Asynchronous FMGProcess Web services

4/12/2009

CGL

Jun Ji

In this asynchronous web services, we planned a 'ticket returning' model to avoid waiting a result of the submitted job. This model is implemented by 3 parts: submit(), getstatus(), getresult().

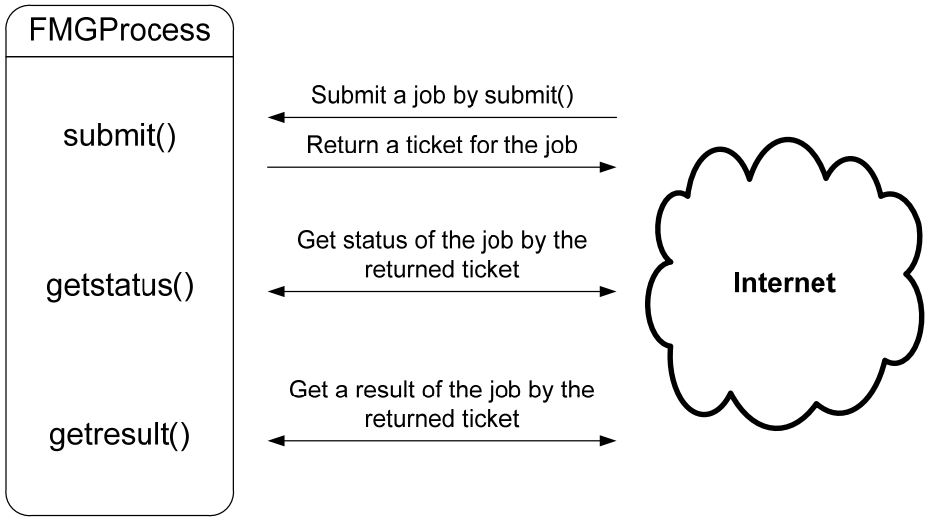

Figure 1 Ticket returning model

This model returns a ticket immediately as soon as a job is submitted to allow clients to track the job's result with the ticket later. It means the clients don't have to wait until the web service makes a result. Following Figure 2, 3 shows this advantage.

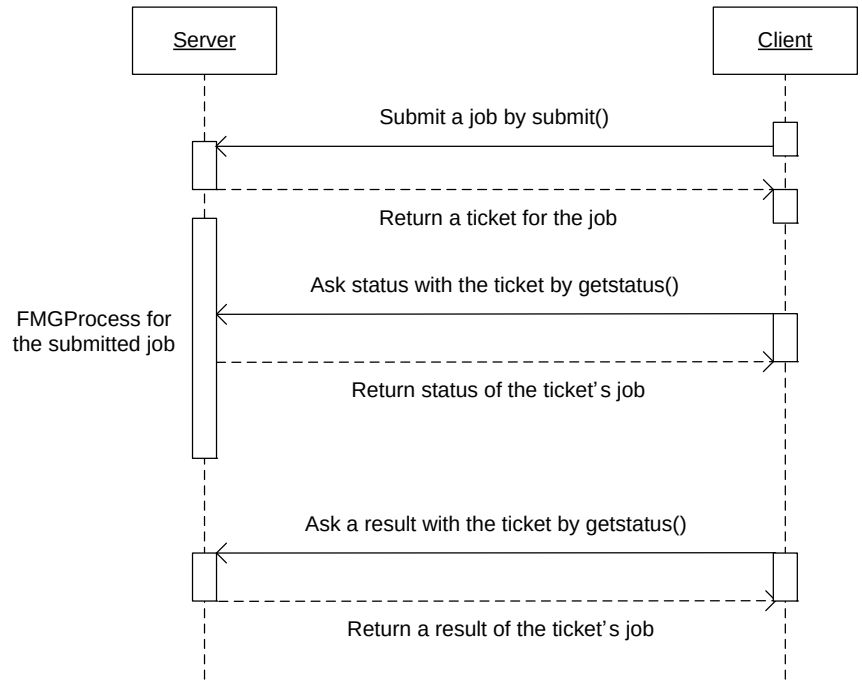

Figure 2 Asynchronous FMGProcess Web services

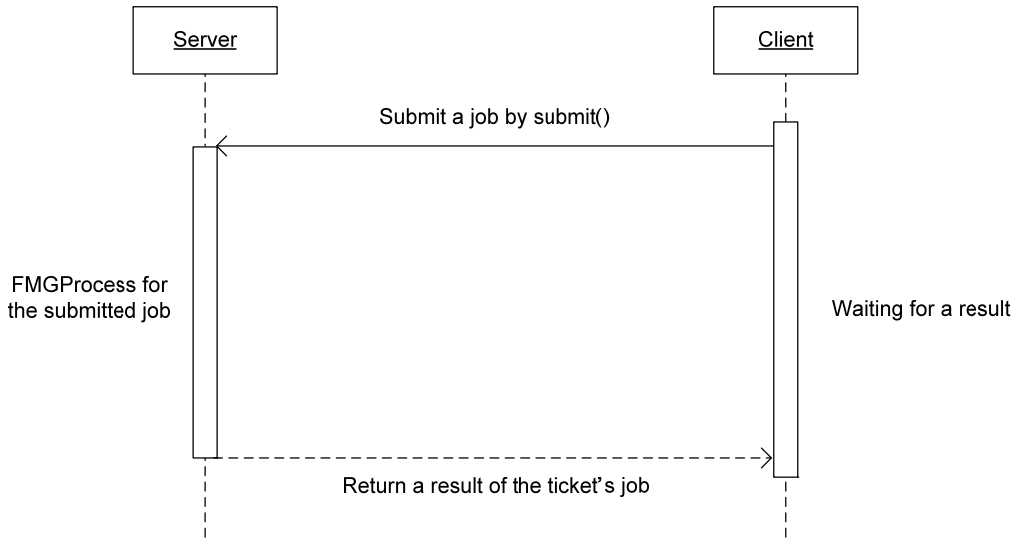

Figure 3 Synchronous FMGProcess Web services

Ticket returning model

As you can see in the Figure 1, this model is achieved by 3 methods: submit(), getstatus(), getresult(). When a job is sent, it decides the job's validity and makes a ticket. After that, executes a main process as a thread to return the job's ticket immediately. Each ticket will be generated as unique string by each thread. But, because the each thread will not have any shared memory, the ticket information should not be in memory, but should exist in a stable storage as like files. For this fact, the ticket string will be also used as the ticket job's directory name to pair them easily.

[submit]

A client will propose a job by this method. It will judge the job's validity by checking the requested input file's URL and its file name. If the URL is valid and does have a correct file (.cgn), the job will be successfully accepted and a client will receive a ticket of the job.

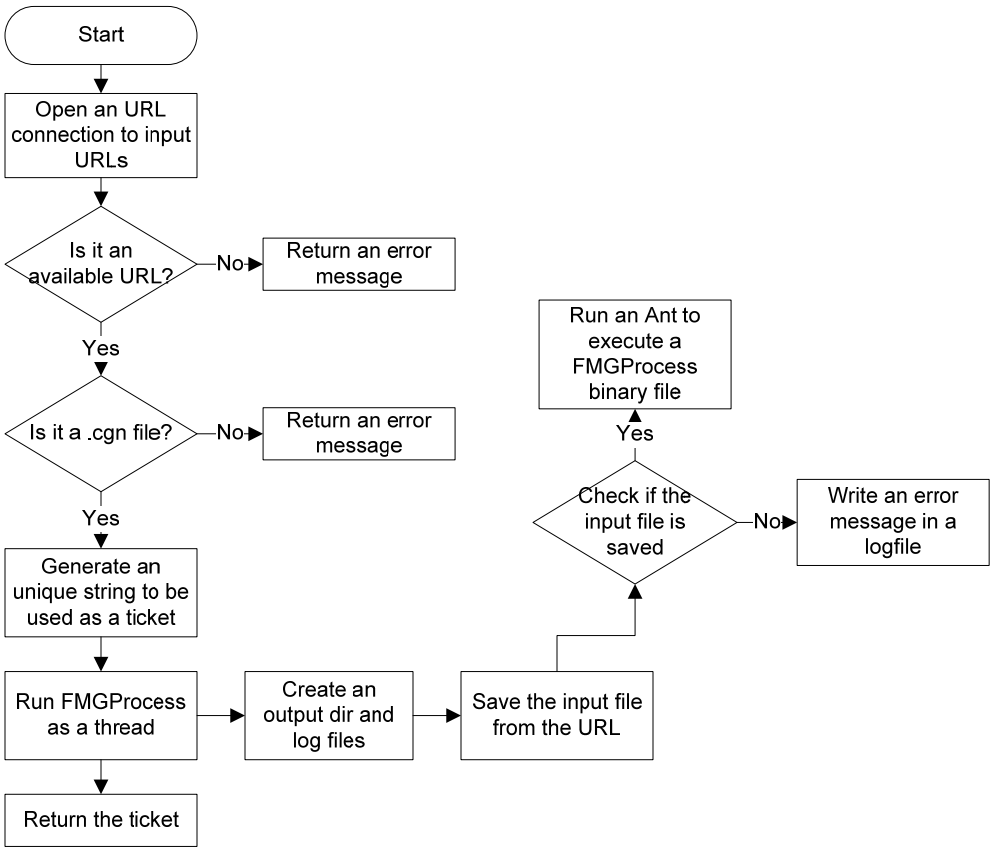

Figure 4 submit() flowchart

[getstatus]

This method checks status of a job. Because the main FMGProcess records transactions to each job's log file according each step, the log file can be used to know its progress. If the job was finished successfully, the last line of log file ends by 'finished'.

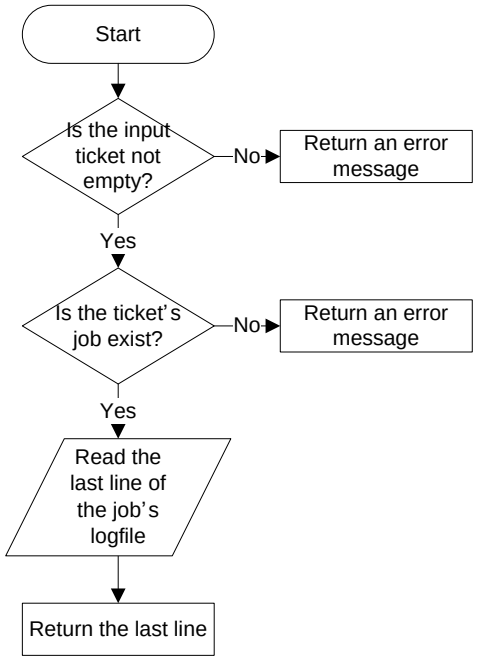

Figure 5 getstatus() flowchart

[getresult]

A client uses this method to receive a result of the submitted job. It runs getstatus() and if the job was finished, generates the output files URL and return them.

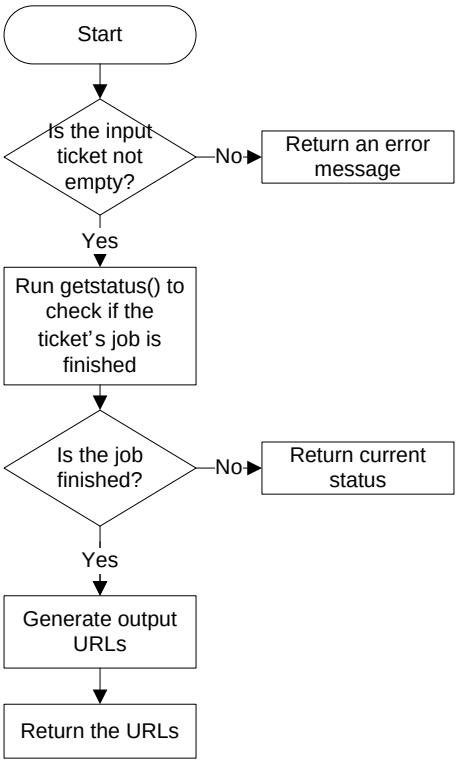

Figure 6 getresult flowchart

[Implementation and packaging]

This asynchronous web services model has been implemented by JAVA JDK 6, MAVEN 2.1.0, TOMCAT 6.0.18, AXIS2 1.4.1 (Also, I used eclipse 3.4.2 Ganymede).

Each web service has one Java file which consists of two classes.

[FMGProcessOutput]

It contains the 3 methods: submit(), getstatus(), getresult() which will be exposed by webservice. Another method getmyip() is used to write URLs of output files.

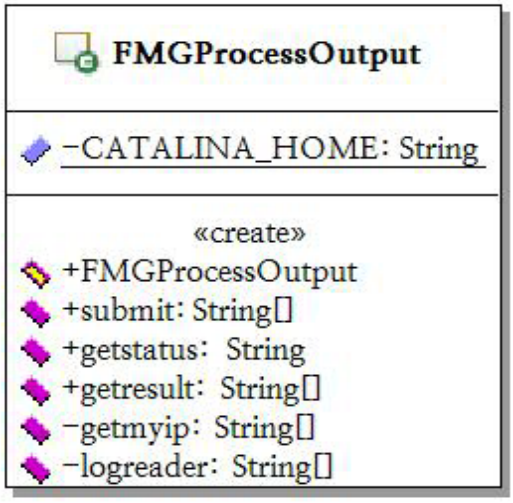

Figure 7 class FMGProcessOutput

[RunFMGProcessOutput]

This class extends a class Thread. It does main process of each FMGProcess (Currently, we have two kinds of FMGProcess: FMGProcessOutput, FMGProcessInput).

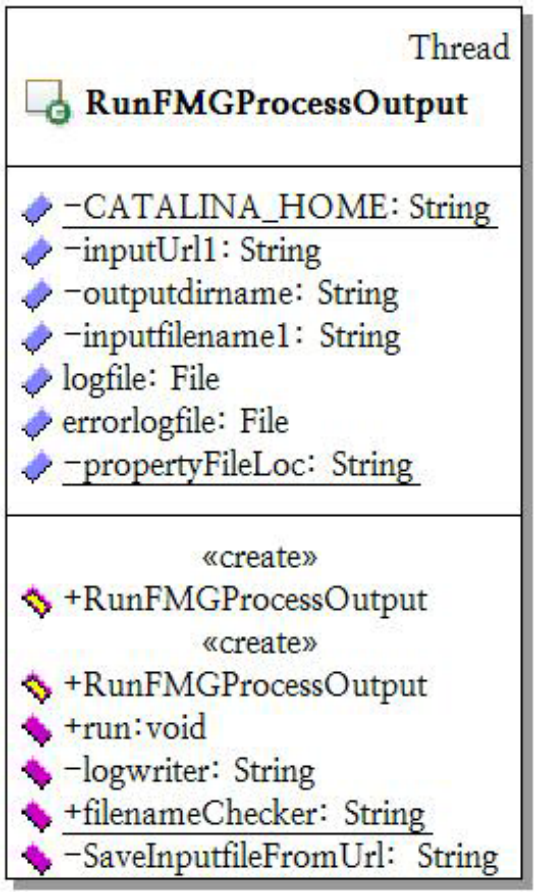

Figure 8 class RunFMGProcessOutput

In a method run(), it executes a binary file to make outputs and to do this, this class uses Ant APIs such like setExecutable(), execute(). The setExecutable(String value) sets the target file to be executed.

This webservice is distributed with a property file which has a location of the binary file which will be read by the method run() when this method should find the binary file to execute it.

[Testing]

To test the implementation, we use JUnit and when JUnit tests, tested methods temporally should be modified to have public property. Also, other codes which exceptionally should be edited will be referred in the related test codes.

* The test codes are based on each method's flowchart to make every possible case can happen.

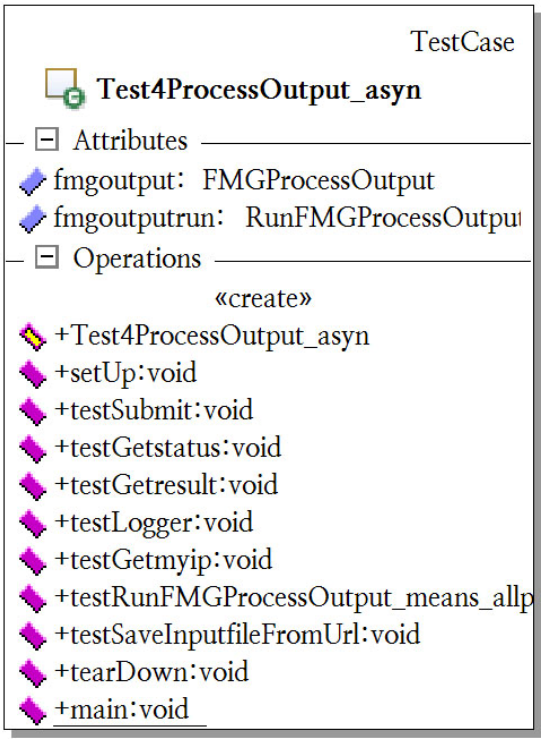

Figure 9 class Test4ProcessOutput_asyn

[testSubmit]

- If it accepts null or wrong input URLs.

[testGetstatus, testGetresult]

- If it accepts null or not existing tickets.

[testLogger]

- If it writes and reads correctly.

[testGetmyip]

- If it reads an exact IP address.

[testGetmyip]

- If it reads an exact IP address.

[testRunFMGProcessOutput_means_allprocess]

** Every time before this method, submit() will check validity of input values. So this method can be free from the risk.

- It gets correct input values to run whole process and check if it makes a correct result.

[testSaveInputfileFromUrl]

- It receives wrong input values to check if it can detect the wrong, and check if it can read a file correctly from correct input URLs.

[Install]

This package deploys maven 2.1.0 to provide one-step installation process for all of services.

(FMGProcessOutput_asyn, FMGProcessInput_asyn, FMGRealTimePredication)

** The system should have a correct system variable $CATALINA_HOME and Axis2 should be located in ${env.CATALINA_HOME}/webapps/axis2.

1. Download the service package.

   svn co http://crisisgrid.svn.sourceforge.net/svnroot/crisisgrid/FloodGrid/FMGProcess

2. Unpack and move to the FMGProcess directory to compile it.

   cd FMGProcess

   mvn compile

3. Check if the aar files are moved to ${env.CATALINA_HOME}/webapps/axis2/WEB-INF/services

4. Both asynchronous webservice have a test code. To use this, every method property of each webservice should be changed to 'public'. If there is other instruction, it might be written in the test code file by commentaries.

   mvn test or mvn clean install