JSON FILES Read and Post Data to SQL SERVER Using ADO.NET

StartUp.cs:

public StartUp(IConfiguration configuration, IWebHostEnvironment env )

{

Configuration = configuration;

\_env = env;

}

public void ConfigureServices(IServiceCollection services)

{

// read jsonfile config:

// ConfigJsonFiles

// For RSAClientData.json

var path = Path.Combine(Environment.CurrentDirectory, @"JSONFiles\");

IConfiguration config = new ConfigurationBuilder()

.SetBasePath(path)

.AddJsonFile("RSAClientData.json")

.AddJsonFile("Applied.json")

.Build();

// configure the RSAClient and Coupon Data

var rSAClients = config.GetSection("RSAClientData").Get<List<RSAClient>>();

var coupons = config.GetSection("NewApplied").Get<List<Coupon>>();

// // Register the list of items as a singleton

services.AddSingleton(rSAClients);

services.AddSingleton(coupons);}

Create json file into your project folder:

JSONFiles

--- RSAClinet.json

--- RSAClinetData.json

JSONFiles\\RSAClient.json:

NOTE: we con not add this type of json file into StartUp.cs. So we have to use

System.IO.File.ReadAllText("D:\\csharpprojects\\JWTRoleAuthentication\\JWTRoleAuthentication\\JSONFiles\\RSAClient.json");

[

{

"RSAClientId": 1,

"RSAClientName": "Client one",

"Stores": "store one",

"UserName": "pusamsaikumar@gmail.com",

"Email": "pusamsaikumar@gmail.com",

"MobileNumber": "9959608677",

"Address": "Manikonda",

"SharedKey": "saikumar123",

"SecretKey": "1994sai",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 2,

"RSAClientName": "Client two",

"Stores": "store two",

"UserName": "vinay@gmail.com",

"Email": "vinay@gmail.com",

"MobileNumber": "9959608678",

"Address": "Khammam",

"SharedKey": "vinay127",

"SecretKey": "vinay123",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 3,

"RSAClientName": "Client three",

"Stores": "store three",

"UserName": "vijay@gmail.com",

"Email": "vijay@gmail.com",

"MobileNumber": "9959608679",

"Address": "Hyderabad",

"SharedKey": "vijay123",

"SecretKey": "vijaypusam",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 4,

"RSAClientName": "Client four",

"Stores": "store four",

"UserName": "ashok@gmail.com",

"Email": "ashok@gmail.com",

"MobileNumber": "9959608676",

"Address": "Hyderabad",

"SharedKey": "ashok123",

"SecretKey": "ashoknupa",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 5,

"RSAClientName": "Client five",

"Stores": "store five",

"UserName": "rajesh@gmail.com",

"Email": "rajesh@gmail.com",

"MobileNumber": "9959608678",

"Address": "Miyapur",

"SharedKey": "rajesh123",

"SecretKey": "rajeshnupa",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

}

]

--- RSAClientData.json:

NOTE: We have add into AddJsonFile() into the StartUp.cs

{

"RSAClientData": [

{

"RSAClientId": 1,

"RSAClientName": "Client one",

"Stores": "store one",

"UserName": "pusamsaikumar@gmail.com",

"Email": "pusamsaikumar@gmail.com",

"MobileNumber": "9959608677",

"Address": "Manikonda",

"SharedKey": "saikumar123",

"SecretKey": "1994sai",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 2,

"RSAClientName": "Client two",

"Stores": "store two",

"UserName": "vinay@gmail.com",

"Email": "vinay@gmail.com",

"MobileNumber": "9959608678",

"Address": "Khammam",

"SharedKey": "vinay127",

"SecretKey": "vinay123",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 3,

"RSAClientName": "Client three",

"Stores": "store three",

"UserName": "vijay@gmail.com",

"Email": "vijay@gmail.com",

"MobileNumber": "9959608679",

"Address": "Hyderabad",

"SharedKey": "vijay123",

"SecretKey": "vijaypusam",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 4,

"RSAClientName": "Client four",

"Stores": "store four",

"UserName": "ashok@gmail.com",

"Email": "ashok@gmail.com",

"MobileNumber": "9959608676",

"Address": "Hyderabad",

"SharedKey": "ashok123",

"SecretKey": "ashoknupa",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

},

{

"RSAClientId": 5,

"RSAClientName": "Client five",

"Stores": "store five",

"UserName": "rajesh@gmail.com",

"Email": "rajesh@gmail.com",

"MobileNumber": "9959608678",

"Address": "Miyapur",

"SharedKey": "rajesh123",

"SecretKey": "rajeshnupa",

"ServerName": "DESKTOP-1JNARAI\\SAISQLSERVER",

"DatabaseName": "AuthenticationDB",

"UserId": "sa",

"Password": "sai123"

}

]

}

FilesController.cs:

using JWTRoleAuthentication.CommonLayer.Models;

using JWTRoleAuthentication.JWTBLL;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

namespace JWTRoleAuthentication.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class FilesController : ControllerBase

{

private readonly List<RSAClient> \_rSAClients;

private readonly IFileService \_fileService;

public FilesController(

List<RSAClient> rSAClients,

IFileService fileService

)

{

\_rSAClients = rSAClients;

\_fileService = fileService;

}

[HttpGet]

[Route("ReadJsonFile")]

public async Task<IActionResult> ReadJsonFile(string sharedKey, string secretKey)

{

var result = await \_fileService.ReadJsonFile(sharedKey, secretKey);

if (result == null)

{

return NotFound(new {message = result?.StatusMessage});

}

return Ok(result);

}

public IActionResult ReadJsonFile(string sharedKey,string secretKey)

{

// we have to read json file like below:

var json = System.IO.File.ReadAllText("D:\\csharpprojects\\JWTRoleAuthentication\\JWTRoleAuthentication\\JSONFiles\\RSAClient.json");

var jsonData = JsonConvert.DeserializeObject<RSAClientDataWrapper>(json);

var data = jsonData?.RSAClientData.FirstOrDefault(item => item.SharedKey == sharedKey && item.SecretKey == secretKey);

// var model = new RSAClient

// {

// RSAClientId = data.RSAClientId,

// RSAClientName = data?.RSAClientName,

// UserName = data?.UserName,

// Email = data?.Email,

// MobileNumber = data?.MobileNumber,

// Address = data?.Address

// };

// var connectionStringBuilder = new AuthDBCon

// {

// DataSource = data.ServerName,

// InitialCatalog = data.DatabaseName,

// UserID = data.UserId,

// Password = data.Password,

// };

// var connectionstring = connectionStringBuilder.BuildConnectionString();

// Console.WriteLine(connectionstring);

// return Ok(model);

//}

[HttpPost]

[Route("CreateCustomerwithJsonfile")]

public async Task<IActionResult> AddCustomer(string sharedKey,string secretKey,CustomerModel model)

{

var result = await \_fileService.AddCustomerWithJsonfile(sharedKey, secretKey, model);

if(result == null)

{

return NotFound();

}

return Ok(result);

}

[HttpPost]

[Route("LoginWithJSONFile")]

public async Task<IActionResult> LoginWithJSONFile(string sharedKey,string secretKey,LoginModel model)

{

var result = await \_fileService.LoginWithJSONFile(sharedKey, secretKey, model);

if (result == null)

{

return NotFound();

}

return Ok(result);

}

}

}

FileAuthRepo.cs:

using JWTRoleAuthentication.CommonLayer.Models;

using Newtonsoft.Json;

using System.Data.SqlClient;

namespace JWTRoleAuthentication.JWTDAL

{

public class FileRepo : IFileRepo

{

private readonly List<RSAClient> \_rSAClients;

private readonly IAuthRepo \_authRepo;

private readonly Helpers \_helpers;

private readonly IHttpContextAccessor \_httpContextAccessor;

public FileRepo(

List<RSAClient> rSAClients,

IAuthRepo authRepo,

Helpers helpers,

IHttpContextAccessor httpContextAccessor

)

{

\_rSAClients = rSAClients;

\_authRepo = authRepo;

\_helpers = helpers;

\_httpContextAccessor = httpContextAccessor;

}

public async Task<CustomerResponse> AddCustomerWithJsonfile(string sharedKey, string secretKey, CustomerModel model)

{

CustomerResponse customerResponse = null;

customerResponse = new CustomerResponse();

try

{

var jsonData = await ReadJsonFile(sharedKey,secretKey);

model.UserName = jsonData.RSAClient.UserName;

model.Email = jsonData.RSAClient.Email ;

model.MobileNumber = jsonData.RSAClient.MobileNumber;

model.Address = jsonData.RSAClient.Address;

Console.WriteLine(model);

//var authCon = new AuthDBCon();

//if(jsonData?.RSAClient != null)

//{

// authCon.DataSource = jsonData.RSAClient.ServerName;

// authCon.InitialCatalog = jsonData.RSAClient.DatabaseName;

// authCon.UserID = jsonData.RSAClient.UserId;

// authCon.Password = jsonData.RSAClient.Password;

//}

//string connectionString = authCon.BuildConnectionString();

string connectionString = await \_helpers.BuildConnectionString(jsonData?.AuthDBCon);

Console.WriteLine(connectionString);

SqlHelpers sqlHelpers = new SqlHelpers(connectionString);

var storedProcName = "InsertCustomer";

// USING SQL COMMANDS:

var parameters = new SqlParameter[]

{

new SqlParameter("@UserName",model.UserName),

new SqlParameter("@Email",model.Email),

new SqlParameter("@MobileNumber",model.MobileNumber),

new SqlParameter("@Address",model.Address),

};

int rowsAffected = await sqlHelpers.InsertTable(storedProcName, parameters);

if(rowsAffected > 0)

{

customerResponse.StatusCode = 200;

customerResponse.StatusMessage = "Customer record inserted successfully";

customerResponse.Customer = model;

return customerResponse;

}

customerResponse.StatusCode = 400;

customerResponse.StatusMessage = "failed";

// customerResponse.Customer = model;

}

catch (Exception ex)

{

customerResponse.StatusCode = 500;

customerResponse.StatusMessage = "Something went wrong. Please try again.";

}

return customerResponse;

}

public async Task<LoginResponse> LoginWithJSONFile(string sharedKey, string secretKey, LoginModel model)

{

LoginResponse response = null;

response = new LoginResponse();

try

{

var jsonData = await ReadJsonFile(sharedKey,secretKey);

string connectionString = await \_helpers.BuildConnectionString(jsonData?.AuthDBCon);

SqlHelpers sqlHelpers = new SqlHelpers(connectionString);

var stroedProcName = "Check\_UserName\_Password";

int existedUserNamePassword = 0;

var parameters = new SqlParameter[]

{

new SqlParameter("@UserName",model.UserName),

new SqlParameter("@Password",\_helpers.EncryptedPassword(model.Password))

};

existedUserNamePassword = await sqlHelpers.ExecuteIntScalar(stroedProcName,parameters);

if(existedUserNamePassword > 0)

{

var getUser = await \_authRepo.GetUserDetails(model.UserName);

var user = await \_authRepo.GetTokensFromDB(model.UserName);

response.TokenModel = new TokenModel();

if (\_helpers.IsRefreshTokenExpired(user.Register.RefreshToken.ToString()))

{

response.TokenModel.RefreshToken = \_helpers.GenerateRefreshToken();

response.TokenModel.Token = \_helpers.GenerateJwtToken(getUser.Register.UserName, getUser.Register.Email, getUser.Register.StoreID, getUser.Register.DateOfBirth, getUser.Register.Role);

await \_authRepo.UpdateTokenToDB(response.TokenModel, getUser.Register.UserID.ToString().ToUpper());

\_httpContextAccessor?.HttpContext?.Response.Headers.Add("AccessToken", response.TokenModel.Token);

response.StatusCode = 200;

response.StatusMessage = "User loggedin successfully.";

return response;

}

response.TokenModel.Token = user.Register.Token.ToString();

response.StatusCode = 200;

response.StatusMessage = "User loggedin successfully.";

\_httpContextAccessor?.HttpContext?.Response.Headers.Add("AccessToken", response.TokenModel.Token);

}

else

{

response.StatusCode = 400;

response.StatusMessage = "User login has been failed.";

}

}

catch (Exception ex)

{

response.StatusCode = 500;

response.StatusMessage = "Something went wrong. Please try again.";

}

return response;

}

public async Task<RSAClinetResponse> ReadJsonFile(string sharedKey, string secretKey)

{

RSAClinetResponse response = null;

response = new RSAClinetResponse();

try

{

// Read from local folder json file:

var fileName = @"RSAClientData.json";

var currentDirectory = Directory.GetCurrentDirectory();

string[] fullFilePath = Directory.GetFiles(currentDirectory, fileName,SearchOption.AllDirectories);

if(fullFilePath.Length > 0)

{

using(StreamReader reader = new StreamReader(fullFilePath[0]))

{

string jsonStringData = reader.ReadToEnd();

var dataResult = JsonConvert.DeserializeObject<RSAClientDataWrapper>(jsonStringData);

Console.WriteLine(dataResult?.RSAClientData);

}

}

var json = await File.ReadAllTextAsync("D:\\csharpprojects\\JWTRoleAuthentication\\JWTRoleAuthentication\\JSONFiles\\RSAClientData.json");

var jsonData = JsonConvert.DeserializeObject<RSAClientDataWrapper>(json);

var data = jsonData?.RSAClientData.FirstOrDefault(item => item.SharedKey == sharedKey && item.SecretKey == secretKey);

if (data != null)

{

response.RSAClient = new RSAClient

{

RSAClientId = data.RSAClientId,

RSAClientName = data.RSAClientName,

Stores = data.Stores,

UserName = data.UserName,

Email = data.Email,

MobileNumber = data.MobileNumber,

Address = data.Address,

SharedKey = data.SharedKey,

SecretKey = data.SecretKey,

ServerName = data.ServerName,

DatabaseName = data.DatabaseName,

UserId = data.UserId,

Password = data.Password,

};

response.AuthDBCon = new AuthDBCon

{

DataSource = data.ServerName,

InitialCatalog = data.DatabaseName,

UserID = data.UserId,

Password = data.Password,

};

response.StatusCode = 200;

response.StatusMessage = "OK";

return response;

}

response.StatusCode = 404;

response.StatusMessage = "Json file data not found.";

}

catch (Exception ex)

{

response.StatusCode = 500;

response.StatusMessage = "Something went wrong. Please try again.";

}

return response;

}

}

}

Models:

namespace JWTRoleAuthentication.CommonLayer.Models

{

public class RSAClient

{

public int RSAClientId { get; set; }

public string RSAClientName { get; set; }

public string Stores { get;set; }

public string UserName { get; set; }

public string Email { get; set; }

public string MobileNumber { get; set; }

public string Address { get; set; }

public string SharedKey { get; set; }

public string SecretKey { get; set; }

public string ServerName { get; set; }

public string DatabaseName { get; set; }

public string UserId { get; set; }

public string Password { get; set; }

}

public class RSAClientData

{

public List<RSAClient> RSAClients { get; set;}

}

public class RSAClientDataWrapper

{

public List<RSAClient> RSAClientData { get; set; }

}

public class RSAClinetResponse

{

public int StatusCode { get; set; }

public string StatusMessage { get; set; }

public RSAClient RSAClient { get; set; }

public AuthDBCon AuthDBCon { get; set; }

}

}

SqlHelper.cs:

private readonly string \_connectionString;

private SqlConnection \_connection;

public SqlHelpers(string connectionString)

{

\_connectionString = connectionString;

\_connection = new SqlConnection(connectionString);

}

//public void Dispose()

//{

// if (\_connection != null)

// {

// if(\_connection.State != System.Data.ConnectionState.Closed)

// {

// \_connection.Close();

// }

// \_connection.Dispose();

// \_connection = null;

// }

//}

public async ValueTask DisposeAsync()

{

if (\_connection != null)

{

if (\_connection.State != System.Data.ConnectionState.Closed)

{

await \_connection.CloseAsync();

}

\_connection.Dispose();

\_connection = null;

}

}

public async Task EnsureConnection()

{

if (\_connection != null)

{

if(\_connection.State != ConnectionState.Open)

{

await \_connection.OpenAsync();

}

}

}

// ExecuteNonQuery: for Update and Insert Data Records

// using following methods:

public async Task<int> ExecuteNonQueryAsync(SqlCommand command)

{

await EnsureConnection();

return await command.ExecuteNonQueryAsync();

}

public async Task<int> InsertTable(string storedProcName,params SqlParameter[] parameters)

{

await EnsureConnection();

using (var command = CreateStoredProcedureCommand(storedProcName))

{

if(parameters != null)

{

command.Parameters.AddRange(parameters);

}

int rowAffected = await ExecuteNonQueryAsync(command);

return rowAffected;

}

}

Add JSON File with Array with out Object Name

If your jsonfile like this below:

[

{

"SharedKey": "rsaecrsapi",

"SecretKey": "rsaecrsapi123",

"RetailerName": "",

"RemoveCheckDigitFromBasketUPC": "0",

"MemberNumberLength": "10",

"MaxCouponValueForClient": "200",

"RSACoreAPIUrl": "",

"SystemNotificationsARN": "",

"EnableRemoveNCRImpression": "0",

"EnterpriseId": "",

"EnterpriseSecret": "",

"NCRImpressionAPIUrl": "",

"UPCCouponTypeIds": "5,12,13",

"CrossSellCouponTypeIds": "7,15",

"MFRUPCCouponTypeIds": "4",

"SetCouponMultiLineIdsToZero": "1",

"CrossSellCouponsPurchaseUPCsOrderByAmount": "ASC",

"EnableWriteLogs": "1",

"EnableEmailNotificationAlerts": "1",

"SendTransactionsToCloud": "1",

"POSType": "ECRS",

"DBInstanceName": "",

"DBName": "",

"CloudRetailerName": "",

"RetailerTimeZone": "CST",

"SendTransactionToCloudURL": "",

"SendCommitToCloudURL": "",

"GetTransactionCloudURL": "",

"MoveTransactionToProcessedFolderCloudURL": "",

"RetailerConnectionString": "Data Source=DESKTOP-1JNARAI\\SAISQLSERVER;Initial Catalog=RSAECRSAPIDB;uid=sa;Password=sai123",

"SqlCommandTimeout": "7",

"ApplyOnlyOneBasketCouponFlag": "1",

"SaveRealtimeTransactionData": "1"

},

{

"SharedKey": "sai123",

"SecretKey": "sai12345",

"RetailerName": "",

"RemoveCheckDigitFromBasketUPC": "0",

"MemberNumberLength": "10",

"MaxCouponValueForClient": "200",

"RSACoreAPIUrl": "",

"SystemNotificationsARN": "",

"EnableRemoveNCRImpression": "0",

"EnterpriseId": "",

"EnterpriseSecret": "",

"NCRImpressionAPIUrl": "",

"UPCCouponTypeIds": "5,12,13",

"CrossSellCouponTypeIds": "7,15",

"MFRUPCCouponTypeIds": "4",

"SetCouponMultiLineIdsToZero": "1",

"CrossSellCouponsPurchaseUPCsOrderByAmount": "ASC",

"EnableWriteLogs": "1",

"EnableEmailNotificationAlerts": "1",

"SendTransactionsToCloud": "1",

"POSType": "ECRS",

"DBInstanceName": "",

"DBName": "",

"CloudRetailerName": "",

"RetailerTimeZone": "CST",

"SendTransactionToCloudURL": "",

"SendCommitToCloudURL": "",

"GetTransactionCloudURL": "",

"MoveTransactionToProcessedFolderCloudURL": "",

"RetailerConnectionString": "Data Source=DESKTOP-1JNARAI\\SAISQLSERVER;Initial Catalog=RsaecrsApiDB02;uid=sa;Password=sai123",

"SqlCommandTimeout": "7",

"ApplyOnlyOneBasketCouponFlag": "1",

"SaveRealtimeTransactionData": "1"

}

]

StartUp.cs:

public void ConfigureServices(IServiceCollection services)

{

var path = Path.Combine(Environment.CurrentDirectory, "AppConfigurations.json");

var json = File.ReadAllText(path);

var appConfigurations = JsonConvert.DeserializeObject<List<AppConfigurations>>(json);

services.AddSingleton(appConfigurations);

}

Using json file data from StartUp.cs:

public class S3Repo : IS3Repo

{

private readonly IAmazonS3 \_amazonS3;

private readonly List<AppConfigurations> \_appConfigurations;

private readonly S3BucketHelpers \_s3BucketHelpers;

public S3Repo(

IAmazonS3 amazonS3,

List<AppConfigurations> appConfigurations,

S3BucketHelpers s3BucketHelpers

)

{

\_amazonS3 = amazonS3;

\_appConfigurations = appConfigurations;

\_s3BucketHelpers = s3BucketHelpers;

}

public async Task<UpdateTransactionResponse> transaction(string site, string customer, string transaction, UpdateTransactionRequest TranRequest)

{

UpdateTransactionResponse response = new UpdateTransactionResponse();

try

{

// using like this

var appConfigurationData = \_appConfigurations.ToArray();

AppConfigurations appConfigurations = new AppConfigurations();

// appConfigurations = appConfigurationData.FirstOrDefault(x => x.SharedKey == TranRequest.sharedKey && x.SecretKey == TranRequest.Secret);

var jsonData = \_appConfigurations.ToArray().FirstOrDefault(x => x.SharedKey == TranRequest.sharedKey && x.SecretKey == TranRequest.Secret);

TransactionData transactionData = null;

}

}