Estrutura que compõe uma Aplicação de Integração

Projeto

Os projetos são um meio de agrupar as integrações de uma forma lógica. Normalmente, os limites de um projeto devem ser iguais a uma unidade implantável. Isso significa que o escopo ou contexto do projeto devem ser aqueles fluxos que devem ser agrupados quando implantados. Ou seja, se dois fluxos dependem um do outro e são tão fortemente acoplados que mudanças em um ditam mudanças no outro, então esses fluxos devem ser agrupados em um único projeto. O projeto é o nível no qual os fluxos e seus componentes são conectados e controlados.

Versão

Os projetos podem ter várias versões. Quando um projeto é criado pela primeira vez, uma versão inicial também é criada com o nome master. Depois que o projeto mestre é lançado (implantado na produção), as alterações futuras no projeto podem ser concluídas em uma nova versão de forma que cada migração de produção do projeto possa ser rastreada. Ao tornar cada lançamento de produção de um projeto em uma nova versão do projeto, a visibilidade das diferenças entre as versões também está disponível. Novas versões de um projeto podem ser criadas a partir de qualquer versão subsequente. Ex: se você tiver as versões 1.0.0, 1.1.0 e 1.2.0, uma nova versão pode ser criada a partir de QUALQUER uma das 3 versões anteriores. Uma versão de lançamento de patch (1.1.1) pode ser criada a partir da versão 1.1.0 ao mesmo tempo em que são feitas alterações na versão 1.2.0 para alguma data de lançamento futura.

Fluxo de trabalho

Uma integração é chamada de fluxo. Um fluxo define a entrada a ser lida, as regras de processamento a serem aplicadas e a saída a ser gravada.

Fluxos são um conjunto gráfico de instruções sobre como os dados são integrados (fluxos) de um sistema de origem para um sistema de destino.

Componente

Os fluxos consistem em um conjunto de **componentes**(gatilhos, conectores, processadores , produtores, consumidores ou transformadores) conectados entre si por links de componentes.

Cada componente possui uma porta de entrada e uma porta de saída através das quais são realizados os links entre eles. Um componente é capaz de validar com quais componentes ele pode se conectar através do tipo de saída produzido pelo componente adjacente. Um componente pode receber várias conexões na mesma porta de entrada de componentes diferentes. Assim como na entrada ele também pode produzir conteúdo de saída para vários outros componentes.

Todo componente possuí um conjunto de metadados cuja finalidade é descrever todas as propriedades que o compõe e que são necessárias para sua criação e configuração. Estas propriedades podem ser valores que alteram seu comportamento ou suas características e também o tratamento de eventos que podem ocorrer durante o ciclo de vida do componente.

As propriedades dos metadados de um componente são usadas também em ferramenta de design visual para a criação de fluxos visuais através de métodos de arrastar e soltar.

Todo componente executa uma ação quando é acionado pelo gerenciador de fluxo de trabalho. Quando conclui a ação ele retorna os dados de saída(relatório) que são repassados para o(s) próximo(s) componentes do fluxo. Isso ocorre até que o fluxo execute todas as ações para todos os componentes.

Quando um fluxo é executado, todos os componentes do fluxo são inicializados em paralelo e ficam aguardando instrução para executarem suas ações. Qualquer componente **que não tenha link** de componente de entrada será comunicado para executar sua ação logo no inicio da execução do fluxo ou por algum evento/mensagem externa.

Tipos de componentes

Conector

> Os conectores são componentes que permitem acesso aos recursos(fontes de dados) que serão usados para a extração dos dados para transformação ou carregamento dos dados de saída que serão gerados na integração.

Coletor de dado

> Os coletores de dados realizam o processo de extração dos dados em sua forma bruta assim como estão armazenados. Para isto usam um conector para ter acesso ao recurso(fonte de dados) onde estão armazenados os dados.

Processador

> Durante o fluxo de trabalho pode ocorrer a necessidade de executarmos/processarmos rotinas através da execução de chamadas a outras aplicações externas, comandos dos sistema operacional, chamadas a funções/procedimentos de um banco de dados, etc.

Produtor

> Para que possa ser feita integração/notificação com outros sistemas externos ao fluxo de trabalho precisamos produzir eventos/mensagens para que esses sistemas possam ser avisados quer seja da conclusão do trabalho, da chegada de novas informações ou alterações realizadas em alguma fonte de dados.

Consumidor

> Assim como os produtores de eventos/mensagens externas são importantes ao fluxo de trabalho, poderemos precisar ouvir o mundo externo durante o processo de execução de algum fluxo de trabalho que seja executado de forma repetida ou num ciclo de vida indeterminado e consumir mensagens/eventos gerados por outras aplicações/sistemas.

Transformador

> Todos os dados coletados podem precisar serem transformados(manipulados) antes de seguirem em frente no fluxo ou antes de serem carregados/disponibilizados no seu destino final.

Gatilho

> Um componente de gatilho tem a função de ouvir eventos produzidos por diversas fontes de dados e disparar um ou mais fluxos de trabalho sempre que necessário. Ao alterar um dado no banco de dados por exemplo podemos querer executar um fluxo de trabalho que realize alguma ação.
> Ao recebermos um determinado tipo de e-mail poderemos querer disparar alguma ação no sistema.
> Os gatilhos são sempre carregados quando a aplicação é iniciada. Eles são a porta de entrada para os eventos/mensagens que disparam os fluxos de trabalho.

Vejamos alguns exemplo de gatilhos:

- Evento gerado por um ouvinte OUTBOX/CDC;
- Chegada de um e-mail;
- Chamada de um Webhook;
- Modificação de uma pasta ou arquivo;
- Agendamento;
- etc;

A tabela a seguir mostra uma lista dos componentes disponíveis:

Conectores

Conector	Descrição
JDBC	Permite acesso a maioria de bancos de dados relacionais do mercado.
MongoDB	Permite conexão com o banco de dados NoSQL MongoDB.
Redis	Permite conexão com o banco de dados NoSQL Redis.
Http Client	Permite a realização de conexões http em um servidor para a captura ou envio de dados
Ftp	Permite a conexão com servidores FTP para o download ou upload de arquivos.
RestAPI	Permite a conexão com Api's REST.

Coletores de dados

Coletor	Descrição
SQLDataCollector	Possibilita a execução de comandos em um banco SQL para obtenção dos dados.
FileReader	Possibita a leitura de arquivos para captura de dados.

Processadores

Processador	Descrição
Command SQL	Permite a execução de um comando SQL em um banco de dados.
Function SQL	Permite a execução de um função armazenada em um banco da de dados SQL
StoredProc SQL	Permite a execução de um procedimento armazenado em um banco de dados SQL
Shell Script	Possibilita a execução de um script do sistema operacional.

Produtores

Produtor	Descrição
Kafka Producer	Gera mensagens em um ou mais tópicos permitindo assim a integração com outros sistemas externos ou até mesmo disparar outros gatilhos que consomem mensagens em outros fluxos de trabalho.
MQTT Producer	Gera mensagens no protocolo MQTT que pode ser consumida por outras aplicações

** Consumidores **

Consumidor	Descrição
Kafka Consumer	Ouve tópicos para receber e processar novas mensagens de uma ou mais filas. Este componente pode ser usado em um fluxo de trabalho repetitivo/contínuo.
MQTT Consumer	Ouve tópicos de mensagens MQTT de um determinado Broker. Este componente pode ser usado em um fluxo de trabalho repetitivo/contínuo.

** Transformadores**

Transformador	Descrição
TransformToJson	Transforma um conteúdo recebido em alguns formatos estabelecidos para o formato **JSON**
TransformToXML	Transforma um conteúdo recebido em alguns formatos estabelecidos para o formato **XML**
TransformToCSV	Transforma um conteúdo recebido em alguns formatos estabelecidos para o formato **CSV**
TransformToFixedDelimited	Transforma um conteúdo recebido em alguns formatos estabelecidos para o formato **Fixed Delimited**

** Gatilhos **

Gatilho	Descrição
OutBox or CDC Trigger	Gatilho que dispara um ou mais fluxo(s) de trabalho baseado na alteração de algum dado/tabela no banco de dados.
Schedulling Trigger	Gatilho que dispara um ou mais fluxo(s) de trabalho basedo no agendamento de uma data hora, dia, hora, minutos ou segundos.
WebHook Trigger	Gatilho que dispara um ou mais fluxo(s) de trabalho sempre que receber um conexão HTTP.
MQTT Trigger	Gatilho que dispara um ou mais fluxo(s) de trabalho sempre que receber uma nova mensagem em um ou mais tópicos sendo ouvidos.
Kafka Message Trigger	Gatilho que dispara um ou mai(s) fluxo(s) de trabalho sempre que receber uma nova mensagem em um mais tópicos sendo ouvidos

Plugins

Um plugin é uma maneira de terceiros estenderem a funcionalidade de um aplicativo. Um plug-in implementa pontos de extensão declarados pelo aplicativo ou outros plug-ins. Além disso, um plugin pode definir pontos de extensão.

Toda a arquitetura da aplicação é baseada em componentes e composta de plugins visando torná-la extensível. Com isso é possível adicionar novas funcionalidades para resolver problemas diferentes encontrados em outros cenários de integração.

Para a criação de um plugin basta implementar um novo módulo Maven em linguagem java e iniciar a construção a partir de alguns dos pontos de extensão já disponbilizadas pela aplicação ou por plugins de terceiros.

Cada projeto terá disponível os plugins padrões da aplicação e mais os compartilhados por terceiros ou construídos pelo usuário.

Pontos de extensão

Para estender a funcionalidade de um aplicativo, ele deve definir um chamado ponto de extensão. Esta é uma interface ou classe abstrata, que define um comportamento específico para uma extensão.

Extensões

Uma extensão é uma implementação concreta de um ponto de extensão. Uma extensão pode gerar novos pontos de extensão.