DESim++

Developer Guide

using Visual Studio 2010

Intelligent System Research Lab

By SH Shin

1. 라이브러리 구성

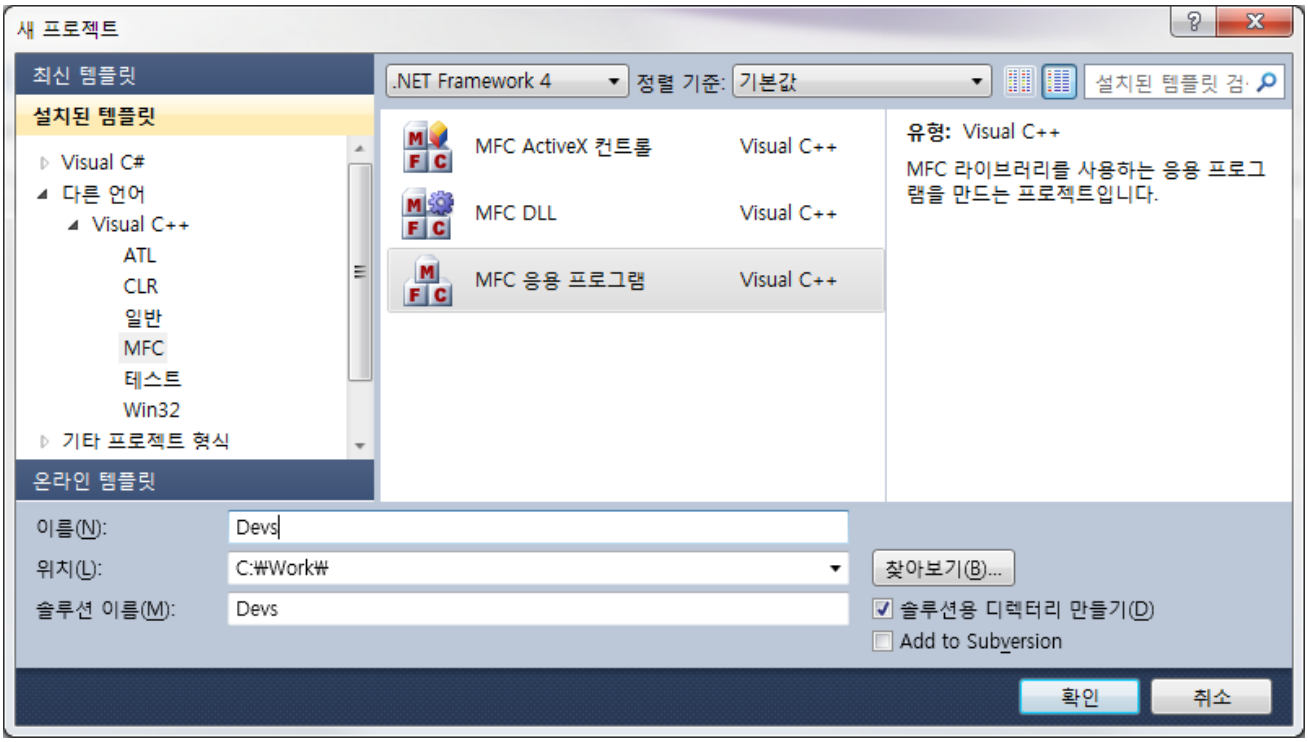
Atomic.h	Coupled.h
DEVSDEF.H	Digraph.h
Entities.h	Entstr.h
LIST.H	MESSAGE.H
MLIST.H	Models.h
Port.h	Tglobal.h
Atomic.cpp	Coupled.cpp
Digraph.cpp	Entities.cpp
Entstr.cpp	LIST.cpp
MakeSES.cpp	MESSAGE.cpp
MLIST.cpp	Models.cpp
Output.cpp	Port.cpp

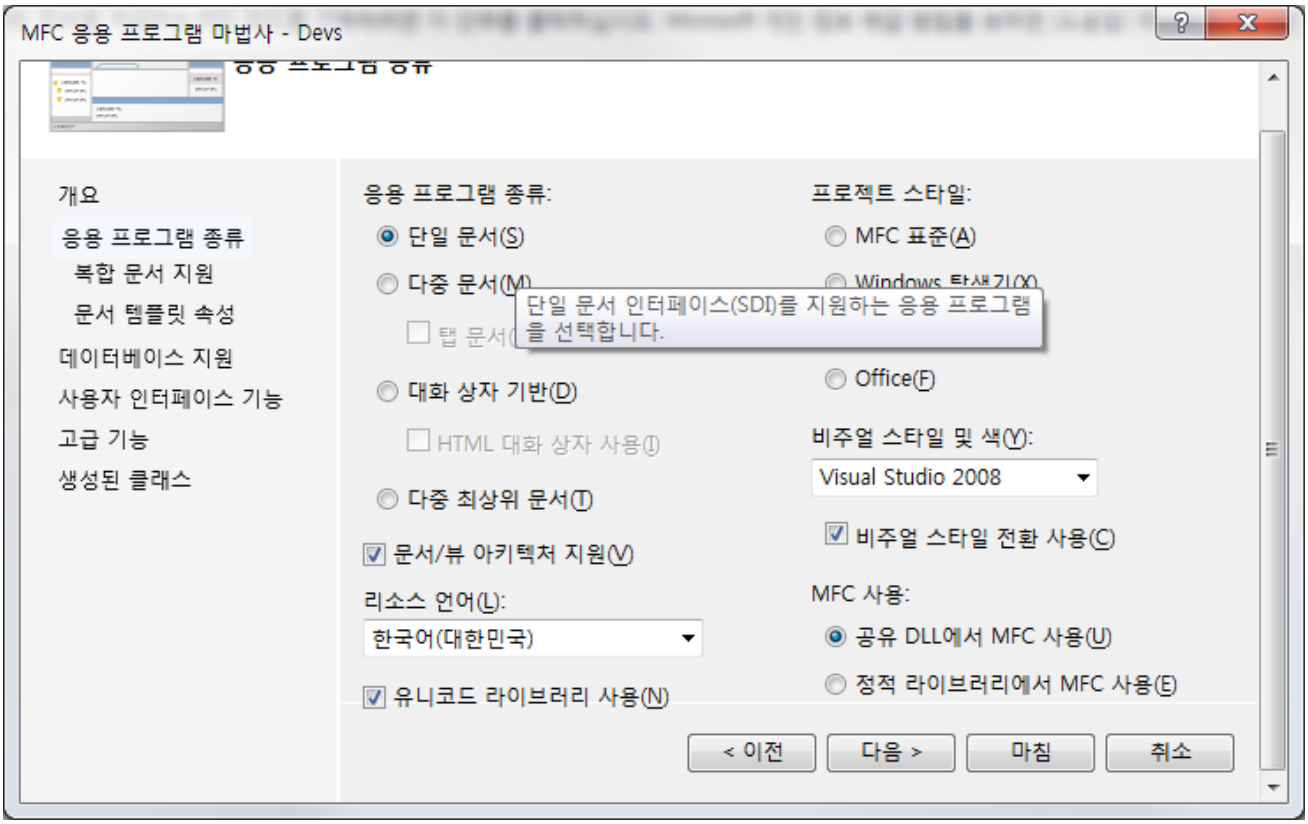
커널소스 - 프로젝트에 로드하고 수정X

genr.h	Process.h
Transd.h	genr.cpp
Process.cpp	Transd.cpp

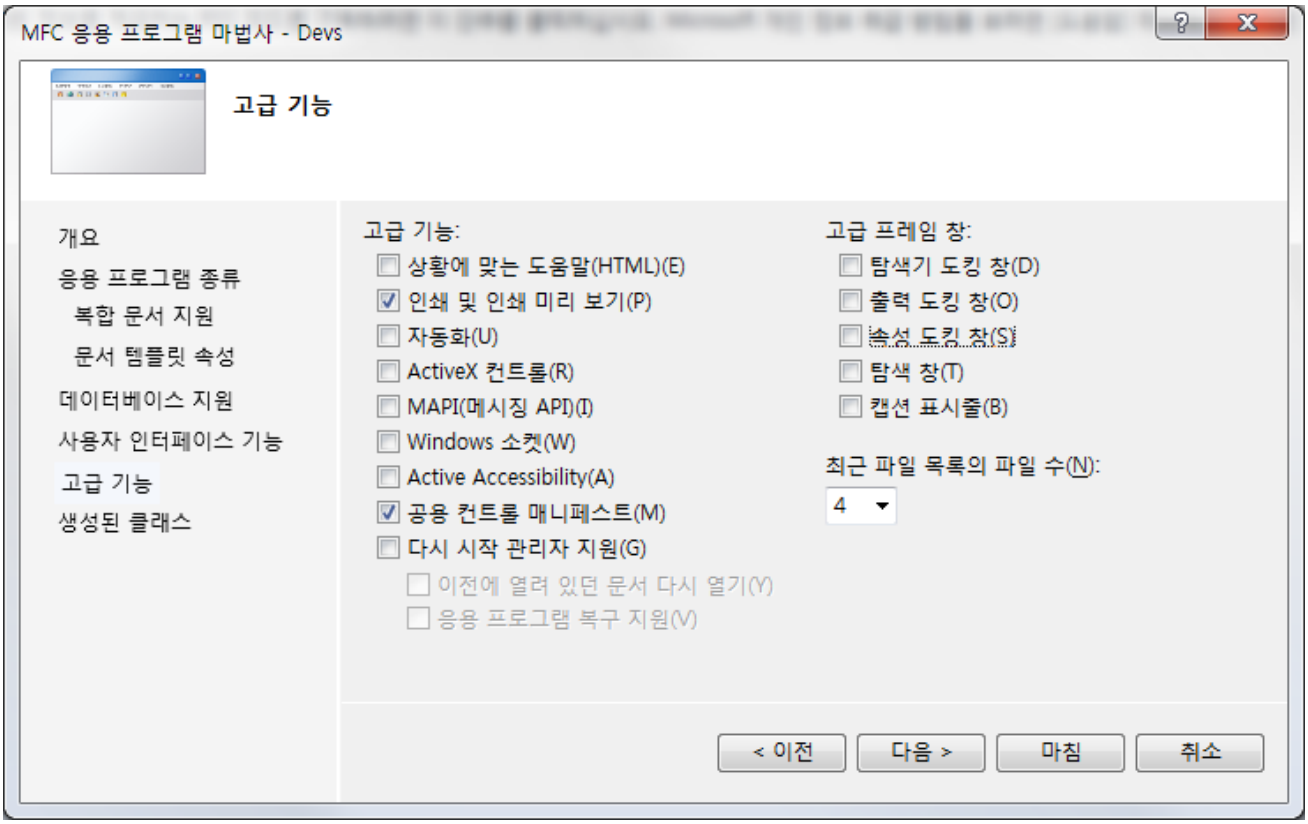
사용자 정의 모델 - 프로젝트에 추가하는 모델

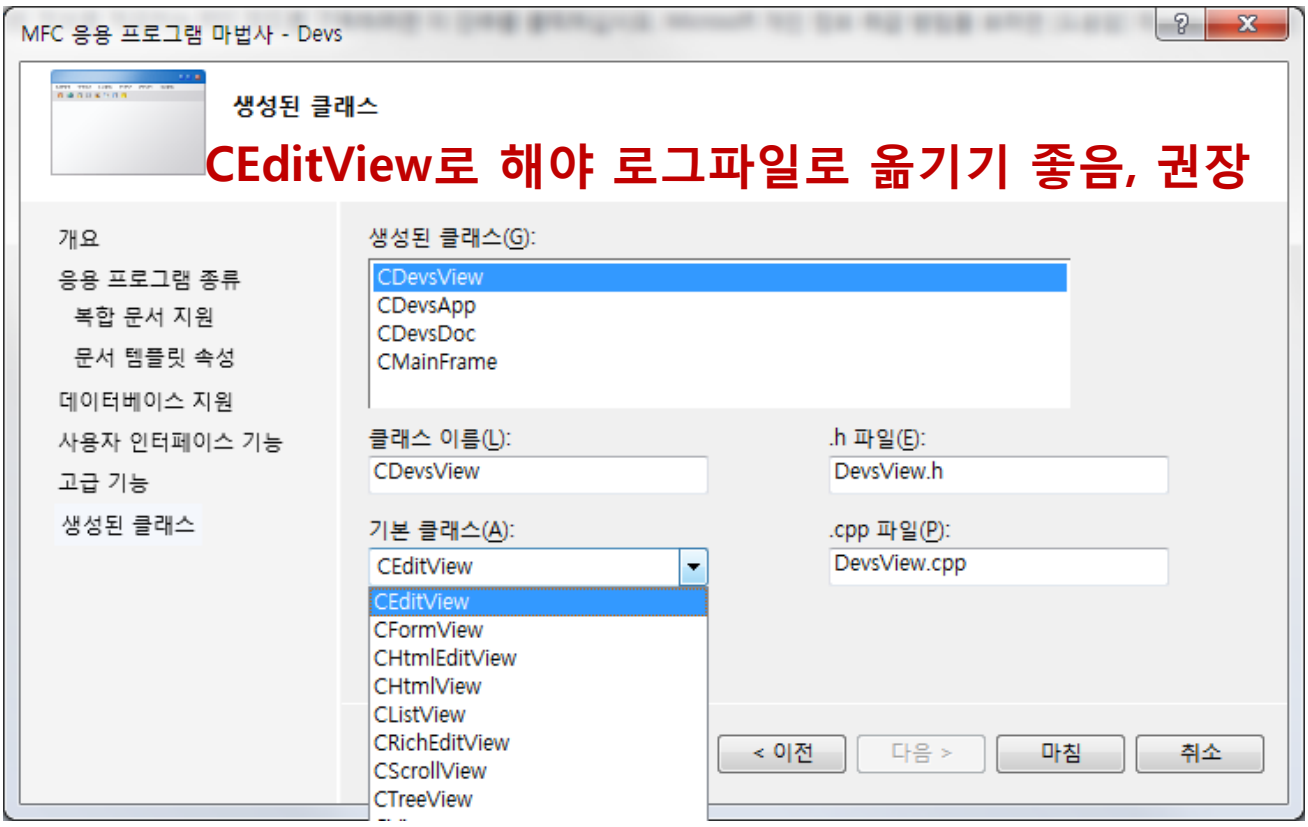
2. 프로젝트 만들기


새 프로젝트
최신 템플릿
설치된 템플릿
Visual C#
다른 언어
Visual C++
ATL
CLR
일반
MFC
테스트
Win32
기타 프로젝트 형식
온라인 템플릿
.NET Framework 4
정렬 기준:
기본값
설치된 템플릿 검:
MFC ActiveX 컨트롤
Visual C++
MFC DLL
Visual C++
MFC 응용 프로그램
Visual C++
유형: Visual C++
MFC 라이브러리를 사용하는 응용 프로그램을 만드는 프로젝트입니다.
이름(N):
Devs
위치(L):
C:₩Work₩
찾아보기(B)...
솔루션 이름(M):
Devs
솔루션용 디렉터리 만들기(D)
Add to Subversion
확인
취소


MFC 응용 프로그램 마법사 - Devs
개요
응용 프로그램 종류
복합 문서 지원
문서 템플릿 속성
데이터베이스 지원
사용자 인터페이스 기능
고급 기능
생성된 클래스
응용 프로그램 종류:
단일 문서(S)
다중 문서(M)
탭 문서(
단일 문서 인터페이스(SDI)를 지원하는 응용 프로그램을 선택합니다.
대화 상자 기반(D)
HTML 대화 상자 사용(I)
다중 최상위 문서(T)
문서/뷰 아키텍처 지원(V)
리소스 언어(L):
한국어(대한민국)
유니코드 라이브러리 사용(N)
프로젝트 스타일:
MFC 표준(A)
Office(F)
비주얼 스타일 및 색(Y):
Visual Studio 2008
비주얼 스타일 전환 사용(C)
MFC 사용:
공유 DLL에서 MFC 사용(U)
정적 라이브러리에서 MFC 사용(E)
< 이전
다음 >
마침
취소

2. 프로젝트 만들기


MFC 응용 프로그램 마법사 - Devs
고급 기능
개요
응용 프로그램 종류
복합 문서 지원
문서 템플릿 속성
데이터베이스 지원
사용자 인터페이스 기능
고급 기능
생성된 클래스
고급 기능:
상황에 맞는 도움말(HTML)(E)
인쇄 및 인쇄 미리 보기(P)
자동화(U)
ActiveX 컨트롤(R)
MAPI(메시징 API)(I)
Windows 소켓(W)
Active Accessibility(A)
공용 컨트롤 매니페스트(M)
다시 시작 관리자 지원(G)
이전에 열려 있던 문서 다시 열기(Y)
응용 프로그램 복구 지원(V)
고급 프레임 창:
탐색기 도킹 창(D)
출력 도킹 창(O)
속성 도킹 창(S)
탐색 창(T)
캡션 표시줄(B)
최근 파일 목록의 파일 수(N):
4
< 이전
다음 >
마침
취소


MFC 응용 프로그램 마법사 - Devs
생성된 클래스
CEditView로 해야 로그파일로 옮기기 좋음, 권장
개요
응용 프로그램 종류
복합 문서 지원
문서 템플릿 속성
데이터베이스 지원
사용자 인터페이스 기능
고급 기능
생성된 클래스
생성된 클래스(G):
CDevsView
CDevsApp
CDevsDoc
CMainFrame
클래스 이름(L):
CDevsView
.h 파일(E):
DevsView.h
기본 클래스(A):
CEditView
CEditView
CFormView
CHtmlEditView
CHtmlView
CListView
CRichEditView
CScrollView
CTreeView
.cpp 파일(P):
DevsView.cpp
< 이전
다음 >
마침
취소

2. 프로젝트 만들기

MFC 응용 프로그램 마법사 - Devs

고급 기능

- 개요
- 응용 프로그램 종류
  - 복합 문서 지원
  - 문서 템플릿 속성
- 데이터베이스 지원
- 사용자 인터페이스 기능
- 고급 기능
- 생성된 클래스

고급 기능:

- [ ] 상황에 맞는 도움말(HTML)(E)
- [x] 인쇄 및 인쇄 미리 보기(P)
- [ ] 자동화(U)
- [ ] ActiveX 컨트롤(R)
- [ ] MAPI(메시징 API)(I)
- [ ] Windows 소켓(W)
- [ ] Active Accessibility(A)
- [x] 공용 컨트롤 매니페스트(M)
- [ ] 다시 시작 관리자 지원(G)
  - [ ] 이전에 열려 있던 문서 다시 열기(Y)
  - [ ] 응용 프로그램 복구 지원(V)

고급 프레임 창:

- [ ] 탐색기 도킹 창(D)
- [ ] 출력 도킹 창(O)
- [ ] 속성 도킹 창(S)
- [ ] 탐색 창(T)
- [ ] 캡션 표시줄(B)

최근 파일 목록의 파일 수(N): 4

< 이전 | 다음 > | 마침 | 취소

MFC 응용 프로그램 마법사 - Devs

생성된 클래스

- 개요
- 응용 프로그램 종류
  - 복합 문서 지원
  - 문서 템플릿 속성
- 데이터베이스 지원
- 사용자 인터페이스 기능
- 고급 기능
- 생성된 클래스

생성된 클래스(G):

- CDevsView
- CDevsApp
- CDevsDoc
- CMainFrame

클래스 이름(L): CDevsView

.h 파일(E): DevsView.h

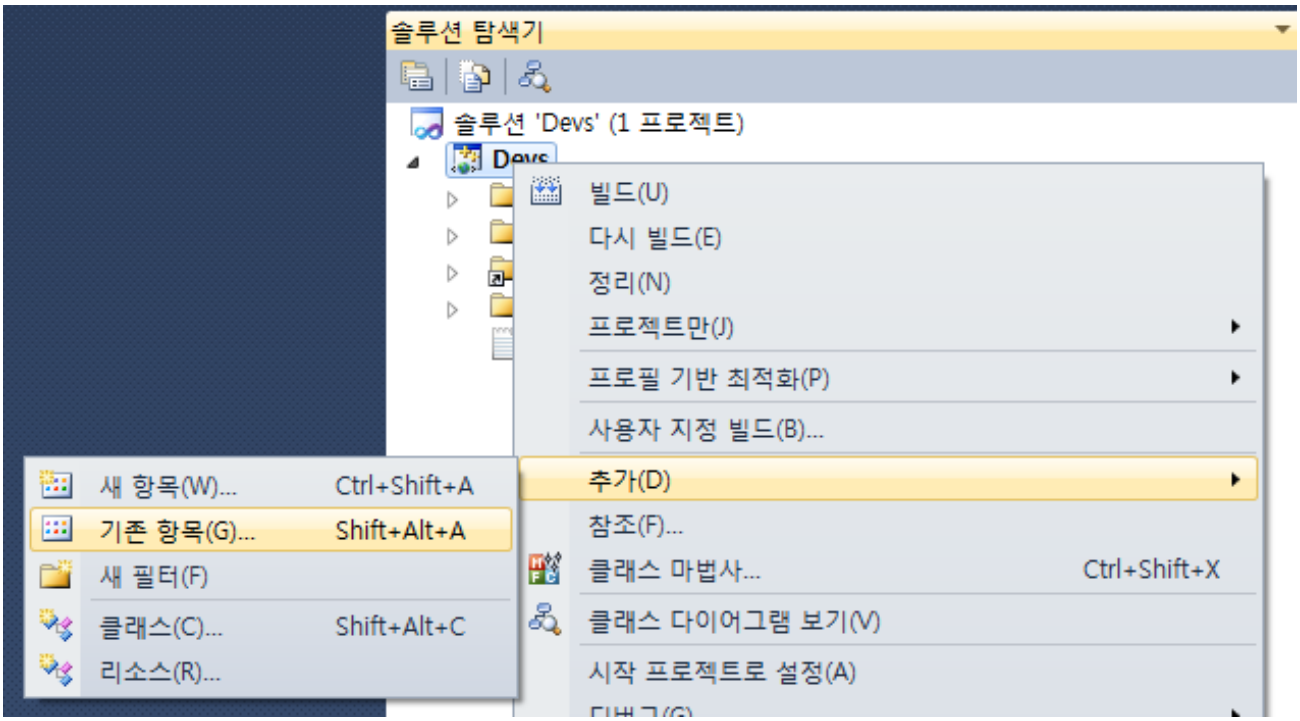
기본 클래스(A): CEditView

.cpp 파일(P): DevsView.cpp

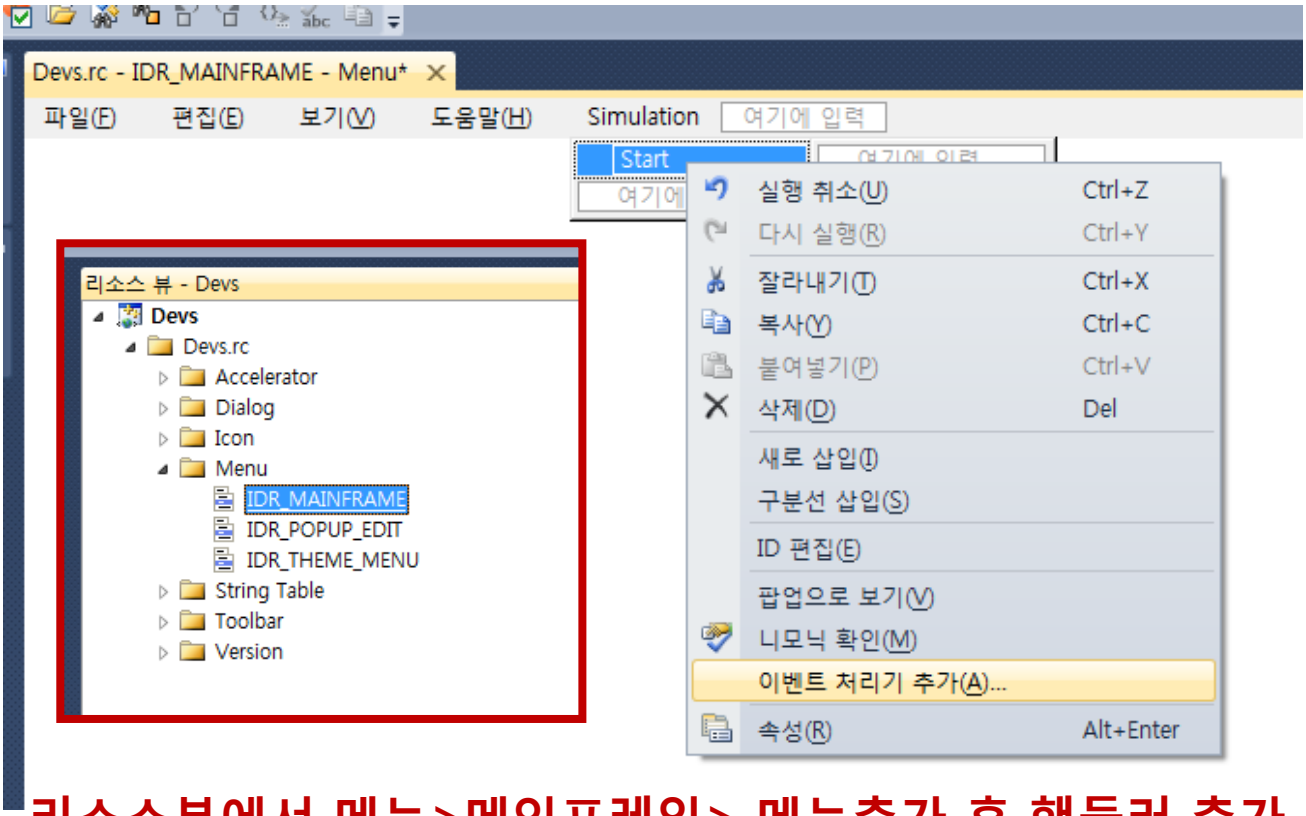
- CEditView
- CFormView
- CHtmlEditView
- CHtmlView
- CListView
- CRichEditView
- CScrollView
- CTreeView

< 이전 | 다음 > | 마침 | 취소

2. 프로젝트 만들기

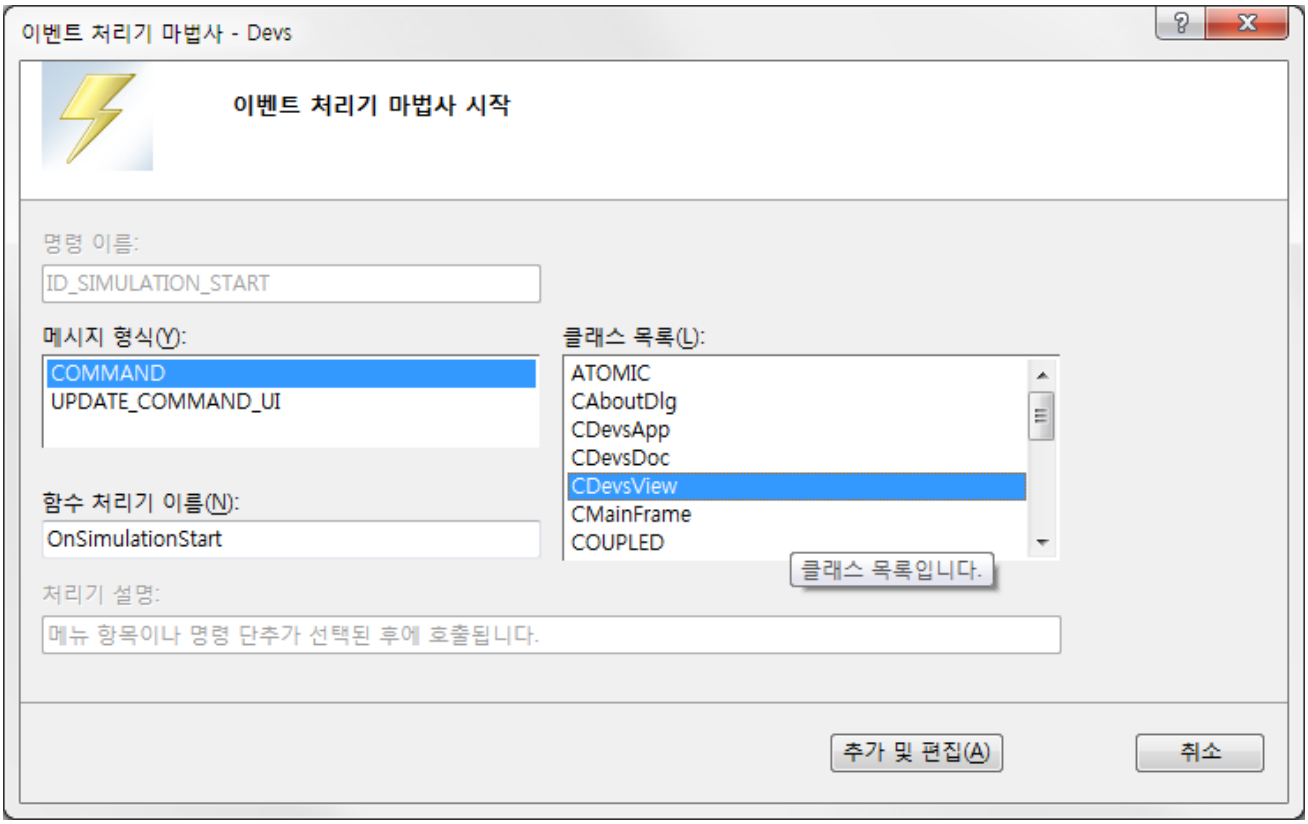


라이브러리 파일 모두 소스폴더에 붙여 넣기 후 추가 권장



리소스뷰에서 메뉴>메인프레임> 메뉴추가 후 핸들러 추가

2. 프로젝트 만들기

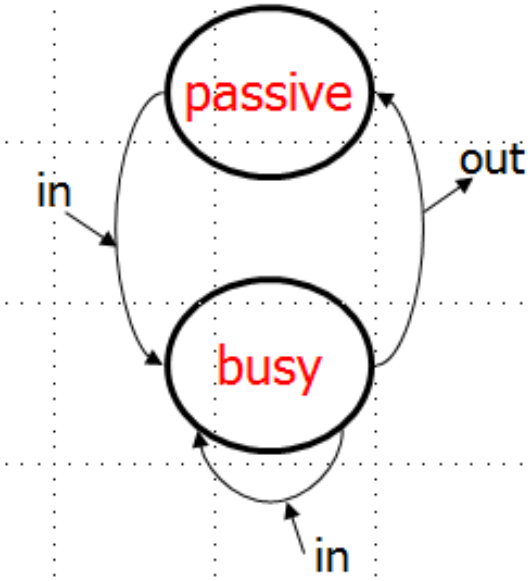



```
// CDevsView 메시지 처리기

#include "Tglobal.h"
void CDevsView::OnSimulationStart()
{
    // TODO: 여기에 명령 처리기 코드를 추가합니다.
    MakeSES();
    StartSimulation();
}
```

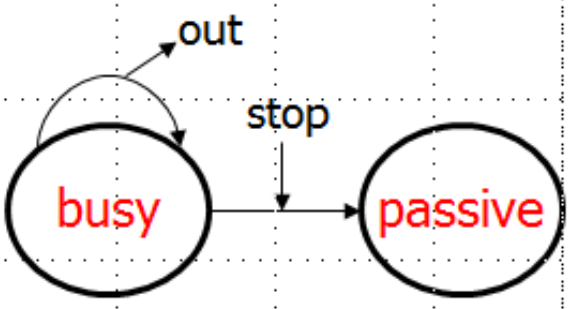
추가된 핸들러위에 tglobal.h 포함,
내용으로 MakeSES(); StartSimulation(); 추가

3. 사용자 모델 만들기 (예제 심플아키텍처)

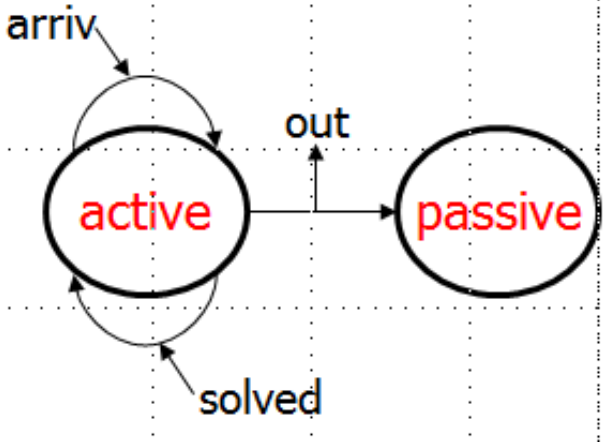
- **Processor**



- **Genr**



- **Transd**



3. 사용자 모델 만들기 (예제 심플아키텍처)

Process.h

(전역 범위)

```cpp
#pragma once
#include "stdafx.h"
#include "Tglobal.h"
#include "atomic.h"

class Process : public ATOMIC {
public:
    CString JobID;
    double PTime;
    CString Queue[50];
    int Front,Tail;
public:
    Process(CString);
    Process(char *);

    void ExtTransitionFN(double,DevsMessage);
    void IntTransitionFN(void);
    void OutputFN(void);
    void InitializeFN(void);
};
```

Process.cpp

Process | ExtTransitionFN(double Time, Devs

```cpp
#include "stdafx.h"
#include "process.h"


Process::Process(CString EName) : ATOMIC(EName) {
    SetName(EName);
}

Process::Process(char *EName) : ATOMIC(EName)  {
    SetName(EName);
}

void Process::ExtTransitionFN(double Time, DevsMessage MSG) {
    Display(Name); Display("(EXT) --> ");
    if (MSG.ContentPort() == "in") {

        // Put job into the Queue
        if(Tail < 50)
        {
            Queue[Tail++] = *((CString *)(MSG.ContentValue()));
            Display(MSG.ContentPort()); Display(":"); Display(JobID);
        }

        if (Phase == "busy"){
            Continue();
        }
        else
        {
            if(Front != Tail)
                HoldIn("busy",0.0);
        }
    }
    else Continue();
    NewLine();
}

void Process::IntTransitionFN(void) {
    Display(Name); Display("(INT) --> ");
    if (Phase == "busy"){
        // Get job from the Queue
        if(Front != Tail)
        {
            // processing
            JobID = Queue[Front++];
            Display(" process : "); Display(JobID);
            HoldIn("busy",PTime);
        }
        else
            Passivate();
    }
    else Continue();
    NewLine();
}

void Process::OutputFN(void) {
    Display(Name); Display("(OUT) --> ");

    if (Phase == "busy"){
        MakeContent("out",&JobID);
    }
    else MakeContent();
    NewLine();
}

void Process::InitializeFN(void){
    PTime = (double)7.0;
    Passivate();
    Front = Tail = 0;
}
```

Process.h/cpp

3. 사용자 모델 만들기 (예제 심플아키텍처)

genr.h

(전역 범위)

```cpp
#pragma once
#include "stdafx.h"
#include "Tglobal.h"
#include "atomic.h"

class GENR : public ATOMIC {
public:
    int     InterArrivalTime;
//  int     ProcessingTime;
//  int     ProblemLevel;
    int     Count;

public:
    GENR( );
    GENR(CString);
    GENR(char *);

    void ExtTransitionFN(double,DevsMessage);
    void IntTransitionFN(void);
    void OutputFN(void);
    void InitializeFN(void);
};
```

genr.cpp

(전역 범위)

```cpp
#include "stdafx.h"
#include "genr.h"

GENR::GENR( ) : ATOMIC( ) {
    SetName("GENR");
}

GENR::GENR(CString EName) : ATOMIC(EName) {
    SetName(EName);
}

GENR::GENR(char *EName) : ATOMIC(EName) {
    SetName(EName);
}

void GENR::ExtTransitionFN(double E, DevsMessage X){
    char GENRMessage[100]={0,};

    Display(Name); Display("(EXT) --> :");
    Display(X.ContentPort( )); Display(": ");
    sprintf(GENRMessage,"When: %lf",AddTime(GetLastEventTime( ),E));
    Display(GENRMessage);

    if (X.ContentPort( ) == "stop") Passivate( );
    NewLine( );
}

void GENR::IntTransitionFN(void){
    char GENRMessage[100]={0,};

    Display(Name); Display("(INT) --> ");
    sprintf(GENRMessage,"Sigma: %lf  When: %lf",Sigma, AddTime(GetLastEventTime( ),S
    Display(GENRMessage);
    if (Phase == "busy") { HoldIn("busy", InterArrivalTime); }
    else { Passivate( ); }
    NewLine( );
}

void GENR::OutputFN(void){
    CString O;
    char Num[10]={0,};
    char GENRMessage[100]={0,};

    Display(Name); Display("(OUT) --> ");
    sprintf(GENRMessage,"Phase: %s  Sigma: %lf  When: %lf",Phase,Sigma,GetNextEvent
    Display(GENRMessage); NewLine( );

    if (Phase == "busy"){
        sprintf(Num,"%d",Count++);
        O = "Job-";
        O += Num;
        MakeContent("out",&O);
    }
    else MakeContent( );
}

void GENR::InitializeFN(void){
    InterArrivalTime = 3;
    Count = 0;

    HoldIn("busy",0.0);
}
```

Gern.h/cpp

3. 사용자 모델 만들기 (예제 심플아키텍처)

Transd.h

(전역 범위)

```cpp
#pragma once
#include "stdafx.h"
#include "Tglobal.h"
#include "atomic.h"

typedef struct {
    CString ID;
    double  Time;
} JOB;

typedef struct {
    JOB Jobs[100];
    int Num;
} JOBS;

class Transd : public ATOMIC {
public:
    CString JobID;
    double  clock;
    JOBS    Arrive, Solve;

public:
    Transd(CString);
    Transd(char *);

    void ExtTransitionFN(double,DevsMessage);
    void IntTransitionFN(void);
    void OutputFN(void);
    void InitializeFN(void);

    void PrintArrive(void);
    void PrintSolve(void);
};
```

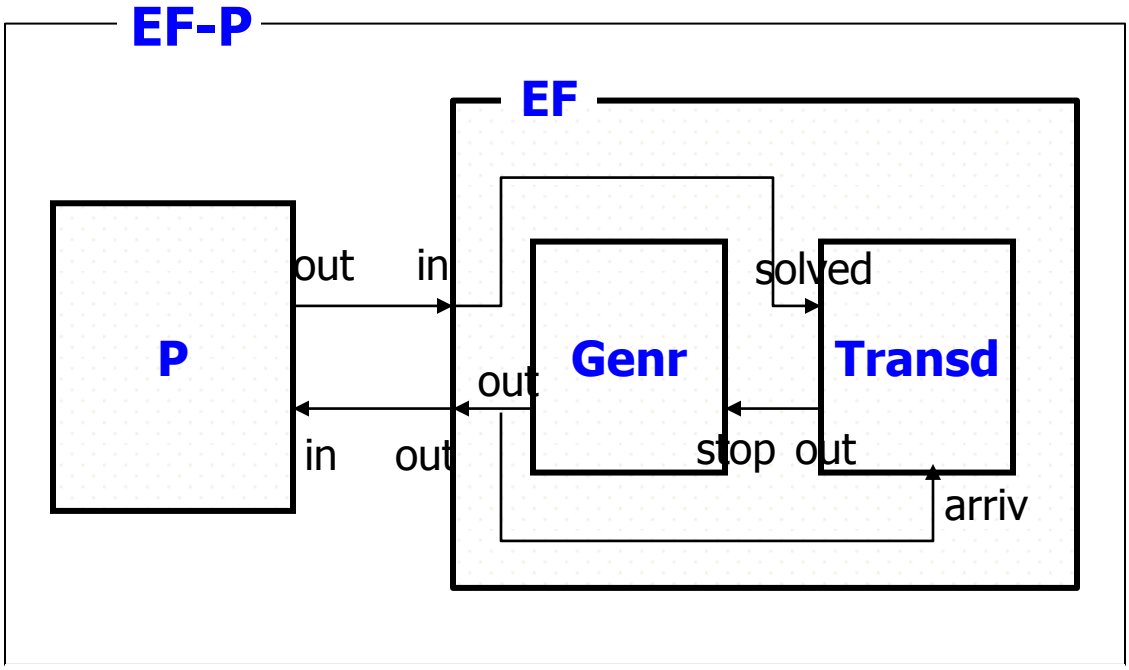
Transd.cpp

(전역 범위)

```cpp
#include "stdafx.h"
#include "transd.h"


Transd::Transd(CString EName) : ATOMIC(EName){
        SetName(Name);
}

Transd::Transd(char *EName) : ATOMIC(EName){
        SetName(Name);
}

void Transd::ExtTransitionFN(double E, DevsMessage X){
    JobID = *(CString *)X.ContentValue();
    clock += E;

    Display(Name); Display("(EXT) --> :");
    Display(X.ContentPort());
    Display(":"); Display(JobID);
    Display(" at "); Display(clock);
    NewLine();
    if (Phase == "active"){
        if (X.ContentPort() == "arriv")       {
            Arrive.Jobs[Arrive.Num].ID = JobID;
            Arrive.Jobs[Arrive.Num].Time = clock;
            Arrive.Num++;
        }
        else if (X.ContentPort() == "solved") {
            Solve.Jobs[Solve.Num].ID = JobID;
            Solve.Jobs[Solve.Num].Time = clock;
            Solve.Num++;
        }
    }
    Continue();
}

void Transd::IntTransitionFN(void) {
    Display(Name); Display("(INT) --> "); NewLine();

    if (Phase == "active") {
        PrintArrive();
        PrintSolve();
        Passivate();
    }
    else Continue();
}

void Transd::OutputFN(void) {
    Display(Name); Display("(OUT) --> "); NewLine();
    if (Phase == "active")
        MakeContent("out",NULL);
    else MakeContent();
}

void Transd::InitializeFN(void){
     clock = (double)0.0;

     Arrive.Num = 0;
     Solve.Num = 0;

     HoldIn("active",(double)100.0);
}


void Transd::PrintArrive(void){
    NewLine();
    Display("   ---------------------< Arrived Jobs >----------------------");
    NewLine();
    for (int i=0; i<Arrive.Num; i++){
        Display("("); Display(Arrive.Jobs[i].ID);
        Display(","); Display(Arrive.Jobs[i].Time);
        Display(") ");
    }
    NewLine();
```

Transd.h/cpp

3. 시뮬레이션 구조 만들기(예제 심플아키텍처)




```
void MakeSES(void)
{

    EFP = new ENTSTR("ef-p");
    EFP->AddItem(new Process("Process"));
    EFP->AddItem(new DIGRAPH("ef"));

    EFP->AddCouple("ef", "Process", "OUT", "in");
    EFP->AddCouple("Process","ef", "out", "IN");

    EFP->SetCurrentItem("ef");
    EFP->AddItem(new GENR("genr"));
    EFP->AddItem(new Transd("transd"));
    EFP->AddCouple("ef", "transd", "IN", "solved");
    EFP->AddCouple("transd", "genr", "out", "stop");

    EFP->AddCouple("genr", "ef", "out", "OUT");
    EFP->AddCouple("genr", "transd", "out", "arriv");
}
```

MakeSES.cpp의 함수 내용을 수정, 구조 생성
MakeSES.cpp외의 다른 라이브러리파일은 절대 수정 금지

4. 시뮬레이션(예제 심플아키텍처)

빌드 후 실행

제목 없음 - Devs

파일(F) 편집(E) 보기(V) 도움말(H) Simulation

```
One step ended...Global Clock (Root): 91.000000
genr(OUT) --> Phase: b  Sigma: 3.000000  When: 93.000000
Process(EXT) --> in:Job-13
transd(EXT) --> :arriv:Job-31 at 93.000000
genr(INT) --> Sigma: 3.000000  When: 93.000000
One step ended...Global Clock (Root): 93.000000
genr(OUT) --> Phase: b  Sigma: 3.000000  When: 96.000000
Process(EXT) --> in:Job-13
transd(EXT) --> :arriv:Job-32 at 96.000000
genr(INT) --> Sigma: 3.000000  When: 96.000000
One step ended...Global Clock (Root): 96.000000
Process(OUT) --> transd(EXT) --> :solved:Job-13 at 98.000000

Process(INT) -->  process : Job-14
One step ended...Global Clock (Root): 98.000000
genr(OUT) --> Phase: b  Sigma: 3.000000  When: 99.000000
Process(EXT) --> in:Job-14
transd(EXT) --> :arriv:Job-33 at 99.000000
genr(INT) --> Sigma: 3.000000  When: 99.000000
One step ended...Global Clock (Root): 99.000000
transd(OUT) -->
genr(EXT) --> :stop: When: 100.000000
transd(INT) -->

  ----------------------< Arrived Jobs >----------------------
(Job-0,0.000000) (Job-1,3.000000) (Job-2,6.000000) (Job-3,9.000000)
(Job-4,12.000000) (Job-5,15.000000) (Job-6,18.000000) (Job-
7,21.000000) (Job-8,24.000000) (Job-9,27.000000) (Job-
10,30.000000) (Job-11,33.000000) (Job-12,36.000000) (Job-
13,39.000000) (Job-14,42.000000) (Job-15,45.000000) (Job-
16,48.000000) (Job-17,51.000000) (Job-18,54.000000) (Job-
19,57.000000) (Job-20,60.000000) (Job-21,63.000000) (Job-
22,66.000000) (Job-23,69.000000) (Job-24,72.000000) (Job-
25,75.000000) (Job-26,78.000000) (Job-27,81.000000) (Job-
28,84.000000) (Job-29,87.000000) (Job-30,90.000000) (Job-
31,93.000000) (Job-32,96.000000) (Job-33,99.000000)

  ----------------------< Solved Jobs >----------------------
(,0.000000) (Job-0,7.000000) (Job-1,14.000000) (Job-2,21.000000)
(Job-3,28.000000) (Job-4,35.000000) (Job-5,42.000000) (Job-
```

준비