

**Creating and Using Server-Side Scripts**

Template Version: 2.0

**Introduction**

During this lab, you will learn how to create and call stored procedures, triggers, and user defined functions (UDF).

**Estimated Time**

20 minutes.

**Objectives**

At the end of this lab, you will be able to:

         Create server-side scripts

         Understand development best practices

         Use server-side scripts

Lab: Creating and Using Server-Side Scripts

**Exercise 1: Start the Server-Side Scripts Solution**

This exercise shows you how to view, create, and use server-side scripts using the provided java solution.

**Tasks**

**Open your java solution**

1. Open the **server-side-scripts** solution in your IDE or text editor and edit the **com.microsoft.azure.cosmosdb.sample.AccountSettings.java** file.
2. Change the **ACCOUNT\_HOST** propertyvalue to the **endpoint URI** you recorded in Exercise 1.
3. Change the **ACCOUNT\_KEY** propertyvalue to the **Primary Key** you recorded in Exercise 1.

**Examine the project files**

1. Notice the **RunSimpleScript** method call on the **com.microsoft.azure.cosmosdb.sample.ScriptManager.java** class.
2. The method is has the steps required to create and execute a stored procedure using a server-side script. These steps include:
   1. Create a stored procedure from the js script.
   2. Create a sample document to be used in the stored proc logic.
   3. Run the stored procedure and output the results.
3. The server-side script details are stored in the **SimpleScript.js** file. You can view the .js file by navigating to the **src/main/Resources/JS** folder.
4. Within the **SimpleScript.js** file, review the comments and best practices.

**Exercise 2: Complete the Server-Side Scripting Solution**

This method will create a stored procedure that simply queries all records and returns the first one. The return is pre-fixed with a user-defined string that is passed in as a parameter. It will then create and upload the document that it will query. Finally, it will call the stored procedure with the pre-fix “Hello”.

**Tasks**

**Create a stored procedure from the js script**

1. Access the SimpleScripts.js file by using a class loader and passing the relative path:

ClassLoader classLoader = getClass().getClassLoader();

File file = new File(classLoader.getResource("JS/SimpleScript.js").getFile());

1. Next, create variables for the scripts Id and contents. It is import to encode with UTF 8 in order to avoid invalid characters in the generated script:

String scriptBody = new String(Files.readAllBytes(Paths.get(file.getPath())), StandardCharsets.UTF\_8);

String scriptId = file.getName();

1. Now create the stored procedure object, passing in the scriptId and scriptBody.:

StoredProcedure sproc = new StoredProcedure();

sproc.setId(FilenameUtils.removeExtension(scriptId));

sproc.setBody(scriptBody);

1. Finally, create the stored procedure:

client.createStoredProcedure(collectionLink, sproc, null)

.subscribe(spResultPage -> {

System.out.println("\n" + "Stored procedure created ");

});

**Create a sample document to be used in the stored proc logic**

1. We will need to create a document to run our stored procedure against. Add a simple Json string and pass it to a new document object:

String jsonString = "{'LastName':'Estel','Headquarters':'Russia','Locations': {'Country':'Russia','City':'Novosibirsk'},'Income':'50000'}";

Document document = new Document(jsonString);

1. Use the AsyncClient to write the document to the CosmosDB collection and output the activity Id:

client.createDocument(collectionLink, document, null, false).subscribe(documentResourceResponse -> {

System.out.println("Document created " + documentResourceResponse.getActivityId() + "\n");

});

**Run the script. Pass "Hello, " as parameter**

1. Create an Object array to store our stored procedure arguments and pass it the string “Hello”:

Object[] storedProcedureArgs = new Object[] { "Hello" };

1. Add code to execute the stored procedure with the arguments specified above:

client.executeStoredProcedure(collectionLink + "/sprocs/" + sproc.getId(), storedProcedureArgs)

.subscribe(storedProcedureResponse -> {

System.out.println(storedProcedureResponse.getResponseAsString());

});

1. Finally package and run the solution by using Maven commands:

mvn clean package

mvn exec: java

1. Observe the output of the stored procedure logic. The contents of first document in the collection is printed with the stored procedure argument prefixed.
2. Navigate to the Azure Portal and the newly created stored procedure and document.

**Code Review**

The code in this project has been extensively commented to help developers understand the best practices at use. Please take the time to review both the Program.cs files and the JavaScript files to view the samples and learn more.