Creating an application for LaCOLLA 1.0.X

Author: zperise@uoc.edu
Date: 03/03/08

Introduction

This document provides a few guidelines for building an application for LaCOLLA from the scratch. In order to keep it simple, some details are skipped.
It is assumed that users following this document must have a basic knowledge of Java and LaCOLLA inner mechanisms.
For a completely developed application, refer to the putGo application on the web site.

Application structure

The application has two sides with respect to LaCOLLA. One is responsible of sending information to other nodes and the other one is responsible of receiving this information from other nodes.

Implementation

A small application will be built for demonstration purposes consisting only in two classes: **`TlcApp`** i **`AppSideApi`**. The former is the application itself and the latter receives the events happening in the group and implements the **`ApplicationsSideApi`** interface.

Initialization and login

The first step of an application over LaCOLLA is to log in into the system. We need to know where a GAPA component is (IP address and Port) in order to authenticate the application on the system. We need, as well, to know a UA component in order to be able to communicate with other nodes.
First code snippet shows how initialization is carried out. It can be summarized as follows:

1. Initialize object for receiving callbacks
2. Retrieve LaCOLLA API
3. Log into LaCOLLA
4. Wait to be correctly logged in.

On the second and third code snippet, we can see the code for initializing functions.

1. Resolve Api: Retrieves an object of LaCOLLA for invoking commands on the group.
2. initialize: Initializes and publishes an object for receiving the callbacks from the system.

```
System.out.println("Initializing");
System.out.println("HOST: "+AppIP+":"+AppPort);

// Object for getting callbacks
AppSideApi asa = initialize();
if (asa==null) return;

// Object for performing actions in the group
Api api = resolveApi(ApiPort);

try {
	// login into LaColla
	// Application ID is provided by LaCOLLA
	String apiid = api.login(group, username, pwd, GAPAIP,
		GAPAPort, AppIP, AppPort);
	System.out.println("Logged IN: "+apiid);
	int count =0;
	// Wait for MemberId. It will be provided by LaCOLLA
	// There may be some delay when receiving it.
	while (asa.getMemberId()== null){
		try {
			Thread.sleep(1000);
		} catch (InterruptedException e) {}
		System.out.println("Waiting for MemberId ["+count+"]");
		count++;
	}

	// Retrieve memberId
	memberId = asa.getMemberId();
	System.out.println("MemberId: "+memberId);

} catch (RemoteException e) {
	e.printStackTrace();
}
```

```
private Api resolveApi() {
	Api api = null;
	try {
		// Get the Api object from the registry
		Registry reg = LocateRegistry.getRegistry(RMIHost,
			RMIPort);
		api = (Api) reg.lookup(
			constant.DEFAULT_API_LOCATION+ApiPort);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return api;
	}
```

```
private AppSideApi initialize() {
	try {
		// Create an AppSideApi object
		AppSideApi aplicapi = new AppSideApi();
		System.out.println("AppSideApi created");

		// Publish it on the registry
		Registry registry = LocateRegistry.getRegistry(RMIHost,
			RMIPort);
		registry.rebind(constant.DEFAULT_APSAPI_LOCATION +
			AppPort, aplicapi);
		System.out.println("AppSideApi published: "+
			constant.DEFAULT_APSAPI_LOCATION+AppPort);

		return aplicapi;
	} catch (RemoteException e) {
		e.printStackTrace();
	}
	// On error, return null
	return null;
}
```

The Callback object

This object will be used to allow communication between the middleware and the application. It must implement the **`ApplicationsSideApi`** interface. This class implements the behavior of the application when an event is received. Refer to the javadoc to know more about the meaning of each event.
Next code snippet must be added in order to get the memberId when the application is connected to the group

```
public void newConnectedMember(String groupId, String userId, String
		memberId) throws RemoteException {
	this.memberId=memberId;
}
```

Calling some functionalities from LaCOLLA

The application can use the LaCOLLA object retrieved at the initialization function in order to communicate with the other nodes in the system. The next code snippet shows how an event is generated and disseminated over the system. It will be received by all nodes through the callback object.

```
LaColla.core.data.app.Event evt= new LaColla.core.data.app.Event();

api.disseminateEvent(group, evt);
```

Complete working example

Here you can find a complete example. Some parameters must be edited according to your system setup, i.e. ports, usernames and password. You may find them on the main() method.

TlcApp.java

```
package edu.uoc.tlc;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import LaColla.Api.Api;
import LaColla.core.util.constant;

public class TlcApp {

	public final int RMIPort = 11099;
	public String RMIHost;
	private int AppPort;
	private int ApiPort;
	private int GAPAPort;
	private String AppIP;
	private String ApiIP;
	private String GAPAIP;
	private String username;
	private String pwd;
	private String group;
	private String memberId;

	public static void main(String[] args) {
		TlcApp a = new TlcApp();
		String currentIP = null;
		try {
			currentIP = InetAddress.getLocalHost().
				getHostAddress();
		} catch (UnknownHostException e) {
			e.printStackTrace();
			return;
		}
		a.setApiIP(currentIP);
		a.setAppIP(currentIP);
		a.setGAPAIP(currentIP);
		a.setRMIHost(currentIP);

		a.setApiPort(19000);
		a.setGAPAPort(2000);
		a.setAppPort(445546);

		a.setGroup("hola");
		a.setUsername("xavo");
		a.setPwd("lipa");

		a.run();


	}
```

```
public void run() {

	System.out.println("Initializing");
	System.out.println("HOST: "+AppIP+":"+AppPort);

	// Object for getting callbacks
	AppSideApi asa = initialize();
	if (asa==null) return;

	// Object for performing actions in the group
	Api api = resolveApi(ApiPort);

	try {
		// login into LaColla
		// Application ID is provided by LaCOLLA
		String apiid = api.login(group, username, pwd,
				GAPAIP, GAPAPort,AppIP, AppPort);
		System.out.println("Logged IN: "+apiid);
		int count =0;
		// Wait for MemberId. It will be provided by LaCOLLA
		// There may be some delay when receiving it.
		while (asa.getMemberId()== null){
			try {
				Thread.sleep(1000);
			} catch (InterruptedException e) {}
			System.out.println("Waiting for MemberId ["+
					count +"]");
			count++;
		}
		// Retrieve memberId
		memberId = asa.getMemberId();
		System.out.println("MemberId: "+memberId);

		LaColla.core.data.app.Event evt= new
				LaColla.core.data.app.Event();

		api.disseminateEvent(group, evt);

		// Some delay for receiving events
		try {
			Thread.sleep(60000);
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		// Logout from LaCOLLA
		api.logout(group, memberId, apiid);

	} catch (RemoteException e) {
		e.printStackTrace();
	}


	System.out.println("END");

}
```

```
	private Api resolveApi(int apiPort2) {
		Api api = null;
		try {
			// Get the Api object from the registry
			Registry reg = LocateRegistry.getRegistry(
					this.RMIHost, this.RMIPort);
			api = (Api) reg.lookup(
					constant.DEFAULT_API_LOCATION+ApiPort);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return api;
	}


	private AppSideApi initialize() {
		try {
			// Create an AppSideApi object
			AppSideApi aplicapi = new AppSideApi();
			System.out.println("AppSideApi created");

			// Publish it on the registry
			Registry registry = LocateRegistry.getRegistry(
					RMIHost, RMIPort);
			registry.rebind(
					constant.DEFAULT_APSAPI_LOCATION +AppPort,
					aplicapi);
			System.out.println("AppSideApi published: "+
					constant.DEFAULT_APSAPI_LOCATION +AppPort);

			return aplicapi;
		} catch (RemoteException e) {
			e.printStackTrace();
		}
		// On error, return null
		return null;
	}

	public void setApiIP(String apiIP) { ApiIP = apiIP; }

	public void setApiPort(int apiPort) { ApiPort = apiPort; }

	public void setAppIP(String appIP) { AppIP = appIP; }

	public void setAppPort(int appPort) { AppPort = appPort; }

	public void setGAPAIP(String gapaip) { GAPAIP = gapaip; }

	public void setGAPAPort(int port) { GAPAPort = port; }

	public void setGroup(String group) { this.group = group; }

	public void setPwd(String pwd) { this.pwd = pwd; }

	public void setUsername(String username) { this.username =
			username; }

	public void setRMIHost(String host) { RMIHost = host; }
}
```

AppSideApi.java

```
package edu.uoc.tlc;

import java.rmi.RemoteException;
import LaColla.Api.ApplicationsSideApi;
import LaColla.core.data.Event;
import LaColla.core.data.GroupInfo;

public class AppSideApi extends java.rmi.server.UnicastRemoteObject
                                        implements ApplicationsSideApi {
      private String memberId;
      protected AppSideApi() throws RemoteException {
            super();
            // TODO Auto-generated constructor stub
      }
      public boolean AppIsAlive(String arg0) throws RemoteException {
            // TODO Auto-generated method stub
            return true;
      }
      public void exception(String arg0, String arg1) throws
                                                      RemoteException {
            // TODO Auto-generated method stub
      }
      public void memberDisconnected(String arg0, String arg1) throws
                                                      RemoteException {
            // TODO Auto-generated method stub
      }
      public void newConnectedMember(String groupId, String userId,
                              String memberId) throws RemoteException {
            this.memberId=memberId;
            System.out.println("newConnectedMember "+groupId+" "+
                                                  userId+" "+memberId);
      }
      public void newEvent(String arg0, Event arg1) throws
                                                      RemoteException {
            // TODO Auto-generated method stub
            System.out.println(arg1);
      }
      public void newInfoGroup(String arg0, String arg1, String arg2,
                               GroupInfo arg3) throws RemoteException {
            // TODO Auto-generated method stub
      }
      public void newInstantMsg(String arg0, String arg1, String arg2,
                                  Object arg3) throws RemoteException {
            // TODO Auto-generated method stub
      }
      public void notifyStopTask(String arg0, String arg1, String
                                          arg2) throws RemoteException {
            // TODO Auto-generated method stub
      }
      public void notifyTaskState(String arg0, String arg1, String
                             arg2, String arg3) throws RemoteException {
            // TODO Auto-generated method stub
      }
      public String getMemberId() { return memberId; }
}
```