d.velop

d.velop connect for salesforce
CRM-API

Inhaltsverzeichnis

1. d.velop connect for Salesforce CRM-API

1.1. Apex-Entwicklungsleitfaden

In diesem Abschnitt erhältst du einen Überblick über die verschiedenen API-Funktionen in der Entwicklung innerhalb von Salesforce.

1.1.1. Hochladen von Salesforce-Dateien in das DMS

In diesem Kapitel erfährst du, wie du Salesforce-Dateien (**ContentDocument**, **Attachment** und **EmailMessage**) als Dokumente in das DMS hochlädst. Du kannst die Dateien und Datensätze einzeln hochladen oder direkt alle Anhänge eines Ursprungsdatensatzes in das DMS migrieren.

Anfordern einer gültigen Benutzersitzung für den Prozess

Um sicherzustellen, dass der Hochladevorgang im inbegriffenen **Queueable**-Prozess mit einer gültigen Benutzersitzung durchgeführt wird, muss die Benutzersitzung vorher in d.velop documents angefordert werden. Die Benutzersitzung kannst du entweder manuell anfordern oder vom Prozess durchführen lassen.

Wie du eine gültige Benutzersitzung erhältst, erfährst du in den folgenden Abschnitten:

Inhalt

Authentifizieren im Vorfeld (empfohlen)

Um im Vorfeld eine gültige Benutzersitzung anzufordern, bietet die Klasse DocumentUploader verschiedene Methoden. Du kannst z.B. direkt eine Sitzung für den angemeldeten Benutzer bzw. Servicebenutzer abrufen. Alternativ kannst du auch eine Sitzung anhand eines zuvor erlangten Cookies (z.B. mithilfe von Methoden der Klasse OpenAPI) verwenden.

Im Folgenden erfährst du, wie du die verschiedenen Methoden verwendest.

Authentifizieren im Vorfeld – angemeldeter Benutzer

```
// Create the document uploader - This can happen anywhere in your code
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Authenticate as the currently logged in user
docUploader.authenticateInCurrentUserContext();

// Define upload request(s)
// ...

// Start the process
docUploader.startUpload(requests);
```

Authentifizieren im Vorfeld – Servicebenutzer

```
// Create the document uploader - This can happen anywhere in your code
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Authenticate as the service user
docUploader.authenticateInServiceUserContext();

// Define upload request(s)
// ...

// Start the process
docUploader.startUpload(requests);
```

Authentifizieren im Vorfeld – mit Cookie

```
// Create the document uploader - This can happen anywhere in your code
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Fetch a cookie (cached or completely fresh) and pass it to the uploader
that starts the process
String cookie = dvelop_docs_dev.OpenAPI.getValidCookieForConfigUser();
docUploader.setAuthentication(cookie);

// Define upload request(s)
// ...

// Start the process
docUploader.startUpload(requests);
```

Authentifizieren während des Prozesses

Für den Fall, dass im Vorfeld keine gültige Benutzersitzung erhalten wurde, fordert der Upload-Prozess einmalig mithilfe der internen **Queueable**-Klasse **AuthenticationJob** eine Sitzung in d.velop documents an. Dabei wird entweder eine bereits mittels **Salesforce Plattform Cache** zwischengespeicherte Sitzung verwendet oder eine komplett neue Sitzung in **Identityprovider** angefragt.

Du kannst das Verhalten von **AuthenticationJob** mithilfe der Klasse DocumentUploadOptions beeinflussen und so z.B. den Prozess mit den Zugangsdaten des Servicebenutzers ausführen.

> **Anmerkung**
> Wenn im Vorfeld keine Sitzung angefordert und **DocumentUploadOptions** nicht an den Prozess übergeben wurde, werden die **Fallback-Optionen** (**useConfigUser = false** – **skipValidate = false**) verwendet.

Nachfolgend erfährst du im Beispiel, wie du die Rahmenparameter des Prozesses mit der Klasse anpassen kannst.

Authentifizieren während des Prozesses – Beeinflussen der Optionen

```
// Create the document uploader - This can happen anywhere in your code
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Define your custom options
```

```
dvelop_docs_dev.DocumentUploadOptions options = new
dvelop_docs_dev.DocumentUploadOptions(true, false);

// Define upload request(s)
// ...

// Start the process - Use the overloaded method signature
"startUpload(requests, options)"
docUploader.startUpload(requests, options);
```

Siehe auch

- Verwenden der Klasse “DocumentUploader”
- Verwenden der Klasse "SubscriberInterface"
- Verwenden der Klasse “OpenAPI”
- Verwenden der Klasse "DocumentUploadOptions"

Hochladen von "ContentDocument"- und "Attachment"-Dateien

In diesem Abschnitt erfährst du, wie du einzelne oder mehrere **ContentDocument**- oder **Attachment**-Dateien aus Salesforce in das DMS hochlädst.

Inhalt

Hochladen einer ContentDocument-/Attachment-Datei

Im Kontext eines Datensatzes kannst du mithilfe der Klasse **ContentDocumentUploadRequest** und der Methode **startUpload(requests)** eine **ContentDocument**- oder **Attachment**-Datei hochladen.

Im nachfolgenden Beispiel erfährst du, wie du die Klasse verwendest.

```
// Create a new uploader instance
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Authenticate against connected DMS
docUploader.authenticateInCurrentUserContext();

// Define upload request(s)
dvelop_docs_dev.ContentDocumentUploadRequest request = new
dvelop_docs_dev.ContentDocumentUploadRequest();
request.relatedEntityId = '0019Z00000I7cpRQAR';
request.contentDocumentId = '0699Z000001ygrqQAA';
request.documenttypeKey = 'Schriftverkehr_Kunde';
request.preserveFileAfterUpload = true;

List<dvelop_docs_dev.IDocumentUploadRequest> uploadRequests = new
List<dvelop_docs_dev.IDocumentUploadRequest>{
    request
};

// Perform the action - Start the upload process
docUploader.startUpload(uploadRequests);
```

Um mehrere **ContentDocument-/Attachment**-Dateien oder andere Objekte nacheinander hochzuladen, erweitere die Liste um zusätzliche **IDocumentUploadRequest**-Instanzen.

Siehe auch

- Verwenden der Klasse “DocumentUploader”
- Verwenden der Klasse "ContentDocumentUploadRequest"

Hochladen von "EmailMessage"-Objekten

In diesem Kapitel erfährst du, wie du einzelne oder mehrere **EmailMessage**-Objekte in das DMS hochlädst.

Inhalt

Hochladen eines EmailMessage-Objekts

Im Kontext eines Datensatzes kannst du mithilfe der Klasse **EmailMessageUploadRequest** und der Methode **startUpload(requests)** ein **EmailMessage**-Objekt hochladen.

Im folgenden Beispiel siehst du, wie du die Klasse verwendest.

```
// Create a new uploader instance
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Authenticate against connected DMS
docUploader.authenticateInCurrentUserContext();

// Define upload request
dvelop_docs_dev.EmailMessageUploadRequest request = new
dvelop_docs_dev.EmailMessageUploadRequest();
request.relatedEntityId = '0019Z00000I7cpRQAR';
request.emailMessageIds = new List<Id>{ '02s9Z000003G7TeQAK' };
request.documenttypeKey = 'Schriftverkehr_Kunde';

List<dvelop_docs_dev.IDocumentUploadRequest> uploadRequests = new
List<dvelop_docs_dev.IDocumentUploadRequest>{
    request
};

// Perform the action - Start the upload process
docUploader.startUpload(uploadRequests);
```

Um mehrere **EmailMessage**- oder andere Objekte nacheinander hochzuladen, erweitere die Liste um zusätzliche **IDocumentUploadRequest**-Instanzen.

Siehe auch

- Verwenden der Klasse “DocumentUploader”
- Verwenden der Klasse "EmailMessageUploadRequest"

Hochladen neuer Versionen

In diesem Kapitel erfährst du, wie du **ContentDocument**- oder **Attachment**-Dateien als neue Versionen von DMS-Dokumenten hochlädst.

Inhalt

Hochladen von ContentDocument-/Attachment-Dateien als neue Version

Du kannst **ContentDocument** - und **Attachment**-Dateien mithilfe der Klasse **UploadNewVersionRequest** und der Methode **startUpload(requests)** als neue Version eines DMS-Dokuments hochladen.

Im folgenden Beispiel erfährst du, wie du die Klasse verwendest.

```
// Create a new uploader instance
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Authenticate against connected DMS
docUploader.authenticateInCurrentUserContext();

// Define upload request
dvelop_docs_dev.UploadNewVersionRequest request = new
dvelop_docs_dev.UploadNewVersionRequest();
request.contentDocumentId = '0699Z000001ygrqQAA';
request.dmsDocumentId = 'XH00020875';

List<dvelop_docs_dev.IDocumentUploadRequest> uploadRequests = new
List<dvelop_docs_dev.IDocumentUploadRequest>{
    request
};

// Perform the action - Start the upload process
docUploader.startUpload(uploadRequests);
```

Um mehrere neue Versionen oder Objekte nacheinander hochzuladen, erweitere die Liste um zusätzliche **IDocumentUploadRequest**-Instanzen.

Siehe auch

- Verwenden der Klasse “DocumentUploader”
- Verwenden der Klasse "UploadNewVersionRequest"

Migrieren von mit Datensätzen verbundenen Salesforce-Dateien und E-Mails

In diesem Kapitel erfährst Du, wie du mit Datensätzen verbundene **ContentDocument**- und **Attachment**-Dateien sowie **EmailMessage**-Objekte in das DMS migrierst.

Inhalt

Migrieren von Salesforce-Dateien und EmailMessage-Objekten

Du kannst Dateien und **EmailMessage** -Objekte mithilfe der Klasse **AttachmentMigrationRequest** und der Methode **startUpload(requests)** von einem bestimmten Datensatz lösen und migrieren. Die migrierten Elemente kannst du entweder am entsprechenden Ort beibehalten oder anschließend aus Salesforce löschen, um Speicherplatz einzusparen.

Im folgenden Beispiel erfährst du, wie du die Klasse verwendest.

```
// Create a new uploader instance
dvelop_docs_dev.DocumentUploader docUploader = new
dvelop_docs_dev.DocumentUploader();

// Authenticate against connected DMS
```

```
docUploader.authenticateInCurrentUserContext();

// Define upload request
dvelop_docs_dev.AttachmentMigrationRequest request = new
dvelop_docs_dev.AttachmentMigrationRequest();
request.relatedEntityId = '0019Z00000I7cpVQAR';
request.docTypeKey = 'Schriftverkehr_Kunde';
request.fileTypesToProcess = '*';
request.transferAndRemoveEmails = true;
request.preserveFilesAfterTransfer = true;
request.preserveEmailsAfterTransfer = true;

List<dvelop_docs_dev.IDocumentUploadRequest> uploadRequests = new
List<dvelop_docs_dev.IDocumentUploadRequest>{
    request
};

// Perform the action - Start the upload process
docUploader.startUpload(uploadRequests);
```

Um mehrere Datensätze nacheinander abzuarbeiten oder Objekte hochzuladen, erweitere die Liste um zusätzliche **IDocumentUploadRequest**-Instanzen.

Siehe auch

- Verwenden der Klasse “DocumentUploader”
- Verwenden der Klasse "AttachmentMigrationRequest"

1.1.2. Verwalten und Verarbeiten von DMS-Dokumenten

In diesem Kapitel erfährst du, wie du effizient und einfach in Apex mit DMS-Dokumenten arbeitest und DMS-Dokumente verwalten und verarbeiten kannst.

Authentifizieren in d.velop documents

In diesem Kapitel erfährst du, wie und aus welchen Gründen du die Klasse **DocumentManager** in d.velop documents authentifizierst.

Inhalt

Hintergrund

Um die Methoden der Klasse **DocumentManager** in deinem eigenen Code zu verwenden, ist eine vorherige Authentifizierung nicht zwingend erforderlich. Dennoch empfehlen wir dir, eine Instanz einmalig mithilfe der verschiedenen Authentifizierungsmethoden mit einer gültigen Benutzersitzung zu versehen.

Auf diese Weise stellst du sicher, dass eine gültige Sitzung existiert und alle ausgehenden HTTP-Anfragen an d.velop documents im Kontext des jeweiligen Benutzers ausgeführt werden.

Zudem vermeidest du dadurch, dass eine im Salesforce Platform-Cache zwischengespeicherte Benutzersitzung noch einmal in der Identityprovider-App validiert werden muss. In diesem Falle müsstest du die Benutzersitzung unter Umständen neu anfordern, was in Salesforce potenziell zu einem **Mixed DML Operation**-Fehler führen kann.

Authentifizieren der Klasse

Im folgenden Beispiel erfährst du, wie du die Klasse **DocumentManager** schnell und einfach authentifizieren kannst.

```
// Create the document manager
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Pick one of the authentication methods and authenticate
documentManager.authenticateInCurrentUserContext();

// Perform an operation
// ...
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse "SubscriberInterface"

Aktualisieren eines DMS-Dokuments

In diesem Kapitel erfährst du, wie du Eigenschaften und Kategorien eines DMS-Dokuments aktualisierst.

Inhalt

Aktualisieren eines Dokuments

Um ein DMS-Dokument zu aktualisieren, verwende die Methode **updateDocument(documentId, updatedCategory, updatedAttributes)** der Klasse DocumentManager.

Im folgenden Beispiel erfährst du, wie du die Methode anwendest.

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Define the updated details
List<dvelop_docs_dev.DocumentAttribute> updatedAttributes = new
List<dvelop_docs_dev.DocumentAttribute>{
    new dvelop_docs_dev.DocumentAttribute('objecttitle', 'Ja'),
    new dvelop_docs_dev.DocumentAttribute('multivalue_property__c', new
List<String>{ 'A', 'B' })
};

// Perform the action - Update the document
documentManager.updateDocument('XH00014213', 'correspondence__c',
updatedAttributes);
```

Fehlerbehebung

Beim Aktualisieren eines Dokuments erhältst du unter Umständen folgende Fehlermeldung: "A mapping is not possible because several values were transferred to the same destination".

Im Folgenden erfährst du möglichen Fehlerursachen und Lösungen:

Ein einzelnes Attribut ist mehrfach in der Liste der Attribute vorhanden

Stelle sicher, dass die übermittelte Liste jedes Attribut maximal einmal enthält. Identische Attribute kannst du entfernen, indem du die Liste in ein Set umwandelst. Anschließend kannst du das Set wieder zurück in eine Liste konvertieren.

Mehrere Quelleigenschaften sind demselben d.3-Zielfeld zugeordnet

Prüfe die Zuordnungen und entferne eventuell vorhandene doppelte Zuordnungen.

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse “DocumentAttribute”

Abrufen eines DMS-Dokuments

In diesem Kapitel erfährst du, wie du Eigenschaften und Kategorien eines DMS-Dokuments aus dem verbundenen DMS ermittelst.

Inhalt

Abrufen eines Dokuments

Um ein einzelnes Dokument aus dem DMS abzurufen, verwende die Methode **getDocument(documentId)** der Klasse DocumentManager.

Im folgenden Beispiel erfährst du, wie du die Methode anwendest.

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Perform the action - Get the document and work with ouput
dvelop_docs_dev.Document dmsDocument =
documentManager.getDocument('XH00014213');
System.debug(dmsDocument);
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse "Document"

Erstellen einer Akte

In diesem Kapitel erfährst du, wie du eine Akte erstellst.

Inhalt

Erstellen einer Akte

Um eine Akte im DMS zu erstellen, verwende die Methode **createFolder(category, attributes)** der Klasse DocumentManager.

Im folgenden Beispiel erfährst du, wie du die Methode anwendest.

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();
```

```
// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Define the details
List<dvelop_docs_dev.DocumentAttribute> attributes = new
List<dvelop_docs_dev.DocumentAttribute>{
    new dvelop_docs_dev.DocumentAttribute('objecttitle', 'Account4711'),
    new dvelop_docs_dev.DocumentAttribute('objectkey',
'001AW00000B53c5YAB'),
    new dvelop_docs_dev.DocumentAttribute('account', 'Account4711'),
    new dvelop_docs_dev.DocumentAttribute('accountnumber',
'001AW00000B53c5YAB'),
    new dvelop_docs_dev.DocumentAttribute('foldertype', 'Account')
};

// Perform the action - Create the folder
documentManager.createFolder('akte_safo__c',  attributes);
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse “DocumentAttribute”

Herunterladen von DMS-Dokumenten

In diesem Kapitel erfährst du, wie du Dokumente aus dem DMS herunterladen und an einen Datensatz anhängst.

Inhalt

Technische Limitierungen

Wie bei allen Apex-Transaktionen musst du auch beim Herunterladen von Dokumenten die Salesforce Governor Limits beachten. Folgende technische Limitierungen gelten hinsichtlich der Größe von HTTP-Antworten:

- In einer synchronen Transaktion können nur Dokumente mit einer maximalen Dateigröße von 6 MB heruntergeladen werden.
- In einer asynchronen Transaktion können nur Dokumente mit einer maximalen Dateigröße von 12 MB heruntergeladen werden.

Herunterladen eines Dokuments

Um ein einzelnes Dokument aus dem DMS herunterzuladen, verwende die Methoden **downloadDocument(documentId)** und **downloadDocumentToRecord(recordId, documentId)** der Klasse DocumentManager.

In den folgenden Beispielen erfährst du, wie du die beiden Methoden anwendest.

Herunterladen eines Dokuments – Weiterverarbeiten des Dokuments

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
```

```
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Perform the action - Download the document and work with ouput
dvelop_docs_dev.DocumentDownloadResult downloadResult =
documentManager.downloadDocument('XH00014213');
System.debug(downloadResult.getFilename());
System.debug(downloadResult.getBody());
```

Herunterladen eines Dokuments – Speichern des Dokuments am Datensatz

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Perform the action - Download the document and attach it to a record
Id createdContentDocumentId =
documentManager.downloadDocumentToRecord('001Aa0000059kyPIAQ',
'XH00014213');
System.debug(createdContentDocumentId);
```

Herunterladen mehrerer Dokumente

Um mehrere Dokumente mit einer Aktion aus dem DMS herunterzuladen, verwende die Methode **downloadDocuments(documentIds,finisher, proceedOnFailure)** der Klasse DocumentManager.

Da die Methode mehrere potenziell große Dateien für die DMS-Dokumente herunterlädt, werden die Downloads in einem asynchronen Prozess in einer Queue ausgeführt. Damit du auf den Abschluss der Queue reagieren kannst und Zugriff auf die heruntergeladenen Dokumente erhältst, musst du einen **Queueable**-Job vom Typ dvelop_docs_dev.AsyncProcessFinisher beim Aufruf der Methode übermitteln. Dieser Job wird nach Abschluss des letzten Downloads an die Queue angehängt. Die Ausgabe (die heruntergeladenen Dokumente) wird als Liste aus dvelop_docs_dev.DocumentDownloadResults übertragen.

Im folgenden Beispiel erfährst du, wie du einen Finish-Job definierst, um mehrere Dokumente asynchron herunterzuladen:

Herunterladen mehrerer Dokumente – Definieren des Finish-Jobs

```
public class DownloadProcessFinishJob extends
dvelop_docs_dev.AsyncProcessFinisher {
    private static final String DEFAULT_PARAM_NAME = 'downloadResults';

    public DownloadProcessFinishJob() {
        // Call of parent constructor with an identifier for the default
parameter name
        super(DEFAULT_PARAM_NAME);
    }

    public override void execute(Map<String, Object> processOutput) {
        // Read the default parameter and cast to correct type
        Object defaultParamOutput = processOutput.get(DEFAULT_PARAM_NAME);
        List<dvelop_docs_dev.DocumentDownloadResult> downloadResults =
```

```
(List<dvelop_docs_dev.DocumentDownloadResult>) defaultParamOutput;

        // Process the results
        for (dvelop_docs_dev.DocumentDownloadResult result :
downloadResults) {
            System.debug(result.getFilename());
            System.debug(result.getBody());
        }
    }
}
```

Herunterladen mehrerer Dokumente – Start des Prozesses

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Defining download params
List<String> documentIds = new List<String>{ 'XH00014226', 'XH00014222',
'XH00014225' };
DownloadProcessFinishJob finishJob = new DownloadProcessFinishJob();

// Perform the action - Download the documents in a queue
documentManager.downloadDocuments(documentIds, finishJob, false);
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse "DocumentDownloadResult"
- Verwenden der Klasse "AsyncProcessFinisher"

Löschen von DMS-Dokumenten

Dieses Kapitel zeigt dir, wie du Dokumente und Akten aus dem DMS löschst.

Inhalt

Löschen eines DMS-Objekts

Um ein Dokument bzw. Objekt aus dem DMS zu löschen, verwende die Methode **deleteDocument(documentId, ?reason)** der Klasse DocumentManager.

Im folgenden Beispiel erfährst du, wie du die Methode anwendest.

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Perform the action - Delete the document
documentManager.deleteDocument('XH00014213', 'My custom reason');
```

Siehe auch

- Verwenden der Klasse “DocumentManager”

Suchen nach DMS-Dokumenten

In diesem Kapitel erfährst du, wie du Dokumente an einem bestimmten Salesforce-Datensatz oder Salesforce-Objekt im DMS suchst und findest.

Inhalt

Suchen nach DMS-Dokumenten

In diesem Beispiel siehst du, wie du mithilfe der Methode **search(options)** der Klasse DocumentManager eine einfache Suche nach Dokumenten an einem bestimmten Datensatz durchführst.

Die Suchkriterien, mit denen die Dokumente eingeschränkt werden, werden standardmäßig aus den in der d.velop documents-Konfiguration vorgenommenen Einstellungen ermittelt.

> **Anmerkung**
> Die Methode **search(options)** benötigt eine Instanz der Klasse **DocumentSearchOptions** als Eingabe.
>
> Wie du Suchparameter vereinfacht erstellst, erfährst du unter Verwenden der Klasse “DocumentSearchOptions.Builder”.

Suchen nach DMS-Dokumenten – einfache Suche

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Define the search options
dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useRecordContext('00011100001tBgaxAAC');

// Perform the action - Search for documents using the defined options
List<dvelop_docs_dev.Document> documents =
documentManager.search(searchOptions);
```

Suchen nach DMS-Dokumenten – Ermitteln aller Dokumente an einem Salesforce-Datensatz

```
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();
```

```
documentManager.authenticateInCurrentUserContext();

dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useRecordContext('00011100001tBgaxAAC');

List<dvelop_docs_dev.Document> documents =
documentManager.search(searchOptions);
```

Suchen nach DMS-Dokumenten – Ermitteln aller Dokumente an einem Salesforce-Objekt

```
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();
documentManager.authenticateInCurrentUserContext();

dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useObjectContext('Account');

List<dvelop_docs_dev.Document> documents =
documentManager.search(searchOptions);
```

Suchen nach DMS-Dokumenten – Ermitteln aller Dokumente an einem Salesforce-Objekt mit überschriebenen Suchkriterien

```
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();
documentManager.authenticateInCurrentUserContext();

dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useObjectContext('Account');
searchOptions.addSearchAttribute('objectkey', '0100X00QGA');
searchOptions.addSearchAttribute('string1', 'Max Mustermann');

List<dvelop_docs_dev.Document> documents =
documentManager.search(searchOptions);
```

Suchen von Dokumenten anhand von IDs

Du kannst mehrere Dokumente auch anhand der jeweiligen IDs ermitteln. Verwende dafür die Methode **searchDocumentsWithIds(searchAttributeKey, documentIds)** der Klasse DocumentManager.

```
dvelop_docs_dev.DocumentManager docManager = new
dvelop_docs_dev.DocumentManager();
docManager.authenticateInCurrentUserContext();

List<String> documentIds = new List<String>{ 'C200000024', 'C200000023' };

List<dvelop_docs_dev.Document> documents =
docManager.searchDocumentsWithIds('externaldocumentnumber', documentIds);
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse “DocumentSearchOptions”
- Verwenden der Klasse "Document"

Teilen von DMS-Dokumenten

In diesem Kapitel erfährst du, wie du DMS-Dokumente mit weiteren Personen teilst.

Inhalt

Erstellen von Links zum Teilen von Dokumenten

Um Links zum Teilen von Dokumenten zu erstellen, verwende die Methode **shareDocuments(downloadUrls)** der Klasse DocumentManager.

Im folgenden Beispiel erfährst du, wie du die Methode anwendest.

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Get a document
dvelop_docs_dev.Document dmsDocument =
documentManager.getDocument('FP00000010');

// Define the urls
Set<String> downloadUrls = new Set<String>{
    dmsDocument.downloadUrl
};

// Perform the action - Create the temporary links
Map<String, String> linksByDownloadUrl =
documentManager.shareDocuments(downloadUrls);
System.debug(linksByDownloadUrl.get(dmsDocument.downloadUrl));
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse "Document"

Versenden von E-Mails mit Anhängen

In diesem Kapitel erfährst du, wie du E-Mails mit Salesforce- und DMS-Dokumenten als Anhang versendest.

Inhalt

Technische Limitierungen

Wie bei allen Apex-Transaktionen musst du auch beim Versenden von E-Mails mit Dokumenten die Salesforce Governor Limits beachten. Durch die maximale Heap Size (Größe des dynamischen Datenspeichers) für E-Mail-Services ergeben sich folgende technische Limitierungen:

- Die Anhänge der E-Mail dürfen zusammen maximal 36 MB groß sein.
- Da DMS-Dokumente vor dem Versenden temporär aus dem DMS heruntergeladen werden müssen, gelten zusätzlich die im Kapitel Herunterladen von DMS-Dokumenten beschriebenen technischen Limitierungen.

Versenden einer E-Mail

Um Dokumente aus Salesforce und dem DMS via E-Mail zu versenden, verwende die Methode **sendEmail(options)** der Klasse DocumentManager.

Im folgenden Beispiel erfährst du, wie du die Methode anwendest.

```
// Create a new manager instance
dvelop_docs_dev.DocumentManager documentManager = new
dvelop_docs_dev.DocumentManager();

// Authenticate against connected DMS
documentManager.authenticateInCurrentUserContext();

// Define the email options
dvelop_docs_dev.DocumentEmailOptions emailOptions = new
dvelop_docs_dev.DocumentEmailOptions();
emailOptions.setSubject('Important Message');
emailOptions.setBody('Have a look at this.');
emailOptions.addRecipient('astro@salesforce.com');
emailOptions.addContentDocumentId('069AP0000006hQbYAI');
emailOptions.addDmsDocumentId('XH00014846');

// Perform the action - Send an email with documents from Salesforce and
the DMS
documentManager.sendEmail(emailOptions);
```

Siehe auch

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse “DocumentEmailOptions”

1.1.3. Dynamisches Ermitteln von Werten mit Apex

In der Konfiguration von d.velop documents können Zuordnungen (Mappings) zwischen Salesforce-Objekten und Datensatztypen erstellt werden. Somit wird eine direkte Verbindung zwischen Salesforce-Feldern und Eigenschaften von DMS-Dokumenten geschaffen. Mit diesem No-Code-Ansatz lassen sich viele Anwendungsfälle einfach und ohne Programmierkenntnisse umsetzen. Für komplexere Anwendungsfälle, bei denen Werte nicht aus Datensätzen und anderen Salesforce-Quellen ermittelt werden können, kannst du die d.velop documents-Konfiguration mit einer eigenen Apex-Logik erweitern. Somit kannst du die Werte eines Datensatzes dynamisch ermitteln.

In folgendem Beispiel erfährst du, wie du mithilfe der Klasse **dvelop_docs_dev.RecordValueProvider** eine dynamische Werteermittlung erstellst.

Beispiel

```
// Extend the provider parent class to be able to configure your class in
the d.velop documents configuration
global class CustomRecordValueProvider extends
dvelop_docs_dev.RecordValueProvider {

    // Implement the "provide" method, which is then called by the internal
process you want to integrate in
    global override dvelop_docs_dev.RecordValueProviderOutput
provide(dvelop_docs_dev.RecordValueProviderInput input) {
        // Read info from the given input
        Id recordId = input.getRecordId();
```

```
        // Work with the input data / perform further logic / call an
endpoint
        List<System.SelectOption> selectOptions = new
List<System.SelectOption>{
            new System.SelectOption('valueA', 'Label A'),
            new System.SelectOption('valueB', 'Label B'),
            new System.SelectOption(recordId, 'Record ID')
        };

        // Use one of the factory methods to return an appropriate output
object for the internal process
        return
dvelop_docs_dev.RecordValueProviderOutput.picklist(selectOptions);
    }

}
```

1.1.4. Testen von d.velop-Apex-Code

In Apex kannst du Unit- und Integration-Tests auf verschiedene Arten und Weisen schreiben. Salesforce bietet dir mehrere Möglichkeiten, Tests effizient, sicher und zuverlässig zu gestalten.

In diesem Kapitel erfährst du, wie du deine verwendeten Apex-Klassen in deinen Tests erfasst. Außerdem erfährst du, auf was du dabei achten musst, um Probleme zu vermeiden.

Generieren der Testumgebung

Der Aufbau und das Einfügen von Testdaten ist häufig aufwändig und ineffizient. Vermeide diesen Aufwand bei Unit-Tests, da schlecht aufgebaute Unit-Tests meist teuer und unübersichtlich sind.

> **Anmerkung**
> Wie du ineffizient generierte Testdaten und unabhängige, isolierte Unit-Tests vermeidest, erfährst du im Kapitel Verwenden von Mocks für isolierte Unit-Tests.

Verwende für Integration- oder Unit-Tests mit Zugriff auf Testdaten die Klasse dvelop_docs_dev.OpenAPI. Die Methode **initializeTestEnvironment()** sorgt dafür, dass nur wenige Testdaten und HTTP-Callout-Mocks aufgebaut werden. Auf diese Weise kannst du die d.velop-Klassen und Methoden erfolgreich in deinen Tests aufrufen.

Im folgenden Beispiel erfährst du, wie du die Methode verwendest:

```
@IsTest
private class MyTestClass {
    @TestSetup // Notice the "TestSetup" annotation
    static void makeData() {
        dvelop_docs_dev.OpenAPI.initializeTestEnvironment();
    }

    @IsTest
    static void myUnitTest() {
        // TODO: implement unit test
    }
}
```

Verwenden von Mocks für isolierte Unit-Tests

Das sogenannte Mocking ist ein häufig verwendetes Muster in der Softwareentwicklung, um effiziente, isolierte und zuverlässige Unit-Tests schreiben zu können. Aus diesem Grund wurden Mocking-Frameworks wie Javas Mockito entwickelt.

Auch mit Salesforce kannst du Abhängigkeiten zwischen Apex-Klassen mithilfe der Stub API simulieren. Auf diese Weise kannst du Tests unabhängig voneinander und ohne voriges Einfügen von Testdaten durchführen. ApexMocks ist eines der führenden Mocking-Frameworks, das sich stark an Mockito orientiert und auf dem Konzept der Dependency Injection basiert.

Du kannst Abhängigkeiten zwischen Klassen in Salesforce auch ohne Frameworks mit der Stub API simulieren, etwa d.velop-Klassen wie z.B. dvelop_docs_dev.DocumentManager.

> **Anmerkung**
> Beim Simulieren einer Apex-Klasse mit der Stub API kannst du lediglich nicht-statische Methoden simulieren. Klassen wie dvelop_docs_dev.OpenAPI können aufgrund ihres ausschließlich statischen Aufbaus nicht effektiv mit der Stub API oder etwaigen Frameworks simuliert werden.

Nachfolgend findest du ein Beispiel, in dem die Stub API ohne Framework zusammen mit d.velop-Klassen verwendet wird.

Verwenden der Stub API mit d.velop-Klassen

In diesem Beispiel sollen Unit-Tests für die Klasse **MyConsumerClass** geschrieben werden. Diese Klasse verwendet die Methode **getDocument(documentId)** der Klasse **DocumentManager** und ist von der jeweiligen Methode abhängig. Ein Unit-Test ohne Mocking stellt sicher, dass alle notwendigen Testdaten und HTTP-Mocks generiert und festgelegt sind.

```
public with sharing class MyConsumerClass {
    private final dvelop_docs_dev.DocumentManager docManager;

    // The document manager is injected into the constructor as a dependency
    public MyConsumerClass(dvelop_docs_dev.DocumentManager docManager) {
        this.docManager = docManager;
    }

    public dvelop_docs_dev.Document doSomeMagic(String documentId) {
        if (String.isBlank(documentId)) {
            throw new System.IllegalArgumentException('Document ID must not
be blank');
        }

        return docManager.getDocument(documentId);
    }
}
```

Durch die Verwendung der Stub API kannst du diese Abhängigkeit durch einen Stub ersetzen. Dafür erstellst du eine Hilfsklasse, die das **System.StubProvider**-Interface verwendet. Genauere Details und Hinweise zur Verwendung des Interfaces erhältst du im Apex-Leitfaden.

```
@IsTest
public class MyStubProvider implements System.StubProvider {
    // Implement the handleMethodCall method
    public Object handleMethodCall(
        Object stubbedObject,
        String stubbedMethodName,
        Type returnType,
        List<Type> listOfParamTypes,
        List<String> listOfParamNames,
        List<Object> listOfArgs
    ) {
```

```
        // Check the name of the called method on the mocked instance
(dvelop_docs_dev.DocumentManager) and return a mocked value
        switch on stubbedMethodName {
            when 'getDocument' {
                dvelop_docs_dev.Document mockedDocument = new
dvelop_docs_dev.Document();
                mockedDocument.id = '123';
                return mockedDocument;
            }
            when else {
                return null;
            }
        }
    }
}
```

Abschließend können die Tests für die Klasse **MyConsumerClass** mit **OpenAPI.createStub(parentType, stubProvider)** auf den Provider (**MyStubProvider**) und auf die simulierten Rückgabewerte zugreifen.

> **Anmerkung**
> Da die Stub API lediglich die Erstellung von Stubs für Klassen im selben Namespace erlaubt, musst du den Aufruf innerhalb der Klasse **OpenAPI** im Namespace **dvelop_docs_dev** durchführen, anstelle **Test.createStub(parentType, stubProvider)** aufzurufen.

```
@IsTest
private class MyTestClass {
    static dvelop_docs_dev.DocumentManager stubbedManager =
(dvelop_docs_dev.DocumentManager) dvelop_docs_dev.OpenAPI.createStub(
        dvelop_docs_dev.DocumentManager.class,
        new MyStubProvider()
    );

    static MyConsumerClass underTest = new MyConsumerClass(stubbedManager);

    @IsTest
    static void myUnitTest() {
        Test.startTest();
        dvelop_docs_dev.Document result = underTest.doSomeMagic('123');
        Test.stopTest();

        Assert.areEqual('123', result.id, 'Document ID should be 123');
    }
}
```

1.2. Apex-Referenz

In diesem Kapitel erhältst du Details und Hinweise zu Apex-Klassen und Apex-Schnittstellen im Namespace **dvelop_docs_dev**.

1.2.1. Verwenden der Klasse "AsyncProcessFinisher"

Du kannst die Klasse **dvelop_docs_dev.AsyncProcessFinisher** verwenden, um einen asynchronen **Queueable**-Prozess abzuschließen und dabei auf die Ausgabe des Prozesses zu reagieren.

Der **AsyncProcessFinisher**-Job implementiert das Queueable-Interface, das du über **System.enqueueJob()** starten kannst.

Signatur

```
global with sharing virtual class AsyncProcessFinisher implements
IServiceProcessFinisher
```

Konstruktoren

Du kannst eine Instanz des **AsyncProcessFinisher**-Jobs mit folgenden Konstruktoren erzeugen:

- AsyncProcessFinisher(defaultParamName)

AsyncProcessFinisher(defaultParamName)

Erzeugt einen **AsyncProcessFinisher**-Job mit dem Namen des Standardparameters, der für die Ausgabe der meisten asynchronen Prozesse verwendet wird.

Signatur

```
global AsyncProcessFinisher(String defaultParamName)
```

Parameter

defaultParamName: Der Name des Standardausgabeparameters. Dieser Parameter wird von den meisten asynchronen Prozessen verwendet, um die Ausgabe der Prozesse zu übermitteln.

Datentyp: **String**

Methoden

Die Klasse **AsyncProcessFinisher** bietet folgende Methoden:

- execute(processOutput)

execute(processOutput)

Die Methode führt eine eigene Logik nach Abschluss eines **Queueable**-Prozesses aus. Um die ausgeführte Logik zu beeinflussen, überschreibe die Methode in deiner eigenen Implementierung.

Signatur

```
protected virtual void execute(Map<String, Object> processOutput)
```

Parameter

processOutput: Eine Wörterbuchsammlung von Ausgabewerten des vorangegangenen Prozesses.

- **Datentyp**: **Map<String, Object>**
- **Standardwert**: **new Map<String, Object>()**

Verwendung

Das folgende Kapitel zeigt Dir einige Anwendungsfälle, die die richtige Verwendung der Klasse darstellen.

Beispiel

```
public class CustomFinishJob extends dvelop_docs_dev.AsyncProcessFinisher {
    private static final String DEFAULT_PARAM_NAME = 'default';

    public CustomFinishJob() {
        super(DEFAULT_PARAM_NAME);
    }

    public override void execute(Map<String, Object> processOutput) {
```

```
        System.debug(processOutput);

        Object defaultOutput = processOutput.get(DEFAULT_PARAM_NAME);
        System.debug(defaultOutput);
    }
}
```

1.2.2. Verwenden der Klasse "AttachmentMigrationRequest"

Die Klasse **dvelop_docs_dev.AttachmentMigrationRequest** definiert die Parameter, mit denen **EmailMessage**-Objekte und Dateien an einem Datensatz in das DMS migriert werden sollen.

Signatur

Verwenden des Interfaces "IDocumentUploadRequest"

```
global with sharing class AttachmentMigrationRequest implements
IDocumentUploadRequest
```

Konstruktoren

Du kannst einen **AttachmentMigrationRequest** mit folgenden Konstruktoren erzeugen:

- AttachmentMigrationRequest()

AttachmentMigrationRequest()

Der Konstruktor erstellt einen **AttachmentMigrationRequest** ohne Parameter.

Signatur

```
global AttachmentMigrationRequest()
```

Eigenschaften

Die Klasse **AttachmentMigrationRequest** hat folgende Eigenschaften:

- **relatedEntityId**: Die ID eines Datensatzes, von dem Elemente migriert werden sollen.
  Datentyp: **String**
- **docTypeKey**: Der Schlüssel der Kategorie, mit der die Dokumente im DMS gespeichert werden sollen.
  Datentyp: **String**
- **fileTypesToProcess**: Eine durch Kommata getrennte Liste von Dateitypen (Dateiendungen), die vom Datensatz migriert werden sollen. Um alle Dateitypen zu migrieren, gib * an.
  - **Datentyp**: **String**
  - **Standardwert**: *
- **preserveFilesAfterTransfer**: Legt fest, ob die Dateien nach erfolgreichem Hochladen in Salesforce gespeichert werden sollen.
  - **Datentyp**: **Boolean**
  - **Standardwert**: **false**
- **transferAndRemoveEmails**: Legt fest, ob E-Mails ebenfalls migriert werden sollen.
  - **Datentyp**: **Boolean**
  - **Standardwert**: **true**
- **preserveEmailsAfterTransfer**: Legt fest, ob E-Mails nach erfolgreichem Hochladen in Salesforce gespeichert werden sollen.
  - **Datentyp**: **Boolean**
  - **Standardwert**: **false**
- **reallocateRemainingFiles**: Legt fest, ob die Anhänge der E-Mails für den Datensatz (zweite Ebene) nach erfolgreichem Hochladen der E-Mail zum Ursprungsdatensatz hinzugefügt werden sollen.
  - **Datentyp**: **Boolean**

  - **Standardwert**: **false**
- **documentAttributes**: Eine Liste von Eigenschaften, die berechnete Eigenschaften (aus Datensatz und Kategorie) überschreiben sollen.
  Datentyp: **List<**dvelop_docs_dev.DocumentAttribute**>**
- **useConfigUserForUpload**: Legt fest, ob der Prozess mit den Zugangsdaten des Servicebenutzers anstelle des Prozesses des aktuell angemeldeten Benutzers ausgeführt wird.

> **Anmerkung**
> Die Eigenschaft **useConfigUserForUpload** ist veraltet und wird nur noch in Flows unterstützt. Um den Uploadprozess mit anderen Zugangsdaten auszuführen, verwende stattdessen die Klasse DocumentUploadOptions.

  - **Datentyp**: **Boolean**
  - **Standardwert**: **false**

1.2.3. Verwenden der Klasse "ContentDocumentUploadRequest"

Die Klasse **dvelop_docs_dev.ContentDocumentUploadRequest** definiert die Parameter, mit denen eine **ContentDocument**- oder **Attachment**-Datei aus Salesforce in das DMS hochgeladen werden soll.

Signatur

Verwenden des Interfaces "IDocumentUploadRequest"

```
global class ContentDocumentUploadRequest implements IDocumentUploadRequest
```

Konstruktoren

Du kannst einen **ContentDocumentUploadRequest** mit folgenden Konstruktoren erzeugen:

- ContentDocumentUploadRequest()

ContentDocumentUploadRequest()

Der Konstruktor erstellt einen **ContentDocumentUploadRequest** ohne Parameter.

Signatur

```
global ContentDocumentUploadRequest()
```

Eigenschaften

Die Klasse **ContentDocumentUploadRequest** hat folgende Eigenschaften:

- **relatedEntityId**: Die ID eines Datensatzes, in dessen Kontext die Datei hochgeladen werden soll.
  Datentyp: **String**
- **contentDocumentId**: Die ID der **ContentDocument**- oder **Attachments**-Datei, die hochgeladen werden soll.
  Datentyp: **String**
- **documenttypeKey**: Der Schlüssel der Kategorie, mit der das Dokument im DMS abgelegt werden soll.
  Datentyp: **String**
- **preserveFileAfterUpload**: Legt fest, ob der Datei-Datensatz nach erfolgreichem Hochladen in Salesforce behalten werden soll.
  - **Datentyp**: **Boolean**
  - **Standardwert**: **false**
- **documentAttributes**: Eine Liste von Eigenschaften, die berechnete Eigenschaften (aus Datensatz und Kategorie) überschreiben sollen.
  Datentyp: **List<**dvelop_docs_dev.DocumentAttribute**>**
- **useConfigUserForUpload**: Legt fest, ob der Prozess mit den Zugangsdaten des Servicebenutzers anstelle des Prozesses des aktuell angemeldeten Benutzers ausgeführt wird.

> **Anmerkung**
> Die Eigenschaft **useConfigUserForUpload** ist veraltet und wird nur noch in Flows unterstützt. Um den Uploadprozess mit anderen Zugangsdaten auszuführen, verwende stattdessen die Klasse DocumentUploadOptions.

- **Datentyp**: **Boolean**
- **Standardwert**: **false**

1.2.4. Verwenden der Klasse "Document"

Die Klasse **dvelop_docs_dev.Document** umfasst alle Metadaten eines DMS-Dokuments und erlaubt die strukturierte und einfache Arbeit mit Dokumenten in Apex.

Signatur

```
global with sharing class Document
```

Konstruktoren

Du kannst eine Instanz von **Document** mit folgenden Konstruktoren erzeugen:

- Document()

Document()

Die Methode erstellt eine einfache **Document**-Instanz.

Signatur

```
global Document()
```

Eigenschaften

Die Klasse **Document** hat folgende Eigenschaften:

- **id**: Die eindeutige ID des DMS-Dokuments.
  Datentyp: **String**
- **viewingUrl**: Eine absolute URL, die zur Anzeige des Dokuments in d.velop documents verwendet werden kann.
  Datentyp: **String**
- **title**: Der Titel (**filecaption**) des Dokuments.
  Datentyp: **String**
- **downloadUrl**: Eine absolute URL, die zum Herunterladen des Dokuments verwendet werden kann.
  Datentyp: **String**
- **categories**: Eine Liste der Schlüssel der Kategorien, die der DMS-Kategorie des Dokuments zugeordnet sind.
  Datentyp: **List<String>**
- **defaultCategory**: Die erste Kategorie aus der Liste **categories**.
  Datentyp: **String**
- **attributes**: Die Liste an Attributen des Dokuments.
  Datentyp: **List<**dvelop_docs_dev.DocumentAttribute**>**

Methoden

Die Klasse **Document** bietet folgende Methoden:

- getDocumentAttributes()

getDocumentAttributes()

Die Methode gibt die Attribute des Dokuments im generischen Format aus.

Signatur

```
global Map<String, Object> getDocumentAttributes()
```

Rückgabe

Die Attribute des Dokuments in einem generischen Wörterbuch. Die Werte können ein einzelner Textwert (**String**) oder eine Liste an Textwerten (**List<String>**) sein.

1.2.5. Verwenden der Klasse “DocumentAttribute”

Die Klasse **dvelop_docs_dev.DocumentAttribute** kapselt alle Metadaten und Informationen eines Attributs bzw. einer Eigenschaft von DMS-Dokumenten.

Signatur

```
global class DocumentAttribute
```

Konstruktoren

Du kannst eine Instanz von **DocumentAttribute** mit folgenden Konstruktoren erzeugen:

- DocumentAttribute()
- DocumentAttribute(key, value)

DocumentAttribute()

Dieser Konstruktor erstellt eine einfache **DocumentAttribute**-Instanz.

Signatur

```
global DocumentAttribute()
```

DocumentAttribute(key, value)

Dieser Konstruktor erstellt eine **DocumentAttribute**-Instanz mit einem Schlüssel und einem oder mehreren Werten des Attributs.

Signatur

```
global DocumentAttribute(String key, Object value)
```

Parameter

- **key**: Der Schlüssel des Attributs bzw. der Eigenschaft.
  Datentyp: **String**
- **value**: Der Wert des Attributs bzw. der Eigenschaft. Du kannst einen einzelnen Textwert oder eine Liste an Textwerten übergeben.
  Datentyp: **Object**

Eigenschaften

Die Klasse **DocumentAttribute** hat folgende Eigenschaften:

- **key**: Der Schlüssel des Attributs bzw. der Eigenschaft.
  Datentyp: **String**
- **values**: Eine Liste mit Werten der Eigenschaft. Bei Mehrfachwerten enthält die Liste mehr als einen Wert.
  Datentyp: **List<String>**
- **value**: Liest den ersten Wert der Liste **values** aus. Das Festlegen dieser Eigenschaft füllt automatisch die **values**-Eigenschaft mit einer Liste mit dem angegebenen Wert.
  Datentyp: **String**

1.2.6. Verwenden der Klasse "DocumentDownloadResult"

Die Klasse **dvelop_docs_dev.DocumentDownloadResult** bündelt die Ausgabeparameter eines erfolgreichen Downloads eines DMS-Dokuments.

Signatur

```
global with sharing class DocumentDownloadResult
```

Methoden

Die Klasse **DocumentDownloadResult** stellt folgende Methoden zur Verfügung:

- getFilename()
- getBody()

getFilename()

Diese Methode gibt den Namen der heruntergeladenen Originaldatei des DMS-Dokuments zurück.

Signatur

```
global String getFilename()
```

Rückgabe

Der Name der Originaldatei.

getBody()

Diese Methode gibt den Inhalt der heruntergeladenen Originaldatei des DMS-Dokuments zurück.

Signatur

```
global Blob getBody()
```

Rückgabe

Der binäre Inhalt der Originaldatei.

Verwendung

Der folgende Abschnitt zeigt dir einige Anwendungsfälle, die die Verwendung der Klasse darstellen.

Behandeln der Ausgabe eines erfolgreichen Downloads

```
dvelop_docs_dev.DocumentDownloadResult downloadResult =
documentManager.downloadDocument('XH00014562');
String filename = downloadResult.getFilename();
Blob filebody = downloadResult.getBody();
```

1.2.7. Verwenden der Klasse “DocumentEmailOptions”

Die Klasse **dvelop_docs_dev.DocumentEmailOptions** definiert die Parameter, mit denen eine E-Mail mit Salesforce- und DMS-Anhängen versendet werden kann.

Signatur

```
global with sharing class DocumentEmailOptions implements
IDocumentEmailOptions
```

Konstruktoren

Du kannst eine Instanz von **DocumentEmailOptions** mit folgenden Konstruktoren erzeugen:

DocumentEmailOptions()
Der Konstruktor erstellt eine **DocumentEmailOptions**-Instanz ohne eigene Werte.

Signatur

```
global DocumentEmailOptions()
```

DocumentEmailOptions(subject, body, recipients)
Der Konstruktor erstellt eine Instanz von **DocumentEmailOptions** mit Betreff, Inhalt und Empfängern.

Signatur

```
global DocumentEmailOptions(String subject, String body, List<String>
recipients)
```

Parameter

- **subject**: Der Betreff der E-Mail.
  Datentyp: **String**
- **body**: Der Textinhalt der E-Mail.
  Datentyp: **String**
- **recipients**: Eine Liste von E-Mail-Adressen für die Empfänger.
  Datentyp: **List<String>**

Methoden
Die Klasse **DocumentEmailOptions** stellt folgende Methoden zur Verfügung:

setSubject(subject)
Die Methode fügt den entsprechenden Wert als Betreff der E-Mail ein.

Signatur

```
global void setSubject(String subject)
```

Parameter
subject: Der Betreff der E-Mail.

Datentyp: **String**

setBody(body)
Die Methode fügt den entsprechenden Wert als Inhalt der E-Mail ein.

Signatur

```
global void setBody(String body)
```

Parameter
body: Der Inhalt der E-Mail.

Datentyp: **String**

addRecipient(recipient)
Die Methode fügt eine weitere E-Mail-Adresse zur bestehenden Liste von Empfängern hinzu.

Signatur

```
global void addRecipient(String recipient)
```

Parameter
recipient: Die E-Mail-Adresse des neuen Empfängers.

Datentyp: **String**

addRecipients(recipients)
Die Methode fügt mehrere weitere E-Mail-Adressen zur bestehenden Liste von Empfängern hinzu.

Signatur

```
global void addRecipients(List<String> recipients)
```

Parameter
recipients: Eine Liste mit E-Mail-Adressen der neuen Empfänger.

Datentyp: **List<String>**

addCcRecipient(ccRecipient)
Die Methode fügt eine weitere E-Mail-Adresse zur bestehenden Liste von Cc-Empfängern hinzu.

Signatur

```
global void addCcRecipient(String ccRecipient)
```

Parameter
ccRecipient: Die E-Mail-Adresse des neuen Cc-Empfängers.

Datentyp: **String**

addCcRecipients(ccRecipients)
Die Methode fügt mehrere weitere E-Mail-Adressen zur bestehenden Liste von Cc-Empfängern hinzu.

Signatur

```
global void addCcRecipients(List<String> ccRecipients)
```

Parameter
ccRecipients: Eine Liste mit E-Mail-Adressen der neuen Cc-Empfänger.

Datentyp: **List<String>**

addBccRecipient(bccRecipient)
Die Methode fügt eine weitere E-Mail-Adresse zur bestehenden Liste von Bcc-Empfängern hinzu.

Signatur

```
global void addBccRecipient(String bccRecipient)
```

Parameter
bccRecipient: Die E-Mail-Adresse des neuen Bcc-Empfängers.

Datentyp: **String**

addBccRecipients(bccRecipients)
Die Methode fügt mehrere weitere E-Mail-Adressen zur bestehenden Liste von Bcc-Empfängern hinzu.

Signatur

```
global void addBccRecipients(List<String> bccRecipients)
```

Parameter
bccRecipients: Eine Liste mit E-Mail-Adressen der neuen Bcc-Empfänger.

Datentyp: **List<String>**

addContentDocumentId(contentDocumentId)
Die Methode fügt ein **ContentDocument**-Element anhand der ID als E-Mail-Anhang hinzu.

Signatur

```
global void addContentDocumentId(Id contentDocumentId)
```

Parameter
contentDocumentId: Die ID des **ContentDocument**-Elements, das als Anhang der E-Mail versendet werden soll.

Datentyp: **Id**

addContentDocumentIds(contentDocumentIds)
Die Methode fügt mehrere **ContentDocument**-Elemente anhand der jeweiligen IDs als E-Mail-Anhang hinzu.

Signatur

```
global void addContentDocumentIds(List<Id> contentDocumentIds)
```

Parameter
contentDocumentIds: Eine Liste von IDs der **ContentDocument**-Elemente, die als E-Mail-Anhang versendet werden sollen.

Datentyp: **List<Id>**

addDmsDocumentId(dmsDocumentId)
Die Methode fügt ein DMS-Dokument anhand der ID als E-Mail-Anhang hinzu.

Signatur

```
global void addDmsDocumentId(String dmsDocumentId)
```

Parameter
dmsDocumentId: Die ID des DMS-Dokuments, das als Anhang der E-Mail versendet werden soll.

Datentyp: **String**

addDmsDocumentIds(dmsDocumentIds)
Die Methode fügt mehrere DMS-Dokumente anhand der jeweiligen IDs als E-Mail-Anhang hinzu.

Signatur

```
global void addDmsDocumentIds(List<String> dmsDocumentIds)
```

Parameter
dmsDocumentIds: Eine Liste von IDs der DMS Dokumente, die als E-Mail-Anhang versendet werden sollen.

Datentyp: **List<String>**

setRelatedRecordId(recordId)
Die Methode definiert die ID des Datensatzes, der mit der zu versendenden E-Mail verknüpft werden soll.

Signatur

```
global void setRelatedRecordId(String recordId)
```

Parameter
recordId: Die ID des Salesforce-Datensatzes, mit dem die E-Mail verknüpft wird.

Datentyp: **String**

setEmailTemplateId(templateId)
Die Methode definiert die E-Mail-Vorlage für die E-Mail, die versendet werden soll.

Signatur

```
global void setEmailTemplateId(String templateId)
```

Parameter
templateId: Die ID der E-Mail-Vorlage, die angewendet wird.

Datentyp: **String**

setUseSignature(useSignature)
Die Methode bestimmt, ob die Signatur des aktuellen Salesforce-Benutzers zum Inhalt der E-Mail hinzugefügt werden soll.

Signatur

```
global void setUseSignature(Boolean useSignature)
```

Parameter
useSignature: Bestimmt, ob die Signatur hinzugefügt wird.

Datentyp: Boolean

Verwendung

Der folgende Abschnitt zeigt dir einige Anwendungsfälle, die die Verwendung der Klasse darstellen.

Versand einer E-Mail ohne Anhänge

```
dvelop_docs_dev.DocumentEmailOptions emailOptions = new
dvelop_docs_dev.DocumentEmailOptions();
emailOptions.setSubject('Urgent d.velop documents for Salesforce Update');
emailOptions.setBody('Hey, please review the release notes of the latest
version of d.velop documents for Salesforce. Regards.');
emailOptions.addRecipient('astro@salesforce.com');
emailOptions.addCcRecipient('codey@salesforce.com');
emailOptions.addBccRecipient('einstein@salesforce.com');
```

Versand einer E-Mail mit DMS-und Salesforce-Anhängen

```
dvelop_docs_dev.DocumentEmailOptions emailOptions = new
dvelop_docs_dev.DocumentEmailOptions();
emailOptions.setSubject('Urgent d.velop documents for Salesforce Update');
emailOptions.setBody('Hey, please review the release notes of the latest
version of d.velop documents for Salesforce. Regards.');
emailOptions.addRecipient('astro@salesforce.com');
emailOptions.addDmsDocumentId('XH00014562');
emailOptions.addContentDocumentId('069AP0000006hn3YAA');
```

1.2.8. Verwenden der Klasse “DocumentManager”

Die Klasse **dvelop_docs_dev.DocumentManager** bietet verschiedene Funktionen an, um über eigenen Apex-Code mit Dokumenten aus dem angebundenen Dokumentenmanagementsystem (DMS) zu arbeiten und die Dokumente zu verwalten.

Signatur

Verwenden der Klasse "SubscriberInterface"

```
global with sharing class DocumentManager extends SubscriberInterface
```

Konstruktoren

Du kannst eine Instanz von **DocumentManager** mit folgenden Konstruktoren erzeugen:

- ???TITLE???

DocumentManager()

Der Konstruktor erstellt eine **DocumentManager**-Instanz mit Standardwerten.

Signatur

```
global DocumentManager()
```

Methoden

Die Klasse **DocumentManager** bietet folgende Methoden:

- downloadDocument(documentId)
- downloadDocuments(documentIds, finisher, proceedOnFailure)
- downloadDocumentToRecord(recordId, documentId)
- searchDocuments(options)

- search(options)
- searchDocumentsWithIds(searchAttributeKey, documentIds)
- getDocument(documentId)
- updateDocument(documentId, updatedCategory, updatedAttributes)
- deleteDocument(documentId, ?reason)
- sendEmail(options)
- createFolder(category, attributes)
- shareDocuments(downloadUrls)
- shareDocuments(options)
- changeDocumentStatus(documentId, newStatus, ?editorId, ?alterationText)

> **Anmerkung**
> Für die Methoden der Oberklasse, wirf einen Blick auf Verwenden der Klasse "SubscriberInterface".

downloadDocument(documentId)

Die Methode lädt ein DMS-Dokument mit der übergebenen ID herunter und gibt als Ergebnis die Klasse **DocumentDownloadResult** mit Namen und Inhalt der Datei des Dokuments zurück.

> **Anmerkung**
> Eine genaue Anleitung, wie du Dokumente herunterladen kannst, findest du hier.

Signatur

```
global DocumentDownloadResult downloadDocument(String documentId)
```

Parameter

documentId: Die ID des DMS-Dokuments, das nach Salesforce heruntergeladen werden soll.

Datentyp: **String**

Rückgabe

Eine Instanz der Klasse dvelop_docs_dev.DocumentDownloadResult, die den Dateinamen und den Inhalt der Datei des Dokuments enthält.

downloadDocuments(documentIds, finisher, proceedOnFailure)

Die Methode lädt mehrere DMS-Dokumente asynchron herunter und gibt die Ergebnisse an den übergebenen **AsyncProcessFinisher**-Job weiter.

> **Anmerkung**
> Eine genaue Anleitung, wie du Dokumente herunterladen kannst, findest du hier.

> **Warnung**
> Diese Methode startet einen asynchronen **Queueable**-Prozess mit **System.enqueueJob()**. Beachte die Limits für Queueable Jobs in dem Kontext, in dem du die Methode aufrufst.

Signatur

```
global void downloadDocuments(List<String> documentIds,
AsyncProcessFinisher finisher, Boolean proceedOnFailure)
```

Parameter

- **documentIds**: Eine Liste der IDs der DMS-Dokumente, die nach Salesforce heruntergeladen werden sollen.
  Datentyp: **List<String>**
- **finisher**: Eine Instanz eines **AsyncProcessFinisher**-Jobs. Dieser Job wird nach Abschluss der Download-Queue angehängt und erhält eine Liste der Klasse **DocumentDownloadResult** als Output für seinen Standardparameter.
  Datentyp: dvelop_docs_dev.AsyncProcessFinisher
- **proceedOnFailure**: Legt fest, ob der Prozess fortgesetzt wird, wenn beim Herunterladen eines Dokuments ein Fehler auftritt.
  - **Datentyp**: **Boolean**
  - **Standardwert**: **false**

Rückgabe (an den "Finisher"-Job)

Eine Liste mit Instanzen der Klasse dvelop_docs_dev.DocumentDownloadResult, die die Ergebnisse der einzelnen Downloads beinhalten.

downloadDocumentToRecord(recordId, documentId)

Die Methode lädt ein DMS-Dokument mit der übergebenen ID herunter und speichert das Dokument als Datei in einem Salesforce-Datensatz.

> **Anmerkung**
> Eine genaue Anleitung, wie du Dokumente herunterladen kannst, findest du hier.

> **Warnung**
> Diese Methode führt DML aus. Beachte die Limits für DML-Statements in dem Kontext, in dem du die Methode aufrufst. Direkt nach einer DML-Operation sind keine HTTP-Callouts mehr möglich.

Signatur

```
global Id downloadDocumentToRecord(Id recordId, String documentId)
```

Parameter

- **recordId**: Die ID des Datensatzes, an den das heruntergeladene DMS-Dokument angehängt werden soll.
  Datentyp: **Id**
- **documentId**: Die ID des DMS-Dokuments, das nach Salesforce heruntergeladen werden soll.
  Datentyp: **String**

Rückgabe

Die ID des **ContentDocument**-Datensatzes, das für das heruntergeladene DMS-Dokument am verwandten Datensatz erstellt wurde.

searchDocuments(options)

Die Methode führt eine Suche nach Dokumenten im DMS aus. Die Parameter der Suche werden durch die übergebene Instanz von **DocumentSearchOptions** bestimmt.

> **Warnung**
> Die Methode **searchDocuments(options)** ist veraltet, wird aber weiterhin unterstützt. Dennoch empfehlen wir den Umstieg auf die neue Methode search(options), um alle Vorteile nutzen zu können.

Signatur

```
global List<Map<String, String>> searchDocuments(DocumentSearchOptions
options)
```

Parameter

options: Eine Instanz von beliebig konfigurierten Suchoptionen, aus denen die Parameter für die Suche ermittelt werden.

Datentyp: dvelop_docs_dev.DocumentSearchOptions

Rückgabe

Eine Liste von gefundenen Dokumenten in generischem Wörterbuchformat (**Map**).

search(options)

Die Methode führt eine Suche nach Dokumenten im DMS aus. Die Parameter der Suche werden durch die übergebene Instanz von **DocumentSearchOptions** bestimmt.

> **Anmerkung**
> Eine genaue Anleitung, wie du nach DMS-Dokumenten mit komplexen Suchparametern suchen kannst, findest du hier.

Signatur

```
global List<Document> search(DocumentSearchOptions options)
```

Parameter

options: Eine Instanz von beliebig konfigurierten Suchoptionen, aus denen die Parameter für die Suche ermittelt werden.

Datentyp: dvelop_docs_dev.DocumentSearchOptions

Rückgabe

Eine Liste von dvelop_docs_dev.Document für die gefundenen DMS-Dokumente.

searchDocumentsWithIds(searchAttributeKey, documentIds)

Die Methode führt eine Suche nach Dokumenten mit bestimmten IDs im DMS aus.

> **Anmerkung**
> Eine genaue Anleitung, wie du nach DMS-Dokumenten anhand ihrer IDs suchen kannst, findest du hier.

Signatur

```
global List<Document> searchDocumentsWithIds(String searchAttributeKey,
List<String> documentIds)
```

Parameter

- **searchAttributeKey**: Der Schlüssel der Quelleigenschaft, die der Dokument-ID im verbundenen DMS zugewiesen ist.
  Datentyp: **String**
- **documentIds:** Die Liste an IDs der gesuchten Dokumente.
  Datentyp: **List<String>**

Rückgabe
Eine Liste von dvelop_docs_dev.Document für die gefundenen DMS-Dokumente.

getDocument(documentId)
Die Methode ruft ein Dokument mit der angegebenen ID aus dem DMS ab und gibt das Ergebnis als Instanz der Klasse **Document** wieder.

> **Anmerkung**
> Eine genaue Anleitung, wie du ein bestimmtes DMS-Dokument abrufen kannst, findest du hier.

Signatur

```
global Document getDocument(String documentId)
```

Parameter
documentId: Die ID des DMS-Dokuments, dessen Eigenschaften, Links und Kategorien ermittelt werden.

Datentyp: **String**

Rückgabe
Eine Instanz der Klasse dvelop_docs_dev.Document für das DMS-Dokument, das mit der ID am verbundenen DMS gefunden wurde.

updateDocument(documentId, updatedCategory, updatedAttributes)
Die Methode aktualisiert die Eigenschaften und optional die Kategorie des DMS-Dokuments mit der gegebenen ID.

> **Anmerkung**
> Eine genaue Anleitung, wie du ein bestimmtes DMS-Dokument aktualisieren kannst, findest du hier.

Signatur

```
global void updateDocument(String documentId, String updatedCategory,
List<DocumentAttribute> updatedAttributes)
```

Parameter

- **documentId**: Die ID des DMS-Dokuments, das aktualisiert werden soll.
  Datentyp: **String**
- **updatedCategory**: Der Schlüssel der neuen Kategorie. Wenn die Kategorie nicht aktualisiert werden soll, kann stattdessen **null** übergeben werden.
  Datentyp: **String**
- **updatedAttributes**: Eine Liste von Eigenschaften, die aktualisiert werden sollen.

Datentyp: **List**<dvelop_docs_dev.DocumentAttribute>

deleteDocument(documentId, ?reason)

Die Methode löscht ein Dokument oder eine Akte mit der gegebenen ID aus dem DMS.

> **Anmerkung**
> Eine genaue Anleitung, wie du ein bestimmtes DMS-Objekt löschen kannst, findest du hier.

Signatur

```
global void deleteDocument(String documentId)
```

```
global void deleteDocument(String documentId, String reason)
```

Parameter

- **documentId**: Die ID des DMS-Objekts, das gelöscht werden soll.
  Datentyp: **String**
- **reason**: Der Grund für die Löschung des Objekts. Die Zeichenkette muss zwischen drei und 80 Zeichen lang sein.
  Datentyp: **String**
  Standardwert: "Von d.velop documents for Salesforce gelöscht."

sendEmail(options)

Die Methode versendet eine E-Mail mit Anhängen aus Salesforce (**ContentDocument**) und dem verbundenen DMS (DMS-Dokumente). Die Details der E-Mail werden durch die übergebene Instanz von **DocumentEmailOptions** bestimmt.

> **Anmerkung**
> Eine genaue Anleitung, wie du E-Mails mit Salesforce- und DMS-Anhängen versendest, findest du hier.

Signatur

```
global void sendEmail(DocumentEmailOptions options)
```

Parameter

options: Eine Instanz von beliebig konfigurierten E-Mail-Optionen, aus denen die nötigen E-Mail-Details und Anhänge ermittelt werden.

Datentyp: dvelop_docs_dev.DocumentEmailOptions

createFolder(category, attributes)

Die Methode erstellt eine Akte mit der angegebenen Kategorie und der Liste an Eigenschaften im DMS.

> **Anmerkung**
> Eine genaue Anleitung, wie du Akten im DMS erstellst, findest du hier.

Signatur

```
global String createFolder(String category, List<DocumentAttribute>
attributes)
```

Parameter

- **category**: Der Schlüssel der Kategorie, die die Akte haben soll.
  Datentyp: **String**
- **attributes:** Eine Liste von Eigenschaften, die für die Akte festgelegt werden sollen.
  Datentyp: **List<**dvelop_docs_dev.DocumentAttribute**>**

Rückgabe
Die ID der erstellten Akte im DMS.

shareDocuments(downloadUrls)
Die Methode erzeugt temporäre URLs, um DMS-Dokumente mit Dritten zu teilen.

> **Anmerkung**
> Eine genaue Anleitung, wie du DMS-Dokumente mit Dritten teilst, findest du hier.

> **Warnung**
> Die Methode steht in SharePoint-Umgebungen nicht zur Verfügung.

Signatur

```
global Map<String, String> shareDocuments(Set<String> downloadUrls)
```

Parameter
downloadUrls: Eine Menge an URLs zum Herunterladen der Dokumente, die geteilt werden sollen.

Datentyp: **Set<String>**

Rückgabe
Ein generisches Wörterbuch (Map) mit den übergebenen URLs zum Herunterladen und den dazugehörigen URLs zum Teilen.

shareDocuments(options)
Erzeugt temporäre URLs mit benutzerdefinierten Einstellungen, um DMS-Dokumente mit Dritten zu teilen.

Signatur

```
global Map<String, String> shareDocuments(DocumentShareOptions options)
```

Parameter
options: Eine Sammlung an Einstellungen, die die erzeugten URLs zum Teilen betreffen.

Datentyp: dvelop_docs_dev.DocumentShareOptions

Rückgabe
Ein generisches Wörterbuch (Map) mit den übertragenen URLs zum Herunterladen und den dazugehörigen URLs zum Teilen, basierend auf den übertragenen Einstellungen.

changeDocumentStatus(documentId, newStatus, ?editorId, ?alterationText)
Aktualisiert den Status und optional die bearbeitende Person eines DMS-Dokuments.

Signatur

```
global void changeDocumentStatus(String documentId, DocumentStatus
newStatus)
```

```
global void changeDocumentStatus(String documentId, DocumentStatus
newStatus, String editorId, String alterationText)
```

Parameter

- **documentId**: Die ID des DMS-Dokuments, das aktualisiert werden soll.
  Datentyp: **String**
- **newStatus**: Der neue Status des Dokuments.
  Datentyp: dvelop_docs_dev.DocumentStatus
- **editorId**: Die ID des Identity Provider-Benutzers, der für die Bearbeitung des Dokuments eingetragen werden soll.
  Datentyp: **String**
- **alterationText**: Ein kurzer Text, der die Änderung begründet oder beschreibt.
  Datentyp: **String**

1.2.9. Verwenden der Klasse “DocumentSearchOptions”

Die Klasse **dvelop_docs_dev.DocumentSearchOptions** definiert die Parameter, mit denen Dokumente aus dem angebundenen Dokumentenmanagementsystem (DMS) abgerufen werden.

Signatur

```
global with sharing class DocumentSearchOptions implements
IDocumentSearchOptions
```

Konstruktoren

Du kannst eine Instanz von **DocumentSearchOptions** mit folgenden Konstruktoren erzeugen:

- DocumentSearchOptions()

DocumentSearchOptions()

Der Konstruktor erstellt eine **DocumentSearchOptions**-Instanz mit Standardwerten.

Signatur

```
global DocumentSearchOptions()
```

Methoden

Die Klasse **DocumentSearchOptions** bietet folgende Methoden:

- buildOptions()
- useRecordContext(recordId)
- useObjectContext(objectApiName)
- useObjectContext(objectApiName, recordTypeId)
- addSearchAttribute(attributeKey, value)
- addSearchAttribute(attributeKey, values)
- addSearchAttributes(searchAttributes)
- ignoreSearchAttributes()
- addSearchCategory(categoryKey)
- addSearchCategories(categoryKeys)
- ignoreSearchCategories()

- hideFolders()
- showFolders()
- sortByAttribute(attributeKey)
- sortAscending()
- sortDescending()
- setSearchText(searchText)
- setMaxAgeInDays(maxAgeInDays)
- setPageIndex(pageIndex)
- setPageSize(pageSize)

buildOptions()

Die Methode erzeugt eine Instanz der Klasse **DocumentSearchOptions.Builder**.

> **Anmerkung**
> Eine genaue Anleitung, wie du die Klasse **DocumentSearchOptions.Builder** verwendest, findest du hier.

Signatur

```
global static DocumentSearchOptions.Builder buildOptions()
```

Rückgabe

Eine neue Instanz von dvelop_docs_dev.DocumentSearchOptions.Builder.

useRecordContext(recordId)

Die Methode ändert den Kontext der Suche für den Datensatz mit der übergebenen ID.

Alle Einstellungen wie Zuordnungen (Mappings), Dokumentartzuordnungen, Eigenschaftszuordnungen und Sortierungseinstellungen werden anschließend anhand der Objekttypen und des Datensatztypen des übergebenen Datensatzes ermittelt.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende zuvor verwendete Methoden unwirksam:
>
> - **useObjectContext(objectApiName)**
> - **useObjectContext(objectApiName, recordTypeId)**
>
> Verwende immer nur eine der verfügbaren Methoden.

Signatur

```
global void useRecordContext(Id recordId)
```

Parameter

recordId: Die ID des Datensatzes, der als Kontext für die Suche verwendet werden soll.

Datentyp: **Id**

useObjectContext(objectApiName)

Die Methode ändert den Kontext der Suche auf das Salesforce-Objekt mit dem übergebenen API-Namen.

Alle Einstellungen wie Zuordnungen (Mappings), Dokumentartzuordnungen, Eigenschaftszuordnungen und Sortierungseinstellungen werden anschließend anhand des übergebenen Objekts und des Standard-datensatztypen **Master** ermittelt.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende zuvor verwendete Methoden unwirksam:
>
> - **useObjectContext(objectApiName)**
> - **useObjectContext(objectApiName, recordTypeId)**
>
> Verwende immer nur eine der verfügbaren Methoden.

Signatur

```
global void useObjectContext(String objectApiName)
```

Parameter

objectApiName: Der API-Name des Salesforce-Objekts, das als Kontext für die Suche verwendet werden soll.

Datentyp: **String**

useObjectContext(objectApiName, recordTypeId)

Die Methode ändert den Kontext der Suche auf das Salesforce-Objekt mit dem übergebenen API-Namen und den Datensatztypen mit der übergebenen ID.

Alle Einstellungen wie Zuordnungen (Mappings), Dokumentartzuordnungen, Eigenschaftszuordnungen und Sortierungseinstellungen werden anschließend anhand des übergebenen Objekts und des übergebenen Datensatztypen ermittelt.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende zuvor verwendete Methoden unwirksam:
>
> - **useObjectContext(objectApiName)**
> - **useObjectContext(objectApiName, recordTypeId)**
>
> Verwende immer nur eine der verfügbaren Methoden.

Signatur

```
global void useObjectContext(String objectApiName, String recordTypeId)
```

Parameter

- **objectApiName**: Der API-Name des Salesforce-Objekts, das als Kontext für die Suche verwendet werden soll.
  Datentyp: **String**
- **recordTypeId**: Die ID des Datensatztypen, der als Kontext für die Suche verwendet werden soll.
  Datentyp: **String**

addSearchAttribute(attributeKey, value)

Die Methode fügt eine benutzerdefinierte Sucheigenschaft mit einem entsprechenden Wert zu den bestehenden Sucheigenschaften hinzu.

Du kannst beliebig viele Eigenschaften und Werte für die Suche definieren, die das Ergebnis der Suche auf Dokumente mit passenden Eigenschaften einschränken.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Mapping**
> - **Eigenschaften > Zuordnungen**

Signatur

```
global void addSearchAttribute(String attributeKey, String value)
```

Parameter

- **attributeKey**: Der Schlüssel der Eigenschaft, die zur Suche verwendet werden soll. Eine vollständige Übersicht über die verfügbaren Schlüssel findest du unter **Eigenschaften > Zuordnung** in deiner d.velop documents-Konfiguration.
  Datentyp: **String**
- **value**: Der Wert der Eigenschaft, der mit den Eigenschaften der Dokumente abgeglichen wird.
  Datentyp: **String**

addSearchAttribute(attributeKey, values)

Die Methode fügt eine benutzerdefinierte Sucheigenschaft mit einer Liste von entsprechenden Werten zu den bestehenden Sucheigenschaften hinzu.

Du kannst beliebig viele Eigenschaften und Werte für die Suche definieren, die das Ergebnis der Suche auf Dokumente mit passenden Eigenschaften einschränken.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Mapping**
> - **Eigenschaften > Zuordnungen**

Signatur

```
global void addSearchAttribute(String attributeKey, List<String> values)
```

Parameter

- **attributeKey**: Der Schlüssel der Eigenschaft, die zur Suche verwendet werden soll. Eine vollständige Übersicht über die verfügbaren Schlüssel findest du unter **Eigenschaften > Zuordnung** in deiner d.velop documents-Konfiguration.
  Datentyp: **String**
- **values**: Eine Liste von möglichen Werten der Eigenschaft, die mit den Eigenschaften der Dokumente abgeglichen wird. Die Werte werden mit **ODER** verknüpft. Wenn ein Wert übereinstimmt, wird ein Dokument gefunden.
  Datentyp: **List<String>**

addSearchAttributes(searchAttributes)

Die Methode fügt mehrere Sucheigenschaften und -werte zu den bestehenden Sucheigenschaften hinzu.

Du kannst beliebig viele Eigenschaften und Werte für die Suche definieren, die das Ergebnis der Suche auf Dokumente mit passenden Eigenschaften einschränken.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Mapping**
> - **Eigenschaften > Zuordnungen**

Signatur

```
global void addSearchAttributes(Map<String, String> searchAttributes)
```

Parameter

searchAttributes: Eine Sammlung von Schlüssel-Wert-Paaren für Sucheigenschaften.

Datentyp: **Map<String, String>**

ignoreSearchAttributes()

Die Methode ignoriert alle globalen Sucheigenschaften und alle benutzerdefinierten Sucheigenschaften, die du zuvor mithilfe der Methoden **addSearchAttribute** oder **addSearchAttributes** eingefügt hast.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Mapping**
> - **Eigenschaften > Zuordnungen**
>
> Wenn du diese Methode verwendest, wird der Effekt folgender Methoden unabhängig von der Reihenfolge der Verwendung überschrieben:
>
> - **addSearchAttribute**
> - **addSearchAttributes**

Signatur

```
global void ignoreSearchAttributes()
```

addSearchCategory(categoryKey)

Die Methode fügt eine benutzerdefinierte Suchkategorie zu den bestehenden Suchkategorien hinzu.

Du kannst beliebig viele Suchkategorien definieren, die das Ergebnis der Suche auf Dokumente mit passenden Kategorien einschränken.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Dokumentarten > Zuordnung**
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen**

Signatur

```
global void addSearchCategory(String categoryKey)
```

Parameter

categoryKey: Der Schlüssel der Kategorie, die für die Suche verwendet werden soll. Eine vollständige Übersicht über die verfügbaren Schlüssel findest du unter **Dokumentarten > Zuordnung** in deiner d.velop documents-Konfiguration.

Datentyp: **String**

addSearchCategories(categoryKeys)

Die Methode fügt mehrere Suchkategorien zu den bestehenden Suchkategorien hinzu.

Du kannst beliebig viele Suchkategorien definieren, die das Ergebnis der Suche auf Dokumente mit passenden Kategorien einschränken.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Dokumentarten > Zuordnungen**
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen**

Signatur

```
global void addSearchCategories(List<String> categoryKeys)
```

Parameter

categoryKeys: Eine Liste von Schlüsseln für Suchkategorien.

Datentyp: **List<String>**

ignoreSearchCategories()

Die Methode ignoriert alle globalen Suchkategorien und alle benutzerdefinierten Suchkategorien, die du zuvor mithilfe der Methoden **addSearchCategory** oder **addSearchCategories** eingefügt hast.

> **Anmerkung**
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Dokumentarten > Zuordnungen**
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen**
>
> Wenn du diese Methode verwendest, wird der Effekt folgender Methoden unabhängig von der Reihenfolge der Verwendung überschrieben:
>
> - **addSearchCategory**
> - **addSearchCategories**

Signatur

```
global void ignoreSearchCategories()
```

hideFolders()

Die Methode blendet Akten in den Suchergebnissen aus.

> **Anmerkung**
>
> Wenn du diese Methode verwendest, werden folgende globale Einstellungen überschrieben:
>
> - **Dokumentenliste > Allgemeine Einstellungen > Ordner anzeigen**
>
> Diese Methode hat ausschließlich einen Effekt, wenn mit einer Suche auch tatsächlich Akten ermittelt werden können. Es gelten folgende Voraussetzungen:
>
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen** ist deaktiviert.
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen** ist aktiviert und mit **Dokumentarten > Zuordnungen** eingefügte Suchkategorien enthalten Akten.
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen** ist aktiviert und mit den Methoden **addSearchCategory** oder **addSearchCategories** eingefügte Suchkategorien enthalten Akten.

Signatur

```
global void hideFolders()
```

showFolders()

Die Methode zeigt Akten in den Suchergebnissen an.

> **Anmerkung**
>
> Der Aufruf dieser Methode überschreibt folgende globale Einstellungen:
>
> - **Dokumentenliste > Allgemeine Einstellungen > Ordner anzeigen**
>
> Diese Methode hat ausschließlich einen Effekt, wenn mit einer Suche auch tatsächlich Akten ermittelt werden können. Es gelten folgende Voraussetzungen:
>
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen** ist deaktiviert.
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen** ist aktiviert und mit **Dokumentarten > Zuordnungen** eingefügte Suchkategorien enthalten Akten.
> - **Dokumentarten > Allgemeine Einstellungen > Nur ausgewählte Dokumentarten suchen** ist aktiviert und mit den Methoden **addSearchCategory** oder **addSearchCategories** eingefügte Suchkategorien enthalten Akten.

Signatur

```
global void showFolders()
```

sortByAttribute(attributeKey)

Die Methode fügt die Eigenschaft mit dem übergebenen Schlüssel als Sortiereigenschaft ein. Das Ergebnis der Suche wird anschließend anhand der Werte der angegebenen Eigenschaft sortiert. Die Sortierung ist standardmäßig aufsteigend.

Signatur

```
global void sortByAttribute(String attributeKey)
```

Parameter

attributeKey: Der Schlüssel der Eigenschaft, die für die Sortierung der Suchergebnisse verwendet werden soll.

Datentyp: **String**

sortAscending()

Die Methode sortiert die Suchergebnisse in aufsteigender Reihenfolge. Die Sortierung erfolgt anhand der mit der Methode **sortByAttribute** definierten Sortiereigenschaft.

> **Anmerkung**
> Diese Methode hat nur einen Effekt, wenn du zusätzlich eine Sortiereigenschaft mithilfe der Methode **sortByAttribute** festgelegt hast.

Signatur

```
global void sortAscending()
```

sortDescending()

Die Methode sortiert die Suchergebnisse in absteigender Reihenfolge. Die Sortierung erfolgt anhand der mit der Methode **sortByAttribute** definierten Sortiereigenschaft.

> **Anmerkung**
> Diese Methode hat ausschließlich einen Effekt, wenn du zusätzlich eine Sortiereigenschaft mithilfe der Methode **sortByAttribute** festgelegt hast.

Signatur

```
global void sortDescending()
```

setSearchText(searchText)

Die Methode sucht in allen Dokumenten zusätzlich via Volltextsuche nach dem übergebenen Suchbegriff und schränkt die Ergebnisse auf passende Dokumente ein.

Signatur

```
global void setSearchText(String searchText)
```

Parameter

searchText: Der Suchbegriff für die Volltextsuche.

Datentyp: **String**

setMaxAgeInDays(maxAgeInDays)

Die Methode setzt ein maximales Alter (in Tagen) für die Suche ein. Alle Dokumente, die älter als die angegebene Anzahl an Tagen sind, werden in der Suche nicht berücksichtigt.

Signatur

```
global void setMaxAgeInDays(Integer maxAgeInDays)
```

Parameter
maxAgeInDays: Das maximale Alter der Dokumente in Tagen.

Datentyp: **Integer**

setPageIndex(pageIndex)
Die Methode setzt die Seitennummer der Dokumente im Suchergebnis ein. Die Anzahl der verfügbaren Seiten wird durch die Anzahl aller verfügbaren Dokumente und der mit der Methode **setPageSize** definierten Seitengröße bestimmt.

Signatur

```
global void setPageIndex(Integer pageIndex)
```

Parameter
pageIndex: Die Nummer bzw. der Index der Seite. Der Startindex ist 1.

- **Datentyp**: **Integer**
- **Standardwert**: 1

setPageSize(pageSize)
Die Methode fügt die Seitengröße ein und bestimmt damit die maximale Anzahl der Dokumente im Suchergebnis.

Signatur

```
global void setPageSize(Integer pageSize)
```

Parameter
pageSize: Die maximale Anzahl der Dokumente pro Seite.

- **Datentyp**: **Integer**
- **Standardwert**: 100

Verwendung
Der folgende Abschnitt zeigt dir einige Anwendungsfälle, die die richtige Verwendung der Klasse darstellen.

Ermitteln aller Dokumente an einem Datensatz (mit globalen Einstellungen)

```
dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useRecordContext('001AP000003aVJhYAM');
```

Ermitteln aller Dokumente an Accounts (ohne globale Einstellungen)

```
dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useObjectContext('Account');
searchOptions.ignoreSearchAttributes();
searchOptions.ignoreSearchCategories();
```

Ermitteln aller Dokumente an Accounts mit passender Eigenschaft "accountnumber", Dokumentart "Schriftverkehr Kunde" oder "Unbestimmte Dokumentart", ohne Akten

```
dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useObjectContext('Account');
```

```
searchOptions.addSearchAttribute('accountnumber', '001AP000003aVJhYAM');
searchOptions.addSearchCategory('Schriftverkehr_Kunde');
searchOptions.addSearchCategory('Unbestimmte_Dokumentart');
searchOptions.hideFolders();
```

Komplexer Anwendungsfall

```
dvelop_docs_dev.DocumentSearchOptions searchOptions = new
dvelop_docs_dev.DocumentSearchOptions();
searchOptions.useRecordContext('001AP000003aVJhYAM');

searchOptions.addSearchCategories(new List<String>{
'Schriftverkehr_Kunde','Eingangsrechnung', 'Unbestimmte_Dokumentart' });
searchOptions.addSearchAttributes(
    new Map<String, String>{
        'accountnumber', '001AP000003aVJhYAM',
        'objecttitle' => 'Testaccount',
        'published' => 'Ja'
    }
);

searchOptions.sortByAttribute('createddate');
searchOptions.sortAscending();

searchOptions.setSearchText('Produkt');
searchOptions.setMaxAgeInDays(1);
searchOptions.setPageIndex(1);
searchOptions.setPageSize(500);
```

1.2.10. Verwenden der Klasse "DocumentSearchOptions.Builder"

Die Klasse **dvelop_docs_dev.DocumentSearchOptions.Builder** erleichtert die Erstellung von Suchparametern durch die Implementierung von verkettbaren Methoden.

Grundlagen

Das Builder-Entwurfsmuster ist ein häufig verwendetes Muster der Softwareentwicklung, um komplexe Datentypen mithilfe einfacher, kompakter Befehle und Methoden zu erstellen. Mit einem Builder (dt. "Erbauer") kannst du unter anderem Methodenaufrufe verketten und so dein Produkt – also eine Instanz des komplexen Datentypen – in einem einzigen Statement erstellen.

Die Klasse **DocumentSearchOptions.Builder** implementiert das Builder-Entwurfsmuster. Die Klasse kannst du über die Methode **buildOptions()** der Oberklasse erstellen. Der Builder kann mithilfe von beliebig vielen verketteten Methoden eine Instanz von **DocumentSearchOptions** konfigurieren.

Du erhältst das Ergebnis ("Produkt") des Builders anschließend über einen Aufruf von **getResult()**.

Beispiel

Folge nachstehendem Beispiel, um mit dem Builder eine Instanz von **DocumentSearchOptions** zu erstellen:

```
dvelop_docs_dev.DocumentSearchOptions searchOptions =
    dvelop_docs_dev.DocumentSearchOptions.buildOptions()    // Create an
instance of the builder
    .withRecordContext('001AP000003aVJhYAM')                // Configure
options - here: record context
    ...                                                     // Configure
further options if necessary
```

```
    .getResult();                                                    // Get
the result - A completely configured instance of
dvelop_docs_dev.DocumentSearchOptions
```

Methoden

Die Klasse **DocumentSearchOptions.Builder** bietet folgende Methoden als entsprechende Gegenstücke zu Methoden der Basisklasse:

Methode (Builder)	Methode(n) (Basisklasse)
withRecordContext(recordId)	useRecordContext(recordId)
withObjectContext(objectApiName)	useObjectContext(objectApiName)
withObjectContext(objectApiName, recordTypeId)	useObjectContext(objectApiName, recordTypeId)
withSearchAttribute(attributeKey, value)	addSearchAttribute(attributeKey, value)
withSearchAttribute(attributeKey, values)	addSearchAttribute(attributeKey, values)
withoutSearchAttributes()	ignoreSearchAttributes()
withSearchCategory(categoryKey)	addSearchCategory(categoryKey)
withSearchCategories(categoryKeys)	addSearchCategories(categoryKeys)
withoutSearchCategories()	ignoreSearchCategories()
withFolders()	showFolders()
withoutFolders()	hideFolders()
sortedAscendingBy(sortAttribute)	sortByAttribute(attributeKey)
	sortAscending()
sortedDescendingBy(sortAttribute)	sortByAttribute(attributeKey)
	sortDescending()
withSearchText(searchText)	setSearchText(searchText)
withMaxAgeInDays(maxAgeInDays)	setMaxAgeInDays(maxAgeInDays)
withPageIndex(pageIndex)	setPageIndex(pageIndex)
withPageSize(pageSize)	setPageSize(pageSize)

Verwendung

Hier erfährst du, wie du den Builder verwendest, um die Anwendungsfälle aus DocumentSearchOptions > Verwendung umzusetzen.

Ermitteln aller Dokumente an einem Datensatz (mit globalen Einstellungen)

```
dvelop_docs_dev.DocumentSearchOptions searchOptions =
    dvelop_docs_dev.DocumentSearchOptions.buildOptions()
    .withRecordContext('001AP000003aVJhYAM')
    .getResult();
```

Ermitteln aller Dokumente an Accounts (ohne globale Einstellungen)

```
dvelop_docs_dev.DocumentSearchOptions searchOptions =
    dvelop_docs_dev.DocumentSearchOptions.buildOptions()
    .withObjectContext('Account')
    .withoutSearchAttributes()
    .withoutSearchCategories()
    .getResult();
```

Ermitteln aller Dokumente an Accounts mit passender Eigenschaft "accountnumber", Dokumentart "Schriftverkehr Kunde" oder "Unbestimmte Dokumentart", ohne Akten

```
dvelop_docs_dev.DocumentSearchOptions searchOptions =
    dvelop_docs_dev.DocumentSearchOptions.buildOptions()
    .withObjectContext('Account')
```

```
    .withSearchAttribute('accountnumber', '001AP000003aVJhYAM')
    .withSearchCategory('Schriftverkehr_Kunde')
    .withSearchCategory('Unbestimmte_Dokumentart')
    .withoutFolders()
    .getResult();
```

Komplexer Anwendungsfall

```
dvelop_docs_dev.DocumentSearchOptions searchOptions =
dvelop_docs_dev.DocumentSearchOptions
    .buildOptions()
    .withRecordContext('001AP000003aVJhYAM')
    .withSearchCategories(new List{ 'Schriftverkehr_Kunde',
'Eingangsrechnung', 'Unbestimmte_Dokumentart' })
    .withSearchAttribute('accountnumber', '001AP000003aVJhYAM')
    .withSearchAttribute('objecttitle', 'Testaccount')
    .withSearchAttribute('published', 'Ja')
    .sortedAscendingBy('createddate')
    .withSearchText('Produkt')
    .withMaxAgeInDays(1)
    .withPageIndex(1)
    .withPageSize(500)
    .getResult();
```

1.2.11. Verwenden der Klasse "DocumentShareOptions"

Die Klasse **dvelop_docs_dev.DocumentShareOptions** bündelt die Parameter zum Teilen von DMS-Dokumenten und Einstellungen zur Gültigkeit der erzeugten URLs.

Signatur

```
global with sharing class DocumentShareOptions
```

Konstruktoren

Die Klasse **DocumentShareOptions** hat folgende Konstruktoren:

- DocumentShareOptions()

DocumentShareOptions()

Erzeugt ein neues **DocumentShareOptions**-Objekt.

Signatur

```
global DocumentShareOptions()
```

Methoden

Die Klasse **DocumentShareOptions** bietet folgende Methoden:

- addEntry(downloadUrl, validUntil, allowMultiUse)

addEntry(downloadUrl, validUntil, allowMultiUse)

Fügt einen neuen Eintrag zum Teilen hinzu. Der Eintrag enthält eine URL, von der das Dokument heruntergeladen werden kann, einen Zeitstempel zur Gültigkeit und eine Angabe zur Mehrfachverwendung.

Signatur

```
global void addEntry(String downloadUrl, Datetime validUntil, Boolean
allowMultiUse)
```

Parameter

- **downloadUrl**: Eine URL, von der das Dokument, das geteilt werden soll, heruntergeladen werden kann.
  Datentyp: **String**
- **validUntil**: Ein Zeitstempel, der die maximale Gültigkeit der erzeugten URL zum Teilen bestimmt. Der Stempel muss in der Zukunft liegen und nicht mehr als 180 Tage betragen.
  Datentyp: **Datetime**
- **allowMultiUse**: Gibt an, ob die erzeuge URL zum Teilen mehrmals aufgerufen werden darf.
  Datentyp: **Boolean**

1.2.12. Verwenden des Enums "DocumentStatus"

Das Enum **dvelop_docs_dev.DocumentStatus** repräsentiert die verschiedenen Status, in denen sich ein DMS-Dokument befinden kann.

Werte

- **PROCESSING**: Das Dokument befindet sich aktuell in Bearbeitung.
- **VERIFICATION**: Das Dokument befindet sich aktuell in Prüfung.
- **RELEASE**: Das Dokument ist veröffentlicht und einsehbar.

1.2.13. Verwenden der Klasse “DocumentUploader”

Verwende verschiedene Funktionen der Klasse **dvelop_docs_dev.DocumentUploader**, um mit eigenem Apex-Code Salesforce-Dateien (**ContentDocument**, **Attachment** und **EmailMessage**) in das angebundene Dokumentenmanagementsystem (DMS) hochzuladen.

Signatur

Verwenden der Klasse "SubscriberInterface"

```
global with sharing class DocumentUploader extends SubscriberInterface
```

Konstruktoren

Du kannst eine Instanz von **DocumentUploader** mit folgenden Konstruktoren erzeugen:

- DocumentUploader()

DocumentUploader()

Der Konstruktor erstellt eine **DocumentUploader**-Instanz mit Standardwerten.

Signatur

```
global DocumentUploader()
```

Methoden

Die Klasse **DocumentUploader** stellt folgende Methoden zur Verfügung:

- getAttributesFromConfig(recordId, doctypeKey, filename)
- startUpload(requests, ?options)

> **Anmerkung**
> Weitere Informationen zu den Methoden der Oberklasse findest du unter Verwenden der Klasse "SubscriberInterface".

getAttributesFromConfig(recordId, doctypeKey, filename)

Verwende diese Methode, um die Eigenschaften für ein Dokument, das du hochladen möchtest, anhand der global konfigurierten Einstellungen zu berechnen. Die Einstellungen werden durch den Objekttypen der übergebenen Datensatz-ID und die Dokumentart bestimmt.

Signatur

```
global List<DocumentAttribute> getAttributesFromConfig(Id recordId, String
doctypeKey, String filename)
```

Parameter

- **recordId**: Die ID des Datensatzes. Die ID bestimmt über die globalen Einstellungen in Verbindung mit der Dokumentart die berechneten Eigenschaften und deren Quelle (Datensatz).
  Datentyp: **Id**
- **doctypeKey**: Der Schlüssel der Dokumentart, die die berechneten Eigenschaften anhand der globalen Einstellungen vorgibt.
  Datentyp: **String**
- **filename**: Ein optionaler Dateiname, der als Quellwert für Objektzuordnungen des Typs **Dateiname zur Laufzeit** verwendet wird. Wenn der Dateiname irrelevant ist, wird **null** übergeben.
  Datentyp: **String**

startUpload(requests, ?options)

Verwende diese Methode, um verschiedene Datensätze (z.B. **ContentDocument**, **Attachment** oder **EmailMessage**) in d.velop documents hochzuladen. Definiere die Parameter für das Hochladen mithilfe verschiedener Typen von Upload-Anfragen ("Requests").

> **Warnung**
> Diese Methode startet einen asynchronen **Queueable**-Prozess mit **System.enqueueJob()**. Beachte die Limits für Queueable Jobs in dem Kontext, in dem du die Methode verwendest.

Signatur

```
global Id startUpload(List<IDocumentUploadRequest> requests)
```

```
global Id startUpload(List<IDocumentUploadRequest> requests,
DocumentUploadOptions options)
```

Parameter

- **requests**: Eine Liste an Anfragen zum Hochladen, die die Rahmenparameter für das Hochladen definieren. Je nach Typ der Anfrage werden entweder einzelne oder mehrere Dateien hochgeladen.
  Datentyp: **List<dvelop_docs_dev.IDocumentUploadRequest>**
- **options**: Eine optionale Sammlung von Einstellungen, mit denen du das Verhalten des Hochladeprozesses beeinflussen kannst.
  Datentyp: dvelop_docs_dev.DocumentUploadOptions

Rückgabe

Die ID des eingereihten **Queueable**-Jobs des Prozesses.

1.2.14. Verwenden der Klasse "DocumentUploadOptions"

Die Klasse **dvelop_docs_dev.DocumentUploadOptions** definiert Einstellungen und Optionen, die das Verhalten des asynchronen Prozesses zum Hochladen von Dateien steuern.

Signatur

```
global with sharing class DocumentUploadOptions
```

Konstruktoren

Du kannst **DocumentUploadOptions** mit folgenden Konstruktoren erstellen:

- DocumentUploadOptions(useConfigUser, skipValidate)

DocumentUploadOptions(useConfigUser, skipValidate)

Der Konstruktor erstellt **DocumentUploadOptions** mit den verfügbaren Parametern.

Signatur

```
global DocumentUploadOptions(Boolean useConfigUser, Boolean skipValidate)
```

Parameter

- **useConfigUser**: Legt fest, ob der Prozess mit den Zugangsdaten des Servicebenutzers oder den Zugangsdaten des aktuell angemeldeten Benutzers ausgeführt wird.
  Datentyp: **Boolean**
- **skipValidate**: Legt fest, ob eine aus dem Cache abgerufene Benutzersitzung noch einmal mit **Identity-provider** validiert und u.U. neu angefordert wird.
  Datentyp: **Boolean**

1.2.15. Verwenden der Klasse "EmailMessageUploadRequest"

Die Klasse **dvelop_docs_dev.EmailMessageUploadRequest** definiert die Parameter, mit denen ein **Email-Message**-Objekt aus Salesforce in das DMS hochgeladen werden soll.

Signatur

Verwenden des Interfaces "IDocumentUploadRequest"

```
global class EmailMessageUploadRequest implements IDocumentUploadRequest
```

Konstruktoren

Du kannst eine Klasse **EmailMessageUploadRequest** mit folgendem Konstruktor erstellen:

- EmailMessageUploadRequest()

EmailMessageUploadRequest()

Der Konstruktor erstellt **EmailMessageUploadRequest** ohne Parameter.

Signatur

```
global EmailMessageUploadRequest()
```

Eigenschaften

Die Klasse **EmailMessageUploadRequest** hat folgende Eigenschaften:

- **relatedEntityId**: Die ID eines Datensatzes, in dessen Kontext die Emails hochgeladen werden sollen.
  Datentyp: **String**
- **emailMessageIds**: Eine Liste mit IDs der **EmailMessage**-Objekte, die hochgeladen werden sollen.
  Datentyp: **List<String>**
- **documenttypeKey**: Der Schlüssel der Kategorie, mit der die Dokumente im DMS gespeichert werden sollen.
  Datentyp: **String**

- **useConfigUserForUpload**: Legt fest, ob der Prozess mit den Zugangsdaten des Servicebenutzers oder den Zugangsdaten des aktuell angemeldeten Benutzers ausgeführt wird.

> **Warnung**
> Die Eigenschaft **useConfigUserForUpload** ist veraltet und wird nur noch in Flows unterstützt. Verwende stattdessen die Klasse DocumentUploadOptions, um den Hochladeprozess mit anderen Zugangsdaten auszuführen.

Datentyp: **Boolean**
Standardwert: **false**

1.2.16. Verwenden des Interfaces "IDocumentUploadRequest"

Verwende das Interface **dvelop_docs_dev.IDocumentUploadRequest** für die Implementierung verschiedener Hochladeanfragen, um unterschiedlichste Anwendungsfälle für das Hochladen von Dateien oder E-Mails aus Salesforce zu realisieren.

In Verbindung mit der Klasse **DocumentUploader** kannst du eine Liste unter **IDocumentUploadRequest** definieren. Anschließend wird die Liste in einem asynchronen Prozess Anfrage für Anfrage abgearbeitet, wobei beliebig viele Elemente (**ContentDocuments**, **Attachments**, **EmailMessages**) hochgeladen werden.

Aktuell werden nur die bereits in der Anwendung enthaltenen Implementierungen des Interfaces für den Prozess unterstützt. Eigene Implementierungen werden ignoriert.

Signatur

```
global interface IDocumentUploadRequest
```

Wird implementiert von:

- Verwenden der Klasse "AttachmentMigrationRequest"
- Verwenden der Klasse "ContentDocumentUploadRequest"
- Verwenden der Klasse "EmailMessageUploadRequest"
- Verwenden der Klasse "UploadNewVersionRequest"

1.2.17. Verwenden der Klasse "RecordValueProvider"

Die Klasse **dvelop_docs_dev.RecordValueProvider** ist eine abstrakte Klasse, die einen Mechanismus zur dynamischen Bereitstellung von Datensatzwerten zur Verfügung stellt.

Diese Klasse implementiert das interne Interface **dvelop_docs_dev.IDynamicValueProvider** und kann von anderen Klassen erweitert werden, um spezifische Implementierungen zum Abruf von Datensatzwerten zu erstellen.

Signatur

```
global with sharing abstract class RecordValueProvider implements
IDynamicValueProvider
```

Methoden

Die Klasse **RecordValueProvider** bietet folgende Methoden:

- provide(input)

provide(input)

Stellt dynamisch Datensatzwerte und Metadaten basierend auf dem gegebenen Input bereit.

Signatur

```
global abstract RecordValueProviderOutput provide(RecordValueProviderInput
input)
```

Rückgabe

Eine Instanz der Ausgabe-Klasse dvelop_docs_dev.RecordValueProviderOutput, die die berechneten Werte und Metadaten enthält.

Verwendung

Der folgende Abschnitt zeigt dir beispielhaft, wie du die Klasse **RecordValueProvider** für deine eigenen dynamischen Werte bereitstellen kannst:

Beispiel

```
global class CustomRecordValueProvider extends
dvelop_docs_dev.RecordValueProvider {

    global override dvelop_docs_dev.RecordValueProviderOutput
provide(dvelop_docs_dev.RecordValueProviderInput input) {
        Id recordId = input.getRecordId();

        List<System.SelectOption> selectOptions = new
List<System.SelectOption>{
            new System.SelectOption('valueA', 'Label A'),
            new System.SelectOption('valueB', 'Label B'),
            new System.SelectOption(recordId, 'Record ID')
        };

        return
dvelop_docs_dev.RecordValueProviderOutput.picklist(selectOptions);
    }

}
```

1.2.18. Verwenden der Klasse "RecordValueProviderInput"

Die Klasse **dvelop_docs_dev.RecordValueProviderInput** stellt Eingabewerte für die dynamische Werte- und Metadatenermittlung mithilfe einer eigenen Implementierung der Klasse dvelop_docs_dev.RecordValueProvider bereit.

Signatur

```
global with sharing class RecordValueProviderInput implements
IDynamicValueProviderInput
```

Methoden

Die Klasse **RecordValueProviderInput** bietet folgende Methoden:

- getRecordId()

getRecordId()

Gibt die ID des Kontextdatensatzes des Prozesses aus.

Ein Kontextdatensatz ist ein Datensatz, auf dem ein Prozess mit dynamischer Werte- und Metadatenermittlung gestartet wurde, z.B. ein Hochladevorgang, bei dem dynamisch Auswahl- bzw. Vorbelegungswerte der Dokumenteigenschaften ermittelt werden sollen.

Signatur

```
global Id getRecordId()
```

Rückgabe

Die ID des Kontextdatensatzes. Wenn ein Prozess keinen Kontextdatensatz hat, wird **null** zurückgegeben.

1.2.19. Verwenden der Klasse "RecordValueProviderOutput"

Die Klasse **dvelop_docs_dev.RecordValueProviderOuput** wird verwendet, um Ausgabewerte für den Prozess einer dynamischen Werte- und Metadatenermittlung zu übertragen.

Signatur

```
global with sharing class RecordValueProviderOutput implements
IDynamicValueProviderOutput
```

Methoden

Die Klasse **RecordValueProviderOutput** bietet folgende Methoden:

- picklist(selectOptions, defaultValue)
- multipicklist(selectOptions, defaultValues)
- of(displayType, value)

picklist(selectOptions, defaultValue)

Fabrikmethode, die eine Klasseninstanz für den Datentyp **Schema.DisplayType.PICKLIST** aus einer Liste (**System.SelectOption**) erzeugt, um eine Auswahlliste darzustellen.

Signatur

```
global static RecordValueProviderOutput picklist(List<System.SelectOption>
selectOptions, String defaultValue)
```

Parameter

- **selectOptions**: Eine Liste an Auswahloptionen für die dargestellte Auswahlliste.
  Datentyp: **List<System.SelectOption>**
- **defaultValue**: Ein optionaler Wert zur Vorbelegung. Wenn kein Wert vorbelegt werden soll, kann **null** übertragen werden.
  Datentyp: **String**

Rückgabe

Eine Klasseninstanz für den Datentyp **Schema.DisplayType.PICKLIST**.

multipicklist(selectOptions, defaultValues)

Fabrikmethode, die eine Klasseninstanz für den Datentyp **Schema.DisplayType.MULTIPICKLIST** aus einer Liste an **System.SelectOption** erzeugt, um eine Mehrfach-Auswahlliste darzustellen.

Signatur

```
global static RecordValueProviderOutput
multipicklist(List<System.SelectOption> selectOptions, List<String>
defaultValues)
```

Parameter

- **selectOptions**: Eine Liste an Auswahloptionen für die dargestellte Mehrfach-Auswahlliste.
  Datentyp: **List<System.SelectOption>**
- **defaultValues**: Eine optionale Liste an vorbelegten Werten. Wenn keine Werte vorbelegt werden sollen, kann **null** übertragen werden.
  Datentyp: **List<String>**

Rückgabe

Eine Klasseninstanz für den Datentyp **Schema.DisplayType.MULTIPICKLIST**.

of(displayType, value)

Fabrikmethode, die eine Klasseninstanz für einen beliebigen Datentyp und Wert erzeugt.

Für bestimmte Datentypen muss der passende Datentyp im Parameter **value** übertragen werden. In diesem Fall empfehlen wir den direkten Aufruf der spezialisierte Methode. Generell gilt dieses Vorgehen für Datentypen, für die eine spezialisierte Fabrikmethode existiert, z.B. **Schema.DisplayType.PICKLIST**.

Signatur

```
global static RecordValueProviderOutput of(Schema.DisplayType displayType,
Object value)
```

Parameter

- **displayType**: Der Datentyp, mit dem der ermittelte Wert im Ablagedialog dargestellt werden soll.
  Datentyp: **Schema.DisplayType**
- **value**: Der dynamisch ermittelte Wert für die Suche oder Ablage.
  Datentyp: **Object**

Rückgabe

Eine Klasseninstanz für den übertragenen Datentyp und den ermittelten Wert.

1.2.20. Verwenden der Klasse “OpenAPI”

Die Klasse **dvelop_docs_dev.OpenAPI** bietet verschiedene Funktionen, die die Arbeit mit DMS-Dokumenten in Apex vereinfachen.

Signatur

```
global class OpenAPI
```

Methoden

Die Klasse **OpenAPI** stellt folgende Methoden zur Verfügung:

salesforceDatetimeToDMSDatetimeString(salesforceDatetime)

Wandelt ein Apex-Datetime-Objekt in eine Zeichenkette um. Die Zeichenkette wird vom angebundenen DMS unterstützt. Eigenschaftsfelder in d.velop documents benötigen unter Umständen ein konkretes

Format für Datumsangaben. In diesem Fall kann die Methode ein bestehendes **salesforceDatetime**-Objekt in den notwendigen String umwandeln.

Signatur

```
global static String salesforceDatetimeToDMSDatetimeString(Datetime
salesforceDatetime)
```

Parameter

- **salesforceDatetime**: Ein **salesforceDatetime**-Objekt in Salesforce, das in eine Zeichenkette umgewandelt werden soll. Die Zeichenkette wird im angebundenen DMS unterstützt.
  Datentyp: **Datetime**

Rückgabe

Eine Zeichenkette, die von Eigenschaften des Typs **Datum** im DMS akzeptiert wird.

getValidCookieForCurrentUser()

Liefert einen gültigen Cookie für den angemeldeten Benutzer.

Signatur

```
global static String getValidCookieForCurrentUser()
```

Rückgabe

Der Cookie des angemeldeten Benutzers.

getValidCookieForConfigUser()

Liefert einen gültigen Cookie für den Servicebenutzer, der in der d.velop documents-Konfiguration gespeichert ist.

Signatur

```
global static String getValidCookieForConfigUser()
```

Rückgabe

Der Cookie des Servicebenutzers aus der d.velop documents-Konfiguration.

generateCredentials()

Meldet den angemeldeten Benutzer beim zugrundeliegenden DMS an. Die Methode speichert Cookies für die weitere Verwendung.

Wir empfehlen, diese Methode am Anfang einer Operationskette zu verwenden und alle folgenden Operationen beispielsweise asynchron in einer Queue abzuarbeiten. So stellst du sicher, dass immer gültige Cookies zur Authentifizierung vorhanden sind.

Signatur

```
global static void generateCredentials()
```

generateConfigUserCredentials()

Meldet den in der d.velop documents-Konfiguration angegebenen Servicebenutzer beim zugrundeliegenden DMS an. Die Methode speichert Cookies für die weitere Verwendung.

Wir empfehlen, diese Methode am Anfang einer Operationskette zu verwenden und alle folgenden Operationen beispielsweise asynchron in einer Queue abzuarbeiten. So stellst du sicher, dass immer gültige Cookies zur Authentifizierung vorhanden sind.

Signatur

```
global static void generateConfigUserCredentials()
```

initializeTestEnvironment()
Erstellt die minimal notwendigen Daten und HTTP-Mocks für Unit-Tests.

Signatur

```
global static void initializeTestEnvironment()
```

createStub(parentType, stubProvider)
Erstellt einen Stub des gegebenen **parentType**-Typs mithilfe der Stub API im **dvelop_docs_dev**-Namespace.

Signatur

```
global static Object createStub(System.Type parentType, System.StubProvider
stubProvider)
```

Parameter

- **parentType**: Der Typ des Stubs, der erstellt werden soll.
  Datentyp: **System.Type**
- **stubProvider**: Die Provider-Klasse, die die Methodenaufrufe an der erstellten Stub-Instanz bearbeiten soll.
  Datentyp: **System.StubProvider**

Rückgabe
Die erstellte Stub-Instanz zur Verwendung in Tests.

1.2.21. Verwenden der Klasse "SubscriberInterface"
Die abstrakte Klasse **dvelop_docs_dev.SubscriberInterface** dient als zentrale Oberklasse für alle Service-Schnittstellen, die dir für das Hochladen und Verwalten von Dokumenten in Apex zur Verfügung stehen.

Die Klasse bietet vor allem einfache Methoden zur Authentifizierung, um erweiternden Unterklassen gültige Benutzersitzungen hinzuzufügen und sicherzustellen, dass alle ausgehenden HTTP-Anfragen die nötige Autorisierung erhalten.

Signatur

```
global abstract with sharing class SubscriberInterface
```

Wird erweitert von:

- Verwenden der Klasse “DocumentManager”
- Verwenden der Klasse “DocumentUploader”

Methoden
Die Klasse **SubscriberInterface** bietet folgende Methoden:

- authenticateInCurrentUserContext()
- authenticateInServiceUserContext()
- setAuthentication(cookie)
- isAuthenticated()

authenticateInCurrentUserContext()

Authentifiziert das Service-Interface im Kontext des angemeldeten Benutzers, indem es die Benutzersitzung mit **setAuthentication(cookie)** übergibt. Die Sitzungsinformationen werden nach einer erfolgreichen Authentifizierung in d.velop documents mittels **Salesforce Plattform Cache** zwischengespeichert und bei Bedarf wieder aus diesem Speicher abgerufen.

Signatur

```
global void authenticateInCurrentUserContext()
```

authenticateInServiceUserContext()

Authentifiziert das Service-Interface im Kontext des Servicebenutzers aus der d.velop documents-Konfiguration, indem es die Benutzersitzung mit **setAuthentication(cookie)** übergibt. Die Sitzungsinformationen werden nach einer erfolgreichen Authentifizierung in d.velop documents mittels **Salesforce Plattform Cache** zwischengespeichert und bei Bedarf wieder aus diesem Speicher abgerufen.

Signatur

```
global void authenticateInServiceUserContext()
```

setAuthentication(cookie)

Übergibt eine Benutzersitzung in Form eines Cookies an das Service-Interface.

Die genaue Implementierung hängt von dem konkreten Service-Interface ab.

Signatur

```
global abstract void setAuthentication(String cookie)
```

Parameter

cookie: Cookie einer gültigen Benutzersitzung, um das Service-Interface zu autorisieren.

Datentyp: **String**

isAuthenticated()

Gibt den Status der Authentifizierung des Service-Interfaces zurück.

Die genaue Implementierung hängt von dem konkreten Service-Interface ab.

Signatur

```
global abstract Boolean isAuthenticated()
```

Rückgabe

Der Status der Authentifizierung (**true** / **false**).

1.2.22. Verwenden der Klasse "UploadNewVersionRequest"

Die Klasse **dvelop_docs_dev.UploadNewVersionRequest** definiert die Parameter, mit denen eine neue Version eines DMS-Dokuments hochgeladen werden soll.

Signatur

Verwenden des Interfaces "IDocumentUploadRequest"

```
global class UploadNewVersionRequest implements IDocumentUploadRequest
```

Konstruktoren

Du kannst die Klasse **UploadNewVersionRequest** mit folgenden Konstruktoren erstellen:

UploadNewVersionRequest()

Der Konstruktor erstellt die Klasse **UploadNewVersionRequest** ohne Parameter.

Signatur

```
global UploadNewVersionRequest()
```

UploadNewVersionRequest(contentDocumentId, dmsDocumentId, useConfigUserForUpdate)

Der Konstruktor erstellt die Klasse **UploadNewVersionRequest** mit allen nötigen Parametern.

Signatur

```
global UploadNewVersionRequest(Id contentDocumentId, String dmsDocumentId,
Boolean useConfigUserForUpdate)
```

Parameter

- **contentDocumentId**: Die ID der **ContentDocument**-Datei, die als neue Version hochgeladen werden soll.
  Datentyp: **String**
- **dmsDocumentId**: Die ID des DMS-Dokuments, das die neue Version erhalten soll.
  Datentyp: **String**
- **useConfigUserForUpdate**: Legt fest, ob der Prozess mit den Zugangsdaten des Servicebenutzers oder den Zugangsdaten des aktuell angemeldeten Benutzers ausgeführt wird.

  > **Warnung**
  > Der Parameter **useConfigUserForUpdate** ist veraltet und wird nur noch in Flows unterstützt. Verwende stattdessen die Klasse **DocumentUploader.Options**, um den Hochladeprozess mit anderen Zugangsdaten auszuführen.

  Datentyp: **Boolean**
  Standardwert: **false**

Eigenschaften

Die Klasse **UploadNewVersionRequest** hat folgende Eigenschaften:

- **contentDocumentId**: Die ID der **ContentDocument**-Datei, die als neue Version hochgeladen werden soll.
  Datentyp: **String**
- **dmsDocumentId**: Die ID des DMS-Dokuments, das die neue Version erhalten soll.
  Datentyp: **String**
- **useConfigUserForUpload**: Legt fest, ob der Prozess mit den Zugangsdaten des Servicebenutzers oder den Zugangsdaten des aktuell angemeldeten Benutzers ausgeführt wird.

  > **Warnung**
  > Die Eigenschaft **useConfigUserForUpdate** ist veraltet und wird nur noch in Flows unterstützt. Verwende stattdessen die Klasse DocumentUploadOptions, um den Hochladeprozess mit anderen Zugangsdaten auszuführen.

  Datentyp: **Boolean**
  Standardwert: **false**

1.3. Weitere Informationsquellen und Impressum

Wenn Sie Ihre Kenntnisse rund um die d.velop-Software vertiefen möchten, besuchen Sie die digitale Lernplattform der d.velop academy unter https://dvelopacademy.keelearning.de/.

Mithilfe der E-Learning-Module können Sie sich in Ihrem eigenen Tempo weiterführende Kenntnisse und Fachkompetenz aneignen. Zahlreiche E-Learning-Module stehen Ihnen ohne vorherige Anmeldung frei zugänglich zur Verfügung.

Besuchen Sie unsere Knowledge Base im d.velop service portal. In der Knowledge Base finden Sie die neusten Lösungen, Antworten auf häufig gestellte Fragen und How To-Themen für spezielle Aufgaben. Sie finden die Knowledge Base unter folgender Adresse: https://kb.d-velop.de/

Das zentrale Impressum finden Sie unter https://www.d-velop.de/impressum.