비동기 Delegate 사용법

리얼웹	배성혁	2010-08-19 오후 6:06


DelegateEx...


DelegateEx...

앞서 ASP.NET 웹 폼
의 비동기 호출 방식에 대한 예를 보였습니다. 근데, 사실 핵심은 Asynchronous Delegate (비동기 대리자) 에 대한 사용법만 알면 금방 응용할 수 있는 것입니다.

MSDN : 비동기 대리자 샘플 을 보면 엄청 복잡하지요? 이거 하라는 거야?

그래서 TPL (Task Parallel Library) 가 좋다는 겁니다.
첨부한 파일을 보실까요. ASP.NET 웹 폼의 비동기 호출 방식 예에서는 Action, Action<T>에 대해서만 작업을 수행했습니다만, TaskFactory.FromAsync 메소드를 보게되면 엄청 많은 overload 함수가 있음을 아실겁니다.

그럼 이걸 어떻게 사용해야 되지요? 그래서 제가 Extension Method 들을 만들었습니다.
사실 만들 필요도 없지만, 처음 접하는 개발자를 위해 만들었습니다.

```
public static Task<TResult> RunAsync<TResult>(this Func<TResult> function, object state)
{
    function.ShouldNotBeNull("function");

    return
        Task<TResult>.Factory.FromAsync(function.BeginInvoke,
                                        function.EndInvoke,
                                        state);
}

public static Task<TResult> RunAsync<T, TResult>(this Func<T, TResult> function, T arg, object state)
{
    function.ShouldNotBeNull("function");

    return
        Task<TResult>.Factory.FromAsync(function.BeginInvoke,
                                        function.EndInvoke,
                                        arg,
                                        state);
}

public static Task<TResult> RunAsync<T1, T2, TResult>(this Func<T1, T2, TResult> function, T1 arg1, T2 arg2, object state)
{
    function.ShouldNotBeNull("function");

    return
        Task<TResult>.Factory.FromAsync(function.BeginInvoke,
                                        function.EndInvoke,
                                        arg1,
                                        arg2,
                                        state);
}
```

특정 함수를 비동기 방식으로 수행하고, 결과를 반환 받는 확장 메소드 (반환받은 Task의 Result 속성이 함수의 결과 값이다)

아래 코드는 int 인자를 받아 double을 반환하는 함수를 비동기적으로 수행하는 방법입니다.

```
[Test]
public void Func_RunAsync()
{
    Func<int, double> @power = x =>
    {
        Thread.Sleep(Rnd.Next(Rnd.Next(1, 10), Rnd.Next(11, 50)));

        var result = Math.Pow(x, 5);
        if(IsDebugEnabled)
            log.DebugFormat("@power({0}) = {1}", x, result);
        return result;
    };

    var tasks = new List<Task<double>>();

    for(var i = 0; i < IterationCount; i++)
    {
        if(IsDebugEnabled)
            log.DebugFormat("@power({0}) called...", i);

        Thread.Sleep(0);

        // 함수를 BeginInvoke, EndInvoke로 비동기 실행할 수 있습니다.
        //
        var task = @power.RunAsync(i, null);
        tasks.Add(task);
    }

    Task.WaitAll(tasks.ToArray());

    tasks.TrueForAll(task => task.IsCompleted);
    tasks.ForEach(task => Console.WriteLine("계산 결과=" + task.Result));
}
```

보시다시피, 여러 함수를 비동기 방식으로 수행하고, 결과를 기다립니다. 로그에 찍어보면, Thread Id 가 시작과 결과에서 전혀 다름을 알 수 있을 것입니다.