**Team Members:**

* Gaurab Dey, UFID: 78029140, Email: [gaurabde@ufl.edu](mailto:gaurabde@ufl.edu), Contact: 9727691170
* Sneha Durvasula , UFID:99442386 ,Email: [snehadurvasula@ufl.edu](mailto:gaurabde@ufl.edu),Contact: 9787273036

**What is Working:**

* Process Flow log details
* Error Simulation and log details
* Log of any Process

**How you Integrated logging with the Actor**

All Actors **inherit the LogHandler** Class and use it log independently. XML is used as base for Regular/Error logging.

Naming of Log Files:

1. Each class/Actor contains its own Regular Log (for Rollback point) and Error Log file
2. Naming convention:

**<ActorReference>.xml** --- Regular Log file

**<ActorReference>\_error.xml** – Error Log file

1. Location details:

Logs related to Boss Actor **“./BossLog/”** folder under project structure

Logs related for Workers or Processes **“./Logs/”** folder under project structure

**Program Details**

**No external JAR file added for Log file creation. Log files are XML based for easy data retrieval and better readability of logs**.

**Structure Details (Methods/Actors):**

1. **example.scala**

* **example:** main object used for starting the simulation, by creating the NodeManager. Its automated and independent of user input for execution
* **NodeManager(ref:Any,no:Int):** NodeManager will create all the process and will monitor the process flows. Randomly 5 processes are trigged from NodeManager and the Flow are recoded in both NodeManager Logs (./BossLog/<NodeManager>.xml) and the Process Nodes Logs (./Logs/<actorLog>.xml)
  + **ref:** references of the root
  + **no:** number of process to be created

1. **actorLog.scala**

* **actorLog(id:Int,bossRef:Actor,size:Int):** used as independent process with self-logging capability.
  + **writeToFile(fileName:String, data:String):** it will erase the content of a particular file and add the new contents (data) in it.
  + **appendToFile(fileName:String, data:String):** it will add the new element into the log file as per the specification in either Regular Log or Error Log.

1. **CaseClass.scala**

Contain all the common cases class shared within the projects among different classes.

1. **LogHandler.scala:**
   1. **writeToFile(fileName:String,** [**data:String**](data:String)**):** it will erase content of any file and add the data string in it.
   2. **appendToFile(fileName:String, textData:String):** it will edit the content of the file and merge the new nodes with the existing data.
   3. **prepBossLog(state:Int,key:String,**[**cId:Int,nId:Int,desp:String):** it](cid:Int,nId:Int,desp:String):it) is used to prepare the xml content of the log files for any particular case for the **Boss (NodeManager**). Depending upon the condition it can generate a ERROR log content or Normal Log content.
   4. **prepNodeLog(state:Int,key:String,**[**cId:Int,nId:Int,desp:String):** it](cid:Int,nId:Int,desp:String):it) is used to prepare the xml content of the log files for any particular case for **Processes (actorLog)**. Depending upon the condition it can generate a ERROR log content or Normal Log content.
   5. **printBossLog(status:Int,key:String):** it is used to print the content of log files in case of Back Tracking as well as Error issues for Boss (NodeManager).
   6. **printNodeLog(status:Int,key:String):** it is used to print the content of log files in case of Back Tracking as well as Error issues for any Process (actorLog).
   7. **addToLog(newNode: Node,root:Node):** it is used to edit the content of log file. Mainly used to add the new log element into the log file content.
   8. **addToLog(newNode:Node,root:Node):** it is used to edit the content into Error log file. Mainly used to add the new log element into the Error Log file contents.

**Working Details:**

1. example class first clear all the log files from “./BossLog/” and “./Logs/” folders to avoid any space issues. Logs are kept Runtime.
2. NodeManager is created from the main process (**example**), number of process threads to be created and the reference of the base is passed while the NodeManager is created.
3. Once the NodeManger is triggered to **Start** it will create number of independent processes according the variable passed from base.
4. All the relative Log details are added in Regular Log file in **./BossLog/<NodeManager>.xml**.
5. After individual process creation NodeManager triggers FIVE independent process flow by initiating and Random process (actorLog) to start the Process Flow.
6. Once the process flow is started, each selected Process (actorLog) will randomly select any other Process for transmitting the updated Message to it. All the changes in the variable of the Process are Logged in its Regular Log file in “**./Logs/<actorLog>.xml**” location.
7. When the message sharing is complete (5 round of message sharing is simulated in current code), it will enlist all the Process IDs undergone changes during Process Flow. **This ID can be used for better log details from Regular/Error Log files.**

List of Process undergoing changes, use IDs for reference

---------------------------

Message Changed in Node: 1

Message Changed in Node: 3

Message Changed in Node: 16

Message Changed in Node: 12

Message Changed in Node: 26

Message Changed in Node: 42

Message Changed in Node: 39

Message Changed in Node: 9

Message Changed in Node: 44

Message Changed in Node: 5

Message Changed in Node: 5

Message Changed in Node: 42

Message Changed in Node: 29

Message Changed in Node: 19

Message Changed in Node: 6

Message Changed in Node: 15

Message Changed in Node: 43

Message Changed in Node: 30

Message Changed in Node: 37

Message Changed in Node: 29

1. It will also enlist all the Process Key once the Process Flow are complete, which act as ID for each flow (unique).

Process flow completed (Key): actorLog@32079df9

Process flow completed (Key): actorLog@5eace9d4

Process flow completed (Key): actorLog@80059f5

Process flow completed (Key): actorLog@d43ccf5

Process flow completed (Key): actorLog@9e76345

1. Once the whole Process Flow is complete it will prompt user with following options:

Distributed Debugging Simulation

================================

1. Process Flow log details

2. Error Simulation and log details

3. Log of any Node

4. Exit

Enter your choice:

1. Selection Details:
   1. **Process Flow log details:** there are various random flows in the system, so it will give option to the user to select any id from the List of Process Keys for **generating the Flow details from the logs**. Following is example:

Process Key List:

1. actorLog@9e76345

2. actorLog@5eace9d4

3. actorLog@32079df9

4. actorLog@d43ccf5

5. actorLog@80059f5

Enter Process Key ID (1-5):

**4**

========================

Process Flow log details

========================

Boss Log Directory

===============

Process Flow details with Key: actorLog@d43ccf5

Root Process ID:28

Message Passed: 28

-----------

Source Process ID: 28

Destination Process ID: 26

Message Passed: 28

-----------

Source Process ID: 26

Destination Process ID: 5

Message Passed: 28

-----------

Source Process ID: 5

Destination Process ID: 19

Message Passed: 28

-----------

Source Process ID: 19

Destination Process ID: 37

Message Passed: 28

-----------

Source Process ID: 37

Destination Process ID: 44

Message Passed: 28

-----------

* 1. **Error Simulation and Log details:** this option will simulate Error cases by terminating any Random Process and generating error Logs for the same. Once the node is terminated it will trigger the error logger and print the status of all the changes of the variables and the status of the process with the Process Key details so that it can be related to any process flow.

Enter your choice:

**2**

========================

Error Simulation and log details

========================

Termination Random Node ID: 43 actorLog@5eace9d4

Accessing Error Logs.

....Error Log for Node: 43

Process Key: actorLog@5eace9d4

Source Process ID: 43

Destination Process ID: 0

Current Message Status-- 13

-----------

Message History

------------

Process ID: 43 Ref Code: actorLog@4c9bbed7

===============

Process Key: actorLog@5eace9d4

Source Process ID: 42

Destination Process ID: 43

Message Changes-- old: 43 new: 13

-----------

Process Key: actorLog@5eace9d4

Source Process ID: 43

Destination Process ID: 0

Message Changes-- old: 43 new: 13

-----------

Status of the Process: Terminated

-------------

* 1. **Log of any Process:** this is used to Log trace of any Process with the ID of the process. For better Log details it is suggested to enter ID of Process undergone changes during the process flow. The list of processes undergone changes is enlisted when the program starts (Section 6). Example of the execution:

Enter your choice:

3

Please Enter Id from the above changed Message List for better log details

Enter Process ID (1-50):

15

Accessing Process ID Logs.

....========================

Process ID: 15 Ref Code: actorLog@116e3a7e

===============

Process Key: actorLog@32079df9

Source Process ID: 5

Destination Process ID: 15

Message Changes-- old: 15 new: 22

-----------

Process Key: actorLog@32079df9

Source Process ID: 15

Destination Process ID: 16

Message Changes-- old: 15 new: 22

-----------