

Die C++ - Schnittstelle

Objektorientierte Problemlösungen sind in den letzten Jahren immer mehr zum Standard geworden. Der Erfolg des Dialog Managers beruht auch darauf, daß schon sehr früh die wesentlichen Merkmale der Objektorientierung wie Klassen, Instanzen oder Vererbung integriert worden sind.

Die Dialogdatei

Wie auch bei der C-Schnittstelle erfolgt der Datenaustausch über das Objekt Record. Zur C++-Schnittstelle wurde allerdings das Recordobjekt erweitert, um Funktionsdefinitionen innerhalb des Objektes realisieren zu können. Es werden für alle Records, die Funktionen besitzen, auf C++-Seite Klassen erstellt. Die Attribute des Records werden als Member-Variablen und die Funktionen als C++-Methoden der C++-Klasse dargestellt. Diese Methoden können Sie nun, wie bei der C-Schnittstelle die Funktionen, in einer C++-Datei ausprogrammieren. Für jeden Record mit Funktionen wird automatisch auf der C++-Seite eine Instanz der Klasse angelegt. Dies geschieht entweder beim Starten des Dialoges oder dynamisch. Damit der C++-Entwickler die Möglichkeit hat zu entscheiden, in welche C++-Datei die Klassen generiert werden sollen, gibt es Definitionsdateien.

Die Definitionsdatei

In der Definitionsdatei können bestimmte Markierungen eingefügt werden, damit die Includes, Klassen, Attribute und Methoden an die richtige Stelle geschrieben werden.

Konventionen:

INCLUDE = # Include Kommando

CLASS = Alle Klassendefinitionen

CLASS/"record" = Definition einer Klasse

MEMBER/"attribute" = Definition eines Klassenmembers

METHOD/"function" = Definition einer Klassenmethode

IMPL = Alle Klassenimplementierungen

IMPL/"record" = Implementierung einer Klasse

Beispiel:

Dialogdatei:

```
dialog D {}
record Rec
{
  string A := "Def";
  integer B := 1000;
  function integer F(string S);
}

on dialog start
{
  print Rec:F("Hello");
}
```

Definitionsdatei:

```
@@@:INCLUDE@@@

class MySuper
{
  int len;
  MySuper()
  {
    len = 3;
  }
}

@@@:CLASS/Rec!/@@@ , MySuper
{
  int MyMember;
  @@@MEMBER/A!/@@@
  int MySecondMember;
  @@@MEMBER/*@@@

  /* Methode kann innerhalb definiert werden*/
  @@@METHOD/F!/@@@
  {
    return len+strlen(S)+strlen(A)+B;
  }
  @@@METHOD/*@@@
}
```

Wie Sie in diesem Beispiel sehen,

können auch Superklassen eingefügt oder die generierte Klasse erweitert werden.

Damit aus der Definitionsdatei nun eine C++-Datei wird, wurden zusätzlich Optionen zum Simulationsprogramm definiert. Die Option +merge mischt die Dialogdatei mit der Definitionsdatei; +writecpp erstellt die C++-Ausgabedatei.

Beispiel:

```
idm "Dialogdatei" +merge
"Definitionsdatei" +writecpp
"C++-Ausgabedatei"
```

Das Hauptprogramm

Im Hauptprogramm müssen die Methoden angebunden werden, dazu gibt es eine Methode :bind(DM_ID parentID, DM_String path) (vergleichbar mit DM_BindFunctions in C).

Um auf diese Methode Zugriff zu haben, müssen Sie die neue Include-Datei "IDMcpp.h" eingebunden haben.

Die Includedatei "IDMcpp.h"

Diese Includedatei beinhaltet alle wichtigen Funktionen, um zwischen dem IDM und der C++-Seite Daten auszutauschen, Instanzen auf der C++-Seite zu erstellen bzw. zu löschen.



ISA Informationssysteme
für computerintegrierte
Automatisierung GmbH

Azenbergstr. 35
D-70174 Stuttgart

Tel.: +49 711 22769 20
Fax: +49 711 22769 29
Email: info@isa.de
www.isa.de