DS Lab Assignment 8

#include "Quiz.hh"

#include <iostream>

/\*\* Name is defined in the server \*/

#define SERVER\_NAME     "Quiz"

Quiz::QuizServer\_ptr service\_server;

using *namespace* std;

*void* insert\_question(const *char*\* *sentence*, *int* *numAnswers*, Quiz::Answer\*\* *answers*, *int* *numCorrectAnswers*, CORBA::Char\* *correctAnswers*);

*void* create\_questions();

*int* main(*int* *argc*, *char* \*\* *argv*)

{

    try {

        //------------------------------------------------------------------------

        // Initialize ORB object.

        //------------------------------------------------------------------------

        CORBA::ORB\_ptr orb = CORBA::ORB\_init(argc, argv);

        //------------------------------------------------------------------------

        // Resolve service

        //------------------------------------------------------------------------

        service\_server = 0;

        try {

            //------------------------------------------------------------------------

            // Bind ORB object to name service object.

            // (Reference to Name service root context.)

            //------------------------------------------------------------------------

            CORBA::Object\_var ns\_obj = orb->resolve\_initial\_references("NameService");

            if (!CORBA::is\_nil(ns\_obj)) {

                //------------------------------------------------------------------------

                // Bind ORB object to name service object.

                // (Reference to Name service root context.)

                //------------------------------------------------------------------------

                CosNaming::NamingContext\_ptr nc = CosNaming::NamingContext::\_narrow(ns\_obj);

                //------------------------------------------------------------------------

                // The "name text" put forth by CORBA server in name service.

                // This same name ("MyServerName") is used by the CORBA server when

                // binding to the name server (CosNaming::Name).

                //------------------------------------------------------------------------

                CosNaming::Name name;

                name.length(1);

                name[0].id = CORBA::string\_dup(SERVER\_NAME);

                name[0].kind = CORBA::string\_dup("");

                //------------------------------------------------------------------------

                // Resolve "name text" identifier to an object reference.

                //------------------------------------------------------------------------

                CORBA::Object\_ptr obj = nc->resolve(name);

                if (!CORBA::is\_nil(obj)) {

                    service\_server = Quiz::QuizServer::\_narrow(obj);

                }

            }

        } catch (CosNaming::NamingContext::NotFound &) {

            cerr << "Caught corba not found" << endl;

        } catch (CosNaming::NamingContext::InvalidName &) {

            cerr << "Caught corba invalid name" << endl;

        } catch (CosNaming::NamingContext::CannotProceed &) {

            cerr << "Caught corba cannot proceed" << endl;

        }

        //------------------------------------------------------------------------

        // Do stuff

        //------------------------------------------------------------------------

        if (!CORBA::is\_nil(service\_server)) {

                        cout << "QuizClient client is running ..." << endl;

                    orb->register\_value\_factory("IDL:Quiz/Answer:1.0", new Quiz::Answer\_init());

                        create\_questions();

                        //

                        // get random question

                        //

                        orb->register\_value\_factory("IDL:Quiz/Question:1.0", new Quiz::Question\_init());

                        Quiz::Question\* received\_question = new OBV\_Quiz::Question();

                        service\_server->getQuestion(received\_question);

                        const *char*\* received\_question\_sentence = received\_question->sentence();

                        CORBA::Long received\_question\_id = received\_question->id();

                        Quiz::Question::AnswerSeq received\_question\_answers = received\_question->answers();

*int* numAnswers = received\_question\_answers.length();

                        cout << "Received Question: id=" << received\_question\_id << ", sentence=" << received\_question\_sentence << endl;

                        for(*int* i = 0; i < numAnswers; i++) {

                            if(received\_question\_answers[i]) {

                                cout << "\t" << received\_question\_answers[i]->id() << ": " << received\_question\_answers[i]->sentence() << endl;

                            }

                        }

        }

        //------------------------------------------------------------------------

        // Destroy OBR

        //------------------------------------------------------------------------

        orb->destroy();

    } catch (CORBA::UNKNOWN) {

        cerr << "Caught CORBA exception: unknown exception" << endl;

    }

}

*void* insert\_question(const *char*\* *sentence*, *int* *numAnswers*, Quiz::Answer\*\* *answers*, *int* *numCorrectAnswers*, CORBA::Char\* *correctAnswers*)

{

        Quiz::Question::AnswerSeq\* answersSeq = new OBV\_Quiz::Question::AnswerSeq(numAnswers, numAnswers, answers, 1);

        Quiz::CompleteQuestion::CharSeq\* correctAnswersSeq = new OBV\_Quiz::CompleteQuestion::CharSeq(numCorrectAnswers, numCorrectAnswers, correctAnswers, 1);

        Quiz::CompleteQuestion\* new\_question = new OBV\_Quiz::CompleteQuestion(0, sentence, \*answersSeq, \*correctAnswersSeq);

    CORBA::Long question\_received\_id = service\_server->insertQuestion(new\_question);

        cout << "send question and received id " << question\_received\_id << endl;

}

*void* create\_questions()

{

    // create first question

        const *char*\* question\_sentence = "It applies to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages. What is it?";

        Quiz::Answer\*\* question0\_answers = new Quiz::Answer\*[3];

        question0\_answers[0] = new OBV\_Quiz::Answer('a', "Hetereogenity");

        question0\_answers[1] = new OBV\_Quiz::Answer('b', "Middleware");

        question0\_answers[2] = new OBV\_Quiz::Answer('c', "Opennes");

        CORBA::Char question0\_correctAnswers[] = {'b'};

        insert\_question(question\_sentence, 3, question0\_answers, 1, question0\_correctAnswers);

        // create second question

        question\_sentence = "It refers to a running program (a process) on a networked computer that accepts requests from programs running on other computers to perform a service and responds appropriately.";

        Quiz::Answer\*\* question1\_answers = new Quiz::Answer\*[3];

        question1\_answers[0] = new OBV\_Quiz::Answer('a', "Server");

        question1\_answers[1] = new OBV\_Quiz::Answer('b', "Middleware");

        question1\_answers[2] = new OBV\_Quiz::Answer('c', "Client");

        CORBA::Char question1\_correctAnswers[] = {'a'};

        insert\_question(question\_sentence, 3, question1\_answers, 1, question1\_correctAnswers);

}