデフォルトを「False」に変更します。

B. <c: contactInfo の内容をコンポーネントに移動します。

C. contactInfo の種類を「マップ」に変更します。

D. showContactInfo の変更イベント ハンドラーを追加します。

正解: **(**正解を表示します**)**

質問: **19**

開発者は、サードパーティの JavaScript フレームワークを使用する Visualforce ページを作成しました。開発者は、JavaScript Remoting for Apex Controllers を使用して JavaScript 関数にデータを提供することにしました。

Apex でリモートメソッドを宣言する正しい構文は何ですか? (2つ選んでください。)

A. @RemoteAction グローバル文字列 getTable()

B. @RemoteAction パブリック静的文字列 getTable()

C. @RemoteAction グローバル静的文字列 getTable()

D. @RemoteObject グローバル静的文字列 getTableQ

正解: **(**正解を表示します**)**

質問: **20**

Aura コンポーネントから呼び出され、クラスにラップされたデータを返す以下のコントローラーコードを考えてみましょう。

```
public class myServerSideController {
    @AuraEnabled
    public static MyDataWrapper getSomeData( String theType ) {
        Some_Object__c someObj = [
                SELECT ID, Name
                FROM Some_Object__c
                WHERE Type__c = :theType
                LIMIT 1
        ];

        Another_Object__c anotherObj = [
                SELECT ID, Option__c
                FROM Another_Object__c
                WHERE Some_Object__c = :someObj.Name
                LIMIT 1
        ];

        MyDataWrapper theData = new MyDataWrapper();

        theData.Name = someObj.Name;
        theData.Option = anotherObj.Option__c;
        return theData;
    }
```


開発者は、クエリがそれぞれ 1 つのレコードを返し、Aura コンポーネントにエラー処理があることを確認しましたが、コントローラー getSemeData を呼び出したときにコンポーネントが何も返さないことを確認しました。
'なにが問題ですか？

A. MyDatsWrapper などの Apex クラスのインスタンスを Lightning コンポーネントに返すことはできません。

B. メンバーの名前とオプションにはゲッターとセッターを含めることはできません。

C. メンバーの名前とオプションは公開宣言されるべきではありません。

D. メンバーのカメとクラス MyDataWrapper のオプションにも @AuraEnabled のアノテーションを付ける必要があります。

正解: **D** (コメントを発表する)

Auraコンポーネントにデータを渡すには、コンポーネントからアクセスされるApexクラスのすべてのプロパティに@AuraEnabledアノテーションを付与する必要があります。このアノテーショ

ンがないと、Auraフレームワークはプロパティをシリアル化してフロントエンドに送信できません。参考資料 :Auraコンポーネント開発者ガイド - AuraEnabledアノテーション

質問: **21**
開発者は、カスタムアクションで Aura コンポーネントを使用したいと考えています。
これを行うために何を考慮する必要がありますか？
A. コンポーネントは force:lightningQuickActionButton インターフェースを実装する必要があります。
B. 必須としてマークされた各コンポーネント属性にデフォルト値を指定する必要があります。
C. コンポーネントの JavaScript コントローラーは、初期化時にメソッドを処理する必要があります。
D. クラス「slds-modal__container」をコンポーネントの最上位要素に追加する必要があります。
正解: **A (**コメントを発表する**)**

質問: **22**
開発者は、特定の取引先レコードが Visualforce コントローラのテスト クラスでテストされていることをどのように確認する必要がありますか?
A. アカウントをテスト クラスに挿入し、テスト クラスでページ参照をインスタンス化してから、System.currentPageReference().getParameters{}.put() を使用してアカウント ID を設定します。
B. テスト クラスでページ参照をインスタンス化し、アカウントをテスト クラスに挿入してから、System.setParentRecordId().get() を使用してアカウント ID を設定します。
C. テスト クラスでページ参照をインスタンス化し、アカウントをテスト クラスに挿入してから、seeAHData-true を使用してアカウントを表示します。
D. アカウントを Salesforce に挿入し、テスト クラスでページ参照をインスタンス化してから、System.setParentRecordId().get() を使用してアカウント ID を設定します。
正解: **(**正解を表示します**)**

質問: **23**
ある会社は、Salesforce を使用して製品を顧客に販売しています。また、製品の記録システムである外部製品情報管理 (PIM) システムも備えています。
開発者は次の要件を受け取りました。
* PIM で商品が作成または更新されるたびに、商品は Salesforce で Product2 レコードとして作成または更新され、PricebookEntry レコードは Salesforce によって自動的に作成または更新される必要があります。
* PricebookEntry は、カスタム設定で指定された Pricebook2 で作成する必要があります。
これらの要件を満たすために、開発者は何を使用する必要がありますか?
A. SObJect ツリー
B. 呼び出し可能なアクション

C. 頂点 REST
D. イベント監視
正解: **(**正解を表示します**)**

質問: **24**
開発者は、どのプロファイルとユーザーがどのシークレットにアクセスできるかを指定する機能を備えたシークレット データを格納する方法を見つけるよう求められます。
このデータを保存するために何を使用する必要がありますか?
A. カスタム メタデータ
B. System.Cookie クラス
C. 静的リソース
D. カスタム設定
正解: **(**正解を表示します**)**

質問: **25**
開発者は、開発者サンドボックスで Visualforce ページを作成してテストしましたが、本番環境で使用するときにビューステートエラーが発生したというレポートを受け取りました。
開発者はこれらのエラーを修正するために何を確認する必要がありますか?
A. クエリがガバナ制限を正味で超えていないことを確認します。
B. プロパティがプライベートとしてマークされていることを確認します。
C. 変数が一時的としてマークされていることを確認します。
D. プロファイルが Visualforce ページにアクセスできることを確認します。
正解: **(**正解を表示します**)**
Visualforceのビューステートは、コンポーネント、項目値、コントローラの状態など、ページの状態を表します。このエラーは通常、ページのビューステートが大きくなりすぎたときに発生します。この問題を解決するには、以下の手順を実行してください。
C: 変数が transient としてマークされていることを確認する: Apex の transient キーワードを使用すると、ビューステートの一部ではなく、保存されないインスタンス変数を宣言できます。これは、ポストバック間で維持する必要のない大規模なデータの場合に特に便利です。
プロパティを非公開に設定する (オプションB) か、クエリがガバナ制限を超えないようにする (オプションA) ことはベストプラクティスかもしれませんが、ビューステートのサイズに直接影響を与えるものではありません。プロファイルがVisualforceページにアクセスできるようにする (オプションD) ことは、可視性とアクセスの問題であり、ビューステートのエラーとは関係ありません。
参考文献:
ビューステートに関する Salesforce 開発者ドキュメント: Visualforce ビューステート

質問: **26**
開発者は JavaScript の降格から Lightning Web コンポーネントに機能を移行し、既存の getOpportunities() メソッドを使用してデータを提供したいと考えています。
メソッドのどの変更が必要ですか?

A. メソッドは (cacheable=true) で装飾する必要があります。
B. メソッドは、シリアル化された JSON 配列の文字列を返す必要があります。
C. メソッドは JSON オブジェクトを返す必要があります。
D. A メソッドは @AuraEnabled で装飾する必要があります。
正解: **D (**コメントを発表する**)**

質問: **27**
Dynamic Apex について誤っているのはどれですか?
A. .getSObjectTypeQ を呼び出して、ID から sObject タイプを取得できます。
B. getDescribe() は、特定のオブジェクト/フィールドに関するさまざまな情報を取得できます
C. Schema.getGlobalDescribeQ は、すべての sObject のマップを提供します
D. 動的 SOQL では、バインド変数とバインド変数項目を使用できます
正解: **D (**コメントを発表する**)**
説明
説明/参照:
動的SOQLでは単純なバインド変数を使用できますが、バインド変数フィールドは使用できません。
(例:myVariable.field1_c)
SOQLインジェクションを防ぐにはescapeSingleQuotesを使用する

質問: **28**
数百万のアカウントが四半期ごとに外部システムから更新されています。Salesforceでこれらを更新する最適な方法は何でしょうか？
A. 複合REST API
B. SOAP API
C. バルクAPI
D. Apex REST Web サービス
正解: **C (**コメントを発表する**)**
包括的かつ詳細な説明：
数十万から数百万のレコードに及ぶ大規模なデータ ボリューム (LDV) を処理する場合、Bulk API (オプション C) が唯一の最適な選択肢です。
Bulk API は REST 原則に基づいていますが、大量のデータセットを非同期的に処理するように最適化されています。
Bulk API を使用すると、レコードを 1 つずつ、または小さなチャンク単位で同期的に処理する（時間制限に達し、多くの API 呼び出しを消費する）代わりに、CSV ファイルまたは JSON バッチをアップロードできます。Salesforce は、これらのレコードを並列処理を使用してバックグラウンドで処理します。
オプションAとB（REST/SOAP）は、限られた数のレコードに対するリアルタイムの同期インタラクション向けに設計されており、数百万レコードに達するとレート制限またはタイムアウトが発

生します。オプションDはカスタムロジック向けであり、標準RESTと同じ同期実行制限が適用されます。

質問: **29**
Visualforce リモート オブジェクトに対する JavaScript リモート処理の利点は何ですか?
A. JavaScript コードは必要ありません。
B. 複雑なサーバー側アプリケーション ロジックをサポート
C. Apex コードは必要ありません。
D. 指定された再レンダリング ターゲットを許可します。
正解: **(正解を表示します)**
VisualforceリモートオブジェクトとJavaScript Remotingの利点は、複雑なサーバサイドアプリケーションロジックをサポートできることです。JavaScript Remotingは、VisualforceページがJavaScriptを使用してVisualforceコントローラを必要とせずにApexメソッドを直接呼び出すことを可能にする技術です。Visualforceリモートオブジェクトは、VisualforceページがJavaScriptを使用してApexコードを必要とせずにSalesforce内のレコードに直接アクセスすることを可能にする機能です。しかし、JavaScript Remotingには、サーバサイドで複雑なビジネスロジックを実行できること、Apexアノテーションを使用してメソッドのセキュリティと共有を制御できること、カスタムデータ型やコレクションを返すことができることなど、Visualforceリモートオブジェクトにはない利点がいくつかあります12。参考 :ApexコントローラのJavaScript Remoting、Visualforceリモートオブジェクト

質問: **30**
バルク API_____。
A. アカウント、見込み客、カスタム オブジェクトなどのレコードを作成、取得、更新、または削除するために使用され、パスワードの維持、検索の実行などを可能にします。
B. REST 原則に基づいており、大量のデータ セットの読み込みまたは削除用に最適化されています。これを使用して、バッチを送信することにより、多くのレコードを非同期でクエリ、queryAll、挿入、更新、アップサート、または削除できます
C. Salesforce と対話するための、強力で便利でシンプルな REST ベースの Web サービス インターフェイスを提供します。その利点には、統合と開発の容易さが含まれ、モバイル アプリケーションや Web プロジェクトで使用するための優れたテクノロジの選択肢です。
D. 組織のカスタマイズを取得、リリース、作成、更新、または削除するために使用されます。最も一般的な用途は、変更を Sandbox またはテスト組織から本番環境に移行することです。
正解: **(正解を表示します)**

質問: **31**
Apex トリガと Apex クラスは、ケースが変更されるたびにカウンタ Edit_Count_c を増やします。

```
public class CaseTriggerHandler {
    public static void handle(List<Case> cases) {
        for (Case c : cases) {
            c.Edit_Count__c = c.Edit_Count__c + 1;
        }
    }
}

trigger on Case(before update) {
    CaseTriggerHandler.handle(Trigger.new);
}
```

ケース オブジェクトの新しいプロセスは、ケースが作成または更新されたときに実稼働環境で作成されたばかりです。< プロセスが作成されてから、ケースの編集時にカウントが 1 を超えてインクリメントされていることが報告されています。

問題を解決する Apex コードの変更はどれですか?

A.

```
public class CaseTriggerHandler {
    public static Boolean firstRun = true;
    public static void handle(List<Case> cases) {
        for (Case c : cases) {
            c.Edit_Count__c = c.Edit_Count__c + 1;
        }
    }
}

trigger on Case(before update) {
    CaseTriggerHandler.firstRun = true;
    if (CaseTriggerHandler.firstRun) {
        CaseTriggerHandler.handle(Trigger.newMap);
    }
    CaseTriggerHandler.firstRun = false;
}
```

B.

```
public class CaseTriggerHandler {
    Boolean firstRun = true;
    public static void handle(List<Case> cases) {
        if (firstRun) {
```

C.

```
trigger on Case(before update) {
    Boolean firstRun = true;
    if (firstRun) {
        CaseTriggerHandler.handle(Trigger.newMap);
    }
    firstRun = false;
}
```

D.

```
public class CaseTriggerHandler {
    public static Boolean firstRun = true;
    public static void handle(List<Case> cases) {
        for (Case c : cases) {
            c.Edit_Count__c = c.Edit_Count__c + 1;
        }
    }
}

trigger on Case(before update) {
    if (CaseTriggerHandler.firstRun) {
        CaseTriggerHandler.handle(Trigger.new);
```

正解: (正解を表示します)

質問: **32**
以下の Lightning コンポーネントを参照してください。
Lightning コンポーネントを使用すると、ユーザはボタンをクリックして変更を保存し、別のページにリダイレクトできます。現在、ユーザーが 「保存」ボタンをクリックすると、レコードは保存されますが、リダイレクトされません。
開発者が JavaScript をデバッグするために使用できる 3 つのテクニックはどれですか?
3 つの答えを選択してください
A. JavaScript で console.log () メッセージを使用します。
B. ユーザの Lightning コンポーネントのデバッグモードを有効にします。
C. ブラウザの開発ツールを使用して JavaScript をデバッグします。
D. 開発者コンソールを使用してデバッグ ログを表示します。
E. 開発者コンソールを使用してチェックポイントを表示します。
正解: **(**正解を表示します**)**

質問: **33**
次のコード スニペットを考えてみましょう。

```
public static List<Account> getAccounts(Date thisDate, Id goldenRT){
List<Account> accountList = [Select Id, Name, Industry FROM Account WHERE CreatedDate = :thisDate OR RecordTypeId = :goldenRT];
return accountList;
}
```

Apex メソッドはアカウントのデータ量が多い環境で実行されており、クエリのパフォーマンスが低下しています。
結果セット全体を保持しながらクエリを最適に実行するには、開発者はどの手法を実装する必要がありますか?
A. createdDate と RecordType の値を組み合わせる数式フィールドを作成し、数式に基づいてフィルターします。
B. クエリを 2 つの個別のクエリに分割し、2 つの結果セットを結合します。
C. メソッドに @Future アノテーションを付けます
D. データベースの queryLocator メソッドを使用してアカウントを取得します。
正解: **D (**コメントを発表する**)**
Salesforce で大量のデータを処理し、クエリの最適なパフォーマンスを確保するには、データ量を効率的に処理し、完全な結果セットを取得するアプローチを選択することが重要です。

* 選択肢Dは正解です。Database.QueryLocatorメソッドの使用は、大量のデータを処理する際のベストプラクティスです。このメソッドは通常、レコードを一括処理できるApexバッチ処理と組み合わせて使用されるため、ガバナ制限に達する可能性が低くなります。このメソッドは、通常のSOQLクエリの制限を超えてしまうような、非常に大規模なデータセットを処理できるように設計されています。
* 選択肢Aは不正解です。数式フィールドを作成してもクエリのパフォーマンスは向上しません。既存のデータを結合する新しいフィールドが作成されるだけで、クエリの実行が本質的に最適化されるわけではありません。
* 選択肢Bは不正解です。クエリを2つの部分に分割してApexで結合すると、効率が低下する可能性があり、結合された結果セットを管理するための追加コードが必要になる可能性があります。このアプローチでは、大量のデータ処理用に設計されたSalesforceの組み込み機能が活用されません。
* オプション C は不正解です。@Future アノテーションによりメソッドは非同期的に実行されますが、クエリのパフォーマンスや大量のデータ管理には役立ちません。
秒:
Salesforce のドキュメント: 非常に大きな SOQL クエリの操作 Salesforce のドキュメント: 非常に大きな SOQL クエリの操作 Batch Apex の使用: Batch Apex の使用

質問: **34**
Aura コンポーネントから呼び出される以下の Apex コントローラについて考えてみましょう。

```
Line 1 public class AttributeTypes
Line 2 {
Line 3     private final String[] arrayItems;
Line 4
Line 5     @AuraEnabled
Line 6     public List<String> getStringArray() {
Line 7         String[] arrayItems = new String[]{ 'red', 'green',
'blue' };
Line 8         return arrayItems;
Line 9     }
Line 10 }
```


このコードのどこが間違っているのでしょうか?
A. 1 行目: クラスはグローバルである必要があります
8. 1 行目と 6 行目: クラスとメソッドはグローバルである必要があります
B. 8 行目: メソッドは返す前に、まずリストを JSON にシリアル化する必要があります。
C. 6 行目: メソッドは静的でなければなりません
正解: **C** (コメントを発表する)

質問: **35**
ユニバーサル ･コンテナーズは、新入社員の面接プロセスを管理するため、Salesforce を使用することにしました。 候補者」というカスタムオブジェクトを作成し、組織全体のデフォルトを 非公開」に設定しました。候補者オブジェクトのルックアップにより、従業員が面接官として設定されます。ルックアップフィールドが 面接官」ユーザーに設定されている場合に、レコードへの読み取りアクセス権を自動的に付与するには、どのような方法を使用すればよいでしょうか？12345
A. レコードはApexクラスを使用して共有できます。678910
B. レコードは権限セットを使用して共有できます。1112131415
C. レコードは共有ルールを使用して共有できます。1617181920
D. レコードは現在の設定と共有できません。2122232425
正解: **(**正解を表示します**)**
包括的かつ詳細な150〜250語の説明:333435
非公開共有モデル36では、レコードへのアクセスは所有者37と、ロール階層でその上位のユーザー38に制限されます。特定の項目（参照項目など）で識別されたユーザーにアクセス権を付与するには、動的な共有メカニズムが必要です。Apexによる共有管理 オプションA）は、開発者が参照項目の値に基づいてプログラムで共有レコードを作成できるため、ここでは適切な選択肢です。 面接官」参照項目にデータが入力されると、トリガによってCandidate__Shareオブジェクトにレコードが挿入され、指定されたユーザーに 読み取り」アクセス権が付与されます。
共有ルール オプションC）は、通常、ユーザーへの動的な参照ではなく、静的な条件または所有権に基づいてレコードを共有するため、このシナリオには一般的に厳格すぎます。権限セット オプションB）は、オブジェクトレベルおよびフィールドレベルの権限を付与しますが、プライベートOWD環境においてユーザーが所有していない特定のレコードへのアクセスは付与しません。Apexを使用することで、面接担当者が変更されても共有アクセスがリアルタイムで更新され、安全かつ機能的な面接プロセスを維持するための必要な自動化が実現します。

質問: **36**
部分的なページ更新を実行する Ajax 動作を開始するために使用できる 3 つの Visualforce コンポーネントはどれですか?
3 つの答えを選択してください
A. <apex:commondLink>
B. <apex:コマンドボタン>
C. <Apex:フォーム>
D. <apex:actionsStatus>
E. <apex:actionSupport>
正解: **(**正解を表示します**)**
Ajax 動作を開始して部分的なページ更新を実行するために使用できる 3 つの Visualforce コンポーネントは、apex:commandLink、apex:commandButton、および apex:actionSupport です。これらのコンポーネントを使用すると、開発者は Apex コントローラメソッドまたは JavaScript 関数を呼び出し、apex:actionStatus または apex:reRender 属性を指定して、ページ全体を更新せずに

ページの一部を更新できます。apex:commandLink コンポーネントと apex:commandButton コンポーネントは、それぞれ HTML リンクまたはボタンとしてレンダリングされ、ユーザがクリックしたときにアクションをトリガするために使用できます。apex:actionSupport コンポーネントを使用すると、他のコンポーネントに Ajax サポートを追加したり、ユーザがマウスオーバー、変更、ぼかしなどのイベントを実行したときにアクションをトリガするために使用できます。apex:form コンポーネントは、サーバと対話する必要があるコンポーネントを囲むために使用されますが、それ自体では Ajax 動作を開始しないため、有効な回答ではありません。apex:actionStatus コンポーネントは、読み込みメッセージやスピナーなど、Ajax リクエストのステータスを表示するために使用されますが、それ自体では Ajax の動作を開始しないため、有効な回答ではありません。
参考 :[Visualforce コンポーネントリファレンス]、[Visualforce 開発者ガイド]

質問: **37**
開発者は、テスト クラス内から組織データにアクセスしようとしています。テストクラスに (seeAllData=true) アノテーションが必要な sObject タイプはどれですか?
A. レコードの種類
B. プロフィール
C. ユーザー
D. レポート
正解: **(正解を表示します)**
説明
Test (SeeAllData=true) アノテーションは、テストクラスと個々のテストメソッドに、組織内のすべてのデータ（テストで作成されなかった既存のデータを含む）へのアクセスを許可するために使用されます。このアノテーションは、Salesforce API バージョン 24.0 以降を使用して保存された Apex コードに必須です。
Profile は、テストクラス内から組織データにアクセスするために isTest (SeeAllData=true) アノテーションを必要とする sObject 型の 1 つです。このアノテーションを必要とする他の sObject 型には、User、UserRole、Group、GroupMember、QueueSobject、CronTrigger などがあります。したがって、Profile を sObject タイプとして使用するには、テスト クラスに isTest (SeeAllData=true) アノテーションが必要です。

質問: **38**
開発者は、次の HTML スニペットを使用して、sObject Lightning ページに常駐する再利用可能な Aura コンポーネントを開発しています。
<aura:component implements="force:hasRecordId,flexipage:availableForAllPageTypesM>
こんにちは!
</aura:コンポーネント>
コンポーネントのコントローラは、追加のテスト範囲を必要とせずに、sObject が存在する Lightning ページのコンテキストをどのように取得できますか?
A. force:hasSobjectName を実装に追加します。
B. sObject タイプをコンポーネント属性として設定します。

C. Apex クラスで getSObjectTypeQ メソッドを使用します。
D. デザイン属性を作成し、アプリ ビルダーを介して構成します。
正解: **A (**コメントを発表する**)**

質問: **39**
開発者は、特定の取引先レコードが Visualforce コントローラのテストクラスでテストされていることをどのように確認する必要がありますか?
A. テスト クラスにアカウントを挿入し、テスト クラスでページ参照をインスタンス化し、System.currentFageReference() .getFarameters() .put{} を使用してアカウント ID を設定します。
B. アカウントを Salesforce に挿入し、テスト クラスでページ参照をインスタンス化し、システムを使用します。
setFarentRecordId() .get() を使用してアカウント ID を設定します。
{of テスト クラスでページ参照をインスタンス化し、テスト クラスにアカウントを挿入し、=seeAllData=trus を使用してアカウントを表示します。
C. テスト クラスでページ参照をインスタンス化し、テスト クラスにアカウントを挿入し、system.setFarentRecordrd() .get() を使用してアカウント ID を設定します。
正解: **(**正解を表示します**)**
特定のレコードを操作するVisualforceコントローラをテストするには、テストクラスに必要なテストデータを挿入する必要があります。次に、VisualforceページのPageReferenceをインスタンス化し、ページパラメータにレコードIDを設定し、System.currentPageReference().getParameters().put('id', recordId)を使用して、そのレコードを含むページへの移動をシミュレートします。
参考資料: Apex 開発者ガイド - カスタムコントローラとコントローラ拡張のテスト

質問: **40**
ある会社には、ユーザの入力に基づいて結果のリストを返す Apex コントローラを呼び出すことで、ユーザが特定のオブジェクト タイプのレコードを検索できるカスタム コンポーネントがあります。検索が完了すると、searchComplete イベントが発生し、結果がイベントの results 属性に入れられます。このコンポーネントは他のコンポーネント内で使用されるように設計されており、1つのページに複数回表示される場合があります。
検索完了時にイベントを発生させるために追加する最適なコードは何ですか?
A.

```
var evt = component.getEvent("searchComplete");
evt.setParams({results: results});
evt.fire();
```

B.

```
var evt = $A.get("e.c.searchComplete");
evt.set("v.results", results);
evt.fire();
```

C.

```
var evt = component.getEvent("searchComplete");
evt.set("v.results", results);
evt.fire();
```

D.

```
var evt = $A.get("e.c.searchComplete");
evt.setParams({results: results});
evt.fire();
```

正解: **(**正解を表示します**)**

検索が完了したときにイベントを発生させるために追加する必要がある最適なコードはオプション D です。

このオプションでは、イベント定義の取得に$A.get("ec:searchComplete")メソッド、結果属性の設定にsetParams()メソッド、イベントの発動にfire()メソッドを使用しています。また、イベント定義の取得にcmp.getEvent("searchComplete")メソッドを使用しています。これは$A.get()メソッドと同等です。オプションAは誤りです。$A.createComponent()メソッドを使用していますが、これはイベントではなくコンポーネントを動的に作成するために使用されます。オプションBは誤りです。$A.eventService.newEvent()メソッドを使用していますが、これは非推奨であり使用すべきではありません。オプションCは誤りです。$A.enqueueAction()メソッドを使用していますが、これはイベントではなくApexコントローラメソッドを呼び出すために使用されます。参考 :[イベントの作成と発動]、[Lightningコンポーネント開発者ガイド]

質問: **41**

以下のコンポーネント コードと要件を参照してください。

要件：

1. モバイル デバイスの場合、情報は 3 行に表示されます。
2. デスクトップとタブレットの場合、情報は 1 行で表示されます。

要件 2 希望どおりに表示されない。

デスクトップとタブレットの要件を満たす正しいコンポーネント コードを持つオプションはどれですか?

A.

B.

C.

D. 

正解: **(**正解を表示します**)**

モバイル デバイスとデスクトップおよびタブレットで異なる方法で情報を表示するための指定された要件を満たすには、開発者は適切なサイズ属性を持つ Lightning レイアウトおよび Lightning レイアウト項目コンポーネントを使用して、フォーム ファクターに基づいてレイアウトを制御する必要があります。

オプションAは正解です。デスクトップとタブレットデバイスではsize属性を『2』に設定しているため、フレームワークはコンテナの幅全体を占め、結果として1行になります。モバイルデバイスではデフォルトで全幅に設定され、レイアウトアイテムが垂直に積み重ねられ、結果として3行になります。

オプション B、C、および D は、デスクトップとタブレットの正しいサイズ属性が設定されていないか、これらのデバイスで 1 行になる適切な構造が維持されていないため、正しくありません。

参考文献:

グリッドシステム上の Lightning デザインシステム: Lightning グリッドシステム

Lightning Web コンポーネントのレイアウトコンポーネントに関するドキュメント: レイアウトコンポーネント

質問: **42**

[アカウント (ID、名前)、連絡先、商談、リードを返すすべてのフィールドで『マップ』を検索] 次の SOSL クエリの有効な戻り値の型は何ですか?

A. リスト<AggregateResult>

B. リスト<sObject>

C. リスト<アカウント>

D. リスト<リスト<sObject>>

正解: **(**正解を表示します**)**

質問: **43**

開発者は、Lightning Web コンポーネントからの入力に基づいてアカウントを更新する Apex クラスを作成しました。

アカウントの更新は、まだ登録されていない場合にのみ行ってください。

ジャワ

account = [SELECT Id, Is_Registered__c FROM Account WHERE Id = :accountId]; if (!
account.Is_Registered__c) { account.Is_Registered__c = true;
// ...他のアカウント フィールドを設定します...
アカウントを更新します。
}
複数のユーザーが同時に同じアカウントを更新した場合に、互いの更新が上書きされないようにするには、開発者は何をすべきでしょうか?
A. 更新の周囲に try/catch ブロックを追加します。
B. 更新の代わりに upsert を使用します。
C. SOQL クエリで FOR UPDATE を使用します。
D. 最近更新されていないことを確認するために、クエリに LastModifiedDate を含めます。
正解: **(**正解を表示します**)**
マルチユーザー環境では、2人のユーザーが同時に同じレコードをクエリし、Is_Registered__c が false であることを確認してから更新処理を進めると、 競合状態」が発生します。これを防ぐには、開発者は行レベルロックを実装する必要があります。
Salesforce SOQLでは、まさにこの目的でFOR UPDATE オプションC）キーワードが使用されます。クエリにFOR UPDATEが含まれている場合、プラットフォームは返されたレコードに排他ロックを設定します。2番目のトランザクションがFOR UPDATEを使用して同じレコードをクエリしようとすると、最初のトランザクションが完了 コミットまたはロールバック）するまで強制的に待機させられます。10秒以内にロックが解除されない場合、2番目のトランザクションはQueryExceptionをスローします。
FOR UPDATE を使用すると、クエリを実行する最初のユーザーがアカウントを 予約」します。2番目のユーザーのリクエストは待機状態となり、クエリが実際に実行されてデータを取得するまでに、最初のユーザーによって Is_Registered__c が true に更新されるため、if (!
account.Is_Registered__c) チェックは失敗し、重複登録は防止されます。オプション D は手動の 楽観的ロック」アプローチであり、プラットフォームのネイティブロックメカニズムよりも信頼性が低くなります。オプション A と B は、最初のチェックのタイミングの問題に対処していません。

質問: **44**
コントローラーを単体テストするときのベスト プラクティスは何ですか? 2つ選んでください。）
A. seeAllData=true を使用してテスト データにアクセスします。
B. Test.setMock() を利用してユーザー操作をシミュレートします。
C. getParameters().put を使用してクエリ パラメーターを設定します。
D. getURL() を使用して正しい参照を確認します
正解: **C,D (**コメントを発表する**)**

質問: **45**

実行されるプロセスの規模や種類によっては、Salesforce のイベントを外部システムで処理する必要がある場合があります。顧客との電話中に、Salesforce で作成中の注文の価格を取得する必要がある Salesforce のユーザーの使用例を考えてみましょう。価格設定ロジックは、サードパーティ システムに既に存在します。このロジックを Salesforce で再作成する代わりに、サードパーティ システムの要求を行うことで活用されます。応答 (この場合は価格) が返され、Salesforce に保存されます。最適なソリューションは何ですか?
A. 保存時に価格を取得し、Salesforce に価格を保存するためのプロセス ビルダー プロセスとアウトバウンド メッセージ
B. 新しく保存された注文のバッチを数分ごとに処理して価格を Salesforce に保存する ETL ツール
C. 注文を保存するとリアルタイムの Apex コールアウトが行われ、価格が Salesforce に保存される Apex トリガー
D. リアルタイムの Apex コールアウトを作成して価格を表示し、Salesforce に保存できる Visualforce ページ
正解: **C** (コメントを発表する)

質問: **46**
コードの実行時に、開発者が LimitException: Too many query rows: 50001 エラーを受け取ります。
予期しない数の行を返す可能性のある特定のコンポーネントを特定するための最も迅速で正確なメカニズムを提供する、開発者コンソールを使用したデバッグ方法はどれですか?
A. SOQL_EXECUTE_END ステートメントでデバッグ ログをフィルタリングして、各 SOQL クエリの結果を追跡します。
B. 実行概要を使用して、各実行ユニットによって返される行数を確認します
C. System.debug(System.getQueryRows()) をコードに追加して、SOQL の使用状況を追跡します。
D. Debug Logs 内の Row Limit 警告メッセージの数を数えます。
正解: (正解を表示します)

質問: **47**
次のコード スニペットを考えてみましょう。

2 つの答えを選択してください

A. エンドポイントの URL を endpointURL という名前のカスタム ラベルに保存します。

B. エンドポイントと資格情報を保存するための名前付き資格情報 endpoint_NC を作成します。

C. reg.setEndpoint(Label.endPoinrURL); を使用します。

D. req.setEndpoint,calloat;eodPolat NC'); を使用します。コールアウトリクエスト内で。

正解: **A,D** (コメントを発表する)

セキュリティを維持しながら LWC がデータを効率的に表示できるようにするために、開発者が Apex クラスに実装する必要がある 2 つの変更は、エンドポイントの URL を endpointURL というカスタム表示ラベルに保存することと、コールアウト要求内で req.setEndpoint('callout:endpoint_NC'); を使用することです。エンドポイントの URL をカスタム表示ラベルに保存することで、開発者は Apex コードに URL をハードコーディングする必要がなくなり、[設定] メニューから URL 値を管理できます。これにより、コードの保守性と柔軟性が向上し、権限のないユーザーに URL が公開されるのを防ぐことができます。コールアウト要求内で req.setEndpoint('callout:endpoint_NC'); を使用すると、開発者は指定ログイン情報を使用してエンドポイントと外部システムのログイン情報を保存できます。これにより、認証プロセスが簡素化されるだけでなく、ログイン情報が安全かつ暗号化されることも保証されます。setEndpoint() メソッドは、callout: プレフィックスに続いて、コールアウト要求に使用する指定ログイン情報の名前を受け入れます。getSubjectType() メソッドは、プラットフォームイベントの件名タイプを取得するために使用されるため、有効な回答ではありません。ただし、データ表示の効率性やセキュリティには影響しません。WHERE 句は、SOQL クエリでレコードを絞り込むために使用されるため、有効な回答ではありません。ただし、データ表示の効率性やセキュリティには影響しません。参考資料: [カスタムラベル]、[指定ログイン情報]、[Apex 開発者ガイド]

質問: **48**

ユニバーサルコンテナーズは、Salesforce標準モバイルアプリを利用して、iOSとAndroid向けの採用アプリを開発したいと考えています。カスタムユーザーインターフェース設計を採用しており、オフラインアクセスは不要です。アプリ開発にはどのようなアプローチが推奨されますか？

A. Salesforce SDK

B. Lightning Web コンポーネント

C. Lightning Experience ビルダー

D. ビジュアルフォース

正解: (正解を表示します)

標準のSalesforceモバイルアプリ内で表示するカスタム機能を開発する場合は、Lightning Web Components（LWC）（オプションB）が推奨されるアプローチです。LWCは、ブラウザでネイティブに実行される最新のWeb標準（ES6+、Web Components）を活用した、Salesforce開発における最新の高パフォーマンス標準です。
LWC が他のオプションよりも優先される理由はいくつかあります。
* パフォーマンス: LWC は軽量で、Visualforce (オプション D) よりもはるかに高速で応答性に優れたユーザー インターフェースを提供します。これは、モバイル ユーザー エクスペリエンスにとって重要です。
* 標準化: アプリは標準の Salesforce モバイル アプリからアクセスされるため、LWC はモバイル ナビゲーション、タブ、アクション メニューにシームレスに統合されます。
* カスタマイズ: LWC は CSS と HTML を完全に制御できるため、開発者は Lightning Data Service を活用して効率的なデータ処理を実現しながら、「カスタム ユーザー インターフェイス デザイン」の要件を満たすことができます。
オプションA（Salesforce SDK）は、「カスタム/ネイティブ」スタンドアロンアプリの構築を目的としており、標準Salesforceアプリの拡張を目的としたものではありません。オプション C（Experience Builder）は、主に外部サイト（コミュニティ）向けです。LWCは、社内モバイルユーザー向けに「カスタム設計」と「ネイティブ統合」の最適なバランスを提供します。

質問: **49**
Universal Containers（UC）は、翻訳ワークベンチを有効化し、選択リストの値を翻訳しました。UCの取引先オブジェクトには、営業担当者が取引先が既に所有しているUC製品を指定できるカスタムの複数選択リスト項目「Products__c」があります。開発者は、取引先レコード（「Products__c」項目を含む）を取得するApexメソッドを作成する必要があります。開発者は、Products__cの値が現在のユーザーの言語で表示されるようにするために、どのような対策を講じるべきでしょうか？
40
A. SOQL クエリのフィールド リストで toLabel(Products__c) を使用します。
B. SOQL 結果リスト内の各レコードに対して translate() メソッドを呼び出します。
C. SOQL 結果リスト内の各レコードのロケールを設定します。
D. SOQL クエリで locale 句を使用します。
正解: **(**正解を表示します**)**
Salesforce組織でトランスレーションワークベンチを使用すると、選択リストの値を複数の言語に翻訳できます。デフォルトでは、SOQLクエリは選択リスト項目の「マスタ」値またはAPI値（通常は英語）を返します。開発者が返されるデータに、現在のユーザーの言語設定に基づいて翻訳されたラベルを反映させる必要がある場合は、SOQL SELECT文内でtoLabel()関数を使用する必要があります。
SELECT toLabel(Products__c) FROM Account (オプションA) を使用すると、Salesforceクエリエンジンは結果セット内の各選択リストエントリの翻訳された値を検索してから、Apexに返します。これは、ユーザーが母国語で値を表示することを期待するUIコンポーネントやレポートで特に役立ちます。標準の選択リストと複数選択の選択リストの両方で機能します。

その他のオプションは、Apexの標準機能として存在しないため、正しくありません。SObjectレコードには translate() メソッドがありません（オプションB）。また、Apexでは、個々のレコードに手動で「ロケールを設定」して翻訳をトリガーすることはできません（オプションC）。SOQL構文にも「locale句」はありません（オプションD）。toLabel() の使用は、プラットフォーム標準の最も効率的な方法で、データアクセス層内で直接ローカライズされたデータ取得を処理します。

質問: **50**
ビジネス ルールでは、新しいアカウントを作成するときに常に連絡先を作成する必要があります。連絡先の作成に失敗した場合にアカウントが作成されないようにするためのカスタム画面を開発するときに使用できるものは何ですか?
A. setSavePoint() と rollback() を try/catch ブロックで使用します。
B. 連絡先の挿入に失敗した場合は、Database.Delete メソッドを使用します。
C. allOrNone を False に設定して Database.Insert メソッドを使用します。
D. 連絡先の検証ルールを無効にし、トリガーで既定値を設定します。
正解: **A (**コメントを発表する**)**

質問: **51**
Salesforce 管理者と Cloud Kicks は、米国内のすべての郵便番号とその郵便番号が属する Cloud Kicks 販売地域を保存するために、Region_c というカスタム オブジェクトを作成しました。

Cloud Kicks は、リードの郵便番号に基づいてリージョンを設定するためのリード上のトリガーを必要としています。
この要求を満たす最も効率的な方法はどのコード セグメントですか?
A.

```
Set<String> zips = new Set<String>();
for(Lead l : Trigger.new) {
    if(l.PostalCode != Null) {
        zips.add(l.PostalCode);
    }
}

List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips];

for(Lead l : Trigger.new) {
    for(Region__c r :regions) {
        if(l.PostalCode == r.Zip_Code__c) {
            l.Region__c = r.Region_Name__c;
        }
    }
}
```

B.

```
for(Lead l : Trigger.new) {
     Region__c reg = [SELECT Region_Name__c FROM Region__c WHERE Zip_Code__c = :l.PostalCode];
     l.Region__c = reg.Region_Name__c;
}
```

C.

```
Set<String> zips = new Set<String>();
for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          zips.add(l.PostalCode);
     }
}

for(Lead l : Trigger.new) {
     List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips];
     for(Region__c r :regions) {
          if(l.PostalCode == r.Zip_Code__c) {
               l.Region__c = r.Region_Name__c;
          }
     }
          }
     }
}
```

D.

```
Set<String> zips = new Set<String>();
for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          zips.add(l.PostalCode);
     }
}

List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips];

Map<String, String> zipMap = new Map<String, String>();
for(Region__c r :regions) {
     zipMap.put(r.Zip_Code__c, r.Region_Name__c);
}

for(Lead l : Trigger.new) {
     if(l.PostalCode != Null) {
          l.Region__c = zipMap.get(l.PostalCode);
     }
}
```

正解: **(**正解を表示します**)**

質問: **52**

Universal Containers は、カスタム Lightning ページを使用して、アカウントの段階的なウィザード検索を実行するメカニズムを提供します。ウィザードの手順の 1 つは、ユーザーがテキストフィールド ERF Number_c にテキストを入力できるようにすることです。このテキストは、一致するアカウントを見つけるためのクエリで使用されます。

```
erpNumber = erpNumber + '%';
List<Account> accounts = [SELECT Id, Name FROM Account WH
ERP_Number__c LIKE :erpNumber]
```

問題を解決するにはどの手順を実行する必要がありますか?
A. SO0L クエリを非同期プロセス内に移動します。
B. SOQL クエリを for ループの一部として実行します。
C. ERP_Number__ c フィールドを必須としてマークします。
D. ERP_Numker = フィールドを外部 10 としてマークします。
正解: **D (**コメントを発表する**)**

質問: **53**
以下のオプションのクエリと次の情報を考慮してください。
* これらのクエリでは、200,000 を超える取引先レコードがあると想定しています。
* これらのレコードには、論理的に削除されたレコードが含まれます。つまり、削除されたレコードがまだごみ箱に残っています。
* には外部 ID としてマークされているフィールドが 2 つあります。
アカウント。これらのフィールドは、customer_Number_c と ERR_Key_s です。
大量のデータに対して最適化されている 2 つのクエリはどれですか?
2 つの答えを選択してください
A. アカウントから ID を選択、名前が != ' ' かつ IsDeleted = false
B. アカウントから I4 を選択 WHERE 名前 !- NULL
C. アカウントから 1d を選択 WHERE 名前 != '°
AND Customer_Number_c- "ValueA'
D. :aListVariable 内の id を含む Account から ID を選択
正解: **(**正解を表示します**)**

質問: **54**
開発者は、取引先の最近連絡した 5 人の取引先責任者を表示する取引先レコードページの Lightning Web コンポーネントを作成しました。Apex メソッド Contacts は連絡先のリストを返し、コンポーネントのプロパティに関連付けられます。

Apex メソッドを接続できるようにするには、上記のコードのどの 2 行を変更する必要がありますか?

2 つの答えを選択してください

A. @AuraEnabled {cacheable=trues) を 08 行目に追加します。

B. 09 行目から private を削除します。

C. 04行目にpublicを追加します。

D. @AuraEnabled {cacheable=true) を行 03 に追加します。

正解: **(**正解を表示します**)**

Apex メソッドをワイヤリングできるようにするには、開発者はコードの 2 行を変更し、行 4 に public を追加し、行 3 に @AuraEnabled(cacheable=true) を追加する必要があります。行 4 に public を追加すると、クラスの外部から Apex メソッドにアクセスできるようになります。これは、@wire デコレータがメソッドを呼び出すために必要です。行 3 に @AuraEnabled(cacheable=true) を追加すると、Apex メソッドがキャッシュ可能になり、@wire デコレータを使用する Lightning Web コンポーネントからメソッドを呼び出すことができるようになります。また、サーバとのやり取りの回数が減り、クライアント側のキャッシュを活用できるため、コンポーネントのパフォーマンスも向上します。開発者は、頻繁に変更されないデータやユーザコンテキストに依存しないデータを返すメソッドにこのアノテーションを使用する必要があります。行 8 に @AuraEnabled(cacheable=true) を追加する必要はありません。cacheable 属性は、@wire デコレータによって呼び出される最上位メソッドにのみ適用され、クラス内で呼び出されるヘルパーメソッドには適用されないためです。9行目のprivateを削除する必要はありません。ヘルパーメソッドはクラス内から呼び出される限りprivateのままで構いません。参考：「Lightning Webコンポーネントから Apexメソッドを呼び出す」、「メソッドを保存可能またはキャッシュ可能としてマークする」

質問: **55**

ある組織では、取引先責任者と取引先の住所を保存するたびに、Apex コードを使用して会社の標準に正規化する必要があると規定されています。これを実装する最適な方法は何でしょうか？

A. 連絡先と取引先で住所を正規化するApexトリガー

B. 取引先責任者トリガーを呼び出して住所を正規化する取引先責任者のApexトリガー
C. 取引先責任者のApexトリガーで、取引先トリガーを呼び出して住所を正規化する
D. 連絡先と取引先で、住所を正規化するためのヘルパークラスを呼び出す Apex トリガー
正解: (正解を表示します)
ソフトウェアエンジニアリングにおいて、DRY（Don't Repeat Yourself：同じことを繰り返さない）原則は保守性にとって不可欠です。住所の正規化ロジックが取引先と取引先責任者で同一である場合、そのロジックは2つの別々のトリガーファイルに重複して記述されるべきではありません。最適な実装（オプションD）は、トリガーヘルパーまたはユーティリティクラスを1つ作成することです。このクラスには、正規化ロジックを実行する静的メソッド（例：AddressService.normalize(List<sObject> records)）が含まれます。次に、Accountオブジェクトと Contactオブジェクトにそれぞれ小さなトリガーを作成します。どちらのトリガーも、この共有ヘルパーメソッドを呼び出すだけです。
このアプローチにはいくつかの利点があります。
* 保守性: 会社の住所標準が変更された場合 (例: 「St.」を「Street」に変更する)、開発者は 1 か所のコードのみを更新する必要があります。
* 再利用性: ヘルパー メソッドは、匿名 Apex スクリプトやバッチ ジョブなどの他のコンテキストからも呼び出すことができます。
* 読みやすさ: トリガーは「薄く」読みやすく、複雑なビジネス ロジックを含むのではなく、エントリ ポイントとしてのみ機能します。
オプションBとCは技術的に不可能です。あるオブジェクトのトリガーは、別のオブジェクトのトリガーを直接「呼び出す」ことはできません。オプションAはコードの重複につながり、一方のトリガーでロジックが更新され、もう一方のトリガーではその更新が忘れられることでバグが発生するリスクが高まるため、あまり最適ではありません。

質問: **56**
Universal Containersは、Convention_Attendee__cに非公開共有モデルを実装しています。参照フィールドEvent_Reviewer__cが作成されています。経営陣は、イベントレビュー担当者に、担当するすべてのレコードへの読み取り/書き込みアクセス権を自動的に付与したいと考えています。最適なアプローチは何でしょうか？
A. Convention Attendee カスタム オブジェクトに after insert トリガーを作成し、Apex Sharing Reasons と Apex Managed Sharing を使用します。
B. Convention Attendee カスタム オブジェクトに before insert トリガーを作成し、Apex Sharing Reasons と Apex Managed Sharing を使用します。
C. コンベンション参加者カスタム オブジェクトに条件に基づく共有ルールを作成し、イベントレビュー担当者とレコードを共有します。
D. コンベンション参加者カスタム オブジェクトに条件に基づく共有ルールを作成し、イベントレビュー担当者のグループとレコードを共有します。
正解: (正解を表示します)

非公開共有モデルでは、アクセスは所有者と階層内でその上位のユーザーに制限されます。参照項目（Event_Reviewer__cなど）で特定のユーザーが識別された場合、共有ルールはロール、公開グループ、またはテリトリーのみを対象としているため、標準の共有ルールではその特定のユーザーに動的にアクセス権限を付与できません。1 解決策はApexによる共有管理（オプションA）です。after insert（およびおそらくafter update）トリガを使用することで、開発者はオブジェクトの「Share」テーブル（例Convention_Attendee__Share）にプログラム的にレコードを作成できます。各共有レコードには、UserOrGroupId（参照項目から取得）、AccessLevel（読み取り/書き込みの場合は「Edit」）、およびRowCause（Apexの共有理由）が指定されます。
Convention_Attendee__cレコードにShareレコードを関連付けるには、そのレコードのIDが存在している必要があるため、「After」トリガが必要です。「Before」トリガ（オプションB）は、レコード自体の項目更新に使用され、関連する共有レコードの作成には使用されません。オプションCとDは、レコード自体の項目に保存されているユーザーIDを動的に参照できないため、実行できません。Apex管理共有は、この動的なセキュリティ要件を最もきめ細かく自動的に処理する方法を提供します。

質問: **57**
開発者は、システムに入力されたメールアドレスとカスタムオブジェクト「Survey_Response__c」が、ブロック対象ドメインのリストに含まれていないことを確認する必要があります。ブロック対象ドメインのリストは、ユーザーによるメンテナンスを容易にするため、カスタムオブジェクトに保存されています。Survey_Response__c オブジェクトへの入力は、カスタムVisualforceページから行います。これを実装する最適な方法は何でしょうか？
A. Contact および Survey_Response__c オブジェクトの検証ルールにロジックを実装します。
B. 連絡先の Apex トリガーおよびカスタム Visualforce ページ コントローラから呼び出されるヘルパー クラスにロジックを実装します。
C. 連絡先の Apex トリガーにロジックを実装し、カスタム Visualforce ページ コントローラ内にもロジックを実装します。
正解: **(正解を表示します)**
この要件では、2つの異なるオブジェクト（連絡先とSurvey_Response__c）にビジネスルールを適用し、異なるエントリポイント（連絡先用の標準UI/トリガーとアンケート回答用のカスタムVisualforceページ）を介して適用する必要があります。トリガーとコントローラにロジックを個別に実装する（オプションC）ことは、コードが重複し、メンテナンスの手間が増え、将来ロジックを変更する必要がある場合に不整合が発生するリスクが高まるため、アーキテクチャ的に適切ではありません。検証ルール（オプションA）も、複雑でスケーラブルでない回避策なしに、別の「ブロックされたドメイン」オブジェクト内のレコードリストに対してネイティブに動的なルックアップを実行できないため、ここでは不適切です。
最適なアプローチは、検証ロジックを単一の「ヘルパー」クラスまたは「サービス」クラスに一元化することです。これは、ソフトウェア開発におけるDRY（Don't Repeat Yourself：繰り返しを避ける）原則に準拠しています。ヘルパークラスに共有メソッドを作成することで、開発者はContactトリガとVisualforceコントローラの両方が、メールドメインの検証に全く同じロジックを使用する

ことを保証できます。この一元化されたメソッドは、ブロックされたドメインを含むカスタムオブジェクトをクエリし、検証結果を返します。このアーキテクチャにより、テストが簡素化され、プラットフォーム全体でビジネスルールが均一に適用され、管理者はコードを追加変更することなく、ブロックされたドメインリストを更新できます。

質問: **58**
ある会社には、参照データをキャッシュするものなど、多くの Lightning コンポーネントを含む Lightning ページがあります。このページには常に最新の参照データが表示されるわけではないことが報告されています。
開発者は Lightning ページの問題を分析および診断するために何を使用できますか?
A. Salesforce Lightning Inspector の [トランザクション] タブ
B. Salesforce Lightning Inspector の [アクション] タブ
C. Salesforce Lightning Inspector の [イベント ログ] タブ
D. Salesforce Lightning Inspector の [ストレージ] タブ
正解: **D (**コメントを発表する**)**
Salesforce Lightning Inspector の 「ストレージ」タブでは、キャッシュなどクライアント側に保存されているデータを表示および管理できます。これは、最新の参照データが表示されなくなる可能性のある、古くなったキャッシュデータに関連する問題の診断に役立ちます。
参考資料: Salesforce Lightning Inspector Chrome 拡張機能

質問: **59**
組織には、取引先責任者と取引先のアドレスが保存されるたびに、Apex コードによって会社標準に正規化される必要があるという要件があります。
これを実装する最適な方法は何ですか?
A. ヘルパークラスを呼び出してアドレスを正規化する、取引先責任者と取引先の Apex トリガ
B. 連絡先トリガーを呼び出してアドレスを正規化する取引先の Apex トリガー
C. アドレスを正規化する取引先責任者と取引先の Apex トリガー
D. アカウント トリガーを呼び出してアドレスを正規化する取引先担当者の Apex トリガー
正解: **(**正解を表示します**)**
ヘルパークラスを使用した取引先責任者および取引先の Apex トリガーは、コードの再利用とクリーンなトリガー ロジックの維持のためのベスト プラクティスです。
参考資料: Apex 開発者ガイド - トリガーと実行順序

質問: **60**
Aura コンポーネントから呼び出される以下の Apex コントローラについて考えてみましょう。

このコードのどこが間違っているのでしょうか?

A. 行 1: クラスはグローバルである必要があります

8. 1 行目と 6 行目: クラスとメソッドはグローバルである必要があります

B. 6 行目: メソッドは静的でなければなりません

C. 8 行目: メソッドは返す前に、まずリストを JSON にシリアル化する必要があります。

正解**: C (**コメントを発表する**)**

Apexクラスで定義されたメソッドは、Aura対応メソッドとして使用するには静的メソッドである必要があります。Aura対応メソッドは、クラスをインスタンス化することなくLightningフレームワークによって呼び出されるため、静的メソッドである必要があります。参考資料 :Apex開発者ガイド - LightningへのApexメソッドの公開

質問**: 61**

開発者は、次の HTML スニペットを使用して、sObject Lightning ページに常駐する再利用可能な Aura コンポーネントを開発しています。

Caura: コンポーネントの実装 - "forcethaaRecordid, flexipage:availableForAllPageTypes">

こんにちは!

</aura:コンポーネント>

コンポーネントのコントローラーは、sobject がない Lightning ページのコンテキストをどのように取得できますか?

追加のベストカバレッジが必要ですか?

A. force:hasSobjectName を implements 属性に追加します。

B. オブジェクト タイプをコンポーネント属性として設定します。

C. Apex クラスで getSObjectType () メソッドを使用する

D. デザイン属性を作成し、App Builder で構成します

正解**: (**正解を表示します**)**

質問: **62**
Lightning Web コンポーネントを構築する際に、開発者がデータ取得の最高のパフォーマンスを得るために実装する必要がある 3 つのアプローチはどれですか?
3つの答えを選択してください
A. getRecordUi を使用してメタデータを取得します。
B. layoutTypes : ['Full'] を使用して一連のフィールドを表示します。
C. 可能な限り (cacheable-true) を使用します。
D. 頻繁にアクセスされるデータには遅延ロードを使用します。
E. Lightning データ サービスを使用します。
正解: **C,D,E (**コメントを発表する**)**

質問: **63**
Universal Containers は、顧客がレンタルしているコンテナの数と返却期限を追跡できるように、Customer Community Plus ライセンスでカスタマー コミュニティを使用したいと考えています。顧客の多くは、同じ組織内のさまざまな部門を代表する複雑なアカウント階層を持つグローバル企業です。要件の 1 つは、部門を表すさまざまな取引先レコードに連絡先を関連付けるジャンクション オブジェクトに基づいて、同じ取引先階層内の特定のコミュニティ ユーザーが複数の部門のコンテナを表示できることです。これらの要件を解決するソリューションはどれですか?
A. 所有者に基づいて適切なレコードを表示するフィルターを備えたジャンクション オブジェクトのカスタム リスト ビュー
B. 共有せずにレコードを公開することを指定するカスタム コントローラを使用する Visualforce ページ
C. コミュニティホームページのカスタムレポートタイプとレポート Lightning コンポーネント
D. 連結オブジェクトのリレーションに基づいて Apex 管理共有レコードを作成する Apex トリガ
正解: **(**正解を表示します**)**

質問: **64**
Universal Charities (UC) は、Salesforce を使用して個人や法人からクレジット カード引き落としの形で電子寄付を集めています。
カスタマー サービス エージェントがクレジット カード情報を入力すると、寄付が処理されるようにその情報が 8 つのサードパーティ決済プロセッサに送信される必要があります。UC は個人向けに 1 つの決済プロセッサを使用し、法人向けには別の決済プロセッサを使用します。

システム管理者が展開後に必要に応じて設定を変更できるように、開発者はさまざまな決済プロセッサの設定を保存するために何を使用する必要がありますか?
A. カスタムオブジェクト
B. カスタムメタデータ
C. 階層カスタム設定
D. リストのカスタム設定
正解: **C** (コメントを発表する)
階層カスタム設定は、様々な決済処理業者の設定を保存するための手段です。これにより、システム管理者は導入後に必要に応じて設定を変更できます。階層カスタム設定は、ユーザーやプロファイルごとに異なる設定データを保存できるカスタム設定の一種です。階層カスタム設定はApexコードからアクセスでき、ユーザーのコンテキストに基づいてコードの動作を制御するために使用できます。階層カスタム設定を使用することで、開発者は個人顧客と法人顧客の決済処理業者のURLと認証情報を保存できるカスタム設定を作成できます。開発者は、Apexコードでこのカスタム設定を使用することで、ユーザーのプロファイルまたはIDに基づいて使用する決済処理業者を決定できます。また、システム管理者は、決済処理業者の設定が変更された場合、コードを変更することなく、ユーザーインターフェースでカスタム設定値を変更することもできます。参照: [階層カスタム設定]、[カスタム設定へのアクセス]

質問: **65**
開発者は、連絡先用および調査応答 c と呼ばれるカスタム オブジェクト用にシステムに入力された電子メール アドレスが、ブロックされたドメインのリストに属していないことを確認する任務を負っています。
ブロックされたドメインのリストは、ユーザーによるメンテナンスを容易にするためにカスタム オブジェクトに保存されます。アンケートの応答 c オブジェクトは、カスタム Visualforce ページを介して設定されます。
これを実装する最適な方法は何ですか?
A. Contact オブジェクトと Burvey Response_c オブジェクトの検証ルールにロジックを実装します。
B. Contact の Apex トリガおよびカスタム Visualforce ページコントローラから呼び出されるヘルパークラスにロジックを実装します。
C. Contact の Apex トリガにロジックを実装し、カスタム Visualforce ページコントローラ内にもロジックを実装します。
D. カスタム Visualforce ページコントローラにロジックを実装し、Contact の Apex トリガからそのメソッドを呼び出します。
正解: **B** (コメントを発表する)
連絡先オブジェクトやカスタムオブジェクトに入力されたメールアドレスから、ブロックされているドメインを除外するための最適な方法は、ヘルパークラスを使用することです。このクラスは、連絡先オブジェクトのApexトリガ、およびアンケート回答に使用するカスタムVisualforceページのコントローラから呼び出すことができます。これにより、ロジックが一元化され、保守性と再利用性が向上します。参考資料 :Apex開発者ガイド - Apexトリガ

質問: **66**
ある会社にはカスタム オブジェクト Order__c があり、そのオブジェクトにはカスタム選択リスト フィールド Status__c があり、その値は「New」です。
「進行中」または「完了」と検索フィールド、
Contact__c、連絡先へ。
「履行された」注文がないすべての取引先責任者レコードの一意のリストを返す SOQL クエリはどれですか?
A. ID が入っていない連絡先から ID を選択 (Order__c WHERE Status__c から ID を選択
「満たされました」)
B. SELECT Contact__c FROM Order__c WHERE Status__c <> '完了'
C. SELECT Id FROM Contact WHERE Id NOT IN (SELECT Contact__c FROM Order__c WHERE Status__c = 'Fulfilled')
D. SELECT Contact__c FROM Order__c WHERE Id NOT IN (SELECT Id FROM Order__c Where Status__c = 'Fulfilled')
正解: **(正解を表示します)**
「Fulfilled」注文がない連絡先レコードの一意のリストを返す正しいSOQLクエリは、連絡先オブジェクトから、その連絡先IDが「Fulfilled」注文に関連付けられた連絡先IDセットに含まれていないIDを選択するクエリです。これは、Status__cが「Fulfilled」であるOrder__cレコードからContact__cフィールドを選択するサブクエリによって実行されます。
参考資料: SOQL および SOSL リファレンス ガイド - サブクエリ

質問: **67**
Apex メソッドを Lightning Web コンポーネントから命令的に呼び出す必要がある 2 つのシナリオはどれですか?
2つの答えを選択してください
A. Lightning Web コンポーネントのメイン コントローラの外部にあるメソッドの呼び出し
B. ボタンをクリックしてメソッドを呼び出す
C. Web サービスの呼び出しを行うメソッドの呼び出し
D. cacheable=true のアノテーションが付けられていないメソッドの呼び出し
正解: **(正解を表示します)**

質問: **68**
開発者は、lightning-record-edit-form を使用してリードに関する情報を収集する Lightning Web コンポーネントを作成しました。ユーザーは、潜在顧客レコードを保存しようとすると、入力に関するエラー メッセージが一度に 1 つしか表示されないと不満を抱いています。
JavaScript の介入を最小限に抑えながら、複数のフィールドで検証を実行し、複数のエラー メッセージを同時に表示するための推奨されるアプローチは何ですか?
A. トライ/キャッチ/最終的にブロック

B. 外部JavaScriptライブラリ
C. 検証ルール
D. 頂点トリガー
正解: **(**正解を表示します**)**
複数の項目の検証を実行し、JavaScript の介入を最小限に抑えて複数のエラーメッセージを同時に表示するには、外部 JavaScript ライブラリを使用することをお勧めします。外部 JavaScript ライブラリは、Lightning Web コンポーネントに追加の機能や特徴を提供できる再利用可能なコードのコレクションです。開発者は、jQuery Validation や Parsley.js など、フォーム検証をサポートする外部 JavaScript ライブラリを使用して、Lightning Web コンポーネントにインポートできます。その後、ライブラリのメソッドとオプションを使用して、各項目の検証ルールとメッセージを定義し、コンポーネントにエラーを表示できます。try/catch/finally ブロックはコード内の例外やエラーを処理するために使用されるため役に立ちませんが、項目の検証は実行されず、エラーメッセージも表示されません。検証ルールも役に立ちません。検証ルールはサーバー側でデータの品質と整合性を強化するために使用されますが、クライアント側で検証は実行されず、コンポーネントにエラーメッセージも表示されません。Apexトリガは、レコードの挿入、更新、削除、または復元の前後にロジックを実行するために使用されますが、クライアント側での検証やコンポーネントへのエラーメッセージの表示は行われないため、役に立ちません。参考 :[サードパーティ製JavaScriptライブラリの使用]、[Lightning Webコンポーネント開発者ガイド]

質問: **69**
Universal Containers は、Convention Attendee co カスタム オブジェクトのプライベート共有モデルを実装します。新しい品質保証の取り組みの一環として、同社はオブジェクトに Event_Reviewer_c ユーザー検索フィールドを作成しました。
管理者は、イベントレビュー担当者が、割り当てられているすべてのレコードへの読み取り/書き込みアクセスを自動的に取得できるようにしたいと考えています。
割り当てられたレビュー担当者がレコードへの読み取り/書き込みアクセスを確実に取得するための最善のアプローチは何ですか?
A. 大会出席者カスタムオブジェクトに挿入前トリガーを作成し、Apex 共有理由と Apex 管理共有を使用します。
B. 大会出席者カスタムオブジェクトに挿入後トリガーを作成し、Apex 共有理由と Apex 管理共有を使用します。
C. 大会出席者カスタム オブジェクトに基準ベースの共有ルールを作成して、イベント審査担当者とレコードを共有します。
D. 大会出席者カスタム オブジェクトに基準ベースの共有ルールを作成して、イベントレビュー担当者のグループとレコードを共有します。
正解: **(**正解を表示します**)**
Apex Managed Sharing を after insert トリガーと組み合わせて使用します。これにより、Event_Reviewer_c 項目などのユーザー定義の条件に基づいて、動的なレコード共有が可能になります。

参考資料: Apex 管理共有

質問: **70**
ある企業は、関連オブジェクトを通じて収益を追跡したいと考えています。約10万件の商談について、複雑なロジックに基づいて収益レコードを作成し、一度だけデータをシードする必要があります。これを自動化する最適な方法は何でしょうか？
A. Database.executeBatch() を使用して、Database.Batchable クラスを呼び出します。
B. System.enqueueJob() を使用して Queueable クラスを呼び出します。
C. Database.executeBatch() を使用して Queueable クラスを呼び出します。
D. System.scheduleJob() を使用して、Database.Scheduleable クラスをスケジュールします。
正解: **(**正解を表示します**)**
複雑なロジックを伴う大量データ処理（10万件のレコードのシードなど）には、Batch Apex（オプションA）が標準的かつ最も堅牢なソリューションです。Batch Apexは、レコードセット全体を管理しやすい小さなチャンク（デフォルトでは1バッチあたり200件）に分割することで、最大5,000万件のレコードを処理できるように設計されています。
各バッチ実行には、それぞれ独自のガバナ制限が設定されます。これは「複雑なロジック」において非常に重要です。10万件のレコードすべてを単一の同期トランザクションで処理しようとした場合に発生するCPU時間やヒープサイズの制限にトランザクションが達するのを防ぐためです。Batch Apexには、組み込みの状態管理とエラー処理（Database.RaisesPlatformEventsまたはfinishメソッド経由）も用意されています。
オプションB（Queueable）は、小規模で連鎖的なタスクに適しています。ある程度のボリュームは処理できますが、Batch ApexのQueryLocatorのように10万件のレコード処理には最適化されていません。オプションCは、executeBatchがBatchableクラスのみを受け入れるため、構文上不可能です。オプションD（Schedulable）はタイミング制御に使用されますが、10万件のレコード処理に必要な実際の処理は、タイムアウトエラーを回避するためにBatchクラスに委譲する必要があります。したがって、大規模なデータシード処理にはDatabase.Batchableが最適です。

質問: **71**
以下のテスト方法を参照してください。

```
Line 1: @isTest
Line 2: static void testMyTrigger()
Line 3: {
Line 4:     //Do a bunch of data setup
Line 5:     DataFactory.setupDataForMyTriggerTest();
Line 6:
Line 7:     List<Account> acctsBefore = [SELECT Is_Customer__c FROM Account WHERE Id IN :DataFactory.accounts];
Line 8:
Line 9:     //Utility to assert all accounts are not customers before the update
Line 10:    AssertUtil.assertNotCustomers(acctsBefore);
Line 11:
Line 12:    //Set accounts to be customers
Line 13:    for(Account a : DataFactory.accounts)
Line 14:    {
Line 15:        a.Is_Customer__c = true;
Line 16:    }
Line 17:
Line 18:    update DataFactory.accounts;
Line 19:
Line 20:    List<Account> acctsAfter = [SELECT Number_Of_Transfers__c FROM Account WHERE Id IN :DataFactory.accounts];
Line 21:
Line 22:    //Utility to assert Number_Of_Transfers__c is correct based on test data
Line 23:    AssertUtil.assertNumberOfTransfers(acctsAfter);
Line 24: }
```


このテストメソッドは、多くの取引先が同時に顧客に更新されるときに、開発者が多くのクエリを実行することがわかっている Apex トリガをテストします。
SOQL クエリが多すぎるため、テストメソッドは 20 行目で失敗します。
これを修正する正しい方法は何ですか?

A.

Use `Limits.getLimitQueries()` to find the total number of queries that can be issued.

B.

Change the `DataFactory` class to create fewer Accounts so that the number of queries in the trigger is reduced.

C.

Add `Test.startTest()` before and add `Test.stopTest()` after both Line 7 and Line 20 of the code.

D.

Add `Test.startTest()` before and add `Test.stopTest()` after Line 18 of the code.

正解: **(**正解を表示します**)**
これを修正する正しい方法は、コードの18行目の前にTest.startTest()を追加し、後にTest.stopTest()を追加することです。これにより、これら2つのメソッド間で実行されるコードのガバナ制限がリセットされ、SOQLクエリ制限に達することなくテストを実行できるようになります。Test.startTest()メソッドとTest.stopTest()メソッドは、大量のクエリを実行したり、非同期メソッドを呼び出したりするコードをテストするために使用されます12。これらのメソッドを使用することで、開発者はより多くのリソースを必要とするコードを特定し、ガバナ制限を超えていないことを確認できます。
参照：
Limits、startTest、stopTest の使用 | Apex 開発者ガイド | Salesforce 開発者 Test.StartTest と Test.StopTest - Salesforce 開発者コミュニティ 開発者が Salesforce で test startTest と test stopTest を使用する理由 | ForceTalks

質問: **72**

Aura コンポーネントにはアカウントに関する情報を表示するセクションがあり、デスクトップでは適切に機能しますが、ユーザーはモバイルデバイスやタブレットで説明フィールドの出力を確認するには水平にスクロールする必要があります。

```
<lightning:layout multipleRows="false">
    <lightning:layoutItem size="6">{!v.rec.Name}
    </lightning:layoutItem>

    <lightning:layoutItem size="6">{!v.rec.Description__c}
    </lightning:layoutItem>
</lightning:layout>
```

開発者は、モバイルおよびタブレットデバイスに応答できるようにコンポーネントをどのように変更する必要がありますか?

A.

```
<lightning:layout multipleRows="true">
    <lightning:layoutItem size="12" largeDeviceSize="6">
{!v.rec.Name}
    </lightning:layoutItem>

    <lightning:layoutItem size="12" largeDeviceSize="6">
{!v.rec.Description__c}
    </lightning:layoutItem>
</lightning:layout>
```

B.

```
<lightning:layout multipleRows="false">
    <lightning:layoutItem smallDeviceSize="12"
```

C.

```
<lightning:layout multipleRows="true">
    <lightning:layoutItem size="12" largeDeviceSize="6">{!v.rec.Name}
    </lightning:layoutItem>

    <lightning:layoutItem size="12" largeDeviceSize="6">{!v.rec.Description__c}
    </lightning:layoutItem>
</lightning:layout>
```

D.

```
<lightning:layout multipleRows="false">
    <lightning:layoutItem smallDeviceSize="12" largeDeviceSize="6">{!v.rec.Name}
    </lightning:layoutItem>

    <lightning:layoutItem smallDeviceSize="12" largeDeviceSize="6">{!v.rec.Description__c}
    </lightning:layoutItem>
</lightning:layout>
```

正解: **A (**コメントを発表する**)**

オプション A は、コンポーネントをモバイルおよびタブレットデバイス向けにレスポンシブに変更する正しい方法です。オプション A では、lightning:layout コンポーネントと lightning:layoutItem コンポーネントを使用して、さまざまな画面サイズやデバイスに適応できるレスポンシブレイアウトを作成します。lightning:layout コンポーネントは複数の行と列を持つことができるレイアウトを定義し、lightning:layoutItem コンポーネントはデバイスに基づいて異なるサイズにまたがるアイテムを定義します。smallDeviceSize 属性と largeDeviceSize 属性を使用することで、開発者は小型デバイスまたは大型デバイスでアイテムが占める列数を指定できます。例えば、開発者は次のコードを使用して、名前と説明のフィールドを大型デバイスでは同じ行に、小型デバイスでは別々の行に表示するレスポンシブレイアウトを作成できます。
<lightning:layout multipleRows="true"> <lightning:layoutItem smallDeviceSize="12" largeDeviceSize="6">{!v.rec.Namelightning:layoutItem> <lightning:layoutItem smallDeviceSize="12" largeDeviceSize="6">{!v.rec.Descriptionlightning:layoutItem> </lightning:layout>

質問: **73**

次のコード スニペットを考えてみましょう。

```
public static List<Account> getAccounts(Date thisDate, Id goldenRT){
List<Account> accountList = [Select Id, Name, Industry FROM Account WHERE CreatedDate = :thisDate OR RecordTypeId = :goldenRT];
return accountList;
}
```

Apex メソッドはアカウントのデータ量が多い環境で実行されており、クエリのパフォーマンスが低下しています。

結果セット全体を保持しながらクエリを最適に実行するには、開発者はどの手法を実装する必要がありますか?

A. CreateData と RecordType の値を組み合わせる数式フィールドを作成し、数式に基づいて Piter を計算します。
B. メソッドに @Future アノテーションを付けます。
C. クエリを 2 つの個別のクエリに分割し、2 つの結果セットを結合します。
D. Database、queryLocation メソッドを使用してアカウントを取得します。
正解**: C (**コメントを発表する**)**

質問**: 74**
Cloud Kicks の Salesforce 管理者は、米国のすべての郵便番号と、その郵便番号が属する Cloud Kicks の販売地域を保存するための Region__c というカスタム オブジェクトを作成しました。
オブジェクト名: Region__c
フィールド: Zip_Code__c (テキスト)、Region_Name__c (テキスト)
Cloud Kicks は、リードの郵便番号に基づいて地域情報を入力するトリガーをリードに作成したいと考えています。このリクエストを満たす最も効率的なコードセグメントはどれですか？1234
A. 5678
ジャワ
Set<String> zips = 新しい Set<String>();
for(リード l : トリガー.new) {
if(l.PostalCode != Null) {
zips.add(l.PostalCode);
}
}
for(リード l : トリガー.new) {
リスト<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips]; for(Region__c r : regions) { if(l.PostalCode == r.Zip_Code__c) {
B. Region__c = r.Region_Name__c;
}
}
}
C. Java
Set<String> zips = 新しい Set<String>();
for(リード l : トリガー.new) {
if(l.PostalCode != Null) {
zips.add(l.PostalCode);
}
}
リスト<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c WHERE Zip_Code__c IN :zips]; for(Lead l : Trigger.new) { for(Region__c r : regions) { if(l.PostalCode == r.Zip_Code__c) {
D. Region__c = r.Region_Name__c;

}
}
}
E. ジャワ
for(リードl : トリガー.new) {
Region__c reg = [SELECT Region_Name__c FROM Region__c WHERE Zip_Code__c = :l.
郵便番号];
F. Region__c = reg.Region_Name__c;
}
G. ジャワ
Set<String> zips = 新しい Set<String>();
for(リードl : トリガー.new) {
if(l.PostalCode != Null) {
zips.add(l.PostalCode);
}
}
List<Region__c> regions = [SELECT Zip_Code__c, Region_Name__c FROM Region__c
WHERE Zip_Code__c IN :zips]; Map<String, String> zipMap = new Map<String, String>();
for(Region__c r : regions) { zipMap.put(r.Zip_Code__c, r.Region_Name__c);
}
for(リードl : トリガー.new) {
if(l.PostalCode != Null) {
H. Region__c = zipMap.get(l.PostalCode);
}
}
正解**: D (**コメントを発表する**)**
Salesforce Apex のベストプラクティスに従うには、特にトリガーを扱う際には、コードを「一括処理」する必要があります。つまり、プラットフォームガバナの制限（トランザクションあたり 100 件の SOQL クエリの制限など）に達することなく、数百件のレコードを一度に効率的に処理できるコードが必要です。
オプション D は、標準の Set-Query-Map パターンに従うため、最も効率的なセグメントです。
* 設定: 最初にトリガー内のすべてのリードを反復処理して、一意の郵便番号を Set<String> に収集します。
* クエリ: ループ外で単一のSOQLクエリを実行し、関連するRegion__cレコードのみを取得します。これにより、オプションAとCで見られる「ループ内のSOQL」アンチパターンを回避できます。これらのアンチパターンでは、トリガが100件を超えるレコードを処理した場合に失敗します。
* マップ: クエリ結果をMap<String, String>に格納します。マップは検索複雑度が$0(1)$であるため、データ取得に非常に効率的です。
* 最終ループ: 最終ループでは、コードは zipMap.get() を使用して領域を検索します。

対照的に、オプションBはクエリに関しては一括処理されていますが、データのマッチングにはネストされたループを使用しています。200件のリードと200の地域がある場合、ネストされたループは40,000回の反復処理 200×200）を実行し、大規模なデータセットでは 最大CPU時間を超えました」というエラーが発生する可能性があります。オプションDは、マップを使用して郵便番号を直接検索することで、この不要なCPUオーバーヘッドを排除し、最もスケーラブルでパフォーマンスの高いソリューションとなっています。

質問: **75**
Universal Containers は Big Object を使用して、customer_Transaction_ と呼ばれるほぼ 10 億件の顧客トランザクションを保存します。
顧客トランザクション b のフィールドは次のとおりです。

次のフィールドは、Customexr_Transaction__b オブジェクトのインデックス フィールドとして識別されています: Account__g、Program__c、および Transaction Date_ co。
customer_Transaction_b Big Object で有効な SOQL クエリはどれですか?

A.

B.

正解: **(**正解を表示します**)**

オプションAは、customer_Transaction_b Big Objectに対する有効なSOQLクエリです。オプションAでは、Big Objectのインデックス項目（Account__c、Program__c、Transaction_Date__c）を使用してレコードをフィルタリングします。また、このクエリでは、Big Objectクエリに必要な=演算子を使用して、インデックス項目の正確な値を一致させます。さらに、SELECT Id、Account__c、Program__c、Transaction_Date__c句を使用してBig Objectのインデックス項目のみを返します。これは、すべての項目を返すよりも効率的です。また、LIMIT 1000句を使用して、返されるレコード数をBig Objectクエリの上限である1000に制限します。参考：[Big ObjectのSOQL]、[Big Objectインデックス]

質問: **76**

開発者は、ユーザが特定の取引先を名前で検索できるようにする Visualforce ページを作成したいと考えています。アカウントが見つかった場合は、アカウントの詳細が画面に表示されます。アカウントが見つからない場合は、ユーザーにエラー メッセージが表示されます。

これはどのように達成できますか？（2つ選んでください。）

A. accountaddErrorQ メソッドを使用してエラー メッセージを追加します。

B. <apex:pageMessages> タグを使用してエラーメッセージを表示します

C. (apex:情報)タグを使用してエラーメッセージを表示します

D. ApexPages.addMessage() メソッドを使用してエラー メッセージを追加します。

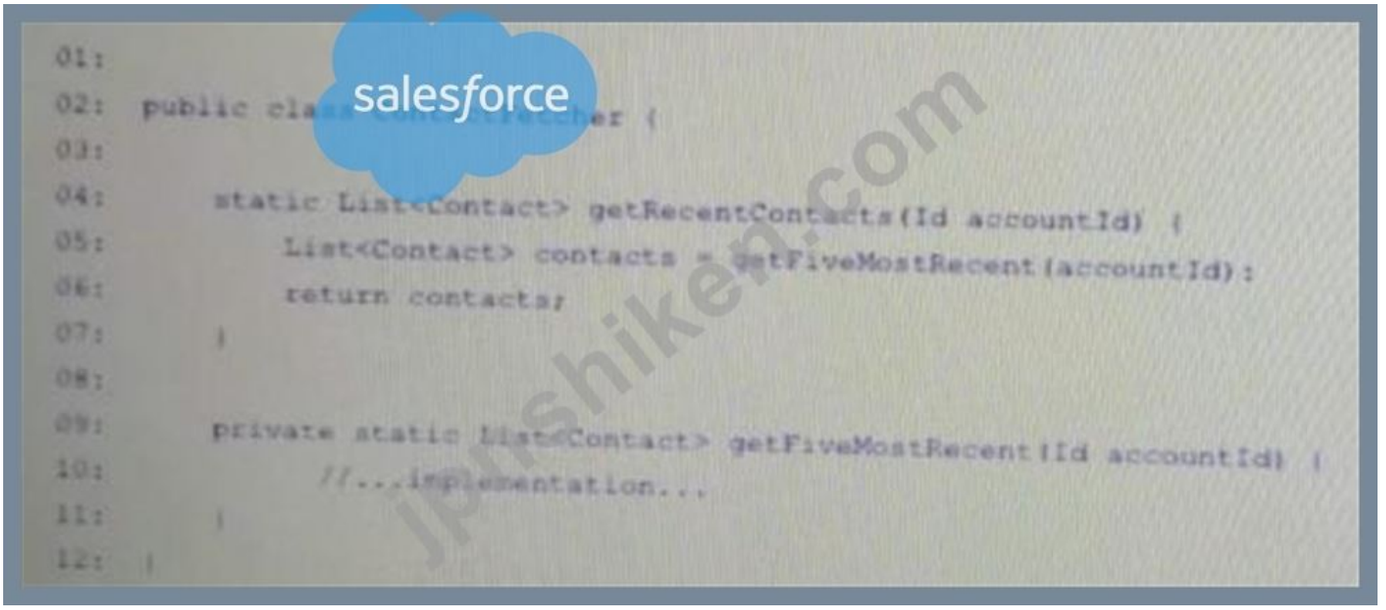
正解: **B,D (**コメントを発表する**)**

質問: **77**

開発者は、カスタム画面用に Visualforce コンポーネントを作成するか Lightning コンポーネントを作成するかを決定しようとしています。

最終的な決定に影響を与える機能の考慮事項はどれですか?
A. 画面は Lightning Experience UI からアクセスできる必要がありますか?
B. サードパーティのアプリケーションを使用せずに、画面を PDF としてレンダリングする必要がありますか?
C. 画面は JavaScript フレームワークを使用しますか?
D. 画面はモバイル アプリ経由でアクセスされますか?
正解: **B** (コメントを発表する)
VisualforceはページをPDFとしてレンダリングできますが、Lightningコンポーネントではレンダリングできません。PDFレンダリングが必要な場合、この機能は非常に重要です。
参考資料: Visualforce 開発者ガイド - ページを PDF としてレンダリングする

質問: **78**
開発者は、取引先の最近連絡した 5 つの取引先責任者を表示する取引先レコード ページの Lightning Web コンポーネントを作成しました。Apex メソッド getRecentContacts は取引先責任者のリストを返し、コンポーネントのプロパティに関連付けられます。



Apex メソッドを配線できるようにするには、上記の頌歌でどの 2 行を変更する必要がありますか?
2つの答えを選択してください
A. 行 04 に public を追加します。
B. ライン 09 からプライベートを削除します。
C. @AuraEnabled (cacheabletrue) を 03 行目に追加します。
D. 08 行目に AuraEnanled(cacheabletrue) を追加します。
正解: (正解を表示します)

質問: **79**
開発者は、Apex で商談トリガから呼び出される複雑なコミッション計算エンジンを構築しています。QA中に、計算が間違っていることが報告されました。

開発者は、開発者サンドボックスに代表的なテスト データと合格したテスト メソッドを持っています。
開発者がコードを実行し、重要な行で一時停止してさまざまな Apex 変数の値を視覚的に検査するために使用できるツールまたはテクニックを 3 つ選択してください。
3 つの答えを選択してください
A. Apex 対話型デバッガ
B. ワークベンチ
C. 開発者コンソール
D. ブレークポイント
E. Apex リプレイデバッガ
正解**: A,C,E (**コメントを発表する**)**
コミッション計算エンジンをデバッグおよびトラブルシューティングするために、開発者は Salesforce が提供するいくつかのツールを使用して、Apex コードの実行を検査および一時停止できます。
Apexインタラクティブデバッガ :Apexコード実行のリアルタイムデバッグを可能にします。このツールを使用すると、開発者はブレークポイントの設定、コードのステップ実行、変数の検査、式の評価を行うことができます。
開発者コンソール：開発者コンソールでは対話型のデバッグはできませんが、コード実行を記録したログを表示する機能は備えています。デバッグログを調べることで、実行フローや様々な時点における変数の値を把握できます。
Apex Replay Debugger: このツールは、Salesforce Extensions for Visual Studio Code の一部です。開発者は、コードを1行ずつステップ実行しているかのようにデバッグログを再生できるため、実行中の特定のポイントにおける変数の状態を検査するのに非常に便利です。
参考文献:
Apex インタラクティブ デバッガ
開発者コンソールのデバッグ
Apex リプレイ デバッガー

質問**: 80**
カスタムインターフェースの一部として、開発者チームはさまざまな新しい Lightning Web コンポーネントを作成します。各コンポーネントはトースト メッセージを使用してエラーを処理します。開発が完了すると、すべてのコンポーネントが同じ Lightning ページに追加されます。
受け入れテスト中に、コンポーネントの読み込み中にエラーが発生したときに表示されるトースト メッセージの長いチェーンについてユーザーから苦情が寄せられました。
ユーザー エクスペリエンスを向上させるために開発者が実装する必要がある 2 つのテクニックはどれですか?
2 つの答えを選択してください
A. Lightning Web コンポーネントを使用して、すべてのエラーを集計して表示します
B. window.alert() メソッドを使用してエラー メッセージを表示します
C. <tamplate> タグを使用して、インプレース エラー メッセージを表示します。

D. 各コンポーネントの public プロパティを使用して、エラー メッセージを表示します。
正解**: (**正解を表示します**)**
このシナリオでは、同じ Lightning ページで複数のコンポーネントが使用され、各コンポーネントが個別にエラー処理を管理する場合によく発生する問題について説明します。その結果、ユーザーにとって負担になる可能性のある 「トースト メッセージの長いチェーン」が発生します。
ユーザー エクスペリエンスを向上させるために、開発者は次のテクニックを実装できます。
A: Lightning Webコンポーネントを使用して、すべてのエラーを集約して表示します。この方法では、すべてのエラーメッセージを処理する一元化されたコンポーネントを作成します。このコンポーネントは、他のコンポーネントからエラー通知を受け取り、ユーザーフレンドリーな方法で表示します。このアプローチの利点は、エラーメッセージに単一の一貫性のあるインターフェースを提供することで、複数のトーストメッセージによって発生する混乱を軽減できることです。
D: 各コンポーネントのパブリックプロパティを使用してエラーメッセージを表示する :パブリックプロパティを使用することで、コンポーネントはエラー状態を親コンポーネントまたはオーケストレーションコンポーネントに公開でき、親コンポーネントまたはオーケストレーションコンポーネントはこれらのエラーを単一の統合された方法で表示できます。これにより、複数のトースト通知が表示される問題を回避し、より統合されたエラー処理エクスペリエンスを実現できます。
提示された他のオプションはあまり理想的ではありません。
B: window.alert() メソッドの使用は、エラーを表示するための邪魔な方法であると考えられており、Salesforce Lightning デザイン システムと一致していないため、プロフェッショナルな Salesforce 環境では推奨されません。
C: <template> タグを使用してインプレース エラー メッセージを表示することは、コンポーネント自体の中にエラー メッセージを表示する実行可能なオプションですが、シナリオで説明されているように、異なるコンポーネントからの複数のエラーが積み重なる問題には対処しません。
参考文献:
集中型のエラー処理コンポーネントの構築: Lightning Web コンポーネント開発者ガイド - エラー処理 パブリック プロパティの実装: Lightning Web コンポーネント開発者ガイド - パブリック プロパティ

質問**: 81**
カスタム開発の一環として、開発者は Lightning コンポーネントを作成して、特定の商談が時間とともにどのように進行するかを示します。次のフィールドのいずれかが変更された場合、コンポーネントは日付スタンプを表示する必要があります。
*金額、確率、段階、または完了日
開発者は、表示する必要があるデータにどのようにアクセスする必要がありますか?
A. Lightning コンポーネントで OpportunityHistory Change Data Capture イベントにサブスクライブします。
B. 各フィールドの商談でカスタムのカスタム日付フィールドを作成して、前の日付を追跡し、日付フィールドの SOQL クエリを実行します。

C. Lightning コンポーネントで Opportunity Change Data Capture イベントにサブスクライブします。
D. OpportunityHistory オブジェクトで、Amount、Probability、Stage、および Close Date の SOQL クエリを実行します。
正解: **B** (コメントを発表する)

質問: **82**
Aura コンポーネントから呼び出され、クラスにラップされたデータを返す以下のコントローラーコードを考えてみましょう。

開発者は、クエリがそれぞれ 1 つのレコードを返し、Aura コンポーネントにエラー処理があることを確認しましたが、コントローラー getSemeData を呼び出したときにコンポーネントが何も返さないことを確認しました。
'なにが問題ですか？
A. MyDatsWrapper などの Apex クラスのインスタンスを Lightning コンポーネントに返すことはできません。
B. メンバーの名前とオプションは公開宣言されるべきではありません。
C. メンバーの名前とオプションにはゲッターとセッターを含めることはできません。
D. メンバーのカメとクラス MyDataWrapper のオプションにも @AuraEnabled のアノテーションを付ける必要があります。
正解: **D** (コメントを発表する)

質問: **83**

ある企業には、いくつかの異なるオブジェクトといくつかの共通オブジェクトを含むさまざまなSalesforce 組織があり、すべての異なる組織で共通のオブジェクト レコードを作成、取得、更新できる単一の Java アプリケーションを構築したいと考えています。
アプリケーションはどの統合方法を使用する必要がありますか?
A. パートナー WSDL を使用した SOAP API
B. Apex REST Web サービス
C. Enterprise WSDL を使用した SOAP API
D. メタデータ APT
正解**: (**正解を表示します**)**
Partner WSDL を使用した SOAP API は、さまざまな組織で共通のオブジェクトレコードを作成、取得、更新するために使用できる統合方法です。SOAP API は、Java などの SOAP をサポートする任意のプログラミング言語から Salesforce のデータや機能にアクセスできる Web サービスです。Partner WSDL は、Salesforce メタデータに疎結合された SOAP クライアントクラスを生成するために使用できる WSDL (Web サービス記述言語) ファイルの一種です。Partner WSDL は、異なる組織のさまざまなオブジェクトや項目を処理し、実行時にスキーマを動的に検出できます。SOAP API と Partner WSDL を使用することで、開発者はオブジェクトや項目の違いに関係なく、あらゆる Salesforce 組織と連携できる単一の Java アプリケーションを作成できます。参照: [SOAP API]、[Partner WSDL]

質問**: 84**
開発者は、取引先の最近連絡した 5 つの取引先責任者を表示する取引先レコード ページの Lightning Web コンポーネントを作成しました。Apex メソッド getRecentContacts は取引先責任者のリストを返し、コンポーネントのプロパティに関連付けられます。

Apex メソッドを配線できるようにするには、上記の頌歌でどの 2 行を変更する必要がありますか?
2つの答えを選択してください
A. @AuraEnabled (cacheabletrue) を 03 行目に追加します。

B. ライン 09 からプライベートを削除します。
C. 行 04 に public を追加します。
D. 08 行目に AuraEnanled(cacheabletrue) を追加します。
正解: **(**正解を表示します**)**

質問: **85**
テストクラスで Vizualforce ページを
初期化するベストプラクティスは何ですか?
A. テストを使用します。setCurrentPage (Page.MyTeatPage);
B. テストを使用します。currantPage .getParamatars put (MyTaestPaga) ;
C. コントローラーを使用します。現在のページ。setPage (MyTastfage) ;
D. テストを使用します。setCurrentPage MyTestPags;
正解: **A (**コメントを発表する**)**
テストクラスでVisualforceページを初期化する際のベストプラクティス
は、Test.setCurrentPage(Page.MyTestPage)を使用することです。このメソッドは、テストクラスの現在のページコンテキストを設定し、開発者がページパラメータ、コントローラ変数、標準コントローラメソッドにアクセスできるようにします。これにより、開発者はVisualforceページとそのコントローラの機能と動作をさまざまなシナリオでテストできます。その他のオプションは、構文が無効であるか、現在のページコンテキストが設定されません。参考 :[Test.setCurrentPageメソッド]、[カスタムコントローラとコントローラ拡張のテスト]、[Visualforceコントローラのテスト]

質問: **86**
次のアノテーションのうち、単一の頂点メソッドを呼び出す正しい方法はどれですか?
A. @InvokableApex()
B. @InvokableAction()
C. @InvokableMethod()
正解: **(**正解を表示します**)**

質問: **87**
ある会社には、ユーザの入力に基づいて結果のリストを返す Apex コントローラを呼び出すことで、ユーザが特定のオブジェクト タイプのレコードを検索できるカスタム コンポーネントがあります。検索が完了すると、searchComplete イベントが発生し、結果がイベントの results 属性に入れられます。このコンポーネントは他のコンポーネント内で使用されるように設計されており、1 つのページに複数回表示される場合があります。
検索完了時にイベントを発生させるために追加する最適なコードは何ですか?

A.

```
var evt = $A.get("e.c.searchComplete");
evt.setParams({results: results});
evt.fire();
```

B.

```
var evt = $A.get("e.c.searchComplete");
evt.set("v.results", results);
evt.fire();
```

C.

```
var evt = component.getEvent("searchComplete");
evt.setParams({results: results});
evt.fire();
```

D.

```
var evt = component.getEvent("searchComplete");
evt.set("v.results", results);
evt.fire();
```

正解: **(**正解を表示します**)**

質問: **88**

Lightning Web コンポーネントはシステム内に存在し、コンテキスト内のレコードに関する情報をメダルとして表示します。Salesforce 管理者は、Lightning アプリケーションビルダー内でこのコンポーネントを使用する必要があります。開発者が XML リソースファイル内で構成する必要がある 2 つの設定はどれですか?

2 つの答えを選択してください

A. ターゲットを lightning_RecordPage に指定します

B. IsExused=d 属性を true に設定します。

C. ターゲットを lightning_AppPage に指定します

D. IsVisible 属性を true に設定します。

正解: **(**正解を表示します**)**

Lightning WebコンポーネントをLightning App Builderで利用できるようにするには、開発者は.xmlファイルでコンポーネントを配置できるターゲット（レコードページなど）を指定し、isExposed属性をtrueに設定する必要があります。参考資料 :フロー画面、クイックアクション、App Builder、Experience Builder向けのLightning Webコンポーネントの公開

質問: **89**
Account オブジェクトには、アカウントに必要な監査の種類を指定するために使用されるフィールド Audit_Code_c と、割り当てられた監査人であるユーザーのルックアップ zudizar_c があります。アカウントが最初に作成されるとき、ユーザーは Audit_Code を指定します。組織内の各ユーザーには、固有のテキスト フィールド Audit_Code _e があり、これはアカウントの Auditor_c フィールドに正しいユーザーを自動的に割り当てるために使用されます。

```
trigger AccountTrigger on Account(before insert) {
    AuditAssigner.setAuditor(Trigger.new);
}

public class AuditAssigner {
    public static void setAuditor(List<Account> accounts) {
        for (User u : [SELECT Id, Audit_Code__c FROM User]) {
            for (Account a : accounts) {
                if (u.Audit_Code__c == a.Audit_Code__c) {
                    a.Auditor__c = u.Id;
                }
            }
        }
    }
}
```


コードの効率を最大限に最適化するには何を変更する必要がありますか?
2 つの答えを選択してください
A. 最初の SOQL クエリを追加して、すべての個別の監査コードを取得します。
B. 監査コードからアカウントへの Map<string, List<Account>> を構築します。
C. コードを監査するためのアカウント ID のマップ<Id, リスト<文字列>>を構築します。
D. SOQL クエリに WHERE 句を追加して、監査コードでフィルタリングします。
正解: (正解を表示します)
監査コードに基づいてアカウントの監査担当者フィールドにユーザーを割り当てるときにコードの効率を最適化するには、次の変更を考慮する必要があります。

B: 監査コードとアカウントのマッピング（Map<string, List<Account>>）を作成することで、監査コードごとにアカウントを効率的にグループ化できます。これにより、同じ監査コードを持つアカウントを1回の操作で処理しやすくなり、必要なSOQLクエリとDML操作の数を削減できます。
D: SOQLクエリにWHERE句を追加して監査コードでフィルタリングすると、クエリによって返されるレコード数が制限されます。つまり、コードは一致する監査コードを持つ関連するユーザーレコードのみを処理するため、処理時間とメモリ消費量が削減されます。
オプションAは、処理対象のアカウントのコンテキストを考慮せずにすべての異なる監査コードをクエリするのは非効率的であるため、推奨されません。オプションCは、アカウントIDと監査コードのマッピングを作成する必要があり、監査担当者をアカウントに効率的に割り当てることができないため、それほど効果的ではありません。
参考文献:
Salesforce 開発者ガイド: マップ: Apex マップ
Salesforce 開発者ガイドの SOQL クエリ: SOQL SELECT 構文

質問: **90**
ポイント ツー ポイント統合の一環として、開発者は外部の Web サービスを呼び出す必要があります。これは、需要が高いため、応答を提供するのに長い時間がかかります。要求の一部として、開発者はコールアウトを行う前にエンド ユーザーからキー入力を収集する必要があります。
これらのビジネス要件を実装するために開発者が使用する必要がある 2 つの要素はどれですか?
2つの答えを選択してください
A. Continuation オブジェクトを返す Apex メソッド
B. プロセスビルダー
C. 今すぐスクリーン
D. Lightning Web コンポーネント
正解: **(正解を表示します)**

質問: **91**
開発者は、Lightning Web コンポーネントのスタイルと動作を制御するために変数を保存する必要があります。本番環境とすべてのサンドボックスの両方で変数をテストできるようにするには、どの機能を使用できますか？
A. カスタムメタデータ
B. カスタムオブジェクト
C. カスタム設定
D. カスタム変数
正解: **A (コメントを発表する)**
異なる環境間で動作やスタイルを制御するアプリケーション設定の場合、カスタムメタデータ型（オプションA）が業界標準の選択肢です。カスタム設定やカスタムオブジェクトと比較したカスタムメタデータの主な利点は、レコード自体がメタデータとして扱われることです。つまり、変更セット、Salesforce CLI、またはパッケージを使用してデプロイできます。開発者がコードをSandboxから本番環境に移行すると、関連する設定レコードもデプロイに含まれるた

め、Lightning Webコンポーネントは手動でデータを設定することなく、両方の環境で同じように動作します。
さらに、カスタムメタデータは非常に「テストしやすい」という特徴があります。Apex単体テストでは、(SeeAllData=true) アノテーションを必要とせずにカスタムメタデータレコードをクエリできます。コンポーネントのApexコントローラロジックがこれらの変数に依存している場合でも、テストは堅牢で環境に依存しません。カスタム設定（オプションC）とカスタムオブジェクト（オプションB）はレコードをデータとして保存しますが、変更セット経由では展開できず、新しいサンドボックスや本番組織ごとに手動で入力またはデータを読み込む必要があるため、設定エラーのリスクが高まります。カスタムメタデータは、コンポーネントレベルの定数と動作フラグを管理するための最も「拡張可能」で展開しやすい方法を提供します。

質問: **92**
コンプライアンス目的のため、企業は組織内での製品の長期使用を追跡する必要があります。ログに記録する必要がある情報は複数のオブジェクトから収集されるため、時間が経つにつれて、数億のレコードが存在するようになると予測されています。
開発者はこれを実装するために何を使用する必要がありますか?
A. 監査証跡の設定
B. フィールド監査証跡
C. 大きなオブジェクト
D. フィールド履歴の追跡
正解: **C** (コメントを発表する)
Big Objectは、Salesforce内で大量のデータを保存・管理するためのソリューションを提供します。時間の経過とともに膨大なレコード数に蓄積される長期的な使用状況データを追跡する場合、コンプライアンスの観点から最適です。参考資料：Salesforce開発者ガイド - Big Objects

質問: **93**
開発者は、カレンダーを表示する Lightning Web コンポーネントを作成しています。このコンポーネントは複数の国で使用される予定です。一部のロケールでは、週の最初の日が月曜日、土曜日、または日曜日になります。「すべてのロケールのユーザーに対してカレンダーが正確に表示されるようにするには、開発者は何をすべきでしょうか?」
A. @salesforce/i18n モジュールをインポートして使用します

firstdayofweek 国際化プロパティ。
B. カスタム メタデータ タイプを使用してキーと値のペアを保存します。
C. 現在のユーザーのロケールから FirstDayofweek フィールドをクエリします。
D. コンポーネント内で UserInfo.getLocale() を使用します。
正解: **D (**コメントを発表する**)**

質問: **94**
開発者は、新しい AppFxchange アプリケーションを開発するように求められます。プログラムの機能は、ケースが特定の段階に達し、特定のレコード タイプになると、調査レコードを作成します。Salesforce インスタンスごとに異なるタイミングでアンケートが必要になるため、この機能は構成可能である必要があります。さらに、すぐに使える AppExchange アプリには、ほとんどの顧客に適用される一連のベスト プラクティス設定が付属している必要があります。
開発者はアプリのカスタム構成設定を保存およびパッケージ化するために何を使用する必要がありますか?
A. カスタムオブジェクト
B. カスタム設定
C. カスタムメタデータ
D. カスタムラベル
正解: **(**正解を表示します**)**
カスタムメタデータは、構成可能な設定を保存するのに最適です。デプロイ、パッケージ化、アップグレードが可能です。
インスタンスごとに異なる構成を許可します。参照: カスタムメタデータタイプ

質問: **95**
開発者は、新しいリードが作成されたときにタスクを挿入する新しいトリガーを作成しました。本番環境にデプロイした後、外部統合によって定期的にエラーが報告されます。
ビジネスロジックへの影響を最小限に抑えて統合が影響を受けないようにするために、開発者はどの変更を加える必要がありますか?
A. 統合を実行する前にトリガーを無効にします。
B. allOrNone を False に設定して Database メソッドを使用します。
C. insert ステートメントの後に Try/Catch ブロックを使用します。
D. 統合ユーザーのプロファイルから Apex クラスを削除します。
正解: **C (**コメントを発表する**)**

質問: **96**
Universal Containers には、取引先が顧客としてマークされたときにアカウントプランレコードを作成する取引先に対する Apex トリガーがあります。
最近、レコードによってトリガーされるフローが追加され、アカウントが顧客としてマークされるたびに、「以降の顧客」日付フィールドが今日の日付で更新されるようになりました。フローの

追加以来、アカウントがマークされるたびに 2 つのアカウント プラン レコードが作成されます。顧客として。
これが起こる原因は何でしょうか?
A. Apex トリガは、確実に 1 回だけ起動するために静的変数を使用しません。
B. フローは 〖レコードの更新」要素を使用するように構成されています。
C. Apex トリガは一括安全ではなく、for ループ内で insert を呼び出します。
D. フローは、レコードの作成時と編集のたびに評価するように構成されています。
正解: **(**正解を表示します**)**
取引先が顧客としてマークされるたびに 2 つのアカウントプランレコードが作成される理由は、フローがレコードの作成時と編集時に評価するように設定されているためです。つまり、フローは取引先が挿入されたときと、Apex トリガーによって取引先が更新されたときの両方で実行されます。フローとトリガーの両方がアカウントプランレコードを作成するため、同じ取引先に対して 2 つのアカウントプランレコードが作成されます。これを回避するには、レコードの作成時またはレコードに指定された変更が加えられたときのみ評価するようにフローを設定する必要があります。これにより、フローとトリガーが競合することがなくなり、アカウントプランレコードは 1 つだけ作成されます。参考: [レコードトリガーフロー]、[レコードトリガーフローの実行タイミングの設定]

質問: **97**
SOQL クエリのトランザクション制限は?
A. 150 (同期)、200 (非同期)
B. 150 (同期)、20 (非同期)
C. 200 (同期)、100 (非同期)
D. 20 (同期)、200 (非同期)
E. 100 (同期)、200 (非同期)
正解: **(**正解を表示します**)**

質問: **98**
Number 型のカスタム フィールド Exec_Count_c が Account オブジェクトに作成されます。a: Exec__Count_c の値が 〖」のアカウント レコードが保存されます。ワークフロー フィールドの更新は Exec_Count_c フィールドで定義され、アカウント レコードが作成または更新されるたびにその値を増やします。アカウントには次のトリガーが定義されています。
アカウントで ExecOrderTrigger をトリガーする (挿入前、更新前、挿入後、更新後){ for (Account accountInstance: Trigger.New){ if (Trigger . isBefore){ accountInstance Exec_Count_c += 1; } システム、デバッグ (accountInstance.Exec_Count_c); } }
A. 2、2、4、4
B. 1,2,3,3
C. 2,2,3,3
D. 1,2,3,4

正解: (正解を表示します)

質問: **99**
以下のマークアップを参照してください。

Lightning Web コンポーネントには、取引先名と 2 つのカスタム項目が表示されます。
荒地に存在する275。カスタム フィールドが正しく宣言され、値が設定されています。ただし、開発者は、コンポーネントの動作が遅いという苦情を受けています。
開発者はパフォーマンスを向上させるために何ができるでしょうか?
A. Layout-type ="Full1" を Layout-type="Partial" に置き換えます。
B. レイアウト タイプ ="1" をフィールド =(フィールド} に置き換えます。
C. コンポーネントにdensity="compat"を追加します。
D. コンポーネントにcache="true"を追加します。
正解: (正解を表示します)
パフォーマンスを向上させるには、開発者はlightning-record-formのfields属性を使用して必要なフィールドのみを指定し、layout-type="Full"ですべてのフィールドを読み込むのではなく、必要なフィールドのみを指定する必要があります。これにより、取得および処理されるデータの量が削減され、コンポーネントのパフォーマンスが向上します。参考資料 :Lightning Web Components Developer Guide - lightning-record-form

質問: **100**
開発者は、組織内のすべてのテスト アカウントを見つけるために次のメソッドを作成しました。

```
public static Account[] searchTestAccounts(){
      List<List<SObject>> searchList = [ FIND 'test' IN ALL FIELDS
                                         RETURNING Account(Name)];
      return (Account[]) searchList[0];
   }
```

However, the test method below fails.

```
@isTest
  public static void testSearchTestAccounts(){
      Account a = new Account(name='test');
      insert a;
      Account[] accounts = AccountFinder.searchTestAccounts();
      System.assert(accounts.size() == 1 );
   }
```


この落下テストを修正するには何を使用する必要がありますか?

A. テスト。期待されるデータを設定する loaddata

B. テスト用の組織データにアクセスするための @isTest (AllData=true を参照)

C. テスト。予想されるデータを設定する fixedSearchResults() メソッド

D. 期待されるデータを設定する @testSetup メソッド

正解: **(**正解を表示します**)**

質問: **101**

以下のテスト方法を参照してください。

```
@isTest
static void testIncrement() {
    Account acct = new Account(Name = 'Test');
    acct.Number_Of_Times_Viewed__c = 0;
    insert acct;

    AuditUtil.incrementViewed(acct.Id);
    Account acctAfter = [SELECT Number_Of_Times_Viewed__c FROM Account
WHERE Id = :acct.Id][0];

    System.assertEquals(1, acctAfter.Number_Of_Times_Viewed__c);
}
```


テスト メソッドは、値をインクリメントする @future メソッドを呼び出します。

Number_of_Times_Viewed__c 値。アサーションは失敗します。

閲覧回数_c は 0 です。

これを修正する最適な方法は何ですか?
A. Test.atartTeat() を前に追加し、Teat.stopTest() を inser= acct の後に追加します。
B. アサーションを system.asserciquals (0, acctAfter Number _Cf_Timea_Viewed__c) に変更します。
C. 初期化を acct に変更します。Number_Of_Times_Viewed_c = 1。
D. 前にrest.staztTest()を追加し、AuditUcil.incrementViewedの後にTest.stopTess()を追加します。
正解: **(**正解を表示します**)**

質問: **102**
Universal Containers は、ユーザー設定を階層カスタム設定 User_prefs_c に、チェックボックスフィールド show_Help_c とともに保存します。会社レベルのデフォルトは組織レベルで保存されますが、ユーザー レベルで上書きされる可能性があります。ユーザーが設定を上書きしていない場合、デフォルトを使用する必要があります。
現在のユーザーの Show_Help_c 設定を取得するにはどうすればよいですか?
A. ブール値 show = User_Prefs_c, getValues ( ). _Help_c を表示;
B. ブール値の show = User prefs_c, Show_Help_c;
C. ブール値 show = User_Prefs_c, getinstance( ), Show_Help _c;
D. ブール値の show = User_Prefs_c, getvaluesUserInfo.getUserid()).Show_Help_c;
正解: **D (**コメントを発表する**)**

質問: **103**
コンプライアンス目的のため、企業は組織内での製品の長期使用を追跡する必要があります。ログに記録する必要がある情報は複数のオブジェクトから収集されるため、時間が経つにつれて、数億のレコードが存在するようになると予測されています。
開発者はこれを実装するために何を使用する必要がありますか?
A. 監査証跡の設定
B. フィールド監査証跡
C. 大きなオブジェクト
D. フィールド履歴の追跡
正解: **(**正解を表示します**)**
これを実装する最良の方法は、Big Object を使用することです。Big Object は、Salesforce プラットフォーム上で膨大な量のデータを保存および管理できるカスタムオブジェクトです。Big Object は数億件のレコードを処理でき、非同期 SOQL クエリをサポートします。Big Object はパフォーマンスと拡張性にも最適化されており、レポート、ダッシュボード、トリガなどの他の Salesforce 機能と統合できます。設定監査証跡、項目監査証跡、項目履歴追跡は、追跡できる項目数、レコード数、または保持期間に制限があるため、このシナリオには適していません。参考資料: [Big Object]、[Trailhead: Big Object の基礎]

質問: **104**
開発者は、無限ループをトリガーするアプリケーション イベントを作成します。

この問題の原因は何でしょうか?
A. イベントはカスタム レンダラから発生します。
B. イベントにはプロジェクトに複数のハンドラーが登録されています。
C. イベント ハンドラーがトリガーを呼び出します。
D. イベントは δntouchend」で発生し、処理されません。
正解: **(**正解を表示します**)**
アプリケーションイベントの作成時に無限ループが発生する原因として考えられるのは、イベントがカスタムレンダラーから発行されていることです。カスタムレンダラーとは、コンポーネントのデフォルトのレンダリング動作をオーバーライドするJavaScriptファイルです。カスタムレンダラーがアプリケーションイベントを発行すると、そのコンポーネント、またはイベントを処理する他のコンポーネントの再レンダリングがトリガーされる可能性があります。これにより、イベントが再度発行され、無限ループが発生する可能性があります。プロジェクトに複数のハンドラーが登録されているイベント、イベントハンドラーがトリガーを呼び出すイベント、または δntouchend」で発行されて未処理のイベントは、無限ループを引き起こす可能性は低いです。参考 :[アプリケーションイベント]、[カスタムレンダラー]

質問: **105**
開発者は、Salesforce モバイルアプリで使用される App Builder でページを作成しています。
ページが最適なパフォーマンスで動作するようにするために、開発者が従うべき 2 つの方法はどれですか?
2つの答えを選択してください
A. ページに表示されるコンポーネントを 5 つに制限します。
B. App Builder のパフォーマンス分析でページを分析します。
C. タブとアコーディオン コンポーネントの数を制限します。
D. レコード詳細ページのフィールドを 25 に制限します。
正解: **(**正解を表示します**)**

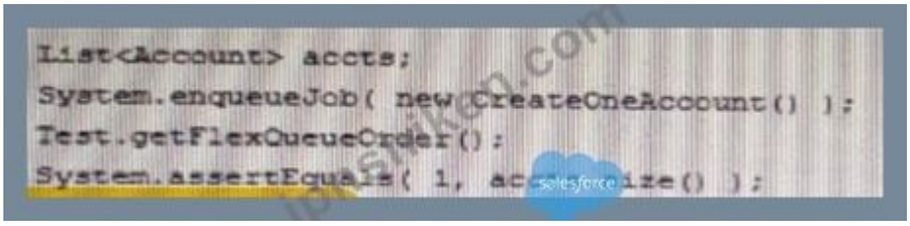
質問: **106**
営業チームが個々の顧客レコードを表示する場合、顧客の最近のやり取りを確認する必要があります。これらのやり取りは、販売注文、電話、またはケースです。最近のやり取りの日付範囲は、顧客レコードの種類ごとに異なります。
これはどのように達成できますか？
A. バッチ Apex を使用して、顧客ビュー画面がロードされたときに最新のインタラクションをクエリします。
B. Lightning フローを使用して顧客のレコードタイプを読み取り、最近のやり取りに対して動的クエリを実行して、表示ページに表示します。
C. Lightning コンポーネントを使用して、Lightning ページから design:attribute を使用して渡されたレコード タイプに基づいて対話を照会および表示します。
D. 顧客記録ページで動的フォームを使用して、最近のやり取りを表示します。

正解: (正解を表示します)

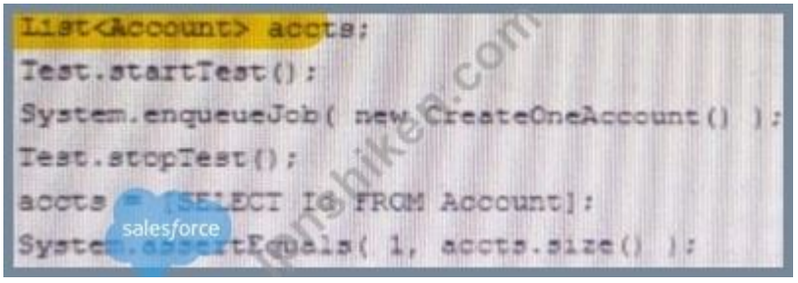
質問: **107**

CreateOneAccount クラスが 1 つのアカウントを作成し、Queuetable インターフェースを実装すると仮定すると、どの構文が Apex コードをテストしますか?

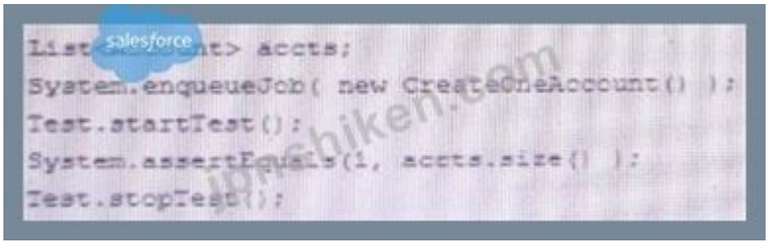
A.



B.



C.



D.

正解: (正解を表示します)

質問: **108**

開発者は、特定の取引先レコードが Visualforce コントローラのテストクラスでテストされていることをどのように確認する必要がありますか?

A. アカウントを Salesforce に挿入し、テスト クラスでページ参照をインスタンス化し、システムを使用します。setFarentRecordId() .get() を使用してアカウント ID を設定します。
{of テスト クラスでページ参照をインスタンス化し、テスト クラスにアカウントを挿入し、=seeAllData=trus を使用してアカウントを表示します。

B. テスト クラスでページ参照をインスタンス化し、テスト クラスにアカウントを挿入し、system.setFarentRecordrd() .get() を使用してアカウント ID を設定します。

C. テスト クラスにアカウントを挿入し、テスト クラスでページ参照をインスタンス化し、System.currentFageReference() .getFarameters() .put{} を使用してアカウント ID を設定します。

正解: **C** (コメントを発表する)

質問: **109**

開発者は次のテスト メソッドを作成しました。

```
@isTest(SeeAllData= true)
  public static void testDeleteTrigger(){

      Account testAccount = new Account(name = 'Test1');
      insert testAccount;

      List<Account> testAccounts = [SELECT Id, Name from Account WHERE Name like 'Test%'];
      System.assert(testAccounts.size() > 0);

      delete testAccounts;
     testAccounts = [SELECT Id, Name from Account WHERE Name like 'Test%'];
     System.assert(testAccounts.size() == 0);
```


開発者組織には、名前が ₸est」で始まるアカウントが 5 つあります。開発者は、開発者コンソールでこのテストを実行します。

テストコードが実行された後、どのステートメントが真になりますか?

A. テストは失敗します。

B. 名前が ₸est」で始まるアカウントは存在しません。

C. 名前が ₸est」で始まるアカウントは 5 つあります。

D. 名前が ₸est」で始まるアカウントは 6 つあります。

正解: (正解を表示します)

SeeAllData=true アノテーションが使用されているため、テストクラスは独自のテストデータを持たず、組織内のデータに依存しています。組織には既に ₸est」で始まる名前のアカウントが存在するため、テストメソッドはスコープ内で作成されたアカウントのみを削除するため、削除後にアカウントが存在しないというアサーションは失敗します。

参考文献:

Apex 開発者ガイド

質問: **110**
Azure コンポーネントを表す次のコード スニペットを検討してください。

コンポーネントをクイック アクションとして使用できるようにするために、開発者が実装できる 2 つのインターフェイスはどれですか?
2つの答えを選択してください
A. ライトニング QuickActionAPI
B. hasObjectName を強制する
C. 強制:lightningQuickActionWithoutHeader
D. フォース:ライトニングクイックアクション
E. 強制: hasRecordId
正解: (正解を表示します)

質問: **111**
Universal Containers は、外部 REST API との頻繁な対話を必要とする Salesforce アプリケーションを開発します。
コードの重複を避け、保守性を向上させるには、コードを再利用するために APL 統合をどのように実装する必要がありますか?
A. API エンドポイントごとに個別の Apex クラスを使用して、統合ロジックをカプセル化します。
B. API 統合コードを必要とする各 Apex クラスに直接組み込みます。
C. AFL 統合用の再利用可能な Apex クラスを作成し、関連する Apex クラスから呼び出します。
D. APT 統合コードを静的リソースとして保存し、各 Apex クラスで参照します。
正解: (正解を表示します)
API統合用に再利用可能なApexクラスを作成することで、コードのカプセル化と保守性が向上します。このクラスは、APIとのやり取りを必要とする他のApexクラスから呼び出すことができるため、コードの重複を防ぐことができます。
参考資料: Apex 開発者ガイド - Apex クラス

質問: **112**
以下のテスト方法を参照してください。

```
@isTest
static void testAccountUpdate() {
    Account acct = new Account(Name = 'Test');
    acct.Integration_Updated__c = false;
    insert acct;

    CalloutUtil.sendAccountUpdate(acct.Id);

    Account acctAfter = [SELECT Id, Integration_Updated__c FROM Account
WHERE Id = :acct.Id][0];

    System.assert(true, acctAfter.Integration_Updated__c);
}
```


テスト メソッドは、外部システムをアカウント情報で更新する Web サービスを呼び出し、完了時に Accounts integration_Updated__c チェックボックスを True に設定します。
テストは実行に失敗し、「TestMethod として定義されたメソッドは Web サービス コールアウトをサポートしていません。」というエラーが表示されて終了します。

A. Add `if (!Test.isRunningTest())` around `CalloutUtil.sendAccountUpdate`.

B.
Add `Test.startTest()` and `Test.setMock` before and `Test.stopTest()` after `CalloutUtil.sendAccountUpdate`.

C.
Add `Test.startTest()` before and `Test.setMock` and `Test.stopTest()` after `CalloutUtil.sendAccountUpdate`.

D.
Add `Test.startTest()` before and `Test.stopTest()` after `CalloutUtil.sendAccountUpdate`.

正解: **(正解を表示します)**
オプションCはテストメソッドを修正する正しい方法です。オプションCでは、Test.setMock()メソッドを使用して、HttpCalloutMockインターフェースを実装するモッククラスを指定します。モッククラスとは、事前定義されたレスポンスを返すことで外部サービスの動作をシミュレートするクラスです。これにより、開発者はWebサービスへの実際のコールアウトを実行せずに、レスポンスを処理するApexコードのロジックをテストできます。モッククラスは、取引先レコードのintegration_Updated__cチェックボックスをTrueに設定するレスポンスを返す必要があります。
参考 :[モックコールアウトを使用したApexテスト]、[HttpCalloutMockインターフェース]

質問: **113**
開発者は、何百回も呼び出されるクラス内に Debug メソッドを持っています。
メソッドの呼び出し回数をカウントするための開発者コンソールの最適な機能は何ですか?
A. スタック ツリー パネルの [実行ツリー] タブ

B. 実行ログ」パネル
C. 実行スタック」パネル
D. 実行概要パネルの 実行済みユニット」タブ
正解: **D** (コメントを発表する)

質問: **114**
次のコード スニペットを考えてみましょう。

```
<c-selected-order>
<template for:each={orders.data} for:item="order">
    <c-order orderId={order.Id}></c-order>
</template>
</c-selected-order>
```

<e-orders> コンポーネントは、注文がユーザーによって選択されたことを <c-selected-orders コンポーネントにどのように伝達する必要がありますか?
A. アプリケーション イベントを作成して起動します。
B. カスタム イベントを作成して送信します。
C. コンポーネント イベントを作成して起動します。
D. 標準 DOM イベントを作成して起動します。
正解: (正解を表示します)

質問: **115**
開発者は、開発者サンドボックスで Visualforce ページを作成してテストしましたが、本番環境で使用するときにビューステートエラーが発生したというレポートを受け取りました。
開発者はこれらのエラーを修正するために何を確認する必要がありますか?
A. クエリがガバナ制限を正味で超えていないことを確認します。
B. プロパティがプライベートとしてマークされていることを確認します。
C. 変数が一時的としてマークされていることを確認します。
D. プロファイルが Visualforce ページにアクセスできることを確認します。
正解: **C** (コメントを発表する)
開発者は、ビューステートエラーを修正するために、変数が transient としてマークされていることを確認する必要があります。transient キーワードは変数を transient としてマークするために使用されます。つまり、変数はビューステートの一部ではなく、サーバーとクライアント間で永続化または転送されません。開発者は、一時的な保存や計算に使用され、ポストバック間で保持される必要のない変数に transient キーワードを使用する必要があります。これにより、開発者はビューステートのサイズを縮小し、ビューステートの制限超過を回避できます。クエリがガバナ制限を超えないようにすることは、SOQL クエリの制限を回避するだけで、ビューステートのサイズを縮小することはできないため、役に立ちません。プロパティが private としてマークされているこ

とを確認することは、プロパティへのアクセスを制限するだけで、ビューステートのサイズを縮小することはできないため、役に立ちません。プロファイルが Visualforce ページにアクセスできるようにすることは、ページを表示する権限を付与するだけで、ビューステートのサイズを縮小することはできないため、役に立ちません。参照: [transient キーワード]、[Visualforce ビューステート]、[Visualforce 開発者ガイド]

質問: **116**
開発者は、次のように Queueable インターフェースを実装するクラスを作成しました。
ジャワ
共有なしのパブリッククラス OrderQueueableJob は Queueable を実装します {
パブリック void 実行(QueueableContext コンテキスト) {
// 実装ロジック
System.enqueueJob(新しい FollowUpJob());
}
}
デプロイメントプロセスの一環として、開発者は対応するテストクラスを作成する必要があります。テストクラスを正常に実行するために、開発者が実行すべき2つのアクションはどれですか？
1
A. Queueable ジョブが一括モードで実行できるようにするには、seeAllData=true を実装します。2
B. テスト実行中にジョブの連鎖を防ぐには、Te3st.isRunningTest() を実装します。
C. テスト クラスの実行ユーザーが、少なくとも Order オブジェクトに対する 「すべて表示」権限を持っていることを確認します。
D. System.enqueueJob(new OrderQueueableJob()) を Test.startTest と Test.stopTest() で囲みます。
正解: **(**正解を表示します**)**
Queueable インターフェースなどの非同期 Apex をテストするには、テスト実行中にジョブが実際に実行され、プラットフォームの制限に準拠していることを確認するための特別な処理が必要です。
まず、Salesforce は単体テスト中の Queueable ジョブのチェーニングに厳しい制限を設けています。具体的には、テスト内でジョブをチェーニング（別のジョブからキューに追加）することはできません。OrderQueueableJob が FollowUpJob をキューに追加しようとするため、テストはエラーをスローします。この問題を解決するには、開発者は execute メソッド内で Test.isRunningTest() （オプション B）を使用し、テスト実行中に System.enqueueJob の呼び出しを条件付きでバイパスする必要があります。
2つ目に、非同期ジョブはシステムによってキューに登録され、すぐには実行されません。Queueableジョブを強制的に実行して結果をアサートするには、System.enqueueJob呼び出しをTestでラップする必要があります。

startTest() と Test.stopTest() (オプション D)。Test.stopTest() が呼び出されると、そのブロック内のすべてのキュー内の非同期ジョブの実行が完了するまで実行が一時停止されます。
選択肢Aは不正解です。seeAllDataは非同期処理モードに影響を与えません。選択肢Cは不正解です。クラスがwithout sharingとして定義されているため、実行ユーザーの権限に関わらず共有ルールがバイパスされます。再帰/連鎖ガードとstartTest/stopTestブロックを組み合わせることで、開発者はexecuteメソッド内のロジックを確実にテストできます。

質問: **117**
ページが連絡先に対して ﬁnull オブジェクトを逆参照しようとしました」エラーをスローします。
コントローラーのどのような変更を行うとエラーが修正されますか?
A. ゲッターの署名を静的な Contact に変更します。
B. (o] ゲッターの条件を使用して、新しい連絡先が null の場合にそれを返します。
G 連絡先を返すように設定者の署名を変更します。
C. コントローラーの上部で静的な最終コンタクトを宣言します。
正解: **(**正解を表示します**)**

質問: **118**
開発者は、任意の 2 つのオブジェクトレコード間の Salesforce 名項目を比較する汎用 Apex メソッドを作成したいと考えています。たとえば、取引先と商談の名前フィールドを比較します。またはアカウント名と連絡先。
Name フィールドが存在すると仮定すると、開発者はこれをどのように行うべきでしょうか?
A. 各オブジェクトを sObject にキャストし、sObject.get を使用して Name フィールドを比較します。
B. description) 関数を使用して、各 Name フィールドの値を比較します。
C. Salesforce Metadata API を使用して各オブジェクトの値を抽出し、名前フィールドを比較します。
D. 文字列を使用します。Replace () メソッドを使用して、各 Name フィールドの内容を解析し、結果を比較します。
正解: **A (**コメントを発表する**)**
Salesforce の任意の 2 つのオブジェクトレコードの Name 項目を比較する汎用 Apex メソッドを記述する最適な方法は、各オブジェクトを sObject にキャストし、sObject.get を使用して Name 項目を比較することです。sObject は、任意のオブジェクトレコードを表すことができる汎用抽象型で、レコードの項目にアクセスして操作するためのメソッドがあります。開発者は、フィールド名を文字列パラメータとして渡すことで、sObject.get メソッドを使用してフィールドの値を取得できます。その後、== 演算子を使用して、2 つの sObject レコードの Name 項目の値を比較できます。describe 関数を使用して各 Name 項目の値を比較することはできません。describe 関数は、ラベル、データ型、選択リスト値など、オブジェクトまたは項目に関するメタデータ情報を取得するために使用されるものであり、項目の実際の値を取得するものではないためです。
Salesforce メタデータ API を使用して各オブジェクトの値を抽出し、Name 項目を比較すること

はできません。Salesforce メタデータ API は、オブジェクト、項目、レイアウト、ワークフローなどの組織の設定およびカスタマイズメタデータの取得と展開に使用されますが、オブジェクトのデータレコードの取得や展開は行われません。string.replace メソッドを使用して各 Name 項目の内容を解析し、結果を比較することもできません。string.replace メソッドは文字列内の部分文字列を別の部分文字列に置換するために使用されますが、オブジェクトレコードから項目の値を抽出することはできないためです。参考 :[sObject クラス]、[Apex 開発者ガイド]

質問: **119**
Lightning Web コンポーネントのロード時にカスタムロジックを実行できる手法はどれですか?
A. connectedCallback { } メソッドを使用します。
B. <aura:handler> init イベントを使用して関数を呼び出します。
C. randersd loack { } メソッドを使用します。
D. 52.engqueusiction を呼び出し、呼び出すメソッドを渡します。
正解: **(**正解を表示します**)**

質問: **120**
開発者は、取引先の登録に使用される Lightning Web コンポーネントからの入力に基づいて取引先を更新する Apex クラスを作成しました。アカウントの更新は、まだ登録されていない場合にのみ行う必要があります。

```
account = [SELECT Id, Is_Registered__c FROM Account WHERE Id = :accountId];

if (!account.Is_Registered__c) {
    account.Is_Registered = true;
    // ...set other account fields...
    update account;
```


ユーザーが同時に更新を行った場合に、同じアカウントに対する互いの更新を上書きしないようにするには、開発者は何をすべきでしょうか?

A. Add a try/catch block around the update.
B. Use upsert instead of update.
C. Use FOR UPDATE in the SOQL query.
D. Include LastModifiedDate in the query to make sure it wasn't recently updated.

正解: **D (**コメントを発表する**)**
複数のユーザーが同時に同じレコードを更新すると、互いの変更が上書きされてしまうリスクがあります。Salesforce では、これを防ぐために 「レコードロック」と呼ばれるメカニズムを提供しています。
* 選択肢Dは正解です。SOQLクエリにFOR UPDATEを含めると、トランザクションの実行中は取得されたレコードがロックされます。これにより、現在のトランザクションが完了するまで他のトランザクションによるレコードの更新が防止されます。これは、2人のユーザーが互いの更新を上書きしてしまう競合状態を回避するために不可欠です。

* オプション A は不正解です。更新操作の周囲に try/catch ブロックを追加することは例外処理には有効ですが、同時更新による上書きを防ぐことはできません。
* オプションBは不正解です。updateではなくupsertを使用すると、同時更新の問題は解決されません。upsert操作は、一致するIDまたは外部IDを持つレコードが既に存在するかどうかに基づいて、新しいレコードを挿入するか、既存のレコードを更新するために使用されます。
* オプション C は不正解です。レコードをロックするには、SELECT ステートメント自体ではなく、SOQL クエリで FOR UPDATE を使用する必要があるためです。
参考文献:
Salesforce 開発者向けドキュメント 「ステートメントのロック」: ステートメントのロック Apex での同時実行の処理に関する Salesforce 開発者ブログ: Apex での同時実行の処理

質問: **121**
次のコード スニペットを考えてみましょう。

```
public class searchFeature{
  public static List<List<sObject>> searchRecords(string searchquery){
     return [FIND searchquery IN ALL FIELDS RETURNING Account, Opportunity, Lead];
  }
}
```

開発者は、上記のスニペットに適切なコード カバレッジを提供するために、次のテスト クラスを作成しました。

```
@isTest
private class searchFeature_Test{

   @TestSetup
    private static void makeData(){
          //insert opportunities, accounts and lead
    }

    @isTest
    private static searchRecords_Test(){
        List<List<sObject>> records = searchFeature.searchRecords('Test');
        System.assertNotEquals(records.size(),0);
    }
}
```

ただし、テストを実行するとデータが返されず、アサーションは失敗します。
テストクラスが正常に実行されることを確認するには、開発者はどの編集を行う必要がありますか?

A. searchFeatures Apex クラスにキーワードを共有せずに実装します。
B. @1sTest アノテーションに seeAllData=true 属性を実装します。
C. メソッド呼び出しを Test 内に囲みます。startbest i) および @Test_stopTast |)。
D. テスト クラスに setFixedSearchResult= メソッドを実装します。
正解: **D** (コメントを発表する)

Salesforceの検索機能のテストクラスが正常に実行されることを確認するには、開発者はTest.setFixedSearchResults()メソッドを使用する必要があります。これにより、テストで検索によって返されるレコードを指定できるため、検索動作が予測可能になり、アサーションが適切に評価されることが保証されます。参考資料 :Apex開発者ガイド - SOSLクエリのテスト

質問: **122**
組織には、Apex コントローラと、ユーザが選択した選択リスト値の組み合わせで構成されるカスタムフィルタを使用してレコードを表示する Visualforce ページがあります。
一部の入力の組み合わせではページに結果が表示されるまでに時間がかかりすぎますが、他の入力の選択肢では ビュー ステートの最大サイズ制限を超えました」という例外がスローされます。
この問題を解決するには、開発者はどのような手順を実行する必要がありますか?
A. インデックスが作成されていないため、選択リストの値によってフィルタリングするコードを調整します。
B. ビューステート エラーを回避するために、Apex コントローラから transient キーワードのインスタンスを削除します。
C. Apex コントローラで StandardSetController または SOQL LIMIT を使用して、一度に表示されるレコード数を制限します。
D. レイアウトを分割して 1 つの Visualforce ページでレコードをフィルタし、同じ Apex コントローラを使用して 2 番目のページにレコードのリストを表示します。
正解: **C** (コメントを発表する)
StandardSetController または SOQL LIMIT 句を使用すると、Visualforce ページに一度に表示されるレコード数を制限し、ページのパフォーマンスとユーザーエクスペリエンスを向上させることができます。StandardSetController は、開発者がリストコントローラを作成できるクラスです。リストコントローラは、ページ区切り、フィルタリング、並べ替えなどの機能を使用してレコードセットを表示できます。SOQL LIMIT 句は、クエリから返されるレコードの最大数を指定するために使用できるキーワードです。これらのいずれかの方法を使用することで、開発者はページで転送および処理されるデータの量を削減し、ビューステートのサイズが 135 KB の制限を超えた場合に発生するビューステートエラーを回避できます。参照: [StandardSetController クラス]、[SOQL LIMIT 句]、[ビューステート]

質問: **123**

Universal Containers は、未処理の例外が発生した場合に、Apex を使用してカスタムイベントを公開することで外部システムに通知したいと考えています。
この要件を満たす適切なパブリッシュ/サブスクライブ ロジックは何ですか?
A. addError() メソッドを使用してエラー イベントを発行し、イベントをサブスクライブして外部システムに通知するトリガーを作成します。
B. Eventtrus.publish() メソッドを使用してエラー イベントを発行し、CometD を使用して外部システムにイベントをサブスクライブさせます。
C. 外部システムにイベント チャネルをサブスクライブさせます。出版する必要はありません。
D. addError() メソッドを使用してエラー イベントを発行し、CometD を使用して外部システムにイベントをサブスクライブさせます。
正解: **(**正解を表示します**)**

質問: **124**
エンドユーザーが情報にアクセスして更新できるアプリケーション中心の機能を作成する任務を負った開発者 15。この機能は、デスクトップ、電話、タブレットなどの複数のデバイス フォーム ファクタでシームレスに動作しながら、lightning Experience で使用できる必要があります。さらに、機能はアドレス指定可能な URL タブをサポートし、Salesforce コンソール API と対話する必要があります。
アプリケーションを構築し、ビジネス要件をサポートするために、開発者が取ることができるアーム 2 アプローチはどれですか?
2つの答えを選択してください
A. Aura コンポーネントにラップされた Lightning Web コンポーネントを使用してアプリケーションを作成します。
B. Aura コンポーネントを使用してアプリケーションを作成します。
C. Lightning Web コンポーネントにラップされた Aura コンポーネントを使用してアプリケーションを作成します。
D. Lightning Experience Builder を使用してアプリケーションを作成します。
正解: **(**正解を表示します**)**

質問: **125**
開発者は、組織内のすべてのカスタム オブジェクトのドロップダウン リストを表示する Lightning Web コンポーネントを作成します。
Apexメソッドはデータを準備し、コンポーネントに返します。開発者はレスポンスに含めるオブジェクトを決定するために何をすべきでしょうか？
A. @salesforce/schema からすべてのカスタム オブジェクトのリストをインポートします。
B. sObject の記述結果で、getObjectType() の値が 'Custom' か 'Standard' かを確認します。
C. sObject の記述結果の isCustom() 値を確認します。
D. Schema クラスの getCustomObjects() メソッドを使用します。
正解: **(**正解を表示します**)**
包括的かつ詳細な説明：

Apex でカスタムオブジェクトをプログラム的に識別するために、開発者は Schema Describe 情報を利用します。Schema.getGlobalDescribe() メソッドは、組織内のすべてのオブジェクトトークンのマップを返します。これらのトークンを反復処理し、それぞれに対して getDescribe() メソッドを呼び出すことで、開発者はオブジェクトのメタデータのさまざまなプロパティにアクセスできます。
標準オブジェクトとカスタムオブジェクトを区別するための特定のプロパティは、isCustom() メソッド（オプション C）です。このブール型メソッドは、オブジェクトがカスタムオブジェクト（カスタム設定とカスタムメタデータ型を含む）の場合は true を返し、標準プラットフォームオブジェクトの場合は false を返します。
選択肢Aは不正解です。LWCでは@salesforce/schemaは特定の既知のオブジェクト/項目への参照をインポートするために使用され、組織全体のメタデータを動的に検出するためには使用されません。選択肢Bは不正解です。getObjectType()は「Custom」のような文字列値ではなく、SObjectTypeトークンを返すためです。選択肢Dは不正解です。標準スキーマクラスには getCustomObjects()メソッドがないため、検出はグローバル記述マップを介して行う必要があります。

質問: **126**
開発者はどうすれば複数の JavaScript ライブラリを Aura コンポーネントに効率的に組み込むことができますか?
A. JavaScript リモーティングとスクリプト タグを使用します。
B. スクリプト属性を持つ CDN を使用します。
C. ライブラリを個別のヘルパー ファイルに実装します。
D. 静的リソースから複数のアセットを結合します。
正解: **D (**コメントを発表する**)**
Auraコンポーネントに複数のJavaScriptライブラリを組み込む場合、複数のアセットを単一の静的リソースに結合するのがベストプラクティスです。このアプローチは効率的で、非同期読み込みやライブラリ間の潜在的な競合の問題を回避できます。参考資料 :Auraコンポーネント開発者ガイド - 外部JavaScriptライブラリの使用

質問: **127**
次のコード スニペットを考えてみましょう。

```
public static List<Account> getAccounts(Date thisDate, Id goldenRT){
List<Account> accountList = [Select Id, Name, Industry FROM Account WHERE CreatedDate = :thisDate OR RecordTypeId = :goldenRT];
return accountList;
}
```

Apex メソッドはアカウントのデータ量が多い環境で実行されており、クエリのパフォーマンスが低下しています。
結果セット全体を保持しながらクエリを最適に実行するには、開発者はどの手法を実装する必要がありますか?
A. createdDate と RecordType の値を組み合わせる数式フィールドを作成し、数式に基づいてフィルターします。

B. クエリを 2 つの個別のクエリに分割し、2 つの結果セットを結合します。
C. メソッドに @Future アノテーションを付けます
D. データベースの queryLocator メソッドを使用してアカウントを取得します。
正解: **B** (コメントを発表する)
クエリのパフォーマンスを最適化しつつ結果セット全体を保持するために開発者が実装すべき手法は、クエリを2つの個別のクエリに分割し、それらを結合することです。これは、コードスニペットのクエリが、デフォルトではインデックス付けされていないCreatedDateフィールドとRecordTypeフィールドを組み合わせた複雑なフィルター条件を使用しているためです。つまり、クエリはAccountオブジェクトのテーブル全体をスキャンすることになり、データ量が多い場合は非常に遅く、非効率的になる可能性があります。これを回避するには、開発者はクエリを2つのクエリに分割し、1つはCreatedDateフィールドでフィルターし、もう1つはRecordTypeフィールドでフィルターします。そして、両方の条件に一致するAccountのIDをSetまたはMapに格納し、そのSetまたはMapを使用してデータベースからAccountレコードを取得します。こうすることで、クエリはId、CreatedDate、およびRecordTypeフィールドに標準のインデックスを使用し、より高速かつ効率的に実行されます。

質問: **128**
ある企業には、リクエストとリクエスト行を更新し、更新されたレコードを使用して外部 ERP システムの REST エンドポイントにコールアウトを行うコードがあります。

```
public void updateAndMakeCallout(Map<Id, Request__c> reqs,
Map<Id, Request_Line__c> reqLines) {
    Savepoint sp = Database.setSavepoint();

    try {
        insert reqs.values();
        insert reqLines.values();
        HttpResponse response =
CalloutUtil.makeRestCallout(reqs.keySet(), reqLines.keySet());
    } catch (Exception e) {
        Database.rollback(sp);
    }
}
```


callousUtil.makeRestCallout は、保留中のコミットされていない作業があります。cut を呼び出す前にコミットまたはロールバックしてください」エラーで失敗します。
問題に対処するには何をすべきでしょうか?

A. callousUtil makeRestCallout を @InvocsblsMethod メソッドに変更します。
B. データベースを削除します。setSavepoint とデータベース。ロールバック。
C. CallousUtill .makeRestCallout を @future メソッドに変更します。
正解: **(**正解を表示します**)**
このエラーは、コールアウトの前にDML操作が実行されたために発生します。この問題を解決するには、@future(callout=true)アノテーションを付与した非同期メソッドなどからコールアウトを実行する必要があります。これにより、DML操作をコールアウトとは別にコミットできます。参考資料 :Apex開発者ガイド - 非同期コールアウトの作成

質問: **129**
ページのロード時に false に設定されてレンダリングされた VisualForce コンポーネントを含む VisualForce ページで、開発者は再レンダリング時にそれが表示されることをどのように確認できますか?
A. コンポーネントの render 属性を true に設定し、親コンポーネントを再レンダリングします。
B. コンポーネントの re-render 属性を true に設定します。
C. レンダリングされた要素はリフレッシュしないと再レンダリングできないため、ページ全体のリフレッシュを実行します。
D. コンポーネントの render 属性を true に設定し、コンポーネントを再レンダリングします。
正解: **A (**コメントを発表する**)**

質問: **130**
以下のコード スニペットを参照してください。

```
public static void updateCreditMemo(String customerId, Decimal newAmount){
  List<Credit_Memo__c> toUpdate = new List<Credit_Memo__c>();
  for(Credit_Memo__c creditMemo : [Select Id, Name, Amount__c FROM Credit_Memo__c WHERE Customer_Id__c = :customerId LIMIT
50]){
      creditMemo.Amount__c = newAmount;
      toUpdate.add(creditMemo);
  }
  Database.update(toUpdate,false);
}
```


Credit_Memo__c というカスタム オブジェクトが Salesforce 環境に存在します。このタイプのレコードを取得して操作する新機能開発の一環として、開発者は、Apex トランザクション内でレコードのセットが変更されるときに競合状態が確実に防止されるようにする必要があります。
前述の Apex コードで、開発者はクエリステートメントを変更して SOQL 機能を使用し、トランザクション内の競合状態を防ぐにはどうすればよいでしょうか?
A.

```
[Select Id, Name, Amount__c FROM Credit_Memo__c WHERE Customer_Id__c = :customerId LIMIT 50 FOR UPDATE]
```

B.

```
[Select Id, Name, Amount__c FROM Credit_Memo__c WHERE Customer_Id__c = :customerId LIMIT 50 FOR VIEW]
```

C.

```
[Select Id, Name, Amount__c FROM Credit_Memo__c WHERE Customer_Id__c = :customerId USING SCOPE LIMIT 50]
```

D.

```
[Select Id, Name, Amount__c FROM Credit_Memo__c WHERE Customer_Id__c = :customerId LIMIT 50 FOR REFERENCE]
```

正解: **A (**コメントを発表する**)**

Apexトランザクション中の競合状態を防ぐには、SOQLクエリに「FOR UPDATE」キーワードを追加して変更します。このキーワードは、クエリによって返されるレコードをロックし、現在のトランザクションが完了するまで他のトランザクションによる変更を防止します。これにより、現在のトランザクションが完了するまでは、取得したレコードが他のプロセスによって変更されることがなくなり、競合状態を回避できます。提供されているコードスニペットで使用する正しいオプションは次のとおりです。

[SELECT Id, Name, Amount__c FROM Credit_Memo__c WHERE Customer_Id__c = :customerId LIMIT

50 アップデート用]

これにより、トランザクションの期間中、指定された Customer_Id__c を持つ Credit_Memo__c のレコードが最大 50 件ロックされます。

参考文献:

アップデート用

レコード更新の競合を防ぐ

質問: **131**

開発者は、取引先責任者を検索する Lightning Web コンポーネントを構築しています。コンポーネントは、検索が完了したときに、別の DOM ツリー内にある他の無関係な Lightning Web コンポーネントに検索結果を伝達する必要があります。

通信を実装するには開発者は何をすべきでしょうか?

A. メッセージ チャネルにメッセージを公開します。

B. イベント チャネルでイベントを公開します。

C. アプリケーションイベントを発生させます。

D. カスタム コンポーネント イベントを起動します。

正解: **(**正解を表示します**)**

質問: **132**

以下のコンポーネント コードと要件を参照してください。

要件：

1. モバイル デバイスの場合、情報は 3 行に表示されます。
2. デスクトップとタブレットの場合、情報は 1 行で表示されます。
要件 2 希望どおりに表示されない。
デスクトップとタブレットの要件を満たす正しいコンポーネント コードを持つオプションはどれですか?

A.

B.

C.

D.

正解: **(**正解を表示します**)**
モバイル デバイスとデスクトップおよびタブレットで異なる方法で情報を表示するための指定された要件を満たすには、開発者は適切なサイズ属性を持つ Lightning レイアウトおよび Lightning レイアウト項目コンポーネントを使用して、フォーム ファクターに基づいてレイアウトを制御する必要があります。

* オプションAは正解です。デスクトップとタブレットデバイスではsize属性を「2」に設定しているため、フレームワークはコンテナの幅全体を使用するよう指示され、結果として1行になります。モバイルデバイスではデフォルトで全幅に設定され、レイアウトアイテムが垂直に積み重ねられ、結果として3行になります。
* オプション B、C、および D は、デスクトップとタブレットの正しいサイズ属性が設定されていないか、これらのデバイスで 1 行になる適切な構造が維持されていないため、正しくありません。
参考文献:
グリッドシステム上の Lightning デザインシステム: Lightning グリッドシステム
Lightning Web コンポーネントのレイアウトコンポーネントに関するドキュメント: レイアウトコンポーネント

質問: **133**
開発者が作成したテストクラスでは、モックコールアウトを行う前にテストデータを作成していましたが、「コミットされていない作業が保留中です。コールアウトを行う前にコミットまたはロールバックしてください」というエラーが発生します。このエラーを解決するには、どのような手順を踏む必要がありますか？
A. 挿入とモック呼び出しの両方が Test.stopTest() の後に行われることを確認します。1
B. レコードが Test.startTest() ステートメントの前に挿入され、モックコールアウトが @testSetup.2 でアノテーションされたメソッド内で発生することを確認します。
C. レコードが Test.startTest() ステートメントの前に挿入され、モックコールアウトが Test.startTest() の後に発生することを確認します。45
D. 挿入とモックコールアウトの両方が Test.startTest() の後に行われることを確認します。67
正解: **(正解を表示します)**
包括的かつ詳細な150～250語の説明:
このエラーは、SalesforceがDML操作の実行後に同じトランザクション内でのコールアウト（模擬コールアウトも含む）の実行を禁止しているために発生します。テストデータを挿入すると、データベースに「保留中」のトランザクションが作成されます。その後すぐにコールアウトを実行しようとすると、プラットフォームはデータの不整合を防ぐため、CalloutExceptionをスローします。テスト コンテキストでこれを解決するには、Test を使用して DML 操作をコールアウトから分離する必要があります。
startTest() メソッドと Test.stopTest() メソッド。Test.startTest() が呼び出されると、Salesforce は新しいガバナ制限セットを提供し、後続のコードに対して新しいトランザクションコンテキストを効果的に作成します。Test.startTest() の前にレコードを挿入し、Test.startTest() の後にコールアウトをトリガするロジックを実行することで、開発者はコールアウトが開始される前に DML 操作がテストデータベースコンテキストに「コミット」されることを保証できます。
選択肢Cはこのパターンを正しく特定しています。選択肢Bは不正解です。@testSetupはすべてのテストメソッドにわたるグローバルデータ作成に使用されており、コールアウトトランザクションの分割を具体的には考慮していません。選択肢AとDは、DMLとコールアウトを同じコンテキス

トに保持するか、後続のコールアウトリクエストに干渉するコンテキストにDMLを配置するため、この問題を解決できません。

質問: **134**
Visualforce ページの GET 要求の実行順序で、このステップの後に何が起こりますか? コントローラーと拡張機能のコンストラクターを評価する
A. HTML レスポンスをブラウザに送信する
B. <apex: form> が存在する場合はビューステートを作成します
C. 式、アクション属性、およびメソッド呼び出しを評価します
D. カスタム コンポーネントのコンストラクタと式を評価する
正解: **(**正解を表示します**)**

質問: **135**
コールアウト数のトランザクション制限は何ですか?
A. 100
B. 制限なし
C. 200
D. 150
E. 50
正解: **(**正解を表示します**)**

質問: **136**
RESTful Web サービスを呼び出す場合、開発者は双方向 SSL 認証を実装してセキュリティを強化する必要があります。Salesforce 管理者は、 ERPSecCertificate」という一意の名前を持つ自己署名証明書を Salesforce 内で生成しました。
次のコード スニペットを検討してください。
証明書を使用して HTTP 要求に署名するために、開発者が実装する必要がある方法はどれですか?
A. req.setSecureCertificate('ERPSecCertificate');
B. req.setHeader('証明書', 'ERPSecCertificate');
C. req.setClientCertificateName('ERPSecCertificate');
D. req.setSecure('ERPSecCertificare)';
正解: **(**正解を表示します**)**

質問: **137**
開発者は、オポチュニティが処理されるたびに、オポチュニティのアカウントのカスタムの最終販売日フィールドを更新するトリガーをオポチュニティに作成しました。トリガーのテスト クラスで、Last Sold Date フィールドを検証するアサーションが失敗します。
失敗したアサーションの原因は何ですか?
A. テスト クラスは System.runAs() を使用して、Salesforce 管理者としてテストを実行していません。
B. テスト クラスは、テスト データを挿入するときにアカウント所有者を定義していません。
C. テスト クラスは、商談の更新後にアカウント レコードを再クエリしていません。
D. テスト クラスは、テスト メソッドの seeAllData-true を実装していません。
正解: (正解を表示します)

質問: **138**
開発者は、Salesforce からデータを取得してレコードプロパティに割り当てる Lightning Web コンポーネントを構築しています。

```
import { LightningElement, api, wire } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';

export default class Record extends LightningElement {
    @api fields;
    @api recordId;
    record;
}
```

Salesforce からデータを取得するにはコンポーネントで何を行う必要がありますか?

A.
```
Add the following code above record;
@wire(getRecord, { recordId: '$recordId', fields: '$fields' })
```

B.
```
Add the following code above record;
@api(getRecord, { recordId: '$recordId' })
```

C.
```
Add the following code above record;
```

正解: (正解を表示します)
選択肢Aが正解です。@wireデコレータはlightningのgetRecordと組み合わせて使用されます。Salesforceからレコードを取得するには、/uiRecordApi を使用します。構文 @wire(getRecord, { recordId: '$recordId', fields: '$fields' }) はリアクティブプロパティを設定します。つまり、recordIdまたはfieldsプロパティが変更されるたびに自動的に再実行されます。
参考文献:
レコードデータを取得

質問: **139**
次のコード セグメントは、Opportunity トリガーのトリガー ハンドラー クラスから生成されます。

このコードを改善し、より効率的にする必要がある 2 つの変更はどれですか? 2つの答えを選択してください
A. SOQL を移動して、for ループの外で取引先レコードをフェッチします。
B. ビジネス ロジックを Opportunity トリガー内に移動します。
C. DML を for ループの外に移動します。
D. Trigger.new の代わりに Triginstead.old を使用します。
正解: **(**正解を表示します**)**

質問: **140**
ライトニング オーラ コンポーネント

```
<aura:component controller="SimpleServerSideController">
    <aura:attribute name="firstName" type="String" default="world"/>
    <lightning:button label="Call server" onclick="{!c.echo}"/>
</aura:component>
```

Lightning Aura Controller

```
01 ({
02     "echo" : function(cmp) {
03         // create a one-time use instance of the serverEcho action
04         // in the server-side controller
05         var action = cmp.get("c.serverEcho");
06         action.setParams({ firstName : cmp.get("v.firstName") });
07
08         // Create a callback that is executed after
09         // the server-side action returns
10         action.setCallback(this, function(response) {
11             var state = response.getState();
12             if (state === "SUCCESS") {
13                 // Alert the user with the value returned
14                 // from the server
15                 alert("From server: " + response.getReturnValue());
16
17                 // You would typically fire an event here to trigger
18                 // client-side notification that the server-side
19                 // action is complete
20             }
21             else if (state === "INCOMPLETE") {
22                 // do something
23             }
24             else if (state === "ERROR") {
37         // A client-side action could cause multiple events,
38         // which could trigger other events and
39         // other server-side action calls.
40         // $A.enqueueAction adds the server-side action to the queue.
41         $A.enqueueAction(action);
42     }
43 })
```

上記のコードを考えると、コードが機能するために Apex コントローラで行う必要がある 2 つの変更はどれですか? 2 つの回答を選択してください

A. 単一のメソッドだけでなく、クラス全体に @AuraEnabled としてアノテーションを付けます。

B. 引数を JSONObject から String に変更します。

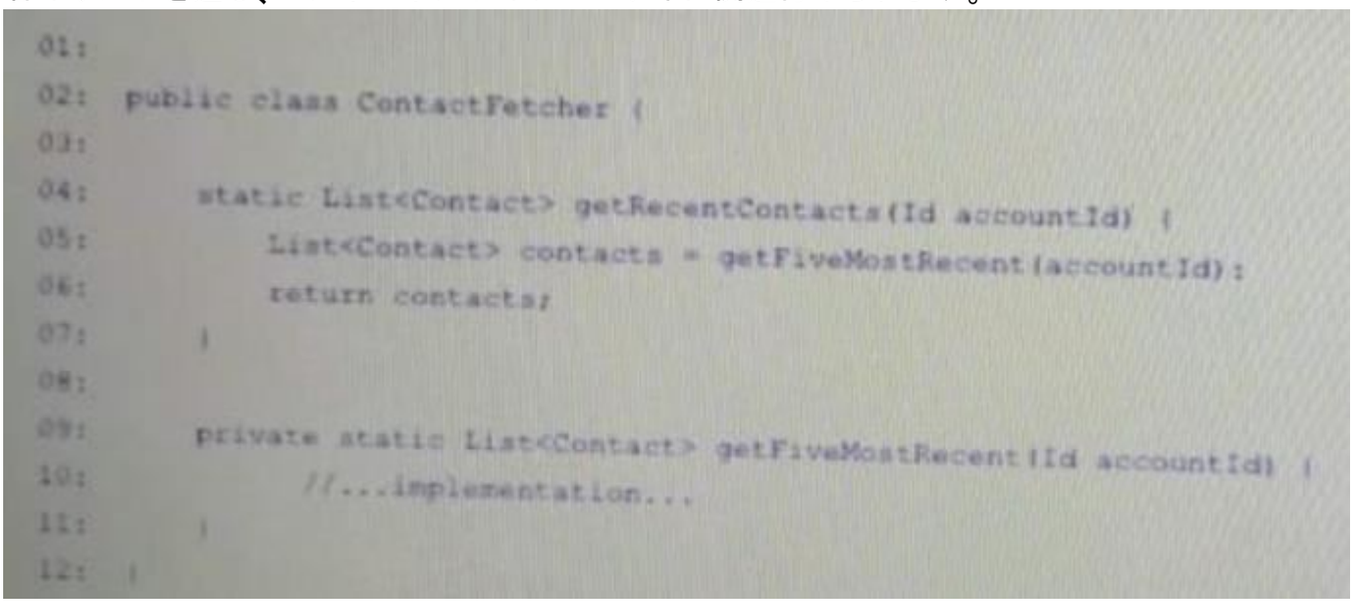
C. Apex コントローラから 06 行目を削除し、リターンで firstName を使用します。

D. メソッド シグネチャを public static ではなく global static に変更します。

正解: **B,C** (コメントを発表する)

質問: **141**

開発者は、取引先の最近連絡した 5 つの取引先責任者を表示する取引先レコード ページの Lightning Web コンポーネントを作成しました。Apex メソッド getRecentContacts は取引先責任者のリストを返し、コンポーネントのプロパティに関連付けられます。



```
01:
02: public class ContactFetcher {
03:
04:     static List<Contact> getRecentContacts(Id accountId) {
05:         List<Contact> contacts = getFiveMostRecent(accountId);
06:         return contacts;
07:     }
08:
09:     private static List<Contact> getFiveMostRecent(Id accountId) {
10:         //...implementation...
11:     }
12: }
```

Apex メソッドを配線できるようにするには、上記の頌歌でどの 2 行を変更する必要がありますか?
2つの答えを選択してください
A. 08 行目に AuraEnanled(cacheabletrue) を追加します。
B. @AuraEnabled (cacheabletrue) を 03 行目に追加します。
C. 行 04 に public を追加します。
D. ライン 09 からプライベートを削除します。
正解: **(**正解を表示します**)**

質問: **142**
Apex トリガーと Apex クラスは、ケースが変更されるたびにカウンター `Edit_Count__c` を増分します。
```java
パブリッククラス CaseTriggerHandler {
パブリック静的voidハンドル(List<Case>ケース) {
(ケースc:ケース)の場合{
c.Edit_Count__c = c.Edit_Count__c + 1;
}
}
}
ケースのトリガー(更新前) {
CaseTriggerHandler.handle(Trigger.new);
}
```
「」

ケースオブジェクトに、ケースの作成または更新時に実行される保存前レコードトリガフローを本番環境に新しく追加しました。このプロセスを追加して以来、ケースの編集時に「Edit_Count__c」が複数回増加しているという報告を受けています。この問題を修正するApexコードはどれでしょうか？

A. ```java
パブリッククラス CaseTriggerHandler {
パブリック静的ブール値 firstRun = true;
パブリック静的voidハンドル(List<Case>ケース) {
(ケースc:ケース)の場合{
B. Edit_Count__c = c.Edit_Count__c + 1;
}
}
}
ケースのトリガー(更新前) {
CaseTriggerHandler.firstRun = true;
if (CaseTriggerHandler.firstRun) {
CaseTriggerHandler.handle(Trigger.newMap);
}
CaseTriggerHandler.firstRun = false;
}
「」

C. ```java
パブリッククラス CaseTriggerHandler {
パブリック静的ブール値 firstRun = true;
パブリック静的voidハンドル(List<Case>ケース) {
(ケースc:ケース)の場合{
D. Edit_Count__c = c.Edit_Count__c + 1;
}
}
}
ケースのトリガー(更新前) {
if (CaseTriggerHandler.firstRun) {
CaseTriggerHandler.handle(Trigger.new);
}
CaseTriggerHandler.firstRun = false;
}
「」

E. ```java
パブリッククラス CaseTriggerHandler {
ブール値 firstRun = true;

パブリック静的voidハンドル(List<Case>ケース) {
if (firstRun) {
(ケースc:ケース)の場合{
F. Edit_Count__c = c.Edit_Count__c + 1;
}
}
firstRun = false;
}
}
ケースのトリガー(更新前) {
CaseTriggerHandler.handle(Trigger.new);
}
「」
G. ```java
ケースのトリガー(更新前) {
ブール値 firstRun = true;
if (firstRun) {
CaseTriggerHandler.handle(Trigger.newMap);
}
firstRun = false;
}
「」
正解: **(**正解を表示します**)**
このシナリオは、Salesforce の実行順序によって発生する典型的なトリガ再帰の問題を示しています。レコードが更新されると、Salesforce は特定のシーケンスを実行します。まず 「保存前」のレコードトリガフローが実行され、次に 「保存前」トリガが実行され、 「保存後」トリガが実行され、最後に 「保存後」フローとプロセスが実行されます。プロセスまたはフローがトランザクションを開始したレコードと同じレコードを更新すると、その単一のトランザクション内で Apex トリガのサイクル全体が再度呼び出される可能性があります。
これらの再入サイクル中にロジックが複数回実行されるのを防ぐため、開発者は 「再帰ガード」として静的ブール型変数を実装します。Apexの静的変数は、単一のトランザクションの実行中ずっと保持されます。トリガの開始時に静的ブール型変数の値をチェックすることで、コードは現在のレコードセットを既に処理済みかどうかを判断できます。
オプションBはこのパターンの正しい実装です。ハンドラークラスで 「public static Boolean firstRun = true;」を定義します。トリガーは 「firstRun」がtrueかどうかを確認し、ハンドラーロジックを実行し、すぐに設定を行います。
`firstRun` を false に設定します。同じトランザクション内での Case トリガーの以降の実行では、変数が false に設定され、増分ロジックがスキップされます。

オプション A は、トリガーが開始するたびに `firstRun` を true にリセットし、ガードが無効になるため、正しくありません。
オプションCは、呼び出しごとに再作成されるインスタンス変数（非静的を誤って使用しています。オプションDは、トリガー本体内のローカル変数を使用していますが、トリガーが実行されるたびにtrueに再初期化されるため、再帰に対する保護が提供されません。

質問: **143**
Universal Containers は、経費精算のための新しい承認プロセスを導入しています。このプロセスでは、経費金額、従業員の役割、プロジェクトの種類などの要素に基づいて適切な承認者を決定するための複雑なロジックが必要です。このソリューションでは、承認ルールの柔軟性と将来の変更が可能になる必要があります。
このロジックを実装するにはどのアプローチが最も適していますか?
A. 指定された基準に基づいて適切な承認者を決定するメソッドを含むカスタム Apex クラスを作成します。
B. 承認ロジックを処理するカスタム Lightning コンポーネントを開発し、経費精算レコード ページに統合します。
C. カスタム数式フィールドを実装して、指定された基準に基づいて適切な承認者を計算および決定します。
D. Salesforce の承認プロセス機能を使用し、入力基準と承認の割り当てを含む複数の承認ステップを定義します。
正解: **(**正解を表示します**)**
複雑なロジックと柔軟性が求められる場合、適切な承認者を決定するメソッドを備えたカスタム Apexクラスを作成するのが最適なアプローチです。このアプローチにより、ビジネス要件の変化に応じて容易に保守・更新できる一元化されたロジックを構築できます。
参考資料: Apex 開発者ガイド - Apex クラス

質問: **144**
Apex CPU 制限に関して推奨される方法は何ですか? （2つ選んでください。）
A. Map コレクションを使用して sObject をキャッシュする
B. Visualforce ページのビュー ステートを減らす
C. ネストされた Apex イテレーションを避ける
D. SOQL クエリのパフォーマンスを最適化する
正解: **(**正解を表示します**)**

版のアクセス、https://www.jpntest.com/shiken/PDII-JPN-mondaishu　**163**問、３０%ディスカウント、特別な割引コード: **JPNshiken**」