Parallel LINQ Operators

In this, the final chapter of this section and of the book, we’ll walk though the key operators that support Parallel LINQ (PLINQ). As you may have noticed, PLINQ operators are expressed as a set of extension methods in the ParallelEnumerable class that are applied to the ParallelQuery type. We’ll show you the parallel operators and take a look at how they fit together.

We have only included the operators that allow you to create ParallelQuery instances or control the execution of the parallel query. Most of the PLINQ operators are identical to their LINQ to Objects counterparts, other than they are applied to ParallelQuery queries. You can see how to use these operators by looking at their LINQ to Objects equivalents in Chapters 3 and 4.

ParallelQuery Creation Operators

The following are the ParallelQuery creation operators.

AsParallel

The AsParallel method is the doorway to PLINQ. It converts data sequence into a ParallelQuery. The LINQ engine detects the use of a ParallelQuery as the source in a query and switches to PLINQ execution automatically. You are likely to use the AsParallel method every time you use PLINQ.

Prototypes

The AsParallel method has two prototypes that we will cover.

The First AsParallel Prototype

```csharp
public static ParallelQuery<T> AsParallel<T>(
    this IEnumerable<T> source
)
```

This prototype operates on an IEnumerable<T> and returns a ParallelQuery<T>, which can be used as the basis for a PLINQ query. You’ll see that we used this method in all the PLINQ examples in the previous chapter—and in almost all the examples in this chapter, too.
The Second AsParallel Prototype

```csharp
public static ParallelQuery AsParallel(
    this IEnumerable source
)
```

The second prototype creates a `ParallelQuery` from an `IEnumerable` and exists to support legacy collections, such as `System.Collections.ArrayList`. The `ParallelQuery` is not strongly typed and cannot be used as the basis for a PLINQ query without being converted to a `ParallelQuery<T>`. You can cast a `ParallelQuery` to a `ParallelQuery<T>` by using the `Cast<T>` operator or filter the sequence to get the items that are instances of `T` by using the `OfType<T>` operator.

Examples

Listing 24-1 uses the first `AsParallel` prototype to create a `ParallelQuery`, which is then used as the source for a PLINQ query. You will recognize this as the example we used often in the previous chapter—finding the presidents’ names that contain the letter `o`.

**Listing 24-1. Creating a ParallelQuery with the First AsParallel Prototype**

```csharp
string[] presidents = {
    "Grant", "Harding", "Harrison", "Hayes", "Hoover", "Jackson",

ParallelQuery<string> pq = presidents.AsParallel();

IEnumerable<string> results = from p in pq
    where p.Contains('o')
    select p;

foreach (string president in results) {
    Console.WriteLine("Match: {0}", president);
}
```

When we compile and run Listing 24-1, we get the following results:

```
Match: Roosevelt
Match: Clinton
Match: Ford
Match: Lincoln
Match: Taylor
Match: Coolidge
```