namespace Core.DAL.Adapter

{

/// <summary>

/// The praxis adapter is the data access layer for all IPraxis entities.

/// </summary>

internal static class PraxisDataAdapter

{

/// <summary>

/// Saves a new entity to the database, where that entity is a root entity (i.e., it does not have

/// a parent).

/// </summary>

/// <param name="praxisEntity">

/// The praxisEntity.

/// </param>

internal static void CreateEntity(ref IPraxis praxisEntity)

{

using (var sqlConnection = new SqlConnection(DataAccess.ConnectionString))

using (

SqlCommand sqlCmd =

DataAccess.CreateCommand(DB\_SCHEMA + ".[" + CoreUtility.GetEntityTypeName(praxisEntity) + "Insert]"))

{

sqlCmd.Connection = sqlConnection;

sqlConnection.Open();

DataAccess.CreateUpdateParametersForObject(praxisEntity, sqlCmd);

praxisEntity.SetId((int) sqlCmd.ExecuteScalar());

}

}

/// <summary>

/// Creates the dynamic basic search parameters.

/// </summary>

/// <param name="praxisEntity">The praxis entity.</param>

/// <param name="sqlCmd">The SQL CMD.</param>

internal static void CreateUpdateParametersForObject(IPraxis praxisEntity, SqlCommand sqlCmd)

{

// get all public instance properties of MyClass type

PropertyInfo[] properties = praxisEntity.GetType().GetProperties(BindingFlags.Public |

BindingFlags.Instance);

string entityName = CoreUtility.GetEntityTypeName(praxisEntity).ToLower();

foreach (PropertyInfo propertyInfo in properties)

{

if (!propertyInfo.CanWrite) continue;

// Get the property name

string propertyName = propertyInfo.Name.ToLower();

// Create the command, assume non-indexed properties

// Create even if null, because we may want to set entity values to null

object propertyValue = propertyInfo.GetValue(praxisEntity, null);

string propertyDataType = propertyInfo.PropertyType.ToString().ToLower();

//If this is not the Id property of the entity, then create a parameter

if (propertyName != entityName + "id")

{

// Create the Parameters for the SQL

switch (propertyDataType)

{

case "system.string":

if (propertyValue != null)

sqlCmd.Parameters.Add(new SqlParameter("@" + propertyName, SqlDbType.NVarChar,

propertyValue.ToString().Length)).Value =

propertyValue;

else

goto default;

break;

default:

sqlCmd.Parameters.Add(new SqlParameter("@" + propertyName, GetSqlType(propertyDataType)))

.Value = propertyValue ?? DBNull.Value;

break;

}

}

}

}