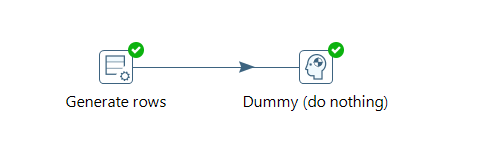
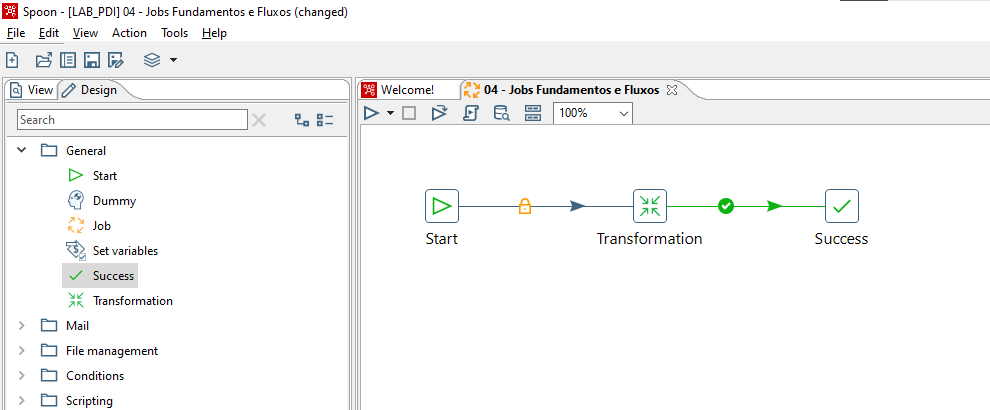
**Exercício 04 – Jobs**

* 1. Crie uma nova transformação, chamada “04 – Jobs Fundamentos”, gere 10.000 linhas de input e jogue o resultado em um dummy:



* 1. Agora será criado um elemento novo, o “Job”, que deve ser salvo como “04 – Jobs Fundamentos e Fluxos”. Na sessão “Design”, traga os steps “Start”, “Transform” e “Success”:



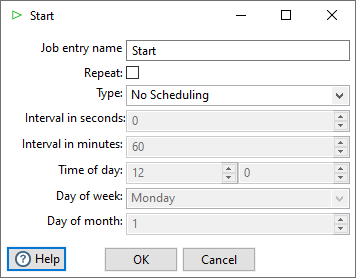
* 1. No step “Start”, nada precisa de ser configurado. É o step que funciona como um marcador para início dos fluxos de qualquer Job, e permite também o agendamento utilizando o agendador do próprio Pentaho. Para nosso exercício, não será necessário realizar configurações para este step:



Detalhamento Step:

**Repeat** – quando marcado faz com que o job inicie automaticamente uma nova execução sempre que uma é concluída com sucesso, e não já uma data para encerramento.

**Type** – escolha sobre a forma com qual ao job será automatizado, em um intervalo de minutos ou segundos, diariamente, semanalmente ou mensalmente.



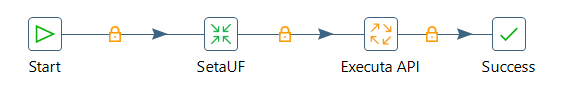
**Exercício 05 – Jobs HOPS Fluxos**

* 1. De forma similar aos HOPs de transformação, o Pentaho integra um step de job ao outro por meio de HOPs, com algumas possibilidades a mais

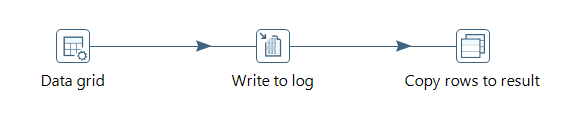
|  |  |
| --- | --- |
|  | HOP Verde: fluxo vai passar para o próximo step se e somente se o anterior der ok. |
|  | HOP Cadeado Amarelo: indica que o fluxo irá passar para o próximo step independente do estado do hop anterior. |
|  | O HOP vermelho permite que um step seja acionamento se e somente se o step anterior acusar erro. |

**Exercício 06 – Jobs Loops e Variáveis**

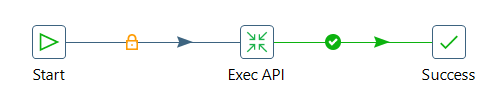
* 1. Este exercício consiste na utilização de Jobs e Transformações para realização de processos em loop, de forma similar ao funcionamento de cursores SQL, nos quais uma lista de parâmetros é repassada para o código do cursor, que executa toda sua lógica para cada um dos itens de entrada. Será composto por 2 Jobs e 2 Transformações, cada um com uma função específica. A seguir, uma visão geral destes 4 elementos:
     1. Job principal, “06 – Jobs API Loop”, a partir do qual o processo completo será instanciado:



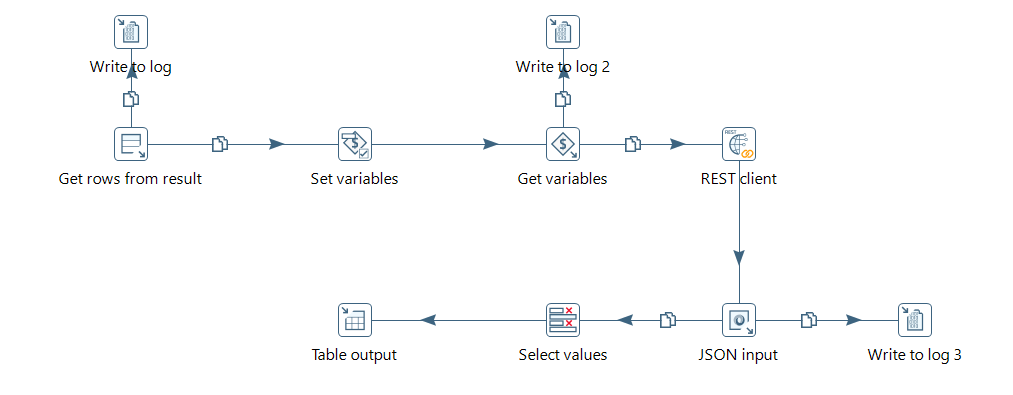
* + 1. A primeira transformação, “Seta UF”, irá disparar a transformação “06 – Jobs Seta UF”:



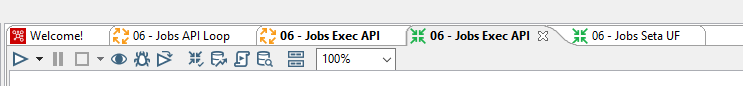
* + 1. O segundo Job, “Executa API”, irá disparar o Job “06 – Jobs Exec API”:



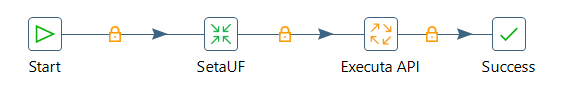
* + 1. Dentro do Job, “06 – Jobs Exec API”, será disparada a transformação “06 – Jobs Exec API”:



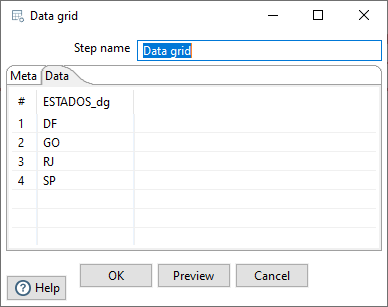
* + 1. Ao final, todos estes elementos deverão ser construídos e validados:



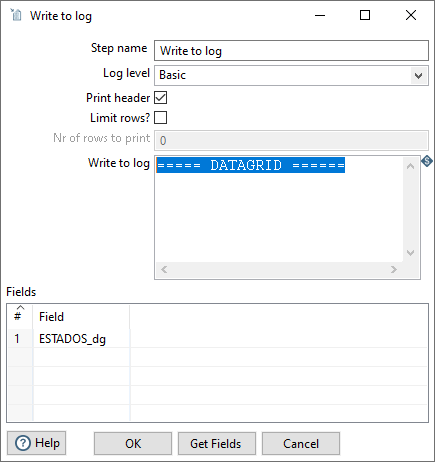
* 1. Crie um job chamado “06 – Jobs API Loop”, e adicione os steps “Start”, “Tranformation”, “Job” e “Success”.



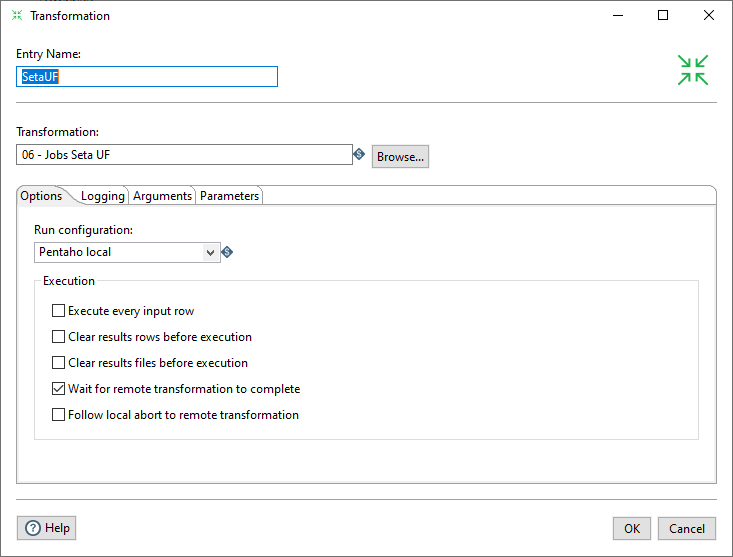
* 1. Os steps “Start” e “Success” não necessitam de configuração, a não ser que um agendamento seja feito, conforme descrito no item 4.3 deste documento. Salve o Job e passe para o próximo item do exercício.
  2. Crie uma transformação chamada “06 – Jobs Seta UF”, e adicione os steps “Start”, “Tranformation”, “Job” e “Success”. Crie uma lista estática com os seguintes valores:



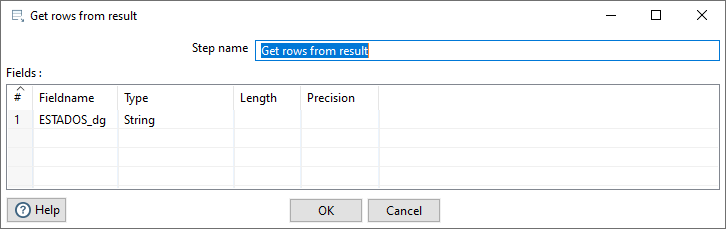
* 1. Adicione o step “Write to Log”, que irá permitir a visualização do log desta transformação, a partir do monitor de execução do Job principal “06 – Jobs API Loop”. Adicione o campo cujo valor se deseja imprimir em tela, e também um comentário na caixa “Write to log”, para facilitar a localização.



* 1. Adicione o step “Copy rows to result”, para levar o conteúdo da transformação para um outro elemento fora desta transformação. No caso, a transformação irá encerrar a partir do step “Copy rows to result” (que não precisa de configurações), e irá deixar em memória o seu conteúdo, que será configurado para recebimento em outra transformação. Salve esta transformação e passe para o próximo item deste exercício.
  2. Volte ao Job principal, e edite o step “Seta UF”, para apontá-lo para a transformação “06 – Jobs Seta UF”, recém criada. Salve a transformação e vá para o próximo passo do exercício:

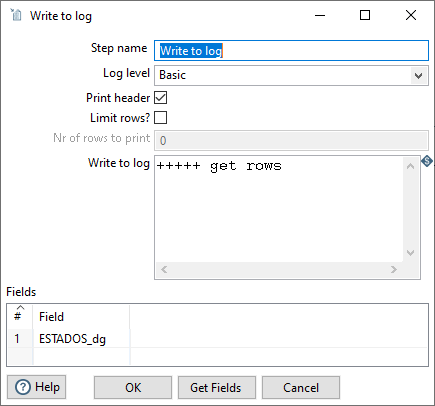


* 1. Crie um novo Job, e o salve como “06 – Jobs Exec API”, e adicione os campos obrigatórios “Start” e “Success”, além de um step de “Transformation”, e vá par ao próximo item do exercício:
  2. Crie uma nova transformação, chamada “06 – Jobs Exec API”, e adicione o step “Get rows from result”, para poder receber os dados do fluxo de execução de outra transformação, conforme definido no item 6.6:

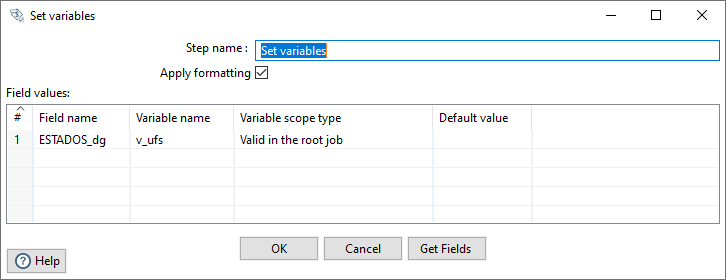


Obs.: O campo definido na seção “Fields”, coluna “Fieldname” deve ser inserido manualmente, e com o mesmo nome utilizado na configuração de saída do item 6.6 .

* 1. Adicione um step “Write to log”.



* 1. Adicione um step “Set variables”, para declarer a variável que irá armazenar a lista de execução. Na seção “Field values”, campo “Field name”, insira o nome definido no step anterior. Em “Variable name”, defina o nome da variável que será repassada à API. Em “Variable scope type”, definia quais níveis de execução irão conseguir consumir a variável, sendo que em nosso caso, iremos marcar “Valid in the root job”:



Obs.: Os escopos de variável que poderão ser escolhidos são:

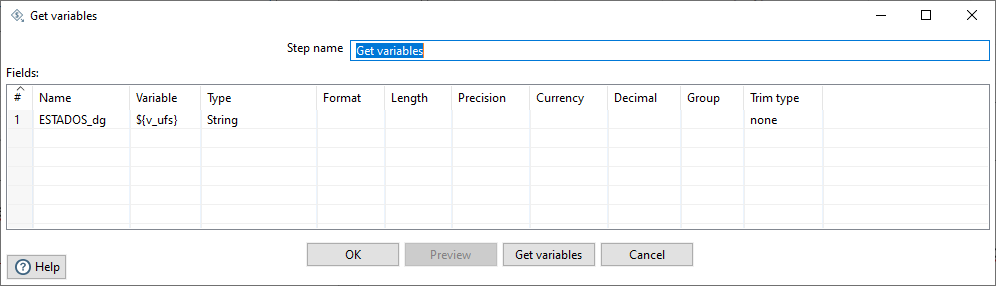
Valid in the root job – O job principal consome o conteúdo

Valid in the grand-parent job – O maior job da thread consome o conteúdo

Valid in the parent job – Apenas o job anterior consome o conteudo

Valid in the Java Virtual Machine – Conteúdo na memória do Java, para acesso fora o pentaho

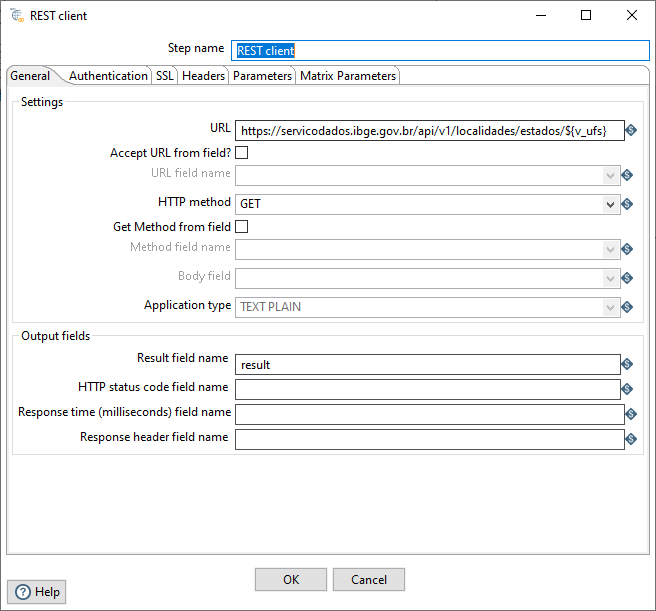
* 1. Adicione um step “Get variables”, para preencher o conteúdo da variável. Dê um nome ao campo que irá acomodar a variável, e preencha em “Variable” o nome da variável definida no step anterior, dentro da estrutura de ponteiros ${<nome da variável>}, sendo que no caso deste exercício, o valor a ser preenchido nesta coluna será **${v\_ufs}**:



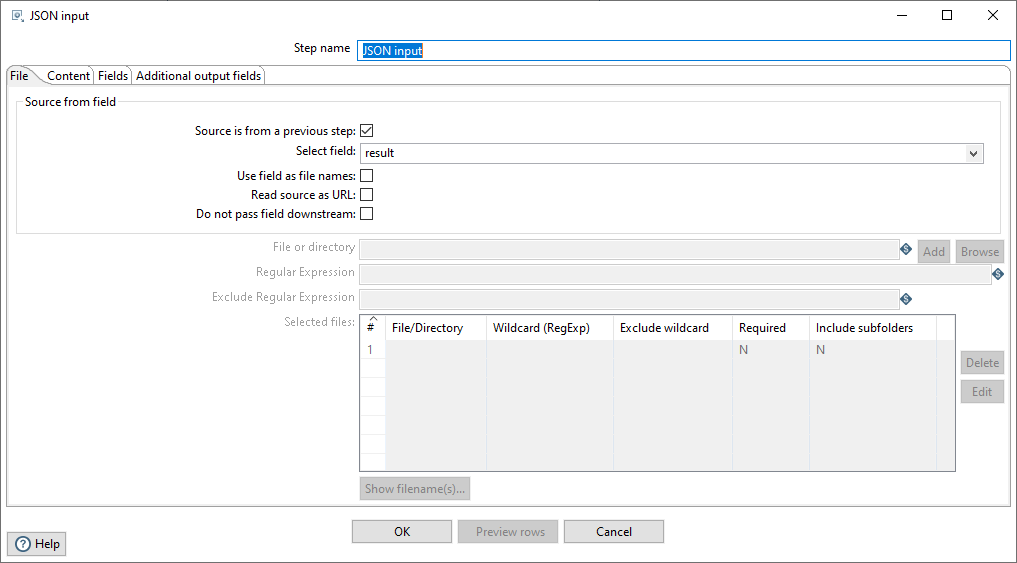
Obs.: Adicione também um step de gravação de log.

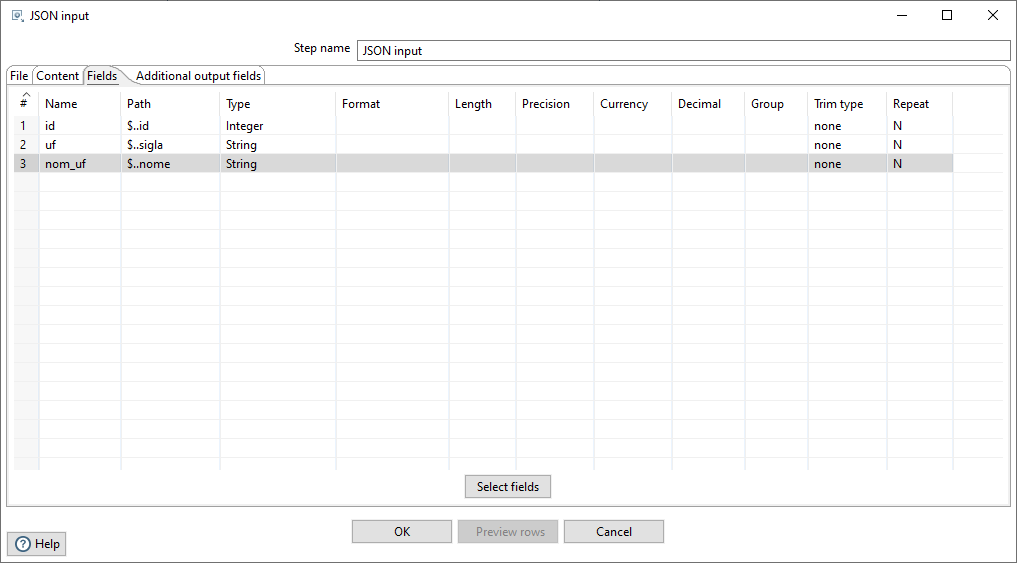
* 1. Adicione um step “REST Client”, e o configure para receber dados da API do IBGE ( [https://servicodados.ibge.gov.br/api/v1/localidades/estados/](https://servicodados.ibge.gov.br/api/v1/localidades/estados/$%7bv_ufs%7d) ), adicionando como parâmetro ao seu final, a variável criada anteriormente:

URL: [https://servicodados.ibge.gov.br/api/v1/localidades/estados/${v\_ufs}](https://servicodados.ibge.gov.br/api/v1/localidades/estados/$%7bv_ufs%7d)



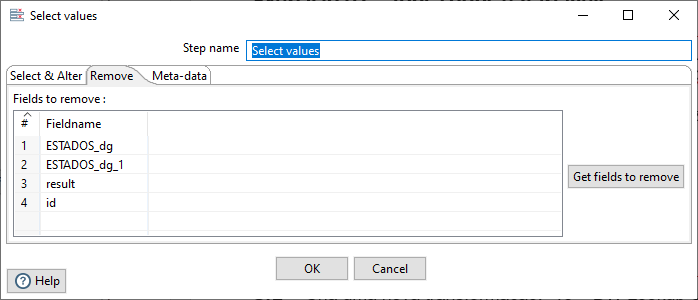
* 1. Adicione um step “JSON input”, e na aba “File”, marque a opção “Source is from a previous step” e indique o campo de origem dos dados. Já na aba “Fields”, configure manualmente os campos de entrada provenientes do resultado do REST:



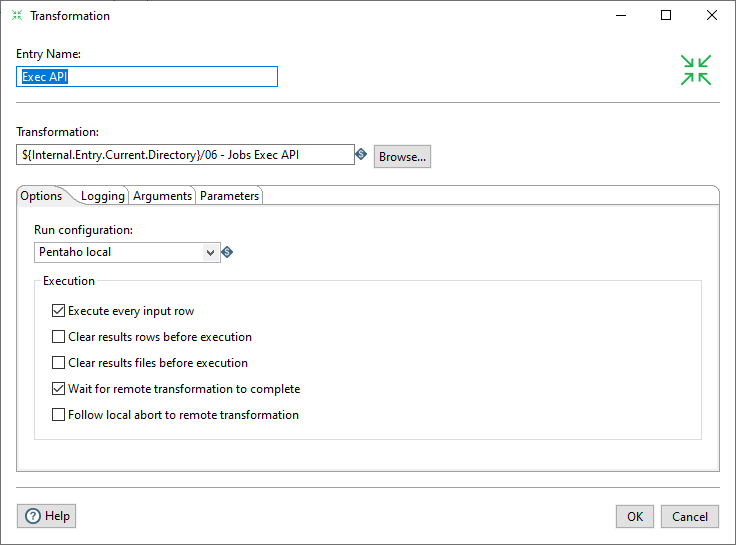


Obs.: Adicione também mais um step de gravação em log.

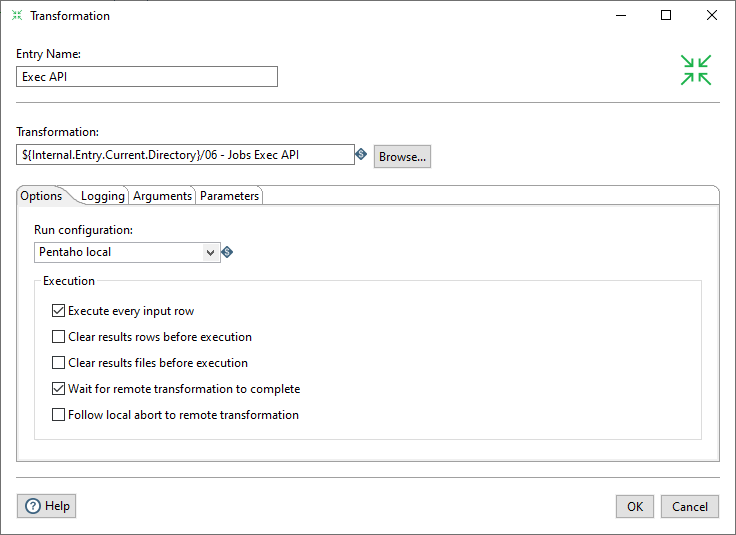
* 1. Selecione apenas o campos necessários na saída do JSON:



* 1. Grave o resultado em uma tabela chamada “api\_loop”, salve a transformação, e volte para o job “06 – Jobs Exec API”, configurando o step de job para transformação “Exec API” , apontando para invocar a transformação “06 – Jobs Exec API” recém criada. Na configuração, marque “Execute every input row”, para permitir o loop. Salve o Job.



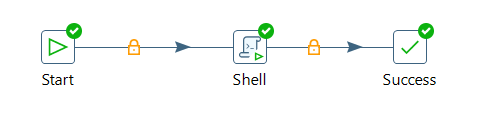
* 1. Grave o resultado em uma tabela chamada “api\_loop”, salve a transformação, e volte para o job “06 – Jobs Exec API”, configurando o step de job para transformação “Exec API” , apontando para invocar a transformação “06 – Jobs Exec API” recém criada. Na configuração, marque “Execute every input row”, para permitir o loop. Salve o Job.



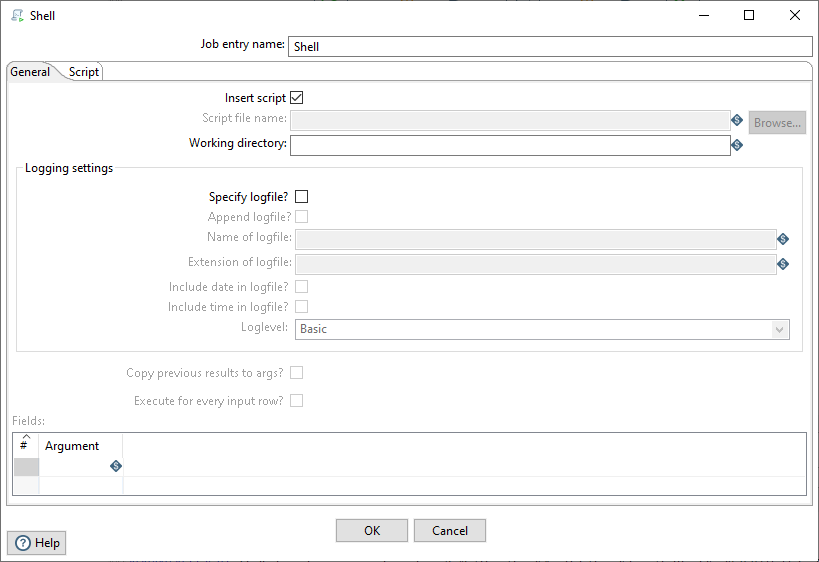
* 1. Retorne ao job principal “06 – Jobs API Loop” e o execute. Observe no log a passagem dos parâmetros, e verifique a carga da tabela.

**Exercício 07 – Jobs Shell**

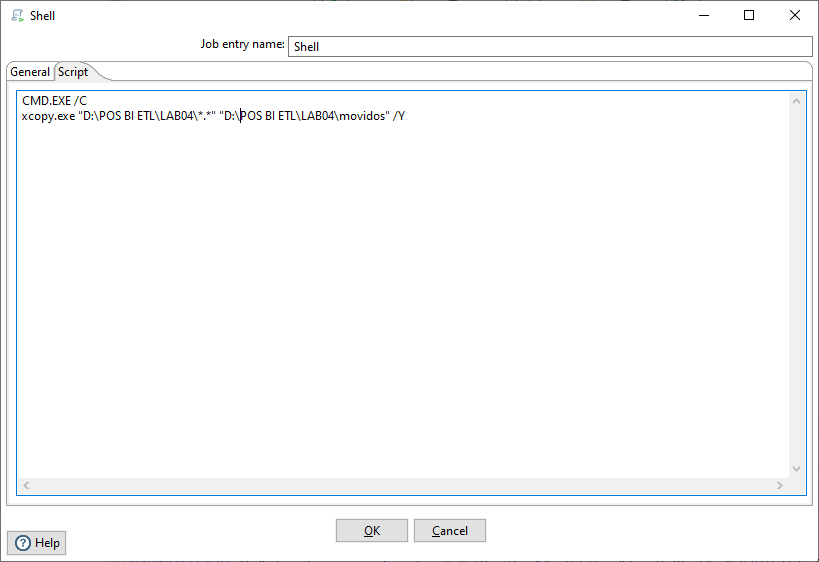
* 1. Cria uma nova transformação, “07 – Jobs Shell” , adicione os steps necessários ao Job, e também adicione o step “Shell”, de acordo com a figura a seguir:



* 1. Na configuração do step “Shell”, marque a opção “Insert script”, para permitir a escrita de um novo código na aba “Script”:



Configuração geral do script



Inserção do script