UCW Job Scheduler

We provide UCW IoT Cloud where you can run multiple instances of the **UCW Platform** for needs of your projects. You can start by registering h ere and get access to the IoT platform with unique functionality. To learn more check out the UCW Platform overview.

Tutorial overview

Using UCW Platform users are also able to schedule jobs. The job scheduler provides support for automation process in distribution of tasks across the platform infrastructure. It represents task distribution for system management, distributed computing, monitoring or any another task which can do a specific job. In this tutorial basic work with them will be shown and more advanced job which works data streams will be explained. To test code shown in this tutorial you can use any project from other UCW Portal tutorials. UCW Portal Dashboard and Gadget project is recommended since SensorDataGener atorJob will be explained at the end of the tutorial.

Requirements

- Git
- Maven

Step 01: Download tutorial sources

As mentioned above pick any UCW Portal tutorial and complete first step as instructed there.

Step 02: Create a job

In this step you will create simple job. It will just log "Hello World from job!" message to a log. Do as follows:

1. Create *HelloWorldJob.java* class which extends AbstractJob.
2. Implement the *execute* function with following code

```
@Override
public void execute(final Map<String, Object> map) {
    getLog().info("Hello World from job!");
}
```

Step 03: Register and plan job

In the previous step you created a job. However there is no code which runs it. That's what you are going to do now. Follow the steps:

1. Open your **Plugin.java* file (e.g. *DashboardPlugin*.java)
2. Paste following snippet into *scheduleJobs* method (or any other method which is called from plugin *load* function)

```
SchedulerService scheduler = ServiceProvider.getInstance().getServiceHost(SchedulerService.class);

Date delayTime = TimeUtils.addSeconds(TimeUtils.getNow(), 10);
long repeatInterval = 10 * 1000; // 10sec

scheduler.scheduleJob(HelloWorldJob.class, null, delayTime, repeatInterval);
```

3. Build and run the application and watch the log. Every ten seconds there should appear new line with this content "INFO : Hello World from job! - HelloWorldJob"

Yes. It is that simple. Let's explain a bit what is going on.

The SchedulerService takes four inputs:

- **Class** - The class needs to implement interface Job. When scheduler gets the impulse to run the registered job it will create an instance of this class and run job's execute function.
- **Map<String, Object>** - map which is passed to an instance when execute function is called.
- **Date** - The dates object which states from when the job should be scheduled.
- **long** - number of miliseconds between each run of the job

Since *HelloWorldJob* does not need any input argument second argument is set to null in the example. DelayTime is set to current time plus ten secods. In general, the delay time is recommended to be set to some reasonable value since the application can be still loading and some components may not be initialized yet. In such case the job may not be working properly. In this case ten seconds is sufficient since the job does not need any components to be initialized.

That's it. Your first job created under UCW Platform.

More realistic use cases

HelloWorldJob is very simple job and actually does not do anything particularly useful. Jobs will usually need to work with database or device or both. As mentioned in the tutorial overview there is a more complex job in UCW Portal Dashboard and Gadget tutorial. Let's explain what is going on there.

TestDataUtils

Firstly take a look at the *TestDataUtils* class. As you can see it uses many of the default Managers which are provided by UCW platform, namely *ProjectManager, DeviceManager, DataManager*. The whole purpose of this class is to create a sample project, device and data stream for further use in the application. You can check that *createData* method is called from *DashboardPlugin* while loading the plugin. In reality you would not have to do this step since you will have real devices and real data but for testing purposes this could be useful.

DashboardPlugin#scheduleJobs

In the *HelloWorldJob* you did not need to pass any component into the job. The situation with SensorDataGeneratorJob is different. It needs to work with *DeviceManager* and *DataManager* components. Hence they are added to *jobDataMap* parameter of *scheduleJob* function.

SensorDataGeneratorJob

Last step which needs to be explained is in SensorDataGeneratorJob class. The job just takes all which has been prepared and uses it. Let's see how

1. Takes out *DeviceManager* and *DataManager* instances from *jobDataMap*
2. Asks DeviceManager if there is a device with specific identifier in specific project
3. Asks DataManager if there is a data stream with specific identifier in specific project
4. Generates random data
5. Store the data into the data stream as if it were generated by the device

The data stored by the job are eventually used in a dashboard gadgets as you can see for example in SensorDataGadget.