ThriftUsageObjectiveC-0.8.0

Getting started

This guide will walk you through creating an application that uses Thrift with a Java server and a client written in Objective-C. It is assumed that you will have a basic knowledge of Objective-C and Xcode to complete this tutorial.

Requirements

Make sure that your system meets the requirements as noted in ThriftRequirements

- Thrift 0.2.0
- Mac OS X 10.5+
- Xcode 3.0+

The following are required for the Java server; but, not the Objective-C client.

- Java 1.4+ (already installed in Mac OS X 10.5+)
- SLF4J (available at http://www.slf4j.org/download.html)

Building Thrift cocoa as a framework

It is relatively straight forward to build the thrift cocoa library sources as a framework. As of Thrift 0.2.0 this is not done automatically. Hopefully it will in future releases.

1. Download Thrift 0.2.0 and install. Follow the instructions at ThriftInstallationMacOSX to install on Mac OS X. 2. Download the Thrift.framework project for XCode ThriftFramework.zip 3. Open the Xcode project file and compile.

Creating the Thrift file

We will use the sample thrift file described on the main page of the Thrift project.

- Create a new directory to contain the project sources.
- Create a new file called 'profile.thrift'

```
struct UserProfile {
  1: i32 uid,
  2: string name,
  3: string blurb
}
service UserStorage {
  void store(1: UserProfile user),
  UserProfile retrieve(1: i32 uid)
}
```

- Run the thrift compiler to build project sources.

```
thrift --gen java profile.thrift
thrift --gen cocoa profile.thrift
```

Creating the Java Server

- cd into the gen-java directory
- make sure that your classpath is properly setup. You will need to ensure that libthrift.jar, slf4j-api, and slf4j-simple are in your classpath.
- create the file UserStorageImpl.java

```
import org.apache.thrift.TException;
import java.util.HashMap;

class UserStorageImpl implements UserStorage.Iface {
    private HashMap<Integer, UserProfile> profiles;

   public UserStorageImpl() {
       profiles = new HashMap<Integer, UserProfile>();
   }

   @Override
   public void store(UserProfile user) throws TException {
       long time = System.currentTimeMillis();
       System.out.println("store() called: " + time);
       System.out.println("UID: " + user.uid);
       System.out.println("Name: " + user.name);
       System.out.println("Blurb: " + user.blurb);
       profiles.put(new Integer(user.uid), user);
    }

    @Override
    public UserProfile retrieve(int uid) throws TException {
        return profiles.get(new Integer(uid));
    }
}
```

- create a file called 'Server.java'

```
import java.io.IOException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.protocol.TBinaryProtocol.Factory;
import org.apache.thrift.server.TServer;
import org.apache.thrift.server.TThreadPoolServer;
import org.apache.thrift.transport.TServerSocket;
import org.apache.thrift.transport.TTransportException;

public class Server {

    private void start() {
        try {
            TServerSocket serverTransport = new TServerSocket(7911);
            UserStorage.Processor processor = new UserStorage.Processor(new UserStorageImpl());
            Factory protFactory = new TBinaryProtocol.Factory(true, true);

            TServer server = new TThreadPoolServer(processor, serverTransport, protFactory);

            System.out.println("Starting server on port 7911 ...");
            server.serve();
        } catch (TTransportException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String args[]) {
        Server srv = new Server();
        srv.start();
    }
}
```

- compile the classes

```
javac *.java
```

- run the server

```
java Server
```

- This will run a server that implements the service UserStorage on port 7911. The service simply stores UserProfile structures in a HashMap and retrieves them when requested.

Create the Objective-C client

The objective-c client is a simple cocoa application that allows the user to fill out the values of a UserProfiel structure and store/retrieve them from the server.

- Create a new cocoa project.
- Right click on the classes group and Add existing files. Browse to the gen-cocoa directory and add the files profile.h and profile.m
- Change the import lines in profile.h and profile.m from:

```
#import <TProtocol.h>
#import <TApplicationException.h>
#import <TProtocolUtil.h>
#import <TProcessor.h>
```

to

```
#import <Thrift/TProtocol.h>
#import <Thrift/TApplicationException.h>
#import <Thrift/TProtocolUtil.h>
#import <Thrift/TProcessor.h>
```

- Copy the following text to the app delegate header file

```
#import <Cocoa/Cocoa.h>

#import <Thrift/TSocketClient.h>
#import <Thrift/TBinaryProtocol.h>

#import "profile.h"

@interface ThriftCocoaAppDelegate : NSObject <NSApplicationDelegate> {
  NSWindow *window;
  NSTextField *uid;
  NSTextField *name;
  NSTextField *blurb;

  TSocketClient *transport;
  TBinaryProtocol *protocol;
  UserStorageClient *service;
}

@property (assign) IBOutlet NSWindow *window;
@property (assign) IBOutlet NSTextField *uid;
@property (assign) IBOutlet NSTextField *name;
@property (assign) IBOutlet NSTextField *blurb;

- (IBAction)store:(id)sender; // call thrift service store function.
- (IBAction)retrieve:(id)sender; // call thrift service retrieve function.

@end
```

5. Open the XIB file in Interface builder and drag out 3 NSTextField objects. Connect these to the uid, name, and blurb outlets.
6. Drag out 2 buttons. Label the first as 'Store' and connect it to the store action. Label the other as retrieve and connect it to the retrieve action.
7. Copy the following code into the app delegate implementation file (the .m file)

```
#import "ThriftCocoaAppDelegate.h"
#import "profile.h"


@implementation ThriftCocoaAppDelegate

@synthesize window;
@synthesize uid;
@synthesize name;
@synthesize blurb;

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
  // Talk to a server via TCP sockets, using a binary protocol
  transport = [[TSocketClient alloc] initWithHostname:@"localhost"
                                                 port:7911];
  protocol = [[TBinaryProtocol alloc] initWithTransport:transport
                                            strictRead:YES
                                           strictWrite:YES];

  // Use the service defined in profile.thrift
  service = [[UserStorageClient alloc] initWithProtocol:protocol];
}

- (void)store:(id)sender {
  NSLog(@"Called store.");
  // Make an object
  UserProfile *up = [[UserProfile alloc] initWithUid:[[uid stringValue] intValue]
                                              name:[name stringValue]
                                             blurb:[blurb stringValue]];

  // call the store function on the service
  [service store:up];
}

- (void)retrieve:(id)sender {
  NSLog(@"Called retrieve.");
  // Retrieve something as well
  UserProfile *up = [service retrieve:[[uid stringValue] intValue]];
  NSLog(@"Received User Profile: %@", up);
  [uid setStringValue:[NSString stringWithFormat:@"%d", [up uid]]];
  [name setStringValue:[up name]];
  [blurb setStringValue:[up blurb]];
}

@end
```

- This code creates a new transport object that connects to localhost:7911, it then creates a protocol using the strict read and strict write settings. These are important as the java server has strictRead off and strictWrite On by default. In the cocoa app they are both off by default. If you omit these parameters the two objects will not be able to communicate. We then create a UserStorage service.
- If the user presses the store button a new UserProfile object is created with the values in the text field and the store function is called on the service.
- If the user presses the retrieve button the uid is read and retrieve is called. If a result is returned the values are unpacked into the GUI.
- The last thing you need to do is add the Thrift framework to your project. Right click on the Frameworks->Linked Frameworks group and add existing frameworks. Select 'add other' and locate the compiled Thrift.framework on disk (this is under the build directory of the project you downloaded and compiled earlier).
- Under target right click and Add New Build Phase->New copy file build phase. Change the destination to Frameworks and press CTRL+click and drag the Thrift.framework object to the copy files build phase. Make sure that the Thrift.framework object is copied out of the previous area ... not moved.
- At this point you should be able to compile the code and run it.

You can download the Xcode project for this step: ThriftObjectiveCClient.zip