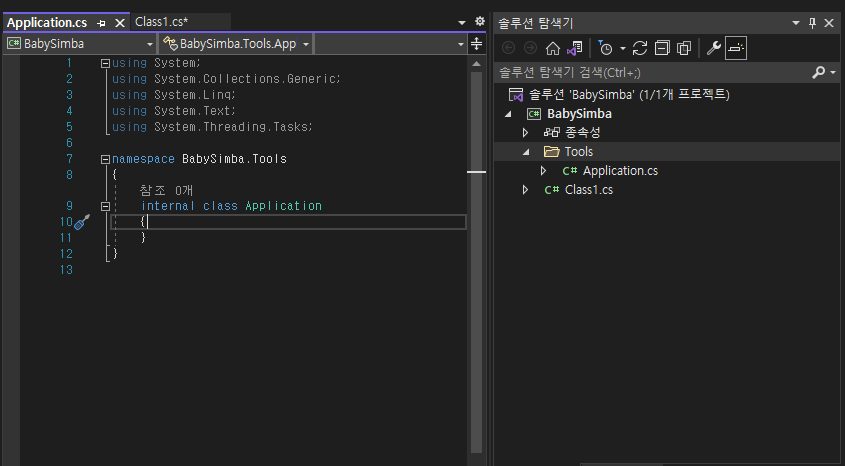
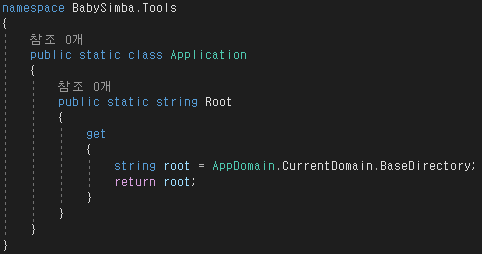
[C# 중급강좌] 라이브러리를 만들자

**2022.4.20**

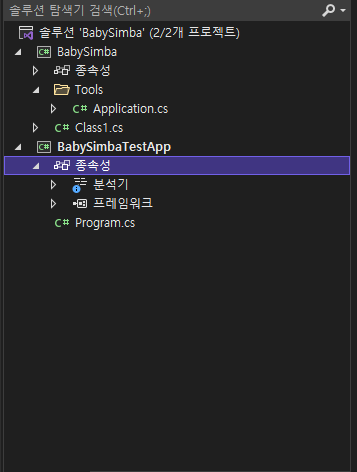
* 라이브러리를 왜 만들어야 할까?
  + 회사 내에서 여러 프로젝트를 해나가야 할텐데
  + 그 안에서 해결해야 할 것들, 솔루션을 만들면서 밤도 새고 함
  + 그런 코드를 structure로 만들어두면 그게 쌓이고 쌓여서 나중에 훨씬 쉽고 빠르게 구축할 수 있음
  + 개발자의 재산과 같음. 부를 축적해 나가는 과정.
* 새로만들기.. : VS > New Project.. > Class Library
  + 여기에서의 name이 나중에 reference할 때 사용이 될 이름. Namespace.
* 그룹화를 하자.
  + 예로서 System.Collections.Generic; 에서 System은 namespace, Collections는 그 안에서 비슷한 기능을 가진 class들끼리 group을 만든것.
* 이처럼 내가 만든 클래스를 비슷한것끼리 모아서 그룹화 하게 되므로, 이를 위해 이번 강의에서는 솔루션 디렉토리 내에 간단하게 folder를 만든다.



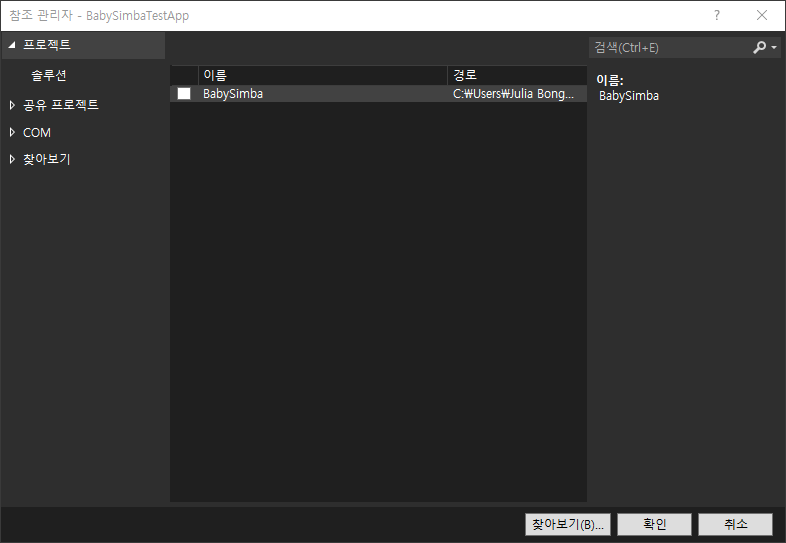
* 추후 상기를 사용하려면 using BabySimba.Tools; 를 선언해주어야 하는것임.



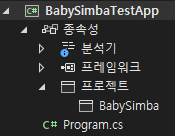
* 상기는 class 내부에 Root라는 Property를 생성해준 화면.
  + 상기 코드는 현재 application이 돌고있는 base directory를 return해주는 것임
  + 예로서 상기 코드(AppDomain.CurrentDomain.BaseDirectory) 명령어구에 대해 나는 기억을 항상 못한다고 가정하자. (예로서지만, 자주사용하는 코드를 모아놓을수도 있고, 주로 프로젝트에서 사용하는 패턴을 넣을수도 있을 듯) 그럴 때 나만의 라이브러리를 만들어서 get으로 호출만 하면 저 명령어구를 모두 기억하지 않아도 내 라이브러리만 기억하고있다면 원하는 결과값을 도출할수가 있게 됨



* 또 신기한거. 하나의 솔루션 안에 여러 프로젝트를 생성할 수 있고, 프로젝트 우클릭 후 “Set as startup project(시작프로젝트로 설정)” 선택 시 run default값으로 해당 프로젝트가 선택되는 것임.



* [종속성] 우클릭 후 “프로젝트 참조” 클릭하면 동일 솔루션 내에 있는 다른 프로젝트를 referencing 할 수 있다.



* 그럼 이렇게 됨.

|  |
| --- |
| Console.WriteLine(BabySimba.Tools.Application.Root);  *// Application(Class).Root(Property) 식으로 호출할 수 있다.* |

==========================================================

**2022.4.21**

* LogManager Library를 만들어보자.
  + 상태값, 에러메세지, process 중간중간 어디처리중이고 어디를 처리하고있는지에 대해 남겨야할 필요가 있음.
  + file을 많이 이용
* Design 먼저해보자.
  + class 정의하고 인스턴스 생성해서 function을 두개를 만들어서 쓰고싶다.
  + write은 로그작성, writeline은 로그작성 후 줄바꿈
  + class 명이 LogManager라고 한다면 Logmanager log = new LogManager();가 될 것
  + 인자값이 들어오면 그 인자에 저장, 안들어오면 현재돌고있는 폴더에 저장하게
  + 사용할때는 log.Wirte(“Processing…”); 등
* 이제 생성해보자.
  + 어제만들어두었던 Tools namespace를 동일하게 가져가도 됨. 로그매니저도 하나의 툴로 생각할 수 있으므로.
  + class를 생성하고 나면 public class가 되어야됨. 외부에서 참조해야 하므로.
* 생각해야 할 것
  + 로그는 File을 다루는 것
  + 파일도 하나의 리소스. 항상 사용 가능한지 존재한지 체크하고 다 사용했으면 빠르게 놓아주는게 중요함.
  + 파일을 생성하거나 파일이 있다면 append
* **TIP) 진짜 신박한거 하나**
  + # region 이란?
  + 개발자가 잘 알아볼 수 있도록 region을 구분해서 묶는것.
  + #region과 #endregion을 한 쌍으로 사용한다.
  + 아래는 그 예

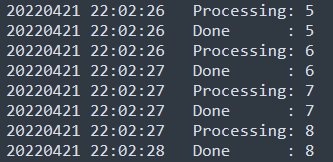
|  |
| --- |
| *// path가 안들어오는 생성자* # region Constructors public LogManager() {  \_path = System.IO.Path.Combine(Application.Root, "Log");   \_SetLogPath(); }  *// path가 들어오는 생성자* public LogManager(string path) {  \_path = path;  \_SetLogPath(); } #endregion |

* **TIP) 또하나. Property 명이나 변수명, 함수명 지정할 때.**
  + 앞에 \_ 왜붙이는지 몰랐음.
  + private 속성일때 이름 앞에 \_를 붙이는 규칙이 암묵적으로(?) 있었음.
  + 아래는 그 예(private의 사용법까지)

|  |
| --- |
| private string \_path;  *// path가 안들어오는 생성자* # region Constructors public LogManager() {  *// Log 폴더를 새로 생성해서 그 안에 적는것으로 하자. 깔끔하게*  \_path = System.IO.Path.Combine(Application.Root, "Log");   \_SetLogPath(); }  *// path가 들어오는 생성자* public LogManager(string path) {  \_path = path;  \_SetLogPath(); } #endregion   #region Methods *// 보통 name 앞에 \_를 붙이면 private라는 의미인 것 같다.* *// 이제 이걸 생성자에서 호출하면 됨* private void \_SetLogPath() {  if (!Directory.Exists(\_path))  Directory.CreateDirectory(\_path);   string logFile = DateTime.Now.ToString("yyyyMMdd" + ".txt");  \_path = Path.Combine(\_path, logFile); } |

* using이라는 키워드는
  + window resource 사용할 때 closing을 해주지 않아서 파일이 리소스를 물고있는 경우를 using이 모두 해결해주는것이죠.
  + using문이 끝나면 자동으로 점유하던 resource를 뱉으니까.
* 이렇게 새로 만들어 본 LogManager의 활용

|  |
| --- |
| using BabySimba.Tools; *// Library import!*  *// day 2* LogManager log = new LogManager();  *// i가 각각의 프로세스라고 생각하면*  *// 아래와 같이 로그를 남길수도 있을 것임* for (int i = 0; i < 10; i++) {  log.WriteLine("Processing".PadRight(10) + ": " + i);   *// Do Processing*  System.Threading.Thread.Sleep(500);   log.WriteLine("Done".PadRight(10) + ": " + i);  }  log.WriteLine("[End Processing]------"); |



* + PadRight(혹은 PadLeft)를 해주면, 보기 예쁘게 로그를찍고 Console에 출력할 수가 있다~

==========================================================

**2022.4.25**

* LogManager Library 확장편
  + Daily와 Monthly 로그를 찍어보자
  + Daily의 경우에는 yyyy\MM의 subfolder 생성하고 파일이름은 날짜
  + Monthly의 경우에는 yyyy의 subfolder 생성하고 파일이름은 yyyymm
* ENUM(열거형)
  + 기본적으로 선언은 아래와 같이 수행한다.

|  |
| --- |
| public enum Season {  Spring  Summer  Autumn  Winter } |

* + 기본 정수 숫자 형식의 명명된 상수 집합에 의해 정의되는 값 형식이다.
  + 열거형을 정의하려면 enum 키워드를 정의하고 enum의 이름을 지정한다.
  + 상기의 경우, Season.Summer를 Console.WriteLine으로 출력 시 Summer로 출력하고, (int)Season.Summer 를 Console.WriteLine으로 출력 시 1을 출력한다.(index)
* prefix와 postfix
  + 어떤 변수의 앞과 뒤에 붙이는 변수.(즉 접두어와 접미어)
  + ★ 왜 필요한가?
    - 여러 프로그램이 구동될 경우 하나의 파일을 공유하게 될 경우가 생기므로, 이를 방지하기 위함