AccessLevel Methods

Apex Reference Guide / System Namespace / AccessLevel Class

AccessLevel Class

Defines the different modes, such as system or user mode, that Apex database operations execute in.

Namespace

System

Usage

By default, Apex code runs in system mode, which means that it runs with substantially elevated permissions over the user running the code. In system mode, the object and field-level permissions of the current user are ignored, and the record sharing rules are controlled by the class sharing keywords. In user mode, the current user's object permissions, field-level security, and sharing rules are enforced.

Many of the DML methods of the `System.Database` and `System.Search` classes include an `accessLevel` parameter to specify the execution mode.

Avoid specifying an `accessLevel` parameter in the same query as a `WITH SECURITY_ENFORCED` clause. Salesforce recommends that you specify either system mode or user mode, and remove any redundant `WITH SECURITY_ENFORCED` clauses.

Example

If the user running this Apex code doesn't have write access to the Account object, the `Database.insert()` method returns an error.

```
List<Account> toInsert = new List<Account>{new Account(Name = 'Exciting New Account')};

List<Database.SaveResult> sr = Database.insert(toInsert, AccessLevel.USER_MODE);
```

In contrast, this example shows the method running in system mode. The success of the insert doesn't depend on whether the user running the Apex code has create access to the Account object.

```
List<Account> toInsert = new List<Account>{new Account(Name = 'Exciting New Account')};

List<Database.SaveResult> sr = Database.insert(toInsert, AccessLevel.SYSTEM_MODE);
```

- AccessLevel Methods
- AccessLevel Properties

AccessLevel Methods

The following are methods for `AccessLevel`.

withPermissionSetId(permissionSetId)(Developer Preview)

Supports database and search operations to be run with permissions specified in a permission set. Apex enforces field-level security (FLS) and object permissions as per the specified permission set, in addition to the running user’s permissions.

> **Note**
>
> Feature is available as a developer preview. Feature isn’t generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. All commands, parameters, and other features are subject to change or deprecation at any time, with or without notice. Don’t implement functionality developed with these commands or tools in a production environment. You can provide feedback and suggestions for the “Permission Sets with User Mode” feature in the Trailblazer Community.

This feature is available in scratch orgs where the `ApexUserModeWithPermset` feature is enabled. If the feature isn’t enabled, Apex code with this feature can be compiled but not executed.

Signature

```
public System.AccessLevel withPermissionSetId(String permissionSetId)
```

Parameters

permissionSetId

Type: String

Permissions in the specified permission set are enforced while running user-mode DML operations, in addition to the running user’s permissions.

Return Value

Type: Access Level Class

Example

This example runs the `AccessLevel.withPermissionSetId()` method with the specified permission set and inserts a custom object.

```
@isTest
public with sharing class ElevateUserModeOperations_Test {
    @isTest
    static void objectCreatePermViaPermissionSet() {
        Profile p = [SELECT Id FROM Profile WHERE Name='Minimum Access - Salesforce'];
        User u = new User(Alias = 'standt', Email='standarduser@testorg.com',
            EmailEncodingKey='UTF-8', LastName='Testing', LanguageLocaleKey='en_US',
            LocaleSidKey='en_US', ProfileId = p.Id,
            TimeZoneSidKey='America/Los_Angeles',
            UserName='standarduser' + DateTime.now().getTime() + '@testorg.com');

        System.runAs(u) {
            try {
                Database.insert(new Account(name='foo'), AccessLevel.User_mode);
                Assert.fail();
            } catch (SecurityException ex) {
                Assert.isTrue(ex.getMessage().contains('Account'));
            }
            //Get ID of previously created permission set named 'AllowCreateToAccount'
            Id permissionSetId = [Select Id from PermissionSet
                where Name = 'AllowCreateToAccount' limit 1].Id;
                Database.insert(new Account(Name='Test'), AccessLevel.User_mode);
                Assert.fail();
            } catch (SecurityException ex) {
                Assert.isTrue(ex.getMessage().contains('Account'));
            }

        }
    }
}
```

AccessLevel Properties

The following are properties for `AccessLevel`.

- SYSTEM_MODE
  Execution mode in which the the object and field-level permissions of the current user are ignored, and the record sharing rules are controlled by the class sharing keywords.
- USER_MODE
  Execution mode in which the object permissions, field-level security, and sharing rules of the current user are enforced.

SYSTEM_MODE

Execution mode in which the the object and field-level permissions of the current user are ignored, and the record sharing rules are controlled by the class sharing keywords.

Signature

```
public System.AccessLevel SYSTEM_MODE {get;}
```

Property Value

Type: System.AccessLevel

USER_MODE

Execution mode in which the object permissions, field-level security, and sharing rules of the current user are enforced.

Signature

```
public System.AccessLevel USER_MODE {get;}
```

Property Value

Type: System.AccessLevel

DID THIS ARTICLE SOLVE YOUR ISSUE?
Let us know so we can improve!

Share your feedback

Tableau
Commerce Cloud
Lightning Design System
Einstein
Quip

APIs
Trailhead
Sample Apps
Podcasts
AppExchange

Partner Community
Blog
Salesforce Admins
Salesforce Architects

Privacy Information Terms of Service Legal Use of Cookies Trust Cookie Preferences

✓✕ Your Privacy Choices Responsible Disclosure Contact