Manejar variables como POJO

Recursos Relacionados

- Flujo de datos

Prerrequisitos

1. Ejecución de Zeebe Broker con punto final `localhost:26500` (predeterminado)
2. Ejecute el ejemplo Implementar un flujo de trabajo

HandleVariablesAsPojo.java

Fuente en github

```java
/*
 * Copyright Camunda Services GmbH and/or licensed to Camunda Services GmbH
under
 * one or more contributor license agreements. See the NOTICE file distributed
 * with this work for additional information regarding copyright ownership.
 * Licensed under the Zeebe Community License 1.0. You may not use this file
 * except in compliance with the Zeebe Community License 1.0.
 */
package io.zeebe.example.data;

import io.zeebe.client.ZeebeClient;
import io.zeebe.client.ZeebeClientBuilder;
import io.zeebe.client.api.response.ActivatedJob;
import io.zeebe.client.api.worker.JobClient;
import io.zeebe.client.api.worker.JobHandler;
import java.util.Scanner;

public class HandleVariablesAsPojo {
  public static void main(final String[] args) {
    final String broker = "127.0.0.1:26500";

    final ZeebeClientBuilder builder =

ZeebeClient.newClientBuilder().brokerContactPoint(broker).usePlaintext();

    try (ZeebeClient client = builder.build()) {
      final Order order = new Order();
      order.setOrderId(31243);

      client
          .newCreateInstanceCommand()
          .bpmnProcessId("demoProcess")
          .latestVersion()
          .variables(order)
          .send()
          .join();

      client.newWorker().jobType("foo").handler(new DemoJobHandler()).open();

      // run until System.in receives exit command
      waitUntilSystemInput("exit");
    }
  }

  private static void waitUntilSystemInput(final String exitCode) {
    try (Scanner scanner = new Scanner(System.in)) {
      while (scanner.hasNextLine()) {
        final String nextLine = scanner.nextLine();
        if (nextLine.contains(exitCode)) {
          return;
        }
      }
    }
  }

  public static class Order {
    private long orderId;
```

```
    private double totalPrice;

    public long getOrderId() {
      return orderId;
    }

    public void setOrderId(final long orderId) {
      this.orderId = orderId;
    }

    public double getTotalPrice() {
      return totalPrice;
    }

    public void setTotalPrice(final double totalPrice) {
      this.totalPrice = totalPrice;
    }
  }

  private static class DemoJobHandler implements JobHandler {
    @Override
    public void handle(final JobClient client, final ActivatedJob job) {
      // read the variables of the job
      final Order order = job.getVariablesAsType(Order.class);
      System.out.println("new job with orderId: " + order.getOrderId());

      // update the variables and complete the job
      order.setTotalPrice(46.50);

      client.newCompleteCommand(job.getKey()).variables(order).send();
    }
  }
}
```