

salesforce

---

# Salesforce Knowledge Developer Guide

Version 36.0, Spring '16



 @salesforcedocs

Last updated: April 27, 2016



# CONTENTS

<b>Chapter 1: Introduction to Developing with Salesforce Knowledge</b> .....	1
Salesforce Knowledge API Objects .....	2
<b>Chapter 2: Setting Up a Development Organization</b> .....	7
<b>Chapter 3: Building a Search Page with Visualforce Tags</b> .....	11
Create an Article List with Visualforce .....	12
Add Pagination to the Visualforce Article List .....	14
Add a Search Field to the Visualforce Article Search page .....	17
Add a Filter to the Visualforce Article Search Page .....	20
Add a Last Updated Promotions Box to the Visualforce Article Search Page .....	32
<b>Chapter 4: Managing Articles Using Apex and REST APIs</b> .....	36
PublishingService Class .....	38
PublishingService Methods .....	38
Archive the Master Version of an Article .....	49
Assign a Task Related to a Translation .....	50
Assign a Task Related to a Master Article .....	51
Delete a Master Version of an Article .....	51
Delete Translated Version of an Article .....	52
Edit an Online Version of a Master Article .....	52
Publish the Master Version of an Article .....	53
Restore an Archived Version of an Article .....	54
Retrieve Article Metadata .....	54
Retrieve a Version of an Article .....	55
Search for Metadata Elements of a Master Version .....	55
Search for Metadata Elements of a Translated Version .....	56
Set a Translated Article Version to Complete .....	56
Set a Translated Article Version to Incomplete .....	57
Submit an Article for Translation .....	57
Unpublish the Master Version of an Article .....	58
Unpublish the Online Version of a Translated Article .....	58
Update View Statistics with the Salesforce Object Query Language .....	59
Track Search Keywords with the Salesforce Object Search Language .....	59
Update Article View Statistics with the Salesforce Object Search Language .....	60
<b>Index</b> .....	61



# CHAPTER 1 Introduction to Developing with Salesforce Knowledge

## In this chapter ...

- [Salesforce Knowledge API Objects](#)

Build your knowledge base and give your website visitors, clients, partners, and service agents the ultimate in support. Salesforce Knowledge lets you create and manage your company information and securely share it when and where it is needed.

Your Salesforce Knowledge base is built from knowledge articles, which are documents of information. Articles can include information on process, like how to reset your product to its defaults, or frequently asked questions like, how much storage your product supports.

Experienced service agents and internal writers write the articles. The articles are then published to a range of channels: internal database, customer and partner communities, or public websites. Where and what information is published is based on the article layout profile and the field level security.

This guide is for developers who are responsible for customizing Salesforce Knowledge according to their company's needs. It provides several examples to help you understand and build [an article search page using Visualforce tags](#).

# Salesforce Knowledge API Objects

---

Salesforce Knowledge uses articles and data categories. *Articles* capture information about your company's products and services that you want to make available in your knowledge base. In Salesforce Knowledge, *data categories* are a set of criteria organized hierarchically into category groups. Articles in the knowledge base can be classified according to multiple categories that make it easy for users to find the articles they need. For example, to classify articles by sales regions and business units, create two category groups, Sales Regions and Business Units. The Sales Regions category group could consist of a geographical hierarchy, such as All Sales Regions as the top level, North America, Europe, and Asia at the second level, and so on up to five levels. Authors assign categories to articles. Administrators can use data categories to control access to articles.

Data categories are administrator-defined values that help organize articles into logical subgroups within a knowledge base. For example, from the Articles tab you can filter a list of articles by selecting the most relevant data categories. This section lists Salesforce Knowledge API objects for articles and data categories and resources for working with them.

## Articles

*Articles* capture information about your company's products and services that you want to make available in your knowledge base. Articles in the knowledge base can be classified by using one or more data categories to make it easy for users to find the articles they need. Administrators can use data categories to control access to articles.

Articles are based on article types, which rely on:

- Article-type layouts to organize the content in sections.
- Article-type templates to render articles.

Every article is managed in a publishing cycle.

### Article Type

All articles in Salesforce Knowledge are assigned to an *article type*. An article's type determines the type of content it contains, its appearance, and which users can access it. For example, a simple FAQ article type might have two custom fields, `QUESTION` and `ANSWER`, where article managers enter data when creating or updating FAQ articles. A more complex article type may require dozens of fields organized into several sections. Using layouts and templates, administrators can structure the article type in the most effective way for its particular content. User access to article types is controlled by permissions. For each article type, an administrator can grant "Create," "Read," "Edit," or "Delete" permissions to users. For example, the article manager might want to allow internal users to read, create, and edit FAQ article types, but let partner users only read FAQs.

### Article-Type Layout

An *article-type layout* enables administrators to create sections that organize the fields on an article, as well as choose which fields users can view and edit. One layout is available per article type. Administrators modify the layout from the article-type detail page.

### Article-Type Template

An *article-type template* specifies how the sections in the article-type layout are rendered. An article type can have a different template for each of its four channels. For example, if the Customer Portal channel on the FAQ article-type is assigned to the Tab template, the sections in the FAQ's layout appear as tabs when customers view an FAQ article. For the Table of Contents template, the sections defined in the layout appear on a single page (with hyperlinks) when the article is viewed. Salesforce provides two standard article-type templates, Tab and Table of Contents. Custom templates can be created with Visualforce.

### Channel

A channel refers to the medium by which an article is available. Salesforce Knowledge offers four channels where you can make articles available.

- Internal App: Salesforce users can access articles in the Articles tab depending on their role visibility.

- **Customer:** Customers can access articles if the Articles tab is available in a community or Customer Portal. Customer users inherit the role visibility of the manager on the account. In a community, the article is only available to users with Customer Community or Customer Community Plus licenses.
- **Partner:** Partners can access articles if the Articles tab is available in a community or partner portal. Partner users inherit the role visibility of the manager on the account. In a community, the article is only available to users with Partner Community licenses.
- **Public Knowledge Base:** Articles can be made available to anonymous users by creating a public knowledge base using the *Sample Public Knowledge Base for Salesforce Knowledge* app from the AppExchange. Creating a public knowledge base requires Sites and Visualforce.
- **Your own website.** Articles can be made available to users through your company website.

### Publishing Cycle

Salesforce Knowledge Articles move through a publishing cycle from their creation to their deletion. The publishing cycle includes three different statuses: `Draft` is the stage when a new article is being created or an existing one is being updated. Articles with the `Online` status are draft articles that have been published and are now available to their different channels. Eventually, when a published article is at the end of its life, it can be moved to the `Archived` status or sent back to `Draft` to be updated in a subsequent version.

The following table lists API resources for working with articles.

Name	Type	Description
<b>Article</b> <i>Type__DataCategorySelection</i>	Object	A data category selection represents a data category that classifies an article.
<b>Article Type__ka</b>	Object	Gives access to an article from a specific article type independent of its version. This object is read-only and can't be used in a SOQL clause or in a WITH DATA CATEGORY <i>DataCategorySpec</i> SOSL clause. For more information, see <code>KnowledgeArticle</code> .
<b>Article Type__kav</b>	Object	Gives access to all articles from a specific article type depending on their version. This object gives access to the fields available in <code>KnowledgeArticleVersion</code> . For more information, see <code>KnowledgeArticleVersion</code> .
<b>Article Type__Feed</b>	Object	Represents a single feed item in the feed displayed on the detail page for an article.
<b>Article Type__ViewStat</b>	Object	Provides statistics on the number of views for an article from a specific article type. For more information, see <code>KnowledgeArticleViewStat</code> .
<b>Article Type__VoteStat</b>	Object	Provides the weighted rating for an article from a specific article type on a scale of 1 to 5. For more information, see <code>KnowledgeArticleVoteStat</code> .
<code>CaseArticle</code>	Object	Represents the association between a Case and a <code>KnowledgeArticle</code> .
<code>FeedComment</code>	Object	Represents a comment added to a feed by a user. Represents a comment added to a feed by a user.
<code>FeedItem</code>	Object	<code>FeedItem</code> represents an entry in the feed, such as changes in a record feed, including text posts, link posts, and content posts.

Name	Type	Description
KnowledgeArticle	Object	Gives access to an article independent of its version. This object is read-only and can't be used in a SOQL clause or in a WITH DATA CATEGORY <i>DataCategorySpec</i> SOSL clause.
KnowledgeArticleVersion	Object	Provides a global view of standard article fields across all article types depending on their version.
KnowledgeArticleViewStat	Object	Provides statistics on the number of views for the specified article across all article types.
KnowledgeArticleVoteStat	Object	Provides the weighted rating for the specified article on a scale of 1 to 5 across all article types.
NewsFeed	Object	Represents a single feed item on a user's home tab. A Chatter feed shows recent changes to records that the user is following.
UserProfileFeed	Object	Represents a user profile feed, which tracks all actions by a user on records that can be tracked in a feed. This feed is displayed on the user profile page.
WITH DATA CATEGORY <b><i>filteringExpression</i></b>	SOQL clause	Filters articles depending on their status in the publishing cycle and their data categories. For more information, see the <a href="#">Salesforce SOQL and SOSL Reference Guide</a> .
WITH DATA CATEGORY <b><i>DataCategorySpec</i></b>	SOSL clause	Finds articles based on their categorization. For more information, see the <a href="#">Salesforce SOQL and SOSL Reference Guide</a> .

## SOQL and SOSL with KnowledgeArticleVersion

- Always filter on a single value of `PublishStatus` unless the query filters on one or more primary key IDs. To support security, only users with the "Manage Articles" permission see articles whose `PublishStatus` value is `Draft`.
- Archived article versions are stored in the `articletype_kav` object. To query archived article versions, specify the article `Id` and set `sLatestVersion='0'`.
- Always filter on a single value of `Language`. However, in SOQL, you can filter on more than one `Language` if there is a filter on `Id` or `KnowledgeArticleId`.

## Data Categories

Data categories are organized by category group and let:

- Users classify and find records.
- Administrators control access to records.

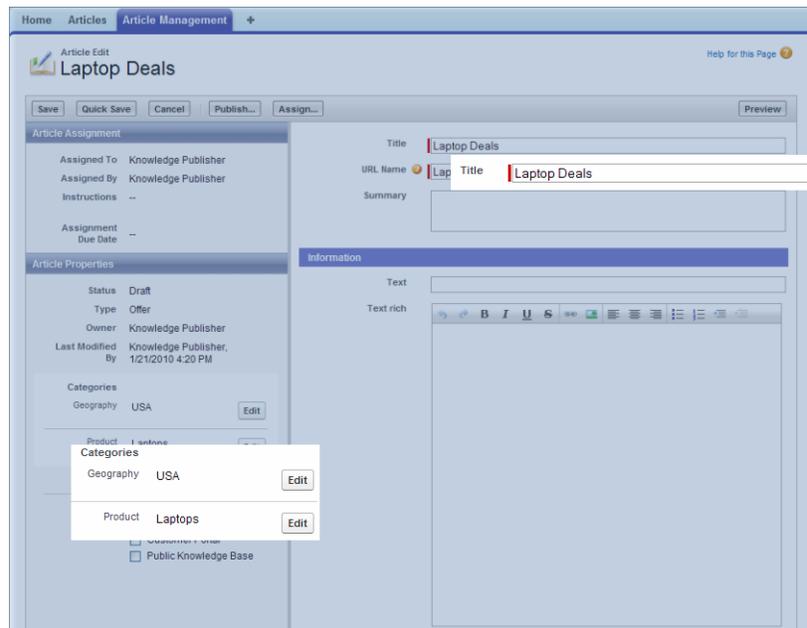
Data categories can be used by Salesforce Knowledge (articles) and answers communities (questions).

### Data Categories and Articles

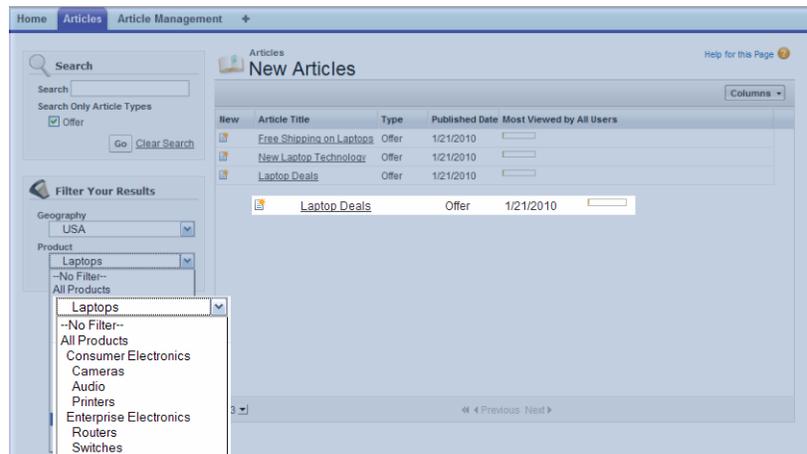
Salesforce Knowledge supports a five-level hierarchy of data categories within each category group. You can classify articles in the knowledge base according to multiple categories that make it easy for users to find the articles they need. For example, to classify articles by sales regions and business units, create two category groups, Sales Regions and Business Units. The Sales Regions category

group could consist of a geographical hierarchy, such as All Sales Regions as the top level, North America, Europe, and Asia at the second level.

The figure below shows a knowledge base administrator's view of an article about laptop deals. Using the article edit page, the administrator has classified the article with `Laptops` in the `Product` category group, and `USA` in the `Geography` category group.



The next figure illustrates an agent finding that same article published on the Articles tab. The agent selects `Laptops` and `USA` in the `Products` and `Geography` drop-down lists respectively.



The following table lists API resources for working with data categories.

Name	Type	Description
<b>Article</b>	Object	Gives access to article categorization.
<b>Type__DataCategorySelection</b>		

Name	Type	Description
QuestionDataCategorySelection	Object	Gives access to question categorization.
WITH DATA CATEGORY <b><i>filteringExpression</i></b>	SOQL clause	Filters articles depending on their status in the publishing cycle and their data categories. For more information, see the <a href="#">Salesforce SOQL and SOSL Reference Guide</a> .
WITH DATA CATEGORY <b><i>DataCategorySpec</i></b>	SOSL clause	Finds articles based on their categorization. For more information, see the <a href="#">Salesforce SOQL and SOSL Reference Guide</a> .
describeDataCategoryGroups ()	Call	Retrieves available category groups for objects specified in the request.
describeDataCategoryGroupStructures ()	Call	Retrieves available category groups along with their data category structure for objects specified in the request.
describeDataCategoryGroups	Apex method	Returns a list of the category groups associated with the specified objects. See the <a href="#">Apex Developer Guide</a> .
describeDataCategoryGroupStructures	Apex method	Returns available category groups along with their data category structure for objects specified in the request. See the <a href="#">Apex Developer Guide</a> .

# CHAPTER 2 Setting Up a Development Organization

The samples provided with this guide refer to specific category groups, categories, and article types. To use these samples without altering them, you must create a development organization with the following parameters for [article types](#), [data categories](#), [articles](#) and a [custom app](#). You can also update the samples with the categories and article types from your development organization.

## Article Types

This table describes two article types used with the samples. To use the samples without altering them, add these article types to your development organization. For more information on how to add article types, see *Create Article Types* in the Salesforce online help.

Parameter	Offer	Promotion
Label	Offer	Promotion
API Name	Offer__kav	Promotion__kav
Custom Field 1	<ul style="list-style-type: none"><li>Field Label: Reference</li><li>API Name: Reference__c</li><li>Type: text</li></ul>	<ul style="list-style-type: none"><li>Field Label: Start Date</li><li>API Name: Start_Date__c</li><li>Type: date/time</li></ul>
Custom Field 2	<ul style="list-style-type: none"><li>Field Label: Description</li><li>API Name: Description__c</li><li>Type: rich text area</li></ul>	<ul style="list-style-type: none"><li>Field Label: End Date</li><li>API Name: End_Date__c</li><li>Type: date/time</li></ul>
Custom Field 3	<ul style="list-style-type: none"><li>Field Label: Launch Date</li><li>API Name: Launch_Date__c</li><li>Type: date</li></ul>	<ul style="list-style-type: none"><li>Field Label: Description</li><li>API Name: Description__c</li><li>Type: rich text area</li></ul>
Custom Field 4	<ul style="list-style-type: none"><li>Field Label: Size</li><li>API Name: Size__c</li><li>Type: picklist</li><li>Values:<ul style="list-style-type: none"><li>XS</li><li>S</li><li>M</li><li>L</li><li>XL</li><li>XXL</li></ul></li></ul>	None

Parameter	Offer	Promotion
Custom Field 5	<ul style="list-style-type: none"> <li>• Field Label: Data Sheet</li> <li>• API Name: Data_Sheet__c</li> <li>• Type: file</li> </ul>	None

## Data Categories

The customized search page provided in the code samples relies on the Fashions, Stores, and Products category groups. This table details the categories used in these category groups. The API lets you select a category by its unique name. When adding these categories to your sample organization, ensure that you set the category unique name as shown in parentheses.

Fashions	Stores	Products
All (All)	All (All)	All (All)
<ul style="list-style-type: none"> <li>• Rockers (Rockers)</li> <li>• Late 50's/early 60's (Late_50_s_early_60_s)</li> <li>• Futuristic (Futuristic)</li> <li>• Grunge (Grunge)</li> <li>• Old Hollywood (Old_Hollywood)</li> <li>• The 80's (The_80_s)</li> <li>• French Chic (French_Chic)</li> <li>• Flashy Disco (Flashy_Disco)</li> <li>• Gothic (Gothic)</li> </ul>	<ul style="list-style-type: none"> <li>• Online store (Online_Store)</li> <li>• US (US)                             <ul style="list-style-type: none"> <li>- San Francisco (San_Francisco)</li> <li>- New-York (New_York)</li> </ul> </li> <li>• Asia (Asia)                             <ul style="list-style-type: none"> <li>- Hong-Kong (Hong_Kong)</li> <li>- Tokyo (Tokyo)</li> </ul> </li> <li>• Europe (Europe)                             <ul style="list-style-type: none"> <li>- Paris (Paris)</li> <li>- London (London)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Men (Men)                             <ul style="list-style-type: none"> <li>- Shoes (Shoes_m)</li> <li>- Jackets (Jackets_m)</li> <li>- Trousers (Trousers_m)</li> <li>- Shirts (Shirts_m)</li> <li>- Hats (Hats_m)</li> </ul> </li> <li>• Women (Women)                             <ul style="list-style-type: none"> <li>- Shoes (Shoes_w)</li> <li>- Jackets (Jackets_w)</li> <li>- Trousers (Trousers_w)</li> <li>- Skirts (Shirts_w)</li> <li>- Hats (Hats_w)</li> </ul> </li> <li>• Children (Children)                             <ul style="list-style-type: none"> <li>- Shoes (Shoes_k)</li> <li>- Trousers (Trousers_k)</li> <li>- Jackets (Jackets_k)</li> </ul> </li> </ul>

## Articles

You can create articles to make the sample search page more realistic. Create `.csv` files with the content provided in this section to import into your sample organization.

Follow these steps to create a `.csv` file for each of the Offer and Promotion article types. For information on how to import articles, see *Import Existing Information into Salesforce Knowledge* in the Salesforce online help.

1. Type the article type field names, separated by commas, into the first line of the `.csv` file. Do not add spaces between the fields.
2. Cut and paste the value content below the field names.
3. Save the file and import the `.csv` file into Salesforce Knowledge.



**Note:** You must publish the articles after import. Non-published articles do not display on the custom search pages.

### Offer Articles

The following content contains articles with the Offer article type. The bulleted list has the available fields. The following lines are the field values.

- Title
- Summary
- URLName
- datacategorygroup.Fashions
- datacategorygroup.Stores
- datacategorygroup.Products
- channels
- Reference\_\_c
- Description\_\_c
- Launched\_Date\_\_c
- Size\_\_c
- Data\_Sheet\_\_c

```
Aviator Jacket,5
Zippers,,The_80_s+Grunge,San_Francisco+Paris+London,Jackets_m,,,,,
Army Jacket,Men's Grenade jacket for
snowboarding,,Grunge,US,Jackets_m,,,,,
Futuristic Boots,Integrates state-of-the-art
biotechnics,,Futuristic,All,Shoes_w,,,,,
60s Mini Skirt,Go back with this mini
skirt,,Late_50_s_early_60_s,All,Skirts_w,,,,,
Dotted 80's Skirt,Who said you were a material
girl?,,The_80_s,All,Skirts_w,,,,,
80's Punk Skirts,London Calling!,,The_80_s+Grunge,US,Skirts_w,,,,,
French Design shoes,Magnifique!,,French_Chic,All,Shoes_w,,526398,,,,
Vintage Platform Shoes,Get high!,,The_80_s,All,Shoes_w,,,,,
Metallic Jacket,We need you for vintage
army!,,The_80_s+Futuristic,All,Jackets_m,,,,,
Gothic Boots,Are you from the Adam's family?,,Gothic,All,Shoes_w,,,,,
Motorcycle Jacket,Live to ride, ride to
live!,,Rockers,US,Jackets_m,,,,,
Men's Leather Pants,Hey
man!,,Late_50_s_early_60_s,US+Europe,Trousers_m,,,,,
Retro Disco Jacket,Saturday's night
```

```
fever!,,Flashy_Disco,US+Europe,Jackets_m,,,,,  
French_Beret,Red  
beret,,French_Chic,All,Hats_m+Hats_w,,,Beret_DataSheet.htm,2010-02-25,S,
```

### Promotion Articles

The following content contains articles with the Promotion article type. The bulleted list has the available fields. The following lines are the field values.

- Title
- URLName
- datacategorygroup.Fashions
- datacategorygroup.Stores
- datacategorygroup.Products
- channels
- Start\_Date\_\_c
- End\_Date\_\_c,Description\_\_c

```
60's_Pants,,Late_50_s_early_60_s,US+Europe,Trousers_m+Trousers_w,,,,,  
French_Trousers,,French_Chic,All,Trousers_w,,,,,  
Futuristic_Flip-Flops,,Futuristic,US,Shoes_w,,,,,  
Green_Kids_Gogo_Boots,,The_80_s+Futuristic,All,Shoes_k,,,,,  
Dark_Shirts,,Gothic,All,Shirts_m,,,,,  
Christmas_Tree_Skirts,,Late_50_s_early_60_s,US,Skirts_w,,,,,  
Leather_Rocker_Jacket,,Rockers,US,Jackets_m,,,,,
```

## Custom Application for Knowledge

---

To create a custom application for your Salesforce Knowledge base:

1. From Setup, enter *Apps* in the Quick Find box, then select **Apps**.
2. If you see an introductory splash page, simply click **Continue**.
3. Click **New**.  
The New Custom App wizard appears.
4. In the App Label field, enter *Knowledge*.
5. In the Description field, enter *My Custom Knowledge Base*.
6. Click **Next**.
7. Click **Next**.
8. Select the Home, Articles, Article Management tab and click **Next**.
9. In the Assign to Profiles page, select the Visible checkbox next to the Standard User and System Administrator profiles.
10. Click **Save**.

The application menu located at the top-right corner of your browser now shows a Knowledge custom application. Select this application to run your code samples.

# CHAPTER 3 Building a Search Page with Visualforce Tags

In this chapter ...

- [Create an Article List with Visualforce](#)
- [Add Pagination to the Visualforce Article List](#)
- [Add a Search Field to the Visualforce Article Search page](#)
- [Add a Filter to the Visualforce Article Search Page](#)
- [Add a Last Updated Promotions Box to the Visualforce Article Search Page](#)

## Including Test Classes When Deploying to a Production Organization

---

Some of the code samples in this section have corresponding test classes. If you are deploying to a production organization, include the test classes with your code.

Test classes contain unit tests for validating the behavior of Apex classes and triggers. Unit tests are class methods that verify whether a particular piece of code is working properly. Unit test methods take no arguments, commit no data to the database, and are flagged with the `testMethod` keyword in the method definition.

All test methods in an organization are executed whenever Apex code is deployed to a production organization to confirm correctness, ensure code coverage, and prevent regressions. All Apex classes are required to have at least 75% code coverage in order to be deployed to a production organization. In addition, all triggers must have some code coverage.

The `@isTest` class annotation indicates this class only contains test methods. Classes defined with the `@isTest` annotation do not count against the organization size limit for all Apex scripts. See the [Apex Developer Guide](#) for more information about testing and code coverage.

In this section you'll build a search page using Visualforce. Complete these steps:

1. [Create an Article List with Visualforce](#)
2. [Add Pagination to the Visualforce Article List](#)
3. [Add a Search Field to the Visualforce Article Search page](#)
4. [Add a Filter to the Visualforce Article Search Page](#)
5. [Add a Last Updated Promotions Box to the Visualforce Article Search Page](#)

## Create an Article List with Visualforce

---

Let's build the first component of the Visualforce search page, the article list.

1. In the address bar of your browser, replace everything to the right of `salesforce.com/` with `apex/ArticleList`. Don't change anything to the left of `salesforce.com/`.

The resulting URL should look something like this: `https://instance.salesforce.com/apex/ArticleList`.

2. Press **Enter**.

A Visualforce error page appears indicating that the page doesn't exist yet.

3. Click the **Create Page ArticleList** link.

4. In the footer at the bottom of your new Visualforce page, click **ArticleList** to display the Visualforce development mode page editor.

5. Delete all the default markup in the Visualforce development mode page editor and replace it with the markup shown in the [Visualforce Article List Code Sample](#) on page 12.

6. Click **Save** (🏠).

When you save your markup, the Force.com platform checks to make sure it's valid and lets you know if there are errors. If the markup is valid, the new version of your Visualforce page is saved and rendered in your browser. You can now see the Article List page displaying a list of articles.

## Visualforce Article List Code Sample

Lines in bold are described below the code sample.

```
<apex:page sidebar="false" title="Article List">
  <style>
    td{
      vertical-align : top;
      text-align: left;
    }
  </style>
  <apex:form >
    <apex:pageBlock title="Article List" >
      <apex:panelGrid width="100%">
        <table width="99%">
          <tr>
            <th width="33%">Title</th>
            <th width="33%">Article Type</th>
            <th width="33%">Summary</th>
          </tr>
        </table>
        <knowledge:articleList articleVar="article" hasMoreVar="false" pageSize="10">
        <table width="99%">
          <tr>
            <td width="33%">
              <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View,
article.id, ['popup' = 'true'])}">{!article.title}</apex:outputLink>
            </td>
            <td width="33%"><apex:outputText >{!article.articleTypeLabel}</apex:outputText></td>
```

```

        <td width="33%"><apex:outputText >{!article.abstract}</apex:outputText></td>
    </tr>
</table>
</knowledge:articleList>
</apex:panelGrid>
</apex:pageBlock>
</apex:form>
</apex:page>

```

In the code fragment below, the `knowledge:articleList` Visualforce component lets you create your first article list. The `pageSize` attribute prevents it from showing more than 10 articles in the list.

```
<knowledge:articleList articleVar="article" hasMoreVar="false" pageSize="10">
```

## Creating the Visualforce Article Search tab

Now, we're going to create a new custom tab for the Visualforce page that displays the article list. We'll call this page the Visualforce Article Search page.

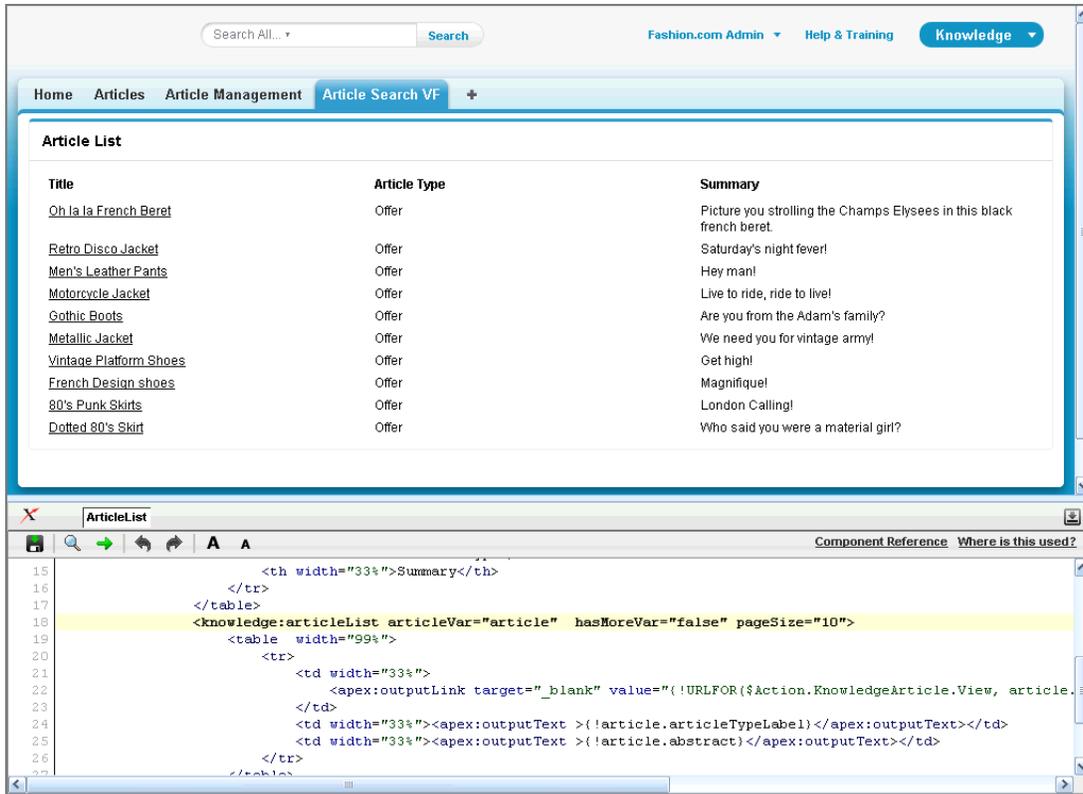
1. From Setup, enter *Tabs* in the *Quick Find* box, then select **Tabs**.
2. In the Visualforce tabs area, click **New**.
3. In the *Visualforce Page* drop-down list, select *ArticleList*.
4. In the *Tab Label* field, enter *Visualforce Article Search*.
5. Leave the default value in the *Tab Name* field.
6. Click the *Tab Style* lookup icon to select a style for your new tab.
7. Leave the *Salesforce Classic Mobile Ready* checkbox deselected and the *Splash Page Custom Link* drop-down list set to *--None--*.
8. In the *Description* field, enter *A tab for the Visualforce Article Search page*.
9. Click **Next**.
10. Click **Next** again to accept the default user profile visibility.
11. In the *Add to Custom Apps* page, deselect all of the *Include Tab* checkboxes except the one for our Knowledge app.
12. Select the *Append tab to users' existing personal customizations* checkbox.
13. Click **Save**.

You'll notice when the page refreshes that the Visualforce Article Search tab has automatically been added to your application tabs at the top of the page.

## Look at What We've Done

Select the Visualforce Article Search tab to display the article list.

Every item in the list shows the article's title, the article type and the article's summary. So far, your list only shows 10 articles per page as set in the `knowledge:articleList` tag from the Visualforce List code sample.



## Add Pagination to the Visualforce Article List

Let's add a **Next** and a **Previous** link to navigate from one page to another on the Visualforce Article Search tab.

1. In the footer at the bottom of the Visualforce Article Search tab, click **ArticleList** to display the Visualforce development mode page editor.
2. Delete all the existing markup in the Visualforce development mode page editor and replace it with the markup shown in the [Visualforce Article List Pagination Code Sample](#) on page 15.
3. Click **Save** (📁).

Since your new Visualforce code refers to Apex classes and methods, subsequent error messages appear and ask you to declare the expected class or method. Instead of declaring these items one at a time, let's just create the appropriate Controller.

4. From Setup, enter *Apex Classes* in the **Quick Find** box, then select **Apex Classes**.
5. Click **New**
6. Copy and paste the content of the [Visualforce Pagination Controller Code Sample](#) on page 16.
7. Click **Save**.
8. Repeat steps 1 to 3.

## Visualforce Article List Pagination Code Sample

This code sample is based on the Visualforce page you first saved but includes some enhancements to enable pagination. Lines in bold are described below the code sample.

```
<apex:page sidebar="false" title="Article List" controller="vfListPaginationController">
  <style>
    td{
      vertical-align : top;
      text-align: left;
    }
  </style>
  <apex:form >
    <apex:pageBlock title="Article List" >
      <apex:panelGroup id="theSearchResults" >
        <apex:panelGrid width="100%">
          <table width="99%">
            <tr>
              <th width="33%">Title</th>
              <th width="33%">Article Type</th>
              <th width="33%">Summary</th>
            </tr>
          </table>
          <knowledge:articleList articleVar="article" pageNumber="{!currentPageNumber}"
            hasMoreVar="false" pageSize="10">
            <table width="99%">
              <tr>
                <td width="33%">
                  <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View,
                    article.id,['popup' = 'true'])}">{!article.title}</apex:outputLink>
                </td>
                <td width="33%"><apex:outputText >{!article.articleTypeLabel}</apex:outputText></td>
                <td width="33%"><apex:outputText >{!article.abstract}</apex:outputText></td>
              </tr>
            </table>
          </knowledge:articleList>
        </apex:panelGrid>
        <apex:panelGrid columns="2">
          <apex:commandLink action="{!previous}" value="Previous" style="{!IF(prevRequired =
            true, 'display:block', 'display:none')}" reRender="theSearchResults"/>
          <apex:commandLink action="{!next}" value="Next" style="{!IF(nextRequired =
            true, 'display:block', 'display:none')}" reRender="theSearchResults"/>
        </apex:panelGrid>
      </apex:panelGroup>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

The code `controller="vfListPaginationController"` references the controller used with the Visualforce page, while `pageNumber="{!currentPageNumber}"` sets the pagination for the article list. The `currentPageNumber` variable is updated each time a user clicks the **Next** or **Previous** link.

The code fragment below, calls the `previous` method from the controller. The `reRender` attribute refreshes the article list only and not the whole page.

```
<apex:commandLink action="{!previous}" value="Previous" style="{!IF(prevRequired = true, 'display:block', 'display:none')}"/> reRender="theSearchResults"/>
```

## Visualforce Pagination Controller Code Sample

Lines in bold are described below the code sample.

```
public with sharing class vfListPaginationController {

    //Page Size
    private Static Final Integer PAGE_NUMBER = 10;

    public vfListPaginationController() {
        String qryString = 'SELECT Id, title, UrlName, LastPublishedDate, LastModifiedById FROM KnowledgeArticleVersion WHERE (PublishStatus = \'online\' and Language = \'en_US\')';
        List<KnowledgeArticleVersion> articleList= Database.query(qryString);
        maxSize = articleList.size() ;
    }

    //Keeps track of current page & max size of article list
    Integer currentPage = 1;
    Integer maxSize = 1;

    // Returns whether we need to see previous button or not
    public boolean getPrevRequired() {
        return currentPage > 1;
    }

    // Returns whether we need to see next button or not
    public boolean getNextRequired() {
        return currentPage * PAGE_NUMBER < maxSize;
    }

    //Returns current page number
    public Decimal getCurrentPageNumber() {
        return this.currentPage;
    }

    //action for next click
    public PageReference next() {
        if(maxSize > this.currentPage * PAGE_NUMBER) {
            this.currentPage = this.currentPage + 1;
        }
        return null;
    }

    //action for previous click
    public PageReference previous() {
        if(this.currentPage > 1)
            this.currentPage = this.currentPage - 1;
        return null;
    }
}
```

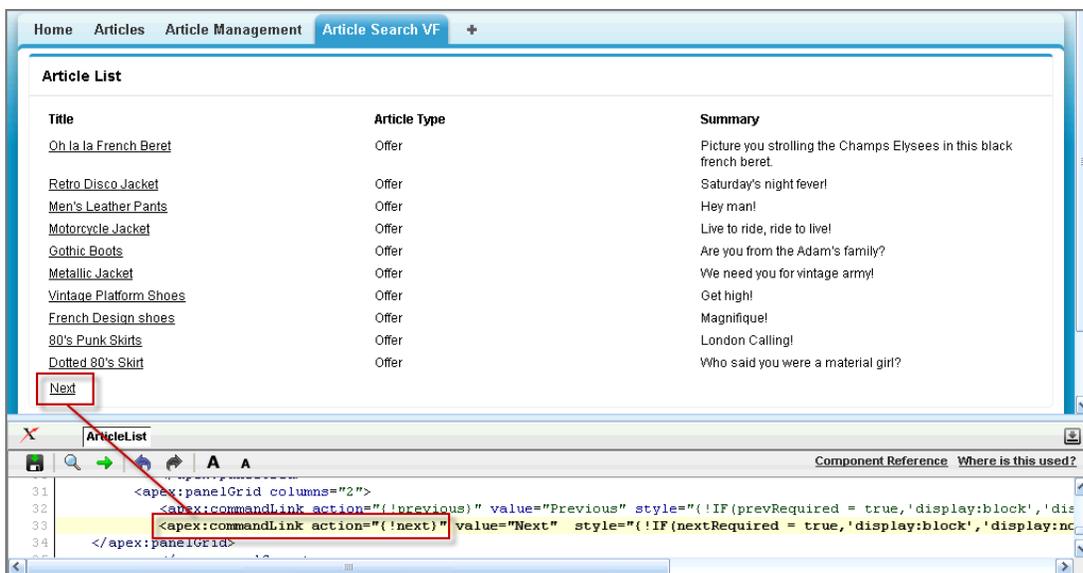
```
}
}
}
```

The code fragment below is an SOQL clause that retrieves the article information to display in the list. The `WHERE` clause specifies that only published articles are displayed in the Article List.

```
SELECT Id, title, UrlName, LastPublishedDate, LastModifiedById FROM KnowledgeArticleVersion
WHERE (PublishStatus = \'online\' and Language = \'en_US\')
```

## Look at What We've Done

When you save your work, the Visualforce Article Search page refreshes and now displays a **Next** link at the bottom of the list. If you click **Next**, the page now shows a **Previous** link.



## Add a Search Field to the Visualforce Article Search page

Let's create a keyword search field above the Article List. Entering text in this field refreshes the list and displays only articles containing this keyword.

Using a search field requires new methods to set and retrieve the search string. That's why we'll add a new controller before modifying the code from the Visualforce Article Search page.

1. From Setup, enter *Apex Classes* in the Quick Find box, then select **Apex Classes**.
2. Click **New**
3. Copy and paste the content of the [Visualforce Keyword Search Controller Code Sample](#) on page 19.
4. Click **Save**.
5. In the footer of the Visualforce Article Search tab, click **ArticleList** to display the Visualforce development mode page editor.
6. Delete all the existing markup in the Visualforce development mode page editor and replace it with the markup shown in the [Visualforce Keyword Search Code Sample](#) on page 18.

## Visualforce Keyword Search Code Sample

Lines in bold are described below the code sample.

```

<apex:page sidebar="false" title="Article List" controller="vfKeywordSearchController">
  <style>
    td{
      vertical-align : top;
      text-align: left;
    }
  </style>
  <apex:form >
    <apex:pageBlock title="Search" >
      <apex:inputText value="{!searchstring}" id="theSearchstring" maxlength="100" size="110"/>
      &nbsp;
      <apex:commandButton value="Go" id="submitButton" style="width:30"
reRender="theSearchResults" />
    </apex:pageBlock>
    <apex:messages />
    <apex:pageBlock title="Article List" >

    <apex:panelGroup id="theSearchResults" >
      <apex:panelGrid width="100%">
        <table width="99%">
          <tr>
            <th width="33%">Title</th>
            <th width="33%">Article Type</th>
            <th width="33%">Summary</th>
          </tr>
        </table>
        <knowledge:articleList articleVar="article" pageNumber="{!currentPageNumber}"
Keyword="{!searchstring}" hasMoreVar="false" pageSize="10">
        <table width="99%">
          <tr>
            <td width="33%">
              <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View,
article.id,['popup' = 'true'])}">{!article.title}</apex:outputLink>
            </td>
            <td width="33%"><apex:outputText >{!article.articleTypeLabel}</apex:outputText></td>

            <td width="33%"><apex:outputText >{!article.abstract}</apex:outputText></td>
          </tr>
        </table>
        </knowledge:articleList>
      </apex:panelGrid>
      <apex:panelGrid columns="2">
        <apex:commandLink action="{!previous}" value="Previous" style="{!IF(prevRequired =
true,'display:block','display:none')}" reRender="theSearchResults"/>
        <apex:commandLink action="{!next}" value="Next" style="{!IF(nextRequired =
true,'display:block','display:none')}" reRender="theSearchResults"/>
      </apex:panelGrid>
    </apex:panelGroup>
  </apex:pageBlock>
</apex:form>
</apex:page>

```

The code fragment below uses the `Keyword` attribute. The value for this attribute invokes the `searchstring` method of the `vfKeywordSearchController` controller.

```
<knowledge:articleList articleVar="article" pageNumber="{!currentPageNumber}"
Keyword="{!searchstring}" hasMoreVar="false" pageSize="10">
```

## Visualforce Keyword Search Controller Code Sample

Lines in bold are described below the code sample.

```
public with sharing class vfKeywordSearchController {

    //Page Size
    private static final Integer PAGE_NUMBER = 10;

    //Search String used in ArticleList tag
    public String searchstring { get; set; }

    public vfKeywordSearchController() {
        String qryString = 'SELECT Id, title, UrlName, LastPublishedDate, LastModifiedById FROM
KnowledgeArticleVersion WHERE (PublishStatus = \'online\' and Language = \'en_US\')';
        List<KnowledgeArticleVersion> articleList= Database.query(qryString);
        maxSize = articleList.size() ;
    }

    //Keeps track of current page & max size of article list
    Integer currentPage = 1;
    Integer maxSize = 1;

    // Returns whether we need to see previous button or not
    public boolean getPrevRequired() {
        return currentPage > 1;
    }

    // Returns whether we need to see next button or not
    public boolean getNextRequired() {
        return currentPage * PAGE_NUMBER < maxSize;
    }

    //Returns current page number
    public Decimal getCurrentPageNumber() {
        return this.currentPage;
    }

    //action for next click
    public PageReference next() {
        if(maxSize > this.currentPage * PAGE_NUMBER) {
            this.currentPage = this.currentPage + 1;
        }
        return null;
    }

    //action for previous click
    public PageReference previous() {
```

```

if(this.currentPage > 1)
    this.currentPage = this.currentPage - 1;
return null;
}
}

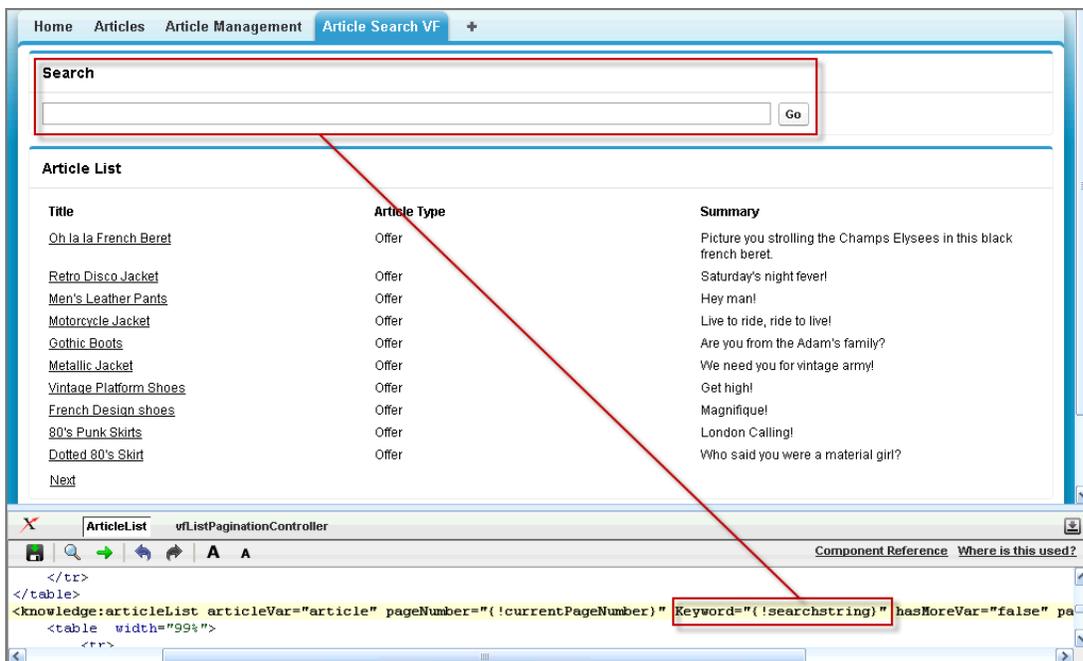
```

The code fragment below sets and retrieves the search keyword.

```
public String searchstring { get; set; }
```

## Look at What We've Done

The Visualforce Article Search page now shows a Search field above the article list.



## Add a Filter to the Visualforce Article Search Page

In addition to the Search field, let's show the data categories to filter the article list so users can refine the search results by store, country, or fashion.

Adding a filter to a Visualforce page requires that you set the methods to list, retrieve, and insert data categories. You'll also need to add the new controller to call the appropriate methods and process the page.

1. From Setup, enter *Apex Classes* in the Quick Find box, then select **Apex Classes**.
2. Add the following code samples:
  - a. [Data Category Information Code Sample](#)
  - b. [Data Category Group Information Code Sample](#)
  - c. [Data Category Utility Code Sample](#)

#### d. Visualforce Search Controller Code Sample

3. In the footer at the bottom of the Visualforce Article Search tab, click **ArticleList** to display the Visualforce development mode page editor.
4. Delete all the existing markup in the Visualforce development mode page editor and replace it with the markup in the [Visualforce Search Code Sample](#) on page 29.
5. Click **Save** (🏠).

## Data Category Information Code Sample

```
/**
 * This class holds the Data Category Info. from
 * the describe result
 */
public class DataCategoryInfo {

    private String name;
    private String label;
    private String displayName;

    public DataCategoryInfo(String name, String label, String displayName) {
        this.name = name;
        this.label = label;
        this.displayName = displayName;
    }

    public String getName() {
        return this.name;
    }

    public String getLabel() {
        return this.label;
    }

    public String getDisplayName() {
        return this.displayName;
    }
}
```

## Data Category Information Test

The following is the test class for the [Data Category Information Code Sample](#) on page 21.

```
@isTest
private class DataCategoryInfoTestClass {

    static testMethod void validateDataCategoryInfoObject() {
        DataCategoryInfo dataCategoryInfo = new DataCategoryInfo('Products__c', 'Products__c',
'Products');
        System.assertEquals('Products__c', dataCategoryInfo.getName());
        System.assertEquals('Products__c', dataCategoryInfo.getLabel());
        System.assertEquals('Products', dataCategoryInfo.getDisplayName());
    }
}
```

```
}
}
```

## Data Category Group Information Code Sample

```
/**
 * This class holds the Data Category Group Info.
 * from the describe result
 */
public class DataCategoryGroupInfo {

    private String name;
    private String label;
    private String description;
    private DataCategoryInfo[] groupStructure;

    public DataCategoryGroupInfo (String name, String label, String description,
DataCategoryInfo[] structure) {
        this.name = name;
        this.label = label;
        this.description = description;
        this.groupStructure = structure;
    }

    public String getName(){
        return this.name;
    }

    public String getLabel(){
        return this.label;
    }

    public String getDescription(){
        return this.description;
    }

    public DataCategoryInfo[] getGroupStructure(){
        return this.groupStructure;
    }

}
```

## Data Category Group Information Test

The following is the test class for the [Data Category Group Information Code Sample](#) on page 22.

```
@isTest
private class DataCategoryGroupInfoTestClass {

    static testMethod void validateDataCategoryGroupInfoObject(){
        DataCategoryInfo[] structure = new DataCategoryInfo[1];
```

```

    structure[0] = new DataCategoryInfo('Product','Product__c','Product__c');
    DataCategoryGroupInfo dataCategoryGroupInfo = new
DataCategoryGroupInfo('Product','Product__c','Test DataCategoryInfoClass',structure);
    System.assertEquals('Product', dataCategoryGroupInfo.getName());
    System.assertEquals('Product__c', dataCategoryGroupInfo.getLabel());
    System.assertEquals('Test DataCategoryInfoClass', dataCategoryGroupInfo.getDescription());

    System.assertEquals(structure.size(),dataCategoryGroupInfo.getGroupStructure().size());

}

}

```

## Data Category Utility Code Sample

```

/**
 * This class contains utility functions used by
 * the controller
 */
public with sharing class DataCategoryUtil {

    private static final DataCategoryUtil INSTANCE = new DataCategoryUtil();
    private static final String NON_BREAKING_SPACE = '&nbsp;&nbsp; ';
    private final String ARTICLE_TYPE = 'KnowledgeArticleVersion';

    private DataCategoryUtil() { }

    public static DataCategoryUtil getInstance() { return INSTANCE; }

    /**
     * Retrieves a List of DataCategoryGroupInfo of all the data category
     * groups associated with Knowledge Article Version
     */
    public DataCategoryGroupInfo[] getAllCategoryGroups() {
        Describedatacategorygroupresult[] results = getDescribeDataCategoryGroupResults();
        DataCategoryGroupInfo[] dataCategoryGroups = new DataCategoryGroupInfo[] { };
        for (Describedatacategorygroupresult singleResult : results) {
            dataCategoryGroups.add(
                new DataCategoryGroupInfo(singleResult.getName(), singleResult.getLabel(),
                    singleResult.getDescription(), getCategoryStructure(singleResult.getName())));
        }
        return dataCategoryGroups;
    }

    private Describedatacategorygroupresult[] getDescribeDataCategoryGroupResults() {
        String[] objTypes = new String[] {ARTICLE_TYPE};
        return Schema.describeDataCategoryGroups(objTypes);
    }

    /**
     * Retrieves an ordered list of all Categories for the specified category group
     */
    private DataCategoryInfo[] getCategoryStructure(String categoryName) {

```

```

    DataCategory root = getRootCategory(categoryGroupName);
    DataCategory[] yetToBeProcessed = root.getChildCategories();
    DataCategoryInfo[] allCategories = new DataCategoryInfo[] { processCategory(root, '')
};
    getAllCategories(yetToBeProcessed, allCategories, NON_BREAKING_SPACE);
    return allCategories;
}

private void getAllCategories(DataCategory[] yetToBeProcessed, DataCategoryInfo[] processed,
String labelPrefix) {
    for (DataCategory category : yetToBeProcessed) {
        processed.add(processCategory(category, labelPrefix));
        getAllCategories(category.getChildCategories(), processed, labelPrefix +
NON_BREAKING_SPACE);
    }
}

private DataCategoryInfo processCategory(DataCategory category, String labelPrefix) {
    return new DataCategoryInfo(category.getName(), category.getLabel(), labelPrefix +
category.getLabel());
}

private DataCategory getRootCategory(String categoryGroupName) {
    Describedatacategorygroupstructureresult structureResult =
getDescribeDataCategoryGroupStructureResults(categoryGroupName);
    return structureResult.getTopCategories()[0];
}

private Describedatacategorygroupstructureresult
getDescribeDataCategoryGroupStructureResults(String categoryGroupName) {
    Datacategorygroupsubjecttypepair pair = new Datacategorygroupsubjecttypepair();
    pair.setSubject(ARTICLE_TYPE);
    pair.setDataCategoryGroupName(categoryGroupName);
    return Schema.describeDataCategoryGroupStructures(new Datacategorygroupsubjecttypepair[]
{ pair }, false)[0];
}
}

```

## Data Category Utility Test

The following is the test class for the [Data Category Utility Code Sample](#) on page 23.

```

@isTest
private class DataCategoryUtilTestClass {

    static testMethod void validateCategoryGroups(){
        String[] objTypes = new String[] { 'KnowledgeArticleVersion' };
        Describedatacategorygroupresult[] datacategorygroupresult =
Schema.describeDataCategoryGroups(objTypes);
        DataCategoryGroupInfo[] dataCategoryGroupInfo =
DataCategoryUtil.getInstance().getAllCategoryGroups();
        System.assertEquals(datacategorygroupresult.size(), dataCategoryGroupInfo.size());
    }
}

```

```

static testMethod void validateCategoryGroupsResults() {
    DataCategoryGroupInfo[] dataCategoryGroupInfo =
DataCategoryUtil.getInstance().getAllCategoryGroups();
    String[] objTypes = new String[] { 'KnowledgeArticleVersion' };
    Describedatacategorygroupresult[] datacategorygroupresult =
Schema.describeDataCategoryGroups(objTypes);
    for(Integer i=0;i< dataCategoryGroupInfo.size(); i++) {

System.assertEquals(dataCategoryGroupInfo[i].getName(),datacategorygroupresult[i].getName());

System.assertEquals(dataCategoryGroupInfo[i].getLabel(),datacategorygroupresult[i].getLabel());

System.assertEquals(dataCategoryGroupInfo[i].getDescription(),datacategorygroupresult[i].getDescription());

    }
}
}

```

## Visualforce Search Controller Code Sample

Lines in bold are described below the code sample.

```

/**
 * Controller for Visual force Page.
 * author SFDC
 */
public with sharing class VfSearchController{

    //Page Size
    private Static Final Integer PAGE_NUMBER = 10;

    //Search String used in ArticleList tag
    public String searchstring { get; set; }

    //Is new List reqd
    private boolean isRefRequired = true;
    //Exclude filter criteria for UI only
    private static final String EXCLUDE_CRITERIA_FILTER = 'All';

    //Keeps track of current page & max size of article list
    Integer currentPage = 1;
    Integer maxSize = 1;

    //Returns array of Category Groups
    public DataCategoryGroupInfo[] getDataCategoryGroupInfo() {
        return DataCategoryUtil.getInstance().getAllCategoryGroups();
    }

    //Returns category keyword required to filter articleList.
public String getCategoryKeyword() {
    DataCategoryGroupInfo[] categoryGroups =

```

```

DataCategoryUtil.getInstance().getAllCategoryGroups();
String categoryCondition = '';
for (DataCategoryGroupInfo categoryGroup : categoryGroups) {
    String selectedCategoryName =
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

    if(selectedCategoryName != null && !selectedCategoryName.equals('NoFilter')) {
        if(categoryCondition==' ' && selectedCategoryName != null){
            categoryCondition=categoryCondition+categoryGroup.getName() + ':' +
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

        }else {
            categoryCondition=categoryCondition + ',' +categoryGroup.getName() + ':' +
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

        }
    }
}
String categoryFilter = '';
for (DataCategoryGroupInfo categoryGroup : categoryGroups) {
    String categoryType =
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

    if(categoryType != null && !categoryType.equals('NoFilter')) {
        if(categoryFilter == ' '){
            categoryFilter = categoryGroup.getName() + '__c ABOVE_OR_BELOW ' + categoryType
+'__c';
        } else {
            categoryFilter = categoryFilter + ' AND ' + categoryGroup.getName() +'__c ABOVE_OR_BELOW
' + categoryType +'__c';
        }
    }
}
try {
    if(categoryFilter.length()>0) {
        if(searchString != null && searchString.length() >0 ) {
            String searchquery = 'FIND \'' + searchString + '*\'IN ALL FIELDS RETURNING
KnowledgeArticleVersion(Id, title, UrlName, LastPublishedDate,LastModifiedById where
PublishStatus =\'online\' and Language = \'en_US\') WITH DATA CATEGORY '+categoryFilter ;

            List<List<SObject>>searchList = search.query(searchquery);
            List<KnowledgeArticleVersion> articleList =
(List<KnowledgeArticleVersion>)searchList[0];
            maxSize = articleList.size() ;
//                maxSize = maxSize.divide(PAGE_NUMBER,2, System.RoundingMode.UP);
        } else {
            String qryString = 'SELECT Id, title, UrlName, LastPublishedDate,LastModifiedById FROM
KnowledgeArticleVersion WHERE (PublishStatus = \'online\' and Language = \'en_US\') WITH
DATA CATEGORY '+categoryFilter;
            List<KnowledgeArticleVersion> articleList= Database.query(qryString);
            maxSize = articleList.size() ;
//                maxSize = maxSize.divide(PAGE_NUMBER,2, System.RoundingMode.UP);
        }
    } else {

```

```

    String qryString = 'SELECT Id, title, UrlName, LastPublishedDate,LastModifiedById FROM
KnowledgeArticleVersion WHERE (PublishStatus = \'online\' and Language = \'en_US\')';
    List<KnowledgeArticleVersion> articleList= Database.query(qryString);
    maxSize = articleList.size() ;
//          maxSize = maxSize.divide(PAGE_NUMBER,2,System.RoundingMode.UP);
    }
    } catch(Exception e) {
        Apexpages.addmessages( e );
    }
    if(categoryFilter =='') {
//          maxSize = 0;
        categoryCondition = 'Fashions:All,Stores:All,Products:All' ;
    }
    return categoryCondition;
}

// Action call when the new list needs to be fetched
public PageReference refreshSearchResult() {
    maxSize = currentPage = 1;
    return null;
}

// Returns whether we need to see previous button or not
public boolean getPrevRequired() {
    return currentPage > 1;
}

// Returns whether we need to see next button or not
public boolean getNextRequired() {
    return currentPage * PAGE_NUMBER < maxSize;
}

//Returns current page number
public Decimal getCurrentPageNumber() {
    return this.currentPage;
}

//action for next click
public PageReference next() {
    if(maxSize > this.currentPage * PAGE_NUMBER) {
        this.currentPage = this.currentPage + 1;
    }
    return null;
}

//action for previous click
public PageReference previous() {
    if(this.currentPage > 1)
        this.currentPage = this.currentPage - 1;
    return null;
}
}

```

The code fragment below retrieves the categories selected in the filter.

```
public String getCategoryKeyword() {
    DataCategoryGroupInfo[] categoryGroups =
DataCategoryUtil.getInstance().getAllCategoryGroups();
    String categoryCondition = '';
    for (DataCategoryGroupInfo categoryGroup : categoryGroups) {
        String selectedCategoryName =
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

        if(selectedCategoryName != null && !selectedCategoryName.equals('NoFilter')) {
            if(categoryCondition==' ' && selectedCategoryName != null){
                categoryCondition=categoryCondition+categoryGroup.getName() + ':' +
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());
            }else {
                categoryCondition=categoryCondition + ',' +categoryGroup.getName() + ':' +
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());
            }
        }
    }
}
```

## Visualforce Search Controller Test

The following is the test class for the [Visualforce Search Controller Code Sample](#) on page 25.

```
@isTest
private class VfSearchControllerTestClass {

    static testMethod void validateDataCategoryGroupInfo(){
        DataCategoryGroupInfo[] dataCategoryGroupInfo =
DataCategoryUtil.getInstance().getAllCategoryGroups();
        VfSearchController vfSearchControllerObj = new VfSearchController();
        System.assertEquals(vfSearchControllerObj.getDataCategoryGroupInfo().size(),
dataCategoryGroupInfo.size());
    }

    static testMethod void testClassVariables() {
        VfSearchController vfSearchControllerObj = new VfSearchController();
        vfSearchControllerObj.refreshSearchResult();
        System.assertEquals(vfSearchControllerObj.getCurrentPageNumber(),1.0);
        vfSearchControllerObj.next();
        System.assertEquals(vfSearchControllerObj.getCurrentPageNumber(),1.0);
        vfSearchControllerObj.previous();
        System.assertEquals(vfSearchControllerObj.getCurrentPageNumber(),1.0);
        System.assertEquals(vfSearchControllerObj.getPrevRequired(),false);
        System.assertEquals(vfSearchControllerObj.getNextRequired(),false);
    }

    static testMethod void validateCategoryKeyword() {
        VfSearchController vfSearchControllerObj = new VfSearchController();
        DataCategoryGroupInfo[] categoryGroups =
```

```

DataCategoryUtil.getInstance().getAllCategoryGroups();
String categoryCondition = '';
for (DataCategoryGroupInfo categoryGroup : categoryGroups) {

ApexPages.currentPage().getParameters().put('categoryType_'+categoryGroup.getName(), 'All');

    if(categoryCondition==''){
        categoryCondition=categoryCondition+categoryGroup.getName() + ':' +
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

    }else {
        categoryCondition=categoryCondition + ',' +categoryGroup.getName() + ':' +
System.currentPageReference().getParameters().Get('categoryType_'+categoryGroup.getName());

    }
}
System.assertEquals(categoryCondition, vfSearchControllerObj.getCategoryKeyword());
}
}

```

## Visualforce Search Code Sample

Lines in bold are described below the code sample.

```

<apex:page controller="VfSearchController" sidebar="false" title="Knowledge Search">
<style>
    td{
        vertical-align : top;
        text-align: left;
    }
</style>
<apex:form >
    <apex:panelGrid columns="2" >
        <apex:panelGroup >
            <apex:pageBlock >
                <apex:outputText value="Filter Your Results" />
                <apex:pageBlockSection columns="1">
                    <apex:dataTable value="{!dataCategoryGroupInfo}" var="dataCategory" id="dataCategory">

                        <apex:column width="20%">
                            <apex:outputLabel
for="categoryType_{!dataCategory.name}">{!dataCategory.name}</apex:outputLabel>
                            <br />
                            <select id="categoryType_{!dataCategory.name}"
name="categoryType_{!dataCategory.name}" onchange = "refreshSearchResult()" >
                                <option value="NoFilter">No Filter</option>
                                <option value="All">All</option>
                                <knowledge:categoryList categoryVar="category" categoryGroup="{!dataCategory.name}"
rootCategory="All" level="-1">
                                    <option value="{!category.name}">
                                        <apex:outputText escape="false" value="{!LPAD(' ',6*category.depth,'&nbsp;')}>
/>>

```

```

        {!category.label}
    </option>
</knowledge:categoryList>
</select>
</apex:column>
</apex:dataTable>
</apex:pageBlockSection>
</apex:pageBlock>

</apex:panelGroup>
<apex:panelGroup >
  <apex:pageBlock title="Search" >
    <apex:inputText value="{!searchstring}" id="theSearchstring" maxlength="100" size="110"
    onkeypress="if (event.keyCode == 13) {refreshSearchResult();return false;} "/> &nbsp;
    <apex:commandButton value="Go" id="submitButton" style="width:30"
reRender="theSearchResults" />
  </apex:pageBlock>
  <apex:messages />
  <apex:panelGroup id="theSearchResults" >
  <apex:pageBlock title="Search Results" >
    <apex:panelGrid width="100%">
      <table width="99%">
        <tr>
          <th width="33%">Title</th>
          <th width="33%">Article Type</th>
          <th width="33%">Summary</th>
        </tr>
      </table>
      <knowledge:articleList articleVar="article" categories="{!categoryKeyword}"
Keyword="{!searchstring}" pageNumber="{!currentPageNumber}" hasMoreVar="false" pageSize="10">

      <table width="99%">
        <tr>
          <td width="33%">
            <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View,
article.id, ['popup' = 'true'])}">{!article.title}</apex:outputLink>
          </td>
          <td width="33%"><apex:outputText >{!article.articleTypeLabel}</apex:outputText></td>

          <td width="33%"><apex:outputText >{!article.abstract}</apex:outputText></td>
        </tr>
      </table>
    </knowledge:articleList>
  </apex:panelGrid>
</apex:pageBlock>
<apex:panelGrid columns="2">
  <apex:commandLink action="{!previous}" value="Previous" style="{!IF(prevRequired =
true, 'display:block', 'display:none')} " reRender="theSearchResults"/>
  <apex:commandLink action="{!next}" value="Next" style="{!IF(nextRequired =
true, 'display:block', 'display:none')} " reRender="theSearchResults"/>
</apex:panelGrid>
</apex:panelGroup>
</apex:panelGroup>
</apex:panelGrid>

```

```

<apex:actionFunction action="{!refreshSearchResult}" name="refreshSearchResult"
rerender="theSearchResults" >
  </apex:actionFunction>

</apex:form>
</apex:page>

```

In the code fragment below, the `<apex:dataTable>` tag uses the `getDataCategoryGroupInfo` method (declared in the [Visualforce Search Controller Code Sample](#)) to iterate each category group.

```

<apex:dataTable value="{!dataCategoryGroupInfo}" var="dataCategory" id="dataCategory">
  <apex:column width="20%">
    <apex:outputLabel
for="categoryType_{!dataCategory.name}">{!dataCategory.name}</apex:outputLabel>
    <br />

```

In the code fragment below, the `<select>` tag builds the category filter. The `onchange` attribute and its `refreshSearchResult` value update the article list every time a user changes the selection in the filter.

```

<select id="categoryType_{!dataCategory.name}" name="categoryType_{!dataCategory.name}"
onchange = "refreshSearchResult()" >
  <option value="NoFilter">No Filter</option>
  <option value="All">All</option>

```

In the code fragment below, the `knowledge:categoryList` tag performs a loop for all the categories under All in the category group.

```

<knowledge:categoryList categoryVar="category" categoryGroup="{!dataCategory.name}"
rootCategory="All" level="-1">
  <option value="{!category.name}">
    <apex:outputText escape="false" value="{!LPAD(' ',6*category.depth,'&nbsp;')}" />
    {!category.label}
  </option>
</knowledge:categoryList>
</select>
</apex:column>
</apex:dataTable>

```

The code fragment below calls the `refreshSearchResult` method (in the [Visualforce Search Controller Code Sample](#)) when the page needs to be refreshed. Only the Search Results section is refreshed in the Visualforce Article Search page.

```

<apex:actionFunction action="{!refreshSearchResult}" name="refreshSearchResult"
rerender="theSearchResults" >

```

## Look at What We've Done

The Visualforce Article Search page now displays a `Filter Your Results` box. Changing the filter selection refreshes the Search Results box. Only articles matching the selected categories are shown.

The screenshot shows a Visualforce page titled "Article Search VF". On the left, there is a "Filter Your Results" sidebar with three dropdown menus for "Fashions", "Stores", and "Products", each set to "No Filter". The main content area is a search results table with the following data:

Title	Article Type	Summary
<a href="#">Oh la la French Beret</a>	Offer	Picture you strolling the Champs Elysees in this black french beret.
<a href="#">Retro Disco Jacket</a>	Offer	Saturday's night fever!
<a href="#">Men's Leather Pants</a>	Offer	Hey man!
<a href="#">Motorcycle Jacket</a>	Offer	Live to ride, ride to live!
<a href="#">Gothic Boots</a>	Offer	Are you from the Adam's family?
<a href="#">Metallic Jacket</a>	Offer	We need you for vintage army!
<a href="#">Vintage Platform Shoes</a>	Offer	Get high!
<a href="#">French Design shoes</a>	Offer	Magnifique!
<a href="#">80's Punk Skirts</a>	Offer	London Calling!
<a href="#">Dotted 80's Skirt</a>	Offer	Who said you were a material girl?
<a href="#">Next</a>		

Below the table, a code editor shows the following markup:

```
<apex:outputText value="Filter Your Results" />
<apex:pageBlockSection columns="1">
  <apex:dataTable value="{!dataCategoryGroupInfo}" var="dataCategory" id="dataCategory">
    <apex:column width="20%">
      <apex:outputLabel for="categoryType_{!dataCategory.name}">{!dataCategory.name}</apex:outputLabel>
      <br />
      <select id="categoryType_{!dataCategory.name}" name="categoryType_{!dataCategory.name}" onchange = "refre
        <option value="NoFilter">No Filter</option>
        <option value="All">All</option>
        <knowledge:categoryList categoryVar="category" categoryGroup="{!dataCategory.name}" rootCategory="All
        <option value="{!category.name}">
          <apex:outputText escape="false" value="{!LPAD(' ',6*category.depth,'&nbsp;')}"/>
            {!category.label}
          </option>
        </knowledge:categoryList>
```

## Add a Last Updated Promotions Box to the Visualforce Article Search Page

Finally, we'll add a Last Updated Promotions box.

1. In the footer at the bottom of the Visualforce Article Search tab, click **ArticleList** to display the Visualforce development mode page editor.
2. Delete all the existing markup in the Visualforce development mode page editor and replace it with the markup shown in the [Visualforce Box Code Sample](#) on page 32.
3. Click **Save** (📁).

### Visualforce Box Code Sample

Lines in bold are described below the code sample.

```
<apex:page controller="VfSearchController" sidebar="false" title="Knowledge Search">
<style>
  td{
    vertical-align : top;
    text-align: left;
  }
</style>
```

```

<apex:form >
  <apex:panelGrid columns="2" >
    <apex:panelGroup >
      <apex:pageBlock >
        <apex:outputText value="Filter Your Results" />
        <apex:pageBlockSection columns="1">
          <apex:dataTable value="{!dataCategoryGroupInfo}" var="dataCategory" id="dataCategory">

            <apex:column width="20%">
              <apex:outputLabel
for="categoryType_{!dataCategory.name}">{!dataCategory.name}</apex:outputLabel>
              <br />
              <select id="categoryType_{!dataCategory.name}"
name="categoryType_{!dataCategory.name}" onchange = "refreshSearchResult()" >
                <option value="NoFilter">No Filter</option>
                <option value="All">All</option>
                <knowledge:categoryList categoryVar="category" categoryGroup="{!dataCategory.name}"
rootCategory="All" level="-1">
                  <option value="{!category.name}">
                    <apex:outputText escape="false" value="{!LPAD(' ',6*category.depth,'&nbsp;')}>
                </option>
              </select>
            </apex:column>
          </apex:dataTable>
        </apex:pageBlockSection>
      </apex:pageBlock>
      <apex:pageBlock >
        <apex:outputText value="Last Updated Promotions" />
        <apex:pageBlockSection columns="1">
          <knowledge:articleList articleVar="article" articleTypes="Promotion__kav" pageSize="10"
          >
            <li>
              <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View,
article.id, ['popup' = 'true'])}">{!article.title}</apex:outputLink>
            </li>
          </knowledge:articleList>
        </apex:pageBlockSection>
      </apex:pageBlock>
    </apex:panelGroup>
    <apex:panelGroup >
      <apex:pageBlock title="Search" >
        <apex:inputText value="{!searchstring}" id="theSearchstring" maxLength="100" size="110"
onkeypress="if (event.keyCode == 13) {refreshSearchResult();return false;} "/> &nbsp;
        <apex:commandButton value="Go" id="submitButton" style="width:30"
reRender="theSearchResults" />
      </apex:pageBlock>
      <apex:messages />
      <apex:panelGroup id="theSearchResults" >
        <apex:pageBlock title="Search Results" >
          <apex:panelGrid width="100%">
            <table width="99%">

```

```

<tr>
  <th width="33%">Title</th>
  <th width="33%">Article Type</th>
  <th width="33%">Summary</th>
</tr>
</table>
<knowledge:articleList articleVar="article" categories="{!categoryKeyword}"
Keyword="{!searchstring}" pageNumber="{!currentPageNumber}" hasMoreVar="false" pageSize="10">

<table width="99%">
  <tr>
    <td width="33%">
      <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View,
article.id,['popup' = 'true'])}">{!article.title}</apex:outputLink>
    </td>
    <td width="33%"><apex:outputText >{!article.articleTypeLabel}</apex:outputText></td>

    <td width="33%"><apex:outputText >{!article.abstract}</apex:outputText></td>
  </tr>
</table>
</knowledge:articleList>
</apex:panelGrid>
</apex:pageBlock>
<apex:panelGrid columns="2">
  <apex:commandLink action="{!previous}" value="Previous" style="{!IF(prevRequired =
true,'display:block','display:none')}" reRender="theSearchResults"/>
  <apex:commandLink action="{!next}" value="Next" style="{!IF(nextRequired =
true,'display:block','display:none')}" reRender="theSearchResults"/>
</apex:panelGrid>
</apex:panelGroup>
</apex:panelGroup>
</apex:panelGrid>
<apex:actionFunction action="{!refreshSearchResult}" name="refreshSearchResult"
reRender="theSearchResults" >
  </apex:actionFunction>

</apex:form>
</apex:page>

```

In the code fragment below, the `<knowledge:articleList>` tag retrieves by default any last published articles. The `articleTypes` attribute specifies that only Promotion articles are displayed in the Last Updated Promotions box.

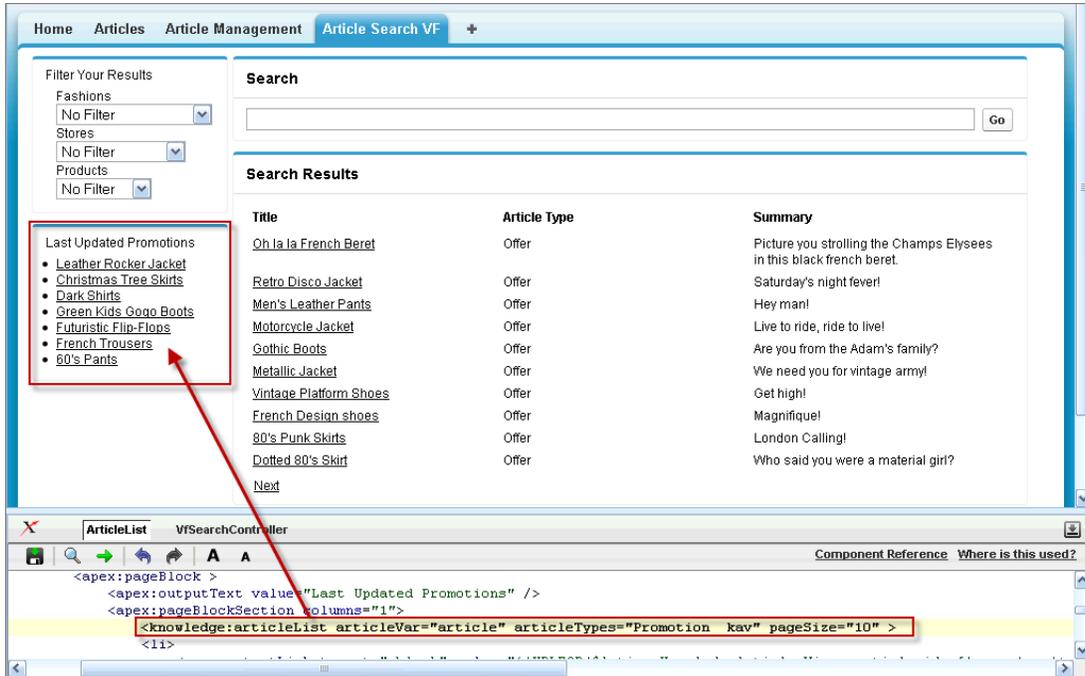
```

<apex:outputText value="Last Updated Promotions" />
<apex:pageBlockSection columns="1">
  <knowledge:articleList articleVar="article" articleTypes="Promotion__kav" pageSize="10"
  >
  <li>
    <apex:outputLink target="_blank" value="{!URLFOR($Action.KnowledgeArticle.View, article.id,
['popup' = 'true'])}">{!article.title}</apex:outputLink>
  </li>
</knowledge:articleList>

```

## Look at What We've Done

You can now see the Last Updated Promotions box on the left pane of the Visualforce Article Search page. This box displays any new or the most recently updated promotions.



# CHAPTER 4 Managing Articles Using Apex and REST APIs

## In this chapter ...

- [PublishingService Class](#)
- [Archive the Master Version of an Article](#)
- [Assign a Task Related to a Translation](#)
- [Assign a Task Related to a Master Article](#)
- [Delete a Master Version of an Article](#)
- [Delete Translated Version of an Article](#)
- [Edit an Online Version of a Master Article](#)
- [Publish the Master Version of an Article](#)
- [Restore an Archived Version of an Article](#)
- [Retrieve Article Metadata](#)
- [Retrieve a Version of an Article](#)
- [Search for Metadata Elements of a Master Version](#)
- [Search for Metadata Elements of a Translated Version](#)
- [Set a Translated Article Version to Complete](#)
- [Set a Translated Article Version to Incomplete](#)
- [Submit an Article for Translation](#)

## Using Apex or REST APIs to Manage Articles

---

This section describes the APIs that provide programmatic access to many essential actions you can perform on an article and its translations including:

- Publishing
- Updating
- Retrieving
- Deleting
- Submitting for translation
- Settings a translation to complete or incomplete status.

The first topic in this chapter describes the Apex Knowledge Management Publishing Service class. The following topics describe publishing actions that you can perform using the REST API.

For more information on using Apex to manage articles, see the [Apex Developer Guide](#). For more information on using REST to manage articles, see the [Force.com REST API Developer Guide](#).

- Unpublish the Master Version of an Article
- Unpublish the Online Version of a Translated Article
- Update View Statistics with the Salesforce Object Query Language
- Track Search Keywords with the Salesforce Object Search Language
- Update Article View Statistics with the Salesforce Object Search Language

# PublishingService Class

---

Use the methods in the `KbManagement.PublishingService` class to manage the lifecycle of an article and its translations.

## Namespace

`KbManagement`

## Usage

Use the methods in the `KbManagement.PublishingService` class to manage the following parts of the lifecycle of an article and its translations:

- Publishing
- Updating
- Retrieving
- Deleting
- Submitting for translation
- Setting a translation to complete or incomplete status
- Archiving
- Assigning review tasks for draft articles or translations

 **Note:** Date values are based on GMT.

To use the methods in this class, you must enable Salesforce Knowledge. See [Salesforce Knowledge Implementation Guide](#) for more information on setting up Salesforce Knowledge.

### IN THIS SECTION:

[PublishingService Methods](#)

## PublishingService Methods

The following are methods for `PublishingService`. All methods are static.

### IN THIS SECTION:

[archiveOnlineArticle\(articleId, scheduledDate\)](#)

Archives an online version of an article. If the specified `scheduledDate` is null, the article is archived immediately. Otherwise, it archives the article on the scheduled date.

[assignDraftArticleTask\(articleId, assigneeId, instructions, dueDate, sendEmailNotification\)](#)

Assigns a review task related to a draft article.

[assignDraftTranslationTask\(articleVersionId, assigneeId, instructions, dueDate, sendEmailNotification\)](#)

Assigns a review task related to a draft translation.

[cancelScheduledArchivingOfArticle\(articleId\)](#)

Cancels the scheduled archiving of an online article.

`cancelScheduledPublicationOfArticle(articleId)`

Cancels the scheduled publication of a draft article.

`completeTranslation(articleVersionId)`

Puts a translation in a completed state that is ready to publish.

`deleteArchivedArticle(articleId)`

Deletes an archived article.

`deleteArchivedArticleVersion(articleId, versionNumber)`

Deletes a specific version of an archived article.

`deleteDraftArticle(articleId)`

Deletes a draft article.

`deleteDraftTranslation(articleVersionId)`

Deletes a draft translation.

`editArchivedArticle(articleId)`

Creates a draft article from the archived master version and returns the new draft master version ID of the article.

`editOnlineArticle(articleId, unPublish)`

Creates a draft article from the online version and returns the new draft master version ID of the article. Also, unpublishes the online article, if `unPublish` is set to `true`.

`editPublishedTranslation(articleId, language, unPublish)`

Creates a draft version of the online translation for a specific language and returns the new draft master version ID of the article. Also, unpublishes the article, if set to `true`.

`publishArticle(articleId, flagAsNew)`

Publishes an article. If `flagAsNew` is set to `true`, the article is published as a major version.

`restoreOldVersion(articleId, versionNumber)`

Creates a draft article from an existing online article based on the specified archived version of the article and returns the article version ID.

`scheduleForPublication(articleId, scheduledDate)`

Schedules the article for publication as a major version. If the specified date is null, the article is published immediately.

`setTranslationToIncomplete(articleVersionId)`

Sets a draft translation that is ready for publication back to “in progress” status.

`submitForTranslation(articleId, language, assigneeId, dueDate)`

Submits an article for translation to the specified language. Also assigns the specified user and due date to the submittal and returns new ID of the draft translation.

**`archiveOnlineArticle(articleId, scheduledDate)`**

Archives an online version of an article. If the specified `scheduledDate` is null, the article is archived immediately. Otherwise, it archives the article on the scheduled date.

**Signature**

```
public static Void archiveOnlineArticle(String articleId, Datetime scheduledDate)
```

## Parameters

*articleId*

Type: String

*scheduledDate*

Type: Datetime

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';
Datetime scheduledDate = Datetime.newInstanceGmt(2012, 12, 1, 13, 30, 0);
KbManagement.PublishingService.archiveOnlineArticle(articleId, scheduledDate);
```

## **assignDraftArticleTask(articleId, assigneeId, instructions, dueDate, sendEmailNotification)**

Assigns a review task related to a draft article.

## Signature

```
public static Void assignDraftArticleTask(String articleId, String assigneeId, String instructions, Datetime dueDate, Boolean sendEmailNotification)
```

## Parameters

*articleId*

Type: String

*assigneeId*

Type: String

*instructions*

Type: String

*dueDate*

Type: Datetime

*sendEmailNotification*

Type: Boolean

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';
String assigneeId = '';
String instructions = 'Please review this draft.';
Datetime dueDate = Datetime.newInstanceGmt(2012, 12, 1);
KbManagement.PublishingService.assignDraftArticleTask(articleId, assigneeId, instructions,
    dueDate, true);
```

## **assignDraftTranslationTask(articleVersionId, assigneeId, instructions, dueDate, sendEmailNotification)**

Assigns a review task related to a draft translation.

## Signature

```
public static Void assignDraftTranslationTask(String articleVersionId, String assigneeId,
String instructions, Datetime dueDate, Boolean sendEmailNotification)
```

## Parameters

*articleVersionId*

Type: String

*assigneeId*

Type: String

*instructions*

Type: String

*dueDate*

Type: Datetime

*sendEmailNotification*

Type: Boolean

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';
String assigneeId = 'Insert assignee ID';
String instructions = 'Please review this draft.';
Datetime dueDate = Datetime.newInstanceGmt(2012, 12, 1);
KbManagement.PublishingService.assignDraftTranslationTask(articleId, assigneeId,
instructions, dueDate, true);
```

## **cancelScheduledArchivingOfArticle(articleId)**

Cancels the scheduled archiving of an online article.

## Signature

```
public static Void cancelScheduledArchivingOfArticle(String articleId)
```

## Parameters

*articleId*  
Type: String

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';  
KbManagement.PublishingService.cancelScheduledArchivingOfArticle (articleId);
```

## **cancelScheduledPublicationOfArticle (articleId)**

Cancels the scheduled publication of a draft article.

## Signature

```
public static Void cancelScheduledPublicationOfArticle(String articleId)
```

## Parameters

*articleId*  
Type: String

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';  
KbManagement.PublishingService.cancelScheduledPublicationOfArticle (articleId);
```

## **completeTranslation (articleVersionId)**

Puts a translation in a completed state that is ready to publish.

## Signature

```
public static Void completeTranslation(String articleVersionId)
```

## Parameters

*articleVersionId*  
Type: String

## Return Value

Type: Void

## Example

```
String articleVersionId = 'Insert article ID';  
KbManagement.PublishingService.completeTranslation(articleVersionId);
```

## **deleteArchivedArticle(articleId)**

Deletes an archived article.

## Signature

```
public static Void deleteArchivedArticle(String articleId)
```

## Parameters

*articleId*  
Type: String

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';  
KbManagement.PublishingService.deleteArchivedArticle(articleId);
```

## **deleteArchivedArticleVersion(articleId, versionNumber)**

Deletes a specific version of an archived article.

## Signature

```
public static Void deleteArchivedArticleVersion(String articleId, Integer versionNumber)
```

## Parameters

*articleId*  
Type: String

*versionNumber*  
Type: Integer

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';  
Integer versionNumber = 1;  
KbManagement.PublishingService.deleteArchivedArticleVersion(articleId, versionNumber);
```

## **deleteDraftArticle (articleId)**

Deletes a draft article.

## Signature

```
public static Void deleteDraftArticle(String articleId)
```

## Parameters

*articleId*  
Type: String

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';  
KbManagement.PublishingService.deleteDraftArticle(articleId);
```

## **deleteDraftTranslation (articleVersionId)**

Deletes a draft translation.

## Signature

```
public static Void deleteDraftTranslation(String articleVersionId)
```

## Parameters

*articleVersionId*  
Type: String

## Return Value

Type: Void

## Example

```
String articleVersionId = 'Insert article ID';  
KbManagement.PublishingService.deleteDraftTranslation (articleVersionId);
```

## **editArchivedArticle(articleId)**

Creates a draft article from the archived master version and returns the new draft master version ID of the article.

## Signature

```
public static String editArchivedArticle(String articleId)
```

## Parameters

*articleId*  
Type: String

## Return Value

Type: String

## Example

```
String articleId = 'Insert article ID';  
String id = KbManagement.PublishingService.editArchivedArticle(articleId);
```

## **editOnlineArticle(articleId, unpublish)**

Creates a draft article from the online version and returns the new draft master version ID of the article. Also, unpublishes the online article, if *unpublish* is set to `true`.

## Signature

```
public static String editOnlineArticle(String articleId, Boolean unpublish)
```

## Parameters

*articleId*  
Type: String

*unpublish*  
Type: Boolean

## Return Value

Type: String

## Example

```
String articleId = 'Insert article ID';  
String id = KbManagement.PublishingService.editOnlineArticle (articleId, true);
```

## **editPublishedTranslation(articleId, language, unpublish)**

Creates a draft version of the online translation for a specific language and returns the new draft master version ID of the article. Also, unpublishes the article, if set to `true`.

## Signature

```
public static String editPublishedTranslation(String articleId, String language, Boolean  
unpublish)
```

## Parameters

*articleId*

Type: String

*language*

Type: String

*unpublish*

Type: Boolean

## Return Value

Type: String

## Example

```
String articleId = 'Insert article ID';  
String language = 'fr';  
String id = KbManagement.PublishingService.editPublishedTranslation(articleId, language,  
true);
```

## **publishArticle(articleId, flagAsNew)**

Publishes an article. If *flagAsNew* is set to `true`, the article is published as a major version.

## Signature

```
public static Void publishArticle(String articleId, Boolean flagAsNew)
```

## Parameters

*articleId*

Type: String

*flagAsNew*

Type: Boolean

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';
KbManagement.PublishingService.publishArticle(articleId, true);
```

## **restoreOldVersion(articleId, versionNumber)**

Creates a draft article from an existing online article based on the specified archived version of the article and returns the article version ID.

## Signature

```
public static String restoreOldVersion(String articleId, Integer versionNumber)
```

## Parameters

*articleId*

Type: String

*versionNumber*

Type: Integer

## Return Value

Type: String

## Example

```
String articleId = 'Insert article ID';
String id = KbManagement.PublishingService.restoreOldVersion (articleId, 1);
```

## **scheduleForPublication(articleId, scheduledDate)**

Schedules the article for publication as a major version. If the specified date is null, the article is published immediately.

## Signature

```
public static Void scheduleForPublication(String articleId, Datetime scheduledDate)
```

## Parameters

*articleId*

Type: String

*scheduledDate*

Type: Datetime

## Return Value

Type: Void

## Example

```
String articleId = 'Insert article ID';
Datetime scheduledDate = Datetime.newInstanceGmt(2012, 12, 1, 13, 30, 0);
KbManagement.PublishingService.scheduleForPublication(articleId, scheduledDate);
```

## **setTranslationToIncomplete(articleVersionId)**

Sets a draft translation that is ready for publication back to “in progress” status.

## Signature

```
public static Void setTranslationToIncomplete(String articleVersionId)
```

## Parameters

*articleVersionId*

Type: String

## Return Value

Type: Void

## Example

```
String articleVersionId = 'Insert article ID';
KbManagement.PublishingService.setTranslationToIncomplete(articleVersionId);
```

## **submitForTranslation(articleId, language, assigneeId, dueDate)**

Submits an article for translation to the specified language. Also assigns the specified user and due date to the submittal and returns new ID of the draft translation.

## Signature

```
public static String submitForTranslation(String articleId, String language, String assigneeId, Datetime dueDate)
```

## Parameters

*articleId*  
Type: String

*language*  
Type: String

*assigneeId*  
Type: String

*dueDate*  
Type: Datetime

## Return Value

Type: String

## Example

```
String articleId = 'Insert article ID';
String language = 'fr';
String assigneeId = 'Insert assignee ID';
Datetime dueDate = Datetime.newInstanceGmt(2012, 12, 1);
String id = KbManagement.PublishingService.submitForTranslation(articleId, language,
assigneeId, dueDate);
```

# Archive the Master Version of an Article

Archives the master version of an article. The actions are defined by the field change you request on the resource. To archive the master version, use "publishStatus": "Archived". To schedule a date for archiving, use "archiveScheduleDate" : <date>.

## URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/<versionID>
```

## Formats

JSON, XML

## HTTP Method

PATCH

## Authentication

Authorization: OAuth *access\_token*

## Parameters

Parameter	Description
publishStatus	Publishing status of the article. Use archived.
archiveScheduleDate	Date to archive the article.

**Example request body**

Archive:

```
{
  "publishStatus": "Archived"
}
```

Schedule for archiving (using GMT date format):

```
{
  "archiveScheduleDate" : "2012-04-19T07:00:00.000+0000"
}
```

**Example response body**

HTTP status code 204 is returned when an existing record is updated.

## Assign a Task Related to a Translation

Assigns a task to a user for a translated article, including due date and instructions. The actions are defined by the field change you request on the resource.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations/<translationVersionId>
```

**Formats**

JSON, XML

**HTTP Method**

PATCH

**Authentication**Authorization: OAuth *access\_token***Parameters**

Parameter	Description
assigneeId	Assigns the master article to a user ID.
dueDate	Date that the task is due.
instruction	Instructions for the task.

**Example request body**

```
{
  "assigneeId": "05Dxx0000dsads"
  "dueDate": "2012-04-19T07:00:00.000+0000"
  "instruction": "Please review."
}
```

**Example response body**

HTTP status code 204 is returned when an existing record is updated.

## Assign a Task Related to a Master Article

---

Assigns a task to a user for a master article, including due date and instructions. The actions are defined by the field change you request on the resource.

### URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/<versionID>
```

### Formats

JSON, XML

### HTTP Method

PATCH

### Authentication

Authorization: OAuth *access\_token*

### Parameters

Parameter	Description
assigneeId	Assigns the master article to a user ID.
dueDate	Date that the task is due.
instruction	Instructions for the task.

### Example request body

```
{
  "assigneeId": "05Dxx0000dsads"
  "dueDate": "2012-04-19T07:00:00.000+0000"
  "instruction": "Please review."
}
```

### Example response body

HTTP status code 204 is returned when an existing record is updated.

## Delete a Master Version of an Article

---

Deletes the master version of an article.

### URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/<versionID>
```

### Formats

JSON, XML

### HTTP Method

DELETE

### Authentication

Authorization: OAuth *access\_token*

**Parameters**

None

**Example request body**

None required

**Example response body**

HTTP status code 204 is returned when an existing record is deleted.

## Delete Translated Version of an Article

---

Deletes a translated version of an article.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations/<versionID>
```

**Formats**

JSON, XML

**HTTP Method**

DELETE

**Authentication**Authorization: OAuth *access\_token***Parameters**

None

**Example request body**

None needed

**Example response body**

HTTP status code 204 is returned when an existing record is deleted.

## Edit an Online Version of a Master Article

---

Creates a draft copy of the online version of a master article. This does not unpublish the online version.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions
```

**Formats**

JSON, XML

**HTTP Method**

POST

**Authentication**Authorization: OAuth *access\_token***Parameters**

Parameter	Description
articleId	The ID of the article.

**Example request body**

```
{
  "articleId":<articleID>
}
```

## Publish the Master Version of an Article

---

Publishes the master version of an article. The publishing actions are defined by the field change you request on the resource. To publish a minor version, use "publishStatus": "Online". To publish a major version, use "publishStatus": "Online" and "versionNumber": "NextVersion". To schedule a publication date, use "publishScheduleDate" : <date>.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/<versionId>
```

**Formats**

JSON, XML

**HTTP Method**

PATCH

**Authentication**

Authorization: OAuth *access\_token*

**Parameters**

Parameter	Description
publishStatus	Publishing status of the article. Use <i>online</i> .
versionNumber	Version of the article.
publishScheduleDate	Date to publish the article.

**Example request body**

Publish a minor version:

```
{
  "publishStatus": "Online"
}
```

Publish a major version:

```
{
  "publishStatus": "Online"
  "versionNumber": "NextVersion"
}
```

Schedule for publication (using GMT date format):

```
{
  "publishScheduleDate" : "2012-05-19T07:00:00.000+0000"
}
```

**Example response body**

HTTP status code 204 is returned when an existing record is updated.

## Restore an Archived Version of an Article

---

Restores an archived version of the article. If `versionNumber` isn't specified, restores the latest version of the archived article.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/
```

**Formats**

JSON, XML

**HTTP Method**

POST

**Authentication**

Authorization: OAuth *access\_token*

**Parameters**

Parameter	Description
<code>articleId</code>	The ID of the article.
<code>versionNumber</code>	Version of the article. If this field is not specified, the latest version of the archived article is restored.

**Example request body**

```
{
  "articleId":<articleID>
  ("versionNumber":<number>)
}
```

## Retrieve Article Metadata

---

Retrieves the metadata of an article.

**URI**

```
/services/data/v25.0/knowledgeManagement/articles/<articleId>
```

**Formats**

JSON, XML

**HTTP Method**

GET

**Authentication**

Authorization: OAuth *access\_token*

**Parameters**

None

**Example request body**

None required

## Retrieve a Version of an Article

---

Retrieves the version ID of an article.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/<articleVersionId>
```

**Formats**

JSON, XML

**HTTP Method**

GET

**Authentication**Authorization: OAuth *access\_token***Parameters**

None

**Example request body**

None required

## Search for Metadata Elements of a Master Version

---

Searches for metadata elements of the online master version of an article.

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions?  
filterArticleId=value1&FilterPublishStatus=value2
```

**Formats**

JSON, XML

**HTTP Method**

GET

**Authentication**Authorization: OAuth *access\_token***Example**

This example searches for the online version of the master article 'kA0x5000000jsh':

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions?  
filterArticleId=kA0x5000000jsh&filterPublishStatus=online"
```

## Search for Metadata Elements of a Translated Version

---

Searches for metadata elements of a translated version of an article.

### URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations?filterArticleId=value1&filterLanguage=value2&FilterPublishStatus=value3";
```

### Formats

JSON, XML

### HTTP Method

GET

### Authentication

Authorization: OAuth *access\_token*

### Example

This example searches for the German online translation of the article 'kA0x5000000jsh':

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations?filterArticleId=kA0x5000000jsh&filterLanguage=de&filterPublishStatus=online"
```

## Set a Translated Article Version to Complete

---

Sets a translated article version to complete.

### URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations/<translationVersionID>
```

### Formats

JSON, XML

### HTTP Method

PATCH

### Authentication

Authorization: OAuth *access\_token*

### Parameters

Parameter	Description
complete	Set this value to <code>true</code> for translations that are complete.

### Example request body

```
{
  "complete": "true"
}
```

### Example response body

HTTP status code 204 is returned when an existing record is updated.

## Set a Translated Article Version to Incomplete

---

Sets a translated article version to incomplete.

### URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations/<translationVersionID>
```

### Formats

JSON, XML

### HTTP Method

PATCH

### Authentication

Authorization: OAuth *access\_token*

### Parameters

Parameter	Description
complete	Set this value to <code>false</code> to set a translation to incomplete.

### Example request body

```
{
  "complete": "false"
}
```

### Example response body

HTTP status code 204 is returned when an existing record is updated.

## Submit an Article for Translation

---

Submits an article for translation and keeps the online version. Copies or creates a new draft translation if there is already a translation in the specified language. The publishing actions are defined by the field change you request on the resource.

### URI

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations
```

### Formats

JSON, XML

### HTTP Method

POST

### Authentication

Authorization: OAuth *access\_token*

### Parameters

Parameter	Description
articleID	ID of the article

Parameter	Description
language	ISO code for the language

**Example request body**

```
{
  "articleId":<articleID>
  "language":"fr"
}
```

## Unpublish the Master Version of an Article

---

Unpublishes the online master version of an article when there isn't an existing draft article. The publishing actions are defined by the field change you request on the resource. To unpublish the master version, use "publishStatus": "draft".

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/masterVersions/<versionId>
```

**Formats**

JSON, XML

**HTTP Method**

PATCH

**Authentication**

Authorization: OAuth *access\_token*

**Parameters**

Parameter	Description
publishStatus	Publishing status of the article. Use <i>draft</i> .

**Example request body**

```
{
  "publishStatus":"draft"
}
```

**Example response body**

HTTP status code 204 is returned when an existing record is updated.

## Unpublish the Online Version of a Translated Article

---

Unpublishes the online version of a translated article. The publishing actions are defined by the field change you request on the resource. To edit and remove a translation from online state, use "publishStatus": "draft".

**URI**

```
/services/data/v25.0/knowledgeManagement/articleVersions/translations/<translationVersionID>
```

**Formats**

JSON, XML

**HTTP Method**

PATCH

**Authentication**Authorization: OAuth *access\_token***Parameters**

Parameter	Description
publishStatus	Publishing status of the article. Use <code>draft</code> .

**Example request body**

```
{
  "publishStatus": "draft"
}
```

## Update View Statistics with the Salesforce Object Query Language

The `UPDATE VIEWSTAT` clause is used in a `SELECT` statement to report on Salesforce Knowledge article searches and views. It allows developers to update an article's view statistics.

You can use this syntax to increase the view count for every article you have access to online:

```
SELECT Title FROM FAQ__kav
WHERE PublishStatus='online' and
Language = 'en_US' and
KnowledgeArticleVersion = 'ka230000000PCiy'
UPDATE VIEWSTAT
```

## Track Search Keywords with the Salesforce Object Search Language

The `UPDATE TRACKING` clause is used to report on Salesforce Knowledge article searches and views. It allows developers to track the keywords used in Salesforce Knowledge article searches. Also, the language attribute can be used to search by a specific language (locale). However, only one language can be specified in a single query. Make a separate query for each language that you want. Use the Java format, which uses the underscore (for example, `fr_FR`, `jp_JP`, and so on), to supply locales. Search the Web for “java locale codes” to get a list of supported locales.

You can use this syntax to track a keyword used in Salesforce Knowledge article search:

```
FIND {Keyword}
RETURNING KnowledgeArticleVersion (Title WHERE PublishStatus="Online" and language="en_US")
UPDATE TRACKING
```

## Update Article View Statistics with the Salesforce Object Search Language

---

The optional `UPDATE VIEWSTAT` clause is used to report on Salesforce Knowledge article searches and views. It allows developers to update an article's view statistics. Also, the `language` attribute can be used to search by a specific language (locale). However, only one language can be specified in a single query. Make a separate query for each language that you want. Use the Java format, which uses the underscore (for example, `fr_FR`, `jp_JP`, and so on), to supply locales. Search the Web for "java locale codes" to get a list of supported locales.

You can use this syntax to increase the view count for every article you have access to online in US English:

```
FIND {Title}
RETURNING FAQ__kav (Title WHERE PublishStatus="Online" and
language="en_US" and
KnowledgeArticleVersion = 'ka230000000PCiy')
UPDATE VIEWSTAT
```

# INDEX

## C

### Classes

KbManagement.PublishingService [38](#)

## K

### KbManagement.PublishingService

class [38](#)

### Knowledge

Apex sample code [36](#)

REST delete article version [51](#)

REST sample code [36](#)

## R

### REST

archive master version [49](#)

assign task to master article [51](#)

assign task to translation [50](#)

complete translated article version [56](#)

delete translated version [52](#)

### REST (*continued*)

edit online master [52](#)

publish master version [53](#)

restore article version [54](#)

retrieve article metadata [54](#)

retrieve article version [55](#)

search for master version metadata [55](#)

search for translated article metadata [56](#)

set translation to incomplete [57](#)

submit for translation [57](#)

unpublish master version [58](#)

unpublish online translation [58](#)

## S

### SOQL

UPDATE VIEWSTAT [59](#)

### SOSL

UPDATE TRACKING [59](#)

UPDATE VIEWSTAT [60](#)