



Callable Interface

Enables developers to use a common interface to build loosely coupled integrations between Apex classes or triggers, even for code in separate packages. Agreeing upon a common interface enables developers from different companies or different departments to build upon one another's solutions. Implement this interface to enable the broader community, which might have different solutions than the ones you had in mind, to extend your code's functionality.

Note

This interface is not an analog of the Java Callable interface, which is used for asynchronous invocation. Don't confuse the two.

Namespace

[System](#)

Usage

To implement the `Callable` interface, you need to write only one method: `call(String action, Map<String, Object> args)`.

In code that utilizes or tests an implementation of `Callable`, cast an instance of your type to `Callable`.

This interface is not intended to replace defining more specific interfaces. Rather, the `callable` interface allows integrations in which code from different classes or packages can use common base types.

- [Callable Methods](#)
- [Callable Example Implementation](#)

Callable Methods

The following are methods for `Callable`.

- [call\(action, args\)](#)
Provides functionality that other classes or packages can utilize and build upon.

call(action, args)

Provides functionality that other classes or packages can utilize and build upon.

Signature

```
public Object call(String action, Map<String,Object> args)
```

Parameters

action

Type: [String](#)



Arguments to be used by the specified action.

Return Value

Type: Object

The result of the method invocation.

Callable Example Implementation

This class is an example implementation of the `System.Callable` interface.

```
public class Extension implements Callable {

    // Actual method
    String concatStrings(String stringValue) {
        return stringValue + stringValue;
    }

    // Actual method
    Decimal multiplyNumbers(Decimal decimalValue) {
        return decimalValue * decimalValue;
    }

    // Dispatch actual methods
    public Object call(String action, Map<String, Object> args) {
        switch on action {
            when 'concatStrings' {
                return this.concatStrings((String)args.get('stringValue'));
            }
            when 'multiplyNumbers' {
                return this.multiplyNumbers((Decimal)args.get('decimalValue'));
            }
            when else {
                throw new ExtensionMalformedCallException('Method not implemented');
            }
        }
    }

    public class ExtensionMalformedCallException extends Exception {}
}
```

The following test code illustrates how calling code utilizes the interface to call a method.

```
@IsTest
private with sharing class ExtensionCaller {

    @IsTest
    private static void givenConfiguredExtensionWhenCalledThenValidResult() {

        // Given
        String extensionClass = 'Extension'; // Typically set via configuration
        Decimal decimalTestValue = 10;

        // When
        Callable extension =
            (Callable) Type.forName(extensionClass).newInstance();
        Decimal result = (Decimal)
            extension.call('multiplyNumbers', new Map<String, Object> {
                'decimalValue' => decimalTestValue
            });

        // Then
        System.assertEquals(100, result);
    }
}
```



- [Apex Developer Guide: Classes and Casting](#)

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

[Share your feedback](#)



DEVELOPER CENTERS

- [Heroku](#)
- [MuleSoft](#)
- [Tableau](#)
- [Commerce Cloud](#)
- [Lightning Design System](#)
- [Einstein](#)
- [Quip](#)

POPULAR RESOURCES

- [Documentation](#)
- [Component Library](#)
- [APIs](#)
- [Trailhead](#)
- [Sample Apps](#)
- [Podcasts](#)
- [AppExchange](#)

COMMUNITY

- [Trailblazer Community](#)
- [Events and Calendar](#)
- [Partner Community](#)
- [Blog](#)
- [Salesforce Admins](#)
- [Salesforce Architects](#)

© Copyright 2025 Salesforce, Inc. [All rights reserved.](#) Various trademarks held by their respective owners. Salesforce, Inc.
Salesforce Tower, 415 Mission Street, 3rd Floor, San Francisco, CA 94105, United States

[Privacy Information](#) [Terms of Service](#) [Legal](#) [Use of Cookies](#) [Trust](#) [Cookie Preferences](#)

☒ [Your Privacy Choices](#) [Responsible Disclosure](#) [Contact](#)