ApplicationComposer with JBatch

ApplicationComposer can be a way to run a JBatch not needing any HTTP connector.

Here is an example making batch integration easy - note you can extract the generic part in a library very easily:

TIP	if you didn't check yet BatchEE provides some standalone utilities for JBatch but the idea of this page can be reused for a lot of applications.

```
// helper class reusable for any batch
abstract class BatchApplication {
    private static final DateTimeFormatter DATE = DateTimeFormatter.ofPattern(
"YYYYMMddHHmmss");

    protected Report runBatch(final String batchName, final Properties config) {
        final JobOperator operator = BatchRuntime.getJobOperator();
        final long id = operator.start(batchName, config);
        Batches.waitForEnd(operator, id);
        return new Report(operator.getJobExecution(id), operator.getParameters(id));
    }

    @Module // we enforce BatchEE to be initialized as an EJB context to get JNDI for
JTA init, needed for TomEE 1
    public EjbModule ensureBatchEESetupIsDoneInTheRightContext() {
        final EjbJar ejbJar = new EjbJar().enterpriseBean(new SingletonBean
(BatchEEBeanManagerInitializer.class));

        final Beans beans = new Beans();
        beans.addManagedClass(BatchEEBeanManagerInitializer.Init.class);

        final EjbModule ejbModule = new EjbModule(ejbJar);
        ejbModule.setModuleId("batchee-shared-components");
        ejbModule.setBeans(beans);
        return ejbModule;
    }

    public static class Report {
        private final JobExecution execution;
        private final Properties properties;

        public Report(final JobExecution execution, final Properties properties) {
            this.execution = execution;
            this.properties = properties;
        }

        public JobExecution getExecution() {
            return execution;
        }

        public Properties getProperties() {
            return properties;
```

```
        }
    }
}

@Classes(cdi = true, value = { MyFilter.class, MoveFile.class, InputFile.class,
MyReader.class, LoggingListener.class })
public class MyBatch extends BatchApplication {
    private final Properties config;

    public Mybatch(final String[] args) { // main args
        this.config = new Properties() {{ // create the batch config
            setProperty("input-directory", args[0]);
        }};
    }

    public Report execute(final String inputDirectory) {
        return runBatch("sunstone", config);
    }

    public static void main(final String[] args) throws Exception {
        ApplicationComposers.run(MyBatch.class, args);
    }
}
```