InstallHandler Interface

Enables custom code to run after a managed package installation or upgrade.

Namespace

System

Usage

App developers can implement this interface to specify Apex code that runs automatically after a subscriber installs or upgrades a managed package. The package install or upgrade can be customized based on details of the subscriber's organization. For instance, you can use the script to populate custom settings, create sample data, send an email to the installer, notify an external system, or kick off a batch operation to populate a new field across a large set of data.

The post install script is invoked after tests have been run, and is subject to default governor limits. It runs as a special system user that represents your package, so all operations performed by the script appear to be done by your package. You can access this user by using UserInfo. You only see this user at runtime, not while running tests.

If the script fails, the install or upgrade is aborted. Any errors in the script are emailed to the user specified in the **Notify on Apex Error** field of the package. If no user is specified, the install or upgrade details are unavailable.

The post install script has the following additional properties.

- It can initiate batch, scheduled, and future jobs.
- It can't access Session IDs.
- It can only perform callouts using an async operation. The callout occurs after the script is run and the install is complete and committed.
- It can't call another Apex class in the package if that Apex class uses the `with sharing` keyword. This keyword can prevent the package from successfully installing. To learn more, see the Apex Developer Guide.

The `InstallHandler` interface has a single method called `onInstall`, which specifies the actions to be performed on install or upgrade.

```
public interface InstallHandler {
  void onInstall(InstallContext context)
};
```

The `onInstall` method takes a context object as its argument, which provides the following information.

- The org ID of the organization in which the installation takes place.
- The user ID of the user who initiated the installation.
- The version number of the previously installed package (specified using the `Version` class). The version is always a three-part number, such as 1.2.0.
- Whether the installation is an upgrade.
- Whether the installation is a push.

```
public interface InstallContext {
  ID organizationId();
  ID installerId();
  Boolean isUpgrade();
  Boolean isPush();
  Version previousVersion();
}
```

- InstallHandler Methods
- InstallHandler Example Implementation

InstallHandler Methods

The following are methods for `InstallHandler`.

- onInstall(context)
  Specifies the actions to be performed on install/upgrade.

onInstall(context)

Specifies the actions to be performed on install/upgrade.

Signature

```
public Void onInstall(InstallContext context)
```

Parameters

context

Type: System.InstallContext

Return Value

Type: Void

InstallHandler Example Implementation

The following sample post install script performs these actions on package install/upgrade.

- If the previous version is null, that is, the package is being installed for the first time, the script:
  - Creates a new Account called Newco and verifies that it was created.
  - Creates a new instance of the custom object Survey, called Client Satisfaction Survey.
  - Sends an email message to the subscriber confirming installation of the package.
- If the previous version is 1.0, the script creates a new instance of Survey called ”Upgrading from Version 1.0”.
- If the package is an upgrade, the script creates a new instance of Survey called ”Sample Survey during Upgrade”.
- If the upgrade is being pushed, the script creates a new instance of Survey called ”Sample Survey during Push”.

```
public class PostInstallClass implements InstallHandler {
  global void onInstall(InstallContext context) {
    if(context.previousVersion() == null) {
      Account a = new Account(name='Newco');
      insert(a);
  String[] toAddresses = new String[]{toAddress};
  Messaging.SingleEmailMessage mail =
   new Messaging.SingleEmailMessage();
  mail.setToAddresses(toAddresses);
  mail.setReplyTo('support@package.dev');
  mail.setSenderDisplayName('My Package Support');
  mail.setSubject('Package install successful');
  mail.setPlainTextBody('Thanks for installing the package.');
  Messaging.sendEmail(new Messaging.Email[] { mail });
  }
 else
  if(context.previousVersion().compareTo(new Version(1,0)) == 0) {
  Survey__c obj = new Survey__c(name='Upgrading from Version 1.0');
  insert(obj);
  }
 if(context.isUpgrade()) {
  Survey__c obj = new Survey__c(name='Sample Survey during Upgrade');
  insert obj;
  }
 if(context.isPush()) {
  Survey__c obj = new Survey__c(name='Sample Survey during Push');
  insert obj;
  }
 }
}
```

You can test a post install script using the new `testInstall` method of the `Test` class. This method takes the following arguments.

- A class that implements the `InstallHandler` interface.
- A `Version` object that specifies the version number of the existing package.
- An optional Boolean value that is `true` if the installation is a push. The default is `false`.

This sample shows how to test a post install script implemented in the `PostInstallClass` Apex class.

```
@isTest
static void testInstallScript() {
 PostInstallClass postinstall = new PostInstallClass();
  Test.testInstall(postinstall, null);
  Test.testInstall(postinstall, new Version(1,0), true);
  List<Account> a = [Select id, name from Account where name ='Newco'];
  System.assertEquals(1, a.size(), 'Account not found');
 }
```

DID THIS ARTICLE SOLVE YOUR ISSUE?

Let us know so we can improve!

Share your feedback

DEVELOPER CENTERS
Heroku
MuleSoft

POPULAR RESOURCES
Documentation
Component Library

COMMUNITY
Trailblazer Community
Events and Calendar

Quip AppExchange

© 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