iRobot Programmer's Guide

Catalog

1. What is *iRobot*

iRobot (named from *Internet Robots*) is a general Internet robot development platform. It provides a visual platform for Internet robots creation and a robot engine for Web data integration. Using iRobot, you can create your own robots to navigate Web sites, fill forms, extract Web data and even compute scientific data online. You don't need to have programming skills to create Internet robots, but with programming skills, you can create more powerful robots! You can also enjoy fun robots other peoples have already created for you!

2. General Guide for iRobot Programmers

IRobot allows you to develop your own Web-enabled programs to access and integrate Web data. For this purpose, you create and test a robot using the iRobot visual platform, embed the robot in your program, and control and interact with the robot using the iRobotX activeX control.

For guidance to Internet robot creation with iRobot Visual interfaces, refer the iRobot User's Manual.

The IRobotX control "IRobotX.OCX" and the iRobot visual interfaces can be downloaded from http://irobotsoft.com/download.htm. The OCX can be used in Visual Basic and Visual C++ application.

To program with IRobotX , follow these three steps:

1) Register the IRobotX control by command:
   C:\irobot\> regsvr32 IRobotX.OCX
2) Create and test a robot for its basic functionality
3) Define variables in the robot for so that your program can access them.
4) Embed and run the robot in your program using the IRobotX control

3. An Example Application Using Existing Robots

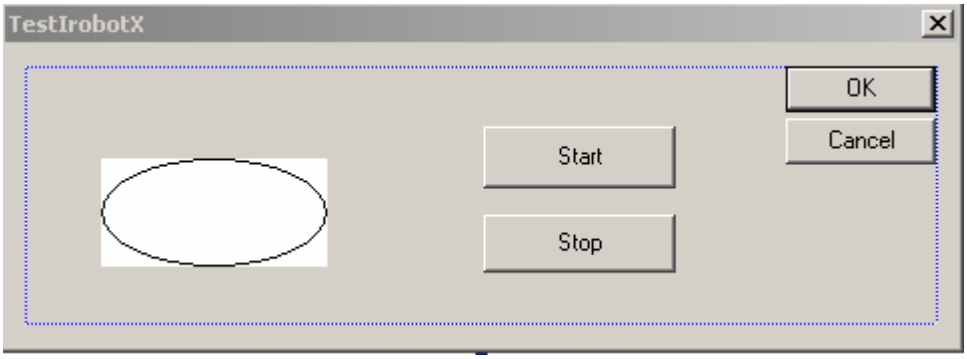
This example shows how to do the BLAST search from NCBI in your program and retrieve the matching sequences automatically. The robot file for BLAST has been created and can be downloaded from http://irobotsoft.com/robots/blastx.irb. It completes the following steps:

1) Submit a nucleotide sequence for Blast
2) Retrieve the title, description and GI number for each matching entry

3) Retrieve the references and sequences for each entry
4) Save the matching entries into an XML file

We are going to create a Visual C++ program to collect all sequences matching a candidate sequence with eval<1e-6. (Note that we can also create a Visual Basic program to complete a same functionality in a similar procedure). The steps are:

1) Install iRobot visual interface and the iRobotX control (follows the instructions from the irobotsoft Web site).
2) Use MFC application wizard to create a simple dialog-based application.
3) In the dialog interface, insert an IRobotX control (right-click the interface, select "insert ActiveX contrl …", and select the "IRobotX" control).
4) Use the wizard interface to assign a variable to the new IRobotX control. (right click the IRobotX control, select "ClassWizard …", select the "member variable" tab, double click the IDC_IROBOTXCTRL1, give it a name such as "m_Robot", and press OK).
5) Add two buttons in the dialog. Let one be a "Start" button, and the other be a "Stop" button. The dialog interface looks like the following now:



6) Double-click on the "Start" button to add a function handler for the "Start" button. Add example codes:

```
void CTestIrobotXDlg::OnButtonStart()
{
	int err=0;
	err=m_Robot.setIrbFile("C:\\Program Files\\irobot\\blastx.irb","");
	if (err>=0) err=m_Robot.showBrowser(true);
	if (err>=0) err=m_Robot.setLogFile("blast.log", 255);
	if (err>=0) err=m_Robot.prepareRobot("C:\\Program Files\\irobot\\");
	if (err>=0) err=m_Robot.setVariable("RetrieveSequence","1");
	if (err>=0) err=m_Robot.setVariable("FromXML","1");
	if (err>=0) err=m_Robot.setVariable("ToSaveFile","0");
	if (err>=0) err=m_Robot.setVariable("MaxEval","1e-6");
	if (err>=0) err=m_Robot.setVariable("BlastSequence", BlastSequence);
	if (err>=0) err=m_Robot.run("blast");
}
```

7) Double-click on the "Stop" button to add a function handler for the "Start"

button. Add example codes:

```
void CTestIrobotXDlg::OnButtonStop()
{
    m_Robot.reset();
}
```

8) Right-click on the IRobotX control to add an event handler for the "AssignVariable" event. (Right-click, select "Events …", double-click the "AssignVariable", accept the default name, and OK). Add example codes:

```
void CTestIrobotXDlg::OnAssignVariableIrobotxctrl1(LPCTSTR name, LPCTSTR value)
{
    // TODO: Add your control notification handler code here
    CString varname=name;
    if (varname == "NewMatch"){
        CString MatchTitle=m_Robot.getVariable("MatchTitle");
        CString MatchDescription=m_Robot.getVariable("MatchDescription");
        CString MatchGi=m_Robot.getVariable("MatchGi");
        CString MatchUrl=m_Robot.getVariable("MatchUrl");
        CString Score=m_Robot.getVariable("Score");
        CString Eval=m_Robot.getVariable("Eval");
        CString Locus=m_Robot.getVariable("Locus");
        CString Length=m_Robot.getVariable("Length");
        CString Definition=m_Robot.getVariable("Definition");
        CString Accession=m_Robot.getVariable("Accession");
        CString References=m_Robot.getVariable("References");
        CString Sequence=m_Robot.getVariable("Sequence");
        CString XmlSequence=m_Robot.getVariable("XmlSequence");

        MessageBox(Sequence);
    }
}
```

9) Now your can compile and run your application. You can start and stop the robot using the "Start" and "Stop" buttons. Your program will display a new sequence whenever it is retrieved.

A complete example for MFC 6.0 can be downloaded from http://irobotsoft.com/robots/blastdemo.zip.

4. IRobotX Function API

IRobotX provides the following functions to interact with a robot.

1) long setIrbFile(BSTR irbfilename, BSTR irbpassword)

BSTR irbfilename: The location of the .irb robot file;
BSTR irbpassword: If the robot file is password protected, set the password here;
Returns <0: the error code; >=0: success.

Use this function to open a robot file.

2) long prepareRobot(BSTR run_dir)

BSTR run_dir: A directory for irobot to store temporary files;
Returns <0: the error code; >=0: success.

IRobot may generate some temporary files. Use this function to specify where the temporary files will be stored.

3) long setVariable(BSTR name, BSTR value)

BSTR name: The variable name;
BSTR value: The variable value;
Returns <0: the error code; >=0: success.

Use this function to set variables for the robot.

4) BSTR getVariable(BSTR name)

BSTR name: The variable name:
Returns: the variable value.

Use this function to get variable value from the robot.

5) long run(BSTR task_name)

BSTR task_name: The task name in the robot for run.
Returns <0: the error code; >=0: success.

A robot may include several tasks. Use this function to run a task.

6) long pause()

Returns <0: the error code; >=0: success.

Use this function to pause the robot in execution. Note the robot may not pause immediately after the pause function. You need to set up a timer and call the isStopped() function to detect when the robot is paused. A paused robot can be

resume by the resume() function.

7) long resume()
Returns <0: the error code; >=0: success.

Use this function to resume a paused robot.

8) boolean isStopped()
Returns <0: the error code; >=0: success.

Use this function to test if the robot is stopped. A robot is stopped before running, being paused, or after completed the task.

9) boolean isCompleted()
Returns <0: the error code; >=0: success.

Use this function to test if the robot has completed its execution.

10) void reset()

Use this function to reset a robot. If the robot is currently running, this function will try to pause the robot first. You may need to call another reset() to clear the robot out of memory. The reset function is implicitly called by the setIrbFile() function.

11) long showBrowser(boolean to_show)
boolean to_show: 1: to show the browsers; 0: not to show browsers.
Returns <0: the error code; >=0: success.

A robot can be run in a visible or hidden mode. In the visible mode, browser windows and a control bar are shown. In the hidden mode, nothing is shown to the users.

12) long setLogFile(BSTR logfilename, long logflag)
BSTR logfilename: The file name to log robot actions;
long logflag: Binary flags to indicate which messages to log (use 255 to log all messages);
Returns <0: the error code; >=0: success.

Use this function, you can log the robot actions in a file.

5. IRobotX Event API

IRobotX dispatches several events at run time. You may set up event handlers to process the events.

Even 1t: void AssignVariable(BSTR name, BSTR value)
BSTR name: variable name;
BSTR value: variable value;

The robot fires an AssignVariable event whenever a new variable is assigned. You may use this to monitor the variables in the robot.

Event 2: void Completed()

The robot fires a Completed event when it completed the task.

Event 3: void Stopped ()

The robot fires a Stopped event when it is stopped its execution either by the paused function, or completed the task.

Event 4: void Paused ()

The robot fires a Stopped event when it is stopped by the pause() function.

6. Additional Information

Technical Support: support@irobotsoft.com
Website Support: support@irobotsoft.com
Collaboration: info@irobotsoft.com

Copyright