Explication du fonctionnement de l'interface IEngineRunner et des classes associees

==========

Version francaise

L'interface IEngineRunner permet de definir un moteur d'execution generique pour un traitement structure en plusieurs etapes (steps). L'implementation EquityEngineRunner orchestre l'execution de ces etapes de maniere dynamique a partir d'un plan d'execution.

1. Classe Launcher :

- Initialise le contexte Spring (AnnotationConfigApplicationContext)
- Recupere les beans Spring (SparkSession, EngineRunner, etc.)
- Execute le moteur via engineRunner.runEngine(...)

2. Classe EquityEngineRunner (implemente IEngineRunner) :

- Injecte les dependances Spring necessaires (ApplicationContext, SparkContext, etc.)
- Construit dynamiquement la configuration d'execution via la methode getRunConfigurationEngine(...)
- Enchaine les steps dans l'ordre defini, avec nom, type et parametres.

3. getRunConfigurationEngine(...):

- Cree dynamiquement une instance Step pour chaque etape configuree
- Renseigne nom, type, moteur, parametres
- Regroupe les etapes dans une Map<String, List<Step>> injectee dans le bean principal Run

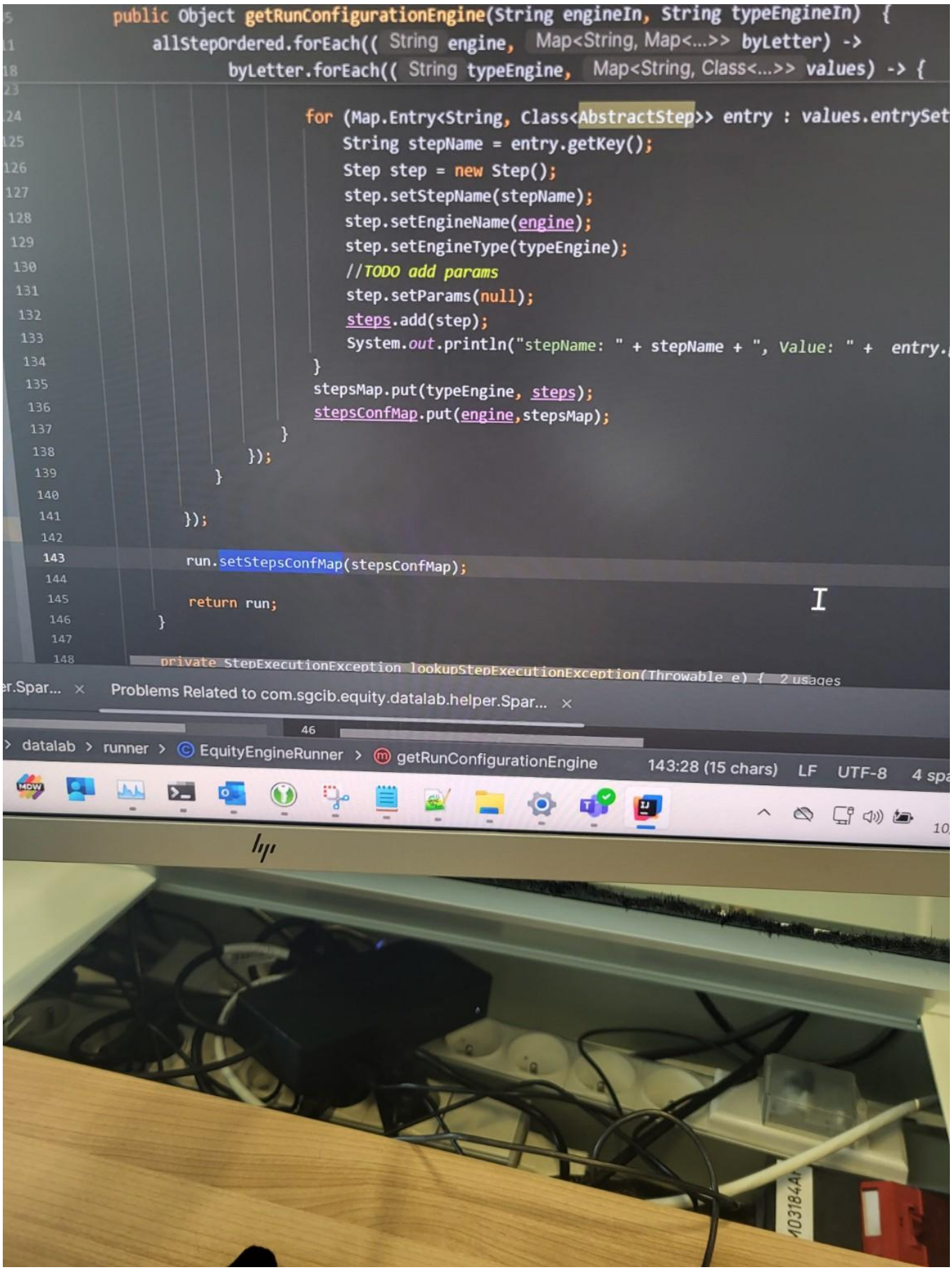
English version

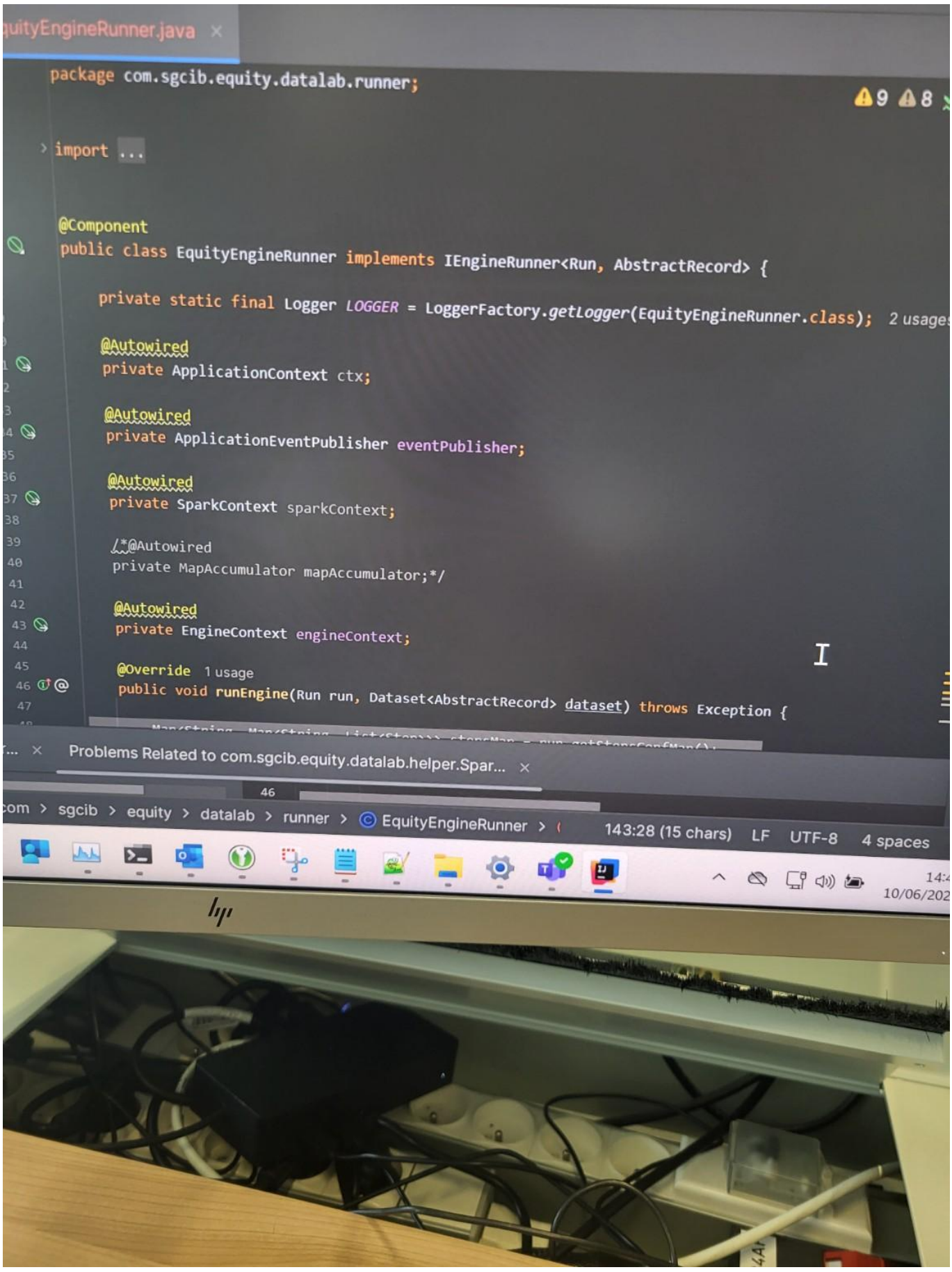
The IEngineRunner interface defines a generic execution engine to process structured workflows composed of multiple steps. The EquityEngineRunner class implements this logic and dynamically orchestrates the configured steps.

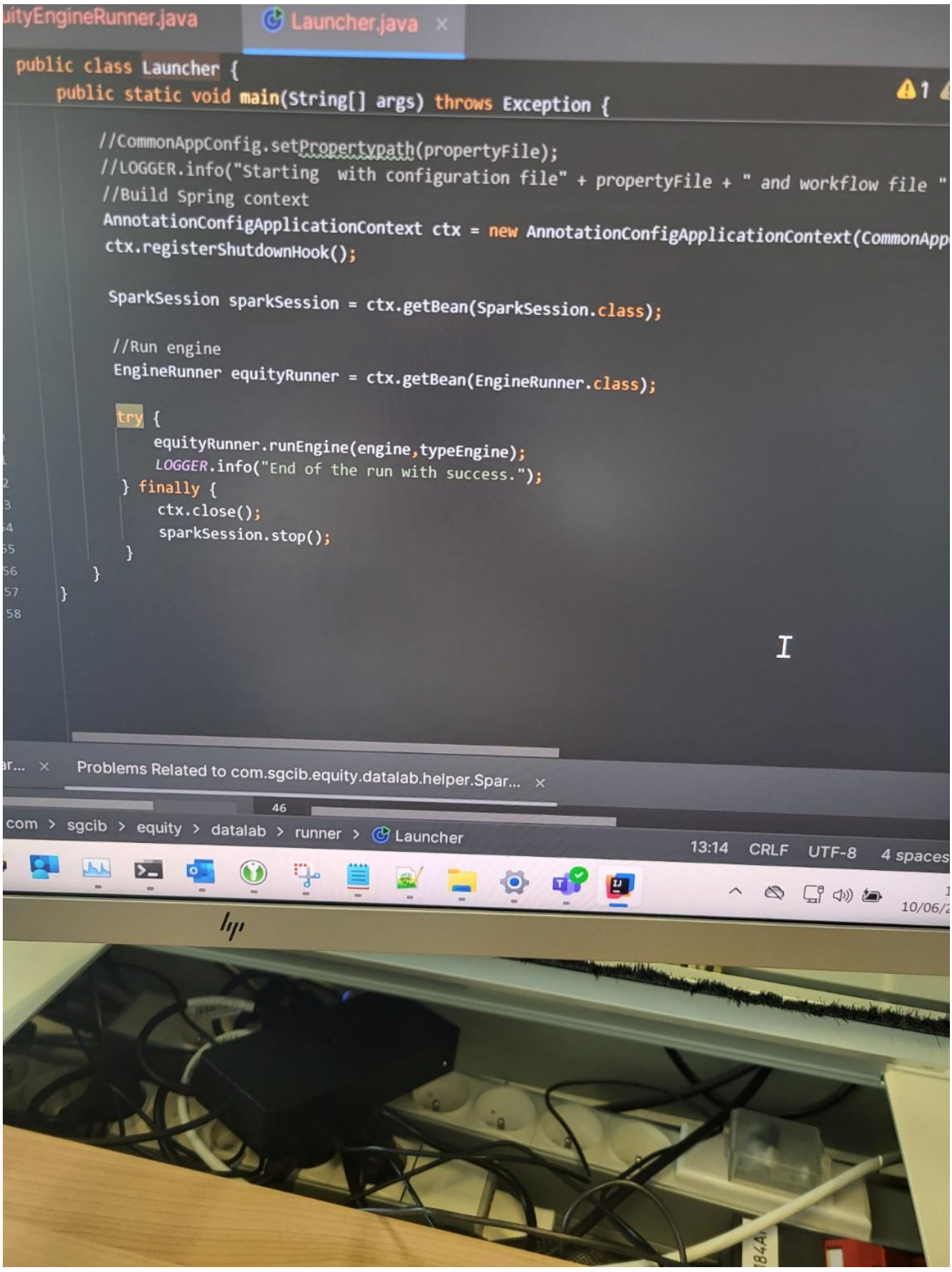
1. Launcher class:
   - Initializes the Spring context (AnnotationConfigApplicationContext)
   - Retrieves Spring-managed beans (SparkSession, EngineRunner, etc.)
   - Calls engineRunner.runEngine(...) to trigger the workflow

2. EquityEngineRunner class (implements IEngineRunner):
   - Uses Spring dependency injection (ApplicationContext, SparkContext, etc.)
   - Builds the execution configuration via getRunConfigurationEngine(...)
   - Dynamically chains the steps with name, type, and parameters

3. getRunConfigurationEngine(...):
   - Iterates over configured steps and creates dynamic Step instances
   - Populates name, engine, type, and parameters
   - Organizes steps into a Map<String, List<Step>> structure, then injects it into the main Run bean


public Object getRunConfigurationEngine(String engineIn, String typeEngineIn) {
allStepOrdered.forEach((String engine, Map<String, Map<...>> byLetter) ->
byLetter.forEach((String typeEngine, Map<String, Class<...>> values) -> {
for (Map.Entry<String, Class<AbstractStep>> entry : values.entrySet
String stepName = entry.getKey();
Step step = new Step();
step.setStepName(stepName);
step.setEngineName(engine);
step.setEngineType(typeEngine);
//TODO add params
step.setParams(null);
steps.add(step);
System.out.println("stepName: " + stepName + ", Value: " + entry.
}
stepsMap.put(typeEngine, steps);
stepsConfMap.put(engine,stepsMap);
});
run.setStepsConfMap(stepsConfMap);
return run;
}
private StepExecutionException lookupStepExecutionException(Throwable e) { 2 usages
Problems Related to com.sgcib.equity.datalab.helper.Spar...
datalab > runner > EquityEngineRunner > getRunConfigurationEngine
143:28 (15 chars)
LF
UTF-8


quityEngineRunner.java
package com.sgcib.equity.datalab.runner;
import ...
@Component
public class EquityEngineRunner implements IEngineRunner<Run, AbstractRecord> {
private static final Logger LOGGER = LoggerFactory.getLogger(EquityEngineRunner.class);
@Autowired
private ApplicationContext ctx;
@Autowired
private ApplicationEventPublisher eventPublisher;
@Autowired
private SparkContext sparkContext;
/*@Autowired
private MapAccumulator mapAccumulator;*/
@Autowired
private EngineContext engineContext;
@Override 1 usage
public void runEngine(Run run, Dataset<AbstractRecord> dataset) throws Exception {
Problems Related to com.sgcib.equity.datalab.helper.Spar...
com > sgcib > equity > datalab > runner > EquityEngineRunner
143:28 (15 chars)
LF
UTF-8
4 spaces


ityEngineRunner.java
Launcher.java
public class Launcher {
public static void main(String[] args) throws Exception {
//CommonAppConfig.setPropertypath(propertyFile);
//LOGGER.info("Starting with configuration file" + propertyFile + " and workflow file "
//Build Spring context
AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(CommonApp
ctx.registerShutdownHook();
SparkSession sparkSession = ctx.getBean(SparkSession.class);
//Run engine
EngineRunner equityRunner = ctx.getBean(EngineRunner.class);
try {
equityRunner.runEngine(engine,typeEngine);
LOGGER.info("End of the run with success.");
} finally {
ctx.close();
sparkSession.stop();
}
}
}
Problems Related to com.sgcib.equity.datalab.helper.Spar...
com > sgcib > equity > datalab > runner > Launcher
13:14 CRLF UTF-8 4 spaces