

LINQ Select Switch Pattern Matching.

In C#, LINQ doesn't have a direct equivalent to a switch statement, but you can achieve similar behavior using several techniques depending on use case. The most common approach is to use a combination of the select, where, and case expressions. Below are four options for achieving this behavior. These methods are flexible alternatives to the traditional switch inside LINQ queries.

Switch-like functionality by using a conditional expression (if-else or switch) inside a select statement.

Switch-like functionality using select with a switch expression.

Example Implementation

```
var result = data.Select(item =>
item switch
{
    1 => "One",
    2 => "Two",
    3 => "Three",
    _ => "Other"
});
```

Switch-like functionality using a series of if-else statements inside a select method.

Example Implementation

```
var result = data.Select(item =>
{
    if (item == 1)
        return "One";
    else if (item == 2)
        return "Two";
    else if (item == 3)
        return "Three";
    else
        return "Other";
});
```

Switch-like functionality by using `IEnumerable<T>.Select` and a dictionary object.

Switch-like functionality using a dictionary to map keys to values and a select method to iterate over the items.

Example Implementation

```
var map = new Dictionary<int, string>
{
    { 1, "One" },
    { 2, "Two" },
    { 3, "Three" }
};

var result = data.Select(item => map.ContainsKey(item) ? map[item] : "Other");
```

Switch-like functionality by using pattern matching with a switch statement.

For more complex types, you can use LINQ matching pattern with the switch expression.

Example Implementation

```
var result = data.Select(item =>
item switch
{
    int i when i < 10 => "Small Number",
    int i when i >= 10 && i < 100 => "Medium Number",
    _ => "Large Number"
});
```

<http://www.seandrew.info>- <https://www.linkedin.com/in/seanmdrew>