Starting/Stopping a Strategy from a Strategy

When you desire to start and stop a strategy from within another strategy, follow these steps. You may wish to do this to automatically control the lifecycle of a strategy.

Step-by-step guide

1. Create a command file as normal that creates (but does not start) all your strategies: master and slave (or slaves, this sample shows only one, but there could be more)

Command File

```
#
# Create the two strategies
#
createModule;metc:strategy:system;slave,Slave,JAVA,src/Slave.java,,true,metc:remote:receiver
#
createModule;metc:strategy:system;master,Master,JAVA,src/Master.java,,true,metc:remote:receiver
#
# start Master only from Photon (may instead start here, if you'd like by un-commenting the next line)
#
#startModule;metc:strategy:system:master
```

2. From the Master strategy, when the time is right, start the slave strategy. In this sample, look at `onCallback` for where the slave is started and stopped.

Master Strategy

```
import org.marketcetera.module.ModuleException;
import org.marketcetera.module.ModuleManager;
import org.marketcetera.module.ModuleURN;
import org.marketcetera.strategy.StrategyModuleFactory;
import org.marketcetera.strategy.java.Strategy;
/* $License$ */
/**
 * Controls starting and stopping another strategy.
 *
 * @author <a href="mailto:colin@marketcetera.com">Colin DuPlantis</a>
 * @version $Id$
 * @since $Release$
 */
public class Master
        extends Strategy
{
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onStart()
     */
    @Override
    public void onStart()
    {
        moduleManager = ModuleManager.getInstance();
        strategyURN = new ModuleURN(StrategyModuleFactory.PROVIDER_URN,
                                    "slave");
        warn("Master strategy started");
        requestCallbackAfter(10000,
                             true);
    }
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onCallback(java.lang.Object)
     */
    @Override
    public void onCallback(Object inData)
    {
        boolean startFlag = (Boolean)inData;
        try {
            if(startFlag) {
                warn("Starting slave strategy");
                moduleManager.start(strategyURN);
                requestCallbackAfter(10000,
                                     false);
            } else {
                warn("Stopping slave strategy");
                moduleManager.stop(strategyURN);
            }
        } catch (ModuleException e) {
            throw new RuntimeException(e);
        }
    }
    /**
     * handle to module manager instance
     */
    private ModuleManager moduleManager;
    /**
     * handle to strategy to start/stop
     */
    private ModuleURN strategyURN;
}
```

Dynamically Creating the Strategy

You can create the slave strategy completely from within Master rather than the command file, if desired. This would allow you to dynamically create, start, stop, and destroy slave strategy instances

Inter-Strategy Communication

You can communicate between strategies with the common key/value properties available from `getProperties`. All strategies in the same process share this space.

Related articles