**Tuple and Value tuple in C#**

In C#, both tuples and value tuples are used to store multiple values in a single variable. However, they have some key differences in terms of syntax, features, and performance.

**Tuples**

Tuples are a reference type and have been part of .NET for a long time. They are created using the System. Tuple class.

Example

using System;

class Program

{

static void Main()

{

// Creating a Tuple

Tuple<int, string, bool> tupleExample = new Tuple<int, string, bool>(1, "Hello", true);

// Accessing tuple elements

Console.WriteLine($"Item1: {tupleExample.Item1}, Item2: {tupleExample.Item2}, Item3: {tupleExample.Item3}");

}

}

C#

Copy

ValueTuples

ValueTuples are a more recent addition and are value types, introduced with C# 7.0. They are more lightweight and offer better performance compared to the older System. Tuple. ValueTuples are created using the System.ValueTuple struct.

Example

class Program

{

static void Main()

{

// Creating a ValueTuple

(int, string, bool) valueTuple = (1, "Hello", true);

// Accessing ValueTuple elements

Console.WriteLine($"Item1: {valueTuple.Item1}, Item2: {valueTuple.Item2}, Item3: {valueTuple.Item3}");

// Assigning names to ValueTuple elements

var namedValueTuple = (Id: 1, Message: "Hello", IsActive: true);

Console.WriteLine($"Id: {namedValueTuple.Id}, Message: {namedValueTuple.Message}, IsActive: {namedValueTuple.IsActive}");

}

}

C#

Copy

Key Differences

1. **Tuple:**Reference type (System.Tuple).
   * **ValueTuple:** Value type (System.ValueTuple).
2. **Tuple:**Slower due to reference type and heap allocation.
   * **ValueTuple:**Faster due to value type and stack allocation.
3. **Tuple:**Verbose requires a new keyword.
   * **ValueTuple:** Concise, more readable, can use deconstruction and named elements.
4. **Tuple:**Immutable values cannot be changed after creation.
   * **ValueTuple:** Mutable values can be changed after creation. Use in

Tuple and ValueTuple in a real-world scenario where you retrieve data from an SQL Server database.

Execute the following SQL Server query

CREATE TABLE EmployeesDetails (

Id INT PRIMARY KEY,

Name NVARCHAR(100),

IsActive BIT

);

INSERT INTO EmployeesDetails (Id, Name, IsActive)

VALUES (1, 'Alice', 1);

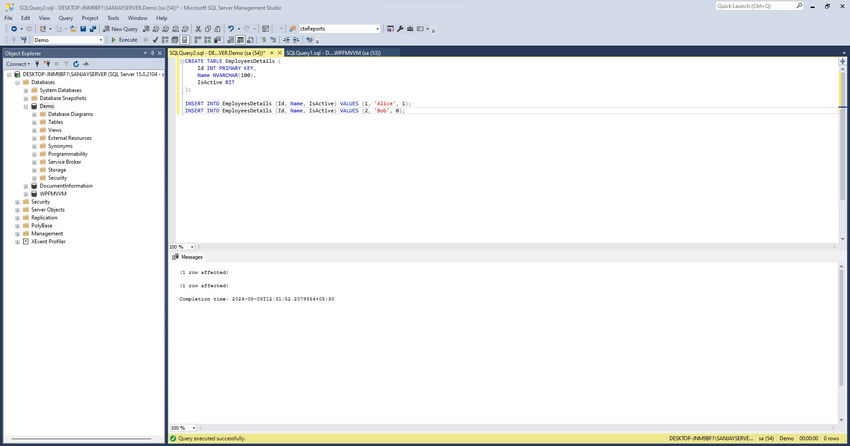
INSERT INTO EmployeesDetails (Id, Name, IsActive)

VALUES (2, 'Bob', 0);

SQL

Copy

Query Window



Create a class like Below

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace WpfApp2

{

public class TupleExamples

{

static string connectionString = "Server=DESKTOP-JNM9BF1\\OMatrixtechSERVER;Database=Demo;User Id=sa;Password=1234;"; // Write your connection string here

static string query = "SELECT Id, Name, IsActive FROM EmployeesDetails";

// Method to retrieve employee details from the database and return them as a Tuple

public static Tuple<int, string, bool> GetEmployeeTupleResult()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

return new Tuple<int, string, bool>(

reader.GetInt32(0),

reader.GetString(1),

reader.GetBoolean(2)

);

}

}

return null;

}

// Method to retrieve employee details from the database and return them as a ValueTuple

public static (int Id, string Name, bool IsActive) GetEmployeeValueTupleResult()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

return (

reader.GetInt32(0),

reader.GetString(1),

reader.GetBoolean(2)

);

}

}

return (0, null, false); // Default value if no data is found

}

}

}

// Example of how to call the implemented methods to retrieve and display employee details

private void ValueTuple\_Click(object sender, RoutedEventArgs e)

{

Tuple<int, string, bool> tupleValue = TupleExamples.GetEmployeeTupleResult();

(int Id, string Name, bool IsActive) = TupleExamples.GetEmployeeValueTupleResult();

}

C#

Copy

Modern C#

* **Tuple:**Less frequently utilized in contemporary C# programming.
* **ValueTuple:**Favored in current C# practices due to its enhanced performance and improved readability.

Conclusion

ValueTuple is the preferred option for the majority of contemporary C# applications because of its performance benefits and more succinct syntax. While Traditional Tuple remains accessible and can still be utilized, it is typically not as popular unless compatibility with older .NET versions is necessary.