Guia do Desenvolvedor

Guia de implementação para aplicativos Batch do FIBRA.

As aplicações Batch do FIBRA utilizam como base o framework **Spring-Batch**, dessa forma, é fundamental o domínio de sua arquitetura e seus componentes (introdução básica aqui). A documentação de referência da versão 2.2.7 do Spring-Batch: http://docs.spring.io/spring-batch/2.2.x/reference/html/index.html.

Link para apresentação do Workshop sobre Spring-Batch realizado no DERCE:

wiki.serpro/unidades/supde/derce/sistemas/fibra/batch/apresentacao-spring-batch.pdf

---

Criação do Projeto

Para iniciar um projeto Batch fibra existem 2 archetypes do maven (disponíveis no catálogo do Nexus nexus.aic.serpro/service/local/repo_groups/public/content/archetype-catalog.xml) fibra-batch-sample-archetype e fibra-batch-sample-hibernate-archetype.

- O fibra-batch-sample-archetype cria um projeto básico com todas as dependências necessárias e alguns Jobs de exemplo.
- O fibra-batch-sample-hibernate-archetype cria um projeto básico com as dependências necessárias para a utilização de Hibernate e do componente fibraInfra. A utilização do Hibernate neste tipo de aplicação é bastante controverso.

Ambiente de Execução

O empacotamento dos projetos, criados a partir dos archetypes citados anteriormente, cria um arquivo JAR que poderá ser executado a partir da linha de comando, por exemplo:

java -Dfibra.config.dir=/opt/approtinas/tes_37099_fibrabatch -jar fibra-xxx-batch-1.0.0-executable.jar xxx-jobs/meuJob.xml xxxJob ator="..." ecu="..." paramX="..."

O "VM argument" **-Dfibra.config.dir** define o caminho do diretório onde estão os arquivos **log4j.properties e fibra-batch.properties.**

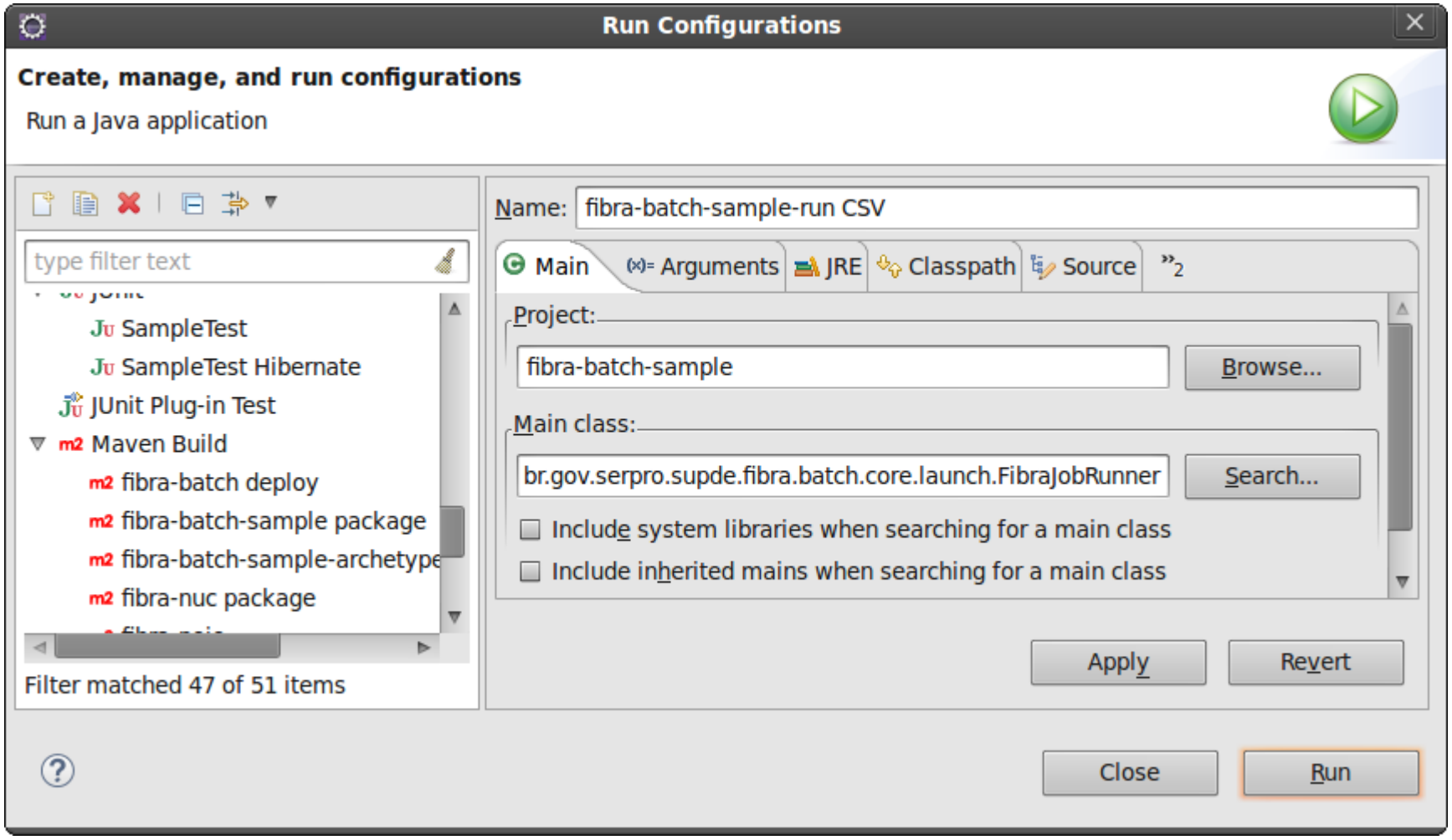
- O log4j.properties define as configurações de log do processamento do Job. Exemplo: wiki.serpro/unidades/supde/derce/sistemas/fibra/batch/log4j.properties
- O fibra-batch.properties define as configurações de execução do Job (propriedades de conexão com o banco de dados, diretório de entrada e saída de recursos utilizados pelo Job, configuração de threads para processamento paralelo, configurações de fila JMS para processamento distribuído e etc). Exemplo de um arquivo fibra-batch.properties: wiki.serpro/unidades/supde/derce/sistemas/fibra/batch/fibra-batch.properties.

Após o nome do jar, é necessário informar os seguintes parâmetros (exatamente na ordem indicada abaixo):

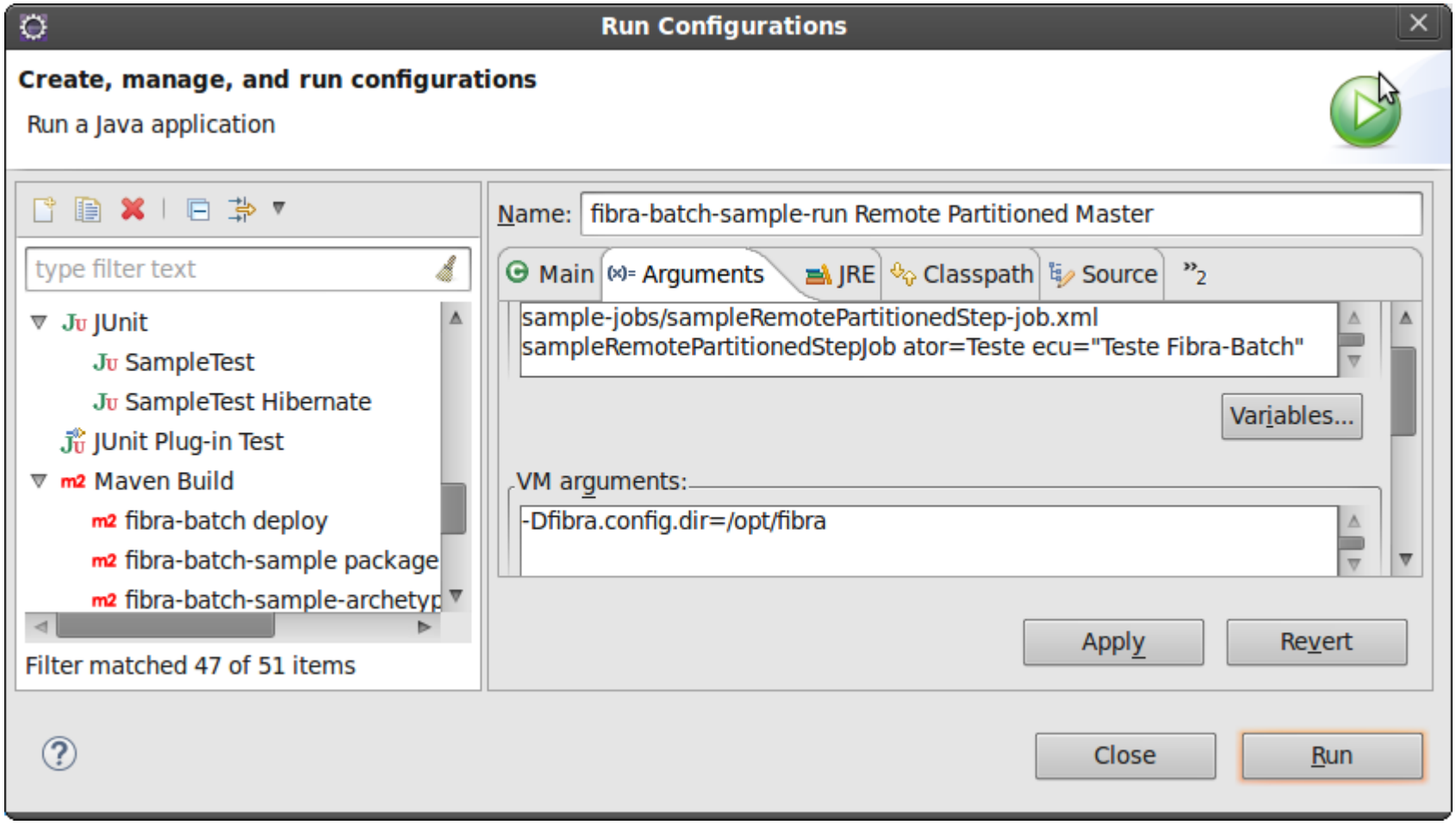
1. Caminho para o arquivo xml com a definição do Job
2. Nome do Job a ser executado
3. Demais parâmetros do Job no formato chave=valor (o FibraJobRunner utiliza o docs.spring.io/spring-batch/apidocs/org/springframework/batch/core/launch/support/CommandLineJobRunner.html, verifique a documentação do mesmo para ver todos os parâmetros permitidos)

Executando a partir do Eclipse

Para executar a partir do eclipse, basta configurar a main class br.gov.serpro.supde.fibra.batch.core.launch.FibraJobRunner para o seu projeto em Run Configurations.



Na aba *Arguments*, na caixa de texto "*Program Arguments*" deverão ser informados os parâmetros do Job como definido anteriormente e no "*VM Arguments*" deverá ser informada a *system property* **-Dfibra.config.dir**. Exemplo:



Obs1: Para evitar erros **"Job Terminated in error: A job execution for this job is already running"** no ambiente de **desenvolvimento**, adicione o parâmetro **-next**, em Program arguments, para que o incrementer seja acionado e gerada uma nova JobInstance.

Obs2: Também é possível adicionar, em *VM arguments*, o parâmetro **-Dfibra.log.dir** para definir uma pasta específica para os arquivos de log (se essa propriedade não for definida os logs serão armazenados em uma pasta "log" dentro do diretório configurado em -Dfibra.config.dir). Para direcionar o log também para o console (stdout), adicione a System Property **-Dfibra.verbose=true**.

Obs3: As *System properties* sobreescrevem as propriedades configuradas no arquivo fibra-batch.properties.

Testes Unitários: As aplicações geradas a partir dos archetypes citados anteriormente possuem exemplos de uso do jUnit para testar os Jobs.

Execução de testes unitários a partir do maven: Para executar a fase de testes no maven é necessário passar o seguinte VM Argument para a execução do mesmo: **-DargLine="-Dfibra.config.dir=**<<diretório do arquivo fibra-batch.properties>>"

Shell Script e Control-M

Os jobs serão executados a partir do seguinte shell script: wiki.serpro/unidades/supde/derce/sistemas/fibra/batch/fibra-batch.sh/view acionado pelo Control-M. Execute o script com a opção -h para uma ajuda.

Ambiente de Testes

A máquina utilizada para testes dos Jobs do fibra é a 10.200.220.158.
As builds dos projetos devem ser transferidas para a pasta **/opt/approtinas/tes_37099_fibrabatch**

O script fibra-batch.sh já está configurado na máquina, assim como o fibra-batch.properties. Para o testador executar o job, deverá acessar via ssh a máquina 10.200.220.158 e então executar o fibra-batch. Exemplo:

```
 $ ssh root@10.200.220.158
# fibra-batch --config /opt/approtinas/tes_37099_fibrabatch -jar fibra-batch-sample-2.0.0-executable.jar -x sample-jobs/sampleJDBC-job.xml -i sampleJDBCJob --params -next
```

FIBRA-BATCH-CORE

O fibra-batch-core é o componente responsável por implementar o comportamento comum entre os projetos do Fibra e fornecer algumas facilidades para o desenvolvimento dos Jobs.

Para utilizar as funcionalidades do fibra-batch-core é necessário importar as definições do contexto do spring no Job, exemplo:

<beans:import resource="classpath:/spring/context/fibra/batch/fibra-batch-context.xml" />

Este contexto possui as configurações do jobRepository, jobLauncher, configuração de conexão com o banco de dados e etc. A aplicação só precisa se preocupar em implementar os Jobs. A infraestrutura necessária já está definida no **fibra-batch-context.xml** e os parâmetros de configuração são definidos no arquivo **fibra-batch.properties**.

ParentJob

Todos os Jobs do fibra possuem alguns passos em comum (verificar se as triggers de Banco de Dados estão ativadas "auditaTriggers", verificar agendamento e baixar os arquivos de entrada do Job no "fibra-nuc", enviar relatório de execução para o "fibra-ted"), o Job **fibraParentJob** define esse comportamento. Portanto, todos os Jobs do fibra devem herdar do **fibraParentJob,** exemplo:

```
<batch:job id="sampleCSVJob" parent="fibraParentJob">
...
</batch:job>
```

ParentStep

Também existem algumas definições padrões para os steps, dessa forma, todo step deve herdar de **fibraParentStep** (em steps que executem tasklets) ou **fibraParentChunkedStep** (para steps que executem chunks). Definem a transação, o commit-interval (para os chunks), NoRollbackException (se for lançada esse tipo de exception, o rollback não é efetuado) e SkippableException (para os chunks).

```
<batch:step id="step1" parent="fibraParentStep" next="step2">
   <batch:tasklet ref="sampleTasklet" />
</batch:step>
<batch:step id="step2" parent="fibraParentChunkedStep">
   <batch:tasklet task-executor="syncTaskExecutor">
      <batch:chunk reader="sampleJDBCReader" writer="cvsFileItemWriter"/>
   </batch:tasklet>
</batch:step>
```

Listeners

Tanto o fibraParentJob, quanto os fibraParentStep e fibraParentChunkedStep definem listeners. Dessa forma, caso precise definir algum listener específico, é necessário utilizar o atributo merge="true", para preservar o comportamento comum. Exemplo:

```
<batch:job id="sampleCSVJob" parent="fibraParentJob">
...
   <batch:listeners merge="true">
   <!-- meus listeners específicos -->
   </batch:listeners>
</batch:job>
```

FibraJobContext

O FibraJobContext é configurado, a partir do arquivo **fibra-batch.properties**, no início do processamento e contém informações sobre o contexto de execução do Job, por exemplo:

- Diretório onde estão os arquivos de entrada do Job (**inputPath**).
- Diretório onde deverão ser gerados os arquivos de saída do Job (**outputPath**).
- Modo de simulação. Se ativado, nenhum commit é executado no banco de dados (**simulationMode**).
- Ignorar verificação de agendamento e download de arquivos de entrada do Job (**ignoreSchedule**).
- Ignorar gravação do relatório de execução no fibra-ted (**ignoreReport**).
- Remover arquivos de entrada e/ou saída após a execução do Job (**removeInputFiles** e **removeOutputFiles**).

As informações do FibraJobContext podem ser acessadas através de expression language na definição dos Jobs. Exemplo:

```
<beans:bean name="sampleCSVReader"     class="br.gov.serpro.supde.fibra.batch.core.support.CSVFileItemReader">
   <beans:property name="lineMapper" ref="XXXLineMapper" />
   <beans:property name="resource" value="file:#{fibraJobContext.inputPath}/XXX.csv"/>
</beans:bean>
```

Sobreescrevendo Configurações do fibra-batch.properties

O arquivo fibra-batch.properties é único para todo o Fibra e define configurações gerais para todos os módulos. Entretanto, algumas configurações podem ser customizadas para um Módulo ou Job específico. O carregamento das propriedades segue a seguinte ordem de prioridade:

1. **Linha de Comando (opcional)**: Os parâmetros passados como System Property na linha de comando tem prioridade sobre todos os demais arquivos de configuração.
2. **jobId.properties (opcional)**: Um arquivo .properties na pasta /resources do projeto com o mesmo nome do jobId (ou seja, o arquivo vai embarcado no jar). Carrega suas propriedades quando o job específico executar.
3. **fibra-context.properties (obrigatório)**: Todos os módulos devem possuir um arquivo fibra-context.properties na pasta /resources do projeto (ou seja, o arquivo vai embarcado no jar). Nesse arquivo a propriedade fibra.system deve ser definida com a sigla do módulo.
4. **fibra-batch.properties**: Depois de carregar todas as configurações dos passos anteriores, o componente finalmente carrega as configurações gerais do fibra-batch.properties, mantendo as carregadas anteriormente.

Integração com o FIBRA-TED / FIBRA-NUC

Alguns Jobs possuem integração com o FIBRA-TED para o agendamento de sua execução (cblist), o armazenamento do relatório e possíveis arquivos de saída.

Verificação de Agendamento

No início do processamento do Job, o componente chama o serviço **cblist** do FIBRA-NUC para verificar se existe uma execução agendada para o Job e se for o caso, baixar possíveis arquivos de entrada cadastrados. Para que o componente chame a rotina, é necessário que o Job seja executado com 2 parâmetros obrigatórios: **ecu** e **ator**. Exemplo:

java -Dfibra.config.dir=/opt/approtinas/tes_37099_fibrabatch -jar fibra-acr-batch-1.0.0-executable.jar acr-jobs/meuJob.xml xxxJob ator="ACR" ecu="Caso de Uso X" paramZ="..."

É possível obter o retorno da chamada ao serviço cblist através do método getRetornoCblistDTO() do FibraJobContext.

Relatório de Execução do Job

Cada Job é responsável por gerar o arquivo do relatório de execução como especificado no caso de uso, entretanto, o envio do arquivo para o FIBRA-TED é feito através do componente. Para evitar problemas de permissão no sistema de arquivos, é recomendado que o arquivo seja gerado na pasta configurada no **fibra-batch.properties** como **fibra.outputPath** (pode ser obtido através do objeto FibraJobContext.getInstance().getOutputPath()). O Job deve indicar, através do fibraJobContext o arquivo do relatório que ele gerou da seguinte forma:

```
File relatorio = new File(...);
FibraJobContext.getInstance().setReport(relatorio);
```

Para os Jobs que geram outros arquivos de saída que precisam ser armazenados no FIBRA-TED, deve registrar da seguinte forma:

```
File arquivoSaida = new File(...);
FibraJobContext.getInstance().getOutputFiles().add(arquivoSaida);
```

O tempo de execução e o status da execução são atualizadas através do componente.

ATIVAR/DESATIVAR a Integração

A Ativação/Desativação dessa integração com o FIBRA-TED pode ser feita através do arquivo de configuração fibra-batch.properties (ou através de alguma das formas discutidas na seção anterior, para habilitar e/ou desabilitar para um módulo ou job específico):

```
#Desabilitar a verificação do agendamento (cblist)
fibra.ignoreSchedule=true
#Habilitar a criação do relatório no fibra-ted
fibra.ignoreReport=false
```

Permissão de usuário do Banco de Dados

Para essa integração com o FIBRA-TED funcionar, o usuário do banco de dados precisa ter as seguintes permissões:

SELECT nas tabelas: ARQUIVO_DADOS, AGEND_PROCESS_LOTES, CASO_USOS e ATORS

INSERT e UPDATE nas tabelas: ARQUIVO_DADOS e AGEND_PROCESS_LOTES

Audita Triggers

Antes da execução de cada JOB a função **fc_audita_triggers_c** é chamada para verificar se as Triggers do Banco de Dados estão válidas e habilitadas. O fibraJobListener executa essa chamada no método beforeJob. Para definir os parâmetros da função audita triggers, deve-se utilizar 2 propriedades:

- fibra.auditaTriggers.modules (lista de módulos, separados por vírgula, a serem verificados. Exemplo: FIBR-NEG,FIBR-DOM)
- fibra.auditaTriggers.owners (lista de owners, separados por vírgula, a serem verificados (seguindo a ordem dos módulos acima). Exemplo: FIBR_MAN,TABF_MAN)

Para desabilitar a chamada ao auditaTriggers a propriedade fibra.auditaTriggers.ignore=true compre esse papel. Esta configuração deve ser utilizada apenas para ambientes de desenvolvimento e NUNCA em ambientes de homologação/produção.

Performance e Escalabilidade

No Spring-Batch, é possivel definir um task-executor para a execução dos steps de um Job. O fibra-core-batch define beans para cada tipo de task-executor (que pode ser configurado no arquivo **fibra-batch.properties**):

- **syncTaskExecutor** - Uma única Thread executa todo o step
- **simpleAsyncTaskExecutor** - Múltiplas threads disponíveis para a execução do step (as threads são criadas sempre que necessário e a quantidade máxima de threads concorrentes é definido na propriedade **fibra.executor.maxThreads** do arquivo fibra-batch.properties)
- **threadPoolTaskExecutor** - Pool de threads disponível para execução do step (as propriedades *fibra.executor.threads*, *fibra.executor.maxThread*, *fibra.executor.queueCapacity* e *fibra.executor.keepAliveSeconds* definem as configurações do pool)
- **concurrentTaskExecutor** - Cria um task-executor baseado nos tipos de Executor do pacote java.util.concurrent (a

propriedade **fibra.executor.type** define o tipo do executor: SINGLE, FIXED ou CACHED)

Obs: Ao utilizar algum task-executor multithread, certifique-se de que a execução do step em questão é thread-safe. Na aplicação gerada a partir do archetype fibra-batch-sample-archetype existem alguns exemplos de utilização de task-executor multithread (sampleJDBCPartitionStep-job.xml, sampleQueueParallelStep-job.xml e sampleCSVPartitionStep-job.xml).

Também é possível melhorar a performance do Job tornando apenas a execução da fase "process" assíncrona (nos casos onde não é possível, ou viável, paralelizar a execução da fase "read") utilizando as classes **AsyncItemProcessor** e **AsyncItemWriter**. Exemplo:

```
<beans:bean id="asyncProcessor"
      class="org.springframework.batch.integration.async.AsyncItemProcessor">
      <beans:property name="delegate">
            <beans:bean class="br.gov.serpro.supde.fibra.batch.sample.SampleProcessor" />
      </beans:property>
      <beans:property name="taskExecutor" ref="threadPoolTaskExecutor"/>
</beans:bean>

<beans:bean id="asyncWriter"
      class="org.springframework.batch.integration.async.AsyncItemWriter">
      <beans:property name="delegate">
            <beans:bean class="br.gov.serpro.supde.fibra.batch.sample.SampleWriter" />
      </beans:property>
</beans:bean>
```

Processamento Distribuído

Os principais métodos, definidos no spring-batch, para processamento distribuído são o Remote Chunking e o Remote Partitioning.

- **Remote Chunking** - O processamento do step é distribuído entre os vários nós slave, entretanto, a leitura e a escrita do step é sempre executada em um nó master. A utilização deste método é recomendada apenas nos casos onde as fases de escrita e leitura não são o gargalo e sim a execução do processamento. Na aplicação gerada a partir do archetype fibra-batch-sample-archetype veja um exemplo de implementação em sampleRemoteChunk-job.xml.
- **Remote Partitioning** - Toda a execução do step (leitura, processamento e escrita) é executada de forma distribuída entre o master e os nós slave. Na aplicação gerada a partir do archetype fibra-batch-sample-archetype veja um exemplo de implementação em sampleRemotePartitionedStep-job.xml.

Obs1: Para iniciar nós escravos para processamento distribuído de um Job, é necessário adicionar o seguinte VM Argument: **-Dfibra.slave=true**

Obs2: A comunicação entre o nó master e os nós escravos é feita através de JMS, a configuração relativa a conexão entre os nós está nas propriedades com prefixo fibra.jms no arquivo fibra-batch.properties.

Obs3: Para processamento MUITO pesado, é interessante avaliar o uso de Hadoop com o projeto Spring-XD projects.spring.io/spring-xd/

Monitorando a Execução dos Jobs

A execução dos Jobs pode ser monitorada através da linha de comando, gerando saída no console ou em PDF, ou através de uma interface WEB.

Monitorando a Execução a partir da linha de comando

É possível gerar relatórios de monitoração a partir da linha de comando passando o parâmetro **-report** para o job, exemplo:

java -Dfibra.config.dir=/opt/approtinas/tes_37099_fibrabatch -jar fibra-xxx-batch-1.0.0-executable.jar xxx-jobs/meuJob.xml xxxJob -report

Utilizando a Interface WEB

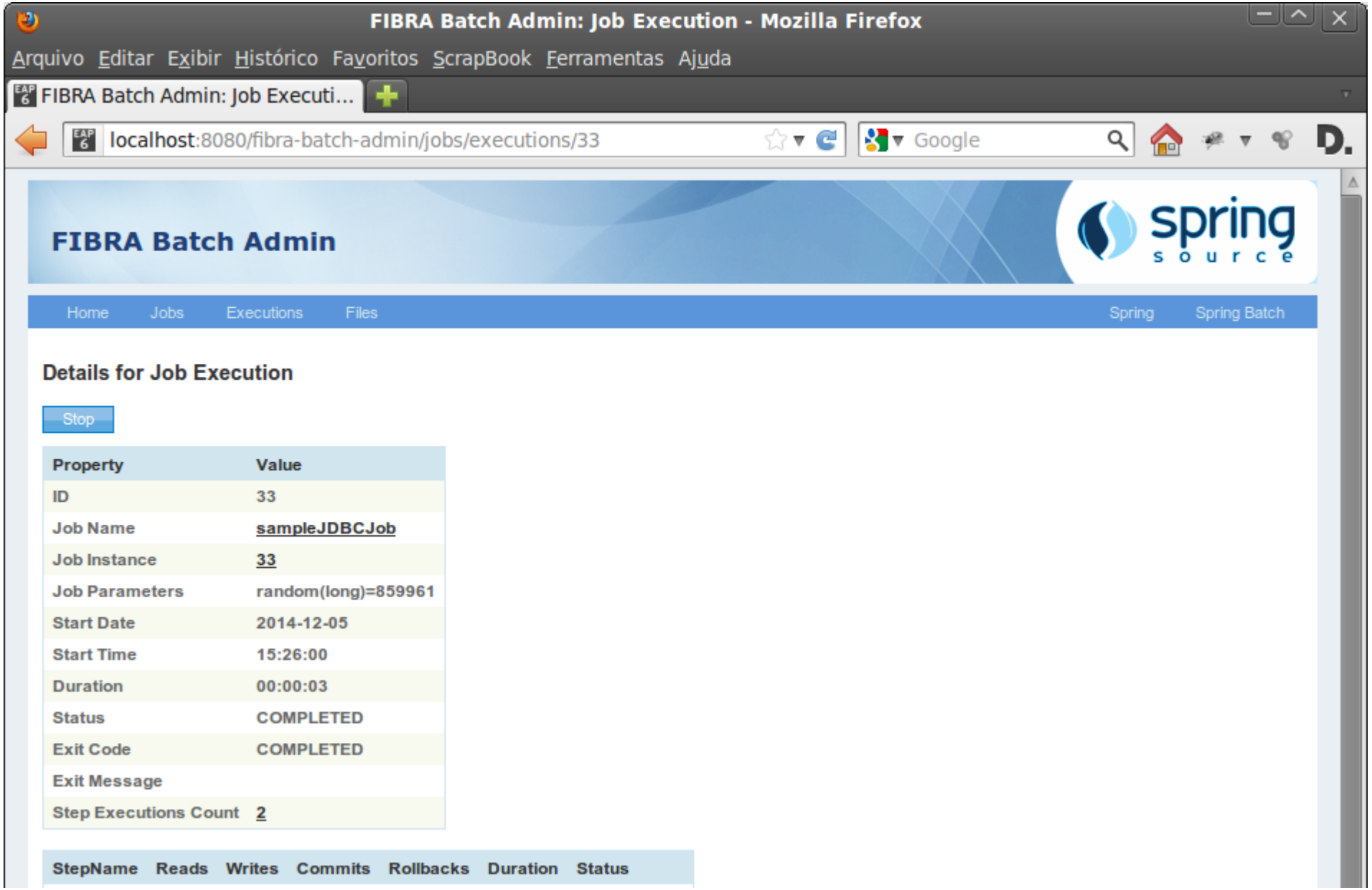
A execução dos Jobs do Fibra pode ser acompanhada a partir da **FIBRA Batch Admin** (que nada mais é do que o spring-batch-admin com algumas customizações no visual e na forma de referenciar o jobRepository para o Fibra). Uma versão do FIBRA-Batch-Admin pode ser obtida no nexus em nexus.aic.serpro/#nexus-search;quick~fibra-batch-admin. Baixe o .war mais recente e o mesmo poderá ser implantado em um servidor de aplicação Java (JBoss, Tomcat e etc).

A única configuração necessária para executar o FIBRA Batch Admin é a *System Property* **fibra.config.dir** definindo o diretório do arquivo fibra-batch.properties com as configurações de acesso ao *jobRepository*. Exemplo de configuração da System Property no servidor de aplicação JBoss EAP 6+:

No arquivo **standalone.xml**, definir a system-property FIBRA_CONFIG_DIR com o valor do caminho do diretório onde está o arquivo fibra-batch.properties:

```
<system-properties>
      <property name="fibra.config.dir" value="/opt/approtinas/tes_37099_fibrabatch"/>
      <!-- Outras propriedades de sistema...  -->
</system-properties>
```

A aplicação será iniciada no contexto **/fibra-batch-admin** e todos os Jobs executados no jobRepository configurado no fibra-batch.properties poderão ser monitorados.


FIBRA Batch Admin: Job Execution - Mozilla Firefox
Arquivo Editar Exibir Histórico Favoritos ScrapBook Ferramentas Ajuda
FIBRA Batch Admin: Job Executi...
localhost:8080/fibra-batch-admin/jobs/executions/33
Google
FIBRA Batch Admin
spring source
Home Jobs Executions Files Spring Spring Batch
Details for Job Execution
Stop
Property Value
ID 33
Job Name sampleJDBCJob
Job Instance 33
Job Parameters random(long)=859961
Start Date 2014-12-05
Start Time 15:26:00
Duration 00:00:03
Status COMPLETED
Exit Code COMPLETED
Exit Message
Step Executions Count 2
StepName Reads Writes Commits Rollbacks Duration Status