Assert Class

Contains methods to assert various conditions with test methods, such as whether two values are the same, a condition is true, or a variable is null.

Namespace

System

Assert Methods

The following are methods for `Assert`.

- **areEqual(expected, actual, msg)**
  Asserts that the first two arguments are the same.
- **areEqual(expected, actual)**
  Asserts that the two arguments are the same.
- **areNotEqual(notExpected, actual, msg)**
  Asserts that the first two arguments aren't the same.
- **areNotEqual(notExpected, actual)**
  Asserts that the two arguments aren't the same.
- **fail(msg)**
  Immediately return a fatal error that causes code execution to halt.
- **fail()**
  Immediately return a fatal error that causes code execution to halt.
- **isFalse(condition, msg)**
  Asserts that the specified condition is `false`.
- **isFalse(condition)**
  Asserts that the specified condition is `false`.
- **isInstanceOfType(instance, expectedType, msg)**
  Asserts that the instance is of the specified type.
- **isInstanceOfType(instance, expectedType)**
  Asserts that the instance is of the specified type.
- **isNotInstanceOfType(instance, notExpectedType, msg)**
  Asserts that the instance isn't of the specified type.
- **isNotInstanceOfType(instance, notExpectedType)**
  Asserts that the instance isn't of the specified type.
- **isNotNull(value, msg)**
  Asserts that the value isn't null.
- **isNotNull(value)**
  Asserts that the value isn't null.
- **isNull(value, msg)**
  Asserts that the value is null.
- **isNull(value)**
  Asserts that the value is null.
- **isTrue(condition, msg)**
  Asserts that the specified condition is `true`.

Asserts that the first two arguments are the same.

Signature

```
public static void areEqual(Object expected, Object actual, String msg)
```

Parameters

expected

Type: Object

Expected value.

actual

Type: Object

Actual value.

msg

Type: String

(Optional) Custom message returned as part of the error message.

Return Value

Type: void

Usage

If the first two arguments aren't the same, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String sub = 'abcde'.substring(2);
Assert.areEqual('cde', sub, 'Expected characters after first two'); // Succeeds
```

areEqual(expected, actual)

Asserts that the two arguments are the same.

Signature

```
public static void areEqual(Object expected, Object actual)
```

Parameters

expected

Type: Object

Expected value.

actual

Type: Object

Actual value.

Return Value

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String sub = 'abcde'.substring(2);
Assert.areEqual('cde', sub); // Succeeds
```

areNotEqual(notExpected, actual, msg)

Asserts that the first two arguments aren't the same.

Signature

```
public static void areNotEqual(Object notExpected, Object actual, String msg)
```

Parameters

notExpected

Type: Object

Value that's not expected.

actual

Type: Object

Actual value.

msg

Type: String

(Optional) Custom message returned as part of the error message.

Return Value

Type: void

Usage

If the first two arguments are the same, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String sub = 'abcde'.substring(2);
Assert.areNotEqual('xyz', sub, 'Characters not expected after first two'); // Succeeds
```

areNotEqual(notExpected, actual)

Asserts that the two arguments aren't the same.

Signature

```
public static void areNotEqual(Object notExpected, Object actual)
```

Parameters

notExpected

Type: Object

Actual value.

Return Value

Type: void

Usage

If the two arguments are the same, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String sub = 'abcde'.substring(2);
Assert.areNotEqual('xyz', sub); // Succeeds
```

fail(msg)

Immediately return a fatal error that causes code execution to halt.

Signature

```
public static void fail(String msg)
```

Parameters

msg

Type: String

(Optional) Custom message returned as part of the error message.

Return Value

Type: void

Usage

Commonly used in a try/catch block test case where an exception is expected to be thrown. You can't, however, catch the assertion failure in the try/catch block even though it's logged as an exception.

Example

```
// test case where exception is expected
try {
    SomeClass.methodUnderTest();
    Assert.fail('DmlException Expected');
} catch (DmlException ex) {
    // Add assertions here about the expected exception
}
```

fail()

Immediately return a fatal error that causes code execution to halt.

Signature

Usage

Commonly used in a try/catch block test case where an exception is expected to be thrown. You can't, however, catch the assertion failure in the try/catch block even though it's logged as an exception.

Example

```
// test case where exception is expected
try {
    SomeClass.methodUnderTest();
    Assert.fail();
} catch (DmlException ex) {
    // Add assertions here about the expected exception
}
```

isFalse(condition, msg)

Asserts that the specified condition is `false`.

Signature

```
public static void isFalse(Boolean condition, String msg)
```

Parameters

condition

Type: Boolean

Condition you're checking to determine if it's `false`.

msg

Type: String

(Optional) Custom message returned as part of the error message.

Return Value

Type: void

Usage

If the condition is `true`, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Boolean containsCode = 'Salesforce'.contains('code');
Assert.isFalse(containsCode, 'No code'); // Assertion succeeds
```

isFalse(condition)

Asserts that the specified condition is `false`.

Signature

```
public static void isFalse(Boolean condition)
```

Condition you're checking to determine if it's `false`.

Return Value

Type: void

Usage

If the condition is `true`, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Boolean containsCode = 'Salesforce'.contains('code');
Assert.isFalse(containsCode); // Assertion succeeds
```

isInstanceOfType(instance, expectedType, msg)

Asserts that the instance is of the specified type.

Signature

```
public static void isInstanceOfType(Object instance, System.Type expectedType, String msg)
```

Parameters

instance

Type: Object

Instance whose type you're checking.

expectedType

Type: System.Type

Expected type.

msg

Type: String

(Optional) Custom message returned as part of the error message.

Return Value

Type: void

Usage

If the instance isn't of the specified type, a fatal error is returned that causes code execution to halt

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Account o = new Account();
Assert.isInstanceOfType(o, Account.class); // Succeeds
```

isInstanceOfType(instance, expectedType)

Parameters

instance

Type: Object

Instance whose type you're checking.

expectedType

Type: System.Type

Expected type.

Return Value

Type: void

Usage

If the instance isn't of the specified type, a fatal error is returned that causes code execution to halt

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Account o = new Account();
Assert.isInstanceOfType(o, Account.class); // Succeeds
```

```
Account o = new Account();
Assert.isInstanceOfType(o, Account.class, 'Expected type.'); // Succeeds
```

isNotInstanceOfType(instance, notExpectedType, msg)

Asserts that the instance isn't of the specified type.

Signature

```
public static void isNotInstanceOfType(Object instance, System.Type notExpectedType, String msg)
```

Parameters

instance

Type: Object

Instance whose type you're checking.

notExpectedType

Type: System.Type

Type that's not expected.

msg

Type: String

(Optional) Custom message returned as part of the error message.

Return Value

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Contact con = new Contact();
Assert.isNotInstanceOfType(con, Account.class, 'Not expected type'); // Succeeds
```

isNotInstanceOfType(instance, notExpectedType)

Asserts that the instance isn't of the specified type.

Signature

```
public static void isNotInstanceOfType(Object instance, System.Type notExpectedType)
```

Parameters

instance

Type: Object

Instance whose type you're checking.

notExpectedType

Type: System.Type

Type that's not expected.

Return Value

Type: void

Usage

If the instance is of the specified type, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Contact con = new Contact();
Assert.isNotInstanceOfType(con, Account.class); // Succeeds
```

isNotNull(value, msg)

Asserts that the value isn't null.

Signature

```
public static void isNotNull(Object value, String msg)
```

Parameters

value

Type: Object

Value you're checking to determine if it's not null.

msg

Type: void

Usage

If the value is null, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String myString = 'value';
Assert.isNotNull(myString, 'myString should not be null'); // Succeeds
```

isNotNull(value)

Asserts that the value isn't null.

Signature

```
public static void isNotNull(Object value)
```

Parameters

value

Type: Object

Value you're checking to determine if it's not null.

Return Value

Type: void

Usage

If the value is null, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String myString = 'value';
Assert.isNotNull(myString); // Succeeds
```

isNull(value, msg)

Asserts that the value is null.

Signature

```
public static void isNull(Object value, String msg)
```

Parameters

value

Type: Object

Value you're checking to determine if it's null.

Return Value

Type: void

Usage

If the value isn't null, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String myString = null;
Assert.isNull(myString, 'String should be null'); // Succeeds
```

isNull(value)

Asserts that the value is null.

Signature

```
public static void isNull(Object value)
```

Parameters

value

Type: Object

Value you're checking to determine if it's null.

Return Value

Type: void

Usage

If the value isn't null, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
String myString = null;
Assert.isNull(myString); // Succeeds
```

isTrue(condition, msg)

Asserts that the specified condition is `true`.

Signature

```
public static void isTrue(Boolean condition, String msg)
```

Parameters

condition

Type: Boolean

Condition you're checking to determine if it's `true`.

Return Value

Type: void

Usage

If the specified condition is `false`, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Boolean containsForce = 'Salesforce'.contains('force');
Assert.isTrue(containsForce, 'Contains force'); // Assertion succeeds
```

isTrue(condition)

Asserts that the specified condition is `true`.

Signature

```
public static void isTrue(Boolean condition)
```

Parameters

condition

Type: Boolean

Condition you're checking to determine if it's `true`.

Return Value

Type: void

Usage

If the specified condition is `false`, a fatal error is returned that causes code execution to halt.

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

Example

```
Boolean containsForce = 'Salesforce'.contains('force');
Assert.isTrue(containsForce); // Assertion succeeds
```

Tableau
Commerce Cloud
Lightning Design System
Einstein
Quip

APIs
Trailhead
Sample Apps
Podcasts
AppExchange

Partner Community
Blog
Salesforce Admins
Salesforce Architects