Leadfeeder job interview assignment

Problem

European Central Bank provides daily exchange rates for converting US Dollars to Euros. The CSV file for up-to-date exchange rates can be downloaded from:

http://sdw.ecb.europa.eu/export.do?type=&trans=N&node=2018794&CURRENCY=USD&FREQ=D&start=01-01-2012&q=&submitOptions.y=6&submitOptions.x=51&sfl1=4&end=&SERIES_KEY=120.EXR.D.USD.EUR.SP00.A&sfl3=4&DATASET=0&exportType=csv

The file looks roughly like below. (The first number here denotes line number, not in actual file).

```
   1 Dataset name: Exchange Rates; Frequency: Daily; Currency: US dollar;
Currency denominator: Euro; Exchange rate type: Spot; Series variation - EXR
context: Average or standardised measure for given frequency
   2 ,EXR.D.USD.EUR.SP00.A
   3 ,"ECB reference exchange rate, US dollar/Euro, 2:15 pm (C.E.T.)"
   4 Collection:,Average of observations through period (A)
   5 Period\Unit:,[US dollar ]
   6 2015-04-01,1.0755
   7 2015-03-31,1.0759
   8 2015-03-30,1.0845
   9 2015-03-27,1.0856
  10 2015-03-26,1.0973
  11 2015-03-25,1.0985
  12 2015-03-24,1.0950
  13 2015-03-23,1.0912
```

Build an utility that can be used to convert an USD value to Euros on any data since year 2000.

To do this you should:

1. Have the application download, parse and store the exchange rates to a database
2. Have a class to handle conversion of a USD value - date pair to Euro value.
   E.g.
   `ExchangeRateConverter.convert(120, '2011-03-05')`

   Should return what 120 USD was in euros on March 5, 2011.

Note that the ECB file only includes days when they had agreed on the exchange rates - these are typically non-holiday weekdays. To convert values on weekends and holidays you should use the previous available exchange rate.

Instructions & requirements

- Build the assignment in Ruby. You can use Rails, but it is not required.
- You can use any database to store the exchange rates. Choose something sensible and suitable.
- Include an easy way to download and update the latest values. For example a rake task.
- Make the updating procedure idempotent. Eg. such that it can be ran multiple times without it being destructive or adding duplicate records.
- Write tests preferably with Rspec.

Evaluation

What we will look from your completed assignment:

- Correctness of results. I.e. does it work right.
- Clarity and understandability of the code.
- Use of best practises, DRY
- Quality of the tests.
- Performance of the exchange rate lookup.
- Chosen database.
- Error handling.
- Completion time.

Returning

Send your complete app with any setup and use instructions as a zip or tar archive to herkko.kiljunen@leadfeeder.com