**Perigal Repeater** : API to offload CRON handling , leveraging native implantation of business logic in a JAVA Application

We often come across situations requiring us to run CRON based programs *not only in planned development sprints* but in *emergency patch-up activities* also . One of the finest and quickest approach is using OS’s build-in CRON service(E.g. **crontab** *for* **Linux** *or* **schtasks** *for* **Windows**)

**But what if** –

* Developer does not have sufficient permission instantly (In big projects, allowing permission etc. are strict formalities) ?
* Developer’s machine’s is a different platform from the platform that’s on target (maybe, PRODUCTION) machine?
* The tasks to be performed contain native OS commands and the code must work simultaneously across both Linux and Windows?
* The developer might not be readily acquainted with a CRON library such as Quartz. ?

1. **BASIC INFORMATION :**

Keeping all these under consideration, aim of Perigal Repeater, a JAVA-based API, is to leverage Separation of Concern ; so that handling CRON tasks etc. are offloaded from developer who may write the business logic for the Ecosystem of the project. One step further, it provisions codes which contain OS-specific commands.

In its current implementation , which is its first, the above are taken care of; *with a plan to further enhance it – by introducing different flavours of Linux and Windows and provisioning MAC and other popular Operating Systems and different Architecture.*

Also, *future release will generate CRON expressions on its own; so that the developer or the user who runs it, should be just fine having provided details like time of the day, frequency per week or month etc*.

1. **INTERNAL ARCHITECTURE:**

The basic classes/interfaces are:

1. com.soham.tools.cronjob.model.**AbstractWorkModel**
2. com.soham.tools.cronjob.main.**AppConfigPoint**
3. com.soham.tools.cronjob.model.**DTO\_BusinessModel**
4. com.soham.tools.cronjob.cron.**CronJob**

Among these, developer need to use the **first three only**. The last one is *final*.

**A screenshot of a cell phone

Description generated with high confidence**

1. **DEVELOPER’S GUIDE:**
   1. **Pre-requisite**: **Java SE 1.8 or above [Intentionally, I kept it incompatible to 1.7 or an earlier version though the API itself does not rely on any feature of Java 1.8 and would have just compiled with compiler level of 1.6.** *This was to just discourage using earlier versions for new development works. For systems, where older versions of JRE is being used, there’s a version, which will be available shortly* ;

*But for now , please email me at* [*soham.jisit@gmail.com*](mailto:soham.jisit@gmail.com) *for a copy of a free-to-use-commercially version*

* 1. I assume we’re using Eclipse family of IDEs – but that’s not a requirement. Screenshots etc. will pertain to an Eclipse Workbench- nothing beyond that!
  2. Create a Java Project
  3. Add this **JAR** to the Build Path of the project.
  4. Now follow these steps:
     + **STEP-1**: Get object of an com.soham.tools.cronjob.main.**AppConfigPoint**

AppConfigPoint appConfigPoint = AppConfigBuilder.*buildAppConfigPoint*();

* + - **STEP-2**: Call com.soham.tools.cronjob.main.AppConfigPoint.**scheduleService**()

appConfigPoint.scheduleService();

**package** com.perigalrepeter.example.mainclasses;

**import** org.quartz.SchedulerException;

**import** com.perigalrepeter.example.configbuilderfactory.AppConfigBuilder;

**import** com.soham.tools.cronjob.main.AppConfigPoint;

/\*\*

\* **@author** ssengup1 The simplest demo of using Perigal Repeater

\*/

**public** **class** MainClassExample1 {

/\*\*

\* **@param** args

\* **@throws** SchedulerException

\*

\*/

**public** **static** **void** main(String[] args) **throws** SchedulerException {

/\*\*\*

\* STEP-1: Get object of an com.soham.tools.cronjob.main.AppConfigPoint

\*/

AppConfigPoint appConfigPoint = AppConfigBuilder.*buildAppConfigPoint*();

/\*\*

\* STEP-2: Call com.soham.tools.cronjob.main.AppConfigPoint.scheduleService()

\*/

appConfigPoint.scheduleService();

/\*\*\*

\* NOTE: Inside

\* com.perigalrepeter.example.configbuilderfactory.AppConfigBuilder.buildAppConfigPoint()

\* ,

\*

\* call to CronJob.setAppConfigPoint(appConfigPoint) has already been made

\*/

}

}

* + - **For achieving [STEP-1]**

**[STEP 1.1]:** sub-class*com.soham.tools.cronjob.model***.AbstractWorkModel**

**[STEP-1.2]** Create an object of the same**.**

**[STEP-1.3**] Create object of*com.soham.tools.cronjob.model***.DTO\_BusinessModel**

**[STEP-1.4]** Configure the above businessModel object

**[STEP-1.5]** Pass arguments if any, else null; to businessModel object

**[STEP-1.6]** Call the setter **CronJob.***setAppConfigPoint***(appConfigPoint***); without this , scheduler should not be started. Else, you would come across "ERROR-CODE:0x1001 \n AppConfigPoint is not set! "*

**public** **class** AppConfigBuilder {

/\*\*

\* <ul>

\* <li>

\* STEP-1: Create object of an implementation of {@link AbstractWorkModel}

\* </li>

\* <li>

\* STEP-2: Create object of and configure your {@link DTO\_BusinessModel}

\* </li>

\* <li>

\* STEP-3:Pass arguments if any, else null; to the object of {@link DTO\_BusinessModel}

\* </li>

\* <li>

\* STEP-4: Call the setter CronJob.setAppConfigPoint(appConfigPoint); <b> without this , scheduler should not be started. Else, you would come across "ERROR-CODE:0x1001 \n AppConfigPoint is not set! "

\* </li>

\*

\* </ul>

\*/

**public** **static** AppConfigPoint buildAppConfigPoint() {

AbstractWorkModel workModel = **new** Impl\_WorkModel();

DTO\_BusinessModel businessModel = **new** DTO\_BusinessModel(workModel);

businessModel.setObjArr(**null**);

**final** AppConfigPoint appConfigPoint = **new** AppConfigPoint(businessModel);

appConfigPoint.setTriggerExpression("0/5 \* \* \* \* ?"); // Setting CRON Expression- execute every 5 second

CronJob.*setAppConfigPoint*(appConfigPoint);

**return** appConfigPoint;

}

}

* + **API Level Exceptions - Troubleshooting:**

In present version, there’re two API level error-codes that may be come across.

|  |  |  |  |
| --- | --- | --- | --- |
| ERROR CODE | MESSAGE | Reason | Remedy |
| 0x1001 | AppConfigPoint is not set! com.soham.tools.cronjob.cron.**CronJob**.*setAppConfigPoint*(appConfigPoint) must be called before *com.soham.tools.cronjob.main*.**AppConfigPoint**.*scheduleService*() | You **omitted** to invoke before calling scheduleService the following:  com.soham.tools.cronjob.cron.**CronJob**.*setAppConfigPoint*(appConfigPoint) | Invoke com.soham.tools.cronjob.cron.**CronJob**.*setAppConfigPoint*(appConfigPoint)  before calling  *com.soham.tools.cronjob.main*.**AppConfigPoint**.*scheduleService*() |
| 0x1002 | java.lang.IllegalArgumentException: Error Code: 0x1002 -com.soham.tools.cronjob.main.**AppConfigPoint**.triggerExpression has not been set properly | CRON needs an expression to trigger. No trigger expression has been set-up/configured by invoking  *com.soham.tools.cronjob.main*.**AppConfigPoint**.*setTriggerExpression*(**String**) | Invoke  *com.soham.tools.cronjob.main*.**AppConfigPoint**.*setTriggerExpression*(**String**)  before calling  *com.soham.tools.cronjob.main*.**AppConfigPoint**.*scheduleService*() |

* + **A sample output:**

**package** com.perigalrepeter.example.model;

**import** java.util.Date;

**import** com.soham.tools.cronjob.model.AbstractWorkModel;

/\*\*

\* **@author** ssengup1

\* Sub-class com.soham.tools.cronjob.model.AbstractWorkModel

\*

\*/

**public** **class** Impl\_WorkModel **extends** AbstractWorkModel {

/\* (non-Javadoc)

\* @see com.soham.tools.cronjob.model.AbstractWorkModel#repeatJobOnLinux(java.lang.Object[])

\*/

@Override

**public** Object repeatJobOnLinux(Object[] arg0) {

String x="From linux "+**new** Date();

System.***out***.println("Impl\_WorkModel.repeatJobOnLinux()"+x);

**return** x;

}

/\* (non-Javadoc)

\* @see com.soham.tools.cronjob.model.AbstractWorkModel#repeatJobOnWindows(java.lang.Object[])

\*/

@Override

**public** Object repeatJobOnWindows(Object... arg0) {

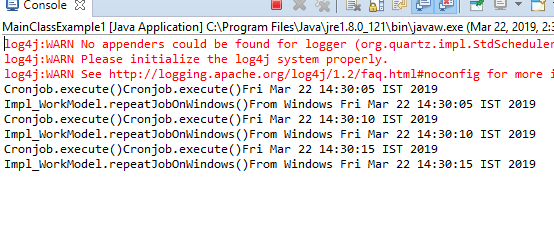
String x="From Windows "+**new** Date();

System.***out***.println("Impl\_WorkModel.repeatJobOnWindows()"+x);

**return** x;

}

}



1. **Resources**:
   * **Perigal Repeater v1.0** –

* **JAR:** <https://github.com/trainerpb/PerigalRepeater/blob/master/PerigalRepeaterV1.0.jar>

* **JAVA DOCS:** <https://github.com/trainerpb/PerigalRepeater/blob/master/docs/index.html>
* **Example Project** <https://github.com/trainerpb/PerigalRepeater/tree/master/Example_PerigalRepeater>
  + **Java SE 1.8** : <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
  + **Eclipse:** - https://www.eclipse.org/downloads/