The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

The NCBI C++ Toolkit

25: Examples and Demos

Last Update: January 19, 2011.

Overview

The overview for this chapter consists of the following topics:

- Introduction
- Chapter Outline

Introduction

See Getting Started for basic information on using the NCBI C++ Toolkit.

Chapter Outline

- Examples
  - Sample Applications Available with the new_project script
    - A basic example using the xncbi core library
    - An example CGI application using the xcgi and xfcgi libraries.
    - An example for serializable ASN.1 objects and the Object Manager using the xobjects library.
  - id1_fetch ID1 and Entrez2 client
- Examples from the Programming Manual
  - applic.cpp
  - smart.cpp
  - ctypeiter.cpp
  - justcgi.cpp
  - xml2asn.cpp
  - traverseBS.cpp
  - Web-CGI demo
- Test and Demo Programs in the C++ Tree
  - asn2asn.cpp
  - cgitest.cpp
  - cgidemo.cpp
  - coretest.cpp

ID1_FETCH - the ID1 and Entrez2 client

- Synopsis
- Invocation
  - Output Data Formats
  - Lookup Types

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Location: c++/src/app/id1_fetch/id1_fetch.cpp (compiled executable is $NCBI/c++/Release/bin/id1_fetch on NCBI systems)

Synopsis:

- Accept a list of sequences, specified either directly by ID or indirectly by an Entrez query.
- Produce more information about the sequences, either as data from the ID server or as Entrez document summaries.

This program corresponds to idfetch from the C Toolkit.

Invocation

See Table 1.

Note: You must specify exactly one of the options indicating what to look up: -gi, -in, -flat, -fasta, -query, -qf.

Output Data Formats

The possible values of the -fmt argument are shown in Table 2.

Lookup Types

The possible values of the -lt argument are shown in Table 3.

Output Complexity Levels

The possible values of the -maxplex argument are shown in Table 4.

Flattened Sequence ID Format

A flattened sequence ID has one of the following three formats, where square brackets [...] surround optional elements:

- type([name or locus][,[accession][,[release][,version]]])
- type=accession[.version]
- type:number

The first format requires quotes in most shells.

The type is a number, indicating who assigned the ID, as follows:

- Local use
- GenInfo backbone sequence ID
- GenInfo backbone molecule type
- GenInfo import ID
- GenBank
- The European Molecular Biology Laboratory (EMBL)
- The Protein Information Resource (PIR)

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

- SWISS-PROT
- A patent
- RefSeq
- General database reference
- GenInfo integrated database (GI)
- The DNA Data Bank of Japan (DDBJ)
- The Protein Research Foundation (PRF)
- The Protein DataBase (PDB)
- Third-party annotation to GenBank
- Third-party annotation to EMBL
- Third-party annotation to DDBJ
- TrEMBL

FASTA Sequence ID Format

This format consists of a two- or three-letter tag indicating the ID's type, followed by one or more data fields, which are separated from the tag and each other by vertical bars (|). The vertical bar is a special character in most command-line shells, so command-line arguments containing ID's usually must be quoted. Table 5 shows the specific possibilities.

Example Usage

```
id1_fetch -query '5-delta4 isomerase' -lt none -db Nucleotide

34

id1_fetch -fmt genbank -gi 34

LOCUS BT3BHSD 1632 bp mRNA MAM 12-SEP-1993
DEFINITION Bovine mRNA for 3 beta hydroxy-5-ene steroid dehydrogenase/delta
 5-delta4 isomerase (EC 1.1.1.145, EC 5.3.3.1).
ACCESSION X17614
VERSION X17614.1 GI:34
KEYWORDS 3 beta-hydroxy-delta5-steroid dehydrogenase;
 steroid delta-isomerase.
...
FEATURES Location/Qualifiers
...
 CDS 105..1226
 /codon_start=1
 /transl_table=1
 /function="3 beta-HSD (AA 1-373)"
 /protein_id="CAA35615.1"
 /db_xref="GI:35"
 /translation="MAGWSCLVTGGGGFLGQRIICLLVEEKDLQEIRVLDKVFRPEVR
 EEFSKLQSKIKLTLLEGDILDEQCLKGACQGTSVVIHTASVIDVRNAVPRETIMNVNV
 KGTQLLLEACVQASVPVFIHTSTIEVAGPNSYREIIQDGREEEHHESAWSSPYPYSKK
 LAEKAVLGANGWALKNGGTLYTCALRPMYIYGEGSPFLSAYMHGALNNNGILTNHCKF
 SRVNPVYVGNVAWAHILALRALRDPKKVPNIQGQFYYISDDTPHQSYDDLNYTLSKEW
 GFCLDSRMSLPISLQYWLAFLLEIVSFLLSPIYKYNPCFNRHLVTLSNSVFTFSYKKA
```

```
 QRDLGYEPLYTWEEAKQKTKEWIGSLVKQHKETLKTKIH"
 /db_xref="SWISS-PROT:P14893"
...
 1441 ggacagacaa ggtgatttgc tgcagctgct ggcaccaaaa tctcagtggc agattctgag
 1501 ttatttgggc ttcttgtaac ttcgagtttt gcctcttagt cccactttct ttgttaaatg
 1561 tggaagcatt tcttttaaaa gttcatattc cttcatgtag ctcaataaaa atgatcaaca
 1621 ttttcatgac tc
//

id1_fetch -fmt genpept -gi 35

LOCUS CAA35615 373 aa MAM 12-SEP-1993
DEFINITION Bovine mRNA for 3 beta hydroxy-5-ene steroid dehydrogenase/delta
 5-delta4 isomerase (EC 1.1.1.145, EC 5.3.3.1), and translated
 products.
ACCESSION CAA35615
VERSION CAA35615.1 GI:35
PID g35
SOURCE cow.
 ORGANISM Bos taurus
 Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
 Mammalia; Eutheria; Cetartiodactyla; Ruminantia; Pecora; Bovoidea;
 Bovidae; Bovinae; Bos.
...
ORIGIN
 1 magwsclvtg gggflgqrii cllveekdlq eirvldkvfr pevreefskl qskikltlle
 61 gdildeqclk gacqgtsvvi htasvidvrn avpretimnv nvkgtqllle acvqasvpvf
 121 ihtstievag pnsyreiiqd greeehhesa wsspypyskk laekavlgan gwalknggtl
 181 ytcalrpmyi ygegspflsa ymhgalnnng iltnhckfsr vnpvyvgnva wahilalral
 241 rdpkkvpniq gqfyyisddt phqsyddlny tlskewgfcl dsrmslpisl qywlafllei
 301 vsfllspiyk ynpcfnrhlv tlsnsvftfs ykkaqrdlgy eplytweeak qktkewigsl
 361 vkqhketlkt kih
//

id1_fetch -fmt fasta -gi 35 -maxplex bioseq

>emb|CAA35615.1||gi|35 Bovine mRNA for 3 beta hydroxy-5-ene steroid
dehydrogenase/delta
 5-delta4 isomerase (EC 1.1.1.145, EC 5.3.3.1), and translated products
MAGWSCLVTGGGGFLGQRIICLLVEEKDLQEIRVLDKVFRPEVREEFSKLQSKIKLTLLEGDILDEQCLK
GACQGTSVVIHTASVIDVRNAVPRETIMNVNVKGTQLLLEACVQASVPVFIHTSTIEVAGPNSYREIIQD
GREEEHHESAWSSPYPYSKKLAEKAVLGANGWALKNGGTLYTCALRPMYIYGEGSPFLSAYMHGALNNNG
ILTNHCKFSRVNPVYVGNVAWAHILALRALRDPKKVPNIQGQFYYISDDTPHQSYDDLNYTLSKEWGFCL
DSRMSLPISLQYWLAFLLEIVSFLLSPIYKYNPCFNRHLVTLSNSVFTFSYKKAQRDLGYEPLYTWEEAK
QKTKEWIGSLVKQHKETLKTKIH

id1_fetch -lt ids -gi 35

ID1server-back ::= ids {
 embl {
 accession "CAA35615",
```

```
 version 1
 },
 general {
 db "NCBI_EXT_ACC",
 tag str "FPAA037960"
 },
 gi 35
}

id1_fetch -lt state -fasta 'emb|CAA35615' -fmt xml

<?xml version="1.0"?>
<!DOCTYPE ID1server-back PUBLIC "-//NCBI//NCBI ID1Access/EN"
"NCBI_ID1Access.dtd">
<ID1server-back>
 <ID1server-back_gistate>40</ID1server-back_gistate>
</ID1server-back>

id1_fetch -lt state -flat '5=CAA35615.1' -fmt asnb | od -t u1

0000000 166 128 002 001 040 000 000
0000007

id1_fetch -lt state -flat '5(,CAA35615)' -fmt fasta

gi = 35, states: LIVE

id1_fetch -lt history -flat '12:35' -fmt fasta

GI Loaded DB Retrieval No.
-- ------ -- -------------
35 03/08/1999 EMBL 274319

id1_fetch -lt revisions -gi 35 -fmt fasta

GI Loaded DB Retrieval No.
-- ------ -- -------------
35 03/08/1999 EMBL 274319
35 06/06/1996 OLD02 84966
35 05/27/1995 OLDID 1524022
35 11/29/1994 OLDID 966346
35 08/31/1993 OLDID 426053
35 04/20/1993 OLDID 27

id1_fetch -fmt quality -gi 13508865

>AL590146.2 Phrap Quality (Length: 121086, Min: 31, Max: 99)
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
```

The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

```
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
...
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 54 54 56 56 54 54 54 56 56 56 56 65 65 57 60 56 56 59 59
 56 56 56 49 99 31 31 49 49 54 63 63 54 51 53 55 51 51 49 58
 58 58 58 53 52 49 51 51 51 52 55 51 51 51 49 49 49 63 63 60
 65 65 59 54 54 54 54 54 56 60 60 65 65 65 65 70 70 65 65 65
 65 65 65 65 60 59 59 66 66 66 67 65 65 63 46 65 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
...
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
 99 99 99 99 99 99
```

Examples from the Programming Manual

- applic.cpp
- smart.cpp
- An Example of a Web-based CGI Application - Source Files
  - car.cpp
  - car.hpp
  - car_cgi.cpp
  - Makefile.car_app
  - car.html

applic.cpp

```
// File name: applic.cpp
// Description: Using the CNcbiApplication class with CNcbiDiag, CArgs
// and CArgDescription classes

#include <corelib/ncbistd.hpp>
#include <corelib/ncbiutil.hpp>
#include <corelib/ncbiargs.hpp>
#include <corelib/ncbiapp.hpp>
#include <corelib/ncbienv.hpp>

USING_NCBI_SCOPE;

class CTestApp : public CNcbiApplication {
public:
 virtual int Run();
};
int CTestApp::Run() {
```

```
 auto_ptr<CArgs> args;

 // create a CArgDescriptions object to constrain the input arguments;
 // Argument descriptions are added using:

 // void AddKey(string& name, string& synopsis, string& comment, EType,
TFlags);
 // void AddOptionalKey(string& name, string& synopsis, string& comment,
EType,
 // string& default, TFlags);
 // void AddFlag(string& name, string& comment);

 {
 CArgDescriptions argDesc;

 // Required arguments:
 argDesc.AddKey("n","integer","integer between 1 and
10",argDesc.eInteger);
 argDesc.AddKey("f","float","float between 0.0 and 1.0",argDesc.eDouble);
 argDesc.AddKey("i","inputFile","data file
in",CArgDescriptions::eInputFile);

 // optional arguments:
 argDesc.AddOptionalKey("l","logFile","log errors to <logFile>",
 argDesc.eOutputFile);

 // optional flags
 argDesc.AddFlag("it", "input text");
 argDesc.AddFlag("ib", "input binary");

 try {
 args.reset(argDesc.CreateArgs(GetArguments()));
 }
 catch (exception& e) {
 string a;
 argDesc.SetUsageContext(GetArguments()[0],
 "CArgDescriptions demo program");

 cerr << e.what() << endl;
 cerr << argDesc.PrintUsage(a);
 return (-1);
 }
 }

 int intIn = (*args)["n"].AsInteger();
 float floatIn = (*args)["f"].AsDouble();
 string inFile = (*args)["i"].AsString();

 // process optional args
 if ( args->Exest("l") ) {
 SetDiagStream(&(*args)["l"].AsOutputFile());
```

```
 }

 bool textIn = args->Exist("it");
 bool binIn = (*args)["ib"].AsBoolean();

 if (! (textIn ^ binIn) ) {
 ERR_POST_X(1, Error << "input type must be specified using -it or -ib");
 }

 string InputType;
 if ( textIn ) {
 InputType = "text";
 } else if ( binIn ) {
 InputType = "binary";
 }

 ERR_POST_X(2, Info << "intIn = " << intIn << " floatIn = " << floatIn
 << " inFile = " << inFile << " input type = " << InputType);

 return 0;
}
int main(int argc, const char* argv[])
{
 CNcbiOfstream diag("moreApp.log");
 SetDiagStream(&diag);

 // Set the global severity level to Info
 SetDiagPostLevel(eDiag_Info);

 CTestApp theTestApp;
 return theTestApp.AppMain(argc, argv);
}
```

smart.cpp

```
// File name: smart.cpp
// Description: Memory management using auto_ptr versus CRef

#include <corelib/ncbiapp.hpp>
#include <corelib/ncbiobj.hpp>

USING_NCBI_SCOPE;

class CTestApp : public CNcbiApplication {
public:
 virtual int Run(void);
};
```

```
//////////////////////////////////////////////////////////////////
//
// 1. Install an auto_ptr to an int and make a copy - then try to
// reference the value from the original auto_ptr.
//
// 2. Do the same thing using CRefs instead of auto_ptrs.
//
//////////////////////////////////////////////////////////////////

int CTestApp::Run()
{
 auto_ptr<int> orig_ap;
 orig_ap.reset(new int(5));
 {
 auto_ptr<int> copy_ap = orig_ap;

 if ( !orig_ap.get() ) {
 cout << "orig_ap no longer exists - copy_ap = " << *copy_ap << endl;
 } else {
 cout << "orig_ap = " << *orig_ap << ", copy_ap = "
 << *copy_ap << endl;
 }
 }
 if ( orig_ap.get() ) {
 cout << "orig_ap = " << *orig_ap << endl;
 }

 CRef< CObjectFor<int> > orig(new CObjectFor<int>);
 *orig = 5;
 {
 CRef< CObjectFor<int> > copy = orig;

 if ( !orig ) {
 cout << "orig no longer exists - copy = " << (int&) *copy << endl;
 } else {
 cout << "orig = " << (int&) *orig << ", copy = "
 << (int&) *copy << endl;
 }
 }
 if ( orig ) {
 cout << "orig = " << (int&) *orig << endl;
 }
 return 0;
}


int main(int argc, const char* argv[])
{
 CTestApp theTestApp;
```

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

```
 return theTestApp.AppMain(argc, argv);
}
```

An Example of a Web-based CGI Application - Source Files

car.cpp

```
// File name: car.cpp
// Description: Implement the CCarAttr class

#include "car.hpp"

BEGIN_NCBI_SCOPE

/////////////////////////////////////////////////////////////////////////////
///
// CCarAttr::

set<string> CCarAttr::sm_Features;
set<string> CCarAttr::sm_Colors;


CCarAttr::CCarAttr(void)
{
 // make sure there is only one instance of this class
 if ( !sm_Features.empty() ) {
 _TROUBLE;
 return;
 }

 // initialize static data
 sm_Features.insert("Air conditioning");
 sm_Features.insert("CD Player");
 sm_Features.insert("Four door");
 sm_Features.insert("Power windows");
 sm_Features.insert("Sunroof");

 sm_Colors.insert("Black");
 sm_Colors.insert("Navy");
 sm_Colors.insert("Silver");
 sm_Colors.insert("Tan");
 sm_Colors.insert("White");
}

// dummy instance of CCarAttr -- to provide initialization of
// CCarAttr::sm_Features and CCarAttr::sm_Colors
static CCarAttr s_InitCarAttr;
```

```
END_NCBI_SCOPE
```

car.hpp

```
// File name: car.hpp
// Description: Define the CCar and CCarAttr classes

#ifndef CAR_HPP
#define CAR_HPP

#include <coreilib/ncbistd.hpp>
#include <set>

BEGIN_NCBI_SCOPE

//////////////////////
// CCar

class CCar
{
public:
 CCar(unsigned base_price = 10000) { m_Price = base_price; }

 bool HasFeature(const string&amp;amp; feature_name) const
 { return m_Features.find(feature_name) != m_Features.end(); }
 void AddFeature(const string& feature_name)
 { m_Features.insert(feature_name); }

 void SetColor(const string& color_name) { m_Color = color_name; }
 string GetColor(void) const { return m_Color; }

 const set<string>& GetFeatures() const { return m_Features; }
 unsigned GetPrice(void) const
 { return m_Price + 1000 * m_Features.size(); }

private:
 set<string> m_Features;
 string m_Color;
 unsigned m_Price;
};



//////////////////////
// CCarAttr -- use a dummy all-static class to store available car
attributes
```

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

```
class CCarAttr {
public:
 CCarAttr(void);
 static const set<string>& GetFeatures(void) { return sm_Features; }
 static const set<string>& GetColors (void) { return sm_Colors; }
private:
 static set<string> sm_Features;
 static set<string> sm_Colors;
};


END_NCBI_SCOPE

#endif /* CAR__HPP */
```

car_cgi.cpp

```
// File name: car_cgi.cpp
// Description: Implement the CCarCgi class and function main

#include <cgi/cgiapp.hpp>
#include <cgi/cgictx.hpp>
#include <html/html.hpp>
#include <html/page.hpp>

#include "car.hpp"

USING_NCBI_SCOPE;

/////////////////////////////////////////////////////////////////////////////
///
// CCarCgi:: declaration

class CCarCgi : public CCgiApplication
{
public:
 virtual int ProcessRequest(CCgiContext& ctx);

private:
 CCar* CreateCarByRequest(const CCgiContext& ctx);

 void PopulatePage(CHTMLPage& page, const CCar& car);

 static CNCBINode* ComposeSummary(const CCar& car);
 static CNCBINode* ComposeForm (const CCar& car);
 static CNCBINode* ComposePrice (const CCar& car);

 static const char sm_ColorTag[];
```

```
 static const char sm_FeatureTag[];
};


/////////////////////////////////////////////////////////////////////////////
///
// CCarCgi:: implementation


const char CCarCgi::sm_ColorTag[] = "color";
const char CCarCgi::sm_FeatureTag[] = "feature";


int CCarCgi::ProcessRequest(CCgiContext& ctx)
{
 // Create new "car" object with the attributes retrieved
 // from the CGI request parameters
 auto_ptr<CCar> car;
 try {
 car.reset( CreateCarByRequest(ctx) );
 } catch (exception& e) {
 ERR_POST_X(1, "Failed to create car: " << e.what());
 return 1;
 }

 // Create an HTML page (using the template file "car.html")
 CRef<CHTMLPage> page;
 try {
 page = new CHTMLPage("Car", "car.html");
 } catch (exception& e) {
 ERR_POST_X(2, "Failed to create the Car HTML page: " << e.what());
 return 2;
 }

 // Register all substitutions for the template parameters <@XXX@>
 // (fill them out using the "car" attributes)
 try {
 PopulatePage(*page, *car);
 } catch (exception& e) {
 ERR_POST_X(3, "Failed to populate the Car HTML page: " << e.what());
 return 3;
 }

 // Compose and flush the resultant HTML page
 try {
 const CCgiResponse& response = ctx.GetResponse();
 response.WriteHeader();
 page->Print(response.out(), CNCBINode::eHTML);
 response.Flush();
 } catch (exception& e) {
 ERR_POST_X(4, "Failed to compose and send the Car HTML page: " << e.what
```

```
());
 return 4;
 }

 return 0;
}


CCar* CCarCgi::CreateCarByRequest(const CCgiContext& ctx)
{
 auto_ptr<CCar> car(new CCar);

 // Get the list of CGI request name/value pairs
 const TCgiEntries& entries = ctx.GetRequest().GetEntries();

 TCgiEntriesCI it;

 // load the car with selected features
 pair<TCgiEntriesCI,TCgiEntriesCI> feature_range =
 entries.equal_range(sm_FeatureTag);
 for (it = feature_range.first; it != feature_range.second; ++it) {
 car->AddFeature(it->second);
 }

 // color
 if ((it = entries.find(sm_ColorTag)) != entries.end()) {
 car->SetColor(it->second);
 } else {
 car->SetColor(*CCarAttr::GetColors().begin());
 }

 return car.release();
}


 /************ Create a form with the following structure:
 <form>
 <table>
 <tr>
 <td> (Features) </td>
 <td> (Colors) </td>
 <td> (Submit) </td>
 </tr>
 </table>
 </form>
 ********************/

CNCBINode* CCarCgi::ComposeForm(const CCar& car)
{
 set<string>::const_iterator it;
```

```
 CRef<CHTML_table> Table = new CHTML_table();
 Table->SetCellSpacing(0)->SetCellPadding(4)
 ->SetBgColor("#CCCCCC")->SetAttribute("border", "0");

 CRef<CHTMLNode> Row = new CHTML_tr();

 // features (check boxes)
 CRef<CHTMLNode> Features = new CHTML_td();
 Features->SetVAlign("top")->SetWidth("200");
 Features->AppendChild(new CHTMLText("Options: <br>"));

 for (it = CCarAttr::GetFeatures().begin();
 it != CCarAttr::GetFeatures().end(); ++it) {
 Features->AppendChild
 (new CHTML_checkbox
 (sm_FeatureTag, *it, car.HasFeature(*it), *it));
 Features->AppendChild(new CHTML_br());
 }

 // colors (radio buttons)
 CRef<CHTMLNode> Colors = new CHTML_td();
 Colors->SetVAlign("top")->SetWidth("128");
 Colors->AppendChild(new CHTMLText("Color: <br>"));

 for (it = CCarAttr::GetColors().begin();
 it != CCarAttr::GetColors().end(); ++it) {
 Colors->AppendChild
 (new CHTML_radio
 (sm_ColorTag, *it, !NStr::Compare(*it, car.GetColor()), *it));
 Colors->AppendChild(new CHTML_br());
 }

 Row->AppendChild(&*Features);
 Row->AppendChild(&*Colors);
 Row->AppendChild
 ((new CHTML_td())->AppendChild
 (new CHTML_submit("submit", "submit")));
 Table->AppendChild(&*Row);

 // done
 return (new CHTML_form("car.cgi", CHTML_form::eGet))->AppendChild
(&*Table);
}


CNCBINode* CCarCgi::ComposeSummary(const CCar& car)
{
 string summary = "You have ordered a " + car.GetColor() + " model";

 if ( car.GetFeatures().empty() ) {
 summary += " with no additional features.<br>";
```

```
 return new CHTMLText(summary);
 }

 summary += " with the following features:<br>";
 CRef<CHTML_ol> ol = new CHTML_ol();

 for (set<string>::const_iterator i = car.GetFeatures().begin();
 i != car.GetFeatures().end(); ++i) {
 ol->AppendItem(*i);
 }
 return (new CHTMLText(summary))->AppendChild((CNodeRef&)ol);
}


CNCBINode* CCarCgi::ComposePrice(const CCar& car)
{
 return
 new CHTMLText("Total price: $" + NStr::UIntToString(car.GetPrice()));
}


void CCarCgi::PopulatePage(CHTMLPage& page, const CCar& car)
{
 page.AddTagMap("FORM", ComposeForm (car));
 page.AddTagMap("SUMMARY", ComposeSummary (car));
 page.AddTagMap("PRICE", ComposePrice (car));
}



////////////////////////////////////////////////////////////////////////////
///
// MAIN


int main(int argc, char* argv[])
{
 SetDiagStream(&NcbiCerr);
 return CCarCgi().AppMain(argc, argv);
}
```

Makefile.car_app

```
# Makefile: /home/zimmerma/car/Makefile.car_app
# This file was originally generated by shell script "new_project"

### PATH TO A PRE-BUILT C++ TOOLKIT ###
builddir = /netopt/ncbi_tools/c++/GCC-Debug/build
```

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

```
# builddir = $(NCBI)/c++/Release/build


### DEFAULT COMPILATION FLAGS -- DON'T EDIT OR MOVE THESE 4 LINES !!! ###
include $(builddir)/Makefile.mk
srcdir = .
BINCOPY = @:
LOCAL_CPPFLAGS = -I.


############################################################################
###
### EDIT SETTINGS FOR THE DEFAULT (APPLICATION) TARGET HERE ###
APP = car.cgi
SRC = car car_cgi

# PRE_LIBS = $(NCBI_C_LIBPATH) .....
LIB = xhtml xcgi xncbi

# LIB = xser xhtml xcgi xncbi xconnect
# LIBS = $(NCBI_C_LIBPATH) -lncbi $(NETWORK_LIBS) $(ORIG_LIBS)

# CPPFLAGS = $(ORIG_CPPFLAGS) $(NCBI_C_INCLUDE)
# CFLAGS = $(ORIG_CFLAGS)
# CXXFLAGS = $(ORIG_CXXFLAGS)
# LDFLAGS = $(ORIG_LDFLAGS)
# ###
############################################################################
###

### APPLICATION BUILD RULES -- DON'T EDIT OR MOVE THIS LINE !!! ###
include $(builddir)/Makefile.app


### PUT YOUR OWN ADDITIONAL TARGETS (MAKE COMMANDS/RULES) BELOW HERE ###
```

car.html

```
<html>
<head>
<title>Automobile Order Form</title>
</head>
<body>
<h1>Place your order here</h1>
<@FORM@>
<@SUMMARY@>
<@PRICE@>
</body>
</html>
```

The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

Table 1. Invocation flags

Argument	Value	Effect
-h		Print usage message and exit.
-gi N	integer	GenInfo ID of sequence to look up.
-fmt fmt	format type	Output data format; default is asn (ASN.1 text).
-out file	filename	Write output to specified file rather than stdout.
-log file	filename	Write errors and messages to specified file rather than stderr.
-db str	string	Use specified database. **Mandatory for** Entrez **queries**, where it is normally either Nucleotide or Protein. Also specifies satellite database for sequence-entry lookups.
-ent N	integer	Dump specified subentity. Only relevant for sequence-entry lookups.
-lt type	lookup type	Type of lookup; default is entry (sequence entry).
-in file	filename	Read sequence IDs from file rather than command line. May contain raw GI IDs, flattened IDs, and FASTA-format IDs.
-maxplex m	complexity	Maximum output complexity level; default is entry (entire entry).
-flat id	flat ID	Flattened ID of sequence to look up.
-fasta id	FASTA ID	FASTA-style ID of sequence to look up.
-query str	string	Generate ID list from specified Entrez query.
-qf file	file	Generate ID list from Entrez query in specified file.

The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

Table 2. Output data formats

Value	**Format**	**Comments**
asn	ASN.1 text (default)	
asnb	ASN.1 binary	
docsum	Entrez document summary	Lookup type is irrelevant.
fasta	FASTA	Produces state as simple text; produces history in tabular form.
genbank	GenBankflat-file format	Lookup type must be entry (default).
genpept	GenPept flat-file format	Lookup type must be entry (default).
quality	Quality scores	Lookup type must be entry (default); data not always available.
xml	XML	Isomorphic to ASN.1 output.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Table 3. Lookup types

Value	Description
entry	The actual sequence entry (default)
history	Summary of changes to the sequence data
ids	All of the sequence's IDs
none	Just the GI ID
revisions	Summary of changes to the sequence data or annotations
state	The sequence's status

The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

Table 4. Maximum output complexity level values

Value	Description
bioseq	Just the bioseq of interest
bioseq-set	Minimal bioseq-set
entry	Entire entry (default)
nuc-prot	Minimal nuc-prot
pub-set	Minimal pub-set

Table 5. FASTA sequence ID format values

Type	**Format(s)** [1]	**Example(s)**
local	lcl|integer lcl|string	lcl|123 lcl|hmm271
GenInfo backbone seqid	bbs|integer	bbs|123
GenInfo backbone moltype	bbm|integer	bbm|123
GenInfo import ID	gim|integer	gim|123
GenBank	gb|accession|locus	gb|M73307|AGMA13GT
EMBL	emb|accession|locus	emb|CAM43271.1|
PIR	pir|accession|name	pir||G36364
SWISS-PROT	sp|accession|name	sp|P01013|OVAX_CHICK
patent	pat|country|patent|sequence	pat|US|RE33188|1
pre-grant patent	pgp|country|application-number|seq-number	pgp|EP|0238993|7
RefSeq [2]	ref|accession|name	ref|NM_010450.1|
general database reference	gnl|database|integer gnl|database|string	gnl|taxon|9606 gnl|PID|e1632
GenInfo integrated database	gi|integer	gi|21434723
DDBJ	dbj|accession|locus	dbj|BAC85684.1|
PRF	prf|accession|name	prf||0806162C
PDB	pdb|entry|chain	pdb|1I4L|D
third-party GenBank	tpg|accession|name	tpg|BK003456|
third-party EMBL	tpe|accession|name	tpe|BN000123|
third-party DDBJ	tpd|accession|name	tpd|FAA00017|
TrEMBL	tr|accession|name	tr|Q90RT2|Q90RT2_9HIV1
genome pipeline	gpp|accession|name	gpp|GPC_123456789|
named annotation track	nat|accession|name	nat|AT_123456789.1|

[1] Spaces should not be present in ID's. It's okay to leave off the final vertical bar for most text ID types (such as gb) when the locus is absent; apart from that, vertical bars must be present even if an adjacent field is omitted.

[2] Some RefSeq accessions have additional letters following the underscore. See the RefSeq accession format reference for details.

[3] For NCBI internal use.