Let's create a Spring Boot application to connect to Azure Blob Storage using a SAS token and list blobs in a container. Here's a complete example:

* First, add the required dependency to your pom.xml:

xml

<dependency>

<groupId>com.azure</groupId>

<artifactId>azure-storage-blob</artifactId>

<version>12.27.2</version>

</dependency>

* Create a configuration class to set up the Azure Blob Storage client:

java

package com.example.azureblobdemo.config;

import com.azure.storage.blob.BlobServiceClient;

import com.azure.storage.blob.BlobServiceClientBuilder;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AzureBlobConfig {

@Value("${azure.storage.account-name}")

private String accountName;

@Value("${azure.storage.sas-token}")

private String sasToken;

@Value("${azure.storage.endpoint}")

private String endpoint;

@Bean

public BlobServiceClient blobServiceClient() {

return new BlobServiceClientBuilder()

.endpoint(endpoint)

.sasToken(sasToken)

.buildClient();

}

}

* Create a service class to handle blob operations:

java

package com.example.azureblobdemo.service;

import com.azure.storage.blob.BlobContainerClient;

import com.azure.storage.blob.BlobServiceClient;

import com.azure.storage.blob.models.BlobItem;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

public class AzureBlobService {

private final BlobServiceClient blobServiceClient;

@Value("${azure.storage.container-name}")

private String containerName;

public AzureBlobService(BlobServiceClient blobServiceClient) {

this.blobServiceClient = blobServiceClient;

}

public List<String> listBlobs() {

BlobContainerClient containerClient = blobServiceClient.getBlobContainerClient(containerName);

List<String> blobNames = new ArrayList<>();

for (BlobItem blobItem : containerClient.listBlobs()) {

blobNames.add(blobItem.getName());

}

return blobNames;

}

}

* Create a controller to expose the blob listing endpoint:

java

package com.example.azureblobdemo.controller;

import com.example.azureblobdemo.service.AzureBlobService;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/api/blobs")

public class BlobController {

private final AzureBlobService azureBlobService;

public BlobController(AzureBlobService azureBlobService) {

this.azureBlobService = azureBlobService;

}

@GetMapping

public List<String> listBlobs() {

return azureBlobService.listBlobs();

}

}

* Add configuration properties to application.yml:

yaml

azure:

storage:

account-name: your-account-name

container-name: your-container-name

endpoint: https://${your-account-name}.blob.core.windows.net

sas-token: ?your-sas-token-here

* Create the main Spring Boot application class:

java

package com.example.azureblobdemo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AzureBlobDemoApplication {

public static void main(String[] args) {

SpringApplication.run(AzureBlobDemoApplication.class, args);

}

}

Important notes:

* SAS Token:
  + Generate the SAS token from the Azure Portal with appropriate permissions (at least 'Read' and 'List' for listing blobs)
  + The SAS token should start with a '?' character
  + Keep the SAS token secure and don't hardcode it in the code
* Security:
  + Store sensitive information like SAS token in environment variables or a secure vault
  + Consider using Azure Key Vault for production environments
  + Ensure the SAS token has the minimum required permissions
* Error Handling: You might want to add proper error handling. Here's an enhanced version of the service method:

java

public List<String> listBlobs() {

try {

BlobContainerClient containerClient = blobServiceClient.getBlobContainerClient(containerName);

if (!containerClient.exists()) {

throw new IllegalStateException("Container " + containerName + " does not exist");

}

List<String> blobNames = new ArrayList<>();

for (BlobItem blobItem : containerClient.listBlobs()) {

blobNames.add(blobItem.getName());

}

return blobNames;

} catch (Exception e) {

throw new RuntimeException("Failed to list blobs: " + e.getMessage(), e);

}

}

* Testing:
  + Start the application
  + Access the endpoint: GET http://localhost:8080/api/blobs
  + You should see a list of blob names in the specified container
* Additional Dependencies: If you're using Spring Boot, ensure you have these basic dependencies in your pom.xml:

xml

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.4</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

* Permissions:
  + Ensure your SAS token has permissions for:
    - Service: Blob
    - Resource types: Container, Object
    - Permissions: Read, List
    - Appropriate start/expiry dates

To use this code:

* Replace your-account-name, your-container-name, and your-sas-token-here with your actual Azure Storage account details
* Ensure the container exists in your Azure Storage account
* Make sure the SAS token is valid and has the necessary permissions
* The endpoint /api/blobs will return a list of blob names in the specified container

This code provides a basic implementation for listing blobs. You can extend it to include additional functionality like uploading, downloading, or deleting blobs based on your needs.

Explain SAS token security

Azure Key Vault integration