VISUAL STUDIO 2010 응용 프로그램 모델링 완전 정복 백서

Visual Studio Korea Team Blog http://vsts2010.net

작성자 : Microsoft Visual Studio ALM MVP 엄준일

http://blog.powerumc.kr

powerumc@gmail.com

감수자 : 강성재 부장

한국마이크로소프트 개발자 및 플랫폼 사업 총괄 Win2kin2@hotmail.com

도움주신 분 : 김남영 부장

한국마이크로소프트 개발자 및 플랫폼 사업 총괄 inykim@empal.com

Contents

1. 들어가기에 앞서...

이 문서는 모델링의 기본적인 이해와 더 나아가 어플리케이션이나 기능을 Visual Studio 2010 통합 개발 도구로 어떻게 만들어 나가는지 여러분들에게 보여주며 개발자를 대상으로 하는 문서입니다. 모델링이 어렵거나 모델링 작업이 매우 비효율적인 작업이라고 느끼셨던 분들은 문서에서 각 UML 다이어그램의 완성된 모델링을 통해 얼마나 더 이해하기 쉬운지 관전 포인트를 두는 것도 나쁘지 않을 것입니다. 더 나아가 명세서라는 글로 표현되는 모든 것들은 주로 사용하는 용어나 지식, 문장의 흐름에 따라 이해하기 힘들기도 하지만, UML 다이어그램을 통해 여러 사람들과 오해의 소지 없이 올바른 소통이 가능한지도 관전 포인트를 두는 것도 좋습니다.

Visual Studio 통합 개발 도구는 여러분이 개발하기 쉬운 환경을 제공해 주지만, UML 다이어그램 기능으로 여러분들의 개발/설계 능력과 생각의 폭을 한껏 높여줄 것이라 믿어 의심치 않습니다. 혹시 독자 여러분들이, '모델링은 나와 거리가 멀어!', '모델링은 관심 없어' 라고 느끼시는 분들은 특히 관심 있게 보시길 부탁 드리며, 좀 더 개발을 잘 하기 위해 안내해 드리는 백서임을 다시 한번 강조 드립니다.

자! 이제 Visual Studio 2010 응용 프로그램 모델링 완전 정복 백서를 함께 시작해 봅시다.

2. VISUAL STUDIO 2010 VISUALIZATION & MODELING 개요

2.1. 통합 개발 환경

일반적으로 Visual Studio 와 같은 도구를 일컬어 통합 개발 환경(IDE-Integrated Development Environment)이라고 부르며, 최근에는 대부분의 언어적/플랫폼적인 환경에서 이러한 통합 개발 환경의 도구를 지원해 주고 있습니다. 이것은 단순히 개발을 하기 위한 도구가 필요한 것을 넘어 개발자 친숙한 도구와 다양한 어플리케이션의 개발에 필수적인 요소라고 할 수 있습니다.

최근 Visual Studio 2010 은 통합 개발 환경을 넘어 어떻게 초기에 설계를 잘 할 것이며, 테스트와 릴리즈를 잘 할 수 있도록 많은 부분을 지원해 주기도 합니다. 개발 영역뿐만 아니라 그 외적인 생산성/협업성/품질관리 등 Visual Studio 2010 는 다른 통합 개발 도구에 비해 굉장히 진보하였으며, 점차적으로 개발 도구는 이러한 추세로 더욱 발전하리라 의심치 않습니다.

2.2. 개발 프로세스 속의 모델링

어떤 프로젝트를 할 때 얼마나 설계에 비중을 둘 것인지는 매우 중요한 문제입니다. 전체 일정이 6 개월일 때 에서 요구사항과 현황을 분석하고 설계를 하는 기간을 3 개월로 잡는다면, 실제로 구현을 할 기간은 나머지 3 개월 밖에 없습니다. 하지만, 초기에 좀 더 잘 설계를 하는 것이 구현에 이점이 되기도 하지만, 실제로 구현과 테스트, 릴리즈, 그리고 인수인계(또는 서비스 되는 시점) 등 초기 설계에 많은 시간을 투자하여 정작 제대로 동작하는 소스 코드 산출물이 제때 나오지 못할 수 있습니다. 반대로, 초기 요구사항과 현황 분석 및 설계를 1 개월의 일정으로 잡는다면 막상 구현은 제대로 설계가 되지 않은 문제로 원하는 어플리케이션의 결과를 얻기 힘들 수 있으며, 그 개발 과정은 제대로 컨텍스트 공유가 되지 않아 많은 불화음이 발생할 수 있을 것입니다.

특히 분석-설계-개발의 과정 동안 설계되는 다양한 문서와 모델링은 실제로 각 단계별로 생산적으로 도움이 되는 경우는 매우 드물기도 합니다. 우리 나라의 SI 와 같은 프로젝트의 특성상 인력의 이동이 매우 빈번하게 일어나기도 하며, SI 프로젝트가 아니더라도 분석/설계 인원이 개발까지 적극적으로 참여하여 올바른 어플리케이션의 개발에 도움이 되는 경우도 드물기도 합니다.

즉, 소프트웨어는 설계-분석도 중요하지만 어떻게 잘 개발할지에 대한 고민도 필요합니다. 설계자는 개발자가 잘 이해하고 설계를 이행할 수 있는 모델이 필요하며, 개발자는 모델을 올바르게 이해하고 직/간접적으로 개발에 도움이 되는 그런 모델을 추구합니다. 기존의 많은 통합 개발 도구는 개발에 필요한 기능을 통합하였을 뿐, 이러한 각 역할간의 통합이 매우 부족했던 것이 사실입니다.

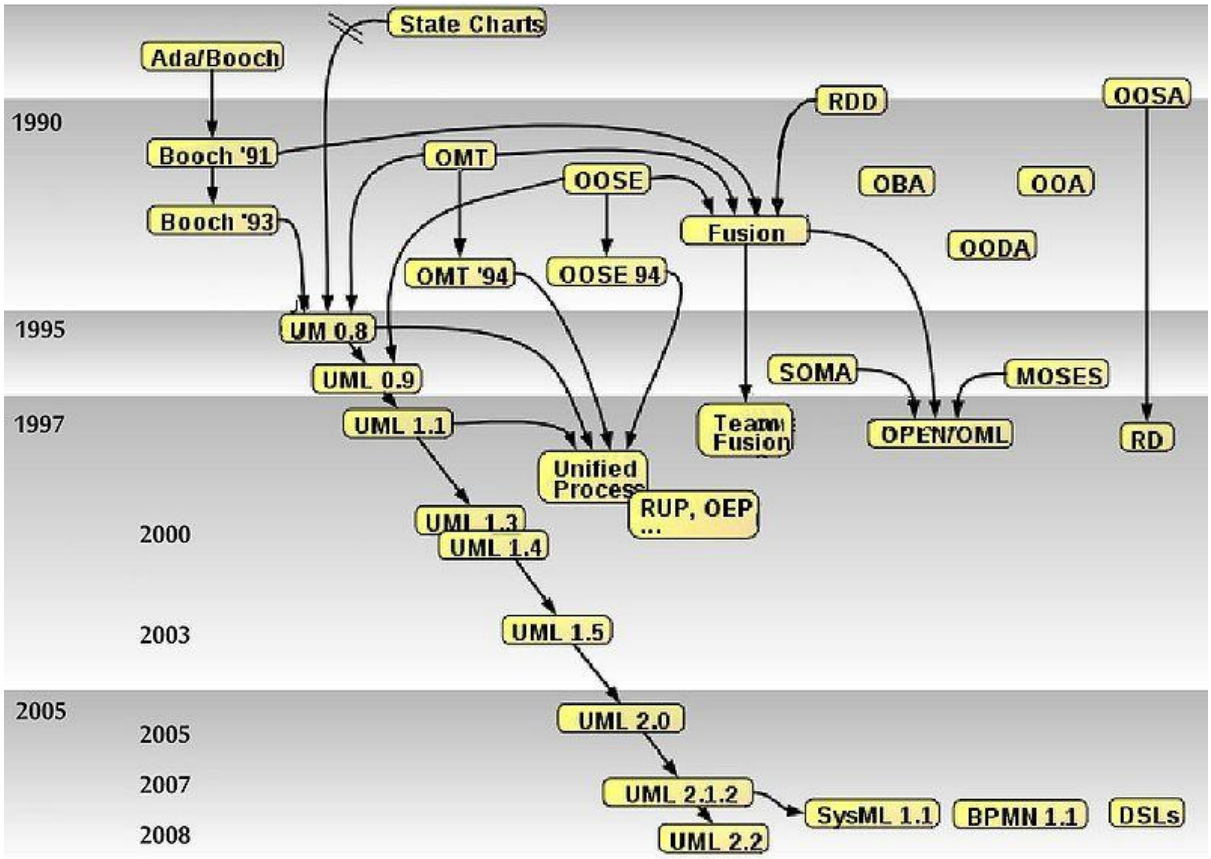
Visual Studio 2010 의 모델링 기능은 이런 여러 가지 괴리감을 상당히 좁혔습니다. 모델링을 통해 서로가 이해할 수 있는 다양한 형태의 UML 다이어그램을 제공하고 있으며, 여러 가지 UML 다이어그램은 그저 감상용이 아닌, 개발자가 사용할 수 있는 코드로 변환이 되거나, 의도대로 코드를 작성하는데 도움이 됩니다.

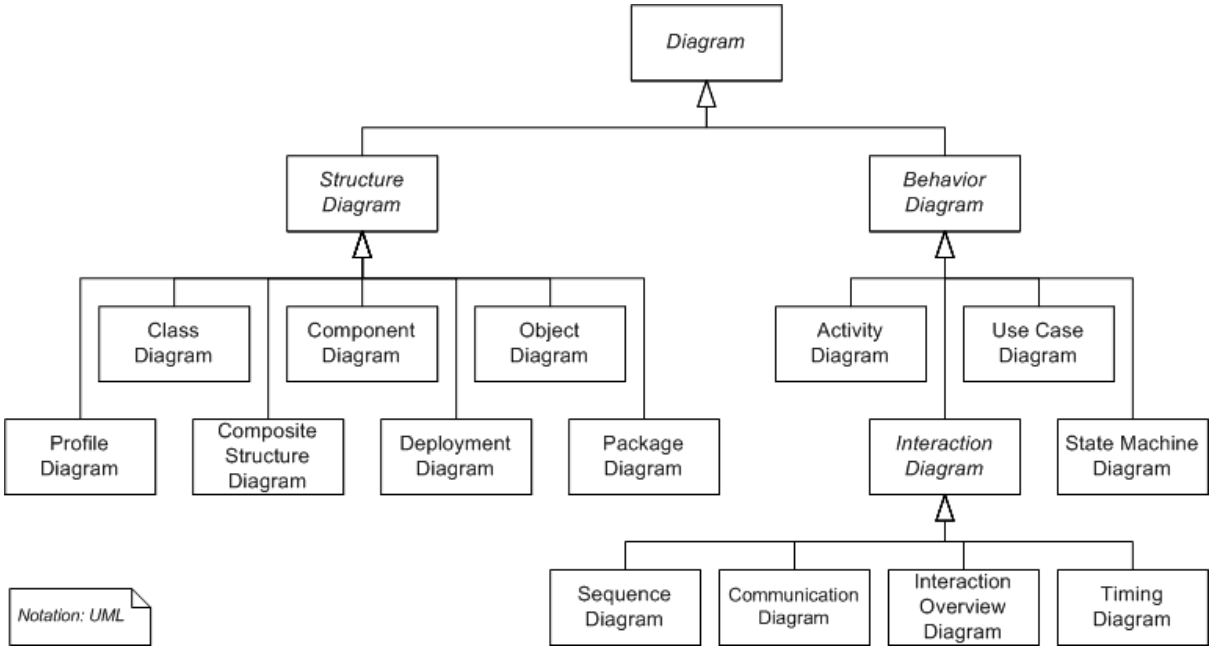
3. 모델링 시작하기

3.1. 왜 모델링인가?

일반적으로 UML 이라고 하면 Unified Modeling Language(통합 모델링 언어)라고 부르는 소프트웨어 공학의 표준화된 언어를 일컫는 용어입니다. 과거 1980 년과 1990 대 사이에 많은 객체 지향 모델링 기법이 생겨났었지만, 여러 사람들에 의해 다양한 모델링 기법과 표기법을 사용하였으며 어플리케이션을 위한 모델링, 또는 데이터베이스만을 위한 모델링 등 특정한 목표에 의해 사용되어져 왔습니다.

1990 년대 중반 Rational Software 에서 여러 가지 노력 끝에 모델링 방법(Unified Method) 버전 0.8 을 내놓았고 그 이후 컨소시엄 형태로 여러 기업이 합류하게 되었습니다. 이들은 1997 년에 UML(Unified Modeling Language)로 이름을 바꾼 버전 1.0 을 OMG 에 제출하면서 지속적으로 UML 을 이끌어 왔습니다.





참고 : http://en.wikipedia.org/wiki/Unified_Modeling_Language

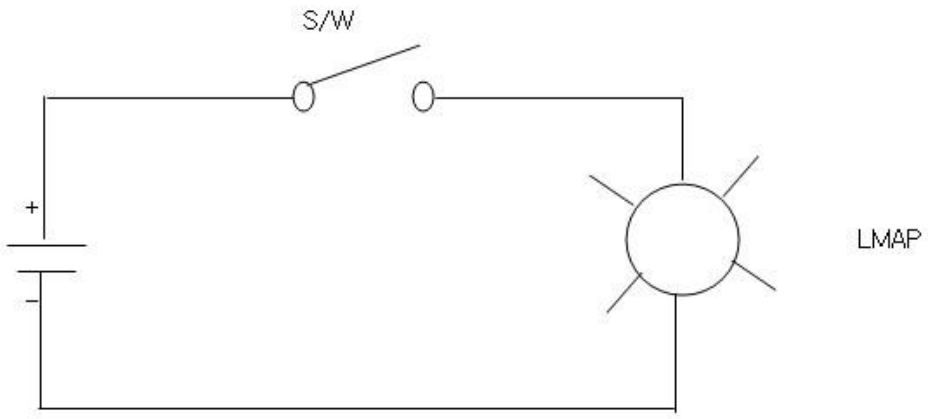
이전에는 여러 가지 독립적인 표기법을 사용했던 것과 달리, UML 은 많은 업체 전문가들, 소프트웨어 개발 회사 등이 함께 만들어 오면서 소프트웨어 개발을 위한 전세계적인 표준 언어 모델링 역사가 시작된 것입니다.

모델링은 어떤 다이어그램을 선택하느냐에 따라서 그 내용과 추상화 정도가 달라지게 되지만, 일반적으로 어떤 특정한 개체를 나타내고자 할 때, 그리고 어떤 디자인(목표)를 정의할 때, 목표에 대한 예시(예제)등을 표현할 때 매우 적절하기도 합니다.

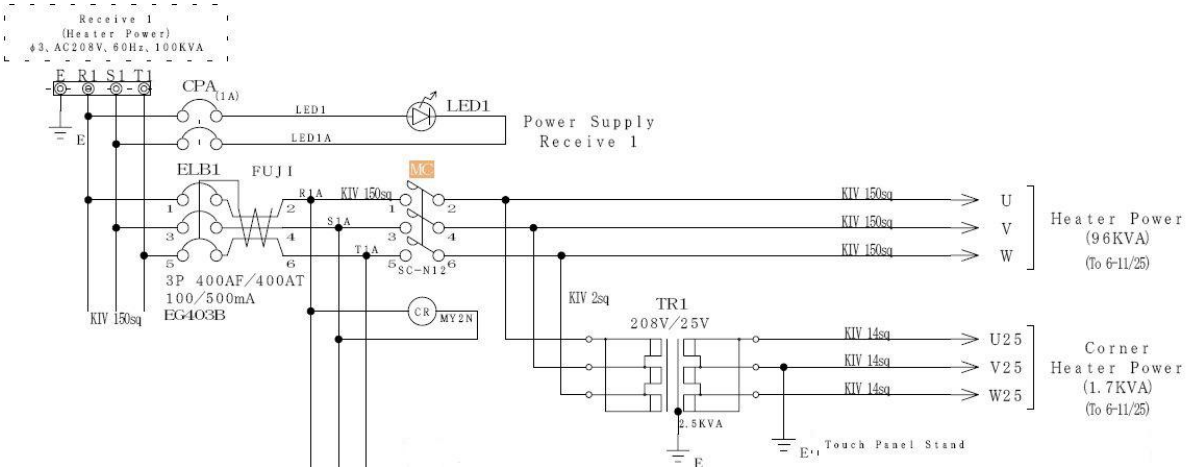
일상생활에서도 이러한 모델링 속에 우리가 살고 있다고 해도 과언이 아닙니다. 지하철 운행 정보와 지하철 노선도, 자동차 운전시에 네비게이션, 체계적으로 교통을 통제하는 신호등 같은 것들이 대표적이기도 합니다. 이런 생활 속의 모델링은 직접적으로 나에게 어떤 작용을 하는 것은 아니지만, 이러한 모델들은 나에게 의사 소통을 도와주고, 복잡한 것들은 단순화 시켜줍니다.

3.2. 왜 모델링을 해야 하는가?

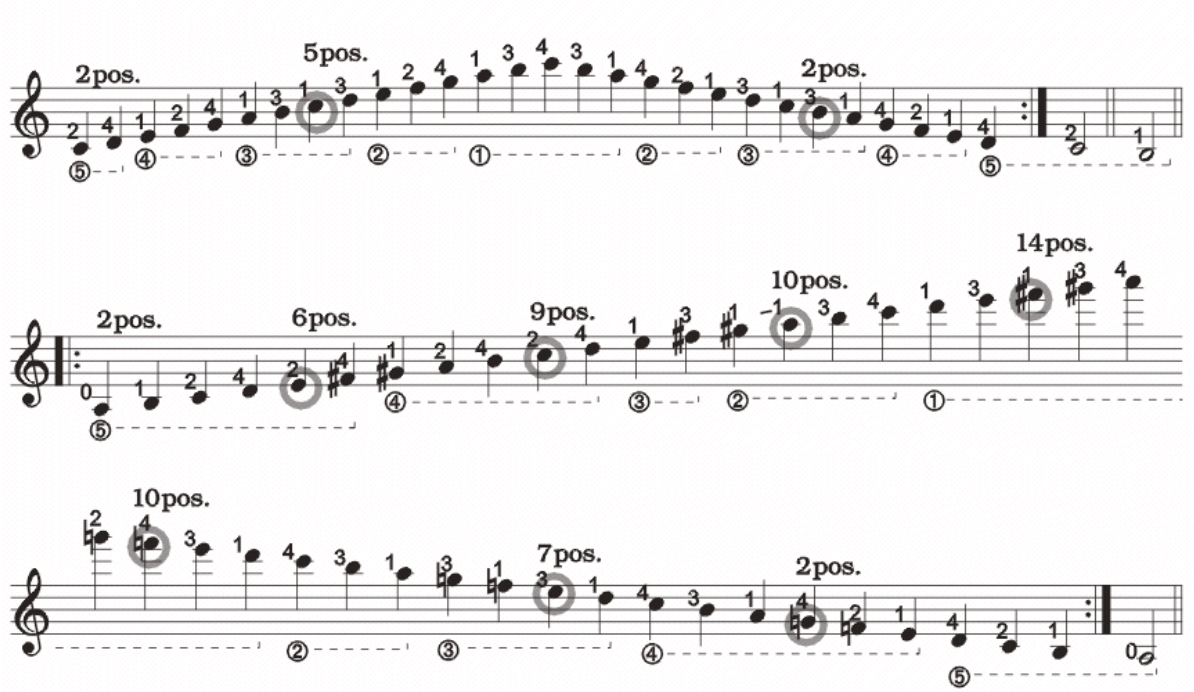
아래와 같이 초등학교 교과서에 나올법한 전기 도면이 있습니다. 스위치를 누르면 불이 들어오는 도면입니다.



사실 위와 같은 간단한 전기 도면은 도면 자체로써 크게 가치가 없을지도 모릅니다. 하지만 더 복잡한 전기 도면은 실제로 도면을 통해 시뮬레이션을 하여 큰 오류를 미리 알아낼 수 있기도 합니다. 아래와 같은 표기법 등은 공통된 이해 관계자간에 어떻게 회로가 설계 되었는지, 어떻게 작동하는지 쉽게 이해할 수 있는 것입니다.



다음과 같은 악보도 마찬가지 입니다. 이 악보는 음악을 어떻게 표현하는지에 대한 표기법으로 구성되어 있습니다. 음악의 느낌과 연주 방법을 말로 하는 것보다 아래의 악보를 통해 연주자는 어떻게 연주를 해야 하는지 일관성 있게 알려줄 수 있는 표기법입니다.



우리가 모델링을 해야 하는 것도 이런 이유들과 크게 다르지 않습니다. 단순한 기능, 요구 사항의 구현에서는 이와 같은 모델링이 필요하지 않을 수 있겠지만, 이런 기능 명세나 요구 사항, 그리고 시스템, 아키텍처적인 측면에서는 완성되기 이전에 올바르게 그 목표를 정의하고 이해하는지 소통할 수 있는 방법은 모델링이라는 것입니다.

이러한 모델링은 여러 이해 관계자나 업무의 형태에 따라 매우 효과적으로 이용될 수 있습니다.

업무적인 측면에서 모델링은 업무의 프로세스를 표현하거나 이해하는데 매우 도움이 됩니다. 업무의 흐름에서 각 작업의 흐름을 파악할 수 있으며, 전체적으로 나와 관련되는 조직 체계를 이해하는데 효과적입니다.

아키텍처 측면에서 모델링은 구축하고자 하는 시스템의 추상화된 이해와 다른 연계 시스템과의 데이터나 연계적인 흐름을 정의하여 시스템 관리자 및 설계자, 개발자 간에 의사 소통에 용이합니다.

어플리케이션 측면에서 모델링은 시스템 자체에서 동작되는 방식을 정의하여 추상화된 아키텍처를 저수준에서 이해하는데 효과적입니다.

데이터베이스 측면에서 모델링은 데이터베이스에서 데이터의 구조를 정의하고 어플리케이션과 어떻게 교류하는지에 대해 이해하는데 효과적입니다.

3.3. 모델링을 위한 어떤 다이어그램이 있는가?

UML 은 구조 다이어그램(structure diagram)과 행동 다이어그램(behavior diagram) 이라는 두 가지의 기본적인 다이어그램을 포함합니다. 구조 다이어그램은 시스템의 정적인 구조를 나타냅니다. 다양한 구조 다이어그램들은 다음과 같습니다.

- Class diagram(클래스 다이어그램)은 UML 모델링에서 사용되는 가장 일반적인 다이어그램으로 시스템, 그 구조 그리고 그들의 상호 관계 내에 존재하는 정적인 것을 나타냅니다. 시스템의 논리적 및 물리적 설계를 나타내는 데 주로 사용됩니다.
- Component Diagram(컴포넌트 다이어그램)은 컴포넌트집합 내의구성과 의존관계를 보여줍니다. 구현되는 시스템과 시스템 내에서 부품들이 어떻게 상호 작용하는지를 보여줍니다.
- Object Diagram(개체 다이어그램)은 시스템 내의일련의 개체들 사이의 관계를 보여주는데, 특정시점에 시스템의 스냅샷을 보여줍니다
- Deployment Diagram(배치 다이어그램)은 물리적 시스템의 실행 시점의 아키텍처를 보여줍니다. 배치 다이어그램은 하드웨어와 이 하드웨어 내에 있는 소프트웨어의 설명을 포함할 수 있습니다.

UML 2.0 은다음과 같은 구조 다이어그램을 추가되었습니다.

- Composite Structure Diagram(복합체 구조 다이어그램)은 모델링 요소들의 내부적 구조를 보여줍니다.
- Package Diagram(패키지 다이어그램)은 패키지 사이의 의존 관계를 나타냅니다. (패키지는 다른 모델 요소들을 그룹화하는데 사용되는 모델 요소입니다).

Behavior Diagram (행동 다이어그램)은 시스템 내 요소들의 동적인 행동을 나타냅니다. 다양한 행동 다이어그램들은 다음과 같습니다.

- Activity Diagram(활동 다이어그램)은 시스템 내의 활동들의 흐름을 보여준다. 여러 업무 프로세스들을 설명하는데 자주 사용됩니다.
- Use Case Diagram(사용 사례 다이어그램)은 시스템이 구현할 업무 프로세스를 다룹니다. Use Case Diagram 은 시스템이 작동하는 방법과 시스템과 교류하는 사람들을 나타냅니다.
- Statechart Diagram(상태 다이어그램)은 개체의 상태와, 이 개체가 상태들 사이를 어떻게 전이하는지 보여줍니다. 상태 다이어그램은 상태, 전이, 이벤트, 그리고 활동을 포함할 수 있습니다. 상태 다이어그램은 동적 뷰를 제공하며, 이벤트 중심의 행동을 모델링할 때 매우 중요하죠. 예를 들어, 전화 교환 시스템 내의 스위치를 나타내기 위해 상태 다이어그램을 사용할 수 있습니다. 이 스위치는 여러 이벤트들에 기초하여 상태를 바꾸며, 어떻게 이 스위치가 작동하는지를 이해하기 위해 상태 다이어그램에서 이 이벤트들을 모델링할 수 있습니다. UML 2.0 에서는 State Machine Diagram(상태 기계 다이어그램)이라고 합니다.
- Collaboration Diagram(협력 다이어그램)은 Sequence Diagram(순차 다이어그램) 처럼 Interaction Diagram(교류 다이어그램)의 일종입니다. 협력 다이어그램은 개체들이 어떻게 협력하고 교류하는지를 강조합니다. UML 2.0 에서 협력 다이어그램과 대등한 것은 Communication Diagram(통신 다이어그램)입니다.
- Sequence Diagram(순차 다이어그램)은 또 다른 종류의 교류 다이어그램입니다. Sequence Diagram 은 시스템의 다른 요소들 사이의 메시지들의 시간 순서를 강조합니다.

UML 2.0 은 다음과 같은 행동 다이어그램을 추가합니다.

- Timing Diagram(타이밍 다이어그램)은 또 다른 종류의 교류 다이어그램이다. 세부 타이밍 정보와, 교류하는 요소들의 상태 또는 조건 정보에 대한 변경 사항을 나타냅니다.

- Interaction Overview Diagram(교류 개요 다이어그램)은 Interaction Sequences(교류 순차)들 사이의 제어 흐름에 대한 개요를 보여주는 데 사용되는 고수준의 다이어그램입니다.

3.4. VISUAL STUDIO 2010 의 모델링 지원

Visual Studio 2010 부터 지원하는 모델링은 다음과 같은 제품에서 지원합니다. 더 자세한 내용은 Visual Studio 2010 제품 페이지를 참고하십시오. (http://www.microsoft.com/visualstudio/ko-kr/products)

Visual Studio 2010 버전의 모델링은 현재 UML 2.0 기준의 13 가지의 모든 다이어그램을 제공하지 않습니다. Visual Studio 2010 에서 제공하는 다이어그램은 다음과 같습니다.

- UML Activity Diagrams
- UML Use Case Diagrams
- UML Sequence Diagrams
- UML Class Diagrams
- Layer Diagrams

제품 기능	Professional (MSDN Essentials 포함)	Professional (MSDN 포함)	Premium (MSDN 포함)	Ultimate (MSDN 포함)	Test Professional (MSDN 포함)
아키텍처 및 모델링					
아키텍처 탐색기				✔	
UML® 2.0 규격 다이어그램(활동, 사용 사례, 시퀀스, 클래스, 구성 요소)				✔	
Layer Diagrams 및 종속성 유효성 검사				✔	
읽기 전용 다이어그램(UML, 레이어, DGML 그래프)			✔		

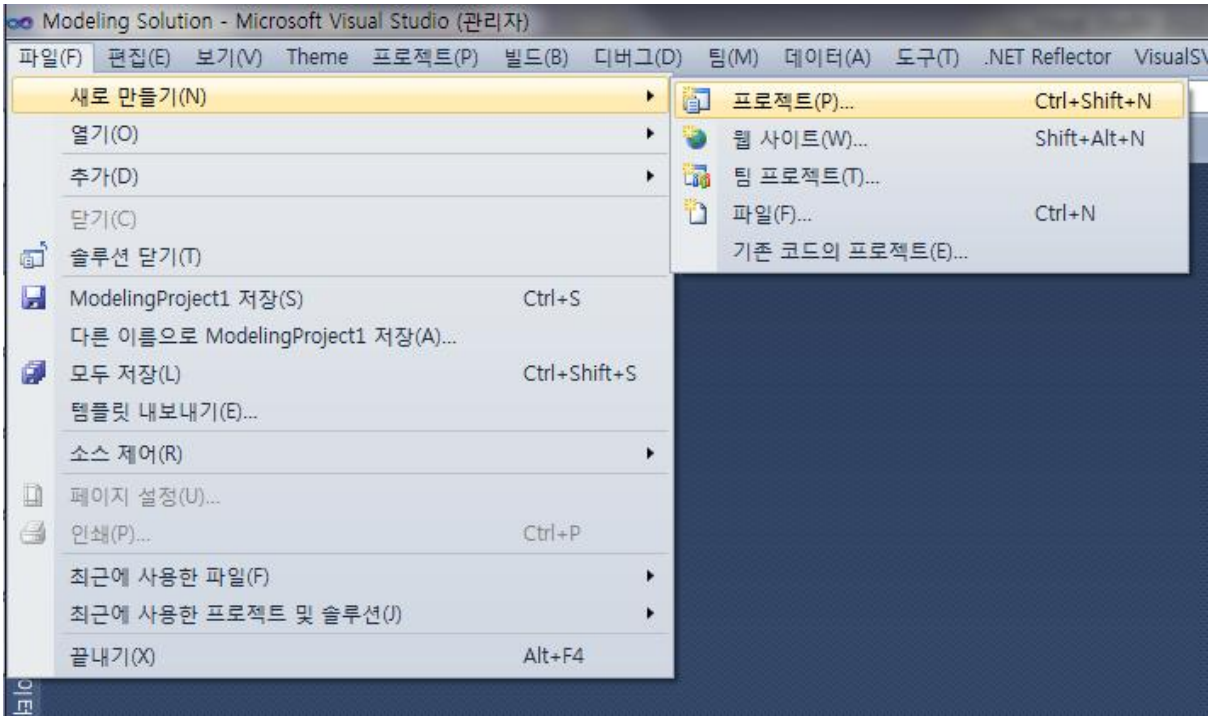
3.5. 모델링 프로젝트 생성

Visual Studio 2010 에서는 모델링을 위한 프로젝트 형식을 지원해 줍니다. 모델링 프로젝트 형식은 여러 가지 다이어그램을 만들 수 있는 아이템 템플릿을 제공을 해줍니다. 우리가

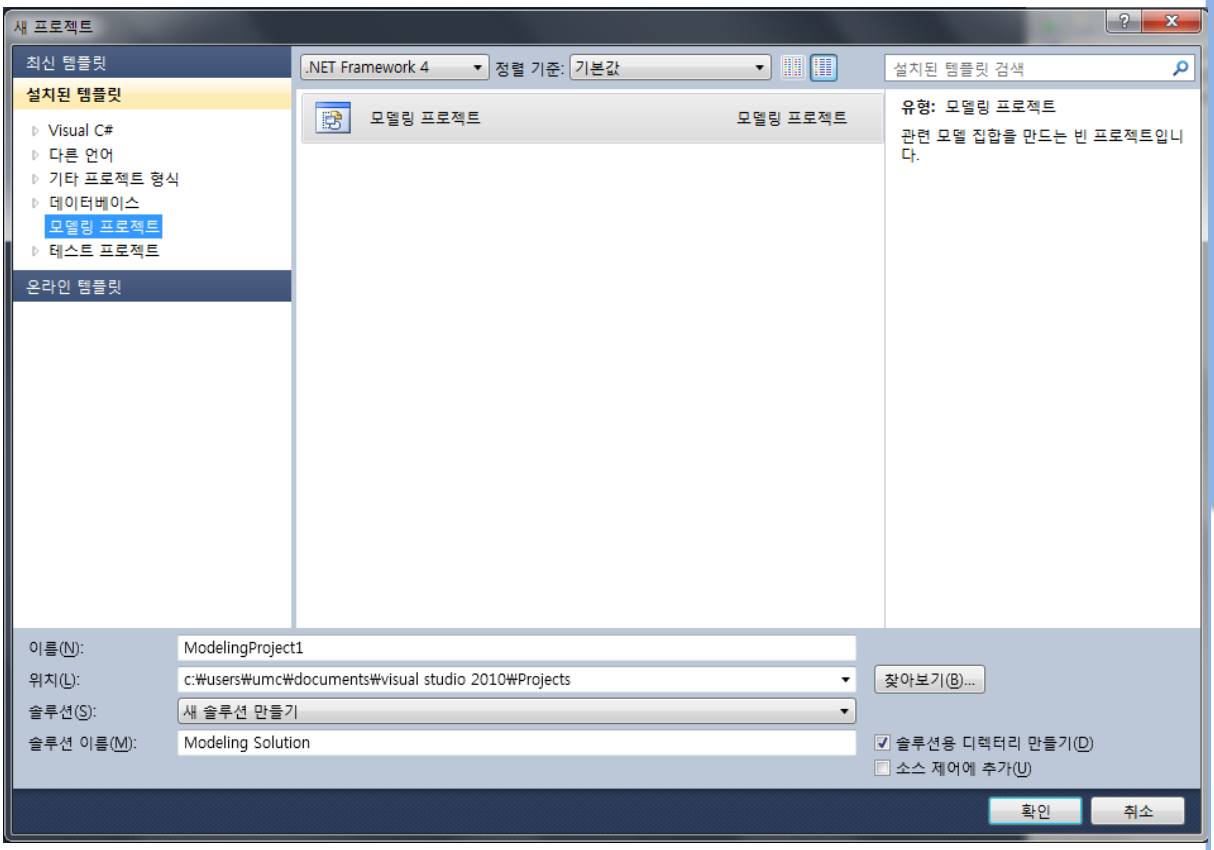
다이어그램을 만들기 전에 먼저 모델링 프로젝트를 어떻게 만드는지 아래의 단계를 따라 하면서 살펴봅시다.

모델링 프로젝트를 생성하기 위해 Visual Studio 2010 을 실행합니다.

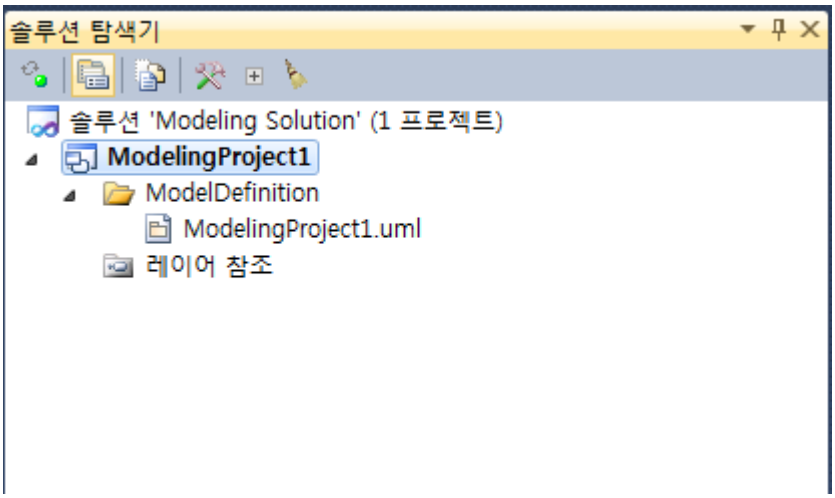
1. 파일->새로 만들기-> 프로젝트를 선택합니다.



2. 프로젝트 템플릿에서 "모델링 프로젝트"를 선택하고, 프로젝트의 이름과 솔루션 이름을 입력하고 확인 버튼을 클릭합니다.



3. 솔루션 탐색기에 모델링 프로젝트가 생성된 것을 확인합니다.



모델링 프로젝트를 정상적으로 생성이 완료가 되었다면 이제부터 모델링을 할 수 있는 환경이 모두 완료되었습니다.

4. 모델링을 하기 앞서

다음 회차에서 Visual Studio 2010 모델링을 배우기 앞서, 많은 개발자들은 모델링을 어떻게 시작하면 좋은지에 대해 고민을 하시는 분들도 많습니다. 모델링의 목적이 복잡하고 추상적인 것으로부터 명확하게 요소를 구분하여 표현하는 것이긴 하지만, 무엇을 명확하게 표현해야 할지도 고민해야 할 부분 입니다.

예를 들어, 구현을 하기 위해 모델링을 하는 것인지... 아니면 그 상위 시스템과 비즈니스 워크플로우를 명확하게 하고자 하는 것인지... 모델링이라는 도구를 잡는 사람들마다 그 목적이 다르고 사용하는 방법도 다를 것이라고 생각합니다.

한 가지 UML 에 대해 잠깐 동안 되집어 생각해 볼 것이 있습니다. 왜 UML 이 성공했는가? 라는 물음입니다. UML 이 추구하고자 했던 것은 UML 을 이용하여 소프트웨어를 설계하거나 객체지향적인 설계에 등의 어떠한 노하우도 포함시키지 않았습니다. 다시 말해, UML 은 [모델의 표기 방법]만을 규정하였습니다.

어떠한 소프트웨어를 만들던지 그 소프트웨어에 포함되는 구성요소나 설계가 완전히 틀릴 수 있지만, UML 을 이용하여 표기법이 통일 된다면 이해 관계자간에 의사소통은 쉬워질 것입니다.

다음 회차에서 다룰 모델링의 표현 방법 등은 잠시 뒤로 미루고 무엇을 모델링할 것인지에 대해 먼저 함께 고민해야 할 것 같습니다.

4.1. 모델링의 관점 및 측면

모델러(모델링의 하는 주체)는 어떤 관점과 어느 측면을 바라보고 다양한 방법으로 모델링을 할 수 있습니다. 이런 모델러의 관점과 측면을 이해할 수 있다면 정확하게 정보를 전달할 수 없을지도 모릅니다.

모델러가 생각하는 관점과 측면으로 올바르게 모델링의 정보가 전달되지 않는다면, 위와 같은 그림을 해석은 매우 달라질 수 있을 것입니다. 마치 착시현상을 일으키는 것처럼, 오히려 잘못된 정보로 논쟁의 대상이 될 수 도 있기 때문입니다. 그렇기 때문에 모델러는 자신의 의도를 명확하게 모델링 하는 것이 좋습니다.

모델링은 다양한 측면으로 나뉠 수 있지만, 아래와 같이 4 개로 정의하기도 합니다.

- **정적인 측면**
  단지 오브젝트들의 구조적인 관계를 나타냅니다. 정적인 측면은 이름에서도 알 수 있듯이 시간적인 흐름은 전혀 포함되지 않습니다.
- **기능적인 측면**
  시스템이 할 수 있는 행동적인 것을 표현하고자 합니다.
- **동적인 측면**
  오브젝트가 시간의 흐름이나 이벤트에 의해 어떻게 상태가 변화해 가는지에 대해 나타냅니다.
- **물리적인 측면**

시스템이 동작하기 위해 필요한 물리적인 것들을 나타냅니다. 서버나 데이터베이스 간의 관계를 어떻게 기술 할지에 대해 나타내기도 합니다.

그리고 모델링을 하려는 관점이 있을 수 있습니다. 이는 보통 "관점(Perspective)" 또는 "레벨(Level)" 이라고 부르기도 합니다. 이러한 관점은 추상화 정도에 따라서 3 가지로 구분을 합니다.

- 개념적 관점
  개념적인 관점은 구현 등의 영역을 전혀 고민하지 않은 채, 실질적인 큰 범위의 영역을 해석하는 관점입니다. 범위에 포함되는 여러 가지 문제를 깊이 있게 이해하는 것이 목적입니다.
- 사양적 관점
  사양적 관점은 문제를 해결하여 완성시키기 위해 필요한 기능 등을 효과적으로 이행할 수 있는 방안을 찾기 위한 관점입니다.
- 구현적 관점
  구현적 관점은 사양적 관점에서 고안된 방안을 실질적으로 구현하기 위한 관점에서 작성합니다.

이러한 관점은 완벽히 독립적인 것은 아닙니다. 필요에 따라 사양적 관점/구현적 관점이 동시에 진행될 수 있습니다.

위와 같은 모델링의 관점과 측면은 개개인에 따라서, 또는 모델링에 참여하는 사람에 따라서 전혀 목적이 일치하지 않을 수 있는 문제가 있습니다. 그렇기 때문에 모델링 이전에 정확하게 목적을 설정하고 개념적 관점의 모델을 작성하는 것이 올바른 시작이라고 할 수 있습니다.

4.2. 모델의 표기 방법

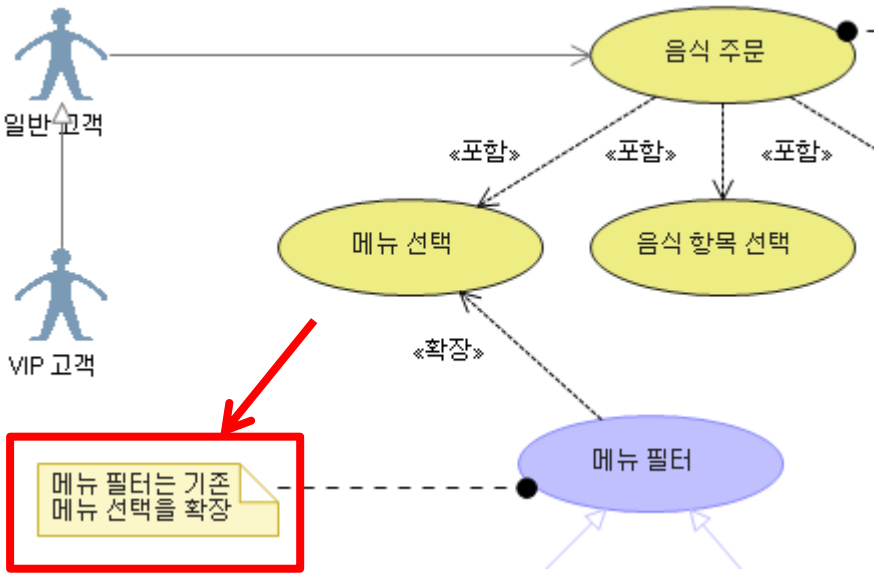
기본적으로 모델링을 하기 위해 모델을 표현하는 공통적인 표기법을 알아보겠습니다. 매우 기본적이고 공통적인 표기 방법이므로 모델링에 들어가기 앞서 가장 중요한 부분이기도 합니다. 기본적인 모델링의 표기 방법을 알아야 모델을 통해 전달하고자 하는 메시지를 최소한 이해할 수 있을 것입니다.

4.2.1. 공통적인 표기 방법

공통적인 표기 방법의 기본적인 도형은 4 가지 입니다. 즉, 이 4 가지만 알면 UML 다이어그램을 읽는 것이 한결 쉽겠지요.

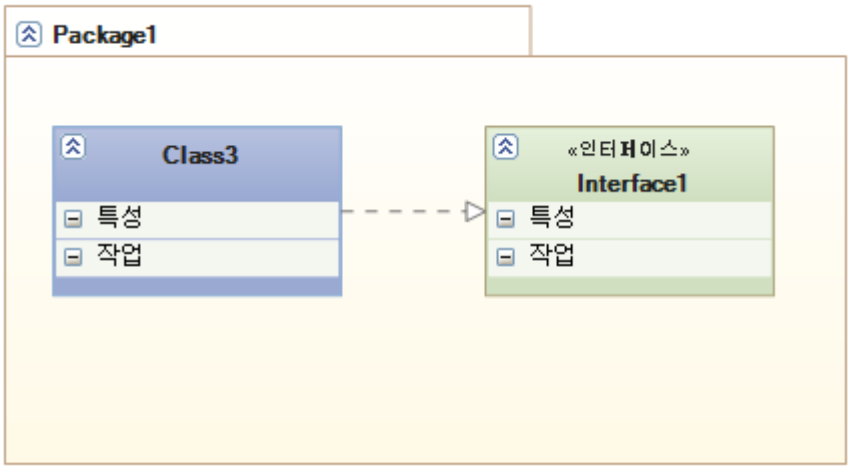
1. 주석(Comment)

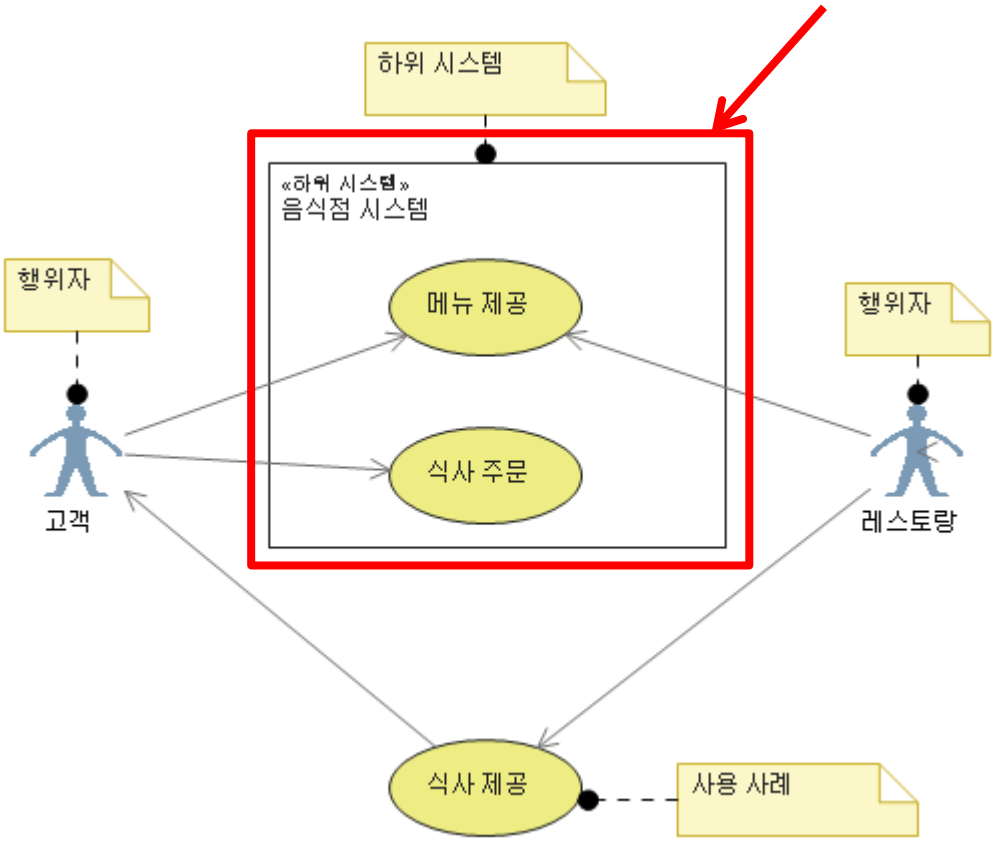
   제약 조건이나 주석이나 정보를 표시하는데 사용하는 기호입니다. 이 기호는 한쪽 귀퉁이가 접혀진 직사각형을 사용합니다. 특정한 모델의 요소와 연관이 되는 주석인 경우 점선으로 연결 됩니다.



2. 패키지

   모델 요소들을 하나로 묶어주는 것으로 폴더 아이콘과 유사하게 표시됩니다. 비슷한 요소의 집합이나 연관성이 있는 요소들을 묶어주는 역할로 주로 사용하기도 하지만, Use Case Diagram 등에서 하위 패키지나 하위 시스템 등을 표현할 때 사용하기도 합니다.

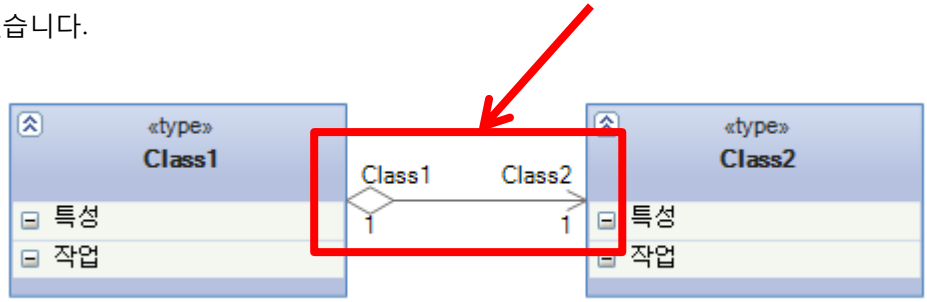


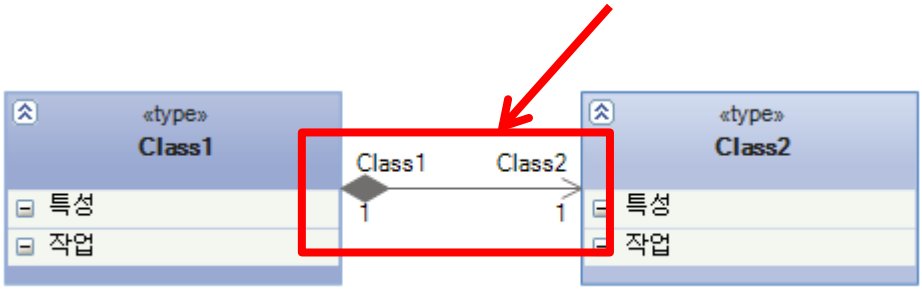


3. 의존 관계

모델 요소간의 의존 관계를 표기하는 방법으로 사용이 됩니다. 모델 요소가 다른 모델의 내부를 알고 있다거나 의존의 방향을 나타내는데 사용합니다.

아래와 같은 속이 비었는지 색이 채워졌는지에 따라서 Aggregation(집합체) 인지 Composition(합성) 관계인지 알 수 있습니다. 예를 들어, Composition(합성) 관계는 Class1 의 정보가 소멸될 때 Class2 의 정보도 함께 소멸되는 것을 미리 짐작할 수 있습니다.





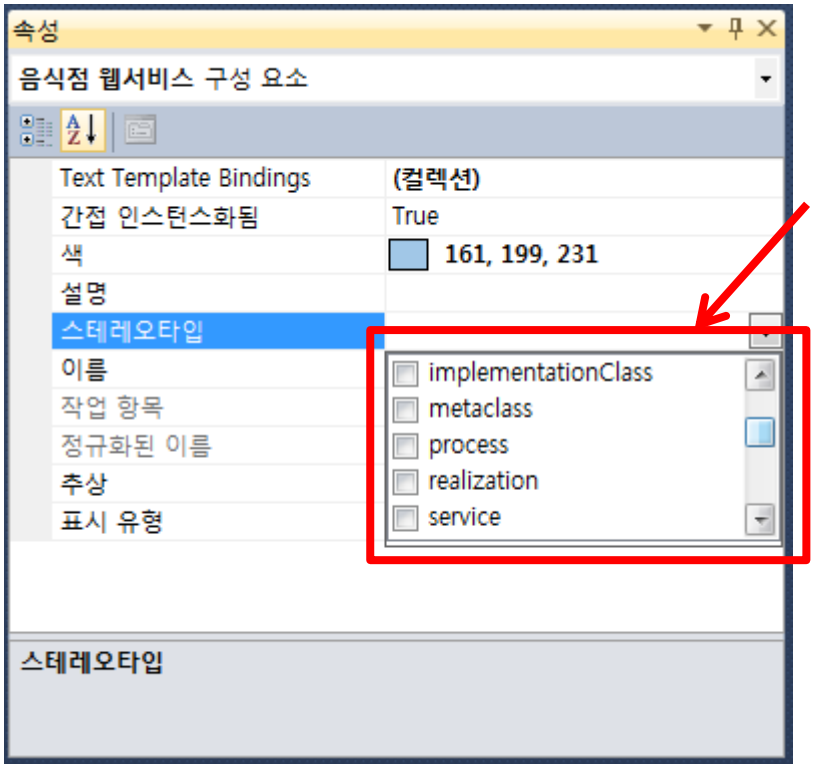
4.2.2. 표기 방법의 확장

UML 에서는 표기 방법을 확장할 수 있는 많은 장치들이 존재합니다. 이런 표기 방법의 확장은 모델링을 좀 더 정확하게 세세하게 정의할 수 있으며, 확장 표기 방법을 통해 유효성을 검증할 수 있는 수단이 될 수 있습니다.

그 중 가장 많이 사용되는 한 가지를 소개할 예정이며, 이것을 아느냐 모르느냐는 모델링의 퀄리티에 큰 차이가 있을 수 있습니다. 그 밖에 Constrains(제약) 도 모델 요소 간의 성립 관계를 나타낼 수 있습니다.

1. Stereotype(스테레오 타입)

   모델 기호의 의미를 새롭게 재정의 하는 방법으로 Stereotype 을 사용할 수 있습니다. 기존적인 Stereotype 은 <<type>>, <<interface>>, <<implementation>> 등이 있으며, 이런 Stereotype 을 요구사항에 따라 새롭게 정의할 수 있습니다.


속성
음식점 웹서비스 구성 요소
Text Template Bindings
(컬렉션)
간접 인스턴스화됨
True
색
161, 199, 231
설명
스테레오타입
이름
작업 항목
정규화된 이름
추상
표시 유형
implementationClass
metaclass
process
realization
service
스테레오타입

5. VISUAL STUDIO 2010 MODELING

Visual Studio 2010 의 모델링 다이어그램은 여러 가지 요구 사항 및 코드를 이해하고 명확하게 의견을 교환하는데 도움을 줍니다. 예를 들면, 사용자의 요구 사항을 Use Case Diagram 이나 Sequence Diagram 은 개발자에게 매우 유용합니다. 또한 시스템적으로는 Component Diagram, Activity Diagram, Layer Diagrams 을 사용합니다.

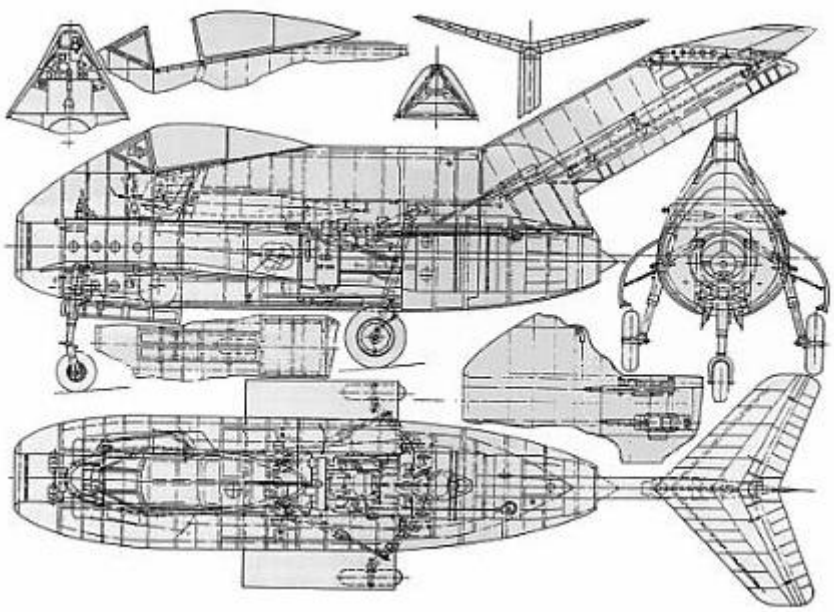
이러한 여러 가지 다이어그램은 이해 관계자를 이해하고 큰 틀을 표현하는데 매우 효과적이지만, 가장 중요한 것은 이것들을 코드로 표현하는 것이며, 특히 Visual Studio 2010 Modeling 은 개발자가 올바르게 모델을 구현할 수 있는 수단이 됩니다. 애자일에서 소스 코드가 산출물이라는 잘못된 강박관념에서 벗어나, 오히려 적절한 시점에 초기에 모델링을 하는 것은 매우 효과적이기도 합니다.

위의 그림에 정말로 아름다운 보름달이 있습니다. 하늘에 구름 한 점 없는 추석에 볼 수 있는 보름달이죠. 지금 독자 여러분은 모두 이 보름달을 바라보고 있지만, 사실 현실세계에서는 반드시 저 보름달을 모두 함께 바라보리라는 보장을 할 수 없을 겁니다.

필자와 독자 여러분들과 함께 저 보름달에 가기 위해서 여러 가지 이야기를 나누면서 제가 저 보름달을 손가락으로 가리키고 있습니다. 전자인 경우 몇몇의 대부분은 저 보름달에 가기 위한

목표로 즐거운 대화를 나눌 수 있지만, 어떤 누군가는 전혀 그렇지 않을 수도 있습니다. 아쉽게도 후자의 몇몇의 사람들은 보름달을 가리키고 있는 제 손가락 끝의 손톱 때를 보면서 잡담을 나누기도 합니다.

이런 현상은 개발하는 조직에서 특히 쉽게 찾아볼 수 있는 현상입니다. 우리 팀이 나아갈 방향과 목표, 그리고 개발해야 할 이상적인 구조를 이야기 할 때, 어떤 누군가는 어떻게 구현할지 머리 속으로 코드만 생각하는 사람들도 있습니다. 반대로 올바른 코드를 작성하고 적절한 리팩토링에 대해 이야기를 할 때, 발전적인 개선할 점에 대한 내용이 아닌 그 이상의 무언가가 잘못됐다고 불평을 하는 사람들도 있습니다. 왜 그럴까요? 서로가 주제에 대한 문제점에 대해서 인식을 하고 있지만, 그것을 바라보는 시각은 전혀 다르기 때문입니다.



분명 필자가 보름달을 가리키면서 야망에 사로잡혀 허황된 꿈을 꾸는 것처럼 보일 수 있지만, 만약 위와 같은 목표를 이룰 수 있는 우주선 설계 도면과 같은 계획을 한다면 필자의 손가락 끝의 손톱 때가 아닌 더 발전적인 방향의 이야기가 될 가능성이 충분하다는 것입니다. (물론 처음부터 완벽한 설계 도면을 만들기란 어렵겠지요.)

분명 개발자와 개발자 또는 개발자와 관리자 간에 발생할 수 있는 이러한 커뮤니케이션의 실수 가운데 모델링을 통하여 서로 공감대를 이루기 쉬운 커뮤니케이션을 할 수 있는 수단이 됩니다. 머릿속에 있는 자신만의 생각을 뚝딱 뚝딱 코드로 완성하여 "짜잔~" 하고 보여주더라도, 완벽한

코드가 아닐 수 있으며 얼마든지 리팩토링 대상이 될 수 있습니다. 자신의 생각을 코드로만 표현할 수 있는 개발자는 안타깝지만 현재의 시대가 원하는 개발자가 아니라고 필자는 생각이 듭니다.

최근의 대부분 개발 도구는 통합 개발 환경을 제공하고 있으며, 특히 Visual Studio 2010 부터는 UML 다이어그램도 지원을 해 주고 있습니다. 다음 단원부터 이런 UML 을 통해 독자 여러분들이 어떻게 활용할 수 있을지도 함께 생각하면서 보시면 더욱 이 백서가 빛을 발휘할 것이라고 생각합니다.

5.1. UML ACTIVITY DIAGRAMS (동작 다이어그램)

Activity Diagrams 은 소프트웨어의 일련의 프로세스나 업무/비즈니스 프로세스에 대한 일련의 워크플로우를 나타냅니다. 이런 프로세스나 워크플로우의 흐름을 제어하거나 분기, 조인 등의 프로세스를 기술할 수 있습니다. 또한 소프트웨어에서 사용되는 프로토콜을 정의하거나 컴포넌트 상호간에 상호작용을 이해하거나 기술하는데 사용할 수 있습니다.

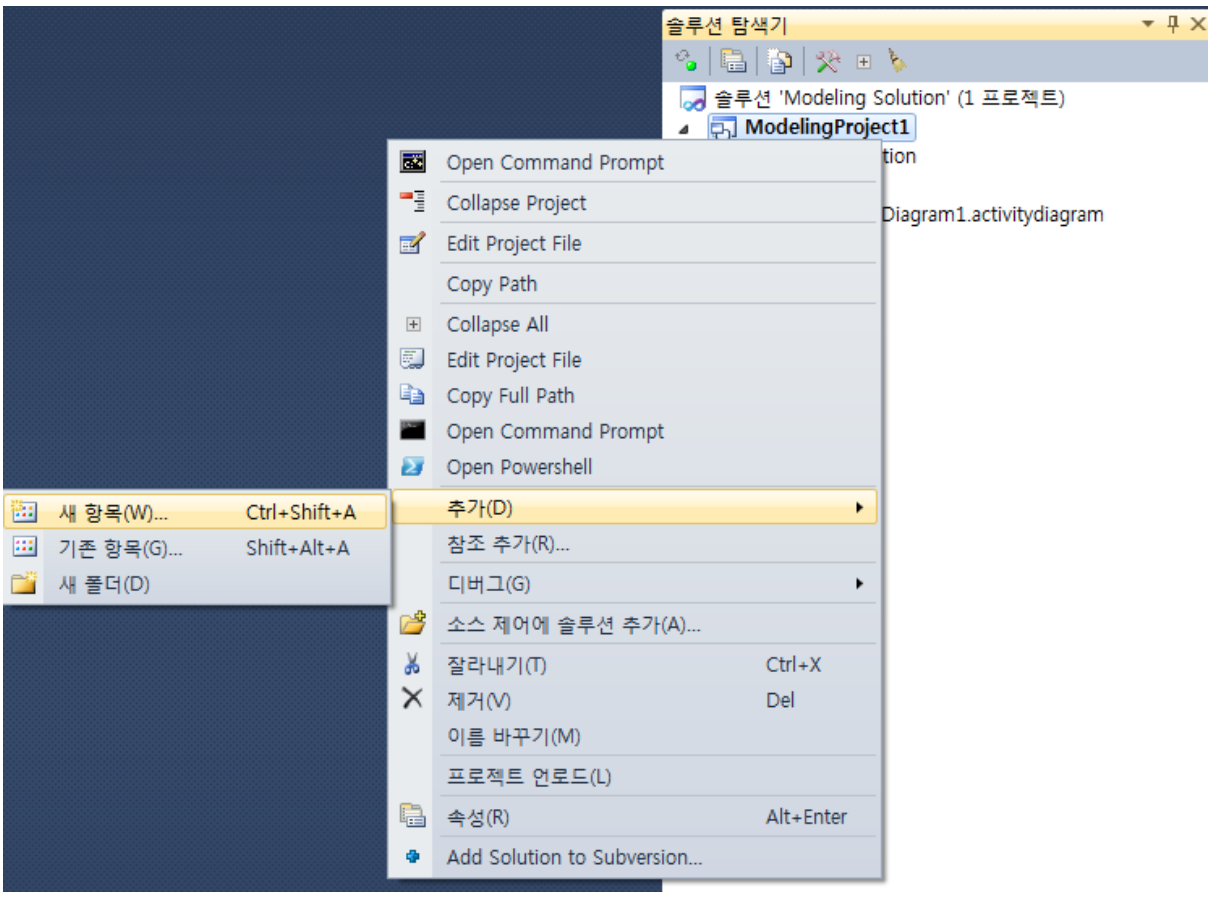
이러한 Activity Diagrams 을 사용하면 여러 가지 이점이 있습니다.

- 시스템 내부 동작과는 별개로 외부 동작에 집중할 수 있습니다.
- 자연어를 사용하는 것에 의한 모호성을 줄일 수 있습니다.
- 사용자/개발자/테스트 등 일관적인 용어를 정의할 수 있습니다.
- 요구 사항의 차이점의 불일치를 줄일 수 있습니다.
- 요구 사항 변경을 처리하는 시간을 줄일 수 있습니다.

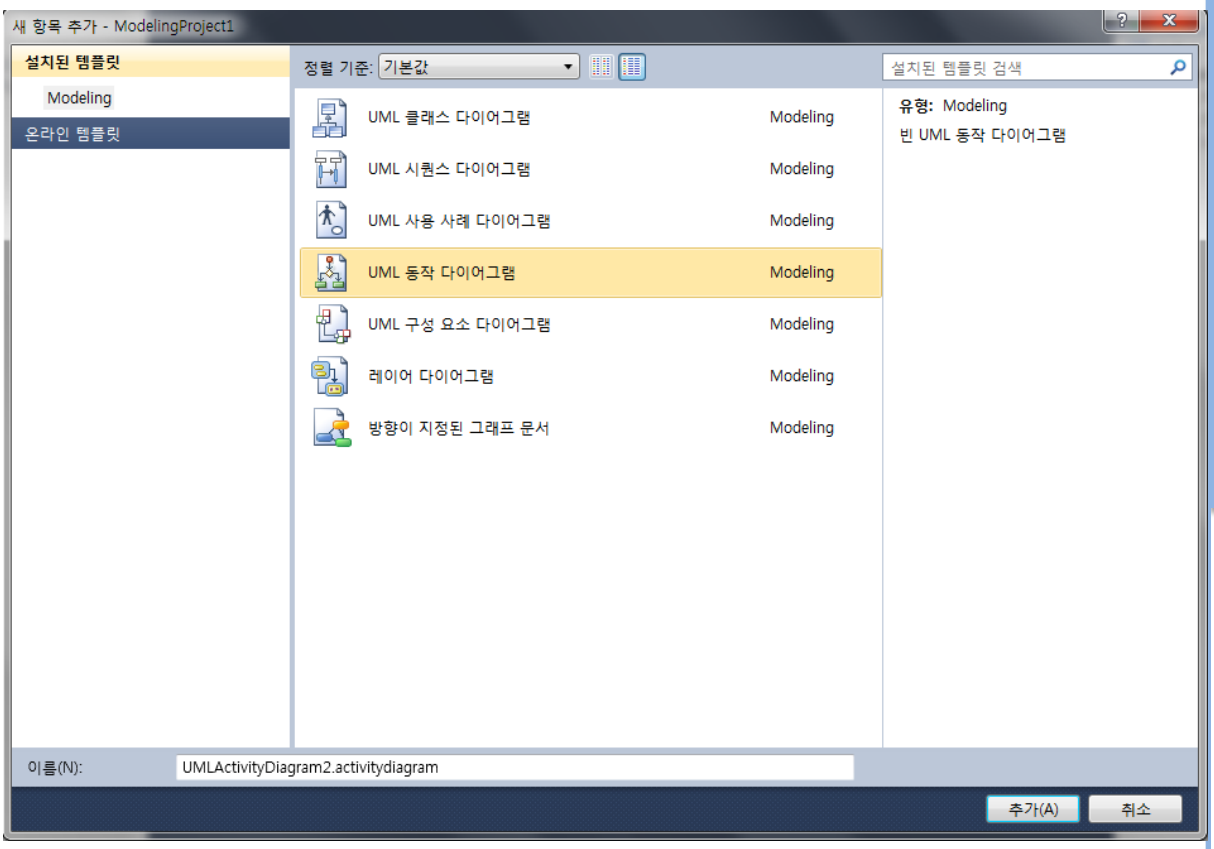
5.1.1. ACTIVITY DIAGRAMS 만들기

Activity Diagrams 은 모델링 프로젝트에서 새로운 항목을 추가하여 생성할 수 있습니다.

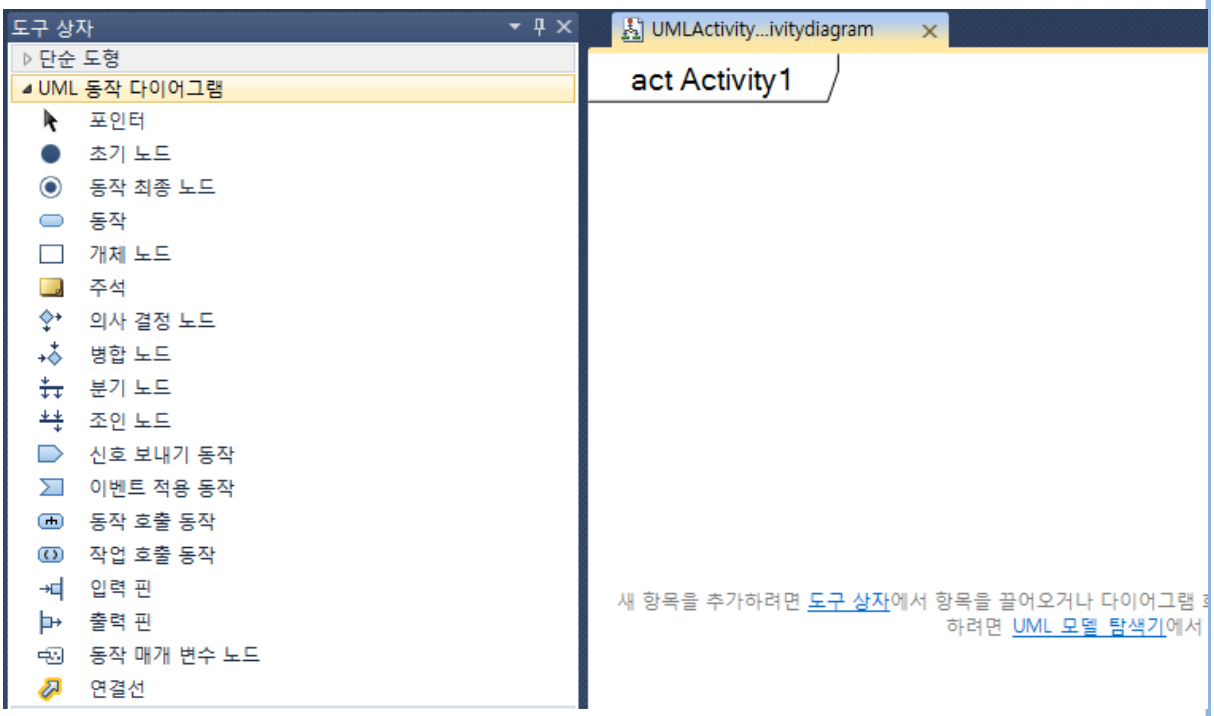
1. 솔루션 탐색기에서 모델링 프로젝트에서 마우스 오른쪽 버튼을 클릭하여 추가-> 새 항목을 클릭합니다.



2. 새 항목 추가 창에서 "UML Activity Diagrams"을 선택하고 추가 버튼을 클릭하여 완료합니다.

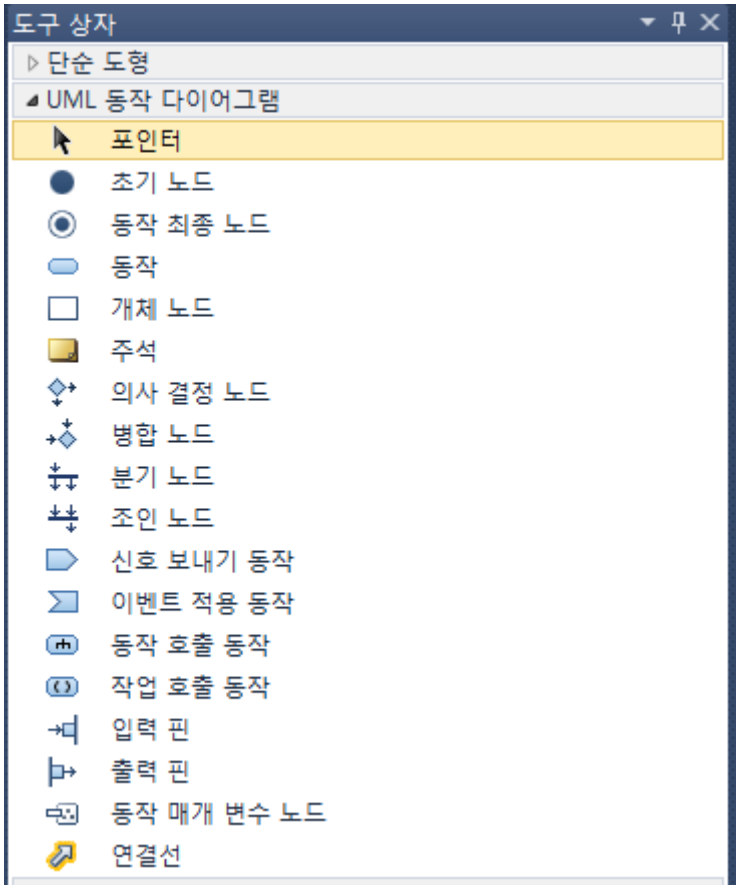


3. Activity Diagrams 이 완성되었습니다.



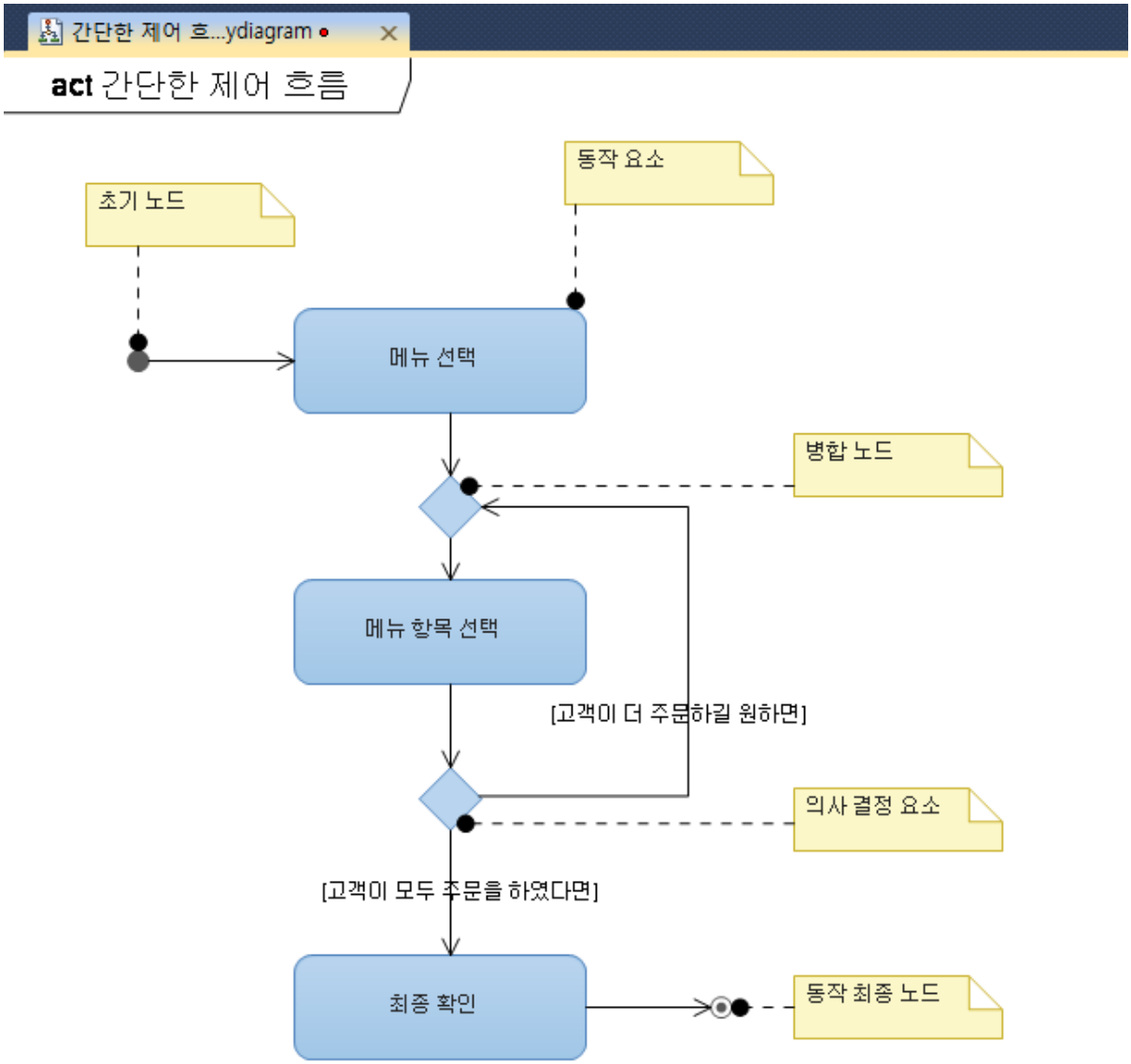
5.1.2. 간단한 흐름 제어

Activity Diagrams 에서 하나의 요소는 하나의 작업을 의미하게 됩니다. 그리고 요소를 연결선으로 연결함으로써 흐름의 제어를 구성할 수 있습니다.



도구 상자에 여러 가지 요소가 포함이 되어 있고, 이 요소를 Activity Diagrams 의 디자인 화면에 추가하기 위해서 도구 상자에서 요소를 선택하고 디자인 화면으로 드래그&드랍 동작을 수행합니다.

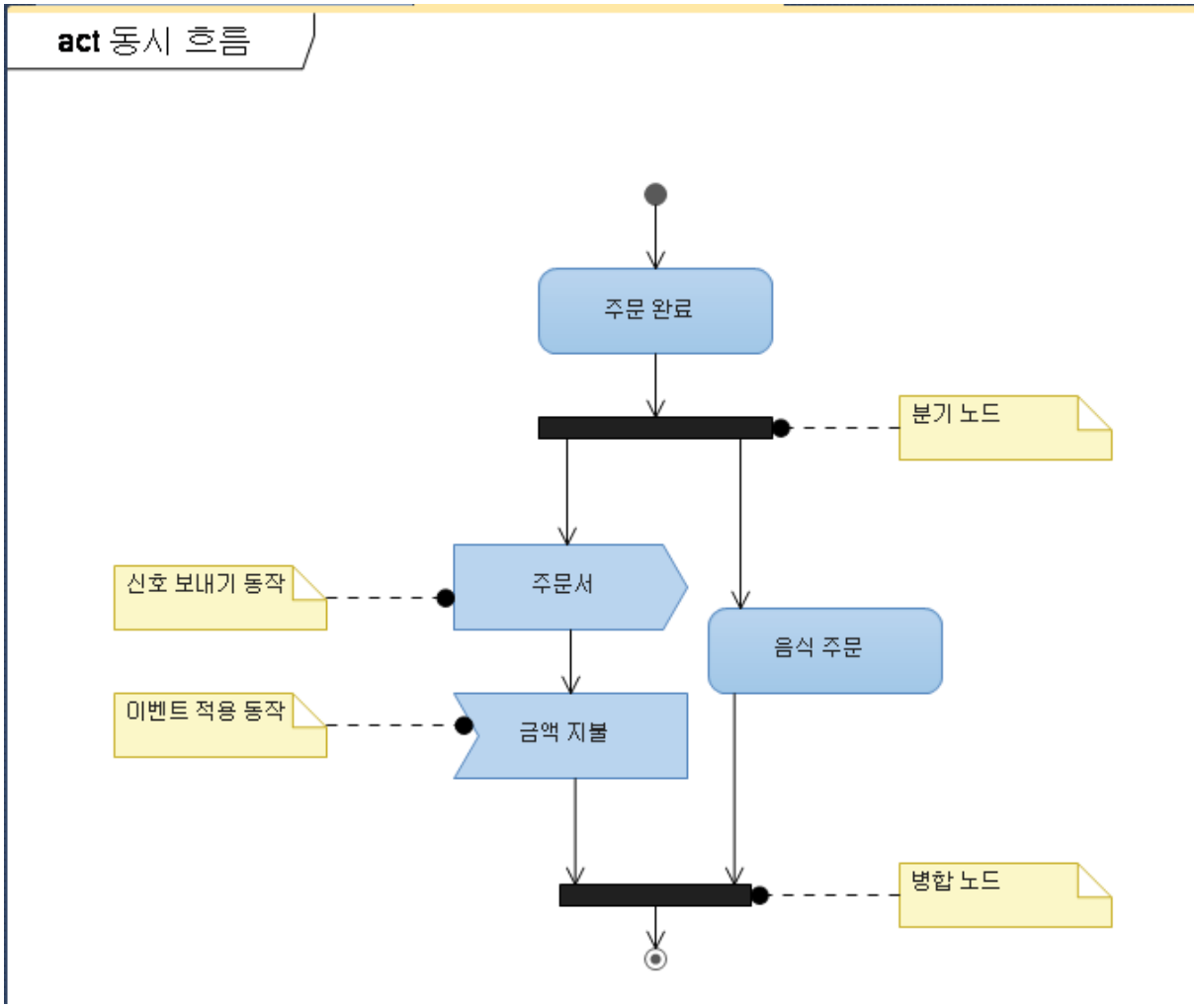
아래의 간단한 제어 흐름 예제는 한 동작에서 다음 동작의 흐름을 나타냅니다. 순차적으로 흐름이 전달되는 방식으로 일반적으로 이전 동작이 완료된 후에 다음 동작을 수행합니다. Activity Diagrams 은 동작에 대한 흐름을 기술하지만 동작이 실행되거나 동작 사이의 제어가 전달되는 방식은 기술하지 않습니다.



위의 간단한 제어 흐름에서 초기 노드를 시작하여 동작 최종 노드까지 하나의 비즈니스 프로세스가 완료되는 것을 알 수 있습니다. 그 중 의사 결정 요소를 통해 조건에 대한 흐름이 분기가 되며, 병합 노드는 분기된 흐름을 병합하는 역할을 수행합니다.

5.1.3. 동시 흐름

동시 흐름은 동시에 동작하는 요소를 기술할 수 있습니다. 다음은 동시 흐름을 기술하는 다이어그램의 예 입니다.

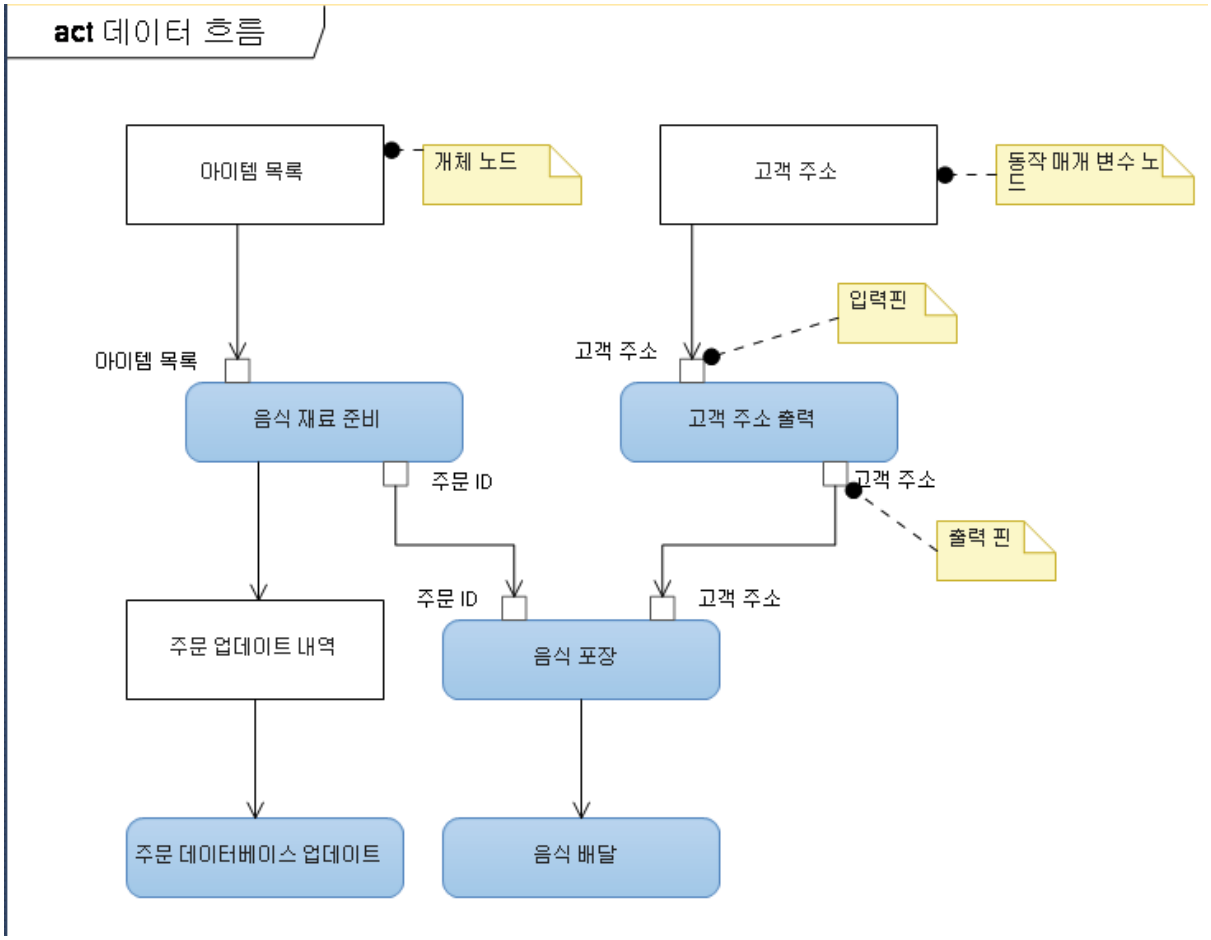


처음의 분기 노드는 분기되는 요소에 대해 각 토큰을 생성하여 마지막 병합 노드에서 들어오는 각 토큰에 대해 동시 흐름을 단일 흐름으로 병합하게 됩니다.

"주문서"에 해당하는 신호 보내기 동작은 메시지의 신호를 보냅니다. "금액 지불" 이베트 적용 동작은 다음 동작을 계속하기 전에 메시지나 신호를 기다리는 동작을 수행합니다.

5.1.4. 데이터 흐름

데이터 흐름은 각 동작간에 데이터 흐름을 기술하게 됩니다. 다음은 데이터 흐름의 예 입니다.



위의 개체 노드는 흐름에 따라 전달되는 데이터를 의미 합니다. 이 개체 노드는 입력 핀과 출력 핀을 통해 데이터를 입력 받거나 작업의 처리에 대해 출력을 나타내며, 동작 매개 변수 노드는 동작에 의해 데이터를 생성하거나 받을 때 사용합니다.

5.2. UML USE CASE DIAGRAMS (사용 사례 다이어그램)

Activity Diagrams 에 정의되는 시스템 내부적인 워크플로우나 업무/비즈니스 등을 정의하는 것이 초점이라면, Use Case Diagrams 은 시스템이 구현할 업무/비즈니스 프로세스를 다루는 것이 다릅니다.

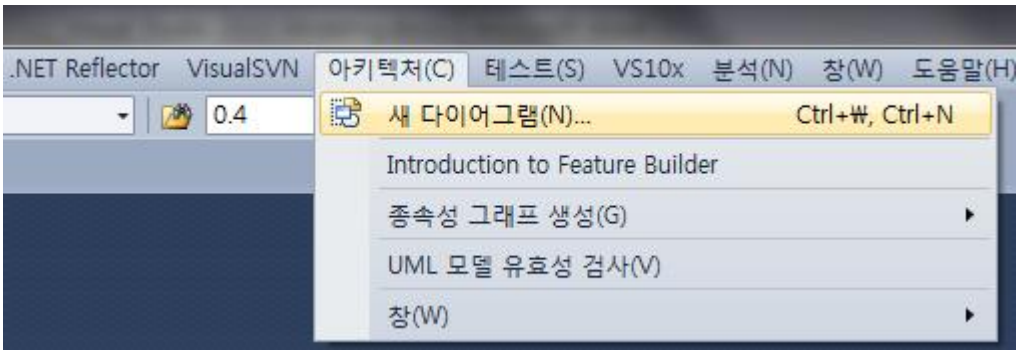
Use Case Diagrams 은 어플리케이션 또는 사용자가 어플리케이션의 시스템으로 수행할 수 있는 작업에 대해 요약합니다. 특히 Use Case Diagrams 은 사용자의 요구 사항을 기술하기 위한 중심 역할을 하게 됩니다. 하지만 이러한 요소 사항은 너무 자세하게 기술하지 않으며 별도의 다이어그램으로 세부적인 내용을 기술할 수 있습니다.

Use Case Diagrams 을 사용하면 다음과 같은 이점이 있습니다.

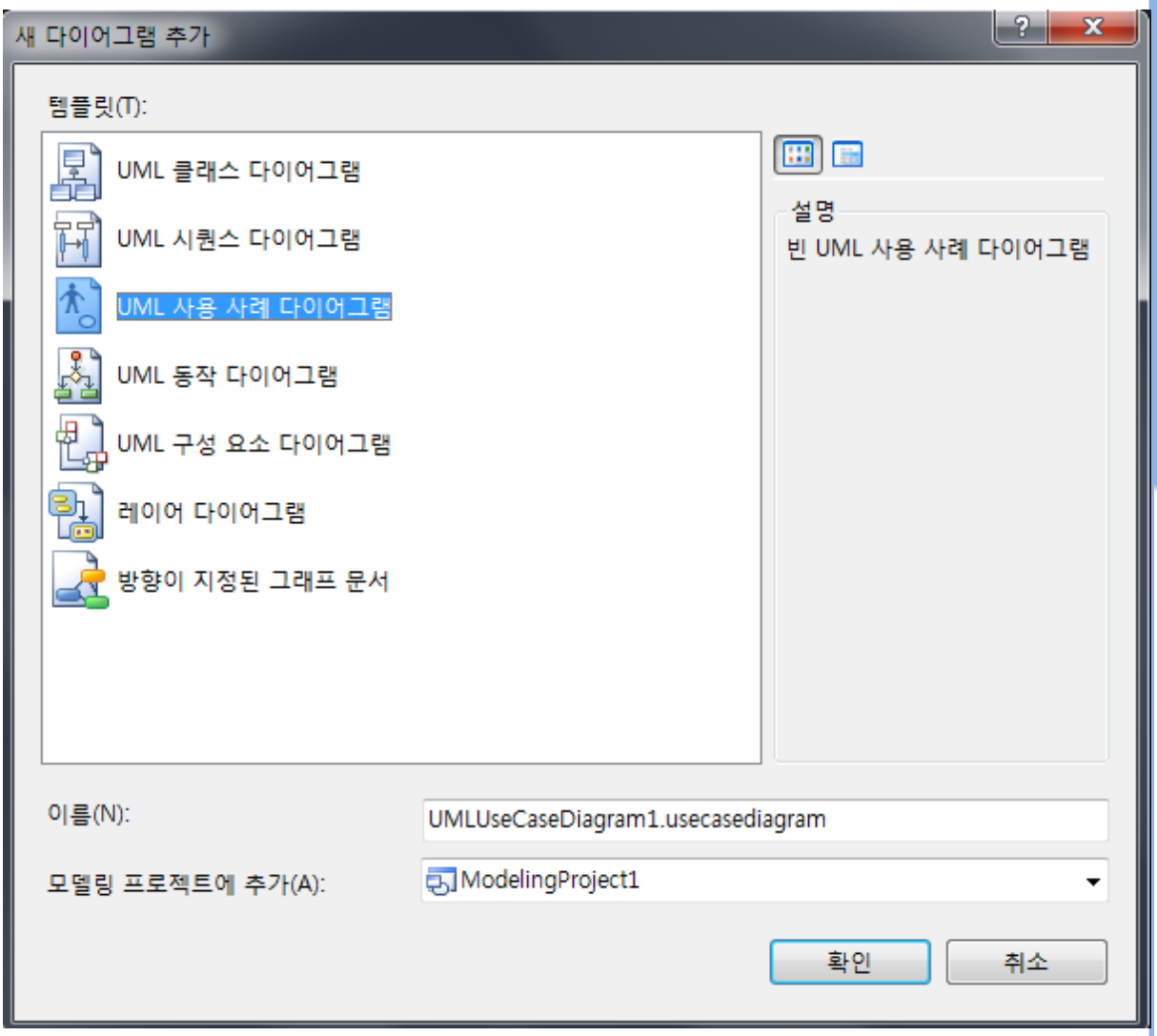
- 시스템이나 어플리케이션이 사용자/조직 또는 외부 시스템과 상호 작용하는 흐름의 이해가 쉽습니다.
- 해당 행위자가 달성해야 하는 목표를 쉽게 이해할 수 있습니다.
- 시스템의 범위를 이해할 수 있습니다.

5.2.1. USE CASE DIAGRAMS 만들기

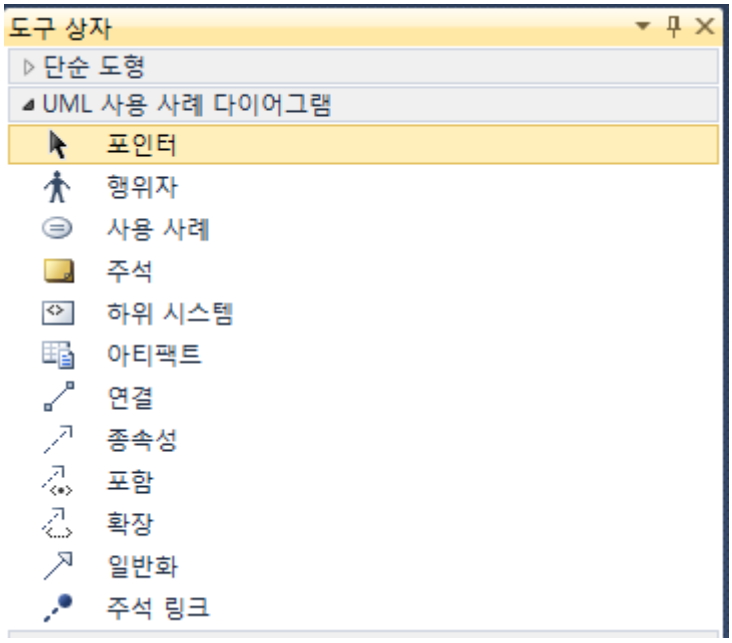
1. 메뉴의 아키텍처->새 다이어그램을 클릭합니다.



2. 새 다이어그램 추가 창에서 "UML Use Case Diagrams"을 선택하고 확인 버튼을 클릭합니다.

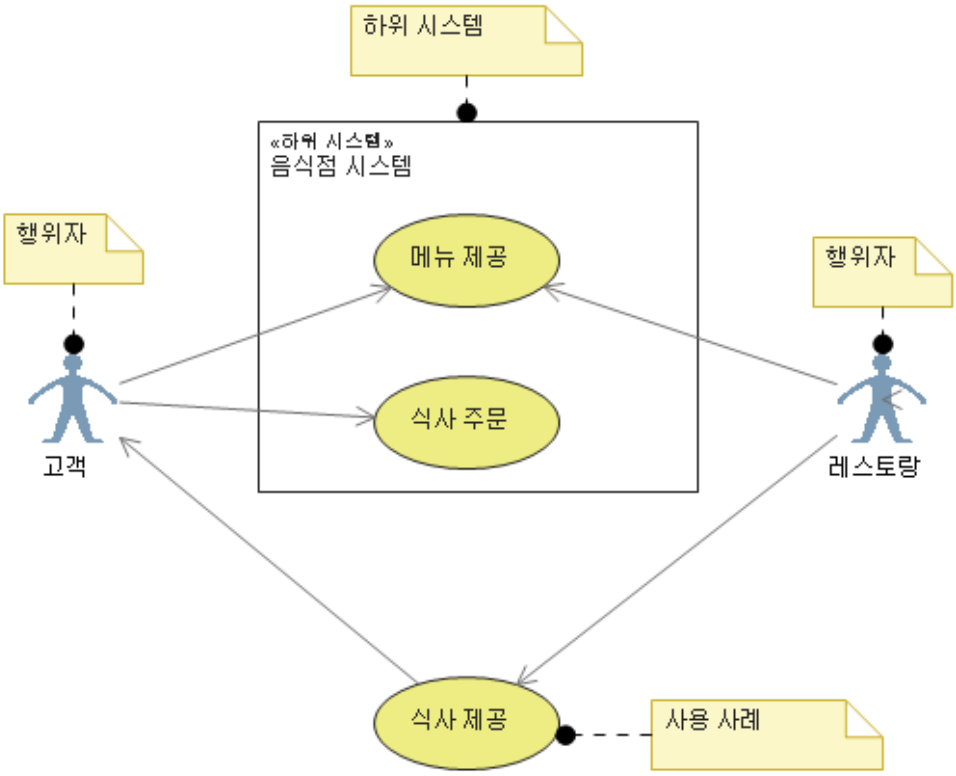


Use Case Diagrams 이 제공하는 도구 모음은 다음과 같습니다.



5.2.2. 행위자, 사용 사례, 하위 시스템

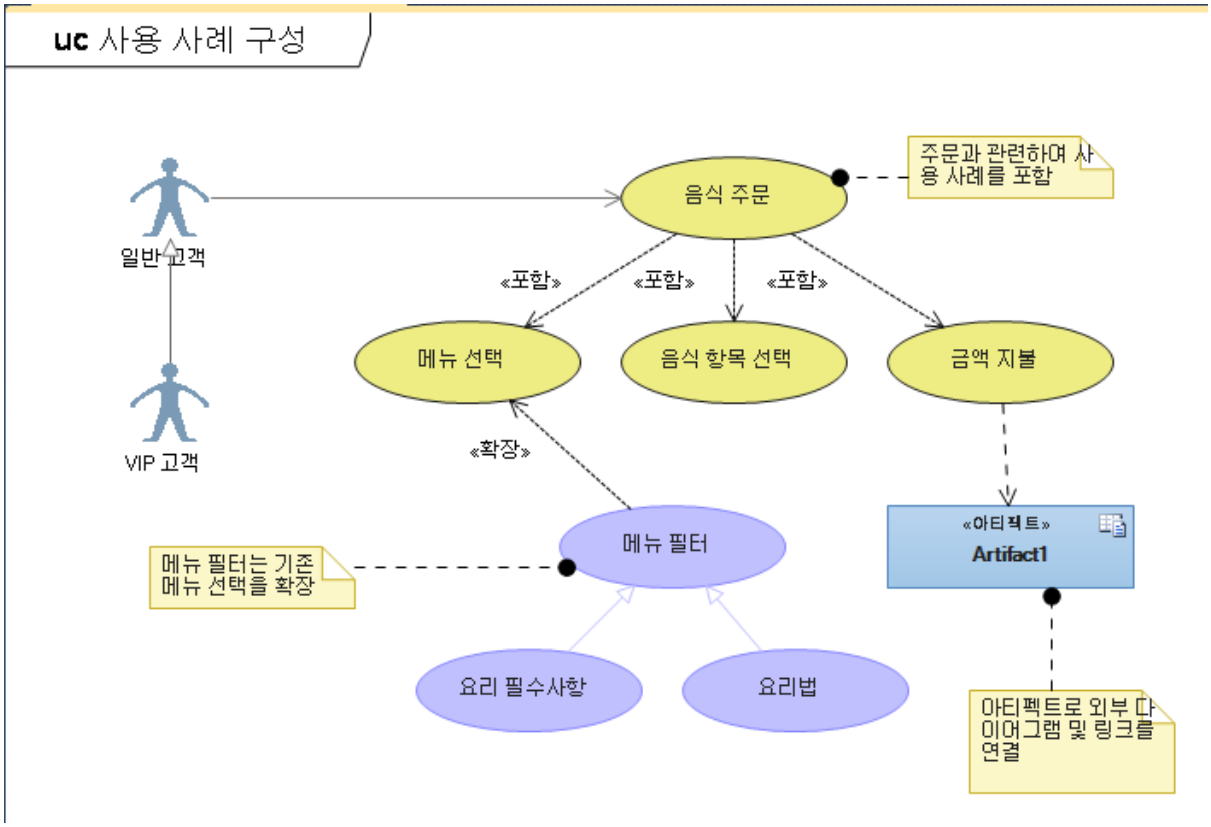
행위자, 사용 사례, 하위 시스템은 행위자가 하위 시스템에 참여하여 사용 사례를 만드는 예제입니다.



각 행위자인 고객, 레스토랑 중 고객은 레스토랑이 제공하는 음식점 시스템에 참여하게 됩니다. 레스토랑 행위자 또한 음식점 시스템에 참여하여 메뉴를 주문 받고, 고객 행위자에게 적절한 서비스를 제공한다는 사용 사례를 나타냅니다.

5.2.3. 사용 사례 구성

사용 사례 구성은 사용 사례 요소 간에 흐름이 연결되거나 상속, 포함, 아티펙트에 대한 예를 보여 줍니다.



일반 고객과 VIP 고객은 "일반화" 요소를 이용하여 상속을 구성합니다. 음식 주문은 음식 주문과 연관되는 사용 사례를 포함하도록 구성하였고, 메뉴 필터는 메뉴 선택 사례를 확장하고 메뉴 필터는 각 요리 필수사항과 요리법을 상속하게 됩니다.

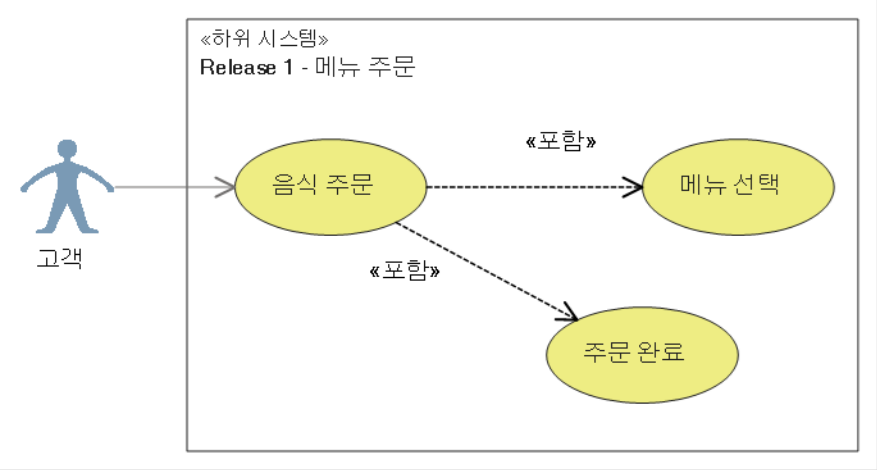
그리고 사용 사례의 아티팩트는 다른 다이어그램이나 외부 링크를 제공합니다. 또는 솔루션 폴더에서 솔루션 항목을 드래그&드랍하게 되면 그 문서에 대한 아티팩트가 자동으로 생성합니다.

이러한 형태로 시스템을 작동하는 방법과 이 시스템과 교류하는 흐름을 나타내며, 구현 레벨의 좀 더 상위 고수준의 관계를 이해하는데 도움이 됩니다.

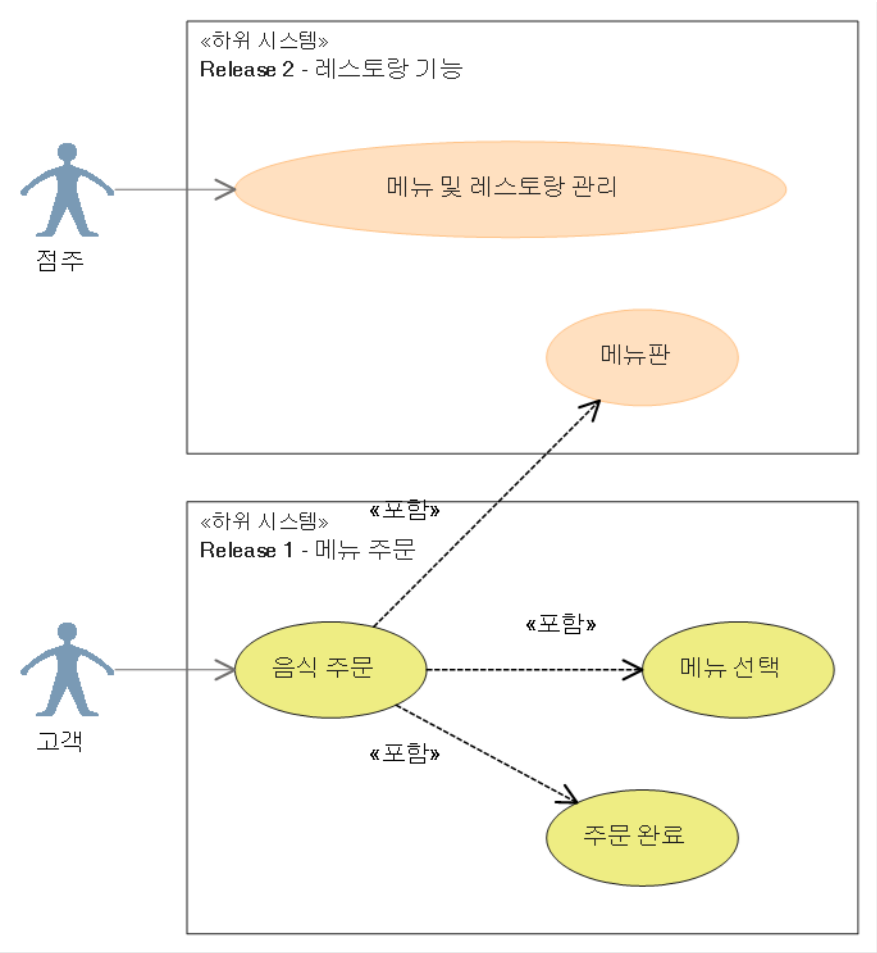
5.2.4. 시스템의 버전 정보 기술하기

시스템의 버전 정보를 기술하기 위해 단계적으로 기능 또는 사용자 측면의 사례를 기술할 수 있습니다.

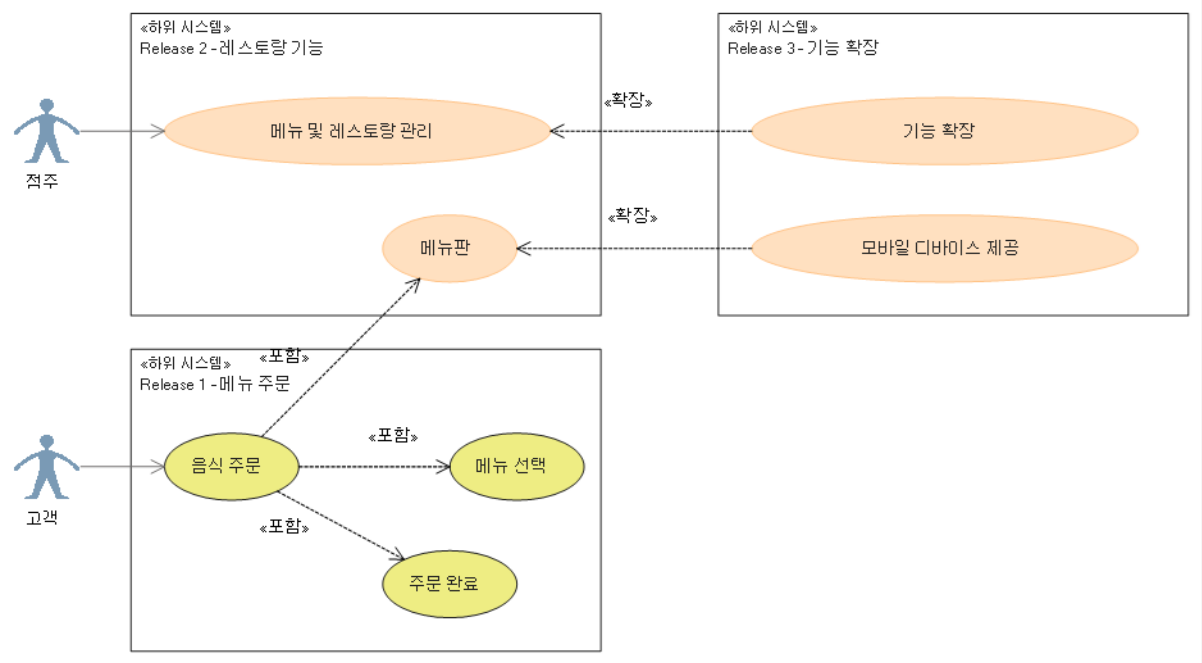
아래와 같이 Release 1 에서는 메뉴 주문 기능을 구현한다는 것으로 이해할 수 있습니다. 음식 주문을 위해 메뉴를 선택하고 주문을 완료하는 단순한 기능의 범위만 제공한다는 것을 알 수 있으며, 이는 설계자 외에 사용자도 명확하게 Release 1 의 시스템을 이해할 수 있을 것입니다.



아래와 같이 Release 2 에서는 기존의 메뉴 주문 기능을 포함하도록 기능을 확장한다는 내용이 포함이 됩니다. 점주에 의해 레스토랑의 메뉴와 레스토랑 관리 기능을 포함시켜 레스토랑의 시스템을 이용하여 메뉴를 선택하고 금액을 지불하거나 메뉴를 관리하는 범위임을 이해할 수 있습니다.



아래와 같이 Release 3 에서는 기본적인 기능을 확장하고 모바일 디바이스를 통해 메뉴를 제공하는 시스템의 기능을 범위로 한다는 것을 알 수 있습니다.



이렇게 Use Case Diagrams 을 이용하여 시스템의 버전 정보를 기술하기 위한 용도로 사용이 가능하며 이렇게 함으로써 점진적인 시스템 구축의 전체 윤곽을 이해 관계자 간에 이해하는데 많은 도움을 줄 수 있습니다.

5.2.5. 아티펙트의 문서를 열기 및 연결

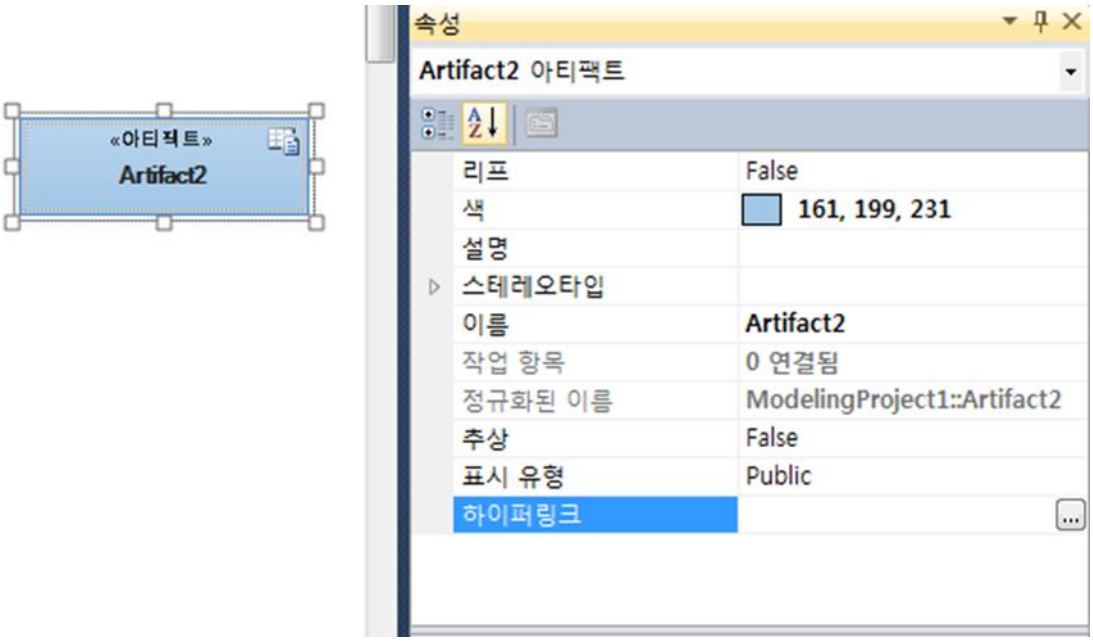
아티펙트는 다양한 방법으로 연계되는 다이어그램 및 문서, 그리고 웹 문서 등을 포함시킬 수 있습니다. 다음은 여러 가지 방법으로 아티펙트를 연결하거나 추가하는 방법입니다.

A. 아티팩트의 연결된 문서를 열려면?
아티펙트의 요소에서 마우스를 두 번 클릭합니다.

B. 솔루션에 포함된 문서를 아티펙트로 추가하려면?
솔루션에 포함된 항목을 Use Case Diagrams 으로 드래그&드랍합니다.

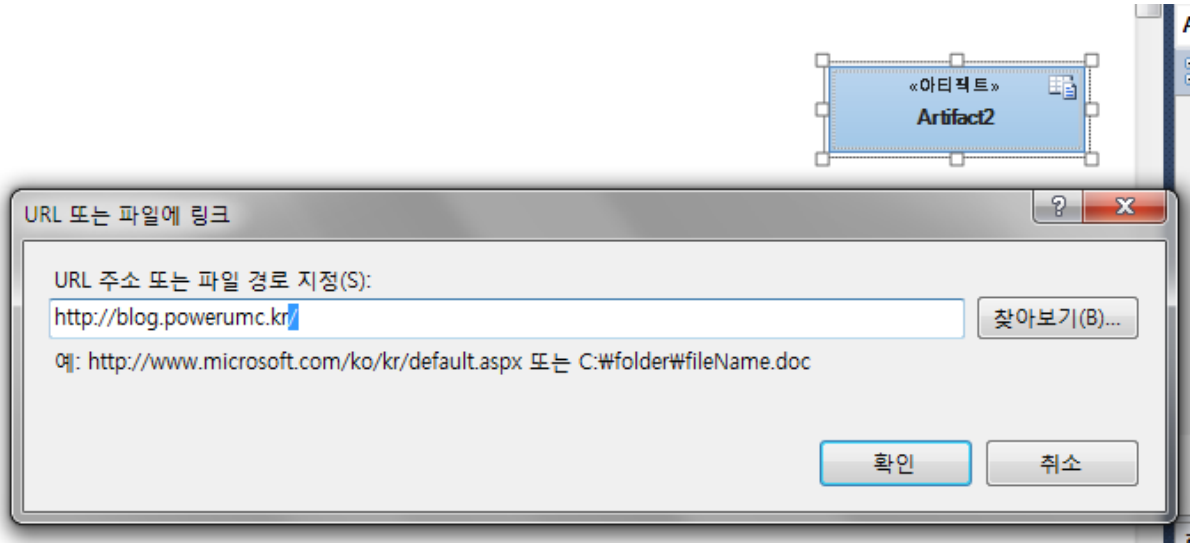
C. Word 및 PowerPoint 문서를 추가하려면?
아티펙트를 선택하여 속성 창에서 하이퍼링크 항목을 클릭하여 추가하고자 하는 문서를

찾아 선택합니다.



D. HTTP/HTTPS 또는 공유 문서를 열려면?

아티펙트를 선택하여 속성 창에서 하이퍼링크를 선택합니다. "URL 또는 파일에 링크" 창에서 링크하고자 하는 HTTP/HTTPS 링크를 입력합니다. 만약 OneNote 의 공유 문서에 링크를 원하면 onenote:// 의 링크를 입력하면 됩니다.



5.3. UML COMPONENT DIAGRAMS (구성 요소 다이어그램)

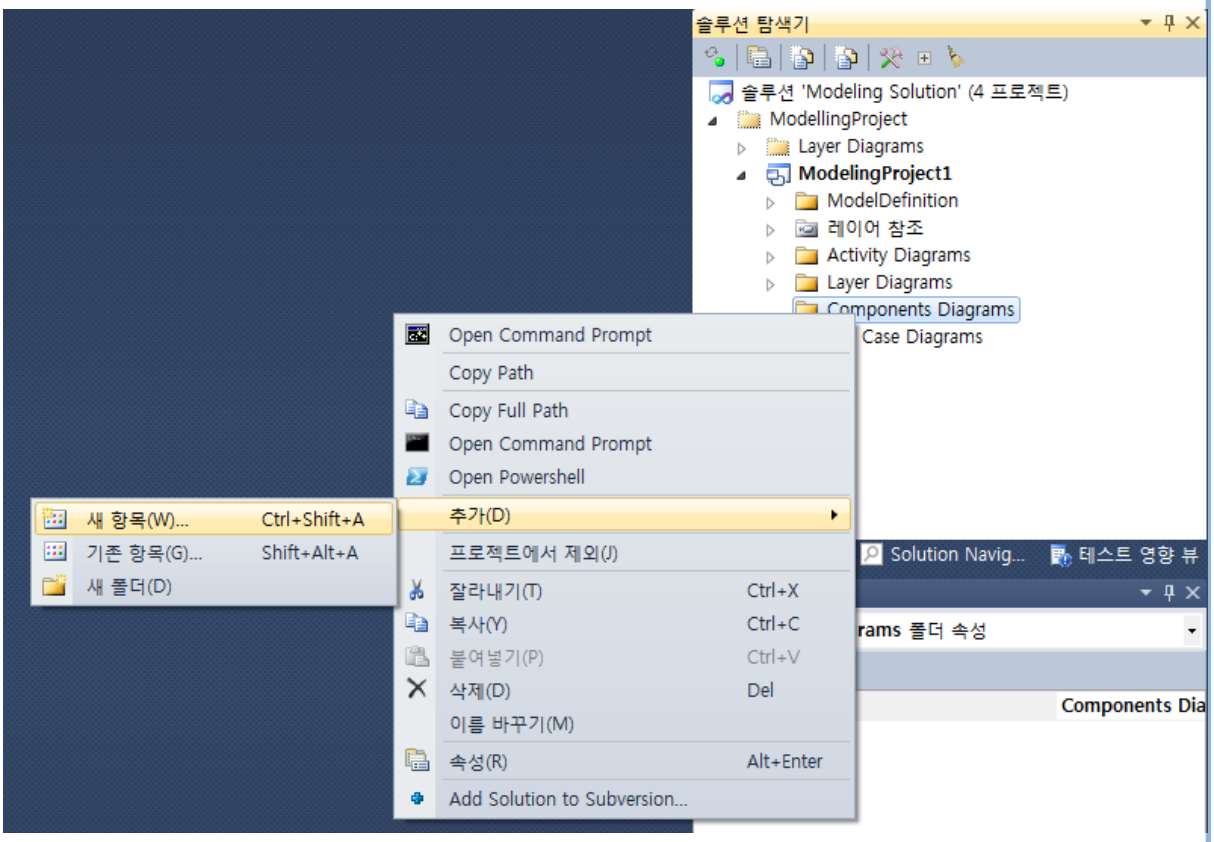
Component Diagrams 은 소프트웨어 시스템의 각 구성 요소를 디자인할 수 있습니다. 시스템 구조에 따른 컴포넌트와 인터페이스를 확인하고 이들 컴포넌트간에 서비스의 동작을 이해하는데 중요한 역할을 합니다.

Component Diagrams 을 이용하면 다음과 같은 이점이 있습니다.

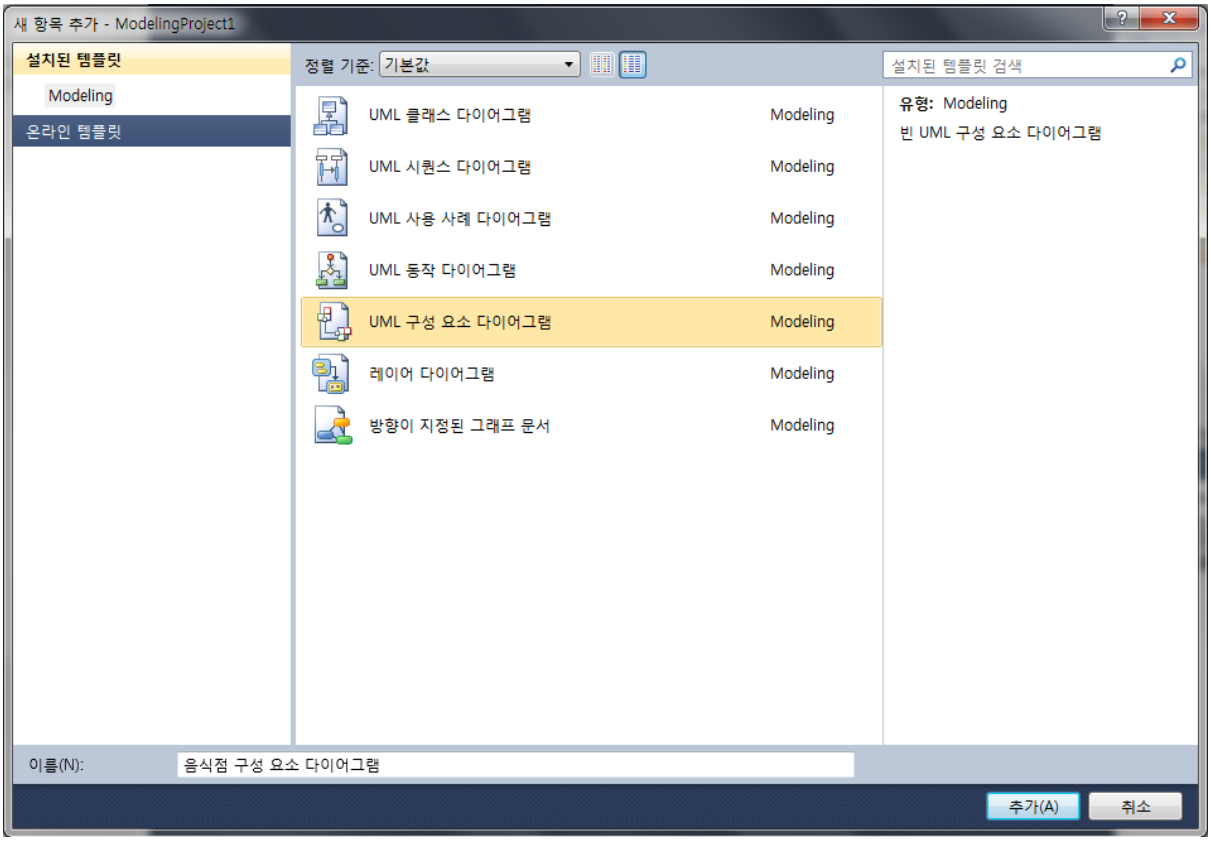
- 기존 디자인을 이해하거나 새로운 디자인을 하는데 용이합니다.
- 시스템이 제공하는 인터페이스와 필요한 인터페이스가 잘 정의되면 구성 요소를 분리하여 개선하거나 변경에 쉽게 대처할 수 있습니다.

5.3.1. COMPONENT DIAGRAMS 만들기

1. 모델링 프로젝트에서 마우스 오른쪽 버튼을 클릭한 후, 추가->새 항목을 클릭합니다.



2. 새 항목 추가 대화 상자에서 UML Component Diagrams 을 선택하고 추가를 클릭합니다.



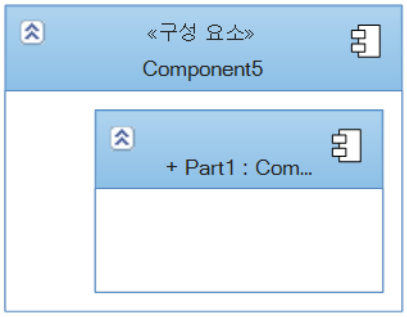
Component Diagrams 이 제공하는 도구 상자의 요소는 다음과 같습니다.

5.3.2. 음식점 COMPONENT DIAGRAMS 만들기

간단하게 어떤 구성 요소가 필요한지 간단하게 다음과 같이 구성 요소를 배치 합니다. 주방 서버는 파트로 구성되어있습니다.

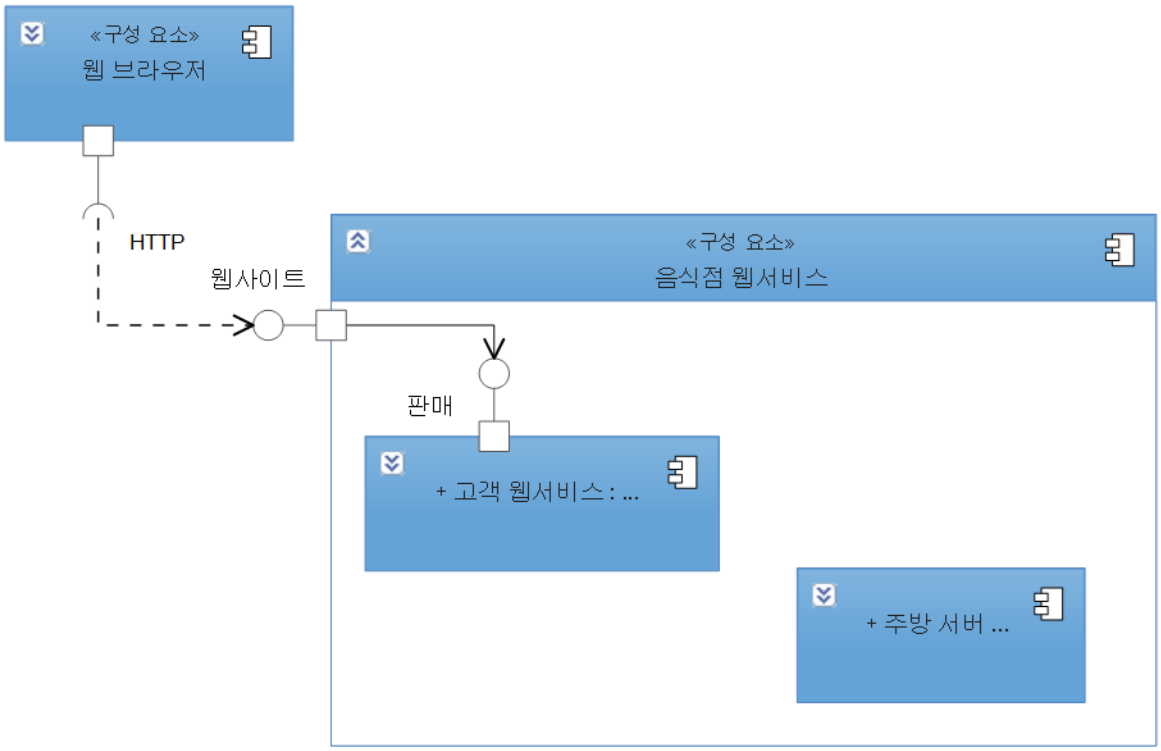
파트 구성 요소를 만들려면?

도구 상자에서 구성 요소를 클릭하고 디자인에 포함된 구성 요소 하나를 클릭합니다. 그러면 구성 요소 안에 파트가 아래와 같이 파트가 추가됩니다.

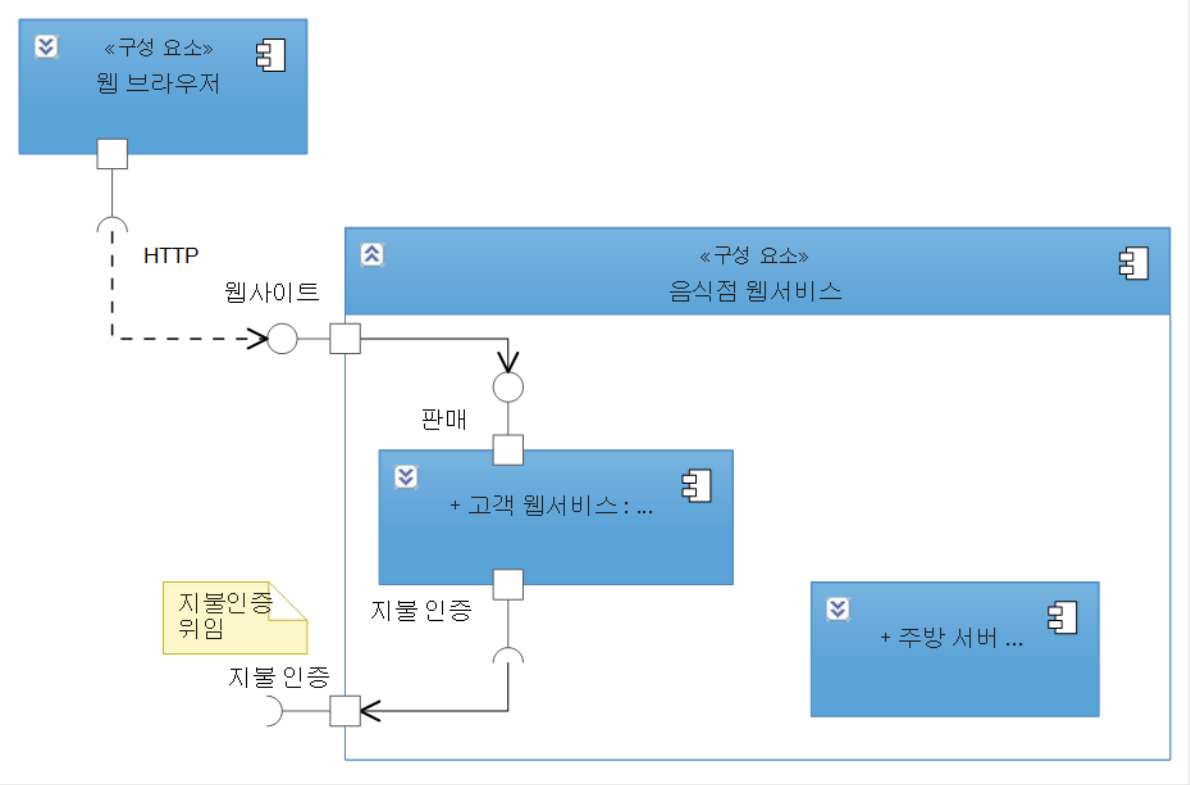


각 구성 요소간에 인터페이스 포트가 필요한데, 이 인터페이스 포트는 다른 구성 요소간의 흐름과 데이터를 표현하는데 사용합니다.

웹 브라우저 구성 요소는 필요한 인터페이스로 HTTP 인터페이스를 생성하였습니다. 음식점 웹사이트와 고객 웹 서비스의 제공된 인터페이스를 통해 웹 브라우저가 음식점 웹서비스에 접근할 수 있도록 구성합니다.



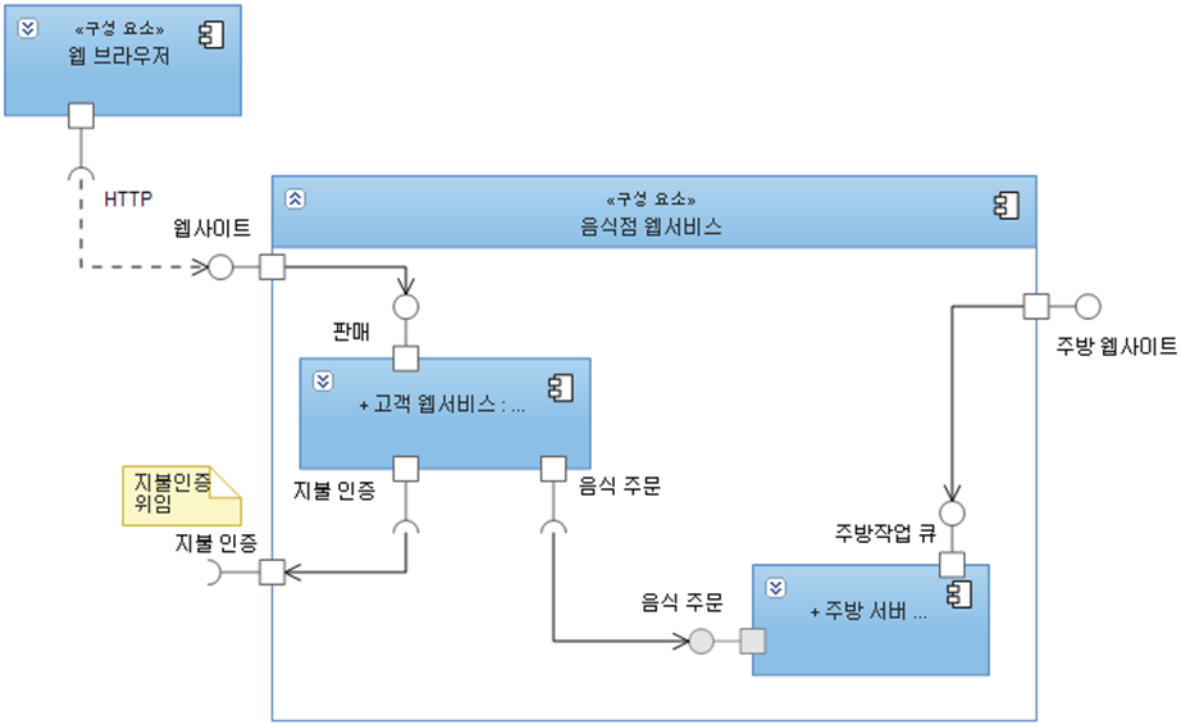
고객 웹 서비스는 필요한 인터페이스가 필요하며, 이 컴포넌트를 외부 시스템과 연결하기 위해 음식점 웹서비스도 지불 인증에 대한 필요한 인터페이스가 존재합니다. 고객 웹 서비스는 이 필요한 인터페이스를 통해 외부에서 결제 시스템을 제공하는 인터페이스로 연결하기 위해 이 필요한 인터페이스간에 위임을 통해 연결합니다.



고객 웹 서비스는 주방 서버가 제공하는 인터페이스가 필요합니다. 고객 웹 서비스에 필요한 인터페이스를 추가하고, 주방 서버는 제공된 인터페이스를 추가하여 이 둘 간에 파트 어셈블리를 이용하여 연결합니다.

연결된 파트 어셈블리는 구성 요소에 따라 다르게 구현될 수 있음을 의미하지만, 이 연결된 파트는 동일한 부모 구성 요소를 필요로 합니다.

주방 서버는 필요한 인터페이스로 주방작업 큐를 추가하여 주방 웹 사이트의 제공된 인터페이스를 위임을 통해 연결하여 주방 서버의 작업을 완료합니다.



5.4. UML CLASS DIAGRAMS (클래스 다이어그램)

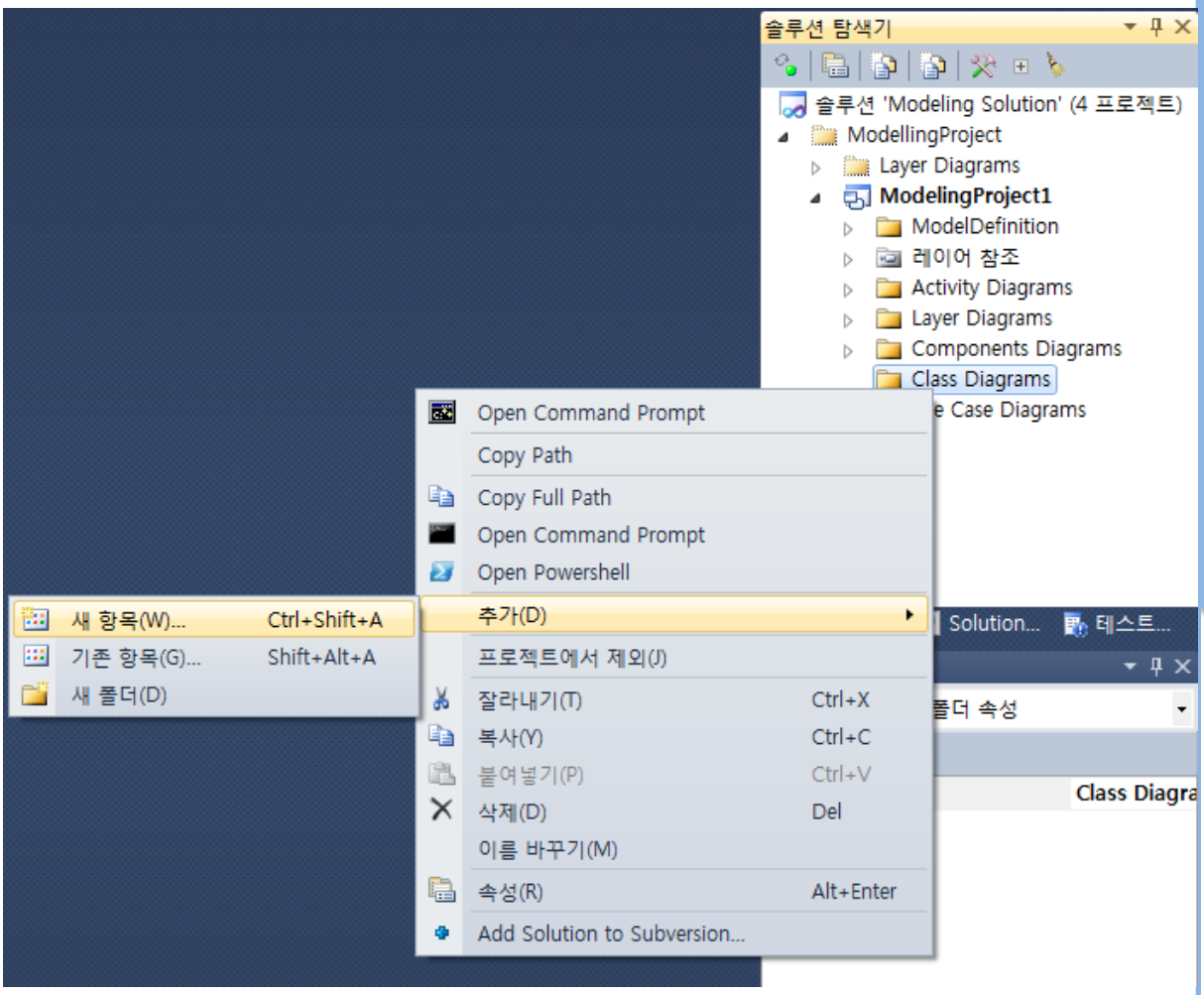
Class Diagrams 은 컴포넌트 또는 어플리케이션의 내부적으로 사용하는 개체 정보에 대한 구조를 기술합니다. 데이터베이스 테이블이나 XML, 또는 개체의 관계를 표현할 수 있습니다. Class Diagrams 은 데이터의 형식이나 관계를 구현 및 분리할 수 있습니다. Class Diagrams 은 주로 논리적인 측면에 중점을 두어 정의합니다.

Class Diagrams 을 이용하면 다음과 같은 이점이 있습니다.

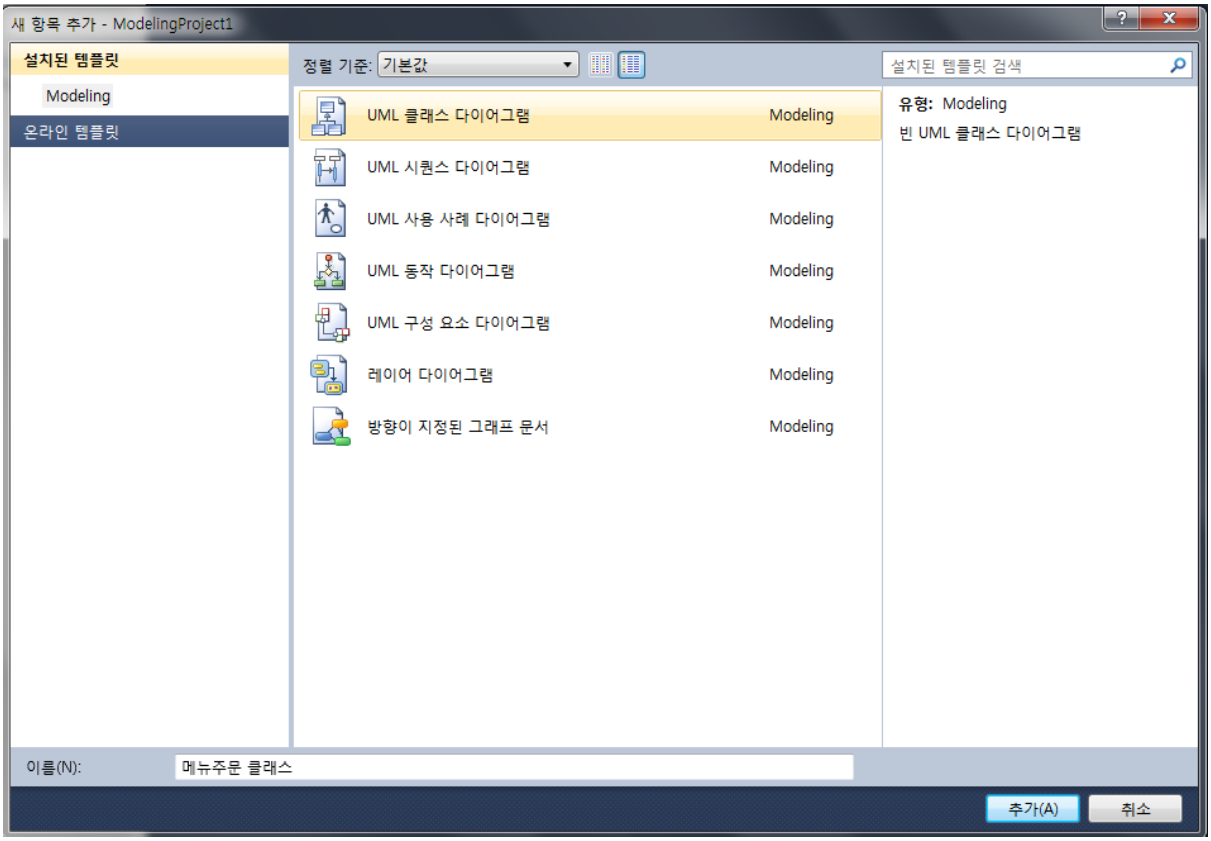
- 시스템 간에 전달되는 요소에 대한 구현과 독립적인 설명을 제공
- 어플리케이션과 사용자 간의 통신에 사용되는 용어를 명확하게 정의

5.4.1. CLASS DIAGRAMS 만들기

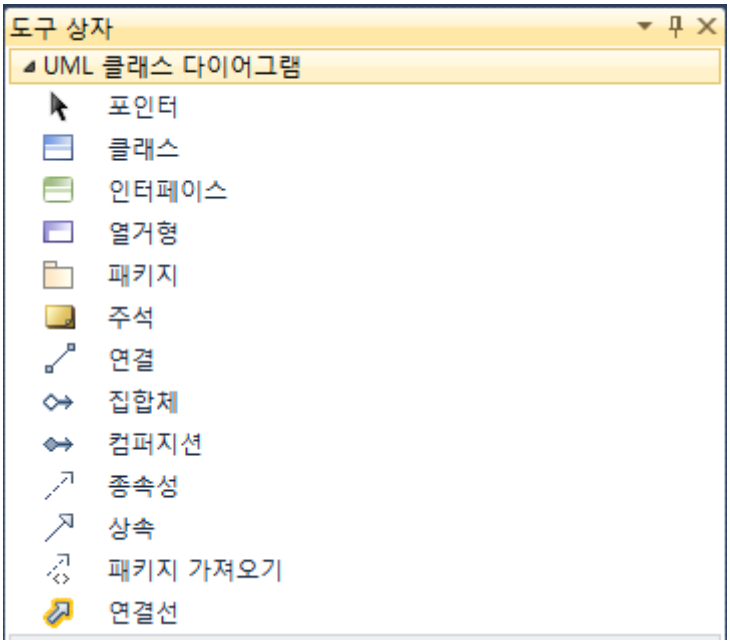
1. 모델링 프로젝트에서 마우스 오른쪽 버튼을 클릭한 후, 추가->새 항목을 선택합니다.



2. 새 항목 추가 대화 상자에서 "UML Class Diagrams"을 선택한 후 추가를 클릭합니다.

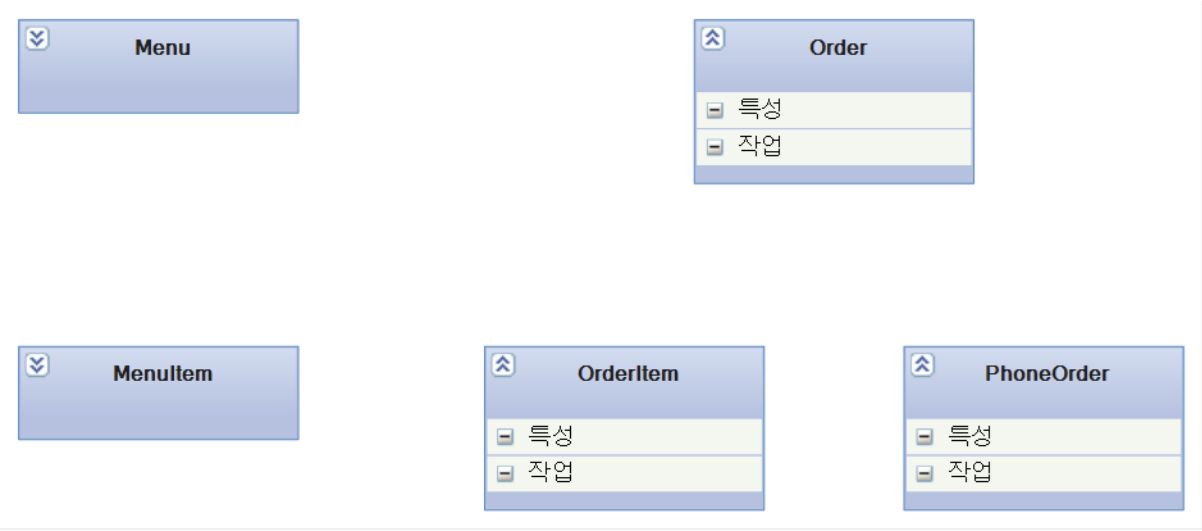


다음은 Class Diagrams 의 도구 상자에서 지원하는 요소입니다.

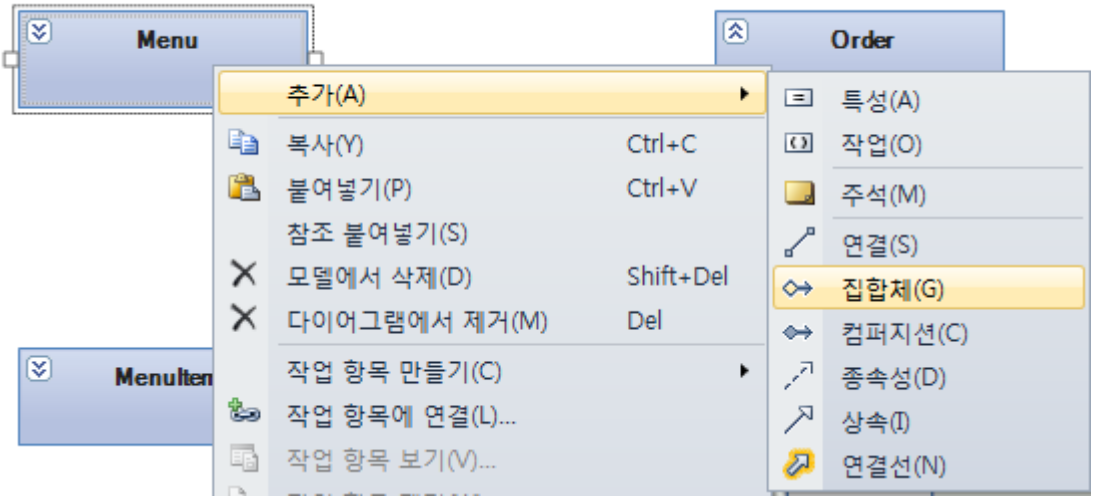


5.4.2. 메뉴 및 주문 CLASS DIAGRAMS

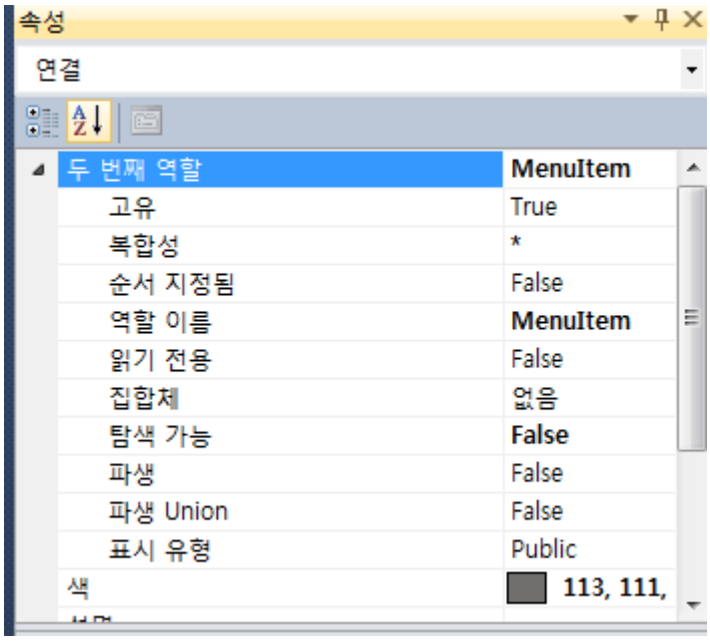
메뉴와 주문에 필요한 클래스를 도구 상자에서 끌어와 각 클래스를 만듭니다.



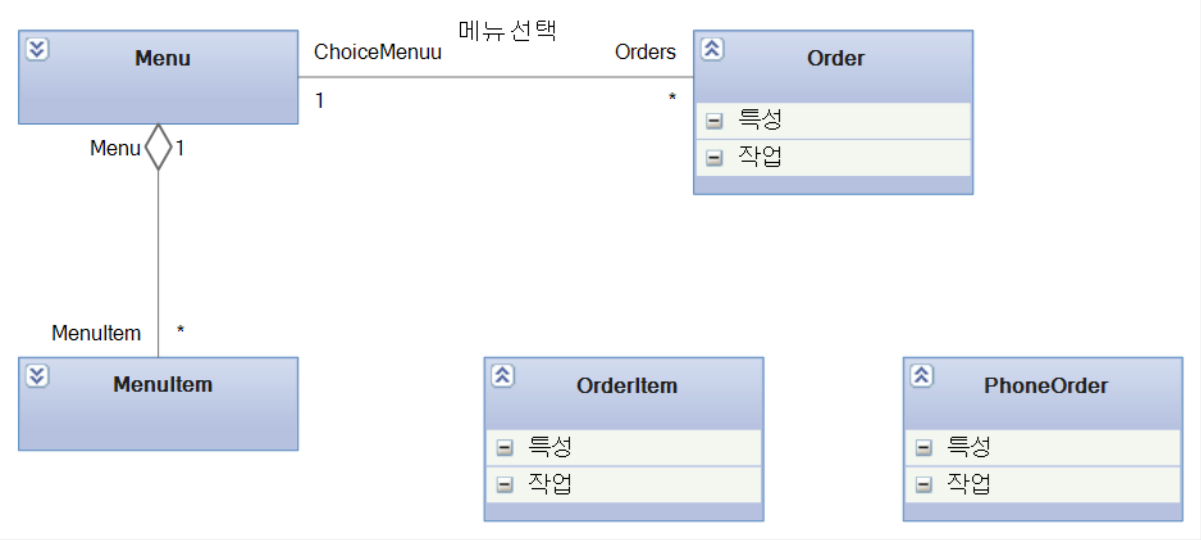
그리고 Menu 와 MenuItem 은 집합체로 구성하기 위해 Menu 를 선택하고 에서 추가->집합체를 선택합니다.



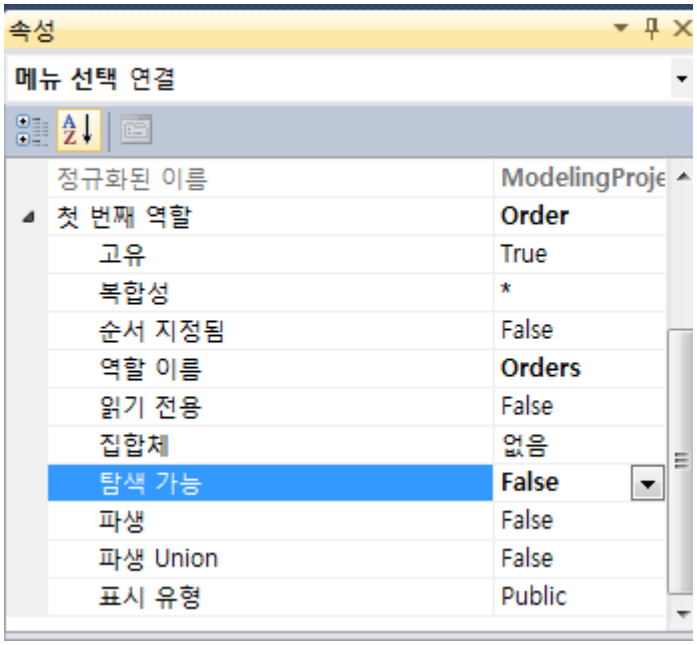
집합체의 MenuItem 을 Menu 의 1:* 관계를 만들기 위해 MenuItem 을 선택한 후 속성 창에서 "복합성"을 * 로 선택합니다. 그리고 탐색 가능을 False 로 설정하여 단방향 탐색이 가능하도록 합니다.



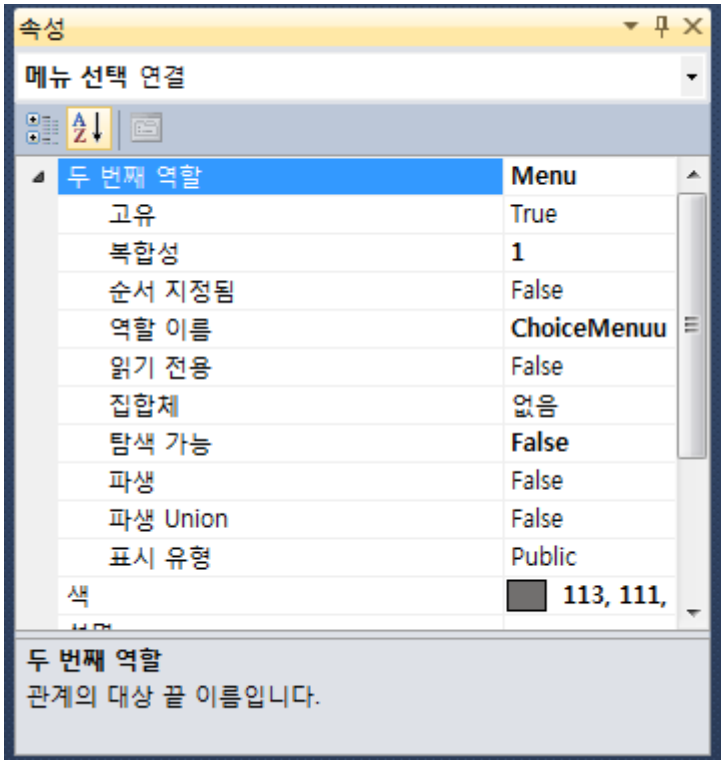
Menu 와 Order 클래스간의 연결 관계를 만듭니다.



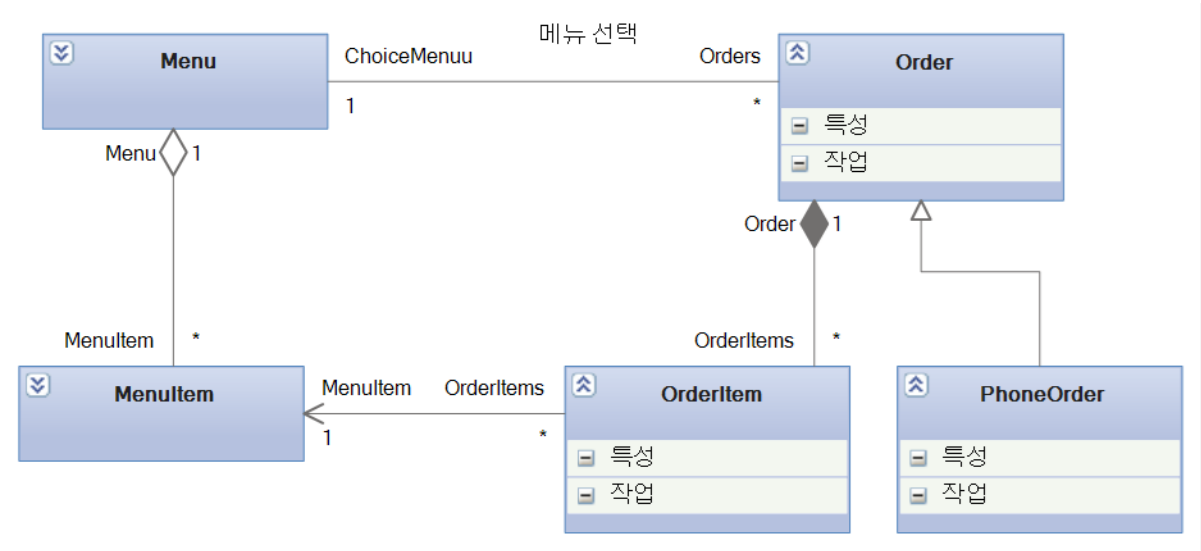
Order 클래스는 연결 요소의 속성 창에서 복합성을 * 로 선택하고 탐색 가능을 False 로 설정합니다.



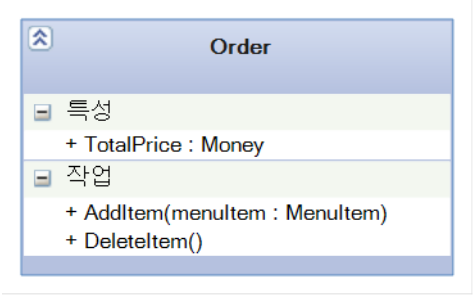
Menu 클래스도 연결 요소의 속성 창에서 탐색 가능을 False 로 설정합니다.



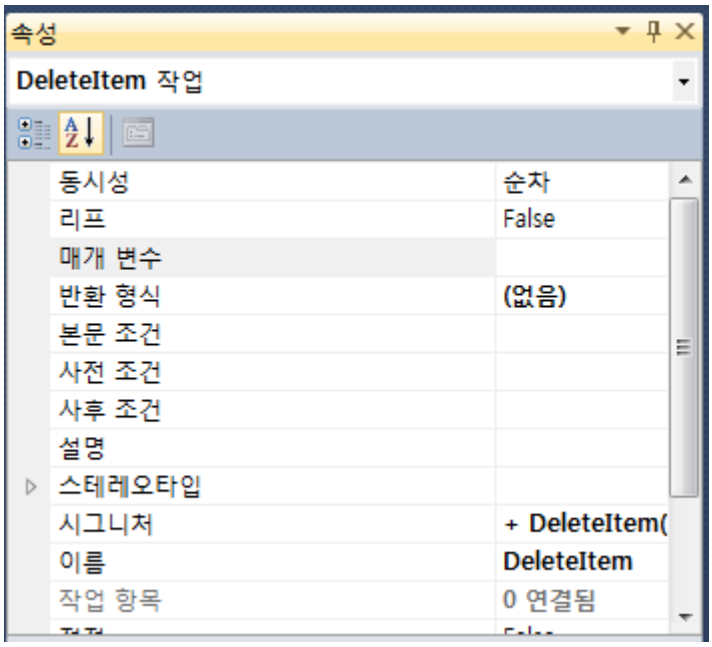
다음은 위와 같은 방법으로 관계를 완성한 Class Diagrams 입니다.



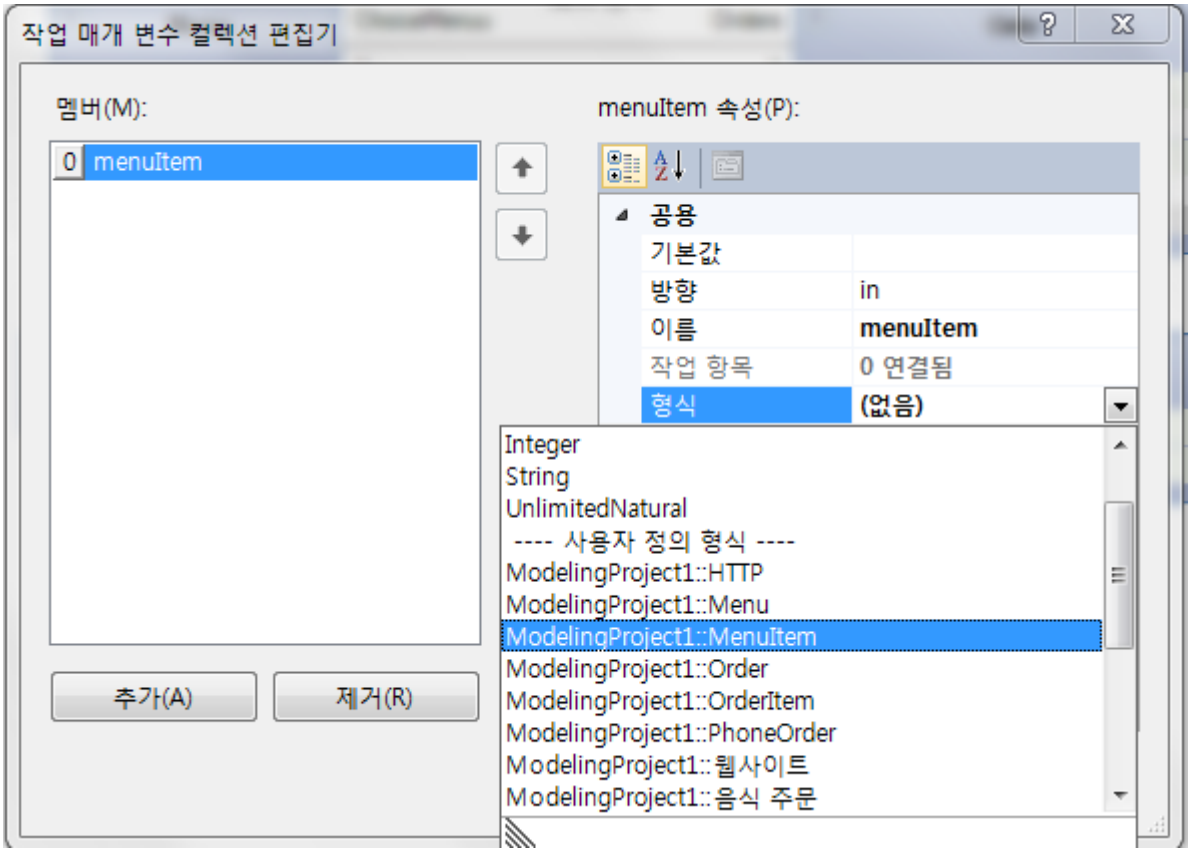
이제 각 클래스에 속성과 메서드 등의 작업을 추가합니다. 메서드(작업)에 아래와 같이 임의로 정의한 매개 변수를 추가를 할 수 있습니다.



Order 클래스의 DeleteItem 을 선택한 후 속성 창에서 매개 변수를 클릭하여 대화 상자를 엽니다.

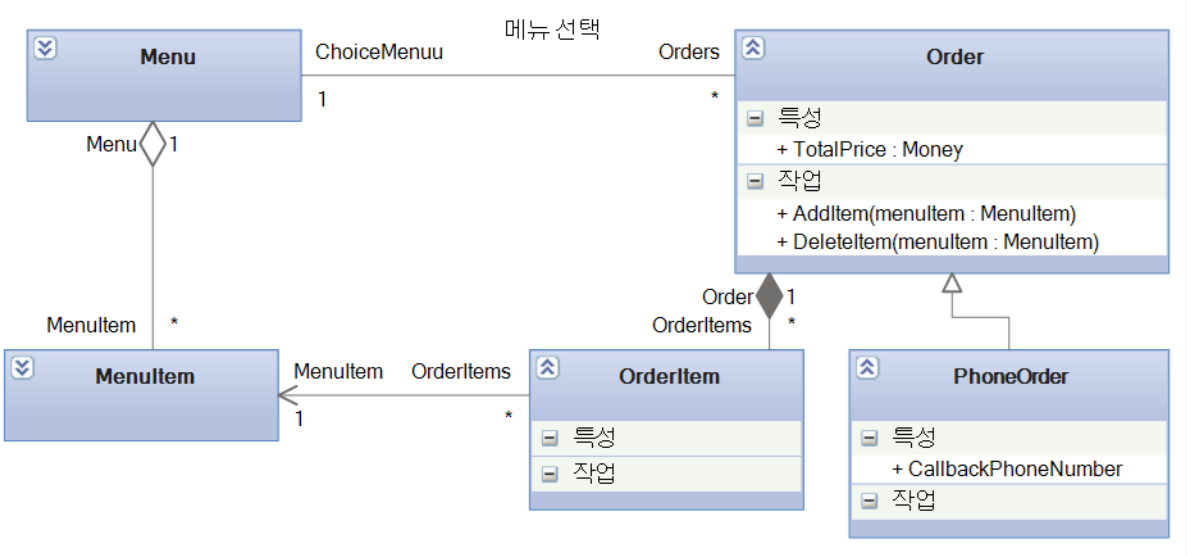


작업 매개 변수 컬렉션 편집기 대화 상자에서 매개 변수의 이름을 입력한 후 형식에서 MenuItem 형식을 선택하여 확인을 클릭합니다.



그럼 다음과 같이 MenuItem 형식의 매개 변수를 갖는 메서드를 추가할 수 있습니다.

다음은 완성된 메뉴 및 주문 Class Diagrams 입니다.



5.5. UML SEQUENCE DIAGRAMS (시퀀스 다이어그램)

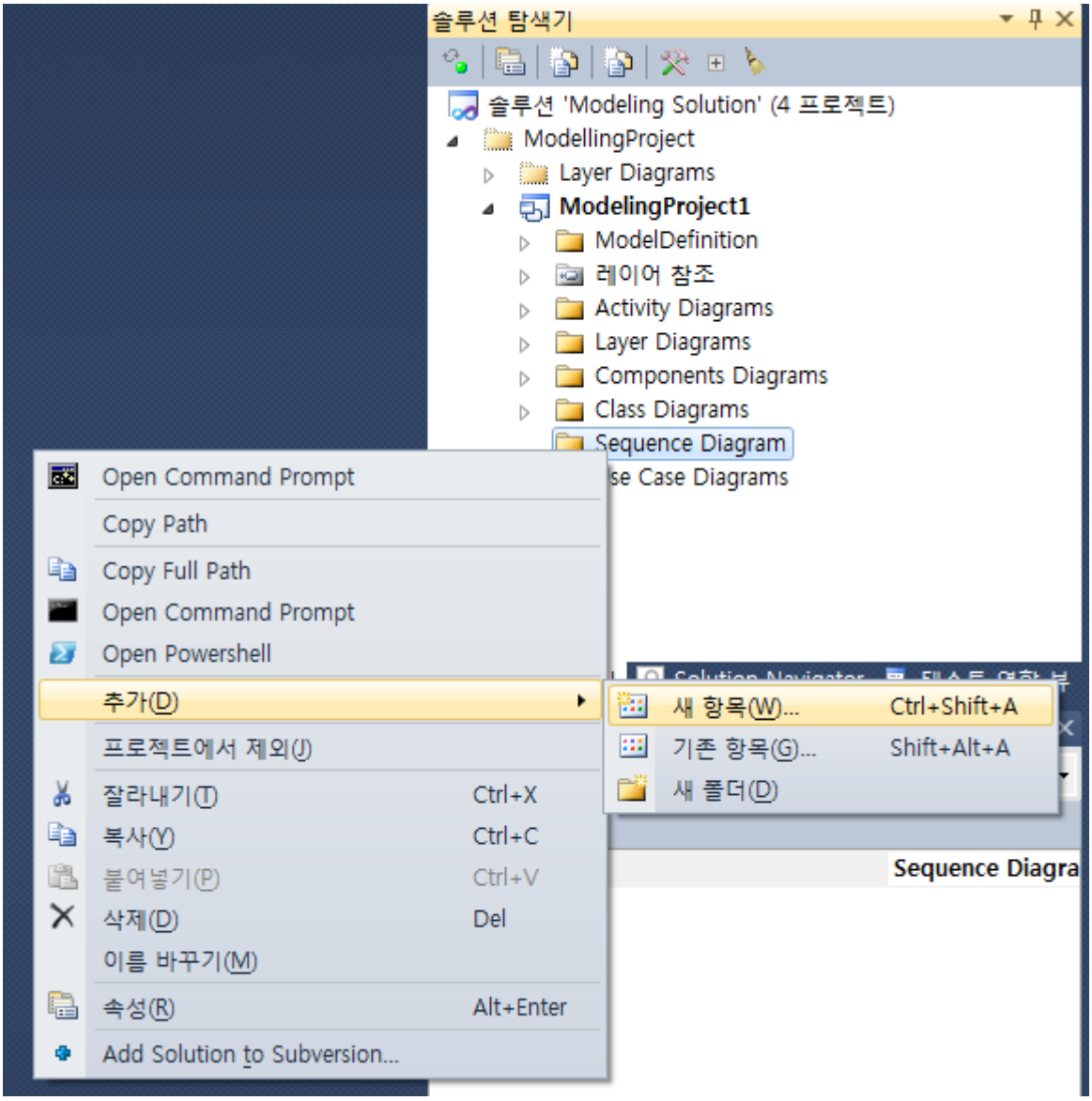
Sequence Diagrams 은 클래스나 구성 요소, 인스턴스, 시스템 간의 메시지 동작의 순서를 나타내는 다이어그램입니다. 일반적으로 Sequence Diagrams 은 위에서 아래로 시간적인 흐름을 갖고 있습니다. Sequence Diagrams 의 수평선의 상호 작용 참가자 간의 이벤트를 나타냅니다.

Sequence Diagrams 을 사용하면 다음과 같은 이점이 있습니다.

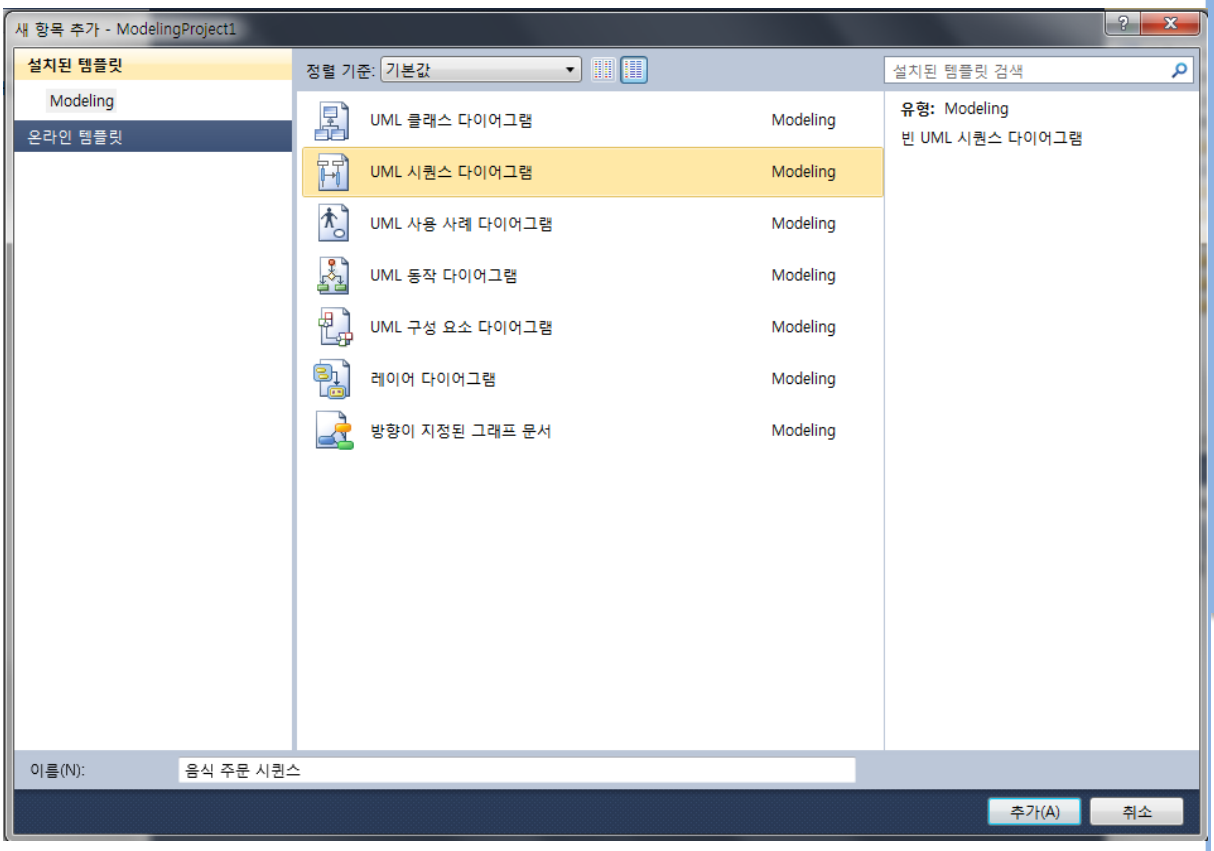
- 구성 요소간의 작업이 진행되는 흐름을 쉽게 확인할 수 있습니다.
- 어플리케이션에서 상호 작용을 어렵게 하는 패턴을 식별할 수 있습니다.

5.5.1. SEQUENCE DIAGRAMS 만들기

1. 모델링 프로젝트의 항목에서 마우스 오른쪽 버튼을 클릭하여 추가->새 항목을 선택합니다.



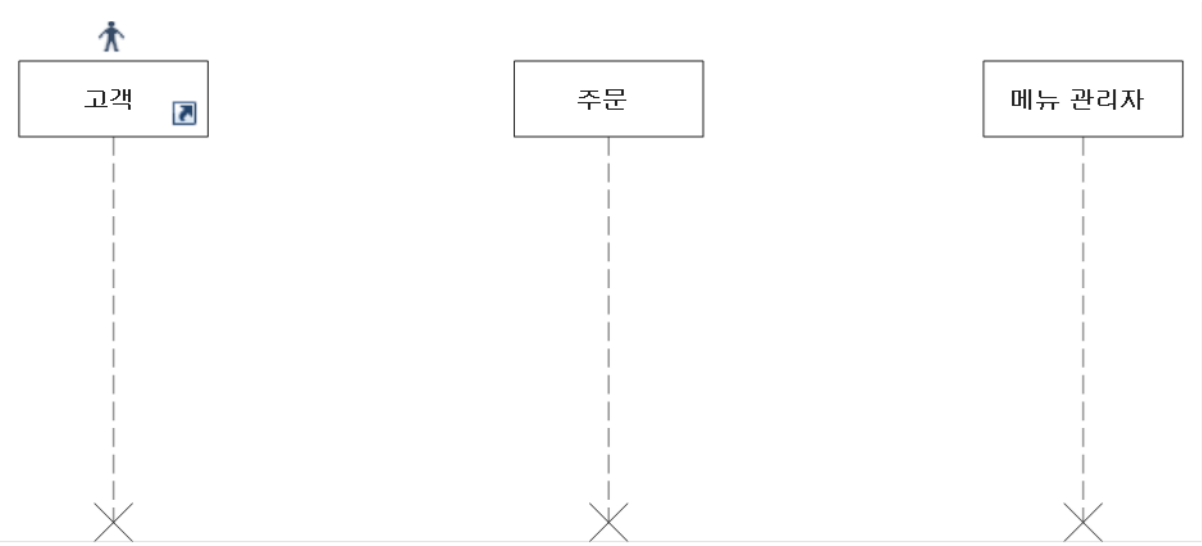
2. 새 항목 추가 대화 상자에서 "UML Sequence Diagrams"을 선택하고 추가를 클릭합니다.



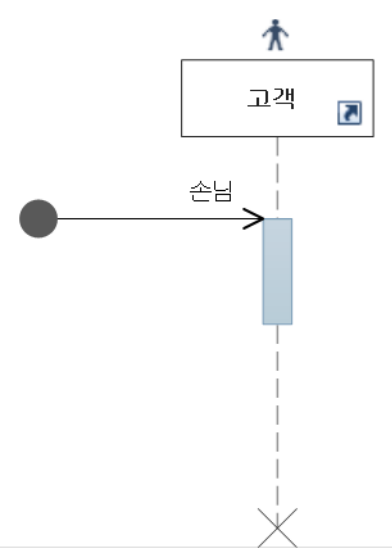
다음은 Sequence Diagrams 이 제공하는 대화 상자의 요소입니다.

5.5.2. 음식 제공 서비스 SEQUENCE DIAGRAMS

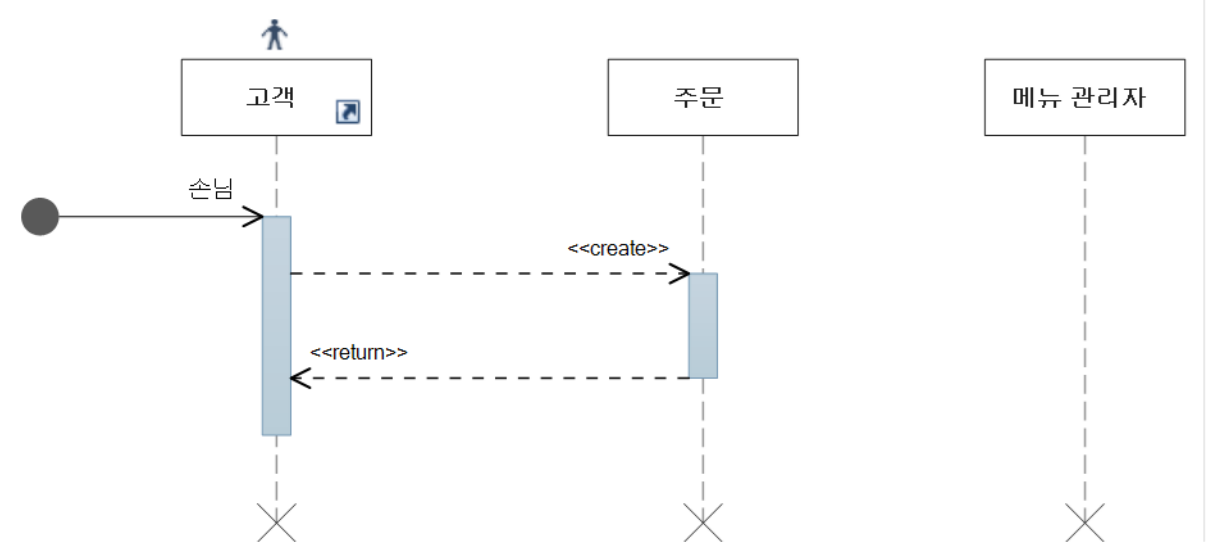
우선 각 메시지 이벤트를 송수신하는 주체인 참가자를 추가해야 합니다. 도구 상자의 수평선을 이용하여 세 개의 참가자를 추가합니다.



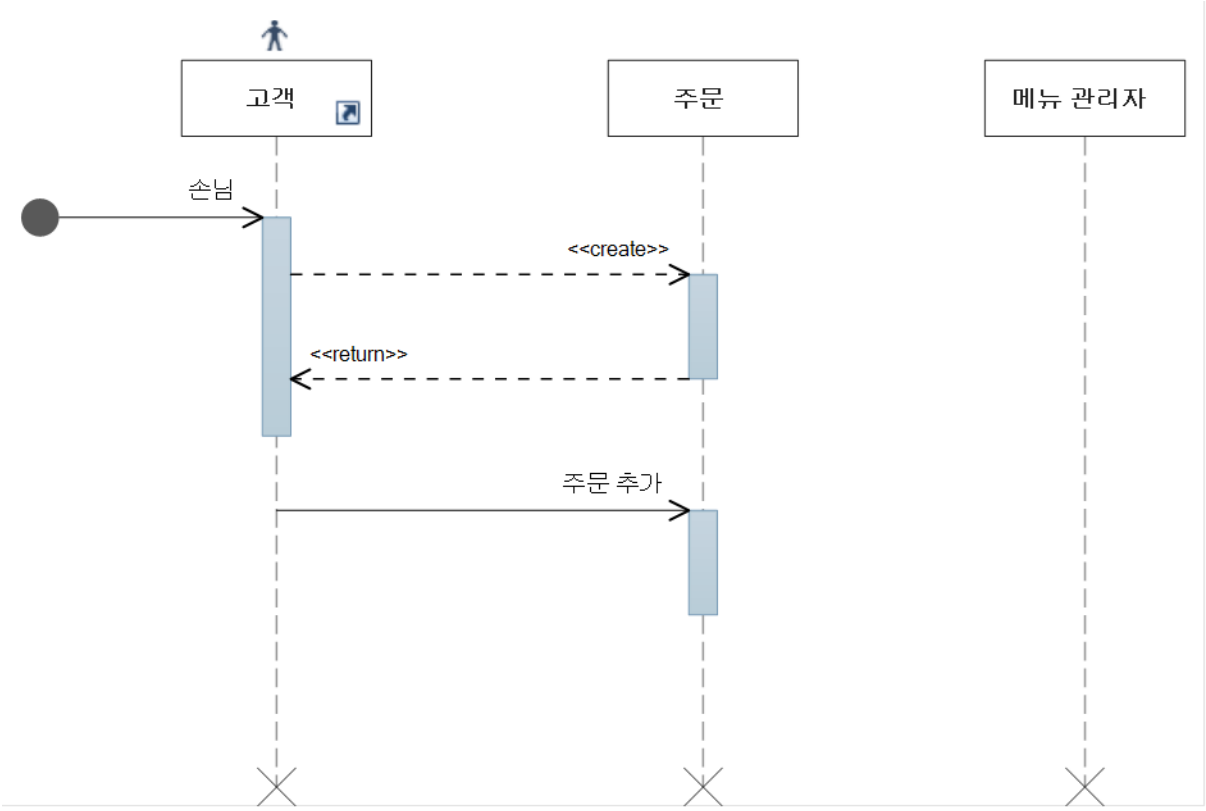
비동기 요소를 이용하여 알 수 없는 특정 메시지를 고객 타임라인에 추가 합니다.



손님이 주문을 하기 위해 주문 타임라인에 주문에 대한 메시지를 보내야 합니다.

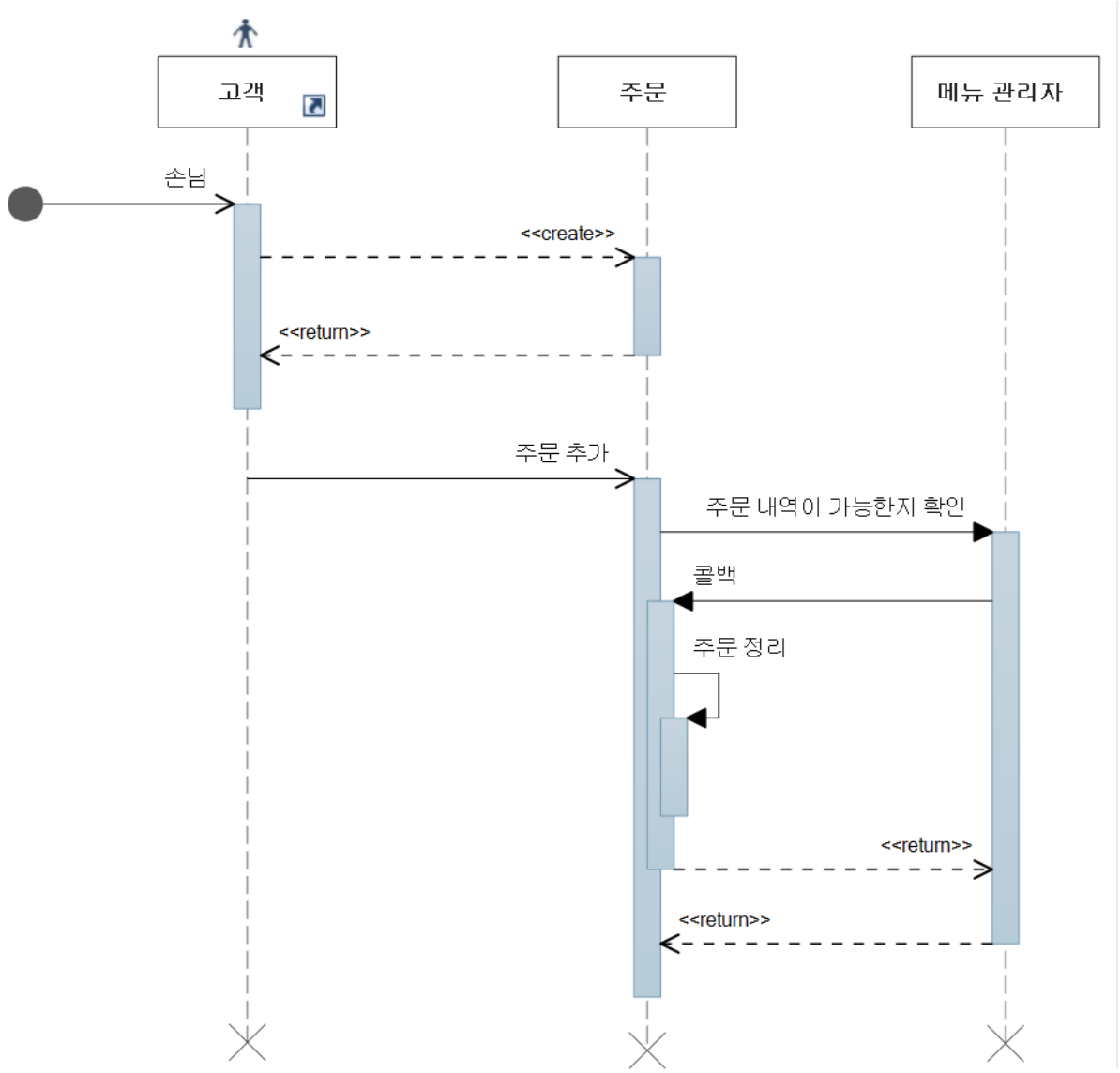


비동기 메시지를 통해 주문 타임라인에 주문한 내용을 추가하는 메시지를 보냅니다.

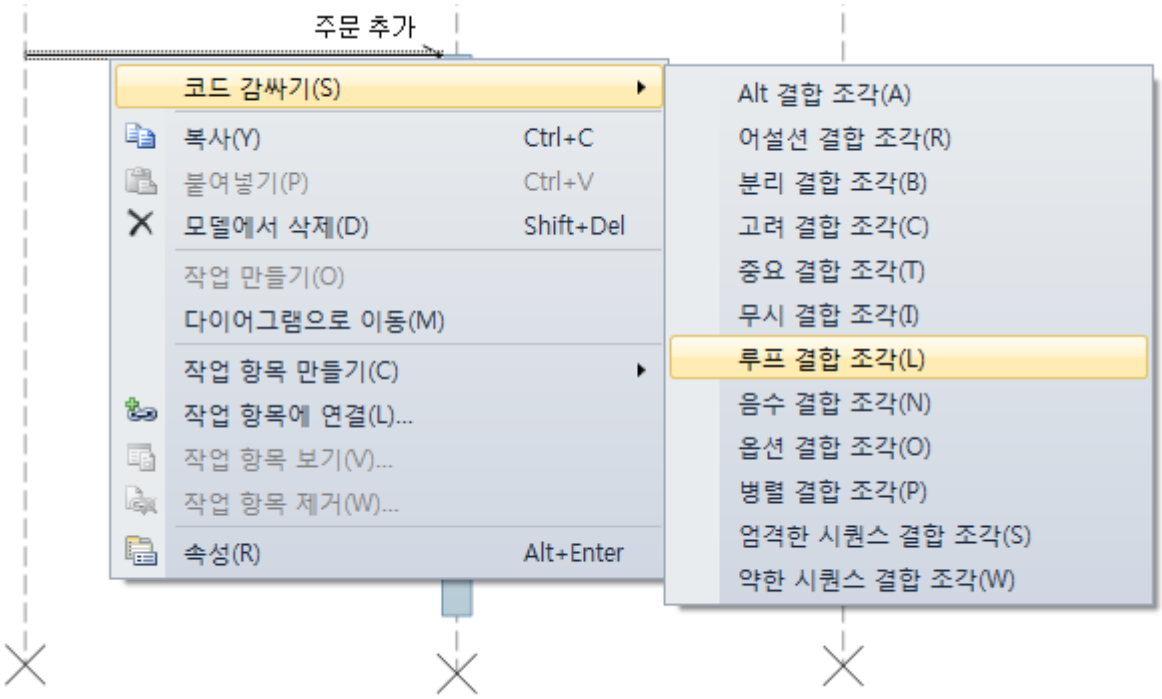


주문을 받으면 메뉴 관리자에게 주문 내역이 요리 가능한지 확인을 하는 동기 메시지를 보내고, 메뉴 관리자는 메시지의 응답을 기다리는 호출자에게 어떤 메시지를 콜백합니다. 그럼 주문

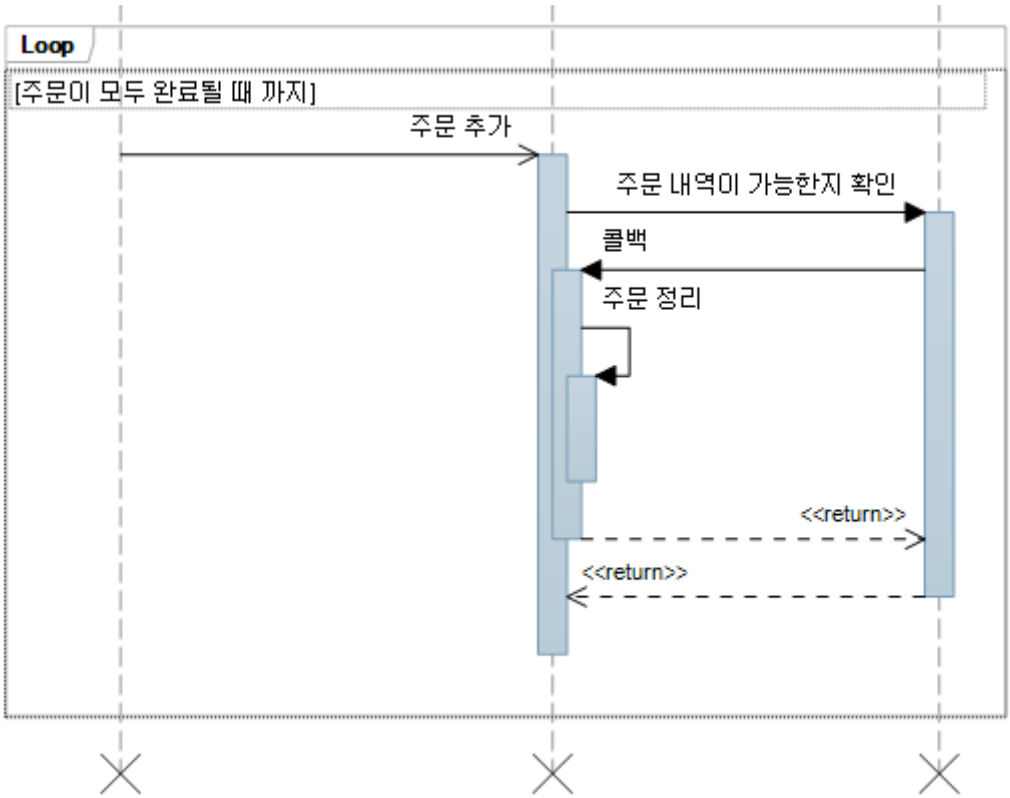
타임라인은 그 주문에 대해 최종적으로 정리 작업을 진행하고 주문을 하기 위한 모든 작업을 완료 합니다.

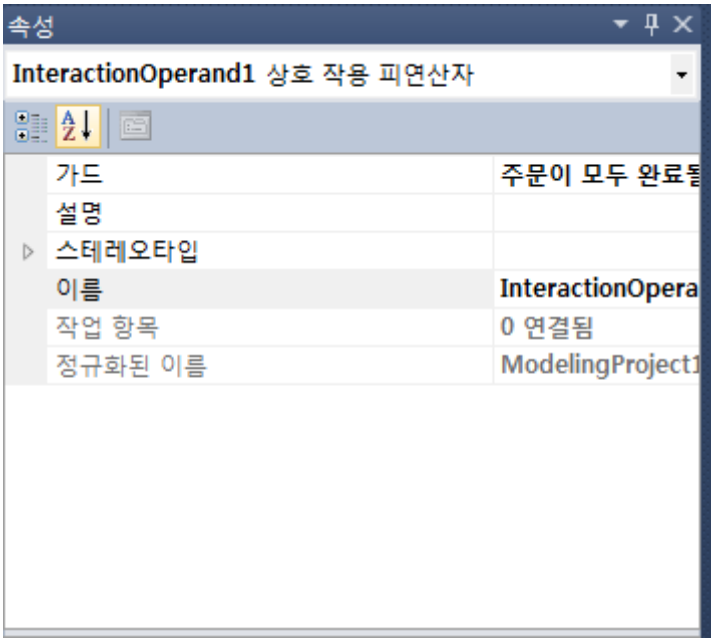


여러 개의 주문에 대해 처리를 할 경우 주문 추가 비동기 메시지부터 루프 코드를 감싸도록 합니다.

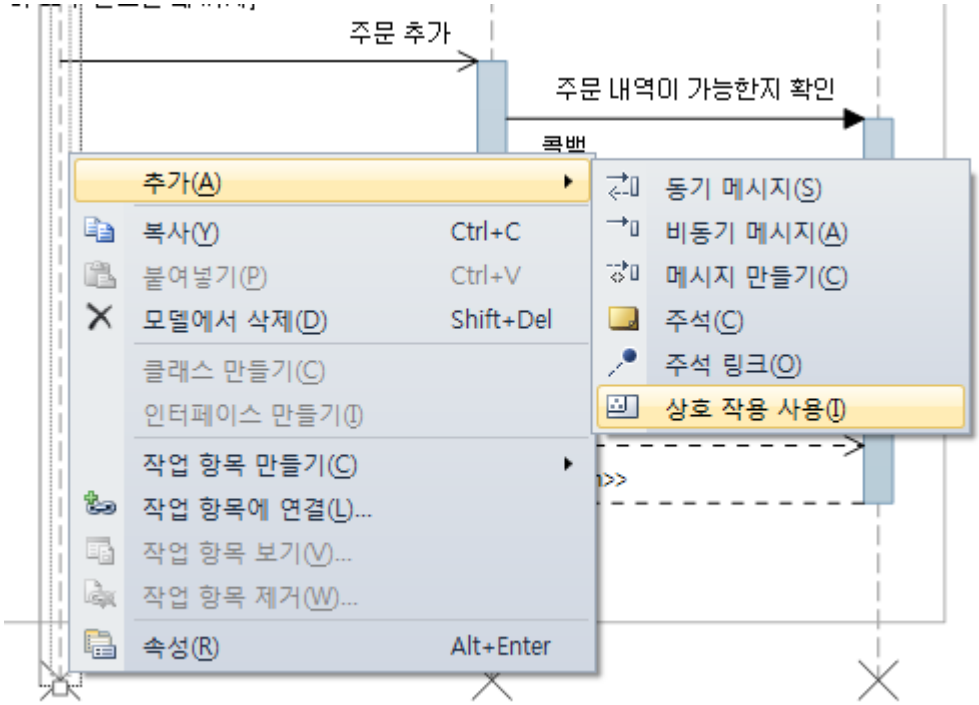


그리고 루프 코드의 가드를 이용하여 루프가 벗어나게 되는 조건을 기술해 줍니다.



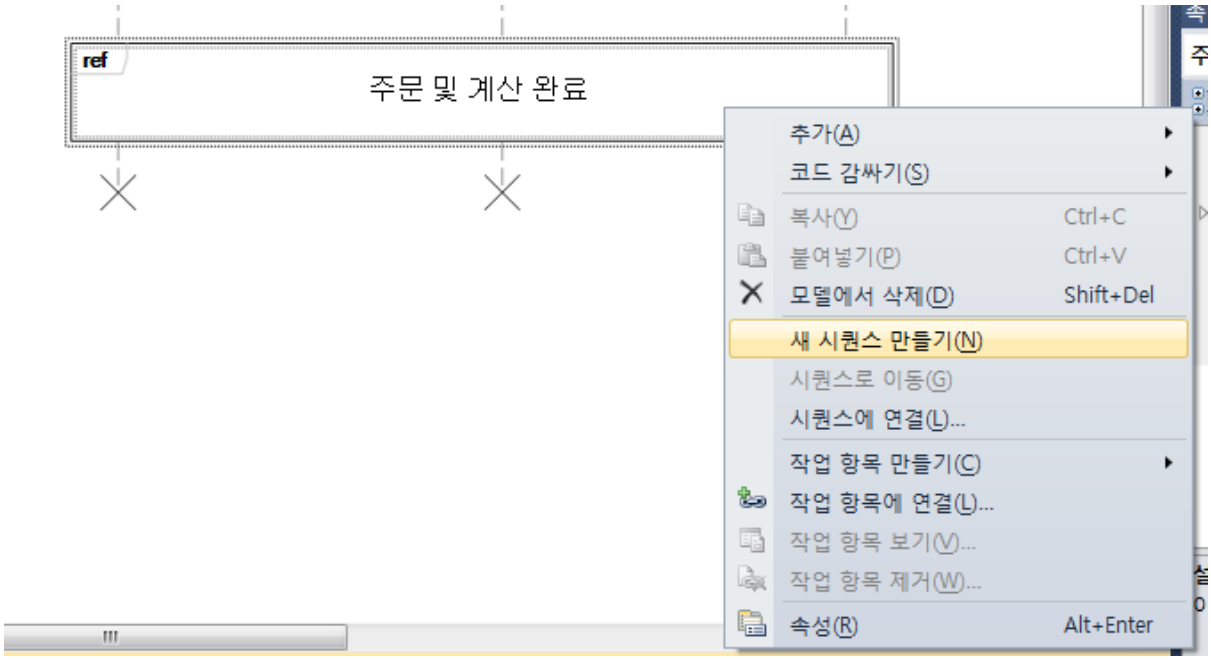


주문에 대한 루프 작업이 완료되었다면, 루프가 완료되는 시점의 상호 작용 요소를 추가합니다.

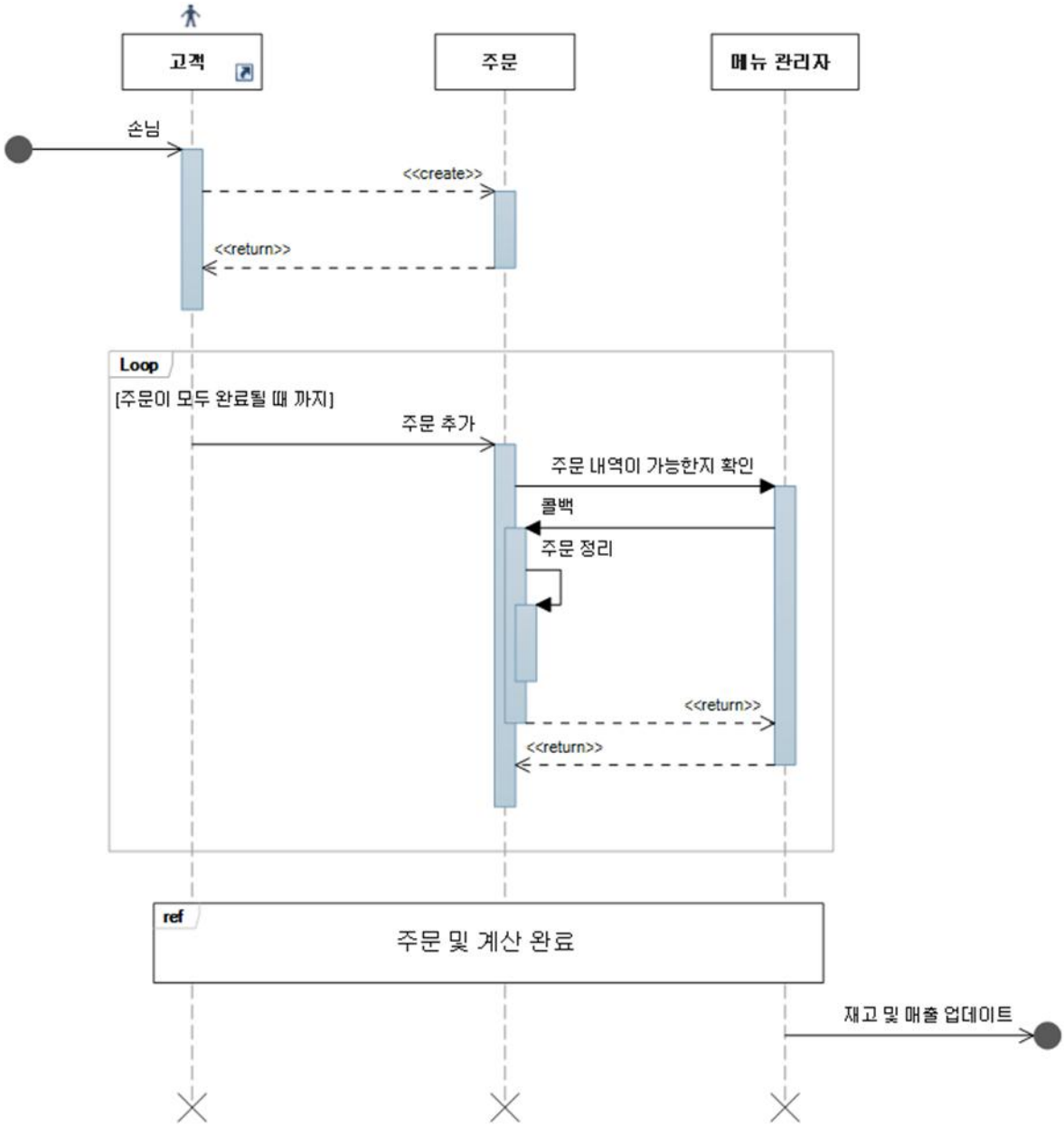


상호 작용 요소는 다른 다이어그램과 연결할 수 있는 요소입니다. 현재 Sequence Diagrams 과 연결된 새로운 시퀀스를 만들려면 "새 시퀀스 만들기"를 선택합니다. 새로운 시퀀스 만들기를 선택하면 새로운 Sequence Diagrams 파일이 생성이 됩니다.

또는 시퀀스에 연결을 선택하면 이미 존재하는 Sequence Diagrams 과 연결을 할 수 있습니다.



마지막으로 비동기 메시지를 이용하여 재고 및 매출 업데이트 메시지를 보내고, 시퀀스는 종료됩니다.



5.6. LAYER DIAGRAMS (레이어 다이어그램)

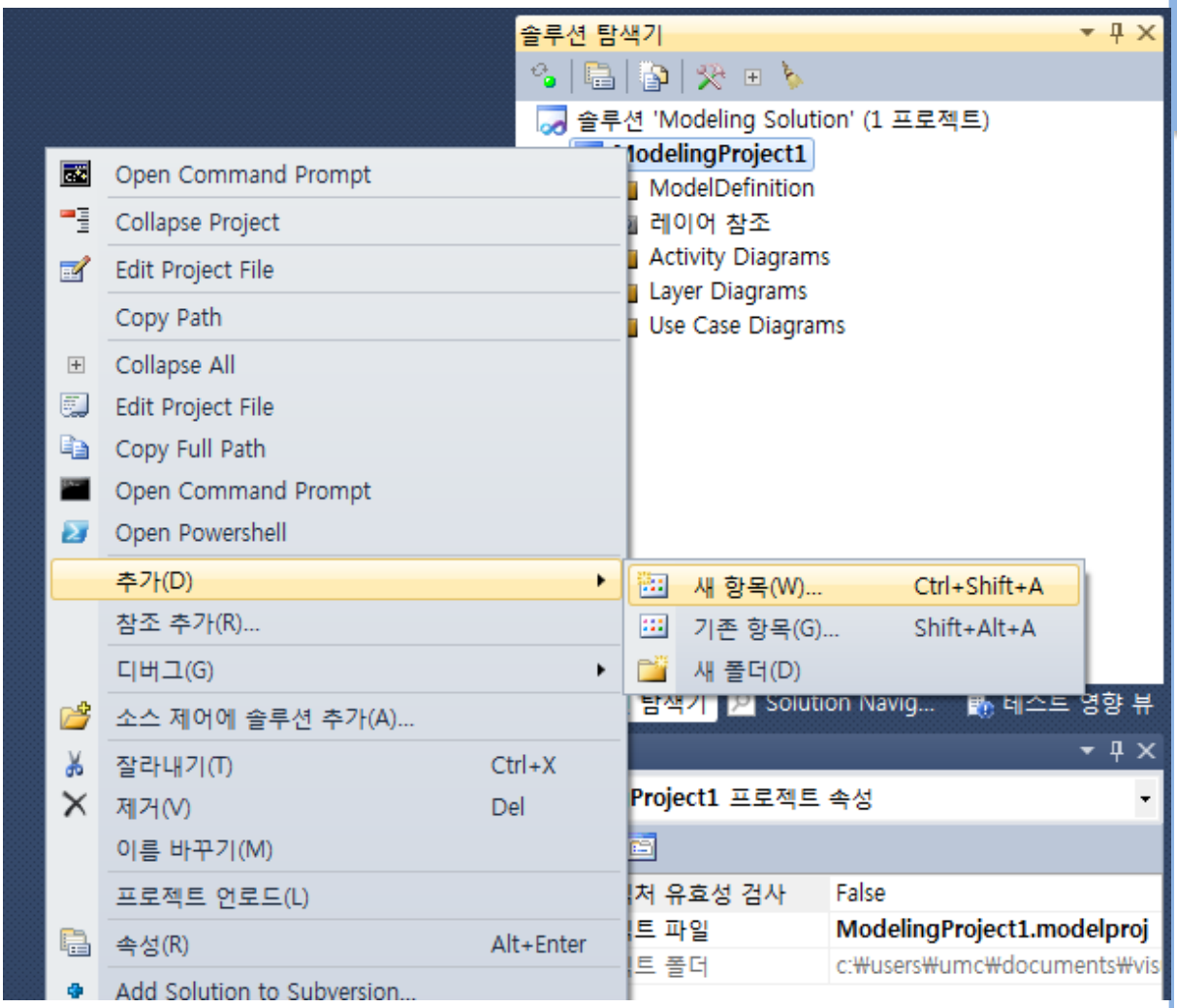
Layer Diagrams 은 시스템의 논리적인 아키텍처를 시각화할 수 있습니다. Layer Diagrams 은 물리적인 아티펙트를 논리적인 추상 그룹으로 구성합니다.

Layer Diagrams 은 논리적인 아키텍처를 시각화하는 것 이외에, 실제 Visual Studio 2010 에서는 동작하는 코드와 논리적 아키텍처간의 충돌을 검사하기도 합니다. 시스템을 리팩토링하거나

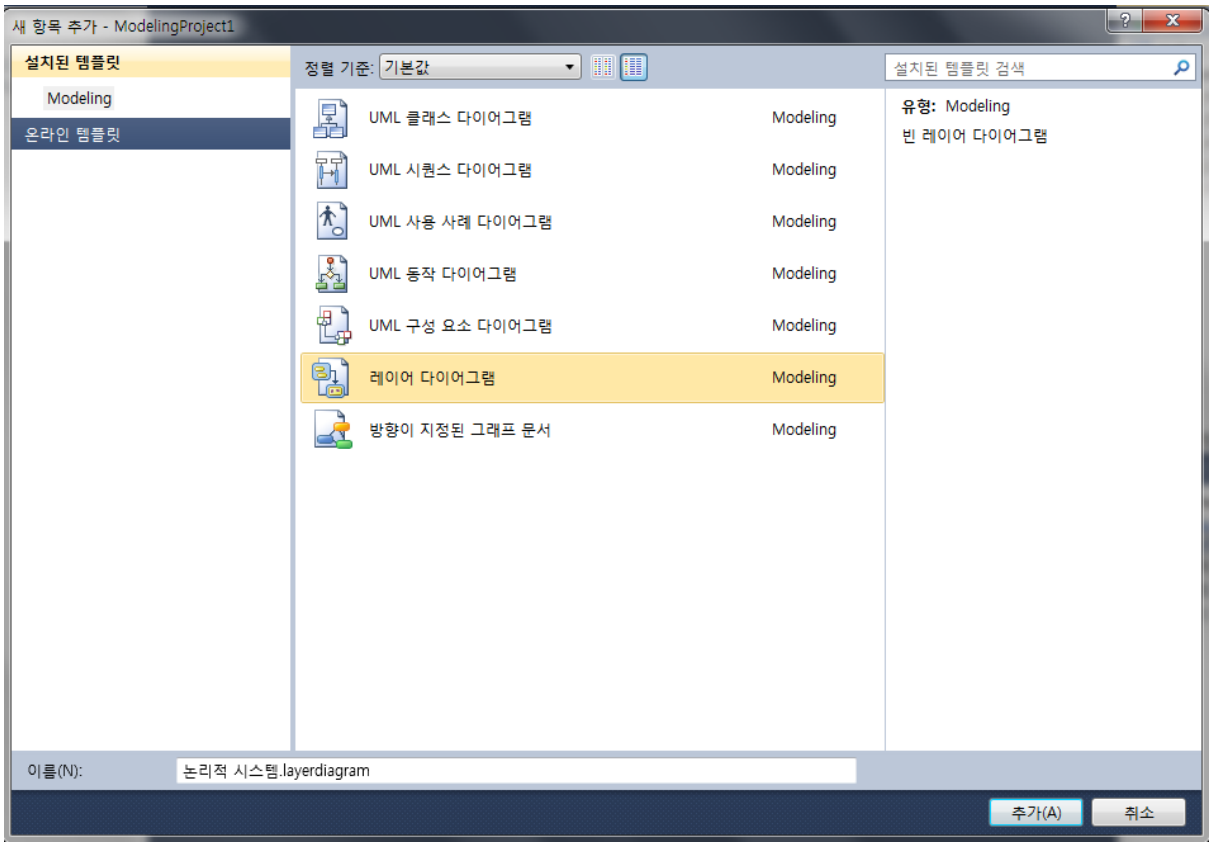
업데이트 시에 변경에 미치는 영향을 분석하며 체크인 또는 빌드 작업에 충돌에 대한 유효성을 검사하도록 하여 기존의 계획을 강화할 수 있는 방법으로 사용할 수 있습니다.

5.6.1. LAYER DIAGRAMS 만들기

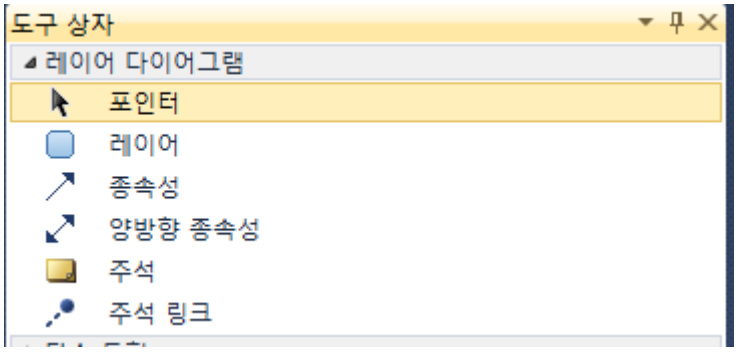
1. 솔루션 탐색기의 모델링 프로젝트에서 마우스 오른쪽 버튼을 클릭하여 추가->새 항목을 선택합니다.



2. 새 항목 추가 대화상자에서 Layer Diagrams 을 선택하고 추가를 클릭합니다.

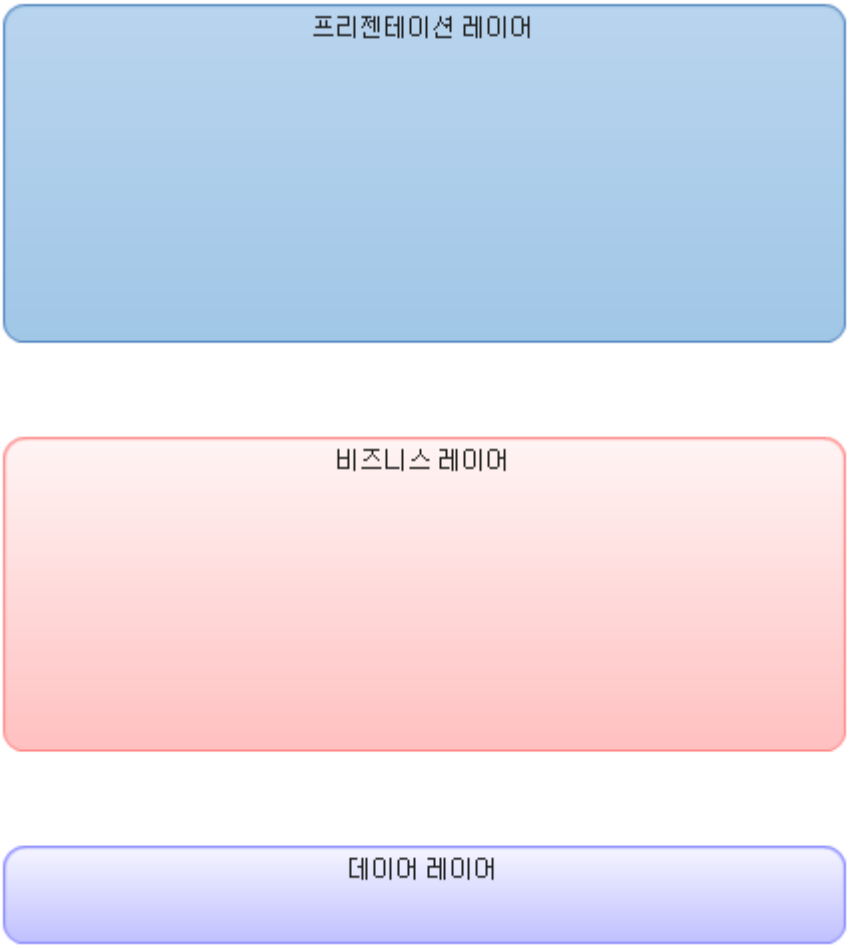


Layer Diagrams 은 도구 상자에 다음과 같은 요소를 제공합니다.

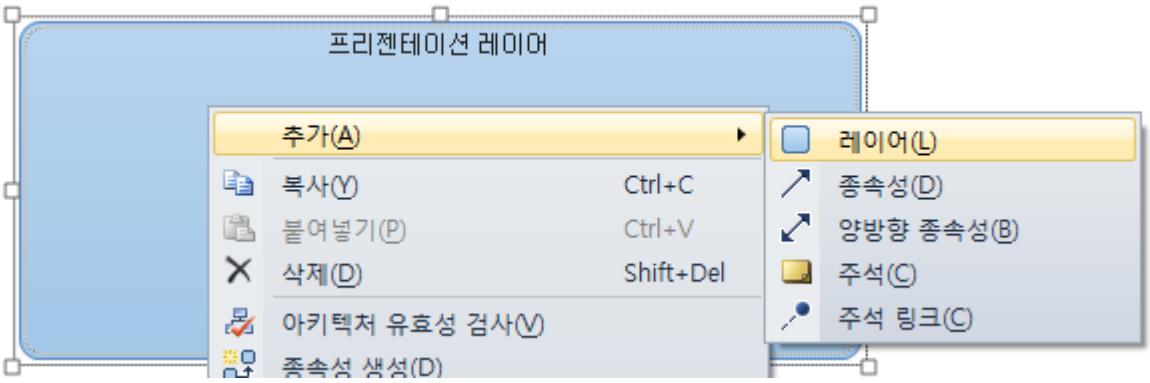


5.6.2. LAYER DIAGRAMS

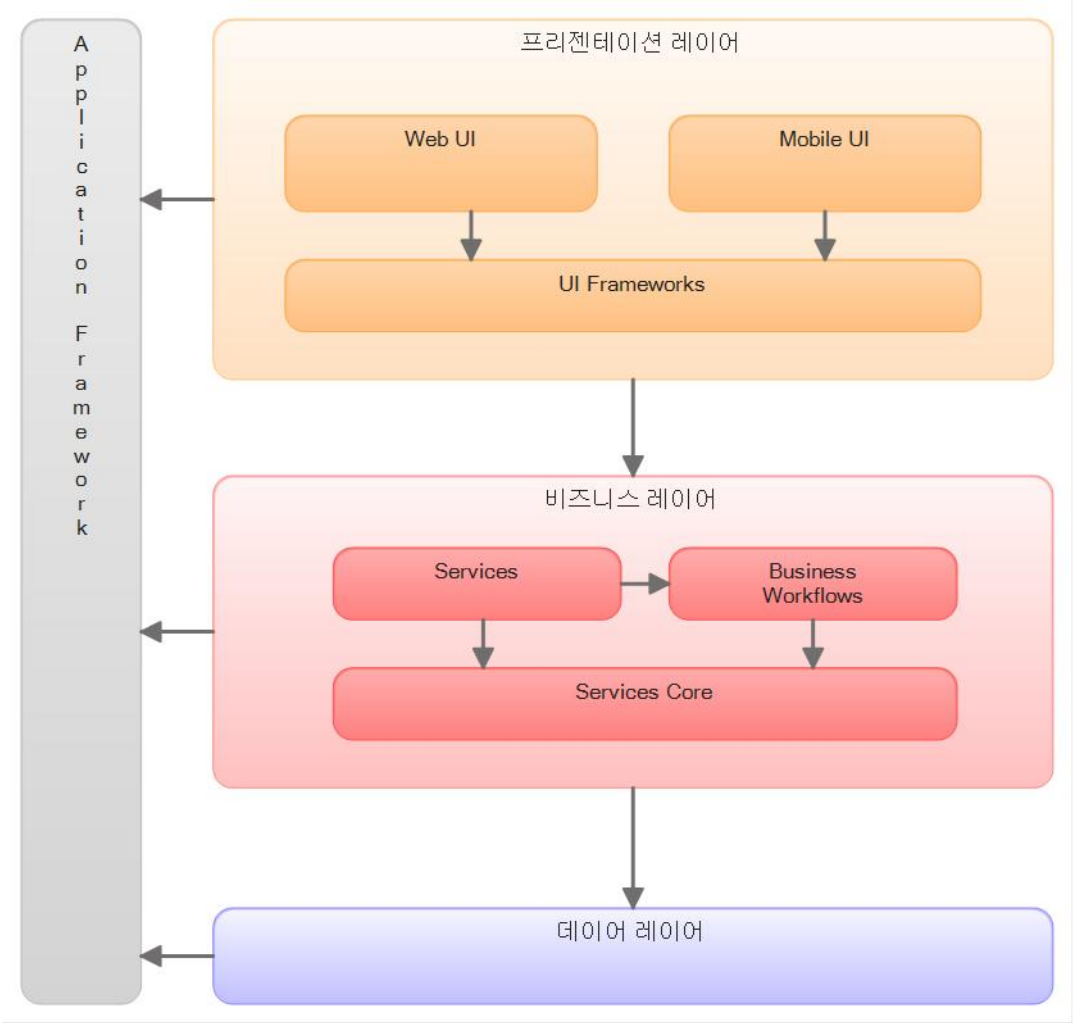
Layer Diagrams 으로 논리적인 3-tier 어플리케이션을 다음과 같이 구성할 수 있습니다.



각 레이어에 종속되는 레이어를 추가 하기 위해서 레이어를 선택하여 마우스 오른쪽 버튼을 클릭한 후 추가->레이어를 선택하여 종속된 레이어를 추가할 수 있습니다.



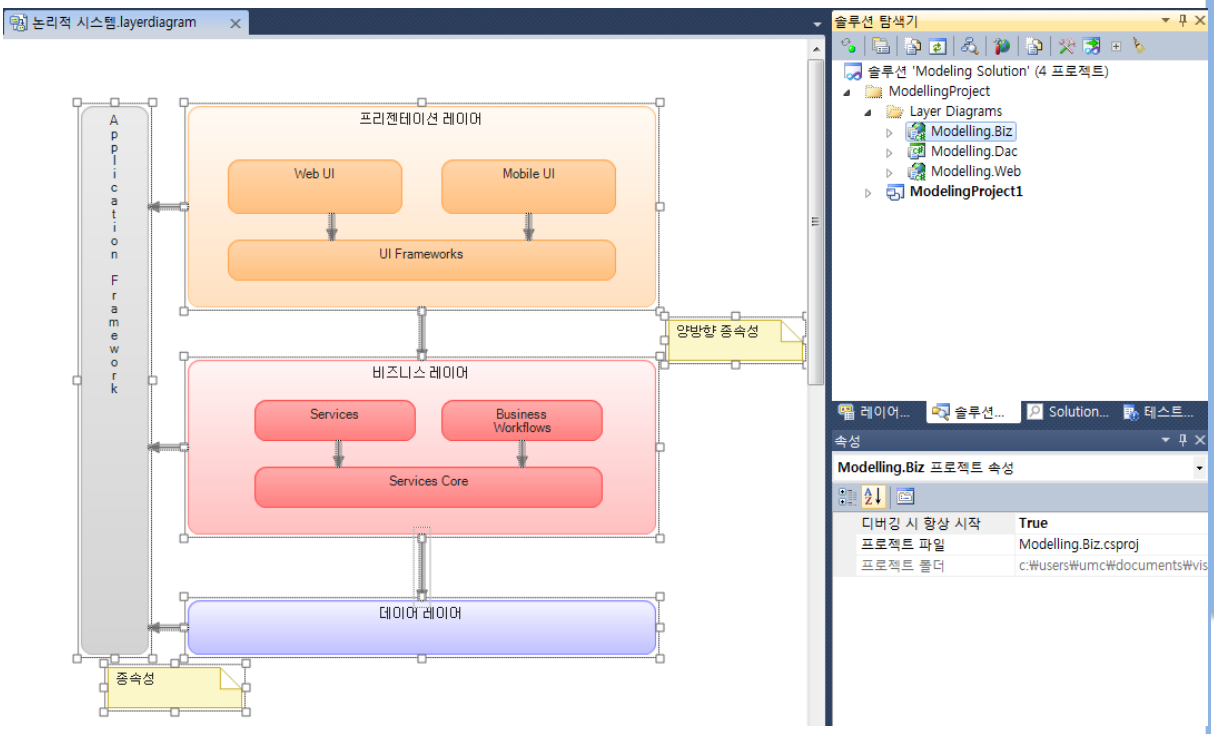
다음은 완성된 Layer Diagrams 의 예입니다.



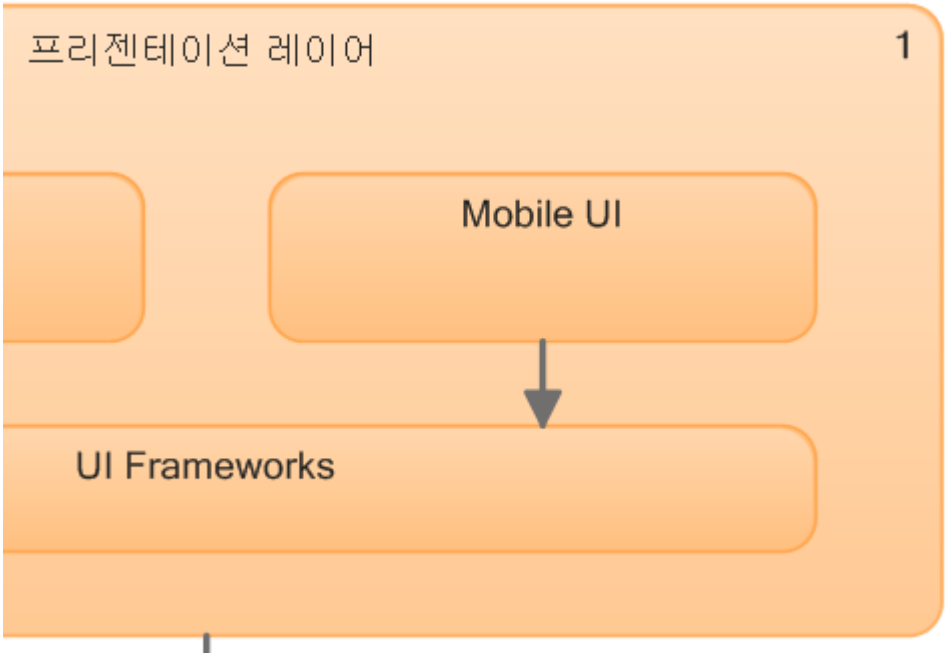
5.6.3. 코드와 LAYER DIAGRAMS 연동

실제 물리적인 코드와 Layer Diagrams 간의 연관 관계를 만들어 아키텍처를 분석하고 유효성을 검사할 수 있습니다.

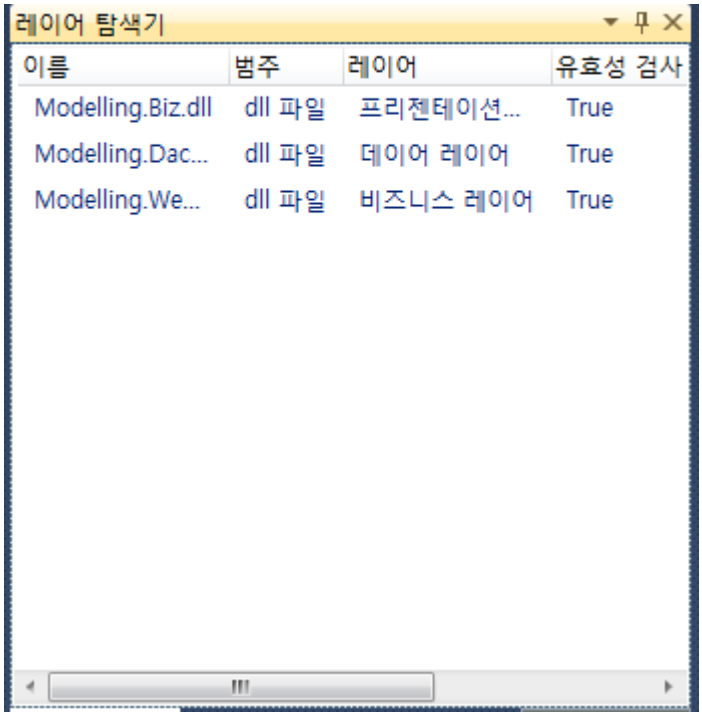
1. 솔루션 안에 간단한 논리적인 3 tier 를 구현한 프로젝트를 각 레이어에 맞게 드래그&드랍 합니다.



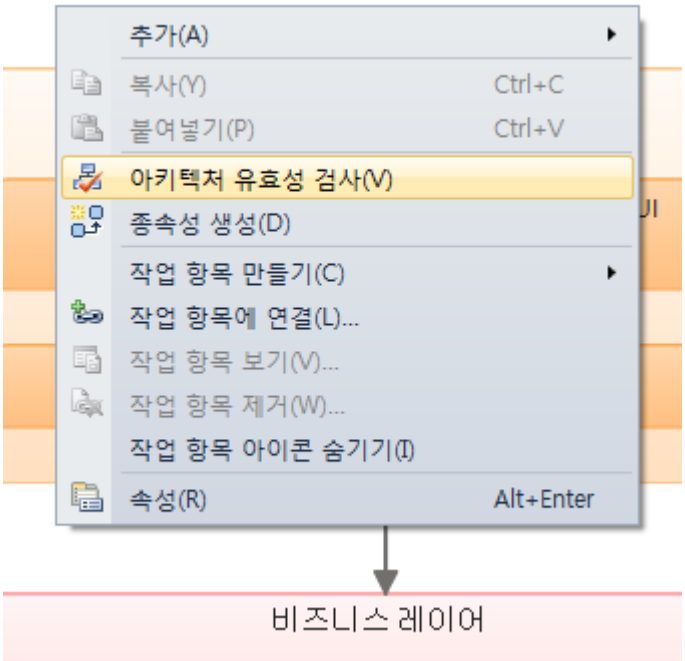
2. 각 레이어에 추가된 프로젝트의 개수가 표시되는지 확인합니다.



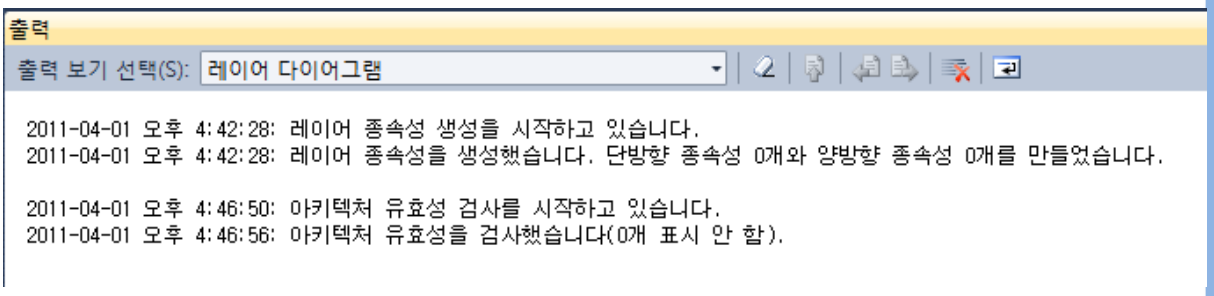
3. 레이어 탐색기에 추가된 프로젝트가 존재하는 것을 확인할 수 있습니다.



4. 레이어 디자이너에서 마우스 오른쪽 버튼을 클릭하여 "아키텍처 유효성 검사"를 클릭합니다.

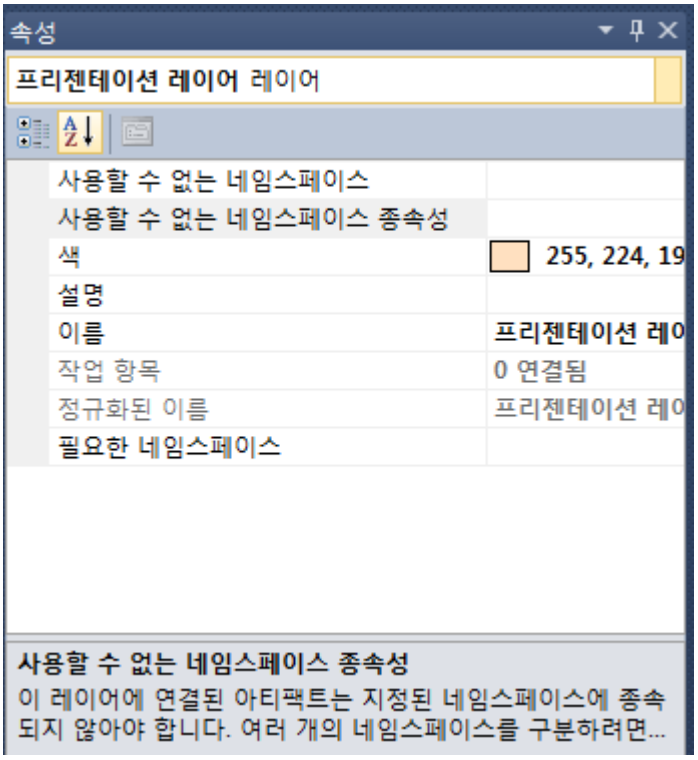


5. 출력 창에 아키텍처 유효성 검사 결과가 표시됩니다.

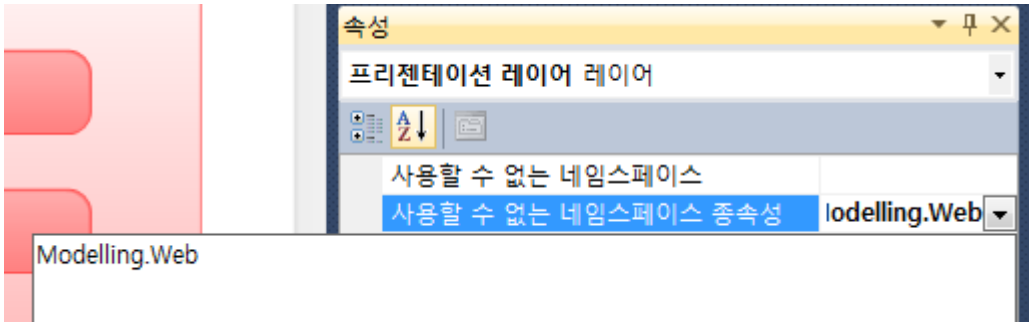


5.6.4. 네임스페이스로 실제 코드의 유효성 검사

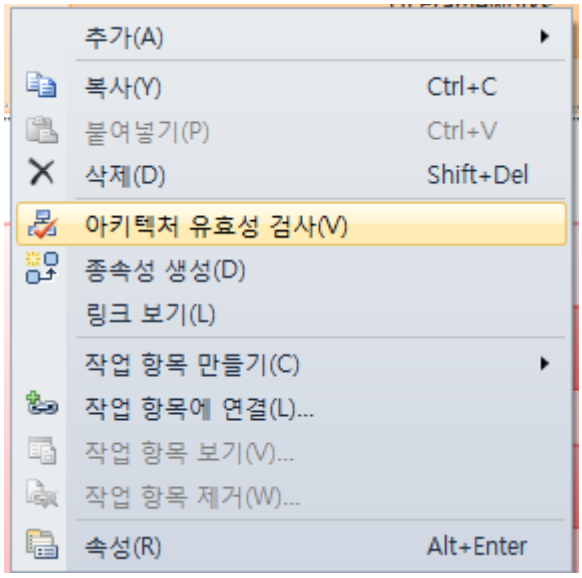
1. 레이어를 선택하여 속성 창을 확인합니다.



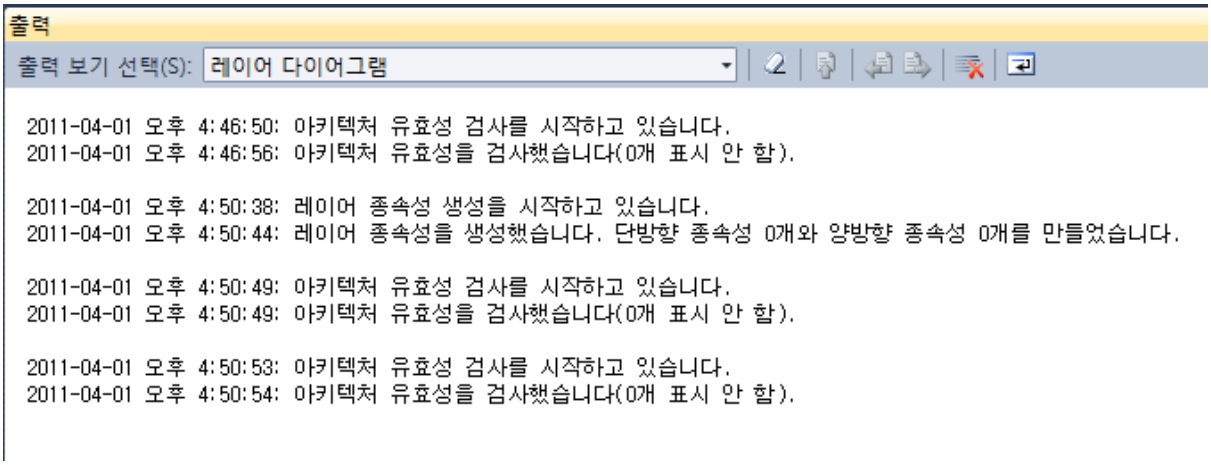
2. 사용할 수 없는 네임스페이스 또는 사용할 수 없는 네임스페이스 종속성에 적절한 네임스페이스를 입력합니다.



3. Layer Diagrams 에서 마우스 오른쪽 버튼을 클릭하여 아키텍처 유효성 검사를 클릭합니다.



4. 출력 창에 아키텍처 유효성 검사 결과가 표시됩니다.



5.6.5. 명령 프롬프트로 유효성을 검사

Visual Studio 2010 에서 명령 프롬프트를 이용하여 빌드를 수행할 때 Layer Diagrams 의 아키텍처 유효성을 검사할 수 있습니다.

이 방법을 이용하면 개발 도구가 없는 경우, 그리고 Team Foundation 의 Team Build 시 아키텍처 유효성을 검사할 때 유용하게 사용할 수 있습니다.

5.6.5.1. MSBUILD 명령 프롬프트로 아키텍처 유효성을 검사하려면?

Visual Studio 2010 명령 프롬프트를 실행한 수 다음과 같이 MSBUILD 명령 옵션으로 모델링 프로젝트의 아키텍처 유효성을 검사합니다.

```
C:\> msbuild <FilePath+ModelProjectFileName>.modelproj
/p:ValidateArchitecture=true
```

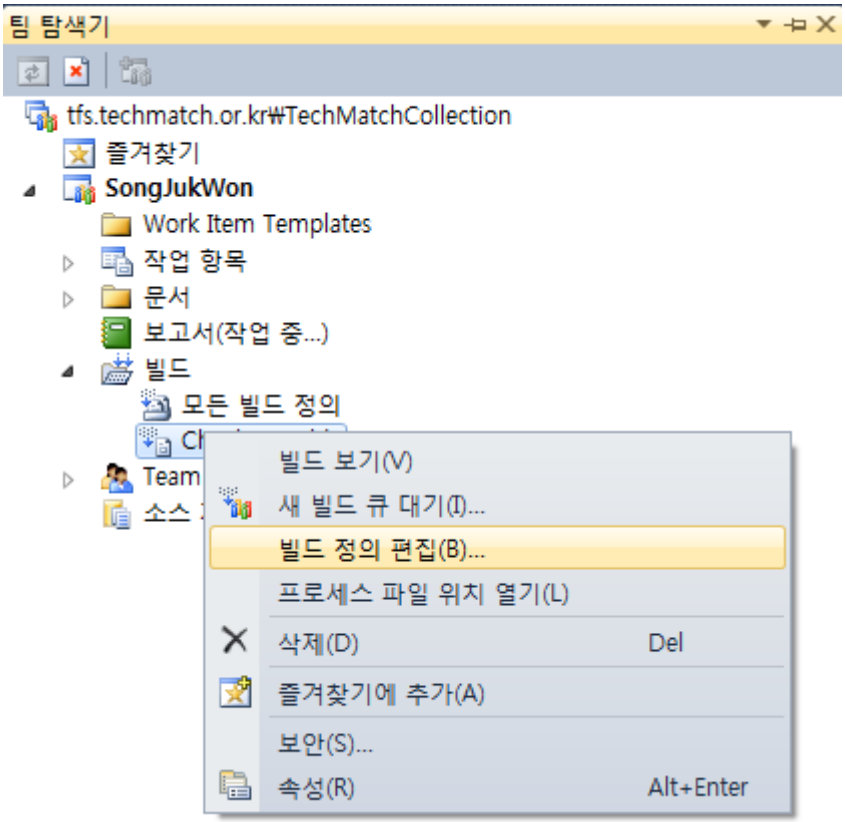
또는 전체 솔루션 중 모델링 프로젝트의 아키텍처 유효성을 검사하려면 다음과 같이 솔루션 파일을 입력한 후 옵션을 지정합니다.

```
C:\> msbuild <FilePath+SolutionName>.sln
/p:ValidateArchitecture=true
```

5.6.5.2. TEAM FOUNDATION 의 TEAM BUILD 로 아키텍처 유효성을 검사하려면?

Team Foundation 의 Team Build 에서 아키텍처 유효성을 검사하려면 다음과 같은 방법으로 MSBUILD 옵션을 지정해 줄 수 있습니다.

1. 팀 프로젝트를 연결하고 아키텍처 유효성을 검사할 빌드 항목을 선택한 후 빌드 정의 편집을 선택합니다.



2. 프로세스 탭으로 이동한 후 3. 고급 탭으로 이동한 후 MSBUILD 인수 항목에 /p:ValidateArchitecture=true 인수를 입력한 후 저장합니다.

3. 고급	
MSBuild 인수	**/p:ValidateArchitecture=true**
MSBuild 플랫폼	Auto
개인 저장 위치	
버전 가져오기	
변경 집합 및 작업 항목 연결	True
소스에 레이블 지정	True
실패 시 작업 항목 만들기	**False**
에이전트 설정	이름이 *이고 태그가 비어 있는 경
저장 폴더에 출력 복사	True
테스트 영향 분석	True
테스트를 사용하지 않도록 설정	False

6. VISUAL STUDIO 2010 VISUALIZATION & MODELING FEATURES PACK

Visual Studio 2010 Visualization & Modeling Features Pack(이하 Modeling Features Pack)은 2010 년 12 월 4 일 MSDN 구독자에게 공개가 되었습니다. 이 Modeling Features Pack 은 기존의 모델링 프로젝트에서 지원하지 않았던 몇 가지 기능을 보강하는 도구로써 Modeling Features Pack 과 Modeling Features Pack Runtime 을 제공해 줍니다.

6.1. MODELING FEATURES PACK 개요

Visual Studio 2010 의 Modeling Features Pack 은 다음과 같은 기능을 제공해 줍니다.

- C++ 및 ASP.NET 프로젝트의 종속성 그래프 지원
- C++ 코드에서 Layer Diagrams 의 아키텍처 유효성 지원
- UML Class Diagrams 을 코드로 변환하거나 코드를 UML 다이어그램으로 변환
- XMI 2.1 파일로 내보내기 기능
- 다이어그램의 요소와 Team Foundation Server 의 작업 항목 연동 강화
- Layer Diagrams 확장

6.2. ASP.NET 웹 프로젝트의 종속성 그래프 분석

기존에 종속성 그래프 분석은 닷넷의 어셈블리나 클래스를 기준으로 분석이 되어 웹 프로젝트의 경우 ASPX 파일이나 그에 해당하는 종속성을 분석하기란 사실상 불가능 하였습니다. 다만 웹 프로젝트를 클래스나 어셈블리 수준에서 분석이 가능하였으나 웹 프로젝트 특성을 잘 반영한 종속성을 분석하기는 어려움이 많았습니다.

이번 ASP.NET 종속성 그래프는 웹 프로젝트라는 특성을 잘 반영하여 페이지 파일이나 공통되는 마스터 페이지 등 종속성 분석이 한결 자연스러워졌습니다.

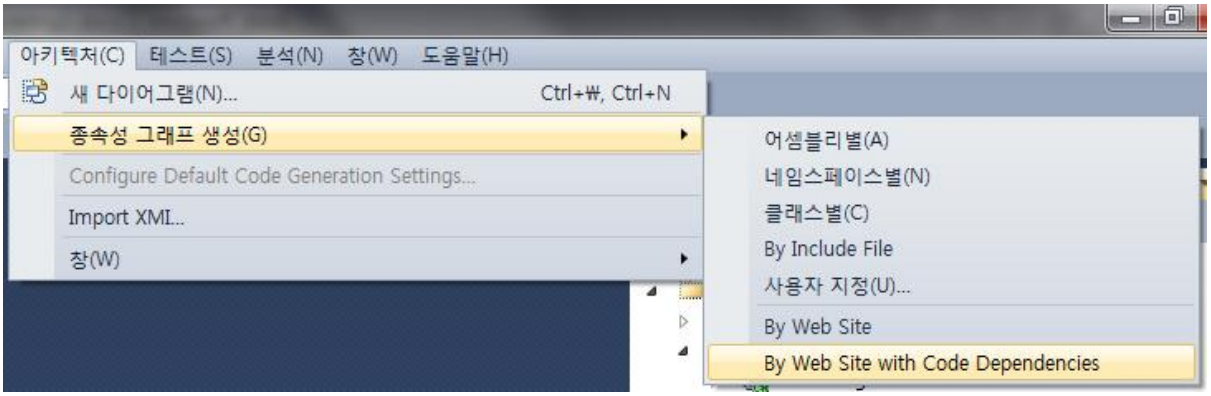
ASP.NET 종속성 그래프 생성은 다음과 같은 웹 프로젝트 형식을 지원합니다.

- ASP.NET 웹 사이트
- ASP.NET 웹 프로젝트

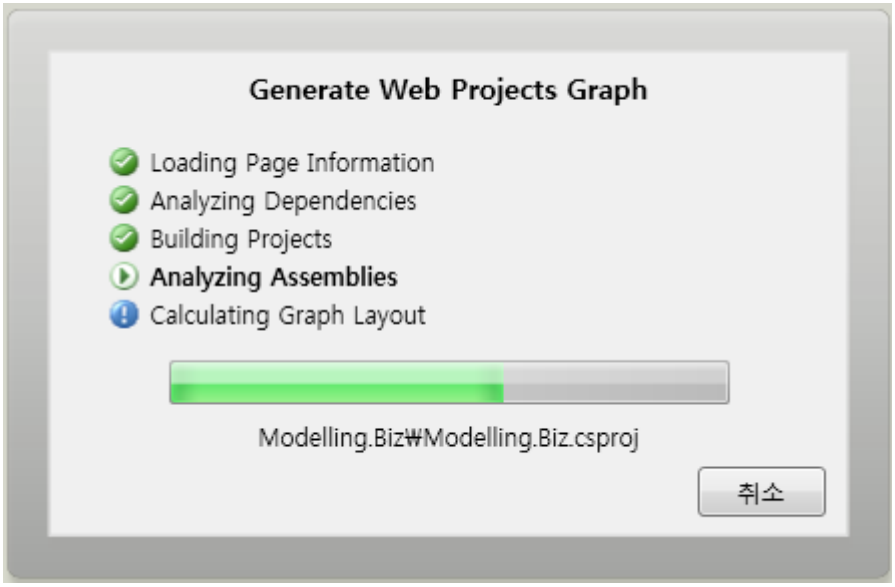
- ASP.NET MVC 웹 프로젝트

6.2.1. 웹 프로젝트 종속성 그래프를 만들려면?

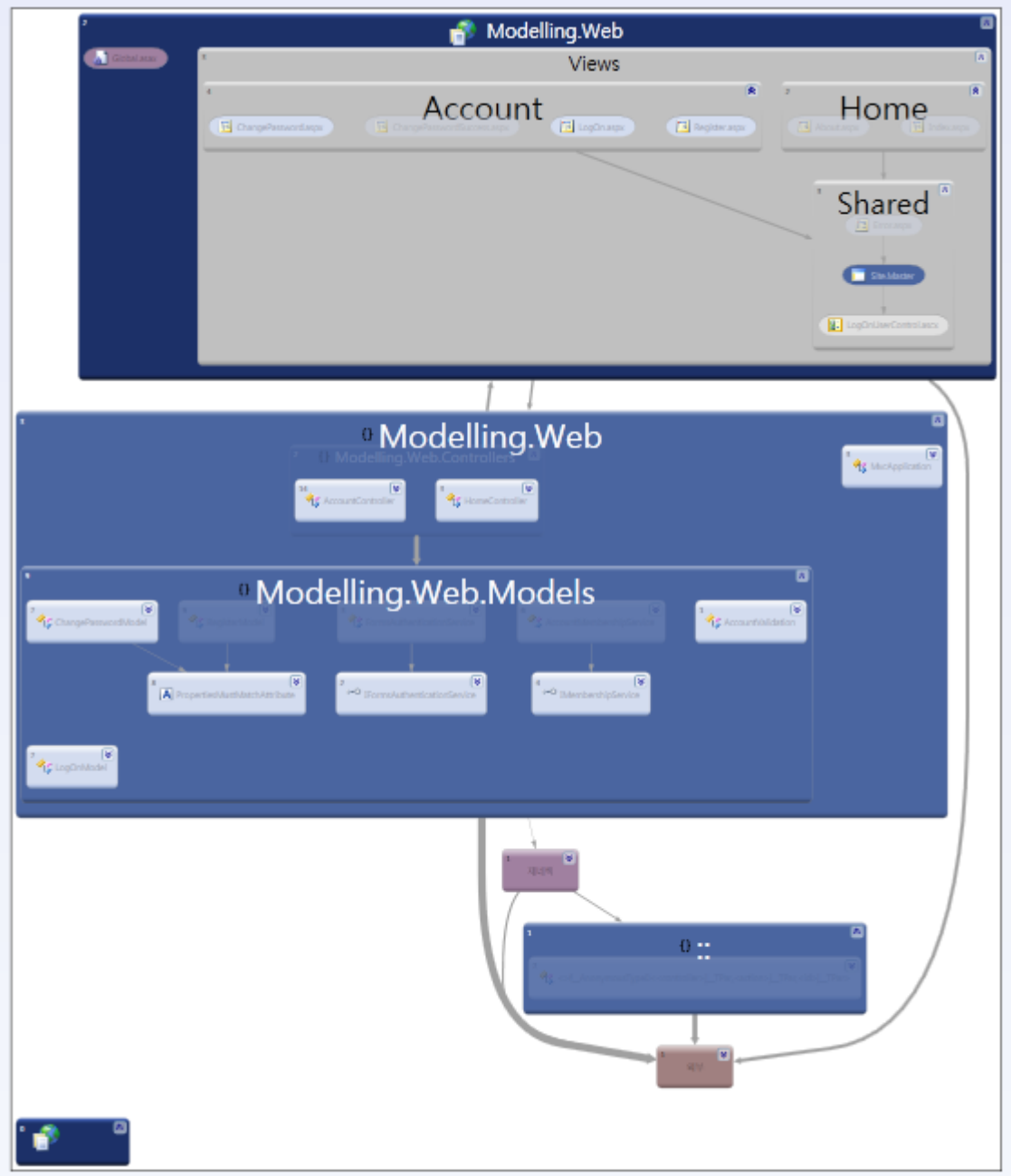
1. 종속성 그래프를 분석할 웹 프로젝트를 로드 합니다.
2. 아키텍처->종속성 그래프 생성에 By Web Site 외 모두 두 가지 항목이 새로 생겼습니다. 코드 종속성과 함께 ASP.NET 웹 프로젝트를 분석하려면 By Web Site with Code Dependencies 를 선택합니다.



3. 솔루션에서 웹 프로젝트의 종속성을 분석합니다.

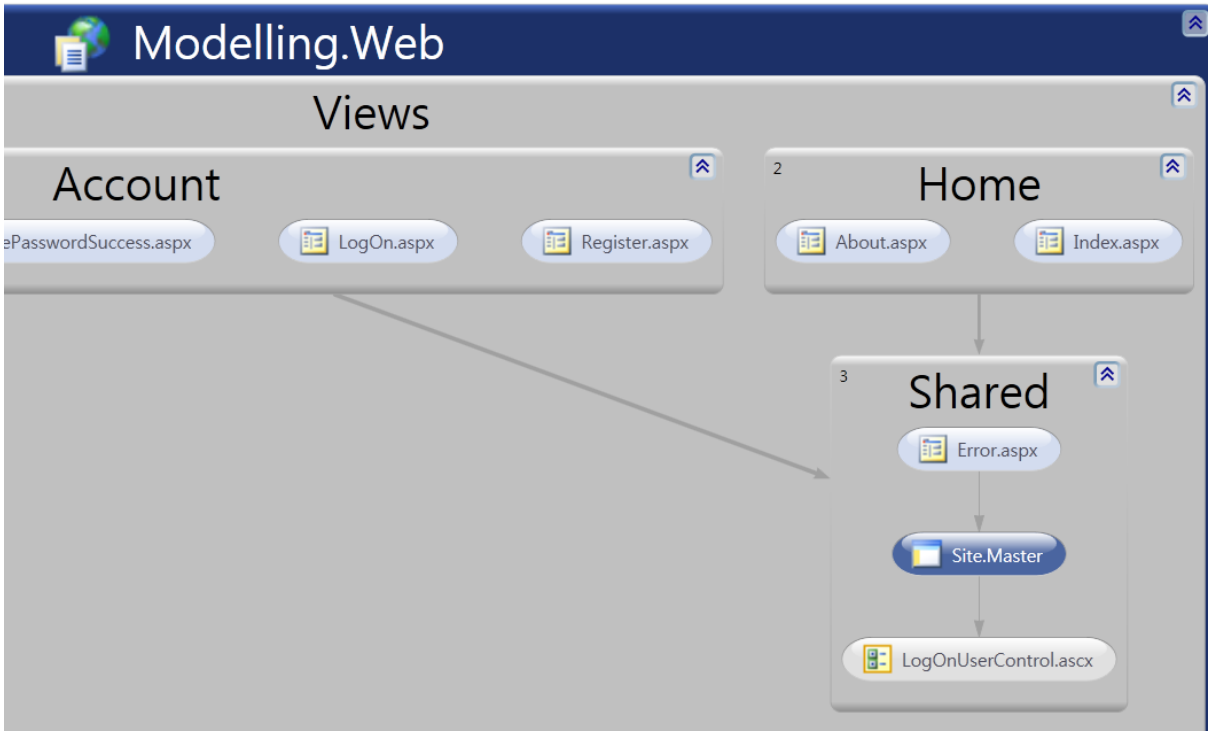


4. 다음은 웹 프로젝트의 전체를 코드 종속성 분석과 함께 분석된 결과 입니다.



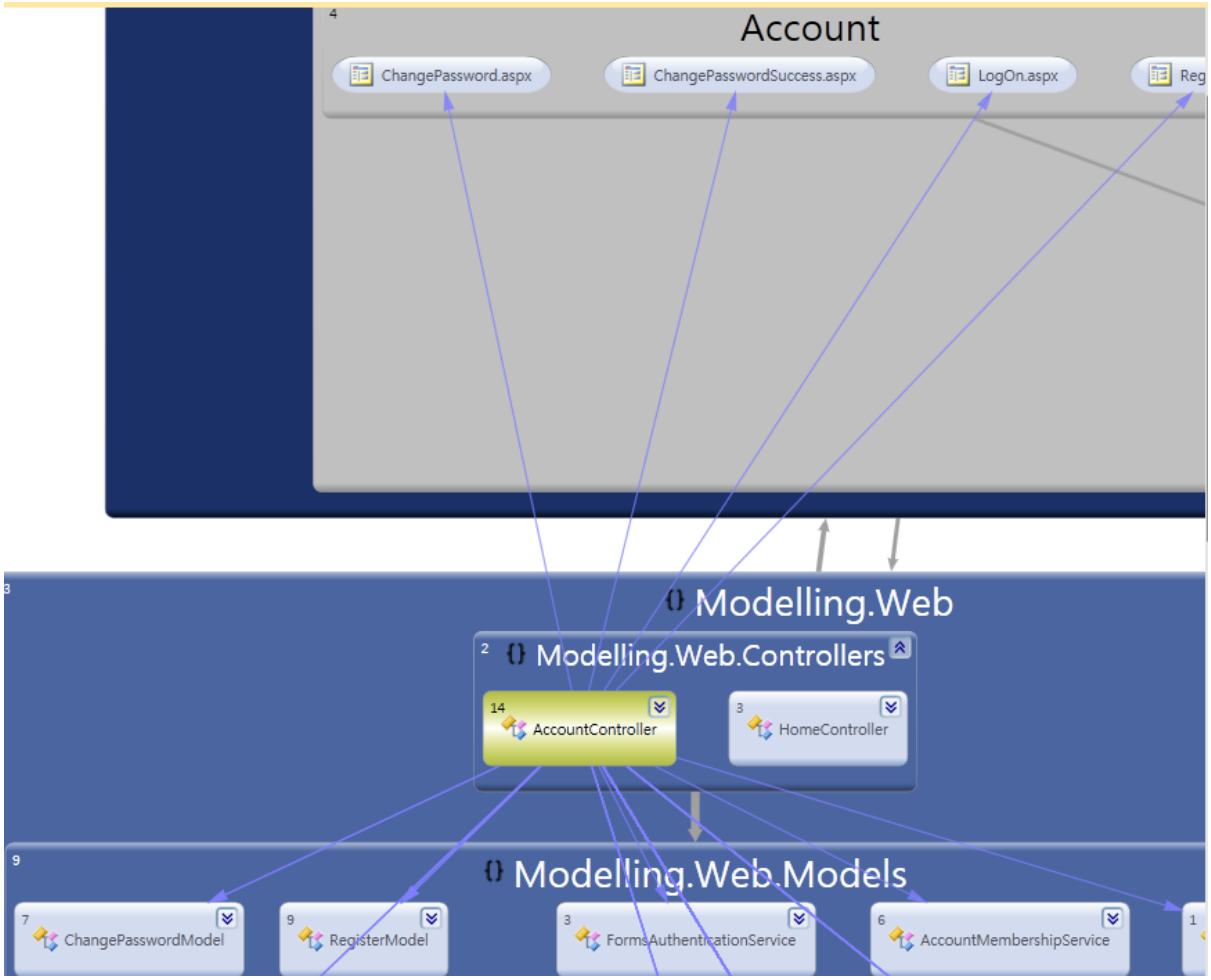
6.2.2. 종속성 그래프를 분석하려면?

다음과 같이 종속성 그래프를 확대하면 각 페이지 간의 종속성을 보여줍니다.



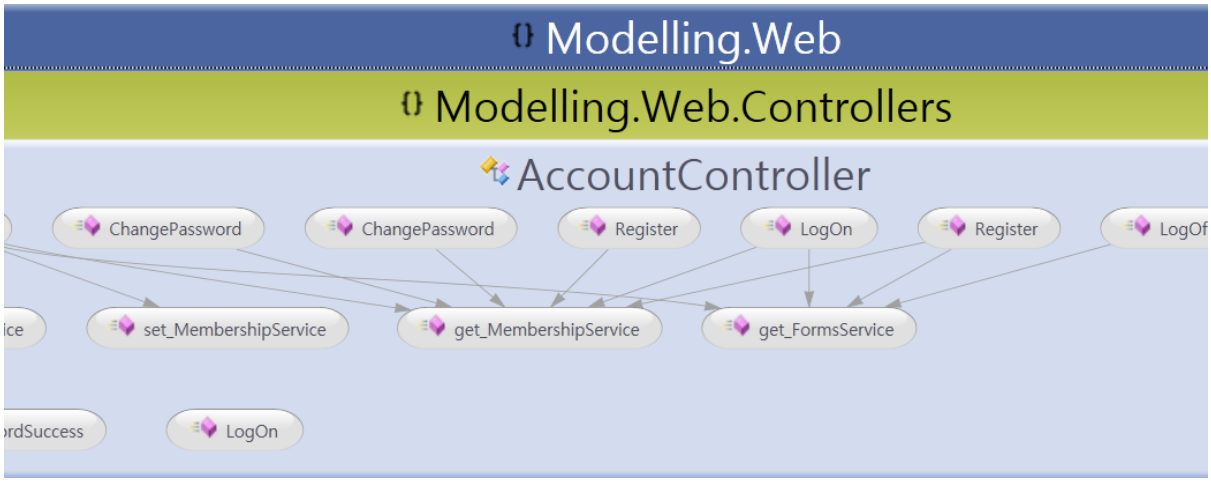
이 종속성 그래프는 DGML(Directed Graph Markup Language)로써 웹 프로젝트 뿐만 아니라 Visual Studio 2010 의 전반적인 분석에 사용되는 전용 마크업 언어이기도 합니다.

만약 특정 페이지가 어떤 모듈이나 페이지간에 종속성을 미치는지 확인하기 위해서 해당 노드를 선택하면 됩니다. 예를 들어 아래의 AccountController 클래스가 가지는 종속적인 관계는 ChangePassword.aspx, ChangePasswordSuccess.aspx 등의 각 페이지들과 인증과 관련된 FormAuthenticationService, ChangePasswordModel 클래스 등에 종속적인 관계를 갖는 것을 확인할 수 있습니다.

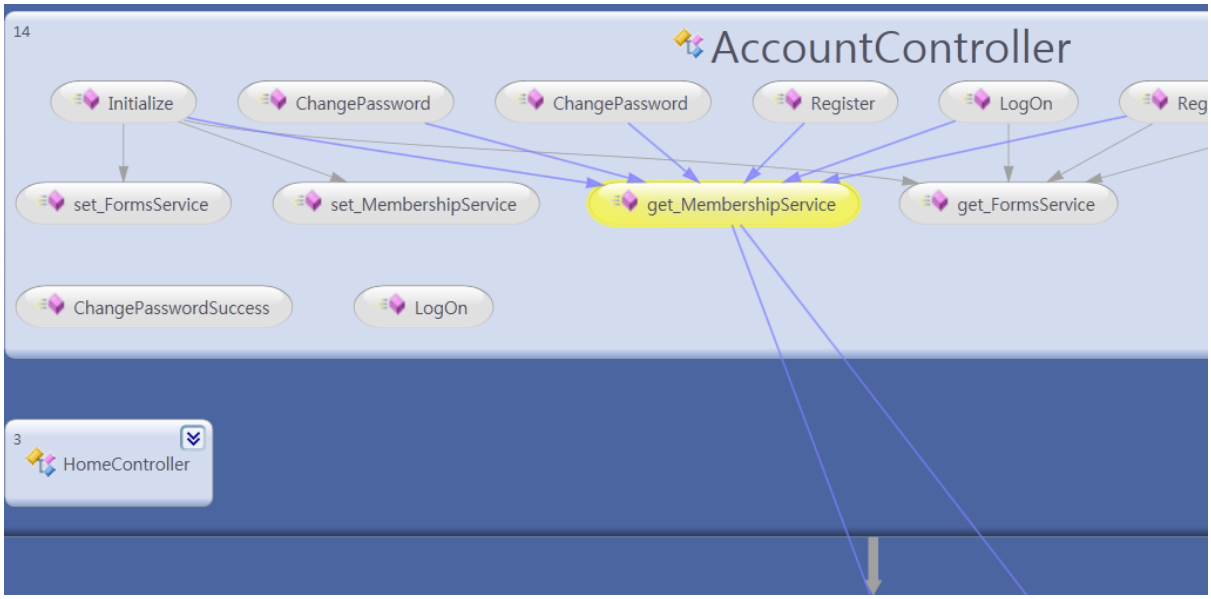


이 AccountController 클래스의 내부를 좀더 자세히 보기 위해서는 AccountController 노드의 오른쪽 상단의 화면표 아이콘을 선택합니다.

그럼 AccountController 가 구현하는 메서드나 프로퍼티 정보를 표시합니다.

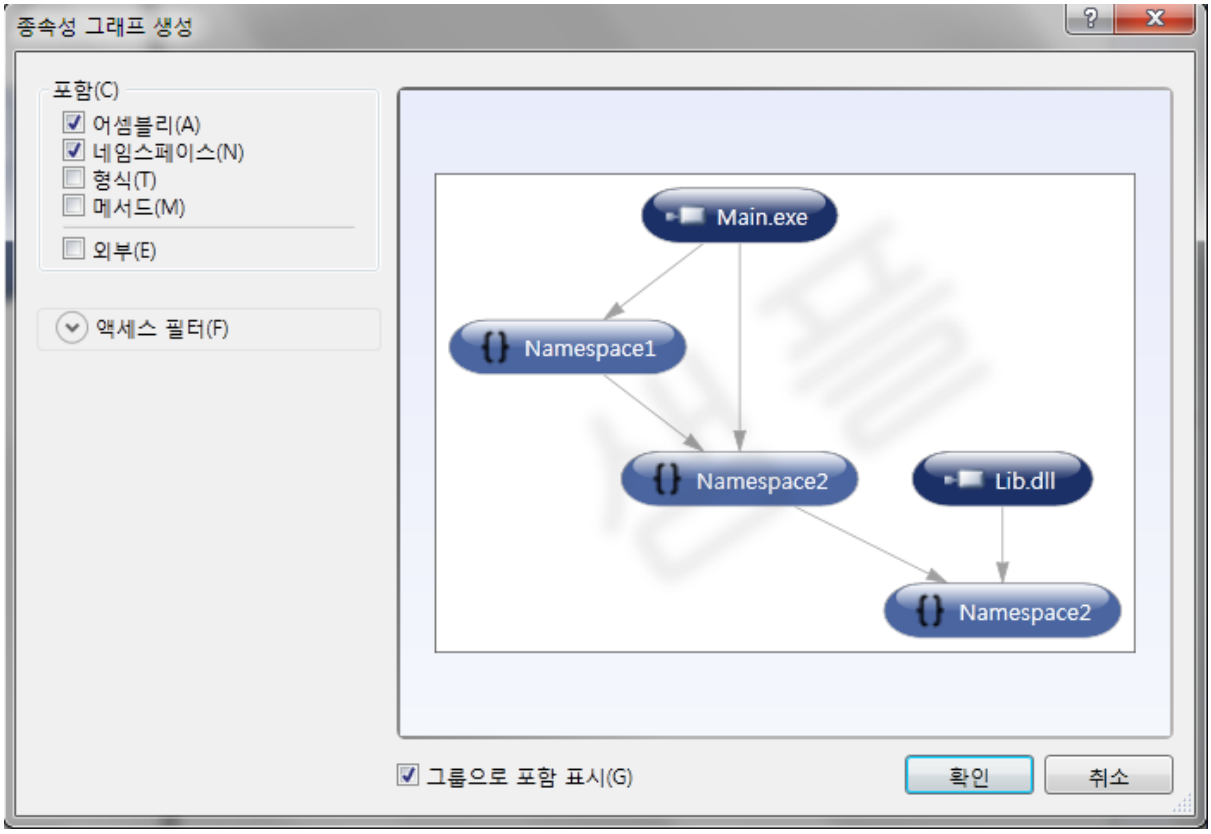


만약 이 메서드들이 어떤 종속성을 갖는지 알기 위해서 메서드의 하나의 노드를 선택하면 됩니다. 예를 들어 아래의 get_MembershipService (원래는 속성(Property)임)은 다양한 메서드에서 호출되는 속성인 것을 확인할 수 있습니다.

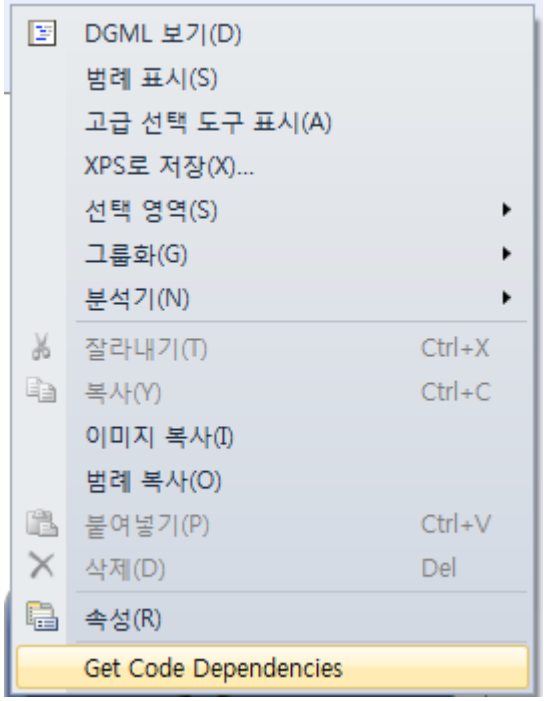


이러한 종속성 정보는 얼마나 복잡하게 서로간에 영향력을 미치는지 알 수 있습니다. 예를 들어, 화살표를 많이 받으면 받을수록 그 코드의 변경이 주는 전파력은 상당히 높아지며, 코드에 버그가 있다면 그 버그의 전파력은 종속성이 갖는 크기만큼인 것이 될 것이라 예상할 수 있습니다.

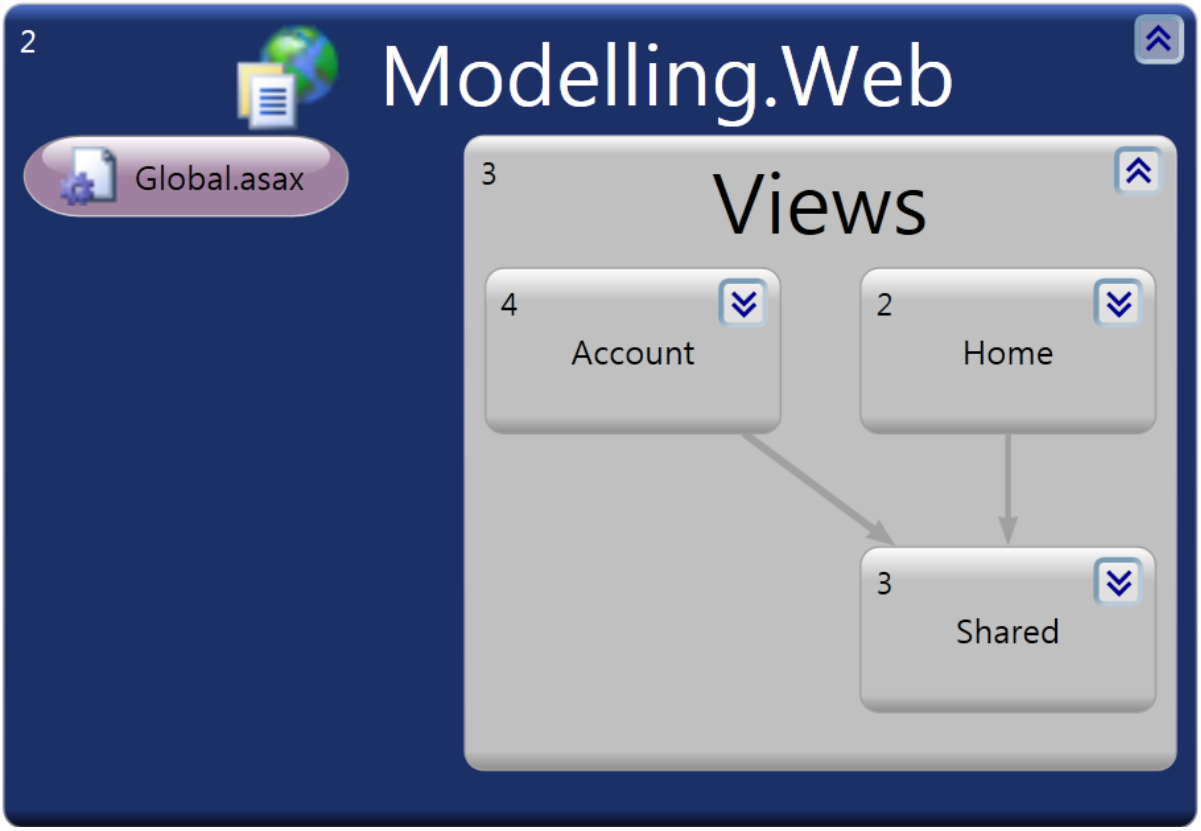
종속성 그래프에서 화살표를 더블 클릭하여 분석할 종속성의 레벨을 지정할 수 있습니다.



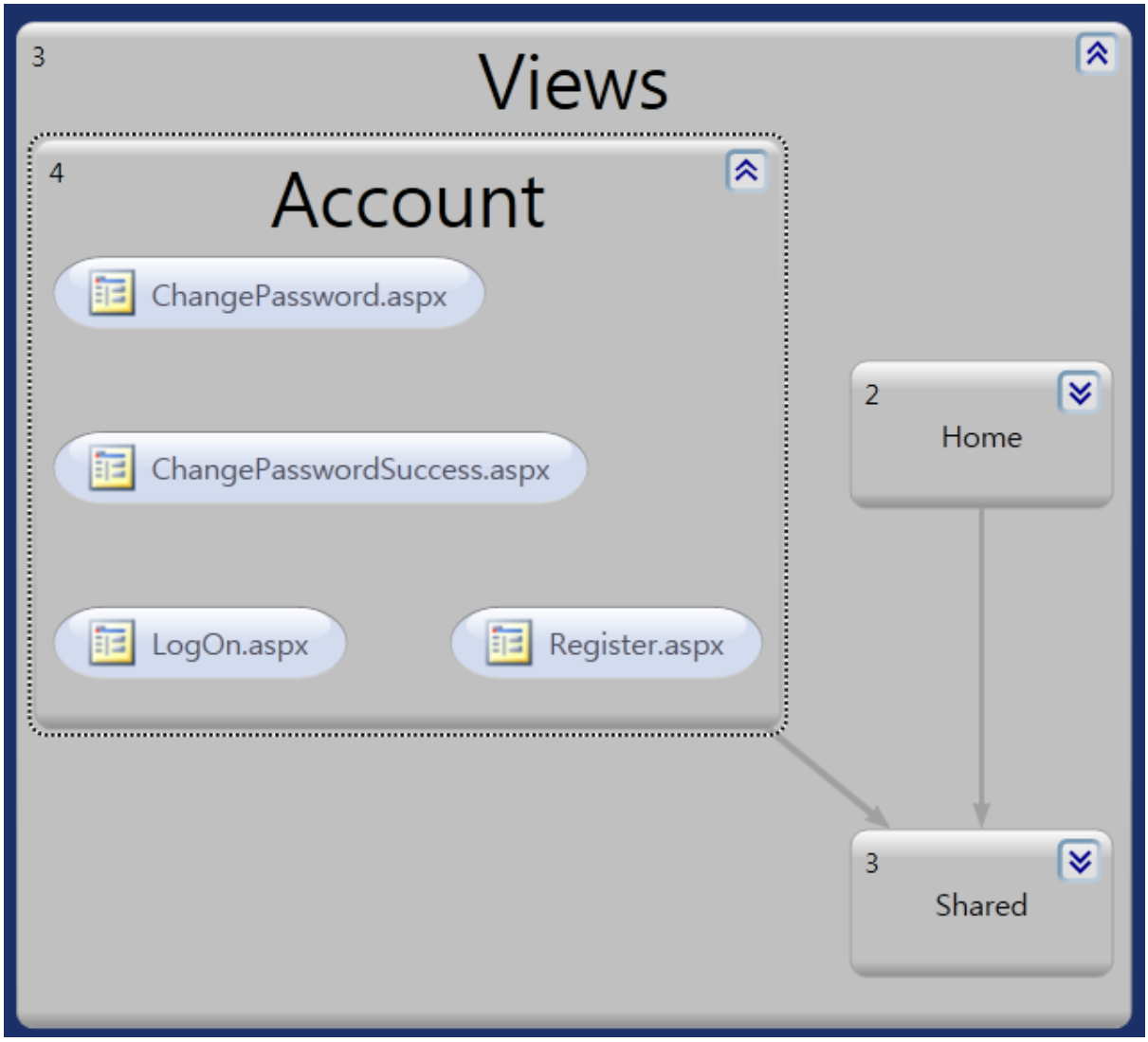
만약 웹 프로젝트를 코드 레벨에서 분석을 하길 원한다면 종속성 그래프의 컨텍스트에서 마우스 오른쪽 버튼을 클릭하여 Get Code Dependencies 를 선택합니다.

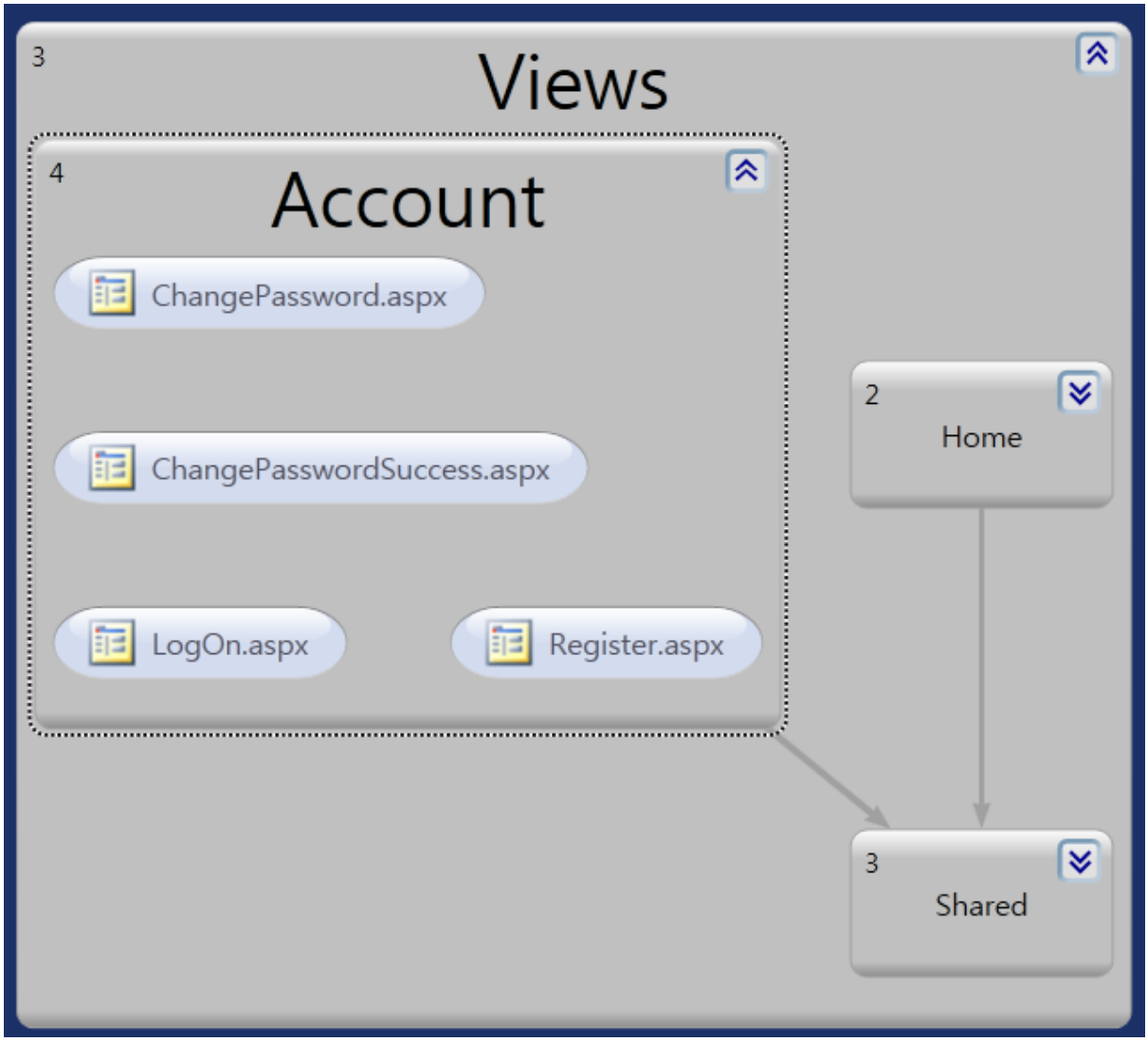


그럼 웹 프로젝트의 노드를 확장하면 웹 프로젝트의 페이지 등의 종속성의 분석이 아닌 코드 레벨에서 종속성을 분석해 줍니다.



코드 레벨의 종속성을 분석하면 다음 그림과 같이 해당 노드와 종속되는 페이지를 확인할 수 있습니다.


3
Views
4
Account
ChangePassword.aspx
ChangePasswordSuccess.aspx
LogOn.aspx
Register.aspx
2
Home
3
Shared

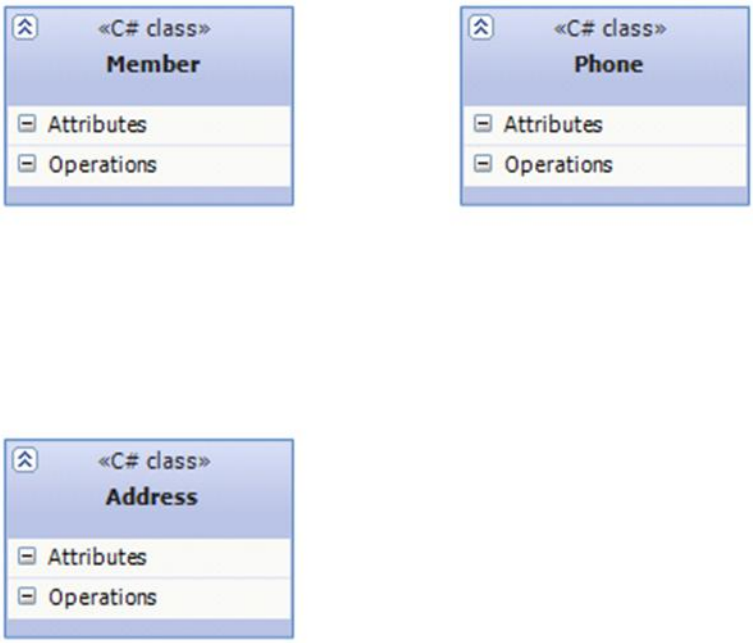


6.3. UML 다이어그램과 코드 간의 상호 변환

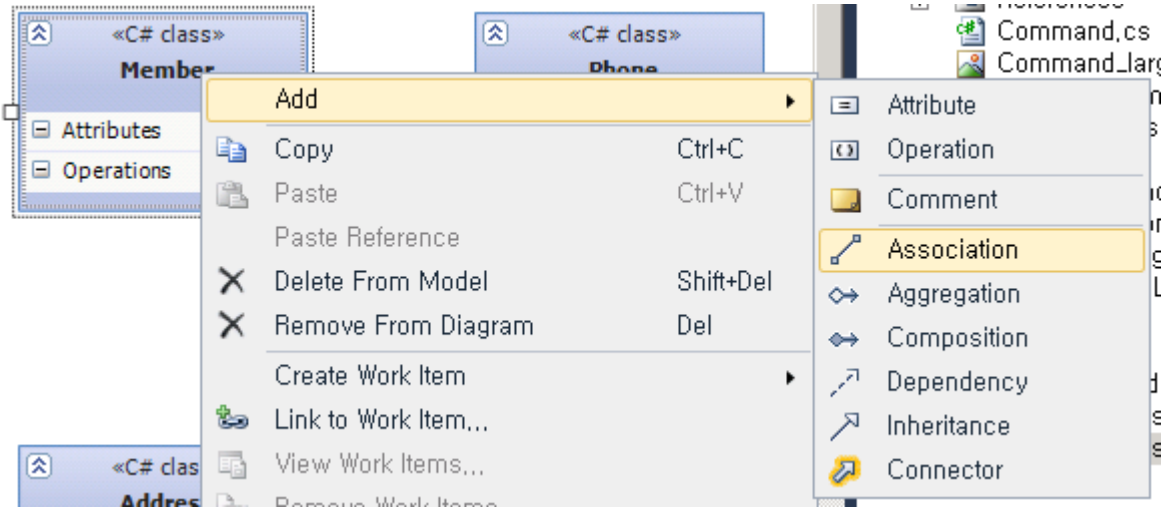
6.3.1. CLASS DIAGRAMS 을 코드로 변환하기

Class Diagrams 을 코드로 변환하기 위해서 모델링 프로젝트에서 UML Class Diagrams 을 새로 추가 합니다.

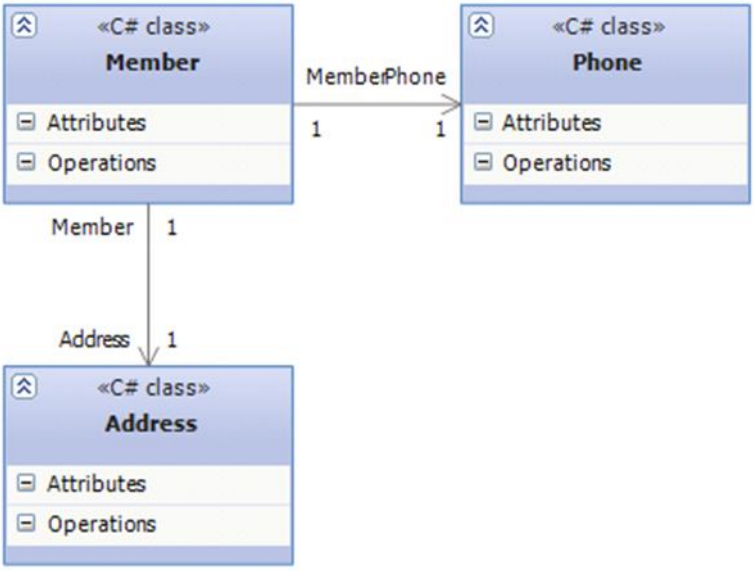
1. 간단하게 다음과 같이 클래스 모델링을 해 봅시다. 각각 Member, Phone, Address 클래스 요소를 추가 합니다.



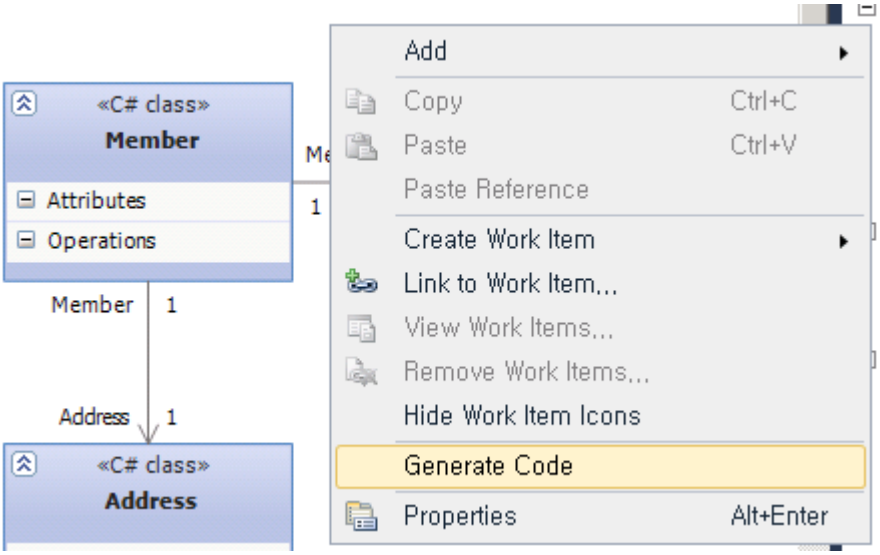
2. Member 클래스 요소를 Phone, Address 클래스 요소와 연결(Association) 관계로 만듭니다. 연결 관계로 만들기 위해서 Member 클래스 요소를 마우스 오른쪽 버튼으로 클릭한 후 추가->연결(Association) 항목을 클릭하여 각각 Phone, Address 클래스 요소와 연결합니다.



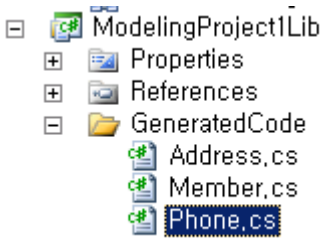
3. 다음의 그림은 연결(Association) 을 Phone, Address 클래스 요소와 연결한 그림입니다.



4. 모든 작업이 완료 되었으면 Class Diagrams 의 컨텍스트 메뉴에서 Generate Code 를 클릭합니다.



5. 다음과 같이 새로운 프로젝트가 생성이 되고, GenerateCode 폴더에 Class Diagrams 의 클래스 요소들이 코드로 변환이 됩니다.



6. Member.cs 클래스 파일은 다음과 같이 Phone, Address 클래스와 연결 관계가 속성으로 변환이 된 것을 확인할 수 있습니다.

```
Member.cs   UMLClassDiagram1.classdiagram
Member                                  Addre
//------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Changes to this file will be lost if th
// </auto-generated>
//------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Member
{
    public virtual Phone Phone
    {
        get;
        set;
    }

    public virtual Address Address
    {
        get;
        set;
    }

}
```

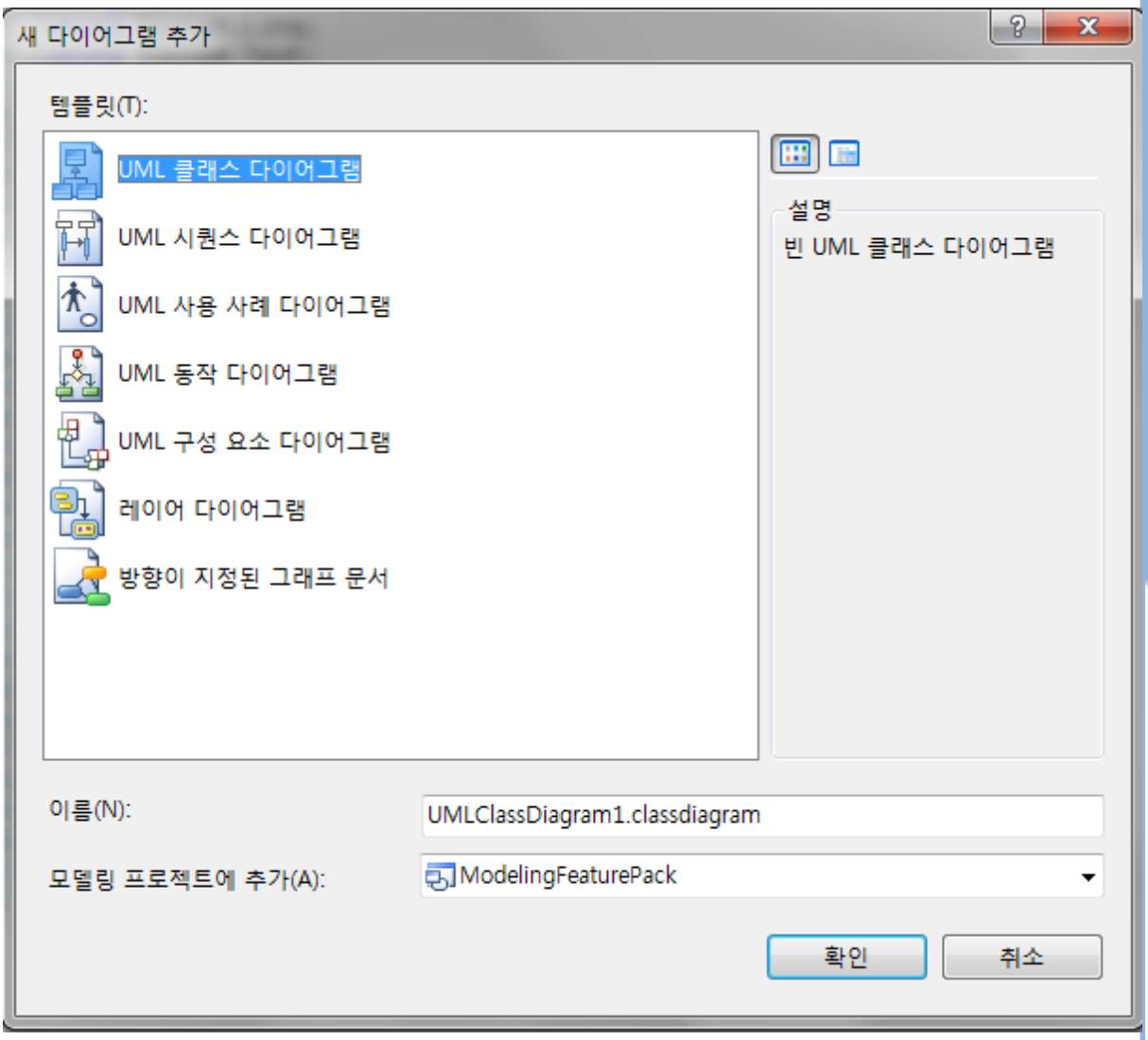
6.3.2. 코드를 CLASS DIAGRAMS 으로 변환하기

마찬가지로 Visual Studio 2010 Modeling Features Pack 은 기존에 존재하는 코드를 UML Class Diagrams 으로 변환할 수 있습니다.

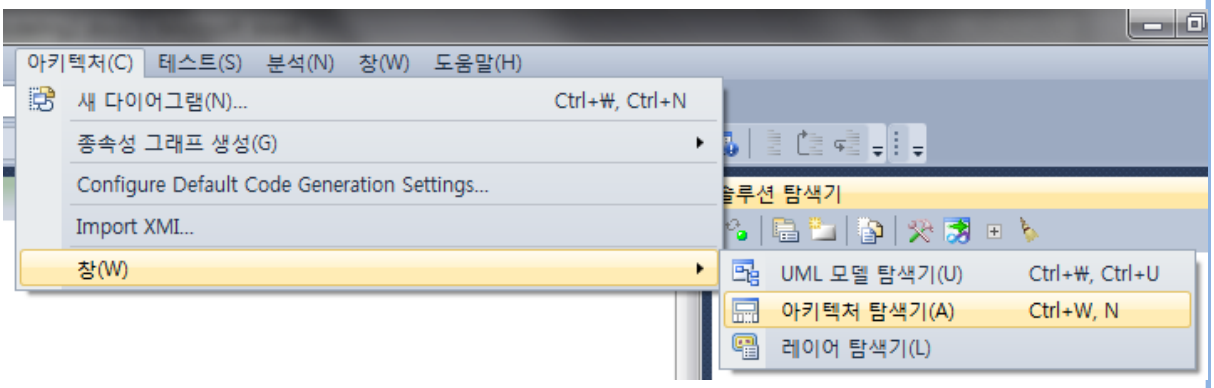
1. 아래와 같이 간단하게 코드를 작성합니다. 기존 코드가 있으시면 기존 코드를 사용하셔도 무방합니다.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace Modelling.Dac
7  {
8      public class Member
9      {
10         public string MemberID { get; set; }
11         public Phone Phone { get; set; }
12         public Address Address { get; set; }
13     }
14
15     public class Phone
16     {
17         public string MemberID { get; set; }
18         public string Code { get; set; }
19         public string Number1 { get; set; }
20         public string Number2 { get; set; }
21     }
22
23     public class Address
24     {
25         public string MemberID { get; set; }
26         public string Country { get; set; }
27         public string ZipCode { get; set; }
28         public string Address1 { get; set; }
29         public string Address2 { get; set; }
30     }
31 }
32
```

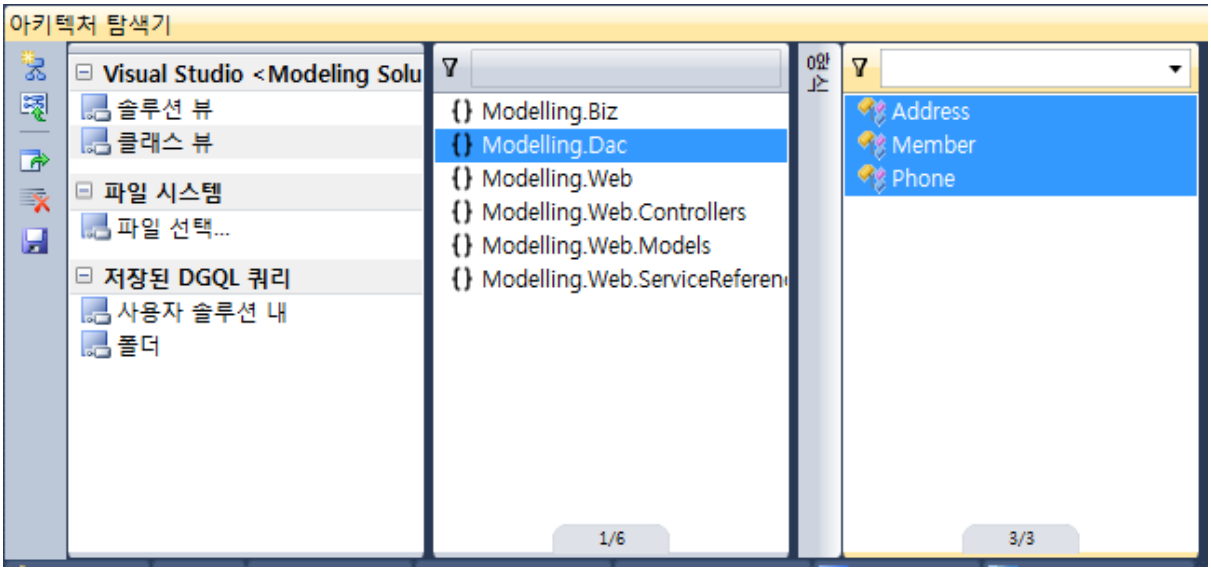
2. 만약 모델링 프로젝트가 현재 솔루션에 없다면 모델링 프로젝트를 생성합니다. 아키텍처->새 다이어그램을 선택하여 UML Class Diagrams 을 생성합니다.



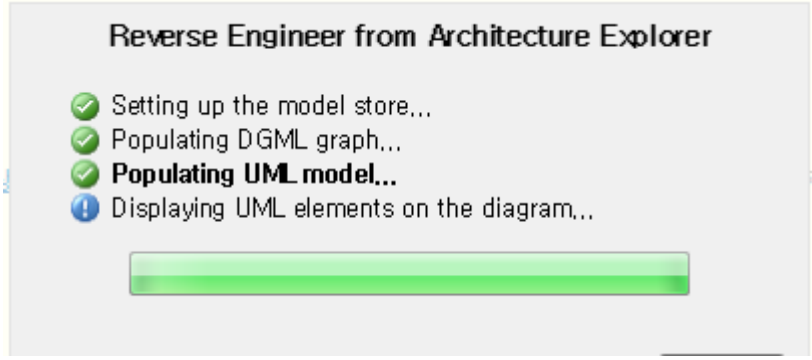
3. 아키텍처->창->아키텍처 탐색기를 선택합니다.



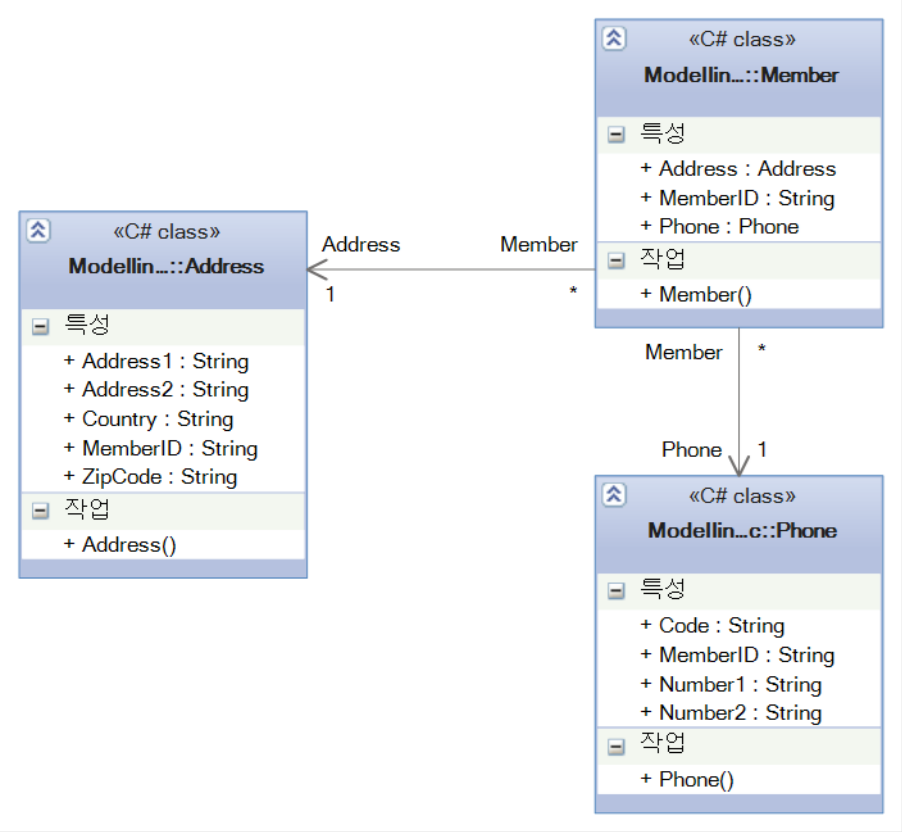
4. 아키텍처 탐색기에서 UML Class Diagrams 으로 변환할 코드를 찾아 선택합니다.



5. 선택한 클래스 또는 코드 파일을 UML Class Diagrams 디자이너로 드래그&드랍 합니다.



6. 그럼 기존의 코드가 아래와 같이 UML Class Diagrams 으로 변환이 완성되었습니다.



6.4. XMI 가져오기

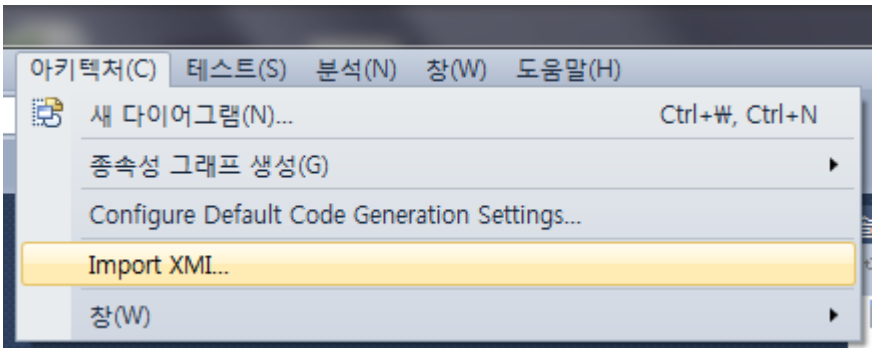
XMI(XML Metadata Interchange)는 메타데이터를 통해 정보를 교환하기 위한 표준으로 XML 형식의 데이터를 사용합니다. 다양한 모델링 프레임워크나 도구에서 독자적인 포멧을 사용하지만 XMI 내보내기 등의 기능으로 다양한 모델링 도구에서 모델링 정보를 교환하거나 공유할 수 있습니다. (참고 http://en.wikipedia.org/wiki/.xmi)

이 표준 모델링 다이어그램의 XMI 를 Visual Studio 2010 에서 불러올 수 있는 기능을 제공합니다. XMI 파일을 가져오려면 다음과 같이 진행하시면 됩니다.

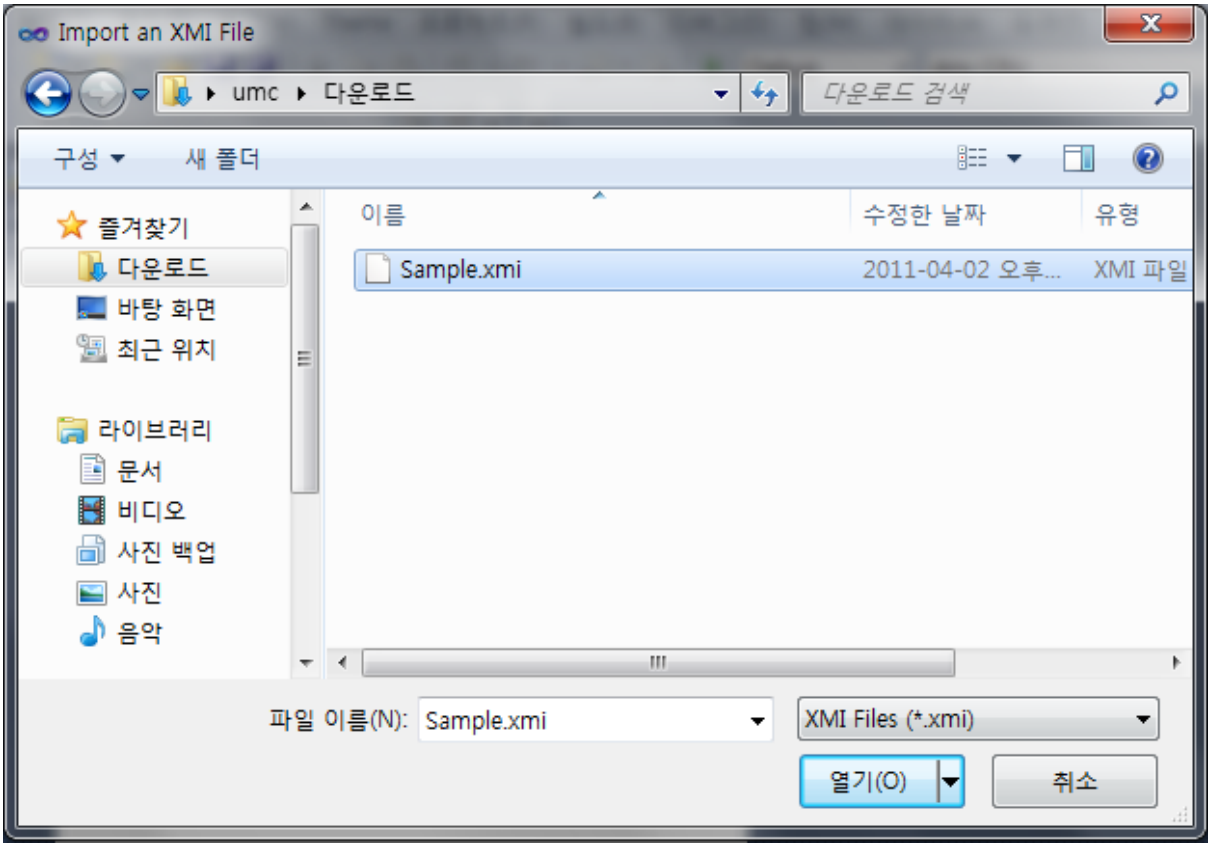
만약 사용할 수 있는 .XMI 파일이 없다면 다음의 전자정부 표준 프레임워크 사이트에서 샘플을 다운로드 받으실 수 있습니다.
(http://open.egovframe.go.kr/scm/viewvc.php/trunk/DEV_IDE_PLUGIN/egovframework.dev.imp.code

gen.modeltest/xmi_sample/sample.xmi?diff_format=s&logsort=cvs&sortby=log&view=markup&root=egovframedev)

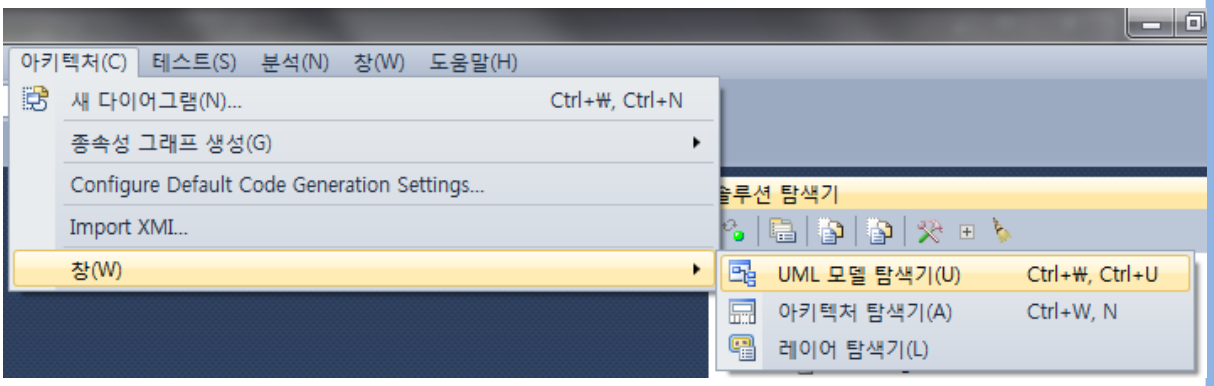
1. 아키텍처->Import XMI... 을 선택합니다.



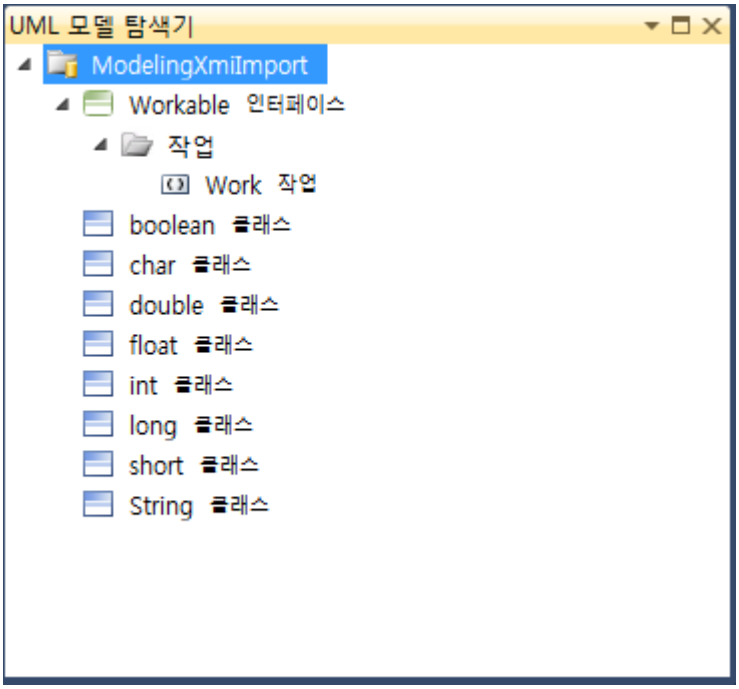
2. 확장명이 .XMI 인 파일을 찾아 선택한 후 확인을 클릭합니다.



3. 가져온 XMI 요소의 형식을 확인하려면 아키텍처->창->UML 모델 탐색기를 선택합니다.



4. UML 모델 탐색기에 XMI 파일에서 가져온 요소가 나타납니다.



6.5. TEAM FOUNDATION 2010 연동

기존의 Visual Studio 2010 모델링 프로젝트는 Team Foundation Server 작업 항목과의 연동을 이미 지원했습니다. 하지만 UML 다이어그램의 요소와 작업 항목과의 연결이나 관리 작업 부분에서 불편한 점이 있었습니다.

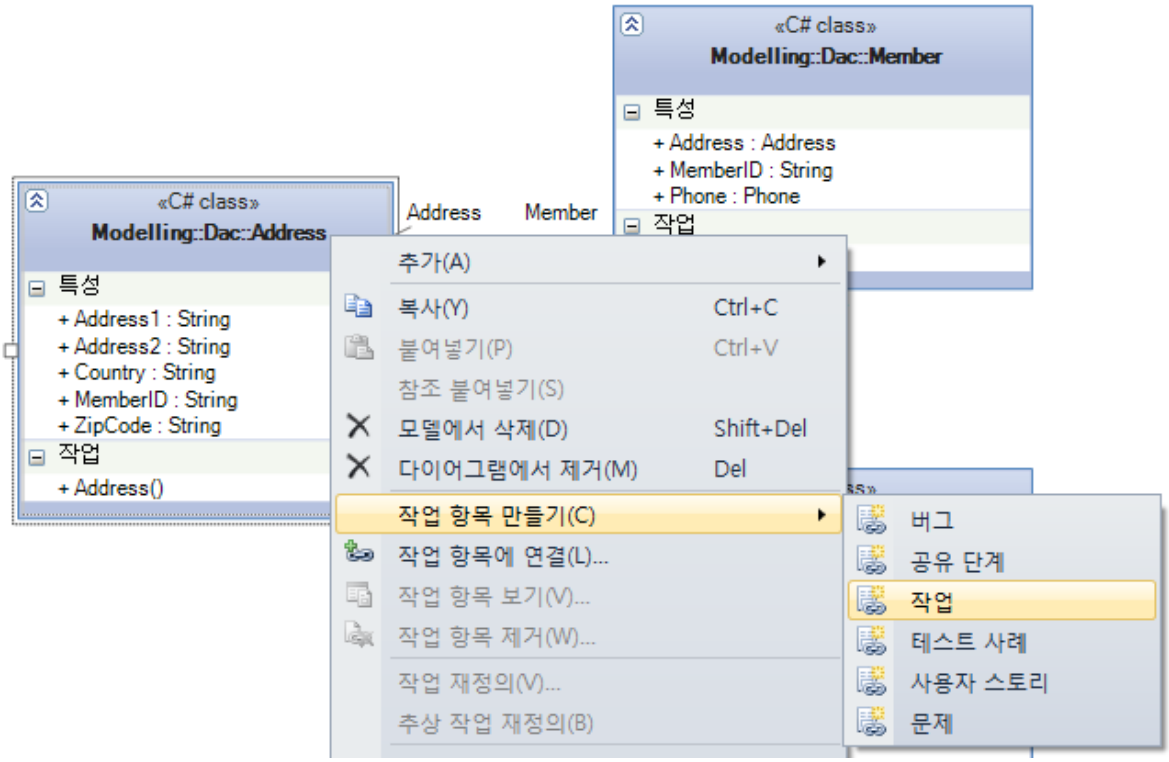
이번 Modeling Features Pack 은 Team Foundation Server 와 연동하면서 바로 작업을 연결하거나 만들고, 또는 추적할 수 있는 부분이 강화되어 다이어그램과 작업 간의 관리가 훨씬 용이해 졌습니다.

6.5.1. 다이어그램의 요소와 작업 연결

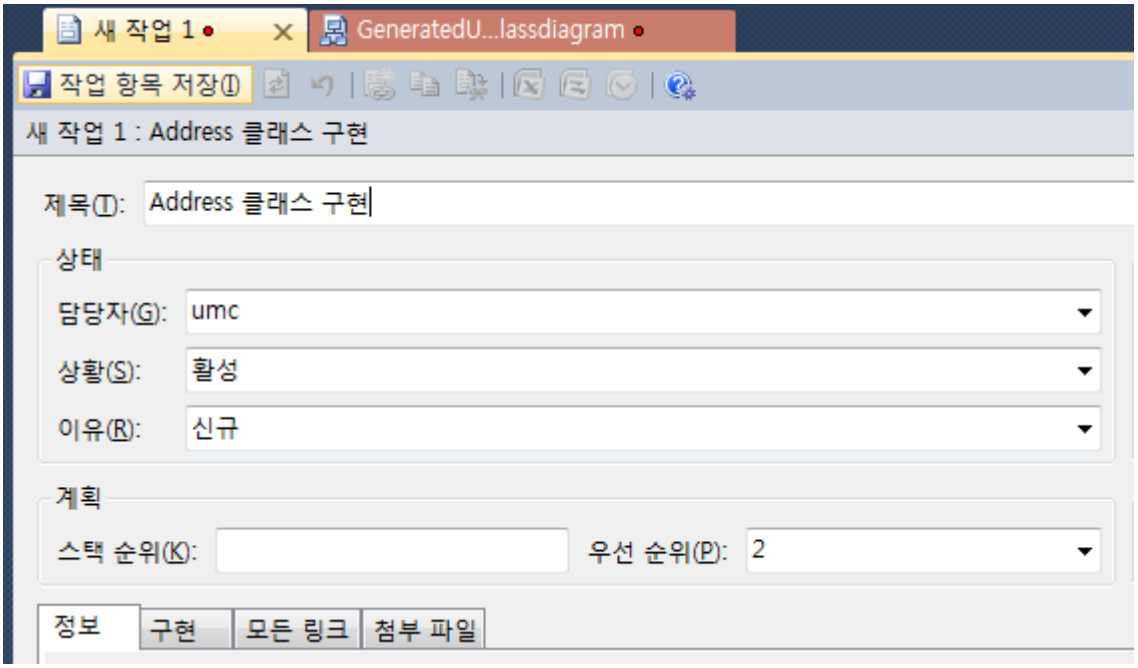
모델링 프로젝트의 다이어그램의 모든 요소는 Team Foundation Server 에 작업을 만들거나 연결할 수 있습니다. 가령, 모델리한 컴포넌트를 특정 개발자에게 작업을 할당하거나 그 작업의 진척율을 확인하고 보고를 받을 수 있습니다.

6.5.1.1. 모델링 프로젝트에서 작업을 만들려면?

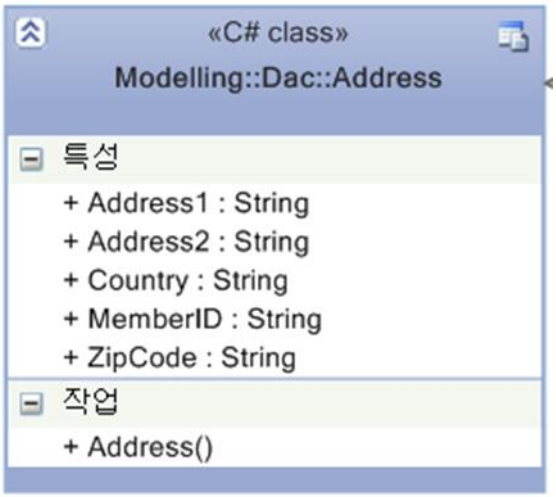
1. 먼저 작업과 연결할 모델링 다이어그램을 엽니다. 작업과 연결할 요소에 마우스 오른쪽 버튼을 클릭하여 작업 항목 만들기-><작업항목형식> 을 선택합니다.

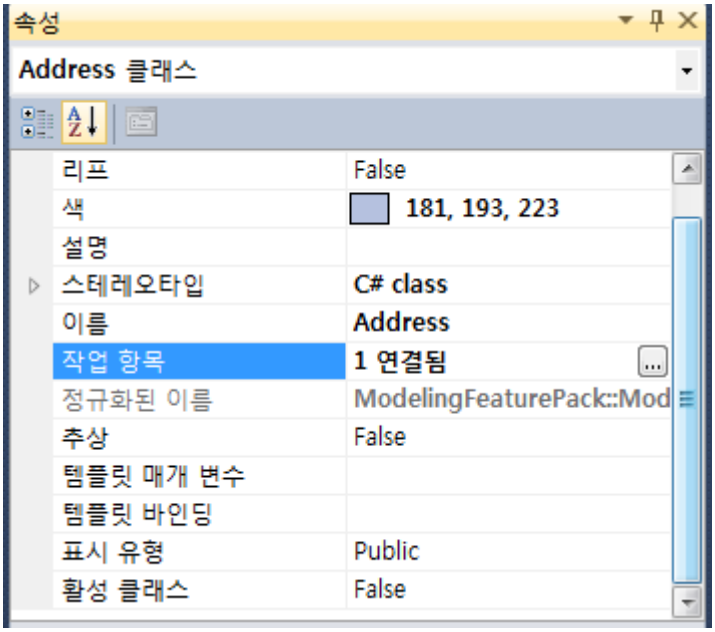


2. 작업 항목 만들기 화면에서 필요한 항목을 입력한 후 작업 항목 저장 버튼을 클릭합니다.



3. Class Diagrams 의 요소에 작업 항목이 연결되었음을 의미하는 아이콘이 표시되고, 요소의 속성 창에 1 개의 작업 항목이 연결된 표시가 나타납니다.

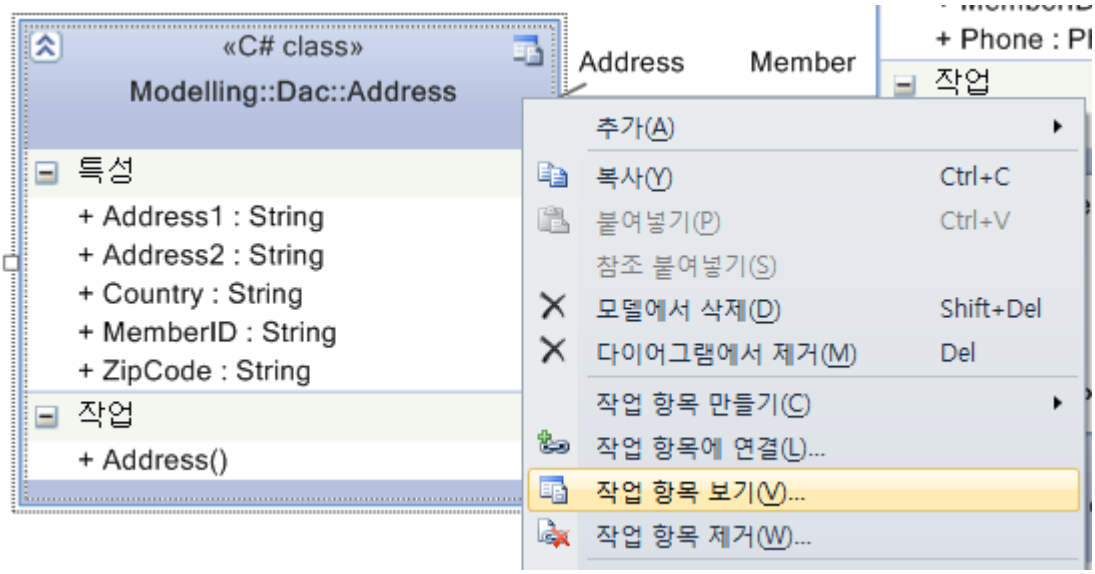




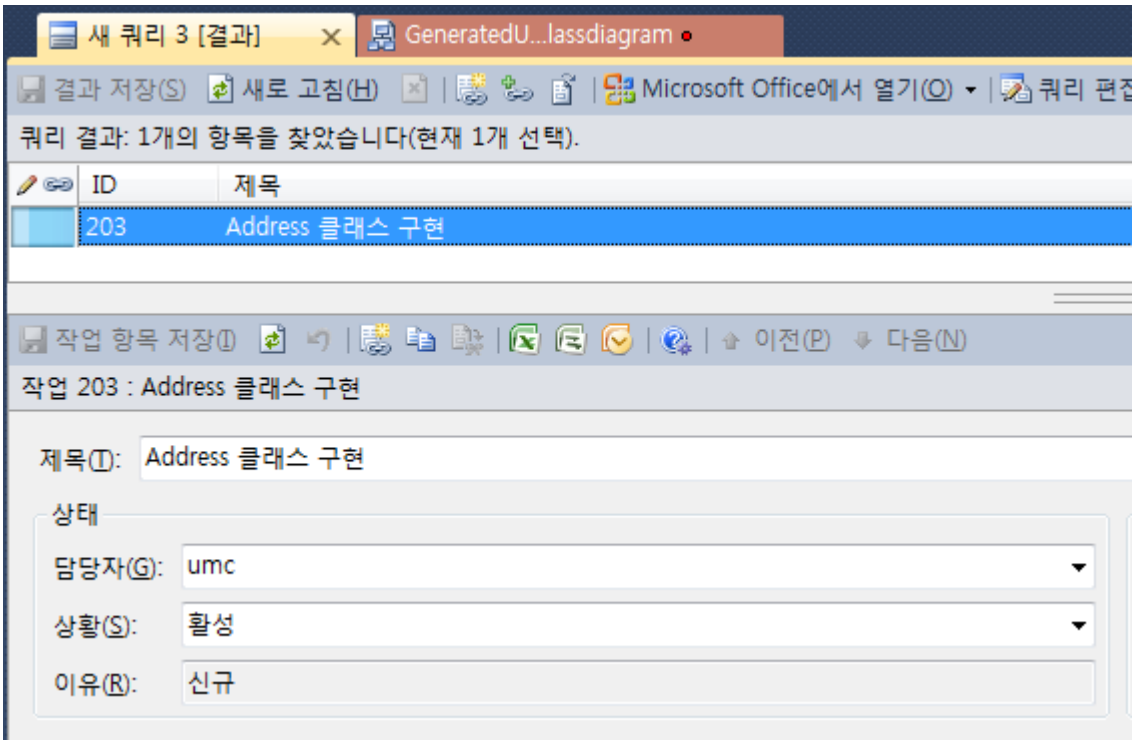
6.5.1.2. 다이어그램 요소에 연결된 작업 항목을 보려면?

요소에 작업 항목이 연결이 되면 요소의 오른쪽 상단에 작업 항목이 연결되었음을 의미하는 아이콘이 표시됩니다. 이 아이콘이 표시된 다이어그램의 요소는 어떤 작업 항목이 연결되었는지 확인할 수 있습니다.

1. 다이어그램 요소와 작업 항목이 연결된 요소에 마우스 오른쪽 버튼을 클릭하여 작업 항목 보기를 선택합니다.



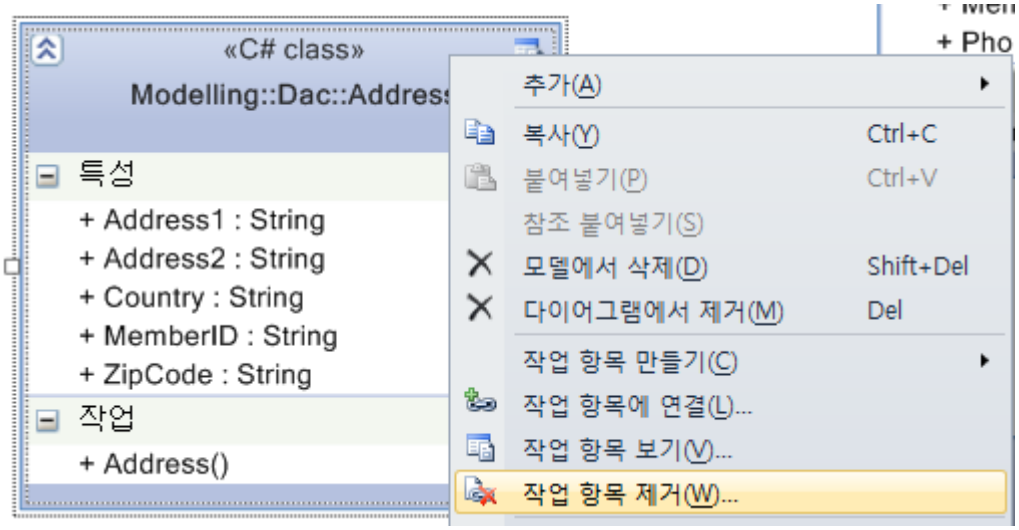
2. 다이어그램 요소와 연결된 작업 항목 목록이 나타납니다.



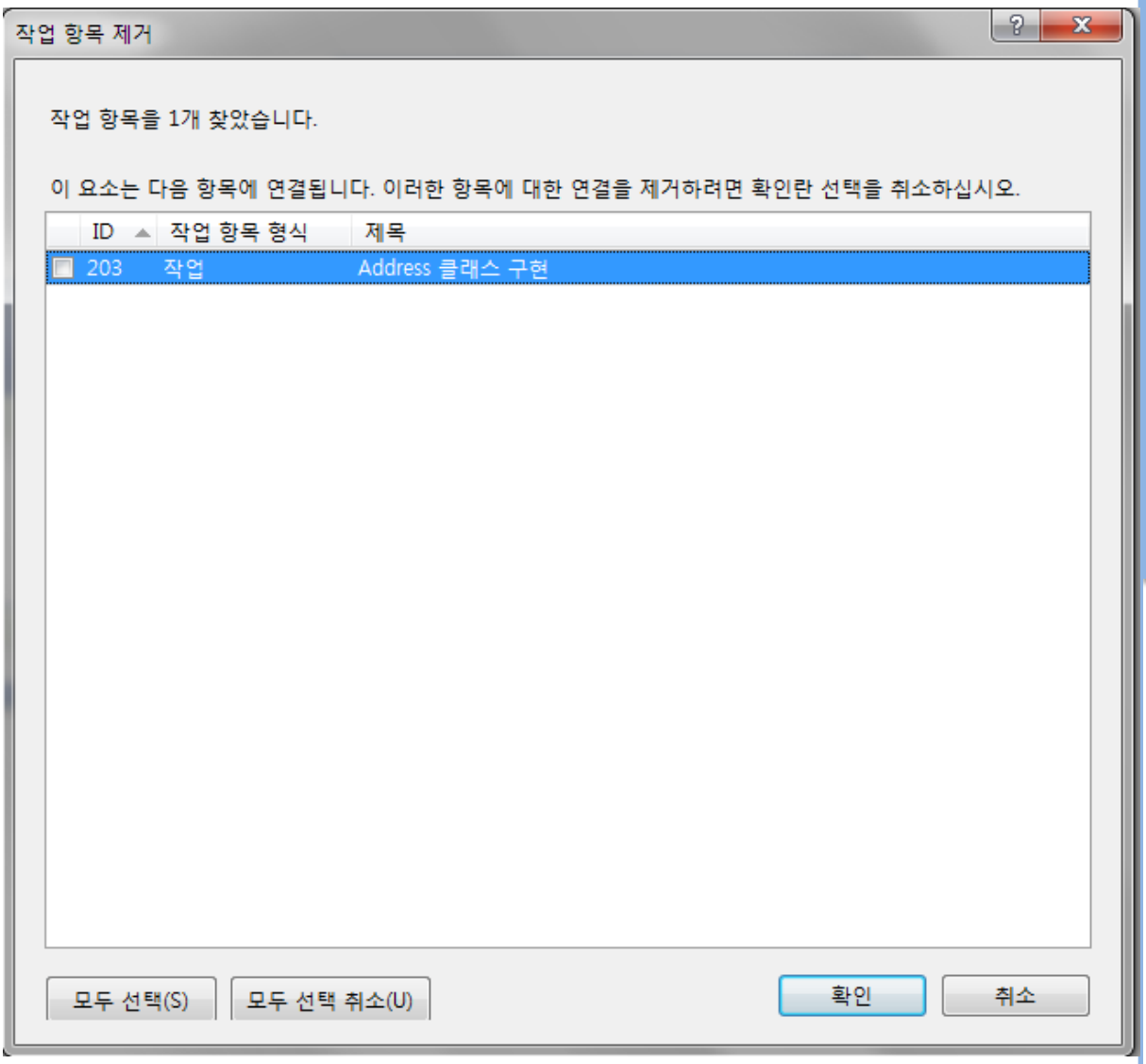
6.5.1.3. 다이어그램의 요소와 연결된 작업 항목을 제거하려면?

다이어그램의 요소와 연결된 작업 항목을 삭제하기 위해서 다음의 순서로 진행하십시오.

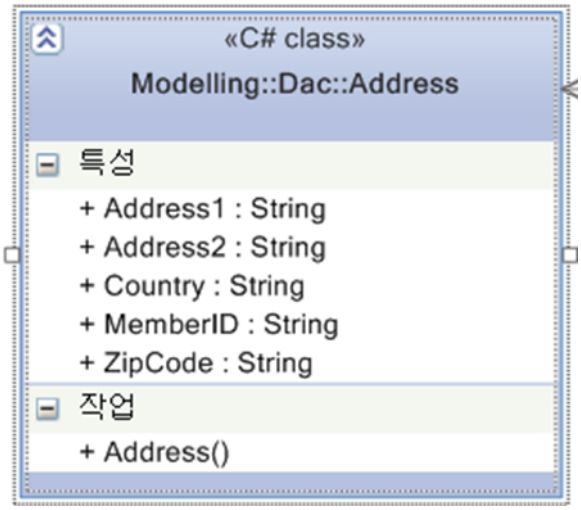
1. 다이어그램 요소와 작업 항목이 연결된 요소에 마우스 오른쪽 버튼을 클릭하여 작업 항목 제거를 선택합니다.



2. 작업 항목 제거 대화 상자에서 연결을 해제할 작업 항목의 체크 박스를 해지합니다. 체크 박스 해지 작업이 완료되면 확인을 클릭합니다.



3. 다이어그램의 요소가 작업 항목에서 제거되어 작업 항목 연결을 의미하는 아이콘이 삭제된 것을 확인합니다.



6.6. VISUAL STUDIO 2010 LAYER DIAGRAM EXTENSION

Visual Studio 2010 Modeling Features Pack 은 Layer Diagrams 의 유효성을 검사하거나 아키텍처 유효성을 검증하는 기능이 포함되어 있습니다. 이 Layer Diagrams 의 유효성 검증과 명령 등을 확장하기 위해서 추가적인 Visual Studio 2010 SDK 가 필요합니다.

이 단원은 [Visual Studio 2010 Modeling SDK] 단원을 참고 하십시오.

7. VISUAL STUDIO 2010 MODELING SDK

Visual Studio 2010 SDK(Software Development Kit)은 기존의 Visual Studio 2010 의 기능을 확장할 수 있는 도구를 지원해 주고 있습니다. 이 내용에 대해 자세한 내용은 필자의 블로그의 다음의 글을 참고 하십시오.

참고
[VSX] 1. Visual Studio Extensibility,, 그 시작
http://blog.powerumc.kr/232

7.1. 개발 환경 및 설치 요구 사항

Visual Studio 2010 모델링 프로젝트를 확장하기 위해 먼저 Visual Studio 2010 SDK 를 설치해야 합니다. Visual Studio 2010 SDK 는 아래의 다운로드 주소를 통해 다운로드 받으실 수 있습니다.

Visual Studio 2010 SDK 다운로드
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=47305cf4-2bea-43c0-91cd-1b853602dcc5&displaylang=en

시스템 요구 사항

- **Supported Operating Systems:** Windows 7;Windows Server 2003 R2 (32-Bit x86);Windows Server 2003 R2 x64 editions;Windows Server 2003 Service Pack 2;Windows Server 2008 R2;Windows Server 2008 Service Pack 2;Windows Vista Service Pack 2;Windows XP Service Pack 3
  - Windows XP (x86) with Service Pack 3 - all editions except Starter Edition
  - Windows Vista (x86 & x64) with Service Pack 2 - all editions except Starter Edition
  - Windows 7 (x86 and x64)
  - Windows Server 2003 (x86 & x64) with Service Pack 2 - all editions
  - Users will need to install MSXML6 if not already present
  - Windows Server 2003 R2 (x86 and x64) - all editions
  - Windows Server 2008 (x86 and x64) with Service Pack 2 - all editions
  - Windows Server 2008 R2 (x64) - all editions

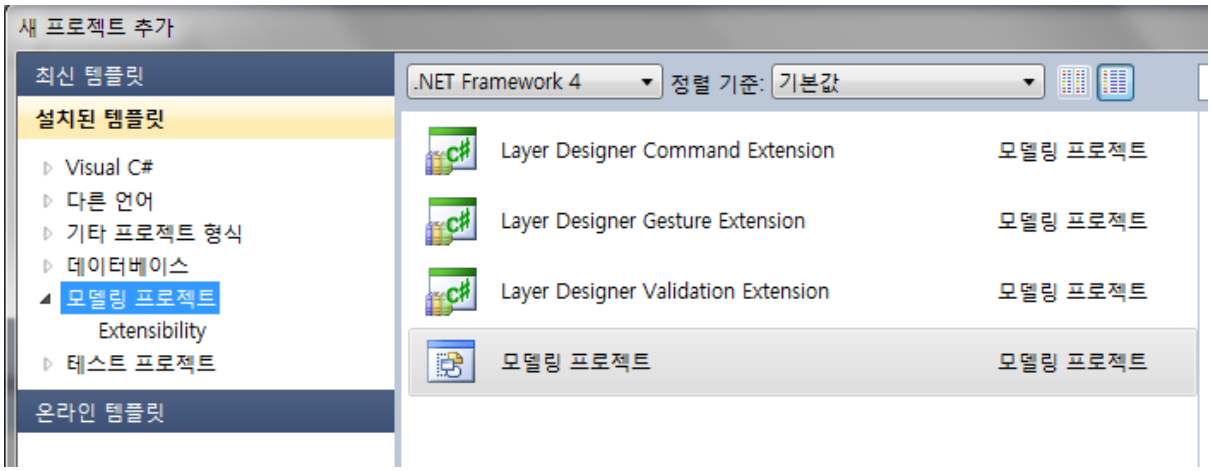
Visual Studio 2010 요구 사항

- Visual Studio 2010 Professional or better

7.2. LAYER DIAGRAMS 확장하기

Visual Studio 2010 SDK 와 Visual Studio 2010 Visualization & Modeling Features Pack 을 설치하면 Layer Diagrams 을 확장할 수 있는 프로젝트 템플릿이 설치가 됩니다.

새 프로젝트 만들기에서 모델링 프로젝트->Extensibility 를 선택하면 세 가지의 새로운 템플릿이 생성이 된 것을 확인할 수 있습니다.



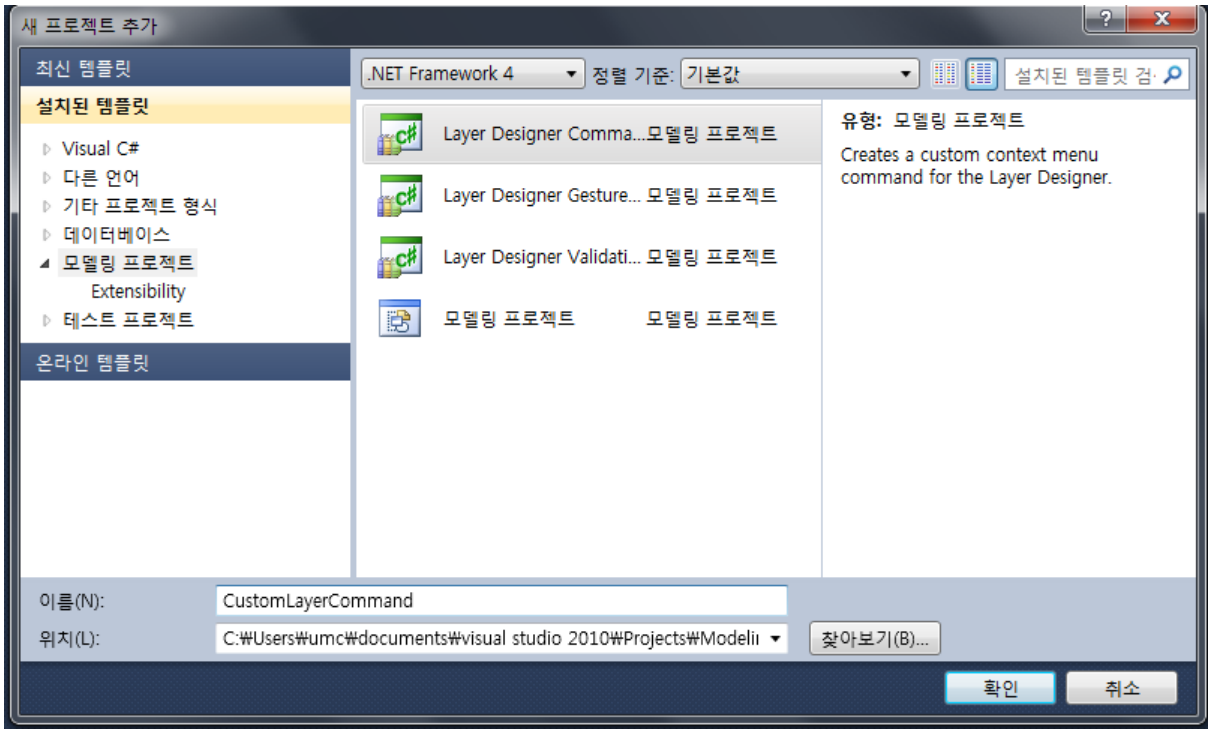
각 템플릿은 Layer Diagrams 을 확장하기 위해 아래와 같은 기능을 수행하는 템플릿입니다.

- Command Extension
  Layer Diagrams 에 특정 컨텍스트 메뉴를 통해 명령을 수행할 수 있는 확장 기능을 만들 수 있습니다.
- Gesture Extension
  Layer Diagrams 으로 끌어오기 동작 등의 확장 기능을 만들 수 있습니다.
- Validation Extension
  Layer Diagrams 과 코드의 구조가 유효한지 검사하는 확장 기능을 만들 수 있습니다.

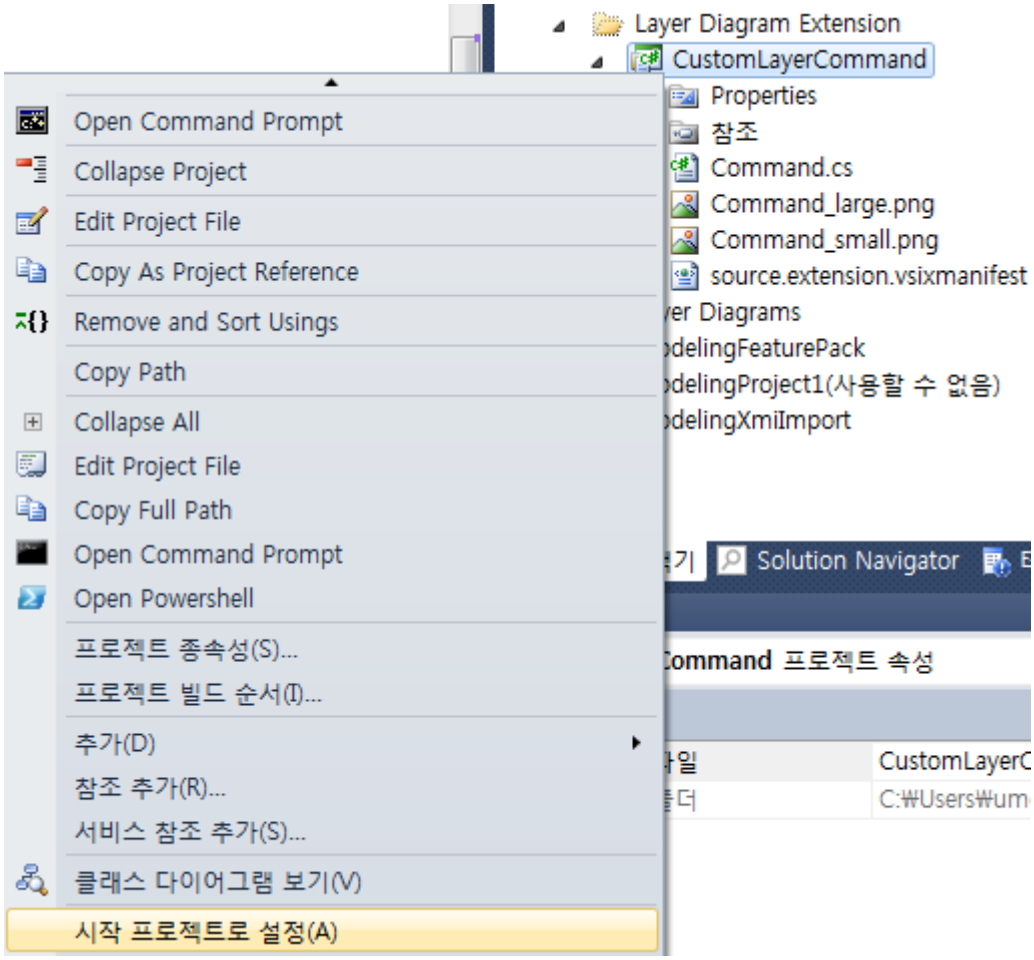
7.2.1. 레이어 개수를 세는 COMMAND EXTENSION 만들기

7.2.1.1. LAYER DIAGRAMS COMMAND EXTENSION 프로젝트 생성

우선 간단한 Command Extension 을 만들기 위해 Command Extension 템플릿으로 프로젝트를 생성합니다. 프로젝트의 이름을 CustomLayerCommand (또는 희망하는 이름으로) 프로젝트를 생성합니다.



간단히 생성된 프로젝트를 실행 또는 디버깅 하기 위해서 생성된 Command Extension 프로젝트를 선택하여 마우스 오른쪽 버튼을 클릭하여 시작 프로젝트로 설정합니다.



그런 다음 F5 키를 눌러 디버깅을 시작합니다. F5 키를 눌러 디버깅을 시작하면 Visual Studio 2010 은 별도의 레지스트리를 사용하는 Experimental 모드로 실행을 하게 되어 기존의 Visual Studio 2010 에 영향이 없도록 동작하게 됩니다.

7.2.1.2. LAYER DIAGRAMS 에 명령 코드 만들기

Visual Studio 2010 의 새로운 확장 기능은 .NET Framework 4.0 에 포함되는 MEF(Managed Extensibility Framework)를 사용하여 확장 기능을 만들 수 있습니다. 모델링 확장 기능 또한 MEF 를 사용하여 확장할 수 있으며 MEF 에 대한 자세한 내용은 필자의 블로그를 참고 하시기 바랍니다.

참고
MEF(Managed Framework Extensibility) 연제
http://blog.powerumc.kr/189

우선 클래스의 선언은 다음과 같이 되어 있습니다 아래와 같이 ExportAttribute 을 사용하여 ICommandExtension 형식을 Visual Studio 2010 모델링에서 인식할 수 있도록 컴포넌트 계약을 합니다. 그리고 LayerDesignerExtensionAttriburte 을 추가하면 Layer Diagrams 에서 ICommandExtension 확장 기능을 사용할 수 있도록 합니다.

```
[Export(typeof(ICommandExtension))]
[LayerDesignerExtension]
public partial class CustomLayerCommandExtension : ICommandExtension
```

클래스 내부에 Import 된 IDiagramContext 는 현재 다이어그램의 컨텍스트 개체로 이 개체를 이용하여 특정 명령을 내릴 때 다양한 작업을 수행할 수 있습니다.

```
[Export(typeof(ICommandExtension))]
[LayerDesignerExtension]
public partial class CustomLayerCommandExtension : ICommandExtension
{
        [Import(typeof(IDiagramContext))]
        public IDiagramContext DiagramContext { get; set; }
```

ICommandExtension 의 인터페이스는 Text 속성을 구현해야 하는데 이 속성은 다이어그램 컨텍스트 메뉴에서 표시되는 메뉴 이름입니다.

```
    public string Text
    {
        get
        {
         return "레이어 개수";
        }
    }
```

ICommandExtension 의 인터페이스는 QueryStatus 를 구현해야 하는데, 현재 상태에서 컨텍스트에 표시될 수 있는지 여부를 나타내게 됩니다. 조건에 따라 다른 표시/보이기 등의 동작이 필요하다면 아래의 코드를 수정하시면 됩니다.

```
    public void QueryStatus(IMenuCommand command)
        {
                command.Visible = true;
                command.Enabled = true;
        }
```

마지막으로 ICommandExtension 의 Execute 메서드를 구현합니다. 이 메서드는 Layer Diagrams 의 레이어 개수를 계산하기 위해 재귀 호출을 통해서 레이어 요소의 개수를 카운트합니다.

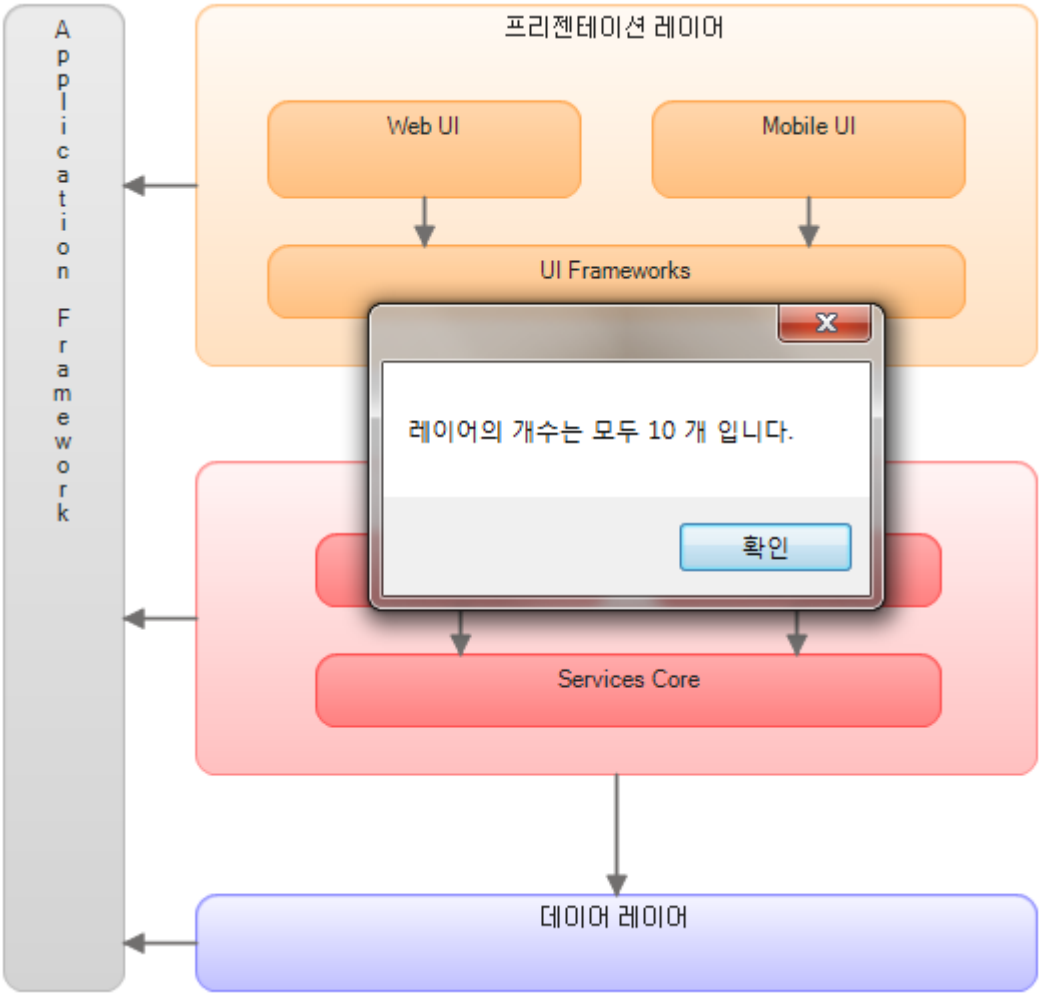
```
public void Execute(IMenuCommand command)
{
    int layerCount = 0;
    this.VisitDiagramElement(this.DiagramContext.CurrentDiagram.Diagram.
    ChildShapes, ref layerCount);

    MessageBox.Show(String.Format("레이어의 개수는 모두 {0} 개 입니다.", lay
    erCount));
}

private void VisitDiagramElement(IEnumerable<IShape> shapes, ref int la
    yerCount)
{
    foreach (var shape in shapes)
    {
     if (shape != null && shape.GetLayerElement() is ILayer)
     {
            layerCount++;

            if( shape.ChildShapes.Count() > 1)

     this.VisitDiagramElement(shape.ChildShapes.Skip(1), ref layerCount);
     }
    }
}
```

7.2.1.3. LAYER DIAGRAMS 코드 실행 결과

먼저 작성된 코드를 실행하기 위해 Visual Studio 2010 에서 Command Extension 을 시작 프로젝트로 설정한 후에 F5 키를 눌러 Visual Studio 2010 Experimental 모드로 실행을 할 수 있습니다. 만약 디버깅이 필요하지 않다면 Ctrl+F5 키를 눌러 디버깅 없이 실행하여 실행 속도를 높일 수 있습니다.

위의 코드는 다음과 같이 Layer Diagrams 에서 마우스 오른쪽 버튼을 눌러서 활성화할 수 있습니다. 다음과 같이 Text 속성의 반환 값이 메뉴의 이름이 되는 것을 확인할 수 있습니다.

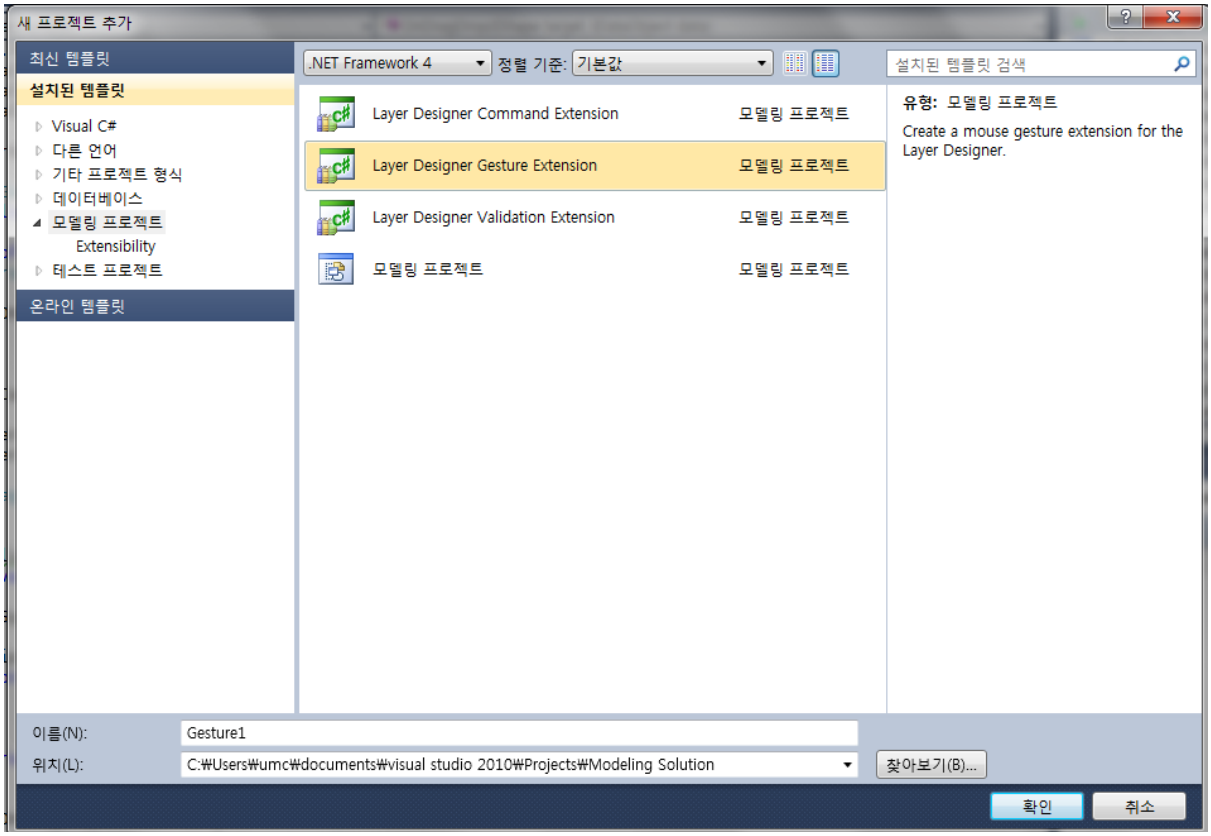
해당 메뉴를 클릭하면 Execute 메서드가 실행이 되고 메시지 상자에 Layer Diagrams 에 포함된 모든 레이어의 개수를 표시해 줍니다.



7.2.2. 폴더 구조를 DRAG&DROP 하는 GESTURES EXTENSION 만들기

7.2.2.1. GEATURES EXTENSION 프로젝트 만들기

새로운 프로젝트를 생성하는 메뉴에서 모델링 프로젝트->Extensibility 항목에서 Layer Designer Gesture Extension 을 선택하고 확인을 클릭합니다.



이 후 설정은 Layer Diagrams Command Extension 프로젝트 생성 항목을 참고하십시오.

7.2.2.2. 폴더를 DRAG&DROP 하는 코드 만들기

Gesture Extension 은 IGestureExtension 인터페이스를 구현합니다. 이 인터페이스를 상속하는 클래스를 만든 후에 ExportAttribute 을 통해 Visual Studio 2010 의 모델링 Geature Extension 과 계약 관계를 형성합니다.

```
[Export(typeof(IGestureExtension)), LayerDesignerExtension]
public partial class CustomLayerGestureExtension : IGestureExtension
```

폴더를 Drag&Drop 하는 코드 전체는 다음과 같습니다.

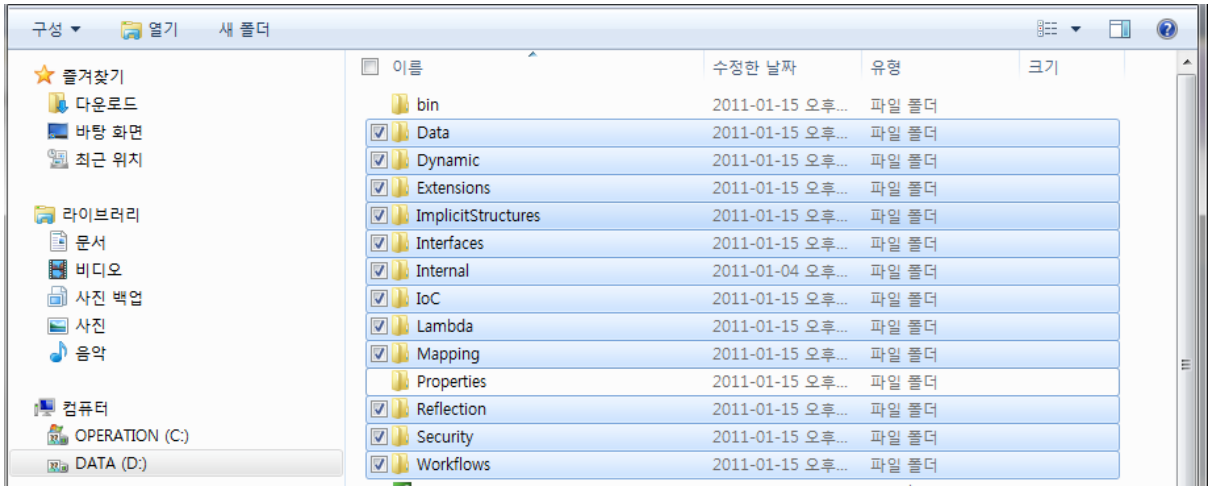
```
[Export(typeof(IGestureExtension)), LayerDesignerExtension]
    public partial class CustomLayerGestureExtension : IGestureExtension
    {
     [Import(typeof(IDiagramContext))]
     public IDiagramContext DiagramContext { get; set; }

     public void OnDoubleClick(IShape target)
```

```
        {
        }

        public bool CanDragDrop(IShape target, IDataObject data)
        {

        var folders = data.GetData(DataFormats.FileDrop) as String[];
                if (folders == null)
                {
                        MessageBox.Show("가져올 데이터가 없습니다.");
                }

                var isValidate = true;
                foreach(var folder in folders)
                {
                        if( System.IO.Directory.Exists(folder) == false )
                        {
                                isValidate = false;
                                break;
                        }
                }

                return true;
        }

        public void OnDragDrop(IShape target, IDataObject data)
        {

        var folders = data.GetData(DataFormats.FileDrop) as String[];

                foreach (var folder in folders)
                {
                        var dirInfo = new System.IO.DirectoryInfo(folder);

        target.Diagram.GetLayerModel().CreateLayer(dirInfo.Name);
                }
        }
       }
```

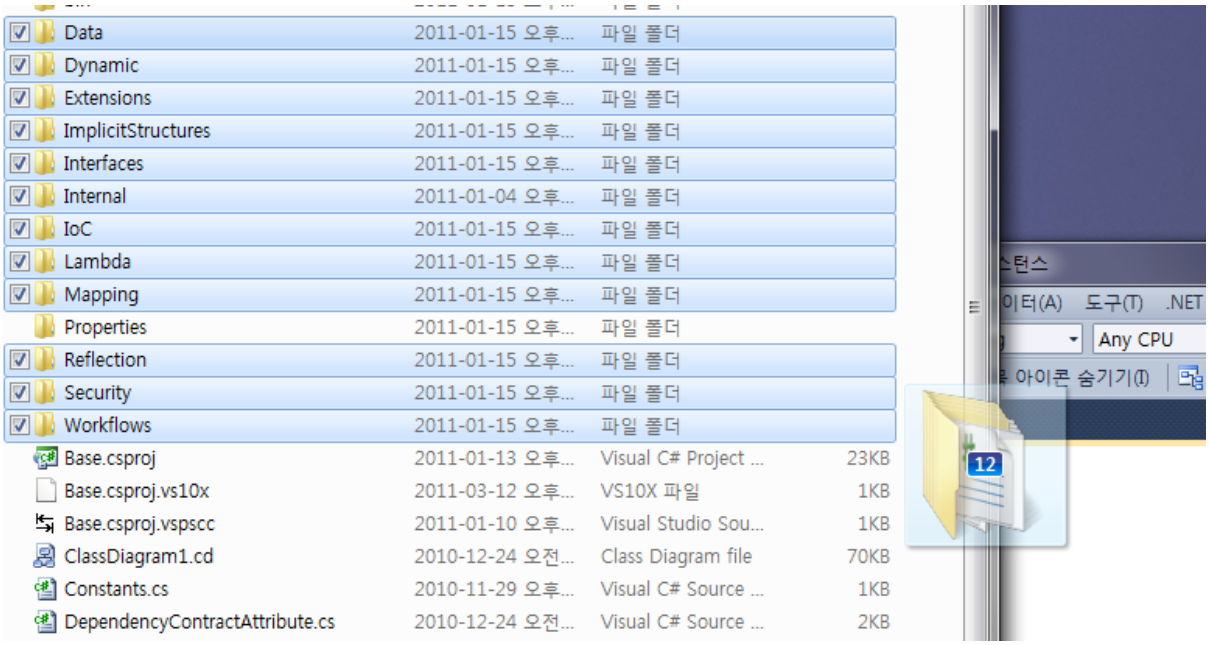
7.2.2.3. 폴더를 DRAG&DROP 하는 GESTURE EXTENSION 코드 실행 결과

먼저 테스트로 진행할 모델링 프로젝트를 생성한 후에 Layer Diagrams 항목을 추가 합니다.

그리고 윈도우 탐색기를 열어 드래그할 폴더를 다음과 같이 선택합니다.



선택할 폴더 항목을 Visual Studio 2010 의 다이어그램 디자이너에 Drag 하여 Drop 합니다.



Drop 결과 해당하는 폴더의 이름대로 Layer Diagrams 의 레이어 요소가 생성이 되는 결과를 확인할 수 있습니다.


LayerDiagram2.layerdiagram*
IoC
Data
Internal
Mapping
Lambda
Reflection
Security
Workflows

8. 결론

이 백서에서 지금까지 Visual Studio 2010 을 활용하여 어플리케이션을 모델링하고 그리고 모델링을 확장하여 잘 이용할 수 있는 여러 가지 기능과 방법을 설명하였습니다. 혹여 이 백서를 읽는 독자가 회사에서 UML 또는 모델링을 하는 업무나 직책을 갖지 않더라도 분명히 모델링은 개발자 사이에도 어느 정도 필요로 합니다.

소위 개발자들은 "코드로 말한다" 라고 강조하기도 합니다. 하지만 코드로 말하기 전에 자신의 의도와 노력을 시각적으로 표현하기 위해 노력하는 방법도 매우 중요합니다. 그것은 코드가 완성이 되기 전에 서로 간의 커뮤니케이션을 할 수 있으므로 이해 관계가 복잡해 질 때 먼저 자신의 의도와 노력을 상대에게 이해시킬 수 있는 좋은 방법이 바로 모델링이기 때문입니다.

이러한 모델링 습관은 매우 좋은 현상입니다. 예전에 값 비싼 도구로 모델링을 할 수 있었지만 이제는 Visual Studio 2010 으로 시스템의 설계나 코드의 설계, 그리고 기존 코드를 시각화 함으로써 서로 간에 시스템이나 어플리케이션의 설계 또는 코드를 쉽게 이해하고 접근하는 좋은 방법이기도 합니다.

더불어 Visual Studio 2010 의 모델링은 지속적으로 발전하고 있고 다양한 모델링 확장 기능을 Visual Studio Gallery 사이트에서 제공하고 있습니다. 모델링은 통합 개발 도구와 통합하여 보다 기존보다 좀더 생산성 있고 가치 있는 활동을 하나의 도구에서 모두 할 수 있으며, 이러한 여러분의 노력이 한 걸음 더 뻗어 나갈 수 있는 좋은 스킬이 되리라 필자는 확신을 합니다.

- 저자 소개

현재 NCsoft 에 재직 중이며, 2007 년도부터 Microsoft ASP.NET MVP 를 거쳐, 현재는 Microsoft ALM MVP 로 활동하고 있다. 개인 블로그(http://blog.powerumc.kr) 와 트위터(@powerumc) 를 통해 .NET 플랫폼 기술을 전파하고 있으며, 더불어 Visual Studio Korea 공식 팀 블로그(http://vsts2010.net) 과 팀 트위터 @vsts2010 을 운영하고 있다.

- MSDN 출판
  - MSDN 공개 및 출판 : Visual Studio 2010 을 활용한 ALM(Application Lifecycle Management)
  - MSDN 공개 : Team Foundation Server 2010 설치 가이드 (단일 서버)
  - MSDN 공개 : Team Foundation Server 2010 설치 가이드 (다중 서버)
  - MSDN 공개 : Team Foundation Server 2010 설치 가이드 (Lab 구성)
  - MSDN 공개 : Team Foundation Server 2010 활용 가이드 (FQDN 설정)
  - MSDN 공개 : Visual Source Safe 사용자를 위한 TFS2010 시리즈 1 : WORKGROUP 설치가이드
  - MSDN 공개 : Visual Source Safe 사용자를 위한 TFS2010 시리즈 2 : 활용가이드
  - MSDN 공개 : Visual Source Safe 사용자를 위한 TFS2010 시리즈 3 : 마이그레이션가이드

- 오픈 소스 활동
  - MEF Generic : http://MEFGeneric.codeplex.com (개발 및 관리)
  - vutpp for VS2010 : http://vutpp.codeplex.com (참여 및 관리)
  - UmcBlog : http://blog.powerumc.kr/30 (개발 및 관리)

- 무료 배포 소프트웨어
  - VSGesture for VS2010
    - Visual Studio Gallery : http://visualstudiogallery.msdn.microsoft.com/en-us/e03c91ff-e20d-4dcc-822b-172a68c40f5b
    - 개인 블로그 : http://blog.powerumc.kr/305
  - VSGesture for VS2008, VS2005 : http://blog.powerumc.kr/206
  - VSHelper for VS2008, VS2005 : http://blog.powerumc.kr/166
  - VSExplorer for VS2008, VS2005 : http://blog.powerumc.kr/171
  - VSCMD for VS2008, VS2005 : http://blog.powerumc.kr/178
  - VSComment for VS2008, VS2005 : http://blog.powerumc.kr/89