Call from Service to return List

/// <summary>

/// Get List of UserRoles

/// </summary>

/// <param name="null"> <see cref="UserRoles"/></param>

/// <returns>List <see cref="List<contractor>"/>.</returns>

public async Task<List<UserRoles>> GetUserRoles()

{

try

{

int Id = 1;

SqlParameter[] parameters = {

new SqlParameter("@Id", Id)

};

List<UserRoles> userRoles = \_dbContext.ExecStoredProcedureListWithOutput<UserRoles>("GetRoleDetails", parameters.Length, parameters).ToList();

return userRoles;

}

catch (Exception ex)

{

throw new UserDefinedException("Invalid exception.");

}

}

DBContext.cs File

public IList<TEntity> ExecStoredProcedureListWithOutput<TEntity>(string commandText, int totalOutputParams, params object[] parameters) where TEntity : class, new()

{

var connection = this.Database.GetDbConnection();

IList<TEntity> result = new List<TEntity>();

try

{

totalOutputParams = totalOutputParams == 0 ? 1 : totalOutputParams;

if (connection.State == ConnectionState.Closed) { connection.Open(); }

using (var cmd = connection.CreateCommand())

{

AddParametersToDbCommand(commandText, parameters, cmd);

using (var reader = cmd.ExecuteReader())

{

result = DataReaderMapToList<TEntity>(reader);

reader.NextResult();

}

}

return result;

}

catch (Exception e)

{

return null;

}

finally

{

connection.Close();

}

}

public async Task<string?> SetDefaultPermission(int roleId, int DepartmentId, int UserId)

{

SqlParameter[] parameters = {

new SqlParameter("RoleId",roleId),

new SqlParameter("@DepartmentId",DepartmentId ),

new SqlParameter("@UserId",UserId),

};

\_dbContext.ExecStoredProcedureWithOutput<Staff>("CreateDefaultPermission", parameters.Length, parameters);

return null;

}

public TEntity ExecStoredProcedureWithOutput<TEntity>(string commandText, int totalOutputParams, params object[] parameters) where TEntity : class, new()

{

var connection = this.Database.GetDbConnection();

TEntity result = new TEntity();

try

{

totalOutputParams = totalOutputParams == 0 ? 1 : totalOutputParams;

if (connection.State == ConnectionState.Closed) { connection.Open(); }

using (var cmd = connection.CreateCommand())

{

AddParametersToDbCommand(commandText, parameters, cmd);

using (var reader = cmd.ExecuteReader())

{

result = DataReaderMap<TEntity>(reader);

reader.NextResult();

}

}

return result;

}

catch (Exception e)

{

return null;

}

finally

{

connection.Close();

}

}