Custom data source (Java)

PDF (/pdfs/kendra/latest/dg/kendra-dg.pdf#custom-java-sample) | RSS (amazon-kendra-release-notes.rss)

The following code provides a sample implementation of a custom data source using Java. The program first creates a custom data source and then uses that data source to add and remove documents from an index. Finally, it gets the metrics of the data source synchronization run.

The following code demonstrates creating and using a custom data source in the same sample. When you are using a custom data source in your application it isn't necessary to create a new data source each time that you synchronize your index.

```
import com.amazonaws.services.kendra.AWSkendra;
import com.amazonaws.services.kendra.AWSkendraClientBuilder;
import
com.amazonaws.services.kendra.model.BatchDeleteDocumentRequest;
import
com.amazonaws.services.kendra.model.BatchDeleteDocumentResult;
import com.amazonaws.services.kendra.model.BatchPutDocumentRequest;
import com.amazonaws.services.kendra.model.BatchPutDocumentResult;
import com.amazonaws.services.kendra.model.CreateDataSourceRequest;
import com.amazonaws.services.kendra.model.CreateDataSourceResult;
import
com.amazonaws.services.kendra.model.DataSourceSyncJobMetricTarget;
import com.amazonaws.services.kendra.model.DataSourceType;
import com.amazonaws.services.kendra.model.Document;
import com.amazonaws.services.kendra.model.DocumentAttribute;
import com.amazonaws.services.kendra.model.DocumentAttributeValue;
import
com.amazonaws.services.kendra.model.ListDataSourceSyncJobsRequest;
import
com.amazonaws.services.kendra.model.ListDataSourceSyncJobsResult;
import
com.amazonaws.services.kendra.model.StartDataSourceSyncJobRequest;
import
com.amazonaws.services.kendra.model.StartDataSourceSyncJobResult;
import
```

```
com.amazonaws.services.kendra.model.StopDataSourceSyncJobRequest;
import
com.amazonaws.services.kendra.model.StopDataSourceSyncJobResult;

public class SampleSyncForCustomDataSource {

  public static void main(String[] args) {

    final AWSkendra awskendraClient =
AWSkendraClientBuilder.standard().build();

    final String indexId = "Amazon Kendra index ID";

    // Create custom data source.
    final CreateDataSourceRequest createDataSourceRequest = new
CreateDataSourceRequest()
        .withName("sample-custom-data-source")
        .withType(DataSourceType.CUSTOM)
        .withDescription("description of sample-custom-data-
source")
        .withIndexId(indexId);
    final CreateDataSourceResult createDataSourceResult =
awskendraClient.createDataSource(createDataSourceRequest);

    // Get the data source id from createDataSourceResult.
    final String datasourceId = createDataSourceResult.getId();

    // Wait for the custom data source to become active.
    // You can use the DescribeDataSource API to check the status
    .
    .
    .
    .

    // Start the sync by calling StartDataSourceSync and providing
your index id
    // and your custom data source id
    final StartDataSourceSyncJobResult startDataSourceSyncJobResult
= awskendraClient.startDataSourceSyncJob(
        new StartDataSourceSyncJobRequest()
```

```
            .withIndexId(indexId)
            .withId(datasourceId)
    );
    final String executionId =
startDataSourceSyncJobResult.getExecutionId();

    // To associate documents with an synchronization run, add the
data source ID and
    // execution ID as attributes to the BatchPutDocument request.
The key for the
    // data source ID is "_data_source_id" and the key for the
execution run ID is
    // "_data_source_sync_job_execution_id".
    final BatchPutDocumentRequest batchPutDocumentRequest = new
BatchPutDocumentRequest()
        .withIndexId(indexId)
        .withDocuments(
            new Document()
                .withAttributes(
                    new DocumentAttribute()
                        .withKey("_data_source_id")
                        .withValue(
                            new DocumentAttributeValue()
                                .withStringValue(datasourceId)
                        ),
                    new DocumentAttribute()

.withKey("_data_source_sync_job_execution_id")
                        .withValue(
                            new DocumentAttributeValue()
                                .withStringValue(executionId)
                        )
                ).withId("first_document_id")
                .withBlob(..)
                .
                .
            , new Document()
                .withAttributes(
                    new DocumentAttribute()
                        .withKey("_data_source_id")
```

```
                        .withValue(
                            new DocumentAttributeValue()
                                .withStringValue(datasourceId)
                        ),
                    new DocumentAttribute()

.withKey("_data_source_sync_job_execution_id")
                        .withValue(
                            new DocumentAttributeValue()
                                .withStringValue(executionId)
                        )

                ).withId("second_document_id")
                .withBlob(..)
                .
                .
            , .....More documents

        );
    // Call the BatchPutRequest API.
    final BatchPutDocumentResult batchPutDocumentResult =
awskendraClient.batchPutDocument(batchPutDocumentRequest);

    // To delete documents, provide you custom data source ID and
job execution ID in the
    // DataSourceSyncJobMetrics parameter in the
BatchDeleteDocument request.
    BatchDeleteDocumentRequest batchDeleteDocumentRequest = new
BatchDeleteDocumentRequest()
        .withDocumentIdList(
            "id_of_first_document_to_delete",
            "id_of_second_document_to_delete",
            "id_of_third_document_to_delete",
            .
            .
            .
        )
        .withDataSourceSyncJobMetricTarget(
            new DataSourceSyncJobMetricTarget()
                .withDataSourceSyncJobId(executionId)
```

```
                .withDataSourceId(datasourceId)
        )
        .withIndexId(indexId);
    // Make BatchPutRequest call.
    final BatchDeleteDocumentResult batchDeleteDocumentResult =
awskendraClient.batchDeleteDocument(batchDeleteDocumentRequest);

    // Repeat BatchPutDocument and BatchDeleteDocument requests for
all the documents in your
    // repository to sync with Amazon Kendra.

    // After you are finished, call the StopDataSourceSyncJob API
to signal the end of the sync job.
    final StopDataSourceSyncJobResult stopDataSourceSyncJobResult =
awskendraClient.stopDataSourceSyncJob(
        new StopDataSourceSyncJobRequest()
            .withIndexId(indexId)
            .withId(datasourceId)
    );

    // After you call the StopDataSourceSyncJob API, you can start
another sync job.
    // You can't use the BatchPutDocument or BatchDeleteDocument
API requests with a
    // stopped job execution ID.

    // It can take time to index all of the documents submitted.
Use the ListDataSourceSyncJobs
    // API to get the status of a sync job and number of documents
added, modified, failed
    // or deleted as part of this sync with your Amazon Kendra
index.

    // If the sync job status is SYNCING_INDEXING, documents are
still being indexed.

    // Jobs are sorted in reverse order of their start time with
the most recent first.
    final ListDataSourceSyncJobsResult listDataSourceSyncJobsResult
= awskendraClient.listDataSourceSyncJobs(
```

```
        new ListDataSourceSyncJobsRequest()
            .withIndexId(indexId)
            .withId(datasourceId)
    );
  }

}
```

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.