C#과 .NET Framework

321190
2010년 가을학기
9/3/2010
박경신

Overview

- .NET Framework
- .NET Environment

Problem Statement

- 훌륭한 프로그램을 작성하는 것은 매우 어려운 일
  - 어떤 프로그래밍 언어를 사용해야 하는가?
- COM/DCOM을 사용하면 해결될 수 있지 않을까?
  - 필요한 컴포넌트를 자신의 프로그램에 attach 하여 사용하면 편하기는 하지만..
- COM은 너무 구체적인 코딩 지식을 요구함
  - COM은 각 응용마다 부수적인 인프라 구조가 필요함 – 예) Class Factory, Interface Marshaler
  - COM은 클라이언트와 서버가 지리적으로 멀리 떨어져있다고 가정하여 구현상의 문제발생 – 예) 문자열 처리
- 개발자 코드를 다양한 플랫폼에서 수행시킬 수 있길 원함
  - Platform-independent application
- 효율적인 자동 메모리 관리의 필요성
  - Garbage collection

Problem Statement

- 소프트웨어 버전 관리(version control)의 자동화 필요
  - 서버가 프로그램의 버전을 자동으로 감지하여 최신버전을 로딩?
  - 서버가 갖고 있는 프로그램의 버전을 알려줄 수 있는 표준 메커니즘이 필요
- 객체지향 프로그래밍을 모든 프로그램 언어 사이에서 사용할 수 있길 원함
  - 예) VB으로 작성된 COM 객체를 VC++에서 상속받아 확장?
- 안정성(safety)을 위해 코드 연산을 제한할 수 있길 원함
  - 예) 다운로드한 코드는 로컬 디스크에 있는 파일을 읽을 수는 있어도 쓰지는 못하도록 하고 싶다면?
- OS 기능들을 논리적으로 연관된 그룹으로 나누어 관리
  - 원하는 기능을 쉽게 찾아 쓸 수 있도록 하여 OS를 효과적으로 이용
- 그리고, 기존의 COM 객체와 문제없이 상호연동 되어야 함!

.NET Framework

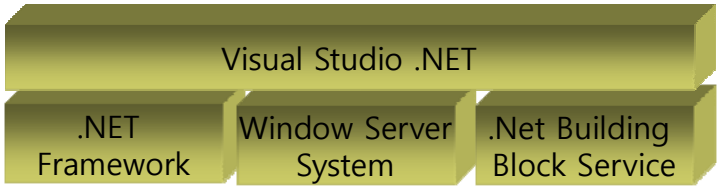
- .NET Framework란 윈도우 기반 응용프로그램을 만들기 위하여 서로 다른 프로그래밍 언어와 라이브러리들이 같이 개발되고 수행되는 환경
  - 언어 독립적 - 다양한 언어 사용 개발 가능
  - 웹 응용프로그램, GUI 등 다양한 Windows 응용 프로그램 지원
  - 방대한 코드라이브러리로 구성 & 용도와 목적에 따라 다른 모듈로 구성
- .NET 응용프로그램은 .NET Framework 상에서만 동작
  - 즉, 모든 .NET 응용프로그램은 .NET Framework에 의해서 관리
  - .NET Framework 상에서 관리되어 실행되는 응용프로그램은 Managed Code라 하고, 기존 Windows에서 직접 동작하는 응용프로그램은 Unmanaged Code라 함
- .NET Framework에 의한 서로 다른 기종들간의 통합
  - 현재 Windows 운영체제에서만 실행, 차후 FreeBDS, Linux, Mac., PDA 등에서 실행되는 버전 개발

.NET Framework

- .NET Framework 특징
  - Garbage Collection 을 통해 자동으로 메모리를 관리해줌
  - 표준화된 소프트웨어 버전 관리 기능 제공
  - 객체 지향 프로그래밍의 특성을 모든 Visual Studio .NET에서 지원하는 모든 언어에서 사용할 수 있게 해줌
  - 시스템의 기능을 여러 계층의 네임스페이스로 정리하여 제공
  - 코드에 대한 보안을 지원함 (Code Access Security)
  - COM 클라이언트나 COM 서버 모두와의 상호 운용성 (Interoperability)를 지원 – 즉, COM 객체가 .NET 객체인 것처럼 wrapper 객체로 둘러쌈
- .NET Framework의 단점
  - 운영체제가 무거워짐
  - 시스템 하드웨어 요구사항이 커짐

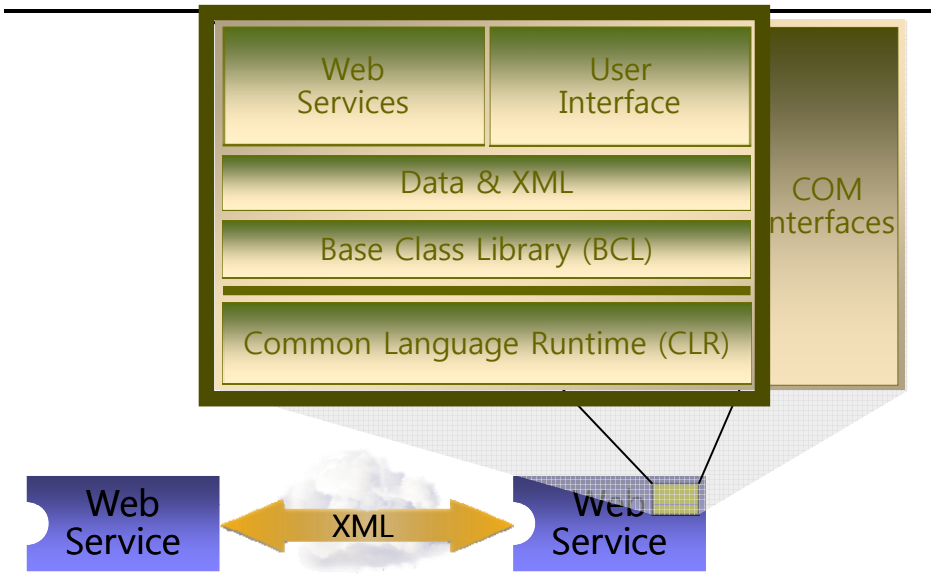
.NET Framework

- 닷넷 (.NET) 구성도

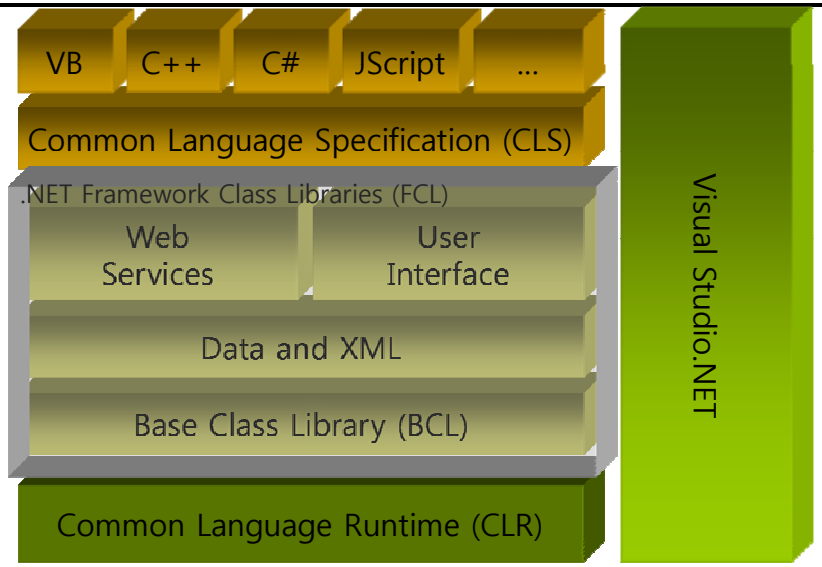


  - Visual Studio .NET : C# 등의 닷넷 언어로 프로그램을 만들기 위한 통합개발환경
  - 닷넷 프레임워크 : 닷넷을 사용할 수 있는 환경을 제공
  - 윈도우 서버 시스템 : 닷넷을 이용한 윈도우 프로그램 개발이나 웹사이트 구축시 필요한 윈도우 요소를 지원
  - 닷넷 빌딩 블록 서비스 : 닷넷의 문제해결을 위해 설계된 시스템

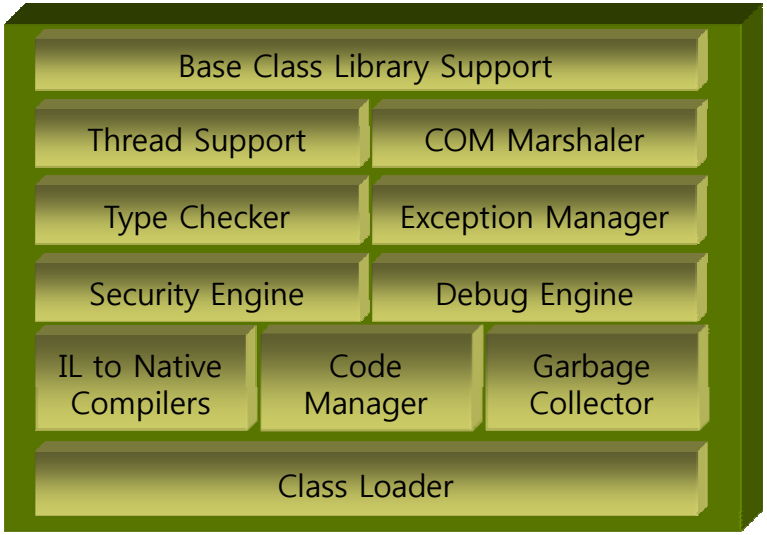
.NET Framework



.NET Framework의 구조



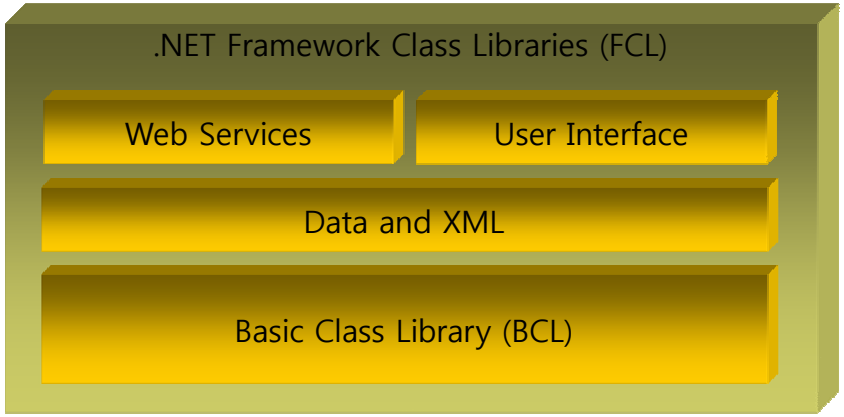
CLR의 구성



CLR

- CLR(Common Language Runtime)
  - "Managed Code" 응용프로그램을 개발하기 위한 언어 독립적인 개발, 실행환경
  - Java의 Virtual Machine 같은 .NET Framework Application을 실행하는데 필요한 실행 엔진
  - 응용 프로그램 실행 시 자동 연결•실행
  - Visual Basic .NET, C# 등과 같은 닷넷 언어는 Microsoft Intermediate Language (MSIL)로 컴파일
  - 닷넷 응용프로그램은 윈도우에서 바로 실행할 수 없고, 닷넷 프레임워크의 CLR에 있는 JIT (Just-In-Time) 컴파일러가 MSIL 코드를 Native Code로 변환해 실행 가능하게 해줌

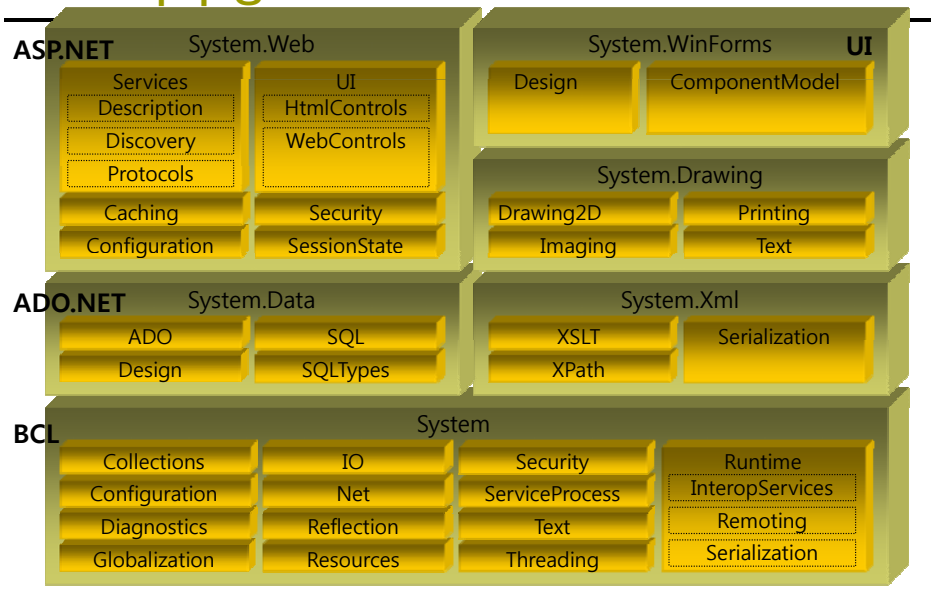
FCL의 구성



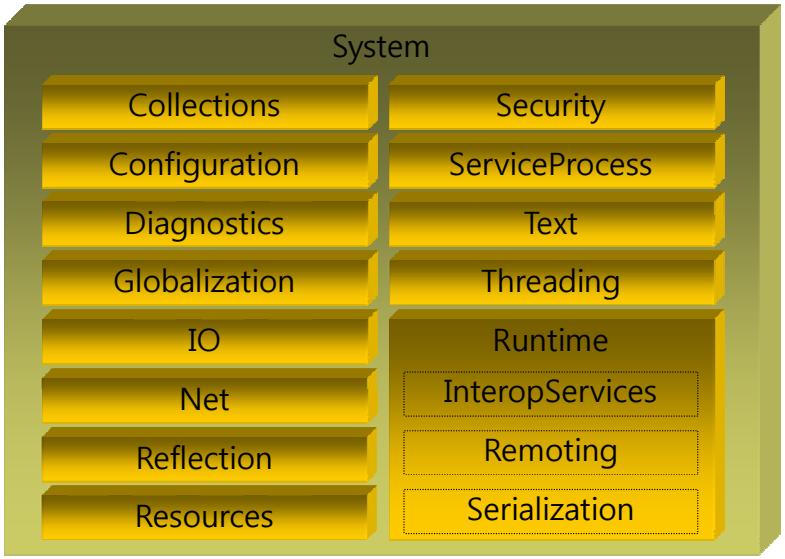
FCL

- FCL (Framework Class Libraries)
  - CLR에서 수행되기 위해 필요한 기능을 제공해주는 클래스들의 집합
  - BCL (Basic Class Library)는 클래스 라이브러리 중 가장 핵심적인 기능을 하는 클래스의 모음
  - ADO.NET는 데이터베이스를 사용할 때 지원하는 클래스
  - ASP.NET는 닷넷 언어로 웹 프로그래밍을 할 때 지원하는 클래스
  - 윈도우 UI는 닷넷 언어로 윈도우 프로그램을 만들 때 지원하는 클래스

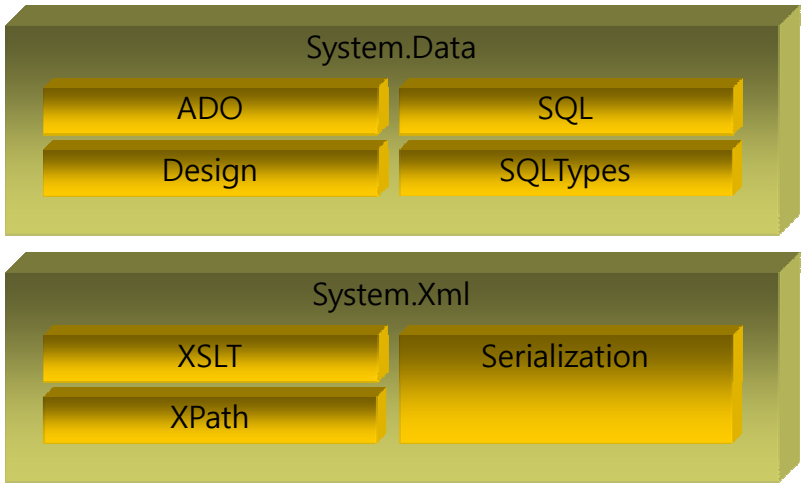
FCL의 구성



BCL



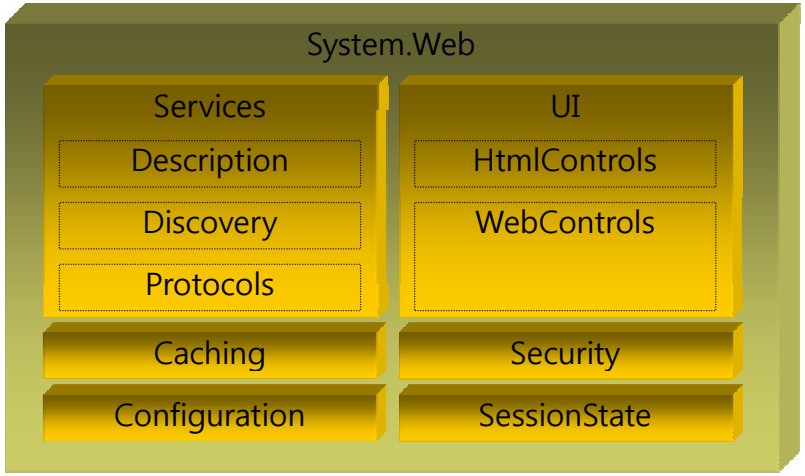
Data & XML



Data & XML

- ADO.NET은 닷넷 프레임워크의 일부로 System.Data 네임스페이스와 하위 네임스페이스에서 제공하는 데이터 접근 기술로 이루어짐
- System.Data 네임스페이스는 ADO.NET 객체 모델을 구성하는 클래스로 이루어짐
- System.Xml 네임스페이스는 W3C 호환 XML 파서와 XML 작성기 (XML Write) 그리고 XSLT와 XPath 등 많은 XML 관련 기술을 위한 클래스로 이루어짐

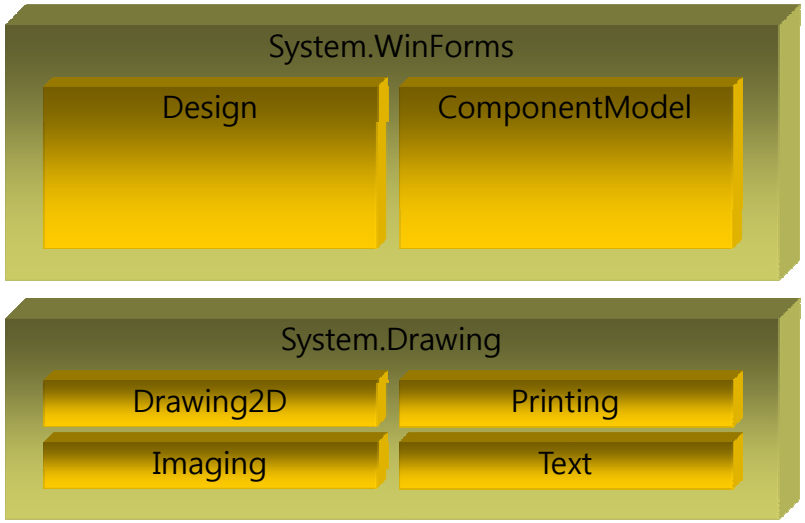
Web Forms & Services



Web Forms & Services

- ASP.NET은 강력한 웹 응용프로그램을 개발하기 위한 프로그래밍 프레임워크
- ASP.NET 웹폼 (Web Forms)은 쉬우면서도 강력한 방법으로 웹 사용자 인터페이스 (Web UI) 페이지를 생성할 수 있게 함

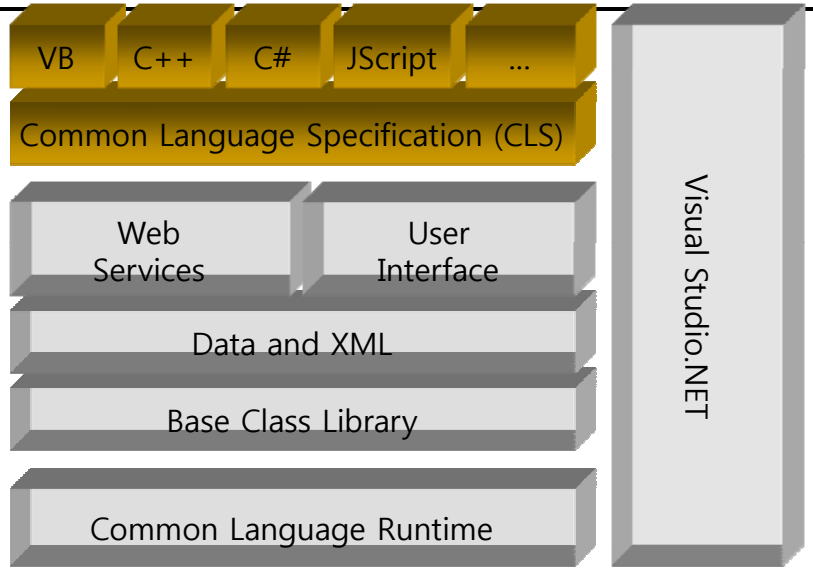
Win Forms



Win Forms

- 닷넷 프레임워크 기반의 윈도우 응용프로그램을 개발하기 위한 새로운 플랫폼
- 윈도우 UI 클래스는 System.Windows.Forms 네임스페이스에 포함되어 있음
- 윈도우 폼 유형은 크게 표준 윈도우, MDI 윈도우, 대화상자 등으로 나눌 수 있음

.NET Language



.NET Language

- 닷넷 플랫폼은 언어에 중립적 (language neutral)
  - 닷넷 언어로 VB, C++, C#, Jscript, 등이 있음
  - APL, COBOL, Pascal, Eiffel, Python, Smalltalk 등 third-parties 닷넷 언어가 현재 개발되고 있음
- CTS(Common Type System)
  - 상호 운용성을 위한 데이터의 기본적인 특징 정의
  - .NET 언어가 지원해야 할 타입과 연산자들 집합
- CLS(Common Language Specification)
  - 모든 .NET 언어가 최소한 지원해야 할 CTS 부분집합
  - 무부호 정수 (unsigned integer) 타입은 CTS에는 기술되어 있지만 C# 등 일부 .NET 언어는 지원하지 않음

CTS

CTS 타입	CLS 타입여부	C# 타입	비고
System.Boolean	Yes	bool	True/false
System.Object	Yes	object	모든 클래스의 기반클래스
System.String	Yes	string	문자열
System.Double	Yes	float	32비트 부동소수
System.Byte	Yes	double	64비트 부동소수
System.Sbyte	No	byte	무부호 8비트 정수
System.Int32	Yes	int	부호 32비트 정수
System.Uint32	No		
System.Int64	Yes	long	부호 64비트 정수
System.Uint64	No		
System.Char	Yes	char	16비트 유니코드 문자
System.DateTime	Yes	System.DateTime	날짜와 시간
System.Decimal	Yes	decimal	10진 부동 소수

C#

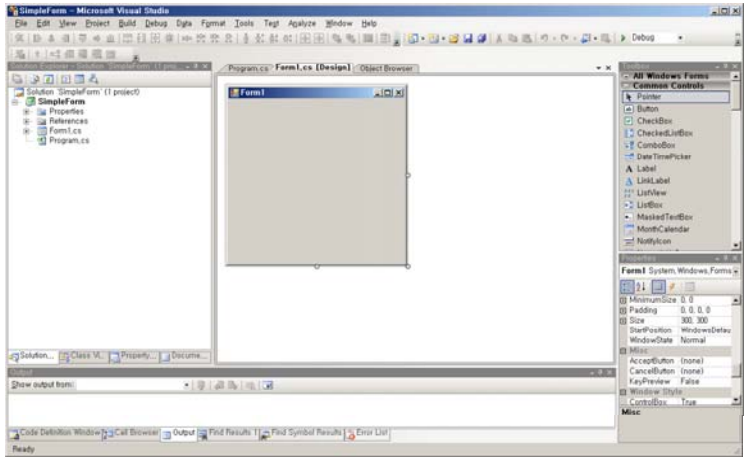
- C++를 기반으로 하고 자바의 장점을 혼합하고 닷넷의 개념을 도입해서 만든 언어
- 컴포넌트 기반의 객체지향적 언어
  - 여러 개의 컴포넌트화된 코드가 서로 유기적으로 연결되어 사용자가 원하는 프로그램을 쉽게 개발할 수 있음
- C# 언어의 장점
  - 개발 속도가 빠름
  - Windows API를 사용 가능함
  - 프로그램 컴포넌트를 사용하기 간편함
  - 닷넷 프레임워크 프로그래밍 모델을 가장 잘 반영함
  - 빠른 표준화 작업
- C#의 Garbage Collector
  - 객체를 검사하여 더 이상 사용되지 않는 객체를 메모리에서 자동으로 제거함

.NET Environment

- Visual Studio.NET
  - 차세대 운영환경을 개발하기 위한 도구
  - .NET Framework가 상위레벨로 존재하는 유연한 구조
  - CLS(Common Language Specification, 개발언어 인터페이스)는 VB.NET, C#에 맞는 런타임 인터페이스 제공
  - 상위의 언어(C#, VB.NET, C++, Jscript 등)는 동일한 하부구조의 존재로 자신에게 맞는 언어로 프로그래밍
  - 하나의 프로그램을 여러 언어로 작성가능
  - 자유로운 디버깅
- 통합개발환경
  - 사용자를 위한 환경설정
  - MSDN 도움말 활용 가능
  - 다양한 기능이 있는 도구 모음
  - 시작페이지는 웹 형식으로 구성 (웹 브라우저 기본 내장)

.NET Environment

- Visual Studio.NET 2008 초기화면



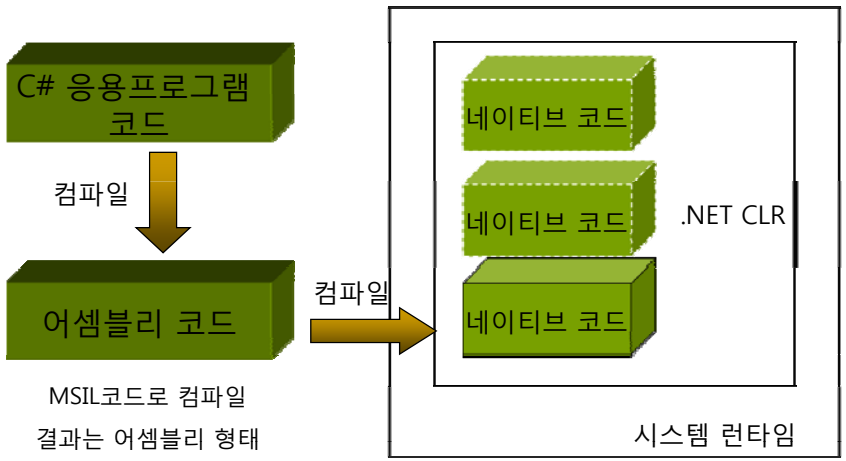
Visual Studio.NET 설치하기

- Component Update
  - 컴포넌트(Component) – 프로그램을 설치하기 위한 구성요소
  - Visual Studio.NET 설치의 1단계 과정
  - Windows Component Update – 컴포넌트들을 VS.NET에 필요한 수준으로 업데이트
- Visual Studio.NET 설치
  - 설치를 위한 리소스 임시 디렉토리에 복사
  - 사용권 계약
  - 설치 옵션 설정

.NET Framework

- 응용 프로그램 작성
  - C# 언어와 VS.NET의 추가적인 도구 사용
  - 컴파일 – 2단계의 과정을 통한 실행 가능한 형태로 변환
  - MSIL(MicroSoft Intermediate Language) – 운영체제에 국한되지 않은 중간 언어, 직접 실행 불가능
  - JIT(Just-In-Time) – MSIL 코드를 원시코드로 컴파일, OS가 실행할 수 있는 응용프로그램의 형태
- Assembly
  - .NET이 사용하는 프로그램의 실행 및 배포 단위
  - .NET run-time이 수행하는 모든 코드는 어셈블리 안에 들어있음
  - 응용 프로그램 코드와 리소스를 갖고있는 실행파일(.exe) 이나 라이브러리파일(.dll)의 논리적인 집합
  - 어셈블리는 Manifest(어셈블리 내부에 들어있는 코드와 리소스를 설명하는 메타데이터) 목록도 포함
  - 어셈블리는 단일파일 또는 다중파일로 구성 – Manifest는 어셈블리의 exe 파일이나 dll 파일 중의 하나에 들어있을 수 있음

.NET Framework

- Managed Code
  - CLR에 의한 관리 기능을 이용하여 실행되는 코드
- Garbage Collection
  - 응용프로그램이 사용하던 메모리를 해제
- Linking
  - 응용 프로그램 코드를 여러 개의 소스코드 파일로 나눈 후, 그것을 단일한 어셈블리로 합쳐서 컴파일 하는 것
  - 코드를 여러 파일로 나누어 작업 – 완료 시 파일은 코드 단위
  - 작업 효율성 증가

.NET Framework