Interface Documentation

Jiawei Wang

public interface ICallback

{

void sendMessage(Message msg);

}

This interface is used in the Test Harness Server, with which it will output the message by sendMessage(Message msg);

public interface ITestHarness

{

void setClient(IClient client);

void sendTestRequest(Message testRequest);

}

This interface will be called Test Executive Project, which is the main function of Test Server.

With setClient(), Test Harness Server will create a Client object for them, in sendTestRequest(), Test Harness will enqueue the test request.

public interface ITest

{

bool test();

string getLog();

}

ITest is the interface for the Test Driver. Any Test Driver should have the function test() to check if the test is approved or not.

getLog() is the function if we want to output more information, instead of just “passed” or “failed”.

public interface IRepository

{

bool getFiles(string path, string fileList); // fileList is comma separated list of files

List<string> queryLogs(string queryText);

}

IRepository is the interface for repository, and in Project 4, there will be a temporary folder to save the files downloaded from Repository, and then copy the required files to the running folder.

getFiles() can copy files to the running folder.

queryLogs() can query the logs by keywords.

public interface IClient

{

void sendResults(Message result);

}

IClient is the interface in TestHarness.

sendResult() is the function to display the result.

public interface ILoadAndTest

{

ITestResults test(IRequestInfo requestInfo);

void setCallback(ICallback cb);

void loadPath(string path);

}

ILoadAndTest is the interface to load the test. And test() is the function to run the test in AppDomain.

setCallback() pass the parameter of interface ICallback

loadPath() pass the running path.

public interface ITestInfo

{

string testName { get; set; }

List<string> files { get; set; } }

public interface IRequestInfo

{

List<ITestInfo> requestInfo { get; set; } // an array of test compose a request

}

There two interfaces are important for the tests.

ITestInfo is one test, including one test Driver, several test code, and the test name.

IRequestInfo is an array of ITestInfo.

public interface ITestResult //interface for testresult

{

string testName { get; set; }

string testResult { get; set; }

string testLog { get; set; }

}

ITestResult is the result of Test. testName is the name of a test.

testResult and TestLog save the output of the tests.

This interface is just for one test, not one request.

public interface ITestResults

{

string testKey { get; set; }

DateTime dateTime { get; set; }

List<ITestResult> testResults { get; set; }

}

ITestResults is the interface for the results of one request.

testKey is the {Author}{DATE\_TIME} combined in order to identify every test request.

TestResults saves results of one request.

[ServiceContract]

public interface IMessage //IMessage is the interface between Client and TestHarness

{

[OperationContract(IsOneWay = true)]

void PostMessage(TestHarness.Message msg);

// used only locally so not exposed as service method

TestHarness.Message GetMessage();

}

IMessage is the interface of WCF Peer to Peer. With this interface, two processes can get connected. And in the receiver thread, it can wait for GetMessage() and dequeue the Message. In the sender, it can put the message into the BlockingQueue.

This interface is responsible for the communications between clients and Test Harness Server.

[ServiceContract(Namespace = "http://testHarness")]

public interface IStreamService //IStreamService is the interface for the Repository Server

{

[OperationContract(IsOneWay = true)]

void upLoadFile(FileTransferMessage msg);//upload files

[OperationContract]

Stream downLoadFile(string filename);//download files

[OperationContract]

string[] showfiles(string filetype); //show files according to file type

[OperationContract]

void deleteFile(string filename); //delete files according to file name

[OperationContract]

List<string> ClientQuery(string msg);//query the logs according to the keyword.

}

IStreamService is the interface for Repository. In Repository Service, we define the functions in this interface.

With upLoadFile(), Client can upload files to Repository, Test harness Server can upload testing logs to Repository.

With downLoadFile(), Client can download files or logs from Repository, Test Harness Server can download the required files from Repository, then run the test.

With showfiles(), Client can see all the files in the Repository, and check if the files is in repository, or download the files or delete.

With deleteFile(), Client can delete the files in the Repository.

With ClientQuery(), we can pass the key word, and find the matched logs in the Repository.

[MessageContract]

public class FileTransferMessage //this class is for uploading files in IStreamService

{

[MessageHeader(MustUnderstand = true)]

public string filename { get; set; }

[MessageBodyMember(Order = 1)]

public Stream transferStream { get; set; }

}

FileTransferMessage is the interface for the IStreamService, including the filename and the file stream for transferring.

Reference:

1. <http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project4HelpF16/InstrSol-Project2-WithThreads/>
2. <http://ecs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/SocketsAndRemoting/code/WCF_Fawcett_Examples/Peer-Comm-SelfHosted/>
3. <http://ecs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/SocketsAndRemoting/Code/WCF_Fawcett_Examples/FileStreaming/>